Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Курсовой проект

по дисциплине: «Проектирование операционных систем»

Тема: «Разработка демона звонков»

Работу выполнил студент

13541/4 Зорин А. Г.

Преподаватель

_____ Душутина Е.В.

Санкт-Петербург
2017

Оглавление

1 Цель работы

Целью данной работы является разработка демона звонков. Такой демон позволит получить информацию о том, что на sim-модуле был изменен звонок. Например, удаление или добавление звонка.

2 Описание задачи

Данная курсовая работа выполнена в раках проекта по разработке мобильного устройства на платформе Raspberry Pi Zero [1]. Данный проект включает в себя несколько задач:

- разработка аппаратной платформы мобильного устройства на основе Raspberry Pi Zero — подбор необходимых компонентов мобильного устройства (GSM модуль, динамик, микрофон, аккумулятор и т.д.) и их размещение на плате устройства
- установка и конфигурирование ОС для Raspberry Pi
- разработка стека драйверов для комплектующих
- разработка сервисного слоя (в виде демонов UNIX), который будет предоставлять необходимую информацию клиентским приложениям
- разработка мобильного оконного менеджера, который позволит запускать и отображать на экране графические пользовательские приложения
- разработка клиентских приложений (для осуществления звонков)

Исходя из приведенных выше пунктов, можно сказать, что целью данной работы является разработка демона, который, в зависимости от типа изменения звонка и его статуса, будет открывать соответствующее графическое приложение.

Создаваемый демон звонков должен выполнять следующие задачи:

- Запускаться при старте системы
- Активация sim-модуля для обеспечения возможности дальнейшей работы с ним
- Получение информации от модуля
- Отслеживание изменений на модуле

3 Теоретические сведения

3.1 D-Bus

D-Bus представляет из себя систему межпроцессорного взаимодействия, которая позволяет приложениям, находящимся в операционной системе (ОС), общаться между собой. D-Bus является частью проекта freedesktop.org [2]. Данная система обладает высокой скоростью работы, не зависит от рабочей среды и работает на POSIX-совместимых ОС.

D-Bus предоставляет несколько шин:

- Системная шина. Создается при старте демона D-Bus. С ее помощью происходит общение между различными демонами.
- Сессионная шина. Создается для пользователя, авторизовавшегося в системе. Для каждой сессионной шины запускается отдельная копия демона. Посредством этой копии общаются приложения, с которыми работает пользователь.

Каждое сообщение, передоваемое по шине, имеет своего отправителя. В том случае, когда сообщение не является широковещательным сигналом, оно имеет, в добавок к отправителю, своего получателя. Адреса отправителей и получаетлей, в контексте D-Bus, называются путями объектов по той причине, что каждое приложение состоит из набора объектов и сообщение происходит именно между этими объектами, а не между приложениями.

D-Bus также предусматривает концепцию сервисов. Сервис — уникальное местоположение приложения на шине. Приложение, при запуске, регистрирует один или несколько сервисов, которыми оно будет владеть до тех пор, пока самостоятельно не освободит. До этого момента никакое другое приложение, претендующее на тот же сервис, занять его не сможет. Именуются сервисы аналогично интерфейсам. После закрытия приложения ассоциированные сервисы также удаляются, а D-Bus посылает сигнал о том, что сервис закрыт.

Сервисы делают доступной ещё одну функцию — запуск необходимых приложений в случае поступления сообщений для них. Для этого должна быть включена автоактивация, а в конфигурации D-Bus за этим сервисом должно быть закреплено одно приложение.

После подключения к шине, приложение должно указать, какие сообщения оно желает получать, путём добавления масок совпадений (matchers). Маски представляют собой наборы правил для сообщений, которые будут доставляться приложению. Фильтрация может основываться на интерфейсах, путях объектов и методах.

Сообщения в D-Bus бывают четырёх видов: вызовы методов, результаты вызовов методов, сигналы (широковещательные сообщения) и ошибки.

В D-Bus у каждого объекта своё уникальное имя, которое выглядит как путь в файловой системе. Архитектура D-Bus показана с импользованием D-Bus интерфейса org.freedesktop.DBus.ObjectManager на рис. 1.

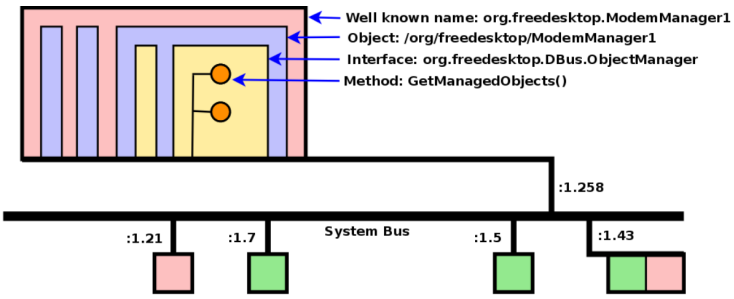

Рис. 1: Архитектура D-Bus

3.2 oFono

Для организации общения с используемым sim-модулем был использолван программный проект oFono. Данный проект является бесплатным и распространяется под лицензией GNU GPL v2 [3]. Проект oFono построен на стандарте 3GPP (3rg Generation Partnership Project) и использует D-Bus API для общения. Архитектура проекта oFono показана на рис. 2.

