Санкт-Петербургский государственный университет

Программная инженерия
Кафедра системного программирования

Гагина Лада Владиславовна

Сравнение промежуточного программного обеспечения для робототехнических платформ

Бакалаврская работа

Научный руководитель:
д. ф.-м. н., профессор Терехов А. Н.

Рецензент:
к. т. н., асс. каф. СУИ НИУ ИТМО Капитонов А. А.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Software Engineering
Software Engineering Department

Lada Gagina

Comparison of middleware for robotic platforms

Graduation Thesis

Scientific supervisor:
professor Andrey Terekhov

Reviewer:
assistant Aleksandr Kapitonov

Saint-Petersburg
2016

Оглавление

Введение

Робототехника является одной из самых быстроразвивающихся отраслей в современном мире. Роботизированные системы получают применение в военном деле, медицине, искусстве и в быту. Роботы способны выполнять многие виды работы быстрее и качественнее, чем человек, чем объясняется их возрастающая популярность. В связи с вышеуказанным, можно отметить высокий спрос на специалистов в области робототехники, а отсюда вытекает потребность в робототехнических образовательных комплексах. В ответ на такую потребность на математико-механическом факультете СПбГУ усилиями кафедр системного программирования и теоретической кибернетики разрабатывается контроллер ТРИК.

Вместе с развитием аппаратных платформ идёт также развитие ПО в сфере робототехники. Алгоритмы становятся всё более сложными и многоуровневыми, а приложения зачастую нелегко масштабировать. С другой стороны, так же как и в программировании компьютерных систем, в программировании для робототехнических платформ существует множество типичных задач. Такие действия, как работа с данными датчиков и приводов, организация межпроцессного взаимодействия, сбор информации о состоянии робототехнической системы, реализация распространённых алгоритмов робототехники: избегание препятствий, планирование пути и многие другие, решаются схожим образом на разных системах и могут быть вынесены на отдельный уровень абстракции. Эту задачу успешно решает программное обеспечение промежуточного уровня (middleware).

Дейв Баккен описывает промежуточное программное обеспечение как класс технологий, призванных поддерживать сложность и гетерогенность, присущую распределённым системам [4]. Действительно, за многими известными роботами стоят различные системы, управляющие периферией, структурирующие код и упорядочивающие процесс

разработки приложений. Зачастую инструментарий создаётся исключительно для одной робототехнической системы, что объясняет большое количество существующих систем промежуточных уровня.

Главное преимущество промежуточного программного обеспечения — возможность переиспользования большого количества готовых компонентов. Существует множество научных статей, описывающих интересные разработки различных лабораторий, результаты которых реализованы в рамках некоторой системы промежуточного уровня. При адаптации таких систем для контроллера ТРИК можно будет использовать эти достижения при минимуме затраченных на реализацию усилий. Таким образом, появляется задача выбора и адаптирования некоторой промежуточной программной системы для контроллера ТРИК.

Стоит отметить, что на кафедре системного программирования уже существуют работы по использованию ROS (Robot Operating System) на контроллере ТРИК, однако существуют интересные разработки, основывающиеся и на других инструментариях, поэтому есть необходимость рассмотреть и другие системы промежуточного уровня.

1. Постановка задачи

Таким образом, целью данной работы является исследование промежуточного программного обеспечения (middleware) робототехники: сбор и анализ информации о существующих вариантах, после чего выработка критериев, по которым далее будут отобраны и адаптирваны подходящие на ТРИК.

В ходе работы цель была разбита на подзадачи:

1. Провести сравнение существуюих систем промежуточного уровня робототехники.
2. Разработать критерии для выбора вариантов промежуточного ПО, которые будут адаптированы для контроллера ТРИК.
3. Адаптировать выбранные системы для контроллера ТРИК.
4. Провести апробацию результатов адаптирования выбранных систем промежуточного уровня.

