Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет систем управления и робототехники

Отчет по лабораторной работе №1

«Движение робота к точке с заданными координатами»

по дисциплине «Введение в профессиональную деятельность»

Выполнили: студенты гр. R3135
Дупак А. А.,
Щтенников Р. А.,
Зорькина А. А.

Преподаватель: Перегудин А. А.

Санкт-Петербург
2018

Цель работы

Получить опыт построения математической модели робота, освоить алгоритм движения робота с дифференциальным приводом к точке с заданными координатами.

Материалы работы

Описание

Рассмотрим конструкцию робота с дифференциальным приводом собранного на базе конструктора Lego Mindstorms EV3. Робот представляет собой двухколесную платформу с гироскопом и парой ультразвуковых дальномеров. Моторы имеют встроенные энкодеры способные определять угол поворота выходного вала. Гироскоп нужен для определения угловой скорости, хотя в последствии только одного датчика будет недостаточно для решения этой задачи. Ультразвуковыми дальномерами воспользуемся для определения расстояния до препятствий.

Такая кинематическая схема имеет следующее математическое описание:

$$\begin{cases} \dot{x} = \cos\psi \cdot \frac{\omega_1+\omega_2}{2} R, \\ \dot{y} = \sin\psi \cdot \frac{\omega_1+\omega_2}{2} R, \\ \dot{\psi} = (\omega_1 - \omega_2)\frac{R}{B}, \end{cases}$$

где ω_1, ω_2 — соответствующие угловые скорости вращения колес, x, y — соответствующие координаты, R — радиус колеса, B — расстояние между колесами, ψ — курс робота.

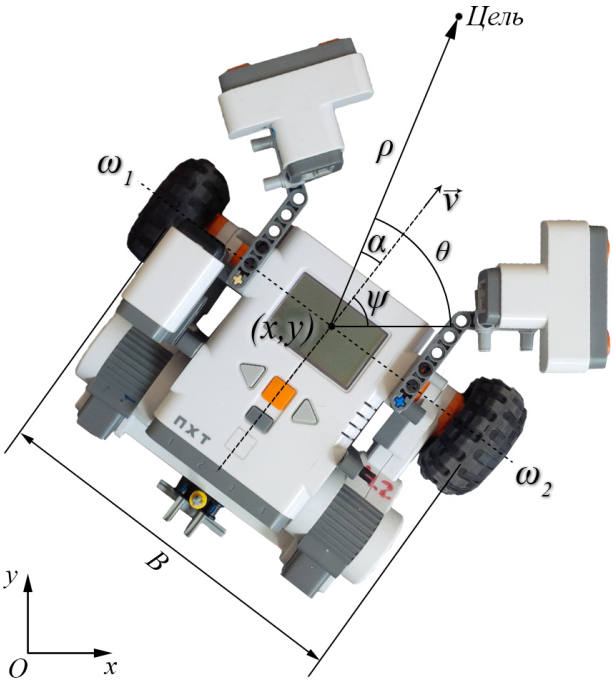

Рис. 1. Мобильный робот: ρ — расстояние до целевой точки, θ — угол между осью координат x и направлением на цель, α — курсовой угол, v — линейная скорость робота.

Задача

Робот должен достигнуть заданных координат ($\rho \to 0$, $\alpha \to 0$). То есть точка между колесами робота должна попасть в окружность с центром в заданных координатах и радиусом, равным допустимой погрешности.

Математическая модель

Рассмотрим математическую модель, описывающую навигацию робота в полярных координатах:

$$\begin{cases} \dot{\rho} = -v\cos\alpha, \\ \dot{\alpha} = -\omega + v\frac{\sin\alpha}{\rho}, \\ \dot{\theta} = -v\frac{\sin\alpha}{\rho}. \end{cases}$$

Фактически управление роботом возможно с помощью значений угловой и линейной скоростей (ω, v), поэтому достаточно найти такие их значения, при которых выполняется условие поставленной задачи. Для этого предлагается воспользоваться аппаратом функции Ляпунова, включающей в себя расстояние до цели и курсовой угол:

$$V(\rho,\alpha) = \frac{1}{2}\rho^2 + \frac{1}{2}\alpha^2.$$

Производная по времени должна быть не положительна для того, чтобы расстояние до цели и курсовой угол не возрастали.

