УДК 004.75 DOI: 10.31799/2077-5687-2023-3-44-60

МОНИТОРИНГ СОСТОЯНИЯ АППАРАТНО-ПРОГРАММНОГО СТЕНДА ДЛЯ ОТРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Н. И. Синёв, Д. М. Астахов, И. М. Воронов, Д. И. Яцкевич
Санкт-Петербургский государственный университет аэрокосмического приборостроения

В статье рассматривается проблема диагностики вычислительных и аппаратно-программных устройств. Представлено программное обеспечение, позволяющее в автоматическом режиме диагностировать оборудование уникальной научной установки "Аэрокосмический стенд SpaceWire для исследования, сертификации и тестирования". Информация о состоянии аппаратно-программного узла выводится в графическом виде после синтаксического разбора.

Ключевые слова: графическая программа, управляющий компьютер, последовательный порт, диагностика, журнал событий, linux, minicom, Qt, C++.

Для цитирования:
Синёв, Н. И. Мониторинг состояния аппаратно-программного стенда для отработки программного обеспечения / Н. И. Синёв, Д. М. Астахов, И. М. Воронов, Д. И. Яцкевич // Системный анализ и логистика. – 2023. – № 3(37). – с. 44 – 60. DOI: 10.31799/2077-5687-2023-3-44-60.

MONITORING THE STATUS OF THE HARDWARE AND SOFTWARE STAND FOR TESTING SOFTWARE

N. I. Sinyov, D. M. Astakhov, I. M. Voronov, D. I. Yatskevich
St. Petersburg State University of Aerospace Instrumentation

The article deals with the problem of diagnostics of computing and hardware-software devices. The software is presented, which allows to automatically diagnose the equipment of the unique scientific stand "Aerospace stand SpaceWire for research, certification and testing". Information about the state of the hardware and software node is displayed graphically after parsing.

Keywords: graphics program, control computer, serial port, diagnostics, event log, linux, minicom, Qt, C++.

For citation:
Sinyov, N. I. Monitoring the status of the hardware and software stand for testing software / N. I. Sinyov, D. M. Astakhov, I. M. Voronov, D. I. Yatskevich // System analysis and logistics. – 2023. – № 3(37). – p. 44 – 60. DOI: 10.31799/2077-5687-2023-3-44-60.

Введение

Разработка аппаратно-программных комплексов является важной задачей, позволяющей продемонстрировать новую технологию перед этапом ее внедрения. При этом наиболее важной частью разработки является диагностика и мониторинг состояния комплектующих.

В большинстве случаев компьютерные программы скрывают процесс своего выполнения, выдавая только конечный результат. В таких ситуациях, без возможности отлаживать программный код, не представляется возможным отследить причину ошибочного результата или почему результат выполнения не совпадает с ожидаемым выводом. Для решения данной проблемы необходимо сохранять информацию о произошедших событиях на устройствах в журнал, который в дальнейшем можно проанализировать и выявить причину сбоя программы.

1. Проблемы диагностики последовательных устройств

Если иметь дело только с одним несвязным устройством, то ручная диагностика не будет занимать много времени. В другом случае, когда имеется большой набор из вычислительных узлов, которые связаны между собой через маршрутизаторы, диагностика каждого устройства по отдельности занимает много времени сотрудников лаборатории или предприятия, а также аппаратных и финансовых средств.

Еще одной важной проблемой является то, что обычному пользователю бывает сложно

обойтись без графического программного обеспечения. Если инженер может обойтись консольными приложениями, то менее квалифицированному сотруднику или стажеру, как правило, не хватает знаний для диагностики устройства через командную строку. Для того, чтобы решить эту проблему необходим программный комплекс с удобным графическим окружением, которое бы могло собирать, обрабатывать, систематизировать и представлять в наглядном виде пользователю всю информацию о состоянии устройств в уникальной научной установки “Аэрокосмический стенд SpaceWire для исследования, сертификации и тестирования” (Далее - УНУ АССИСТ) [1].

2. Алгоритм извлечения журнала событий

УНУ АССИСТ включает в себя отладочные модули с подключённым процессором Салют-ЭЛ24ОМ1 [2] и МС-30SF6EM-6U [3], а также коммутатор МСК-02REM-3U [4]. Каждое из данных устройств соединено с управляющим компьютером Raspberry Pi 3 Model B+ (Далее - Raspberry Pi) [5] через UART (Далее - последовательный порт), с помощью которого и производится диагностика устройства. Между собой устройства соединены интерфейсом SpaceWire [6], через который осуществляется передача пакетов данных.

Подключение к управляющему компьютеру производилось через сетевой протокол SSH. В сети Raspberry Pi имел название pi@stand-rpi.

Так как на Raspberry Pi установлена операционная система Raspberry Pi OS [7], всем подключенным внешним устройствам присваивается свой файл TTY (teletype), который лежит по пути /dev/. В нашем случае устройствам файлы внешних последовательных устройств имели имена /dev/ttyUSBn, где n - номер устройства от 0 до 2 (Далее - ttyUSB).

Начиная с пункта 2.3 в статье описываются команды терминала, которые входят в стандартный набор linux-команд.

Для реализации графического программного обеспечения и алгоритма извлечения информации с устройств был использован язык программирования C++ с фреймворком Qt версии 5.12.10 (Далее - Qt) [8].

На момент написания статьи на УНУ АССИСТ только вычислительный модуль Салют-ЭЛ24ОМ1 имел возможность хранить журналы внутренних событий. Поэтому дальнейшая работа проводилась лишь с этим устройством. На этом устройстве установлена операционная система Linux. Название устройства в операционной системе - mcom02.

Первой задачей являлась разработка алгоритма извлечения и обработки журналов событий устройства через последовательный порт. Есть несколько способов для решения данной задачи, рассмотрим их ниже.

2.1. Программа на Си/C++

Языки программирования Си и C++ имеют несколько библиотек для работы с последовательным портом [9]. Для настройки последовательного соединения с внешним устройством используется библиотека *termios.h*.

Создав объект структуры *struct termios*, можно получить текущие настройки последовательного порта через функцию *tcgetattr()*:

```
tcgetattr(fileno(stdin), &termios_object);
```

Для записи новых настроек в терминал используется функция *tcsetattr():*

```
tcsetattr(fileno(stdin), TCSANOW, &termios_object);
```

Когда последовательный порт настроен, необходимо получить файловый дескриптор файла ttyUSB для дальнейшего вызова таких функций, как read() и write() из библиотеки *unistd.h*.

Сохранение файлового дескриптора в переменную *port_fd*:

```
int port_fd = open ("/dev/ttyUSB", O_RDONLY);
```

Чтение информации из порта 20 символов и сохранение их в массив символов *buffer*:

read(port_fd, &buffer, 20);

Запись 20 символов из массива *buffer* в последовательный порт:

write(port_fd, buffer, 20);

Недостатки способа:

- Функция *read()* считывает определенное количество символов из строки. Если строка оказывается меньше указанной в параметре длины, то после считывания символов 13 и 10 из таблицы ASCII, символ возврата каретки и символа перехода на новую строку соответственно, функция *read()* начинает выдавать неопределенное поведение, перекидывая ошибки на считывание следующей строки. Количество символов в каждой строке является разным, а количество строк достигает до 2500. Предугадать программным кодом количество символов в строке перед ее непосредственным чтением невозможно.
- Устройство, на которое мы отправляли команды через функцию *write()*, не воспринимало их без дополнительных скриптов и обработчиков.

