МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

Национальный исследовательский ядерный
университет «МИФИ»

ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ

Направление подготовки: 01.03.02 Прикладная математика и информатика

ПРОЕКТНАЯ ПРАКТИКА

Пояснительная записка к проекту

«Математические методы в робототехнике»

Выполнили:

Чистый А.С., Б19-511

Хохлов Н.М., Б19-501

Молчанов Е.М., Б19-501

Научный руководитель:

Мисюрин Сергей Юрьевич

Москва, 8 июня 2020 года

ОГЛАВЛЕНИЕ

Введение 3

Аналитическая часть 4

Теоретическая часть 6

Походки 7

Практическая часть 9

Экспериментальная часть 11

Заключение и выводы 13

Список литературы 14

Приложение А 15

Приложение Б 16

ВВЕДЕНИЕ

Технологический прогресс не стоит на месте. Всё сильней технологии вливаются в нашу жизнь. Если сейчас большинство роботов используемых в быту реализованы в виде программ ("Привет, Алиса"), то в недалёком будущем физические роботы (со сложной механической структурой) станут повседневной жизнью.

Физический робот представляет собой пространственный механизм (как правило с несколькими степенями свободы), который способен выполнять автономные или предварительно запрограммированные задачи. В качестве движущей силы используются электро-, гидро- или пневмоприводы. Роботы могут работать как под управлением независимой программы, так и выполнять команды напрямую от оператора.

У производственных или медицинских роботов чаще всего точность - это основной показатель качества, для других видов роботов высокая точность движения как правило не нужна; поэтому в подобные роботы не устанавливаются дорогостоящие компоненты, а алгоритмы, управляющие движениями таких роботов, допускают погрешность.

Есть ли простой способ усовершенствовать физические качества робота (например, точность позиционирования), оставив цену в приемлемом диапазоне? Модернизация движения робота с помощью программных средств - основная задача проекта.

Для исследований, а также для тестирования создаваемых алгоритмов был приобретён шестиногий робот-паук (англ. *hexapod*) *Hiwonder SpiderPi,* это робот низкой ценовой категории, а значит отлично подходит для целей проекта. Важно отметить, что в процессе работы над проектом один из сервоприводов робота сгорел, таким образом, были демонтированы две ноги (для симметрии) и робот стал четырёхногий (англ. *quadruped*). На поставленной задаче проекта это отразилось незначительно.

АНАЛИТИЧЕСКАЯ ЧАСТЬ

Для того, чтобы подробнее разобраться в теме проекта и изучить связанные с ней существующие математические модели был сделан обзор литературы. В работе [1] вводятся важнейшие термины для подобного типа робота, рассматриваются базовые вещи, такие как походка (англ. *gait*), взаимодействие компонентов робота друг с другом. Сравниваются характеристики различных походок.

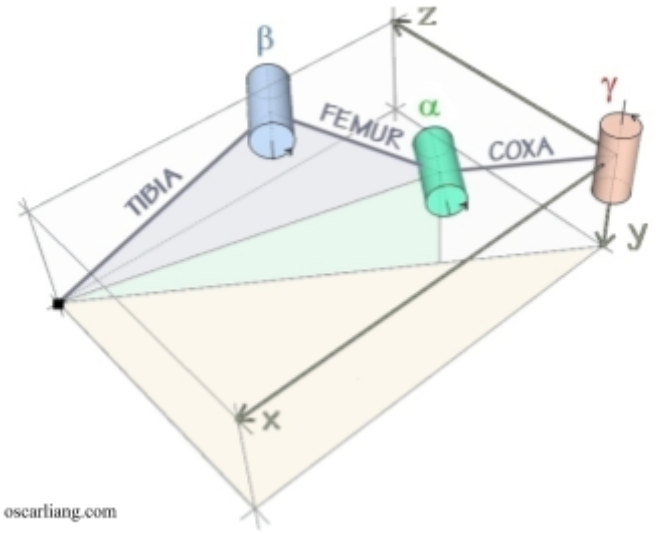


рис. 1

Coxa, femur и *tibia* - эти звенья представлены на рис. 1. Соединения звеньев (*α, β, γ* на рисунке) называются кинематическими парами. Эти же звенья изображены на роботе *SpiderPi* (рис. 2).

