# Глава 3. Практика

## 3.1 Мобильный автономный робот

Для решения задачи данной работы необходимо было проводить «живые» тестирования работы алгоритмов. Такая необходимость обусловлена прежде всего тем, что помимо существующей задачи данной ВКР стояла задача в создании робота для распознавания объектов. По этой причине было принято решение делать алгоритмы на реальном роботе[1] с пребыванием данного робота во вполне реальных условиях.

### 3.1.1 Подбор шасси

Как было сказано в первой главе данной работы, существует большое количество различных шасси, на которых можно располагать различное оборудование. Наш выбор остановился на гусеничном шасси, которое изображено на Рисунке .

Данное шасси за счёт своих размеров является очень мобильным средством передвижения робота и может проникнуть в относительно узкие для роботов пространства и без проблем оттуда выбраться, не повредившись. Для оборудования на данном шасси место тоже нашлось: для этого было принято решение заказать металлическую пластину, которая играла роль второго этажа. поставить рисунок?

[1]Задание данной ВКР можно было бы сделать и в любом симуляторе или игровом движке. Однако решение делать всё в реальной жизни сильно усложнило данную задачу.

### 3.1.2 Движение шасси

К сожалению, по неизвестной причине к данному шасси не пришёл комплектный контроллер движения, который бы принимал команды от компьютера и заставлял двигаться установленные гусеницы. По этой причине пришлось немного изучить ещё одну предметную область, которая не изучалась в течении университетского курса - электротехнику.

#### Требования к контроллеру движений

Компьютер, который будет в последствии установлен на робота будет управлять роботом посредством бинарных сигналов с напряжением 3.3В через порт GPIO, где 0 (или по-другому нет напряжения) - это движение не требуется и 1 (когда есть напряжение +3.3В), когда движение требуется.

Контроллер должен, также, уметь по отдельности управлять двумя гусеницами, заставлять их ездить вперёд и назад. Это основные требования. Из дополнительных требований можно выделить умение каким-то образом регулировать скорость движения. Общая структура желаемой модели контроллера изображена на Рисунке .

#### Схема будущего контроллера

Здесь теоретическое объяснение как работает контроллер

#### Производство контроллера

В качестве основы для контроллера было принято решение взять текстолитовую пластину с медным покрытием, изображённую на рисунке .

Далее на нём при помощи перманентного маркера были нанесены дорожки, а места вставки деталей были просверлены советской стоматологической бур-машинкой. Далее всё это прошло ванну хлорида железа. Процесс и результаты работы изображены на Рисунке .

## Результаты работы

Таким образом был получен полноценный контроллер, который умеет управлять роботом медленной и быстрой скоростями. Однако, не обошлось без недостатков и трудностей. Главной трудностью стал неудачный способ стравливания дорожек на плате, поэтому пришлось дополнительно при помощи олова проводить дорожки и искать потенциально уязвимые места.

После большой ручной работы плату удалось запустить, но появилась другая проблема. И эта проблема заключалась в том, что сопротивление двигателей робота немного отличалось друг от друга, также как и сопротивление резисторов, установленных на плату. Данное отличие было ничтожным, но этого вполне хватало, чтобы робот начинал ездить не очень ровно и его всё время приходилось дополнительно корректировать.

## Вторая версия контроллера

Необходимость во второй версии контроллера возникла в первую очередь из-за того, что медленная скорость на роботе работала слишком медленно, а быстрая слишком быстро. Ездить в магазин и вручную подбирать резисторы, каждый раз перепаивая плату совсем не хотелось, а каждый резистор в местных магазинах обходится в неприятную для таких расходных деталей сумму.

Переделка первой версии платы обошлась бы слишком дорого в плане времени, да и к тому же не хотелось портить, то, что и так уже работает. Поэтому вторая версия платы была собрана по той же схеме, что и первая, но с несколькими важными отличиями. Из стабилизаторов остался только ста-

билизатор на 9В, а обычные резисторы были заменены на соответствующие подстроечные резисторы. Таким образом должна была появилась возможность регулировать движение в зависимости от того, в каком положении будет установлен сам резистор. Получившаяся плата изображена на Рисунке .

