УДК 004.42 Научная статья

https://doi.org/10.32603/2071-8985-2023-16-7-39-46

Мобильный симулятор эхолота

А. А. Коротков, С. В. Букунов✉

Санкт-Петербургский государственный архитектурно-строительный университет, Санкт-Петербург, Россия

✉ sergeybukunov@yandex.ru

Аннотация. Описывается разработанное на кафедре информационных технологий Санкт-Петербургского архитектурно-строительного университета мобильное приложение, предназначенное для симулирования настоящего эхолота. Приложение может быть отнесено к классу программного обеспечения для бизнеса. Оно может применяться при тестировании систем, связанных с морской навигацией, а также для создания морских и речных карт. Приложение написано на языке C++ с использованием кросс-платформенной библиотеки Qt. Пользовательский интерфейс приложения разработан с помощью декларативного языка программирования QML. Реализация мобильной версии приложения позволила отказаться от использования дополнительных средств связи для обеспечения связи между симулятором и тестируемым устройством.

Ключевые слова: морская навигация, эхолот, кросс-платформенное программирование, библиотека Qt

Для цитирования: Коротков А. А., Букунов С. В. Мобильный симулятор эхолота // Изв. СПбГЭТУ «ЛЭТИ». 2023. Т. 16, № 7. С. 39–46. doi: 10.32603/2071-8985-2023-16-7-39-46.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Original article

Mobile Echo Sounder Simulator

A. A. Korotkov, S. V. Bukunov✉

Saint Petersburg State University of Architecture and Civil Engineering, Saint Petersburg, Russia

✉ sergeybukunov@yandex.ru

Abstract. A mobile application, designed to simulate real echo sounder, was developed at the Department of Information Technologies of the Saint Petersburg State University of Architecture and Civil Engineering (St. Petersburg, Russia). This application, potentially categorized as business software, can be used to test systems related to maritime navigation, as well as to create sea and river maps. The application is developed by C++ programming language using the cross-platform Qt library. The Graphic User Interface of the application is designed using the declarative programming language QML. The implementation of the mobile version of the application made it possible to refuse the use of additional communication devices to ensure communication between the simulator and the device under test.

Keywords: maritime navigation, echo sounder, cross-platform programming, Qt library

For citation: Korotkov A. A., Bukunov S. V. Mobile Echo Sounder Simulator // LETI Transactions on Electrical Engineering & Computer Science. 2023. Vol. 16, no. 7. P. 39–46. doi: 10.32603/2071-8985-2023-16-7-39-46.

Conflict of interest. The authors declare no conflicts of interest.

Введение. В настоящее время важность использования морских транспортных путей не вызывает сомнения. Огромное количество различных товаров ежегодно транспортируется именно на морских грузовых судах. Морская навигация традиционно относится к наиболее важным зада-

чам при осуществлении морских перевозок вообще и морских грузоперевозок в частности. Однако при этом важно не только провести судно по наиболее короткому маршруту в целях экономии финансовых затрат на перевозку груза, но и обеспечить максимальную безопасность самого процесса перевозки. Ведь если судно сядет на мель, то это принесет компании огромные убытки.

В качестве основного средства навигации традиционно используются эхолоты, которые измеряют расстояние от судна до морского дна и передают эти данные на судно. Информация, получаемая от эхолотов, в дальнейшем используется различными приборами и приложениями, управляющими работой этих приборов. От правильности работы эхолотов зависит безопасность судна, поэтому вопрос их тестирования представляется достаточно важным. Однако не всегда есть возможность тестировать приложения на реальных эхолотах. Возможным решением этой проблемы может служить использование различных симуляторов.

Эхолот представляет собой измерительный прибор, использующий звуковые импульсы для исследования структуры и рельефа дна, обнаружения подводных объектов. Эта информация очень важна для создания и обновления карт водных бассейнов, чтобы судоводители могли из карты знать примерную глубину в определенном районе. Также эхолот обычно устанавливают на судно для отслеживания расстояния до дна, чтобы судно не нашло на мель [1].

Отправленный датчиком эхолота импульс отражается от различных объектов, которые встречает на своем пути, или от дна. Это позволяет собрать информацию обо всем, что есть в толще воды: о количестве, размерах, плотности объектов, о структуре дна. Также эхолот обычно собирает и передает данные о температуры воды.

