СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Роботы сегодня входят в нашу жизнь в различных областях. Они летают в космос, исследуют другие планеты; помогают в военных целях – разминируют бомбы и разведывают обстановку с воздуха. В промышленности многие отрасли уже немыслимы без роботов: они собирают автомобили, помогают находить новые лекарства. Многие устройства, принимающие решения на основе полученных от сенсоров данных, тоже можно считать роботами, например, лифты, стиральные машины, системы антиблокировочного торможения, помогающие избежать аварий. Робот может управляться оператором, либо работать по заранее составленной программе. Использование роботов позволяет облегчить или вовсе заменить человеческий труд на производстве, в строительстве, при рутинной работе, при работе с тяжёлыми грузами, вредными материалами, а также в других тяжёлых или небезопасных для человека условиях [2-3, 5].

Современные подходы должны делать процесс обучения наглядным, а сам предмет увлекательным. Последние технологические достижения позволяют поднять обучение в области теории управления на качественно новый уровень. Среди них, в первую очередь, можно выделить мехатронные и робототехнические комплексы. В лабораторных работах студентов подобные комплексы могут послужить отличной заменой традиционному компьютерному моделированию [1, 3, 4, 7].

С образовательной точки зрения, данный подход обладает рядом преимуществ. Так, используя такое лабораторное оборудование, студенты имеют возможность понять основные принципы теории управления и оценить на практике результаты своих расчетов. Также студенты могут получить дополнительные знания в других областях, таких как теория информации, программирование, схемотехника и других. Таким образом, используя мехатронные и робототехнические лабораторные комплексы, мы можем предоставить студентам возможность создавать систему управления для устройства, начиная с расчета формул и заканчивая экспериментальной апробацией без риска повредить дорогое оборудование.

На данный момент создано множество мехатронных и робототехнических комплексов, которые повсеместно используются в образовательном процессе: VexRobotis, Mechatronics Control Kit, FestoDidactic, LEGO Mindstorms, OLLO, fischertechnik, Bioloid и др. Среди указанного разнообразия целесообразно выделить два наиболее популярных ввиду своей доступности, модульности и гибкости платформ: LEGO Mindstorms и Bioloid.

Данный выбор подтверждается множеством научных статей, посвященных как использованию данных наборов в образовательных целях, так и научным изысканиям на тему составления математических моделей, синтеза закона управления той или иной модели робота и т.д. Так, например, существует множество работ, посвященных устойчивости гуманоидного робота Bioloid, стоящего на одной ноге, и управлению мультиагентными системами роботов Lego [19-21].

Настоящее учебное пособие ставит целью ознакомить студентов с двумя наиболее популярными в образовательной среде робототехническими наборами (Lego Mindstorms 2.0 и Bioloid Premium Kit), их ПО и способами программирования. Помимо этого, пособие содержит ряд примеров программирования моделей роботов и их законов управления с применением методов современной теории управления.

Пособие ориентировано на студентов старших курсов, обучающихся по программе бакалаврской подготовки по направлению 221000.62 «Мехатроника и робототехника». При изложении материала авторы исходили из того, что читатель знаком с основами программирования на языке С, электроники, цифровой техники и теории автоматического управления в рамках дисциплин «Информатика», «Общая электротехника и электроника», «Микропроцессорные устройства систем управления» и «Теория автоматического управления», изучаемые при подготовке бакалавров на более ранних курсах.

Авторы надеются, что учебное пособие будет полезно студентам других специальностей, интересующихся вопросами реализации своих идей на реальном объекте, а также специалистам.

1 РОБОТОТЕХНИЧЕСКИЙ КОМЛЕКС LEGO MINDSTORMS

LEGO Mindstorms — это конструктор (набор сопрягаемых деталей и электронных блоков) для создания программируемого робота. Впервые представлен компанией LEGO в 1998 году. Через 8 лет (2006) в свет вышла модель LEGO Mindstorms NXT, а в 2009 — LEGO Mindstorms NXT 2.0.

Комплектация набора LEGO Mindstorms NXT 2.0:

- программируемый контроллер NXT с четырьмя входными и тремя выходными портами (рисунок 1.1).

Рисунок 1.1 – Программируемый контроллер NXT компании LEGO

- 4 сенсора (рисунки 1.2 – 1.4):

Ультразвуковой сенсор — позволяет роботу измерять расстояние до объекта и реагировать на движение;

Два сенсора нажатия – позволяют роботу реагировать на прикосновения;

Сенсор цвета включает в себя сразу три функции: умеет определять 6 цветов (белый, черный, желтый, красный, зеленый и голубой), интенсивность освещения и быть лампой подсветки.

Рисунок 1.2 – Датчик расстояния

Рисунок 1.3 – Датчик освещенности

Рисунок 1.4 – Датчик касания

- 3 интерактивных сервомотора со встроенными энкодерами (рисунок 1.5).

Рисунок 1.5 – Сервомотор со встроенным энкодером

- USB кабель для подключения контроллера к компьютеру.
- 7 соединительных кабелей.
- Инструкция.
- Диск с программным обеспечением для Windows и Mac OS.
- Тестовое поле для калибровки сенсоров.
- 613 различных Lego деталей (рисунок 1.6).

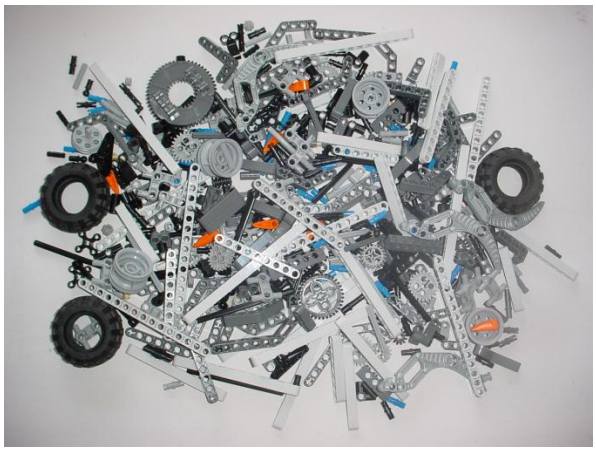
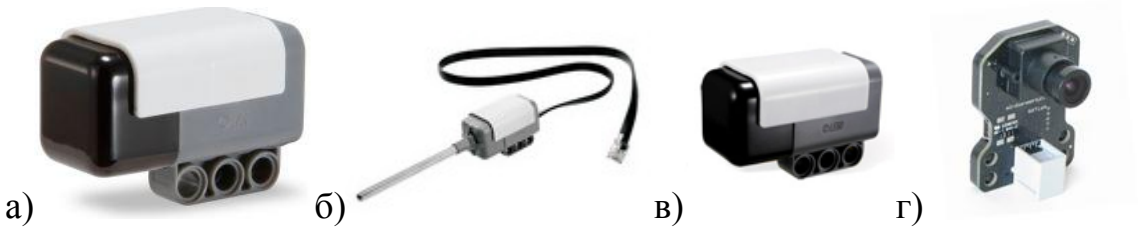


Рисунок 1.6 - Lego детали набора LEGO Mindstorms NXT 2.0

Помимо сенсоров, входящих в набор, существует множество других сенсоров как от компании LEGO, так и от сторонних производителей

(HiTechnic, Mindsensors, Dexter Industries). Некоторые из них приведены на рисунке 1.6:



а) б) в) г)

Рисунок 1.6 – а) Датчик-гироскоп HiTechnic NXT; б) Датчик температуры Lego 9749; в) Датчик-компас HiTechnic NXT; г) Видеосистема NXTCam-v4 от Mindsensors

На базе Lego Mindstorms можно получить различные модели мобильных роботов, балансиров, маятниковых систем, в том числе и нетривиальных, и т.д. В зависимости от поставленных задач и имеющегося опыта пользователь может выбрать среди множества необходимое ПО и язык программирования.

1.1 ОБЗОР СРЕД ПРОГРАММИРОВАНИЯ ДЛЯ LEGO MINDSTORMS NXT

Стандартное программное обеспечение LEGO MINDSTORMS NXT Software [9] и язык NXT-G приемлемы лишь на начальных этапах программирования, их ограниченные возможности не позволяют создавать сложные алгоритмы управления. В целях совершенствования представления алгоритмов управления было разработано множество различных сред и приложений для программирования NXT, многие из которых находятся в свободном доступе (таблица 1.1). В случае высоких требований к реализации проекта (многопоточность, возможности планирования заданий, обеспечения отказоустойчивости, взаимодействие с внешними устройствами в реальном времени и др.) более предпочтительным является использование объектно-ориентированных языков программирования.

Таблица 1.1 ПО для NXT, находящееся в свободном доступе в сети Интернет

ПО	Сайт
NXC	www.bricxcc.sourceforge.net
NXT-G	www.mindstorms.lego.com
RobotC	www.robotc.net
RoboLab	www.mindstorms.lego.com
Microsoft Robotics Developer Studio	www.mrds.com
LabVIEW Education Edition	www.ni.com
NBC	www.bricxcc.sourceforge.net
pbLua	www.hempeldesigngroup.com
ICON	www.teamhassenplug.org
leJOS NXJ	www.lejos.sourceforge.net
LEJOS OSEK	www.lejos-osek.sourceforge.net

Кроме того, возможность интеграции с LabView и MATLAB/Simulink позволяет использовать NXT в научно-исследовательских целях, в частности, при создании алгоритмов управления техническими системами. Так, когда алгоритм управления, основанный на математической модели, отлажен в одной из инженерных сред, интеграция с платформами MATLAB/Simulink или LabView позволяет в краткие сроки провести испытания на реальном объекте, без каких-либо дополнительных усилий.

В таблице 1.2 приведены некоторые из довольного большого многообразия сред для программирования LEGO Mindstorms NXT и их основные характеристики.

Таблица 1.2 Некоторые среды программирования LEGO Mindstorms NXT.

	Robot C	NXC	leJOS NXJ	LabView Toolkit	nxtOSEK	MATLAB/ Simulink
Язык	Текстовый	Текстовый	Текстовый	Графический	Текстовый	Графический
Операционная система	Стандартная	Стандартная	leJOS	Стандартная	leJOS	leJOS
Синтаксис	C	C-подобный	Java	Блоки	C/C++	C, Simulink-блоки
Многопоточность	Есть	Есть	Есть	Есть	Есть	Есть
События	Есть	Нет	Есть	Нет	Есть	Есть
Операции с плавающей точкой	Есть	Нет	Есть	Нет	Есть	Нет

Необхо-димая лицензия	Лицензия	В свободном доступе	В свободном доступе	LabView	В свободном доступе	MATLAB/ Simulink

В академической среде наиболее популярны среды LEGO MINDSTORM Toolkit для LabView и Embedded Coder Robot NXT для MATLAB/Simulink. Программа, написанная в LabView, использует стандартную прошивку LEGO. Однако LabView является компилятором своего собственного языка, что зачастую приводит к необходимости написания собственных библиотек при решении задач, требующих сложные вычисления. Для использования MATLAB необходимо произвести предварительную установку системы leJOS.

Программные продукты, основанные на языках высокого уровня и находящиеся в свободном доступе, представлены системами NXC и leJOS. NXC (Not Exactly C) – это C-подобный язык для программирования контроллера NXT с использованием многозадачности, функционирующий в рамках стандартной оболочки LEGO. С использованием среды программирования IDE, также находящейся в свободном доступе, можно создавать сложные алгоритмы управления в реальном времени, включающие расширенный обмен данными и управление файлами.

LeJOS является аналогом операционной системы для контроллера NXT [9]. В основе leJOS лежит объектно-ориентированный язык программирования (Java). leJOS позволяет поддерживать работу с распределением потоков, массивами, рекурсиями и т.д.

1.2 ИСПОЛЬЗОВАНИЕ ПАКЕТА EMBEDDED CODER ROBOT NXT ДЛЯ ГРАФИЧЕСКОГО МОДЕЛИРОВАНИЯ И ГЕНЕРАЦИИ КОДА НА ПЛАТФОРМЕ nxtOSEK

NxtOSEK (до мая 2008 года – LEJOS OSEK) является гибридом двух общедоступных платформ: leJOS NXJ и TOPPERS OSEK [11, 12]. LeJOS NXJ предоставляет возможность ANSI C/C++ программирования для NXT с использованием API-функций. Доступ к сенсорам, моторам и другим

устройствам NXT реализуется напрямую, что позволяет существенно сэкономить время. TOPPERS OSEK обеспечивает поддержку многозадачности в реальном времени.

Кроме программирования на языках высокого уровня nxtOSEK дает возможность использовать Embedded Coder Robot NXT (далее ECRobot) – общедоступный специализированный пакет расширения MATLAB/Simulink, среду для графического моделирования и генерации кода. Пакет позволяет создавать и быстро конвертировать готовые Simulink-модели в исполняемый роботом файл. Именно этот аспект использования nxtOSEK будет более подробно рассмотрен далее.

1.2.1 Установка и основы работы с пакетом ECRobot

Для корректной работы ECRobot необходимо предварительно установить ряд необходимых программ и сменить операционную систему NXT. Далее приведен подробный процесс установки [13]:

1. Убедитесь, что на вашем компьютере MATLAB установлен в директорию, путь к которой не содержит пробелов. Например:
   а. “C:\MATLAB” – **Правильно**
   б. “C:\MATLAB R2010a” – **Неправильно**
2. Убедитесь, что у вас установлены MATLAB (версии не ниже R2008b), Simulink, Real-Time Workshop and Real-Time Workshop Embedded Coder. Для этого в командном окне MATLAB необходимо набрать команду «ver», которая выведет информацию о версиях MATLAB и компонентов.
3. Установка Cygwin / GNU Make
   а. Скачайте Cygwin 1.5.x или более новую версию с веб-сайт Cygwin: http://www.cygwin.com/
   б. Установите Cygwin в “C:\cygwin” (рисунок 1.6)
   в. Выберите “make” в части “Devel” списка файлов.

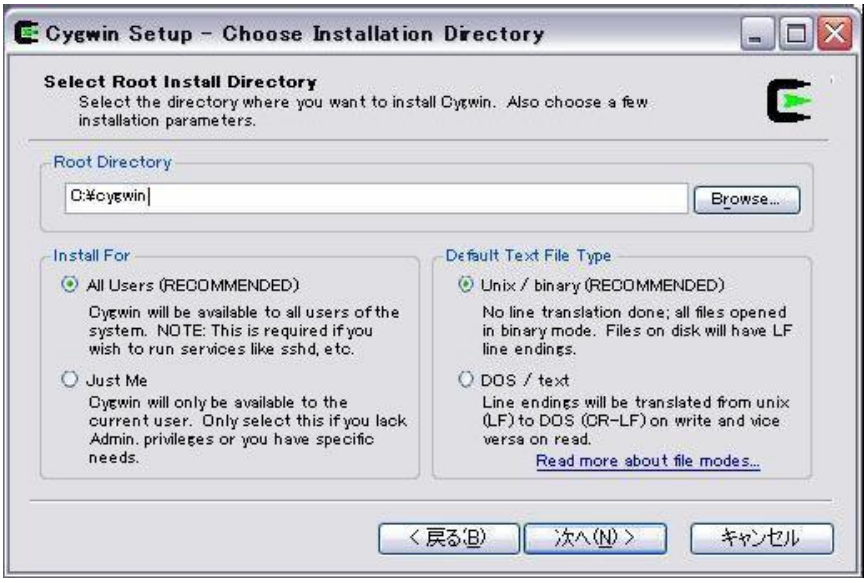


Рисунок 1.6 – Установщик Cygwin

4. Установка GNU ARM
   а. Скачайте GCC-4.0.2 бинарный установщик из секции FILES с веб-сайта GNU ARM: http://www.gnuarm.com/
   б. Запустите установщик и установите GNU ARM в C:/cygwin/GNUARM
   в. Сконфигурируйте установочный диалог как показано на рисунке 1.7
   г. Уберите галочку с пункта Install Cygwin DLLs, потому что Cygwin уже установлен (рисунок 1.8)
   д. В конце установки на вопрос о добавлении установочной директории ответьте нет.

Рисунок 1.7 – установщик GNU ARM(1)

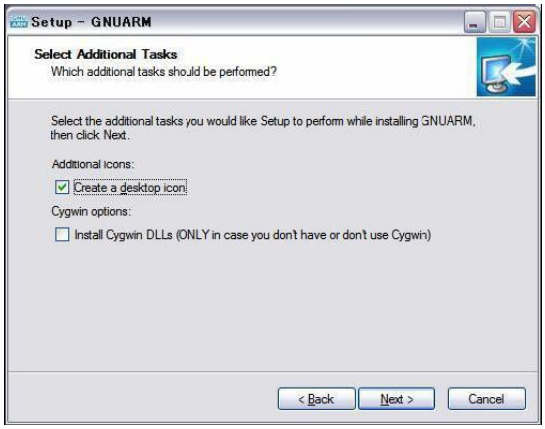


Рисунок 1.8 – установщик GNU ARM(2)

5. Установка драйвера NXT USB:
   а. (Если стандартное программное обеспечение LEGO уже установлено на вашем компьютере пропустите этот шаг)
   Скачайте MINSTORMS NXT Driver v1.02 с веб-сайта обновлений ПО для LEGO:
   http://mindstorms.lego.com/Support/Updates/ - см. Рис. 1.9

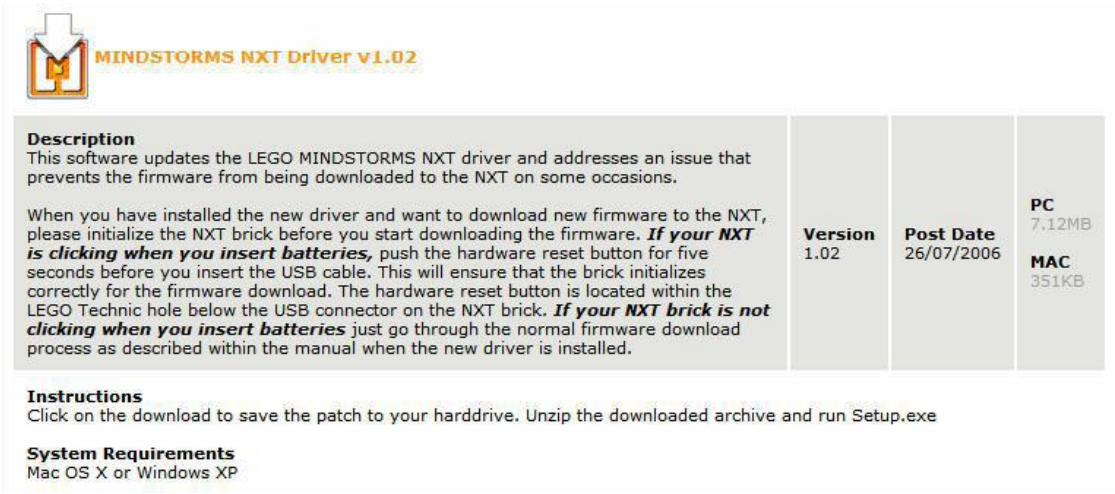


Рисунок 1.9 – драйвер Mindstorms

6. Установка NeXTTool:
   a. Скачайте NeXTTool с http://bricxcc.sourceforge.net/utilities.html и распакуйте файлы в “c:\cygwin/nexttool”
7. Установка стандартной прошивки NXT:
   a. Скачайте модернизированную прошивку NXT и установите по шагам, описанным ниже: http://bricxcc.sourceforge.net/lms_arm_jch.zip (версии 1.06 или более поздней)
   б. Распакуйте архив в папку “C:\cygwin\nexttool” – вы должны увидеть файл с названием типа “lms_arm_nbcnxc_**X**”, где **X**=106 для версии 1.06
   в. Откройте командную строку – В Windows Пуск=>Выполнить=> и затем введите “cmd” и выберите ok
   г. Напечатайте “c:” в строке
   д. Напечатайте “cd cygwin\NeXTTool” в строке
   е. Подключите контроллер NXT к порту USB и включите его.
   Для загрузки прошивки, напечатайте (не скопируйте и вставьте) команды, приведенные ниже, где “lms_arm_nbcnxc_106” имя файла, который вы загрузили на шаге b.

```
nexttool /COM=usb –firmware=lms_arm_nbcnxc_106.rfw
```

8. Установка Embedded Coder Robot NXT:
   a. Скачайте и распакуйте ECROBOT NXT из http://www.mathworks.fr/matlabcentral/fileexchange/13399 в корневой каталог установочной директории MATLAB.
9. Установка nxtOSEK и запуск ecrobotnxtsetup.m:

а. Скачайте и распакуйте nxtOSEK с веб-сайта nxtOSEK(http://lejos-osek.sourceforge.net/download.htm?group_id=196690) в папку ecrobotNXT/environment (рисунок 1.10).

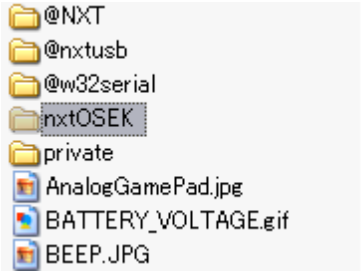


Рисунок 1.10 – nxtOSEK в папке ecrobotNXT/environment

б. Зайдите в папку и запустите m-файл ecrobotmxtsetup.m показанный на рисунке 1.11. Далее выполнится оставшийся процесс установки.

Рисунок 1.11 – ecrobotnxtsetup

1.2.2 Функционирование пакета ECRobot

При создании модели в дополнение к стандартным средствам MATLAB/Simulink используются блоки ECRobot NXT Blockset, которые появятся в Simulink Library Browser после установки ECRobot. Блоки представляют интерфейс взаимодействия контроллера NXT с внешними устройствами (моторами, датчиками, Bluetooth-устройствами и т.д.), процесс управления и обмена информацией. Подробное описание блоков приведено в разделе 1.2.3 «Основные блоки ECRobot NXT»

После того, как окончательная модель сформирована и отлажена, запускается генерация С-кода. Эту функцию выполняет пакет расширения MATLAB Real-Time Workshop Embedded Coder совместно с Real-Time Workshop.

Далее необходимо скомпилировать готовый код на С в исполняемый процессором файл с расширением «.rxe». Для этого служит приложение-компилятор GNU ARM toolchain 4.0.2. Для корректной работы приложения необходим Cygwin (версии не ниже 1.5) – Linux-подобная среда для Windows, которая позволяет запускать некоторые Linux-приложения под управлением Windows. Cygwin содержит GCC (GNU Compiler Collection), обеспечивая поддержку функционирования главного процессора NXT.

Последний шаг – загрузка полученного файла в NXT. Для этого используется NeXTTool – консольное приложение, позволяющее загружать с компьютера на борт NXT исполняемые файлы (с расширением «.rxe») и операционные системы (с расширением «.rfw»). Для корректной работы NeXTTool необходим LEGO Mindstorms NXT Driver – драйвер для связи Windows с NXT через USB-кабель. Однако загружаемая таким образом программа не должна превышать по объему 64 кБ. Для загрузки более «тяжелых» проектов служит NXT BIOS (максимальный объем 224 кБ).

Для прохождения всех этапов от генерации С-кода до загрузки исполняемого файла на NXT пользователю необходимо нажать всего на две кнопки (рисунок 1.12). Кнопка «Generate code and built the generated code» - кнопка запуска генерации исходного кода модели на языке С и последующей компиляции в исполняемый модуль. В итоге в каталоге «nxtprj» текущей директории появятся файл с расширением .c с инициализацией функций программы появится файл с расширением .rxe, который может быть напрямую загружен в NXT, а в каталоге «***_app_ert_rtw» (где *** - название проекта) появится файл «***_app.c» с кодом, отражающим функционирование будущего исполняемого модуля. Кнопка «Download (NXT enhanced firmware)» - кнопка запуска процесса записи исполняемого модуля в память блока NXT (предварительно необходимо установить соединение по USB-кабелю). После этого файл может быть запущен на NXT. При этом в командном окне MATLAB будут появляться служебные сообщения о результатах завершения каждого из этапов.

Рисунок 1.12 – Кнопки в Simulink-модели для генерации, компиляции и загрузки С-кода в контроллер NXT

Для работы загруженного файла необходимо, чтобы на NXT была установлена операционная система nxtOSEK, которая позволяет NXT наряду с обычными LEGO MINDSTORMS NXT Sofware программами выполнять двоичные файлы, скомпилированные из С кода («.rxe»-приложения).

Таким образом, nxtOSEK программы выполняют в машинном коде главным процессором NXT, что обеспечивает быстрое исполнение при малом потреблении оперативной памяти (сама nxtOSEK потребляет около 10 кБ).

Одной из ключевых особенностей использования nxtOSEK совместно с ECRobot является автономность робота. Благодаря этому становится возможным создание распределенных систем управления группой роботов, основанных на принципах децентрализованного принятия решений о действиях каждого робота группы.

1.2.3 Основные блоки ECRobot NXT

Если ECRobot NXT был установлен успешно, то набор блоков ECRobot NXT появится в браузере библиотек Simulink (рисунок 1.13).

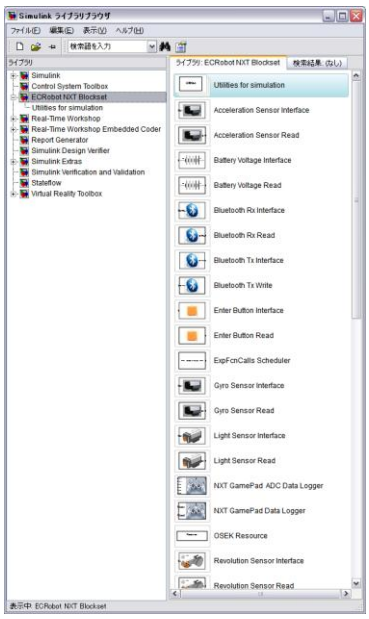


Рисунок 1.13 - Набор блоков ECRobot в браузере библиотек Simulink

Детально рассмотрим основные блоки, используемые в ECRobot NXT.

Revolution Sensor блоки:

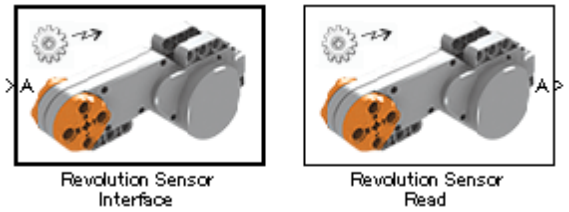


Рисунок 1.14 – Блоки Revolution Sensor

Блоки Revolution Sensor используются для измерения изменения положения сервопривода. Блок Revolution Sensor Interface - это блок, считывающий данные с модели контроллера NXT и передающий их на сервопривод. Блок Revolution Sensor Read используется для получения данных с сервопривода. В режиме симуляции эти блоки не используются; обычно они используются для включения API предполагаемого устройства в генерируемый код (при моделировании они выполняют функцию сигнальных блоков In\Out).

Таблица 1.3 Характеристики блоков Revolution Sensor

Тип данных	int32
Измерение	[1 1]
Диапазон данных	Диапазон int32 [град]
ID порта	A/B/C

Servo Motor блоки:

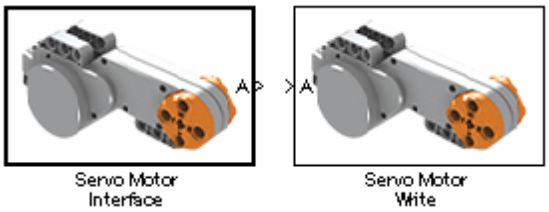


Рисунок 1.15 – Блоки Servo Motor

Блоки Servo Motor используются для управления сервоприводами. Servo Motor состоит из двух блоков. Блок Servo Motor Interface считывает данные с модели контроллера NXT и передает их на сервопривод. Блок Servo Motor Write используется для задания параметров вращения серводвигателей. При моделировании они выполняют функцию сигнальных блоков In\Out.

Таблица 1.4 Характеристики блоков Servo Motor

Тип данных	int8
Измерение	[1 1]
Диапазон данных	от -100 до 100
ID порта	A/B/C
Режим	Торможение/Движение

Battery Voltage блоки:

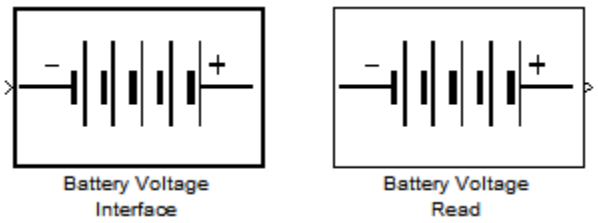


Рисунок 1.16 – Блоки Battery Voltage

Блоки Battery Voltage используются для определения заряда аккумулятора NXT. Блок Battery Voltage Interface это блок, для вывода данных с модели контроллера NXT. Блок Battery Voltage Read используется для считывания данных об уровне заряда аккумулятора.

Таблица 1.5 Характеристики блоков Battery Voltage

Тип данных	Int16
Измерение	[1 1]
Диапазон данных	0 до возможного максимального значения напряжения в [мВ] (например, 9000=9.000В)
ID порта	Нет

System Clock блоки:

Рисунок 1.17 – Блоки System Clock

Блоки System Clock используются для чтения данных системных часов NXT. Блок System Clock Interface не имеет вход/выход, однако он доставляет значения блоку чтения при моделировании. Как правило, поддержку системных часов во время симуляции обеспечивает блок System Clock Read. Блок System Clock Read используется для получения данных с системных часов. В реальном NXT системные часы начинают работать с 0 в момент включения устройства.

Таблица 1.5 Характеристики блоков System Clock

Тип данных	uint32
Измерение	[1 1]
Диапазон данных	от 0 до максимально возможного значения uint32 в миллисекундах
ID порта	Нет

Sound Volume Tone блоки:

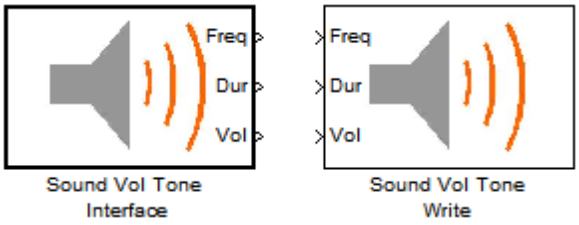


Рисунок 1.18 – Блоки Sound Volume Tone

Блоки Sound Volume Tone используются для генерации звука, который определяется громкостью, частотой и длительностью.

Таблица 1.6 Характеристики блоков Sound Volume Tone

Тип данных	Частота: uint32, длительность: uint32
Измерение	[1 1]
Диапазон данных	Частота: от 31 [Гц] до 14080 [Гц] Длительность: от 10 до 2560 [мс] Звук: от 0 до 100 (0 - звук выключен)
ID порта	Нет

Ultrasonic Sensor блоки:

Рисунок 1.19 - Блоки Ultrasonic Sensor

Блоки Ultrasonic Sensor используются для измерения дистанции до препятствий или бесконтактного определения самих препятствий. Ultrasonic Sensor Interface – выходной блок интерфейса контроллера NXT. Ultrasonic

Sensor Read используется для получения данных с ультразвукового датчика расстояния.

Таблица 1.7 Характеристики блоков Ultrasonic Sensor

Тип данных	Int32
Измерение	[1 1]
Диапазон данных	от 0 до 255 [см], -1 (датчик не готов для измерений)
ID порта	S1/S2/S3/S4

Light Sensor блоки:

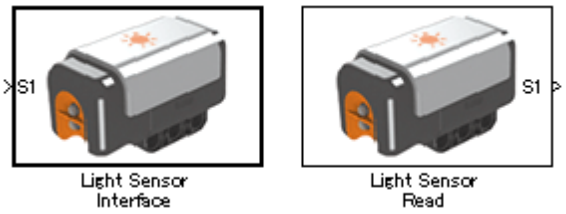


Рисунок 1.20 - Блоки Light Sensor

Блоки Light Sensor используются для измерения яркости света, попадающего на диод датчика. Чем больше измеряемое значение, тем темнее поверхность (или меньше отражение). Датчик света состоит из двух блоков. Light Sensor Interface – выходной блок интерфейса контроллера NXT. Light Sensor Read используется для получения данных с датчика света.

Таблица 1.8 Характеристики блоков Light Sensor

Тип данных	Unt16
Измерение	[1 1]
Диапазон данных	от 0 до 1023
ID порта	S1/S2/S3/S4

Sound Sensor блоки:

Рисунок 1.21 - Блоки Sound Sensor

Блоки Sound Sensor используются для измерения громкости звука. Чем меньше значение, тем громче звук. Sound Sensor Interface – выходной блок интерфейса контроллера NXT. Sound Sensor Read используется для получения данных с датчика звука.

Таблица 1.9 Характеристики блоков Sound Sensor

Тип данных	Unt16
Измерение	[1 1]
Диапазон данных	от 0 до 1023
ID порта	S1/S2/S3/S4

Touch Sensor блоки:

Рисунок 1.22 - Блоки Touch Sensor

Блоки Touch Sensor используются для обнаружения контакта с препятствием. Если датчик имеет контакт с препятствием, датчик возвращает 1. Touch Sensor Interface – выходной блок интерфейса контроллера NXT. Touch Sensor Read используется для получения данных с датчика касания.

Таблица 1.10 Характеристики блоков Touch Sensor

Тип данных	Unt8
Измерение	[1 1]
Диапазон данных	0 (нет касания), 1 (есть касание)
ID порта	S1/S2/S3/S4

Bluetooth Rx блоки:

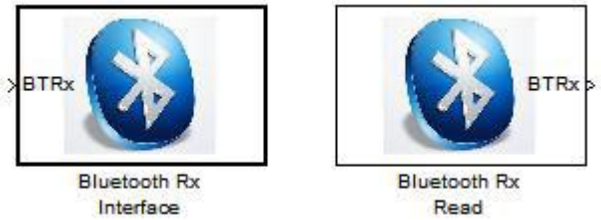


Рисунок 1.23 - Блоки Bluetooth Rx

Блоки Bluetooth Rx представляют беспроводной протокол Bluetooth для получения Bluetooth-пакетов. Процесс получения пакетов представлен двумя блоками. Bluetooth Rx Interface – интерфейсный блок получения пакетов для взаимодействия с контроллером NXT. Bluetooth Rx Read используется для получения данных. Блоки Bluetooth Rx позволяют использовать только однонаправленную связь с моделью (прием данных), а также только соединение ПК-NXT (не поддерживают NXT-NXT).

Таблица 1.11 Характеристики блоков Bluetooth Rx

Тип данных	Uint8
Измерение	32
Диапазон данных	от 0 до 255
ID порта	Нет

Bluetooth Tx блоки:

Рисунок 1.24 - Блоки Bluetooth Tx

Блоки Bluetooth Tx представляют беспроводной протокол Bluetooth для отправки Bluetooth-пакетов. Процесс отправки пакетов представлен двумя блоками. Bluetooth Tx Interface – блок интерфейса, для взаимодействия с контроллером NXT. Bluetooth Tx Write используется для передачи данных. Блоки Bluetooth Tx позволяют использовать только однонаправленную связь с моделью (передача данных) и поддерживают соединение ПК-NXT и NXT-NXT.

Таблица 1.12 Характеристики блоков Bluetooth Tx

Тип данных	Uint8
Измерение	32
Диапазон данных	от 0 до 255
ID порта	Нет

NXT GamePad Data Logger блок:

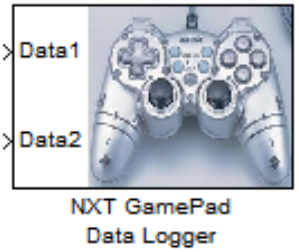


Рисунок 1.25 - Блок NXT GamePad Data Logger

NxtOSEK содержит утилиту, которая называется NXT GamePad. Эта утилита позволяет пользователю удаленно управлять NXT, который содержит nxtOSEK через Bluetooth, используя при этом аналоговый контроллер (джойстик). Начиная с версии 1.01 пользователю также предоставлены возможности выбора функций управления с помощью джойстика “Analog Stick Control” и сбора данных с устройства “NXT Data Acquisition”, в зависимости от целей. Даже если у пользователя нет аналогового игрового контроллера, NXT GamePad можно использовать только для получения данных о работе устройства за время использования [15].

Для передачи данных на ПК необходимо запустить приложение NXTGamePad.exe и установить значение COM-порта в соответствии с установленным соединением. Входы Data1 и Data2 можно использовать для получения данных с Analog GamePad. Полученные данные могут быть

сохранены в CSV файл и это может быть полезно для анализа работы приложения NXT в MATLAB. NXT GamePad Data Logger не должен использоваться вместе с блоком Bluetooth Tx Write в модели.

Таблица 1.13 Характеристики блока NXT GamePad Data Logger

Тип данных	int8
Измерение	[1 1]
Диапазон данных	от -128 до 127
ID порта	Нет

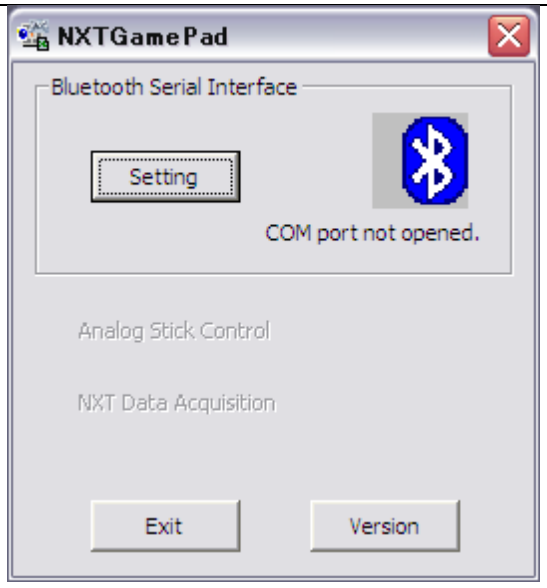


Рисунок 1.26 - Утилита NXT GamePad

NXT GamePad ADC Data Logger блок:

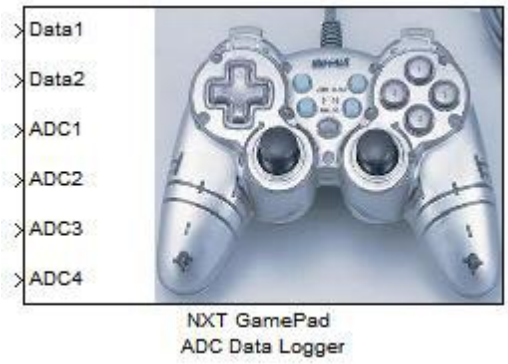


Рисунок 1.27 - Блок NXT GamePad ADC Data Logger

NXT GamePad ADC Data Logger имеет такой же функционал как NXT GamePad Data Logger, но пользователь может сконфигурировать 4 сигнала для получения и записи данных. NXT GamePad ADC Data Logger не должен использоваться с блоками Bluetooth Tx Write в модели.

Таблица 1.11 Характеристики блока NXT GamePad ADC Data Logger

Название входа	Тип данных	Диапазон данных	Размерность
Data1/Data2	Int8	от -128 до 127	1
ADC1/ADC2/ADC3/ADC4	Int16	от -32768 до 32767	1

Gyro Sensor блоки:

Рисунок 1.28 - Блоки Gyro Sensor

Блоки Gyro Sensor (гироскоп) используются для считывания значений с гироскопа, представляющих изменение угловых градусов в секунду. Gyro Sensor Interface - блок интерфейса для модели контроллера NXT. Блок Gyro Sensor Read используется для получения данных с гироскопа.

Таблица 1.12 Характеристики блоков Gyro Sensor

Тип данных	Uint16
Измерение	[1 1]
Диапазон данных	от 0 до 1023
ID порта	S1/S2/S3/S4

Exported Function-Calls Scheduler блок:

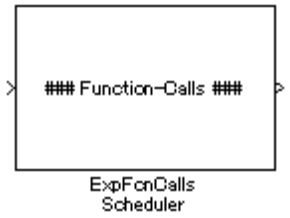


Рисунок 1.29 - Блок Exported Function-Calls Scheduler

Обычно системы управления встроенными системами требуют выполнения функций в определенные моменты времени (например, при инициализации, вызовах функций по прерыванию или с заданным периодом). Блок Exported Function-Calls Scheduler управляет моментами исполнения функций в подсистеме вызова функций Function-Call во время симуляции, согласно установленным параметрам блока Block Parameters.

OSEK Resource блок:

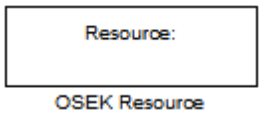


Рисунок 1.30 - Блок OSEK Resource

Блок OSEK Resource осуществляет доступ к управлению ресурсами OSEK GetResource/ReleaseResource API в начале/конце критических участков кода, занимающих ресурсы, которые нельзя использовать одновременно с другими процессами. В Simulink, критические участки вместе с используемыми ресурсами следует располагать в подсистемы Atomic Subsystem. Для задания идентификатора ресурса его необходимо определить

в блоке Exported Function-Call Scheduler. Блок OSEK Resource не влияет на моделирование и необходим только для генерации кода.

