ГЛАВА 9. СОЗДАНИЕ МОДЕЛИ РОБОТА В СРЕДЕ ВИРТУАЛИЗАЦИИ

Робототехника – прикладная наука, занимающаяся разработкой автоматизированных технических систем и являющаяся важнейшей технической основой интенсификации производства [21]. Слово «робото-техника» было впервые использовано в печати американским писателем-фантастом Айзеком Азимовым в научно-фантастическом рассказе «Лжец», опубликованном в 1941 г.

В основу слова «робототехника» легло слово «робот», которое было придумано чешским писателем Карелом Чапеком и его братом и впервые использовано в научно-фантастической пьесе «Россумские универсальные роботы» в 1920 году [7]. До появления промышленных роботов считалось, что роботы должны выглядеть подобно людям.

В данный момент развития научно-технического прогресса робототехника вносит существенный вклад в повышение производительности труда и качества выпускаемой продукции. Роботы все активнее применяются в различных областях, начиная с ухода за больными и заканчивая военной сферой.

9.1. Краткие теоретические сведения

Новые сферы применения роботов требуют новых технологий и высококвалифицированных кадров. Однако ввод в эксплуатацию робота, а так же обучение специалистов (в том числе и студентов) в условиях реального производства может оказаться весьма затратным с точки зрения финансов и времени. Для ознакомления студентов с роботами и обучения технологическому программированию рациональнее использовать учебные модели роботов или трехмерные виртуальные модели, с системой управления соответствующей системе реального робота (рис. 9.1).

Технологии моделирования роботов имеют ряд преимуществ и особенностей, таких как:

– модель имеет все степени свободы присущие реальному объекту;

– предусмотрен режим восстановления после выхода из строя;

– среда симуляции позволяет рассматривать сцену в различных ракурсах в трёхмерном пространстве;

– можно изменять уровень освещения сцены, выбрать раскраску, включать и отключать объекты, которые в данный момент мешают увидеть перемещение модели.

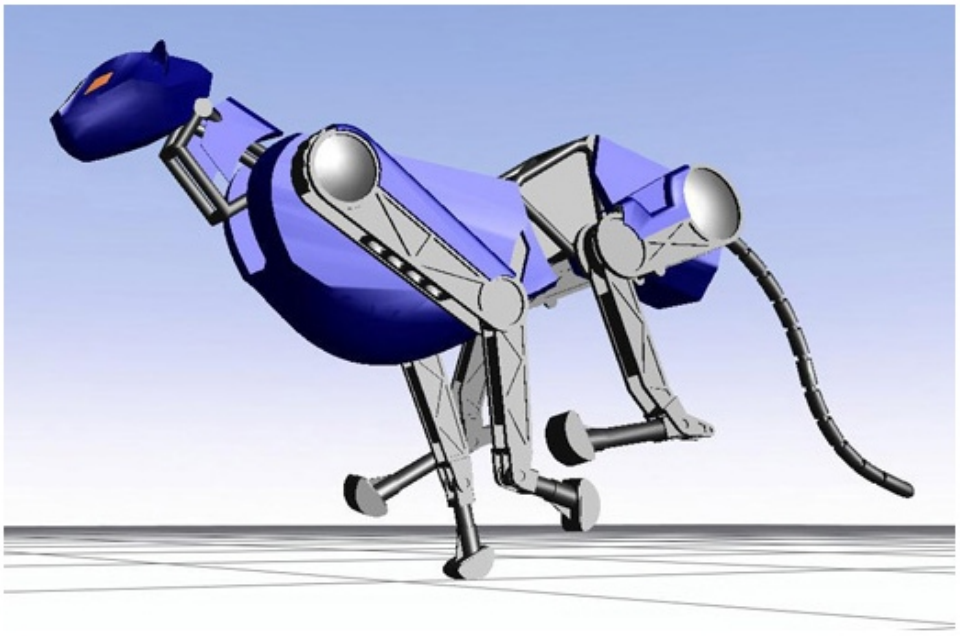


Рисунок 9.1 – Трехмерная виртуальная модель (иллюстрация с сайта компании Boston Dynamics)

Учитывая описанные выше преимущества, в данной главе будут рассмотрена методика создания виртуальных моделей мобильных роботов, которые наряду с манипуляционными роботами относятся к важнейшим классам **роботов широкого назначения** [7].

Манипуляционный робот – автоматическая машина (стационарная или передвижная), состоящая из исполнительного устройства в виде манипулятора, имеющего несколько степеней подвижности, и устройства программного управления, которая служит для выполнения в производственном процессе двигательных и управляющих функций. Такие роботы производятся в напольном, подвесном и портальном исполнениях. Получили наибольшее распространение в машиностроительных и приборостроительных отраслях [21].

Мобильный робот – автоматическая машина, в которой имеется движущееся шасси с автоматически управляемыми приводами. Такие роботы могут быть колёсными, шагающими и гусеничными (существуют также ползающие, плавающие и летающие мобильные робототехнические системы) [21].

Среда Microsoft Robotics Developer Studio R4 поддерживает несколько роботизированных платформ сразу после установки, т. е. содержит основные службы, используемые для управления системой привода и встроенными контактными датчиками робота. Microsoft RDS поддерживает следующие модели роботов: LEGO Mindstorms NXT, iRobot Create, Pioneer 3Dx, Parallax Boe-Bot, CoroBot, KUKA LBR3 и другие.

Одним из самых простых роботов, которых вы сможете собрать из базового комплекта конструктора Lego NXT Mindstorms, является **TriBot**. Это трехколесный робот, который может быть оснащен различными датчиками (рис. 9.2).

Рисунок 9.2 – Внешний вид робота Lego NXT Tribot (иллюстрация с сайта http://walkuere.tistory.com/)

iRobot Create – робот, разработанный компанией iRobot на базе робота-пылесоса Roomba (рис. 4.4). Create предназначен для разработчиков роботов, позволяет программировать поведение робота. Одним из отличий от робота Roomba является замена вакуумного оборудования (применяемого для осуществления уборки) на специальный отсек для полезной нагрузки,

снабженный 25-контактным разъёмом, используемым для связи с устанавливаемым в отсек оборудованием путём передачи цифровых или аналоговых сигналов [7].

Робот **Pioneer 3DX**, созданный компанией Active Media Robots оборудован панорамной камерой, колесной платформой, электроприводами, ультразвуковым сонаром и лазерным дальномером для ориентации и обнаружения препятствий (рис. 9.3).

Рисунок 9.3 – Внешний вид мобильного робота Pioneer 3 DX с лазерным дальномером (иллюстрация с сайта http://www.mobilerobots.com/)

Робот **Boe-Bot** – роботизированная платформа фирмы Parallax на базе модуля управления BASIC Stamp включает в себя алюминиевую платформу, сервоприводы непрерывного вращения, колеса, датчики. На платформе имеется макетная плата, что расширяет возможность подключения дополнительных компонентов (рис. 9.4).

Робот **Corobot** компаниии CoroWare оборудован компьютером на базе x86-совместимого процессора VIA с частотой 1,2 ГГц, датчиками и цветной камерой с разрешением 640 × 480 пикселей. В конфигурацию компьютера входит 512 Мб оперативной памяти и жесткий диск объемом 20 Гб. Робот может быть оборудован рукой с захватом, имеющей четыре степени свободы (рис. 9.5).

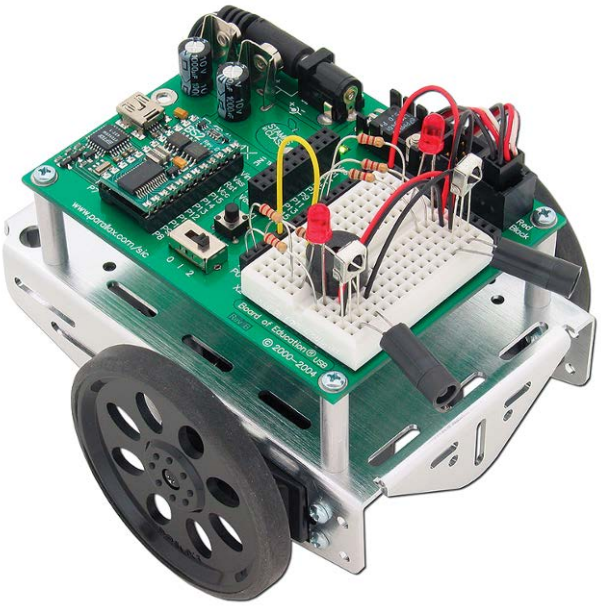


Рисунок 9.4 – Внешний вид робота Parallax Boe-Bot (иллюстрация с сайта http://www.parallax.com/)

Рисунок 9.5 – Внешний вид робота Corobot (иллюстрация с сайта http://www.robotcombat.com/)

9.2. Методические рекомендации

9.2.1. Методика создания объекта типа «Boe-Bot»

1. Создайте проект содержащий сцену по методике 7.2.2.

2. Добавьте класс, который является производным класса DifferentialDriveEntity. Для этого можно использовать следующую программу, которая описывает поведение робота Boe-Bot при движении в моделируемой среде [28].

```
   [DataContract]
    public class BoeBot : DifferentialDriveEntity
    {
        Port<EntityContactNotification> _notifications =
               new Port<EntityContactNotification>();

        // Конструктор класса по умолчанию, используется для создания
объекта из XML
        public BoeBot() { }

        // Собственный конструктор, который используется для создания
объекта из кода

        public BoeBot(Vector3 initialPos)
        {
            MASS = 0.454f; // масса в килограммах
            // настройки по умолчанию для шасси BoeBot
            CHASSIS_DIMENSIONS = new Vector3(0.09f, // метров в ширину
                                             0.09f,  // метров в высоту
                                             0.13f); // метров в длину
            FRONT_WHEEL_MASS = 0.01f;
            CHASSIS_CLEARANCE = 0.015f;
            FRONT_WHEEL_RADIUS = 0.025f;
            CASTER_WHEEL_RADIUS = 0.0125f;
            FRONT_WHEEL_WIDTH = 0.01f;
            CASTER_WHEEL_WIDTH = 0.008f;
            FRONT_AXLE_DEPTH_OFFSET = 0.01f; // расстояние от центра
робота

            base.State.Name = "BoeBot";
            base.State.MassDensity.Mass = MASS;
            base.State.Pose.Position = initialPos;

            // позиция шасси робота
            BoxShapeProperties motorBaseDesc =
                new BoxShapeProperties("BoeBot Body", MASS,
                    new Pose(new Vector3(
                    0, // Используем 0 как смещение по оси X
                    CHASSIS_CLEARANCE + CHASSIS_DIMENSIONS.Y / 2,
                    0.03f)), // вторичное смещение по оси Z (глубина)
                    CHASSIS_DIMENSIONS);

            motorBaseDesc.Material =
                 new MaterialProperties("high friction", 0.0f, 1.0f,
20.0f);
            motorBaseDesc.Name = "Chassis";
            ChassisShape = new BoxShape(motorBaseDesc);
```

```
            // заднее колесо
            CASTER_WHEEL_POSITION = new Vector3(0, // центр шасси
                 CASTER_WHEEL_RADIUS,              // расстояние от земли
                 CHASSIS_DIMENSIONS.Z / 2);        // и задней части
робота

            RIGHT_FRONT_WHEEL_POSITION = new Vector3(
                 +CHASSIS_DIMENSIONS.X / 2,// слева от центра
                 FRONT_WHEEL_RADIUS,       // расстояние от низа оси
                 FRONT_AXLE_DEPTH_OFFSET); // расстояние от центра по оси
Z

            LEFT_FRONT_WHEEL_POSITION = new Vector3(
                 -CHASSIS_DIMENSIONS.X / 2,// справа от центра
                 FRONT_WHEEL_RADIUS,       // расстояние от низа оси
                 FRONT_AXLE_DEPTH_OFFSET); // расстояние от центра по оси
Z

            MotorTorqueScaling = 30;

            // Устанавливаем Mesh объект по умолчанию
            State.Assets.Mesh = "boe-bot.bos";
    }
```

Конструктор для класса BoeBot используется для установки значений нескольких переменных, определенных в классе DifferentialDriveEntity. Эти параметры были получены путем взвешивания и измерения реального робота. Например, параметр Mass имеет значение 0,454, что соответствует массе в килограммах. Шасси Boe-Bot определяется в терминах ширины, длины и высоты.

Положение модели робота Boe-Bot определяется с помощью набора координат, которые передаются после того, как объект создан. Эти координаты соответствуют точкам осей координат X, Y и Z.

Конструктор BoeBot также определяет положение шасси и колес внутри объекта. Класс DifferentialDriveSystem исходит из предположения, что робот имеет два основных колеса и одно колесо, которое используется для придания устойчивости. Питание подается на левый и правый двигатели, которые управляют главными колесами. Разница в напряжении, подаваемом на каждое из колес, определяет, движется ли робот вперед, назад, влево или вправо. Таким же методом приводится в движение и реальный робот.

3. Добавьте объект Boe-Bot. Для этого введите код метода AddBoeBot:

```
BoeBot boeBot = new BoeBot(new Vector3(0f, 2f, 0));
boeBot.State.Name = "SimulatedBoeBot";

// Запускаем симулированный Boe-Bot Drive сервис
CreateService(drive.Contract.Identifier,
   Microsoft.Robotics.Simulation.Partners.CreateEntityPartner(
      "http://localhost/" + boeBot.State.Name));

SimulationEngine.GlobalInstancePort.Insert(boeBot);
```

Метод AddBoeBot принимает входной параметр Vector3, который используется для отображения местоположения робота. Это положение передается конструктору объекта BoeBot, который был создан ранее. Метод AddBoeBot также запускает в качестве объекта-партнера службу движения модели Boe-Bot.

4. В заголовке добавьте строку:

```
using drive = Microsoft.Robotics.Services.Drive.Proxy;
```

5. Добавьте ссылку на библиотеку RoboticsCommon.proxy.

9.2.2. Методика преобразования графическихфайлов

Для создания 3D модели робота можно воспользоваться практически любым трехмерным редактором (рис. 9.6). Важно помнить, что материалы, которые используются для закраски физических объектов, и все объекты модели должны ассоциироваться с одним или несколькими материалами. Поэтому когда вы создаете файл obj, желательно добавить дополнительный файл материалов с расширением .mtl. Он также может содержать файл картинки, которая будет использоваться в качестве текстуры [28].

Пакет Microsoft RDS позволяет считать 3D-модель из obj файла. Программа командной строки Obj2bos.exe может преобразовывать файлы с расширением .obj в оптимизированные двоичные файлы bos, которые загружаются гораздо быстрей, чем файлы obj. Файл obj и все связанные с ним файлы должны располагаться в папке /store/media, относящейся к установленной программе SDS.

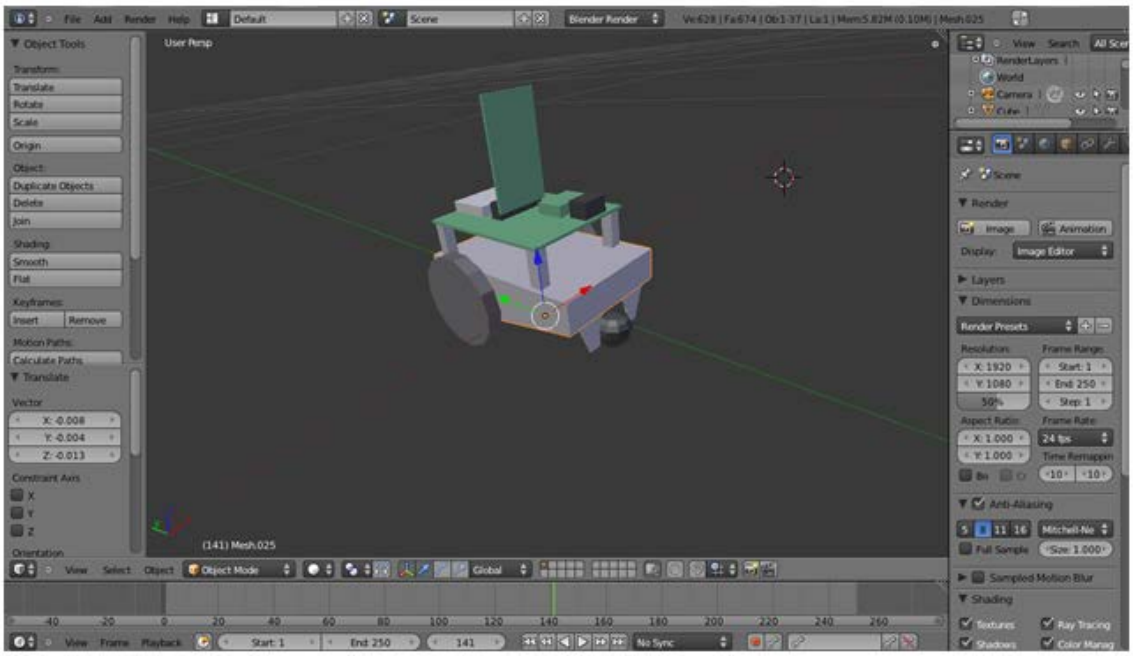


Рисунок 9.6 – Создание модели робота в 3D редакторе Blender

1. Поместите файл boe-bot.obj и связанные файлы (Aluminum.png, Boe-bot.mtl, Boe-bot.obj, BoebotWheel.png и Pcb.png) в указанную папку.

2. Откройте командную строку и введите:

```
Obj2Bos.exe /c:"store\media\boe-bot.obj"
```

3. Программа конвертирования создаст файл с именем Boe-bot.bos, который будет находиться в упомянутой папке. Оставьте файл здесь, он понадобится нам позже.

9.2.3. Методика создания манифеста

Большинство служб взаимодействуют с другими службами с целью получения доступа к данным и функциям, необходимым данной службе. Такие партнерские службы указываются в верхней части класса реализации с атрибутом Partner. Например, служба моделирования Boe-Bot взаимодействует со службой обработчика модели, которая уже была добавлена в проект.

Службы DSS используют файл, называемый манифестом, в котором перечисляются партнерские службы, ассоциированные с основной службой. Он указывает среде выполнения RDS, каким образом базовая служба должна загружать партнерские службы и взаимодействовать с ними.

Манифест представляет собой файл XML, который можно редактировать в любом текстовом редакторе, но лучше использовать для этого редактор манифестов DSS (DSS Manifest Editor), входящий в комплект Microsoft RDS. Редактор манифестов DSS представляет собой графическое средство, позволяющее размещать партнерские службы на рабочей области методом перетаскивания. Перед тем как использовать редактор манифестов DSS, необходимо правильно скомпилировать основную службу. В результате этого будут созданы файлы сборки, используемые редактором манифестов DSS.

1. Проведите загрузку редактора манифестов DSS. В левом окне появится полный список доступных служб (рис. 9.7).

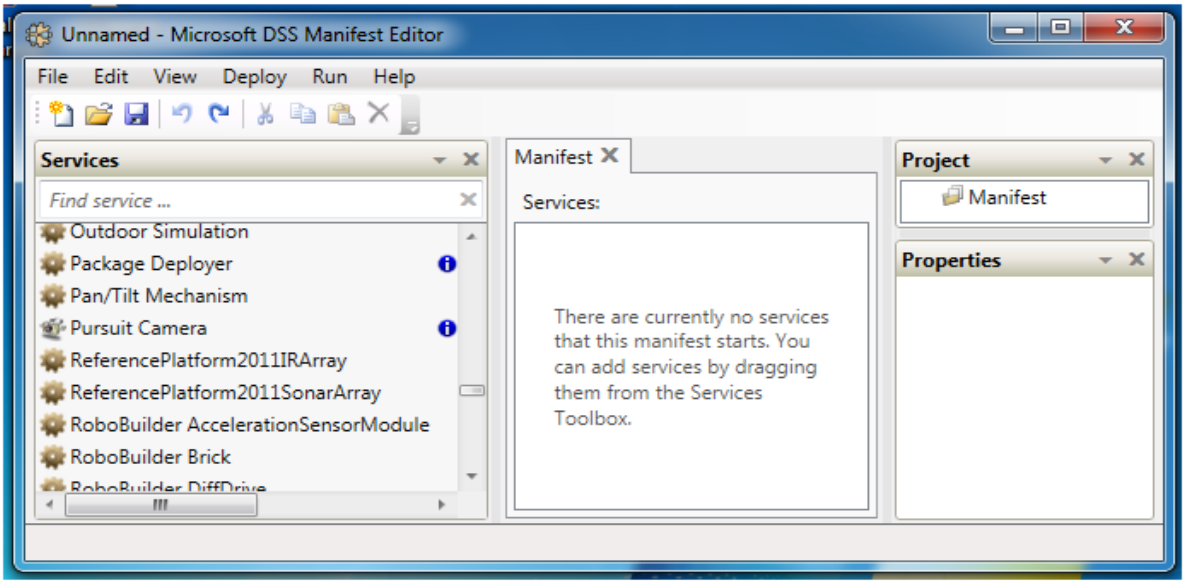


Рисунок 9.7 – Окно редактора манифестов DSS

2. В начале работы вам необходимо найти в списке доступных служб основную службу и перетащить ее объект в рабочую область. Сделаете это со службой моделирования Boe-Bot, в рабочей области появится узел для службы RTS_Lab9 и вспомогательный узел для службы механизма моделирования.

Установленая среда Microsoft RDS имеет несколько служб, которые находятся в папке \samples. Одна из них, SimpleDashboard использовалась нами в 6 главе в качестве панели управления.

Служба SimpleDashboard позволяет запускать другие службы, используемые для управления специфическими функциями вашего робота.

Например, код службы RTS_Lab9 заполняет сцену моделирования объектами, один из которых – модель робота. Кода, который управляет движением робота, в данной службе нет. Этот код берется из службы Drive.

3. Чтобы загрузить и использовать простой пульт управления, добавьте его в манифест, поместив объект SimpleDashboard в список служб и перетащив этот объект в нижнюю часть рабочей области (рис. 9.8).

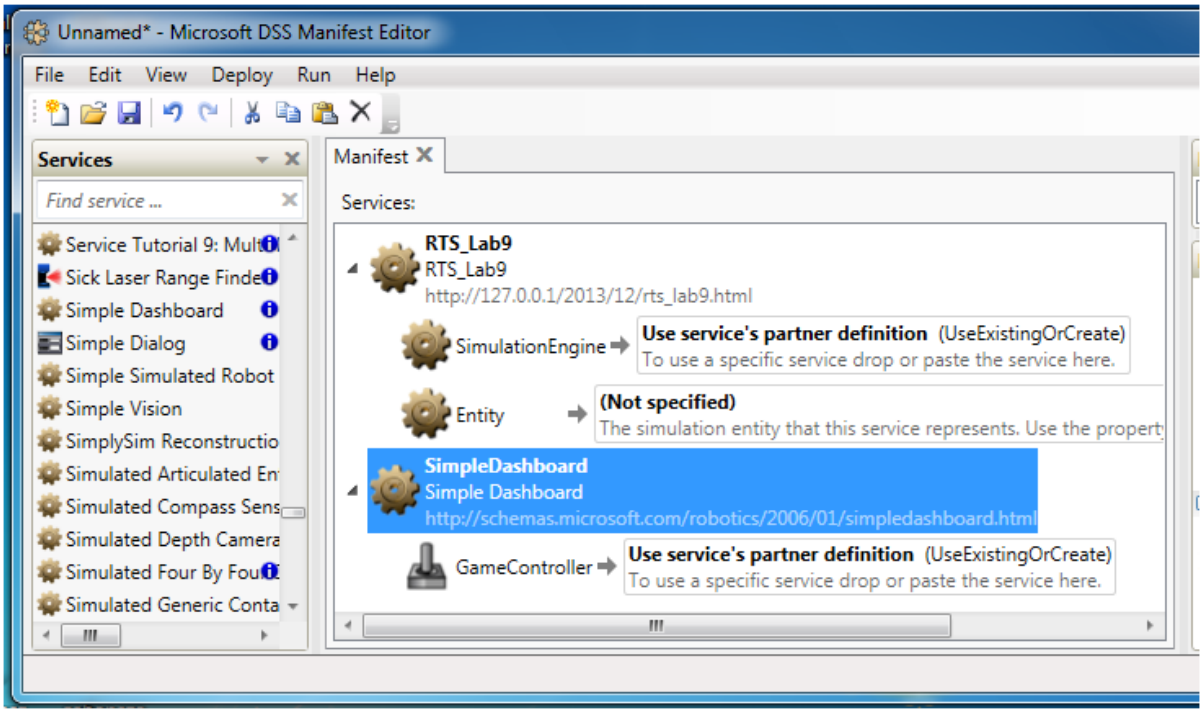


Рисунок 9.8 – Окончательный вид манифеста в редакторе

4. Сохраните манифест в папке проекта службы RTS_Lab9, под именем RTS_Lab9.manifest.xml.

9.3. Лабораторная работа 9. Основы разработки модели робота в среде симуляции Visual Simulation Environment

9.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков создания и конфигурирования моделей в среде симуляции VSE Microsoft RDS с применением Microsoft Visual Studio.

Содержание

1. Создание модели робота в среде Microsoft Visual Studio.
2. Преобразование графических файлов.
3. Создание манифеста в редакторе DSS Manifest Editor.

9.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4.
3. Установленная среда XNA Game Studio 4.
4. Установленная интегрированная среда разработки программного обеспечения Visual Studio 2010 Express.

9.3.3. Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Создайте сцену и модель робота средствами Visual Studio согласно методике 9.2.1 и с учетом задания в табл. 9.1.
2. При необходимости создайте 3D модель робота согласно методике 9.2.2 и с учетом задания в табл. 9.1.
3. Разработайте манифест согласно методике 9.2.3.
4. Запустите симуляцию и прокомментируйте результат.

9.4. Контрольные вопросы

1. Какие роботизированные платформы поддерживает среда Microsoft Robotics Developer Studio R4?
2. Каким образом получены значения переменных, определенных в классе DifferentialDriveEntity?

Таблица 9.1. Варианты заданий для самостоятельного выполнения

№ вар.	Тип робота	№ вар.	Тип робота
1.	Lego NXT Tribot	2.	Pioneer 3 DX
3.	Corobot	4.	iRobot Create
5.	Lego NXT Tribot	6.	Pioneer 3 DX
7.	Corobot	8.	iRobot Create
9.	Lego NXT Tribot	10.	Pioneer 3 DX
11.	Corobot	12.	iRobot Create

3. В какой файл преобразуюся файлы с расширением .obj в программе Obj2bos.exe?

4. Прокомментируйте особенности работы в редакторе манифестов DSS.

5. Каким образом службы DSS используют файл, называемый манифестом?

ГЛАВА 10. УПРАВЛЕНИЕ РОБОТОМ ПРИ ДВИЖЕНИИ ПО ЛИНИИ С ИСПОЛЬЗОВАНИЕМ ДАТЧИКОВ

10.1. Краткие теоретические сведения

Обычно в состав робота входит несколько датчиков (сенсоров), которые используются для сбора информации об окружающей среде. На основе полученной информации принимается решение о поведении робота. Рассмотрим наиболее часто встречающиеся сенсоры [1].

Контактный сенсор, бампер (contact sensor) – переключатель, который позволяет детектировать соприкосновение робота с некоторым предметом (рис. 10.1).

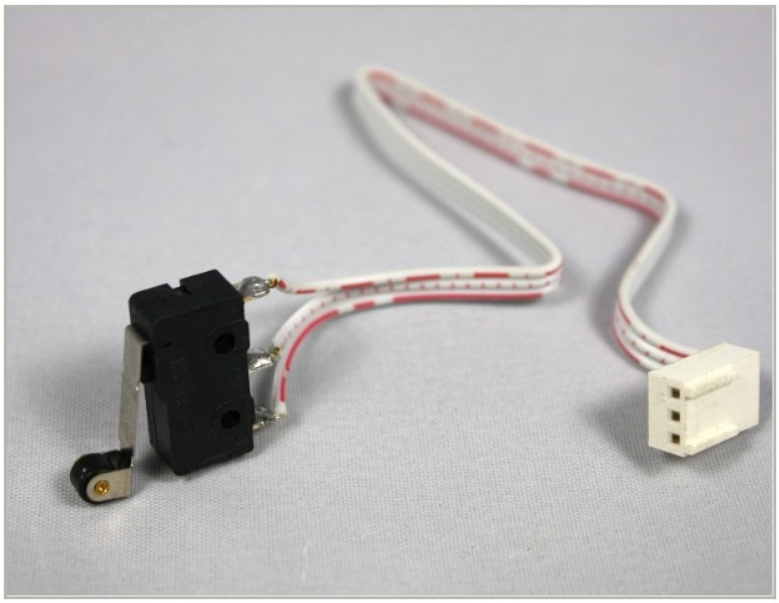


Рисунок 10.1 – Внешний вид контактного сенсора (иллюстрация с сайта http://www.hobbyist.co.nz/)

Сенсор расстояния. В зависимости от типа используемого сигнала различают: инфракрасный сенсор, сонар, лазерный дальномер. Принцип работы сенсора расстояния основан на отправке сигнала (инфракрасный свет, ультразвуковой сигнал, лазерный импульс) и получении отраженного сигнала. При этом замеряется время задержки, которое определяет расстояние до препятствия.

Инфракрасный сенсор (infrared sensor) является наиболее дешевым, однако не всегда значения, считываемые с такого сенсора, непосредственно соответствуют расстоянию от робота до препятствия. Дальность сенсора составляет от 5 до 50 сантиметров.

Пассивный инфракрасный сенсор (passive infrared sensor) – PIR-сенсор – принимает отраженные инфракрасные сигналы (рис. 10.2). Активный инфракрасный сенсор (active infrared sensor) – отправляет импульсы инфракрасного света фиксированной частоты и регистрирует время прихода отраженного сигнала.

Рисунок 10.2 – Внешний вид PIR-сенсора SEN-08630 (иллюстрация с сайта https://www.sparkfun.com/)

Рисунок 10.3 – Устройство активного инфракрасного сенсора (иллюстрация с сайта http://www.made-in-china.com/)

Сонар использует технологию SOund NAvigation and Ranging (звук, передвижение и расположение, сокращенно – SONAR) разработанную в 1940-х годах для слежения за подводными лодками. Дальность измерения сонара составляет от нескольких сантиметров до нескольких метров, точность – около двух сантиметров.

Рисунок 10.4 – Внешний вид ультразвукового сонара LV-MAXSONAR-EZ1 (иллюстрация с сайта http://www.terraelectronica.ru/)

Технология сонара использует звуковые волны, которые распространяются от источника в виде конуса. В результате сенсор позволяет получить расстояние только до ближайшего объекта, находящегося в секторе в несколько градусов. Как правило, используют несколько сонаров, расположенных по окружности (обычно 12 или 24 сенсора). Они не должны располагаться слишком близко друг к другу, чтобы избежать взаимного влияния.

Лазерный дальномер (laser rangefinder) использует лазер малой мощности, обычно красного цвета. В современных лазерных дальномерах применяется инфракрасный диапазон. Если лазерный дальномер использует сканирующий луч и поворотные зеркала, то это позволяет выполнить сканирование пространства по широкой дуге (рис. 9.3). Обычно угол между лучами фиксированный и составляет один градус. Диапазон действия лазерного дальномера составляет от 8 до 100 м и поле зрения от 100 до 180 градусов.

Навигация. Для определения позиции робота в пространстве используется спутниковая навигационная система (рис. 10.5). В настоящее время работают или находятся в стадии развёртывания следующие системы спутниковой навигации [7]:

Рисунок 10.5 – Внешний вид GPS/GLONASS сенсора (иллюстрация с сайта http://www.mynewsdesk.com/)

1. GPS (Global Positioning System). Принадлежит министерству обороны США. Космический сегмент состоит из 32 спутников, вращающихся на средней орбите Земли. Устройства с GPS являются самыми распространёнными в мире.

2. ГЛОНАСС (Глобальная навигационная спутниковая система). Принадлежит министерству обороны России. Основой системы являются 24 спутника, движущиеся над поверхностью Земли в трёх орбитальных плоскостях.

3. Бэйдоу. Развёртываемая Китаем система навигации изначально предназначалась для использования только в этой стране. В 2012 году включала в себя 16 спутников, расположенных на геостационарной орбите и обеспечивала определение географических координат в Китае и на соседних территориях. Планируется, что на полную мощность система выйдет к 2020 году и космический сегмент будет состоять из 5 спутников на геостационарной орбите и 30 спутников на орбитах, отличных от геостационарной. При этом система сможет работать как глобальная.

4. Галилео (Galileo) – совместный проект спутниковой системы навигации Европейского союза и Европейского космического агентства. Ожидается, что «Галилео» войдёт в строй в 2014 – 2016 годах, когда на орбиту будут выведены все 30 запланированных спутников (27 операционных и 3 резервных). В данный момент космический сегмент состоит из 4 спутников.

Компас (compass sensor). Ориентация робота в пространстве – одна из основных частей информации, необходимых для навигации. Если робот дезориентирован в результате столкновения с препятствием или снижения скорости колес, можно использовать компас, который предоставляет абсолютные значения угла, т. е. ошибка измерения не накапливается со временем (рис. 10.6).

Рисунок 10.6 – Внешний вид модуля компаса HMC6352 (иллюстрация с сайта http://www.hobbytronics.co.uk/)

Сенсор скорости (speed sensor). Данные сенсоры могут использоваться для определения скорости вращения колеса (wheel speed sensor), скорости транспортного средства (vehicle speed sensor), скорости вращения двигателя (motor speed sensor, рис. 10.7).

Рисунок 10.7 – Внешний вид сенсора скорости вращения двигателя компании wRobot (иллюстрация с сайта http://emartee.com/)

Сенсор угла наклона (инклинометр) используют для определения угла наклона поверхности, по которой движется робот. Шагающий робот может использовать такой сенсор для определения критического угла падения.

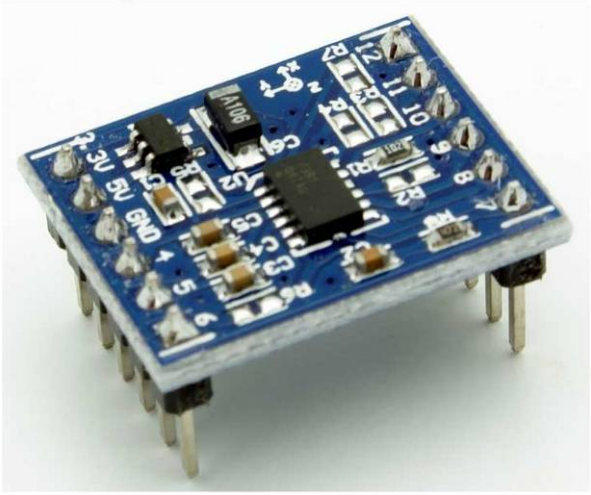


Рисунок 10.8 – Внешний вид модуля Arduino MMA7361 с датчиком угла наклона и акселерометром (иллюстрация с сайта http://www.chinics.com/)

Звуковой сенсор (acoustic sensor). Одним из примеров звукового сенсора является микрофон, при этом для получения достоверной информации необходима высокая частота дискретизации.

