Содержание

Введение

Цель использования систем позиционирования для различных пользователей, в частности, для больниц и торговых центров, состоит в том, чтобы предоставлять навигационную помощь, улучшать бизнес процессы в различных сферах применения.

В настоящее время очень быстро развивается направление робототехники, стремительно появляются новые технологии и разрабатываются более совершенные роботы. Роботы-пылесосы, роботы для очистки бассейнов, игровые роботы и роботы телеприсутсвия с каждым днем становятся более доступными населению, тем самым спрос стремительно увеличивается.

Большинству роботов**,** которым необходима визуальная обратная связь для перемещения по рабочему пространству, с избеганием столкновений с препятствиями, для совместной работы с людьми, для определения местоположения рабочих частей , необходима система позиционирования, и задача по её разработке становится одной из наиболее важных, при их создании. Относительное недавнее появление направления вызвало интерес и соответственно привело к развитию различных технологий позиционирования.

1. Техническое задание

1.1 Назначение разработки и область применения

Разрабатываемая программно-аппаратная система предназначена для определения местоположения робота в помещении. Для работы системы необходимо спроектировать и собрать аппаратную часть системы, а также разработать программную составляющую.

Область применения разрабатываемой системы:

- Определение местоположения объекта в пространстве.

Разрабатываемая система предназначается для использования в робототехнике.

1.2 Технические требования

Рассмотрим, какой функциональностью должна обладать проектируемая программно-аппаратная система позиционирования робота в помещении:

1) Возможность использования данной системы, как с существующими проектами, так и при создании нового, на основе клиентской части.

2) Серверная часть определяет расстояние до клиентской части.

3) После определения расстояния, происходит вычисление координат положения клиентской части, то есть робота, и вывод полученной информации, в удобном виде, на экран компьютера или в систему управления робота, для управления движения.

2. Анализ технического задания

2.1 Выбор системы позиционирования робота

Первостепенной задачей анализа методов навигации является оценка точности позиционирования. В большинстве случаев, допустимая погрешность при решении данной задачи, должна быть меньше чем половина минимальных размеров робота. Кроме того из-за наличия помех или шума, окружающая робота среда вносит высокий уровень неточности в каналы связи и соответственно, в связи с этим возникает необходимость принимать во внимание способность системы функционировать в описанных условиях.

На данный момент выделяют четыре основных вида систем позиционирования глобальная, локальная, персональная и автономная. Рассмотрим каждую из них подробнее:

Глобальная система.

В основном глобальные системы позиционирования используются при движении по довольно длительным маршрутам. Это связано с их зависимостью от ситуаций использования, например, невозможно или в высокой степени сложно использовать данные системы внутри помещений, подземных сооружениях. Главной целью таких систем является определение конечных координат, т. е. широты и долготы. Примерами глобальных систем являются Beidou, GPS, Глонасс, Galileo, эти системы используют спутники для позиционирования. Точность данных систем зависит от многих факторов, но в условиях, приближенных к идеальным, наилучшая из данных систем, GPS, обеспечивающая погрешность всего в 60-90 см.

Персональная система.

Для позиционирования отдельных частей и взаимодействии с окружением, используется персональная система. Это актуально для устройств с манипуляторами. Данные системы могут применяться для позиционирования в рамках ограниченной территории, а также для следования по заданной, метками линии. Примером использования персональных систем можно назвать навигацию для робота-сборщика. Но в связи с узкой направленностью относительно заданной местности, и, следовательно, плохой адаптацией к изменениям условий, а также высокой стоимостью, использование данных систем вызывает затруднения.

Автономная система.

Автономные системы применяются, в случаях, когда осуществление приема и передачи сигналов извне осложнено или вовсе невозможно. Учитывая это, стоит отметить, что данные системы актуальны в условиях ограниченных пространств, так они имеют высокую зашумленность среды. Существенным их недостатком является чувствительность к неровностям

поверхности. Примерами таких систем навигации являются гироскопы и цифровые компасы.

Локальная система.

Применение данных систем целесообразно на достаточно больших, замкнутых площадках. Предметом позиционирования в локальных системах является некоторая стартовая точка. Данные локальной системы могут, для тактических беспилотных самолетов, работающих в рамках известной территории. Примером локальной системы можно назвать систему навигации A-GPS. На данный момент наиболее частое применение имеют системы, использующие дальномеры. Есть большое количество методов для обработки информации, которая поступает с дальномеров, например, потенциальное поле, триангуляция, трилатерация.[1]

2.2 Выбор метода определения расстояния

Выбор системы определения расстояния от робота до серверной части является очень важной задачей. Определение конкретного метода, основывается на следующих условиях: специфика разработки, навыки разработчика в робототехнике, а также доступность и простота реализации выбранного метода.

Метод потенциальных полей.

Метод потенциальных полей - алгоритм потенциального поля является одним из наиболее фундаментальных и традиционных алгоритмов, используемых для задачи определения расстояния. Суть данного алгоритма в том, что назначается искусственное потенциальное поле каждой точке, используя функции потенциального поля. Робот моделирует поле от самого высокого потенциала до самого низкого потенциала. Внутри системы создаются два вида искусственных потенциальных полей: положительное и отрицательное. Целевой узел обладает положительным полем, а препятствия в системе создают отрицательные поля. В данном методе робот через заданные интервалы времени, должен вычислять вектор, являющийся функцией целевой точки и препятствий в окружении. Полученный вектор формируются как суперпозиция векторов притягивания и отталкивания.

Проблема алгоритма потенциального поля – это проблема локальных минимумов. Это проблема возникает, когда все искусственные силы (положительные и отрицательные) нейтрализуют друг друга, например, в ситуациях, когда препятствие находится точно между роботом и целью, или когда препятствия расположены близко друг к другу.

Этот алгоритм имеет разумное временное решение, но он демонстрирует слабую способность преодолевать ловушку локальных минимумов и дает неоптимальные результаты. В зависимости от сценариев качество работы данного алгоритма может изменяться, так как алгоритм сложно реализовать в реальных сценариях использования. Из результатов

ясно, что существует компромисс между оптимальностью и вычислительными требованиями времени. Выбор алгоритма на практике должен основываться на эксплуатационных требованиях с точки зрения вычислительного времени и оптимальности.

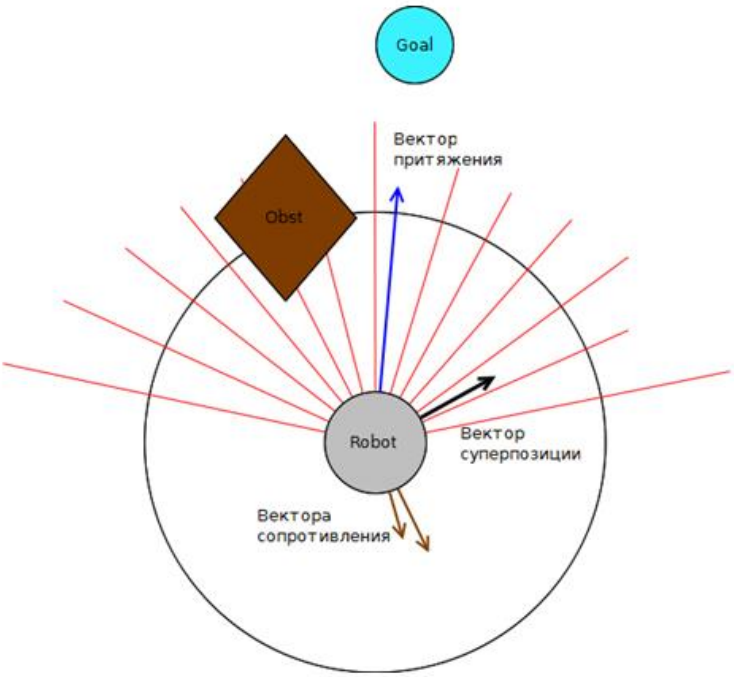

Рисунок 1. – Метод потенциальных полей

Метод триангуляции.

Триангуляция — один из научных методов создания опорной геодезической сети, решающий задачу по высокоточному измерению больших расстояний между заданными пунктами с построением примыкающих друг к другу треугольников и измерений внутри них.

Суть его состоит в построении целого ряда или сети соединенных треугольников и, соответственно, в определении положения их вершин в выбранной системе координат. В каждом из треугольников измеряют все углы, а одну из сторон определяют из вычислений путём последовательного решения предыдущих треугольников, начиная с того из них, в котором одна из его сторон получена в результате измерений. Длины сторон треугольника вычисляются по формуле синусов. Дирекционные углы вычисляются по измеренным горизонтальным углам.

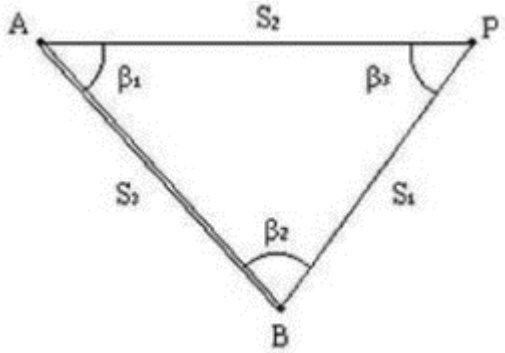

Рисунок 2. – Метод триангуляции

Метод трилатерации.