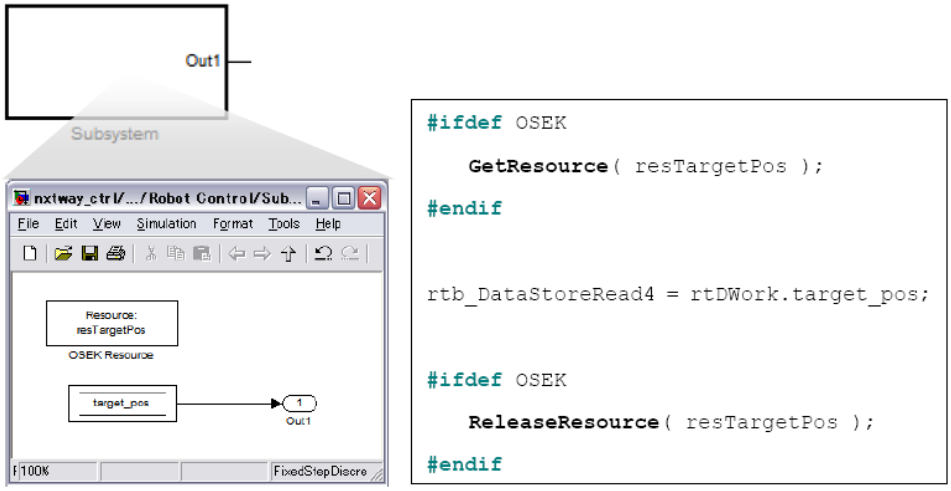


Рисунок 1.31 - Пример блока OSEK Resource и сгенерированного кода

1.3 МОДЕЛИРОВАНИЕ И ПРОГРАММИРОВАНИЕ РОБОТОТЕХНИЧЕСКИХ КОМПЛЕКСОВ НА ОСНОВЕ НАБОРА LEGO MINDSTORMS

В данной части представлены некоторые модели роботов на основе LEGO Mindstorms, описаны их математические модели и программы управления, написанные с использованием пакета ECRobot NXT. Инструкции по сборке моделей приведены в приложении учебного пособия.

В качестве примера рассмотрим модель робота «исследователь» (рисунок 1.32). [14] Датчик касания подключен к входному порту № 1, ультразвуковой дальномер – к входному порту № 4. Левый мотор подключен к выходному порту C, правый мотор – к выходному порту B.

Рисунок 1.32 – Модель робота «Исследователь»

Для «исследователя» написана простейшая Simulink-модель «Test.mdl» (рисунок 1.33), реализующая объезд роботом препятствий. Робот движется прямолинейно, пока не «увидит» вблизи препятствие либо пока не столкнется с ним. После этого он поворачивает и продолжает движение.

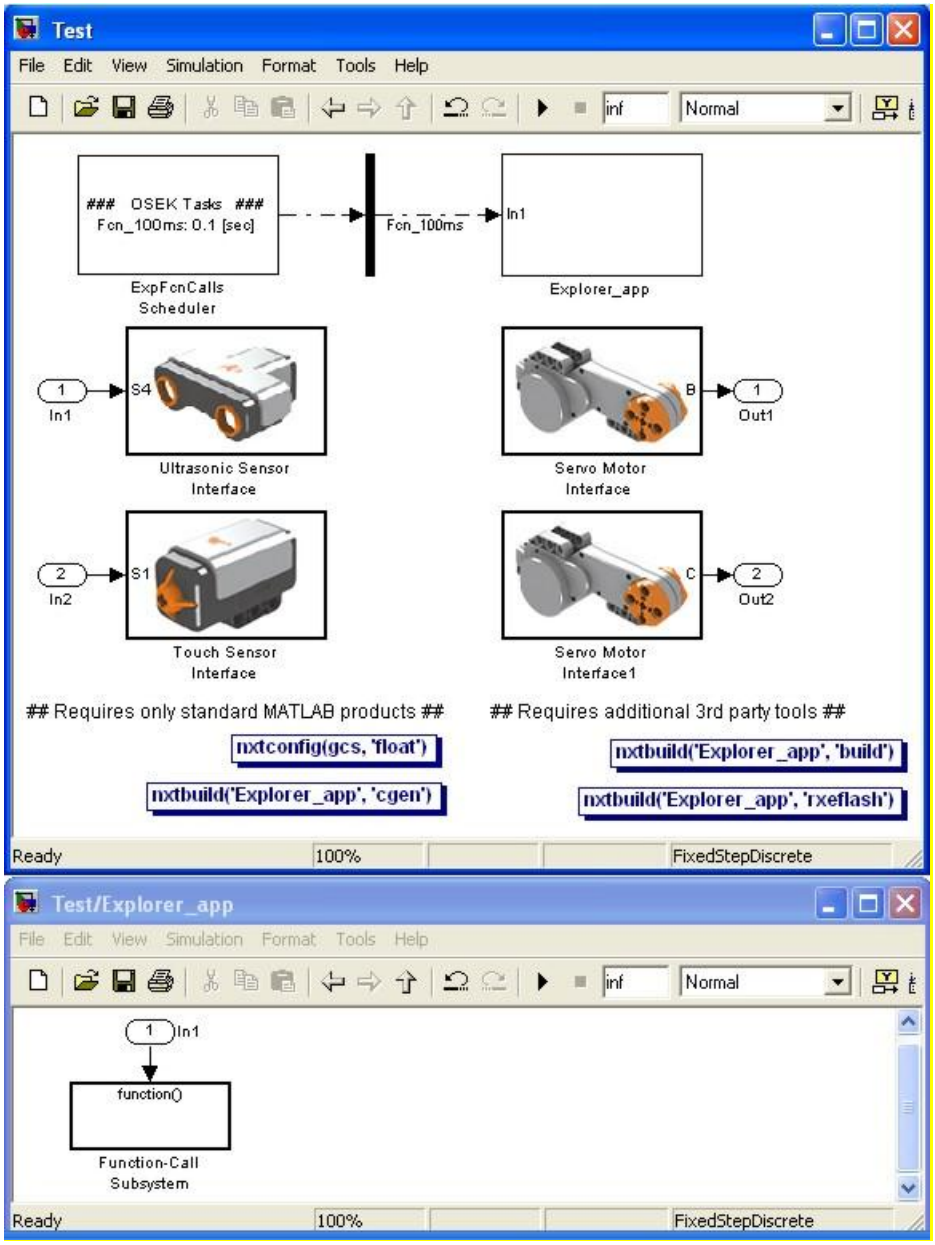


Рисунок 1.33 – Simulink-модель управления роботом «test.mdl»

Блок «ExpFcnCalls Scheduler» (планировщик) служит для задания имен функций и их периодичности запуска. В данном примере задана одна функция – «Fcn_100ms» задающая период равным 0,1 сек.

В блоке «Explorer_app» описана функция управления роботом. Данный блок отражает функциональность будущего исполняемого файла. Он содержит в себе один вход и одну подфункцию. Планировщик подает импульсы на вход с заданной частотой (0,01 сек), запуская выполнение подфункции «Function-Call Subsystem».

Подфункция «Function-Call Subsystem» работает следующим образом (рисунок 1.34). Если сигнал с ультразвукового датчика превышает 20 см (препятствие далеко), то на вход сумматора поступает 0, в противном случае 1. На втором входе сумматора появится 1 в случае срабатывания датчика касания (блок «Touch Sensor Read») или 0 в противном случае.

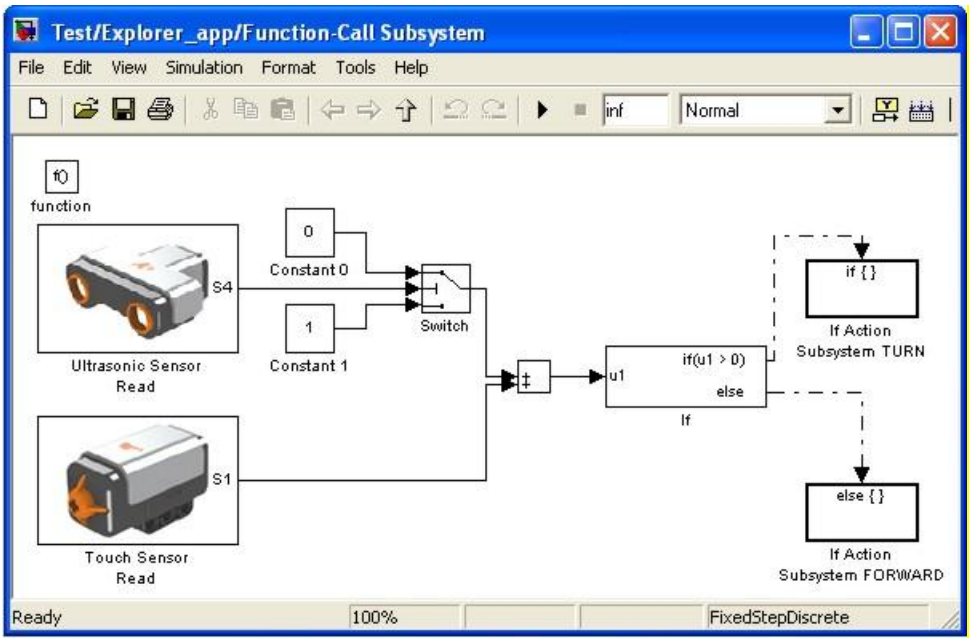


Рисунок 1.34 - Подфункция «Function-Call Subsystem»

Таким образом, на выходе сумматора будет величина, большая 0, если хотя бы один из датчиков зафиксировал препятствие. В зависимости от этого будет выполнена либо подфункция «If Action Subsystem TURN» (поворот робота), либо подфункция «If Action Subsystem FORWARD» (движение вперед). В первом случае один из моторов прекращает вращение, а другой на полной мощности осуществляет движение назад. В случае отсутствия препятствий на оба мотора подается одинаковая мощность, равная 70, и робот движется вперед (рисунок 1.35).

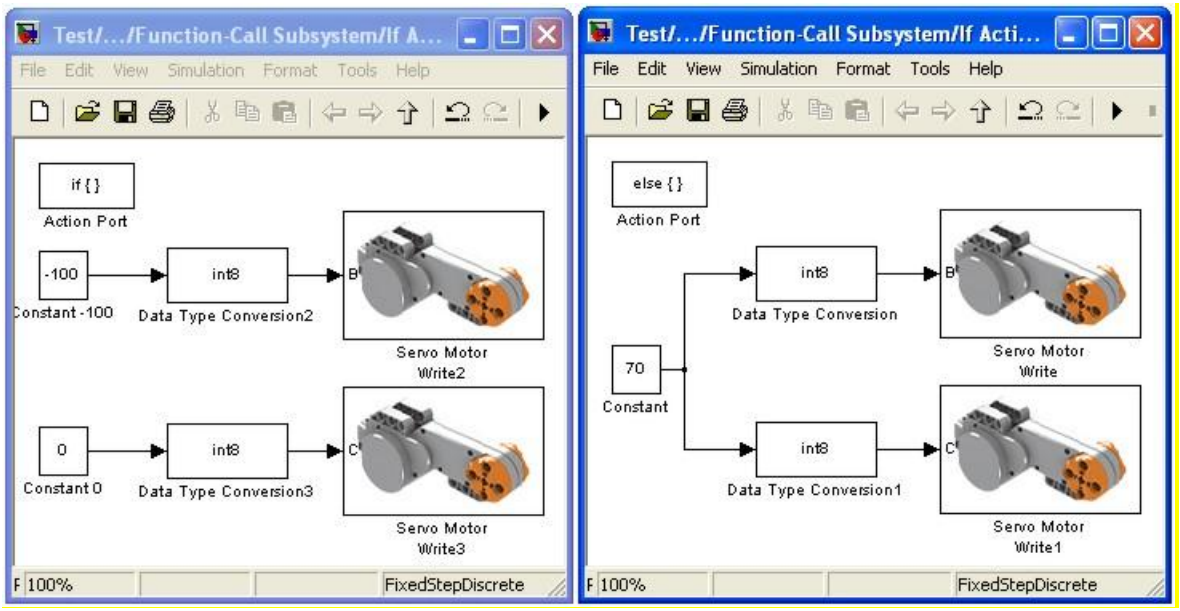


Рисунок 1.35 - Подфункция управления моторами

Далее рассмотрим управляющие кнопки, реализующие преобразование модели в исполняемый код.

Кнопка «nxtconfig (gsc, 'float')» - кнопка начальной инициализации блока генерации кода.

Кнопка «nxtbuild ('Explorer_app', 'cgen')» - кнопка запуска генерации исходного кода модели (код генерируется на языке С для последующей компиляции в исполняемый модуль).

По окончании генерации в каталоге «nxtprj» текущей директории появится файл «ecrobot_main.c» с инициализацией функции «Fcn_10ms», а в каталоге «Explorer_app_ert_rtw» появится файл «Explorer_app.c» с кодом, отражающем функционирование блока «Explorer_app» и будущего исполняемого модуля.

Кнопка «nxtbuild ('Explorer_app', 'build')» - кнопка компиляции исполняемого модуля. В итоге в каталоге «nxtprj» текущей директории должен появиться файл «Explorer_app.rxe» который напрямую может быть загружен в NXT.

Кнопка «nxtbuild ('Explorer_app', 'rxeflash')» - кнопка запуска процесса записи исполняемого модуля («Explorer_app.rxe») в память NXT (при условии, что с ним установлено соединение через USB-кабель).

Далее этот файл может быть запущен на NXT. Таким образом, будет реализовано желаемое поведение робота в автономном режиме. [14]

В разделах 1.3.1-1.3.3 представлены сложные модели балансирующих роботов, методы расчета их математических моделей, расчета регуляторов,

алгоритмы управления и их программная реализация в nxtOSEK. Инструкции по сборке представлены на сайте www.nxtprograms.com/NXT2/explorer/index.html [14].

1.3.1 Модель одноколейного двухколесного робота «Мотобот»

В данном разделе описывается макет автономного мобильного робота «Мотобот», являющегося моделью двухколесного мотоцикла (рисунок 1.36).

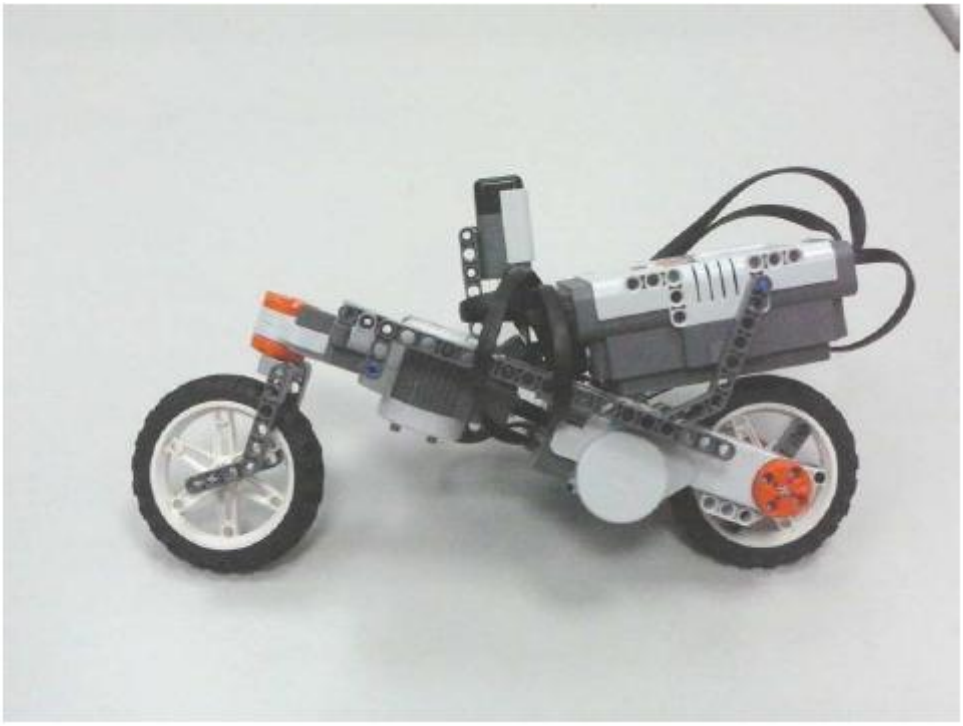


Рисунок 1.36 - Автономный мобильный робот «Мотобот»,

Анализируется математическая модель движения двухколесного одноколейного робота на плоскости, на основе которой синтезируются алгоритмы автоматического управления. Задача может быть решена путем управления рулевым колесом и скоростью ведущего колеса. Решены локальные задачи стабилизации вертикального положения равновесия, движения вдоль прямой с заданной скоростью. Реализовано дистанционное управление траекторией движения с помощью джойстика и радиоканала связи.

1.3.1.1 Математическая модель робота «Мотобот»

Математическая модель мотоцикла является нелинейной и включает несколько важных параметров таких, как масса мотоцикла, коэффициент трения, расстояние между проекциями колес, моменты инерции составных частей относительно центра масс и другие. Математическая модель состоит из четырех основных частей: рамы, двух колес и рулевой вилки. Приемлемая модель имеет 7 степеней свободы, из них 3 степени свободы угловых

скоростей. Модель может быть соответственно параметризованной с помощью следующих переменных: угол крена вдоль поперечной оси ϕ, угол поворота с помощью рулевого колеса δ, и изменение уровня вращения заднего колеса θ. [5]

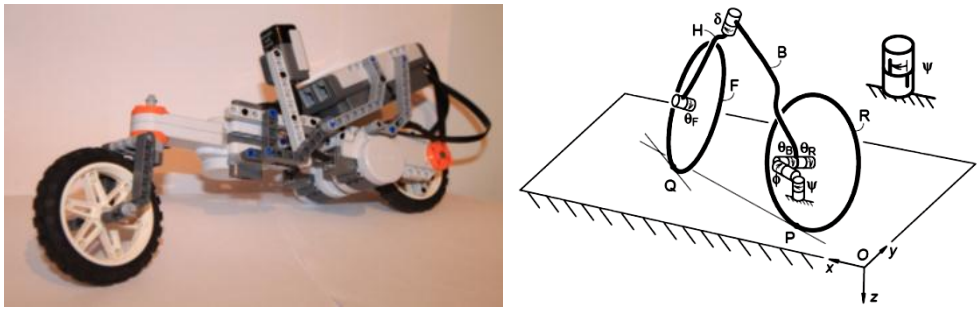


Рисунок 1.37 - Прототип мобильного робота «Мотобот» и схематическое изображение углов положения составных частей

Рассмотрим Декартову систему координат с центром в точке O и осями x, y и z (рисунок 1.37). Заднее колесо R, соединенное с рамой при наличии водителя имеет центр масс в точке B. Центр масс B размещен в следующей области: $x_B = 0$, $y_B = 0$, $z_B < 0$. Момент инерции рамы относительно центра масс представлен матрицей моментов инерции размерности (3×3), где все массы симметричны относительно проекции xz. Центр масс передней части рамы с вилкой и рулем H находится в следующей области $x_H = 0$, $y_H = 0$, $z_H < 0$. Матрицы моментов инерции передней и задней сборок:

$$\mathbf{I}_B = \begin{bmatrix} I_{Bxx} & 0 & I_{Bxz} \\ 0 & I_{Byy} & 0 \\ I_{Bxz} & 0 & I_{Bzz} \end{bmatrix}, \ \mathbf{I}_H = \begin{bmatrix} I_{Hxx} & 0 & I_{Hxz} \\ 0 & I_{Hyy} & 0 \\ I_{Hxz} & 0 & I_{Hzz} \end{bmatrix}.$$

Представим линеаризованные уравнения движения:

$$P = \sum F_i \cdot \Delta v_i = T_\phi \Delta\dot{\phi} + T_\delta \Delta\dot{\delta} + T_{\theta_R} \Delta\dot{\theta}_R, \qquad (1)$$

где T_ϕ, T_δ, T_{θ_R} – обобщенные силы каждой линейной комбинации компонентов различных приложенных сил F.

Принятие полной линеаризации и движения без завала по прямой может быть достигнуто при любой скорости, удовлетворяющей условию $v = -\dot{\theta}_R r_R$. Боковая симметрия системы, совмещенная с линейностью уравнений, позволяет получить зависимость между прямолинейным движением и углами падения и поворота. Исходя из этого, линеаризованное уравнение движения для первой степени свободы (движения вперед) просто получить, исследуя систему по осям xz, как:

$$[r_R^2 m_T + I_{Ryy} + (r_R / r_F)^2 I_{Fyy}]\ddot{\theta}_R = T_{\theta_R}, \qquad (2)$$

Линеаризованные уравнения для двух других степеней свободы, угла падения и поворота рулевого колеса это пара совмещенных

дифференциальных уравнений второго порядка. Уравнение в канонической форме, где первое уравнение отражает крен, а второе поворот руля $\mathbf{M}\ddot{\mathbf{q}}+\mathbf{C}\dot{\mathbf{q}}+\mathbf{K}\mathbf{q}=\mathbf{f}$. Для модели мотоцикла запишем в форме $\mathbf{M}\ddot{\mathbf{q}}+v\mathbf{C}_1\dot{\mathbf{q}}+[g\mathbf{K}_0+v^2\mathbf{K}_2]\mathbf{q}=\mathbf{f}$, где изменяемые во времени переменные $\mathbf{q}=[\phi,\ \delta]^{\mathrm{T}}$ и $\mathbf{f}=[T_\phi,T_\delta]^{\mathrm{T}}$, $\mathbf{M}$ – симметричная матрица масс, которая отображает кинетическую энергию мотоцикла как. Матрица затухания $\mathbf{C}=v\mathbf{C}_1$ линейной скорости v и охватывает кососимметрические гироскопические моменты, возникающие из-за уровней крена мотоцикла и рулевого управления. C_1также содержит инерциальные реакции, возникающие во время рулевого управления. Матрица жесткости $\mathbf{K}$ состоит из двух частей: независимой от скорости симметричной части $g\mathbf{K}_0$ пропорциональной гравитационному ускорению, которая может быть использована при расчете изменения потенциальной энергии, и части $v^2\mathbf{K}_2$, которая пропорциональна квадрату скорости и зависит от гироскопического и центробежного эффектов. Таким образом, имеем характеристическое уравнение:

$$\det(\mathbf{M}\lambda^2+v\mathbf{C}_1\lambda+g\mathbf{K}_0+v^2\mathbf{K}_2)=0. \qquad (3)$$

Далее следует рассматривать решения (3) при различных скоростях. На низких и высоких скоростях асимптотическая устойчивость не достижима, в силу физических ограничений, наложенных на управление рулем. На начальных этапах рассматривается диапазон средних скоростей, где решается задача асимптотической устойчивости.

В общем виде модель движения выглядит следующим образом:

$$\dot{\mathbf{q}}=\mathbf{S}(\mathbf{q},t)\mathbf{u},\ \dot{\mathbf{u}}=[\mathbf{M}(\mathbf{q},t)]^{-1}\mathbf{Q}(\mathbf{q},u,t), \qquad (4)$$

где $\mathbf{q}$ – обобщенная координата, u – обобщенная скорость, $\mathbf{S}$ – кинематическая матрица, которая отображает отношение уровней обобщенных координат к обобщенным скоростям, $\mathbf{M}$ – система матриц масс, $\mathbf{Q}$ содержит все силы и зависимости от скорости инерции.

1.3.1.2 Нахождение параметров линеаризованной модели движения робота

Для описания введем следующие индексы: R – для заднего колеса, B – для задней части рамы совместно с мотоциклистом, H – для передней части с рулем, F – для переднего колеса, T – для системы в целом, A – для сборки передней части рамы с рулем.

Найдем общую массу и расположение центра масс (относительно точки контакта заднего колеса с землей P):

$$m_T=m_R+m_B+m_H+m_F,\ x_T=(x_Bm_B+x_Hm_H+wm_F)/m_T,$$

$z_T = (-r_R m_R + z_B m_B + z_H m_H - r_F m_F)/m_T$.

Для вывода момента инерции для всего мотоцикла вдоль оси z используем соответствующие моменты масс и значения инерций по отношению к контактной точке P вдоль общих осей, с учетом зависимости осесимметричных моментов инерции переднего и заднего колес:

$$I_{Tzz} = I_{Rzz} + I_{Bzz} + I_{Hzz} + I_{Fzz} + m_B x_B^2 + m_H x_H^2 + m_F w^2$$

Аналогичным образом описывается сборка рамы с рулем A. Рассмотрим вектор $\lambda = (\sin\lambda, 0, \cos\lambda)^{\mathrm{T}}$, направленный вниз вдоль рулевой оси, где λ – угол на плоскости xz. Центр масс передней сборки расположен выше руля, перпендикулярно $u_A = (x_A - w - c)\cos\lambda - z_A \sin\lambda$. Для сборки A требуются три специальных величины инерции: момент инерции относительно оси руля и значения инерции, относительно смещенных и косых осей, взятых около точки, где они пересекаются. Из этих соотношений берется момент около одной оси, возникающий из-за углового ускорения около других.

Отношение механического трейла (расстояние между осью передней вилки и нормалью из передней оси на проекции земли) к базе колес $\mu = (c/w)\cos\lambda$.

Угловые моменты заднего и переднего колес вдоль оси y, вычисленные с помощью линейной скорости, вместе с их суммой из гиростатических коэффициентов:

$$S_R = I_{Ryy}/r_R, S_F = I_{Fyy}/r_F, S_T = S_R + S_F.$$

Сформируем параметры линеаризованных уравнений движения.

$$\mathbf{M} = \begin{bmatrix} M_{\phi\phi} & M_{\phi\delta} \\ M_{\delta\phi} & M_{\delta\delta} \end{bmatrix},\ \mathbf{K}_0 = \begin{bmatrix} K_{0\phi\phi} & K_{0\phi\delta} \\ K_{0\delta\phi} & K_{0\delta\delta} \end{bmatrix},\ \mathbf{K}_2 = \begin{bmatrix} K_{2\phi\phi} & K_{2\phi\delta} \\ K_{2\delta\phi} & K_{2\delta\delta} \end{bmatrix},\ \mathbf{C}_1 = \begin{bmatrix} C_{1\phi\phi} & C_{1\phi\delta} \\ C_{1\delta\phi} & C_{1\delta\delta} \end{bmatrix}, \quad (5)$$

где $M_{\phi\phi} = I_{Txx}$, $M_{\phi\delta} = I_{A\lambda x} + \mu I_{Txz}$, $M_{\delta\phi} = M_{\phi\delta}$, $M_{\delta\delta} = I_{A\lambda\lambda} + 2\mu I_{A\lambda z} + \mu^2 I_{Tzz}$ – элементы симметричной матрицы масс $\mathbf{M}$; $K_{0\phi\phi} = m_T z_T$, $K_{0\phi\delta} = -S_A$, $K_{0\delta\phi} = K_{0\phi\delta}$, $K_{0\delta\delta} = -S_A \sin\lambda$ – гравитационные коэффициенты жесткости (умноженные на g); $K_{2\phi\phi} = 0$, $K_{2\delta\phi} = 0$, $K_{2\phi\delta} = ((S_T - m_T z_T)/w)\cos\lambda$, $K_{2\delta\delta} = ((S_A + S_F \sin\lambda)/w)\cos\lambda$ – скоростные коэффициенты жесткости (умноженные на v^2); $C_{1\phi\phi} = 0$, $C_{1\phi\delta} = \mu S_T + S_F \cos\lambda + (I_{Txz}/w)\cos\lambda - \mu m_T z_T$, $C_{1\delta\delta} = (I_{A\lambda z}/w)\cos\lambda + \mu(S_A + (I_{Txz}/w)\cos\lambda)$, $C_{1\delta\phi} = -(\mu S_T + S_F \cos\lambda)$ – коэффициенты матрицы «затухания» $\mathbf{C}$.

1.3.1.3 Расчет параметров регулятора

Запишем уравнения баланса угловых моментов. Баланс углового момента крена относительно продольной оси:

$$-m_T \ddot{y}_P z_T + I_{Txx}\ddot{\phi} + I_{Txz}\ddot{\psi} + I_{A\lambda x}\ddot{\delta} + \dot{\psi} v S_T + \dot{\delta} v S_F \cos\lambda = T_{B\phi} - g m_T z_T \phi + g S_A \delta.$$

Баланс углового момента рыскания относительно точки P:

$m_T \ddot{y}_P x_T + I_{Txz}\ddot{\phi} + I_{Tzz}\ddot{\psi} + I_{A\lambda z}\ddot{\delta} - \dot{\phi} v S_T - \dot{\delta} v S_F \sin\lambda = w F_{Fy}$.

Баланс углового момента руля для передней сборки:

$m_A \ddot{y}_P u_A + I_{A\lambda z}\ddot{\psi} + I_{A\lambda\lambda}\ddot{\delta} + v S_F(-\dot{\phi}\cos\lambda + \dot{\psi}\sin\lambda) = T_{H\delta} - c F_{Fy}\cos\lambda + g(\phi + \delta\sin\lambda) S_A$.

Используя выражение, характеризующее изменение угла крена относительно изменения угла рыскания рамы получим итоговое выражение баланса мотоцикла:

$\ddot{y}_P = ((v^2\delta + vc\dot{\delta})/w)\cos\lambda$

Уравнения задания положения мотоцикла относительно задней контактной точки:

$\dot{x}_P = v\cos\psi$, $\dot{y}_P = v\sin\psi$.

Для модели вход-состояние-выход, где $\mathbf{x} = [\dot{\phi}, \dot{\delta}, \phi, \delta]^{\mathrm{T}}$ – вектор состояния, найдем закон управления $u = T_\delta$. Расчет регулятора для случая адаптивной системы будем проводить с помощью алгоритма «полоска».[27] Для неадаптивной системы воспользуемся линейно-квадратичным регулятором с критерием качества $J = \int_0^\infty (x^{\mathrm{T}}\mathbf{Q}_u x + u^{\mathrm{T}}\mathbf{R}_u u)dt$, где $\mathbf{Q}_u$ и $\mathbf{R}_u$ – диагональные матрицы штрафов по состоянию и управлению. Выберем $\mathbf{Q}_u = diag\{0,0,1,0\}$, так как итоговая цель балансирования мотоцикла, удержание вертикального положения, может быть достигнута минимизацией угла крена. Для контроля усилий возьмем $\mathbf{R}_u = 1$.

Алгоритм управления рулевым колесом по измерению угла крена выберем в виде ПИД-регулятора (пропорционально-интегрально-дифференциальный), коэффициенты которого подберем методом Циглера-Никольса $k_p = 8{,}23$ – для пропорционального, $k_i = 1{,}48$ – для интегральный и $k_d = 2{,}76$ для дифференциального составляющих сигнала управления.

1.3.1.4 Физические параметры модели

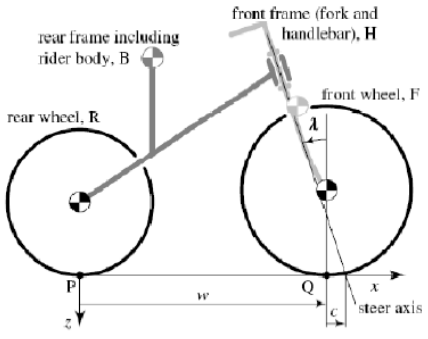


Рисунок 1.38 – Схематический вид мотоцикла

Для этой модели параметры приведены в таблице 1.

Таблица 1.13 Параметры модели

Параметр	Единица измерения	Описание
g=9.81	м/с^2	Гравитационная постоянная
m=0.03	Кг	Масса колеса
R=0.04	М	Радиус колеса
M=0.678	Кг	Общая масса модели
M_front=0.2	Кг	Масса передней части рамы
M_rear=0.478	Кг	Масса задней части рамы
L_base =0.225	М	Расстояние между точками соприкосновения колес с поверхностью (w)
L_trail = 0.011	М	Треил, расстояние между осью передней вилки и нормалью из передней оси на проекцию земли (c)
Head_angle =67	Град	Угол наклона рулевой вилки относительно перпендикуляра к поверхности
V=0.6	м/с	Начальная скорость мотоцикла

1.3.1.5 Описание программы Motobot'a

Главная управляющая схема (рисунок 1.39). Схема, приведенная ниже, отражает главное окно программы. В левой части расположены входные блоки, в правой - выходные. Это довольно общая схема, которая состоит из k-подсистем, отвечающих за различные функции. Непосредственно на схеме расположены такие блоки, как: уровень заряда батарей, системные часы, протоколы передачи и приема сигналов через Bluetooth, звуковые генераторы, гироскопический датчик, ультразвуковой датчик расстояния, энкодеры сервоприводов и сами сервопривода, а также модуль согласования задач, обобщенный блок основной программы и кнопки интерфейса для пользователя. Остальные схемы содержатся в блоке nxtway_app.

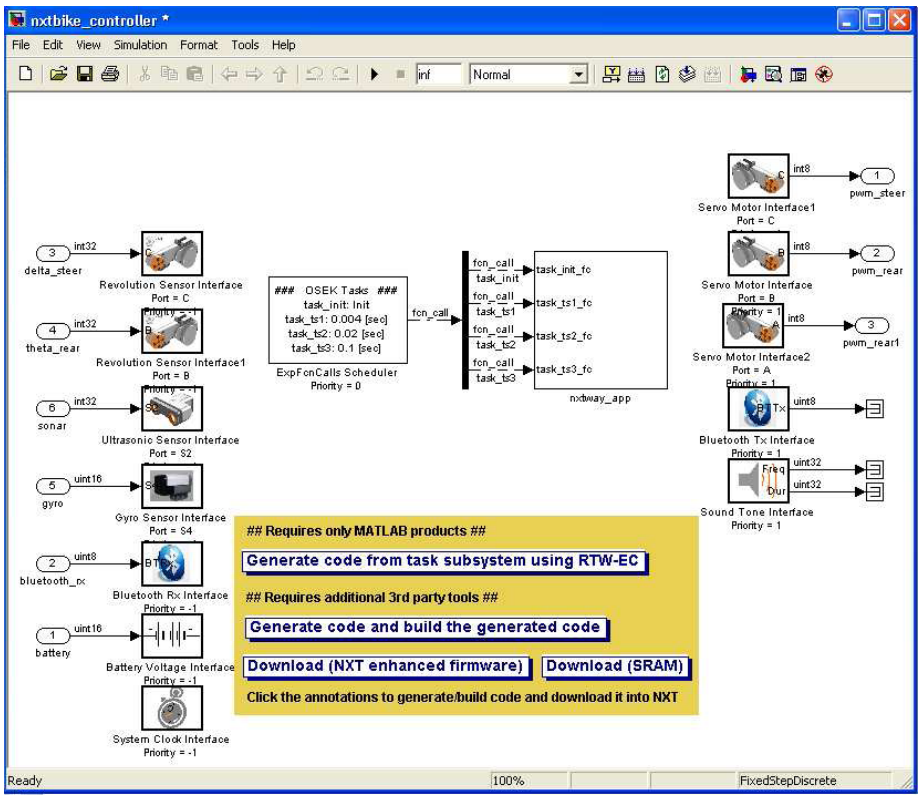


Рисунок 1.39 - Главная управляющая схема

На рисунке 1.40 изображена общая схема контроллера со всеми входами/выходами. Также показаны сигналы, передающие данные на запись, которая ведется через протокол Bluetooth. Сервопривода A и B принимают одинаковые управляющие сигналы, так как оба являются приводами ведущего колеса и расположены с двух сторон колеса для симметрии масс модели.

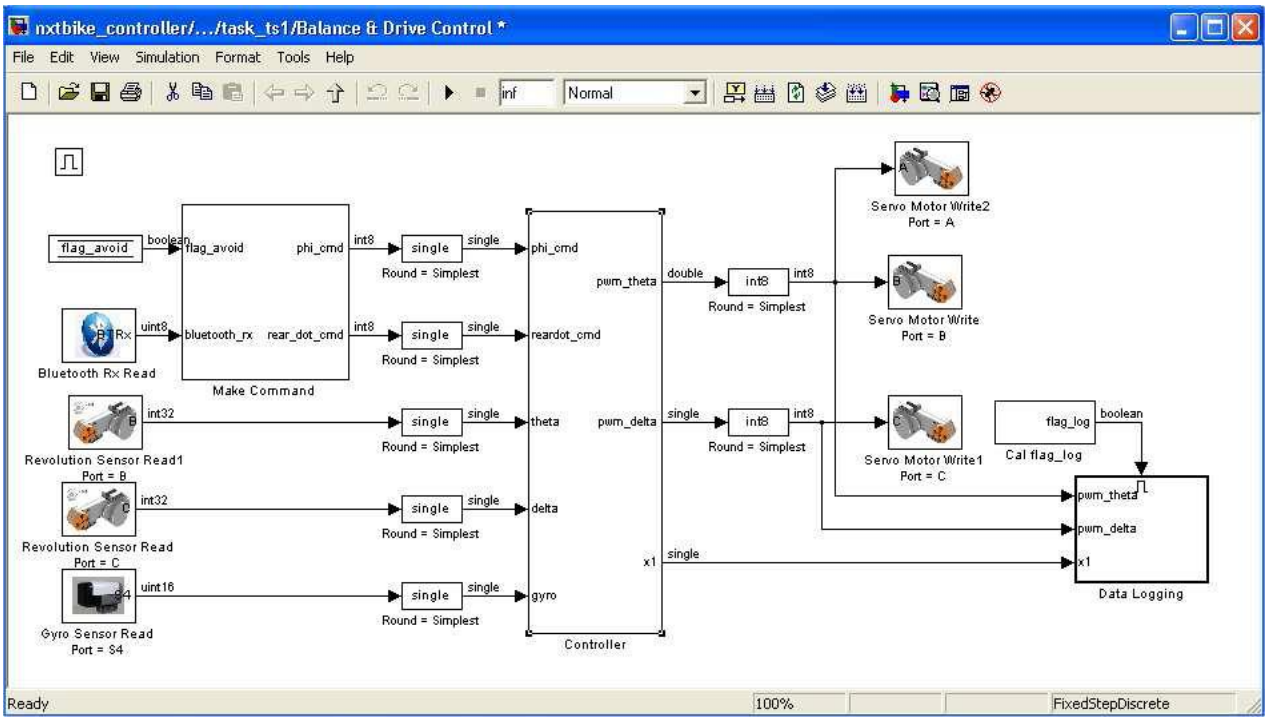


Рисунок 1.40 - Общая схема контроллера

На рисунке 1.41 представлена схема управления. Открыть ее можно через:

nxtway_app/Balance & Drive control/Balance & Drive Control/Controller

Верхний усилитель k_thetadot используется в качестве коэффициента П-регулятора сигнала thetadot_ref, для регулирования скорости заднего колеса. Второй усилитель умножается на ошибку сигнала (x1_ref-x1), где x1=[phi phidot phidotdot] – вектор, отражающий угол наклона рамы motobot'a и две его производных.

