

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Курсовая работа
по дисциплине «Объектно-ориентированное программирование»
на тему
**«Разработка графического интерфейса для управления роботом
KUKA youBot»**

Студенты
гр.3331506/10401

Мельникова А.И.,
Щелканова А.Р.
Громов И.С.

Преподаватель

Ананьевский М.С.

Санкт-Петербург
2024 г.

Содержание

Введение.....	3
1. Настройка инструментов разработки.....	4
1.1 Установка менеджера пакетов MSYS2	4
1.2 Установка Qt и дополнительных утилит	5
2. Разработка графического интерфейса.....	7
2.1 Архитектура проекта.....	7
2.2 Код	8
3. Результаты.....	18
Заключение	19
Список использованных источников.....	20

Введение

KUKA youBot - это мобильный манипулятор, разработанный для образовательных и исследовательских целей.

Система youBot состоит из двух частей:

- Всенаправленная мобильная платформа. Состоит из шасси робота, четырех механических колес, двигателей, блока питания и платы бортового компьютера. Пользователи могут либо запускать программы на этой плате, либо управлять ею с удаленного компьютера.
- Рука. Имеет пять степеней свободы (DOF) и двухпальцевый захват. При подключении к мобильной платформе манипулятором можно управлять с помощью встроенного ПК. Альтернативно, манипулятором можно управлять без мобильной платформы, используя собственный ПК, подключенный через кабель Ethernet.

Приложение, разработанное в процессе выполнения данной курсовой работы, позволяет управлять как мобильной платформой, так и рукой. Данный графический интерфейс разработан на языке программирования C++ с использованием фреймворка Qt от компании Qt Project и дополнительных библиотек, устанавливаемых для работы в указанной среде. Разработка ведётся на ОС Windows 10. Сборка производится при помощи средства автоматизации сборки ПО из исходного кода CMake.

Целью данной работы является создание графического интерфейса, который позволит управлять манипулятором KUKA youBot с ПК, на котором будет установлена данная программа.

1. Настройка инструментов разработки

Было принято решение вести разработку приложения с помощью среды разработки Qt вследствие её доступности, удобства работы, большого количества документации, а также возможности сборки программы как на ОС Windows, так и на Linux.

1.1 Установка менеджера пакетов MSYS2

В качестве наиболее простого и доступного способа установки фреймворка Qt и необходимых для работы библиотек была выбрана установка через инструмент MSYS2 – набор инструментов, позволяющий устанавливать в ОС Windows библиотеки и утилиты с помощью пакетного менеджера, подобно установке в Unix – подобных ОС.

Установка MSYS2 производится с официального сайта <https://www.msys2.org/> и является типовой установкой приложения в ОС Windows. После установки будет открыто следующее окно:

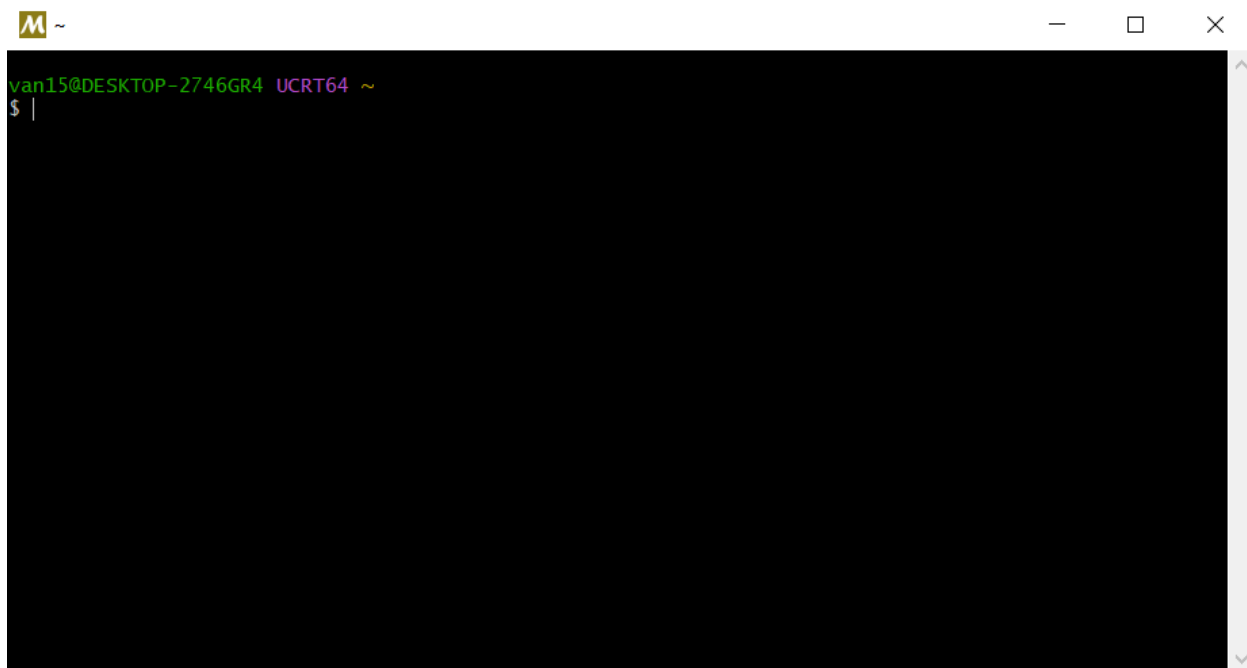
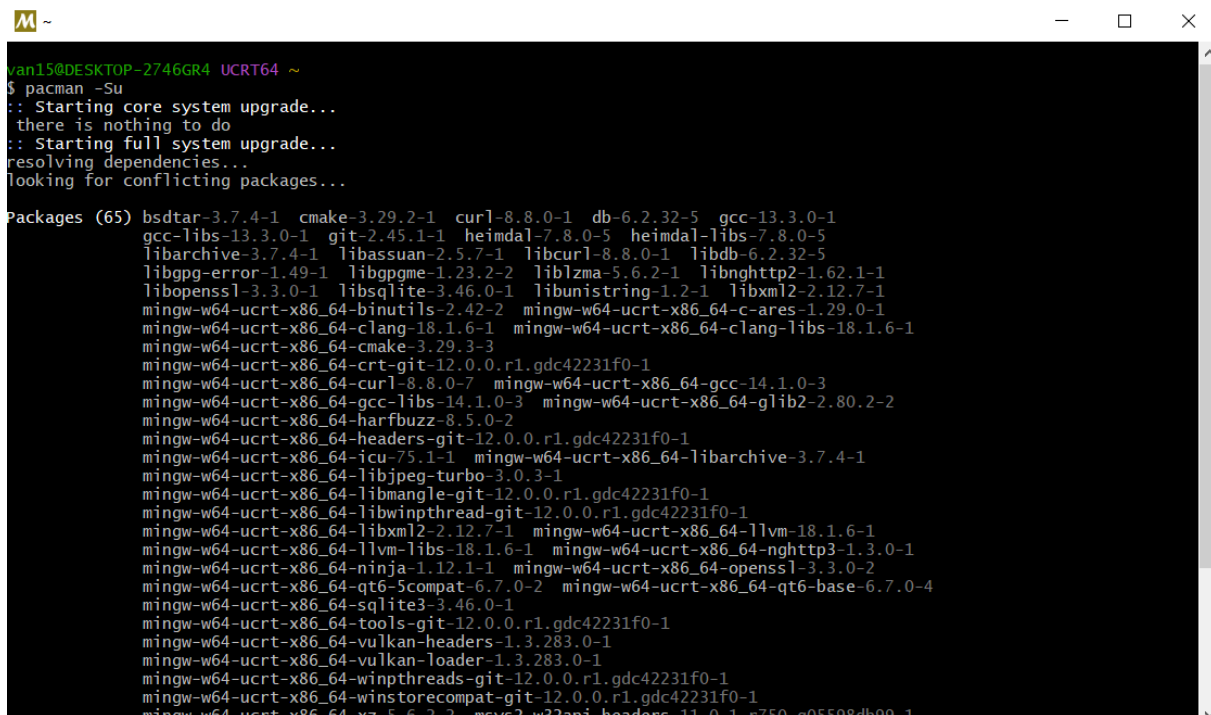