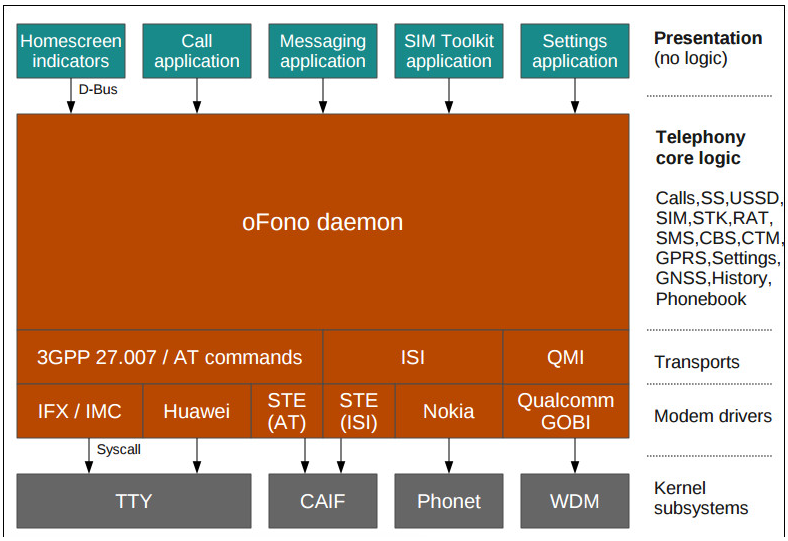

Рис. 2: Архитектура ofono

Проект oFono был анонсирован компаниями Intel и Nokia в мая 2009 года. Последняя версия 1.4 была представлена в августе 2016 года. Работа над данным проектом ведется до сих пор. Исходные коды oFono находятся в свободном доступе, в гит репозитории.

Программный стек oFono поддерживает различные модули, такие как:

- 2G/3G
- LTE
- CDMA(Code-division multiple access)
- GSM
- Bluetooth и т.д.

Общаение между oFono и sim-модулем будет производится через различные AT-команды. В свою очередь, разрабатываемый демон будет общаться с oFono посредством D-Bus.

4 Анализ способов общения через D-Bus

4.1 QtDbus

Qt — это кросплатформенный инструментарий разработки программного обеспечения (ПО) на языке программирования C++ [4]. Qt позваляет запускать различные приложения, написанные с его помощью, на различных ОС, без необходимости переписывать исходный код. Одна из основных отличительных особенностей Qt - использование Meta Object Compiler (MOC). MOC - система предварительной обработки исходного кода. Она позволяет во много раз увеличить мощь библиотек вводя таки понятия, как слоты и сигналы.

Qt позволяет создавать собственные плагины и размещать их непосредственно в панели визуального редактора. Также существует возможность расширения привычной функциональности виджетов, связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна.

Одним из весомых преимуществ проекта Qt является наличие качественной документации. Статьи документации снабжены большим количеством примеров. Исходный код самой библиотеки хорошо форматирован, подробно комментирован и легко читается, что также упрощает изучение Qt.

Помимо «чистого» Qt, для реализации графического интерфейса можно использовать связку Qt + QML. QML представляет из себя декларативный язык программирования, основанный на JavaScript, предназначенный для создания дизайна приложений. QML документ выглядит как дерево элеиентов. Сам QML элемент представляет из себя совокупность блоков:

- Графических
  - Rectangle
  - Image и т.д.
- Поведенческих
  - State
  - Transition
  - Animation и т.д.

Qt обеспечивает возможностью работать с D-Bus через собственный модуль, который называется QtDBus. Данный модуль полностью инкапсулирует низкоуровневую концепцию обмена сообщений в более простую - объектно ориентированную модель. Для работы с данном модулем существует огромное количество классов, каждый из которых хорошо задокументирован.

4.2 GLib

GLib — набор из низкоуровневых системных библиотек, написанных на языке программирования C и разрабатываемых, в основном, GNOME [5]. Исходные коды GLib были отделены от GTK+ и могут быть использованы ПО отличным от GNOME. GLib распространяется под лицензией GNU GPL и исходные коды находятся на github.

GLib предоставляет такие структуры данных, как:

- Одно- и двусвязные списки
- Хэш-таблицы
- Динамические массивы
- Динамическиие строки
- Сбалансированные двоичные деревья и т.д.

Основные моменты D-Bus, в библиотеках GLib реализованы двумя, приблизительно идентичными, путями: dbus-glib и GDBus. В обеих реализациях прокси классы и вызовы методов D-Bus реализованы в виде объектов. Однако, существуют некоторые различия:

- dbus-glib использует реализацию libdbus. GDBus, в отличии от dbus-glib использует GIO потоки в качестве транспортного уровня и имеет собственную реализацию для настройки подключения и аутентификации D-Bus. Помимо использования потоков в качестве транспорта, GDBus позволяет избежать некоторых проблем связанных с многопоточными функциями.
- dbus-glib использует систему типа GObject для аргументов методов, возвращаемых значений, а также - механизма сигналов. GDbus, в свою очередь, пологается на систему типа GVariant, которая разработана для соответствия типам D-Bus.
- dbus-glib моделирует только интерфейсы D-Bus и не предоставляет никаких типов для объектов. GDBus моделирует интерфейсы D-Bus (через типы GDBusInterface, GDBusProxy и GDBusInterfaceSkeleton) и объекты (через типы GDBusObject, GDBusObjectSkeleton и GDBusObjectProxy).
- GDBus предоставляет встроенную поддержку для org.freedesktop.DBus.Properties через тип GDBusProxy и интерфейсы org.freedesktop.DBus.ObjectManager D-Bus, dbus-glib - нет.
- Типичный способ экспорта объекта с помощью dbus-glib включает создание кода из данных XML с использованием dbus-binding-tool. GDBus предоставляет аналогичный инструмент под названием gdbus-codegen, который также может генерировать Docbook D-Bus документацию интерфейсов.
- dbus-glib не предоставляет каких-либо удобных API для просмотра имен шин, GDBus предоставляет семейство удобных функций g_bus_own_name и g_bus_watch_name.
- GDBus предоставляет API для анализа, генерации и работы с XML, dbus-glib - нет. [2]