$$\dot{V}(\rho,\alpha) = \rho\dot{\rho} + \alpha\dot{\alpha}.$$

Выразив производную через математическую модель получим:

$$\dot{V}(\rho,\alpha) = -\rho v\cos\alpha + \alpha(-\omega + v\frac{\sin\alpha}{\rho}).$$

Эта производная отрицательно определена, если мы выберем в качестве управляющего воздействия следующие значения скоротей:

$$\begin{cases} v = v_{max}\tanh\rho\cos\alpha, \\ \omega = k_\omega\alpha + v_{max}\frac{\tanh\rho}{\rho}\sin\alpha\cos\alpha, k_\omega > 0. \end{cases}$$

Для избегания препятствий введем поправку на вычисление курсового угла:

$$\begin{cases} \alpha = \theta - \psi - K_p(d_{min} - d), & d_{min} - d > 0; \\ \alpha = \theta - \psi, & d_{min} - d \le 0. \end{cases}$$

где K_p — коэффициент пропорциональной составляющей, d — расстояние до препятствия, d_{min} — минимальное расстояние до препятствия.

Также стоит добавить интегральную составляющую в расчет линейной скорости, чтобы скомпенсировать трение моторов.

Возможная погрешность достижения роботом точки с заданными координатами обусловлена проскальзыванием колес.

Результаты необходимых расчетов и построений

На рис. 1 — рис. 6 красный график постоен на основе результататов моделирования, синий по экспериментальным данным. Кроме того, на рис. 6 черным цветом обозначен объект используемый в качестве препятствия (см. видео).

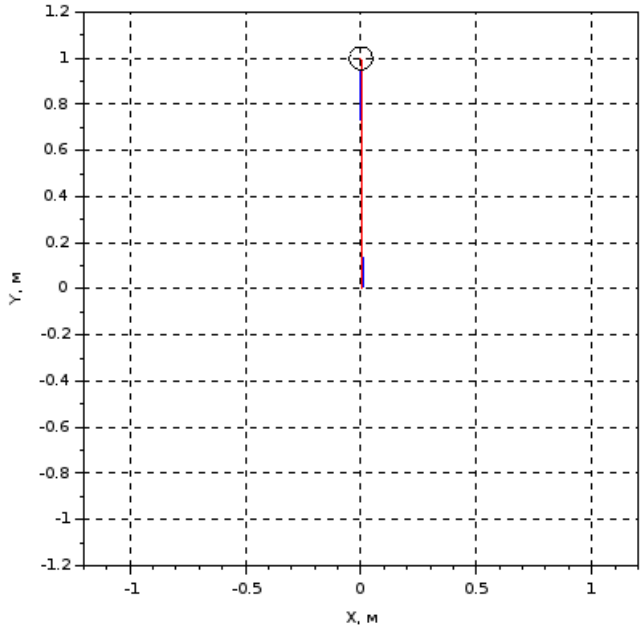

Рис. 1. Графики зависимостей координат Y от координат X, движение робота в точку (0, 1).

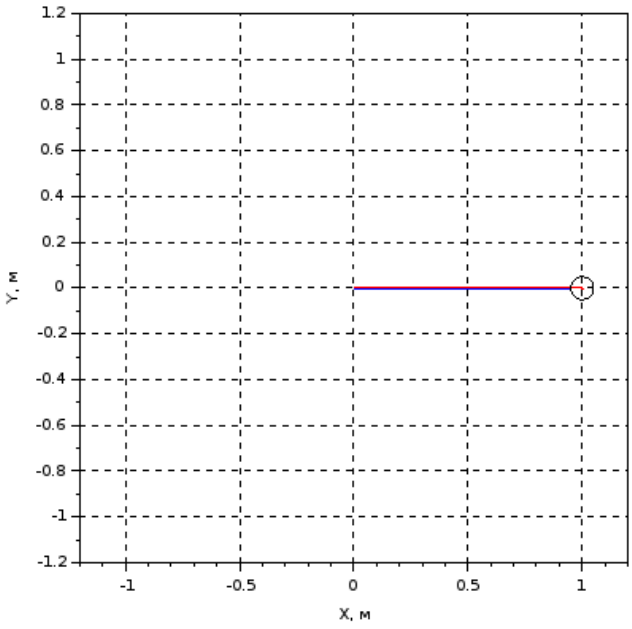

Рис. 1. Графики зависимостей координат Y от координат X, движение робота в точку (1, 0).

