ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»

(Н И У « Б е л Г У »)

ИНИСТИТУТ ИНЖЕНЕРНЫХ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

МОБИЛЬНАЯ СИСТЕМА ДЛЯ ОПТИМАЛЬНОГО ПОСТРОЕНИЯ 2-D СХЕМЫ ЗАМКНУТОГО ПРОСТРАНСТВА С ПРЕПЯТСТВИЯМИ НА МК ARDUINO

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем
очной формы обучения, группы 07001402
Лескина Кирилла Александровича

Научный руководитель
к.т.н., доцент
Чашин Ю.Г.

БЕЛГОРОД 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время развитие робототехники привело к активному росту интереса к мобильным роботам, предназначенным для решения широкого круга задач.

В результате популяризации робототехнических средств, задача о построении схемы замкнутого пространства с препятствиями, а также позиционирование робота в пространстве представляют большой интерес для современного мира. Задачи навигации и локализации сложны и до сих пор не решены в степени, достаточной для повсеместного использования роботов.

Наличие схем препятствий позволяет мобильным системам самостоятельно принимать решение о дальнейшем передвижении, поэтому построение схематической карты необходимо для обеспечения автономности мобильным системам.

Существуют инерциальные системы навигации и системы, основанные на использовании технологий GPS, однако такие системы не в состоянии обеспечить необходимую точность в построении карты препятствий для текущего местоположения мобильной системы, в то время как развитие способов локальной навигации на основе получения данных, получаемых с различных датчиков расстояния, позволяет с высокой степенью достоверности осуществлять оценку не только координат самого робота на местности, но и окружающих его объектов.

В рамках автономной системы навигации применяются гироскопы, цифровые компасы. Существенным недостатком таких систем является их чувствительность к неравномерностям поверхности: наклонам, кочкам и т. д. Это вносит определенные ограничения на их использование.

Локальные системы используют для позиционирования некоторую точку, обычно стартовую. Данные системы могут применяться на относительно больших локациях, например, для тактических беспилотных самолетов, работающих в рамках известной территории. Система навигации A-GPS,

использующая для позиционирования сотовые сети, также является локальной. В условиях замкнутого пространства целесообразно применение локальной системы позиционирования. В настоящее время наиболее часто применяются системы, использующие дальномеры: лазерные, инфракрасные, ультразвуковые и т. д. Системы датчиков способны получать и обрабатывать информацию о среде робота, а затем передавать эту информацию системе управления для проведения расчетов и построения карты препятствий. От качества этих данных зависит возможность оперативного реагирования на изменения окружающей обстановки и, в конечном итоге, успех выполнения роботом поставленной задачи. Сложность технического процесса определения текущего местоположения и построения карты обусловлена низкой точностью приборов, участвующих в процессе вычисления расстояний до препятствий относительно местоположения робота.

Актуальность данной дипломной работы заключается в необходимости расширения методов и алгоритмов для построения схем замкнутых локальных пространств, с использованием интегрированных в мобильную систему датчиков. Актуальность также обуславливается быстрым распространением различных робототехнический устройств, поскольку робототехника приобретает все большее значение и становится общедоступной.

Актуальность данной дипломной работы заключается в необходимости расширения методов и алгоритмов для построения схем замкнутых локальных пространств в связи с быстрым распространением различных робототехнический устройств, а также необходимостью исследования труднодоступных или опасных для человека мест.

Цель работы – разработать мобильную систему, позволяющую строить схематическую двумерную карту окружающего пространства и решать задачу позиционирования модуля на ней.

Система должна уметь обрабатывать данные с датчиков, строить по этим данным карту препятствий, вычислять маршрут к ближайшей неизведанной территории, а также адекватно реагировать на изменения этой карты в процессе

передвижения по построенному маршруту и корректировать его при необходимости. В ходе работы были сформулированы следующий список задач, которые необходимо решить при реализации:

- анализ проблемы построения карты препятствий и навигации на ней робота;
- выбор комплектующих и робототехнических модулей для создания мобильной системы;
- выбор инструментальных средств и технологий создания системы;
- реализация методов и алгоритмов для осуществления передвижения робота;
- реализация методов приема-передачи данных и их последующая обработка;
- тестирование мобильной системы.

В первой главе проводится анализ построения 2-D схемы препятствий и обзор программных и аппаратных компонентов.

Во второй главе описываются основные моменты разработки и проектирования мобильной системы.

Исследование основных алгоритмов и реализация основного функционала описана в третьей главе.

В главе «Апробация и тестирование системы» проводится анализ работоспособности реализованной системы, приводятся результаты работы и корректировки методов системы.

В заключении сделан вывод о степени достижения поставленных целей и задач.

Данная работа содержит 71 страницу, 4 главы, 17 рисунков, 1 таблицу.

ГЛАВА 1. АНАЛИЗ ЗАДАЧИ ПОСТРОЕНИЯ 2-D СХЕМЫ ПРЕПЯТСТВИЙ. ОБЗОР ПРОГРАММНЫХ И АППАРАТНЫХ КОМПОНЕНТОВ

1.1 Постановка задачи

Основной задачей реализуемой мобильной системы будет построение 2-D схемы препятствий для некого замкнутого контура. Для решения этой задачи необходимо из определенного набора компонентов, таких как датчик расстояния, контроллер, Bluetooth-модуль собрать мобильную систему, которая бы выполняла 2 основные функции – перемещение по заданному направлению и сканирование территории с последующей отправкой данных на удаленный сервер.

Типичные мобильные роботы имеют следующие компоненты: контроллер, управляющее программное обеспечение датчиков и исполнительные механизмы.

Контроллером выступает, как правило, микропроцессор, встроенный микроконтроллер или персональный компьютер (ПК).

Программное обеспечение может быть написано, как на языках высокого уровня, так и на языках низкого уровня, таких как C, C ++, Pascal, Fortran, Assembler или же с использованием специального программного обеспечения в режиме реального времени.

Используемые датчики зависят от требований, которые возникают в зависимости от поставленных перед роботом задач (тактильные датчики, дальномеры, определение местоположения и т.д.).

Также, мобильные роботы могут иметь компоненты для беспроводного соединения и получения информации извне, такие как Bluetooth или GPS модули. В дополнение к этим модулям могут быть интегрированы модули вычисления расстояния, такие как ультразвуковые или лазерные датчики.

1.2 Метод построения схемы препятствий в замкнутом пространстве.

Построение карты — это проблема интеграции информации, собранной с датчиков робота. Выполняя этот процесс, робот собирает информацию с имеющихся в системе датчиков и осуществляет построение карты препятствий, опираясь на полученные данные, либо передает информацию на удаленный сервер и все необходимые вычисления производятся удаленно, что позволяет снизить затраты на оборудование робота, ограничившись только необходимым минимумом для осуществления передвижения и сбора информации.

Главными аспектами в построении карты являются представление данных об окружающей среде и интерпретация данных датчиков.

На сегодняшний день задача позиционирования и построения карты описана в методе одновременной локализации и построения карты – SLAM[7].

SLAM используется в мобильных автоматизированных системах построения карты в неисследованном ранее пространстве или для обновления имеющейся карты с возможностью контроля пройденного пути. Подобные подходы используются при создании систем автоматического управления автомобилем, беспилотных летательных аппаратов, подводных аппаратов и т.д.

Обычно карты используются для определения позиции в пространстве и для графического изображения плана местности или для навигации. Они используются для оценки фактического местоположения путём записи информации, полученной от формы восприятия и сравнивая его с текущим набором представлений. Вклад карт в оценку текущего местоположения в пространстве возрастает с понижением точности и качества сенсоров восприятия пространства.

Карты в основном отражают вид пространства, зафиксированный в момент их построения. Совсем не обязательно, что вид пространства будет тем же в момент использования карт.

Сложность технического процесса определения текущего местоположения и построения карты обусловлена низкой точностью приборов, участвующих в процессе вычисления текущего местоположения.

Метод одновременной навигации и построения карты (SLAM) — это концепция, которая связывает два независимых процесса в непрерывный цикл последовательных вычислений. При этом результаты одного процесса участвуют в вычислениях другого процесса[9].

В настоящее время существует несколько основных подходов к решению этой задачи:

- EKF-SLAM;
- FastSLAM;
- DP-SLAM.

Структурное представление карты местности зависит от среды функционирования робота.

Для выбора наилучшей реализации задач SLAM вводится условная классификация сред функционирования:

- с многочисленными ярко выраженными ориентирами (например, поле с отдельно стоящими кустами);
- с отсутствием ярко выраженными ориентиров (например, коридоры, комнаты, где в качестве ориентиров могут выступать углы)[8].

Если в исследуемой среде нет возможности найти ориентиры, то ее рационально представить в виде массива, где элементы, отражающие положение препятствий, имеют значение 1, а все остальные — 0 (такое представление карты применяется, например, в алгоритме DP-SLAM[11]).

Для хранения структуры такой карты проще всего использовать картографическую базу данных, которая отражает положение ориентиров, их уникальные свойства и взаимосвязи. Матрица оценок состояния динамической системы на основе расширенного Фильтра Калмана использует именно этот вариант представления карты.

В качестве дальномеров в настоящее время используются лазерные дальномеры, гидролокаторы, стереосистемы. Для определения перемещения и поворотов робота могут использоваться одометры.

В простом варианте карту наиболее удобно представлять в виде сетки с заполнением ячеек, занятых препятствиями.

Хранить такую карту удобно в виде массива, где элементы, отражающие положение препятствий, имеют значение 1, а все остальные — 0.

Такое представление карты описывается в одном из подходов решения задач SLAM – DP-SLAM[6].

1.3 Обзор контроллера Arduino Robot.

Arduino Robot — первая официальная робо-платформа от Arduino, разработанная на основе Arduino Leonardo.

Arduino Robot состоит из двух плат — платы управления и моторной платы. Каждая плата представляет из себя полноценную Arduino-платформу на основе контроллера ATmega32u4, которая программируется с помощью Arduino IDE. Arduino Robot может быть запитан от USB или от 4 AA аккумуляторов. Источник питания выбирается автоматически. Arduino Robot реализован на базе Arduino Leonardo.

Arduino Leonardo — это плата схожая по характеристикам с Arduino Uno, но с несколько отличающимся микроконтроллером и его обвязкой.

В качестве микроконтроллера используется ATmega32u4. Он же используется и в качестве USB-UART преобразователя для прошивки.

Этим Arduino Leonardo выделяется среди остальных плат Arduino, где для коммуникации используется дополнительный микроконтроллер.

Платформа состоит из аппаратной и программной частей; обе достаточно гибки и просты в использовании. Для программирования используется упрощённая версия C++, известная так же как Wiring. Разработку можно вести как с использованием бесплатной среды Arduino IDE, так и с помощью

произвольного C/C++ инструментария. Поддерживаются операционные системы Windows, MacOS X и Linux.

Для программирования и общения с компьютером вам понадобится USB-кабель. Для автономной работы потребуется блок питания на 7,5—12 В.

Arduino Uno может питаться как от USB подключения, так и от внешнего источника: батарейки или обычной электрической сети. Источник определяется автоматически.

Платформа может работать при наличии напряжения от 6 до 20 В. Однако при напряжении менее 7 В работа может быть неустойчивой, а напряжение более 12 В может привести к перегреву и повреждению. Поэтому рекомендуемый диапазон: 7−12 В.

Платформа оснащена 32 КБ flash-памяти, 2 КБ из которых отведено под так называемый bootloader. Он позволяет прошивать Arduino с обычного компьютера через USB. Эта память постоянна и не предназначена для изменения во время работы устройства. Её предназначение — хранение программы и сопутствующих статичных ресурсов.

Также имеется 2 КБ SRAM-памяти, которые используются для хранения временных данных вроде переменных программы. По сути, это оперативная память платформы. SRAM-память очищается при обесточивании.

Ещё имеется 1 КБ EEPROM-памяти для долговременного хранения данных. По своему назначению это аналог жёсткого диска для Arduino.

На платформе расположены 14 контактов (pins), которые могут быть использованы для цифрового ввода и вывода.

Все они работают с напряжением 5 В, и рассчитаны на ток до 40 мА. Также каждый контакт имеет встроенный, но отключённый по умолчанию резистор на 20 - 50 кОм. Некоторые контакты обладают дополнительными ролями:

- serial: 0-й и 1-й. Используются для приёма и передачи данных по USB;
- внешнее прерывание: 2-й и 3-й. Эти контакты могут быть настроены так, что они будут провоцировать вызов заданной функции при изменении входного сигнала;

- PWM: 3-й, 5-й, 6-й, 9-й, 10-й и 11-й. Могут являться выходами с широтно-импульсной модуляцией (pulse-width modulation) с 256 градациями;
- LED: 13-й. К этому контакту подключен встроенный в плату светодиод. Если на контакт выводится 5 В, светодиод зажигается; при нуле — светодиод гаснет.

Помимо контактов цифрового ввода/вывода на Arduino имеется 6 контактов аналогового ввода, каждый из которых предоставляет разрешение в 1024 градации. По умолчанию значение меряется между землёй и 5 В.