Рисунок 10.9 – Внешний вид модуля Arduino со звуковым сенсором (иллюстрация с сайта http://www.chinics.com/)

Счетчик оборотов колеса, одометр (odometer) используется для определения расстояния, пройденного роботом. Точность таких сенсоров определяется по количеству измерений, выполняемых на один оборот колеса.

Сенсор интенсивности света (light sensor). Определяет количества света, которое попадает на сенсор (рис. 10.10). Сенсоры такого типа могут с определенной точностью выступать в роли датчиков цвета. Они не могут

использоваться для реализации машинного зрения, так как не создают изображение.

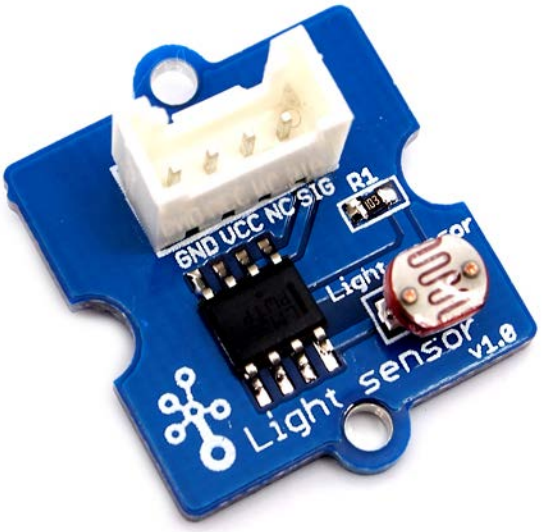


Рисунок 10.10 – Внешний вид сенсора инсвета Grove (иллюстрация с сайта http://electronica.bashel.ru/)

Ниже приведен перечень манифестов с моделями мобильных роботов, присутствующий в среде Microsoft RDS и сенсоры, которыми они оснащены [1]. В ряде случаев состав сенсоров робота зависит от сцены, в которой он находится.

1. TestWallSensorSim.Manifest.xml (робот – iRobot). С помощью стандартных сервисов доступ к сенсорам робота в этом манифесте получить нельзя. Дополнительный активный инфракрасный сенсор расположен в передней части робота и немного смещен вправо.

2. SimpleVisionSim.Manifest.xml (робот – Pioneer 3DX). Сенсоры: веб-камера, лазерный дальномер и два бампера (один в передней части робота, другой – в задней части).

3. IRobotCreate.Simulation.Manifest.xml (робот – iRobot). Сенсор – бампер, расположенный в передней части робота.

4. MobileRobots.P3DX.Simulation.Manifest.xml (робот – P3DX). Сенсоры: лазерный дальномер, веб-камера, бампер.

5. demoscene.Manifest.xml (робот – P3DX). Сенсоры: лазерный дальномер, веб-камера, бампер.

6. ApartmentScene.Manifest.xml (робот – шаблон робота). Сенсоры: веб-камера.

7. Lego.NXT.Tribot.Simulation.Manifest.xml (робот – Lego NXT). Сенсоры: бампер, расположенный в передней части робота.

Рассмотрим особенности моделирования сенсоров в среде Microsoft RDS R4. Сцена для проведения экспериментов показана на рис. 10.11 [1]. Робот в сцену добавлен с помощью манифеста samples\config\EntityUI.manifest.xml (пункт меню File → Open Manifest...). Данный робот включает следующие сенсоры: веб-камера, сенсор цвета, компас, сенсор интенсивности света, сонар, лазерный дальномер, GPS сенсор, инфракрасный сенсор.

После формирования сцена была сохранена, обработана с помощью Manifest Editor и используется при разработке диаграмм, связанных с моделированием работы сенсоров. В настройках DifferentialDrive и сервисов, связанных с сенсорами, указывается манифест, полученный после сохранения сцены.

При использовании сенсоров необходимо учитывать, что сенсоры, измеряющие расстояние в вертикальной плоскости, находятся на некотором расстоянии от земли. Следовательно, они не могут отслеживать предметы, которые меньше высоты, на которой установлен датчик на роботе.

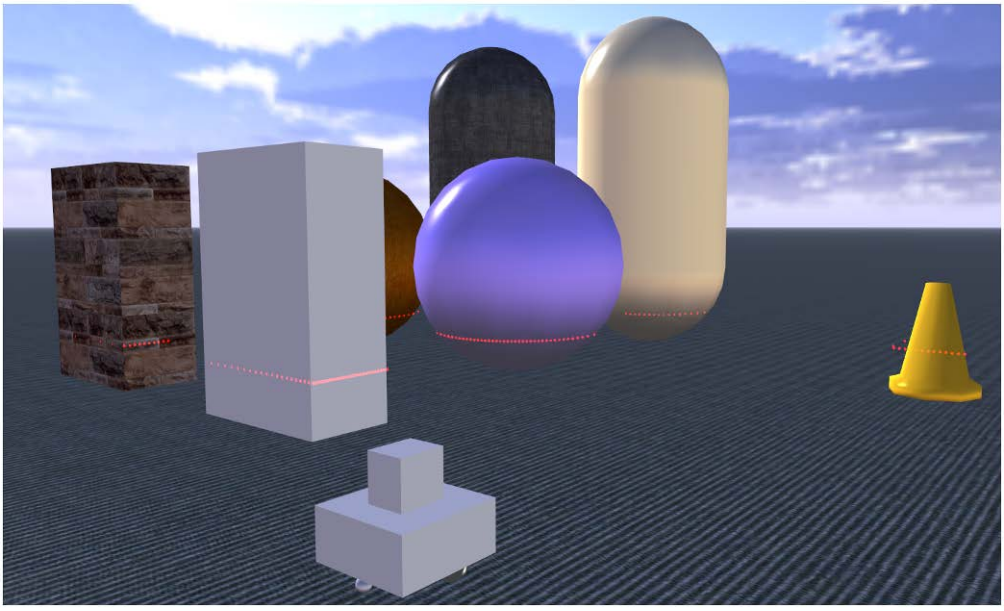


Рисунок 10.11 – Сцена с роботом оборудованным сенсорами

В симуляторе существует возможность визуализации показаний сенсоров интенсивности света, цвета, а также веб-камеры. Для этого нужно воспользоваться пунктом меню Camera. Показания каждого такого сенсора можно вывести в отдельном окне.

Существует два способа опроса сенсоров:

– с использованием уведомляющих сообщений сервиса, связанного с сенсором;

– с использованием запроса состояния с помощью таймера и действия Get.

Уведомляющие сообщения сервиса могут содержать дополнительную информацию о сенсоре, например его параметры. Необходимо отметить, что не каждый тип сенсоров можно опросить обоими способами.

Диаграмма чтения показаний сенсора интенсивности света (яркости объекта) показана на рис. 10.12. Для получения значений с сенсора использован сервис SimulatedPhotoCellBrightnessSensor.

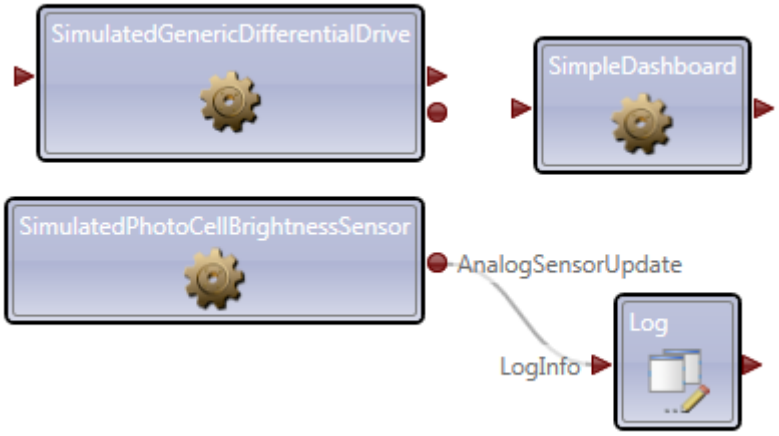


Рисунок 10.12 – Диаграмма чтения значений сенсора SimulatedBrightness Sensor

Результат работы сенсора записывается в поле NormalizedMeasurement (RawMeasurement) сообщения AnalogSensorUpdate (рис. 10.13). Обращение к данным сенсора также можно организовать с использованием запроса Get.

В случае если параметры сенсора недоступны для изменения в симуляторе, можно изменить их в XML-файле, в котором хранится описание сцены (при сохранении сцены на диск записываются два файла).

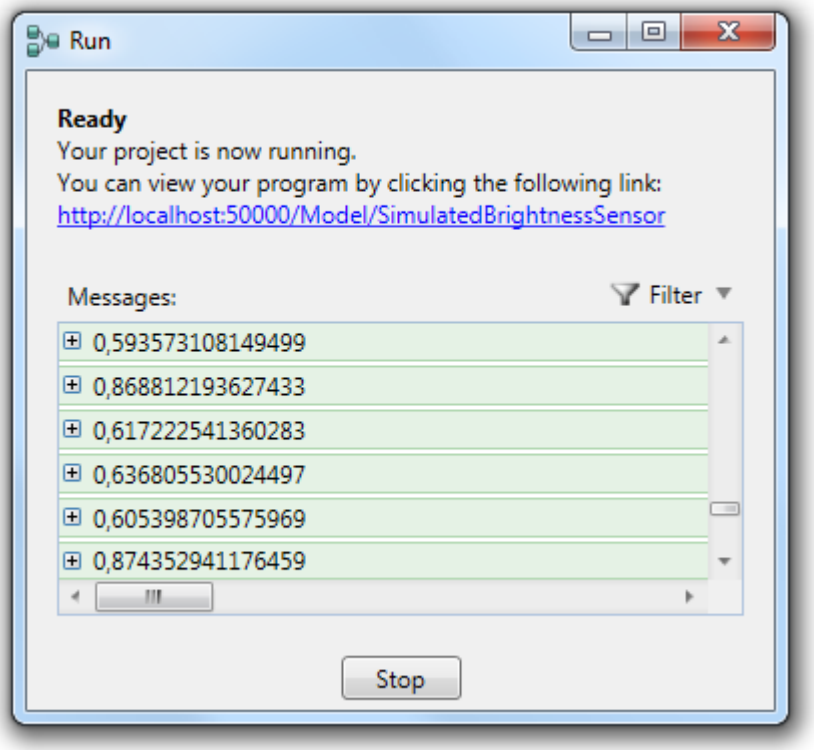


Рисунок 10.13 – Результат измерения яркости объекта

Для рассмотренной модели работы сенсора интенсивности света сцена сохранена с именем mytest. В первом файле (MyTest.Manifest.xml) хранится описание робота, а также перечень входящих в его состав сенсоров. Во втором (mytest.xml) – описание всех объектов, входящих в сцену и настройки каждого сенсора. Все сенсоры именуются так же, как и в симуляторе. Открыв данный файл несложно найти описание состояния нужного сенсора и выполнить изменение его параметров, которые в редакторе симулятора доступны только для чтения.

Одна из классических задач для мобильного робота – это движение по черной линии на белом поле с использованием датчиков интенсивности света или освещенности. Существует множество алгоритмов движения по линии, рассчитанных на различное число датчиков. Рассмотрим влияние количества сенсоров на способ и качество решения задачи [1].

Один сенсор

Одного сенсора достаточно для решения задачи отслеживания линии. В этом случае отслеживаются переходы от темного цвета к светлому и

наоборот, т. е. робот следует контуру линии. Предположим, что используется робот, использующий одно неуправляемое колесо (шар) и два управляемых (рис. 10.14).

Рисунок 10.14 – Внешний вид робота Lego NXT Tribot с одним датчиком освещенности (иллюстрация с сайта http://blogs.microsoft.co.il/)

В этом случае одно из колес будет активироваться, когда линия будет видна, другое колесо – когда линия будет пропадать из области действия сенсора. Данный подход работает при невысокой скорости передвижения робота. При высоких скоростях, если робот пересечет другую сторону линии, он может двигаться в противоположном направлении, если же он потеряет линию, то будет постоянно двигаться по окружности.

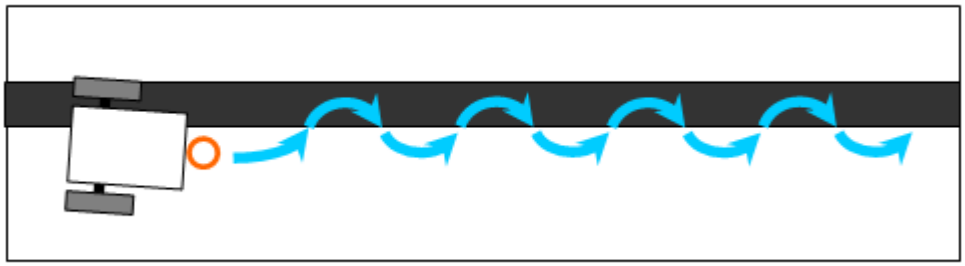


Рисунок 10.15 – Траектория движения робота вдоль линии с одним датчиком (иллюстрация с сайта http://nnxt.blogspot.ru/)

В бинарных терминах сенсор описывает два состояния:

– 0 – над линией;

– 1 – за пределами линии.

Рассмотрим особенности расположения датчика освещенности относительно управляемых колес робота [12]. Точки соприкосновения колес с трассой и центр датчика освещенности образуют между собой равнобедренный треугольник. В зависимости от расположения датчика вид треугольника может изменяться (рис. 10.16). Поскольку датчики находятся на разном расстоянии от колес, то при одном и том же угле смещения колес, отклонение датчиков будет разным (рис. 10.17).

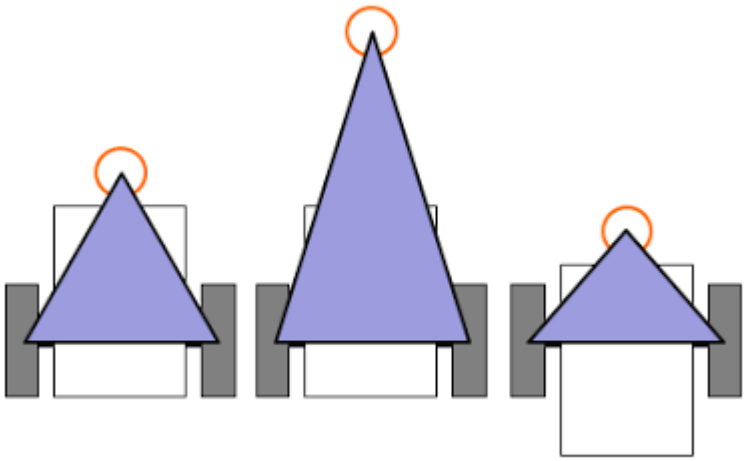


Рисунок 10.16 – Варианты расположения датчика освещенности относительно управляемых колес робота (иллюстрация с сайта http://nnxt.blogspot.ru/)

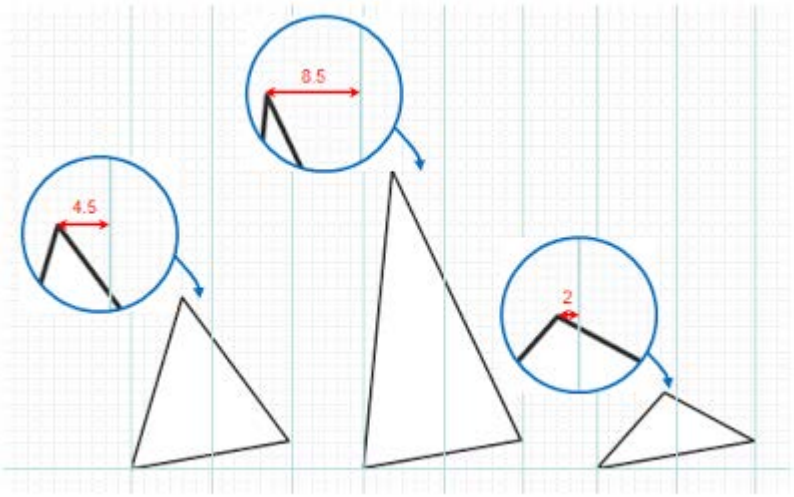


Рисунок 10.17 – Величины отклонения датчика (иллюстрация с сайта http://nnxt.blogspot.ru/)

У робота, датчик которого расположен далеко от оси колес, отклонение больше, чем у всех остальных. Это значит, что даже маленький угол поворота робота будет приводить к большому смещению датчика и большему изменению показаний датчика. Движение робота будет более зигзагообразным, угловатым. Это можно исправить либо уменьшением средней скорости, либо более частым опросом датчика. Также очевидно, что радиус кривизны, который робот может спокойно проходить, для данной конкретной модели значительно больше, чем у остальных – этому механизму будет сложно проходить крутые повороты.

У тележки, датчик которой расположен наиболее близко к оси колес, отклонение самое маленькое. Если в этот момент датчик находился на светлой части поля, то тележке нужно еще больше переместиться, чтобы достигнуть границы черной линии. Максимальное перемещение будет достигаться только при существенной разнице в скоростях вращения колес, т. е. при практически полной остановке одного из них. В итоге, при движении практически не будет фаз прямого движения – робот практически перпендикулярно будет переходить через границу линии.

На основе вышесказанного можно сделать вывод, что в общем случае при движении вдоль линии треугольник, образованный колесами и датчиком, должен стремиться к тому, чтобы быть равносторонним (левый вариант на рис. 10.16).

Два сенсора

В данном случае каждый сенсор управляет определенным мотором [1]. Существуют две модификации расположения двух сенсоров, при которых выполняется одно из двух правил: оба сенсора никогда не детектируют линию или оба сенсора детектируют линию (рис. 10.18). В первом случае, можно определить следующие варианты состояния сенсоров (рис. 10.19):

I. Сенсоры находятся по обе стороны от линии (00), нормальная ситуация, в которой линия находится между датчиками, поэтому робот должен ехать прямо (особый случай – линия потеряна).

II. Линия найдена справа (01), робот должен повернуть направо, подав на левое колесо больше энергии, чем на правое.

III. Линия найдена слева (10), робот должен повернуть налево, подав на правое колесо больше энергии, чем на левое.

IV. Оба сенсора находятся над линией, которая пересекает линию движения (11), в общем случае, робот продолжает двигаться прямо.

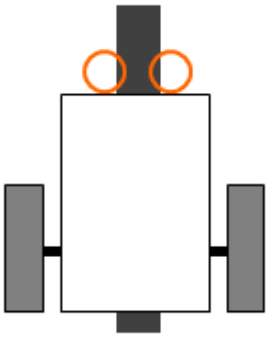


Рисунок 10.18 – Расположение двух датчиков освещенности (иллюстрация с сайта http://nnxt.blogspot.ru/)

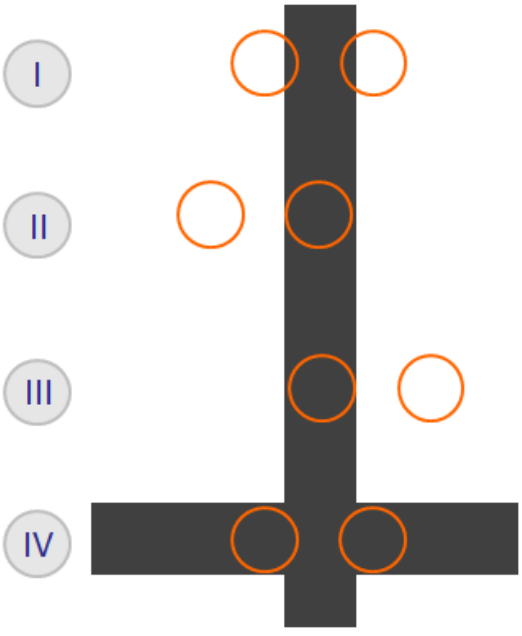


Рисунок 10.19 – Варианты состояний датчиков освещенности (иллюстрация с сайта http://nnxt.blogspot.ru/)

При указанной конфигурации сенсоры устанавливаются так, чтобы центр отрезка соединяющего два сенсора совпатал с вершиной равностороннего треугольника.

Недостатком набора из двух сенсоров является то, что робот не может различить две ситуации: линия потеряна и сенсоры находятся по обе стороны от линии.

Три сенсора

Данная схема считается наиболее часто используемой и позволяет более гибко адаптироваться к форме линии [1]. При её использовании можно увеличить скорость передвижения робота вдоль линии. Допустим, k – базовый уровень энергии, подаваемый на левое и правое колеса, тогда поаналогии с предыдущими конфигурациями можно установить следующие состояния сенсоров:

– 001 – линия ушла вправо, установка энергии: левое колесо = k; правое колесо = 0,1 × k;

– 010 – робот находится над линией, установка энергии: левое колесо = 0,5 × k; правое колесо = 0,5 × k;

– 011 – линия находится правее, установка энергии: левое колесо = 0,5 × k; правое колесо = 0,1 × k;

– 100 – линия ушла влево, установка энергии: левое колесо = 0,1 × k; правое колесо = k;

– 110 – линия находится левее, установка энергии: левое колесо = 0,1 × k; правое колесо = 0,5 × k;

Особые случаи (робот потерял линию):

– 111 – данное состояние возникает, когда все три сенсора находятся над линией пересекающей линию движения, установка энергии: левое колесо = 0,5 × k; правое колесо = 0,07 × k; еще один вариант поведения в данной ситуации – развернуться на 90 градусов и продолжить движение;

– 101 – робот должен двигаться по окружности, установка энергии: левое колесо = 0,5 × k; правое колесо = 0,07 × k;

– 000 – линия потеряна, и робот должен двигаться по окружности, установка энергии: левое колесо = 0,5 × k; правое колесо = 0,07 × k.

При указанной конфигурации сенсоры устанавливаются на одной линии так, чтобы центр отрезка соединяющего два крайних сенсора совпатал с вершиной равностороннего треугольника в которой находился бы третий сенсор.

Пять сенсоров

Добавление двух датчиков освещенности к предыдущей схеме обеспечивает более тонкую степень контроля, которая позволяет увеличить скорость робота на прямых, чтобы компенсировать время, затраченное на прохождение поворотов. Наиболее интересными для рассмотрения в данном случае являются следующие состояния сенсоров:

00000 – линия потеряна, и робот должен двигаться по окружности;

00001 – робот почти потерял линию, необходимо снизить скорость и полностью повернуть направо;

00011 – линия ушла вправо, необходимо повернуть направо;

00010 – линия ушла вправо, необходимо держаться правого края;

00110 – линия ушла чуть правее от центра, необходима небольшая коррекция направо;

00100 – робот находится по центру линии, допустимо увеличить скорость движения для прямых участков;

01100 – линия ушла чуть левее от центра, необходима небольшая коррекция влево;

01000 – линия ушла влево, необходимо держаться левого края;

11000 – линия ушла влево, необходимо повернуть налево;

10000 – робот почти потерял линию, необходимо снизить скорость и полностью повернуть налево;

11111 – все сенсоры находятся над линией, которая пересекает линию движения.

При указанной конфигурации сенсоры освещенности устанавливаются с учетом предыдущих указаний на одной линии равномерно с таким межсенсорным расстоянием, которое обеспечивает возможность генерации рассмотренных состояний датчиков.

10.2. Методические рекомендации

10.2.1 Методика создания сцены

Для организации сцены для моделирования движения робота по линии выполните следующие действия [9].

1. Запустите среду Microsoft VPL и поместите на диаграмму сервис SimulatedGenericDifferentialDrive. Откройте свойства созданного сервиса (панель Properties, пункт меню View → Toolboxes → Properties). Вставьте в сцену робота (например, iRobot Create) выбрав соответствующий манифест IRobot.Create.Simulation.Manifest.xml, который определяет робота и сцену, которые будут добавлены в симулятор.

2. Запустите диаграмму на выполнение. Откройте редактор симулятора: пункт Mode → Edit главного меню окна. Удалите со сцены все объекты, кроме greybox, ground, IRobotCreateMotorBase, MainCamera, Sky и Sun. Объект greybox будет использоваться в качестве элемента линии.

3. Выделите объект greybox в списке объектов и измените его положение в разделе Rotation (рис. 10.20).

4. Нажмите кнопку «Edit Entity», при этом откроется окно свойств объекта. Ширину параллелипипеда (объекта greybox) установите так, чтобы она была меньше чем расстояние между сенсорами освещенности, размещенными на роботе (свойство MeshScale, см. рис. 10.21).

5. Создаваемая линия должна находиться на горизонтальной поверхности, однако робот не должен взаимодействовать с ней. Чтобы определить такое поведение, перейдите в раздел Misc (Прочее) → EntityState нажав на «...» и в пункте Misc (Прочее) → Flags установите два флага: Kinematic и DisableCollisions. Это позволит вручную указать координаты объекта на сцене, а также определить, что с данным объектом не будут взаимодействовать другие объекты сцены.

6. В пункте BoxShape (раздел Shape) нажмите на «...», на экране откроется окно, в котором, в разделе Dimensions, можно будет изменить размеры объекта.

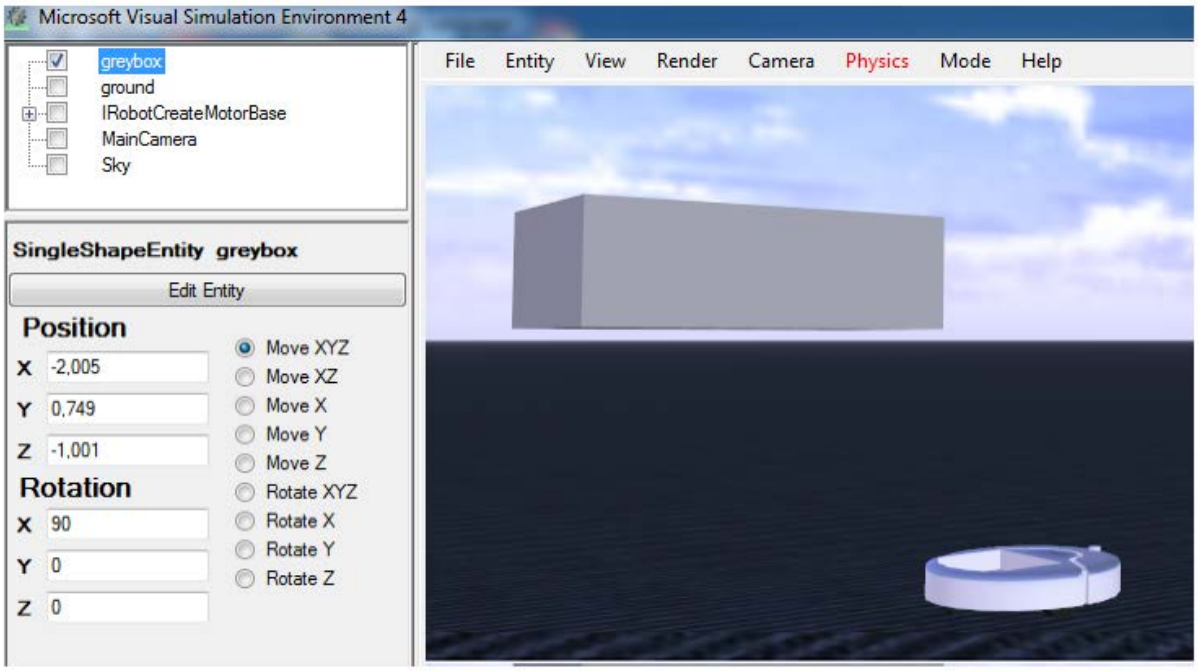


Рисунок 10.20 – Окно параметров объекта greybox

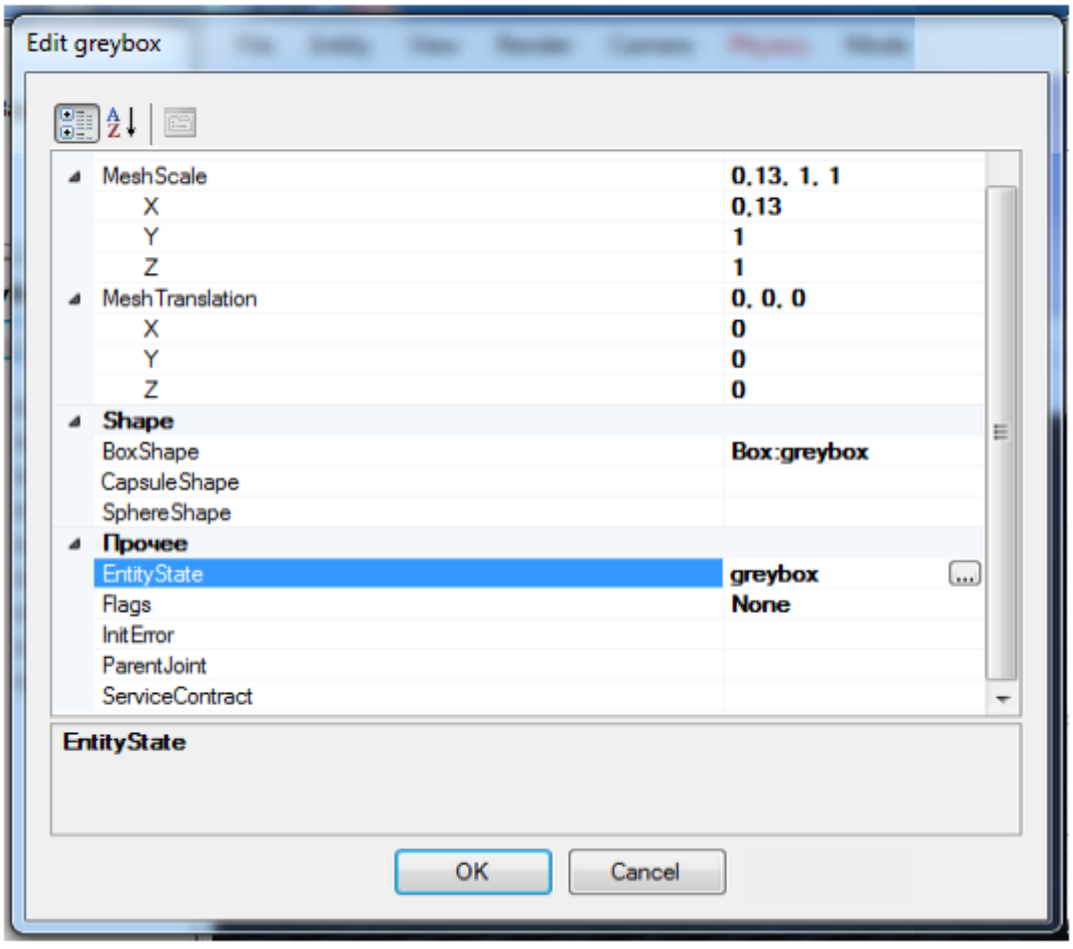


Рисунок 10.21 – Окно параметров объекта greybox

7. Линия движения робота состоит из множества элементов, которые получаются путем выделения объекта greybox в списке, копирования его и вставки в симулятор. Координаты созданной копии объекта будут такие же, как и у исходного объекта (то есть исходный объект и копия будут перекрываться в пространстве). Используя описанный прием, разместите блоки так, чтобы они образовали линию движения робота.

8. После создания линии движения поместите робота в начале маршрута над линией.

9. Сохраните созданную сцену, выбрав пункт меню File → Save Scene as... (имя файла для сохранения – line). В результате будут сгенерированы два файла – line.xml и line.Manifest.xml. Сохранять сцену нужно в каталог установки RDS. Файл line.xml включает описание сцены и объектов, которые в ней находятся, line.Manifest.xml – описание используемых сервисов.

10. Откройте сохраненную сцену (файл line. Manifest.xml) в программе DSS Manifest Editor и пересохраните ее в тот же каталог. Исходный и пересохраненный файлы отличаются содержимым. Закройте редактор манифеста. Закройте среду VPL и снова откройте в ней созданную диаграмму (данное действие нужно выполнить для того, чтобы VPL перезагрузил список манифестов). В панели Properties свяжите сервис SimulatedGeneric DifferentialDrive с манифестом line.Manifest.xml.

10.2.2 Методика добавления сенсоров к роботу

В среде Microsoft RDS R4 существует возможность подключения к роботу дополнительных сенсоров, которые затем могут быть использованы при решении различных задач. Добавление сенсора связывает его как дочерний объект с родительским объектом, при этом их перемещение по сцене осуществляется совместно [1].

Для подключения сенсора к роботу выполните следующие действия.

1. Запустите симулятор, открыв сцену с роботом, который нужно модифицировать, добавив к нему дополнительные сенсоры.

2. Откройте в симуляторе манифест \samples\Config\EntityUI.manifest.xml (File → Open Manifest).

3. На экране появится окно вставки в сцену робота, в котором нужно указать сенсоры, входящие в состав робота (рис. 10.22). Нажмите в данном окне кнопку Add Motor Base, не изменяя состав сенсоров. В результате в сцену будет вставлен робот.

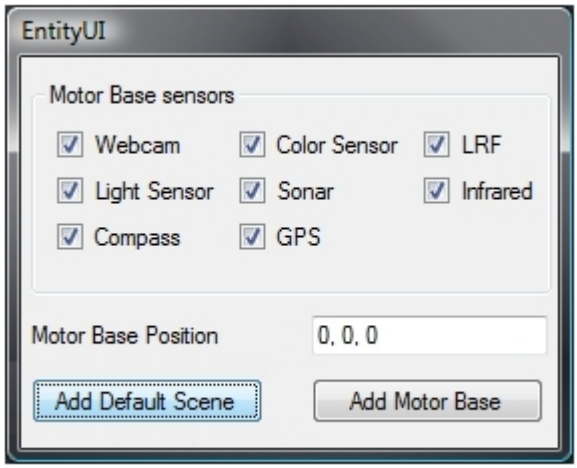


Рисунок 10.22 – Окно вставки робота в сцену

4. В симуляторе перейдите в режим редактирования. В списке объектов вставленного робота выберите сенсор освещенности SimulatedBrightnessSensor и скопируйте его. В списке объектов выберите пункт IRobotCreateMotorBase и добавьте скопированный сенсор в состав робота, используя пункт меню Edit → Paste as Child. В результате вставленный сенсор появится в списке дочерних объектов робота Lego NXT (рис. 10.23).

5. Перейдите в режим Run и сохраните полученную сцену. Откройте сцену в программе DSS Manifest Editor и сохраните загруженный манифест под тем же именем (пункт меню File → Save). После сохранения полученный манифест можно использовать при разработке диаграммы.

Вставленные сенсоры не визуализируются на сцене. Для обозначения сенсора на сцене можно загрузить для него графическую модель, например Arrow.obj (данная модель входит в поставку RDS).