Рисунок 1.1 – Окно MSYS2

Установка и обновление пакетов далее ведётся с использованием команды `pacman`, например обновление уже установленных пакетов будет выглядеть так:



```
van15@DESKTOP-2746GR4 UCRT64 ~  
$ pacman -Su  
:: Starting core system upgrade...  
there is nothing to do  
:: Starting full system upgrade...  
resolving dependencies...  
looking for conflicting packages...  
  
Packages (65) bsdtar-3.7.4-1 cmake-3.29.2-1 curl-8.8.0-1 db-6.2.32-5 gcc-13.3.0-1  
gcc-libs-13.3.0-1 git-2.45.1-1 heimdal-7.8.0-5 heimdal-libs-7.8.0-5  
libarchive-3.7.4-1 libassuan-2.5.7-1 libcurl-8.8.0-1 libdb-6.2.32-5  
libgpg-error-1.49-1 libgpgme-1.23.2-2 liblzma-5.6.2-1 libnghttp2-1.62.1-1  
libopenssl-3.3.0-1 libsqlite-3.46.0-1 libunistring-1.2-1 libxml2-2.12.7-1  
mingw-w64-ucrt-x86_64-binutils-2.42-2 mingw-w64-ucrt-x86_64-c-ares-1.29.0-1  
mingw-w64-ucrt-x86_64-clang-18.1.6-1 mingw-w64-ucrt-x86_64-clang-libs-18.1.6-1  
mingw-w64-ucrt-x86_64-cmake-3.29.3-3  
mingw-w64-ucrt-x86_64-crt-git-12.0.0.r1.gdc42231f0-1  
mingw-w64-ucrt-x86_64-curl-8.8.0-7 mingw-w64-ucrt-x86_64-gcc-14.1.0-3  
mingw-w64-ucrt-x86_64-gcc-libs-14.1.0-3 mingw-w64-ucrt-x86_64-glib2-2.80.2-2  
mingw-w64-ucrt-x86_64-harfbuzz-8.5.0-2  
mingw-w64-ucrt-x86_64-headers-git-12.0.0.r1.gdc42231f0-1  
mingw-w64-ucrt-x86_64-icu-75.1-1 mingw-w64-ucrt-x86_64-libarchive-3.7.4-1  
mingw-w64-ucrt-x86_64-libjpeg-turbo-3.0.3-1  
mingw-w64-ucrt-x86_64-libmangle-git-12.0.0.r1.gdc42231f0-1  
mingw-w64-ucrt-x86_64-libwinpthread-git-12.0.0.r1.gdc42231f0-1  
mingw-w64-ucrt-x86_64-libxml2-2.12.7-1 mingw-w64-ucrt-x86_64-llvm-18.1.6-1  
mingw-w64-ucrt-x86_64-llvm-libs-18.1.6-1 mingw-w64-ucrt-x86_64-nghttp3-1.3.0-1  
mingw-w64-ucrt-x86_64-ninja-1.12.1-1 mingw-w64-ucrt-x86_64-openssl-3.3.0-2  
mingw-w64-ucrt-x86_64-qt6-5compat-6.7.0-2 mingw-w64-ucrt-x86_64-qt6-base-6.7.0-4  
mingw-w64-ucrt-x86_64-sqlite3-3.46.0-1  
mingw-w64-ucrt-x86_64-tools-git-12.0.0.r1.gdc42231f0-1  
mingw-w64-ucrt-x86_64-vulkan-headers-1.3.283.0-1  
mingw-w64-ucrt-x86_64-vulkan-loader-1.3.283.0-1  
mingw-w64-ucrt-x86_64-winthreads-git-12.0.0.r1.gdc42231f0-1  
mingw-w64-ucrt-x86_64-winstorescompat-git-12.0.0.r1.gdc42231f0-1  
mingw-w64-ucrt-x86_64-xz-5.6.2-2 msys2-w32api-headers-11.0.1.r750.g05598dh99-1
```

Рисунок 1.2 – Пример исполнения команды в MSYS2

1.2 Установка Qt и дополнительных утилит

Для дальнейшей разработки таким образом устанавливаем компилятор MinGW, инструмент сборки CMake, саму среду разработки Qt и необходимые утилиты. Список пакетов для установки приведён ниже.

```
mingw-w64-x86_64-qt-creator  
mingw-w64-x86_64-clang-libs  
mingw-w64-x86_64-cmake  
mingw-w64-x86_64-gdb  
mingw-w64-x86_64-qbs  
mingw-w64-x86_64-qt6-doc  
mingw-w64-x86_64-qt6-quicktimeline  
mingw-w64-x86_64-cc  
mingw-w64-x86_64-clang  
mingw-w64-x86_64-clang-tools-extra  
mingw-w64-x86_64-litehtml  
mingw-w64-x86_64-ninja  
mingw-w64-x86_64-qt6-5compat  
mingw-w64-x86_64-qt6-declarative  
mingw-w64-x86_64-qt6-doc  
mingw-w64-x86_64-qt6-quick3d  
mingw-w64-x86_64-qt6-quicktimeline  
mingw-w64-x86_64-qt6-serialport  
mingw-w64-x86_64-qt6-svg  
mingw-w64-x86_64-qt6-tools  
mingw-w64-x86_64-qt6-translations  
mingw-w64-x86_64-yaml-cpp
```

После установки данных пакетов на ПК будут установлены указанные выше утилиты. Внешний вид графического интерфейса CMake:

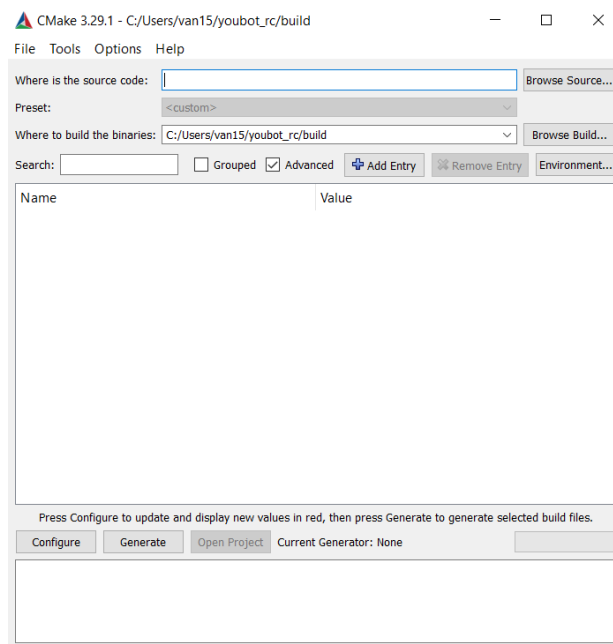


Рисунок 1.3 – Внешний вид графического интерфейса CMake

Внешний вид приложения QtCreator, в котором и будет вестись разработка:

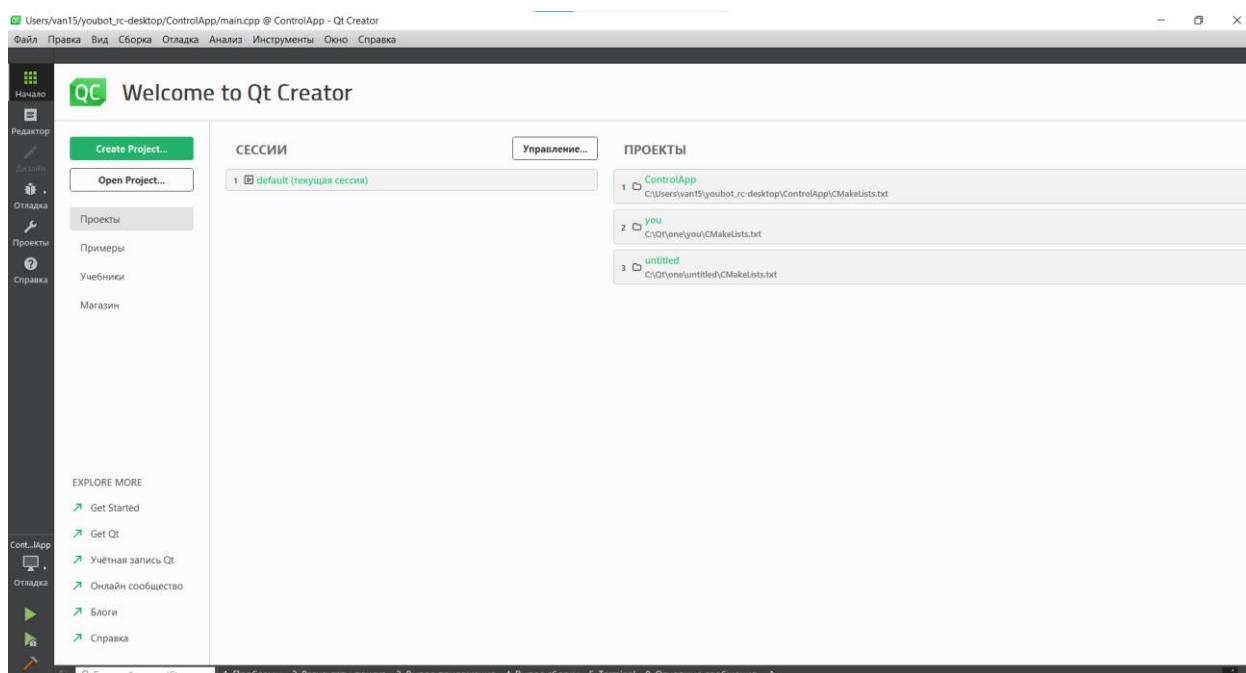


Рисунок 1.4 – Внешний вид графического интерфейса QtCreator

На этом установка необходимых для разработки инструментов завершена.

2. Разработка графического интерфейса

2.1 Архитектура проекта

В качестве архитектуры проекта была выбрана предлагаемая по умолчанию в Qt архитектура с условным разделением на разделы с заголовочными файлами (расширение .h), файлами с исходным кодом (расширение .cpp), файл с описанием пользовательского интерфейса в формате XML (расширение .ui) и сборочный файл (CMakeLists.txt), представленная на рисунке ниже. Блок icons не является автоматически создаваемым в Qt – этот раздел добавлен разработчиками для добавления иконки приложения.

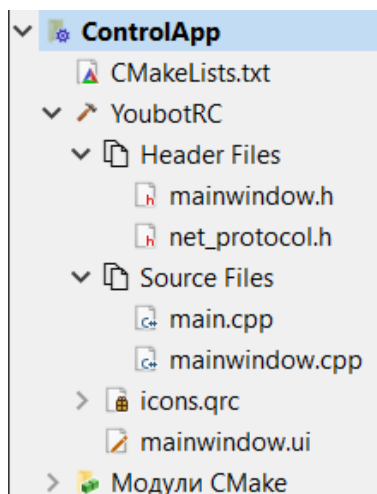


Рисунок 2.1 – Архитектура проекта в QtCreator

Если архитектуру проекта смотреть в папке этого проекта через Проводник Windows, выглядеть она будет ещё проще.

build	30.05.2024 18:23	Папка с файлами	
icons	20.05.2024 22:17	Папка с файлами	
.gitignore	20.05.2024 22:17	Исходный файл G...	1 КБ
CMakeLists.txt	20.05.2024 22:17	Файл "TXT"	3 КБ
CMakeLists.txt.user	30.05.2024 18:23	VisualStudio.user.0...	31 КБ
icons.qrc	20.05.2024 22:17	Файл "QRC"	1 КБ
main.cpp	20.05.2024 22:17	Файл "CPP"	1 КБ
mainwindow.cpp	20.05.2024 22:17	Файл "CPP"	15 КБ
mainwindow.h	20.05.2024 22:17	Файл "H"	2 КБ
mainwindow.ui	20.05.2024 22:17	Файл "UI"	28 КБ
net_protocol.h	20.05.2024 22:17	Файл "H"	1 КБ

Рисунок 2.2 – Архитектура проекта в папке проекта

Такой тип архитектуры является интуитивно понятным, простым для внесения изменений, установления взаимодействий между файлами и упрощает сборочный файл.

