DOI: 10.24412/2413-2527-2023-234-10-14

Проектирование мультипроцессорного автоматизированного рабочего места для управляющей аппаратуры вагонных замедлителей ВУПЗ-15Э

к.т.н. А. А. Блюдов, Е. А. Волков, А. Ю. Идуков
Петербургский государственный университет путей сообщения Императора Александра I
Санкт-Петербург, Россия
blyudov@pgups.ru, gole00201@gmail.com

***Аннотация*. Описан процесс и представлены результаты разработки программы для удаленного контроля и настройки воздухозаборника тормозной позиции ВУПЗ-15Э на станции Санкт-Петербург-Сортировочный-Московский. Главная задача разработки заключается в обеспечении возможности получения и обработки диагностической информации от тормозной позиции с использованием канала RS-485, а также предоставлении этой информации сотрудникам бригады автоматического регулирования скорости.**

***Ключевые слова:* железнодорожный транспорт, сортировочная горка, графический интерфейс, тормозная позиция, разработка программы.**

Введение

Воздухосборник ВУПЗ-15Э предназначен для дистанционного электропневматического управления потоком сжатого воздуха между компрессорной, вагонным замедлителем и атмосферой, поступающим к воздухосборнику ВУПЗ-15Э по пневмомагистрали с компрессорной. Управление может осуществляться оператором непосредственно с панели управления БУК ЭП-М ведущего воздухосборника ВУПЗ-15Э или дистанционно с рабочего места за пультом оператора из кабины наблюдения или с помощью аппаратуры автоматического управления, размещенной на горочном посту сортировочной горки [1].

В данный момент для настройки и контроля за состоянием замедлителей на четвертой сортировочной горке станции Санкт-Петербург-Сортировочный-Московский сотрудникам бригады АРС необходимо вмешиваться в работу устройства и при этом находиться в непосредственной близости с проходящими отцепами. Данная процедура подвергает сотрудников опасности и в случае нарушения технологии проведения работ может привести к опасным последствиям [2].

Для исключения подобных ситуаций сотрудниками четвертой горки было выдвинуто предложение разработать программный продукт (ПП) в качестве дополнения к действующему автоматизированному рабочему месту механиков АРС, чтобы иметь возможность с поста централизации контролировать состояние замедлителей и при необходимости настраивать их параметры.

Основными требованиями к ПП являются: реализация с использованием оконных интерфейсов, кроссплатформенность, реализация УГИ в соответствии с руководствами ОАО «РЖД», масштабируемость.

Особенности протокола связи с ВУПЗ-15Э

В данном проекте решается задача дистанционного контроля и настройки воздухозаборника ВУПЗ-15Э с помощью подключения его к автоматизированному рабочему месту электромеханика АРС по каналу RS-485.

Для решения этих задач используется 2-проводной интерфейс RS-485.

Для настройки и контроля за состоянием тормозной позиции разработчиками ВУПЗ-15Э предусмотрена возможность взаимодействия с блоком управления при помощи отправки специальных строк в кодировке UTF-8.

Формат запроса имеет следующий вид. Структура команды: bmk:XXX:command, где XXX — номер БУК ЭП-М, command — команда для получения/установки параметров. Каждая строка в посылах запросов и ответов заканчивается <0D 0A>.

Основные запросы:

- bmk:XXX:getCount — запрос счетчиков срабатывания катушек и ступеней торможения;
- bmk:XXX:getDelta — запрос коэффициентов, для расчета скорости нарастания и спада давления;
- bmk:XXX:getStatus — запрос общих параметров системы;
- bmk:XXX:gPr — запрос давления.