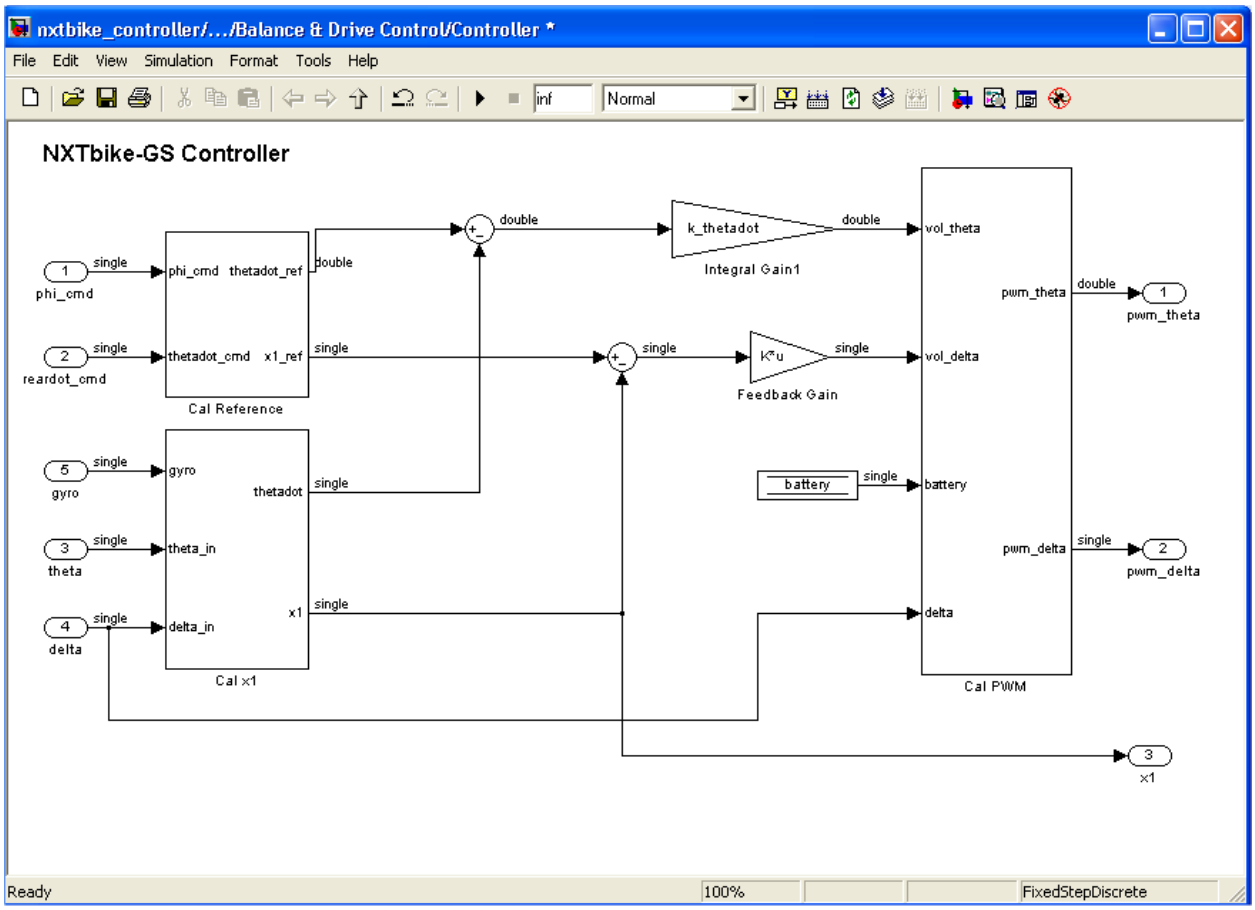


Рисунок 1.41 - Схема управления

Управление положением модели в пространстве. На следующем рисунке phi_cmd – задаваемый угол положения рамы motobot'a, изначально равный 0. Как правило, когда этот угол изменяется (например после подачи управляющих сигналов с джойстика) появляется возможность рулевого управления для стабилизации.

Также здесь определен параметр thetadot_cmd. Набор постоянных значений для данной скорости изменения угла доступен благодаря переключателю. Скорость может быть изменена с помощью управления через джойстик по Blutooth. [15]

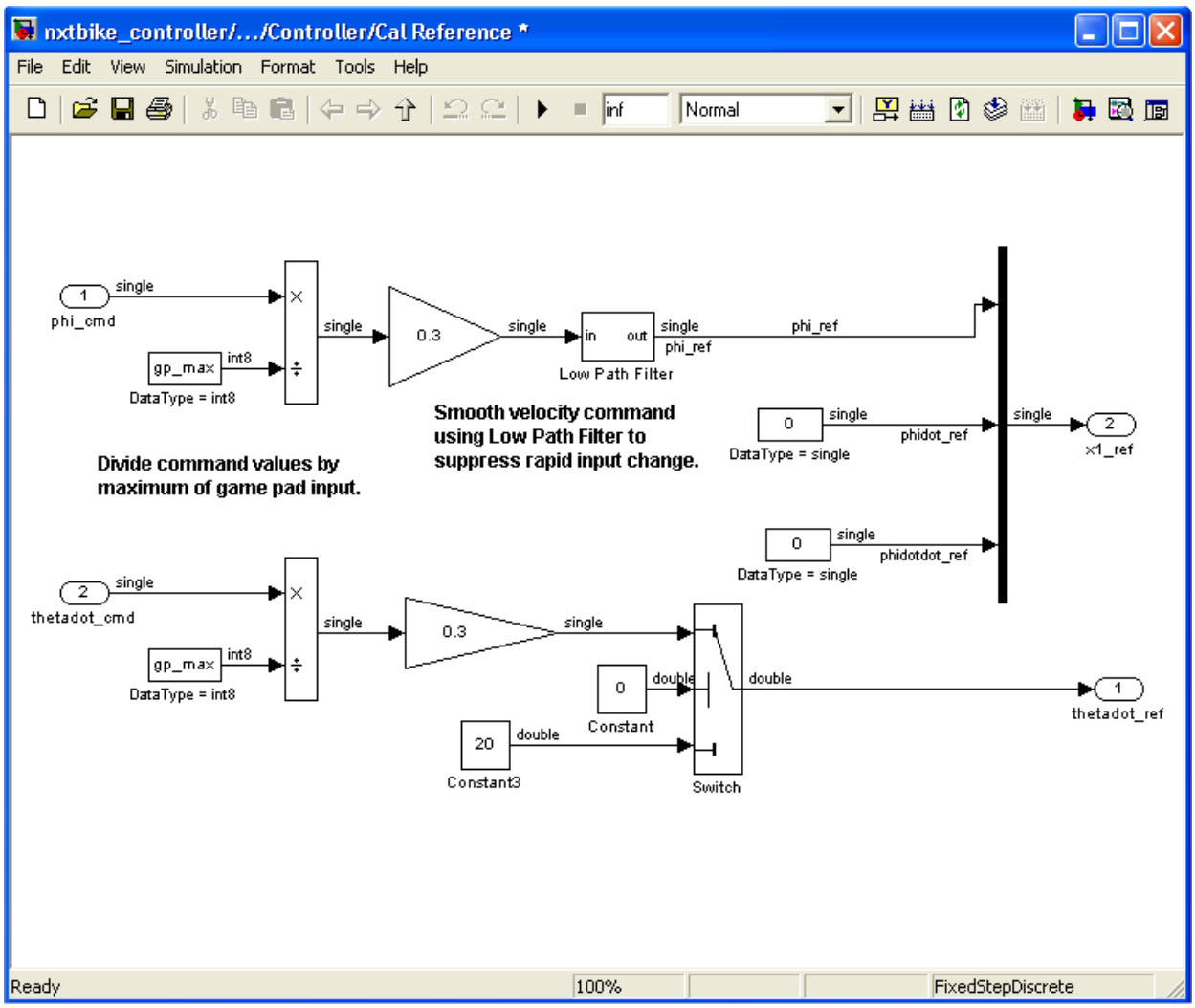


Рисунок 1.42 - Схема управления положением модели в пространстве

Вычисление входных сигналов. Как представлено на следующей схеме (рисунок 1.43), theta_in (значение положения заднего колеса) конвертируется в радианы и дифференцируется для получения скорости thetadot (рад/с). Производная для получения скорости вычисляется с периодом 0.004 мс.

Данные с гироскопа конвертируются и затем трансформируются в массив x1, который получается путем дифференцирования в phi_dotdot и интеграции в phi. Значения обновляются с периодом 0.004 мс.

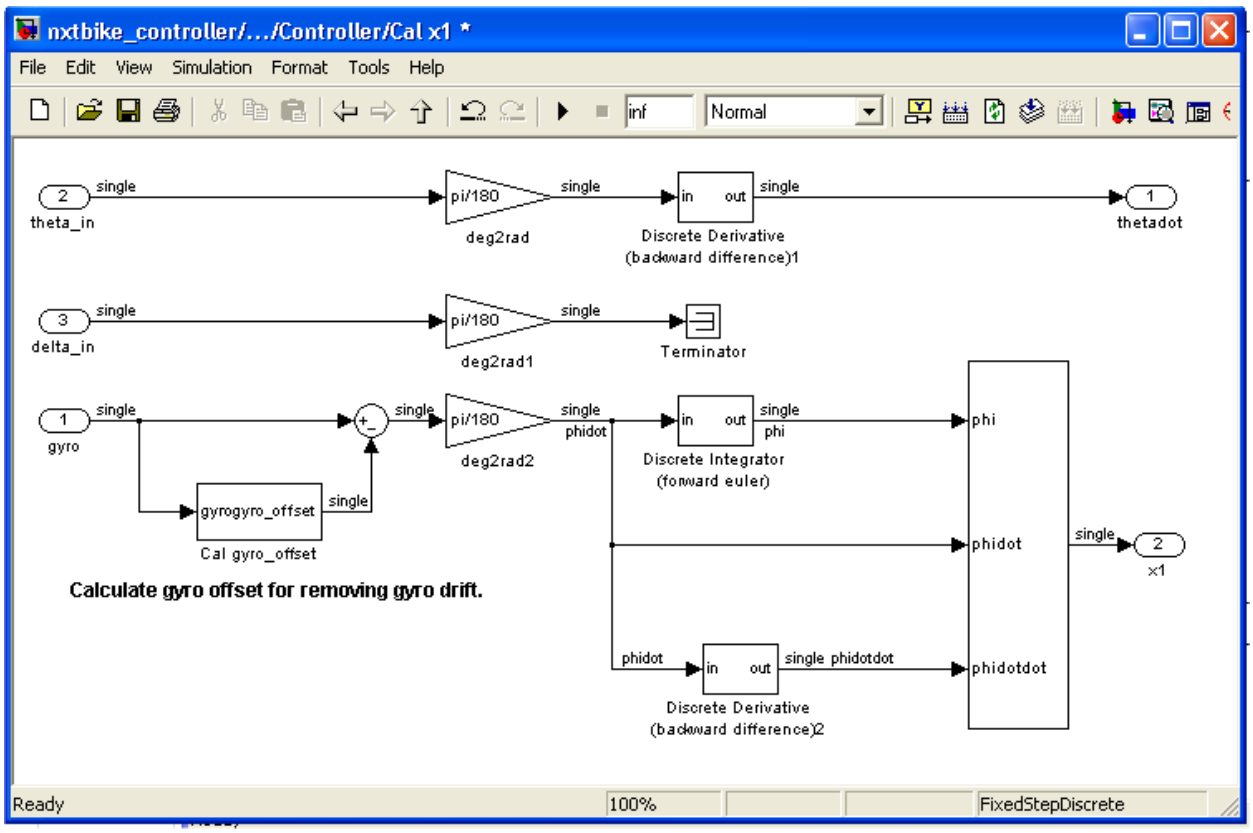


Рисунок 1.43 – Схема вычисления входных сигналов

Вычисление сигналов, поступающих на сервоприводы. Далее показано как сигналы для сервоприводов корректируются с текущим значением заряда аккумулятора (рисунок 1.44). Также на схеме присутствуют компенсатор колебаний и ограничитель сигналов.

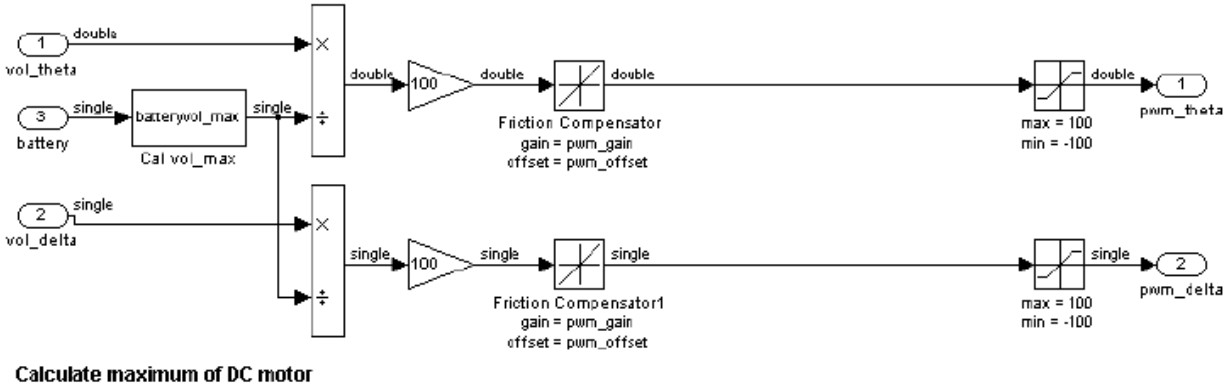


Рисунок 1.44 – Схема вычисление сигналов, поступающих на сервоприводы

1.3.1.6 Результаты эксперимента

Используя утилиту NXT GamePad можно получать и записывать значение угла крена мотоцикла с гироскопа и угол поворота рулевого колеса с энкодера серводвигателя. На рисунке 1.45 представлены графики данных, полученных при тестировании модели.

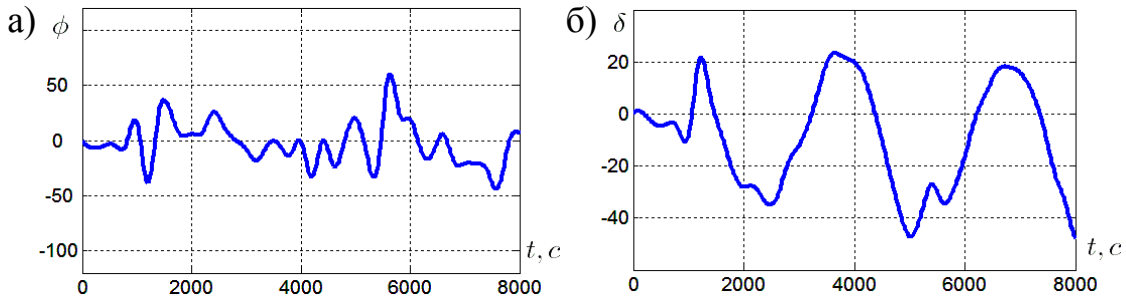


Рисунок 1.45 – Временные диаграммы процессов в роботе «Мотобот»: а) угол крена мотоцикла ϕ [град./сек.]; б) угол поворота рулевого колеса δ [град.]

1.3.2 Модель робота «Segway»

1.3.2.1Расчет математической модели

Балансирующий робот «Segway» (рисунок 1.46) представляет собой однозвенный перевернутый маятник. Стабилизация его верхнего неустойчивого положения равновесия относится к числу классических задач механики и теории управления. Задача может быть решена путем горизонтального перемещения точки подвеса маятника. Перевернутый маятник - это модель системы, управляющей положением корпуса ракеты на активном участке полета.

Рисунок 1.46 - Балансирующий робот «Segway»

Математическая модель перевернутого маятника является нелинейной и включает несколько важных параметров таких, как масса маятника,

коэффициент трения, расстояние между осью маятника и его центром масс, момент инерции маятника относительно центра масс.

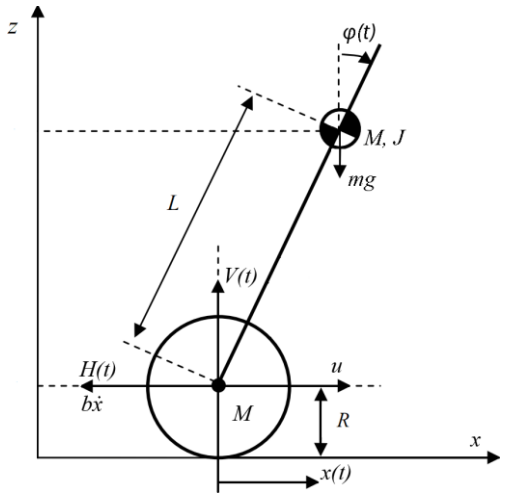


Рисунок 1.47 – Схема сил, действующих на балансирующий робот.

M – масса основания;

m – масса маятника;

b – коэффициент трения при движении;

L – расстояние между осью маятника и центром его масс;

J – момент инерции маятника относительно центра масс;

u(t) – управление, прикладываемое к основанию;

x(t) – координата основания;

φ(t) – угол отклонения маятника от вертикали;

H(t) – горизонтальная сила реакции на оси маятника;

V(t) – вертикальная сила реакции на оси маятника;

g – ускорение свободного падения.

Для представленной системы справедливы следующие уравнения:

$$m\frac{d^2}{dt^2}\left[x(t)-L\sin\varphi(t)\right]=H(t) \tag{6}$$

$$m\frac{d^2}{dt^2}\left[L\cos\varphi(t)\right]=V(t)-mg \tag{7}$$

$$J\frac{d^2\varphi(t)}{dt^2}=L\cdot H(t)\cdot\cos\varphi(t)+L\cdot V(t)\cdot\sin\varphi(t) \tag{8}$$

$$M\frac{d^2x(t)}{dt^2}=u(t)-H(t)-b\frac{dx(t)}{dt} \tag{9}$$

Раскрывая скобки в уравнениях (6) и (7), получим:

$$m\ddot{x} + mL\dot{\varphi}^2 \sin\varphi - mL\ddot{\varphi}\cos\varphi = H(t) \quad (10)$$

$$-mL\dot{\varphi}^2 \cos\varphi - mL\ddot{\varphi}\sin\varphi = V(t) - mg \quad (11)$$

Подставив выражение для H(t) из уравнения (10) в уравнение (9), получим:

$$(M+m)\ddot{x} + b\dot{x} - mL\ddot{\varphi}\cos\varphi + mL\dot{\varphi}^2 \sin\varphi = u(t) \quad (12)$$

Подставив выражения для H(t) из уравнения (10) и для V(t) из уравнения (11) в уравнение (8), получим:

$$\left(J + mL^2\right)\ddot{\varphi} - mL\ddot{x}\cos\varphi - mgL\sin\varphi = 0 \quad (13)$$

Уравнения (12) и (13) являются нелинейными уравнения, составляющими математическую модель перевернутого маятника с подвижной точкой опоры.

Для решения задачи классическими методами теории автоматического управления необходимо линеаризовать уравнения (12) и (13) и записать их в форме уравнений в переменных состояния.

Линеаризацию в окрестностях $\varphi(t) \equiv 0$, $x(t) \equiv 0$, $u(t) \equiv 0$ легко выполнить, разлагая $\sin\varphi(t)$ и $\cos\varphi(t)$ в ряды Тейлора и подставляя в уравнения (12) и (13) только первые члены рядов:

$$(M+m)\ddot{x} + b\dot{x} - mL\ddot{\varphi} = u(t) \quad (14)$$

$$\left(J + mL^2\right)\ddot{\varphi} - mL\ddot{x} - mgL\varphi = 0 \quad (15)$$

Основываясь на линеаризованных уравнениях системы найдем ее передаточную функцию, полагая, что входом является управляющие воздействие u(t), а выходом – угол отклонения маятника от вертикали φ(t). Для этого применим к линеаризованным уравнениям (12) и (13) преобразование Лапласа и обозначив Φ(s) и U(s) изображения по Лапласу функций φ(t) и u(t) соответственно получим:

$$\Phi(s)\left[(M+m)\left(\frac{J+mL^2}{mL} - \frac{g}{s^2}\right)s^2 + b\left(\frac{J+mL^2}{mL} - \frac{g}{s^2}\right)s - mLs^2\right] = U(s) \quad (16)$$

И искомая передаточная функция примет вид:

$$W_p(s)=\frac{\Phi(s)}{U(s)}=\frac{mLs}{\left((M+m)\left(J+mL^2\right)-m^2L^2\right)s^3+b\left(J+mL^2\right)s^2-(M+m)mgLs-bmgL} \quad (17)$$

Управление подвижным основанием реализуется через электродвигатель, упрощенная модель которого представляет собой апериодическое звено первого порядка:

Теперь запишем уравнения данной системы в переменных состояния. Введем следующие переменные состояния системы:

$$x_1(t)=x(t),\ \ x_2(t)=\dot{x}_1(t)=\dot{x}(t),\ \ x_4(t)=\varphi(t),\ \ x_4(t)=\dot{x}_3(t)=\dot{\varphi}(t). \quad (18)$$

Тогда уравнения (12) и (13) запишутся в виде:

$$(M+m)\dot{x}_2+bx_2-mL\dot{x}_4-u(t)=0, \quad (19)$$

$$\left(J+mL^2\right)\dot{x}_4-mL\dot{x}_2-mgLx_3=0. \quad (20)$$

Преобразуя данные уравнения можно получить:

$$\dot{x}_2=-\frac{b\left(J+mL^2\right)}{J(M+m)+MmL^2}x_2+\frac{m^2gL^2(M+m)}{J(M+m)+MmL^2}x_3+\frac{J+mL^2}{J(M+m)+MmL^2}u(t)=0, \quad (21)$$

$$\dot{x}_4=-\frac{bmL}{J(M+m)+MmL^2}x_2+\frac{mgL(M+m)}{J(M+m)+MmL^2}x_3+\frac{mL}{J(M+m)+MmL^2}u(t)=0. \quad (22)$$

Окончательно, в матричной форме уравнение можно записать:

$$\dot{x}=Ax+Bu(t), \quad (23)$$

где

$$x=\begin{bmatrix}x_1(t)\\x_2(t)\\x_3(t)\\x_4(t)\end{bmatrix},\ \ A=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & -\frac{b\left(J+mL^2\right)}{J(M+m)+MmL^2} & \frac{m^2gL^2(M+m)}{J(M+m)+MmL^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & -\frac{bmL}{J(M+m)+MmL^2} & \frac{mgL(M+m)}{J(M+m)+MmL^2} & 0\end{bmatrix},$$

$$B=\begin{bmatrix}0\\ \frac{J+mL^2}{J(M+m)+MmL^2}\\ 0\\ \frac{mL}{J(M+m)+MmL^2}\end{bmatrix}.$$

Уравнение выхода запишется (при искомых $\varphi(t)$ и $x(t)$) в следующем виде:

$$y=Cx+Du(t), \quad (24)$$

где $y = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, $D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

1.3.2.2 Расчет регулятора

1. Устойчивость

Будем называть систему асимптотически устойчивой, если ее установившееся значение будет стремиться к нулю независимо от начальных условий, при отсутствии входных воздействий:

$\lim_{t\to\infty} x(t) = 0$

Состояние объекта описывается следующим уравнением

$$\dot{x}(t) = Ax(t) + Bu(t)$$

Для того, чтобы система была асимптотически устойчива необходимо и достаточно, чтобы действительные части всех собственных чисел матрицы А были отрицательны.

2.Управление состоянием объекта по обратной связи

Управляющее воздействие по обратной связи представляет собой произведение коэффициента пропорциональности К и разницы между желаемым значением x_{ref} и измеренным x.

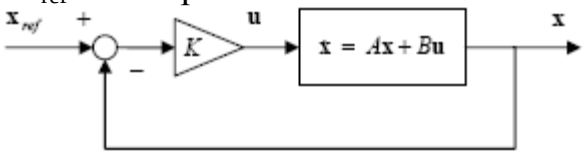


Рисунок 1.48 – Управление по обратной связи

Управляющее воздействие и состояние системы, изображенной на рисунке 1.48, описывается следующими уравнениями:

$u(t) = -K\big(x(t) - x_{ref}\big)$

$\dot{x}(t) = (A - BK)x(t) + BKx_{ref}$

Далее сделаем систему устойчивой, рассчитав значения матрицы К, так как изменение значений влечет за собой изменение собственных чисел матрицы А-ВК.

Для регулирования системы необходимо, чтобы она была управляемой.

Система управляема, если ранг матрицы М совпадает с рангом матрицы А, где: $M_c = [B, AB, \dots, A^{n-1}B]$

В среде Control System Toolbox для определения управляемости можно использовать команду ctrb.

Существует два основных метода расчета коэффициентов обратной связи.

1. Расчет по желаемым корням.

Метод заключается в расчете коэффициентов К на основании желаемых собственных чисел матрицы А-ВК. Выбор желаемых чисел осуществляется подбором. Для расчета в Control System Toolbox для расчета можно воспользоваться функцией place.

Например, рассчитайте коэффициенты обратно связи для системы, с матрицами A=[0, 1; -2; -3], B=[0;1]
Желаемые собственные числа: [-5, -6]

```
>> A = [0, 1; -2, -3]; B = [0; 1];
>> poles = [-5, -6];
>> K = place(A, B, poles)
K =
   28.0000    8.0000
```

2. Линейный квадратичный регулятор

Метод заключается в расчете коэффициентов матрицы K на основании минимизации значения функционала J, рассчитываемого следующим образом:

$$J = \int_0^\infty (x\,(t)^T Q x(t) + u(t)^T R u(t))dt$$

Настройка параметров обеспечивается подбором весовых матриц состояния Q и входного воздействия R, которые выбираются исходя из физической природы процессов. В Control System Toolbox для расчета регулятора используется функции lqr.
Например:
A=[0, 1; -2, -3], B=[0; 1], Q=[100, 0; 0, 1]; R=1

```
>> A = [0, 1; -2, -3]; B = [0; 1];
>> Q = [100, 0; 0, 1]; R = 1;
>> K = lqr(A, B, Q, R)
K =
   8.1980    2.1377
```

1.3.2.3 Управление сервоприводами

Управление сервоприводами необходимо для того, чтобы вырабатывать желаемое механическое движение системы. Для этой цели чаще всего используется ПИД-регулятор - устройство в цепи обратной связи, используемое в системах автоматического управления для формирования управляющего сигнала. ПИД-регулятор формирует управляющий сигнал, являющийся суммой трёх слагаемых, первое из которых пропорционально входному сигналу, второе — интеграл входного сигнала, третье — производная входного сигнала.

Назначение ПИД-регулятора — в поддержании заданного значения x0 некоторой величины x с помощью изменения другой величины u. Значение x0 называется уставкой, а разность e = (x_{ref} − x) — невязкой или рассогласованием.

Выходной сигнал регулятора u определяется тремя слагаемыми:

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

где Kp, Ki, Kd — коэффициенты усиления пропорциональной, интегральной и дифференциальной составляющих регулятора, соответственно.

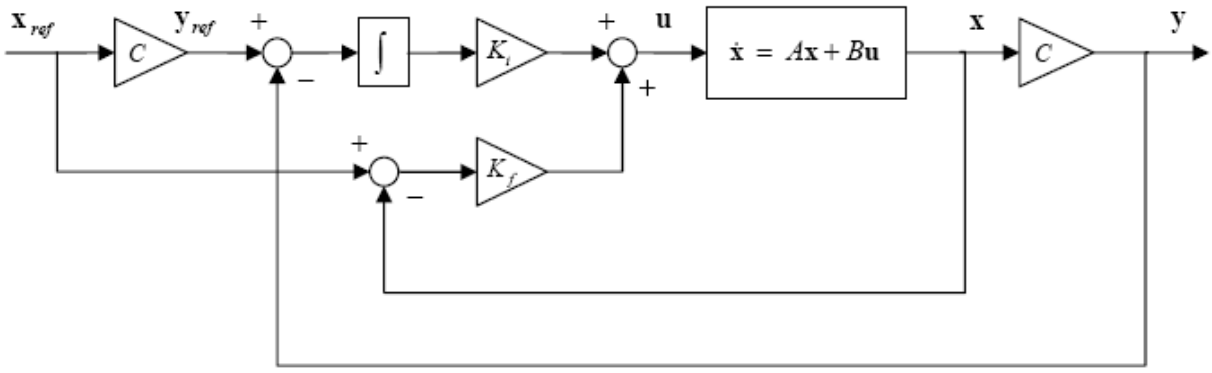


Рисунок 1.49 – ПИД-регулятор

1.3.2.4 Физические параметры модели

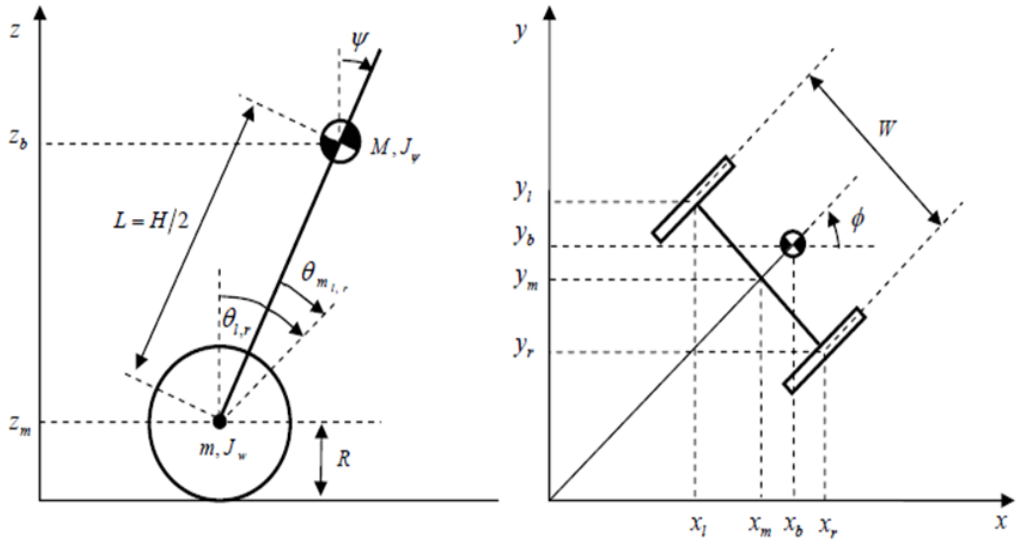


Рисунок 1.50 – Обозначение параметров модели

Таблица 1.14 Описание параметров модели

g = 9.81 (м/сек^2)	ускорение свободного падения
m = 0.03 (кг)	масса колеса
R = 0.035 (м)	радиус колеса
J_w = mR^2/2 (кгм^2)	момент инерции колеса
M = 0.6 (кг)	масса тела
W = 0.14 (м)	ширина тела
D = 0.04 (м)	толщина тела
H = 0.144 (м)	высота тела
L = H/2 (м)	расстояние до центра масс от оси колес

$J_\psi = ML^2/3$ (кгм^2)	момент инерции тела под наклоном
$J_\varphi = M(W^2+D^2)/12$ (кгм^2)	вращательный момент инерции
$J_m = 10^{-5}$ (кгм^2)	момент инерции двигателя
$K_b = 0.468$ (Всек/рад)	постоянная противо-ЭДС
$K_t = 0.317$ (Нм/А)	постоянная момента двигателя

1.3.2.5 Описание программы Segway

Главная управляющая схема. Схема, приведенная ниже, отражает главное окно программы. Ключевые блоки программы описаны в предыдущем разделе, при рассмотрении робота Motobot. Остальные схемы содержатся в блоке nxtway_app.

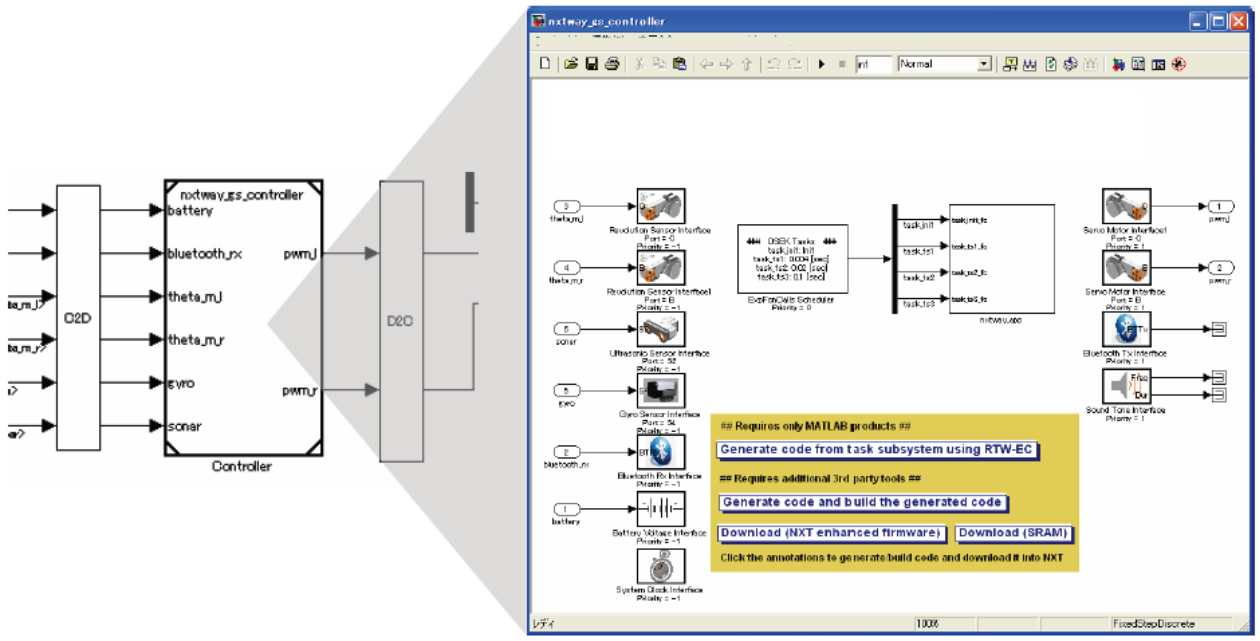


Рисунок 1.51 – Блок контроллера

На следующем рисунке изображена общая схема модели со всеми входами/выходами. Также показаны сигналы, передающие данные на запись, которая ведется через протокол Bluetooth.

Рассмотрим детально некоторые из составных блоков.

Reference Generator – это блок, передающий и ограничивающий управляющие воздействия на контроллер.

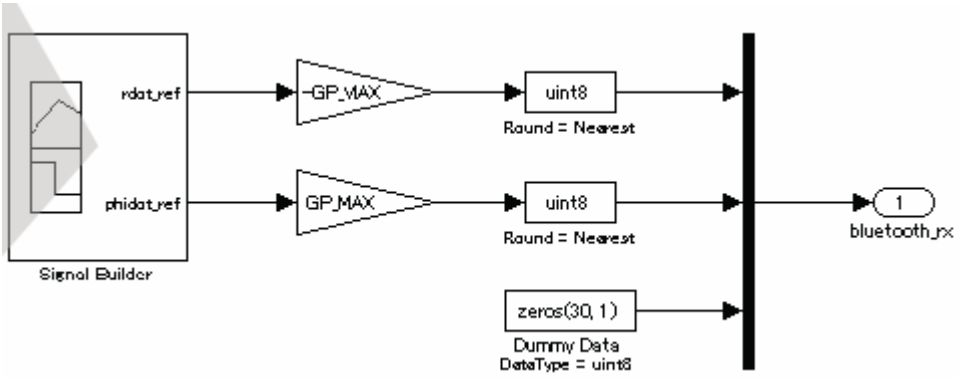


Рисунок 1.52 – Схема генератора управляющих воздействий

На рисунке 1.51 «Controller» – это блок, обозначающий контроллер NXT. С левой стороны расположены блоки, служащие для приема данных с датчиков, энкодеров, Bluetooth, с правой стороны расположены блоки, служащие для передачи сигналов на двигатели и Bluetooth

Блок контроллера работает в системе дискретного времени, а блок модели в режиме непрерывного времени, поэтому необходимо преобразование передаваемых величин.

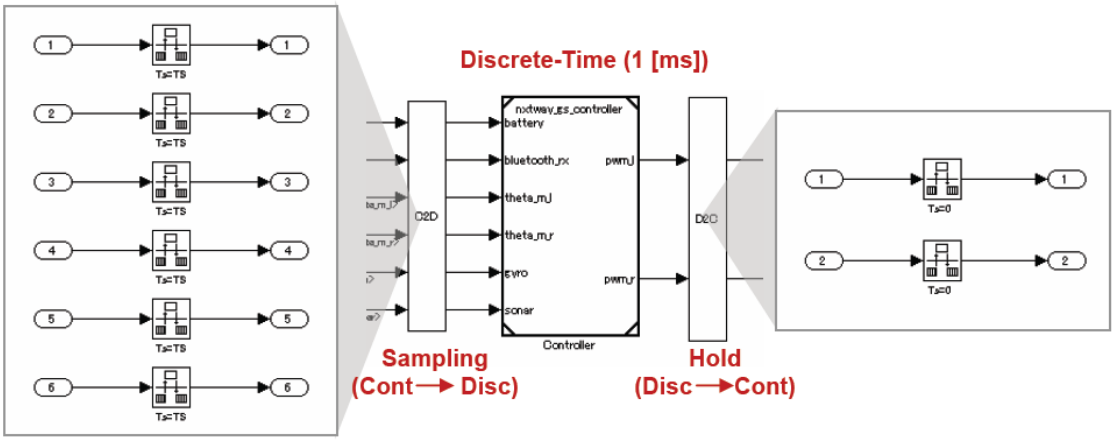


Рисунок 1.53 – Схема АЦ и ЦА преобразования сигналов

На рисунке 1.54 «NXTway-GS» - подсистема, состоящая из датчиков, приводов, и линеаризованной модели робота. Она преобразует тип входных данных сигналов в вещественный, рассчитывает динамику робота, и выводит результаты после дискретизации. Эта подсистема определяет параметры среды.

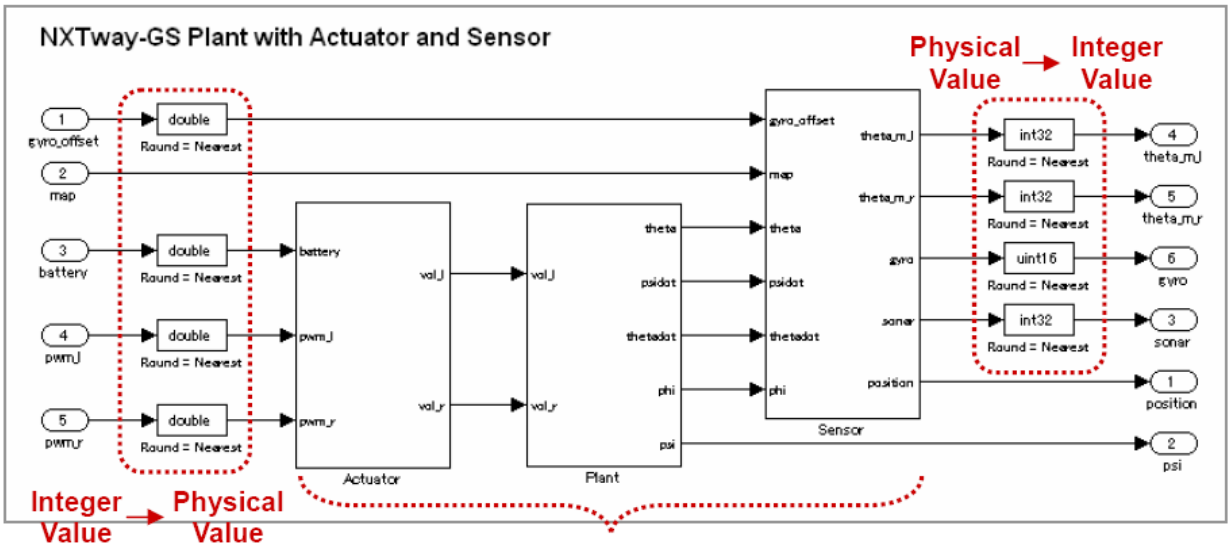


Рисунок 1.54 – Схема управления сервоприводами в зависимости от показаний датчиков

На рисунке 1.54 «Actuator»– это подсистема, преобразующая мощность задаваемую контроллером в напряжение, подаваемое на двигатели.

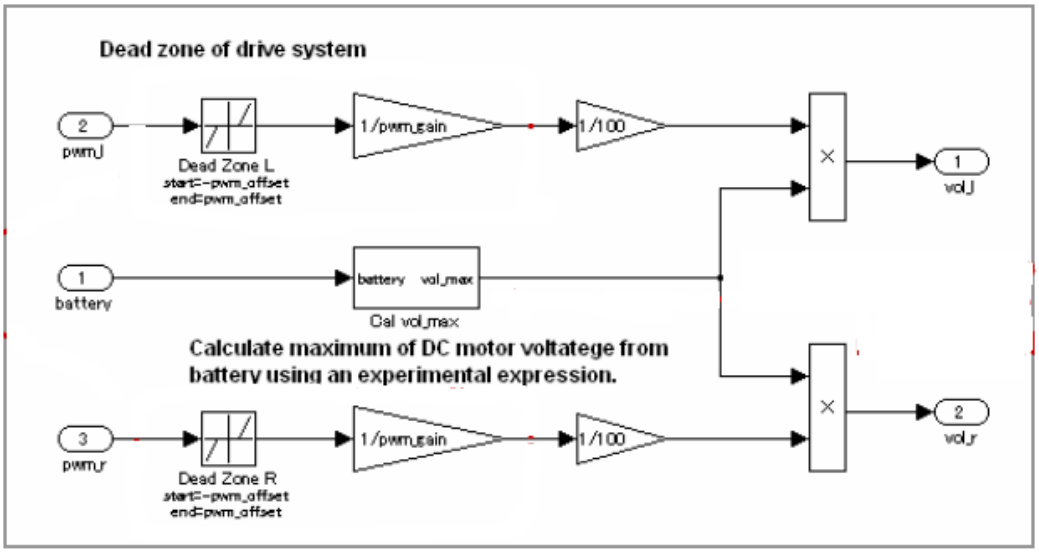


Рисунок 1.55 – Подсистема приводов

«Plant» - модель, описываемая уравнениями двойного перевернутого маятника, которая работает с учетом калибровки гироскопа.