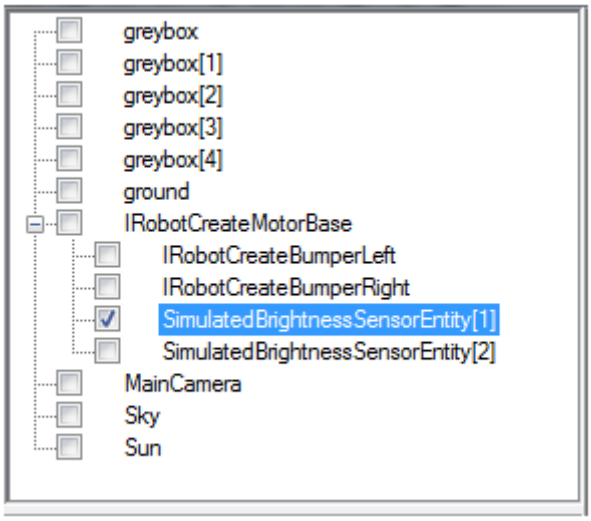


Рисунок 10.23 – Результат вставки сенсоров

6. Откройте окно параметров объекта SimulatedBrightnessSensorEntity, перейдите в раздел Misc (Прочее) → EntityState и в пункте Graphic Assets → Mesh укажите, что графическая модель для сенсора загружается из файла Arrow.obj. После визуализации сенсора можно увидеть, как сенсор располагается относительно робота (рис. 10.24), а также наглядно выполнить его перемещение по сцене. Координаты и направления в пространстве сенсоров при схеме из 3 датчиков приведены в табл. 10.1.

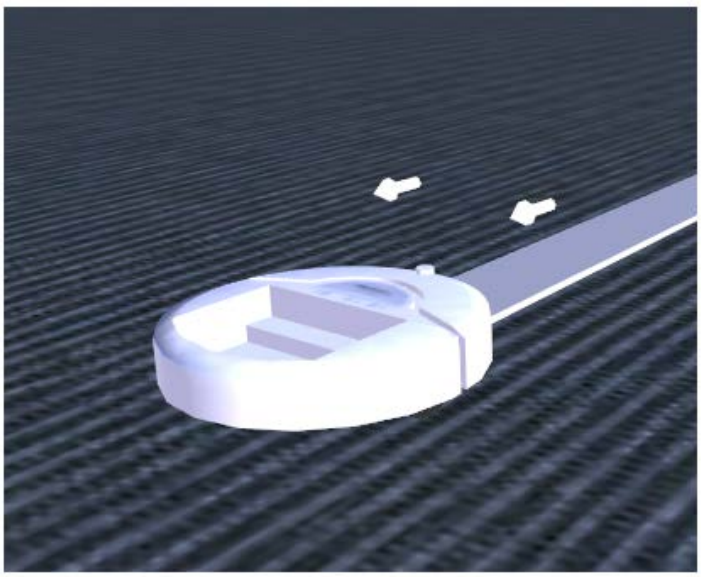


Рисунок 10.24 – Результат визуализации сенсоров освещенности

Таблица 10.1. Координаты и направления в пространстве сенсоров

	Номер сенсора					
	Координаты			Направление		
Параметр	1	2	3	1	2	3
X	– 0,1	0,1	0	– 90	– 90	– 90
Y	0,15	0,15	0,15	0	0	0
Z	– 0,2	– 0,2	– 0,2	0	0	0

10.2.3 Методика разработки диаграммы движения робота по линии

Для организации движения робота по линии диаграмма должна предоставлять возможность запускать и останавливать робота, а также учитывать время движения робота. Условно диаграмму управления роботом можно разбить на три части [9]:

1) первая часть включает блоки для инициализации переменных, учета времени движения робота и его отображения;

2) вторая часть отвечает за запуск и остановку движения робота;

3) третья часть выполняет управление движением робота на основе показаний сенсоров.

Рассмотрим переменные, необходимые для реализации алгоритма:

– drive (тип bool) – используется для разрешения движения робота;

– k (тип int) – используется для управления энергией, подаваемой на колеса робота;

– time (тип int) – время движения робота;

– Left, Right, Center (тип bool) – описывают состояние левого, правого и центрального сенсоров яркости.

Блоки, относящиеся к первой части, показаны на рис. 10.25. С помощью пары блоков Data (1) и Variable (1) устанавливается значение переменной k. Data хранит число, которое будет записано в переменную k. При активации блока Data значение, находящееся в текстовом поле, помещается в сообщение DataValue (поле Value) и отправляется на вход блока Variable. Блок Variable с помощью действия SetValue устанавливает значение переменной k.

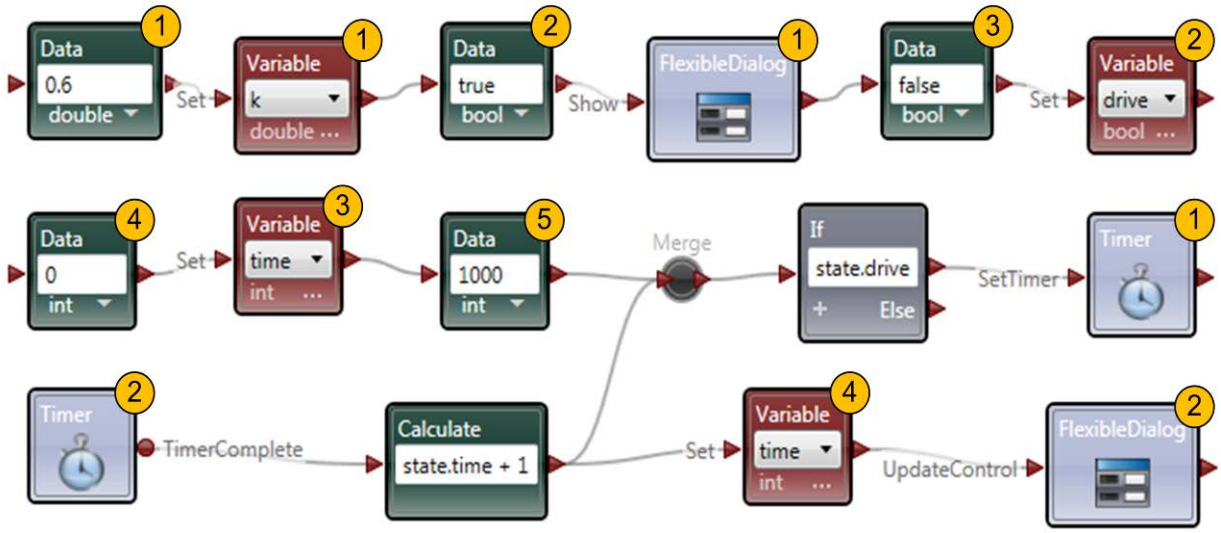


Рисунок 10.25 – Первая часть диаграммы управления роботом

1. Для передачи сообщения от блока Data (1) к блоку Variable (1) соедините их. Появится диалог создания связи между блоками (Connections). В диалоге Connections указывается исходящее сообщение блока (поле From) и действие (поле To), на вход которого сообщение отправляется. Сообщение DataValue передается с выхода блока Data на вход действия SetValue блока Variable.

2. Соедините блоки Variable (1) и Data (2). Управление движением робота (запуск, остановка), а также отображение времени движения робота выполняется в диалоговом окне, которое создается сервисом FlexibleDialog. Поместите данный сервис на диаграмму и сконфигурируйте его в панели Properties так, как показано на рис. 10.26.

В диалоговое окно сервиса FlexibleDialog можно поместить компоненты двух видов: управляющие элементы (раздел Controls панели Properties) и кнопки (раздел Buttons). Элементы управления размещаются в диалоговом окне в той же последовательности, в какой они были созданы в панели инструментов Properties. Параметр Id используется для программного управления элементом.

После настройки FlexibleDialog будет содержать две кнопки и статический текст. Для отображения диалогового окна на экране нужно вызвать метод Show, передав ему в качестве параметра значение true.

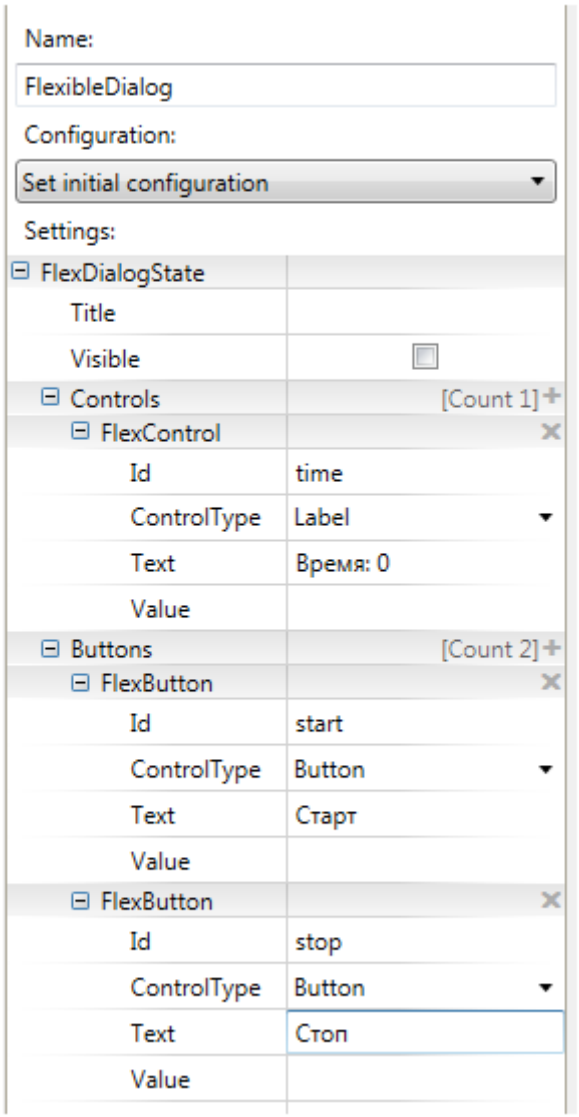


Рисунок 10.26 – Настройка сервиса FlexibleDialog

3. При соединении Data (2) и FlexibleDialog (1) открывается диалоговое окно Connections, в котором укажите, что сообщение с выхода блока Data отправляется на вход действия Show сервиса FlexibleDialog. Далее на экране появится диалоговое окно Data Connections, в котором нужно установить соответствие между полями сообщения (Value) и параметрами вызываемого действия (Target).

4. Результатом выполнения сервисом FlexibleDialog действия Show может быть Success или Fault. Поэтому при создании связи между блоками FlexibleDialog (1) и Data (3) укажите, что в случае успешного отображения диалогового окна на экране (Show – Success), управление будет передано блоку

Data. Для этого передайте сообщение Show – Success на вход действия Create блока Data.

5. Соедините блоки Data (3) и Variable (2) для установки значения переменной drive.

Во второй строке диаграммы инициализируется значение переменной time и устанавливается таймер на срабатывание через 1000 миллисекунд.

6. Соедините блоки Data (4) и Variable (3), Variable (3) и Data (5). Блок Data (5) нужно соединить с блоком If (блок Merge будет добавлен позднее), который в зависимости от истинности или ложности условия, заданного в текстовом поле, отправляет сообщение, полученное на вход, на первый или второй выход. Если условие оказывается истинным, то входящее сообщение передается на первый выход, если ложно – на второй (Else). При записи условия могут использоваться логические операторы, показанные на рис. 10.27.

Connections:	
From:	To:
TrueChoice	SetTimer
Data Connections:	
Value	Target
1000	Interval

Рисунок 10.27 – Параметры соединения If и Timer

В рассматриваемой диаграмме блок If анализирует значение переменной drive (доступ к значению данной переменной выполняется с помощью выражения state.drive). В случае истинности значения переменной таймер устанавливается на срабатывание через 1000 миллисекунд.

7. Соедините If и Timer (1). Сервис Timer служит для генерации сообщений через указанный промежуток времени. При соединении блоков укажите, что вызывается действие SetTimer. Данное действие запускает таймер, который должен сработать через указанный промежуток времени (параметр Interval в диалоговом окне Data Connections). Параметры соединения блоков приведены на рис. 10.27.

8. Для того чтобы указать значение 1000 в поле Value, установите флажок «Edit values directly» в нижней части диалогового окна Data Connections.

9. Поместите на диаграмму блоки, размещенные в третьей строке, и соедини уведомляющий выход сервиса Timer (2) и вход блока Calculate. В результате на вход Calculate будет отправлено сообщение TimerComplete. Соедините блоки Calculate и Variable (4).

По истечении 1000 миллисекунд таймер срабатывает и генерируется уведомляющее сообщение TimerComplete. Далее управление передается на блок Calculate, который прибавляет к текущему значению переменной time единицу и отправляет полученное значение на вход блока Variable (4), который обновляет значение переменной time.

10. Для изменения значения времени в диалоговом окне нужно вызвать метод UpdateControl сервиса FlexibleDialog. Для этого нужно соединить блоки Variable (4) и FlexibleDialog (2). Параметры соединения приведены на рис. 10.28.

Connections:	
From:	To:
SetValue	UpdateControl
Data Connections:	
Value	Target
"time"	Id
FlexControlType.Label	ControlType
"Время: " + state.time	Text
""	Value

Рисунок 10.28 – Параметры соединения Variable и FlexibleDialog

Таймер, установленный с помощью SetTimer, может сработать только один раз. Для того чтобы таймер срабатывал периодически, с помощью блока Merge создана связь. Блок Merge отправляет на свой выход сообщение сразу, как только оно приходит на его вход. В результате после срабатывания таймера сообщение снова отправляется на вход сервиса Timer.

Рассмотрим вторую часть диаграммы (см. рис. 10.29).

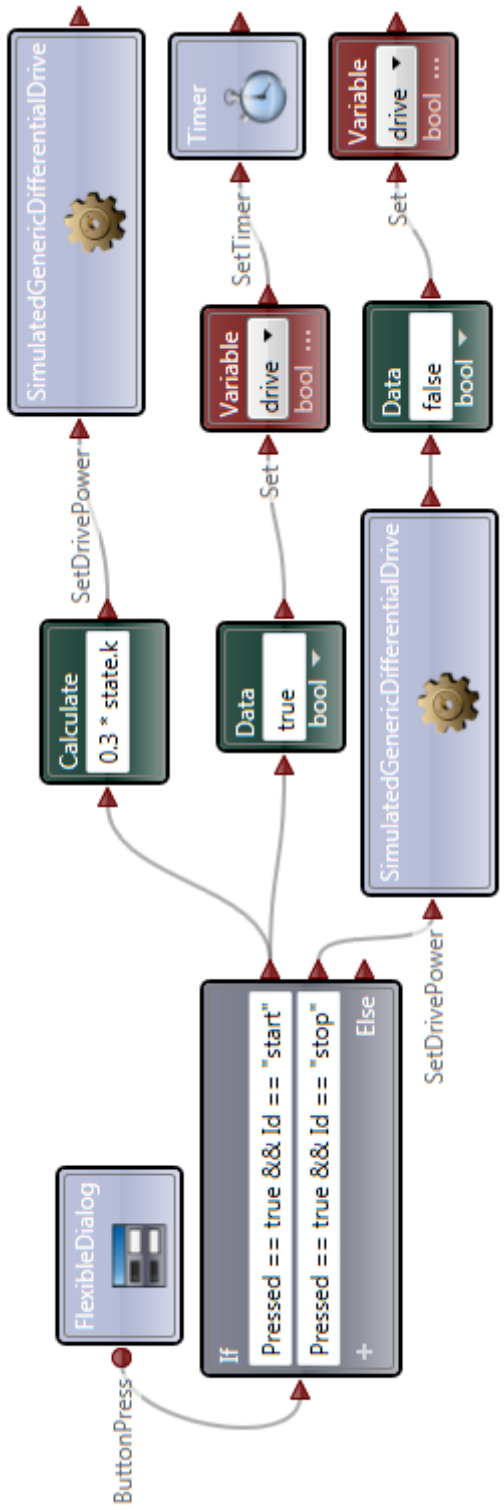


Рисунок 10.29 – Вторая часть диаграммы управления роботом

В данной части диаграммы выполняется обработка нажатий кнопок диалогового окна и отправка управляющих команд на запуск и остановку движения робота.

11. Разместите на диаграмме блоки, показанные на рис. 10.29. Соедините уведомляющий выход сервиса FlexibleDialog и блок If, отправив на вход If сообщение ButtonPress. Условия, указанные в полях блока If, позволяют определить кнопку, которая была нажата в диалоговом окне. Если нажата кнопка «Старт», тогда:

– в переменную drive записывается значение true и запускается счетчик времени движения робота;

– с помощью сервиса SimulatedGenericDifferentialDrive роботу отправляется команда на движение (для этого вызывается метод SetDrivePower). Метод SetDrivePower задает значение энергии, подаваемое на левое и правое колеса робота. Параметры соединения блоков Calculate и SimulatedGenericDifferentialDrive приведены на рис. 10.30.

Connections:	
From:	To:
CalculatedResult	SetDrivePower
Data Connections:	
Value	Target
Value	LeftWheelPower
Value	RightWheelPower

Рисунок 10.30 – Параметры соединения Calculate и SimulatedGeneric DifferentialDrive

Если нажата кнопка «Стоп», сервис SimulatedGeneric DifferentialDrive останавливает движение робота и в переменную drive записывается false.

В третьей части диаграммы (см. рис. 10.31) выполняется обработка данных от сенсоров и управление движением робота. Для доступа к сенсору яркости робота используется сервис SimulatedPhotoCellBrightnessSensor. Данный сервис периодически запрашивает у сенсора данные и отправляет их на свой выход в

уведомляющем сообщении AnalogSensorUpdate. Сообщение AnalogSensor Update содержит поле NormalizedMeasurement, которое хранит значение яркости, считанное сенсором (значение нормализовано на отрезке [0; 1]).

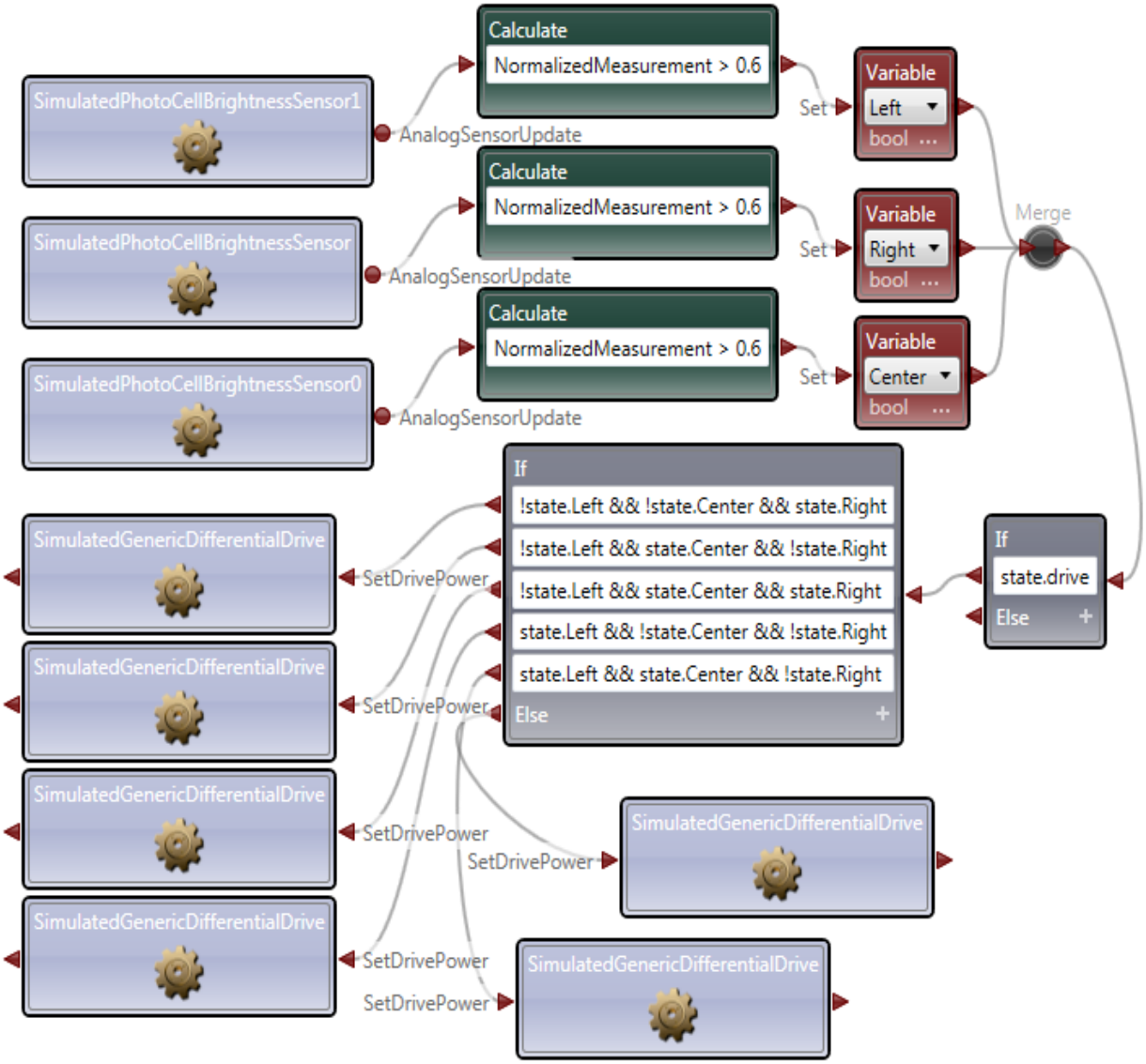


Рисунок 10.31 – Третья часть диаграммы управления роботом

12. Добавьте на диаграмму три сервиса SimulatedPhotoCellBrightnessSensor. При добавлении второго и третьего сервисов на экране появится диалоговое окно, в котором укажите, что создается новый сервис (выберите пункт «Add a new activity»), а не копия существующего.

13. Свяжите каждый сервис SimulatedPhotoCellBrightnessSensor с сенсором яркости. Для этого в панели Properties для сервиса SimulatedPhoto CellBrightnessSensor необходимо указать, что он связан с «simbrightnessensor0 in

line. Manifest.xml», сервис SimulatedPhotoCellBrightnessSensor0 – с «simbrightnessensor1 in line.Manifest.xml», сервис SimulatedPhotoCellBrightness Sensor1 – с «simbrightnessensor in line.Manifest.xml».

Значение каждого сенсора анализируется в блоке Calculate. Если значение яркости больше 0,6, то значение переменной (Left, Right или Center), связанной с сенсором, устанавливается в 1, иначе – в 0. Далее на основе полученных значений переменных Left, Right и Center с помощью блоков If и Simulated GenericDifferentialDrive устанавливается количество энергии, подаваемое на левое и правое колеса робота.

14. Параметры соединения 1 выхода блока If и сервиса SimulatedGeneric DifferentialDrive приведены на рис. 10.32. Остальные настройки связи выходов блока If и SGDD выполните сами, используя рекомендации, приведенные в пункте 10.1 по обеспечению уровня подаваемой на колеса энергии.

Connections:	
From:	To:
TrueChoice	SetDrivePower
Data Connections:	
Value	Target
state.k	LeftWheelPower
0.1*state.k	RightWheelPower

Рисунок 10.32 – Параметры соединения If и SimulatedGenericDifferential Drive

15. Запустите разработанную диаграмму на выполнение. В результате будет открыт симулятор со сформированной сценой и робот начнет движение по линии.

После запуска среда разработки активирует блоки Data, находящиеся в первой строке. Далее запускается процесс установки таймеров и обработки данных, поступающих от сенсоров, на экране отображается симулятор и диалоговое окно управления роботом. После нажатия на кнопку «Старт» робот начинает движение вдоль линии, запускается отсчет времени. После нажатия на

кнопку «Стоп» робот прекращает движение, отсчет времени приостанавливается.

10.3. Лабораторная работа 10. Моделирование движения робота по линии с использованием датчиков

10.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков моделирования движения робота по линии с использованием сенсоров освещенности в среде Visual Programming Language.

Содержание

1. Создание сцены для моделирования движения робота по линии.
2. Организация добавления сенсоров к роботу.
3. Разработки управляющей диаграммы моделирования движения робота по линии с использованием сенсоров освещенности.

10.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4.

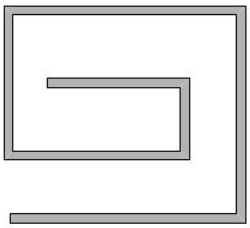
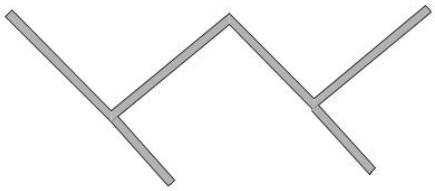
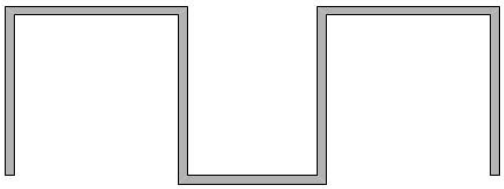
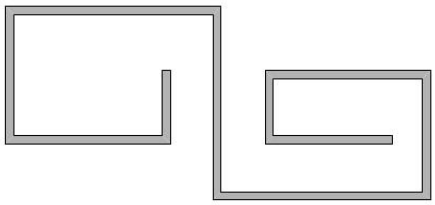
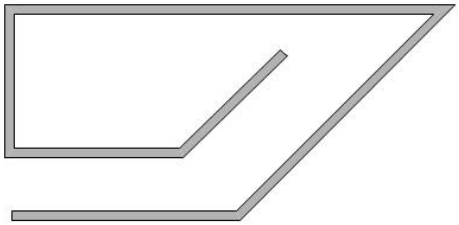
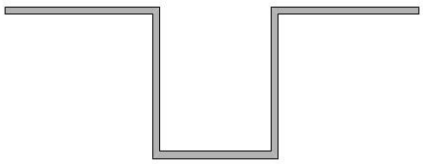
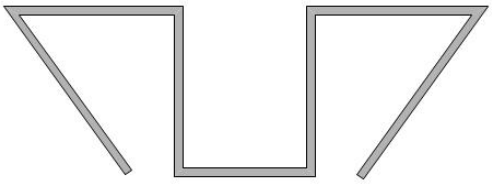
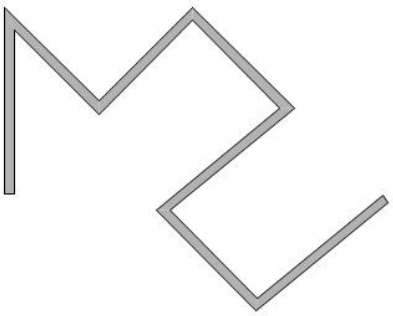
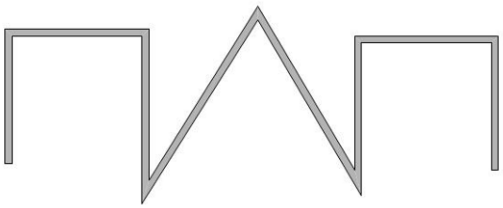
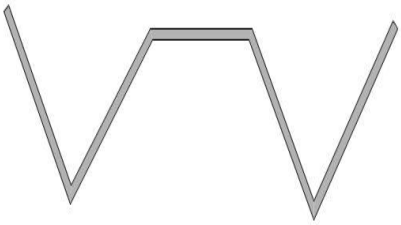
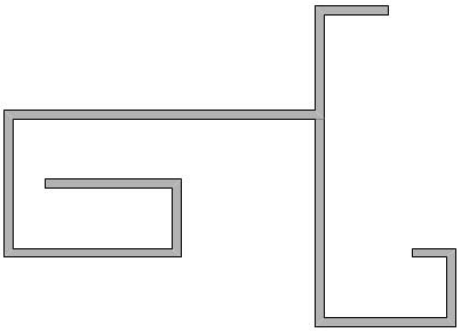
10.3.3 Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Создайте сцену согласно методике 10.2.1 с учетом задания в табл. 10.2.
2. Добавьте сенсоры к роботу согласно методике 10.2.2 и с учетом того, что для нечетных номеров вариантов задания используется 3 сенсора освещенности, а для четных номеров вариантов задания используется 2 сенсора.
3. Разработайте управляющую диаграмму согласно методике 10.2.3.

Таблица 10.2. Варианты заданий для самостоятельного выполнения

№ п/п	Рисунок	№ п/п	Рисунок
1.		2.	
3.		4.	
5.		6.	
7.		8.	
9.		10.	
11.		12.	

10.4. Контрольные вопросы

1. Охарактеризуйте наиболее часто встречающиеся в составе мобильных роботов сенсоры.
2. Перечислите особенности моделирования сенсоров в среде Microsoft RDS R4.
3. Назовите алгоритмы движения по линии, рассчитанные на различное число датчиков.
4. Какие параметры используются для организации движения робота по линии?
5. Какие состояния сенсоров мобильного робота возможны при использовании одного датчика?
6. Какие состояния сенсоров мобильного робота возможны при использовании двух сенсоров?
7. В чем преимущество использования трех сенсоров?
8. Какие состояния сенсоров мобильного робота возможны при использовании пяти сенсоров?

ГЛАВА 11. УПРАВЛЕНИЕ РОБОТОМ ПРИ ОБХОДЕ ЛАБИРИНТА

Проход робота по лабиринту является одной из классических задач для всякого рода состязаний по робототехнике. Задача сама по себе интересна, тем, что ее базовый алгоритм достаточно прост. Но в тоже время он предоставляет большой простор как для усложнения поведения робота для выхода из сложных лабиринтов, так и, наоборот, для оптимизации поведения робота, для обхода лабиринтов специфичных видов [13].

11.1. Краткие теоретические сведения

Одним из самых простых правил для прохождения лабиринта является правило «одной руки»: двигаясь по лабиринту, надо все время касаться правой или левой рукой его стены [23]. Этот алгоритм, вероятно, был известен еще древним грекам. Придется пройти долгий путь, заходя во все тупики, но в итоге цель будет достигнута. Хотя у этого правила и есть один недостаток, но о нем мы поговорим позже.

Попробуем описать робота, действующего в соответствии с правилом «правой руки». В начале своей работы робот должен найти стену, по которой он будет следовать. Для этого он может просто двигаться вперед, пока не упрется в преграду. После того как робот наткнулся на препятствие, он начинает передвигаться в соответствии с правилом «правой руки».

Двигаясь вдоль стены, робот следит, есть ли проход справа. Если проход есть, робот должен идти по нему, чтобы не оторваться от стены справа. Если прохода нет – впереди стена – робот поворачивает налево. Если прохода снова нет, он еще раз поворачивает налево, таким образом, разворачиваясь на 180 градусов, и идет в обратном направлении.

Блок-схема алгоритма для робота, работающего по правилу «правой руки», представлена на рис. 11.1.

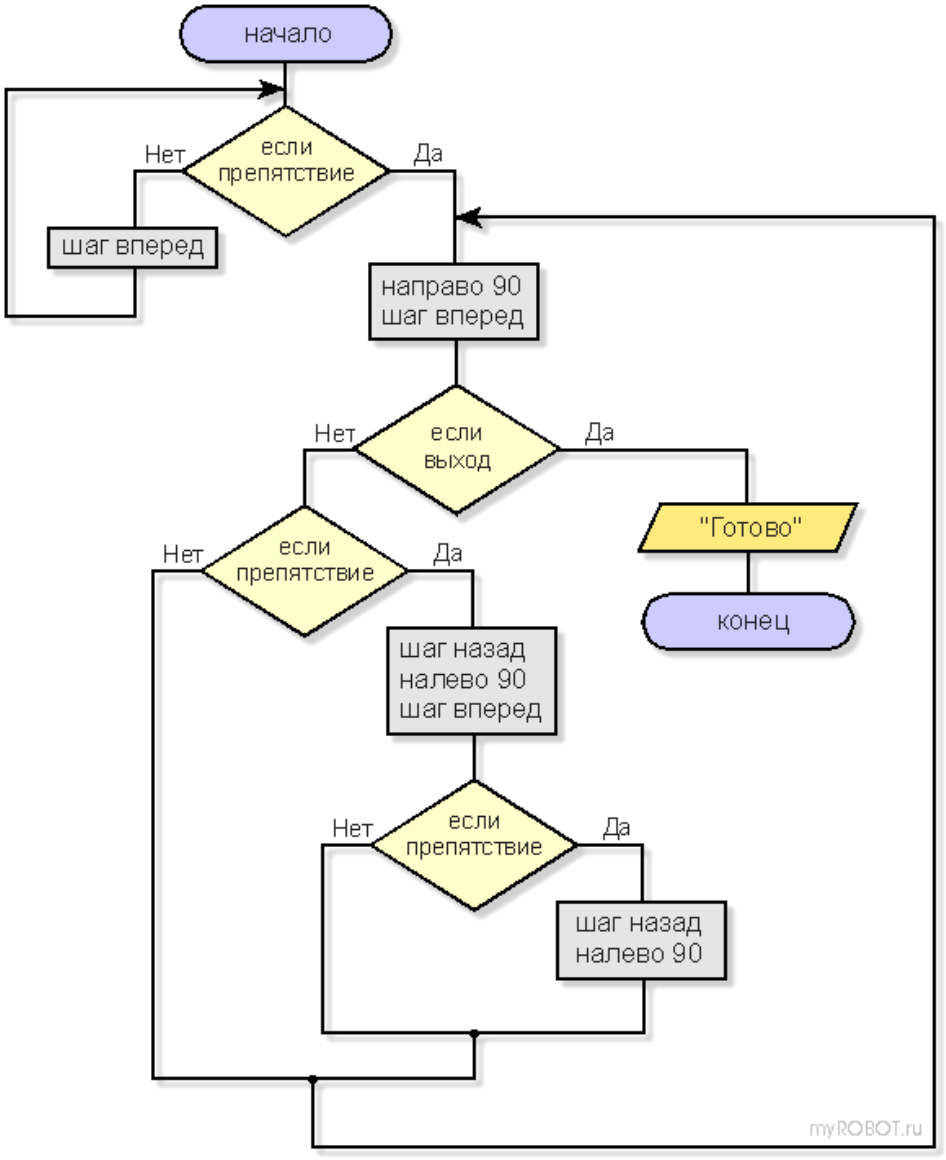
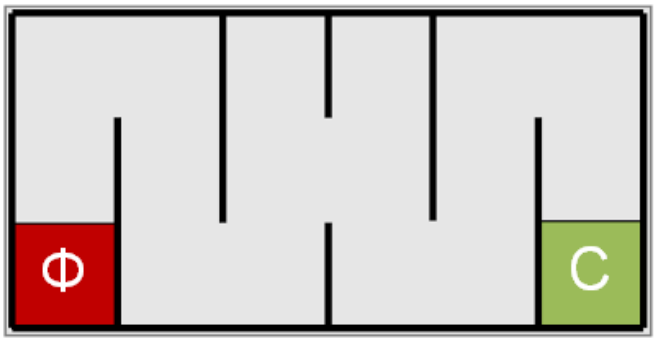


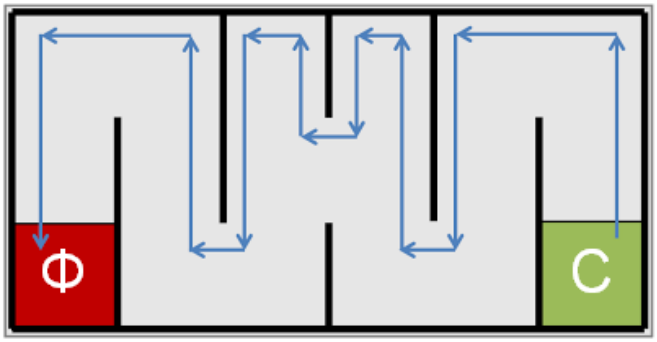
Рисунок 11.1 – Блок-схема алгоритма обхода лабиринта по правилу «правой руки» (иллюстрация с сайта http://myrobot.ru)

Рассмотрим, как работает данное правило в задании для младшей группы Международной робототехнической олимпиады. Согласно заданию, лабиринт состоит из шести секций, соединенных между собой проходами. Конфигурация проходов заранее неизвестна. Таким образом, одно из возможных конфигураций лабиринта будет выглядеть следующим образом (рис. 11.2а) – секция С – место старта робота, а секция Ф – место финиша.