Данный метод позволяет определять положение на основе 3 известных расстояний на плоскости или 4 известных расстояний на сфере путем обработки этих точек в
качестве вершин одного треугольника или треугольников
из которого эти углы и стороны могут быть измерены.

Этот алгоритм очень важен, поскольку он используется практически всеми технологиями позиционирования, будь то в помещении или на улице. Раньше трилатерация мало использовалась по сравнению с триангуляцией (способом описанным выше) из-за сложности вычислений. Но разработка электронных приборов для измерения расстояния сделала трилатерацию общей и предпочтительной системой для решения данного класса задач. За исключением того, что измеряются только линии, а все углы вычисляются, полевые процедуры для трилатерации аналогичны процедурам для триангуляции.

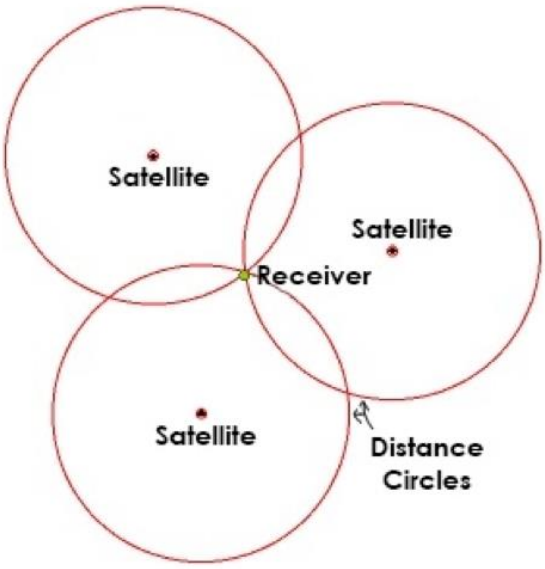

Рисунок 3. – Метод трилатерации

В разрабатываемой нами программно-аппаратной системе для определения расстояния между устройствами системы был выбран метод трилатерации, так как в данном методе нет избыточных измерений, тем самым, вычисление результата будет занимать меньше времени.

2.3 Выбор микроконтроллера

Выбор микроконтроллера является важнейшим решением, от которого зависит успешная реализация задуманного проекта. При его выборе необходимо отталкиваться от большого количества факторов.

Одной из основных задач считается выбор менее дорогостоящего микроконтроллера для снижения общей стоимости разрабатываемой системы, и удовлетворения требованиям по надежности и производительности. Общая стоимость системы включает все затраты от начала проектирования и до полной реализации проекта: Рассмотрим различные варианты существующих микроконтроллеров:

Atmel – лидирующая компания в области производства и разработки микроконтроллеров. Разрабатываемые в ней микроконтроллеры используются в большом количестве встраиваемых решений, 8-битные контроллеры серии megaAVR с AVR архитектурой положили начало платформе Arduino, сделавшей программирование и использование микроконтроллеров гораздо проще. Кроме AVR, компания производит микроконтроллеры на базе архитектур ARM и MCS-51. Но микроконтроллеры AVR всё-таки являются самым популярным продуктом производства Atmel, благодаря хорошему соотношению производительности, оптимизированности под программирование на языке C, энергоэффективности и цены. Также их можно программировать на собственном языке ассемблера. Первые 8-битные микроконтроллеры этой архитектуры были представлены в 1996 году, 32-битные AVR32 появились через 10 лет – в 2006 году. AVR имеет гарвардскую архитектуру (данные программы и переменных хранятся в разных адресных пространствах) и систему команд RISC (сокращенный набор команд, увеличивающий быстродействие). Вычислительное ядро, память и остальная периферия находятся на одном кристалле, благодаря чему микроконтроллеры AVR представляют собой SoC (System on chip, система на кристалле).

STMicroelectronics —микроэлектронная компания, занимающаяся разработкой, изготовлением и продажей различных полупроводниковых электронных компонентов.

Компания STMicroelectronics одной из первых вывела на рынок семейство микроконтроллеров на ядре ARM Cortex-M3 и на сегодняшний день по праву занимает лидирующее место среди производителей микроконтроллеров на этом ядре. Все началось в 2007 году с двух семейств — Performance Line (STM32F103) и Access Line (STM32F101). Компания постоянно работает как над расширением номенклатуры семейства, так и над улучшением характеристик, не забывая при этом также пополнять программную составляющую продукта. На сегодняшний момент STM32 уже состоит из более 10 линеек для всевозможных применений — микроконтроллеры с высокой производительностью, недорогие микроконтроллеры общего применения, микроконтроллеры с ультранизким

энергопотреблением, микроконтроллеры со встроенным радиомодулем для беспроводных решений, и все это — на одном ядре ARM Cortex-M3! Нельзя не отметить pin-to-pin и программную совместимость по всем линейкам.[4]

В данной работе, в качестве микроконтроллеров, были выбраны решения от компании Atmel, так как они обладают неплохой производительностью, большой разнообразностью в выборе, наиболее популярны и просты в использовании.

2.4 Выбор платформы для построения проекта

Так как ранее были выбраны микроконтроллеры производства Atmel, в качестве платформы будут использованы, построенные на их основе, контроллеры от Arduino.

Arduino — это удобная электронная платформа быстрой разработки электронных устройств, предназначенная как для новичков, так и профессионалов. Данная платформа имеет огромную популярность во всем мире благодаря удобству и простоте языка программирования, а также открытой архитектуре и программному коду. Устройство программируется через USB без использования программаторов.

Arduino позволяет компьютеру выйти за рамки виртуального мира в физический и взаимодействовать с ним. Устройства на базе Arduino могут получать информацию об окружающей среде посредством различных датчиков, а также могут управлять различными исполнительными устройствами.

Микроконтроллер на плате программируется при помощи языка Arduino (основан на языке Wiring) и среды разработки Arduino (основана на среде Processing). Проекты устройств, основанные на Arduino, могут работать самостоятельно, либо же взаимодействовать с программным обеспечением на компьютере (напр.: Flash, Processing, MaxMSP). Платы могут быть собраны пользователем самостоятельно или куплены в сборе. Программное обеспечение доступно для бесплатного скачивания. Исходные чертежи схем (файлы CAD) являются общедоступными, пользователи могут применять их по своему усмотрению.

Оригинальные версии Arduino производятся в Италии.

Используемые контроллеры:

1. Arduino UNO – наиболее распространенная стандартная плата, основанная на чипе Atmel ATmega328, имеющем на борту 32 КБ флэш-памяти, 2 Кб SRAM и 1 Кбайт EEPROM памяти. На периферии имеет 14 дискретных (цифровых) каналов ввода / вывода и 6 аналоговых каналов ввода / вывода, эти очень разносторонне-полезные девайсы, позволят покрывать большинство любительских задач в области микроконтроллерной техники. Чип ATmega16u2 на борту управляет последовательной связью. Данная плата контроллера является одной из самых дешевых и наиболее часто используемых.

2. Arduino Leonardo данный микроконтроллер, основанный на ATmega32u4, имеет расширенные возможности USB и, следовательно, не требует отдельного микрочипа для последовательной связи по USB, как это требуется в Uno. Следовательно, это ведет к уменьшению стоимости и означает, что разработчик может использовать микроконтроллер в качестве родного устройства USB, увеличивая гибкость при коммуникации с компьютером. Arduino Leonardo может эффективно эмулировать клавиатуру и мышь через USB HID.

3. Arduino Mega 2560 – многоканальная плата. В общей сложности каналов ввода – вывода у нее - 54 цифровых линий ввода / вывода и 16 аналоговых входов. Данная плата также имеет большое количество флэш-памяти: 256 КБ, что позволяет хранить большие программы, чем это позволяла Uno. Она также имеет немалую SRAM и EEPROM памяти: 8 КБ и 4 КБ и 4 аппаратных асинхронных UART порта. Данные платы используются там, где необходимо большое количество входов и выходов.

2.5 Выбор датчиков для определения расстояния

Так как в данном проекте могут использоваться различные подходы и решения, необходимо выбрать наиболее подходящий датчик измерения расстояния.

Ультразвуковой дальномер.

Самым грубым для измерения расстояния активного типа является ультразвуковой дальномер. В основе его работы лежит принцип эхо-локации, которым пользуются даже некоторые животные, например, дельфины. Устройство создает звуковой импульс и улавливает эхо – звуковые волны, которые отражаются от объекта.

Для точности измерения используется звук высокого диапазона частот – 40 кГц. Поскольку скорость звука известна, а время его движения несложно измерить, остается вычислить только расстояние, что и делает ультразвуковой дальномер.

На показания ультразвукового дальномера не воздействуют блики солнца или цвет объекта. В том числе и просвечивающая поверхность не будет для него преградой. Но имеется большая вероятность возникновения проблем с определением расстояния до крайне тонких и предметов имеющим неоднородную поверхность, например шерсть.

Лазерный дальномер.

