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 _____ Шарыгин А. В.
(подпись)

Работу принял _____ Ананьевский М. С
(подпись)

Санкт-Петербург
2025 г.

Оглавление

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 на языке программирования C++ (Jurgen Bocklage-Ryannel, Cyrill Lorquet, Johan Thelin Qt 6 QML). 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

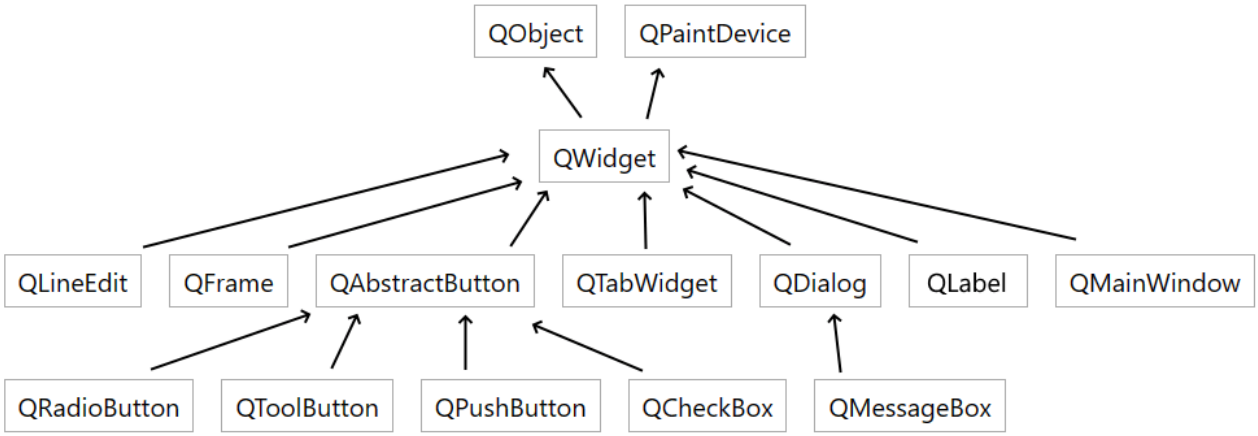
Среда разработки Qt Creator не является неотъемлимым компонентом для разработки приложений с помощью фреймворка Qt, тем не менее он упрощает многие вещи, прежде всего конфигурацию и настройку построения приложения. Кроме того, Qt Creator предоставляет унифицированный интерфейс для основных операционных 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 (рисунок с сайта https://metanit.com/cpp/qt/):



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 (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.

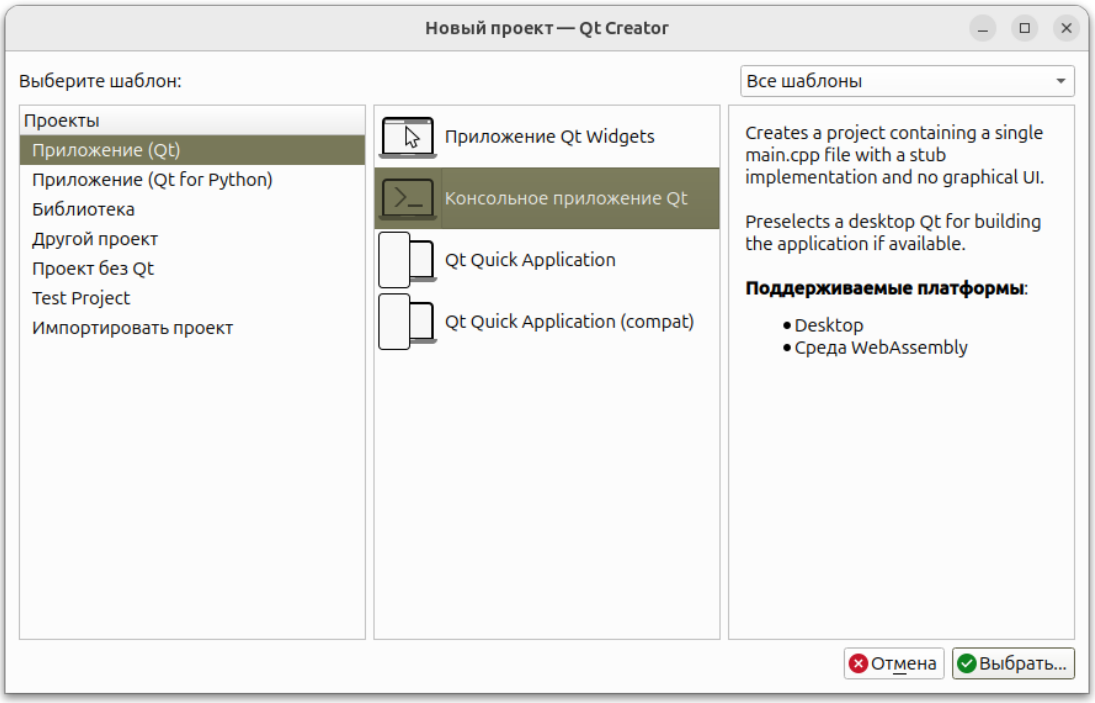
Базовым типом для всех визуальных элементов в Qt Quick является тип Item, который предоставляет общий набор свойств. Фактически он представляет собой прозрачный визуальный элемент, который можно использовать в качестве контейнера. Все остальные визуальные элементы в Qt Quick наследуются от Item. Для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