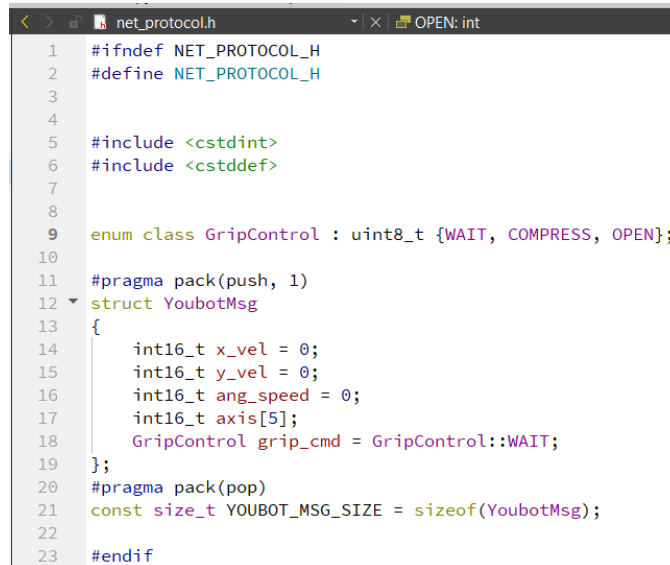
2.2 Код

Код сборочного файла представлен ниже.

```
CMakeLists.txt
1 cmake_minimum_required(VERSION 3.5)
2
3 project(ControlApp VERSION 1.0 LANGUAGES CXX)
4
5 set(CMAKE_INCLUDE_CURRENT_DIR ON)
6
7 set(CMAKE_AUTOUIC ON)
8 set(CMAKE_AUTOMOC ON)
9 set(CMAKE_AUTORCC ON)
10
11 set(CMAKE_CXX_STANDARD 17)
12 set(CMAKE_CXX_STANDARD_REQUIRED ON)
13
14 find_package(QT NAMES Qt6 Qt5 REQUIRED COMPONENTS Widgets Network)
15 find_package(Qt${QT_VERSION_MAJOR} REQUIRED COMPONENTS Widgets Network)
16
17 set(PROJECT_SOURCES
18     main.cpp
19     mainwindow.cpp
20     mainwindow.h
21     net_protocol.h
22     mainwindow.ui
23     icons.qrc
24 )
25
26 if(${QT_VERSION_MAJOR} GREATER_EQUAL 6)
27     qt_add_executable(YoubotRC
28         MANUAL_FINALIZATION
29         ${PROJECT_SOURCES}
30     )
31     # Define target properties for Android with Qt 6 as:
32     # set_property(TARGET YoubotRC APPEND PROPERTY QT_ANDROID_PACKAGE_SOURCE_DIR
33     #             ${CMAKE_CURRENT_SOURCE_DIR}/android)
34     # For more information, see https://doc.qt.io/qt-6/qt-add-executable.html#target-creation
35 else()
36     if(ANDROID)
37         add_library(YoubotRC SHARED
38             ${PROJECT_SOURCES}
39         )
40     # Define properties for Android with Qt 5 after find_package() calls as:
41     # set(ANDROID_PACKAGE_SOURCE_DIR "${CMAKE_CURRENT_SOURCE_DIR}/android")
42     else()
43         add_executable(YoubotRC
44             ${PROJECT_SOURCES}
45         )
46     endif()
47 endif()
48
49 target_link_libraries(YoubotRC PRIVATE Qt${QT_VERSION_MAJOR}::Widgets Qt${QT_VERSION_MAJOR}::Network)
50
51 # Qt for iOS sets MACOSX_BUNDLE_GUI_IDENTIFIER automatically since Qt 6.1.
52 # If you are developing for iOS or macOS you should consider setting an
53 # explicit, fixed bundle identifier manually though.
54 if(${QT_VERSION} VERSION_LESS 6.1.0)
55     set(BUNDLE_ID_OPTION MACOSX_BUNDLE_GUI_IDENTIFIER com.example.YoubotRC)
56 endif()
57 set_target_properties(YoubotRC PROPERTIES
58     ${BUNDLE_ID_OPTION}
59     MACOSX_BUNDLE_BUNDLE_VERSION ${PROJECT_VERSION}
60     MACOSX_BUNDLE_SHORT_VERSION_STRING ${PROJECT_VERSION_MAJOR}.${PROJECT_VERSION_MINOR}
61     MACOSX_BUNDLE TRUE
62     WIN32_EXECUTABLE TRUE
63 )
64
65 include(GNUInstallDirs)
66 install(TARGETS YoubotRC
67     BUNDLE DESTINATION .
68     LIBRARY DESTINATION ${CMAKE_INSTALL_LIBDIR}
69     RUNTIME DESTINATION ${CMAKE_INSTALL_BINDIR}
70 )
71
72 if(QT_VERSION_MAJOR EQUAL 6)
73     qt_finalize_executable(YoubotRC)
74 endif()
```

Рисунок 2.3 – Код сборочного файла

Файл `net_protocol.h` содержит описание данных, которые приложение посылает манипулятору по установленному TCP – соединению. Код этого файла представлен на рисунке ниже.



```
1  #ifndef NET_PROTOCOL_H
2  #define NET_PROTOCOL_H
3
4
5  #include <stdint>
6  #include <stddef>
7
8
9  enum class GripControl : uint8_t {WAIT, COMPRESS, OPEN};
10
11 #pragma pack(push, 1)
12 struct YoubotMsg
13 {
14     int16_t x_vel = 0;
15     int16_t y_vel = 0;
16     int16_t ang_speed = 0;
17     int16_t axis[5];
18     GripControl grip_cmd = GripControl::WAIT;
19 };
20 #pragma pack(pop)
21 const size_t YOUBOT_MSG_SIZE = sizeof(YoubotMsg);
22
23 #endif
```

Рисунок 2.4 – Код файла `net_protocol.h`

Здесь:

- `x_vel` – линейная скорость движения манипулятора в горизонтальном направлении `x`;
- `y_vel` – линейная скорость движения манипулятора в горизонтальном направлении `y`;
- `ang_speed` – угловая скорость поворота манипулятора;
- массив `axis` – углы поворота руки манипулятора (в 5 осях);
- `grip_cmd` – данные о поведении “клешни” манипулятора (ожидать команду, сжать, разжать).

Далее рассмотрим файл `mainwindow.h`, содержащий описание подключаемых библиотек, класса основного окна графического интерфейса, функций и переменных, используемых при работе приложения.

Среди переменных:

- `resendTime` – интервал отправки команд в миллисекундах;
- `sliderShortcutStep` – скорость движения слайдерами;
- `ui` – главное окно, на котором и располагаются все виджеты;
- `tcpSocket` – сокет, через который устанавливается соединение;
- `tcpResendTime` – таймер для отслеживания периодичности отправки данных;
- `txMsg` – структура с данными для отправки.

Код файла `mainwindow.h` представлен ниже.

```

1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5  #include <QLineEdit>
6  #include <QPushButton>
7  #include <QDebug>
8  #include <QTcpSocket>
9  #include <QKeyEvent>
10 #include <QIntValidator>
11 #include <QMessageBox>
12 #include <QTimer>
13 #include <QRandomGenerator>
14
15 #include "net_protocol.h"
16
17
18 QT_BEGIN_NAMESPACE
19 namespace Ui { class MainWindow; }
20 QT_END_NAMESPACE
21
22 class MainWindow : public QMainWindow
23 {
24     Q_OBJECT
25
26 public:
27     MainWindow(QWidget *parent = nullptr);
28     ~MainWindow();
29
30 protected:
31     void keyPressEvent(QKeyEvent *keyEvent) override;
32     void keyReleaseEvent(QKeyEvent *keyEvent) override;
33
34     void pressBut(QPushButton* button);
35     void releasBut(QPushButton* button);
36     void buttonInit();
37     void buttonHandle();
38     void buttonRandHandle();
39
40     void sliderInit();
41     void sliderHandle(int value);
42     bool sliderKeyHandle(QKeyEvent *keyEvent);
43
44     void uiValidator();
45
46     void requestNewConnection();
47     void disconnect();
48     void sendTcp();
49     void displayNetError(QAbstractSocket::SocketError socketError);
50     void disconnectedHandle();
51     void sendTcpComand();
52
53 private:
54     Ui::MainWindow *ui;
55     QTcpSocket *tcpSocket = nullptr;
56     QTimer *tcpResendTimer = nullptr;
57
58     YoubotMsg txMsg;
59     const int resendTime = 100; // ms
60     const int sliderShortcutStep = 2;
61 };
62
63 #endif // MAINWINDOW_H

```

Рисунок 2.5 – Код файла mainwindow.h

Более подробное рассмотрение функций приведено далее при описании файла mainwindow.cpp.