2. Обзор

2.1. ТРИК

ТРИК (см. рис. 1) — кибернетический контроллер под управлением операционной системы семейства Linux, разрабатываемый на математико-механическом факультете преподавателями и студентами кафедр системного программирования и теоретической кибернетики. Он предназначен для управления роботами, беспилотными летательными аппаратами, средствами передвижения и встраиваемыми устройствами. Существует набор библиотек `trikRuntime`, позволяющий реализовывать алгоритмы, использующие контроллер и подключенную к нему периферию.
Логическая модель робота представлена библиотекой `trikControl` (см. рис. 2), которая предоставляет интерфейс для программирования роботов с помощью `QtScript`[1] или C++ с использованием Qt. Ниже перечислены некоторые классы `trikControl`: `Brick` — класс, отвечающий за контроллер, инициализирующий периферию робота и дающий к ней доступ, `Sensor`, `ServoMotor` и `PowerMotor` — классы, отвечающие за работу с сенсорами, сервомоторами и силовыми моторами, соответственно. Каждый из них даёт возможность прочитать или выставить значение, что выполняется путём записи или чтения значения в файле, отвечающем за соответствующее устройство.

2.2. YARP

YARP (Yet Another Robot Platform) [11] — набор библиотек, протоколов и инструментов, призванный разделить программные модули и механику. YARP позволяет организовать связи между датчиками, процессором и исполнительными устройствами, оставляя систему слабо связанной, поэтому в неё легко вносить изменения. YARP создан для гуманоидного робота iCub, но утверждается, что его архитектура позволяет легко наладить взаимодействие со сторонним кодом. Имеется

[1] `http://doc.qt.io/qt-5/qtscript-index.html` (дата обращения: 21.05.2016)

Рис. 1: Робот на базе контроллера ТРИК. Источник: http://www.pushkin-news.ru/news/2015-09-02/obuchis-robototekhnike-v-trik-studii/ (дата обращения: 20.05.2016)

достаточно обширная документация.

YARP весьма известен в робототехническом сообществе, однако существует мало информации о возможностях его работы на платформах, отличных от iCub.

2.3. MOOS

MOOS (Mission Oriented Operating Suite)[2] — кроссплатформенная система промежуточного уровня, представленная набором слоёв: Core MOOS — сетевая архитектура для поддержки коммуникации программных модулей и Essential MOOS — набор приложений, использующих Core MOOS. Essential MOOS предлагает обширный набор приложений

[2] http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php/Main/Introduction (дата обращения: 20.05.2016)