В работе [2] рассматривается прямая и обратная задача кинематики робота. Также затрагиваются общие идеи и математические преобразования для решения таких задач.

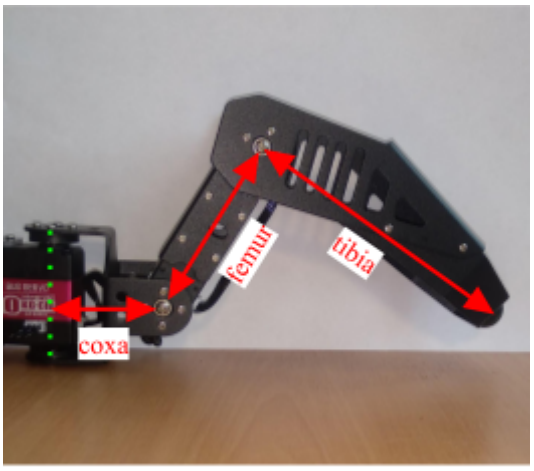


рис. 2

Работа [3] описывает примитивную походку четырёхногого робота. Рассматриваются условия при которых данная походка устойчива, детали перемещения ноги в пространстве не рассматриваются.

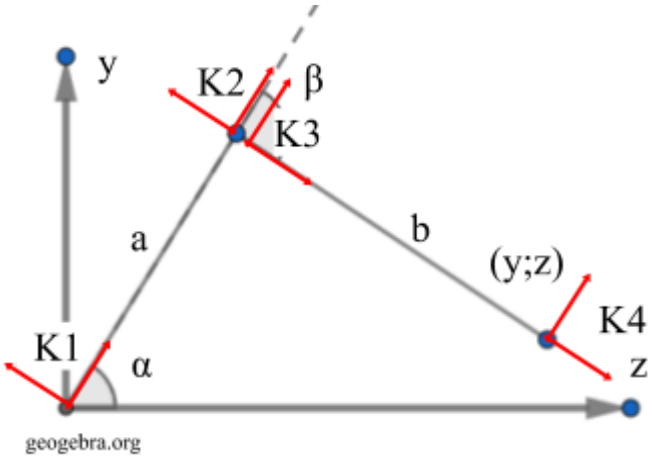
В работе [4] применялись математические техники для описания

походки шагающего двухногого робота, которые были применены нами для решения прямой задачи кинематики.

Во всех работах, где упоминается кинематическая задача, не рассматривается то, что сервоприводы робота могут иметь низкую точность установки положения. Возможно, решение обратной задачи для таких роботов не оправдано.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Как уже говорилось ранее, существует прямая и обратная задача кинематики. Грубо говоря, прямая задача - это преобразование текущего состояния кинематических пар (углов) в координаты, т.е. по заданным значениям углов определить положение выходного звена (ноги робота). Для подобных механизмов прямая задача всегда решается в явном виде. Чтобы упростить вычисления используются элементы линейной алгебры, например, матрицы перехода. Ниже приведён пример решения прямой



$$\begin{pmatrix} z \\ y \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} z_1 \\ y_1 \end{pmatrix}$$

$$\begin{pmatrix} z_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} z_2 \\ y_2 \end{pmatrix} + \begin{pmatrix} a \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} z_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} z_3 \\ y_3 \end{pmatrix}$$

$$\begin{pmatrix} z_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} z_4 \\ y_4 \end{pmatrix} + \begin{pmatrix} b \\ 0 \end{pmatrix}$$

рис. 3

задачи последовательными преобразованиями координат для ноги робота (рис. 3).

