МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет Механико-Математический

Кафедра Программирования

Направление подготовки Математика и Компьютерные Науки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Новокрещёнова Льва Александровича

Тема работы:

Разработка распределённой системы управления движением группы мобильных роботов

«К защите допущена»	**Научный руководитель**
Заведующий кафедрой,	к.т.н., с.н.с ИАиЭ
д.ф.-м.н., профессор	
Марчук А.Г. /……..	Котов К.Ю./……
(фамилия , И., О.) / (подпись, МП)	(фамилия , И., О.) / (подпись, МП)
«……»……20…г.	«……»……20…г.

Дата защиты: «……»……20…г.

Новосибирск, 2018

Реферат

Название работы: Разработка распределённой системы управления движением группы мобильных роботов
Страниц: 25
Ключевые слова: управление группой мобильных роботов, децентрализованное управление, ROS, bluetooth
Иллюстраций: 11

На данный момент одной из наиболее важных задач в сфере робототехники является разработка распределённых систем управления. Главным преимуществом таких систем перед централизованными является надёжность, отказоустойчивость и масштабируемость. Целью данной работы является разработка прототипа распределённой системы управления движением группы мобильных роботов. Автор сосредоточился на решении проблем совместного движения мобильных роботов, а также на проблемах распределённого управления такими группами. Решение данных проблем является первоочерёдными в задачах, где позиционирование мобильных роботов друг относительно друга является ключевым, например перемещение груза группой роботов или же исследование обширной территории.

Разработанная система представлена программно-аппаратным комплексом с модульной архитектурой. В качестве программной инфраструктуры для проекта была выбран фреймворк Robotics Operation System(ROS), а разрабатываемая система была представлена модулями, взаимодействующими между собой средствами ROS. Аппаратная часть представлена стендом, оснащённым камерами, а также группой автономных мобильных роботов, имеющих трёхуровневую архитектуру. Мобильные роботы, оснащённые радиомодулями bluetooth, получают команды с удалённого устройства, и для каждого момента времени при помощи радиомодулей определяют расстояния друг друга, обмениваются актуальными данными между собой и за счёт полученных данных сохраняют построение при групповом перемещении.

По окончанию работ был проведён ряд экспериментов, подтверждающих действенность разрабатываемой системы.

ВВЕДЕНИЕ

Стремительно развивающееся и удешевляющееся производство микроэлектроники с каждым годом позволяет достигать всё большей мобильности и автономности различных устройств. В то же время, востребованность информационных систем всё более широким кругом пользователей и постоянно растущие требования к их отказоустойчивости влечёт за собой потребность в переходе от централизованных систем к распределённым. Данные тенденции не могут не сказываться и на робототехнике, не только проецируя уже готовые решения из других сфер, но и порождая абсолютно новые способы применения. Разработка распределённой системы управления движением группы мобильных роботов является комплексной задачей, предполагает поиск эффективных программно-аппаратных решений, что накладывает дополнительные сложности, такие как возможная несовместимость оборудования, отсутствие исчерпывающей документации, а также помехами, вносимыми в работу системы внешней средой. Среди главных преимуществ таких систем можно выделить четыре главных:

- устойчивость системы как целого, при потерях отдельных её элементов;
- модульность, позволяющая заменять элементы на всех уровнях;
- масштабируемость системы за счёт добавления в неё новых элементов при необходимости;
- покрытие элементами системы большего объёма задач, в сравнении с одиночными элементами.

Несмотря на важность и общую заинтересованность данной задачей, отсутствуют решения, которые бы в полной мере удовлетворяли потребностям. Существующие системы либо малофункциональны и не выполняют необходимых требований, либо слишком сложны в развёртке и изучении, что негативно сказывается на процессе внедрения. Помимо всего этого, даже сложные системы распределённого управления группами мобильных роботов ещё далеки от выполнения необходимых задач.

Автор данной работы сосредоточился на решении проблем совместного движения мобильных роботов, а также на проблемах распределённого управления такими группами. Решение данных проблем являются первоочередными в задачах, где позиционирование мобильных роботов друг относительно друга является ключевым, например перемещение груза группой роботов или же исследование обширной территории.

Среди задач, решаемых в данной работе можно выделить:

- обзор существующих решений;
- обзор инструментальных средств и оборудования;
- разработка архитектуры аппаратного обеспечения;
- разработка архитектуры программного обеспечения;
- реализация аппаратного обеспечения;
- реализация программного обеспечения;
- тестирование системы.

1 ОБЗОР СУЩЕСТВУЮЩИХ РАБОТ В ОБЛАСТИ ГРУППОВОГО УПРАВЛЕНИЯ

1.1 Системы группового управления мобильными роботами

На данный момент существует ряд систем распределённого управления группами мобильных автономных робототехнических платформ, целью которых является совместное выполнение определённых действий группой мобильных устройств. Примером подобных систем является проект DARPA Scout [5]. Одним из недостатков в подобных системах при совместном движении являются проблемы с сохранением строя группы при перемещениях. Также возникает сложность в сохранении строя в случае нахождения в заранее неизвестной среде. На данный момент DARPA ведёт разработку новой системы группового управления мобильными роботами OFFSET [9]. Проект имеет военное назначение и целью его повысить уровень абстракции при управлении группой роботов в условии городской среды. К сожалению, результаты исследований на текущем этапе не являются публичными.

Другим примером группового управления мобильными роботами является проект Shooting Stars [8] компании Intel. Он собой группу роботов, оборудованных LED-освещением, квадророторных БПЛА(беспилотных летательных аппаратов), выполняющих различные перестроения. Позиционирование в данной системе осуществляется за счёт GPS, при этом стоит заметить, что БПЛА не оснащены камерами. Недостатком данной системы в контексте обозначенной в рамках данной работы задачи является централизованное управление оператором с земли [8].

1.2. Системы отслеживания положения

На данный момент существуют некоторые разработки в области систем отслеживания положения объектов. Такие системы представлены такими проектами, как vicon [12] и optitrack [13] и уже представляют достаточную точность для промышленного использования. К сожалению, каждый из них требует большое количество дорогостоящего оборудования. В первую очередь

таким оборудованием являются специализированные камеры, разрабатываемые исключительно для данных задач. А также сложное специализированное программное обеспечение.

В данной работе для определения положения роботов и глобального позиционирования группы роботов использовались средства ROS, в частности ar_track_alvar [14]. Данный модуль захватывает из видеопотока метки и по ним устанавливает координаты объектов.