## 3.2 Визуальный анализ пространства

Для анализа окружающего пространства существует довольно большое количество различных датчиков и прочего оборудования. Но закупать сразу всё не выгодно экономически, затратно в плане места размещения на роботе и расточительно в плане потребления электроэнергии этими самыми датчиками. Также для обработки всех этих сигналов нужны соответствующие вычислительные мощности.

Таким образом робот должен иметь совсем небольшое количество сенсоров и при этом не быть «слепым». Исходя из этих соображений, было решено установить на робота два основных сенсора: лазерный сканер YDLIDAR X4 и CSI камеру Sony IMX217. Первый поможет видеть препятствия вокруг робота, второй поможет видеть объекты, размещённые перед роботом.

Схема размещения сенсоров размещена на Рисунке .

## 3.3 Формирование поведенческой стратегии робота

Основная задача робота - ездить и искать целевые объекты делится на две подзадачи.

### 3.3.1 Езда

Этот режим поездки можно также назвать режимом исследователя. В идеальном случае, робот в начале работы алгоритма движения должен объездить всё доступное ему пространство, составить карту окружающей местности, параллельно при этом запоминая увиденные им объекты и их примерное местоположение[2], а затем после полного сканирования объехать всю территорию и убедиться в том, что целевые объекты действительно были найдены на этих местах. И это должно стать конечной целью робота. Однако, для упрощения данной задачи будет использоваться другой куда более простой алгоритм.

Режим исследователя будет подразумевать под собой то, что робот будет просто ехать вперёд, параллельно разыскивая целевые объекты и объезжая препятствия.

#### Объезд препятствий

Робот должен уметь объезжать хотя-бы самые простейшие препятствия, по типу стен, диванов или прочих перегородок. В идеале, он должен уметь справляться и с тонкими препятствиями по типу ножек стула и мягкие поверхности.

Алгоритм объезда препятствий, представленный на данном роботе сводится к схеме, изображённой на Рисунке .

Подсчёт того, где свободнее: слева или справа идёт из соображений того, где находится больше препятствий. Как это считается? Условно робот поделён на несколько направлений. В данном случае он подразделён на «перед», «лево», «право» и «зад». Обозначим значения, которые высчитываются на этих направлениях, как $f, l, r$ и $b$ соответственно.

Лидар выдаёт данные в формате массива значений float: обозначим его как числовой ряд $a$. В этом числовом ряду находятся числа, обознача-

[2] его нужно вычислять из расчётов того, куда смотрит в данный момент робот, угла прямоугольника на изображении, на котором объект был обнаружен

ющие расстояние до точки об которое отразился лазер. Чем больше число, тем дальше находится объект об который отразился лазер. Если лазерный сканер не нашёл в этом месте отражения, то он возвращает значение -1. Фактически, данный массив является аналогом полярных координат, где позиция значения в массиве - это угол, а само значение является расстоянием. Всего этих чисел 720, из чего можно сделать вывод, что цена деления лидара это полградуса.

Таким образом каждый поворот лидара вычисляются 4 переменные[3]:

$$l = \frac{\sum_{i=90}^{270} a_i}{180} \quad b = \frac{\sum_{i=270}^{450} a_i}{180} \quad r = \frac{\sum_{i=450}^{630} a_i}{180} \quad f = \frac{\sum_{i=630}^{720} a_i + \sum_{i=0}^{90} a_i}{180}$$

После вычисления этих средних арифметических значений по каждой из сторон проверяются значения массива $a_i$, где $630 < i < 720$ и $0 < i < 90$ (передняя сторона робота) и если среди этих чисел находится хоть одно удовлетворяющее условию $0 < a_i < 0{,}3$, то считается в данный момент перед роботом находится какое-то препятствие.

Если это так, то далее сравниваются значения ранее высчитанных переменных $l$ и $r$. Если значение $l > r$, то робот поедет налево, так как слева нашлось меньше препятствий, чем справа. Иначе, роботу следует ехать направо. Однако представленного выше алгоритма ещё недостаточно чтобы объезжать часто встречаемые препятствия.

### Обнаружение застревания

Робот может попасть в ситуацию, когда впереди внезапно образовалась преграда, невидимая для лазерного сканера (например, очень низкая преграда). Для обнаружения застревания при столкновении с такими преградами необходимо как-то понять, что робот перестал двигаться.

[3]Важное замечание: значения -1, когда лазерный сканер не нашёл отражения заменяются на значение 1 для того чтобы значения переменных прибавлялись, а не уменьшались.