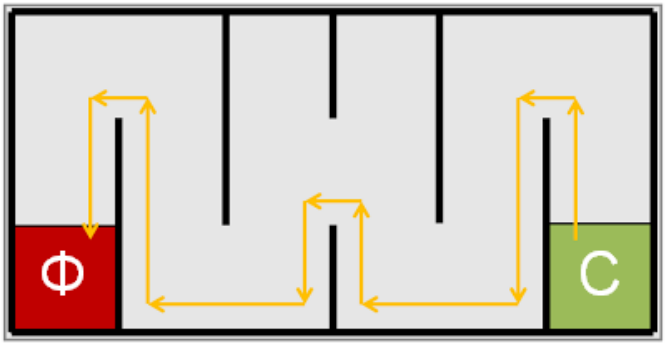
Траектория обхода лабиринта по правилу «правой руки» приведена на рис. 11.2б, а результат использования правила «левой руки» приведен на рис. 11.2в.



а)



б)



в)

Рисунок 11.2 – Пример лабиринта (а), траектории обхода по правилу «правой руки» (б) и по правилу «левой руки» (в)

Если известно, что у лабиринта нет отдельно стоящих стенок, то есть, нет замкнутых маршрутов, по которым можно возвращаться в исходную точку, то такой лабиринт называют односвязным и его всегда можно обойти полностью, применив правило «одной руки».

Если же лабиринт содержит отдельно стоящие стенки, то, применяя правило «одной руки», не всегда можно пройти все коридоры и тупики. Лабиринты с отдельно стоящими стенками и с замкнутыми маршрутами

называются многосвязными. При этом многосвязные лабиринты можно разделить на две группы: без «петли» вокруг цели (замкнутый маршрут не проходит вокруг цели) и с замкнутой «петлей» вокруг цели (цель можно обойти по замкнутому маршруту).

В многосвязных лабиринтах второй группы правило «одной руки» не работает и, применяя его, достичь цели невозможно. Но и эти лабиринты можно пройти, полагаясь на точный алгоритм.

Универсальный алгоритм прохождения любых лабиринтов был описан в книге французского математика Э. Люка «Recreations matematiques», изданной в 1882 году. Интересно, что Люка при описании алгоритма указал на первенство другого французского математика М. Тремо. Таким образом, алгоритм стал известен как алгоритм Люка-Тремо.

Тремо предлагает следующие правила: выйдя из любой точки лабиринта, надо сделать отметку на его стене (крест) и двигаться в произвольном направлении до тупика или перекрестка; в первом случае вернуться назад, поставить второй крест, свидетельствующий, что путь пройден дважды – туда и назад, и идти в направлении, не пройденном ни разу, или пройденном один раз; во втором – идти по произвольному направлению, отмечая каждый перекресток на входе и на выходе одним крестом; если на перекресте один крест уже имеется, то следует идти новым путем, если нет – то пройденным путем, отметив его вторым крестом.

В наши дни роботы, проходящие лабиринт, являются участниками одного из самых интересных состязаний, которое проходит в нескольких странах мира. Эти соревнования носят общее название «Micromouse competition» и по своим техническим новациям относятся к лидерам робототехнического спорта.

Алгоритм решения роботом задачи обхода лабиринта можно описать с помощью конечного автомата (машины состояний). Конечный автомат задает правило определения нового состояния робота с учетом произошедшего события и текущего состояния [1], т. е. если робот находился в состоянии А и произошло событие Х, то робот переходит в состояние В.

Таким образом, для определения конечного автомата нужно задать:

– набор состояний, в которых может находиться робот;

– события, при которых происходит переход между состояниями;

– действия, выполняющиеся при переходе робота между состояниями.

Конечный автомат может быть описан с помощью ориентированного графа. Число узлов такого графа равно числу конечных состояний робота, а число ребер – всей совокупности возможных переходов из одного состояния в другое. Одно из состояний выбирается в качестве исходного, из этого состояния робот может попасть в некоторые (все) другие состояния, выполнив некоторую последовательность переходов. Граф конечного автомата описывающего поведение робота при обходе лабиринта по правилу «правой руки» приведен на рис. 11.3 [1].

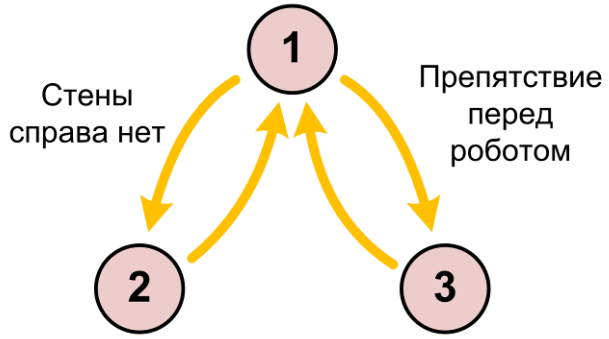


Рисунок 11.3 – Конечный автомат алгоритма обхода лабиринта по правилу «правой руки»

На рисунке приведены следующие состояния:

1. Движение прямо: робот выполняет движение прямо до тех пор, пока справа от него находится стена, выполняется проверка наличия стены справа и перед роботом – если справа стена кончается, робот переходит в состояние «поворот направо», если стена перед ним детектируется, то он переходит в состояние «поворот налево».

2. Поворот направо: робот выполняет поворот направо и переходит в состояние «движение прямо».

3. Поворот налево: робот выполняет поворот налево и переходит в состояние «движение прямо».

Рассмотрим возможности лазерного дальномера присутствующего в среде Microsoft RDS R4. В данном случае используется три луча лазерного дальномера (рис. 11.4). Диаграмма чтения результатов измерения расстояния приведена на рис. 11.5.

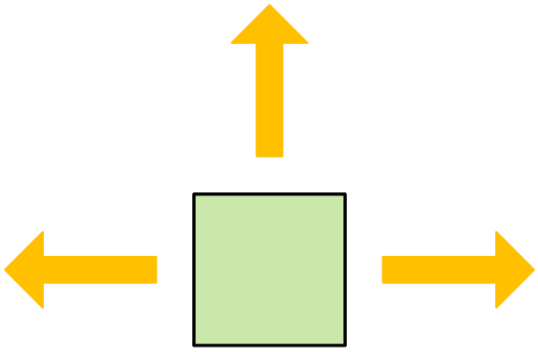


Рисунок 11.4 – Используемые лучи дальномера

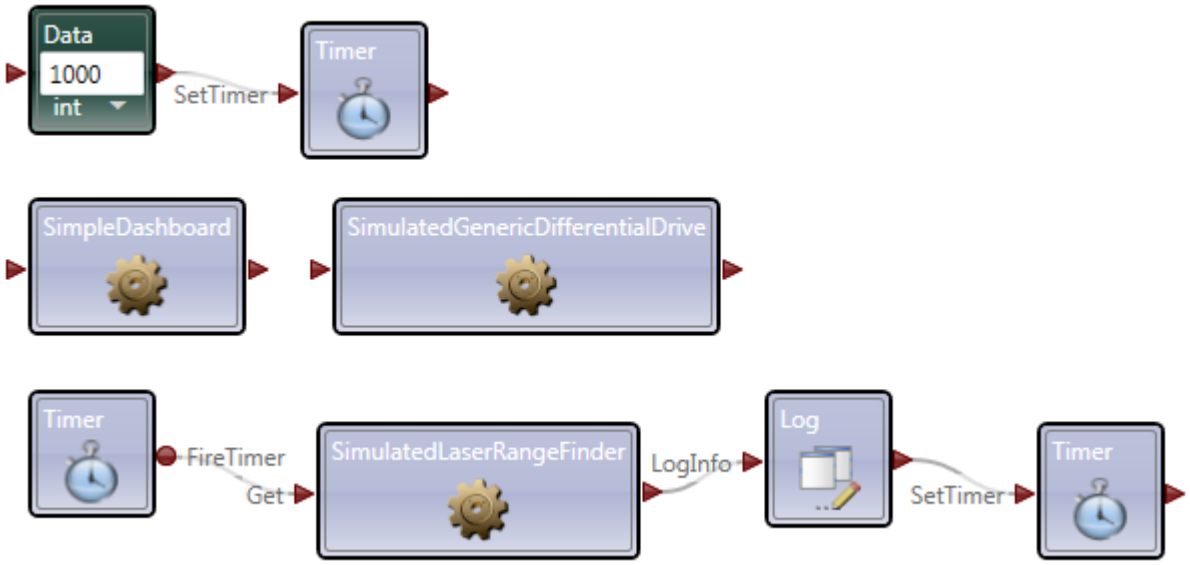


Рисунок 11.5 – Диаграмма чтения результатов измерения расстояния

Сервис SimulatedLRF предназначен для взаимодействия с лазерным дальномером, установленном на роботе. Угловой диапазон сенсора составляет 180 градусов, угловое разрешение – 0,5 градуса. Результат работы сенсора – массив расстояний до объектов сцены в миллиметрах, включающий 361 элемент (имя массива – DistanceMeasurements).

Элемент массива с номером 180 соответствует расстоянию до объекта, расположенного строго перед роботом. Максимальная дальность действия датчика составляет 8 м.

Connections:	
From:	To:
Get-Success	LogInfo
DataConnections:	
Value	Target
value. DistanceMeasurements[180]	Message
null	Category

Рисунок 11.6 – Параметры соединения SimulatedLaserRangeFinder и Log

На приведенной диаграмме запрос данных с сенсора выполняется с помощью операции Get с периодичностью в 1 секунду.

Вывод результатов измерения выполняется с помощью сервиса Log (см. рис. 11.7). Чтение значений сенсора можно организовать с помощью уведомляющего сообщения Measurement, которое генерируется сервисом четыре раза в секунду.

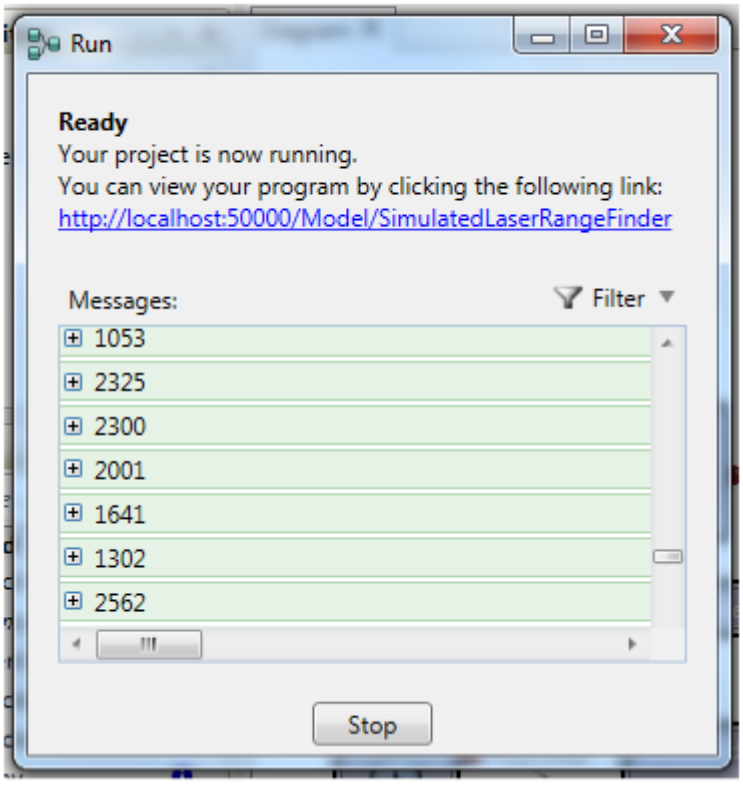


Рисунок 11.7 – Результат измерения расстояний с помощью лазерного дальномера

11.2. Методические рекомендации

11.2.1 Методика разработки диаграммы движения робота при обходе препятствия

1. Для организации случайного движения робота с обходом препятствий разработайте диаграмму, представленную на рис. 11.8, 11.9 [1]. В данном случае для определения препятствий используется лазерный дальномер (рис. 11.10).

Процесс функционирования диаграммы заключается в следующем. Сразу после запуска диаграммы робот начинает движение. Одновременно лазерный дальномер сканирует пространство перед роботом, в результате чего сервис SimulatedLaserRangeFinder генерирует уведомляющие сообщения четыре раза в секунду. Уведомления с сервиса SimulatedLaserRangeFinder передаются на три блока If, каждый из которых запускает выполнение одной из копий блока Worker, передавая на их входы определенные параметры.

Назначение блока Worker – анализ массива расстояний, получаемого с лазерного дальномера. Диаграмма организована так, что полученный массив расстояний разбивается на три части, каждая из которых обрабатывается в отдельном блоке Worker. Это сделано для того, чтобы выполнять учет препятствий, находящихся слева, справа и перед роботом. На каждый блок Worker подаются исходный массив расстояний и различные значения параметров Offset и Limit, которые определяют обрабатываемую часть массива. Самый верхний блок обрабатывает первые 3/8 объема данных, второй – следующие 2/8, третий – оставшиеся 3/8. На вход каждого блока поступает строковое значение (Right, Center, Left), которое характеризует обрабатываемый диапазон данных. Результат работы каждого блока – минимальное расстояние, отличное от нуля, найденное в рассматриваемом диапазоне.

2. Введите значения параметров, передаваемых на каждый из блоков Worker согласно рис. 11.11, 11.12. Используемые переменные:

Right – минимальное значение расстояния, найденное в правой части массива расстояний (тип int).

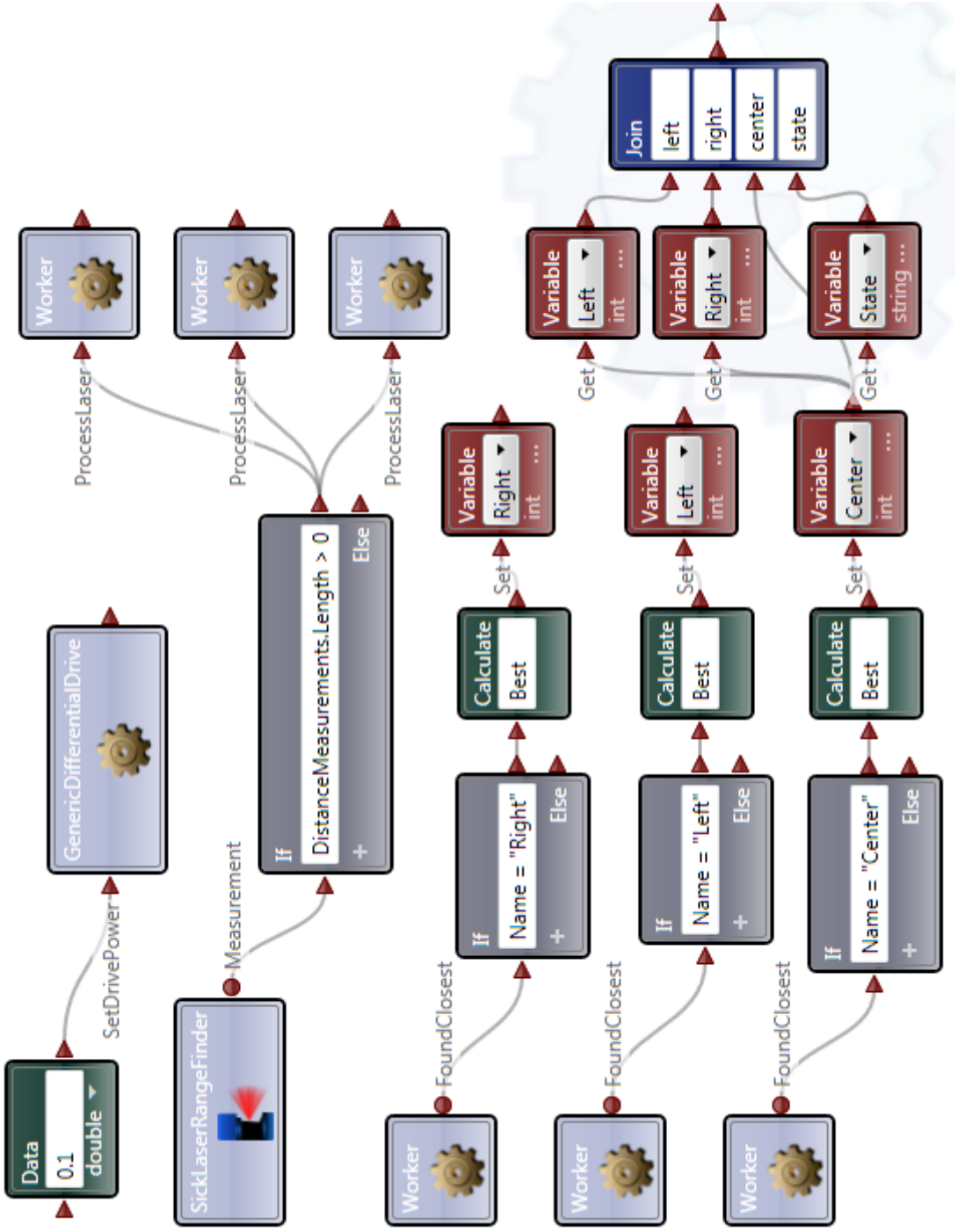


Рисунок 11.8 – Первая часть главной диаграммы

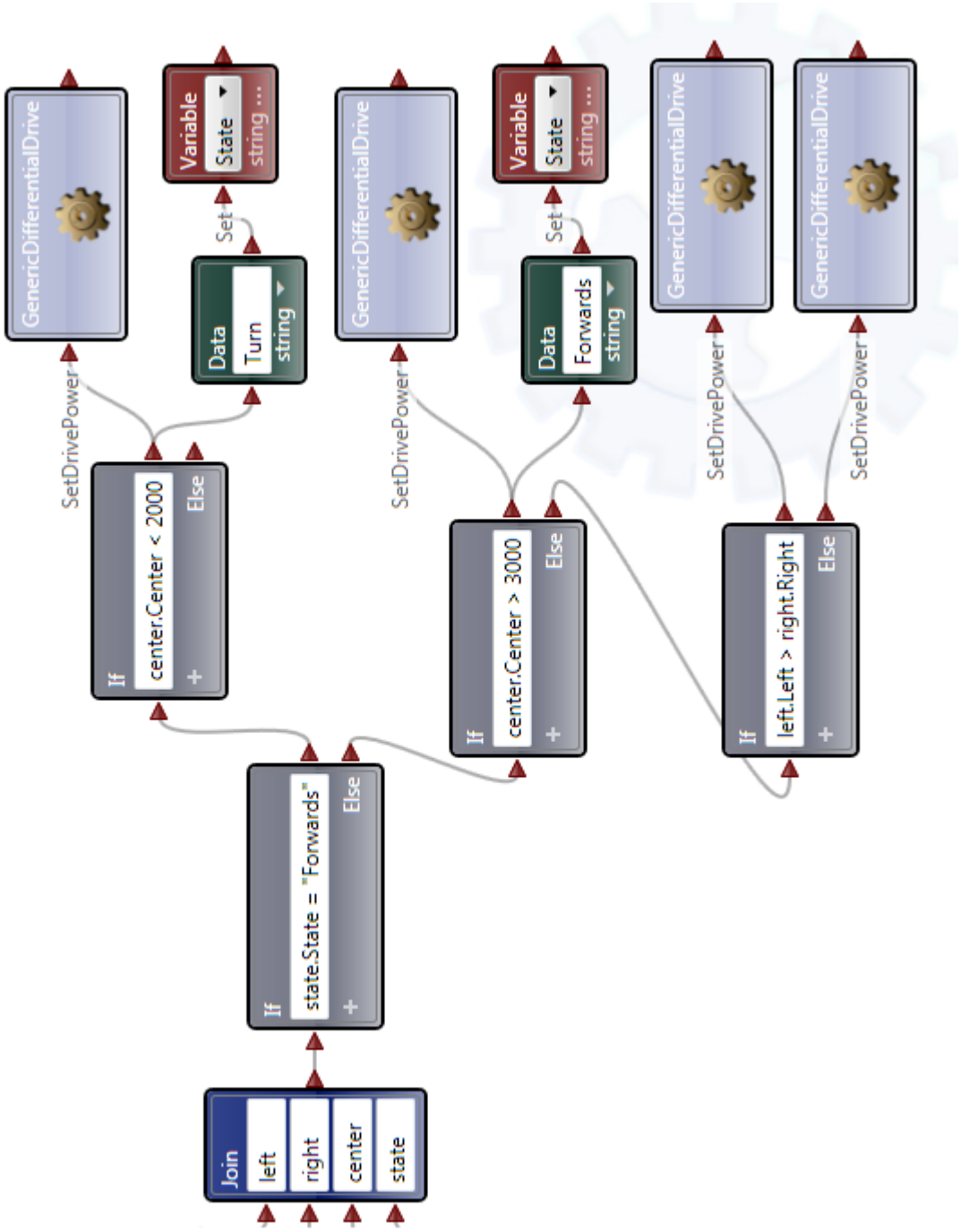


Рисунок 11.9 – Вторая часть главной диаграммы

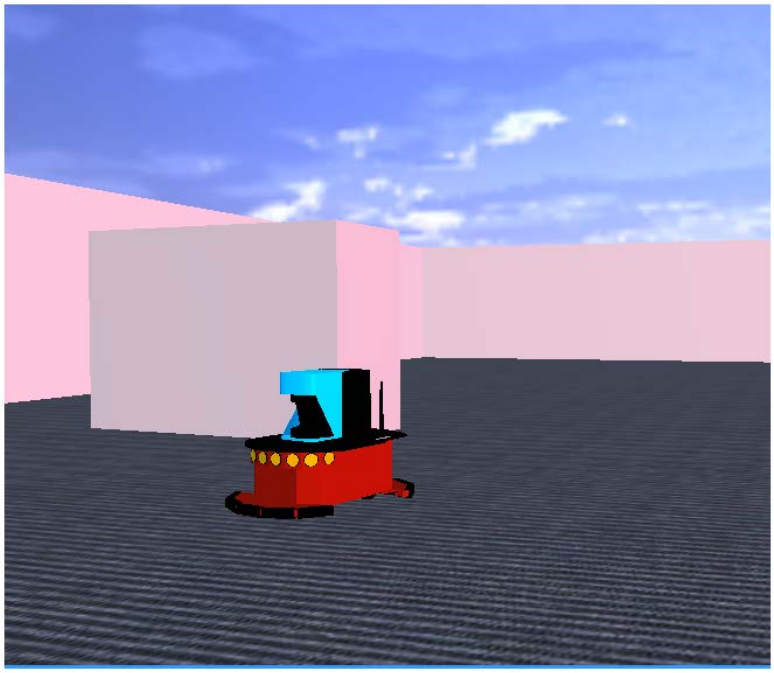


Рисунок 11.10 – Сцена моделирования движения робота при обходе препятствия

Data Connections (If и Worker, первый сверху)	Data Connections (If и Worker, второй сверху):	
Value	Value	Target
0	3 * DistanceMeasurements. Length / 8	Offset
DistanceMeasurements	DistanceMeasurements	Array
3 * DistanceMeasurements. Length / 8	5 * DistanceMeasurements. Length / 8	Limit
10	10	Increment
8000	8000	Best
«Right»	«Center»	Name

Рисунок 11.11 – Параметры, передаваемые на первый и второй блоки Worker

Left – минимальное значение расстояния, найденное в левой части массива расстояний (тип int).

Center – минимальное значение расстояния, найденное в центральной части массива расстояний (тип int).

State – описывает состояние, в котором находится робот (тип – string).

Data Connections (If и Worker, третий сверху):	
Value	Target
5 * DistanceMeasurements.Length / 8	Offset
DistanceMeasurements	Array
DistanceMeasurements.Length	Limit
10	Increment
8000	Best
«Left»	Name

Рисунок 11.12 – Параметры, передаваемые на третий блок Worker

Результат вычисления в каждом блоке Worker сохраняется в переменную Right, Left или Center. После генерации уведомляющего сообщения с блока, выполняющего обработку центральной части массива расстояний (Center), создается уведомляющее сообщение из значений переменных Right, Center, Left и State.

Далее выполняется анализ значений указанных переменных, на основе которого строится поведение робота.

Если робот находится в состоянии Forwards и center.Center < 2000, то препятствие находится прямо перед роботом, и он должен остановиться для предотвращения соударения с препятствием, значение переменной State устанавливается в Turn.

Если робот находится в состоянии Forwards и center.Center > 3000, то перед ним в непосредственной близости препятствий нет и моторы робота включаются для перемещения вперед, значение переменной State устанавливается в Forwards.

Если перед роботом детектировано препятствие, то он поворачивает налево или направо в зависимости от того, в какой области находится ближайшее препятствие.

Состояния, в которых находится робот, для удобства программирования и отладки лучше обозначать с помощью строковых значений.

3. Для обработки данных о расстояниях до препятствий разработайте блок Worker, структура которого приведена на рис. 11.13.

Для обработки переданного диапазона данных в блоке Worker используется рекурсия. Рассмотрим действия, выполняемые в блоке Worker.

Первый блок If выполняет проверку окончания обработки всего диапазона. Для этого используется условие: Offset + Increment > Limit. Если указанное условие выполняется, то входящее сообщение, а, следовательно, и найденное минимальное значение в обрабатываемом диапазоне, отправляется на уведомляющее сообщение FoundClosest. В противном случае входящее в блок сообщение отправляется на вход блока Join. С выхода блока Join сообщение передается на вход ProcessLaser блока Worker (т. е. выполняется рекурсивный вызов). Блок Worker продолжает работать, передавая сообщение на собственный вход до тех пор, пока не будет достигнуто значение Limit, после чего будет отправлено уведомляющее сообщение. Первый блок If используется для ограничения рекурсии.

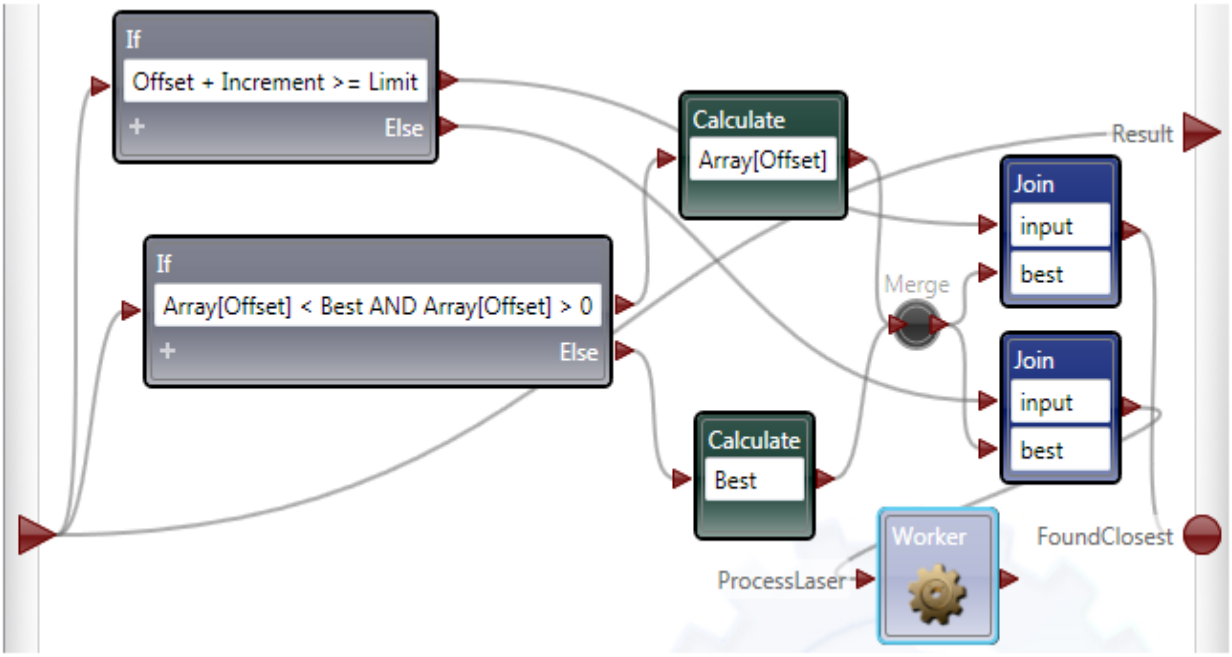


Рисунок 11.13 – Блок Worker

Второй блок If выполняет поиск минимума в диапазоне. Если значение рассматриваемого элемента меньше чем предыдущее найденное, то оно записывается в поле Best блока Join. В противном случае в поле Best

записывается предыдущее найденное значение. Данный процесс организован с использованием Merge.

В пределах разработанного блока (Activity) подписываться на уведомления сервиса можно только в действии Start. Если полученное уведомляющее сообщение нужно расширить и передать дальше, то для этого можно передать полученное сообщение в другое действие блока и выполнить его дополнительную обработку.

11.3. Лабораторная работа 11. Моделирование движения робота при обходе лабиринта

11.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков моделирования движения робота при обходе лабиринта с использованием лазерного дальномера в среде Visual Simulation Environment.

Содержание

1. Создание сцены для моделирования движения робота при обходе лабиринта.
2. Разработки управляющей диаграммы моделирования движения робота при обходе лабиринта с использованием лазерного дальномера.

11.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4.

11.3.3 Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Создайте сцену моделирования движения робота при обходе лабиринта согласно методике 7.2.2. и варианту задания в табл. 7.1. Измените расположение стен в сцене с учетом необходимости исключения отдельностоящих стен и соответственно отсутствия замкнутых маршрутов.

2. Разработайте управляющую диаграмму моделирования движения робота при обходе лабиринта взяв за основу программу, разработанную согласно методике 11.2.1. При разработке примените правило «правой руки» для четных номеров вариантов задания и по правилу «левой руки» для нечетных номеров.

11.4. Контрольные вопросы

1. Прокомментируйте правила для обхода лабиринта, который не является многосвязным?
2. Назовите ограничения присущие правилу «одной руки».
3. Прокомментируйте особенности универсального алгоритма обхода любого лабиринта?
4. Что такое конечный автомат?
5. Пркомментируйте возможности лазерного дальномера в среде Microsoft RDS R4.
6. Сколько блоков Worker используется при построении диаграммы согласно методике 11.2.1.
7. Какие действия выполняются в блоке Worker?

ГЛАВА 12. УПРАВЛЕНИЕ ДВИЖЕНИЕМ РОБОТА С ИСПОЛЬЗОВАНИЕМ ПИД-РЕГУЛЯТОРА

Пропорционально-интегрально-дифференциальный (ПИД) регулятор – устройство в управляющем контуре с обратной связью. Используется в системах автоматического управления для формирования управляющего сигнала с целью получения необходимых точности и качества переходного процесса [7].

ПИД-регулятор относится к наиболее распространённому типу регуляторов. Порядка 90 – 95 % регуляторов, находящихся в настоящее время в эксплуатации, используют ПИД-алгоритм. Причинами столь высокой популярности являются простота построения и промышленного использования, ясность функционирования, пригодность для решения большинства практических задач и низкая стоимость [8].

После появления дешёвых микропроцессоров и аналого-цифровых преобразователей в промышленных ПИД-регуляторах используются автоматическая настройка параметров, адаптивные алгоритмы, нейронные сети, генетические алгоритмы, методы нечёткой логики. Усложнилась структура регуляторов: появились регуляторы с двумя степенями свободы, с применением принципов разомкнутого управления в сочетании с обратной связью, со встроенной моделью процесса. Кроме функции регулирования, в ПИД-контроллер были введены функции аварийной сигнализации, контроля разрыва контура регулирования, выхода за границы динамического диапазона и др.

12.1. Краткие теоретические сведения

ПИД-регулятор формирует управляющий сигнал, являющийся суммой трёх слагаемых, первое из которых пропорционально разности входного сигнала и сигнала обратной связи (сигнал рассогласования), второе – интеграл сигнала рассогласования, третье – производная сигнала рассогласования.

Назначение ПИД-регулятора – в поддержании заданного значения x_0 некоторой величины $x(t)$ с помощью изменения другой величины u. Значение x_0 называется **заданным значением**, а разность $e = (x_0 - x)$ – рассогласованием или отклонением величины от заданной.

Выходной сигнал регулятора u определяется тремя слагаемыми [7]:

$$u(t) = P + I + D = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de}{dt},$$

где K_p, K_i, K_d – коэффициенты усиления пропорциональной, интегральной и дифференциальной составляющих регулятора, соответственно.