В ответ на отправляемую команду управляющая аппаратура тормозной позиции ответит строкой о текущем состоянии. Так, например, строка ответа на запрос getStatus выглядит следующим образом: «bmk=009 bmkS=007 bmkSK=2 pr=000 pr0=000 pr1=000 temp=+232 P05=064 P10=125 P15=219 P20=316 P25=401 P30=489 P35=581 Err=00000000 uPit=23 temHeart=+05 timeW=00003053 prAtmCal0=+00 prAtmCal1=+00 Styp=00 l=000 temp2=+242 timeR=000006 cs=114». Суммарно в этой строке содержится 232 байта в кодировке UTF-8.

Рассмотрим ситуацию, когда необходимо опросить все 13 средних тормозных позиций командой bmk:XXX:getStatus.

Учитывая битрейт передачи в 38 400 бит в секунду, передача этой строки займет 0,04 секунды, что приводит к слишком большому времени опроса всей средней тормозной позиции. Получается, что каждая тормозная позиция будет опрашиваться в лучшем случае раз в полсекунды. Этого недостаточно для того, чтобы считать данный опрос опросом в реальном времени [3]. Исправить возникшую ситуацию мог бы измененный протокол передачи данных,

не требующий продвинутого парсинга строк. Так, например, можно представлять данные о состояниях в виде массива или Си-структуры, где каждый индекс (или поле) обозначал бы конкретный атрибут состояния тормозной позиции, полученный в результате исполнения конкретной команды.

Разработка исходного кода

В качестве парадигмы программирования была выбрана парадигма функционального программирования. Для оптимизации и ускорения процесса разработки основным инструментом стал язык программирования Python версии 3.11 в комплексе с графической библиотекой *Dear PyGui*. Для компиляции программы в бинарный исполняемый формат использовался Python-модуль *PyInstaller*.

Подразумевается разработка двух версий приложения: Windows-совместимый вариант и GNU Linux-совместимый вариант. Было решено не прибегать к кросс-компиляции и для сборки приложения использовать две сепарированные машины с Windows 10 и GNU Linux на них.

Для реализации связи с управляющей аппаратурой тормозной позиции было принято решение использовать конвертацию сигнала RS-485 в USB с помощью конвертера интерфейсов ARC-485, основным преимуществом которого является наличие защиты от перенапряжений до 1 000 В [4]. Для связи с COM-портом была выбран Python-модуль *serial* и методы из него: .readline() и .write().

Для ускорения процесса приема сообщений от управляющей аппаратуры тормозной позиции было принято решение реализовывать логику графической составляющей программы в одном процессе, а связь и обмен сообщениями с управляющей аппаратурой тормозной позиции — в другом. Для реализации многопроцессорности был выбран Python-модуль *multiprocessing*, а средством межпроцессорного взаимодействия — *multiprocessing.PIPE*.

Корень проекта содержит в себе два модуля: *backend*, *gui*. Модуль *backend* в свою очередь содержит в себе три сепарированных модуля: *serial*, *logs*, *parser*.

Модуль *serial* решает вопросы коммуникации между рабочим местом электромеханика и тормозной позиции и отправки ответов в *multiprocessing.PIPE*. Модуль *logs* берет на себя задачи по логированию принятых и отправленных сообщений. Модуль *parser* реализует парсинг принимаемых от управляющей аппаратуры тормозной позиции сообщений с последующей генерацией словаря dict_to_return: dict[str, str], где ключом является имя параметра, а значением — показатель данного параметра, принятый от тормозной позиции. Сам парсер представляет из себя достаточно простую функцию, код которой представлен ниже.

```
def parse_string(string, params):
    read_string = string.replace(' = ', '  ')
    tokens= read_string.split( )
    if tokens:
        if tokens[0] == 'bmk' and tokens[len(tokens)-2] == 'cs':
            for token in tokens:
                if token in params.keys( ):
                    params [token] = tokens [tokens.index (token) + 1]
            if 'default' in params.values( ):
                return False
            return params
        return False
```

В результате имеем весьма эффективный парсер строки, который выполняет парсинг за $O(N)$ операций, где N — число аргументов строки в требуемом словаре. На этих двух модулях и базируется все разработанное приложение.

