Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Санкт-Петербургский государственный университет»
Кафедра системного программирования

Иванова Марина Андреевна

Инфраструктура для разработки и отладки алгоритмов движения роботов

Курсовая работа

Научный руководитель:
к.т.н., доц. кафедры СП Литвинов Ю.В.

Санкт-Петербург
2017

Введение

В настоящее время робототехника является востребованным направлением развития автоматизированных технологических систем, а также она активно используется в различных областях, таких как медицина, телекоммуникация, военное и промышленное дело, образование. Наиболее активно развивающейся её частью являются автономные мобильные системы, главная черта которых заключается в самостоятельном определении местоположения, ориентировании и движении в пространстве. Примером использования мобильных систем являются различные робототехнические соревнования и дисциплины, такие как “Робофутбол”, “Роботлон”. В таких соревновательных дисциплинах участвуют различные возрастные команды от студентов до школьников. Но часто школьникам не хватает опыта программирования, что затрудняет их участие в таких соревнованиях, и остаются лишь простые с алгоритмической точки зрения состязания. Появилась идея создать новую соревновательную дисциплину “Танчики роботов” и программно-аппаратный комплекс, позволяющий разрабатывать школьникам более сложные алгоритмы простыми средствами реализации.

Основная суть состязаний “Танчики роботов” заключается в том, что у нас есть две или более команд, состоящих из нескольких мобильных роботов — трехколесных тележек, которые находятся на полигоне с различными препятствиями. Команды роботов пытаются поразить противника, попав своим лазерным лучом по датчикам на корпусе робота противника. Чтобы в этих соревнованиях могли участвовать школьные команды, возникла необходимость в разработке программного

обеспечения, средствами которого можно было бы выполнять довольно сложные задачи, такие как локализация роботов, поиск пути на карте, следование по траектории и низкоуровневое управление устройствами робота. В итоге появился студенческий проект, в рамках которого ведется разработка программно-аппаратной части стенда для дисциплины “Тачики роботов”.

Для взаимодействия частей стенда и роботов используется ROS — библиотека для программирования роботов, позволяющая работать распределенно, для локализации роботов на полигоне – программный пакет SSL-Vision, используемый в робофутболе для определения местоположения роботов. Общая архитектура стенда показана на рисунке 1 и представляет собой следующее: над полигоном находится видеокамера, которая передает данные в SSL-Vision. Далее полученная информация о координатах положения роботов и препятствий средствами ROS передается командам роботов. Сами роботы представляют собой трехколесную тележку, собранную из робототехнического конструктора ТРИК. На самих роботах находится программа, которая управляет поведением и действиями робота, и среда времени выполнения TRIK Runtime, отвечающая за аппаратную часть.