Одним из способов понять и распознать застревание может стать анализ облака точек, которые выдаёт LIDAR. Если вектор движения большинства точек на плоскости облака стал достаточно мал, то можно сделать вывод о том, что робот либо плохо двигается, либо вообще застрял.

В данный проект была встроена система Google Cartographer, которая по облаку точек может строить окружающую карту местности, а также определять местоположение робота на ней. Информацию о местоположении можно использовать как раз в целях определения застревания. Если в течении секунды координаты робота менялись недостаточно сильно, то значит робот застрял.

Для выезда из застревания используется простой алгоритм, который состоит из 3 шагов:

1. Ехать назад 2 секунды;
2. Выбрать сторону, в которую будет совершён поворот по значениям выше упомянутых переменных $l$ и $r$;
3. Ехать дальше.

Как показала практика, этот алгоритм работает достаточно эффективно для того чтобы не застревать в большинстве ситуаций.

### 3.3.2 Поиск целевого объекта

Данный режим предполагается включать только в случае, если на видеосигнале, получаемом от CSI камеры был распознан целевой объект. В этом случае робот останавливается и поворачивается в ту сторону, где расположен центр предполагаемого целевого объекта. Далее робот начинает ехать вперёд и по мере необходимости продолжает центрировать шасси до тех пор пока не подъедет к объекту. Далее робот останавливается, конечная цель робота выполнена: целевой объект найден.

Определение того, что робот подъехал к объекту происходит по размеру прямоугольника, на котором обозначен целевой объект. Если прямоугольник уже достиг краёв кадра видеосигнала, значит робот приблизился к объекту максимально близко. Подъезд вплотную к объекту является

не самой лучшей идеей, так как целевым объектом может быть стеклянная бутылка, которую можно просто сбить и разбить.

## 3.4 Подробнее о программной части робота

На одноплатный компьютер Nvidia Jetson Nano была установлена операционная система Ubuntu LTS 18.04 со специальным от Nvidia программным обеспечением JetPack 4.3, которое предоставляет удобные инструменты для вычислений в области искусственного интеллекта при помощи встроенного в Jetson NANO видеочипа и ядер CUDA. Также на компьютер была установлен фреймворк для программирования роботов ROS, аббревиатура которого расшифровывается как «Операционная система для роботов».

### 3.4.1 ROS

ROS предоставляет удобные и мощные функции, помогающие разработчикам в таких задачах, как передача сообщений различного типа, распределение вычислений между компьютерами, повторное использование кода и реализация современных алгоритмов для роботизированных приложений. В общем случае, ROS представляет собой инструмент, позволяющий связывать несколько независимых программных модулей при помощи сервисов и узлов, которые могут передавать друг другу сообщения в различном формате.

Большими преимуществами использования данного фреймворка является возможность передачи сообщений по локальной сети и обширная библиотека уже реализованного ПО, которое можно без относительно больших затрат по времени интегрировать в свой собственный проект. На момент написания данной ВКР глобальный репозиторий ROS Index насчитывает 2120 подключенных к нему сторонних репозиторием и 5827 пакетов.

### 3.4.2 Концепции ROS

Ниже приведён список концепций рассматриваемого фреймворка:

– **Узел** - это процесс, выполняющий вычисления. Каждый узел написание с использованием клиентских библиотек ROS. Используя методы связи, узлы могут общаться друг с другом и обмениваться данными. Для этого создаются узлы-подписчики, и узлы-публикаторы.

– **Мастер** - обеспечивает регистрацию и работоспособность запущенных узлов.

– **Сообщение** - простая структура данных, содержащая типизированное поле, которое может содержать целый набор данных, отправляемых на другой узел. Помимо стандартных типов сообщений[4] возможна отправка заранее обозначенных собственных типов сообщений.

– **Тема** - именованная шина данных, используемая узлами для отправки сообщений. Публикующий и подписанный узел не знают о существовании друга друга. Благодаря тому что каждая тема имеет уникальное имя, любой узел может получить доступ к данной теме и отправляет через неё данные, при условии соблюдении заранее оговорённых передаваемых типов, данной темой.