Поскольку размер файла mainwindow.cpp – более 400 строк, в данном отчете будут приведены только некоторые, наиболее важные и основные функции.

На рисунке ниже приведён фрагмент кода, содержащий подключение заголовочных файлов и инициализацию окна приложения.



```

1  #include "mainwindow.h"
2  #include "./ui_mainwindow.h"
3
4  MainWindow::MainWindow(QWidget *parent)
5      : QMainWindow(parent)
6      , ui(new Ui::MainWindow)
7      , tcpSocket(new QTcpSocket(this))
8  {
9      ui->setupUi(this);
10
11     buttonInit();
12     sliderInit();
13     uiValidator();
14
15     tcpResendTimer = new QTimer(this);
16     connect(tcpResendTimer, &QTimer::timeout, this, &MainWindow::sendTcp);
17     tcpResendTimer->start(resendTime);
18
19     connect(tcpSocket, &QAbstractSocket::errorOccurred, this, &MainWindow::displayNetError);
20     connect(tcpSocket, &QAbstractSocket::disconnected, this, &MainWindow::disconnectedHandle);
21 }

```

Рисунок 2.5 – Код инициализации окна приложения

На следующем рисунке представлена функция инициализации слайдеров (ползунков), при помощи которых регулируются повороты руки по осям, линейная и угловая скорости.



```

24 void MainWindow::sliderInit()
25 {
26     ui->labelLinVel->setText(QString::number(ui->sliderLinVel->value() / 100.0, 'f', 2));
27     ui->labelAngVel->setText(QString::number(ui->sliderAngVel->value() / 100.0, 'f', 2));
28     ui->labelAxis1->setText(QString::number(ui->sliderAxis1->value() / 100.0, 'f', 2));
29     ui->labelAxis2->setText(QString::number(ui->sliderAxis2->value() / 100.0, 'f', 2));
30     ui->labelAxis3->setText(QString::number(ui->sliderAxis3->value() / 100.0, 'f', 2));
31     ui->labelAxis4->setText(QString::number(ui->sliderAxis4->value() / 100.0, 'f', 2));
32     ui->labelAxis5->setText(QString::number(ui->sliderAxis5->value() / 100.0, 'f', 2));
33
34     txMsg.axis[0] = ui->sliderAxis1->value();
35     txMsg.axis[1] = ui->sliderAxis2->value();
36     txMsg.axis[2] = ui->sliderAxis3->value();
37     txMsg.axis[3] = ui->sliderAxis4->value();
38     txMsg.axis[4] = ui->sliderAxis5->value();
39
40     connect(ui->sliderLinVel, &QAbstractSlider::valueChanged, this, &MainWindow::sliderHandle);
41     connect(ui->sliderAngVel, &QAbstractSlider::valueChanged, this, &MainWindow::sliderHandle);
42     connect(ui->sliderAxis1, &QAbstractSlider::valueChanged, this, &MainWindow::sliderHandle);
43     connect(ui->sliderAxis2, &QAbstractSlider::valueChanged, this, &MainWindow::sliderHandle);
44     connect(ui->sliderAxis3, &QAbstractSlider::valueChanged, this, &MainWindow::sliderHandle);
45     connect(ui->sliderAxis4, &QAbstractSlider::valueChanged, this, &MainWindow::sliderHandle);
46     connect(ui->sliderAxis5, &QAbstractSlider::valueChanged, this, &MainWindow::sliderHandle);
47
48     connect(ui->sliderLinVel, &QAbstractSlider::sliderReleased, this, &MainWindow::sendTcpComand);
49     connect(ui->sliderAngVel, &QAbstractSlider::sliderReleased, this, &MainWindow::sendTcpComand);
50     connect(ui->sliderAxis1, &QAbstractSlider::sliderReleased, this, &MainWindow::sendTcpComand);
51     connect(ui->sliderAxis2, &QAbstractSlider::sliderReleased, this, &MainWindow::sendTcpComand);
52     connect(ui->sliderAxis3, &QAbstractSlider::sliderReleased, this, &MainWindow::sendTcpComand);
53     connect(ui->sliderAxis4, &QAbstractSlider::sliderReleased, this, &MainWindow::sendTcpComand);
54     connect(ui->sliderAxis5, &QAbstractSlider::sliderReleased, this, &MainWindow::sendTcpComand);
55 }

```

Рисунок 2.6 – Код инициализации слайдеров

На рисунке ниже приведена функция обработки событий при взаимодействии с уже указанными слайдерами.

```
58 void MainWindow::sliderHandle(int value)
59 {
60     QObject* pObject = sender();
61
62     if (pObject == ui->sliderLinVel) {
63         ui->labelLinVel->setText(QString::number(value / 100.0, 'f', 2));
64     } else if (pObject == ui->sliderAngVel) {
65         ui->labelAngVel->setText(QString::number(value / 100.0, 'f', 2));
66     } else if (pObject == ui->sliderAxis1) {
67         ui->labelAxis1->setText(QString::number(value / 100.0, 'f', 2));
68         txMsg.axis[0] = value;
69     } else if (pObject == ui->sliderAxis2) {
70         ui->labelAxis2->setText(QString::number(value / 100.0, 'f', 2));
71         txMsg.axis[1] = value;
72     } else if (pObject == ui->sliderAxis3) {
73         ui->labelAxis3->setText(QString::number(value / 100.0, 'f', 2));
74         txMsg.axis[2] = value;
75     } else if (pObject == ui->sliderAxis4) {
76         ui->labelAxis4->setText(QString::number(value / 100.0, 'f', 2));
77         txMsg.axis[3] = value;
78     } else if (pObject == ui->sliderAxis5) {
79         ui->labelAxis5->setText(QString::number(value / 100.0, 'f', 2));
80         txMsg.axis[4] = value;
81     }
82 }
```

Рисунок 2.7 – Код обработки событий на слайдерах

Важными являются функции, отвечающие за соединение и передачу данных по нему. Эти функции приведены ниже.

```
85 void MainWindow::sendTcpComand()
86 {
87     sendTcp();
88     tcpResendTimer->start(resendTime);
89 }
90
91
92 void MainWindow::requestNewConnection()
93 {
94     ui->butConnect->setEnabled(false);
95     tcpSocket->abort();
96     tcpSocket->connectToHost(ui->lineIp->text(), ui->linePort->text().toInt());
97 }
98
99
100 void MainWindow::disconnect()
101 {
102     tcpSocket->abort();
103     ui->butConnect->setEnabled(true);
104 }
105
106
107 void MainWindow::sendTcp()
108 {
109     if (tcpSocket->state() != QAbstractSocket::ConnectedState) {
110         return;
111     }
112
113     tcpSocket->write(reinterpret_cast<char*>(&txMsg), YOUBOT_MSG_SIZE);
114 }
```

Рисунок 2.8 – Код сетевых функций

Данные функции выполняют подключение по указанным IP-адресу и порту, разрыв соединения и отправку команд на манипулятор.