Работа эхолота заключается в том, чтобы после приема импульса отправить данные о расстоянии до дна на судно. Эти данные эхолот отправляет навигационному оборудованию на капитанском мостике по сети в виде NMEA-сообщений. Прибор на мостике, получающий данное сообщение, обрабатывает и визуализирует его.

NMEA (*англ*. National Marine Electronics Association) – это стандарт, который описывает протокол обмена тестовыми данными между различным оборудованием, связанным с морской навигацией. Данный стандарт стал популярен после того, как в навигации нашли применение различные GPS-приемники, использующие его [2].

NMEA-сообщение (или NMEA-сентенция) представляет собой сложное выражение, состоящее из следующих составных частей:

• символ «$» или «!»;

• две буквы, отвечающие за идентификатор источника сообщения – так называемый толкер;

• три буквы, отвечающие за идентификатор формата сообщения – так называемое название сентенции;

• данные, разделенные запятыми и состоящие из букв, цифр и точек для разделения в числах с плавающей запятой. В случае отсутствия данных в этой части сообщения ничего не пишется, но запятые все равно ставятся (т. е. возможен случай «,,»);

• символ «*»;

• восьмибитная XOR-сумма всех строк между «$» и «*», приведенная к двум ASCII-символам в верхнем регистре для шестнадцатеричного представления байта;

• комбинация <CR><LF> – признак окончания сентенции (по сути – это обычный перевод строки).

Примеры NMEA-сентенций эхолота:

• DPT (Depth) – глубина: $--DPT, x.x1, x.x2, x.x3*hh<CR><LF>, где x.x1 – глубина в метрах, x.x2 – поправка в метрах, x.x3 – максимальное значение шкалы. Сентенция DPT содержит значение глубины, измеренной от вибратора эхолота, и значение поправки на расположение вибратора. Положительная поправка означает расстояние от вибратора до поверхности воды, отрицательная – расстояние от вибратора до киля;

• DBT (Depth Below Transducer) – глубина, измеренная от вибратора эхолота: $--DBT, x.x1, f2, x.x3, M4, x.x5, F6*hh<CR><LF>, где x.x1 – глубина в футах; f2 – признак измерения футов; x.x3 – глубина в метрах; M4 – признак измерения в метрах; x.x5 – глубина в саженях; F6 – признак измерения в саженях;

• MTW (Water Temperature) – температура воды: $--MTW, x.x1, C2*hh<CR><LF>, где x.x1 – температура в градусах Цельсия; C2 – признак градусов Цельсия.

Из существующих в настоящее время симуляторов можно отметить приложение [3], которое изначально представляло собой расширение для браузера Google Chrome, но впоследствии было преобразовано в полноценное настольное приложение. Оно симулирует движение судна и ветра,

а также информацию о расстоянии до дна, температуре воды и посылает все данные в виде NMEA-сентенций через сеть, а именно через TCP/IP-соединение, всем подключенным устройствам. Однако реализация симулятора в виде настольного приложения существенно сужает возможности его практического использования.

Поскольку симулятор и устройство с приложением должны находиться в одной сети, то мобильное устройство представляет собой оптимальное решение, так как оно само может стать роутером или подключиться через провод. И тогда оба устройства будут входить в одну сеть и смогут через нее обмениваться сообщениями. Соответственно, в этом случае пропадает необходимость в каких-либо устройствах для соединения симулятора и целевого устройства в одну сеть, что значительно упрощает процесс тестирования.

Кроме того, для более точного управления сентенциями необходимо разделение DPT- и DBT-сентенций, а также добавление уровня шума, чтобы сентенции отличались друг от друга в определенном диапазоне. Это необходимо для большего соответствия реальным условиям и наведением за поступлением новых данных.

Постановка задачи. Цель настоящей статьи заключается в разработке мобильного приложения, представляющего собой симулятор эхолота и позволяющего проводить тестирование различных приложений, управляющих работой навигационного оборудования.

