Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет «Московский институт электронной техники»

Лабораторные работы по курсу «Робототехника» для 11 класса

Москва 2021

Лабораторная работа №1 «Реализация алгоритма движения робота по траектории»

Цель

1. Ознакомление с интерфейсом 3D-симулятора разработки роботизированных средств Webots;
2. освоение теоретический сведений по разработке алгоритмов движения по траектории;
3. реализация алгоритма движения робота e-Puck по заданной траектории на языке Python.

Краткие теоретические сведения

Знакомство с 3D-симулятором разработки роботизированных средств Webots

Webots - это многоплатформенное настольное приложение с открытым исходным кодом, используемое для моделирования работы роботизированных систем. Оно предоставляет собой полную среду разработки для программирования и моделирования поведения роботов.

Webots был разработан для профессионального использования и широко используется в промышленности, образовании и исследованиях. Cyberbotics Ltd. непрерывно поддерживает Webots в качестве основного продукта с 1998 года.

Установка программного обеспечения Webots

Для того, чтобы установить Webots на компьютер, необходимо:

1. Перейти по ссылке на сайт Cybertronics: https://cyberbotics.com/.
2. Нажать на кнопку «Download» в центре экрана (рисунок 1), предварительно выбрав версию Вашей операционной системы (если требуется).

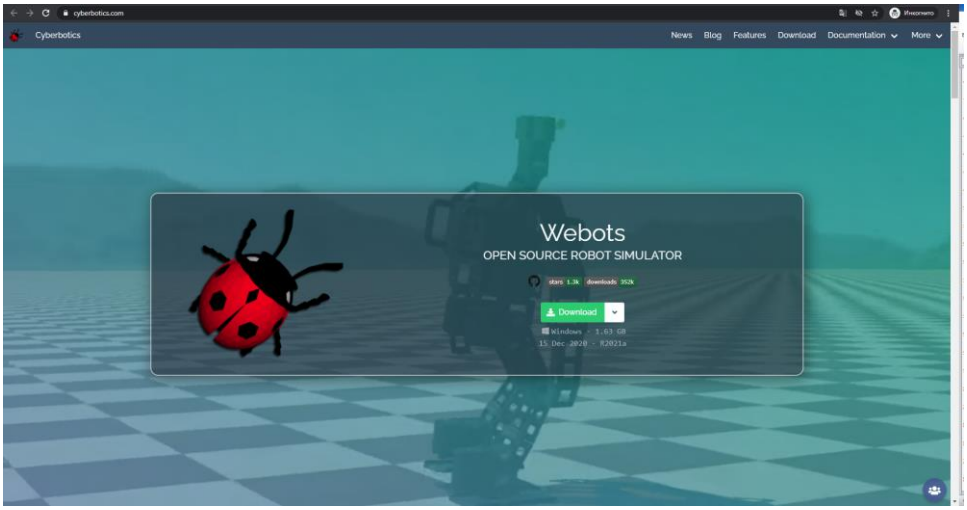


Рисунок 1. Главная страница сайта Cybertronics.

3. Установить Webots на Ваш компьютер.
4. Добавить установленную на Ваш компьютер текущую версию Python (не ниже Python 3.7) в переменную окружения PATH.

Python в качестве языка программирования в 3D-симуляторе Webots

Первым же делом следует сказать, что Python выбран в качестве языка программирования в среде Webots прежде всего с целью упрощения задачи алгоритмизации за счет простоты синтаксиса и легкого восприятия кода. Данный курс не ставит главной задачей обучить основам программирования в микропроцессорной технике, поэтому целесообразней обеспечить учащихся простым инструментом для реализации алгоритмов управления и обработки данных. Python является самым популярным языком для обучения основ объектно-ориентированному программированию. Безусловно, описание алгоритмов в среде Python имеет ряд функциональных преимуществ, нацеленных на упрощение базовых задач алгоритмизации, что, несомненно, имеет весомую роль в разработке систем управления в робототехнике. Язык максимально раскрывает свой потенциал в решении задач адаптивного и интеллектуального видов управления.

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и

читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой набор полезных функций.

Python поддерживает структурное, обобщенное, объектно-ориентированное, функциональное и аспектно-ориентированное программирование.

Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Создание мира симуляции в Webots

Открыв 3D-симулятор Webots впервые, Вы попадаете на пример-демонстрацию работы симулятора. Прежде, на предстоит разобраться с интерфейсом программы и начнем мы с создания нового (собственного) проекта. Чтобы это сделать, нажмите на вкладку «Wizards», на панели-ленте в верхней левой части экрана, как на рисунке 2 и выбираем «New Project Directory».

Рисунок 2. Создание собственного проекта.

В контекстном меню нажимаем «Next» и в следующем окне определяем, где будет размещаться директория с файлами мира симуляции и называем файл «Lab_work_1_*Ваша фамилия латиницей*», нажимаем «Next» для перехода в следующее окно, как на рисунке 3.

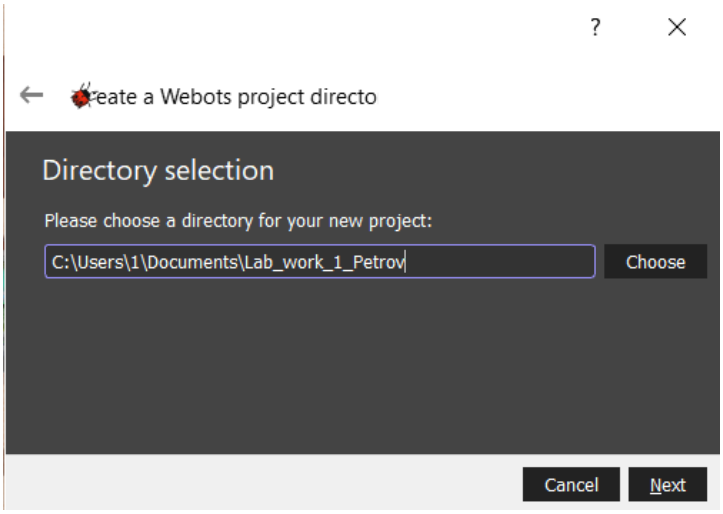


Рисунок 3. Название проекта.

Затем переходим в окно конфигурации нового мира симуляции, отмечаем последнюю (нижнюю) галочку «Add a rectangle arena», что в последствии добавит в новый мир сразу объект пола, по которому может передвигаться робот, как на рисунке 4. Называем мир «Lab_1_world», нажимаем «Next» и в следующем окне, в котором отображаются все создаваемые файлы – «Finish».

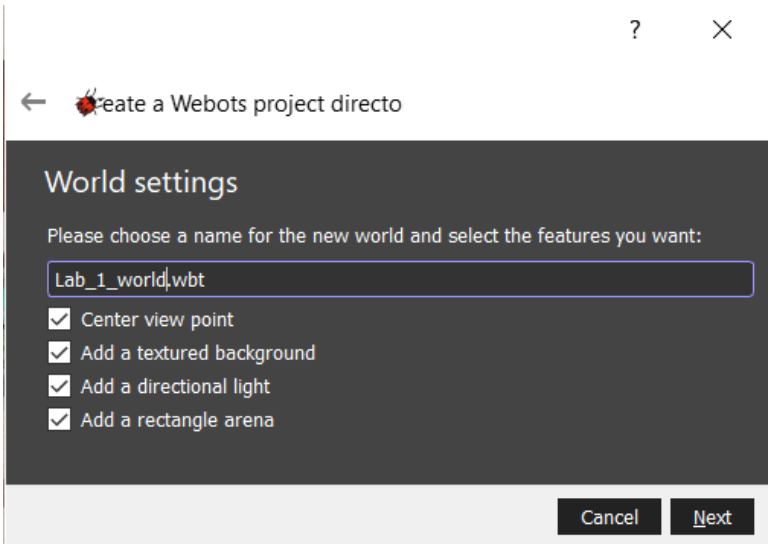


Рисунок 4. Создание мира симуляции.

Итак, перед нами пустой проект с созданной ареной – рисунок 5. Обратим внимание на дерево объектов в левой части экрана. Там мы впоследствии произведем некоторые манипуляции по настройке созданного мира и добавлению робота для программирования. В правой части экрана есть текстовый редактор, необходимый для программирования. В настоящее время он пуст.

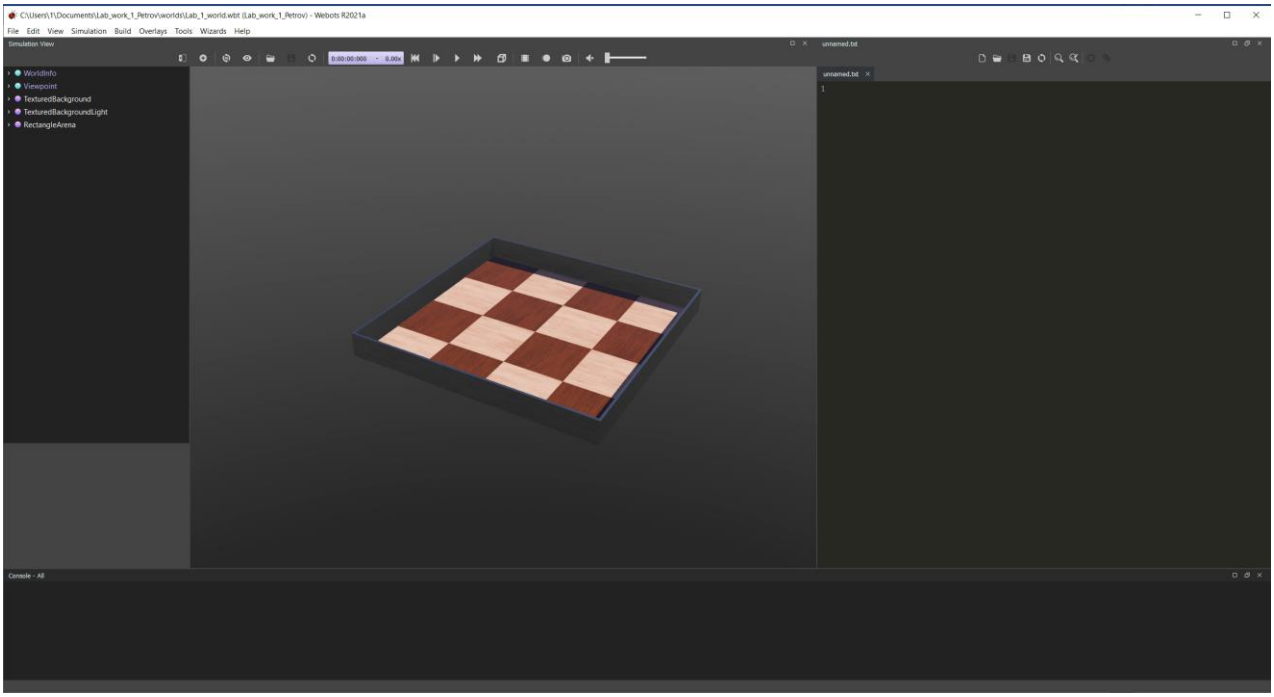


Рисунок 5. Пустой созданный мир симуляции.

Произведем настройку размеров прямоугольной арены. Для этого откроем в дереве объектов выпадающий список «RectangleArena» (рисунок 6), и увидим, какие параметры имеет данный объект. Все линейные размеры имеют размерность в метрах. Угловые (rotation) – в радианах. Поля размером метр на метр - более, чем достаточно для решения задач по реализации алгоритмов в рамках данной лабораторной работы.

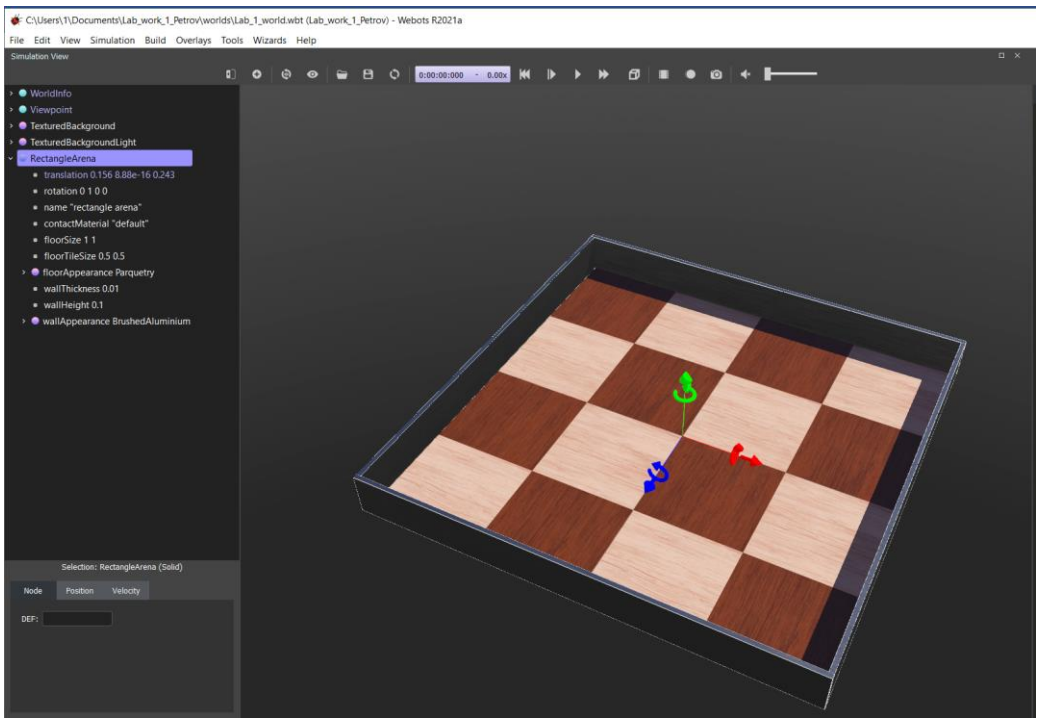


Рисунок 6. Настройка поля RectangleArena.

Допустим, поменяем другой параметр, например величину клетки шахматной разметки. Для этого выберем поле «FloorTileSize» и уменьшим каждое значение в два раза, так чтобы каждая клетка была размером 0,25 на 0,25 м. как на рисунке 7.

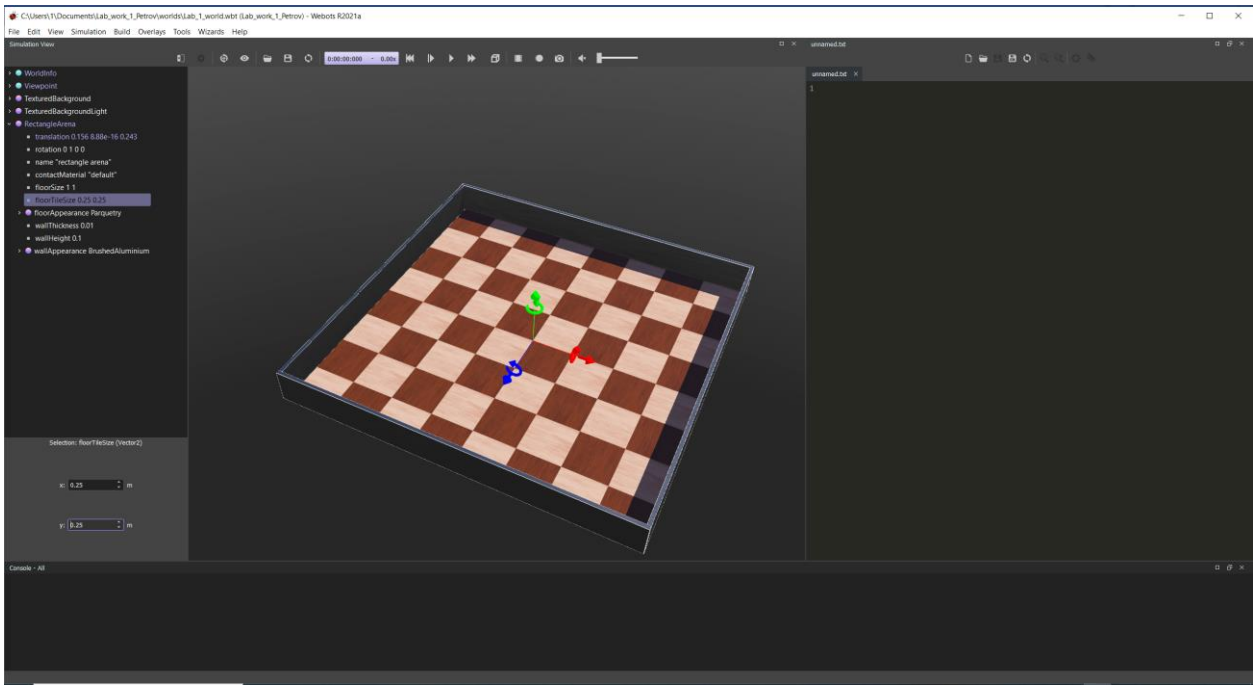


Рисунок 7. Настройка размеров клетки шахматной разметки.

Далее, добавим объект «Box» - коробку, как пример физического статического объекта в симуляторе. Для этого нажмем ЛКМ на элемент списка в дереве объектов (например, на вкладку «Viewpoint»), тогда нам станет доступна возможность добавлять новые объекты в среду. Нажмем сочетание клавиш (Ctrl+Shift+A), или нажав ПКМ в свободной части дерева объектов и выберем «Add New» (также это можно сделать, нажав на специальный значок, графически схожий со знаком «плюс»  на панели-ленте в верхней части экрана). Появится окно, как на рисунке 8. В этом окне нажимаем на стрелочку рядом с «PROTO nodes (Webots Projects)» и в выпадающем списке выбираем «objects». Таким же образом добавляются все объекты в среду мира симуляции. Также, можно воспользоваться поиском в поле «Find».

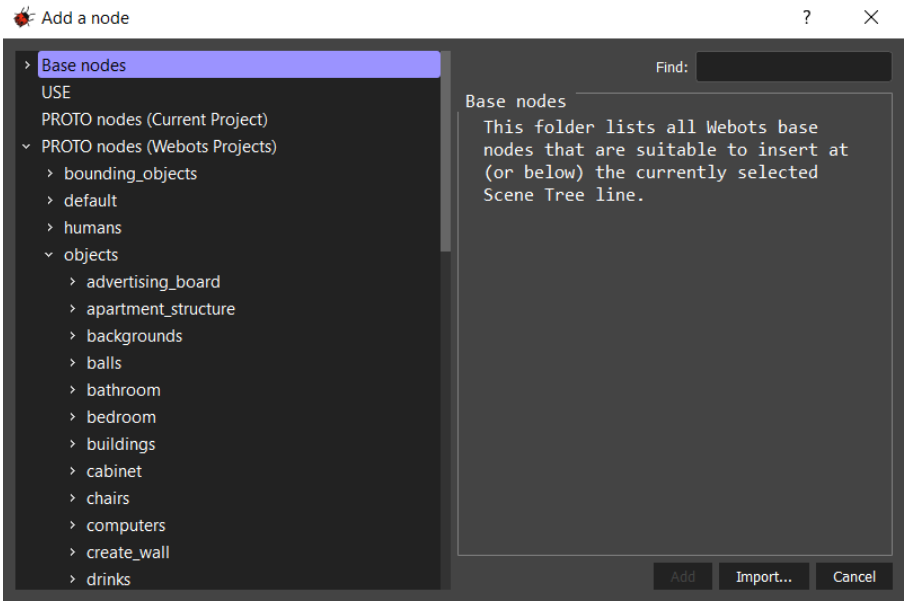


Рисунок 8. Добавление объекта.

В этом списке найдем поле «factory», вложенный список «containers» и внутри списка выбираем объект «WoodenBox». Чтобы добавить выбранный объект, нажмите «Add», как на рисунке 9.

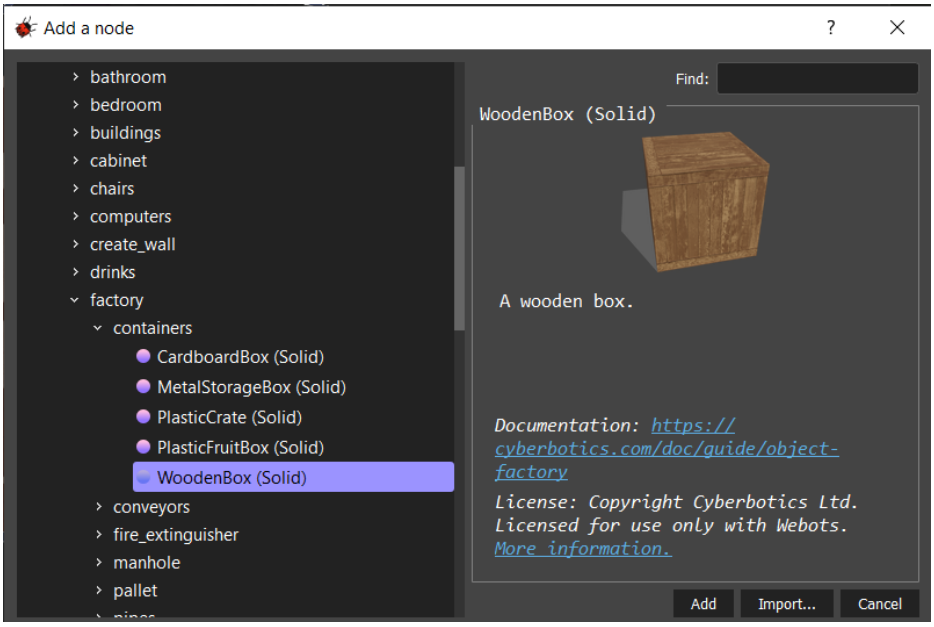


Рисунок 9. Добавление объекта «WoodenBox» в среду.

Объект успешно добавлен в среду симуляции, что отражено на рисунке 10. Теперь нам необходимо его настроить, а именно – его габариты и массу.

Симулятор Webots имеет симуляцию физики. В частности, здесь реализована гравитация, действие законов Ньютона и, даже, оптические

эффекты. Именно, в связи с этим, необходимо уделять внимание настройке параметров объектов, так как они могут взаимодействовать друг с другом.

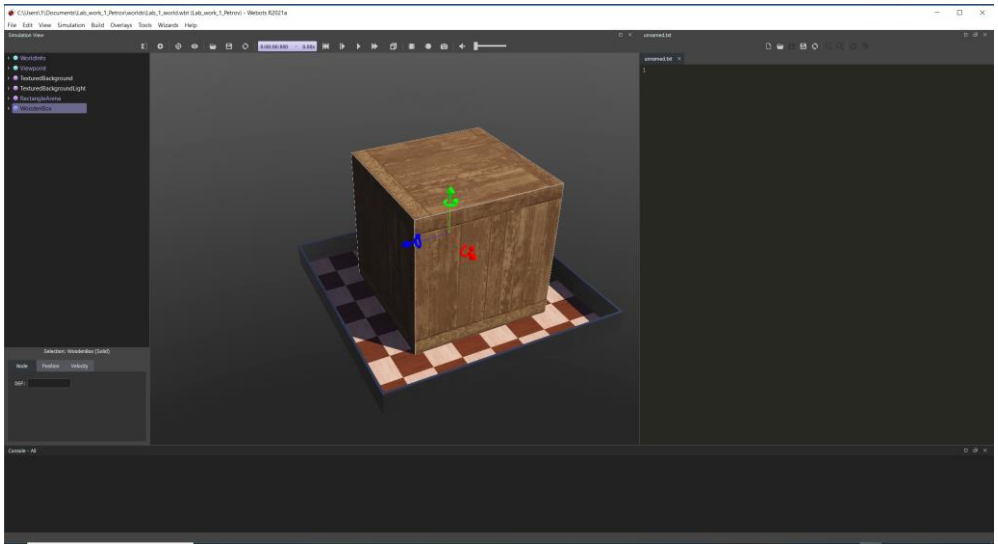


Рисунок 10. Добавленный объект «WoodenBox».

Настроим объект так: размеры (поле «size») по осям: 0.1x0.1x0.1, масса (поле «mass») – 1 кг, первоначальное положение (поле «translation» по осям должно быть: 0x0.05x0).

Если масса будет равна нулю, то в соответствии со вторым законом Ньютона к данному объекту не может быть приложена ни одна сила (объект жестко заделан к месту его расположения на полу). Должно получиться как на рисунке 11.

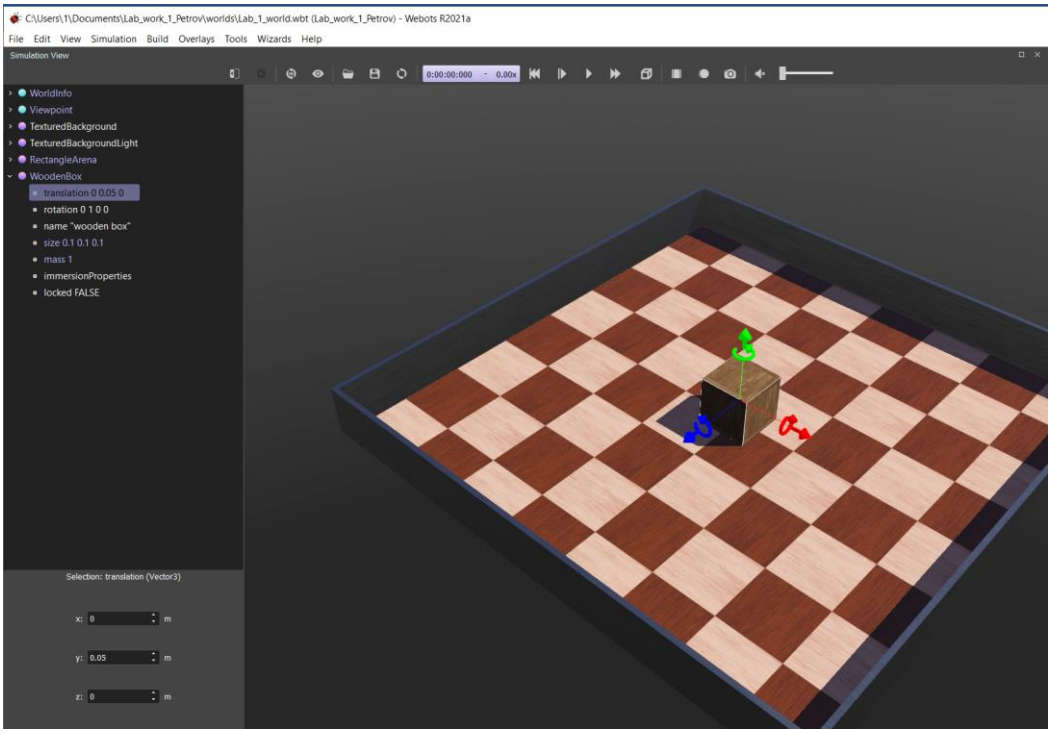


Рисунок 11. Настройка объекта «WoodenBox».

Перемещать объекты можно с помощью мыши, предварительно зажав клавишу Shift, а также, с помощью стрелок по осям, исходящим из выбранного объекта. Копировать и вставлять объекты – с помощью сочетания клавиш (Ctrl+C, Ctrl+V).

Переместите коробку в сторону от центра, чтобы проверить инструкцию.

Нажмите сочетание клавиш (Ctrl+Shift+S), или нажмите на иконку, графически схожую с дискетой, в левой верхней части экрана для сохранения созданного мира.

Добавление робота e-puck в Webots

Теперь, добавим робота e-puck в созданный ранее мир симуляции. Аналогично добавлению любого объекта образом, представленным выше (рисунок 9), добавим робота по следующему пути поиска:

PROTO nodes (Webots Projects) -> robots -> gctronic -> e-puck -> E-puck (Robot)

Должно получиться, как на рисунке 12.

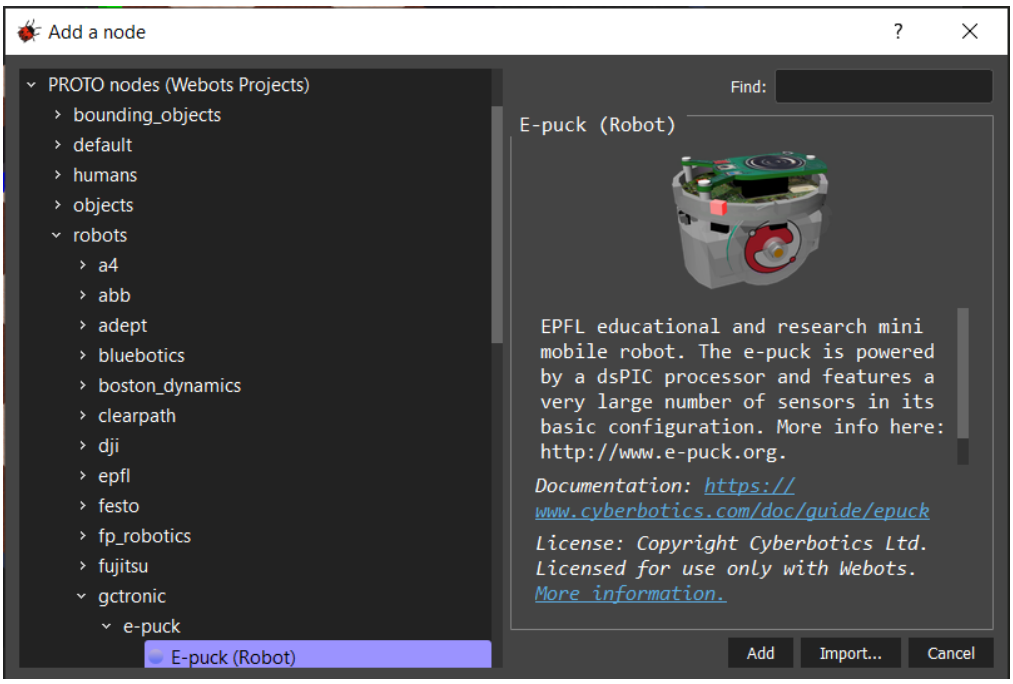


Рисунок 12. Добавление робота e-puck в среду симуляции.

Обратите внимание, что в описании робота (рисунок 12) имеется ссылка на документацию на робота – это очень важно для нас в будущем, так как со страницы этой документации будет возможно узнать параметры, необходимые для программирования робота.

Добавив робота, сразу посмотрим на программу-контроллер (поле «controller») – прошивку робота (в дальнейшем просто - контроллер) – программу, которую будет исполнять робот. В данный момент, используется контроллер «e-puck_avoid_obstacles».

Если запустить симуляцию, нажав сочетание клавиш (Ctrl+2), или кнопку «Run the simulation in real-time» в верхней части экрана по центру (Рисунок 13), мы увидим, как робот начинает передвигаться по арене, избегая врезания в препятствия.

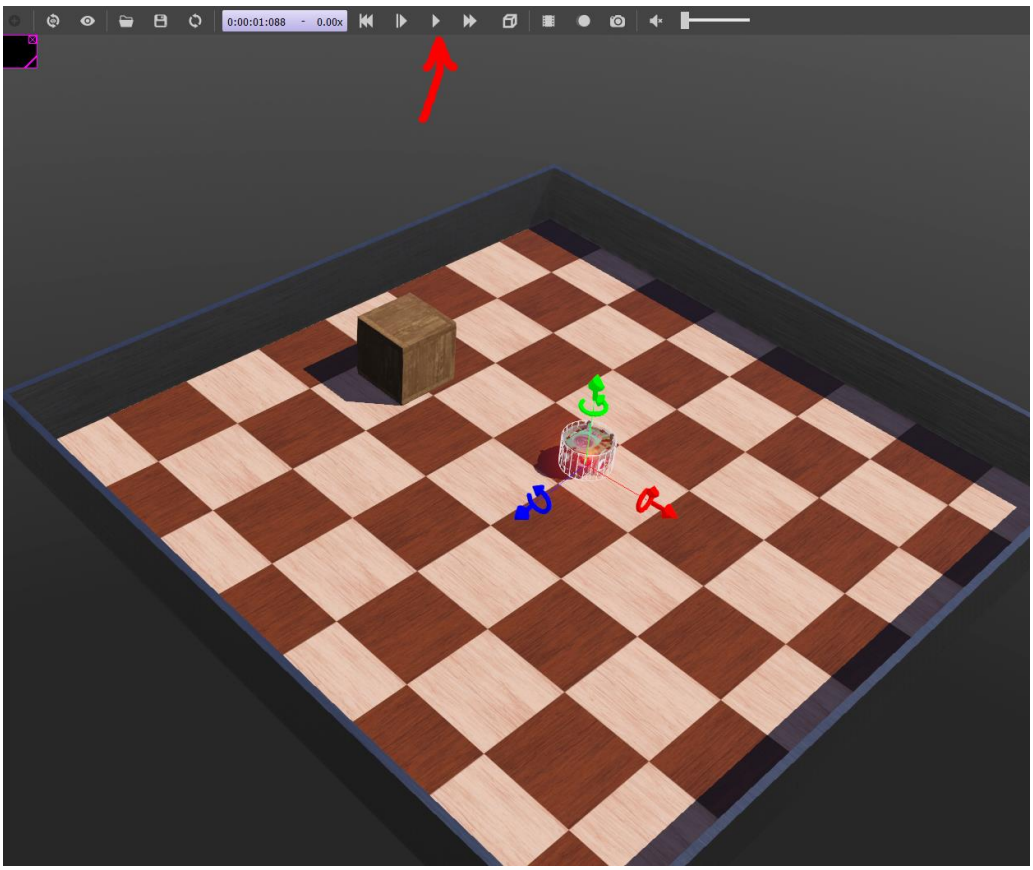


Рисунок 13. Пример симуляции контроллера «e-puck_avoid_obstacles».

Создание программы-контроллера на языке Python

Чтобы добавить контроллер, необходимо выбрать в ленте вкладку «Wizards», и нажать на вкладку в выпадающем меню «New Robot Controller…», как на рисунке 14.

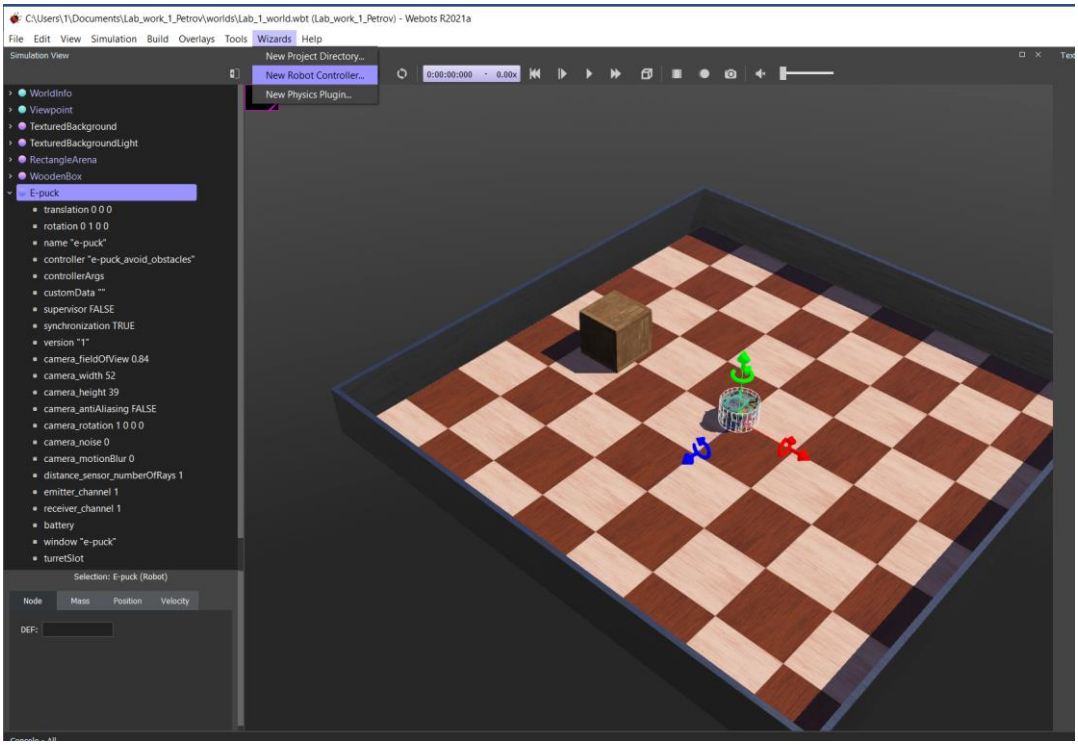


Рисунок 14. Создание нового контроллера.

Откроется диалоговое окно, нажмите «Next», чтобы продолжить. Следующая страница требует выбора языка программирования – выбираем Python и нажимаем далее (Next). В следующем окне – называем контроллер так: «Lab_work_1_controller_*Ваша фамилия латиницей*» и переходим далее.

Затем обязательно ставим галочку около «Open ‘Lab_work_1_controller_*Ваша фамилия латиницей*.py’ in Text Editor», чтобы открыть текст контроллера в редакторе, как на рисунке 15, и нажимаем «Finish».

Рисунок 15. Завершение создания собственного контроллера.

Теперь необходимо подключить контроллер к роботу e-puck, чтобы мы смогли сразу тестировать написанные программы.

Для этого необходимо выбрать в дереве объектов робота, нажав ЛКМ на поле с названием робота (E-puck), найти поле «Controller» и нажать кнопку «select…», как на рисунке 16.

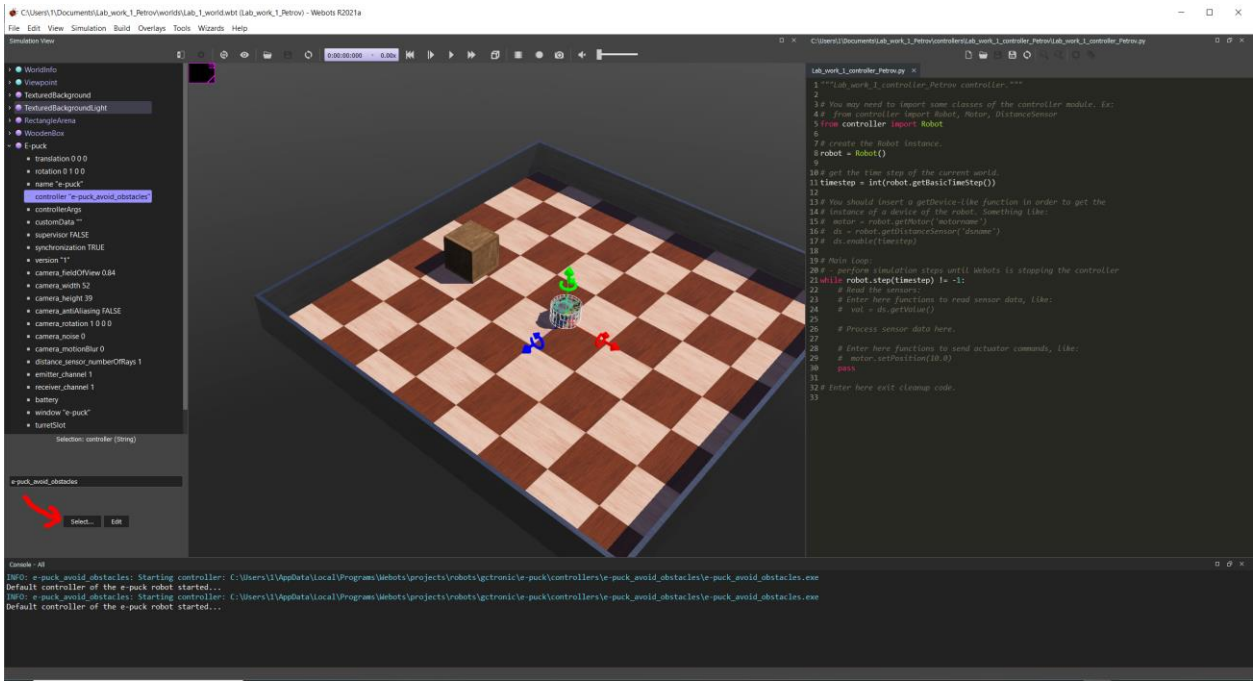


Рисунок 16. Подключение собственного контроллера.

В диалоговом окне выбираем только что созданный контроллер с именем «Lab_work_1_controller_*Ваша фамилия латиницей*» (рисунок 17) и нажимаем «ОК».

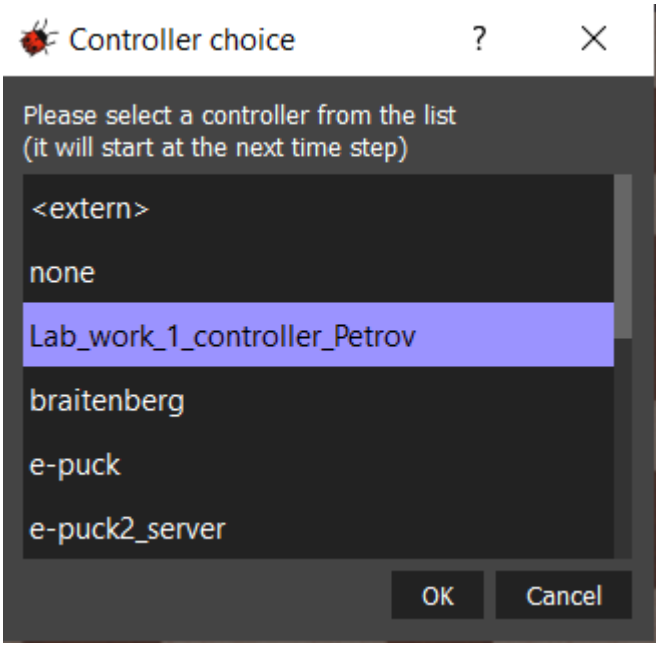


Рисунок 17. Подключение собственного контроллера. Диалоговое окно.

Обязательно сохраним файлы, нажав сочетание (Ctrl+Shift+S), чтобы не потерять настройки и добавления.

Разработка программы движения по траектории на языке Python

Приступим к созданию первой программы, описывающую движение робота e-puck по траектории на языке Python.

Когда Вы создали контроллер, Вы увидите скелет код-листинга программы в правой части экрана (см. рисунок 16). Давайте рассмотрим его поподробнее и подключим элементы управления моторами в программе (рисунок 18). Без них движение робота в пространстве неосуществимо.

Lab_work_1_controller_Petrov.py*

```
"""Lab_work_1_controller_Petrov controller."""

# You may need to import some classes of the controller module. Ex:
#  from controller import Robot, Motor, DistanceSensor
from controller import Robot

# create the Robot instance.
robot = Robot()

# get the time step of the current world.
timestep = int(robot.getBasicTimeStep())

# You should insert a getDevice-like function in order to get the
# instance of a device of the robot. Something like:
#  motor = robot.getMotor('motorname')
#  ds = robot.getDistanceSensor('dsname')
#  ds.enable(timestep)

# Main loop:
# - perform simulation steps until Webots is stopping the controller
while robot.step(timestep) != -1:
    # Read the sensors:
    # Enter here functions to read sensor data, like:
    #  val = ds.getValue()

    # Process sensor data here.

    # Enter here functions to send actuator commands, like:
    #  motor.setPosition(10.0)
    pass

# Enter here exit cleanup code.
```

Рисунок 18. Автоматически сгенерированный код-листинг программы-контроллера.

Добавим немного пояснений в комментариях к код-листингу (Рисунок 19). Комментарии обозначаются символом «#», после которого вся оставшаяся часть строки является текстом комментария и не исполнятся компилятором.

Lab_work_1_controller_Petrov.py

```python
"""Lab_work_1_controller_Petrov controller."""

# Вам может потребоваться импортировать некоторые классы модуля контроллера. Пример:
# from controller import Robot, Motor, DistanceSensor
# from math import fabs, pi - добавляем из библиотеки math функцию абсолютного значения числа и число Пи
from controller import Robot
from math import pi

# Создаём обьект класса Robot
robot = Robot()

# Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ
timestep = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит тип данных в целочисленный

# Создадим экземпляры левого и правого моторов.*(2) Название движков и устройств берется из документации
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

# Зададим поворот колес на определнный угол (Пи радиан - т.е. полоборота)относительно поворота в 0 радиан
leftMotor.setPosition(pi)
rightMotor.setPosition(pi)
# Main loop: - бесконечный цикл
# -выполняет шаги симуляции, пока Webots не остановит контроллер
while robot.step(timestep) != -1:
    # Здесь можно читать данные с сенсоров, исполнять бесконечно циклирующиеся команды
    pass #команда - заглушка, ничего не делает
```

Рисунок 19. Код-листинг с комментариями.

По выполнении данной немного модифицированной от изначальной программы, робот проедет на расстояние πR, где R – радиус колеса (т.е. сделает половину оборота). Значения радиуса, габаритов, а также компонентов робота, необходимых для инициализации можно найти по ссылке: https://cyberbotics.com/doc/guide/epuck?version=master

Функция «setPosition(*вещественное число радиан*)» принимает в аргумент угол, на который нужно повернуться мотору, относительно 0 радиан.

Выполнение шага моделирования — это атомарная операция: его нельзя прервать. Следовательно, измерение датчика или приведение в действие двигателя может происходить только между двумя шагами симуляции. По этой причине шаг управления, указанный с каждым вызовом функции robot.step(timestep), должен быть кратным шагу симуляции. Так, например, если шаг симуляции составляет 32 мс, тогда аргумент тика (шага) контроллера, переданный в функцию robot.step(TIME_STEP), может быть 32, 64, 128 и т. д.

Именно для этого и служит функция:

timestep = int(robot.getBasicTimeStep())

Переменная TIME_STEP используется для хранения возвращаемого значения этой функции – число миллисекунд одного шага симуляции. Такой принцип объясняется в блок-схеме на картинке 20.

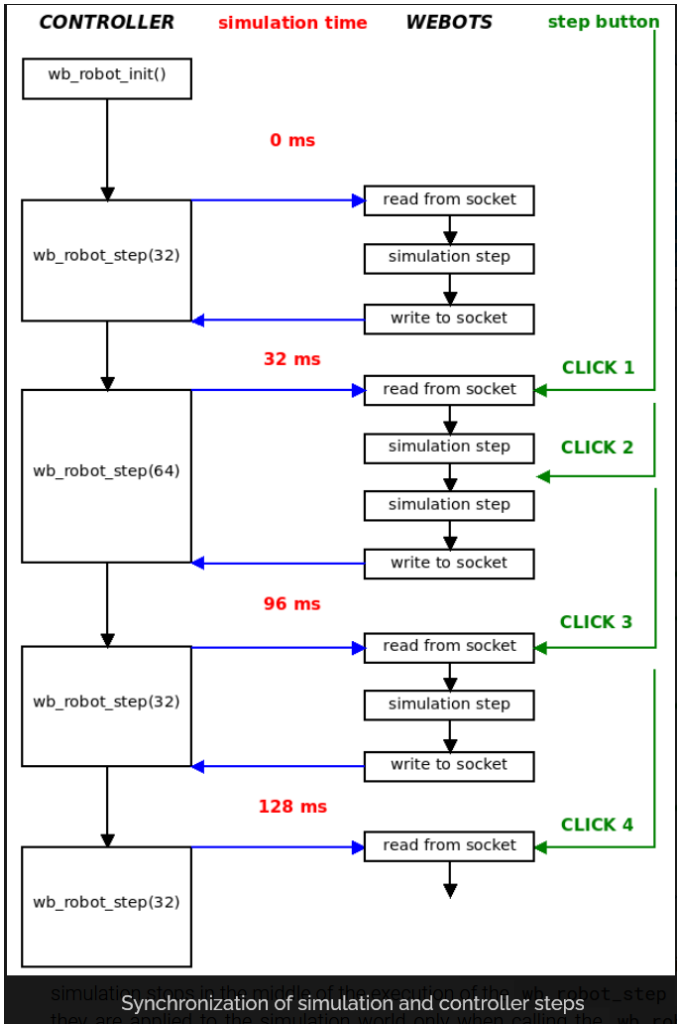


Рисунок 20. Синхронизация шагов симуляции мира и контроллера.

Таблица с названиями компонентов робота e-puck, приведенная на рисунке 21, также доступна по (ссылке).

Device	Name
Motors	'left wheel motor' and 'right wheel motor'
Position sensors	'left wheel sensor' and 'right wheel sensor'
Proximity sensors	'ps0' to 'ps7'
Light sensors	'ls0' to 'ls7'
LEDs	'led0' to 'led7' (e-puck ring), 'led8' (body) and 'led9' (front)
Camera	'camera'
Accelerometer	'accelerometer'
Gyro	'gyro'
Ground sensors (extension)	'gs0', 'gs1' and 'gs2'
Speaker	'speaker'

Рисунок 21. Таблица с названиями компонентов робота e-puck.

Очевидно, что задание движения робота посредством прецизионной установки угла поворота колеса имеет применение, но весьма ограниченное – когда необходимо точно проехать на некоторое небольшое расстояние (порядка нескольких метров). Если же задача подразумевает передвижение мобильной роботизированной системы на средние и дальние дистанции, при этом точность позиционирования может варьироваться порядка нескольких сантиметров, то целесообразней задавать движение иным путём – а именно, посредством указания скорости вращения колес и времени передвижения.

Безусловно, такая функция (задание скорости вращения колес) присутствует в Webots. Однако для того, чтобы отмерить время, нам необходимо воспользоваться таймер-службой, или в конце концов, небезызвестной задержкой (delay), реализуемой на любом реальном микроконтроллере. Таймера, как устройства в контроллере, в Webots нет (причина описана выше), однако мы можем написать функцию задержки, которая свяжет шаг симуляции и фактическое время симуляции мира. Такая функция должна принимать аргумент – время, например, в миллисекундах и обеспечивать задержку на требуемый срок.

Рассмотрим следующий код-листинг (рисунок 22), демонстрирующий задание необходимых параметров для двигателей, а также создание и проверка в функции задержки.

```python
"""Lab_work_1_controller_Petrov controller."""

# Вам может потребоваться импортировать некоторые классы модуля контроллера. Пример:
# from controller import Robot, Motor, DistanceSensor
# from math import fabs, pi - добавляет из библиотеки math функцию абсолютного значения числа и число Пи
from controller import Robot
from math import fabs, pi
# Создаём объект класса Robot
robot = Robot()

MAX_SPEED = 6.28 # максимальная скорость вращения колеса (рад/с)
WHEEL_RAD = 0.0205 # радиус колеса (м)
AXLE_LENGTH = 0.052 # длина оси конструкции робота (м)

# Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ
TIME_STEP = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит тип данных в целочисленный

def delay(delay_ms): #обьявляем функцию, аргумент - переменная delay_ms - в мс.
    control_time = robot.getTime()
    while robot.step(TIME_STEP) != -1:
        if robot.getTime() - control_time > delay_ms / 904:  # если разница текущего и замеренного
            break                                            # если прошло времени больше отмеренного - выходим из цикла
                                                             # 904 - константа, полученная из 1000 мс - 3 * 32 мс.
                                                             # В теле функции задержки 3 раза вызывается getTime() и step(TIME_STEP)
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

leftMotor.setPosition(float('inf')) # Разрешает поворачиваться колесу на полный оборот
rightMotor.setPosition(float('inf')) # аналогично для другого мотора
leftMotor.setVelocity(0.0)    # устанавливает скорость вращения колеса 0 (рад/с)
rightMotor.setVelocity(0.0)   # аналогично для другого мотора

print (robot.getTime()," seconds. The code has started.")
delay(840) # задержка около секунды. Так как каждый вызов getTime() или step(TIME_STEP) приостанавливает ход времени на 32 мс.
print (robot.getTime()," seconds") # Для пяти вызовов задержка составляет 160 мс. Подробнее: в тексте ЛР

# Main loop: - бесконечный цикл
# -выполняем шаги симуляции, пока Webots не остановит контроллер
while robot.step(TIME_STEP) != -1:
    pass #команда - заглушка, ничего не делает
```

Рисунок 22. Код-листинг инициализации, конфигурации моторов и задержки.

Терминал-консоль находится в нижней части экрана (показано на рисунке 23).

```
Console - All
INFO: Lab_work_1_controller_Petrov: Starting controller: python.exe -u Lab_work_1_controller_Petrov.py
0.0  seconds. The code has started.
0.96  seconds passed
```

Рисунок 23. Вывод замеренного времени задержки в консоль.

Теперь мы обладаем инструментарием для решения задачи движения по траектории.

Для начала попробуем заставить e-puck проехать некоторое расстояние по прямой. Для этого нам необходимо написать функцию, входными параметрами которого будут: расстояние (в метрах) и скорость движения (или, что будет удобнее, процент от максимальной скорости передвижения, так как максимальная скорость вращения колес – константное значение, которое можно найти в документации к роботу).

Ниже (на рисунке 24) приведены габаритные и справочные характеристики робота e-puck. Нас интересуют прежде всего: радиус колеса (Wheel radius) и максимальная скорость

E-puck Model

Characteristics	Values
Diameter	71 mm
Height	50 mm
Wheel radius	20.5 mm
Axle Length	52 mm
Weight	0.16 kg
Max. forward/backward speed	0.25 m/s
Max. rotation speed	6.28 rad/s

Рисунок 24. Габаритные и справочные характеристики робота e-puck

Входные параметры функции движения по прямой: расстояние (в метрах) и скорость движения (в процентах).

Если рассматривается прямолинейное равномерное движение, то расстояние (s, м) связано со скоростью (v, м/с) и временем передвижения (t, с) как:

$$s = vt \quad (1)$$

Для того, чтобы найти линейную скорость (м/с) передвижения робота, необходимо перемножить угловую скорость (рад/с) на радиус колеса R (м) - константа:

$$v = \omega R \quad (2)$$

Выразим угловую скорость ω (рад/с) через процент P (%) и максимальную угловую скорость вращенияω_{max} (рад/с) - константа:

$$v = P \cdot \omega_{max} \cdot R \quad (3)$$

Основной задачей функции будет являться подсчет времени (t, с), в течение которого необходимо проехать с постоянной заданной скоростью некоторое заданное расстояние:

$$t = \frac{s}{P \cdot 0.01 \cdot \omega_{max} \cdot R} \quad (4)$$

Теперь представим эти вычисления в виде функции движения вперед, предварительно указав необходимые константы (R, ω_{max}) из справочных данных (рисунок 24).

Для удобства создадим функцию остановки робота stop(). Напишем разрабатываемую функцию по формулам 1 – 4. И вызовем ее один раз, для проезда робота на расстояние 0.3 метра со скоростью 50% от максимальной.

Получится следующий код-листинг (рисунок 25).

```
# from math import fabs, pi - добавляет из библиотеки math функцию абсолютного значения числа и число Пи
from controller import Robot
from math import fabs, pi
# Создаём объект класса Robot
robot = Robot()

MAX_SPEED = 6.28 # максимальная скорость вращения колеса (рад/с)
WHEEL_RAD = 0.0205 # радиус колеса (м)
AXLE_LENGTH = 0.052 # длина оси конструкции робота (м)

# Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ
TIME_STEP = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит тип данных в целочисленный

def delay(delay_ms): #объявляем функцию, аргумент - переменная delay_ms - в мс.
    control_time = robot.getTime()
    while robot.step(TIME_STEP) != -1:          # пока программа не остановлена... Функция step() не может вернуть -1.
          if robot.getTime() - control_time > delay_ms / 904:  # если разница текущего и замеренного
                break                                 # если прошло времени больше отмеренного - выходим из цикла
                                                      # 904 - константа, полученная из 1000 мс - 3 * 32 мс.
                                                      # В теле функции задержки 3 раза вызывается getTime() и step(TIME_STEP)
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

leftMotor.setPosition(float('inf')) # Разрешает поворачиваться колесу на полный оборот
rightMotor.setPosition(float('inf')) # аналогично для другого мотора
leftMotor.setVelocity(0.0)    # устанавливает скорость вращения колеса 0 (рад/с)
rightMotor.setVelocity(0.0)   # аналогично для другого мотора

print (robot.getTime()," seconds. The code has started.")
delay(860) # задержка около секунды. Так как каждый вызов getTime() или step(TIME_STEP) приостанавливает ход времени на 32 мс.
print (robot.getTime()," seconds passed") # Для пяти вызовов задержка составляет 160 мс. Подробнее: в тексте ЛР

def stop():                   # создадим для удобства функцию остановки робота
     leftMotor.setVelocity(0)
     rightMotor.setVelocity(0)

def move_straight(s, coef):
    t = abs( s / ( coef * 0.01 * MAX_SPEED * WHEEL_RAD ) ) # расчет времени движения в секундах
    leftMotor.setVelocity(coef * 0.01 * MAX_SPEED)        # если коэф. < 0, то
    rightMotor.setVelocity(coef * 0.01 * MAX_SPEED)       # движение назад,иначе - вперед
    delay(t * 1000)  # расчитанное время - в секундах, а функция delay принимает миллисекунды
    stop() # вызов функции остановки

move_straight(0.2, 50) # Движение вперед на 0.3 м со скоростью 50 % от максимальной.
delay(1000)            # Задержка на одну секунду
move_straight(0.2, -50) # Движение назад на 0.3 м со скоростью 50 % от максимальной.

# Main loop: - бесконечный цикл
# -выполняет шаги симуляции, пока Webots не остановит контроллер
while robot.step(TIME_STEP) != -1:
    pass #команда - заглушка, ничего не делает
```

Рисунок 25. Код-листинг описание и применение функции движения по прямой.

Убедимся, что все сделано верно (рисунок 26). Для этого запустим режим симуляции и после остановки робота, передвинем его на точку старта (центр

арены), при этом появляется надпись “0.3 m”, на какое расстояние был перемещен робот.

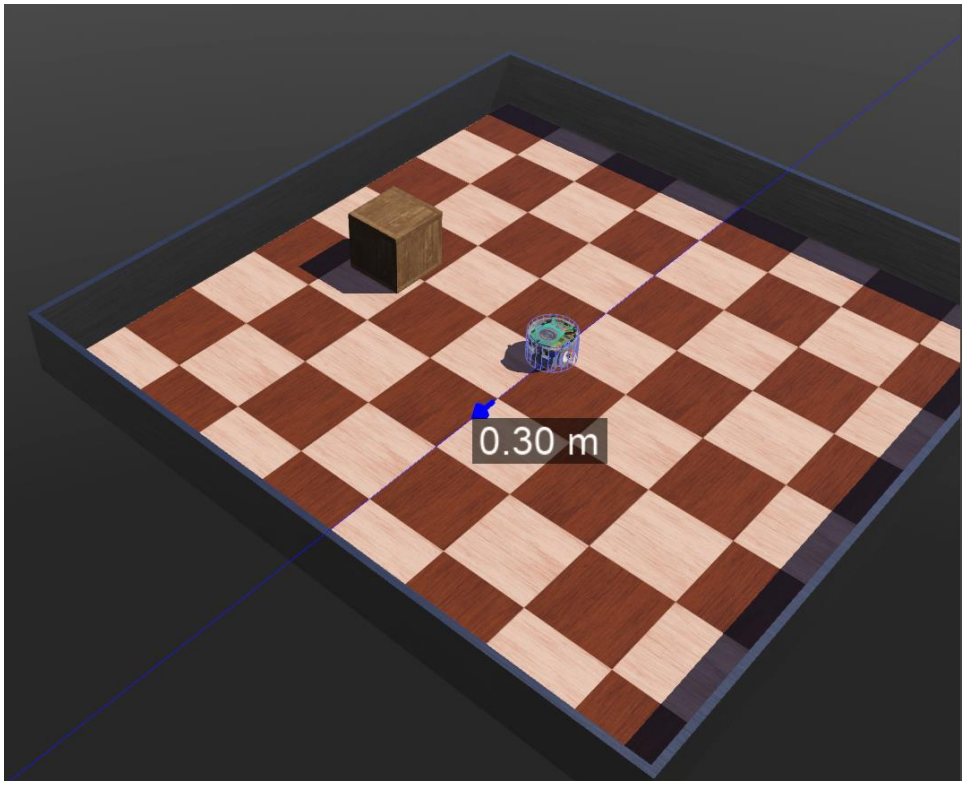


Рисунок 26. Проверка правильности исполнения кода функции движения вперед.

Попробуйте поменять входные данные функции и убедиться, что код работает верно.

Рассмотрим теперь другую задачу - осуществление поворота на месте.

Конструкция робота e-puck геометрически позволяет совершить такой маневр, так как на оси, к которой прикреплены колеса, лежит геометрический центр корпуса робота. Такой поворот на месте можно сравнить с поворотом гусеничного танка: одна гусеница крутится вперед, другая – назад.

Входными данными будут: угол (□, градусы), на который осуществляется поворот, и скорость поворота (P, % скорости от максимальной).

Выходными данными будет являться время (t, с) поворота, которое необходимо рассчитать.

Нарисуем схематично робота e-puck в виде сверху (рисунок 27). Отметим угол (α, градусы), опирающийся на дугу (выделена красным цветом). Длина дуги

(l, м) – путь, проходящий колесом, при повороте на месте выражается через угол α так:

$$l = 2\pi r \cdot \frac{\alpha}{360} \quad (5)$$

Где r – радиус, проведенный от центра робота до точки касания колесом поверхности земли. Этот радиус будем именовать эффективным радиусом, который является половиной оси, к которой прикреплены колеса робота. Длину оси робота (Axle Length) можно найти в таблице справочных данных, приведенных на рисунке 24, равный 0,052 м. Обозначим эту длину как A. Также для решения.

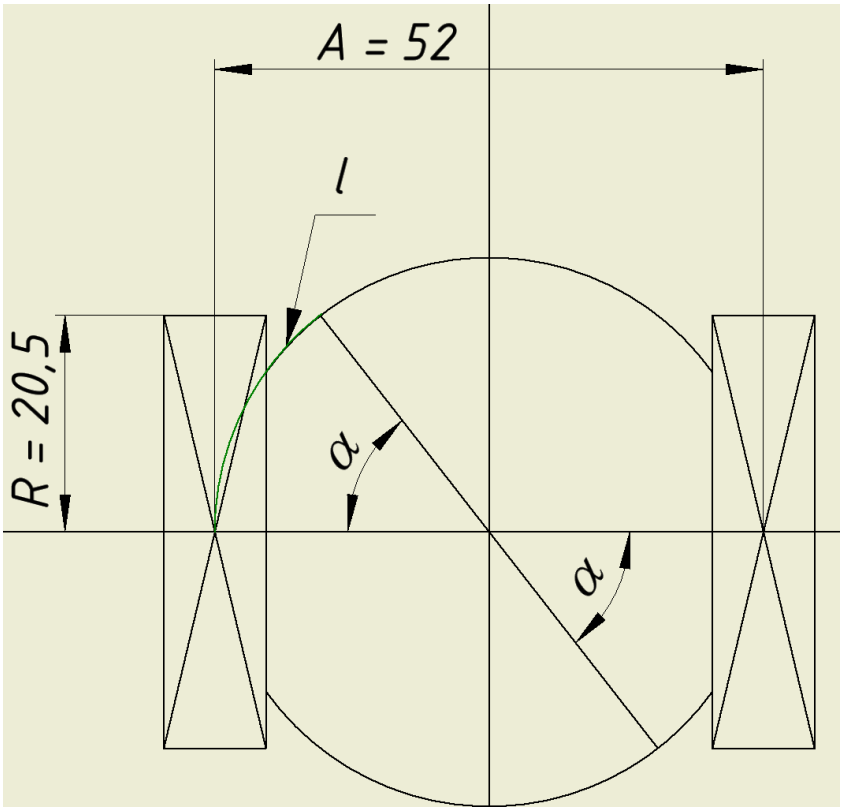


Рисунок 27. Схематичное изображение робота e-puck. Вид сверху.

Тогда в соответствии с формулой (5) и замечаниями выше, получим:

$$l = \pi \cdot A \cdot \frac{\alpha}{360} \quad (6)$$

Выразим время, необходимое для поворота на угол a (градусы) со скоростью P (% скорости от максимальной):

$$t = \frac{\pi \cdot A \cdot}{360 \cdot P \cdot 0.01 \cdot \omega_{max} \cdot R} \quad (7)$$

Теперь представим эти вычисления в виде функции поворота на угол, предварительно указав необходимые константы (R, ω_{max}, Al) из справочных данных (рисунок 24).

Напишем функцию по формулам 5-7. И вызовем ее один раз, для поворота робота на угол 90 градусов со скоростью 30% от максимальной.

Уточним, что поворот по часовой стрелке объявляется положительным углом, а против часовой стрелки – отрицательным. Учтем этот факт при создании функции.

Заметим, что в вычислениях используется число Пи, а также будет необходима функция взятия абсолютной величины числа (abs). Эти оба объекта входят в библиотеку math, из которой можем подключить необходимые модули в начале программы.

Получится следующий код-листинг:

```
1.  # from math import fabs, pi - добавляет из библиотеки math функцию абсолютного значения числа и число Пи
2.  from controller import Robot
3.  from math import fabs, pi
4.  # Создаём объект класса Robot
5.  robot = Robot()
6.
7.  MAX_SPEED = 6.28 # максимальная скорость вращения колеса (рад/с)
8.  WHEEL_RAD = 0.0205 # радиус колеса (м)
9.  AXLE_LENGTH = 0.052 # длина оси конструкции робота (м)
10.
11. # Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ
12. TIME_STEP = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит тип данных в целочисленный
13.
14. def delay(delay_ms): #объявляем функцию, аргумент - переменная delay_ms - в мс.
15.     control_time = robot.getTime()
16.     while robot.step(TIME_STEP) != -1:          # пока программа не остановлена... Функция step() не может вернуть -1.
17.         if robot.getTime() - control_time > delay_ms / 904:  # если разница текущего и замеренного
18.             break                       # если прошло времени больше отмеренного - выходим из цикла
19.                                 # 904 - константа, полученная из 1000 мс - 3 * 32 мс.
20.                                 # В теле функции задержки 3 раза вызывается getTime() и step(TIME_STEP)
```

```
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

leftMotor.setPosition(float('inf')) # Разрешает поворачиваться колесу на полный оборот
rightMotor.setPosition(float('inf')) # аналогично для другого мотора
leftMotor.setVelocity(0.0)    # устанавливает скорость вращения колеса 0 (рад/с)
rightMotor.setVelocity(0.0)   # аналогично для другого мотора

print (robot.getTime()," seconds. The code has started.")
delay(860) # задержка около секунды. Так как каждый вызов getTime() или
step(TIME_STEP) приостанавливает ход времени на 32 мс.
print (robot.getTime()," seconds passed") # Для пяти вызовов задержка составляет 160
мс. Подробнее: в тексте ЛР

def stop():                 # создадим для удобства функцию остановки робота
    leftMotor.setVelocity(0)
    rightMotor.setVelocity(0)

def move_straight(s, coef):
    t = abs( s / ( coef * 0.01 * MAX_SPEED * WHEEL_RAD ) ) # расчет времени
движения в секундах
    leftMotor.setVelocity(coef * 0.01 * MAX_SPEED)       # если коэф. < 0, то
    rightMotor.setVelocity(coef * 0.01 * MAX_SPEED)      # движение назад,иначе -
вперед
    delay(t * 1000)  # расчитанное время - в секундах, а функция delay принимает
миллисекунды
    stop() # вызов функции остановки

def turn_in_place(angle, coef = 25):    # Здесь второй аргумент является заданным по
умолчанию.
    t = abs(angle) * AXLE_LENGTH  * pi/ (3.6 * WHEEL_RAD  * coef * MAX_SPEED) #
Это означает, что поворот по умолчанию
                                                # проиходит со скоростью 25 % от
максимальной.
    velocity = coef * MAX_SPEED / 100
    if angle < 0:                   # За направление поворота отвечает угол.
        leftMotor.setVelocity(-velocity)  # Если он отрицательный, то поворот против
часовой стрелки.
        rightMotor.setVelocity(velocity)
    else:
        leftMotor.setVelocity(velocity)   # Иначе - по часовой.
        rightMotor.setVelocity(-velocity)
    delay(1000 * t)
    stop()

move_straight(0.2, 50) # Движение вперед на 0.3 м со скоростью 50 % от
максимальной.
delay(1000)           # Задержка на одну секунду
move_straight(0.2, -50) # Движение назад на 0.3 м со скоростью 50 % от
максимальной.
delay(1000)           # Задержка на одну секунду
```

```
61. turn_in_place(90, 30)  # Поворот по часовой стрелке на 90 градусов, со скоростью
    30 % от максимальной
62. delay(1000)           # Задержка на одну секунду
63. turn_in_place(-90, 30)  # Поворот против часовой стрелки на 90 градусов, со
    скоростью 30 % от максимальной
64. # Main loop: - бесконечный цикл
65. # -выполняет шаги симуляции, пока Webots не остановит контроллер
66. while robot.step(TIME_STEP) != -1:
67.     pass #команда - заглушка, ничего не делает
```

Убедимся, что все сделано верно (рисунок 28). Для этого запустим режим симуляции и после остановки робота, повернем его на изначальный угол (0 радиан), при этом появляется надпись “6/12π rad” – это и есть угол, на который был повернут робот.

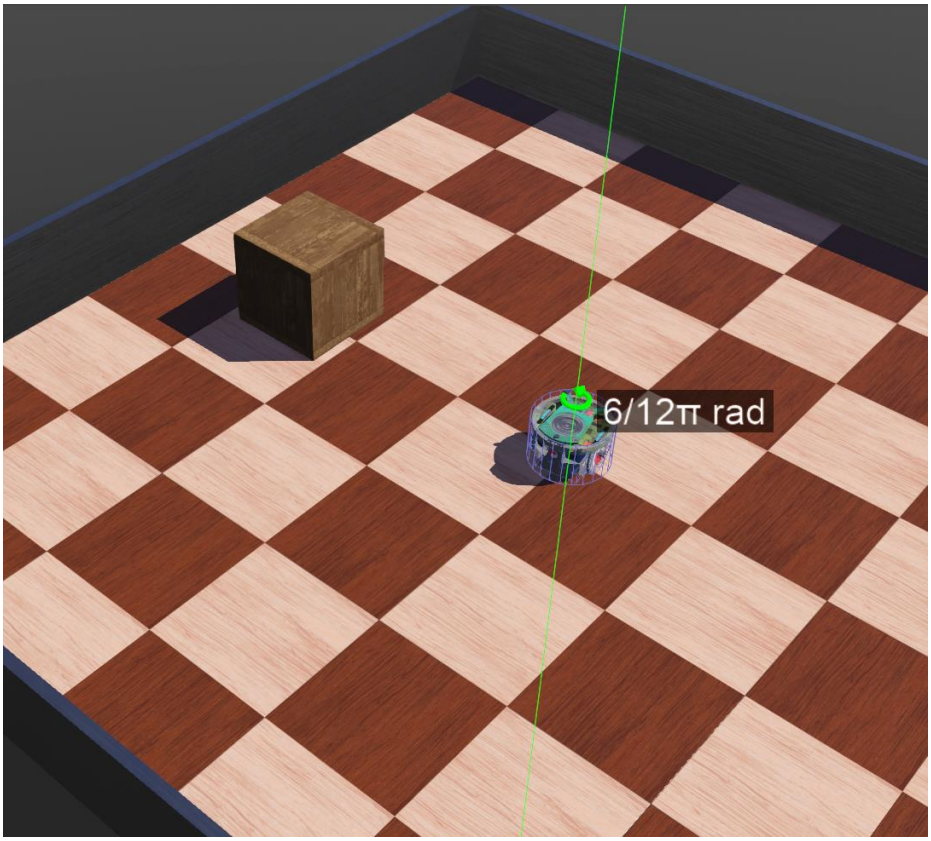


Рисунок 28. Проверка правильности исполнения кода функции поворота на месте.

Теперь усложним задачу: необходимо, чтобы робот проехал замкнутый контур по траектории «квадрат», со стороной 0,3 метра.

