УДК 004.896 DOI 10.23683/2311-3103-2018-1-294-309

Р.О. Лавренов, И.А. Маврин, Р.Н. Сафин, Е.А. Магид

РОБОТ «СЕРВОСИЛА ИНЖЕНЕР»: РАЗРАБОТКА СЕРВЕРА ПЕРЕДАЧИ ВИДЕОПОТОКА И ИНТЕРФЕЙСА УПРАВЛЕНИЯ ПОД ФРЕЙМВОРК ROS

Представлена разработка программного обеспечения (ПО) для управления и передачи видеопотока с камер российского гусеничного робота «Сервосила Инженер». Описываются узлы робота, его технические характеристики, характеристики операционной системы, особенности встроенного API и ПО, протокол данных для управления роботом и его телеметрии, и перечисляются недостатки системы и протокола, которые сделали необходимой разработку собственного ПО. При помощи разработанного ПО пользователь может управлять роботом, не используя джойстик. Разработанное ПО состоит из библиотеки удаленного управления и графического интерфейса пользователя (GUI) для управления роботом. Библиотека поддерживает работу с робототехнической операционной системой ROS. В статье описывается состав и взаимодействие модулей разработанного ПО, а также подробно описывается архитектура библиотеки удаленного управления. Для корректного управления роботом из фреймворка ROS, команды для управления роботом должны подаваться в системе СИ. Для этого был проведен ряд экспериментов по вычислению параметров преобразования линейных и угловых скоростей. ROS-программа позволяет удаленно использовать робота, что планируется задействовать в режиме автономного планирования маршрута робота и его использования в задачах автономного одновременного картографирования и локализации (SLAM). Был создан ROS-пакет, позволяющий независимо захватывать видео с четырех разных камер робота «Сервосила Инженер». Конфигурация и получение исходных данных с камер осуществляется при помощи библиотеки V4L2 (Video for Linux 2). Метод проецирования буферов устройства в память приложения повысил производительность, устранив излишние копирования. В статье демонстрируются результаты сравнения разработанного пакета с существующим пакетом, основанном на библиотеке OpenCV. При этом созданный пакет работает с различными форматами изображений и не зависит от OpenCV. По результатам экспериментальных сравнений программных библиотек при различной нагрузке были сделаны выводы о существенном снижении нагрузки на CPU при использовании разработанного пакета.

Программное обеспечение; передача видеопотока; V4L2; робототехническая операционная система ROS; OpenCV; мобильный робот; гусеничный робот; API.

R.O. Lavrenov, I.A. Mavrin, R.N. Safin, E.A. Magid

ROBOT SERVOSILA ENGINEER: DEVELOPMENT OF VIDEO STREAMING SERVER AND CONTROL SYSTEM GUI FOR ROS FRAMEWORK

This paper presents the development of software for control and video streaming from the cameras of Russian crawler robot Servosila Engineer. The technical characteristics of the robot, the characteristics of its operating system, the built-in API and software of the robot, a data protocol for controlling the robot and its telemetry are presented in the article. A user can control the robot without using the joystick with the developed software. The developed software consists of a remote control library and a graphical user interface (GUI) for the robot control. This library supports work with robot operating system ROS. The paper describes the developed software modules and their interactions and describes in detail the architecture of the remote control library. For correct control of the robot from the ROS framework, commands for controlling the robot should be submitted to the SI system. There are a series of experiments of calculating the transformation parameters of the linear and angular velocities. ROS-program allows you to remotely use the robot, which is planned to be used in the autonomous route-planning mode of the robot and its use in tasks of autonomous simultaneous localization and mapping (SLAM). ROS

package was created that allows to independently capture video from 4 different cameras of the robot. The configuration and acquisition of the initial data from the cameras is carried out using V4L2 (Video for Linux 2) library. The method of projecting device buffers into application memory increased performance by eliminating unnecessary copying. Created package is compared with an existing package based on OpenCV. The created package works with various image formats and does not depend on OpenCV. Based on the results of experimental comparisons of software libraries under different operating conditions, conclusions are drawn about a significant reduction in CPU load when using the developed package.

Software; video streaming; V4L2; ROS; OpenCV; mobile robot; crawler robot; API.

Введение. В настоящее время мобильные роботы все чаще используются для решения широкого спектра задач – от социального взаимодействия с человеком [1] до опасных поисково-спасательных операций [2]. Современные мобильные роботы могут работать внутри помещений и на открытой местности, но требуют эффективной системы автономной навигации в неизвестном окружении, где невозможно использование систем глобального позиционирования ГЛОНАСС, GPS и других [3]. Также роботу требуется большая вычислительная мощность для осуществления работы алгоритма одновременной локализации и картографирования (SLAM) [4], возможность передачи информации между роботами [5] и навыки командного взаимодействия с другими роботами [6]. Одной из ключевых функций наземных гусеничных роботов является их использование в поисково-спасательных операциях в городских условиях [7], особенно в ситуациях, когда поисково-спасательная операция происходит в опасных условиях и имеется потенциальная угроза жизни и здоровью спасателей (например, осуществление поиска пострадавших в зданиях с опасностью обрушения).

В данной работе в качестве поисково-спасательного робота будет рассмотрен российский гусеничный робот «Сервосила Инженер». Он оборудован четырьмя камерами и имеет клиент-серверный программный интерфейс для управления роботом, позволяющий производить последовательное переключение между всеми камерами в предоставляемом графическом интерфейсе пользователя (GUI); однако возможность одновременного получения видео данных со всех четырех камер отсутствует. В состав встроенного программного обеспечения (ПО) робота входит GUI, который позволяет работать только в простом режиме телеоперации с использованием джойстика и информации с одной камеры робота или 3D-очков. Поэтому наша задача заключалась в разработке ПО с возможностью интеграции с ROS для управления роботом без использования джойстика. В ROS реализовано множество алгоритмов локализации, поиска маршрута, распознания объектов, SLAM и других алгоритмов. Именно поэтому библиотека для удаленного управления должна иметь возможность управления через ROS, а не только через GUI [8].

