

|  |  |
| --- | --- |
|  |  |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Кибернетики  *(наименование факультета)* | | |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры при АО «ВЕГА»  *(наименование кафедры)* | | |

|  |  |
| --- | --- |
| **КУРСОВОЙ ПРОЕКТ (РАБОТА)** | |
| **по дисциплине** | |
| **«**Методы и стандарты программирования**»**  *(наименование дисциплины)* | |
| **Тема курсового проекта (работы) «**Симулятор электронных схем**»**  *(наименование темы)* | |
| Студент группы КМБО - 05 -19  *(учебная группа)* | *Минеев С. А.* |
| Руководитель курсового проекта (работы)  *должность, звание, ученая степень* | *Милонов Г. А.* |
| Рецензент (*при наличии*)  *должность, звание, ученая степень* | *-* |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2020 г. | *(подпись студента)* |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_2020 г. | *(подпись руководителя)* |

Москва 2020

Оглавление

[1. Введение 2](#_Toc60042835)

[2. Основная часть 3](#_Toc60042836)

[**2.1** **Структура проекта** 3](#_Toc60042837)

[**2.1.1 Шаблон MVC** 3](#_Toc60042838)

[**2.1.2 Предопределенные классы проекта.** 4](#_Toc60042839)

[**2.1.3 Принципы построения электронной схемы с помощью предопределенных классов.** 8](#_Toc60042840)

[**2.1.4 Принцип симуляции электрического тока в электронной схеме.** 8](#_Toc60042841)

[**2.2** **Руководство пользователя** 9](#_Toc60042842)

[**2.3 Требования для работы программы** 10](#_Toc60042843)

[3. Заключение 11](#_Toc60042844)

[4. Список используемой литературы 12](#_Toc60042845)

# Введение

Основной целью данной курсовой работы является написание программы «Симулятор электронных схем» на IDE Qt Creator 4.13.2 с использованием языка программирования С++.

Программа на прямую связана с темой «Симулятор электронных схем», которая позволяет увидеть пользователю увидеть то, как течет ток в электронных цепях, узнать из каких компонентов может состоять электронная схема и получить первичные навыки по построению электронных схем.

Данная тема является наиболее актуальной в современном мире. Все чаще обывателю приходится сталкиваться с электронной техникой и пробовать разобраться в том, как устроено то или иное электронное устройство, какие процессы происходят внутри электронной схемы, как взаимодействуют компоненты электронной схемы. Какими характеристиками обладает ток, магнитное поле и другие явления, связанные с электричеством. Выбранная тема охватывает многие из этих вопросов и позволяет углубить навыки в научной области информатика. Поэтому объектом исследования в работе является электронные схемы, а предметом исследования является устройство и работа электронных схем.

Дополнительными целями являются:

* Получение навыков по работе с IDE Qt Creator 4.13.2 С++
* Закрепление полученных навыков программирования на С++.
* Углубление в изучении С++.

Задачами данного проекта являются:

* Знакомство с IDE Qt Creator.
* Получение навыков работы с IDE Qt Creator.
* Изучение особенностей работы, отладки и компиляции проектов в IDE Qt Creator.
* Получение основных сведений по устройству электронных схем.
* Написание схемы программы структуры программы и программирование.
* Разработка программы под выбранную тему.

В этом проекте используются следующие методы исследования:

* Теоретические:
* Метод абстрагирование (изучение литературы.)
* Метод аналитического обобщения.
* Метод синтеза.
* Практические:
* Метод моделирования.

# Основная часть

## **Структура проекта**

### **2.1.1 Шаблон MVC**

#### **2.1.1.1 Основные сведения**

Данный проект полностью основывается на паттерне проектирования MVC (Model – View – Controller), который включает в себя:

* View (QGraphicsView)
* Model (QGraphicsScene)
* Controller (Event Propagation)

Так же имеются сами элементы, которые могут быть расположены на QGraphicsScene и наследуются от QGraphicsItem.

На рис.1 представлена связь между объектами MVC.

