МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ВСЕРОССИЙСКАЯ ГРУППА ТЕОРИИ ИНФОРМАЦИИ IEEE
АКАДЕМИЯ ИНФОРМАТИЗАЦИИ ОБРАЗОВАНИЯ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ООО «ОТКРЫТЫЕ РЕШЕНИЯ»
ОБЩЕСТВО «ЗНАНИЕ» РОССИИ
ПРИВОЛЖСКИЙ ДОМ ЗНАНИЙ

XXII Международная
научно-техническая конференция

ПРОБЛЕМЫ ИНФОРМАТИКИ В ОБРАЗОВАНИИ, УПРАВЛЕНИИ, ЭКОНОМИКЕ И ТЕХНИКЕ

Сборник статей

Декабрь 2022 г.

Пенза

УДК 004
ББК 32.81я43+74.263.2+65.050.2я43
П781

П781 **ПРОБЛЕМЫ ИНФОРМАТИКИ В ОБРАЗОВАНИИ, УПРАВЛЕНИИ, ЭКОНОМИКЕ И ТЕХНИКЕ :**
сборник статей XXII Международной научно-технической конференции. – Пенза: Приволжский Дом знаний, 2022. – 356 с.

ISBN 978-5-8356-1800-2
ISSN 2311-0406

Под редакцией ***В.И. Горбаченко,*** доктора технических наук, профессора;
В.В. Дрождина, кандидата технических наук, профессора

Информация об опубликованных статьях предоставлена в систему Российского индекса научного цитирования (РИНЦ) по договору № 573-03/2014К от 18.03.2014.

ISBN 978-5-8356-1800-2
ISSN 2311-0406

XXII International
scientific and technical conference

PROBLEMS OF INFORMATICS IN EDUCATION, MANAGEMENT, ECONOMICS AND TECHNICS

December, 2022

Penza

6. Правила стрельбы и управления огнем артиллерии. Дивизион, батарея, взвод, орудие (ПС и УО - 96). Часть 1. – М.: Воениздат, 1996.

Ханжин Евгений Вячеславович
Военная академия материально-технического обеспечения им. генерала армии А.В. Хрулева,
филиал в г. Пензе, Россия

KhanzinE.V.
Military Academy Logistics them. Army General A.V. Khruleva, branch in Penza, Russia

---

УДК 004.9
ГРНТИ 143507

РАЗРАБОТКА ВНЕШНИХ КОМПОНЕНТ 1С С ИСПОЛЬЗОВАНИЕМ NATIVE API

А. П. Шутков, Н. Г. Яковлева

DEVELOPMENT OF EXTERNAL COMPONENTS 1C USING NATIVE API

A. P. Shutkov, N. G. Yakovleva

Аннотация. Рассматриваются методы создания внешних компонент на базе 1С:Предприятие 8 с использованием Native API на примере разработки компоненты сбора данных от подключаемого оборудования по COM-порта.

Ключевые слова: 1С Предприятие, внешние компоненты, Arduino, Native API.

Abstract. Methods for creating external components based on 1C:Enterprise 8 using the Native API are considered using the example of developing a component for collecting data from connected equipment via a COM port.

Key words: 1C Enterprise, external components, Arduino, Native API.

Платформа 1С 8.3 предназначена для решения широкого спектра задач автоматизации организационной деятельности, предоставляя широкие возможности настройки системы на особенности обработки информации в заданных условиях. Однако в ряде случаев встроенного функционала не достаточно для решения поставленных задач, тогда разработчикам

приходится прибегать к механизму внешних компонент, которые могут быть подключены как к серверу приложения 1С:Предприятия, так и клиентским приложениям, в т.ч. и веб-клиенту.

Рассматривается процесс написания компоненты получения данных от микроконтроллера Arduino посредством использования механизма NativeAPI. Использование этого механизма обосновано отсутствием поддержки технологии COM в linux-версиях дистрибутивов 1С Предприятия.