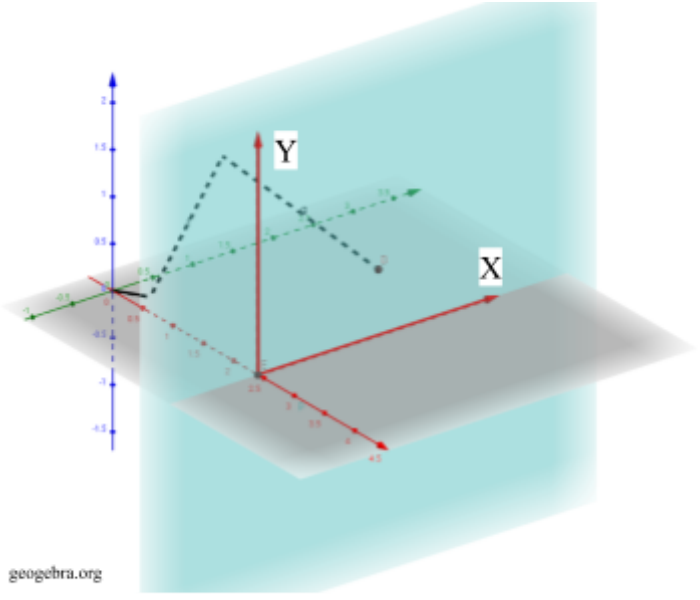


рис. 4

В примере в отличие от реальной модели не учитывается звено *coxa,* а также его поворот относительно вертикальной оси.

Нами была решена как прямая так и обратная задача, получены зависимости углов поворота от координат ноги. Для применения модели к

реальному роботу необходимо преобразовать углы в позиции сервоприводов.

Полученные функции углов размещены в Приложении А. Аргументами этих функций являются координаты ноги, *x,y* - как показано на рис. 4, *z* - расстояние от крепления *coxa* к роботу до плоскости движения

Походки

С помощью созданного математического аппарата были реализованы две походки робота паука. Первая - *Creep Gait* - простейшая походка

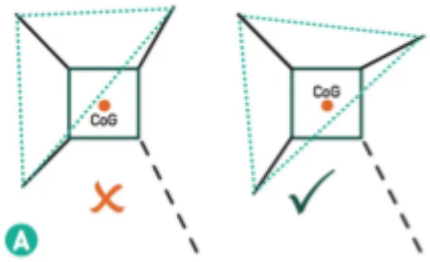


рис. 5

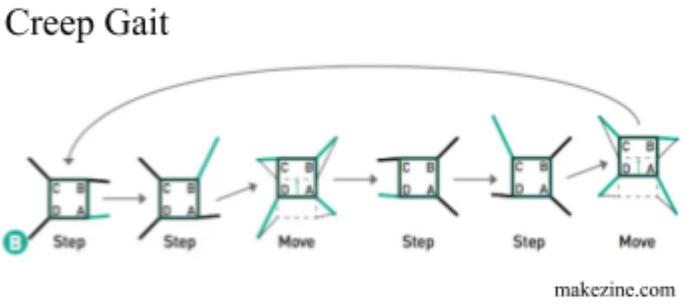


рис. 6

робота. Она характеризуется устойчивостью, ведь в каждый момент времени три опорные ноги паука образуют треугольник таким образом, что центр масс робота расположен внутри него (рис. 5). Алгоритм движения робота изображён на (рис. 6). Однако эта походка малоэффективна, как показано в работе [2].

Trot Gait. Один цикл движения изображен на (рис. 7). По сравнению с первой походкой, в этой во время фазы шага используются две ноги, что делает походку быстрой, но менее устойчивой.

