Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра Систем управления и информатики

Отчет о прохождении производственной практики.

с 08.02.2016 по 06.03.2016

Студент:	Дема Н.Ю.
Группа:	P3335
Место прохождения практики:	Университет ИТМО кафедра СУиИ
Руководитель:	Капитонов А. А.

‹‹Санкт-Петербург 2016››

Содержание

Содержание 2
Описание основных этапов прохождения практики 3
Практическая работа 4
    Описание платформы 4
    Используемые программные средства 5
    Описание работы программы 5
Подведение итогов 6
Список используемых материалов 7

Описание основных этапов прохождения практики

Прохождение производственной практики было организовано на базе кафедры систем управления и информатики Университета ИТМО. Руководителем практики, Капитоновым А.А., был предложен следующий порядок работ:

- В течении первых двух недель предлагалось самостоятельно ознакомиться с основными программными пакетами для взаимодействия с робототехническими комплексами, в том числе теми, которые предоставляет кафедра. За это время я изучил основные концепции и приобрел практические навыки работы с ROS (Robot Operating System – фреймворк для программирования робототехнических систем), Stage и Gazebo – средства симуляции, позволяющие создавать виртуальное пространство для моделирования объектов и закономерностей реального мира специфичных для робототехники.

- Третья неделя была посвящена изучению одной из наиболее важных составляющих ROS – стеку пакетов навигации. В рамках этого этапа мной, сначала посредством симулятора, а затем и на реальной робототехнической платформе были опробованы стандартные решения для таких задач как локализация, картирование, slam и нахождение оптимального пути по заданным координатам. Так же, мной были приобретены навыки работы с такими робототехническими комплексами как "Robotino" и "Kuka YouBot".

- В рамках прохождения практики Александр Александрович предложил ряд проектов на выбор и, соответственно, изучение всех програмных инструментов с уклоном в специфику выбранного проекта. В течении четвертой недели мной, в качестве практического задания, был написан промежуточный узел в цепи управления любой робототехнической платформы с всенаправленной ходовой частью (omni) и лидаром. Суть его в предотвращений столкновений с препятствиями. Детальное описание представлено дальнейших разделах отчета.

В частности, как одно из направлений деятельности, было предложено начать подготовку к международным робототехническим соревнованиям RoboCup@Work с последующим возможным участием. Во время прохождения практики предполагалось ознакомиться с основными типовыми заданиями ставящимися в рамках данных состязаний и возможными методами их решения.

Одним из основных условий для участия в соревнованиях является обеспечение безопасности[1] как для людей, так или иначе взаимодействующих с роботом, так и для площадки, на которой проводятся различные этапы соревнований. Если команда не уделила достаточно внимания вопросу безопасности, то техническая комиссия еще на этапе проверок исключит её из списка участников. Исходя из этого, а так же списка задач соревнования, я, в рамках выбранного направления, начал изучать вопросы локализации и маршрутизации, в частности – стек пакетов навигации под ROS.

Как практическое задание, я решил написать программу, изменяющую вектор скорости робота при ручном управлении, в зависимости от расстояния до препятствий. То есть, если робот движется в направлении какого либо объекта и оператор не сбавляет скорость самостоятельно, то вектор скорости постепенно изменяется и робот плавно объезжает объект. В качестве входного сигнала, на основании которого принимаются решения, использовался лидар. В общем, данный узел можно использовать и как промежуточное звено при автономном управлении роботом. В качестве рабочей платформы использовался Robotino от Festo Didactic. С исходным кодом проекта можно ознакомиться по адресу: https://github.com/Ram2301/Robotino.

[1] С основным документом, регламентирующим правила допуска к соревнованиям, их список и суть, можно ознакомиться на сайте по адресу robocupatwork.org/rules.