4.3 Сравнение QtDBus и GLib

В предыдущих секциях были рассмотренны такие способы общения по D-Bus, как QtDBus, GDBus, dbus-glib. Если проводить аналогию между рассмотренными библиотеками, то можно сделать вывод о том, что QtDBus и GDBus очень похожи между собой, а различия между GDBus и dbus-glib были рассмотренны в секции выше. Таким образом, в качестве общего заключения можно сделать вывод о том, что dbus-glib является более низкоуровневой библиотекой для общения с D-Bus, чем QtDBus или GDBus.

5 Выполнение работы

5.1 Описание тестового стенда

Для выполнения работы использовалось два тестовых стенда:

- Платформа Raspberry Pi 1 с ОС ArchLinux и модуль SIM-808
- Компьютер с ОС Ubuntu 16.04

В реальном стенде будет использована плата Raspberry Pi Zero и сим модуль SIM-808L. А благодаря тому, что модуль SIM-808 полностью совместим с модулем SIM-808L, то демон, который был разработан с использованием одного модуля будет совместим с другим модулем. Так как демон создавался независимым от реализации платформы, то не имеет особого значения, на какой ОС вести разработку.

5.2 Выбор библиотеки

Как было рассмотренно, для реализации общения по D-Bus существует несколько различных библиотек. Исходя из проведенного анализа, была выбрана библиотека dbus-glib по той причине, что используемое устройствоя является маломощным, а так как у выбранной библиотеки малый уровень абстракции, то данный выбор позволяет нам увеличить производительность.

5.3 Разработка демона

Исходный код демона приведен в листинге 4. В основной функции приложения main производятся следующие действия:

- Запись в лог информации о старте демона
- Подключение к системной шине D-Bus
- Получение списка доступных модемов
- Выбор модема
- Попытка активации модема
- Получение оператора

Парсинг полученного ответа и запись в лог-файл осуществляется с помощью заголовочного файла, код которого приведен в листинге 5. В данном заголовочном файле реализована структура, которая соответствует большинству ответов. В структуре два поля:

- Строка, соответствующая пути оъъекта (например, /sim900_0 — путь используемого модуля).
- Словарь, который соответствует свойствам, где ключ - свойство, а значение - значение свойства.

Также, в демоне реализовано подключение к сигналу и его обработка. Далее будет подробнее рассмотрен этап подключения к сигналу.

- dbus_error_init производит инициализацию структуры ошибки D-Bus.
- dbus_bus_get подключение к системной шине D-Bus.

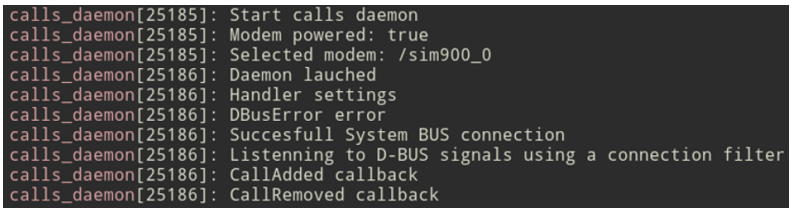

Рис. 3: Пример исправной работы демона

- dbus_error_is_set проверка на возникновение ошибки D-Bus.
- dbus_connection_setup_with_g_main устанавливает функции наблюдения и тайм-аута DBusConnection для интеграции соединения с основным циклом.
- dbus_bus_add_match добавляет правило соответствия для сообщений, проходящих через шину.
- dbus_connection_add_filter дообавляет фильтрпацию сообщений. Фильтры — это обработчики сообщений, которые вызываются для обработки всех входящих сообщений, относящихся к зарегистрированному объекту.

Помимо файла с исходным кодом и заголовочного файла, в структуру проекта входит CMakeLists — файл, предназначенный для сборки проекта. Содержание данного файла приведено в листинге 6.

Пример исправной работы демона можно увидеть на рис. 3

6 Дополнительная работа

В ходе проекта, помимо создания демона звонков, была создана графическая оболочка для приложения телефона. В качестве платформы для создания графических приложений рассматривались две, одна из которых была описана выше, — Qt и GTK+. В конечном итоге, была выбрана платформа Qt из-за того, что графика, написанная на Qt, на конечной платформе выглядит приятнее, чем графика, написанная на GTK+.

Для создания графической оболочки была использована связка QT + QML, а основная логика приложения — C++. Самым первым этапом создания основной логики приложения является создание самого приложения QApplication (Листинг 1).

