Министерство науки и образования Российской федерации
Государственное образовательное учреждение высшего профессионального образования
«Томский государственный университет систем управления и радиоэлектроники»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ЛАБОРАТОРНЫМ ЗАНЯТИЯМ
«Программирование LEGO Mindstorms NXT на языке NXC»

Томск 2012 г.

Оглавление

Введение

Комплект LEGO Mindstorms NXT

LEGO Mindstorms – это конструктор 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.

Наборы LEGO Mindstorms комплектуются набором стандартных деталей LEGO (балки, оси, колеса, шестерни) и набором, состоящим из сенсоров, двигателей и программируемого блока. Наборы делятся на базовый набор и расширенный.

Базовый набор поставляется в двух версиях: версия для широкой продажи и базовый обучающий набор. Оба набора могут быть использованы для участия в соревнованиях робототехники (например во Всемирной олимпиаде роботов (англ. World Robot Olympiad)). Расширенный набор содержит большее количество деталей.

Для программирования робота Mindstorms можно использовать среду визуального программирования NXT-G, созданную компанией National Instrumensts (создатель LabView). Кроме этого, независимыми разработчиками созданы десятки средств и языков для программирования робота. Одним из наиболее популярных таких средств является язык NXC.

Язык NXC

Task main () – это точка входа в программу. Эта задача является главным последовательным потоком выполняемых команд.

Программа состоит из набора команд – операторов. Каждый оператор заканчивается точкой с запятой. Это позволяет понять, где кончается один оператор и начинается следующий.

Типы данных

Int – целочисленный тип данных диапазон значений: 0 … 65 535

Short, int - целочисленный тип данных диапазон значений: –32 768 … 32 767

Char – целочисленный тип данных, диапазон значений: –128 … 127

Byte – целочисленный тип данных, диапазон значений: 0 … 255

Float – вещественный тип данных, диапазон значений: $3.4*10^{-38}$ … $3.4*10^{38}$

Bool – логический тип данных, принимает значения true и false

Операторы

В языке NXC используется следующий набор операторов:

= оператор присваивания

== оператор сравнения «равно»

!= оператор сравнения «не равно»

> оператор сравнения «больше»

< оператор сравнения «меньше»

>= оператор сравнения «больше или равно»

<= оператор сравнения «меньше или равно»

- оператор арифметической операции «вычитание»

+ оператор арифметической операции «сложение»

* оператор арифметической операции «умножение»

/ оператор арифметической операции «деление»

% оператор арифметической операции «деление по модулю»

Управляющие структуры

if – конструкция ветвления

При использовании оператора `if` можно задать условие, при верности которого выполниться инструкция. При использовании оператора `else` можно добавить альтернативную инструкцию, которая будет выполнена, если условие не верно.

```
if (условие) инструкция
```

```
else  альтернативаная инструкция
```

Примеры использования оператора:

```
if (x==1) y = 2;
```

Y примет значение 2, если X равен единице. Если X не равен единице, ничего не произойдет.

```
if (x==1) y = 2; else y = 3;
```

При такой записи Y примет значение 2, если X равен единице. Если нет, то Y примет значение 3.

```
if (x==1) { y = 2; z = 2; }
```

В данной записи Y и Z примут значение 2, если X равен 1.

Switch – переключатель

Переключатель `switch` служит для выбора одного из многих вариантов. Если выражение не удовлетворяет ни одному варианту, то выполняются операторы, следующие за `default`.

```
switch(выражение)
{
     case 1:
          список операторов;
     break;
     case 2:
          список операторов;
     break;
     default:
          список операторов;
     break;
}
```

Пример использования:

```
switch( x )
{
     case 1:
          x = y;
     break;
     case 1:
```

```
            x = z;
      break;
      default:
            x = 0
}
```

Циклы – разновидность управляющей конструкции, предназначенная для организации многократного исполнения набора инструкций.

while

При использовании цикла `while` можно задать условие, пока оно верно операции в теле цикла будут выполняться.

```
while (условие)
{
тело цикла
}
```