Прием и отправка сообщений реализуется в функции *Serial.send_command(**args)*.

```
def send_command(commandToSend, comPort):
    dictToWrite = { }
    for _ in range (ATTEMPTS):
        if comPort.write (commandToSend.encode( ) ):
            line = comPort.readline( )
            commandToSend_name = f ' {commandToSend[8:]} '
            dictToWrite[commandToSend_name] = \
                parser.parse_com_str (line, commandToSend_name)
            if dictToWrite[commandToSend_name]:
                logs.success_parsing_log(str(line))
                break
            else:
                logs.error_parsing_log(str(line))
                continue
        else:
            logs.error_write_log(commandToSend)
            continue
    return dictToWrite
```

Задачи по отрисовке графики возложены на модуль Python *Dear PyGui*, который отличается достаточно простыми и понятными методами по реализации GUI, а также, что очень важно, он кросс-платформенный.

Реализация всей логики графического интерфейса и наполнения окон находится в функции, код которой представлен ниже.

```
def show_bmk(q: Any) -> None:
    global current_buks_list
    if not q.empty( ):
        params_dict = q.get_nowait( )
        current_bmk: str = params_dict[ 'bmk' ]
        try:
            if params_dict[ 'data' ][ 'getStatus\r\n' ]:
                if current_bmk not in current_buks_list:
                    current_buks_list.append(current_bmk)
                    redraw_bmk_window(params_dict)
                redraw_data(params_dict)
                refresh_pr(params_dict)
                manage_error_in_get_status(current_bmk, params_dict)
                dpg.set_item_user_data(f "bmk_{current_bmk}", params_dict)
            else:
                current_buks_list = find_false(current_bmk, current_buks_list)
        except KeyError:
            try:
                if params_dict[ 'data' ][ 'gPr\r\n' ]:
                    redraw_pr_plot(params_dict[ 'data' ][ 'gPr\r\n' ])
                else:
                    dpg.delete_item(f 'line_{current_bmk}')
                    dpg.draw_line(parent=f "BMK: {current_bmk}", p1=(0, 50),
                                  p2=(150, 50), thickness=4,
                                  color=(255, 0, 255, 255),
                                  tag=f 'line_{current_bmk}')
            except KeyError:
                pass
    blinker(current_buks_list)
```

Данная функция запускается в цикле while True и отрисовывает состояние всех тормозных позиций и дополнительных окон. Отрисовка происходит на основе словаря, получаемого из *multiprocessing.PIPE*. Из этого словаря вычитывается значение поля «bmk» и относительно него выбирается, какую позицию необходимо отрисовать в этом такте. Перечень основных функций следующий:

- redrawBmkWindow — перерисовка основного окна тормозной позиции;
- redrawData — перерисовка таблицы значений, если такая сейчас открыта;
- refreshPr — перерисовка значений текущего давления тормозной позиции;
- manageErrorInGetStatus — обработка ошибок самодиагностики в принятом сообщении от управляющей аппаратуры тормозной позиции;
- redrawPrPlot — обновление графика с давлением тормозной позиции, если таковой сейчас открыт;
- blinker — моргание элементов окна тормозной позиции, если это сейчас необходимо.

Элементы графического интерфейса и цветовая палитра

При выборе цветовой палитры основным руководством был стандарт [5]. Выбранная цветовая палитра представлена в таблице 1.

Таблица 1

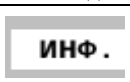
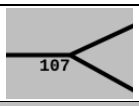
Цветовая палитра графического интерфейса

Назначение	**Цвет**	**Код RGB**
Исправная тормозная позиция		(0, 0, 255)
Неисправная тормозная позиция		(255, 0, 0)
Ошибка в канале связи		(255, 0, 255)
Путь на схематическом плане, контур кнопки		(0, 0, 0)
Фон окна		(191, 191, 191)
Кнопка		(255, 255, 255)
Активная кнопка		(0, 255, 0)

В соответствии с выбранной цветовой палитрой и стандартом [6] были разработаны представления тормозной позиции, представленные в таблице 2.