Алгоритм программы будет таков, как на рисунке 29:

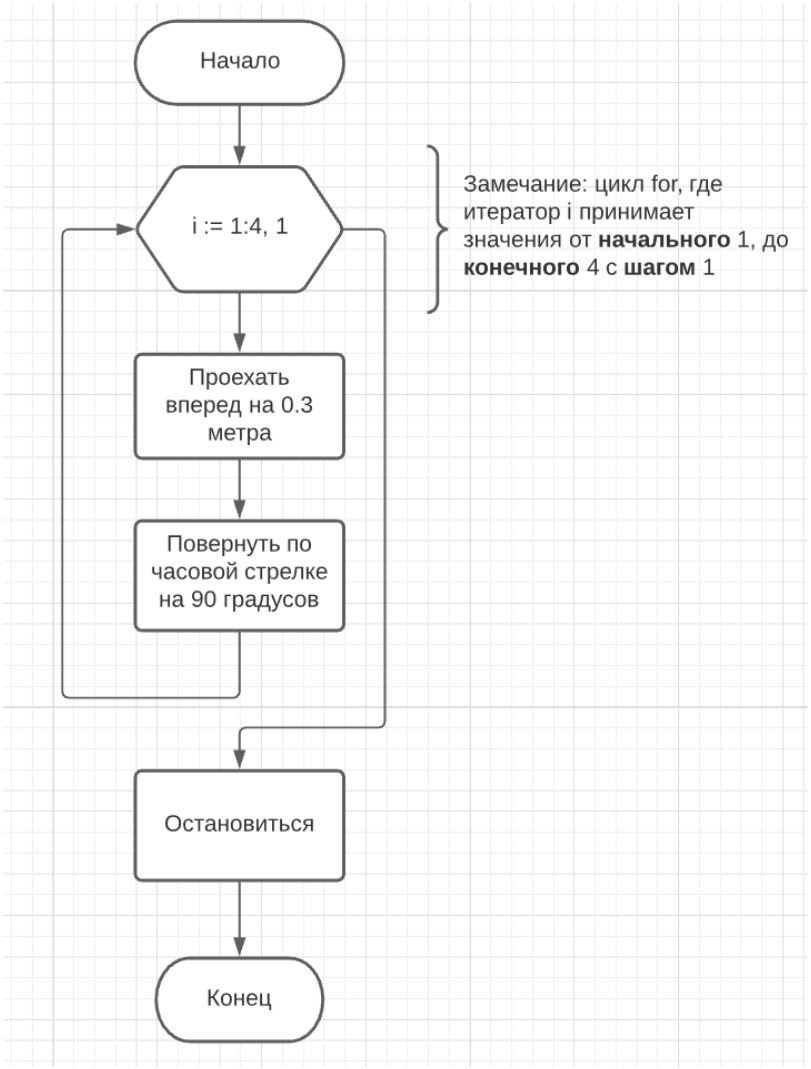


Рисунок 29. Алгоритм работы программы «движение по квадрату».

Внимание! Возможна погрешность поворота угла, связанная с вычислением времени. Ошибку можно нивелировать, введя коэффициент, используемый в тексте код-листинга выше.

Задание

Написать программу для передвижения робота e-puck по траектории, позволяющей объехать установленный в центре поля деревянный ящик (WoodenBox) с размерами 0.1x0.1x0.1 м., не касаясь объекта-препятствия, используя готовые функции передвижения.

Порядок выполнения работы

1. Запустить Webots.
2. Ознакомиться с интерфейсом 3D-симулятора разработки роботизированных средств Webots, выполнив все задания в разделе «Краткие теоретические сведения».

3. Реализовать алгоритм движения робота e-Puck по заданной траектории на языке Python в соответствии с заданием.

Содержание отчета по работе

Отчет должен содержать выполненные следующие пункты:

1. Составленная блок-схема программы.
2. Настроенный мир симуляции для выполнения задания.
3. Код-листинг программы по заданию.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Webots.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора.
3. Изложить материал раздела «Краткие теоретические сведения» и проделать практические предписания.
4. Ответить на вопросы.
5. Озвучить задание лабораторной работы.
6. После завершения работы проверить успешность выполнения учащимися самостоятельного задания.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учащимся.
2. Проверить работу моделирования в симуляторе – в случае ошибок, исправить программу вместе с учащимся.

Лабораторная работа №2 «Реализация алгоритма объезда препятствий»

Цель работы

1. Настроить мир и контроллер для лабораторной работы;
2. ознакомиться с реализацией алгоритма объезда препятствий с помощью датчиков приближения;
3. реализовать алгоритм объезда препятствий для робота E-Puck на языке Python.

Краткие теоретические сведения

Симуляция работы ИК датчиков приближения в Webots

Робот e-puck обладает инфракрасными датчиками приближения, датчиками света и светодиодами для индикации работоспособности датчиков. Их расположение и ориентация показаны на рисунке 1. Нумерация датчиков организована по часовой стрелке.

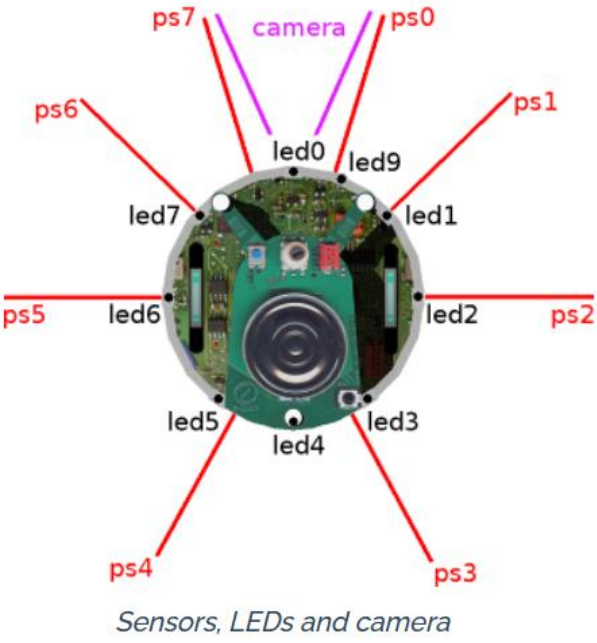


Рисунок 1. Расположение датчиков и сенсоров на роботе e-puck.

Device	x (m)	y (m)	z (m)	Orientation (rad)
ps0	0.010	0.033	-0.030	1.27
ps1	0.025	0.033	-0.022	0.77
ps2	0.031	0.033	0.00	0.00
ps3	0.015	0.033	0.030	5.21
ps4	-0.015	0.033	0.030	4.21
ps5	-0.031	0.033	0.00	3.14159
ps6	-0.025	0.033	-0.022	2.37
ps7	-0.010	0.033	-0.030	1.87
camera	0.000	0.028	-0.030	4.71239

Рисунок 2. Таблица расположения датчиков.

В последнем столбце приведено угловое расположение датчиков приближения в радианах относительно оси ps5-ps2 на рисунке 1. Обратите внимание, что датчики приближения и датчики света расположены на тех же местах, что и на настоящем роботе, и поэтому симуляция их работы максимально приближена. Отклики датчика приближения моделируются в соответствии со справочной диаграммой на рисунке 3.

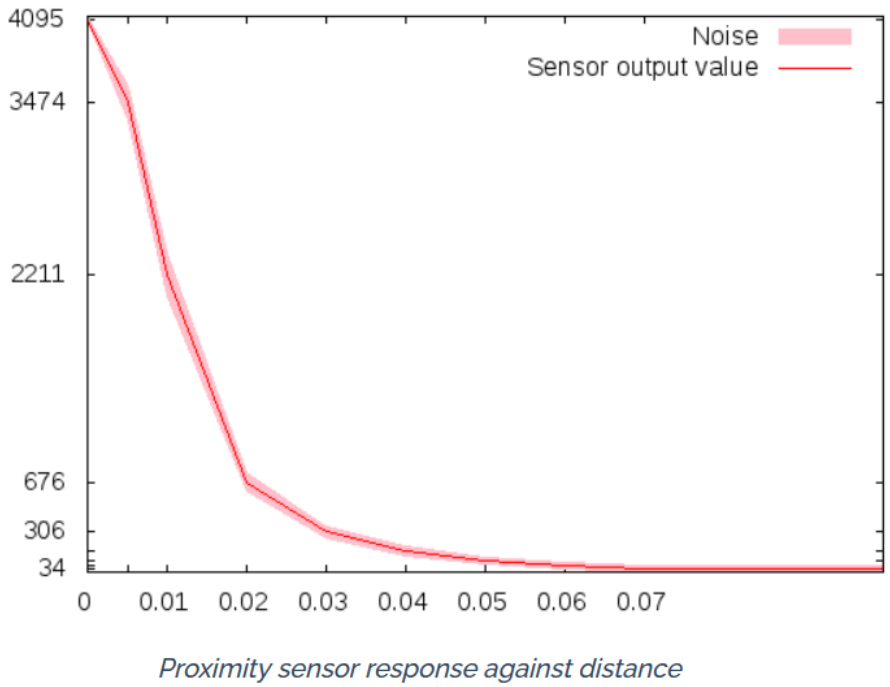


Рисунок 3. Диаграмма модели работы ИК-датчиков приближения.

Эта диаграмма является результатом калибровочных измерений, выполненных на реальном роботе.

Напоминаем, что весьма полезно более подробное ознакомление с функциональными возможностями робота e-puck по ссылке.

Создание программы-контроллера на языке Python

Для того, чтобы обеспечить организацию работы в рамках курса и более эффективно продолжать изучать систему Webots, нам понадобится вести учет разных файлов для лабораторных работ. Для этого необходимо создавать новые контроллеры и миры симуляции, при этом к функционалу, реализованному в уже изученных работах, придется возвращаться. Процесс разработки программного обеспечения, как правило, цикличен, и практически всегда требует отладки и доработки. В связи с этим, предлагаем создать репозиторий GitHub, основанный на системе контроля версий, или иным образом обеспечить хранение программ-контроллеров и миров симуляции.

Для выполнения данной работы нам потребуется уже созданный из лабораторной работы 1 мир симуляции (обычно последний мир симуляции открывается автоматически при запуске программы Webots). Однако нам необходимо его немного настроить.

Создадим новый проект также, как это было в лабораторной работе 1. При этом назовите проект «Lab_work_2_*Ваша фамилия латиницей*», а название мира будет «Lab_2_world.wbt». Затем, просто скопируйте всю папку «Lab_work_1_*Ваша фамилия латиницей*» из директории лабораторной работы 1 в папку с лабораторной работой 2, заменив все файлы. Затем, перезапустите симуляцию, нажав сочетание клавиш Ctrl-Shft_R, или нажав на иконку в верхней панели программы «замкнутая стрелка»  .

В этой работе нам необходимо создать новый контроллер. Чтобы добавить контроллер, необходимо выбрать в ленте вкладку «Wizards», и нажать на вкладку в выпадающем меню «New Robot Controller…», как на рисунке 4.

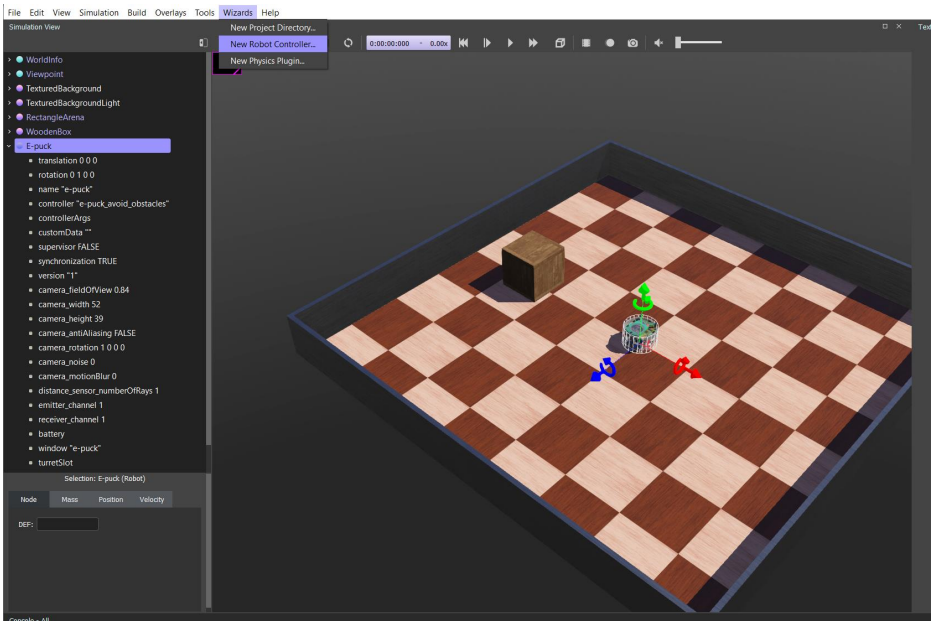


Рисунок 4. Создание нового контроллера.

Откроется диалоговое окно, нажмите «Next», чтобы продолжить. Следующая страница (рисунок 5) требует выбора языка программирования – выбираем Python и нажимаем далее (Next). В следующем окне – называем контроллер так: «Lab_work_2_controller_*Ваша фамилия латиницей*.

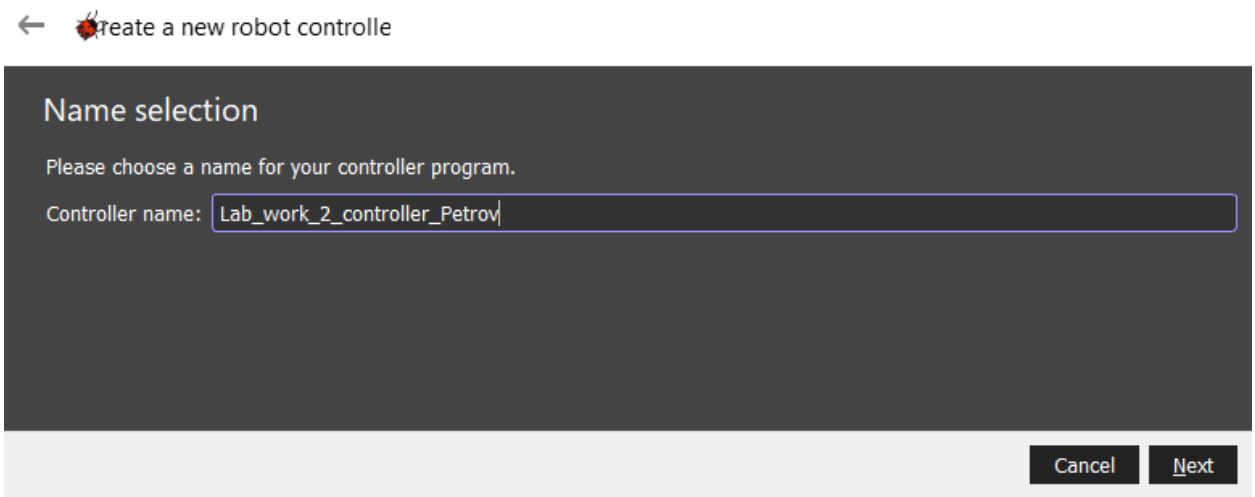


Рисунок 5. Создание нового контроллера.

Затем обязательно ставим галочку около «Open ‘Lab_work_2_controller_*Ваша фамилия латиницей*.py’ in Text Editor», чтобы открыть текст контроллера в редакторе, и нажимаем «Finish».

Теперь необходимо подключить контроллер к роботу e-puck, чтобы мы смогли сразу тестировать написанные программы.

Для этого необходимо выбрать в дереве объектов робота, найти поле «Controller» и нажать кнопку «select…», как на рисунке 6.

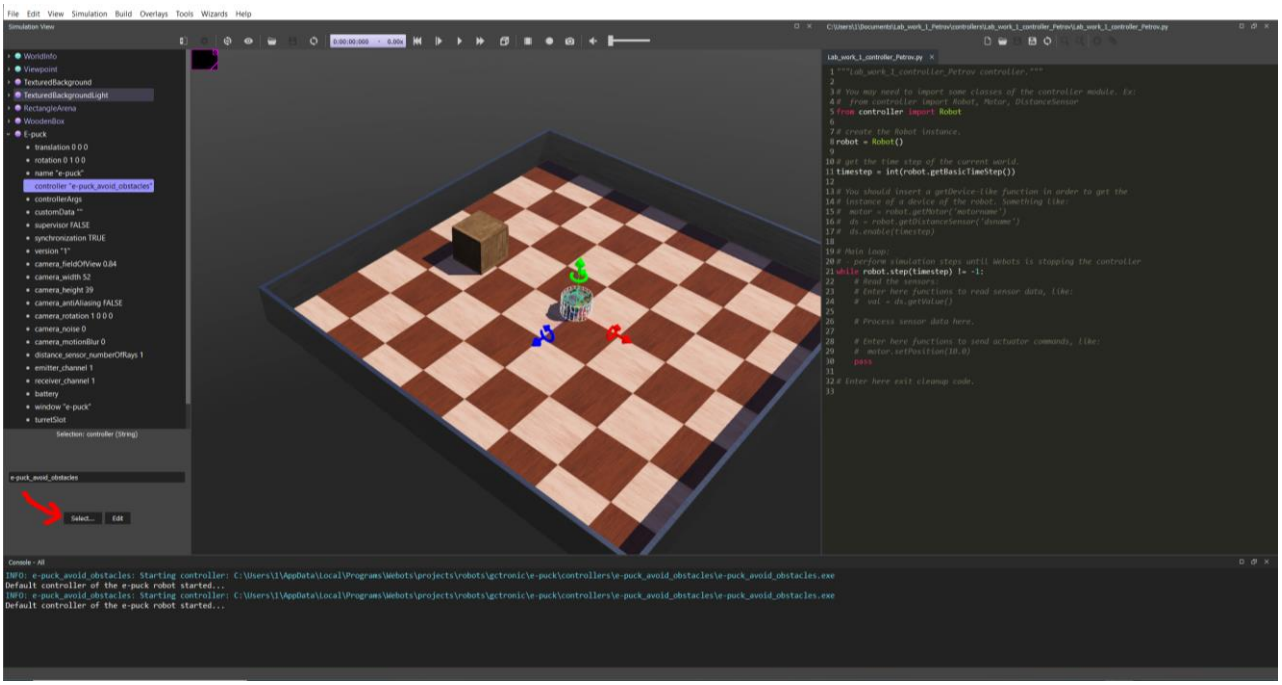


Рисунок 6. Подключение нового контроллера.

В диалоговом окне выбираем только что созданный контроллер с именем «Lab_work_2_controller_*Ваша фамилия латиницей*» (рисунок 7) и нажимаем «ОК».

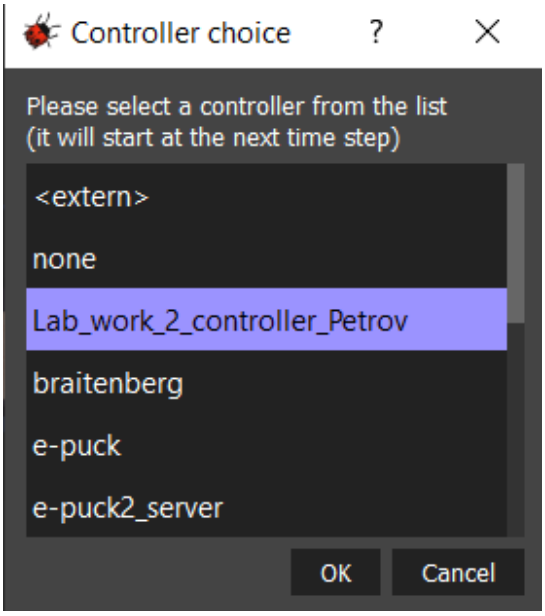


Рисунок 7. Подключение нового контроллера. Диалоговое окно.

Конфигурация мира симуляции в Webots

Для решения задачи нам будет необходимо размножить уже имеющийся объект «Box» (рисунок 8) так, чтобы создать некоторые препятствия на арене, не позволяющие проехать роботу от одной стенки арены до противоположной не свернув, как на рисунке 9.

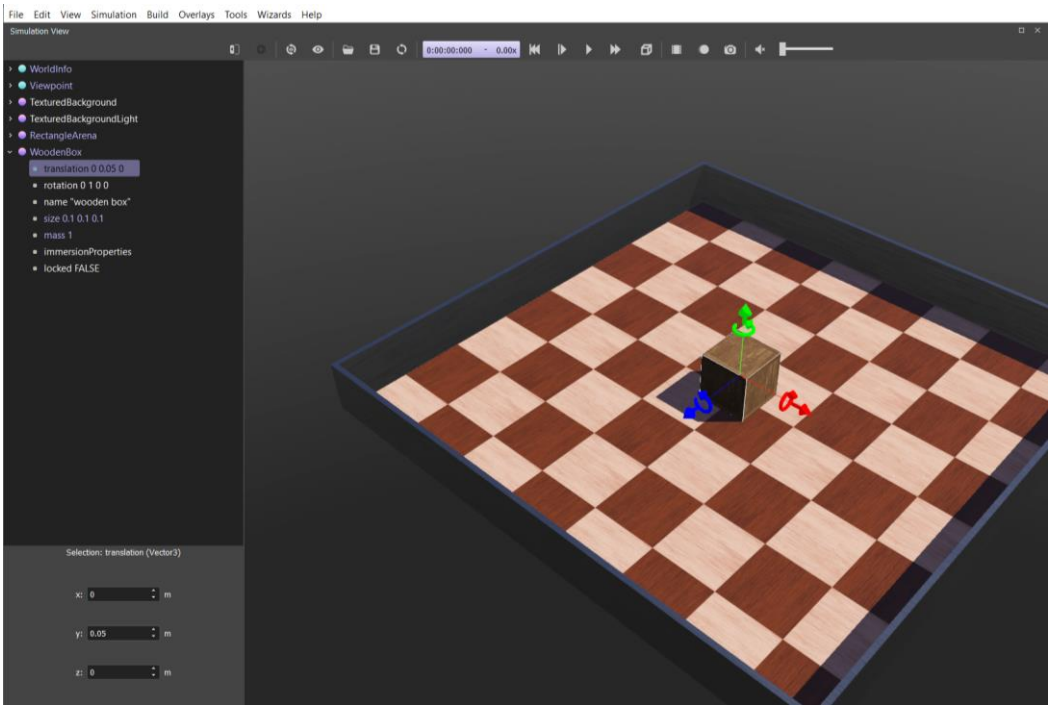


Рисунок 8. Настройка объекта «WoodenBox».

Перемещать объекты можно с помощью мыши, предварительно зажав клавишу Shift, а также, с помощью стрелок по осям, исходящим из выбранного объекта. Копировать и вставлять объекты – с помощью сочетания клавиш (Ctrl+C, Ctrl+V).

Нажмите сочетание клавиш (Ctrl+Shift+S), или нажмите на иконку, графически схожую с дискетой, в левой верхней части экрана для сохранения созданного мира.

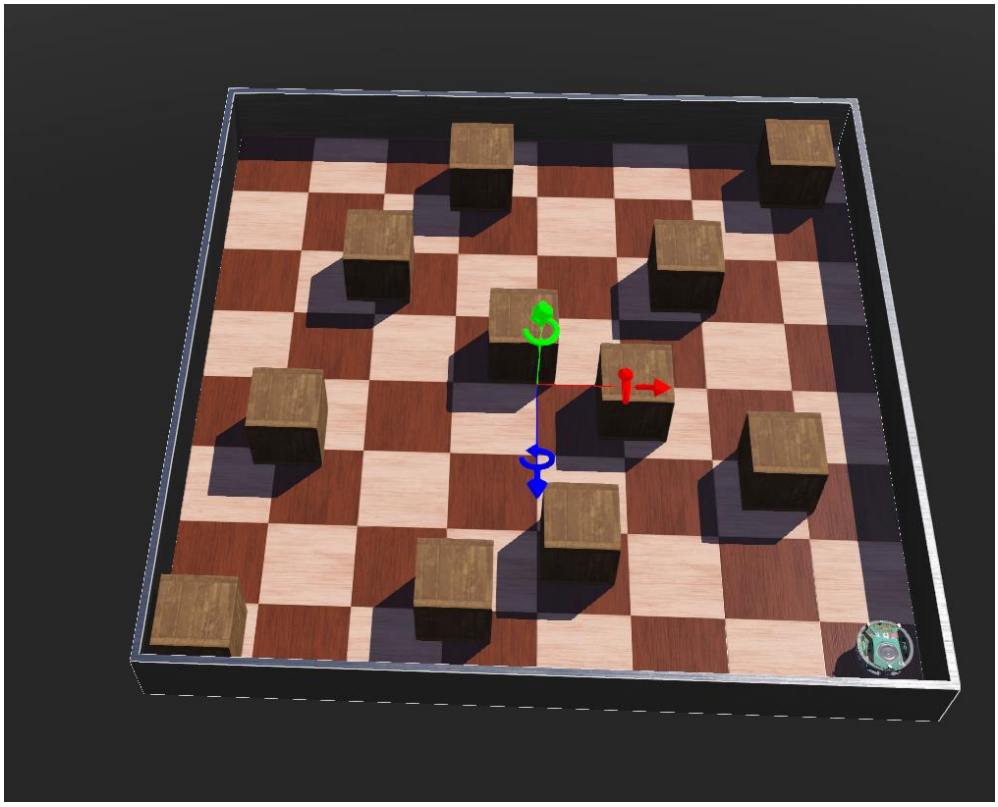


Рисунок 9. Пример созданной арены.

Обязательно сохраним файлы, нажав сочетание (Ctrl+Shift+S), чтобы не потерять настройки и добавления.

На примере текущей практической работы напишем программу, реализующую алгоритм объезда препятствий по принципу определения свободного пути следования.

Разработка алгоритма объезда препятствий роботом e-puck на языке Python

Алгоритм будет заключаться в том, чтобы считывать данные с датчиков, закрепленных по периметру робота постоянно для определения стороны, с которой было обнаружено препятствие и дальнейший разворот робота в противоположную сторону от него во избежание столкновения с препятствием.

Для того, чтобы инициализировать датчики необходимо обратиться к таблице, приведенной на рисунке 10.

Device	Name
Motors	'left wheel motor' and 'right wheel motor'
Position sensors	'left wheel sensor' and 'right wheel sensor'
Proximity sensors	'ps0' to 'ps7'
Light sensors	'ls0' to 'ls7'
LEDs	'led0' to 'led7' (e-puck ring), 'led8' (body) and 'led9' (front)
Camera	'camera'
Accelerometer	'accelerometer'
Gyro	'gyro'
Ground sensors (extension)	'gs0', 'gs1' and 'gs2'
Speaker	'speaker'

Рисунок 10. Таблица с названиями устройств на роботе.

Теперь рассмотрим алгоритм программы (рисунок 11), позволяющей объезжать препятствия по принципу определения наименьшей плотности скопления объектов.

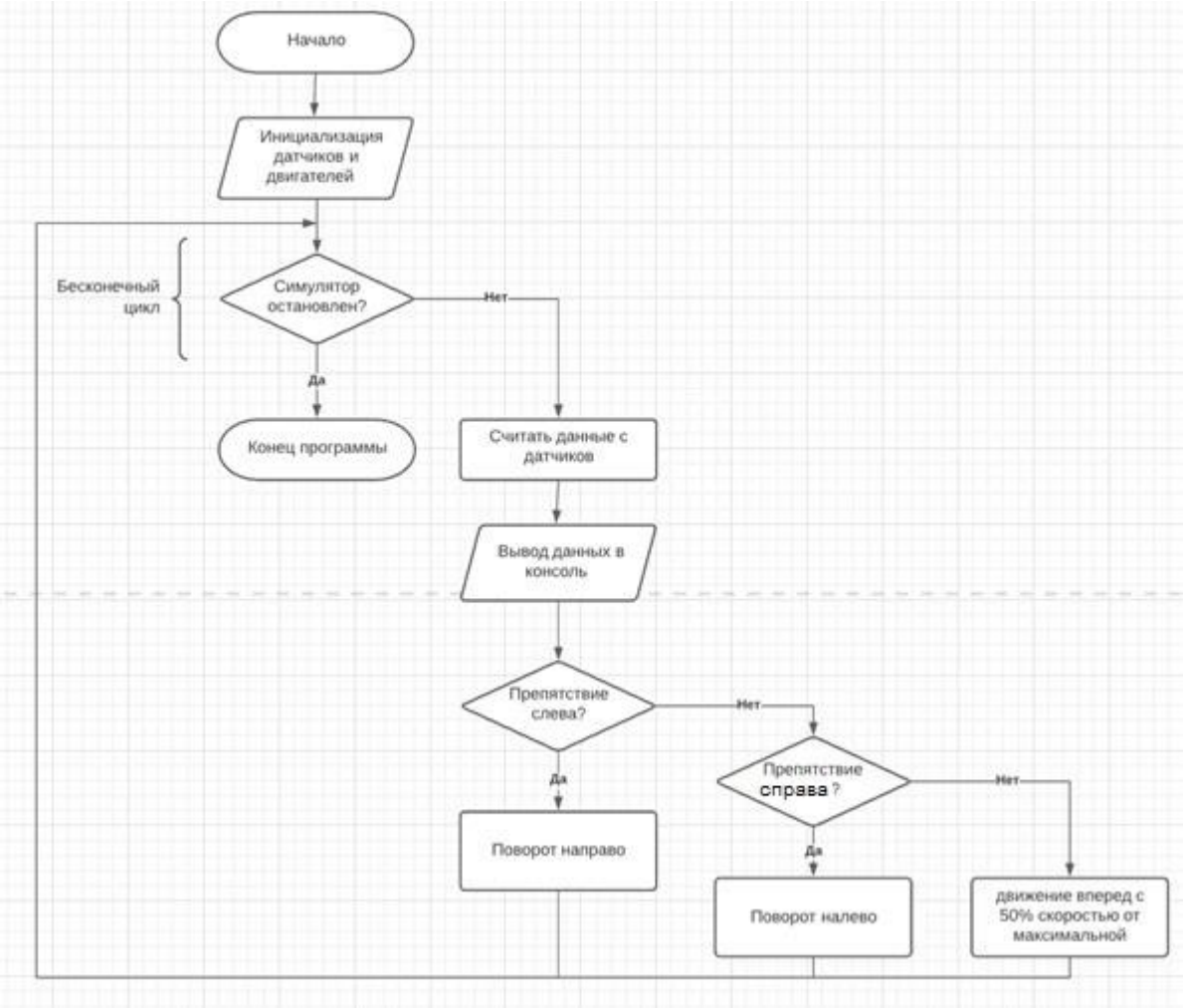


Рисунок 11. Алгоритм работы робота. Объезд препятствий.

Данный алгоритм получил такую реализацию в коде (рисунок 12).

Дадим некоторые пояснения. Для того, чтобы указать, с каким именно датчиком или устройством будет осуществляться работа, необходимо, прежде всего, импортировать библиотеку (модуль) с этим датчиком или устройством. В данном случае нас интересует – DistanceSensor. Затем, нам следует инициализировать датичик приближения, которые имеют собственные имена, узнать которые можно по таблице, изображенной на рисунке 10. Каждое устройство в Webots требует инициализации, которая занимает время, равное длительности одного TIME_STEP. Без этого датчики не будут работать.

```python
"""Lab_work_2_controller_Petrov controller."""
from controller import Robot, DistanceSensor, Motor

robot = Robot() # Создаём объект класса Robot

# Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ 1
TIME_STEP = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит тип данных в целочисленный

MAX_SPEED = 6.28 # максимальная скорость вращения колеса (рад/с)

ps = []  # создаем список для хранения имен датчиков
# обьявляем имена датчиков приближения (заданы в документации к роботу)
psNames = [
    'ps0', 'ps1', 'ps2', 'ps3',
    'ps4', 'ps5', 'ps6', 'ps7'
]

for i in range(8):
    ps.append(robot.getDevice(psNames[i]))  #добавляем по очереди имя датчика в список ps
    ps[i].enable(TIME_STEP) #ждем один такт времени симуляции для установки каждого датчика

leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

leftMotor.setPosition(float('inf')) # Разрешает поворачиваться колесу на полный оборот
rightMotor.setPosition(float('inf')) # аналогично для другого мотора
leftMotor.setVelocity(0.0)    # устанавливает скорость вращения колеса 0 (рад/с)
rightMotor.setVelocity(0.0)   # аналогично для другого мотора

# Main loop: - бесконечный цикл
# -выполняет шаги симуляции, пока Webots не остановит контроллер
while robot.step(TIME_STEP) != -1:
    # считываем данные с датчиков
    psValues = []
    for i in range(8):
        psValues.append(ps[i].getValue())
    print(psValues) #выводим значения с датчиков в монитор порта

    # определяем, с какой стороны препятствие
    right_obstacle = psValues[0] > 80.0 or psValues[1] > 80.0 or psValues[2] > 80.0
    left_obstacle = psValues[5] > 80.0 or psValues[6] > 80.0 or psValues[7] > 80.0

    # пусть робот едет с 50% скоростью
    leftSpeed  = 0.5 * MAX_SPEED
    rightSpeed = 0.5 * MAX_SPEED
    #реакция робота на обнаруженной препятствие по сторонам
    if left_obstacle: #если препятствие замечено слева - повернуть направо
        # поворот направо
        leftSpeed  = 0.5 * MAX_SPEED
        rightSpeed = -0.5 * MAX_SPEED
    elif right_obstacle: #если препятствие замечено справа - повернуть налево
        # поворот налево
        leftSpeed  = -0.5 * MAX_SPEED
        rightSpeed = 0.5 * MAX_SPEED
    # устанавливаем скорость
    leftMotor.setVelocity(leftSpeed)
    rightMotor.setVelocity(rightSpeed)
```

Рисунок 12. Код-листинг программы для объезда препятствий.

Порядок выполнения работы

1. Запустить Webots.
2. Ознакомиться с реализацией алгоритма объезда препятствий с помощью датчиков приближения, приведенным в разделе «Краткие теоретические сведения».
3. Подготовить мир симуляции в соответствии с заданием.

4. Реализовать алгоритм движения робота e-Puck на языке Python в соответствии с заданием.

Задание

Необходимо создать и настроить мир симуляции, в соответствии с разделом «Краткие теоретические сведения» и картой препятствий, изображенной на рисунке 13.

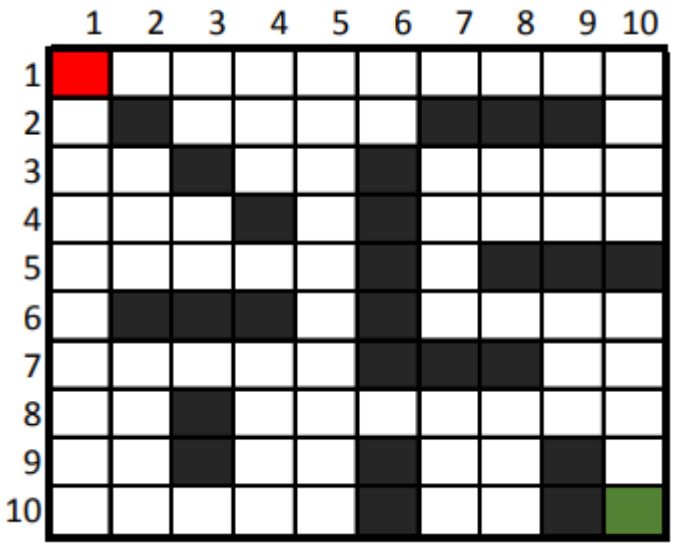


Рисунок 13. Карта препятствий.

Точка старта движения робота e-puck находится в зеленой клетке. Точка финиша – точка, куда должен следовать робот во время работы программы, находится в красной клетке. Черные клетки обозначают ящики-препятствия.

Требуется написать программу для объезда препятствий роботом e-puck на языке Python и настроить параметры движения робота, так, чтобы обеспечить его следование от точки старта до точки финиша, не касаясь объектов-препятствий. Допускается использовать функции, разработанные в лабораторной работе 1 для передвижения, при условии, что в алгоритме задействованы датчики приближения.

Содержание отчета по работе

Отчет должен содержать выполненные следующие пункты:

1. Составленная блок-схема программы.
2. Настроенный мир симуляции для выполнения задания.

3. Код-листинг программы по заданию.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Webots.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора.
3. Изложить материал раздела «Краткие теоретические сведения» и проделать практические предписания.
4. Ответить на вопросы.
5. Озвучить задание лабораторной работы.
6. После завершения работы проверить успешность выполнения учащимися самостоятельного задания.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учащимся.
2. Проверить работу моделирования в симуляторе – в случае ошибок, исправить программу вместе с учащимся.

Лабораторная работа №3 «Изучение алгоритмов навигации мобильных роботов с помощью датчиков окружения»

Цель работы

1. Настроить мир и контроллер для практической работы;
2. ознакомиться с теоретическими сведениями о работе датчиков освещенности;
3. ознакомиться с реализацией алгоритма поиска источника освещенности в ограниченном пространстве;
4. реализовать и оптимизировать алгоритм автономного поиска источника освещения в лабиринте для робота e-puck на языке Python;
5. подготовить блок-схему полученного алгоритма.

Краткие теоретические сведения

Симуляция работы датчиков освещенности в Webots

Помимо функциональности, связанной с роботами, в симуляторе Webots реализована физика объектов окружения, включая, даже, световые эффекты. Такое многообразие физических эффектов в среде позволяют проводить моделирование и симуляцию работы роботов для широкого спектра задач.

Так, например, благодаря возможности добавления точечных или направленных источников света, симулятор позволяет разрабатывать программы для роботов, обладающих светочувствительными датчиками.

Светочувствительные датчики или датчики освещенности нашли широкое применение в области управления наружным освещением, ракетостроении, разработке космических спутников, в робототехнике. Принцип их работы основан на том, что светочувствительный элемент изменяет свою проводимость в зависимости от степени освещенности. В качестве такого элемента используют:

1. Фоторезисторы (чаще и дешевле всего);
2. Фотодиоды;
3. Фототранзисторы.

Рисунок 1. Пример фотоприборов.

Все три типа светочувствительных элементов объединяет то, что их проводимость возрастает вместе с освещенностью. Простым языком, они проводят ток тогда, когда на них попадает свет. Отличием является лишь чувствительность. Сигнал с датчика освещения приходит на усилитель, который в свою очередь управляет силовым коммутационным прибором – электромагнитным реле или симистором. В дешевых малогабаритных устройствах в качестве усилителя используется 1 транзистор. А в дорогих – микросхемы.

Как уже говорилось в предыдущих работах, робот e-puck обладает инфракрасными датчиками приближения, датчиками света и светодиодами для индикации работоспособности датчиков. Их расположение и ориентация показаны на рисунке 2. Нумерация датчиков организована по часовой стрелке. Обратите внимание, что датчики препятствия и освещенности находятся на одинаковых местах.

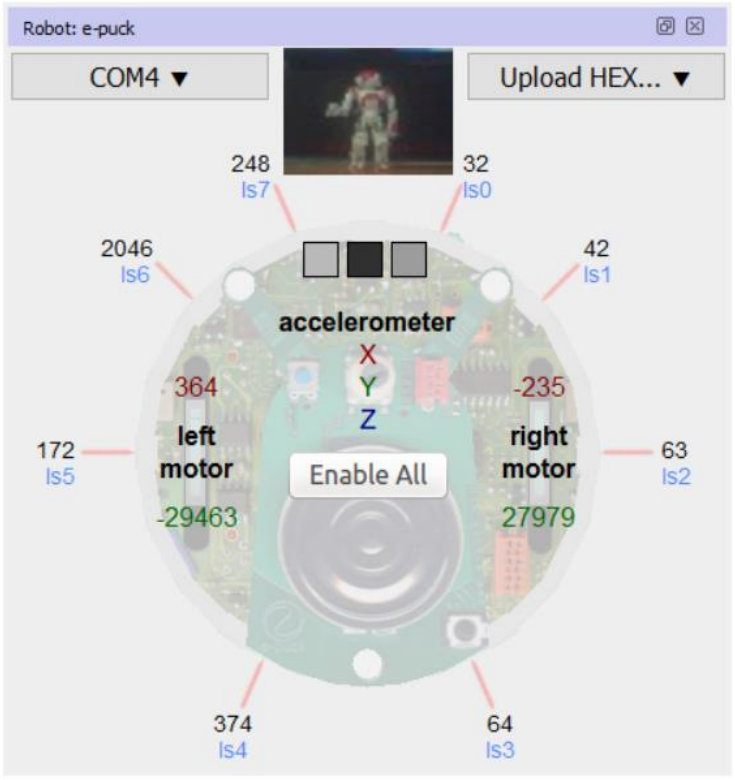


Рисунок 2. Расположение датчиков и сенсоров на роботе e-puck.

Device	x (m)	y (m)	z (m)	Orientation (rad)
ps0	0.010	0.033	-0.030	1.27
ps1	0.025	0.033	-0.022	0.77
ps2	0.031	0.033	0.00	0.00
ps3	0.015	0.033	0.030	5.21
ps4	-0.015	0.033	0.030	4.21
ps5	-0.031	0.033	0.00	3.14159
ps6	-0.025	0.033	-0.022	2.37
ps7	-0.010	0.033	-0.030	1.87
camera	0.000	0.028	-0.030	4.71239

Рисунок 3. Таблица расположения датчиков.

В последнем столбце последнего перечислены углы между отрицательной осью X и направлением устройств, при этом плоскость ZX ориентирована против часовой стрелки. Обратите внимание, что датчики приближения и датчики света расположены на тех же местах, что и на настоящем роботе, и поэтому симуляция их работы максимально приближена.

Напоминаем, что весьма полезно более подробно ознакомиться с функциональными возможностями робота e-puck по ссылке.

Создание программы-контроллера на языке Python

Создадим новый проект также, как это было в лабораторной работе 1. При этом назовите проект «Lab_work_3_*Ваша фамилия латиницей*», а название мира будет «Lab_3_world.wbt». Добавьте робота e-puck в мир симуляции.

Чтобы добавить контроллер, необходимо выбрать в ленте вкладку «Wizards», и нажать на вкладку в выпадающем меню «New Robot Controller…», как на рисунке 4.

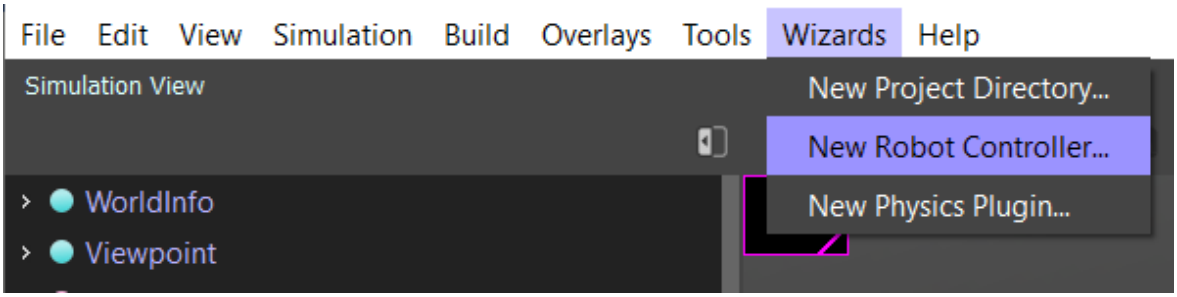


Рисунок 4. Создание нового контроллера.

Откроется диалоговое окно, нажмите «Next», чтобы продолжить. Следующая страница требует выбора языка программирования – выбираем Python и нажимаем далее (Next). В следующем окне (рисунок 5) – называем контроллер так: «Lab_3_controller_*Ваша фамилия латиницей*» и переходим далее.

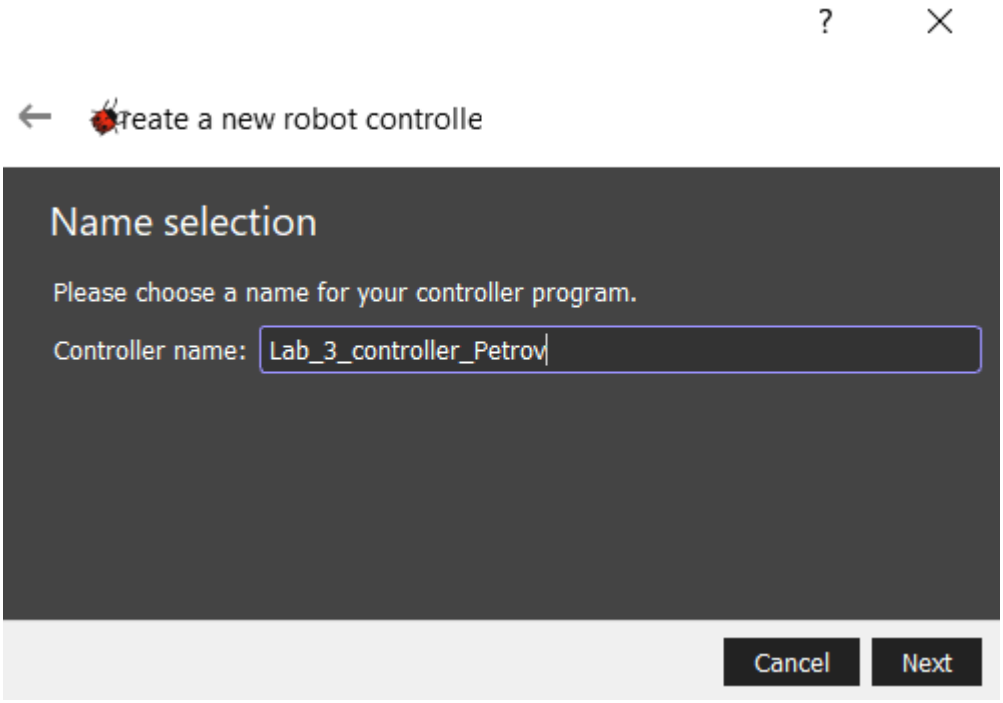


Рисунок 5. Создание нового контроллера.

Затем обязательно ставим галочку «Open ‘Lab_work_3_controller_*Ваша фамилия латиницей*.py’ in Text Editor», чтобы открыть текст контроллера в редакторе, и нажимаем «Finish».

Теперь необходимо подключить контроллер к роботу e-puck, чтобы мы смогли сразу тестировать написанные программы.

Для этого необходимо выбрать в дереве объектов робота, найти поле «Controller» и нажать кнопку «select…», как на рисунке 6.

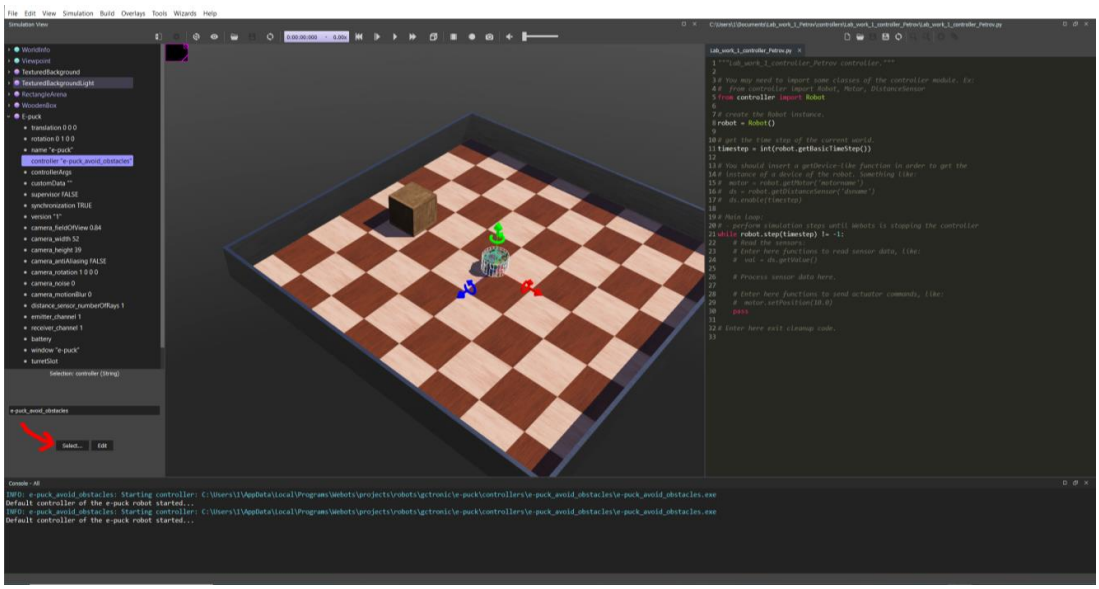


Рисунок 6. Подключение нового контроллера.

В диалоговом окне выбираем только что созданный контроллер с именем «Lab_3_controller_*Ваша фамилия латиницей*» и нажимаем «ОК».

Конфигурация мира симуляции в Webots

Для решения задачи нам будет необходимо погасить внешнее освещение, чтобы заменить его точечным. Для этого необходимо выбрать в панели объектов «TextureBackgroundLight» параметр «luminosity», отвечающий за освещенность, и выставить его значение в 0, чтобы получилось, как на рисунке 7.

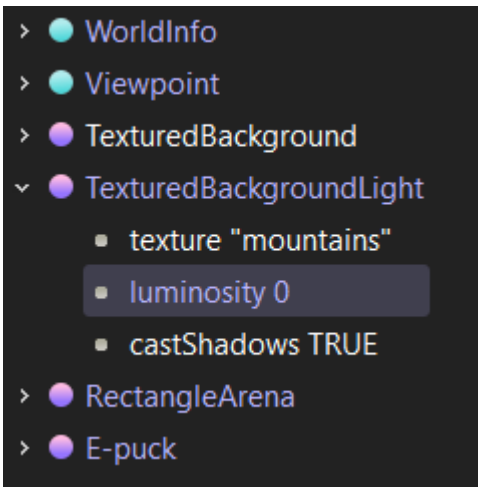


Рисунок 7. Выключаем внешний свет арены.

Далее добавим точечный источник света, который будет размещаться где-то над ареной. Для этого выберем с помощью ЛКМ любой объект в панели объектов и станет доступной команда «Add a new object or import an object» (быстрое сочетание клавиш Ctrl + Shift + A или кнопка в верхнем левом углу экрана ). Нажав эту кнопку, мы попадаем в диалоговое окно выбора нового объекта для добавления и в поиске делаем запрос «light», как на рисунке 8, и выбираем «CeilingSpotLight», далее – нажимаем кнопку «Add» чтобы добавить.

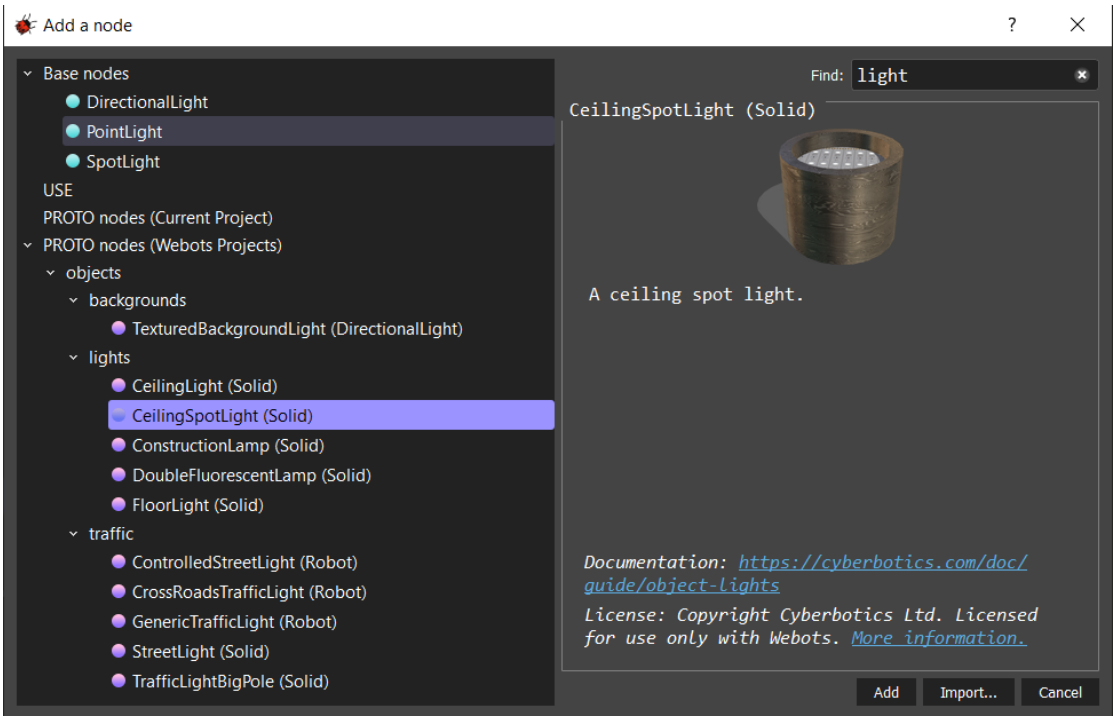


Рисунок 8. Добавление точечного освещения.

Далее произведите следующую настройку добавленного объекта, как на рисунке 9.

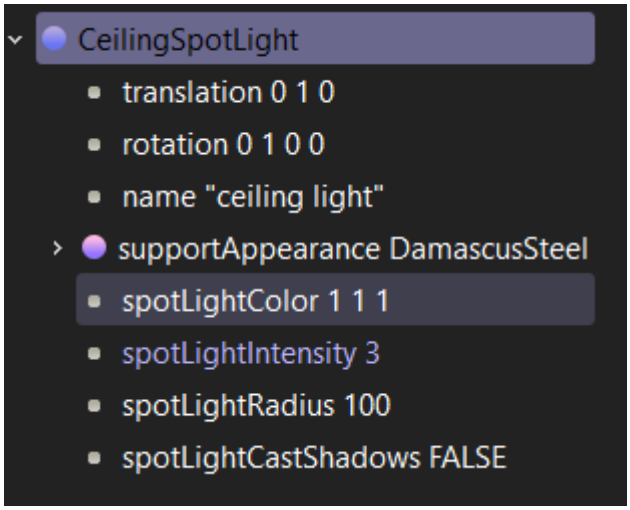


Рисунок 9. Настройка светильника.

Должно получиться примерно следующим образом (рисунок 10):

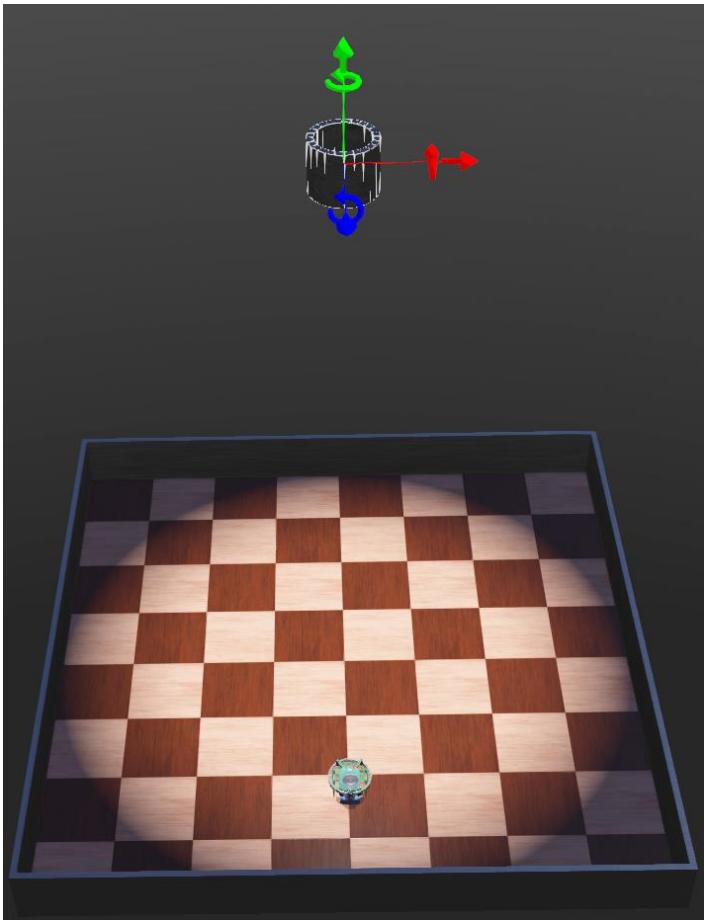


Рисунок 10. Добавленный источник света.

Перемещать объекты можно с помощью мыши, предварительно зажав клавишу Shift, а также, с помощью стрелок по осям, исходящим из выбранного

объекта. Копировать и вставлять объекты – с помощью сочетания клавиш (Ctrl+C, Ctrl+V).

Нажмите сочетание клавиш (Ctrl+Shift+S), или нажмите на иконку, графически схожую с дискетой  , в левой верхней части экрана для сохранения созданного мира.

На примере данной практической работы напишем программу, реализующую алгоритм автономного поиска источника освещения в замкнутом пространстве для робота e-puck на языке Python

Разработка алгоритма автономного поиска источника освещения в замкнутом пространстве для робота e-puck на языке Python

Алгоритм будет заключаться в том, чтобы считывать данные сразу с двух видов датчиков, закрепленных по периметру робота постоянно – датчиков препятствия и датчиков освещенности. Первые служат для определения стороны, с которой может быть обнаружено препятствие, чтобы произвести дальнейший поворот робота в противоположную сторону от препятствия во избежание столкновения с ним. Вторые – для решения основной задачи – найти центр освещенной области – проекцию лампы, освещающей арену сверху (так как именно непосредственно под центром лампы будет наблюдаться наибольшая освещенность).

Далее следует, найдя это место с некоторой погрешностью error (например, 1 %), прервать цикл и остановиться.

Поиск самого освещенного места будет происходить с помощью нахождения медианы и процентного отклонения error от нее.

Медиана (от лат. mediāna «середина») набора чисел — число, которое находится в середине этого набора, если его упорядочить по возрастанию. Таким образом, медиана — это такое число, что половина из элементов набора больше него, а другая половина меньше (строго говоря, это верно только если все элементы набора различны).

Для того, чтобы инициализировать датчики необходимо обратиться к таблице, приведенной на рисунке 11.

Device	Name
Motors	'left wheel motor' and 'right wheel motor'
Position sensors	'left wheel sensor' and 'right wheel sensor'
Proximity sensors	'ps0' to 'ps7'
Light sensors	'ls0' to 'ls7'
LEDs	'led0' to 'led7' (e-puck ring), 'led8' (body) and 'led9' (front)
Camera	'camera'
Accelerometer	'accelerometer'
Gyro	'gyro'
Ground sensors (extension)	'gs0', 'gs1' and 'gs2'
Speaker	'speaker'

Рисунок 11. Таблица с названиями устройств на роботе.

Теперь рассмотрим алгоритм программы (рисунок 12), позволяющей автономно производить поиск центра освещенности арены в замкнутом пространстве для робота e-puck на языке Python.

При изменении положения источника освещенности робот должен сам двигаться в сторону конечного положения центра источника света.

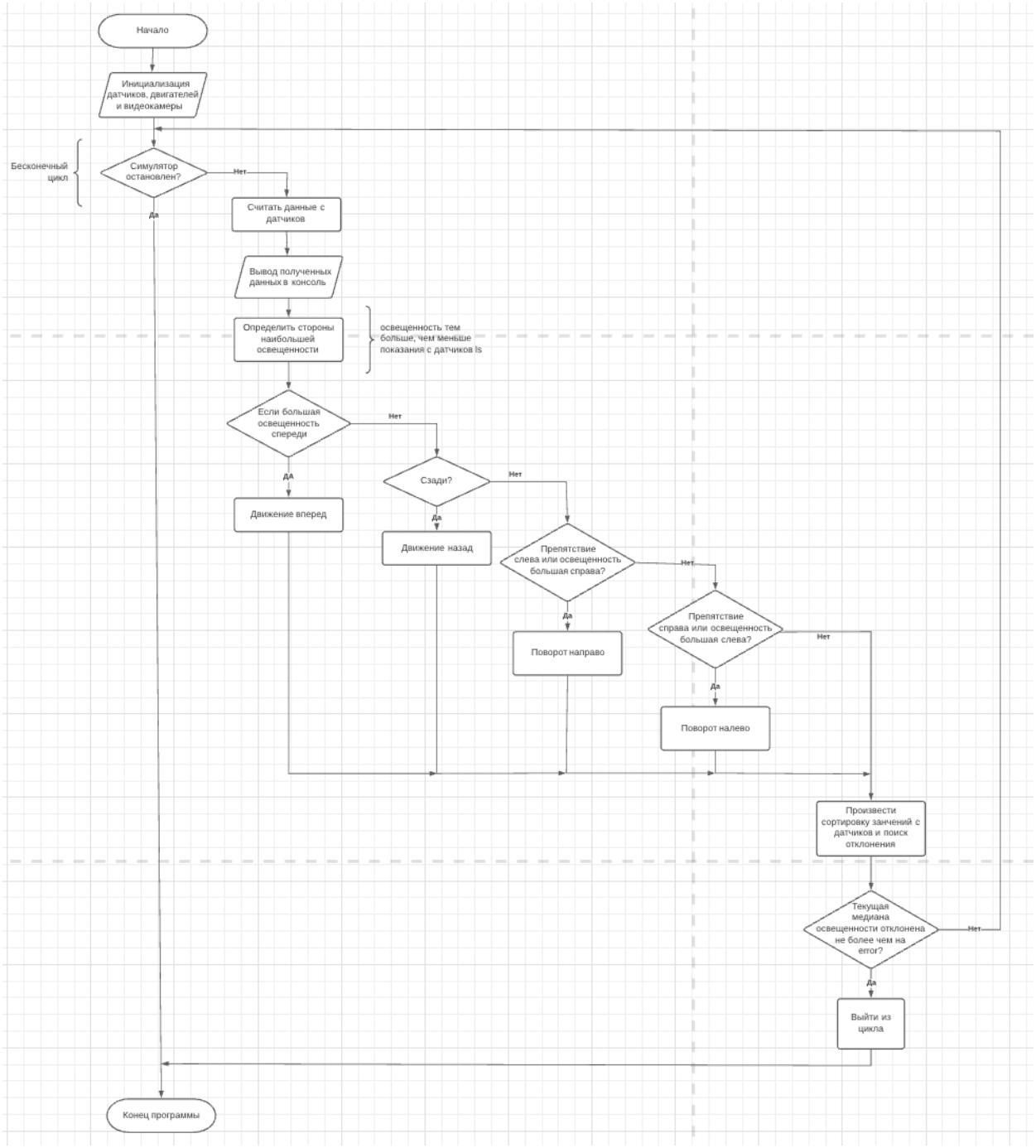


Рисунок 12. Алгоритм работы робота. Навигация по освещенности.

И теперь взглянем на пример кода-листинга полученной реализации.

```
"""Pr_work_3_controller_Petrov controller."""
from controller import Robot, DistanceSensor, LightSensor, Motor
from math import fabs, pi

# Создаём объект класса Robot
robot = Robot()

MAX_SPEED = 6.28 # максимальная скорость вращения колеса (рад/с)
WHEEL_RAD = 0.0205 # радиус колеса (м)
AXLE_LENGTH = 0.052 # длина оси конструкции робота (м)
```

```
11.
12. # Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В
    ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ 1
13. TIME_STEP = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит
    тип данных в целочисленный
14.
15. ps = []  # создаем список для хранения имен датчиков приближения
16. # объявляем имена датчиков приближения (заданы в документации к роботу)
17. psNames = [
18.     'ps0', 'ps1', 'ps2', 'ps3',
19.     'ps4', 'ps5', 'ps6', 'ps7'
20. ]
21. ls = [] # создаем список для хранения имен датчиков освещенности
22. # объявляем имена датчиков приближения (заданы в документации к роботу)
23. lsNames = [
24.     'ls0', 'ls1', 'ls2', 'ls3',
25.     'ls4', 'ls5', 'ls6', 'ls7'
26. ]
27.
28. def delay(delay_ms): #объявляем функцию, аргумент - переменная delay_ms - в мс.
29.     control_time = robot.getTime()
30.     while robot.step(TIME_STEP) != -1:          # пока программа не остановлена...
    Функция step() не может вернуть -1.
31.         if robot.getTime() - control_time > delay_ms / 904:  # если разница текущего и
    замеренного
32.             break                       # если прошло времени больше отмеренного - выходим
    из цикла
33.                                     # 904 - константа, полученная из 1000 мс - 3 * 32 мс.
34.                                     # В теле функции задержки 3 раза вызывается getTime()
    и step(TIME_STEP)
35.
36. def stop():                 # создадим для удобства функцию остановки робота
37.     leftMotor.setVelocity(0)
38.     rightMotor.setVelocity(0)
39.
40. def move_straight(s, coef = 25):
41.     t = abs( s / ( coef * 0.01 * MAX_SPEED * WHEEL_RAD ) ) # расчет времени
    движения в секундах
42.     leftMotor.setVelocity(coef * 0.01 * MAX_SPEED)        # если коэф. < 0, то
43.     rightMotor.setVelocity(coef * 0.01 * MAX_SPEED)       # движение назад,иначе -
    вперед
44.     delay(t * 1000)  # расчитанное время - в секундах, а функция delay принимает
    миллисекунды
45.     stop() # вызов функции остановки
46.
47. def turn_in_place(angle, coef = 25):    # Здесь второй аргумент является заданным по
    умолчанию.
48.     t = abs(angle) * AXLE_LENGTH  * pi/ (3.6 * WHEEL_RAD  * coef * MAX_SPEED) #
    Это означает, что поворот по умолчанию
49.                                                 # происходит со скоростью 25 % от
    максимальной.
50.     velocity = coef * MAX_SPEED / 100
```

```
    if angle < 0:                    # За направление поворота отвечает угол.
        leftMotor.setVelocity(-velocity)  # Если он отрицательный, то поворот против часовой стрелки.
        rightMotor.setVelocity(velocity)
    else:
        leftMotor.setVelocity(velocity)   # Иначе - по часовой.
        rightMotor.setVelocity(-velocity)
    delay(1000 * t)
    stop()

def median(lst): # функция определения медианного значения в списке
    n = len(lst)
    s = sorted(lst)
    return (sum(s[n//2-1:n//2+1])/2.0, s[n//2])[n % 2] if n else None

for i in range(8):
    ps.append(robot.getDevice(psNames[i]))  # добавляем по очереди имя датчика в список ps
    ls.append(robot.getDevice(lsNames[i]))  # добавляем по очереди имя датчика в список ls
    ps[i].enable(TIME_STEP) # ждем один такт времени симуляции для установки датчиков препятствий
    ls[i].enable(TIME_STEP) # то же самое для датчиков освещенности

leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

leftMotor.setPosition(float('inf')) # Разрешает поворачиваться колесу на полный оборот
rightMotor.setPosition(float('inf')) # аналогично для другого мотора
leftMotor.setVelocity(0.0)    # устанавливает скорость вращения колеса 0 (рад/с)
rightMotor.setVelocity(0.0)   # аналогично для другого мотора

leftSpeed  = 0
rightSpeed = 0
K = 0.5 # коэффициент скорости
error = 1 #% процентное отклонение от медианы

def check_if_reached(list, errorVal): # функция проверки завершенности поиска позиции с самой высокой освещенностью
    list.sort()
    check = False
    thisCheck = [False, False, False, False ,False, False, False, False]
    print(list)
    print(median(list))
    for i in range(8):
        if abs(list[i] - median(list)) < errorVal * median(list)/100:
            thisCheck[i] = True
    if not False in thisCheck:
            stop()
            check = True
    return check
```

```
97.
98. # Main loop: - бесконечный цикл
99. # -выполняет шаги симуляции, пока Webots не остановит контроллер
100.        while robot.step(TIME_STEP) != -1:
101.
102.            # считываем данные с датчиков
103.            psValues = []
104.            lsValues = []
105.            for i in range(8):
106.                psValues.append(ps[i].getValue())
107.                lsValues.append(ls[i].getValue())
108.            print(lsValues)   #выводим значения с датчиков в монитор порта
109.            print(lsValues.index(min(lsValues)))  #выводим индекс минимального значения
    с датчика в монитор порта
110.            #если источник света справа
111.            right_light =  lsValues[1] == lsValues[lsValues.index(min(lsValues))] or
    lsValues[2] == lsValues[lsValues.index(min(lsValues))]
112.            #если источник света слева
113.            left_light = lsValues[5] == lsValues[lsValues.index(min(lsValues))] or lsValues[6]
    == lsValues[lsValues.index(min(lsValues))]
114.            #если источник света спереди
115.            light_towards = lsValues[0] == lsValues[lsValues.index(min(lsValues))] or
    lsValues[7] == lsValues[lsValues.index(min(lsValues))]
116.            #если источник света сзади
117.            light_backwards = lsValues[3] == lsValues[lsValues.index(min(lsValues))] or
    lsValues[4] == lsValues[lsValues.index(min(lsValues))]
118.            # определяем, с какой стороны препятствие
119.            right_obstacle = psValues[0] > 80.0 or psValues[1] > 80.0 or psValues[2] > 80.0
120.            left_obstacle = psValues[5] > 80.0 or psValues[6] > 80.0 or psValues[7] > 80.0
121.            #выведем показания определителя стороны большей освещенности
122.            print(left_light, right_light,light_towards, light_backwards )
123.            #реакция робота на окуржающую обстановку
124.            if not left_obstacle and not right_obstacle:
125.                if light_towards:
126.                    leftSpeed  = K * MAX_SPEED
127.                    rightSpeed = K * MAX_SPEED
128.                if light_backwards:
129.                    leftSpeed  = -K * MAX_SPEED
130.                    rightSpeed = -K * MAX_SPEED
131.                if right_light or left_obstacle:
132.                    leftSpeed  = -K * MAX_SPEED
133.                    rightSpeed = K * MAX_SPEED
134.                if left_light or right_obstacle:
135.                    leftSpeed  = K * MAX_SPEED
136.                    rightSpeed = -K * MAX_SPEED
137.                 # устанавливаем скорость
138.                leftMotor.setVelocity(leftSpeed)
139.                rightMotor.setVelocity(rightSpeed)
140.            #реакция робота на обнаруженной препятствие по сторонам
141.            else:
142.                if left_obstacle: #если препятствие замечено слева - повернуть направо,
    отъехать вперед, развернуться
```

```
143.                 print("There is an obstacle on the left!")
144.                 turn_in_place(45)
145.                 move_straight(0.04) # 0.04 м. - расстояние до опознаваемого
    препятствия в соотв. с графиком датчика препятствий
146.                 turn_in_place(-45)
147.             if right_obstacle: #если препятствие замечено справа - повернуть налево,
    отъехать вперед, развернуться
148.                 # поворот налево
149.                 print("There is an obstacle on the right!")
150.                 turn_in_place(-45)
151.                 move_straight(0.04)
152.                 turn_in_place(45)
153.         #проверим, найден ли источник света: если да - выход из цикла программы
154.         if check_if_reached(lsValues, error):
155.             print ("The most lit point found!")
156.             break
```

Порядок выполнения работы

1. Запустить Webots.
2. Ознакомиться с реализацией алгоритма поиска источника освещенности в ограниченном пространстве, приведенной в разделе «Краткие теоретические сведения».
3. Подготовить мир симуляции в соответствии с заданием.
4. Реализовать алгоритм движения робота e-Puck на языке Python в соответствии с заданием.

Задание

Необходимо создать и настроить мир симуляции, в соответствии с разделом «Краткие теоретические сведения» и картой препятствий, изображенной на рисунке 13.

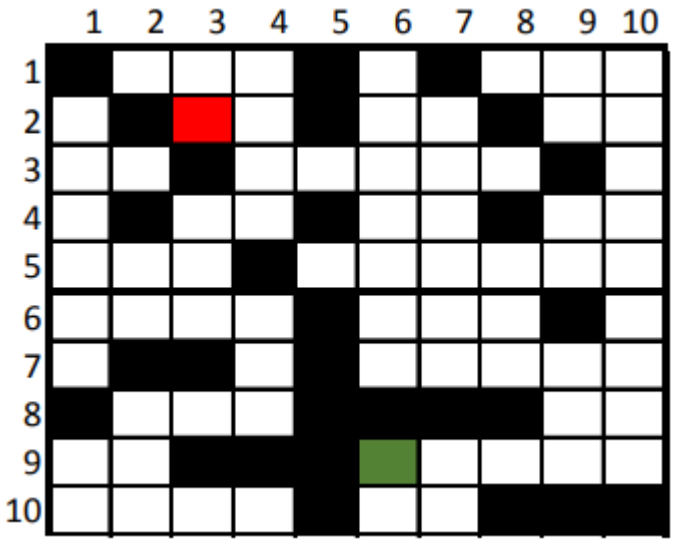


Рисунок 13. Карта препятствий.

