

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р. Е. АЛЕКСЕЕВА»
(НГТУ)

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

**Методические указания
по выполнению лабораторных работ
по дисциплине «Аппаратное и программное обеспечение
робототехнических систем»**

Направление подготовки

09.03.01 Информатика и вычислительная техника

код и наименование направления подготовки

Профиль подготовки

Программное обеспечение средств вычислительной техники и
автоматизированных систем

наименование профиля подготовки

Уровень высшего образования

Бакалавриат

Форма обучения

очная

Нижний Новгород

2021

Разработчик(и)/составитель(и) методических указаний по выполнению лабораторных работ по дисциплине «Аппаратное и программное обеспечение робототехнических систем»:

к.т.н., доцент, Гай В. Е.

ученое звание, степень, фамилия, инициалы

Кафедра «Вычислительные системы и технологии»

Дата, подпись _____

Методические указания по выполнению лабораторных работ по дисциплине «Аппаратное и программное обеспечение робототехнических систем» рассмотрены на заседании кафедры

«Вычислительные системы и технологии»

наименование кафедры

Протокол № ____ от «__» _____ г.

Заведующий кафедрой д.т.н, доцент Жевнерчук Д. В.

ученое звание, степень фамилия, имя, отчество

Дата, подпись 4 марта 2016 г.

Методические указания по выполнению лабораторных работ по дисциплине «Аппаратное и программное обеспечение робототехнических систем» утверждены учебно-методическим советом образовательно-научного института «Института радиоэлектроники и информационных технологий»

Протокол № ____ от «__» _____ г.

Методические указания по выполнению лабораторных работ по дисциплине «Аппаратное и программное обеспечение робототехнических систем» предназначены для бакалавров третьего курса, обучающихся по направлению подготовки 09.03.01 «Информатика и вычислительная техника», и содержат указания для проведения лабораторных занятий по курсу «Аппаратное и программное обеспечение робототехнических систем».

Цель методических указаний: помочь студентам при изучении учебной программы с использованием лекционных материалов и рекомендуемой учебно-методической литературы при формировании необходимых компетенций при изучении технологий построения сетей.

В процессе выполнения лабораторных работ по дисциплине «Аппаратное и программное обеспечение робототехнических систем» студент должен:

- строго выполнять весь объем самостоятельной подготовки, указанный в описаниях соответствующих лабораторных работ;

- знать, что выполнению каждой работы предшествует проверка готовности студента, которая проводится преподавателем;

- знать, что после выполнения работы студент должен представить отчет о проделанной работе с обсуждением полученных результатов и выводов.

В процессе выполнения лабораторных работ по данной дисциплине студент формирует и демонстрирует следующие компетенции:

ПКС-2. Способен разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования

Окончательная оценка за выполнение лабораторной работы студенту ставится после представления отчета о проделанной работе с обсуждением полученных результатов и выводов.

Порядок выполнения пропущенных работ по уважительным и неуважительным причинам: практические работы взаимосвязаны, поэтому пока не будет выполнена первая работа, студент не допускается к выполнению второй работы и т.д.

В ходе выполнения лабораторной работы студент должен выполнить предложенное задание и подготовить отчет о проделанной работе. Форма сдачи лабораторной работы предполагает демонстрацию выполненного задания и знаний теоретической части вопроса, рассмотренного в лабораторной работе.

Отчет по лабораторной работе должен содержать:

1. Тему и цель работы;
2. Вариант задания на практическую работу;
3. Краткие теоретические сведения и описание процесса выполнения работы;
4. Результаты выполнения практической работы;
5. Выводы.

Лабораторная работа № 1. Введение в программирование алгоритмов управления роботами с использованием симулятора Webots

Цель: получение навыков работы с симулятором Webots

Краткие теоретические сведения

Итак, представляю вашему вниманию webots. Из коробки с этим симулятором идёт большое количество демок, которые сразу можно открыть и поразглядывать. Вот одна из них. **Robots - gctronics - e-puck - e-puck.wbt**. После того как файл загрузился появляется виртуальное пространство в котором находятся статические объекты и робот. Этим роботом можно управлять с клавиатуры.

Webots для моделирования физики твёрдого тела использует Open Dynamics Engine. Поэтому в Webots можно создавать объекты с массой, моделировать трение, обнаруживать столкновения.

У роботов могут быть разные схемы движения, то есть роботы могут быть колёсным, летающими, шагающим, плавающими. В состав робота могут входить различные сенсоры и приводы.

Самое важное, что для робота можно написать программу управления, или как она называется в Webots - контроллер. Сейчас исходный код контроллер робота отображается в правой части окна.

Управлять движением робота в симуляторе можно на Си, Си++, Питоне, Джаве и Матлабе. Также можно создавать свои модели роботов и статичных объектов, но для этого нужны небольшие знания по созданию трёхмерных моделей и языка VRML97.

Вообще, в Webots симуляция включает следующие элементы:

1. Файл виртуального пространства Webots с расширением wbt, в котором находятся описания роботов и окружающего пространства.
2. Один или несколько контроллеров для роботов
3. Опционально, плагин, который может модифицировать физику Webots симулятора.

Виртуальное пространство в Webots содержит описание свойств робота и его окружения. Описание каждого объекта включает в себя: положение, ориентацию, геометрию, внешний вид, физические свойства. Это область в левой части экрана.

Виртуальное пространство организовано в виде иерархической структуры, то есть объект может включать другие объекты. Например, робот может включать четыре колеса, датчик расстояния и манипулятор, а в состав манипулятора может входить, например камера и ещё какие-то элементы.

Файл описания виртуального пространства не включает контроллер, который описывает поведение робота. Этот файл содержит только имя контроллера для робота.

Что такое контроллер? Контроллер это программа, которая управляет поведением робота. Он запускается при старте симуляции, при этом каждый контроллер запускается как отдельный процесс. И процесс контроллера связывается с моделью робота. Несколько роботов могут использовать один и тот же код контроллера, но для каждого робота в этом случае будут запускаться отдельные процессы.

При использовании для написания контроллера Си или Си++ контроллер должен быть скомпилирован. При использовании Матлаба или Питона в операционной системе должен быть установлен соответствующий интерпретатор, который нужно подключить к Webots.

Исходные и бинарные файлы каждого контроллера хранятся вместе в каталоге контроллера, который называется “controllers”. Эта папка должна находиться в папке проекта Webots.

В нижней части экрана отображаются результат компиляции и вывод контроллера.

Давайте теперь посмотрим на меню симулятора.

Меню Файл включает стандартные элементы по созданию нового виртуального пространства (мира), его сохранения или открытия существующего мира. Пункт Reload World загружает текущее виртуальное пространство из сохранённой версии и начинает симуляцию с начала.

Пункт Reset Simulation восстанавливает исходное состояние симуляции, то элементы виртуального пространства получают начальное состояние, а виртуальное пространство заново не загружается.

В меню Файл также можно создать и сохранить новый текстовый файл, импортировать VRML модель в конец дерева объектов. Обычно это используется для добавления пользовательских объектов в среду симулятора. Также виртуальное пространство можно экспортировать в VRML. Можно сделать скриншот сцены и запустить видеозапись симуляции.

В меню Правка или Edit находятся классические операции по правке текста.

Пункты меню View предназначены для управления обзором за виртуальным пространством.

Пункт меню Follow Object позволяет переключаться между статической точкой обзора на мобильный объект и точкой обзора, которая следует за объектом. Чтобы камера следила за объектом нужно выбрать объект на сцене и затем выбрать один из пунктов меню Follow Object.

Пункт Restore Viewpoint сбрасывает положение точки обзора в начальное значение. Обычно выполнить такое действие требуется когда вы потерялись в сцене и хотите вернуться к изначальной точке обзора.

Пункт Move viewpoint to object помещает точку обзора на выбранный элемент сцены.

Пункт Change View устанавливает точку зрения в соответствии с одним из шести положений: и показывает вид слева, справа, снизу, сверху, спереди или сзади.

Кнопка Projection позволяет выбрать между перспективной и ортографической проекцией, то есть проекцией в которой отображаются объекты в окне симулятора. Перспективная проекция соответствует естественной проекции в которой чем дальше объект от наблюдателя, тем меньше он выглядит. В ортографической проекции расстояние до наблюдателя не влияет на размер объекта. В Ортографической проекции линии, параллельные в модели рисуются параллельными на экране. В ортографической проекции не отображаются тени.






Кнопка Rendering позволяет выбирать между полным и каркасным рендером. В полном рендере все объекты отображаются с учётом материалов, текстур, цветов. В

каркасном рендере отображаются только геометрические примитивы из которых состоят объекты. Такой режим полезен при отладке вновь создаваемых объектов и отладке столкновений. Также есть меню Optional Rendering, в нём можно выбирать какие именно элементы сцены и какая дополнительная информация должны отображаться.