2.2. Программа на Qt C++

Фреймворк Qt добавляет в язык программирования C++ библиотеку для работы с последовательным портом *QSerialPort* [10]. Создав указатель на объект данного класса, для начала необходимо вызвать метод *setPortName()*, в параметр которого вводится название необходимого файла ttyUSB:

```
QSerialPort* serial_port = new QSerialPort();
serial_port->setPortName("/dev/ttUSB");
```

В дальнейшем можно задавать настройки работы с последовательным портом, например, задать скорость передачи информации через метод *setBaundRate()* или поменять количество читаемых бит с помощью метода *setDataBits():*

```
serial_port->setBaundRate(QSerialPort::Baud115200);
    serial_port->setDataBits(QSerialPort::Data8);
```

Для дальнейшей работы с последовательным портом необходимо вызвать метод *open()* у объекта класса *QSerialPort* для открытия устройства:

```
serial_port->open(QIODevice::ReadWrite)
```

Для записи данных в последовательный порт используется метод *write()*, а для считывания информации - метод *readAll()*:

```
serial_port->write("something");
serial_port->readAll();
```

Преимущества способа

- Если сравнивать с библиотеками *termios.h* и *unistd.h*, то класс *QSerialPort* является более удобным инструментом для работы с последовательным портом.

Недостатки способа:

- Аналогично программе на Си/C++ из предыдущего пункта программа, написанная на Qt C++, требует обработчик с обратной стороны связи.

2.3. Работа с последовательным терминалом

2.3.1. Передача команд через TTY

Чтобы получить информацию об устройстве, нужно вводить команды в терминал. С

помощью команды перенаправления вывода *“>”* можно посылать текст на последовательное устройство через ttyUSB. Например:

uname -a > /dev/ttyUSB0

Однако команда выполняется на управляющем компьютере, а результат выполнения будет перенаправлен в ttyUSB. Данную проблему можно решить командой echo.

echo “uname -a” > /dev/ttyUSB0

Экспериментальным путем было выяснено, что утилита последовательного терминала minicom [11] способна перехватывать перенаправленные команды и выполнять их на принимающем устройстве (Рис. 1).

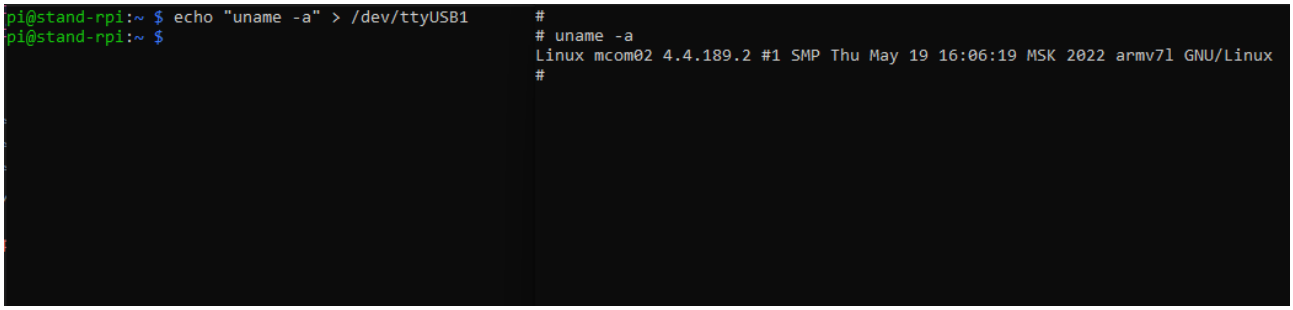

Рис. 1. Перехват команды minicom-ом

2.3.2. Чтение результата

minicom предоставляет возможность обмениваться файлами с последовательным устройством через протоколы xmodem, ymodem и zmodem [12]. Однако пользователю требуется прожимать комбинации клавиш, чтобы запустить передачу файла, найти сам файл и выбрать адрес копирования. Данные комбинации не считываются при перенаправлении командой *echo*. Для автоматизированной диагностики данный способ не подходит.

Тогда был найден способ мониторить все, что происходит во время выполнения minicom. Команда *cat /dev/ttyUSB* выводит все, что печатается в последовательный порт. Однако экспериментальным путем было выявлено, что некоторые строки не успевают считываться, тем самым пропадала часть информации. Для решения этой проблемы был найден аналог команды *cat - strace*.

Команда *strace* выводит все вызовы функций, которые вызывает процесс в linux. Для работы была настроена команда *strace* на процесс *minicom*, с заданием фильтра на вызов функции *read()*:

strace -e trace=read -s 150 -p <minicom pid> -o output.txt

После завершения выполнения программы создается текстовый документ *output.txt*, содержащий в себе весь журнал внутренних событий. Содержимое этого файла представлено на рисунке 2.

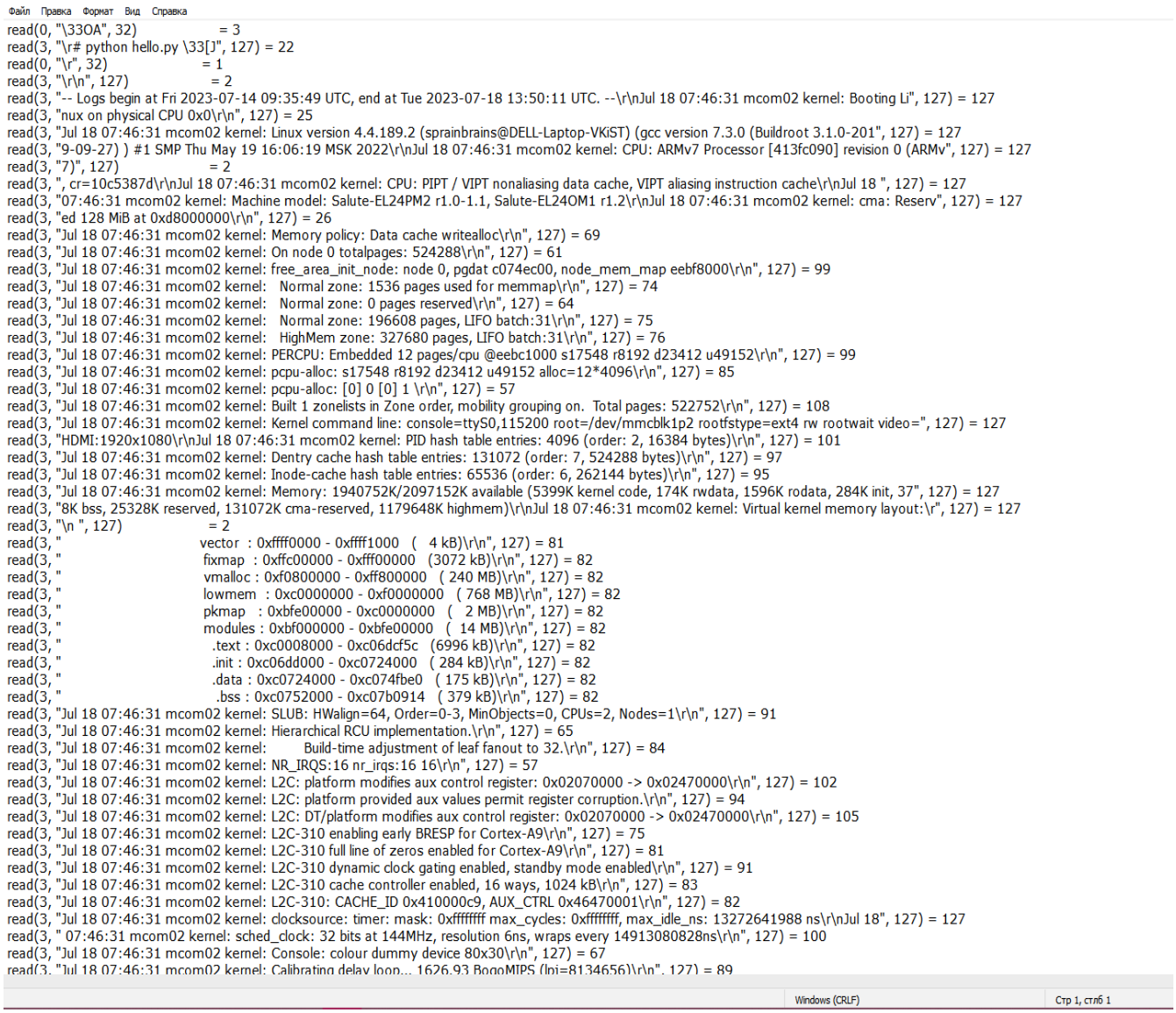

Рис. 2. Результат работы strace

Преимущества способа:

- Для диагностики устройства достаточно запустить программное обеспечение на управляющем компьютере. Нет необходимости ставить дополнительные скрипты и обработчики на диагностируемые устройства.