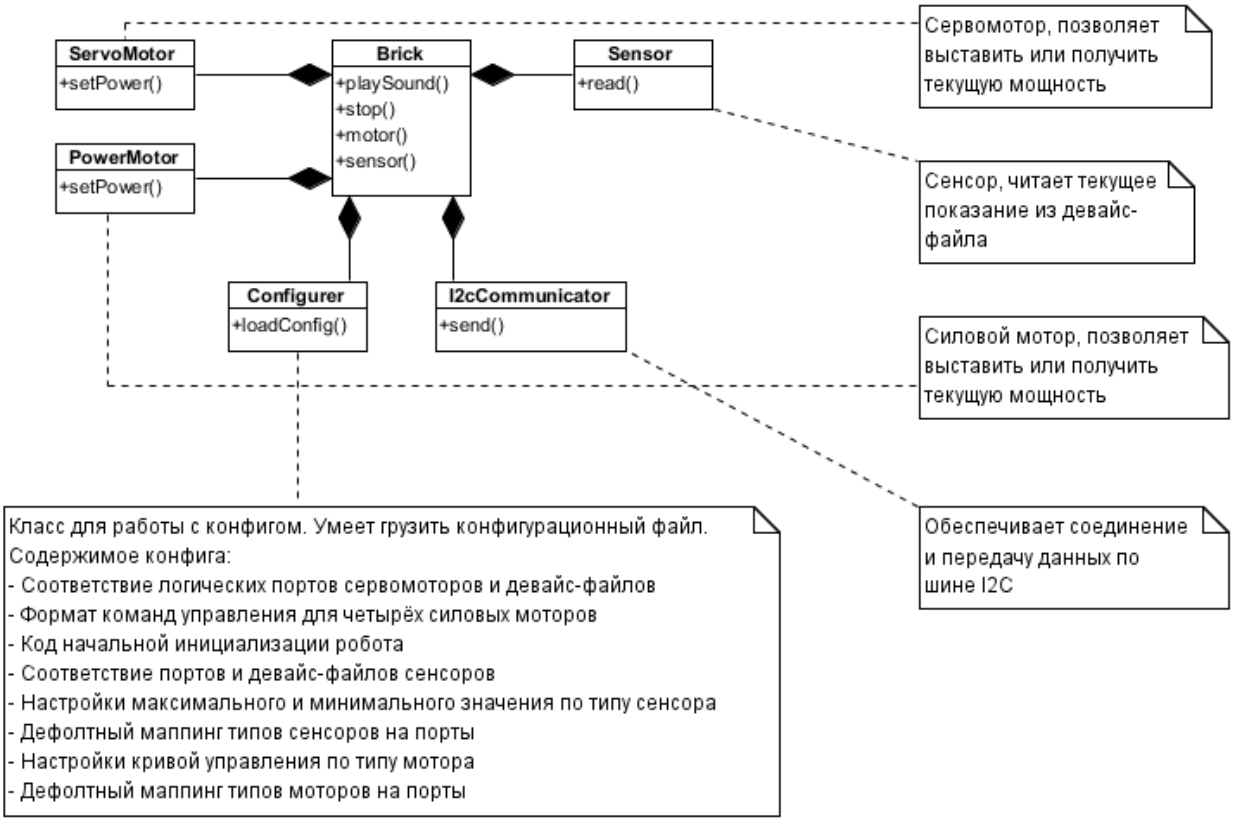

Рис. 2: Состав `trikControl`. Источник: `https://github.com/trikset/trikRuntime/wiki/Архитектура` (дата обращения: 20.05.2016)

для типичных задач, например, для управления процессами, логирования, и многих других. [14]
Большинство компонентов MOOS предназначены для водных видов роботов, поэтому адаптация его для роботов, использующих контроллер ТРИК, может оказаться менее продуктивной, чем адаптация других вариантов.

2.4. OPRoS

OPRoS (Open Platform for Robotic Services)[3] — робототехническая программная платформа для управления компонентами, за которыми скрывается как аппаратное обеспечение, так и распространённые алгоритмы робототехники. Имеет собственную интегрированную среду разработки для создания компонентов, робототехнический симулятор, а также средства тестирования и верификации. [8] Как и во многих других, компоненты используют клиент-серверную модель взаимодействия, обмениваясь сообщениями согласно паттерну издатель-подписчик.
К сожалению, последняя активность на сайте датируется 2012 годом, кроме того, ссылки на исходный код системы неактивны.

2.5. CLARAty

CLARAty (Coupled Layer Architecture for Robotic Autonomy)[4] — робототехнический инструментарий, созданный совместными усилиями лаборатории реактивного движения (Jet Propulsion Laboratory), исследовательского центра Эймса (NASA Ames Research Center), университета Карнеги — Меллон и университета Миннесоты. [13, 7] Архитектура CLARAty состоит из двух слоёв: функционального слоя и слоя принятия решений. Функциональный слой включает в себя общие компоненты, такие как цифровые и аналоговые входы и выходы, управление дви-

[3] `https://sites.google.com/a/odm-tech.com/opros/home/about-opros` (дата обращения: 20.05.2016)

[4] `https://claraty.jpl.nasa.gov/man/overview/index.php` (дата обращения: 20.05.2016)

жением, компьютерное зрение, навигация, построение карт, планирование пути. Слой принятия решений занимается сбором информации о ресурсах и состоянии системы. Он объединяет под собой планировщики и их эвристики, исполнителей, базы действий. Слои взаимодействуют по клиент-серверной модели, используя паттерн издатель-подписчик для обмена сообщениями.
Несмотря на всю привлекательность CLARAty, часть модулей являются проприетарными, что уменьшает ценность её адаптирования.

2.6. MIRO

MIRO[5] — объектно-ориентированная система промежуточного уровня, призванная упростить управление мобильными роботами. Утверждается, что MIRO успешно используется в нескольких проектах с потребностью в автономном управлении мобильными роботами, таких как построение карты окружающего пространства, локализации себя в ней и навигации. [12, 9]
К сожалению, официальный сайт долгое время недоступен и альтернативной документации нет. По этой причине MIRO не может быть выбрана для адаптации на контроллер ТРИК.