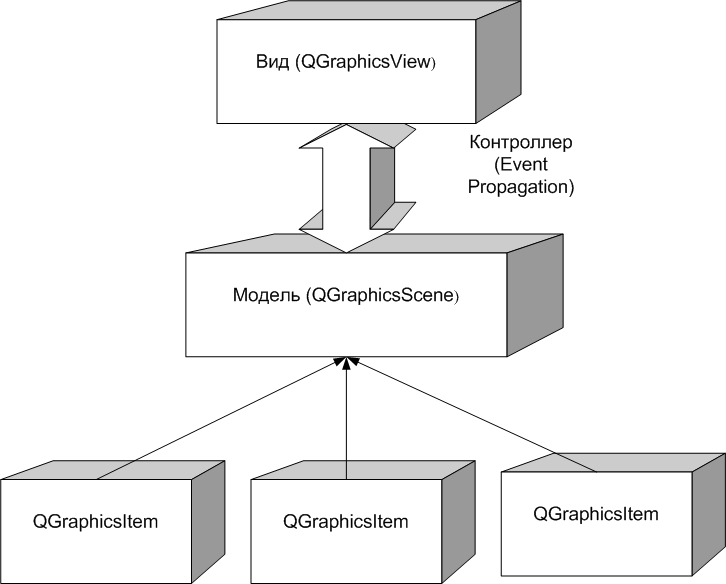


Рис. 1 паттерн MVC в Qt

#### **2.1.1.2 Описание элементов паттерна MVC.**

**QGraphicsView** - класс представления, он отвечает за отрисовку всех элементов.

**QGraphicsScene** - класс сцены, хранит в себе элементы.

**QGraphicsItem** - класс элемента, все элементы сцены должны быть потомками от этого класса. Сам класса **QGraphicsItem** является абстрактным, от него нужно создать собственный класс и переопределить в нем хотя бы методы **paint** и **boundingRect**.

#### **2.1.1.3 Описание взаимодействия элементов паттерна MVC на уровне самого паттерна.**

Пользователь приложения по средствам нажатия на кнопки с правой стороны от **MainWindows** вызывает соответствующий метод, реализованный в **MainWindows.** Метод создает выбранный объект и передает сигнал по стеку событий в **QGraphicsView. QGraphicsView** начинает отрисовку элемента на **QGraphicsScene** использую зарезервированную зону отрисовки, которая описывается методом **boundingRect**, расположенном в соответствующем **QGraphicsItem** и сам метод отрисовки **paint**, который также реализован в **QGraphicsItem**.

Соответственно в каждом **QGraphicsItem** реализованы следующие методы:

* **boundingRect;**
* **paint**;

### **2.1.2 Предопределенные классы проекта.**

#### **2.1.2.1 Класс** MainWindow**. Описание, свойства и методы.**

**Описание:** Класс **MainWindow** представляет собой основное окно программы с точки зрения графики и несет в себе методы, которые позволяют:

* Реагировать на события прокрутки колесика(wheelEvent).
* Внести сведения в окно программы и задать свойства программы (MainWindow - конструктор).

Именно класс **MainWindow** отвечает за создание экземпляров **QGraphicsItem** на **QGraphicsScene**. Поэтому, также в этом классе реализованы шесть слотов, которые отвечают за создание элементов электронной схемы(**QGraphicsItem**) и за запуск и остановку симуляции.

Кроме описанных выше методов, класс **MainWindow** в реализации содержит метод рандомного вывода чисел(randomBetween). Результат данного метода используется для рандомного расположения элементов на **QGraphicsScene**.

#### **2.1.2.2 Класс FormationConstructorView. Описание, свойства и методы.**

**Описание:** Данный класс представляет реализацию **QGraphicsView** и полностью от него наследуется. Кроме основных свойств и методов, предопределенных паттерном **MVC**, данный класс отвечает за:

* Изменение размеров окна(*resizeEvent*).
* Отслеживание положение курсора мышки в окне программы (в конструкторе включен соответствующий флаг).

Также в данном классе имеется две переменные:

**scene** – для хранения указателя на **QGraphicsScene.**

**DETECTORS** – реестр хранения спец. датчиков, о которых речь идет на стр. [].

#### **2.1.2.3 Класс FormationConstructorScene. Описание, свойства и методы.**

**Описание:** Данный класс представляет реализацию **QGraphicsScene** и полностью от него наследуется. Кроме основных свойств и методов, предопределенных паттерном **MVC**, в данном классе реализованы следующие методы:

**KeyPressEvent** - Метод улавнивания нажатия клавиш клавиатуры.

**MousePressEvent** - Метод улавливания нажатия кнопки мыши.

**MouseReleaseEvent** - Метод улавливания отпускания кнопки мыши.

**MouseMoveEvent** - Метод отслеживания Координат курсора мыши.

**DrawBackground** - Метод отрисовки сцены.

#### **2.1.2.4 Класс Detector. Описание, свойства и методы.**

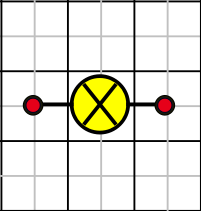


Рис. 2 Программное устройство элемента микросхемы.

Элемент электронной схемы – Лампочка-(Electric\_Lamp)

Датчик(Detector)

**Описание:** Рассмотримпрограммное устройство элемента микросхемы на примере осветительного элемента - лампочки. Данный **QGraphicsItem** состоит из слоя для отрисовки, границы которого описываются методом **boundingRect** и двумя датчиками**(Detectors),** которые позволяют**:**

* Соединять два элемента микросхемы между собой линиями**(Line\_Point).**

Поэтому в программе реализована следующая схема слоев взаимодействия элементов микросхемы:

Рис. 3 – Схема взаимодействия двух элементов микросхемы.