Пункт Disable selection запрещает изменение выбранного объекта. Это полезно при моделировании, когда нужно поменять точку обзора без изменения видимых и выбранных объектов сцены.

Пункт Lock Viewpoint запрещает изменение положения и ориентации точки обзора.

Меню симуляции (кнопки на панели)

На панели инструментов есть кнопки, которые позволяют поставить симуляцию на паузу , выполнить пошагово , в реальном времени , в быстром  и очень быстром  режимах. В последнем режиме рендер отключается и этот режим подходит для симуляций, которые интенсивно используют процессор.

Меню Overlays. Это меню управляет окнами, в которых отображается показания камеры, лазерного дальномера и других датчиков. Некоторые пункты этого меню будут активны только при выбранном роботе.

Меню Tools показывает окна, которые открыты в Webots.

Пункт меню Tools-Preferences включает настройки Webots: выбранный язык, внешний вид среды, режим симуляции в котором запускается Webots, число потоков, отводимых Webots, то, как выглядит команда для запуска питона, настройки рендера, связанные с OpenGL.

Меню помощников (Визардов) содержит команды для запуска диалоговых окон для создания новых проектов и контроллеров.

Меню Help. Как мне кажется самое интересное, что есть в меню Help это информация об OpenGL, видеокарточке и драйвере. Почему? Чтобы понять проблемы с рендером.

На главной панели инструментов есть кнопка для добавления новых объектов и ярлыки на пункты меню Файл, View и Simulation.

Спидометр и виртуальное время. Спидометр показывает скорость симуляции на вашем компьютере. Он отображается на главной панели инструментов и показывает насколько симуляция быстрее или медленнее реального времени. Если там отображается 2, тогда симуляция быстрее реального времени в два раза.

Слева от спидометра показывается виртуальное время: часы, минуты, секунды и миллисекунды.

Шаг времени симуляции устанавливается в поле basicTimeStep. Он измеряется в миллисекундах и показывает время выполнения одного шага симуляции в режиме выполнения по шагам.

Давайте перейдём к созданию первой симуляции в Webots. Webots запущен. Теперь нужно поставить на паузу симуляцию и создать новый проект с помощью меню Wizards, выбрав пункт New Project Directory. Установим имя каталога, в котором будет храниться проект first_sim, имя файла в котором хранится виртуальное пространство first_sim.wbt и выберем все галочки, включая последний пункт add a rectangle area.

Webots отображает список каталогов и файлов, которые он создал и это стандартная иерархия файлов в проекте.

Теперь после создания симуляции можно изучить структуру сцены, которая расположена в левой части экрана. В ней сейчас находятся следующие узлы:

- WorldInfo: - глобальные параметры симуляции
- Viewpoint: - определяет параметры точки зрения главной.
- TexturedBackground: - фон сцены (горы в текущем примере)
- TexturedBackgroundLight: - подсветка фона
- defines the light associated with the above background.
- RectangleArena: объект, который мы добавили в сцену.

Можно перемещать точку обзора с помощью мыши: левой и правой кнопкой мыши и колёсиком. Колёсико мыши масштабирует сцену, зажатая левая кнопка выполняет вращение, зажатая правая кнопка выполняет линейное перемещение.

У каждого узла или элемента сцены есть настраиваемые параметры. Давайте модифицируем некоторые параметры прямоугольной области. Нужно щёлкнуть два раза на узле RectangleArena в дереве сцены и это отобразит его поля.

Установим для поля floorTileSize значения 0.25 и 0.25 вместо 0.5 и 0.5. Это размер плитки, эффект можно увидеть сразу.

Установите значение поля wallHeight в 0.05 вместо 0.1. В результате уменьшится высота стены.

Теперь нужно два раза щёлкнуть на RectangleArena и нажать кнопку Add. В результате откроется диалоговое окно в котором надо выбрать пункт PROTO nodes (Webots Projects) / objects / factory / containers / WoodenBox (Solid). В результате в середине сцены появится большая коробка. Теперь давайте изменим поля этой коробки

1. Установим размер в 0.1 0.1 0.1 вместо 0.6 0.6 0.6.
2. Изменим положение в 0 0.05 0 вместо 0 0.3 0. Также можно потянуть за зелёную стрелку, чтобы изменить координату Игрек.
3. Теперь нужно нажать Shift и переместить коробку в любой угол области
4. Выберите коробку, нажмите Ctrl-C, Ctrl-V для того, чтобы копировать и вставить её, а после с шифтом переместите коробку в какой-нибудь угол.
5. Сохраните виртуальное пространство.

Давайте теперь добавим в сцену робота E-puck. E-puck это маленький робот с дифференциальным приводом, 10 светодиодами и несколькими сенсорами, включая 8 сенсоров расстояния и камеру.

Чтобы добавить робота сначала нужно остановить симуляцию и сбросить время в 0. Это стандартная операция перед добавлением любых объектов. То есть модификация виртуального пространства всегда должна выполняться в следующем порядке: пауза, сброс, модификация и сохранения симуляции.

Выберем последнюю добавленную коробку и нажмем кнопку Add и далее надо выбрать PROTO nodes (Webots Projects) / robots / gctronic / e-puck / E-puck (Robot). Робот должен появиться в центре прямоугольной области. Если теперь запустить симуляцию робот должен начать движение, мигать светодиодами и пытаться обходить препятствия. Это поведение запрограммировано в контроллере робота. Также есть в левом верхнем углу чёрное окошко. Это вид с камеры робота. Оно будет оставаться чёрным, пока в контроллере не будет подключена камера. Окошко можно передвигать и закрыть, нажав на крестик. Снова открыть камеру можно через меню Overlays.

Теперь давайте попробуем применить силовое воздействие к роботу. Для этого нужно в режиме симуляции нажать Alt зажать левую клавишу мыши, потянуть и отпустить. Силовое воздействие нельзя применить к коробке, поскольку она не имеет массы и рассматривается как приклеенная к полу. Чтобы сделать так, чтобы коробка что-то весила нужно установить в поле mass определённое значение, например 0.2.

Давайте попробуем создать контроллер для управления роботом. Контроллер будем делать на Питон, который сначала нужно установить. Питон нужен версии 3.7 и обязательно при установке выберите пункт Customize installation (настройка установки) и добавьте Питон в переменную среды для Windows.

Поле controller робота указывает какой контроллер соответствует роботу. Создадим новый контроллер и укажем для него имя e_puck_go_forward. Для этого нужно выбрать пункт меню Wizards / New Robot Controller. После выполнения этого действия в той папке, где хранятся все файлы виртуального пространства будет создана папка controllers. Шаблон контроллера теперь показан в редакторе кода. Теперь нужно связать робота и созданный контроллер. Для этого нужно выбрать поле controller робота E-puck, нажать Select и выбрать только что созданный контроллер. После этого нужно сохранить виртуальное пространство.

Теперь давайте модифицируем имеющийся код контроллера. Добавим две строчки, которые получают доступ к моторам робота:

```
# get the motor devices
leftMotor = robot.getMotor('left wheel motor')
rightMotor = robot.getMotor('right wheel motor')
```

И две строчки для установки расстояния, которые должны пройти колёса.

```
# set the target position of the motors
leftMotor.setPosition(10.0)
rightMotor.setPosition(10.0)
```


Информацию о роботе, о том как называются его моторы, о сенсорах можно найти в документации к Webots. Вот например на этой странице <https://cyberbotics.com/doc/guide/epuck> можно покрутить колёса, посмотреть как именуются сенсоры и где они установлены.

Давайте перейдём в симулятор и я покажу как в реальном времени посмотреть на показания датчиков робота. Для этого нужно два раза щёлкнуть по роботу левой клавишей мыши. Откроется окно, в котором показана схема робота и показания датчиков, но только тех, которые подключены в контроллере. Чтобы отобразить показания всех датчиков нужно нажать кнопку Enable All. После нажатия видны сразу показания датчиков расстояния, датчиков освещённости и другие. Если выбрать пункт меню View - Optional rendering - Show Distance Sensor Rays можно увидеть рендер лучей датчиков расстояния.

Датчик расстояния здесь работает достаточно интересно, зависимость между числом, которое он выдаёт и расстоянием до объекта указана в справочной информации о роботе. Чем большее значение показывает датчик, тем ближе робот к препятствию.

<https://cyberbotics.com/doc/guide/epuck>

```
from controller import Robot, Motor
TIME_STEP = 64
# create the Robot instance.
robot = Robot()

# get the motor devices
leftMotor = robot.getMotor('left wheel motor')
rightMotor = robot.getMotor('right wheel motor')
# set the target position of the motors
leftMotor.setPosition(10.0)
rightMotor.setPosition(10.0)
while robot.step(TIME_STEP) != -1:
    pass
```

Небольшие пояснения кода программы. В самом начале импортируются функции. После этого из виртуального пространства считывается значение параметра BasicTimeStep.

