Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра вычислительных систем
Допустить к защите
зав. кафедрой д.т.н. доцент
_____ Курносов М.Г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Проектирование и разработка модуля для работы с дисплеем и клавиатурой на IP-телефоне VP-15(P)

Пояснительная записка

Студент: Сенченко А.П.

Факультет ИВТ Группа ИВ-621

Руководитель Крамаренко К.Е.

Новосибирск - 2020

Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

КАФЕДРА
вычислительных систем

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

СТУДЕНТУ Сенченко А.П. ГРУППЫ ИВ-621

«УТВЕРЖДАЮ»
«_____» _____
зав. кафедрой ВС
д.т.н. доцент
_____ Курносов М.Г.

Новосибирск, 2020 г.

1. **Тема выпускной квалификационной работы бакалавра**: «Проектирование и разработка модуля для работы с дисплеем и клавиатурой на IP-телефоне VP-15(P)» утверждена приказом СибГУТИ от «22» мая 2020 г. № 4/700о-20
2. **Срок сдачи студентом законченной работы:** 11 июня 2020 г.

3. **Исходные данные к работе**

1 Qt Documentation Snapshots. URL: https://doc-snapshots.qt.io/qt5-5.11/index.html (дата обращения: 08.05.2020)

4. Содержание пояснительной записки (перечень подлежащих разработке вопросов)	**Сроки выполнения по разделам**
Постановка задачи	01.03.20 - 07.03.20
Проектирование архитектуры приложения	08.03.20 - 27.03.20
Изучение характеристик устройства	28.03.20 - 29.03.20
Кросс-компиляция Qt под устройство	30.03.20 - 30.04.20
Выбор формата конфигурации меню и его парсера	01.05.20 - 08.05.20
Проектирование конфигурации	09.05.20 - 16.05.20
Парсинг файла конфигурации	17.05.20 - 24.05.20
Реализация модуля при помощи Qt	25.05.20 - 11.06.20

Дата выдачи задания: «03» февраля 2020 г.

Руководитель _____ Крамаренко К.Е.

Задание принял к исполнению «03» февраля 2020 г.

Студент _____ Сенченко А.П.

АННОТАЦИЯ

Выпускная квалификационная работа Сенченко А.П.
по теме «Проектирование и разработка модуля для работы с дисплеем и клавиатурой на IP-телефоне VP-15(P)»

Объём работы 38 страниц, на которых размещены 8 рисунков. При написании работы использовалось 7 источников.

Ключевые слова: вычислительная система, трансляционные обмены.

Работа выполнена на кафедре ВС СибГУТИ.
Руководитель – старший преподаватель Крамаренко К.Е.,

Целью бакалаврской работы было проектирование и разработка модуля для работы с дисплеем и клавиатурой на IP-телефоне VP-15(P).

Инструментами разработки стали C++ и Qt. C++ – компилируемый, статически типизированный язык программирования. Qt – это кросс-платформенный фреймворк предназначенный для упрощенного создания пользовательского интерфейса.

В работе описаны характеристики устройства, этап подготовки к разработке, проектировка конфигурации и реализация модуля. Программное обеспечение написано для устройства под управлением операционной системы Linux. Данный модуль позволяет пользователю взаимодействовать с устройством.

В рамках дипломного проекта был спроектирован и разработан модуль для работы с дисплеем и клавиатурой на IP-телефоне VP-15(P).

Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

ОТЗЫВ

на выпускную квалификационную работу студента
группы ИВ-621 Сенченко А.П.
по теме «Проектирование и разработка модуля для работы с дисплеем и клавиатурой на IP-телефоне VP-15(P)»

Выпускная квалификационная работа Сенченко А.П. посвящена Проектирование и разработка модуля для работы с дисплеем и клавиатурой на IP-телефоне VP-15(P). Инструментами реализации были язык C++ и фреймворк Qt. Дано обоснование средств реализации. Рассмотрены кросс-компиляция под целевую платформу, возможные форматы для конфигурации, основы проектирования приложений под операционную систему Linux.

В результате анализа предметной области студентом была разработана конфигурация, описывающая меню, и модуль применяющий загруженные настройки. Материал в работе изложен с соблюдением логики, между разделами существует логическая взаимосвязь.

Выпускная квалификационная работа имеет практическое значение, реализованный модуль дает возможность пользователю взаимодействовать с устройством.

Оригинальности работы составляет 92,76%. Проверка осуществлялась на сайте antiplagiat.ru.

К недостаткам работы следует отнести присутствие «артефактов» при переходе между экранами меню, но это не мешает пользователю, так как на итоговом экране их нет, тем не менее, данная работа полностью реализует поставленные задачи и заслуживает оценки «отлично», а ее автор, Сенченко А.П. – присвоения квалификации «бакалавр» по направлению 09.03.01 «Информатика и вычислительная техника».

Оценка уровней сформированности общекультурных и профессиональных компетенций обучающегося:

Компетенции		Уровень сформированности Компетенций		
		Высокий	Средний	Низкий
Общекультурные	ОК-7 - способностью к самоорганизации и самообразованию	V		
Общепрофессиональные	ОПК-4 - способностью участвовать в настройке и наладке программно-аппаратных комплексов	V		
	ОПК-5 - способностью решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	V		
Профессиональные	ПК-1 - способностью разрабатывать модели компонентов информационных систем, включая модели баз данных и модели и интерфейсов человек – электронно-вычислительная машина	V		
	ПК-2 - способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	V		
	ПК-3 - способностью обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности	V		

Работа имеет практическую ценность	v	Тема предложена предприятием	v
Работа внедрена		Тема предложена студентом	
Рекомендую работу к внедрению	v	Тема является фундаментальной	
Рекомендую работу к опубликованию	v	Рекомендую студента в магистратуру	v
Работа выполнена с применением ЭВМ	v	Рекомендую студента в аспирантуру	

старший преподаватель кафедры ВС
ФГОБУ ВПО «СибГУТИ» _____ Крамаренко К.Е.
(Крамаренко Константин Евгеньевич)
«_____» _____

Содержание

1 ВВЕДЕНИЕ 8
2 ПОСТАНОВКА ЗАДАЧИ 9
3 ХАРАКТЕРИСТИКИ УСТРОЙСТВА 10
4 ПОДГОТОВКА УСТРОЙСТВА 11
5 РЕАЛИЗАЦИЯ 13
5.1 Конфигурация 13
5.2 Кроссплатформенный фреймворк Qt 17
5.3 Фрейм-буфер LinuxFB 19
5.4 Работа с клавиатурой 20
5.5 Модуль для работы с дисплеем и клавиатурой 20
6 РЕЗУЛЬТАТЫ 23
7 ЗАКЛЮЧЕНИЕ 26
ПРИЛОЖЕНИЕ А 27
ПРИЛОЖЕНИЕ Б 28
ПРИЛОЖЕНИЕ В 29

1 ВВЕДЕНИЕ

IP-телефония – реализация телефонной сети на основе сетевого оборудования. Данная технология позволяет решить ряд вопросов, которые могут возникнуть у компаний:

- Сокращение затрат на связь
- Повышение информационной безопасности, так как все разговоры проходят по внутренней сети компании.
- Взаимодействие с сотрудниками работающими удаленно.

2 ПОСТАНОВКА ЗАДАЧИ

В рамках выпускной квалификационной работы необходимо спроектировать и разработать модуль для работы с дисплеем и клавиатурой для IP-телефоне VP-15(P) производства «Элтекс».

В процессе работы нужно:

1) Изучить характеристик IP-телефона
2) Подготовить устройство к разработке модуля
3) Выбор формата конфигурации
4) Проектирование конфигурации
5) Синтаксический анализ конфигурации
6) Реализация модуля в соответствии с разработанной конфигурацией

3 ХАРАКТЕРИСТИКИ УСТРОЙСТВА

Характеристики IP-телефона VP-15(P):

- ОС: Linux
- Процессор: Realtek 8972C
- SDRAM: 128 МБайт
- SPI Flash: 16 МБайт

VP-15(P) присутствует на рисунке 3.1.