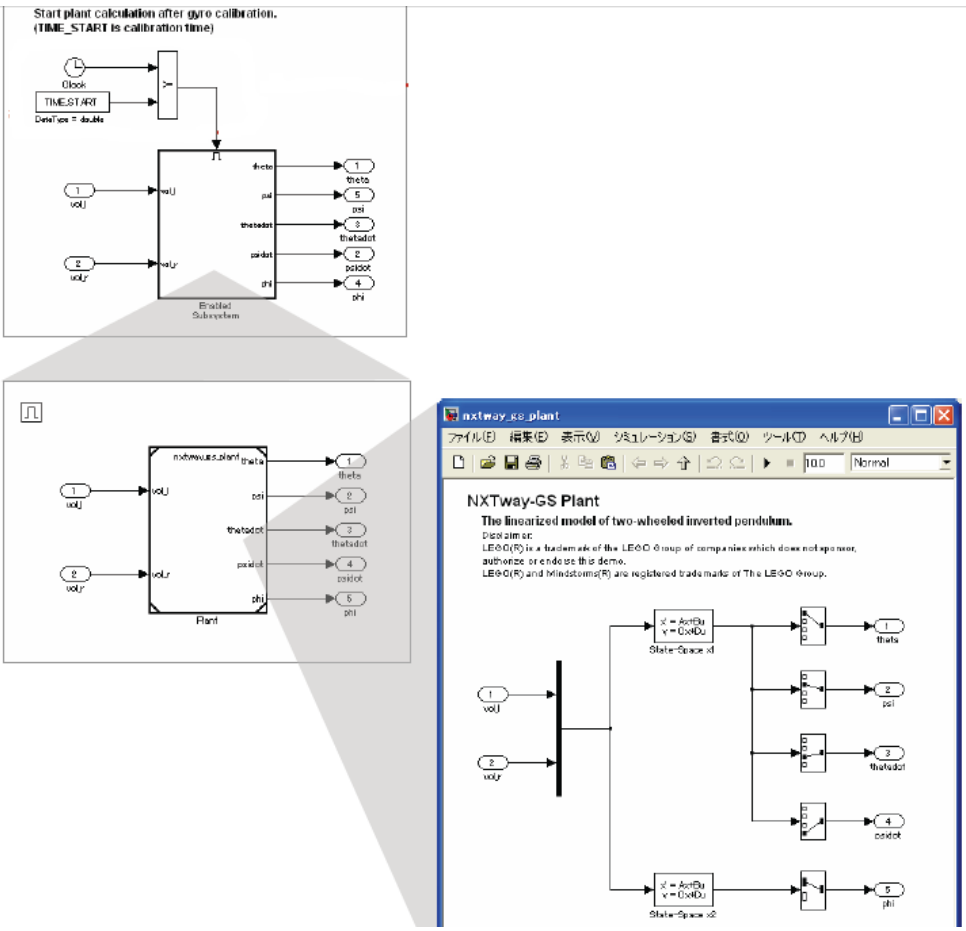


Рисунок 1.56 – Подсистема состояния модели

Блок Sensor преобразует значения, полученные о состоянии модели в выходные сигналы датчиков. Дополнительно вычисляется расстояние до препятствий, получаемое с ультразвукового датчика расстояния. Информация может быть использована для определения препятствий и избегания столкновений с ними.

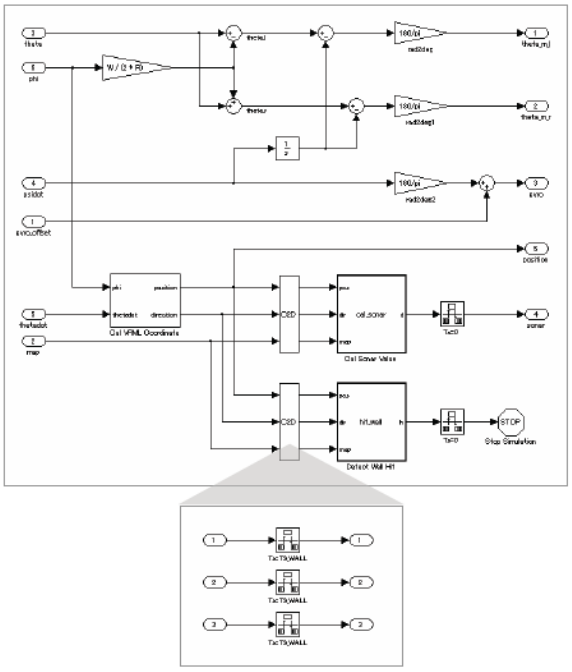


Рисунок 1.57 – Подсистема датчиков

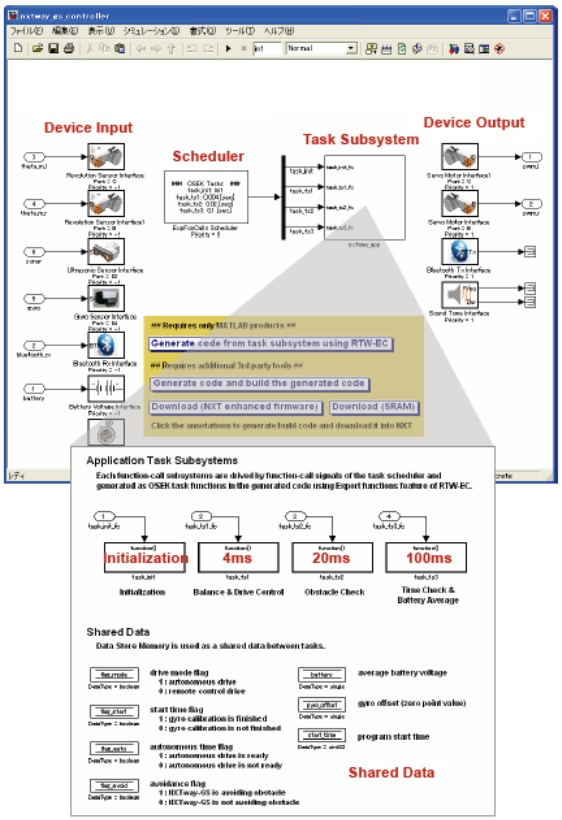


Рисунок 1.58 - nxtway_gs_controller.mdl

Блоки Data Store Memory используются, как общие блоки данных для распределения данных между подзадачами.

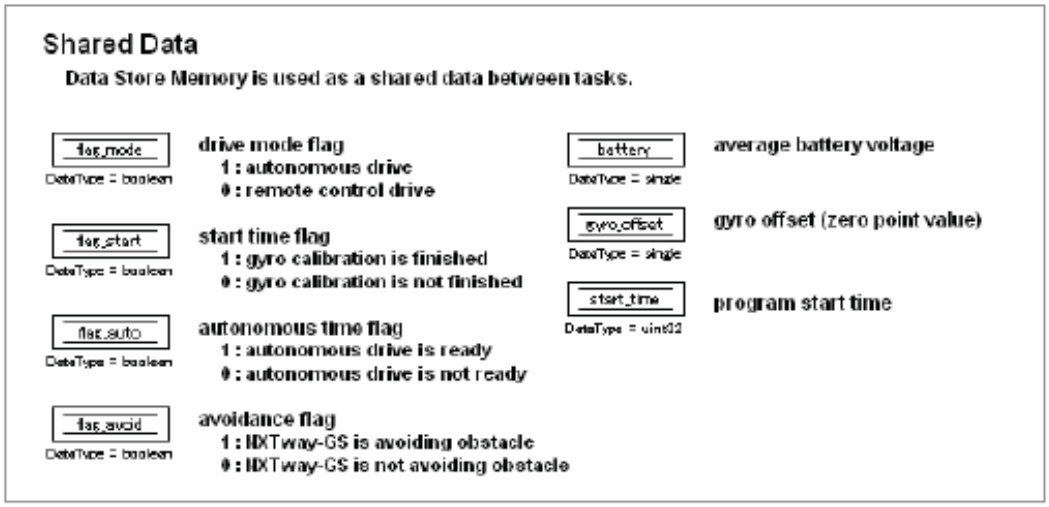


Рисунок 1.59 – Распределенные блоки данных

task_ts2: отвечает за обнаружение и уклонение от препятствий.

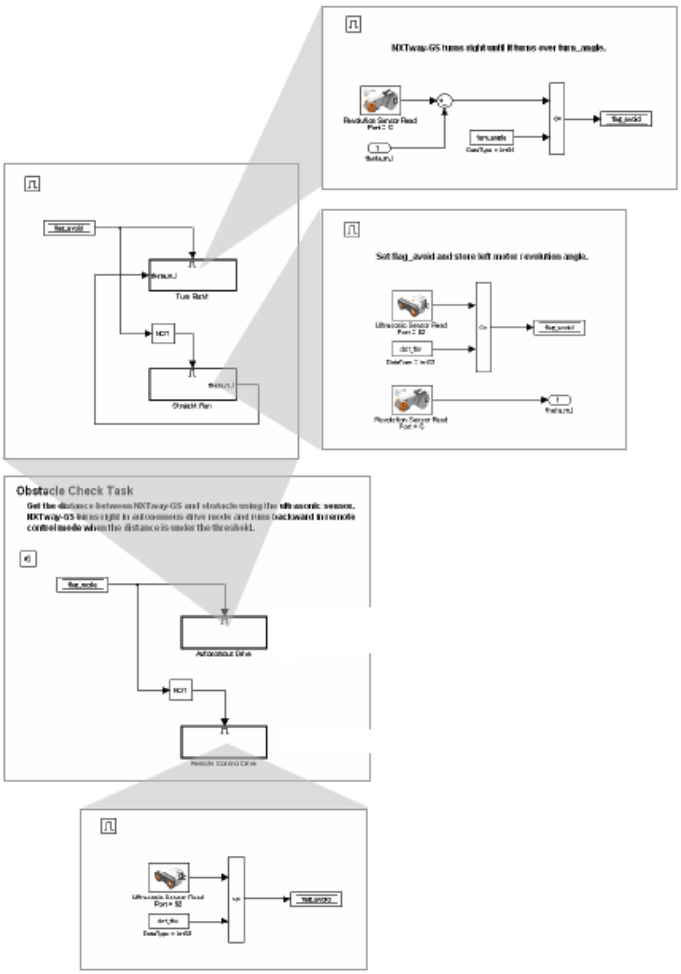


Рисунок 1.60 – Подсистема *task_ts2*

task_ts3: отвечает за подсчет времени и проверку уровня заряда батареи.

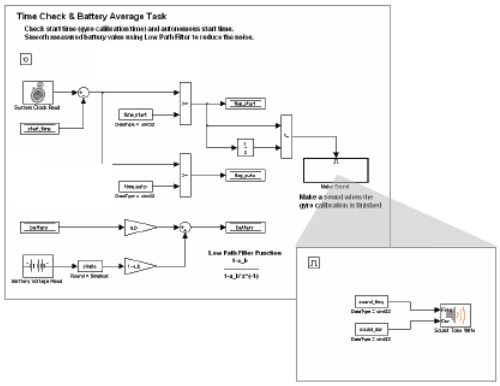


Рисунок 1.61 - Подсистема *task_ts3*

task_ts1: служит для калибровки гироскопа, балансировки; управления и записи данных. Балансировка и управление запускаются после калибровки гироскопа. Время калибровки сохраняется как time_start.

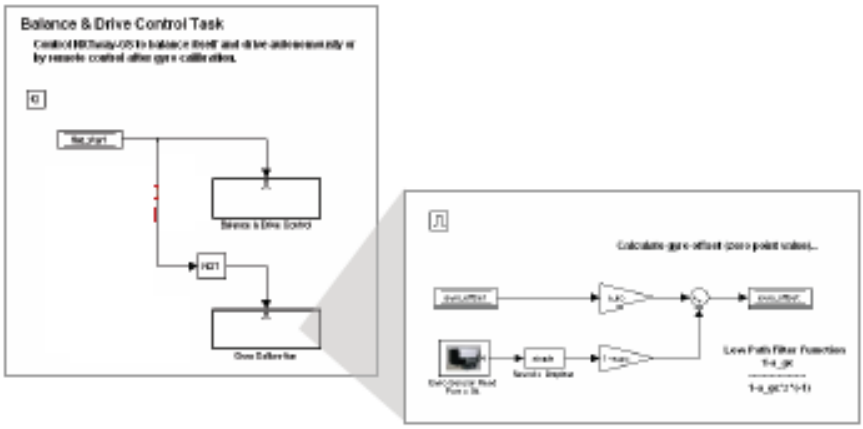


Рисунок 1.62 - Подсистема *task_ts1*

Далее представлены подсистемы балансировки и управления.

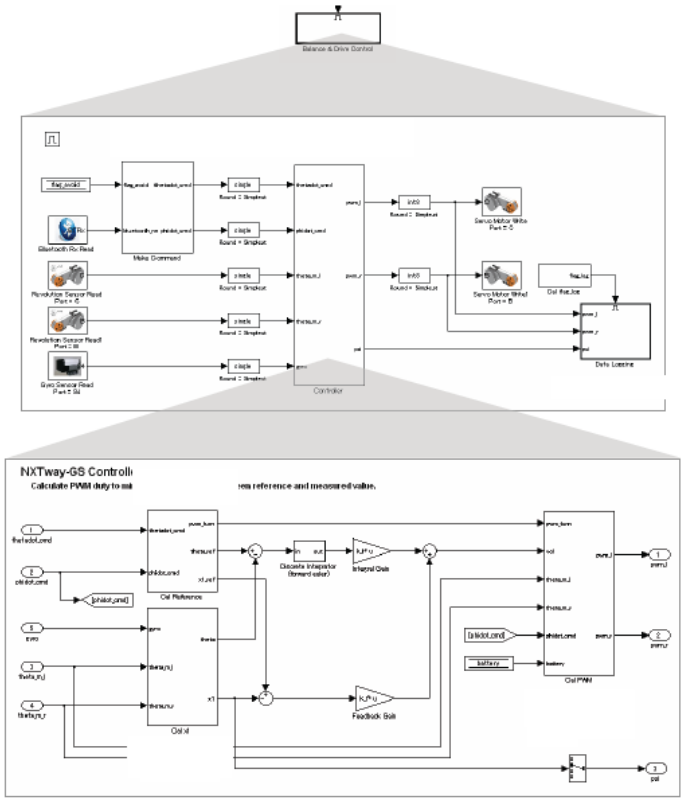


Рисунок 1.63 – Модель контроллера для управления в режиме балансировки

На схеме, показанной на рисунке 1.64 представлен «Discrete Derivative Block», который вычисляет постоянную времени методом обратного дифференцирования и «Discrete Integrator Block», который вычисляет временной интеграл методом Эйлера.

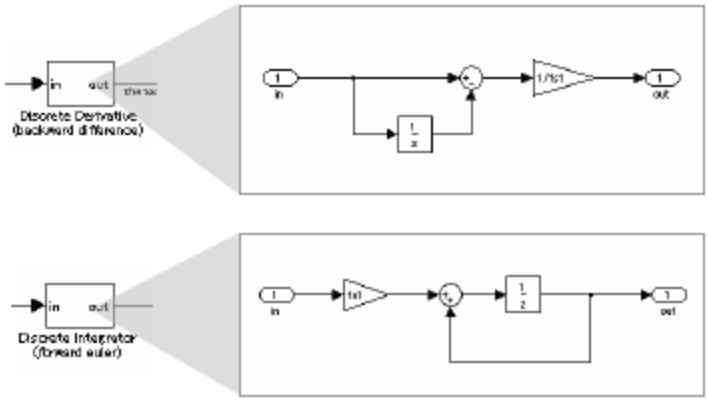


Рисунок 1.64 - Discrete Derivative и Discrete Integrator Block

Также на рисунке 1.64 присутствуют блоки, отвечающие за расчет уровня сигнала, использующие низкочастотный фильтр для уменьшения резких скачков, вызванных резкими изменениями скорости генерируемого сигнала.

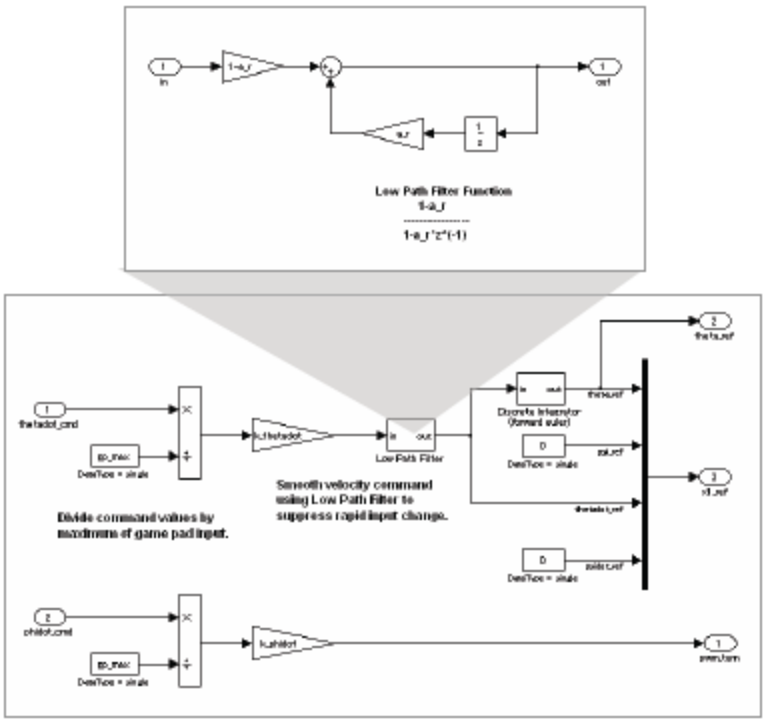


Рисунок 1.65 – Расчет генерируемых управлений

Следующая подсистема рассчитывает состояние системы, используя выходные сигналы датчиков. Используется длительное усреднение данных гироскопа для удаления гиродрифта, а также низкочастотный фильтр для удаления шумов в сигнале скорости.

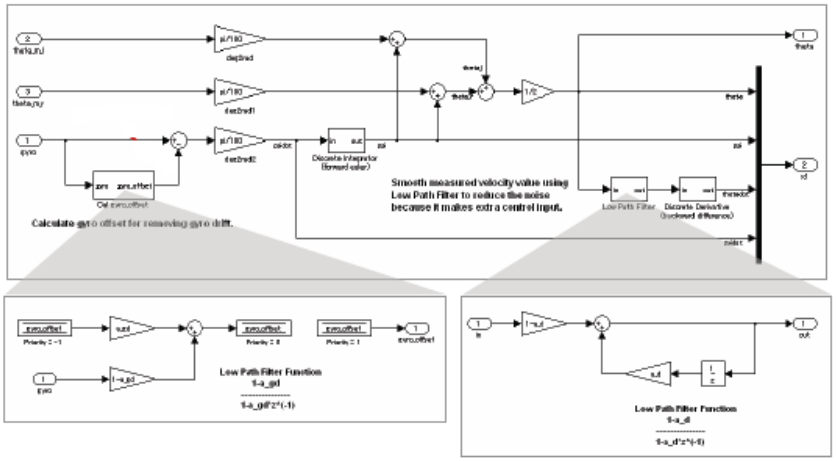


Рисунок 1.66 – Состояние системы

Следующая подсистема отвечает за расчет мощности, подаваемой на сервоприводы.

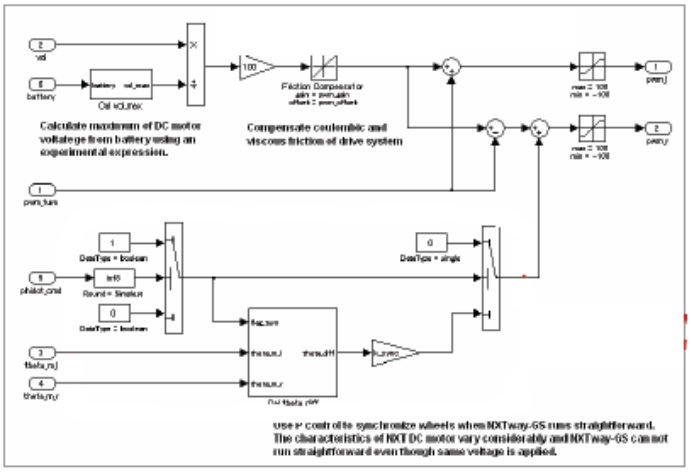


Рисунок 1.67 – Расчет мощности

1.3.2.6 Результаты моделирования

Используя утилиту NXT GamePad можно получать и записывать значение угла наклона робота с гироскопа и положения робота относительно начала движения используя значения энкодера одного из серводвигателей. На рисунках 1.68 и 1.69 представлены графики данных, полученных при балансировании робота на месте и движении робота соответственно.

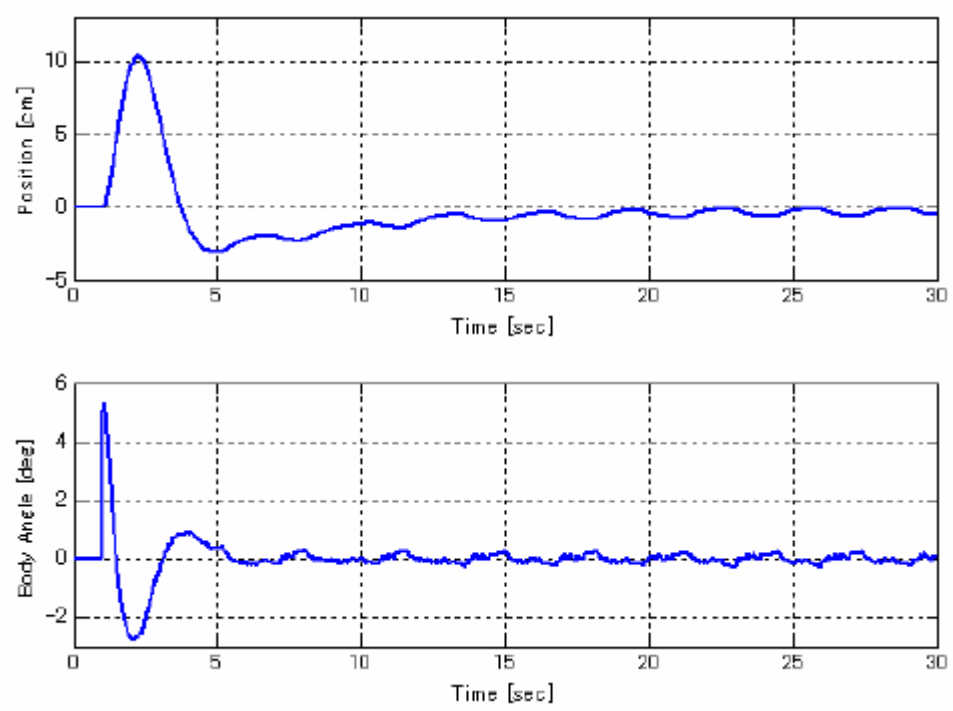


Рисунок 1.68 – Результат моделирования при балансировании

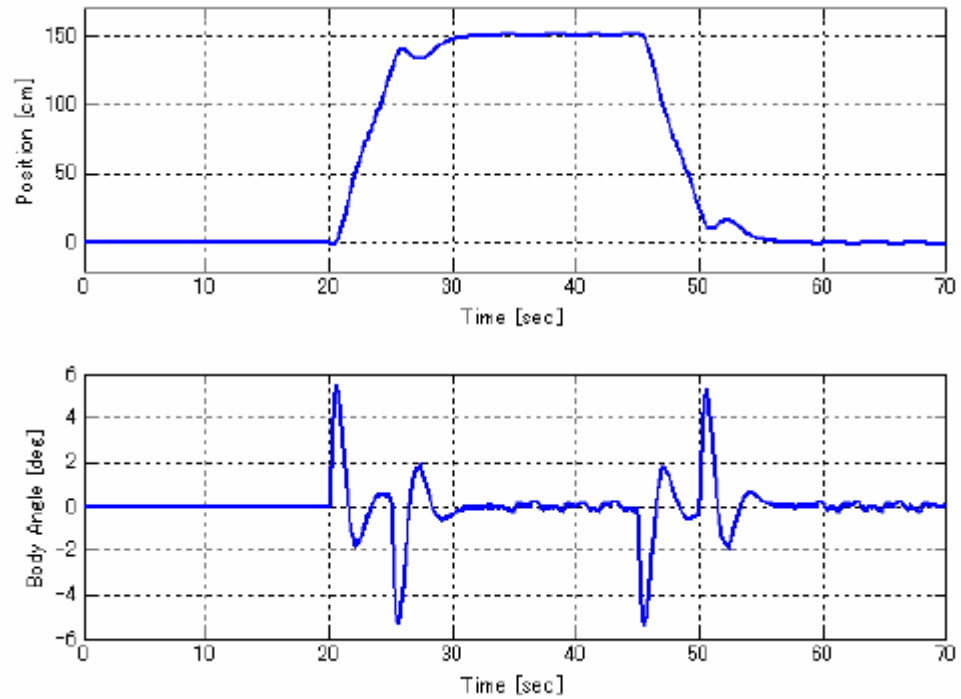


Рисунок 1.69 – Результаты моделирования при движении

1.3.3 Модель робота «Болбот»

1.3.3.1 Расчет математической модели

Рассматривается модель робота «Болбот» (рисунок 1.70), в которой движения в продольном и поперечном направлениях не связаны, и уравнения

движения в этих двух плоскостях одинаковы. Тогда болбот можно рассматривать как две модели отдельных одинаковых перевернутых маятников на сферическом катке [1], как на рисунке 1.65.

Рисунок 1.70 – Балансирующий робот «Болбот»

Двигатель вращает сферический каток через колесо с резиновой покрышкой. Допускается, что между ними нет скольжения. θ_m – угол поворота двигателя. $R_w\theta_m = R_s\theta_s$ (рисунок 1.71).

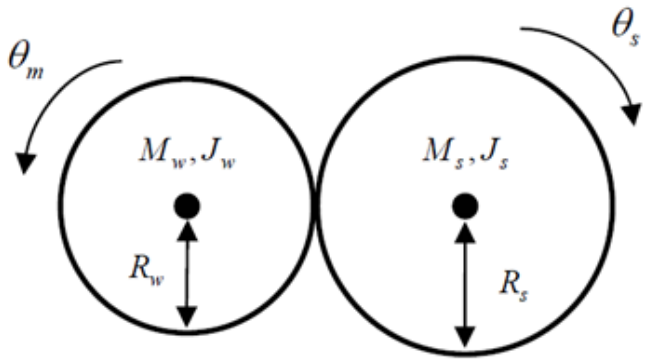


Рисунок 1.71 – Двигатель с присоединенным колесом и сферический каток

Опишем уравнение движения перевернутого маятника на сферическом катке методом Лагранжа, основываясь на системе координат, указанной ранее. Если $\theta = 0$ при $t=0$, тогда каждая координата задаётся следующим образом:

$(x_s, z_s) = (R_s\theta, z_b)$, $(\dot{x}_s, \dot{z}_s) = (R_s\dot{\theta}, 0)$,

$(x_b, z_b) = (x + L\sin\psi, z + L\cos\psi)$, $(\dot{x}_b, \dot{z}_b) = (R_s\theta + L\dot{\psi}\cos\psi, -L\dot{\psi}\sin\psi)$.

Кинетическая энергия поступательного движения T_1, вращательного движения T_2, потенциальная энергия U записываются следующим образом:

$$T_1 = \frac{1}{2} M_s (\dot{x}_s^2 + \dot{z}_s^2) + \frac{1}{2} M_b (\dot{x}_b^2 + \dot{z}_b^2) \ ,$$

$$T_2 = \frac{1}{2} J_s \dot{\theta}_s^2 + \frac{1}{2} J_\psi \dot{\psi}^2 + \frac{1}{2} J_w \dot{\theta}_m^2 + \frac{1}{2} J_m \dot{\theta}_m^2 = \frac{1}{2} J_s \dot{\theta}_s^2 + \frac{1}{2} J_\psi \dot{\psi}^2 + \frac{1}{2} (J_m + J_w) \frac{R_s^2}{R_w^2} (\dot{\theta} - \dot{\psi}),$$

$$U = M_s g z_s + M_b g z_b \ .$$

Лагранжиан L имеет вид:

$$L = T_1 + T_2 - U \ .$$

Используем θ и ψ как обобщенные координаты. Уравнение Лагранжа имеет вид:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \theta}\right) - \frac{\partial L}{\partial \theta} = F_\theta,$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \psi}\right) - \frac{\partial L}{\partial \psi} = F_\psi .$$

С учетом постоянного крутящего момента двигателя и вязкого трения, обобщенные силы описываются:

$$F_\theta = K_t i - f_m \dot{\theta}_m - f_s \dot{\theta}, \ F_\psi = -K_t i - f_m \dot{\theta}_m,$$

где i – ток двигателя; $\dot{\theta}_m = k(\dot{\theta} - \dot{\psi})$ – угловая скорость двигателя.

Пренебрегая трением внутри двигателя и индуктивностью, обобщенные силы выражаются [4]:

$$F_\theta = \alpha\upsilon - (\beta + f_s)\dot{\theta} + \beta\dot{\psi}, F_\psi = -\alpha\upsilon + \beta\dot{\theta} - \beta\dot{\psi}, \alpha = \frac{K_t}{R_m}, \beta = k(\frac{K_t K_b}{R_m} + f_m) .$$

Линеаризуем уравнения состояния, рассматривая предел $\psi \to 0 (\sin\psi \to \psi, \cos\psi \to 1)$ и пренебрегая слагаемым второго порядка $\dot{\psi}^2$, запишем в векторно матричной форме:

$$E\begin{bmatrix}\ddot{\theta}\\ \ddot{\psi}\end{bmatrix} + F\begin{bmatrix}\dot{\theta}\\ \dot{\psi}\end{bmatrix} + G\begin{bmatrix}\theta\\ \psi\end{bmatrix} = H\upsilon,$$

$$E = \begin{bmatrix}(M_b + M_s)R_s^2 + J_s + J_m & M_b L R_s - k^2 J_m \\ M_b L R_s - k^2 J_m & M_b L^2 + J_\psi + k^2 J_m\end{bmatrix}, \ F = \begin{bmatrix}\beta + f_s & -\beta \\ -\beta & \beta\end{bmatrix},$$

$$G = \begin{bmatrix}0 & 0 \\ 0 & -M_b g L\end{bmatrix}, \ H = \begin{bmatrix}\alpha \\ -\alpha\end{bmatrix}.$$

Обозначим через x – вектор состояния; u – вход.

$$x = \begin{bmatrix}\theta & \psi & \dot{\theta} & \dot{\psi}\end{bmatrix}^T; \ u = \upsilon;$$

Таким образом, представим уравнения состояния в форме Коши:

$$\dot{x} = A(t)x(t) + B(t)u(t)$$

$$A = \begin{bmatrix}0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A(3,2) & A(3,3) & A(3,4) \\ 0 & A(4,2) & A(4,3) & A(4,4)\end{bmatrix}, \ B = \begin{bmatrix}0 \\ 0 \\ B(3) \\ B(4)\end{bmatrix}$$

$$A(3,2) = M_b g L E(1,2) / \det(E)$$

$$A(4,2) = -M_b g L E(1,1) / \det(E)$$

$$A(3,3) = -[(\beta + f_s)E(2,2) + \beta E(1,2)] / \det(E)$$

$$A(4,3) = [(\beta + f_s)E(1,2) + \beta E(1,1)] / \det(E)$$

$$A(3,4) = -\beta[E(2,2) + E(1,2)] / \det(E)$$

$$A(4,4) = -\beta[E(1,1) + E(1,2)] / \det(E)$$

$$B(3) = -\alpha[E(2,2) + E(1,2)] / \det(E)$$

$$B(4) = -\alpha[E(1,1) + E(1,2)] / \det(E)$$

$$\det(E) = E(1,1)E(2,2) - E(1,2)^2 .$$

1.3.3.2 Расчет регуляторов

На вход системы управления подается напряжение управления двигателями. Выходом системы являются значения с энкодеров угла поворота двигателя θ_m и угловая скорость отклонения конструкции от вертикали $\dot{\psi}$. Численное значение угла отклонения ψ получается путем интегрирования угловой скорости $\dot{\psi}$. Начальные условия при интегрировании определяются из условия вертикального запуска, выбранного положения объекта.

Положение равновесия является неустойчивым. При минимальном отклонении, необходимо двигать болбот в направлении угла наклона конструкции, чтобы удержать баланс.

Линейно-квадратичный регулятор. Для решения задачи удержания перевернутого маятника в положении неустойчивого равновесия синтезируется пропорционально-интегральный регулятор (рисунок 1.72). C_θ – матрица выхода для получения θ из x.

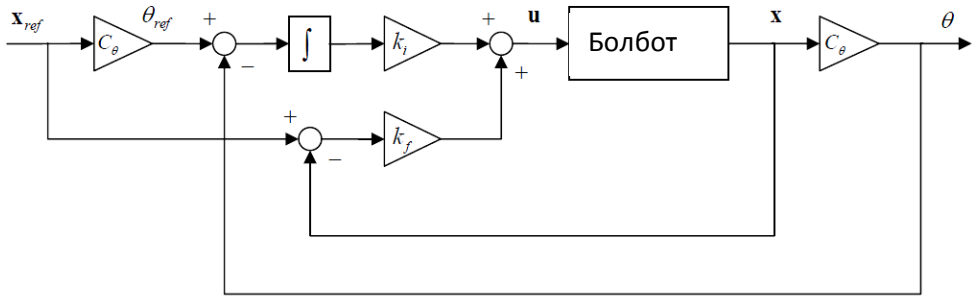


Рисунок 1.72 - Схема пропорционально-интегрального регулятора

Рассчитаем коэффициенты пропорциональной и интегральной составляющих на основе метода линейно-квадратичного регулятора. Линейно-квадратичный регулятор – в теории управления один из видов оптимальных регуляторов, использующий квадратичный функционал качества [28]. Выберем весовые матрицы *Q* и *R* [29]:

$$Q=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 0 & 6*10^5 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 10^3\end{bmatrix},\ R=6*10^3*(\frac{180}{\pi})^2\ ,$$

где $Q(2,2)$ – элемент весовой матрицы состояния, характеризующий вес значения угла отклонения конструкции, $Q(5,5)$ – элемент весовой матрицы состояния, характеризующий вес по времени интегрирования разницы между ранее измеренным углом и полученным.

Используя полученные ранее математическую модель робота и весовую матрицу, рассчитаем в среде Matlab численные значения коэффициентов для пропорциональной и интегральной составляющих. Функция *lqr* вычисляет матрицу коэффициентов регулирования со среднеквадратичным функционалом качества [30].

В результате вычислений получается набор коэффициентов:

k_f = [–0,015 –1,5698 –0,027 –0,2325], k_i = –0,0071.

Качественная экспоненциальная устойчивость. Построим регулятор, обеспечивающий экспоненциальную сходимость со следующими показателями качества: $t_n = 0{,}36$ с; $\delta_n = 0$.

В линейных системах подлежит минимизации квадратичный критерий качества:

$$J(x,u)=\int_0^\infty\left[x^T(t)Q(t)x(t)+u^T(t)R(t)u(t)\right]dt\ .$$

Непрерывная система экспоненциально устойчива в точке $x=0$, если существуют такая квадратичная функция Ляпунова

$$V(x(t))=x^T(t)P(t)x(t),$$

где $P=P^T$ – положительно определенная $n\times n$ матрица, и такой параметр $\lambda:\lambda<0$, при которых на всех траекториях движения системы в любой момент времени $t\ge 0$ выполняется условие:

$$\dot V(x(t))\le 2\lambda V(x(t)).$$

Данное условие имеет место, если справедливо уравнение:

$$\dot V(x(t))-2\lambda V(x(t))=-x^T(t)Q(t)x(t)-u^T(t)R(t)u(t).$$

Подставив квадратичную функцию Ляпунова и ее производную, получим:

$$[A(t)x(t)+B(t)u(t)-\lambda x(t)]^T P(t)x(t)+x^T(t)P(t)[A(t)x(t)+B(t)u(t)-\lambda x(t)]+x^T(t)\dot P(t)x(t)+$$
$$+x^T(t)Q(t)x(t)+u^T(t)R(t)u(t)=0,$$

на основе которого, воспользовавшись методом локальной оптимизации [31], находим оптимальное управление:

$u(t) = R^{-1}(t)B^T(t)P(t)x(t)$.

Получим систему матричных уравнений типа Риккати [31]:

$\dot{P}(t) + [A(t) - B(t)K(t) - \lambda I]^T P(t) + P(t)[A(t) - B(t)K(t) - \lambda I] + K^T(t)R(t)K(t) + Q(t) = 0$,

$K(t) = R^{-1}(t)B^T(t)P(t)$.

Так как матрицы A, B, Q, R постоянны, то процесс достигает установившегося состояния в том смысле, что становится постоянной матрица P ($\dot{P} = 0$):

$[A - BK - \lambda I]^T P + P[A - BK - \lambda I] + K^T RK + Q = 0$, $K = R^{-1}B^T P$.

Из матричных алгебраических уравнений, получается следующая матрица обратных связей K регулятора: $K_f = [-0{,}0117 \quad -1{,}4993 \quad -0{,}0284 \quad -0{,}2432]$.

1.3.3.3 Физические параметры модели

Таблица 1.15 – Физические параметры системы

g	9,81	м/с^2	Ускорение свободного падения
M_s	0,013	Кг	Масса мяча
R_s	0,026	М	Радиус мяча
J_s	2 M_s M_s^2/3	кг м^2	Момент инерции мяча
M_w	0,015	Кг	Масса колеса, соединенного с двигателем
R_w	0,021	м	Радиус колеса, соединенного с двигателем
L_w	0,022	м	Высота колеса, соединенного с двигателем
J_w	M_w L_w^2/2	кг м^2	Момент инерции колеса, соединенного с двигателем
M_b	0,682	кг	Масса конструкции
W	0,135	м	Длина конструкции
D	0,135	м	Ширина конструкции
L	0,17	м	Расстояние между центром конструкции и центром мяча
J_ψ	M_b L^2/3	кг м^2	Момент инерции наклона конструкции
J_m	0,00001	кг м^2	Момент инерции двигателя
R_m	6,69	Ω	Сопротивление двигателя
K_b	0,468	В с/рад	Возвращаемое ЭДС двигателя

1.3.3.4 Описание программы Ballbot

Главная управляющая схема

Ключевые блоки главной схемы схожи с представленными в предыдущих разделах, поэтому остановимся на ключевых, отличительных особенностях подсистем данного робота.

Рассмотрим детально некоторые из составных блоков.

На рисунке 1.73 представлена подсистема «NXT Ballbot», состоящая из датчиков, приводов, и линеаризованной модели робота. Данная подсистема преобразует тип входных данных сигналов в вещественный, рассчитывает динамику робота и выводит результаты после дискретизации. Эта подсистема определяет параметры среды. Данная подсистема, состоит из датчиков, приводов, и линеаризованной модели робота. Она преобразует тип входных данных сигналов в вещественный, рассчитывает динамику робота, и выводит результаты после дискретизации.

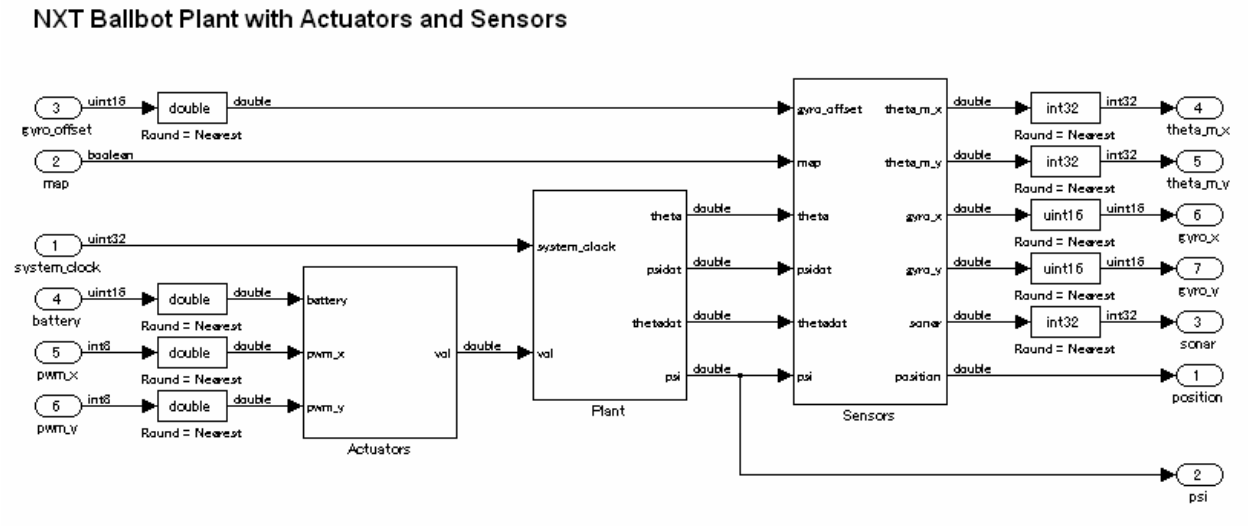


Рисунок 1.73 – Подсистема NXT Ballbot

Некоторые сигналы векторизованы для одновременного расчета по осям X и Y.

