ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

им. А. Н. ТИХОНОВА

**«Системы контроля и управления»**

***УЧЕБНО – МЕТОДИЧЕСКОЕ ПОСОБИЕ***

***ПРАКТИКУМ***

**Москва 2025**

**Оглавление**

[1 Знакомство с QtCreator 5](#_Toc187248327)

[1.1 Цели работы 5](#_Toc187248328)

[1.2 Краткие теоретические сведения 5](#_Toc187248329)

[1.2.1 Общие сведения о фреймворке 5](#_Toc187248330)

[1.2.2 Элементы графического интерфейса и их использование 6](#_Toc187248331)

[1.2.3 QTimer 14](#_Toc187248332)

[1.2.4 Объектная модель Qt. Механизм сигналов и слотов 15](#_Toc187248333)

[1.3 Порядок выполнения работы 20](#_Toc187248334)

[1.4 Задания…. 21](#_Toc187248335)

[2 Интернационализация приложения 27](#_Toc187248336)

[2.1 Цели работы 27](#_Toc187248337)

[2.2 Краткие теоретические сведения 27](#_Toc187248338)

[2.3 Порядок выполнения работы 28](#_Toc187248339)

[2.4 Задание…. 29](#_Toc187248340)

[3 Работа с XML 34](#_Toc187248341)

[3.1 Цели работы 34](#_Toc187248342)

[3.2 Краткие теоретические сведения 34](#_Toc187248343)

[3.2.1 Описание формата XML 34](#_Toc187248344)

[3.2.2 Работа с xml-файлами в Qt 35](#_Toc187248345)

[3.2.3 Чтение XML файла 38](#_Toc187248346)

[3.3 Задание…. 42](#_Toc187248347)

[4 Работа с последовательными портами 49](#_Toc187248348)

[4.1 Цели работы 49](#_Toc187248349)

[4.2 Краткие теоретические сведения 49](#_Toc187248350)

[4.2.1 Описание последовательного порта 49](#_Toc187248351)

[4.2.2 QSerialPort 50](#_Toc187248352)

[4.2.3 Настройка QT для работы с последовательным портом 51](#_Toc187248353)

[4.2.4 Руководство по созданию канала связи между приложениями через виртуальные последовательные порты 52](#_Toc187248354)

[4.3 Порядок выполнения работы 55](#_Toc187248355)

[4.4 Задания…. 55](#_Toc187248356)

[5 Программирование последовательных портов микроконтроллеров 58](#_Toc187248357)

[5.1 Цели работы 58](#_Toc187248358)

[5.2 Краткие теоретические сведения 58](#_Toc187248359)

[5.2.1 Создание проекта в Proteus 58](#_Toc187248360)

[5.3 Порядок выполнения работы 61](#_Toc187248361)

[5.3.1 Настройка первой программы 61](#_Toc187248362)

[5.3.2 Подключение последовательного порта и установка 7-сегментных индикаторов 63](#_Toc187248363)

[5.4 Задания…. 66](#_Toc187248364)

[6 Управление ЦАП в Proteus 72](#_Toc187248365)

[6.1 Цели работы 72](#_Toc187248366)

[6.2 Краткие теоретические сведения 72](#_Toc187248367)

[6.2.1 Цифро-аналоговый преобразователь 72](#_Toc187248368)

[6.2.2 Аналогово-цифровой преобразователь 72](#_Toc187248369)

[6.3 Порядок выполнения работы 73](#_Toc187248370)

[6.3.1 Построение схемы с ЦАП в Proteus 73](#_Toc187248371)

[6.3.2 Построение схемы с АЦП в Proteus 75](#_Toc187248372)

[6.4 Задания…. 77](#_Toc187248373)

[7 Темы проектов семинара по направлению «Системы контроля и управления» 79](#_Toc187248374)

# Знакомство с QtCreator

## Цели работы

1. Ознакомиться с возможностями фреймворка Qt и элементами графического интерфейса.
2. Изучить сигнально-слотовые соединения с использованием возможностей Qt.

## Краткие теоретические сведения

### Общие сведения о фреймворке

Qt – C++ фреймворк для создания графических приложений.

При разработке используется специализированная среда – Qt Creator – полнофункциональная IDE с поддержкой автодополнения кода и т.п.

Qt дает возможность описывать пользовательские интерфейсы, используя простой формат – XML (eXtensible Markup Language).

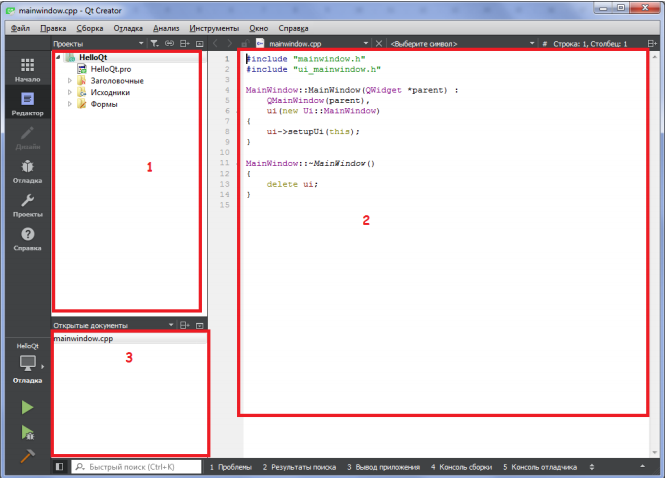


Рисунок 1 – Интерфейс программы

После создания проекта отображается окно, показанное на рисунке 1, в котором можно определить несколько функциональных областей:

1. окно «Проект», содержит все файлы, относящиеся к проекту (исходники, заголовочные, файлы ресурсов);
2. редактор кода;
3. список открытых документов.

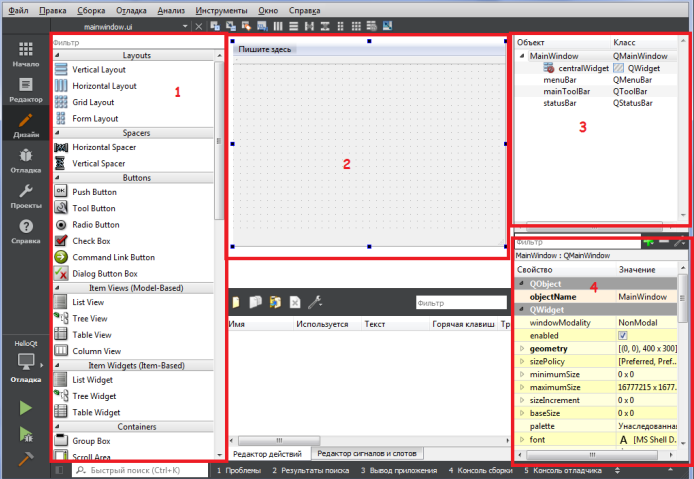


Рисунок 2 – Окно для создания форм

Создание образа окна выполняется с использованием Qt Designer. Для его вызова необходимо нажать слева кнопку «Дизайн» или выполнить двойной щелчок мыши по файлу widget.ui в навигаторе.

Внешний вид дизайнера состоит из следующих функциональных областей (рисунок 2):

1. список доступных компонентов;
2. окно редактирования формы;
3. иерархия документов на форме;
4. свойства объекта.

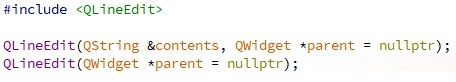
### Элементы графического интерфейса и их использование

Все виджеты в Qt наследуют от класса **QWidget**. Класс QWidget предоставляет базовую функциональность общую для всех виджетов. Среди свойств, которые наследуют от QWidget, — свойство enabled, которое позволяет разрешить или запретить взаимодействие пользователя с элементом управления (методы void setEnabled(bool) и bool isEnabled()). Когда свойство установлено (логическое значение false) — визуальный элемент деактивирован, и пользователь больше не может с ним взаимодействовать. Обычно такие деактивированные элементы изменяют внешний вид, чтобы пользователь смог их отличить от активных. Свойство visible (методы void setVisible(bool) и bool isVisible()) определяет видимый виджет (значение true) или нет (значение false). Эти свойства влияют не только на сам визуальный элемент, но и на дочерние элементы.

Кроме этих общих свойств, каждый виджет обладает собственными уникальными особенностями, которые позволяют создавать удобные и практичные в использовании пользовательские интерфейсы.

Класс **Однострочное текстовое поле** (QLineEdit) предназначено для ввода и редактирования текста небольшого объема. С его помощью можно также отобразить вводимые символы в виде звездочек (например, чтобы скрыть пароль) или вообще не отображать их (например, чтобы скрыть длину пароля). Поле по умолчанию поддерживает технологию drag & drop, стандартные комбинации клавиш быстрого доступа, работу с буфером обмена и многое другое. Однострочное текстовое поле реализуется с помощью класса **QLineEdit**. Иерархия наследования: *(QObject, QPaintDevice) – QWidget – QLineEdit.*

Конструктор класса QLineEdit имеет два формата:



В параметре parent передается указатель на родительский компонент. Если параметр не указан или имеет значение nullptr, то компонент будет обладать своим собственным окном. Параметр contents позволяет задать текст, который будет отображен в однострочном текстовом поле. Пример:



Класс QLineEdit содержит следующие методы (перечислены только основные методы, полный список смотрите в документации):

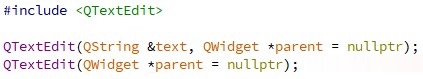
* setReadOnly() – если в качестве параметра указано значение true, то поле будет доступно только для чтения.
* isReadOnly() – возвращает значение true, если поле доступно только для чтения, и false – в противном случае.
* ext() – вставляет указанный текст в поле. Метод является слотом.
* insert() – вставляет текст в текущую позицию текстового курсора. Если в поле был выделен фрагмент, то он будет удален.
* text() – возвращает текст, содержащийся в текстовом поле.
* displayText() – возвращает текст, который видит пользователь. Результат зависит от режима отображения, заданного с помощью метода setEchoMode(). Например, в режиме Password строка будет состоять из символов \*.
* selectedText() – возвращает выделенный фрагмент или пустую строку.
* clear() – удаляет весь текст из поля. Метод является слотом.
* backspace() – удаляет выделенный фрагмент. Если выделенного фрагмента нет, то удаляет символ, стоящий слева от текстового курсора.
* del() – удаляет выделенный фрагмент. Если выделенного фрагмента нет, то удаляет символ, стоящий справа от текстового курсора.
* setSelection() – выделяет фрагмент длиной length, начиная с позиции start. Во втором параметре можно указать отрицательное значение.
* selectAll() – выделяет весь текст в поле. Метод является слотом.
* copy() – копирует выделенный текст в буфер обмена при условии, что есть выделенный фрагмент и используется режим Normal. Метод является слотом.
* paste() – вставляет текст из буфера обмена в текущую позицию текстового курсора при условии, что поле доступно для редактирования. Метод является слотом.
* undo() – отменяет последнюю операцию ввода пользователем при условии, что отмена возможна. Метод является слотом.
* redo() – повторяет последнюю отмененную операцию ввода пользователем, если это возможно. Метод является слотом.

Класс QLineEdit содержит следующие сигналы:

* cursorPositionChanged(int,int) – генерируется при перемещении текстового курсора. Внутри обработчика через первый параметр доступна старая позиция курсора, а через второй параметр – новая позиция;
* editingFinished() – генерируется при нажатии клавиши <Enter> или потере полем фокуса ввода;
* returnPressed() – генерируется при нажатии клавиши <Enter>;
* selectionChanged() – генерируется при изменении выделения;
* inputRejected() – генерируется при недопустимом вводе;
* textChanged(const QString&) – генерируется при изменении текста внутри поля пользователем или программно. Внутри обработчика через параметр доступно новое значение;
* textEdited(const QString&) – генерируется при изменении текста внутри поля пользователем. Сигнал не генерируется при изменении текста с помощью метода setText(). Внутри обработчика через параметр доступно новое значение.

Класс **Многострочное текстовое поле** (QTextEdit) предназначено для ввода и редактирования как простого текста, так и текста в формате HTML. Поле по умолчанию поддерживает технологию drag & drop, стандартные комбинации клавиш быстрого доступа, работу с буфером обмена. Многострочное текстовое поле реализуется с помощью класса **QTextEdit**. Иерархия наследования: (QObject, QPaintDevice) – QWidget – QFrame – QAbstractScrollArea – QTextEdit.

Конструктор класса QTextEdit имеет два формата:



В параметре parent передается указатель на родительский компонент. Если параметр не указан или имеет значение nullptr, то компонент будет обладать своим собственным окном. Параметр text позволяет задать текст в формате HTML, который будет отображен в текстовом поле.

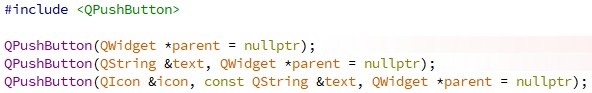
Класс QTextEdit содержит следующие методы (перечислены только основные методы; полный список смотрите в документации):

* setText() – вставляет указанный текст в поле. Текст может быть простым или в формате HTML. Метод является слотом.
* setPlainText() – вставляет простой текст. Метод является слотом.
* setHtml() – вставляет текст в формате HTML. Метод является слотом.
* insertPlainText() – вставляет простой текст в текущую позицию текстового курсора. Если в поле был выделен фрагмент, то он будет удален. Метод является слотом.
* insertHtml() – вставляет текст в формате HTML в текущую позицию текстового курсора. Если в поле был выделен фрагмент, то он будет удален. Метод является слотом.
* append() – добавляет новый абзац с указанным текстом в формате HTML в конец поля. Метод является слотом.
* setDocumentTitle() – задает текст заголовка документа (для тега <title>).
* documentTitle() – возвращает текст заголовка (из тега <title>).
* toPlainText() – возвращает простой текст, содержащийся в текстовом поле.
* toHtml() – возвращает текст в формате HTML.
* clear() – удаляет весь текст из поля. Метод является слотом.
* selectAll() – выделяет весь текст в поле. Метод является слотом.
* zoomIn() – увеличивает масштаб шрифта. Метод является слотом.
* zoomOut() – уменьшает масштаб шрифта. Метод является слотом.
* cut() – копирует выделенный текст в буфер обмена, а затем удаляет его из поля при условии, что есть выделенный фрагмент. Метод является слотом.
* copy() – копирует выделенный текст в буфер обмена при условии, что есть выделенный фрагмент. Метод является слотом.
* paste() – вставляет текст из буфера обмена в текущую позицию текстового курсора при условии, что поле доступно для редактирования. Метод является слотом.
* canPaste() – возвращает true, если из буфера обмена можно вставить текст, и false – в противном случае.
* undo() – отменяет последнюю операцию ввода пользователем при условии, что отмена возможна. Метод является слотом.
* redo() – повторяет последнюю отмененную операцию ввода пользователем, если это возможно. Метод является слотом.
* setUndoRedoEnabled() – если в качестве значения указано значение true, то операции отмены и повтора действий разрешены, а если false, то запрещены.
* isUndoRedoEnabled() – возвращает значение true, если операции отмены и повтора действий разрешены, и false, если запрещены.
* createStandardContextMenu() – создает стандартное меню, которое отображается при щелчке правой кнопкой мыши в текстовом поле. Чтобы изменить стандартное меню, следует создать класс, наследующий класс QTextEdit, и переопределить метод contextMenuEvent(). Внутри этого метода можно создать свое собственное меню или добавить новый пункт в стандартное меню.
* find() – производит поиск фрагмента (по умолчанию в прямом направлении без учета регистра символов) в текстовом поле. Если фрагмент найден, то он выделяется и метод возвращает значение true, в противном случае – значение false. В необязательном параметре options можно указать комбинацию (через оператор |) следующих констант:
  + QTextDocument::FindBackward – поиск в обратном направлении, а не в прямом;
  + QTextDocument::FindCaseSensitively – поиск с учетом регистра символов;
  + QTextDocument::FindWholeWords – поиск слов целиком, а не фрагментов;

Класс **QTextEdit** содержит следующие сигналы:

* currentCharFormatChanged(const QTextCharFormat&) – генерируется при изменении формата. Внутри обработчика через параметр доступен новый формат;
* cursorPositionChanged() – генерируется при изменении положения текстового курсора;
* selectionChanged() – генерируется при изменении выделения;
* textChanged() – генерируется при изменении текста внутри поля;
* copyAvailable(bool) – генерируется при изменении возможности скопировать фрагмент. Внутри обработчика через параметр доступно значение true, если фрагмент можно скопировать, и false – в противном случае;
* undoAvailable(bool) – генерируется при изменении возможности отменить операцию ввода. Внутри обработчика через параметр доступно значение true, если можно отменить операцию ввода, и false – в противном случае;
* redoAvailable(bool) – генерируется при изменении возможности повторить отмененную операцию ввода. Внутри обработчика через параметр доступно значение true, если можно повторить отмененную операцию ввода, и false – в противном случае.

Класс **QPushButton** представляет кнопку. Данный класс наследуется от класса QAbstractButton, который, в свою очередь, наследуется от QWidget. Для создания кнопки можно использовать один из трех конструкторов класса:



В конструкторы можно передать следующие параметры:

* parent – указатель на родительский виджет, на котором будет отображаться кнопка.
* text – текст кнопки в виде объекта QString
* icon – картинка для кнопки в виде объекта QIcon

Фреймворк Qt предоставляет набор классов, которые позволяют управлять макетом/компоновкой окна приложения и автоматически размещать вложенные виджеты по определенному принципу. Базовым классом для таких контейнеров является класс **QLayout**.

Основные классы для управления компоновкой:

* QVBoxLayout – располагает вложенные виджеты в ряд по вертикали;
* QHBoxLayout – располагает вложенные виджеты в ряд по горизонтали;
* QGridLayout – располагает в виде грида/таблицы, где каждый виджет занимает определенную ячейку;
* QFormLayout – представляет компоновку виджетов в виде формы ввода;
* QStackedLayout – представляет стек виджетов, в котором одномоменто виден только один виджет.

Класс **QLayout** содержит следующие методы (перечислены только основные методы; полный список смотрите в документации):

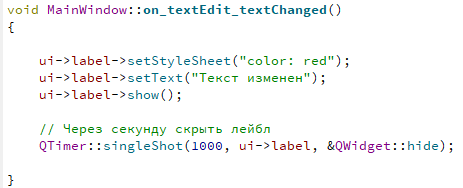
* addWidget(QWidget \*w) – добавляет новый виджет в контейнер
* count() – возвращает количество элементов в контейнере
* itemAt(int index) – возвращает элемент по определенному индексу;
* parentWidget() – возвращает родительский виджет;
* removeWidget(QWidget \*widget) – удаляет виджет из контейнера;
* replaceWidget() – заменяет один виджет другим;
* setAlignment(QWidget \*w, Qt::Alignment alignment) – устанавливает выравнивание;
* setContentsMargins(int left, int top, int right, int bottom) – устанавливает внешние отступы;
* setSpacing(int) – устанавливает размер пространства между вложенными виджетами;
* sizeConstraint() – устанавливает ограничения по размеру;
* spacing() – возвращает размер пространства между вложенными виджетами
* takeAt(int index) – возвращает элемент по индексу.

### QTimer

Класс QTimer унаследован от QObject, а значит умеет работать с сигналами и слотами. Для использования нужно подключить: #include <QTimer>.

Таймер запускается с помощью QTimer::start(). Остановить таймер можно вызовом QTimer::stop(). При этом класс имеет статический метод singleShot(), он посылает сигнал через заданный промежуток времени лишь один раз. События таймера будут доставлять только тогда, когда работает цикл обработки событий.

Пример использования:



В примере по нажатию на кнопку отображается лэйбл, и через секунду он перестает быть видимым по сигналу от QTimer.

### Объектная модель Qt. Механизм сигналов и слотов

В Qt реализована концепция функций обратного вызова (callback functions) – в результате действий пользователя вызываются обычные методы класса типа void. Чтобы сопоставить код с кнопкой, необходимо передать в функцию указатель на кнопку. Элементы графического интерфейса пользователя оказываются тесно связаны с функциональными частями программы. Для обеспечения связей сообщения и методов обработки используются макросы – карты сообщений.

Механизм сигналов и слотов основан на следующих принципах:

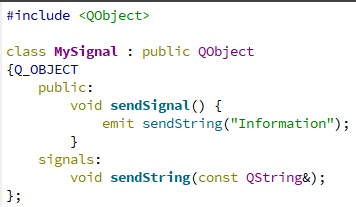
* каждый класс, унаследованный от QObject, может иметь любое количество сигналов и слотов;
* сообщения, посылаемые посредством сигналов, могут иметь множество аргументов любого типа;
* сигнал можно соединять с различным количеством слотов? При этом отправляемый сигнал поступит ко всем подсоединенным слотам;
* слот может принимать сообщения от многих сигналов, принадлежащих разным объектам;
* соединение сигналов и слотов можно производить в любой точке приложения;
* сигналы и слоты являются механизмами, обеспечивающими связь между объектами. Связь также может выполняться между объектами, которые находятся в различных потоках;
* при уничтожении объекта происходит автоматическое разъединение всех сигнально-слотовых связей. Это гарантирует, что сигналы не будут отправляться к несуществующим объектам.

Особенности работы механизма сигналов и слотов, следующие:

* сигналы и слоты не являются частью языка C++, поэтому требуется запуск дополнительного препроцессора перед компиляцией программы;
* отправка сигналов происходит медленнее, чем обычный вызов функции, который производится при использовании механизма функций обратного вызова;
* существует необходимость в наследовании класса QObject;
* в процессе компиляции не производится никаких проверок: имеется ли сигнал или слот в соответствующих классах или нет; совместимы ли сигнал и слот друг с другом и могут ли они быть соединены вместе. Об ошибке можно будет узнать лишь тогда, когда приложение будет запущено. Вся эта информация выводится на консоль, поэтому, для того чтобы увидеть ее в Windows, в проектном файле необходимо в секции CONFIG добавить опцию console.

**Сигналы** (signals) – это методы, которые в состоянии осуществлять пересылку сообщений. Сигналы определяются в классе, как обычные методы без реализации с типом void (так как не должны ничего возвращать). Всю дальнейшую заботу о реализации кода для этих методов берет на себя препроцессор. Сигнал не обязательно соединять со слотом. Если соединения не произошло, то он просто не будет обрабатываться.

Выслать сигнал можно при помощи ключевого слова **emit.** Сигналы могут отправляться из классов, которые их содержат. Сигналы также имеют возможность высылать информацию, передаваемую в параметре.

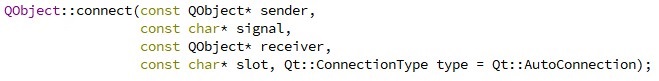


Обратите внимание, что в прототипе функции-сигнала не указываются имена параметров, а только типы.

**Слоты** (slots) – это методы, которые присоединяются к сигналам. Основное их отличие от обычных методов состоит в возможности принимать сигналы. В слотах **нельзя** использовать параметры по умолчанию, например slotMethod (int n = 8), или определять слоты как static.

Внутри слота вызовом метода sender() можно узнать, от какого объекта был выслан сигнал. Он возвращает указатель на объект типа QObject.

Соединение объектов осуществляется при помощи статического метода connect(), который определен в классе QObject, выглядит следующим образом:



Методу передаются пять следующих параметров:

1. sender – указатель на объект, отправляющий сигнал;
2. signal – это сигнал, с которым осуществляется соединение. Прототип (имя и аргументы) метода сигнала должен быть заключен в специальный макрос SIGNAL(method());
3. receiver – указатель на объект, который имеет слот для обработки сигнала;
4. slot – слот, который вызывается при получении сигнала. Прототип слота должен быть заключен в специальном макросе SLOT(method());
5. type – управляет режимом обработки. Имеется три возможных значения:
   * Qt::DirectConnection – сигнал обрабатывается сразу вызовом соответствующего метода слота
   * Qt::QueuedConnection – сигнал преобразуется в событие и ставится в общую очередь для обработки
   * Qt::AutoConnection – это автоматический режим, который действует следующим образом: если отсылающий сигнал объект находится в одном потоке с принимающим его объектом, то устанавливается режим Qt::DirectConnection, в противном случае – режим Qt::QueuedConnection. Этот режим (Qt::AutoConnection) определен в методе connection() по умолчанию.

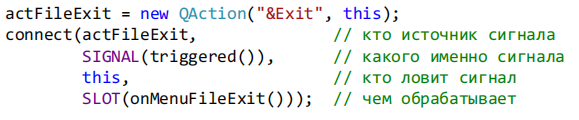


Рисунок 3 – Пример сигнально-слотового соединения

Соединять сигналы со слотами возможно не только программно. В режиме дизайна формы нажмите клавишу F4 для доступа к интерфейсу управления сигналами и слотами. Там же можно добавить в список новые, заданные программистом слоты.

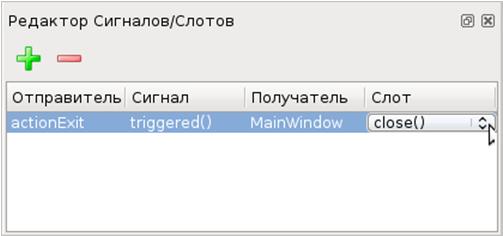


Рисунок 4 – Соединение сигналов и слотов в режиме дизайна формы

Проиллюстрируем также «визуальное» соединение сигналов со слотами на примере обработки текстового поля QLineEdit и кнопки QPushButton, размещённых на форме виджета.

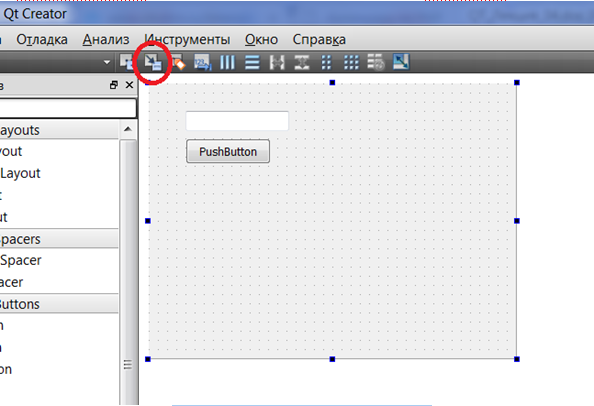


Рисунок 5 – Вид формы в режиме дизайна

Для вызова меню необходимо нажать в режиме дизайна формы клавишу F4 или кнопку «Изменение сигналов/слотов» (на рисунке 5 отмечена красным), затем зажать левой кнопкой мыши PushButton и протянуть красную линию в окно виджета (пример на рисунке 6).

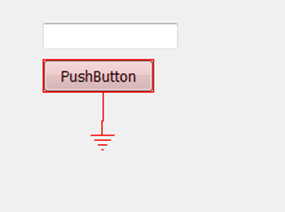


Рисунок 6 – Вызов настройки соединения

После отпускания кнопки мыши появится окно «Настройка соединения». В левой части окна отображаются сигналы, необходимо выбрать 1 сигнал (для примера выбрать clicked()), а справа нажать кнопку «Изменить», затем в новом окне Сигналы/Слоты кнопку «+» под списком слотов. К виджету добавился слот slot1(), после нажатия OK появился в окне настройки соединения (рис. 7).

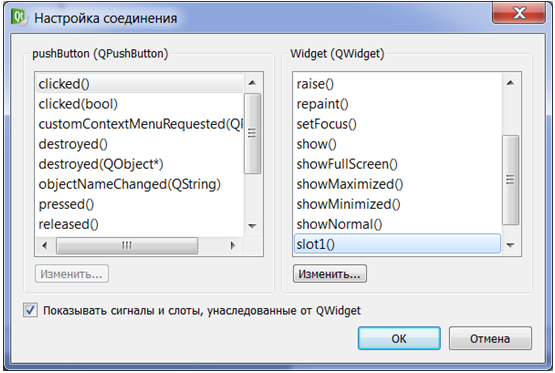


Рисунок 7 – Окно настройки соединения

После нажатия OK связь создана и отображена на форме, вернуться к обычном виду можно нажатием клавиши F3.

Также можно автоматически создать пустую функцию-слот, для этого нужно нажать правой кнопкой мыши на PushButton и выбрать пункт меню «Перейти к слоту...», а затем сигнал clicked(), для которого создаётся слот.

## Порядок выполнения работы

1. Запустить приложение QtCreator.
2. Нажать кнопку «Новый проект».