Рисунок 3.1 – VP-15(P)

В процессе разработки стало понятно, что из-за цепочки инструментов устаревшей версии, лежащей в основе VP-15(P), нет возможности полноценной реализовать работу. В связи с этим было принято решение продолжить разработку на устройстве VP-20. VP-20 присутствует на рисунке 3.2.

Характеристики IP-телефона VP-20:

- ОС: Ubuntu 14.04 LTS
- Процессор: ARM Cortex-A53 1.2 GHz × 2
- DRAM: 1 ГБайт
- Flash: 128 МБайт

Рисунок 3.2 – VP-20

4 ПОДГОТОВКА УСТРОЙСТВА

Сборка Qt осуществлялось при помощи набора инструментов версии 1.5.5, предоставляемый Realtek, который совместим с центральными процессорами Lexra на базе MIPS, которые не реализуют выровненные нагрузки и хранилища. Версией устанавливаемой библиотеки Qt была выбрана 4.8, так как более новый выпуски данной библиотеки, вызывали множество ошибок при кросс-компиляции.

Первым делом было необходимо настроить файл конфигурации для своей архитектуры. Данная конфигурация присутствует в листинге 4.1.

Листинг 4.1 – Демонстрация конфигурации под целевое устройство

```
MAKEFILE_GENERATOR = UNIX
CONFIG += incremental
QMAKE_INCREMENTAL_STYLE = sublib

include(../common/linux.conf)
include(../common/gcc-base-unix.conf)
include(../common/g++-unix.conf)

QMAKE_COMPILER_DEFINES += QT_ENABLE_CXX11

CROSS_COMPILE = mips-linux-

QMAKE_CC = $${CROSS_COMPILE}gcc
QMAKE_CXX = $${CROSS_COMPILE}g++
QMAKE_LINK = $${CROSS_COMPILE}g++

QMAKE_CFLAGS += -Os
QMAKE_CXXFLAGS += -Os

QMAKE_AR = $${CROSS_COMPILE}ar cqs
QMAKE_STRIP = $${CROSS_COMPILE}strip
QMAKE_RANLIB = $${CROSS_COMPILE}ranlib

load(qt_config)
```

В конфигурации необходимо указать:

- QMAKE_COMPILER_DEFINES – стандарт
- QMAKE_CC – определяет компилятор C
- QMAKE_CXX – определяет компилятор C ++
- QMAKE_LINK – компоновщик
- QMAKE_CFLAGS – определяет флаги компилятора C
- QMAKE_CXXFLAGS – определяет флаги компилятора C++
- QMAKE_AR – команда для создании общей библиотеки
- QMAKE_STRIP – имя файла для удаления таблицы символов из объектных файлов
- QMAKE_RANLIB – определяет файл построения библиотеки

При конфигурации библиотеки были оставлены только необходимые компоненты, так как на устройстве нет достаточно места, чтобы хранить Qt со всем возможным функционалом. Опции которые были использованы:

- -shared – создать и использовать динамические библиотеки Qt
- -embedded mips – включить встроенную сборку
- -xplatform qws/linux-mips-g++ – целевая платформа при кросс-компиляции
- -big-endian – целевая платформа с прямым порядком байтов
- -no-audio-backend – не компилировать в платформе аудио бэкэнд
- -no-accessibility – не компилировать поддержку Windows Active Accessibility
- -no-webkit – не компилировать в модуле WebKit
- -no-gif – не компилировать плагин для поддержки чтения GIF
- -no-libtiff – не компилировать плагин для поддержки TIFF
- -no-libpng – не компилировать в поддержку PNG
- -no-libmng – не компилировать плагин для поддержки MNG
- -no-libjpeg – не компилировать плагин для поддержки JPEG
- -no-multimedia – не компилировать мультимедийный модуль
- -no-script – не собирайте модуль QtScript.
- -no-system-proxies – не использовать системные сетевые прокси по умолчанию
- -no-stl – не компилировать поддержку STL

Сборка и установка прошла. При компиляции тестового приложения произошла ошибка.

В процессе анализа проблемы было выяснено, что ядро Linux и все библиотеки были собраны при помощи набора инструментов версии 1.3.6. Версия компилятора была 3.4.6, а это означало, что Qt не получится установить на это устройство, так как данная версия компилятора является слишком, низкой. Было принято решение перейти на устройство VP-20, где в будущем и должен был использоваться разрабатываемый модуль.

На VP-20 появилась возможность использовать версию Qt 5.11.

5 РЕАЛИЗАЦИЯ

5.1 Конфигурация

5.1.1 Выбор формата

Для конфигурации рассматривалось 3 формата:

- XML
- JSON
- YAML

Чтобы сделать выбор, необходимо составить примеры в разных форматах и сравнить их между собой.

Примеры массивов и строк приедены в листингах 5.1, 5.2 и 5.3.

Листинг 5.1 – Демонстрация массивов и строк в формате XML

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

Листинг 5.2 – Демонстрация массивов и строк в формате JSON

```
{
  "menu": {
    "id": "file",
    "value": "File",
    "popup": {
      "menuitem": [
        {
          "value": "New",
          "onclick": "CreateNewDoc()"
        },
        {
          "value": "Open",
          "onclick": "OpenDoc()"
        },
        {
          "value": "Close",
          "onclick": "CloseDoc()"
        }
      ]
    }
  }
}
```

Листинг 5.3 – Демонстрация массивов и строк в формате YAML

```
menu:
  id: file
  value: "File"
  popup:
    menuitem:
      - value: 'New'
        onclick: "CreateNewDoc()"
      - value: 'Open'
        onclick: "OpenDoc()"
      - value: 'Close'
        onclick: "CloseDoc()"
```

Спец символы, в листингах 5.4, 5.5 и 5.6 присутствуют примеры по выделению строки в кавычки.

Листинг 5.4 – Демонстрация спец символа «кавычки» в формате XML

```
<file name="&amp;quot;filename&amp;quot;"/>
```

Листинг 5.5 – Демонстрация спец символа «кавычки» в формате YAML

```
{
  "file":
  {
    name: "\"filename\""
  }
}
```

Листинг 5.6 – Демонстрация спец символа «кавычки» в формате YAML

```
file:
  name: "\"filename\""
```

Примеры длинных строк продемонстрированы в листингах 5.7, 5.8 и 5.9.

Листинг 5.7 – Демонстрация длинных строк в формате XML

```
<text>
  line1
  line2
  line3
  line4
</text>
```

Листинг 5.8 – Демонстрация длинных строк в формате JSON

```
{
  "text" : "line1\nline2\nline3\nline4"
}
```

Листинг 5.9 – Демонстрация длинных строк в формате YAML

```
text: >
  line1
```

```
  line2
  line3
  line4
```

Сравнивая конфигурации можно выделить следующее:

- Строки в форматах ничем не отличаются пользоваться одинаково удобно везде
- Длинными строками, JSON не позволяет разбивать длинную строку на несколько строк.
- Спец символы удобнее использовать в JSON и YAML, так как применяется простое экранирование. В XML же чтобы воспользоваться одним из зарезервированных слов, требуется знать псевдонимы соответствующие им.
- Массивы имеются везде, но лучше всего показался YAML, так нет лишних скобочек или множественное повторение принадлежности списку, как в JSON или XML.

Конечным выбором стал YAML.

5.1.2 Проектирование конфигурации

Пример конфигурации присутствует в листинге 5.10.