Точка старта движения робота e-puck находится в зеленой клетке. Точка расположения точечного источника света – точка, куда должен следовать робот по ходу выполнения программы, находится в красной клетке. Черные клетки обозначают ящики-препятствия.

Требуется написать программу, позволяющую автономно производить поиск центра освещенности арены в замкнутом пространстве для робота e-puck на языке Python и настроить параметры движения робота, так, чтобы обеспечить его следование от точки старта до искомой точки, не касаясь объектов-препятствий. Допускается использовать функции, разработанные в предыдущих лабораторных работах для передвижения, при условии, что в алгоритме задействованы датчики приближения и датчики освещенности.

Содержание отчета по работе

Отчет должен содержать выполненные следующие пункты:

1. Составленная блок-схема программы.
2. Настроенный мир симуляции для выполнения задания.
3. Код-листинг программы по заданию.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Webots.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора.
3. Изложить материал раздела «Краткие теоретические сведения» и проделать практические предписания.
4. Ответить на вопросы.
5. Озвучить задание лабораторной работы.
6. После завершения работы проверить успешность выполнения учащимися самостоятельного задания.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учащимся.
2. Проверить работу моделирования в симуляторе – в случае ошибок, исправить программу вместе с учащимся.

Лабораторная работа №4 «Изучение принципов работы и применение инерциальных датчиков в робототехнике»

Цель работы

1. Настроить мир и контроллер для практической работы;
2. ознакомиться с теоретическими сведениями о принципах работы инерциальных датчиков и их применениях;
3. ознакомиться с реализацией алгоритма определения скорости подвижной платформы в момент столкновения с препятствием для робота e-puck на языке Python;
4. реализовать и оптимизировать алгоритм определения неподвижности препятствия при столкновении для робота e-puck на языке Python;
5. подготовить блок-схему полученного алгоритма.

Краткие теоретические сведения

Симуляция работы инерциальных датчиков в Webots

Симулятор Webots представляет собой систему с конфигурируемой окружающей средой и физикой объектов моделирования, что позволяет обеспечить синергетическую взаимосвязь этой среды с роботизированными средствами, делая возможным моделировать поведение различных устройств в разных окружающих средах.

Так, например, благодаря тонкой настройки явлений механики окружающего пространства (виртуального мира симуляции) становится возможным применение широко используемых в технике инерциальных датчиков.

Инерциальные датчики – измерительные электронные устройства, способные измерять с достаточно высокой точностью параметры механического воздействия посредством эффектов инерции. Пример инерциальных датчиков: акселерометр, гироскоп, магнитометр.

Акселерометр — это прибор, измеряющий проекцию ускорения. Типичный акселерометр состоит из трех взаимно перпендикулярных измерительных осей, регистрирующих гравитационное и линейные ускорения. Акселерометр

использует для каждой оси отдельную пробную массу, которая смещается при возникновении ускорения вдоль данной оси. Смещение фиксируются емкостными датчиками. Архитектура акселерометра снижает подверженность температурному дрейфу и вариациям электрических параметров. При расположении устройства на плоской поверхности оно измерит 0g по X- и Y-осям и +1g по Z-оси. Масштабный коэффициент (scale factor — отношение изменения выходного сигнала к изменению выходного измеряемого сигнала) калибруется на заводе и не зависит от напряжения питания.

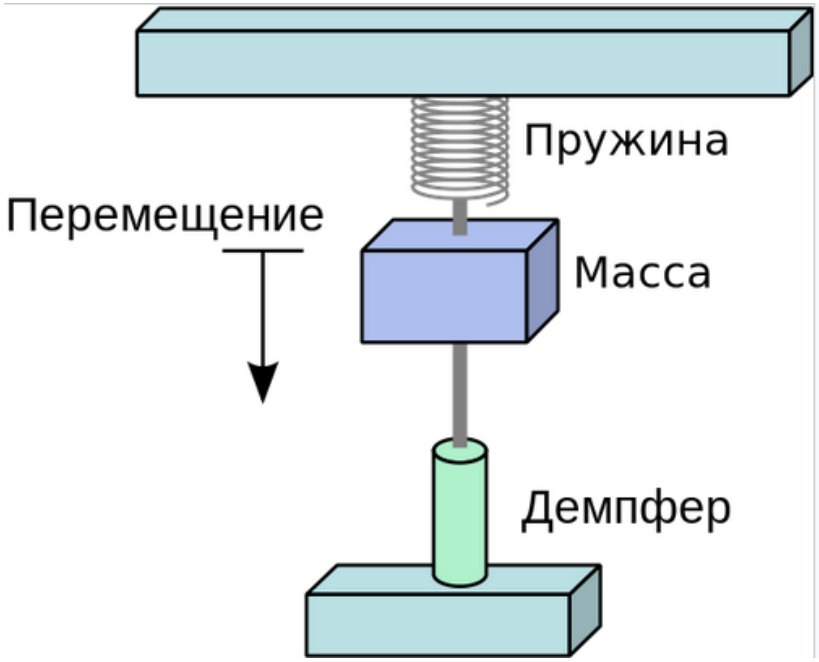


Рисунок 1. Схема простейшего акселерометра (одномерная модель).

Как видно на рисунке 1 выше модель имеет в своем составе некоторый груз, подвешенный на пружине, а также, демпфер для подавления колебаний груза. Чем больше кажущееся ускорение, тем сильнее деформируется пружина, изменяя показания прибора.

Модель описываемого выше принципа работы может быть представлена на рисунке 2.

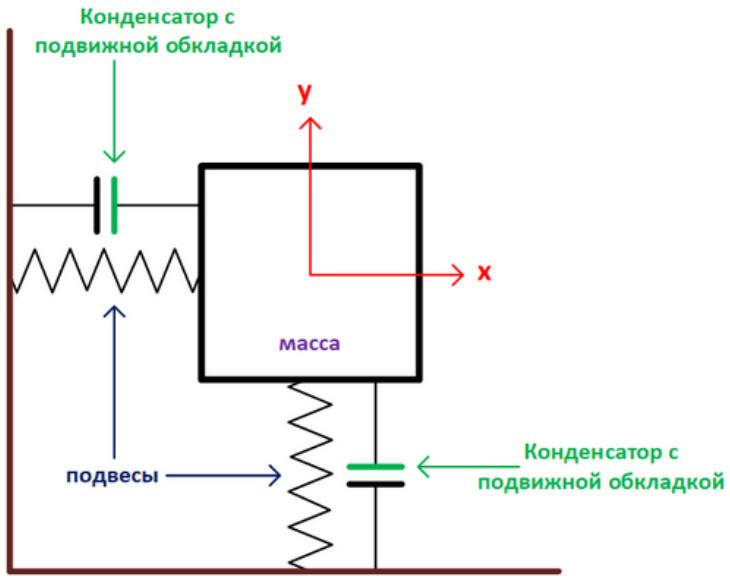


Рисунок 2. Модель 2-осевого емкостного акселерометра.

В реальности 3-х осевой акселерометр представляет собой более сложную конструкцию, как например, на рисунке 3 изображен 3-х осевой акселерометр, выполненный по технологии TSMC.

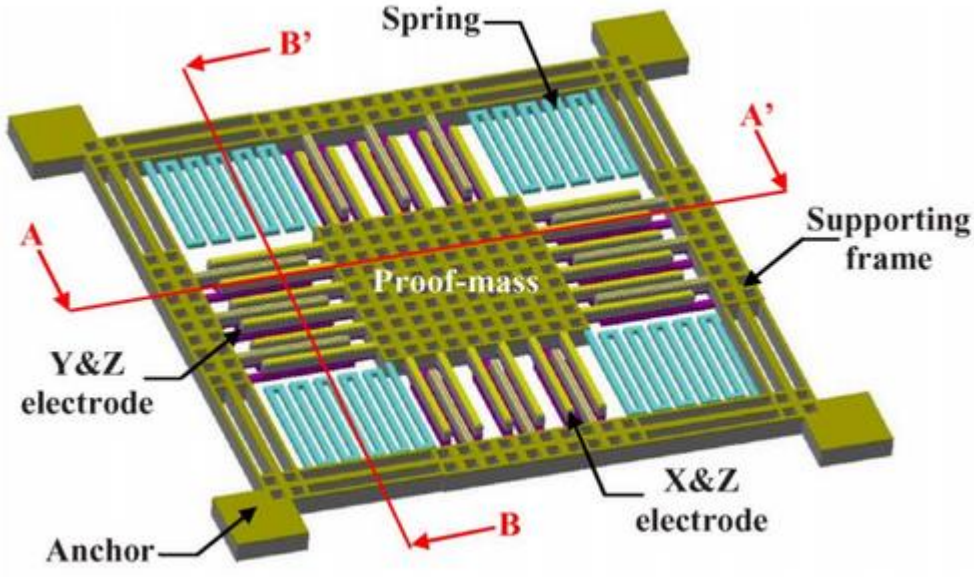


Рисунок 3. Трехосевой емкостной полумост от TSMC

Гироскоп - устройство, способное реагировать на изменение углов ориентации тела, на котором оно установлено, относительно инерциальной системы отсчёта. Гироскоп состоит из трёх независимых одноосных вибрационных датчиков угловой скорости (MEMS гироскопов), которые

реагируют на вращение вокруг X-, Y-, Z- осей. Две подвешенные массы совершают колебания по противоположным осям. С появлением угловой скорости эффект Кориолиса вызывает изменение направления вибрации, которое фиксируется емкостным датчиком. Измеряемая дифференциальная емкостная составляющая пропорциональна углу перемещения. Получившийся сигнал усиливается, демодулируется и фильтруется, давая в итоге напряжение, пропорциональное угловой скорости вращения.

Магнитометр — (от гр. μαγνητό — магнит + гр. μετρεω измеряю), прибор для измерения характеристик магнитного поля и магнитных свойств материалов. Принцип работы основан на высокоточной технологии эффекта Холла. Типичный магнитометр включает в себя магнитные сенсоры, определяющие напряжённость магнитного поля земли по осям, схему управления, цепь усиления сигнала и вычислительную схему для обработки сигналов с каждого датчика.

При помещении в магнитное поле пластины-проводника или полупроводника под 90 °к направлению силовых линий магнитного потока произойдет перемещение электронов по поперечине пластины под действием силы Лоренца. Их направление зависит от того, в какую сторону идет сила тока и силовые линии магнитного потока. Иначе говоря, (ЭХ) эффект Холла – это частный случай действия силы Лоренца, то есть действия магнитного поля на заряженную частицу.

На практике эти датчики редко используются сами по себе для решения определенных технических задач, таких как инерционная навигация, активно применяемая в средствах противовоздушной обороны и авиации, регистрация инерционных эффектов как косвенных измеряемых факторов событий (например, крушение воздушного или наземного агрегата) и многие другие.

Помимо традиционной и хорошо изученной задачи определения ориентации устройства, инерциальные датчики могут использоваться для:

1. Сбора данных о магнитной карте помещения. Пример такой карты приведен на рисунке ниже. Видно, что в различных частях помещения изменяется не только магнитуда магнитного поля (в мкТл), но и направление вектора магнитной индукции (обозначено розовой линией). Такую карту можно использовать для уточненного позиционирования объекта в дополнении к традиционным картам радиосигнала.

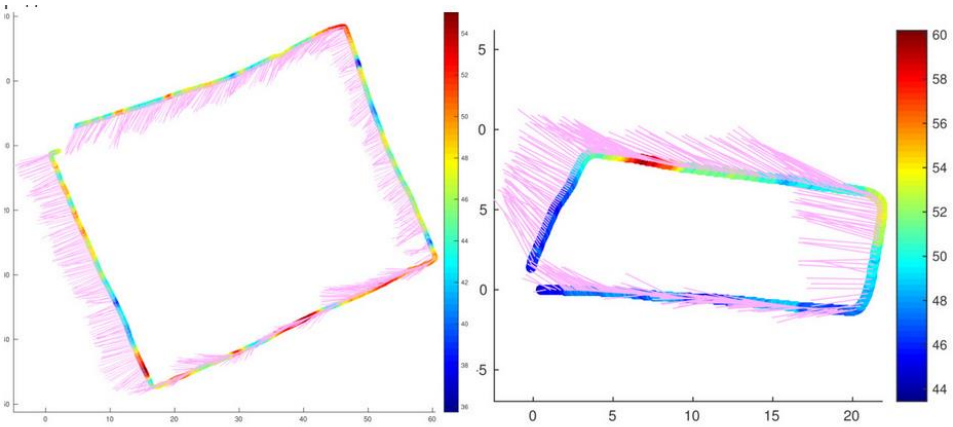


Рисунок 4. Магнитная карта перемещений в пространстве.

2. Восстановления траектории движения объекта. Таким объектом может быть пешеход или автомобиль. В отдельных случаях, например при креплении устройства на ноге и предварительной точной калибровке датчиков можно добиться ошибки возврата в точку начала движения, не превышающую десятков сантиметров для длины пути, превышающей 100 метров. Пример восстановленной траектории методом ZUPT(при сбросе ошибки в периоды неподвижности), дополненным измерениями датчика атмосферного давления приведен на следующем рисунке (траектория движения включала в себя проход по коридору, спуск по лестнице, еще один проход и подъем на лифте).

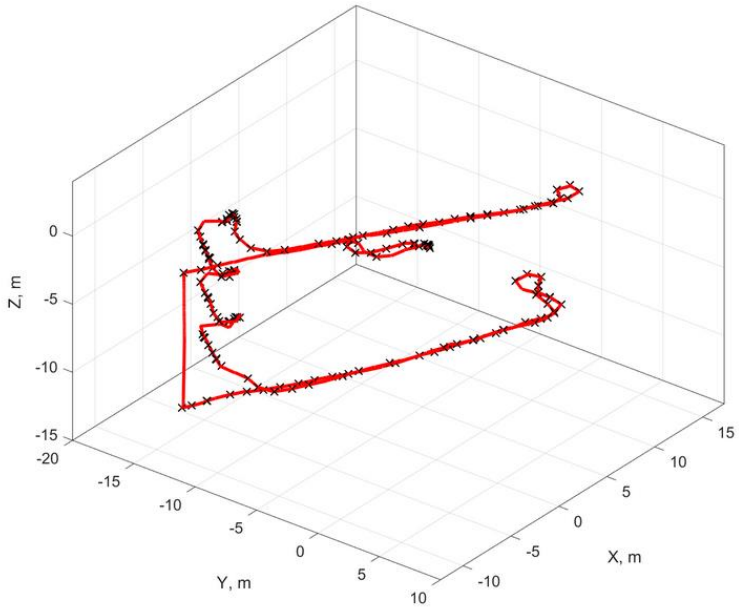


Рисунок 5. Восстановленная карта траектории пройденного пути.

3. При произвольном креплении устройства на теле человека ошибка возврата к исходной точке, как правило, куда больше и составляет 15-20% от пройденной дистанции. Такое её значение обусловлено, во-первых, ошибкой в определении длины шага, а во-вторых, ошибкой в определении направления движения.
4. Инициирования каких-либо событий или управления устройством. Это возможно сделать при помощи "рисования" устройством какой-либо фигуры или образа в воздухе, например символ ∞ может использоваться для запуска калибровки магнитометра, продольные взмахи устройством — для генерации экстренного сообщения, тройной "тап" — для выключения. Данные задачи решаются при помощи заранее обученных классификаторов.
5. Определения текущей активности пользователя. Например, при использовании устройств в офисе может быть полезным знание о том, насколько много человек двигался в течение рабочего дня и типе движения — какую часть времени он провел стоя, сидя, сколько времени потратил на различные переходы по зданию.

В данной работе речь пойдет о применении акселерометра и гироскопа для решения различных задач в мобильной робототехнике.

Робот e-puck обладает инфракрасными датчиками приближения, датчиками света и светодиодами для индикации работоспособности датчиков. Их расположение и ориентация показаны на рисунке 6.

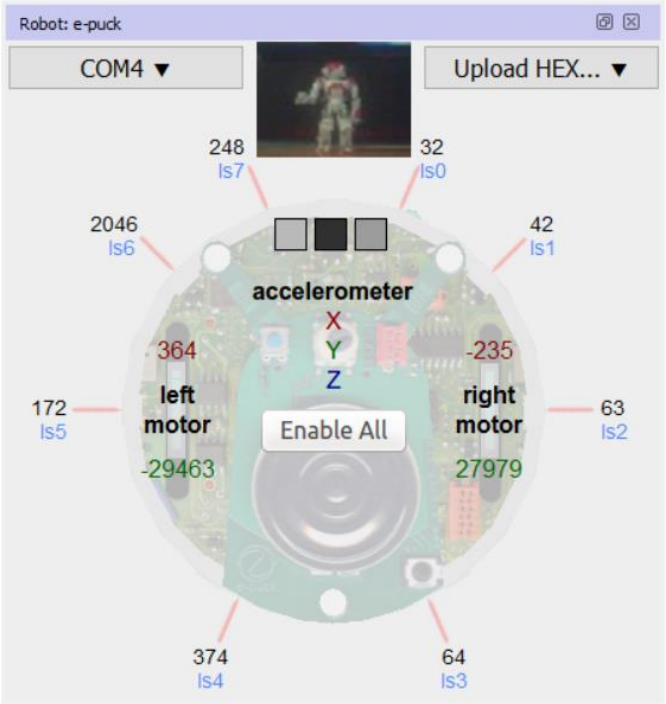


Рисунок 6. Расположение датчиков и сенсоров на роботе e-puck.

Для ознакомления с периферическим функционалом робота e-puck, можно обратить внимание на рисунок 7.

Device	Name
Motors	'left wheel motor' and 'right wheel motor'
Position sensors	'left wheel sensor' and 'right wheel sensor'
Proximity sensors	'ps0' to 'ps7'
Light sensors	'ls0' to 'ls7'
LEDs	'led0' to 'led7' (e-puck ring), 'led8' (body) and 'led9' (front)
Camera	'camera'
Accelerometer	'accelerometer'
Gyro	'gyro'
Ground sensors (extension)	'gs0', 'gs1' and 'gs2'
Speaker	'speaker'

Рисунок 7. Таблица с названиями устройств на роботе.

Напоминаем, что весьма полезным является более подробное самостоятельное ознакомление с функциональными возможностями робота e-puck по ссылке.

Создание программы-контроллера на языке Python

Создадим новый проект также, как это было в лабораторной работе 1. При этом назовите проект «Lab_work_4_*Ваша фамилия латиницей*», а название мира будет «Lab_4_world.wbt». Добавьте робота e-puck в мир симуляции.

Чтобы добавить контроллер, необходимо выбрать в ленте вкладку «Wizards», и нажать на вкладку в выпадающем меню «New Robot Controller…», как на рисунке 8.

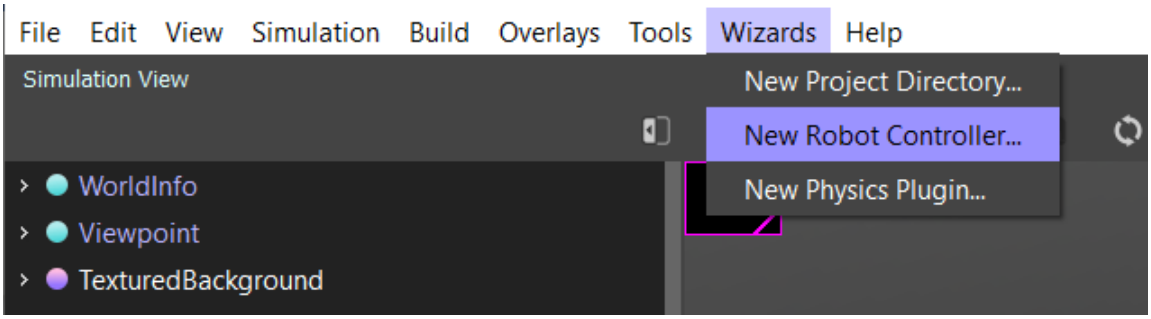


Рисунок 8. Создание нового контроллера.

Откроется диалоговое окно, нажмите «Next», чтобы продолжить. Следующая страница требует выбора языка программирования – выбираем Python и нажимаем далее (Next). В следующем окне – называем контроллер так: «Lab _4_controller_*Ваша фамилия латиницей*», как показано на рисунке 9, и переходим в следующее окно.

Затем обязательно ставим галочку около «Open ‘Lab _4_controller_*Ваша фамилия латиницей*.py’ in Text Editor», чтобы открыть текст контроллера в редакторе и нажимаем «Finish».

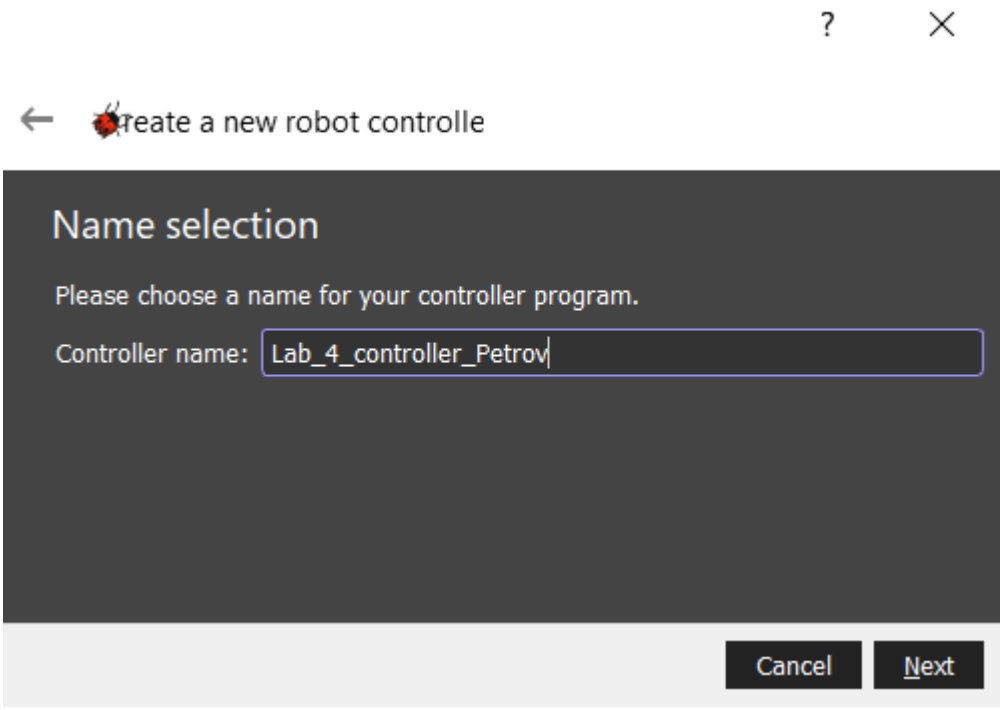


Рисунок 9. Завершение создания нового контроллера.

Теперь необходимо подключить контроллер к роботу e-puck, чтобы мы смогли сразу тестировать разрабатываемые программы.

Для этого необходимо выбрать в дереве объектов робота, найти поле «Controller» и нажать кнопку «select…», как на рисунке 10.

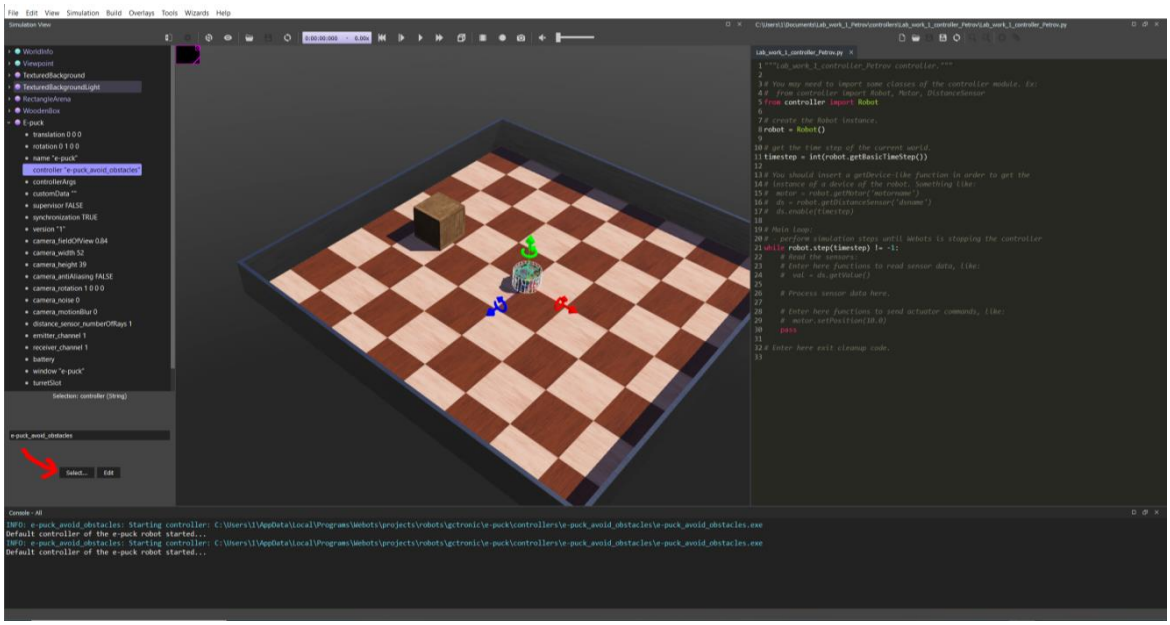


Рисунок 10. Подключение нового контроллера.

В диалоговом окне выбираем только что созданный контроллер с именем «Lab_4_controller_*Ваша фамилия латиницей*» и нажимаем «ОК».

Конфигурация мира симуляции в Webots

Для решения данной задачи нам будет необходимо добавить в мир симуляции объект «WoodenBox» и настроить его массу, сделав этот параметр равным единице, и габариты, как на рисунке 11.

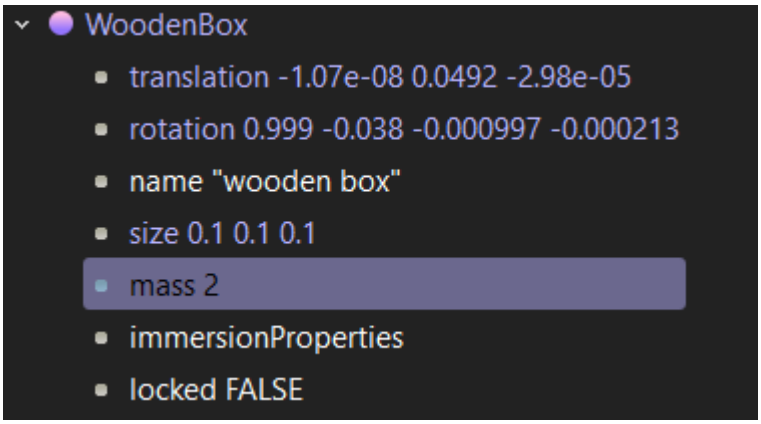


Рисунок 11. Настройка объекта «WoodenBox».

Далее следует расположить объект в центре арены, как показано на рисунке 12.

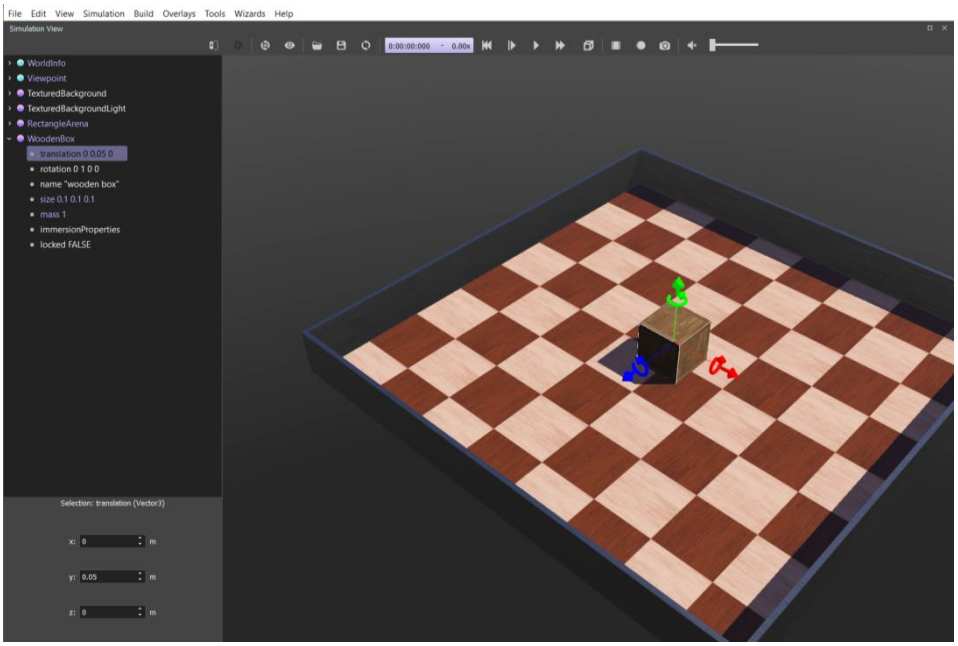


Рисунок 12. Настройка положения объекта «WoodenBox».

Перемещать объекты можно с помощью мыши, предварительно зажав клавишу Shift, а также, с помощью стрелок по осям, исходящим из выбранного объекта.

Копировать и вставлять объекты – с помощью сочетания клавиш (Ctrl+C, Ctrl+V).

Нажмите сочетание клавиш (Ctrl+Shift+S), или нажмите на иконку, графически схожую с дискетой  , в левой верхней части экрана для сохранения созданного мира.

На примере текущей практической работы напишем программу, реализующую алгоритм определения установившейся скорости платформы с помощью столкновения робота e-puck с препятствием на языке Python.

Разработка алгоритма определения скорости подвижной платформы в момент столкновения с препятствием для робота e-puck на языке Python

Идея алгоритма заключается в следующем: современные автомобильные краш-тесты проводятся под контролем многих измерительных систем, одной из которых в качестве дополняющей является и инерционная. Разогнав автомобиль до некоторой постоянной скорости, акселерометр будет «чувствовать» только силу реакции опоры по оси, перпендикулярной земле и небольшое ускорение торможения, связанное с трением вдоль вектора движения. Т. е. так как скорость, можно считать, остается постоянной, ускорение будет варьироваться около нулевого значения. А при столкновении платформы с препятствием произойдет резкая остановка платформы с некоторой скорости до нуля, а значит значение ускорения значительно возрастет. По величине такого возрастания можно судить об изначальной скорости, предшествующей мгновению столкновения.

Алгоритм будет заключаться в том, чтобы, запустив робота двигаться в прямом направлении на определенной скорости, считывать ускорение по оси направления движения робота с помощью акселерометра. Ввиду того, что таймер позволяет осуществлять чтение данных дискретно, то необходимо перейти к анализу значений ускорения в дискретных отсчетах.

Тогда можно представить определение скорости (1):

$$a = \frac{dv}{dt} \qquad (8)$$

следующим образом (2):

$$a = \frac{dv}{dt} = \frac{v_к - v_н}{t_к - t_н} \quad (9)$$

Где $v_к$ и $v_н$ – означают скорость в конечный момент времени и скорость в начальный момент времени, соответственно. $t_к$ и $t_н$ – конечный момент времени и начальный момент времени, соответственно.

Так как в любой величина $t_к - t_н$ – постоянна, и равна T – периоду считывания данных с датчиков, то выражение 2 примет вид:

$$a = \frac{v_к - v_н}{T} \quad (10)$$

Так как задача состоит в том, чтобы определить скорость до столкновения, то примем, возникновение реактивного ускорения при столкновении $a_р$ обуславливается ненулевой скоростью $v_н$ и $v_к = 0$.

В таком случае выразим из (3) искомую $v_н$:

$$v_н = -a_р \cdot T \quad (11)$$

Период считывания данных – TIME_STEP (подробнее о мире симуляции в лабораторной работе 1).

Теперь рассмотрим алгоритм программы (рисунок 14), позволяющей определять скорость столкновения мобильной платформы с препятствием.

Примите, также, к сведению наличие погрешности измерения (причины погрешности присутствуют в тексте практической работы).

Результат работы представлен на рисунке 13.

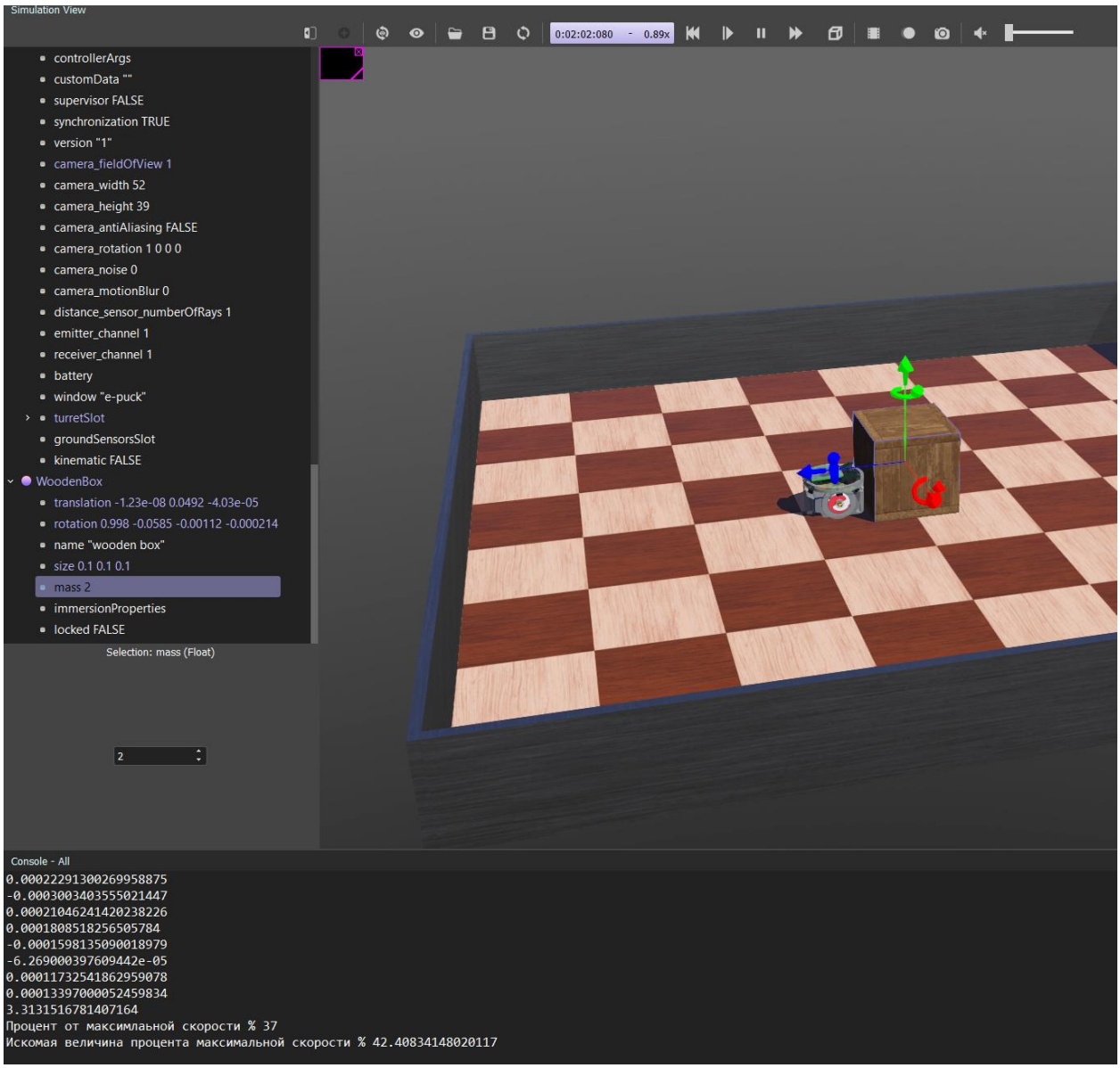


Рисунок 13. Результат работы программы.

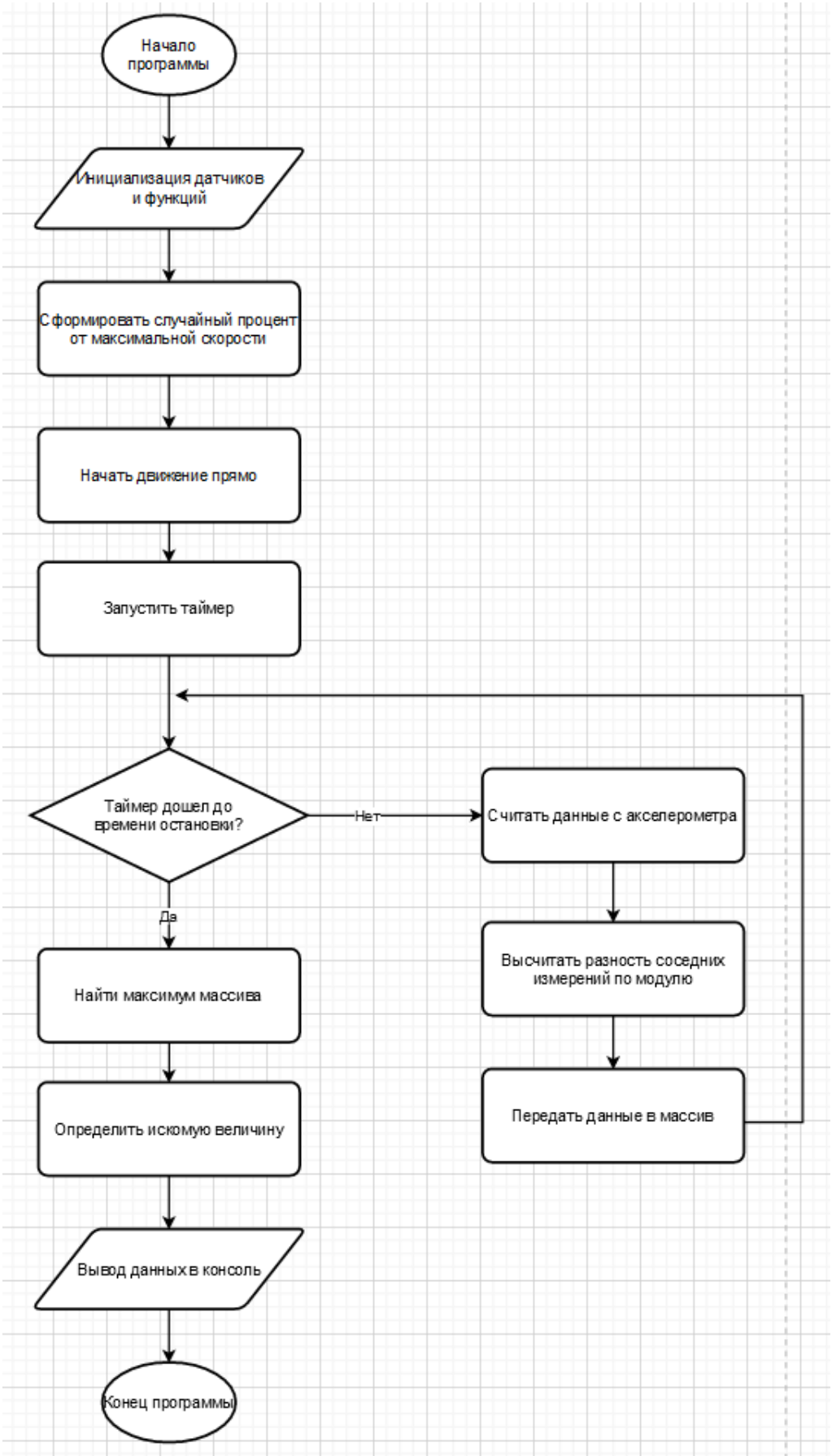


Рисунок 14. Алгоритм программы для определения скорости столкновения мобильной платформы с препятствием.

И теперь взглянем на код-листинг полученной реализации ниже.

```
"""Pr_work_4_controller_Petrov controller."""

# Вам может потребоваться импортировать некоторые классы модуля контроллера. Пример:
# from controller import Robot, Motor, DistanceSensor
# from math import fabs, pi - добавляет из библиотеки math функцию абсолютного значения числа и число Пи
from controller import Robot, Accelerometer
from math import fabs, pi, degrees
import random
# Создаём объект класса Robot
robot = Robot()

MAX_SPEED = 6.28 # максимальная скорость вращения колеса (рад/с)
WHEEL_RAD = 0.0205 # радиус колеса (м)
AXLE_LENGTH = 0.052 # длина оси конструкции робота (м)
MAX_VELOCITY = 0.25 # m/s

# Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ
TIME_STEP = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит тип данных в целочисленный

accel = robot.getDevice('accelerometer')

accel.enable(TIME_STEP)


def delay(delay_ms): #объявляем функцию, аргумент - переменная delay_ms - в мс.
    control_time = robot.getTime()
    accelVals = [0, 0, 0]
    array = []
    while robot.step(TIME_STEP) != -1:
        prev = accelVals[1]
        accelVals = accel.getValues()
        accelDif = accelVals[1] - prev
        array.append(abs(accelDif))
        print(accelDif)
        if robot.getTime() - control_time > delay_ms / 904:  # если разница текущего и замеренного больше установленного
            print(max(array))
            velocity_stable = max(array) / MAX_VELOCITY* TIME_STEP * 0.1
            print ("Процент от максимлаьной скорости %", percentOfVelocity)
            print ("Искомая величина процента максимальной скорости %",  velocity_stable if  velocity_stable <= 100 else 100)
            break  # если прошло времени больше установленного - выходим из цикла
                # 904 - константа, полученная из 1000 мс - 3 * 32 мс.

leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')
```

```
45.
46. leftMotor.setPosition(float('inf')) # Разрешает поворачиваться колесу на полный
    оборот
47. rightMotor.setPosition(float('inf')) # аналогично для другого мотора
48. leftMotor.setVelocity(0.0) #устанавливает скорость вращения колеса(рад/с)
49. rightMotor.setVelocity(0.0) # аналогично для другого мотора
50.
51. def stop():              # создадим для удобства функцию остановки робота
52.     leftMotor.setVelocity(0)
53.     rightMotor.setVelocity(0)
54.
55. def move_straight(s, coef = 25):
56.     t = abs( s / ( coef * 0.01 * MAX_SPEED * WHEEL_RAD ) ) # расчет времени
    движения в секундах
57.     leftMotor.setVelocity(coef * 0.01 * MAX_SPEED)       # если коэф. < 0, то
58.     rightMotor.setVelocity(coef * 0.01 * MAX_SPEED)      # движение назад,иначе -
    вперед
59.     delay(t * 1000)  # рассчитанное время - в секундах, а функция delay принимает
    миллисекунды
60.     stop() # вызов функции остановки
61.
62.
63. percentOfVelocity = random.randint(10, 100)
64. move_straight(0.3, percentOfVelocity) #0.3 метров - путь прямолинейного движения
65. # Main loop: - бесконечный цикл
66. # -выполняет шаги симуляции, пока Webots не остановит контроллер
67. while robot.step(TIME_STEP) != -1:
68.     pass
```

Порядок выполнения работы

1. Запустить Webots.
2. Ознакомиться с реализацией алгоритма определения скорости подвижной платформы в момент столкновения с препятствием, приведенной в разделе «Краткие теоретические сведения».
3. Подготовить мир симуляции в соответствии с заданием.
4. Реализовать алгоритм движения робота e-Puck на языке Python в соответствии с заданием.

Задание

Необходимо создать и настроить мир симуляции и программу-контроллер, в соответствии с разделом «Краткие теоретические сведения».

Требуется написать программу и оптимизировать алгоритм определения неподвижности препятствия при столкновении для робота e-puck на языке Python. Иначе говоря, определить по столкновению робота с препятствием,

остается ли оно неподвижным, или возможно его перемещение под механическим воздействием со стороны робота e-puck. Подсказка: зная изначальную скорость, предшествующую моменту столкновения по изменению ускорения можно судить о перемещаемости препятствия – если рассчитанная скорость меньше номинальной – перемещение возможно, больше или равно – перемещения препятствия невозможны.

Результат программы – вывести в консоль в виде сообщения.

Примечание: необходимо настроить массу объекта «WoodenBox», выступающую в качестве препятствия в соответствии с рисунком 15, поле «Weight», характеризующее массу робота. Очевидно, что многократное превышение массы объекта-препятствия над массой робота не позволит зафиксировать никакие перемещена препятствия при столкновении.

Feature	Description
Size	7.4 cm in diameter, 4.5 cm high
Weight	150 g

Рисунок 15. Таблица массогабаритных параметров робота e-puck.

Содержание отчета по работе

Отчет должен содержать выполненные следующие пункты:

1. Составленная блок-схема программы.
2. Настроенный мир симуляции для выполнения задания.
3. Код-листинг программы по заданию.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Webots.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора.
3. Изложить материал раздела «Краткие теоретические сведения» и проделать практические предписания.
4. Ответить на вопросы.
5. Озвучить задание лабораторной работы.
6. После завершения работы проверить успешность выполнения учащимися самостоятельного задания.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учащимся.
2. Проверить работу моделирования в симуляторе – в случае ошибок, исправить программу вместе с учащимся.

Лабораторная работа №5 «Изучение принципов работы и применение оптических датчиков в робототехнике»

Цель работы

1. Настроить мир и контроллер для практической работы;
2. ознакомиться с теоретическими сведениями о принципах работы оптических датчиков и их применениях;
3. ознакомиться с реализацией алгоритма движения мобильной платформы по контрастной линии для робота e-puck на языке Python;
4. подготовить блок-схему полученного алгоритма.

Краткие теоретические сведения

Симуляция работы оптических датчиков в Webots

Симулятор Webots представляет собой систему с конфигурируемой окружающей средой и физикой объектов моделирования, что позволяет обеспечить синергетическую взаимосвязь этой среды с роботизированными средствами, позволяя моделировать поведение различных устройств в разных окружающих средах.

В данной работе речь пойдет о применении датчиков линии (оптические инфракрасные датчики) для решения различных задач в мобильной робототехнике.

Если на полу размечена контрастная линия достаточной ширины, она может быть использована в качестве цветографического изображения траектории движения, по которой может следовать робот. Данный вид управления зачастую применяется мобильными роботами в складской транспортной логистике с целью упрощения алгоритма движения и обеспечения наиболее эффективного логистического процесса. Следует отметить, что одной из главных отраслей, где используется датчик линии является робототехника. Помимо логистики, существует профессиональная коммерческая сфера спортивно-развлекательных мероприятий, в которой применяется данный алгоритм движения роботов, как основной. Даже устраиваются целые соревнования, в рамках гонок роботов, по разработке лучшего алгоритма и достижении скорости прохождения маршрута

роботом. В таких проектах, как правило, используется не один, а группа датчиков, с помощью которых робот определяет границы маршрута. Помимо роботов для соревнований, подобные датчики применяются и в бытовых роботах-пылесосах.

Робот e-puck обладает слотом для размещения разных датчиков, помимо уже встроенных, таких как: инфракрасные датчики приближения, датчики света и светодиоды для индикации работоспособности датчиков. Расположение штатных датчиков и их

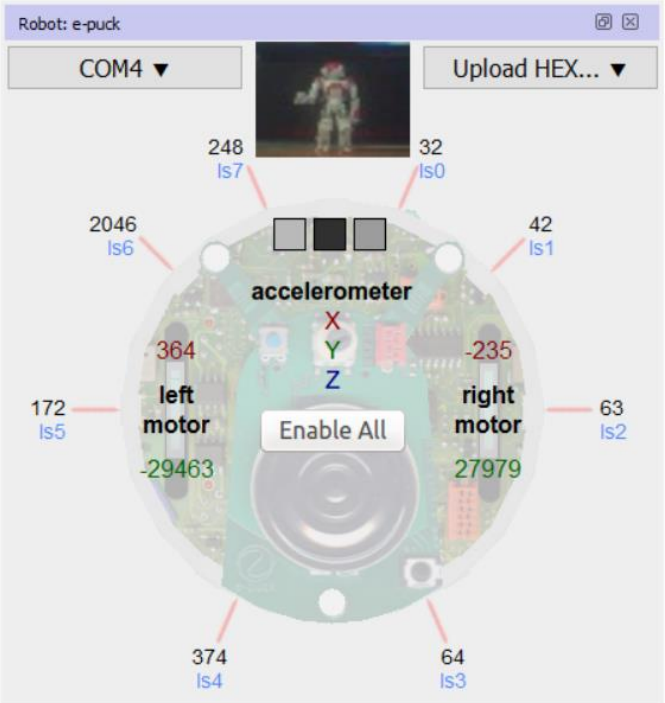


Рисунок 1. Расположение датчиков и сенсоров на роботе e-puck.

Однако в упомянутый слот, именуемый «groundSensorSlot», есть возможность установки других датчиков (и не только), которые перечислены на рисунке 2.

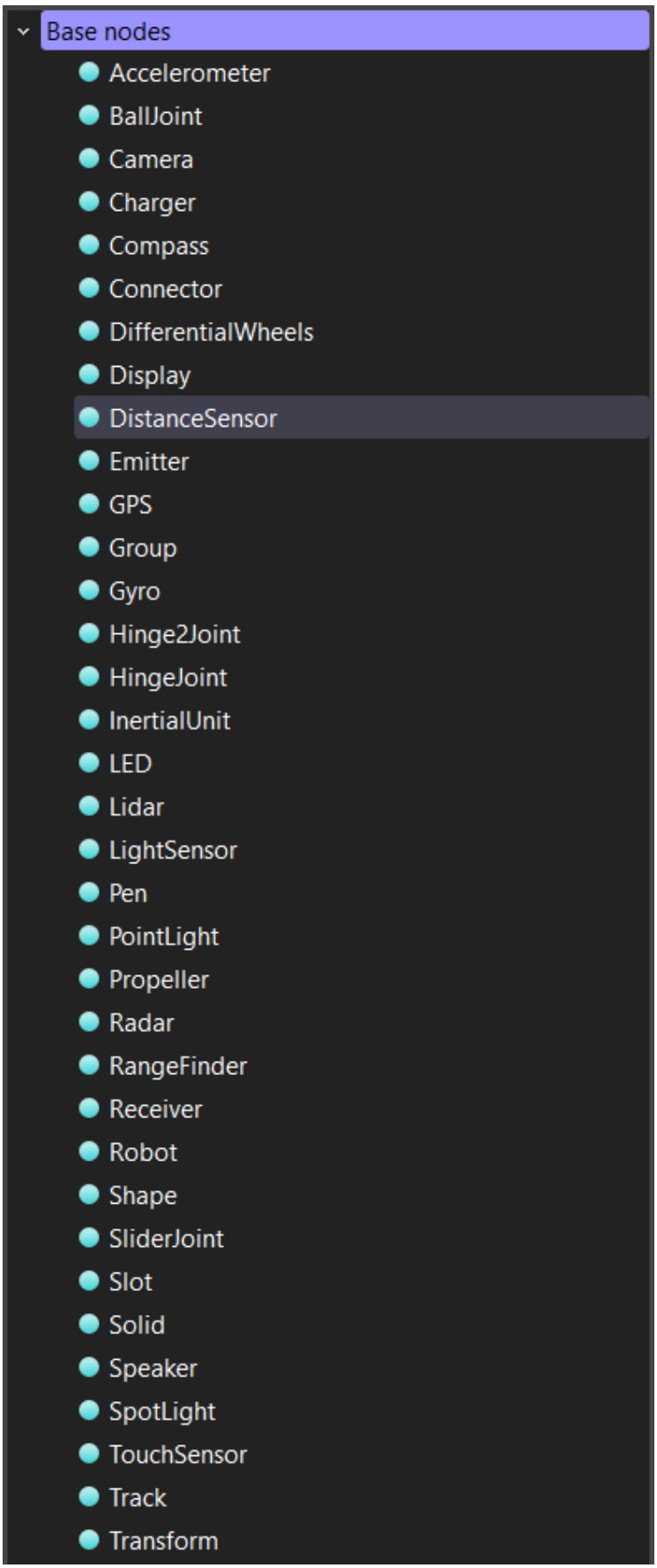


Рисунок 2. Возможные устройства для размещения в слоте «groundSensorSlot», на роботе e-puck.

Для ознакомления с периферическим функционалом робота e-puck, можно обратить внимание на рисунок 3.

Device	Name
Motors	'left wheel motor' and 'right wheel motor'
Position sensors	'left wheel sensor' and 'right wheel sensor'
Proximity sensors	'ps0' to 'ps7'
Light sensors	'ls0' to 'ls7'
LEDs	'led0' to 'led7' (e-puck ring), 'led8' (body) and 'led9' (front)
Camera	'camera'
Accelerometer	'accelerometer'
Gyro	'gyro'
Ground sensors (extension)	'gs0', 'gs1' and 'gs2'
Speaker	'speaker'

Рисунок 3. Таблица с названиями устройств на роботе.

Напоминаем, что весьма полезным является более подробно самостоятельно ознакомиться с функциональными возможностями робота e-puck по ссылке.

Датчик линии – это оптический модуль, предназначенный для обнаружения препятствий в виде белых или чёрных линий. Основным его элементом является оптопара, состоящая из инфракрасного светодиода и фототранзистора. Внешний вид датчика показан на рисунке 4.

Рисунок 4. Внешний датчика линии.

Согласно документации, инфракрасный диод излучает свет с длиной волны инфракрасного спектра (~ 950 нм), что позволяет достоверно определять препятствия на расстоянии от 1 мм до 25 мм. Также в состав модуля входят компаратор, подстроечный резистор и контрольный светодиод. Подстроечным резистором выбирается порог срабатывания датчика на разный оттенок чёрного цвета. При максимальном сопротивлении датчик сработает на сером оттенке, а при минимальном сопротивлении – только на чёрном. Факт срабатывания сопровождается загоранием контрольного светодиода и подачей логического нуля на сигнальный вывод датчика. Следует отметить, что для удобства монтажа, по обе стороны оптического элемента расположены два крепёжных отверстия. Это позволяет более точно позиционировать датчик или группу датчиков на необходимом расстоянии от препятствия. Для исключения взаимного влияния светодиода и фототранзистора они конструктивно разделены небольшой перегородкой.

Что касается принципа работы, то он очень прост. При подаче питания на модуль, инфракрасный светодиод начинает излучать свет, который отражаясь от белой поверхности попадает на фототранзистор. В таком режиме на выводе OUT (SIGNAL) будет установлена логическая единица. Как только в зону видимости датчика попадает чёрный объект, световой поток, поглощаясь этим самым объектом, перестаёт доходить до фототранзистора и компаратор переключает вывод OUT (SIGNAL) в логический ноль. Более наглядно этот процесс показан на рисунке 5 на примере распространенного датчика линии KY-033, имеющего в своем составе оптопару TCRT5000.

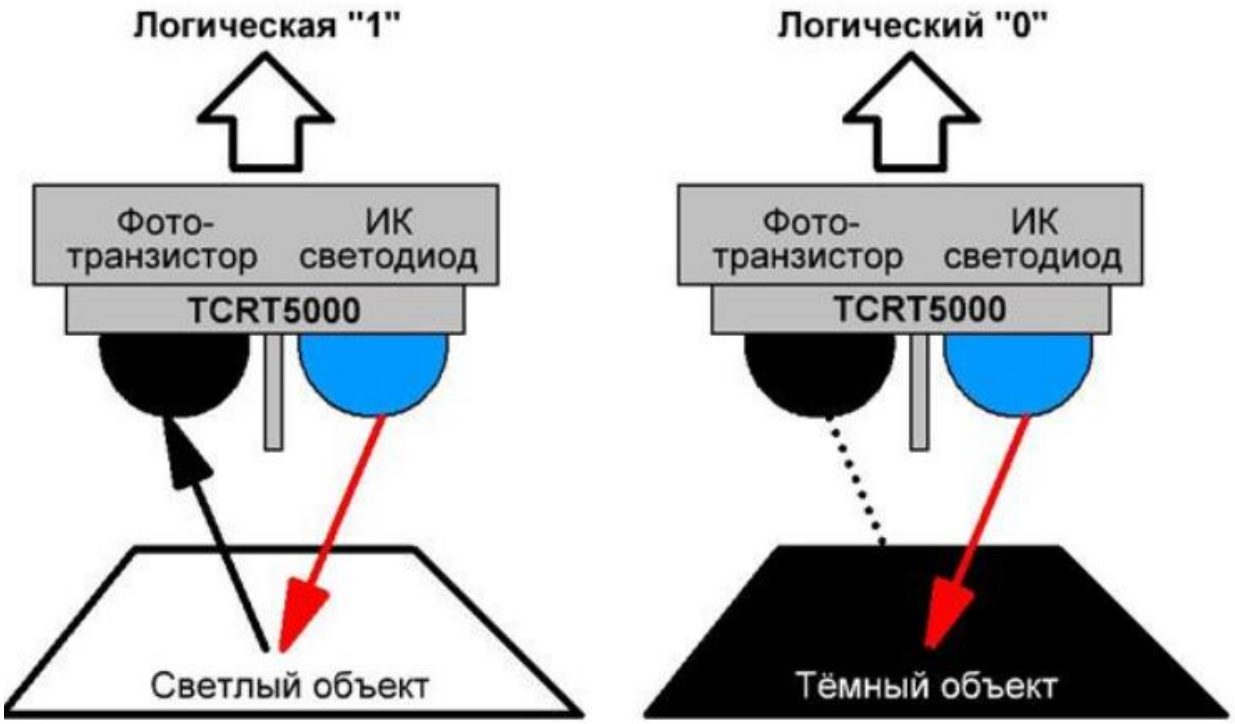


Рисунок 5. Логика работы датчика линии KY-033.

При проектировании устройств с использованием датчика линии, необходимо чётко следить за границами приближения/удаления модуля от исследуемой поверхности. Если фотоэлемент будет слишком приближен, то перегородка не даст пройти световому потоку от ИК-диода к фототранзистору, даже при самой белой поверхности. Та же ситуация произойдёт и при чрезмерном удалении фотоэлемента, так как весь световой поток попросту рассеется в воздушной среде, не дойдя до объекта-отражателя.

Создание программы-контроллера на языке Python

Создадим новый проект также, как это было в лабораторной работе 1. При этом назовите проект «Lab_work_5_*Ваша фамилия латиницей*», а название мира будет «Lab_5_world.wbt». Добавьте робота e-puck в мир симуляции.

Чтобы добавить контроллер, необходимо выбрать в ленте вкладку «Wizards», и нажать на вкладку в выпадающем меню «New Robot Controller…», как на рисунке 6.

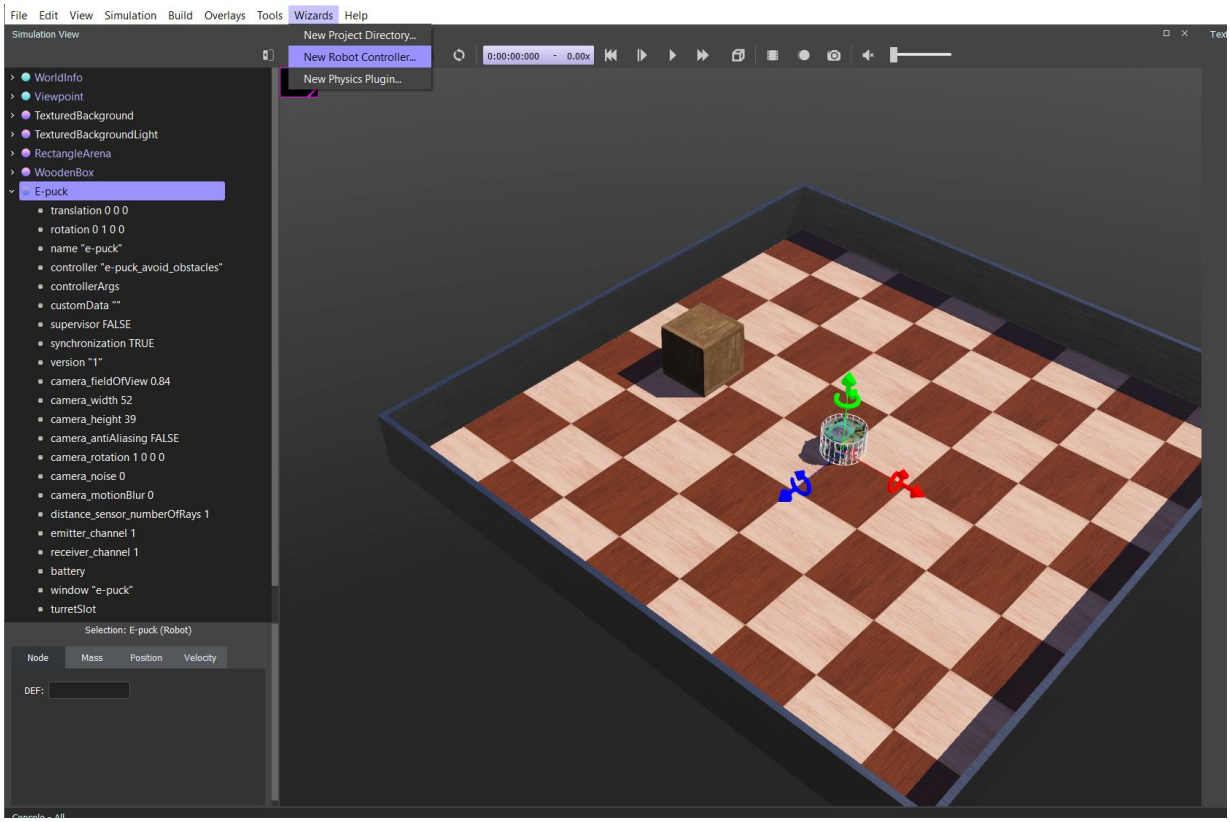


Рисунок 6. Создание нового контроллера.

Откроется диалоговое окно, нажмите «Next», чтобы продолжить. Следующая страница требует выбора языка программирования – выбираем Python и нажимаем далее (Next). В следующем окне (рисунок 7) – называем контроллер так: «Lab_5_controller_*Ваша фамилия латиницей*» и переходим в следующее окно.

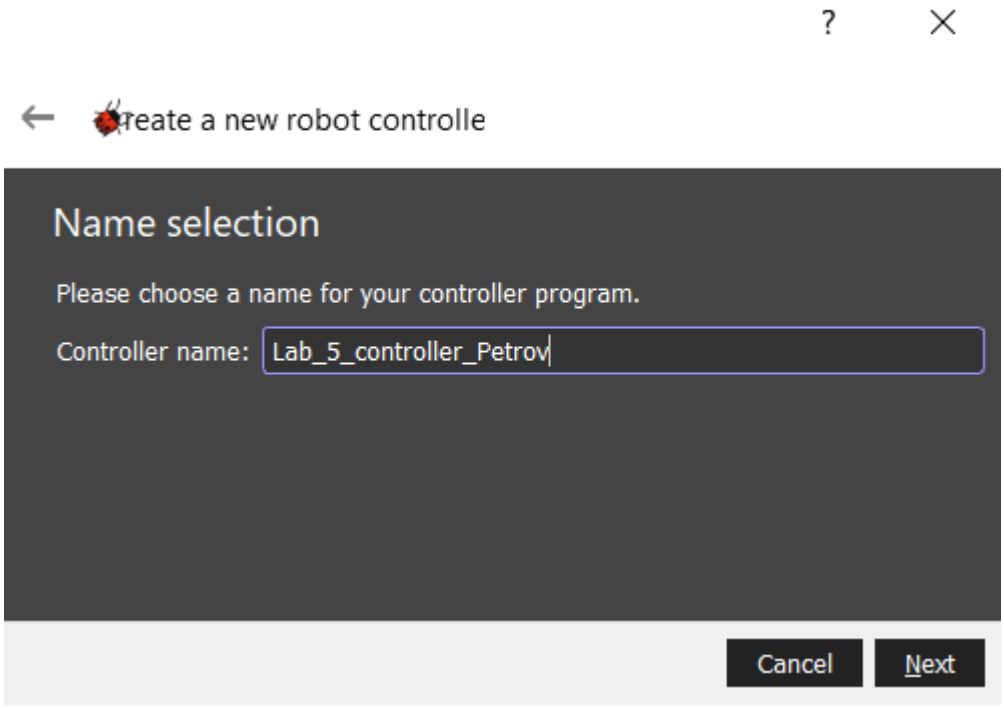


Рисунок 7. Наименование нового контроллера при создании.

Затем обязательно ставим галочку около «Open ‘Lab_work_5_controller_*Ваша фамилия латиницей*.py’ in Text Editor», чтобы открыть текст контроллера в редакторе и нажимаем «Finish».

Теперь необходимо подключить контроллер к роботу e-puck, чтобы мы смогли сразу тестировать разрабатываемые программы.

Для этого необходимо выбрать в дереве объектов робота, найти поле «Controller» и нажать кнопку «select…», как на рисунке 8.

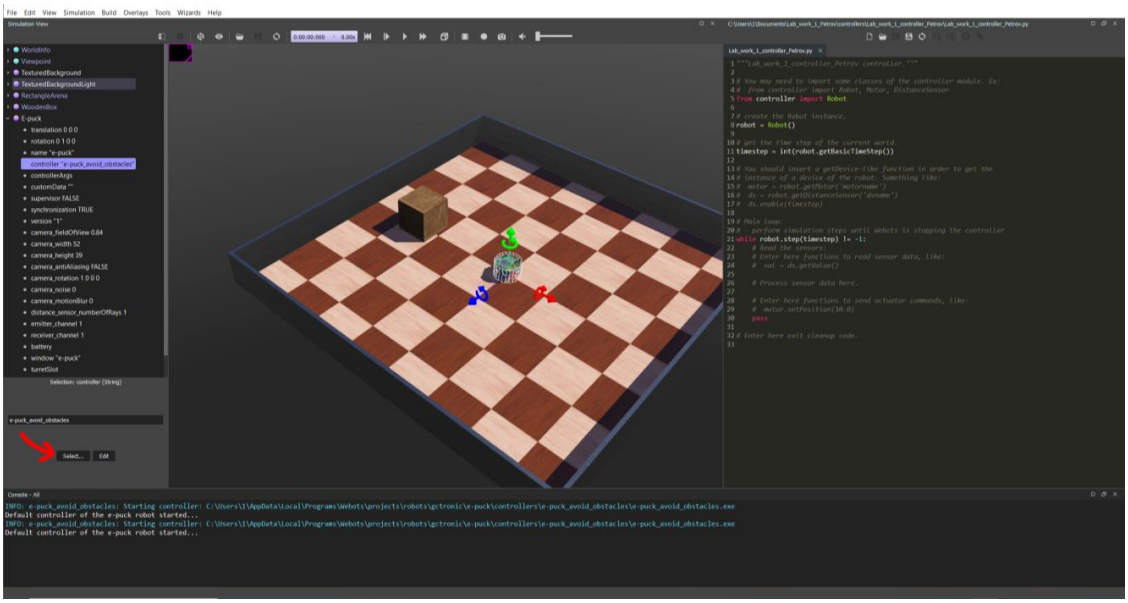


Рисунок 8. Подключение нового контроллера.

В диалоговом окне выбираем только что созданный контроллер с именем «Lab_5_controller_*Ваша фамилия латиницей*» и нажимаем «ОК».

Конфигурация мира симуляции в Webots

Для решения данной задачи нам будет необходимо добавить в мир симуляции объект – траекторию, по которой должен передвигаться робот. В Webots возможно поменять текстуры пола, однако проще создать 3D-объект очень малой толщины и импортировать его в мир симуляции. Именно так были созданы траектории для этого занятия.

Чтобы добавить 3D объект выберите в панели инструментов во вкладке «File» «Import 3D Model…», как на рисунке 9. Важно отметить, что 3D модель должна принадлежать к одному из перечисленных на рисунке 10 форматов.

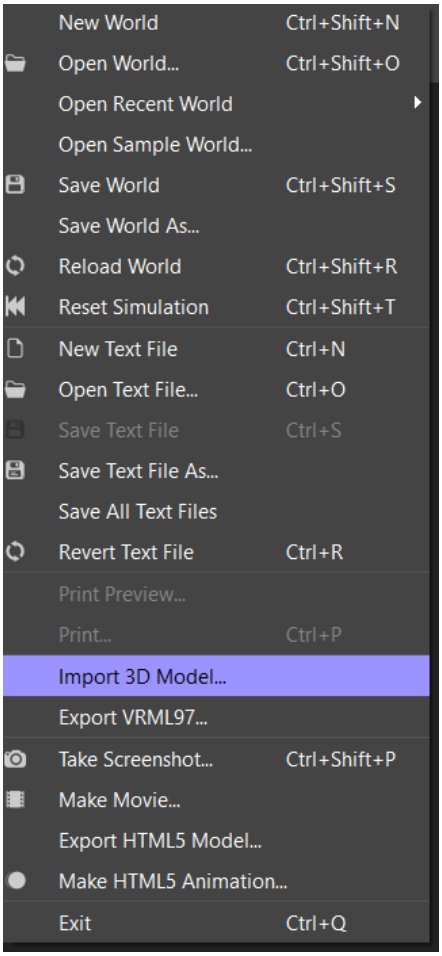


Рисунок 9. Импорт 3D-модели.

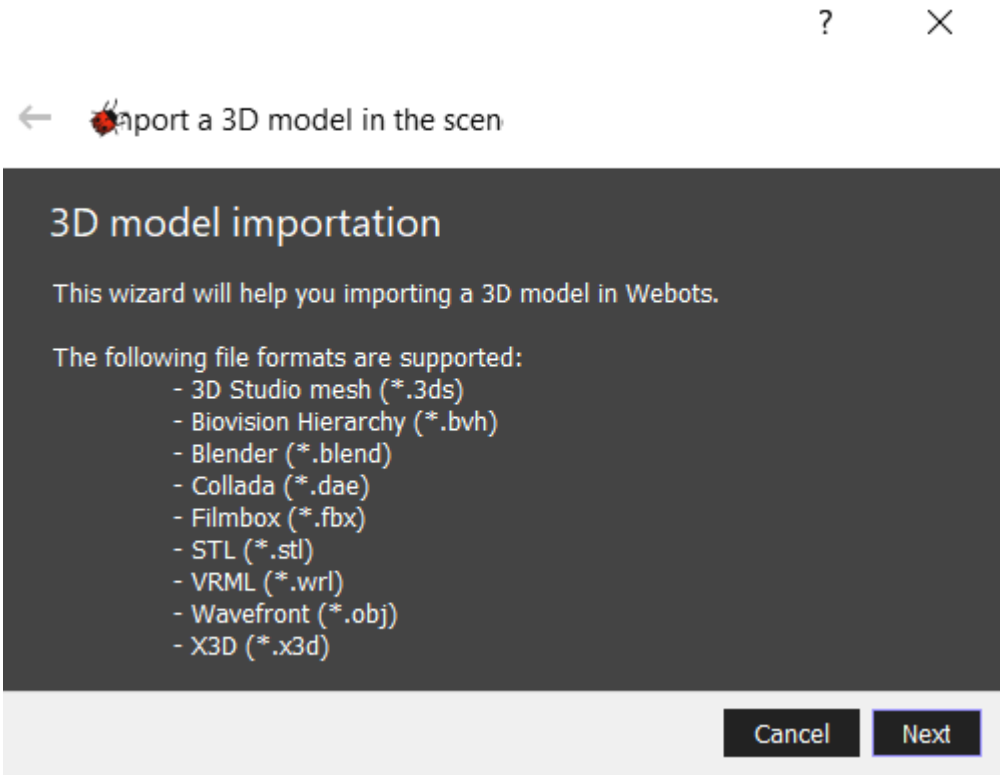


Рисунок 10. Поддерживаемые Webots форматы 3D-САПРов.

Далее, нажмите кнопку «Choose», чтобы выбрать файл для импорта. Выберете файл формата .obj в соответствии с Вашим вариантом. Внимание! Варианты моделей выданы отдельным файлом. Их распределение уточняется на лабораторной работе у преподавателя. Задания лежат в папке «Трассы». В случае примера (рисунок 11) – название трассы «Тестовый вариант».