Листинг 1: Создание приложения

```
1 int main(int argc, char *argv[])
2 {
3         QApplication a(argc, argv);
4         MainWindow w;
5         return a.exec();
6 }
```

Класс MainWindow представляет из себя основную логику графического окна. Реализация данного класса приведена в листингах 7 — 8. В данном классе выполняются следующия действия:

- Подключение к созданному приложению графической оболочки
- Получение активированного модема
- Совершение вызова
- Завершение вызова

Для подключения графической состовляющей и различных изображений используются ресурсы. Использование ресурсов позволяет облегчить сборку приложения, тем самым, платформа нагружается немногим меньше, а время сборки уменьшается. Файл ресурсов выглядит следующим образом (листинг 2).

Листинг 2: Файл ресурсов

```
<RCC>
        <qresource prefix="/">
                <file>qml/main.qml</file>
                <file>qml/dialing.qml</file>
                <file>qml/core/Button.qml</file>
                <file>qml/call.qml</file>
                <file>pics/dial.png</file>
                <file>pics/erase.png</file>
                <file>pics/back.png</file>
                <file>pics/hang.png</file>
        </qresource>
</RCC>
```

Окна с графической реализацией имеют расширение .qml. Главное окно и реализация класса Button приведены в листингах 9—10. Так как вместо стандартного класса Button был использован свой класс, то приложению необходимо сообщить о том, что когда программист пытается реализовать класс Button, вместо стандартного, нужно реализовывать созданный. Для этого нужна всего одна строчка в файле qmldir (Листинг 3).

Листинг 3: Подключение класса Button

```
Button Button.qml
```

Для сборки проекта написанного на Qt принято использовать утилиту qmake. Для ее использования создается файл с расширением .pro и в него прописываются все зависимости. Используемый файл phone.pro показан в листинге 11.

Таким образом, было написано графическое приложение, позволяющее:

- Набирать номер
- Совершать звонок
- Отображать звонок
- Сбрасывать звонок

Пример приложения приведен на рисунке 4.

7 Выводы

В ходе работы были проанализированы и сравнены различные способы работы с D-Bus. В результате анализа, для организации работы была выбрана библиотека dbus-glib. Она позволяет облегчить работу платформы и увеличить производительность, так как являются низкоуровневой и имеет низкий уровень абстракции. В данной работе был реализован демон звонков. Данный демон разрабатывался в рамках проекта по разработке смартфона. Разработанный демон позволяет отслеживать различные изменения голосовых вызовов, произошедших на используемом модуле SIM-808. Помимо демона звонков было разработано графическое приложение телефона. Созданный демон взаимодействует с данным приложением в зависимости от статуса звонка.

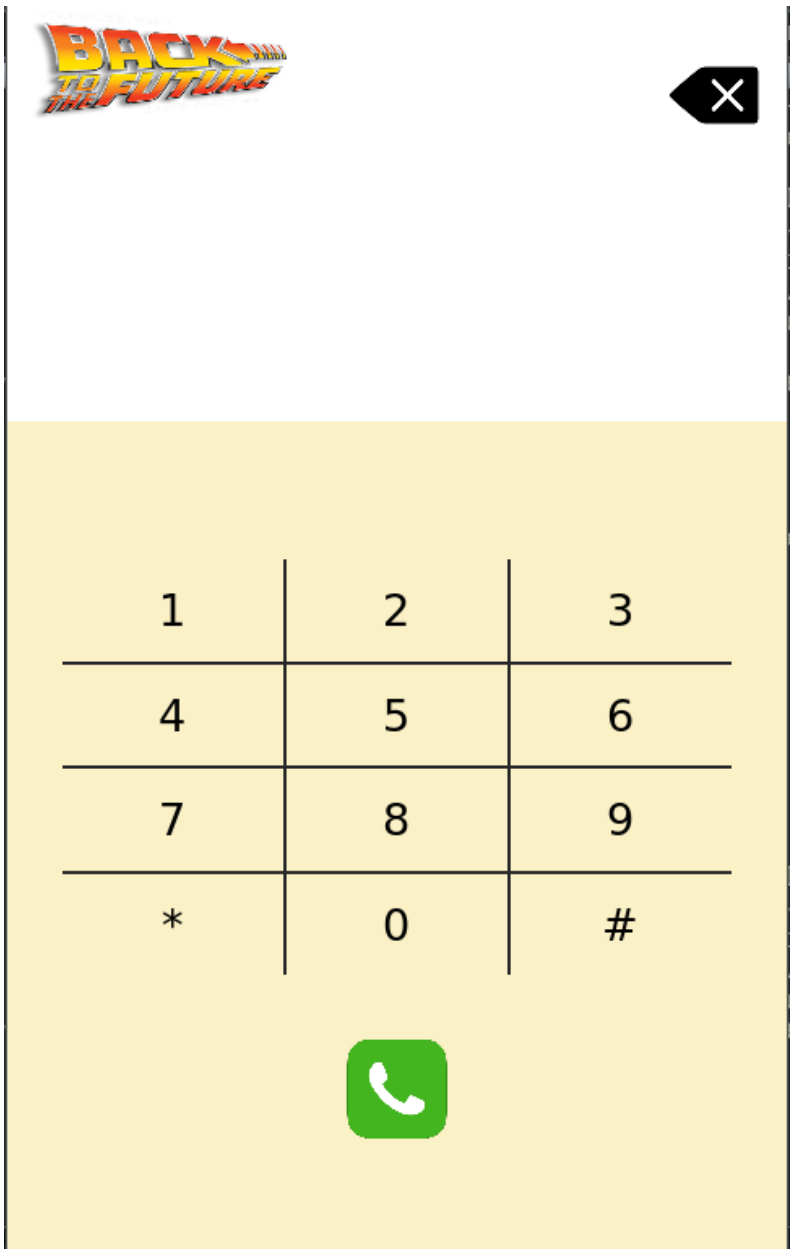

Рис. 4: Пример работы графического приложения

Литература

[1] Официальный сайт Raspberry Pi. — https://www.raspberrypi.org/products/pi-zero/. — 2017. — Accessed: 2017-06-1.