Листинг 5.10 – Демонстрация спроектированной конфигурации

```
screensTree:
  menu:
    status: null
    story: null

buttons:
  buttonBack:
    typeButton: back
    label:
      ru: "Назад"
      eng: "Back"
    key: "f1"
  buttonOk:
    typeButton: ok
    label:
      ru: "Ок"
      eng: "Ok"
    key: f4
  buttonGotoMenu:
    typeButton: goto
    screen: "menu"
    key: "f4"

screens:
  menu:
    label:
      ru: "Меню"
      eng: "Menu"
    typeScreen: list
```

```
    states:
      default:
        includeButtons:
          f1: buttonBack
          f3: buttonOk
  status:
    label:
      ru: "Статус"
      eng: "Status"
    typeScreen: list
    states:
      default:
        includeButtons:
          f1: buttonBack
          f3: buttonOk
  story:
    label:
      ru: "История"
      eng: "Story"
    typeScreen: list
    states:
      default:
        includeButtons:
          f1: buttonBack
          f3: buttonOk

root:
  label: "Root"
  typeScreen:
  icons:
    missedCall:
      file: "missedCall.bitmap"
      position:
        x: 1
        y: 1
    message:
      file: "message.bitmap"
      position:
        x: 2
        y: 1
  states:
    default:
      includeButtons:
        f1: buttonBack
        f3: buttonGotoMenu
```

Назначение параметров:

- screensTree – отображает древовидную структуру меню.
- buttons – массив с описанием кнопок.
- typeButton – тип кнопки. Существует кнопки с уже определенным поведением: back, ok, call, add, delete, info. Тип goto требует, чтобы поведение кнопки было определено далее в конфигурации.

- label – массив, содержащие информацию какую подпись требуется отобразить на экран.
- ru – поле, отображающее подпись на русском языке.
- eng – поле, отображающее подпись на английском языке.
- key – псевдоним клавиши на клавиатуре устройства. Этот параметр означает привязку кнопки на экране к клавише клавиатуры.
- screen – на какой экран необходимо перейти, при нажатии кнопки. Используется только при типе кнопки goto.
- screens – массив описывающий все возможные экраны.
- typeScreen – тип экрана. Данное поле говорит, как именно расположить дочерние элементы на экране. Возможные значения: grid, list, input.
- states – список различных состояний экрана. Каждый экран может иметь множество состояний.
- includeButtons – массив кнопок, которые присутствуют на экране. Для каждого состояния отдельно можно прописать поведение кнопок.
- root – описание начального экрана, когда устройство запустилось.
- icons – массив описывающий иконки предназначенных для определенных событий. Например, пропущенный звонок (missedCal) или входящее сообщение (message).
- file – обозначает имя файла, в котором содержится иконка.
- position – массив с позициями иконки.
- x – числовое значение по оси Ox.
- y – числовое значение по оси Oy.

С помощью данного конфигурации появится возможность создавать новые пункты меню и добавлять объекты на экран без необходимости делать изменения в исходном коде.

5.1.3 Синтаксический анализ

Для синтаксического анализа конфигурации в формате YAML была выбрана библиотека yaml-cpp.

В листингах В.1, В.2, В.3, В.4, В.5, В.6 и В.7 продемонстрированы классы созданные для хранения информации прочитанной из конфигурации.

В листинге В.8 продемонстрирован пример синтаксического анализа одного из экранов блока screens.

5.2 Кроссплатформенный фреймворк Qt

Qt – это полная среда разработки с инструментами, предназначенными для упрощения создания приложений и пользовательских интерфейсов для настольных, встраиваемых и мобильных платформ [6].

QWidget – является базовым классом всех объектов пользовательского интерфейса. Виджет является атомом пользовательского интерфейса: он получает мышь, клавиатуру и другие события из оконной системы и отображает свое представление на экране. Каждый виджет прямоугольный, и они отсортированы в Z-порядке. Виджет обрезается его родителем и виджетами перед ним [6].

Виджет, который не встроен в родительский виджет, называется окном. Обычно окна имеют рамку и строку заголовка, хотя также возможно создавать окна без такого оформления, используя подходящие флажки окон) [6].

Z-порядок – это порядок наложения двухмерных объектов, таких как окна в оконном менеджере стека, формы в редакторе векторной графики или объекты в 3D-приложении. Одной из особенностей типичного графического интерфейса пользователя является то, что окна могут перекрываться, так что одно окно скрывает часть или все другое. Когда два окна перекрываются, их Z-порядок определяет, какое из них появляется поверх другого [6].

QMainWindow – главное окно приложения. Главное окно предоставляет структуру для создания пользовательского интерфейса приложения. Qt имеет QMainWindow и связанные с ним классы для управления главным окном. QMainWindow имеет свой собственный макет, к которому можно добавить QToolBars, QDockWidgets, QMenuBar и QStatusBar. Макет имеет центральную область, которая может быть занята любым виджетом [6].

QPushButton – командная кнопка. Командная кнопка, наиболее часто используемый виджет в любом графическом интерфейсе пользователя [6].

Командная кнопка имеет прямоугольную форму и обычно отображает текстовую метку, описывающую ее действие [6].

Кнопка генерирует сигнал clicked(), когда она активируется мышью или сочетанием клавиш. Чтобы выполнить действие кнопки, нужно подключитесь к этому сигналу. Кнопки также предоставляют менее часто используемые сигналы, например, pressed() и released() [6].

Командные кнопки в диалоговых окнах по умолчанию являются автоматическими кнопками, то есть они автоматически становятся кнопками по умолчанию, когда получают фокус ввода с клавиатуры. Кнопка по умолчанию – это кнопка, которая активируется, когда пользователь нажимает клавишу Enter или Return в диалоговом окне. Это можно изменить с помощью setAutoDefault() [6].

QLabel используется для отображения текста или изображения. Функциональность взаимодействия с пользователем не предоставляется. Внешний вид метки можно настраивать различными способами, и его можно использовать для указания мнемонического ключа фокуса для другого виджета [6].

QGridLayout берет пространство, предоставленное ему (его родительским макетом или parentWidget()), делит его на строки и столбцы и помещает каждый виджет, которым он управляет, в правильную ячейку. Столбцы и строки ведут себя одинаково [6].

Обычно каждый управляемый виджет или макет помещается в отдельную ячейку с помощью addWidget(). Также возможно, чтобы виджет занимал несколько ячеек, используя перегрузки строк и столбцов addItem() и addWidget(). Если сделаете это, QGridLayout будет угадывать, как распределить размер по столбцам/строкам (основываясь на коэффициентах растяжения) [6].

Чтобы удалить виджет из макета, нужно вызвать removeWidget(). Вызов QWidget::hide() для виджета также эффективно удаляет виджет из макета до вызова QWidget::show() [6].

После добавления макета, можно помещать виджеты и другие макеты в ячейки сетки, используя addWidget(), addItem() и addLayout() [6].

QString хранит строку 16-битных QChar, где каждый QChar соответствует одной единице кода UTF-16. (Символы Unicode со значениями кода выше 65535 хранятся с использованием суррогатных пар, то есть двух последовательных символов QChars.) [6].

Unicode – это международный стандарт, который поддерживает большинство систем письма, используемых сегодня. Это расширенный набор символов US-ASCII (ANSI X3.4-1986) и Latin-1 (ISO 8859-1), и все символы US-ASCII / Latin-1 доступны в одинаковых позициях кода [6].

Внутри QString использует неявное совместное использование (копирование при записи), чтобы уменьшить использование памяти и избежать ненужного копирования данных. Это также помогает сократить накладные расходы на хранение 16-битных символов вместо 8-битных [6].

В дополнение к QString, Qt также предоставляет класс QByteArray для хранения необработанных байтов и традиционных 8-разрядных строк с завершением '\0'. QString используется в Qt API, а поддержка Unicode гарантирует, что ваши приложения будут легко переводить, если вы захотите расширить рынок приложений в какой-то момент. Два основных случая, когда QByteArray уместен, это когда вам нужно хранить необработанные двоичные данные и когда критически важно сохранение памяти (как во встраиваемых системах) [6].

QBoxLayout выстраивает дочерние виджеты горизонтально или вертикально [6].