1.3 Современные системы проектирования, моделирования и разработки робототехнических систем

С целью упростить исследования и разработку в области робототехники, целесообразно строить работу на основе уже имеющихся библиотек, позволяющих снизить сложность взаимодействия как с аппаратной частью мобильный роботов, так и программных модулей между собой. Существует целый ряд библиотек, призванных упростить подобные исследования, рассмотрим некоторые из них.

Mobile Robot Programming Toolkit (MRPT) [15] — Кросс-Платформенная библиотека с открытым кодом, предназначенная для исследований в области робототехники. Она включает в себя набор инструментов, предназначенных для решения задач одновременной локализации и картографирования, визуализации и манипуляции большими объёмами данных, обработки данных с сенсоров.

Microsoft Robotics Developer Studio (MRDS) [16] — среда управления робототехническими системами, разработанная для семейства операционных систем Windows. Среда поддерживает как свой собственный язык программирования Microsoft Visual Programming Language, так и другие: C#, Visual Basic .NET, JScript и IronPython. Данная система поддерживает большое число разнообразного оборудования.

Carnegie Mellon Robot Navigation Toolkit (CARMEN) [17] — набор свободного программного обеспечения для управления робототехническими платформами. CARMEN обладает модульной архитектурой и представляет

возможность управления мобильными платформами, получения данных с сенсоров, локализации, планирования пути и картографирования.

Player Project [18] — проект, предназначенный для разработки и исследований в области робототехники и сенсорных системах. Player работает на POSIX-совместимых системах и на Microsoft Windows, поддерживает широкий список оборудования, а также языки программирования C, C++, Python and Ruby.

Universal Real-time Behaviour Interface (URBI) [19] — программная платформа с открытым кодом, предназначенная для разработки и управления в робототехнических системах. URBI имеет распределённую модульную архитектуру UObject, а также имеет свой собственный скриптовый язык urbiscript.

Robotics Operation System (ROS) [7] Для разработки и апробации данного алгоритма было необходимо выбрать программную платформу, для удобного взаимодействия модулей. Выбор пал на ROS (Robotic Operation System). Данная система представляет собой фреймворк, содержащий в себе разные функциональные модули и позволяющий устанавливать взаимодействие между ними. Фреймворк был первоначально разработан в Лаборатории Искусственного Интеллекта Стэнфордского университета и на данный момент развивается и поддерживается силами сообщества Open Robotics Foundation.

ROS содержит в себе множество модулей, для выполнения широкого спектра задач в сфере робототехнике, таких как задача одновременной локализации и картографирования.

ROS ориентирован на unix-like системы, вместе с тем основными поддерживаемыми системами являются Ubuntu Linux/Debian Linux, а остальные, такие как MacOS X являются экспериментальными.

На данный момент под MacOS X 10.11.6 ROS работал крайне нестабильно, и вследствии этого была использована связка Ubuntu 16.04 и ROS Lunar.

Взаимодействие между модулями ROS реализовано по принципу подписок. Модуль может создавать топики(topic) — разделы которые может

писать сообщения определённого формата, другие же модули могут читать данное сообщение из раздела, и наоборот. При разработке собственных модулей можно воспользоваться готовыми форматами сообщений, или же сделать свои, описывая тип передаваемого поля сообщения, и его название.

2 РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ

2.1 Постановка задачи

В работе рассматривается группа мобильных роботов и их совместное перемещение с сохранением строя. (рис. 2.1.1) Положение каждого робота в пространстве характеризуется координатами центра масс в неподвижной декартовой системе. Будем использовать два целевых положения роботов: положение роботов друг относительно и положение в пространстве для каждого момента времени для всей группы роботов.

В качестве способа коррекции управляющего воздействия предполагается триангуляция на каждом роботе группы и определение отклонения от заданного строя.

Используя терминологию, приведённую авторами [4], опишем задачу. Обозначим одиночного мобильного робота R как функцию для каждого момента времени, зависящую от следующих параметров:

x_{exti}, y_{ext} — внешние координаты

$\mathfrak{g}$ — информация о построении

x_{int}, y_{int} — внутренние координаты

И внешние и внутренние координаты находятся в декартовой системе. Внешние координаты охватывают всё пространство, на котором происходит перемещение группы мобильных роботов, началом координат может являться любая, заранее заданная точка пространства. Внутренние координаты описывают расположение робота внутри группы, и началом координат является центр масс группы. $R(t) = R(x_{ext}, y_{ext}, \mathfrak{g}, x_{int}, y_{int})$

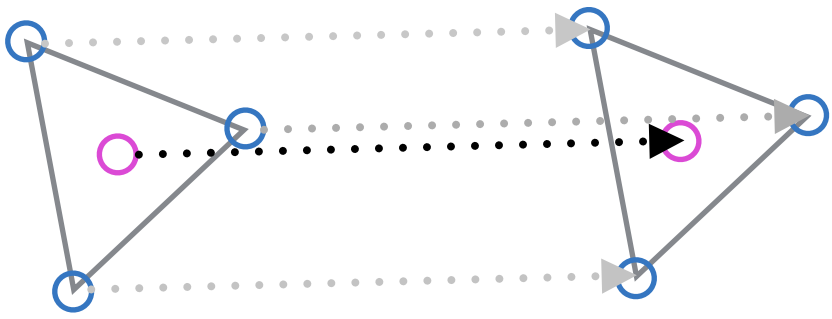

Рис. 2.1.1 Перемещение группы роботов с сохранением строя

Группу роботов обозначим $\mathfrak{R}(t) = \mathfrak{R}(R_1(t), R_2(t),\ldots,R_N(t))$, где $R_j \in \mathfrak{R}$ (j=1, …,N) — одиночный мобильный робот $R_j(t) = R_j(x_{ext}^j, y_{ext}^j, \mathfrak{g}, x_{int}^j, y_{int}^j)$.