– **Сервисы** - реализация удалённого вызова процедур[5] в ROS. В некоторых случаях модель связи публикации и подписки может не подходить. В этих случаях и применяют взаимодействия в виде сервисов (схема запрос/ответ), при котором один узел может запросить выполнение процедуры для другого узла, ожидая какого-то обязательного ответа[6].

---

[4] Такие как целые, с плавающей точкой, логические, строковые...

[5] RPC

[6] В случае использования схемы с подписчиками и публикаторами доставка сообщений и ответ не гарантируются

### 3.4.3 Реализуемые узлы

В рамках работы над данной ВКР были реализованы следующие узлы и сервисы:

- Сервис, управляющий сигналами на разъёме GPIO;
- Узел записи видео с видеокамеры;
- Узел распознавания объектов;
- Узел, управляющий движением робота и формирующий поведенческую стратегию робота.

Также в роботе используются следующие сторонние узлы:

- Узел передачи изображения с CSI видеокамеры;
- Google Cartographer;
- Узел YDLIDAR.

#### Узел видеокамеры

Написать

#### Google Cartographer

Написать

#### Узел YDLIDAR

Написать

### Сервис GPIO

Написать

### Узел записи видео

Написать

### Узел распознавания объектов

Написать

### Узел движения

Написать

## 3.5 Таблица обыкновенная

Так размещается таблица:

Таблица 7 — Название таблицы

| Месяц | $T_{min}$, К | $T_{max}$, К | $(T_{max} - T_{min})$, К |
|---|---|---|---|
| Декабрь | 253.575 | 257.778 | 4.203 |
| Январь | 262.431 | 263.214 | 0.783 |
| Февраль | 261.184 | 260.381 | −0.803 |

Таблица 8

| Оконная функция | $2N$ | $4N$ | $8N$ |
|---|---|---|---|
| Прямоугольное | 8.72 | 8.77 | 8.77 |
| Ханна | 7.96 | 7.93 | 7.93 |
| Хэмминга | 8.72 | 8.77 | 8.77 |
| Блэкмана | 8.72 | 8.77 | 8.77 |

Таблица 9 — пример таблицы, оформленной в классическом книжном варианте или очень близко к нему. ГОСТу по сути не противоречит. Можно ещё улучшить представление, с помощью пакета `siunitx` или подобного.

Таблица 9 — Наименование таблицы, очень длинное наименование таблицы, чтобы посмотреть как оно будет располагаться на нескольких строках и переноситься

| Оконная функция | $2N$ | $4N$ | $8N$ |
|---|---|---|---|
| Прямоугольное | 8.72 | 8.77 | 8.77 |
| Ханна | 7.96 | 7.93 | 7.93 |
| Хэмминга | 8.72 | 8.77 | 8.77 |
| Блэкмана | 8.72 | 8.77 | 8.77 |

### 3.6 Таблица с многострочными ячейками и примечанием

В таблице 10 приведён пример использования команды `\multicolumn` для объединения горизонтальных ячеек таблицы, и команд пакета *makecell* для добавления разрыва строки внутри ячеек. При форматировании таблицы 10 использован стиль подписей `split`. Глобально этот стиль может быть включён в файле `Dissertation/setup.tex` для диссертации и в файле `Synopsis/setup.tex` для автореферата. Однако такое оформление не соответствует ГОСТ.

Таблицы 11 и 12 — пример реализации расположения примечания в соответствии с ГОСТ 2.105. Каждый вариант со своими достоин-

Таблица 10

Пример использования функций пакета *makecell*

| Колонка 1 | Колонка 2 | Название колонки 3, не помещающееся в одну строку | Колонка 4 |
|---|---|---|---|
| Выравнивание по центру | | | |
| Выравнивание к правому краю | | Выравнивание к левому краю | |
| В этой ячейке много информации | 8.72 | 8.55 | 8.44 |
| А в этой мало | 8.22 | 5 | |

ствами и недостатками. Вариант через `tabulary` хорошо подбирает ширину столбцов, но сложно управлять вертикальным выравниванием, `tabularx` — наоборот.

Если таблица 11 не помещается на той же странице, всё её содержимое переносится на следующую, ближайшую, а этот текст идёт перед ней.

## 3.7 Таблицы с форматированными числами

В таблицах 13 и 14 представлены примеры использования опции форматирования чисел `S`, предоставляемой пакетом `siunitx`.

## 3.8 Параграф — два

Некоторый текст.