2.7. ORoCoS

ORoCoS (Open Robot Control Software)[6] — модульный инструментарий для управления роботами, который предоставляет инфраструктуру и некоторые готовые компоненты для написания приложений на C++. Можно выбрать компоненты, написанные сообществом, или написать свои. Среди приложений, использующих ORoCoS: отслеживание движения в 3D, система автономного управления автомобилем и ещё несколько проектов. [6]
ORoCoS несколько отличается от большинства остальных систем про-

[5] `https://github.com/hhutz/Miro` (дата обращения: 20.05.2016)
[6] `http://www.orocos.org/` (дата обращения: 20.05.2016)

межуточного уровня отсутствием симулятора и возможности коммуникации компонентов, запущенных на разных машинах.

2.8. URBI

URBI (Universal Real-time Behavior Interface)[7] — робототехническая программная платформа, разработанная компанией Gostai. Включает в себя параллельный событийный язык `urbiscript`, с помощью которого можно управлять компонентами (`UObject`). Совместима с ROS и симулятором Webots. [3]
На официальном сайте компании Gostai отсутствуют рабочие ссылки на документацию и на исходный код URBI, по этой причине в дальнейшем платформа не была рассмотрена.

2.9. OpenDaVINCI

OpenDaVINCI[8] — компактная система промежуточного уровня, написанная на стандартном C++. Она имеет встроенные возможности параллельного программирования, где межпроцессное взаимодействие реализовано посредством использования разделяемой памяти и семафоров. Есть встроенные средства симуляции и визуализации. Инструментарий OpenDaVINCI позволяет производить вычисления в реальном времени.
Возможности OpenDaVINCI активно применяются в программировании беспилотных автомобилей [5] и, несмотря на то, что система весьма новая и публикаций о ней мало, она представляет большой интерес.

2.10. MAVLink

MAVLink (Micro Air Vehicle Link)[9] — легковесная header-only библиотека обмена сообщениями для беспилотных летательных аппаратов (БПЛА). Она реализует эффективное взаимодействие БПЛА между

[7] `http://www.gostai.com/products/urbi/` (дата обращения: 20.05.2016)
[8] `http://opendavinci.cse.chalmers.se/www/` (дата обращения: 20.05.2016)
[9] `http://qgroundcontrol.org/mavlink/start` (дата обращения: 20.05.2016)

собой и с наземной станцией управления. К сожалению, сейчас в лаборатории библиотеки управления БПЛА не так актуальны, как промежуточное программное обеспечение для наземных роботов, однако использование MAVLink с контроллером ТРИК может стать перспективным направлением для будущей работы.

2.11. Skilligent

Skilligent — коммерческая робототехническая платформа, в рамках которой реализовано множество компонентов, таких как распознавание речи, алгоритмы компьютерного зрения, координация движения и машинное обучение.
Несмотря на большое количество готовых компонентов, платформа Skilligent не может быть использования по причине того, что она является коммерческим продуктом.

2.12. Player

Player[10] — сетевой сервер для управления роботом, который предоставляет простой интерфейс для взаимодействия с сенсорами и исполнительными устройствами робота по IP-сети. Клиентские программы используют TCP-сокет для получения данных с датчиков, передачи команд исполнительным устройствам и конфигурирования оборудования на лету. Player поддерживает несколько робототехнических систем и множество распространённых сенсоров; утверждается, что его модульная архитектура позволяет легко добавлять поддержку новых. Запускается на Linux (PC и embedded), Solaris и *BSD. [17]
На сайте есть огромный список организаций из стран по всему миру, использующих Player для своих исследований, но России среди них ещё нет.[11]

[10] `http://playerstage.sourceforge.net/` (дата обращения: 20.05.2016)
[11] `http://playerstage.sourceforge.net/wiki/PlayerUsers` (дата обращения: 20.05.2016)

2.13. ROS

ROS — наиболее популярная на данный момент система промежуточного уровня. Она предоставляет низкоуровневое управление датчиками и приводами, реализацию типичных для робототехники алгоритмов, передачу сообщений между процессами и управление пакетами. ROS состоит из узлов, сообщений, которыми они обмениваются посредством одноранговой сети и топиков[12] для обмена сообщениями.
В дальнейшей работе ROS не будет рассмотрен, поскольку на кафедре системного программирования существует две работы [19, 18], посвящённые запуску и использованию ROS на контроллере ТРИК.

