

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ПЕТРА ВЕЛИКОГО»

Институт машиностроения, материалов и транспорта  
Высшая школа автоматизации и робототехники

**Курсовой проект**  
по дисциплине «Объектно-ориентированное программирование»  
на тему: **«Создание графического интерфейса с использованием  
фреймворка Qt»**

Выполнил студент  
гр. 3331506/20401

\_\_\_\_\_  
(подпись) Шарыгин А. В.

Работу принял

\_\_\_\_\_  
(подпись) Ананьевский М. С

Санкт-Петербург  
2025 г.

## Оглавление

Техническое задание.....	3
1. Введение.....	3
2. Теоретические сведения.....	3
2.1 Qt Creator.....	3
2.2 Виджеты.....	3
2.3 QML и Qt Quick.....	4
3. Ход работы.....	6
3.1 Создание проекта в Qt Creator.....	6
3.2 Редактирование файла CmakeLists.txt.....	7
3.3 Добавим класс sensors_handler.....	7
3.3.1 Конструктор Sensors_and_leds_handler(QObject *parent).....	7
3.3.2 Деструктор ~Sensors_and_leds_handler().....	7
3.3.3 Геттеры класса Sensors_and_leds_handler.....	7
3.3.4 Метод startReading. Настройка COM-порта.....	7
3.3.5 Метод sendCommand.....	7
3.3.6 Метод readData.....	7
3.4 Файл main.cpp.....	7
3.5 Файл main.qml.....	7
4. Результат работы.....	8
5. Заключение.....	10
6. Литература.....	11

## **Техническое задание**

Необходимо создать графический интерфейс для „умной“ теплицы. С помощью графического интерфейса должно осуществляться взаимодействие с теплицей: чтение данных с датчиков, управление приборами.

### **1. Введение**

Графический интерфейс — важная часть проекта. С его помощью осуществляется удобное взаимодействие с программой. Пользователю намного удобнее управлять объектом через графический интерфейс, вместо того, чтобы разбираться в большом количестве строк кода. Графический интерфейс содержит основные данные, с которыми чаще всего взаимодействует пользователь.

### **2. Теоретические сведения**

В данном проекте для создания графического интерфейса взят фреймворк Qt. Qt — фреймворк для разработки кроссплатформенного программного обеспечения [1, 2, 4] на языке программирования C++ [6, 8]. Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путём простой компиляции программы для каждой системы без изменения исходного кода. Включает в себя все основные классы которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью базами данных и XML. Является полностью объектно-ориентированным, расширяемым и поддерживающим технику компонентного программирования.

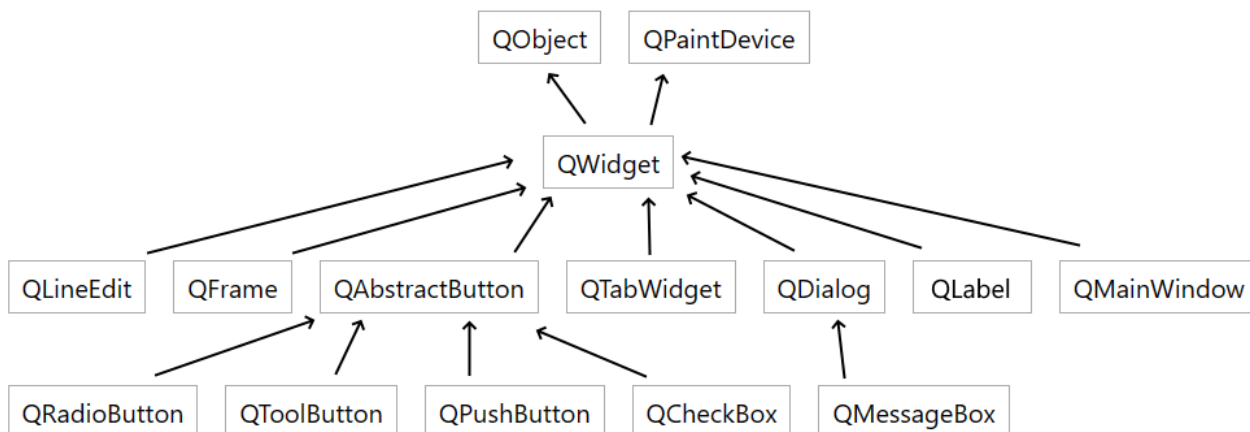
#### **2.1 Qt Creator**

Среда разработки Qt Creator [3] не является неотъемлемым компонентом для разработки приложений с помощью фреймворка Qt, тем не менее он упрощает многие вещи, прежде всего конфигурацию и настройку построения приложения. Кроме того, Qt Creator предоставляет унифицированный интерфейс для основных операционных систем.

