ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ОБРАБОТКА ИНФОРМАЦИИ

УДК 004.42 10.5281/zenodo.3904110

С.Н. Верзунов, *verzunov@hotmail.com*
Институт машиноведения и автоматики НАН КР, Бишкек, Кыргызстан

РАЗРАБОТКА КРОССПЛАТФОРМЕННОГО ПРОГРАММНОГО КОМПОНЕНТА ТРАССОИСКАТЕЛЯ

В настоящей статье предложена архитектура кроссплатформенного программного компонента трассоискателя, основанная на разделении кода, зависимого от целевой платформы от кода, который может без каких-либо изменений запускаться на любой целевой платформе. Кроссплатформенность достигнута с помощью таких тулкитов как Kivy и инструментальных средств сборки приложений CMake, Swig и Buildozer, позволяющих портировать программный компонент трассоискателя на мобильную операционную систему Android, сохранив при этом его работоспособной и на десктопных операционных системах Windows и Linux. Это позволяет увеличить потенциальный круг возможных пользователей, и расширить функциональные возможности трассоискателя за счет применение встроенного во многие мобильные устройства gps-приемника и магнитного компаса, что также повышает удобство его практического использования.

Ключевые слова: кроссплатформенная архитектура, Kivy, buildozer, Android, Swig, ЛКАРД E502, ARM, python-for-android, Docker.

Введение

В работе [1] разработана архитектура программного компонента трассоискателя на базе устройства сбора данных ЛКАРД E502, основанная на использовании языка Python и модулей, предназначенных для обработки и визуализации данных. Как показали лабораторные испытания программной реализации данной архитектуры [2], важнейшим требованием, предъявляемым к такому программному обеспечению, является его кроссплатформенность, то есть способность работать на стационарных и портативных вычислительных устройствах с процессорами различной архитектуры под управлением разнообразных операционных систем. Это необходимо как в целях увеличения потенциального круга пользователей, так и для расширения функциональных возможностей трассоискателя, связанных с применение встроенного во многие мобильные устройства gps-приемника и магнитного компаса, а также удобства его практического использования. Трассоискатель с кроссплатформенным программным компонентом можно использовать на стационарных компьютерах с архитектурой процессора x86-64 для отладки и настройки аналоговой части трассоискателя, так и без изменения исходного кода на смартфонах и планшетных компьютерах с архитектурой процессора с архитектурой ARM под управлением операционных систем Android, iOS, и Windows 10 Mobile. В разработанном ранее варианте программного компонента кроссплатформен-

ность, по большей части уже была достигнута за счет применения языка программирования Python, и поэтому разработанный программный компонент может успешно работать под управлением операционных систем Windows и Linux, и, в частности возможна реализация трассоискателя в виде независимого устройства на базе одноплатного компьютера Raspberry Pi. Однако, некоторые части программного компонента трассоискателя, такие как графический интерфейс пользователя, библиотека визуализации данных matplotlib и драйвер платы сбора данных, тесно связаны с оборудованием и поэтому так или иначе должны быть портированы, т.е. адаптированы соответствующим образом, чтобы успешно взаимодействовать с различными устройствами используемой программно-аппаратной среды.

Постановка задачи

Задачей настоящего исследования является разработка и практическая реализация кроссплатформенной архитектуры программного компонента трассоискателя, выбор оптимальных методов и инструментальных средств обеспечения портирования программного компонента трассоискателя, позволяющих с минимальными усилиями, без изменения исходного кода использовать его на вычислительных устройствах с разнообразным аппаратным окружением управляемым различными операционными системами.

Материалы и методы

Kivy – это современное инструментальное средство (toolkit, тулкит) построения графического интерфейса пользователя, позволяющее создавать эргономичные интерфейсы для широкого спектра устройств. Скорость выполнения Kivy сопоставима с нативной мобильной альтернативой Java для Android или Objective C для iOS [3]. Основная идея, являющаяся ключевой для понимания всех преимуществ данного тулкита – это модульность и абстракция. Архитектура Kivy абстрагирована от таких базовых задач, как открытие окна, отображение графики и текста, воспроизведение звука, получение и отправка информации с устройств ввода вывода, и так далее. Это делает API (application programming interface, программный интерфейс приложения) расширяемым и простым в использовании. И, что важнее, это позволяет использовать низкоуровневые модули операционной системы, в которой запускается приложение. Например, в iOS, Andoroid и Windows существуют собственные API для подобного рода базовых задач.