[12] `http://wiki.ros.org/Topics` (дата обращения: 16.05.2016)

3. Сравнение

3.1. Критерии

В результате прочтения обзорных статей [10, 16, 15] и при ближайшем рассмотрении вариантов были сформулированы критерии для выявления наиболее подходящего промежуточного программного обеспечения для адаптации на контроллер ТРИК:

1. Открытый исходный код.
2. Поддержка симуляторов.
3. Распределённая архитектура.
4. Популярность (количество статей, упоминающих данную систему промежуточного уровня).
5. Поддержка (дата выхода последней версии или последней ревизии в системе контроля версий).
6. Готовые компоненты (необходимые по мнению участников лаборатории робототехники).
7. Переносимость (по количеству платформ, на которые уже был выполнен перенос данной системы промежуточного уровня).

При их применении была получена таблица (см. табл. 3.1)

3.2. Выбор промежуточного ПО

Исходя из таблицы, наиболее перспективным вариантом для попыток адаптирования для платформы ТРИК является система промежуточного уровня Player. Большое количество статей, документации

	Открытый исх.код	Симулятор	Распр. арх.	Популярность	Поддержка	Готовые комп.	Переносимость
Player	Да	Да	Да	>10000	2010	работа с камерой, картами, звуком	>100
MIRO	Да	Да	Да	944	2015	работа с картами, навигация	-
CLARAty	Да	Да	Да	878	2014	-	>10
OPRoS	Да	Да	Да	373	2012	работа с камерой, картами, звуком	>10
ORoCoS	Да	Нет	Нет	1730	2012	работа с камерой, картами	>5
Skilligent	Нет	Нет	Да	47	-	работа с камерой, звуком, машинное обучение	-
YARP	Да	Да	Да	857	2016	работа с камерой, картами, звуком	9
MOOS	Да	Да	Да	1260	2015	работа с картами	-
URBI	Да	Да	Да	1070	-	-	-
OpenDaVINCI	Да	Да	Да	14	2016	беспилотные автомобили	6
MAVLink	Да	Да	Да	294	2016	беспилотные летательные аппараты	>10

Таблица 1: Сравнительная таблица систем промежуточного уровня.

и уже написанного доступного в сети кода позволит успешно адаптировать её даже с учётом того, что уже долгое время обновлений не происходит.

Кроме Player, интерес представляет система промежуточного уровня OpenDaVINCI, так как оно предоставляет интересные возможности навигации для беспилотных автомобилей по правилам дорожного движения. Оно весьма новое, однако существует несколько интересных его применений в моделях беспилотных автомобилей. Вместе с тем фактом, что сейчас в России и мире набирают популярность соревнования беспилотных автомобилей [1, 2], возможности OpenDaVINCI могут оказаться очень полезны.

4. Описание решения

4.1. Адаптация инструментария Player для контроллера ТРИК

Система промежуточного уровня Player состоит из сервера, запускающегося на роботе, и прикладных программ, которые запускаются поверх него. Сервер включает в себя драйверы, реализующие как работу с аппаратным обеспечением робота, так и распространённые алгоритмы, использующие эти данные. Получить доступ к данным драйверов можно с помощью прокси.

Работа с данными в Player реализована с помощью *интерфейсов* и *драйверов* (см. рис. 3). Интерфейсы определяют типы и содержание сообщений, которые будет обрабатывать и генерировать драйвер, также не представляется сложным добавить новый интерфейс при необходимости. Драйверы могут реализовывать один или несколько интерфейсов и могут быть как статическими (компилируются вместе с Player), так и поставляться как плагины к нему. Обычно драйвер создаётся либо для конкретной робототехнической платформы (и поставляет все необходимые ей интерфейсы), либо реализует некоторый алгоритм над данными других драйверов.

Player выделяется среди остального промежуточного ПО количеством платформ, на которых он используется. Поэтому первым шагом необходимо было найти готовые драйверы, способные работать с ТРИКом или его периферией. Таковым оказался `cameraUVC` — драйвер, реализующий взаимодействие с USB-камерой. Он был запущен на контроллере; с помощью утилиты `playerv`, поставляющейся вместе с инструментарием Player и предоставляющей визуализацию данных с датчиков, изображение камеры, подключенной к контроллеру, было выведено на экран ноутбука (см. рис. 4). Далее поверх драйвера `cameraUVC` был успешно запущен драйвер `CMVision` (Color Mashine Vision), позволяющий распознать на изображении монохромные объекты (см. рис. 5). Нужно отметить, что, в отличие от алгоритмов видеозрения ТРИК, `CMVision`

получает любое число объектов заданных цветов.

