УДК 004.3

Анализ работы с последовательными UART-портами и их настройка для передачи данных между устройствами

Ковров Данила Александрович kovrovdanila@gmail.com
Борсук Наталья Александровна borsuk65@yandex.ru
Задоркин Сергей Михайлович zadorkinsm@student.bmstu.ru

КФ МГТУ им. Н.Э. Баумана, Калуга, Россия

Аннотация. *Рассмотрена настройка интерфейсов UART на Raspberry Pi 4 при использовании операционной системы Ubuntu. Подробно описаны принципы работы интерфейса UART, его применение в современных встроенных системах и особенности реализации на аппаратном уровне в Raspberry Pi 4. Особое внимание уделено описанию структуры GPIO-разъема и выделению пинов, используемых для последовательного обмена данными. Рассмотрены различия между UART0, UART1 и дополнительными последовательными портами, а также приведены рекомендации по их практическому использованию. В заключительной части работы показана методика проверки корректности конфигурации UART при помощи программы, разработанной в Qt Creator на языке C++ с использованием класса QSerialPort.*

Ключевые слова: *Raspberry Pi 4, Ubuntu, UART, последовательный порт, GPIO, настройка интерфейсов, QSerialPort, Qt Creator, встроенные системы, IoT, обмен данными, микроконтроллеры*

Analysis of work with serial UART ports and their configuration for data transmission between devices

Kovrov Danila Alexandrovich kovrovdanila@gmail.com
Borsuk Natalia Alexandrovna borsuk65@yandex.ru
Zadorkin Sergey Mikhailovich zadorkinsm@student.bmstu.ru

BMSTU (KB), Kaluga, Russia

Abstract. *The article discusses the configuration of UART interfaces on the Raspberry Pi 4 when using the Ubuntu operating system. The principles of UART operation, its applications in modern embedded systems, and its hardware implementation on the Raspberry Pi 4 are described in detail. Special attention is given to the structure of the GPIO connector and the identification of pins used for serial data exchange. Differences between UART0, UART1, and additional serial ports are examined, along with practical recommendations for their use. In the final part of the work, a method for verifying the correctness of UART configuration is presented using a program developed in Qt Creator in C++ with the QSerialPort class.*

Keywords: *Raspberry Pi 4, Ubuntu, UART, serial port, GPIO, interface configuration, QSerialPort, Qt Creator, embedded systems, IoT, data exchange, microcontrollers*

Введение. Одноплатные компьютеры Raspberry Pi в последние годы получили широкое распространение в образовательных проектах, исследовательской деятельности и разработке встроенных систем [1]. Благодаря низкой стоимости, компактным размерам и широким функциональным возможностям они стали важным инструментом в области Интернета вещей (IoT), робототехники и автоматизации. Одним из наиболее востребованных способов взаимодействия с внешними устройствами остается асинхронный последовательный интерфейс UART (Universal Asynchronous Receiver-Transmitter).

Актуальность исследования заключается в том, что последовательные интерфейсы до сих пор применяются как основной канал связи в системах реального времени, где требуется надежная и простая передача данных. Несмотря на существование современных интерфейсов (USB, Ethernet, Wi-Fi), UART остается незаменимым в ряде областей: от отладки микроконтроллеров до взаимодействия с датчиками, GPS-модулями и беспроводными модемами.

Научная новизна работы состоит в практической реализации методики включения и настройки сразу нескольких UART-интерфейсов на Raspberry Pi 4 под управлением операционной системы Ubuntu, а также в разработке прикладного программного обеспечения для тестирования их работоспособности с использованием Qt Creator и библиотеки QSerialPort.

Цель работы — исследовать особенности настройки UART-портов на Raspberry Pi 4 и продемонстрировать их использование для обмена данными между устройствами.

Для достижения поставленной цели были сформулированы следующие задачи:

– рассмотреть принципы работы UART и его аппаратную реализацию в Raspberry Pi 4;

– выполнить настройку UART-портов в операционной системе Ubuntu;

– разработать и протестировать программное обеспечение для проверки корректности передачи данных;

– оценить возможности применения полученной методики для образовательных и практических задач в области встроенных систем и IoT.