Элемент микросхемы

Элемент микросхемы

Датчики(Detectors)

Датчики(Detectors)

Линии (Line\_ point)

Из схемы видно, что элементы могут взаимодействовать только по средствам **Датчик → Линия → Датчик.** Поэтому датчик представляет собой отдельный класс со своими методами и свойствами.

Любой датчик хранит в своей классовой структуре:

Переменные описания к какой части элемента относится датчик (Левой/правой или верхней/нижней) – L\_V, R\_N.

#### **2.1.2.5 Классы с одним датчиком (Detector). Описание, свойства и методы.**

**Описание:** К классам с одним датчиком в предварительной реализации относятся:

* Ground(Земля).

Данный класс имеет следующие методы**:**

**MouseMoveEvent** - Метод отслеживания координат курсора мыши.

**DrawBackground** - Метод резервирования зоны повторной отрисовки.

**Point** - Метод отрисовки.

Стоит отметить, что **detector** элементов **Ground** является двухполюсным т. е. при соединении **Ground** с любым правым или левым от него элементом от принимает соответствующий знак направления тока для дальнейшей симуляции, которая расcматривается на стр. []

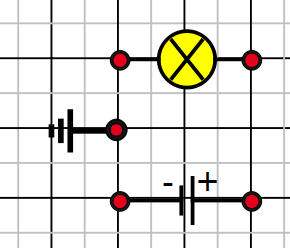


Рис. 4 – Распределение полюсов у элементов с одним датчиком (detectors).

**+**

**-**

**+**

**-**

+

**-**

Данные знаки определены в переменных SIGN объекта Detector.

Значение SIGN:

True (+), False (-).

#### **2.1.2.6 Классы с двумя датчиками (Detectors). Описание, свойства и методы.**

**Описание:** К классам с двумя датчиком в предварительной реализации относятся:

* Electric\_Lemp (Электрическая лампа).
* Battery (Батарейка).
* Selector (Переключатель).

Данные классы имеет следующие методы**:**

**MouseMoveEvent** - Метод отслеживания координат курсора мыши.

**DrawBackground** - Метод резервирования зоны повторной отрисовки.

**Point** - Метод отрисовки.

Стоит отметить, что **detectors данных** элементов является однополюсным т. е. при присоединении к **detectors** данных элементов линий от других правых или левых от них элементом **detectors** принимают соответствующие знаки направления тока для дальнейшей симуляции, которая расcматривается на стр. []

#### **2.1.2.7 Класс Line\_ point. Описание, свойства и методы.**

**Описание:** Экземпляры данного класса соединяет между собой компоненты электронной схемы по средствам **[**detector 1 элемента → Line\_Point → detector 2 элемента]. При этом:

* **Detector** 1 элемента, **Detector** 2 элемента – знают о том, какая **Line\_point** к ним подсоединена.
* **Line\_point** знает какие Detectors её удерживают.

### **2.1.3 Принципы построения электронной схемы с помощью предопределенных классов.**

**Описание:**

Чтобы построить электронную схему нужно:

1. C помощью кнопок, расположенных справа от поля размещения компонентов, создать элементы электронной схемы.
2. Разместить компоненты электронной схемы на поле размещения компонентов.
3. Соединить компоненты электронной схемы линией.

### **2.1.4 Принцип симуляции электрического тока в электронной схеме.**

**Описание:** Принцип симуляции основывается на том как реально течет ток (От + к -). Поэтому для того чтобы ток тек по схеме нужно:

1. В электронную схему встроить (подключить к электронным компонентам) источник питания т. е. **Battery**.
2. Не следует соединять **Battery** между собой одинаковыми полюсами.

**Принцип симуляции**:

Detectors **Battery** имеют **фиксированный уровень (Level = True), а detectors** других компонентов **нефиксированный (Level = False).**

При соединении компонентов. Алгоритм соединения компонентов Line\_point одновременно распределяет полюса таким образом, чтобы:

* Линия соединялась двумя датчиками(detectors) с разными знаками(полюсами).
* Изменять знак(SIGN) можно только у **Detectors**, которые нефиксированных т. е. у всех кроме **Battery**.

При этом стоит отметить после того, как нефиксированным **detectors** назначены знаки, они все равно *остаются* нефиксированными, чтобы в дальнейшем их можно было изменять, при изменении.

* Знак распределялся от фиксированного положительного **Detector**, к фиксированному отрицательному **Detector**.Так и обратно. От фиксированного отрицательного **Detector**, к фиксированному положительному **Detector.**

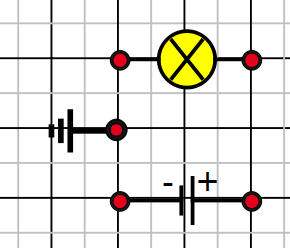


Рис. 4 – Распределение полюсов у элементов с одним датчиком (detectors).