Таблица 2

Элементы графического интерфейса

Элемент	**Вид**
Кнопка	
Активная кнопка	
Путь	
Стрелка	
Горочный пост	
Часы и название участка	
Тормозная позиция исправная	
Тормозная позиция неисправная	
Неисправность линии связи	

Внешний вид программы представлен на рисунке 1.

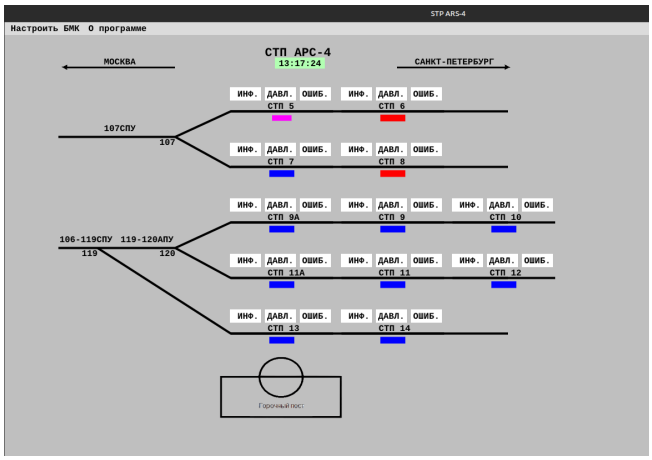


Рис. 1. Итоговый графический интерфейс

Заключение

Выполнение данной работы позволило успешно решить поставленную задачу по разработке программы для удаленной настройки и контроля воздухозаборника тормозной позиции ВУПЗ-15Э.

Разработанная программа имеет потенциал для дальнейшего применения на других станциях, где используются подобные вагонные замедлители. Ее внедрение может значительно улучшить эффективность и безопасность процессов обслуживания, а также сократить затраты на регламентное обслуживание [7].

Литература

1. Технологии передачи данных в современных системах релейной защиты и автоматики и их показатели качества / И. Н. Лизунов, А. Н. Васев, Р. Ш. Мисбахов, [и др.] // Известия высших учебных заведений. Проблемы энергетики. 2017. Т. 19, № 1–2. С. 52–63.

2. Хорошев, В. В. Повышение отказоустойчивости устройств автоматического роспуска составов на железнодорожных сортировочных горках с помощью непрерывного мониторинга // Автоматика на транспорте. 2018. Т. 4, № 3. С. 355–379.

3. Шершень, С. А. Разработка метода опроса устройств на основе стандарта RS-485 для систем реального времени / С. А. Шершень, Е. Е. Бизянов // Вестник кибернетики. 2021. № 2 (42). С. 31–37.
DOI: 10.34822/1999-7604-2021-2-31-37.

4. Сергеев, А. В. Защита оборудования АСУ от перенапряжений со стороны линии RS-485 // Территория «Нефтегаз». 2012. № 9. С. 14–17.

5. ГОСТ Р 52870-2007. Средства отображения информации коллективного пользования. Требования к визуальному отображению информации и способы измерения = Image of information means for collective use. Requirements for visual image of information and measurement methods: национальный стандарт Российской Федерации: утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 27 декабря 2007 года № 530-ст: дата введения 2009-01-01. — Москва: Стандартинформ, 2008. — 27 с.

6. СТО РЖД 1.19.005-2008. Системы и устройства железнодорожной автоматики и телемеханики. Условные графические изображения: стандарт ОАО «РЖД»: утвержден и введен в действие распоряжением ОАО «РЖД» от 30 декабря 2008 года № 2881р: дата введения 2009-02-01. — Москва: ОАО «РЖД», 2008. — 36 с.

7. Автоматизированное рабочее место проверки параметров сложных систем / С. В. Добровольский, А. М. Чирухин, И. П. Куропатка, В. А. Буряк // Известия Таганрогского государственного радиотехнического университета. 2003. № 3 (32). С. 145–150.