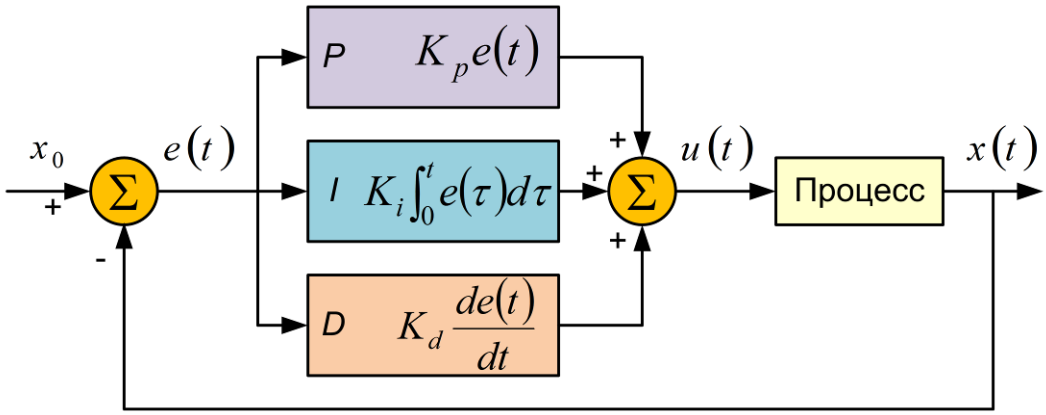


Рисунок 12.1 – Система управления с обратной связью с использованием ПИД-регулятора

Пропорциональная составляющая формулы выходного сигнала ПИД-регулятора вырабатывает выходной сигнал, противодействующий отклонению регулируемой величины от заданного значения, наблюдаемому в данный момент времени. Он тем больше, чем больше это отклонение. Если входной сигнал равен заданному значению, то выходной равен нулю.

Однако при использовании только пропорционального регулятора значение регулируемой величины никогда не стабилизируется на заданном значении. Существует так называемая статическая ошибка, которая равна такому отклонению регулируемой величины, которое обеспечивает выходной сигнал, стабилизирующий выходную величину именно на этом значении.

Интегральная составляющая пропорциональна интегралу от отклонения регулируемой величины. Её используют для устранения статической ошибки. Она позволяет регулятору со временем учесть статическую ошибку.

Дифференциальная составляющая пропорциональна темпу изменения отклонения регулируемой величины и предназначена для противодействия отклонениям от целевого значения, которые прогнозируются в будущем. Отклонения могут быть вызваны внешними возмущениями или запаздыванием воздействия регулятора на систему [19].

Впервые методику расчета параметров ПИД-регуляторы предложили Зиглер и Никольс в 1942 году [39]. В методике применяется два метода настройки ПИД-регуляторов. Один из них основан на параметрах отклика объекта на единичный скачок; второй метод основан на частотных характеристиках объекта управления.

Расчет параметров по формулам не может дать оптимальной настройки регулятора, поскольку аналитически полученные результаты основываются на сильно упрощенных моделях объекта. В частности, в них не учитывается всегда присутствующая нелинейность типа «ограничение» для управляющего воздействия. Кроме того, модели используют параметры, идентифицированные с некоторой погрешностью. Поэтому после расчета параметров регулятора желательно сделать его подстройку.

Подстройку можно выполнить на основе правил, которые используются для ручной настройки. Эти правила получены из опыта, теоретического анализа и численных экспериментов. Они сводятся к следующему [30, 31]:

– увеличение пропорционального коэффициента увеличивает быстродействие и снижает запас устойчивости;

– с уменьшением интегральной составляющей ошибка регулирования с течением времени уменьшается быстрее;

– уменьшение постоянной интегрирования уменьшает запас устойчивости;

– увеличение дифференциальной составляющей увеличивает запас устойчивости и быстродействие.

Правила целесообразно применять после предварительной настройки регулятора по формулам. Попытки настроить регулятор без начального

приближенного расчета коэффициентов могут быть безуспешными. Сформулированные выше правила справедливы только в окрестности оптимальной настройки регулятора.

Информация о параметрах управляемого процесса может поступать на ПИД-регулятор с датчиков. Рассмотрим возможности контроллера Kinect присутствующего в среде Microsoft RDS R4.

Kinect – бесконтактный сенсорный игровой контроллер, первоначально представленный для консоли Xbox 360, и значительно позднее для персональных компьютеров под управлением операционой системы Windows, был разработан фирмой Microsoft. Он позволяет пользователю взаимодействовать с вычислительной системой через устные команды, позы тела и показываемые объекты или рисунки.

Контроллер Kinect состоит из следующих элементов (рис. 12.2) [17]:

1) сенсор глубины (3D depth sensors);
2) цветная видеокамера (RGB camera);
3) подставка (motorized tilt);
4) микрофонная решетка (multi-array mic) состоящая из 4 микрофонов.

Рисунок 12.2 – Внешний вид контроллера Kinect (иллюстрация с сайта https://www.microsoft-careers.com)

Сенсор глубины состоит из инфракрасного проектора, объединенного с монохромной КМОП-матрицей, что позволяет Kinect получать трёхмерное изображение при любом естественном освещении [7].

В Kinect применяется 3D-технология от PrimeSense, которая использует структурированный свет, инфракрасные камеры и специализированный

процессор для измерения расстояния от камеры до сцены. В результате, получается облако точек, состоящее из 307200 измерений расстояния между сенсором и сценой. Облако точек представляет собой структуру трехмерных данных, используемых для представления набора многомерной точки. В 3D облако точек, как правило, представляют точки X, Y, Z и геометрические координаты основной поверхности.

Когда доступна информация о цвете (рис. 12.3), облако точек становится 4D. Назначив цвет для оси Z, облака точек могут быть использованы для создания изображения карты глубины. На рис. 12.4, светлым тоном показаны области, находящиеся ближе, а темным – дальше от сенсора.

Рисунок 12.3 – Облако точек 4D

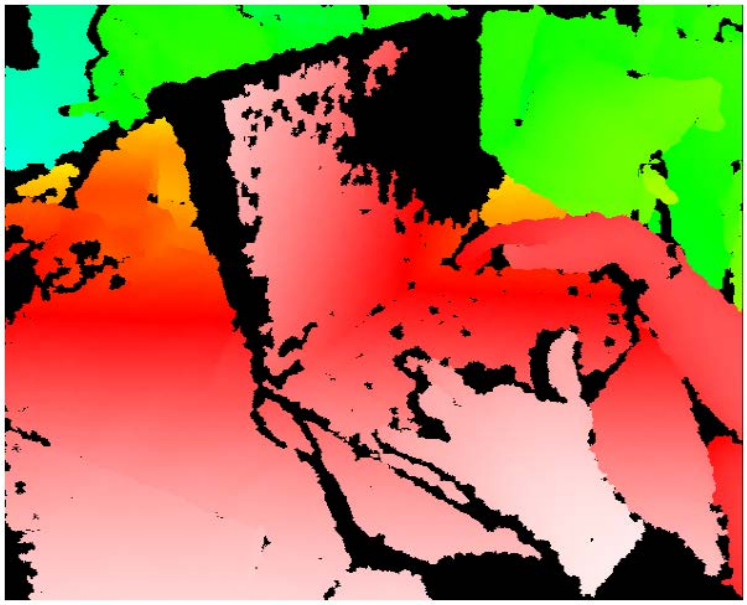


Рисунок 12.4 – Карта глубины (иллюстрация с сайта http://ru.wikipedia.org)

Контроллер Kinect имеет практическую дальность от 0,5 до 3 метров. Облако точек имеет разрешение по X, Y – 640 × 480, а глубина кодируется 11 битами (от 0 до 2047). Диапазон данных не является линейным, и для объектов, находящихся слишком близко, слишком далеко или в «тени» сенсор возвращает значение 2047 (на рис. 12.4 показаны черным цветом).

В основании Kinect находится электрический мотор для изменения положения в вертикальной плоскости так, чтобы подобрать для камеры оптимальный угол обзора.

Фирма Parallax, Inc. разработала роботизированную платформу Eddie Robot Platform (рис. 12.5) – решение для создания мобильного робота с собственным ноутбуком и датчиком Kinect [18]. Данная платформа совместима со средой Microsoft RDS R4.

Рисунок 12.5 – Внешний вид Eddie Robot Platform (иллюстрация с сайта http://www.parallax.com/)

12.2. Методические рекомендации

12.2.1 Методика моделирования движения робота с использованием ПИД-регулятора

1. Откройте файл VPLTutorial4 находящийся в папке Microsoft Robotics Dev Studio 4\samples\VPLTutorials\Tutorial4. Диаграмма моделирования движения робота представлена на рис. 12.6.

2. Перейдите на вкладку GenericDifferentialDrive. Подключите манифест ObstacleAvoidanceDrive in ReferencePlatform2011.

3. Запустите на выполнение среду симуляции, при этом откроется несколько окон:

а) окно среды симуляции (рис. 12.7);

б) окно сервиса интерфейса оператора DirectionDialog (рис. 12.8);

в) окно специализированной панели управления роботом Robot Dashboard, в котором можно получить информацию о состоянии сенсоров, уровне заряда батареи, изменить наклон сенсора и т. д. (рис. 12.9);

г) окно сенсора глубины DepthCam View (рис. 12.10а);

д) окно цветной камеры WebCam View (рис. 12.10б);

е) окно с настройками ПИД регулятора Obstacle Avoidance (рис. 12.11).

4. Организуйте управление мобьной роботизированной платформой с использованием поочередно сервиса DirectionDialog и панели управления Robot Dashboard.

5. Проанализируйте зависимость эффективности управления роботом от угла наклона контролера Kinect.

12.2.2 Методика настройки ПИД-регулятора

1. Настройка пропорциональной компоненты ПИД-регулятора

1.1. Обнулите коэффициенты K_i и K_d. Будем разбираться с K_p, варьируя его значение, скажем, от 1 до 100. Начните со значения $K_p = 1$. Если система очень медленно реагирует на воздействие, то K_p надо увеличивать. Если же система ведет себя неустойчиво, то K_p надо уменьшать.

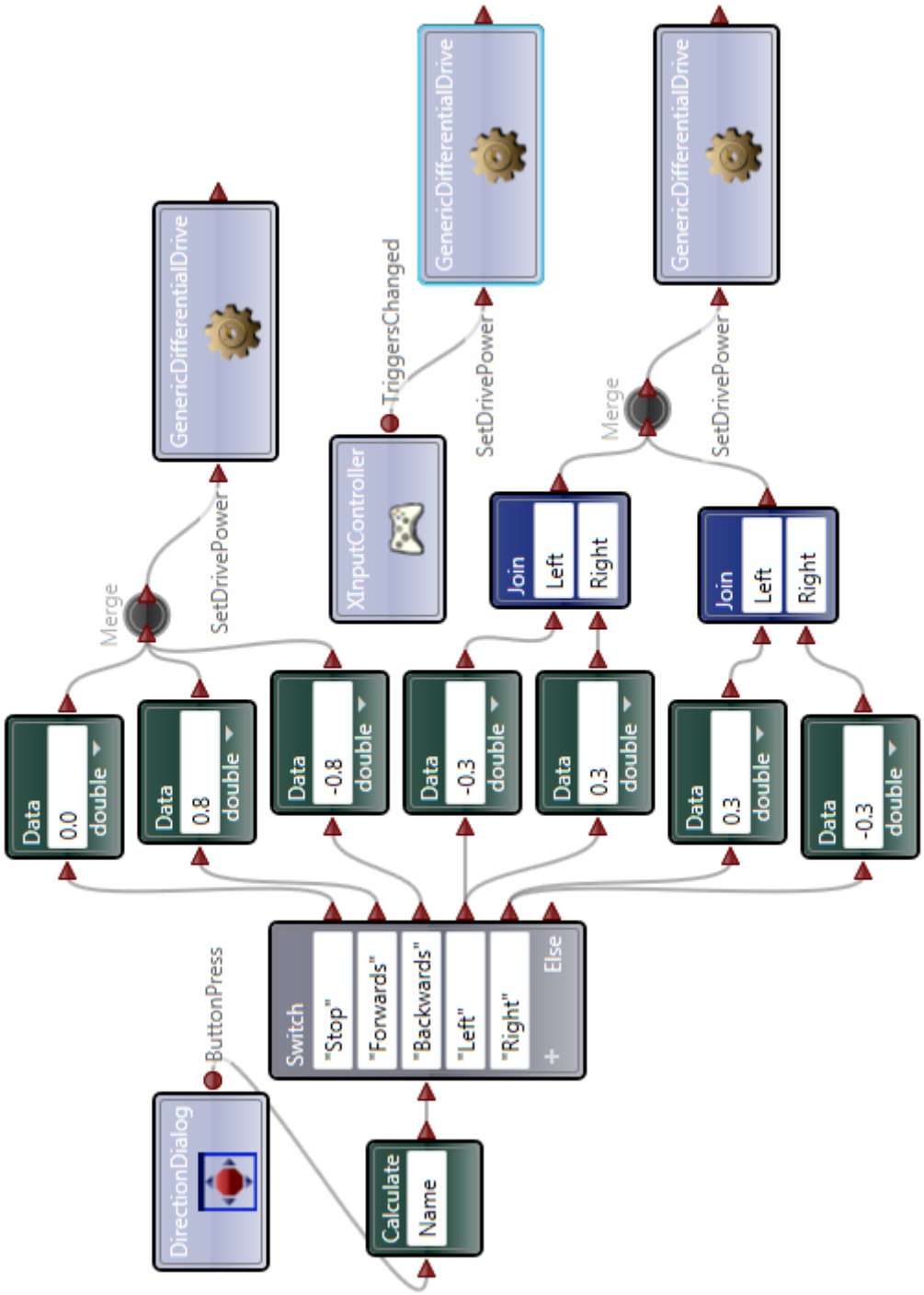


Рисунок 12.6 – Диаграмма моделирования движения робота

Рисунок 12.7 – Окно среды симуляции

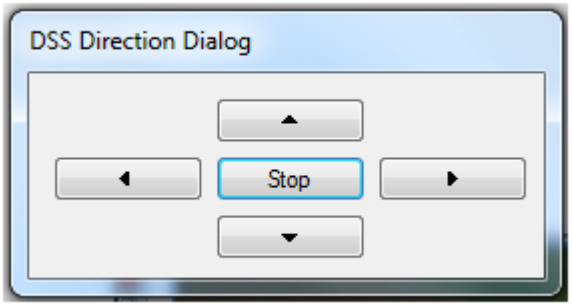


Рисунок 12.8 – Окно сервиса интерфейса оператора DirectionDialog

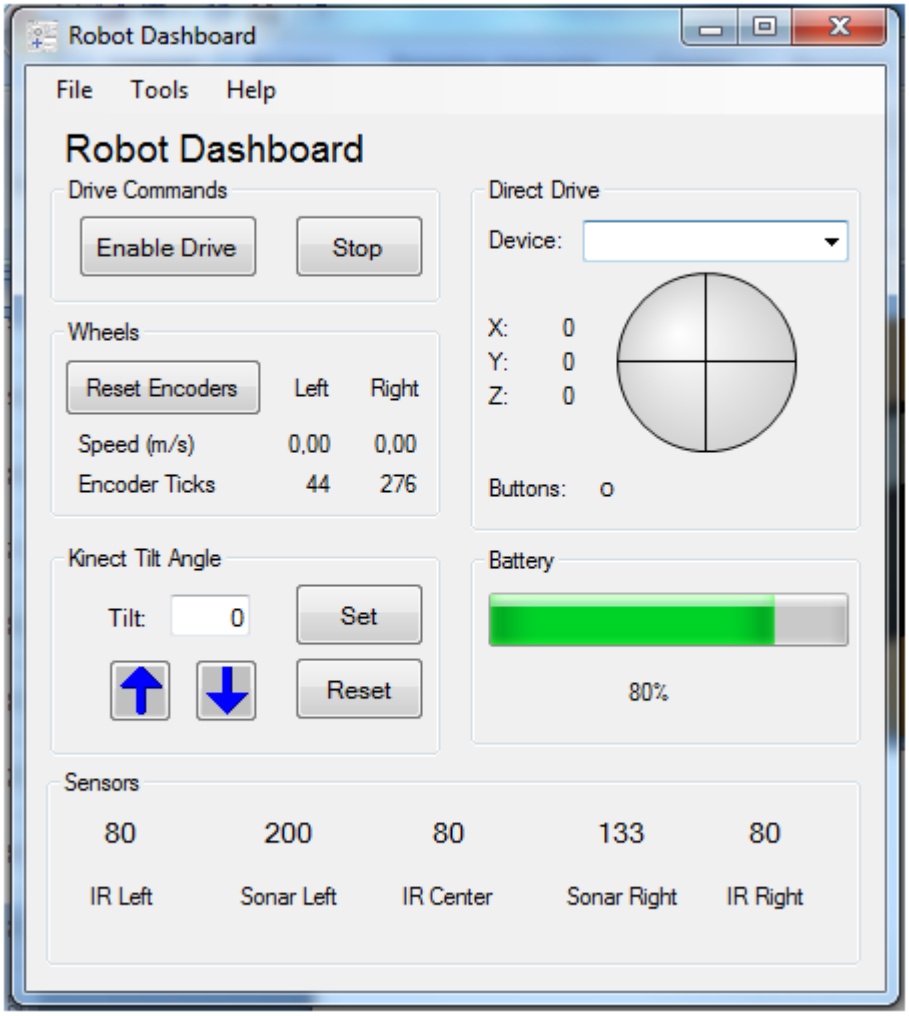
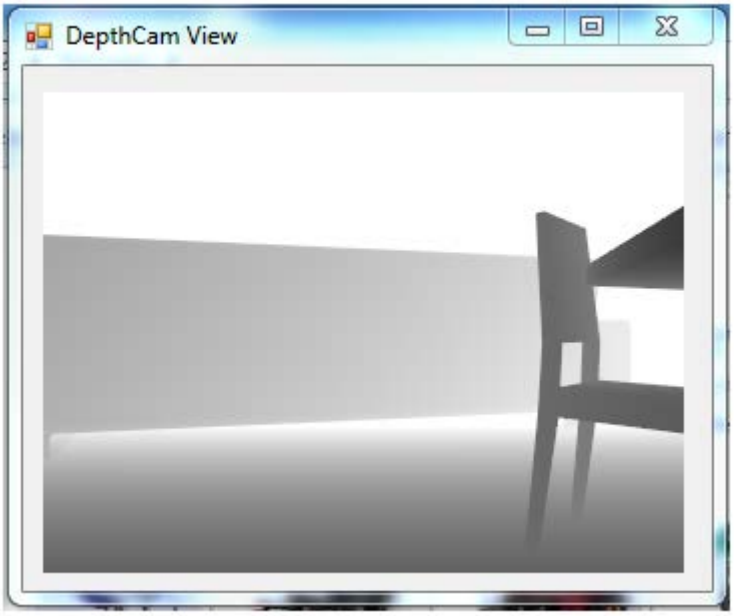


Рисунок 12.9 – Окно панели управления роботом Robot Dashboard

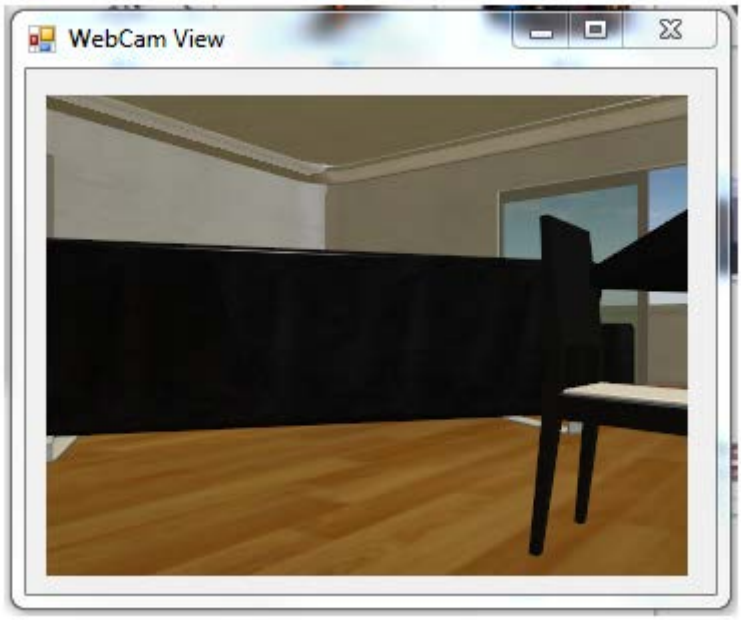
При этом рекомендуется использовать следующий принцип изменения значений коэффициента [11].

1.2. Установите сначала маленькое значение K_p. Допустим, что нестабильной работы системы еще нет.

Далее увеличивайте это значение в 10 раз, пока не начнутся колебания системы управления. Теперь уменьшите значение коэффициента K_p, но не в 10 раз, а в 2 раза. И так до тех пор, пока колебания не прекратятся. И так далее. Таким образом, мы ищем искомое значение, сначала используя большие шаги, а затем все меньшие.



а)



б)

Рисунок 12.10 – Окна сенсора глубины (а) и цветной камеры (б) робота

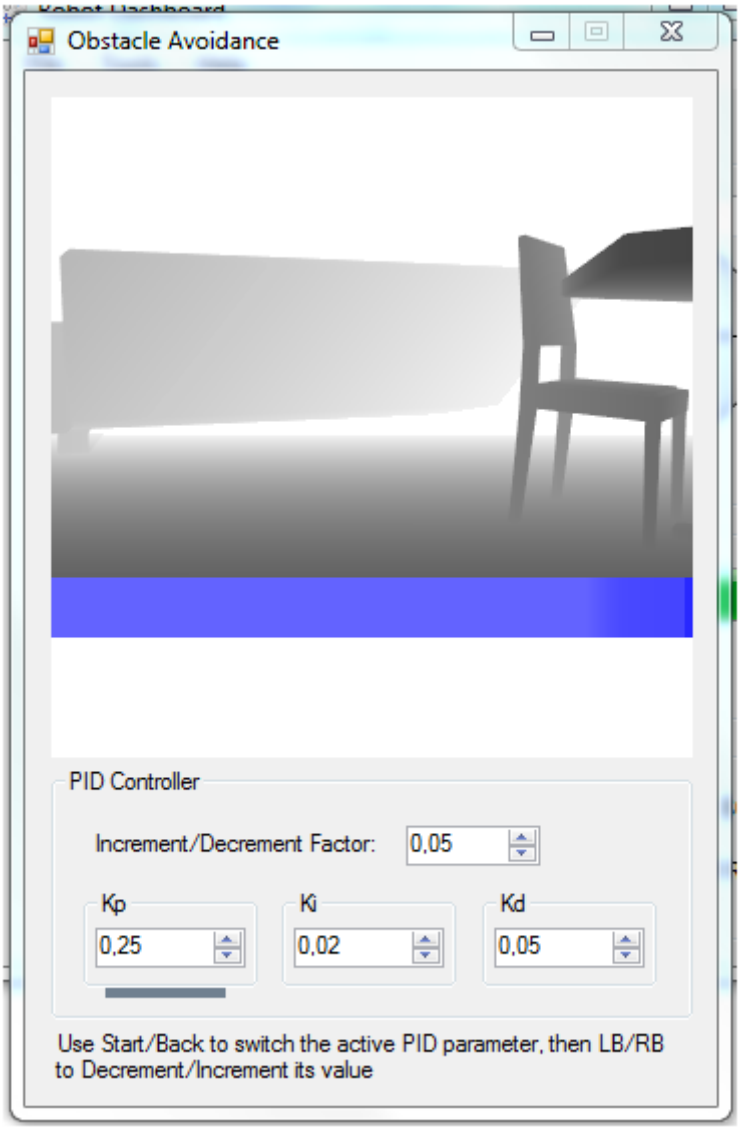


Рисунок 12.11 – Окно с настройками ПИД регулятора

2. Настройка интегральной компоненты ПИД-регулятора

Выберите значение коэффициента интегральной компоненты K_i, которое должно быть мало по сравнению с K_p. В качестве начального значения коэффициента K_i рекомендуется брать число от 0,0001 до 0,01.

Процедура подбора коэффициента K_i точно такая же, как и коэффициента пропорциональной составляющей (сначала большие шаги, а затем маленькие). Слишком большое значение коэффициента K_i также проявляется в появлении неустойчивости в работе системы управления.

3. Настройка дифференциальной компоненты ПИД-регулятора

Дифференциальную компоненту ПИД-регулятора целесообразно вводить в случае если шумы в нашей системе на слишком велики.

3.1. Для настройки K_d установите значение этого коэффициента, равное 0. Далее установите какое-нибудь небольшое значение коэффициента пропорционального звена K_p (например, $K_p = 1$). Главное, что значение K_p должно быть таким, чтобы система при нулевом значении K_d не совершала колебаний.

3.2. Установите какое-нибудь небольшое начальное значение коэффициента K_d (например, $K_d = 0{,}1$). Будем увеличивать коэффициент K_d до тех пор, пока не станут проявляться ошибочные колебания системы, вызванные малыми шумами. При этом колебания от слишком большого коэффициента происходят значительно быстрее, чем колебания от недостаточного коэффициента. Рекомендуется устанавливать коэффициент в половину или четверть от того, при котором начинаются колебания от слишком большой его величины. Главное в этом процессе – вовремя убедиться в том, что поведение системы является адекватным (роботизированная платформа ведет себя на трассе движения хорошо).

Описанные процедуры настройки коэффициентов содержат много субъективизма [11]. Главным критерием того, что мы нашли подходящие коэффициенты (не оптимальные, конечно), является внешнее поведение тележки. Но помимо таких сугубо экспериментальных подстроек, имеются и вполне объективные принципы, которые надо соблюдать при создании регулятора.

1. Скорость работы программы. Управляющая программа должна работать максимально быстро. Это означает, что необходимо использовать максимально лаконичные алгоритмы и простые операции.

2. Равномерность работы программы. Не менее важно то, чтобы формирование управляющего воздействия происходило в как можно более равномерные моменты времени. Нельзя допускать, чтобы программа

«задумалась» на непредсказуемое время, т. к. в этом случае сбой дадут и интегральное, и дифференциальное звено.

3. Частота сбора данных и выполнения цикла управления. Помимо стабильности частоты сбора, необходимо определиться с тем, какова должна быть частота управления. Дело в том, что если частота управления слишком маленькая, то мы не получим эффективную систему. Более того, при малой частоте управления мы можем получить систему, которую вообще невозможно стабилизировать. Но и при слишком большой частоте мы тоже получим плохой результат: начнут возникать серьезные шумы в дифференциальной компоненте, а также начнет переполняться интегратор.

12.3. Лабораторная работа 12. Моделирование движения робота с использованием ПИД-регулятора

12.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков моделирования движения робота с использованием ПИД-регулятора в среде Visual Simulation Environment.

Содержание

1. Создание сцены для моделирования движения робота с использованием ПИД-регулятора.
2. Настройка параметров ПИД-регулятора.

12.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4.

12.3.3 Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.

2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Запустите сцену моделирования движения робота с использованием ПИД-регулятора и ознакомтесь с особеностями её работы согласно методике 12.2.1.

2. Проведите настройку ПИД-регулятора согласно методике 12.2.2. Добейтесь максимальной скорости движения при преодолении различных препятствий.

12.4. Контрольные вопросы

1. Перечислите особенности работы ПИД-регулятора.
2. Какие составные части включает ПИД-регулятор?
3. Что такое пропорциональная составляющая ПИД-регулятора?
4. Что такое интегральная составляющая ПИД-регулятора?
5. Что такое дифференциальная составляющая ПИД-регулятора?
6. Назовите правила ручной настройки ПИД-регулятора.
7. Из каких частей состоит контроллер Kinect?

ГЛОССАРИЙ

Адресная привязка (*англ.* Fixup) – перевод относительных адресов, адресов вызова функций и т. д. в абсолютные значения адресного пространства исполнения.

Базовый блок – блок, позволяющий управлять процессом выполнения сервисов.

Безопасная копия (*англ.* Secure copy) – локальная копия данных переданного буфера.

Веб-камера – цифровая видео- или фотокамера, фиксирующая изображения в реальном масштабе времени.

Встроенный указатель (*англ.* Embedded Pointer) – указатель, передаваемый в функцию API внутри структуры данных или буфера.

Графический движок – программа, работающая в режиме реального времени и предназначенная для визуализации сцены в симуляторе.

Дальномер – устройство для измерения расстояния до объекта.

Двухуровневая модель драйверов (*англ.* Layered Driver Model) – модель драйверов, когда реализация функциональности драйвера разбивается на две части: не зависящую от платформы и зависящую от аппаратной реализации.

Динамический объект – объект, находящийся под контролем физического движка. После создания такого объекта его позиция и движение определяются только гравитацией и взаимодействием с другими объектами.

Дифференциальный привод – шасси, в состав которого входят два независимо управляемых колеса и пассивное колесо, используемое для баланса.

Драйвер пользовательского режима (*англ.* User Mode Driver) – драйвер, который загружается в специализированные пользовательские процессы (Udevice.exe).

Драйвер потокового интерфейса или потоковый драйвер (*англ.* Stream Interface Driver или Stream Driver) – драйвер, реализующий потоковый интерфейс.

Драйвер режима ядра (*англ.* Kernel Mode Driver) – драйвер, который загружается в адресное пространство ядра операционной системы.

Драйвер, зависящий от платформы или платформенный драйвер устройства (*англ.* Platform Dependent Driver или Platform Device Driver, PDD) – библиотека, реализующая часть функционала драйвера, относящегося к конкретной аппаратной реализации, предоставляя библиотеке MDD соответствующего класса драйверов требуемую функциональность. Компонуясь с библиотекой MDD. для соответствующего класса драйверов, собирается в конечный драйвер.

Инфракрасный дальномер – устройство, позволяющее определять расстояние до объектов с помощью инфракрасного излучения.

Каталог (*англ.* catalog) – база возможностей ОС, компонентов и модулей.

Кинематический объект – объект, расположение которого определяется вручную. Кинематический объект может взаимодействовать с другими объектами. Примером кинематического объекта является доска, висящая в воздухе.

Компонент (*англ.* component) – минимальная единица функциональности ОС, которую можно добавить в дизайн операционной системы.

Лазерный дальномер – устройство, позволяющее определять расстояние до объектов с помощью лазерного луча.

Манифест – XML-файл, содержащий список запускаемых сервисов, связанных с диаграммой.

Маршалинг (*англ.* Marshalling) – отображение; подготовка данных/указателя к использованию в другом процессе.

Менеджер устройств (*англ.* Device Manager) – часть ядра системы, отвечающая за работу с потоковыми драйверами.

Менеджер хранилищ (*англ.* Storage Manager) – подсистема операционной системы, отвечающая за работу с носителями, включая, работу с разделами, файловыми системами и фильтрами.

Модельный драйвер устройства (*англ.* Model Device Driver, MDD) – библиотека, реализующая общую часть функционала определенного класса драйверов. Компонуясь с библиотекой PDD (драйвером, зависящим от платформы), собирается в конечный драйвер.

Модуль (*англ.* module) – исполняемый в системе код (EXE и DLL), часть образа операционной системы. Модули состоят или собираются из компонентов.

Независимый от ядра транспортный слой (*англ.* Kernel Independent Transport Layer, KITL) – слой который абстрагирует коммуникационный протокол от аппаратной реализации соединения (обычно, устройства со станцией разработчика).

Нить (*англ.* Fiber) – единица исполнения, управляемая приложением.

Образ времени исполнения (*англ.* Run-Time Image) – бинарный файл, содержащий все модули из дизайна ОС и собранный в соответствии с конфигурационными файлами для указанного BSP.

Объект – элемент сцены.

Одометр – устройство для подсчета количества оборотов колеса.

Пакет поддержки платформы (платы) (*англ.* Board Support Package, BSP) состоит из OAL, драйверов и файлов конфигурации, предоставляет поддержку определенной платформы (платы).

Поток (*англ.* Thread) – базовая единица исполнения, управляемая планировщиком; каждый процесс имеет, по крайней мере, один поток, обычно называемый, основным потоком; количество потоков ограничено количеством доступных дескрипторов в процессе.

Поток обработки прерывания (*англ.* Interrupt Service Thread, IST) – поток обработки прерывания; часть драйвера; выполняет основную часть работы.

Приложение (*англ.* Application) – логически сгруппированный исполняемый код.

Примитив/объект синхронизации (*англ.* Synchronization primitive/object) – объект, позволяющий выполнять задачи синхронизации в многопоточной среде.

Проверка доступа (*англ.* Access Checking) – проверка, что вызывающий процесс имеет достаточно привилегий для доступа к буферу.

Программа (на языке VPL) – диаграмма потоков данных, передаваемых между сервисами и блоками.

Прокси драйвер пользовательского режима (*англ.* UI Proxy User Interface) – драйвер пользовательского режима, используемый драйвером режима ядраоперационной системы для отображения пользовательского интерфейса. Также и название технологии ядра, которая это реализует.

Процесс (*англ.* Process) – экземпляр приложения; статический контекст, в котором исполняются один или более потоков.

Распределенное приложение – приложение, которое выполняется на нескольких ЭВМ.

Результирующий выход – выход, на который отправляется результат работы сервиса или блока.

Рендеринг – процесс получения изображения по модели с помощью графического движка.

Родной код (*англ.* Native code) – код, который компилируются в бинарный код; для исполнения не требует никакой среды исполнения.

Сервис – интерфейс к аппаратному или программному обеспечению робота.

Сервис рефлектор (*англ.* Reflector Service) – центральная часть фрейворка драйверов режима пользователя. Предоставляет необходимый сервис драйверам режима пользователя. Скрывает от остальной части системы, разницу между драйверами режима ядра операционной системы и пользовательского режима.

Симулятор – программа, включающая в себя графический и физический движки.

Синхронный доступ (*англ.* Synchronous Access) – доступ к буферу во время вызова API.

Слой абстракции ядра (*англ.* OEM Abstraction Layer, OAL) – низкоуровневый аппаратно-зависимый код, абстрагирующий ядро операционной системы от специфики аппаратной реализации.

Сонар – устройство для определения расстояния до объектов с помощью звуковых волн.

Статический объект – объект, который в процессе симуляции не изменяет свою позицию и состоит из фигур с нулевой массой. Статический объект может взаимодействовать с другими объектами.

Сцена – совокупность объектов, находящихся в виртуальном мире и взаимодействующих друг с другом.

Уведомляющий выход – выход блока, используемый для вывода сообщений при изменении внутреннего состояния сервиса (например, сервис, связанный с некоторым сенсором, использует этот выход для выдачи данных о состоянии сенсора).

Указатель-параметр (*англ.* Pointer Parameter) – указатель, передаваемый в функцию API как параметр.

Управляемый код (*англ.* Managed code) – код, написанный на C#|VB.NET под .NET Compact Framework.

Физический движок – программа, предназначенная для моделирования физических законов реального мира в виртуальном мире с той или иной степенью точности.

Фреймворк драйверов режима пользователя (*англ.* User Mode Driver Framework) – инфраструктура поддержки драйверов режима пользователя.

Хост драйверов режима пользователя (*англ.* User Mode Drivers Host) – процессы Udevice.exe, в которые загружаются драйвера пользовательского режима.

Ярлык (*англ.* Shortcut) – ссылка на программу или команду с параметрами, щелчок по которой приводит к их запуску.

Concurrent and Coordination Runtime (CCR) – среда организации параллельной обработки данных.

Decentralized Software Services (DSS) – среда, которая позволяет запускать алгоритмы обработки данных на разных ЭВМ, организовать асинхронное взаимодействие процессов управления различными подсистемами робота.