Недостатки способа:

- Без дополнительного окна терминала, в котором будет запущен minicom, способ работать не будет.
- Необходимо запустить команду strace в отдельном терминале, так как через основной терминал еще необходимо останавливать выполнение minicom и strace.

2.4. minicom и Qt C++

Для решения недостатков из предыдущего пункта, было принято использовать фреймворк Qt с его механизмами многопоточности, так как при этом способе не было необходимости ставить дополнительные обработчики и скрипты на диагностируемые устройства.

Для вызова функции в новом потоке необходимо создать объект класса QFuture и присвоить ему значение, которая возвращает функция *QtConcurent::run()*:

QFuture<void> name_thread = QtConcurent::run(thread_function);

В главном потоке программы отсылаются команды через функцию system() с перенаправлением в файл ttyUSB:

system("echo \"uname -a\" > /dev/ttyUSB ");

После выполнения программы, на управляющем компьютере создается файл с журналом всех внутренних событий диагностируемого устройства. После синтаксической обработки текстового файла информация готова к отображению в графической оболочке.

3. Графический интерфейс программного обеспечения для мониторинга состояния устройств в УНУ АССИСТ

3.1. Описание архитектуры

Графическое программное обеспечение (Далее - ПО) имеет несколько классов, которые выполняют свои определенные функции. Опишем их ниже.

Application Context - управляющий класс, который принимает в себя все зависимости и является связующим звеном между классами. Через этот класс осуществляется управление всем приложением.

Execution Logs - класс, объект которого нельзя создать. Этот класс имеет только статические методы и статический массив, который является журналом событий, включающий в себя все, что происходит по мере выполнения приложения.

Data Collector - класс, который собирает всю необходимую информацию. Первое - документация о каждом устройстве стенда УНУ АССИСТ. Второе - этот состояние устройств.

Собрав всю информацию из Data Collector, Application Context направляет данные в класс Main Window, в котором они будут отображаться графически.

При этом Main Window использует в себе функционал класса Scheme Label Widget. Это пользовательский виджет, который включает в себя изображение, реагирует на нажатие на себя и выдает сигнал об этом событии.

Для удобного просмотра передаваемого пакета SpaceWire создан класс Package Panel. Это пользовательское диалоговое окно, содержащее свои виджеты, которые, в свою очередь, отображает один из всех переданных пакетов.

Journals Panel - класс - пользовательское диалоговое окно - который предоставляет доступ к имеющимся журналам последовательных устройств и возможность генерировать журналы.

Схема архитектуры программного обеспечения продемонстрирована на рисунке 3.

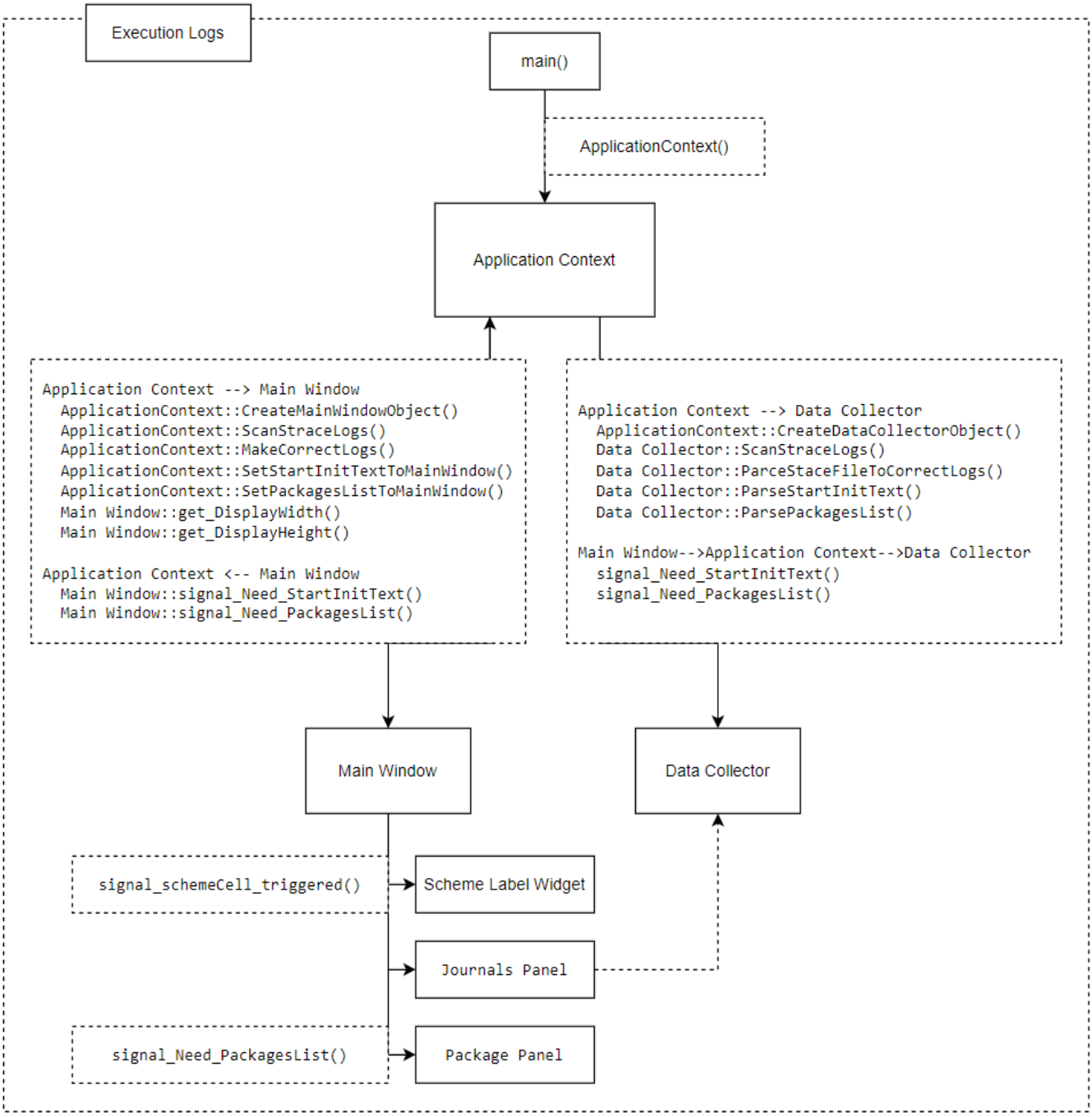

Рис. 3. Схема архитектуры ПО

3.2. Демонстрация работы

При запуске приложения открывается интерактивная схема стенда, на которой можно выбрать устройство. По умолчанию устройство не выбрано (Рис. 4).