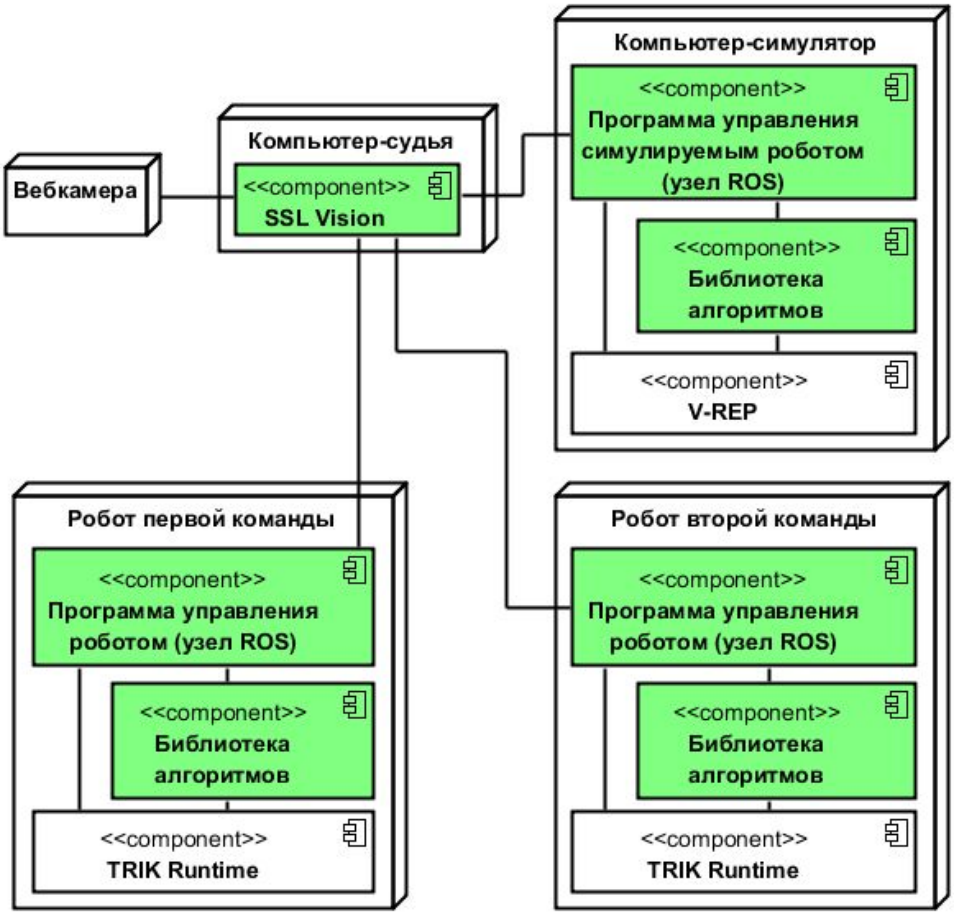

Рис 1: Архитектура стенда.

Изначально соревнования планировалось проводить на тележках, собранных из конструктора ТРИК, но так как данный конструктор является не очень доступным, было решено использовать 3D-симулятор, который позволит писать и отлаживать программы без доступа к реальному роботу. Чтобы управлять роботом в 3D-симуляторе, необходимо создать инфраструктуру и разработать недостающую функциональность.

Таким образом, данная работа посвящена конкретно алгоритмам управления движения роботов в среде для их моделирования и отладки.

1. Постановка задачи

Целью данной работы является построение инфраструктуры, позволяющей разрабатывать алгоритмы движения робота в 3D-симуляторе, в рамках студенческого проекта “Танчики роботов”.

Для этого были поставлены следующие задачи.

- Интеграция разрабатываемой системы с симулятором:
    - выбор симулятора
    - реализация средств интеграции
    - реализация недостающей функциональности в симуляторе
- Реализация метода, осуществляющего движение робота в заданную точку (следование по траектории, заданной набором точек).
- Апробация реализованных методов в симуляторе.

2. Обзор

2.1 Конструктор ТРИК

Робототехнический конструктор ТРИК — это набор, позволяющий проектировать различные модели роботов. Включает в себя контроллер с ARM-процессором и подключаемое к нему оборудование для считывания информации с различных датчиков (цифровых и аналоговых) и управления двигателями. Управление аппаратной частью осуществляется средой времени выполнения робота TRIK Runtime.

Модель робота, выбранная в симуляторе, должна быть аналогична реальной трехколесной тележке, собранной на базе робототехнического конструктора ТРИК.

2.2 ROS

ROS (Robot Operating System) — это фреймворк для программирования роботов, предоставляющий функциональность для распределенной работы. Данная система поддерживается программным обеспечением робототехнического конструктора ТРИК.

Библиотека ROS основана на архитектуре графов, где вычисления и обработка данных происходит в узлах, которые могут “общаться” между собой (передавая и получая сообщения).

Для того чтобы узлы могли обмениваться информацией, необходимо запустить roscore — ядро ROS, которое состоит из мастер-узла, потоков ввода/вывода и сервера параметров, который используется узлами для хранения и извлечения параметров во время работы. Задача roscore — это поиск и соединение работающих узлов ROS.