#### **2.2 Виджеты**

Исторически первый подход к построению графического интерфейса на Qt представляли виджет. Виджеты представляют различные элементы пользовательского интерфейса, например, кнопки, текстовые поля и прочие компоненты, из которых состоит окно приложения. Виджет позволяет обрабатывать различные пользовательские события, например, события мыши и клавиатуры. И таким образом пользователь может взаимодействовать с приложением. Базовый встроенный набор виджетов Qt расположен в модуле QtWidgets.

При этом Qt широко использует концепцию наследования. Все виджеты наследуются от встроенного типа QWidget. Это базовый виджет и базовый класс всех виджетов пользовательского интерфейса. Он содержит большинство свойств, необходимых для описания виджета, а также такие свойства позиционирования виджета, цвет и т. д. Иерархию виджетов Qt еще можно представить следующим образом [1]:



## 2.3 QML и Qt Quick

Изначально использование виджетов из модуля Qt Widgets представляло основной подход к созданию графических приложений на Qt. Однако впоследствии появился второй подход, в котором используется специальный язык Qt Modeling Language [4] (или сокращенно QML). QML представляет декларативный язык описания пользовательского интерфейса. Он появился вместе с развитием мобильных устройств с сенсорными экранами и позволяет создавать гибкие пользовательские интерфейсы с минимальным написанием кода. Основа функциональности языка QML сосредоточена в одноименном модуле Qt QML, который определяет и реализует язык и его инфраструктуру, а также предоставляет интерфейсы API для интеграции языка QML с JavaScript и C++. Дополнительно модули Qt Quick и Qt Quick Controls предоставляют множество визуальных элементов, анимацию и других компонентов, которые применяются в связке с QML. Таким образом, вместо использования виджетов Qt для проектирования пользовательского интерфейса также можно использовать QML и Qt Quick.

Базовым типом для всех визуальных элементов в Qt Quick является тип Item, который предоставляет общий набор свойств. Фактически он представляет собой прозрачный визуальный элемент, который можно использовать в качестве контейнера. Все остальные визуальные элементы в Qt Quick наследуются от

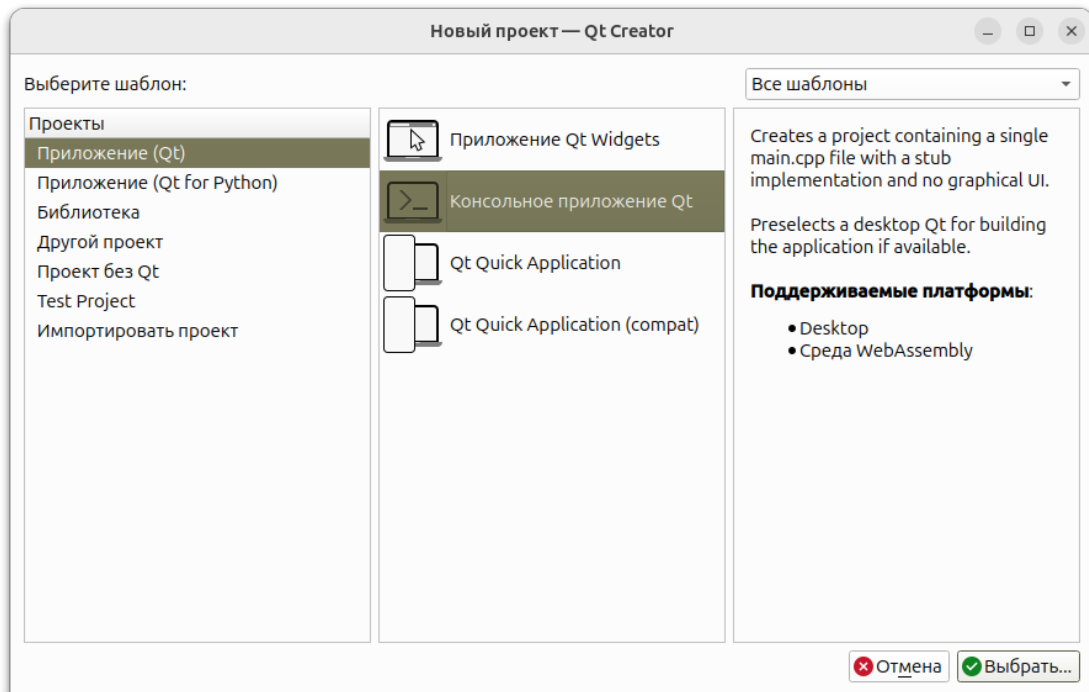
Item. Для создания графического интерфейса в QML модуль Qt Quick Controls предоставляет набор встроенных компонентов, из которых отмечу основные из них:

- `ApplicationWindow`: представляет окно верхнего уровня с заголовком и футером
- `BusyIndicator`: индикатор загрузки
- `Button`: кнопка
- `CheckBox`: флажок, который может быть в отмеченном или неотмеченном состоянии
- `ComboBox`: кнопка со всплывающим окном
- `Dial`: компонент в виде кругового набора (как на старых стационарных телефонах)
- `Dialog`: диалоговое окно
- `Label`: метка с текстом
- `Popup`: всплывающее окно
- `ProgressBar`: индикатор прогресса операции
- `RadioButton`: радиокнопка или переключатель
- `ScrollBar`: вертикальные и горизонтальные полосы прокрутки
- `ScrollView`: визуальный компонент, который поддерживает прокрутку
- `Slider`: слайдер для выбора числового значения из некоторого диапазона
- `SpinBox`: выпадающий список
- `Switch`: кнопка-переключатель
- `TextArea`: элемент для ввода многострочного текста
- `TextField`: элемент для ввода однострочного текста
- `ToolTip`: всплывающая подсказка
- `Tumbler`: прокручиваемый список элементов для выбора

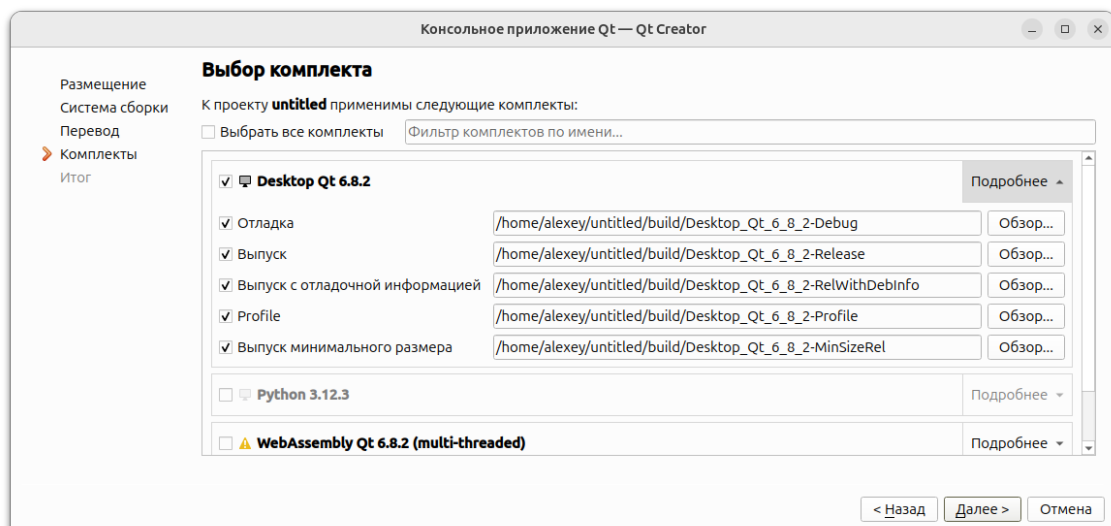
## 3. Ход работы

### 3.1 Создание проекта в Qt Creator

- 1) Создадим новый проект в Qt Creator [7], выберем **консольное приложение Qt**:



- 2) Далее назовем проект **smart\_farm\_app** и выберем размещение.
- 3) Выберем систему сборки CMake.



- 4) На шаге **Выбор комплекта** сделаем так:

## **3.2 Редактирование файла CmakeLists.txt**

На этом шаге редактируется файл сборки [5]. В нем прописываем минимально используемую версию Cmake, используемые библиотеки, подключаем классы и файлы.

## **3.3 Добавим класс sensors\_handler**

### **3.3.1 Конструктор Sensors\_and\_leds\_handler(QObject \*parent)**

В данном конструкторе происходит инициализация последовательного порта и переменных для хранения данных. Также происходит соединение метода readyRead со слотом readData

### **3.3.2 Деструктор ~Sensors\_and\_leds\_handler()**

В данном деструкторе происходит прекращение работы последовательного порта.

### **3.3.3 Геттеры класса Sensors\_and\_leds\_handler**

Геттеры нужны для обращения к данным.

### **3.3.4 Метод startReading. Настройка COM-порта**

В этом методе настраиваем COM-порт. Задаем имя, скорость передачи, битность. Взаимодействие с Arduino происходит через COM-порт. Настройки порта на стороне Arduino должны быть такими же, это необходимо для правильной передачи данных.

### **3.3.5 Метод sendCommand**

Этот метод нужен для отправки команд Arduino по COM-порту.