Начальный момент для группы мобильных роботов $\mathfrak{R}^0 = \mathfrak{R}(t_0)$, а конечный момент — $\mathfrak{R}^f = \mathfrak{R}(t_f)$

Среда для каждого мобильного робота представлена функцией E(t). В совокупности для всех системы среда представлена

$\mathfrak{E}(t) = \mathfrak{E}(E_1(t), E_2(t), \ldots, E_N(t))$, где $E_i \in \mathfrak{E}$ (i = 1, …, N). Начальный момент времени для окружающей среды $\mathfrak{E}^0 = \mathfrak{E}(t_0)$, а конечный момент времени

$\mathfrak{E}^f = \mathfrak{E}(t_f)$

Для перемещения из начального момента в конечный, группа роботов осуществляет некоторый ряд действий. Для одного робота одно действие представлено как a_i (i = 1, …, N), а совокупность всех действий одного робота представлено $A = [a_1(t), a_2(t),\ldots, a_N(t)]^T$. Для группы мобильных роботов $\mathfrak{R}$ множество всех возможных действий $\mathfrak{A}$ представлено объединением множеств действий каждого мобильного робота $\mathfrak{A}=\{A_1\}\cup\{A_2\}\cup\{A_N\}$, где $A_j = [a_{1,j}(t),a_{2,j}(t),$ …, $a_{N,j}(t)]^T$ (j = 1, …, N) Система будет представлена дифференциальным уравнением, $\dot{\mathfrak{S}} = f(\mathfrak{R}, \mathfrak{E}, \mathfrak{A}, t)dt$

Задача состоит в программном нахождении действий для каждого робота, приводящих из начальной точки в конечную, с сохранением строя

$\forall R_j$ (j=1,…,N) | $\mathfrak{A}$

$d(R_{int}^0, R_{int}^f) < C$, $d(R_{ext}^0, p^0) < C$, $d(R_{ext}^f, p^F) < C$

$p^0 = (x^0, y^0)$ - начальная точка движения мобильного робота, выраженная глобальной координатой. $p^f = (x^f, y^f)$ - конечная точна движения мобильного робота, выраженная глобальной координатой. C - некоторая константа, которой ограничевается максимальное отклонение от заданной точки.

2.2 Архитектура системы

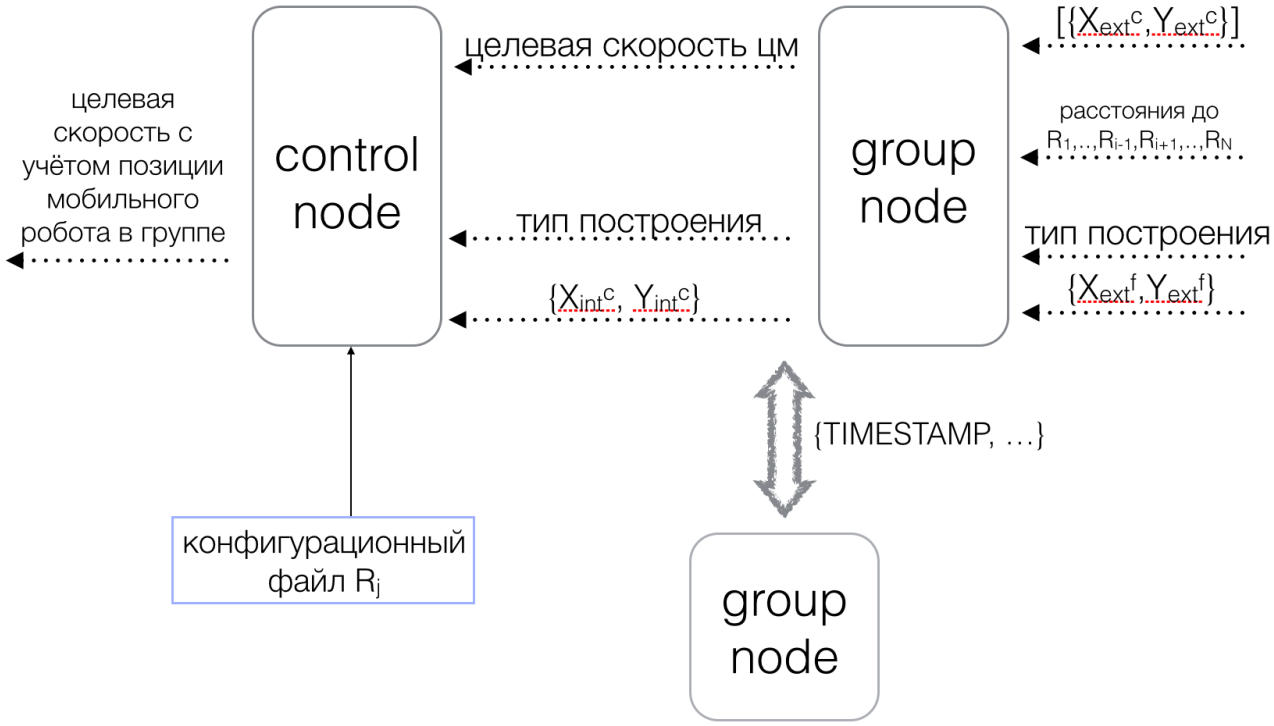

Рис. 2.2.1 Архитектура

Управление группой роботов осуществляется как единой системой и внешнее управление задаёт единую задачу для всей группы роботов. В данной работе задача была сведена к групповому перемещению, где группе роботов отдаётся целевая глобальная координата центра масс группы роботов, в которую группа должна переместиться из текущего положения. Для внешнего пользователя, отдающего команды, группа мобильных роботов неотличима от одного робота.

Архитектура системы выглядит следующим образом. (рис. 2.2.1) Команды группе отдаются с компьютера пользователя, через Rule node. Команда, вместе с системной информацией передаётся на group node какого-либо робота, реплицируется на всю группу и далее каждый group node передаёт необходимые данные на соответствующий control node. Каждый control node преобразует полученные с group node данные о перемещении центра масс системы и данные о положении мобильных роботов внутри группы и преобразует их в параметры

движения данного мобильного робота. Полученные данные непосредственно управляют перемещением в пространстве конкретного робота.

Далее для каждого модуля в начале будут описаны его входные и выходные топики, через которые осуществляется межмодульное взаимодействие, а также описание внутреннего устройства и принципа работы.

2.3 Rule node

выход: */group*

Данный модуль находится на управляющем устройстве пользователя и предоставляет интерфейс ввода целевой координаты для центра масс группы, а также информации о типе построения группы роботов, необходимых для выполнения данного задания. При отправке сообщения к нему также добавляется временная метка, соответствующая текущему времени системы управляющего устройства. Предполагается, что на всех устройствах время одинаковое, актуальное, и синхронизируется встроенными средствами их операционных систем.

2.4 Group node

вход: /group

выход: /group, /control

Виртуальный модуль является высокоуровневым слоем логики между конечными роботами и внешней системой управления.

На вход виртуальный модуль получает команду от внешнего управляющего, содержащую в себе временную метку(TIMESTAMP), координаты конечной точки (X_{ext}^{f}, Y_{ext}^{f}), расстояния между роботами, а так же тип построения.

Команда может быть получена как от управляющей системы, так и от равнозначного виртуального модуля другого робота. Решение об актуальности сообщения принимается на основании временной метки, соответственно более новые команды при получении вытесняют старые. Также в случае получения новой команды, и при отсутствии её в топике */group* модуль публикует данную команду в топик.