Для связи узлов необходим мастер-узел, который регистрирует запущенные узлы и предоставляет информацию о сервисах и топиках — названиях каналов, по которым будут передаваться сообщения. Перед тем, как передать сообщение в канал, узел должен оповестить об этом мастер-узел, а для получения сообщений узлам необходимо подписаться на каналы. Мастер-узел используется только для того, чтобы узлы могли узнать адреса друг друга, а дальше они передают сообщения уже на прямую.

2.3 Робосимуляторы

2.3.1 V-REP

V-REP — робосимулятор с интегрированной средой разработки, где каждый объект/модель может управляться с помощью встроенного сценария, плагина, узла ROS, клиента удаленного API. Таким образом, V-REP является средой, основанной на распределенной архитектуре управления (Рис. 2).

Управляющие программы могут быть написаны на языках C/C++, Python, Java, Lua, Matlab или Octave.

V-REP используется для быстрой разработки алгоритмов, моделирования автоматизации производства, быстрого прототипирования и верификации, робототехники, связанной с образованием, удаленный мониторинг. Часть внешнего интерфейса системы представлена на рисунке 3.

V-REP поддерживает 4 физических движка:

- Bullet Physics — движок реального времени;

- Open Dynamics Engine (ODE) — его основные компоненты это система обнаружения столкновений и система динамики абсолютно твёрдого тела;
- Newton — для имитации реального мира;
- Vortex Dynamics — в основном используется для задач, требующих высокой точности/производительности

Данный симулятор работает под операционными системами Windows, Mac, Linux.

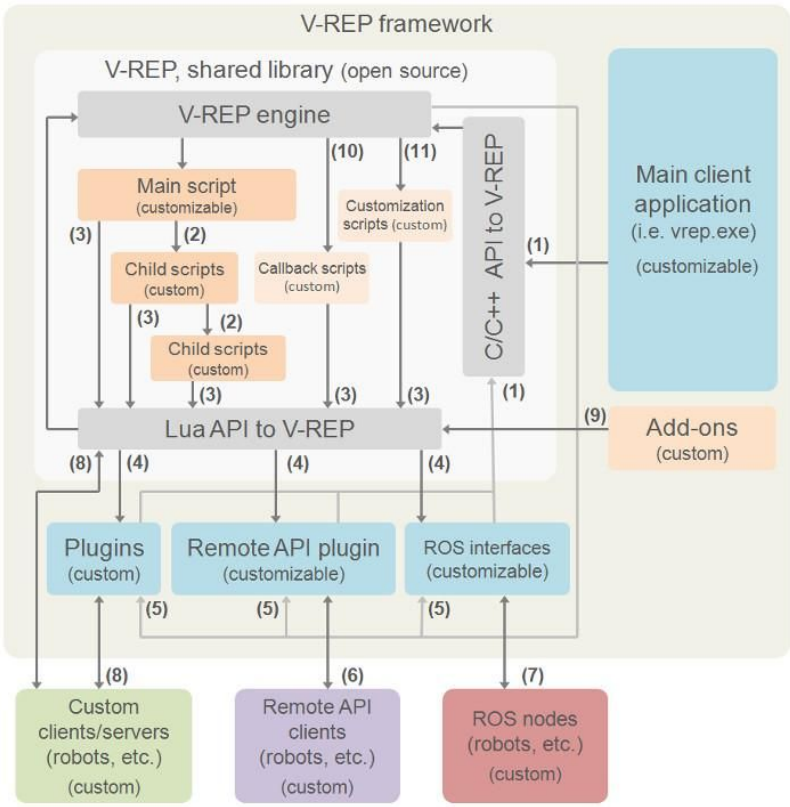

Рис 2:

V-REP. Цветные области являются настраиваемыми или могут быть настроены, [1]

