Министерство образования, науки, молодежи и спорта Украины

Донецкий национальный университет

Физический факультет

Кафедра компьютерных технологий

**Курсовая работа**

Тема: “**Модель ситуационного управления LEGO-роботом**”

Отчет

Исполнитель:

студент 2 курса кафедры КТ

Базь М.В.

Руководитель:

д.т.н., проф. Каргин А.А.

Донецк, 2011

Министерство образования, науки, молодежи и спорта Украины

Донецкий национальный университет

Физический факультет

Кафедра компьютерных технологий

Утверждаю

Зав. кафедрой

подпись

дата

**Задание**

На курсовую работу студента 2 курса

Базь Максима Витальевича

(фамилия, имя, отчество)

Специализация программирование

Тема курсовой работы “Модель ситуационного управления LEGO-роботом”

Вид работы (исследовательская, прикладная) прикладная

Краткая постановка задачи Решение задачи движения LEGO-робота по заданному маршруту

Исходные данные литературные источники и библиотека lejOS NXT

Ожидаемые результаты движение LEGO-робота по заданному маршруту

Календарный план работы

|  |  |  |
| --- | --- | --- |
| Дата проведения консультации | Этап выполнения работы | Отметка о выполнении |
| 04.03.11 | Постановка задачи |  |
| 18.03.11 | Изучение предметной области |  |
| 08.04.11 | Разработка структуры проекта |  |
| 18.05.11 | Предоставление результатов,  подведение итогов проделанной работы |  |

Оценка курсовой работы (выставляется руководителем до защиты)

Дата выдачи задания

Руководитель (Ф.И.О, подпись)

Студент (подпись)

РЕФЕРАТ

Отчет состоит из 27 с., 3 рис., 8 табл., 3 приложений, 7 источников.

Объект исследования — модель управления LEGO-роботом.

Цель курсовой работы — разработка приложения с помощью WinAPI функций для управления движением LEGO-робота по заданному маршруту.

Метод исследования — анализ возможностей WinAPI функций для реализации движения LEGO-робота.

В работе был разработан модуль управления LEGO-роботом. Библиотека позволяет выполнить движение по указанному маршруту, а также управлять дополнительными возможностями робота — звуковым и световым устройствами.

Во время движения по маршруту система анализирует показания датчиков расстояния и столкновения. При возникновении неожиданного препятствия управляющая программа прекращает движение робота и уведомляет пользователя.

Модуль спроектирован в соответствии с принципами объектно-ориентированного дизайна. Пользователи не зависят от моей библиотеки, поэтому могут подключать ее в режиме реального времени.

Для тестирования библиотеки было создано Windows-приложение с использованием WinAPI функций. Программа содержит кнопки для тестирования каждой функциональности модуля, а также окно выбора маршрута для движения.

Модуль управления может быть использован в любой сфере применения роботов: производство, исследования, обслуживание или работа с опасными веществами.

Дальнейшее развитие приложения связано с реализацией автоматического корректирования маршрута при возникновении неожиданных препятствий.

LEGO-РОБОТ, lejOS NXT, модуль, библиотека, WinAPI, C++, WINDOWS-ПРИЛОЖЕНИЯ, интерпретатор, транслятор.

СОДЕРЖАНИЕ

[Введение 5](#_Toc293872506)

[1. Постановка задачи 6](#_Toc293872507)

[2. Обзор подходов 7](#_Toc293872508)

[2.1 Обзор существующих библиотек 7](#_Toc293872509)

[2.2 Сравнение моделей управления роботом. 7](#_Toc293872510)

[3. Модель управления LEGO-роботом 8](#_Toc293872511)

[3.1 Формальное описание 8](#_Toc293872512)

[3.2 Интерпретатор команд 8](#_Toc293872513)

[4. Программная реализация 10](#_Toc293872514)

[4.1 Модуль управления LEGO-роботом 10](#_Toc293872515)

[4.1.1 Проектирование программного продукта 10](#_Toc293872516)

[4.1.2 Разработка библиотеки 13](#_Toc293872517)

[4.2 Тестовое Windows-приложение 14](#_Toc293872518)

[4.2.1 Создание класса окна 14](#_Toc293872519)

[4.2.2 Разработка интерфейса программы 14](#_Toc293872520)

[5. Описание контрольного примера 16](#_Toc293872521)

[6. Необходимые аппаратные и программные средства 17](#_Toc293872522)

[7. Инструкция пользователя 18](#_Toc293872523)

[Выводы 19](#_Toc293872524)

[Список ссылок 20](#_Toc293872525)

[Приложение A 21](#_Toc293872526)

[Приложение B 25](#_Toc293872527)

[Приложение С 26](#_Toc293872528)

## Введение

На сегодняшний день существует множество библиотек для управления LEGO-роботом. Они предлагают управление роботом посредством элементарных команд. Моя работа заключается в решении задачи движения по заданному маршруту.