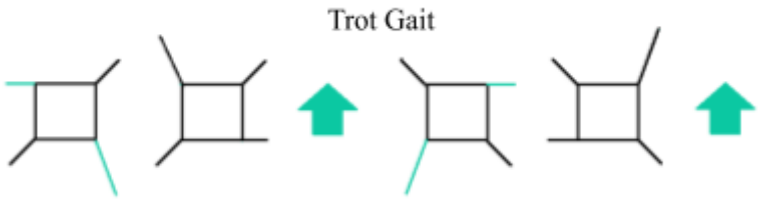


рис. 7

Чтобы повысить устойчивость данной походки, была проведена её модификация. Так в новой походке в фазе шага передняя опорная нога немного смещена назад, чтобы центр масс оказался на линии, соединяющей опорные точки (рис. 8). Расстояние, на которое смещена передняя нога, подбиралось вручную, за счёт чего “заваливание” робота на фазе шага было значительно уменьшено.

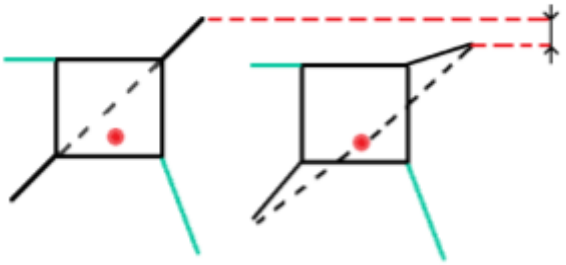


рис. 8

ПРАКТИЧЕСКАЯ ЧАСТЬ

Помимо относительно простой математической модели, был создан достаточно простой интерфейс для создания походок робота.

Будем называть функцию, описывающую траекторию движения, *траекторной функцией.* Траекторная функция принимает параметр, нормированный диапазоном [0;1] и возвращает точку в части двухмерного пространства нестрого ограниченного координатами (0,0) и (1,1). Была необходимость в двух траекторных функциях - одна для перемещения ноги в фазе шага, вторая - для перемещения в фазе опоры.

$$f(t) = \left(\frac{1 - \cos \pi t}{2}, \sin \pi t \right)$$

$$g(t) = (1 - t, 0)$$

Первая функция *f(t)* сделана так, что она описывает полуэллипс в пространстве. Вторая *g(t)* описывает линию лежащую на оси *OX.* Если “соединить” эти две функции, получим замкнутую кривую, по которой и будет двигаться нога.

Конечно, в реальности невозможно точно описать эту траекторию, потому что интерфейс сервопривода для движения принимает только конечное положение и интервал времени, за который необходимо