В настоящее время Kivy активно развивается и имеет большое количество пользователей и разработчиков по целому ряду причин:

- Kivy имеет удобную в использовании встроенную поддержку мультитач-устройств с сенсорным экраном;
- Kivy специально создан для разработки кроссплатформенных графических приложений на языке Python и имеет более лучшую поддержку этого языка, по сравнению с адаптированными для него Qt и GTK +3;
- Kivy предоставляет более удобные API по сравнению с более ранними инструментами, такими как HTML и CSS) для построения графического интерфейса. Язык разметки интерфейса Kivy представляет собой адаптированный для использования совместно с Python вариант языка HTLM, отличающийся лаконичным, и поэтому более удобным синтаксисом.

- Kivy позволяет разрабатывать и поддерживать приложение без изменения исходного кода для множества операционных систем.

Последний пункт реализуется благодаря python-for-android [4] – инструменту для сборки с открытым исходным кодом, позволяющим упаковывать код на языке Python в отдельные APK (Android Package – формат архивных исполняемых файлов-приложений) для Android. Их можно передавать, устанавливать или загружать в магазин приложений Play Store, как и в любое другое приложение для Android. Этот инструмент изначально был разработан для кроссплатформенного графического тулкита Kivy, но в настоящее время поддерживает множество различных опций упаковки и может быть применен для создания любых приложений, написанных на языке Python.

Тулкит python-for-android реализует основные методы портирования приложения на операционную систему Android. В частности, он может скомпилировать интерпретатор Python, его зависимости для конкретной версии операционной системы, используемые в приложении библиотеки и код приложения Python для устройств с операционной системой Android. Этот этап является полностью настраиваемым, при этом самыми важным параметром является список зависимостей, т. е. модулей которые требуются приложению для работы. В случае программного компонента трассоискателя это модули для работы с ГИС (геоинформационной системой), вычислительными методами обработки и визуализации данных, подробнее о которых говорится в работе [1]. Аналогичный тулкит существует и для операционной системы iOS – Kivy iOS [5].

Для удобного управления всеми параметрами компиляции и сборки приложения под конкретную платформу используется такое инструментальное средство как Buildozer [6]. Это инструмент, позволяющий быстро упаковать мобильное приложение. Он автоматизирует весь процесс сборки, загружает необходимые компоненты, такие как python-for-android, Android SDK, NDK и т. д. В настоящее время Buildozer поддерживает упаковку для таких операционных систем как:

- Android: с помощью python-for-android (необходим компьютер с Linux или OSX, чтобы компилировать приложения для операционной системы Android);
- iOS: с помощью Kivy iOS. (для этого необходим компьютер с операционной системой OSX).

Поддержка других платформ включена в план (например, .exe для Windows, .dmg для OSX и т. д.) и будет реализована в ближайшее время. Процесс упаковки с помощью Buildozer управляется правилами, описанными в файле с именем buildozer.spec в каталоге приложения, определяя необходимые для его работы условия и такие параметры как заголовок, значок, используемые ресурсы (такие как, например, изображения, электронные подписи и д.р.) и библиотеки. Один и тот же файл спецификации используется для создания пакета для Android, iOS и т. д. Таким образом, реализуется возможность работы приложения на различных операционных системах без изменения исходного кода. В этом файле необходимо указать необходимые версии SDK, NDK, и другие зависимости приложения от среды, в которой оно будет работать. В частности, для портируемого программного компонента обязательно необходимо в качестве зависимостей указать модуль pyler [7], из которого импортируется набор функций, необходимых для работы со встроенным во многие устройства gps датчиком, который, в свою очередь, необходим для работы модуля ГИС mapview [8] – виджета Kivy для отображения ин-

терактивных карт. Кроме того импортируется и модуль graph, используемый в программном компоненте трассоискателя для отображения интерактивных графиков.