Захват видеопотока с камер робота является важной задачей для многих целей, например, для осуществления визуального SLAM [15], планирования маршрута [13], человекомашинного взаимодействия [11], телеоперации в городских поисково-спасательных работах. В некоторых ситуациях необходимо получать и обрабатывать видеоданные в режиме реального времени. Более того, в то время, как одна камера может использоваться в моноSLAM-алгоритмах [15] для беспилотных летательных аппаратов (БЛА) и обычных роботов, в более сложных робототехнических системах могут использоваться несколько камер, например, стереопара, для реализации алгоритмов стереоSLAM. Перечисленные алгоритмы могут потребовать больших вычислительных ресурсов, тем самым мобильный робот будет нуждаться в передаче сенсорных данных на более мощное вычислительное устройство для обработки и анализа данных.

Поэтому был создан ROS-пакет, позволяющий одновременно захватывать видеопоток с четырех камер. Следующей целью является передача видеопотока через беспроводные сети в режиме реального времени и реализация RTP-сервера, что позволит клиентам получать видеоданные с робота и улучшит процесс телеоперации, так как оператор сможет получать больше информации об окружающей среде.

Физические характеристики робота «Сервосила Инженер». Гусеничный робот «Сервосила Инженер» спроектирован и произведен российской компанией «Сервосила» для работы в сложных условиях местности (например, исследование туннелей и трубопроводов, днища автомобилей и т.д.) [19]. Робот «Сервосила Инженер» (рис. 1) оснащен набором бортовых датчиков, который включает в себя камеру с оптическим зумом и три широкоугольные камеры, укомплектован фонарем для работы в условиях недостаточной освещенности и манипулятором для захвата и перемещения потенциально опасных объектов и открывания дверей в режиме телеоперации. Три из четырех камер на голове робота расположены в передней части и одна камера – на задней: передняя камера с оптическим увеличением, передняя пара широкоугольных камер (стереопара), широкоугольная камера заднего вида. В роли бортового вычислителя выступает двухъядерный (физические ядра) процессор Intel® Core ™ i7-3517UE (1.70ГГц). На робота предустановлена операционная система Ubuntu 14.04 LTS (Trusty Tahr).

Рис. 1. Мобильный робот Сервосила Инженер

API робота и встроенная программа удаленного управления. Поставляемое с роботом «Сервосила Инженер» программное обеспечение состоит из двух частей:

1. Серверное ПО. Это программа, запускаемая на роботе при его старте. Она получает команды от клиентской программы и отправляет на неё в ответ телеметрию робота.

2. Клиентское ПО (Operator Control Unit - OCU). Эта программа устанавливается на компьютер пользователя и отправляет команды на сервер. Команды передаются пакетами (табл. 1). Для управления роботом необходим джойстик XBox. Перед каждым использованием джойстик должен быть откалиброван. Программа отображает изображение с одной камеры и текущее схематическое изображение робота в углу окна.

Таблица 1

Пакет данных для управления роботом

Поле	Frame type ID	Axis #0	…	Axis #15	Button #0	…	Button #15	Video bit rate telemetry	Общий размер
Размер (байт)	1	2	…	2	1	…	1	8	57
Тип	uint8	int16	…	int16	uint8	…	uint8	double	

Клиентская программа отправляет пакеты данных на сервер пять раз в секунду. Эти пакеты содержат информацию о кнопках и джойстиках контроллера Xbox 360. Робот отправляет в OCU пакеты телеметрии (табл. 2), т.е. информацию о камерах и сервоприводах, один раз в секунду. Информация от каждого двигателя представляет из себя пакет данных, подробно описанный в табл. 3. Также этот протокол имеет возможность отправлять видеокадры с одной камеры робота с задержкой в 1–2 секунды.

Следует отметить, что протокол имеет следующие недостатки:

- Сильная зависимость протокола от игрового контроллера. Поэтому любое движение, которое невозможно сделать с контроллером на графическом интерфейсе от компании «Сервосила», невозможно повторить и с созданным графическим интерфейсом. Например, одно шасси робота невозможно задействовать отдельно от другого как с базового, так и с графического интерфейса.
- Протокол позволяет передавать только значения скорости на сервоприводы.
- Отсутствует опция проверки уровня заряда аккумулятора с помощью протокола. По этой причине оператор должен постоянно подключать дисплей к встроенному компьютеру робота, чтобы проверять уровень зарядки аккумулятора.
- Отсутствует опция вращения первой оси манипулятора одновременно с шасси робота. Таким образом, невозможно вращать эту ось, пока робот находится в движении. Робот должен сначала остановиться.
- Отсутствует опция проверки статуса фонаря робота. Оператор должен проверить визуально по камерам, включен ли фонарь.
- Отсутствует опция поворота захвата робота.

Таким образом, из-за особенностей протокола, написанная библиотека, эмулирует взаимодействие с игровым контроллером.

Таблица 2

Пакет телеметрии

Поле	Frame type ID	Tick number	Number of motors	Motor data #0	…	Motor data #9	Not used	Общий размер
Размер (байт)	1	8	1	24	…	24	25	275
Тип	uint8	uint64	uint8	struct	…	struct	-	

Таблица 3

Данные с каждого двигателя

Поле	Размер	Тип
Device ID	1 byte	uint8
Device state	1 byte	uint8
Operation mode	1 byte	uint8
Position	4 bytes	uint32
Speed	2 bytes	int16
Amps	2 bytes	int16
Status bits	2 bytes	int16
Position command	4 bytes	uint32
Speed command	2 bytes	int16
Amps command	2 bytes	int16

Разработка программного обеспечения для управления роботом. Графический интерфейс робота от компании “Сервосила” обеспечивает только базовые функции. Оператор может управлять роботом в реальном времени только с помощью игрового контроллера Xbox 360. По этой причине было решено разработать новую графическую оболочку и библиотеку удаленного управления с расширенными функциональными возможностями. Также разработанное программное обеспечение должно обеспечивать управление роботом из Фреймворка ROS. Поэтому принято решение разделить программное обеспечение на три части (рис. 2).

