Санкт-Петербургский государственный университет

Кафедра системного программирования

Аудучинок Евгений Павлович

Интеграция робототехнических библиотек ROS с контроллером ТРИК

Курсовая работа

Научный руководитель:
ст.пр. Кириленко Я. А.

Санкт-Петербург
2016

Оглавление

Введение

Возможность переиспользования кода и масштабируемость — важные свойства программных систем. Для достижения этих свойств в робототехнических системах используются наборы библиотек, упрощающих взаимодействие между отдельными частями системы, а также включающих в себя реализации многих алгоритмов, часто используемых при создании робототехнических моделей, например, алгоритмы для навигации, симуляции и визуализации. Такие наборы библиотек называются robotics middleware (наборы библиотек промежуточного уровня для программирования роботов).

В последние годы усилиями различных институтов активно развивается ROS[1] (Robot Operating System) middleware, часто используемая как для решения задач быстрого прототипирования или тестирования алгоритмов, так и при создании промышленных роботов. ROS включает в себя множество реализованных алгоритмов и драйверов для периферии роботов, является распределённой системой, работающей на одном или нескольких устройствах.

Контроллер ТРИК — компьютер, работающий под управлением операционной системы Linux, оснащённый портами для подключения периферийных устроств, таких как моторы, сервоприводы, аналоговые и цифровые датчики. Использование библиотек ROS на контроллере ТРИК может облегчить задачи прототипирования робототехнических моделей для образовательных и научных целей.

В рамках дипломной работы Евгения Новожилова в 2015 [3] году был реализован прототип системы, который выявил недостатки, связанные со сложностью интеграции в прошивку контроллера и низкой производительностью, которые необходимо решить для дальнейшего использования ROS на контроллере ТРИК.

[1] http://www.ros.org

1. Обзор

1.1. ТРИК

Прошивка контроллера ТРИК содержит набор библиотек trikRuntime, который включает библиотеки для управления компонентами контроллера, реализации пользовательского интерфейса, предоставления линеаризованных данных с различных датчиков. Для взаимодействия trikRuntime с клиентскими программами может использоваться библиотека trikControl, предоставляющая доступ к периферии контроллера.

1.2. BitBake

Для сборки прошивки контроллера используется инструмент BitBake[2]. Каждый компонент прошивки собирается по правилам, заданым в файле, называемом рецептом. Существуют рецепты для сборки ядра Linux, множества библиотек и утилит, что позволяет, комбинируя рецепты, собрать прошивку с необходимым функционалом. BitBake анализирует все заданные рецепты и выполняет необходимые задачи в порядке, учитывающем зависимости между рецептами.

1.3. ROS

Распределённая система ROS состоит из узлов (nodes), обменивающихся сообщениями с помощью каналов (topics) с использованием TCP или UDP и предоставляющих доступ к сервисам по протоколу XML-RPC. Все сообщения в рамках одного канала имеют один тип и определают тип данного канала. Реализация ROS включает в себя часто используемые типы сообщений, описывающие различные данные, например, изображения, облака точек, перемещение робота или примитивные типы.

[2]https://www.yoctoproject.org/docs/1.6/bitbake-user-manual/bitbake-user-manual.html

Для осуществления связи между узлами сети используется мастер-узел (master), регистрирующий запущенные узлы и предоставляющий информацию о каналах и сервисах. Перед началом публикации сообщений в канал каждый узел сообщает об этом мастер-узлу, для получения сообщений узлы подписываются на каналы. ROS предоставляет клиентские библиотеки для реализации узлов на различных языках программирования.

2. Постановка задачи

2.1. Модификация рецепта сборки trikRuntime

Было обнаружено, что реализация рецепта trikRuntime не позволяла рецептам других компонетов прошивки использовать данные библиотеки как зависимости. Для решения данной проблемы необходимо изменить рецепт сборки trikRuntime, сделав возможным разрешение ссылок на него из других исполняемых файлов.

2.2. Реализация узлов для взаимодействия контроллера со средой ROS

Для использования периферии контроллера в распределённой сети ROS, необходимо реализовать узел, публикующий информацию, полученную с сенсоров контроллера, и способный получать команды от других узлов сети.

2.3. Тестирование производительности и оптимизация автономных моделей

Для оценки эффективности использования библиотек ROS была поставлена задача реализовать тестовые модели, позволяющие оценить производительность контроллера при использовании автономной и распределённой моделей. В случае неудовлетворительных результатов требуется изучить способы оптимизации работы ROS на контроллере, провести оптимизацию и повторно измерить производительность моделей.