Другой важной частью, зависимой от аппаратного обеспечения частью является драйвер платы сбора данных Л КАРД E502. В целом он предоставляет собой три библиотеки:

- x502api – содержит общие функции для обоих модулей. Должна включаться в любой проект, работающий с одним из модулей, за исключением проектов, написанных только для L502 до появления библиотеки x502api, которые могут использовать только l502api;
- l502api – содержит специфические функции для модуля L502, а также функции, оставленные для совместимости с проектами, написанными до появления x502api;
- e502api – содержит специфические функции для модуля E502 [9].

Для портирования программного компонента необходимо адаптировать две библиотеки – x502api и e502api (т. к. в нем в настоящее время используется только модуль E502). Для ОС Windows предоставляется общий установщик «L-Card L502/E502 SDK», автоматически устанавливающий все необходимые драйвера, динамические библиотеки в системную директорию, а также все файлы, необходимые для подключения библиотеки к проекту приложения и примеры работы с ним в указанную директорию. Установка для ОС Linux так же не предоставляет особых усилий. Можно, например воспользоваться готовыми собранными пакетами, предоставляемыми «ЛКард». Это рекомендованный способ для дистрибутивов, для которых предоставляются собранные пакеты. Список поддерживаемых дистрибутивов приведен в документе [10].

Тем не менее, предоставляемые производителем исходные коды драйвера не являются полностью кроссплатформенными, так как ООО Л Кард не предоставляет поддержку операционных систем Android и iOS. Однако анализ исходных кодов показывает, что для них используется система сборки CMake. Эта система является свободным инструментом с открытым исходным кодом, основным разработчиком которого выступает компания Kitware. Название системы расшифровывается как «cross-platform make» (кроссплатформенная система сборки). Разработка инструмента ведётся с 1999 г., в качестве прототипа была использована утилита pcmaker, написанная в 1997 г. одним из авторов CMake. В настоящее время инструмент внедряется в процесс разработки многих программных продуктов, в качестве примеров широко известных проектов с открытым кодом можно привести KDE , MySQL , Blender , LLVM + clang и многие другие [11]. Принцип работы инструмента CMake напоминает принцип работы Buildozer: из каталога исходных кодов считывается файл CMakeLists.txt с описанием проекта, на выходе инструмент генерирует файлы проекта для одной из множества целевых операционных систем. Требования для сборки самого CMake включают наличие утилиты make и интерпретатора сценариев на языке bash либо скомпилированного инструмента CMake одной из предыдущих версий, а также компилятора C++ . При этом в исходных кодах CMake преднамеренно используются только возможности языка и стандартной библиотеки, поддерживаемые достаточно старыми версиями компиляторов. Таким образом, CMake является кроссплатформенным инструментом, переносим на большое количество платформ, что и позволяет модифицировать и скомпилировать драйвер устройства сбора данных Л КАРД E502 для мобильных операционных систем.

Для создания провайдера устройства ввода данных с платы Л КАРД Е 502 для библиотеки Kivy требуется SWIG – инструмент разработки программного обеспечения, который связывает программы, написанные на C и C ++, с различными языками программирования высокого уровня. SWIG используется с различными типами целевых языков, включая распространенные языки сценариев, такие как Javascript, Perl, PHP, Python, Tcl и Ruby. Список поддерживаемых языков также включает языки, не относящиеся к сценариям, такие как C#, D, Go, Java, включая Android, Lua, OCaml, Octave, Scilab и R. Также поддерживаются несколько интерпретируемых и скомпилированных реализаций Scheme (Guile, MzScheme / Racket) [12].