Пример использования:

```
while(x < 10)
{
      x = x+1;
      y = y+1;
}
```

В данном примере X и Y будут увеличиваться на единицу, пока X не станет равен 10.

do

Цикл `do` очень похож на цикл `while` разница между ними заключается в том, что при использовании цикла `do` тело цикла предшествует условию, а значит выполнится минимум 1 раз.

```
do тело цикла while (условие)
```

for

Цикл `for` – это цикл со счетчиком, в котором присутствуют 3 выражения, первое – это инициализация (выполняется один раз, перед тем как войти в цикл), второе – это условие выполнения цикла. Третье – это приращение шага, после чего идет проверка условия выполнения цикла.

```
for (выражение; условие; выражение)
{
     тело цикла
}
```

Пример использования:

```
for (i = 0; i < 10; ++i)
{
     x = x+y;
}
```

В данном примере к X прибавится Y десять раз.

Функции

Общие функции.

Wait

Функция `Wait(время)` служит для создания задержек на определенное время, указанное в миллисекундах.

Пример использования:

```
Wait(1000); // ждать 1 секунду
```

Функции управления двигателем.

Двигатели подключаются к портам A,B,C. Для указания какого-либо порта используются константы: OUT_A,

- OUT_A – порт A
- OUT_B – порт B
- OUT_C – порт C

Для указания нескольких портов используются константы:

- OUT_AB – порт A,B
- OUT_AC – порт A,C
- OUT_BC – порт B,C
- OUT_ABC – порт A,B,C

Off

Функция `Off(порты)` останавливает двигатель или двигатели. В зависимости от того, какие указаны порты.

Пример использования:

```
Off(OUT_AB)
```

OnFwd

Функция `OnFwd(порты,мощность)` заставляет вращаться двигатели с разной скоростью, в зависимости от указанной мощности, от -100 до 100. При указании отрицательного значения двигатель будет вращаться назад.

Пример использования:

```
OnFwd(OUT_A, 75)
```

OnRev

Функция `OnRev(порты,мощность)` заставляет вращаться двигатели с разной скоростью, в зависимости от указанной мощности, от -100 до 100. При указании отрицательного значения двигатель будет вращаться вперед.

Пример использования:

```
OnRev(OUT_BC, 75)
```

RotateMotor

С помощью функции `RotateMotor (порты, мощность, градус)` можно вращать двигатель на определенный градус.

Пример использования:

```
RotateMotor (OUT_BC, 20, 73); // вращение вперед
RotateMotor (OUT_BC, -20, 73); // вращение назад
```

RotateMotorEx

Функции `RotateMotorEx(порты, мощность, угол, распределение мощности по двигателям, синхронизация, остановка)` имеет больше возможностей, чем функция `RotateMotor`, с её помощью можно распределить мощность по двигателям, синхронизировать работу моторов, остановить двигатели в конце вращения на определенный угол.

Примеры использования:

```
RotateMotorEx(OUT_AB, 75, 360, 50, true, true);
```

// движение по дуге налево, после чего остановка.

```
RotateMotorEx(OUT_AB, 75, 360, 0, true, false);
```

// движение прямо.

```
RotateMotorEx(OUT_AB, 75, 360, -100, true, true);
```
// повернуть направо на месте.

Функции инициализации датчиков.

Датчики подключаются к портам, обозначенными цифрами от 1 до 4. Для указания какого-либо порта используют константы:

- S1 – первый порт
- S2 – второй порт
- S3 – третий порт
- S4 – четвертый порт

SetSensorLight

Функция `SetSensorLight(порт)` инициализирует датчик, подключенный к указанному порту. Датчик будет использоваться как датчик освещенности.

Пример использования:

```
SetSensorLight(S2)
```

SetSensorTouch

Функция `SetSensorToch(порт)` инициализирует датчик, подключенный к указанному порту. Датчик будет использоваться как датчик касания.

Пример использования:

```
SetSensorToch(S1)
```

SetSensorLowspeed

Функция `SetSensorLowspeed(порт)` инициализирует датчик, подключенный к указанному порту. Датчик будет использоваться как низкоскоростной датчик.