2.4. Обновление библиотек ROS

Запуск ROS на контроллере ТРИК предполагает наличие клиентских библиотек ROS в составе прошивки контроллера, для этого в сборку прошивки были включены рецепты сборки из проекта meta-ros[3]. На

[3]https://github.com/bmwcarit/meta-ros

момент начала данной работы прошивка содержала устаревшие рецепты meta-ros, имеющие проблемы с производительностью ROS на контроллере и совместимостью с некоторыми библиотеками, например, драйвером для Kinect. Для решения данных проблем было необходимо обновить рецепты meta-ros и проверить успешность сборки прошивки контроллера.

2.5. Добавление недостающих утилит ROS в прошивку

ROS содержит утилиты, позволяющие анализировать компоненты распределённой сети, однако они не были включены в сборку прошивки. Было решено включить недостающие утилиты ROS в сборку прошивки контроллера для анализа работы автономных моделей.

2.6. Включение в прошивку поддержки Kinect

Для многих реализованных алгоритмов навигации в поставке ROS требуется наличие поддержки сенсоров, позволяющих строить облака точек (3D-реконструкцию сцены, видимой сенсору). Для дальнеших экспериментов было решено добавить поддержку Kinect в прошивку контроллера и добавить рецепты для сборки узлов, предоставляющих к нему доступ из среды ROS.

3. Решение

3.1. Модификация рецепта сборки trikRuntime

Для простоты обновления библиотек trikRuntime с помощью среды TRIK Studio в проекте ТРИК было решено вынести библиотеки trikRuntime в отдельную директорию /home/root/trik. Для успешной компоновки других компонентов во время сборки прошивки с помощью BitBake необходимо, чтобы данные библиотеки, так же как и заголовочные файлы, были расположены в /usr/lib и /usr/include, согласно стандарту FHS [1]. Перенос библиотек в /usr/lib может создать проблему обновления прошивки устройств у других пользователей с использованием TRIK Studio. Рецепт сборки trikRuntime был модифицирован[4] таким образом, чтобы сделать возможным компоновать данные библиотеки с другими программами, но избежать проблемы обновления прошивки у других пользователей.

Также была решена проблема запуска программ, использующих данные библиотеки, при вызове их извне /home/root/trik, добавлением переной окружения с указанием пути установки данных библиотек.

3.2. Анализ производительности и оптимизация автономных моделей

Набор клиентских библиотек ROS для C++ (roscpp) включает в себя библиотеку pluginlib[5], позволяющую узлам динамически загружать и выгружать плагины, меняя поведение узлов или расширяя их возможности. В числе таких плагинов есть библиотека Nodelet[6], позволяющая запустить несколько узлов внутри одного процесса и предоставляющая каждому из них доступ к ROS API. Использование нодлетов вместо узлов системы, работающих в разных процессах (на одном устройстве), позволяет значительно снизить нагрузку системы, так как исключают-

[4]https://github.com/trikset/meta-trik/pull/36

[5]http://wiki.ros.org/pluginlib

[6]http://wiki.ros.org/nodelet

ся шаги сериализации и десериализации сообщений. На текущий момент нодлеты не поддерживают предоставление сервисов.

Использование нодлетов накладывает ограничения на реализацию таких узлов с учётом особенностей Nodelet API [2] и не позволяет использовать клиентскую библиотеку для Python, что усложняет решение задачи быстрого прототипирования модели, но позволяет значительно повысить производительность при запуске нескольких узлов на одном устройстве.

В рамках работы были реализованы[7] узел TrikNode и нодлет TrikNodelet, предоставляющие доступ к периферии контроллера.

Для сравнении производительности автономных моделей с использованием узлов и нодлетов, контроллере по очереди были запущены два набора узлов:

- rosmaster, TrikNode и узел с логикой
- rosmaster, TrikNodelet и нодлет с логикой

С помощью утилиты top была измерена нагрузка CPU, создаваемая узлами системы и инфраструктурой ROS. Приблизительные значения изображены в таблице:

	Узлы	rosmaster
Nodes	70%	10%
Nodelets	10%	10%

Существуют способы существенно уменьшить нагрузку rosmaster, отключив логгирование в файловую систему.