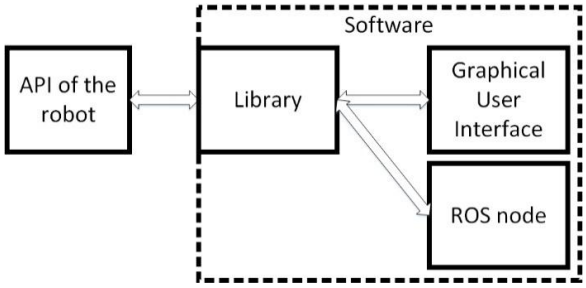

Рис. 2. Схема созданного программного обеспечения

Библиотека удаленного управления является посредником между графическим интерфейсом и программой ROS на компьютере пользователя и роботом. Она используется для отправки команд роботу, а также получения информации с камер и сервоприводов. Библиотека представляет собой набор функций (API), используемых для управления роботом. Робот управляется через Wi-Fi или радиосвязь. Команды инкапсулированы протоколом дистанционного управления, разработанным компанией «Сервосила» (табл. 1).

Библиотека удаленного управления реализована как динамически-подключаемый файл (* .so-файл). Такой подход дает возможность использовать библиотеку в разных приложениях. Например, программа с алгоритмом автономного поиска пути для робота может быть реализована с использованием этой библиотеки удаленного управления.

Созданная библиотека содержит следующие объекты (рис. 3):

- Структура RobotConfiguration содержит набор максимальных скоростей для сочленений робота и другую информацию о нём.
- Класс Robot служит оберткой контроллера и отправляет скорости в методы контроллера.
- Класс RobotController запускает отдельный поток для класса UDPClient и управляет роботом, изменяя значения скоростей в пакете удаленного управления.
- Класс RobotPositionController расширяет RobotController, он может управлять суставами по положению.
- Структура RobotSI преобразует команды в систему SI (Международная система единиц), поэтому может обрабатывать значения скорости в м/с и рад/с.
- Структура RobotPackets содержит реализацию пакетов из протокола API.
- Класс UDPClient представляет собой службу отправки данных на робота.

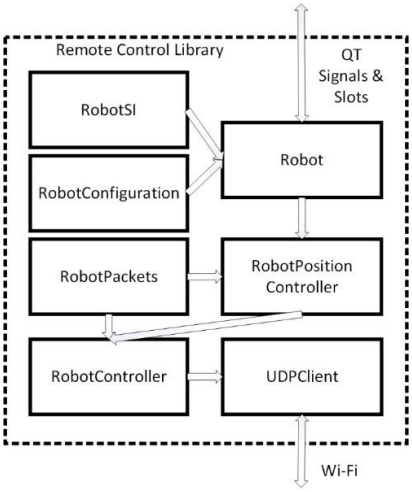

Рис. 3. Схема библиотеки внешнего управления

Библиотека была построена с применением qmake и g++ с использованием фреймворка Qt [9]. Библиотека работает в своем потоке и соединяется с графическим интерфейсом и программой ROS с использованием сигналов Qt и слотов. Это решение позволило довольно быстро разработать ПО управления роботом, но необходимо подчеркнуть некоторые недостатки этого решения. Прежде всего, каждое приложение, которое планирует использовать библиотеку, должно включать библиотеки Qt для поддержки механизмов сигналов и слотов. По этой причине приложение должно включать библиотеки Qt в свой двоичный файл (что резко увеличивает размер двоичного файла) или динамически подключать библиотеки Qt (что в свою очередь устанавливает ограничения на OCU). Поэтому в будущем у нас есть планы переписать библиотеку без задействования механизмов Qt на языке C++ без использования каких-либо библиотек, фреймворков или других зависимостей.

Класс UDPClient в библиотеке инкапсулирует статический объект для реализации UDP-сокета. UDPClient работает в своем потоке и отправляет пакет по таймеру пять раз в секунду. Операции с GUI (или команды из программы ROS) изменяют пакет.

Пакеты телеметрии (см. табл. 2) несут в себе следующую информацию:

- Состояние двигателя.
- Положение вала двигателя.

- Частота вращения вала двигателя.
- Сила тока на двигателе.
- Скорость.
- Флаг обнаружения ошибки.
- Счетчик ошибок.

Пакет телеметрии содержит информацию о каждом двигателе. Информация о положении вала двигателя используется в управлении положением. Основная проблема заключается в том, что двигатели шасси не имеют информации о телеметрии и, таким образом, единственный способ расчета расстояния – это произведение скорости и времени движения. Это решение существенно менее точно, чем использование энкодеров.

Значение скорости в протоколе является целым числом в диапазоне от –32768 до +32767. Причиной такого диапазона является кодирование скорости как позиции управляющих джойстиков игрового контроллера Xbox 360. Библиотека удаленного управления должна иметь возможность принимать значения скорости в м/с, что необходимо для удобства пользователя и для ROS-программы. Чтобы идентифицировать соответствие между абстрактными командами API и движением в реальном мире, были проведены эксперименты для прямого и вращательного движений робота.

Основная идея эксперимента заключается в следующем: чтобы преобразовать числа API в единицы измерения системы СИ, следует сопоставить скорость вращения треков в зависимости от команд API, отправленных роботу. Корреляция между этими значениями должна быть выражена как линейное соотношение между значениями команд API и реальной скоростью в метрах в секунду.