Практическая работа

Описание платформы

Компания-разработчик позиционирует Robotino, как робототехническую платформу для обучения и разработки для университетов и колледжей. По сути это автономный мобильный робот (Рис. 1) с всенаправленной ходовой частью (omni wheel). Одной из главных его особенностей является то, что он выполнен модульно, это позволяет отключить от робота любой технический элемент и изучить отдельно.

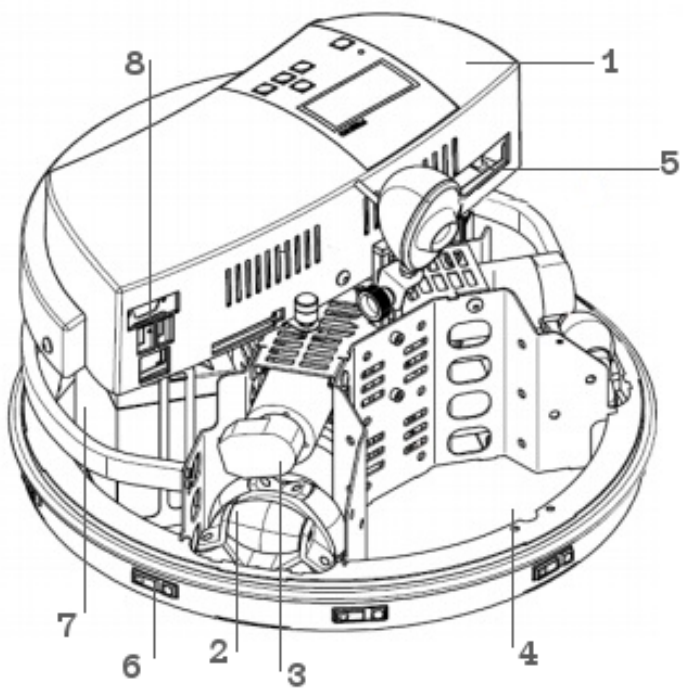

Рис. 1: Robotino.

Основные составные элементы:

1. Корпус. В нем сосредоточена вся основная электронная часть робота[2]. Установлен небольшой монитор на котором отображается состояние робота.
2. Ходовая часть. Три двигателя постоянного тока расположенных по углом 120 градусов. Каждый соединяется ременной передачей с роликовым колесом. Такая схема расположения и специфика колес позволяют роботу отрабатывать достаточно сложные движения на плоскости.
3. Одометрия. Представлена инкрементными энкодерами.
4. Небольшая площадка со множеством различных отверстий под крепления. В моем случае в этом месте был закреплен лидар. Так же может быть подключен например манипулятор или что-то еще.
5. Интерфейс ввода\вывода.
6. Бампер. На нем расположены 9 инфракрасных дальномеров, так же по всему периметру сам бампер может реагировать на нажатие (черная полоска из мягкой резины).

[2]В частности, одноплатный промышленный компьютер формата MicroPC (он, кстати, тоже модульный, что-то вроде шилдов для arduino) с процессором на 300Мгц. В качестве памяти используется флеш-карта на которую устанавливается версия GNU\Linux

7. Элементы питания. Две аккумуляторные батареи (12В/5Ач каждая, кислотно-свинцовые).

8. Интерфейсы для связи. Стандартно в наличии два USB, Ethernet и VGA.

Более наглядно расположение основных частей изображено на рис. 2.