Материалы и методы. На практике UART используется для подключения микроконтроллеров (например, STM32 или Arduino), датчиков, GPS-модулей, беспроводных модемов и других устройств, где необходим обмен текстовыми или бинарными данными. Скорость передачи в UART определяется параметром baud rate и может варьироваться от 9600 до нескольких мегабит в секунду, хотя наиболее распространенными являются значения 9600, 57600 и 115200 бод. Дополнительно конфигурируются параметры формата кадра: количество бит данных (обычно 8), число стоп-битов (1 или 2) и наличие/отсутствие контроля четности.

Raspberry Pi 4 оснащен несколькими аппаратными контроллерами UART. Основной порт, обозначаемый как UART0 или /dev/ttyAMA0, может использоваться системой для вывода сообщений при загрузке и предоставления до-

ступа к консоли [2]. Вторичный порт (UART1) и дополнительные контроллеры могут быть активированы пользователем при помощи механизма device tree overlays. В операционной системе Ubuntu, установленной на Raspberry Pi 4, работа с последовательными интерфейсами возможна через устройства /dev/ttyAMA* и /dev/ttyS*, где первые обычно соответствуют аппаратным UART, а вторые могут быть псевдопортами [3].

Для работы с UART на Raspberry Pi используется разъем GPIO (General Purpose Input/Output) [4], расположенный на краю платы и содержащий 40 контактов (рис. 1). Каждый контакт может выполнять несколько функций: цифровой ввод/вывод, шину I²C, SPI, а также линии UART. По умолчанию для основного UART задействуются контакты:

– GPIO14 (TXD0) — передача данных, вывод № 8 на разъеме;

– GPIO15 (RXD0) — прием данных, вывод № 10 на разъеме.