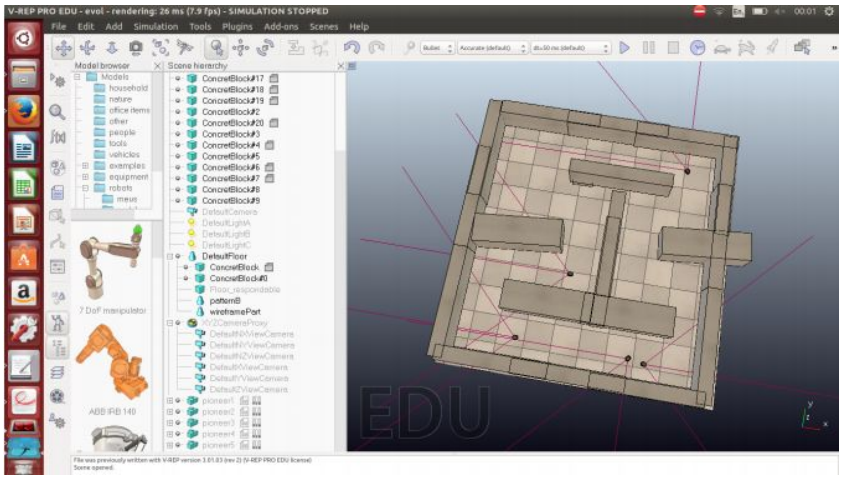

Рис 3: Интерфейс среды V-REP

2.3.2 Gazebo

Gazebo — робосимулятор, разработанный для операционной системы Linux. Приложение состоит из графической части (использует библиотеку OGRE), которая визуализирует модель окружающей среды и роботов, и имитационной части взаимодействия между твердыми телами (ODE), предназначенной для моделирования динамики твёрдых тел и включающей в себя функции соединения, обнаружение столкновений, инерционность (Рис. 4). Часть внешнего интерфейса системы представлена на рисунке 5.

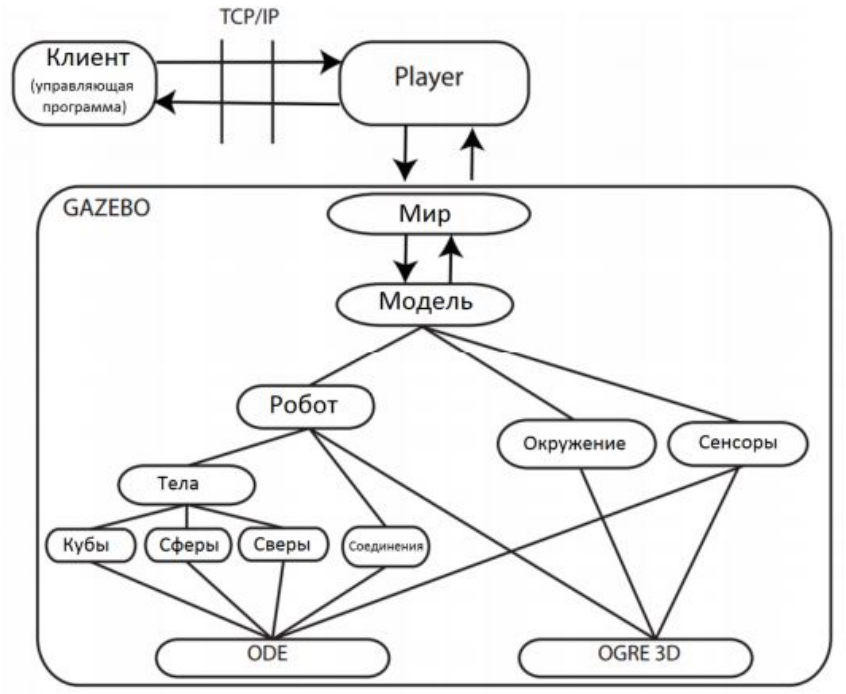

Рис 4: Диаграмма показывает связь между Клиентом, Player и Gazebo [2]