## 3.9 Параграф с подпараграфами

### 3.9.1 Подпараграф — один

Некоторый текст.

### 3.9.2 Подпараграф — два

Некоторый текст.

Таблица 11 — Нэ про натюм фюйзчыт квюальизквюэ

| доминг лаборамюз эи ыам (Общий съём цен шляп (юфть)) | Шеф взъярён | адвыр-жаряюм | тебик-вюэ эльъэеф-энд мэдио-крета-тым | Чэнзэ-рет мны-жарк-хюм |
|---|---|---|---|---|
| Эй, жлоб! Где туз? Прячь юных съёмщиц в шкаф Плюш изъят. Бьём чуждый цен хвощ! | ≈ | ≈ | ≈ | + |
| Эх, чужак! Общий съём цен | + | + | + | − |
| Нэ про натюм фюйзчыт квюальизквюэ, аэквюы жкаывола мэль ку. Ад граэкйж плььатонэм адвыржаряюм квуй, вим empыдит коммюны ат, ат шэа одео | ≈ | − | − | − |
| Любя, съешь щипцы, — вздохнёт мэр, — кайф жгуч. | − | + | + | ≈ |
| Нэ про натюм фюйзчыт квюальизквюэ, аэквюы жкаывола мэль ку. Ад граэкйж плььатонэм адвыржаряюм квуй, вим empыдит коммюны ат, ат шэа одео квюаырэндум. Вёртюты ажжынтиор эффикеэнди эож нэ. | + | − | ≈ | − |

Примечание — Плюш изъят: «+» — адвыржаряюм квуй, вим empыдит; «−» — empыдит коммюны ат; «≈» — Шеф взъярён тчк щипцы с эхом гудбай Жюль. Эй, жлоб! Где туз? Прячь юных съёмщиц в шкаф. Экс-граф?

Таблица 12 — Любя, съешь щипцы, — вздохнёт мэр, — кайф жгуч

| доминг лаборамюз эи ыам (Общий съём цен шляп (юфть)) | Шеф взъярён | адвыр-жаряюм | тебиквюэ эльъэеф-энд мэдио-крета-тым | Чэнзэрет мны-жарк-хюм |
|---|---|---|---|---|
| Эй, жлоб! Где туз? Прячь юных съёмщиц в шкаф Плюш изъят. Бьём чуждый цен хвощ! | ≈ | ≈ | ≈ | + |
| Эх, чужак! Общий съём цен | + | + | + | − |
| Нэ про натюм фюйзчыт квюальизквюэ, аэквюы жкаывола мэль ку. Ад граэкйж плльатонэм адвыржаряюм квуй, вим емпыдит коммюны ат, ат шэа одео | ≈ | − | − | − |
| Любя, съешь щипцы, — вздохнёт мэр, — кайф жгуч. | − | + | + | ≈ |
| Нэ про натюм фюйзчыт квюальизквюэ, аэквюы жкаывола мэль ку. Ад граэкйж плльатонэм адвыржаряюм квуй, вим емпыдит коммюны ат, ат шэа одео квюаырэндум. Вёртюты ажжынтиор эффикеэнди эож нэ. | + | − | ≈ | − |

Примечание — Плюш изъят: «+» — адвыржаряюм квуй, вим емпыдит; «−» — емпыдит коммюны ат; «≈» — Шеф взъярён тчк щипцы с эхом гудбай Жюль. Эй, жлоб! Где туз? Прячь юных съёмщиц в шкаф. Экс-граф?

Таблица 13 — Выравнивание столбцов

| Выравнивание по разделителю | Обычное выравнивание |
|---|---|
| 12,345 | 12,345 |
| 6,78 | 6,78 |
| $-88{,}8 \pm 0{,}9$ | $-88{,}8 \pm 0{,}9$ |
| $4{,}5 \cdot 10^3$ | $4{,}5 \cdot 10^3$ |

Таблица 14 — Выравнивание с использованием опции `S`

| Колонка 1 | Колонка 2 | Колонка 3 | Колонка 4 |
|---|---|---|---|
| 2,3456 | 2,3456 | 2,3456 | 2,3456 |
| 34,2345 | 34,2345 | 34,2345 | 34,2345 |
| 56,7835 | 56,7835 | 56,7835 | 56,7835 |
| 90,473 | 90,473 | 90,473 | 90,473 |