И наконец, для того, чтобы обеспечить повторяемую сборку программного компонента «Перспектива», результаты которой бы не зависели от используемой операционной системы и программного окружения, необходимо использовать инструментальное средство Docker. Контейнеры Docker предоставляют простые быстрые и надёжные методы разработки, распространения и запуска программного обеспечения, особенно в динамических и распределённых средах [13]. Docker позволяет создать контейнер, содержащий новейшие версии операционной системы и всех необходимых инструментов: Python, Kivy, Buildozer, CMake и др.

Таким образом, для портирования программного компонента необходимо решить целый ряд задач: разработать общую архитектуру, объединяющую графический интерфейс пользователя, драйвер платы сбора данных, тулкит Kivy и провайдер платы сбора данных в единую систему, отделив, однако, при этом платформозависимые и платформонезависимые части. Для реализации этой архитектуры потребуется доработать драйвер платы сбора данных и процесс его сборки, так, чтобы его можно было использовать на мобильных операционных системах, разработать архитектуру провайдера платы сбора данных и реализовать ее, обеспечив при этом повторяемую сборку платформозависимой части программного компонента трассоискателя для различных операционных систем.

Предлагаемое решение

Как уже было сказано выше, графический интерфейс пользователя, драйвер платы сбора данных и некоторые другие части программного компонента, как уже было сказано выше, тесно связаны с оборудованием и, в связи с этим, должны быть соответствующим образом адаптированы для обеспечения возможности его работы на операционных системах Android, iOS или Windows 10 Mobile. Схематически кроссплатформенная архитектура программного компонента, способного работать на мобильных операционных системах выглядит так, как показано на рис. 1. В предложенной архитектуре основной код программного компонента трассоискателя отделен от кода, зависящего от целевой рабочей платформы. Для портирования программного компонента требуется лишь собрать для конкретной операционной системы и типа процессора, показанную на рис. 1 платформозависимую часть. Тогда основной код программного компонента на языке Python можно будет запускать на любой платформе без изменений.

Сетевой стек TCP/IP, необходимый для связи с устройством сбора данных с помощью Ethernet или Wi-fi в настоящее имеется в любой современной мобильной и настольной операционной системе, однако не все функции поддерживаются одинаково хорошо.

Для того чтобы обеспечить работу на различных платформах в драйвере Л КАРД Е502 необходимо отключить поддержку автоматического поиска устройств в локальной сети, недоступную в настоящее время в операционной системе Android, изменив в файле *CMakeLists.txt* библиотеки e502api опцию