Инициализация кнопок управления и их обработчик событий представлены в функциях на рисунках ниже.

```

153 void MainWindow::buttonInit()
154 {
155     connect(ui->butConnect, &QPushButton::pressed, this, &MainWindow::requestNewConnection);
156     connect(ui->butDisconnect, &QPushButton::pressed, this, &MainWindow::disconnect);
157
158     connect(ui->butLeftForward, &QPushButton::pressed, this, &MainWindow::buttonHandle);
159     connect(ui->butLeftForward, &QPushButton::released, this, &MainWindow::buttonHandle);
160     connect(ui->butLeft, &QPushButton::pressed, this, &MainWindow::buttonHandle);
161     connect(ui->butLeft, &QPushButton::released, this, &MainWindow::buttonHandle);
162     connect(ui->butLeftBack, &QPushButton::pressed, this, &MainWindow::buttonHandle);
163     connect(ui->butLeftBack, &QPushButton::released, this, &MainWindow::buttonHandle);
164     connect(ui->butForward, &QPushButton::pressed, this, &MainWindow::buttonHandle);
165     connect(ui->butForward, &QPushButton::released, this, &MainWindow::buttonHandle);
166     connect(ui->butStop, &QPushButton::pressed, this, &MainWindow::buttonHandle);
167     connect(ui->butStop, &QPushButton::released, this, &MainWindow::buttonHandle);
168     connect(ui->butBack, &QPushButton::pressed, this, &MainWindow::buttonHandle);
169     connect(ui->butBack, &QPushButton::released, this, &MainWindow::buttonHandle);
170     connect(ui->butRightForward, &QPushButton::pressed, this, &MainWindow::buttonHandle);
171     connect(ui->butRightForward, &QPushButton::released, this, &MainWindow::buttonHandle);
172     connect(ui->butRight, &QPushButton::pressed, this, &MainWindow::buttonHandle);
173     connect(ui->butRight, &QPushButton::released, this, &MainWindow::buttonHandle);
174     connect(ui->butRightBack, &QPushButton::pressed, this, &MainWindow::buttonHandle);
175     connect(ui->butRightBack, &QPushButton::released, this, &MainWindow::buttonHandle);
176     connect(ui->butRotLeft, &QPushButton::pressed, this, &MainWindow::buttonHandle);
177     connect(ui->butRotLeft, &QPushButton::released, this, &MainWindow::buttonHandle);
178     connect(ui->butRotRight, &QPushButton::pressed, this, &MainWindow::buttonHandle);
179     connect(ui->butRotRight, &QPushButton::released, this, &MainWindow::buttonHandle);
180     connect(ui->butCompress, &QPushButton::pressed, this, &MainWindow::buttonHandle);
181     connect(ui->butCompress, &QPushButton::released, this, &MainWindow::buttonHandle);
182     connect(ui->butOpen, &QPushButton::pressed, this, &MainWindow::buttonHandle);
183     connect(ui->butOpen, &QPushButton::released, this, &MainWindow::buttonHandle);
184
185     connect(ui->butRandomMove, &QPushButton::pressed, this, &MainWindow::buttonRandHandle);
186     connect(ui->butRandomMove, &QPushButton::released, this, &MainWindow::buttonHandle);
187 }

```

Рисунок 2.9 – Код инициализации кнопок управления

```

218 void MainWindow::buttonHandle()
219 {
220     if (ui->butStop->isDown()) {
221         txMsg.x_vel = 0;
222         txMsg.y_vel = 0;
223         txMsg.ang_speed = 0;
224     } else {
225         txMsg.x_vel = ui->sliderLinVel->value()
226         * (static_cast<int>(ui->butForward->isDown() || ui->butLeftForward->isDown() || ui->butRightForward->isDown())
227         - static_cast<int>(ui->butBack->isDown() || ui->butLeftBack->isDown() || ui->butRightBack->isDown()));
228
229         txMsg.y_vel = ui->sliderLinVel->value()
230         * (static_cast<int>(ui->butLeft->isDown() || ui->butLeftBack->isDown() || ui->butLeftForward->isDown())
231         - static_cast<int>(ui->butRight->isDown() || ui->butRightBack->isDown() || ui->butRightForward->isDown()));
232
233         txMsg.ang_speed = ui->sliderAngVel->value()
234         * (static_cast<int>(ui->butRotLeft->isDown()) - static_cast<int>(ui->butRotRight->isDown()));
235     }
236
237     if (ui->butOpen->isDown()) {
238         txMsg.grip_cmd = GripControl::OPEN;
239     } else if (ui->butCompress->isDown()) {
240         txMsg.grip_cmd = GripControl::COMPRESS;
241     } else {
242         txMsg.grip_cmd = GripControl::WAIT;
243     }
244
245     sendTcp();
246     tcpResendTimer->start(resendTime);
247 }

```

Рисунок 2.10 – Код обработки событий на кнопках

Как уже было написано выше, IP-адрес и порт для подключения указываются в специальных полях. Контроль вводимых в эти поля параметров представлен в функции на рисунке ниже.

```
190 void MainWindow::uiValidator()  
191 {  
192     QString IpRange = "(?:[0-1]?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])";  
193     QRegularExpression IpRegex ("^" + IpRange  
194     + "(\\.|" + IpRange + ")"  
195     + "(\\.|" + IpRange + ")"  
196     + "(\\.|" + IpRange + ")$");  
197     QRegularExpressionValidator *ipValidator = new QRegularExpressionValidator(IpRegex, this);  
198     ui->lineIp->setValidator(ipValidator);  
199  
200     ui->linePort->setValidator(new QIntValidator(1024, 65535, this));  
201 }
```

Рисунок 2.11 – Код полей сетевых настроек

Управление слайдерами, а с ними и манипулятором может осуществляться через клавиатуру (цифры). Код функции, которая позволяет это осуществлять, представлен на рисунке ниже.

```
250 bool MainWindow::sliderKeyHandle(QKeyEvent *keyEvent)  
251 {  
252     int controlModifier = (keyEvent->modifiers() == Qt::ControlModifier) ? -1 : 1;  
253  
254     switch (keyEvent->key())  
255     {  
256     case Qt::Key_1:  
257         ui->sliderAxis1-> setValue(ui->sliderAxis1->value() + sliderShortkeyStep * controlModifier);  
258         break;  
259     case Qt::Key_2:  
260         ui->sliderAxis2-> setValue(ui->sliderAxis2->value() + sliderShortkeyStep * controlModifier);  
261         break;  
262     case Qt::Key_3:  
263         ui->sliderAxis3-> setValue(ui->sliderAxis3->value() + sliderShortkeyStep * controlModifier);  
264         break;  
265     case Qt::Key_4:  
266         ui->sliderAxis4-> setValue(ui->sliderAxis4->value() + sliderShortkeyStep * controlModifier);  
267         break;  
268     case Qt::Key_5:  
269         ui->sliderAxis5-> setValue(ui->sliderAxis5->value() + sliderShortkeyStep * controlModifier);  
270         break;  
271     default:  
272         return false;  
273         break;  
274     }  
275  
276     return true;  
277 }
```