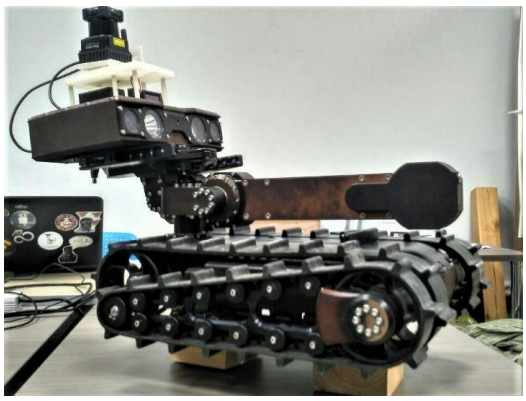

Рис. 4. Робот в фиксированной позиции при проведении экспериментов

План эксперимента:

1. Установим робот таким образом, чтобы ничто не влияло на скорость вращения гусениц (включая трение).

2. Измеряем длину гусениц.

3. Отправляем роботу команды постоянной скорости.

4. Выбираем относительно большое количество N полных оборотов дорожек для отслеживания.

5. Измеряем время t, необходимое для того, чтобы гусеницы полностью прокрутились N раз.

6. Вычисляем реальную скорость в метрах в секунду по уравнению: $vr = l*N/t$.

Таблица 4

Эксперименты по поиску коэффициента преобразования линейной скорости

Скорость	Обороты	Длина гусеницы	Время замера (с)	Скорость (м/с)
6000	40	1,15	552	0,08333333333
9000	40	1,15	368	0,125
12000	40	1,15	276	0,1666666667
15000	40	1,15	221	0,2081447964
18000	40	1,15	184	0,25
21000	40	1,15	158	0,2911392405
24000	40	1,15	138	0,3333333333
27000	40	1,15	122	0,3770491803
30000	40	1,15	111	0,4144144144

Робот был установлен на двух блоках (см. рис. 4) так, чтобы гусеницы робота не касались блоков во время движения. На основном корпусе робота и на гусенице были помещены метки, чтобы следить за количеством оборотов. При совпадении меток фиксируется оборот гусеницы. Длина гусеницы l составляет 1,15 метра. Число оборотов N для отслеживания времени было определено как 40 оборотов.

Результаты экспериментов показали, что целые значения скорости, которые отправляются роботу, прямо пропорциональны (линейны) полученной скорости прямолинейного движения и вращения. Были вычислены эти два коэффициента путем преобразования целочисленных значений в систему СИ. Для линейного значения движения lv = 72021,73913 и для вращательного движения lr = 24000. Чтобы получить скорость в м/с и рад/с, целочисленные значения делились на эти константы. Результаты экспериментов представлены в табл. 4–7.

Таблица 5

Эксперименты по поиску коэффициента преобразования угловой скорости

Скорость	Обороты	Длина гусеницы	Время замера (с)	Линейная скорость (м/с)	Радиус окружности поворота на месте	Угловая скорость (рад/с)
10000	10	1,15	276	0,0416667	0,1	0,4166666667
15000	10	1,15	184	0,0625	0,1	0,625
20000	10	1,15	138	0,0833333	0,1	0,8333333333
25000	10	1,15	110	0,1045454	0,1	1,045454545
30000	10	1,15	92	0,125	0,1	1,25

Таблица 6

Таблица расчета коэффициента преобразования линейной скорости

Скорость	Коэффициент	Средний коэфициент	Погрешность коэффициента (относительная)	Погрешность коэффициента (абсолютная)
6000	72000	72021,73913	0,0268115942	1930,434783
9000	72000		0,02717391304	1956,521739
12000	72000		0,02753623188	1982,608696
15000	72065,21739		0,02789691127	2010,396975
18000	72000		0,02826086957	2034,782609
21000	72130,43478		0,02861860209	2064,272212
24000	72000		0,02898550725	2086,956522
27000	71608,69565		0,02936564505	2102,835539
30000	72391,30435		0,02969056013	2149,338374

Таблица 7

Таблица расчета коэффициента преобразования угловой скорости

Скорость	Угловая скорость (рад/с)	Коэффициент	Погрешность коэффициента (относительная)	Погрешность коэффициента (абсолютная)
10000	0,4166666667	24000	0,02753623188	660,8695652
15000	0,625	24000	0,02826086957	678,2608696
20000	0,8333333333	24000	0,02898550725	695,6521739
25000	1,045454545	23913,04348	0,02972332016	710,7750473
30000	1,25	24000	0,03043478261	730,4347826

Анализ полученных данных позволил определить коэффициенты корреляции, представленные в табл. 6, 7. Эти коэффициенты используются далее для преобразования API-команд в реальные скорости. Полученные результаты были записаны в классе RobotSI.

Создание ROS-пакета по управлению роботом на основе написанной библиотеки. Робот «Сервосила Инженер» будет использоваться для разработки алгоритмов автономной локализации, картографирования и планирования пути [20]. Для этого необходим модуль (программа) фреймворка ROS для управления роботом. В программах фреймворка ROS используются команды перемещения в формате \geometry_msgs\Twist, который содержит линейные (по всем осям) и угловые скорости (также по всем осям), измеренные в метрах и радианах соответственно. Поэтому здесь нужна функция преобразования скорости, записанная в удаленной библиотеке.

Разработанная библиотека управления включается при сборке в программу ROS. Программа принимает команды ROS из потока \move_base и отправляет команды роботу через соединение Wi-Fi. Для этого библиотека была статически

скомпилирована и ее заголовки были включены в исходный файл программы ROS. Программа ROS – это консольная программа. Рабочий процесс происходит в цикле следующим образом:

1. Ждем сообщений в указанном именованном потоке данных (например, \cmd_vel\).

