Исходный код программы

Зорин А.Г.

9 июня 2017 г.

Оглавление

1 Исходный код

1.1 Демон

Листинг 1: 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 2: Заголовочный 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;
37  }
38
39  static void writeLog(const char* message, int status) {
40      openlog("calls_daemon", LOG_CONS | LOG_PID | LOG_NDELAY, LOG_LOCAL1);
41
42      switch(status){
43          case ERROR:
44              syslog(LOG_ERR, message);
45              break;
46          case INFO:
47              syslog(LOG_INFO, message);
48              break;
49          default:
50              syslog(LOG_ALERT, message);
51              break;
52      }
53
54      closelog();
55  }
56
57  #endif //DAEMON_STRUCT_H
```

Листинг 3: Файл сборки CMake

```
1  cmake_minimum_required(VERSION 3.7)
2  project(daemon)
3
4  set(CMAKE_CXX_STANDARD 11)
5
6  find_package(PkgConfig)
7  find_package (Qt5 CONFIG REQUIRED DBus)
8  find_package(PkgConfig)
9
10 pkg_check_modules(GLIB REQUIRED glib-2.0)
11
12 include_directories(${GLIB_INCLUDE_DIRS})
13 include_directories(/usr/include/dbus-1.0/)
14 include_directories(/usr/lib/x86_64-linux-gnu/dbus-1.0/include)
15
16 set (LIBS dbus-1 dbus-glib-1)
17 set(SOURCE_FILES main.cpp Struct.h)
18
19 add_executable(calls_daemon ${SOURCE_FILES})
20
21 target_link_libraries(calls_daemon Qt5::DBus ${DBUS_LIBRARIES} ${GLIB_LIBRARIES} $
     {LIBS})
```

1.2 Графическое приложение

Листинг 4: Файл main.cpp

```
1  #include "mainwindow.h"
2
3  int main(int argc, char *argv[])
4  {
```

```
5       QApplication a(argc, argv);
6       MainWindow w;
7       return a.exec();
8   }
```

Листинг 5: Файл mainwindow.cpp

```
1   #include "mainwindow.h"
2   #include <QQmlComponent>
3   #include <QQuickItem>
4
5   MainWindow::MainWindow(QObject *parent)
6       : QQmlApplicationEngine(parent)
7   {
8           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 6: Заголовочный 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 7: Файл сборки qmake

```
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]
```

```
sources.files = $$SOURCES $$HEADERS $$RESOURCES phone.pro
sources.path = $$[QT_INSTALL_EXAMPLES]
INSTALLS += target sources
```

Листинг 8: Файл ресурсов res.qrc

```
<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>
```

Листинг 9: Файл main.qml

```
import QtQuick 2.3
import QtQuick.Window 2.2
import QtQuick.Controls 1.4
import "core"

Window{
        id: phone
        height: Screen.height
        maximumHeight: Screen.height
        minimumHeight: Screen.height
    width:  480
    minimumWidth: 480
    maximumWidth: 480
        title: "Phone"
        visible: true

        Image {
                id: buttonBack
                width: 170
                height: 70
                source: "qrc:///pics/back.png"

            anchors{
                        top: parent.top
                        left: parent.left
                        leftMargin: 10
                }

                MouseArea{
                        anchors.fill: parent
                        onClicked:Qt.quit()
                }
        }

        Image {
                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 {
129                                 top: table.bottom
130                                 topMargin: height / 2
131                                 horizontalCenter: parent.horizontalCenter
132                                 verticalCenter: parent.varticalCenter
133                         }
134
135                         MouseArea{
136                                 anchors.fill: parent
137                                 onClicked:dial(phoneNumber)
```

```
                        }
                }


        }

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

Листинг 10: Файл dialing.qml

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

Window{
        id: phone
        height: Screen.height
        maximumHeight: Screen.height
        minimumHeight: Screen.height
    width: 480
    minimumWidth: 480
    maximumWidth: 480
        title: "Calling"
    visible: true

        Rectangle{
                id: call_info
                color: "blue"
                border.color: "black"
                border.width: 5
                anchors{
                        top: parent.top
                }
        height: parent.height / 2
        width: parent.width

        Text {
            id: dialingText
            text: "Dialing:"
            color: "#fbf1c7"
            anchors{
                top: parent.top
                left: parent.left

                topMargin: parent.height / 3
                leftMargin: 20
            }
            renderType: Text.NativeRendering
            horizontalAlignment: Text.AlignHCenter
            verticalAlignment: Text.AlignVCenter
            font.family: "SF"
            font.pointSize: 20
        }