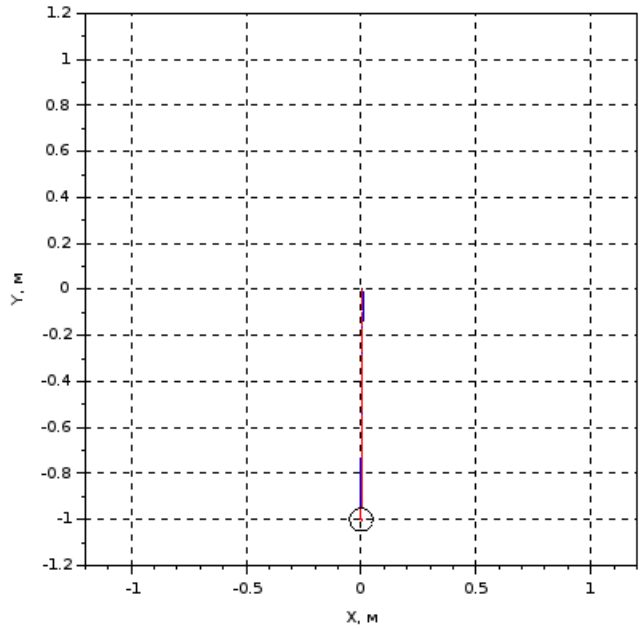

Рис. 3. Графики зависимостей координат Y от координат X, движение робота в точку (0, -1).

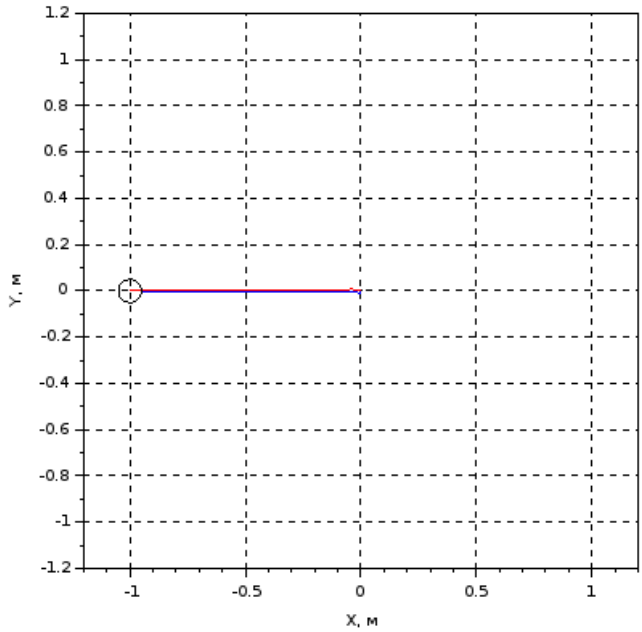

Рис. 4. Графики зависимостей координат Y от координат X, движение робота в точку (-1, 0).