В цикле while вызывается функция robot.step(timestep). Эта функция Синхронизирует данные контроллера и симулятора. Она должна быть в каждом контроллере и должна вызываться через одинаковые промежутки времени. Параметр функции - время, которое она выполняется, то есть после указанного промежутка времени функция возвращает управление в контроллер, функция вычисляет указанное количество миллисекунд симуляции и возвращает управление. И это время симулятора, а не реальное время. То есть выполнение этой функции может продлиться и одну секунду и две, может и меньше, в зависимости от сложности симуляции.

Условие выхода из цикла - это когда функция step вернёт -1. Эта функция может вернуть -1 когда Webots остановит контроллер. Обычно контроллер останавливается, когда симуляция сбрасывается, выполняется выход из webots, изменяется имя контроллера или загружается новое виртуальное пространство.

Pass - Оператор-заглушка, равноценный отсутствию операции.

Теперь нужно код сохранить, и запустить симуляцию. Если всё хорошо, робот поедет вперёд и остановится когда колёса повернутся на угол в 10 радиан.

Рассмотрим способ управления скоростью колёс. Для этого перед циклом `while` нужно изменить две строки и добавить три новых. Первые две строки указывают, что колёса робота будут вращаться бесконечно, а вторые две - устанавливают скорость на левом и правом колесе. Запускаю симуляцию и видно, что робот едет со скоростью, меньшей чем в предыдущем примере.

```
MAX_SPEED = 6.28
```

```
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))
```

```
# set up the motor speeds at 10% of the MAX_SPEED.
leftMotor.setVelocity(0.1 * MAX_SPEED)
rightMotor.setVelocity(0.1 * MAX_SPEED)
```

```
from controller import Robot, Motor
TIME_STEP = 64
MAX_SPEED = 6.28
# create the Robot instance.
robot = Robot()
# get a handler 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'))
# set up the motor speeds at 10% of the MAX_SPEED.
leftMotor.setVelocity(0.1 * MAX_SPEED)
rightMotor.setVelocity(0.1 * MAX_SPEED)
while robot.step(TIME_STEP) != -1:
    pass
```

Задание

1. Добавить робота, указанного в списке ниже и соответствующего выданному варианту в симулятор Webots
2. Создать для робота контроллер таким образом, чтобы робот двигался не по прямой (для этого можно для разных моторов робота установить разную скорость)

1. <https://cyberbotics.com/doc/guide/bb8>
2. <https://cyberbotics.com/doc/guide/altino>
3. <https://cyberbotics.com/doc/guide/boebot>
4. <https://cyberbotics.com/doc/guide/create>
5. <https://cyberbotics.com/doc/guide/elisa3>
6. <https://cyberbotics.com/doc/guide/epuck>
7. <https://cyberbotics.com/doc/guide/firebird6>
8. <https://cyberbotics.com/doc/guide/hemisson>
9. <https://cyberbotics.com/doc/guide/khepera1>
10. <https://cyberbotics.com/doc/guide/khepera2>
11. <https://cyberbotics.com/doc/guide/khepera3>

12. <https://cyberbotics.com/doc/guide/khepera4>
13. <https://cyberbotics.com/doc/guide/koala>
14. <https://cyberbotics.com/doc/guide/microbot>
15. <https://cyberbotics.com/doc/guide/mindstorms>
16. <https://cyberbotics.com/doc/guide/moose>
17. <https://cyberbotics.com/doc/guide/pioneer2>
18. <https://cyberbotics.com/doc/guide/pioneer-3at>
19. <https://cyberbotics.com/doc/guide/surveyor>
20. <https://cyberbotics.com/doc/guide/pioneer-3dx>
21. <https://cyberbotics.com/doc/guide/sojourner>

Контрольные вопросы

1. Что такое webots?
2. Какие в Webots существуют основные элементы интерфейса?
3. Какие команды используются для управления моторами робота?

Лабораторная работа № 2. Программирование алгоритмов управления роботом в Webots

Цель: получение навыков работы с алгоритмами управления роботами

Краткие теоретические сведения

Задание

Задание: выполнить вариант и загрузить программу на платформу для соревнований, записать видео работы, подготовить отчет с подробным описанием результатов. В отчет вставить результаты с соревнования. Есть примеры, можно на них посмотреть (видео на сайте).

Задача 1. Обход препятствий

https://robotbenchmark.net/benchmark/obstacle_avoidance

Этот тест направлен на создание надежного и эффективного алгоритма обхода препятствий для робота Thymio II с использованием языка программирования Python. Цель состоит в том, чтобы робот пересек комнату и как можно быстрее достиг противоположной стены, избегая при этом всех столкновений с препятствиями.

Для проверки работоспособности алгоритма, препятствия располагаются случайным образом при каждом пробеге. Эталонная метрика t-это время, необходимое роботу, чтобы пересечь комнату. Минимизация этого времени является целью для этого сценария. Таймер останавливается после того, как робот находится в пределах 40 см от задней стенки или прошло более 1 минуты 20 секунд (максимальное время). Любое столкновение с препятствиями в комнате считается немедленным провалом задания.

Как улучшить свое время?

Thymio II в базовом контроллере считывает значения с датчиков переднего расстояния и использует их для непосредственного управления скоростью своих колес. Чтобы успешно избегать препятствий с помощью механизма реагирования датчика, реализованного здесь, робот должен двигаться на полной скорости. Чтобы заставить робота

двигаться на полной скорости, необходимо изменить силу воздействия датчиков расстояния на управление колесом.

Не зная абсолютного направления движения, робот может дезориентироваться и не достичь другой стороны комнаты. Для борьбы с этим Thymio II оснащен устройством компаса, хотя контроллер по умолчанию не использует его. Компас можно включить, используя следующий фрагмент кода:

```
# import Compass module
from controller import Compass
# get robot's Compass device
compass = robot.getCompass("compass")
# enable the Compass
compass.enable(timestep)
# to read values
values = compass.getValues()
```

Полную документацию по устройству Compass можно найти [здесь](#).

Транспортные средства Брайтенберга

Как и робот e-ruck, использованный в других тестах, робот Thymio II приводится в движение двумя колесами, каждое со своим двигателем и свободным вращением.

Этот контроллер робота и многие другие средства обхода препятствий основаны на автомобиле Брайтенберга. Автомобиль Брайтенберга, названный в честь нейробиолога Валентино Брайтенберга, представляет собой автомат, который может свободно перемещаться, используя данные от датчиков, установленных на автомобиле.

В этом примере сенсорные входы с левой стороны робота управляют скоростью правого колеса и наоборот. Хотя механизмы, приводящие в движение автомобили Брайтенберга, просты, поведение может быть сложным, как показывает этот тест.

Задача 2. Движение по квадрату

https://robotbenchmark.net/benchmark/square_path/

Этот тест направлен на разработку программы, которая управляет роботом Pioneer, чтобы следовать квадратному пути размером 2 на 2 метра.

Метрика, используемая для оценки робота, применяется к 4 отдельным сегментам пути, которые соответствуют 4 сторонам квадрата.

Каждый сегмент определяется как коридор, лежащий на одном краю квадрата. «Цель» одного сегмента определяется как вершина между текущим и следующим сегментом. Чтобы добраться до следующего сегмента, робот должен пересечь линию, проходящую через центр квадрата и вершину «цели».

Для каждого отдельного сегмента мы вычисляем производительность, которая основана на 3 различных параметрах: путь (насколько хорошо роботу удалось приблизиться к "идеальному" маршруту), время, необходимое для прохождения этого сегмента, и расстояние до цели, которое в основном используется для оценки того, насколько робот близок к цели в текущем сегменте.

Элемент пути - это линейная величина, обратно пропорциональная ширине коридора, по которому проходит путь робота для этого сегмента. Она изменяется от 100%, когда робот двигался по идеальной линии, до 0%, когда коридор шире, чем его длина.

Элемент времени представляет собой линейное значение, обратно пропорциональное времени, необходимому для прохождения этого сегмента. Оно изменяется от 100% (при 0 секунд) до 0% (при 20 и более секундах).

Элемент расстояния показывает, как далеко до цели (одна вершина квадрата) находится робот в текущем сегменте. Если робот уже прошел этот сегмент, это будет то, как далеко был робот, когда он покинул этот сегмент. Это значение изменяется от 100%, для расстояния 0, до 0%, когда расстояние больше одной стороны квадрата. Значение определяется как:

$$10 * (1 - \text{расстояние} / 2) + 90 * \exp(-3 * \text{расстояние})$$

Экспоненциальная часть предназначена для поощрения робота, который очень точно достигает цели, поэтому робот, который движется в правильном направлении, не теряет очки из-за временной части производительности. Только медленный робот должен терять очки, тогда как средний по скорости робот просто набирает меньше, чем быстрый.