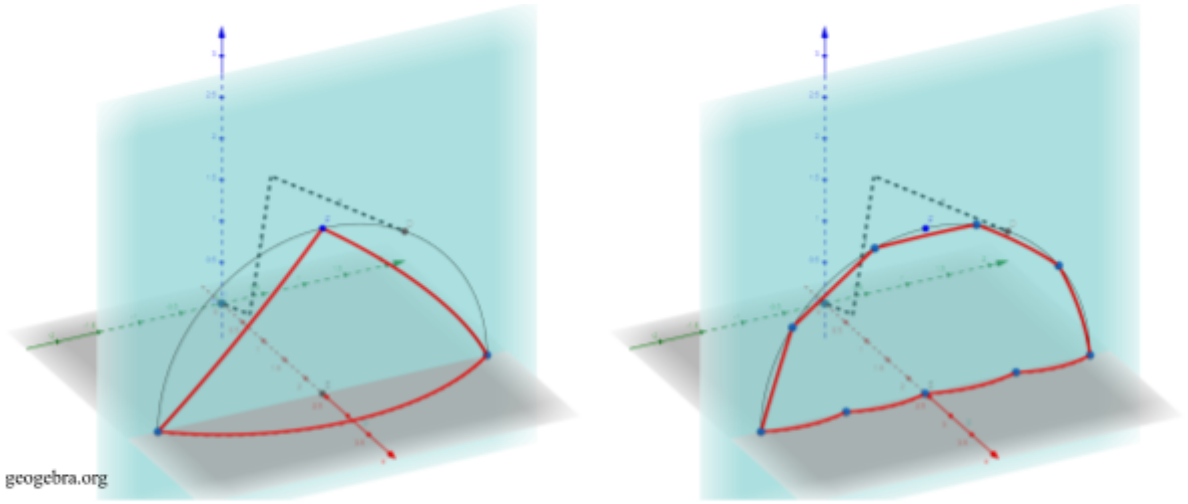


рис. 9

переместиться в конечное положение. Таким образом, траекторию

необходимо разбить на некоторое количество частей (рис. 9), чем больше этих частей - тем точнее траектория. Точки, которыми траектория была разбита на несколько частей, назовём *точками синхронизации.* Например, на рис. 9 слева в фазе шага три точки синхронизации.

ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

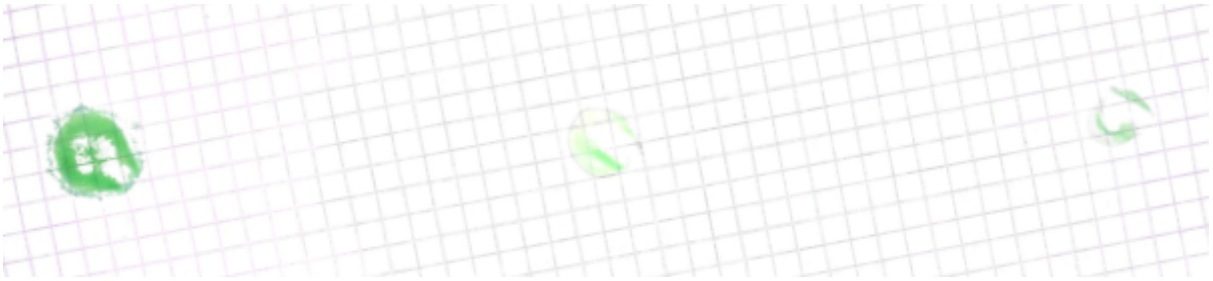
Если походка робота “неаккуратная”, то его ноги будут скользить по поверхности в процессе передвижения. Вследствие этого увеличивается изнашиваемость концов ног робота. На ногу робота наносилась краска, после чего совершался шаг. По следам, оставленным роботом на листах бумаги, можно сделать выводы об аккуратности походки.

Размером следа считалось максимальное расстояние между любыми двумя точками этого следа.

1. Стандартная походка - походка, которая была заложена сделана производителем. Невооружённым глазом видно, что следы достаточно крупные, имеются длинные участки, где нога скользит по поверхности. Размер следа (правого) составил 40 мм.

2. *Creep Gait,* в фазе опоры которого две точки синхронизации (по сути рис. 9, слева). След кажется меньше того, что в первом случае, на самом деле его размер 46 мм.

3. *Creep Gait* с тремя точками синхронизации в фазе опоры. Размер следа - 19 мм. Видимо, люфта сервоприводов достаточно, чтоб размер следа был таким небольшим уже при трёх точках синхронизации.



ЗАКЛЮЧЕНИЕ И ВЫВОДЫ

В работе было показано, что изменив алгоритм управления (основанный на кинематической модели механизма), можно значительно улучшить походку роботов-шагоходов. Это повышает ресурс работы механизма (уменьшает износ).

Была разработана удобная математическая модель для применения её на практике и программный интерфейс, с помощью которого пользователь может управлять роботом, а также создавать различные походки.

Была модернизирована походка *Trot Gait,* причем без использования динамического балансирования, что сказывается на экономии программного времени и электрозатрат.

В процессе тестирования было выяснено, что при использовании разработанных методов, скольжение робота о поверхность сильно уменьшается по сравнению со стандартными. Точность походки остаётся небольшой, но благодаря дешевизне компонентов и люфту в сервоприводах скольжения ноги почти не происходит.