Требования к приложению. Приложение должно представлять собой мобильный симулятор эхолота и отвечать следующим требованиям:

– приложение должно быть мобильным;

– оно должно генерировать сентенции эхолота (DPT, DBT, MTW) и отправлять их по сети;

– возможность выбора генерируемых сентенций;

– все поля генерируемых сентенций должны быть редактируемыми;

– возможность отправки сентенций как с правильной, так и с неправильной контрольными суммами;

– возможность настройки частоты данных;

– в приложении должна отражаться информация о том, куда подключаться клиентам (IP-адрес, порт);

– возможность подключения и отключения нескольких клиентов;

– отображения последних отправленных сентенций, а также времени их отправки;

– отображения счетчика текущих подключенных клиентов;

– возможность поставить на паузу отправку сентенций, а затем запустить снова без потери подключенных клиентов.

Используемые технологии. Приложение разработано с помощью языка программирования C++ и библиотеки Qt. Это позволило обеспечить кросс-платформенность приложения, т. е. возможность его работы в устройствах, действующих под управлением различных операционных систем (Android, IOS и др.).

Qt представляет собой универсальную кроссплатформенную библиотеку с большим количеством разработанных классов и шаблонов для решения самых разнообразных задач [4]. Язык программирования C++ отлично подходит для разработки самостоятельных приложений, так как позволяет напрямую работать с памятью устройства, что в свою очередь открывает возможности для оптимизации приложения и улучшения его быстродействия [5].

Для создания интерфейса приложения использовался декларативный язык QML. Этот язык работает в связке с языком JavaScript и позволяет создавать пользовательские интерфейсы любой сложности как для настольных, так и для мобильных приложений [6].

Архитектура приложения. Архитектура разработанного приложения представлена на рис. 1.

Приложение состоит из нескольких модулей, основными из которых являются модули EchoSounderModel и EchoSounderServer. Модуль EchoSounderModel отвечает за связь с пользовательским интерфейсом и генерацией сентенций на основе того, что выбрал пользователь. Модуль EchoSounderServer отвечает за отправку сентенций подключенным пользователям.

Связь между этими двумя модулями заключается в обмене определенными данными. Сервер сообщает модели информацию об IP и порте, а также о количестве подключенных клиентов. Сервер получает от модели информацию о необходимости отправлять сентенции, частоте отправки, а также последние сгенерированные сентенции.

Программная реализация приложения. Приложение реализовано на языке C++ в объектно-