```

```
45          Text {
46              id: call_number
47              objectName: "call_number"
48              color: "#fbf1c7"
49              anchors{
50                  top: dialingText.bottom
51                  left: parent.left
52
53                  topMargin: dialingText.height
54                  leftMargin: 20
55              }
56              renderType: Text.NativeRendering
57              horizontalAlignment: Text.AlignHCenter
58              verticalAlignment: Text.AlignVCenter
59              font.family: "SF"
60              font.pointSize: 20
61          }
62          }
63  }
```

Листинг 11: Файл call.qml

```
 1  import QtQuick 2.0
 2  import QtQuick.Window 2.2
 3  import QtQuick.Controls 1.4
 4
 5  Window{
 6      id: phone
 7      height: Screen.height
 8      maximumHeight: Screen.height
 9      minimumHeight: Screen.height
10      width: 480
11      minimumWidth: 480
12      maximumWidth: 480
13      title: "Calling"
14      visible: true
15
16      Rectangle{
17          id: call_info
18          color: "blue"
19          border.color: "black"
20          border.width: 5
21          anchors{
22              top: parent.top
23          }
24          height: parent.height / 2
25          width: parent.width
26
27          Text {
28              id: dialingText
29              text: "Dialing:"
30              color: "#fbf1c7"
31              anchors{
32                  top: parent.top
33                  left: parent.left
34
35                  topMargin: parent.height / 3
```

```
36                  leftMargin: 20
37              }
38              renderType: Text.NativeRendering
39              horizontalAlignment: Text.AlignHCenter
40              verticalAlignment: Text.AlignVCenter
41              font.family: "SF"
42              font.pointSize: 20
43          }
44
45          Text {
46              id: call_number
47              objectName: "call_number"
48              color: "#fbf1c7"
49              width: parent.height / 2
50              anchors{
51                  top: dialingText.bottom
52                  left: parent.left
53
54                  topMargin: 10
55                  leftMargin: 20
56              }
57              renderType: Text.NativeRendering
58              horizontalAlignment: Text.AlignHCenter
59              verticalAlignment: Text.AlignVCenter
60              font.family: "SF"
61              font.pointSize: 20
62          }
63      }
64
65      Text{
66          id: call_timer
67          objectName: "call_timer"
68          color: "#000000"
69          anchors{
70              top: parent.top
71              right: parent.right
72
73              topMargin: parent.height / 3
74              leftMargin: 10
75          }
76          renderType: Text.NativeRendering
77          horizontalAlignment: Text.AlignHCenter
78          verticalAlignment: Text.AlignVCenter
79          font.family: "SF"
80          font.pointSize: 20
81
82      }
83
84      Image {
85          id: buttonHang
86          width: 100
87          height: 50
88          source: "qrc:///pics/hang.png"
89
90          anchors {
91              bottom: parent.bottom
92              bottomMargin: 20
```

```
93              topMargin: height / 2
94              horizontalCenter: parent.horizontalCenter
95              verticalCenter: parent.varticalCenter
96          }
97
98          MouseArea{
99              anchors.fill: parent
100             onClicked: hang()
101         }
102     }
103
104     Timer{
105         interval:1000
106         running: true
107         onTriggered: getTime()
108     }
109
110     function hang(){
111         window.hangUp();
112     }
113
114     function getTime(){
115         window.getTime();
116     }
117
118 }
```

Листинг 12: Файл Button.qml

```
1  import QtQuick 2.0
2  import QtQuick.Window 2.2
3  import QtQuick.Controls 1.4
4
5  Rectangle {
6
7          property string caption: ""
8          property int spacing
9
10         id: button1
11         width: parent.width / 3 - 2 * spacing / 3
12         height: parent.height / 4 - 3 * spacing / 4
13
14         Text {
15                 renderType: Text.NativeRendering
16                 horizontalAlignment: Text.AlignHCenter
17                 verticalAlignment: Text.AlignVCenter
18                 font.family: "SF"
19                 font.pointSize: 20
20                 text: caption
21                 width: parent.width
22                 height: parent.height
23          }
24
25
26         MouseArea{
27                 opacity:1
28                 anchors.fill: parent
```

```
29                  onClicked: phoneNumber.text += caption
30                  onDoubleClicked: {
31                          if(caption != 0) return
32                          phoneNumber.text = phoneNumber.text.substr(0, phoneNumber.
    text.length-1)
33                          phoneNumber.text += "+"
34                  }
35          }
36  }
```

Листинг 13: Файл qmldir

```
1  Button  Button.qml
```