2. Когда приходит сообщение, оно разделяется на две части: линейную и угловую скорости. Если линейная часть не равна нулю, то: если линейная часть больше или равна по модулю 0,2 м/с – она отправляется роботу; иначе – отправляется линейная скорость 0,2 м/с. Если угловая часть не равна нулю, то: если угловая часть по модулю больше или равна 0,2 рад/с – она отправляется роботу; иначе отправляется угловая скорость 0,2 рад/с. Это было сделано потому, что существуют текущие ограничения для двигателей робота, и робот не двигается с низкой скоростью.

Когда программа закрывается, то посылает ноль в качестве угловой и линейной скоростей, чтобы остановить робота и предотвратить неконтролируемое движение.

Низкоуровневый захват видео с камер робота. Для захвата видео с камер была использована вторая версия библиотеки Video4Linux. Данная библиотека была предустановлена на оригинальной системе создателями робота. Программирование V4L2 устройства состоит из следующих шагов [18]:

- Открытие устройства.
- Конфигурация параметров устройства.
- Выбор видеостандарта.
- Конфигурация формата видеоданных.
- Конфигурация методов ввода-вывода.
- Цикл чтения/записи.
- Закрытие устройства.

Параметры видеоустройства настраиваются путем вызова специальных запросов (ioctl), представляющих из себя системный вызов, задающий параметры устройства (количество кадров в секунду, разрешение кадра). Процесс работы V4L2 программы (рис. 5) может меняться в зависимости от ситуации. Например, реализация не отображает получаемые видеоданные. Первоначальным шагом является открытие устройства для последующей настройки его параметров для потокового захвата видео. Для того чтобы использовать системные вызовы ioctl, файловый дескриптор устройства (камеры) должен быть открыт на операции чтения и записи.

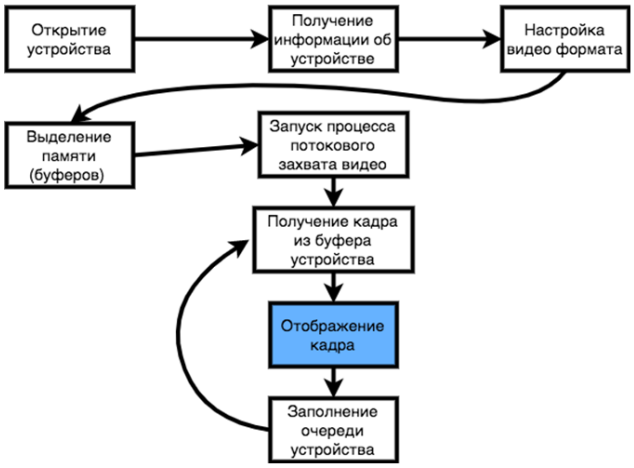

Рис. 5. Принцип работы V4L2-приложения

Необходимо также настроить формат получаемых данных с камеры. «Сервосила Инженер» имеет три камеры (стереопара и камера заднего вида), которые могут выдавать кадры в формате шаблона Байера [16]. Кадры с передней камеры с оптическим увеличением представлены в цветовом пространстве yuv (yuyv422).

После настройки формата кадров видеопотока, выделяется память под буферы устройства. Буфер состоит из изображения с камеры, полученного путем обмена данными приложением и драйвером устройства. Для управления процессом обмена данными между приложением и устройством используется метод проецирования памяти устройства в память приложения (mmap). Это позволяет увеличить производительность захвата видео с камеры, избавляясь от ненужного копирования данных. Приложение и драйвер обмениваются указателями на данные, которые становятся доступными приложению.

Далее осуществляется запрос на получение информации о выделенных буферах. Эта информация включает в себя сведения о расположении буферов в памяти (указатели) и их количестве (может отличаться от запрашиваемого размера). Процесс получения кадров заключается в двух этапах (рис. 6):

1) этап заполнения очереди пустым буфером;

2) этап получения буфера с данными.

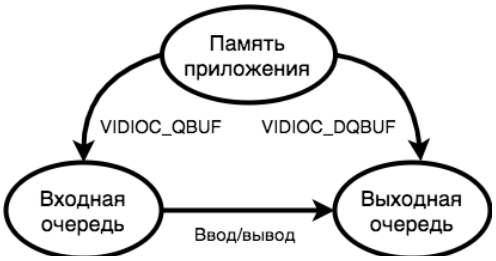

Рис. 6. Процесс получения кадров V4L2

Для каждого этапа вызывается специальный запрос (VIDIOC_QBUF и VIDIOC_DQBUF). На первом этапе буфер вставляется в очередь драйвера устройства. Буфер ожидает момента, когда драйвер устройства заполнит его данными. На втором этапе обработанный буфер (с видеоданными) извлекается из очереди [17].

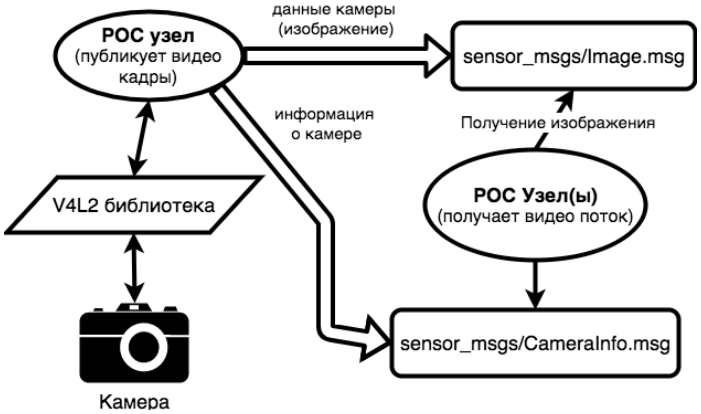

Рис. 7. Архитектура создаваемого ROS-пакета

Реализация ROS-пакета для публикации с камер робота. Изначально в системе робота не был установлен фреймворк ROS, поэтому был установлен совместимый с Ubuntu 14.04 дистрибутив ROS Indigo. Архитектура ROS-пакета состоит из следующих главных частей (см. рис. 7):