Структура кода, требования и ограничения к написанию интерфейсов механизма внешних компонент достаточно подробно представлены в официальной документации на сайте 1С [1], поэтому основное внимание сосредотачивается на описании механизма их взаимодействия с основным телом компоненты.

```
class TTYADDIN_EXPORT TtyAddin : public Component
{
public:
   const char *Version = u8"1.0.0";
explicitTtyAddin();
     //наименование класса выполнения основного тела программы
std::stringextensionName() override;
private:
std::shared_ptr<variant_t>sample_property;
};

#include "ttyaddin.h"
std::string TtyAddin::extensionName() {
     // Класс, передающийсяв 1с
return "TTY";
}
TtyAddin::TtyAddin()
{
   //УНиверсальнаятипизация
sample_property = std::make_shared<variant_t>();
AddProperty(L"SampleProperty",  L"ОбразецСвойства",  sample_property);
// Полный пакет регистрационных компонент
AddProperty(L"Version", L"ВерсияКомпоненты", & {
auto s = std::string(Version);
returnstd::make_shared<variant_t>(std::move(s));
});
}
```

Передача класса в процедуры NativeAPI:

```
const WCHAR_T *GetClassNames() {
   // Might contain multiple class names seperated by |
static char16_t cls_names[] = u"TTY";
   return reinterpret_cast<WCHAR_T *>(cls_names);
}

longGetClassObject(const WCHAR_T *clsName, IComponentBase **pInterface) {
if (!*pInterface) {
autocls_name = std::u16string(reinterpret_cast<const char16_t *>(clsName));
      if (cls_name == u"TTY") {
         *pInterface = new TTYAddIn;
      }
      return (long) *pInterface;
}
return 0;
}
```