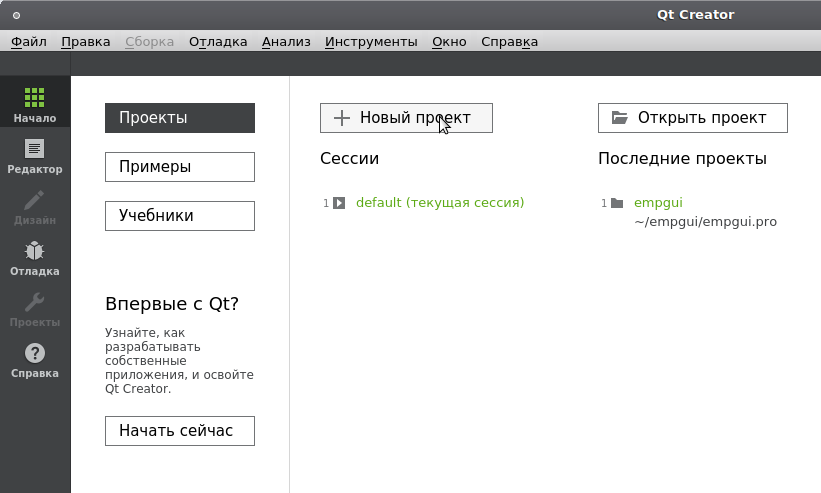


Рисунок 8 – Окно приложения

1. В качестве шаблона выбирать «Приложение/Приложение Qt Widgets»

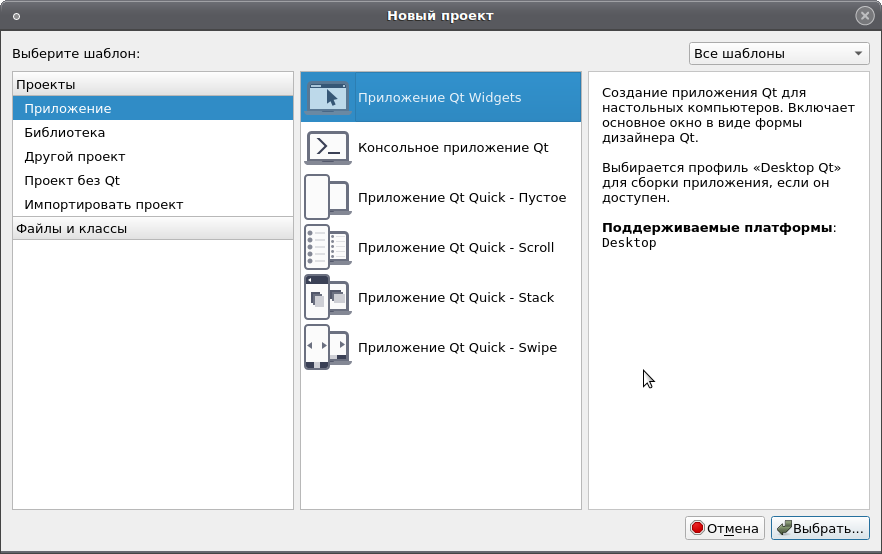


Рисунок 9 – Окно выбора типа нового проекта

1. Выбрать папку для сохранения нового проекта (в пути сохранения файлов не должно быть русскоязычных символов).
2. Выбрать наименование проекта «Фамилия\_\_LR1\_Variant№» и нажать кнопку «Создать».

## Задания

Используя возможности фреймворка Qt и среды Qt Creator, разработать приложение на языке C++, в котором будут все элементы графического интерфейса в соответствии с заданием, выполняющие определенный функционал.

Задание выполнять в соответствие с выданным вариантом. Работу разработанного приложения продемонстрировать преподавателю, ответить на дополнительные вопросы, получить оценку. Архив проекта приложить к отчету.

**Таблица 1 – Варианты заданий**

| **Номер варианта** | **Структура приложения** | **Функционал** |
| --- | --- | --- |
| 1 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QPushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) введенный текст добавляется в QTextEdit. По нажатию одной из кнопок каждый 3 символ в QTextEdit становится синего цвета, по нажатию второй кнопки цвет всего текста меняется на зеленый. |
| 2 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) введенный текст добавляется в QTextEdit. По нажатию одной из кнопок очищаются QLineEdit и QTextEdit. Пока зажата 2 кнопка во всех текстовых полях отображается символ «\*» (после отжатия отображается исходный текст). |
| 3 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) введенный текст добавляется в QTextEdit. По нажатию одной из кнопок весь текст в QTextEdit становится жирным (при повторном нажатии жирность убирается), по нажатию второй кнопки очищается QTextEdit. |
| 4 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | После ввода имени и окончания редактирования в QLineEdit (нажатие Enter) одна из кнопок меняет свой цвет на зеленый. Далее по нажатию на эту кнопку в QTextEdit отображается приветствие: «Привет, <имя>!» и кнопка меняет цвет на исходный. По нажатию на 2 кнопку все текстовые поля очищаются. |
| 5 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) введенный текст добавляется в QTextEdit. По нажатию одной из кнопок все буквы в QTextEdit становятся заглавными и каждая цифра становится синего цвета, по нажатию второй кнопки – строчными и выделение цветом убирается. |
| 6 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) введенный текст устанавливается в QTextEdit. При нажатии на кнопку текст из QTextEdit копируется в QLineEdit и удаляется из QTextEdit. По нажатию на 2 кнопку все текстовые поля очищаются. |
| 7 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) введенный текст устанавливается в QTextEdit. При нажатии на кнопку весь текст в QTextEdit меняется на символ «\*», а в QLineEdit отображается длина измененной строки. По нажатию на 2 кнопку все текстовые поля очищаются. |
| 8 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | По нажатию на 1 кнопку в QLineEdit выводится количество слов, введенных в QTextEdit. По нажатию на 2 кнопку текст в QTextEdit становится жирным, а каждая цифра в тексте выделяется красным (при повторном нажатии жирность убирается). |
| 9 | 1 однострочное текстовое поле (QLineEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | Текст в QLineEdit при каждом нажатии первой кнопки сдвигается влево (то есть символ, который был первым, становится последним), по нажатию на 2 кнопку происходит обратное действие (последний символ становится в начало строки). |
| 10 | 1 однострочное текстовое поле (QLineEdit), 1 QLabel, 1 кнопка (QpushButton). Использовать компоновку виджетов. | В QLineEdit есть ограничения на ввод данных: должно быть введено число от 0 до 100 (отобразить это ограничение для пользователя, например, с помощью метода setPlaceholderText). При нажатии на кнопку в QLabel запускается таймер (QTimer) обратного отсчета от введенного в QLineEdit значения до 0. |
| 11 | 1 однострочное текстовое поле (QLineEdit), 1 QLabel, 1 кнопка (QpushButton). Использовать компоновку виджетов. | В QLineEdit есть ограничения на ввод данных: должно быть введено число от 0 до 100 (отобразить это ограничение для пользователя, например, с помощью метода setPlaceholderText). При нажатии на кнопку в QLabel запускается таймер (QTimer) отсчета от 0 до введенного в QLineEdit значения. |
| 12 | 1 однострочное текстовое поле (QLineEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | В QLineEdit вводится текст. Кнопка запускает таймер (QTimer), который удаляет один символ из QLineEdit каждую секунду, пока поле не станет пустым. |
| 13 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | В QTextEdit по умолчанию число 0. В QLineEdit вводится число. При нажатии кнопки запускается таймер, который в QTextEdit увеличивает от 0 до числа в QLineEdit каждую секунду, каждые 2 секунды шрифт меняет цвет. |
| 14 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | В QLineEdit вводится слово. Первая кнопка ищет его в QTextEdit и выделяет жирным, вторая кнопка заменяет найденное слово на «\*\*\*». При изменении QTextEdit текст в QLineEdit меняет цвет. |
| 15 | 1 однострочное текстовое поле (QLineEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | В QLineEdit вводится число. При нажатии кнопки запускается таймер, который уменьшает число в QLineEdit каждую секунду до 0. |
| 16 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | В QTextEdit по умолчанию число 0. В QLineEdit вводится число. При нажатии кнопки запускается таймер, который в QTextEdit увеличивает от 0 до числа в QLineEdit каждую секунду. |
| 17 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | QLineEdit и QTextEdit доступны только для чтения. По нажатию на 1 кнопку в QLineEdit отображается текущая дата, в QTextEdit – текущее время. По нажатию на 1 кнопку в QTextEdit отображается текущая дата, в QLineEdit – текущее время. |
| 18 | 1 однострочное текстовое поле (QLineEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | По нажатию кнопки в QLineEdit запускается секундомер (использовать QTimer). Повторное нажатие кнопки останавливает таймер. |
| 19 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | В QLineEdit вводится только 1 символ (установить ограничения на ввод данных). По нажатию на 1 кнопку запускается таймер (использовать QTimer), который каждую секунду добавляет в QTextEdit введенный символ, по нажатию на 2 кнопку все останавливается. |
| 20 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) введенный текст добавляется в QTextEdit, а в QLineEdit заменяется на его длину. При нажатии на кнопку весь текст в QTextEdit очищается. Пока зажата 2 кнопка во всех текстовых полях отображается символ «\*» (после отжатия отображается исходный текст). |
| 21 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) введенный текст добавляется в QTextEdit с новой строки с указанием текущего времени (использовать QTimer). По нажатию на кнопку очищаются все текстовые поля. |
| 22 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) размер кнопки увеличивается. По нажатию на кнопку ее размер возвращается к исходному и запускается таймер, который добавляет один символ из введенного текста в QTextEdit каждую секунду. |
| 23 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) запускается таймер, который добавляет один символ из введенного текста в QTextEdit каждую секунду (использовать QTimer). По нажатию на кнопку таймер останавливается. |
| 24 | 1 многострочное текстовое поле (QTextEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | В QLineEdit вводится число. При нажатии кнопки запускается таймер, который уменьшает число в QLineEdit каждую секунду до 0, каждые 2 секунды шрифт меняет цвет. |
| 25 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | По нажатию на 1 кнопку в QLineEdit выводится количество слов, введенных в QTextEdit. По нажатию на 2 кнопку текст в QTextEdit становится выделен курсивом (при повторном нажатии все изменения в выделении текста убираются). |
| 26 | 1 многострочное текстовое поле (QTextEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | В QTextEdit вводится текст. Кнопка запускает таймер (QTimer), который удаляет один символ из QTextEdit каждую секунду, пока поле не станет пустым. |
| 27 | 1 QLabel, 1 многострочное текстовое поле (QTextEdit), 1 кнопка (QpushButton). Использовать компоновку виджетов. | По нажатию на кнопку текст в QTextEdit становится выделен курсивом, а каждый 10 символ выделен красным цветом (при повторном нажатии все изменения в выделении текста убираются). При изменении текста в QTextEdit отображается QLabel = «Текст изменен» и через 1 секунду пропадает (QTimer). |
| 28 | 1 QLabel,1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | После окончания редактирования в QLineEdit (нажатие Enter) отображается QLabel = «Текст изменен» и через 5 секунд пропадает (QTimer). При нажатии на 1 кнопку весь текст QLineEdit добавляется в QTextEdit. По нажатию на 2 кнопку все текстовые поля очищаются. |
| 29 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | По нажатию на 1 кнопку в QLineEdit выводится количество слов, введенных в QTextEdit. По нажатию на 2 кнопку все цифры в QTextEdit выделены курсивом, все буквы русского алфавита выделены жирным (при повторном нажатии все изменения в выделении текста убираются). |
| 30 | 1 однострочное текстовое поле (QLineEdit), 1 многострочное текстовое поле (QTextEdit), 2 кнопки (QpushButton). Использовать компоновку виджетов. | Текст в QLineEdit при каждом нажатии первой кнопки сдвигается влево (то есть символ, который был первым, становится последним). По нажатию на 2 кнопку весь текст QLineEdit добавляется в QTextEdit и становится другого цвета. |

# Интернационализация приложения

## Цели работы

1. Ознакомиться с методами интернационализации приложений фреймворка Qt.
2. Разработать приложение на языке С++, решающее задачи интернационализации.

## Краткие теоретические сведения

Объект класса **QTranslator** используется для загрузки переводов из специального файла с расширением. qm, который является шестнадцатеричным файлом переводов. Данный файл компилируется из файла переводов в XML формате, который имеет расширение ts и прописывается в pro файле проекта. Данный файл содержит все строки приложения, которые были заключены в функцию tr().

Для создания перевода необходимо пользоваться функциями tr(), trUtf8(), translate() в приложении и т.д. То есть весь текст, который будет требовать перевода необходимо обрамлять именно в эти функции, чтобы потом создать файл перевода.

После того, как в приложении помечены все необходимые для перевода строки, необходимо будет создать файл переводов, сделать непосредственно сам перевод и скомпилировать итоговый файл. Для этого используется следующие программы:

* **lupdate** – программа для формирования файла перевода из поля TRANSLATIONS в pro файле, а также обновления информации о всех новых строках в приложении, требующих перевода.
* **lrelease** – программа для сборки итогового файла переводов, который будет использоваться в приложении.
* **Qt Linguist** – непосредственно сам пакет для создания переводов.

Макрос **QT\_TRANSLATE\_NOOP3**(context, sourceText, contextId) используется для перевода текста в приложениях Qt на разные языки и содержит следующие параметры:

* context – строка, представляющая контекст или модуль, в котором используется текст. Это помогает различать похожие текстовые строки в разных частях приложения.
* sourceText – исходная текстовая строка, подлежащая переводу.
* contextId – строка, которая дополнительно определяет контекст в рамках заданного context. Это можно использовать для различения разных переводов в рамках одного контекста.

## Порядок выполнения работы

Чтобы запустить приложение с переводом надо:

1. Создать новый проект, именуя его: Фамилия\_\_LR1\_Variant№.
2. Добавить в .pro файл строки TRANSLATIONS += файл. Имя файла должно быть вида (пример для русского языка) *Имя\_проекта\_ru\_RU.ts*. Файлы соответствуют языкам, используемым в задании.
3. Запустить утилиту lupdate (Инструменты -> Внешние -> Linguist), которая создаст файлы \*.ts, указанные в \*.pro файле.
4. Отредактировать файл \*.ts. (Добавить созданный \*.ts файл в проект. Далее открыть с помощью).
5. Запустить утилиту lupdate. (Инструменты -> Внешние -> Linguist).
6. Запустить утилиты lrelease. (Инструменты -> Внешние -> Linguist).
7. При необходимости использовать справку Qt по нужному классу.
8. Для использования перевода необходимо создать объект QTranslator (При вызове внутри методов – создать поле класса).
9. Использовать метод класса .load(). В качестве параметров передать имя файла, полученного утилитой lrelease. (\*.qm файл). И путь к директории файла.
10. Применить к объекту класса QApplication (QApplication a(*argc*, argv); или qApp в методе объекта) метод installTranslator и передать в качестве параметра указатель на объект QTranslator.
11. Для перевода строк необходимо использовать функции [QObject::tr](file:///C:\Users\User\Downloads\qobject.html#tr)() или QCoreApplication::translate().
12. При необходимости смены локалей (например, формат даты) использовать класс QLocale.
13. При переводе строк, которые создаются вне метода tr() применять соответствующие макросы: QT\_TRANSLATE\_NOOP3 и т. д. (Документация находится в справке Qt).

## Задание

Используя возможности фреймворка Qt и среды Qt Creator, разработать приложение на языке C++, позволяющее решить задачи интернационализации и локализации приложения. Разрабатываемая программа должна поддерживать 3 языка: русский, английский, немецкий. Язык должен выбираться в самом приложении. Необходимо использовать сущности, указанные в скобках.

Задание выполнять в соответствие с выданным вариантом. Работу разработанного приложения продемонстрировать преподавателю, ответить на дополнительные вопросы, получить оценку. Архив проекта приложить к отчету.

**Таблица 2 – Варианты заданий**

| **Номер варианта** | **Структура приложения** |
| --- | --- |
| 1 | Label с фразой «Привет, Мир», текст находится в константной строке внутри класса. (QT\_TR\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 2 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке внутри класса. Программа считает количество нажатий. (QT\_TR\_N\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 3 | Label с фразой «Привет, Мир», текст находится в константной строке вне класса. (QT\_TRANSLATE\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 4 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке вне класса. Программа считает количество нажатий. (QT\_TRANSLATE\_N\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 5 | Label с фразой «Привет, Мир», текст находится в константной строке внутри класса. (QT\_TR\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 6 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке внутри класса. Программа считает количество нажатий. (QT\_TR\_N\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 7 | Label с фразой «Привет, Мир», текст находится в константной строке вне класса. (QT\_TRANSLATE\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 8 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке вне класса. Программа считает количество нажатий. (QT\_TRANSLATE\_N\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 9 | Label с фразой «Привет, Мир», текст находится в константной строке внутри класса. (QT\_TR\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 10 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке внутри класса. Программа считает количество нажатий. (QT\_TR\_N\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 11 | Label с фразой «Привет, Мир», текст находится в константной строке вне класса. (QT\_TRANSLATE\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 12 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке вне класса. Программа считает количество нажатий. (QT\_TRANSLATE\_N\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 13 | Label с фразой «Привет, Мир», текст находится в константной строке внутри класса. (QT\_TR\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 14 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке внутри класса. Программа считает количество нажатий. (QT\_TR\_N\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 15 | Label с фразой «Привет, Мир», текст находится в константной строке вне класса. (QT\_TRANSLATE\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 16 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке вне класса. Программа считает количество нажатий. (QT\_TRANSLATE\_N\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (tr())). |
| 17 | Label с фразой «Привет, Мир», текст находится в константной строке внутри класса. (QT\_TR\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 18 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке внутри класса. Программа считает количество нажатий. (QT\_TR\_N\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 19 | Label с фразой «Привет, Мир», текст находится в константной строке вне класса. (QT\_TRANSLATE\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 20 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке вне класса. Программа считает количество нажатий. (QT\_TRANSLATE\_N\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 21 | Label с фразой «Привет, Мир», текст находится в константной строке внутри класса. (QT\_TR\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 22 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке внутри класса. Программа считает количество нажатий. (QT\_TR\_N\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 23 | Label с фразой «Привет, Мир», текст находится в константной строке вне класса. (QT\_TRANSLATE\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 24 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке вне класса. Программа считает количество нажатий. (QT\_TRANSLATE\_N\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображение единиц измерения длины (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 25 | Label с фразой «Привет, Мир», текст находится в константной строке внутри класса. (QT\_TR\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 26 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке внутри класса. Программа считает количество нажатий. (QT\_TR\_N\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 27 | Label с фразой «Привет, Мир», текст находится в константной строке вне класса. (QT\_TRANSLATE\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 28 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке вне класса. Программа считает количество нажатий. (QT\_TRANSLATE\_N\_NOOP)  TextEdit с числом 12345.67 (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 29 | Label с фразой «Привет, Мир», текст находится в константной строке внутри класса. (QT\_TR\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 30 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке внутри класса. Программа считает количество нажатий. (QT\_TR\_N\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 31 | Label с фразой «Привет, Мир», текст находится в константной строке вне класса. (QT\_TRANSLATE\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |
| 32 | PushButton с фразой «Сейчас произведено n нажатий», текст находится в константной строке вне класса. Программа считает количество нажатий. (QT\_TRANSLATE\_N\_NOOP)  TextEdit с текущей датой (QLocale).  Label с отображением названия страны из (QLocale).  3 RadioButton с выбором языка. (Перевод через (QCoreApplication::translate())). |

# Работа с XML

## Цели работы

## Краткие теоретические сведения

### Описание формата XML

XML (eXtensible Markup Language) – это язык разметки, который используется для структурирования, хранения и передачи данных между различными системами и приложениями. XML был разработан как упрощённая и расширяемая форма SGML (Standard Generalized Markup Language), применяемый для создания документов в форматах, поддерживающих различные стили отображения. XML часто используется для описания и передачи данных через Интернет, потому что он легко читаем как человеком, так и машиной.

Стандартная структура файла показана на рисунке 10.

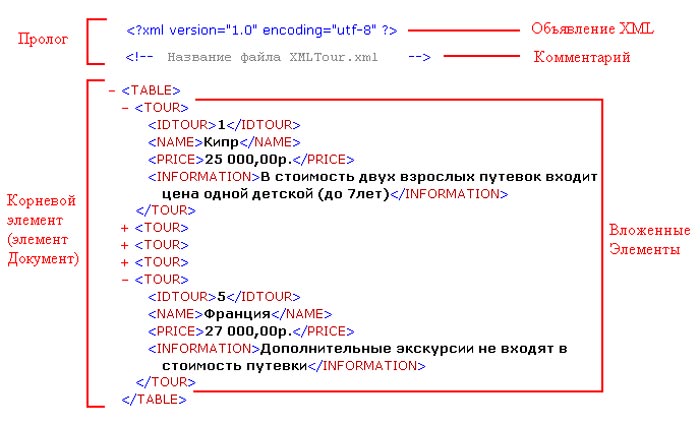


Рисунок 10 – Структура xml-файла

Первая строка листинга представляет собой пролог, определяющий параметры документа: версию стандарта языка XML (version), кодировку документа (encoding): **<?xml version="1.0" encoding="UTF-8"?>.**

Далее идет тело XML документа. Тело представляет собой набор иерархически организованных элементов. Элемент состоит из начального тега, содержимого и конечного тега. Начальный и конечный теги заключаются в угловые скобки с тем отличием, что для конечного тега после открывающей угловой скобки требуется добавить символ прямого слеша “/”. Помимо имени начальный тег может содержать атрибуты и их значения.

Каждый элемент может содержать как данные, так и вложенные элементы.

Элементом верхнего уровня (корневым элементом) может быть только один элемент.

### Работа с xml-файлами в Qt

Средства разработки Qt включают в себя функционал по работе с XML файлами, который позволяет как создавать XML файлы, так и читать их. В библиотеку Qt включены классы, реализующих различные подходы к обработке XML-документов: **QXmlStreamReader, QXmlSimpleReader, QDomDocument.**

**QXmlStreamReader** воспринимает документ в виде потока тегов. Интерфейс класса позволяет перемещаться по этому набору. Класс предоставляет самый быстрый обработки XML-документов, однако об иерархической структуре документа класс ничего не знает и не подходит для добавления информации в документ; Класс имеет ряд методов:

* **readNext**() – метод переводит XML-считыватель к следующему событию в XML потоке. Событиями могут быть начало и конец элемента, текстовые данные и т.д. Используется для итеративного прохода через элементы XML документа.
* **readElementText**() – при вызове для элемента, метод считывает и возвращает весь текст, содержащийся в текущем элементе, включая его дочерние элементы, и затем перемещает считыватель к следующему событию после закрывающего тега текущего элемента. Очень удобен для извлечения текстового содержимого элементов.
* **name**() – возвращает имя текущего элемента (тега) без префикса пространства имен как объект QStringRef. Этот метод полезен для получения имени тега, чтобы проверить, какой элемент сейчас обрабатывается XML-считывателем.
* **atEnd**() – проверяет, достиг ли XML-считыватель конца документа. Возвращает true, если XML-считыватель находится в конце документа и дальнейшее чтение невозможно. Этот метод полезен для определения завершения анализа XML документа.
* **hasError**() – проверяет, возникли ли ошибки при чтении XML документа. Возвращает true, если при анализе документа была обнаружена ошибка. Этот метод может использоваться для валидации корректности и структуры XML документа и обработки возможных ошибок.

**QXmlSimpleReader**, реализует SAX-интерфейс (Java API) для работы с XML. Фактически, является тем же QXmlStreamReader, но в другой обертке – при считывании тега в объекте-обработчике вызывается виртуальная функция, соответствующая типу тега. Функция-обработчик ничего не знает о предыдущих вызовах и обрабатывает только текущий тег, поэтому требует также мало памяти, как QXmlStreamReader и не подходит для изменения документа;

**QDomDocument**, реализует DOM-интерфейс (W3C API) обработки XML. Считывает весь документ в память в виде дерева, поэтому не рекомендуется для обработки больших документов. Позволяет изменять дерево и, соответственно, документ.

Класс QDomElement создан для представления элементов. Иерархия DOM содержит узлы различного типа. Например, узел элемента соответствует открытому и закрытому тегу. Данные, находящиеся между этими тегами, представляют собой узлы потомков типа «элемент». Все узлы иерархии DOM являются объектами класса QDomNode, которые способны содержать в себе любые типы узлов. Для проведения операций над узлом его, прежде всего, необходимо преобразовать к нужному типу. Для преобразования объектов QDomNode в QDomElement следует воспользоваться методом QDomNode::toElement(). Нужно всегда проверять возвращаемое этим методом значение, ведь в случае ошибки будет возвращено нулевое значение, которое можно проверить методомisNull().

Список базовых функций, которые часто используются при работе с QDomDocument:

* setContent() – загружает XML документ;
* createElement() – создаёт новый элемент с заданным тегом.
* createAttribute() – создаёт новый атрибут.
* createTextNode() – создаёт текстовый узел.
* createProcessingInstruction() – создаёт обработчик инструкций.
* createDocumentFragment() – создаёт фрагмент документа.
* documentElement() – возвращает корневой элемент документа.
* firstChildElement() – возвращает первый дочерний элемент с заданным тегом.
* elementsByTagName() – возвращает список всех элементов с заданным тегом.
* hasChildNodes() – проверяет, есть ли у документа дочерние узлы.

**QXmlStreamWriter –** это класс в Qt, предназначенный для удобного и эффективного создания XML файлов. Этот класс предоставляет возможность писать XML данные в поток, абстрагируя пользователя от деталей форматирования XML структуры. Это удобно для генерации правильно сформированных XML документов с минимальными затратами времени и кода. Основные функции:

* writeStartDocument() – начинает XML документ с указанной версией и атрибутом standalone.
* writeEndDocument() – заканчивает XML документ.
* writeStartElement() – Начинает новый элемент с указанным именем.
* writeEndElement() – заканчивает текущий элемент.
* writeAttribute() – добавляет атрибут к текущему элементу.
* writeAttributes() – добавляет группу атрибутов к текущему элементу.

Таблица 3 – Сравнение инструментов обработки xml-файлов в Qt

|  | **QXmlStreamReader** | **QXmlSimpleReader (SAX)** | **QDomDocument** |
| --- | --- | --- | --- |
| Преимущества | Самый быстрый обработчик XML файлов | В память помещаются только фрагменты, а не весь XML-документ | Позволяет изменять дерево и документ |
| Недостатки | Не подходит для добавления информации в документ Ничего не знает об иерархической структуре документа | Уступает QXmlStreamReader в быстродействии | Не рекомендуется при обработке больших документов |

### Чтение XML файла

**QFile** – это класс в библиотеке Qt, предоставляющий интерфейс для работы с файлами на файловой системе. Он наследуется от QIODevice и предлагает возможности для открытия, чтения, записи, закрытия файла, а также выполнения других операций, таких как перемещение указателя в файле, проверка существования файла, удаление файла и получение его свойств (например, размера).



Рисунок 11 – Пример работы с QFile

Поддержка XML в Qt — это отдельный модуль QtXml, для использования которого необходимо указать его имя в проектном файле. Просто добавьте следующую строку: **QT += xml**

А для того, чтобы работать с классами этого модуля, необходимо включить заголовочный метафайл QtXml (**#include <QtXml>**).

**QXmlStreamReader** – это класс в библиотеке Qt, который предоставляет удобный интерфейс для чтения данных в формате XML.

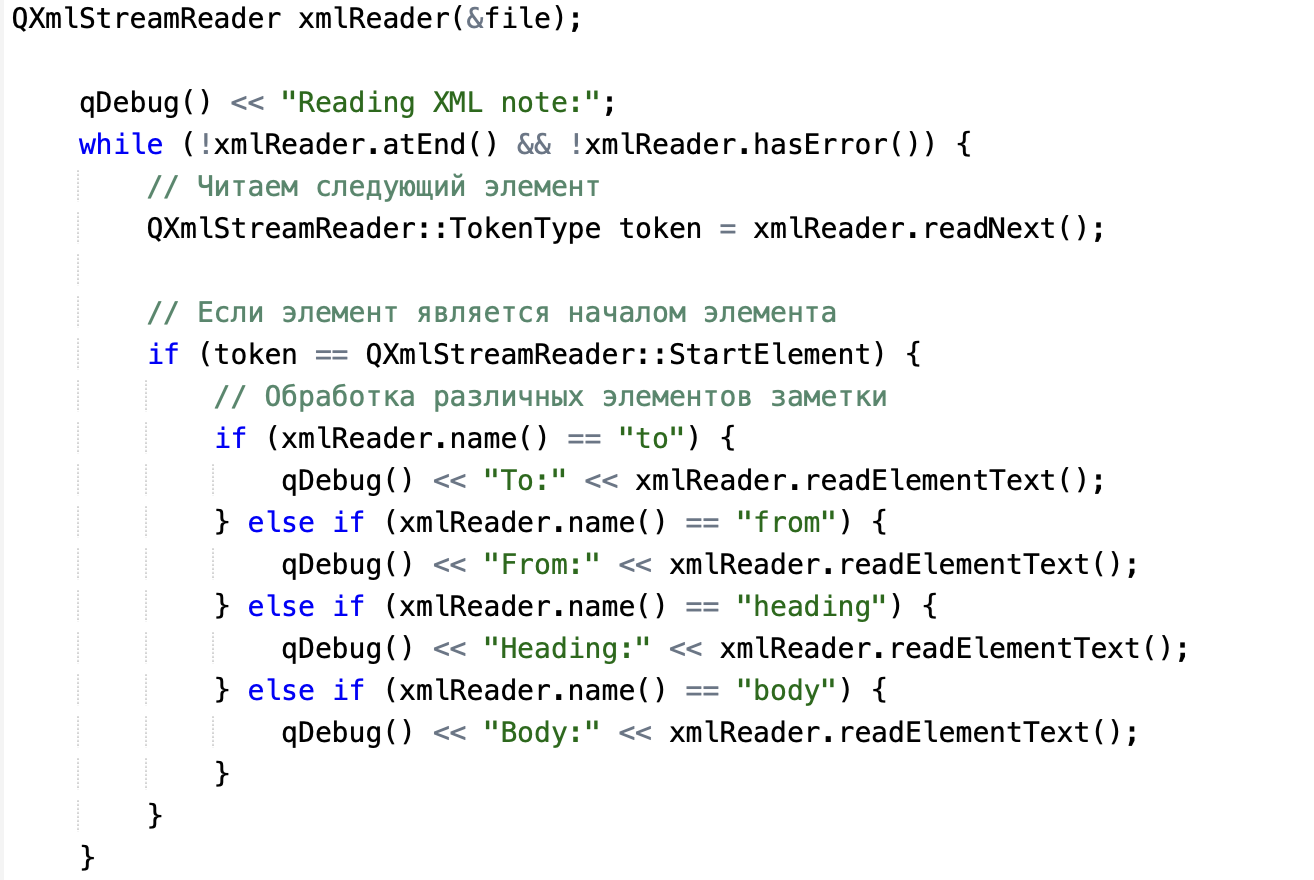


Рисунок 12 – Пример чтения xml-файла с помощью QXmlStreamReader

**QXmlSimpleReader** работает как SAX-парсер, что означает, что он предоставляет интерфейс для секвенциального чтения XML-данных. Чтобы поместить XML-документ в SAX-анализатор, нужно создать объект класса QXmlInputSource, передав ему указатель на QIODevice. Созданный объект нужно передать в метод parse() объекта класса QXmlSimpleReader. Этот метод запустит процесс анализа XML-документа.