**+**

**-**

**+**

**-**

+

**-**

## **Руководство пользователя**

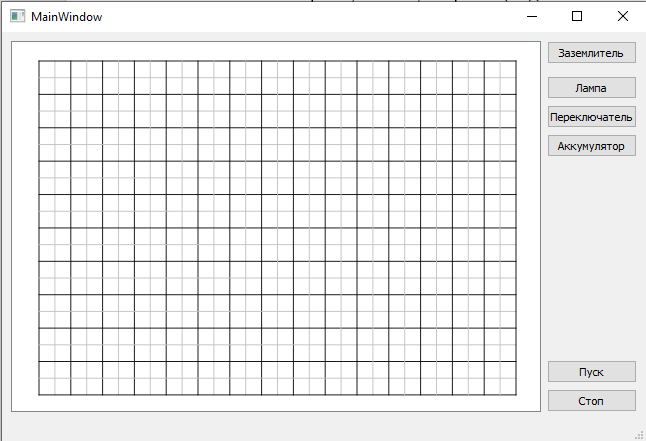


Рис. 5 – Внешний вид программы

Панель компонентов

Область размещение компонентов

* Чтобы построить электронную схему нужно на **панели компонентов** электронной схемы выбрать нужные вам компоненты. После нажатия на соответствующие кнопки компонентов, их экземпляры будут созданы в **области построения** электронной схемы.
* Разместите с помощью мышки компоненты электронной схемы в поле построения электронной схемы так, как вам нужно.
* Соедините компоненты между собой линией, нажимая на красные датчики соответствующих компонентов.

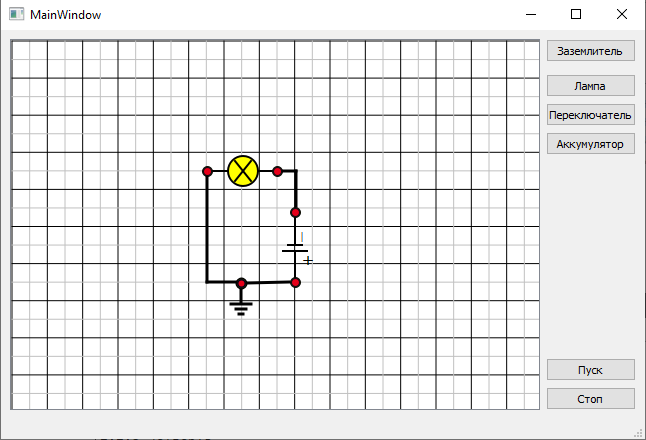


Рис. 6 – Пример электронной схемы.

Готово! Электронная схема построена.

## **2.3 Требования для работы программы**

Для того, чтобы программа работала корректно достаточно распаковать архив программы в нужное вам место и не отделять бинарный файл от dll библиотек программы.

* 1. **Требования для разработчиков**

IDE Qt Creator 4.13.2 С++ и более свежей версии.

# Заключение

В результате работы над проектом были получены следующие навыки:

- Навык по работе с IDE Qt Creator.

- Навыки самостоятельно решать сложные программные моменты с помощью языка с++.

- Отладка и компиляция проектов в IDE Qt Creator.

- Умение мыслить стратегически.

Закреплены:

- теоретические сведения по объектно-ориентированному программированию.

- Знания в области программирования на С++.

- Знания в области устройства и работы электронной схемы.

Стоит отметить, что проект может иметь дальнейшее развитие. В зависимости от фантазии и возможностей программиста. На данном этапе работы в программе можно:

Создавать электронные схемы с использованием батареек, лампочек, переключателей. т. е. согласно теме проекта строить электронные схемы.

В будущем планируется развитие проекта и включение в проект следующих возможностей:

* Корректной симуляции.
* Новых элементов электронной схемы.
* Возможность записи в файл электронных схем, созданных с помощью данной программы.
* Перевод в web – среду.

Результаты разработки и документацию по данному проекту можно найти на GitHub:

Ссылка на проект: https://github.com/MineevS/Simulator-v1.git

# Список используемой литературы

1. Объектно - ориентированное программирование 4-е издание. Автор: Р. Лафоре.
2. Род Стивенс “Алгоритмы”.
3. Шлее Макс - Профессиональное программирование на C++. +CD. Qt 4.8. 2012.
4. Импульсные источники питания, Элементная база, архитектура и ремонт, Коростелин А.В., 2020
5. АВТОЭЛЕКТРОНИКА. Зарядные и пуско-зарядные устройства №2” Ходасевич Т. И., Ходасевич А. Г.