Если тот же метод применить со световым импульсом, получится точный лазерный дальномер импульсного типа. Дело в том, что скорость света настолько высока (300 000 км/с), что для небольших расстояний, которые измеряются в строительстве (20, 30, 50 м), речь идет о долях наносекунд. Измерить время с такой точностью очень сложно. Главное преимущество такого устройства – оно посылает короткие световые

импульсы, а не постоянный луч. Это значит, что можно использовать лазер высокой мощности. Такой мощный импульс может без особых сложностей «слетать» туда и обратно на расстояние 100 км за доли секунд. Это свойство применяется чаще всего в военной отрасли, а сам прибор стоит гораздо дороже аналогов.

Инфракрасный дальномер.

Принцип работы лазерного дальномера фазового типа основан на сравнении и определении сдвига фазы световой волны. Устройство генерирует световой луч инфракрасного спектра. Луч движется с известной скоростью до цели измерения, отражается и возвращается. Инструмент сравнивает фазу световой волны в начале движения и в конце. Замер производится дважды, после чего устройство выдает результат в метрах. Одно из преимуществ такого вида измерителей расстояния – цена. Они значительно дешевле импульсных, ведь нет необходимости оборудовать лазерную рулетку сверхточным и дорогостоящим секундомером. Кроме того, при фазовом методе погрешность составляет не больше половины фазы, то есть меньше миллиметра. Это поразительный результат, однако есть у этого устройства и недостатки.

Так как светить приходится не короткими импульсами, а постоянно на протяжении всего измерения, установить мощный лазер не получится. А это значит, что на дальние расстояния устройство не применяется. Однако для строительства дальности его действия более чем достаточно.

Из-за наибольшей доступности и наименьшей стоимости были выбраны ультразвуковые датчики.

3. Разработка системы на структурном уровне

3.1 Общая структурная схема

Аппаратные средства системы позиционирования робота, в данной работе, состоят из двух частей: клиентской и серверной. Клиентской стороной является устройство с ультразвуковым передатчиком, устанавливающееся на робота, а серверной — три однотипных устройства с ультразвуковыми приёмниками.

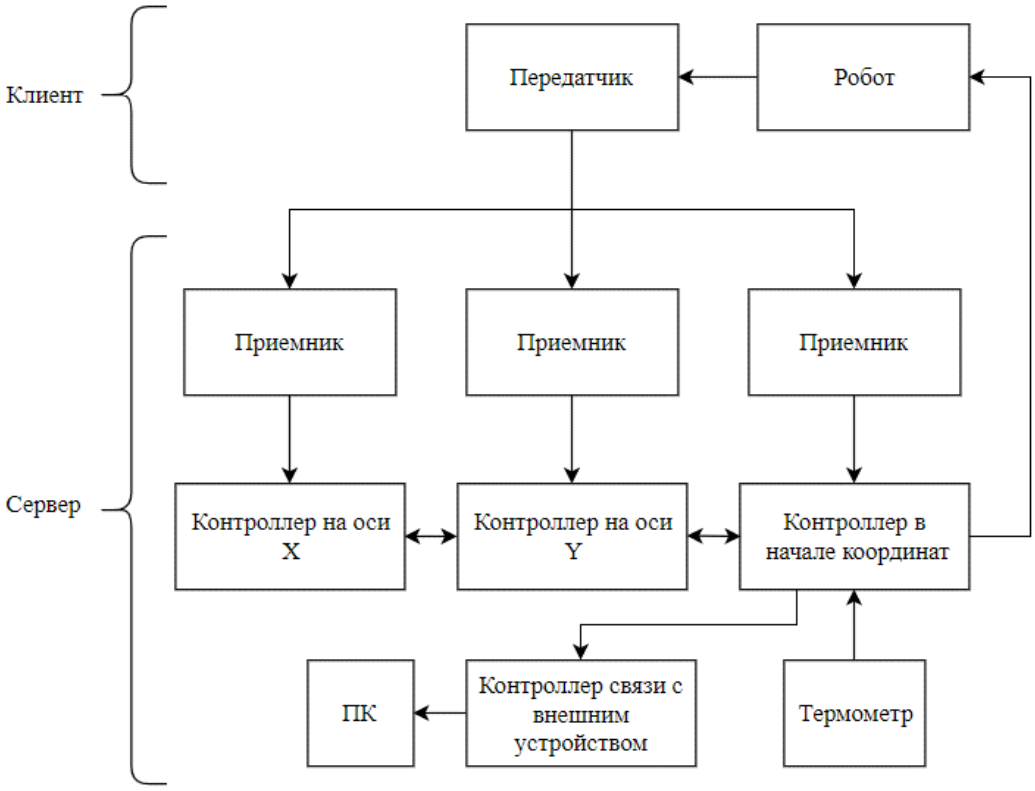

Рисунок 4. – Структурная схема системы

3.2 Клиентская часть

Клиентское устройство представляет из себя робота, который время от времени генерирует ультразвуковые сигналы, позволяющие серверной стороны вычислить его местоположение. Включает в себя: плату Arduino Mega 2560, модуль RS-485, для синхронизации отправки ультразвукового сигнала, и сам ультразвуковой дальномер работающий как передатчик. Работает данное устройство от двух аккумуляторов форм-фактора 18650. В качестве самого передатчика выступает отпаянный от ультразвукового модуля HC-SR04 трансмиттер.

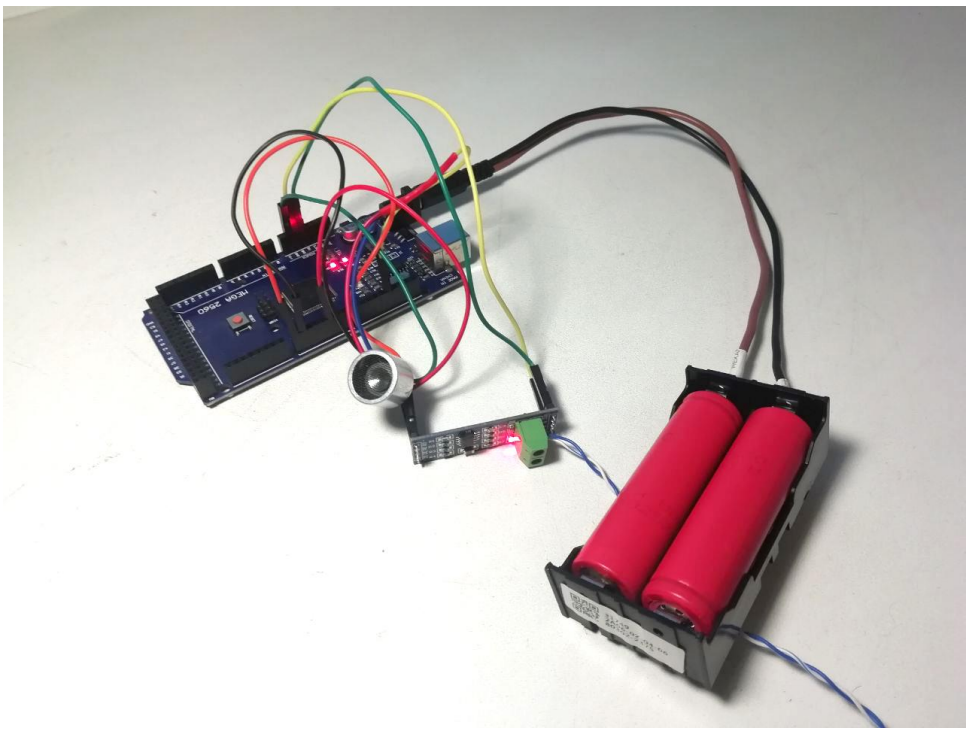

Рисунок 5. – Реализация клиентской части

3.3 Серверная часть

Серверная часть состоит из четырех устройств, три из которых состоят из Arduino Uno, ультразвукового дальномера, работающий как приёмник. По мимо того, устройство, находящееся в начале координат, также имеет аналоговый термометр LM35 и модуль RS-485 для соединения с клиентской стороной.

Четвертым устройством является Arduino Leonardo, которая соединяет остальные устройства серверной стороны и компьютер.

Вся серверная часть крепится на деревянную конструкцию, позволяющая разместить устройства в правильном положении для наилучшего приёма сигнала от робота.

Вся система запитывается от одного источника, что позволяет запускать все устройства одновременно.