Внутри модуля происходят вычисления и по их результатам модуль делает запись в */control_i*, которая содержит в себе целевую скорость центра масс, тип построения, а также внутреннюю координату робота.

Алгоритм репликации команд в группе
команда_н — новая команда
команда_т — текущая команда данного мобильного робота

```
команда_н := считать /group
если команда_н != пусто то
    если команда_н.временная_метка < команда_т.временная_метка то
        команда_т = команда_н
    иначе записать команда_т в /group
```

Угловая скорость рассчитывается исходя их текущей и конечной координат, полученных на основании алгоритма, описанного в [3]. Полученные данные записываются в выходной топик /control

2.5 Internal visual node

вход: */group*

Данный модуль разрабатывался с целью визуализации данных (рис. 2.5.1) о расположении роботов внутри группы. Данная программная компонента представлена модулем ROS, подписанным на раздел */group* и визуализирующем данные из него при помощи графической библиотеки wx-python.

Модуль читает из */group* данные, полученные каждым роботом при сканировании окружающих их роботов группы и представляют собой массивы расстояний до каждого робота группы. Таким образом имея значения расстояния от каждого до каждого робота мы можем построить текущее относительное расположение всех роботов в группе. Также данным способом мы получаем расстояние между каждыми двумя роботами дважды, что позволяет нам усреднять данные для повышения точности.

2.6 BluetoothLE

В статье [5] Тор-Инге Кваксруд провёл исследования по определению расстояния между bluetooth LE адаптерами и получил корректные результаты. В рамках данной работы проделан ряд экспериментов для подобного определения данных о расстоянии между адаптерами. В первом случае в качестве адаптера на базовом устройстве взят адаптер ugreen bt4.0, а устройством, до которого измерялось расстояние был Mi Band 2 с Bluetooth Low Energy. Измеряемым параметром являлось уровень сигнала RSSI. Измерения проводились на расстояниях от 0 до 0.5м с шагом в 0.05м, в каждой точке проводилось 5 измерений. Второй эксперимент проводился между cc2650 и iPhone 5s. Измерения также проводились на расстояниях от 0 до 0.5м с шагом в 0.05м, в каждой точке также проводилось по 5 измерений. (рис. 2.6.1)

Построив графики для каждого из экспериментов можно увидеть логарифмическую зависимость уровня сигнала RSSI от расстояния, что совпадает с экспериментом, поведённым Тор-Инге Кваксрудом[5].

К сожалению, большинство бытовых bluetooth адаптеров не позволяет корректно получать данные об уровне сигнала до другого устройства, либо не позволяет получать таковые данные совсем. Решением данной проблемы может быть использование специализированных радиомодулей, таких как cc2510, выпускаемых компанией Texas Instruments. В своей статье [6] авторы приходят к выводу, что определение расстояния до объекта за счёт уровня сигнала может быть эффективным только в случае использования специализированного точного оборудования.