Visual Programming Language (VPL) – язык, предназначенный для разработки программ управления роботами. Программа представляется в виде последовательности блоков, которые выполняют обработку данных, и связей между ними. VPL рассчитан на управление, как реальными роботами, так и моделями роботов в симуляторе.

Visual Simulation Environment (VSE) – среда визуализации, которая позволяет экспериментировать с моделями роботов, тестировать алгоритмы управления роботами.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА И ИСТОЧНИКИ

Основная литература

1. Гай В. Е. Microsoft Robotics Developer Studio. Программирование алгоритмов управления роботами. – М. : ЭКОМ Паблишерз, 2012. 184 с.

2. Лучин Р. М. Программирование встроенных систем: от модели к роботу. – СПб. : Наука, 2011. 184 с.

3. Филиппов С. А. Робототехника для детей и родителей. – СПб. : Наука, 2011. 263 с.

4. Яковлев С. В. Системы реального времени : учебное пособие. – Ставрополь : СевКавГТУ, 2012. 308 с.

Дополнительная литература и источники

5. Бесекерский В. А., Попов Е. П. Теория систем автоматического управления – 4-е изд., перераб. и доп. – СПб. : Изд-во «Профессия», 2004. 752 с.

6. Большой политехнический энциклопедический словарь : CD-ROM. – М. : Мультитрэйд, 2004.

7. Википедия. Свободная энциклопедия [Электронный ресурс] // http://ru.wikipedia.org.

8. Волянский С. М., Волянская Я. Б. Сравнительный анализ регуляторов, применяемых в системах управления энергосберегающим электроприводом постоянного тока. // Вестник КГПУ им. Михаила Остроградского. – 2008. – Выпуск 4(51). – С. 106-108.

9. Гай В.Е. Стань робототехником // Хакер. 2013. № 1(168). С. 96 – 101.

10. Загружаемые файлы для Visual Studio [Электронный ресурс] // http://www.visualstudio.com/downloads/download-visual-studio-vs.

11. Карпов В. Э. ПИД-управление в нестрогом изложении. Лаборатория «Робототехника», Московский институт электроники и математики НИУ ВШЭ [Электронный ресурс] // http://robofob.ru/materials/articles/pages/Karpov mobline1.pdf.

12. Колотов А. Не-алгоритмы: черно-белое движение. Часть X. Lego Mindstorms NXT: робототехника для школ и ВУЗов Нижнего Новгорода [Электронный ресурс] // http://nnxt.blogspot.ru/.

13. Колотов А. Робот для состязаний: выход из лабиринта. Lego Mindstorms NXT: робототехника для школ и ВУЗов Нижнего Новгорода [Электронный ресурс] // http://nnxt.blogspot.ru/.

14. Международный университет инновационных технологий [Электронный ресурс] // http://moodle.intuit.kg.

15. Меженин А. В. Опыт обучения программированию роботов на платформе Robotics Developer Studio. // Материалы Всероссийской научно-практической конференции «Информационные технологии в науке и образовании» [Электронный ресурс] // http://chb.ito.edu.ru/2012/section/194/93595/.

16. Негосударственное образовательное частное учреждение высшего профессионального образования «Национальный Открытый Университет «ИНТУИТ» [Электронный ресурс] // http://www.intuit.ru.

17. Новости Русского MSDN [Электронный ресурс] // http://blogs.msdn.com.

18. Официальный сайт компании Parallax, Inc. [Электронный ресурс] // http://www.parallax.com.

19. ПИД-регулятор [Электронный ресурс] // http://ru.wikipedia.org/wiki/ПИД-регулятор.

20. ПИД-регуляторы: принципы построения и модификации [Электронный ресурс] // www.cta.ru/cms/f/342946.pdf.

21. Попов Е. П., Письменный Г. В. Основы робототехники: Введение в специальность. – М. : Высшая школа, 1990. 224 с.

22. Программное создание собственного виртуального робота в Visual Studio для пакета MRDS [Электронный ресурс] // umc.info.urfu.ru/images/documents/MRDS.doc.

23. Прохождение лабиринта: правила и алгоритмы. myROBOT.ru – Роботы, робототехника, микроконтроллеры. [Электронный ресурс] // http://myrobot.ru/index.php.

24. Русскоязычное сообщество разработчиков управляющих программ для роботов в среде MRDS. [Электронный ресурс] // http://www.mrds.ru.

25. Сайт компании Boston Dynamics. [Электронный ресурс] // http://www.bostondynamics.com/index.html.

26. Сайт компании MOVEMEN. Мехатронные станочные системы [Электронный ресурс] // http://www.movemen.ru.

27. Сайт Microsoft Robotics Developer Studio 4 [Электронный ресурс] // http://www.microsoft.com/robotics/.

28. Сара Морган Моделирование мира с помощью Microsoft Robotics Studio [Электронный ресурс] // http://msdn.microsoft.com/ru-ru/magazine/cc546547.aspx

29. Черчнев Д. А. Технология разработки программного обеспечения : учебное пособие. – Ташкент : Изд-во «Mehnat», 2004. 223 с.

30. Энциклопедия АСУ ТП. [Электронный ресурс] // http://bookasutp.ru/.

31. Astrom K.J., Hagglund T. Advanced PID control. – ISA – The Instrumentation, Systems, and Automation Society, 2006. 460 p.

32. Build Simulation Environment [Электронный ресурс] // http://pds17.egloos.com/pds/200912/09/42/FirstBeginner_SPL_Basic_03_Build Environment.ppp.

33. HelloApps Co., Ltd. [Электронный ресурс] // http://www.helloapps.com.

34. Johns K., Taylor T. Professional Microsoft Robotics Developer Studio. – Indianapolis, Indiana : Wiley Publishing, Inc., 2008. 826 p.

35. Microsoft Robotics. Параллельная обработка данных. Хабрахабр. [Электронный ресурс] // http://habrahabr.ru/post/204466/.

36. Microsoft XNA Game Studio 4.0 [Электронный ресурс] // http://www.microsoft.com/en-us/download/details.aspx?id=23714.

37. MRDS: how to add your custom simulated objects. [Электронный ресурс] // http://blog.martinperis.com/2011/02/mrds-how-to-add-your-custom-simulated.html.

38. Sara Morgan Programming Microsoft Robotics Studio. – Washington : Microsoft Press, 2008. 288 p.

39. Ziegler J. G., Nichols N. B. Optimum settings for automatic controllers. // Transactions of the A.S.M.E., vol. 64. 1942. p. 759 – 768.

ПРИЛОЖЕНИЕ 1. СОДЕРЖАНИЕ, ФОРМА И ПРАВИЛА ОФОРМЛЕНИЯ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

Содержание

1. Титульный лист с указанием названия лабораторной работы, ФИО студента.
2. Цель работы.
3. Основная часть, в которой должны присутствовать комментарии о производимых действиях при выполнении задания, предлагаемого в лабораторной работе.
4. Выводы по результатам выполнения задания, предлагаемого в лабораторной работе.

Текст отчета подготавливается в текстовом редакторе, согласно ГОСТ 2.105-95 «Общие требования к текстовым документам», и предоставляется в электронном и в распечатанном виде в одном экземпляре. К отчету прилагаются файлы, созданные в ходе выполнения задания, предлагаемого в лабораторной работе.

ПРИЛОЖЕНИЕ 2. КРАТКОЕ ОПИСАНИЕ СРЕДЫ MICROSOFT ROBOTICS DEVELOPER STUDIO

В 2006 году компания Microsoft объявила о создании платформы Robotics Developer Studio (RDS). Microsoft RDS – это Windows-ориентированная среда разработки приложений для робототехники и симуляции. На данный момент актуальной является версия Microsoft Robotics Developer Studio 4.

Среди особенностей: язык визуального программирования Microsoft Visual Programming Language (VPL), Web- и Windows- ориентированные интерфейсы, виртуальная среда симулирования Visual Simulation Environment (VSE), упрощенный доступ к датчикам, микроконтроллеру и исполнительным механизмам робота, поддержка языка программирования C#, библиотеки для многопоточного программирования Concurrency and Coordination Runtime (CCR) и распределенного выполнения приложений Decentralized Software Services (DSS), поддержка многих робототехнических платформ (Eddie, BoeBot, CoroBot, iRobot, LEGO NXT и т.д.). На рис. П2.1 приведено окно среды программирования VPL с готовой программой. На рис. П2.2 изображено окно симулятора среды с моделью робота Eddie фирмы Parallax.

Минимальные системные требования:

1. Видеокарта совместимая с Microsoft DirectX 9.0c.
2. Процессор Dual-Core.
3. 10 Гб сбодного дискового пространства.
4. 2 Гб оперативной памяти.
5. Разъем USB.
6. Bluetooth-адаптер.
7. Операционная система Windows 7.
8. Среда программирования Visual Studio C# 2010 (достаточно версии Exress).

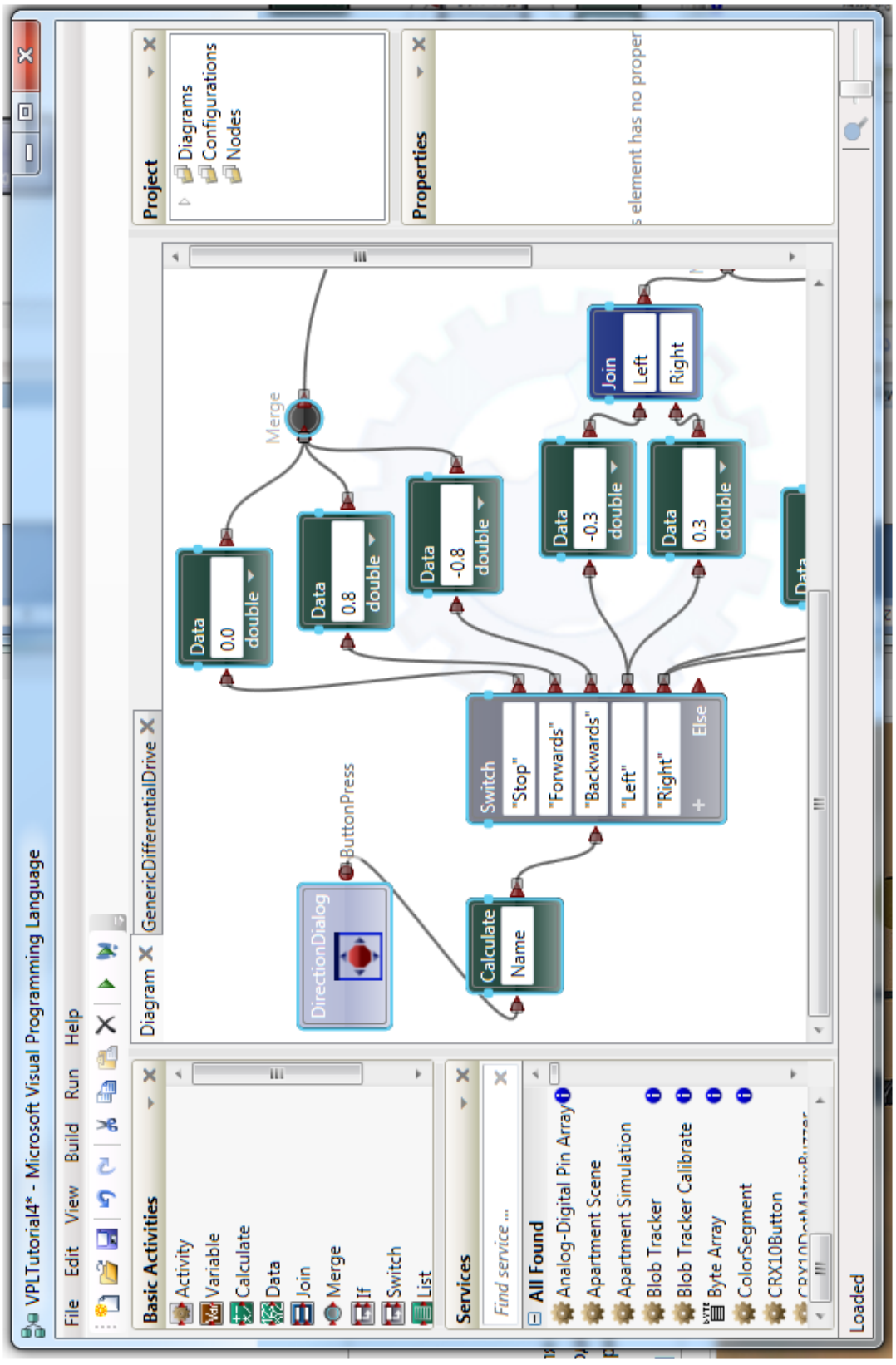


Рисунок П2.1 – Окно среды программирования VPL

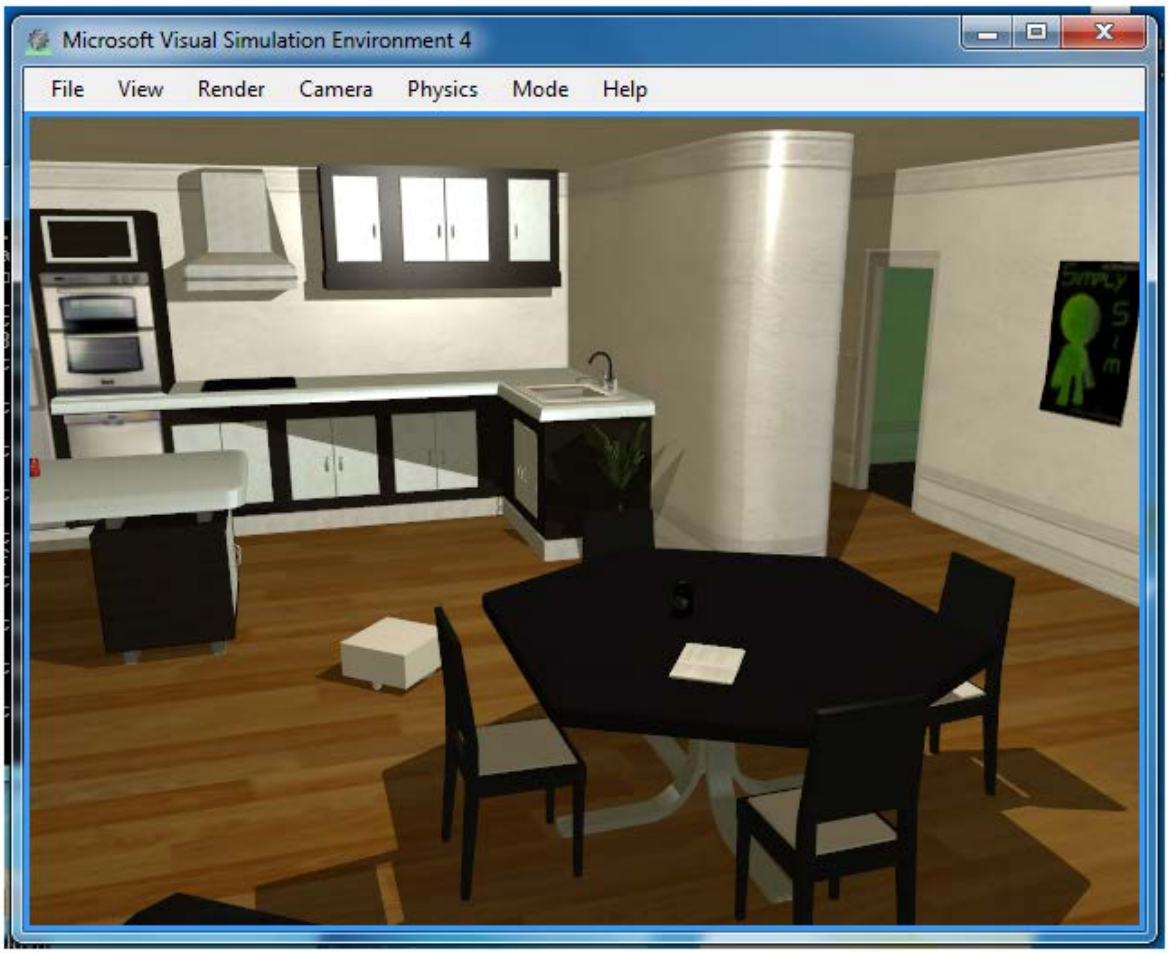


Рисунок П2.2 – Окно среды симулирования VSE

Развертывание платформы Microsoft RDS 4. Наиболее актуальную версию программы можно загрузить на официальной странице продукта компании Microsoft (www.microsoft.com/robotics). После завершения скачивания программы MRDS запускаем установочный файл и следуем указаниям мастера установки (рис. 4.3).

Особое условие: имя учетной записи, в которой будет устанавливаться платформа RDS, должна состоять из латинских букв, в противном случае симулятор будет работать не корректно.

После инсталляции в меню Пуск появиться пункт Microsoft Robotics Developer Studio 4. Выберем из списка строку Build All Samples, это необходимо для того что бы были доступны все примеры программ предоставляемые в справочном руководстве.

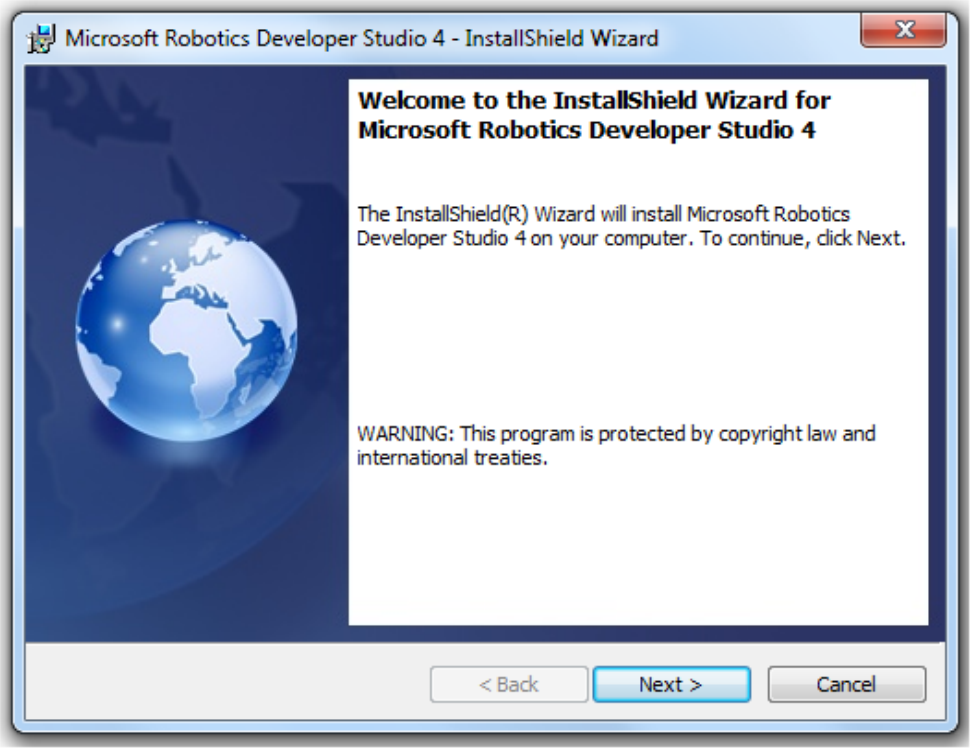


Рисунок П2.3 – Окно мастера установки платформы MRDS

При отладке программы написанной в RDS, в момент, когда запускается браузер, может появиться окно, запрашивающее логин и пароль. Для того, что бы отключить систему защиты, необходимо открыть директорию содержащую файлы MRDS, найти папку **store**, и в ней с помощью блокнота создать xml-файл содержащий следующие строки:

```
<?xml version="1.0"?>
<SecuritySettings
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/robotics/2008/02/security.html">
   <AuthenticationRequired>false</AuthenticationRequired>
   <OnlySignedAssemblies>false</OnlySignedAssemblies>
   <Users />
</SecuritySettings>
```

ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ СРЕДЫ VPL

Для запуска среды в меню Пуск найдите пункт Microsoft Robotics Developer Studio 4, выберите в нем Visual Programming Language 4. После загрузки среды появится окно с панелями и окнами (рис. П2.4).

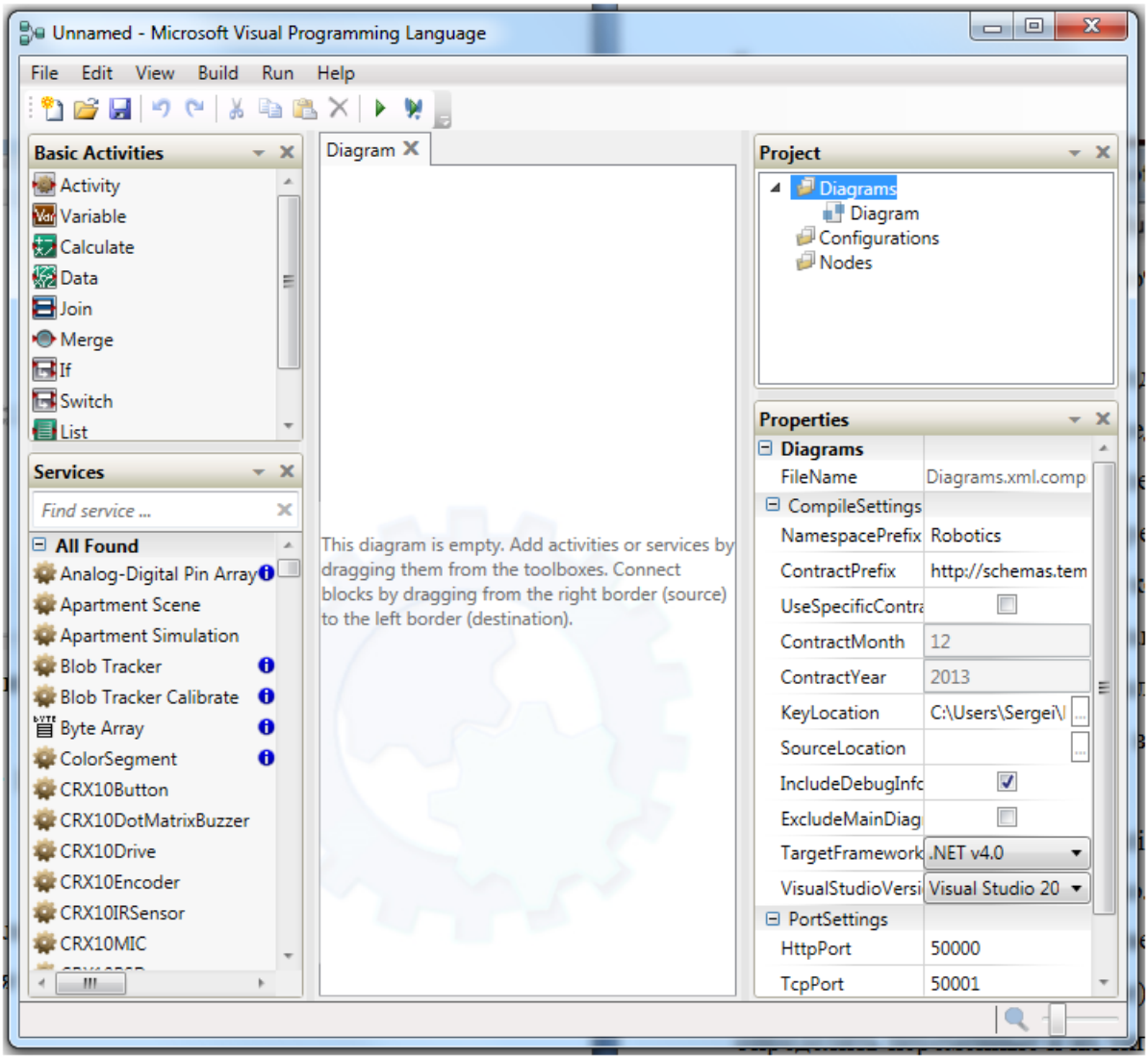


Рисунок П2.4 – Пользовательский интерфейс среды VPL

Пользовательский интерфейс среды VPL состоит из нескольких элементов.

1. Панель меню содержит следующие раскрывающиеся списки (рис. П2.5).

1.1. File – в данном меню содержатся стандартные команды для работы с проектом:

– New (Создать) – новый проект;

– Open (Открыть) – открыть существующий проект;

– Save (Сохранить) – сохранить текущий проект в директорию по умолчанию;

– Save as (Сохранить как) – сохранить проект с указанием директории;

– Print (Печать) – вывод на печать созданной диаграммы;

– Add Diagram (Добавить диаграмму) – добавление в проект диаграммы;

– Recent Project (Последний проект) – открытие последнего созданного проекта;

– Exit (Выход) – выход из среды VPL.

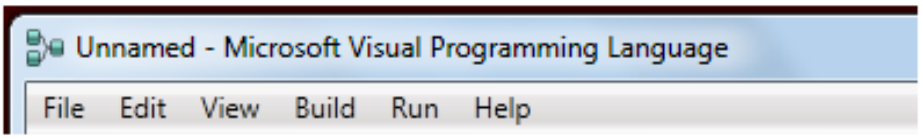


Рисунок П2.5 – Панель меню

1.2. Edit – в данном меню содержатся команды редактирования диаграммы:

– Undo (Отменить) – предыдущая выполненная операция;

– Redo (Повторить) – следующая выполненная операция;

– Cute (Вырезать) – вырезать элемент диаграммы;

– Copy (Копировать) – копировать элемент диаграммы;

– Paste (Вставить) – вставить элемент диаграммы;

– Delete (Удалить) – удалить элемент диаграммы;

– Insert (Вставить) – вставить элемент диаграммы (раскрывается список);

– Action and Notifications (Действия и Уведомления) – вызов диалогового окна, которое позволяет добавить или отредактировать входные и выходные значения, а так же уведомления;

– Variables (Переменные) – вызов диалогового окна, которое позволяет определить переменные и их тип;

– Connections (Соединения) – вызов диалогового окна, для редактирования данных пересылаемых между блоками и сервисами;

– Data Connections (Данные соединения) – вызов диалогового окна, для редактирования соединения;

– Set Configuration (Конфигурация сервиса) – открывает вкладку в рабочем поле для определения конфигурации (манифеста) сервиса.

1.3. View – в данном меню содержаться команды конфигурирования пользовательского интерфейса:

– Toolboxes (Инструменты) – раскрывает список с помощью которого можно скрыть или отобразить в среде VPL окна базовых блоков, сервисов, проекта, свойств, ошибок;

– Diagram (Диаграмма) – раскрывает список в котором можно переключаться между созданными диаграммами;

– Reload Services (Перезагрузить сервисы) – обновляет список доступных сервисов в окне сервисов;

– Grid (Сетка) – показывает или скрывает сетку в рабочем окне диаграммы;

– Toolbar (Панель инструментов) – показывает или скрывает кнопочную панель;

– Align (Выровнять) – раскрывает список с помощью которого можно выравнивать блоки в диаграмме относительно левого края, правого, нижнего верхнего, центрировать по вертикали или по горизонтали;

– Bring to front (На передний план) – располагает блок на переднем плане относительно других элементов в диаграмме;

– Send to back (На задний план) – перемещает блок на задний план относительно других элементов в диаграмме.

1.4. Build – содержит одну команду – Compile as a Service (Компилировать как сервис) – компиляция текущего проекта как сервиса.

1.5. Run – содержатся команды относящиеся к запуску проекта:

– Start (Запуск) – запуск текущего проекта;

– Debug Start (Запуск отладки) – отладка текущего проекта;

– Run Compiled Services (Запуск скомпилированных сервисов) – запускает ранее скомпилированные сервисы;

– Run on distributed nodes (Запуск на распределенных узлах) – запускает на выполнение распределенный в сети проект;

– Port Settings (Настройки портов) – показывает диалоговое окно, которое позволяет программисту установить порты, которые использует проект, когда он запущен.

1.6. Help – данное меню содержит справочную информацию:

– **Contents (Содержание)** – запускает файл помощи платформы MRDS;

– About – (О продукте) – показывает информацию об авторском праве и о версии VPL.

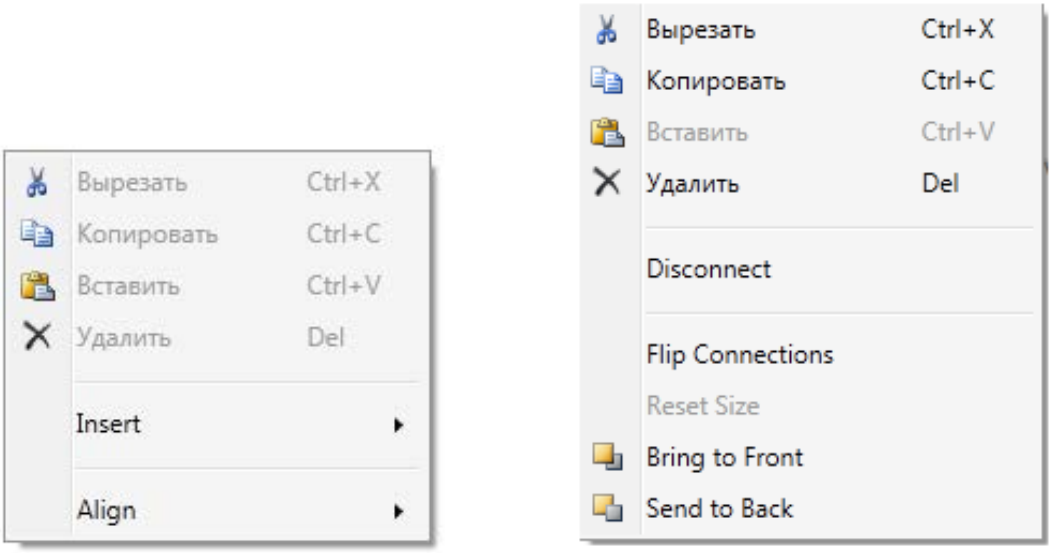
2. Панель кнопок дублирует некоторые пункты из панели меню (рис. П2.5):

1. New (Создать);
2. Open (Открыть);
3. Save (Сохранить);
4. Undo (Отменить);
5. Redo (Повторить);
6. Cut (Вырезать);
7. Copy (Копировать);
8. Paste (Вставить);
9. Delete (Удалить);
10. Start (Запуск);
11. Debug Start (Запуск отладки)**.**

Рисунок П2.5 – Внешний вид панели кнопок

3. Контекстное меню вызывается нажатием правой кнопки мыши, когда курсор находится в рабочем окне диаграммы (рис. П2.6а). Контекстное меню позволяет выполнить следующие операции:

1. Cut (Вырезать);
2. Copy (Копировать);
3. Paste (Вставить);
4. Delete (Удалить);
5. Insert (Добавить);
6. Align (Выровнять).



а) б)

Рисунок П2.6 – Контекстное меню VPL

Если выделить какой-либо базовый блок или сервис, то в контекстном меню появляются следующие операции (рис. 4.8 б):

1. Flip Connection – меняет местами входные и выходные элементы на блоке или сервисе;
2. Disconnect (Разъединить) – разъединяет блоки;
3. Reset Seize (Сбросить размер) – Сбрасывает установленный пользователем размер блока;
4. Bring to Front (На передний план);
5. Send to Back (На задний план).

4. Окна базовых блоков, сервисов, проекта, свойств, ошибок

Данные окна можно свободно перемещать в среде VPL, а так же перемещать их за ее пределы, например, переместить на второй дисплей, тем самым конфигурируя пользовательский интерфейс согласно требованиям программиста.

4.1. В окне **Basic Activities** (Базовые Блоки) содержатся основные блоки такие как переменная, условие, выбор, массив, комментарии и т. д. (рис. П2.7).

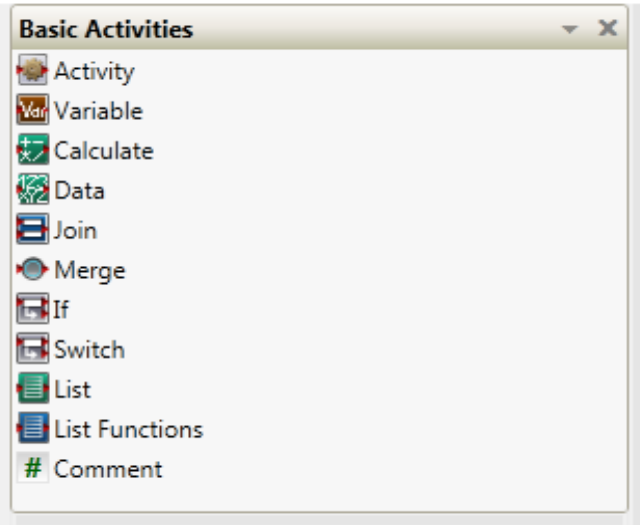


Рисунок П2.7 – Окно базовых блоков

4.2. В окне **Services** (Сервисы) содержатся различные сервисы которые определяют функциональные возможности программы (рис. П2.8).

Возле некоторых сервисов расположен синий значок, нажимая который, открывается ссылка на страницу сайта msdn.microsoft.com с описанием данного сервиса (рис. П2.9).

Так как сервисов большое количество, то предусмотрена **система сортировки**, которая позволяет найти нужные сервисы при сохранении результата.

Для осуществления сортировки сервисов необходимо набрать в поле для ввода символ или комбинацию символов, с которых начинается название сервиса или группа сервисов, например «Generic» (рис. П2.10, слева).

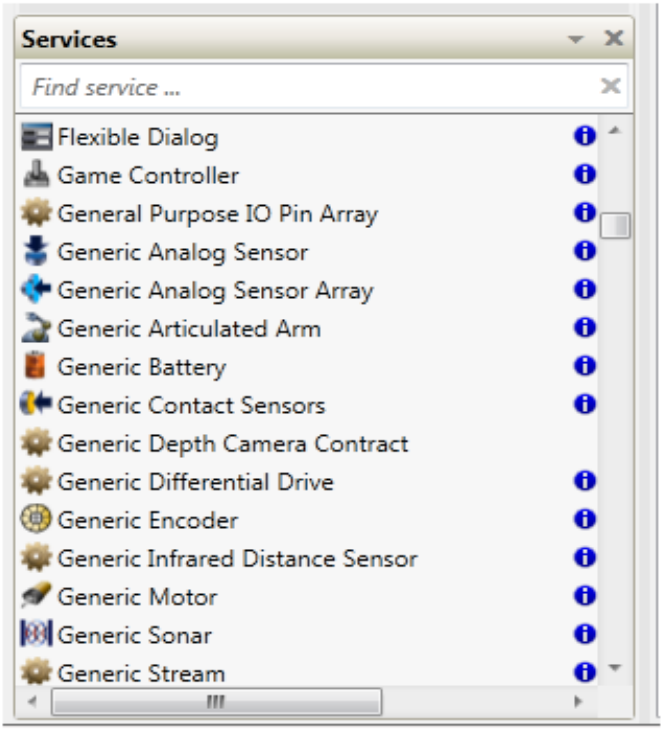


Рисунок П2.8 – Окно сервисов

Рисунок П2.9 – Описание сервиса Flexible Dialog

В данном окне отображены все сервисы, в имени которых присутствует слово Generic. В случае необходимости список можно сохранить для данного проекта (рис. П2.10, справа). Удаление списка производят нажав на знак × возле слова «Generic».