Однако, для того, чтобы полноценно использовать алгоритмы, реализованные в системе Player, необходимо иметь минимальный набор интерфейсов, поэтому основной частью работы над адаптированием стала реализация драйвера для роботов ТРИК, реализующего интерфейсы `position2d`[13] (управление перемещением двухколёсного робота с дифференциальным приводом, а также локальная навигация — определение его местоположения относительно точки старта с помощью текущей скорости и показаний гироскопа) и `ranger`[14] (обработка данных с инфракрасного датчика расстояния — легко может быть обобщён для лазеров и ультразвуковых датчиков). Эти драйверы были реализованы путём наследования от базового класса `ThreadedDriver`, имеющегося в Player. Для обработки сообщений был переопределён метод `ProcessMessage` для каждого интерфейса, а отправка данных серверу (`Player server`) выполнялась посредством использования метода `Publish`. Для взаимодействия с моторами и датчиками расстояния робота использовались возможности набора библиотек `trikRuntime`, разрабатывающегося на кафедре системного программирования. При этом класс `Brick`, отвечающий за работу с контроллером в целом и дающий доступ к периферии, инстанциируется в отдельном потоке. Обработка событий гироскопа также происходит в отдельном потоке. Синхронизация потоков происходит с помощью семафоров и флагов.

4.2. Апробация

Для апробации инструментария Player на контроллере ТРИК была выбрана задача следования за монохромным объектом с избеганием препятствий. С помощью драйвера `cameraUVC` было получено изображение USB-камеры, на котором средствами драйвера `CMVision` детектиру-

[13] `http://playerstage.sourceforge.net/doc/Player-cvs/player/group__interface__position2d.html`

[14] `http://playerstage.sourceforge.net/doc/Player-cvs/player/group__interface__ranger.html`

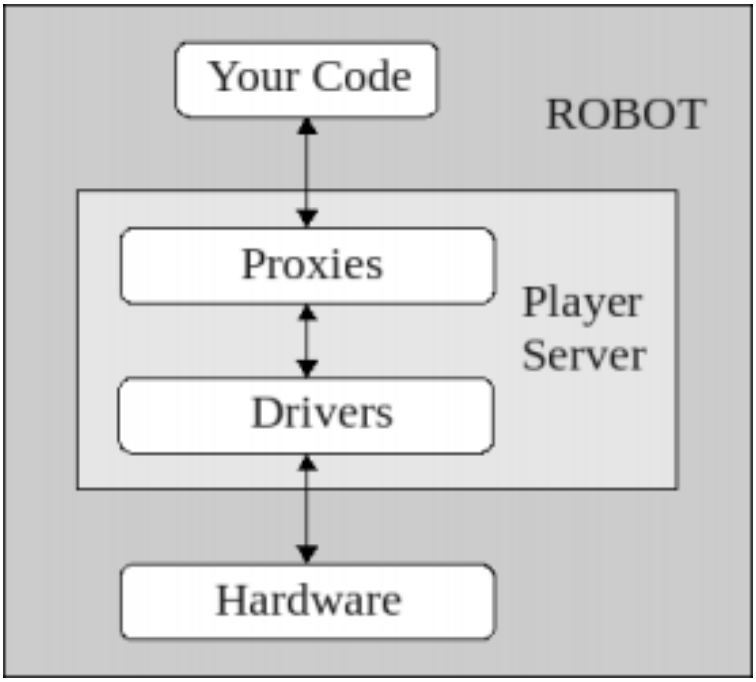

Рис. 3: Архитектура инструментария Player. Источник: `http://playerstage.sourceforge.net/doc/playerstage_instructions_2.0.pdf`

