ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ГАЗА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
ИМЕНИ И.М. ГУБКИНА»

ФАКУЛЬТЕТ КОМПЛЕКСНОЙ БЕЗОПАСНОСТИ ТЭК
КАФЕДРА БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторная работа №11

по дисциплине «Специализированные языки и технологии программирования»

на тему «Установка, знакомство и проверка работы Aurora SDK»

Выполнил студент:
группы КА-22-06
Воронин Алексей Дмитриевич

Преподаватель:
Греков Владимир Сергеевич

Москва, 2025

Оглавление

Цель работы 3

Часть 1: Установка и настройка среды 4

Часть 2: Создание нового проекта 4

Часть 3: Разработка и тестирование 4

Часть 4: Документация и поддержка 5

Задание для самостоятельной работы 5

ЗАКЛЮЧЕНИЕ 8

Контрольные вопросы 9

Цель работы

- Ознакомиться с инструментами и возможностями Аврора SDK.
- Создать простое приложение для ОС Аврора.
- Освоить процесс сборки и запуска приложения в эмуляторе.

Часть 1: Установка и настройка среды

1. Установка VirtualBox или Docker: Убедитесь, что VirtualBox или Docker установлены на вашей системе. Если нет, установите необходимое программное обеспечение с официального сайта.
2. Установка Аврора SDK: Следуйте инструкциям графического инсталлятора для установки Аврора SDK на вашу операционную систему.

Часть 2: Создание нового проекта

1. Запуск Аврора IDE: Откройте Аврора IDE. На начальном экране выберите создание нового проекта.
2. Настройка нового проекта:
    - В меню "Файл" выберите "Создать файл или проект...".
    - В разделе "Проекты" выберите "Приложение Qt Quick для ОС Аврора" и нажмите "Выбрать...".
    - Заполните информацию о проекте (название организации, имя проекта, директорию для размещения).
    - Определите необходимые разрешения для вашего приложения.
    - Выберите комплекты для сборки (например, arm7hl для мобильных устройств или i486 для эмулятора).

Часть 3: Разработка и тестирование

1. Редактирование кода: В режиме редактора внесите необходимый код и ресурсы в ваш проект.
2. Сборка проекта:
    - В режиме "Проекты" настройте параметры сборки.
    - Выберите тип сборки: Debug для разработки или Release для финальной версии.
    - Нажмите на кнопку сборки для компиляции вашего проекта.
3. Запуск и тестирование в эмуляторе:

- Настройте параметры запуска, выберите устройство или эмулятор для тестирования.
- Запустите собранное приложение в эмуляторе Aurora OS Emulator, чтобы проверить его функциональность.

Часть 4: Документация и поддержка

1. Использование справки: Изучите доступную документацию в режиме "Справка" для углубленного понимания работы с Аврора SDK.
2. Подключение к системам контроля версий: Настройте интеграцию вашего проекта с выбранной системой контроля версий через Аврора IDE.

Задание для самостоятельной работы

В рамках самостоятельной работы необходимо реализовать таймер с обратным отсчетом. Для этого в MainPage.qml добавить следующее:

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    // Время таймера
    property int seconds: 0
    property int minutes: 0
    property int hours: 0
    property bool running: false
    property int totalSeconds: 0

    Timer {
        id: timer
        interval: 1000
        repeat: true
        onTriggered: {
            if (totalSeconds > 0) {
                totalSeconds--
                hours = Math.floor(totalSeconds / 3600)
                minutes = Math.floor((totalSeconds % 3600) / 60)
                seconds = totalSeconds % 60
            } else {
                timer.stop()
                running = false
            }
        }
    }

    function resetTimer() {
```

```
        timer.stop()
        running = false
        seconds = 0
        minutes = 0
        hours = 0
        totalSeconds = 0
    }

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Countdown Timer")
        extraContent.children: [
            IconButton {
                objectName: "aboutButton"
                icon.source: "image://theme/icon-m-about"
                anchors.verticalCenter: parent.verticalCenter
                onClicked: pageStack.push(Qt.resolvedUrl("AboutPage.qml"))
            }
        ]
    }

    Column {
        width: parent.width
        anchors.centerIn: parent
        spacing: Theme.paddingLarge

        // Поля ввода времени (только цифры)
        Row {
            spacing: Theme.paddingLarge
            anchors.horizontalCenter: parent.horizontalCenter

            TextField {
                id: hoursInput
                width: Theme.itemSizeHuge
                placeholderText: qsTr("HH")
                inputMethodHints: Qt.ImhDigitsOnly
            }
            TextField {
                id: minutesInput
                width: Theme.itemSizeHuge
                placeholderText: qsTr("MM")
                inputMethodHints: Qt.ImhDigitsOnly
            }
            TextField {
                id: secondsInput
                width: Theme.itemSizeHuge
                placeholderText: qsTr("SS")
                inputMethodHints: Qt.ImhDigitsOnly
            }
        }