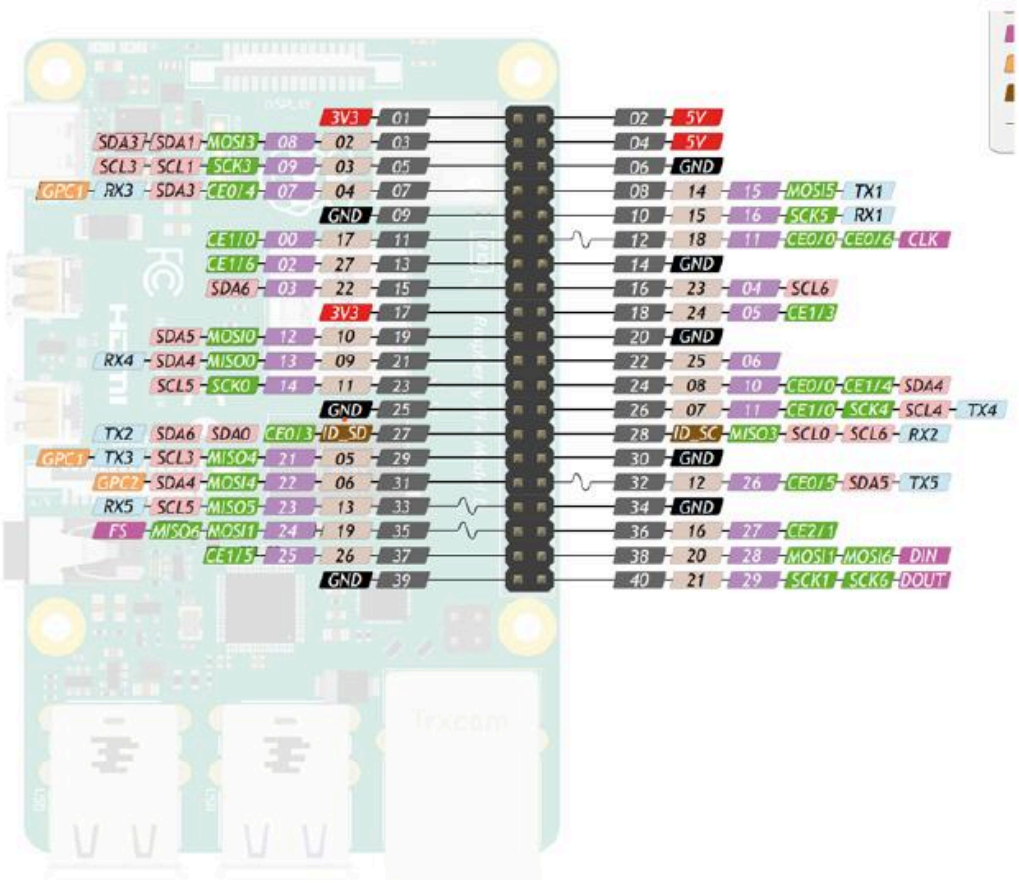

Рис. 1. Распиновка Raspberry Pi 4 model B

Эти контакты находятся в верхнем ряду 40-контактного разъема и позволяют напрямую подключать внешние устройства. Дополнительно могут быть задействованы другие GPIO для работы с дополнительными UART, если они включены в конфигурации.

Стандартная схема распиновки Raspberry Pi 4 включает два ряда по 20 контактов, из которых первые два обеспечивают питание: вывод № 1 —

3,3 В, вывод № 2 — 5 В, вывод № 6 — земля (GND). Восьмой вывод соответствует TX основного UART, а десятый — RX. Это позволяет подключить кабель или модуль UART непосредственно к GPIO-разъему без использования дополнительных переходников, если уровни напряжений совпадают (Raspberry Pi работает на 3,3 В).

Таким образом, прежде чем переходить к настройке пинов в Ubuntu, необходимо понимать, что Raspberry Pi 4 имеет несколько уровней конфигурации UART:

– аппаратная поддержка на уровне SoC, где реализовано несколько контроллеров;

– программная конфигурация в Ubuntu, включающая отключение системной консоли и активацию дополнительных портов;

– физическое подключение через GPIO-разъем с учетом схемы распиновки и напряжений.

Именно эта последовательность обеспечивает корректную работу интерфейса и дальнейшую возможность тестирования передачи данных.

Однако при первом 5 UART интерфейсов не будут доступны для работы. Поэтому необходимо сначала их включить.