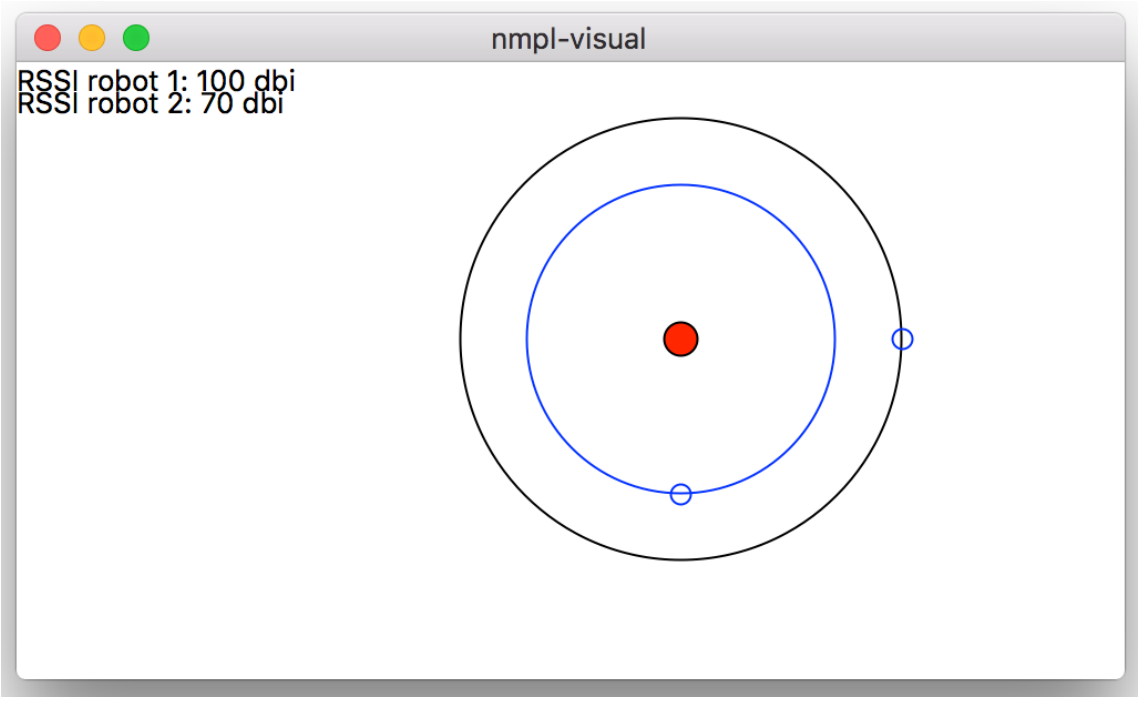

Рис. 2.5.1 Визуализация положения роботов в группе

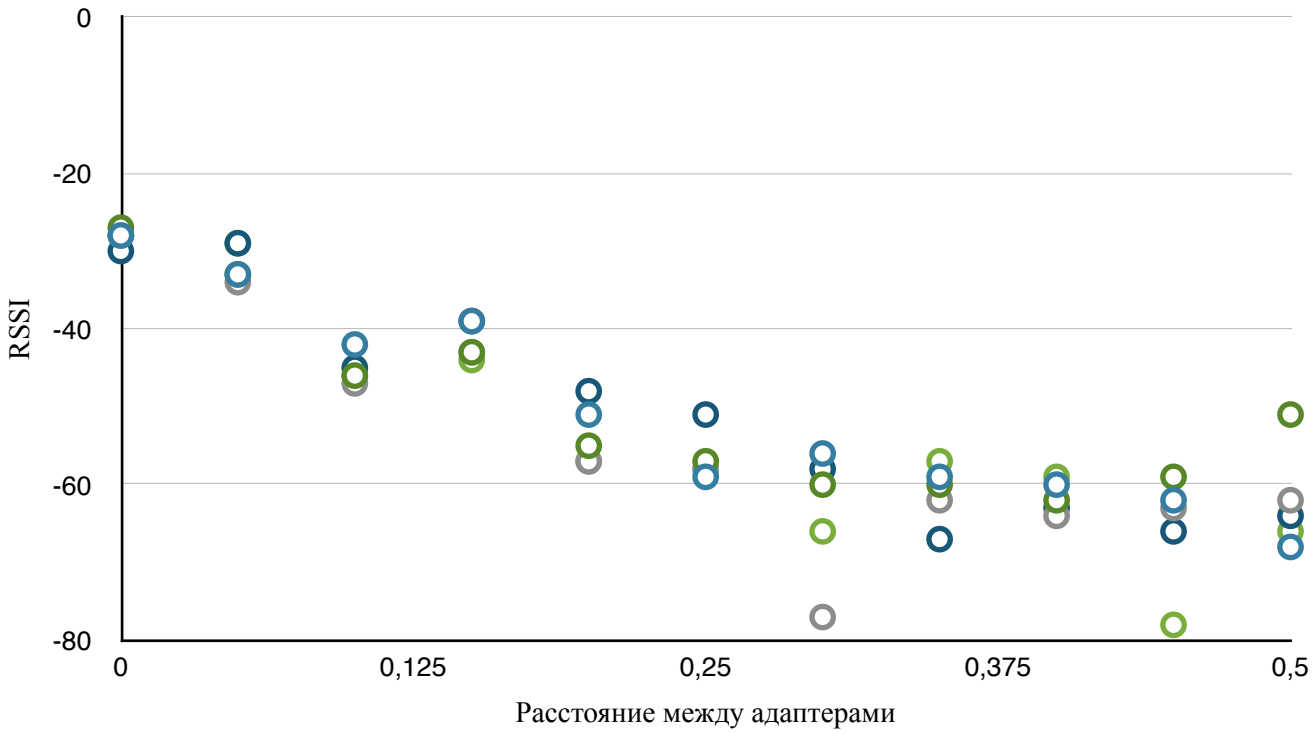

Рис. 2.6.1 График зависимости RSSI от расстояния между bluetooth адаптерами

2.7 Control node

вход: /control
выход: /driver

Данный модуль отвечает за преобразования общих данных группы, полученных из раздела */group* в данные, необходимые для управления перемещением одного мобильного робота.

На каждом мобильном устройстве содержится конфигурационный файл, в котором содержится информация о всех возможных построениях, и позиция данного робота в данном построении. На основании этих данных, и получаемых из */group* целевой скорости центра масс, типа построения, и внутренней координаты, вычисляется конечная целевая скорость данного робота и пишется в раздел */driver_i*

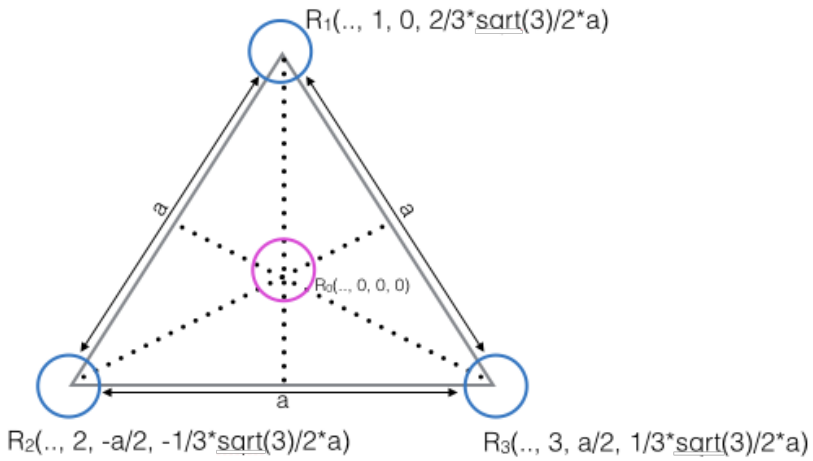

Рис. 2.7.1 Целевое положение роботов в группе

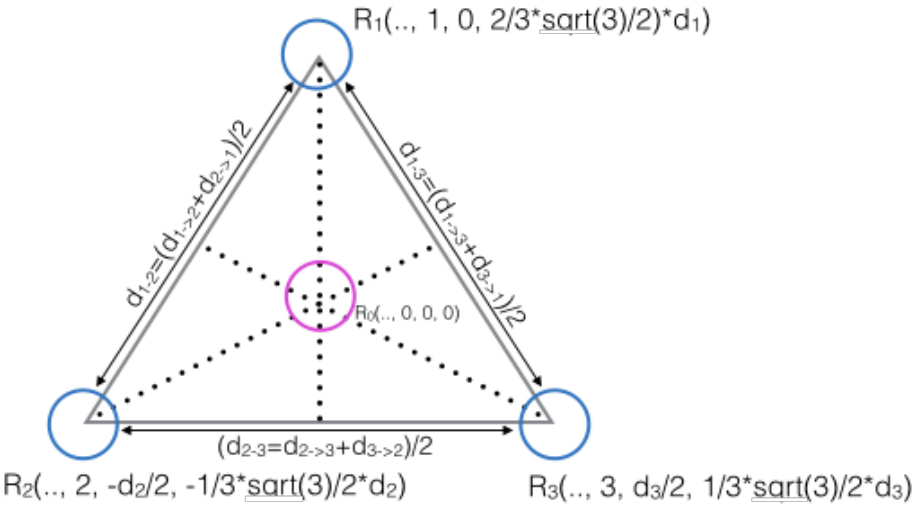

Рис. 2.7.2 Текущее положение роботов в группе

Внутренняя координата определяется относительно центра масс построения. Например если фигурой построения является треугольник, то точкой начала координат будет являться центр масс данного треугольника. Тогда

1. Получаем требуемую внутреннюю координату относительно центра
2. Получаем реальное положение относительно остальных роботов

Новое корректирующее значение получается из разности требуемого положения (рис. 2.7.1) и текущего (рис. 2.7.2). Расчёт текущего положения (рис. 2.7.2) производится по формулам:

$d_1=(d_{1-2}+d_{1-3})/2$

$d_2=(d_{1-2}+d_{2-3})/2$

$d_3=(d_{2-3}+d_{1-3})/2$

$v=sqrt((X_{expected}-X_{actual})^2+(Y_{expected}-Y_{actual})^2)$

$\varphi=\arcsin((Y_{expected}-Y_{actual})/2)$

К считанной скорости и углу добавляем корректирующее значение Итоговое значение скорости и угловой скорости записывается в */drive*

2.8 Drive node

вход: /drive

Данный модуль подписан на раздел */drive* и получает из него данные о целевой скорости данного робота R_i и угла поворота данного робота. Для каждого мобильного робота хранится конфигурация с указанием его параметров, необходимых для преобразования угловой скорости в конечное напряжение, передаваемое на каждый двигатель. Полученные значение модуль отправляет по интерфейсу UART на микроконтроллер stm32. Передача данных подобным способом представляет собой запись двух целочисленных значений в формате “%d %d” в /dev/ttyS0. Stm32 находится в состоянии непрерывной обработки сообщений и в случае получения, изменяет напряжения, выдаваемые на соответствующие выходы драйвера двигателей L293D. Таким образом осуществляется управление двигателями, и собственно самим мобильным роботом.