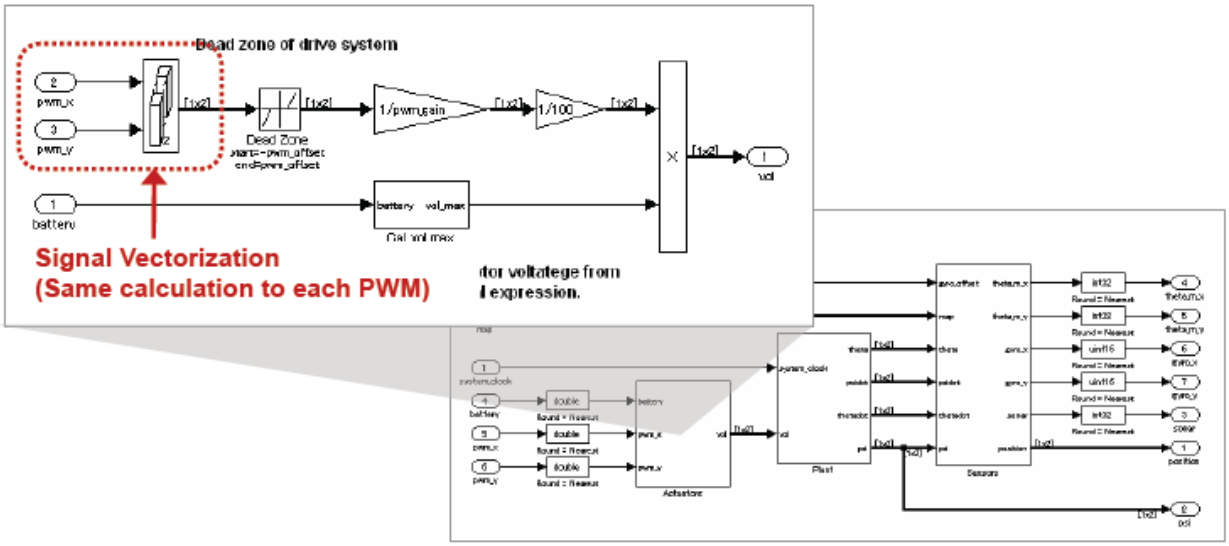


Рисунок 1.74 – Перевод сигналов в векторную форму

На рисунке 1.75 представлена подсистема, преобразующая мощность задаваемую контроллером в напряжение, подаваемое на двигатели.

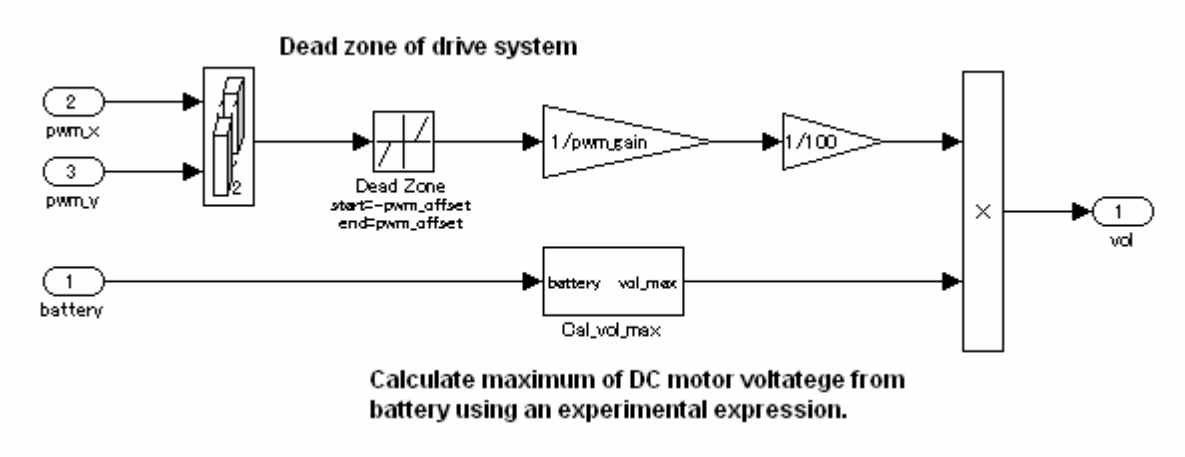


Рисунок 1.75 – Подсистема сервоприводов

На рисунке 1.76 представлена модель, описываемая уравнениями перевернутого маятника на сферическом основании, которая работает с учетом калибровки гироскопа.

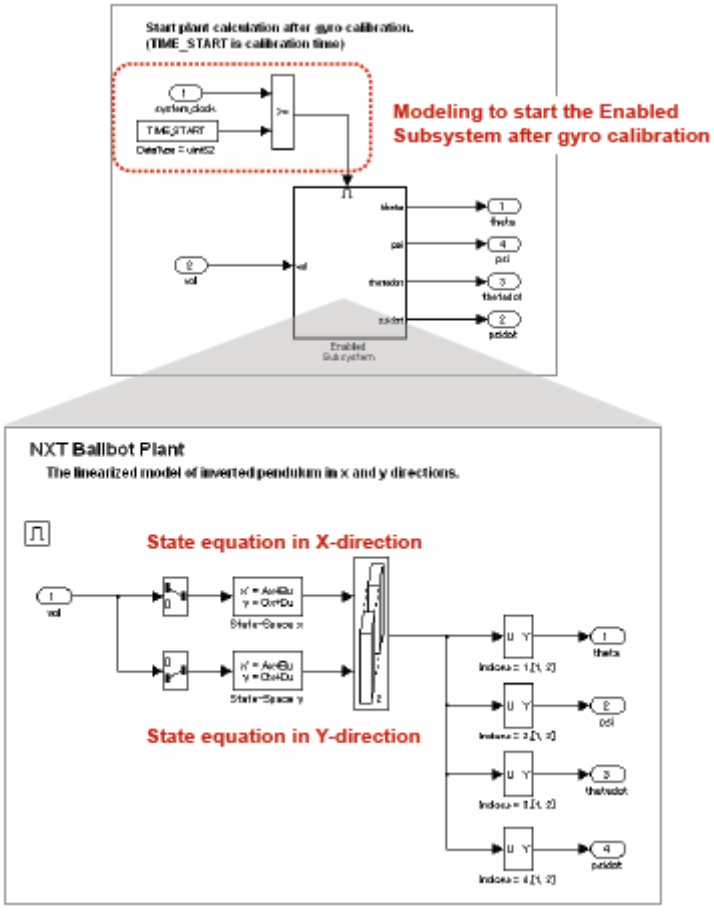


Рисунок 1.76 – Подсистема состояния модели

Блок Sensor преобразует значения, полученные от подсистемы состояния модели в выходные сигналы.

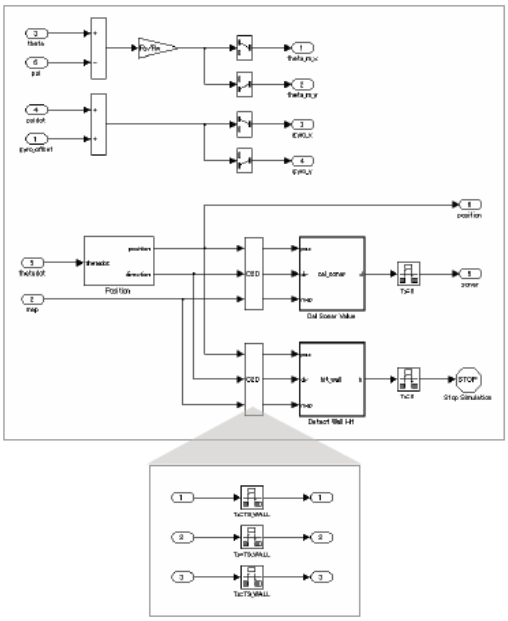


Рисунок 1.77 – Подсистема датчиков

Блок «ExpFcnCalls» содержит конфигуратор имен задач, периодичности задач, платформ и размера стека.

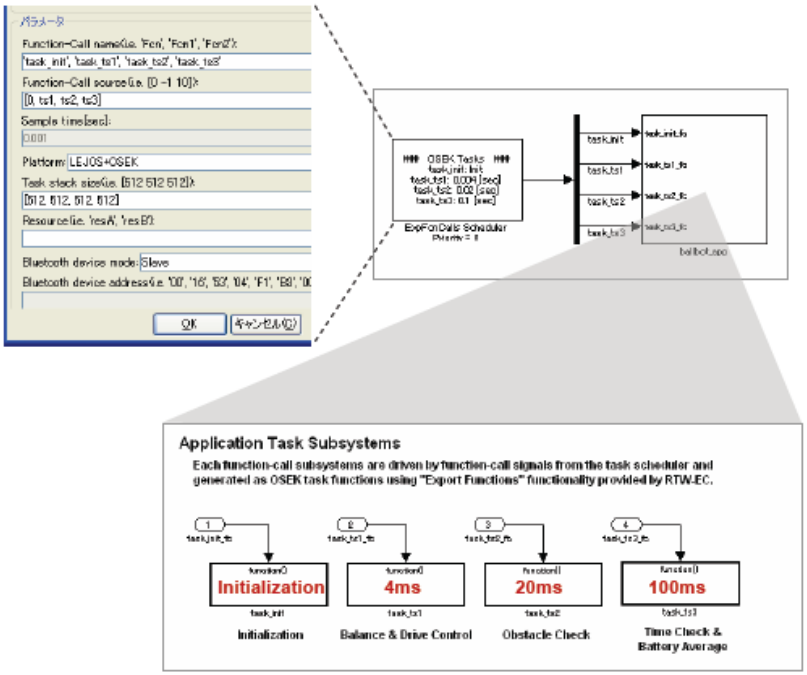


Рисунок 1.78 – Планировщик и задачи

Блоки «Data Store Memory» используются, как общие блоки данных для различных задач.

Инициализатор задач задает задачам начальные значения. Мы можем переключать режим работы (автономный или управление с помощью джойстика)

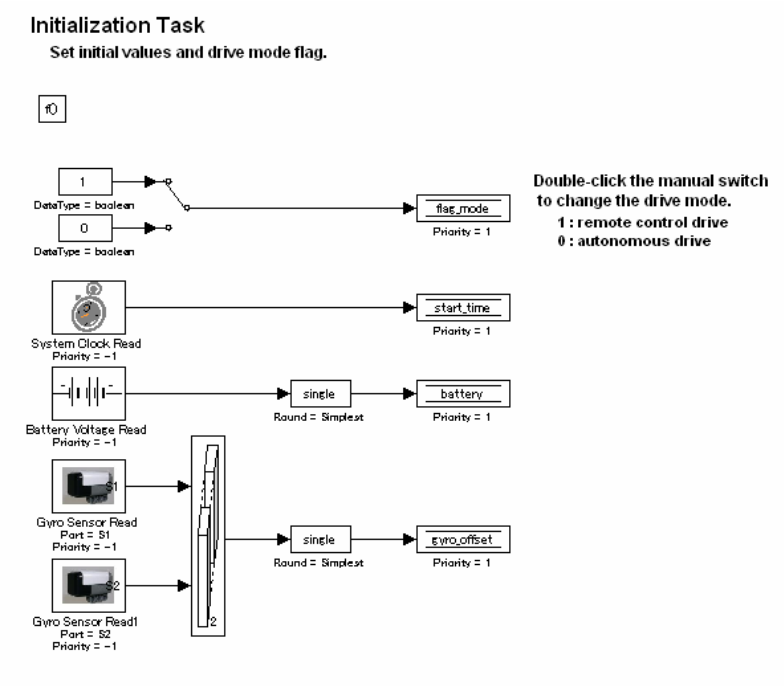


Рисунок 1.79 – Подсистема task_init

Подсистемы task_ts2-4 имеют вид, аналогичный с подсистемами Segway.

Подсистемы балансировки и управления.

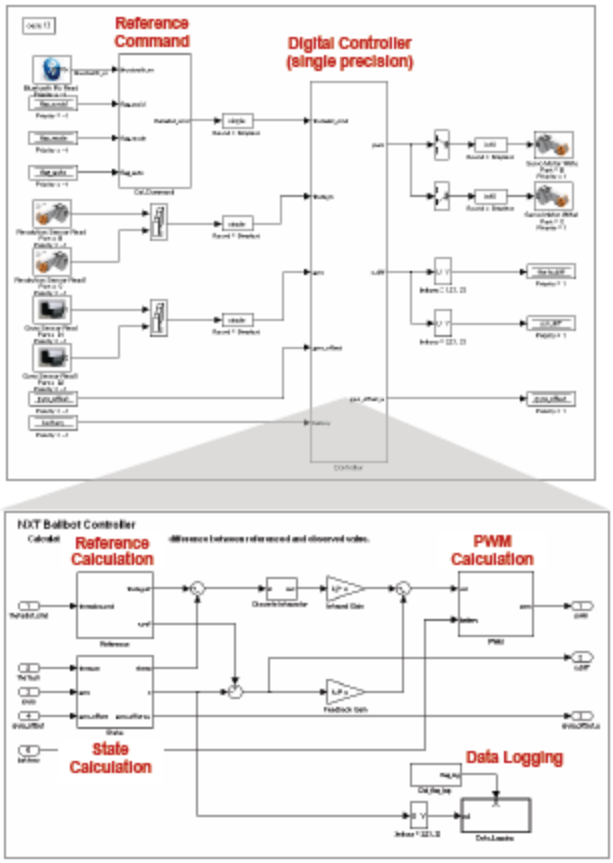


Рисунок 1.80 – Подсистемы балансировки и управления

Следующая подсистема рассчитывает управляющие команды в зависимости от режима работы и показания ультразвукового датчика.

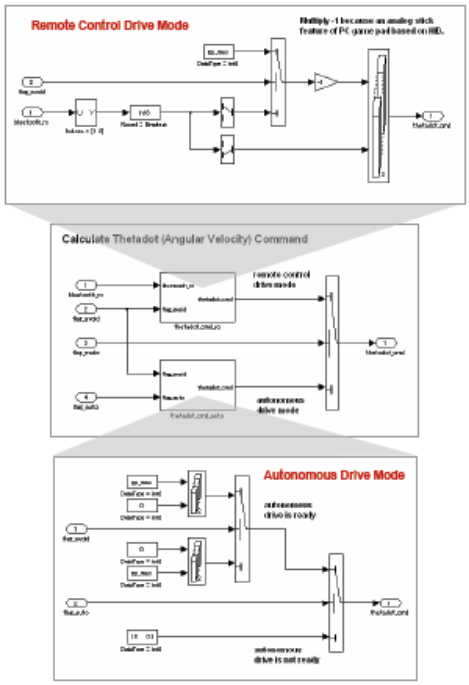


Рисунок 1.81 – Подсистема управляющих команд

Следующая подсистема рассчитывает состояние системы, используя выходные сигналы датчиков. Используются усредненные показания гироскопов для компенсации негативного эффекта от гиродрифта, и низкочастотный фильтр для исключения шумов в сигнале скорости.

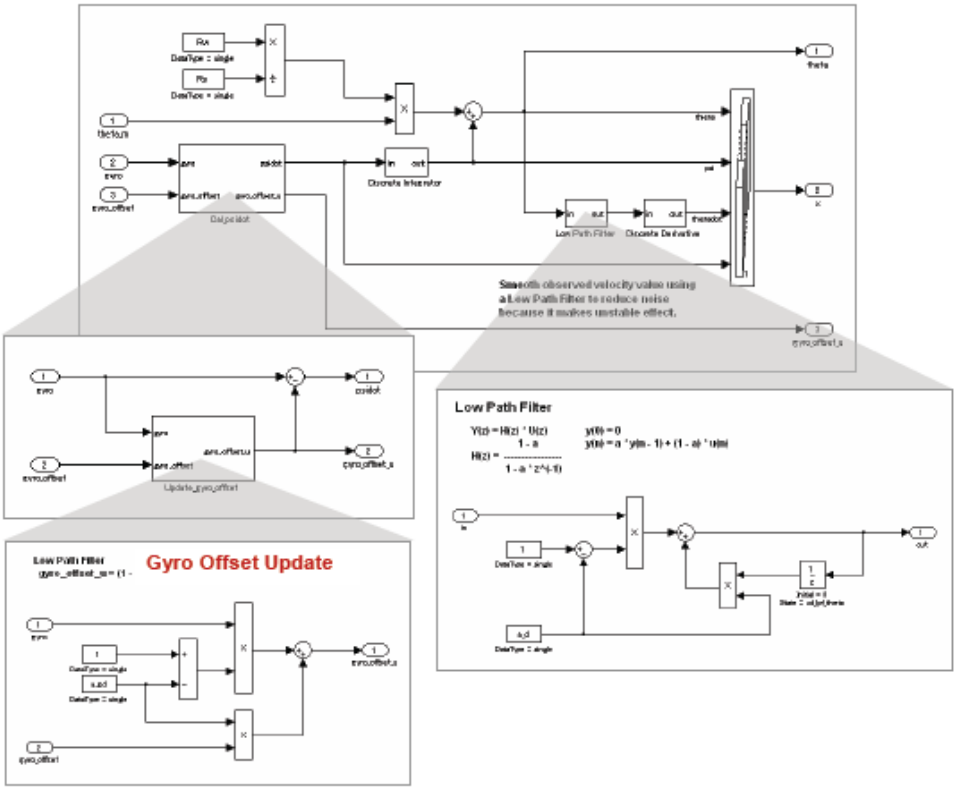


Рисунок 1.82 – Подсистема состояния робота

Следующая подсистема отвечает за расчет мощности, подаваемой на сервоприводы.

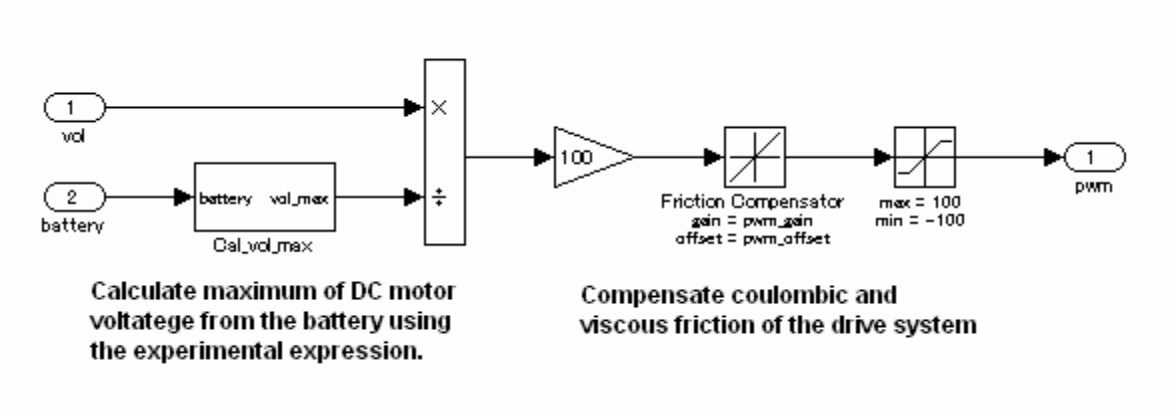
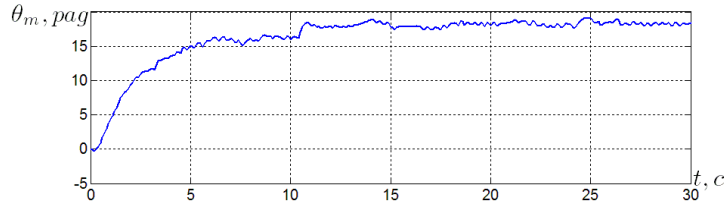


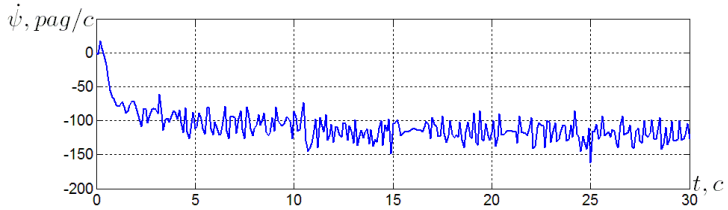
Рисунок 1.83 – Подсистема расчета величины управлений

1.3.3.5 Результаты моделирования

Используя утилиту NXT GamePad можно получать и записывать значения углов наклона робота по осям и скорости их изменения используя данные с гироскопов. Также можно получить положение робота относительно начала движения используя значения энкодеров, расположенных в серводвигателях. На рисунках 1.84 и 1.85 представлены графики данных, полученных при балансировании робота с использованием двух описанных выше регуляторов. [4]

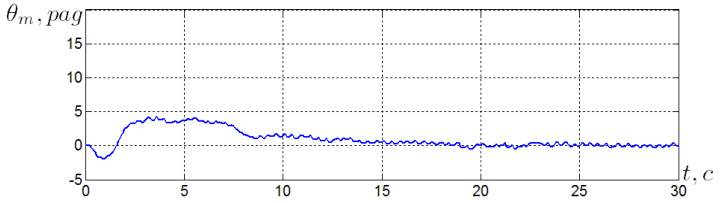


а

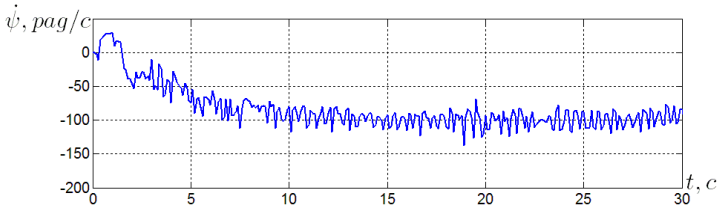


б

Рисунок 1.84 - Показания при использовании линейно-квадратичного регулятора:
угол поворота двигателя (а); скорость изменения угла крена (б)



а



б

Рисунок 1.85 - Показания при использовании регулятора на основе метода качественной экспоненциальной устойчивости: угол поворота двигателя (а); скорость изменения угла крена (б)

Таким образом, используемая система nxtOSEK является универсальной в том смысле, что предоставляет возможность разноуровневого подхода к реализации алгоритма управления роботом. С одной стороны nxtOSEK – это инструмент наглядного графического моделирования с использованием широких возможностей MATLAB/Simulink. На начальных этапах оптимальным будет использование ECRobot для быстрой конвертации готовой Simulink-модели в исполняемый роботом файл. С другой стороны nxtOSEK – это мощный инструмент программирования на языках высокого уровня. Поэтому на дальнейших этапах совершенствования возможны либо корректировка сгенерированных С-файлов, либо независимое создание более сложных модулей на языках С/С++ с реализацией всех возможностей объектно-ориентированного программирования.

2 РОБОТОТЕХНИЧЕСКИЙ КОМЛЕКС BIOLOID PREMIUM KIT

Набор Bioloid предназначен для обучения в области разработки и отладки сложных робототехнических систем. Набор включает в себя небольшие сервоприводы, называемые Dynamixel и представляющие собой самостоятельные модули, с помощью которых могут быть собраны роботы различной конструкции, например колёсные или шагающие роботы [18]. Набор Bioloid схож с наборами LEGO Mindstorms от компании LEGO и Vex Robotics Design System от компании VEX Robotics. Набор используется в Военно-морской академии США как учебное оборудование в курсе машиностроения [16]. Так же набор Bioloid часто используют участники международных соревнований RoboCup [22].

BIOLOID Premium Kit – робототехнический набор (рисунок 2.1), на базе которого пользователь может создать различные механизмы. Но в отличие от множества других популярных конструкторов, любая модель, разработанная пользователем, может быть приведена в действие при помощи специальных приводов и управляющих контроллеров. Название BIOLOID получено путем сложения слов «bio»+ «all»+ «oid», что свидетельствует о том, что любое живое существо может быть представлено в виде робота (рисунок 2.1).

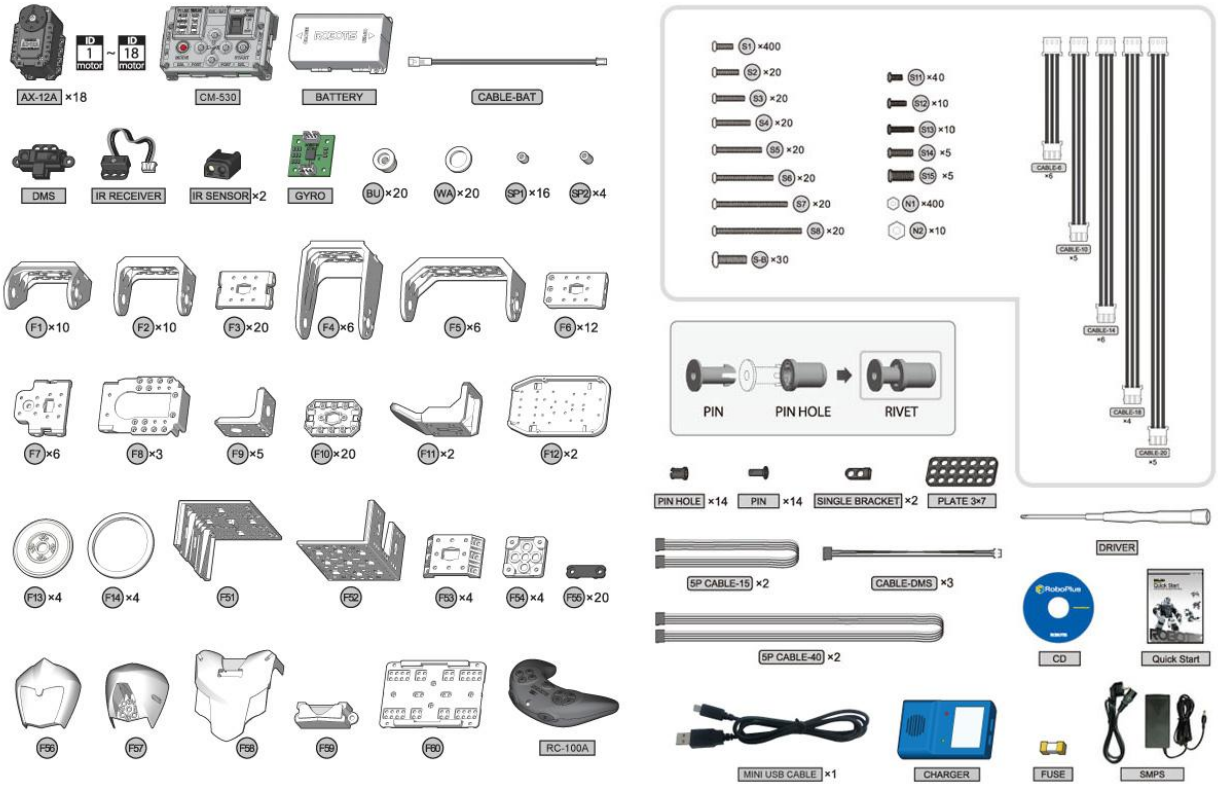


Рисунок 2.1 – Набор BIOLOID Premium Kit

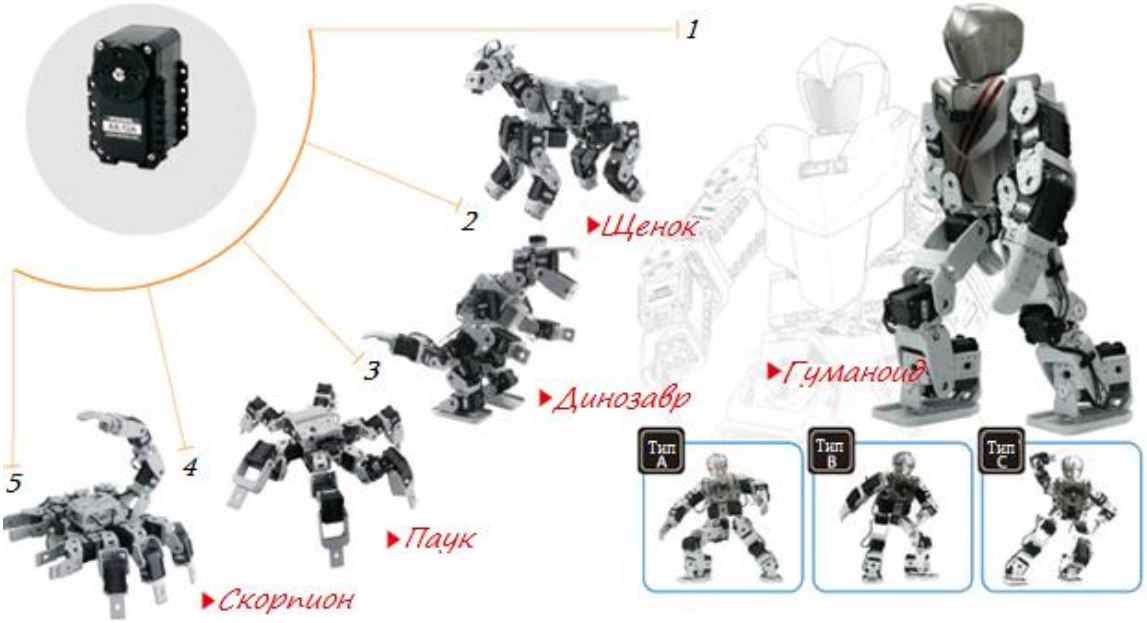


Рисунок 2.1 – Возможные модели из набора Bioloid

Используя универсальные модули и различные аппаратные компоненты (рисунок 2.2), входящие в набор, а так же с помощью специального программного обеспечения, можно собрать робота и запрограммировать его движения. В комплект Bioloid входят сервоприводы Dynamixel, набор сенсоров, программное обеспечение, включающее в себя среду 3D

моделирования и среду программирования на С-подобном языке. Количество приводов достаточно, чтобы изготовить механизм с восемнадцатью степенями свободы.

Рисунок 2.2 – Компонентная база набора Bioloid Premium Kit

CM-510 – микроконтроллер на базе ATmega2561 с портами для датчиков и сервомоторов DYNAMIXEL (рисунок 2.3):

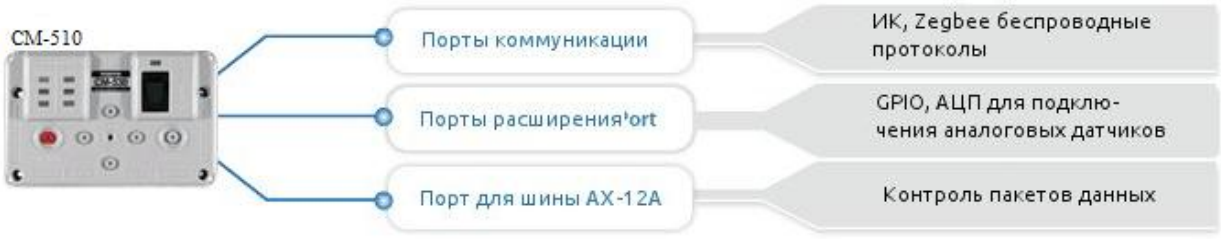


Рисунок 2.3 – Возможности микроконтроллера CM-510

2.1 ВОЗМОЖНОСТИ СЕРВОПРИВОДОВ DYNAMIXEL

Сервопривода Robotis Dynamixel Servos одинаково востребованы как у профессионалов, так и у любителей в области робототехники. Они по праву занимают свое место между легкодоступными сервоприводами, применяемыми в различных любительских управляемых моделях, и

прецизионными профессиональными сервоприводами, обладающими дорогой и точной механической конструкцией (например, на базе волновых зубчатых передач) и сложной системой управления.

Специально для использования совместно с робототехническими комплексами BIOLOID разработан управляющий контроллер CM-510 (рисунок 2.4).

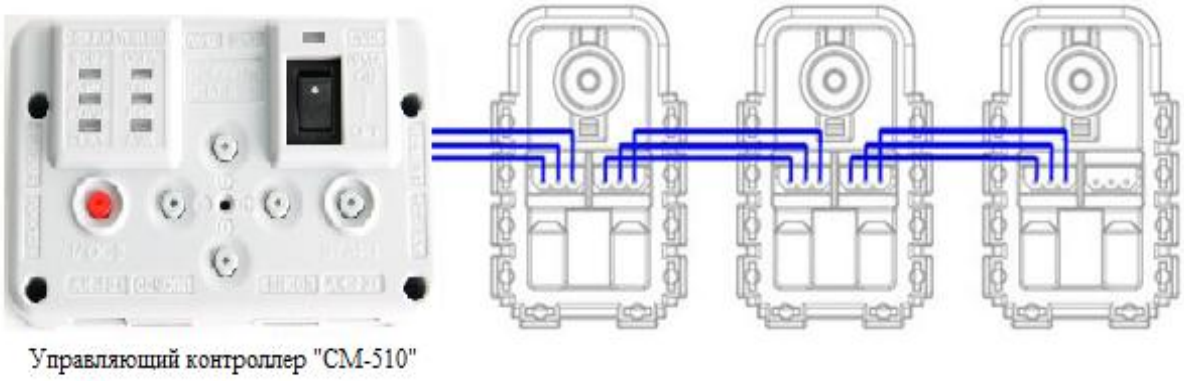


Рисунок 2.4 - Микроконтроллер CM-510 с подключенными сервоприводами Dynamixel

Использование данного контроллера и программного обеспечения для его конфигурирования позволяет облегчить задачу управления робототехническими системами. Функционал системы управления, построенной данным образом, позволяет сосредоточить максимум усилий на алгоритмах управления высокого уровня, освобождая пользователя от необходимости разработки и отладки устройств и приложений для управления отдельными сервоприводами.

Одна из основных черт сервоприводов Dynamixel – это простота и модульность конструкции и системы управления. Многообразие специальных крепежных скоб позволяет легко собрать любую конструкцию. Большой диапазон рабочих моментов от 16 кг*см до 106 кг*см обеспечивает возможность применения этих сервоприводов в различных конструкциях.

В случае, когда крутящего момента одного сервопривода не достаточно, можно объединить два с помощью специальной крепежной скобы. Использование двух сервоприводов Dynamixel, синхронно работающих на

общую нагрузку, позволяет удвоить крутящий момент в любом из пользовательских применений (рисунок 2.5).

Рисунок 2.5 – Скрепление двух сервоприводов Dynamixel крепежной скобой

Возможность решать с помощью сервоприводов Dynamixel задачи, связанные с точным позиционированием в пространстве, обеспечивается благодаря точной механике. В основе Dynamixel применяются всемирно известные привода Maxon серии RE-max, что в совокупности с металлической зубчатой передачей обеспечивает поддержание точности работы 0.35-0.07 угловых градусов во всем диапазоне рабочих моментов (рисунок 2.6).

Рисунок 2.6 – Сервопривод Dynamixel

2.1.1 Система управления сервоприводами Dynamixel

Система управления строится на базе специальной печатной платы, содержащей силовую электронику и управляющий контроллер. В качестве управляющего «мозга» и коммуникационного модуля выступает микроконтроллер AVR ATmega. Благодаря функциональным возможностям микроконтроллера и набору специального программного обеспечения, процесс управления приводами Dynamixel становится легкодоступным как профессионалам, так и любителям.

Рисунок 2.7 – Управления сервоприводами Dynamixel посредством микроконтроллера CM-510

Пользователь может разработать собственную систему управления на базе специального протокола, предоставляемого компанией ROBOTIS (более

подробно см. в руководствах пользователя). Управляющая электроника сервоприводов Dynamixel поддерживает стандартные последовательные протоколы RS-232 и RS-485 в зависимости от модели сервопривода (рисунок 2.8).

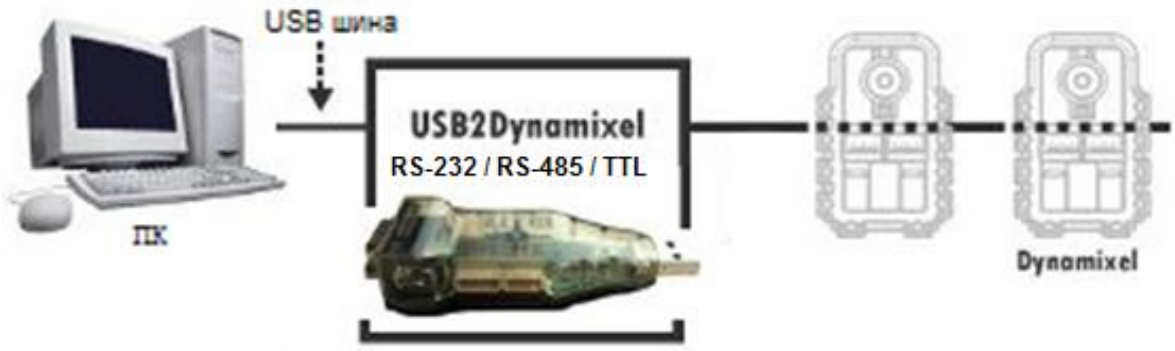


Рисунок 2.8 – Поддержка протоколов RS-232 и RS-485 управляющей электроники сервоприводов Dynamixel

2.1.2 Технические характеристики серводвигателей Dynamixel

Вес: 54.6г;

Размеры: 32мм * 50мм * 40мм;

Разрешение: 0,29°;

Передаточное отношение: 254: 1;

Момент опрокидывания: 1.5Нм (12В, 1.5А);

Скорость без нагрузки: 59rpm (при 12В);

Степень вращения: 1) 0° ~ 300°; 2) бесконечные вращение;

Рабочая температура: -5℃ ~ +70℃;

Напряжение: 9 ~ 12В (рекомендуемое напряжение 11,1В);

Управляющий сигнал: цифровой пакет;

Тип протокола: полудуплексный асинхронный последовательный протокол (8бит, 1стоп-бит, без проверки четности);

ID: 254 ID (0 ~ 253);

Скорость передачи данных: 7343bps ~ 1Mbps;

Обратная связь: положение, температура, нагрузка, входное напряжение, и т.д.;

Материал: технический пластик.

2.2 ОБЗОР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Разработка системы управления представляет собой неотъемлемый процесс «оживления» робота, представляющий собой реализацию алгоритмов и программ управления в специально предназначенной для этого среде разработки. В зависимости от сложности поставленных задач меняется и применяемый разработчиком инструментарий.

Компания Robotis предлагает собственную среду разработки RoboPlus (рисунок 2.9), состоящую из следующих компонентов: RoboPlus Terminal, RoboPlus Task, RoboPlus Manager, RoboPlus Motion, RoboPlus Help & e-Manual. Данное программное обеспечение является единым для всей продукции Robotis и может применяться для семейства OLLO, Bioloid, Expert kit [23].

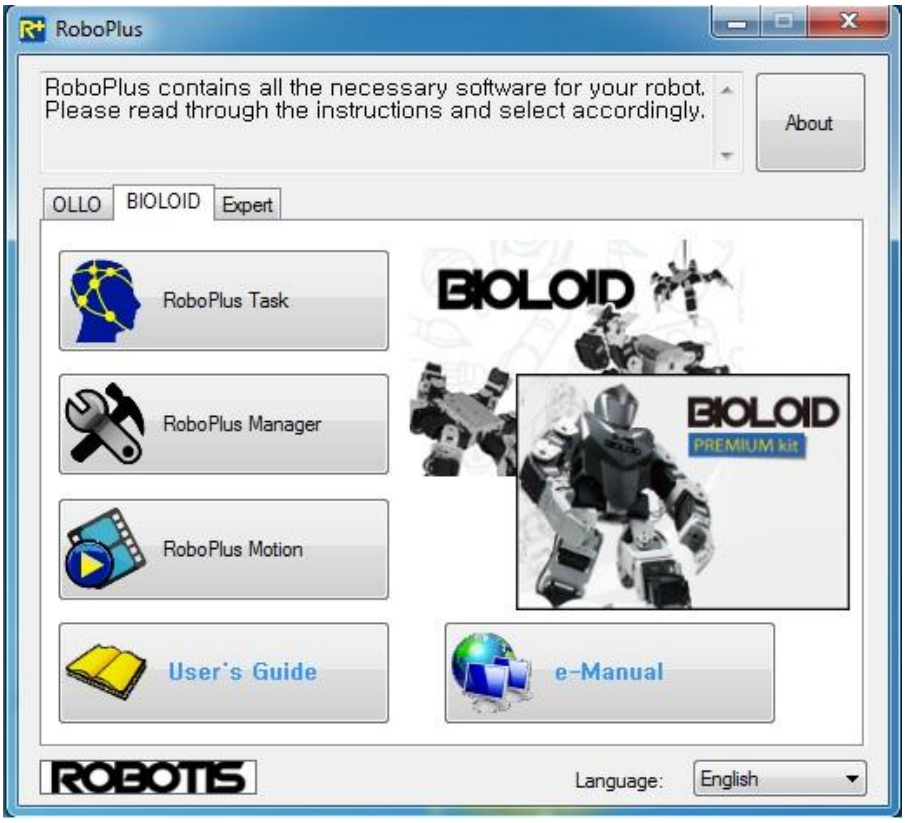


Рисунок 2.9 – Интерфейс RoboPlus

Компонента RoboPlus Manager (рисунок 2.10) предназначена для диагностики оборудования, входящего в состав робота. При помощи RoboPlus Manager можно задавать базовые настройки параметров каждой из компонент робота – сервоприводов, контроллера и т.д. RoboPlus Manager может применяться для обновления драйверов оборудования и контроллеров, выпускаемых компанией Robotis.