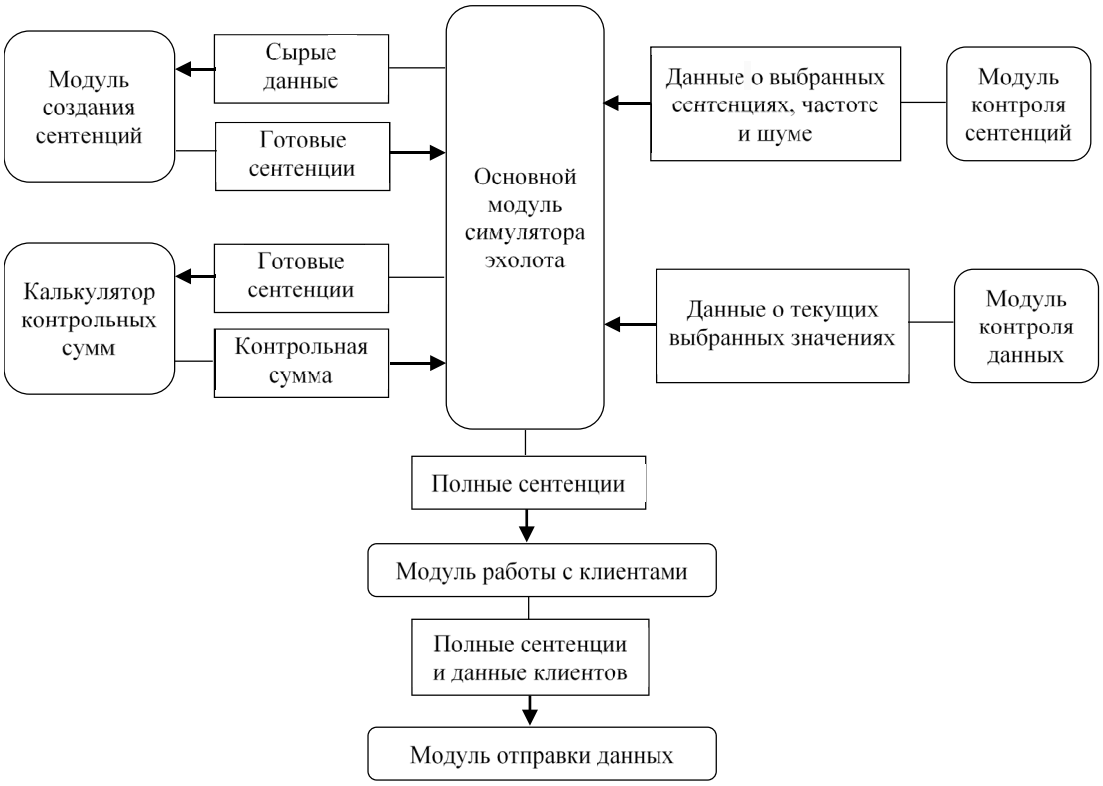

Рис. 1. Архитектура приложения
Fig. 1. Application architecture

ориентированном стиле. Каждый модуль приложения представляет собой соответствующий класс:

• EchoSounderModel – основная модель для взаимодействия с другими классами приложения; осуществляет сбор и обработку всей информации и отправляет ее на сервер;

• SentenceController – отвечает за связь со страницей настроек сентенций; собирает информацию о том, какие сентенции необходимо отправить, их необходимость в правильно посчитанной контрольной сумме, частоту отправки и уровень шума, а затем отправляет эти данные в модель;

• DataController – отвечает за связь со страницей управления данными в сентенциях; собирает информацию обо всех выбранных пользователем сырых значениях, а потом отправляет их в модель;

• NmeaParser – отвечает за составление полноценных NMEA-сентенций из сырых данных; может создавать DPT, DBT и MTW сентенции;

• CheckSumCalculator – отвечает за расчет контрольной суммы сентенции; в случае, если контрольная сумма должна быть посчитана верно, то считает ее; иначе вместо правильной контрольной суммы возвращает «XX»;

• EchoSounderServer – отвечает за серверную часть приложения; создает сервер и находится в режиме ожидания новых подключенных клиентов и хранит их, а также удаляет отключившихся клиентов;

• DataSender – отвечает за отправку NMEA-сентенций всем подключившимся клиентам.

Описание работы приложения на уровне взаимодействия между классами выглядит следующим образом.

На первом шаге SentnceController сообщает EchoSounderModel с какой частотой и какие данные необходимо отправлять. После этого DataController отправляет в EchoSounderModel сырые данные, которые хочет видеть пользователь. При этом оба этих класса реагируют на действия пользователя и при изменении каких-либо параметров сразу сообщают об этом EchoSounderModel.

EchoSounderModel с частотой, полученной от SentenceController, начинает обрабатывать данные, полученные от DataController. Он отправляет сырые данные в модуль NmeaParser, который обрабатывает полученные данные и возвращает готовые сентенции. Затем эти сентенции отправляются в CheckSumCalc, в котором при необходимости вычисляется контрольная сумма сентенций.

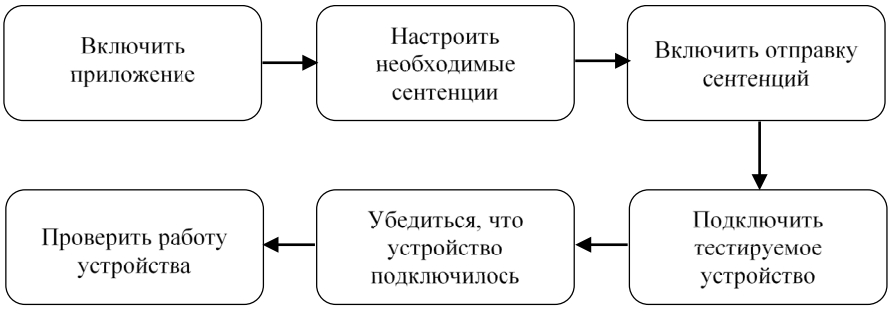

Рис. 2. Схема работы с симулятором эхолота
Fig. 2. The workflow with the echo sounder simulator

После этого полные сформированные сентенции отправляются в EchoSounderServer, который пересылает их в DataSender вместе со списком подключенных клиентов, а тот, в свою очередь, осуществляет отправку полученных сентенций клиентам.