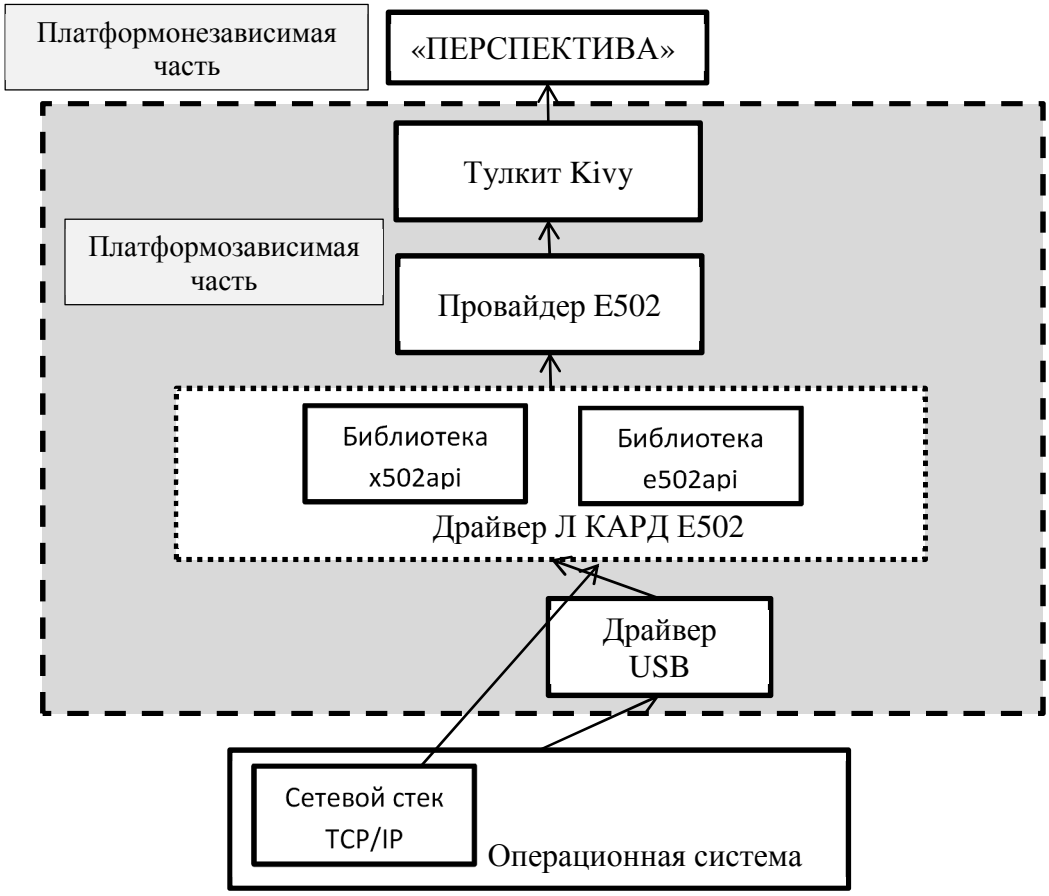

Рисунок 1 – Кроссплатформенная архитектура программного компонента трассоискателя на базе устройства сбора данных Л КАРД Е 502

option(E502API_ENABLE_DNSSD "enable dns-sd service discovery" OFF)

а в настройках программного компонента «Перспектива», в случае подключения устройства сбора данных по сети, необходимо указать IP адрес используемого устройства.

Кроссплатформенный драйвер USB можно найти по адресу [14], и добавить код:

```
if(android)
    include_directories(libusb-1.0 libusb-1.0/android)
…
if(linux)
    include_directories(libusb-1.0)
…
if(WIN32)
    include_directories(libusb-1.0 libusb-1.0/msvc)
…
```

в файл *CMakeLists.txt* библиотеки e502api. В настоящее время кроссплатформенным драйвером USB поддерживаются такие операционные системы как Linux, macOS, Windows, OpenBSD/NetBSD и Haiku.

В главный CMake-файл драйвера платы сбора данных для поддержки Android устройств следует добавить код:

```
if(${CMAKE_SYSTEM_NAME} MATCHES Android)
   set(CMAKE_SYSTEM_VERSION 21) # уровень API
   set(CMAKE_ANDROID_ARCH_ABI armeabi)
   set(CMAKE_ANDROID_STL_TYPE gnustl_static)
endif()
```

обеспечивающий корректную сборку для устройств на базе операционной системы Android. Как видно из приведенного выше кода, гарантируется работа на устройствах с API 21 и выше, что соответствует версии Android не менее чем 5.0. Работа на более старых устройствах, к сожалению невозможна, т.к. для них отсутствует поддержка стандарта POSIX Threads, необходимого для работы драйвера Л КАРД E502, реализующего поддержку многопоточных программ преимущественно ориентированных на исполнение на системах с общей памятью (Symmetric Multiprocessing, сокращённо SMP). Как известно, это такие системы, где установлено несколько процессоров или\и многоядерные процессоры и каждое ядро имеет доступ ко всей оперативной памяти компьютера [15].

Кроме того в файл /lib/osspec/osspec.c [16] необходимо внести исправление для компиляции драйвера платы сбора данных для операционной системы Android:

```
if (0) {/*timeout != OSSPEC_TIMEOUT_INFINITY*/
    struct timespec timeToWait;
    f_get_abs_time(timeout, &timeToWait);
    /*wt_res = pthread_timedjoin_np(thread, NULL, &timeToWait);*/
} else {
        wt_res = pthread_join(thread, NULL);
```

заменив тем самым неподдерживаемую в Android функцию стандарта POSIX Threads pthread_timedjoin_np на поддерживаемую во всех операционных системах функцию pthread_join.