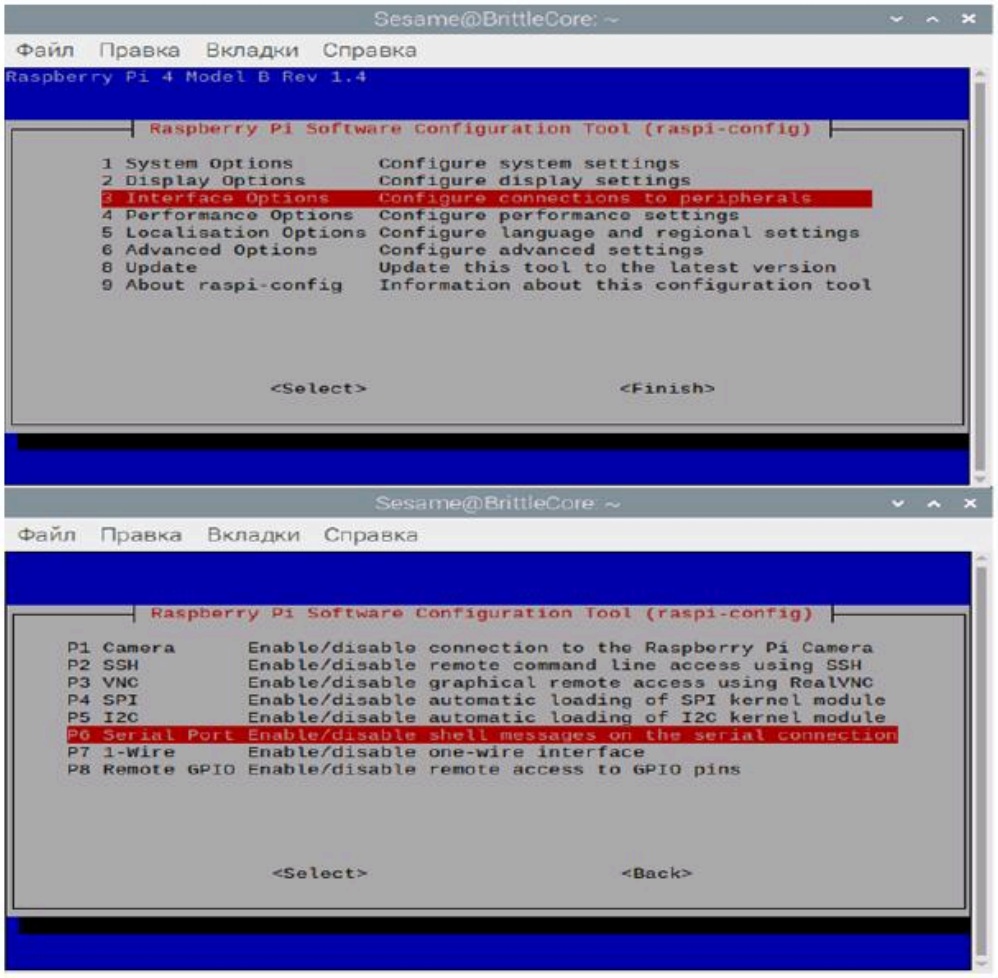

Рис. 2. Окно конфигуратора

Настройка портов на Ubuntu осуществляется через конфигуратор и системные файлы. Первым делом в конфигураторе необходимо отключить оболочку с портов. Для этого в терминале нужно прописать sudo raspi-config, после чего откроется окно конфигуратора (рис. 2), в котором по цепочке Interfacing Options – “Serial Port” необходимо отключить оболочку от порта UART0 и перезапустить плату.

После перезапуска платы надо включить порты UART1, UART2, UART3, UART4, UART5. Для этого необходимо отредактировать файл конфигурации config.txt, выполнив команду: sudo nano /boot/config.txt. В секцию [all] необходимо добавить строки включения портов (рис. 3).