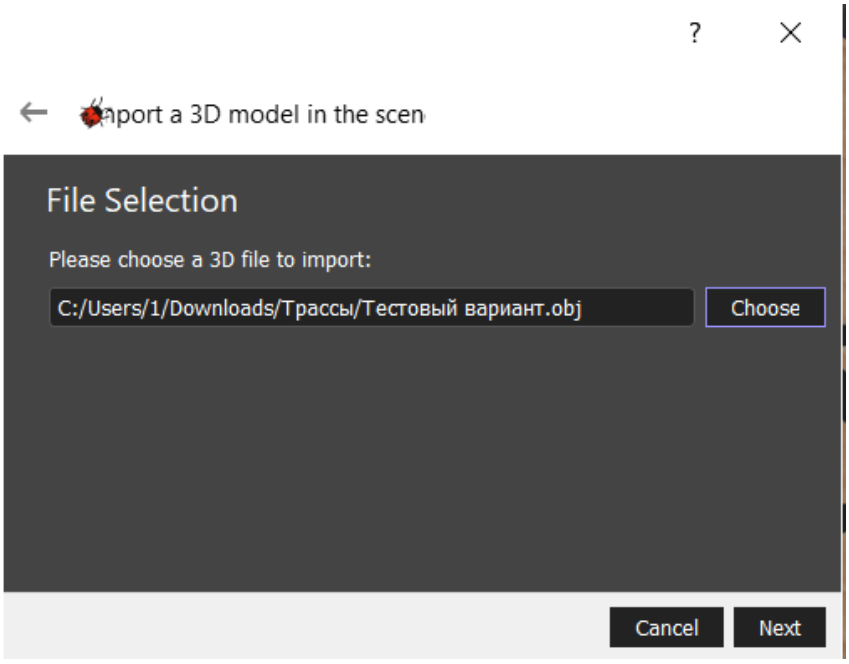


Рисунок 11. Импорт 3D-модели.

Затем убедитесь, что все галочки проставлены, как на рисунке 12, и в следующем окне нажимайте «Finish».

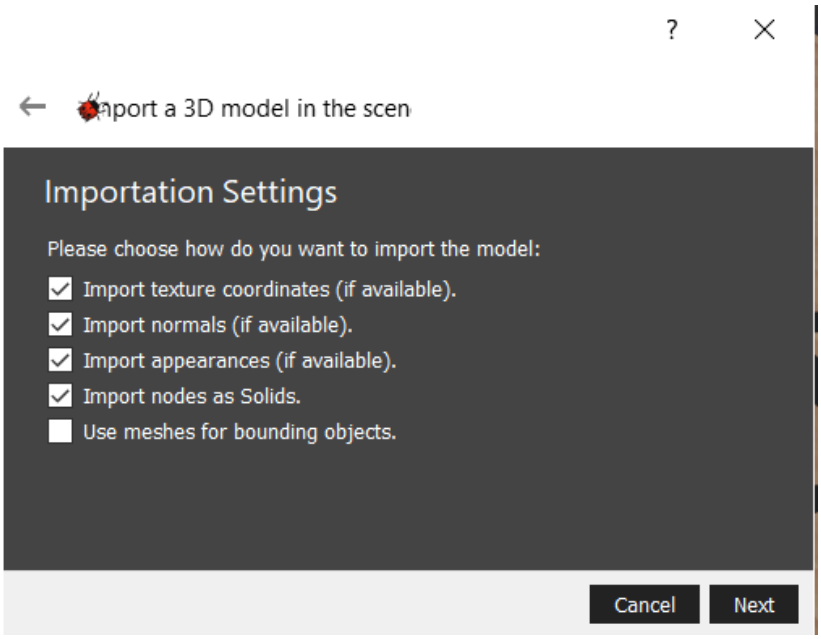


Рисунок 12. Параметры импорта.

После добавления у Вас появится в дереве объектов в левой части экрана новый объект «Solid». Настройте его параметр «scale» так: 0.0005, 0.0005, 0.0005 и переместите объект примерно на середину Вашей арены, как на рисунке 13.

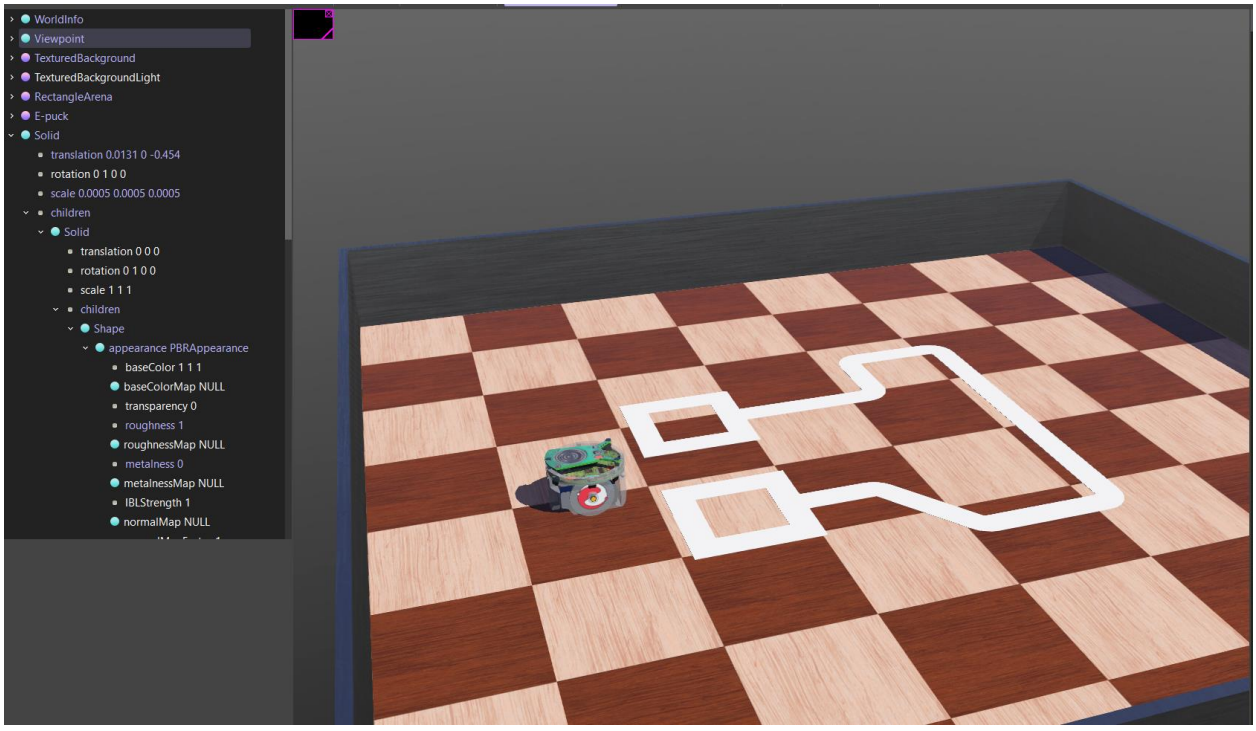


Рисунок 13. Размещение траектории.

Теперь нам нужно настроить цвет добавленного объекта, чтобы он контрастировал с окружающим полом – сделаем его черным. Для этого следует последовательно открыть вкладки в дереве объектов:

Solid -> Solid -> children -> Shape -> appearance PBRAppearance

И настроить параметр «baseColor» так: 0, 0, 0. Должно получиться как на рисунке 14.

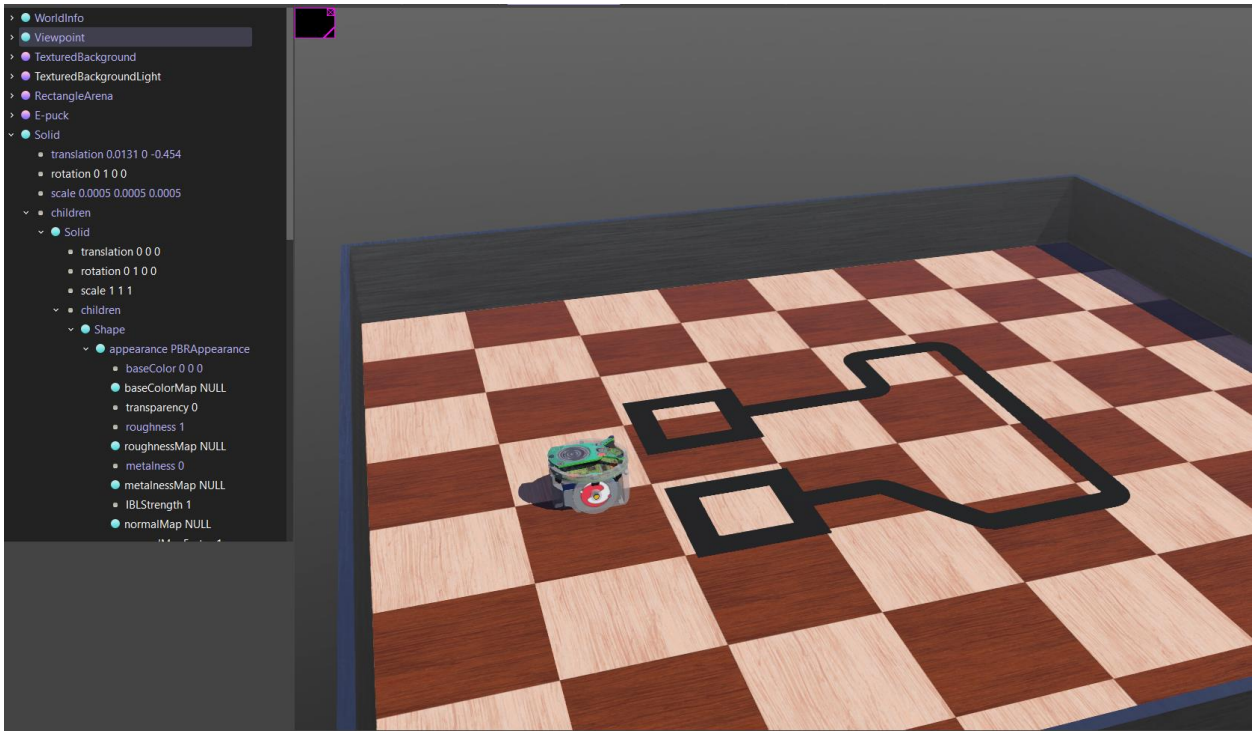


Рисунок 14. Настройка цвета траектории.

Теперь сделаем само поле более светлым для наибольшей контрастности. Для этого откроем теперь вкладку «RectangleArena» и в ней настроим «floorAppearance Parquety» так, чтобы значение поля «type» стало «light strip», как на рисунке 15.

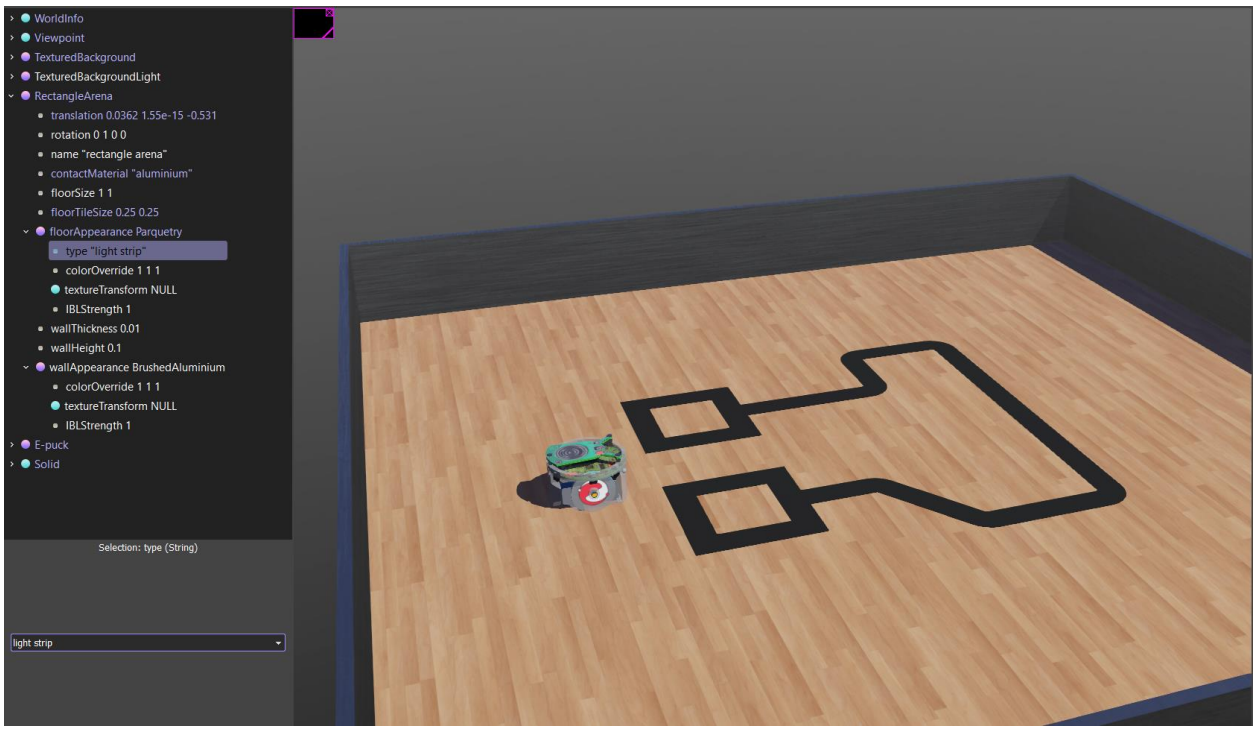


Рисунок 15. Контрастный полигон с траекторией движения.

Последним, но самым важным будет добавить в слот «groundSensorSlot» устройство «E-puckGroundSensor». Для этого выберете в дереве объектов робот e-puck и в соответствующем слоте выберете нужное устройство, как показано на рисунке 16, и нажмите «Add», чтобы добавить.

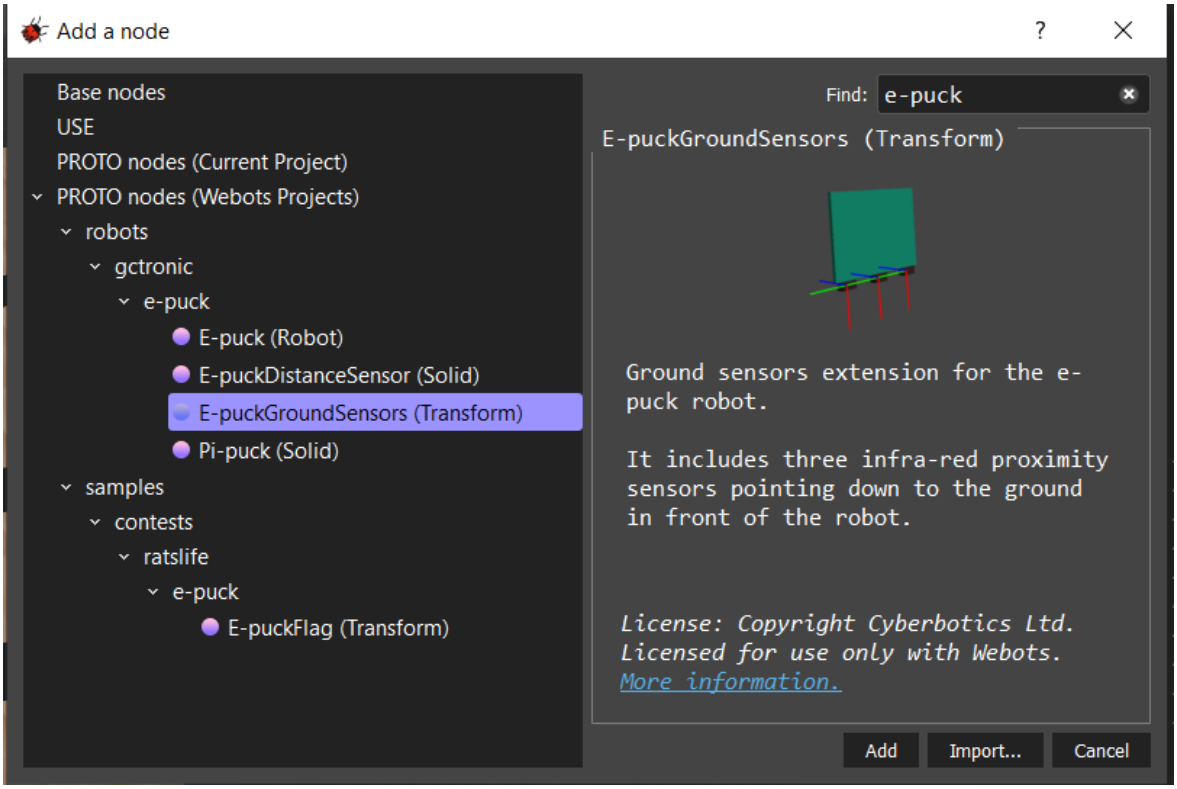


Рисунок 16. Добавление датчика линии для робота e-puck.

Перемещать объекты можно с помощью мыши, предварительно зажав клавишу Shift, а также, с помощью стрелок по осям, исходящим из выбранного объекта. Копировать и вставлять объекты – с помощью сочетания клавиш (Ctrl+C, Ctrl+V).

Переместите робота на точку «старт» (очерченный квадрат) так, чтобы появившаяся в передней части робота панель с датчиками линии была над черной линией в точности, как показано на рисунке 17.

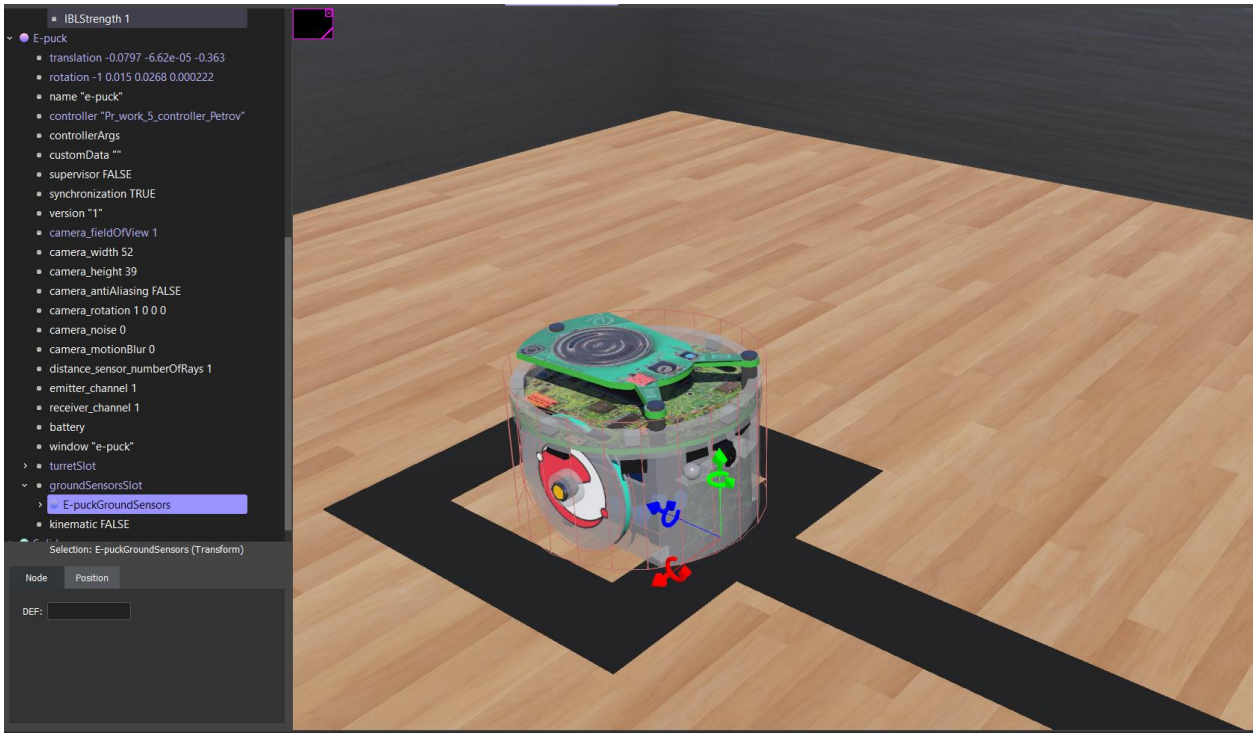


Рисунок 17. Размещение робота на стартовую позицию.

Нажмите сочетание клавиш (Ctrl+Shift+S), или нажмите на иконку, графически схожую с дискетой, в левой верхней части экрана для сохранения созданного мира.

На примере данной практической работы разработаем программу, реализующую алгоритм движения мобильной платформы по контрастной линии для робота e-puck на языке Python.

Разработка алгоритма движения подвижной платформы по контрастной линии для робота e-puck на языке Python

Алгоритм будет заключаться в том, чтобы, запустив робота, тот должен двигаться вдоль контрастной линии по направлению от «старта» к «финишу»

(сделаны особым образом) на определенной скорости, огибая все повороты и останавливаясь только в точке финиша.

Теперь рассмотрим алгоритм программы (рисунок 19), позволяющей передвигаться мобильной платформе вдоль контрастной линии.

Результат работы представлен на рисунке 18.

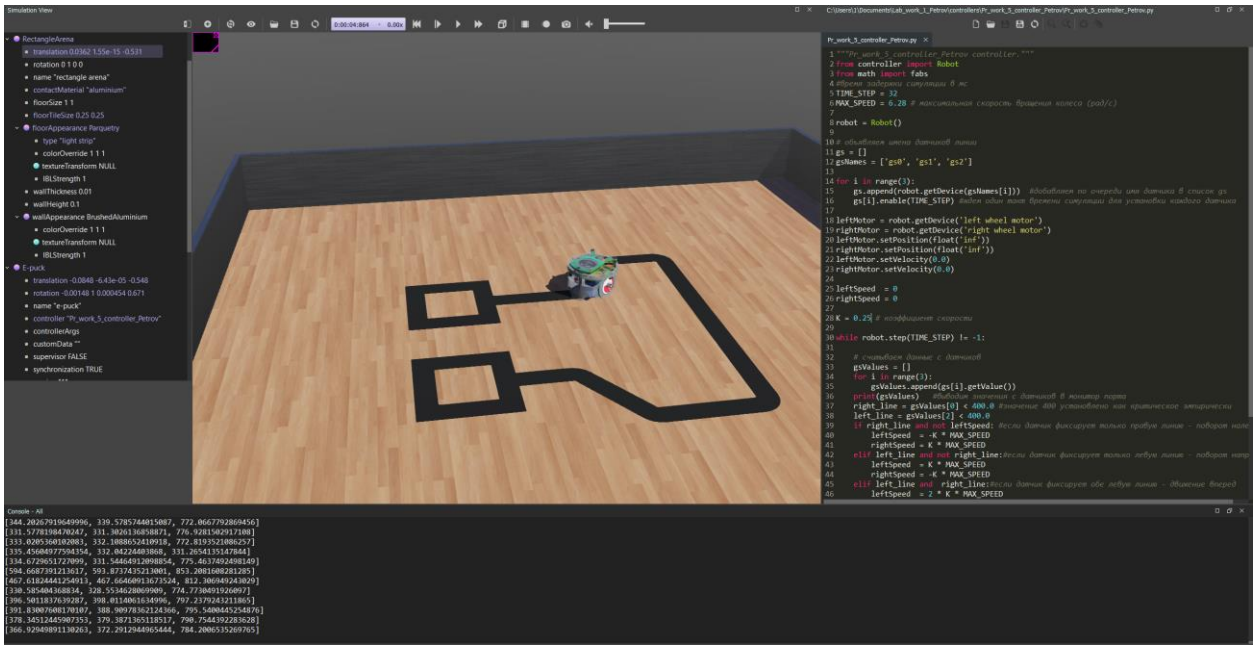


Рисунок 18. Результат работы программы.

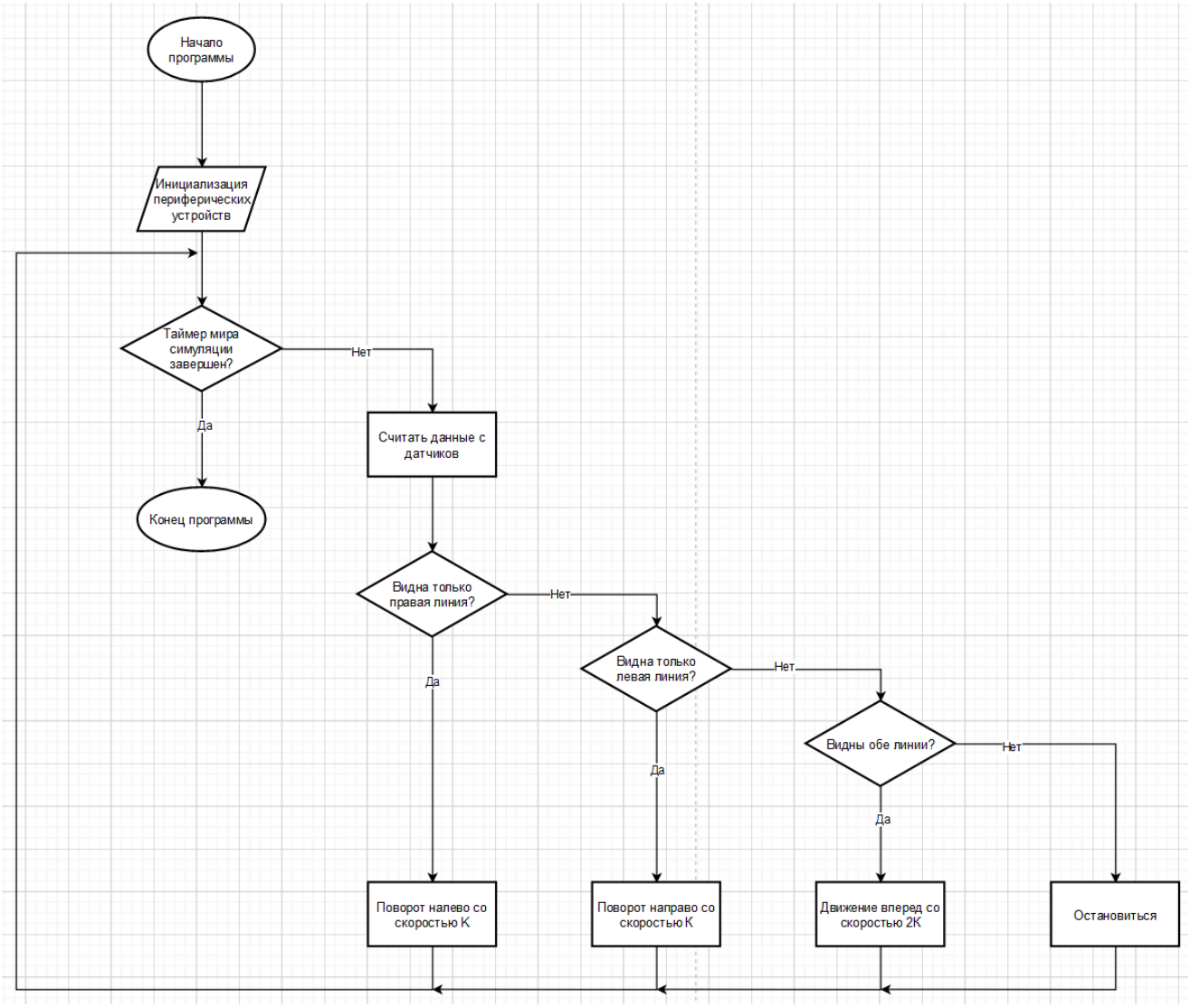


Рисунок 19. Алгоритм программы для движения подвижной платформы по контрастной линии.

И теперь взглянем на код-листинг полученной реализации ниже.

```
"""Lab_5_controller_Petrov controller."""
from controller import Robot
from math import fabs
# Создаём объект класса Robot
robot = Robot()

MAX_SPEED = 6.28 # максимальная скорость вращения колеса (рад/с)

# Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ
TIME_STEP = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит тип данных в целочисленный

gs = []  # создаем список для хранения имен датчиков освещенности
# объявляем имена датчиков приближения (заданы в документации к роботу)
gsNames = ['gs0', 'gs1', 'gs2']

```

```
for i in range(3):
    gs.append(robot.getDevice(gsNames[i]))  #добавляем по очереди имя датчика в список gs
    gs[i].enable(TIME_STEP) #ждем один такт времени симуляции для установки каждого датчика

leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

leftMotor.setPosition(float('inf')) # Разрешает поворачиваться колесу на полный оборот
rightMotor.setPosition(float('inf')) # аналогично для другого мотора
leftMotor.setVelocity(0.0) #устанавливает скорость вращения колеса(рад/с)
rightMotor.setVelocity(0.0) # аналогично для другого мотора

leftSpeed  = 0
rightSpeed = 0

K = 0.25 # коэффициент скорости

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

    # считываем данные с датчиков
    gsValues = []
    for i in range(3):
        gsValues.append(gs[i].getValue())
    print(gsValues)   #выводим значения с датчиков в монитор порта
    right_line = gsValues[0] < 400.0 #значение 400 установлено как критическое эмпирически
    left_line = gsValues[2] < 400.0
    if right_line and not leftSpeed:  # если датчик фиксирует только правую линию - поворот налево
        leftSpeed  = -K * MAX_SPEED
        rightSpeed = K * MAX_SPEED
    elif left_line and not right_line:# если датчик фиксирует только левую линию - поворот направо
        leftSpeed  = K * MAX_SPEED
        rightSpeed = -K * MAX_SPEED
    elif left_line and  right_line:   # если датчик фиксирует обе линии - движение вперед
        leftSpeed  = 2 * K * MAX_SPEED if 2 * K <= 1 else MAX_SPEED
        rightSpeed = 2 * K * MAX_SPEED   if 2 * K <= 1 else MAX_SPEED
    else:                             # если обе линии не видны датчику - остановиться
        leftSpeed  = 0
        rightSpeed = 0
    # устанавливаем скорость
    leftMotor.setVelocity(leftSpeed)
    rightMotor.setVelocity(rightSpeed)
```

Порядок выполнения работы

1. Запустить Webots.

2. Ознакомиться с реализацией алгоритма движения мобильной платформы по контрастной линии, приведенной в разделе «Краткие теоретические сведения».
3. Подготовить мир симуляции в соответствии с заданием.
4. Реализовать алгоритм движения робота e-Puck на языке Python в соответствии с заданием.

Задание

Необходимо создать и настроить мир симуляции и программу-контроллер, в соответствии с разделом «Краткие теоретические сведения» и вариантом трассы, добавляемой в мир симуляции. Внимание! Варианты моделей выданы отдельным файлом, имеющим расширение «.stl». Их распределение уточняется на лабораторной работе у преподавателя. Трассы находятся в папке «Трассы», прилагаемой вместе с текстом лабораторной работы.

Требуется написать программу и оптимизировать алгоритм движения мобильной платформы по контрастной линии для робота e-puck на языке Python.

Содержание отчета по работе

Отчет должен содержать выполненные следующие пункты:

1. Составленная блок-схема программы.
2. Настроенный мир симуляции для выполнения задания.
3. Код-листинг программы по заданию.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Webots.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора.
3. Изложить материал раздела «Краткие теоретические сведения» и проделать практические предписания.
4. Ответить на вопросы.
5. Озвучить задание лабораторной работы.

6. После завершения работы проверить успешность выполнения учащимися самостоятельного задания.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учащимся.
2. Проверить работу моделирования в симуляторе – в случае ошибок, исправить программу вместе с учащимся.

Лабораторная работа №6 «Навигация мобильных роботизированных платформ с помощью цифрового компаса»

Цель работы

1. Настроить мир и контроллер для практической работы;
2. ознакомиться с теоретическими сведениями о принципах работы цифрового компаса и его применении в робототехнике;
3. ознакомиться с реализацией алгоритма движения мобильной платформы, основанным на ориентировании по компасу для робота e-puck на языке Python;
4. подготовить блок-схему полученного алгоритма.

Краткие теоретические сведения

Симуляция работы МЭМС датчиков в Webots и теоретические сведения

В данной работе мы продолжаем изучать особенности применения инерциальных МЭМС для решения задач навигации в мобильной робототехнике.

В связи с широким распространением мобильной платформ и GPS чипов в составе конечных электронных средств, идея применения цифрового компаса для навигации является актуальной как в мобильной робототехнике, так и в применении мобильных портативных средств во встраиваемых системах.

Итак, рассматриваемый нами датчик представляет собой цифровой компас, принцип действия которого основан на взаимодействии магниточувствительных элементов с горизонтальной составляющей магнитного поля Земли. На практике, чаще всего, цифровой компас имеет в своем составе блок магниторезисторов или элементов Холла. Элементы Холла – полупроводниковые приборы, чувствительные к изменению магнитного поля, принцип работы которого основывается на распределении зарядов на кремниевой пластине под влиянием магнитного поля Земли. Приборы на магниторезисторах и элементах Холла представляют собой цифровой компас в его классическом виде, как автономный измерительный инструмент, в отличие от систем «собирательного» типа входная информация для которых поступает опосредованно, в виде сигнала со спутника (GPS - навигация).

Магнитометр — (от гр. μαγνητό — магнит + гр. μετρεω измеряю) - прибор для измерения характеристик магнитного поля и магнитных свойств материалов. Принцип работы основан на высокоточной технологии эффекта Холла. Типичный магнитометр включает в себя магнитные сенсоры, определяющие напряжённость магнитного поля земли по осям, схему управления, цепь усиления сигнала и вычислительную схему для обработки сигналов с каждого датчика.

При помещении в магнитное поле пластины-проводника или полупроводника под 90 ° к направлению силовых линий магнитного потока произойдет перемещение электронов по поперечине пластины под действием силы Лоренца. Их направление зависит от того, в какую сторону идет сила тока и силовые линии магнитного потока. Иначе говоря, (ЭХ) эффект Холла – это частный случай действия силы Лоренца, то есть действия магнитного поля на заряженную частицу. Схематически эффект показан на рисунке 1.

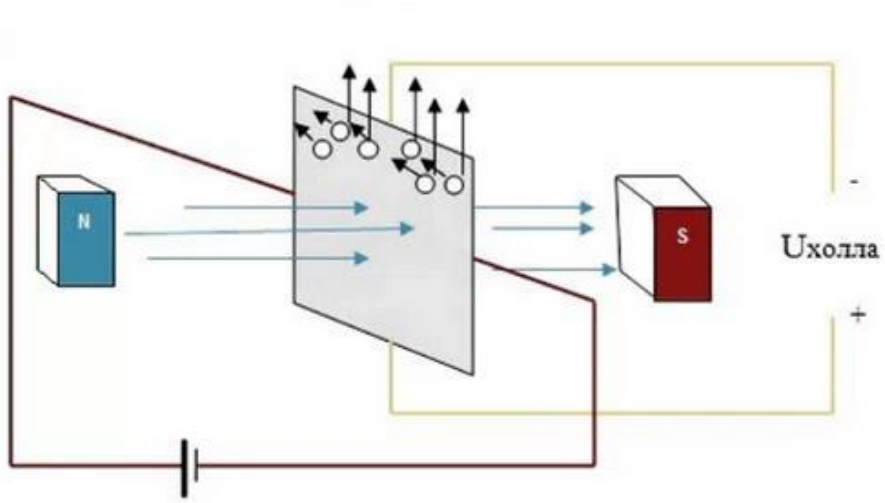


Рисунок 1. Эффект Холла.

На практике эти датчики редко используются сами по себе для решения определенных технических задач, таких как инерционная навигация, активно применяемая в средствах противовоздушной обороны и авиации, регистрация инерционных эффектов как косвенных измеряемых факторов событий (например, крушение воздухоплавательного или наземного агрегата) и многие другие.

На практике мы имеем дело, чаще всего, с определением местоположения и направления посредством навигационных систем внешнего типа, примером тому - Android с его приложением Google Maps.

Принцип работы алгоритма именно такого случая использования:

1. По сигналам со спутников снимаем показания координат приёмника системы спутниковой навигации (и, соответственно, объекта)
2. Засекаем момент времени, в который было сделано определение координат.
3. Выжидаем некоторый интервал времени, достаточно короткий для более качественных результатов.
4. Повторно определяется местоположение объекта.
5. Решается простейшая навигационная задача вычисления вектора скорости движения из полученных координат двух точек и размера временного интервала, после чего, зная вектор, мы с легкостью получаем:
    1. направление движения
    2. скорость движения

В текущей лабораторной работе мы подробнее познакомимся с навигацией on-board, реализуемой с помощью цифрового компаса.

Робот e-puck обладает слотом для размещения разных датчиков, помимо уже строенных, таких как: инфракрасные датчики приближения, датчики света и светодиоды для индикации работоспособности датчиков. Расположение штатных датчиков и их ориентация показаны на рисунке 2.

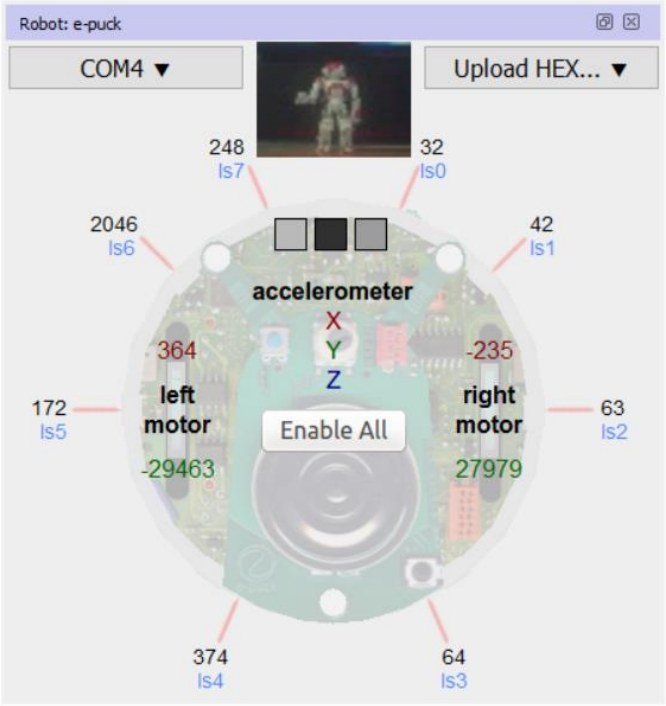


Рисунок 2. Расположение датчиков и сенсоров на роботе e-puck.

Однако в упомянутый слот, именуемый «groundSensorSlot», есть возможность установки других датчиков (и не только), которые перечислены на рисунке 3.

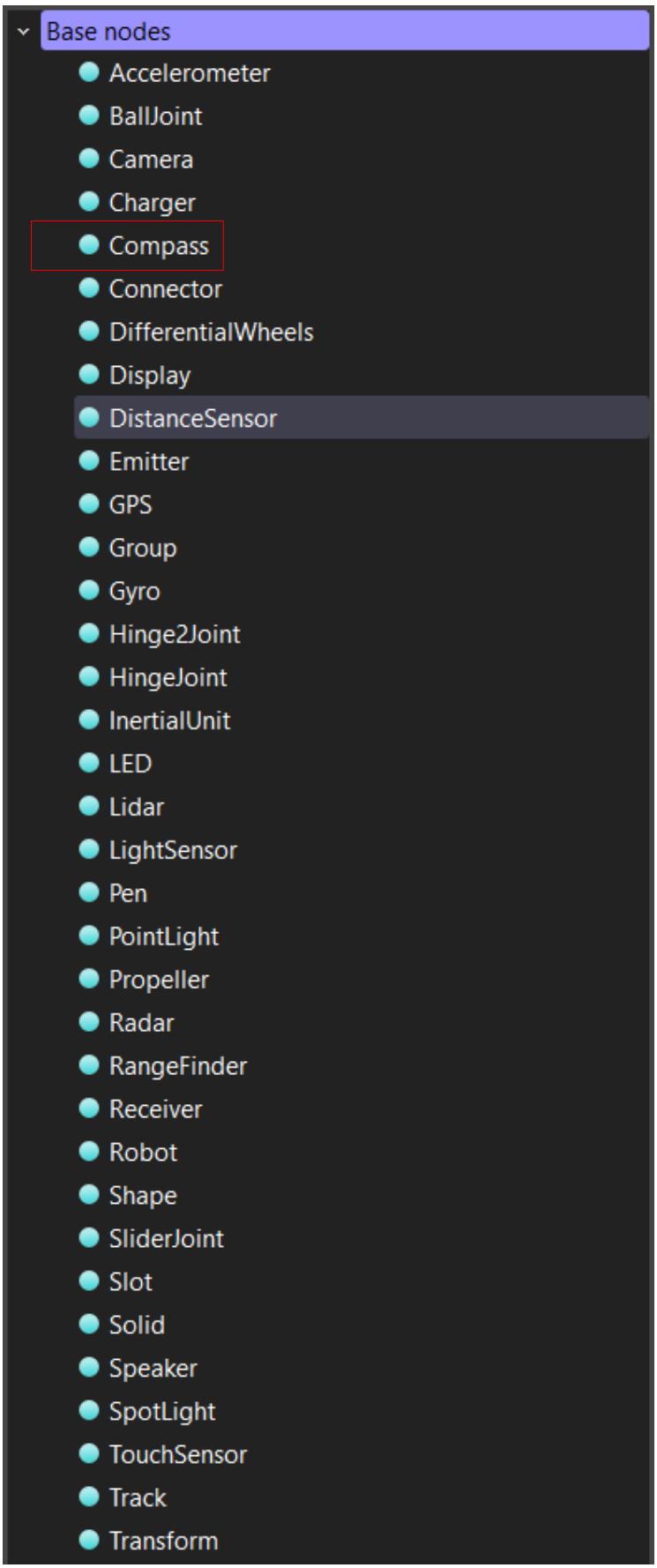


Рисунок 3. Возможные устройства для размещения в слоте «groundSensorSlot», на роботе e-puck.

Среди них можно обнаружить датчик под названием «Compass» - цифровой компас.

Для того, чтобы добавить цифровой компас в состав периферии робота e-puck, выберете в дереве объектов робот e-puck и в соответствующем слоте «groundSensorSlot» выберете нужное устройство, как показано на рисунке 4, и нажмите «Add», чтобы добавить.

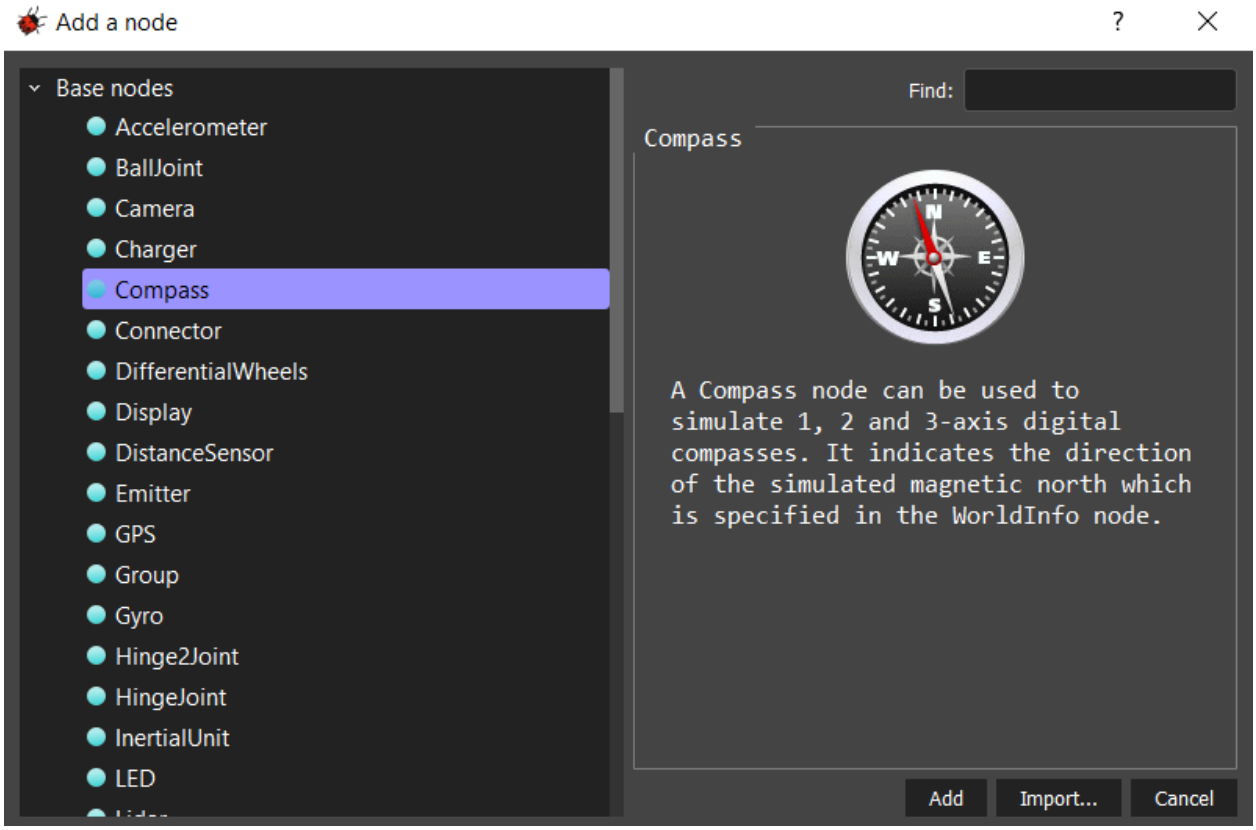


Рисунок 4. Добавление цифрового компаса для робота e-puck.

Для ознакомления с периферическим функционалом робота e-puck, можно обратить внимание на рисунок 5.

Device	Name
Motors	'left wheel motor' and 'right wheel motor'
Position sensors	'left wheel sensor' and 'right wheel sensor'
Proximity sensors	'ps0' to 'ps7'
Light sensors	'ls0' to 'ls7'
LEDs	'led0' to 'led7' (e-puck ring), 'led8' (body) and 'led9' (front)
Camera	'camera'
Accelerometer	'accelerometer'
Gyro	'gyro'
Ground sensors (extension)	'gs0', 'gs1' and 'gs2'
Speaker	'speaker'

Рисунок 5. Таблица с названиями устройств на роботе.

Напоминаем, что весьма полезным является более подробное самостоятельное ознакомление с функциональными возможностями робота e-puck по ссылке.

Создание программы-контроллера на языке Python

Создадим новый проект также, как это было в лабораторной работе 1. При этом назовите проект «Lab_work_6_*Ваша фамилия латиницей*», а название мира будет «Lab_6_world.wbt». Добавьте робота e-puck в мир симуляции.

Чтобы добавить контроллер, необходимо выбрать в ленте вкладку «Wizards», и нажать на вкладку в выпадающем меню «New Robot Controller…», как на рисунке 6.

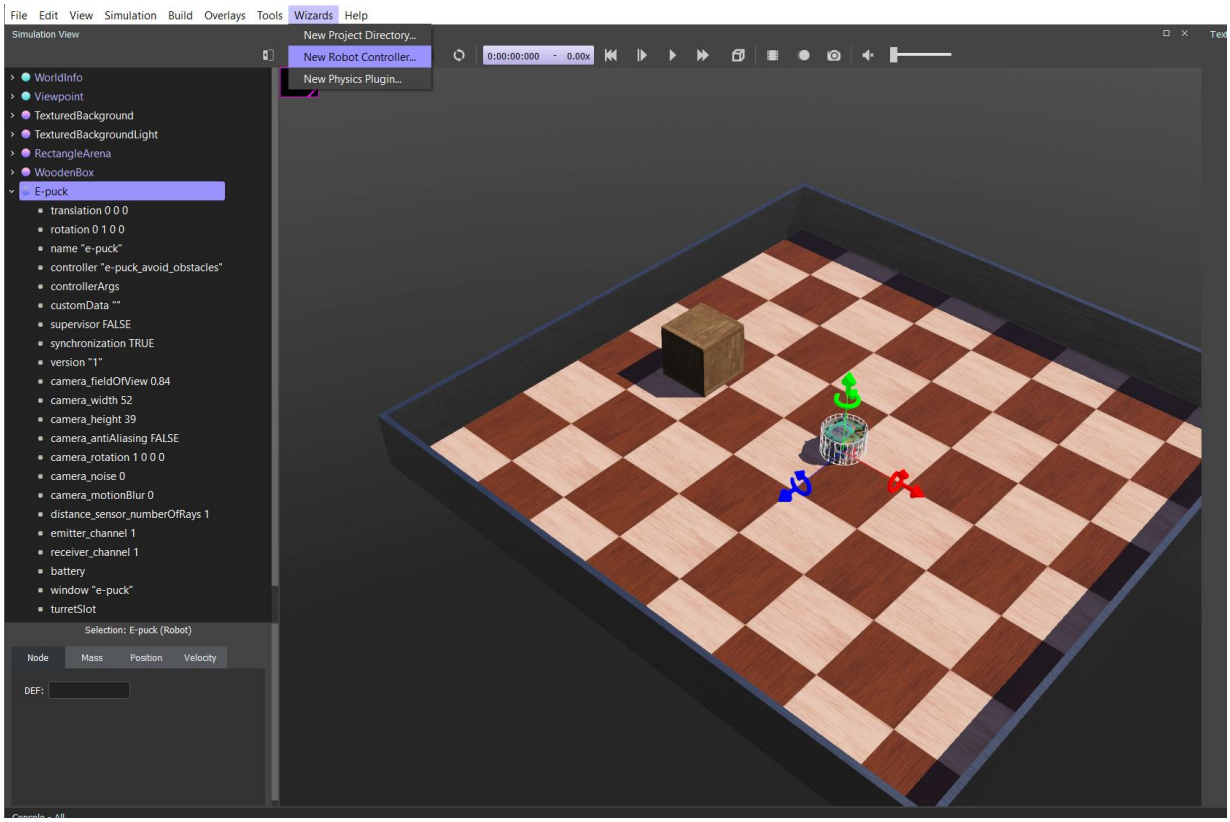


Рисунок 6. Создание нового контроллера.

Откроется диалоговое окно, нажмите «Next», чтобы продолжить. Следующая страница требует выбора языка программирования – выбираем Python и нажимаем далее (Next). В следующем окне (рисунок 7) – называем контроллер так: «Lab_6_controller_*Ваша фамилия латиницей*» и переходим в следующее окно.

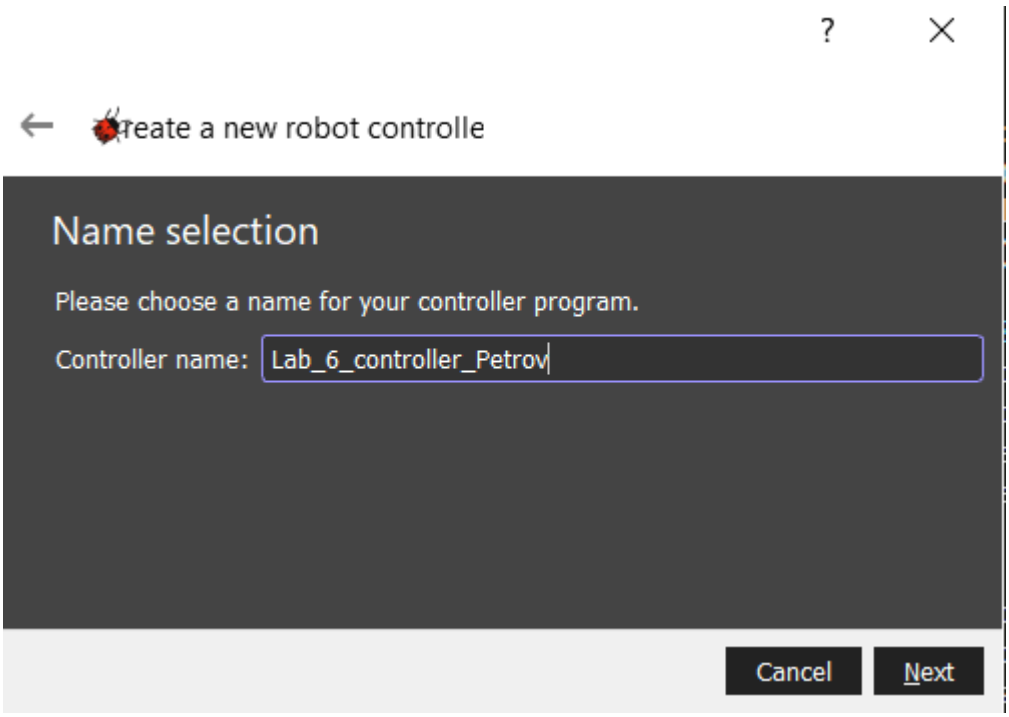


Рисунок 7. Наименование нового контроллера при создании.

Затем обязательно ставим галочку около «Open ‘Lab_6_controller_*Ваша фамилия латиницей*.py’ in Text Editor», чтобы открыть текст контроллера в редакторе, и нажимаем «Finish».

Теперь необходимо подключить контроллер к роботу e-puck, чтобы мы смогли сразу тестировать разрабатываемые программы.

Для этого необходимо выбрать в дереве объектов робота, найти поле «Controller» и нажать кнопку «select…», как на рисунке 8.

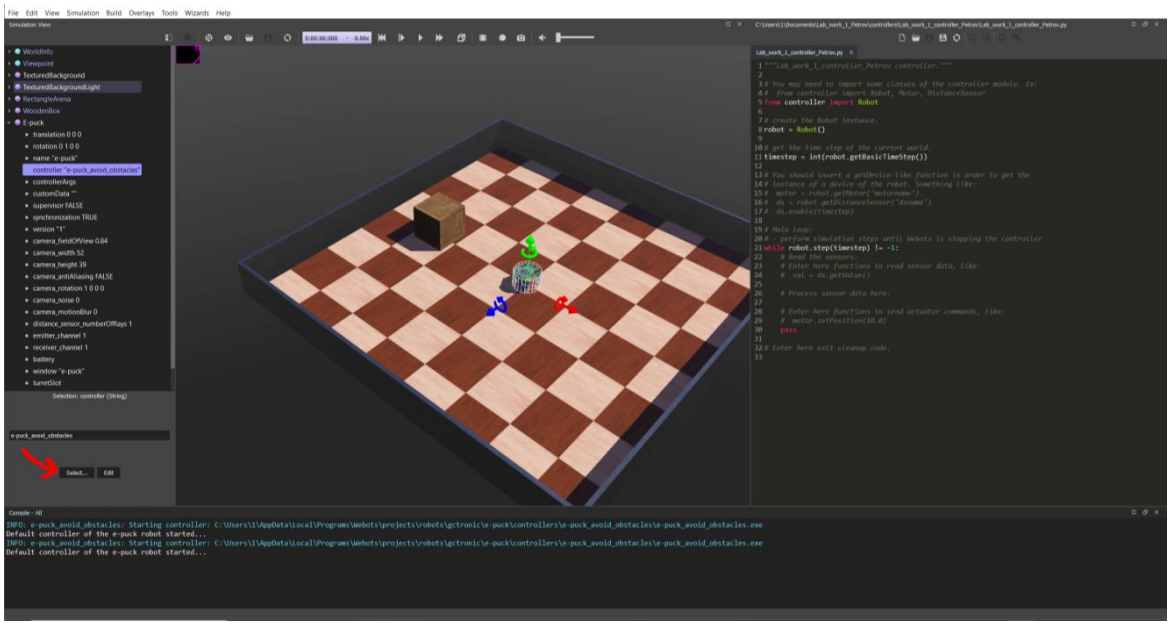


Рисунок 8. Подключение нового контроллера.

В диалоговом окне выбираем только что созданный контроллер с именем «Lab_6_controller_*Ваша фамилия латиницей*», как на рисунке 9, и нажимаем «ОК».

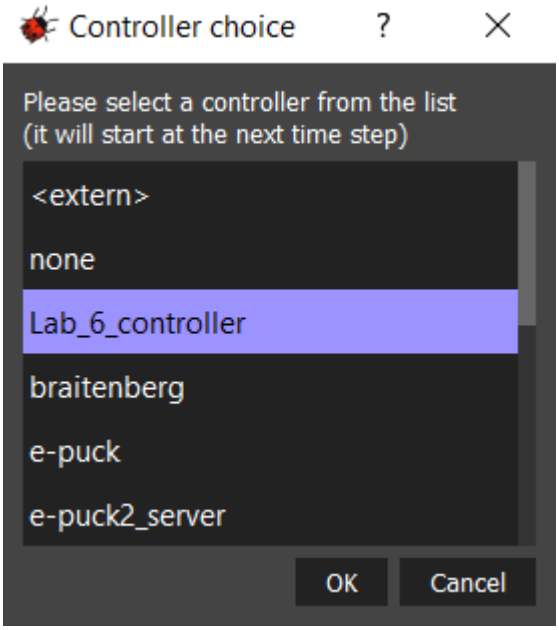


Рисунок 9. Выбор контроллера.

Конфигурация мира симуляции в Webots

После настройки размера арены, как показано в лабораторной работе 1, добавим объект «WoodenBox» и настроить его массу, сделав этот параметр равным единице, и габариты, как на рисунке 10.

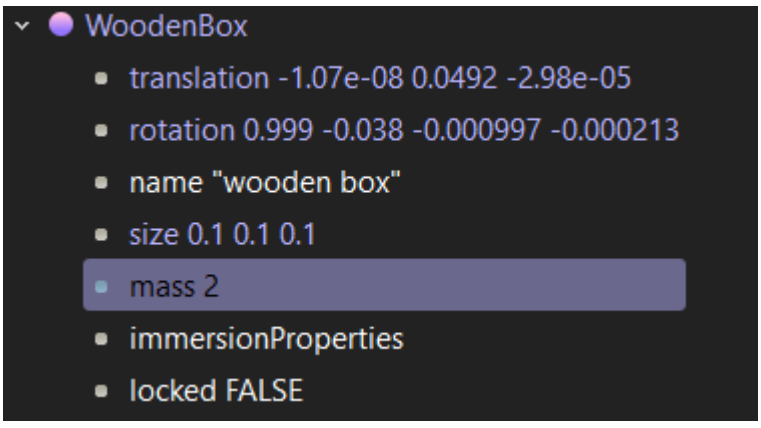


Рисунок 10. Настройка объекта «WoodenBox».

Затем, как и в лабораторной работе 2, нам будет необходимо размножить уже имеющийся объект «Box» (рисунок 11) так, чтобы создать некоторые

препятствия на арене, не позволяющие проехать роботу от одной стенки арены до противоположной не свернув, как на рисунке 12.

Варианты расположения ящиков зависят от Вашего варианта, выданного преподавателем. Сами задания сформулированы в конце текста лабораторной работы №6.

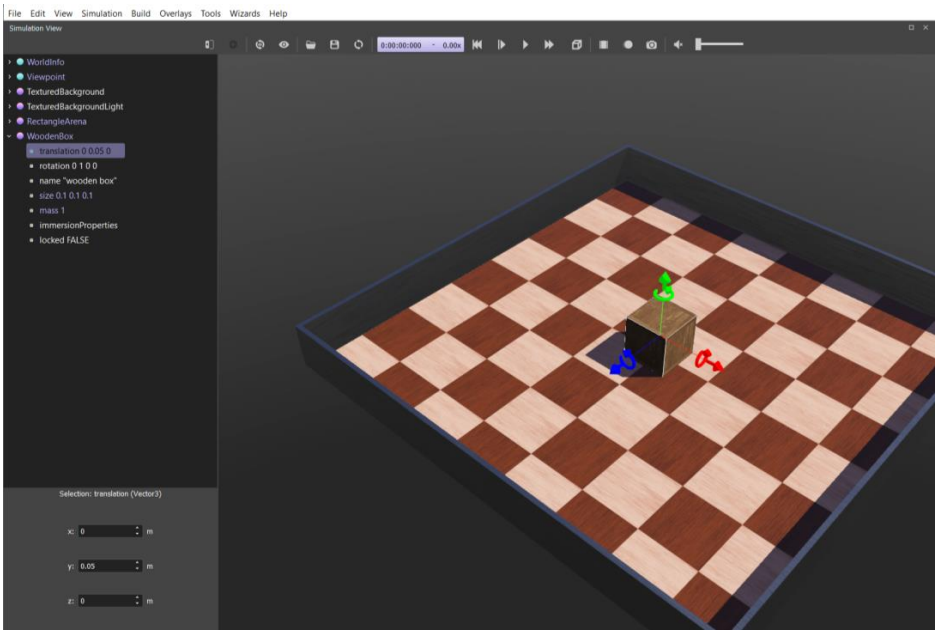


Рисунок 11. Настройка объекта «WoodenBox».

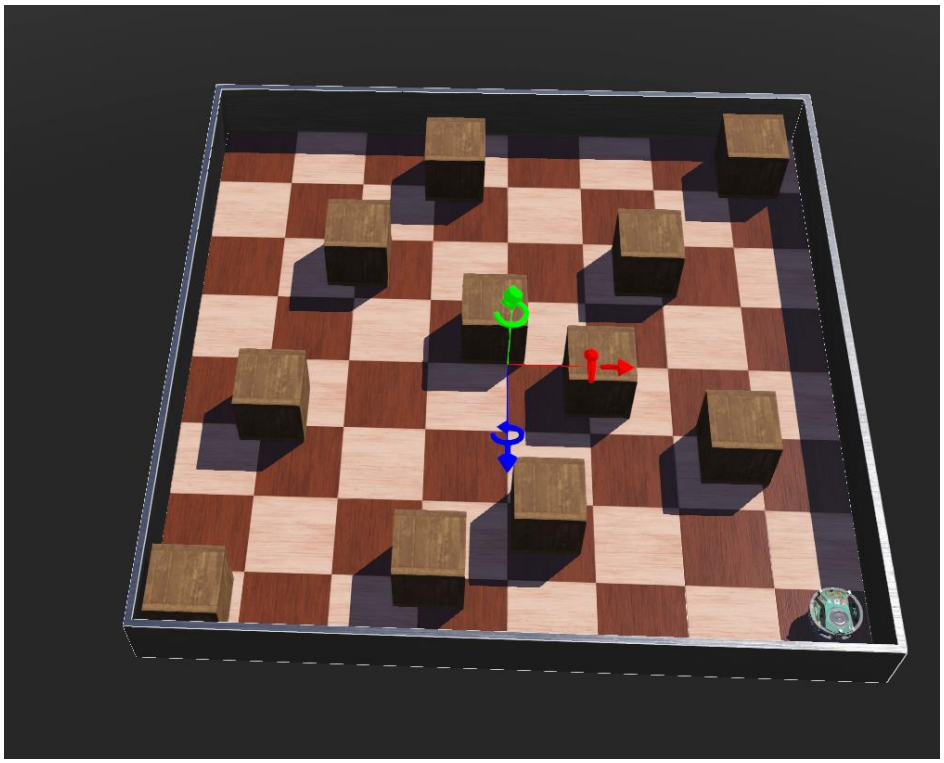


Рисунок 12. Пример созданной арены.

Перемещать объекты можно с помощью мыши, предварительно зажав клавишу Shift, а также, с помощью стрелок по осям, исходящим из выбранного объекта. Копировать и вставлять объекты – с помощью сочетания клавиш (Ctrl+C, Ctrl+V).

Нажмите сочетание клавиш (Ctrl+Shift+S), или нажмите на иконку, графически схожую с дискетой  , в левой верхней части экрана для сохранения созданного мира.

На примере текущей лабораторной работы напишем программу, реализующую алгоритм ориентации робота по направлениям сторон света для робота e-puck на языке Python.

Разработка алгоритма ориентации робота по направлениям сторон света для робота e-puck на языке Python

Алгоритм будет заключаться в том, чтобы в зависимости от текущей ориентации робота в пространстве определять наикратчайший угол поворота к направлению задаваемой стороны света.

Для начала уточним формальности наименований сторон света в мире симуляции. Нажмите на кнопку в панели сверху, имеющую цветографическое сходство с глазом, как на рисунке 13, и выберите в выпадающем меню «Top View», или нажмите сочетание клавиш Alt + 1.

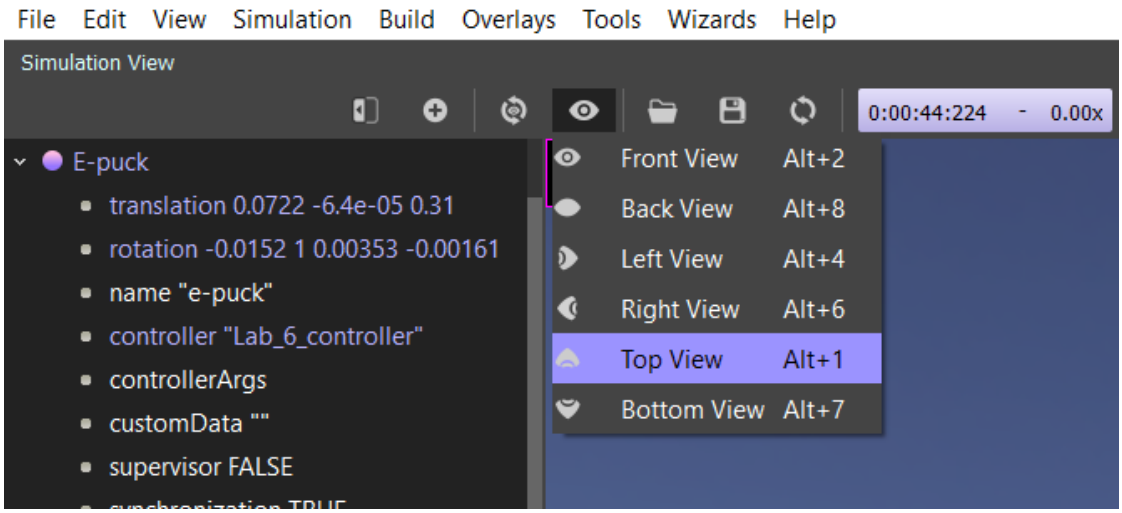


Рисунок 13. Ориентация обзора в мире симуляции.

Проделав так несколько раз, Вы убедитесь, что программа всегда ориентирует мир одинаково относительно Вашего угла обзора. При выборе «Top View» , Вы всегда будете видеть арену мира симуляции сверху, как показано на рисунке 14.

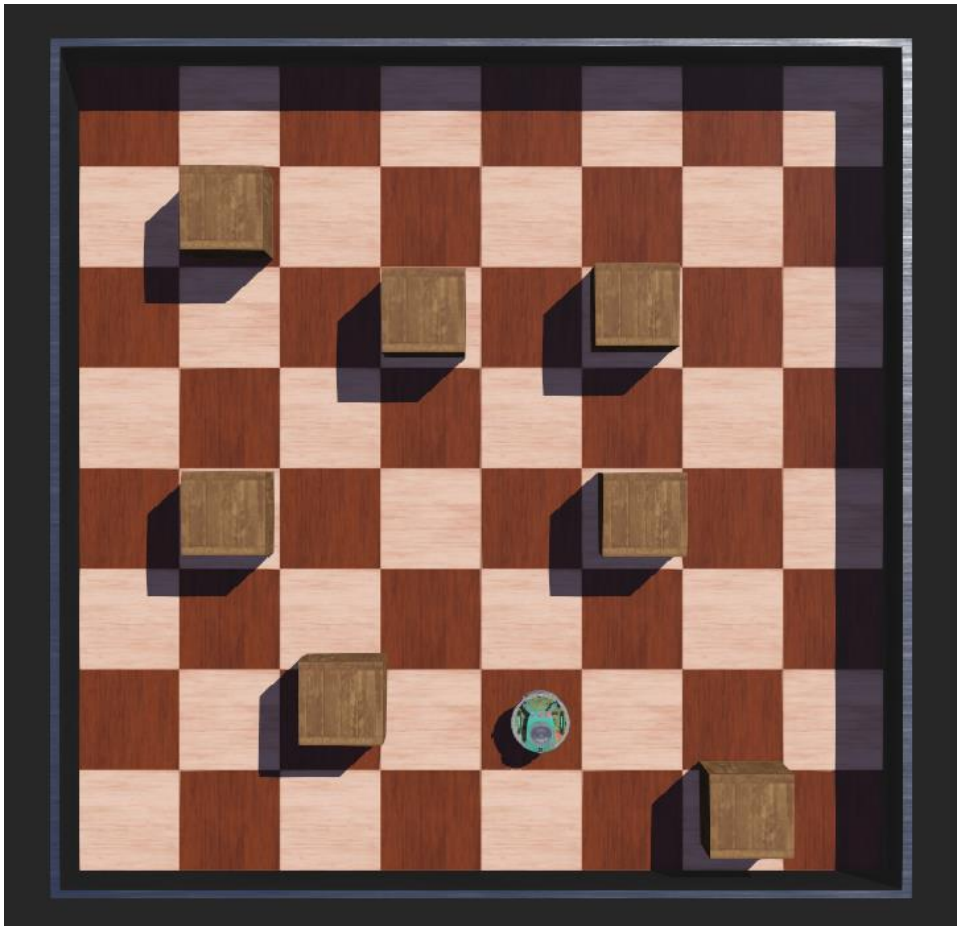


Рисунок 14. Примененная ориентация «Top View».

Определим, что направлением «Север» («North») будет ориентирован сверху при выборе ориентации угла обзора «Top View», согласно рисунку 14.

Компас в системе Webots возвращает три числа, вторая пара которых обозначает направление к Северу следующим образом (Рисунок 15):

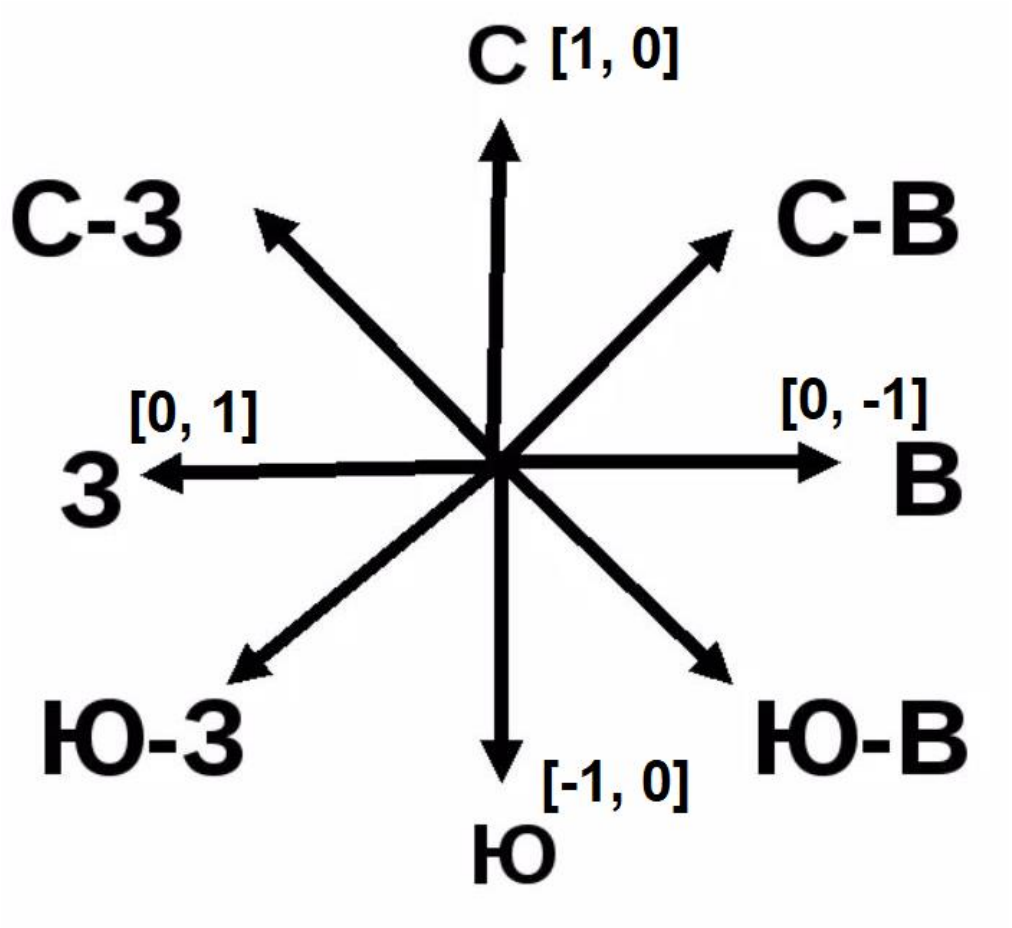


Рисунок 15. Возвращаемые данные цифрового компаса в Webots, характеризующие стороны света.