Кроме этого на плате имеется входной контакт Reset. Его установка в логический ноль приводит к сбросу процессора.

Arduino Uno обладает несколькими способами общения с другими Arduino, микроконтроллерами и обычными компьютерами. Платформа позволяет установить последовательное (Serial UART TTL) соединение через контакты 0 (RX) и 1 (TX). Установленный на платформе чип ATmega16U2 транслирует это соединение через USB: на компьютере становится доступен виртуальный COM-порт. Программная часть Arduino включает утилиту, которая позволяет обмениваться текстовыми сообщениями по этому каналу.

Встроенные в плату светодиоды RX и TX светятся, когда идёт передача данных между чипом ATmega162U и USB компьютера.

Отдельная библиотека позволяет организовать последовательное соединение с использованием любых других контактов, не ограничиваясь штатными 0-м и 1-м.

С помощью отдельных плат расширения становится возможной организация других способов взаимодействия, таких как ethernet-сеть, радиоканал, Wi-Fi.

Arduino Uno обладает предохранителем, защищающим USB-порты вашего компьютера от перенапряжения и коротких замыканий. Хотя большинство компьютеров обладают собственными средствами защиты, предохранитель даёт

дополнительную уверенность. Он разрывает соединение, если на USB-порт подаётся более 500 мА, и восстанавливает его после нормализации ситуации.

Размер платы составляет 6,9 × 5,3 см. Гнёзда для внешнего питания и USB выступают на пару миллиметров за обозначенные границы. На плате предусмотрены места для крепления на шурупы или винты. Расстояние между контактами составляет 0,1″ (2,54 мм), но в случае 7-го и 8-го контакта — расстояние: 0,16″.

1.4 Беспроводное соединение с помощью Bluetooth

Для платформы Arduino разработаны различные аппаратные модули, предназначенные для реализации беспроводной связи.

Это Bluetooth, WI-FI, связь через оптические и радиоканалы, мобильная связь в стандарте GSM. Рассмотрим более подробно Bluetooth-модули.

Наиболее популярно семейство модулей HC. Существуют такие типы модулей:

- HC-03, HC-04(HC-04-M, HC-04-S) на чипе BC417143 – для промышленного применения;
- HC-05, HC-06(HC-06-M, HC-06-S) на чипе BC417143 – для коммерческого применения;
- HC-05-D, HC-06-D (с отладочной платой для оценки и тестирования);
- HC-07 – модуль с чипом CSR 41C6, предназначен для замены HC-06 (полностью с ним совместимый);
- HC-08 – модуль с ультранизким энергопотреблением и протоколом Bluetooth 4.0;
- HC-09 – самый новый модуль, предназначенный для замены HC-06 и HC-07.

Краткие характеристики Bluetooth-модулей:

- чип Bluetooth – BC417143 производства CSR company (Cambridge Silicon Radio);

- протокол связи – Bluetooth Specification v2.0+EDR;
- радиус действия – до 10 метров (уровень мощности 2);
- совместимость со всеми Bluetooth-адаптерами, которые поддерживают SPP;
- объем flash-памяти (для хранения прошивки и настроек) – 8 Мбит;
- частота радиосигнала – 2.40 .. 2.48 ГГц;
- хост-интерфейс – USB 1.1/2.0 или UART;
- энергопотребление – ток в течение связи составляет 30-40 мА. Среднее значение тока около 25 мА. После установки связи потребляемый ток 8 мА Режим сна отсутствует.

Все Bluetooth-модули могут работать в режиме master(ведущий) или slave(ведомый)[12]. В режиме master Bluetooth-модуль является инициатором подключения, а в режиме slave ожидает запрос на соединение. Практически все модули поддерживают оба этих режима, кроме HC-04 и HC-06, где режим функционирования определяется производителем, но изменение этих параметров возможно путем перепрограммирования микрокода ПЗУ.

1.5 Ультразвуковой дальномер

Ультразвуковой дальномер определяет расстояние до объектов точно так же, как это делают дельфины или летучие мыши. Он генерирует звуковые импульсы на частоте 40 кГц и слушает эхо. По времени распространения звуковой волны туда и обратно можно однозначно определить расстояние до объекта.

В отличие от инфракрасных дальномеров, на показания ультразвукового дальномера не влияют засветки от солнца или цвет объекта. Даже прозрачная поверхность будет для него препятствием. Но могут возникнуть трудности с определением расстояния до пушистых или очень тонких предметов.

ГЛАВА 2. РАЗРАБОТКА И ПРОЕКТИРОВАНИЕ МОБИЛЬНОЙ СИСТЕМЫ

2.1 Выбор программных средств

2.1.1 Среда разработки программного кода микроконтроллера

Для разработки программного кода микроконтроллера используется среда разработки Arduino IDE, которая предоставляет обширный функционал и набор библиотек для написания и отладки программ.

Язык программирования Arduino является стандартным C++ (используется компилятор AVR-GCC) с некоторыми особенностями.

Программы, написанные программистом Arduino, называются наброски (или иногда скетчи — калька от англ. sketch) и сохраняются в файлах с расширением ino. Эти файлы перед компиляцией обрабатываются препроцессором Ардуино. Также существует возможность создавать и подключать к проекту стандартные файлы C++[1].

Обязательную в C++ функцию main препроцессор Arduino создает сам, вставляя туда необходимые «черновые» действия.

Программист должен написать две обязательные для Arduino функции setup и loop. Первая вызывается однократно при старте, вторая выполняется в бесконечном цикле.

В текст своей программы (скетча) программист не обязан вставлять заголовочные файлы используемых стандартных библиотек. Эти заголовочные файлы добавит препроцессор Arduino в соответствии с конфигурацией проекта. Однако пользовательские библиотеки нужно указывать.

Менеджер проекта Arduino IDE имеет нестандартный механизм добавления библиотек. Библиотеки в виде исходных текстов на стандартном C++ добавляются в специальную папку в рабочем каталоге IDE[2]. При этом название

библиотеки добавляется в список библиотек в меню IDE. Программист отмечает нужные библиотеки, и они вносятся в список компиляции.

Arduino IDE не предлагает никаких настроек компилятора и минимизирует другие настройки, что упрощает начало работы для новичков и уменьшает риск возникновения проблем.

Простейшая Arduino-программа состоит из двух функций:

- setup-функция вызывается однократно при старте микроконтроллера;
- loop-функция вызывается после setup в бесконечном цикле все время работы микроконтроллера.

Помимо существования двух независимых ветвей оригинальной Arduino IDE (одна на arduino.cc, другая на arduino.org) разработчик может воспользоваться инструментарием, созданным сторонними производителями. Приведем краткий список наиболее известных решений:

- плагин к Eclipse. Подробности скрещивания есть и на русском;
- Visualmicro — плагин к Microsoft Visual Studio для работы с Ардуино;
- плагин к CLion. Особенностью плагина является создание Arduino CMake проекта в один клик;
- MariaMole IDE имеет продвинутые возможности работы с проектами и кодингом;
- Fritzing — простая Ардуино-ориентированная система проектирования и документирования схемотехники;
- поддержка Arduino встроена в Atmel AVR Studio начиная с версии 7;
- поддержка работы с Arduino встроена в IDE C-STEM Studio для языка Ch.

Графические языки программирования

- Minibloq;

- Ardublock;
- Modkit — платный, среди прочих поддерживает аппаратуру Ардуино;
- FLProg — бесплатный. Позволяет создавать программное обеспечение на языках FBD и LAD.

Второй тип представляет собой больше графические среды для создания принципиальных или логических схем.

2.1.2 Среда разработки серверной части и компонент визуализации

Для разработки серверной части необходимо выбрать язык программирования и среду разработки, которая бы в полной мере удовлетворила текущим требованиям к разработке вычисляемого функционала и позволила бы эффективно работать с библиотеками, с помощью которых возможно визуализировать работу системы.

Для визуализации интерфейса наиболее эффективным вариантом будет использование интерфейса программирования приложений – Windows Forms.

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework[4]. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, C++, так и на VB.Net, J# и др.[13]

С одной стороны, Windows Forms рассматривается как замена более старой и сложной библиотеке MFC, изначально написанной на языке C++. С другой стороны, WF не предлагает парадигму, сравнимую с MVC.

Для исправления этой ситуации и реализации данной функциональности в WF существуют сторонние библиотеки. Одной из наиболее используемых

подобных библиотек является User Interface Process Application Block, выпущенная специальной группой Microsoft, занимающейся примерами и рекомендациями, для бесплатного скачивания.

Эта библиотека также содержит исходный код и обучающие примеры для ускорения обучения.

Внутри .NET Framework, Windows Forms реализуется в рамках пространства имён System.Windows.Forms.

Поскольку средств Windows Forms достаточно для визуализации работы мобильной системы, остается выбрать C-подобный язык программирования, с помощью которого будут осуществляться вычисления, прием и передача данных и управление компонентами библиотеки Windows Forms.

Для этих задач был выбран язык C#, поскольку он обладает всеми необходимыми возможностями, легок в использовании и имеет большое количество готовых библиотек и стандартных функций, для решения различных, как считывание данных из COM–порта или взаимодействие с объектами Windows Forms.

2.2 Проектирование общей схемы работы мобильной системы

2.2.1 Схема взаимодействия робота с удаленным сервером

Поскольку задача построения карты препятствий, вычисление оптимального маршрута на каждом шаге и визуализация результата довольно трудоемкая задача, то имеющихся вычислительных ресурсов и памяти микроконтроллера недостаточно.

В этом случае необходимо к разделить эти обязанности между простым исполнителем команд – роботом и удаленным вычислителем – сервером.

Прием и передача данных осуществляется с помощью беспроводного Bluetooth-модуля HC-05.

Общая схема взаимодействия мобильной системы с удаленным сервером представлена на рисунке 2.1.

Рис. 2.1. Общая схема взаимодействия мобильной системы с удаленным сервером.

2.2.2 Проектирование схемы работы робота и его компонентов

Проектируемый мобильный робот должен представлять собой модульную конструкцию, состоящую из различного набора датчиков и компонентов, позволяющих в полной мере решать возложенную на него задачу по исследованию некого замкнутого контура с препятствиями.

Поскольку робот представлен как некий исполнитель команд, для выполнения поставленной задачи необходимо описать общую схему его работы следующим набором функциональных возможностей:

- Прием и передача данных;
- Исполнение команды;
- Сбор данных.

На рисунке 2.2 показана общая схема работы исполнителя.

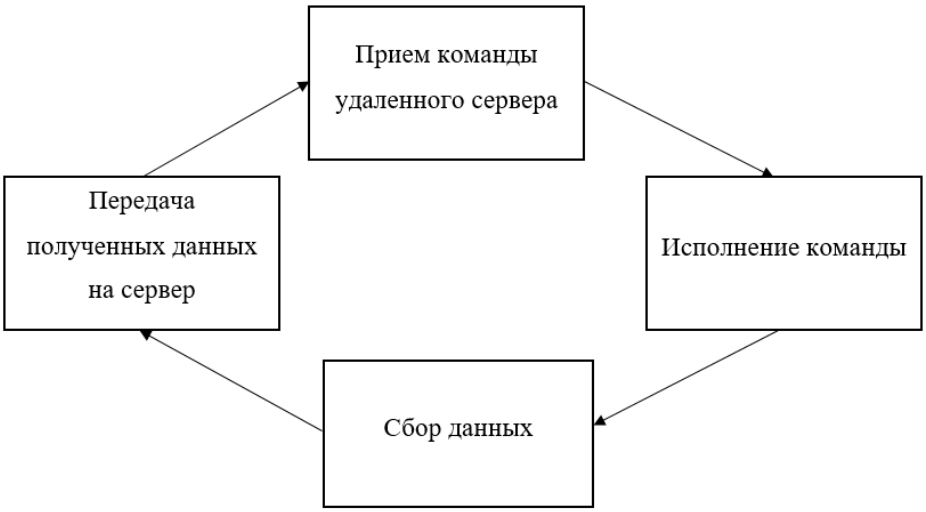

Рис. 2.2. Общая схема работы исполнителя

Для решения этой задачи необходимо определить тот набор аппаратных средств, которые должен содержать робот для того, чтобы полностью отвечать поставленным требованиям к имеющейся задаче.

Для реализации всех необходимых функциональных возможностей робота для текущей задачи понадобится:

- роботизированная платформа Arduino Robot на базе Arduino Leonardo;
- плата Arduino UNO;
- ультразвуковой дальномер HC-SR04;
- сервопривод;
- Bluetooth модуль HC-05.

Чтобы составить схему подключения компонентов и модулей робота, необходимо разобраться с функциональными возможностями каждого компонента, их принципом работы и взаимодействия друг с другом.

Разберем по порядку каждый из этих компонентов что бы определить их свойства, принцип работы и способы взаимодействия с ними.