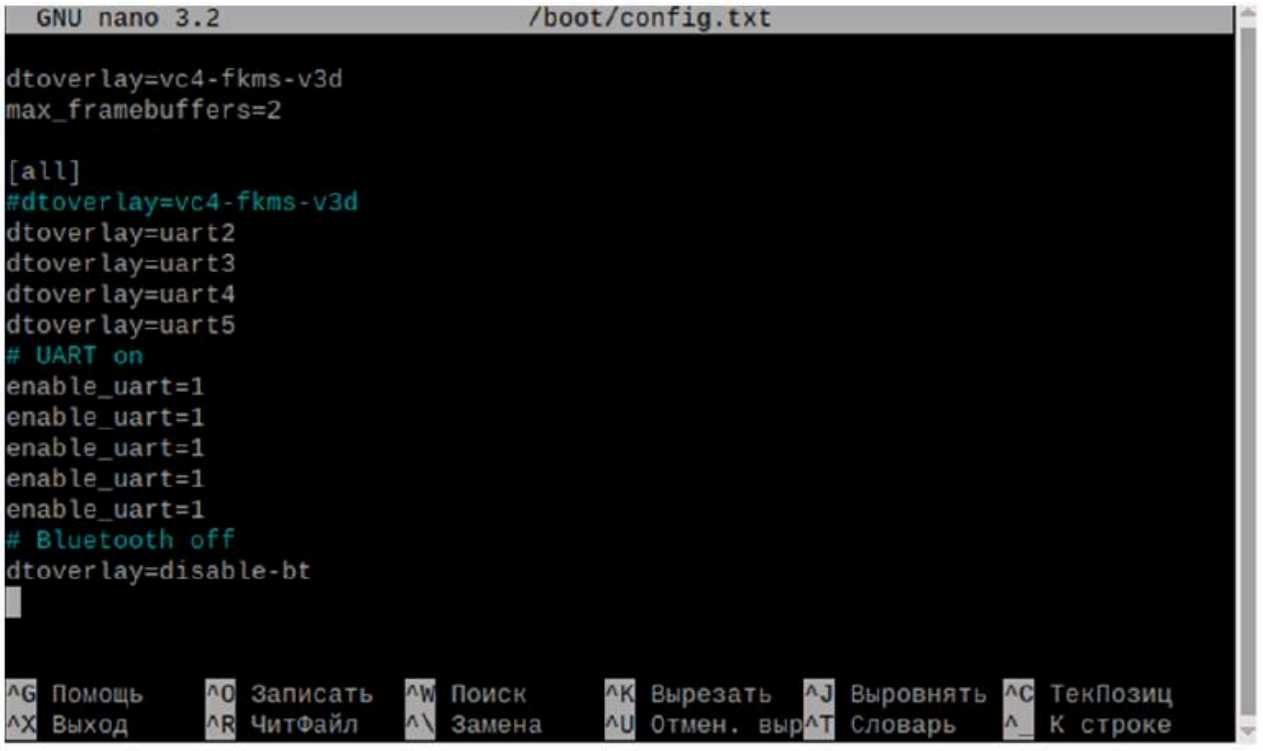

Рис. 3. Файл config.txt

Следующим шагом отключаем сервисы операционной системы, занимающие порты UART, командами sudo systemctl disable hciuart.service, sudo systemctl disable bluealsa.service, sudo systemctl disable bluetooth.service, после чего снова перезагружаем систему.

Работоспособность портов можно проверить, создав небольшой проект с помощью QT Creator для передачи данных между портами Raspberry Pi 4.

Qt Creator представляет собой современную интегрированную среду разработки (IDE) [5], ориентированную на создание кроссплатформенных приложений на языке C++ с использованием фреймворка Qt. Она поддерживает разработку как графических интерфейсов, так и консольных приложений, обеспечивая удобные инструменты для работы с классами Qt, включая

QSerialPort, который используется для организации взаимодействия через последовательные интерфейсы, такие как UART.

Одним из ключевых преимуществ Qt Creator является его кроссплатформенность: разработка приложений может вестись под Windows, Linux и macOS, а готовые программы могут быть собраны для различных аппаратных платформ, включая одноплатные компьютеры Raspberry Pi. Это делает Qt Creator удобным инструментом для встраиваемых систем и проектов в области интернета вещей.

Среда предоставляет встроенные средства редактирования кода с подсветкой синтаксиса, автодополнением и системой навигации по проекту. Важной особенностью является поддержка отладки и профилирования приложений, что позволяет анализировать как корректность работы программы, так и ее производительность.

Для проектов, связанных с последовательными интерфейсами, Qt Creator особенно полезен благодаря встроенной поддержке библиотеки Qt Serial Port. С ее помощью разработчик может открывать, настраивать и использовать UART-порты без необходимости обращаться к низкоуровневым функциям POSIX API. Это значительно сокращает время разработки и упрощает процесс тестирования обмена данными.

Создав проект, необходимо создать интерфейс приложения в файле с расширением .ui (рис. 4).