Решение не сводится к простому посещению контрольных точек. Во время движения необходимо анализировать показания различных датчиков о состоянии окружающей среды и принимать соответствующие меры.

При разработке приложений скорость создания продукта является одним из основных факторов успешности проекта. Когда программисту потребуется реализация движения робота по указанному им маршруту, он использует моё, готовое решение, а не будет изобретать свой способ. Более того, я гарантирую, что моя разработка является протестированной и отказоустойчивой, что нельзя сказать о программе, написанной в спешке.

Сегодня разработка качественных программных продуктов подразумевает объектно-ориентированный дизайн. Я спроектировал свое решение согласно этим принципам. Разработчики маршрута не зависят от моей программы. Более того, программистам не потребуется перекомпилировать свою программу, если я обновлю свой проект.

## Постановка задачи

Ставится задача реализации движения LEGO-робота по заранее известному маршруту. Я решил выполнить задачу в виде создания независимого модуля. Это является наиболее удобным способом предоставления функциональности для сторонних разработчиков. Мое решение позволило расширить список действий для управления LEGO-роботом.

1. Движение по указанному маршруту.
2. Включение/выключение фонаря.
3. Подача звукового сигнала.
4. Получение заряда батареи.

Для тестирования моего модуля я создам демонстрационное Windows-приложение, используя WinAPI функции.

Необходимые действия для выполнения поставленной задачи:

* Ознакомиться с программированием роботов [1].
* Изучить справку по работе с библиотекой lejOS NXT [2].
* Изучить работу с сокетами [3].
* Изучить принципы объектно-ориентированного дизайна [4,5].
* Разработать проект программы.
* Создать интерфейс маршрута движения.
* Разработать интерфейсы управления роботом.
* Реализовать программу.
* Разработать модуль управления LEGO-роботом.
* Создать тестовое Windows-приложение.
* Стабилизировать работу программы.
* Протестировать программный продукт.
* Исправить найденные ошибки.

## Обзор подходов

2.1 Обзор существующих библиотек

На данный момент существует множество библиотек для управления LEGO-роботом на языке C++.

Наиболее популярные из них — это:

* brickOS [6].
* NXT++ [7].
* Lestat [8].

Моя работа направлена на упрощение способа управления роботом — инкапсулирование набора действий в одну задачу.

2.2 Сравнение моделей управления роботом.

Среди программного управления рассматриваются модели управления с прямыми и обратными связями (см. табл 2.1).

Первая модель представляет собой идеализацию действий и окружающего пространства. Однако она более простая в реализации, поэтому на данном этапе я выбрал именно ее.

Таблица 2.1 — Модели управления роботами

|  |  |  |
| --- | --- | --- |
| Ограничение | Прямые связи | Обратные связи |
| Обязательные начальные условия | + | + |
| Жесткие требования к скорости перемещения | + | — |
| Требования к мгновенным режимам разгона/торможения | + | — |

## Модель управления LEGO-роботом

3.1 Формальное описание

Рассматривается движение LEGO-робота по заранее установленному маршруту. Реализация модели управления с прямыми связями подразумевает идеализированную способность робота к мгновенному разгону и торможению. Необходимым условием также является константность скорости движения робота.

Маршрут движения задается в виде абсолютной системы координат. Из маршрута я выделяю независимые контуры — прямолинейный участок маршрута, определяемые по формуле (3.1).

, (3.1)

где — координата точки на оси абсцисс;

— координата точки на оси ординат.

Над каждым контуром будет производиться некоторое действие. Продолжительность исполнения действия определяется временем . Я строю задание как последовательность команд, заданных формулой (3.2)

, (3.2)

где — действие;

— время исполнения действия.