СПИСОК ЛИТЕРАТУРЫ

1. Karakurt T., Durdu A., Yilmaz N. Design of six legged spider robot and evolving walking algorithms //International Journal of Machine Learning and Computing. – 2015. – Т. 5. – №. 2. – С. 96.

2. Xu K., Zi P., Ding X. Gait Analysis of Quadruped Robot Using the Equivalent Mechanism Concept Based on Metamorphosis //Chinese Journal of Mechanical Engineering. – 2019. – Т. 32. – №. 1. – С. 8.

3. -"How to Program a Quadruped Robot with Arduino" (https://makezine.com/2016/11/22/robot-quadruped-arduino-program/)

4. Алексеев Р. А., Мирошник И. В. Алгоритмы управления движением шагающего робота //Научно-технический вестник информационных технологий, механики и оптики. – 2005. – №. 19.

5. Аустен Я., Формальский А. М., Шевалльро К. Виртуальный четырёхногий робот: конструкция, управление, моделирование, эксперименты //Фундаментальная и прикладная математика. – 2005. – Т. 11. – №. 8. – С. 5-28.

6. Исследования молодых ученых : I Междунар. науч. конф. (г. Казань, июнь 2019 г.) / [под ред. И. Г. Ахметова и др.]. — Казань : Молодой ученый, 2019. — iv, 52 с.

ПРИЛОЖЕНИЕ А

Формулы преобразования координат ноги паука (конечной её точки) в углы поворота

$$\alpha = \frac{\pi}{2} - \arcsin \frac{(x - A \cdot \sin\arctan\frac{x}{z})^2 + (z - A \cdot \cos\arctan\frac{x}{z})^2 + y^2 + B^2 - C^2}{2 \cdot B \cdot \sqrt{(x - A \cdot \sin\arctan\frac{x}{z})^2 + (z - A \cdot \cos\arctan\frac{x}{z})^2 + y^2}}$$
$$\pm \arccos \frac{\sqrt{(x - A \cdot \sin\arctan\frac{x}{z})^2 + (z - A \cdot \cos\arctan\frac{x}{z})^2}}{\sqrt{(x - A \cdot \sin\arctan\frac{x}{z})^2 + (z - A \cdot \cos\arctan\frac{x}{z})^2 + y^2)}}$$
$$\beta = \arcsin \frac{(x - A \cdot \sin\arctan\frac{x}{z})^2 + (z - A \cdot \cos\arctan\frac{x}{z})^2 + y^2 + C^2 - B^2}{2 \cdot C \cdot \sqrt{(x - A \cdot \sin\arctan\frac{x}{z})^2 + (z - A \cdot \cos\arctan\frac{x}{z})^2 + y^2}}$$
$$- \arccos \frac{y}{\sqrt{(x - A \cdot \sin\arctan\frac{x}{z})^2 + (z - A \cdot \cos\arctan\frac{x}{z})^2 + y^2}}$$
$$\gamma = \arctan\frac{x}{z}$$

где *z* - расстояние от крепления *coxa* к роботу до плоскости движения, *x, y* - координаты в плоскости движения, *A* - длина *femur, B - tibia, C - coxa*