Результат работы представлен на рисунке 16.

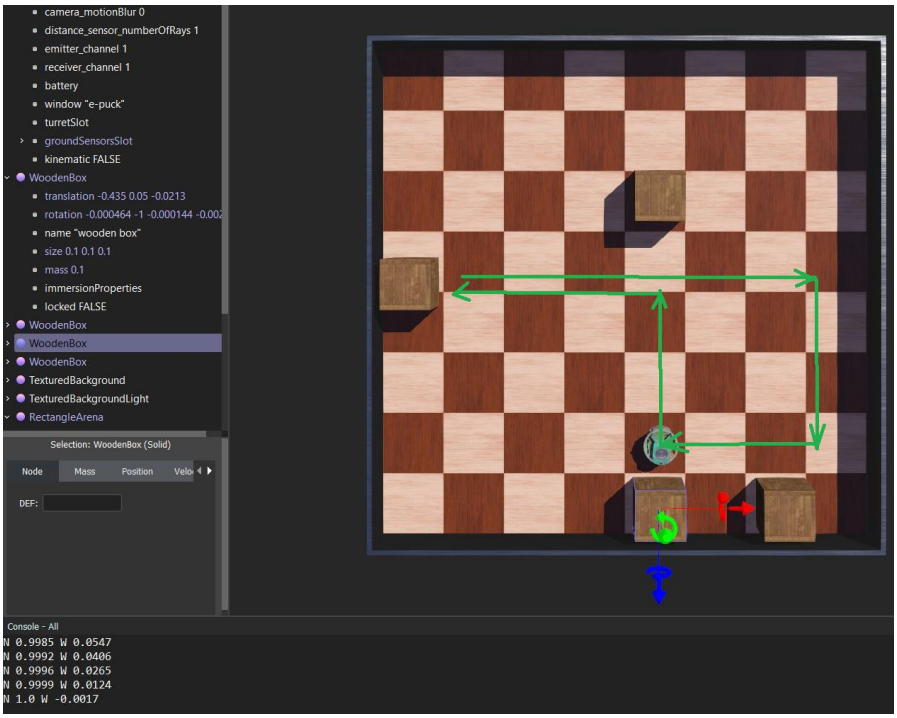


Рисунок 16. Результат работы программы.

В данном случае, реализован алгоритм передвижения по траектории, схематически отмеченной на рисунке 15. Более подробно – читать комментарии в коде-листинге программы.

И теперь взглянем на код-листинг полученной реализации ниже.

```
"""Lab_6_controller controller."""
from controller import Robot, Compass
# Создаём объект класса Robot
robot = Robot()

MAX_SPEED = 6.28 # максимальная скорость вращения колеса (рад/с)
WHEEL_RAD = 0.0205 # радиус колеса (м)
AXLE_LENGTH = 0.052 # длина оси конструкции робота (м)

# Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ
TIME_STEP = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит тип данных в целочисленный

# создаем объект компасса
comp = robot.getDevice('compass')
comp.enable(TIME_STEP)

def delay(delay_ms): #объявляем функцию, аргумент - переменная delay_ms - в мс.
    control_time = robot.getTime()
    while robot.step(TIME_STEP) != -1:         # пока программа не остановлена... Функция step() не может вернуть -1.
        if robot.getTime() - control_time > delay_ms / 904:
            break                          # если прошло времени больше отмеренного - выходим из цикла
                                    # 904 - константа, полученная из 1000 мс - 3 * 32 мс.
                                    # В теле функции задержки 3 раза вызывается getTime() и step(TIME_STEP)
leftMotor = robot.getDevice('left wheel motor') # создаем объект левого мотора
rightMotor = robot.getDevice('right wheel motor') # создаем объект правого мотора

leftMotor.setPosition(float('inf')) # Разрешает поворачиваться колесу на полный оборот
rightMotor.setPosition(float('inf')) # аналогично для другого мотора
leftMotor.setVelocity(0.0)    # устанавливает скорость вращения колеса 0 (рад/с)
rightMotor.setVelocity(0.0)   # аналогично для другого мотора

def stop():                # создадим для удобства функцию остановки робота
    leftMotor.setVelocity(0)
    rightMotor.setVelocity(0)

def move_straight(s, coef = 25):
```

```
    t = abs( s / ( coef * 0.01 * MAX_SPEED * WHEEL_RAD ) ) # расчет времени движения в секундах
    leftMotor.setVelocity(coef * 0.01 * MAX_SPEED)        # если коэф. < 0, то
    rightMotor.setVelocity(coef * 0.01 * MAX_SPEED)       # движение назад,иначе - вперед
    delay(t * 1000)  # расчитанное время - в секундах, а функция delay принимает миллисекунды
    stop() # вызов функции остановки

HEADINGS = {          # Словарь направлений "Север", "Запад", "Юг", "Восток".
    "NORTH": [1, 0],
    "WEST": [0, 1],
    "SOUTH": [-1, 0],
    "EAST": [0, -1]
}

CLOCH_WISE = {        # Словарь вращения робота "По часовой стрелке", "Против часовой стрелки".
    "CW" : [1, -1],
    "CCW": [-1, 1]
}
def orientizeToTheHeading(heading, K = 10):  # Функция ориентации робота по направлениям "Север", "Запад", "Юг", "Восток".
    cpValues = comp.getValues()  # считываем значения с компаса
    L = R = 0 # переменные направления вращения колес
    while round(comp.getValues()[1], 4) != heading[0] and round(comp.getValues()[2], 4) != heading[1]: # пока не достигли
                                                                                                        # нужного направления...
        robot.step(TIME_STEP)  # Обязательно нужен отсчет таймера симулятора! Иначе - сбой работы!
        print('N', round(comp.getValues()[1], 4), 'W', round(comp.getValues()[2], 4)) # Выводим значения с компаса в формате
        N_val = round(comp.getValues()[1], 4)                                         # положительных значений "Север-запад"
        W_val = round(comp.getValues()[2], 4)
        rotation = "CW"           # направление вращения кратчайшего поворота робота вокруг своей оси к желаемому направлению
        if heading == [1, 0]:
            if W_val > 0:
                rotation = "CW"
            else:
                rotation = "CCW"
        if heading == [0, 1]:
            if N_val > 0:
                rotation = "CCW"
            else:
                rotation = "CW"
        if heading == [-1, 0]:
            if W_val > 0:
                rotation = "CCW"
            else:
```

```
                rotation = "CW"
        if heading == [0, -1]:
            if N_val > 0:
                rotation = "CW"
            else:
                rotation = "CCW"
        L = CLOCH_WISE.get(rotation)[0]  # Назначение переменной направления вращения левого колеса
        R = CLOCH_WISE.get(rotation)[1]  # Назначения переменной направления вращения правого колеса
        leftMotor.setVelocity(L * K * 0.01 * MAX_SPEED)  # K % - процент скорости вращения колес
        rightMotor.setVelocity(R * K * 0.01 * MAX_SPEED) # Задан по умолчанию 10 %, при вызове можно задать другой
    stop()
                                    # Изначально робот ориентирован случайным образом относительно Севера
orientizeToTheHeading(HEADINGS.get("NORTH"))  # Поворот на Север
move_straight(0.3, 100)                 # Движение вперед на 0,3 м со скоростью 100% от максимальной
orientizeToTheHeading(HEADINGS.get("WEST"))   # Поворот на Запад
move_straight(0.3, 100)                 # Движение вперед на 0,3 м со скоростью 100% от максимальной
orientizeToTheHeading(HEADINGS.get("EAST"))   # Поворот на Восток
move_straight(0.6, 100)                 # Движение вперед на 0,6 м со скоростью 100% от максимальной
orientizeToTheHeading(HEADINGS.get("SOUTH"))  # Поворот на Юг
move_straight(0.3, 100)                 # Движение вперед на 0,3 м со скоростью 100% от максимальной
orientizeToTheHeading(HEADINGS.get("WEST"))   # Поворот на Запад
move_straight(0.3, 100)                 # Движение вперед на 0,3 м со скоростью 100% от максимальной
orientizeToTheHeading(HEADINGS.get("NORTH"))  # Поворот на Север (Возврат)

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

Порядок выполнения работы

1. Запустить Webots.
2. Ознакомиться с реализацией алгоритма движения мобильной платформы, основанным на ориентировании по компасу для робота e-puck на языке Python, приведенной в разделе «Краткие теоретические сведения».
3. Подготовить мир симуляции в соответствии с заданием.
4. Реализовать алгоритм движения робота e-Puck на языке Python в соответствии с заданием.

Задание

Необходимо создать и настроить мир симуляции и программу-контроллер, в соответствии с разделом «Краткие теоретические сведения» и вариантом карты препятствий, добавляемой в мир симуляции.

Самостоятельным заданием будет следующая задача – реализовать и оптимизировать алгоритм навигации по углам поворота робота относительно сторон света (Север, Юг, Запад, Восток) в лабиринте, составленным в соответствии с выданным Вам вариантом, для робота e-puck на языке Python. Препятствием может служить что угодно, что не может передвинуть робот e-puck – коробки, боксы, контейнеры и т.п. Робот должен доехать от точки старта, до точке финиша, не задев объекты-препятствия.

Внимание! Варианты выданы отдельным файлом, прилагается вместе с текстом лабораторной работы. Их распределение уточняется на лабораторной работе у преподавателя. Название файла «Варианты индивидуальных полигонов.pdf»

Допускается использовать функции, разработанные в предыдущих лабораторных работах для передвижения, при условии, что в алгоритме задействован компас.

Содержание отчета по работе

Отчет должен содержать выполненные следующие пункты:

1. Составленная блок-схема программы.
2. Настроенный мир симуляции для выполнения задания.
3. Код-листинг программы по заданию.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Webots.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора.
3. Изложить материал раздела «Краткие теоретические сведения» и проделать практические предписания.
4. Ответить на вопросы.
5. Озвучить задание лабораторной работы.

6. После завершения работы проверить успешность выполнения учащимися самостоятельного задания.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учащимся.
2. Проверить работу моделирования в симуляторе – в случае ошибок, исправить программу вместе с учащимся.

Лабораторная работа №7 «Применение технологии распознавания образов в телеметрических роботизированных системах»

Цель работы

1. Настроить мир и контроллер для практической работы;
2. ознакомиться с реализацией алгоритма распознавания образов с помощью видеокамеры;
3. реализовать алгоритм автоматизированной работы робота с применением технологии распознавания образов для робота E-Puck на языке Python.

Краткие теоретические сведения

Симуляция работы видеокамеры в Webots

В программе моделирования робототехники Webots предусмотрено применение супервизорного управления роботами, позволяющее создавать роботизированные средства и системы телеприсутствия.

Телеприсутствие — это набор технологий, позволяющих оператору, например с помощью специальных устройств (телеуправляемых роботов), получить визуальную информацию о месте, отличном от его физического местоположения и опосредованно взаимодействовать с окружающим это место пространством.

Одно из таких применений может быть реализовано с помощью видеокамеры, встроенной в робота. Робот e-puck оснащен видеокамерой. Расположение встроенной видеокамеры изображено на рисунке 1 и угловая ориентация – в таблице на рисунке 2.

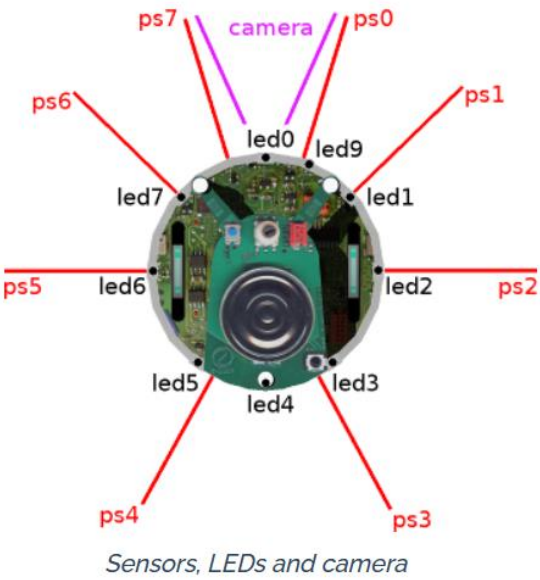


Рисунок 1. Расположение видеокамеры на роботе e-puck.

Device	x (m)	y (m)	z (m)	Orientation (rad)
ps0	0.010	0.033	-0.030	1.27
ps1	0.025	0.033	-0.022	0.77
ps2	0.031	0.033	0.00	0.00
ps3	0.015	0.033	0.030	5.21
ps4	-0.015	0.033	0.030	4.21
ps5	-0.031	0.033	0.00	3.14159
ps6	-0.025	0.033	-0.022	2.37
ps7	-0.010	0.033	-0.030	1.87
camera	0.000	0.028	-0.030	4.71239

Рисунок 2. Таблица расположения датчиков и камеры.

Напоминаем, что весьма полезным было бы более подробное ознакомление с функциональными возможностями робота e-puck по ссылке.

Создание программы-контроллера на языке Python

Перед началом работы создадим отдельный файл-контроллер, чтобы отделять выполнение лабораторных работ друг от друга и обеспечить грамотное хранение и применение собственных наработок, которые обязательно пригодятся в дальнейшем. Чтобы добавить контроллер, необходимо выбрать в ленте вкладку «Wizards», и нажать на вкладку в выпадающем меню «New Robot Controller…», как на рисунке 3.

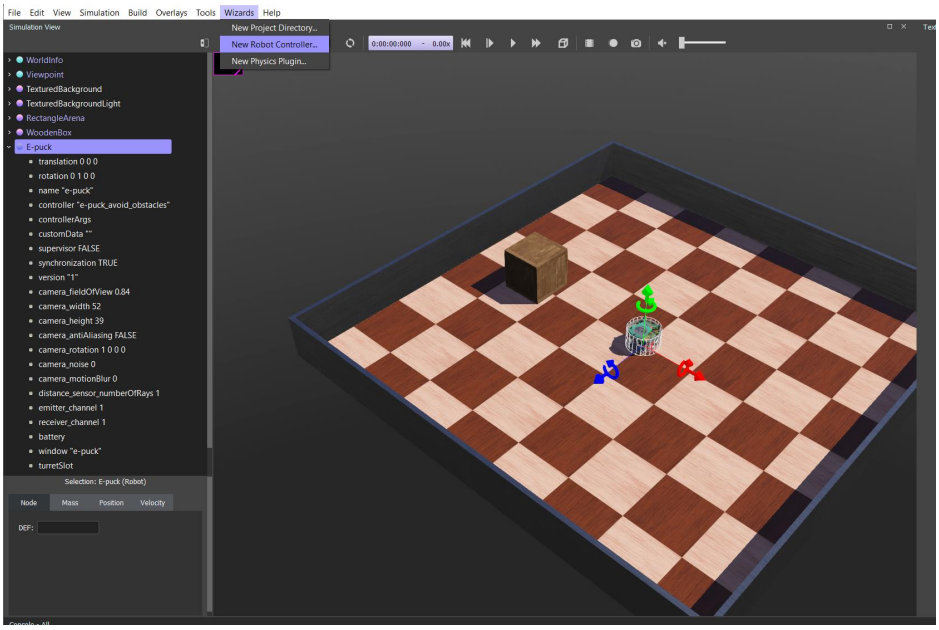


Рисунок 3. Создание нового контроллера.

Откроется диалоговое окно, нажмите «Next», чтобы продолжить. Следующая страница требует выбора языка программирования – выбираем Python и нажимаем далее (Next). В следующем окне – называем контроллер так: «Lab_work_7_controller_*Ваша фамилия латиницей*» и переходим в следующее окно.

Затем обязательно ставим галочку около «Open ‘Lab_work_7_controller_*Ваша фамилия латиницей*.py’ in Text Editor», чтобы открыть текст контроллера в редакторе, как на рисунке 4, и нажимаем «Finish».

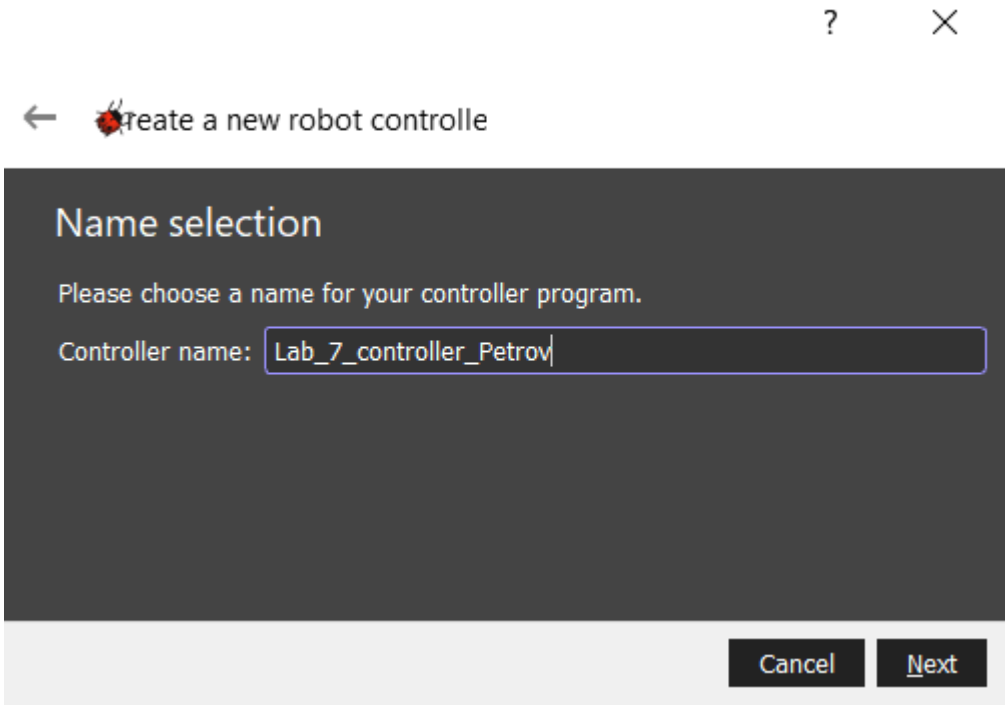


Рисунок 4. Завершение создания нового контроллера.

Теперь необходимо подключить контроллер к роботу e-puck, чтобы мы смогли сразу тестировать написанные программы.

Для этого необходимо выбрать в дереве объектов робота, найти поле «Controller» и нажать кнопку «select…», как на рисунке 5.

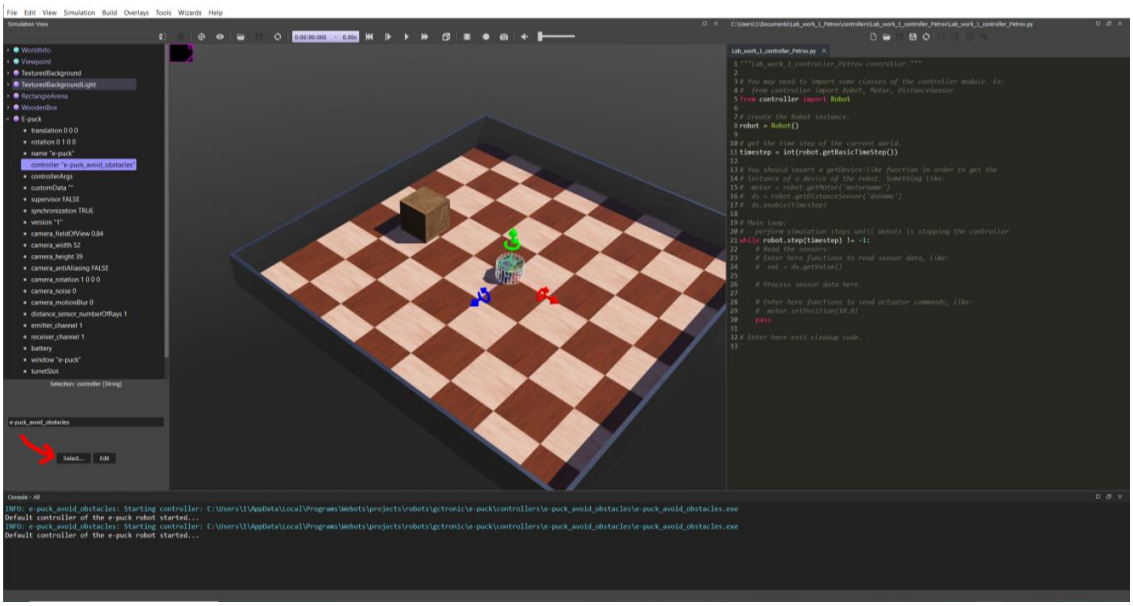


Рисунок 5. Подключение нового контроллера.

В диалоговом окне выбираем только что созданный контроллер с именем «Lab_work_7_controller_*Ваша фамилия латиницей*» (рисунок 6) и нажимаем «ОК».

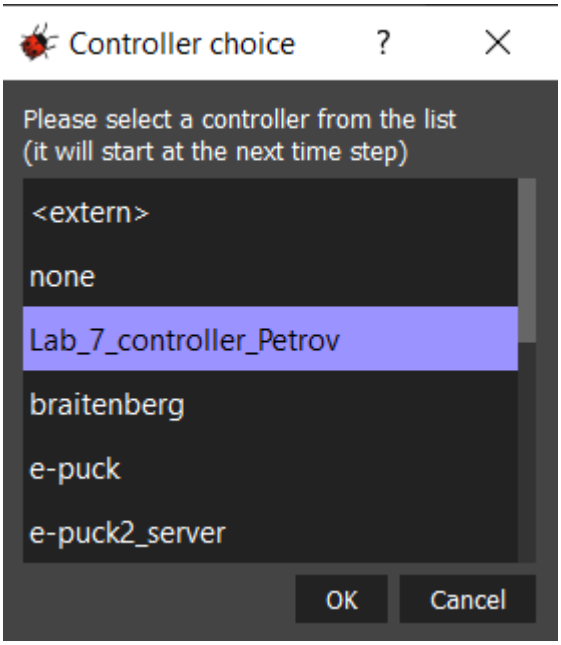


Рисунок 6. Подключение нового контроллера. Диалоговое окно.

Встроенная в робота e-puck видеокамера не поддерживает возможность распознавать, однако у робота присутствует в наличии место для крепления модуля распознавания – дополнительной камеры, которую мы можем добавить и настроить по необходимости.

Для того, чтобы добавить такую камеру, выберете вкладку «turretSlot» в дереве добавленных объектов, во вкладке «E-puck», как на рисунке 7. Далее дважды кликните ЛКМ и выберете в выпадающем меню «Base nodes» поле «Camera» и нажмите кнопку «Add», чтобы добавить камеру, как изображено на рисунке 8.

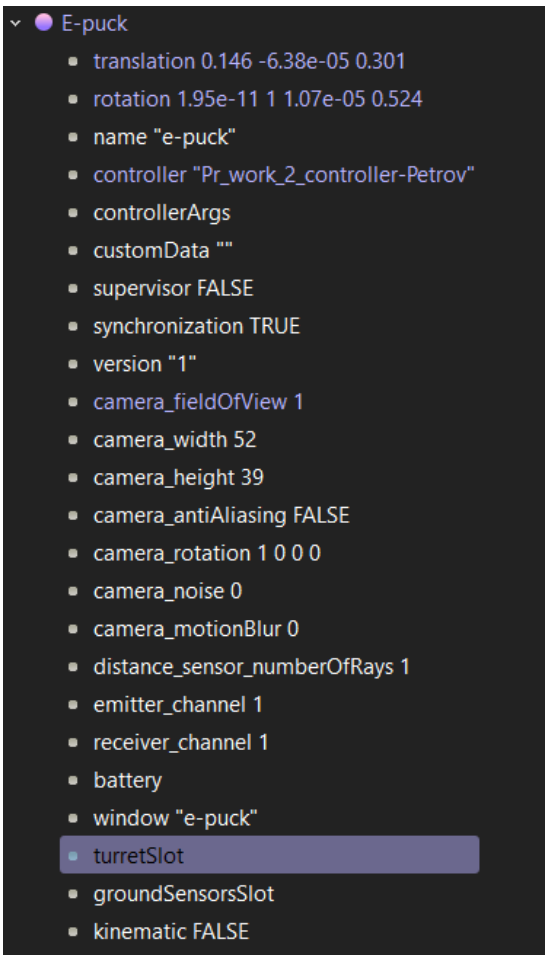


Рисунок 7. Выбор слота дополнительного крепления.

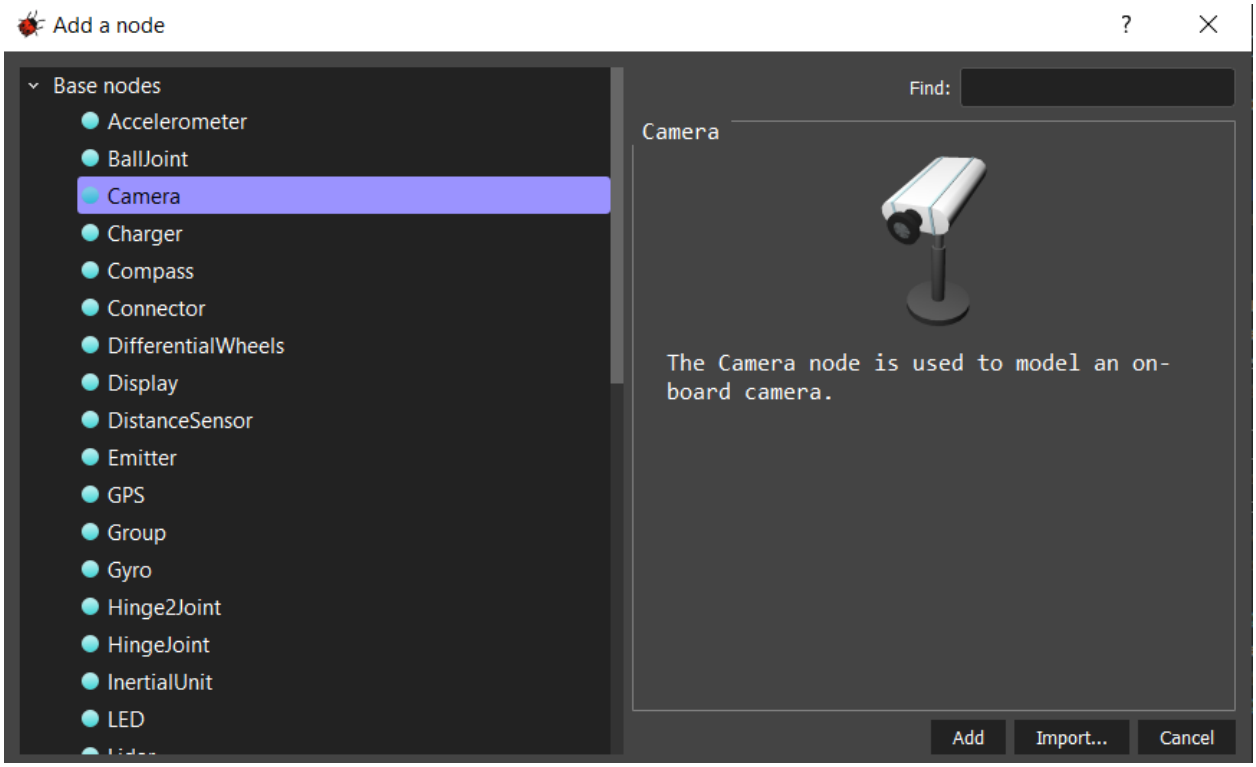


Рисунок 8. Выбор дополнительной камеры.

Далее выполните все настройки, в соответствии с рисунком 9. Настройки width и height можете изменить как Вам удобнее, в зависимости от разрешения монитора.

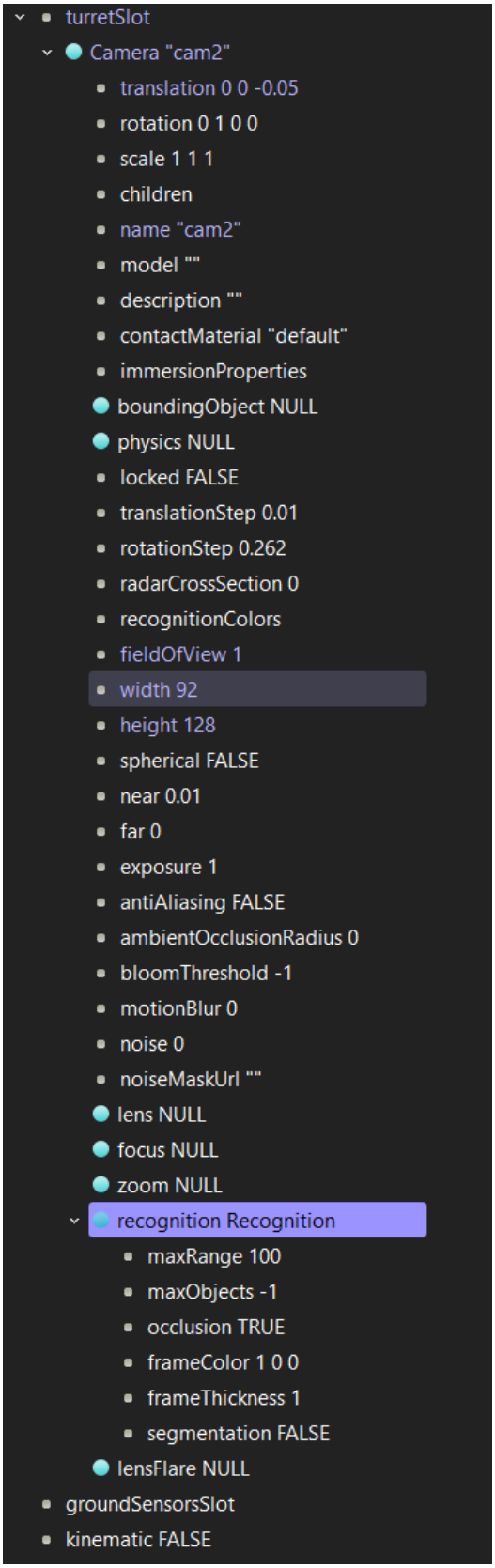


Рисунок 9. Настройки дополнительной камеры «cam2».

Конфигурация мира симуляции в Webots

Для решения сегодняшней задачи нам будет необходимо добавить новые объекты, обладающие свойством «RecognitionColors», которое описывает то, как будет воспринят объект камерой, если в ее объектив попадет предмет с таким свойством. К таким объектам относится класс «Solid».

Теперь, добавим объект класса «Solid» в мир симуляции Webots. Для этого нажмем ЛКМ на любой элемент списка в дереве объектов, тогда нам станет доступна возможность добавлять новые объекты в среду. Нажмем сочетание клавиш (Ctrl+Shift+A), или нажав ПКМ в свободной части дерева объектов и выберем «Add New» (также это можно сделать, нажав на специальный значок, графически схожий со знаком «плюс» на панели-ленте в верхней части экрана). Появится окно, как на рисунке 10. В этом окне нажимаем на стрелочку рядом с «Base nodes (Webots Projects)» и в выпадающем списке выбираем «Solid» и нажимаем кнопку «Add». Также, можно воспользоваться поиском в поле «Find».

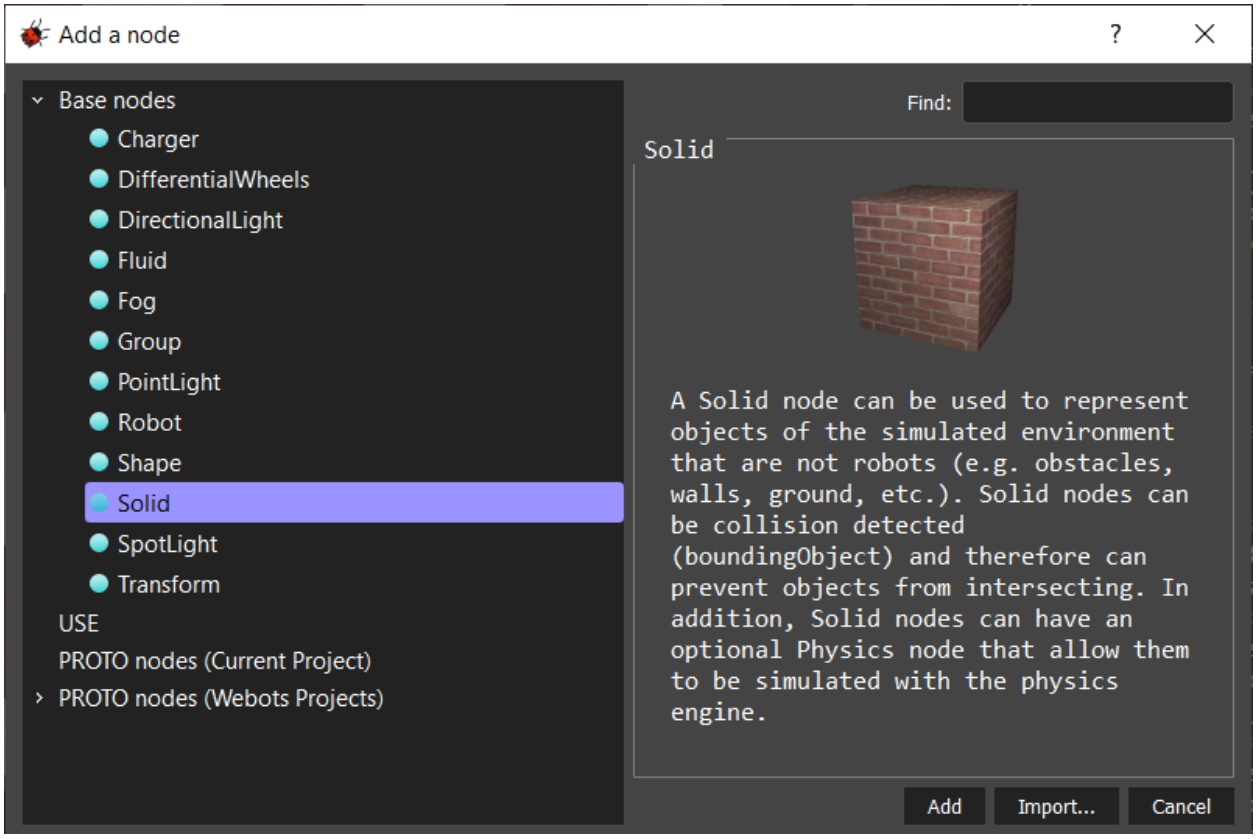


Рисунок 10. Добавление объекта класса Solid.

На панели объектов, добавленных в мир симуляции откроем появившийся класс «Solid» (Рисунок 11).

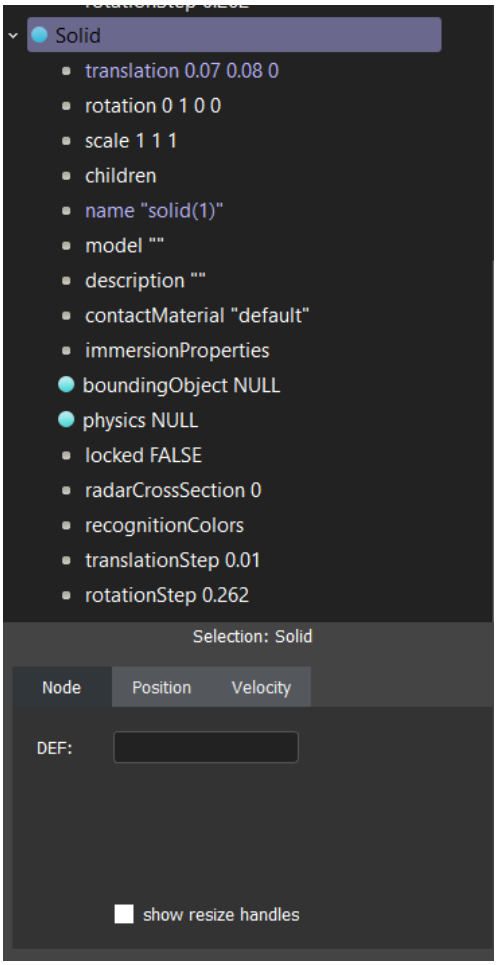


Рисунок 11. Solid на панели объектов в мире симуляции.

На данный момент в мире симуляции нет отображения реального объекта, так как мы добавили только класс. Чтобы добавить его отображение, нам необходимо воспользоваться свойством полиморфизма классов в симуляторе – создать дочерний класс объектов, имеющих отображение и наследующих свойства класса «Solid».

В объектно-ориентированном программировании абстрактные типы данных называются классами.

Суперкласс (англ. super class), родительский класс (англ. parent class), предок, родитель или надкласс — класс, производящий наследование в подклассах, т. е. класс, от которого наследуются другие классы.

Подкласс (англ. subclass), производный класс (англ. derived class), дочерний класс (англ. child class), класс потомок, класс наследник или класс-реализатор — класс, наследуемый от суперкласса или интерфейса, т. е. класс определённый через наследование от другого класса или нескольких таких классов. Подклассом может быть суперкласс.

Базовый класс (англ. base class) — это класс, находящийся на вершине иерархии наследования классов и в основании дерева подклассов, т. е. не являющийся подклассом и не имеющий наследований от других суперклассов или интерфейсов. Базовым классом может быть абстрактный класс и интерфейс. Любой не базовый класс является подклассом.

Интерфейс (англ. interface) — это структура, определяющая интерфейс класса, состоящий из абстрактных методов. Интерфейсы участвуют в иерархии наследований классов и интерфейсов.

Суперинтерфейс (англ. super interface) или интерфейс-предок — это аналог суперкласса в иерархии наследований, т. е. это интерфейс, производящий наследование в подклассах и подинтерфейсах.

Интерфейс-потомок, интерфейс-наследник или производный интерфейс (англ. derived interface) — это аналог подкласса в иерархии наследований интерфейсов, т. е. это интерфейс, наследуемый от одного или нескольких суперинтерфейсов.

Базовый интерфейс — это аналог базового класса в иерархии наследований интерфейсов, т. е. это интерфейс, находящийся на вершине иерархии наследования.

Иерархия наследования или иерархия классов — дерево, элементами которого являются классы и интерфейсы.

Проще говоря, класс — это описание того, какими свойствами и поведением будет обладать объект. А объект — это экземпляр с собственным состоянием этих свойств.

Мы говорим «свойства и поведение», но звучит это весьма абстрактно. Привычнее для программиста будет звучать так: «переменные и функции». На самом деле «свойства» — это такие же обычные переменные, просто они являются атрибутами какого-то объекта (их называют полями объекта). Аналогично «поведение» — это функции объекта (их называют методами), которые тоже являются атрибутами объекта. Разница между методом объекта и обычной функцией лишь в том, что метод имеет доступ к собственному состоянию через поля.

Наследование (англ. inheritance) — концепция объектно-ориентированного программирования, согласно которой абстрактный тип данных может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов программного обеспечения.

Дважды нажимаем ЛКМ на поле «children», и в появившимся окне открываем выпадающий список «Base nodes», в котором выберем поле «Shape», и нажимаем кнопку «Add», как на рисунке 12.

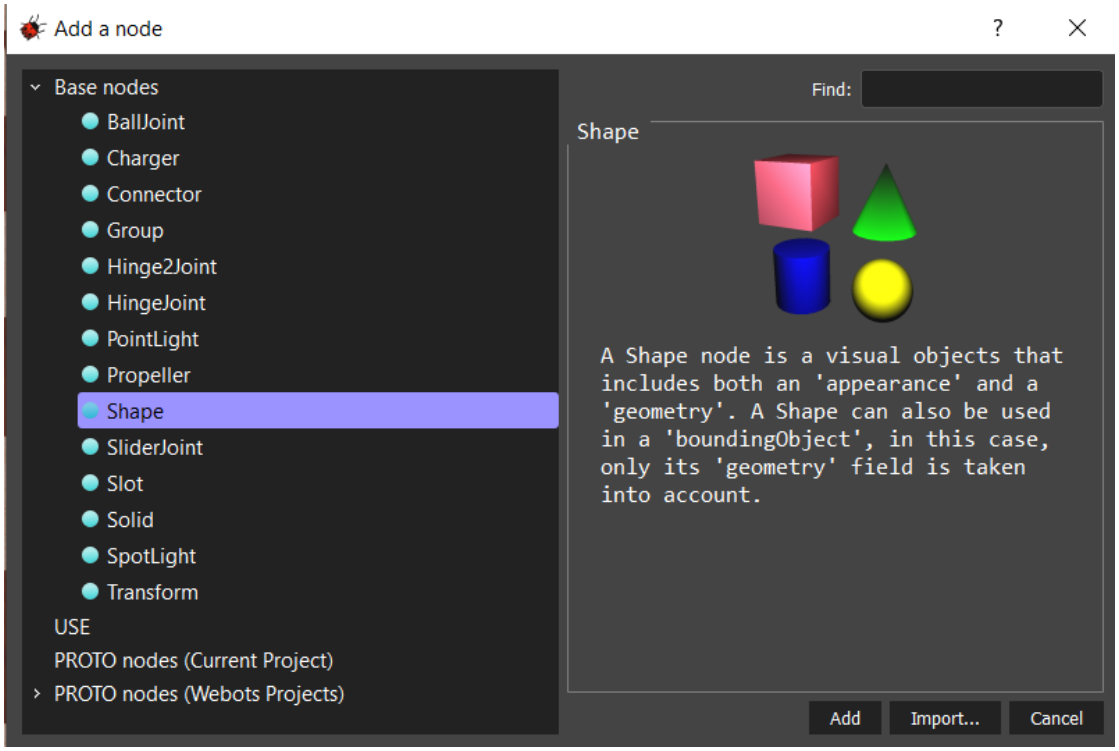


Рисунок 12. Выбор дочернего класса.

Теперь зададим описание внешних свойств «appearance» в появившемся поле «Shape» и геометрию «geometry» (Рисунок 13).

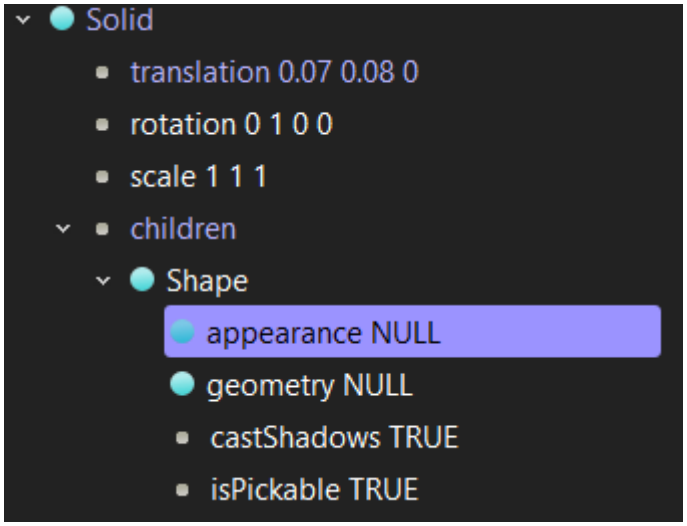


Рисунок 13. Поле настройки формы твердого объекта.

Для этого, дважды кликнув на поле «appearance» с помощью ЛКМ, перейдем в окно, как на рисунке 14, и выберем тип «PBRAppearance», и нажмем «Add».

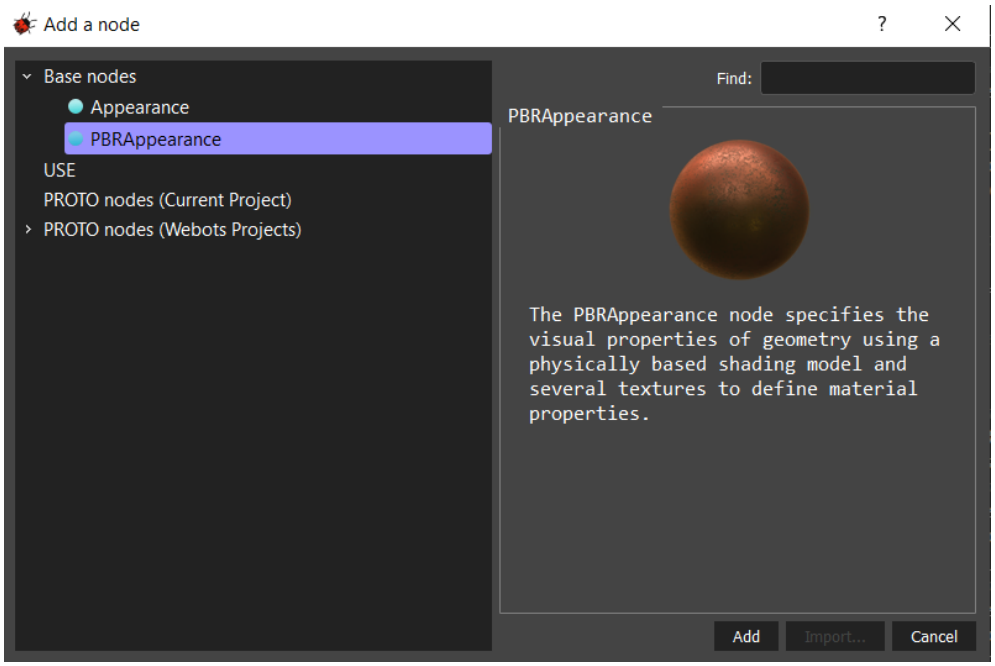


Рисунок 14. Добавление визуальных характеристик.

Аналогично добавим геометрию, определив ее как «Box», в точности, как на рисунке 15.

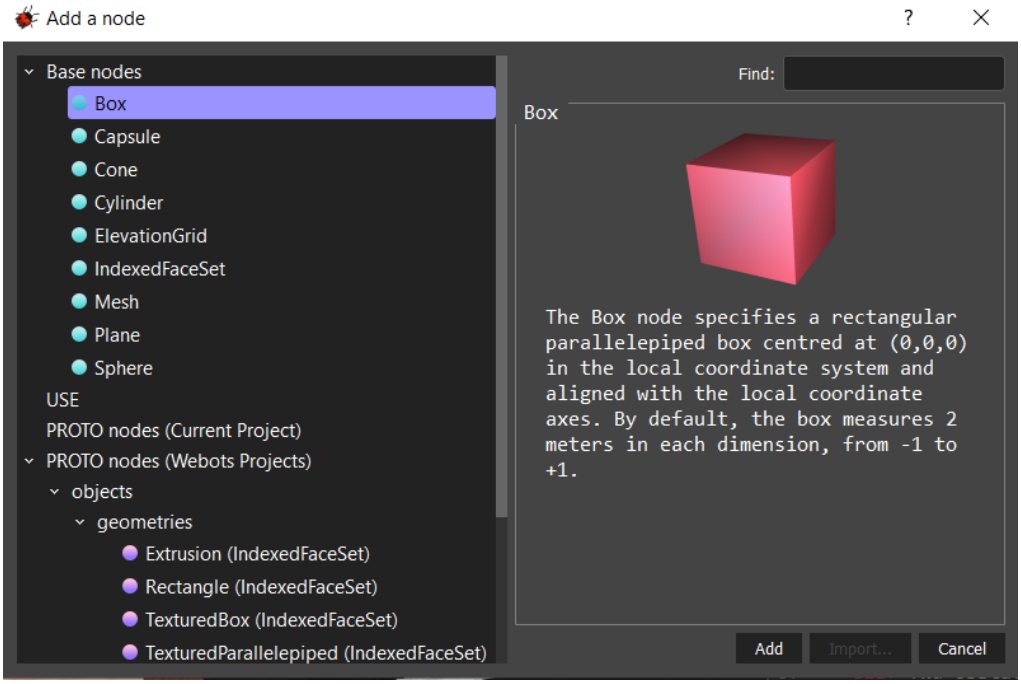


Рисунок 15. Добавление геометрии коробки.

Теперь появится отображение созданного объекта «box» класса «Solid». Полчится также, как на рисунке 16.

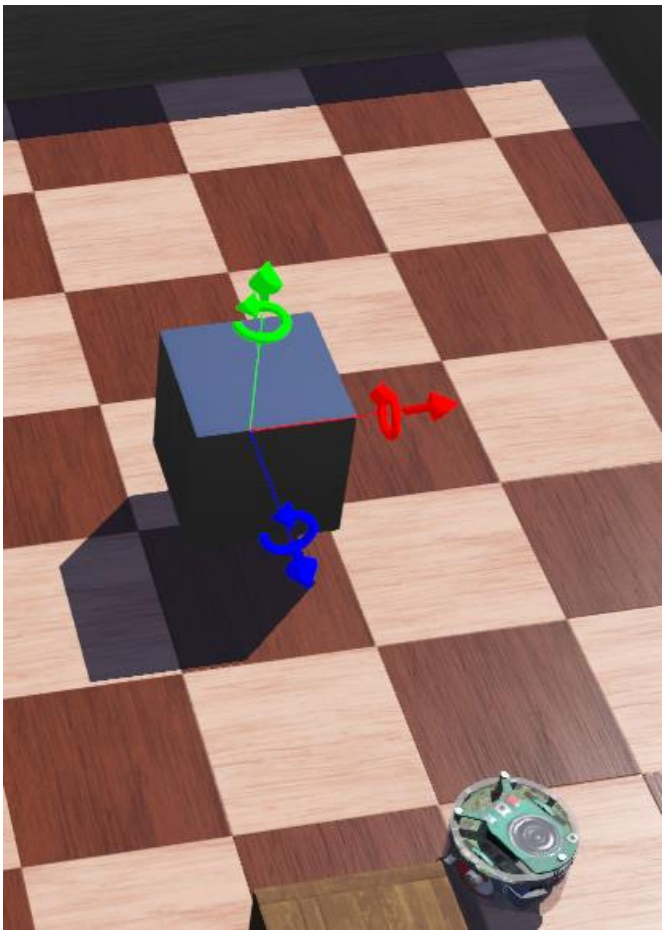


Рисунок 16. Появившийся объект.

Теперь настроим физические свойства появившегося объекта – коробки. Для этого в поле «Physics», аналогичным образом добавляем одноименное свойство, как на рисунке 17.

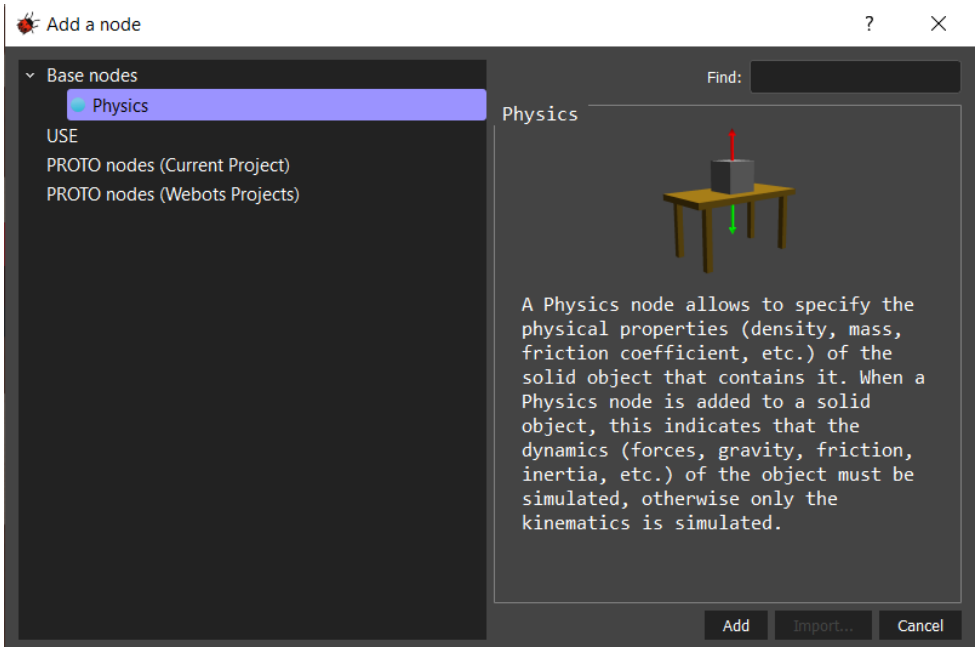


Рисунок 17. Добавление свойств физики.

Теперь в выпадающем меню вкладки «Physics», задайте массу 0.05 кг.

Далее, добавим контур геометрической формы (поле «boundingObject»), чтобы примененные физические свойства ограничивались реальной формой объекта. Сделайте в точности, как на рисунке 18.

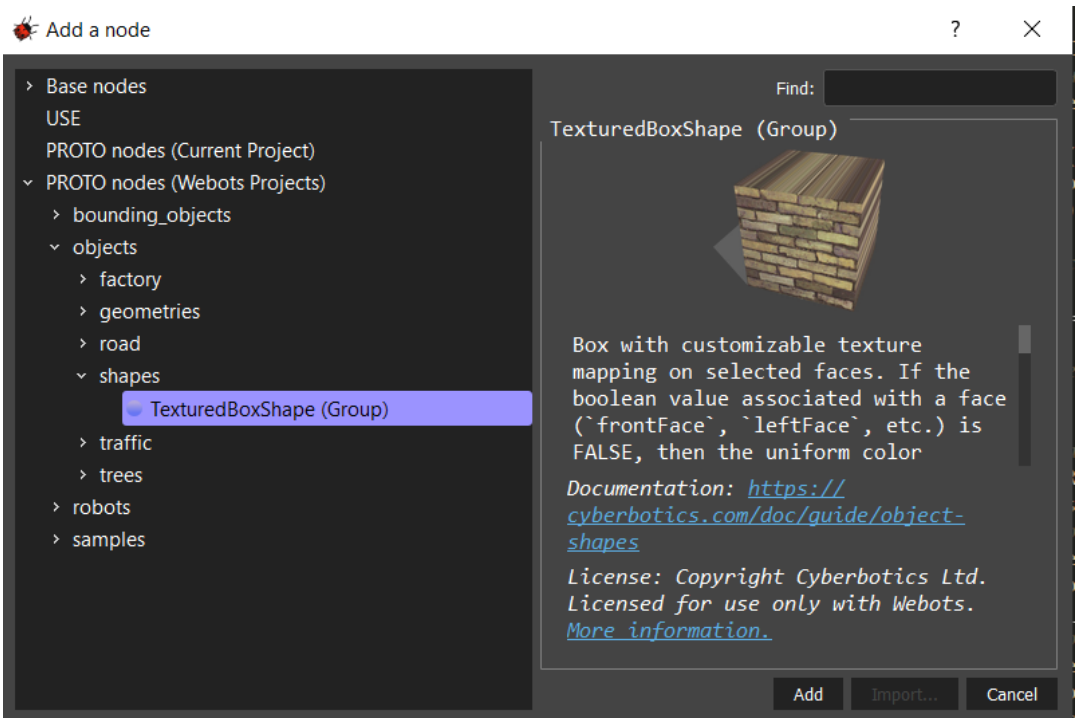


Рисунок 18. Выбор свойства для поля «boundingObject».

Настройте масштаб созданного объекта 0.5 (поле «Scale»).

И самое главное, для чего вообще создавался такой объект – настройка поля «RecognitionColors», параметры которого будут:

red = 1, green = 0, blue = 0

Не трудно догадаться, что видеокамера будет определять контрастный красный образ.

В конце концов должно получиться так, как изображено на рисунке 19.

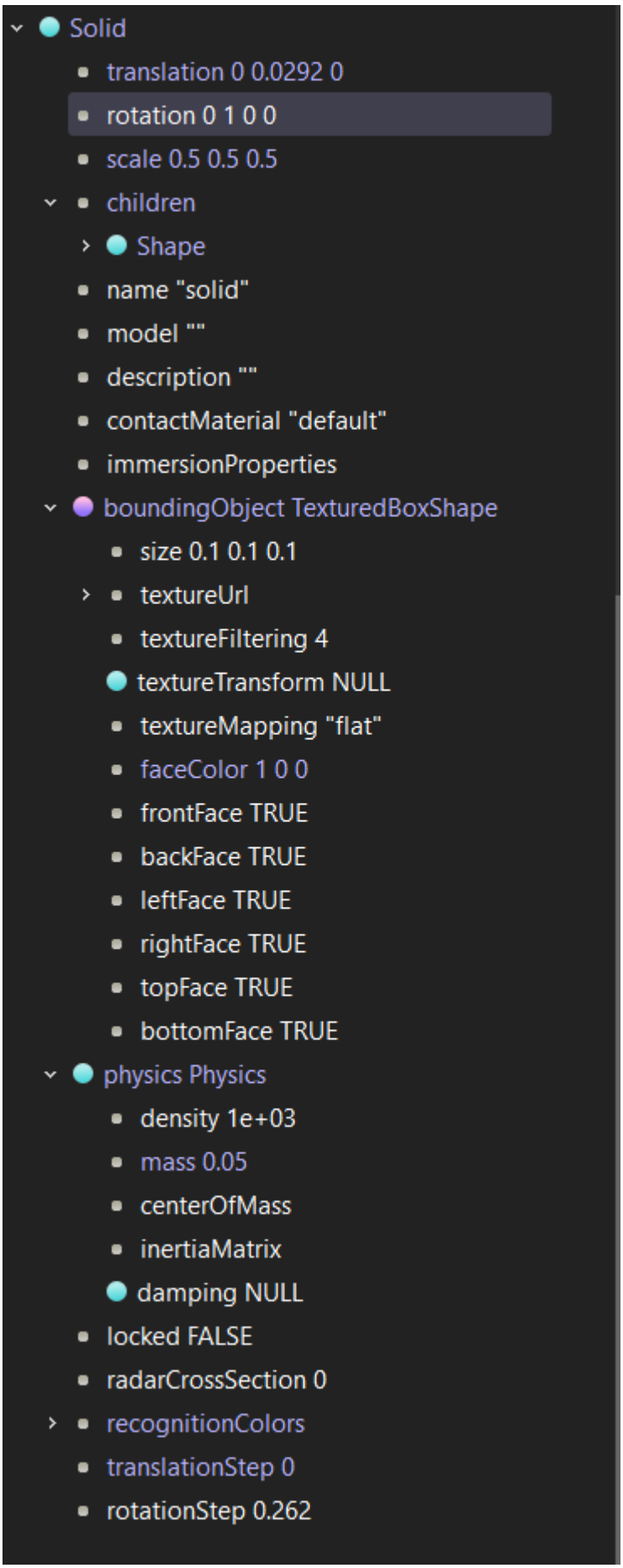


Рисунок 19. Заполненные поля как образец.

Объект успешно добавлен в среду симуляции, что отражено на рисунке 20.

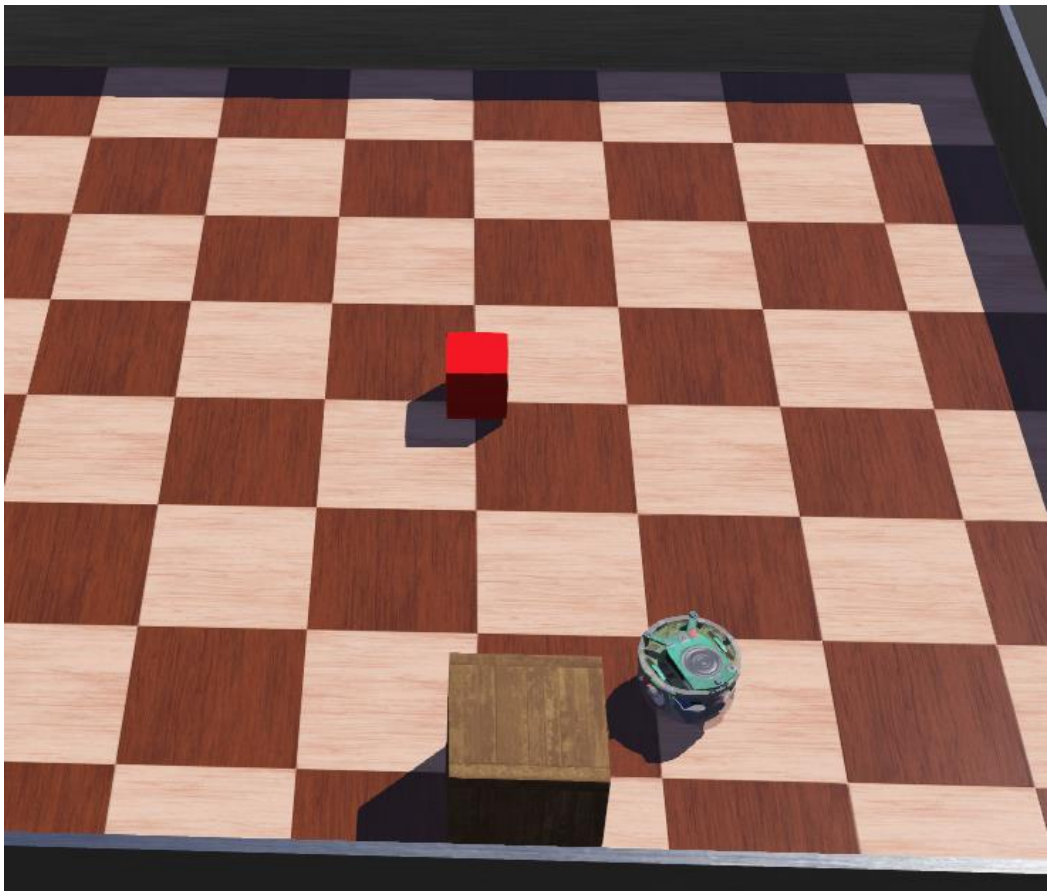


Рисунок 20. Добавленный объект класса «Solid».

Перемещать объекты можно с помощью мыши, предварительно зажав клавишу Shift, а также, с помощью стрелок по осям, исходящим из выбранного объекта. Копировать и вставлять объекты – с помощью сочетания клавиш (Ctrl+C, Ctrl+V).

Дополните поле, случайно расставив такие объекты по Вашей арене.

Нажмите сочетание клавиш (Ctrl+Shift+S), или нажмите на иконку, графически схожую с дискетой, в левой верхней части экрана для сохранения созданного мира.

На примере сегодняшней лабораторной работы напишем программу, реализующую распознавание объектов красного цвета для дальнейшей их идентификации как «вражеские» объекты для взаимодействия их с роботом посредством толкания. Все остальные объекты будут «дружественными», их необходимо осторожно объезжать роботом.

Реализация алгоритма автоматизированной работы робота с применением технологии распознавания образов для робота E-Puck на языке Python

Алгоритм будет заключаться в том, что камеры, которая в соответствии с рисунком 9, была названа «cam2», будет распознавать объекты перед роботом. В случае, если был замечен объект красного цвета, он идентифицируется как «враг», и робот, не меняя курса, едет в его сторону. Если этот объект находится на близком расстоянии, робот ускоряется, толкая «врага» перед собой. Все остальные объекты, не идентифицированные как враги (т.е. не красного цвета) должны быть не затронуты роботом – e-puck должен их объезжать, ориентируясь с помощью датчиков препятствия (proximity sensors), закрепленных по периметру робота (рисунок 1). Таким образом проводится постоянный мониторинг датчиков для определения стороны, с которой было обнаружено препятствие для дальнейшего разворота робота в противоположную сторону от препятствия во избежание столкновения с ним.

Для того, чтобы инициализировать датчики необходимо обратиться к таблице, приведенной на рисунке 21.

Device	Name
Motors	'left wheel motor' and 'right wheel motor'
Position sensors	'left wheel sensor' and 'right wheel sensor'
Proximity sensors	'ps0' to 'ps7'
Light sensors	'ls0' to 'ls7'
LEDs	'led0' to 'led7' (e-puck ring), 'led8' (body) and 'led9' (front)
Camera	'camera'
Accelerometer	'accelerometer'
Gyro	'gyro'
Ground sensors (extension)	'gs0', 'gs1' and 'gs2'
Speaker	'speaker'

Рисунок 21. Таблица с названиями устройств на роботе.

Теперь рассмотрим алгоритм программы (рисунок 22), позволяющей выполнять функции изложенные в описании выше.

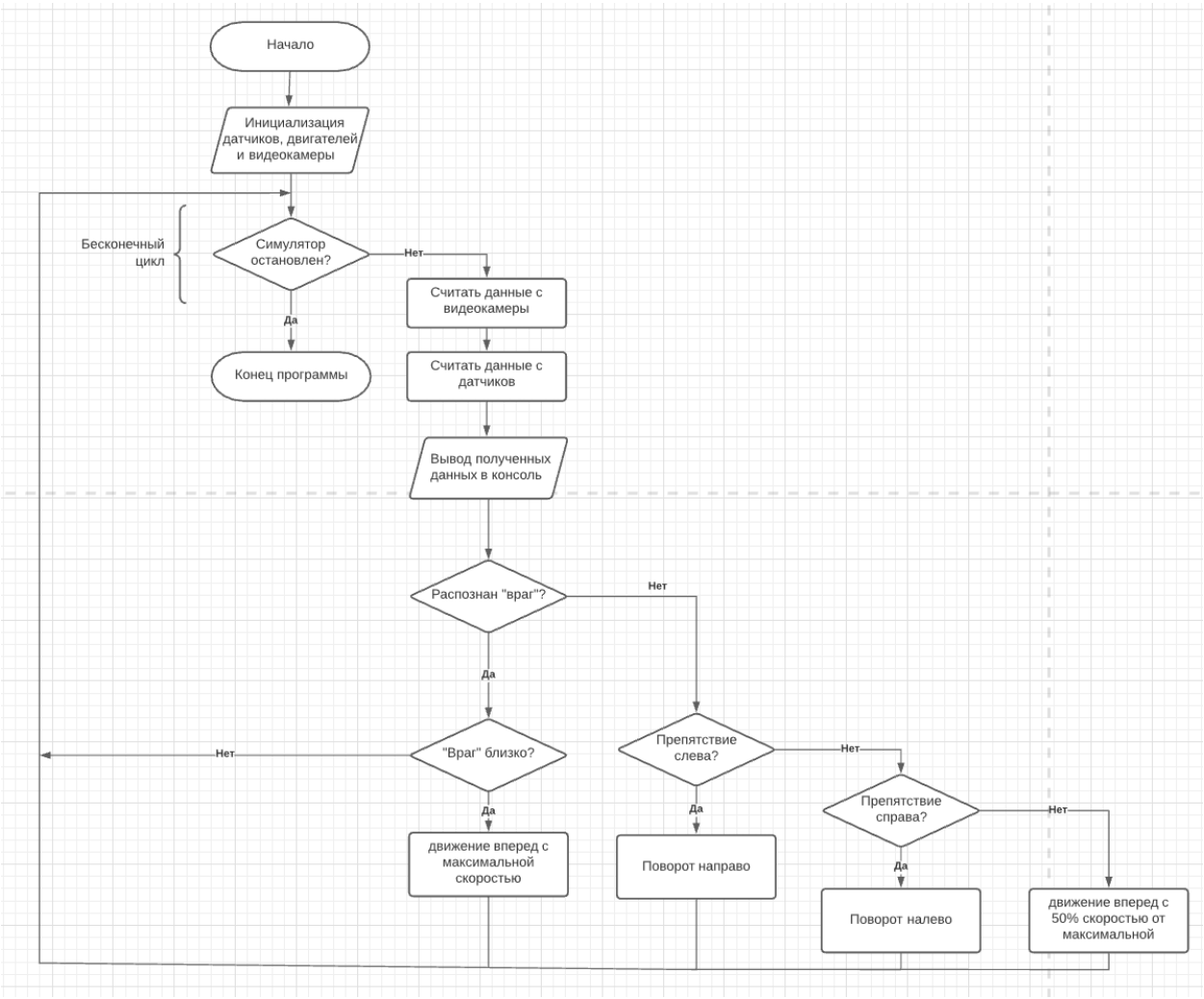


Рисунок 22. Алгоритм работы робота. Распознавание и объезд препятствий.

Результат работы программы (рисунок 23).

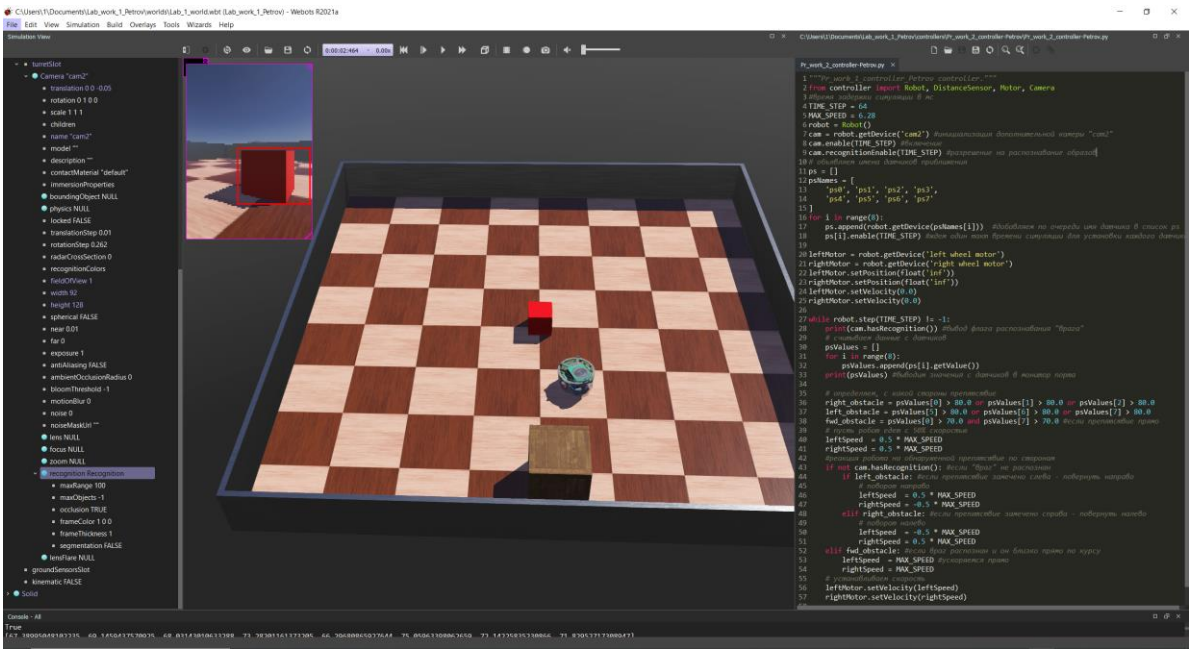


Рисунок 23. Демонстрация работы программы.

Данный алгоритм получил такую реализацию в приведенном коде.