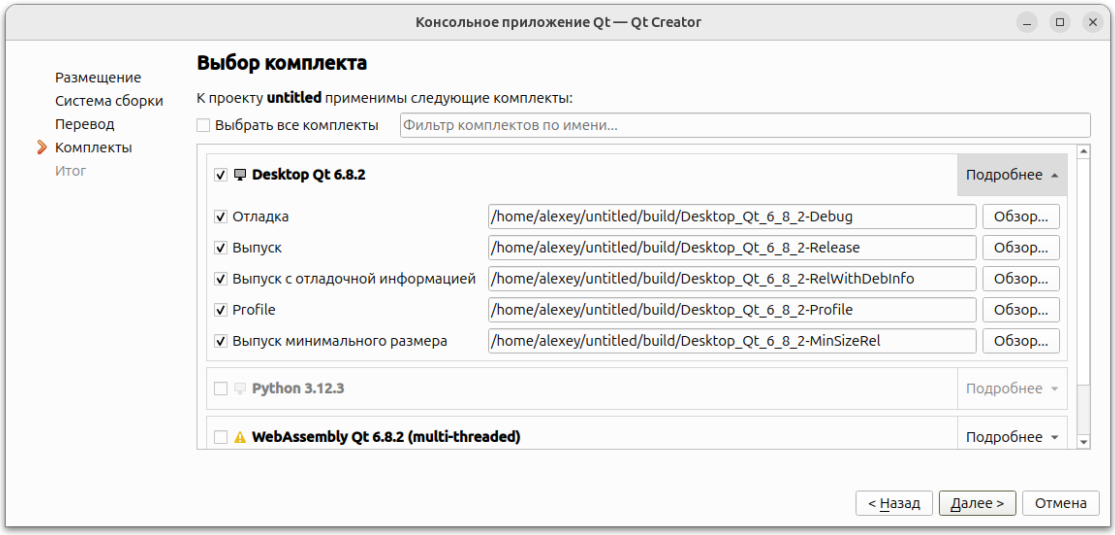
1) Создадим новый проект в Qt Creator (https://doc.qt.io/qtcreator/), выберем **консольное приложение Qt**:



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 (Крейг Скотт, Профессиональный CMake практическое руководство). 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 ниму 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.

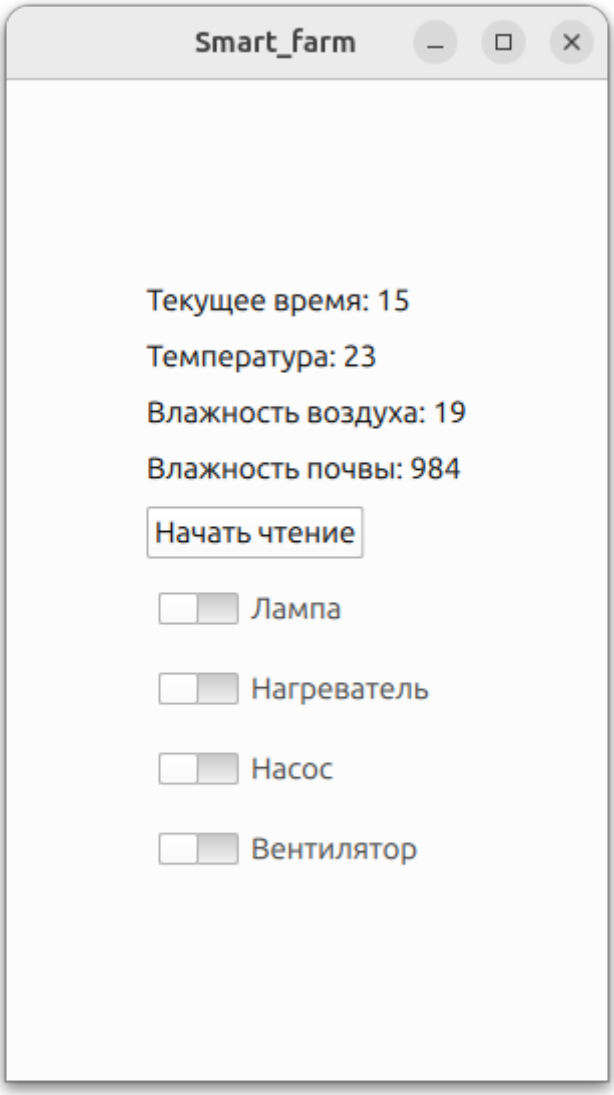
2) Создаем экземпляр класса Sensors_and_leds_handler для работы с датчиками. Прописываем в нем обработку сигнала об изменении значений с датчиков.

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. https://metanit.com/cpp/qt/
2. https://www.qt.io/
3. https://doc.qt.io/qtcreator/
4. Jurgen Bocklage-Ryannel, Cyrill Lorquet, Johan Thelin Qt 6 QML
5. Крейг Скотт, Профессиональный CMake практическое руководство
6. Бьерн Страуструп, Язык программирования C++
7. Саймон Монк, Программируем Arduino