        // Отображение текущего времени
        Label {
            id: timeDisplay
            anchors.horizontalCenter: parent.horizontalCenter
            text: (hours < 10 ? "0" + hours : hours) + ":" +
                  (minutes < 10 ? "0" + minutes : minutes) + ":" +
                  (seconds < 10 ? "0" + seconds : seconds)
            font.pixelSize: Theme.fontSizeHuge
            color: Theme.highlightColor
        }
```

```
        Button {
            text: running ? qsTr("Pause") : qsTr("Start")
            onClicked: {
                if (!running) {
                    if (totalSeconds === 0) {
                        var h = parseInt(hoursInput.text) || 0
                        var m = parseInt(minutesInput.text) || 0
                        var s = parseInt(secondsInput.text) || 0
                        totalSeconds = h * 3600 + m * 60 + s
                        hours = h
                        minutes = m
                        seconds = s
                    }
                    if (totalSeconds > 0) {
                        timer.start()
                        running = true
                    }
                } else {
                    timer.stop()
                    running = false
                }
            }
        }

        Button {
            text: qsTr("Reset")
            onClicked: resetTimer()
        }
    }
}
```

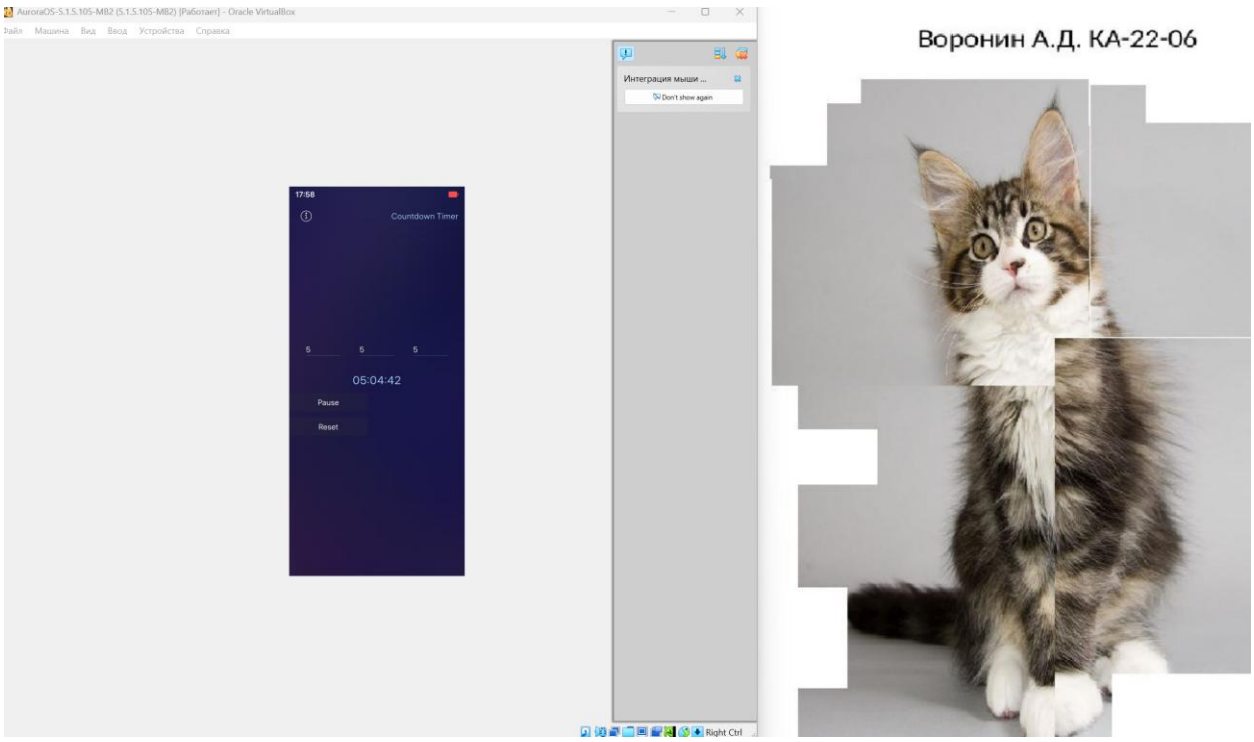

Рисунок 1 – Таймер

ЗАКЛЮЧЕНИЕ

В заключение, подводя итог проделанной работе, все поставленные в начале цели и задания данной лабораторной работы были достигнуты в полном объеме.

Контрольные вопросы

1. Для чего нужен эмулятор Aurora OS?

Эмулятор Aurora OS нужен для тестировании приложений на ПК, что используется в разработке приложений на данную ОС.

2. Какие основные элементы интерфейса можно использовать в Qt Quick?

- TextField
- Label
- Button
- Slider

3. В чем разница между сборкой в режиме Debug и Release?

Debug — режим для разработки и отладки. В программу включается дополнительная отладочная информация, которая позволяет использовать отладчик (например, ставить точки останова, просматривать переменные). Release — режим для финального выпуска программы.

4. Какие типы проектов можно создать в Aurora OS?

- Приложение Qt Quick
- Проект автотестирования
- Проект с подкатегориями
- Пустой проект qmake
- Приложение на C
- Приложение на C++