Рассмотрим пример чтения документа адресной книги (). В этом примере есть класс AddressBookParser реализует виртуальные методы, которые вызываются при прохождении по XML-документу. До вызова метода parse() необходимо установить в методе setContenHandler() объект класса AddressBookparser.

При нахождении тегов вызываются соответствующие методы startElement() и endElement(). Метод startElement() вызывается тогда, когда при считывании встречается открытие тега. Третий параметр, передаваемый в этот метод, — это имя тега. Четвертый — это список атрибутов. В примере, если название атрибута совпадает со строкой number, то производится вывод его значения. Метод characters() производит запись содержимого текущего элемента в переменную m\_strText, а это необходимо для того, чтобы можно было получить доступ к этому содержимому из любого метода класса AddressBookParser. Метод endElement() вызывается всегда, когда при чтении встречается закрытие тега. Третий параметр, передаваемый в этот метод, — имя тега. В случае несовпадения его со строками contact и addressbook, производится отображение данных элемента.

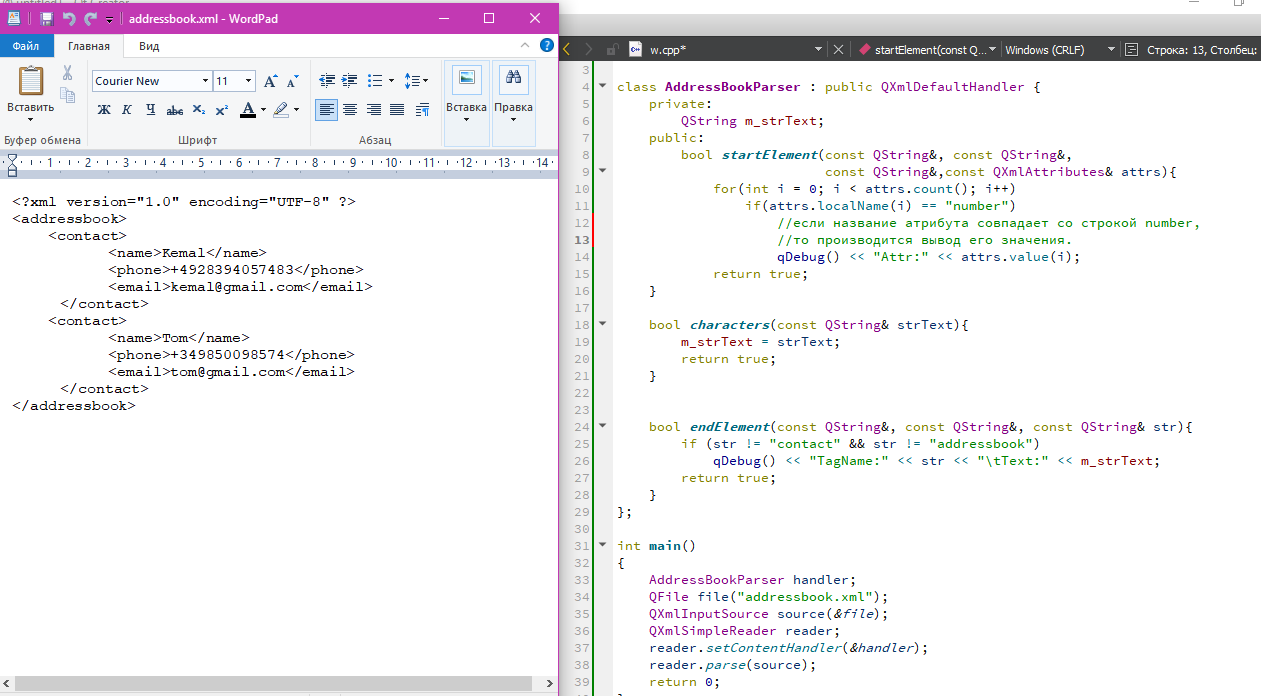


Рисунок 13 – Пример чтения xml-файла с помощью QXmlSimpleReader

Рассмотрим пример чтения документа адресной книги при помощи **QDomDocument** (Рисунок 14). В функции main() создаются объекты классов QDomDocument и QFile. Для считывания XML-документа в метод setContext() объекта класса QDomDocument необходимо передать объект, произведенный от класса, унаследованного от QIODevice. В данном случае это класс QFile. Далее, для получения объекта класса QDomElement производится вызов метода documentElement() объекта класса QDomDocument. Полученный объект передается в функцию traverseNode(), которая обеспечивает прохождение по всем элементам XML-документа.

Для прохождения по иерархии DOM удобно применить рекурсию. Из рекурсивной функции traverseNode() вызываются методы firstChild() и nextSibling() объекта класса QDomNode. Ели метод nextSibling() возвращает нулевое значение, то это значит, что узлов больше нет. Все получаемые узлы проверяются в цикле с помощью метода isElement(). В том случае, если полученный узел является элементом, производится преобразование его к типу QDomElement, с помощью метода toElement(), с последующим присвоением результата объекту класса QDomEiement. При обнаружении элемента с именем «contact», с помощью метода attribute() производится отображение значения его атрибута number. В остальных случаях вызываются методы tagName() и text() для отображения имени и данных элемента.



Рисунок 14 – Пример чтения xml-файла с помощью QDomDocument

## Задание

Создать новый проект, именуя его: Фамилия\_LR3\_Variant№.

Используя возможности фреймворка Qt и среды Qt Creator, разработать приложение на языке C++, позволяющее работать с файлами формата xml и отображать их в требуемом формате.

Задание выполнять в соответствие с выданным вариантом. Работу разработанного приложения продемонстрировать преподавателю, ответить на дополнительные вопросы, получить оценку. Архив проекта приложить к отчету.

**Таблица 4 – Варианты заданий**

| **Номер варианта** | **Формирование файла** | **Структура файла** | **Отображение содержимого файла** | **Метод обработки файла** |
| --- | --- | --- | --- | --- |
| 1 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о студентах: название группы, их имена, возрасты и оценки по различным предметам. Количество групп не меньше 3, в каждой не менее 4 студентов, не менее 3 предметов для каждого студента. | Qt для чтения файла и вывода информации с помощью QTreeView | DOM |
| 2 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о книгах в библиотеке (Жанр, название, авторы, год издания). Жанров - 2, названий в каждом жанре 5, а авторов не менее двух. | Qt для чтения файла и вывода информации с помощью QTableView | DOM |
| 3 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Файл, представляющий структуру файловой системы (Корневая директория, в которой 5 вложенных директорий, в каждой из этих директорий – 3 вложенных, в которых по одной пустой директории и не менее одного файла). | Qt для чтения файла и вывода информации с помощью QTreeView | DOM |
| 4 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о продуктах в магазине (категория, название, цена, описание, срок годности), категорий – 4, названий в каждой категории - 6. | Qt для чтения файла и вывода информации с помощью QListView | DOM |
| 5 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о фильмах (режиссер, название, год выпуска, длительность), 3 режиссера – у каждого не менее 7 фильмов. | Qt для чтения файла и вывода информации с помощью QTableView | DOM |
| 6 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о расписании занятий на 2 недели (номер недели, день, занятия). Каждый день не менее двух дел. | Qt для чтения файла и вывода информации с помощью QColumnView | DOM |
| 7 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о контактах в адресной книге (страна, имя, адрес, номер телефона). 5 стран, 6 имён. | Qt для чтения файла и вывода информации с помощью QListView | DOM |
| 8 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о меню ресторана (тип блюда (закуска, салаты, горячее, суп, десерт), название блюда, описание, цена). Названий блюд не меньше 5 в каждом разделе. | Qt для чтения файла и вывода информации с помощью QTableView | DOM |
| 9 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о погоде на различные дни недели. (День недели, время суток (утро, день, вечер, ночь), точное время, температура, влажность). | Qt для чтения файла и вывода информации с помощью QListView | DOM |
| 10 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о структуре организации (например, отделы, должности, сотрудники, стаж). 4 отдела, 6 должностей. | Qt для чтения файла и вывода информации с помощью QTreeView | DOM |
| 11 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о студентах: название группы, их имена, возрасты и оценки по различным предметам. Количество групп не меньше 3, в каждой не менее 4 студентов, не менее 3 предметов для каждого студента. | Qt для чтения файла и вывода информации с помощью QTreeView | SAX |
| 12 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о книгах в библиотеке (Жанр, название, авторы, год издания). Жанров - 2, названий в каждом жанре 5, а авторов не менее двух. | Qt для чтения файла и вывода информации с помощью QTableView | SAX |
| 13 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Файл, представляющий структуру файловой системы (Корневая директория, в которой 5 вложенных директорий, в каждой из этих директорий – 3 вложенных, в которых по одной пустой директории и не менее одного файла). | Qt для чтения файла и вывода информации с помощью QTreeView | SAX |
| 14 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о продуктах в магазине (категория, название, цена, описание, срок годности), категорий – 4, названий в каждой категории - 6. | Qt для чтения файла и вывода информации с помощью QListView | SAX |
| 15 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о фильмах (режиссер, название, год выпуска, длительность), 3 режиссера – у каждого не менее 7 фильмов. | Qt для чтения файла и вывода информации с помощью QTableView | SAX |
| 16 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о расписании занятий на 2 недели (номер недели, день, занятия). Каждый день не менее двух дел. | Qt для чтения файла и вывода информации с помощью QColumnView | SAX |
| 17 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о контактах в адресной книге (страна, имя, адрес, номер телефона). 5 стран, 6 имён. | Qt для чтения файла и вывода информации с помощью QListView | SAX |
| 18 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о меню ресторана (тип блюда (закуска, салаты, горячее, суп, десерт), название блюда, описание, цена). Названий блюд не меньше 5 в каждом разделе. | Qt для чтения файла и вывода информации с помощью QTableView | SAX |
| 19 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о погоде на различные дни недели. (День недели, время суток (утро, день, вечер, ночь), точное время, температура, влажность). | Qt для чтения файла и вывода информации с помощью QListView | SAX |
| 20 | Приложение «Блокнот» или другой инструмент для формирования XML-документов | Информация о структуре организации (например, отделы, должности, сотрудники, стаж). 4 отдела, 6 должностей. | Qt для чтения файла и вывода информации с помощью QTreeView | SAX |
| 21 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Информация о студентах: название группы, их имена, возрасты и оценки по различным предметам. Количество групп не меньше 3, в каждой не менее 4 студентов, не менее 3 предметов для каждого студента. | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |
| 22 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Информация о книгах в библиотеке (Жанр, название, авторы, год издания). Жанров - 2, названий в каждом жанре 5, а авторов не менее двух. | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |
| 23 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Файл, представляющий структуру файловой системы (Корневая директория, в которой 5 вложенных директорий, в каждой из этих директорий – 3 вложенных, в которых по одной пустой директории и не менее одного файла). | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |
| 24 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Информация о продуктах в магазине (категория, название, цена, описание, срок годности), категорий – 4, названий в каждой категории - 6. | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |
| 25 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Информация о фильмах (режиссер, название, год выпуска, длительность), 3 режиссера – у каждого не менее 7 фильмов. | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |
| 26 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Информация о расписании занятий на 2 недели (номер недели, день, занятия). Каждый день не менее двух дел. | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |
| 27 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Информация о контактах в адресной книге (страна, имя, адрес, номер телефона). 5 стран, 6 имён. | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |
| 28 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Информация о меню ресторана (тип блюда (закуска, салаты, горячее, суп, десерт), название блюда, описание, цена). Названий блюд не меньше 5 в каждом разделе. | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |
| 29 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Информация о погоде на различные дни недели. (День недели, время суток (утро, день, вечер, ночь), точное время, температура, влажность). | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |
| 30 | Разработать приложение Qt, позволяющее сформировать файл указанного формата, используя функционал из столбца «Метод обработки файла» таблицы. | Информация о структуре организации (например, отделы, должности, сотрудники, стаж). 4 отдела, 6 должностей. | Приложение «Блокнот», браузер или другой инструмент для просмотра XML-документов | DOM |

# Работа с последовательными портами

## Цели работы

1. Ознакомиться с классом QSerialPort.
2. Научиться открывать, настраивать, отправлять и принимать данные через последовательный порт.
3. Разработать приложение в Qt для работы с COM-портами.

## Краткие теоретические сведения

### Описание последовательного порта

Последовательный порт (англ. serial port, COM-порт, англ. communications port) – сленговое название интерфейса стандарта RS-232, которым массово оснащались персональные компьютеры. Порт называется «последовательным», так как информация через него передаётся по одному биту, последовательно бит за битом (в отличие от параллельного порта). Данные передаются пакетами (рис. 15).

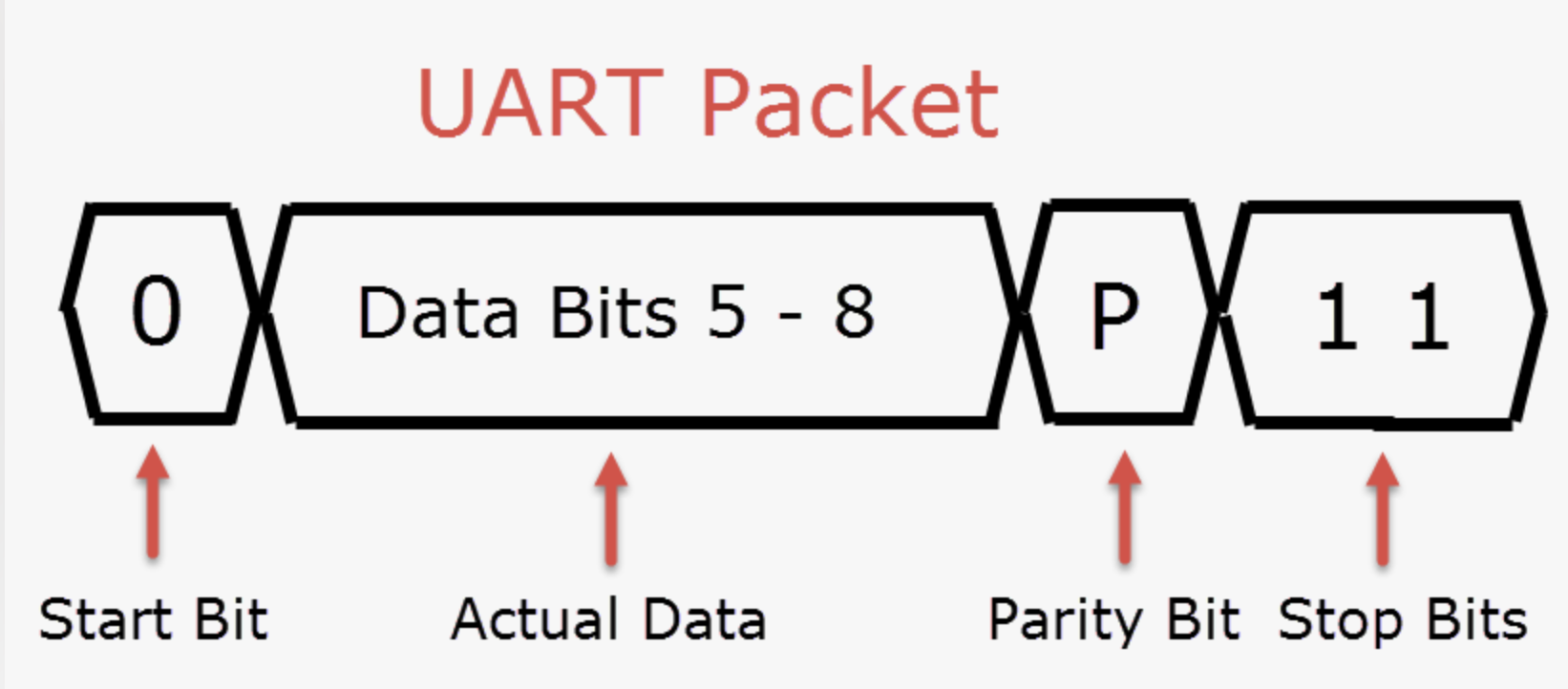


Рисунок 15 – Структура пакета данных, передаваемых по последовательному порту

**Параметры передачи:**

1. **Baudrate**, или скорость передачи данных, определяет количество битов данных, которые передаются за одну секунду. Измеряется в бодах (бит в секунду), при этом распространённые значения включают 9600, 19200, 38400, 57600 и 115200 бод. Выбор baudrate зависит от возможностей оборудования и требуемой пропускной способности.
2. **Паритет** (**parity)** используется для контроля ошибок при передаче данных. Он может быть настроен на одно из следующих значений:

* None – паритет не используется;
* Even – чётный паритет (сумма значащих битов в передаваемом байте должна быть чётной);
* Odd – нечётный паритет (сумма значащих битов должна быть нечётной);
* Mark – паритет устанавливается в высокое состояние (1);
* Space – паритет устанавливается в низкое состояние (0).

1. **Стоп-биты** используются для обозначения конца байта данных и позволяют приёмнику корректно определить конец переданного байта. Может быть установлено:

* 1 stop bit – один стоп-бит;
* 1.5 stop bits – полтора стоп-бита (обычно используется с 5 битами данных);
* 2 stop bits – два стоп-бита.

### QSerialPort

QSerialPort – это класс в Qt, который предоставляет функционал для работы с последовательными портами. Это часть Qt Serial Port модуля, который позволяет устанавливать связь с последовательными портами в кроссплатформенной манере. Класс QSerialPort инкапсулирует доступ к последовательному порту и предоставляет ряд удобных средств для отправки и приёма данных через последовательные порты.

Синхронный подход – в момент ожидания данных поток блокируется. Асинхронный подход – чтение данных производится по сигналам.

Основные методы QSerialPort (перечислены только основные методы, полный список смотрите в документации):

* open() – открывает последовательный порт с указанным режимом (например, для чтения, записи или обоих).
* close() – закрывает последовательный порт.
* setBaudRate() – устанавливает скорость передачи данных для порта.
* setDataBits() – устанавливает количество бит данных.
* setParity() – устанавливает тип паритета.
* setStopBits() – устанавливает количество стоп-битов.
* setFlowControl() – устанавливает управление потоком данных.
* write() – пишет массив байт в последовательный порт.
* readAll() – считывает все доступные данные из последовательного порта.

### Настройка QT для работы с последовательным портом

Для работы с последовательным портом в Qt используется класс QSerialPort. Для его использования в проекте необходимо добавить в файл .pro строчку **QT += serialport**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 16 – Пример инициализации последовательного порта

Для пересылки данных используется метод один из методов **write**().

Для обработки ошибок можно использовать метод **waitForBytesWritten**(), который ждет завершения отправки данных и возвращает true если она прошла успешно и false если в процессе возникли ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 17 – Пример пересылки одного байта

Когда последовательный порт получает новые данные, генерируется сигнал readyRead().

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 18 – Пример получения данных

### Руководство по созданию канала связи между приложениями через виртуальные последовательные порты

Для того чтобы симулировать взаимодействие между приложениями через последовательный порт необходимо создать два виртуальных COM порта и виртуально соединить их. Для этого необходимо выполнить следующие шаги.

**Шаг 1. (опционально).** Проверить список доступных портов на текущий момент. Для того, чтобы это сделать, необходимо откроем пример Qt terminal или приложение Terminal 1.9.

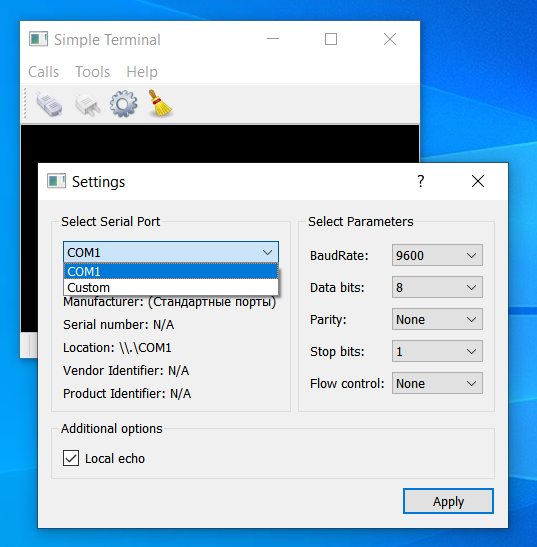


Рисунок 19 – Проверка доступных COM-портов

**Шаг 2.** Открыть приложение «VSPE».



Рисунок 20 – Значок приложения «VSPE»

**Шаг 3**. Создать новое устройство, нажатие кнопки как на рисунке 21.

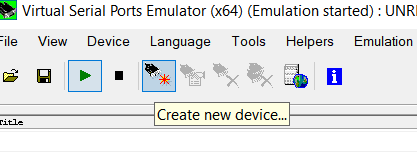


Рисунок 21 – Кнопка создания нового устройства

**Шаг 4**. В качестве типа выбрать Pair.

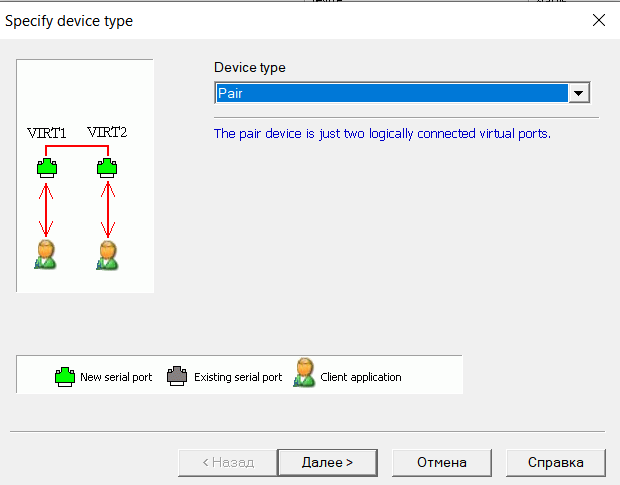


Рисунок 22 – Выбор типа устройства

**Шаг 5**. Найти пару доступных портов.

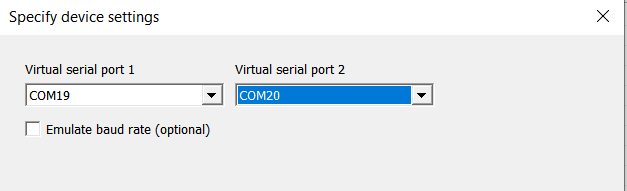


Рисунок 23 – Выбор COM-портов

**Шаг 6**. Проверить, что статус Ready.

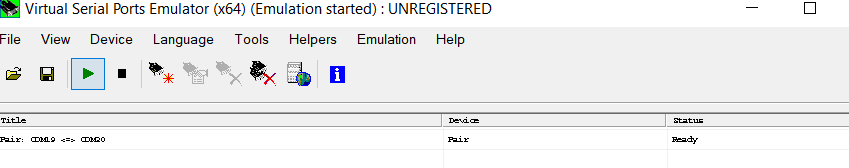


Рисунок 24 – Запуск симуляции COM-портов

**Шаг 7**. Открыть приложение из примера Qt terminal или Terminal 1.9. Убедиться, что созданные COM-порты отображаются в списке.

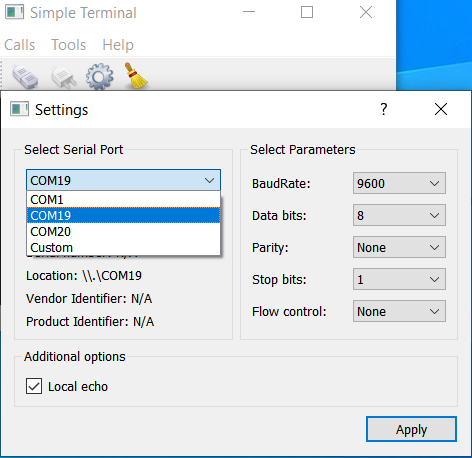


Рисунок 25 – Проверка доступных COM-портов

## Порядок выполнения работы

1. Настроить 2 виртуальных последовательных порта по инструкции п. 4.2.4.
2. Открыть приложение «Terminal-v.1.9b». Подключиться к одному из созданных портов.
3. Выполнить настройку последовательного порта в Qt.

## Задания

Создать новый проект, именуя его: Фамилия\_LR4\_Variant№.

Разработать приложение Qt на языке С++, в котором есть 1 однострочное текстовое поле для ввода данных (QLineEdit), 1 многострочное текстовое поле для вывода данных (QTextEdit), кнопки для подключения/отключения к последовательному порту, кнопка для отправки сообщений. Настроить скорость обмена по последовательным портам равной 9600 бит/с. В QLineEdit вводятся символы и по кнопке отправляются в последовательный порт. Все полученные из последовательного порта данные отображаются в QTextEdit.

Задание выполнять в соответствие с выданным вариантом. Работу разработанного приложения продемонстрировать преподавателю, ответить на дополнительные вопросы, получить оценку. Архив проекта приложить к отчету.

**Таблица 5 – Варианты заданий**

| **Номер варианта** | **Задание** |
| --- | --- |
| 1 | При получении по последовательному порту символа «T» отправлять в ответ текущее время в виде последовательности символов «чч:мм:сс». |
| 2 | При получении по последовательному порту символа «D» отправлять в ответ текущую дату в виде последовательности символов «дд/мм/гг» |
| 3 | При получении по последовательному порту символа проверять четность его ASCII-кода и отправлять в ответ символ «С», если четное. |
| 4 | При получении по последовательному порту символа проверять четность его ASCII-кода и отправлять в ответ символ «N», если нечетное. |
| 5 | При получении по последовательному порту числа от 0 до 9 (другие символы игнорировать) проверять его четность и отправлять в ответ символ «N», если нечетное. |
| 6 | При получении по последовательному порту числа от 0 до 9 (другие символы игнорировать) проверять его четность и отправлять в ответ символ «С», если четное. |
| 7 | При получении по последовательному порту числа от 0 до 9 (другие символы игнорировать) прибавлять к нему его ASCII-код и отправлять обратно в виде последовательности символов полученной суммы. |
| 8 | После получения 10 символов отправлять в ответ символ «N». |
| 9 | При получении по последовательному порту числа от 0 до 9 (другие символы игнорировать) отправлять в ответ символ «N» полученное количество раз с интервалом 1 секунда. |
| 10 | При получении по последовательному порту любого символа отправлять его в ответ через 5 секунд. |
| 11 | При получении по последовательному порту символа «I» отправлять в ответ информацию по последовательному порту (baudrate, portName) в виде строки символов. |
| 12 | При получении по последовательному порту символа «D» отключать подключение по последовательному порту |
| 13 | При получении по последовательному порту символа «X» отправлять в ответ строку с IP-адресом устройства. |
| 14 | При получении по последовательному порту числа от 0 до 9 (другие символы игнорировать) вычислять его факториал и отправлять обратно. |
| 15 | При получении по последовательному порту строчных букв отправлять обратно ту же букву, но сделать ее заглавной. |
| 16 | При получении по последовательному порту заглавных букв отправлять обратно ту же букву, но сделать ее строчной. |
| 17 | Считать количество полученных по последовательному порту сообщений, а при получении символа «N» отправлять обратно значение счетчика и обнулять его. |
| 18 | При получении по последовательному порту в течение 30 секунд 5 символов отправлять в ответ строку «STOP», иначе обнулять счетчик и сбрасывать таймер каждые 30 секунд. |
| 19 | Получить по последовательному порту 5 цифр и отправить в ответ их произведение в виде строки символов. |
| 20 | Получить по последовательному порту 5 цифр и отправить в ответ число четных чисел среди них в виде символа. |
| 21 | Получить по последовательному порту 5 цифр и отправить в ответ число нечетных чисел среди них в виде символа. |
| 22 | Получить по последовательному порту последовательность из 5 символов и отправить их в ответ в обратном порядке. |
| 23 | При получении по последовательному порту строки «PING» отправлять в ответ строку «PONG». |
| 24 | Получить по последовательному порту 5 цифр и отправить в ответ их сумму в виде строки символов. |
| 25 | Получить по последовательному порту 5 цифр и отправить в ответ их среднее арифметическое в виде строки символов. |
| 26 | При получении по последовательному порту буквы отправлять ее обратно через 1 секунду, при получении цифры ничего не отправлять. |
| 27 | При получении по последовательному порту строки из 4 символов отправлять обратно строку, состоящую из этих же символов, но отсортированных в порядке возрастания их ASCII-кодов. |
| 28 | При получении строки с наименованием порта (COMх) отправлять в ответ строку «Success». |
| 29 | Получить по последовательному порту 5 любых символов и отправить в ответ их среднее арифметическое цифр из полученной строки в виде строки символов. |
| 30 | Получить по последовательному порту 5 любых символов и отправить в ответ их произведение цифр из полученной строки в виде строки символов. |

# Программирование последовательных портов микроконтроллеров

## Цели работы

Освоить методы взаимодействия последовательного порта микроконтроллера для передачи и приема данных.