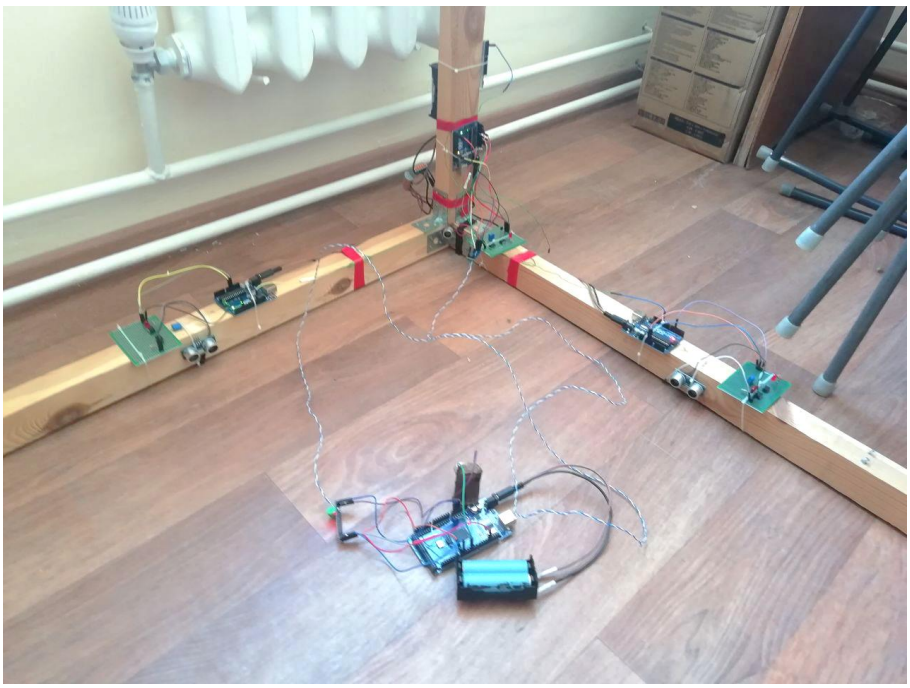

Рисунок 6. – Реализация серверной части

3.4 Реализация приемника

Корректная работа данной системы позиционирования напрямую зависит от правильности принятия сигнала, в противном случае, верно рассчитать расстояния просто-напросто не получится.

Для этого, из ранее использованного ультразвукового модуля HC-SR04, выпаивается ресивер, на котором будет основаться приёмник.

Для создания чистого сигнала используются звуковой усилитель мощности LM386 и компаратор напряжения LM393. Так же в схеме присутствуют следующие элементы: резисторы на 220 Ом и 10 кОм, подстроечный резистор на 10 кОм, конденсатор на 100 нФ и красный светодиод.

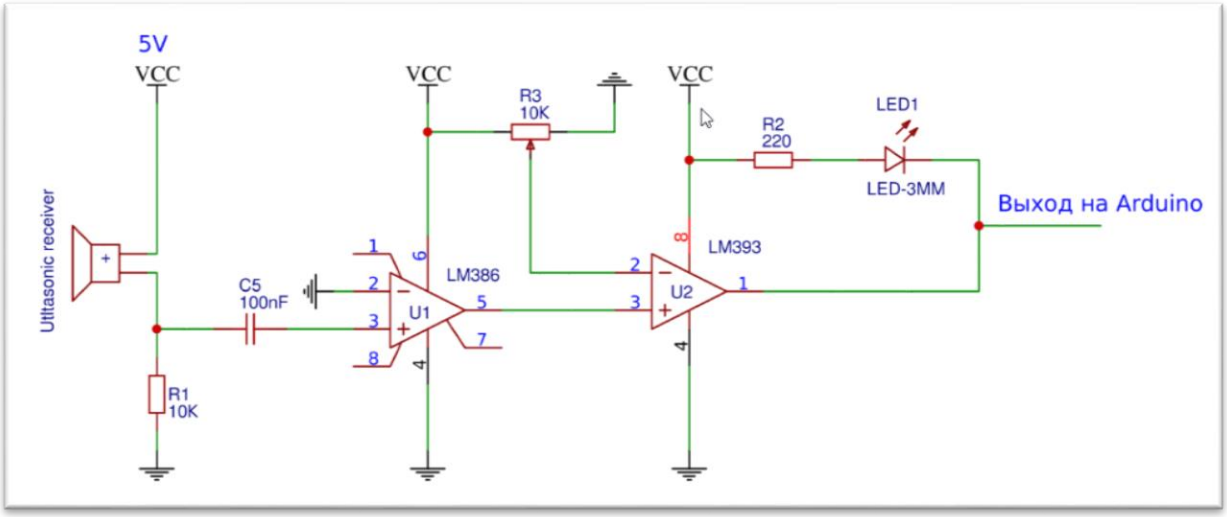

Рисунок 7. – Принципиальная схема приемника

Для тестирования работоспособности, данная схема была собрана на макетной плате, после чего уже распаяна на печатной макетной плате.

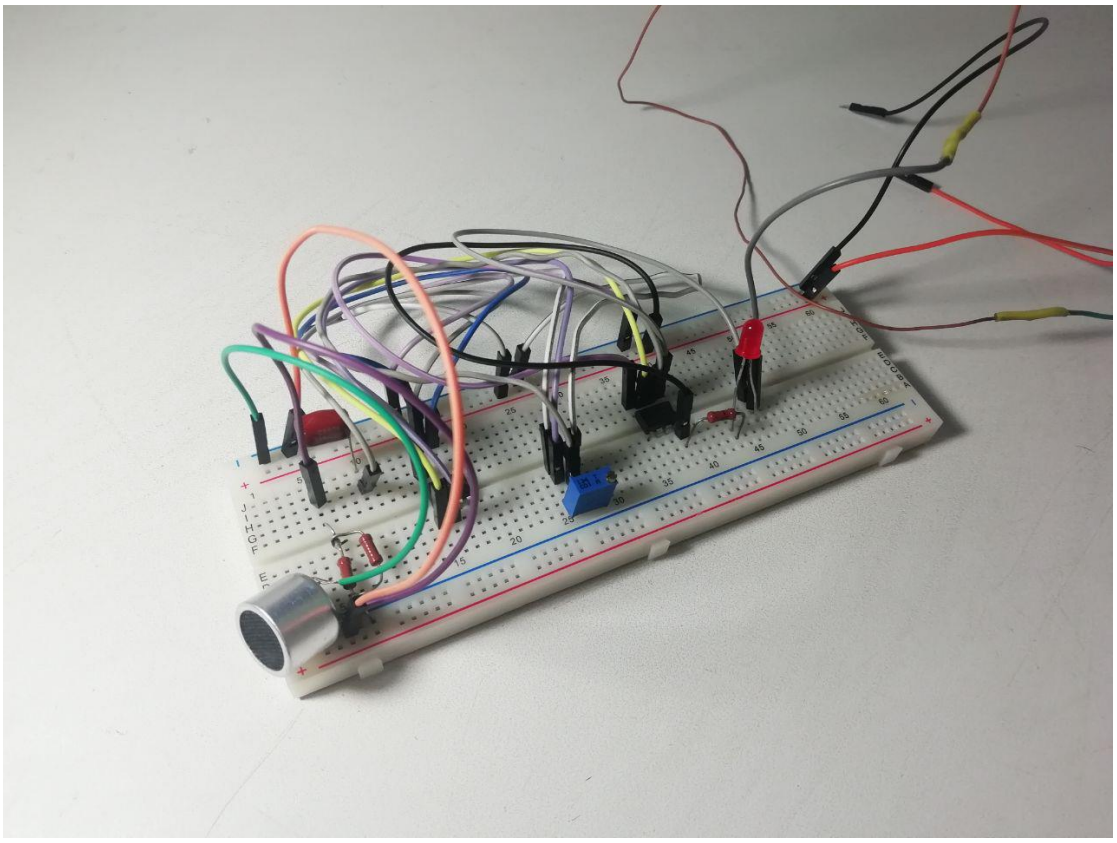

Рисунок 8. – Собранная схема приемника на макетной плате

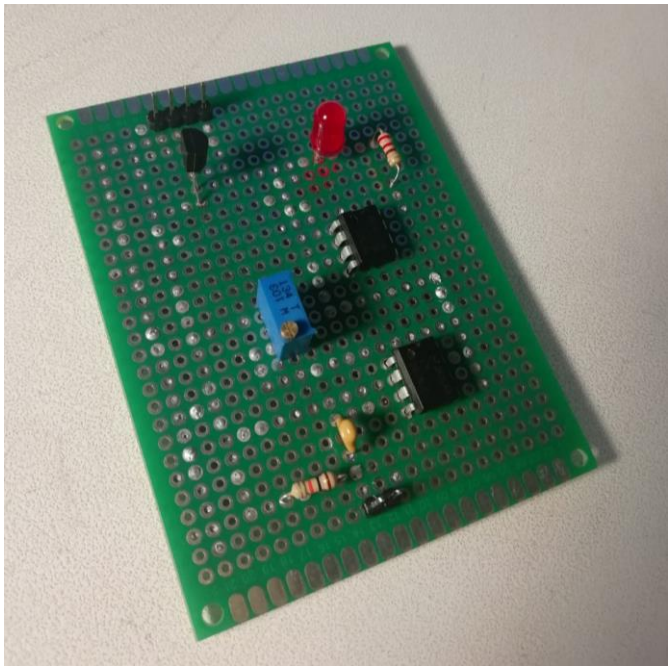

Рисунок 9. – Распаянная схема приемника на печатной макетной плате

На рисунке 9 изображен приёмник для устройства, находящегося в начале координат, так как на данной плате был также распаян термометр, для его жесткого фиксирования и улучшения показаний.

3.5 Способ связи между клиентом и сервером

Так как между клиентом и сервером может быть достаточно большое расстояния, то было принято решение использовать, для связи между ними, интерфейс RS-485.