Роботизированная платформа Arduino Robot на базе Arduino Leonardo

Arduino Robot представляет собой готовую роботизированную платформу, которая содержит большое количество различных датчиков, кнопок а так же разъемов под платы расширения. Конструкция Arduino Robot представляет собой 2 платы с разделенным функционалом – плата управления и плата расширения. Каждая плата представляет собой полноценную платформу на основе контроллера ATmega32u4.Плата управления имеет следующие дополнительные элементы:

- клавиатура на 5 кнопок;
- потенциометр, подключённый к аналоговому входу;
- цветной графический LCD-экран, подключённый через SPI;
- разъём для micro-SD карты в FAT16;
- три площадки под пайку устройств с I²C-интерфейсом;
- четыре площадки для прототипирования.

На рисунке 2.3 показано схематическое изображение платы управления Arduino Robot.

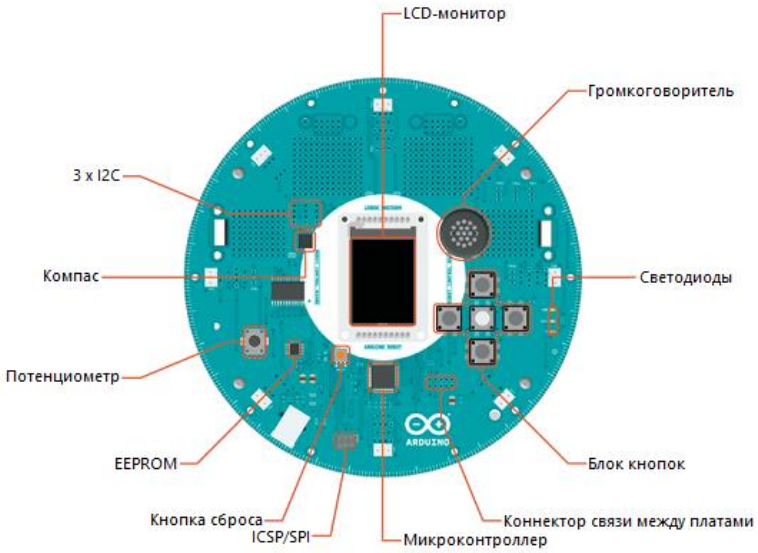

Рис. 2.3. Схематическое изображение платы управления Arduino Robot (вид сверху)

Плата управления представляет собой набор таких функциональных элементов и датчиков, которые позволяют эффективно осуществлять передвижение. Моторная плата имеет следующие дополнительные элементы:

– потенциометр для калибровки двигателей;
– вход для зарядного устройства на 9 В;
– линейный регулятор напряжения, обеспечивающий питанием все устройства робота;
– аккумуляторный отсек на 4 AA NiMH-аккумулятора;
– линейка из 5 ИК-сенсоров для следования за линией;
– разъём I²C;
– 2 площадки для прототипирования.

Моторная плата основана на том же контроллере, что и управляющая плата. Однако, несмотря на схожие технические характеристики платы имеют различный набор функциональных элементов и используются при решении разных задач.

На рисунке 2.4. представлено схематическое изображение моторной платы Arduino Robot для вида сверху.

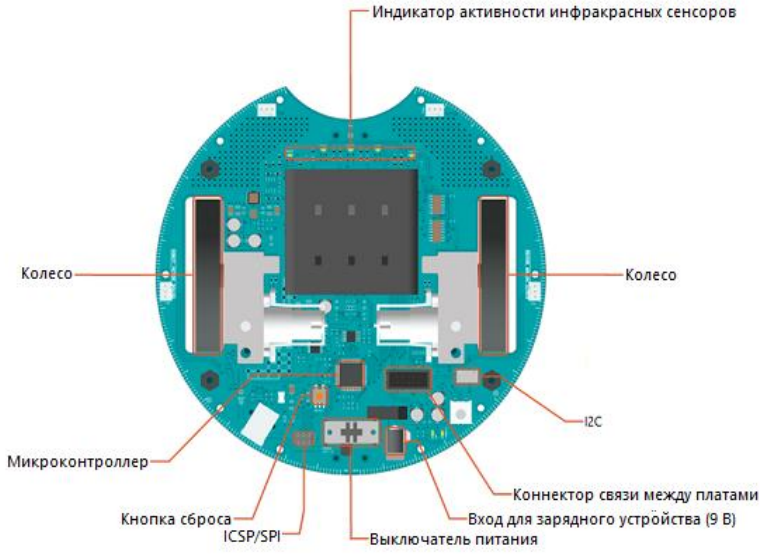

Рис. 2.4. Схематическое изображение моторной платы Arduino Robot вид сверху

На рисунке 2.5 показано схематическое изображение моторной платы Arduino Robot с видом снизу.

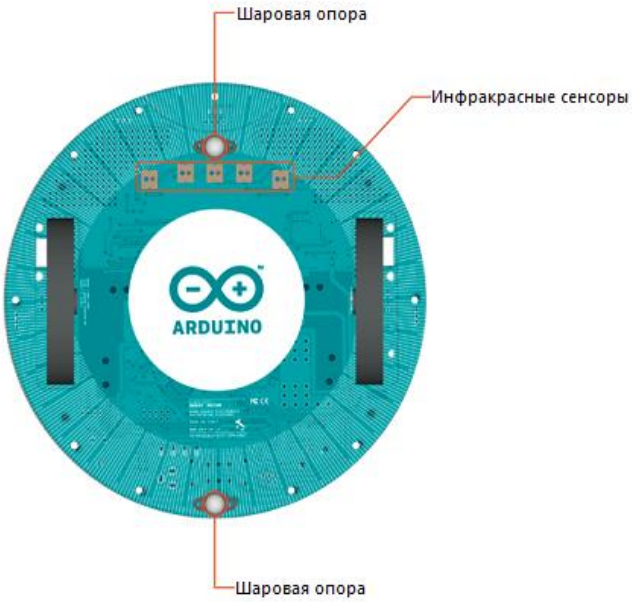

Рис. 2.5. Схематическое изображение моторной платы Arduino Robot вид снизу.

К компьютеру робот подключается через USB-порт, и определяется как виртуальный COM-порт.

Светодиоды LED1/RX и TX горят только во время связи между компьютером и роботом. Обе платы имеют разные USB-идентификаторы устройства, поэтому они будут доступны на разных портах в Arduino IDE.

Связь плат между собой осуществляется с помощью последовательных портов микропроцессоров.

Платы соединены 10-жильным плоским кабелем, через который происходит питание платы управления, производится связь микропроцессоров и передаётся дополнительная информация, например, текущий заряд аккумуляторов.

Arduino UNO

Arduino Uno — флагманская платформа для разработки на базе микроконтроллера ATmega328P.

На Arduino Uno предусмотрено всё необходимое для удобной работы с микроконтроллером: 14 цифровых входов/выходов (6 из них могут использоваться в качестве ШИМ-выходов), 6 аналоговых входов, кварцевый резонатор на 16 МГц, разъём USB, разъём питания, разъём для внутрисхемного программирования (ICSP) и кнопка сброса.

Сердцем платформы Arduino Uno является 8-битный микроконтроллер семейства AVR — ATmega328P[2].

Микроконтроллер ATmega16U2 обеспечивает связь микроконтроллера ATmega328P с USB-портом компьютера. При подключении к ПК Arduino Uno определяется как виртуальный COM-порт. Прошивка микросхемы 16U2 использует стандартные драйвера USB-COM, поэтому установка внешних драйверов не требуется.

Ультразвуковой дальномер HC-SR04

Ультразвуковой дальномер рассчитан на определение расстояния до объектов в радиусе четырёх метров.

Работа модуля основана на принципе эхолокации. Модуль посылает ультразвуковой сигнал и принимает его отражение от объекта. Измерив время между отправкой и получением импульса, не сложно вычислить расстояние до препятствия. Контакты VCC и GND служат для подключения питания, а Trig и Echo— для отправки и приема сигналов дальномера.

В таблице 2.4 представлены основные характеристики ультразвукового дальномера HC-SR04.

Таблица 2.4

Характеристика ультразвукового дальномер HC-SR04

Напряжение питания	5 В
Потребление в режиме тишины	2 мА
Потребление при работе	15 мА
Диапазон расстояний	2–400 см
Эффективный угол наблюдения	15°
Рабочий угол наблюдения	30°

Зная продолжительность высокого сигнала на пине Echo можем вычислить расстояние, умножив время возвращения сигнала к датчику на скорость распространения звука в воздухе (340 м/с).

На рисунке 2.6 представлено схематическое изображение подключения ультразвукового дальномера hc-sr04 к плате Arduino UNO.

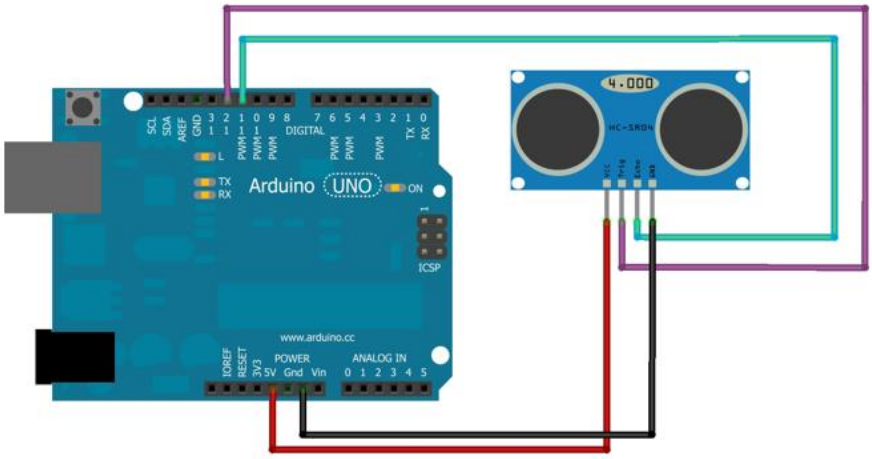

Рис. 2.6. Схематическое изображение подключения ультразвукового дальномера hc-sr04 к плате Arduino UNO

На рисунке 2.7 изображена диаграмма направленности дальномера HC-SR 04.

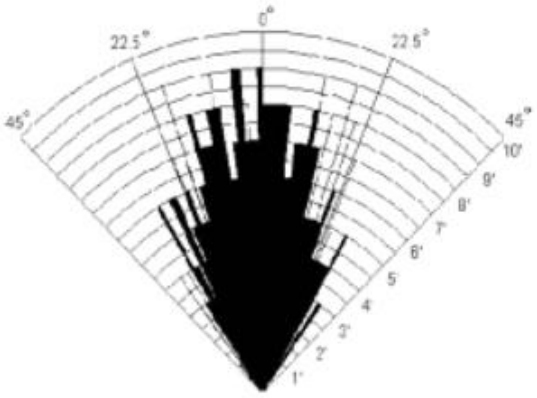

Рис. 2.7. Рисунок диаграммы направленности дальномера HC-SR 04

На рисунке 2.7 видна диаграмма направленности звукового сигнала, которая дает общее представление о работе ультразвукового дальномера.

Рабочая область дальномера составляет конус 30 градусов, однако эффективный диапазон замера составляет 15 градусов.

Сервопривод

Сервопривод – это устройство, с автоматической коррекцией и поддержанием определенного уровня состояния, основываясь на некотором наборе внешних параметров.

При помощи электромоторов это устройство преобразует электричество в движение. В сервоприводах используются потенциометры в качестве датчика обратной связи, преобразующего угол поворота в электрический сигнал. С его помощью определяется и устанавливается угол поворота механизма.

Сервоприводы обычно имеют ограниченный угол вращения 180 градусов, их так и называют «сервопривод 180°».

Но существуют сервоприводы с неограниченным углом поворота оси. Это сервоприводы постоянного вращения или «сервоприводы 360°».

Схематическое изображение подключения сервопривода к батарее питания и плате Arduino UNO показано на рисунке 2.8.

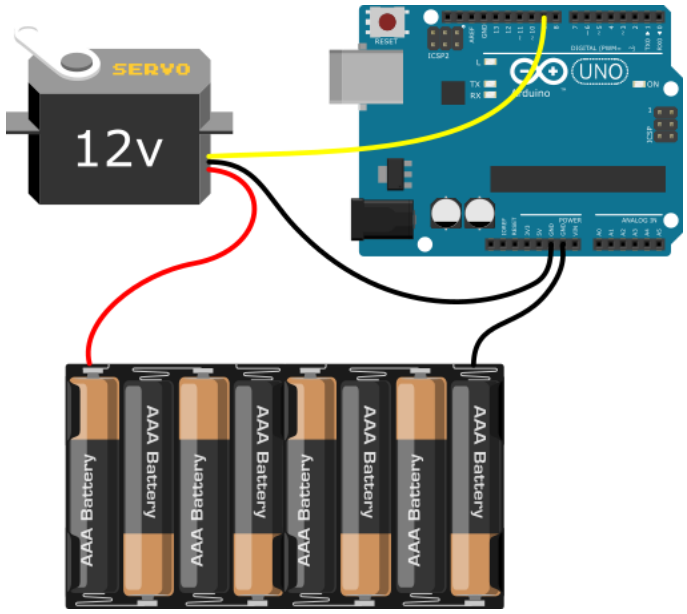

Рис. 2.8. Схематическое изображение подключения сервопривода к батарее питания и плате Arduino UNO

Bluetooth модуль HC-05

Bluetooth-модуль общается с управляющей платой по протоколу UART.