Производительность для сегмента представляет собой средневзвешенное значение трех элементов. Элемент времени имеет половину веса двух других.

Общая производительность - это среднее значение по 4 сегментам.

Как повысить производительность?

Базовый контроллер - это программа, которая просто устанавливает скорость и ждет заданное время, пока робот не достигнет желаемого положения. Самое простое, что можно сделать, чтобы улучшить производительность - это настроить продолжительность шагов, чтобы повысить точность пути.

Более грамотным решением было бы использовать датчики колес робота. Pioneer 3-DX имеет датчики в колесе, которые позволяют роботу знать пройденное расстояние. Используя это значение, вы можете заставить робота остановиться ровно через 2 метра.

Для того, чтобы использовать датчики, вы сначала должны включить их, так как по умолчанию они отключены.

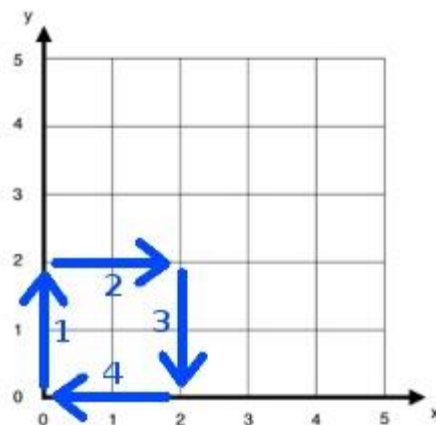
```
rightWheelSensor = robot.getPositionSensor('right wheel sensor')
rightWheelSensor.enable(16) # Refreshes the sensor every 16ms.
You can then access the value of the sensor with its getValue() method.
rightWheelSensor.getValue()
```

Некоторая информация о роботе, которая может быть полезна для улучшения контроллера:

- Диаметр колес - 195 мм.
- Расстояние между двумя колесами составляет 330 мм.

Как выполнить тест с реальным роботом?

Для выполнения бенчмарка на реальном роботе вам понадобится Pioneer 3-DX и средства для записи его положения в режиме реального времени во время бенчмарка. Робот должен двигаться по квадратному пути размером 2 на 2 м, как показано на этом изображении:



При запуске программы контроллера в реальной среде с помощью робота Pioneer 3-DX необходимо сгенерировать CSV-файл, содержащий полученную траекторию. Файл CSV должен содержать список точек с указанием времени, связанного с каждой точкой. Пункты должны быть упорядочены в хронологическом порядке.

Каждая строка должна содержать 3 элемента, разделенных запятыми:

- координата x, в метрах
- координата y, в метрах
- время с начала отсчета, в секундах

Затем вы можете использовать этот сценарий для вычисления производительности. Для запуска скрипта на вашем компьютере должен быть установлен Python. Извлеките все файлы и вызовите “real_robot_performance.py” с вашим CSV-файлом в качестве аргумента для вычисления производительности.

Например, вы можете запустить обработку файла “sample.csv” с помощью следующей команды в терминале:

```
python real_robot_performance.py sample.csv
```

Обратите внимание, что предоставленный скрипт является утилитой для оценки траекторий, полученных с помощью реального робота.

Он не содержит траекторий, записанных на robotbenchmark.net. Если вы хотите сохранить траекторию моделируемого робота при запуске бенчмарка, то вы можете записать точки на консоли, а затем скопировать их в файл.

Контрольные вопросы

1. Какие датчики используются для организации обхода препятствий роботом?
2. Какие датчики используются для организации движения робота вдоль стены?
3. Что такое PID регулятор?

Лабораторная работа № 3. Программирование алгоритмов управления роботом в Webots

Цель: получение навыков работы с алгоритмами управления роботами

Краткие теоретические сведения

См. лекционный материал.

Задание

1) цифровой двойник: создание цифрового двойника робота ElcBot. Компоненты робота: рама, два мотора, два колеса, kinect, ультразвуковые датчики (можно смоделировать с помощью DistanceSensor), два динамика по бокам робота, третье колесо - поддерживающее. Написать контроллер для робота для его движения вдоль стены на заданном расстоянии (с помощью датчиков расстояния).

Справочная информация: размеры и робота указаны на сайте: <https://tinyroboticsteam.github.io/ElcBot/>, информация о положении датчиков указана здесь: <https://www.youtube.com/watch?v=UUsOTW9TLhU>.

Состав сцены: нужно смоделировать часть шестого корпуса в районе аудитории 154, включая саму аудиторию.

2) движение по линии: робот должен выполнить движение по линии, от начала до конца;

Робот: e-puck (<https://cyberbotics.com/doc/guide/epuck>), в исходной модификации этого робота нет датчиков для обнаружения линии. Инструкция по их добавлению и пример использования находится по следующим ссылкам:

https://cyberbotics.com/doc/guide/epuck#e-puck_line-wbt

https://cyberbotics.com/doc/guide/epuck#e-puck_line_demo-wbt

Структура сцены: размер 5*5 метров (ограничена стенами), линия, вдоль которой движется должна быть сплошной с прерывистыми участками, размер разрывов должен быть не больше половины размера робота, на линии также должны находиться препятствия, которые робот должен обходить, линия должна быть проложена не только по ровной поверхности, но и на некотором возвышении над полом (должна быть сделана горка с подъёмом и спуском).

Справочная информация: <http://is.ifmo.ru/projects/robochuck/doc.pdf>

3) обход лабиринта: робот должен найти выход из лабиринта;

Робот: e-puck (<https://cyberbotics.com/doc/guide/epuck>);

Структура сцены: размер лабиринта 3*3 метра, расстояние между стенами лабиринта - 0.15 метра, в лабиринте на пути следования робота должны быть препятствия, которые робот должен успешно обходить; для обнаружения препятствий и ориентирования в лабиринте нужно использовать датчики расстояний;

Генератор лабиринтов: <http://www.mazegenerator.net/>;

Справочная информация: https://myrobot.ru/articles/logo_mazesolving.php.

4) групповое движение роботов: на сцене находится несколько роботов, в начальном положении они стоят друг за другом, первый робот начинает движение, два других анализируя его положение относительно себя также начинают движение ровно по его траектории; роботы должны уметь обходить препятствия; для обнаружения препятствий и других роботов нужно использовать датчики расстояния; препятствия и роботов можно различать по цвету;

Роботы: (из приведённого списка нужно выбрать трёх роботов так, чтобы их датчики расстояния позволяли обнаруживать других роботов):

<https://cyberbotics.com/doc/guide/firebird6>

<https://cyberbotics.com/doc/guide/elisa3>

<https://cyberbotics.com/doc/guide/altino>

<https://cyberbotics.com/doc/guide/pioneer-3dx>

<https://cyberbotics.com/doc/guide/pioneer-3at>

<https://cyberbotics.com/doc/guide/koala>

<https://cyberbotics.com/doc/guide/khepera4>

<https://cyberbotics.com/doc/guide/khepera3>

Структура сцены: размер 5*5 метров (ограничена стенами), на сцене присутствуют препятствия.

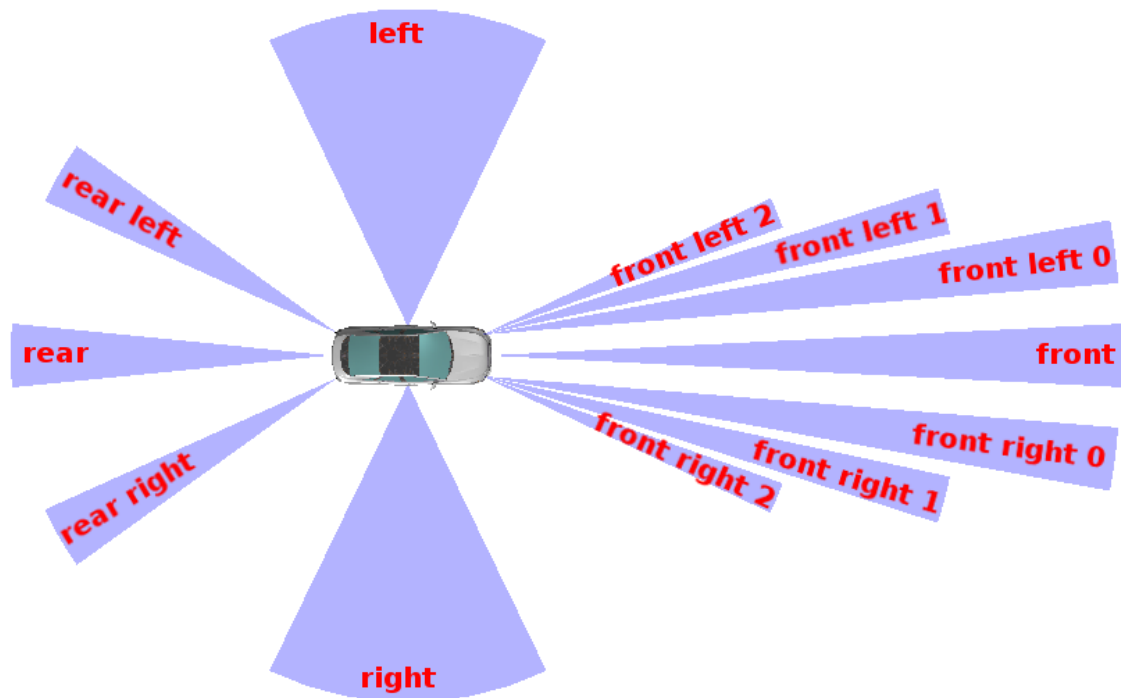
5) управление автомобилем: разработать контроллер для автомобиля, который позволит переместиться автомобилю как можно дальше за одну минуту времени без столкновений с другими транспортными средствами или статическими объектами. Webots включает виртуальное пространство, в котором реализована сцена для решения данной задачи. Есть прототип контроллера, который связан с этим виртуальным пространством: File - Open Sample World - samples - robotbenchmark - highway_driving - highway_driving.wbt. Прототип написан на Python. В этом примере показано как подключиться к датчикам машины, чтобы можно было получать с них данные, а также как управлять скоростью и колёсами машины. В примере контроллера машина едет прямо и снижает скорость, если перед ней детектируется другое транспортное средство.