- Узел, независимо публикующий данные с каждой камеры робота.
- Прослойка V4L2, через которую осуществляется захват видеокадров с камер робота.

```
robot@robot-computer:~$ pidstat 10 -p 9402
Linux 4.4.0-83-generic (robot-computer)         02.11.2017      _x86_64_        (4 CPU)

11:12:07      UID       PID    %usr %system  %guest    %CPU   CPU  Command
11:12:17     1000      9402    0,00    0,40    0,00    0,40     0  video_stream
11:12:27     1000      9402    0,00    0,40    0,00    0,40     1  video_stream
11:12:37     1000      9402    0,00    0,30    0,00    0,30     2  video_stream
11:12:47     1000      9402    0,00    0,40    0,00    0,40     0  video_stream
11:12:57     1000      9402    0,00    0,40    0,00    0,40     0  video_stream
11:13:07     1000      9402    0,00    0,40    0,00    0,40     2  video_stream
11:13:17     1000      9402    0,00    0,50    0,00    0,50     1  video_stream
robot@robot-computer:~$ pidstat 10 -p 9402
Linux 4.4.0-83-generic (robot-computer)         02.11.2017      _x86_64_        (4 CPU)

11:18:29      UID       PID    %usr %system  %guest    %CPU   CPU  Command
11:18:39     1000      9402    0,00    3,90    0,00    3,90     1  video_stream
11:18:49     1000      9402    0,00    4,70    0,00    4,70     3  video_stream
11:18:59     1000      9402    0,00    4,20    0,00    4,20     3  video_stream
11:19:09     1000      9402    0,00    4,40    0,00    4,40     0  video_stream
11:19:19     1000      9402    0,00    4,40    0,00    4,40     1  video_stream
```

Рис. 8. Использование процессора созданным ROS-пакетом (замеры проведены утилитой pidstat в режимах простоя и нагрузки)

Узел, публикующий данные с камеры, транслирует видеопоток в созданные топики. Любой другой узел, подписанный на созданный топик с видеоданными, может получать данные, обрабатывать их или ретранслировать. Используются следующие параметры публикующего узла:

- Количество кадров в секунду задает количество видеокадров, публикуемых в топики каждую секунду.
- Разрешение кадра: высота и ширина видеокадра.
- Формат кадра ROS (напр., bayer_grbg8).
- Видеоустройство (напр., /dev/video0).

Изображение с камер искажено и требует калибровки камер, что является частью планов на будущее.

В процессе разработки пришлось столкнуться с несколькими проблемами. Наиболее серьезной из них являлась проблема получения некорректного времени захвата кадра. Данный параметр является важным для реализации стереоSLAM-алгоритмов для синхронизации двух камер по времени. В V4L2 есть возможность получить время заполнения буфера данными с камеры. Однако данный параметр был в некорректном состоянии, и решение данной проблемы было найдено в GitHub-репозитории библиотеки OpenCV. Данные о времени захвата кадра должны быть извлечены до того, как вновь размещать буфер с данными в очередь драйвера устройства. Все последующие тесты были проведены в системе мобильного робота «Сервосила Инженер».

Стоит отметить, что при отсутствии подписчиков на топики с видеоданными, использование ресурсов процессора минимально (0,4 %). При наличии подписчиков созданный пакет использует 4,4 % ресурсов процессора. Никаких дополнительных действий по конвертации изображения в другие форматы, таких как RGB, не осуществляется. Таким образом, получится снизить нагрузку на процессор мобильного робота, не затрачивая ресурсы на дополнительную обработку данных. Разработанный пакет никак не зависит от библиотеки OpenCV.

Были проанализированы другие ROS-пакеты потокового видеозахвата. Наиболее распространенным оказался пакет, основанный на библиотеке OpenCV [14]. В ходе тестирования пакет потреблял меньше системных ресурсов в отличие от пакета, базирующегося на библиотеке OpenCV (рис. 9). Если в потоковый захват видео вовлечено больше камер, то нагрузка на процессор будет расти. В режиме простоя, при отсутствии клиентов, пакет на OpenCV продолжает активно использовать ресурсы системы.

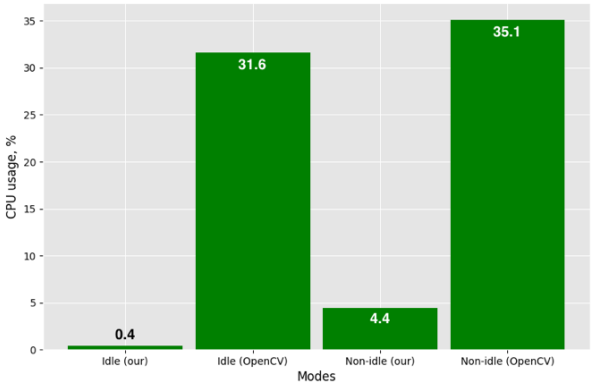

Рис. 9. Среднее использование ресурсов процессора созданным и OpenCV ROS-пакетом (в режимах простоя и нагрузки)

Заключение. В данной статье представлена разработка программного обеспечения (ПО) для управления и передачи видеопотока с камер российского гусеничного робота «Сервосила Инженер». Описаны узлы робота, особенности встроенного API и ПО, и обоснованы причины, которые сделали необходимой разработку собственного ПО. Разработанное ПО состоит из библиотеки удаленного управления и графического интерфейса пользователя (GUI) для управления роботом. Библиотека поддерживает работу с робототехнической операционной системой ROS. В статье описан состав и взаимодействие модулей разработанного ПО. При помощи разработанного ПО пользователь может управлять роботом, не используя джойстик. ROS-программа позволяет удаленно использовать робота, что планируется задействовать в режиме автономного планирования маршрута робота на основе статического баланса [10].