Универсальный асинхронный приёмопередатчик (UART) — узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами. Преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по одной физической цифровой линии другому аналогичному устройству. Метод преобразования хорошо стандартизован и широко применяется в компьютерной технике.

Дополнительный сигнальный пин K служит для перевода модуля в режим AT-команд.

Со стороны управляемого устройства, такого как Arduino, этот модуль выглядит как обычный последовательный интерфейс.

Сфера применения этого модуля не ограничивается управлением. Его можно использовать и для пересылки показаний разнообразных сенсоров.

Рабочее напряжение этого bluetooth-модуля — 3,3 В, но его входы толерантны к 5 В, поэтому он совместим со всеми платами Arduino.

Подключение стандартное для устройств, подключающихся по последовательному интерфейсу:

- контакт RX модуля подключается к контакту TX целевого устройства;
- контакт TX модуля подключается к контакту RX целевого устройства.

Скорость передачи данных 9600 бод. Имя модуля по умолчанию HC-06. Стандартный пароль для подключения1234.Все настройки по умолчанию могут быть изменены с помощью AT-команд. Но Bluetooth-модуль HC-06 может выступать только в slave-режиме. Это означает, что он не может самостоятельно подключаться к другим Bluetooth-устройствам.

Совместная работа модулей

После того как внутреннее устройство используемых модулей изучено необходимо составить общую схему их подключения, поскольку для достижения поставленной задачи необходимо обеспечить корректную совместную работу между ними при помощи проводных соединений и программно. Поскольку Arduino Robot реализован на базе Arduino Leonardo, то при построении схемы выбирается последняя плата в качестве комплектующего элемента. На рисунке 2.9 представлено схематическое изображение подключения используемых модулей и соединение между собой плат.

Плата Arduino Robot является относительно новой и она имеет ряд недостатков – не все стандартные датчики работают корректно с данной платой. Это обусловлено тем, что контакты имеют множество ответвлений и возникают конфликты. Производитель предлагает решить данную проблему использованием специализированных датчиков, однако данные датчики не дают точных показателей, что является недопустимым для решения данной задачи. Поэтому, для решения возникшей проблемы, было решено использовать Arduino Uno, в качестве связующего звена для получения данных с датчиков, а также для передачи данных. Это является допустимым, поскольку робот будет выполнять лишь полученные команды и постоянные замеры показателей датчиков не являются критичными для системы в целом. Для связи используется проводная технология связи I2C.

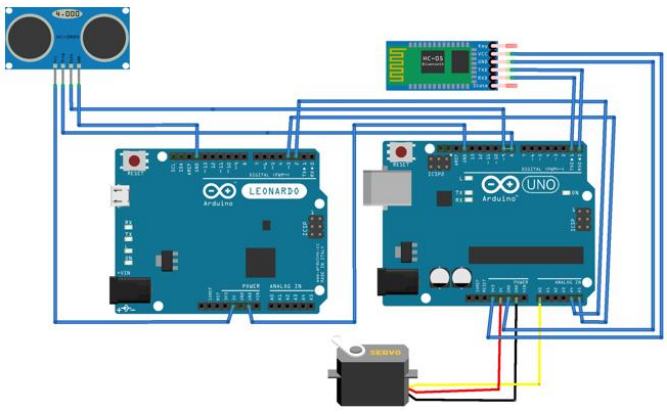

Рис. 2.9. Схематическое изображение подключения используемых модулей и соединение плат между собой

Принципиальная схема соединения показана на рисунке 2.10

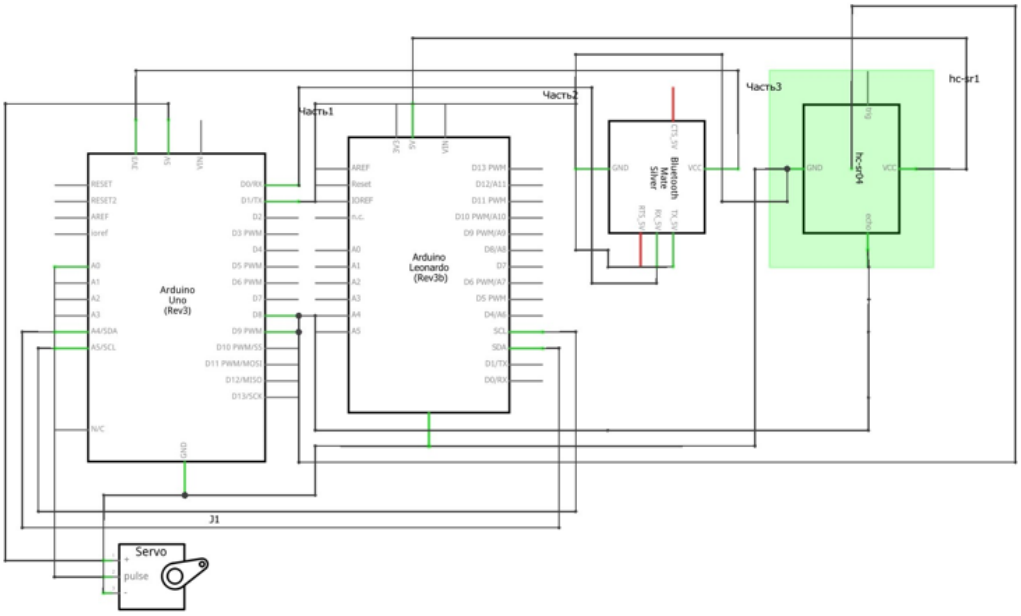

Рис. 2.10. Принципиальная схема соединения модулей

На рисунке 2.10 также показано соединение между двумя платами Arduino, сервоприводом, Bluetooth-модулем и ультразвуковым датчиком.

2.2.3 Проектирование схемы работы удаленного сервера

Удаленный вычислитель (сервер) должен предоставлять ряд функциональных возможностей, с помощью которых будет возможно осуществление корректного функционирования мобильной системы.

Наличие удаленного вычислителя позволяет определить ряд трудоемких задач, которые могут быть выполнены с использованием технических возможностей самого вычислителя и не будет необходимости беспокоится о вычислительных мощностях самого робота.

Для успешного выполнения поставленной задачи удаленный сервер должен будет поддерживать возможность выполнения следующего ряда действий:

- прием и передача данных;
- построение карты препятствий;

– определение следующего оптимального шага робота опираясь на полученную роботом, с помощью датчиков, информацию и построенную карту препятствий.

Составленная схема работы удаленного сервера показана на рисунке 2.11

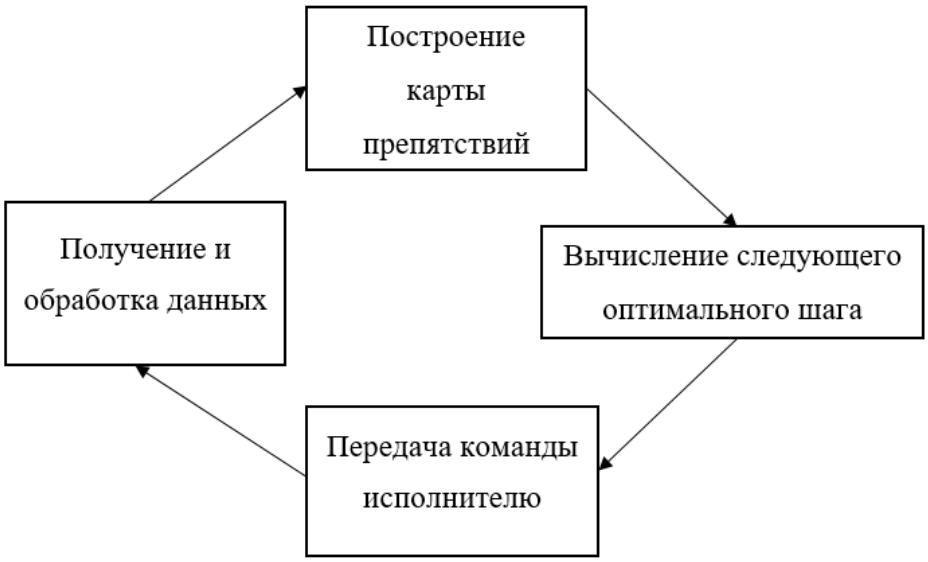

Рис. 2.11. Схема работы удаленного сервера

ГЛАВА 3. РЕАЛИЗАЦИЯ АЛГОРИТМОВ

3.1 Реализация алгоритма для серверной части

Серверная часть системы производит необходимые вычисления на основании полученных данных и затем предоставляет удаленному исполнителю команду, которую тот должен выполнить.

Используя схему работы, спроектированной в главе 2 реализуем получение данных, обработку и построение карты, вычисление следующей команды и ее передача роботу.

Что бы упростить работу с этими функциональными модулями, реализуем базовый функционал алгоритма, который позволит просканировать замкнутую область и построить карту.

Листинг 1. Основной алгоритм передвижения робота.

```
private void moveRobotAlgorithm() {
     map[robo.y][robo.x] = 2;
     while (!scanDone()) {
isCycled = false;
cantMoveCount = 0;
this.isBasicMove = true;
if (!baseMove()) {
this.isBasicMove = false;
anotherWay();
}
     }
     Console.WriteLine("Done!");
     currentPort.Close();
}
```

На листинге 1 приведена основная схема передвижения и сканирование местности роботом.

Для того что бы была возможность отслеживать текущее положение робота относительно карты необходимо создать объект карты – map и объект робота.

Объект робота представлен структурой и включает в себя такие элементы как координаты x и y, направление на карте и методы поворота робота вправо и влево. На листинге 2 представлена основная структура робота.

Листинг 2. Структура робота

```
struct Robot {
      public int x;
      public int y;
      public int direction;
      public void turnRight() {
this.direction = (this.direction + 1) % 4;
      }
      public void turnLeft() {
this.direction = (3 - -this.direction) % 4;
      }
};
```

Карта представляет собой двумерный массив значений произвольного размера. Поскольку размеры исследуемого замкнутого пространства и начальное положение робота в этом пространстве неизвестно, зададим произвольные размеры карты, принимая во внимание размеры клетки.

Используемый робот круглой формы и радиус платформы составляет 185 мм. Размер клетки для упрощения может быть равен 200 мм. В таком случае для стандартных замкнутых пространств – комнат, в среднем достаточно иметь карту 50x50 клеток, что будет покрывать территорию размером 10 м на 10 м, что подходит для исследования и построения карты жилых комнат в помещениях.

Для корректной идентификации исследуемой области на карте, вводятся маркеры, позволяющие однозначно определить состояние клетки и наличие, либо отсутствие препятствий на ней. Для текущей задачи достаточно 4 маркера, позволяющих идентифицировать клетку, среди которых маркер неисследованной или неизвестной территории, маркер просканированной области, маркер посещенной роботом области и маркер препятствия. Все эти маркеры необходимы для обеспечения корректной работы основного алгоритма.

В момент инициализации происходит очистка всех данных. Карта очищается и все клетки приобретают цвет маркера неисследованной территории, а робот устанавливается в центр карты.

Получение и обработка данных

Построение карты и вычисление дальнейшего маршрута передвижения возможно только при наличии обратной связи с роботом. Получение данных является одним из ключевых моментов в работе алгоритма.

На основании полученных данных при сканировании местности осуществляется построение карты, а затем определяется оптимальный маршрут для движения робота и передается роботу командой для исполнения.

Для приема и передачи данных используется технология Bluetooth, которая позволяет установить стабильную связь между двумя модулями. При подключении на удаленном сервере резервируется специальный порт, через который можно обмениваться данными в двустороннем порядке.

Для работы с портами в C# имеется встроенная библиотека System.IO.Ports, которая позволяет задать основные параметры для соединения и работать с подключением как с обычным последовательным портом[13]. Передача данных осуществляется побайтно и поскольку процесс сканирования занимает определенное, не фиксированное время, необходимо дожидаться, пока поступит вся необходимая информация после сканирования, чтобы продолжить работу алгоритма.

Объект последовательного порта предоставляет возможность работать с событиями и определять с помощью функций событийную модель поведения при получении данных.

Для обеспечения этой возможности необходимо реализовать обработчик события на получение данных и закрепить эту функцию для объекта, представляющего открытый последовательный порт. На листинге 3 показана функция для обработчика данных, получаемых через последовательный порт посредством Bluetooth.

Листинг 3. Обработчик получения данных через последовательный порт Bluetooth

```
private static void DataReceivedHandler(object sender, SerialDataRe
ceivedEventArgs e)
{
      SerialPort sp = (SerialPort)sender;
      string data = sp.ReadExisting();
      buffer += data;
      if(buffer.Length == 5) {
waitForData = false;
      }
}
```

Функция, приведенная в листинге 3 позволяет получить и обработать данные по факту их получения. Данные заполняют буффер результатами сканирования местности относительно робота в определенном порядке. Ожидание получения данных прекращается, когда буффер заполнен.