Перед запуском сцены нужно установить модуль shapely (python -m pip install shapely).

С этим соревнованием в комплекте идёт два контроллера: первый для управления машиной, второй для оценки расстояния, которое машина преодолела.

Требования к разрабатываемому контроллеру автомобиля: он должен позволять ехать автомобилю в правой полосе, не заезжая за сплошную полосу и не сталкиваясь с другими автомобилями.

Схема расположения датчиков автомобиля показана на рисунке ниже.



Выравнивать положение машины на дороге можно с помощью датчиков, расположенных слева и справа, поскольку слева и справа у дороги есть барьеры. Также это можно делать с помощью камеры. Машина также оснащена GPS датчиком и передней камерой.

Ссылка на соревнование для загрузки контроллера: https://robotbenchmark.net/benchmark/highway_driving/. Советую изучить страничку, так как там есть информацию о решении этой задачи и можно посмотреть как работают существующие решения.

б) отслеживание положения объекта: разработать контроллер, который отслеживает положение объекта в кадре и управляет камерой, чтобы следить за объектом более точно. Webots включает виртуальное пространство, в котором реализована сцена для решения данной задачи. Есть прототип контроллера, который связан с этим виртуальным пространством: File - Open Sample World - samples - robotbenchmark - visual_tracking - visual_tracking.wbt. Цель контроллера состоит в двух разных задачах:

- детектировать объект изображения;
- повернуть и наклонить камеру так, чтобы объект был в центре изображения.

Сейчас в контроллере оценка положения реализована с помощью поиска объекта по цвету (контроллер ищет жёлтые участки, а потом принимает как гипотезу, что уточка - самый большой жёлтый участок). Искомый объект - уточка. В сцене есть ещё жёлтые объекты (жёлтый автомобиль), поэтому контроллер иногда ошибается.

Улучшение детектирования объекта может заключаться в следующем:

- улучшить алгоритм выделения жёлтых пикселей;
- применить морфологический фильтр (dilation и erosion) для удаления шума;
- использовать информацию из предыдущих фреймов для выбора наиболее подходящего участка изображения;
- использовать для поиска уточки методов: sift, surf.

Другие возможные стратегии:

- перевести изображения для поиска жёлтых объектов в другое цветовое пространство (например, HSL)
 - использовать ограничения на геометрическую фигуру уточки;
 - использовать алгоритмы фильтрации (фильтр Калмана, фильтр частиц) для предсказания положения уточки в следующем кадре
- (https://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html?highlight=kalman, https://docs.opencv.org/master/de/d70/samples_2cpp_2kalman_8cpp-example.html)

Улучшение качества слежения можно достигнуть если выбрать оптимальную скорость слежения, которая указывается в следующем коде:

```
panHeadMotor.setVelocity(-1.5 * dx / width)
tiltHeadMotor.setVelocity(-1.5 * dy / height)
```

Ссылка на соревнование для загрузки контроллера: https://robotbenchmark.net/benchmark/visual_tracking/. Советую изучить страничку, так как там есть информацию о решении этой задачи и можно посмотреть как работают существующие решения.

Контрольные вопросы

1. Что такое PID регулятор?
2. Опишите один из известных вам алгоритмов обнаружения объектов?
3. Зачем используется цветовое пространство HSV?

Лабораторная работа № 4. Программирование алгоритмов управления роботом в Webots

Цель: получение навыков работы с алгоритмами управления роботами

Краткие теоретические сведения

См. лекционный материал.

Задание

Вариант 1. 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.

Эталонная метрика t - это время, прошедшее с начала моделирования, причем наибольшее значение является лучшим. Измерение времени прекратится, как только маятник упадет вниз. Значение F отображает последнюю величину силы возмущения в ньютонах, которая была приложена к маятнику. Это значение силы будет линейно увеличиваться со временем.

Как повысить устойчивость маятника?

Пример программы на Python, управляющей роботом e-puck, использует ПИД-контроллер для установки скорости колес:

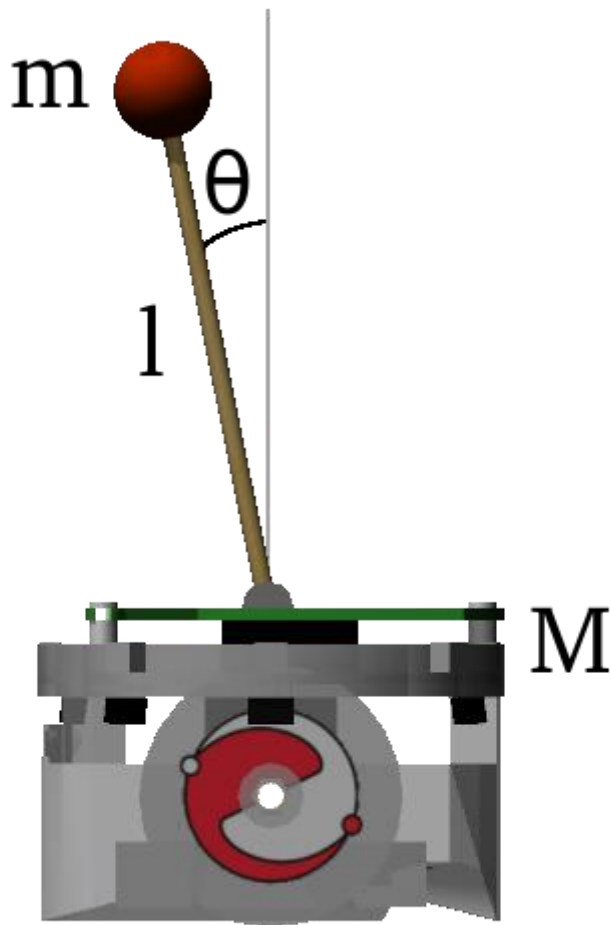
```
# PID control
integral = integral + (position + previous_position) * 0.5 / timestep
derivative = (position - previous_position) / timestep
speed = KP * position + KI * integral + KD * derivative
```

Значения коэффициентов ПИД-контроллера K_P , K_I и K_D могут быть настроены для улучшения стабильности маятника и достижения лучшего результата.

В качестве альтернативы, можно улучшить результат, используя более сложный алгоритм, например, основанный на полученных уравнениях движения перевернутого маятника.

Конфигурация системы:

- масса маятника: $m = 50$ г
- длина маятника: $l = 0.08$ м
- масса робота e-puck: $M = 160$ г



Вариант 2. Побег из ямы (Pit Escape)

Команда:

Литература:

https://link.springer.com/chapter/10.1007/978-3-030-49186-4_6

Этот тест направлен на разработку программы, которая управляет роботом ВВ-8, чтобы выбраться из ямы.

Робот должен выбраться из ямы как можно быстрее. Тест останавливается, если роботу требуется более одной минуты, чтобы выбраться.

Если робот смог выбраться из ямы, метрика будет основана на том, как быстро он это сделал. В противном случае, метрика будет измерять насколько близко он был от побега.

В первом случае метрика колеблется от 50% до 100% и линейно коррелирует с результатом времени. Значение 100% присуждается за мгновенный побег, в то время как значение 50% присуждается за побег в последнюю секунду. Во втором случае метрика колеблется от 0% до 50% и линейно коррелирует с расстоянием от вершины ямы.

Как повысить производительность?

Склон слишком крут, чтобы робот мог просто идти вперед. Вместо этого он должен двигаться вперед и назад, чтобы накапливать импульс, пока у него не будет достаточно сил, чтобы выбраться из ямы.