[2] Главная страница проекта freedesktop.org. — https://www.freedesktop.org/. — 2017. — Accessed: 2017-05-28.

[3] Официальный сайт oFono. — https://01.org/ofono. — 2017. — Accessed: 2017-05-1.

[4] Qt главная страница. — https://www.qt.io. — 2017. — Accessed: 2017-05-28.

[5] GNOME главная страница. — https://www.gnome.org/. — 2017. — Accessed: 2017-05-28.

8 Прилагаемые материалы

Список прилагаемых материалов:

- Техническое задание.docx — техническое задание на проект;
- progManual.pdf — руководство системного программиста;
- sources.pdf — текст программы;
- spec.pdf — описание программы;
- tests.pdf — программа и методика испытаний;
- userManual.pdf — руководство пользователя.

Листинги

Листинг 4: Файл main.cpp

```
1  #include <fstream>
2  #include <QString>
3  #include <QtDBus>
4  #include <glib.h>
5  #include <dbus/dbus.h>
6  #include <dbus/dbus-glib-lowlevel.h>
7  #include "Struct.h"
8
9  #define INFO 0
10 #define ERROR -1
11
12 bool isAnswerValid(QDBusMessage);
13 int callsMonitor();
14 QVariant isModemEnabled = "false";
15 void writeLog(const char*, int);
16 int setupHandler();
17 DBusHandlerResult call_added_callback(DBusConnection*, DBusMessage*, void *);
18
19 int main() {
20
21     writeLog("Start calls daemon", INFO);
22     QDBusConnection bus = QDBusConnection::systemBus();
23
24     if(!bus.isConnected())
25         exit(1);
26
27     QDBusInterface dbus_iface("org.ofono", "/", "org.ofono.Manager", bus);
28     QDBusMessage modem = dbus_iface.call("GetModems");
29
30     if(!isAnswerValid(modem))
31         exit(1);
32
33     const QDBusArgument &dbusArgs = modem.arguments().first().value<QDBusArgument
       >();
34     std::vector<Answer_struct> answers = getStructAnswer(dbusArgs);
35     QString selected_modem;
36
37     if(answers.size() == 0){
38         writeLog("Answer is NULL", ERROR);
39         exit(1);
40     }
41
42
43     if(answers.size() == 1) {
44         selected_modem = answers[0].name;
45         isModemEnabled = answers[0].porp_map["Powered"];
46         writeLog("Modem powered: " + isModemEnabled.toString().toLatin1(), INFO);
47     }else
48         for(Answer_struct modem : answers)
49             if(modem.name.contains("sim900")){
50                 selected_modem = modem.name;
51                 isModemEnabled = modem.porp_map["Powered"].toBool();
```

```
52                  writeLog("Modem powered: " + isModemEnabled.toString().toLatin1(),
    INFO);
53              }
54
55      if(selected_modem.isNull() || selected_modem.isEmpty()) {
56          writeLog("No modem was selected", ERROR);
57          exit(1);
58      }
59
60
61      writeLog("Selected modem: " + selected_modem.toLatin1(), INFO);
62
63      if(isModemEnabled == "false") {
64          QDBusInterface modem_iface("org.ofono", "/", "org.ofono.Modem", bus);
65          QList<QVariant> argumentList;
66          auto reply = modem_iface.call(QString("SetProperty"), QVariant::fromValue(
    QString("Powered")),
67                                                  QVariant::fromValue(QDBusVariant(true)
    ));
68
69          if(!isAnswerValid(reply)){
70              writeLog(reply.errorMessage().toLatin1(), ERROR);
71              exit(1);
72          }
73          writeLog("Modem succesffuly enabled", INFO);
74
75      }
76
77      QDBusInterface network_iface("org.ofono", selected_modem, "org.ofono.
    NetworkRegistration", bus);
78      QList<QVariant> argumentList;
79      QDBusPendingReply<> operators= network_iface
80              .asyncCallWithArgumentList(QStringLiteral("GetOperators"),
    argumentList);
81      auto reply = operators.argumentAt(0).value<QDBusArgument>();
82      answers = getStructAnswer(reply);
83      QString networkOperator;
84      for(Answer_struct answer : answers)
85          networkOperator = answer.porp_map["Name"].toString();
86
87      std::ofstream operName;
88      operName.open("~//operator.txt");
89      operName << networkOperator.toStdString();
90      operName.close();
91
92      qDebug() << "Operator: " << networkOperator;
93
94      int pid = fork();
95      if(pid == -1) {
96          writeLog("Daemon launching failed.\n", ERROR);
97          return -1;
98      }
99      else if(!pid){
100         writeLog("Daemon lauched", INFO);
101         umask(0);
102         setsid();
```

```
103         chdir("/");
104
105         close(STDIN_FILENO);
106         close(STDOUT_FILENO);
107         close(STDERR_FILENO);
108
109         return callsMonitor();
110
111     } else
112         return 0;
113 }
114
115 bool isAnswerValid(QDBusMessage msg) {
116     if(QDBusMessage::ErrorMessage == msg.type()){
117         writeLog(msg.errorMessage().toLatin1(), ERROR);
118         return false;
119     }
120     return true;
121 }
122
123 int callsMonitor() {
124     QDBusInterface calls_inface("org.ofono", "/", "org.ofono.Manager",
    QDBusConnection::systemBus());
125     QDBusMessage modem = calls_inface.call("GetCalls");
126     setupHandler();
127     writeLog("Daemon ends", ERROR);
128 }
129
130 DBusHandlerResult call_added_callback(DBusConnection *con, DBusMessage *msg, void
    *user_data){
131     if(dbus_message_is_signal(msg, "org.ofono.VoiceCallManager", "CallAdded"))
132         writeLog("CallAdded callback", INFO);
133
134
135     if(dbus_message_is_signal(msg, "org.ofono.VoiceCallManager", "CallRemoved"))
136         writeLog("CallRemoved callback", INFO);
137
138     return DBUS_HANDLER_RESULT_NOT_YET_HANDLED;
139 }
140
141 int setupHandler() {
142     writeLog("Handler settings", INFO);
143     GMainLoop *loop = g_main_loop_new(NULL, FALSE);
144     DBusError error;
145     writeLog("DBusError error", INFO);
146     dbus_error_init(&error);
147     DBusConnection *conn = dbus_bus_get(DBUS_BUS_SYSTEM, &error);
148
149     if(dbus_error_is_set(&error)){
150         writeLog(strcat("Cannot get System BUS connection: ", error.message),
    ERROR);
151         dbus_error_free(&error);
152         return EXIT_FAILURE;
153     }
154     writeLog("Succesfull System BUS connection", INFO);
155     dbus_connection_setup_with_g_main(conn, NULL);
156
```

```
157     char *rule = "type='signal',␣interface='org.ofono.VoiceCallManager'";
158     dbus_bus_add_match(conn, rule, &error);
159
160     if(dbus_error_is_set(&error)){
161         writeLog(strcat("Cannot␣add␣D-BUS␣match␣rule,␣cause:␣", error.message),
        ERROR);
162         dbus_error_free(&error);
163         return EXIT_FAILURE;
164     }
165
166     Answer_struct callAddedStruct;
167     writeLog("Listenning␣to␣D-BUS␣signals␣using␣a␣connection␣filter", INFO);
168     dbus_connection_add_filter(conn, call_added_callback, &callAddedStruct, NULL);
169
170     g_main_loop_run(loop);
171
172     return EXIT_SUCCESS;
173 }
```