QBoxLayout берет пространство, которое он получает (из его родительского макета или из parentWidget()), делит его на ряд блоков и заставляет каждый управляемый виджет заполнять один блок [6].

Если ориентация QBoxLayout – Qt::Horizontal, то поля располагаются в ряд с подходящими размерами. Любое избыточное пространство распределяется в соответствии с факторами растяжения [6].

Если ориентация QBoxLayout – Qt::Vertical, поля помещаются в столбец, опять же с подходящими размерами [6].

Самый простой способ создать QBoxLayout – использовать один из вспомогательных классов, например, QHBoxLayout (для блоков Qt::Horizontal) или QVBoxLayout (для блоков Qt::Vertical). Также можно напрямую использовать конструктор QBoxLayout, указав его направление как LeftToRight, RightToLeft, TopToBottom или BottomToTop [6].

Если QBoxLayout не является макетом верхнего уровня (то есть он не управляет всей областью виджета и дочерними элементами), нужно добавить его в родительский макет, прежде чем что-либо делать с ним [6].

Сигналы и слоты используются для связи между объектами. Механизм сигналов и слотов является центральной особенностью Qt, которая больше всего отличается от функций, предоставляемых другими платформами. Сигналы и слоты становятся возможными благодаря мета-объектной системе Qt [6].

Обратный вызов – это указатель на функцию.

5.3 Фрейм-буфер LinuxFB

Qt-приложение выводит информацию на экран при помощи плагина записывающий напрямую в фрейм-буфер через подсистему Linux fbdev.

Для запуска Qt-приложения необходимо указать в переменной окружения платформу, с помощью которой будет производиться вывод на экран. Изменение переменной окружения приведена в листинге 5.11.

Листинг 5.11 – Демонстрация указания используемой платформы

```
export QT_QPA_PLATFORM=LinuxFB
```

Qt найдет устройство в стандартном месте: /dev/fb0.

5.4 Работа с клавиатурой

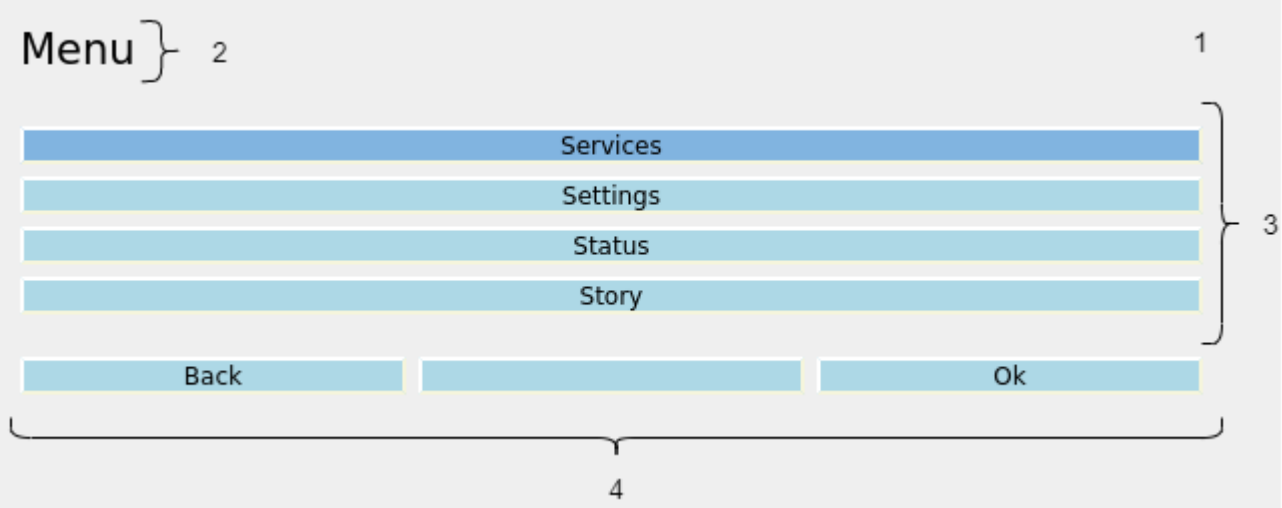
Для того, что бы Qt мог захватить клавиатуру необходимо в переменной окружения указать путь до устройства. В листинге 5.12 продемонстрированно указания пути до устройства.

Листинг 5.12 – Демонстрация указания пути до устройства клвиатуры

```
export TSLIB_TSDEVICE=/dev/input/event1
export QT_QPA_GENRIC_PLUGINS=\
    tslib:/dev/input/event1,\
    evdevkeyboard:/dev/input/event0,\
    evdevkeyboard:/dev/input/event2
```

5.5 Модуль для работы с дисплеем и клавиатурой

Конфигурация меню читается один раз, при запуске устройства. На основе конфигурации происходит построение пользовательского меню. Пример начального экрана приведен на рисунке 5.1.



1 Главное окно, 2 Подпись экрана, 3 Поля пунктов меню, 4 Функциональные клавиши

Рисунок 5.1 – Демонстрация начального экрана меню

5.5.1 Главное окно

Главное окно представлено классом MenuWindow, который представлен в листинге В.9.

Каждому пункту меню соответствует подобный экран, котором располагаются 3 логических элемента: подпись, поля пунктов для следующего

перехода и функциональные клавиши. Данные элементы формируются в соответствие с прочитанной конфигурацией.

5.5.2 Подпись экрана

Данное поле представлено классом QLabel. Отображает текущий экран, на котором находится пользователь.

5.5.3 Поля пунктов меню

Каждое поле представлено классом QPushButton. Данные кнопки нужны, чтобы по их нажатию произошло действие перехода на соответствующий пункт меню. Данное действие осуществляется при помощи привязки сигнала к слоту. Также при переходе на следующий экран необходимо закрыть текущее окно. Пример привязки сигнала к слоту приведен в листинге 5.14.

Листинг 5.13 – Демонстрация привязки сигнала к слоту для перехода на следующий экран

```
MenuWindow *servicesWindow = new MenuWindow();
QPushButton *servicesPushButton = new QPushWindow();

connect(servicesPushButton, SIGNAL(clicked ()),
        servicesWindow, SLOT(show()));
connect(servicesPushButton, SIGNAL(clicked ()),
        this, SLOT(close()));
```

Все пункты меню находятся в QVBoxLayout.

Чтобы показать на какой пункт меню наведен курсор, необходимо поменять стиль кнопки. Пример смены стиля кнопки приведен в листинге 5.15.

Листинг 5.14 – Демонстрация изменения стиля кнопки на активную

```
QString buttonActiveBackgroundColor =
            "background-color: rgb(129, 180, 224)";
button->setStyleSheet(buttonActiveBackgroundColor);
```

Пример изменения наведения курсора на кнопку приведен в листинге 5.16.

Листинг 5.15 – Демонстрация изменения стиля на неактивную

```
QString  buttonDefaultBackgroundColor =
                        "background-color: lightblue";
button->setStyleSheet(buttonDefaultBackgroundColor);
```

Все кнопки находятся в QVBoxLayout.

5.5.4 Функциональные клавиши

Каждая клавиши представлены классом Softkey, представленным в листинге 5.17.

Листинг 5.16 – Демонстрация класса Softkey

```
#ifndef SOFTKEY_H
#define SOFTKEY_H

#include <QObject>
#include <QPushButton>

class Softkey : public QPushButton
{
    Q_OBJECT

public:
    explicit Softkey(QWidget *parent = nullptr);
    explicit Softkey(const QString& text,
                     QWidget *parent = nullptr);
    Softkey(const QIcon& icon,
            const QString &text,
            QWidget *parent = nullptr);
    ~Softkey();

signals:
    void moveToWindow();
};

#endif // SOFTKEY_H
```

При разработке появилась необходимость в дополнительных сигналов у кнопок. Поэтому был создан класс Softkey расширяющий стандартный QPushButton.