Созданный ROS-пакет позволяет независимо захватывать видео с 4 разных камер робота «Сервосила Инженер». Конфигурация и получение исходных данных с камер осуществляется при помощи библиотеки V4L2. Метод проецирования буферов устройства в память приложения повысил производительность, устранив излишние копирования. Ключевым моментом является не только получение кадров с камеры, но и уменьшение количества потребляемых ресурсов и нагрузки на процессор. Сравнение разработанного пакета с существующим пакетом, основанном на OpenCV, показало существенное превосходство предложенного в данной работе решения.

В будущем планируется реализовать RTP-сервер, который позволит передавать видео с робота в режиме реального времени через беспроводную сеть. Исходный код доступен для публичного использования в репозитории на сайте системы контроля версий GitHub [16].

Работа выполнена в рамках реализации проекта «Локализация, картографирование и поиск пути для беспилотного наземного робота (БНР) при помощи группы беспилотных летательных аппаратов (БПЛА) с использованием активного

коллективного технического зрения и планированием в общем доверительном пространстве группы роботов» при поддержке РФФИ 15-57-06010. Часть работ выполнена в соответствии с Государственной программой конкурентоспособности Российской Федерации при Казанском федеральном университете.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Budkov V.Y., Prischepa M.V., Ronzhin A.L., and Karpov A.A.* Multimodal human-robot interaction // In Ultra Modern Telecommunications and Control Systems and Workshops, Int. Congress Ultra Modern Telecommunications, 2010. – P. 485-488.
2. *Birk A., Schwertfeger S., and Pathak K.* A networking framework for teleoperation in safety, security, and rescue robotics // IEEE Wireless Communications. 16.1, 2009: 6-13.
3. *Yakovlev K., Khithov V., Loginov M., and Petrov A.* Distributed Control and Navigation System for Quadrotor UAVs in GPS-Denied Environments // IEEE Int. Conf. on Intelligent Systems, Springer Int. Publishing, 2015: 49-56.
4. *Buyva A., Afanasyev I., and Magid E.* Comparative analysis of ROS-based Monocular SLAM methods for indoor navigation // Ninth International Conference on Machine Vision. – International Society for Optics and Photonics (2017). – P. 103411K-103411K-6.
5. *Li B., Ma S., Liu T., and Liu J.* Cooperative negotiation and control strategy of a shape-shifting robot // IEEE Int. Symp. on Safety, Security and Rescue Robotics, 2008: 53-57.
6. *Michael N., Shen S., Mohta K., Mulgaonkar Y., Kumar V., Nagatani K., Okada Y., Kiribayashi S., Otake K., Yoshiba K., Ohno K., Takeuchi E., and Tadokoro S.* Collaborative mapping of an earthquake-damaged building via ground and aerial robots // Journal of Field Robotics, 2012: 832-841.
7. *Sokolov M., Lavrenov R., Gabdullin A., Afanasyev I., and Magid E.* 3D modelling and simulation of a crawler robot in ROS/Gazebo // Int. Conf. on Control, Mechatronics and Automation, 2016: 61-65.
8. *Alishev N., Lavrenov R., and Gerasimov Y.* Russian mobile robot Servosila Engineer: designing an optimal integration of an extra laser range finder for SLAM purposes // Int. Conf. on Artificial Life and Robotics, 2018: 204-207.
9. Qt framework, https://www.qt.io/.
10. *Magid E., Tsubouchi T., Koyanagi E., and Yoshida T.* Static Balance for Rescue Robot Navigation: Losing Balance on Purpose within Random Step Environment // Int. Conf. on Intelligent Robots and Systems, 2010: 349-356.
11. *Karpov A., Carbini S., Ronzhin A., and Viallet J.* Two similar different speech and gestures multimodal interfaces. Multimodal User Interfaces. (Berlin, Heidelberg, 2008. – P. 155-184.
12. *Lukin A. and Kubasov D.* High-quality algorithm for Bayer pattern interpolation // Programming and Computer Software 30.6 (2004). – P. 347-358.
13. *Magid E., Lavrenov R., and Khasianov A.* Modified spline-based path planning for autonomous ground vehicle. Int. Conf. on Informatics in Control, Automation and Robotics (Madrid, Spain, 2017). – P. 132-141.
14. GitHub video_stream_opencv ROS package: https://github.com/ros-drivers/video_stream_opencv.
15. *Fuentes-Pacheco J., Ruiz-Ascencio J., and Rendón-Mancha J.M.* Visual simultaneous localization and mapping: a survey // Artificial Intelligence Review. – 2015. – Vol. 43 (1). – P. 55-81.
16. ROS-пакет одновременного потокового видеозахвата с мобильного робота; https://github.com/chupakabra1996/lirs_ros_video_streaming.
17. V4L2; https://linuxtv.org/downloads/v4l-dvb-apis/uapi/v4l/v4l2.html.
18. *Yinli L., Hongli Y., and Pengpeng Zh.*. The implementation of embedded image acquisition based on V4L2. In Electronics, Communications and Control (ICECC) // International Conference on IEEE, 2011. – P. 549-552.
19. *Sokolov M., Lavrenov R., Gabdullin A., Afanasyev I., and Magid E.* (2016, December). 3D modelling and simulation of a crawler robot in ROS/Gazebo // In Proceedings of the 4th International Conference on Control, Mechatronics and Automation. – P. 61-65.
20. *Ohno K., Morimura S., Tadokoro S., Koyanagi E., & Yoshida T.* (2007, October). Semi-autonomous control system of rescue crawler robot having flippers for getting over unknown-steps // In Intelligent Robots and Systems, 2007. IROS 2007. – P. 3012-3018.