DOI: 10.24412/2413-2527-2023-234-10-14

Designing of the Multiprocessor Workstation for the VUPZ-15E Car Decelerators Control Equipment

PhD A. A. Blyudov, E. A. Volkov, A.Yu. Idukov
Emperor Alexander I St. Petersburg State Transport University
Saint Petersburg, Russia
blyudov@pgups.ru, gole00201@gmail.com

Abstract. **This article describes the process and presents the results of the development of a program for remote monitoring and tuning of the air intake of the VUPZ-15E brake position at the St. Petersburg-Sortirovochny-Moskovsky station. The main task of the development is to ensure the possibility of receiving and processing diagnostic information from the braking position using the RS-485 channel, as well as providing this information to the employees of the automatic speed control brigade.**

Keywords: **railway transport, sorting slide, graphical interface, braking position, program development.**

REFERENCES

1. Lizunov I. N., Vasev A. N., Misbahov R. Sh., et al. Technologies of Data Transfer in Modern Relay Protection Systems and Automatics and Their Quality Indicators [Tekhnologii peredachi dannykh v sovremennykh sistemakh releynoy zashchity i avtomatiki i ikh pokazateli kachestva], *Power Engineering: Research, Equipment, Technology [Izvestiya vysshikh uchebnykh zavedeniy. Problemy energetiki]*, 2017, Vol. 19, No. 1–2, Pp. 52–63.

2. Khoroshev V. V. Improvement of Fault Tolerance of Automatic Reforming Cars Equipment on Railroad Yards for the Continuous Monitoring [Povyshenie otkazoustoychivosti ustroystv avtomaticheskogo rospuska sostavov na zheleznodorozhnykh sortirovochnykh gorkakh s pomoshchyu nepreryvnogo monitoringa], *Transport Automation Research [Avtomatika na transporte]*, 2018, Vol. 4, No. 3, Pp. 355–379.

3. Shershen S. A., Bizyanov E. E. Development of Device Polling Method Based on the RS-485 Standard for Real-Time Systems [Razrabotka metoda oprosa ustroystv na osnove standarta RS-485 dlya sistem realnogo vremeni], *Proceedings in Cybernetics [Vestnik kibernetiki]*, 2021, No. 2 (42), Pp. 31–37. DOI: 10.34822/1999-7604-2021-2-31-37.

4. Sergeev A. V. Protection of the Automated Control System Equipment from Overvoltage from the RS-485 Line [Zashchita oborudovaniya ASU ot perenapryazheniy so storony linii RS-485], *Neftegaz Territory [Territoriya «Neftegaz»]*, 2012, No. 9, Pp. 14–17.

5. GOST R 52870-2007. Image of information means for collective use. Requirements for visual image of information and measurement methods [Sredstva otobrazheniya informatsii kollektivnogo polzovaniya. Trebovaniya k vizualnomu otobrazheniyu informatsii i sposoby izmereniya]. Effective from January 01, 2009. Moscow, StandartInform Publishing House, 2008, 27 p.

6. STO RZD 1.19.005-2008. Systems and devices of railway automation and telemechanics. Conditional graphic images [Sistemy i ustroystva zheleznodorozhnoy avtomatiki i telemekhaniki. Uslovnye graficheskie izobrazheniya]. Effective from February 01, 2009. Moscow, JSC Russian Railways, 36 p.

7. Dobrovolsky S.V., Chirukhin A. M., Kuropatka I. P., Buryak V. A. Automated Workplace for Checking the Parameters of Complex Systems [Avtomatizirovannoe rabochee mesto proverki parametrov slozhnykh sistem], *Izvestiya Taganrog State University Radio Engineering [Izvestiya Taganrogskogo gosudarstvennogo radiotekhnicheskogo universiteta]*, 2003, No. 3 (32), Pp. 145–150.