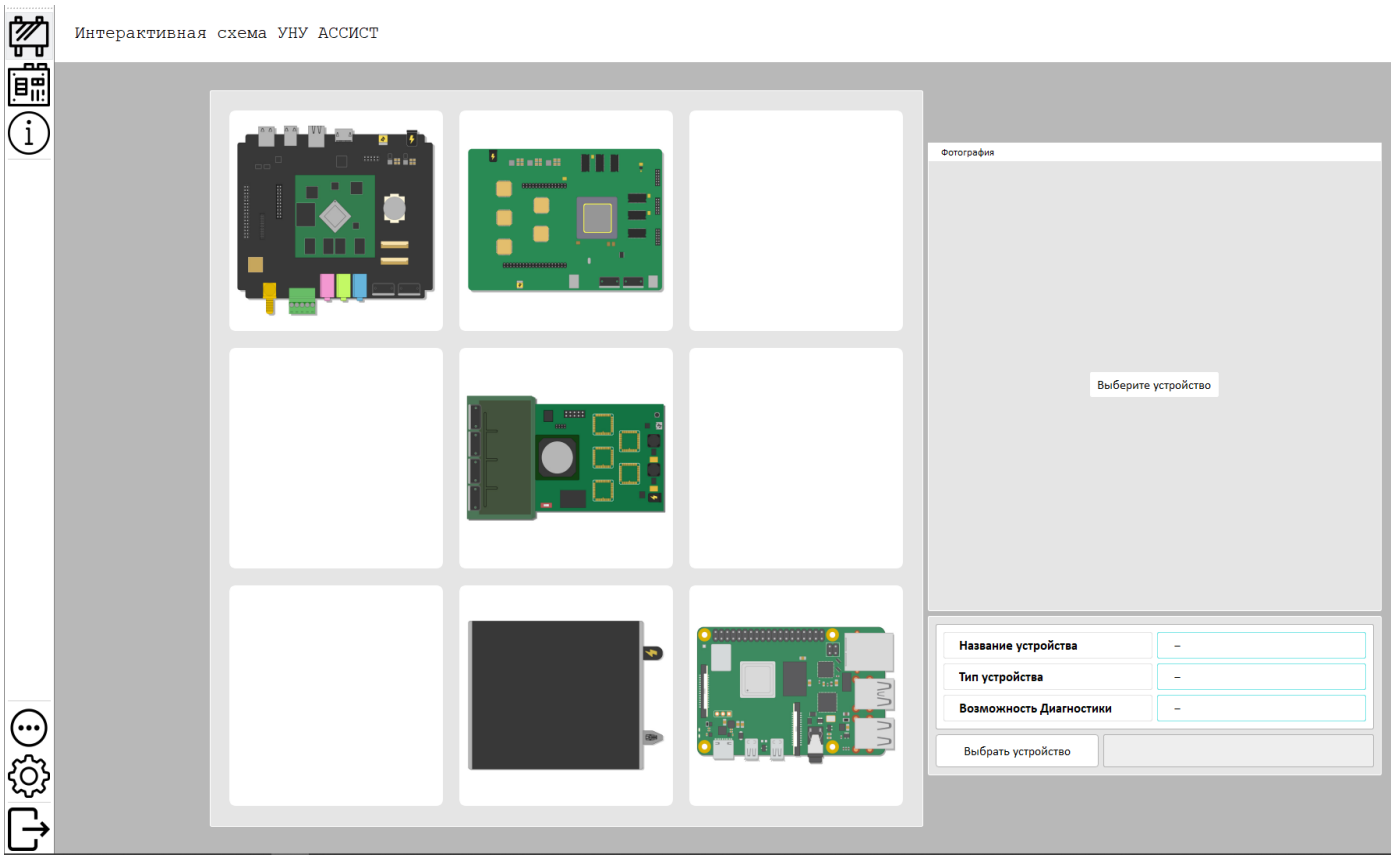

Рис. 4. Запуск приложения

В левой части экрана расположена панель кнопок:

- Интерактивная схема
- Состояние устройства
- Информация о приложении
- Информация о УНУ АССИСТ
- Настройки
- Завершение работы приложения

С помощью этой панели можно переключаться между вкладками приложения или выполнять определенные действия.

Если на интерактивной схеме устройство не выбрано, то при переходе на вкладку “Состояние устройства” появится предупреждение, показанное на рисунке 5, и переход не будет выполнен.

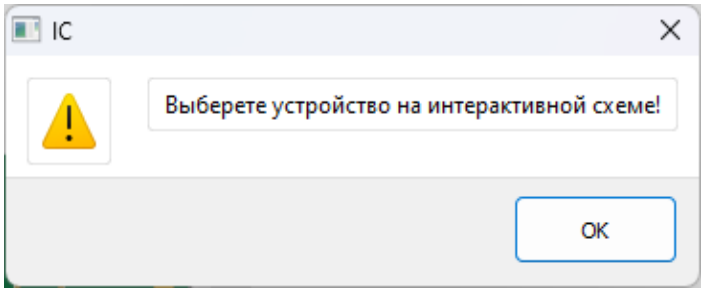

Рис. 5. Переход во вкладку состояния устройств

При нажатии на изображение схемы устройства отображается более крупное изображение устройства и краткая информация о нем (Рис. 6).