## Краткие теоретические сведения

### Создание проекта в Proteus

Для работы с последовательным портом в Proteus необходимо выполнить следующие шаги.

1. Зайти в Proteus и создать новый проект, как показано на рисунках.

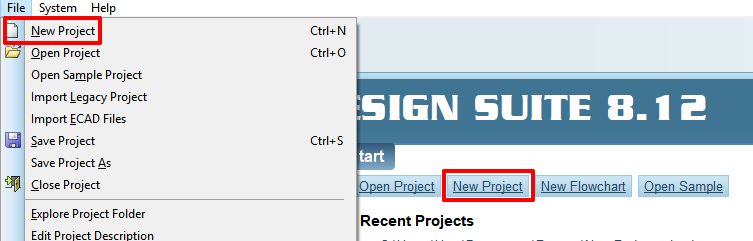


Рисунок 26 – Кнопки для создания проекта

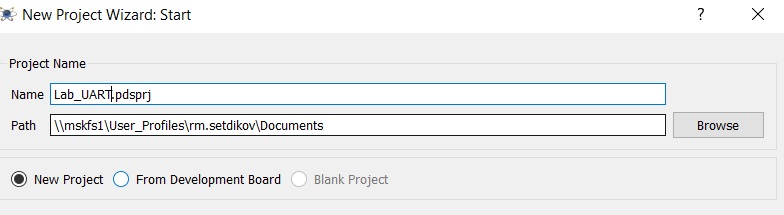


Рисунок 27 – Выбор наименования проекта

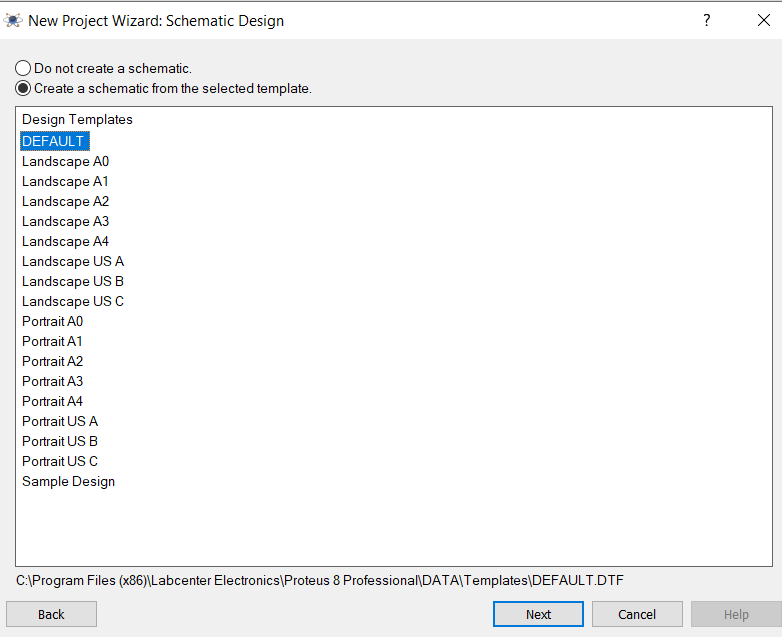


Рисунок 28 – Выбор формата схемы проекта

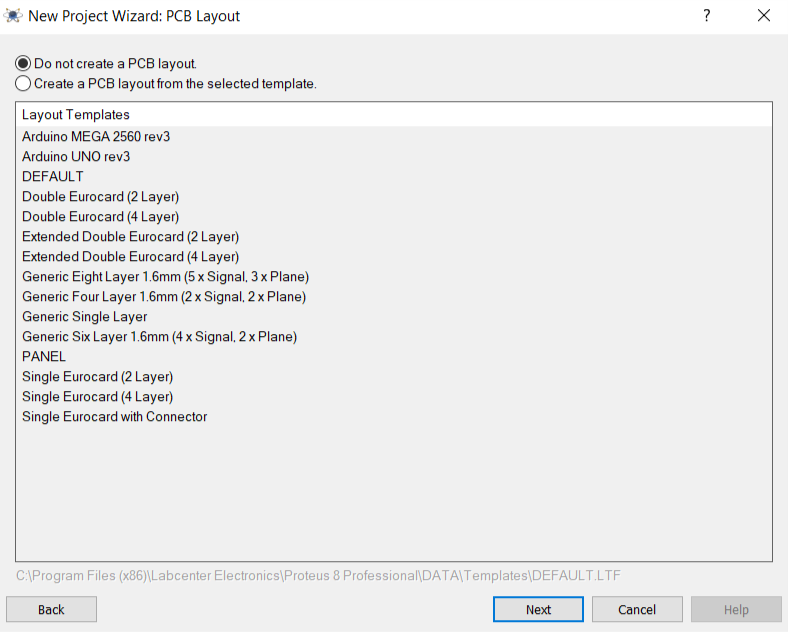


Рисунок 29 – Выбор формата PCB-проекта (для выполнения практических работ этот проект не нужен)

1. В качестве компилятора выбрать SDCC for 8051.

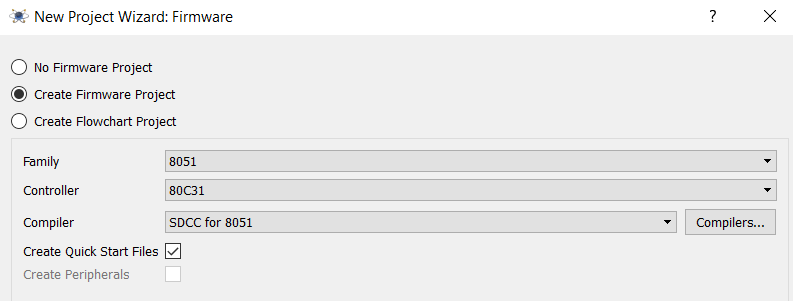


Рисунок 30 – Выбор микроконтроллера и компилятора

Будут созданы две вкладки – со схемой и кодом.

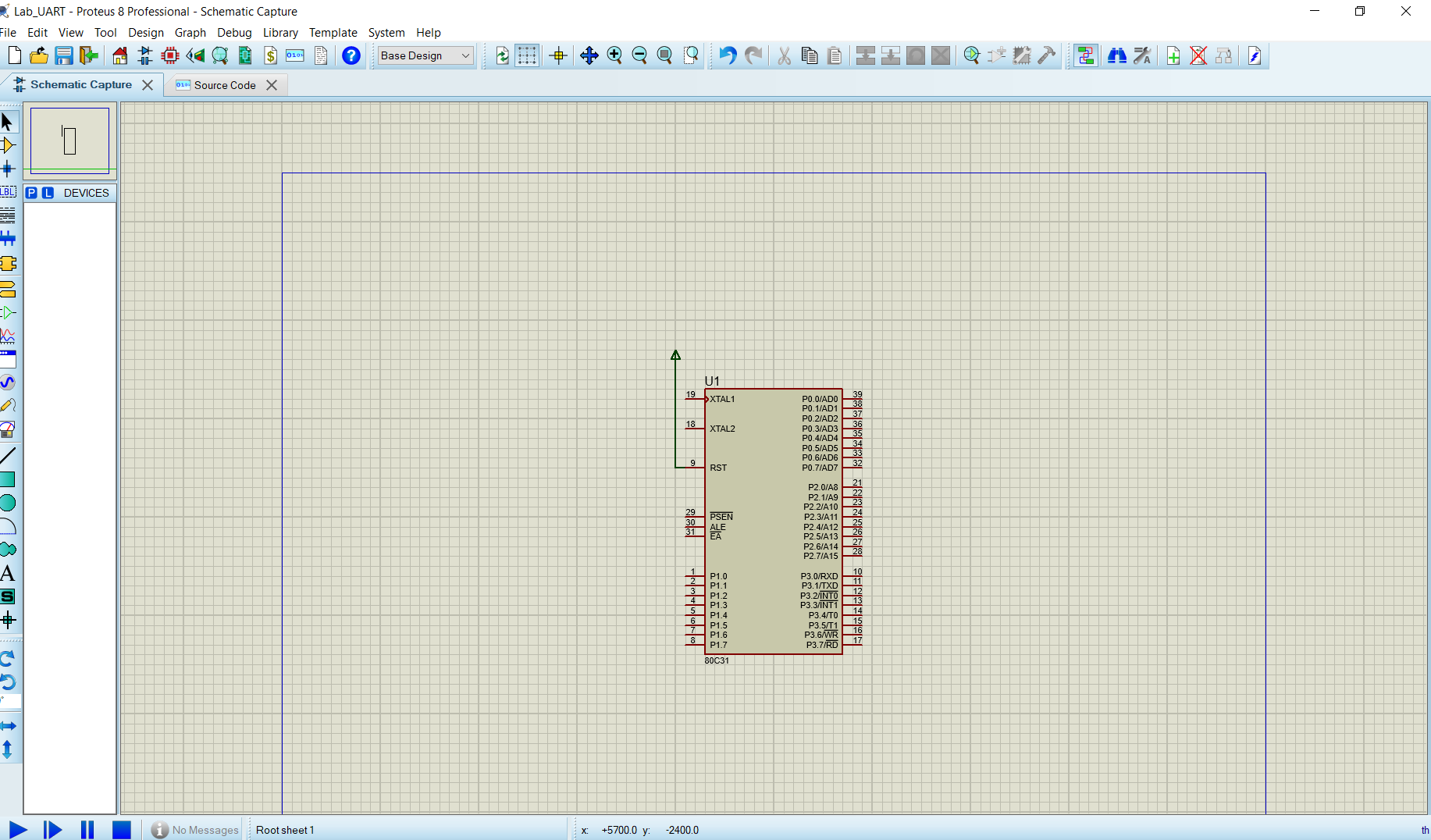


Рисунок 31 – Пример отображения схемы созданного проекта

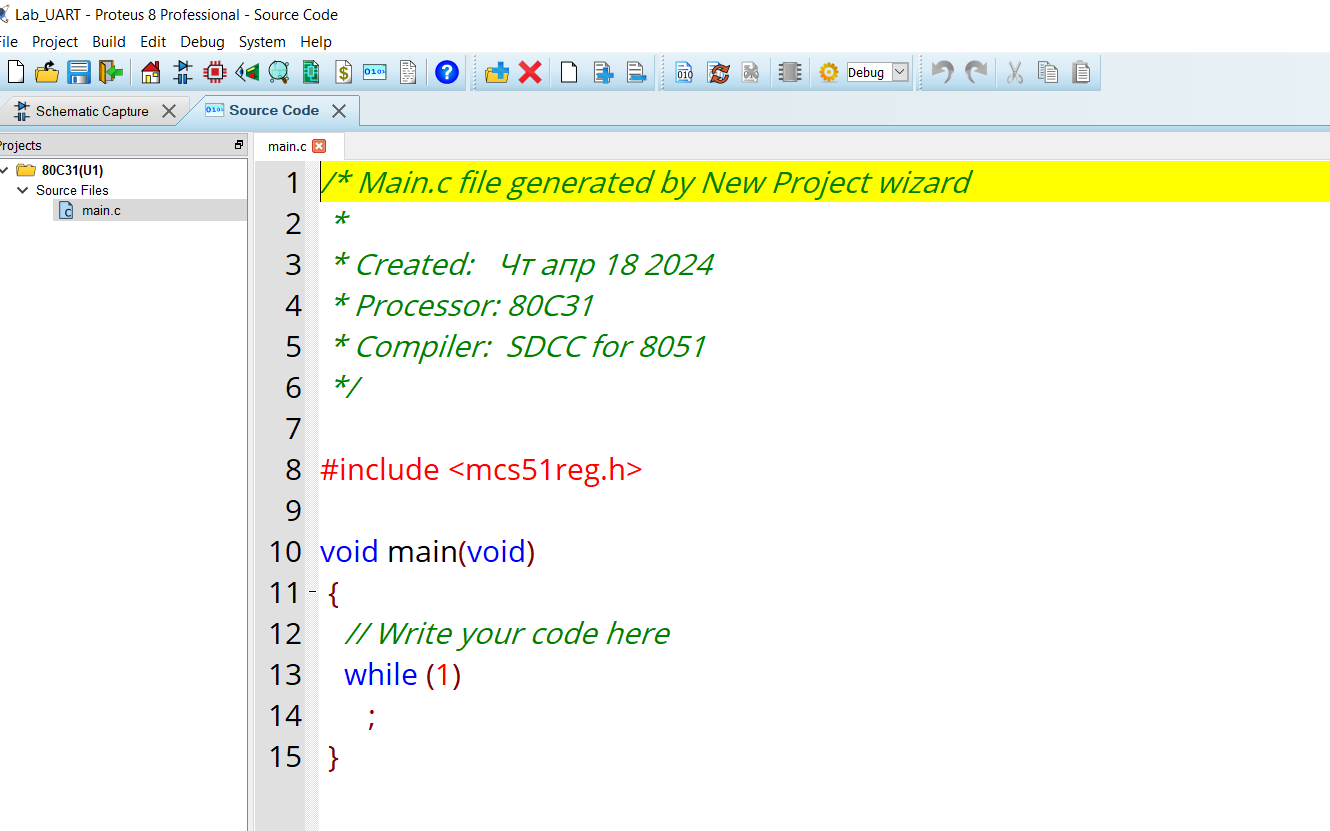


Рисунок 32 – Пример отображения кода созданного проекта

1. Для завершения настройки проекта требуется установить правильную тактовую частоту микроконтроллера, для этого нажать на процессор. Выставить Clock Frequency на 11.059MHz.

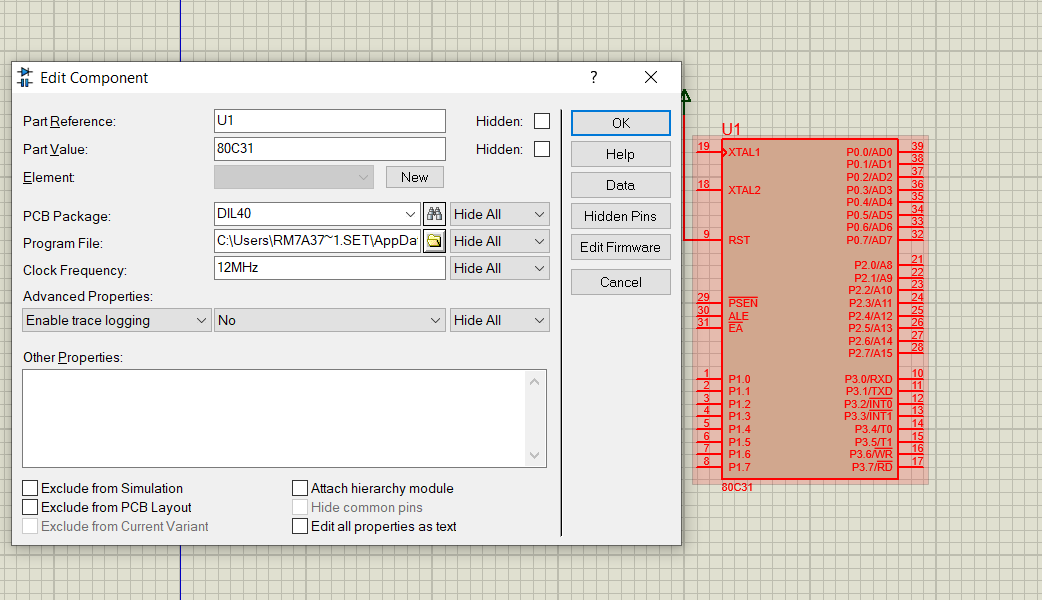


Рисунок 33 – Изменение тактовой частоты микроконтроллера

## Порядок выполнения работы

### Настройка первой программы

Для работы с последовательным портом в Proteus требуется выполнить шаги ниже:

1. Добавить на схему Virtual Terminal.

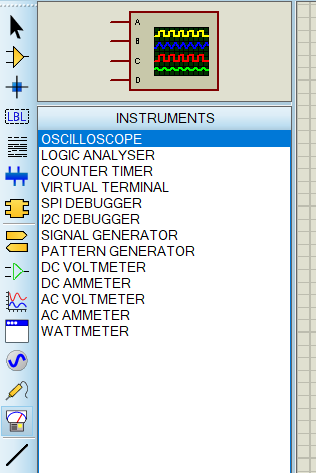


Рисунок 34 – Добавление элемента virtual terminal

1. Соединить RX с TX и TX с RX, как показано на рисунке.

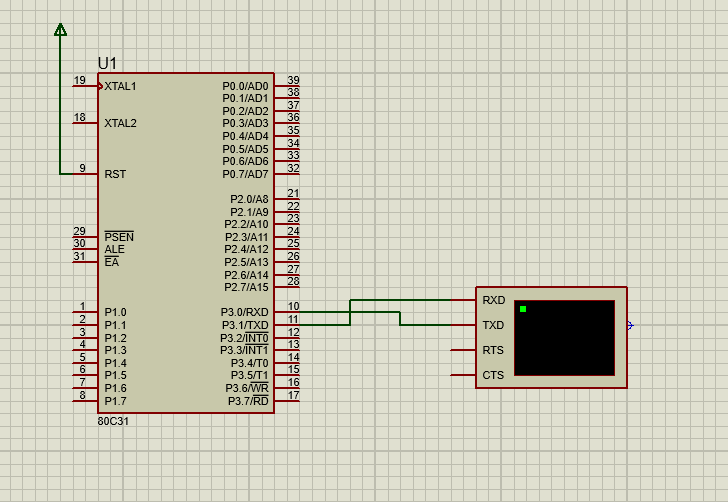


Рисунок 35 – Соединение микроконтроллера с виртуальным портом

1. Настройки оставить по умолчанию.

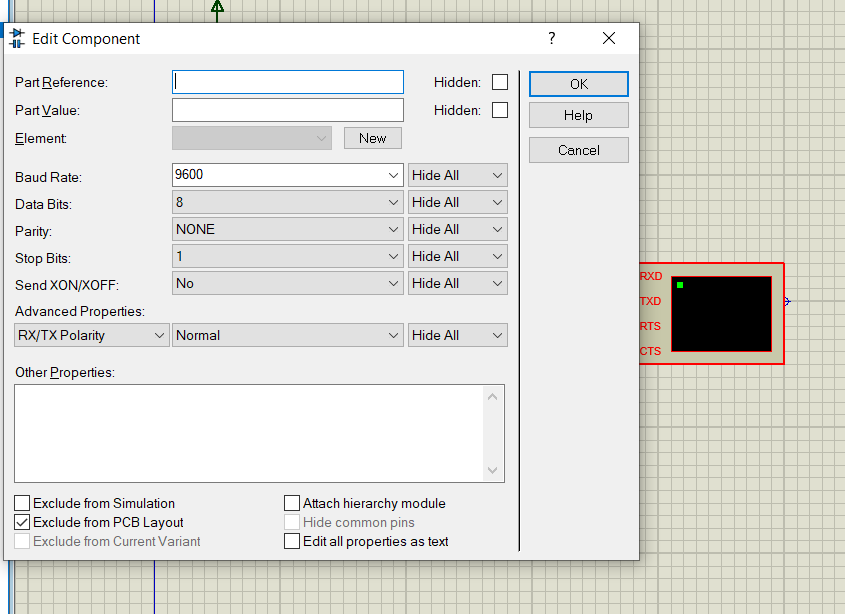


Рисунок 36 – Настройки виртуального порта

1. С помощью следующих функций, написать программу, которая отправляет с небольшим интервалом в последовательный порт последнее принятое значение. Обратите внимание, что сначала необходимо проинициализировать последовательный порт (функция void init\_serial()), то есть вызвать эту функцию до начала всех остальных действий.

|  |
| --- |
| #include <stdint.h>  void delay\_t(uint32\_t ms){//задержка  for (uint32\_t i = 0; i < ms; i++){}  }  void init\_serial(){ //настройка последовательного порта  TMOD = 0x20;  SCON = 0x50;  TH1 = 0xFD;  TL1 = 0xFD;  TR1 = 1;  }  void putc(uint8\_t input){//посылка 1 байта  SBUF = input;  while (TI != 1){}  TI = 0;  }  int UART\_is\_not\_Empty(){//показывает, если ли в буфере непрочитанное значение  return RI;  }  char UART\_read(){//возвращает значение из буфера  RI = 0;  return SBUF;  } |

Если терминал не открылся автоматически, то открыть вручную, выбрав в верхней строке меню «Debug->Virtual Terminal».

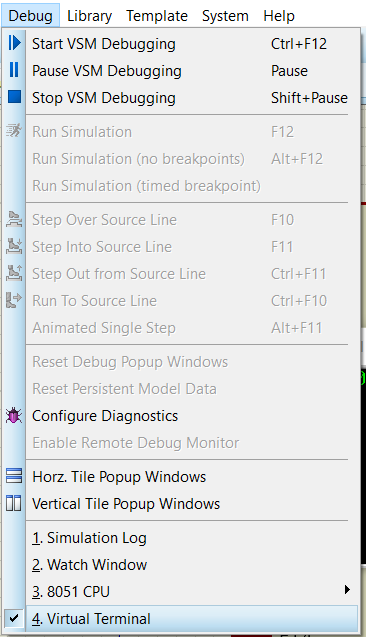


Рисунок 37 – Вызов «Virtual Terminal» из меню

### Подключение последовательного порта и установка 7-сегментных индикаторов

Необходимо создать связь между приложениями через виртуальные порты, для этого требуется выполнить следующие шаги.

1. Найти в библиотеке компонентов модуль COMPIM и подключить его как показано на рисунке 39.

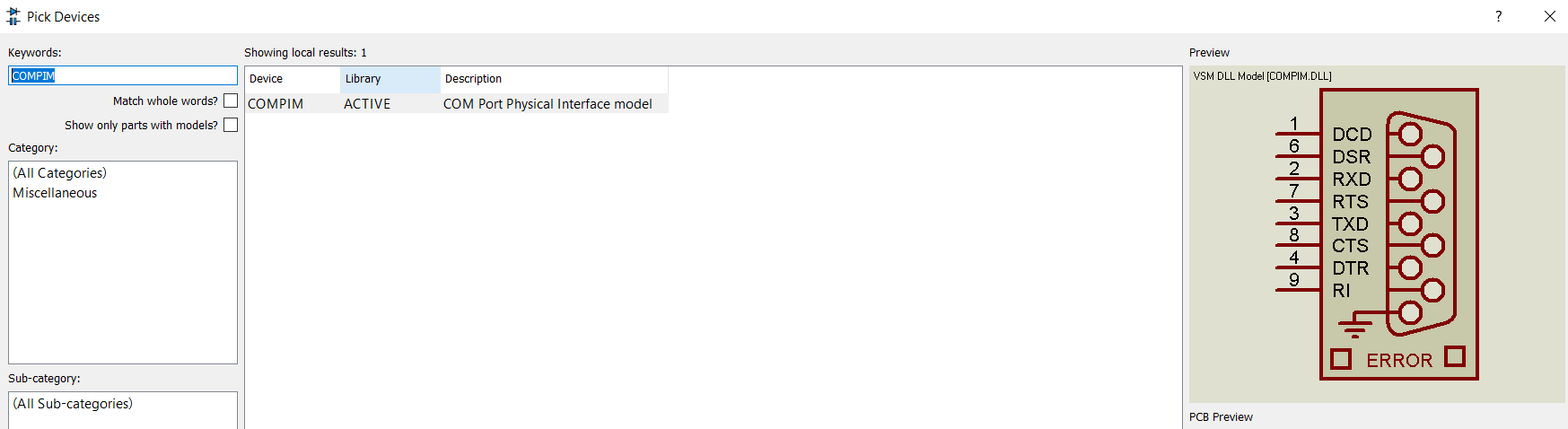


Рисунок 38 – COMPIM в меню компнентов

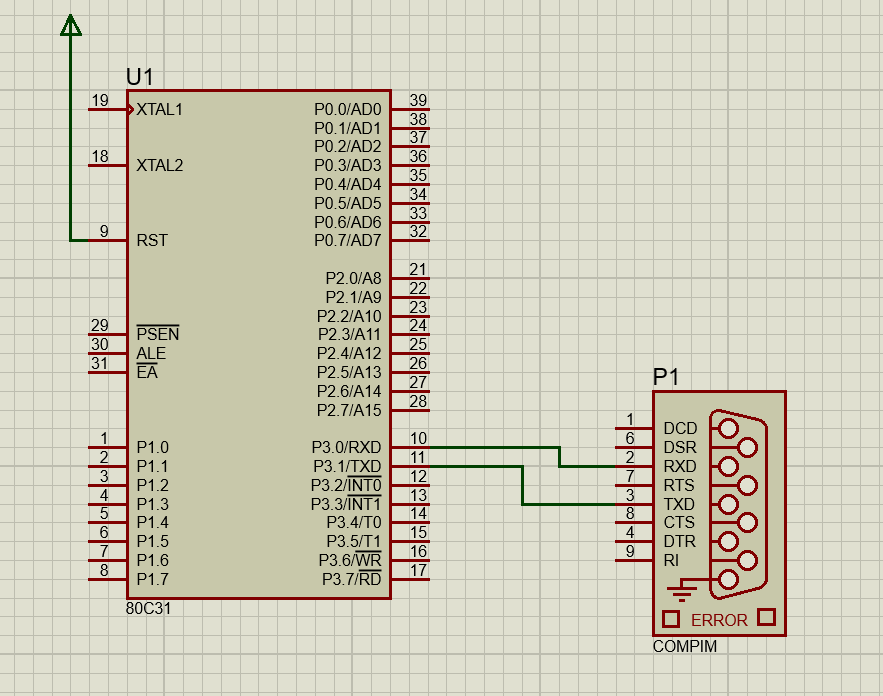


Рисунок 39 – Подключение COMPIM к микроконтроллеру

1. Запустить VSPE. Создать новое устройство (тип «Pair»). Найти доступные COM – порты. В статусе должно появиться состояние Ready (подробная инструкция в п. 4.2.4).
2. Открыть приложение «Terminal-v.1.9b». Подключиться к одному из созданных портов.

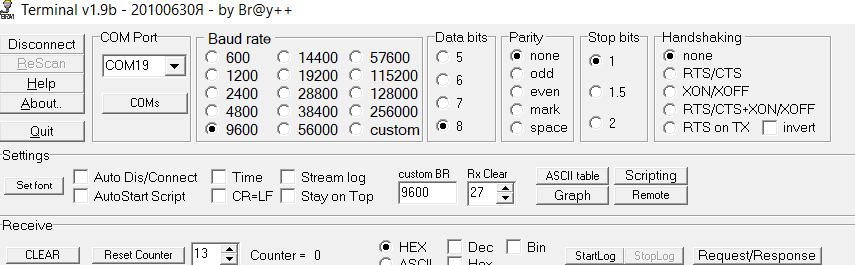


Рисунок 40 – Пример настройки последовательного порта в приложении «Terminal-v.1.9b»

1. Настроить COMPIM (настроить параметры и подключиться ко второму созданному порту).

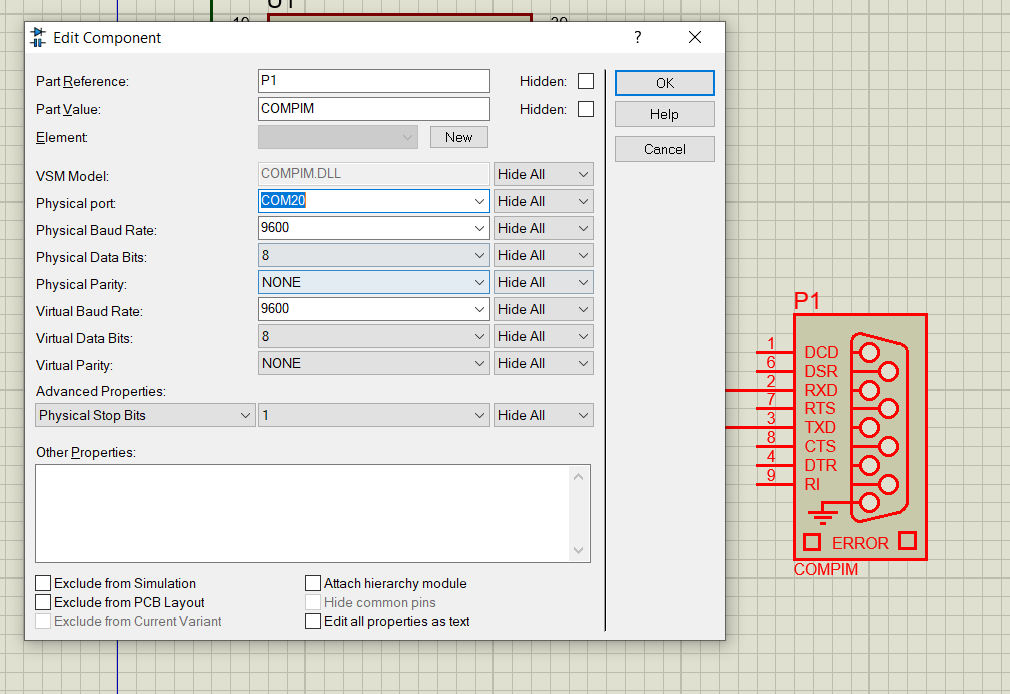


Рисунок 41 – Пример настройки последовательного порта в Proteus

1. Запустить программу Proteus. В результате терминал и Proteus связаны.
2. Добавить на схему 2 семисегментных индикатора с общим анодом.