3.2 Интерпретатор команд

В реализации я использую императивный язык, таким образом, последовательность исполнения команд однозначно определена.

Я выделю множество допустимых команд, которые будут использоваться при написании программы управления.

* Вперед
* Стоп
* Влево
* Вправо
* Задержка *t*

Задержка — это отдельный вид команды. Интерпретатор при обработке этой команды задерживает выполнение следующей команды на *t* единиц.

Рассмотрим пример управляющей программы. Я должен указать начальное условие: робот находится на (1) вдоль прямой (1—2) (на рис. 3.1). Последовательность команд указана ниже.

1. Вперед
2. Задержка
3. Стоп
4. Вправо
5. Вперед
6. Задержка
7. Стоп
8. Влево
9. Вперед
10. Задержка
11. Стоп

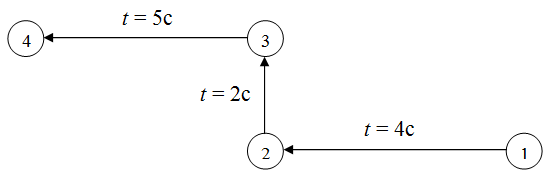


Рисунок 3.1 — Иллюстрация управляющей программы

## 4. Программная реализация

4.1 Модуль управления LEGO-роботом

Для использования низкоуровневых команд я выбрал библиотеку lejOS NXT. Ее возможности позволяют управлять роботом при помощи технологии Bluetooth, что гораздо удобнее управления посредством USB-кабеля.

В моей реализации эта библиотека является единственным незаменяемым объектом системы. Инициализация библиотеки осуществляется через консоль. Связь между ней и моим модулем реализована с помощью сокетов. Поэтому изначально спроектированный интерфейс IRobotConsoleControl был также разработан для консольного управления.

### 4.1.1 Проектирование программного продукта

Первым делом я создал интерфейс IRobotConsoleControl (см. табл. 4.1, листинг A.1). В нем содержатся основные методы управления LEGO-роботом, которые в любом случае необходимо реализовать каждому конкретному модулю.

Таблица 4.1 — Методы интерфейса IRobotConsoleControl

|  |  |
| --- | --- |
| Метод | Описание |
| void OpenConnection(void); | Установить соединение с роботом. |
| void CloseConnection(void); | Прервать соединение с роботом. |
| HWND GetConsoleHandle(void); | Получить дескриптор консоли. |
| void PassRoute(IRobotRoute &route); | Проехать по маршруту. |
| wstring GetBattery(void); | Получить заряд батареи. |

Метод PassRoute осуществляет движение по любому маршруту, поддерживающему интерфейс IRobotRoute (см. табл. 4.2, листинг A.2).

Таблица 4.2 — Методы интерфейса IRobotRoute

|  |  |
| --- | --- |
| Метод | Описание |
| IRobotRoute(double scale); | Конструктор. Параметр — масштаб карты. |
| RouteIterator Begin(void); | Итератор первой путевой точки. |
| RouteIterator End(void); | Итератор точки, следующей за последней. |
| Point GetStartPos(void); | Точка исходного положения робота. |
| Point GetOffset(void); | Условная точка исходного направления робота. |

Маршрут определяется списком абсолютных координат (std::list), которые робот должен последовательно посетить. Генератор маршрута обязан указать масштаб карты в параметре scale (1 ед. = 1 метр \* scale), а также начальные координаты робота и смещение — исходное направление робота.

Посмотрим на схему взаимодействия интерфейсов и пользователей (на рис. 4.1). «Пользователь» представляет собой неопределенное число программистов, использующих библиотеку, совместимую с IRobotConsoleControl. Взаимодействие через интерфейс означает то, что разработчик получает возможность менять библиотеки в режиме онлайн.

В свою очередь, IRobotConsoleControl взаимодействует с интерфейсом маршрута IRobotRoute. Буквально это обозначает следующее: «любая библиотека может успешно работать с любым маршрутом».

Отдельный класс программистов, специализирующихся на разработке маршрута движения, используют собственные классы на основе IRobotRoute.

Таким образом, на данном этапе я добился независимости маршрута и библиотеки управления роботом.