6 РЕЗУЛЬТАТЫ

Для демонстрации работы в целом была сформирована тестовая версия конфигурации, приведенная в листинге В.10. На рисунках 6.1, 6.2, 6.3, 6.4 и 6.5 продемонстрирована визуальная составляющая меню в соответствии с конфигурацией.

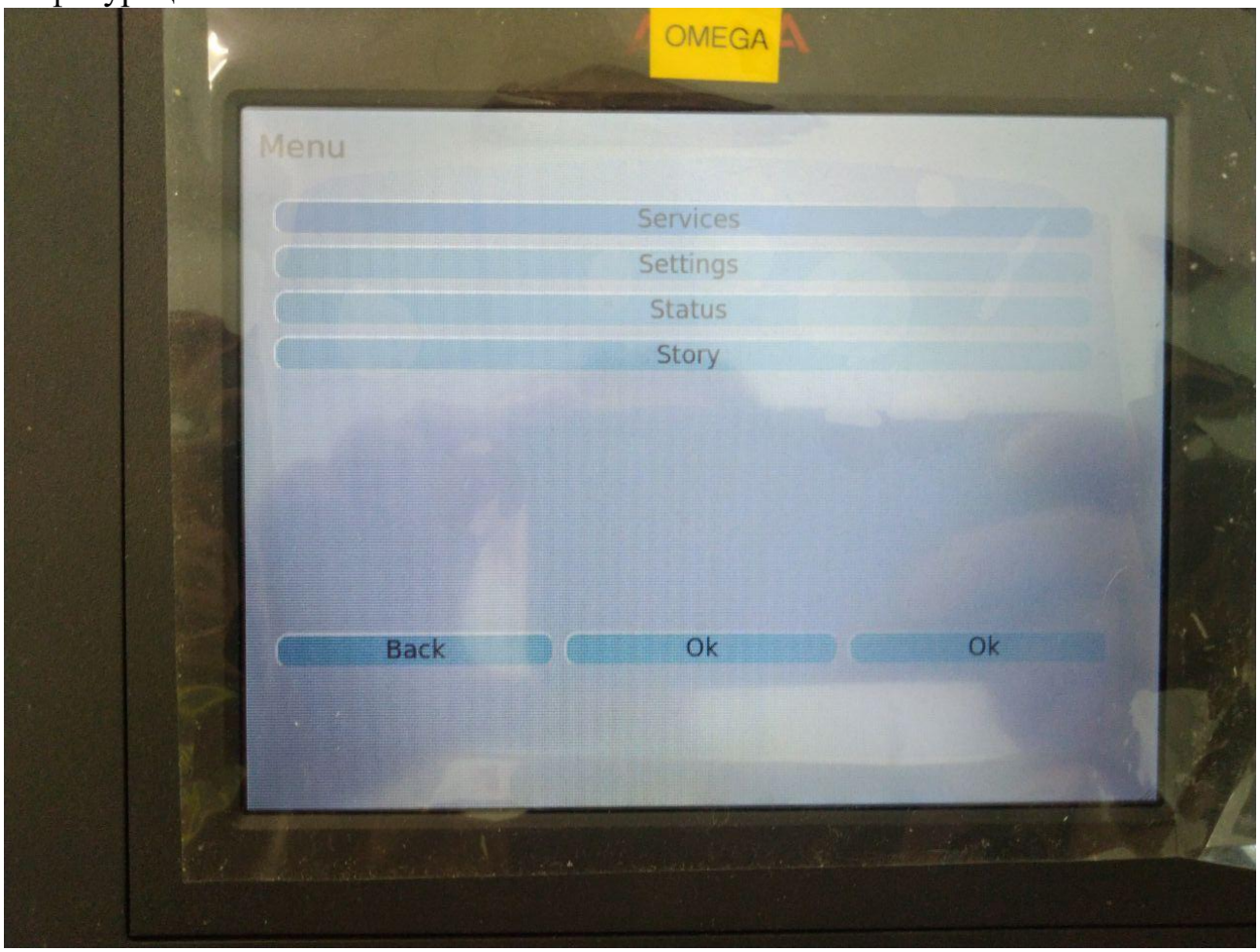


Рисунок 6.1 – Демонстрация экрана Menu

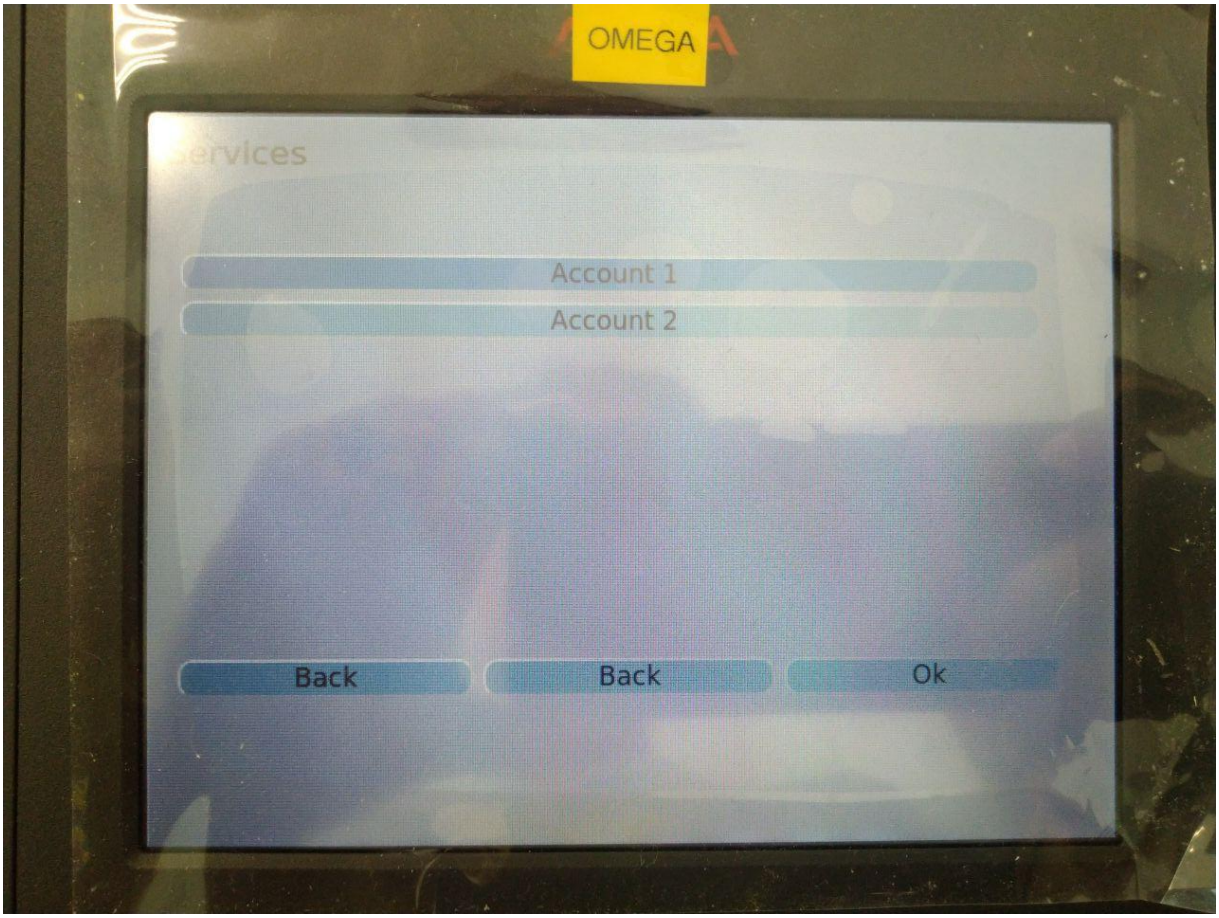


Рисунок 6.2 – Демонстрация экрана Services

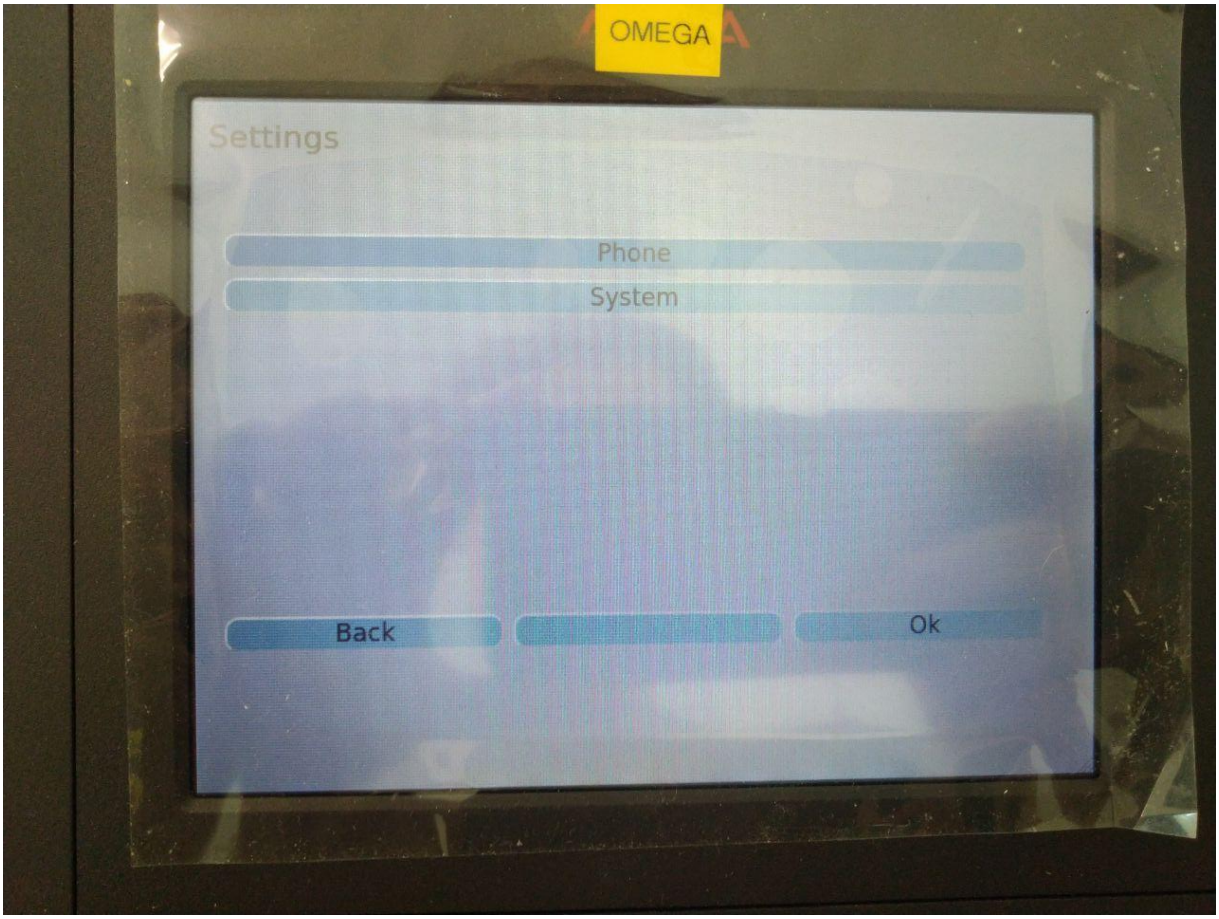


Рисунок 6.3 – Демонстрация экрана Settings

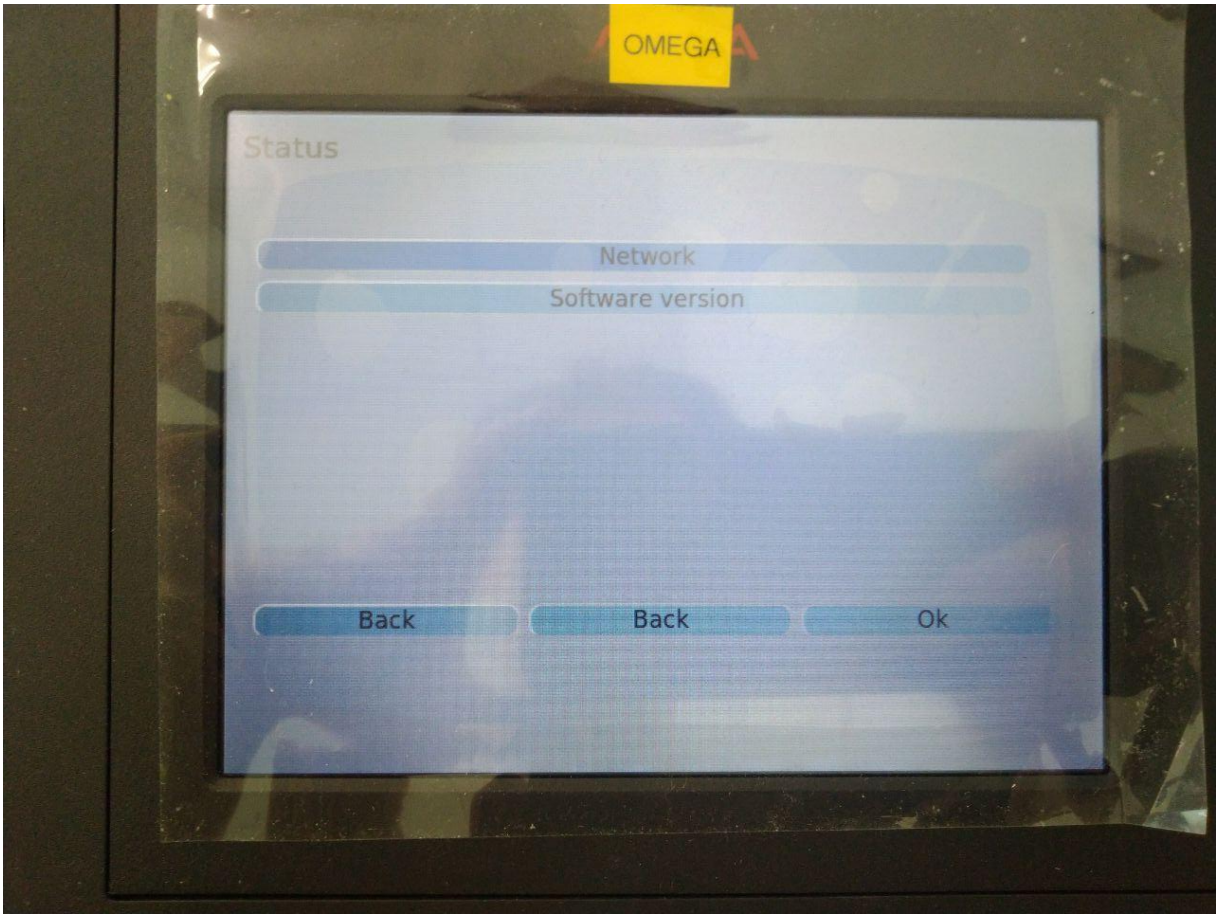


Рисунок 6.4 – Демонстрация экрана Status

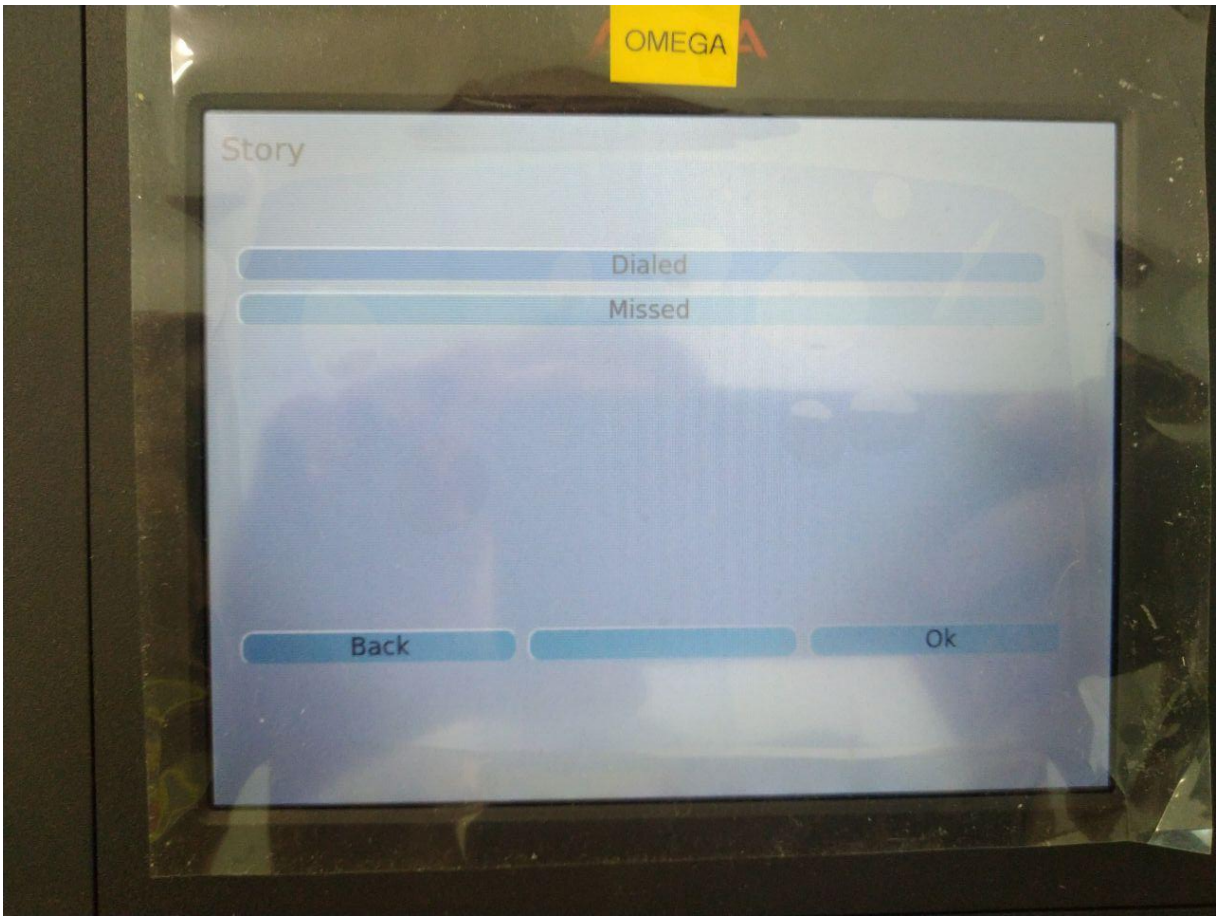


Рисунок 6.5 – Демонстрация экрана Story

7 ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было спроектирован и разработан модуль для работы с дисплеем и клавиатурой для IP-телефоне VP-15(P) производства «Элтекс».

В процессе работы:

1) Изучены характеристик IP-телефона
2) Подготовлено устройство к разработке модуля
3) Выбран формат конфигурации
4) Спроектирована конфигурация
5) Произведен синтаксический анализ конфигурации
6) Реализован модуль в соответствии с разработанной конфигурацией

ПРИЛОЖЕНИЕ А

(справочное)
Библиография

1 Крис Симмондс, Встраиваемые системы на основе Linux. М. : ДМК Пресс, 2017. 360с.
2 Курносов М.Г. Введение в структуры и алгоритмы обработки данных. М. : Автограф, 2015. 179с.
3 Tree – убийца JSON, XML, YAML и иже с ними. URL: https://habr.com/ru/post/248147/ (дата обращения: 08.05.2020)
4 Шлее М. Qt 5.10. Профессиональное программирование на C++. – СПб.: БХВ-Петербург, 2018. – 1072 с.: ил. – (В подлиннике)
5 Саммерфилд М. Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 560 с., ил.
6 Qt Documentation Snapshots. URL: https://doc-snapshots.qt.io/qt5-5.11/index.html (дата обращения: 08.05.2020)
7 Tutorial. URL: https://github.com/jbeder/yaml-cpp/wiki/Tutorial (дата обращения: 08.05.2020)