Рисунок 2.12 – Код управления слайдерами через клавиатуру

Разработанное в ходе выполнения этой курсовой работы приложение позволяет управлять манипулятором при помощи клавиатуры. Функции, рассмотренные далее – `keyPressEvent` и `keyReleaseEvent` (обработчики нажатия и отпускания клавиш соответственно), однотипны и объемны, поэтому ниже будет приведен пример лишь одной из них, а также функций, вызывающих эти обработчики.

```

280 void MainWindow::keyPressEvent(QKeyEvent *keyEvent)
281 {
282     if (sliderKeyHandle(keyEvent)) {
283         keyEvent->accept();
284         return;
285     }
286
287     if(keyEvent->isAutoRepeat()) {
288         keyEvent->accept();
289         return;
290     }
291
292     switch (keyEvent->key())
293     {
294     case Qt::Key_Q:
295         pressBut(ui->butLeftForward);
296         break;
297     case Qt::Key_W:
298         pressBut(ui->butForward);
299         break;
300     case Qt::Key_E:
301         pressBut(ui->butRightForward);
302         break;
303     case Qt::Key_A:
304         pressBut(ui->butLeft);
305         break;
306     case Qt::Key_S:
307         pressBut(ui->butStop);
308         break;
309     case Qt::Key_D:
310         pressBut(ui->butRight);
311         break;
312     case Qt::Key_Z:
313         pressBut(ui->butLeftBack);
314         break;
315     case Qt::Key_X:
316         pressBut(ui->butBack);
317         break;
318     case Qt::Key_C:
319         pressBut(ui->butRightBack);
320         break;
321     case Qt::Key_Comma:
322         pressBut(ui->butRotLeft);
323         break;
324     case Qt::Key_Period:
325         pressBut(ui->butRotRight);
326         break;
327     case Qt::Key_P:
328         pressBut(ui->butRandomMove);
329         break;
330     case Qt::Key_R:
331         if (!ui->butOpen->isDown()) {
332             pressBut(ui->butCompress);
333         }
334         break;
335     case Qt::Key_F:
336         if (!ui->butCompress->isDown()) {
337             pressBut(ui->butOpen);
338         }
339         break;
340
341     default:
342         break;
343     }
344
345     keyEvent->accept();
346 }

```

Рисунок 2.13– Код обработки события нажатия клавиш движения

```

411 void MainWindow::pressBut(QPushButton* button)
412 {
413     button->setDown(true);
414     emit button->pressed();
415 }
416
417
418 void MainWindow::releasBut(QPushButton* button)
419 {
420     button->setDown(false);
421     emit button->released();
422 }

```

Рисунок 2.14 – Код обработки событий нажатия и отпускания клавиш

Рассмотрение файла/mainwindow.cpp на этом завершено.

Окошко редактирования приложения в непосредственно графическом режиме, отработки его внешнего вида, связанное с файлом/mainwindow.ui, представлено на рисунке ниже.

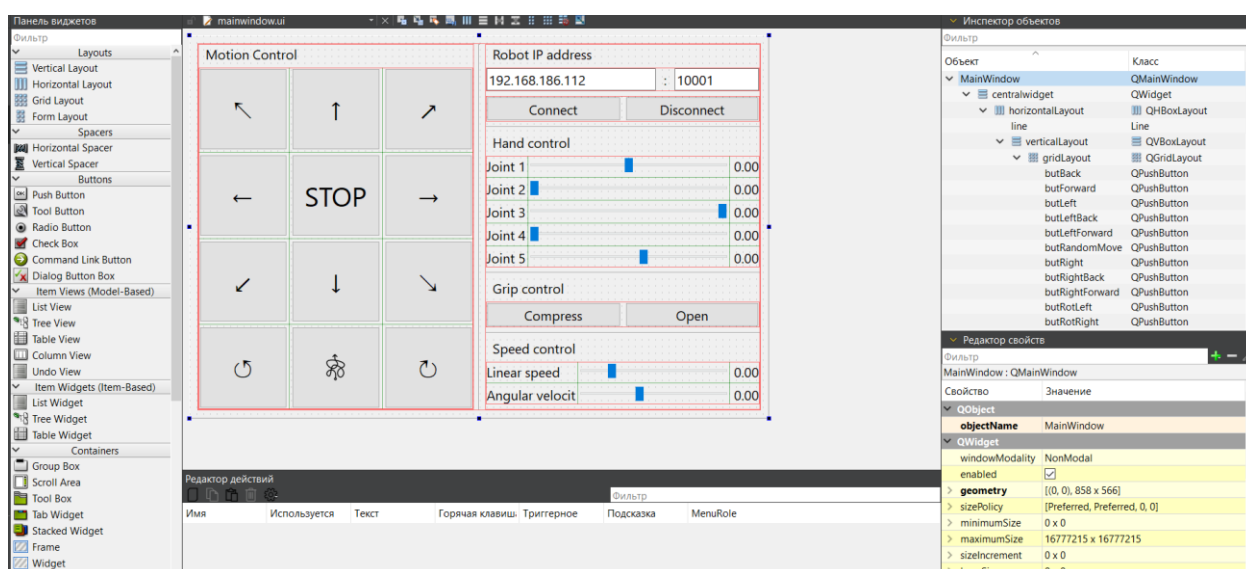
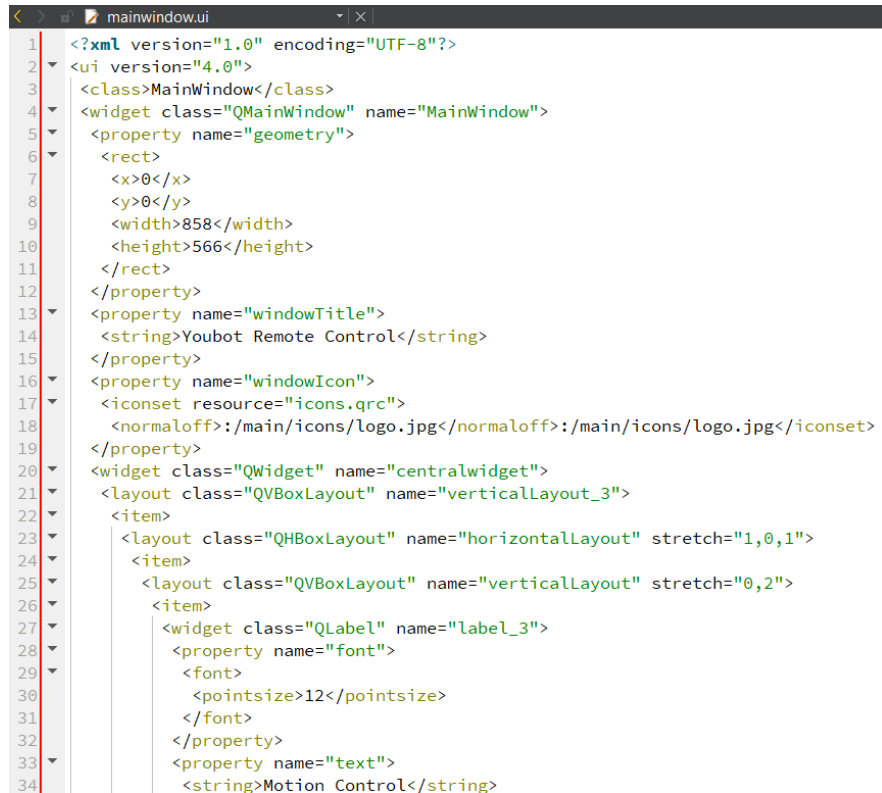