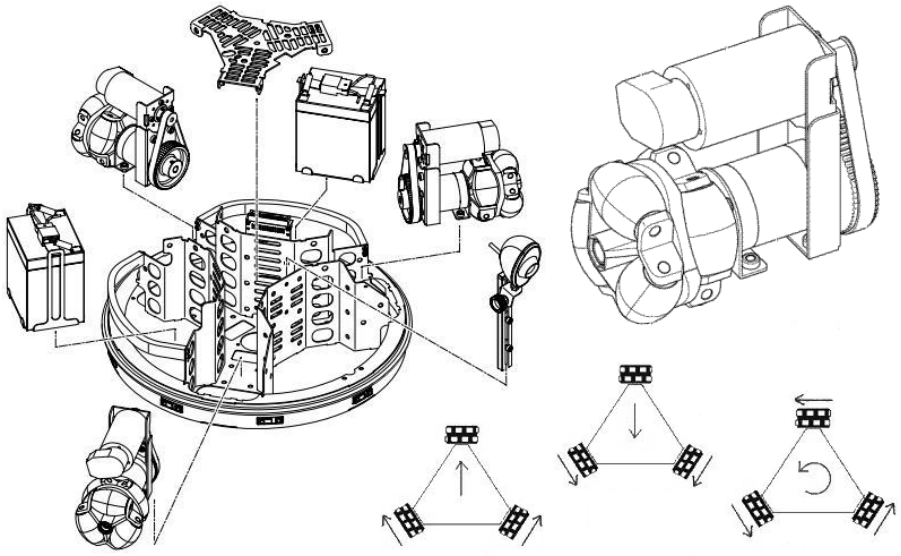

Рис. 2: Схема устройства платформы.

Используемые программные средства

В качестве основного инструмента для написания программы использовался ROS. Как уже говорилось, ROS – это программное обеспечение, облегчающее разработку и объединение разных программных компонентов для робототехнических (и даже больше) систем. Он обеспечивает такие стандартные службы операционной системы, как аппаратную абстракцию, реализацию часто используемых функций, передачу сообщений между процессами, удобное управление пакетами и многое другое. Иначе говоря, ROS избавил людей от потребности постоянно переизобретать велосипеды.

Для управления и получения информации с датчиков робота использовалось API, предоставляемое производителем[3]. Одной важной особенностью является то, что оно позволяет запускать управляющую программу как на самом роботе, так и на любой другой машине, удаленно подключенной к нему по сети.

Описание работы программы

Основная идея программы заключалась в предотвращении столкновения робота с препятствиями при ручном управлении. Изначально предполагалось использовать инфракрасные датчики по периметру робота, но в последствии, для большей универсальности программы и приобретения навыков обработки более сложной фотометрической информации, было решено использовать лидар.

В процессе работы выяснилось, что в стандартном пакете для ROS, предлагаемом производителем робота, уже имеется реализация промежуточного узла, для предотвращения столкновений робота с препятствиями, но его функциональность никуда не годится – робот начинает снижать скорость по всем направлениям пропорционально расстоянию до объекта и в момент полной остановки его невозможно сдвинуть с места, так как

[3]Всю информацию по установке и использованию API, а так же по доступным способам программирования robotino можно найти на wiki страничке по адресу http://wiki.openrobotino.org

даже поворот вокруг оси или движение в сторону, обратную от объекта снижено до минимума.

Основное отличие предлагаемой мной реализации в том, что робот не станет останавливаться, а изменит вектор скорости и плавно объедет препятствие.

Алгоритм работы следующий: Поступающий с лидара массив данных (значений расстояний до окружающих объектов) разбивается на 18 секторов, для каждого такого сектора находиться минимальное значение, если найденное значение меньше заданного (заданное значение определяется исходя из габаритов робота), то в зависимости от сектора вычисляется поправка для значений скорости по каждой составляющей. Для каждого сектора реализован ПИ-регулятор и под вычислением поправки здесь понимается постепенное повышение корректирующей скорости пропорционально расстоянию до объекта. Далее эта поправка вычитается из управляющей скорости и результат публикуется в топик управления роботом.

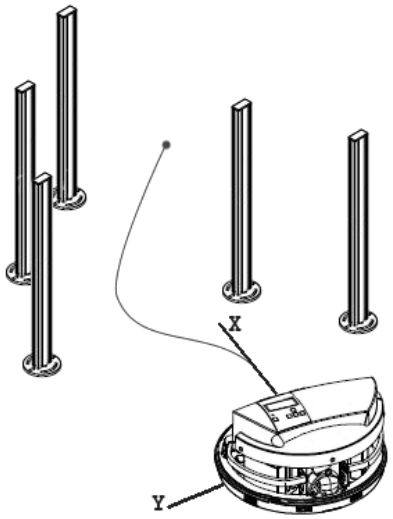

Рис. 3: Траектория движения.