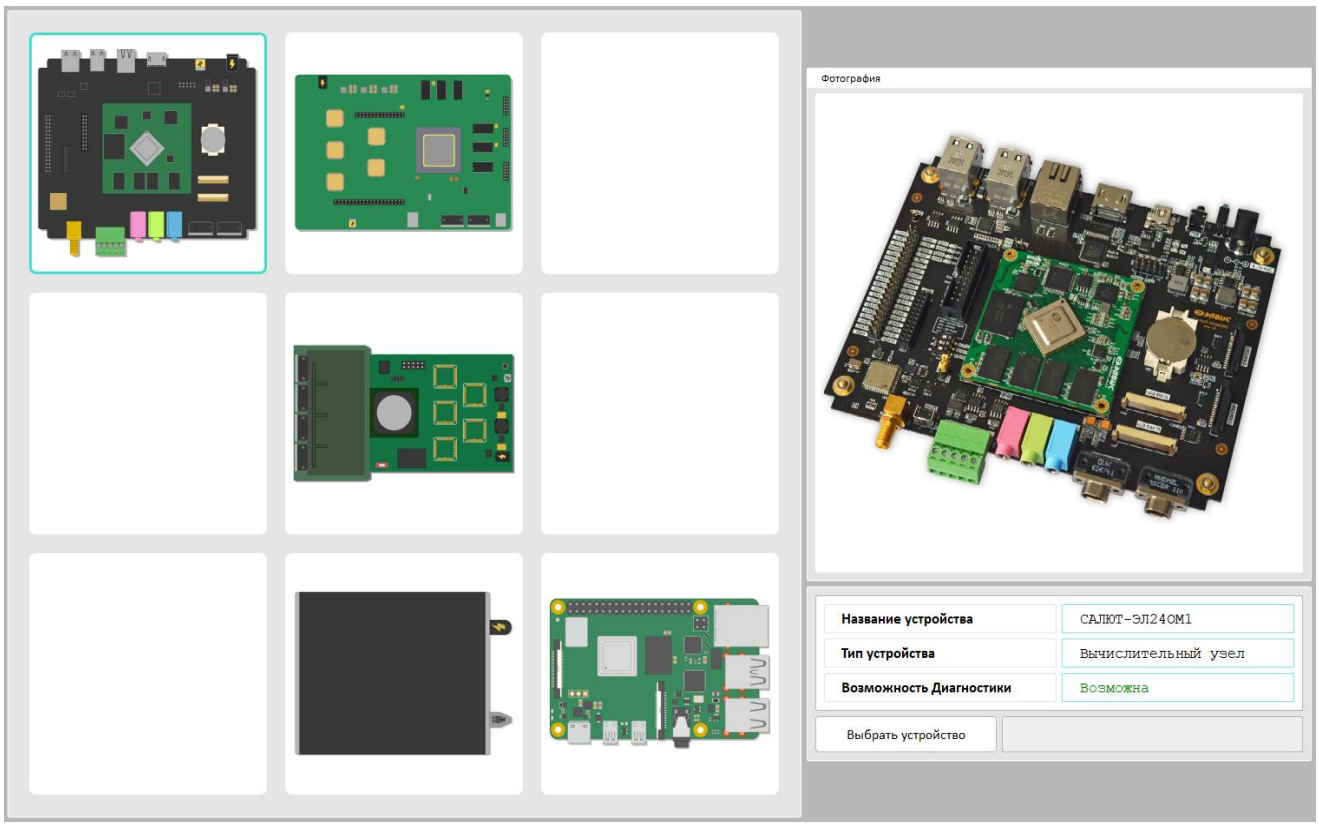

Рис. 6. На схеме выбрано устройство

При нажатии на кнопку “Выбрать устройство” пользователь перейдет во вкладку “Состояние устройства”. В левой части экрана появится панель, содержащая фотографию устройства, его краткое описание и панель кнопок, с помощью которых происходит отображается дополнительная информация (Рис. 7).

Рис. 7. На схеме выбрано устройство

В данной вкладке в первую очередь перед пользователем покажется появится краткое описание устройства (Рисунок 8), его документация (Рисунок 9) и ссылка на официального производителя (Рисунок 10).

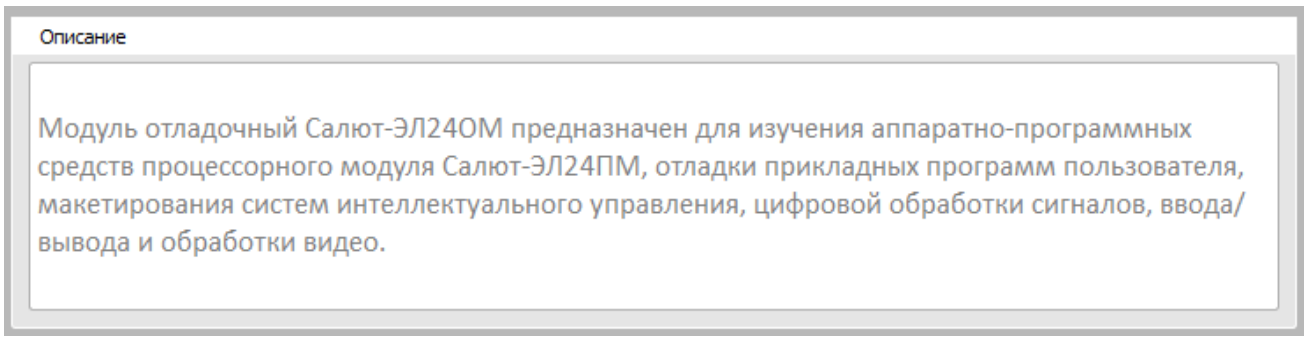

Рис. 8. Краткое описание устройства

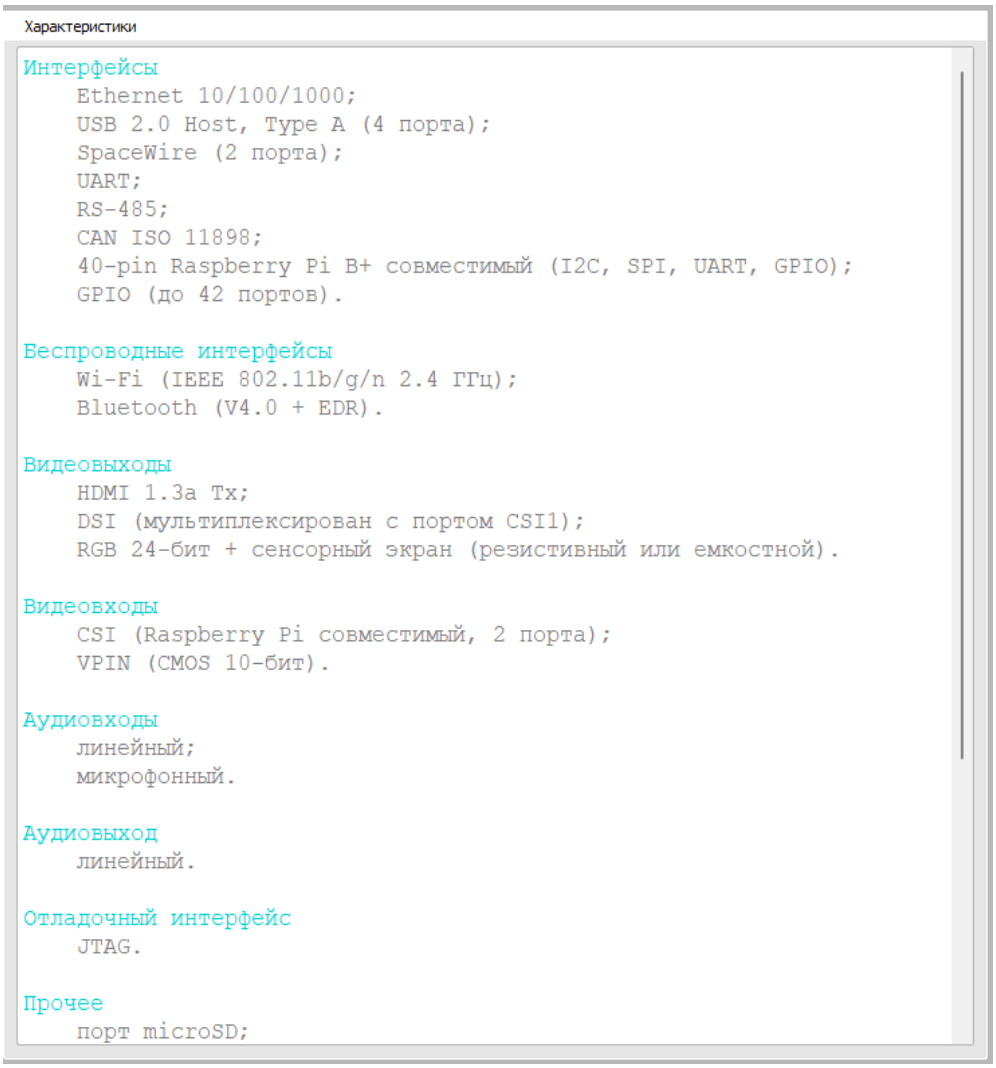

Рис. 9. Документация устройства

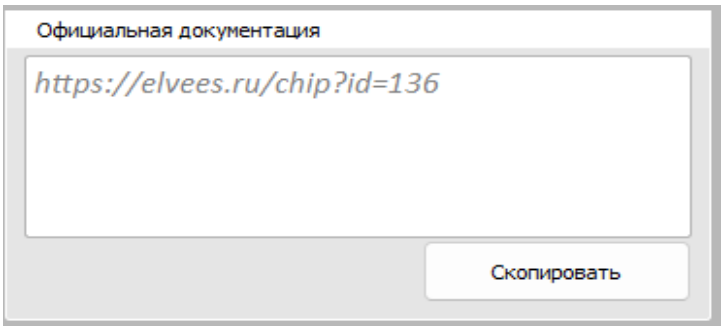

Рис.10. Ссылка на производителя

При нажатии на кнопку “Инициализация Запуска” отобразится информация, которая генерируется устройством при запуске (Рис. 11). В поле “Инициализация при запуске устройства” выведена информация о регистрах и стартовой инициализации устройства. В поле “Состояние каналов SpaceWire” отображается состояние всех каналов SpaceWire (Рис. 12).