RS-485 — это последовательный интерфейс, предком которого является RS-232. Последний получил известность из-за COM-портов старых компьютеров, которые как раз работали по интерфейсу RS-232. Максимальная длина линии при соединении по RS-485 составляет 1200 метров! А если на линии будут специальные усилители, то ещё больше. Конечно, скорость передачи по такому длинному проводу будет всего около 60 кб/с, но для передачи показаний датчиков больше и не требуется. В качестве кабеля для RS-485 используется витая пара. Это два провода, сплетенные друг с другом. Такой кабель ещё используется в Ethernet линиях, так что его легко можно достать. Чтобы передавать данные на дистанции более 500 метров, потребуется экранированная витая пара. К одному кабелю может быть подключено 32 устройства. Но только одно устройство может заниматься передачей данных, в определенный момент времени.

Для того, чтобы подключить две платы Arduino по интерфейсу RS-485 необходим специальный модуль, изображенный на рисунке 10.

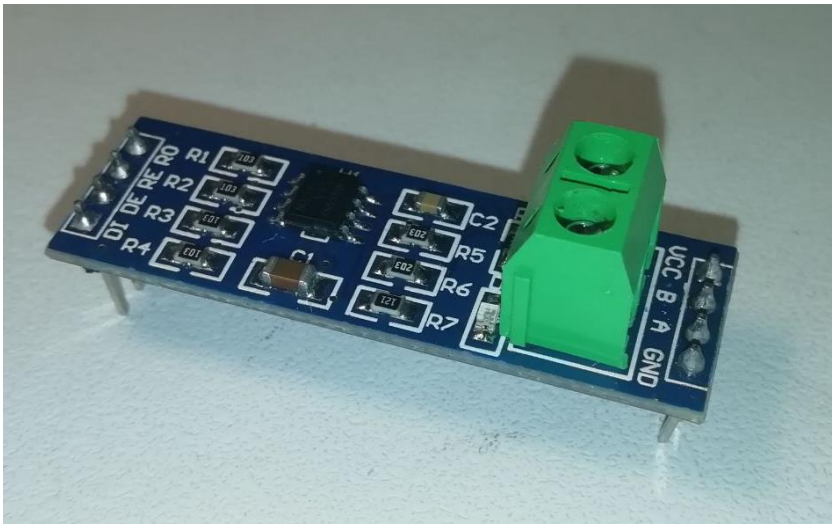

Рисунок 10. – Модуль RS-485

Для соединения двух контроллеров Arduino потребуется два таких модуля. Подключение модуля к контроллеру идентично для обоих устройств.

RS-485	Arduino
DI (Driver Input - вход передатчика)	TX (transmit - передать)
RO (Receiver Out - выход приёмника)	RX (receive - получить)
DE (Driver Enable - разрешение работы передатчика)	Любой цифровой вход
RE (Receiver Enable - разрешение работы приёмника)	Любой цифровой вход
Vcc (Voltage constant current - напряжение постоянного тока)	Vcc
GND (GrouND - земля)	GND

После подключения каждого из модулей с соответствующим контроллером, остается соединить эти модуля одноименными выводами: «A» с «A», «B» с «B».

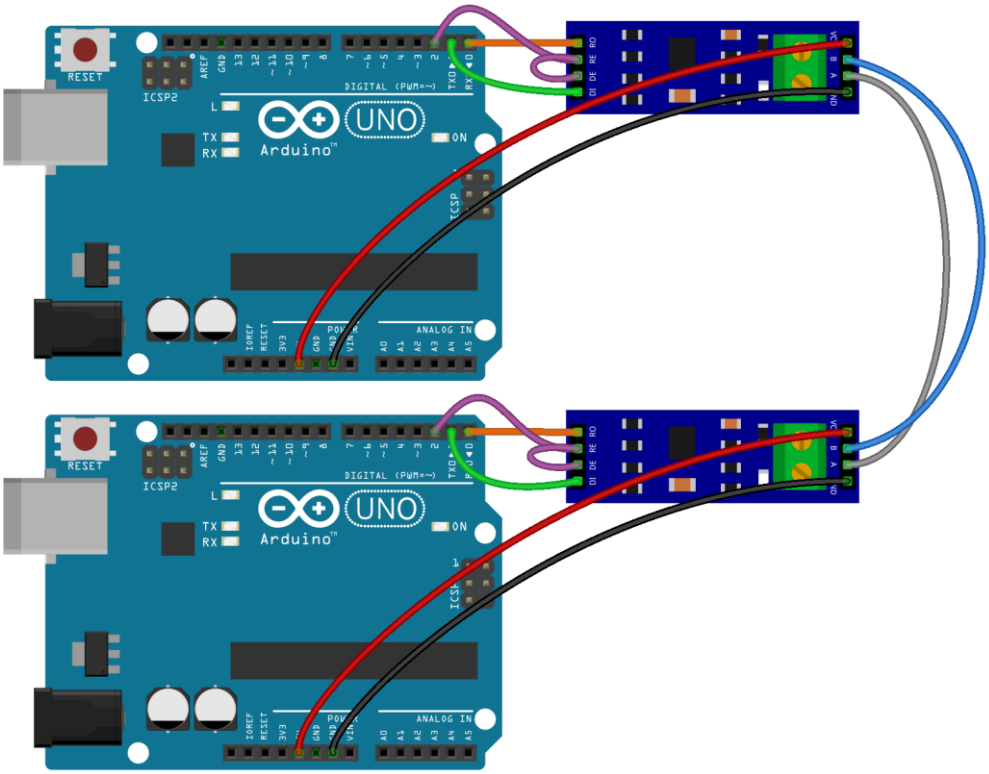

Рисунок 11. – Пример соединения двух Arduino по интерфейсу RS-485

3.6 Способ связи между устройствами серверной стороны

Устройства серверной стороны находятся на меньшей дистанции друг от друга, поэтому в данном случае целесообразнее применять интерфейс I2C.

Протокол Inter-Integrated Circuit (I 2C) - это протокол, предназначенный для обеспечения возможности связи нескольких ведомых (Slave) цифровых интегральных схем («микросхем») с одним или несколькими ведущими (Master) микросхемами.

Master - это устройство, которое всегда инициирует связь и управляет линией синхронизации (SCL). Обычно микроконтроллер или микропроцессор выполняет роль мастера, который должен считывать данные или записывать данные на подчиненные периферийные устройства.

Slave-всегда отвечают ведущему и не будут инициировать какую-либо связь самостоятельно. Обычно такие устройства, как EEPROM, LCD, RTC, действуют как подчиненные устройства. Каждое подчиненное устройство будет иметь уникальный адрес, так что мастер может запрашивать данные или записывать данные на него. Адрес устройства указывается в паспорте (datasheet). К одной шине I2C может быть подключено до 127 устройств, в том числе несколько ведущих. К шине можно подключать устройства в процессе работы, т.е. она поддерживает «горячее подключение».

Как и последовательный периферийный интерфейс (SPI), он предназначен только для связи на короткие расстояния в пределах одного устройства. Подобно асинхронным последовательным интерфейсам (таким как RS-232 или UART), для обмена информацией требуется только две двунаправленные линии связи: SDA и SCL. Эти шины присоединяются к шинам питания с помощью резисторов.

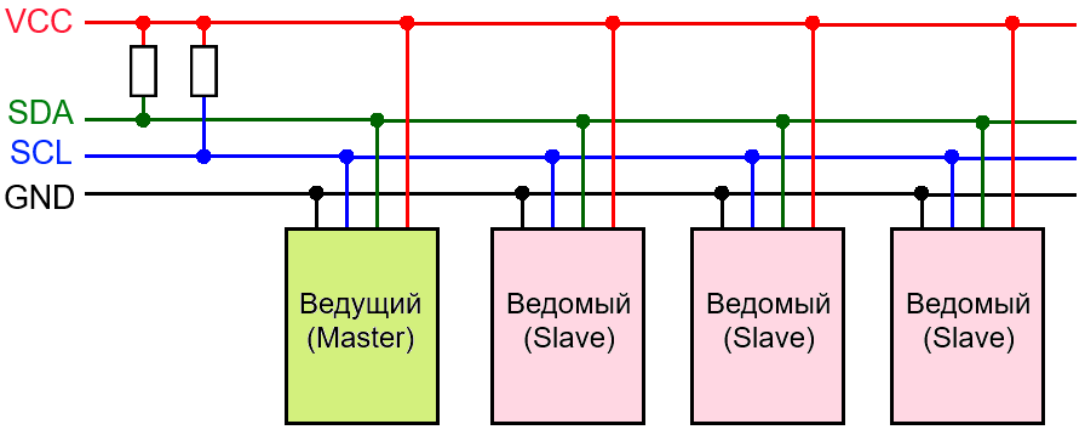

Рисунок 12. – Схема подключения устройств по I2C

4. Разработка системы на программном уровне

4.1 Разработка архитектуры программного уровня системы

Разработка архитектуры программного уровня – одна из важнейших частей при разработке какого-либо робототехнического устройства. Программная часть реализует правильное поведение системы в различных эксплуатационных ситуациях.