Для отправки сообщений по UART Raspberry Pi 3 должен быть сконфигурирован определённым образом, с запретом на автоматической изменение рабочей частоты процессора. В противном случае при корректных настройках serial-порта данные между Stm32 и Raspberry всё равно будут приходить в не читаемом формате.

2.9 Gazebo

вход: /drive

Представляет собой среду моделирования и симуляции поведения роботов в среде. [10] Главным преимуществом этой среды в данном случае это возможность конфигурирования её таким образом, что моделью симуляции можно управлять таким же образом, что и реальным роботом, позволяя подменять данной средой симуляции drive node и без дополнительных изменений тестировать алгоритм.

Модель для симуляции представлена конфигурационным файлом в формате *.world и содержит в себе как информацию о всех моделируемых объектах, так и плагины для взаимодействия с ROS.

2.10 Экспериментальный стенд

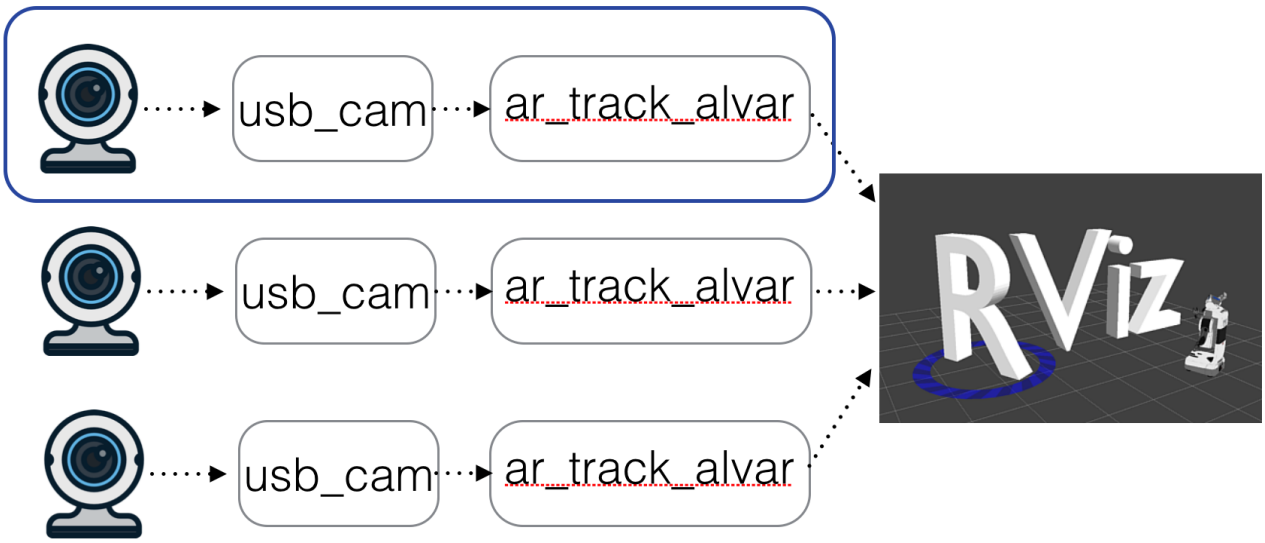

Рис. 2.10.1 Архитектура стенда

В рамках работы использовались две разновидности стенда: первый стенд оборудован одной камерой Logitech c130, второй стенд оборудован группой камер Logitech c920.

Стенд (рис. 2.10.1) в данной задаче является заменой системам глобального позиционирования и представляет собой набор камер, систему обработки видеопотока и систему визуализации. Представлено модулями usb_cam, ar_track_alvar [14] и rviz [20].

Модуль usb_cam является драйвером usb веб-камеры, выполненный в виде модуля ROS.

Данный модуль был сконфигурирован таким образом, что получая данные(видеопоток) из /dev/video# (# — номер устройства камеры) пишет их в раздел */usb_cam*, доступный другим модулям системы. Также камера должна быть предварительно откалибрована при помощи модуля ROS camera_calibration.

Модуль ar_track_alvar извлекает метки из видеопотока и позволяет писать в раздел */marker* глобальные координаты каждой из найденных меток в режиме реального времени.

Модуль rviz является мощной системой визуализации, позволяющей отображать совместно множество данных, получаемых в процессе работы системы, что облегчает отладку и позволяет своевременно определять состояние испытуемой системы. Данных модуль подписывается на различные разделы, и визуализирует данные из них.