Листинг 5: Файл Struct.h

```
1  #ifndef DAEMON_STRUCT_H
2  #define DAEMON_STRUCT_H
3
4  #include <QMetaType>
5  #include <QString>
6  #include <QtDBus>
7  #include <zconf.h>
8  #include <sys/stat.h>
9  #include <syslog.h>
10
11 #define INFO 0
12 #define ERROR -1
13
14 struct Answer_struct{
15     QString name;
16     QMap<QString, QVariant> porp_map;
17 };
18 Q_DECLARE_METATYPE(Answer_struct)
19
20 static std::vector<Answer_struct> getStructAnswer(const QDBusArgument &dbusArgs) {
21     QString selected_modem;
22     Answer_struct answer_struct;
23     std::vector<Answer_struct> answers;
24     dbusArgs.beginArray();
25     while (!dbusArgs.atEnd()) {
26         dbusArgs.beginStructure();
27         if (dbusArgs.currentType() == 0)
28             dbusArgs >> answer_struct.name;
29         if (dbusArgs.currentType() == 4)
30             dbusArgs >> answer_struct.porp_map;
31         dbusArgs.endStructure();
32         answers.push_back(answer_struct);
33     }
34     dbusArgs.endArray();
35
36     return answers;
```

```
}

static void writeLog(const char* message, int status) {
    openlog("calls_daemon", LOG_CONS | LOG_PID | LOG_NDELAY, LOG_LOCAL1);

    switch(status){
        case ERROR:
            syslog(LOG_ERR, message);
            break;
        case INFO:
            syslog(LOG_INFO, message);
            break;
        default:
            syslog(LOG_ALERT, message);
            break;
    }

    closelog();
}

#endif //DAEMON_STRUCT_H
```

Листинг 6: Файл сборки CMakeLists.txt

```
cmake_minimum_required(VERSION 3.7)
project(daemon)

set(CMAKE_CXX_STANDARD 11)

find_package(PkgConfig)
find_package (Qt5 CONFIG REQUIRED DBus)
find_package(PkgConfig)

pkg_check_modules(GLIB REQUIRED glib-2.0)

include_directories(${GLIB_INCLUDE_DIRS})
include_directories(/usr/include/dbus-1.0/)
include_directories(/usr/lib/x86_64-linux-gnu/dbus-1.0/include)

set (LIBS dbus-1 dbus-glib-1)
set(SOURCE_FILES main.cpp Struct.h)

add_executable(calls_daemon ${SOURCE_FILES})

target_link_libraries(calls_daemon Qt5::DBus ${DBUS_LIBRARIES} ${GLIB_LIBRARIES} ${LIBS})
```