Базовый контроллер просто перемещается туда и обратно через регулярные промежутки времени. Вы должны попытаться найти лучший способ решить, когда менять направление, чтобы оптимизировать накопленный импульс. Например, вы можете использовать датчик гироскопа робота для измерения текущей скорости робота.

Гироскоп можно инициализировать с помощью следующего фрагмента кода:

```
gyro = robot.getGyro("body gyro")
gyro.enable(timestep)
```

Затем вы можете использовать

```
gyro.getValues()
```

для считывания текущего значения датчика.

Вариант 3. Pick and place

Этот тест направлен на разработку программы, которая управляет роботом KUKA youBot, чтобы выбрать куб и переместить его в целевую позицию.

Эталонная метрика - это время t , затраченное роботом на то, чтобы выбрать зеленый куб и поместить его в пустой слот коробки, расположенной на тележке. Таймер запускается сразу после начала моделирования и останавливается, когда куб помещается в слот. В этот момент время моделирования записывается как производительность робота.

У робота есть максимум 1 минута и 50 секунд, чтобы выполнить задачу, после этого оценка бенчмарка прекращается.

Как повысить производительность?

Базовый контроллер - это программа, которая просто устанавливает скорость или положение двигателей и ждет заданного времени, пока робот не достигнет желаемого положения. Существуют различные стратегии сокращения времени выполнения. Например, вы можете увеличить скорость двигателей, настроить продолжительность шагов и распараллелить движения.

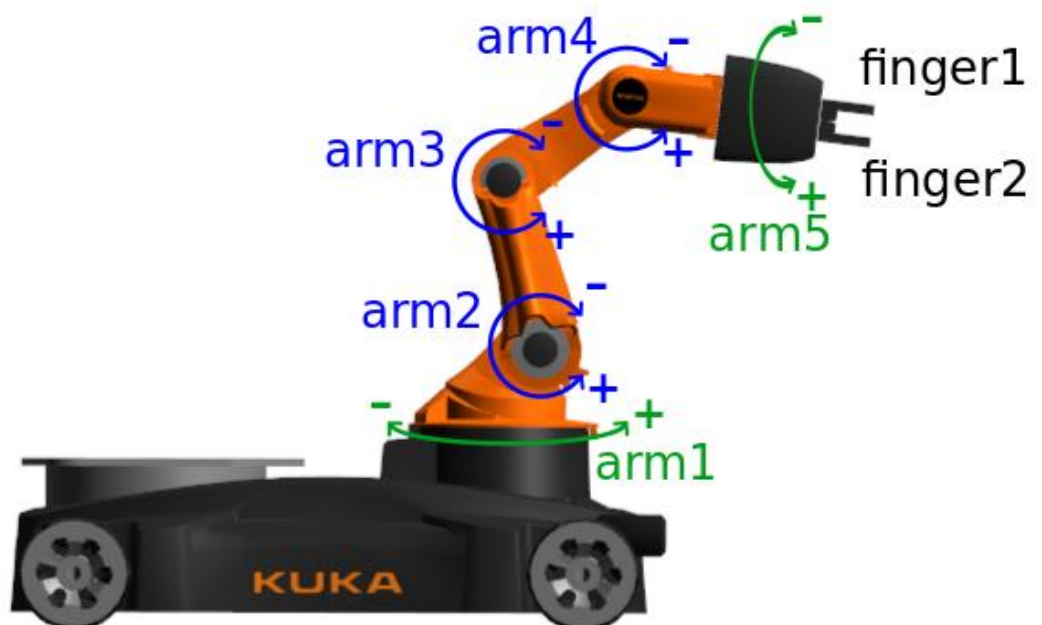
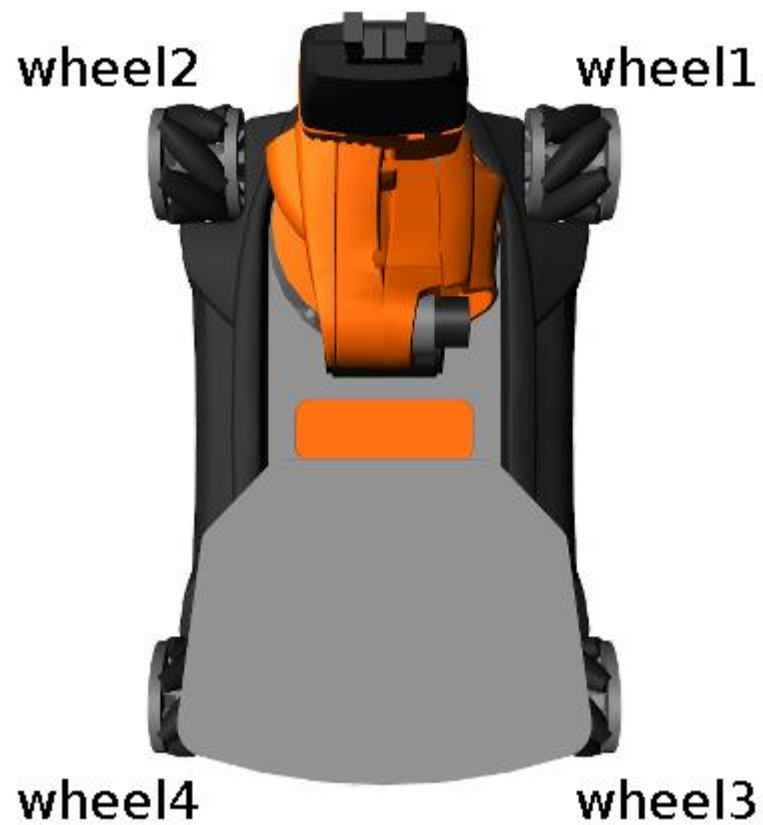
Вместо того, чтобы ждать заранее определенного промежутка времени между различными движениями, более эффективным решением является использование значений [датчика положения](#) для точного определения момента завершения движения. Если вы посмотрите на предоставленную программу на Python, управляющую роботом youBot, вы найдете пример кода для мониторинга положения одного из суставов руки:

```
# Monitor the arm joint position to detect when the motion is completed.
while robot.step(timestep) != -1:
    if abs(armPositionSensors[3].getValue() - (-1.2)) < 0.01:
        # Motion completed.
        break
```

Некоторая информация об окружающей среде и роботе, которая может быть полезна для улучшения контроллера:

- Начальное положение робота [x: -2.4, z: 0.0] м
- Положение куба [x: 1.0, y: 0.205, z: 0.0] м после остановки конвейерной ленты
- Целевое положение [x: -2.185, y: 0.140, z: -0.813] м

- Радіус колеса робота 5 см



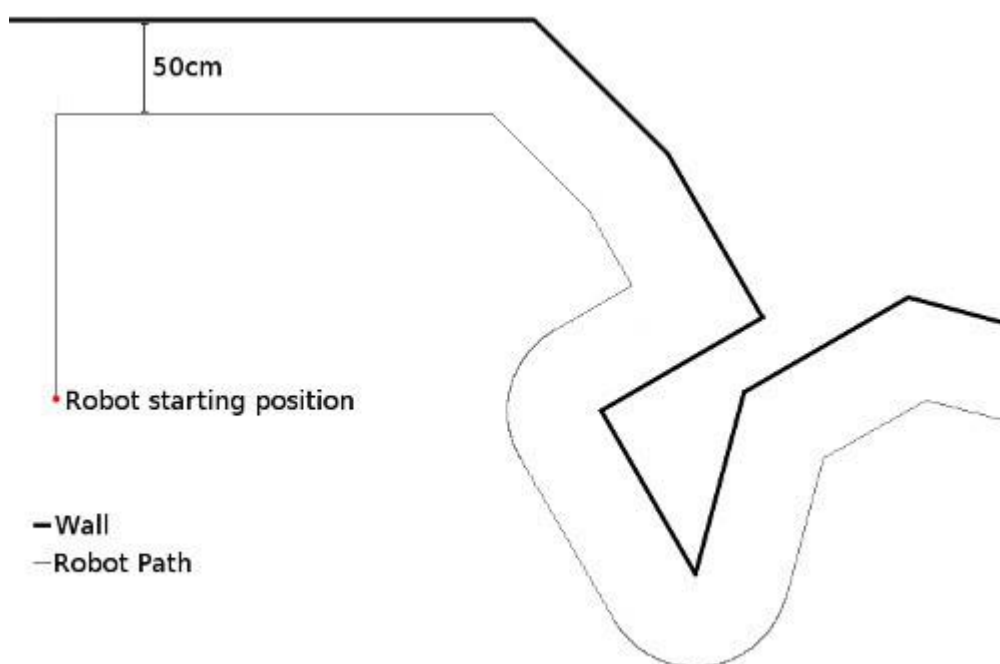
Вариант 4. Держись за стіну (Wall following)

Этот тест направлен на разработку программы управления роботом Pioneer, который должен следовать вдоль стены.