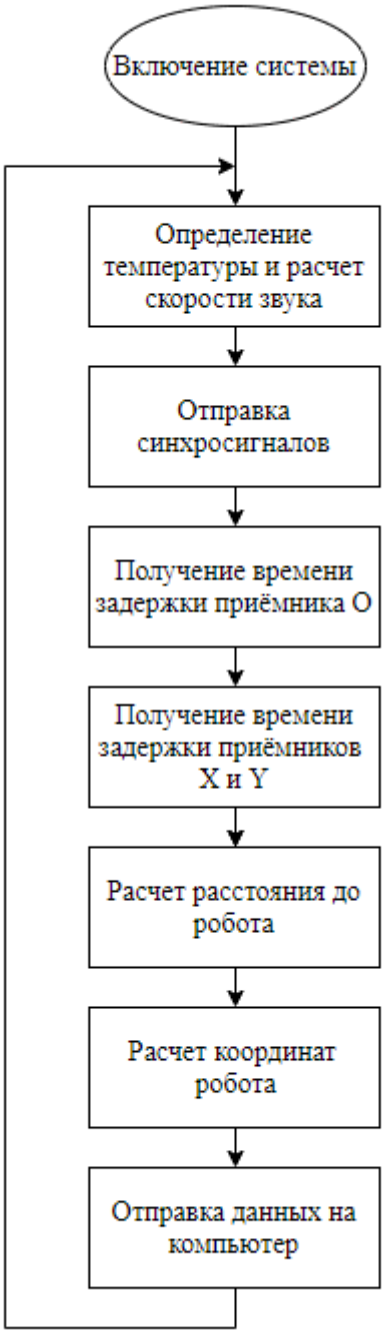

Рисунок 13. – Блок-схема системы

4.2 Реализация метода трилатерации

В данной работе, для позиционирования робота телеприсутсвия в помещении, используется метод трилатерации. С геометрической точки зрения задача трилатерации сводится к нахождению точки пересечения трех или четырех сфер, координаты центра которых известны (ими являются Приёмники O, X, Y), а радиусом которых является расстояние от центра каждой из сфер до робота. Если известно расстояние от робота до одного приёмника, то эта информация позволяет говорить, что робот находится где-то на поверхности этой сферы. Если известны расстояния до двух приёмников, то это сужает область местонахождения робота до пространства, образуемого в месте пересечения двух сфер. Расстояния до трех приемников позволяет узнать две точки, в одной из которых находится робот — это точки пересечения трех сфер. Таким образом, применение системы из 3-х приемников уместно только в том случае, когда робот находится с ними в одной плоскости. В ситуациях, когда предполагается полёт робота — требуется применение системы из 4-х приемников.

Уравнения для трёх сфер:

$$r_1^2 = x^2 + y^2 + z^2; \quad r_2^2 = (x - i)^2 + y^2 + z^2; \quad r_3^2 = x^2 + (y - i)^2 + z^2$$

Где (см. Рисунок 14):

r1, r2, r3 - расстояния от объекта (робота) до приёмников O, X, Y;

x, y, z - координаты робота;

i - смещение приёмника X относительно приёмника O по оси x;

j - смещение приёмника Y относительно приёмника O по оси y.

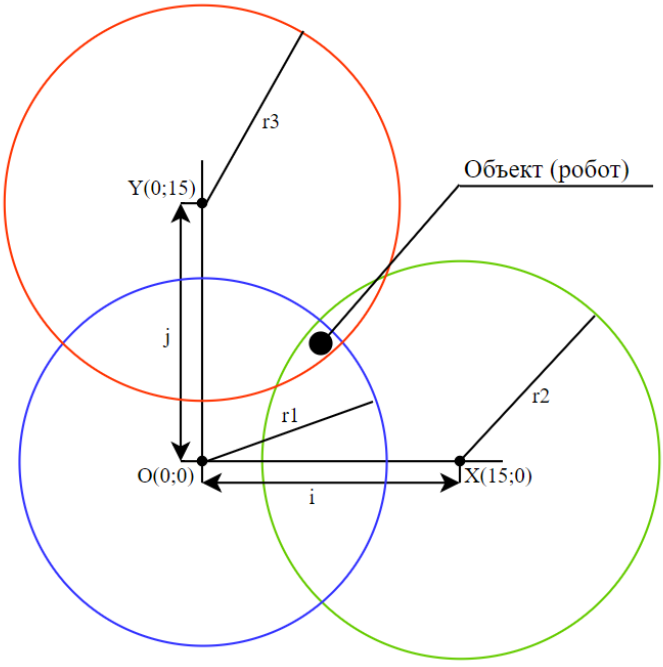

Рисунок 14. – Метод трилатерации

Имеем три неизвестных, выразим их одно через другое:

$$x = \frac{r_1^2 - r_2^2 + i^2}{2i}; \qquad y = \frac{r_1^2 - r_3^2 + j^2}{2j}; \qquad z = \pm\sqrt{r_1^2 - x^2 - y^2}$$

В результате выполнения вычислений становятся известны три координаты, описывающие текущее положение робота. Значение координаты z выражается как корень из числа и может иметь 0, 1 или 2 решения.

4.3 Разработка и реализация серверной части

Программная часть реализации серверной стороны состоит из четырех основных модулей.

1. server_x – этот модуль, написанный для Arduino Uno, используется для получения ультразвукового сигнала от робота, вычисления времени прохождения сигнала и передачи этого значения на устройства расположенного в начале координат.

В функции loop ждем прихода синхронизирующего сигнала, который сообщает, что робот начинает генерировать ультразвуковой сигнал. Как только синхронизирующий сигнал пришел, засекаем время получения (количество микросекунд с момента запуска контроллера) и ждем до тех поп, пока не придет ультразвуковой сигнал, и снова засекаем время. Зная время отправки и получения сигнала, вычисляем время сигнала в пути и передаем его на устройство в начале координат.

```
void loop() {
  if(digitalRead(SYNCHRONIZATION_X) == HIGH){
    time_start = micros();
        while(!digitalRead(RECEIVER));
        time_finish = micros();
        time_send = time_finish - time_start;

    data_x.period_x = time_send / 1000000.0;

    i2c_x.sendData(MASTER);
    }
}
```

2. server_y – данный модуль имеет тот же функционал, что и Server_X, но написанный для устройства, находящегося на оси Y.

В функции loop происходит все тоже самое, что и в одноименной функции модуля server_x.

```
void loop() {
  if(digitalRead(SYNCHRONIZATION_Y) == HIGH){
    time_start = micros();
    while(!digitalRead(RECEIVER));
    time_finish = micros();
    time_send = time_finish - time_start;
```

```
    data_y.period_y = time_send / 1000000.0;
    i2c_y.sendData(MASTER);
    }
}
```

3. server_O – данный модуль является основным в серверной части, он также принимает ультразвуковой сигнал от робота и вычисляет время прохождения сигнала и получается данные о времени прохождения сигнал с двух соседних устройств. На основе полученных данных, вычисляет местоположение робота, по методу трилатерации, заранее вычислив температуру окружающей среды и скорость звука.

Основные функции этого модуля:

Функция loop:

```
void loop()  {
  temp();
  synchronization();
  getRadiusO();
  getRadiusX();
  getRadiusY();
  trilateration();
}
```

В функции loop первым делом замеряется температура окружающей среды и вычисляется скорость звука:

Скорость звука в газе рассчитывается по следующей формуле:

$$c = \sqrt{x * R * T}$$

Где:

x - показатель адиабаты;

R - газовая постоянная (Дж/кг·К);

T - температура газа (К).

```
void temp() {
  tempC = ((analogRead(THERMOMETER) * 5.0) / 1024.0) * 100;
  speed = sqrt(1.4 * 287 * (tempC + 273.15));
}
```

Далее отправляется синхросигнал роботу, после получения которого он начинает генерацию ультразвуковой волны, а также синхросигналы устройствам на оси x и y:

```
void synchronization() {
  Serial.println("S");
  delayMicroseconds(DELAY);
  digitalWrite(SYNCHRONIZATION_X, HIGH);
  digitalWrite(SYNCHRONIZATION_Y, HIGH);
  delayMicroseconds(10);
  digitalWrite(SYNCHRONIZATION_X, LOW);
```

```
  digitalWrite(SYNCHRONIZATION_Y, LOW);
}
```

После того, как робот сгенерировал ультразвуковую волну, захватываем этот сигнал и замеряем время, за которой он дошел от робота до каждого из датчиков и вычисляем расстояния:

```
void getRadiusO() {
  time_start = micros();
  while(!digitalRead(RECEIVER));
  time_finish = micros();
  time_send = time_finish - time_start;
  radius_1 = (time_send / 1000000.0) * speed;
}

void getRadiusX() {
  if(i2c_x.receiveData()) {
    radius_2 = data_x.period_x * speed;
   }
}

void getRadiusY() {
  if(i2c_y.receiveData()) {
    radius_3 = data_y.period_y * speed;
   }
}
```

Когда расстояния от робота до каждого из датчиков найдены, используем метод трилатерации для вычисления координат робота, после чего передаем вычисленные координаты, на устройство которые связывает разрабатываемую систему и компьютер:

```
void trilateration() {
  x = (pow(radius_1, 2) - pow(radius_2, 2) + pow(offset, 2)) / (2 * offset);
  y = (pow(radius_1, 2) - pow(radius_3, 2) + pow(offset, 2)) / (2 * offset);
  z = sqrt(pow(radius_1, 2) - pow(x, 2) - pow(y, 2) );

  data_conn.x_coord = x;
  data_conn.y_coord = y;
  data_conn.z_coord = z;
  i2c_conn.sendData(CONN);
}
```

4. server_conn – модуль, отвечающий за взаимодействие между системой позиционирование и компьютером. Принимает данный от системы по интерфейсу I2C и передает их на компьютер по последовательному порту.