ПРИЛОЖЕНИЕ Б

(рекомендуемое)
Наиболее употребляемые текстовые сокращения

XML(eXtensible Markup Language) – язык разметки
YAML(Yet Another Markup Language) – язык разметки
JSON(JavaScript Object Notation) – формат обмена данными
ОС – операционная система
МБайт – мегабайт
ГБайт – гигабайт
LTS(Long term support) – долгосрочная поддержка
ARM(Advanced RISC Machine) – микропроцессорная архитектура с сокращенным набором команд
GHz – гигагерц
SDRAM(Double Data Rate Synchronous Dynamic Random Access Memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных
SPI(Serial Peripheral Interface) – последовательный периферийный интерфейс
DRAM(Dynamic Random Access Memory) – энергозависимая память
GIF(Graphics Interchange Format) – растровый формат графических изображений
TIFF(Tagged Image File Format) – растровый формат графических изображений
PNG(Portable Network Graphics) – растровый формат графических изображений
MNG(Multiple-image Network Graphics) – растровый формат графических изображений
JPEG(Joint Photographic Experts Group) – растровый формат графических изображений
STL(Standard Template Library) – формат файла для хранения трёхмерных моделей объектов
RISC(Reduced Instruction Set Computer) – архитектура процессора
UTF-16(Unicode Transformation Format) – способов кодирования символов
ASCII(American Standard Code for Information Interchange) – таблица символов
ANSI(American national standards institute) – американский национальный институт стандартов
ISO(International Organization for Standardization) – международная организация по стандартизации
API(Application Programming Interface) – интерфейс прикладного программирования

ПРИЛОЖЕНИЕ В

Листинги

Листинг В.1 – Демонстрация класса Screen

```
enum class TypeScreen {
  list, grid, input, unknown
};