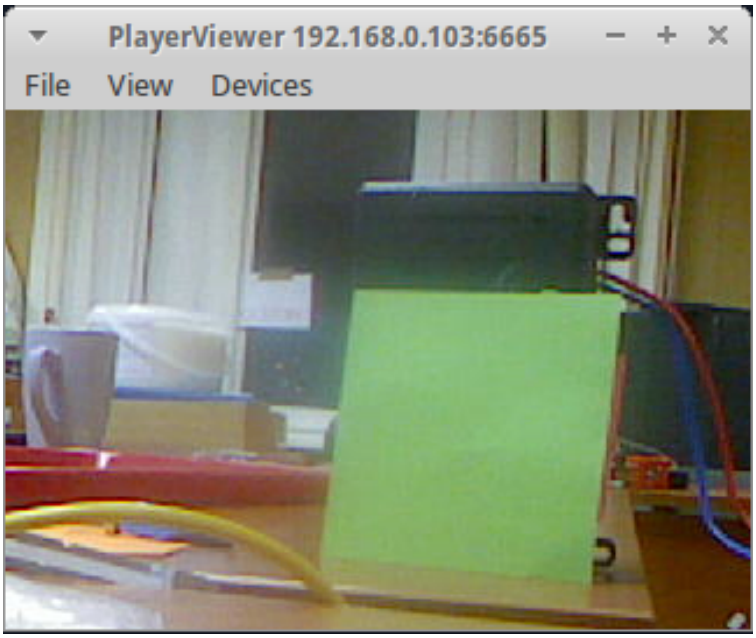

Рис. 4: Изображение камеры, подключенной к контроллеру ТРИК, выведенное на экран ноутбука

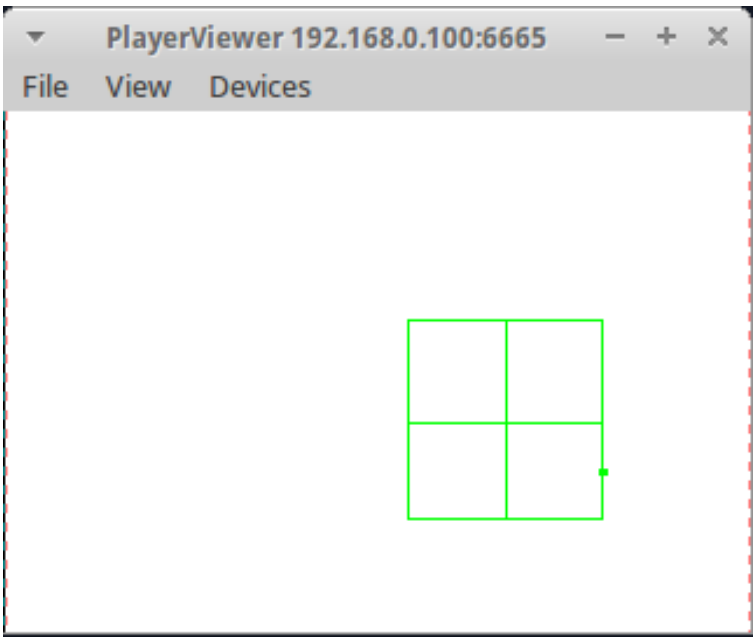

Рис. 5: Объект заданного цвета, распознанный на изображении камеры

ется объект заданного цвета. Была написана программа на языке C++, получающая координаты центра объекта посредством `BlobfinderProxy`[15] и осуществляющая передвижение робота посредством П-регулятора. Управление роботом использует два драйвера: `trik`, дающий возможность управлять моторами и получать данные от датчиков расстояния и `vfh`[16], осуществляющий избегание препятствий.

При автономной работе детекция объектов происходит на центральном процессоре контроллера, обработка сообщений для моторов происходит чаще обработки кадров, поэтому робот недостаточно оперативно реагирует на перемещение объекта. В стандартной реализации алгоритмов видеозрения ТРИК обработка видео происходит на DSP-сопроцессоре. В случае же с Player достаточной скорости обработки можно достигнуть, запустив `CMVision` отдельно на более мощном компьютере, передавая кадры с контроллера посредством протокола UDP.

[15] `http://playerstage.sourceforge.net/doc/Player-cvs/play/classPlayerCc_1_1BlobfinderProxy.html` (дата обращения: 21.05.2016)

[16] `http://playerstage.sourceforge.net/doc/Player-2.0.0/player/group__driver__vfh.html` (дата обращения: 21.05.2016)

Заключение

В рамках работы были получены следующие результаты.