У робота есть ровно 1 минута, чтобы пройти как можно дальше, стараясь при этом держаться на расстоянии 50 см от стены.

Программа сначала вычислит "идеальный" путь. Первый отрезок этого пути представляет собой прямую линию, идущую от начального положения робота до ближайшей точки, которая находится на расстоянии 50 см от стены. Затем путь продолжается параллельно стене на расстоянии ровно 50 см.

Производительность вашего робота будет рассчитываться на основе среднего расстояния до этого "идеального" пути. Обратите внимание, что робот всегда находится лицом к стене в своем исходном положении, поэтому первая часть заключается в движении вперед, пока стена не окажется на расстоянии 50 см.



Кроме того, если робот попытается сократить путь, он получит штраф за каждый пропущенный сегмент. В любой момент времени сегмент пути, ближайший к роботу, считается текущим сегментом, и любой сегмент, предшествующий текущему сегменту, который не был посещен, будет считаться пропущенным.

Как повысить производительность?

Контроллер (на Python) основан на значениях, полученных с датчиков гидролокатора робота.

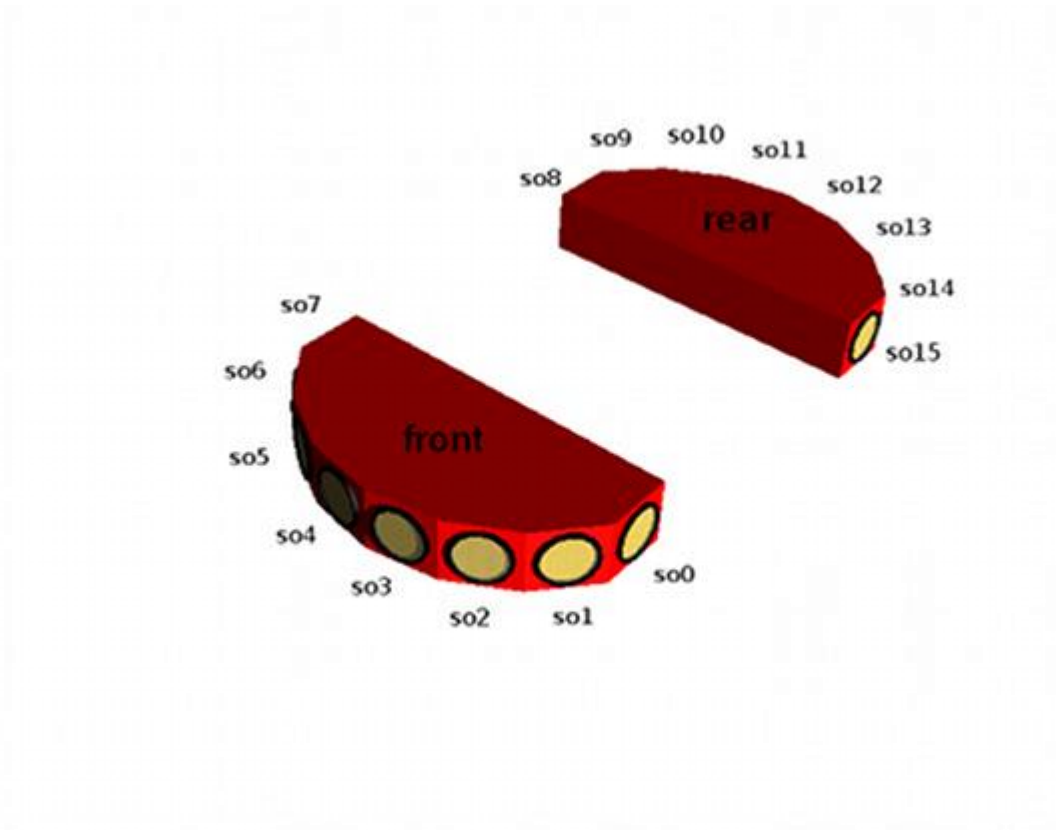
Базовый контроллер чрезвычайно прост и использует только 2 датчика, доступных на роботе. Из-за этого, а также из-за того, как работают датчики гидролокатора (см. ниже), он очень плохо справляется с резкими поворотами и в основном будет работать только до тех пор, пока стена почти прямая.

Первым простым улучшением было бы рассмотреть ориентацию робота относительно стены, например, если вы находитесь слишком далеко от стены, но робот уже направляется к стене, вам не нужно изменять курс, дополнительно поворачивая. Для этого вы можете использовать тот факт, что робот имеет по 2 датчика с каждой стороны, и использовать разницу между ними для оценки ориентации.

Во вторых вам нужно учитывать значения передних датчиков, иначе робот не сможет обнаружить более резкий угол в стене и, скорее всего, столкнется с ней.

Также может быть полезно определить, когда робот ударился о стену, и заставить робота отступить назад, когда это произойдет.

Датчики Pioneer 3-DX имеют следующую конфигурацию:



Существует угол 20° между соседними датчиками, за исключением боковых датчиков (so0, so7, so8 и so15), которые имеют угол 40° . Максимальная дальность действия датчиков составляет 5 м.

Поскольку датчики Pioneer 3-DX являются гидролокаторами, они могут обнаружить препятствие только в пределах угла падения, который в данном случае составляет $22,5^\circ$. Другими словами, лучи гидролокатора, лежащие вне конуса отражения апертуры 45° , не улавливаются гидролокатором, что делает препятствие "невидимым" для этого конкретного датчика.

Вариант 5. Марафон гуманоидов (Humanoid Marathon)

https://robotbenchmark.net/benchmark/humanoid_marathon/

<https://cyberbotics.com/doc/guide/robotis-op2>

Этот тест направлен на разработку компьютерной программы, которая управляет гуманоидным роботом, чтобы пройти как можно дальше, учитывая ограниченное время автономной работы. Язык программирования - Python, модель гуманоидного робота - ROBOTIS OP2.

Окружающая среда представляет собой масштабную модель, основанную на размере робота.

Эталонная метрика - это расстояние, пройденное роботом.

Расстояние - это разница между начальным и конечным положением робота на мировой оси x, параллельной улице.

Если робот падает или пытается встать на четвереньки, тест немедленно завершается.

Как заставить ROBOTIS OP2 преодолевать большее расстояние?

Если вы посмотрите на программу на Python, управляющую роботом ROBOTIS OP2, вы увидите, что эта программа использует [RobotisOp2GaitManager](#), чтобы заставить робота ходить:

```
# At the beginning, start walking on the spot.
```

```
# After 45 timesteps, begin taking steps forward.
```

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

```
    if looptimes == 45:
```

```
        gaitManager.setXAmplitude(gaitAmplitude)
```

```
        gaitManager.step(basicTimeStep)
```

```
        looptimes += 1
```

Параметры менеджера походки настраиваются, и можно найти настройку, которая обеспечивает лучшую производительность. Например, настройка амплитуды походки или отключение функции баланса заставляет робота преодолевать большее расстояние.

Вариант 6. Движение по шоссе

Обгон автомобилей по шоссе.

https://robotbenchmark.net/benchmark/highway_driving/

Этот тест направлен на то, чтобы проехать как можно дальше за одну минуту, не сталкиваясь с другими транспортными средствами или статическими объектами.

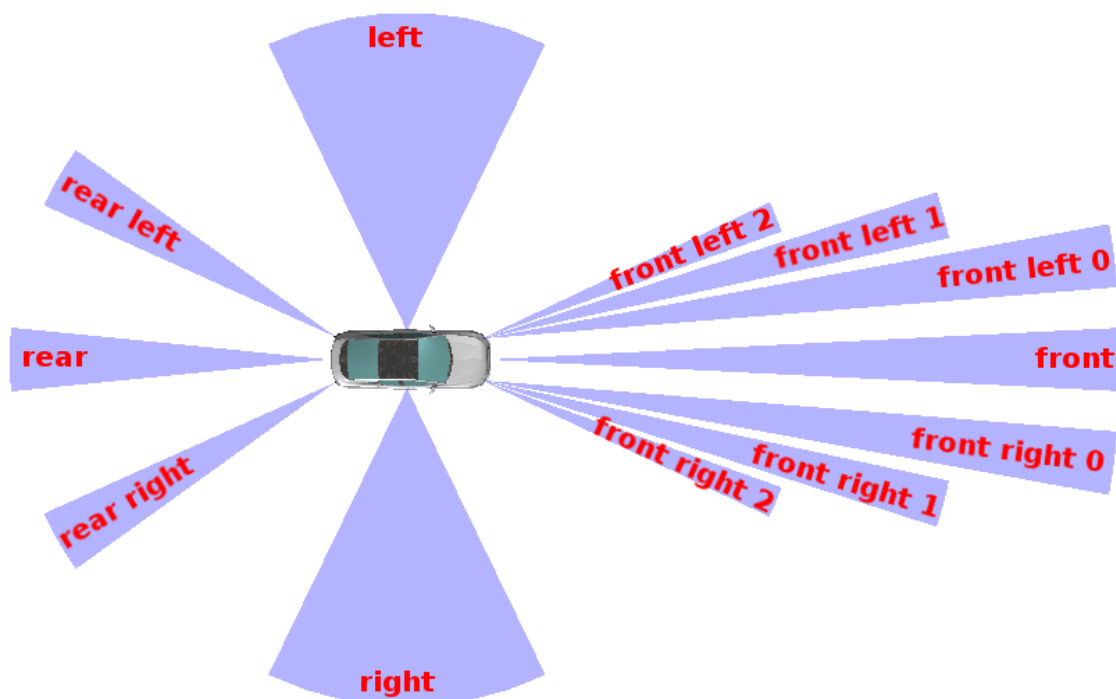
Метрика, используемая для оценки контроллера транспортного средства, - это расстояние, пройденное по дороге за одну минуту.