В качестве примера данные получаются через интерфейсы rs-232, rs-422, rs-485. Для этого используется стандартная библиотека termios.h. Необходимо реализовать методы «Подключения порта», «Настройки порта», «Отключения порта», «Передачи данных», «Приёма данных». Подключение и настройка реализованы одним методом ConnectPort, отключение - DisconnectPort, чтение и запись - ReadPort и SendToPort. Для обмена данными используется стандартная функция ExternalEvent, для чего переопределяются методы ExternalEvent, CleanEventBuffer и GetEventBufferDepth.

```
class TTYADDIN_EXPORT TtyAddin : public Component
{
public:
   const char *Version = u8"1.0.0";
explicitTtyAddin();
      // объявление инициализации порта и подключение к нему
variant_tConnectPort (constvariant_t&, constvariant_t&);
//объявление отключения от порта
     voidDisconnectPort(void);
   //объявление чтения буфера порта
     voidReadPort ();
     //объявление отправки строки в порт
```

```
    voidSendToPort (constvariant_t&);
    std::stringextensionName() override;
    private:
      // переопределение унаследованных методов
    boolExternalEvent(conststd::string &, const std::string &, const std::string
&);
      bool SetEventBufferDepth(long);
    longGetEventBufferDepth();
    intserial_port;    // переменная для хранения дескрипторапорта
    structtermiostty;  // структура данных настроек порта
    charread_buf [256]; // буфер временного хранения принятых данных
    //std::string str;
    std::stringPortNameStr;// хранение пути к порту
    std::shared_ptr<variant_t>sample_property; // наименование класса, ко-
торое будет передано в 1с
    };

    TtyAddin::TtyAddin()
    {
    sample_property = std::make_shared<variant_t>();
    AddProperty(L"SampleProperty",  L"ОбразецСвойства",  sample_prop-
erty);
    AddProperty(L"Version", L"ВерсияКомпоненты", [&]() {
    auto s = std::string(Version);
    returnstd::make_shared<variant_t>(std::move(s));
    });
      // Регистрация методов
      // Названия методов в 1С и связанные с методы классы
    AddMethod(L"Connect",  L"ПодключитьПорт",  this,  &TtyAddin::Con-
nectPort);
    AddMethod(L"Disconnect",  L"ОтключитьПорт",  this,  &TtyAddin::Dis-
connectPort);
    AddMethod(L"Read", L"ЧитатьПорт", this, &TtyAddin::ReadPort);
    AddMethod(L"Send",  L"ОтправитьВПорт",  this,  &TtyAddin::SendTo-
Port);
    }
    // определение метода инициализации порта и подключение к нему
    // принимается строка пути к файлу порта и скорость порта (число)
    // в случае успеха возвращается истина
```

```
variant_tTtyAddin::ConnectPort (constvariant_t&serialPortName, con-
stvariant_t& Baud)
{   variant_tres = false;
   //проверка на соответствие типов
if (std::holds_alternative<std::string>(serialPortName)
                && (std::holds_alternative<int32_t>(Baud)))
{  res = true;
                //помещается путь к порту в строку: пример
"/dev/ttyACM0"
PortNameStr = std::get<std::string>(serialPortName);
// преобразуется в строку (массив char) и сохраняется указатель на нее
     const char *PortName=PortNameStr.c_str();
//открывается указанный порт для чтения и записи
       //(только для чтения O_RDONLY)
     serial_port = open(PortName, O_RDWR);
if (serial_port< 0) {
throwstd::runtime_error(u8"указанный порт не определен в системе");
//если открыть порт не удалось
        }
switch (static_cast<int>(std::get<int32_t>(Baud)))    // в соответствии с
указанной скоростью устанавливается скорость порта
{
case 1200:cfsetspeed(&tty, B1200);
                break;
case 2400:cfsetspeed(&tty, B2400);
                break;
case 4800:cfsetspeed(&tty, B4800);
                break;
case 9600:cfsetspeed(&tty, B9600);
                break;
case 19200:cfsetspeed(&tty, B19200);
                break;
          default: res = false;
throwstd::runtime_error (u8" значение скорости недопустимо");
break;
       }
     // вводятся основные настройки
tty.c_cflag&= ~PARENB; // без паритета
tty.c_cflag&= ~CSTOPB; // 1 стоп бит
tty.c_cflag |= CS8;    // 8 бит
```

```
tty.c_cflag&= ~CRTSCTS; // без RTS/CTS аппаратного управления потоком
//сохраняются настройки
if (tcsetattr(serial_port, TCSANOW, &tty) != 0)
{ throwstd::runtime_error(u8"невозможно сохранить настройки порта");
}
memset(&read_buf, '\0', sizeof(read_buf));//инициализируется буфер
TtyAddin::SetEventBufferDepth(10); //устанавливается размер очереди событий, в 1с функция описана в 1С:ИТС
}
else{ res = false;
throwstd::runtime_error(u8"метод serialSetting - неподдерживаемые типы данных");} //если имя порта не строка, а скорость не число
returnres;
}
//определение метода отключения от порта
voidTtyAddin::DisconnectPort(void) //Отключается порт
{ close(serial_port); }
//определение метода чтения буфера порта
voidTtyAddin::ReadPort ()
{ tcflush(serial_port,TCIOFLUSH); // очистка буфера порта от мусора перед чтением
sleep(1); // ждём (секунды)
intnum_bytes = read(serial_port, &read_buf[0], sizeof(read_buf)); //читаембуферпорта
if (num_bytes<= 0) //если -1 ошибка 0 буфер пуст
{
throwstd::runtime_error(u8"данные в порт не поступают");
}
else {ExternalEvent(extensionName(), PortNameStr, static_cast<std::string>(read_buf));} // выводв 1счерезвнешнеесобытие
}
// определение метода отправки строки в порт
voidTtyAddin::SendToPort (constvariant_t& data)
{if (std::holds_alternative<std::string>(data)) // проверяетсясоответствиетипавведённыхданных
{std::stringdataString = std::get<std::string>(data); // преобразуется в строку
```

```
constchar * msg = dataString.c_str();     //и переводится в строку в стиле си
write(serial_port, msg, sizeof(msg));     // отправляется
}
else {   throwstd::runtime_error(u8"метод serialSetting - неподдерживаемые типы данных");}
}
//переопределяются проунаследованные методы
boolTtyAddin::ExternalEvent(conststd::string &src, const std::string &msg, const std::string &data)
{   return Component::ExternalEvent( src, msg, data);}
boolTtyAddin::SetEventBufferDepth(long depth)
{   return Component::SetEventBufferDepth(depth);}
longTtyAddin::GetEventBufferDepth()
{   returnComponent::GetEventBufferDepth();}
```

После этого собирается проект и производится компиляция с целью получения файла динамической библиотеки *.so(в Windows *.dll).