Листинг 7: Файл mainwindow.cpp

```
#include "mainwindow.h"
#include <QQmlComponent>
#include <QQuickItem>

MainWindow::MainWindow(QObject *parent)
    : QQmlApplicationEngine(parent)
{
        load(QUrl("qrc:///qml/main.qml"));
```

```
 9           rootContext()->setContextProperty("window", this);
10           if(!bus.isConnected())
11                   exit(1);
12
13      GetModem();
14  }
15
16  MainWindow::~MainWindow() {}
17
18  std::vector<Answer_struct> getStructAnswer(const QDBusArgument &dbusArgs) {
19      QString selected_modem;
20      Answer_struct answer_struct;
21      std::vector<Answer_struct> answers;
22      dbusArgs.beginArray();
23      while (!dbusArgs.atEnd()) {
24          dbusArgs.beginStructure();
25          if (dbusArgs.currentType() == 0)
26              dbusArgs >> answer_struct.name;
27          if (dbusArgs.currentType() == 4)
28              dbusArgs >> answer_struct.porp_map;
29          dbusArgs.endStructure();
30          answers.push_back(answer_struct);
31      }
32      dbusArgs.endArray();
33
34      return answers;
35  }
36
37  void MainWindow::isAnswerValid(QDBusMessage msg)
38  {
39      if(QDBusMessage::ErrorMessage == msg.type()){
40          qDebug() << msg.errorMessage();
41          exit(1);
42      }
43  }
44
45  void MainWindow::GetModem() {
46      //QDBusConnection bus = QDBusConnection::systemBus();
47
48      if(!bus.isConnected())
49          exit(1);
50
51      QDBusInterface dbus_iface("org.ofono", "/", "org.ofono.Manager", bus);
52      QDBusMessage modem = dbus_iface.call("GetModems");
53
54      isAnswerValid(modem);
55
56      const QDBusArgument &dbusArgs = modem.arguments().first().value<QDBusArgument
        >();
57      std::vector<Answer_struct> answers = getStructAnswer(dbusArgs);
58
59      if(answers.size() == 0)
60          exit(1);
61
62      if(answers.size() == 1)
63          selected_modem = answers[0].name;
64      else
```

```
65          for(Answer_struct modem : answers)
66              if(modem.name.contains("sim900"))
67                  selected_modem = modem.name;
68
69      if(selected_modem.isNull() || selected_modem.isEmpty())
70          exit(1);
71  }
72
73  void MainWindow::dialNumber(QString call_number){
74          if(call_number.isEmpty() || call_number.isNull())
75                  return;
76
77      load(QUrl("qrc:///qml/call.qml"));
78
79      dialingWindow = this->rootObjects().at(1);
80      QObject* object = dialingWindow->findChild<QObject*>("call_number");
81      if(object)
82          object->setProperty("text", call_number);
83
84      QDBusInterface dbus_iface("org.ofono", selected_modem, "org.ofono.
        VoiceCallManager", bus);
85      auto reply = dbus_iface.call("Dial", QVariant::fromValue(QString(call_number))
        , QVariant::fromValue(QString("")));
86      isAnswerValid(reply);
87      start = std::clock();
88      getTime();
89  }
90
91  void MainWindow::getTime(){
92      //double duration = (std::clock - start) / (double) CLOCKS_PER_SEC;
93      QObject* object = dialingWindow->findChild<QObject*>("call_timer");
94      if(object)
95          object->setProperty("text", "time");//duration);
96  }
97
98  void MainWindow::hangUp(){
99      QDBusInterface dbus_iface("org.ofono", selected_modem, "org.ofono.
        VoiceCallManager", bus);
100     auto reply = dbus_iface.call("HangupAll");
101     isAnswerValid(reply);
102
103     exit(0);
104     //this->rootObjects().removeAt(1);
105 }
```

Листинг 8: Файл mainwindow.h

```
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QtCore/QUrl>
5  #include <QtQml/QQmlApplicationEngine>
6  #include <QQmlContext>
7  #include <QApplication>
8  #include <QString>
9  #include <QtDBus>
10 #include <iostream>
```

```
11 #include <ctime>
12
13 struct Answer_struct{
14     QString name;
15     QMap<QString, QVariant> porp_map;
16 };
17 Q_DECLARE_METATYPE(Answer_struct)
18
19 class MainWindow : public QQmlApplicationEngine{
20     Q_OBJECT
21 public:
22     MainWindow(QObject *parent = 0);
23     ~MainWindow();
24     void GetModem();
25     void isAnswerValid(QDBusMessage msg);
26     Q_INVOKABLE void dialNumber(QString number);
27     Q_INVOKABLE void hangUp();
28     Q_INVOKABLE void getTime();
29
30 private:
31     QDBusConnection bus = QDBusConnection::systemBus();
32     QString selected_modem;
33         QString dialedNumber;
34     QObject*  dialingWindow;
35     std::clock_t start;
36 };
37 #endif // MAINWINDOW_H
```