class Screen {
private:
  Label label;
  TypeScreen typeScreen;
  std::map<std::string, Screen*> childs;
  std::map<std::string, State> states;

public:
  Screen();
  Screen(const Screen &screen);

  const Label& getLabel() const;
  void setLabel(const Label &label);

  TypeScreen getTypeScreen() const;
  void setTypeScreen(const std::string &typeScreen);
  void setTypeScreen(const TypeScreen &typeScreen);

  const std::map<std::string, Screen*>& getChilds() const;
  void setChilds(const std::map<std::string,
                 Screen*> &childs);

  const std::map<std::string, State>& getStates() const;
  void setStates(const std::map<std::string,
                 State> &states);

  TypeScreen stringToTypeScreen(
                 const std::string &typeScreenString);
  std::string typeScreenToString();

  void freeChilds();

  ~Screen();
};
```

Листинг В.2 – Демонстрация класса Button

```
enum class TypeButton {
  back, ok, go_to, unknown
};

class Button {
```

```
private:
  TypeButton typeButton;
  std::string screen;
  Label label;

public:
  Button();

  TypeButton getTypeButton() const;
  void setTypeButton(const std::string &typeButtonString);

  const Label& getLabel() const;
  void setLabel(const Label &label);

  const std::string& getScreen() const;
  void setScreen(const std::string &screen);

  TypeButton stringToTypeButton(
                    const std::string &typeButtonString);
  std::string typeButtonToString();

  std::string toString();

  ~Button();
};
```

Листинг В.3 – Демонстрация класса Icon

```
class Icon {
private:
  std::string imageFile;
  int x;
  int y;

public:
  Icon();

  const std::string& getImageFile() const;
  void setImageFile(const std::string &imageFile);

  int getX() const;
  void setX(int x);

  int getY() const;
  void setY(int y);

  ~Icon();
};
```