```
"""Lab_7_controller_Petrov controller."""
from controller import Robot, DistanceSensor, Motor, Camera, CameraRecognitionObject
# Создаём объект класса Robot
robot = Robot()

MAX_SPEED = 6.28 # максимальная скорость вращения колеса (рад/с)
WHEEL_RAD = 0.0205 # радиус колеса (м)
AXLE_LENGTH = 0.052 # длина оси конструкции робота (м)

# Получить временной шаг для текущего мира симуляции. *(1) ПОДРОБНЕЕ В ТЕКСТЕ ЛАБОРАТОРНОЙ РАБОТЫ 1
TIME_STEP = int(robot.getBasicTimeStep()) #int(*любой тип данных*) - переводит тип данных в целочисленный
cam = robot.getDevice('cam2') # инициализация дополнительной камеры "cam2"
cam.enable(TIME_STEP) #включение камеры
rn = CameraRecognitionObject()
cam.recognitionEnable(TIME_STEP) #разрешение на распознавание образов
cam.enableRecognitionSegmentation()

ps = []  # создаем список для хранения имен датчиков приближения
# объявляем имена датчиков приближения (заданы в документации к роботу)
psNames = [
    'ps0', 'ps1', 'ps2', 'ps3',
    'ps4', 'ps5', 'ps6', 'ps7'
]

for i in range(8):
    ps.append(robot.getDevice(psNames[i]))  #добавляем по очереди имя датчика в список ps
    ps[i].enable(TIME_STEP) #ждем один такт времени симуляции для установки каждого датчика

leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

leftMotor.setPosition(float('inf')) # Разрешает поворачиваться колесу на полный оборот
rightMotor.setPosition(float('inf')) # аналогично для другого мотора
leftMotor.setVelocity(0.0)   # устанавливает скорость вращения колеса 0 (рад/с)
rightMotor.setVelocity(0.0)   # аналогично для другого мотора

CAM2_WIDTH = 256 # Константа ширины камеры (поле width в TurretSlot)
ERROR = 4 # Константа ошибки по ширине камеры

while robot.step(TIME_STEP) != -1:
    print(cam.getRecognitionNumberOfObjects()) #вывод флага распознавания "врага"

    # считываем данные с датчиков
```

```
    psValues = []
    for i in range(8):
        psValues.append(ps[i].getValue())
    print(psValues) #выводим значения с датчиков в монитор порта