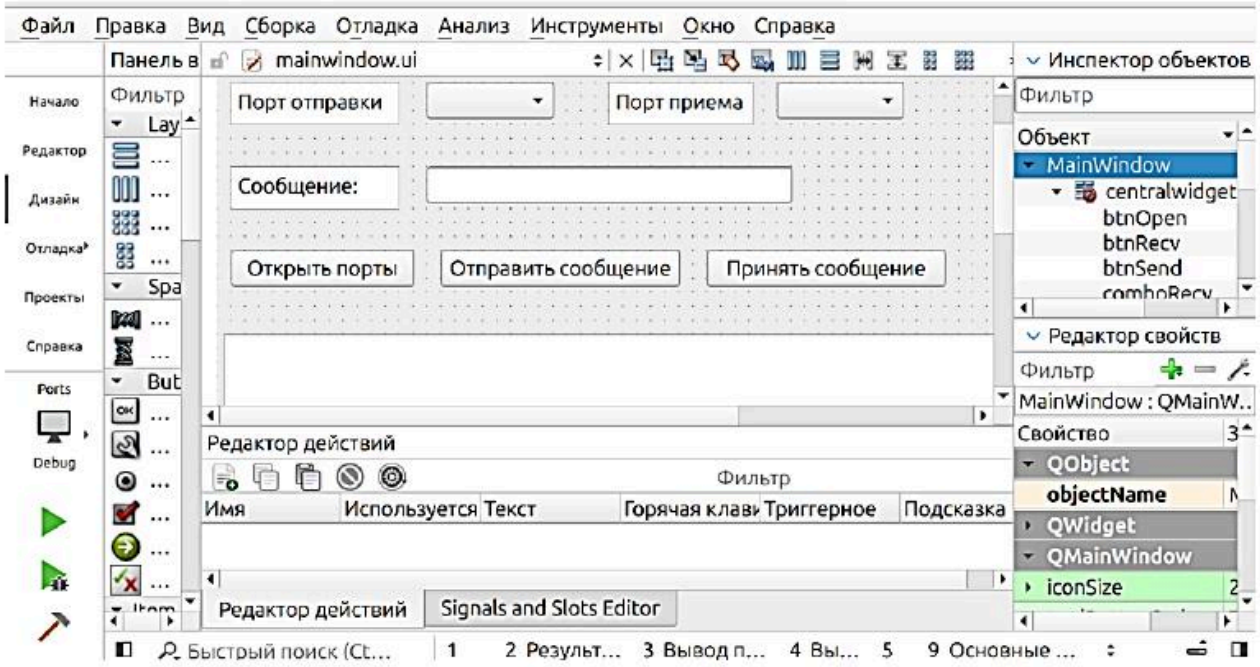

Рис. 4. Файл настройки интерфейса приложения

После создания интерфейса необходимо в файле проекта с расширением .pro подключить библиотеку Serial Port, добавив строчку QT += serialport, а в заголовочном файле .h необходимо вызвать эту библиотеку строчкой #include <QSerialPort>.

Основная логика проекта реализовывается в файле с расширение .cpp, код которого выглядит следующим образом.

Листинг. Код программы для отправки и приема данных по последовательным UART-портам

```
void MainWindow::sendMessage(){
   if (!sendPort || !sendPort->isOpen()){
     logMessage("Порт отправки не открыт");
      return;
   }
 QString text = ui->lineEdit->text();
   if (text.isEmpty()){
      logMessage("Пустое сообщение");
      return;
   }
   QByteArray data = text.toUtf8();
   sendPort->write(data);
   sendPort->flush();
   logMessage("Отправлено: " + text);
}
void MainWindow::recvMessage(){
   if (!recvPort) return;
     QByteArray data2 = recvPort->readAll();
     QString text2 = QString::fromUtf8(data2);
     logMessage("Принято: " + text2);
}
```

Результаты. Данный код запускается благодаря файлу main.cpp. По итогу разработанное приложение (рис. 5) позволяет пользователю выбрать включенные ранее порты как для отправки сообщения, так и для его приема. Введя любое сообщение в соответствующее поле, и открыв оба порта для работы, нажав соответствующую кнопку, пользователь сможет осуществить общение двух портов (рис. 6).