Построение карты

Когда процедура сканирования завершена и успешно получены все необходимые данные можно приступать к построению карты, нанося соответствующие маркеры на карту, опираясь на результаты сканирования.

Получаемые данные имеют строго заданный формат и порядок, что позволяет однозначно определить просканированную ячейку и ее положение на карте опираясь на текущее положение робота и его направление.

Процедура соотношения просканированной области относительно робота к общей карте проста и, зная текущее направление робота, можно определить 4 варианта ситуаций, когда направление робота соответствует одному из четырех возможных направлений на карте.

Соответствующая клетка на карте приобретает окраску маркера с препятствием, когда данные из буффера подтверждают наличие препятствия на соответствующей клетке.

В противном случае клетка может быть окрашена в маркер просканированной области, но только в том случае, если ранее на этой клетке робота еще не было, иначе клетка окрашивается маркером посещенной клетки.

Построение карты является повторяющейся процедурой и производится каждый раз после завершения сканирования роботом и передачей всех результатов сканирования.

Вычисление маршрута передвижения робота

Одной из самых сложных и трудоемких задач в работе удаленного сервера заключается вычисление следующего шага робота и построение маршрута для передвижения.

Для реализации этой процедуры необходимо разобрать один из ключевых моментов в работе основного алгоритма – процедуре передвижения робота.

Процедура передвижения робота разделяется на 2 основных вида передвижения – циклическое передвижение по кругу и передвижение к точке. В данной дипломной работе реализованы оба варианта передвижения в виду особенностей работы каждого из этих видов.

Разберем процедуру передвижения к точке. Данный вариант перемещения используется, когда необходимо проехать из точки А в точку Б по кратчайшему маршруту с учетом имеющихся препятствий на карте. Такая ситуация может быть вызвана в том случае, когда на карте остались некоторые неисследованные участки и робот находится на некотором расстоянии от них, а близлежащая территория уже исследована.

Существует несколько способов построения кратчайших маршрутов на карте с использованием жадных алгоритмов, таких как a*, поиск в ширину, волновой метод, алгоритм дейкстры[3]. Все эти алгоритмы позволяют найти кратчайший путь из точки А в точку Б на некотором графе.

Поскольку задача исследования территории подразумевает выполнение двух подзадач – поиск следующей точки и построение маршрута к точке то такие методы как a* или алгоритм дейкстры можно не рассматривать в виду других

алгоритмов, способных совмещать одновременно и поиск следующей неизвестной области, и построение маршрута к этой области.

С поставленной задачей отлично справится алгоритм волновой трассировки, или алгоритм Ли, который представляет собой алгоритм поиска кратчайшего пути на планарном графе. Он относится к алгоритмам, основанным на методах поиска в ширину[5].

Этот алгоритм позволит совместить две процедуры – поиск ближайшей неисследованной области и построение пути к ней, а также позволит избежать процедуры построения графа, который необходим для других методов, работающих с графами.

Основная работа алгоритма включает в себя такие этапы как инициализация, распространение волны и построение пути. Алгоритм работает, опираясь на информацию о начальной и конечной точках, однако можно изменить поведение алгоритма таким образом, что будет задан некий класс конечных точек, без конкретного указания о их местоположении, заставив алгоритм распространять волну до тех пор, пока не будет найдена хотя бы одна из точек заданного класса конечных точек.

Преимущество такого алгоритма заключается в том, что нам необходимо знать только начальную точку, из которой начинается распространение волны и не обязательно знать конечную точку – алгоритм должен сам ее найти и составить путь к этой точке.

Еще одним полезным свойством этого алгоритма является определение недостижимости областей. Если в процессе работы алгоритма волна распространится на все достижимые точки, но некоторая точка из класса конечных точек не будет найдена, то все не посещенные точки можно считать недостижимыми и на этом работа алгоритма завершается.

Изначально для работы алгоритма необходимо составить карту весов, которая отображает текущую карту препятствий, задавая наибольшее отрицательное значение для весов, обозначающих препятствие и наименьшее для

неисследованной области. На листинге 4 показана процедура построения карты весов для волнового метода с использованием карты препятствий.

Листинг 4. Построение карты весов для волнового метода на основании текущей карты препятствий

```
for (int i = 0; i < numOfCells; i++) {
     waveMap[i] = new int[numOfCells];
     for (int j = 0; j < numOfCells; j++)  {
switch (map[i][j]) {
case 3: waveMap[i][j] = -4; break;
case 2: waveMap[i][j] = -3; break;
case 1: waveMap[i][j] = -2; break;
default: waveMap[i][j] = -1; break;
}
     }
}
```

Затем начинается процедура распространение волны во все стороны в результате чего возможен один из двух вариантов – будет найден непосещенный ранее участок карты, либо такого участка не будет найдено. В случае нахождения неисследованной области запускается процедура восстановления пути. Если такого участка найти не удалось, выполняется процедура закрашивания всех оставшихся непосещенных областей как недостижимых, а работа алгоритма завершается.

Процедура построения маршрута этим способом является комплексной и имеет квадратичную сложность - O(N^2).

Для того что бы снизить трудозатраты на проведение вычислений и построение маршрута, был реализован простой, но эффективный способ обхода пространства с последующим сканированием области. Способ заключается в циклическом обходе пространства по спирали. В этом случае робот движется прямо, пока не достигнет препятствия и затем осуществляет поворот направо или налево. Процедура повторяется до тех пор, пока робот не будет зациклен из-за того, что вся близлежащая территория уже была исследована, или имеются препятствия, которые не позволяют проехать по заданному алгоритму.

Такой способ исследования территории является простым, в плане трудозатрат на вычисления, и эффективным по скорости исследования большого объема пространства за меньший промежуток времени, но не идеальным и пропускает некоторые клетки на пустом пространстве. Для этого используется волновой алгоритм.

Передача команды исполнителю

После определения маршрута, по которому робот будет осуществлять дальнейшее передвижение, построенный путь разбивается на ряд команд, которые отправляются роботу для выполнения. Простая команда представляет собой указание, в какую сторону роботу необходимо проехать относительно своего текущего положения. Передача команды осуществляется с помощью Bluetooth модуля, используя установленное ранее подключение. Затем сервер переходит в режим ожидания, до тех пор, пока робот не выполнит команду и не передаст данные после сканирования. Процедура повторяется.

3.2 Реализация алгоритма для робота

Для реализации алгоритма работы на роботе используется ранее рассмотренную среду программирования Arduino IDE, которая обладает всеми необходимыми возможностями для реализации программ, загружаемых в микроконтроллер.

Поскольку мобильная система состоит из двух микроконтроллеров, то необходимо будет реализовать программы для каждого микроконтроллера с учетом разделенных обязанностей, выполняемых микроконтроллерами, а также необходимо обеспечить передачу данных между платами.

Микроконтроллер на базе Arduino UNO, интегрированный в плату Arduino Robot дает доступ к управлению роботом, посредством библиотеки ArduinoRobot.h, которая инициализирует объект Robot и предоставляет интерфейс для управления этим объектом. Из функциональных возможностей

сюда входит управление моторами робота, управление встроенным LCD дисплеем, управление пьезо-элементом, потенциометром и другими модулями.

Из всего этого списка функциональных возможностей используется только управление моторами. Управление Bluetooth-модулем, сервоприводом и ультразвуковым датчиком осуществляется посредством отдельной платы - Arduino UNO. Данные между двумя микроконтроллерами будут передаваться через проводное соединение с использованием библиотеки Wire.h. Данная библиотека имеет все необходимые инструменты для работы с проводным соединением между двумя микроконтроллерами, а также дополнительные функции-обработчики событий, как например onRecieve, которые передают управление по определенному событию, например, при получении данных. Для управления сервоприводом используется библиотека Servo.h. А показания дальномера снимаются с определенных входов, куда подключается модуль ультразвукового дальномера. На листинге 1 показаны настройки для Arduino UNO, задаваемые основной функцией setup.

Листинг 1. Основная настройка программы для Arduino UNO.

```
const int UNO_ADDRESS=8;
const int ROBOT_ADDRESS=9;
const int SERVO_PIN=A0;
#define echoPin 8;
#define trigPin 9;
Servo _servo;
void setup() {
  Serial.begin(9600);
  Wire.begin(UNO_ADDRESS);
  Wire.onReceive(receiveEvent);
  _servo.attach(SERVO_PIN);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

Задаются основные параметры для сервопривода. Устанавливается основная настройка адреса для проводного подключения между платами Arduino

UNO и Arduino Robot. Применяются настройки для входного и выходного разъемов для ультразвукового датчика.

Для определения расстояния от ультразвукового датчика до объекта напишем функцию, которая будет считывать показания с дальномера и вычислять расстояния.

Листинг 2. Функция для получения данных ультразвукового датчика и вычисление дистанции в сантиметрах.

```
float getDistance()
{
  int duration;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  float distance_cm = (duration/2) / 29.1;
  return distance_cm;
}
```

На соответствующий провод, подсоединенный к ультразвуковому датчику подается напряжение 5 вольт, а затем с другого подсоединенного провода считываются показания датчика. Расстояние в сантиметрах рассчитывается исходя из известной величины скорости распространения звука в воздухе.

Теперь опишем процедуру сканирования, которая будет задействовать ультразвуковой датчик для определения расстояния и управлять сервомотором, для поворота датчика на заданный угол.

Необходимо принять во внимание также и тот факт, что угол, на который может быть повернут сервопривод не может превышать 180 градусов, относительно начального положения мотора.

На листинге 3 приведена функция для сканирования пространства роботом.

Листинг 3. Функция сканирования пространства

```
void scan() {
  left=0;  forward=0;  right=0;
  if(i == 0) {
   for(; i < 180; i++) {
      _servo.write(i);
      if(i>=0 && i<=45 && right==0){
        if(getDistance()<30){
          right=1;
        }
      } else if(i>=45 && i<=135 && forward==0) {
        if(getDistance()<25){
          forward=1;
        }
      } else if(i>=135 && i<=180 && left==0) {
        if(getDistance()<30){
          left=1;
        }
      }
    }
  }
}
```

В функции сканирования использует подключенный сервопривод и присоединенный к нему ультразвуковой датчик для сканирования местности слева, справа и спереди робота, управляя поворотом сервомотора. Для определения расстояния используется ранее рассмотренная функция getDistance, а для поворота мотора на заданный угол используется объект _servo, создаваемый при инициализации программы. В данном методе используются флаги left, right и forward для определения препятствия слева, справа и спереди робота соответственно.

Весь достижимый сектор, который может просканировать ультразвуковой датчик делится на 3 равные части и для каждого флага, описывающий конкретную сторону, используются соответствующие градусные меры, в пределах которых может быть какое-либо препятствие. Если препятствие было определено, то этот флаг устанавливается в 1, иначе по умолчанию флаг равен 0.

Функция onRecieve, представляющая собой обработчик событий, реагирует на получение данных и позволяет получить необходимые данные или отправить данные по Bluetooth на удаленный сервер. Пример передачи данных другому микроконтроллеру показан на листинге 4.

Листинг 4. Передача данных микроконтроллеру, отвечающего за передвижение.

```
Wire.beginTransmission(ARDUINO_ROBOT_ADDRESS);
Wire.write(data);
Wire.endTransmission();
```

Теперь рассмотрим основные моменты программы для микроконтроллера на базе Arduino Robot. Основная настройка робота осуществляется в функции setup и показана на листинге 5.

Листинг 5. Основная настройка Arduino Robot.

```
const int UNO_ADDRESS=8;
const int ROBOT_ADDRESS=9;
int echoPin = D5;
int trigPin = D4;
void setup() {
  Robot.begin();
  Robot.beginSpeaker();
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Wire.begin(ROBOT_ADDRESS);
  Wire.onReceive(receiveEvent);
}
```

Задаются переменные, определяющие адреса Arduino Robot и Arduino UNO. Запускается робот, настраивается соединение и определяется обработчик onRecieve, который будет выполнять указанную функцию в момент получения данных.

Поскольку робот выполняет указания, которые приходят через Bluetooth и передаются через проводное соединение с Arduino UNO, то основная функция loop будет пустой, а все действия будут выполняться по событию в функции onRecieve. На листинге 6 представлена функция отвечающая за событие получения данных.

Листинг 6. Обработчик события получения данных через проводное соединение с Arduino UNO.