Стандартный цикл работы с приложением представлен на рис. 2.

Результаты. Разработанное приложение представляет собой мобильный симулятор, состоящий из двух вкладок.

При включении пользователь видит первую вкладку с элементами управления, представленную на рис. 3.

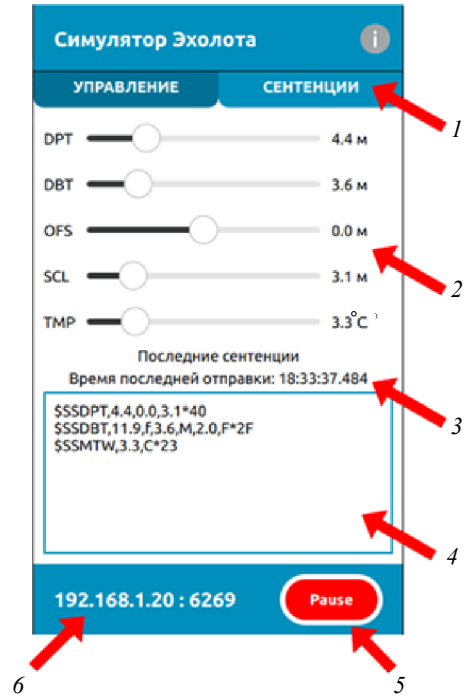

Рис. 3. Страница управления данными сентенций: *1* – вкладки для переключения между страницами приложения; *2* – элементы управления полями сентенций; *3* – время последней отправленной сентенции; *4* – список последних отправленных сентенций; *5* – кнопка включения/выключения симулятора; *6* – IP-адрес и порт для подключения
Fig. 3. Sentence data management page: *1* – tabs to switch between application pages; *2* – sentence fields controls; *3* – time of the last sentences sent; *4* – list of last sentences sent; *5* – simulator on/off button; *6* – IP and port to connect

На этой вкладке пользователь может управлять определенными полями сентенций: DPT отвечает за глубину для сентенции DPT; DBT – за глубину для сентенции DBT; OFS – для поправки в DPT; SCL – для шкалы в DPT; TMP – для температуры в MTB. Все величины измеряются в метрах, кроме температуры, которая измеряется в градусах Цельсия.

Также на этой странице отображено время последней отправки сентенций, а также сами последние отправленные сентенции. В подвале находится информация об IP-адресе и порте, к кото-

Рис. 4. Страница управления сентенциями и настройки отправки: *1* – элементы управления сентенциями и правильностью контрольных сумм; *2* – элементы управления частотой отправки и уровнем шума; *3* – поле для отображения количества подключенных клиентов
Fig. 4. Page for sentence management and sending setting: *1* – control elements for sentences and correctness of checksums; *2* – control elements for send rate and noise level; *3* – a field to display the number of connected clients

рому необходимо подключиться, чтобы начать получать данные от симулятора. Также здесь располагается кнопка старт/стоп, которая отвечает за отправку сентенций.

На второй вкладке приложения (рис. 4) расположены компоненты для управления отправкой сентенций.

Около каждой из трех сентенций можно установить флаги отправки сентенции и выработки правильной/неправильной контрольной суммы. Также на этой странице можно настроить частоту отправки сентенций и уровень шума для значений глубин в сентенциях.

Для тестирования симулятора и демонстрации его работы с приемником сентенций в рамках данных исследований было создано специальное приложение. Оно подключается к серверу по IP-адресу и порту, получает от него сентенции и на их основе прорисовывает контур дна.

Приемник представляет собой окно с двумя полями для ввода IP и порта и кнопкой для подключения к введенному адресу. Под данным блоком расположено поле со списком всех полученных сентенций. При этом если сентенция имеет неверную контрольную сумму, ее текст становится красным. Правая часть приложения представляет собой график, на котором по пришедшим данным рисуется контур дна. Дополнительно внизу расположена информационная строка с текущими значениями DPT, DBT и MTW.