Рассмотрим выше сказанное на примере (Рис. 3). Предположим, что робот управляется с клавиатуры ПК – оператор нажимает клавишу 'вперед', соответствующая программа ловит это нажатие, формирует сообщение типа geometry_msgs/Twist[4] и публикует его в топик bk_cmd_vel. Затем реализуемый узел получает это сообщение, так же получает сообщение с лидара, вносит поправку в скорость и окончательно публикует его в топик cmd_vel. Платформа может двигаться прямолинейно во всех направлениях на плоскости, а так же вращаться вокруг вертикальной оси (см. рис. 2), в связи с этим каждый сектор имеет специфичные только для него коэффициенты пропорциональности. Например крайний левый сектор (соответствует промежутку от 180 до 170 градусов) вносит дополнительную скорость только в правую сторону, угловая скорость и скорость по X не претерпевают изменений. Приближаясь к передним секторам растет поправка по X с отрицательным знаком, а так же поправка по угловой скорости. Если объект находится в переднем правом секторе, то робот сначала сбавит скорость и начнет поворачивать в сторону, где объекты находятся дальше, затем скорость по X начинает возрастать до задаваемого уровня, но робот все еще будет продолжать корректировать угловую и Y составляющие скорости, тем самым огибая объект.

Данный узел можно использовать с любой платформой сходной комплектации. При правильном подборе коэффициентов он не будет мешать выполнению основной задачи решаемой роботом, но поможет предотвратить столкновения с различными объектами в динамически меняющейся среде (например, ваш коллега задумался и вышел на полигон).

Из возможных улучшений программы можно отметить: реализацию дифференциальной составляющей для регуляторов, возможность поправки так же по инфракрасным датчикам по периметру робота, описание поведения узла в ситуациях, когда необходимо протиснуться в очень узкий проем (из возможных решений исключение X-составляющей из поправки или полное отключение функциональности узла).

Подведение итогов

За время прохождения практики, я приобрел ценные практические навыки работы с ROS, системами моделирования Stage, Gazebo. Ознакомился с устройством сложных робототехнических комплексов и научился писать под них приложения. Разобрался с основными стандартными методами решения таких задач, как локализация, одновремен-

[4]Формат сообщения можно посмотреть командой #rosmsg show geometry_msgs/Twist

ное картирование и локализация, планирование и другими входящими в стек пакетов навигации ROS, а так же со множеством других интересных и полезных вещей.

В дальнейшем планирую продолжить заниматься изучением различных аспектов задач, ставящихся перед командами-участниками соревнований RoboCup@Work в рамках выбранного мной направления.

Хотелось бы выразить благодарность сотрудникам студенческого конструкторского бюро на базе кафедры СУиИ Антонову Е.С. и Азбекяну А.А. за лекции и помощь, а так же руководителю скб Капитонову А.А. за организацию всего, что происходило с нами за этот месяц.

Список используемых материалов

1. Липпман С. Б., Язык программирования C++ : базовый курс / Стенли Б. Липпман, Жози Лажойе, Барбара Э. Му; [пер. с англ. и ред.: В.А. Коваленко]. – 5-е изд. – Москва [и др.] : Вильямс, 2014. – 1118 с. : ил., табл. ; 24 см. – Предм. указ.: с. 1103-1118

2. Документация по программному пакету ROS [Электронный ресурс] / Open Source Robotics Foundation. – Электрон. дан. – Режим доступа: http://wiki.ros.org/, свободный. – Загл. с экрана. – Яз. англ.

3. Jason M. O'Kane. A Gentle Introduction to ROS. University of South Carolina: Department of Computer Science and Engineering, 2014.