Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа №1

**Разработка ПИД-регулятора для Lego-робота.**

|  |
| --- |
| Выполнил: студент гр. 5.306М:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лаптев А. В.  Проверил: доц. каф. ВТиЭ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Соловьев А. А.  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

**Цель работы:** разработать прошивку для робота, которая должна стабилизировать прямолинейное движение с заданной скоростью.

**Задачи:**

1. Для Lego-робота реализовать ПИД-регулятор для стабилизации прямолинейного движения с заданной скоростью;
2. Реализовать вывод скорости каждого из моторов в консоль.

**Ход работы:**

1. Для стабилизации скорости в системе используется ПИД-регулятор. ПИД регулятор состоит из трёх составляющих: пропорциональной, интегрирующей и дифференциирующей, формируется как сумма трёх значений, умноженных каждая на свой коэффициент.

Данная сумма после вычислений становится управляющим сигналом, который подаётся на управляющее устройство, в данном случае на каждый из моторов робота. Для корректной работы ПИД-регулятора его коэффициенты требуют предварительной ручной настройки.

Пропорциональная составляющая предоставляет собой разность текущего значения с датчика и установки.

Интегральная составляющая представляет собой сумму ошибки, разность текущего и заданного значения, умноженную на период дискретизации системы.

Дифференциальная составляющая представляет собой разность текущей и предыдущей ошибки, деленную на время между измерениями.

1. При разработке прошивки были использованы следующие элементы API:

* LargeMotor - представляет собой обращение к серводвигателю робота;
* Методы: on - запуск мотора с передачей параметра значения скорости;
* Метод off - остановка мотора;
* Метод speed - для получения текущей скорости мотора.

**Алгоритм работы контроллера, реализующего ПИД-регулятор:**

Начало

1. Инициализация моторов робота.
2. Установка коэффициентов ПИД-регулятора (Kp, Ki, Kd) и желаемой скорости.
3. Создание экземпляра ПИД-реулятора.
4. Задание временного промежутка движения робота.

Цикл

1. Если время работы робота истекло, то переход к п.10.
2. Получение текущей скорости каждого колеса робота.
3. Вычисление средней скорости всех колес робота и вывод этого значения в консоль.
4. Получение корректирующего значения от ПИД-регулятора (метод update класса PIDController).
5. Установка скорректированного значения в качестве параметра скорости для каждого колеса и переход к п.5.

Конец цикла

1. Остановка моторов.

Конец

**Класс ПИД-регулятора:**

Начало

1. Инициализация класса ПИД-регулятора с инициализацией начальных параметров и методов.

Метод update для корректировки

1. Расчет значения ошибки.
2. Корректировка интегратора на значение ошибки.
3. Расчет производной как разности текущей и предыдущей ошибок.
4. Получение выходного значения сигнала по формуле ПИД-регулятора.
5. Сохранение текущего значения ошибки для дальнейшего использование.
6. Возврат выходного значения сигнала.

Конец

Программный код:

from ev3dev2.motor import LargeMotor, OUTPUT\_B, OUTPUT\_C

import time

class PIDController:

    def \_\_init\_\_(self, Kp, Ki, Kd, target\_speed):

        self.Kp = Kp

        self.Ki = Ki

        self.Kd = Kd

        self.target\_speed = target\_speed

        self.old\_err = 0

        self.integral = 0

    # Корректировка скорости

    def update(self, feedback\_speed):

        new\_err = self.target\_speed - feedback\_speed

        self.integral += new\_err

        derivative = new\_err - self.old\_err

    output\_speed = self.Kp \* new\_err + self.Ki \* self.integral + self.Kd \* derivative

        self.old\_err = new\_err

        return output\_speed

# Инициализация моторов

left\_motor = LargeMotor(OUTPUT\_B)

right\_motor = LargeMotor(OUTPUT\_C)

# Коэффициенты ПИД

Kp = 0.06

Ki = 0.01

Kd = 1

# Целевая скорость

target\_speed = 90

pid = PIDController(Kp, Ki, Kd, target\_speed)

start\_time = time.time()

while int(time.time() - start\_time) < 10:

    # Получение текущей средней скорости

    current\_left\_speed = left\_motor.speed

    current\_right\_speed = right\_motor.speed

    avg\_speed = (current\_left\_speed + current\_right\_speed) / 2

    print(avg\_speed)

    # Корректировка скорости

    update\_speed = pid.update(avg\_speed)

    left\_motor.on(update\_speed)

    right\_motor.on(update\_speed)

    time.sleep(0.01)

left\_motor.off()

right\_motor.off()