REFERENCES

1. *Budkov V.Y., Prischepa M.V., Ronzhin A.L., and Karpov A.A.* Multimodal human-robot interaction, *In Ultra Modern Telecommunications and Control Systems and Workshops, Int. Congress Ultra Modern Telecommunications, 2010б* зз. 485-488.
2. *Birk A., Schwertfeger S., and Pathak K.* A networking framework for teleoperation in safety, security, and rescue roboticsб *IEEE Wireless Communications*. 16.1, 2009: 6-13.
3. *Yakovlev K., Khithov V., Loginov M., and Petrov A.* Distributed Control and Navigation System for Quadrotor UAVs in GPS-Denied Environments, *IEEE Int. Conf. on Intelligent Systems*, Springer Int. Publishing, 2015: 49-56.
4. *Buyva A., Afanasyev I., and Magid E.* Comparative analysis of ROS-based Monocular SLAM methods for indoor navigation, *Ninth International Conference on Machine Vision*. International Society for Optics and Photonics (2017), pp. 103411K-103411K-6.
5. *Li B., Ma S., Liu T., and Liu J.* Cooperative negotiation and control strategy of a shape-shifting robot, *IEEE Int. Symp. on Safety, Security and Rescue Robotics*, 2008: 53-57.
6. *Michael N., Shen S., Mohta K., Mulgaonkar Y., Kumar V., Nagatani K., Okada Y., Kiribayashi S., Otake K., Yoshiba K., Ohno K., Takeuchi E., and Tadokoro S.* Collaborative mapping of an earthquake-damaged building via ground and aerial robots, *Journal of Field Robotics*, 2012: 832-841.
7. *Sokolov M., Lavrenov R., Gabdullin A., Afanasyev I., and Magid E.* 3D modelling and simulation of a crawler robot in ROS/Gazebo, *Int. Conf. on Control, Mechatronics and Automation*, 2016: 61-65.
8. *Alishev N., Lavrenov R., and Gerasimov Y.* Russian mobile robot Servosila Engineer: designing an optimal integration of an extra laser range finder for SLAM purposes, *Int. Conf. on Artificial Life and Robotics*, 2018: 204-207.
9. Qt framework, https://www.qt.io/.
10. *Magid E., Tsubouchi T., Koyanagi E., and Yoshida T.* Static Balance for Rescue Robot Navigation: Losing Balance on Purpose within Random Step Environment, *Int. Conf. on Intelligent Robots and Systems*, 2010: 349-356.
11. *Karpov A., Carbini S., Ronzhin A., and Viallet J.* Two similar different speech and gestures multimodal interfaces. Multimodal User Interfaces. Berlin, Heidelberg, 2008, pp. 155-184.
12. *Lukin A. and Kubasov D.* High-quality algorithm for Bayer pattern interpolation, *Programming and Computer Software* 30.6 (2004), pp. 347-358.
13. *Magid E., Lavrenov R., and Khasianov A.* Modified spline-based path planning for autonomous ground vehicle. Int. Conf. on Informatics in Control, Automation and Robotics (Madrid, Spain, 2017). – P. 132-141.
14. GitHub video_stream_opencv ROS package: https://github.com/ros-drivers/video_stream_opencv.
15. *Fuentes-Pacheco J., Ruiz-Ascencio J., and Rendón-Mancha J.M.* Visual simultaneous localization and mapping: a survey, *Artificial Intelligence Review*, 2015, Vol. 43 (1), pp. 55-81.
16. ROS paket odnovremennogo potokovogo video zakhvata s mobil'nogo robota [ROS package simultaneous streaming video capture from mobile robot]. Available at: https://github.com/chupakabra1996/lirs_ros_video_streaming.
17. V4L2; https://linuxtv.org/downloads/v4l-dvb-apis/uapi/v4l/v4l2.html.
18. *Yinli L., Hongli Y., and Pengpeng Zh.*. The implementation of embedded image acquisition based on V4L2. In Electronics, Communications and Control (ICECC), *International Conference on IEEE*, 2011, pp. 549-552.
19. *Sokolov M., Lavrenov R., Gabdullin A., Afanasyev I., and Magid E.* (2016, December). 3D modelling and simulation of a crawler robot in ROS/Gazebo, *In Proceedings of the 4th International Conference on Control, Mechatronics and Automation*, pp. 61-65.
20. *Ohno K., Morimura S., Tadokoro S., Koyanagi E., & Yoshida T.* (2007, October). Semi-autonomous control system of rescue crawler robot having flippers for getting over unknown-steps, *In Intelligent Robots and Systems, 2007. IROS 2007*, pp. 3012-3018.

Статью рекомендовал к опубликованию д.т.н., профессор М.Ю. Медведев.

Магид Евгений Аркадьевич – Казанский федеральный университет; e-mail: magid@it.kfu.ru; г. Казань, ул. Кремлевская, 35; тел.: +78432213433; PhD; профессор; зав. кафедрой интеллектуальной робототехники.

Лавренов Роман Олегович – e-mail: lavrenov@it.kfu.ru; аспирант; м.н.с.

Маврин Илья Анатольевич – e-mail: i.a.mavrin@gmail.com; студент.

Сафин Рамиль Набиуллович – e-mail: safin.ramil@it.kfu.ru; лаборант-исследователь.

Magid Evgeni Arkad'evich – Kazan Federal University; e-mail: magid@it.kfu.ru; Kazan, 35, Kremlevskaya street; phone: +784322-3433; PhD; professor; head of Intelligent Robotics Department.

Lavrenov Roman Olegovich – e-mail: lavrenov@it.kfu.ru; postgraduate student; research associate.

Mavrin Ilya Anatol'evich – e-mail: i.a.mavrin@gmail.com; undergraduate student.

Safin Ramil Nabiulovich – e-mail: safin.ramil@it.kfu.ru; research assistant.