Теперь, для того чтобы платформа 1с смогла взаимодействовать с компонентой, для Linux-систем необходимо скорректировать /etc/ld.so.conf, добавив в него строки include /etc/ld.so.conf.d/ TTY.so, инициализирующая библиотеку в операционной системе, и /home/delphin/.cache/fontconfig/, /home/delphin/.cache/ TTY_1.0/ открывающие ему доступ в кеш-дирректории и позволяющие работать со шрифтами системы.

Регистрация компоненты в операционной системе Windows происходит штатной обвязкой командной строки - regsvr32 ~\ TTY_1.0.dll

Для использования компоненты в ОС Linux, необходимо инициализировать ее в явном виде:

```
&НаКлиенте
Процедура Подключить(Команда)
    ПутьКБиблиотеке="/home/<userCatalog>/TTY_1.0/TTYAddin.so";
    РезультатПодключения = ПодключитьВнешнююКомпоненту(ПутьКБиблиотеке, "libextDLib", ТипВнешнейКомпоненты.Native);
    Сообщить ("Компонента подключена - " + РезультатПодключения );
    Попытка
        Компонента = новый ("AddIn.libextDLib.TTY");
        Сообщить ("Компонента создана");
    Исключение
        Сообщить ("не удалось создать компоненту");
```

```
        КонецПопытки;
        Попытка
                Компонента.ПодключитьПорт ("/dev/ttyACM0", 9600);
//указан путь к файлу порта. в ОС (папка dev, файл ttyACM0)
                Сообщить ("Порт подключен");
        Исключение
                Сообщить ("неудалось подключить порт");
        КонецПопытки;
    КонецПроцедуры

    &НаКлиенте
    Процедура ПередЗакрытием(Отказ, ЗавершениеРаботы, ТекстПредупреждения, СтандартнаяОбработка)
        Компонента.ОтключитьПорт();
    КонецПроцедуры

    &НаКлиенте
    Процедура Получить(Команда)
        Компонента.ОтправитьВПорт("e");
        Компонента.ЧитатьПорт();
    КонецПроцедуры
```

Данная процедура при инициализации выведет сообщение о статусе подключения компоненты, статусе порта, источник данных (номер порта) и выведет данные, полученные с этого порта (рисунок 1).

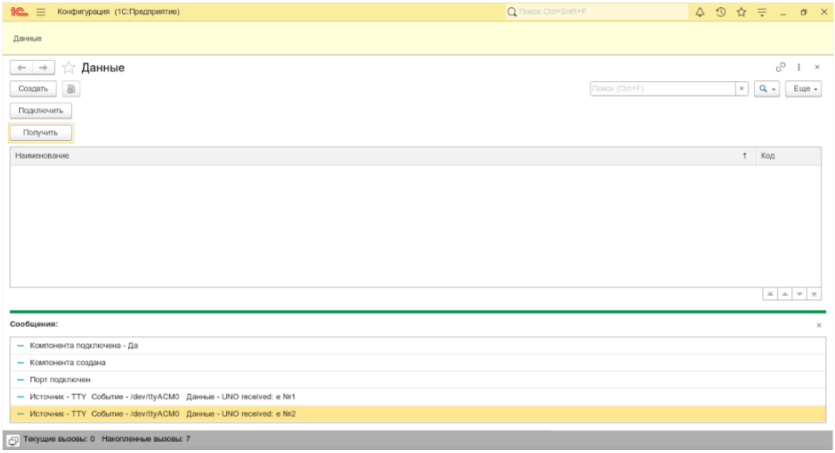

Рис. 1. Вывод информации с Arduinoв 1с

Библиографический список

1. Технология создания внешних компонент // ИТС. URL: https://its.1c.ru/db/metod8dev/content/3221/hdoc (дата обращения: 12.10.2022).

Шутков Андрей Павлович
Яковлева Наталья Геннадьевна
Тверской государственный технический университет, г. Тверь, Россия

Shutkov A. P.
Yakovleva N. G.
Tver State Technical University, Tver, Russia