Листинг В.4 – Демонстрация перечисления Key

```
enum class Key {
  f1, f2, f3, f4, unknown
};
```

Листинг В.5 – Демонстрация класса Label

```
class Label {
private:
  std::string ru;
  std::string eng;

public:
  Label();

  const std::string& getEng() const;
  void setEng(const std::string &eng);

  const std::string& getRu() const;
  void setRu(const std::string &ru);

  std::string toString() const;

  ~Label();
};
```

Листинг В.6 – Демонстрация класса Root

```
class Root {
private:
  std::map<std::string, Icon> icons;
  std::map<Key, Button> buttons;

public:
  Root(/* args */);

  const std::map<std::string, Icon>& getIcons() const;
  void setIcons(const std::map<std::string, Icon> &icons);

  void setButtons(const std::map<Key, Button> &buttons);

  ~Root();
};
```

Листинг В.7 – Демонстрация класса State

```
class State {
private:
  std::map<Key, Button> buttons;

public:
  State(/* args */);

  const std::map<Key, Button>& getButtons() const;
  void setButtons(const std::map<Key, Button> &buttons);

  ~State();
};
```

Листинг В.8 – Демонстрация синтаксического анализа одного элемента блока screens

```
Screen MenuParser::parseScreen(YAML::Node node) {
  Screen screen;
  YAML::const_iterator attribute = node.begin();

  Parser::match(LOCATION, attribute->first, "label");
  screen.setLabel(parseLabel(attribute->second));

  attribute++;
  Parser::match(LOCATION,
                attribute->first, "typeScreen");
  screen.setTypeScreen(attribute->second.as<string>());

  attribute++;
  Parser::match(LOCATION, attribute->first, "states");
  screen.setStates(parseListStates(attribute->second));

  return screen;
}

Label MenuParser::parseLabel(YAML::Node node) {
  Label label;
  YAML::const_iterator attribute = node.begin();

  Parser::match(LOCATION, attribute->first, "ru");
  label.setRu(attribute->second.as<string>());

  attribute++;
  Parser::match(LOCATION, attribute->first, "eng");
  label.setEng(attribute->second.as<string>());

  return label;
}

map<Key, Button> MenuParser::parseListButtons(
    YAML::Node node) {
  map<Key, Button> buttons;

  for (YAML::const_iterator iterator = node.begin();
       iterator != node.end();
       ++iterator) {

    string keyName = iterator->first.as<string>();
    Key key = stringToKey(keyName);

    if (buttons.find(key) == buttons.end()) {
      string buttonName = iterator->second.as<string>();
      auto button = allButtons.find(buttonName);

      if (button != allButtons.end()) {
        buttons[key] = button->second;
```

```
      } else {
        throw Parser::ETokenNotExist(LOCATION,
            iterator->first.Mark().line,
            iterator->first.Mark().column, buttonName);
      }

    } else {
      throw Parser::ETokenExist(LOCATION,
          iterator->first.Mark().line,
          iterator->first.Mark().column, "key", keyName);
    }
  }

  return buttons;
}
```

Листинг В.9 – Демонстрация класса MenuWindow

```
#ifndef MENUWINDOW_H
#define MENUWINDOW_H

#include <QMainWindow>
#include <QMap>
#include <QKeyEvent>
#include <QPushButton>

#include "Screen.h"
#include "Softkey.h"

namespace Ui {
class MenuWindow;
}

class MenuWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MenuWindow(Screen *screen = nullptr,
                        QWidget *parent = 0);
    ~MenuWindow();

    void keyPressEvent(QKeyEvent *k);
    void keyReleaseEvent(QKeyEvent *k);

public slots:
    void showWindow();

private slots:
    void onSoftkeyClicked(int indexSoftkey);

    void closeWindow();

    void actionOk();
```

```
    void actionBack();

signals:
    void back();

private:
    void navigationMoveUp();
    void navigationMoveDown();
    void navigate(int newRow);

    void setColorActiveButton();
    void setColorDefaultButton();

    void createSoftkeys();

    void initSizeWindow();
    void initChildWindow();
    void initStates();
    void initSoftkeys(
                    const std::map<Key,Button> &buttons);
    void initSoftkey(int id, const Button &button);

    int getIdSoftkey(Key key);
    Softkey *getSoftkey(int i);
    void initActionSoftkey(Softkey *pushButton,
                           TypeButton typeButton);

    Ui::MenuWindow *ui;

    QMap<QString, MenuWindow *> childs;

    Screen *screen;

    int currentRow;

    QString buttonDefaultBackgroundColor;
    QString buttonActiveBackgroundColor;

    bool isPushed;
};

#endif // MENUWINDOW_H
```

Листинг В.10 – Демонстрация тестовой конфигурации меню

```
screensTree:
   menu:
      status:
         network: null
         version: null
      services:
         account1: null
         account2: null
      settings:
```

```
         phone: null
         system: null
      story:
         missed: null
         dialed: null

buttons:
   buttonBack:
      typeButton: back
      label:
         ru: Назад
         eng: Back
   buttonOk:
      typeButton: ok
      label:
         ru: Ок
         eng: Ok
   buttonGotoMenu:
      typeButton: goto
      screen: menu

screens:
   menu:
      label:
         ru: Меню
         eng: Menu
      typeScreen: list
      states:
         default:
            buttons:
               f1: buttonBack
               f2: buttonOk
               f3: buttonOk
         default2:
            buttons:
               f1: buttonBack
               f3: buttonOk
   status:
      label:
         ru: Статус
         eng: Status
      typeScreen: list
      states:
         default:
            buttons:
               f1: buttonBack
               f2: buttonBack
               f3: buttonOk
   services:
      label:
         ru: Услуги
         eng: Services
      typeScreen: list
      states:
```

```
      default:
         buttons:
            f1: buttonBack
            f2: buttonBack
            f3: buttonOk
settings:
   label:
      ru: Настройки
      eng: Settings
   typeScreen: list
   states:
      default:
         buttons:
            f1: buttonBack
            f3: buttonOk
story:
   label:
      ru: История
      eng: Story
   typeScreen: list
   states:
      default:
         buttons:
            f1: buttonBack
            f3: buttonOk
account1:
   label:
      ru: "Аккаунт 1"
      eng: "Account 1"
   typeScreen: list
   states:
      default:
         buttons:
            f1: buttonBack
            f3: buttonOk
account2:
   label:
      ru: "Аккаунт 2"
      eng: "Account 2"
   typeScreen: list
   states:
      default:
         buttons:
            f1: buttonBack
            f3: buttonOk
version:
   label:
      ru: "Версия ПО"
      eng: "Software version"
   typeScreen: list
   states:
      default:
         buttons:
            f1: buttonBack
```

```
            f3: buttonOk
   network:
      label:
         ru: "Сеть"
         eng: "Network"
      typeScreen: list
      states:
         default:
            buttons:
               f1: buttonBack
               f3: buttonOk
   phone:
      label:
         ru: "Телефон"
         eng: "Phone"
      typeScreen: list
      states:
         default:
            buttons:
               f1: buttonBack
               f3: buttonOk
   system:
      label:
         ru: "Система"
         eng: "System"
      typeScreen: list
      states:
         default:
            buttons:
               f1: buttonBack
               f3: buttonOk
   missed:
      label:
         ru: "Пропущенные"
         eng: "Missed"
      typeScreen: list
      states:
         default:
            buttons:
               f1: buttonBack
               f3: buttonOk
   dialed:
      label:
         ru: "Набранные"
         eng: "Dialed"
      typeScreen: list
      states:
         default:
            buttons:
               f1: buttonBack
               f3: buttonOk

root:
   typeScreen: null
```

```
icons:
   missedCall:
      file: missedCall.bitmap
      position:
         x: 1
         y: 1
   message:
      file: message.bitmap
      position:
         x: 2
         y: 1
buttons:
   f1: buttonBack
   f3: buttonGotoMenu
```