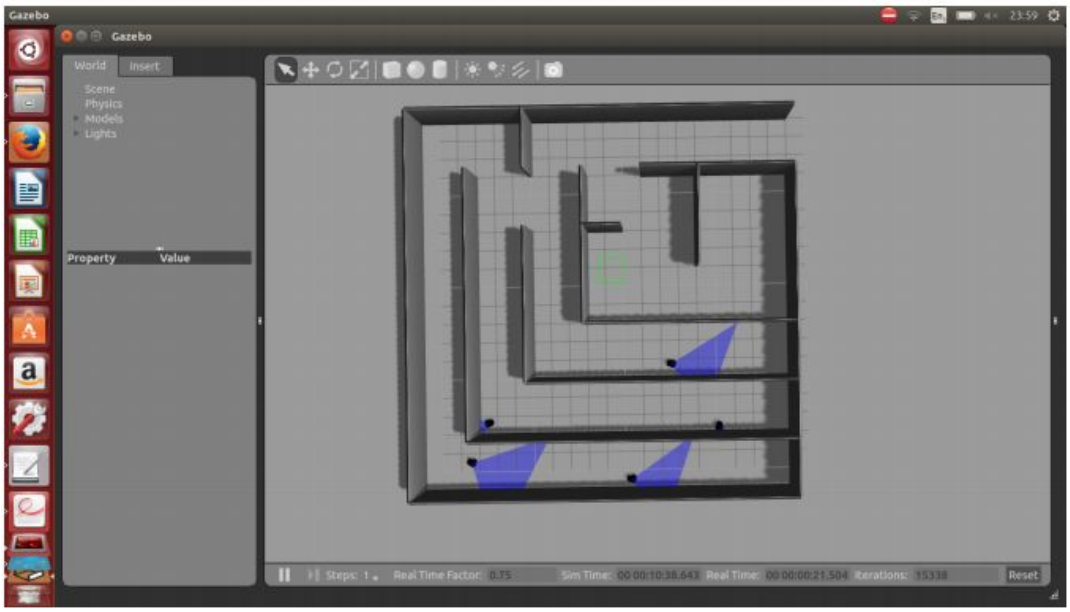

Рис 5: Интерфейс среды Gazebo

2.3.3 Выводы

В качестве симулятора был выбран V-REP, так как он является менее аппаратно-зависимым, чем Gazebo, имеет больше дополнительных функций и более интуитивно понятный пользовательский интерфейс. Также данный симулятор работает во всех основных операционных системах и имеет открытый исходный код. Симулятор V-REP поддерживает эмуляцию физики действий робота, что позволяет более точно представить поведение робота в реальности.

3. Реализация

3.1 Интеграция с V-REP

В нашей среде каждый робот представляет собой отдельный узел, поэтому было решено разработать библиотеку, позволяющую управлять роботами, находящимися на сцене V-REP, с помощью ROS узлов.

В случае такого управления V-REP должен выступать в качестве узла ROS, тогда другие узлы, находящиеся на сцене, смогут общаться через сервисы, издателей и подписчиков ROS. V-REP поддерживает два способа управления через ROS: rosPlugin и rosInterface. В нашей реализации использовался второй подход, т.к. RosInterface поддерживает больше функций для управления сценой, объектами и моделями, и дублирует API C/C++, в отличие от RosPlugin, который является более старой версией и представляет собой более высокий уровень абстракции. Функциональность rosInterface включена через плагины, код которых

является открытым и при необходимости может быть адаптирован. Плагины загружаются при запуске V-REP, но операция загрузки будет успешна только в том случае, когда запущен roscore.

Для того, чтобы управление моделями на сцене происходило через узел ROS, необходимо запустить управляющую программу из V-REP, для этого следует закрепить за объектом или моделью на сцене V-REP скрипт, который представляет собой код на языке Lua. В нем, с помощью стандартных функций V-REP можно вызвать нужный бинарный файл, который будет управлять моделью робота на сцене симулятора. Также в этом скрипте необходимо инициализировать подписчиков и издателей для получения и отправки сообщений.

Запущенный в V-REP исполняемый файл должен инициализировать ROS, создавать узел и после этого подписчиков, которые будут получать сообщения от издателей, инициализированных в V-REP, и издателей, отправляющих сообщения узлам V-REP, которые подписаны на них.

В рамках работы был создан набор методов на языке C++ и Lua, позволяющий создавать ROS-узлы, инициализировать подписчиков и издателей для моторов, акселерометра, гироскопа и реализованного эмулятора энкодеров, а также обрабатывать полученные данные и преобразовывать их к нужному виду.