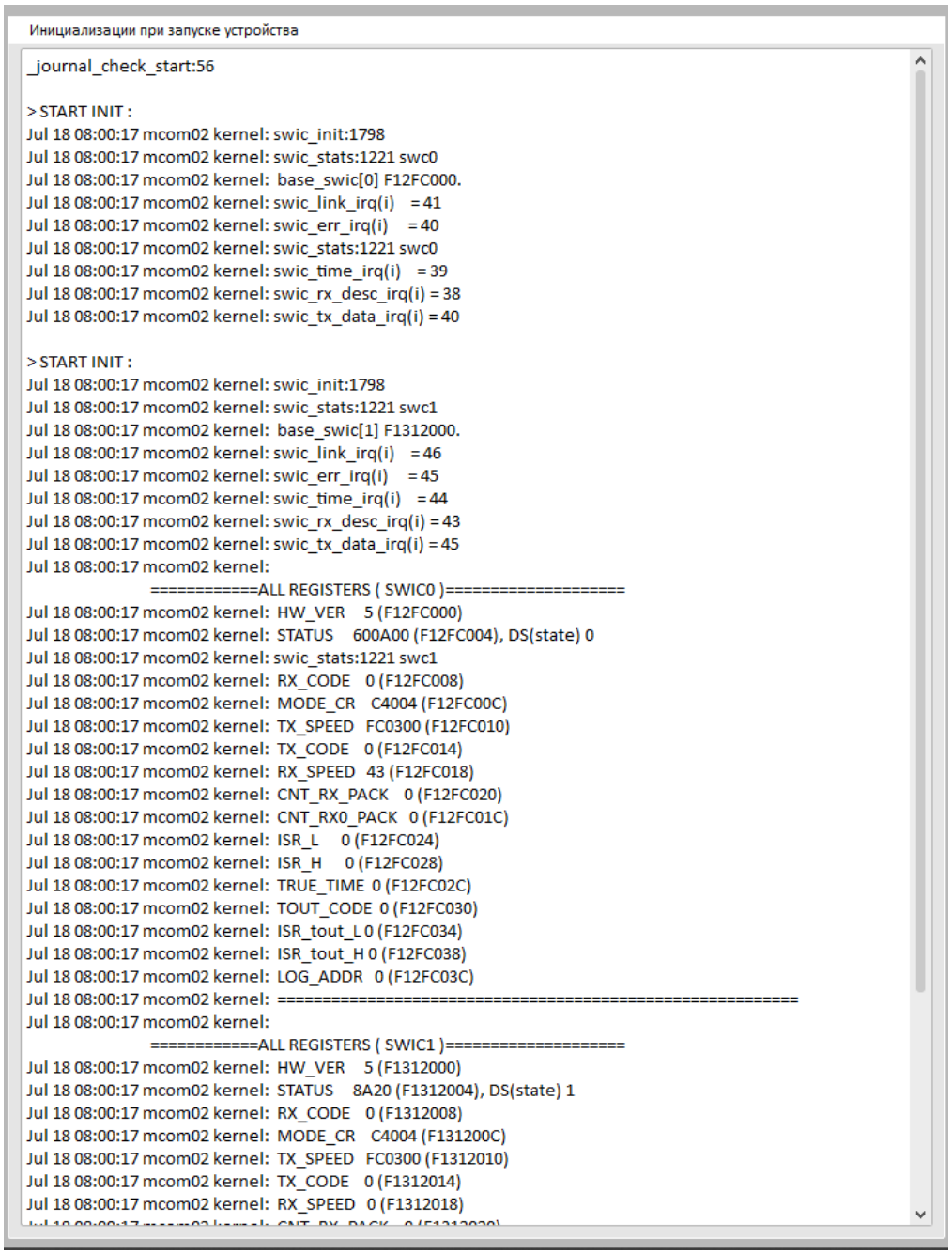

Рис. 11. Инициализация запуска

Рис. 12. Состояние каналов SpaceWire

При нажатии на кнопку “Панель Журналов” покажется диалоговое окно, которое будет включать в себя виджеты, представляющие управление текстовыми документами с журналами событий выбранного устройства (Рис. 13).

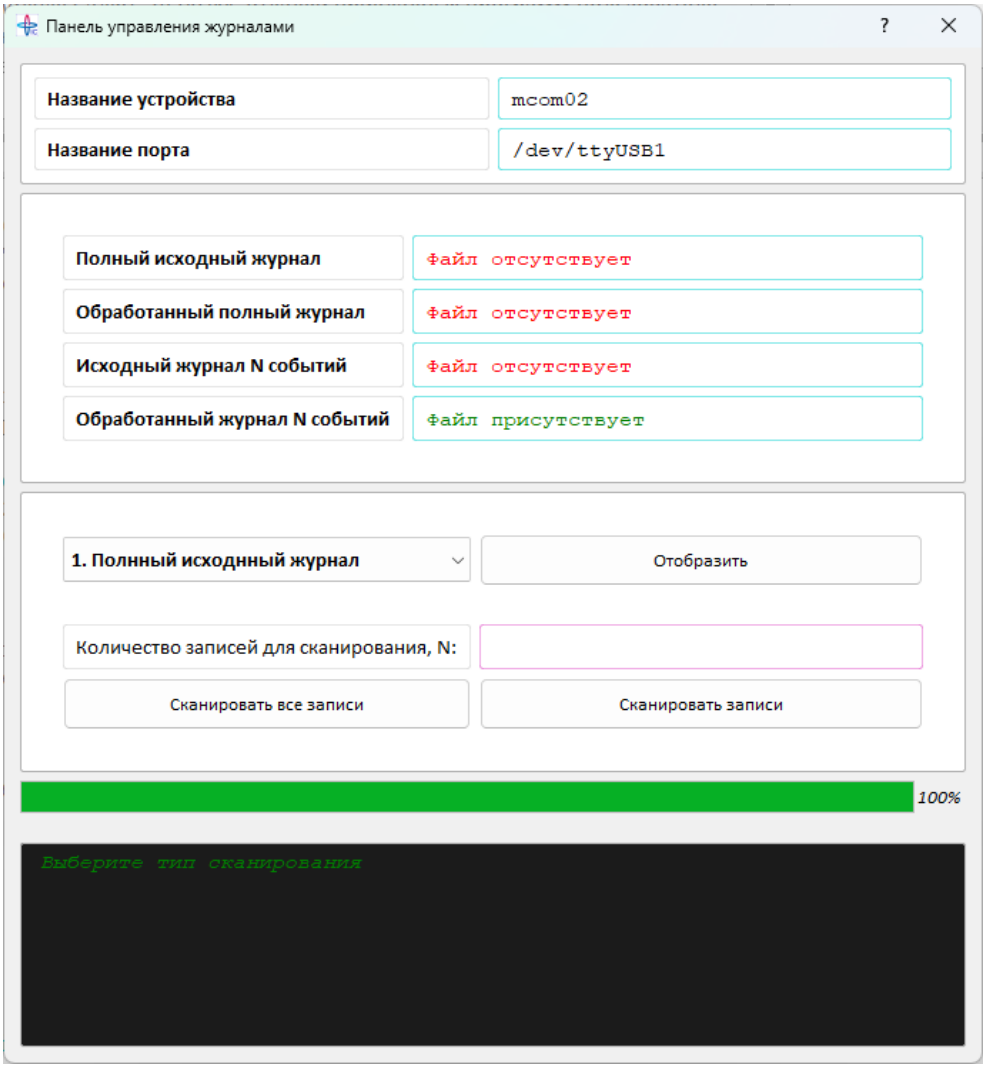

Рис. 13. Панель журналов

При выборе не диагностируемого устройства кнопки “Инициализация запуска”, “Панель Журналов” и “Обработка пакетов” выдают предупреждение о том, что устройство невозможно диагностировать (Рис. 14).