Если транспортное средство касается каких-либо других транспортных средств или статических объектов или выезжает на аварийную полосу, моделирование останавливается (даже до одной минуты) и записывается уже пройденное расстояние.

Как повысить производительность?

Базовый контроллер - это простая программа, которая заставляет автомобиль двигаться прямо и снижать скорость, если впереди обнаружен другой автомобиль, чтобы сохранить дистанцию безопасности.

Вместо снижения скорости можно использовать датчики бокового расстояния, чтобы проверить, можно ли изменить полосу движения для обгона (вам разрешено обгонять с правой стороны). Вот доступные показатели [датчика расстояния](#):



Обратите внимание, что через некоторое время дорога становится извилистой, поэтому вам нужно будет учитывать направление дороги. Простой способ сделать это - использовать боковые датчики расстояния, чтобы узнать расстояние до правого и левого барьеров. Более эффективным способом было бы использовать [камеры](#) для обнаружения дорожных линий.

В дополнение к датчикам расстояния, автомобиль оснащен [GPS](#), расположенным в центре осей задних колес, и фронтальной камерой (с возможностью [распознавания](#) объектов).

Вариант 7. Спринт гуманоидов (Humanoid Sprint)

https://robotbenchmark.net/benchmark/humanoid_sprint

Этот тест направлен на разработку компьютерной программы, которая управляет гуманоидным роботом, чтобы он как можно быстрее шел по 10-метровой дорожке. Язык программирования - Python, модель гуманоидного робота - Aldebaran NAO.

Эталонный показатель - это время t , затраченное роботом на преодоление 10-метрового расстояния. Секундомер запускается, как только начинается моделирование. Когда робот проходит перед секундомером, датчик обнаруживает присутствие робота и останавливает таймер. В этот момент время, отображаемое на секундомере, записывается как производительность робота.

Если робот падает или пытается встать на четвереньки, к таймеру добавляется штраф в размере 30 секунд. Такая ситуация обнаруживается, если вертикальное положение центра его тела опускается ниже 20 см.

У робота есть максимум 1 минута и 50 секунд, чтобы преодолеть 10-метровое расстояние, после чего оценка прекращается.

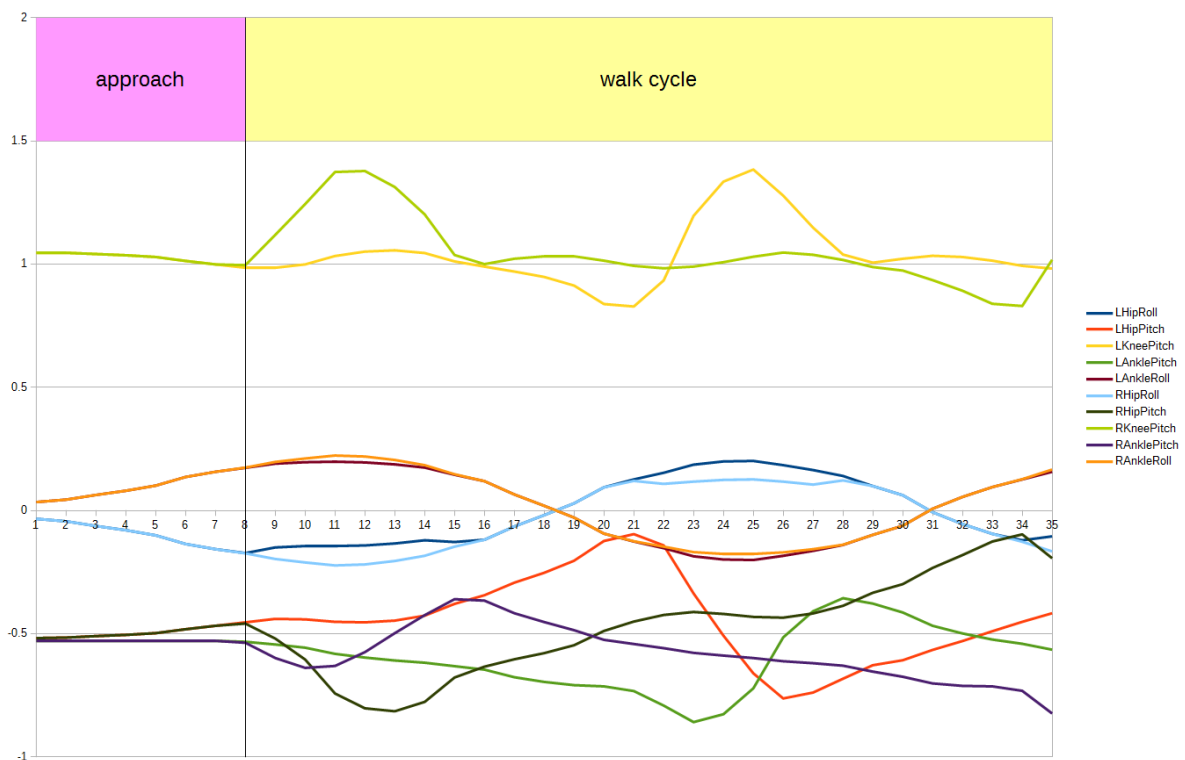
Как заставить робота NAO ходить быстрее?

Если вы посмотрите на программу на Python, управляющую роботом NAO, вы увидите, что эта программа просто воспроизводит файл движения под названием forward.motion:

```
def run(self):  
  
    walk = Motion('forward.motion')  
  
    walk.setLoop(True)  
  
    walk.play()  
  
    while True:  
  
        if walk.getTime() == 1360:  
  
            walk.setTime(360)  
  
            if self.step(self.timeStep) == -1:  
  
                break
```

Движение длится 1360 миллисекунд (1,36 секунды) и делится на две части: первая часть длится 360 мс и перемещает робота из исходного положения в позу, соответствующую началу цикла ходьбы. Вторая часть - это цикл ходьбы, который длится 1000 мс, а затем повторяется. Поэтому контроллер проверит, когда движение закончится (время движения достигает 1360 мс), и перезапустит цикл ходьбы движения, т. е. сбросит время движения до 360 мс, чтобы начать новый цикл ходьбы. Это будет повторяться снова и снова, так что робот будет ходить вечно.

Файл движения содержит положения суставов (в радианах) для каждого сочленения ног, которые используются при ходьбе:



Файл forward.motion можно открыть в программе, такой как LibreOffice Calc или Excel, изменить и сохранить снова.

Для того, чтобы еще больше улучшить скорость ходьбы, вы можете попробовать:

- Еще больше сократите фазу подготовки;
- Измените временной шаг файла движения, чтобы сделать его быстрее (например, уменьшите временной шаг на 40 миллисекунд);
- Измените значения файла движения вручную;
- Двигайте руками робота (спереди, чтобы раньше попасть в датчик финиша, или балансируя, чтобы попытаться идти быстрее);
- Создайте уравнения для каждого сустава, используя математическую функцию синуса (или сплайны), которые аппроксимируют текущие траектории. Затем вы сможете управлять каждым двигателем индивидуально с помощью контроллера на Python, используя свои уравнения (вам больше не понадобится файл forward.motion). Вместо этого вы будете использовать непосредственно метод setPosition() для каждого двигателя. Это позволит вам точно настроить параметры уравнений, чтобы попытаться улучшить движение;
- Используйте гироскоп и акселерометр робота для создания замкнутого контура движения с использованием передовых методов управления;
- Используйте передовые исследования, такие как Zero Moment Point (ZMP), Central Pattern Generator (CPG) или генетические алгоритмы (GA), чтобы создать более быструю прогулку.

Контрольные вопросы

1. Что такое PID регулятор?

2. Опишите один из известных вам алгоритмов обнаружения объектов?
3. Зачем используется цветовое пространство HSV?