1. Проведено сравнение существующего промежуточного программного обеспечения (middleware) робототехники.
2. Выработаны критерии для выбора промежуточного программного обеспечения, которое было адаптировано для контроллера ТРИК.
3. Инструментарий Player адаптирован для контроллера ТРИК.
4. Проведена апробация результатов адаптирования системы промежуточного уровня Player для контроллера ТРИК.

Список литературы

[1] URL: http://robolymp.ru/season-2016/rules-and-regulations/robotraffik/ (дата обращения: 20.05.2016).

[2] URL: http://roborace.com/ (дата обращения: 20.05.2016).

[3] Baillie Jean-Christophe. URBI: Towards a Universal Robotic Low-Level Programming Language // 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. — 2005.

[4] Bakken Dave. Middleware. — Dodrecht, The Netherlands : J. Urban and P. Dasgupta, Eds., Kluwer Academic, 2001.

[5] Berger Christian, Dukaczewski Michael. Comparison of Architectural Design Decisions for Resource-Constrained Self-Driving Cars – A Multiple Case-Study. — 2014. — URL: http://cs.emis.de/LNI/Proceedings/Proceedings232/2157.pdf.

[6] Bruyninck Herman. Open Robot Control Software: the OROCOS project. — 2001. — URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=933002.

[7] CLARAty: Challenges and Steps Toward Reusable Robotic Software. — 2006. — URL: https://claraty.jpl.nasa.gov/main/overview/publications/05_nesnas_challenges_jars.pdf.

[8] Choulsoo Jang Seung-Ik Lee Seung-Woog Jung Byoungyoul Song Rockwon Kim Sunghoon Kim, Lee Cheol-Hoon. OPRoS: A New Component-Based Robot Software Platform. — 2010. — URL: http://etrij.etri.re.kr/etrij/journal/article/article.do?volume=32&issue=5&page=646.

[9] Daniel Kr¨uger Ingo van Lil Niko Sunderhauf Robert Baumgartl Peter Protzel. Using and Extending the Miro Middleware for Autonomous Mobile Robots. — 2006. — URL:

https://www.tu-chemnitz.de/etit/proaut/mitarbeiter/rsrc/taros06-miro.pdf.

[10] Elkady Ayssam, Sobh Tarek. Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography // Journal of Robotics. — 2012. — URL: http://www.hindawi.com/journals/jr/2012/959013/.

[11] Giorgio Metta Paul Fitzpatrick, Natale Lorenzo. YARP: Yet Another Robot Platform. — URL: http://cdn.intechweb.org/pdfs/4161.pdf.

[12] Hans Utz Stefan Sablatnog Stefan Enderle Gerhard Kraetzschmar. Miro — Middleware for Mobile Robot Applications // IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION. — 2002. — URL: ftp://ftp.itam.mx/pub/alfredo/ROBOTICS/middleware/Miro.pdf.

[13] Issa A.D. Nemas Anne Wright Max Bajracharya Reid Simmons Tara Estlin. CLARAty and Challenges of Developing Interoperable Robotic Softwar. — 2003. — URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1249234.

[14] Michael R. Benjamin Henrik Schmidt Paul M. Newman, Leonard John J. Nested autonomy for unmanned marine vehicles with MOOS-IvP. — 2010. — URL: http://onlinelibrary.wiley.com/doi/10.1002/rob.20370/full.

[15] Mohamed Nader, Al-Jaroodi Jameela. Characteristics of Middleware for Networked Collaborative Robots. — 2008. — URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4543973.

[16] Nader Mohamed Jameela Al-Jaroodi, Jawhar Imad. Middleware for Robotics: A Survey. — 2008. — URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4681485.

[17] Vaughan Richard T., Gerkey Brian P. Really Reusable Robot Code and the Player/Stage Project // Springer Tracts on Advanced Robotics. — 2006. — URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.330.9919&rep=rep1&type=pdf.

[18] Аудучинок Евгений. Интеграция робототехнических библиотек ROS с контроллером ТРИК: курсовая работа. — 2016.

[19] Новожилов Евгений. Интеграция робототехнической ОС (ROS) с кибернетическим контроллером ТРИК: дипломная работа. — Санкт-Петербург : Санкт-Петербургский государственный университет, 2015. — URL: http://se.math.spbu.ru/SE/diploma/2015/s/544-Novozhilov-report.pdf.