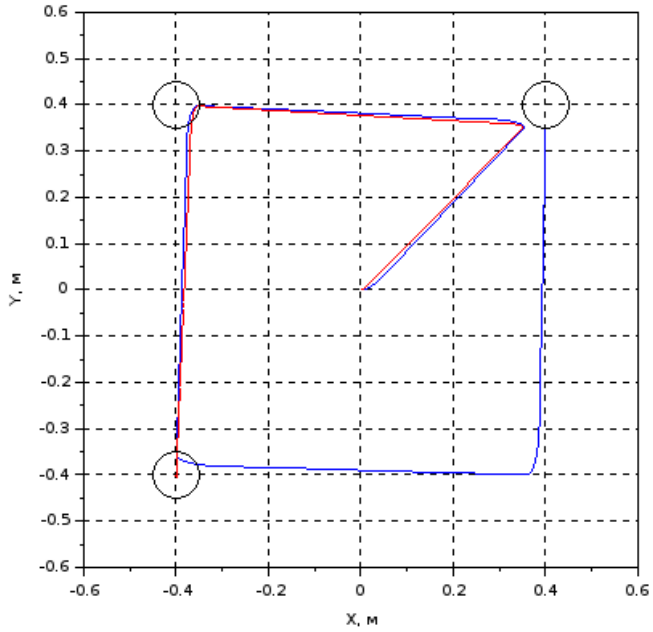

Рис. 5. Графики зависимостей координат Y от координат X, при решении задачи движения робота через координаты задающие квадрат. Ошибка №1.

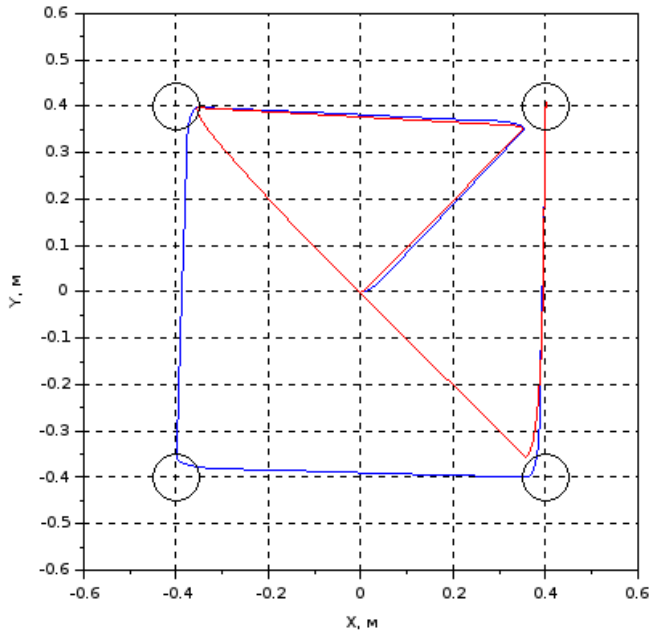

Рис. 5. Графики зависимостей координат Y от координат X, при решении задачи движения робота через координаты задающие квадрат. Ошибка №2.

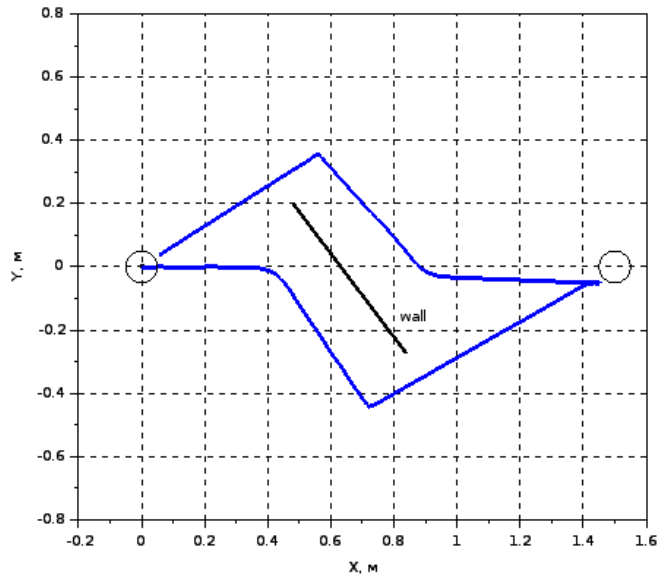

Рис. 6. Графики зависимостей координат Y от координат X, при движении робота к точке с координатами (0, 1.5) с обходом препятствий (см. видео)