3.2 Реализация эмулятора энкодера

В симуляторе V-REP не существует реализации энкодера, поэтому возникла необходимость реализовать функциональность, позволяющую его эмулировать.

Так как мы рассматриваем трехколесную тележку, где два колеса являются управляемыми, а одно поддерживающим, то количество отсчетов

энкодера будет вычисляться только на управляемых колесах. Для этого на каждое управляемое колесо модели робота был добавлен динамический сустав — объект, который имеет две опорные рамки: первая является фиксированной, а вторая выполняет движение относительно первой рамки [1]. Динамический сустав может выполнять движение вместе с колесом, на котором он закреплен. Функции V-REP позволяют получить в радианах угол поворота такого сустава, который изменяется в диапазоне $[-pi, pi]$, и с помощью скрипта, написанного на языке Lua в среде V-REP, и функциональности ROS полученные данные можно передать в метод, написанный на языке C++, где его будет принимать подписчик, и далее его обработать.

Для того, чтобы робот мог выполнять движения, используя энкодеры, был реализован метод, в котором вычисление числа отсчетов энкодера производится по формуле:

$$encVal = \begin{cases} \dfrac{val}{2 * pi} + stepEncoder, val \geq 0 \\ \dfrac{(2 * pi - |val|)}{2 * pi} + stepEncoder, val < 0 \end{cases} \quad (1)$$

где val — значение угла в радианах, полученное от издателя из V-REP, и $stepEncoder$ — количество полных оборотов колеса, $encVal$ — значение энкодера.

3.3 Следование по траектории

Каждый робот имеет в своем распоряжении карту, на которой отмечены препятствия и роботы его команды. Перед тем, как планировать перемещение робота, необходимо научить его выполнять движение по заданным точкам, образующим в совокупности траекторию.

Будем рассматривать шасси модели нашего робота, как жестко закрепленную конструкцию, имеющую локальную систему отсчета, началом которой является центр робота, и которая движется относительно глобальных осей координат. Такую кинематическую модель предлагают авторы книги “Introduction to Autonomous Mobile Robots” Roland Siegwart и Illah Reza Nourbakhsh [2].

На данный момент все движение реализовано на основе энкодеров. Пройденное расстояние вычисляется по формуле

$$distance = encVal * pi * R \quad (2)$$

где R — радиус колеса.

Затем необходимо вычислить угол поворота относительно глобальной системы координат по формуле:

$$angle = 90 - atan((yRobot - yTo)/(xRobot - xTo)) \quad (3)$$

где $angle$ — угол, на который необходимо повернуть роботу, ($xRobot$, $yRobot$) — местоположение робота, (xTo, yTo) — точка, в которую должен прибыть робот.

Координаты интересующих нас точек и местоположение робота задается относительно глобальной системы координат. Так как начало координат локальной системы находится в центре робота, и необходимо переместить эту точку в заданную, значит мы можем рассматривать робота как просто точку. Таким образом, координаты интересующей нас точки в локальном смысле будут равны покоординатной разности между местоположением робота и координатами заданной точки в глобальном смысле.

В итоге движение строится так: сначала выполняется поворот робота так, чтобы он был направлен в сторону заданной точки, а затем

происходит движение по прямой линии в точку. Такое движение обеспечивает наиболее короткий путь.

Была проведена серия экспериментов для движения робота в заданную точку из начала координат глобальной системы.

Для апробации следования по траектории в симуляторе использовался физический движок Newton, который в отличие от других физических движков реального времени в основном уделяет внимание физической точности.

Каждый эксперимент проводился следующим образом. Программа запускалась несколько раз на одной и той же модели робота, затем для каждой серии запусков были получены данные о местоположении робота при помощи координатной сетки V-REP, которые характеризуют результат работы программы.

На основании полученной в эксперименте группы результатов наблюдений оценим меру его приближения к истинному значению. В основу оценки была заложена методика, представленная в [10], где оценка истинного значения $\overline{X}$ рассчитывается как среднее значение полученных результатов при проведении 100 независимых опытов в каждой серии экспериментов. Оценка среднеквадратического отклонения вычисляется по формуле:

$$s_X = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2} \quad (4)$$

где Xi — полученные данные при эксперименте.