### **3.3.6 Метод readData**

Данный метод нужен для чтения данных из COM-порта. Полученная строка разбивается на значения: время, температура, влажность воздуха и влажность почвы.

## **3.4 Файл main.cpp**

В файле main.cpp регистрируем класс Sensors\_and\_leds\_handler, чтобы к нему можно было обращаться из кода Qml. Указываем ссылку на файл main.qml.

### 3.5 Файл main.qml

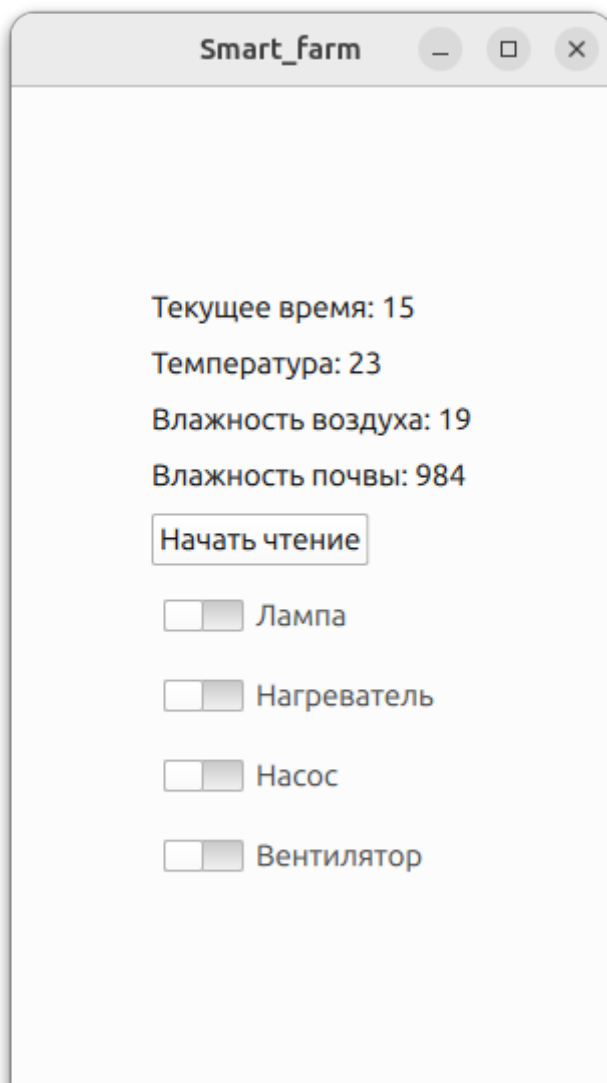
В файле main.qml пишем код для графической части.

- 1) Создаем окно `ApplicationWindow`, задаем его размеры и название.
- 2) Создаем экземпляр класса `Sensors_and_leds_handler` для работы с датчиками. Прописываем в нем обработку сигнала об изменении значений с датчиков.
- 3) Создаем текстовые поля, в которых будут отображаться значения с датчиков, и переключатели для управления приборами в теплице.
- 4) В переключателях прописываем отправку сигнала (команды) при изменении состояния переключателя.



## 4. Результат работы

В результате работы появляется окно **Smart\_farm**, в котором отображаются: время, данные о климате и переключатели приборов. Любой из приборов можно включить или выключить.



## **5. Заключение**

В ходе выполнения курсового проекта были изучены возможности фреймворка Qt. Было создано приложение для удобного взаимодействия с „умной“ теплицей.

## 6. Литература

1. Metanit: Разработка на Qt [Электронный ресурс]. – URL: <https://metanit.com/cpp/qt/> (дата обращения: 12.05.2025).
2. Qt Official Website [Электронный ресурс]. – URL: <https://www.qt.io/> (дата обращения: 12.05.2025).
3. Qt Creator Documentation [Электронный ресурс]. – URL: <https://doc.qt.io/qtcreator/> (дата обращения: 12.05.2025).
4. Jurgen Bocklage-Ryannel, Cyrill Lorquet, Johan Thelin. Qt 6 QML. – 728 с.
5. Крейг Скотт. Профессиональный CMake: практическое руководство. – 2018. – 439.
6. Бьерн Страуструп. Язык программирования C++. – Москва: Бином, 2006. – 1104 с. – ISBN 5-7989-0223-4.
7. Саймон Монк. Програмируем Arduino. – Санкт-Петербург: Питер, 2017. – 272 с. – ISBN 978-5-496-02385-7.
8. Скотт Мейерс. Эффективный и современный C++: 42 рекомендации по использованию C++11 и C++14. – Москва: Вильямс, 2016. – 304 с. – ISBN 978-5-8459-2052-1.