Код программы для EV3

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
import math

k1 = 8
k2 = 0.5
k3 = 0.06
route = [[1.5, 0], [0, 0]]
voltage = 7.00
r = 0.02
B = 0.12
ok_zone = 0.05
min_dist = 35

mA = LargeMotor('outA')
mB = LargeMotor('outB')
us1 = UltrasonicSensor('in1')
us2 = UltrasonicSensor('in2')
us1.mode = 'US-DIST-CM'
us2.mode = 'US-DIST-CM'
gyro = GyroSensor('in3')
gyro.mode = 'GYRO-RATE'
fh = open('robot_data.txt', 'w')

desired_x = route[0][0]
desired_y = route[0][1]
current_x = 0
current_y = 0
complete = 0
mA.position = 0
mB.position = 0
prev_path = 0
integral = 0
```

```
gyro_angle = 0

try:
    gvalues = 0
    for i in range(100):
        gvalues += gyro.value()
        time.sleep(0.01)
    gyro_offset = gvalues/100
    current_time = time.time()
    while complete < len(route):
        dt = time.time() - current_time
        current_time = time.time()
        motorA_pos = mA.position * math.pi / 180
        motorB_pos = mB.position * math.pi / 180
        dist1 = us1.value() / 10
        dist2 = us2.value() / 10
        gyro_raw = gyro.value()
        gyro_speed = (gyro_raw - gyro_offset) * math.pi / 180
        gyro_angle += gyro_speed * dt
        path = (motorA_pos + motorB_pos)*(r/2)
        dpath = path - prev_path
        prev_path = path
        current_x += dpath*math.cos(gyro_angle)
        current_y += dpath*math.sin(gyro_angle)
        dx = desired_x - current_x
        dy = desired_y - current_y
        path_err = math.sqrt(dx**2 + dy**2)
        need_angle = math.atan2(dy, dx)
        cur_dist = min(dist1, dist2)
        if(cur_dist < min_dist):
            dist_err = k3*(min_dist - cur_dist)*math.copysign(1,
dist1 - dist2)
        else:
            dist_err = 0
        angle_err = need_angle - gyro_angle - dist_err
        if abs(angle_err) > math.pi:
            angle_err -= math.copysign(1, angle_err)*2*math.pi
        integral += path_err*dt
        u_straight = voltage*math.tanh
(path_err)*math.cos(angle_err) + k2*integral
        u_rotation = k1*angle_err +
math.sin(angle_err)*u_straight/path_err
        sA = u_straight + u_rotation
        sB = u_straight - u_rotation
        sA = sA * 100 / voltage
        sB = sB * 100 / voltage
        if abs(sA) > 100:
            sA = math.copysign(100, sA)
        if abs(sB) > 100:
            sB = math.copysign(100, sB)
        mA.run_direct(duty_cycle_sp=sA)
        mB.run_direct(duty_cycle_sp=sB)
        fh.write(str(current_x) + ' ' + str(current_y) + '\n \n')
```