```
void receiveEvent(int bytes) {
    message="";
  while(Wire.available()) {
    char c = Wire.read();
    message += c;
  }
  if(message[0]=='t'){
      moveForward();
      Wire.beginTransmission(UNO_ADDRESS);
      Wire.write('&'+dir);
      Wire.endTransmission();
  } else {
      changeDir(message.toInt());
      Wire.beginTransmission(UNO_ADDRESS);
      Wire.write('d');
      Wire.endTransmission();
  }
}
```

Функция срабатывает в момент, когда были переданы некоторые данные и выполняет определенную команду в зависимости от данных, которые поступили, среди которых определены движение вперед с помощью функции moveForward и поворот на заданное направление с помощью функции changeDir.

Функция ChangeDir следит за указателем направления робота относительно начального положения и выполняет поворот на указанное направление, в случае если клетка, в которую необходимо осуществить передвижение находится не спереди робота. После поворота робот проезжает прямо.

Процедура передвижения робота по прямой траектории показана на листинге 7.

Листинг 7. Процедура передвижения робота по прямой траектории

```
void moveForward(){
  float timeToPass=16.0/30.0;
  startPointTime=micros()/1000000;
  while((micros()/1000000)<(startPointTime+timeToPass))
  {
    Robot.motorsWrite(127,127);
  }
  Robot.motorsStop();
}
```

Расстояние, которое необходимо пройти для передвижения вперед составляет 20 сантиметров, что равносильно 1 клетке на сетке, и определяется временем, в течении которого подается напряжение на моторы робота. Время вычисляется в зависимости от расстояния, которое способен проехать робот при соответствующем напряжении за единицу времени.

Поворот робота на заданный градус осуществляется подачей положительного и отрицательного значения на разные моторы в функцию motorsWrite, которая определена библиотекой Arduino Robot.

После выполнения команд робот сообщает Arduino UNO об успешном выполнении команды и ожидает следующих указаний.

ГЛАВА 4. ТЕСТИРОВАНИЕ МОБИЛЬНОЙ СИСТЕМЫ

4.1 Методика тестирования

Чтобы корректно протестировать работу мобильной системы, необходимо задать определенные ограничения к объектам в замкнутом пространстве и месту проведения испытаний. Поскольку конструкция робота не рассчитана на передвижение по пересеченной местности, то поверхность, по которой будет ездить робот должна быть гладкой и не иметь ям или других мелких препятствий, способных задержать робота или изменить его курс при передвижении или повороте.

Для того что бы получать корректные показатели используя ультразвуковой датчик выбранной модели, необходимо учесть некоторые ключевые особенности, которые могут нарушить работу датчика и выдать неверные результаты. К примеру датчик может не зафиксировать расстояние, если объект будет слишком тонким. В этом случае датчик пропустит этот объект и вернет неверные показания. Аналогичные проблемы могут наблюдаться, если объект будет слишком пористым, мягким или если объект будет иметь структуру, способную поглотить звуковую волну, в таком случае датчик также не сможет определить расстояние до объекта.

Так же, препятствия могут быть не определены, если их высота будет меньше высоты, на которой закреплен датчик.

Некоторые ограничения накладываются, принимая во внимания технические характеристики Bluetooth-модуля. Так, например, несмотря на хорошую устойчивость Bluetooth-модуля к широкополосным помехам, рекомендуется убрать различные объекты, которые могут создать помехи для канала, по которому передается информация между Bluetooth-модулем и сервером. Расстояние между модулем и сервером не должно превышать 10 метров, иначе соединение прервется.

Учитывая все эти особенности, можно определить тип объектов, которые можно использовать в качестве препятствий при тестировании.

4.2 Тестирование робота

Протестируем работу мобильной системы в реальных условиях. Для этого необходимо соорудить некоторую закрытую область и разместить твердые объекты в качестве препятствий внутри. Так же необходимо помнить о том, что объекты, размещаемые внутри закрытой области должны быть твердыми, иметь высоту более 15 сантиметров, а также ширину более 5 сантиметров, что бы объекты могли быть безошибочно зафиксированы ультразвуковым датчиком расстояний и затем переданы удаленному серверу для проведения вычислений и определения дальнейшего маршрута для передвижения.

Установим робота в любую клетку, где нет препятствия. На рисунке 4.1 показано начальное положение робота в некотором закрытом пространстве с препятствиями.

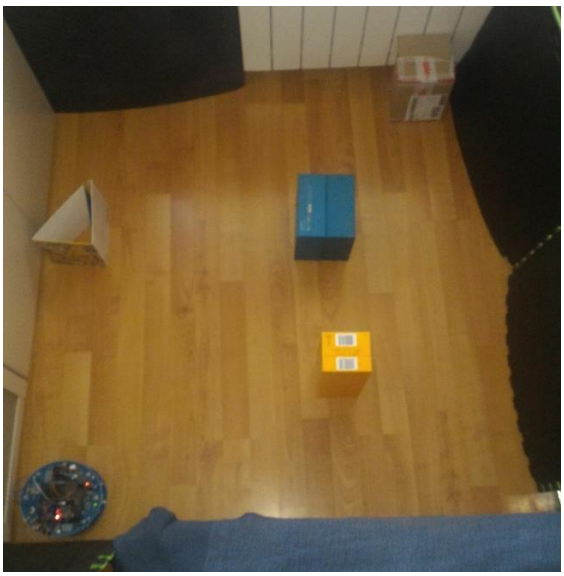

Рис. 4.1. Стартовая позиция робота в некоторой закрытой области с препятствиями.

Теперь запустим программу визуализации на сервере. Программа проинициализирует основные параметры и установит робота по умолчанию в

предполагаемой комнатой. При желании робота можно переместить в любую клетку на сетке в программе. В этом случае комната будет строится относительно стартовой позиции робота на сетке в программе и не будет отличаться.

На рисунке 4.2. изображено стартовое положение робота и интерфейс программы после запуска.

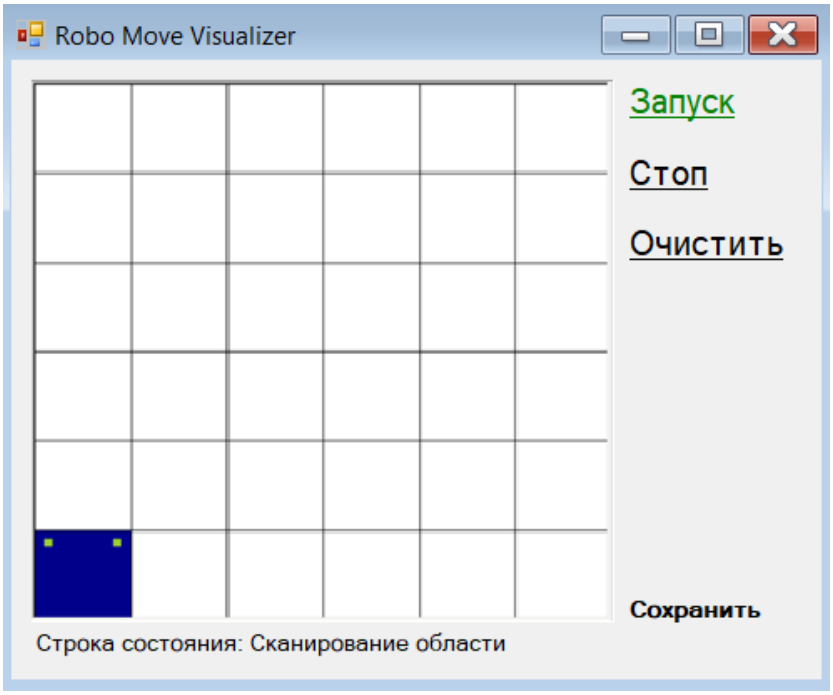

Рис. 4.2. Запуск программы и начальная инициализация

После того как робот был размещен в комнате, а программа была запущена, можно запускать работу системы. После этого будет установлено соединение между удаленным сервером и Bluetooth-модулем, подключенным к роботу. При этом необходимо активировать Bluetooth-модуль на удаленном сервере и убедиться, что в списке найденных устройств отображается установленный модуль. По умолчанию модуль будет иметь стандартное название своей модели, или отображаться как неизвестное устройство. Необходимо установить соединение с этим устройством. Как правило для подключения необходимо ввести стандартный пароль 1234, после чего модуль будет подключен к удаленному серверу и займет последовательный COM порт.

После запуска робота удаленный сервер подключится к Bluetooth-модулю и будет работать согласно описанному алгоритму. На рисунке 4.3. видно, как робот перемещается по комнате, постепенно исследуя ее.

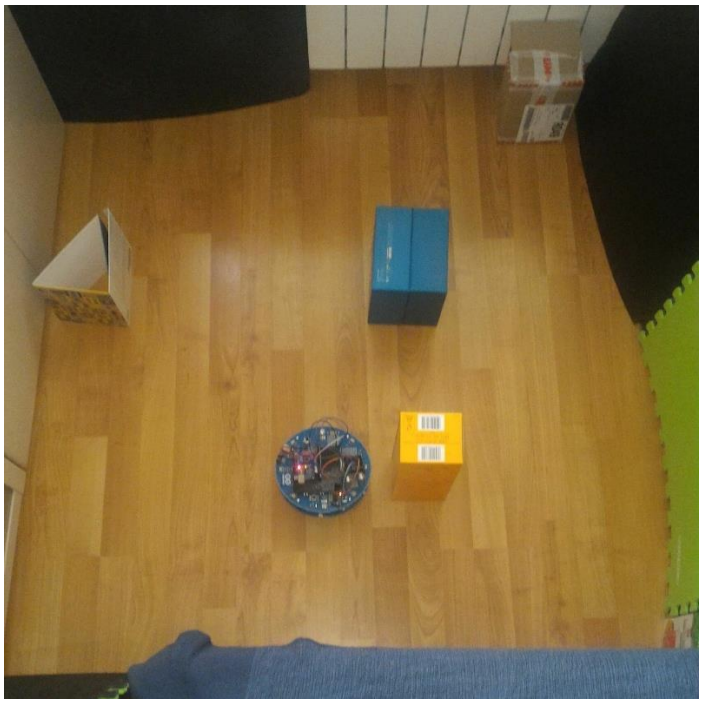

Рис. 4.3. Рисунок робота, перемещающегося по комнате, согласно алгоритму

На рисунке 4.4. показано, как отображается перемещение робота и исследуется область вокруг в программе-визуализаторе.

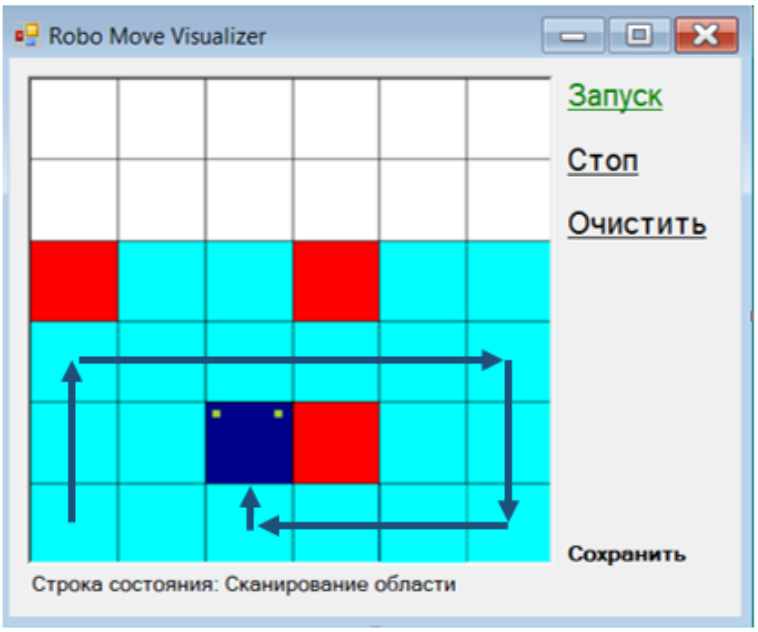

Рис. 4.4. Визуализация передвижения робота в программе

Теперь подождем пока робот завершит исследовать комнату с препятствиями и посмотрим на результат в программе. На рисунке 4.5. робот завершил свою работу и остановился.

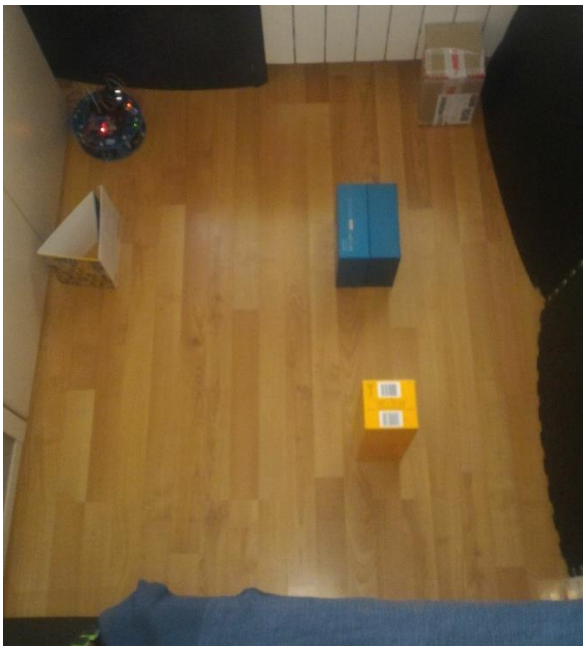

Рис. 4.6. Завершение работы построения карты препятствий

На рисунке 4.7 показывается конечный результат, а робот останавливается.

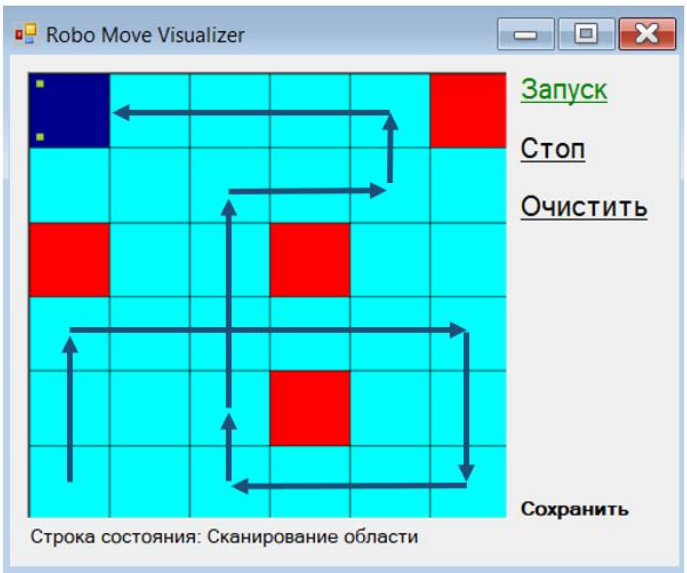

Рис. 4.7. Завершение работы построения карты препятствий

Как видно на рисунках 4.6 и 4.7 робот остановился и закончил свою работу. Так же видно, что робот посетил все неисследованные вначале места и корректно отобразил все препятствия на рисунке, что так же означает корректную работу реализованного алгоритма.

При этом на всех этапах тестирования положение робота соответствует его отображению в программе-визуализаторе, что говорит о правильной и отлаженной работе программы.

По результатам тестирования можно сказать что мобильная система готова к эксплуатации и реализует все функциональные возможности при соблюдении всех установленных ограничений.

ЗАКЛЮЧЕНИЕ

Цель данной выпускной квалификационной работы заключалась в проектировании и разработке мобильной системы, позволяющую строить схематическую двумерную карту окружающего пространства и решать задачу позиционирования модуля на ней.

В процессе исследования были сформированы следующие задачи:

- анализ проблемы построения карты препятствий и навигации на ней робота;
- выбор комплектующих и робототехнических модулей для создания мобильной системы;
- выбор инструментальных средств и технологий создания системы;
- реализация методов и алгоритмов для осуществления передвижения робота;
- реализация методов приема-передачи данных и их последующая обработка;
- тестирование мобильной системы.

Подробное исследование проблем реализации данной системы позволило сделать следующие выводы.

Задача о построении схемы замкнутого пространства с препятствиями, а также позиционирование робота в пространстве является тривиальной, однако имеет достаточно неопределенные способы и методы в реализации.

Мобильные системы, которые создают схемы и подробные карты препятствий могут быть использованы другими системами, которые используют готовые карты препятствий на области для других целей.

Был проведен анализ задачи построения 2-D схемы препятствий, определены основные функции мобильной системы и общий процесс построения карты препятствий мобильной системой. Так же было проведено исследование существующих программных и аппаратных компонентов. На основании этого

исследования был проведен их сравнительный анализ и осуществлен выбор наиболее подходящих компонентов, с учетом их стоимости и доступности.

Исследование показало, что на текущий момент задача позиционирования и построения карты описана в методе одновременной локализации и построения карты – SLAM, которые используется в мобильных и различных автоматизированных системах построения карт. Были определены основные виды плоских карт, процесс их построения и сложность технического процесса определения текущего местоположения. Так же сформированы проблемы, связанные с техническими ограничениями используемых модулей. Проведено исследование существующих методов и алгоритмов, позволяющих решать конкретные задачи в процессе исследования местности, согласно методу SLAM.

Проведена разработка мобильной системы и составление основных схем работы робота, удаленного сервера и общая схема с учетом их взаимодействия между собой. Затем была реализована программа для робота, а также для используемой в мобильной системе дополнительной платы. Так же был реализована программа для серверной части, с учетом указанных технических особенностей, с использованием выбранного алгоритма, который был модифицирован под требования текущих задач.

В конце проведено тестирование работоспособности мобильной системы и показаны результаты ее работы на каждом этапе.

По результатам тестирования было определено, что мобильная система спроектирована и реализована в полном соответствии с поставленными условиями. Работа выполнена в полном объеме, а результаты позволяют определить, что система готова к эксплуатации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SLAM – URL: http://robocraft.ru/blog/technology/724.html Дата обращения: 04.03.2018.

2. Rubin Frank. The Lee path connection algorithm // Computers, IEEE Transactions on. — 1974. — Vol. 100, no. 9. — Pp. 907–914.

3. Grisetti Giorgio et al. A tutorial on graph-based SLAM // Intelligent Transportation Systems Magazine, IEEE 2.4. — 2010. — Pp. 31–43.

4. A tutorial on graph-based SLAM / Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, Wolfram Burgard // Intelligent Transportation Systems Magazine, IEEE. — 2010. — Vol. 2, no. 4. — Pp. 31–43.

5. Ingber L. Simulated Annealing: Practice versus theory // Mathematical and Computer Modelling. 18(11). 1993. P. 2957.

6. Jeffrey Richter CLR via C#; СИНТЕГ - Москва, 2012. - 896 с.

7. Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza Introduction to Autonomous Mobile Robots. Second edition. Massachusetts Institute of Technology. 2004 С.13-50.

8. Simultaneous localization and mapping with sparse extended information filters / Sebastian Thrun, Yufeng Liu, Daphne Koller et al. // The International Journal of Robotics Research. — 2004. — Vol. 23, no. 7-8. — Pp. 693–716

9. Сергей Асмаков Интерфейс Bluetooth: разберемся с нюансами // КомпьютерПресс : журнал. — 2013. — 12 марта (№ 3 (279)). — С. 34—36. — ISSN 0868-6157.

10. Джерими Блум: Изучаем Arduino: инструметы и методы технического волшебства. Пер. с англ. - СПб.: БХВ-Петербург, 2015. - 336 с.

11. Ник Рендольф, Дэвид Гарднер, Майкл Минутилло, Крис Андерсон Visual Studio 2010 для профессионалов.– М.,Диалектика, 2011. – 1184 с.

12. Бьёрн Страуструп. Программирование: принципы и практика использования C++, исправленное издание = Programming: Principles and Practice Using C++. — М.: Вильямс, 2011. — С. 1248. — ISBN 978-5-8459-1705-8.

13. Эндрю Троелсен. Язык программирования C# 2010 и платформа .NET 4.0. – М.,Вильямс, 2010 г. .– 1392 с.

ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММЫ НА ARDUINO ROBOT