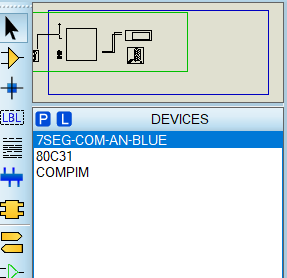


Рисунок 42 – Выбор 7-сегментного индикатора в Proteus

1. Подключить семисегментные индикаторы к портам Р0 и Р1.

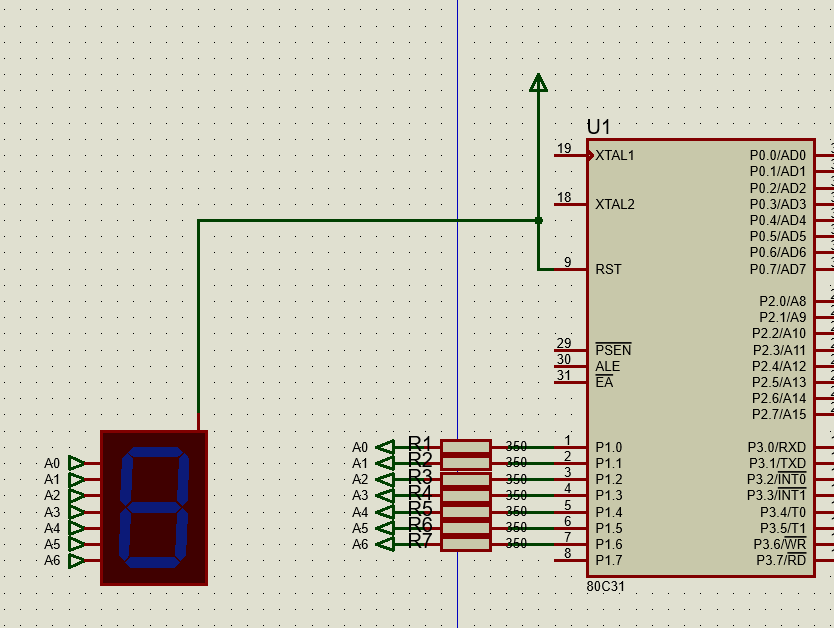


Рисунок 43 – Вариант подключения индикатора

## Задания

Создать новый проект, именуя его: Фамилия\_LR5\_Variant№.

Задание выполнять в соответствие с выданным вариантом. Работу разработанного приложения продемонстрировать преподавателю, ответить на дополнительные вопросы, получить оценку. Архив проекта приложить к отчету.

**Таблица 6 – Варианты заданий**

| **Номер варианта** | **Приложения Qt** | **Функционал** | **Функционал в Proteus** |
| --- | --- | --- | --- |
| 1 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение от 0 до 255, а микроконтроллер выводит на индикаторы значение в 16-ричной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 2 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение от 0 до 63, а микроконтроллер выводит на индикаторы значение в 8-ричной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 3 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение в 2-ичной СС (0-1100011), а микроконтроллер выводит на индикаторы значение в 10-чной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 4 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение в 2-ичной СС (0- 11111111), а микроконтроллер выводит на индикаторы значение в 16-ричной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 5 | Кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений (используется для счета нажатий). | Приложение Qt отправляет в последовательный порт количество нажатий на кнопку, а микроконтроллер выводит на индикаторы значение в 10-чной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 6 | Кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений (используется для счета нажатий). | Приложение Qt отправляет в последовательный порт количество нажатий на кнопку, а микроконтроллер выводит на индикаторы значение в 16-ричной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 7 | Кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений (используется для счета нажатий). | Приложение Qt отправляет в последовательный порт количество нажатий на кнопку, а микроконтроллер выводит на индикаторы значение в 8-ричной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 8 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение от 0 до 9, а микроконтроллер выводит на индикаторы значение от 0 до полученного значения с небольшой задержкой (delay\_t(100)). | 7-сегментный индикатор подключен к порту Р0. |
| 9 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение от 0 до 9, а микроконтроллер выводит на индикаторы значение от полученного значения до 0 с небольшой задержкой (delay\_t(100)). | 7-сегментный индикатор подключен к порту Р1. |
| 10 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение от 0 до 255, а микроконтроллер выводит на индикаторы остаток от деления на 16 прочитанного числа. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 11 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет число (0-255), микроконтроллер выводит на индикаторы сумму цифр этого числа. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 12 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число (0-255), микроконтроллер вычисляет произведение цифр числа и выводит на индикаторы. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 13 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число (0-255), микроконтроллер проверяет четность и выводит "C" (четное) или "H" (нечетное). | 7-сегментный индикатор подключен к порту Р0. |
| 14 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число, микроконтроллер определяет количество четных цифр и выводит на индикаторы. | 7-сегментный индикатор подключен к порту Р1. |
| 15 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число, микроконтроллер определяет количество нечетных цифр и выводит на индикаторы. | 7-сегментный индикатор подключен к порту Р0. |
| 16 | Два однострочных текстовых поля (один для ввода, другой для вывода), кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число от 0 до 9, микроконтроллер выводит значение на индикатор, прибавляет к числу 1 и отправляет обратно в последовательный порт полученное значение. Qt-приложение отображает во втором текстовом поле полученный результат. | 7-сегментный индикатор подключен к порту Р0. |
| 17 | Два однострочных текстовых поля (один для ввода, другой для вывода), кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число от 0 до 9, микроконтроллер выводит значение на индикатор, определяет четность, и, в случае четности, отправляет обратно в последовательный порт, иначе отправляет символ «N». Qt-приложение отображает во втором текстовом поле полученный результат. | 7-сегментный индикатор подключен к порту Р1. |
| 18 | Два однострочных текстовых поля (один для ввода, другой для вывода), кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число от 0 до 9, микроконтроллер выводит значение на индикатор, определяет четность, и, в случае четности, отправляет в последовательный порт символ «С», иначе отправляет символ «N». Qt-приложение отображает во втором текстовом поле полученный результат. | 7-сегментный индикатор подключен к порту Р0. |
| 19 | Кнопка подключения/отключения к последовательному порту, 2 кнопки для отправки сообщений (используется для счета нажатий). | Приложение Qt отправляет в последовательный порт количество нажатий на кнопки, а микроконтроллер выводит на первый индикатор количества нажатий на 1 кнопку в Qt, второй – на другую в 10-ричной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 20 | Два однострочных текстовых поля (один для ввода, другой для вывода), кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число от 0 до 9, микроконтроллер выводит значение на первый индикатор, на втором индикаторе отображается любое значение (постоянно). Далее МК прибавляет к числу на 1 индикаторе значение со 2 индикатора и отправляет обратно в последовательный порт полученное значение. Qt-приложение отображает во втором текстовом поле полученный результат. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 21 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет символ, микроконтроллер выводит на индикаторы ASCII-код полученного символа в 16-ричном формате. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 22 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение от 0 до 127, а микроконтроллер выводит на индикаторы удвоенное значение в 16-ричной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 23 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение от 0 до 85, а микроконтроллер выводит на индикаторы утроенное значение в 16-ричной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 24 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение от 0 до 11111111, а микроконтроллер инвертирует его и выводит на индикаторы полученное значение в 10-чной СС. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 25 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет символ, микроконтроллер выводит на индикаторы инвертированный ASCII-код полученного символа в 16-ричном формате. (Например, символ «0» его ASCII-код 60 (в HEX или 00110000 в 2-чной СС), инверсия это 11001111, или в HEX CF – будет выведено на индикаторы) | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 26 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число от 0 до 255, микроконтроллер определяет количество 1 в двоичной записи числа и выводит полученное значение на индикатор. | 7-сегментный индикатор подключен к порту Р1. |
| 27 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число от 0 до 255, микроконтроллер определяет количество 0 в двоичной записи числа и выводит полученное значение на индикатор. | 7-сегментный индикатор подключен к порту Р0. |
| 28 | Однострочное текстовое поле вывода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений (используется для счета нажатий). | Приложение Qt отправляет в последовательный порт количество нажатий на кнопку, а микроконтроллер выводит на индикаторы значение. По достижении 10 нажатий индикаторы обнуляются, а микроконтроллер отправляет обратно в последовательный порт символ «N». Qt-приложение отображает текстовом поле полученный результат, счетчик нажатий обнуляется. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 29 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Qt-приложение отправляет число (10-255), микроконтроллер вычисляет модуль разности первой и последней цифр числа и выводит на индикаторы. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |
| 30 | Однострочное текстовое поле ввода, кнопка подключения/отключения к последовательному порту, кнопка для отправки сообщений. | Приложение Qt отправляет в последовательный порт значение от 0 до 255, а микроконтроллер выводит на индикаторы остаток от деления на 8 прочитанного числа. | Два 7-сегментных индикатора подключены к портам Р0 и Р1. |

# Управление ЦАП в Proteus

## Цели работы

Освоить методы управления ЦАП и АЦП при помощи микроконтроллера.

## Краткие теоретические сведения

### Цифро-аналоговый преобразователь

Цифро-аналоговый преобразователь (ЦАП) – это устройство для преобразования цифрового кода в аналоговый сигнал по величине, пропорциональной значению кода.

ЦАП применяются для связи цифровых управляющих систем с устройствами, которые управляются уровнем аналогового сигнала. Также, ЦАП является составной частью во многих структурах аналого-цифровых устройств и преобразователей.

Напряжение на выходе 8-разрядного ЦАП устанавливается по следующему принципу: опорное напряжение делится на число каналов (8 бит = 256 каналов), а затем умножается на значение на управляющих входах (D0 –младший бит, D7 - старший) (Vout = Vref \* ).

### Аналогово-цифровой преобразователь

АЦП (аналогово-цифровой преобразователь) – устройство, преобразующее аналоговый сигнал в цифровой код. Как правило, микросхемы АЦП имеют:

1. Аналоговый вход (Vin) – вход, на который подается оцифровываемое аналоговое напряжение.
2. Вход положительного опорного напряжения (VRef+) – вход, на который подается высокостабильное напряжение, соответствующее выходному коду (111 … 11) АЦП.
3. Вход отрицательного опорного напряжения (VRef-) – вход, на который подается высокостабильное напряжение, соответствующее коду (000 … 0) АЦП. Чаще всего, на этот вход подается напряжение, равное 0 (Gnd).
4. Для случая, когда VRef- = 0, выходной код АЦП считается по формуле (N – число разрядов преобразователя):

Value =

Для 8ми разрядного АЦП - D0 – младший бит кода, D7 –старший.

## Порядок выполнения работы

### Построение схемы с ЦАП в Proteus

Перечень элементов в среде Proteus: COMPIM, MOTOR-DC, DAC\_8.

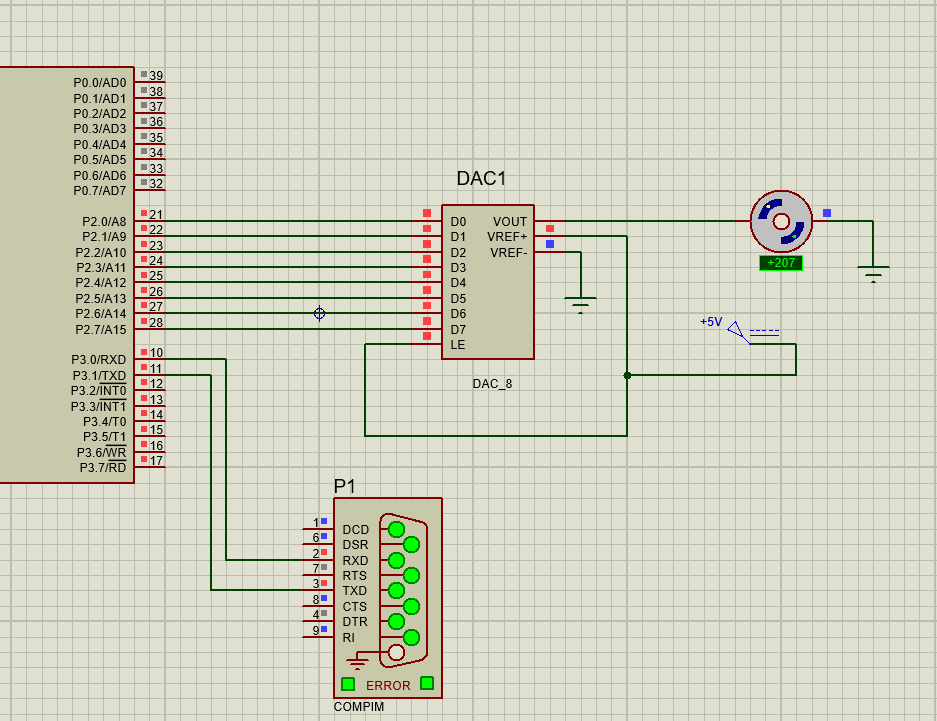


Рисунок 44 – Схема соединения ЦАП и двигателя

Для построения схемы на рисунке 44 требуется выполнить шаги ниже.

1. В качестве источника опорного напряжения + 5В необходимо использовать инструмент DC с вкладки GENERATORS.

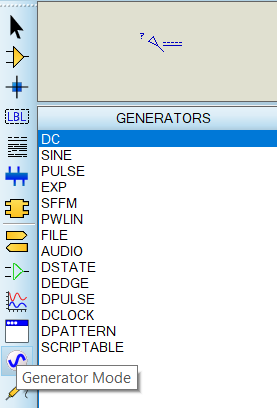


Рисунок 45 – Выбор источника напряжения

1. В качестве ЦАП использовать DAC\_8.

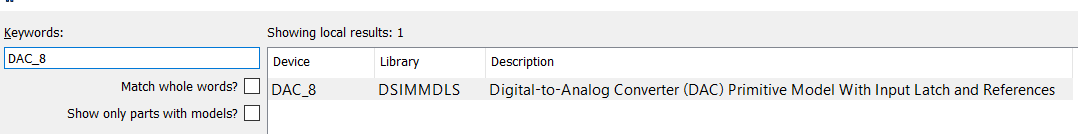


Рисунок 46 – Выбор цифро-аналогового преобразователя

1. В качестве двигателя можно выбрать один из списка.



Рисунок 47 – Выбор двигателя

1. Для контроля выходного напряжения ЦАП можно использовать инструмент Voltage с вкладки ProbeMode.

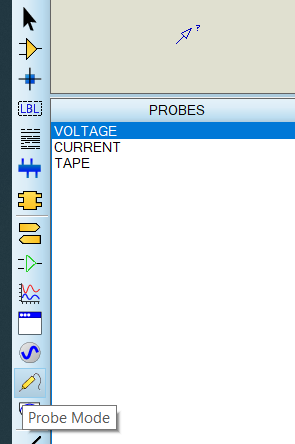


Рисунок 48 – Выбор инструмента для измерения напряжения

Также для контроля напряжения можно использовать осциллограф OSCILLOSCOPE или вольтметр с вкладки INSTRUMENTS.

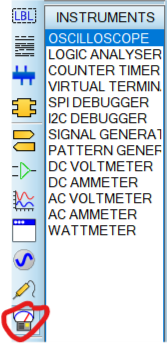


Рисунок 49 – Инструмент «Осциллограф»

Для работы с последовательным портом использовать функции из п. 5.3.1.

### Построение схемы с АЦП в Proteus

Перечень элементов: «ADС\_8», «RES», «BUTTON», «POT», «COMPIM».

Источники сигналов: «DC» с вкладки «GENERATORS».

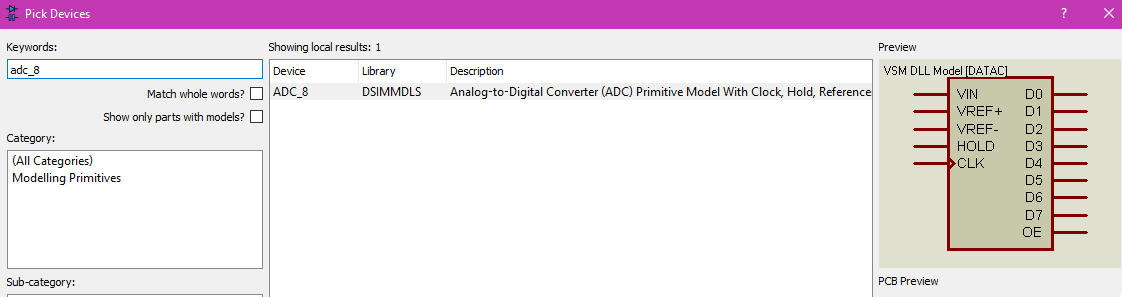


Рисунок 50 – Выбор АЦП

На рисунке 50 показана микросхема 8-разрядного аналогово-цифрового преобразователя. Контакт OE отвечает за выдачу кода на выходы D0 – D7 АЦП. (При высоком уровне – выход кода на контакты D0-D7 разрешен, при низком уровне – выходы в так называемом состоянии высокого импеданса).

При высоком напряжении на контакте HOLD будет оцифровано напряжение, находящееся на входе Vin по фронту сигнала CLK, при низком уровне – значение на выходе не изменяется. Значения на выходе изменяется при переходе из низкого уровня в высокий (фронте) сигнала CLK.

Для получения кода на выходе АЦП необходимо выполнить следующую последовательность:

1. Подать требуемые напряжения на входы Vin, Vref+, VRef-;
2. Подать высокое напряжение на вход «Hold».
3. Подать единичный импульс на вход «Clk».
4. На входе «OE» (Output Enable) – поддерживать напряжение высокого уровня постоянно или только во время чтения.

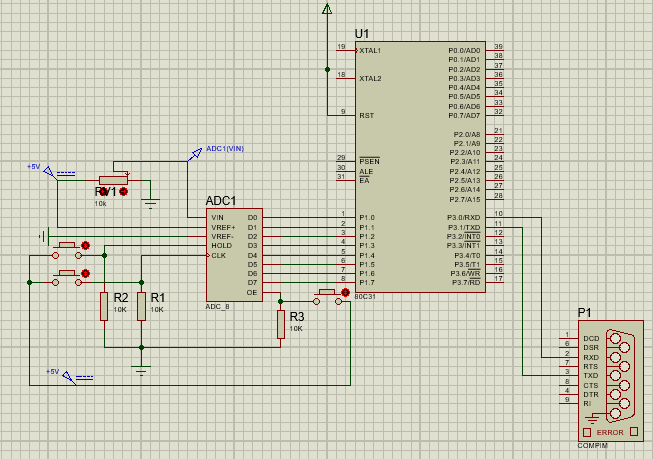


Рисунок 51 – Схема подключения аналогово-цифрового преобразователя к микроконтроллеру

1. Собрать схему, представленную на рисунке 51 без подключения контактов D0 ... D7 к портам микроконтроллера. В качестве источника опорного напряжения + 5В необходимо использовать инструмент DC с вкладки GENERATORS (рис. 45). Также для контроля напряжения можно использовать осциллограф OSCILLOSCOPE или вольтметр с вкладки INSTRUMENTS (рис. 49).
2. Выполнить описанную последовательность оцифровки входного аналогового сигнала АЦП.
3. Отключить кнопки и подключить контакты OE, CLK, HOLD, D0 ... D7 микросхемы АЦП к микроконтроллеру (место и порядок подключения определяет разработчик).

## Задания

Создать новый проект, именуя его: Фамилия\_LR6\_Variant№.

Задание выполнять в соответствие с выданным вариантом. Работу разработанного приложения продемонстрировать преподавателю, ответить на дополнительные вопросы, получить оценку. Архив проекта приложить к отчету.

**Таблица 7 – Варианты заданий**

| **Номер варианта** | **Задание** |
| --- | --- |
| 1 | Разработать программно-аппаратный комплекс, позволяющий из приложения ПК, спроектированного в фреймворке Qt управлять скоростью вращения двигателя схемы, показанной на рис. 44, разработанной в среде Proteus (Выбор элемента, в котором устанавливается скорость вращения двигателя – на усмотрение выполняющих практическую работу). Использовать микроконтроллер семейства i8051. Тактовую частоту микроконтроллера установить равной 11.059 МГц. Скорость обмена по последовательным портам установить равной 9600 бит/с. С помощью приложения VSPE организовать пару виртуальных портов. Имя одного из виртуальных портов использовать, как имя для объекта QSerialPort приложения Qt, имя другого использовать, как имя объекта COMPIM среды Proteus. |
| 2 | Разработать программно-аппаратный комплекс, который выводит на экран ПК (для отображения выбрать самостоятельно наиболее подходящий объект) текущее значение напряжения источника с интервалом 1 раз в секунду.  Использовать микроконтроллер семейства i8051. Тактовую частоту микроконтроллера установить равной 11.059 МГц. Скорость обмена по последовательным портам установить равной 9600 бит/с. С помощью приложения VSPE организовать пару виртуальных портов. Имя одного из виртуальных портов использовать, как имя для объекта QSerialPort приложения Qt, имя другого использовать, как имя объекта COMPIM среды Proteus.  На измерительный вход АЦП подавать сигнал с движка потенциометра, подключенного к источнику постоянного напряжения, как показано на рисунке 51 (место и порядок подключения контактов OE, CLK, HOLD, D0 ... D7 микросхемы АЦП определить самостоятельно). Разработанные в Proteus и Qt приложения соединять виртуальными последовательными портами. В качестве АЦП использовать компонент ADC\_8 библиотеки моделей примитивов DSIMMDLS. |

# Темы проектов семинара по направлению «Системы контроля и управления»

Темы проектов рассчитаны на выполнение в средах разработки Qt и Proteus. Проект выполняется бригадой. По желанию бригады проект (при некоторой корректировке темы) можно выполнять на реальных отладочных платах типа UNI-DS6. Базовый микроконтроллер проекта - i8051, смена микроконтроллера - по согласованию с преподавателем. Выбранные темы не должны повторяться. После выполнения проводится защита проекта. На защите обязательно присутствие всех членов бригады. Срок сдачи (защиты) проекта - не позже, чем за 2 недели до конца 4го модуля. Для проверки степени участия каждого из членов бригады в работе над проектом каждому участнику будет выдано индивидуальное задание, заключающиеся в некотором изменении алгоритма работы программно-аппаратного комплекса, которое необходимо будет выполнить в ограниченное время

**Общее задание для всех проектов**:

Разработать управляемый с ПК программно-аппаратный комплекс согласно выбранной тематике, поддерживающий 3 языка (русский, английский, немецкий). Настройки ПО (размер и положение окон приложения), а также пороги, калибровочные коэффициенты для аппаратуры и т.д. располагать в XML файле в директории приложения. Работу с аппаратурой эмулировать в САПР «Proteus».

**Темы проектов**:

1. Разработать АЦП последовательных приближений на основе ЦАП (DAC\_08) и компаратора. Амплитуду измеренного напряжения индицировать в окне приложения на ПК в виде числа с плавающей точкой с точностью +-1%, а также на графике. Аналоговое напряжение задавать с помощью инструмента «DC» и потенциометра (POT) или «Sine» с вкладки «Generators». Считать напряжение нормальным, если его амплитуда находится в диапазоне между нижним и верхним порогами. Пороги устанавливать в отдельном окне и сохранять в файле XML. В случае, если напряжение нормальное – отображать его зеленым цветом, при превышении – красным, если напряжение ниже порога, то синим. Предполагаемая схемотехника на рисунках 52, 53. Перечень компонентов: AT89C51, COMPI, DAC\_8, POT.

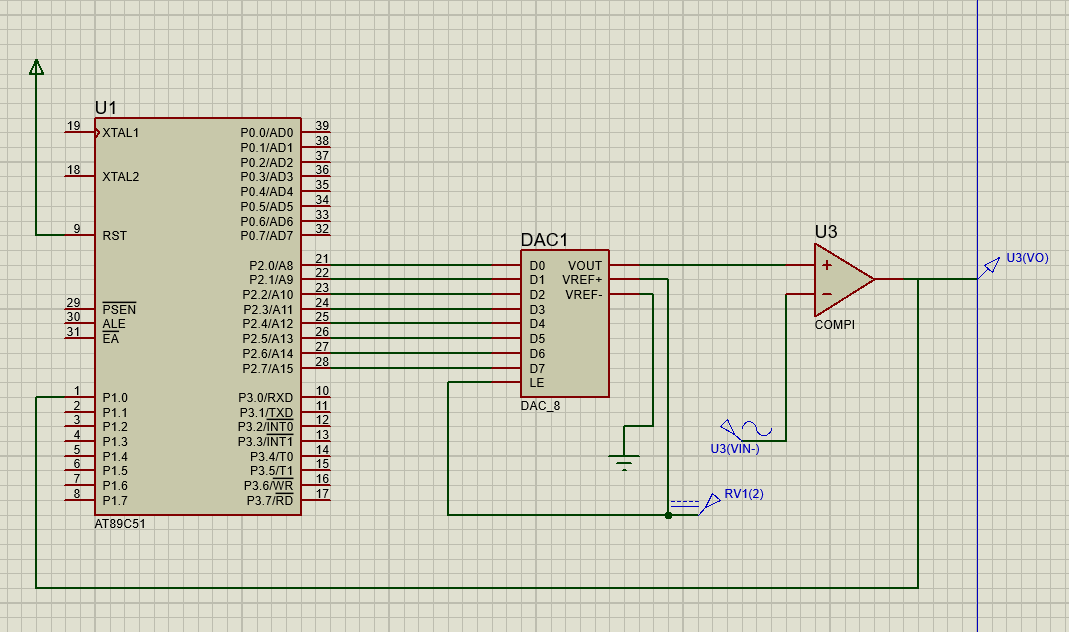


Рисунок 52 – Вариант построения схемы проекта варианта 1

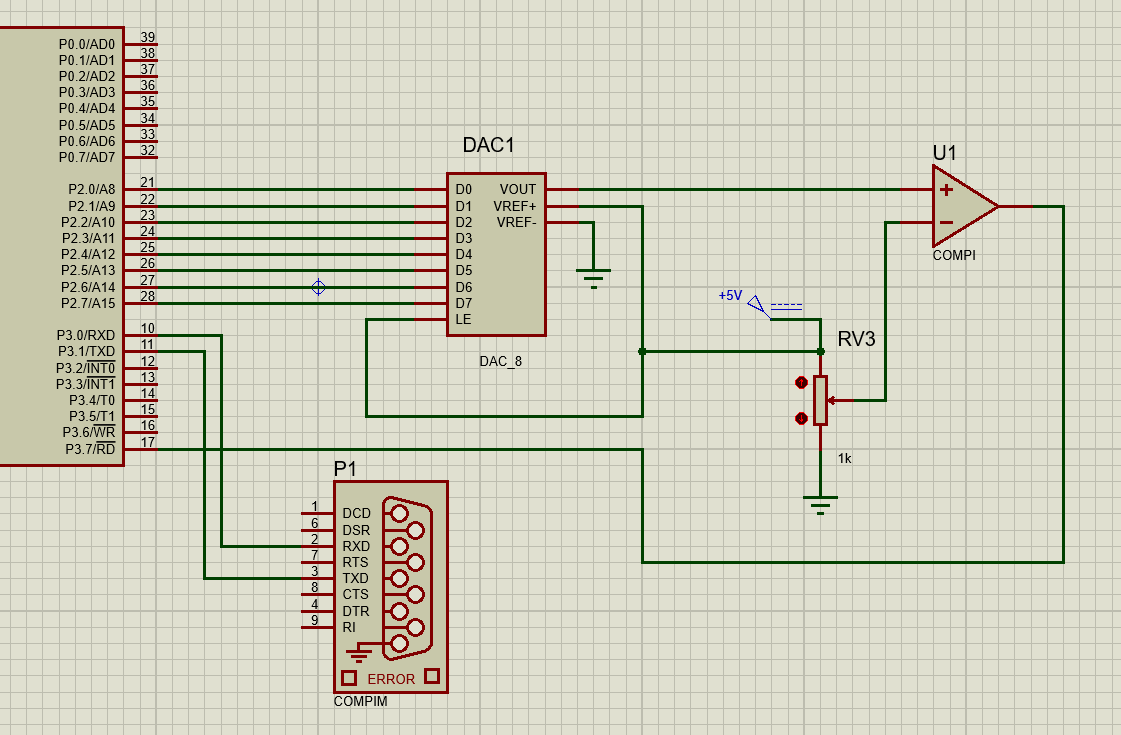


Рисунок 53 – Вариант построения схемы проекта варианта 1

1. Разработать комплекс, измеряющий частоту колебаний симметричного транзисторного мультивибратора (питание 7 Вольт). Частота колебаний мультивибратора задаётся переменными резисторами (изменять симметрично). Измеренную частоту индицировать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимой частоты и сохранять их в XML файле. В случае, если измеренная частота в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если частота ниже нижнего порога, то синим. Предполагаемая схемотехника и перечень элементов на рисунках 55, 56. Выход мультивибратора (точка, подключенная к осциллографу) подключается на ножке порта P1.