Рисунок 2.15 – Окно редактирования внешнего вида приложения

Сам файл `mainwindow.ui`, как было написано ранее, имеет XML – формат, очень объемен и особого интереса не представляет, однако его часть будет приведена ниже.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ui version="4.0">
3   <class>MainWindow</class>
4   <widget class="QMainWindow" name="MainWindow">
5     <property name="geometry">
6       <rect>
7         <x>0</x>
8         <y>0</y>
9         <width>858</width>
10        <height>566</height>
11      </rect>
12    </property>
13    <property name="windowTitle">
14      <string>Youbot Remote Control</string>
15    </property>
16    <property name="windowIcon">
17      <iconset resource="icons.qrc">
18        <normaloff>:/main/icons/logo.jpg</normaloff>:/main/icons/logo.jpg</iconset>
19      </property>
20    <widget class="QWidget" name="centralwidget">
21      <layout class="QVBoxLayout" name="verticalLayout_3">
22        <item>
23          <layout class="QHBoxLayout" name="horizontalLayout" stretch="1,0,1">
24            <item>
25              <layout class="QVBoxLayout" name="verticalLayout" stretch="0,2">
26                <item>
27                  <widget class="QLabel" name="label_3">
28                    <property name="font">
29                      <font>
30                        <pointsize>12</pointsize>
31                      </font>
32                    </property>
33                    <property name="text">
34                      <string>Motion Control</string>
```

Рисунок 2.16 – Часть кода файла `mainwindow.ui`

На этом рассмотрение кода, разработанного в процессе выполнения данной курсовой работы, завершено.

3. Результаты

Результатом выполнения данной курсовой работы является графический интерфейс для управления мобильным манипулятором KUKA youBot.

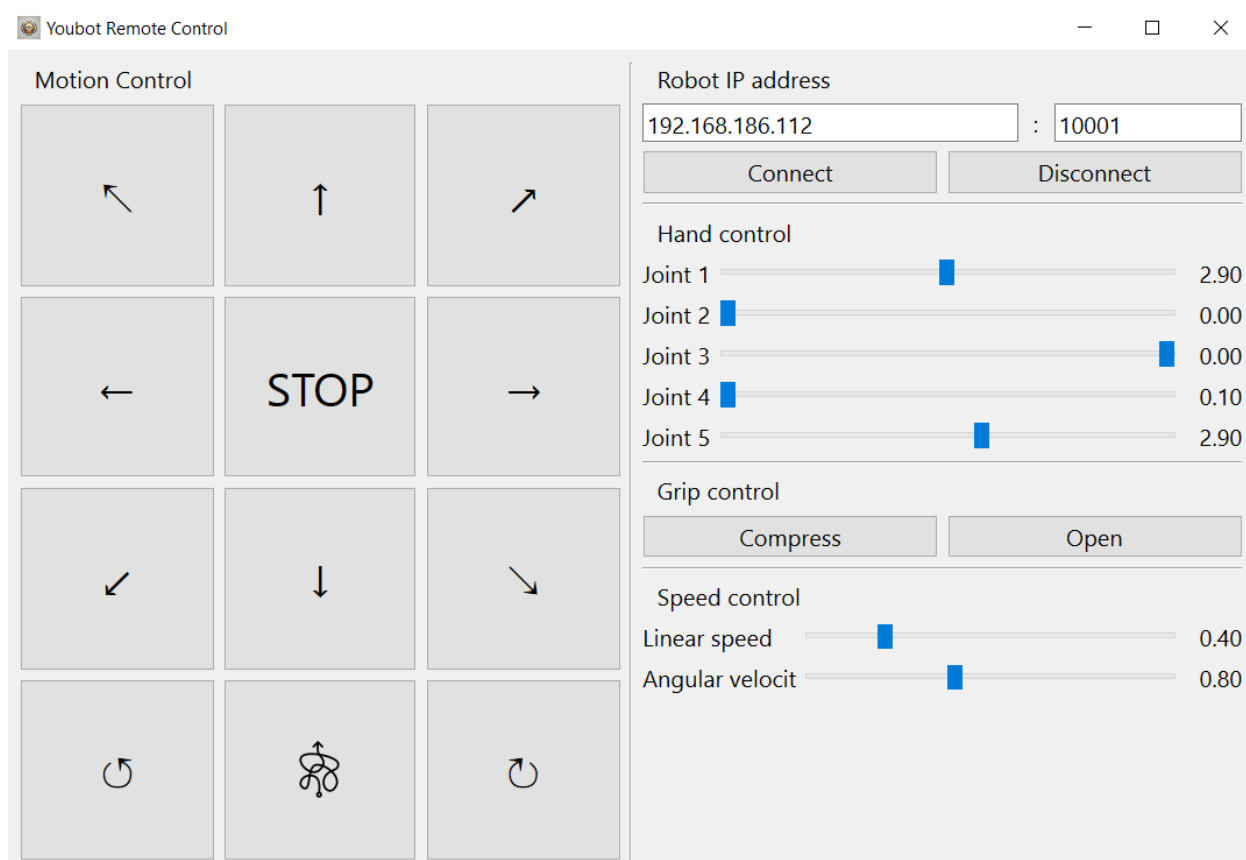


Рисунок 3.1 – Итоговый внешний вид приложения

Заключение

В ходе выполнения курсовой работы в среде разработки Qt был создан кроссплатформенный графический интерфейс для управления мобильным манипулятором KUKA youBot, позволяющий управлять его движениями и перемещениями (углами поворота пяти осей руки, линейной и угловой скоростями, направлением движения и состоянием “клешни” захвата).

При работе над проектом был создан репозиторий на GitHub:

https://github.com/KirillHit/youbot_rc

Список использованных источников

1. Документация Qt – URL: <https://doc.qt.io/qt-6>
2. Устанавливаемые пакеты – URL: <https://pupli.net/2022/12/22/install-qt-6-using-msys2/>