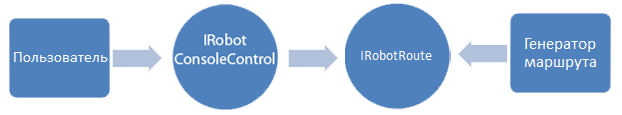


Рисунок 4.1 — Схема взаимодействия интерфейсов и пользователей

Отличительной особенностью LEGO-роботов является способность конструирования новых моделей из разнообразного набора сенсоров и датчиков.

Для удобного варьирования набором компонент я создал интерфейсы дополнительных устройств (см. табл. 4.3, листинги A.3 и A.4). У представленного робота в наличии есть звуковое оборудование и фонарь.

Таблица 4.3 — Методы интерфейсов IRobotHasSound и IRobotHasLight

|  |  |  |
| --- | --- | --- |
| Интерфейс | Метод | Описание |
| IRobotHasSound | void SoundOn(void); | Включить звуковой сигнал. |
| IRobotHasSound | void SoundOff(void); | Выключить звуковой сигнал. |
| IRobotHasLight | void LightOn(void); | Включить фонарь. |
| IRobotHasLight | void LightOff(void); | Выключить фонарь. |

Посмотрим на схему реализации такой возможности (на рис. 4.2).

Для написания собственной реализации модуля управления роботом программисту необходимо наследовать свой класс от IRobotConsoleControl и от всех интерфейсов устройств, которые он хочет поддерживать.

При этом пользователю модуля очень удобно проверить, поддерживает ли текущая библиотека управление звуком. Достаточно проверить, реализует ли она интерфейс IRobotHasSound.

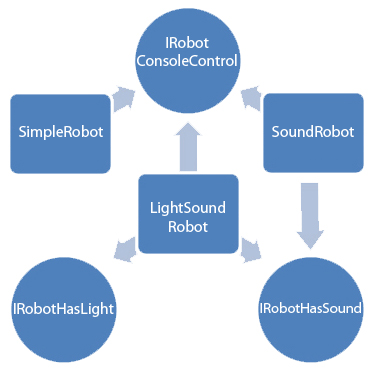


Рисунок 4.2 — Схема реализации различных модулей

### 4.1.2 Разработка библиотеки

Мой модуль реализует интерфейс IRobotConsoleControl, а также функционал IRobotHasLight и IRobotHasSound (см. рис. В, листинг A.5).

При открытии соединения программа выполняет следующие шаги:

* Запускает библиотеку lejOS через консоль.
* Устанавливает с ней соединение через сокеты.

Метод PassRoute выполняет следующие действия:

* Проверяет наличие активного подключения к библиотеке lejOS.
* Создает последовательность команд.
* Учитывая задержку, отправляет команды роботу.
* Проверяет отчет о выполнении команды.
* Анализирует показания датчиков.

Каждая команда и ее задержка хранятся в структуре Action (см. листинг А.6). Команда — это целочисленное значение из перечисления, описанного в той же структуре.

Каждое действие предваряется набором проверок. При возникновении ошибки программа возбуждает соответствующее исключение (см. табл. 4.4).

Таблица 4.4 — Генерируемые исключнения

|  |  |
| --- | --- |
| Исключение | Описание |
| SoketConnectionIsAlreadyOpened | Попытка открыть второе соединение. |
| InitWinSockException | Ошибка при инициализации WSA. |
| InitSocketException | Ошибка при создании сокета |
| InitConnectSocketException | Ошибка при соединении с сервером. |
| SendSocketException | Ошибка при отправке сокета. |
| SoketConnectionIsNotEstablished | Попытка использования закрытого сокета. |
| InvalidAnswerException | Неверный ответ от сервера. |
| UnexpectedObstacleExseption | Обнаружено неизвестное препятствие. |

4.2 Тестовое Windows-приложение

Для тестирования моего модуля я создал демонстрационное SDI приложение, позволяющее вызвать все методы моей библиотеки.

### 4.2.1 Создание класса окна

Работу с созданием, обработкой сообщений и завершением окна я инкапсулировал в класс MainWindow (см. табл. 4.5, листинг А.7).