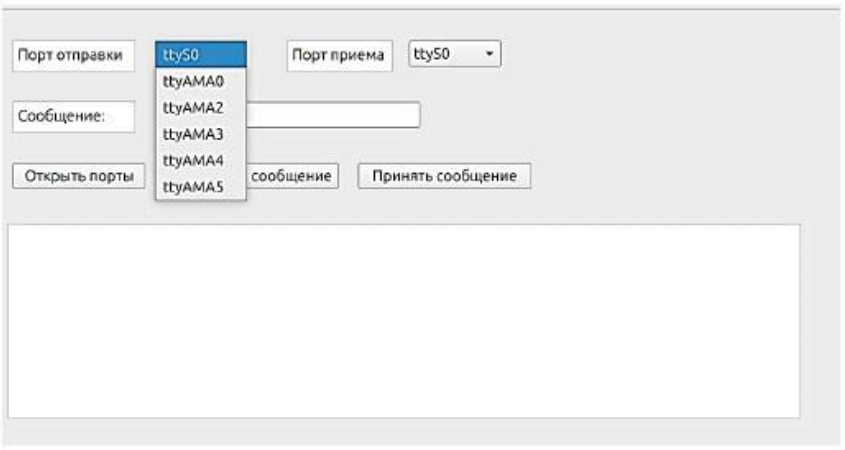

Рис. 5. Выбор включенных портов для отправки и приема данных

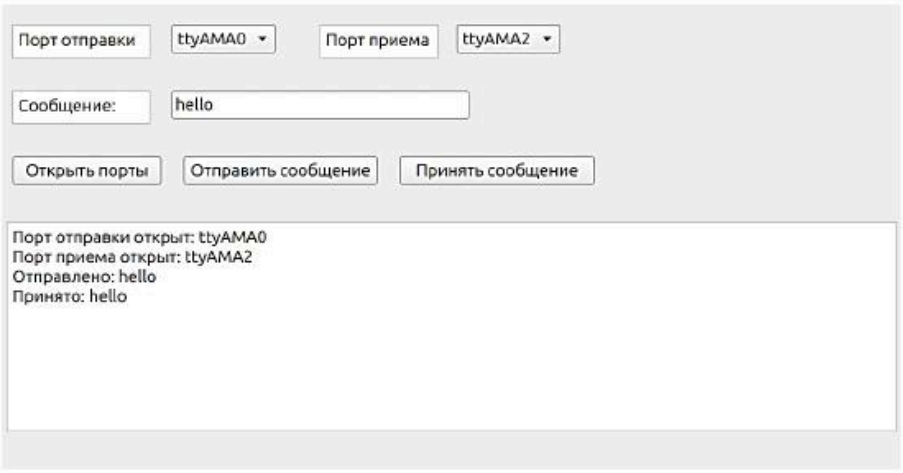

Рис. 6. Результат общения двух портов

Из результатов работы программы видно, что UART-порты настроены для общения, и их можно использовать для передачи сообщений не только между друг другом, но и с другими устройствами, что позволит создавать более сложные системы.

Заключение. В ходе работы была выполнена настройка UART-интерфейсов на Raspberry Pi 4 под управлением Ubuntu и проведена их проверка с использованием специально разработанного приложения в среде Qt Creator. Исследование показало, что последовательные порты после корректной конфигурации могут успешно применяться как для взаимодействия между собой, так и для обмена данными с внешними устройствами. Таким образом, поставленная цель достигнута, а задачи исследования выполнены: рассмотрены теоретические основы работы UART, изучена структура GPIO-разъема Raspberry Pi 4, проведена практическая настройка последовательных портов и протестирована их работоспособность. Практическая значимость работы заключается в том, что предложенная методика может использоваться студентами и инженерами для быстрой настройки UART в проектах, связанных с IoT и встроенными системами. Разработанное программное обеспечение служит удобным инструментом для отладки и проверки передачи данных.

Список источников

[1] Тарасов И.Е. *Схемотехника устройств компьютерных систем. Ч. 2.* Москва, РТУ МИРЭА, 2022, 315 с.

[2] Халфакри Г. *Raspberry Pi 4. Официальное руководство для начинающих.* Москва, ДМК Пресс, 2021, 262 с.

[3] Иванько А.Ф., Иванько М.А. *Системное программное обеспечение информационных мультимедиасистем.* Санкт-Петербург, Лань, 2020, 80 с.

[4] *Raspberry Pi 4 Model B: подключение, настройка и начало работы.* URL: https://wiki.amperka.ru/products:raspberry-pi-4-model-b?ysclid=mfaa0i2ea8786149984 (дата обращения 08.09.2025).

[5] Алексеев Е.Р., Злобин Г.Г., Костюк Д.А., Чеснокова О.В. *Программирование на языке C++ в среде Qt Creator.* Москва, ИНТУИТ, 2016, 715 с.