На рис. 2 показана архитектура разработанного провайдера устройства ЛКард E502. Для начала работы с модулем необходимо установить с ним связь с помощью функции open_usb. Для идентификации устройств используется их серийные номера. Получить список серийных номеров всех подключенных устройств E502 можно с помощью get_usb_serial_list. Данная функция возвращает список, в котором сохранены найденные серийные номера, а принимает максимальное количество присоединенных модулей (по умолчанию это значение равно 16). Следует отметить, что с одним модулем одновременно может быть установлено только одно соединение. При попытке открыть модуль, с которым уже установлено соединение через другой описатель _hnd (возможно, в другой программе) open_usb вернет ошибку. При этом get_usb_serial_list по умолчанию возвращает список всех серийных номеров устройств, включая те, с которыми уже установлено соединение. Если нужно получить список только тех

устройств, с которыми еще не установлено соединения, то в get_usb_serial_list можно передать аргумент *only_not_opened=True*.

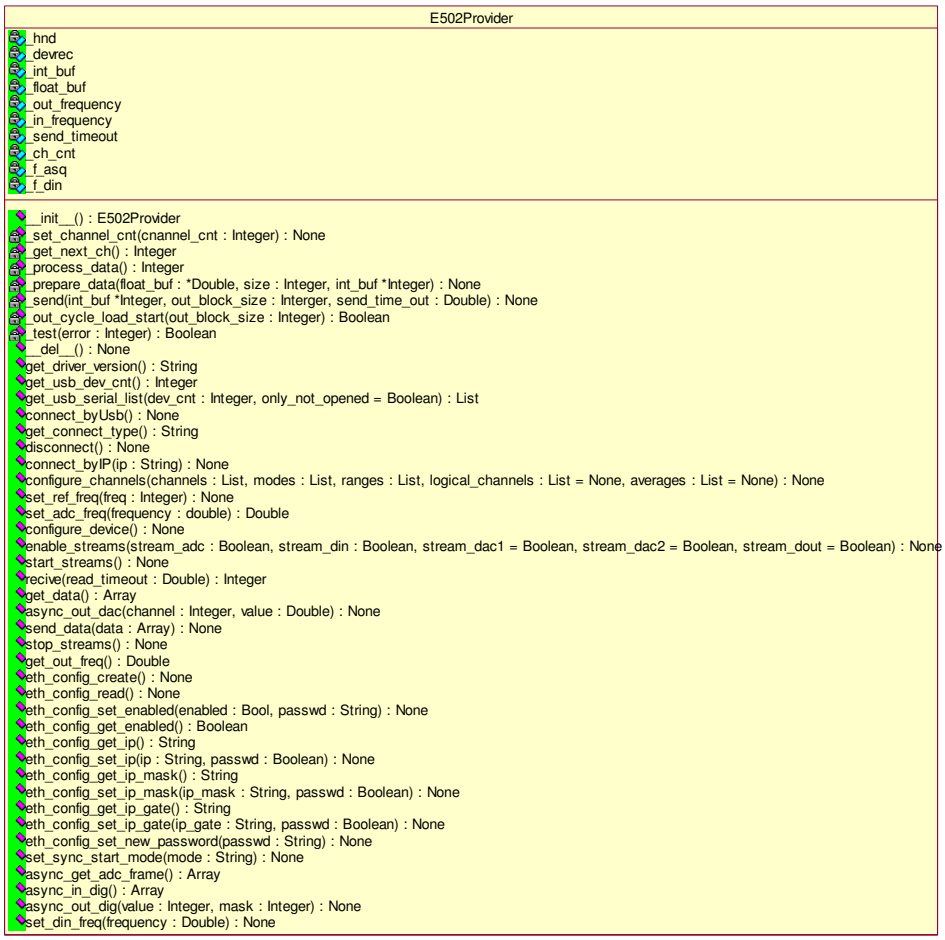

Рисунок 2 – Архитектура провайдера устройства ЛКард E502

Далее была разработана программа на языке Python, автоматизирующую создание контейнера Docker, и уставку в него операционной системы Ubuntu 18.04 и все вышеперечисленные инструментальных средств сборки. В этом контейнере был собран драйвер Л КАРД E502 и провайдер платы сбора данных Л КАРД E502 для библиотеки Kivy, состоящей из динамически разделяемой библиотеки и модуля для интерпретатора языка Python, созданных с помощью SWIG.