3.3. Ускорение запуска мастер-узла на контроллере

Во время выполнения Python-скрипт запуска rosmaster рекурсивно анализирует содержимое директории, куда устанавливаются пакеты ROS, что может привести к значительному увеличению времени запуска rosmaster на контроллере, если данная директория содержит много

[7] https://github.com/auduchinok/trik-ros

файлов[8]. На момент начала работы в сборку прошивки были включены рецепты meta-ros, устанавливающие все пакеты ROS в директорию /usr. В данной директории, помимо пакетов ROS, располагается большое количество системных библиотек и других файлов, что привело к тому, что запуск rosmaster и двух нодлетов занимает около 7 минут.

Решением данной проблемы является перенос файлов ROS в отдельную директорию. Были изучены рецепты сборки ROS и возможные способы их модификации, а также было обнаружено, что в meta-ros данная проблема уже была решена[9] в последующих версиях рецептов. При обновлении рецептов утилиты и библиотеки ROS были перенесены в директорию /opt/ros. Данная оптимизация позволила сократить запуск rosmaster и двух нодлетов на контроллере до одной минуты.

3.4. Использование библиотек ROS и запуск распределённой модели

Для эксперимента с использованием существующих библиотек ROS было решено реализовать управление передвигающимся роботом с помощью геймпада, подключенного к компьютеру. Был использован узел Joy из поставки ROS, считывающий информацию с геймпада и публикующий сообщения с обработанными данными о нажатых клавишах и положении аналоговых джойстиков. Для управления роботом был реализован ещё один узел, принимающий сообщения с информацией о состоянии джойстика, и преобразовывающий их в команды перемещения робота в виде новых сообщений сети. Узел, работающий на контроллере, находившемся в одной беспроводной сети с компьютером, принимал сообщения с командами передвижения с использованием протокола UDP[10].

Запуск данного узла на контроллере занимает несколько секунд, что является удовлетворительным. Измерение нагрузки с помощью утилиты top показало, что данный узел использует около 5% CPU, что поз-

[8]https://github.com/bmwcarit/meta-ros/issues/214

[9]https://github.com/bmwcarit/meta-ros/pull/286

[10]http://docs.ros.org/api/roscpp/html/classros_1_1TransportHints.html

воляет использовать модели с более сложными вычислениями на контроллере и пересылать значительно больше сообщений.

Разделение задач между узлами позволило быстро реализовать данную модель, из чего следует, что ROS можно использовать для быстрого прототипирования при реализации других моделей в будущем.

4. Возможные дальнейшие задачи

Добавление поддержки Kinect в прошивку контроллера может позволить использовать навигационный стек ROS. Данный стек включает алгоритмы для построения карты помещения, передвижения с избежанием столкновений с препятствиями, поиск путей до заданных объектов.

Использование Kinect требует обработки и передачи больших объёмов данных, поэтому реализация модели с использованием данного стека на контроллере ТРИК представляет из себя отдельную большую задачу, которая не была рассмотрена в данной работе.

Отключение логгирования файловой в файловую систему может значительно снизить нагрузку rosmaster.

5. Полученные результаты

В рамках данной работы были получены следующие результаты:

1. Изучены инструменты для сборки прошивки контроллера ТРИК
2. Модифицирован рецепт сборки trikRuntime, что позволило добавлять зависимые от него компоненты в прошивку контроллера
3. Реализован и добавлен в прошивку узел, предоставляющий интерфейс к периферии контроллера
4. Произведен анализ быстродействия и оптимизаци автономных моделей
5. Уменьшено время запуска автономных моделей
6. Проведена апробация интеграции на примере распределённой модели с использованием существующих библиотек ROS
7. Добавлены недостающие утилиты ROS в прошивку контроллера
8. Обновлены рецепты сборки meta-ros
9. Добавлена поддержка Kinect в прошивку для дальнейших экспериментов

Тестирование результатов интеграции показало эффективность использования библиотек ROS для прототипирования робототехнических моделей на контроллере ТРИК для образовательных или научных целей.

Список литературы

[1] Linux Filesystem Hierarchy. — 2004. — URL: http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/.

[2] Nodelet Everything. — 2015. — URL: http://www.clearpathrobotics.com/assets/guides/ros/Nodelet%20Everything.html.

[3] Новожилов Евгений. Интеграция робототехнической ОС (ROS) с кибернетическим контроллером ТРИК. — 2015. — URL: http://se.math.spbu.ru/SE/diploma/2015/s/544-Novozhilov-report.pdf.