2.11 Мобильный робот

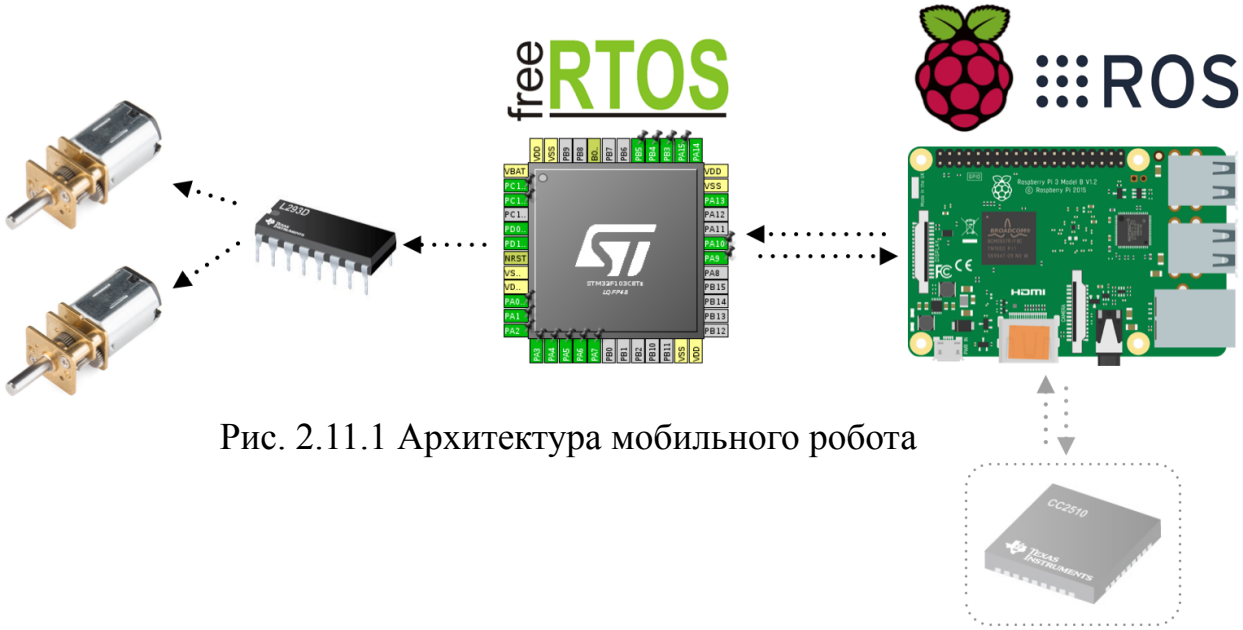

Рис. 2.11.1 Архитектура мобильного робота

Мобильный робот представлен колёсной платформой, с закреплёнными на ней контроллерами управления. Архитектура данного мобильного робота трёхуровневая (рис. 2.11.1). На самом нижнем уровне находятся двигатели и драйвер двигателей L293D. На среднем уровне находится микроконтроллер STM32 с установленной операционной системой реального времени FreeRTOS. На верхнем уровне находится микрокомпьютер Raspberry Pi 3 с установленным lubuntu linux.

Структура программы, записываемой на STM32 предварительно генерируется в специализированном ПО, выпускаемом компанией-разработчиком данных микроконтроллеров STM32CubeMX. Там же производится настройка системных таймеров, входных и выходных контактов микроконтроллера. Для взаимодействия с драйвером двигателей L293D используются аналоговые выходы, а для взаимодействия с Raspberry Pi 3 — цифровые.

Разработка ПО под STM32 наиболее удобна в IDE “System Workbench for STM32”, основанной на Eclipse IDE и разрабатываемой сообществом. Используемые языки программирования для stm32 — C и C++.

3 ЭКСПЕРИМЕНТАЛЬНАЯ ПРОВЕРКА СИСТЕМЫ

3.1 Моделирование работы алгоритма

Моделирование в данной работе проводилось заменой модуля Drive node на среду симуляций Gazebo (рис. 3.1.1). По скольку среда Gazebo не поддерживает возможность отрисовки пройденного пути, для этой цели использовалась система визуализации rviz [20]. Для конвертации и передачи данных из Gazebo в rviz был разработан дополнительный модуль odometry_converter_node. В качестве модели тестового робота использовался встроенный Pioneer2dx. Были произведены запуски моделей группы мобильных роботов с указанием конечной точки для центра масс группы для двух случаев: когда цель лежит на прямой относительно направления мобильных роботов (рис 3.1.2 а), и когда цель находится в стороне (рис. 3.1.2 б). Рисунок 3.2.1а показывает, что при перемещении по прямой мобильные роботы сохраняют построение на всём промежутке пути. Рисунок 3.1.2б показывает, что при перемещении с поворотом мобильные роботы с случае сбоя выравнивают строй.