Наконец, был разработан файл buildozer.spec, требующийся для сборки APK-файла для операционной системы Android. В этом файле, кроме списка зависимостей и описания приложения, о которых уже говорилось выше, были указаны скомпилированные файлы провайдера E502; расширения файлов, содержащих необходимые ресурсы; минимальная версия Android API, необходимая для работы приложения; файлы драйвера Л КАРД E502 и архитектура процессора целевой платформы:

source.include_patterns = aarch64/python/*.so, aarch64/python/*.py
source.include_exts = py,kv,so
android.minapi = 21
android.add_libs_armeabi_v7a = aarch64/*.so, aarch64/devs/e502/*.so
android.arch = armeabi-v7a

garden_requirements = mapview, graph

Практическое исследование и выводы

Получившийся в результате сборки с помощью Buildozer APK-файл для проверки был установлен на планшет Google Nexus 7 3G (2012) и запущен (рис. 3) на

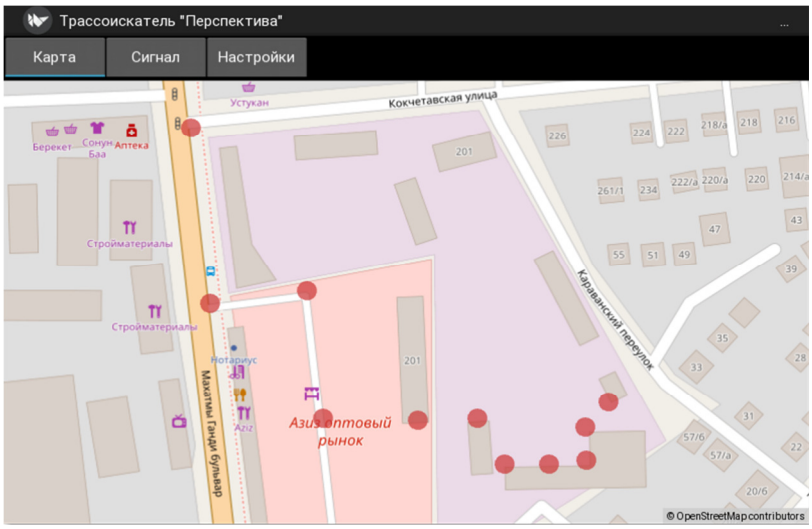

Рисунок 3 – Графический интерфейс кроссплатформенного программного компонента трассоискателя для работы с картой

операционной системе Android 7.2 с использованием ГИС OpenStreetMap [17]. Так же могут быть использованы и другие поставщики географических данных, например, Thunderforest или Google Maps. Портированный программный компонент трассоискателя позволяет наносить на карту найденные трассы кабелей на карту, выделяя их различными цветами, масштабировать карту до любых требуемых размеров и перемещать фокус ввода в любую необходимую для работы локацию.

Тестирование показало полную работоспособность портированного программного компонента трассоискателя и на смартфоне Meizu M2 Note с операционной системой Android 5.1. На рис. 4 показан графический интерфейс программного компонента трассоискателя для отображения принятых с помощью платы ЛКард E502 аналоговых и цифровых сигналов.