Таблица 4.5 — Методы класса MainWindow

|  |  |
| --- | --- |
| Метод | Описание |
| MainWindow \*InitMainWindow(...); | Инициализировать окно. |
| MainWindow \*GetMainWindowHandle(void); | Получить дескриптор класса. |
| HINSTANCE GetWindowInstance(void); | Получить дескриптор процесса. |
| HWND GetWindowHandle(void); | Получить дескриптор окна. |
| int OnCommand (...); | Обработчик WM\_COMMAND. |
| int OnActivate(...); | Обработчик WM\_ACTIVATE. |

Инициализация окна выполняется один раз — при вызове InitMainWindow (см. листинг А.8). При повторном вызове этого метода возвращается дескриптор существующего экземпляра класса. Если при запуске программы другой экземпляр уже запущен, программа переключает пользователя в существующее окно и завершает свою работу. Это осуществляется реализацией паттерна Singleton[5].

### 4.2.2 Разработка интерфейса программы

Мне необходимо представить всю функциональность моего модуля в одном окне, реализованного с помощью WinAPI функций.

Практически все функции являются взаимозаменяемыми (см. табл. 4.6), поэтому удобно представить их кнопками checkbox, указав стиль обычной кнопки. До установки соединения, все остальные кнопки неактивны.

Таблица 4.6 — Кнопки главного окна

|  |  |
| --- | --- |
| Выбрана | Не выбрана |
| Установить соединение с роботом. | Прервать соединение с роботом. |
| Включить фонарь. | Выключить фонарь. |
| Включить подачу звукового сигнала. | Выключить подачу звукового сигнала. |

Единственное действие, требующее обычной кнопки — начало движения по маршруту. Для активации этой функции, пользователю предварительно необходимо указать маршрут.

Среди возможных маршрутов для движения выбор необходимого удобно осуществлять в окне listbox.

Скриншот моей программы представлен на рисунке С.1.

## Описание контрольного примера

Программа при выполнении каждого действия проверяет на наличие открытого подключения к библиотеке lejOS. В случае потери связи, пользователю выдается окно сообщения (на рис. С.2). После потери соединения программа вернется в исходное состояние — все кнопки, кроме «Установить соединение с роботом», будут деактивированы (на рис. С.1).

Для продолжения работы с роботом пользователю необходимо заново подключиться к роботу.

Во время движения по маршруту программа анализирует состояние датчиков расстояния и столкновения. При обнаружении неизвестного препятствия движение робота прекращается, а пользователю выдается окно с информацией об ошибке (на рис. С.3).

Необходимо учитывать, что при возникновении такой ситуации робот не возвращается на исходную позицию, а останавливается перед препятствием. Для продолжения движения необходимо составить новый маршрут, с помощью которого робот сможет преодолеть препятствие.

## 6. Необходимые аппаратные и программные средства

Требования к оборудованию:

1. Процессор с частотой от 1,6 ГГц.
2. 1024 МБ ОЗУ.
3. 20 МБ свободного места на диске.
4. Видеоадаптер с поддержкой DirectX 9 и разрешением от 800 x 600.
5. Дисковод DVD-ROM.
6. Bluetooth.

Для тестирования необходим LEGO-робот серии Mindstorms.

Требования к программному обеспечению:

1. Windows Vista / 7 (x86).
2. Microsoft Visual C++ 2010 Redistributable Package (x86).
3. Библиотека lejOS NXT.

## 7. Инструкция пользователя

В поставку программы входят файлы и папки, описанные в таблице 6.1.

Таблица 6.1 — Поставка программы

|  |  |
| --- | --- |
| Файл/папка | Описание |
| \modules | Папка с модулями управления роботом. |
| RobotControl.lib | Мой модуль управления LEGO-роботом. |
| \nxt | Папка для работы с lejOS NXT |
| StartConnection.bat | Системный файл |
| BTExecutor.class | Системный файл |
| Main.class | Системный файл |
| \routes | Папка с маршрутами для движения |
| Тестовый маршрут.txt | Тестовый маршрут движения. |
| LEGO.exe | Тестовое Windows-приложение. |

Для успешного использования программы необходимо выполнить следующие действия.

1. Поместить все файлы поставки в папку «C:\Program Files\LEGO».
2. В папку routes добавить необходимые маршруты для движения (по умолчанию там расположен тестовый маршрут).
3. Запустить LEGO.exe (на рис. С.1).
4. Подключиться к роботу, нажав кнопку «Установить соединение с роботом».
5. Выбрать маршрут для движения из списка имеющихся маршрутов.
6. Для начала движения нажать кнопку «Начать движение» (на рис. С.4).