ПРИЛОЖЕНИЕ Б

Программный код разработанного интерфейса и походок
(GitHub - https://github.com/arkdchst/spiderpi)

```
#!/usr/bin/env python3

from Serial_Servo_Running import serial_setServo as set_servo

from math import *

from time import sleep


#длины звеньев

coxa = 43

femur = 75

tibia = 138



num_to_ids = {0:(1,2,3), 2:(7,8,9), 3:(10,11,12), 5:(16,17,18)} #номер ноги
-> номера приводов

middle_pos = {1: 300, 2: 461, 3:689, 7: 678, 8: 453, 9: 692, 10: 681, 11:
548, 12: 316, 16: 300, 17: 546, 18: 310}#значния средних положений для
приводов

half_pi_pos = {1: -85, 2: 86, 3: 1071, 7: 315, 8: 87, 9: 1064, 10: 1048,
11: 922, 12: -63, 16: 689, 17: 920, 18: -62}#положения pi/2



def angle_to_pos(angle, id):#принимает угол в радианах, возвращает позицию
привода в диапазоне от 0 до 1000

      pos = (angle / (pi / 2)) * (half_pi_pos[id] - middle_pos[id]) +
middle_pos[id]

      return pos
```

```
def get_angles(x, y, z):#принимает координаты точки в пространстве,
возвращает углы поворота трёх приводов

      coxa_angle = atan(x / z)

      if y >= 0:

            femur_angle =
(pi)/(2)-asin(((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(coxa_angle))**2+y**2
+femur**2-tibia**2)/(2*femur*sqrt((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(c
oxa_angle))**2+y**2)))+acos((sqrt((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(c
oxa_angle))**2))/(sqrt((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(coxa_angle))
**2+y**2)))

      else:

            femur_angle =
(pi)/(2)-asin(((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(coxa_angle))**2+y**2
+femur**2-tibia**2)/(2*femur*sqrt((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(c
oxa_angle))**2+y**2)))-acos((sqrt((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(c
oxa_angle))**2))/(sqrt((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(coxa_angle))
**2+y**2)))

      tibia_angle =
asin(((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(coxa_angle))**2+y**2+tibia**2
-femur**2)/(2*tibia*sqrt((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(coxa_angle
))**2+y**2)))-acos((y)/(sqrt((x-coxa*sin(coxa_angle))**2+(z-coxa*cos(coxa_a
ngle))**2+y**2))) - femur_angle

      return (coxa_angle, femur_angle, tibia_angle)




def step_fun(t):#траекторная функция фазы шага, 0<=t<=1

      return ((1 - cos(pi * t)) / 2, sin(pi * t))


def back_fun(t):#траекторная функция опоры

      return (1 - t, 0)


'''

            front

09 08 07 -|2 cam 5|- 16 17 18
```

```
06 05 04  |1     4|  13 14 15
03 02 01 -|0 rpi 3|- 10 11 12
^                 ^
|                 |
id приводов       num - номера ног
'''


class Leg:
	num = None
	xmin = None
	ymin = None
	xmax = None
	ymax = None
	z = None
	def __init__(self, num, xmin=-50, ymin=-100, xmax=50, ymax=-50, z =
coxa + femur):
		self.num = num
		self.xmin = xmin
		self.ymin = ymin
		self.xmax = xmax
		self.ymax = ymax
		self.z = z
	def set_point(self, point, z, time):#перевести ногу в точку point
		time *= 1000
		angles = get_angles(point[0], point[1], z)
		for i in range(3):#выставление нужного положение трёх приводов
			pos = angle_to_pos(angles[i], num_to_ids[self.num][i])
			set_servo(num_to_ids[self.num][i], int(pos), int(time))
	def set_point_norm(self, point, time):#то же что и set_point, но
координаты нормируются диапазоном [0;1]
```

```
            point = (point[0] * (self.xmax - self.xmin) + self.xmin,
point[1] * (self.ymax - self.ymin) + self.ymin)#разнормировка

            self.set_point(point, self.z, time)




class Move:#класс движения

      leg = None

      from_x = None

      to_x = None

      onAir = None#если True, то движение - шаг, иначе - движение назад по
поверхности

      height = None#высота перемещения