    # определяем, с какой стороны препятствие
    right_obstacle = psValues[0] > 80.0 or psValues[1] > 80.0 or psValues[2] > 80.0 # если препятствие справа
    left_obstacle = psValues[5] > 80.0 or psValues[6] > 80.0 or psValues[7] > 80.0 # если препятствие слева
    fwd_obstacle = psValues[0] > 70.0 and psValues[7] > 70.0 # если препятствие прямо
    # пусть робот едет с 50% скоростью
    leftSpeed  = 0.5 * MAX_SPEED
    rightSpeed = 0.5 * MAX_SPEED
    #реакция робота на обнаруженной препятствие по сторонам
    if not cam.getRecognitionNumberOfObjects(): #если "враг" не распознан
        if left_obstacle: #если препятствие замечено слева - повернуть направо
            # поворот направо
            leftSpeed  = 0.5 * MAX_SPEED
            rightSpeed = -0.5 * MAX_SPEED
        elif right_obstacle: #если препятствие замечено справа - повернуть налево
            # поворот налево
            leftSpeed  = -0.5 * MAX_SPEED
            rightSpeed = 0.5 * MAX_SPEED
    elif fwd_obstacle: #если враг распознан и он близко прямо по курсу
            leftSpeed  = MAX_SPEED #ускоряемся прямо
            rightSpeed = MAX_SPEED
    elif cam.getRecognitionNumberOfObjects():
        firstObject = cam.getRecognitionObjects()[0] #получаем данные о найденном объекте
        id = firstObject.get_id() #достаем id
        position = firstObject.get_position_on_image()[0] # достаем позицию на картинке - список из координат
        print(position)
        if position < CAM2_WIDTH / 2 - ERROR: # если центр объекта-врага левее, чем направление движения
            leftSpeed  = -0.1 * MAX_SPEED     # повернуть навстречу
            rightSpeed = 0.1 * MAX_SPEED
        elif position > CAM2_WIDTH / 2 + ERROR:  # если центр объекта-врага правее, чем направление движения
            leftSpeed  = 0.1 * MAX_SPEED        # повернуть навстречу
            rightSpeed = -0.1 * MAX_SPEED
    leftMotor.setVelocity(leftSpeed)         # Установить скорость моторов
    rightMotor.setVelocity(rightSpeed)

```

Порядок выполнения работы

1. Запустить Webots.

2. Ознакомиться с реализацией алгоритма распознавания образов с помощью видеокамеры для робота e-puck на языке Python, приведенной в разделе «Краткие теоретические сведения».
3. Подготовить мир симуляции в соответствии с заданием.
4. Реализовать алгоритм движения робота e-Puck на языке Python в соответствии с заданием.

Задание

Необходимо создать и настроить мир симуляции и программу-контроллер, в соответствии с разделом «Краткие теоретические сведения» и вариантом карты препятствий, добавляемой в мир симуляции.

Самостоятельным заданием является следующая задача – реализовать алгоритм автоматизированного поиска для робота E-Puck на языке Python с применением технологии распознавания образов объекта-«врага», помеченного красным цветом, и его «устранение» - любое механическое взаимодействие с ним – толкнуть, перевернуть и т.п. Объект-«враг» должен быть меньше по массе, чем робот для наглядной демонстрации взаимодействия. Препятствиями может служить что угодно, что не может передвинуть робот e-puck – коробки, боксы, контейнеры и т.п. Робот должен доехать от точки старта, до точке финиша, не задев объекты-препятствия.

Внимание! Варианты выданы отдельным файлом, прилагается вместе с текстом лабораторной работы. Их распределение уточняется на лабораторной работе у преподавателя. Название файла «Варианты индивидуальных полигонов.pdf»

Допускается использовать функции, разработанные в предыдущих лабораторных работах для передвижения, при условии, что в алгоритме задействован компас.

Содержание отчета по работе

Отчет должен содержать выполненные следующие пункты:

1. Составленная блок-схема программы.
2. Настроенный мир симуляции для выполнения задания.
3. Код-листинг программы по заданию.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Webots.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора.
3. Изложить материал раздела «Краткие теоретические сведения» и проделать практические предписания.
4. Ответить на вопросы.
5. Озвучить задание лабораторной работы.
6. После завершения работы проверить успешность выполнения учащимися самостоятельного задания.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учащимся.
2. Проверить работу моделирования в симуляторе – в случае ошибок, исправить программу вместе с учащимся.

Лабораторная работа №8 «Изучение принципов работы 3D сканера Planeta 3D и сканирование простых объектов»

Цель

Изучение принципов работы, настройки и функциональных возможностей устройства для сканирования объектов Planeta 3D.

Краткие теоретические сведения

Planeta 3D – это оптический сканер, позволяющий производить сканирование среднегабаритных и крупногабаритных объектов. Сканер может захватывать и строить модели объектов, достигающих по площади 4000x4000x4000 мм.

Рисунок 1: Изображение 3D сканера Planeta 3D

Оптический сканер работает по технологии структурированного света. Устройства, построенные на базе технологии структурированного света, работают не с лазерным, а естественным источником света. Такие устройства проецируют определенный ряд световых паттернов на целевой объект. Отраженное изображение затем захватывается камерами и положение каждой точки на поверхности объекта рассчитывается исходя из полученных искажений паттерна.

Встроенная камера для захвата цветовых текстур позволяет получать качественные 3D модели с реалистичной фактурой. Точность и высокое

разрешение полученных моделей позволяют тратить меньше времени на доработку 3D моделей.

Разрешение сканирования у Planeta3D доходит до 768 вокселей. Воксель – единица измерения объёмного изображения “объемный пиксель”, содержащий значение элемента растра в трёхмерном пространстве. Воксели являются аналогами двумерных пикселей для трёхмерного пространства. В компьютерной графике воксели используются как альтернатива полигонам. Под вокселом обычно понимается виртуальный элемент, соответствующий набору из шести прямоугольных полигонов.

Сканер подключается к ПК кабелем со стороны устройства в разъем Type C и со стороны ПК в разъем USB версии не ниже 3.0. После запуска устройства, сканер не требует калибровки.

Сканер можно использовать в двух режимах:

1. Ручной режим сканирования;
2. Фиксированный.

При работе со сканером в ручном режиме необходимо взять сканер за рукоять и путем перемещения сканера вокруг объекта, произвести сканирование.

Фиксированный режим работы будет рассмотрен в следующей лабораторной работе.

Перед началом 3D сканирования необходимо выполнить следующие действия:

Подключите сенсор к компьютеру;
Запустите программное обеспечение сканера Planeta 3D.

После подключения устройства и запуска ПО можно начинать сканирование объекта.

Перед началом сканирования в рабочей области программы воспроизводится изображение с камеры, направленной на объект сканирования,

изображение области захвата изображения и в правой части предварительный результат сканирования.

Основные рекомендации для получения качественной 3D модели:

1. Объект должен заполнять большую часть кадра.
2. Весь объект на фотографии должен быть в зоне резкости.
3. Старайтесь избегать высоких значений ISO, для уменьшения шума на изображениях.
4. Избегайте жестких теней на снимках.
5. Если на снимках присутствуют тени, они не должны перемещаться относительно объекта съемки.
6. Используйте программу для создания несимметричных фактурных объектов.
7. Для создания прямолинейных, плоских, однотонных, блестящих и прозрачных объектов предполагается использование дополнительных приёмов, которые будут описаны в следующих лабораторных работах.
8. Используйте последние версии драйверов видеокарты и библиотеки CUDA.

Программа позволяет выполнить построение модели без GPU, но с меньшей детализацией и с большими затратами времени.

Задание

С помощью сканера 3DQ Planeta3D произведите сканирование простого объекта в ручном режиме.

Порядок выполнения работы

1. Изучите теоретические сведения перед выполнением работы.
2. Произведите подготовку устройства к сканированию.
3. Запустите программное обеспечение Planeta 3D.
4. Установите предложенную модель для сканирования на плоскую поверхность.
5. Возьмите устройство 3DQ Planeta3D за рукоять для работы в ручном режиме.

6. Визуально оцените масштаб модели и по средствам перемещения устройства вокруг объекта сканирования добейтесь наиболее подходящих ракурсов для сканирования.
7. Необходимо определить область захвата, при этом весь сканируемый объект должен отображаться зеленым цветом.
8. При предварительном отображении объекта сканирования в рабочей области интерфейса программы, он должен полностью помещаться в область сканирования в виде куба в правой части интерфейса программы.
9. Для начала сканирования нажмите на кнопку «Начать 3D сканирование».
10. После запуска устройство начнет процесс сканирования, ваша задача полностью отсканировать объект со всех сторон.
11. Проведите сканирования с учетом рекомендаций, изложенных в теоретической части.
12. После завершения сканирования нажмите кнопку «Завершить 3D сканирование». В этот момент программа начнет расчет конечной модели.
13. Сохраните трехмерную модель в одном из предложенных форматов и оцените ее качество.

Содержание отчета по работе

В качестве отчета продемонстрируйте учителю результат сканирования.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Подготовьте устройство 3DQ Planeta3D к работе, запустите программное обеспечение Planeta3D.
2. Откройте «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
3. Изложите материал раздела «Краткие теоретические сведения» перед учениками и ответьте на вопросы.
4. Озвучьте задание лабораторной работы.
5. После завершения работы примите ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить итоговую модель на предмет ошибок в области соответствия формам исходного объекта. При их наличии – предоставить возможность исправить модель.

Лабораторная работа №9 «Настройка и работа с программным обеспечением для 3D сканирования Planeta 3D»

Цель

Изучение принципов работы, настройки и функциональных возможностей программного обеспечения Planeta 3D и сканирования объектов в фиксированном режиме.

Краткие теоретические сведения

В предыдущей работе мы познакомились с устройством для сканирования Planeta 3D, а теперь необходимо изучить интерфейс, принципы настройки и работы с программным обеспечением Planeta 3D.

Произведите предварительную подготовку сканера по материалам предыдущей лабораторной работы и запустите ПО Planeta 3D.

После запуска для начала работы необходимо выбрать сенсор, подключенный к устройству. По умолчанию, при запуске программы сенсор подключается автоматически, если это не произошло, щелкните по вкладке «Сенсор» и нажмите кнопку «Выбрать сенсор». Аналогично для подключения сенсоров к программе служит команда Файл-Открыть сенсор.

Изначально возможно выбрать 2 типа сканирования: с текстурой или без текстуры. При этом с текстурой объект будет выглядеть полноцветным, без текстуры объект будет выглядеть однотонным.

Если объект сканируется для последующей 3D печати нет необходимости использовать текстуру.

Далее выбираем тип и размер зоны сканирования из предложенных вариантов или можно настроить параметр вручную в разделе «Размер зоны сканирования».

Возможны два типа зоны сканирования: Прямоугольная и Цилиндрическая. При выборе прямоугольной зоны сканирования необходимо задать ширину, глубину и высоту области либо воспользоваться одним из шаблонов. При выборе

цилиндрической зоны сканирования необходимо указать диаметр области и высоту, либо воспользоваться одним из шаблонов.

По умолчанию для сканирования объектов доступны следующие зоны сканирования Рисунок 1.

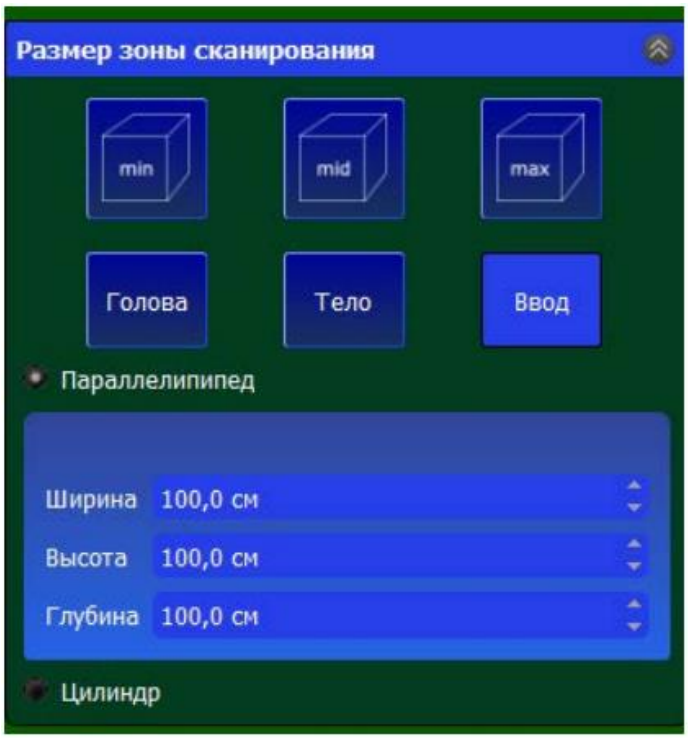


Рисунок 2: Размер зоны сканирования

Минимальный (min), мм	Средний (mid), мм	Максимальный (max), мм
100x100x133	250x250x30	390x390x520

Для сканирования крупногабаритных объектов можно выбрать следующие шаблоны:

Голова, мм	Тело, мм	Произвольный, мм
400x400x400	700x700x2000	

В разделе «Представление зоны сканирования» доступны следующие варианты:

Плотное – в этом режиме память выделяется сразу на все воксели зоны сканирования вне зависимости от того пустые они или нет

Хешированное – в этом режиме память выделяется только на не пустые воксели зоны. При этом разрешение вокселя задается в ручную (во вкладке разрешение зоны сканирования) В данном режиме, кроме ограниченных зон сканирования, доступен неограниченный режим, в котором область сканирования ограничивается областью видимости сенсора. Данный режим позволяет получить повышенную точность сканирования за счет оптимизации ресурсов компьютера.

В разделе «Разрешение зоны сканирования» при выборе режима Плотное можно настроить разрешение сканирования, но чем больше разрешение, тем больше необходимо графической памяти для сканирования. Здесь Вы также можете выбрать значение разрешения из предложенных вариантов или настроить вручную, с помощью ползунка Рисунок 2.

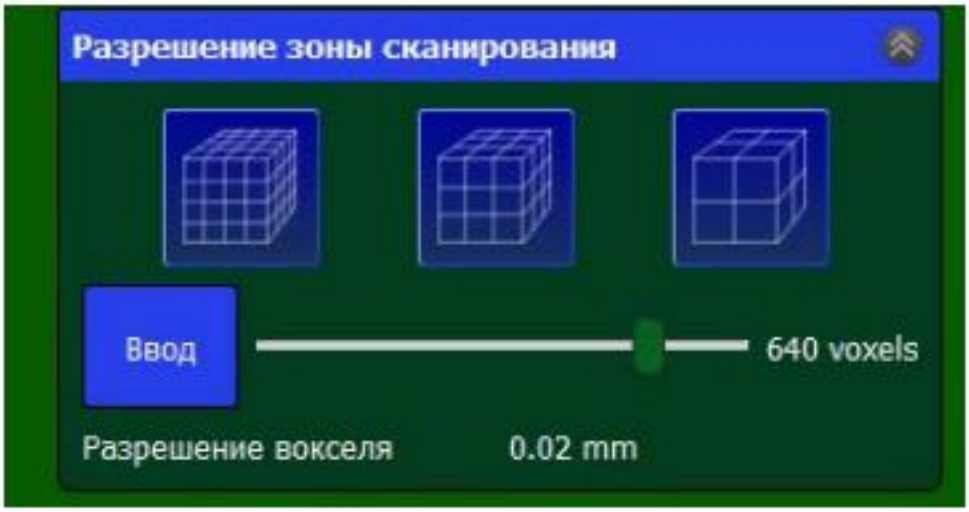


Рисунок 2: Разрешение зоны сканирования

В разделе «Расположение зоны сканирования» настраивается положение зоны сканирования в пространстве. Для изменения объема сканирования Вы можете ввести значения с помощью клавиатуры или прокрутки колеса мыши. Выбираем режим сканирования с текстурой – «Цвет» (по умолчанию) или «Нет цвета» (при этом уменьшается количество требуемой графической памяти) Рисунок 3.

Рисунок 3: Расположение зоны сканирования

Кнопка «Перемещение» служит для перемещения зоны сканирования. При нажатии на нее, в 3D визуализации появляется манипулятор, с помощью которого можно переместить зону сканирования. Для этого щелкните левой кнопкой мыши на манипулятор и перетащите зону сканирования.

Кнопка «Вращение» служит для вращения зоны сканирования.

С её помощью можно выровнять зону сканирования по маркеру, маркером, который идет в комплекте поставки. Для отмены перемещений области сканирования служит кнопка «Сброс». Для сохранения параметров области сканирования и настроек сенсоров необходимо сохранить настройки при помощи команды «Сохранить рабочее пространство» в меню «Файл». Для открытия сохраненных настроек служит команда «Открыть рабочее пространство».

При сохранении рабочего пространства сохраняются следующие параметры:

1. Положение области сканирования;
2. Настройки сенсора;
3. Ориентация сенсора.

В меню «Таймер» можно задать время задержки 3D сканирования и продолжительность 3D сканирования в секундах или же в количестве кадров.

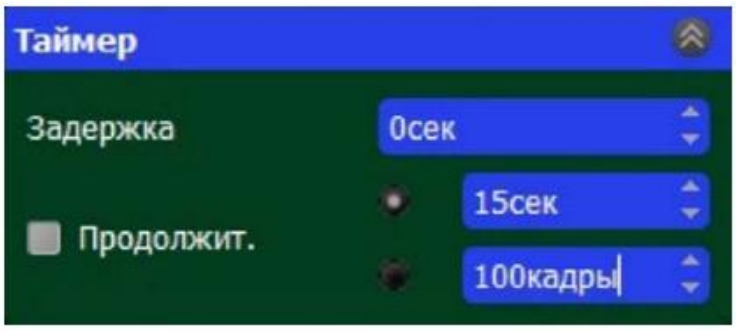


Рисунок 4: Таймер задержки сканирования

После размещения объекта сканирования в области сканирования, можно начать работу путем нажатия на кнопку «Начать 3D сканирование». Во время сканирования вы увидите процесс работы на экране. Индикатор состояния, находиться в верхнем левом углу и показывает статус сканирования. Если все идет хорошо, то он будет гореть зеленым. Красным цветом индикатор загорится при ошибке сканирования.

Во время сканирования частота кадров сенсора и реконструкции отображается слева. Частота кадров реконструкции должна быть больше 10 кадров в секунду, чтобы сканирование проходило хорошо. Чем ниже частота кадров, тем медленнее нужно передвигать сенсор. В противном случае объект будет потерян, о чем сообщит красный индикатор. Достижимая частота кадров зависит от разрешения области сканирования и от вашего GPU. Чтобы получить более высокую частоту кадров попробуйте уменьшить разрешение области сканирования. После того, как Вас устроит результат сканирования, нажмите на кнопку «Завершить 3D сканирование».

Для получения качественной 3D модели пользуйтесь рекомендациями из предыдущей лабораторной работы.

Задание

Произвести настройку программного обеспечения. С помощью сканера 3DQ Planeta3D произведите сканирование простого объекта в фиксированном режиме.

Порядок выполнения работы

1. Изучите теоретические сведения перед выполнением работы.
2. Произведите подготовку устройств к сканированию.
3. Запустите программное обеспечение Planeta 3D.
4. Изучите параметры и возможности работы с ПО Planeta 3D изложенные в теоретической части.
5. Возьмите несколько объектов разного размера для сканирования
6. Определите параметры и произведите настройку ПО.
7. Отсканируйте два предложенных объекта с различными параметрами.
8. Сохраните получившиеся трехмерные модели в одном из предложенных форматов.
9. Сравните получившиеся 3D модели и опишите результата сканирования.

Содержание отчета по работе

В качестве отчета продемонстрируйте учителю результаты сканирования и отчет о результатах сравнения получившихся моделей.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Подготовьте устройство 3DQ Planeta3D к работе, проверьте работоспособность программного обеспечения Planeta3D.
2. Откройте «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
3. Изложите материал раздела «Краткие теоретические сведения» перед учениками и ответьте на вопросы.
4. Озвучьте задание лабораторной работы.
5. После завершения работы примите ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучите итоговые модели на предмет ошибок в области соответствия формам исходного объекта. При их наличии – предоставить возможность исправить модель.
2. Проверьте отчеты и дайте рекомендации в случае ошибочных суждений.

Лабораторная работа №10 «Изучение принципов работы 3D сканера Planeta 3D и сканирование объектов в фиксированном режиме»

Цель

Изучение принципов работы, настройки и функциональных возможностей устройства для сканирования объектов Planeta 3D и сканирования объектов в фиксированном режиме.

Краткие теоретические сведения

Как описывалось ранее 3D сканер Planeta 3D позволяет сканировать объекты в двух режимах работы: ручной режим сканирования и фиксированный. Разберем подробно фиксированный режим сканирования объектов.

Сканирование объектов в фиксированном режиме предполагает использование штатива, на который устанавливается устройство Planeta 3D и поворотного стола.

Для подготовки к сканированию необходимо установить поворотный стол на плоскую поверхность, подключить его к питанию от сети и подключить по usb кабелю к компьютеру.

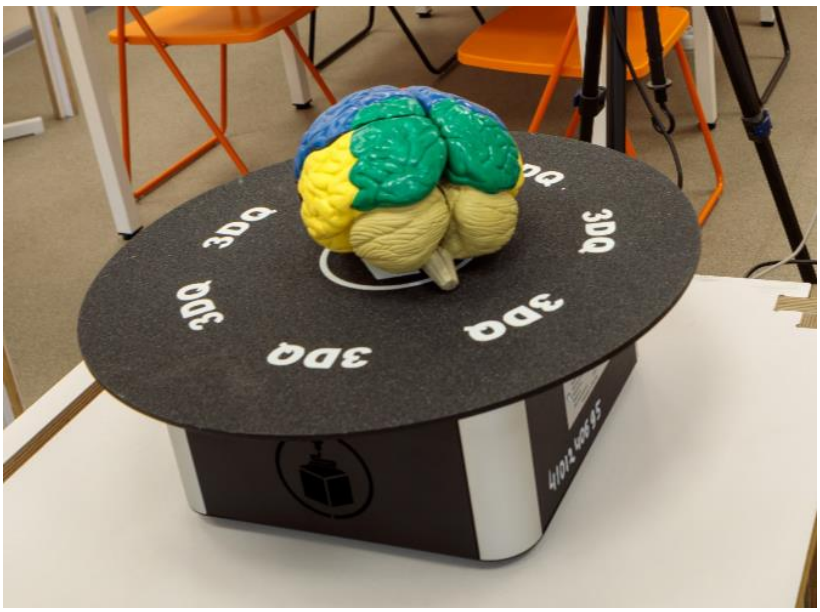


Рисунок 3: Поворотный стол для сканирования объектов

Поворотный стол управляется через компьютер по средствам программного обеспечения и выдерживает нагрузку до 45 кг. Площадка стола вращается и тем самым позволяет произвести сканирование объекта с фиксированным положением сканера.

Далее необходимо зафиксировать устройство на штативе как показано на рисунке 2.

После установки сканера на штатив необходимо выполнить настройку штатива.

Прежде чем начать настройку штатива, определитесь, где он будет размещен. Всегда устанавливайте штатив строго горизонтально – для этого следите за показаниями пузырьковых уровней на нём. Это необходимо для того, чтобы нагрузка приходилась на центр штатива и равномерно распределялась на каждую из трех ног.

В первую очередь выдвиньте толстые секции ног штатива. Это делает штатив более устойчивым. Полностью выдвигайте секции ног. У каждого

Рисунок 2: Установка 3D сканера Planeta 3D на штатив

штатива есть фиксированная точка, дальше которой секция ноги не выдвигается. Устанавливая штатив, всегда проверяйте полностью ли вы выдвинули секцию

ноги. Иначе в самый неподходящий момент неполностью выдвинутая секция может сместиться, из-за чего нарушится равновесие штатива и он может опрокинуться, подвергнув сканер опасности разбиться.

Разместите штатив так, чтобы одна из его ног смотрела в сторону объекта сканирования. Для более удобной настройки устройства на штативе.

Проверьте надежность установки сканера. Убедитесь, что сканер надежно закреплен на штативе. Установив оборудование, попробуйте пошевелить его руками, не должно быть никаких заметных люфтов и вибраций в местах крепления устройства.

Штатив необходимо расположить таким образом, чтобы сканер смог полностью захватить объект, находящийся на площадке поворотного стола.

После установки оборудования запустите ПО Planeta 3D и проведите подстройку сканера на сканируемый объект Рисунок 3.

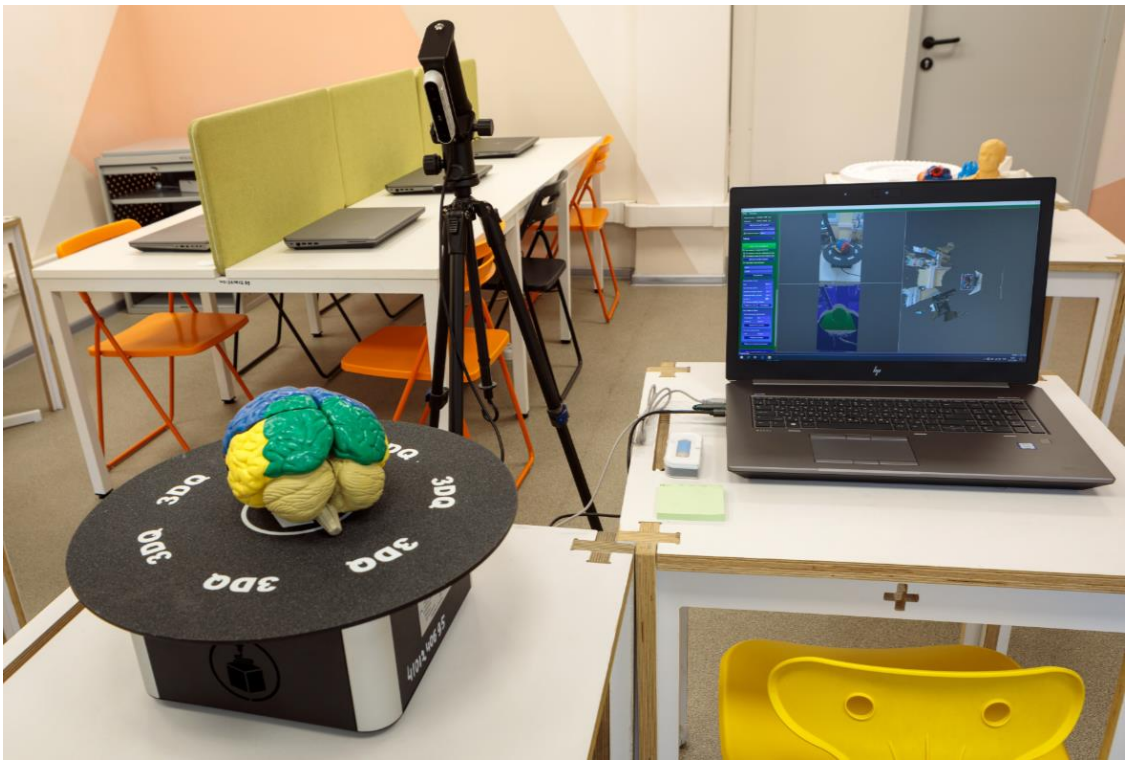


Рисунок 3: Настройка 3D сканера на объект

Обратите внимание на то, чтобы объект полностью находился в рамках рабочей области. И все части объекта были окрашены в зеленый цвет.

Для сканирования модели в фиксированном режиме работы необходимо произвести следующие действия:

1. Поместить сканируемый объект на поворотный столик.
2. Запустить программу 3DQ Route для управления поворотным столом.
3. Запустить программу Planeta3D для работы со сканером.
4. Произвести корректировку сканирующего устройства для получения более точной модели.
5. Нажать на кнопку «начать 3D сканирование».
6. Включить поворотный стол.
7. После полного оборота поворотного стола нажать на кнопку «завершить 3D сканирование».

Для получения качественной 3D модели пользуйтесь рекомендациями из предыдущей лабораторной работы.

Задание

Произведите настройку оборудования для сканирования в фиксированном режиме. С помощью сканера 3DQ Planeta3D произведите сканирование простого объекта в фиксированном режиме.

Порядок выполнения работы

1. Изучите теоретические сведения перед выполнением работы.
2. Произведите подготовку устройств к сканированию.
3. Запустите программное обеспечение Planeta 3D.
4. Запустить программу 3DQ Route для управления поворотным столом.
5. Установите предложенную модель для сканирования на поворотный стол.
6. Определите область захвата, при этом весь сканируемый объект должен отображаться зеленым цветом.
7. При предварительном отображении объекта сканирования в рабочей области интерфейса программы, он должен полностью помещаться в область сканирования в виде куба в правой части интерфейса программы.
8. Для начала сканирования нажмите на кнопку «Начать 3D сканирование».
9. После запуска сканирование запустите поворотный стол.

10. После полного оборота поворотного стола нажать на кнопку «завершить 3D сканирование». В этот момент программа начнет расчет конечной модели.
11. Сохраните трехмерную модель в одном из предложенных форматов и оцените ее качество.

Содержание отчета по работе

В качестве отчета продемонстрируйте учителю результат сканирования.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Подготовьте устройство 3DQ Planeta3D к работе, проверьте работоспособность программного обеспечения Planeta3D.
2. Подготовьте поворотный стол к работе и проверьте работоспособность программного обеспечения.
3. Откройте «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
4. Изложите материал раздела «Краткие теоретические сведения» перед учениками и ответьте на вопросы.
5. Озвучьте задание лабораторной работы.
6. После завершения работы примите ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить итоговую модель на предмет ошибок в области соответствия формам исходного объекта. При их наличии – предоставить возможность исправить модель.

Лабораторная работа №11 «Изучение принципов работы 3D сканера Planeta 3D и сканирование объектов в комбинированном режиме»

Цель

Изучение принципов работы, настройки и функциональных возможностей устройства для сканирования объектов Planeta 3D и сканирование объектов в комбинированном режиме работы.

Краткие теоретические сведения

Ранее были рассмотрены возможности сканирования объектов в ручном и фиксированном режимах. Для получения более качественных 3D моделей необходимо скомбинировать эти режимы сканирования.

В качестве первого режима используйте фиксированный. Произведите полную предварительную подготовку устройства к работе в данном режиме Рисунок 1.

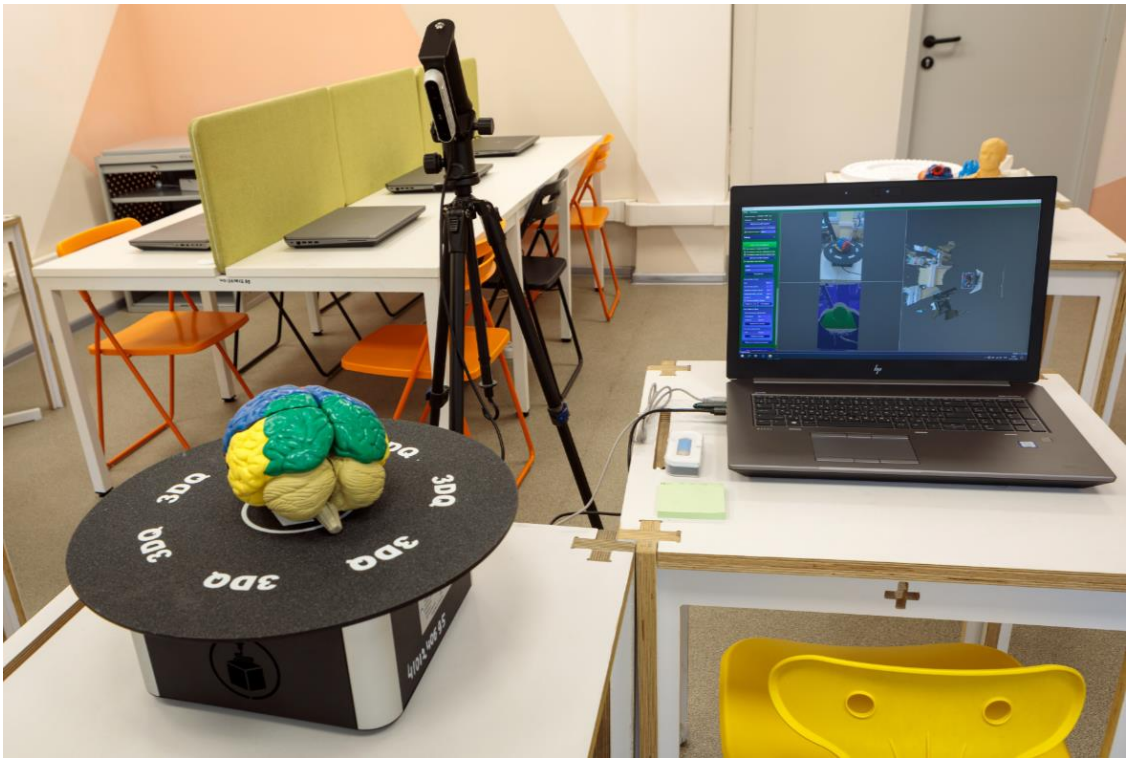


Рисунок 1 Подготовка 3D сканера для работы в фиксированном режиме

После установки оборудования запустите ПО Planeta 3D, произведите настройку ПО и подстройку сканера на сканируемый объект.

Далее будем производить действия как и при работе в фиксированном режиме:

1. Поместите сканируемый объект на поворотный столик.
2. Запустить программу 3DQ Route для управления поворотным столом.
3. Запустить программу Planeta3D для работы со сканером.
4. Произведите корректировку сканирующего устройства для получения более точной модели.
5. Настройте ПО под характеристики исследуемого объекта.
6. Нажмите на кнопку «начать 3D сканирование».
7. Включите поворотный стол.
8. После полного оборота поворотного стола перейдем к ручному режиму сканирования.

Дело в том, что при сканировании объекта в фиксированном режиме не все области объекта могут быть корректно отсканированы. Именно для улучшения качества модели мы можем дополнительно отсканировать области в ручном режиме. Для этого необходимо выполнить следующие действия:

1. Визуально оцените непроработанные области модели через интерфейс.
2. Снимите устройство со штатива и возьмите его за рукоять.
3. Произведите сканирование непроработанных областей объекта в ручном режиме.
4. Убедитесь, что все области объекта отсканированы.
5. Завершите сканирование нажав на кнопку «завершить 3D сканирование».

Важно отметить, что в процессе сканирования и перехода между фиксированным и ручным режимами работы необходимо не потерять захват объекта, для этого обращайте внимание на программный интерфейс и отслеживайте положение устройства.

Задание

Произведите настройку оборудования для сканирования в фиксированном режиме. С помощью сканера 3DQ Planeta3D произведите сканирование простого объекта в комбинированном режиме. Сравните и опишите в отчете результаты сканирования моделей в фиксированном, ручном и комбинированном режимах работы.

Порядок выполнения работы

1. Изучите теоретические сведения перед выполнением работы.
2. Произведите подготовку устройств к сканированию.
3. Запустите программное обеспечение Planeta 3D.
4. Запустить программу 3DQ Route для управления поворотным столом.
5. Установите предложенную модель для сканирования на поворотный стол.
6. Определите область захвата и произведите настройку ПО.
7. Для начала сканирования нажмите на кнопку «Начать 3D сканирование».
8. После запуска сканирование запустите поворотный стол.
9. После полного оборота поворотного стола перейдите к ручному сканированию.
10. После доработки модели в ручном режиме нажать на кнопку «завершить 3D сканирование». В этот момент программа начнет расчет конечной модели.
11. Сохраните трехмерную модель в одном из предложенных форматов и оцените ее качество.

Содержание отчета по работе

Продемонстрируйте учителю результат сканирования. В отчете сравните и опишите результаты сканирования моделей в фиксированном, ручном и комбинированном режимах работы, которые были получены в результате выполнения предыдущих лабораторных работ.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Подготовьте устройство 3DQ Planeta3D к работе, проверьте работоспособность программного обеспечения Planeta3D.

2. Подготовьте поворотный стол к работе и проверьте работоспособность программного обеспечения.
3. Откройте «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
4. Изложите материал раздела «Краткие теоретические сведения» перед учениками и ответьте на вопросы.
5. Озвучьте задание лабораторной работы.
6. После завершения работы примите ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучите итоговые модели на предмет ошибок в области соответствия формам исходного объекта. При их наличии – предоставить возможность исправить модель.
2. Проверьте отчеты и дайте рекомендации в случае ошибочных суждений.

Лабораторная работа №12 «Сканирование объектов сложной формы с помощью 3DQ Planeta3D»

Цель

Изучение правил настройки положения сканера при сканировании объектов сложной формы с помощью 3DQ Planeta3D.

Краткие теоретические сведения

Ранее были рассмотрены возможности сканирования простых объектов в фиксированном и комбинированном режимах работы. Теперь рассмотрим возможности и особенности сканирования объектов сложной формы и структуры, применяя фиксированный режим сканирования.

Когда мы говорим о сканере с подобной конструкцией, следует понимать, что положение сканера на штативе и закрепление его относительно поворотной платформы в пределах одного скана должно оставаться неизменным. Да, в настоящее время, мы уже имеем доступ к сканерам, которые могут самостоятельно отслеживать свое положение относительно объекта, но они обладают двумя достаточно весомыми недостатками. Во-первых, они требуют значительной вычислительной мощности от машины, на которой развернуто программное обеспечение, либо, если все необходимые вычисления проходят внутри устройства, обладают очень высокой стоимостью. И, во-вторых, обеспечивают куда меньшую точность итоговой модели, чем сканеры, закрепленные на неподвижном основании.

Таким образом, при применении сканера с поворотным столом, речь идет о необходимости получить модель высокой точности. Но, исходя из конструкции сканера, мы не можем обеспечить полного покрытия объекта сложной формы, что приводит к необходимости постобработки модели. Пример объекта сложной формы представлен на рисунке 1.

Рисунок 1 Объект сложной формы

Возьмем объект, приведенный на рисунке выше. Он обладает сложной формой, так как имеет углубление в форме чаши сверху, полость внутри нижней части и изгиб между ними. Чтобы определить наиболее выгодный ракурс для сканирования, представим абстрактно, как будет выглядеть сечение (Рисунок 2).

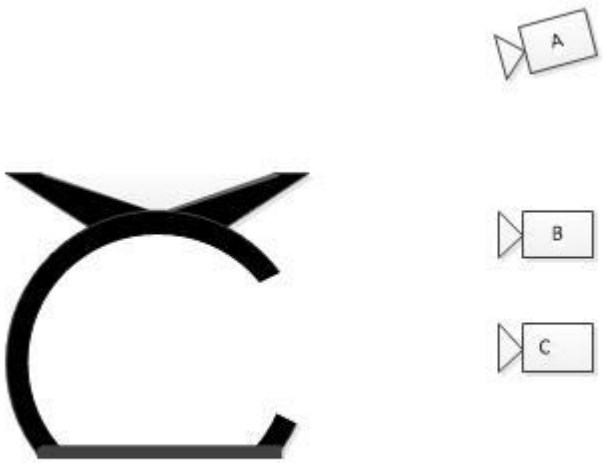


Рисунок 2 Расположение сканера относительно объекта

Теперь давайте рассмотрим основные способы расположения сканера относительно нашего объекта.

1. В положении «А» мы получаем исчерпывающую информацию о форме верхней чаши, но практически полностью игнорируем ту сложную форму, которую имеет внутренность нижней составляющей нашего объекта, а также может появиться неотсканированная область на месте сочленения нижней части объекта с верхней.

2. В положении «В» у нас не возникает проблем с центральной частью нашего объекта и даже присутствует некоторая информация о внутренности нижней части, но она далеко не исчерпывающая и все еще сильно ограничена сверху. Кроме того, полностью неизвестной остается для сканера область чаши сверху. В таких ситуациях сканер закрывает проем в данных прямой плоскостью.

3. В положении «С» наиболее полная информация будет получена о внутреннем устройстве нижней части, но, так же как и в положении «В», абсолютно не будет информации о чаше сверху.

Исходя из изложенного анализа, мы можем заключить, что наиболее невыгодным для нас является положение «В». Но ни одно из двух других состояний не является идеальным. Значит, придется исходить из минимизации усилий на постобработку модели. Для того, чтобы сделать выбор между положениями «А» и «С» рассмотрим наш объект с другого ракурса.

Рисунок 3 Расположение объекта для сканирования

Благодаря такому ракурсу мы можем заключить, что внутренняя форма нашего объекта гораздо сложнее, чем форма чаши сверху, так как для повторения ее с помощью средств моделирования потребуется затратить много усилий. В то время как для восстановления формы верхней чаши просто нужно будет выдавить в объекте сферическое углубление.

Таким образом можно сделать вывод, что для сканирования объектов сложной структуры перед началом сканирования необходимо определить не только положение сканера относительно исследуемого объекта, но и расположить сам объект на поворотном столе таким образом, что бы в дальнейшем минимизировать необходимость постобработки модели.

Задание

С помощью сканера 3DQ Planeta3D и рассчитанного положения штатива создать модель, которая будет наиболее проста для этапа постобработки. Сравните и опишите в отчете особенности построения простых и сложных объектов в фиксированном режиме работы, используя модели прошлых лабораторных работ.

Порядок выполнения работы

1. Изучите теоретические сведения перед выполнением работы.
2. Произведите подготовку устройств к сканированию.
3. Запустите программное обеспечение Planeta 3D.
4. Запустить программу 3DQ Route для управления поворотным столом.
5. Определите наиболее подходящее положение модели для сканирования на поворотном столе.
6. Произведите настройку положения сканера на штативе и самого штатива для получения необходимого ракурса.
7. Определите область захвата и произведите настройку ПО.
8. Для начала сканирования нажмите на кнопку «Начать 3D сканирование».
9. После запуска сканирование запустите поворотный стол.
10. После полного оборота поворотного стола нажмите на кнопку «завершить 3D сканирование». В этот момент программа начнет расчет конечной модели.
11. Сохраните трехмерную модель в одном из предложенных форматов и оцените ее качество.

Содержание отчета по работе

Продемонстрируйте учителю результат сканирования. В отчете сравните и опишите результаты сканирования моделей в фиксированном режиме работы для объектов простой и сложной формы и структуры, которые были получены в результате выполнения текущей и предыдущих лабораторных работ.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Подготовьте устройство 3DQ Planeta3D к работе, проверьте работоспособность программного обеспечения Planeta3D.
2. Подготовьте поворотный стол к работе и проверьте работоспособность программного обеспечения.
3. Откройте «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
4. Изложите материал раздела «Краткие теоретические сведения» перед учениками и ответьте на вопросы.
5. Озвучьте задание лабораторной работы.
6. После завершения работы примите ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучите итоговые модели на предмет ошибок в области соответствия формам исходного объекта. При их наличии – предоставить возможность исправить модель.
2. Проверьте отчеты и дайте рекомендации в случае ошибочных суждений.

Лабораторная работа №13 «Использование дополнительных приспособлений для сканирования объектов»

Цель

Изучение способов фиксации сканируемого объекта на поворотном столе для сканирования с помощью 3DQ Planeta3D.

Краткие теоретические сведения

Как уже упоминалось ранее, сканирование объектов сложной формы на поворотном столе, вынуждает нас выбирать «меньшее из зол» с точки зрения постобработки модели, так как нам необходимо провести сканирование с одного ракурса за один проход по кругу.

Следует внести некоторые коррективы в данное утверждение. Мы можем осуществить несколько сканирований, получить в результате них несколько облаков точек, и потом с помощью программного обеспечения совместить их в единую модель. Почему же данный способ видится худшим решением, чем постобработка из предыдущей лабораторной? Давайте разберемся.

Если мы говорим о трехмерном пространстве, то мы сталкиваемся с так называемыми шестью степенями свободы. Каждое облако точек, полученное в результате сканирования, имеет некий центр, являющийся началом координат этого облака. Если мы захотим совместить несколько облаков точек, то нам нужно будет совместить их центры, то есть убрать сдвиг по трем осям. Почему же тогда шесть степеней свободы? Кроме прямого смещения, выражающегося через координаты (x;y;z), есть еще поворот всего облака относительно каждой из осей. Так вот, для совмещения нескольких сканов необходимо будет выровнять объект по всем этим шести параметрам. Казалось бы, просто начать сканировать с одного и того же положения поворотного стола, не трогая объект на нем, и тогда необходимо будет сместить все облако только по оси z. К сожалению, все немного сложнее.

Ни одно самое точное оборудование не обеспечит выставление своих шаговых двигателей в начальное положение с точностью до секунд. А при

сканировании объекта, границы которого уходят хотя бы на несколько сантиметров от центра вращения, данный сдвиг по градусам приведет к достаточно серьезным помехам на изображении. Не говоря уже о том, что выставить объект четко в центре стола – задача далеко не тривиальная. Можно воспользоваться специализированным ПО для сборки единой модели, но как уже было сказано, оно достаточно тяжелое в плане вычислений. При все этом, предмет на рисунке 1 все равно приведет к ряду сложностей.

Рисунок 1 Объект сложной формы с тонкой структурой

Он обладает крайне тонкой структурой, соответственно, если мы хотим получить не только модель лицевой формы, а полное трехмерное представление этой пластиковой формы, то необходимо будет отсканировать ее с двух сторон. При склейке этих двух сканов с помощью ПО, будет произведено их объединение

с практически полным игнорированием толщины. Если попытаться исправить этот момент постфактум, увеличив толщину всей поверхности, то итоговая модель получиться худшего качества, чем оригинал, так как будет потеряна значительная часть рельефа. Ручная регулировка сопоставления тоже затруднена из-за толщины.

Выходом из ситуации является обеспечение полного сканирования за один проход с помощью поворотного стола. Это даст нам очень высокую точность сканирования, что позволит восстановить толщину детали. Расположить деталь на поворотном столе можно способом, показанным на рисунке 2.

Рисунок 2 Расположение объекта сканирования

Используя простой пластилин мы зафиксировали объект в положении наиболее подходящем для сканирования. Да, таким образом мы заработали необходимость в постобработке облака точек в виде удаления паразитных элементов, но это делается гораздо проще, чем совмещение, о котором шла речь выше. Кроме того, здесь нам на руку играет факт того, что все пробелы (которые останутся у нас после удаления нашей поддерживающей конструкции) ПО склеивает по крайним точкам.

Помимо использования поддерживающих упоров внизу, можно воспользоваться конструкциями которые наподобие стрелы крана будут держать объект на весу для лучшего доступа к нему сканера. Недостатком подобного закрепления является факт необходимости стабилизации объекта в таком положении, так как покачивания объекта в пространстве могут создать помехи в итоговой модели.

Для повышения точности сканирования однотонных или бесцветных объектов рекомендуется проецировать на сканируемый объект узор (например в виде шахматного поля) Рисунок 3.

Рисунок 3 Сканирование однотонного объекта

Для этого нужно сделать следующее:

1. Установить проектор на настольный штатив и направить проекцию на объект.
2. Подключить проектор к компьютеру при помощи HDMI кабеля.
3. В настройках Windows подключить второй экран.
4. Открыть файл с нужным узором и перенести его на второй экран. Рекомендуется использовать команду CTRL+WIN+X.
5. Запустить программу Planeta3D и приступить к 3D сканированию.

Перечисленные в данной лабораторной работе приемы могут показаться достаточно забавными и нелепыми, но для человека, который не владеет на профессиональном уровне 3D-моделированием, удалить паразитные объекты со скана гораздо проще, чем дорабатывать модель в трехмерном редакторе.

Задание

С помощью сканера 3DQ Planeta3D и дополнительных приспособлений создать трехмерную модель предложенного объекта.

Порядок выполнения работы

1. Изучите теоретические сведения перед выполнением работы.
2. Произведите подготовку устройств к сканированию.
3. Запустите программное обеспечение Planeta 3D.
4. Запустить программу 3DQ Route для управления поворотным столом.
5. С помощью дополнительных приспособлений установите объект для сканирования на поворотном столе.
6. Произведите настройку положения сканера на штативе и самого штатива для получения необходимого ракурса.
7. Определите область захвата и произведите настройку ПО.
8. Для начала сканирования нажмите на кнопку «Начать 3D сканирование».
9. После запуска сканирование запустите поворотный стол.
10. После полного оборота поворотного стола нажмите на кнопку «завершить 3D сканирование». В этот момент программа начнет расчет конечной модели.

11. Сохраните трехмерную модель в одном из предложенных форматов и оцените ее качество.

Содержание отчета по работе

Продемонстрируйте учителю результат сканирования. В отчете сравните и опишите результаты сканирования моделей в фиксированном режиме работы для объектов сложной формы и структуры, которые были получены в результате выполнения текущей и предыдущей лабораторных работ.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Подготовьте устройство 3DQ Planeta3D к работе, проверьте работоспособность программного обеспечения Planeta3D.
2. Подготовьте поворотный стол к работе и проверьте работоспособность программного обеспечения.
3. Откройте «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
4. Изложите материал раздела «Краткие теоретические сведения» перед учениками и ответьте на вопросы.
5. Озвучьте задание лабораторной работы.
6. После завершения работы примите ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучите итоговые модели на предмет ошибок в области соответствия формам исходного объекта. При их наличии – предоставить возможность исправить модель.
2. Проверьте отчеты и дайте рекомендации в случае ошибочных суждений.

Лабораторная работа №14 «Постобработка результатов сканирования»

Цель

Изучение возможностей постобработки в программном обеспечении Planeta3D.

Краткие теоретические сведения

Для дальнейшей работы с получившейся в результате сканирования 3D моделью, необходима её постобработка. Постобработку позволяет осуществить программа Planeta3D. Далее рассмотрим возможности постобработки модели в предложенном ПО.

Получившаяся в результате сканирования модель мы можем сохранить в оригинале в виде облака точек, в этом случае дальнейшая печать модели на принтере не представится возможной, а чаще всего модель нам необходима именно для печати на принтере и в этом случае её следует сохранять в виде твердой модели. Но для подготовки модели к сохранению требуется её постобработка.

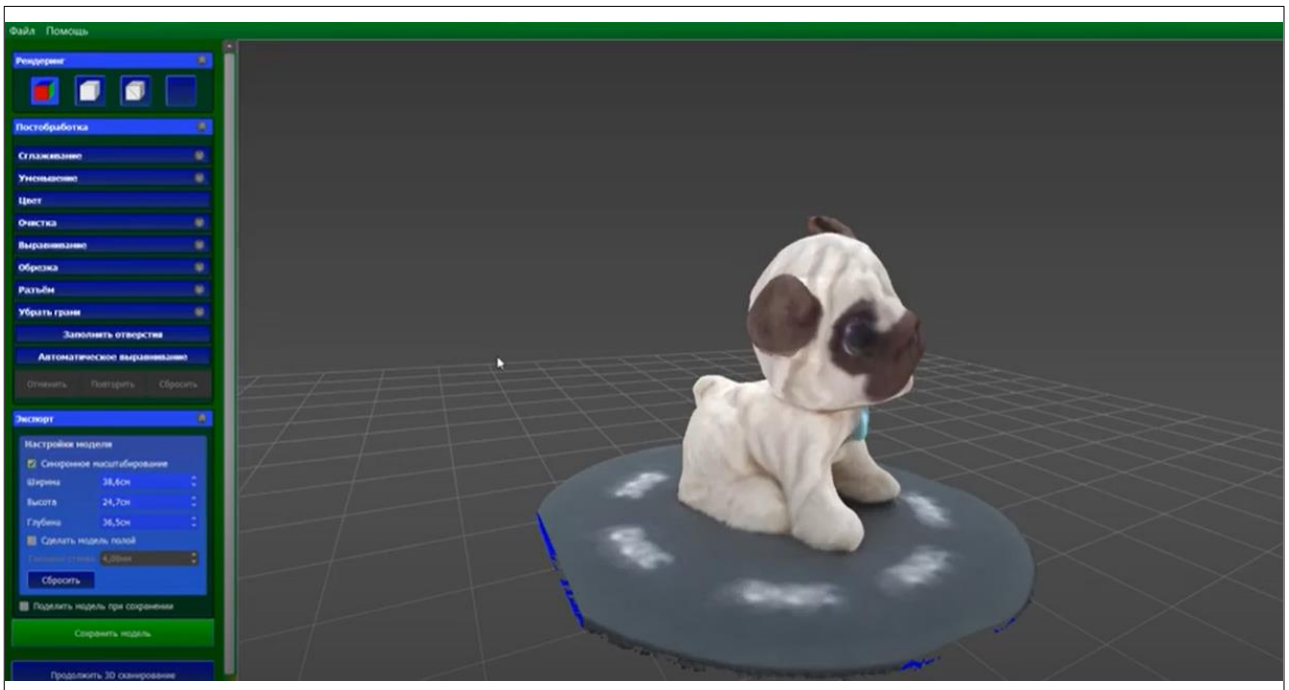


Рисунок 1 Модель до обработки

После сканирования модели необходимо воспользоваться кнопкой «заполнить отверстия». В результате модель становится сплошной и без отверстий.

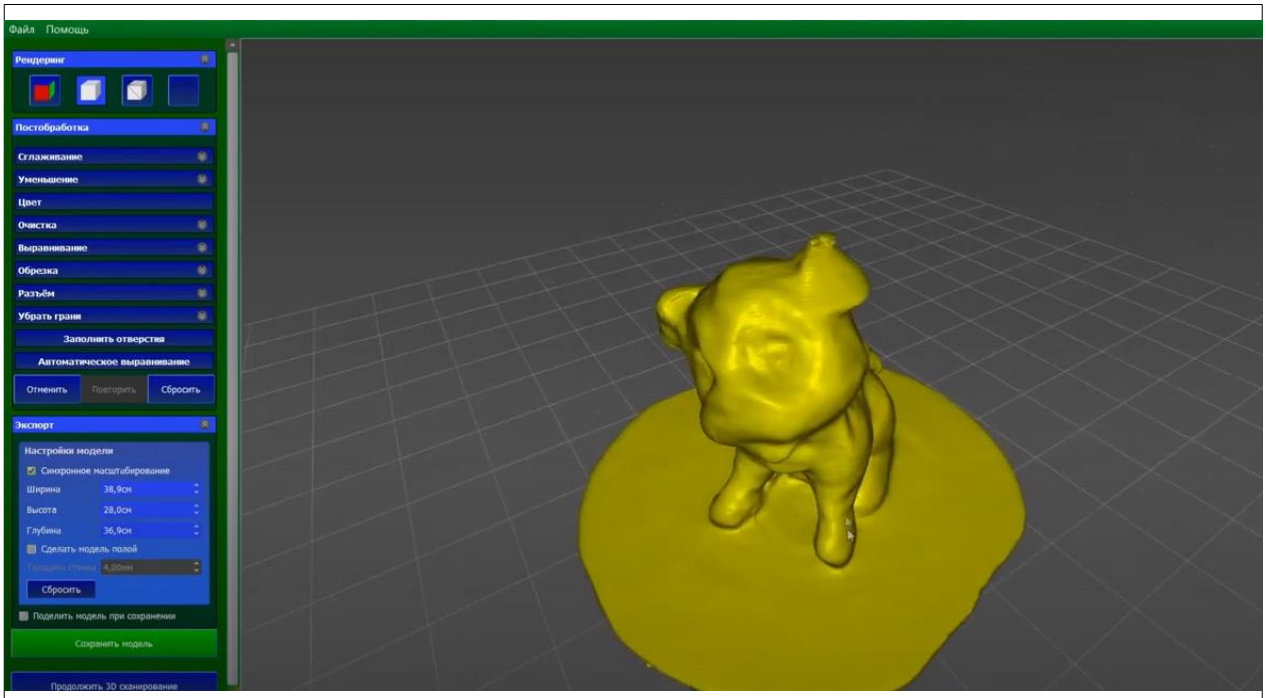


Рисунок 2 Заполнение отверстий модели

При выключенной текстуре видно, что все грани соединены и отсутствуют все отверстия.

На получившейся модели видно, что она по прежнему не пригодна для печати поскольку имеются лишние части и фрагменты. Для более качественной подготовки модели вернемся на предыдущий шаг и выполним еще несколько команд предобработки.

При сканировании могут образовываться различные фрагменты не принадлежащие объекту сканирования. Для их устранения с модели можно воспользоваться ручной или автоматической очисткой. Воспользуемся кнопкой «выбрать наибольший компонент», при этом на изображении красным будут выделены лишние части модели. После чего их необходимо удалить, нажав на кнопку «очистить» рисунок 3.

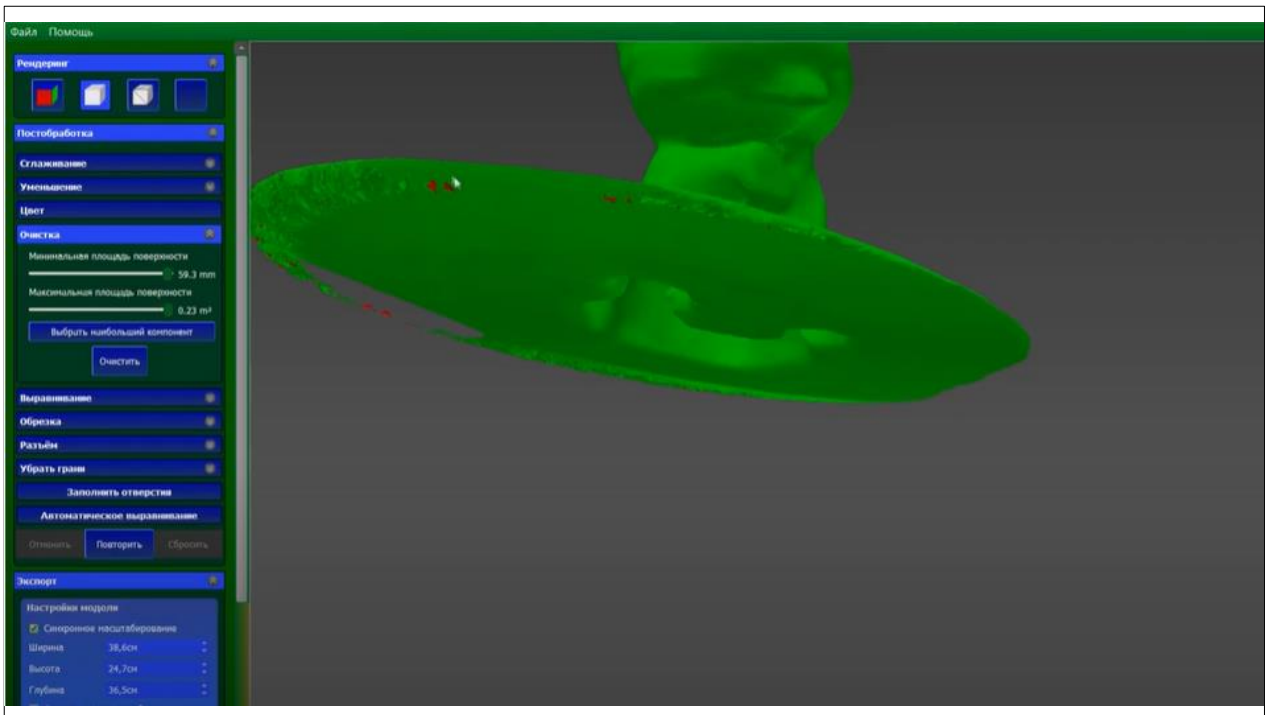


Рисунок 3 Очистка лишних компонентов

Далее переходим к обрезке лишних частей модели. Для этого воспользуемся полем меню обрезка Рисунок 4.

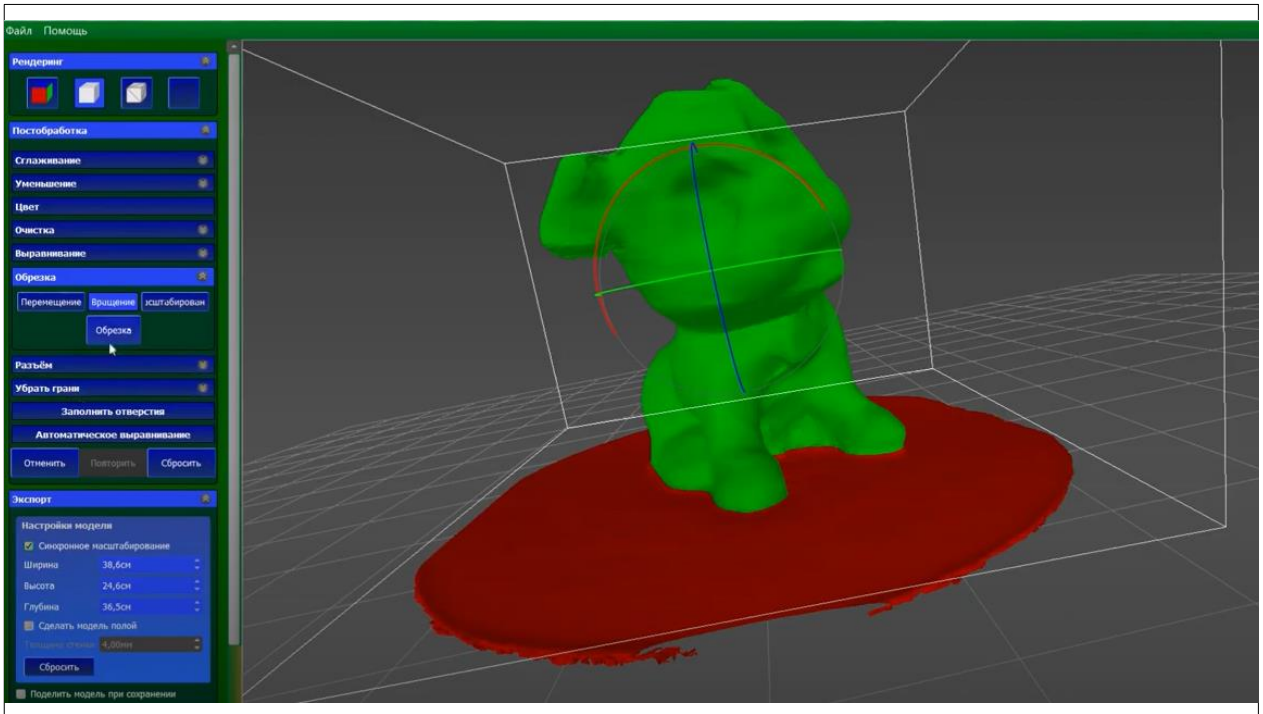


Рисунок 4 Обрезка лишних частей модели

Режим обрезки модели позволяет обрезать модель посредством габаритного контейнера. Можно перемещать, вращать и масштабировать габаритный контейнер. Все части модели, находящиеся за пределами габаритного контейнера

(окрашены в красный цвет), будут удалены. Для перемещения габаритного контейнера необходимо нажимать и перетаскивать манипуляторы в 3D виде. Для удаления объектов внутри габаритного контейнера включите параметр «Очистить внутри коробки».

Обрежем лишние части модели нажав на кнопку «обрезка».

Каждая модель состоит из полигонов их можно рассмотреть выбрав соответствующий режим просмотра без текстуры и приблизи модель Рисунок 5.

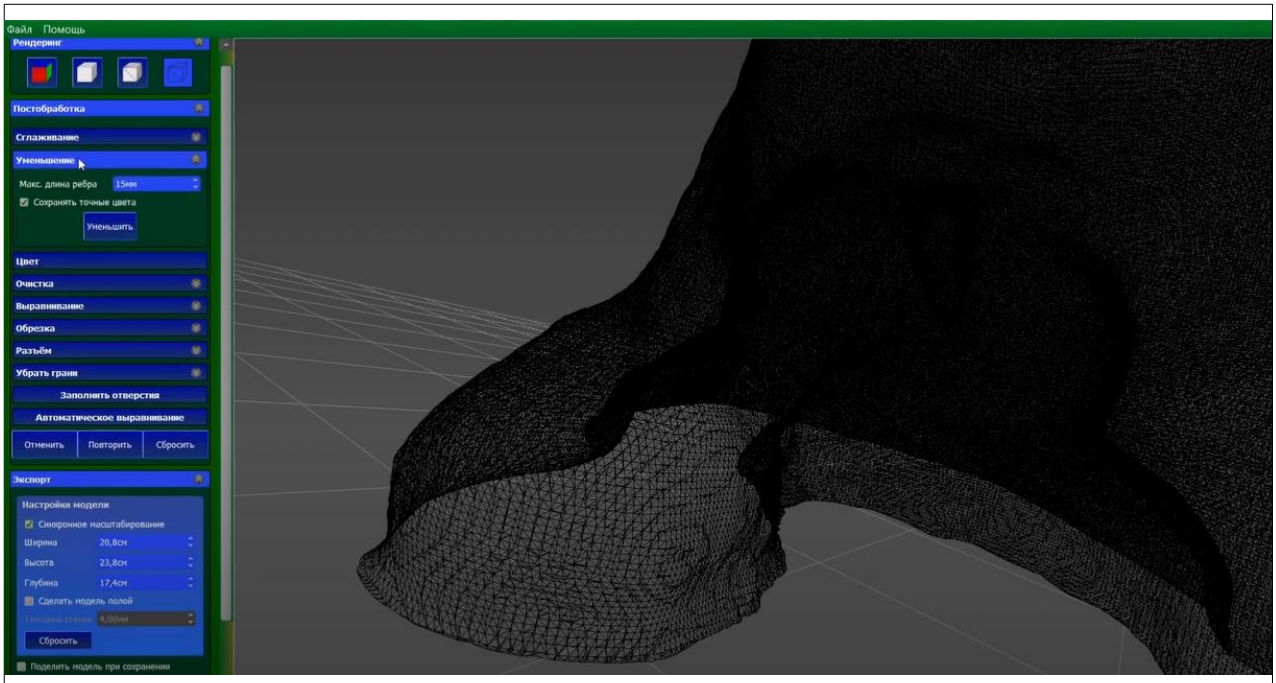


Рисунок 5 Изменение полигональности модели

Как видно из рисунка сканер строит высокополигональную модель, что может затруднить её последующую обработку на компьютерах малой мощности. При этом необходимость в большом количестве полигонов не всегда целесообразна. Для изменения этого параметра можно воспользоваться пунктом меню «Уменьшение» и увеличить размер полигона.

Воспользовавшись этой опцией после пересчета модели можно получить модель с большим размером полигонов.

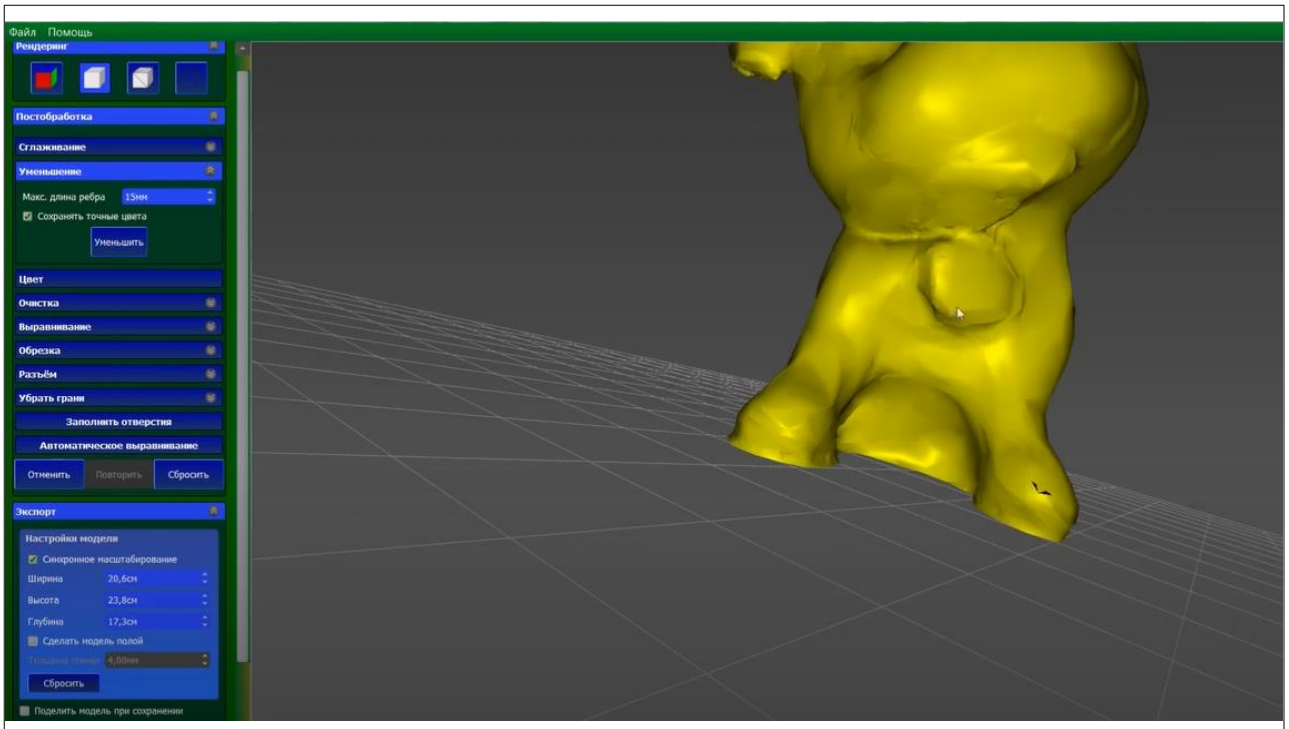


Рисунок 6 Упрощенная модель

Как видно из получившейся модели, она теряет детализацию. Уменьшение размера модели также уменьшит разрешения текстуры, так как в этот момент цвета сохраняются в вершинах треугольников. Поэтому данную функцию можно применять только в случае необходимости.

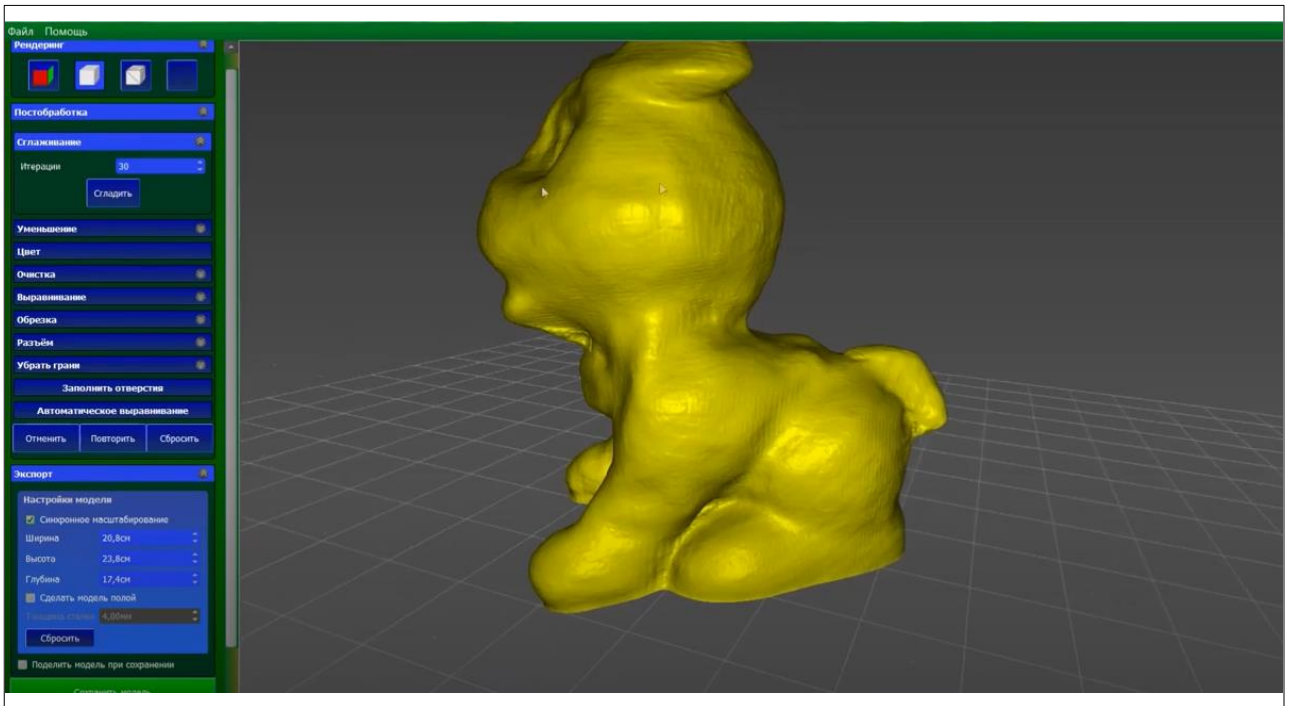


Рисунок 7 Сглаживание модели

Как видно из рисунка 7, модель имеет достаточное количество неровностей для их устранения воспользуемся режимом сглаживания.

Режим сглаживания модели - позволяет сгладить модель, используя заданное число проходов. Для сглаживания необходимо выбрать соответствующий пункт меню и выбрав количество итераций сглаживания нажать на кнопку «сгладить». Сглаживание можно производить несколько раз до достижения необходимого эффекта.

Как говорилось ранее для 3D печати необходима закрытая сетка – сплошная модель. Но при сканировании в модели могут возникать отверстия, которые необходимо устранять.

Режим устранения отверстий в модели – заполняет все отверстия в модели. Результат выполнения команды представлен на рисунке 8.

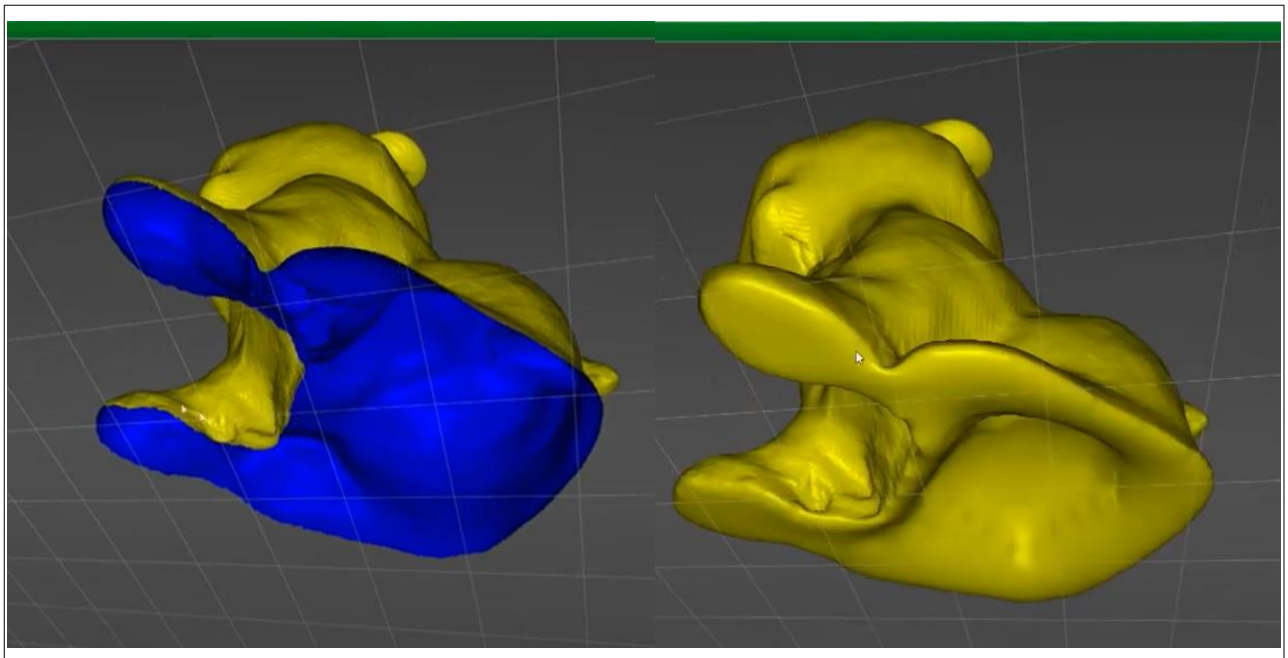


Рисунок 8 Устранение отверстий

После обработки видим, что грани исчезли и модель приобрела твердую форму.

Режим создания постамента – создает постамент. Пользователь может задать высоту постамента, диаметр постамента и цвет постамента Рисунок 9.

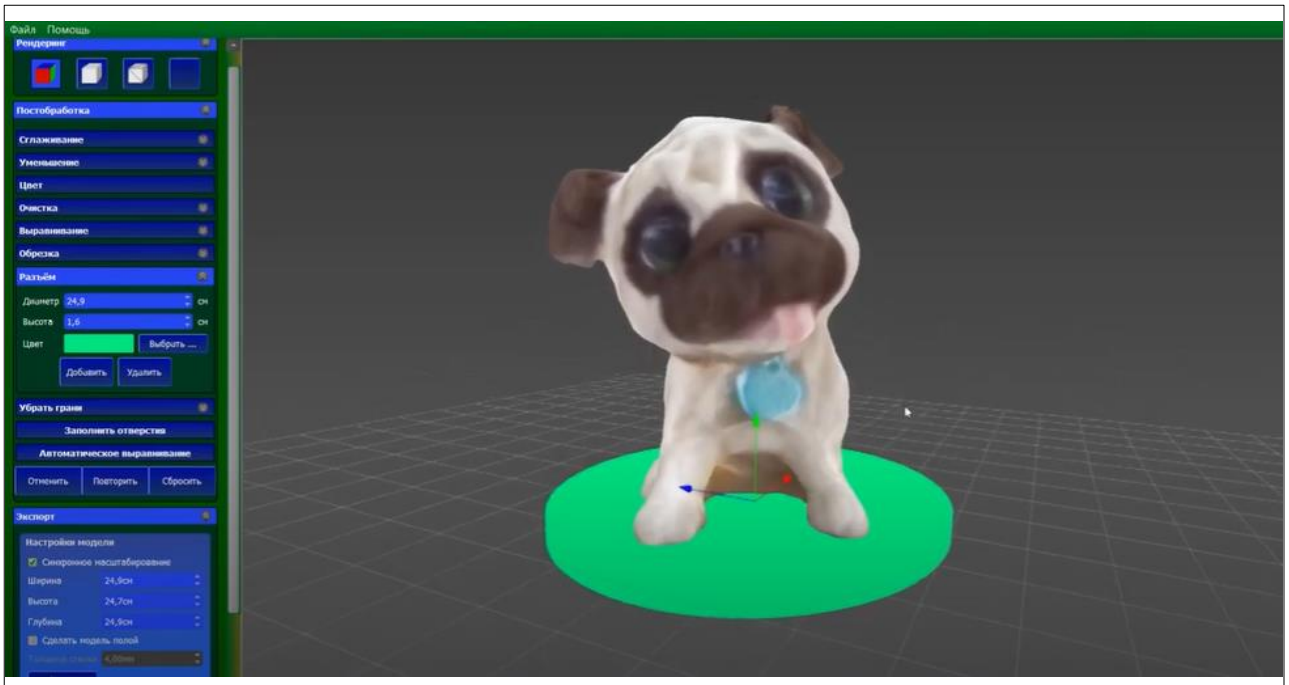


Рисунок 9 Создание постамента

Также есть возможность изменения цветовых параметров текстуры модели таких как яркость, контрастность, гамма.

Режим выравнивания модели по плоскости – позволяет пользователю выровнять модель по заданной плоскости. Обратите внимание, модель должна содержать плоскую поверхность

После завершения постобработки модели необходимо произвести экспорт результатов сканирования.

Задание

С помощью сканера 3DQ Planeta3D создать модель одним из способов описанных в предыдущих лабораторных работах, произведите постобработку модели и сохраните. Опишите в отчете инструменты используемые при постобработке, проведите сравнение одной модели из предыдущих лабораторных работ с моделью после постобработки.

Порядок выполнения работы

1. Изучите теоретические сведения перед выполнением работы.
2. Произведите подготовку устройств к сканированию.

3. Запустите программное обеспечение Planeta 3D.
4. Проведите сканирование объекта одним из изученных ранее способов.
5. Произведите постобработку модели с использованием инструментов описанных в теоретической части.
6. Сохраните трехмерную модель в одном из предложенных форматов и оцените ее качество.

Содержание отчета по работе

Продемонстрируйте учителю результат постобработки модели. Опишите в отчете инструменты используемые при постобработке, проведите сравнение одной модели из предыдущих лабораторных работ с моделью после постобработки.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Подготовьте устройство 3DQ Planeta3D к работе, проверьте работоспособность программного обеспечения Planeta3D.
2. Подготовьте поворотный стол к работе и проверьте работоспособность программного обеспечения.
3. Откройте «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
4. Изложите материал раздела «Краткие теоретические сведения» перед учениками и ответьте на вопросы.
5. Озвучьте задание лабораторной работы.
6. После завершения работы примите ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучите обработанную модель на предмет ошибок в области соответствия формам исходного объекта. При их наличии – предоставить возможность исправить модель.
2. Проверьте отчеты и дайте рекомендации в случае ошибочных суждений.

Лабораторная работа №15 «Дополнительные возможности постобработки и подготовка модели к печати»

Цель

Изучение возможностей программного обеспечения Planeta3D для подготовки модели к печати.

Краткие теоретические сведения

Мы уже ознакомились с возможностями постобработки модели. Теперь необходимо научиться осуществлять экспорт результатов сканирования.

Экспорт результатов сканирования

После завершения постобработки, нажав на кнопку «Сохранить модель», Вы можете сохранить модель на диске используя внешние форматы: PLY, OBJ, VRML, STL. Формат PLY позволяет сохранять модель вместе с текстурой.

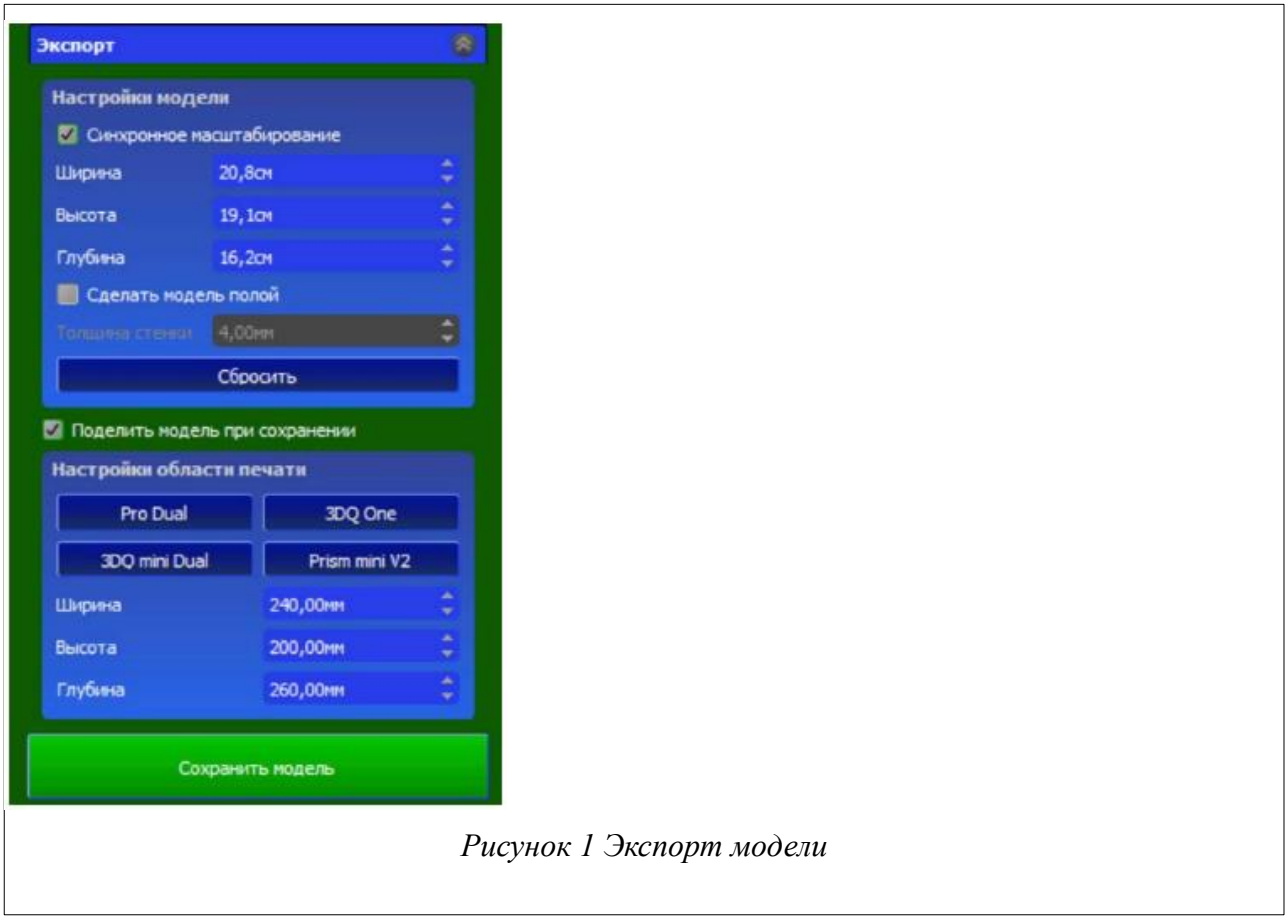


Рисунок 1 Экспорт модели

В настройках экспорта можно указать необходимые размеры экспортируемой модели. Таким образом можно масштабировать модель как

симметрично по осям, так и по каждой оси в отдельности (при выключенном режиме Синхронное масштабирование). Одновременно при экспорте можно сделать модель полой, указав нужную толщину стенок.

Режим «поделить модель при сохранении» позволяет разделить модель на отдельные файлы под требуемую область печати 3D принтера. Вы можете выбрать шаблон 3D принтеров или задать прямоугольную область 3D принтера. В результате будет создано требуемое количество файлов, каждый из которых будет готов к 3D печати.

Создание фотореалистичной текстуры модели

Режим создания фотореалистичной текстуры служит для построения фотореалистичной текстуры на основе фотографий, которые делаются сенсорами в процессе сканирования, согласно параметрам, задаваемым в меню Настройки - текстурирование:

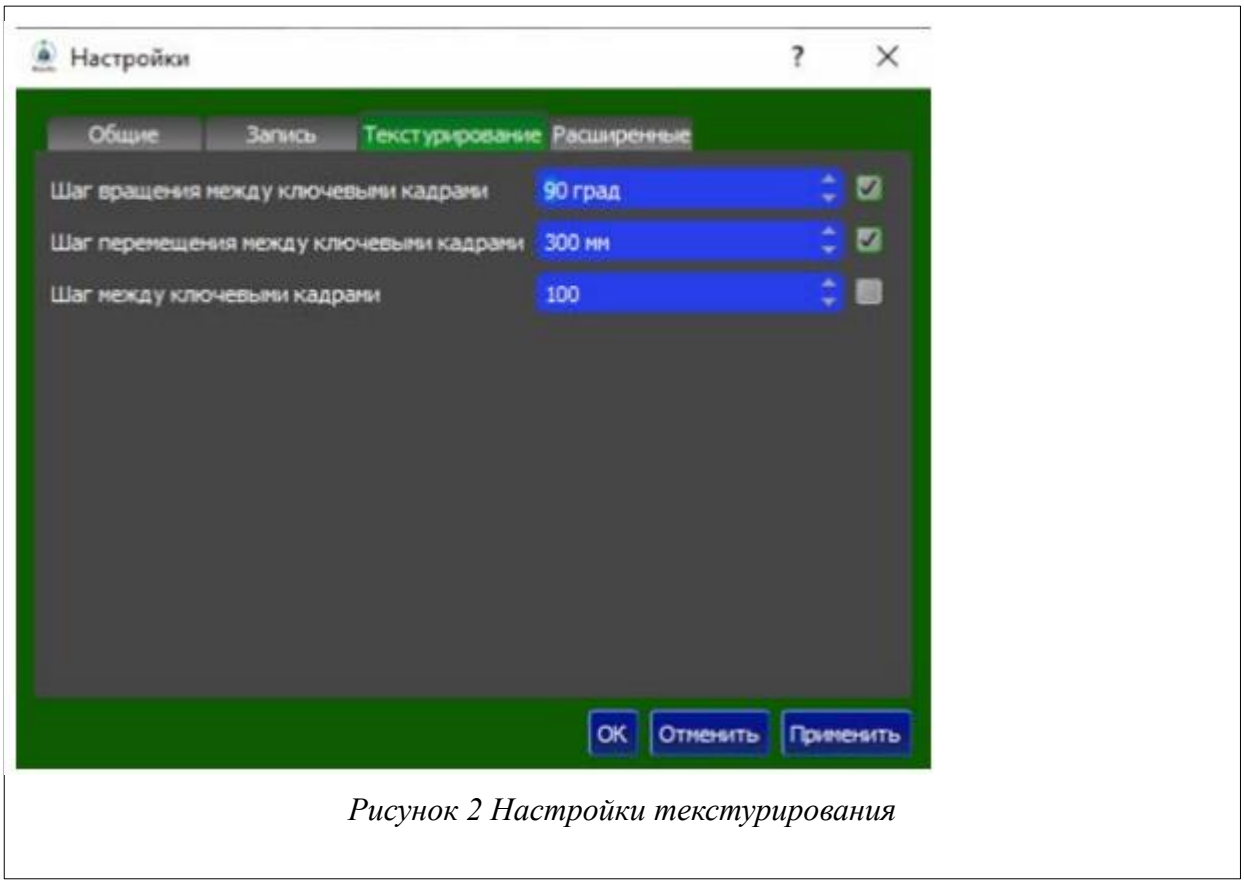


Рисунок 2 Настройки текстурирования

1. Шаг вращения между ключевыми кадрами задает угол поворота объекта между фотографиями;

2. Шаг перемещения между ключевыми кадрами задает линейное перемещение сенсора (объекта) между фотографиями;
3. Шаг между ключевыми кадрами задает количество ключевых кадров между фотографиями. То есть, например, фотография будет создаваться каждый сотый кадр.

Для включения каждого из вариантов, необходимо поставить галочку в соответствующей строке.

После завершения сканирования программа создает стандартную текстуру Рисунок 3.

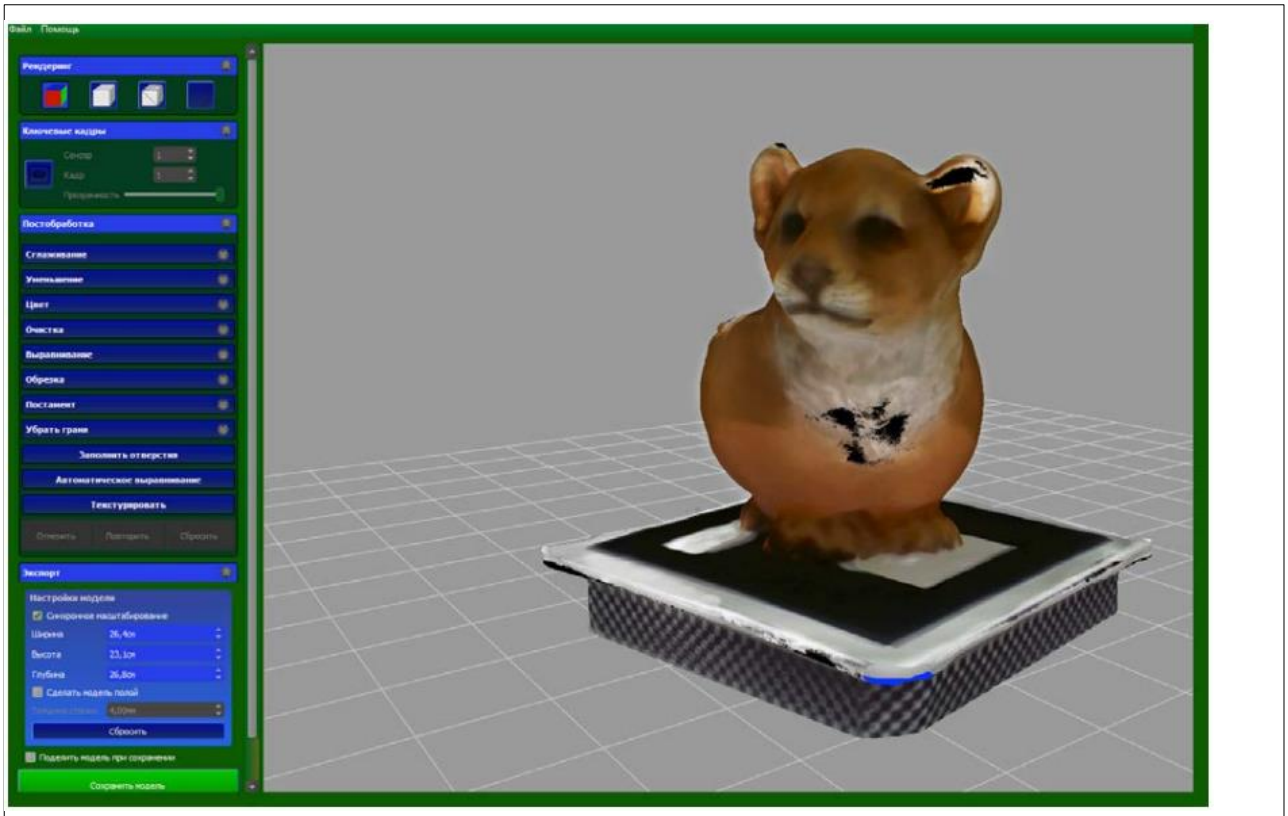


Рисунок 3 Сканироване со стандартной текстурой

При необходимости выбранный ключевой кадр можно исключить из расчета текстуры, сняв с него галочку у параметра «Использовать для текстуры» Рисунок 4.

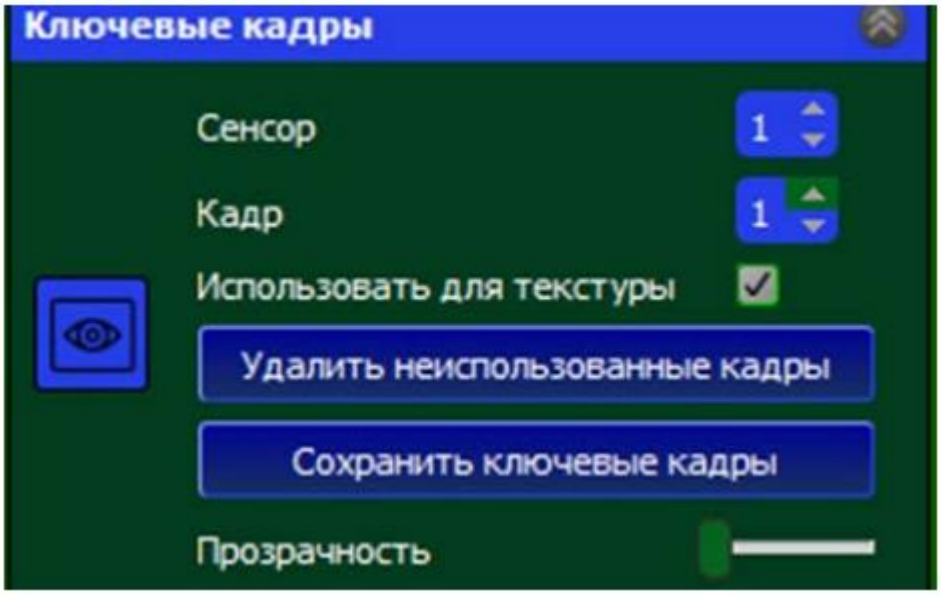


Рисунок 4 Использование ключевых кадров

Для удаления всех неиспользуемых ключевых кадров служит кнопка «Удалить неиспользованные ключевые кадры». Ключевые кадры сохраняются в форматах png и jpeg. Полученные фотографии можно использовать для построения 3D модели методом фотограмметрии в программе Planeta3D Photo. Если нажать Ctrl+R то ключевые кадры будут создаваться независимо от того, происходит 3D сканирование или нет. В этом случае в разделе Настройки-Расширенные задается количество кадров, которые пропускаются при создании ключевых кадров (параметр Пропустить).

Для создания фотореалистичной текстуры нажмите на кнопку «Текстурировать». В зависимости от разрешения сканирования, параметров создания текстуры и производительности компьютера процесс может занимать продолжительное время. Поэтому при нажатии на кнопку «Текстурировать» появляется окно с предупреждением. По окончании операции получится текстура, как на рисунке 5.

Рисунок 5 Модель с фотореалистичной текстурой

В случае если необходимо создать улучшенную текстуру позднее необходимо сохранить все данные как проект, нажав на кнопку Сохранить проект. Одновременно команда доступна в меню Файл (Файл-Сохранить). Для открытия проекта используйте команду Файл-Открыть Проект.

Задание

С помощью сканера 3DQ Planeta3D создайте модель в режиме фотореалистичной текстуры и произведите полную подготовку модели к печати на принтере.

Порядок выполнения работы

1. Изучите теоретические сведения перед выполнением работы.
2. Произведите подготовку устройств к сканированию.
3. Запустите программное обеспечение Planeta 3D.
4. Проведите сканирование в режиме фотореалистичной текстуры.
5. Произведите постобработку модели.
6. Подготовьте модель к печати на 3D принтере.

Содержание отчета по работе

Продемонстрируйте учителю модель для печати.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Подготовьте устройство 3DQ Planeta3D к работе, проверьте работоспособность программного обеспечения Planeta3D.
2. Подготовьте поворотный стол к работе и проверьте работоспособность программного обеспечения.
3. Откройте «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
4. Изложите материал раздела «Краткие теоретические сведения» перед учениками и ответьте на вопросы.
5. Озвучьте задание лабораторной работы.
6. После завершения работы примите ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучите обработанную модель на предмет ошибок в области соответствия формам исходного объекта. При их наличии – предоставить возможность исправить модель.
2. Проверьте качество получившейся модели для печати.

Лабораторная работа №16 «Сканирование объектов сложной формы с помощью Artec EVA»

Цель

Изучение правил сканирования объектов сложной формы с помощью сканера Artec EVA.

Краткие теоретические сведения

Прежде чем вести разговор о сканировании объектов сложной формы, необходимо дать определение подобным объектам. По предыдущим работам мы уже знаем, что ни один объект не может быть полноценно отсканирован с одного ракурса, те или иные грани будут расположены вне поля зрения нашего сканирующего модуля, но, между тем, есть объекты, для наиболее полного охвата которых необходимо будет продумать траекторию перемещения сканера вокруг них.

К сложным для сканирования объектам в первую очередь следует отнести объекты, состоящие из нескольких фигур, которые будут перекрывать сканеру обзор на соседей. Например, подобные сложности могут возникнуть при сканировании диорам или сложных составных конструкций.