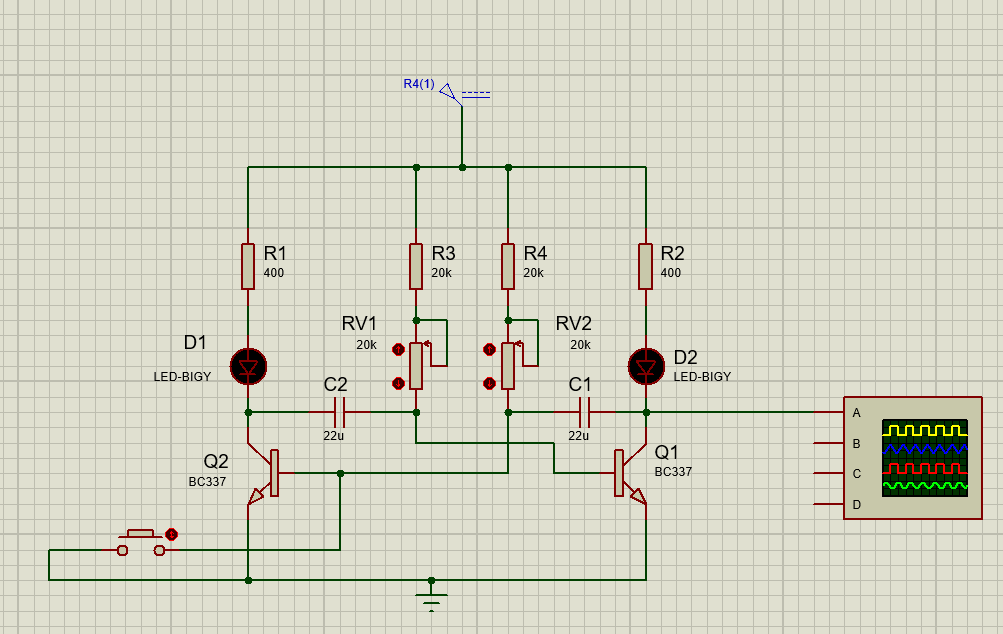


Рисунок 55 – Схема симметричного транзисторного мультивибратора

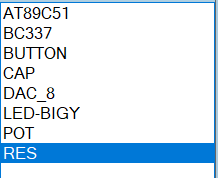


Рисунок 56 – Перечень компонентов симметричного транзисторного мультивибратора

1. Разработать комплекс для задания угла поворота мотора (MOTOR-PWMSERVO, питание 5 В) с помощью программного ШИМ. Внимательно изучить параметры данного элемента. Угол поворота задавать в градусах в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимых углов (заданных программно), а также критических углов (заданных в настройках прибора в Proteus) и сохранять их в XML файле. В случае, если заданный угол в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если частота ниже нижнего порога, то синим. Также в окне должен отображаться реальный угол поворота (совпадает с заданным углом, если заданный угол в диапазоне между двумя критическими углами, равен одному из критических в обратном случае). Предполагаемая схемотехника на рисунке 57.

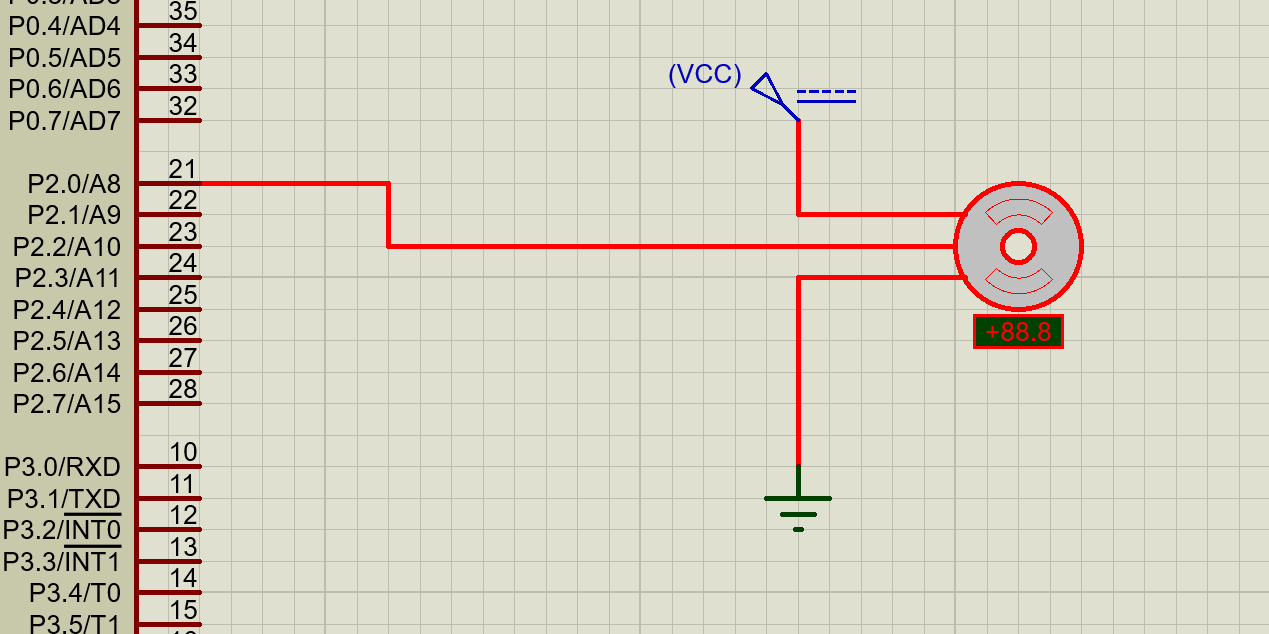


Рисунок 57 – Схема подключения шагового двигателя

1. Разработать комплекс по управление скоростью двигателя (MOTOR-DC) (питание 12 В) с помощью программного ШИМ. Скорость задавать в оборотах в минуту в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимого диапазона скоростей и сохранять их в XML файле. В случае, если заданная скорость в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если частота ниже нижнего порога, то синим. Предполагаемая схемотехника на рисунке 58. Перечень элементов: MOTOR-DC, транзистор BC337, RES.

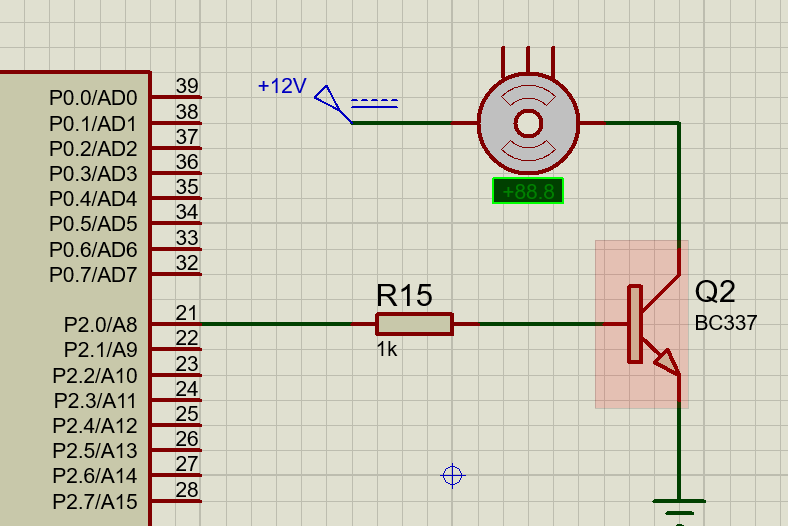


Рисунок 58 – Схема комплекса управления скоростью двигателя

1. Разработать звуковой индикатор (Speaker, library ACTIVE), с громкостью, управляемой ЦАП, и программно-управляемой частотой звучания. Громкость задавать в целых относительных единицах от 0 до 10 (цвет и значение должны совпадать с соответствующей индикацией в Proteus) элементом QSpinBox в окне приложения ПК. Частоту звучания также задавать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимого диапазона частот и сохранять их в XML файле. В случае, если заданная частота в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если частота ниже нижнего порога, то синим. Опорное напряжение ЦАП и параметр Speaker Nominal input задать 12 Вольт. Предполагаемая схемотехника на рисунке 59. Между базой транзистора и ножной микроконтроллера можно поставить резистор в 1 кОм, однако с точки зрения схемотехники портов микроконтроллера серии 8051 предложенный вариант также верный. Перечень элементов: DAC\_08, BC337, Speaker.

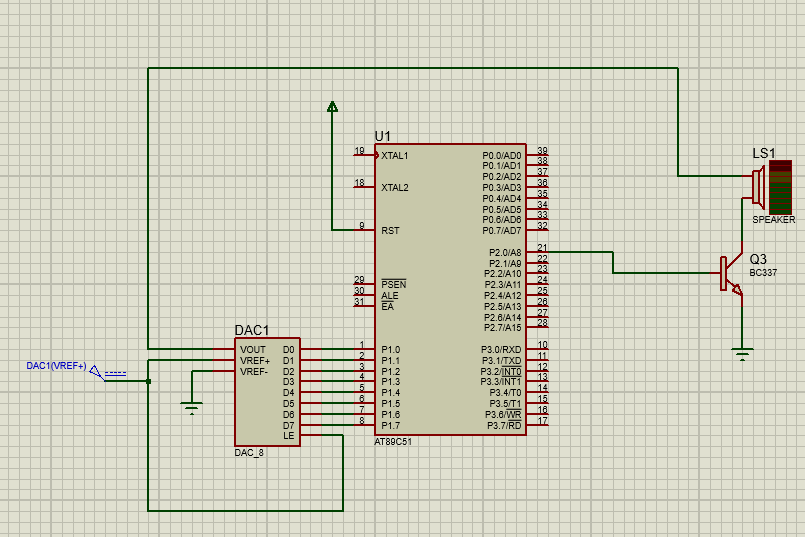


Рисунок 59 – Схема звукового индикатора

1. Разработать комплекс по управлению углом шагового двигателя (MOTOR-BISTEPPER). Скорость задавать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимого диапазона скоростей и сохранять их в XML файле. В случае, если заданная скорость в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если частота ниже нижнего порога, то синим. В случае если скорость равна нулю, необходимо иметь возможность задать положение привода в градусах. Предполагаемая схемотехника на рисунке 60. Перечень элементов: MOTOR-BISTEPPER, L293D (Питание 12 Вольт).

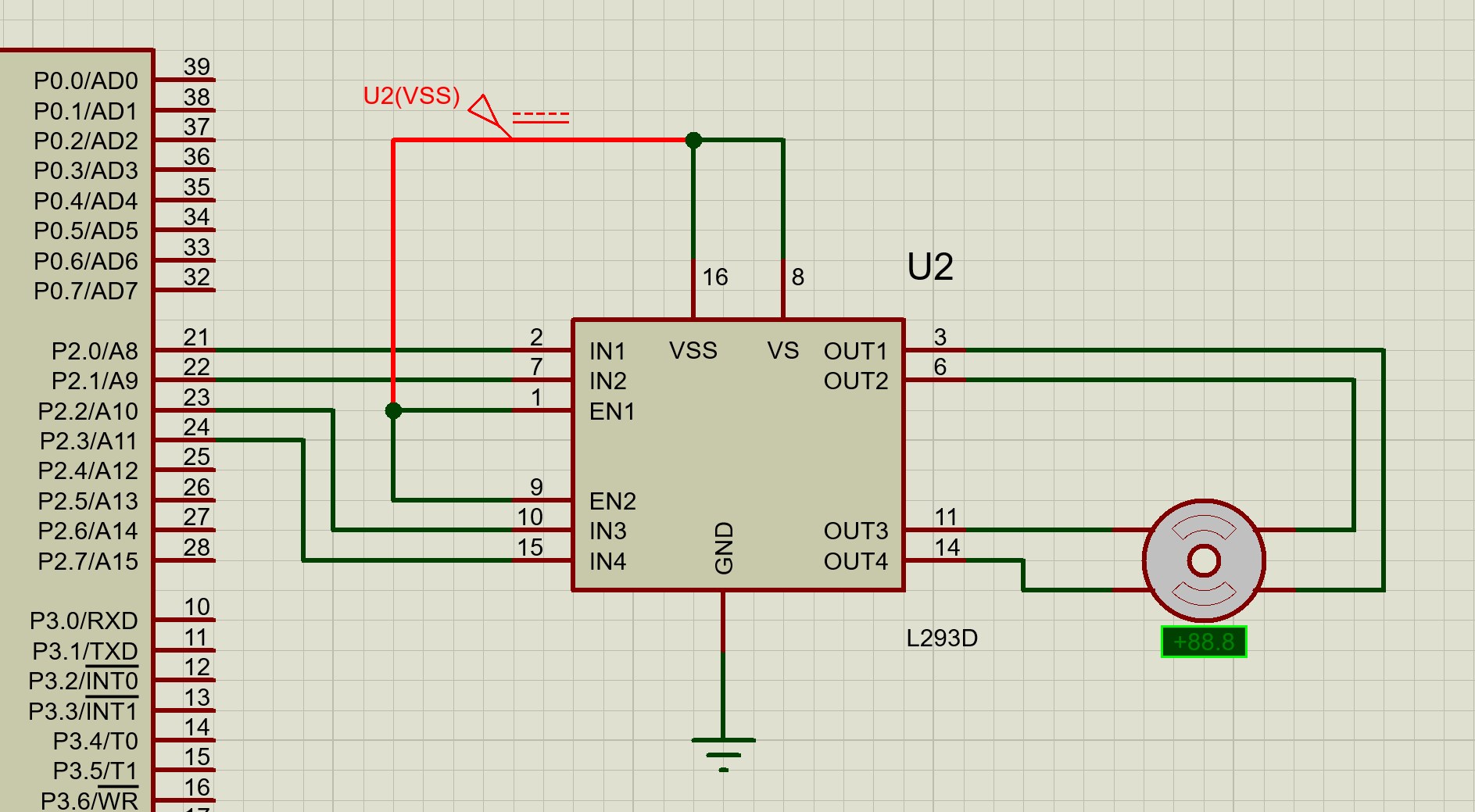


Рисунок 60 – Схема комплекса по управлению углом шагового двигателя

1. Разработать комплекс для мониторинга температуры (питание 5 Вольт) в помещении. Температуру отображать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимого диапазона температур и сохранять их в XML файле. В случае, если заданная температура в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если температура ниже нижнего порога, то синим. Обратить внимание на время установления. Предполагаемая схемотехника на рисунке 61. Перечень элементов: ADC\_08, PTC\_NICKEL.

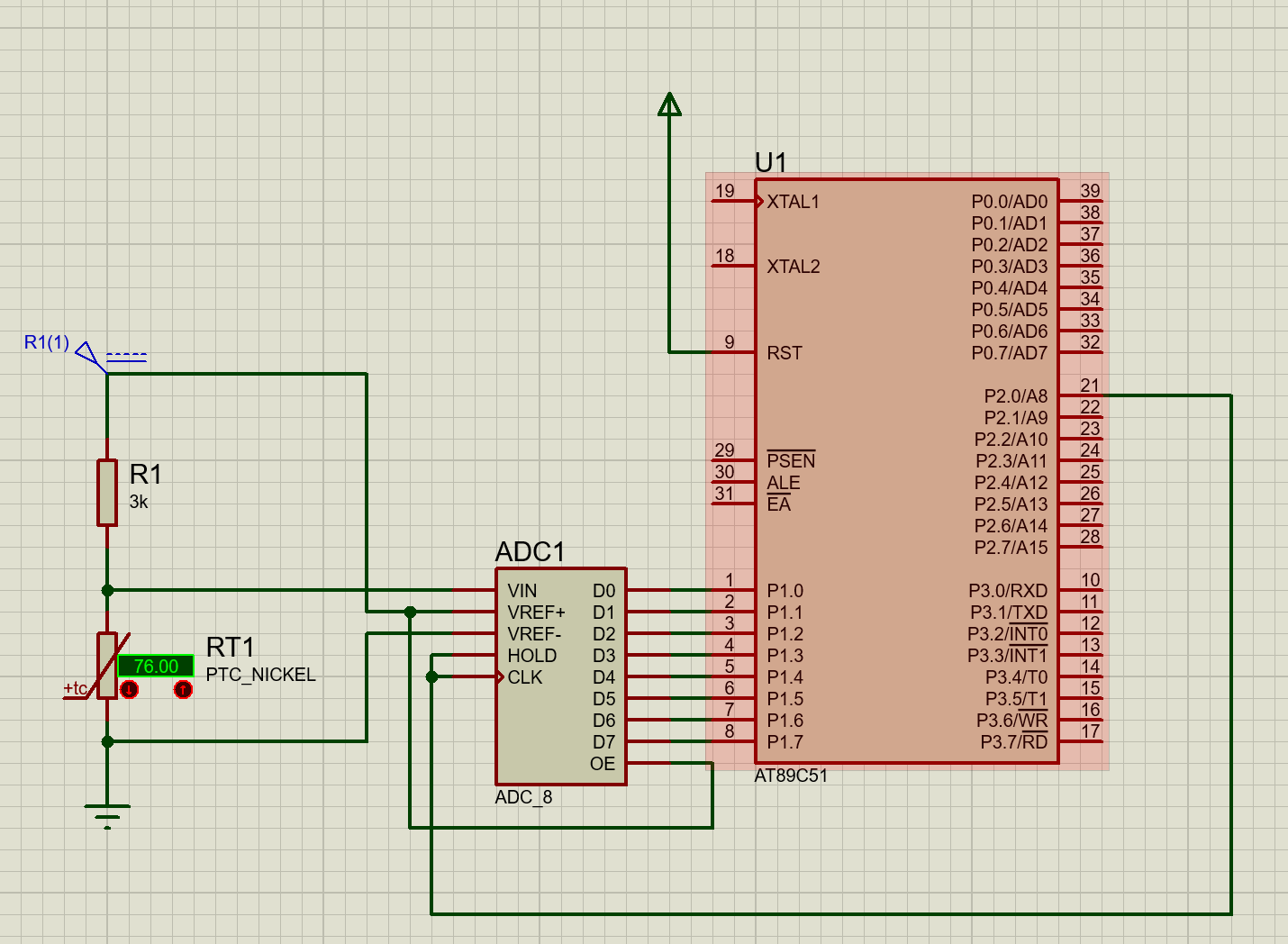


Рисунок 61 – Схема комплекса мониторинга темпеартуры

1. Разработать комплекс для мониторинга скорости ветра. Скорость отображать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимого диапазона скоростей и сохранять их в XML файле. В случае, если заданная скорость в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если скорость ниже нижнего порога, то синим. Предполагаемая схемотехника на рисунке 62. Перечень элементов: ANEMOMETER.

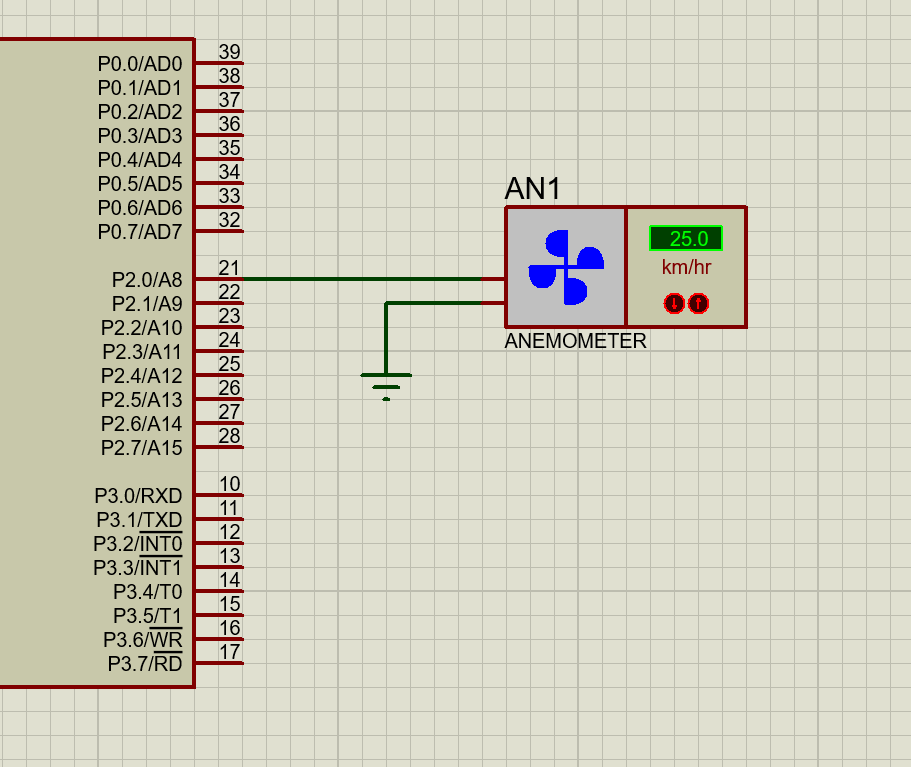


Рисунок 62 – Схема комплекса мониторинга скорости ветра

1. Разработать комплекс для мониторинга уровня освещенности (Питание 5 Вольт). Опрос датчика производить при помощи четырехразрядного АЦП. Освещенность отображать в люксах в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимого диапазона освещенности и сохранять их в XML файле. В случае, если заданная освещенность в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если освещенность ниже нижнего порога, то синим. Предполагаемая схемотехника на рисунке 63. Перечень элементов: ADC\_8, LDR, RES.

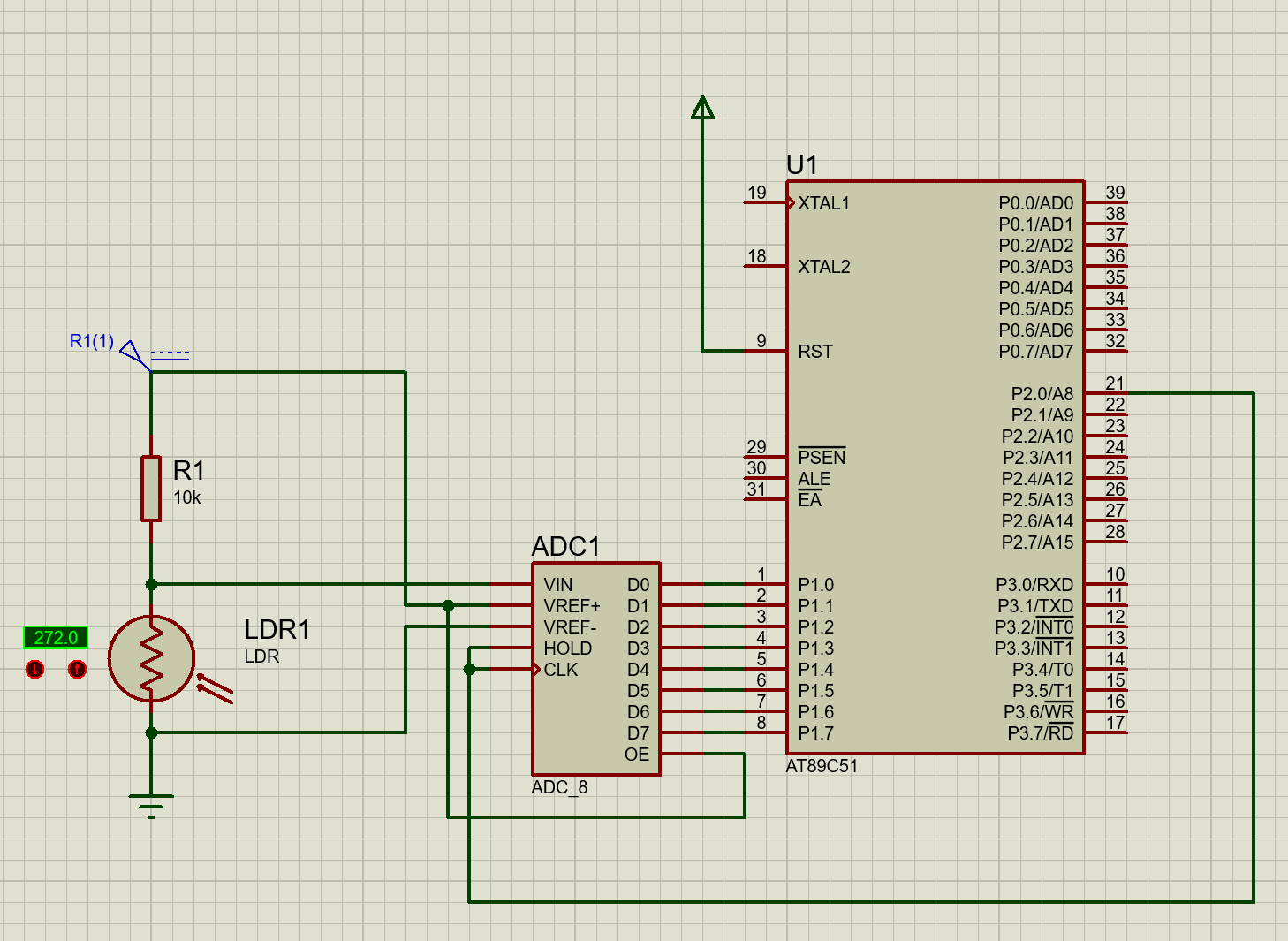


Рисунок 63 – Схема комплекса для мониторинга уровня освещенности

1. Разработать комплекс для игры в крестики-нолики для двух игроков. Подключить к микроконтроллеру 18 светодиодов для отображения нолика и крестика, а также семисегментный индикатор для отображения номера выбранной клетки. Для выбора номера клетки использовать нажатие на кнопку (инкрементировать значение на семисегментном индикаторе при нажатии, сбрасывать в 0 при достижении значения больше 8). Для подтверждения выбора использовать вторую кнопку. Для отображения порядка ходов использовать ещё один светодиод. Через приложение ПК должна быть возможность сделать ход. В окне приложения ПК должен отображаться ход текущей партии, а также итоговая расстановка трех последних партий. Приложение должно сохранять историю трёх последних партий в XML. Часть предполагаемой схемотехники (обратить внимание на особенность подключение диодов к порту P0) на рисунке 64. Перечень элементов: 7SEG-COM-ANODE, LED-BIGY, LED-BIRG, BUTTON, RES

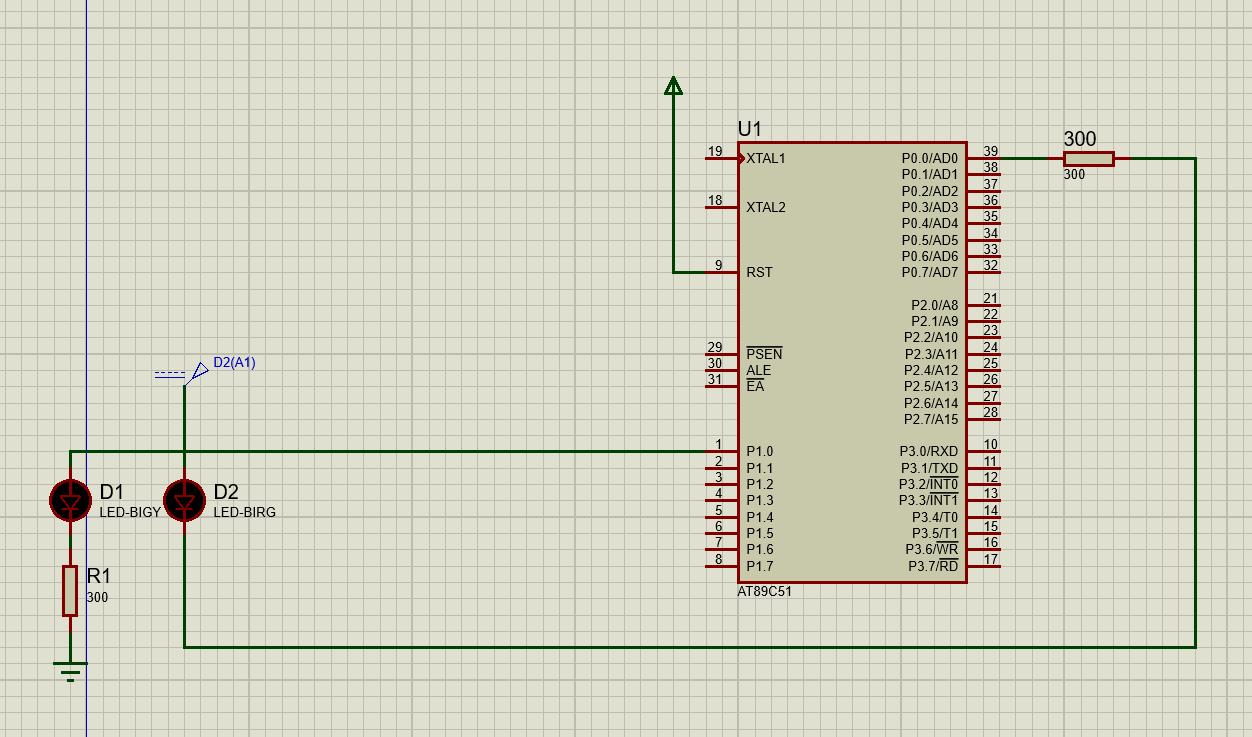


Рисунок 64 – Схема подключения светодиодов

1. Разработать комплекс для мониторинга температуры (питание 5 Вольт) в помещении. Среднюю температуру за минуту отображать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимого диапазона температур и сохранять их в XML файле. В случае, если заданная температура в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если температура ниже нижнего порога, то синим. Обратить внимание на время установления. Предполагаемая схемотехника на рисунке 65. Перечень элементов: ADC\_08, PTC\_NICKEL, RES.

