МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительных систем и технологий

«Программирование алгоритмов управления роботом в Webots»

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине

«Аппаратное и программное обеспечение роботизированных систем»

Вариант № 1

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гай В.Е.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Марчус К.Р.

Гр. 19-В-1

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2022

# Цель работы

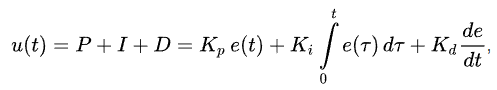
Получение навыков работы с алгоритмом компьютерной программы управления роботом.

# Задание

**Перевернутый маятник (Inverted pendulum)**

Этот тест направлен на разработку компьютерной программы, которая управляет колесным роботом, цель которого, как можно дольше удерживать маятник в равновесии. Язык программирования - Python, модель робота - робот e-puck. Во время моделирования к маятнику прикладывается некоторая случайная сила возмущения.

# Алгоритм



ПИД-регулятор состоит из 3 составляющих:

* Пропорциональная – P = Kpe(t). Необходим для корректировки сильно отклоняющихся значений.
* Интегральная – . Контролирует диапазон значений, до куда не достаёт пропорциональная составляющая, обеспечивает плавный результат.
* Дифференциальная – Предотвращает колебания значений.

# Код программы

**inverted\_pendulum.py:**

"""Sample Webots controller for the inverted pendulum benchmark."""

from controller import Robot

import math

# Get pointer to the robot.

robot = Robot()

# Get the time step of the current world.

timestep = int(robot.getBasicTimeStep())

# Get pointers to the position sensor and enable it.

ps = robot.getPositionSensor('pendulum sensor')

ps.enable(timestep)

# Get pointers to the motors and set target position to infinity (speed control).

leftMotor = robot.getMotor("left wheel motor")

rightMotor = robot.getMotor("right wheel motor")

leftMotor.setPosition(float('+inf'))

rightMotor.setPosition(float('+inf'))

leftMotor.setVelocity(0.0)

rightMotor.setVelocity(0.0)

max\_speed = min(rightMotor.getMaxVelocity(), leftMotor.getMaxVelocity())

# Define the PID control constants and variables.

KP = 100.0

KI = 100.0

KD = 0.0

integral = 0.0

previous\_position = 0.0

# Initialize the robot speed (left wheel, right wheel).

leftMotor.setVelocity(0.0)

rightMotor.setVelocity(0.0)

# Main loop: perform a simulation step until the simulation is over.

while robot.step(timestep) != -1:

# Read the sensor measurement.

position = ps.getValue()

# Stop the robot when the pendulum falls.

if math.fabs(position) > math.pi \* 0.5:

leftMotor.setVelocity(0.0)

rightMotor.setVelocity(0.0)

break

# PID control.

integral = integral + (position + previous\_position) \* 0.5 / timestep

derivative = (position - previous\_position) / timestep

speed = KP \* position + KI \* integral + KD \* derivative

# Clamp speed to the maximum speed.

if speed > max\_speed:

speed = max\_speed

elif speed < -max\_speed:

speed = -max\_speed

# Set the robot speed (left wheel, right wheel).

leftMotor.setVelocity(-speed)

rightMotor.setVelocity(-speed)

# Store previous position for the next controller step.

previous\_position = position

# Результат работы

Было проведено 9 тестов со следующими результатами:

|  |  |  |  |
| --- | --- | --- | --- |
| KP | KI | KD | T, сек |
| 100 | 100 | 0 | 60.41 |
| 100 | 100 | 100 | 60.37 |
| 100 | 80 | 0 | 60.31 |
| 50 | 90 | 100 | 60.29 |
| 70 | 80 | 90 | 60.27 |
| 80 | 110 | 0 | 60.27 |
| 60 | 100 | 100 | 60.25 |
| 60 | 110 | 0 | 60.24 |
| 40 | 110 | 50 | 3.04 |

Исходя из полученных данных, можно выявить закономерности:

* чем выше KP, тем больше времени маятник удерживается в равновесии
* чем ближе отношение KI и KP к 1, тем больше времени маятник удерживается в равновесии
* KD не сильно зависит от времени равновесия
* отмечено сильное ухудшение результата при KP ~ 40

# Вывод

Были получены базовые навыки работы с алгоритмом компьютерной программы управления роботом.