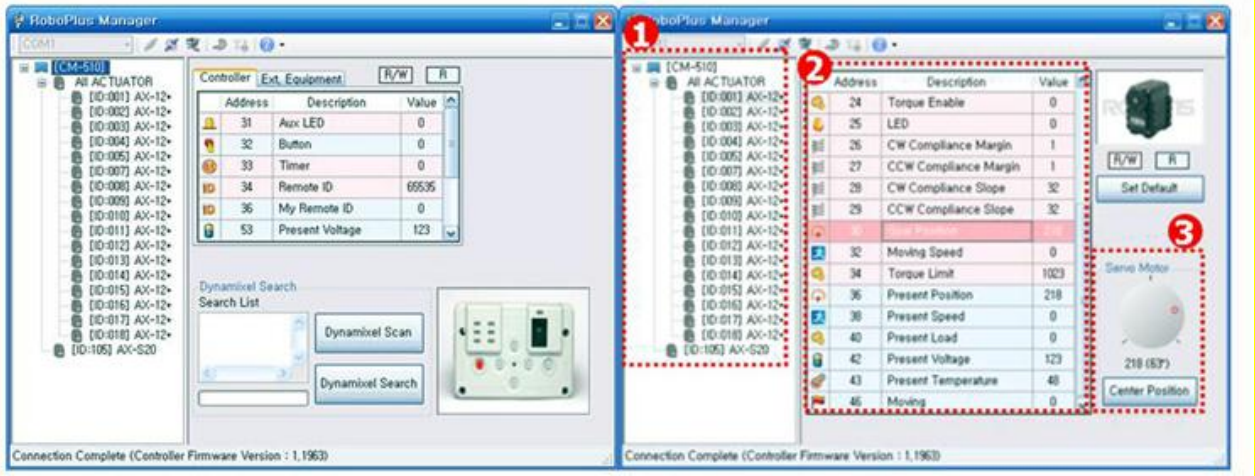


Рисунок 2.10 – Интерфейс RoboPlus Manager

С помощью RoboPlus Manager можно распределять ID между сервоприводами, входящими в состав робота. Для каждого из сервоприводов Dynamixel можно задавать набор различных ограничений: ограничение момента, ограничение углов поворота, ограничение максимальной скорости, предел тепловой защиты и т.д. Также, RoboPlus Manager обеспечивает возможность работы со всем семейством Dynamixel, с контроллерами CM-5,CM-510,CM-700 и модулем беспроводной связи Zigbee.

RoboPlus Terminal (рисунок 2.11) является инструментом, дающим пользователю возможность управлять контроллером CM-510 посредством текстовых команд при помощи специального интерфейса. Данная среда является обычным терминалом последовательного порта, при помощи которого можно осуществлять прием и передачу данных между компьютером и контроллером CM-510. Также прошивка микроконтроллера осуществляется с помощью вышеописанного терминала или Boot Loader.

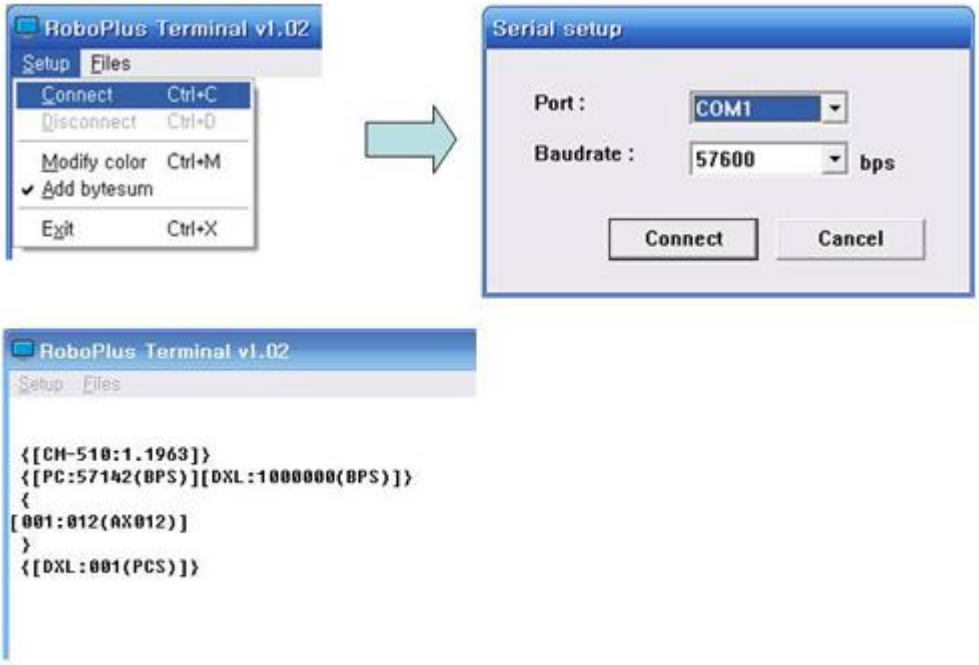


Рисунок 2.11 – Интерфейс RoboPlus Terminal

RoboPlus Motion (рисунок 2.12) – среда разработки, позволяющая задавать последовательность движений исполнительных механизмов и моделировать полученный результат.

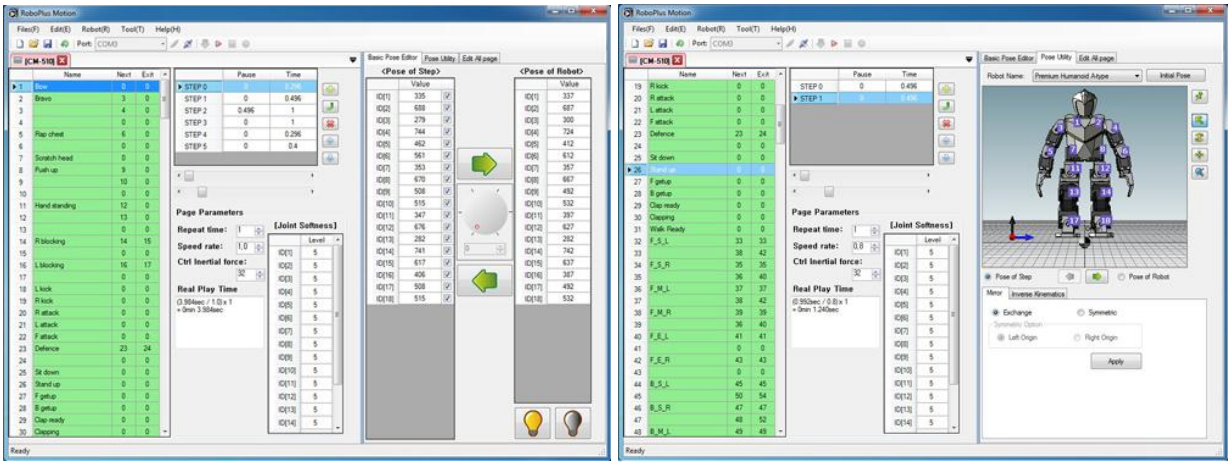


Рисунок 2.12 - Интерфейс RoboPlus Motion

Окно среды разработки состоит из трех частей. Слева расположен список последовательностей действий робота, соответствующий загруженной в контроллер CM-510 программе. Каждое действие, выполняемое роботом, может быть представлено в виде совокупности отдельных шагов, разнесенных по различным временным интервалам.

Каждый шаг редактируется в центральном окне пользовательского интерфейса.

Справа во вкладке Basic Pose Editor отображаются задаваемые положения каждого из приводов робота, соответствующие выбранному шагу, и положение приводов робота на текущий момент. Во вкладке Pose Utility отображается конфигурация робота, получаемая вследствие переходов между определенными шагами или выполнения заданных операций.

Данная среда предоставляет возможность путем простых манипуляций создать программу, заставляющую робота выполнять различные движения. Возможность пошагово наблюдать за ходом разработки в окне моделирования и на реальном макете существенно упрощает процесс разработки и дает представление о будущих результатах [24].

RoboPlus Task (рисунок 2.13) применяется для программирования робота посредством набора стандартных команд, предназначенных для реализации определенных действий и последовательностей, организации циклов, выполнения условий.

Рисунок 2.13 – Интерфейс RoboPlus Task

Однако RoboPlus Motion совместно с RoboPlus Task не обладают необходимым функционалом для построения сложных систем управления роботом и их применение целесообразно лишь на начальных и ознакомительных этапах работы с комплексом.

Для решения более сложных задач компания Robotis предоставляет возможность написания программного обеспечения исполнительного уровня. Специально для этого разработаны множество библиотек для работы с базовым оборудованием на языке C [17]. Программирование контроллеров серии CM, а именно - микроконтроллера семейства ATmega, осуществляется с помощью Atmel Studio, WinAVR или AVRStudio на усмотрение разработчика. Остановимся на среде разработки Atmel Studio 6, так как она является развитием AVR Studio 5, и поэтому унаследовала все её возможности. Она построена на базе Microsoft Visual Studio Shell 10 с надстройкой Visual Assist, что вкупе обеспечивает удобную разработку проектов как на C/C++.

Atmel Studio 6 включает в себя набор библиотек Atmel Software Framework, где в исходных кодах реализованы библиотеки для работы со всеми периферийными устройствами микроконтроллеров, а для упрощения их освоения, в состав Atmel Studio 6.0 включено более 1000 примеров проектов, использующих эти библиотеки. В Atmel Studio 6 также реализован функционал просмотра содержимого набора библиотек Atmel Software Framework (ASF Explorer) и интегрирована библиотека Qtouch composer, служащая для создания сенсорных интерфейсов [25].

Редактор имеет функционал: подсветка синтаксиса, интерактивные "подсказки", автодополнение ключевых слов, и т.д. Интегрированная среда разработки содержит ряд "мастеров", помогающих в подключении нужных драйверов из Framework, настройки периферии, параметров компиляции и т.п.

Особенности программы «Atmel Studio»:

1. Помощь в отладке Atmel приложений;
2. Удобная среда для работы;
3. Редактирование кода;
4. Приложение имеет большую библиотеку с примерами исходных кодов;

5. Atmel Studio находится в свободном доступе.

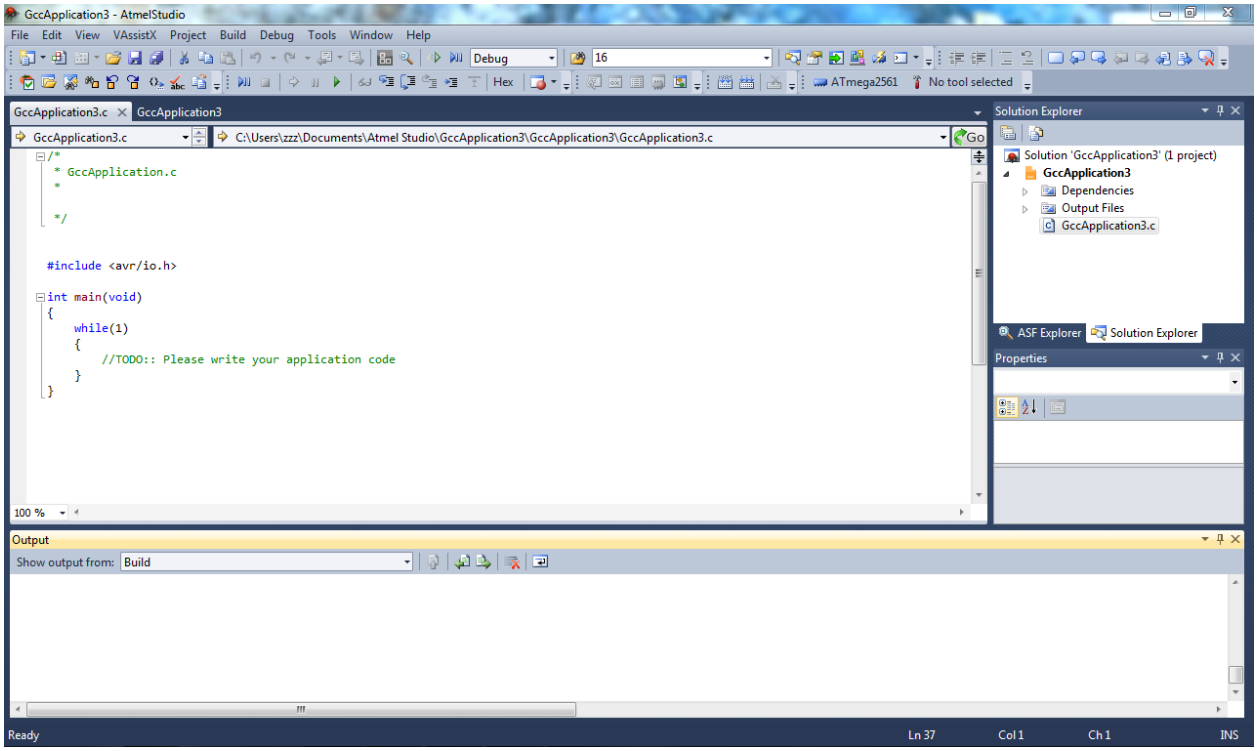


Рисунок 2.14 – Интерфейс Atmel Studio 6.0

После написания кода программы на языке С его необходимо скомпилировать. Полученный после компиляции в Atmel Studio .hex файл необходимо загрузить в микроконтроллер. Для этого можно использовать RoboPlus Terminal:

1) Для создания подключения микроконтроллера с компьютером необходимо выбрать номер COM порта и скорость подключения. Стандартная скорость подключения 57600bps. Серийные настройки подключения приведены ниже:

- Бит четности: без проверки четности;
- Стоп-бит: 1 бит
- Бит данных: 8 бит

2) Далее необходимо удерживая клавиши shift+3 включить микроконтроллер, после чего на экране появится следующая информация (рисунок 2.15):

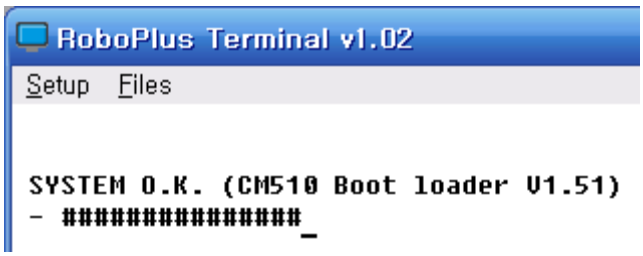


Рисунок 2.15 – Соединение контроллера CM-510 (ATmega2561) с компьютером

3) Ввести команду «L» (рисунок 2.16), выбрать "Transmit File" в меню RoboPlus Terminal, а затем выбрать подготовленный .hex файл;

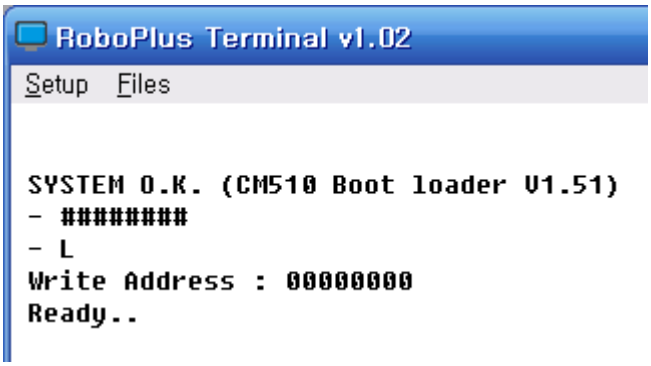


Рисунок 2.16 – Загрузка .hex файла в контроллер

4) После завершения передачи файла, можно выполнить программу с помощью команды "GO".

Положение запуска программы может быть установлено командами APP/SYS. Как только положение запуска программы установлено, программа выполняется со стартовой позиции при включении питания или сбросе. Карта памяти ATmega2561 приведена на рисунке 2.17.

Рисунок 2.17 - Карта памяти ATmega2561

Также RoboPlus Terminal позволяет производить запись в файл, например в формате .txt.

2.3 ПРОГРАММИРОВАНИЕ РОБОТОТЕХНИЧЕСКОГО КОМПЛЕКСА BIOLOID НА ЯЗЫКЕ С

Все необходимые библиотеки для работы с микроконтроллером ATmega2561 содержатся в Atmel Studio. Для использования сервоприводов Dynamixel и организации последовательной связи необходимо подключить специальные библиотеки: serial.h и dynamixel.h. При использовании передачи данных по Zigbee протоколу необходимо подключить библиотеку libzigbee.h [17].

Далее приведена карта портов микроконтроллера CM-510 и таблица управления сервоприводами Dynamixel.

2.3.1 Карта портов микроконтроллера СМ-510

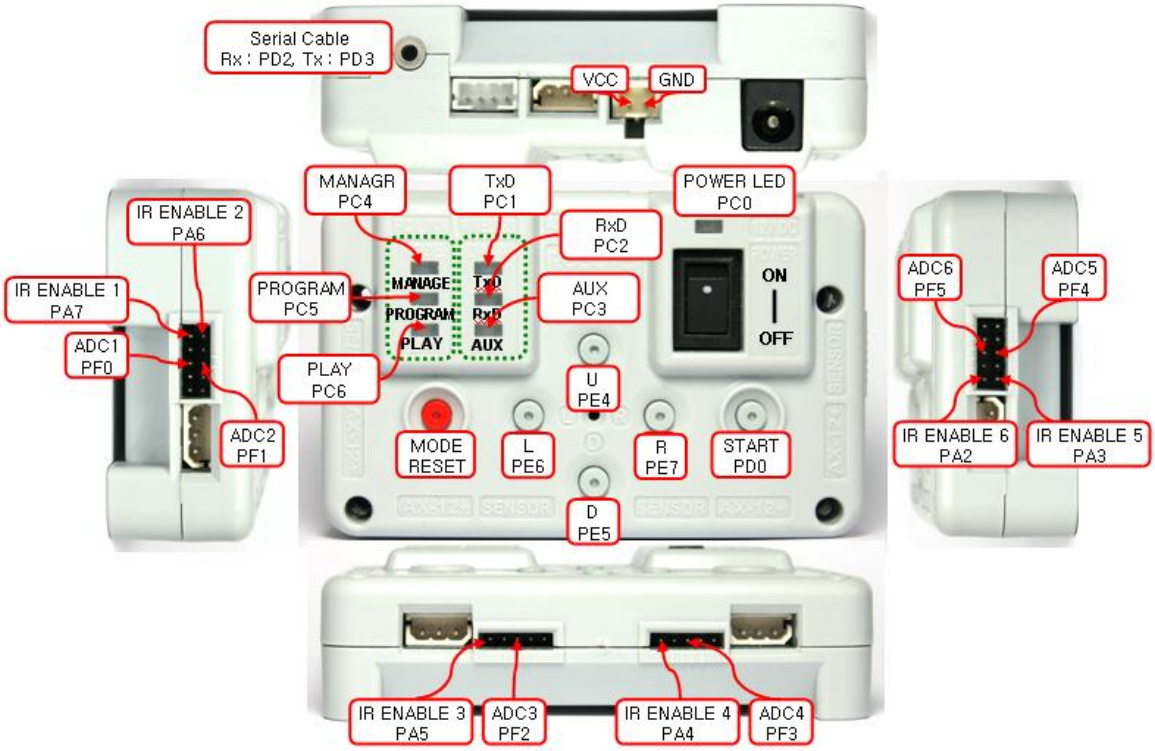


Рисунок 2.18 - Карта портов микроконтроллера СМ-510

Таблица 2.1Порты микроконтроллера СМ-510 и выполняемые ими функции

Название порта	Функция
PORTF1 ~ PORTF6	АЦП
PORTD0	Кнопка старта
PORTD1 ~ PORTD2	Запись (Tx) и чтение (Rx) данных
PORTA2 ~ PORTA7	Выходы (порт с 5 выводами (рисунок 2.19))
PORTC0 ~ PORTC6	Светодиоды (статус, питание)
PORTB5	Порт управления бипером
PORTE4 ~ PORTE6	Кнопки направления (U, D, L, R)
PORTD4 ~ PORTD6	Порты управления подключениями

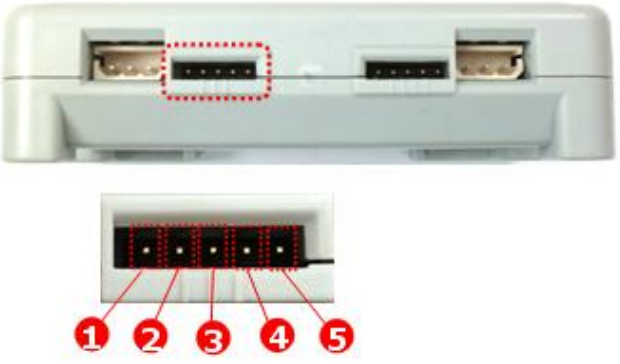


Рисунок 2.19 – Порт А микроконтроллера СМ-510

Порт А микроконтроллера:

1. Выход 5 В;
2. VCC (5 В);
3. АЦП;
4. GND;
5. NC: не используется.

2.3.2 Таблица управления сервоприводами Dynamixel

Таблица управления (таблица 2.2) состоит из данных внутри Dynamixel, касающихся текущего состояния и эксплуатации серводвигателя. Пользователь может управлять Dynamixel путем изменения данных таблицы управления с помощью пакетов инструкций.

Память EEPROM и RAM. Данные в области оперативной памяти (RAM) сбрасываются в начальное значение при включении питания, в то время как записи данных в области EEPROM сохраняются, даже если питание выключено.

Адрес представляет собой расположения данных для чтения или записи в управляющую таблицу. Для этого пользователь должен назначить правильный адрес в пакете инструкций.

Доступ. Dynamixel имеет два вида данных: данные только для чтения (R), которые используются для считывания, и данные для чтения и записи (RW), которые используются для управления.

Начальное значение. Начальные значения данных в области EEPROM обуславливаются заводскими настройками, в случае оперативной памяти указанные в таблице начальные значения устанавливаются при включении питания.

Старший / младший байт. В управляющей таблице, для некоторых данных требуется 16 бит, поэтому эти данные разделены на два адресных пространства по 8 бит – старший и младший байт. Эти два адреса должны быть написаны в одном пакете инструкций одновременно.

Таблица 2.2 Таблица управления серводвигателем Dynamixel

Область памяти	Адрес	Имя	Описание	Доступ	Начальное значение
EEPROM	0 (0x00)	Model Number(L)	Младший байт номера модели	R	12 (0x0C)
	1 (0x01)	Model Number(H)	Старший байт номера модели	R	0 (0x00)
	2 (0x02)	Version of Firmware	Информация о версии прошивки	R	-
	3 (0x03)	ID	ID серводвигателя	RW	1 (0x01)
	4 (0x04)	Baud Rate	Скорость передачи данных серводвигателя	RW	1 (0x01)
	5 (0x05)	Return Delay Time	Время задержки ответа	RW	250 (0xFA)
	6 (0x06)	CW Angle Limit(L)	Ограничение угла при вращении по часовой стрелке (младший байт)	RW	0 (0x00)
	7 (0x07)	CW Angle Limit(H)	Ограничение угла при вращении по часовой стрелке (старший байт)	RW	0 (0x00)
	8 (0x08)	CCW Angle Limit(L)	Ограничение угла при вращении против часовой стрелки (младший байт)	RW	255 (0xFF)
	9 (0x09)	CCW Angle Limit(H)	Ограничение угла при вращении против часовой стрелки (старший байт)	RW	3 (0x03)
	11 (0x0B)	the Highest Limit Temperature	Внутренний предел температуры	RW	70 (0x46)
	12 (0x0C)	the Lowest Limit Voltage	Нижний предел напряжения	RW	60 (0x3C)
	13 (0x0D)	the Highest Limit Voltage	Верхний предел напряжения	RW	140 (0xBE)
	14 (0x0E)	Max Torque(L)	Младший байт максимального момента	RW	255 (0xFF)
	15 (0x0F)	Max Torque(H)	Старший байт максимального момента	RW	3 (0x03)
	16 (0x10)	Status Return Level	Возвращение статуса пакета	RW	2 (0x02)
	17 (0x11)	Alarm LED	Светодиоды для предупреждения	RW	36 (0x24)
	18 (0x12)	Alarm Shutdown	Отключение предупреждения	RW	36 (0x24)
	24 (0x18)	Torque Enable	Включение / выключение момента	RW	0 (0x00)
	25 (0x19)	LED	Включение / выключение светодиода	RW	0 (0x00)

RAM	26 (0x1A)	CW Compliance Margin	Зона нечувствительности при вращении по часовой стрелке	RW	1 (0x01)
	27 (0x1B)	CCW Compliance Margin	Зона нечувствительности при вращении против часовой стрелки	RW	1 (0x01)
	28 (0x1C)	CW Compliance Slope	Зона снижения момента при достижении заданного положения при движении по часовой стрелке	RW	32 (0x20)
	29 (0x1D)	CCW Compliance Slope	Зона снижения момента при достижении заданного положения при движении против часовой стрелки	RW	32 (0x20)
	30 (0x1E)	Goal Position(L)	Младший байт заданной позиции	RW	-
	31 (0x1F)	Goal Position(H)	Старший байт заданной позиции	RW	-
	32 (0x20)	Moving Speed(L)	Младший байт заданной скорости	RW	-
	33 (0x21)	Moving Speed(H)	Старший байт заданной скорости	RW	-
	34 (0x22)	Torque Limit(L)	Младший байт ограничения момента	RW	ADD14
	35 (0x23)	Torque Limit(H)	Старший байт ограничения момента	RW	ADD15
	36 (0x24)	Present Position(L)	Младший байт текущей позиции	R	-
	37 (0x25)	Present Position(H)	Старший байт текущей позиции	R	-
	38 (0x26)	Present Speed(L)	Младший байт текущей скорости	R	-
	39 (0x27)	Present Speed(H)	Старший байт текущей скорости	R	-
	40 (0x28)	Present Load(L)	Младший байт текущей нагрузки	R	-
	41 (0x29)	Present Load(H)	Старший байт текущей нагрузки	R	-
	42 (0x2A)	Present Voltage	Текущее напряжение	R	-
	43 (0x2B)	Present Temperature	Текущая температура	R	-
	44 (0x2C)	Registered	Отмечает, зарегистрирована ли инструкция	R	0 (0x00)
	46 (0x2E)	Moving	Отмечает, происходит ли какое-либо	R	0 (0x00)

			движение		
	47 (0x2F)	Lock	Блокировка EEPROM	RW	0 (0x00)
	48 (0x30)	Punch(L)	Младший байт минимальной силы, прикладываемой для движения	RW	32 (0x20)
	49 (0x31)	Punch(H)	Старший байт минимальной силы, прикладываемой для движения	RW	0 (0x00)

2.3.3 Управление портами

Ниже приведен простейший пример программы управления портами. В данном случае идет управление портом *C*. После успешной компиляции проекта и загрузки полученного .hex файла при выполнении программы на микроконтроллере загорятся все светодиоды.

```
#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
DDRC  = 0x7F;
PORTC = 0x7E;
while (1)
{
int i;
for(i = 0; i <= 6; i++)
{
PORTC = ~(1<<i);
_delay_ms(250);
}
}
return 1;
}
```

2.3.4 Получение входного сигнала с кнопки

Сигналы от устройств, подключенных к микроконтроллеру, могут читаться посредством портов ввода/вывода. Код представленный ниже позволяет видеть статус нажатия кнопки. В зависимости от нажатой кнопки

загораются разные диоды на микроконтроллере. Чтобы получить входные значения с портов *D* и *E*, используются макрофункции *PIND* и *PINE* соответственно. *PIND* и *PINE* сопоставляются побитно друг с другом посредством операции «&».

```
	if(~PINE & BTN_UP)
	PORTC &= ~LED_MANAGE;
	else if(~PINE & BTN_DOWN)
	PORTC &= ~LED_AUX;
	else if(~PINE & BTN_LEFT)
	PORTC &= ~LED_PROGRAM;
	else if(~PINE & BTN_RIGHT)
	PORTC &= ~LED_PLAY;
	else if(~PIND & BTN_START)
	PORTC =
~(LED_BAT|LED_TxD|LED_RxD|LED_AUX|LED_MANAGE|LED_PROGRA
M|LED_PLAY);
	else PORTC =
LED_BAT|LED_TxD|LED_RxD|LED_AUX|LED_MANAGE|LED_PROGRAM|
LED_PLAY;
```

2.3.5 Последовательная передача данных

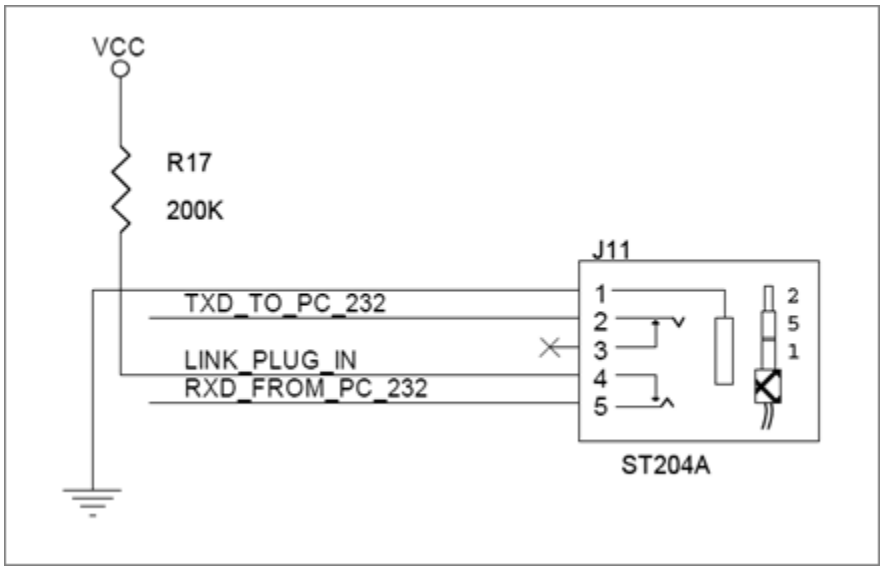


Рисунок 2.20 – Схема осуществления последовательной передачи данных

Последовательная связь является основным методом для передачи данных с микроконтроллера на ПК. Далее приведен пример осуществления последовательной связи.

```
    // инициализация USART (универсальный синхронно-асинхронный приемо-передатчик)
    serial_initialize(57600);
    sei(); // разрешение прерываний
```

В приведенной части функция *serial_initialize* включена в библиотеку serial.h, *sei()* – внутренняя команда, позволяющая использовать прерывания.

```
    unsigned char ReceivedData = getchar();
    if(ReceivedData == 'u')
    Value++;
    else if(ReceivedData == 'd')
    Value--;
    printf("%d\r\n", Value);
```

В данном случае выводится значение переменной *Value*, которая увеличивается на 1 при нажатии кнопки *u* и уменьшается на 1 при нажатии кнопки *d*. Данные получаются с использованием команды *getchar()*.

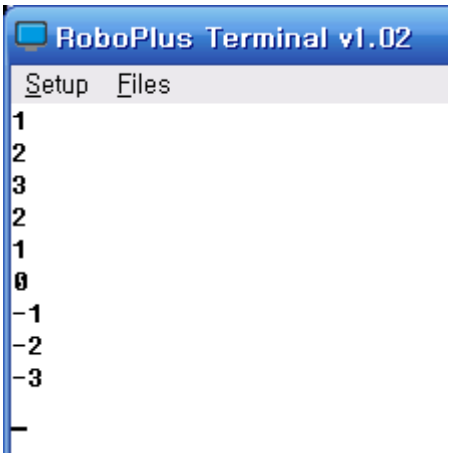


Рисунок 2.21 - Результат работы программы

2.3.6 Управление динамиком

Бипер может играть различные музыкальные ноты путем изменения частоты сигнала. Отношение музыкального размера и частоты приведено в таблице 2.3.

Таблица 2.3 Отношение музыкального размера и частоты для бипера микроконтроллера CM-510

Октава / Размер	1	2	3	4	5	6	7	8
C	32.7032	65.4064	130.8128	261.6256	523.2511	1046.502	2093.005	4186.009
C#	34.6478	69.2957	138.5913	277.1826	554.3653	1108.731	2217.461	4434.922
D	36.7081	73.4162	146.8324	293.6648	587.3295	1174.659	2349.318	4698.636
D#	38.8909	77.7817	155.5635	311.1270	622.2540	1244.508	2489.016	4978.032
E	41.2034	82.4069	164.8138	329.6276	659.2551	1318.510	2637.020	5274.041
F	43.6535	87.3071	174.6141	349.2282	698.4565	1396.913	2793.826	5587.652
F#	46.2493	92.4986	184.9972	369.9944	739.9888	1479.978	2959.955	5919.911
G	48.9994	97.9989	195.9977	391.9954	783.9909	1567.982	3135.963	6271.927
G#	51.9130	103.8262	207.6523	415.3047	830.6094	1661.219	3322.438	6644.875
A	55.0000	110.0000	220.0000	440.0000	880.0000	1760.000	3520.000	7040.000
A#	58.2705	116.5409	233.0819	466.1638	932.3275	1864.655	3729.310	7458.620
G	61.7354	123.4708	246.9417	493.8833	987.7666	1975.533	3951.066	7902.133

В приведенном ниже примере показана работа диодов и бипера в зависимости от нажатия кнопки. Пока нажата кнопка *Start* горят все светодиоды и бипер непрерывно играет некоторую ноту.

```
    if(~PIND & SW_START)
    {
    PORTC =
~(LED_BAT|LED_TxD|LED_RxD|LED_AUX|LED_MANAGE|LED_PROGRA
M|LED_PLAY);
    _delay_ms(1);
    PORTB |= 0x20;
    _delay_ms(1);
    PORTB &= ~0x20;
    }
    else
    {
    PORTC =
LED_BAT|LED_TxD|LED_RxD|LED_AUX|LED_MANAGE|LED_PROGRAM|
LED_PLAY;
    PORTB &= ~0x20;
    }
```

2.3.7 Работа с микрофоном

Если громкость звука превышает некоторый порог, то он может быть обнаружен с помощью микрофона. Код, представленный ниже, позволяет обнаружить звук, после чего загораются все светодиоды в течение 1 секунды.

```
    if(~PIND & MIC_SIGNAL)
    {
    PORTC =
~(LED_BAT|LED_TxD|LED_RxD|LED_AUX|LED_MANAGE|LED_PROGRA
M|LED_PLAY);
    _delay_ms(1000);
    }
    else PORTC =
LED_BAT|LED_TxD|LED_RxD|LED_AUX|LED_MANAGE|LED_PROGRAM|
LED_PLAY;
```

2.3.8 Работа с инфракрасным сенсором

Аналоговые сигналы могут быть преобразованы в цифровые значения с помощью микроконтроллера. Благодаря этому, могут быть получены и обработаны аналоговые величины напряжения внешнего ИК-датчика, датчика-гироскопа и т.д. В коде, приведенном ниже, представлен пример работы с ИК-датчиком, подключенному к порту 1.

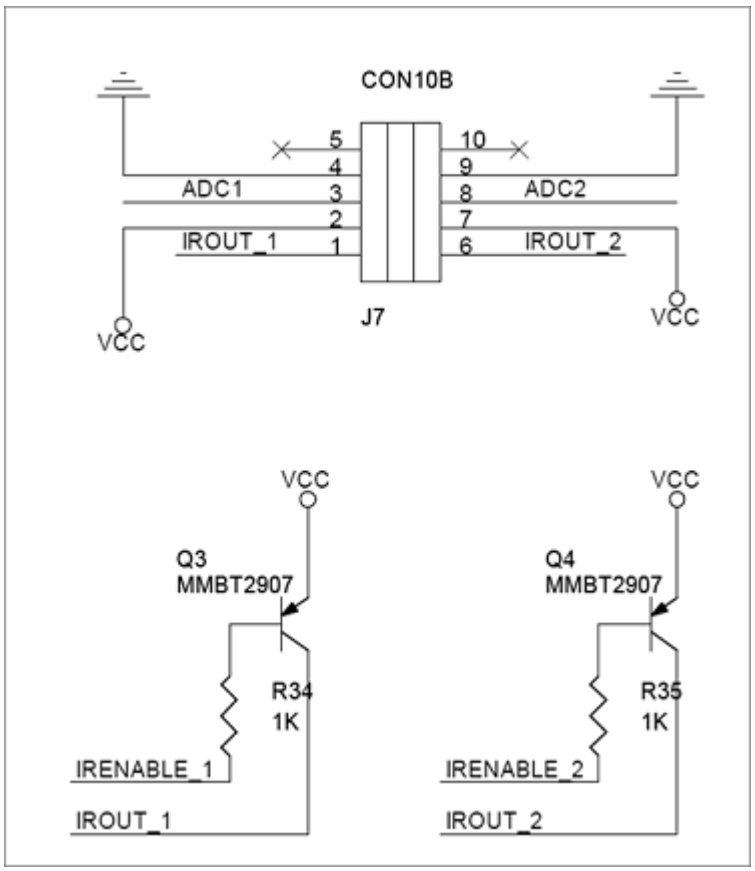


Рисунок 2.22 – Схема подключения ИК-сенсора

```
	serial_initialize(57600);
	sei();
	ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1); // Разрешение АЦП
	ADMUX = ADC_PORT_1; // Выбор порта 1 АЦП

	PORTA &= ~0x80; // Включение ИК-датчика

	_delay_us(12); // Короткая задержка для увеличения сигнала датчика
	ADCSRA |= (1 << ADIF); // Очистить флаг прерывания для АЦ преобразования
	ADCSRA |= (1 << ADSC); // Начало АЦ преобразования

	while( !(ADCSRA & (1 << ADIF)) ); // Ждать пока не завершится АЦ преобразование

	PORTA = 0xFC; // Выключение ИК-датчика

	printf( "%d\r\n", ADC); // Вывод значения на экран
	_delay_ms(50);
```

После включения ИК-датчика происходит преобразование аналогового сигнала в цифровой, а затем, после выключения датчика, на экран выводится значение сигнала.

2.3.9 Управление сервоприводом Dynamixel

Сервоприводом Dynamixel можно управлять с помощью передачи назначенного пакета. Управление положением серводвигателя осуществляется с помощью библиотеки dinamixel.h. Далее приведен пример управления сервоприводом с *ID 1*.

```
 unsigned short GoalPos[2] = {0, 1023};
serial_initialize(57600);
dxl_initialize( 0, 1 );
sei(); // Разрешить прерывания
```

dxl_initialize(int devIndex, int baudnum) позволяет инициализировать все подключенные устройства и привести их в режим готовности. *devIndex* – число подключенных на данный момент устройств, *baudnum* – скорость передачи данных (ниже приведена таблица основных скоростей передачи данных). Возвращаемое значение: 1 при успешной инициализации и 0 в противном случае.