Заключение

Таким образом, в целях увеличения потенциального круга пользователей и для расширения функциональных возможностей трассоискателя, связанных с применением встроенного во многие мобильные устройства gps-приемника и магнитного компаса, а также удобства его практического использования была разработана и протестирована кроссплатформенная архитектура программного компонента трассоискателя и выбраны инструментальные средства ее реализации. Предложенная архитектура основана на разделении кода, зависимого от целевой платформы от кода, который может без каких-либо изменений запускаться на любой платформе. С помощью таких тулкитов как Kivy и инструментальных средств сборки кроссплатформенных приложений CMake и Buildozer удалось портировать программный компонент трассоискателя на мобильную операционную систему Android, сохранив при этом его работоспособной и на десктоп-

ных операционных системах Windows и Linux. В будущем возможно портирование и на другие мобильные устройства на базе операционных систем iOS и Windows 10 Mobile.

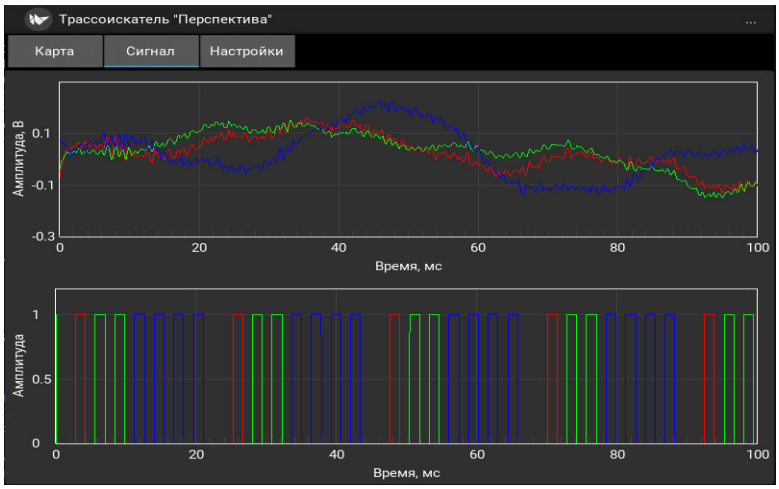

Рисунок 4 – Графический интерфейс кроссплатформенного программного компонента трассоискателя для отображения принятых сигналов.

Литература

1. Верзунов С.Н. Бочкарев И.В. Разработка программного компонента трассоискателя на базе устройства сбора данных Л КАРД E502 // Электротехнические системы и комплексы. 2018, №2(39). – С. 42-48.
2. Верзунов С. Н. Программный компонент трассоискателя на базе устройства сбора данных Л КАРД E502, ПК ПЕРСПЕКТИВА. Свидетельство об официальной регистрации программы для ЭВМ № 519 Кыргызская Республика, 27 августа 2018 г.
3. Dusty Phillips Creating Apps in Kivy – O'Reilly Media, 2014, 125 p.
4. https://python-for-android.readthedocs.io/en/latest/ (дата обращения 06.06.2019)
5. https://kivy.org/doc/stable/guide/packaging-ios.html (дата обращения 06.06.2019)
6. https://buildozer.readthedocs.io/en/latest/ (дата обращения 06.06.2019)
7. https://plyer.readthedocs.io/en/latest/ (дата обращения 06.06.2019)
8. https://mapview.readthedocs.io/en/latest/ (дата обращения 06.06.2019)
9. Борисов А. Современные устройства сбора данных L502/E502. Руководство программиста – М.: – ООО «Л Кард», 2016, 126 с.
10. Борисов А. Использование внешних репозиториев «L Card» для дистрибутивов Linux – М.: – ООО «Л Кард», 2019, 3 с.
11. Дубров Д. В. Система построения проектов CMake: учебник / Д. В. Дубров ; Южный федеральный университет. – Ростов-на-Дону: Издательство Южного федерального университета, 2015. – 419 с.
12. http://www.swig.org/doc.html (дата обращения 07.06.2019)
13. Моуэт Э. Использование Docker –М.: ДМК-Пресс, 2017. –354 с.
14. https://github.com/libusb/libusb (дата обращения 07.06.2019)
15. William Stallings. Operating Systems - Internals and Design Principles, 7th Edition. – Prentice Hall, 2011.
16. https://github.com/verzunov/e502-api-python (25.10.2019)
17. https://www.openstreetmap.org (дата обращения 07.06.2019)