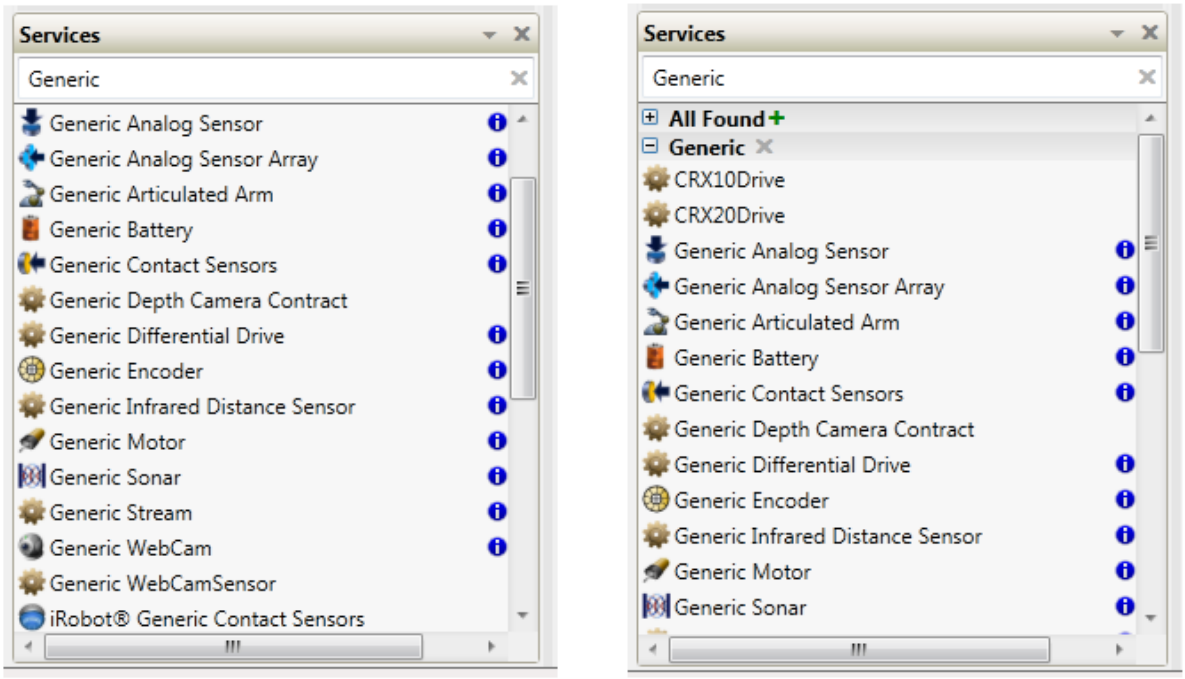


Рисунок П2.10 – Результаты сортировки списка сервисов

4.3. В окне Project (Проект) окне находятся **списки диаграмм** входящих в данный проект (Diagrams), **конфигурации (манифесты) сервисов**, входящих в данный проект (Configurations), а так же компьютеры задействованные в распределенном выполнении данного проекта (Nodes) (рис. П2.11).

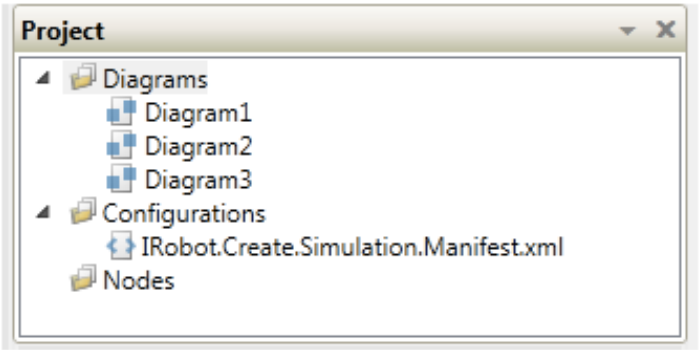


Рисунок П2.11 – Окно проекта

4.4. В окне **Properties** (Свойства) отображаются **свойства выделенного блока или сервиса** (рис. П2.12). Состав этого окна может меняться в зависимости от выделенного элемента. Постоянным атрибутом данного окна является поле ввода комментариев (Comment), а для базовых блоков кнопка сброса размера (Reset Size), для возвращения блоку исходного размера.

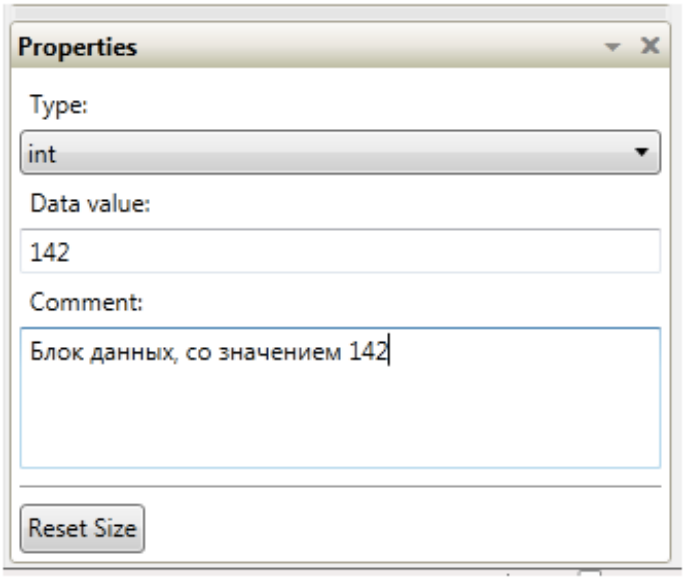


Рисунок П2.12 – Окно свойств

4.5. В окне **Errors** (Ошибки) отображаются ошибки, возникающие при построении диаграммы (рис. П2.13).

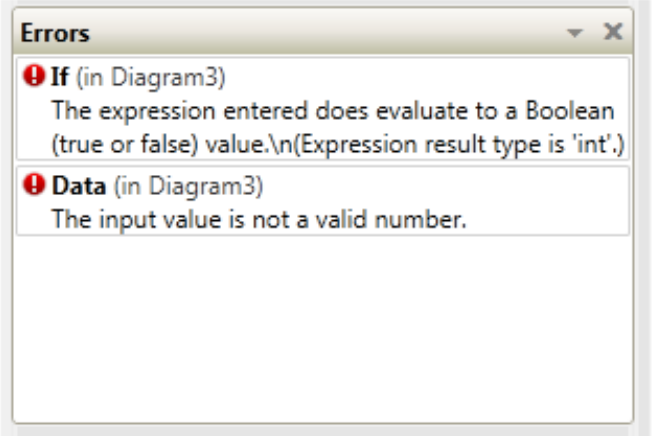


Рисунок П2.13 – Окно ошибок

5. Диаграмма

В **рабочем окне диаграммы** (рис. П2.14) происходит сборка программы, путем перетаскивания с помощью мыши и соединения. Диаграмма имеет имя, которое состоит из латинских букв и цифр без пробелов.

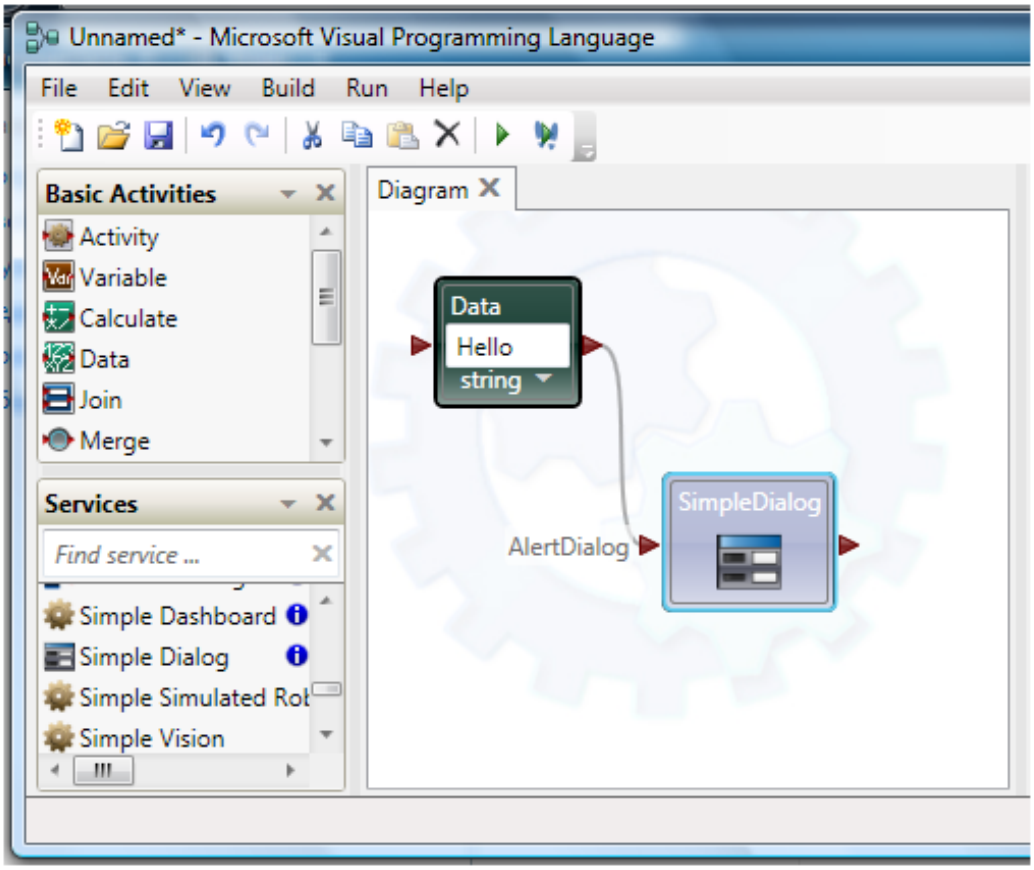


Рисунок П2.14 – Рабочее окно диаграммы

6. Бегунок изменения масштаба

Бегунок изменения масштаба находится в правом нижнем углу окна среды VPL (рис. П2.15), при его перемещении изменяется масштаб диаграммы.

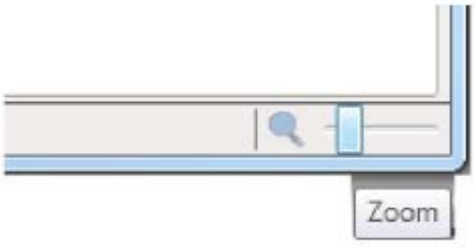


Рисунок П2.15 – Бегунок изменения масштаба

Базовые блоки

Среда Visual Programming Language включает в себя набор **базовых блоков**, которые соответствуют основным командам программирования (константа, переменная, массив, вычисления, условие и др.).

1. Activity 

– позволяет создать новый блок с собственной логикой реализованной пользователем.

2. Variable 

– блок переменной, при нажатии на знак (**...**) появляется окно, которое позволяет задать имя и тип переменной.

3. Calculate 

– блок вычислений позволяет выполнять простейшие арифметические и логические операции, строить выражения с переменными, константами, извлекать данные из сервисов. В таблице П2.1 приведены символы, используемые для обозначения операций в данном блоке.

Таблица П2.1. Значение операций

Операция	Значение
+	сложение
–	вычитание
*	умножение
/	деление
%	остаток после деления
Символы, используемые для обозначения логических операций	
&&	логическое И
||	логическое ИЛИ
!	логическое отрицание

4. Data 

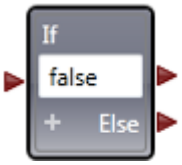
– блок константы, можно выбрать тип данных.

5. Join 

– блок совмещает несколько входящих потоков в один, каждый поток именуется и, далее, используется под своим именем сервисом, который подключен к данному блоку.

6. Merge 

– блок объединяет потоки без их именования, используется для объединения нескольких блоков с одним.

7. If 

– блок проверяет выражение на истину или ложь, позволяет задать несколько условий при помощи нажатия знака (+). В таблице П2.2 приведены символы используемые для обозначения операций в данном блоке.

Таблица П2.2. Значения операций

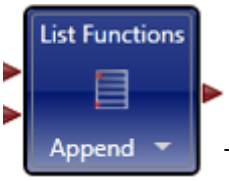
Операция	Значение
= или ==	равно
!= или < >	не равно
<	меньше
>	больше
< =	меньше или равно
> =	больше или равно

8. Switch 

– блок осуществляет выбор того или иного действия в соответствии с совпадением входящих данных с выражениями в полях блока.

9. List 

– блок создает пустой массив данных, тип данных выбирается из выпадающего списка.

10. List Function 

– блок содержит набор операций которые можно выполнять над массивом (таблица П2.3).

Таблица П2.3. Значения операций

Операция	Значение
Append	Добавление элемента
Concatenate	Слияние двух массивов
Reverse	Обращение порядка элементов
Sort	Сортировка списка
RemoveItem	Удаление элемента
InsertItem	Вставка элемента
GetIndex	Получение индекса элемента

11. Comment 

– блок позволяет добавить к диаграмме текстовые сноски (комментарии), блок можно свернуть, нажав треугольник в правом верхнем углу.

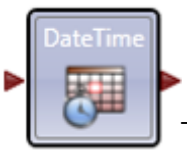
Группы сервисов

Сервис – это программный интерфейс для доступа к таким функциям как взаимодействие с приводами, датчиками, элементами интерфейса пользователя, математическим функциям, таймеру, определению даты времени.

1. Вспомогательные сервисы

1.1. Byte Array 

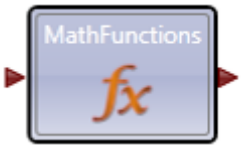
– обеспечивает создание массива байтов.

1.2. Date Time 

– обеспечивает определение даты и времени

1.3. Timer 

– обеспечивает взаимодействие с таймером.

1.4. Math Function 

– обеспечивает вычисление различных математических функций.

1.5. Text Function 

– предоставляет функции для работы с текстовыми строками.

1.6. URL Luncher 

– обеспечивает переход по ссылке.

2. Сервисы интерфейса пользователя

2.1. Sound Player 

– обеспечивает воспроизведение звуковых файлов в формате WAV, так же доступно воспроизведение системных уведомлений Windows.

2.2. Simple Dialog 

– обеспечивает создание диалогового окна Windows трех видов.

2.3. Flexible Dialog 

– обеспечивает создание диалогового окна в котором можно разместить различные элементы ввода/вывода (кнопки, текстовые поля).

2.4. Log 

– обеспечивает вывод служебной информации в окно (Run), отображающем ход выполнения программы.

2.5. Desktop Joystick 

– обеспечивает создание окна с трекболом и цифровой клавиатурой, для управления роботом.

2.6. Direction Dialog 

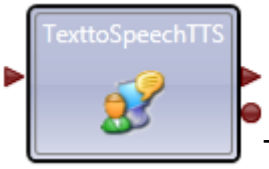
– обеспечивает создание окна с кнопками направления движения.

2.7. Game Controller 

– подключение джойстика.

2.8. Speech Recognizer 

– распознавание речи.

2.9. Text To Speech 

– обеспечивает перевод текста в речь.

2.10. XInput Controller 

– обеспечивает подключение игрового контроллера.

3. Сервисы для взаимодействия с составляющими робота

3.1. Generic Analog Sensor 

– обеспечивает работу с аналоговым датчиком (датчик света).

3.2. Generic Analog Sensor Array 

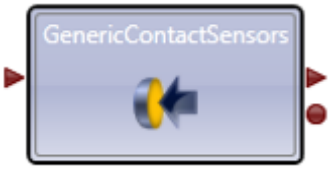
– взаимодействует с группой аналоговых датчиков.

3.3. Generic Articulated Arm 

– взаимодействует с манипулятором.

3.4. Generic Battery 

– обеспечивает взаимодействие с элементом питания.

3.5. Generic Contact Sensor 

– взаимодействует с датчиком касания.

3.6. Generic Differential Drive 

– взаимодействует с дифференциальным приводом.

3.7. Generic Encoder 

– обеспечивает подсчет количества оборотов.

3.8. Generic Motor 

– обеспечивает взаимодействие с мотором.

3.9. Generic Sonar 

– обеспечивает взаимодействие с ультразвуковым датчиком (сонаром).

3.10. Generic WebCam 

– обеспечивает взаимодействие с веб-камерой.

3.11. Wait For Drive Completion 

– обеспечивает опрос привода на предмет завершения движения.

4. Сервисы взаимодействия с аппаратной платформой

Рассмотрим группу сервисов для взаимодействия с аппаратной платформой на примере Lego Mindstorms NXT 2.0.

Для управления микрокомпьютером NXT предусмотрено четыре сервиса: LegoNXTBrick, LegoNXTButtons, LegoNXTBattery, LegoBrickIO.

Для взаимодействия с датчиками используются сервисы LegoNXTTouchSensor, LegoNXTColorSensor, LegoNXTUltrasonicSensor.

4.1. Сервис Lego NXT Brick 

предназначен для следующих целей:

– конфигурирование соединения между компьютером и микроконтроллером,

– подключение и конфигурирование датчиков и портов,

– отправки низкоуровневых команд,

– воспроизведение сигналов через динамик.

Операции для данного сервиса представлены в таблице П2.4.

Таблица П2.4. Операции сервиса Lego NXT Brick

Операции	Значение
Connect to hardware	Настройка подключения к микроконтроллеру NXT
Disconnect from hardware	Отключение от контроллера
Internal adjust device polling frequency	Установка частоты опроса датчиков
Internal detach device	Программное отключение датчика
Internal reserve device port	Резервирование портов
Internal sensor update	Обновление информации о датчиках
Play tone	Воспроизведение звукового сигнала
Send raw command	Отправка низкоуровневых команд

4.2. Сервис Lego NXT Buttons 

обрабатывает события при нажатии кнопок на блоке NXT. Операции доступные для данного сервиса приведены в таблице П2.5.

Таблица П2.5. Операции сервиса Lego NXT Buttons

Операции	Значение
Get	Получение информации о состоянии сервиса
Buttons Update	Обновление информации о состоянии кнопок

4.3. Сервис Lego Brick IO 

взаимодействует с файлами, находящимися в памяти микроконтроллера NXT, позволяет изменять Bluetooth-имя микроконтроллера, запускать программы, находящиеся в памяти NXT. В таблице П2.6 приведены операции доступные для сервиса.

Таблица П2.6. Операции сервиса Lego Brick IO

Операция	Значение
Copy file to brick	Копировать файлы в NXT
Delete file	Удалить файлы
Get	Получение информации о состоянии сервиса
Query brick name	Запрос имени NXT
Query files	Запрос файлов
Query running program	Запрос запущенной программы
Receive Bluetooth message	Прием bluetooth-сообщения
Send Bluetooth message	Отправка bluetooth-сообщения
Set brick name	Установить имя NXT
Start program	Запустить программу
Stop program	Остановить программу

4.4. Сервис Lego NXT Battery 

определяет состояние элемента питания. В таблице П2.7 приведены операции доступные для сервиса.

Таблица П2.7. Операции сервиса Lego NXT Battery

Операция	Значение
Battery update	Обновление состояния батареи
Configure battery	Конфигурирование батареи
Critical level update	Обновление критического уровня заряда
Get	Получение информации о состоянии сервиса
Get (Generic)	Получение информации о состоянии сервиса (используются общие параметры)

ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ СИМУЛЯТОРА СРЕДЫ VSE

Пользовательский интерфейс среды симуляции Visual Simulation Environment RDS состоит из панели меню (рис. П2.16) и основного окна в котором отображается виртуальная среда (рис. П2.2).

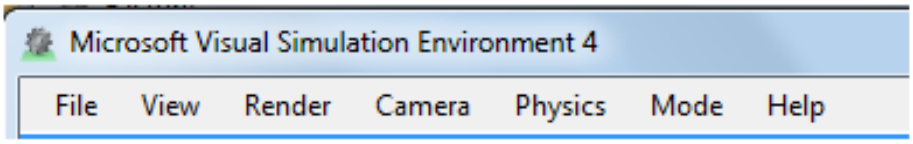


Рисунок 4.19 – Панель меню VSE

Панель меню содержит семь вкладок.

1. File – вкладка для операций с файлами:

– Open Scene – открывает и загружает новую сцену;

– Save Scene As… – вызывает окно «Сохранить сцену как…»;

– Save Material Changes – сохранить изменения в сцене;

– Open Manifest… – открыть манифест (настройки для определенного робота в формате .xml);

– Create Embedded Resources – захватить вложенные ресурсы;

– Capture Image As… – вызывает окно «Захватить образ как…»;

– Exit Simulator – выход из симулятора.

2. View – вкладка для конфигурирования интерфейса:

– Playback Bar – вызывает или закрывает панель воспроизведения и записи (рис. П2.20);

– Status Bar – вызывает (или закрывает) строку статуса. В этой строке отображается отчет времени и позиция камеры (рис. П2.21);

– Profiler – вызывает окно изменения вида интерфейса пользователя (рис. П2.22).

– Look Along – перемещение камеры по осям координат X, Y, Z.

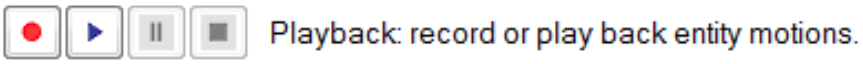


Рисунок П2.20 – Панель воспроизведения и записи

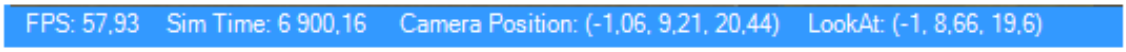


Рисунок П2.21 – Строка статуса

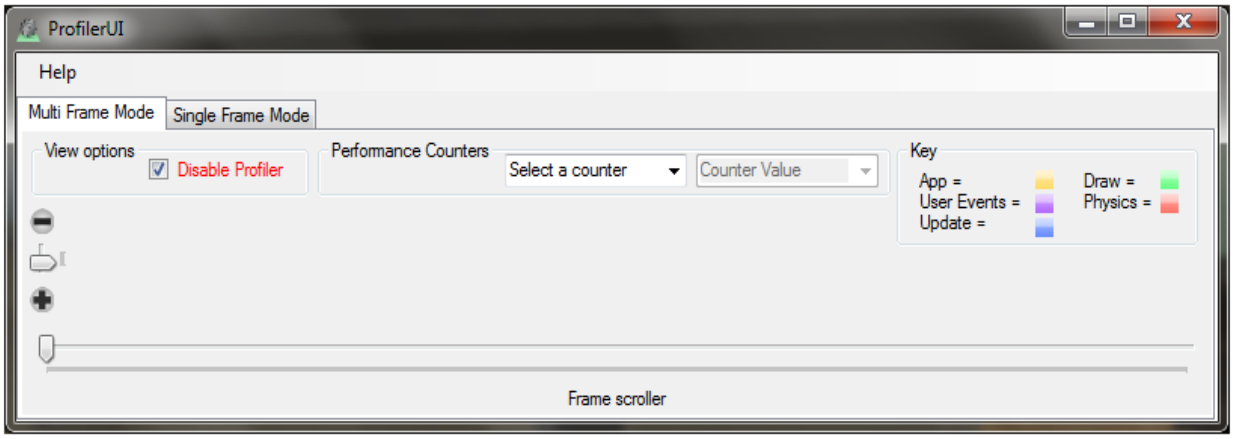


Рисунок П2.22 – Окно Profiler

3. Render – режим изображения:

– Visual – визуализация;

– Wireframe – посмотреть каркас;

– Physics – показать физические свойства;

– Combined – одновременное отображение среды и её каркаса;

– No Rendering – без визуализации, при нажатии, вместо среды, отображается черный экран;

– Graphic Settings – настройки графики;

– Physics View Settings – настройки представления физики.

4. Camera – вкладка камеры:

– Main Camera – главная камера; особенности управления камерой приведены в таблице П2.8.

5. Physics – вкладка свойств физического движка:

– Enabled – активация физики в симуляторе;

– Settings… – настройки свойств физики симулятора.

Таблица П2.8. Клавиши перемещения камеры в среде VSE

Клавиша	Значение
W или Стрелка вверх	Вперед
S или Стрелка вниз	Назад
A или Стрелка влево	Влево
D или Стрелка вправо	Вправо
Q	Вверх
E	Вниз

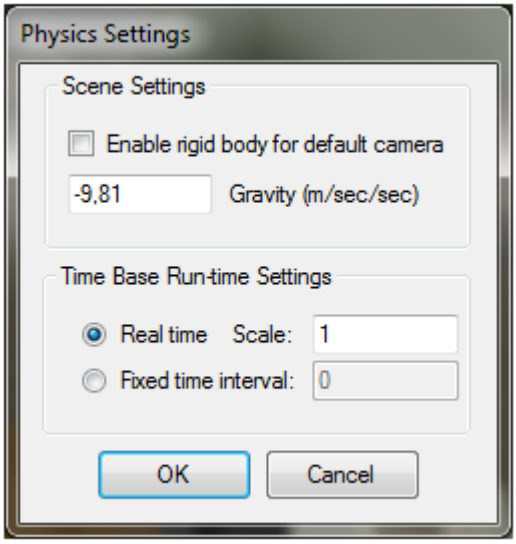


Рисунок П2.23 – Окно свойств физического движка

6. Mode – вкладка выбора режима работы симулятора:

– Run – запуск симуляции;

– Edit – режим редактирования.

Режим редактирования позволяет отображать все объекты сцены и их названия. При выборе объекта и нажатии кнопки Edit Entity отображается панель операций с объектом (рис. П2.24) и окно редактирования (рис П2.25). В панели операций можно задавать положение выбранного объекта, изменяя координаты X, Y, Z, а также вращать объект. В окне редактирования можно задать до 39 различных свойств объекта, таких как: положение, размер, цвет частей и другие свойства. Предоставлена возможность сортировки этих свойств, как по категориям, так и в алфавитном порядке.

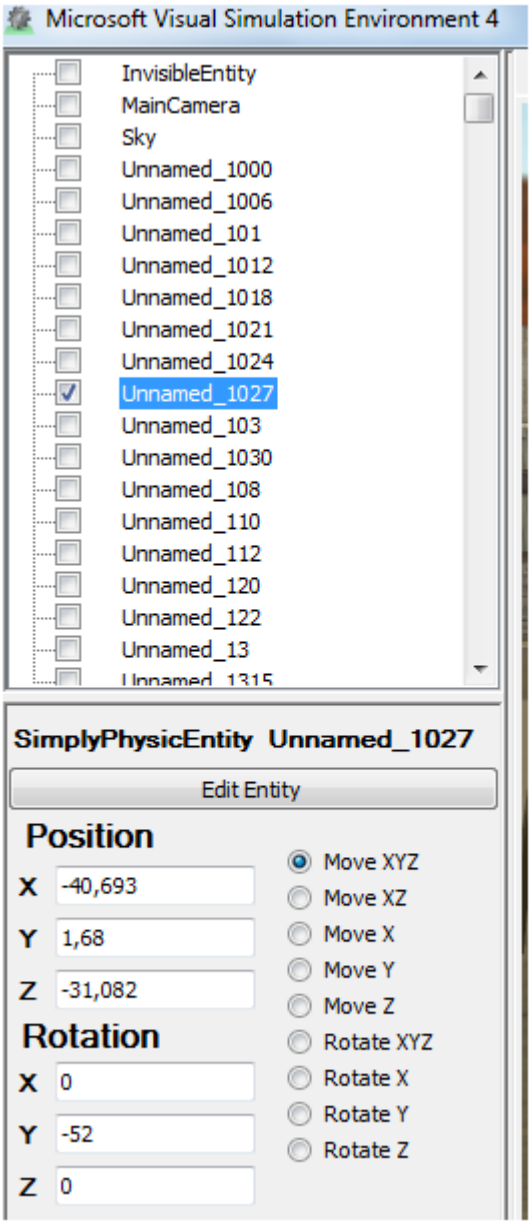


Рисунок П2.24 – Панель операций с объектом

Для того что бы переместить объект в заданную точку, можно использовать два способа:

1. Навести курсор на выбранный объект, и одновременно нажав Ctrl и левую клавишу мыши переместить в необходимое место.

2. Выбрать имя объекта в Панели операций с объектом и задать положение геометрического центра примитива, представляющего данный объект.

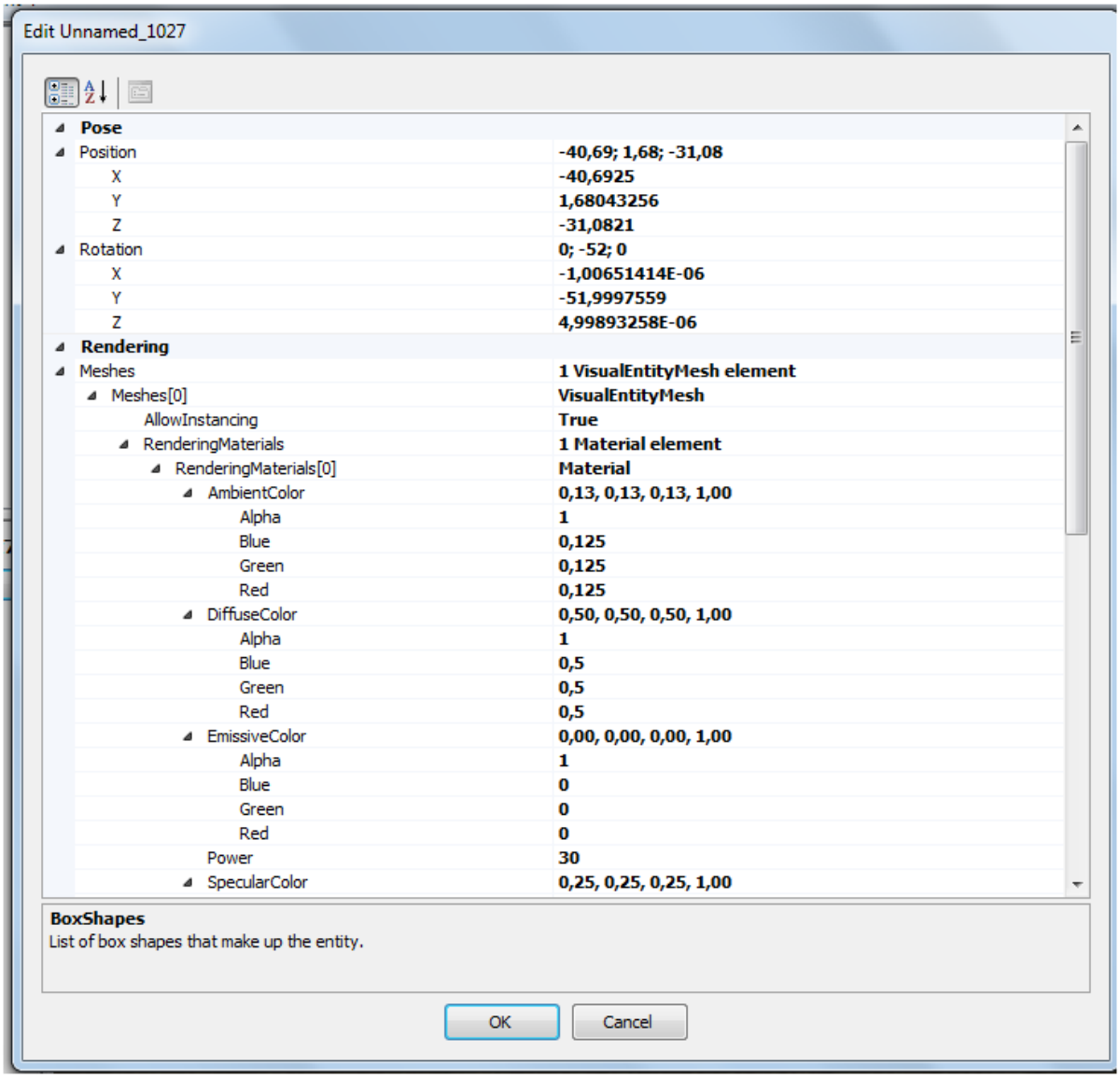


Рисунок П2.25 – Окно редактирования

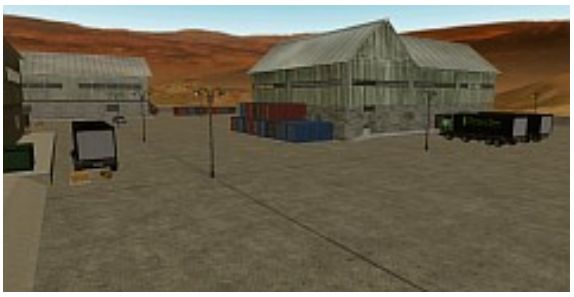
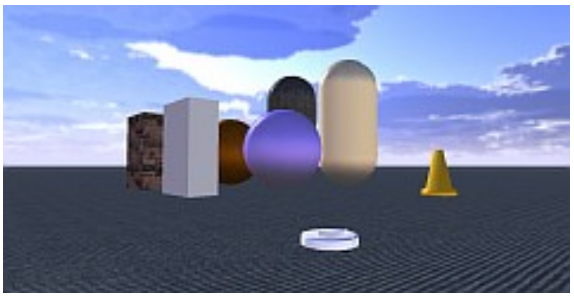
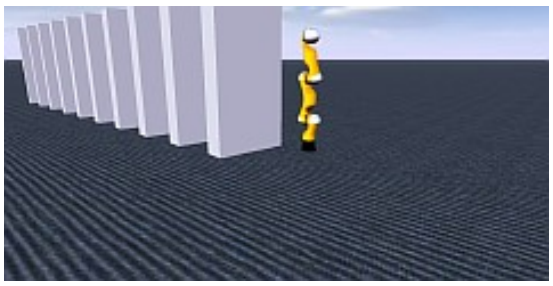
7. Help – помощь:

– Help Contents – вызывает окно справки;

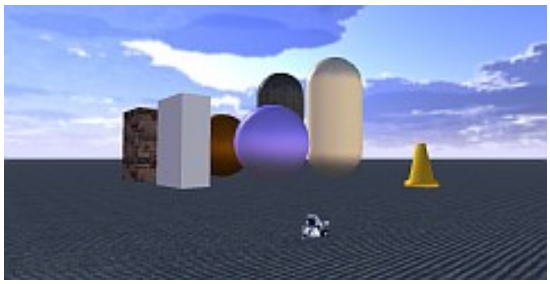
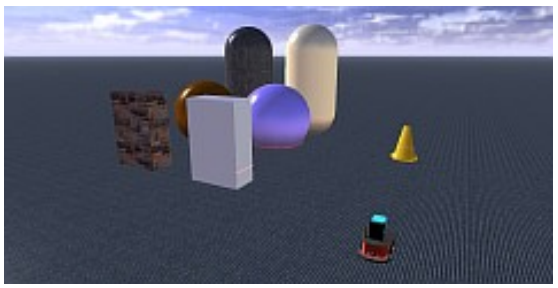
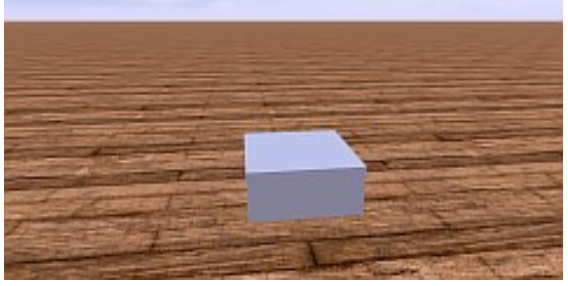
– About Visual Simulation Environment – вызывает окно, в котором содержится информация о программе (версия, фирма – производитель и т. д.).