Таблица 2.4 Основные скорости передачи данных

Адресс 4	Установленная скорость, bps	Целевая скорость, bps	Ошибка
1	1000000.0	1000000.0	0.000 %
3	500000.0	500000.0	0.000 %
4	400000.0	400000.0	0.000 %
7	250000.0	250000.0	0.000 %
9	200000.0	200000.0	0.000 %
16	117647.1	115200.0	-2.124 %
34	57142.9	57600.0	0.794 %
103	19230.8	19200.0	-0.160 %
207	9615.4	9600.0	-0.160 %

```
// Проверка выполнения движения
bMoving = dxl_read_byte( id, P_MOVING );
CommStatus = dxl_get_result();
if( CommStatus == COMM_RXSUCCESS )
{
if( bMoving == 0 )
{
// Изменение целевой позиции
if( index == 0 )
index = 1;
else
index = 0;
// Запись целевой позиции
dxl_write_word( id, P_GOAL_POSITION_L, GoalPos[index] );
}
PrintErrorCode();

// Чтение текущей позиции
wPresentPos = dxl_read_word( id, P_PRESENT_POSITION_L );
printf( "%d   %d\n",GoalPos[index], wPresentPos );
}
else
PrintCommStatus(CommStatus);
```

dxl_write_word(int id, int address, value) – функция движения серводвигателя к назначенной позиции; *id* – ID серводвигателя для записи информации; *address* – адрес информации; *value* – значение целевой позиции. Данная функция не возвращает никакого значения.

В итоге, при выполнении представленного примера, двигатель будет вращаться вперед и назад к назначенным позициям, а текущее положение будет выводиться на экран.

2.3.10 Синхронное управление двигателями

Далее приведен пример синхронного управления тремя двигателями с ID от 1 до 3. Приведенная ниже часть предназначена для расчета начального местоположения и инициализации местоположения каждого сервопривода Dynamixel.

```
int AmpPos = 512;
serial_initialize(57600);
dxl_initialize( 0, 1);
sei(); // Interrupt Enable
for( i=0; i<NUM_ACTUATOR; i++ )
{
id[i] = i+1;
phase[i] = 2*PI * (float)i / (float)NUM_ACTUATOR;
}
// Установка скорости
dxl_write_word( BROADCAST_ID, P_GOAL_SPEED_L, 0 );
// Установка назначенной позиции
dxl_write_word( BROADCAST_ID, P_GOAL_POSITION_L, AmpPos );
_delay_ms(1000);
```

Далее необходимо создать пакет инструкций для синхронного управления. Управление несколькими серводвигателями одновременно производится одной пакетной передачей инструкций. При использовании синхронного управления несколько команд передаются сразу, что сокращает время на сообщение, когда несколько серводвигателей находятся под контролем. Тем не менее, команда SYNC WRITE может использоваться, только если оба адреса и длины команд для записи из таблицы управления серводвигателем идентичны. Как правило, одна команда пакета занимает 4 байта, и при этом можно управлять 26 серводвигателями Dynamixel одновременно. При использовании синхронного управления необходимо убедиться, что длина пакета не превышает 143 байта, так как объем приемного буфера Dynamixel составляет 143 байт.

Пакет строится следующим образом:

ID 0XFE

Length (L+1) X N + 4 (L: длина данных на один серводвигатель, N: номер серводвигателя)

Instruction 0X83

Parameter1 Начальный адресс для записи данных

Parameter2 Длина данных для записи

Parameter3 ID первого серводвигателя

Parameter4 Первые данные первого серводвигателя

Parameter5 Вторые данные первого серводвигателя

…

Parameter L+3 L-ые данные первого серводвигателя

Parameter L+4 ID второго серводвигателя

Parameter L+5 Первые данные второго серводвигателя

Parameter L+6 Вторые данные второго серводвигателя

…

Parameter 2L+4 L-ые данные второго серводвигателя

Таблица 2.5 Пример синхронного управления:

Движения и скорость для каждого двигателя: **Серводвигатель с ID 0**: двигаться к позиции 0x010 со скоростью 0x150 **Серводвигатель с ID 1**: двигаться к позиции 0x220 со скоростью 0x360 **Серводвигатель с ID 2**: двигаться к позиции 0x030 со скоростью 0x170 **Серводвигатель с ID 3**: двигаться к позиции 0x220 со скоростью 0x380	**Пакет инструкций**: 0XFF 0XFF 0XFE 0X18 0X83 0X1E 0X04 0X00 0X10 0X00 0X50 0X01 0X01 0X20 0X02 0X60 0X03 0X02 0X30 0X00 0X70

В дальнейшем коде описывается создание пакета для синхроуправления:

```
// Создание пакета
dxl_set_txpacket_id(BROADCAST_ID);
dxl_set_txpacket_instruction(INST_SYNC_WRITE);
dxl_set_txpacket_parameter(0, P_GOAL_POSITION_L);
dxl_set_txpacket_parameter(1, 2);
for( i=0; i<NUM_ACTUATOR; i++ )
{
dxl_set_txpacket_parameter(2+3*i, id[i]);
```

```
GoalPos = (int)((sin(theta+phase[i]) + 1.0) * (float)AmpPos);
printf( "%d  ", GoalPos );
dxl_set_txpacket_parameter(2+3*i+1, dxl_get_lowbyte(GoalPos));
dxl_set_txpacket_parameter(2+3*i+2, dxl_get_highbyte(GoalPos));
}
dxl_set_txpacket_length((2+1)*NUM_ACTUATOR+4);

printf( "\n" );

dxl_txrx_packet();
CommStatus = dxl_get_result();
if( CommStatus == COMM_RXSUCCESS )
PrintErrorCode();
else
PrintCommStatus(CommStatus);

theta += STEP_THETA;
if( theta > 2*PI )
theta -= 2*PI;
_delay_ms(CONTROL_PERIOD);
```

После получения пакета при наличии ошибки выводится код ошибки. При выполнении приведенного примера сервоприводы двигаются по заданной траектории к назначенным позициям, при этом на экран выводится текущая позиция серводвигателей.

2.3.11 Удаленное управление посредством Zigbee передатчика

ZigBee — спецификация сетевых протоколов верхнего уровня (уровень программирования приложений (API) и сетевого уровня (NWK)), использующих сервисы нижних уровней — уровня управления доступом к среде (MAC) и физического уровня (PHY), регламентированных стандартом IEEE 802.15.4. ZigBee и IEEE 802.15.4 описывают беспроводные персональные вычислительные сети (WPAN). Спецификация ZigBee ориентирована на приложения, требующие гарантированной безопасной передачи данных при относительно небольших скоростях и возможности

длительной работы сетевых устройств от автономных источников питания [25].

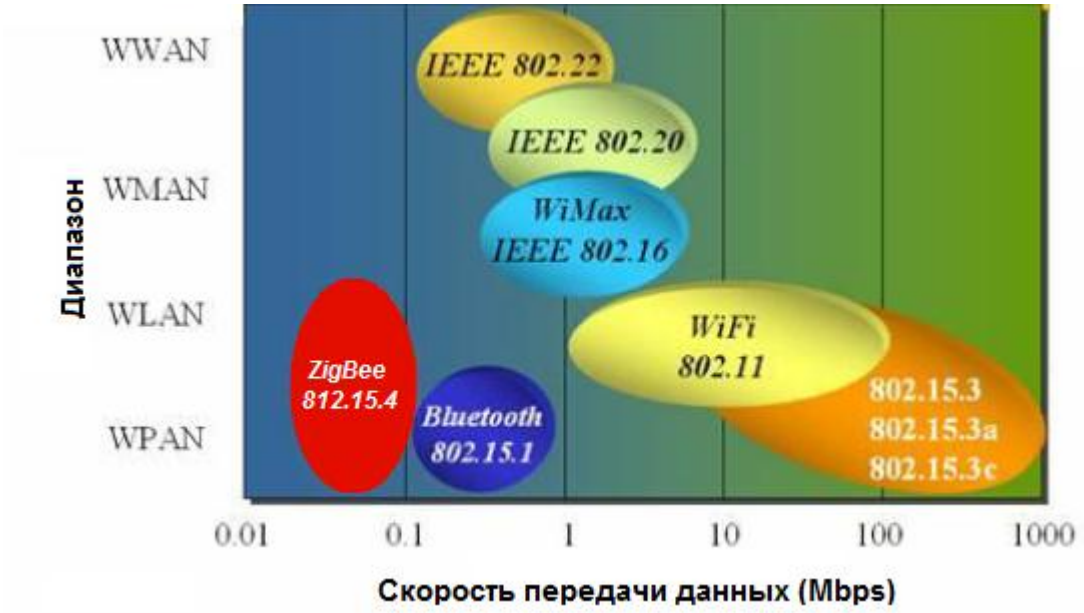


Рисунок 2.23 – Виды передачи данных

Основная особенность технологии ZigBee заключается в том, что она при малом энергопотреблении поддерживает не только простые топологии сети («точка-точка», «дерево» и «звезда»), но и самоорганизующуюся и самовосстанавливающуюся ячеистую (mesh) топологию с ретрансляцией и маршрутизацией сообщений. Кроме того, спецификация ZigBee содержит возможность выбора алгоритма маршрутизации, в зависимости от требований приложения и состояния сети, механизм стандартизации приложений — профили приложений, библиотека стандартных кластеров, конечные точки, привязки, гибкий механизм безопасности, а также обеспечивает простоту развертывания, обслуживания и модернизации. Применение сетей ZigBee в Российской Федерации в частотном диапазоне 2,405-2,485 ГГц не требует получения частотных разрешений и дополнительных согласований (Решение ГКРЧ при Мининформсвязи России от 07.05.2007 № 07-20-03-001).

Основными областями применения технологии ZigBee являются беспроводные сенсорные сети, автоматизация жилья («Умный дом» и «Интеллектуальное здание»), медицинское оборудование, системы

промышленного мониторинга и управления, а также бытовая электроника и «периферия» персональных компьютеров.

Далее приведен пример удаленного управления пультом управления RC100 посредством ZigBee передатчика. В данном примере используются порты C и D. Чтобы использовать ZigBee для начала необходимо инициализировать передатчик и настроить порт D:

```
PORTD &= ~0x80; //PORT_LINK_PLUGIN = 0;
PORTD &= ~0x20; //PORT_ENABLE_RXD_LINK_PC = 0;
PORTD |= 0x40; //PORT_ENABLE_RXD_LINK_ZIGBEE = 1;
zgb_initialize( 0 );
sei(); // Разрешение прерываний
```

Функция инициализации включена в Zigbee библиотеку. Zigbee инициализируется, если передается индекс устройства. По умолчанию индекс устройства 0.

```
if(zgb_rx_check() == 1)
{
RcvData = zgb_rx_data();
if(RcvData & RC100_BTN_1)
PORTC &= ~LED_MANAGE;
else
PORTC |= LED_MANAGE;
if(RcvData & RC100_BTN_2)
PORTC &= ~LED_PROGRAM;
else
PORTC |= LED_PROGRAM;
if(RcvData & RC100_BTN_3)
PORTC &= ~LED_PLAY;
else
PORTC |= LED_PLAY;
}
```

Данные, получаемые от ZigBee модуля, могут быть прочитаны с помощью функции zgb_rx_data(). Прием данных проверяется

командой zgb_rx_check. Значения, получаемые от пульта управления, приведены ниже в таблице 2.6.

Таблица 2.6 Значения, получаемые от пульта управления

Значение	Название кнопки
1	RC100_BTN_U - кнопка U
2	RC100_BTN_D - кнопка D
4	RC100_BTN_L - кнопка L
8	RC100_BTN_R - кнопка R
16	RC100_BTN_1 - кнопка 1
32	RC100_BTN_2 - кнопка 2
64	RC100_BTN_3 - кнопка 3
128	RC100_BTN_4 - кнопка 4
256	RC100_BTN_5 - кнопка 5
512	RC100_BTN_6 - кнопка 6

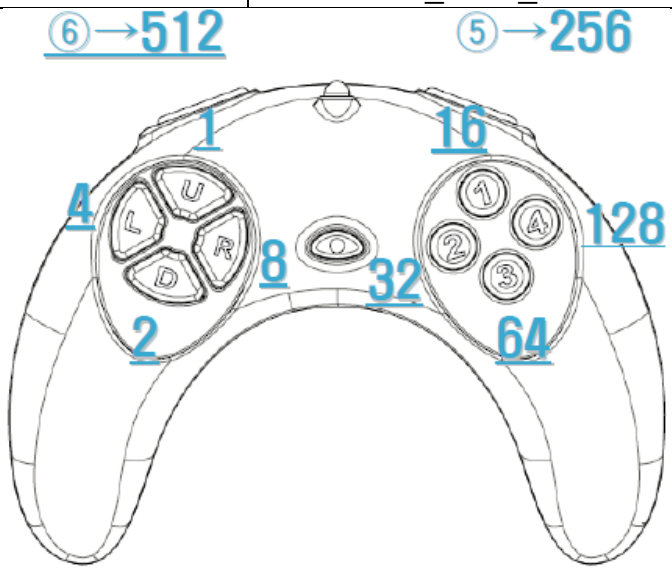


Рисунок 2.23 - Пульт управления RC100

В результате работы приведенного примера при нажатии копок на пульте управления на микроконтроллере СМ-510 загораются соответствующие светодиоды.

2.3.12 Пример организации ходьбы гуманоидного робота Bioloid по номинальной траектории

Цель: организация ходьбы гуманоидного робота Bioloid по номинальной траектории. Управление ведется по шести степеням свободы (см. рисунок 2.2): θ_{11}, θ_{12} (бедро), θ_{13}, θ_{14} (колено), θ_{15}, θ_{16} (ступня).

На рисунке 2.24 представлено расположение серводвигателей с присвоенными им номерами в конструкции гуманоидного робота на базе Bioloid. Такое расположение позволяет с большой точностью повторять роботу движения человека. Инструкция по сборке гуманоидного робота приведена поставляется вместе с набором Bioloid Premium Kit.

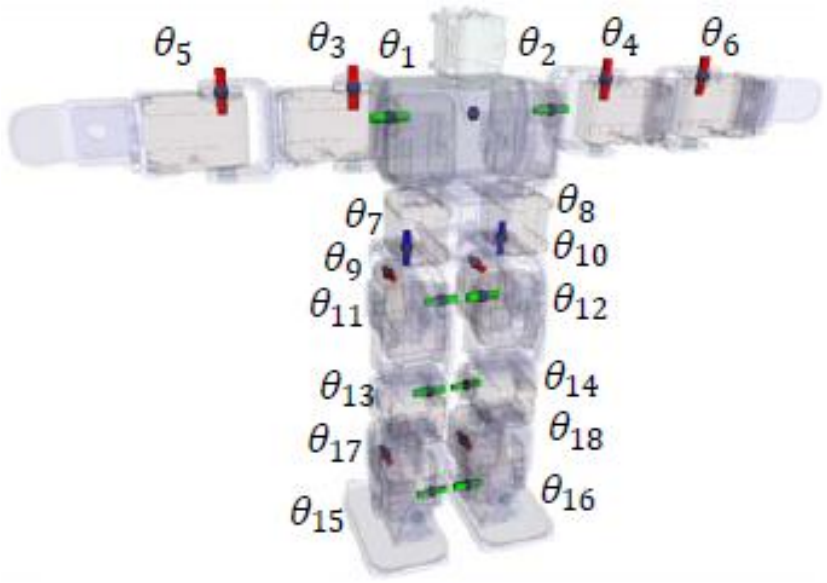


Рисунок 2.24 - Расположение серводвигателей

Содержание эксперимента

Для начала необходимо задать номинальную траекторию для каждого из двигателей. Сгибая ноги гуманоидного робота, необходимо записать значения угла каждого из двигателей, по которому ведется управление. Для этого можно использовать как RoboPlus Motion, так и программу написанную на языке С с использованием функции *dxl_read_word(int id, int address)*, где *int address=36*. Таким образом, вручную задается один цикл номинальной траектории ходьбы путем записи значений с энкодеров серводвигателей. Полученные значения можно записать в виде массива.

Начала кода программы с подключением библиотек и объявлением переменных можно записать следующим образом:

```
#include <avr/io.h>
```

```
#include <avr/interrupt.h>
#include <stdio.h>
#include <math.h>
#include <util/delay.h>

#include "serial.h"
#include "dynamixel.h"

/// Control table address
#define P_GOAL_POSITION_L       30
#define P_GOAL_POSITION_H       31
#define P_GOAL_SPEED_L          32
#define P_GOAL_SPEED_H          33
#define P_PRESENT_POSITION_L    36
#define P_PRESENT_POSITION_H    37

#define DEFAULT_BAUDNUM             1 // 1Mbps
#define NUM_ACTUATOR            6 // Number of actuator
#define CONTROL_PERIOD          (500) // msec

void PrintCommStatus(int CommStatus);
void PrintErrorCode(void);

int main(void)
{
        int id[NUM_ACTUATOR];
        int AmpPos = 512;
        int GoalPos, i, j, Error, l, CommStatus, wPresentPos;
        int u[NUM_ACTUATOR];
```

```
        const int k_hip = 57, k_kn=43, k_an=43;
        const int MaxSpeed = 1023;
        int buffer = 0;

        //Цикл походки для двигателей с ID 11-16
int IDPositions[6][28] = {{363,372,  373,  361,  341,  319,  299,  287,  283,
        286,  291,  293,  293,  297,  302,  308,  314,  321,  327,  332,  335,
        337,  338,  339,  340,  343,  347,  354},
        {721, 715,  709,  702,  696,  691,  688,  686,  685,  684,  683,  680,
        676,  669,  660,  651,  650,  662,  682,  704,  724,  736,  740,  737,
        732,  730,  730,  726},
        {254, 262,  262,  245,  222,  203,  196,  203,  222,  245,  262,  262,
        254,  248,  246,  248,  252,  257,  262,  266,  267,  266,  262,  257,
        252,  248,  246,  248},
        {777, 775,  771,  766,  761,  757,  756,  757,  761,  766,  771,  775,
        777,  775,  769,  761,  761,  778,  801,  820,  827,  820,  801,  778,
        761,  761,  769,  775},
        {675, 675,  677,  682,  685,  681,  669,  649,  627,  607,  595,  596,
        605,  614,  621,  625,  628,  629,  630,  631,  633,  636,  641,  647,
        654,  660,  666,  671},
        {402, 398,  395,  394,  393,  392,  390,  387,  382,  376,  369,  363,
        357,  352,  348,  348,  346,  341,  338,  342,  354,  374,  396,  416,
        428,  427,  418,  409}};

        serial_initialize(57600);
        dxl_initialize( 0, DEFAULT_BAUDNUM ); // Not using device index
        sei();  // Разрешение прерываний
```

Так как ID серводвигателей, по которым идет управление от 11 до 16, то:

```
for( i=0; i<NUM_ACTUATOR; i++ )
{id[i] = i+11;        }
```

Далее зададим роботу начальное положение из которого он начнет идти:

```
//        Set goal position
        dxl_write_word( 7, P_GOAL_POSITION_L, 358 );
        dxl_write_word( 8, P_GOAL_POSITION_L, 665 );
        dxl_write_word( 9, P_GOAL_POSITION_L, 511 );
        dxl_write_word( 10, P_GOAL_POSITION_L, 511 );
        dxl_write_word( 17, P_GOAL_POSITION_L, 511 );
        dxl_write_word( 18, P_GOAL_POSITION_L, 511 );
                _delay_ms(1000);
```

После описанного выше кода можно приступать к основной части программы. Организуем пропорциональное управление по каждому из двигателей:

```
while(1)
        {
                        for( j=0; j<28; j++ )  // Loop for walk position moving
                {
                        for( i=0; i<6; i++ )
                                {       wPresentPos = dxl_read_word( id[i], 36 );
                                        if (j==0) l=27;
                                        else l=j-1;
                                }

        // Серводвигатели в бедре
        for( i=0; i<2; i++ )
```

```
{
GoalPos = IDPositions[i][j];
wPresentPos = dxl_read_word( id[i], P_PRESENT_POSITION_L );
Error = wPresentPos - IDPositions[i][j];
buffer = k_hip * Error;
if(buffer < 0) buffer=-buffer;
if(buffer > MaxSpeed) buffer = MaxSpeed;
else if(buffer < 100) buffer = 100;
u[i] = buffer;
}

// Серводвигатели в коленях
                for( i=2; i<4; i++ )
                {
GoalPos = IDPositions[i][j];
wPresentPos = dxl_read_word( id[i], P_PRESENT_POSITION_L );
Error = wPresentPos - IDPositions[i][j];
buffer = k_kn * Error;
if(buffer < 0) buffer=-buffer;
if(buffer > MaxSpeed) buffer = MaxSpeed                    }

// Серводвигатели в голенях
for( i=4; i<6; i++ )
{
GoalPos = IDPositions[i][j];
wPresentPos = dxl_read_word( id[i], P_PRESENT_POSITION_L );
Error = wPresentPos - IDPositions[i][j];
buffer = k_an * Error;
if(buffer < 0) buffer=-buffer;
```

```
        if(buffer > MaxSpeed) buffer = MaxSpeed          }

                // Создание пакета синхрозаписи
                dxl_set_txpacket_id(BROADCAST_ID);
                dxl_set_txpacket_instruction(INST_SYNC_WRITE);
                dxl_set_txpacket_parameter(0, P_GOAL_POSITION_L);
                dxl_set_txpacket_parameter(1, 4);
                for( i=0; i<NUM_ACTUATOR; i++ )
                {
                GoalPos = IDPositions[i][j];
                dxl_set_txpacket_parameter(2+5*i, id[i]);
                dxl_set_txpacket_parameter(2+5*i+1, dxl_get_lowbyte(GoalPos));
                dxl_set_txpacket_parameter(2+5*i+2, dxl_get_highbyte(GoalPos));
                dxl_set_txpacket_parameter(2+5*i+3, dxl_get_lowbyte(u[i]));
                dxl_set_txpacket_parameter(2+5*i+4, dxl_get_highbyte(u[i]));
                                }
                dxl_set_txpacket_length((4+1)*NUM_ACTUATOR+4);
                dxl_txrx_packet();

                CommStatus = dxl_get_result();
                if( CommStatus == COMM_RXSUCCESS )
                        PrintErrorCode();
                else
                        PrintCommStatus(CommStatus);
                        _delay_ms(CONTROL_PERIOD);
                }
        }
        return 0;
}
```

Для вывода на экран состояния подключения и сообщения бита ошибки из статуса пакета используются функции *PrintCommStatus(int CommStatus)* и *PrintErrorCode()*:

```
// Вывод статуса подключения
void PrintCommStatus(int CommStatus)
{
	switch(CommStatus)
	{
	case COMM_TXFAIL:
		printf("COMM_TXFAIL: Failed transmit instruction packet!\n");
		break;
	case COMM_TXERROR:
		printf("COMM_TXERROR: Incorrect instruction packet!\n");
		break;
	case COMM_RXFAIL:
		printf("COMM_RXFAIL: Failed get status packet from device!\n");
		break;
	case COMM_RXWAITING:
		printf("COMM_RXWAITING: Now recieving status packet!\n");
		break;
	case COMM_RXTIMEOUT:
		printf("COMM_RXTIMEOUT: There is no status packet!\n");
		break;
	case COMM_RXCORRUPT:
		printf("COMM_RXCORRUPT: Incorrect status packet!\n");
		break;
	default:
		printf("This is unknown error code!\n");
		break;
```

```
        }
}

// Вывод кода ошибки
void PrintErrorCode()
{
        if(dxl_get_rxpacket_error(ERRBIT_VOLTAGE) == 1)
                printf("Input voltage error!\n");
        if(dxl_get_rxpacket_error(ERRBIT_ANGLE) == 1)
                printf("Angle limit error!\n");
        if(dxl_get_rxpacket_error(ERRBIT_OVERHEAT) == 1)
                printf("Overheat error!\n");
        if(dxl_get_rxpacket_error(ERRBIT_RANGE) == 1)
                printf("Out of range error!\n");
        if(dxl_get_rxpacket_error(ERRBIT_CHECKSUM) == 1)
                printf("Checksum error!\n");
        if(dxl_get_rxpacket_error(ERRBIT_OVERLOAD) == 1)
                printf("Overload error!\n");
        if(dxl_get_rxpacket_error(ERRBIT_INSTRUCTION) == 1)
                printf("Instruction code error!\n");
}
```

Результаты работы программы

Графики ошибки (отклонение от номинальной траектории) для каждого из двигателей приведены ниже на рисунках 2.25 – 2.30.

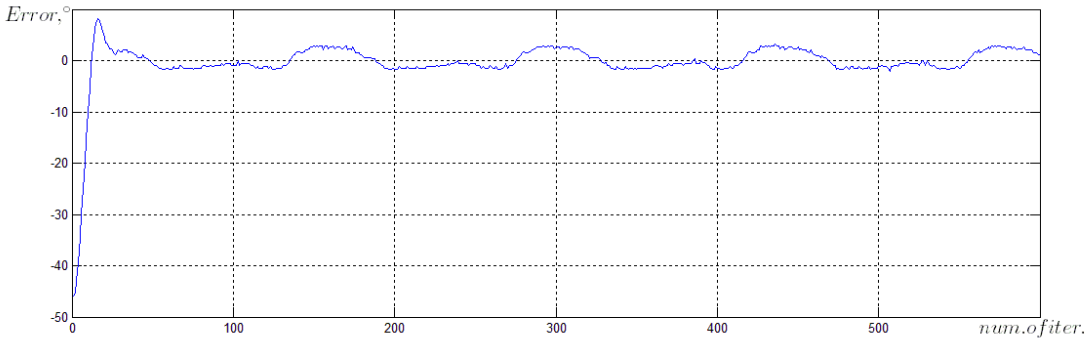


Рисунок 2.25 – Отклонение от номинальной траектории для серводвигателя №11 в бедре

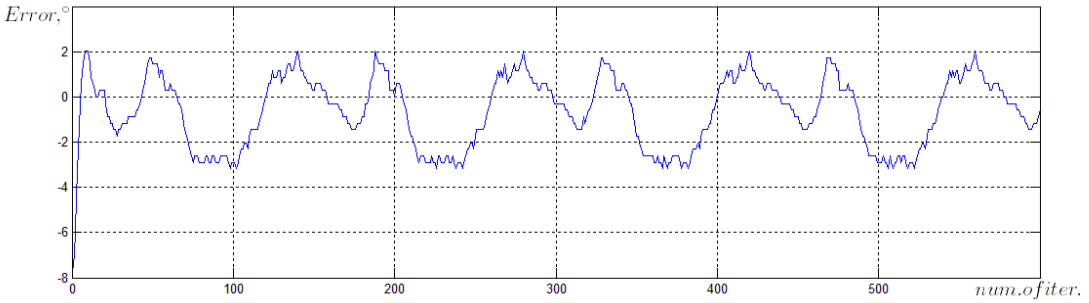


Рисунок 2.26 – Отклонение от номинальной траектории для серводвигателя №12 в бедре

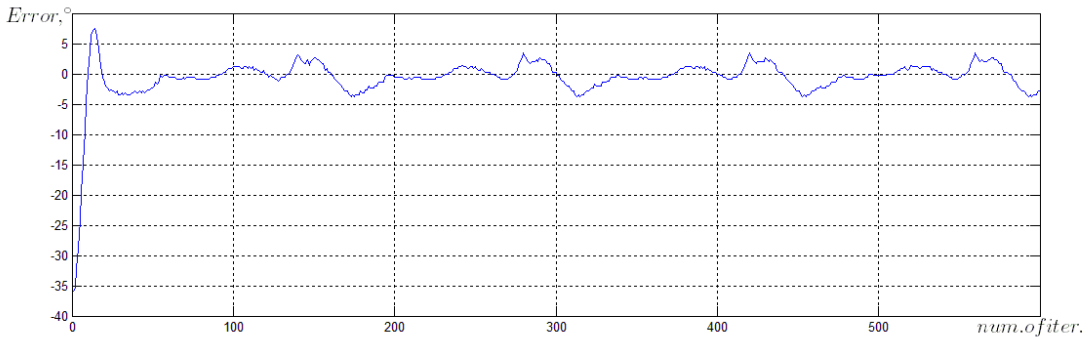


Рисунок 2.27 – Отклонение от номинальной траектории для серводвигателя №13 в колене

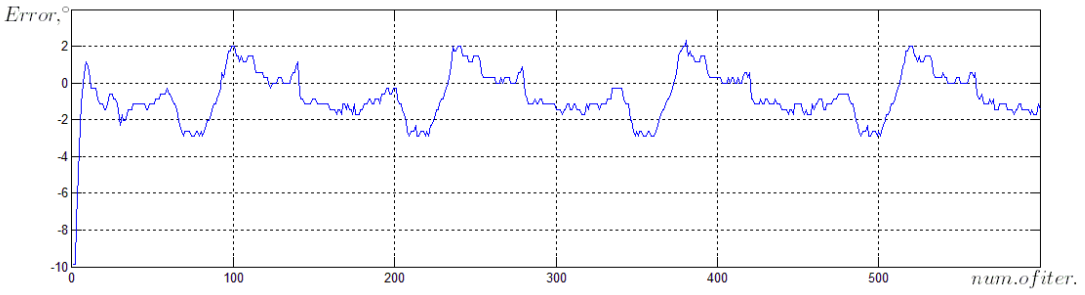


Рисунок 2.28 – Отклонение от номинальной траектории для серводвигателя №14 в колене

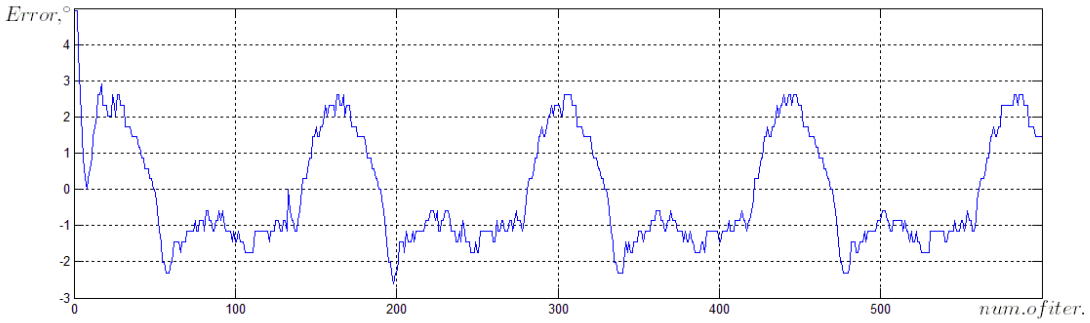


Рисунок 2.29 – Отклонение от номинальной траектории для серводвигателя №15 в ступне

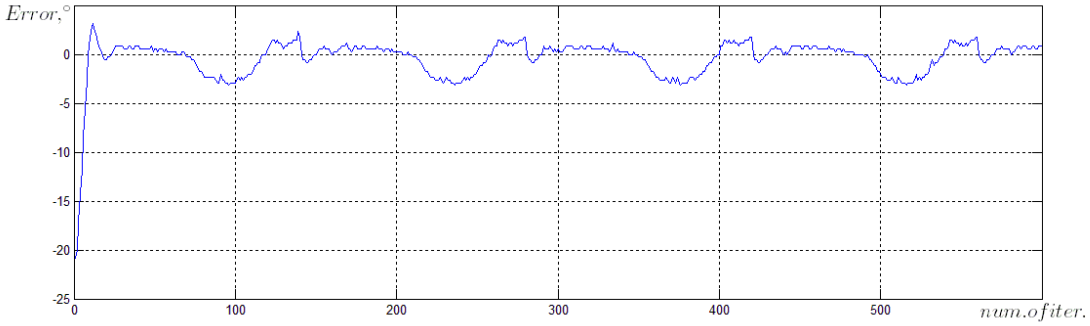


Рисунок 2.30 – Отклонение от номинальной траектории для серводвигателя №16 в ступне

Как видно из графиков отклонения от номинальной траектории устанавливаются в некотором пределе, а максимальная ошибка не превышает 4^0, что незначительно и походка робота остается устойчивой.

Таблица 2.7 Значение ошибок для походки

$e_{ср.кв}$	e_{max}
$e_{ср.кв1} = 1.62$ $e_{ср.кв2} = 1.6$ $e_{ср.кв3} = 1.62$ $e_{ср.кв4} = 1.36$ $e_{ср.кв5} = 1.47$ $e_{ср.кв6} = 1.37$	$e_{max1} = 3.19$ $e_{max2} = 3.19$ $e_{max3} = 3.77$ $e_{max4} = 2.9$ $e_{max5} = 2.61$ $e_{max6} = 3.19$

Протокол визуальных наблюдений за ходом данного эксперимента представлен на рисунке 2.31. В пояснениях к иллюстрации эксперимента указан хронометраж для каждого из пронумерованных снимков.

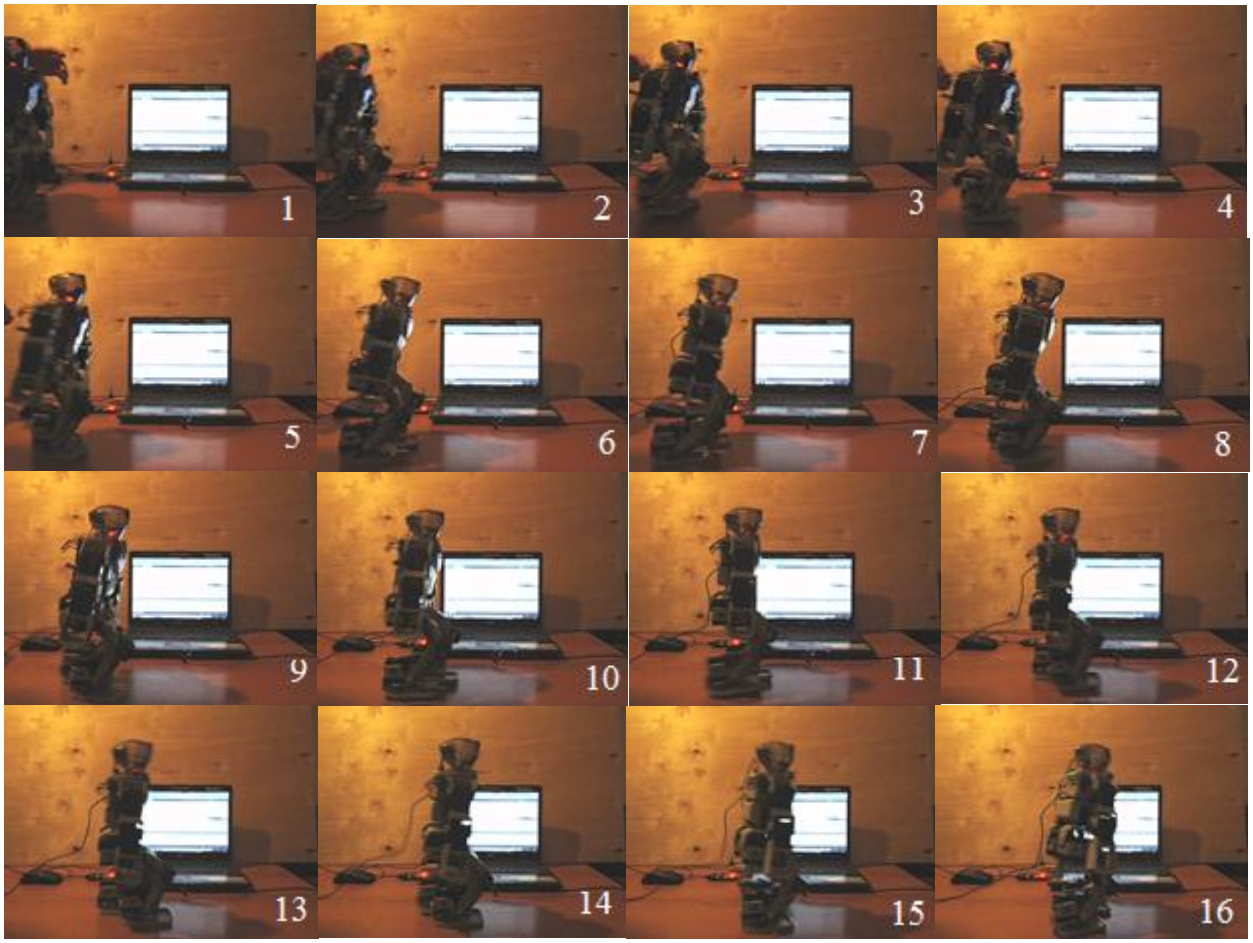


Рисунок 2.31 – Генерирование устойчивой походки: 1 – 0.12с; 2 – 0.25с; 3 – 0.37с; 4 – 0.48с; 5 – 0.6с; 6 – 0.72с; 7 – 0.84с; 8 – 0.95с; 9 – 1.09с; 10 – 1.2с; 11 – 1.31с; 12 – 1.44с; 13 – 1.56с; 14 – 1.56с; 15 – 1.68с; 16 – 1.8с; 17 – 1.90с; 18 – 2.02с

Для придания большей робастности созданной походки необходимо во время ходьбы уменьшить отклонения текущего положения центра масс от точки статической устойчивости. Этого можно достичь, например, введя управления серводвигателями в плечах робота θ_3 и θ_4 таким образом, что при движении вперед левой ноги левая рука движется назад, а правая рука – вперед, и наоборот.

Литература

1. Бобцов А.А., Капитанюк Ю.А., Капитонов А.А., Колюбин С.А., Пыркин А.А., Чепинский С.А., Шаветов С.В. Технология LEGO MINDSTORMS NXT в обучении студентов основам адаптивного управления // Научно-технический вестник СПбГУ ИТМО. 2011. № 1. С.103-108.
2. Information Control Problems in Manufacturing, V. 1 -14.
3. Bobtsov A.A., Pyrkin A.A., Borgul A.S., Zimenko K.A. Control Approaches for Complicated Self-Unstable Plants with Applications for Two-Wheel Mobile Robot Motobot in Educational Purposes // Preprints of the 9th IFAC Symposium on Advances in Control Education (ACE2012), Nizhny Novgorod, Russia,2012, pp. 107-111;
4. Bobtsov A.A., Kolyubin S.A., Pyrkin A.A., Borgul A.S., Zimenko K.A., Rabysh E.Y. Mechatronic and Robotic Setups forModern Control Theory Workshops // Preprints of the 9th IFAC Symposium on Advances in Control Education (ACE2012), Nizhny Novgorod, Russia,2012, pp. 348-353;
5. Бобцов А.А., Пыркин А.А., Боргуль А.С., Зименко К.А. Алгоритмы управления автономным двухколесным мобильным роботом «Мотобот» // Научно-технический вестник СПбГУ ИТМО, 2011, #75, сс. 63–68;
6. Боргуль А.С., Громов В.С., Зименко К.А., Маклашевич С.Ю. Система и алгоритмы стабилизации болбота // Научно-технический вестник СПбГУ ИТМО, 2011, #75, сс. 58–63;
7. Королькова М.А., Краснова С.А. Использование NXTOSEK и EMBEDDED CODER ROBOT NXT для программирования робота LEGO MINDSTORMS NXT // Доклады пятой международной конференции «Параллельные вычисления и задачи управления» PACO ‘2010, Москва. С.1489-1496.
8. Лучин Р.М. Программирование встроенных систем: от модели к роботу // СПб.: Наука, 2011. 184 с.
9. LEGO firmware. http://mindstorms.lego.com
10. leJOS. http://lejos.sourceforge.net
11. nxtOSEK. http://lejos-osek.sourceforge.net
12. OSEK. http://portal.osek-vdx.org
13. ECRobotInstaller. www.mathworks.com/matlabcentral/fileexchange/25207
14. Explorer. www.nxtprograms.com/NXT2/explorer/index.html
15. NXT GamePad http://lejos-osek.sourceforge.net/nxtgamepad.htm.
16. US Navy Academy to Acquire 50 Robotis Bioloid Humanoid Robot Kits from KumoTek http://www.azonano.com/news.aspx?NewsID=4498
17. Программирование робота Bioloid на языке C http://support.robotis.com/en/software/embeded_c/cm510_cm700.htm
18. ROBOTIS BIOLOID http://www.robotis.com/xe/BIOLOID_main_en
19. Benedettelli D., Casini M., Garulli A., Giannitrapani A., Vicino A.A. LEGO Mindstorms experimental setup for multi-agent systems // Control Applications, (CCA) & Intelligent Control, (ISIC), 2009 IEEE, pp. 1230-1235;

20. Brigandi S., Yunfeng Wang, Field J. A LEGO Mindstorms NXT based multirobot system // Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference, pp. 135-139;
21. Akhtaruzzaman M., Shafie A.A. Geometrical analysis on BIOLOID humanoid system standing on single leg // Mechatronics (ICOM), 2011 4th International Conference, pp. 1-5;
22. RoboCup. http://www.robocup.org;
23. RoboPlus - программное обеспечение для робота BIOLOID. http://www.wertech.ru/Blog/23-roboplus-bioloid.aspx;
24. Roboforum. http://roboforum.ru/forum67/topic11643.html#p245158;
25. Atmel Studio 6. http://www.atmel.com/microsite/atmel_studio6;
26. ZigBee. http://ru.wikipedia.org/wiki/ZigBee
27. Фомин В.Н., Фрадков А.Л., Якубович В.А. Адаптивное управление динамическими объектами. – М.: Наука. Главная редакция физико-математической литературы, 1981. 448 с.
28. Anderson B.D.O., Moore J.B. Optimal control: linear quadratic methods // Prentice-Hall, 1989. – 394 p.
29. Ha Y.-S., Yuta S. Trajectory tracking control for navigation of self-contained mobile inverse pendulum // In Proc. IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems, pages 1875-1882, 1994.
30. Linear-Quadratic-Regulator (LQR) design – MATLAB [Электронный ресурс]. – Режим доступа: http://www.mathworks.com/help/toolbox/control/ref/lqr.html, свободный. Яз. Англ.
31. Григорьев В.В. Аналитический синтез регуляторов на основе качественной устойчивости [Текст]: дис. ... д-ра техн. наук: 05.13.01 / В.В. Григорьев. ЛИТМО. – Л.: [б.н.], 1988. – 483 с.