```
            if (abs(dx) < ok_zone) and (abs(dy) < ok_zone):
                complete += 1
                if complete < len(route):
                    desired_x = route[complete][0]
                    desired_y = route[complete][1]
    finally:
        mA.stop(stop_action='brake')
        mB.stop(stop_action='brake')
        fh.close
```

Схема моделирования

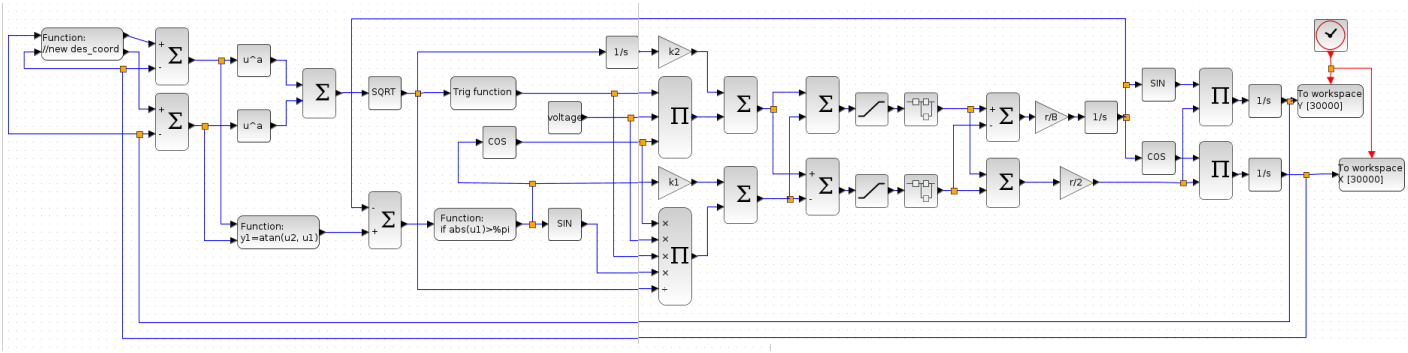

Рис. 7. Схема моделирования исследуемого процесса.

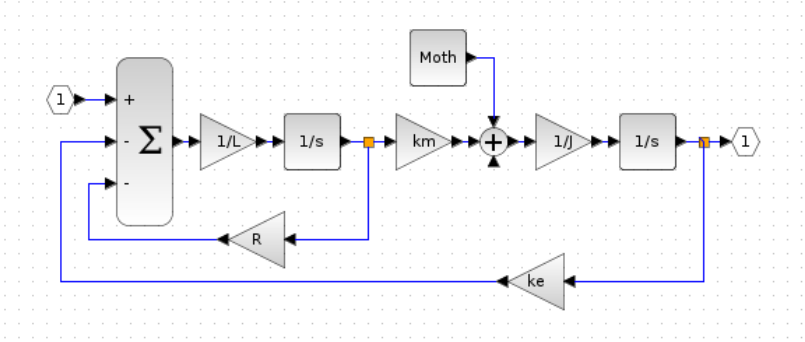

Рис. 9. Схема SuperBlock.

Код блока Function1

```
if abs(u1)>%pi
    y1=u1-sign(u1)*2*%pi
else
    y1=u1
end
```

Код блока Function2

```
//new des_coord
global complete
x_des=array(complete, 1)
y_des=array(complete, 2)
dx=x_des-u2
dy=y_des-u1
if (abs(dx) < zone) and (abs(dy) < zone)
    if complete < length(route)/2
        complete=complete+1
    end
end
y1=x_des
y2=y_des
```

Код программы Scilab

```
ke = 0.5
km = 0.5
R = 6.1
J = 0.0025
L = 0.0047

Moth = 0
r = 0.02 //радиус колеса
B = 0.12 //расстояние между колесами

k1 = 8
k2 = 0.5
k3 = 0.06
zone = 0.05
voltage = 7.00

sim_time = 30
sim_period = 0.01
sim_buffer = sim_time/sim_period

global complete
complete = 1
route = [0.4, 0.4; -0.4, 0.4; -0.4, -0.4; 0.4, -0.4; 0.4, 0.4]
importXcosDiagram("/home/aleksandr/ITMO_lab2/ev3/scilab/model3.zcos");
xcos_simulate(scs_m,4);
res = read("/home/aleksandr/ITMO_lab2/ev3/data/robot_data.txt", -1, 4)
x = res(:, 1)
y = res(:, 2)
xtitle('', 'X, м', 'Y, м')
xgrid(0)
plot2d(0,0,0,'031',' ',[-0.5,-0.5,0.5,0.5]);
plot2d(x, y, 2)
plot2d(X.values, Y.values, 5)

d = 0.1

for i = 1:length(route)/2
   xarc(route(i,1)-d/2, route(i,2)+d/2,d, d, 0, 360*64)
end
xs2png(0, "/home/aleksandr/ITMO_lab2/ev3/photos/graph.png")
```

Выводы

В результате проделанной работы была решена задача локальной навигации мобильного робота с дифференциальным приводом, а также реализовано движение робота через точки с заданными координатами.

Кроме того, была построена схема моделирования исследуемого процесса в среде Xcos. Сравнив график экспериментальных данных с графиком построенным на основании результатов моделирования схемы, мы удостоверились в том, что моделирование исследуемого процесса дает результаты схожие со значениями полученными в ходе прямых измерений.