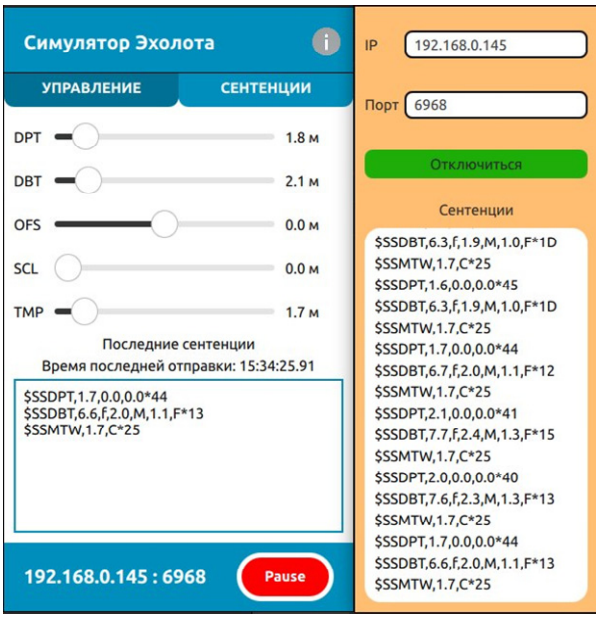

Рис. 5. Взаимодействие симулятора и приемника эхолота
Fig. 5. Interaction between the simulator and the echo sounder receiver

Возможный вариант взаимодействия симулятора и приемника эхолота представлен на рис. 5.

Демонстрация совместной работы симулятора и приемника приведена на рис. 6.

График, представленный на рисунке, можно интерпретировать следующим образом. На первом интервале были получены данные о глубине 2.5 м, равномерно и без шума (сплошной график). Затем симулятор был поставлен на паузу. Так как в этот период никакие данные не принимались, на

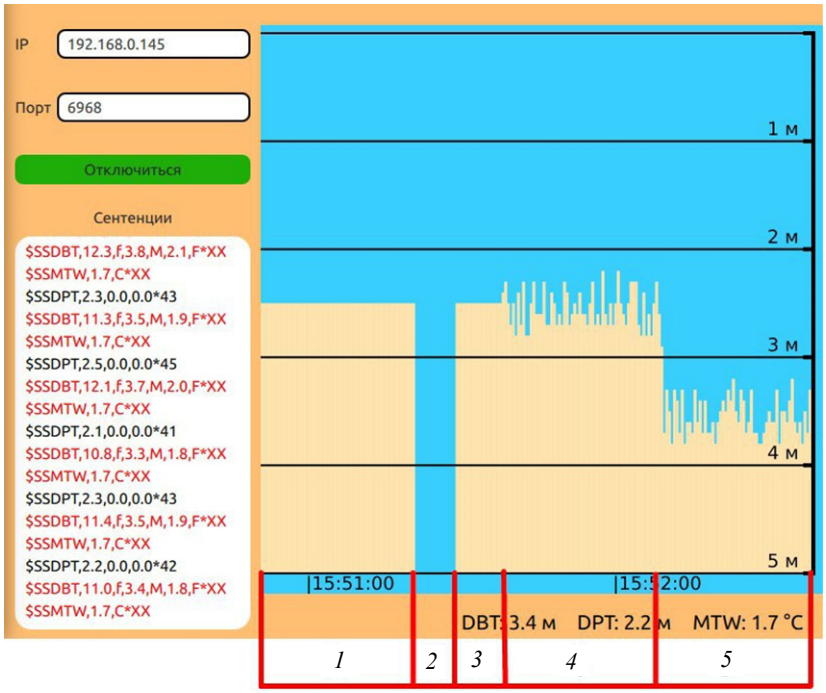

Рис. 6. Совместная работа симулятора и приемника эхолота
Fig. 6. Joint work of the simulator and the echo sounder receiver

графике появился разрыв (интервал *2*). Затем симулятор продолжил работу, и данные снова стали попадать в приемник (интервал *3*). На четвертом интервале в симуляторе был добавлен уровень шума, равный трем, что соответствует отклонению глубины на ±0.3 м. Соответственно, уровень глубины на данном промежутке колеблется от 2.2 до 2.8 м, что и видно на графике, который отображен небольшим и неравномерным «забором». Затем в симуляторе увеличили уровень глубины, что сразу отобразилось в приемнике на пятом интервале.