ПРИЛОЖЕНИЕ

Инструкция по сборке модели «Мотобот»

Сборка заднего колеса







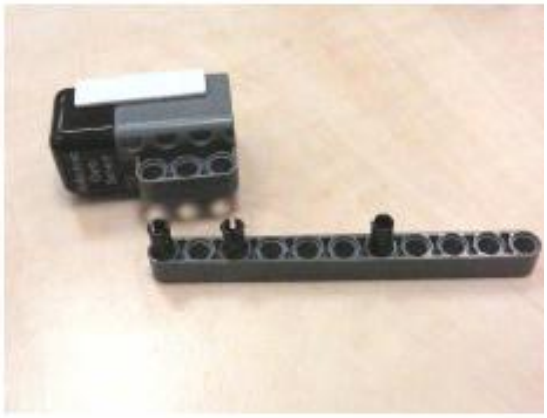

Сборка центральной части рамы

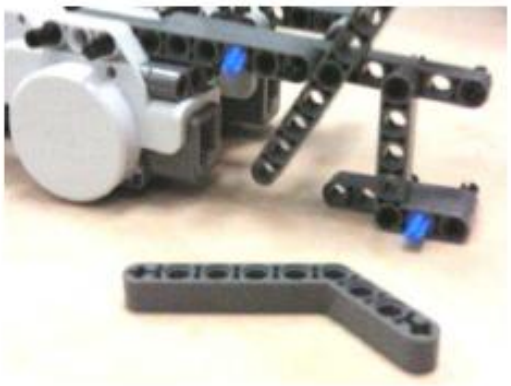


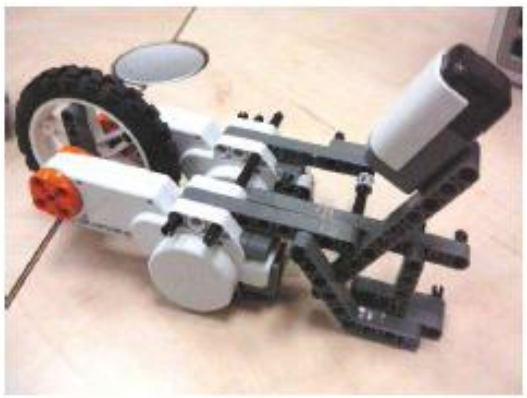





Сборка правой части рамы







Установка контроллера





Установка переднего колеса

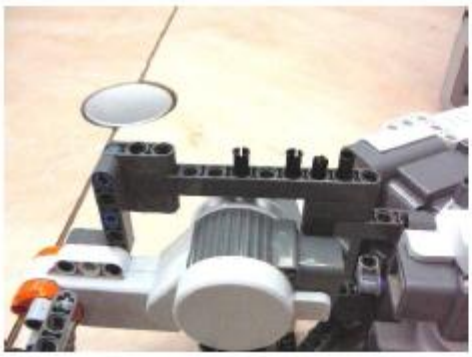
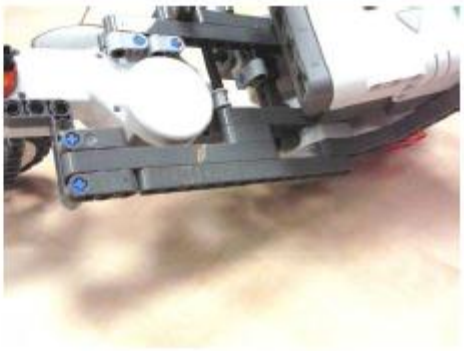


Сборка левой части рамы







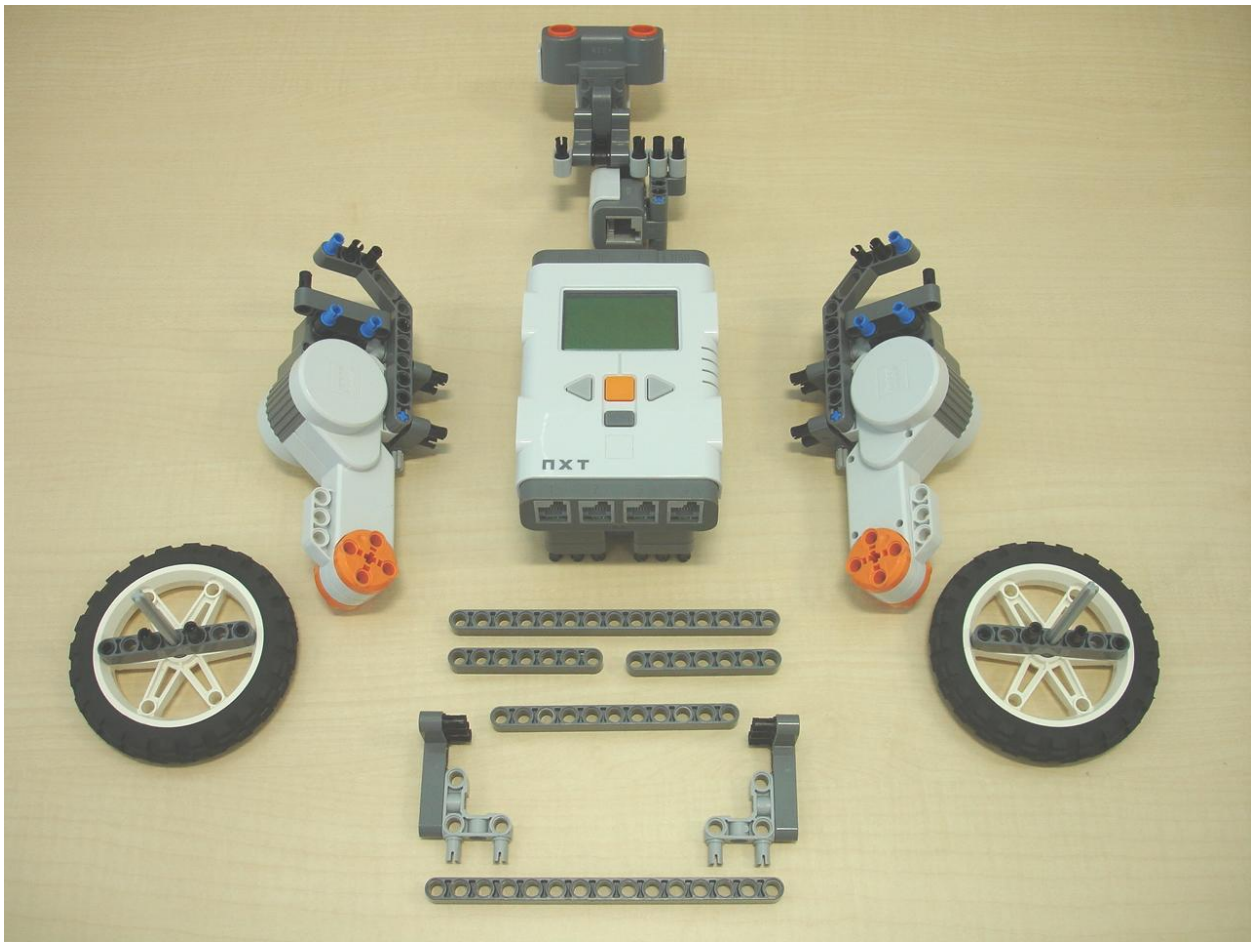
Сборка вилки рулевого колеса

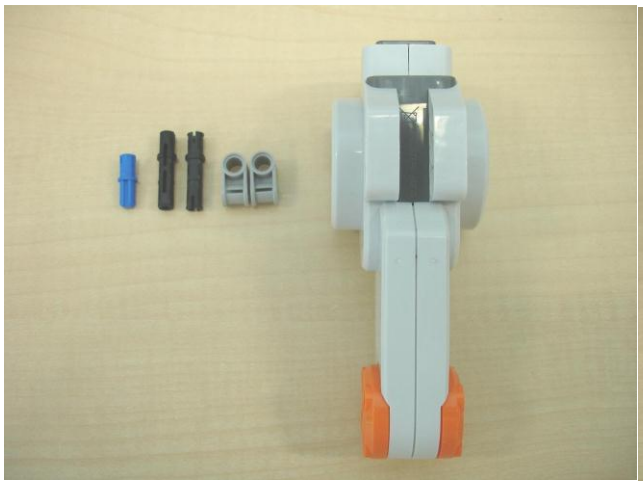


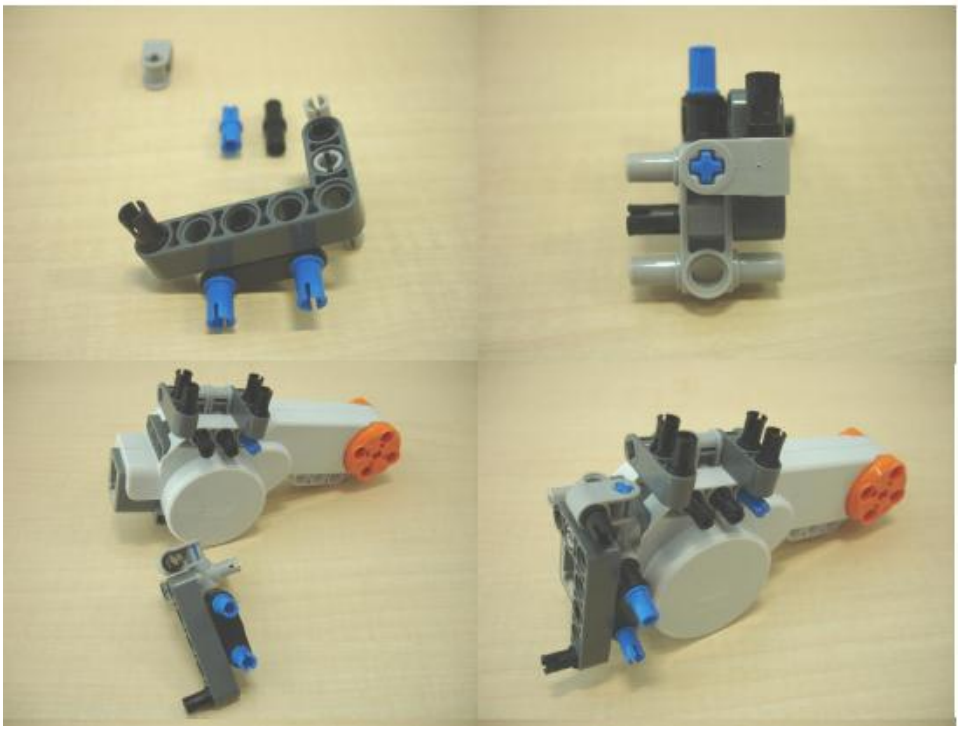


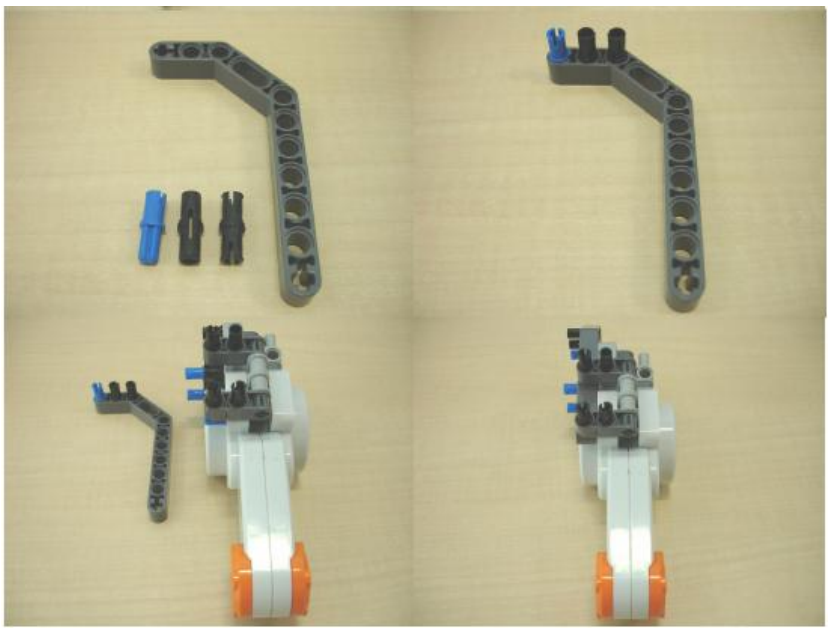


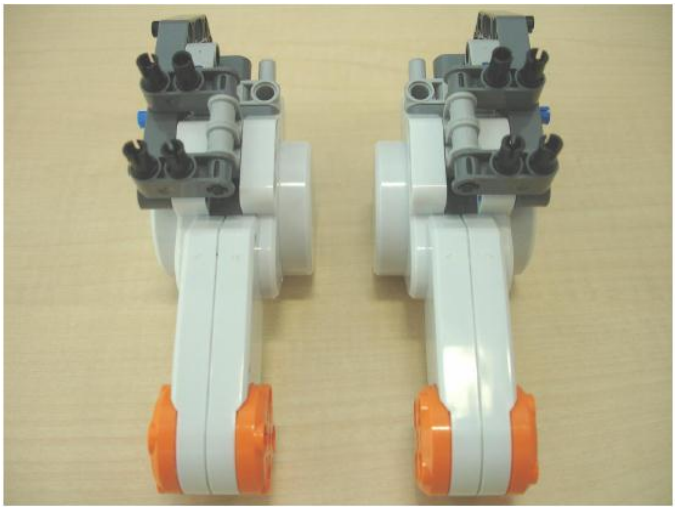

Инструкция по сборке модели «Segway»





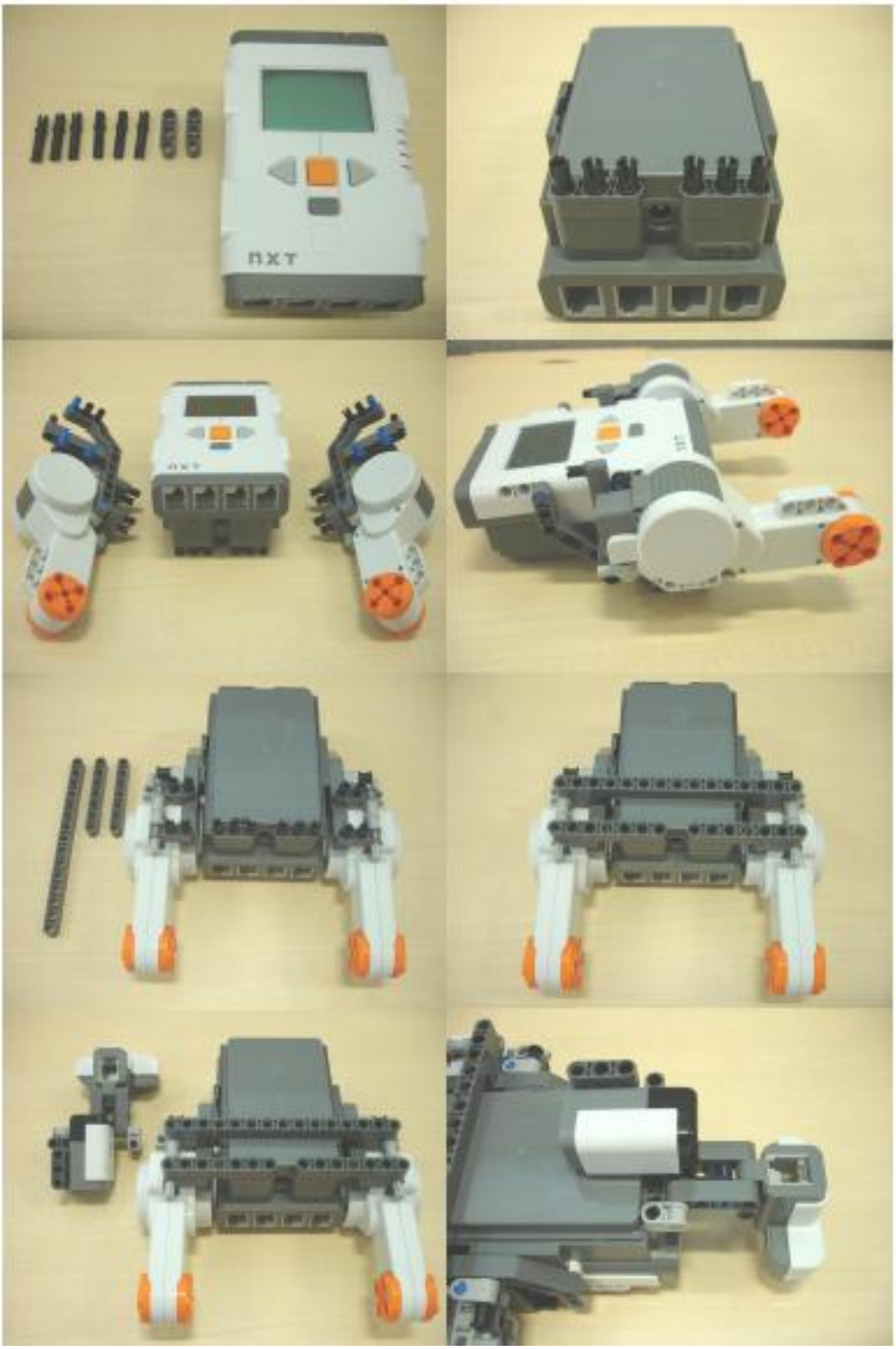






Сборка колес

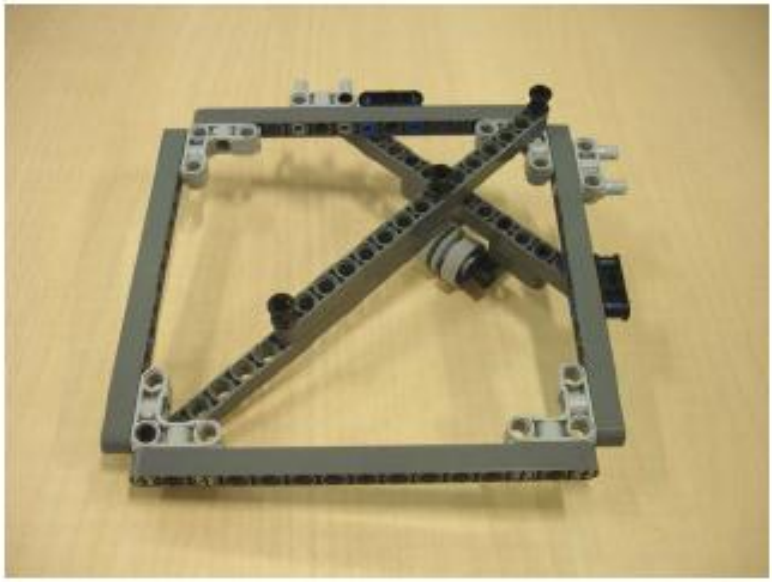
Сборка креплений для датчиков

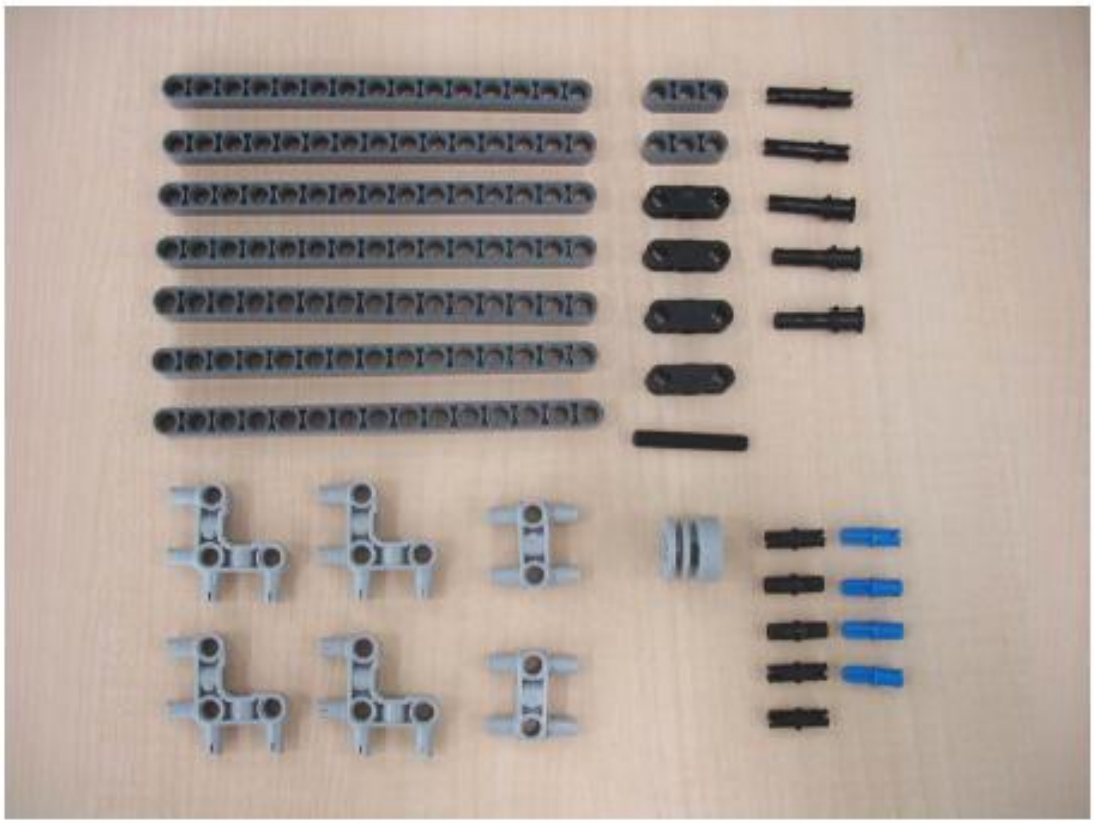

NXT

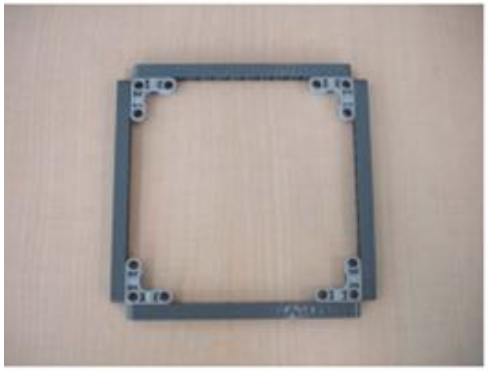

NXT
NXT
NXT

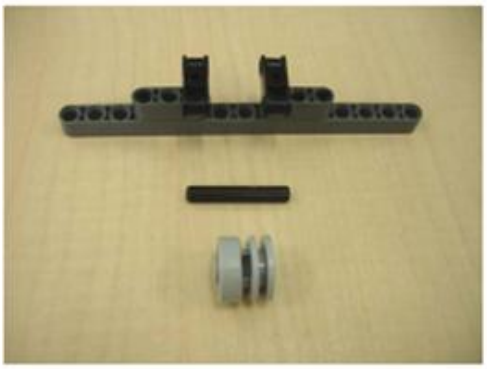
Инструкция по сборке модели «Болбот»

Сборка нижней части рамы



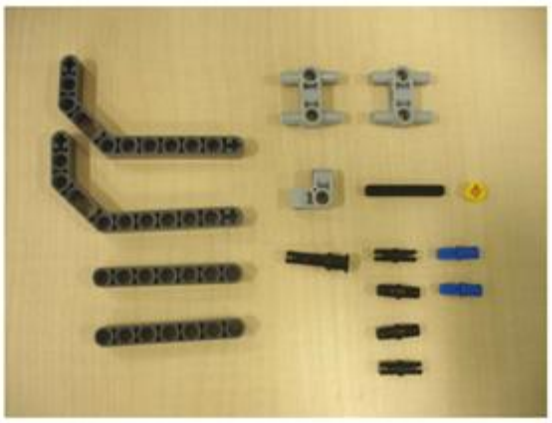




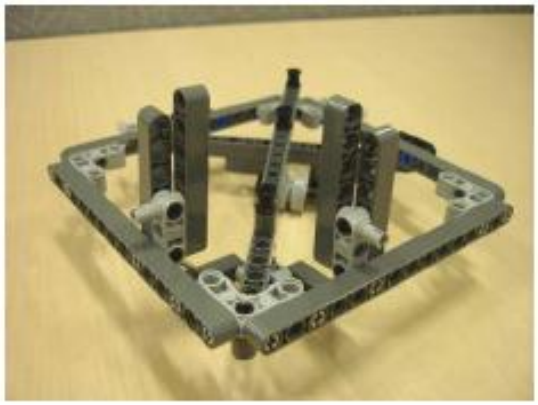





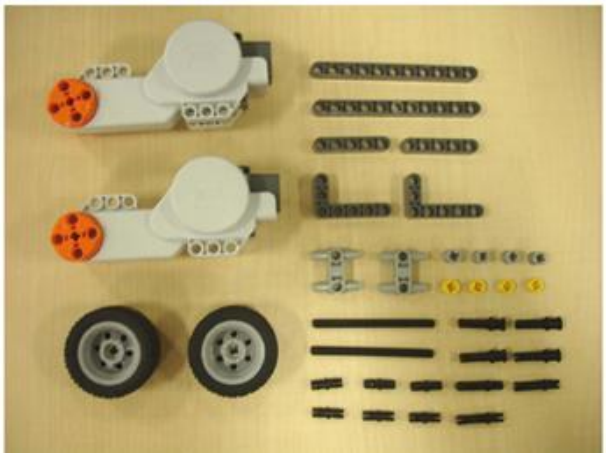
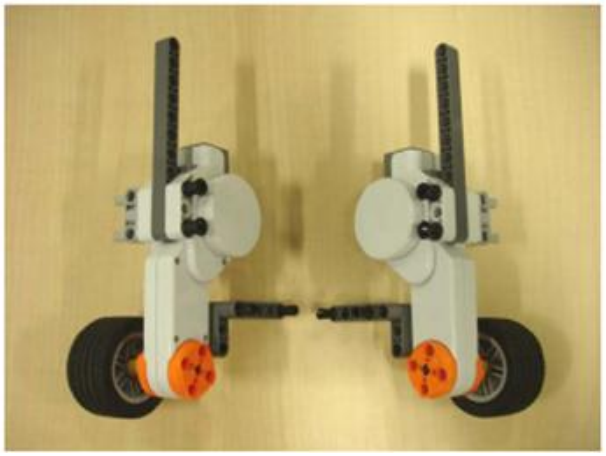

Сборка креплений для двигателей

Соединение креплений для двигателей и рамы

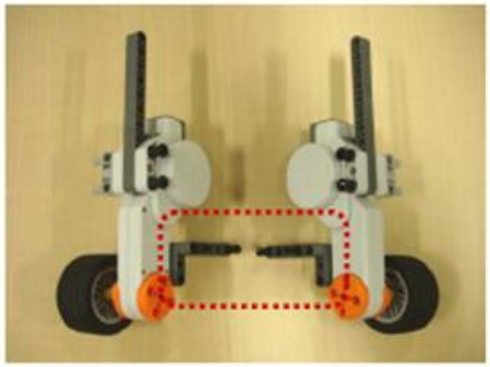


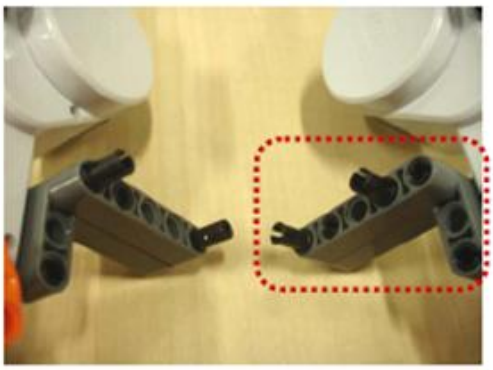
Подсоединение двигателей







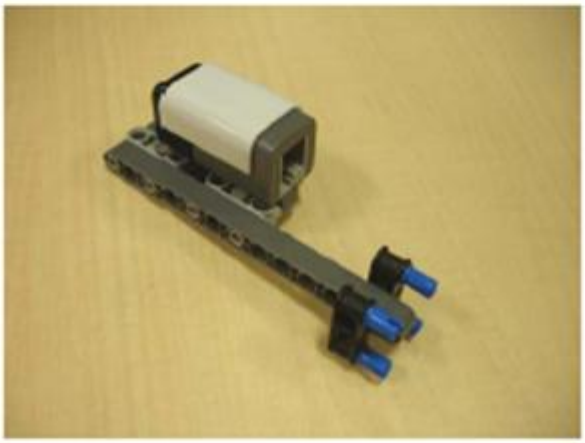


Крепление двигателей к раме

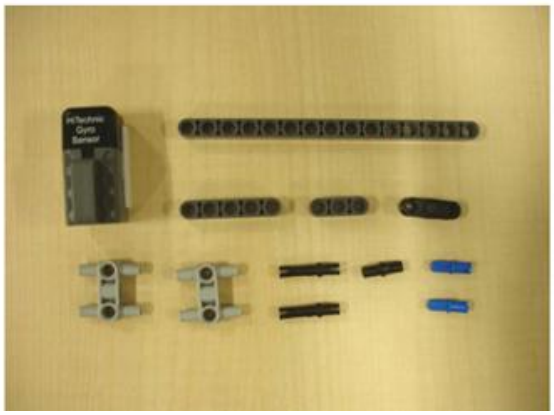
Сборка крепления гироскопа 1



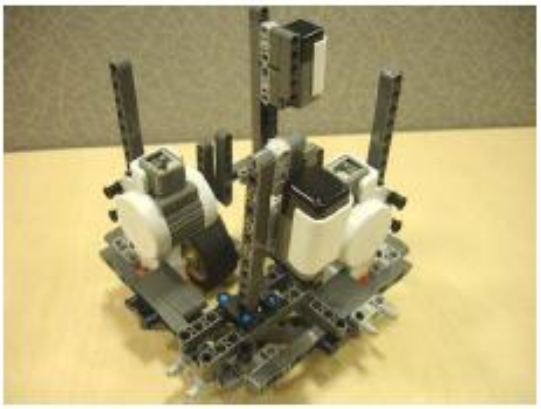
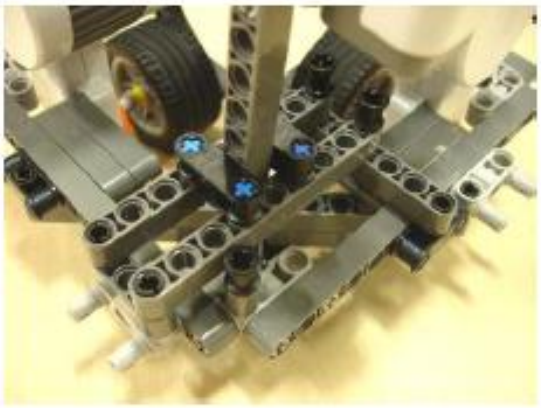


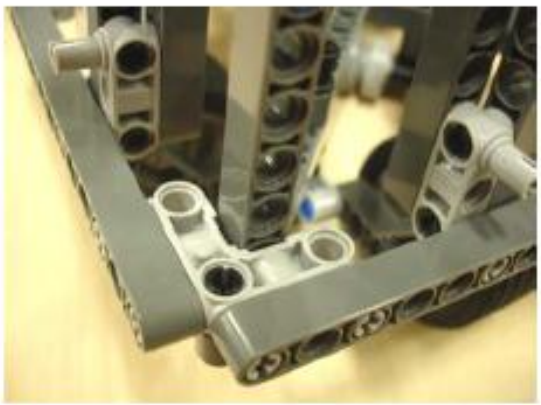



Сборка крепления гироскопа 2

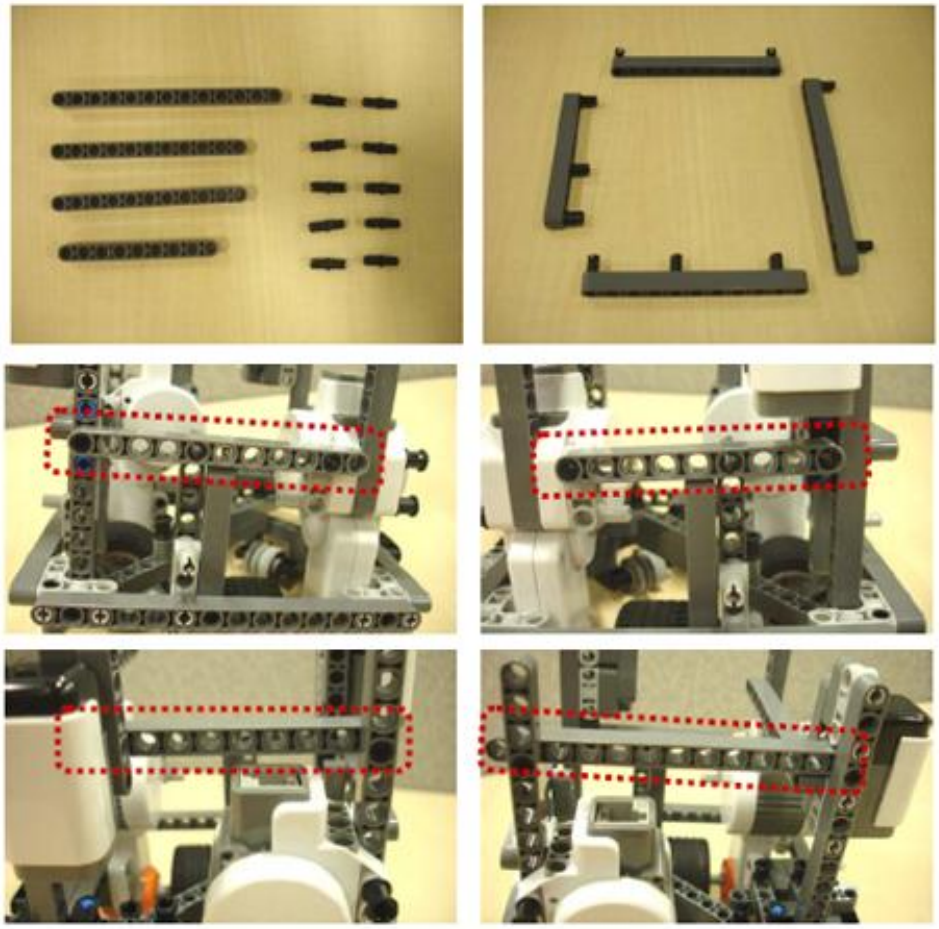



Подсоединение креплений гироскопов

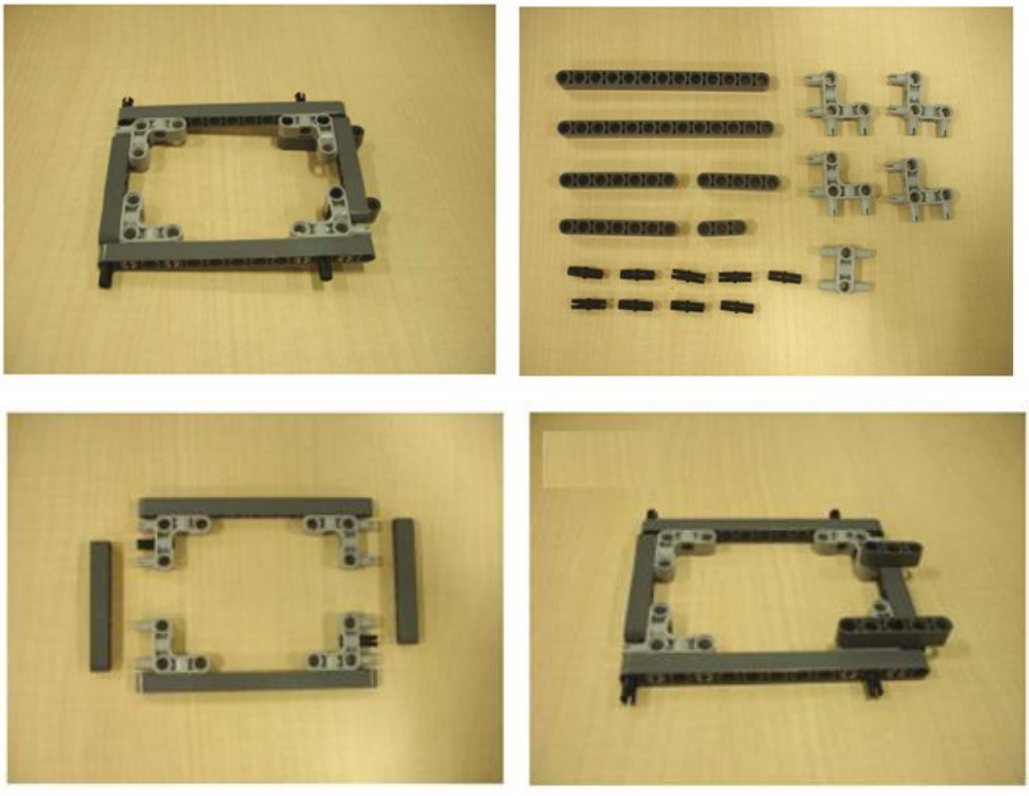





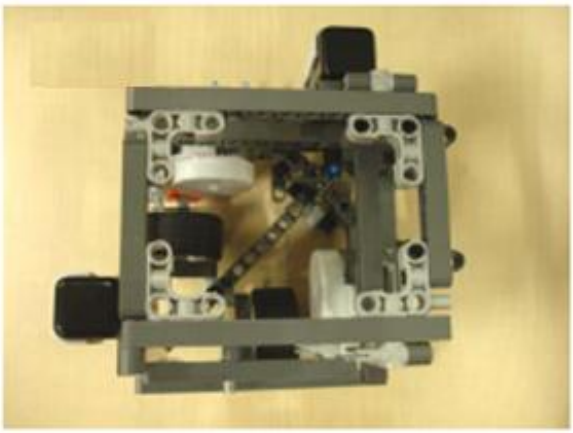
Крепление ребер жесткости

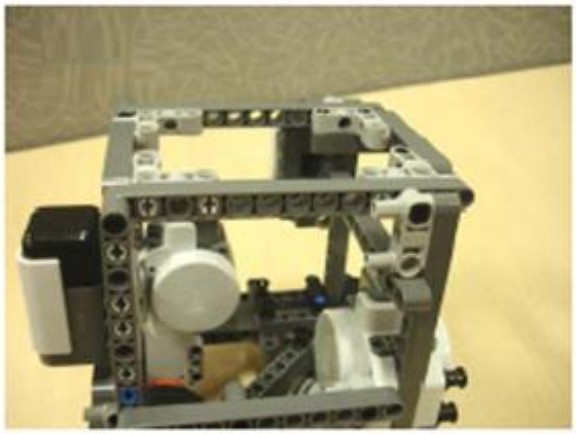


Сборка верхней части рамы



Крепление верхней части рамы

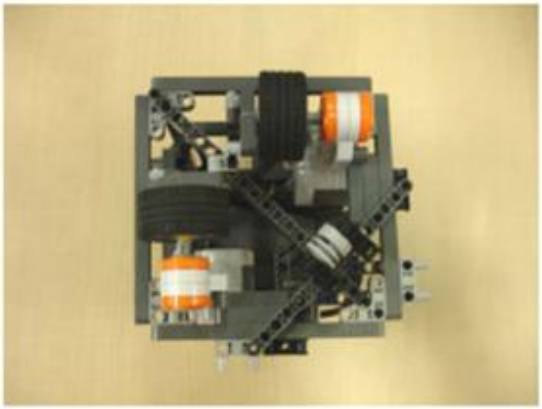




Подключение контроллера

Крепление контроллера к раме

Готовая модель



Запуск

-Нажмите желтую кнопку для включения контроллера

-Перейдите в папку ‘My files’ нажатием желтой кнопки

-Для начала работы программы NXTway-GS, нажмите на нее желтой кнопкой

-Нажмите на правую кнопку для запуска контроллера