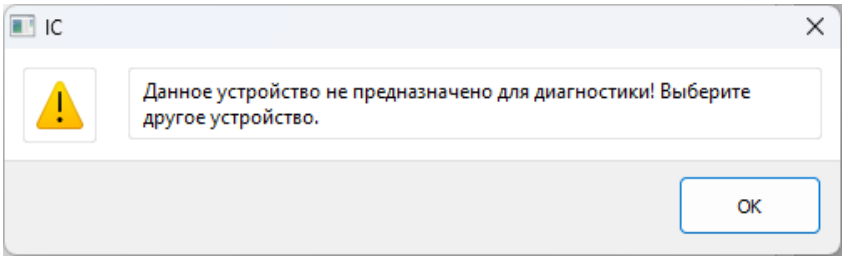

Рис. 14. Предупреждение крупным планом

При переходе на вкладку “Информация о приложении” перед пользователем покажется основная информация о приложении (Рис. 15).

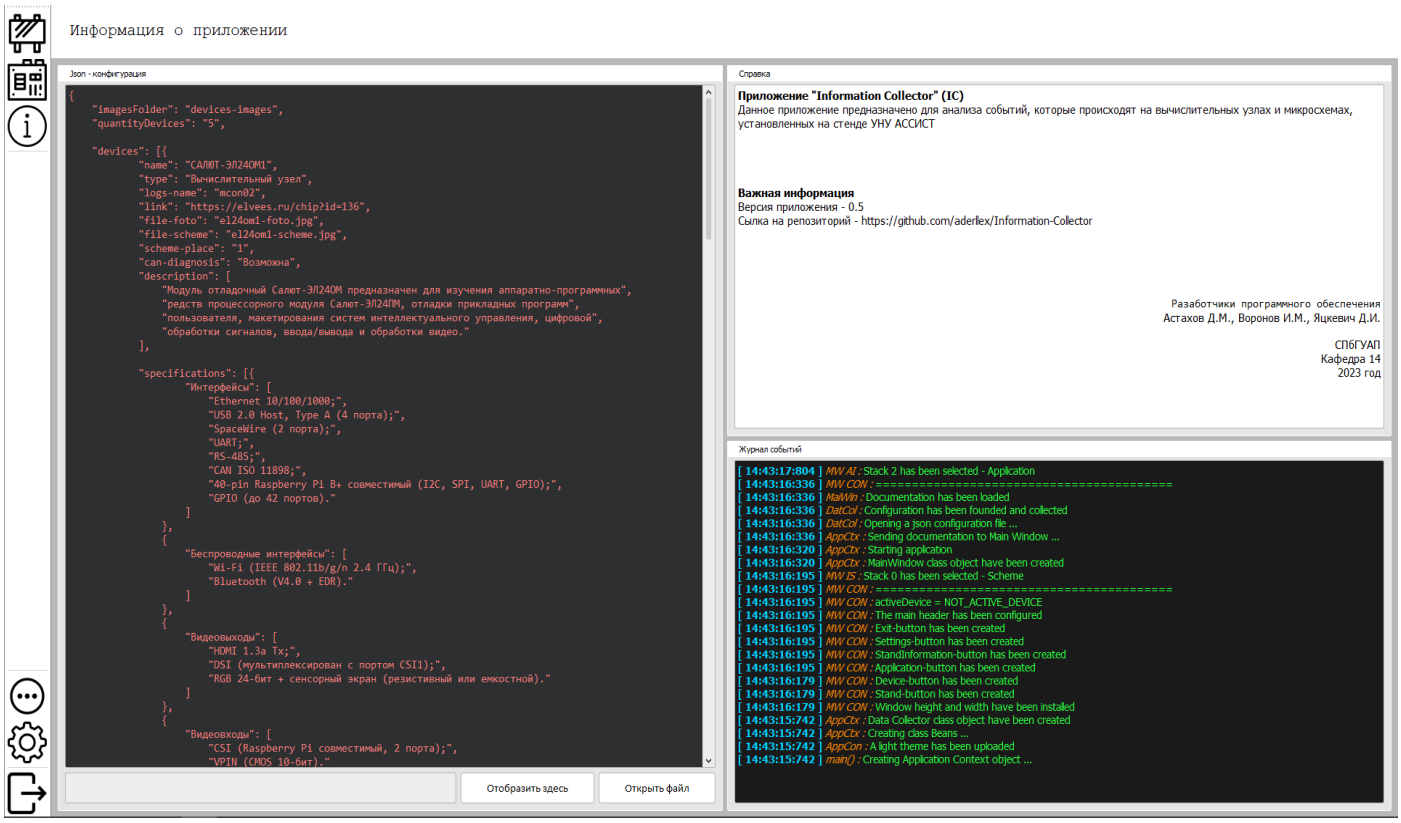

Рис.15. Вкладка информации приложения

В вкладке можно увидеть панель, хранящая в себе записи событий работы приложения (Рис. 16), краткую информацию о приложении (Рис. 17) и описание всего конфигурационного файла.