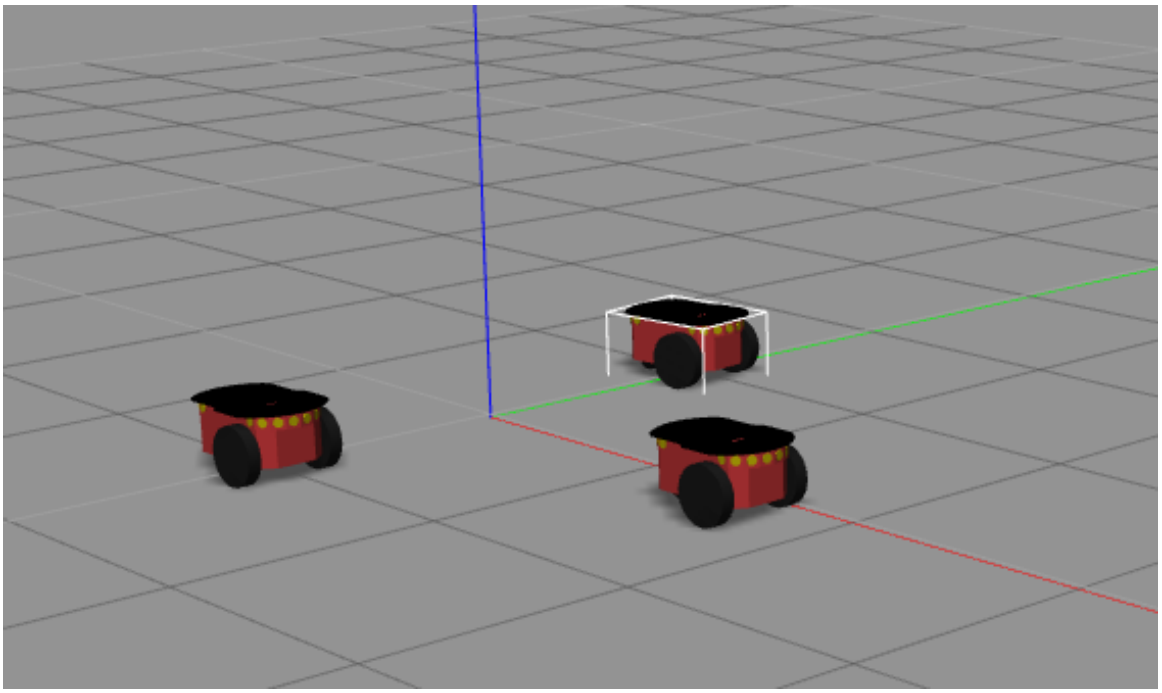

Рис. 3.1.1 Группа роботов Pioneer2dx в среде симуляции Gazebo

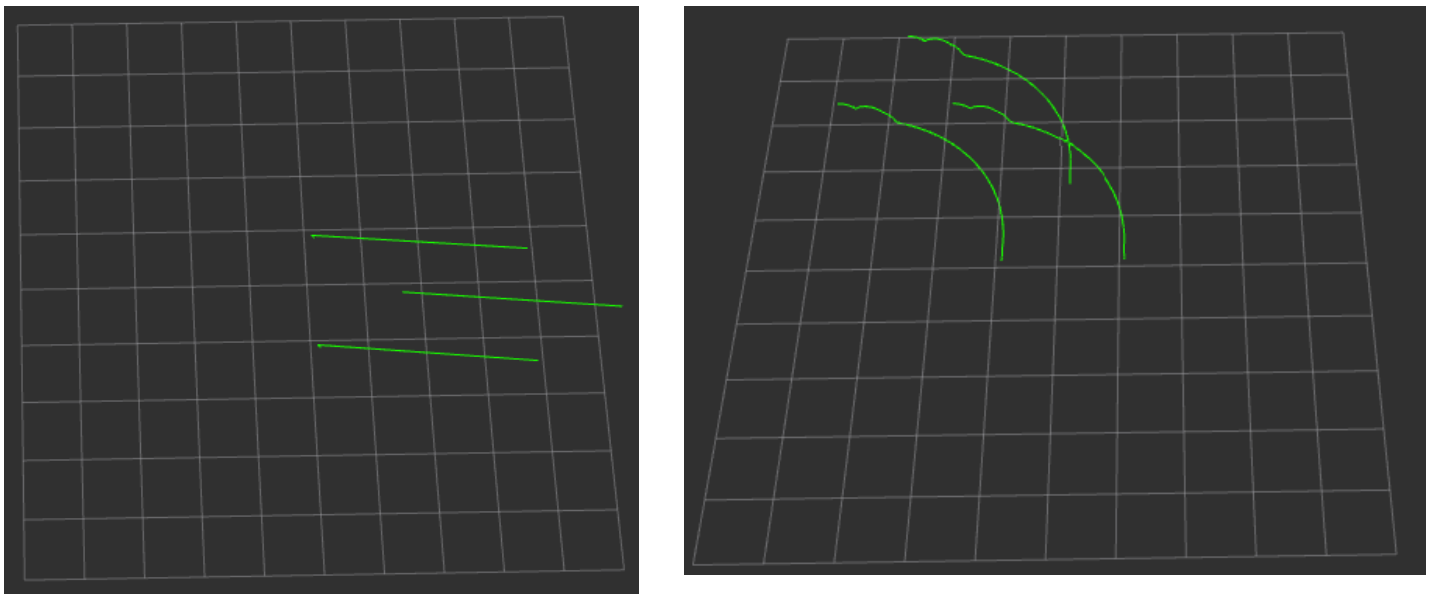

Рис. 3.1.2 Пути мобильный роботов в результате моделирования движения (а) по прямой, (б) с поворотом

ЗАКЛЮЧЕНИЕ

В данной работе было проведено исследование методов управления распределённой группой роботов. В рамках работы был разработан программно-аппаратный комплекс, позволяющий управлять автономной группой мобильных робототехнических платформ как единым целым. Аппаратная часть представлена стендом, позволяющим определять глобальную координату для каждого мобильного робота, а также группой автономных мобильных роботов с трёхуровневой архитектурой. Программный комплекс представлен набором ros-модулей, взаимодействующих между собой и позволяющих давать высокоуровневые задачи группе мобильных роботов. При выполнении работы было произведено исследование по определению расстояний внутри динамической системы при помощи беспроводной сети IEEE 802.15.1 BLE.

В результате работы был произведён обзор существующих решений и инструментальных средств и оборудования, разработаны архитектуры аппаратного и программного обеспечения, реализован программно-аппаратный комплекс. Также была проведена экспериментальная проверка системы, подтверждающая её действенность.

Список литературы

1. **К. Ю. Котов, А. С. Мальцев, А. А. Нестеров и др.** “Децентрализованной управление квадрокоптерами в составе группы лидер — ведомые” // Автометрия, 2017
2. **Ю. Н. Золотухин, К. Ю. Котов, А. С. Мальцев** “Робастное управление подвижными объектами в группе лидер — ведомые с использованием метода структурного синтеза” // Автометрия, 2015
3. **Ю. Н. Золотухин, К. Ю. Котов, А. А. Нестеров** “Децентрализованное управление подвижными объектами в составе маневрирующей группы” // Автометрия, 2007
4. **И.А. Каляев, А.Р. Гайдук, С.Г. Капустян** “Модели и алгоритмы коллективного управления в группах роботов” // ФИЗМАТЛИТ, 2009
5. **Tor-Inge Kvaksrud** “Range Measurements in an Open Field Environment” // 2008
6. **Karel Heurtefeux, Fabrice Valois** “Is RSSI a Good Choice for Localization in Wireless Sensor Network?” //
7. **Документация ROS (Robot Operating System)** [Электронный ресурс] URL: http://wiki.ros.org (дата обращения: 20.05.2018)
8. **April Glaser** “Intel invented a way for a single operator to fly hundreds of drones at once” [Электронный ресурс] URL: https://www.recode.net/2016/11/4/13517550/intel-single-operator-fly-hundreds-drones-shooting-star
9. **DARPA**, “OFFSET Envisions Swarm Capabilities for Small Urban Ground Units” [Электронный ресурс] http://www.defense-aerospace.com/cgi-bin/client/modele.pl?shop=dae&modele=release&prod=179363&cat=3
10. **Документация Gazebo** [Электронный ресурс] URL: http://gazebosim.org/tutorials (дата обращения: 20.05.2018)
11. **S.A. Stoeter, I.T. Burt, N. Papanikolopouslos** “Scout robot motion model” // 2003 IEEE International Conference on Robotics and Automation
12. **VICON**, [Электронный ресурс] URL: https://www.vicon.com (дата обращения 21.04.2018)

13. **OPTITRACK,** [Электронный ресурс] URL: https://optitrack.com (дата обращения 21.04.2018)
14. **AR TRACK ALVAR**, [Электронный ресурс] URL: http://wiki.ros.org/ar_track_alvar (дата обращения 21.04.2018)
15. **Mobile Robot Programming Toolkit**, [Электронные ресурс] URL: https://www.mrpt.org (дата обращения 22.04.2018)
16. **Microsoft® Robotics Developer Studio**, [Электронный ресурс] URL: https://msdn.microsoft.com/en-us/library/bb483024.aspx?f=255&MSPPError=-2147217396 (дата обращения 22.04.2018)
17. **Carnegie Mellon Robot Navigation Toolkit**, [Электронный ресурс] URL: http://carmen.sourceforge.net (дата обращения 22.04.2018)
18. **The Player Project**, [Электронный ресурс] URL: http://playerstage.sourceforge.net (дата обращения 23.04.2018)
19. **Universal Real-time Behavior Interface**, [Электронный ресурс] URL: http://www.gostai.com (дата обращения 22.04.2018)
20. **RVIZ**, [Электронный ресурс] URL: http://wiki.ros.org/rviz (дата обращения 15.04.2018)