В методе loop проверяем, пришли ли какие-нибудь данные. Если пришли, то по последовательному порту передаем координаты робота, извлеченные из полученной структуры данных.

```
void loop()  {
 if(i2c_conn.receiveData()) {
  Serial.print("x: ");
  Serial.println(data_conn.x_coord, 2);
  Serial.print("y: ");
  Serial.println(data_conn.y_coord, 2);
  Serial.print("z: ");
  Serial.println(data_conn.z_coord, 2);
 }
}
```

4.4 Разработка и реализация клиентской части

В клиентской части был разработан один модуль – Client, который отвечает за приём синхронизирующего сигнала от серверной части и генерацию ультразвукового сигнала.

В функции loop проверяем, пришел ли синхронизирующий сигнал. После получения сигнала генерируется ультразвуковой сигнал частотой 40 кГц в течении 10 секунд.

```
void loop() {
 if(Serial.available()) {
  int syncMes = Serial.read();
  if (syncMes == 'S') {
   tone(TRANSMITTER, 40000);
   delay(10);
   noTone(TRANSMITTER);
   delay(1000);
  }
 }
}
```

5. Тестирование системы

В заключение разработки любой программно-аппаратной системы происходит этап тестирования, на котором подтверждается или правильность её работы, или же обнаруживается наличие ошибок разработки и проектирования. А также может быть проведена оценка эффективности работы системы

Для проведения тестирования, в рабочей области системы, были выбраны три контрольные точки, координаты которых были измерены, и в которые был помещен робот для вычисления его координат.

Рисунок 15. – Тестирование системы в точке (1.15; 0.65)

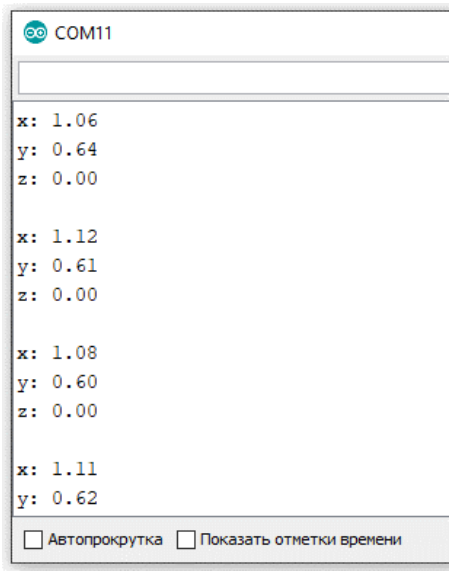

Рисунок 16. – Результат тестирования системы в точке (1.15; 0.65)

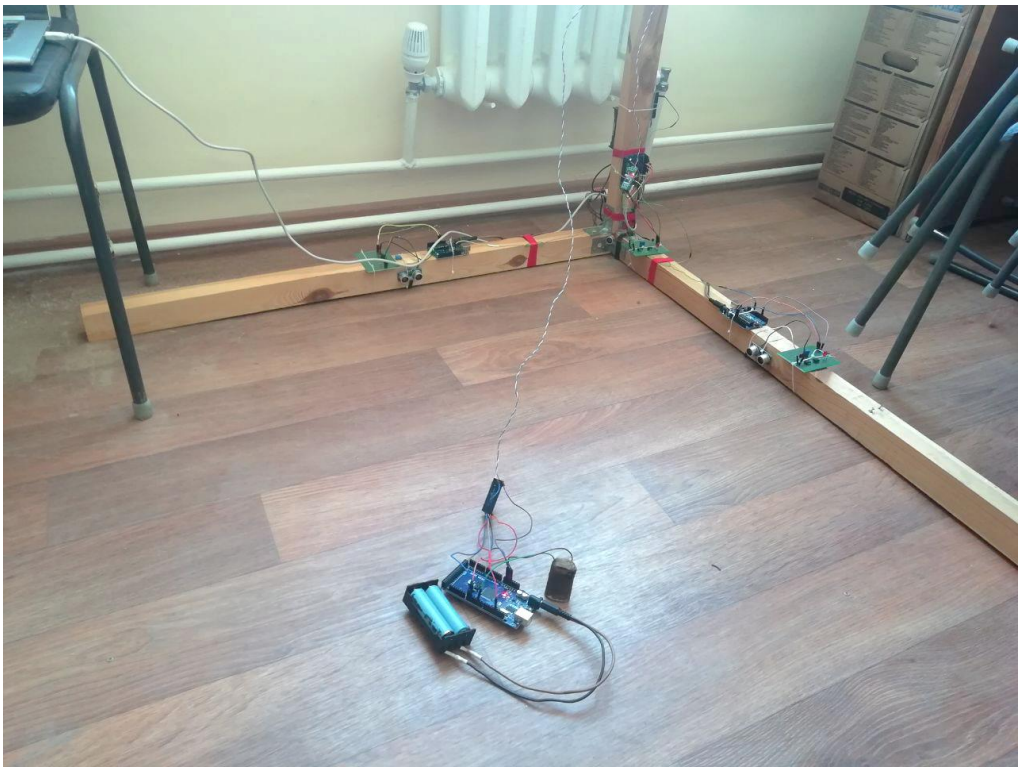

Рисунок 17. – Тестирование системы в точке (0.6; 0.8)

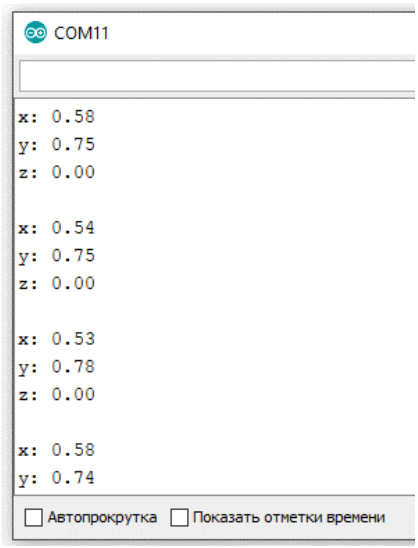

Рисунок 18. – Результат тестирования системы в точке (0.6; 0.8)

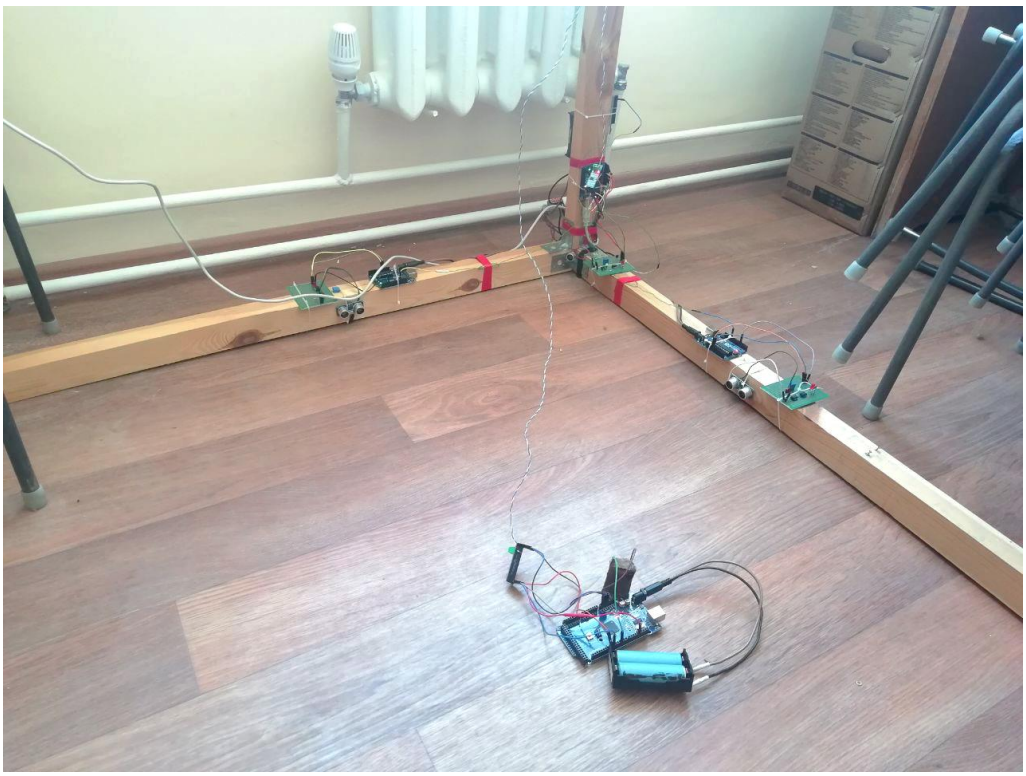

Рисунок 19. – Тестирование системы в точке (0.5; 0.75)

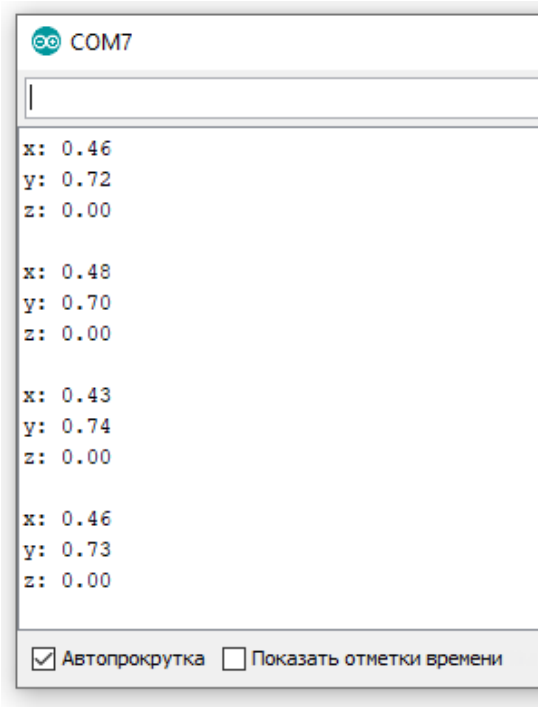

Рисунок 20. – Результат тестирования системы в точке (0.5; 0.75)

Как видно из проведенных экспериментов, вычисленные координаты робота в помещении близки с реальными, за исключением некоторой погрешности.