```
#include <ArduinoRobot.h>
#include <Wire.h>

const int UNO_ADDRESS=8;
const int ROBOT_ADDRESS=9;

int X=0;
int Y=0;

int centerX=60;
int centerY=60;

int maxX=-10000;
int maxY=-10000;
int minX=10000;
int minY=10000;

unsigned long startPointTime;
unsigned long endPointTime;


String message = "";
int echoPin = D5;
int trigPin = D4;

int dir=1;

int global_X=0;
int global_Y=1;

void setup() {
  Robot.begin();
  Robot.beginSpeaker();

  Serial.begin(9600);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  Wire.begin(ROBOT_ADDRESS);
  Wire.onReceive(receiveEvent);
```

```
}

void receiveEvent(int bytes) {
  message="";
  while(Wire.available()) {
    char c = Wire.read();
    Wire.beginTransmission(8);
    Wire.write('4');
    Wire.endTransmission();
    message += c;
  }
  if(message[0]=='t'){
      moveForward();
      Wire.beginTransmission(UNO_ADDRESS);
      Wire.write('&'+dir);
      Wire.endTransmission();
  } else {
      changeDir(message.toInt());
      Wire.beginTransmission(UNO_ADDRESS);
      Wire.write('d');
      Wire.endTransmission();
  }
}

void loop() {
}

void moveForward(){
  float timeToPass=16.0/30.0;
  startPointTime=micros()/1000000;
  while((micros()/1000000)<(startPointTime+timeToPass))
  {
    Robot.motorsWrite(127,127);
  }
  Robot.motorsStop();
}

void changeMinMaxCoord(){
  if(X<minX){
    minX=X;
  }
  if(X>maxX){
```

```
    maxX=X;
  }
  if(Y>maxY){
    maxY=Y;
  }
  if(Y<minY){
    minY=Y;
  }
}

void turn(int side){
  if(side==-1) {
        Robot.motorsWrite(-80,80);
        delay(1000);
  }else{
    Robot.motorsWrite(80,-80);
        delay(1000);
  }
  Robot.motorsStop();
}

void changeDir(int to){
  if(to-dir==1 || to-dir==-3){
    turn(1);
  }else if(to-dir==3 || to-dir==-1){
    turn(-1);
  }else{
    turn(1);
    turn(1);
  }
  dir=to;
}

void changeGlobals(int side){
  if(global_X==0){
    if(global_Y==1 && side==1){
      global_X=-1;
      global_Y=0;
    }else if (global_Y==1 && side==-1){
      global_X=-1;
      global_Y=0;
    }else if (global_Y==-1 && side==1){
```

```
      global_X=-1;
      global_Y=0;
    }
    else{
      global_X=-1;
      global_Y=0;
    }
  } else if(global_Y==0){
    if(global_X==1 && side==1){
      global_X=0;
      global_Y=-1;
    }else if(global_X==1 && side==-1){
      global_X=0;
      global_Y=1;
    }else if(global_X==-1 && side==1){
      global_X=0;
      global_Y=1;
    }else if(global_X==-1 && side==-1){
      global_X=0;
      global_Y=-1;
    }
  }
}
```

ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММЫ НА ARDUINO UNO

```
#include <Wire.h>
#include <Servo.h>

const int UNO_ADDRESS=8;
const int ROBOT_ADDRESS=9;
const int SERVO_PIN=A0;
char character;
String message = "";
int i = 0;

float left=0;
float forward=0;
float right=0;
int ledPin = 13;
#define echoPin 8
#define trigPin 9

Servo _servo;

void setup() {
  Serial.begin(9600);
  Wire.begin(UNO_ADDRESS);
  Wire.onReceive(receiveEvent);
  _servo.attach(SERVO_PIN);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void receiveEvent(int bytes) {
  message="";
  while(Wire.available()) {
    char c = Wire.read();
    Serial.println(c);
    message += c;
  }
  if(message[0]=='d'){
    scan();
    if(forward==0){
    Wire.beginTransmission(ROBOT_ADDRESS);
      Wire.write('t');
```

```
      Wire.endTransmission();
    } else {
      Serial.println('n');
      Wire.beginTransmission(ROBOT_ADDRESS);
      Wire.write('f');
      Wire.endTransmission();
    }
  }
  if(message[0]=='&'){
    scan();
    Serial.println(message+'|'+left+'|'+forward+'|'+right);
  }
  if(message[0]=='n'){
  Serial.println(message);
  }
}

void loop() {
  while(Serial.available()) {
    character = Serial.read();
    Serial.println(character);
    if(character == '1'){
      Serial.println('i');
      Wire.beginTransmission(9);
      Wire.write('1');
      Wire.endTransmission();
    } else {
      Wire.beginTransmission(ROBOT_ADDRESS);
      Wire.write('1');
      Wire.endTransmission();
    }
  }
}
void scan() {
  left=0;
  forward=0;
  right=0;
  if(i == 0) {
   for(; i < 180; i++) {
      _servo.write(i);
      if(i>=0 && i<=45 && right==0){
        if(getDistance()<20){
```