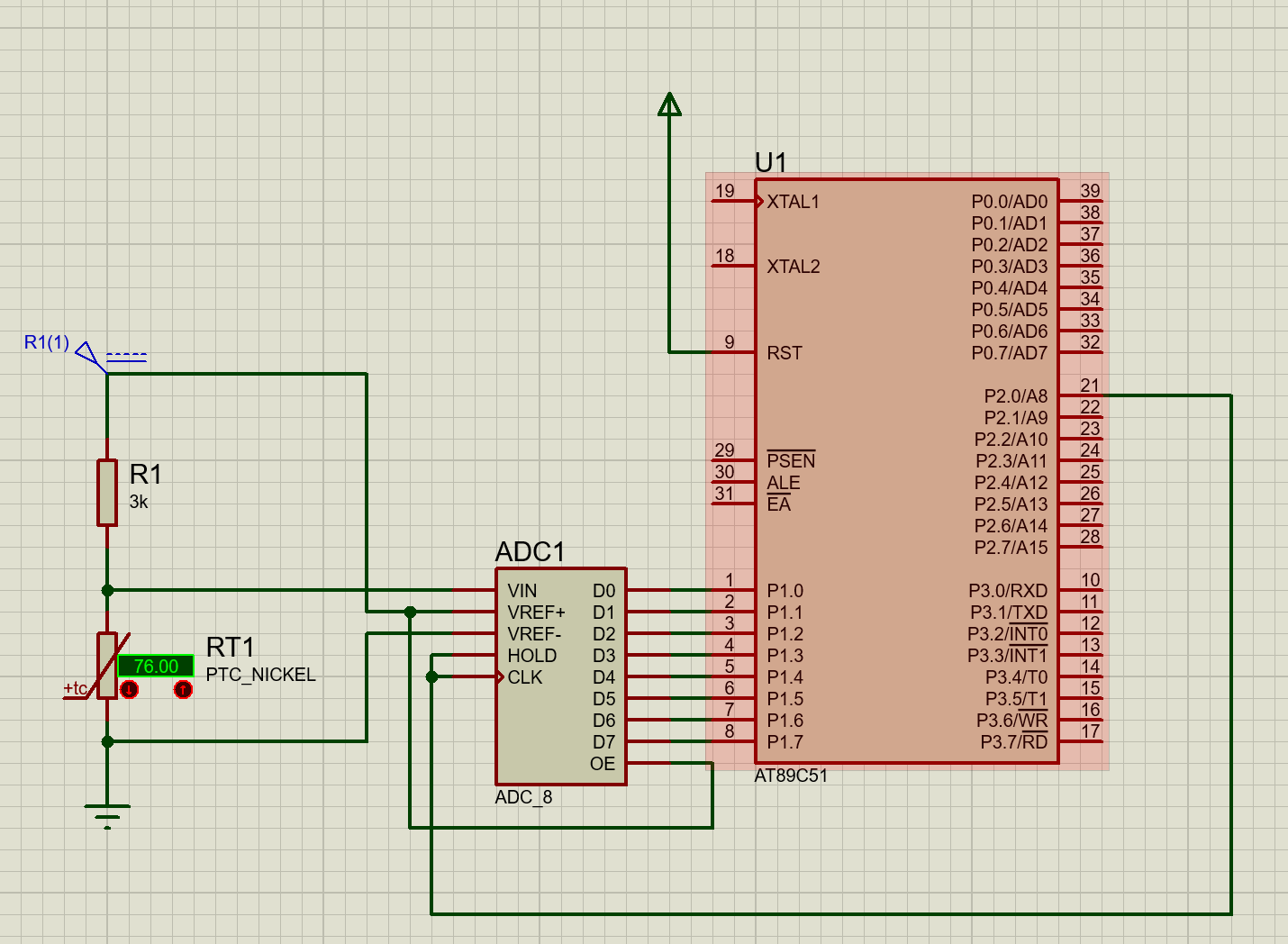


Рисунок 65 – Схема комплекса для мониторинга температуры

1. Разработать комплекс для мониторинга температуры (питание 5 Вольт) в помещении, открывающую окно в случае достижения критического значения (закрывать окно при понижении температуры). Использовать для мониторинга два датчика и вычислять среднее значение. Температуру отображать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и критический) допустимого диапазона температур и сохранять их в XML файле. Приложение ПК должно задавать критическое значение для микроконтроллера. В случае, если заданная температура в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если температура ниже нижнего порога, то синим. Обратить внимание на время установления. Для имитации окна использовать реле и светодиод. Предполагаемая схемотехника на рисунках 65, 66. Перечень элементов: ADC\_08, PTC\_NICKEL, RES, LED-BIGY, RLY-SPNO. Не забыть изменить номинальное сопротивление реле на 5 Вольт.

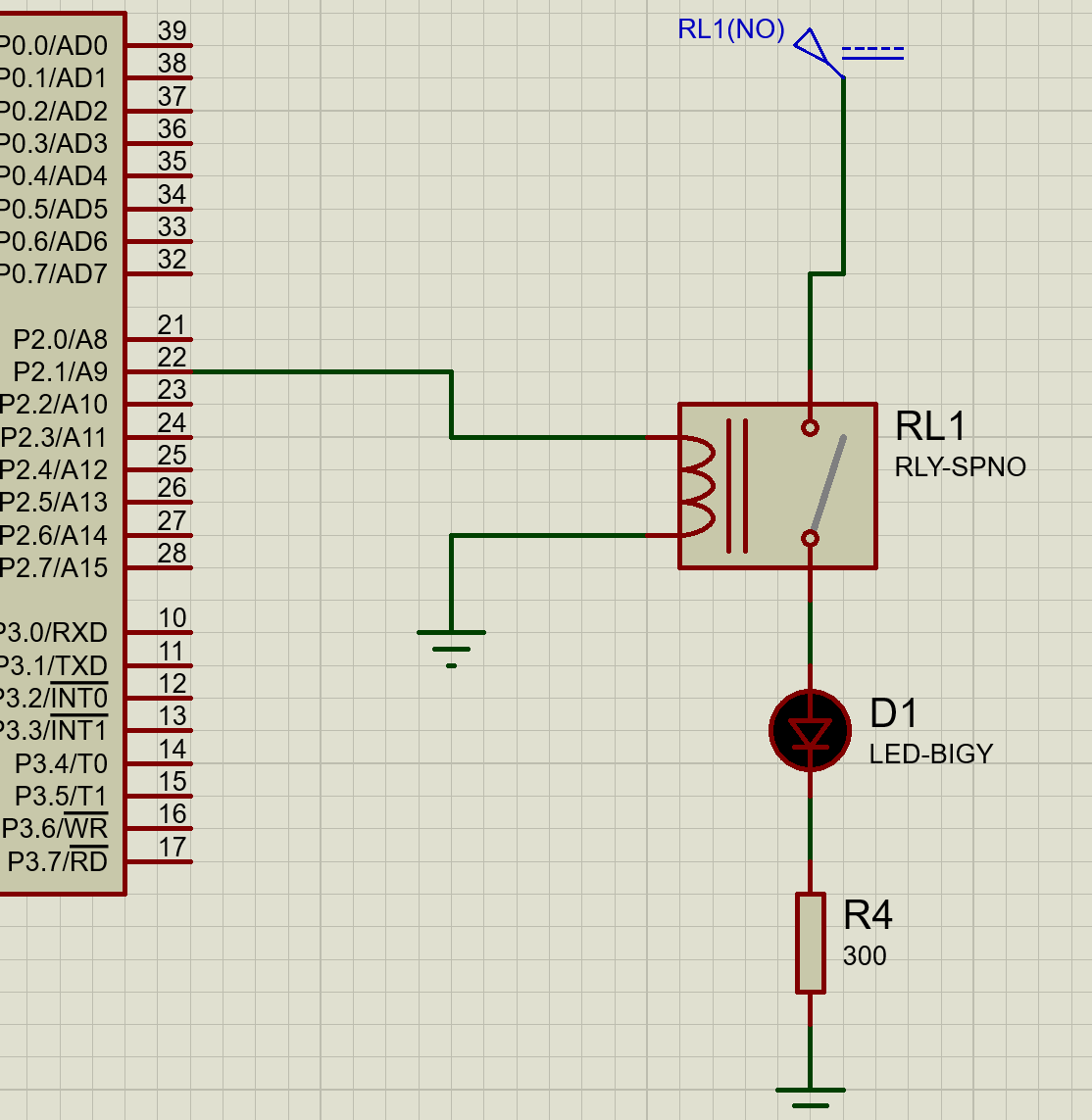


Рисунок 66 – Схема подключения реле

1. Разработать комплекс для автоматизации полива растений (питание 5 Вольт), включающий насос в случае достижения критического значения (выключать насос при повышении влажности). Влажность отображать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимого диапазона влажности и сохранять их в XML файле. Приложение ПК должно давать команду на включение насоса на ограниченный промежуток времени. В случае, если текущая влажность в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если влажность ниже нижнего порога, то синим. В качестве иллюстрации работы насоса использовать светодиод и реле. Предполагаемая схемотехника на рисунках 66, 67. Перечень элементов: HIH-5030, RES, ADC\_8, LED-BIGY, RLY-SPNO. Не забыть изменить номинальное сопротивление реле на 5 Вольт.

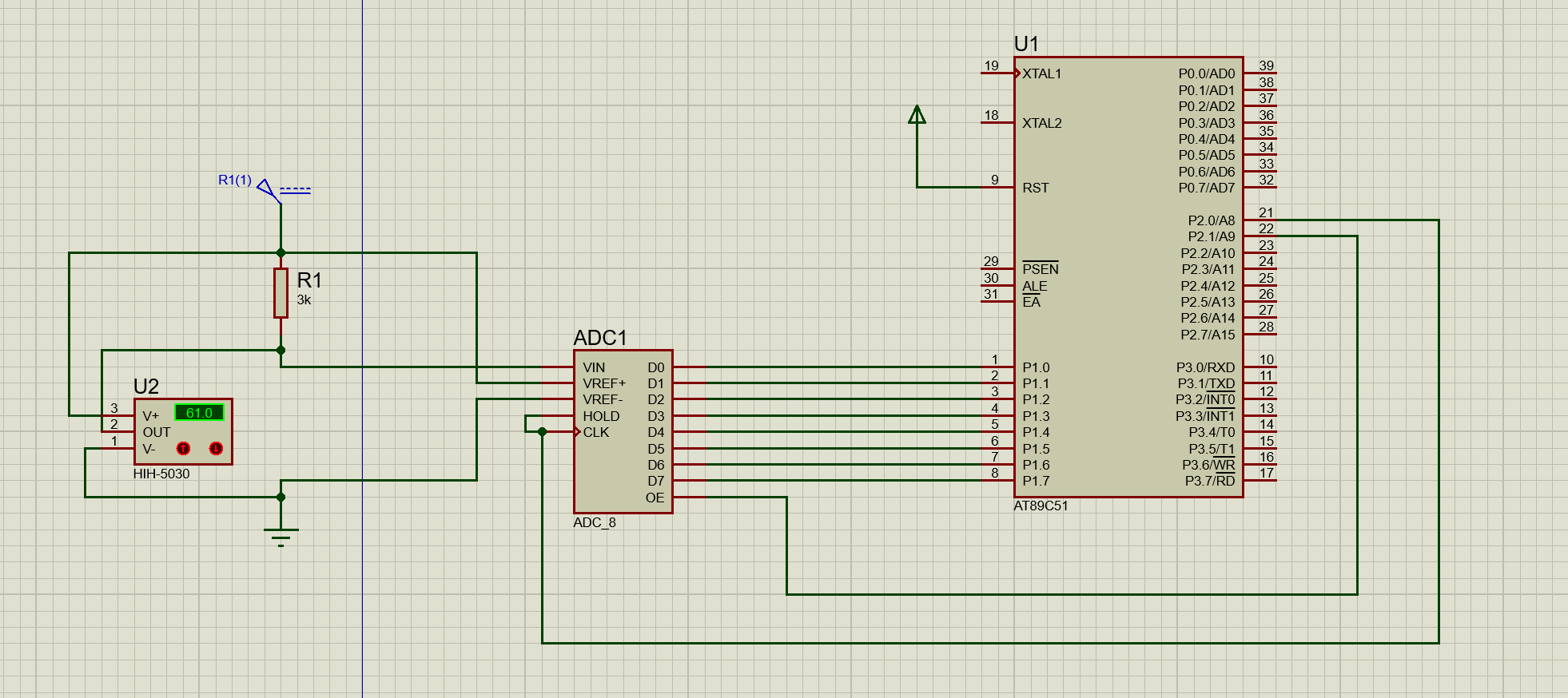


Рисунок 67 – Схема подключения датчика влажности

1. Разработать комплекс для оптимизации работы солнечной батареи. Батарея поворачивается в сторону с наибольшей освещенностью. Для мониторинга использовать два датчика. Освещенность обоих датчиков отображать в окне приложения ПК в виде двух чисел с плавающей точкой. Приложение должно позволять задавать пороги (нижний и критический) допустимого диапазона освещенности (выбирать максимум из двух датчиков) и сохранять их в XML файле. Приложение ПК должно задавать критическое значение для микроконтроллера. В случае, если максимальная освещенность в пределах нормы – отображать оба значения зеленым цветом, при превышении – красным, если ниже нижнего порога, то синим. Обратить внимание на время установления. Для управления поворотом использовать мотор. Предполагаемая схемотехника на рисунке 68. Перечень элементов: ADC\_8, LDR (Питание 5 Вольт), RES, L293D (Питание 12 Вольт), MOTOR-BISTEPPER.

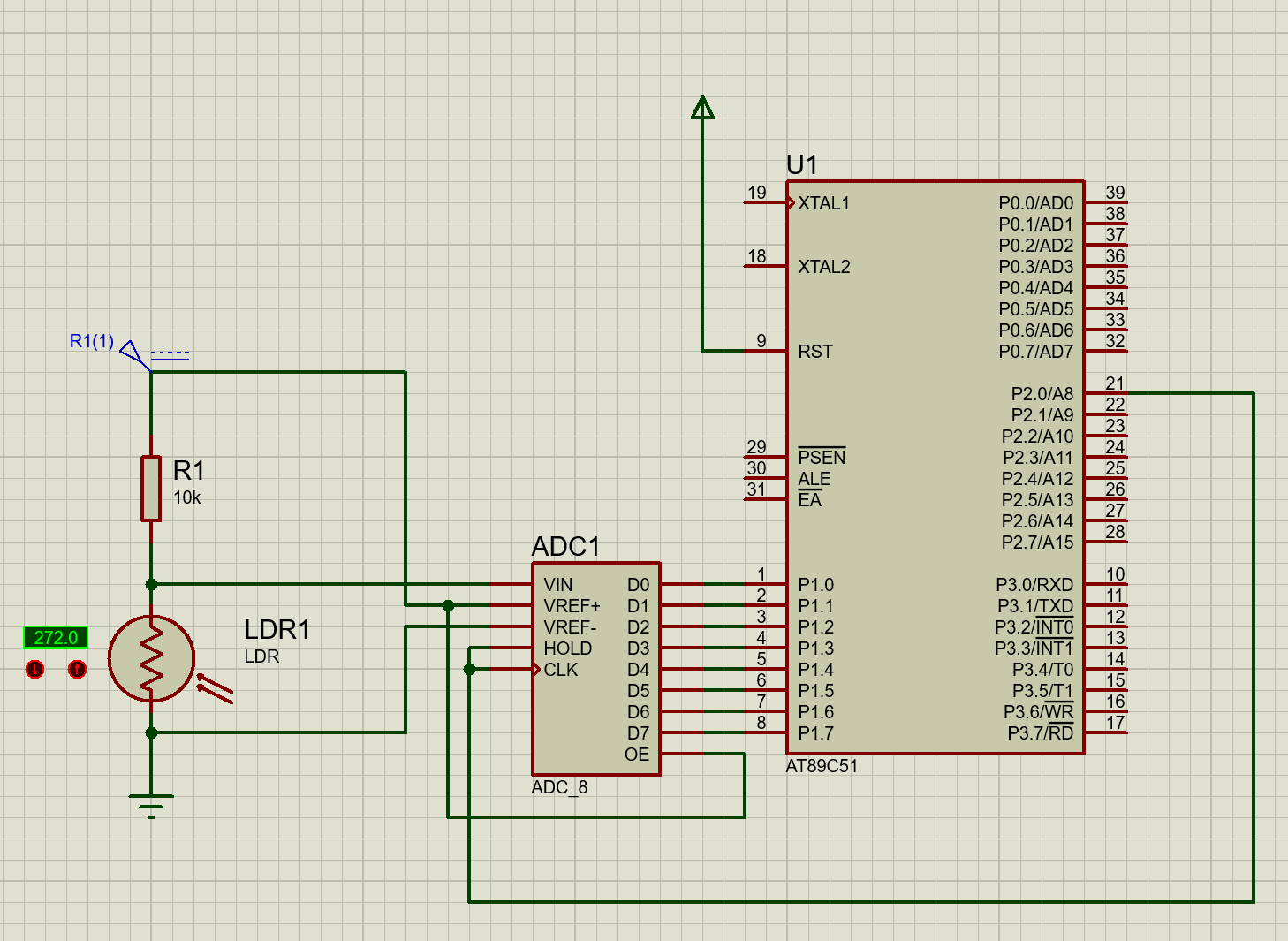


Рисунок 68 – Схема мониторинга уровня освещенности

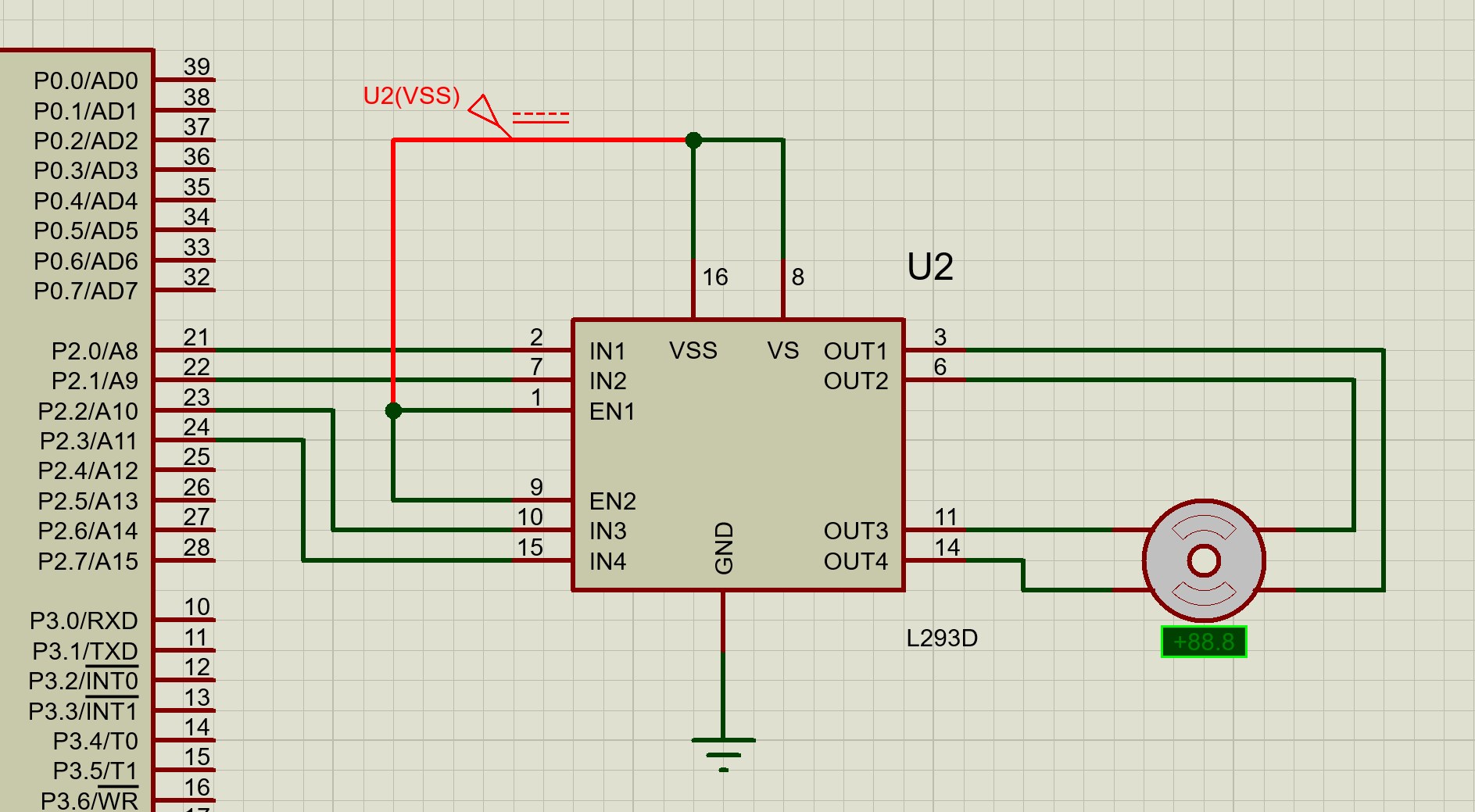


Рисунок 69 – Подключение двигателя

1. Разработать поисковый комплекс. В качестве датчика выступает мультивибратор (питание 7 Вольт). При достижении критической частоты зажигается светодиод. Текущую частоту счёта отображать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и критический) допустимого диапазона частот и сохранять их в XML файле. Приложение ПК должно задавать критическое значение для микроконтроллера. В случае, если частота в пределах нормы – отображать значение зеленым цветом, при превышении – красным, если ниже нижнего порога, то синим. Предполагаемая схемотехника на рисунке 70. Перечень элементов: BC337, POT, RES, CAP, LED-BIGY. Выход мультивибратора (точка, подключенная к осциллографу) подключается на ножке порта P1.

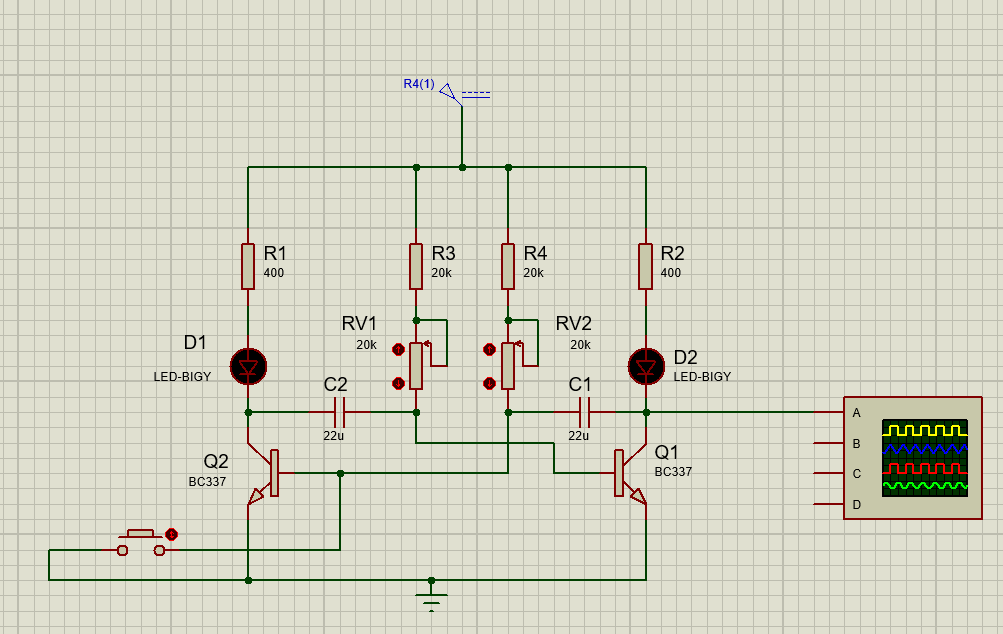


Рисунок 70 – Схема симметричного транзисторного мультивибратора

1. Разработать комплекс для мониторинга температуры (питание 5 Вольт) в помещении, при увеличении температуры увеличивается скорость вращения двигателя. Температуру отображать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и критический) допустимого диапазона температур и сохранять их в XML файле. Приложение ПК должно задавать текущее значение скорости вращения. В случае, если заданная температура в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если температура ниже нижнего порога, то синим. Обратить внимание на время установления. Предполагаемая схемотехника на рисунках 71, 72. Перечень элементов: PTC\_NICKEL, ADC\_8 (Питание 5 Вольт), BC337, MOTOR-DC (питание 12 Вольт), RES.

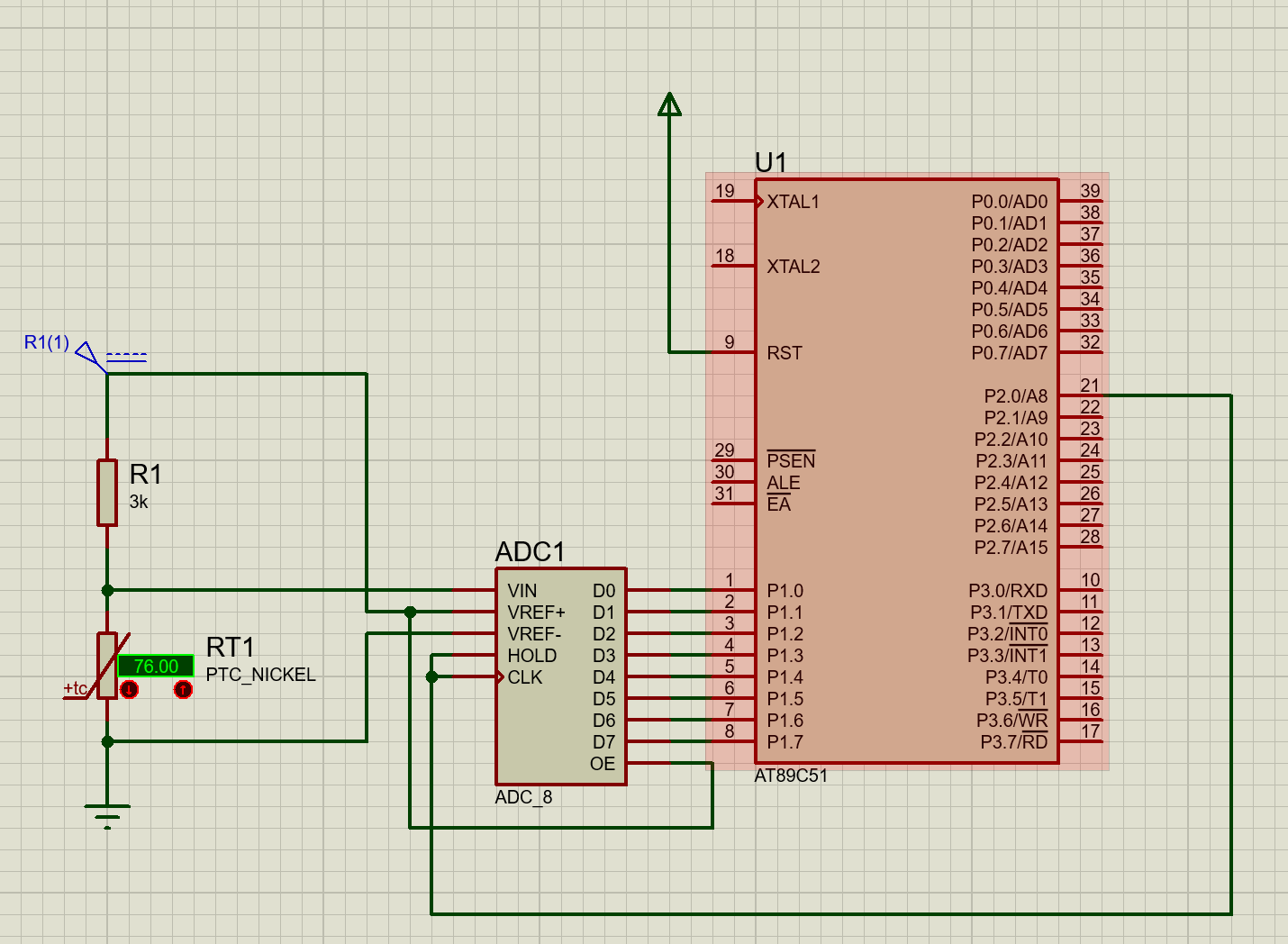


Рисунок 71 – Схема мониторинга температуры

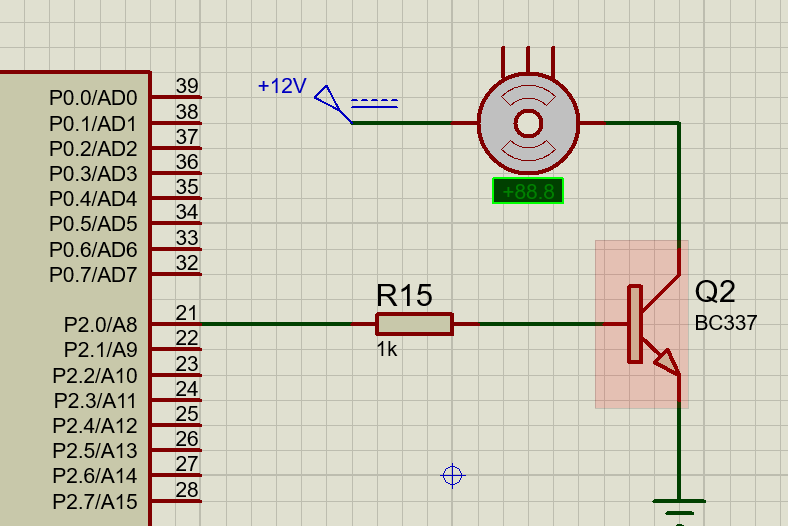


Рисунок 72 – Схема управления скоростью вращения двигателя

1. Разработать комплекс для автоматизации управления освещением, при уменьшении освещенности увеличивать яркость светодиода. Освещенность отображать в люксах в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и критический) допустимого диапазона освещенности и сохранять их в XML файле. Приложение ПК должно задавать текущее значение яркости в относительных единицах. В случае, если заданная освещённость в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если ниже нижнего порога, то синим. Обратить внимание на время установления. Предполагаемая схемотехника на рисунках Рисунок 73Рисунок 74. Перечень элементов: ADC\_8, LDR, DAC\_8, RES, LED-BIGY.