Управление дополнительными возможностями модуля осуществляется с помощью кнопок «Включить фонарь», «Выключить фонарь», «Подать звуковой сигнал» и «Прекратить подачу звукового сигнала» (на рис. С.4, С.5).

## Выводы

В результате написания работы были созданы модуль управления LEGO-роботом и демонстрационное Windows-приложение для ее тестирования.

Разработанная библиотека предоставляет возможности выполнения движения по заданному маршруту и управления сигнальными устройствами.

В работе реализована ситуационная обработка возникновения неожиданных препятствий. Во время управления движением программа анализирует показания датчиков расстояния и столкновения. При несоответствии маршрута действительности система прекращает движение робота и уведомляет пользователя.

Модуль управления разработан с использованием приемов объектно-ориентированного проектирования. Это позволяет независимым разработчикам создавать собственные приложения, подключая мою библиотеку во время работы программы.

Разработанная библиотека может быть успешно использована в таких сферах применения роботов, как производство, исследования, обслуживание и работа с опасными веществами.

## Список ссылок

1. Каргин А. А. Введение в интеллектуальные машины. Книга 1. Интеллектуальные регуляторы. — Донецк, — Донецк: Норд-Пресс, ДонНУ, 2010. — 526 с., ил.
2. Microsoft. MSDN Library. — 2011. (<http://msdn.microsoft.com/en-us/library>). — 10.05.2011.
3. Мартин Р. К. Быстрая разработка программ. Принципы, примеры, практика.: Пер. с англ. — Москва, — М.: Издательский дом «Вильямс», 2004. — 752 с., ил.
4. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. Пер. с англ. — Санкт-Петербург, — СПб: Питер, 2001. — 386 с., ил.
5. BrickOS. — 2004. (<http://brickos.sourceforge.net>). — 11.05.2011.
6. NXT++. — 2011. (<http://nxtpp.clustur.com>). — 11.05.2011.
7. LesTat. — 2009. (<https://csrg.inf.utfsm.cl/twiki4/bin/view/ACS/LesTat>). — 11.05.2011.

|  |
| --- |
| Приложение A Фрагменты листинга |

Листинг A.1 — IRobotConsoleControl.h

\_\_interface IRobotConsoleControl

{

    void OpenConnection(void);

    void CloseConnection(void);

    HWND GetConsoleHandle(void);

    void PassRoute(IRobotRoute &route);

    wstring GetBattery(void);

};

Листинг A.2 — IRobotRoute.h

class IRobotRoute

{

public:

    IRobotRoute(double scale = 0.1) : scale(scale) {}

    virtual ~IRobotRoute(void) {}

    virtual double GetScale(void) { return scale; }

    typedef std::list<Point>::iterator RouteIterator;

    virtual RouteIterator Begin(void) = 0;

    virtual RouteIterator End(void)   = 0;

    virtual Point GetStartPos(void)   = 0;

    virtual Point GetOffset(void)     = 0;

private:

    double scale;

};

Листинг A.3 — IRobotHasLight.h

\_\_interface IRobotHasLight

{

    void LightOn(void);

    void LightOff(void);

};

Листинг A.4 — IRobotHasSound.h

\_\_interface IRobotHasSound

{

void SoundOn(void);

void SoundOff(void);

};

Листинг A.5 — RobotControl.h

class RobotControl :

    public IRobotConsoleControl,

public IRobotHasLight,

public IRobotHasSound

{

public:

    static RobotControl \*InitInstance(void);

    ~RobotControl(void) {}

    void OpenConnection(void);

    void CloseConnection(void);

    void PassRoute(IRobotRoute &route);

    wstring GetBattery(void);

    HWND GetConsoleHandle(void);

    void LightOn(void);

    void LightOff(void);

    void SoundOn(void);

    void SoundOff(void);

};

Листинг A.6 — Структура Action

struct Action

{

    Action(int action = 0, double time = 0)

: action(action), time(time) {}

int action;

    double time;

    enum {LEFT, RIGHT, START, STOP};

};