```
          right=1;
        }
      }else if(i>=45 && i<=135 && forward==0){
        if(getDistance()<20){
          forward=1;
        }
      }else if(i>=135 && i<=180 && left==0){
        if(getDistance()<20){
          left=1;
        }
      }
   }
  } else {
    for(; i > 0; i--) {
      _servo.write(i);
      if(i>=0 && i<=45 && right==0){
        if(getDistance()<20){
          right=1;
        }
      }else if(i>=45 && i<=135 && forward==0){
        if(getDistance()<20){
          forward=1;
        }
      }else if(i>=135 && i<=180 && left==0){
        if(getDistance()<20){
          left=1;
        }
      }
    }
  }
}

float getDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(trigPin, LOW);
  float duration = pulseIn(echoPin, HIGH);
  float distance = duration / 58.2;
  return distance;
}
```

ПРИЛОЖЕНИЕ В. ЛИСТИНГ ПРОГРАММЫ НА WINDOWS FORMS

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Diagnostics;
using System.Collections.Generic;

namespace RoboMoveVisualisation{
     public partial class Form1 : Form     {
public Form1(){
InitializeComponent();
}
int numOfCells = 20;
int cellSize;
int mode = 0;

int[][] map;
struct Robo{
public int x;
public int y;
public int direction;
public void turnRight(){
this.direction = (this.direction + 1) % 4;
}
public void turnLeft(){
this.direction = (3 - -this.direction) % 4;
}
};
Bitmap bmp, bmp2;
Graphics g, r;
Robo robo;
private void Form1_Load(object sender, EventArgs e){
clear();
}
private void linkLabel3_LinkClicked(object sender, LinkLabelLinkClickedEventArgs
e){
this.mode = 0;
this.stop = false;
this.moveRobotAlgorithm();
}
```

```
int cantMoveCount = 0;
bool isBasicMove = true;
bool isCycled = false;
bool stop = false;

private void moveRobotAlgorithm()
{
map[robo.y][robo.x] = 2;
while (!scanDone())
{
isCycled = false;
cantMoveCount = 0;
this.isBasicMove = true;

if (!baseMove()){
this.isBasicMove = false;
anotherWay();}}
Console.WriteLine("Done!");}
 private void anotherWay(){
List<Cell> path = new List<Cell>();
path = this.FindWave(robo.x, robo.y);
if (goToPoint(path) == false && !stop){
anotherWay();}else{
moveRobotAlgorithm();}}
private Boolean goToPoint(List<Cell> path){
foreach (Cell cell in path){
scan();
setRobotDirection(cell);
if (canMove()){
move();
scan();}else{
return false;}
}return true;}
 private Boolean baseMove(){
while (!scanDone()){
if (++cantMoveCount == 5) { break; }
scan();if (canMove() && !isCycled){
cantMoveCount = 0;
move();
scan();}}

if (scanDone()){
```

```
return true;}
return false;}
private void scan(){
scanNearestPositions();}

private void scanNearestPositions()
{
int roboCenterX = robo.x * cellSize + cellSize / 2;
int roboCenterY = robo.y * cellSize + cellSize / 2;

bool up;
bool right;
bool down;
bool left;

if(robo.x != numOfCells -1)
{
right = checkIsObstacleSquare(roboCenterX + cellSize, roboCenterY);
if (right)
{
map[robo.y][robo.x + 1] = 3;
this.fillSquare(Color.Red, robo.x+1, robo.y);
}
else if (map[robo.y][robo.x + 1] != 2)
{
this.fillSquare(Color.Aqua, robo.x+1, robo.y);
map[robo.y][robo.x + 1] = 1;
}

if (robo.y != numOfCells-1)
{
down = checkIsObstacleSquare(roboCenterX, roboCenterY + cellSize);
if (down)
{
     map[robo.y+1][robo.x] = 3;
     this.fillSquare(Color.Red, robo.x, robo.y + 1);
}
else if (map[robo.y + 1][robo.x] != 2)
{
     this.fillSquare(Color.Aqua, robo.x, robo.y+1);
     map[robo.y+1][robo.x] = 1;}
}
```

```
if (robo.y != 0)
{
up = checkIsObstacleSquare(roboCenterX, roboCenterY - cellSize);
if (up)
{
     map[robo.y-1][robo.x] = 3;
     this.fillSquare(Color.Red, robo.x, robo.y - 1);
}
else if (map[robo.y -1][robo.x] != 2)
{
     this.fillSquare(Color.Aqua, robo.x, robo.y-1);
     map[robo.y-1][robo.x] = 1;}}
if (robo.x != 0)
{
left = checkIsObstacleSquare(roboCenterX - cellSize, roboCenterY);
if (left){
     map[robo.y][robo.x-1] = 3;
     this.fillSquare(Color.Red, robo.x-1, robo.y);}
else if (map[robo.y][robo.x - 1] != 2){
     this.fillSquare(Color.Aqua, robo.x-1, robo.y);
     map[robo.y][robo.x-1] = 1;}}}
else if (robo.x != 0)
{
left = checkIsObstacleSquare(roboCenterX - cellSize, roboCenterY);
if (left)
{
map[robo.y][robo.x - 1] = 3;
this.fillSquare(Color.Red, robo.x-1, robo.y);
}
else if (map[robo.y][robo.x - 1] != 2)
{
this.fillSquare(Color.Aqua, robo.x-1, robo.y);
map[robo.y][robo.x - 1] = 1;}
if (robo.y != numOfCells - 1){
down = checkIsObstacleSquare(roboCenterX, roboCenterY + cellSize);
if (down){
     map[robo.y + 1][robo.x] = 3;
     this.fillSquare(Color.Red, robo.x, robo.y+1);
}
else if (map[robo.y + 1][robo.x] != 2){
     this.fillSquare(Color.Aqua, robo.x, robo.y+1);
```

```
     map[robo.y + 1][robo.x] = 1;}}

if (robo.y != 0){
up = checkIsObstacleSquare(roboCenterX, roboCenterY - cellSize);
if (up){
     map[robo.y - 1][robo.x] = 3;
     this.fillSquare(Color.Red, robo.x, robo.y-1);
}
else if (map[robo.y -1][robo.x] != 2){
     this.fillSquare(Color.Aqua, robo.x, robo.y-1);
     map[robo.y - 1][robo.x] = 1;}}
 if (robo.x != numOfCells-1){
right = checkIsObstacleSquare(roboCenterX + cellSize, roboCenterY);
if (right){
     map[robo.y][robo.x + 1] = 3;
     this.fillSquare(Color.Red, robo.x+1, robo.y);}
else if(map[robo.y][robo.x + 1] != 2){
     this.fillSquare(Color.Aqua, robo.x+1, robo.y);
     map[robo.y][robo.x + 1] = 1;}}}}
 public Boolean checkIsObstacleSquare(int x, int y){
if (x > 0 && x <= pictureBox1.Width && y > 0 && y <= pictureBox1.Height){
return bmp.GetPixel(x, y).Name == HexConverter(Color.Red);}
else return false;}

public Boolean checkIsScanedSquare(int x, int y)
{
if (x > 0 && x <= pictureBox1.Width && y > 0 && y <= pictureBox1.Height)
{
return bmp.GetPixel(x, y).Name == HexConverter(Color.Aqua);
}
else return false;
}

private Boolean canMove()
{
if (stop)
{
return false;
}

int roboCenterX = robo.x * cellSize + cellSize / 2;
int roboCenterY = robo.y * cellSize + cellSize / 2;
```

```
        switch (robo.direction){
        case 0:
        if(robo.y ->= 0 && !checkIsObstacleSquare(roboCenterX, roboCenterY - cellSize)){
             if (isBasicMove && !checkIsScanedSquare(roboCenterX, roboCenterY - cellSize
 - cellSize)){
        return true;
             } else if(!isBasicMove){
        return true;}
        } break;
        case 1:if (robo.x + 1 <= numOfCells - 1 && !checkIsObstacleSquare(roboCenterX +
cellSize, roboCenterY)){
             if (isBasicMove && !checkIsScanedSquare(roboCenterX + cellSize + cellSize,
roboCenterY)){return true;}
             else if (!isBasicMove){
        return true;}}
        break;
        case 2:
        if (robo.y + 1 <= numOfCells - && !checkIsObstacleSquare(roboCenterX, roboCenter
Y + cellSize)){
             if (isBasicMove && !checkIsScanedSquare(roboCenterX, roboCenterY + cellSize
 + cellSize)){return true;}else if (!isBasicMove){return true;}}
        break;
        case 3:
        if (robo.x -
1 >= 0 && !checkIsObstacleSquare(roboCenterX - cellSize, roboCenterY)){
             if (isBasicMove && !checkIsScanedSquare(roboCenterX - cellSize - cellSize,
roboCenterY))
             {
        return true;
             }
             else if (!isBasicMove)
             {
        return true;
             }
        }
        break;
        }

        if (isBasicMove)
        {
        robo.turnRight();
```

```
        }

        drawRobo();
        return false;
        }

        private void _pause(int value)
        {
        Stopwatch sw = new Stopwatch();
        sw.Start();
        while (sw.ElapsedMilliseconds < value)
        Application.DoEvents();
        }

        private void move()
        {
        _pause(160); // thats is used to presentate how the robot moving and discover ma
p
        this.fillSquare(Color.Aqua, robo.x, robo.y);
        map[robo.y][robo.x] = 2;

        switch (robo.direction)
        {
        case 0:
        if (map[robo.y - 1][robo.x] != 2 || !isBasicMove)
        {
             robo.y -= 1;
        }
        else { isCycled = true; }
        break;
        case 1:
        if (map[robo.y][robo.x+1] != 2 || !isBasicMove)
        {
             robo.x += 1;
        }
        else { isCycled = true; }
        break;
        case 2:
        if (map[robo.y + 1][robo.x] != 2 || !isBasicMove)
        {
             robo.y += 1;
        }
```

```
        else { isCycled = true; }
        break;
        case 3:
        if (map[robo.y][robo.x - 1] != 2 || !isBasicMove)
        {
             robo.x -= 1;
        }
        else { isCycled = true; }
        break;
        }

        drawRobo();
        }

        private Boolean scanDone()
        {
        if (stop)
        {
        return true;
        }

        for (int i = 0; i < numOfCells; i++)
        {
        for (int j = 0; j < numOfCells; j++)
        {
        if (map[i][j] == 0)
             return false;
        }
        }
        return true;
        }
private dynamic FindWave(int startJ, int startI)
        {
        int targetI = startI;
        int targetJ = startJ;

        bool pathFound = false;
        int step = 0;
        int[][] waveMap = new int[numOfCells][]; // 0 - unknown area, 1 - scanned area,
2 - area with obstacle

        for (int i = 0; i < numOfCells; i++)
```

```
        {
        waveMap[i] = new int[numOfCells];
        for (int j = 0; j < numOfCells; j++)
        {

        switch (map[i][j])
        {
             case 3: waveMap[i][j] = -4; break; // стена
             case 2: waveMap[i][j] = -3; break; // посещена
             case 1: waveMap[i][j] = -2; break; // просканирована
             default: waveMap[i][j] = -1; break; // не посещена
        }
        }
        }

        waveMap[startI][startJ] = 0;

        while (!pathFound)
        {
        for (int i = 0; i < numOfCells; i++)
        {
        for (int j = 0; j < numOfCells; j++)
        {
             if (waveMap[i][j] == step)
             {
             if (j - 1 >= 0 && waveMap[i][j - 1] != -4) // left
        {
        if(waveMap[i][j - 1] == -1)
        {
        targetI = i;
        targetJ = j-1;
        pathFound = true; waveMap[i][j - 1] = step + 1;                                  }
       else if (waveMap[i][j - 1] < 0) {      waveMap[i][j - 1] = step + 1; }}
             if (j + 1 < numOfCells && waveMap[i][j + 1] != -
4) // right{if (waveMap[i][j + 1] == -1) {
        targetI = i;
        targetJ = j + 1;
        pathFound = true;
        waveMap[i][j + 1] = step + 1;
        }
        else if (waveMap[i][j + 1] < 0)
        {
```

```
waveMap[i][j + 1] = step + 1;
}
}
if (i - 1 >= 0 && waveMap[i - 1][j] != -4) // up
{
if (waveMap[i - 1][j] == -1)
{
targetI = i - 1;
targetJ = j;
pathFound = true;
waveMap[i - 1][j] = step + 1;
}
else if (waveMap[i - 1][j] < 0)
{
waveMap[i - 1][j] = step + 1;
}
}
if (i + 1 < numOfCells && waveMap[i + 1][j] != -4) // down
{
if (waveMap[i + 1][j] == -1)
{
targetI = i + 1;
targetJ = j;
pathFound = true;
waveMap[i + 1][j] = step + 1;
}
else if (waveMap[i + 1][j] < 0)
{
waveMap[i + 1][j] = step + 1;
}
}
     }
}
}
step++;

if (step > numOfCells * numOfCells)
{
pathFound = true;
fillUnreachableArea(waveMap);
}
}
```

```
        List<Cell> path = new List<Cell>();

        if (pathFound)
        {

        Cell c = new Cell {
        i = targetI,
        j = targetJ,
        weight = step
        };

        while (c.i != startI || c.j != startJ)
        {
        bool _checked = false;
        if (c.i - 1 >= 0 && waveMap[c.i - 1][c.j] == c.weight - 1) // check up
        {
             c.i = c.i - 1;
             c.weight = waveMap[c.i][c.j];
             _checked = true;
        }
        else if (c.j + 1 < numOfCells && waveMap[c.i][c.j + 1] == c.weight - 1) // check
right
        {
             c.j = c.j + 1;
             c.weight = waveMap[c.i][c.j];
             _checked = true;
        }
        else if (c.i + 1 < numOfCells && waveMap[c.i + 1][c.j] == c.weight - 1) // check
down
        {
             c.i = c.i + 1;
             c.weight = waveMap[c.i][c.j];
             _checked = true;
        }
        else if (c.j - 1 >= 0 && waveMap[c.i][c.j - 1] == c.weight - 1) // check left
        {
             c.j = c.j - 1;
             c.weight = waveMap[c.i][c.j];
             _checked = true;
        }
```

```
if (_checked)
{
     path.Add(c);
}
}
}
path.Reverse();
if(path.Count > 0)
path.RemoveAt(0);
return path;
}

bool isR = false;
private void fillUnreachableArea(int[][] waveMap)
{
for (int i = 0; i < numOfCells; i++)
{
for (int j = 0; j < numOfCells; j++)
{
if (waveMap[i][j] == -1)
{
     map[i][j] = 3;
     this.fillSquare(Color.Red, j, i);}}
     }
}
}
}
```