Также было проведено тестирование работы неправильных контрольных сумм. Для этого в симуляторе была отключена правильность контрольных сумм для DPT- и MTW-сентенций. На рис. 6 видно, что они были выведены красным шрифтом, что говорит об их неверной контрольной сумме.

Заключение. С помощью библиотеки Qt и языка программирования C++ разработано мобильное приложение, представляющее собой симулятор эхолота. Приложение вырабатывает различные сентенции и отправляет их по TCP/IP всем подключенным клиентам. Для разработки пользовательского интерфейса был использован декларативный язык программирования QML.

Приложение относится к классу бизнес-приложений и предназначено для тестирования различных навигационных приборов. Новизна работы заключается в том, что мобильное устройство, использующее данное приложение, может само выступать в качестве сервера. В результате отпадает необходимость в использовании дополнительных средств связи (в виде роутеров и им подобных устройств) для объединения тестируемого навигационного прибора и симулятора в единую сеть.

Список литературы

1. Как работает эхолот. URL: https://sonarmaster.ru/sonars/sonar-works/ (дата обращения 12.02.2023).

2. Описание NMEA протокола. URL: http://yug-gps.narod.ru/docs/000x/st007.htm (дата обращения 12.02.2023).

3. NMEA Simulator. URL: https://chrome.google.com/webstore/detail/nmea-simulator/dfhcgoinjchfcfnnkecjpjcnknlipcll (дата обращения 05.02.2023).

4. Шлее М. Qt 5.10 Профессиональное программирование на C++. СПб.: БХВ-Петербург, 2010. 1052 с.

5. Лафоре Р. Объектно-ориентированное программирование в C++. СПб.: Питер, 2018. 928 с.

6. Qt 6 QML. URL: https://www.qt.io/product/qt6/qml-book (дата обращения 10.02.2023).

Информация об авторах

Коротков Андрей Александрович – магистрант СПбГАСУ, 2-я Красноармейская ул., д. 4, Санкт-Петербург, 190005, Россия.
E-mail: ikorotkov1999@mail.ru

Букунов Сергей Витальевич – канд. техн. наук, доцент кафедры информационных технологий СПбГАСУ, 2-я Красноармейская ул., д. 4, Санкт-Петербург, 190005, Россия.
E-mail: sergeybukunov@yandex.ru
https://orcid.org/0000-0002-5983-0637

References

1. Kak rabotaet jeholot. URL: https://sonarmaster.ru/sonars/sonar-works/ (data obrashhenija 12.02.2023). (In Russ.).

2. Opisanie NMEA protokola. URL: http://yug-gps.narod.ru/docs/000x/st007.htm (data obrashhenija 12.02.2023). (In Russ.).

3. NMEA Simulator. URL: https://chrome.google.com/webstore/detail/nmea-simulator/dfhcgoinjchfcfnnkecjpjcnknlipcll (data obrashhenija 05.02.2023).

4. Shlee M. Qt 5.10 Professional'noe programmirovanie na C ++. SPb.: BHV-Peterburg, 2010. 1052 s. (In Russ.).

5. Лафоре Р. Ob#ektno-orientirovannoe programmirovanie v Ob#ektno-orientirovannoe programmirovanie v S++. SPb.: Piter, 2018. 928 s. (In Russ.).

6. Qt 6 QML. URL: https://www.qt.io/product/qt6/qml-book (data obrashhenija 10.02.2023).

Information about the authors

Andrey A. Korotkov – graduate student of Saint Petersburg State University of Architecture and Civil Engineering, 2nd Krasnoarmeiskaya st., 4, Saint Petersburg, 190005, Russia.
E-mail: ikorotkov1999@mail.ru

Sergey V. Bukunov – Cand. Sci. (Eng.), Assistant Professor of the Department of Information Technology, Saint Petersburg State University of Architecture and Civil Engineering, 2nd Krasnoarmeiskaya st., 4, Saint Petersburg, 190005, Russia.
E-mail: sergeybukunov@yandex.ru
https://orcid.org/0000-0002-5983-0637

Статья поступила в редакцию 17.04.2023; принята к публикации после рецензирования 16.05.2023; опубликована онлайн 25.09.2023.

Submitted 17.04.2023; accepted 16.05.2023; published online 25.09.2023.