Листинг A.7 — MainWindow.h

class MainWindow

{

public:

    static MainWindow \*InitMainWindow(

LRESULT (CALLBACK \*)(HWND, UINT, UINT, LONG),

HINSTANCE hInstance, int nCmdShow);

    static MainWindow \*GetMainWindowHandle(void) { return instance; }

   HINSTANCE GetWindowInstance(void) { return hInstance; }

    HWND GetWindowHandle(void) { return hWnd; }

    ~MainWindow(void) {}

    int OnCommand (HWND hWnd, UINT Message, UINT wParam, LONG lParam);

    int OnActivate(HWND hWnd, UINT Message, UINT wParam, LONG lParam);

    typedef pair<int, HWND> ID\_HWND;

};

Листинг A.8 — Метод InitMainWindow класса MainWindow

void MainWindow::InitMainWindow(

LRESULT (CALLBACK \*WndProc)

(HWND hWnd, UINT Message, UINT wParam, LONG lParam),

                   HINSTANCE hInstance, int nCmdShow)

{

    if(!instance)

    {

        instance = new MainWindow();

        instance -> hInstance = hInstance;

        instance -> WndProc   = WndProc;

        instance -> className = L"LEGO-robot";

        instance -> CheckIsAlreadyOpened();

        instance -> RegisterWindowClass();

        instance -> InitGUI(nCmdShow);

    }

}

|  |
| --- |
| Приложение B Диаграмма классов  D:\Документы\Универ\Dropbox\private\Курсовая\Documents\RobotControlDiagram.jpg |

## Приложение С

Экранные формы

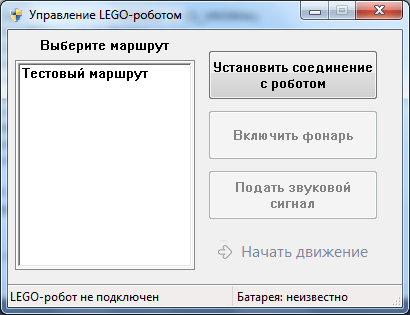


Рисунок С.1 — Запуск программы

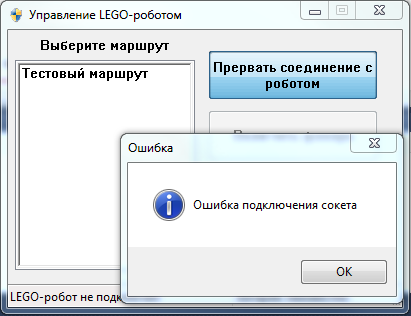


Рисунок C.2 — Неудачное подключение к роботу

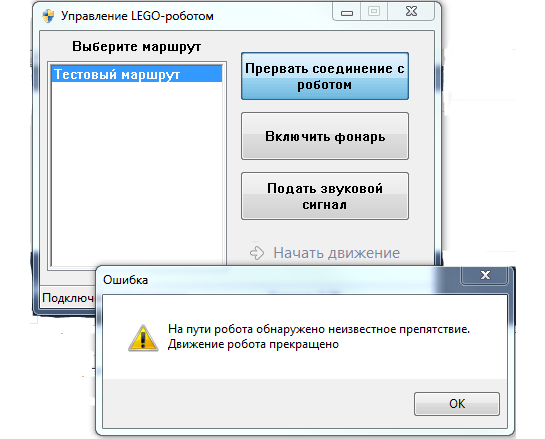


Рисунок C.3 — Обнаружение неожиданного препятствия на пути робота

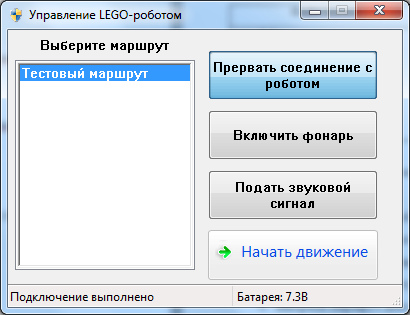


Рисунок C.4 — Успешное подключение к роботу

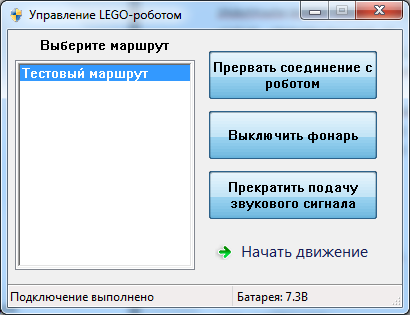


Рисунок C.5 — Демонстрация дополнительных кнопок