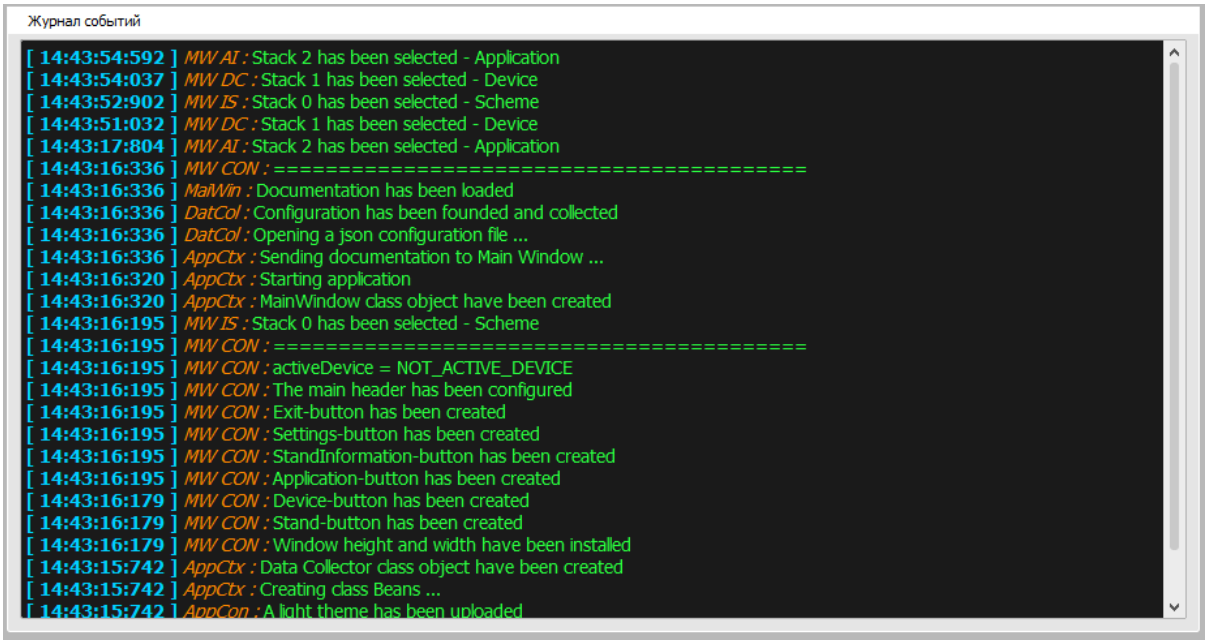

Рис. 16. Журнал событий работы приложения

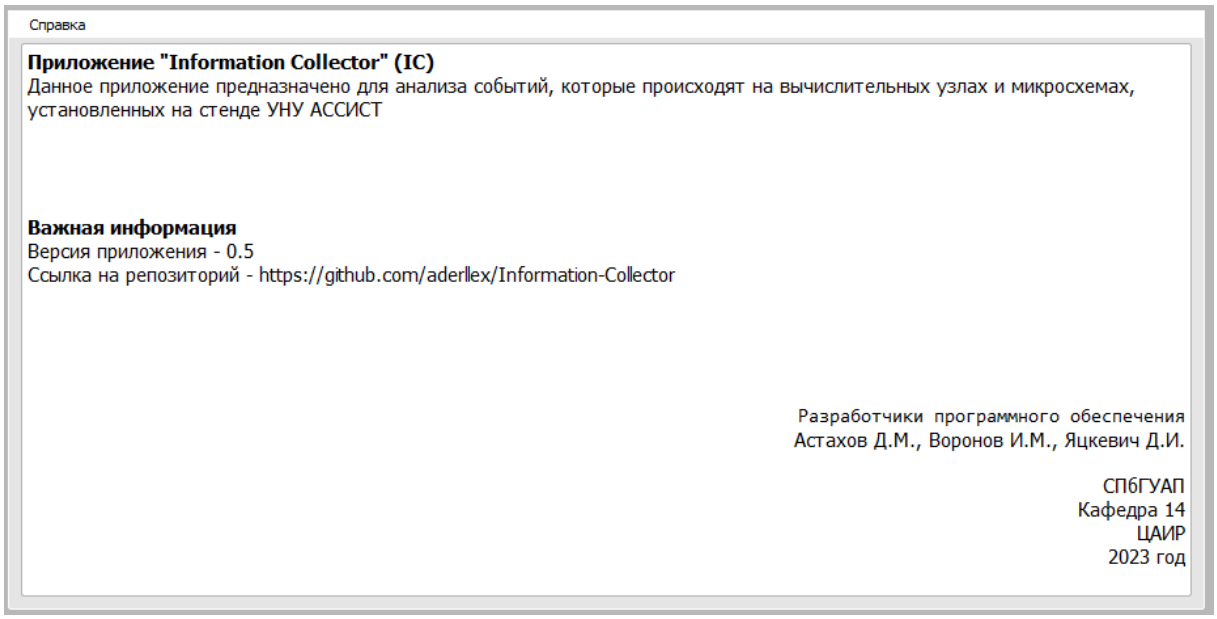

Рис. 17. Краткая информация

Заключение

В данной статье была представлена информация о разработке графического программного обеспечения, которое предназначалось для диагностики внешнего устройства через UART с использованием терминала последовательного соединения minicom. Весь процесс диагностики проводится программным обеспечением, находящимся только на управляющем компьютере. Нет необходимости ставить дополнительное ПО на диагностируемое устройство. В связи с этим данный способ диагностики является особенно актуальным среди остальных.

В результате были проанализированы механизмы диагностики внешнего последовательного устройства, был разработан уникальный алгоритм сбора журналов и было продемонстрировано графическое программного обеспечение для отображения информации внутренних событий.

Ближайшая разработка включает в себя внедрение механизма сбора информации в графическую оболочку, управление которым будет происходить из диалогового окна “Панель Журналов”, добавление информации во вкладку “Информация о УНУ АССИСТ”, указанной на официальном сайте [13], и создание системы настроек приложения, которые будут включать в себя настройки языка, цветовой темы и параметры запуска приложения.

В итоге можно сказать, что поставленная цель по разработке графического программного обеспечения для мониторинга состояния последовательных устройств частично выполнена. На данный момент ведется работа с настройкой ПО под MC-30SF6EM-6U, а также реализацией универсального интерфейса для сторонних разработчиков, которые будут приносить свои устройства для тестирования на УНУ АССИСТ.

Данная работа выполнена при финансовой поддержке Министерства науки и высшего образования Российской Федерации, соглашение № FSRF-2023-0003, "Фундаментальные основы построения помехозащищенных систем космической и спутниковой связи, относительной навигации, технического зрения и аэрокосмического мониторинга".

СПИСОК ЛИТЕРАТУРЫ

1. Синёв Н. И. Аппаратно-программное исследование коммуникационных протоколов для бортовых сетей / Н. И. Синёв, А. А. Карандашев, В. Л. Оленев, Н. Ю. Чумакова, А. Ю. Сыщиков // Санкт-Петербургский государственный университет аэрокосмического приборостроения. – 2022. – С. 18.

2. Салют-ЭЛ24ОМ1 [Электронный ресурс]. – URL: https://elvees.ru/chip?id=136/ (дата обращения: 13.09.2023)
3. МС-30SF6EM-6U [Электронный ресурс]. – URL: https://elvees.ru/chip?id=1405/ (дата обращения: 13.09.2023)
4. МСК-02REM-3U [Электронный ресурс]. – URL: https://elvees.ru/chip?id=1372/ (дата обращения: 13.09.2023)
5. Raspberry Pi [Электронный ресурс]. – URL: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/ (дата обращения: 13.09.2023)
6. SpaceWire Standard. Space Technology Centre. University of Dundee ECSS – Space Engineering. "SpaceWire – Links, Nodes, Routers and Networks". ECSS-E-ST-50-12 Draft H1. – 2018. – 124 p.
7. Raspberry Pi OS [Электронный ресурс]. – URL: https://www.raspberrypi.com/software/ (Дата обращения: 13.09.2023)
8. Qt 5 документация [Электронный ресурс]. – URL: https://www.qt.io/ (дата обращения: 13.09.2023)
9. Habr: Работа с COM-портом на Си в linux [Электронный ресурс]. – URL: https://habr.com/ru/companies/ruvds/articles/578432/ (дата обращения: 13.09.2023)
10. QSerialPort. Qt 5.15 Documentation [Электронный ресурс]. – URL: https://doc.qt.io/qt-5/qserialport.html (дата обращения: 13.09.2023)
11. Ubuntu Manpage: minicom - friendly serial communication program [Электронный ресурс]. – URL: https://manpages.ubuntu.com/manpages/trusty/man1/minicom.1.html (дата обращения: 13.09.2023)
12. Ubuntu Manpage: rx, rb, rz - XMODEM, YMODEM, ZMODEM (Batch) file receive [Электронный ресурс]. – URL: https://manpages.ubuntu.com/manpages/xenial/man1/rz.1.html (дата обращения: 13.09.2023)
13. ГУАП: ЦАИР – УНУ АССИСТ [Электронный ресурс]. – URL: https://guap.ru/m/assist-spw/ (дата обращения: 13.09.2023)

ИНФОРМАЦИЯ ОБ АВТОРАХ

Синёв Николай Иванович –
Ст. преподаватель, мл. науч. сотрудник
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: nikolay.sinyov@guap.ru

Астахов Даниил Максимович –
Студент
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: daniil-actaxov@yandex.ru

Воронов Илья Михайлович –
Студент кафедры аэрокосмических компьютерных программных систем
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: ilya.stud.spb@gmail.com

Яцкевич Даниил Игоревич –
Студент
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: j.daniil@bk.ru

INFORMATION ABOUT THE AUTHORS

Sinev Nikolay Ivanovich –
Senior Lecturer, Associate Junior Research
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: nikolay.sinyov@guap.ru

Astakhov Daniil Maksimovich –
Student
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: daniil-actaxov@yandex.ru

Voronov Ilya Mikhailovich –
Student of the Departments of Aerospace Computer Software Systems
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: ilya.stud.spb@gmail.com

Yatskevich Daniil Igorevich –
Student
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: j.daniil@bk.ru