В таблице П2.8 приведен перечень готовых сцен присутствующих в среде симуляции Visual Simulation Environment 4.

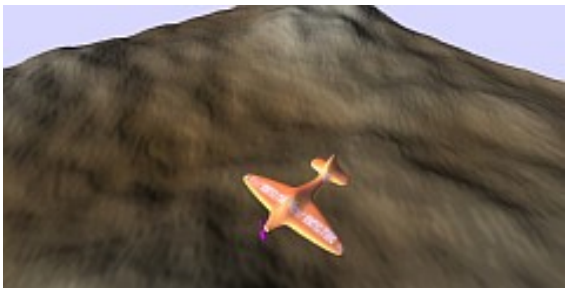
Таблица П2.8. Список готовых сцен среды симуляции

Название сцены	Описание	Изображение
Apartment Environment	Квартира	
Entities	Объекты	
Factory	Завод	
House Floor plan	План дома	
iRobot Create Simulation	Симуляция iRobot	
KUKA LBR3 Arm	Симуляция манипулятора KUKA LBR3	

Продолжение таблицы П2.8

Название сцены	Описание	Изображение
Lego NXT Tribot Simulation	Симуляция Lego NXT Tribot	
Modern House	Современный дом	
Multiple Simulated Robots	Имитация нескольких роботов	
Outdoor Environment	Естественная обстановка	
Pioneer 3DX Simulation	Моделирование Pioneer 3DX	
Simple Simulated Robot	Простой моделируемый робот	

Продолжение таблицы П2.8

Название сцены	Описание	Изображение
Simulated Air Resistance	Моделирование сопротивления воздуха	
Simulated Sumo	Моделирование соревнования Sumo	
Urban Environment	Городская обстановка	

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по организации и проведению самостоятельной работы
по дисциплине «Системы реального времени»
для студентов направления подготовки
09.03.01 Информатика и вычислительная техника
Направленность (профиль)
Программное обеспечение средств высилительной техники и автоматизированных систем

Ставрополь 2026

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Основная задача высшего образования заключается в формировании творческой личности специалиста, способного к саморазвитию, самообразованию, инновационной деятельности. Решение этой задачи вряд ли возможно только путем передачи знаний в готовом виде от преподавателя к студенту. Для решения этой задачи в учебные планы всех специальностей включена самостоятельная работа, составляющая значительную часть учебного времени обучаемого.

В связи с этим, обучаемому из пассивного потребителя знаний необходимо превратиться в активного их творца, умеющего сформулировать проблему, проанализировать пути ее решения, найти оптимальный результат и доказать его правильность. Происходящая в настоящее время реформа высшего образования связана по своей сути с переходом парадигмы обучения к парадигме образования. В этом плане следует признать, что самостоятельная работа студентов является не просто важной формой образовательного процесса, а должна стать его основой. Это предполагает ориентацию на активные методы овладения знаниями, развитие творческих способностей студентов, переход от поточного к индивидуализированному обучению с учетом потребностей и возможностей личности.

В широком смысле под самостоятельной работой следует понимать совокупность всей самостоятельной деятельности студентов как в учебной аудитории, так и вне ее, в контакте с преподавателем и в его отсутствие.

Самостоятельная работа *в контакте с преподавателем* реализуется:

– непосредственно в процессе аудиторных занятий – на лекциях, практических и семинарских занятиях, при выполнении лабораторных работ;

– вне рамок расписания – на консультациях по учебным вопросам, в ходе творческих контактов, при ликвидации задолженностей, при выполнении студентом учебных и творческих задач.

Видами заданий для *внеаудиторной* самостоятельной работы могут быть:

1. **Для овладения знаниями:**

– чтение текста (учебника, первоисточника, дополнительной литературы);

– составление плана текста;

– графическое изображение структуры текста;

– конспектирование текста;

– работа со словарями и справочниками;

– работа с нормативными документами;

– учебно-исследовательская работа;

– использование аудио- и видеозаписей, компьютерной техники, Интернет и др.

2. **Для закрепления и систематизации знаний:**

– работа с конспектом лекции (обработка текста);

– повторная работа над учебным материалом (учебника, первоисточника, дополнительной литературы, аудио- и видеозаписей);

– составление плана и тезисов ответа;

– составление таблиц для систематизации учебного материала;

– изучение нормативных материалов;

– ответы на контрольные вопросы;

– аналитическая обработка текста (аннотирование, рецензирование, реферирование, конспект, анализ и др.);

– подготовка сообщений к выступлению на семинаре, конференции;

– подготовка рефератов, докладов;

– составление библиографии;

– тестирование и др.

3. **Для формирования умений:**

– решение задач и упражнений по образцу;

– решение вариантных задач и упражнений;

– решение ситуационных производственных (профессиональных) задач;

– подготовка к деловым играм;

– проектирование и моделирование разных видов и компонентов профессиональной деятельности;

– подготовка курсовых и дипломных работ (проектов);

– экспериментальная работа;

– рефлексивный анализ профессиональных умений, с использованием аудио- и видеотехники и др.

Границы между этими видами работ достаточно размыты, а сами виды самостоятельной работы пересекаются.

1 ОБЩАЯ ХАРАКТЕРИСТИКА САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТА ПРИ ИЗУЧЕНИИ ДИСЦИПЛИНЫ «СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ»

Целью самостоятельной работы студента при изучении дисциплины «Системы реального времени» является формирование у обучающихся личностных качеств, а также набора профессиональных компетенций будущего бакалавра в области применения систем, функционирующих на основе принципов реального времени и в соответствии с требованиями ФГОС ВО по направлению подготовки 09.03.01 – Информатика и вычислительная техника.

Задачи самостоятельной работы:

– изучение основных способов построения и принципов функционирования систем реального времени;

– овладение перспективными методами анализа систем реального времени;

– формирование практических навыков проектирования систем реального времени.

В рабочей программе дисциплины «Системы реального времени» определены следующие виды самостоятельной работы:

– подготовка к лабораторной работе;

– самостоятельное изучение литературы.

Целью вида самостоятельной работы «*Подготовка к лабораторной работе*» по дисциплине «Системы реального времени» является закрепление студентами теоретического материала по специальности и выработка навыков самостоятельной профессиональной и научно-исследовательской деятельности в области программного обеспечения робототехнических и интеллектуальных систем.

Задачи вида самостоятельной работы «*Подготовка к лабораторной работе*» по дисциплине «Системы реального времени» обусловлены

необходимостью получения студентом знаний, умений, навыков согласно требованиям ФГОС ВО по направлению подготовки 09.03.01 – Информатика и вычислительная техника, на основе которых формируются соответствующие результаты обучения по дисциплине (модулю), соотнесённые с планируемыми результатами освоения образовательной программы.

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ПК-7. Способен осуществлять обслуживание сетевых устройств и серверных операционных систем информационно-коммуникационной системы	ИД-1$_{ПК-7}$ понимает общие принципы функционирования аппаратных, программных и программно-аппаратных средств администрируемой сети; ориентируется в архитектурах аппаратных, программных и программно-аппаратных средств администрируемой сети; понимает принципы установки и настройки программного обеспечения.	На основе понимания общих принципов функционирования аппаратных, программных и программно-аппаратных средств системы реального времени, может определить архитектуру системы реального времени и провести установку и настройку её программного обеспечения.
	ИД-2$_{ПК-7}$ организовывает инвентаризацию периферийных и абонентских технических средств; идентифицирует права пользователей по доступу к программно-аппаратным средствам информационных служб инфокоммуникационной системы и/или ее составляющих; применяет специальные программно-аппаратные средства контроля доступа пользователей к программно-аппаратным средствам информационных служб инфокоммуникационной системы.	Способен организовать инвентаризацию периферийных средств системы реального времени.
	ИД-3$_{ПК-7}$ осуществляет конфигурирование УАТС, периферийных и абонентских устройств.	Способен осуществить конфигурирование периферийных устройств системы реального времени.

Целью вида самостоятельной работы «*Самостоятельное изучение литературы*» по дисциплине «Системы реального времени» является расширение и закрепление знаний, приобретаемых студентом на

традиционных формах занятий и выработка навыков самостоятельной работы с литературой и источниками в области программного обеспечения робототехнических и интеллектуальных систем.

Задачи вида самостоятельной работы «*Самостоятельное изучение литературы*» по дисциплине «Системы реального времени» обусловлены необходимостью получения студентом знаний, умений, навыков согласно требованиям ФГОС ВО по направлению подготовки 09.03.01 – Информатика и вычислительная техника, на основе которых формируются соответствующие результаты обучения по дисциплине (модулю), соотнесённые с планируемыми результатами освоения образовательной программы.

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ПК-7. Способен осуществлять обслуживание сетевых устройств и серверных операционных систем информационно-коммуникационной системы	ИД-1$_{ПК-7}$ понимает общие принципы функционирования аппаратных, программных и программно-аппаратных средств администрируемой сети; ориентируется в архитектурах аппаратных, программных и программно-аппаратных средств администрируемой сети; понимает принципы установки и настройки программного обеспечения.	На основе понимания общих принципов функционирования аппаратных, программных и программно-аппаратных средств системы реального времени, может определить архитектуру системы реального времени и провести установку и настройку её программного обеспечения.
	ИД-2$_{ПК-7}$ организовывает инвентаризацию периферийных и абонентских технических средств; идентифицирует права пользователей по доступу к программно-аппаратным средствам информационных служб инфокоммуникационной системы и/или ее составляющих; применяет специальные программно-аппаратные средства контроля доступа пользователей к программно-аппаратным средствам информационных служб инфокоммуникационной системы.	Способен организовать инвентаризацию периферийных средств системы реального времени.

	ИД-З$_{ПК-7}$ осуществляет конфигурирование УАТС, периферийных и абонентских устройств.	Способен осуществить конфигурирование периферийных устройств системы реального времени.

2 ПЛАН-ГРАФИК ВЫПОЛНЕНИЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

На первом этапе необходимо ознакомиться с рабочей программой дисциплины, в которой рассмотрено содержание тем дисциплины лекционного курса, взаимосвязь тем лекций с лабораторными занятиями, темы и виды самостоятельной работы. По каждому виду самостоятельной работы предусмотрены определённые формы отчетности.

Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием количества часов и видов занятий

№	Раздел (тема) дисциплины и краткое содержание	Формируемые компетенции, индикаторы	очная форма				Формы текущего контроля успеваемости
			Контактная работа обучающихся с преподавателем /из них в форме практической подготовки, часов			Самостоятельная работа, часов	
			Лекции	Практические занятия	Лабораторные работы		
1	**Тема 1. Введение в системы реального времени.** Определение систем реального времени. Назначение систем реального времени. Классификация систем реального времени. Международные и отечественные стандарты систем реального времени.	ПК-7	2	-	-	2	собеседование

2	**Тема 2. Особенности систем реального времени.** Основные понятия систем реального времени. Состояния и типы взаимодействия процессов в операционной системе реального времени. Основные требования к операционной системе реального времени. Особенности архитектуры операционных систем реального времени. Виды ресурсов в системах реального времени. Структурное подобие систем реального времени и аппаратуры.	ПК-7	2	-	-	2	собеседование
3	**Тема 3. Аппаратная среда систем реального времени.** Компоненты интерфейса между техническим процессом и системой управления. Классификация и основные характеристики датчиков. Классификация и основные характеристики исполнительных механизмов.	ПК-7	2	-	-	2	собеседование
4	**Тема 4. Устройство связи с объектом.** Ввод аналоговых сигналов в компьютер. Преобразование цифровых и аналоговых сигналов. Проблемы согласования и передачи сигналов.	ПК-7	2	-	-	2	собеседование
5	**Тема 5. Аппаратная реализация управляющих компьютеров.** Основные требования к аппаратной организации управляющих компьютеров. Программируемые логические контроллеры. Конструктив «Евромеханика». Открытые стандарты системных магистралей.	ПК-7	2	-	24.00/ 24.00	2	собеседование, защита лабораторной работы
6	**Тема 6. Особенности многопоточной организации операционных систем реального времени.** Микроядро QNX Neutrino. Атрибуты и состояния потоков. Диспетчеризация потоков. Управление потоками.	ПК-7	2	-	-	2	собеседование

7	**Тема 7. Механизмы синхронизации потоков.** Мьютексы. Условные переменные. Ждущие блокировки. Барьеры. Семафоры. Блокировки чтения/записи. Синхронизация потоков. Тайм-ауты ядра.	ПК-7	4	-	-	4	собеседование
8	**Тема 8. Механизмы взаимодействия программных потоков на уровне ядра операционных систем реального времени.** Синхронный обмен сообщениями. Векторные сообщения. Многошаговый обмен сообщениями. Локализация сервера синхронного канала. Импульсы. Сигналы. Иерархический принцип обмена синхронными сообщениями. Асинхронный обмен сообщениями.	ПК-7	4	-	-	4	собеседование
9	**Тема 9. Механизмы взаимодействия программных потоков реализуемых вне ядра операционных систем реального времени.** Очереди сообщений. Разделяемая память. Неименованные и именованные каналы.	ПК-7	2	-	-	2	собеседование
10	**Тема 10. Жизненный цикл систем реального времени.** Жизненный цикл и ГОСТ Р ИСО/МЭК 12207. Модель разработки систем реального времени. Основы тестирования систем реального времени.	ПК-7	2	-	-	2	собеседование
11	**Тема 11. Анализ и верификация систем реального времени.** Назначение частотно-монотонного анализа. Тест предела использования процессора. Тест времени завершения. Расширения частотно-монотонного анализа.	ПК-7	2	-	-	2	собеседование

12	**Тема 12. Технологии разработки систем реального времени.** UML и Real-Time UML. Платформа QNX. Платформа Real-Time Java. Технология промежуточного программного обеспечения.	ПК-7	2	-	-	2	собеседование
13	**Тема 13. Инструментальные средства проектирования систем реального времени.** Инструментальная платформа Eclipse. IBM Rational Rhapsody. Инструментальная платформа LabVIEW Real-Time. Средство верификации Cantata++.	ПК-7	2	-	12.00/ 12.00	2	собеседование, защита лабораторной работы
14	**Тема 14. Обзор наиболее распространенных операционных систем реального времени.** QNX Neutrino. VxWorks. RTOS-32. CsLEOS. RTEMS. INTEGRITY. LynxOS. OS-9. Windows Embedded Compact. Symbian OS. Nucleus. Расширения реального времени для Windows NT. Расширения реального времени для Linux. Системы реального времени на базе DOS.	ПК-7	6	-	-	6	собеседование
	ИТОГО		36.00/ 0	-	36.00/ 36.00	36.00	

3 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА

При работе с рекомендованной литературой и источниками студенту необходимо, в первую очередь, обратить внимание на следующие моменты:

1. Внимательно изучить материалы, характеризующие курс и тематику самостоятельного изучения, что изложено в учебно-методическом комплексе по дисциплине. Это позволит четко представить как круг, изучаемых тем, так и глубину их постижения.

2. Составить подборку литературы, достаточную для изучения предлагаемых тем. В учебно-методическом комплексе или методических рекомендациях обязательно представлены основной и дополнительный списки литературы, доступные для студентов. Они носят рекомендательный характер, это означает, что всегда есть литература, которая может не входить в данный список, но является необходимой для освоения темы. При этом следует иметь в виду, что нужна литература различных видов:

– учебники, учебные и учебно-методические пособия;

– монографии, сборники научных статей, публикации в журналах;

– справочная литература – энциклопедии, словари, тематические, терминологические справочники;

3. Обучаемый должен совершать собственные интеллектуальные усилия, а не только механически заучивать понятия и положения.

Вопросы для собеседования по вопросам темы выносимой на самостоятельное изучение

Тема 1. Введение в системы реального времени.

1. Прокомментируйте понятие «реальное время».
2. Назовите основные особенности системы реального времени.
3. Перечислите составные части структуры СРВ.
4. Охарактеризуйте основные области применения СРВ.

5. Какие подходы можно применить для классификации систем реального времени.

6. Назовите отличия между системами жесткого и мягкого реального времени.

Тема 2. Особенности систем реального времени.

1. Дайте определения процесса и потока.
2. Перечислите виды межпроцессного взаимодействия.
3. Какие категории потоков можно определить для потоков?
4. Назовите состояния процесса в ОС РВ и возможные переходы из одного состояния в другое.
5. Перечислите типы взаимодействия процессов.
6. Какие отличительные черты имеет операционная система реального времени по сравнению с обычной ОС?
7. Прокомментируйте особенности программирования для СРВ.

Тема 3. Аппаратная среда систем реального времени.

1. Перечислите компоненты интерфейса между техническим процессом и системой управления.
2. Назовите составные элементы датчика.
3. Назовите составные элементы исполнительного механизма.
4. Назовите факторы, определяющие способ передачи сигналов между компьютером и физическим процессом.
5. Перечислите подходы к классификации датчиков.
6. Назовите основные характеристики датчиков.
7. Перечислите подходы к классификации исполнительных механизмов.
8. Назовите основные характеристики исполнительных механизмов.

Тема 4. Устройство связи с объектом.

1. Прокомментируйте назначение и основные функции микроядра ОС РВ QNX Neutrino.
2. Назовите атрибуты потока.
3. В каких состояниях может находиться поток?
4. Перечислите возможные условия блокировки потоков.

Тема 5. Аппаратная реализация управляющих компьютеров.

1. Прокомментируйте назначение и основные функции микроядра ОС РВ QNX Neutrino.
2. Назовите атрибуты потока.
3. В каких состояниях может находиться поток?
4. Перечислите возможные условия блокировки потоков.

Тема 6. Особенности многопоточной организации операционных систем реального времени.

1. Прокомментируйте назначение и основные функции микроядра ОС РВ QNX Neutrino.
2. Назовите атрибуты потока.
3. В каких состояниях может находиться поток?
4. Перечислите возможные условия блокировки потоков.
5. Дайте определение диспетчеризации потоков.
6. Какие дисциплины диспетчеризации поддерживаются в операционной системе QNX Neutrino?

Тема 7. Механизмы синхронизации потоков.

1. Какие механизмы синхронизации используются в ОС РВ QNX Neutrino?
2. Прокомментируйте понятие мьютекс.
3. Назовите атрибуты мьютекса.

4. Какими системными вызовами поддерживается объект CONDVAR?

5. Какими системными вызовами поддерживаются ждущие группировки?

6. Дайте определение механизма синхронизации «барьер».

7. Какие функции выполняет барьер?

Тема 8. Механизмы взаимодействия прораммных потоков на уровне ядра операционных систем реального времени.

1. Назовите формы взаимодействия процессов в ОС РВ QNX Neutrino.

2. Прокомментируйте особенности реализации синхронного обмена сообщениями.

3. Прокомментируйте особенности реализации обмена векторными сообщениями.

4. Какие дополнительные функции обмена сообщениями реализованы в ОС РВ QNX Neutrino?

5. Дайте характеристику импульсного обмена.

6. Перечислите варианты поведения процесса при получении сигнала.

7. Какие сервисные функции применяются для формирования списка обрабатываемых сигналов?

Тема 9. Механизмы взаимодействия прораммных потоков реализуемые вне ядра операционных систем реального времени.

1. Назовите механизмы взаимодействия программны потоков реализуемые вне ядра ОС РВ QNX Neutrino.

2. Прокомментируйте особенности реализации очередей сообщений.

3. Какие вызовы используются для управления очередью сообщений?

4. Перечислите преимущества использования разделяемой памяти при организации межзадачного взаимодействия.

5. Какие возможности межзадачного взаимодействия появляются при использовании сочетания разделяемой памяти с механизмом обмена сообщениями?

Тема 10. Жизненный цикл систем реального времени.

1. Назовите процессы жизненного цикла программных средств, которые можно отнести к основным.
2. Назовите вспомогательные процессы жизненного цикла программных средств.
3. Назовите организационные процессы жизненного цикла программных средств.
4. Охарактеризуйте стадии разработки компьютерной системы.
5. Дайте определения верификации и валидации.
6. Что такое отладка программного обеспечения?
7. Назовите документы, специфицирующие ПО реального времени применительно к V-модели создания СРВ.

Тема 11. Анализ и верификация систем реального времени.

1. Назовите основные методы формальной верификации систем реального времени.
2. Поясните особенности частотно-монотонного анализа СРВ.
3. Проведите анализ СРВ с использованием теста предела использования процессора.
4. Проведите анализ СРВ с использованием теста времени завершения.

Тема 12. Технологии разработки систем реального времени.

1. Приведите примеры аннотирования моделей в Real-Time UML.
2. Какие инструментальные средства входят в состав QNX Momentics?
3. Какие расширения платформы Java предлагает спецификация RTSJ?

4. Перечислите преимущества использования службы времени в RTSJ.

5. Перечислите преимущества использования функций управления памятью в RTSJ.

6. Назовите количество уровней приоритета в RTSJ.

7. Перечислите преимущества использования службы времени в RTSJ.

8. Перечислите преимущества использования потоков реального времени в RTSJ.

Тема 13. Инструментальные средства проектирования систем реального времени.

1. Какие элементы входят в состав Eclipse SDK?

2. Перечислите графические плагины Eclipse C/C++ Development Tools.

3. Назовите различные комплекты IBM Rational Rhapsody.

4. Какие возможности получает разработчик СРВ при использовании функции трассировки требований в инструменте Rhapsody Requirements Gateway?

5. Какие действия предполагаются при выполнении «анимации моделей» в IBM Rational Rhapsody Developer?

6. Из каких частей состоит программа LabVIEW?

7. Назовите возможные варианты реализации функционирования в реальном времени при использовании LabVIEW Real-Time Module.

8. Какие средства разработки СРВ присутствуют в LabVIEW Real-Time Module?

9. Прокомментируйте базовую архитектуру LabVIEW Real-Time.

Тема 14. Обзор наиболее распространенных операционных систем реального времени.

1. Проведите сравнение рассмотренных ранее ОС РВ по различным параметрам. Сформулируйте рекомендации по их применению.

2. Какую архитектуру имеет операционная система VxWorks?
3. Назовите особенности операционной системы RTOS-32.
4. Какой набор средств разработки имеет ОС РВ CsLEOS?
5. Назовите виды межпроцессного взаимодействия в RTEMS.
6. Какие особенности присущи INTEGRITY-178B?
7. Перечислите основные свойства операционной системы LynxOS.

Тема 15. Операционные системы реального времени, разработанные для портативных устройств. Расширения реального времени.

1. Какую архитектуру имеет операционная система Windows Embedded CE?
2. Назовите основные компоненты Windows Embedded Compact 7.
3. Перечислите особенности Symbian OS.
4. Какую архитектуру имеет операционная система Nucleus OS?
5. Какие возможности нужно предусмотреть в ОС Windows NT для опеспечения её использования во встраиваемых системах и СРВ?
6. Какие функции добавляют расширения реального времени в Windows NT?
7. Прокомментируйте особенности реализации RTX.
8. Прокомментируйте особенности реализации INtime.
9. Прокомментируйте особенности реализации CeWin.
10. Какую архитектуру имеет KUKA Virtual Machine Framework?
11. Прокомментируйте особенности реализации RTLinux.
12. Прокомментируйте особенности реализации RTAI.

Критерии оценивания компетенций

Оценка «отлично» выставляется студенту, если он выполняет в полном объеме и безошибочно задания оценочных средств по дисциплине, относящиеся к данной компетенции (индикаторам достижения компетенции), при этом демонстрирует на высоком уровне способность решать стандартные

и нестандартные прикладные практические задачи проектирования систем реального времени.

Оценка «хорошо» выставляется студенту, если он выполняет, допуская незначительные ошибки, задания оценочных средств по дисциплине, относящиеся к данной компетенции (индикаторам достижения компетенции), при этом демонстрирует на среднем уровне способность решать только стандартные прикладные практические задачи проектирования систем реального времени.

Оценка «удовлетворительно» выставляется студенту, если он выполняет, допуская ошибки, исправление которых осуществляет с помощью наводящих вопросов преподавателя, задания оценочных средств по дисциплине, относящиеся к данной компетенции (индикаторам достижения компетенции), при этом демонстрирует на минимальном уровне способность решать стандартные прикладные практические задачи проектирования систем реального времени.

Оценка «неудовлетворительно» выставляется студенту, если он, выполняя задания оценочных средств по дисциплине, не достигает минимального уровня сформированности компетенции (индикаторам достижения компетенции), при этом не демонстрирует способность решать стандартные прикладные практические задачи проектирования систем реального времени, допускает грубые ошибки, которые не может исправить даже с помощью наводящих вопросов преподавателя.

4 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПОДГОТОВКЕ К ЛАБОРАТОРНЫМ РАБОТАМ

При проведении лабораторных работ по дисциплине «Системы реального времени» студент должен знать и владеть навыками, содержание, которых представлено в программе по дисциплине.

5.1. Методические указания при подготовке к лабораторной работе

Подготовку к лабораторной работе рекомендуется проводить в следующей последовательности:

– уяснить тему и цель, предстоящего занятия;

– изучить теоретический материал в соответствии с темой занятия (рекомендуется использовать рекомендованную литературу, конспект лекций, учебное пособие (практикум).

5.2. Методические указания при выполнении задания лабораторной работы

При выполнении задания лабораторной работы студенты должны строго соблюдать, установленные правила охраны труда.

При выполнении задания студентам рекомендуется:

– уяснить цель, выполняемых заданий и способы их решения;

– задания, выполнять в той последовательности, в которой они указаны в практикуме;

– при выполнении задания и изучении теоретического материала использовать помощь преподавателя;

– оформить отчет по лабораторной работе;

– ответить на контрольные вопросы.

5.3. Методические указания при подготовке к защите лабораторной работы

При подготовке к защите лабораторной работы студентам рекомендуется:

– подготовить отчет по лабораторной работе;

– уметь обосновать, сделанные выводы;

– закрепить знания теоретического материала по теме занятия (рекомендуется использовать контрольные вопросы);

– знать порядок проведения расчетов (проводимых исследований).

Защита лабораторной работы осуществляется путем собеседования студента с преподавателем. При собеседовании студент представляет на проверку отчет по лабораторной работе.

Собеседование со студентом, претендующим на оценку «отлично» проводится по вопросам повышенного уровня обученности.

Собеседование со студентом, не претендующим на оценку «отлично» проводится по вопросам базового уровня, которые приведены в фонде оценочных средств по дисциплине.

Вопросы (задания) для проверки уровня обученности

Лабораторная работа 1. Основы программирования мультипоточных приложений в среде Visual Programming Language.

1. Перечислите типы хранимого значения в блоке данных.
2. Перечислите типы переменных в блоке переменной.
3. Назовите арифметические и логические операции, реализуемые в блоке вычислений.
4. Охарактеризуйте опции работы сервиса симуляции консольного интерфейса.
5. Охарактеризуйте понятия процесса и потока.
6. Перечислите возможные пути создания ячейки потока в блоке слияния.

Лабораторная работа 2. Изучение иерархической архитектуры сервисов.

1. Дайте определение модуля.
2. Перечислите факторы способствующие созданию модульных программ.
3. Дайте определение иерархии.
4. Назовите свойства присущие архитектуре программного обеспечения как иерархической системе.
5. Перечислите шаги позволяющие организовать декомпозицию программы с помощью нотификации.

Лабораторная работа 3. Изучение модульной организации программ.

1. Назовите цели структуризации программных продуктов на этапе разработки.
2. Прокомментируйте особенности многофункциональной обработки сообщений в модуле.
3. Перечислите преимущества использования нотификаций.
4. Прокомментируйте содержание вкладок окна «Actions and Notifications» модуля.

Лабораторная работа 4. Моделирование автоматизированного управления мобильным роботом.

1. Охарактеризуйте особенности системы командного управления.
2. Назовите принципиальные отличия организации автоматизированного управления с помощью виртуального задающего манипулятора от варианта использования фиксированного набора команд.
3. Прокомментируйте возможности использования координат трекбола в сервисе Desktop Joystick.

4. Перечислите и прокомментируйте варианты организации автоматизированного управления с помощью фиксированного набора команд применяемые в данной работе.

Лабораторная работа 5. Моделирование автоматического управления мобильным роботом.

1. Дайте определение системы автоматического управления.
2. Назовите примеры и области применения систем автоматического управления.
3. Охарактеризуйте особенности функционирования блока TurningRadius ToWheelPowers.
4. Прокомментируйте роль блока Timer в организации управления при движении по дуге.
5. Назовите переменную, отвечающую за организацию перераспределения мощности на дифференциальном приводе при движении по траектории в виде «восьмерки».

Лабораторная работа 6. Моделирование супервизорного и интерактивного управления.

1. Назовите принципиальные отличия организации супервизорного и интерактивного способов управления
2. Охарактеризуйте сервисы, используемые для реализации интерактивного управления.
3. Охарактеризуйте сервисы, используемые для реализации системы целеуказания.
4. Прокомментируйте особенности использования трекбола в сервисе SimpleDashboard.
5. Прокомментируйте особенности использования системы целеуказания в сервисе SimpleDashboard.

Лабораторная работа 7. Основы работы в среде симуляции Visual Simulation Environment.

1. Перечислите и охарактеризуйте тип файлов, входящих в шаблон нового DSS-сервиса.
2. Какие библиотеки требуется добавить в шаблон нового DSS-сервиса для работы со средой симуляции?
3. Для чего предназначена среда XNA Game Studio?
4. Прокомментируйте особенности добавления объекта в сцену.

Лабораторная работа 8. Управление средой симуляции Visual Simulation Environment.

1. Перечислите шаги, которые необходимо выполнить для получения трёхмерного изображения на плоскости.
2. Назовите основные функции выполняемые физическим и графическим движками.
3. В каких двух режимах может работать симулятор Microsoft RDS?
4. Как добавить объект в сцену средствами Visual Studio?
5. Как добавить объект в сцену средствами Microsoft RDS?

Лабораторная работа 9. Основы разработки модели робота в среде симуляции Visual Simulation Environment.

1. Какие роботизированные платформы поддерживает среда Microsoft Robotics Developer Studio R4?
2. Каким образом получены значения переменных, определенных в классе DifferentialDriveEntity?
3. В какой файл преобразуюся файлы с расширением .obj в программе Obj2bos.exe?
4. Прокомментируйте особенности работы в редакторе манифестов DSS.

Критерии оценивания компетенций

Оценка «отлично» выставляется студенту, если он выполняет в полном объеме и безошибочно задания оценочных средств по дисциплине, относящиеся к данной компетенции (индикаторам достижения компетенции), при этом демонстрирует на высоком уровне способность решать стандартные и нестандартные прикладные практические задачи проектирования систем реального времени.

Оценка «хорошо» выставляется студенту, если он выполняет, допуская незначительные ошибки, задания оценочных средств по дисциплине, относящиеся к данной компетенции (индикаторам достижения компетенции), при этом демонстрирует на среднем уровне способность решать только стандартные прикладные практические задачи проектирования систем реального времени.

Оценка «удовлетворительно» выставляется студенту, если он выполняет, допуская ошибки, исправление которых осуществляет с помощью наводящих вопросов преподавателя, задания оценочных средств по дисциплине, относящиеся к данной компетенции (индикаторам достижения компетенции), при этом демонстрирует на минимальном уровне способность решать стандартные прикладные практические задачи проектирования систем реального времени.

Оценка «неудовлетворительно» выставляется студенту, если он, выполняя задания оценочных средств по дисциплине, не достигает минимального уровня сформированности компетенции (индикаторам достижения компетенции), при этом не демонстрирует способность решать стандартные прикладные практические задачи проектирования информационно-коммуникационных систем и сетей, допускает грубые ошибки, которые не может исправить даже с помощью наводящих вопросов преподавателя.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Перечень основной литературы

1. Гриценко, Ю. Б. р; Системы реального времени : учебное пособие / Ю.Б. Гриценко ; Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР) ; Кафедра автоматизации обработки информации (АОИ). - Томск : ТУСУР, 2017. - 253 с. : ил. - http://biblioclub.ru/.

2. Мочалов, В. П. Системы реального времени : учеб. пособие (лабораторный практикум) / В. П. Мочалов, С. В. Яковлев ; Сев.-Кав. федер. ун-т. - Ставрополь : СКФУ, 2014. - 278 с. - Гриф: Доп. УМО. - ISBN 978-5-88648-913-2.

Перечень дополнительной литературы

1 Герасимов, А. В; Проектирование АСУТП с использованием SCADA-систем Электронный ресурс : Учебное пособие / А. В. Герасимов, А. С. Титовцев ; ред. Е. И. Шевченко. - Казань : Казанский национальный исследовательский технологический университет, 2014. - 128 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-7882-1514-3

2 Ившин, В.П. Беспроводная сеть сбора и передачи измерительной информации в АСУТП / В.П. Ившин : учебное пособие Электронный ресурс : Казанский национальный исследовательский технологический университет ; Казань, 2016. - 240 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-7882-1848-9

3 Мясников, В. И.; Операционные системы реального времени : лабораторный практикум / В.И. Мясников ; Поволжский государственный технологический университет. - Йошкар-Ола : ПГТУ, 2016. - 140 с. : табл., ил. - http://biblioclub.ru/. - ISBN 978-5-8158-1773-9

4 Федоров, Ю. Н. Справочник инженера по АСУТП. Проектирование и разработка : Учебно-практическое пособие / Федоров Ю. Н. - Вологда :

Инфра-Инженерия, 2016. - 928 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9729-0019-0

5 Яковлев, С. В. Системы реального времени : учеб. пособие / С.В. Яко-влев ; ФГБОУ Сев.-Кав. гос. техн. ун-т. - Ставрополь : Издательство СевКавГТУ, 2012. - 308 с. : ил. - Гриф: Доп. УМО для спец. 230100 "Инф. и выч. техника". - Библиогр.: с. 305. - ISBN 978-5-9296-0625-0.