МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования 

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий   
Кафедра   
Вычислительные системы и технологии

Лабораторная работа № 4

по дисциплине

Программное обеспечение роботизированных систем

РУКОВОДИТЕЛЬ:

Гай В.Е.

СТУДЕНТ:

Тарасов А. В.

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

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

Нижний Новгород 2022

Задача:

Целью данного контрольного задания является разработка компьютерной программы, которая управляет колесным роботом для удержания маятника в равновесии как можно дольше. Язык программирования - Python, модель робота - e-puck. Во время моделирования к маятнику прикладывается случайная возмущающая сила.

Контрольная метрика t - это время, прошедшее с момента начала симуляции, наибольшее значение является наилучшим. Измерение времени прекращается, как только маятник падает вниз. Значение F показывает последнее значение силы возмущения в ньютонах, которая была приложена к маятнику. Это значение силы будет линейно увеличиваться со временем.

Данный робот использует PID регулятор для установки скорости вращения колес:

# ПИД-регулятор

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

derivative = (position - previous\_position) / timestep

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

Значения коэффициентов ПИД-регулирования KP, KI и KD могут быть настроены для улучшения стабильности маятника и достижения лучшего результата.

Листинг:

*"""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)

maxSpeed = min(rightMotor.getMaxVelocity(), leftMotor.getMaxVelocity())

# Define the PID control constants and variables.

KP = 100

KI = 100

KD = 60

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 > maxSpeed:

speed = maxSpeed

elif speed < -maxSpeed:

speed = -maxSpeed

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

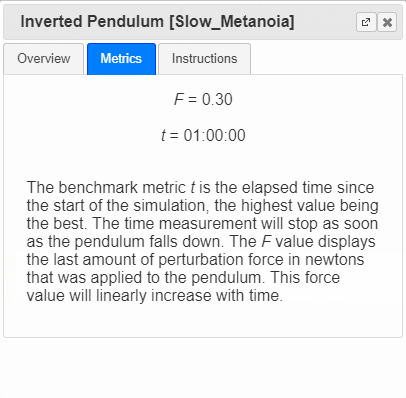
leftMotor.setVelocity(-speed)

rightMotor.setVelocity(-speed)

# Store previous position for the next controller step.

previous\_position = position

Результат:



Для прохождения бенчмарка был использован базовый контроллер с подобранными коэффицентами:

KP = 100

KI = 100

KD = 60