Заключение

В данной выпускной квалификационной работе была спроектирована и реализована программно-аппаратная система позиционирования робота в помещении. Вычисление координат местоположения робота были выполнены по методу трилатерации.

Разработанная система предназначена для определения местоположения робота в помещении. Тестирование программы подтвердило ее работоспособность и возможность использования для решения данной задачи.

Список литературы

1. Ахметов И.Г., Технические науки: проблемы и перспективы: материалы IV Междунар. науч. Конф. (г. Санкт-Петербург, июль 2016 г.). — СПб.: Свое издательство, 2016. — vi, 134 с. Режим доступа: https://moluch.ru/conf/tech/archive/166/10779/

2. Карпов В. Э., Платонова М. В. Система навигации мобильного робота // Московский Энергетический Институт; Москва, лаборатория робототехники и искусственного интеллекта Политехнического музея – 2008

3. *Brinker, R.C. and Minnick, R.* 12. Trilateration // The Surveying Handbook. — Chapman & Hall, 1995

4. Программатор-отладчик и оценочная плата [Электронный ресурс]. Режим доступа: http://zadocs.ru/pravo/51148/index.html

Приложения

server_x.ino

```
#include <Wire.h>
#include <EasyTransferI2C.h>

#define RECEIVER 5

#define ADDRESS 9
#define MASTER 8
#define SYNCHRONIZATION_X 7

long time_start = 0, time_finish = 0, time_send = 0;

EasyTransferI2C i2c_x;
struct STRUCTURE_X {
  float period_x;
};
STRUCTURE_X data_x;

void setup() {
  Wire.begin(ADDRESS);

  i2c_x.begin(details(data_x), &Wire);

  pinMode(RECEIVER, INPUT);
  pinMode(SYNCHRONIZATION_X, INPUT);
}

void loop() {
  if(digitalRead(SYNCHRONIZATION_X) == HIGH){
    time_start = micros();
        while(!digitalRead(RECEIVER));
        time_finish = micros();
        time_send = time_finish - time_start;

    data_x.period_x = time_send / 1000000.0;

    i2c_x.sendData(MASTER);
    }
}
```

server_y.ino

```
#include <Wire.h>
#include <EasyTransferI2C.h>

#define RECEIVER 5

#define ADDRESS 10
#define MASTER 8
#define SYNCHRONIZATION_Y 8

long time_start = 0, time_finish = 0, time_send = 0;

EasyTransferI2C i2c_y;
struct STRUCTURE_Y {
  float period_y;
};
STRUCTURE_Y data_y;

void setup() {
  Wire.begin(ADDRESS);

  i2c_y.begin(details(data_y), &Wire);

  pinMode(RECEIVER, INPUT);
  pinMode(SYNCHRONIZATION_Y, INPUT);
}

void loop() {
  if(digitalRead(SYNCHRONIZATION_Y) == HIGH){
    time_start = micros();
    while(!digitalRead(RECEIVER));
    time_finish = micros();
    time_send = time_finish - time_start;

    data_y.period_y = time_send / 1000000.0;

    i2c_y.sendData(MASTER);
    }
}
```

server_o.ino

```
#include <Wire.h>
#include <EasyTransferI2C.h>

#define RECEIVER 5

#define THERMOMETER A1

#define SYNCHRONIZATION_X 7
#define SYNCHRONIZATION_Y 8

#define SERIAL_TX_CONTROL 13
#define RS485_TRANSMIT HIGH
#define RS485_RECEIVE LOW

#define DELAY 112

#define ADDRESS 8
#define CONN 11

EasyTransferI2C i2c_x;
struct STRUCTURE_X {
  float period_x;
};
STRUCTURE_X data_x;

EasyTransferI2C i2c_y;
struct STRUCTURE_Y {
  float period_y;
};
STRUCTURE_Y data_y;

EasyTransferI2C i2c_conn;
struct SEND_DATA_STRUCTURE_CONN {
  float x_coord, y_coord, z_coord;
};
SEND_DATA_STRUCTURE_CONN data_conn;

float speed, tempC, offset = 0.5;
long time_start = 0, time_finish = 0, time_send = 0;
float radius_1, radius_2, radius_3;
float x, y, z;
```

```
void setup()  {
  Serial.begin(115200);

  Wire.begin(ADDRESS);
  Wire.onReceive(receive);

  i2c_x.begin(details(data_x), &Wire);
  i2c_y.begin(details(data_y), &Wire);
  i2c_conn.begin(details(data_conn), &Wire);
  pinMode(SYNCHRONIZATION_X, OUTPUT);
  pinMode(SYNCHRONIZATION_Y, OUTPUT);

  pinMode(SERIAL_TX_CONTROL, OUTPUT);
  digitalWrite(SERIAL_TX_CONTROL, RS485_RECEIVE);

  pinMode(RECEIVER, INPUT);
}

void loop()  {
  temp();

  synchronization();

  getRadiusO();
  getRadiusX();
  getRadiusY();

  trilateration();
}

void temp() {
  tempC = ((analogRead(THERMOMETER) * 5.0) / 1024.0) * 100;
  speed = sqrt(1.4 * 287 * (tempC + 273.15));
}

void synchronization() {
  Serial.println("S");

  delayMicroseconds(DELAY);

  digitalWrite(SYNCHRONIZATION_X, HIGH);
  digitalWrite(SYNCHRONIZATION_Y, HIGH);
  delayMicroseconds(10);
```

```
  digitalWrite(SYNCHRONIZATION_X, LOW);
  digitalWrite(SYNCHRONIZATION_Y, LOW);
}

void getRadiusO() {
  time_start = micros();
  while(!digitalRead(RECEIVER));
  time_finish = micros();
  time_send = time_finish - time_start;

  radius_1 = (time_send / 1000000.0) * speed;
}

void getRadiusX() {
  if(i2c_x.receiveData()) {
    radius_2 = data_x.period_x * speed;
   }
}

void getRadiusY() {
  if(i2c_y.receiveData()) {
    radius_3 = data_y.period_y * speed;
   }
}

void trilateration() {
  x = (pow(radius_1, 2) - pow(radius_2, 2) + pow(offset, 2)) / (2 * offset);
  y = (pow(radius_1, 2) - pow(radius_3, 2) + pow(offset, 2)) / (2 * offset);
  z = sqrt(pow(radius_1, 2) - pow(x, 2) - pow(y, 2) );

  data_conn.x_coord = x;
  data_conn.y_coord = y;
  data_conn.z_coord = z;
  i2c_conn.sendData(CONN);
}

void receive(int numBytes) {}
```

server_conn.ino

```
#include <Wire.h>
#include <EasyTransferI2C.h>

#define ADDRESS 11

EasyTransferI2C i2c_conn;
struct STRUCTURE_CONN {
  float x_coord, y_coord, z_coord;
};
STRUCTURE_CONN data_conn;

void setup() {
  Serial.begin(9600);

  Wire.begin(ADDRESS);
  Wire.onReceive(receive);

  i2c_conn.begin(details(data_conn), &Wire);
}

void loop()  {
  if(i2c_conn.receiveData()) {
    Serial.print("x: ");
    Serial.println(data_conn.x_coord, 2);
    Serial.print("y: ");
    Serial.println(data_conn.y_coord, 2);
    Serial.print("z: ");
    Serial.println(data_conn.z_coord, 2);
  }
}

void receive(int numBytes) {}
```

client.ino

```
#define TRANSMITTER 5

#define SERIAL_TX_CONTROL 13
#define RS485_TRANSMIT HIGH
#define RS485_RECEIVE LOW

void setup() {
```

```
  Serial.begin(115200);

  pinMode(SERIAL_TX_CONTROL, OUTPUT);
  digitalWrite(SERIAL_TX_CONTROL, RS485_TRANSMIT);
  pinMode(TRANSMITTER, OUTPUT);
}

void loop() {
  if(Serial.available()) {
    int syncMes = Serial.read();

    if (syncMes == 'S') {
      tone(TRANSMITTER, 40000);
      delay(10);
      noTone(TRANSMITTER);
      delay(1000);
    }
  }
}
```