Ниже представлена таблица, содержащая результаты оценки среднеквадратического отклонения результатов движения робота в точку.

Таблица 1: среднеквадратическое отклонения результатов в различных сериях экспериментов.

серия экспериментов	координаты заданной точки в глобальной системе, м	среднеквадратическое отклонения результатов, м	
		x	y
1	(1.318; 0.300)	0.057	0.017
2	(1.318; 0.000)	0.003	0.000
4	(0.000; 0.300)	0.002	0.000
5	(0.000; -0.200)	0.002	0.000
6	(-0.182; -0.200)	0.003	0.002

Таким образом, мы видим, что при перемещении на небольшие расстояния движение робота получается более точное.

4. Результаты

На данный момент получены следующие результаты.

- В качестве робосимулятора выбран V-REP и произведена интеграция с разрабатываемой системой.
- Для симулятора выбрана трехколесная модель робота, схожая с реальной трехколесной тележкой ТРИК, которую в дальнейшем планируется использовать в дисциплине “Танчики роботов”.
- Для выбранной модели робота в симуляторе реализован эмулятор энкодера, а также в эту модель добавлены акселерометр и гироскоп.
- С использованием кинематической модели, описанной в [3], реализовано движение робота с помощью энкодеров по траектории, заданной набором точек.

- Результаты работы были представлены на конференции СПИСОК-2017.
- Исходный код размещен на веб-сервисе для хостинга IT-проектов GitHub ttps://github.com/yurii-litvinov/TrikABCL/tree/lib/libMove/distEncoder и https://github.com/yurii-litvinov/TrikABCL/tree/v-rep.

5. Список литературы

[1] Coppelia Robotics V-REP: Create. Compose. Simulate. Any Robot. — URL: http://www.v-rep.eu/ (дата обращения: 22.05.2017).

[2] Gazebo — URL: http://gazebosim.org/ (дата обращения: 22.05.2017).

[3] Roland Siegwart, Illah Reza Nourbakhsh. “Introduction to Autonomous Mobile Robots” Massachusetts Institute of Technology. 2004 - 321 с.

[4] Программируем роботов — бесплатный робосимулятор V-REP. Первые шаги / Блог компании MakeItLab / Geektimes — URL: https://geektimes.ru/company/makeitlab/blog/288496/ (дата обращения: 22.05.2017).

[5] V-REP — гибкая и масштабируемая платформа для робомоделирования. Продолжение статьи / Geektimes — URL: https://geektimes.ru/post/264666/ (дата обращения: 22.05.2017).

[6] Филиппов С.А. Уроки робототехники. Конструкция. Движение. Управление. — Лаборатория знаний, 2017. — 176 С.

[7] “Обзор и сравнение коммерческих и открытых программных комплексов для моделирования робототехнических систем” Гайнетдинов

А.Ф. — Молодежный научно-технический вестник # 09, 2015 – URL: http://sntbul.bmstu.ru/doc/800839.html (дата обращения: 22.05.2017).

[8] “Comparative Analysis Between Gazebo and V-REP Robotic Simulators” Lucas Nogueira, School of Electrical and Computer Engineering Universidade de Campinas

[9] “Методы оценки положения объекта в пространстве” Жуков Р.В. МГТУ им. Н.Э. Баумана — Молодежный научно-технический вестник # 10, 2013 – URL: http://sntbul.bmstu.ru/doc/636938.html (дата обращения: 22.05.2017).

[10] Д.А.Иванников, Е.Н.Фомичев “ОСНОВЫ МЕТРОЛОГИИ И ОРГАНИЗАЦИИ МЕТРОЛОГИЧЕСКОГО КОНТРОЛЯ” Учебное пособие. — Нижний Новгород: Нижегородский государственный технический университет, 2001. — 116 с.

[11] S. Zickler et al. SSL-vision: The shared vision system for the RoboCup Small Size League. — Robo Cup 2009: Robot Soccer World Cup XIII. — 2010. — pp. 425-436.