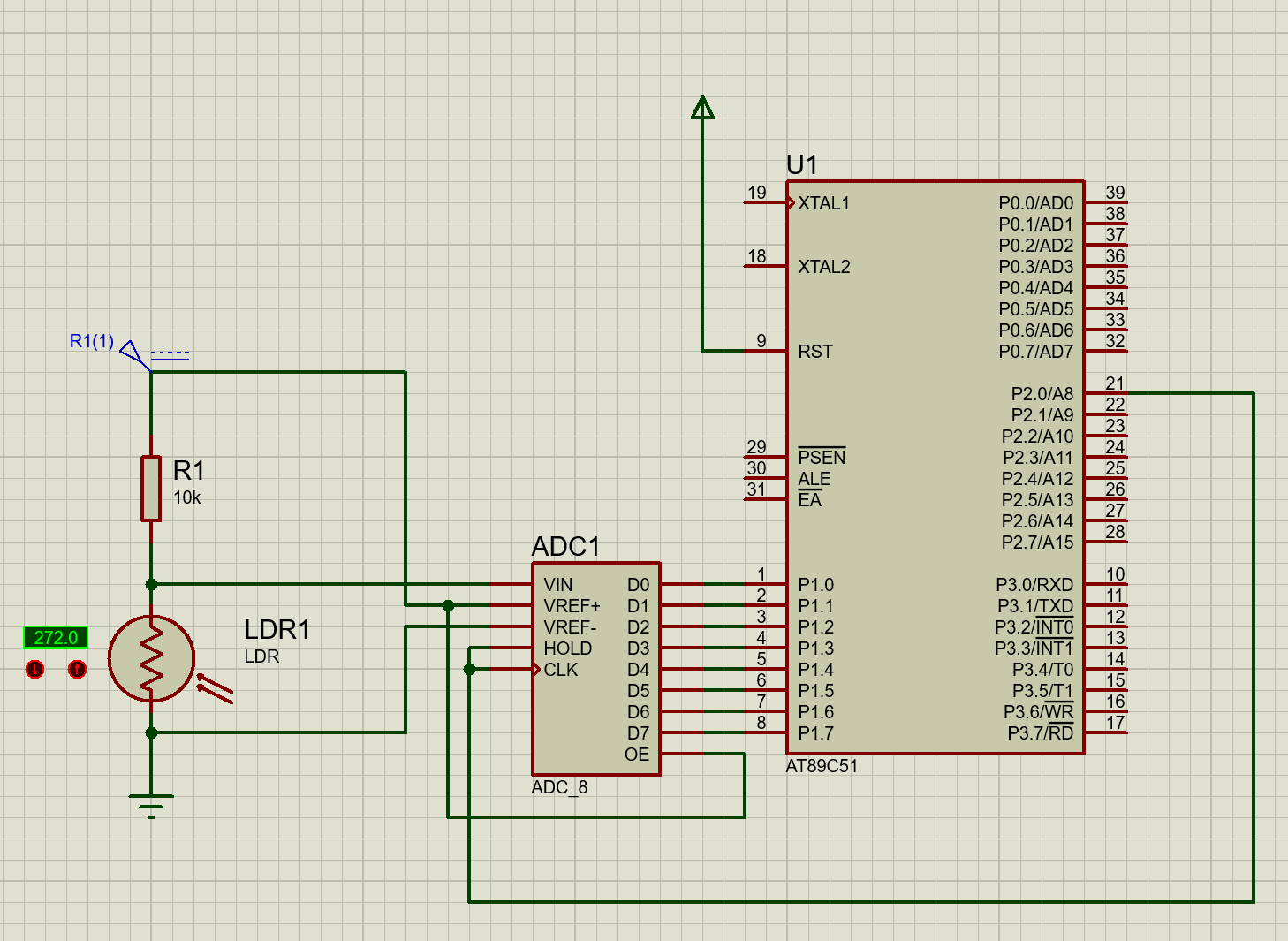


Рисунок 73 – Схема мониторинга уровня освещенности

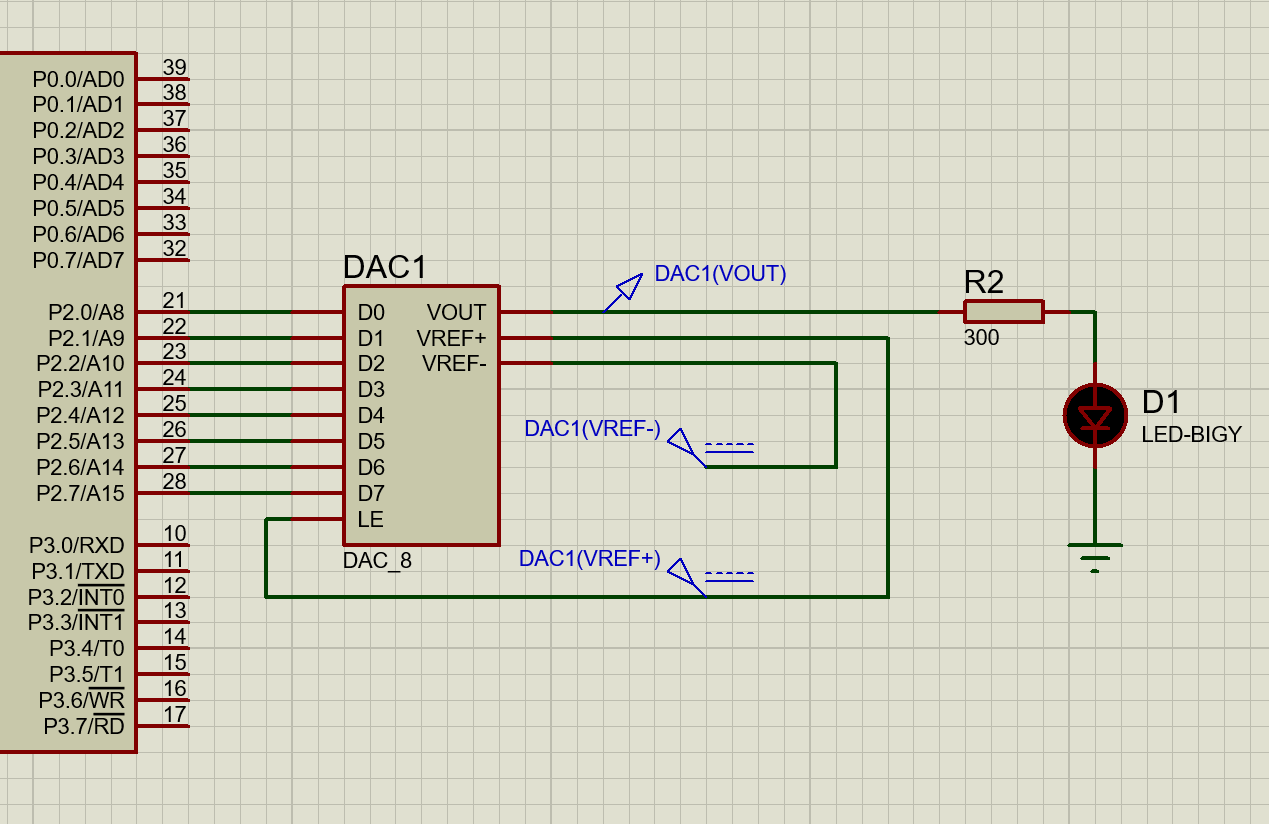


Рисунок 74 – Схема управления яркостью светодиода

1. Разработать систему мониторинга энергопотребления прибора. Токовое потребление отображать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и критический) допустимого диапазона потребления и сохранять их в XML файле. В случае, если заданное потребление в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если ниже нижнего порога, то синим. Предполагаемая схемотехника на рисунке 75. Перечень элементов: INA168, ADC\_8, POT, RES.

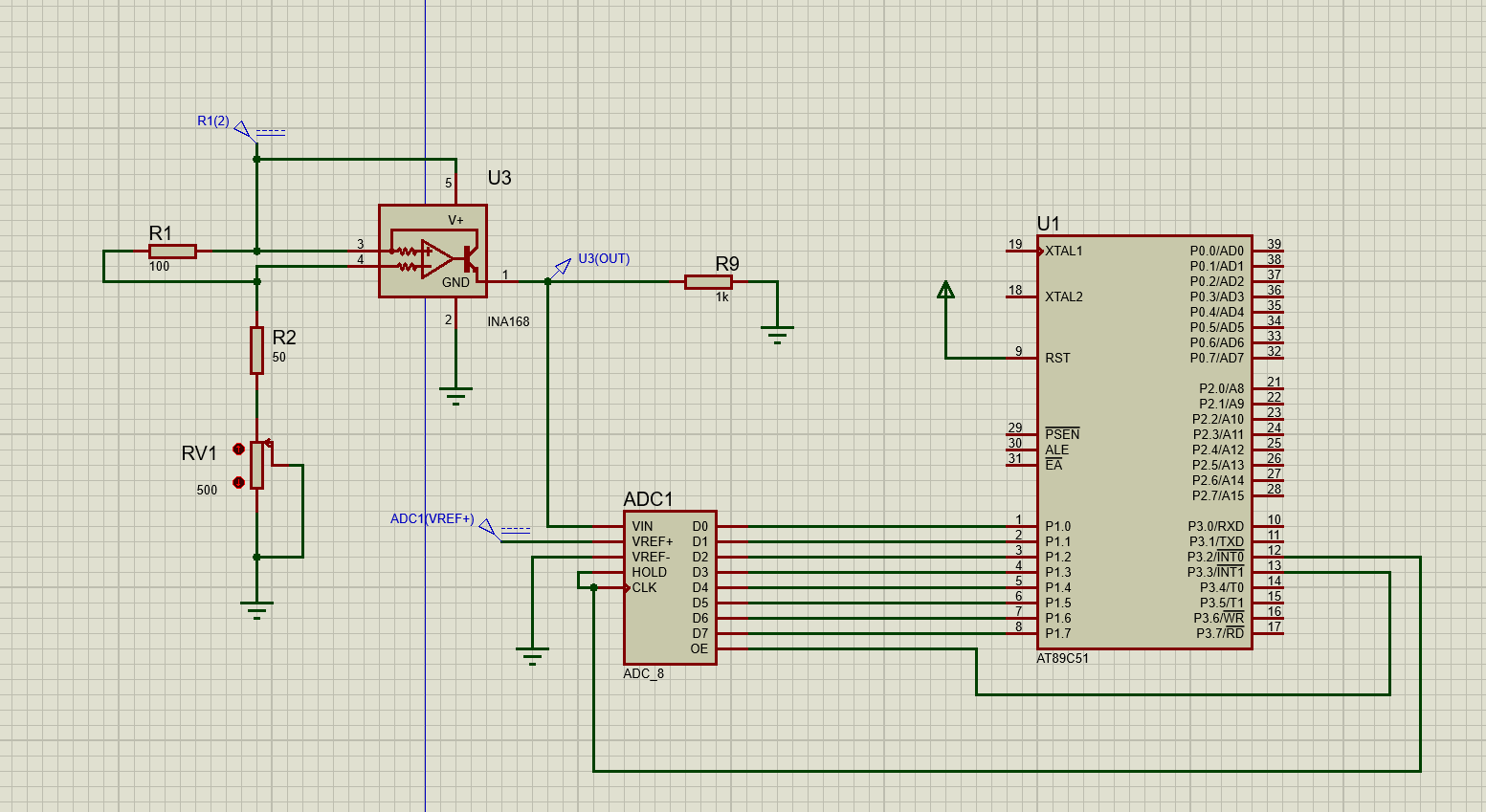


Рисунок 75 – Схема комплекса для мониторинга энергопотребления прибора

1. Разработать систему управления шторами (занавешивать в темное время). Освещенность отображать в люксах в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и критический) допустимого диапазона освещенности и сохранять их в XML файле. Приложение ПК должно принудительно закрывать и открывать шторы. В случае, если заданная освещённость в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если ниже нижнего порога, то синим. Предполагаемая схемотехника на рисунках 76, 77. Перечень элементов: LDR, ADC\_8, RES, RLY-SPNO, LED-BIGY.

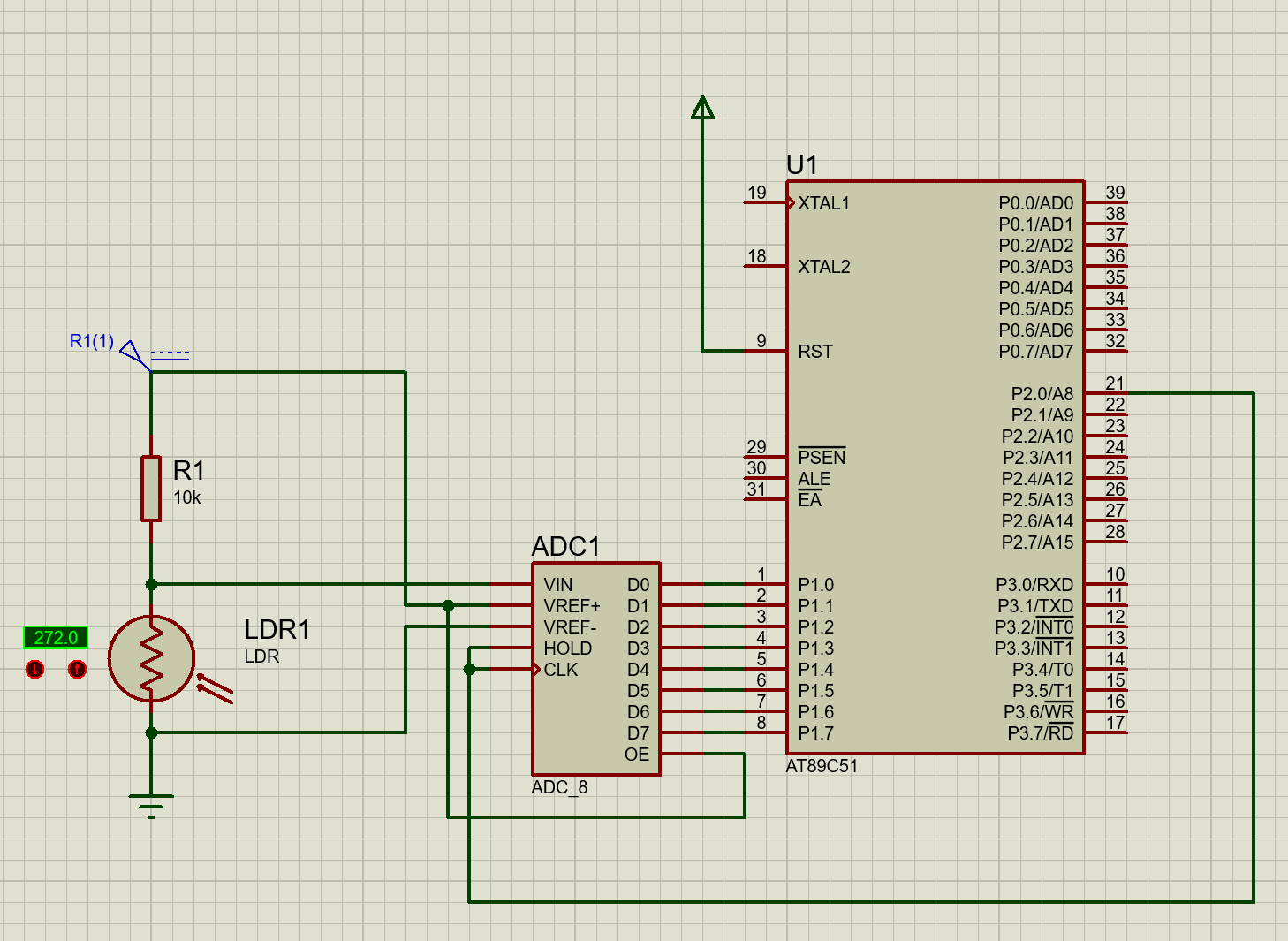


Рисунок 76 – Схема мониторинга уровня освещенности

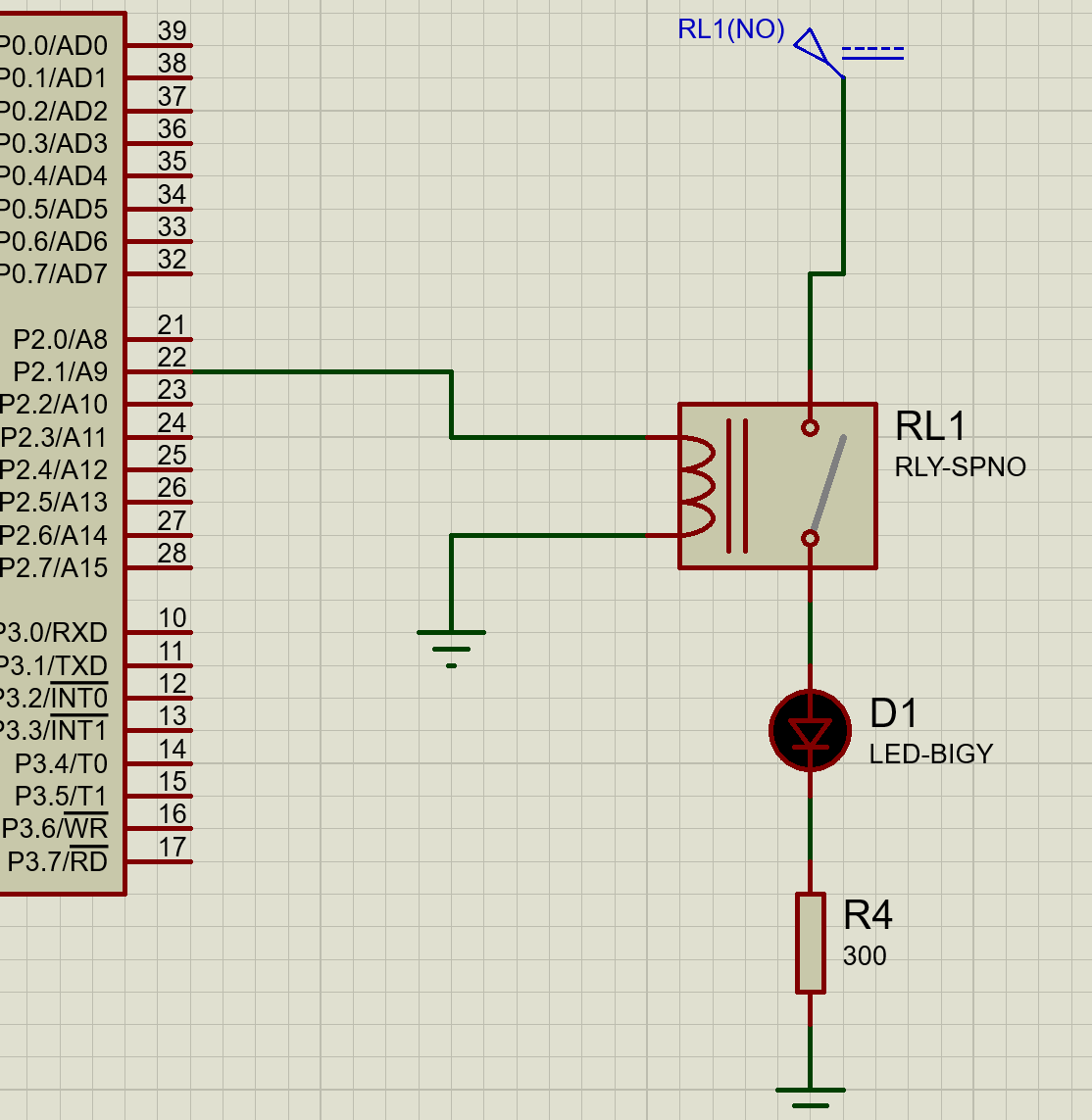


Рисунок 77 – Схема подключения реле

1. Разработать комплекс для контроля влажности (питание 5 Вольт), текущая влажность отображается на индикаторах (использовать динамическую индикацию). Влажность отображать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимого диапазона влажности и сохранять их в XML файле. В случае, если текущая влажность в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если влажность ниже нижнего порога, то синим. Предполагаемая схемотехника на рисунках 78, 79. Перечень элементов: HIH-5030, ADC\_8, RES, 7SEG-COM-ANODE, RLY-SPNO.

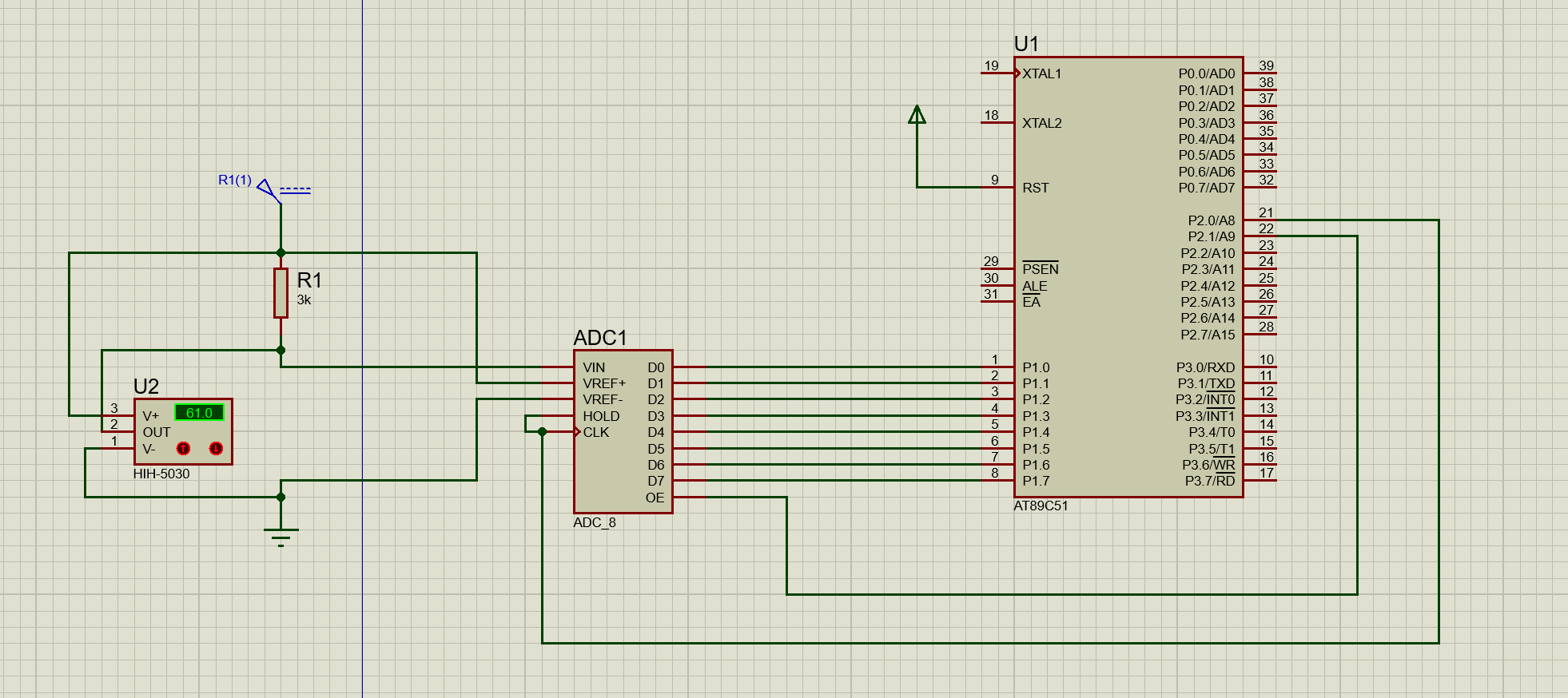


Рисунок 78 – Схема подключения датчика влажности

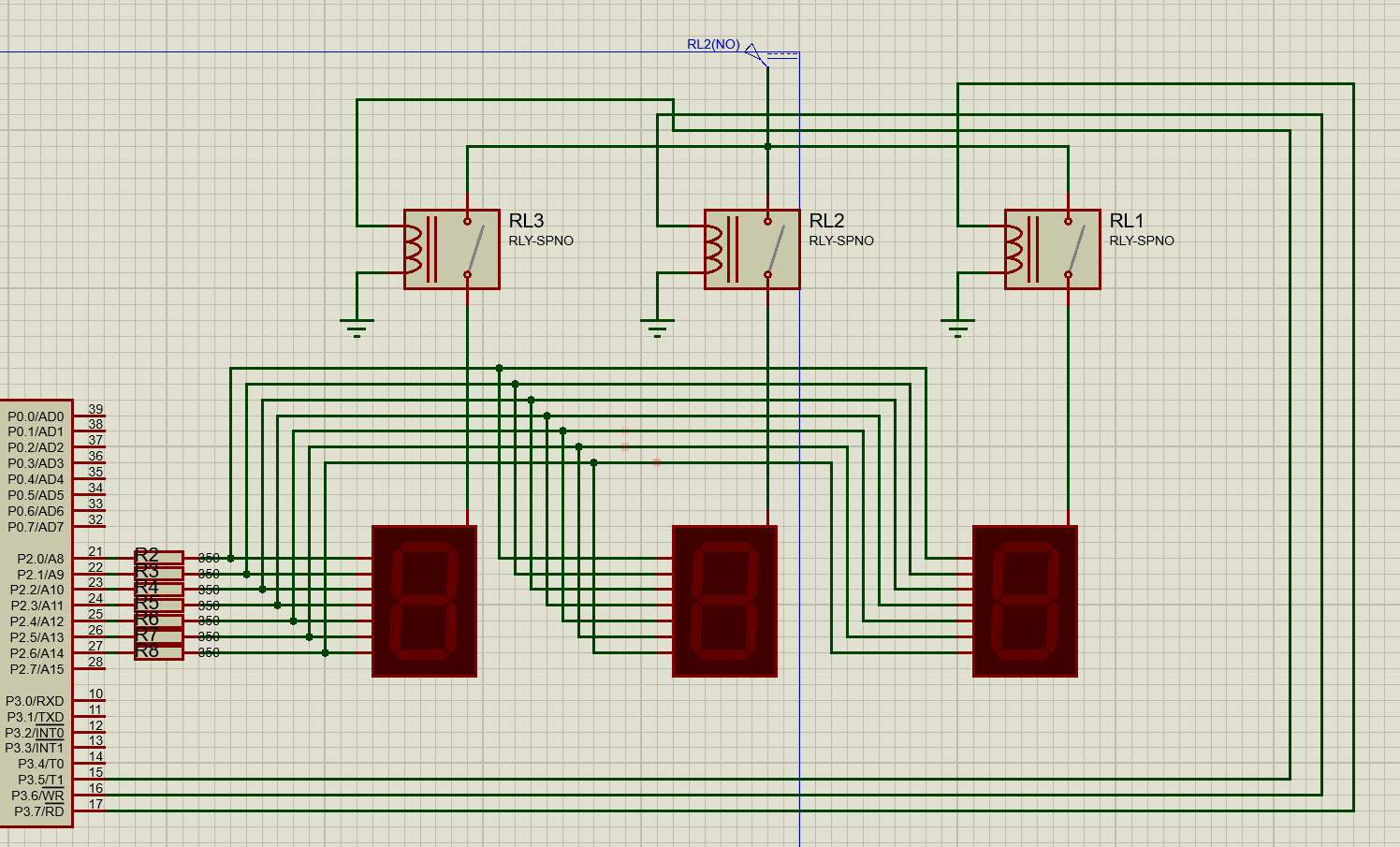


Рисунок 79 – Схема подключения 7-сегментных индикаторов