Очевидным решением в данной ситуации было бы разобрать конструкцию и провести сканирование составляющих элементов по отдельности, но это возможно не всегда. В таком случае нам на помощь приходят портативные сканеры с обработкой данных на основе геометрии, которые добавляют данные в модель, основываясь на снимаемом окружении. Они позволяют размещать

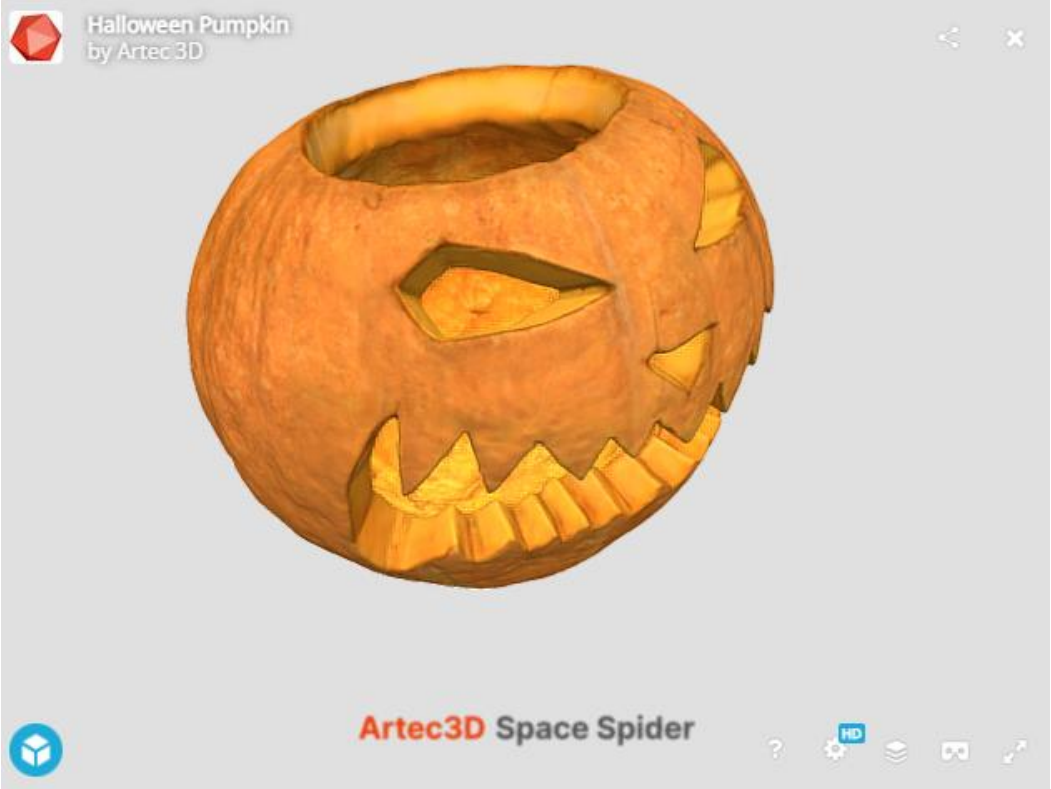


Рисунок 4

сканер с нужных ракурсов и избежать необходимости вручную склеивать результат при достаточной плавности перемещения.

Также сложность в построении вызывают объекты с острыми гранями, так как далеко не все сканеры могут зафиксировать очень мелкие детали. Лучи сканера не могут проецироваться в достаточном количестве на тонких краях таких предметов, как нож, например. Для сканирования предметов подобного рода следует пользоваться некоторыми хитростями. В первую очередь, следует проводить сканирование на фоне объектов с насыщенной текстурой, это позволит сканеру с большей вероятностью определить края интересующего нас объекта. Во-вторых, сканирование должно производиться со всех ракурсов, несмотря на то, что объект может иметь симметричную форму. И в-третьих, желательно отключить инструмент удаления основы, так как он может создать дополнительные трудности в совмещении разрозненных данных из-за отсутствия контекста из окружающих объектов.

Ну и наконец, следует упомянуть объекты с однотонной текстурой и повторяющейся геометрией, что может стать серьезным вызовом для многих

Рисунок 5

сканирующих устройств с привязкой к этим двум характеристикам модели. Проблема заключается в том, что сканер не может различить отдельные повторяющиеся элементы друг от друга по причине их схожести. Для облегчения задачи сканеру следует расположить объект сканирования таким образом, чтобы повторяющаяся геометрия постоянно попадала в условный кадр сразу в нескольких экземплярах, это обеспечит привязку разных деталей модели друг к другу и облегчит этап постобработки.

Рисунок 6

Задание

С помощью сканера Artec EVA создайте модель предложенного объекта, которая будет содержать наименьшее количество «белых пятен».

Порядок выполнения работы

1. Запустите Artec Studio 16.
2. Выполните настройку основных параметров программного обеспечения для проведения сканирования. Например, отключить инструмент удаления основы и подобрать наиболее контрастный фон для сканирования.
3. Проведите сканирование, обеспечив максимально плавное движение сканирующего устройства вокруг объекта. Следует помнить, что объект должен находиться в пределах фокусного расстояния сканера, а также то, что сканирование желательно производить под углом в 90 градусов к исследуемой поверхности.
4. Откройте итоговый файл созданной модели и оцените ее качество.

Содержание отчета по работе

Отчет должен содержать трехмерную модель сканируемого объекта.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Artec Studio 16.
2. Отсканировать демонстрационный объект с образованием «белых пятен» на итоговой модели.
3. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
4. Продемонстрировать искажения трехмерного представления объекта из-за неполного сканирования.
5. Озвучить задание лабораторной работы.
6. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – продемонстрировать ученику неучтенные ракурсы и предоставить возможность доработать модель.

Лабораторная работа №17 «Сканирование прозрачных поверхностей»

Цель

Изучение правил подготовки объекта с прозрачными элементами к сканированию с помощью Artec EVA.

Краткие теоретические сведения

Для абсолютного большинства технологий сканирования объекты с глянцевой, черной или прозрачной поверхностью представляют непосильную сложность. Происходит это за счет того, что перечисленные типы поверхностей отражают, поглощают или пропускают сквозь себя свет, на проекцию которого на объекте полагается большинство архитектурных решений.

Можно попробовать решить проблемы со сканированием черных и блестящих поверхностей с помощью манипуляций с настройками сканера. Если увеличить чувствительность и уменьшить яркость, то качество снимаемых с объекта данных должно значительно возрасти. Кроме того, необходимо помнить о том, что именно для сложных в обработке поверхностей особенно важно соблюдение требований по четко заданному перпендикулярному углу между сканером и поверхностью.

Рисунок 7

Если же перечисленные манипуляции с настройками не привели к желаемому результату, то можно воспользоваться специальным матирующим спреем либо маскирующей лентой наподобие малярного скотча. Кроме того, не стоит забывать об уровне освещения в помещении для сканирования. В темном помещении сканер будет засвечивать гораздо более крупные области, что приведет к снижению качества модели. Идеальным будет свет от многих источников, расположенных в разных направлениях относительно сканируемого объекта.

Рисунок 8

Отдельным пунктом идет сканирование прозрачных поверхностей. Так как сама суть прозрачной поверхности в том, что она пропускает весь свет, или большую его часть, сквозь себя, оптические сканеры оказываются бессильны при решении задачи переноса ее в виртуальное пространство. Если попробовать отсканировать прозрачный объект без предварительной обработки, то на его месте в сформированном облаке точек будет просвет. Самым лучшим выбором при таких исходных данных было бы использование тактильных сканеров. Но неужели оптические сканеры полностью бесполезны при решении подобной задачи?

Прежде всего, остается вариант с нанесением матирующего спрея или маскирующих лент на прозрачную поверхность. Благодаря этому поверхность

приобретает возможность удерживать на себе лучи света и передавать посредством этого данные сканирующему устройству. Но это подойдет только для объектов с простой поверхностью. А что делать, например, с сосудами, у которых нам необходимо узнать не только внешние контуры, но и внутреннюю форму? Давайте разберемся.

Есть достаточно необычный и интересный способ для сканирования стеклянных сосудов. Для его реализации необходимо налить жидкость, максимально контрастирующую с цветом фона и удерживающей световые лучи. Таким образом, при сканировании мы сможем выделить внутреннюю форму сосуда и потом, на стадии постобработки, совместить ее с внешней формой, что в итоге даст подробную модель имеющую реальную толщину стенок образца.

Рисунок 9

Задание

Проведите сканирование предложенного объекта, применяя методы, описанные в теоретической части.

Порядок выполнения работы

1. Запустите Artec Studio 16.
2. Выполните настройку основных параметров программного обеспечения для проведения тестирования. Обработать предмет сканирования с помощью доступных средств повышения отражательной способности поверхности.
3. Проведите сканирование, обеспечив максимально плавное движение сканирующего устройства вокруг объекта. Следует помнить, что объект должен находиться в пределах фокусного расстояния сканера, а также то, что сканирование желательно производить под углом в 90 градусов к исследуемой поверхности.
4. Откройте итоговый файл созданной модели и оцените ее качество.

Содержание отчета по работе

Отчет должен содержать трехмерную модель сканируемого объекта с наименее возможными искажениями.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Artec Studio 16.
2. Провести частичное сканирование объекта с прозрачной поверхностью с помощью Artec EVA.
3. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
4. Продемонстрировать результат сканирования без осуществления дополнительных манипуляций с объектом.
5. Озвучить задание лабораторной работы.
6. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет итогового качества модели. При наличии серьезных недостатков указать на проблемные места и дать советы по устранению негативных эффектов.

Лабораторная работа №18 «Реактивные элементы в цепях синусоидального тока»

Цель

Практическое ознакомление с установившимися режимами в R-, L- и C-ветвях синусоидального тока.

Краткие теоретические сведения

Реактивные сопротивления X_L, X_C и угол сдвига фаз

При анализе цепей синусоидального тока необходимо иметь в виду следующее:

- реактивное индуктивное сопротивление X_L индуктивной катушки и реактивное ёмкостное X_C сопротивление конденсатора зависят от частоты f источника синусоидального напряжения $u=U_m*sin(2\pi f+\Psi_u)$, т. е. $X_L=\omega L=2\pi fL$ и $X_C=1/(\omega C)$, где $\omega=2\pi f$ - угловая частота напряжения, рад/с; $f=1/T$ - циклическая частота, Гц; T - период синусоидального напряжения, с; Ψ_u его начальная фаза, рад или град;

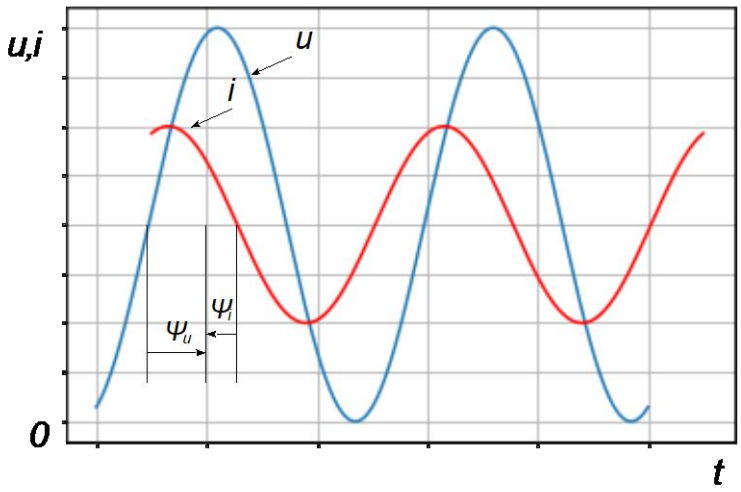


Рисунок 10 Осциллограмма напряжения и тока

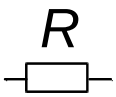
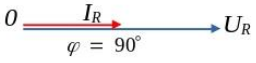
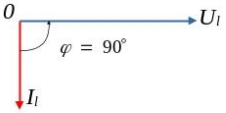
- в ветвях с реактивными элементами L и C между напряжением и током возникает фазовый сдвиг $\varphi = \Psi_u - \Psi_i$, где Ψ_i, где Ψ_i начальная фаза тока (рисунок 1). Угол φ (в рад или град) - алгебраическая величина, изменяющаяся в диапазоне от -90º (-π/2 рад) до +90º (-π/2 рад). Знак и величина угла зависят от типа и величины параметров последовательно соединённых элементов R, L и C и частоты f напряжения.

Векторные диаграммы напряжений и тока

В таблице 1 представлены типовые ветви схемы цепи синусоидального тока, векторные диаграммы напряжений и токов ветвей и углы сдвига фаз между их векторами. Анализ векторных диаграмм показывает, что резистивный элемент R является частотно-независимым элементом: ток и напряжение на его зажимах совпадают по фазе (форма тока i_R повторяет форму напряжения u_R), поэтому при определении (по осциллограммам) угла сдвига фаз между напряжением и током в ветвях цепи в качестве датчика тока можно использовать резистор с сопротивлением R_0, напряжение $u_{R0} = R_{0i}$ с зажимов которого подаётся на один из входов осциллографа.

В индуктивном элементе ток отстаёт по фазе от напряжения на 90º, а в ёмкостном - его опережает на 90º.

Таблица 1

№	Элементы ветви	Векторная диаграмма	Угол $\varphi = \neg_u - \neg_i$
1			$\varphi = 0$
2			$\varphi = 90^\circ$ ($\pi/2$)
3			$\varphi = -90^\circ$ ($-\pi$ /2)

Измерение угла φ

Значение угла сдвига фаз φ в цепях определяют косвенным методом, измеряя временные интервалы на осциллограммах, т. е. $|\varphi| = 360^\circ \Delta t/T$, где Δt временной интервал (рисунок 3) между максимальны значениями синусоид напряжения (синего цвета) и тока (зеленого цвета); угол φ берется со знаком "плюс", если ток отстаёт по фазе от напряжения, и со знаком "минус", если ток опережает по фазе напряжение.

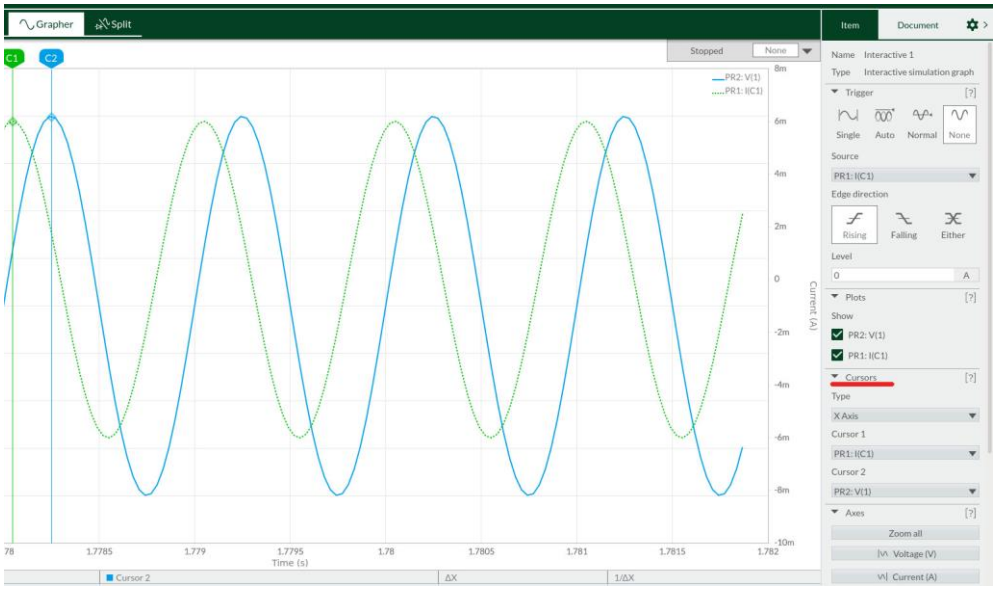


Рисунок 11: Визирные линии

При измерении интервала времени Δt следует использовать визирные линии (визиры). Активировать визиры можно в режиме Grapher или Split в разделе Cursors, расположенном справа от рабочего поля (рисунок 2).

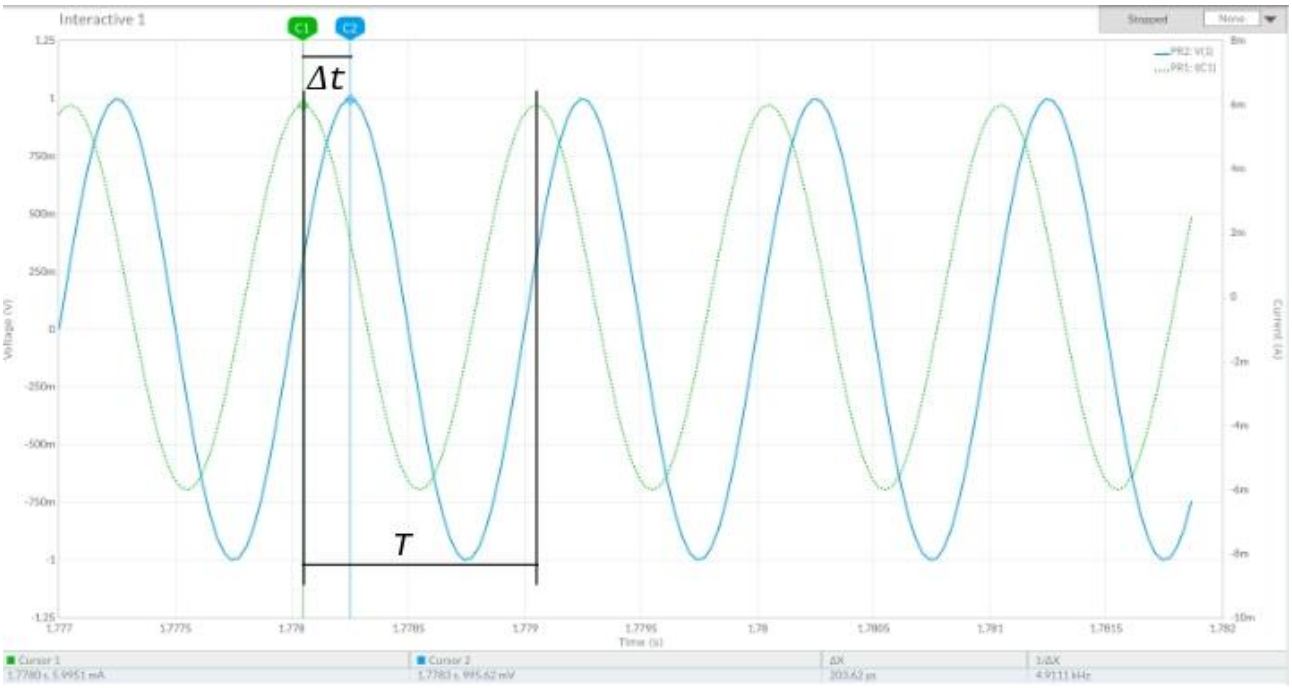


Рисунок 12: Измерение угла сдвига фаз

Multisim Online

Для самостоятельной сборки цепи необходимо воспользоваться Online симулятором программы Multisim, доступным по ссылке https://www.multisim.com.

Задание

Рассчитать индуктивное сопротивление X_L катушки и ёмкостное сопротивление X_C конденсатора при частотах, указанных в таблице 2, и занести полученные значения сопротивлений в таблицу.

Значения индуктивности катушки и ёмкости конденсатора: L = 100, мГн, C = 100, мкФ.

Таблица 2

Сопротивление X		при частоте f, Гц						
		30	40	50	60	80	100	120
Рассчитано	X_L, Ом							
Измерено	U, В							
	I, А							
	X_L, Ом							
Рассчитано	X_C, Ом							
Измерено	U, В							
	I, А							
	X_C, Ом							

Построить на одном графике $X_L(f)$ и $X_C(f)$. Отметить координаты точки пересечения графиков - возможного режима резонанса напряжений при последовательном соединении катушки и конденсатора между собой и с источником синусоидального напряжения.

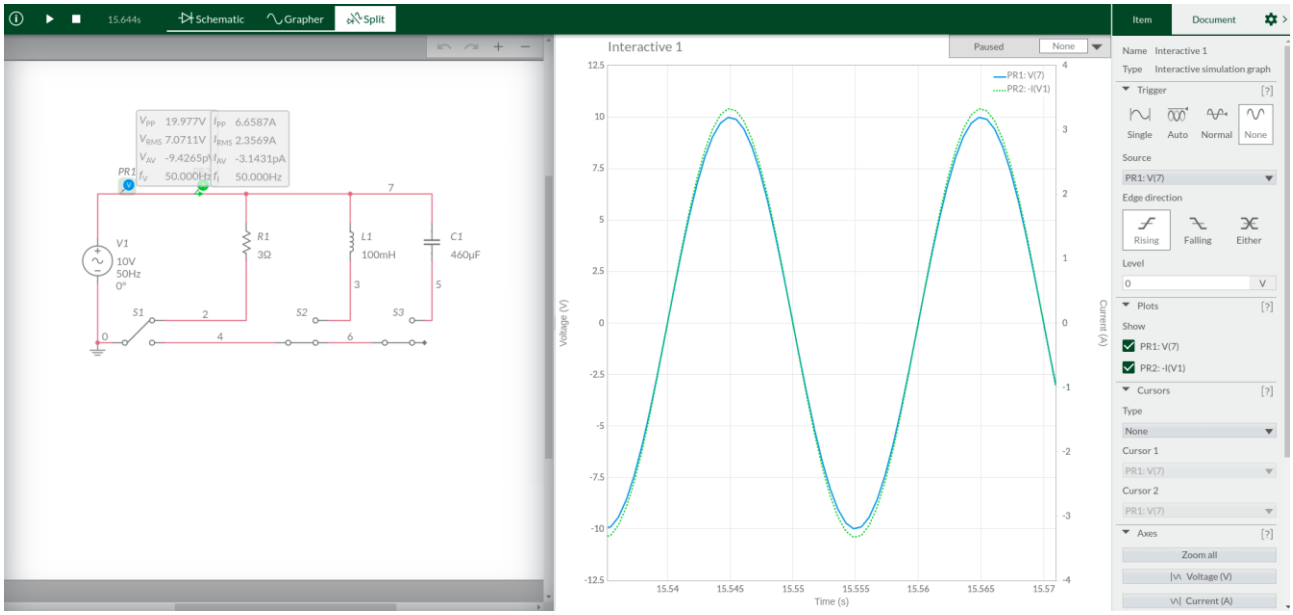


Рисунок 13: Схема и осциллограмма

Провести измерение токов, напряжений и углов сдвига фаз между ними в ветвях, содержащих соответственно резистивный R_1, индуктивный L_1 и емкостный C_1 элементы.

Параметры идеального источника синусоидального напряжения V_1: ЭДС E = 10 В (действующее значение), f = 50 Гц; Ψ_u = 0.

Для получения осциллограмм необходимо установить датчики напряжения и тока как показано на рисунке 4.

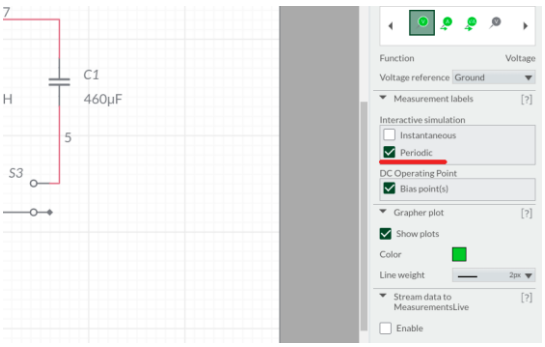


Рисунок 14 Настройки пробника

Порядок выполнение работы

1. подключить резистор R_1 к источнику синусоидального напряжения V_1, запустить моделирование и убедиться (анализируя осциллограммы), что угол сдвига фаз между напряжением и током равен нулю. Ток $I_1 = U/R_1$;
2. отключить резистор R_1 и подключить катушку L_1 к источнику V_1. Показания пробника V_{rms} и амперметра I_{rms} при f = 50 Гц занести в таблицу 2. Изменяя ступенчато частоту напряжения (30, 40, 50, 60, 80, 100, 120 Гц),

заносить показания вольтметра и амперметра в таблицу 2. Рассчитать сопротивление $X_{L2}(f) = U_L/I_L$ и сравнить полученные значения со значениями, полученными при выполнении задания 1. Убедиться (анализируя осциллограммы), что ток i_L отстает по фазе от напряжения u_L на угол $\varphi = 90^{\circ}$. Перенести в отчёт осциллограммы $u_L(t)$ и $i_L(t)$ при $f = 50$ Гц;

3. повторить предыдущее задание относительно конденсатора C_3, предварительно отключив от источника ветвь с элементом L_2 и подключив к источнику V_1 ветвь с конденсатором C_3. Показания приборов заносить в таблицу 2. Рассчитать сопротивление конденсатора $X_{C3}(f) = U_C/I_C$ (см. табл. 2) и сравнить полученные значения со значениями в ходе расчетов. Убедиться (анализируя осциллограммы), что ток i_C опережает по фазе напряжение u_C на угол $\varphi = 90^{\circ}$.
4. Перенести в отчёт осциллограммы $u_C(t)$ и $i_C(t)$ при $f = 50$ Гц.

Содержание отчета по работе

Отчет должен содержать снимки экрана выполнения заданий, расчетные формулы, заполненную таблицу и осциллограммы.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Multisim Online.
2. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
3. Запустить симулятор и продемонстрировать работу программы.
4. Озвучить задание лабораторной работы.
5. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учеником.
2. Проверить работу модели в симуляторе – в случае ошибок, исправить вместе с учеником.

Лабораторная работа №19 «Неразветвлённые цепи синусоидального тока»

Цель

Практическое ознакомление с установившимися режимами в последовательных RL-, RC- и RLC-цепях синусоидального тока.

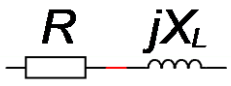
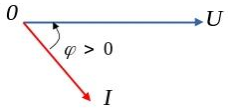
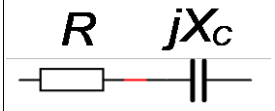
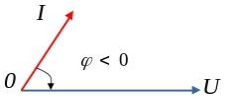
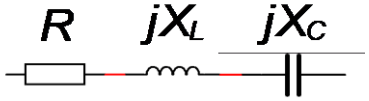
Краткие теоретические сведения

Векторные диаграммы напряжений и тока в RL-, RC- и RLC-ветвях

В таблице 3 представлены типовые ветви схемы цепи синусоидального тока, векторные диаграммы напряжений и токов ветвей и углы сдвига фаз между их векторами.

В индуктивном элементе ток отстаёт по фазе от напряжения на 90º, а в ёмкостном - его опережает на 90º. В RL-, RC- и RLC-ветвях углы сдвига фаз зависят от значений параметров элементов ветвей и определяются, в общем случае, по формуле $\varphi = arctg((X_L - X_C)/R)$.

Таблица 3

№	Элементы ветви	Векторная диаграмма	Угол $\varphi = \Psi u - \Psi i$
1			φ = arctg(X_L/R)
2			φ = arctg(-X_C/R)
3		*а*) $X_L > X_C$; см. ветвь с RL; *б*) $X_L < X_C$; см. ветвь с RC; *в*) $X_L = X_C$; см. ветвь R.	φ = arctg((X_L-X_C)/R)

Измерение угла φ

Значение угла сдвига фаз φ в цепях определяют косвенным методом, измеряя временные интервалы на осциллограммах, т. е. $|\varphi| = 360^{\circ}\Delta t/T$, где Δt временной интервал (рисунок 2) между максимальны значениями синусоид напряжения (синего цвета) и тока (зеленого цвета); угол φ берется со знаком "плюс", если ток отстаёт по фазе от напряжения, и со знаком "минус", если ток опережает по фазе напряжение.

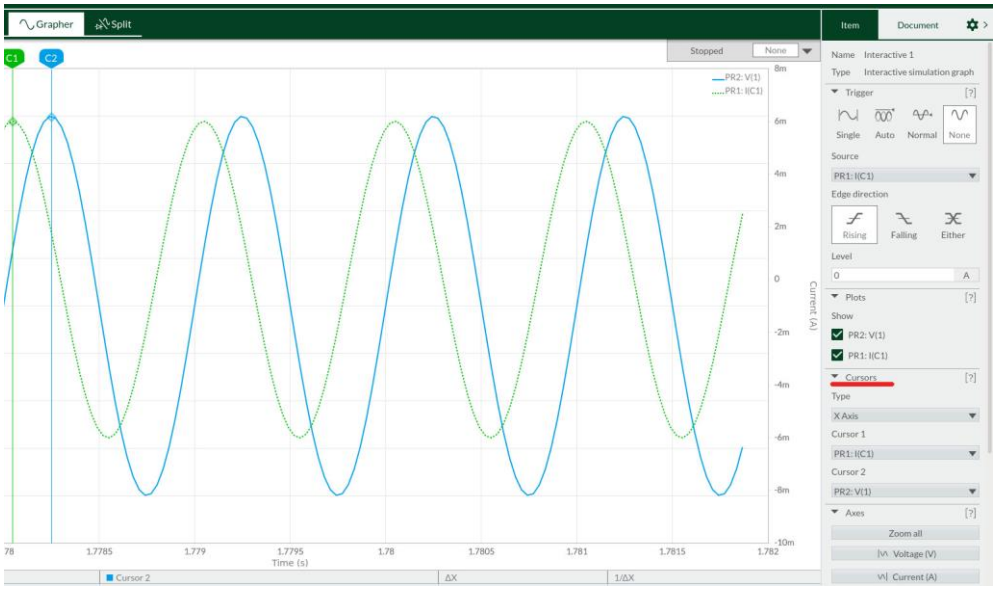


Рисунок 15: Визирные линии

При измерении интервала времени Δt следует использовать визирные линии (визиры). Активировать визиры можно в режиме Grapher или Split в разделе Cursors, расположенном справа от рабочего поля (рисунок 1).

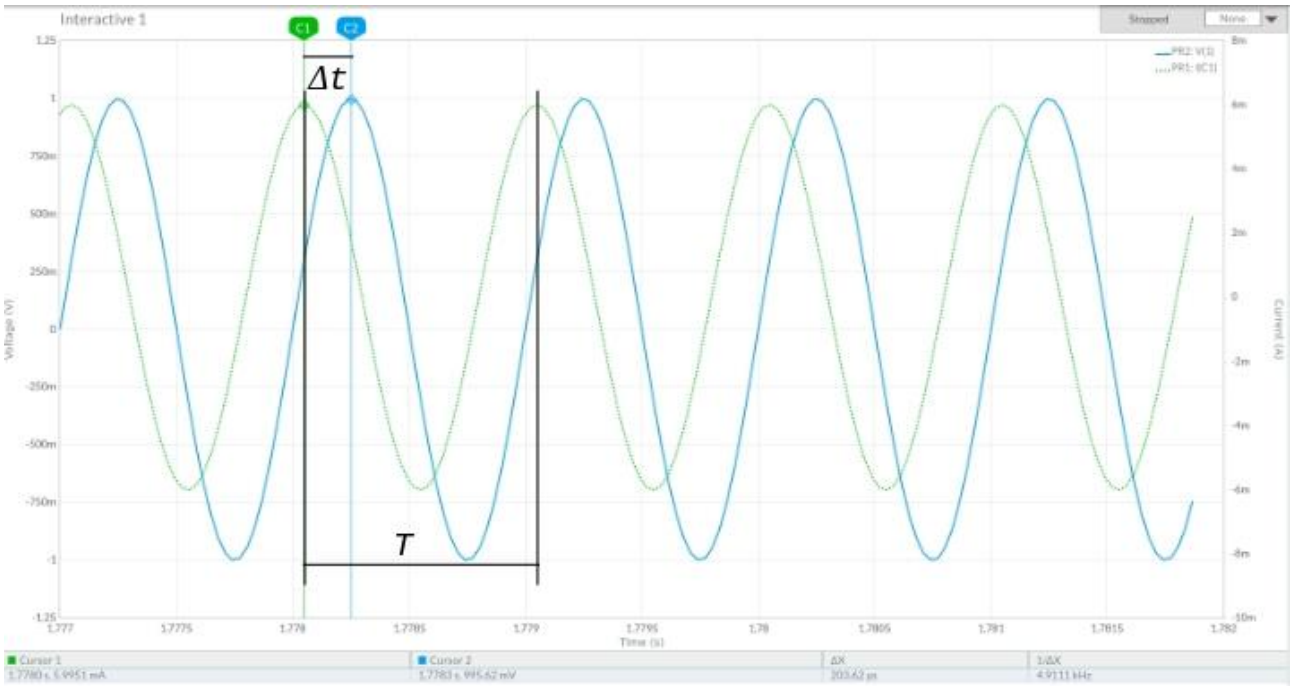


Рисунок 16: Измерение угла сдвига фаз

Задание

Провести измерения токов, напряжений и углов сдвига фаз между ними в ветвях, содержащих соответственно RL-, RC- и RLC-элементы (см.рис.3).

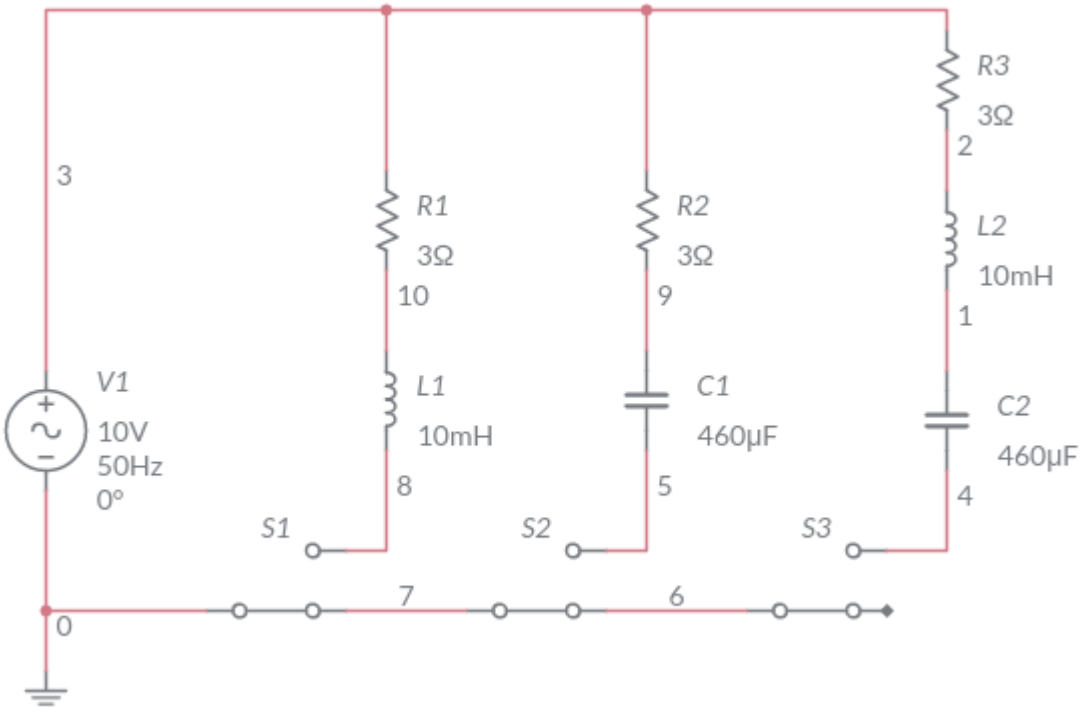


Рисунок 17: Схема

Таблица 4

Ветвь	Измерено			Рассчитано		
	U, В	*I*, А	), град	*Z* = *U*/*I*, Ом	*R* = *Z*cos), Ом	*X* = *Z*sin), Ом
R_4L_4						
R_5C_5						
$R_6L_6C_6$						

Порядок выполнение работы

1. Установить частоту f = 50 Гц источника напряжения V_1 и подключить к нему ветвь R_1L_1. Показания приборов занести в таблицу 3. Угол φ определить косвенным методом, воспользовавшись осциллограммами напряжения и тока ветви. Для удобства измерений изменяйте чувствительность каналов и длительность развёртки осциллографа.
2. Рассчитать полное Z_1, активное R_1 и реактивное X_{L4} сопротивления ветви R_1L_1 и занести их значения в таблицу 3. Убедиться, что ток *i* в RL-ветви отстаёт по фазе от напряжения *u* на угол $\varphi_4 = arctg(X_{L4}/R_4)$; Перенести в отчёт осциллограммы напряжения и тока RL-ветви.
3. Повторить предыдущее задание для ветви R_2C_1, предварительно отключив от источника V_1 ветвь с катушкой индуктивности. Убедиться, что ток *i* в RC-ветви опережает по фазе напряжения u на угол $\varphi_5 = arctg(X_{C5}/R_5)$.

4. Зарисовать осциллограммы напряжения и тока RC-ветви.
5. Повторить предыдущее задание для ветви с элементами R_3, L_2 и C_2, предварительно отключив от источника V_1 ветвь с кондкесатором. Убедится, что в RLC-ветви угол сдвига фаз φ между напряжением и током зависит от величины реактивного сопротивления $X = X_{L2} - X_{C2}$.
6. Если при частоте f = 50 Гц, угол $\varphi = arctg(X_{L2} - X_{C2})/R_3 > 0$, то, уменьшив частоту до 20-30 Гц, угол φ_6 изменит свой знак, и наоборот, если при f = 50 Гц, угол $\varphi_6 < 0$, то, увеличив частоту f до 100…120 Гц, ток будет отставать по фазе от напряжения, при этом угол $\varphi > 0$.

Содержание отчета по работе

Отчет должен содержать снимки экрана выполнения заданий, расчетные формулы, заполненную таблицу и осциллограммы.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Multisim Online.
2. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
3. Запустить симулятор и продемонстрировать работу программы.
4. Озвучить задание лабораторной работы.
5. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учеником.
2. Проверить работу модели в симуляторе – в случае ошибок, исправить вместе с учеником.

Лабораторная работа №20 «Резонанс напряжений в цепях синусоидального тока»

Цель

Исследование явления резонанса в последовательном колебательном контуре и определение параметров колебательных контуров.

Краткие теоретические сведениям

Под резонансом понимают такой режим работы электрической цепи, при котором её входное сопротивление имеет чисто резистивный характер и, следовательно, сдвиг фаз между напряжением u и током i на её входе равен нулю ($\varphi = 0$).

Цепи, в которых возникают резонансные явления, называют резонансными цепями или колебательными контурами. Простейший колебательный контур содержит один индуктивный L и один емкостный C элементы, соединенные между собой и источником синусоидального напряжения последовательно (последовательный колебательный контур) или параллельно (параллельный колебательный контур).

Различают две основные разновидности резонансных режимов: резонанс напряжений и резонанс токов. В данной лабораторной работе будет рассмотрен резонанс напряжений.

Резонанс напряжений

Резонанс напряжений возникает в последовательном колебательном контуре (рисунок 1). В схему замещения цепи, кроме индуктивного L и ёмкостного C элементов, включен также элемент R, учитывающий все виды активных потерь в контуре (в катушке, в конденсаторе, во внутреннем сопротивлении источника питания, в соединительных проводах).

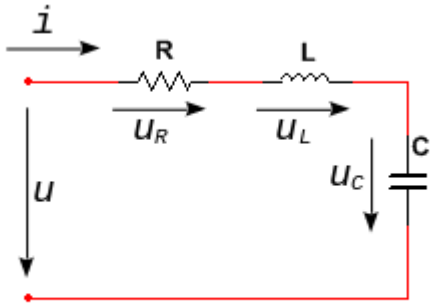


Рисунок 18: Последовательный колебательный контур

Условием наступления резонанса напряжений в схеме (рисунок 1) является равенство нулю реактивного сопротивления на входе цепи:

$X_{PH} = X_{L(PH)} - X_{C(PH)} = 0$ или

$\omega_{PH}L = 1/(\omega_{PH}C)$,

откуда угловая (в рад/с) и циклическая (в Гц) резонансные частоты контура

$$f_{\text{PH}} = \frac{1}{2\pi\sqrt{LC}} \qquad (12)$$

и

$$\omega_{\text{PH}} = \frac{1}{\sqrt{LC}}. \qquad (13)$$

Характеристическое (волновое) сопротивление ρ (в Ом) последовательного колебательного контура равно его индуктивному или ёмкостному сопротивлению при резонансе:

Добротностью Q контура называют отношение характеристического сопротивления ρ контура к активному сопротивлению R при резонансе:

$$Q = \frac{\rho}{R} = \frac{X_{L(\text{PH})}}{R} = \frac{X_{\text{C(PH)}}}{R}. \qquad (14)$$

Чем больше ρ и меньше R, тем "добротнее" контур, тем будут уже частотные характеристики тока и напряжений на элементах контура. В радиотехнических контурах добротность Q = 100…1000; в электрических цепях добротность обычно не превышает 3…5.

Добротность показывает, во сколько раз напряжение на зажимах конденсатора U_C или индуктивное напряжение U_L катушки при резонансе больше напряжения питания контура U:

$$Q = \frac{U_{C(PH)}}{U} = \frac{U_{L(PH)}}{U} = \frac{\rho}{R}. \quad (15)$$

Ток I при резонансе напряжений имеет максимальное значение, т.к.

$$I = \frac{U}{\sqrt{R^2 + (X_{L(\text{PH})} - X_{C(\text{PH})})^2}} = \frac{U}{R} = I_{max}. \quad (16)$$

Задание

Рассчитать резонансную частоту последовательного колебательного контура представленного на рисунке 2 произвести измерения и сравнить полученные значения с расчетным.

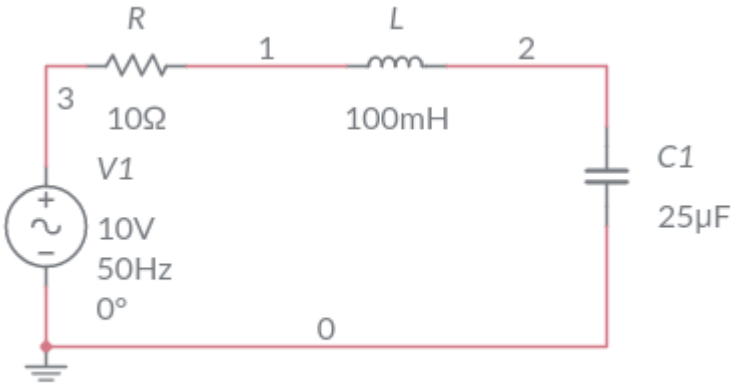


Рисунок 19 Схема

Порядок выполнение работы

1. Установить параметры идеального источника синусоидального напряжения V_1: ЭДС E = 10 В (действующее значение), f = 50 Гц; Ψ_u = 0; Значения резистора, индуктивности катушки и ёмкости конденсатора: R = 10, Ом, L = 100, мГн, C = 25, мкФ.
2. Произвести измерения φ, U_R, U_L, U_C, I при частотах указанных в таблице.

Таблица 5

Параметр	**Частота *f*, Гц**									
	30	40	50	60	70	90	110	130	140	f_{PH}
I, A										
U_R, В										
U_L, В										
U_C, В										
φ, град										

3. Рассчитать параметры колебательного контура:
    1. добротность: $Q_{PH} = U_C / U$;
    2. характеристическое сопротивление: $\rho = U_C / I$;
    3. полосу пропускания: $\Delta f_{PH} = f_{PH} / Q_{PH}$;
4. Построить графики амплитудно-частотных характеристик тока и напряжений и фаза-частотную характеристику. На одном графике зависимость $I_0(f)$; $U_R(f)$; $U_L(f)$; $U_C(f)$ на другом $\varphi(f)$.

Содержание отчета по работе

Отчет должен содержать снимки экрана выполнения заданий, расчетные формулы, заполненную таблицу и построенные графики.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Multisim Online.
2. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
3. Запустить симулятор и продемонстрировать работу программы.
4. Озвучить задание лабораторной работы.
5. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учеником.
2. Проверить работу модели в симуляторе – в случае ошибок, исправить вместе с учеником.

Лабораторная работа №21 «Переходные процессы в цепях постоянного тока»

Цель

Экспериментальное исследование апериодических переходных процессов в линейных электрических цепях первого и сопоставление экспериментальных результатов с предварительно рассчитанными параметрами.

Краткие теоретические сведения

Переходным процессом называют процесс изменения токов и напряжений в цепи при переходе от одного установившегося режима к другому. Причиной, вызывающей начало переходного процесса, является коммутация, под которой понимают отключение цепи или её подключение к внешнему источнику питания, либо скачкообразное изменение топологии или параметров элементов цепи. В общем случае вид кривой переходного процесса зависит как от изменения топологии цепи, накопленной в ней энергии, так и от видов действующих в цепи ЭДС источников напряжения и токов источников тока.

Переходный процесс в цепи может протекать как за счёт начального запаса энергии, накопленного в реактивных L и C элементах, так и за счёт энергии внешнего источника. При этом переходный процесс, протекающий за счёт расходования накопленной в элементах L и C энергии, называют свободным процессом или процессом собственных колебаний, а режим, близкий к стационарному, который устанавливается в цепи по истечении времени переходного процесса с момента коммутации, называют установившимся режимом; напряжения и токи в установившемся режиме называют установившимися напряжениями и токами.

В общем случае напряжения и токи цепи в переходном режиме выражают в виде суммы установившихся и свободных составляющих, т.е.

$u = u_y + u_{св}$ *и* $i = i_y + i_{св}$.

Переходные процессы в линейных цепях первого порядка

На рисунках 1 и 2 изображены схемы RL- и RC-цепей, входы которых подключаются к источникам постоянного напряжения U. В линейных цепях первого порядка переходные процессы описываются экспоненциальными уравнениями.

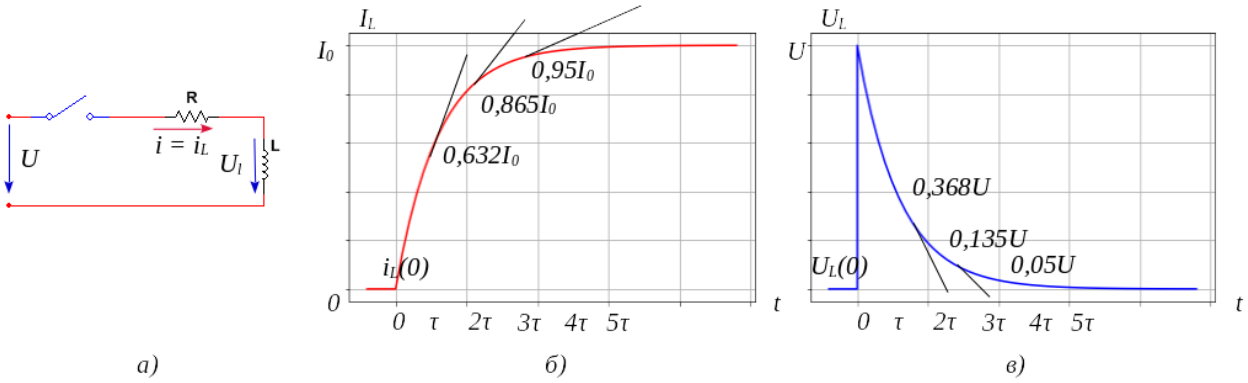
Для RL-цепи (рисунок 1, а) ток и напряжение на индуктивной катушке:

$i_L(t) = I_0(1 - e^{-at}) = I_0(1 - e^{-t/\tau});$

$u_L(t) = L[di_L(t)/dt] = Ue^{-at},$

где $I_0 = U/R$ - установившийся ток; $\tau = L/R$ - постоянная времени в секундах; $a = 1/\tau$ коэффициент затухания переходного процесса (1/с).

Графики тока $i_L(t)$ и напряжения $u_L(t)$ представлены на рисунке 1, б и в, где $t = 0$ означает мгновение до коммутации.



а) б) в)

Рисунок 20: RL- цепь

Анализ графиков показывает, что ток в RL-цепи постепенно нарастает до своего установившегося значения и тем медленней, чем больше постоянная времени τ время, в течение которого переходная величина изменяется на 0,632 от своего размаха I_0.

Если снять осциллограмму переходного тока, то значение τ можно определить по длине подкасательной, получаемой после проведения касательной из точки 0 до пересечения с горизонтальной линией (I_0) и опускания перпендикуляра на ось абсцисс (или используя другие точки осциллограммы для проведения касательной, например, точку $0{,}632I_0$ или точку $0{,}865I_0$ (см. рисунок 1, б)).

В инженерных расчётах время переходного процесса принимают равным $t_{пп} \approx 3\tau$; при этом переходная величина достигает порядка 0,95 своего установившегося значения. При более точных расчётах принимают $t_{пп} \approx 5\tau$, при котором переходная величина, ток, $i_L(5\tau) \approx 0{,}993 I_0$.

На графике $u_L(t)$ (рисунок 1, в) длина подкасательной на оси абсцисс соответствует постоянной времени τ цепи, в течение которого значение напряжения $u_L(0+) = U$ уменьшается в $e \approx 2{,}72$ раз. Чем больше τ, тем медленнее уменьшается напряжение u_L.

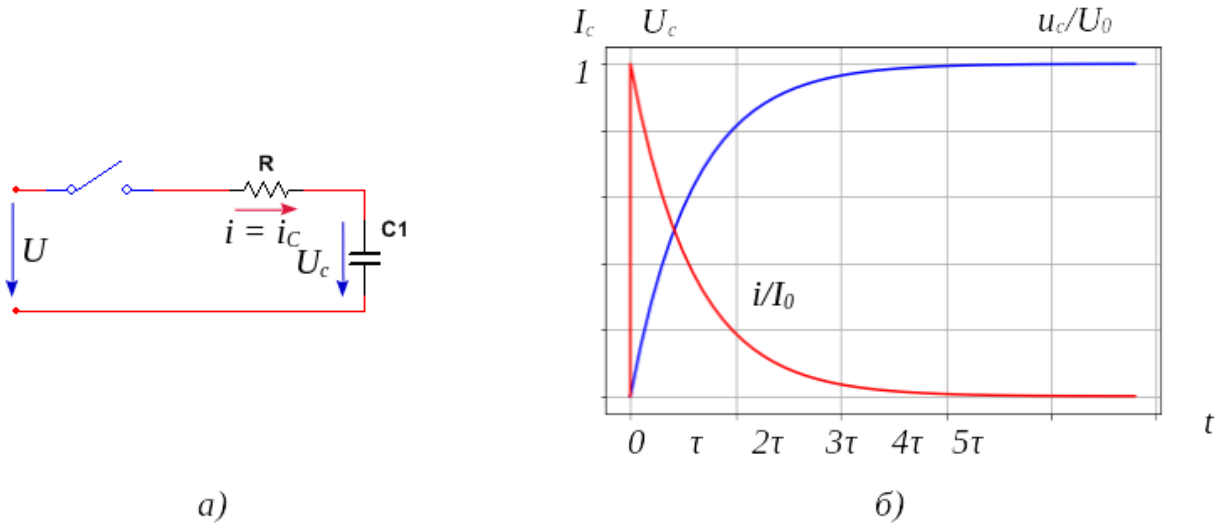


Рисунок 21: RC- цепь

При подключении RC-цепи (рисунок 2, а) к источнику постоянного напряжения U напряжение и ток конденсатора равны:

$$U_C(t) = U(1 - e^{-t/\tau});$$

$$i_C(t) = C(du_C(t)/dt) = I_0 e^{-at} = I_0 e^{-t/\tau},$$

где $I_0 = U/R$ - ток при $t = 0+$; $\tau = RC$ - постоянная времени; $a = 1/\tau$ □ коэффициент затухания переходного процесса; $t = 0+$ - мгновение после коммутации.

Нормированные графики $u_C(t)/U$ и $i_C(t)/U$ представлены на рисунке 2, б.

Если сравнить графики переходного тока i_L и напряжения u_L в RL-цепи (рисунок 2, б) с током i_C и напряжением u_C в RC-цепи (рисунок 2, б), то можно

заметить, что графики i_L и u_C, u_L и i_C внешне идентичны, так как законы изменения переходных величин одинаковые.

Задание

Рассчитать постоянную времени переходного процесса для RC и RL схем представленной на рисунке 3 по расчетным формулам, произвести измерения и сравнить результаты.

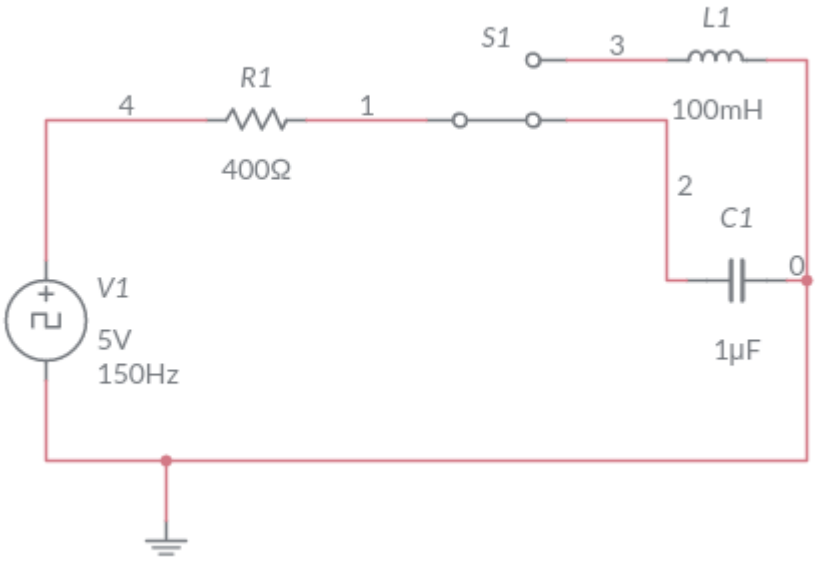


Рисунок 22: Схема для измерений

Порядок выполнения работы

1. Установить параметры генератора V_1: ЭДС E = 5 В, f = 150 Гц; Значения резистора, индуктивности катушки и ёмкости конденсатора: R = 400, Ом, L = 100, мГн, C = 1, мкФ.
2. Ключ S1 предназначен для переключения между RC- и RL-цепями.
3. Установить пробники для измерения напряжения входного с источника, напряжения на резисторе R_1 и тока в ветви. (см. рисунок 4)

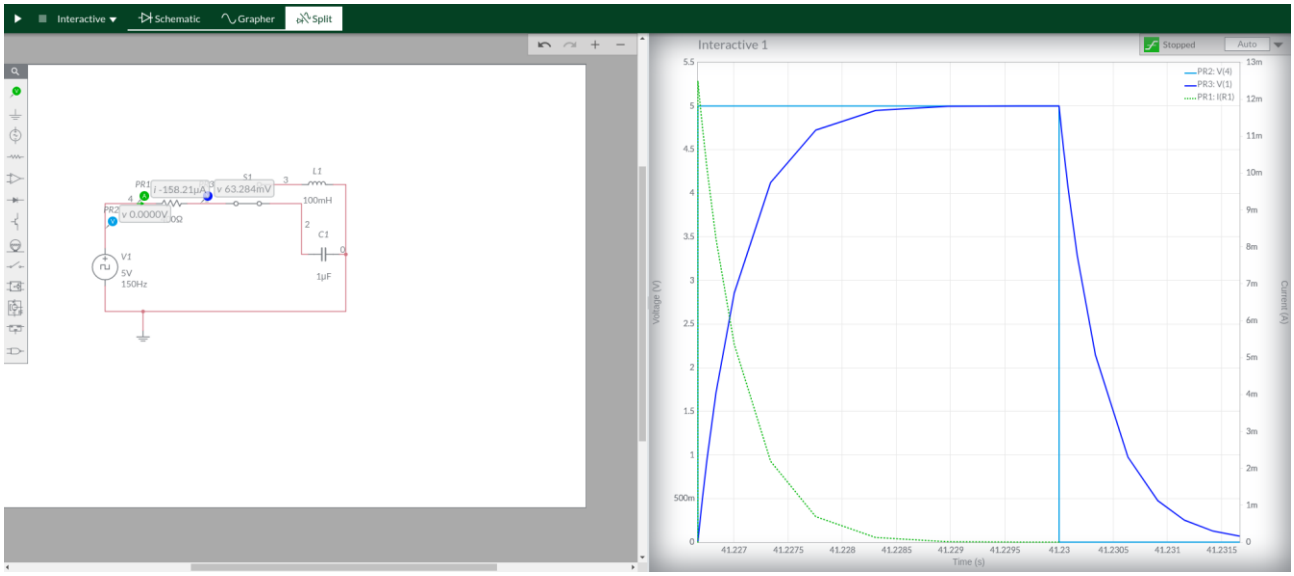


Рисунок 23: Схема с пробниками и график переходного процесса

4. Измерить значения силы тока и напряжения в моменты времени указанные в таблице 1 для RC- и RL- цепей.

Таблица 6

Параметр	**Значения параметра во времени действия**					
	Импульс, мкс			**Пауза, мкс**		
	0	5	10	0	5	10
$I_L(t)$						
$U_L(t)$						
$I_C(t)$						
$U_C(t)$						

Содержание отчета по работе

Отчет должен содержать снимки экрана выполнения заданий, расчетные формулы и заполненную таблицу.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере Multisim Online.
2. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
3. Запустить симулятор и продемонстрировать работу программы.
4. Озвучить задание лабораторной работы.
5. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учеником.
2. Проверить работу модели в симуляторе – в случае ошибок, исправить вместе с учеником.

Лабораторная работа №22 «Изучение среды моделирования LTspice»

Цель

Изучить программу для моделирования электрических схем LTspice.

Краткие теоретические сведения

Моделирование электрических схем непосредственно связано с проектированием соответствующих электронных устройств. В системах проектирования широкое распространение получили программы моделирования, объединенные общим названием – SPICE-программы.

Моделирование аналоговой схемы всегда было неотделимо от аналоговой разработки микросхем. Симуляторы SPICE — это единственный способ тестировать схему до интеграции на чип. Моделирование SPICE позволяет измерить токи и напряжения, что невозможно сделать любым другим способом. Успех симуляторов аналоговой схемы позволил распространить моделирование на разработку схем уровня платы. Схему более просто во многих случаях имитировать, чем макетировать, и возможность в моделировании проанализировать схему для производительности и задач ускоряет разработку устойчивых схем.

Особенность программы моделирования LTSpice состоит в том, что она содержит встроенные макромодели для мощных импульсных контроллеров и регуляторов. Кроме SPICE-программы, система проектирования включает схемный редактор и средства отображения результатов моделирования, позволяющие проводить дальнейший анализ. В состав системы входит также встроенная база данных для большинства импульсных устройств компании Linear Technology и многих пассивных компонентов. База данных устройств, схемный редактор, программа моделирования и система графического отображения результатов интегрированы в одну систему. Пакет LTspice полностью доступен для общего использования.

Программа состоит из высокоэффективного симулятора SPICE, расширенного со смешанной возможностью моделирования режима.

Интегрированное средство просмотра формы сигнала отображает моделируемые формы сигнала и позволяет дальнейшее исследование данных моделирования. Благодаря смешанной возможности моделирования режима и многим другим расширениям по сравнению с предыдущими программами SPICE, значительно улучшается быстродействие моделирования, в то время как точность моделирования сохраняется.

Для знакомства с программой прежде всего вам необходимо установить LTspice на свой компьютер с использованием операционной системы Windows или macOS. Установочный файл LTspice IV можно бесплатно скачать с сайта корпорации Linear Technology. После того, как вы скачали самораспаковывающийся архив LTspiceIV.exe, запустите его и следуйте инструкциям. Процесс установки обычно не вызывает проблем, просто надо соглашаться со всеми предложенными настройками.

Ссылки для загрузки программного обеспечения:

LTspice XVII для Windows 7, 8 и 10
LTspice IV для macOS 10.7+

"Руководство по началу работы" для LTspice можно загрузить по следующей ссылке: Руководство по началу работы Ltspice. После установки программы на рабочем столе появится иконка , а в верхней части меню Пуск появится строчка LTspice IV (рис. 1). Программу можно запустить как с помощью иконки на рабочем столе, таки через меню Пуск.

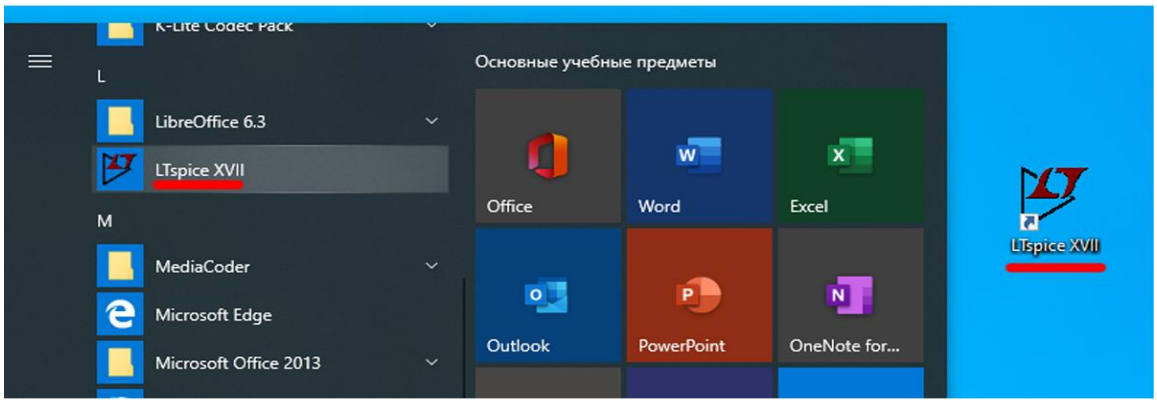


Рисунок 1: Ярлык программы LTspice

Запустите программу Ltspice, результате откроется стартовое окно программы (рис. 2), которое имеет:

- строку заголовка;
- панель команд;
- панель инструментов;
- рабочее поле;
- строку подсказки.

В левой части строки заголовка прописано название программы. В правой части этой строки сгруппированы стандартные кнопки управления размером окна и завершения программы. Цвет строки заголовка отражает активность окна, и если он синий, то окно активно, а если серый, то пассивно. На панели команд расположены стандартные меню, набор которых может меняться в зависимости от активного приложения, расположенного в рабочем поле программы. На панель инструментов вынесены иконки часто используемых команд и настроек. В рабочем поле располагаются под окна рабочих приложений, которыми являются редактор схем, плоттер, редактор символов и окно текстовых сообщений. В строке подсказки отображается текущее состояние программы и актуальные настройки симулятора. В рассматриваемый момент там отображается надпись Ready, которая говорит, что программа загружена и ждёт дальнейших действий пользователя.

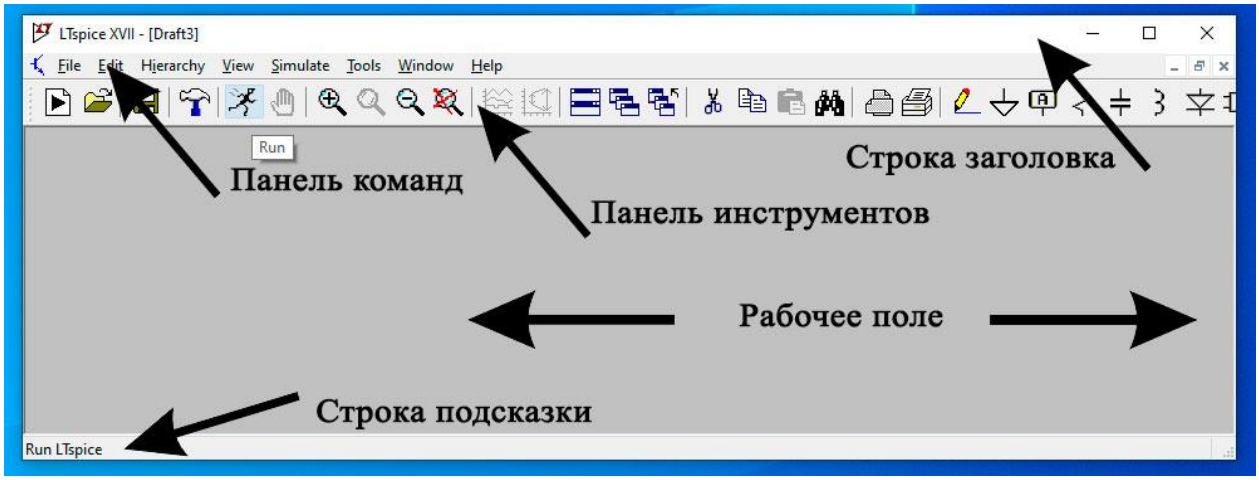


Рисунок 2: Подключение встроенного энкодера

Для выбора часто используемых компонентов, таких как резисторы, конденсаторы, индуктивности и диоды можно воспользоваться соответствующими иконками, расположенными на панели инструментов (рис.3).

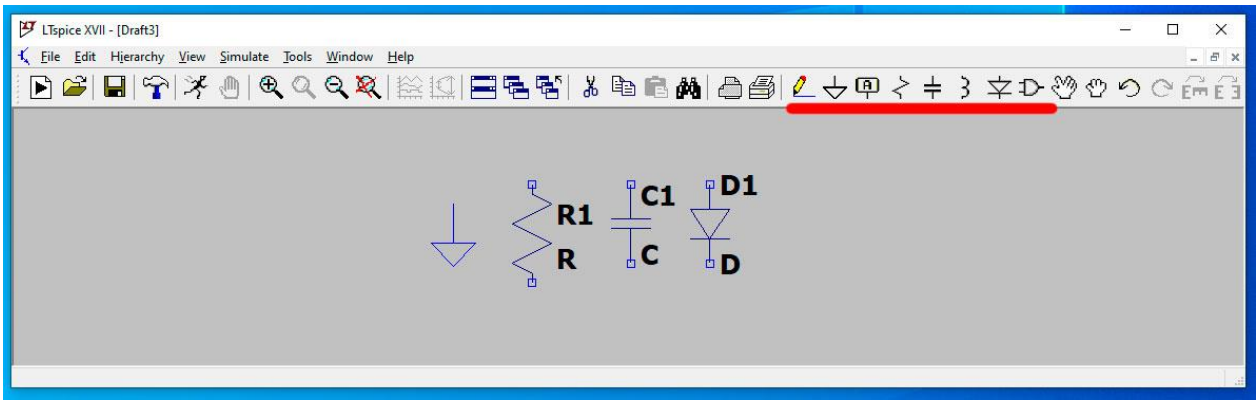


Рисунок 3: Размещение компонентов в окне редактора схем LTspice.

Для выбора остальных компонентов необходимо выполнить команду Edit (Редактировать) => Component (Компонент), которая вызывает диалоговое окно Select Component Symbol (рис. 4). Аналогичное действие производит нажатие функциональной клавиши <F2>, а также щелчок левой кнопкой мышки по иконке, расположенной на панели инструментов. Выбрав требуемый компонент, нужно щёлкнуть левой кнопкой мышки по кнопке OK, после чего компонент переносится на рабочее поле в окне редактора схем.

На схеме, в обязательном порядке, должен присутствовать компонент "Земля", с которым должны быть связаны все остальные компоненты. "Подвешенные цепи", то есть фрагменты схемы гальванически не связанные с компонентом "Земля" недопустимы. Данное требование по началу кажется чрезмерным, но реально не вызывает никаких особых проблем. Для рисования

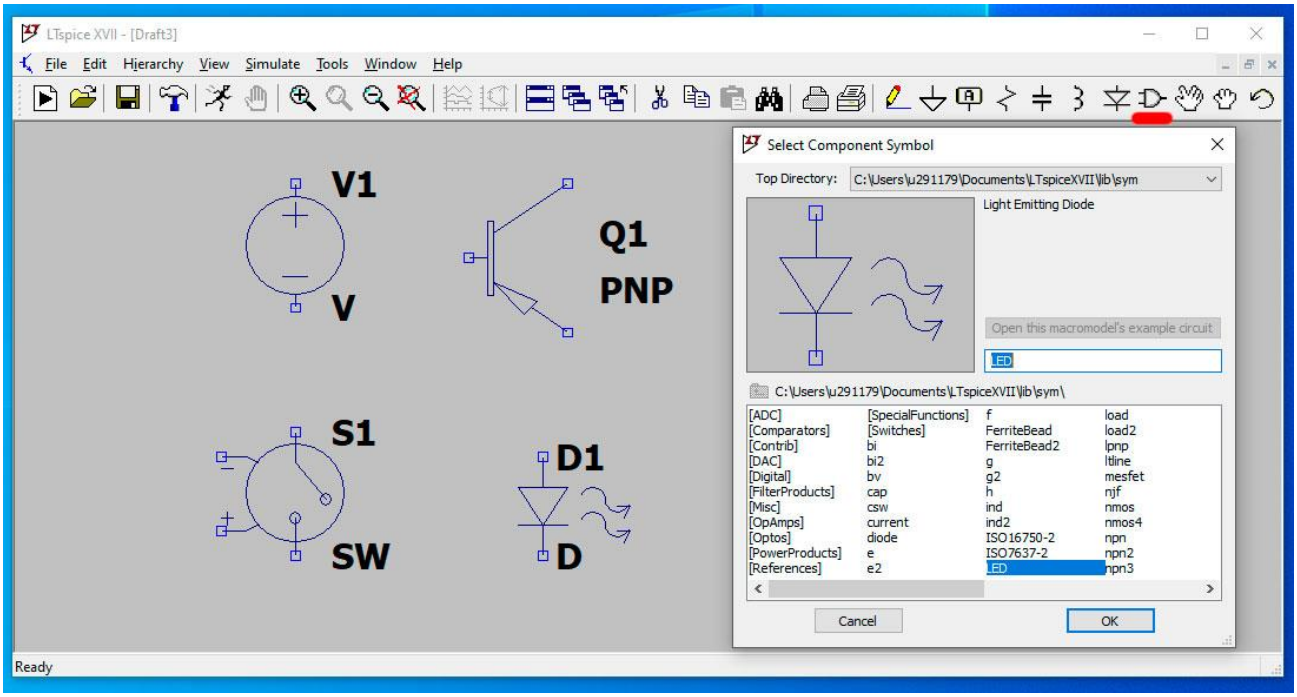


Рисунок 4: Выбор компонентов из диалогового окна Select Component Symbol

связей между компонентами необходимо активизировать иконку карандаша на панели инструментов или в меню Edit (Редактировать) на панели управления.

Для примера, составим схему делителя напряжения и получим ее временные диаграммы тока и напряжения.

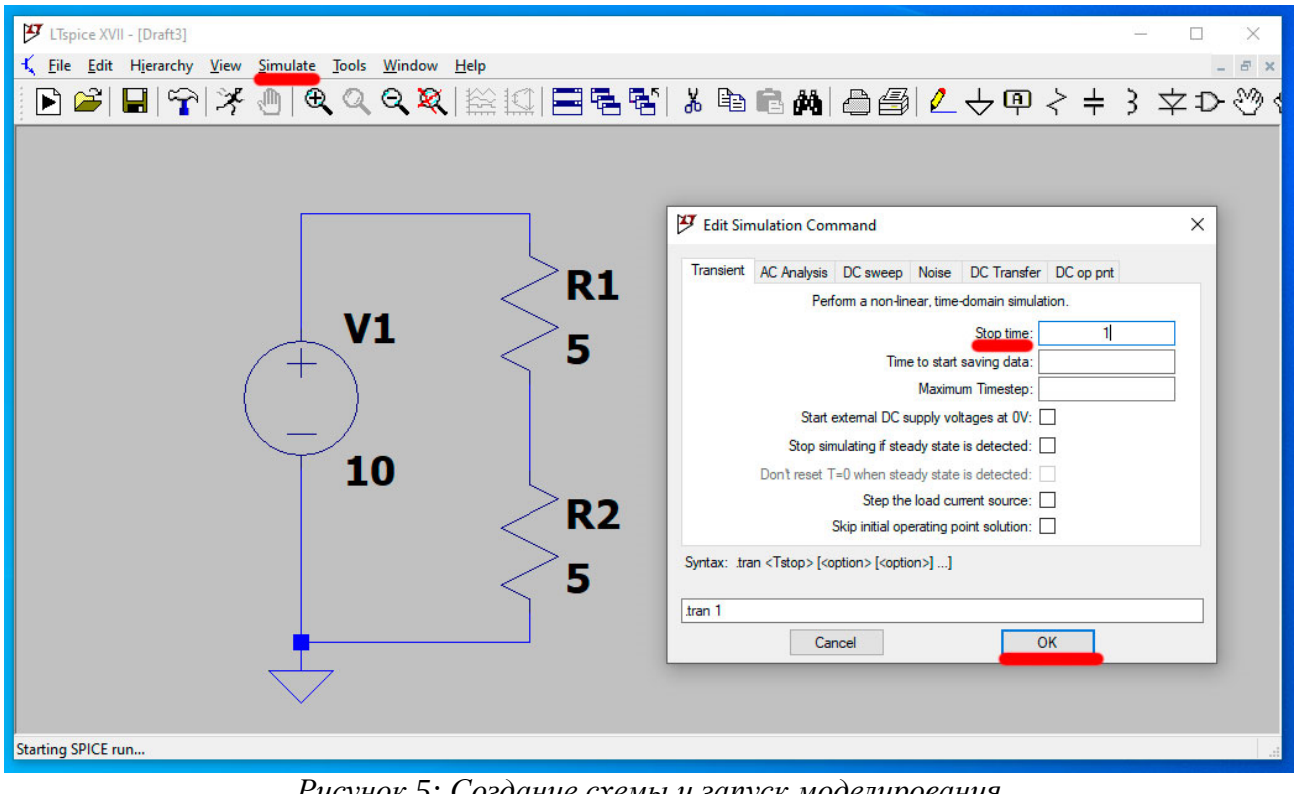


Рисунок 5: Создание схемы и запуск моделирования

Для этого построим схему в рабочей области программы LTspice и запустим процесс моделирования (симуляции). Для запуска необходимо воспользоваться

командой Simulate (Моделирование) => Run (рис. 5), или щёлкнуть левой кнопкой мышки по иконке Run на панели инструментов. После этого активизируется окно Edit Simulation Command (Редактирование команды моделирования), в котором, по умолчанию, выбрана вкладка Transient. В этой вкладке нужно определить время остановки анализа переходного процесса (Stop Time). Это необходимо сделать только при первом запуске моделирования. все последующие запуски производятся согласно установленным значениям.

При необходимости редактирования, окно Edit Simulation Command можно вызвать командой Simulate (Моделирование) => Edit Simulation Cmd (Редактирование команды моделирования) или щёлкнув правой кнопкой мышки по директиве .tran в рабочем поле редактора схем. Кроме времени окончания анализа процесса (Stop Time), в окне Edit Simulation Command, выбрав вкладку Transient, можно определить:

- Time to Start Saving Data - время начала записи результатов моделирования;
- Maximum Timestep - максимальный шаг интегрирования.

Если шаг интегрирования не указан, то программа сама выбирает для него максимально возможное значение.

Чтобы посмотреть форму напряжения на произвольном узле (проводнике) схемы, надо приблизить к нему курсор мышки. Оказавшись в районе проводника, курсор трансформируется в щуп красного цвета. Одновременно в строке подсказки, в левой нижней части окна, отображается приглашение щёлкнуть левой кнопкой мышки для вывода диаграммы напряжения на плоттер — Click to plot V(N001). После щелчка, в окне плоттера будет построена соответствующая временная диаграмма напряжения узла N001, которой будет присвоено название в виде выражения V(N001). Сама диаграмма и её название окрашены в одинаковый цвет, который выбирается автоматически, но может быть изменён

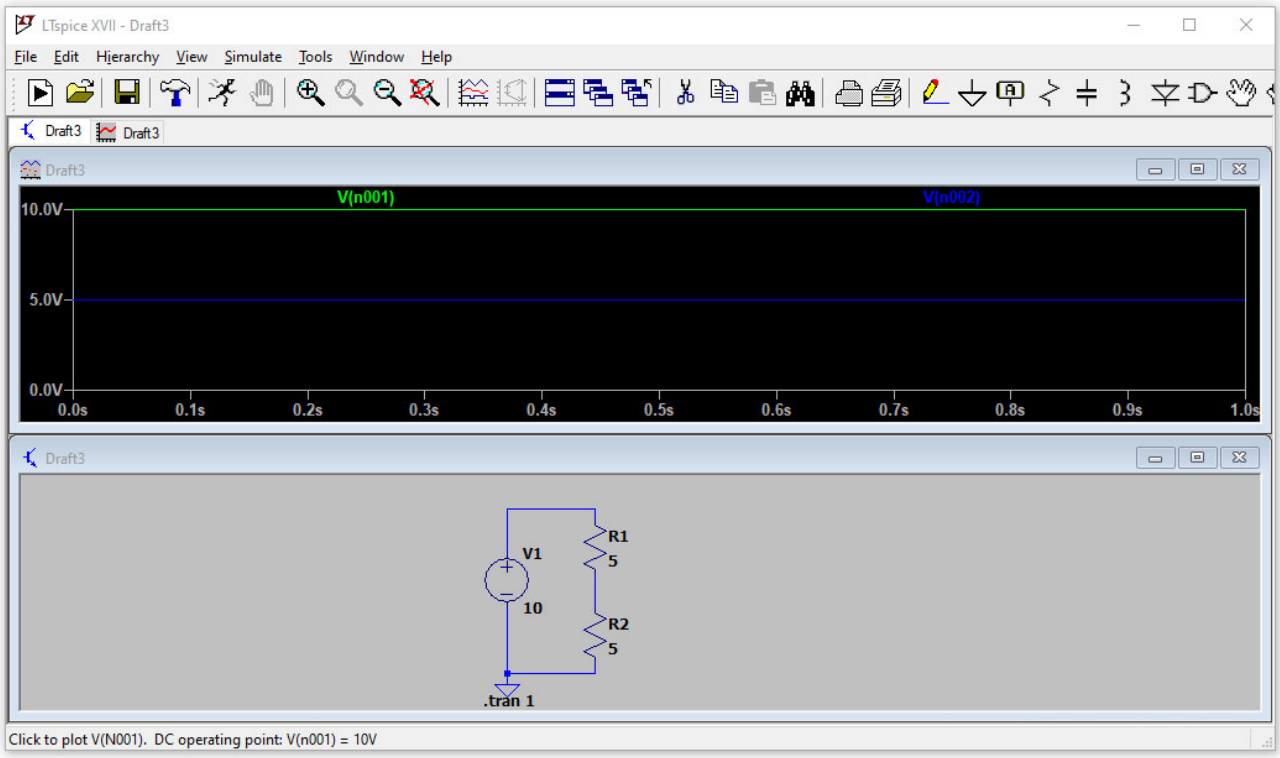


Рисунок 6: Создание схемы и запуск моделирования

вручную в окне Expression Editor (редактирование выражения). Это окно можно вызвать, щелкнув правой кнопкой мышки по названию диаграммы.

Таким образом мы получили временные диаграммы напряжений в схеме делителя напряжения (рис.6). Теоретически рассчитав значение напряжение на R2 мы получим. U(R2)=R2*(Uвх/(R1+R2))=5В. Расчетное значение совпадает с экспериментальным.

Задание

Установите программу LTspice на свой ПК, изучите интерфейс программы и смоделируйте схему делителя напряжения, получив временные диаграммы напряжений.

Порядок выполнения работы

1. Скачайте и установите LTSpice.
2. Изучите интерфейс программы.
3. Соберите схему делителя напряжения и запустите процесс моделирование.
4. Постройте временную диаграмму напряжений делителя напряжений.

Содержание отчета по работе

Отчет должен содержать снимки экрана выполнения этапов задания.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере LTSpice.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
3. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
4. Запустить симулятор и продемонстрировать работу программы.
5. Озвучить задание лабораторной работы.
6. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учеником.
2. Проверить работу модели в симуляторе – в случае ошибок, исправить вместе с учеником.

Лабораторная работа №23 «Исследование полупроводниковых диодов»

Цель

Ознакомиться с основами работы в пакете LTSpice, исследовать свойства полупроводниковых диодов.

Краткие теоретические сведения

Диод представляет собой двухслойную полупроводниковую структуру с двумя выводами, получившими названия анод и катод (рис. 1).

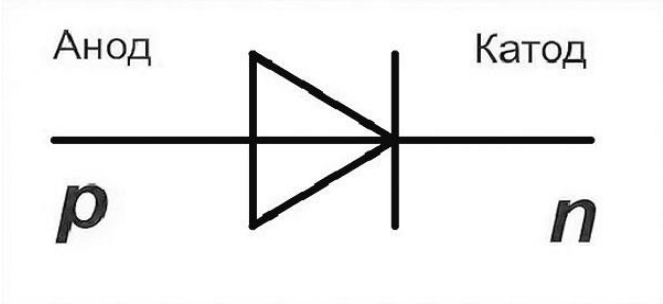


Рисунок 24: Условное обозначение диода

Параметры и режим работы диода определяются его вольт-амперной характеристикой, иллюстрирующей зависимость протекающего через диод тока I от приложенного напряжения U. Поэтому, когда говорят о снятии вольт-амперных характеристик (ВАХ) диода, то фактически идет речь о характеристиках p-n перехода. Если включить диод в электрическую цепь так, чтобы на аноде потенциал был выше чем на катоде, то говорят о прямом смещении p-n перехода (диода), и он находится в проводящем состоянии (прямая ветвь ВАХ). Если потенциал анода будет ниже чем катода, то диод смещен в обратном направлении, и он находится в не проводящем состоянии (обратная ветвь ВАХ). Типовая вольт-амперная характеристика прибора показана на рис. 2.

Качество работы диода в электрической схеме помимо статических характеристик определяется и частотными свойствами диодов. Применение низкочастотных диодов в высокочастотных электрических цепях приводит к их неработоспособности или к аварии. Диод является пассивным нелинейным элементом. Представляет собой p-n-переход, обладающий односторонней проводимостью. Характеризуется прямым током, обусловленным

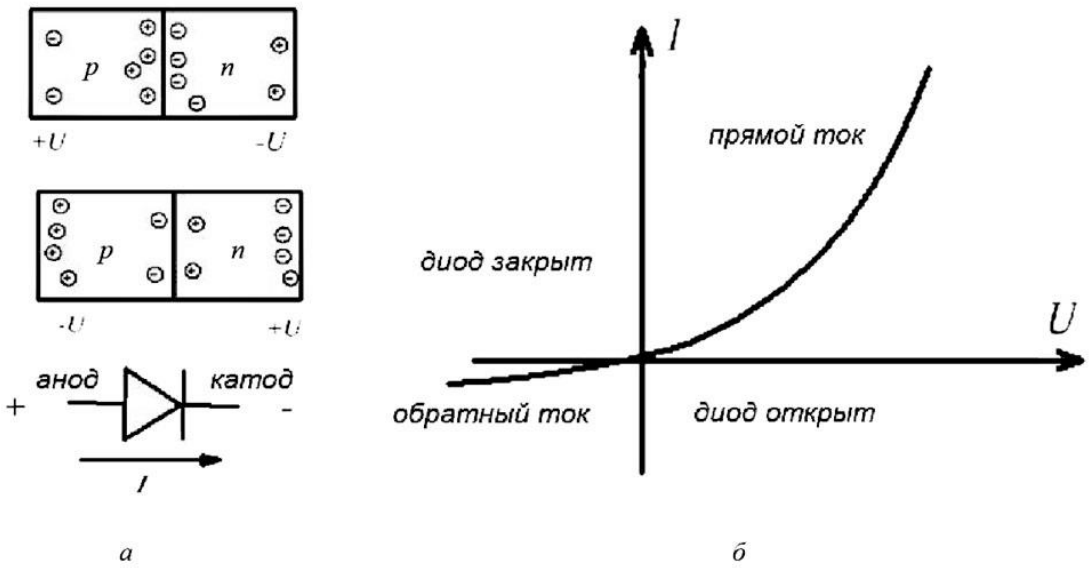


Рисунок 2: Полупроводниковый диод (а) и его ВАХ (б)

рекомбинацией основных носителей на границе p-n-перехода, и обратным током, обусловленным неосновными носителями. Напряжение, достаточное для открывания диода, составляет 0,6–0,7 В. Нелинейность зависимости тока и напряжения не позволяет использовать закон Ома для диодов.

Соберем в LTSpice схему для проверки работы диода в прямом и обратном включении (рис. 3). На левой схеме (рис. 3а) диод включен в прямом направлении. На аноде потенциал выше чем на катоде, диод находится в проводящем состоянии. Как видим из графика ток протекает. На правой схеме (рис. 3б) диод включен в обратном направлении. Потенциал анода будет ниже

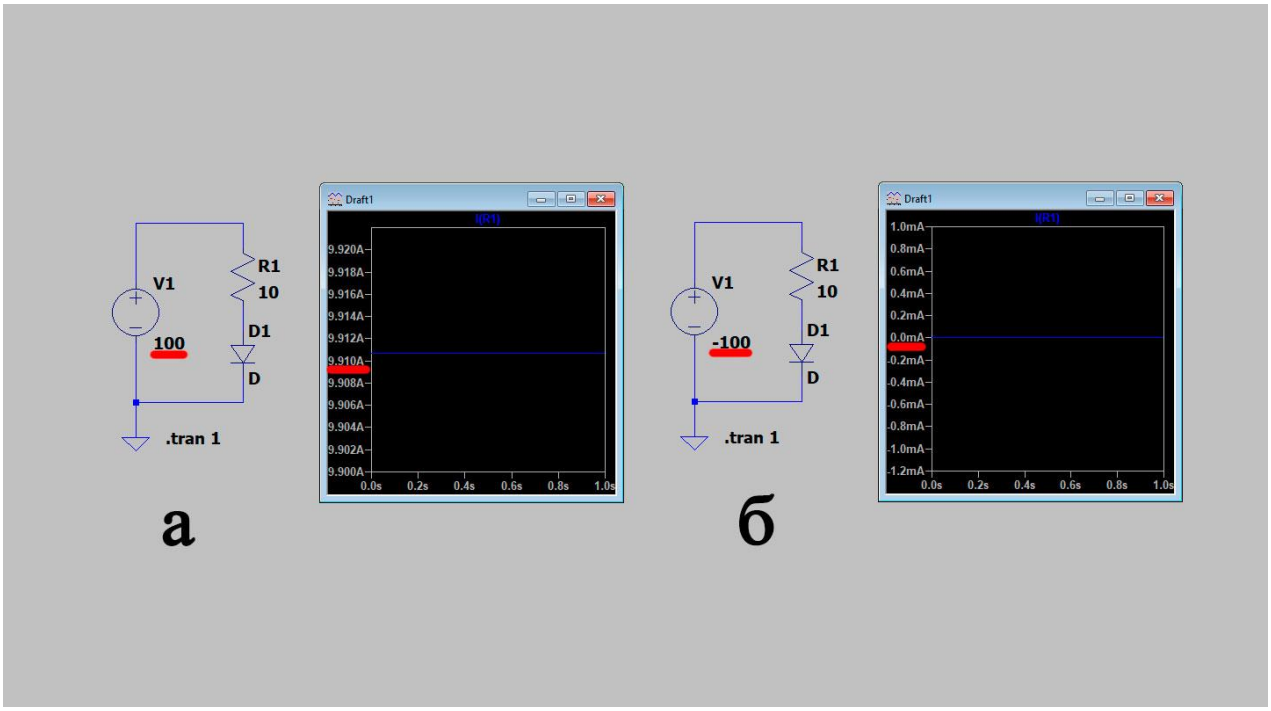


Рисунок 3: Схема для проверки диода в прямом (а) и обратном (б) включении.

чем катода, диод находится в не проводящем состоянии. Как видим из графика ток равен нулю.

Построим вольт-амперную характеристику (ВАХ) диода. Для этого соберем схему в LTSpice (рис. 4). Для построения ВАХ диода будем использовать DC — анализ по постоянному току. Эта директива задает расчет режима по постоянному току, при вариации параметров одного или нескольких источников постоянного напряжения или тока. Этот вид анализа полезен для расчета передаточных функций по постоянному току.

Для настройки и размещения директивы .DC можно использовать окно Edit Simulation Command (Редактирование команд моделирования), которое вызывается с помощью команды Simulate (Моделирование) | Edit Simulation Cmd (Редактирование команды моделирования). Для настройки анализа по постоянному току при вариации параметров выбираем вкладку DC sweep. На этой вкладке обнаруживаем три аналогичные вкладки: 1st Source (Первый источник), 2nd Source (Второй источник) и 3rd Source (Третий источник), необходимые для настройки трех возможных независимых источников. Для решении нашей задачи нам понадобится только первый источник. Для начала необходимо указать имя первого независимого источника V1. Далее

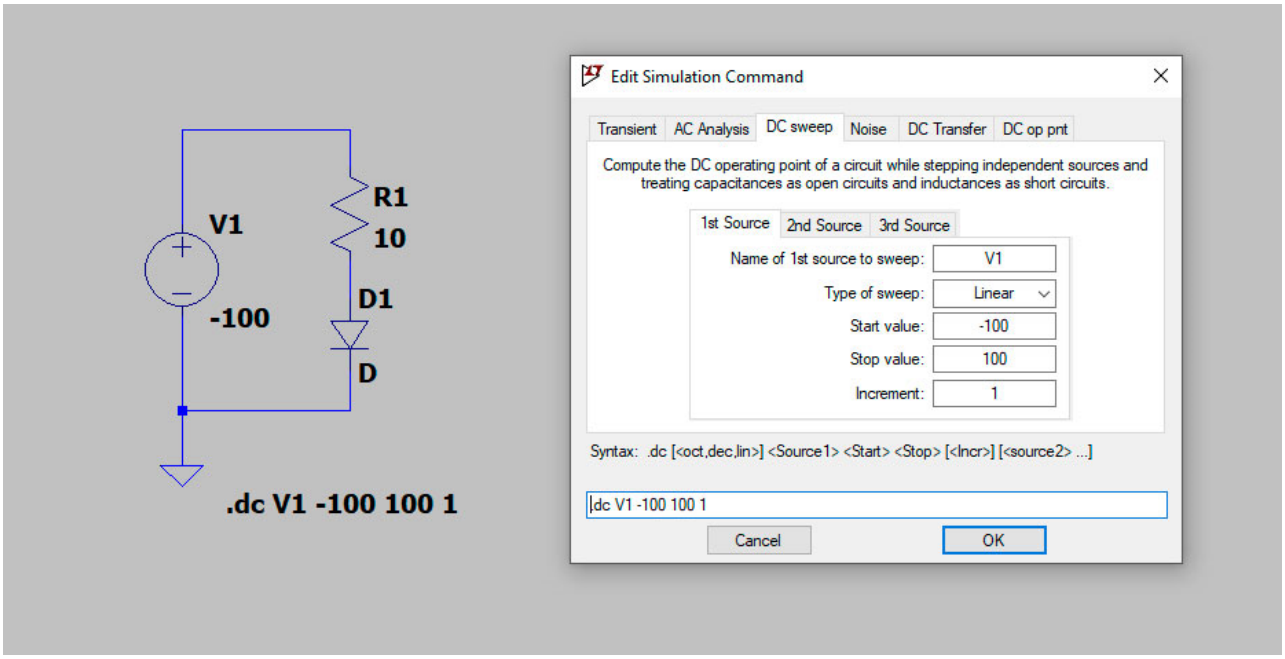


Рисунок 4: Построение ВАХ диода

устанавливаем стартовое значение напряжения источника (Start value) -100В, конечное значение напряжения источника (Stop value) 100В. Далее запускаем симуляцию с помощью меню Simulation | RUN. Смотрим протекающий через диод ток и получаем такой график ВАХ (рис. 5).

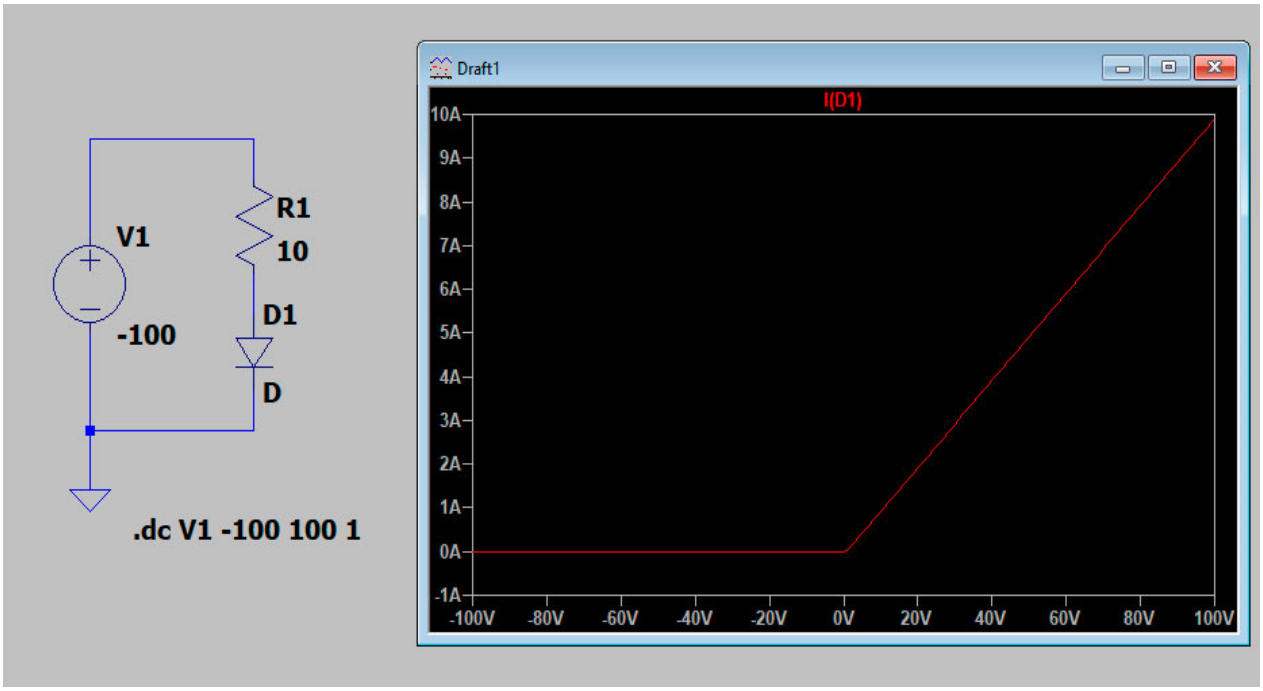


Рисунок 5: ВАХ диода.

Задание

Смоделируйте работу диода в LTSpice, проверьте его работу в прямом и обратном подключении, постройте вольт-амперную характеристику диода.

Порядок выполнения работы

1. Запустите LTSpice.
2. Постройте схему для проверки работу диода в прямом и обратном включении.
3. Убедитесь в правильности работы диода.
4. Постройте схему для построения ВАХ диода.
5. Задайте параметры анализа по постоянному току и запустите моделирование.
6. Получите график ВАХ диода.

Содержание отчета по работе

Отчет должен содержать снимки экрана выполнения этапов задания.

Инструкция по организации работы и проверке работы для преподавателя

Работа преподавателя организуется следующим образом:

1. Запустить на своем компьютере LTSpice.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
3. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
4. Запустить симулятор и продемонстрировать работу программы.
5. Озвучить задание лабораторной работы.
6. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учеником.
2. Проверить работу модели в симуляторе – в случае ошибок, исправить вместе с учеником.

Лабораторная работа №24 «Исследование однополупериодной схемы полупроводникового выпрямителя»

Цель

Изучение принципов построения однополупериодной схемы полупроводникового выпрямителя, построение временных диаграмм.

Краткие теоретические сведения

Для питания электронной аппаратуры, электродвигателей постоянного тока, электролизных и других установок возникает необходимость в выпрямлении переменного тока в постоянный. Под выпрямлением понимается процесс преобразования переменного тока в постоянный с помощью устройств, обладающих односторонней проводимостью (диодов). Выпрямительные устройства обычно состоят из трех основных элементов: трансформатора, электрического вентиля и сглаживающего фильтра (рис. 1).

Трансформатор Тр позволяет изменять значение переменного напряжения, получаемого от источника питания до значения требуемого выпрямленного напряжения. Блок вентилей В выполняет функцию выпрямления переменного тока. Для уменьшения пульсаций выпрямленного напряжения (тока) в цепи нагрузки Н применяют сглаживающий фильтр СФ. В случае управляемого выпрямителя необходим блок управления БУ, содержащий систему управления вентилями и систему автоматического регулирования уровня выходного напряжения. В неуправляемые выпрямители встраивают блок стабилизации СТ, поддерживающий номинальный уровень выходного напряжения или тока нагрузки при колебаниях напряжения сети и при изменении сопротивления

нагрузки. В зависимости от условий работы и предъявляемых требований к

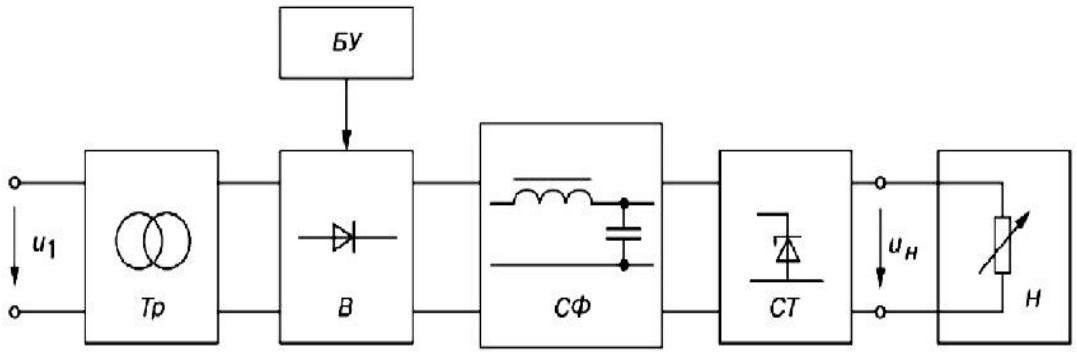


Рисунок 25: Структура выпрямительных устройств

выпрямителю отдельные его узлы могут отсутствовать.

Преобразование переменного тока в постоянный осуществляется с помощью нелинейных элементов с несимметричной ВАХ, односторонней проводимостью. Это свойство характерно для электровакуумных, ионных и полупроводниковых приборов. Мощность однофазных неуправляемых выпрямителей переменного тока колеблется от десятков до нескольких сотен ватт. В данной работе будут исследоваться выпрямители на полупроводниковых приборах, которые в настоящее время находят наибольшее применение.

Однофазная однополупериодная схема выпрямления (рис. 2) с активной нагрузкой является простейшей из известных схем выпрямления. Она состоит из силового трансформатора Тр, одного вентиля (диода) VD и нагрузки RH. Первичная обмотка трансформатора включена в сеть переменного тока с напряжением U1 к вторичной обмотке с напряжением U2 последовательно подключены диод VD и нагрузка резистор RH. Временные диаграммы

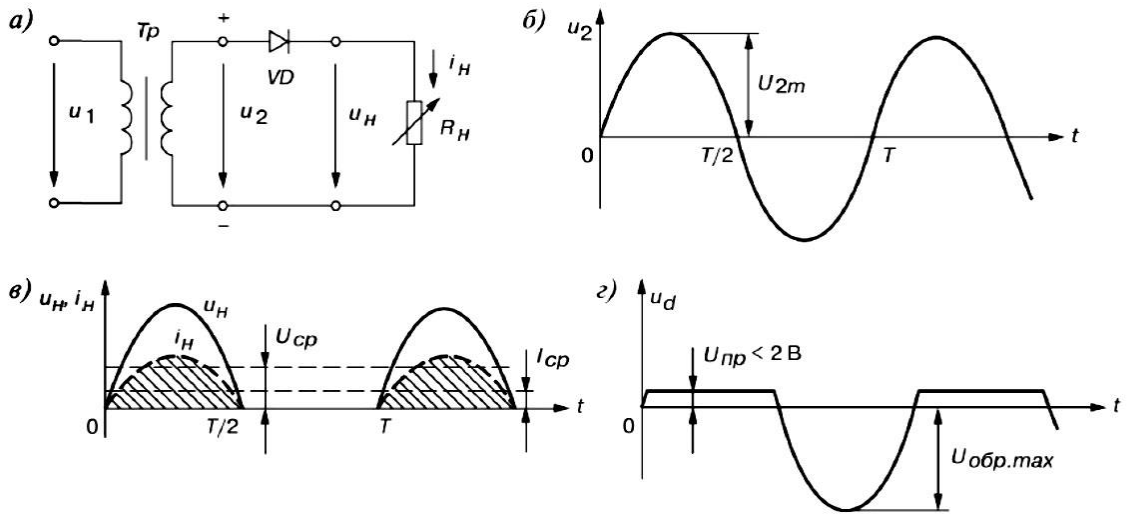


Рисунок 26: Схема однополупериодного полупроводникового выпрямителя

напряжения U2 вторичной обмотки трансформатора, напряжения на нагрузке UH и на вентиле Ud представлены на рис. 2 б, в и г.

Ток Iн в нагрузке протекает только при положительной полуволне вторичного напряжения U2 трансформатора, то есть когда напряжение на аноде диода более положительное, чем на его катоде. При этом напряжение на диоде Uпр<2В. При отрицательной полуволне U2 диод закрыт. Ток в нагрузке RH протекает только в один полупериод синусоидального напряжения, отсюда название выпрямителя — однополупериодный.

Однофазные полупроводниковые выпрямители используют для питания устройств, требующих малого тока и высокого напряжения.

Проверим работу однополупериодного выпрямителя в LTSpice. Для этого построим схему однополупериодного выпрямителя и исследуем ее временные диаграммы. Из рисунка 3 мы видим, что после диода остается только положительная часть синусоидального напряжения. Но постоянного напряжения

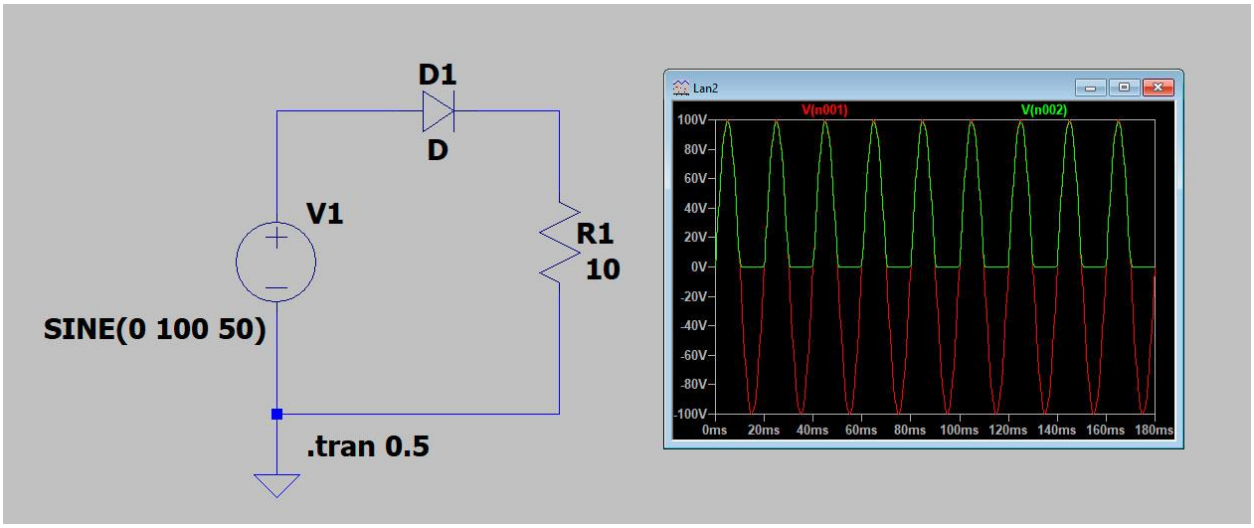


Рисунок 27: Схема и временная диаграмма однополупериодного выпрямителя без конденсатора

мы еще не добились. Для этого, чтобы сгладить полученное после диода напряжение, добавим к схеме фильтрующий конденсатор.

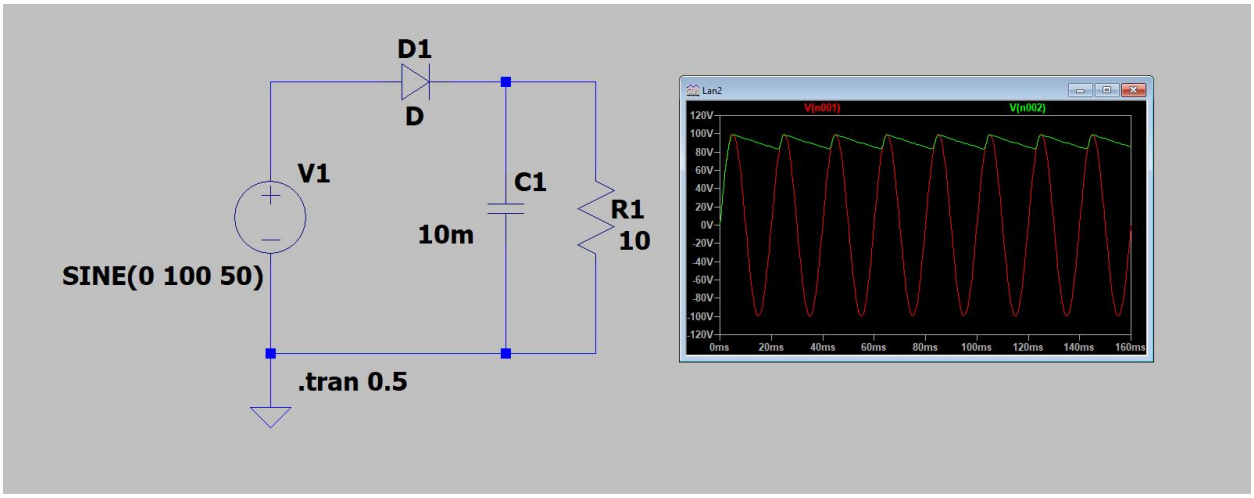


Рисунок 28: Схема и временная диаграмма однополупериодного выпрямителя с конденсатором

Как видно из рисунка 4 наличие конденсатора сглаживает выходное напряжение. Чем больше мы установим конденсатор, тем более низкий уровень пульсаций получим.

Задание

Смоделируйте работу однополупериодного выпрямителя в LTSpice, получите временную диаграмму его работы с фильтрующим конденсатором и без него.

Порядок выполнения работы

1. Запустите LTSpice.

2. Постройте схему для проверки работы однополупериодного выпрямителя.
3. Убедитесь в правильности его работы.
4. Постройте временную диаграмму для однополупериодного выпрямителя без конденсатора.
5. Добавьте в схему фильтрующий конденсатор и постройте временную диаграмму для однополупериодного выпрямителя с конденсатором.

Содержание отчета по работе

Отчет должен содержать снимки экрана выполнения этапов задания.

Инструкция по организации работы и проверке работы для преподавателя

1. Запустить на своем компьютере LTSpice.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
3. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
4. Запустить симулятор и продемонстрировать работу программы.
5. Озвучить задание лабораторной работы.
6. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учеником.
2. Проверить работу модели в симуляторе – в случае ошибок, исправить вместе с учеником.

Лабораторная работа №25 «Исследование двухполупериодной схемы полупроводникового выпрямителя»

Цель

Изучение принципов построения двухполупериодной схемы полупроводникового выпрямителя, построение временных диаграмм.

Краткие теоретические сведения

К недостаткам однополупериодной схемы полупроводникового выпрямителя можно отнести униполярный ток, который, проходя через вторичную обмотку, намагничивает сердечник трансформатора, изменяя его характеристики и уменьшая КПД; малое значение выпрямленного напряжения (Ucp ≈ 1/3*U2); высокий уровень пульсаций и большое обратное напряжение на диоде. Двухполупериодная схема выпрямителя исключает подобные недостатки.

Мостовая схема двухполупериодного выпрямителя (рис. 1) состоит из трансформатора Тр и четырех диодов, собранных по мостовой схеме. Одна из диагоналей моста соединена с выводами вторичной обмотки трансформатора, вторая диагональ — с нагрузкой RH. Положительным полюсом нагрузки является общая точка соединения катодов вентилей, отрицательным — точка соединения анодов. Временные диаграммы выпрямленного напряжения Uн и тока Uн приведены на рисунке 1б. В положительный полупериод синусоидального напряжения U2, когда точка 1 находится под положительным, а точка 2 — под отрицательным потенциалами, ток I2’ протекает через вентиль

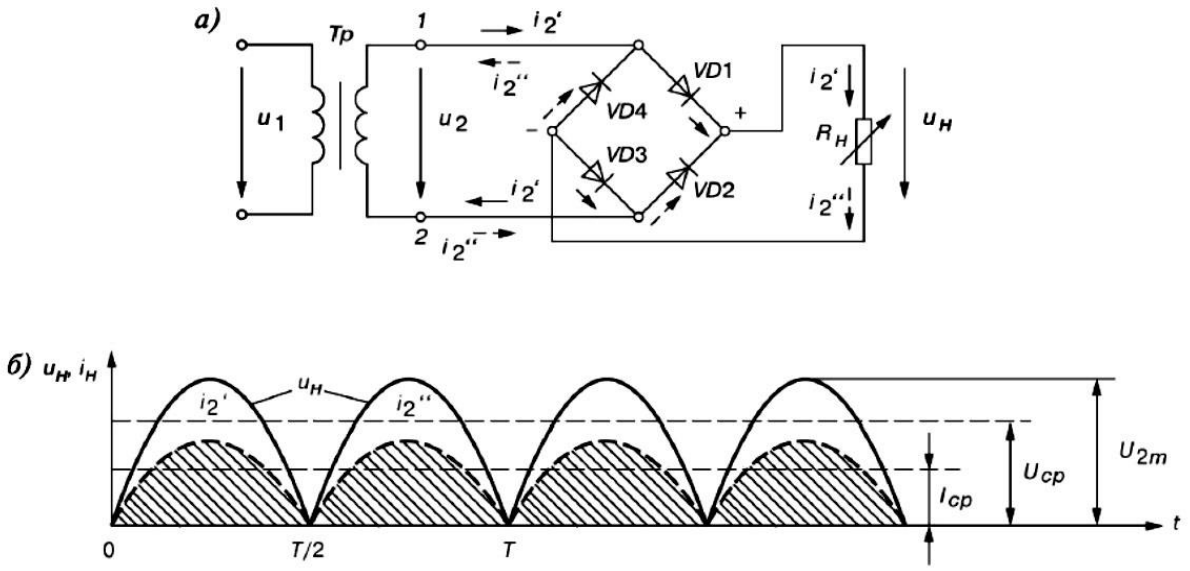


Рисунок 29: Схема двухполупериодного полупроводникового выпрямителя

VD1, сопротивление нагрузки RH и вентиль VD3. Вентили VD2 и VD4 в этот момент закрыты, так как находятся под обратным напряжением.

Во второй полупериод, когда в точке 1 вторичной обмотки отрицательный потенциал, а в точке 2 — положительный, ток I2" протекает через вентиль VD2, резистор RH и вентиль VD4 в направлении, указанном стрелками с одним штрихом. Вентили VD1и VD3 в этот момент закрыты, так как находятся под обратным напряжением. Таким образом, токи I2'и I2'', протекающие через нагрузку Rн, совпадают по направлению. Кривые напряжения и тока на нагрузке (см. рис. 1б) повторяют (при прямом напряжении на диодах Unp ≈ 0) по величине и форме выпрямленные полуволны напряжения и тока вторичной обмотки трансформатора. Они пульсируют от нуля до максимального значения U2m.

В двухполупериодной схеме выпрямления в сравнении с однополупериодной значительно лучше используется трансформатор, меньше коэффициент пульсации, хотя его величина остается значительной.

Проверим работу двухполупериодного выпрямителя в LTSpice. Для этого построим схему двухполупериодного выпрямителя и исследуем ее временные диаграммы. Из рисунка 2 мы видим, что после диодного моста остается все полупериоды синусоидального напряжения. Для того, чтобы сгладить

полученное после диода напряжение, добавим к схеме фильтрующий конденсатор.

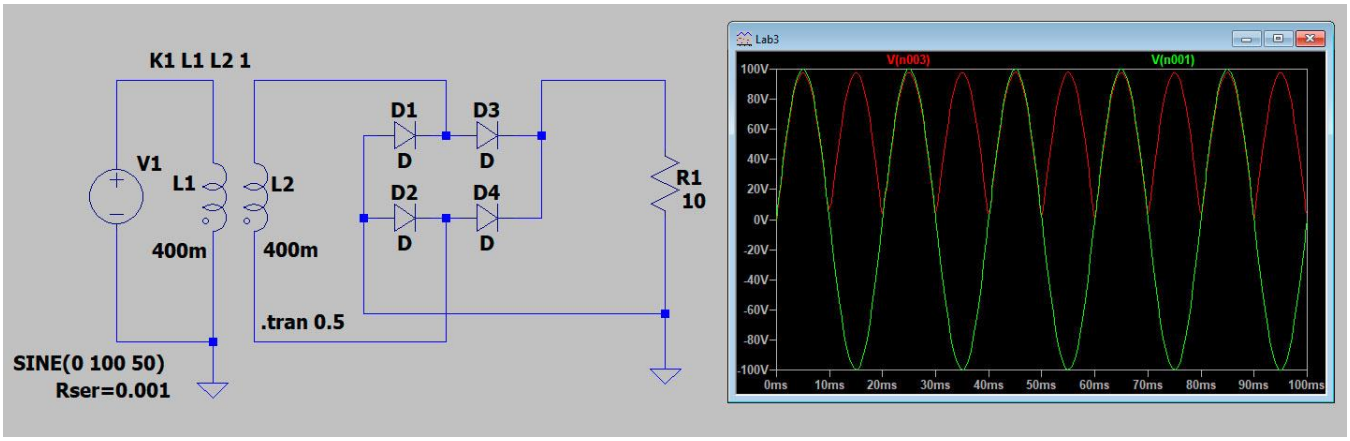


Рисунок 30: Схема и временная диаграмма двухполупериодного выпрямителя без конденсатора

Для того, чтобы сгладить полученное после диода напряжение, добавим к схеме фильтрующий конденсатор. Как видно из рисунка 3 наличие конденсатора сглаживает выходное напряжение. Чем больше мы установим конденсатор, тем более низкий уровень пульсаций получим.

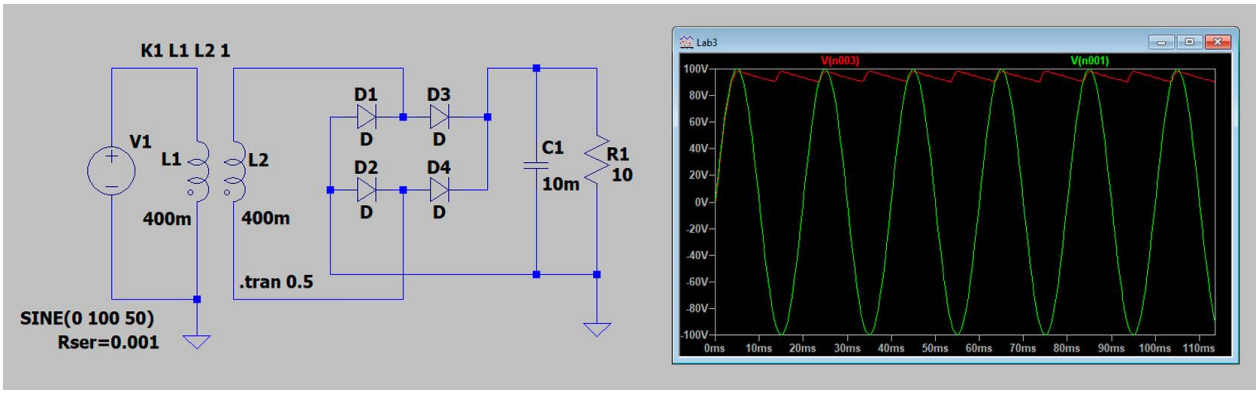


Рисунок 32: Схема и временная диаграмма двухполупериодного выпрямителя без конденсатора

Задание

Смоделируйте работу двухполупериодного выпрямителя в LTSpice, получите временную диаграмму его работы с фильтрующим конденсатором и без него.

Порядок выполнения работы

1. Запустите LTSpice.
2. Постройте схему для проверки работы двухполупериодного выпрямителя.
3. Убедитесь в правильности его работы.

4. Постройте временную диаграмму для двухполупериодного выпрямителя без конденсатора.
5. Добавьте в схему фильтрующий конденсатор и постройте временную диаграмму для двухполупериодного выпрямителя с конденсатором.

Содержание отчета по работе

Отчет должен содержать снимки экрана выполнения этапов задания.

Инструкция по организации работы и проверке работы для преподавателя

1. Запустить на своем компьютере LTSpice.
2. Открыть программу из раздела «Краткие теоретические сведения», отобразив экран своего монитора с помощью проектора на большом экране.
3. Изложить материал раздела «Краткие теоретические сведения» перед учениками и ответить на вопросы.
4. Запустить симулятор и продемонстрировать работу программы.
5. Озвучить задание лабораторной работы.
6. После завершения работы принять ее результаты.

Проверка работы преподавателя происходит следующим образом:

1. Изучить отчет по лабораторной работе на предмет ошибок. При их наличии – исправить совместно с учеником.
2. Проверить работу модели в симуляторе – в случае ошибок, исправить вместе с учеником.