Листинг 9: Главное окно графического приложения

```
1  import QtQuick 2.3
2  import QtQuick.Window 2.2
3  import QtQuick.Controls 1.4
4  import "core"
5
6  Window{
7          id: phone
8          height: Screen.height
9          maximumHeight: Screen.height
10         minimumHeight: Screen.height
11     width:  480
12     minimumWidth: 480
13     maximumWidth: 480
14         title: "Phone"
15         visible: true
16
17         Image {
18                 id: buttonBack
19                 width: 170
20                 height: 70
21                 source: "qrc:///pics/back.png"
22
23             anchors{
24                         top: parent.top
25                         left: parent.left
26                         leftMargin: 10
27                 }
```

```
28
29                  MouseArea{
30                          anchors.fill: parent
31                          onClicked:Qt.quit()
32                  }
33          }
34
35          Image {
36                  id: buttonDelete
37                  width: 70
38                  height: 70
39              source: "qrc:///pics/erase.png"
40
41              anchors{
42                          top: parent.top
43                          topMargin: 20
44                          right: parent.right
45                          rightMargin: 10
46                  }
47
48              MouseArea{
49                  anchors.fill: parent
50                          onClicked: phoneNumber.text = phoneNumber.text.substr(0,
    phoneNumber.text.length - 1)
51                  }
52          }
53
54          Text {
55                  id: phoneNumber
56                  objectName: "number"
57                  text: ""
58                  font.pixelSize: 30
59                  wrapMode: Text.WrapAnywhere
60                  anchors {
61                          left: parent.left
62                          leftMargin: 10
63                          right: buttonDelete.left
64                          rightMargin: 10
65                          top: buttonBack.bottom
66                          topMargin: 20
67                  }
68          }
69
70          Rectangle {
71                  id: buttons
72                  width: parent.width
73                  height: 2 *  parent.height / 3
74                  color: "#fbf1c7"
75                  anchors{
76                          bottom: parent.bottom
77                  }
78
79          Rectangle{
80                          id: table
81                          width: parent.width - parent.width * 0.07 * 2
82                          height: parent.height / 2
83                          color: "#282828"
```

```
84
85                          anchors{
86                                  top: parent.top
87                                  left: parent.left
88                                  leftMargin: parent.width * 0.07
89                                  rightMargin: parent.width * 0.07
90                                  topMargin: height / 3
91                          }
92
93                          Grid {
94                                  id: numbers
95                                  spacing: 2
96                                  columns: 3
97                                  width: parent.width
98                                  height: parent.height
99                                  anchors{
100                                         horizontalCenter: parent.horizontalCenter
101                                         verticalCenter: parent.verticalCenter
102                                 }
103
104                                 Button {caption : "1"; spacing: parent.spacing;
    color: buttons.color}
105                                 Button {caption : "2"; spacing: parent.spacing;
    color: buttons.color}
106                                 Button {caption : "3"; spacing: parent.spacing;
    color: buttons.color}
107
108                                 Button {caption : "4"; spacing: parent.spacing;
    color: buttons.color}
109                                 Button {caption : "5"; spacing: parent.spacing;
    color: buttons.color}
110                                 Button {caption : "6"; spacing: parent.spacing;
    color: buttons.color}
111
112                                 Button {caption : "7"; spacing: parent.spacing;
    color: buttons.color}
113                                 Button {caption : "8"; spacing: parent.spacing;
    color: buttons.color}
114                                 Button {caption : "9"; spacing: parent.spacing;
    color: buttons.color}
115
116                                 Button {caption : "*"; spacing: parent.spacing;
    color: buttons.color}
117                                 Button {caption : "0"; spacing: parent.spacing;
    color: buttons.color}
118                                 Button {caption : "#"; spacing: parent.spacing;
    color: buttons.color}
119                         }
120                 }
121
122                 Image {
123                         id: buttonDial
124                         width: 70
125                         height: 70
126                         source: "qrc:///pics/dial.png"
127
128                         anchors {
```

```
                                top: table.bottom
                                topMargin: height / 2
                                horizontalCenter: parent.horizontalCenter
                                verticalCenter: parent.varticalCenter
                        }

                        MouseArea{
                                anchors.fill: parent
                                onClicked: dial(phoneNumber)
                        }
                }


        }

        function dial(object){
        window.dialNumber(object.text);
        }
}
```

Листинг 10: Реализация класса Button

```
import QtQuick 2.0
import QtQuick.Window 2.2
import QtQuick.Controls 1.4

Rectangle {

        property string caption: ""
        property int spacing

        id: button1
        width: parent.width / 3 - 2 * spacing / 3
        height: parent.height / 4 - 3 * spacing / 4

        Text {
                renderType: Text.NativeRendering
                horizontalAlignment: Text.AlignHCenter
                verticalAlignment: Text.AlignVCenter
                font.family: "SF"
                font.pointSize: 20
                text: caption
                width: parent.width
                height: parent.height
         }


        MouseArea{
                opacity:1
                anchors.fill: parent
                onClicked: phoneNumber.text += caption
                onDoubleClicked: {
                        if(caption != 0) return
                        phoneNumber.text = phoneNumber.text.substr(0, phoneNumber.
    text.length-1)
                        phoneNumber.text += "+"
                }
```

```
35          }
36  }
```

Листинг 11: Файл сборки phone.pro

```
1  QT += gui qml quick core dbus widgets
2  CONFIG += c++11 qtquickcompiler
3
4  HEADERS += mainwindow.h
5
6  SOURCES += main.cpp mainwindow.cpp
7
8  OTHER_FILES = main.qml dialing.qml call.qml button.qml
9
10 RESOURCES += res.qrc
11
12 target.path = $$[QT_INSTALL_EXAMPLES]
13 sources.files = $$SOURCES $$HEADERS $$RESOURCES phone.pro
14 sources.path = $$[QT_INSTALL_EXAMPLES]
15 INSTALLS += target sources
```