      dt = None

      interval = None

      ready = False

      t = 0#состояние движения

      def __init__(self, leg, from_x, to_x, onAir, height, dt, interval,
aSync = False):

            self.leg = leg

            self.from_x = from_x

            self.to_x = to_x

            self.onAir = onAir

            self.height = height#по умолчанию 1

            self.dt = dt

            self.interval = interval

      def tick(self):#тик - увеличение t на dt и элементарное перемещение
ноги

            if self.ready: return

            self.t += self.dt

            if self.t > 1: self.t = 1
```

```
            if self.onAir:
                  point = step_fun(self.t)
            else:
                  point = back_fun(self.t)
            point = (point[0] * (self.to_x - self.from_x) + self.from_x,
point[1] * self.height)
            self.leg.set_point_norm(point, self.interval)
            if self.t == 1: self.ready = True




def move1():#классическая походка из Интернета
      leg0 = Leg(0, -100, -100, 0, -25)
      leg2 = Leg(2, 0, -100, 100, -25)
      leg3 = Leg(3, -100, -100, 0, -25)
      leg5 = Leg(5, 0, -100, 100, -25)

      t1 = 0.1#время одного тика в секундах
      dt = 0.1#увеличиваем t на dt за время t1

      while True:
            move=Move(leg0, 0, 1, True, 1, dt, t1)
            while not move.ready: move.tick(); sleep(t1)#пока не дошли до
конца - продолжаем движение
            move=Move(leg2, 0, 1, True, 1, dt, t1)
            while not move.ready: move.tick(); sleep(t1)
            moves=[Move(leg0, 0.5, 1, False, 1, dt, t1), Move(leg2, 0.5, 1,
False, 1, dt, t1), Move(leg3, 0, 0.5, False, 1, dt, t1), Move(leg5, 0, 0.5,
False, 1, dt, t1)]
            while not moves[0].ready:
                  for x in moves:
```

```
                        x.tick()
                  sleep(t1)
            move=Move(leg3, 0, 1, True, 1, dt, t1)
            while not move.ready: move.tick(); sleep(t1)
            move=Move(leg5, 0, 1, True, 1, dt, t1)
            while not move.ready: move.tick(); sleep(t1)
            moves=[Move(leg0, 0, 0.5, False, 1, dt, t1), Move(leg2, 0, 0.5,
False, 1, dt, t1), Move(leg3, 0.5, 1, False, 1, dt, t1), Move(leg5, 0.5, 1,
False, 1, dt, t1)]
            while not moves[0].ready:
                  for x in moves:
                        x.tick()
                  sleep(t1)



def move2():#походка на диагональных ногах
      a1 = 30#калибровка для уменьшения заваливания
      a2 = 12
      leg0 = Leg(0, -100, -100, 0, -25)
      leg2 = Leg(2, -a1, -100, 100-a1, -25)
      leg3 = Leg(3, -100, -100, 0, -25)
      leg5 = Leg(5, -a2, -100, 100-a2, -25)


      t1 = 0.4
      dt = 0.3
      t2 = 0.3
      dt2 = 0.4


      while True:
            move1=Move(leg2, 0, 1, True, 1, dt, t1)
```

```
            move2=Move(leg3, 0, 1, True, 1, dt, t1)

            while not move1.ready: move1.tick(); move2.tick(); sleep(t1)

            moves=[Move(leg0, 0, 0.5, False, 1, dt2, t2), Move(leg2, 0.5,
1, False, 1, dt2, t2), Move(leg3, 0.5, 1, False, 1, dt2, t2), Move(leg5, 0,
0.5, False, 1, dt2, t2)]

            while not moves[0].ready:

                  for x in moves:

                        x.tick()

                  sleep(t2)

            move1=Move(leg0, 0, 1, True, 1, dt, t1)

            move2=Move(leg5, 0, 1, True, 1, dt, t1)

            while not move1.ready: move1.tick(); move2.tick(); sleep(t1)

            moves=[Move(leg0, 0.5, 1, False, 1, dt2, t2), Move(leg2, 0,
0.5, False, 1, dt2, t2), Move(leg3, 0, 0.5, False, 1, dt2, t2), Move(leg5,
0.5, 1, False, 1, dt2, t2)]

            while not moves[0].ready:

                  for x in moves:

                        x.tick()

                  sleep(t2)
```