Пример использования:

```
SetSensorLowspeed(S4)
```

SetSensorSound

Функция `SetSensorSound(порт)` инициализирует датчик, подключенный к указанному порту. Датчик будет использоваться как датчик звука.

Пример использования:

```
SetSensorSound(S1)
```

В наборе имеется датчик цвета, его можно использовать как датчик освещенности, но также с помощью него можно определять цвета.

Для использования этого датчика в качестве датчика освещенности, инициализировать его необходимо следующим образом:

```
SetSensorColorRed(порт);
//будет использоваться красная подсветка
SetSensorColorBlue(порт);
//будет использоваться синяя подсветка
SetSensorColorGreen(порт);
//будет использоваться зеленая подсветка
```

Если необходимо использовать данный датчик для определения цвета, то инициализацию стоит производить следующим образом:

```
SetSensorColorFull(порт); //будет использоваться RGB подсветка
```

После того, как датчики был инициализированы, с них можно считать текущие значения:

```
x = SENSOR_1
```

Проигрывание звуков.

Для проигрывания звуков существует несколько функций:

PlayTone

При использовании функции `PlayTone(частота,длительность)` воспроизводится звук заданной частоты и длительности, частота указывается в герцах, длительность в микросекундах.

Пример использования:

```
PlayTone(440, 500)
```

PlayFile

При использовании функции `PlayFile(имя файла)` воспроизводится указанный файл. Файл должен быть с разрешением .rso или .rmd.

Пример использования:

```
PlayFile(sound.rso)
```

Работа с дисплеем.

NumOut

Функция `NumOut(координата X,координата Y, переменная)` выводит на дисплей число в указанной координате дисплея.

Пример использования:

```
NumOut(10, 10, x)
```

TextOut

Функция `TextOut(координата X,координата Y, "Выводимый текст")` выводит на дисплей текст в указанной координате дисплея.

Пример использования:

```
TextOut(10, 10, "Hello")
```

CircleOut

Функция `CircleOut(координата X,координата Y, радиус)` позволяет отобразить на дисплее круг, заданного радиуса.

Пример использования:

```
CircleOut(40, 40, 10)
```

LineOut

Функция `LineOut(координата X1,координата Y1, координата X2,координата Y2)` позволяет отобразить на дисплее линию, началом которой служат координаты X1, Y1, а концом координаты X2, Y2.

Пример использования:

```
CircleOut(40, 40, 10)
```

PointOut

Функция `PointOut(координата X,координата Y)` позволяет отобразить на дисплее точку на координате X, Y.

Пример использования:

```
PointOut(40, 40)
```

RectOut

Функция `RectOut(координата X,координата Y, длина, высота)` позволяет отобразить на дисплее прямоугольник с началом в координате X, Y с заданной длиной и высотой.

Пример использования:

```
RectOut(40, 40, 30, 10)
```

ClearScreen

Функция `ClearScreen()` позволяет очистить экран.

Функция для получения случайных чисел

Random

Функция `Random(n)` служит для получения случайного числа в диапазоне от 0 до n.

Пример использования:

```
x = Random(10);
```

// возвратит число в диапазоне от 0 до 9

Создание собственных функций

Создание собственных функций необходимо для облегчения написания программ. Для создания функций используется конструкция:

```
void название_функции(входные параметры) {
   операторы;
   функции;
   циклы;
}
```

Входные параметры могут отсутствовать.

Комментарии

Для написания комментариев используется символ `//` весь текст находящийся в этой строке после него компилятором не воспринимается.

Интегрированая среда программирования и отладки BricxCC

Для написания программ используется язык NXC – это язык программирования. Для написания программ существует среда программирования Bricx Command Center (BricxCC). С помощью неё можно писать программы, загружать их в робота, запускать и останавливать их и многое другое. BricxCC работает по большей части как текстовый редактор, но с некоторыми дополнительными возможностями.

Для программирования используется BricxCC. При запуске программы появится окошко, требующее подключиться к NXT Brick. Следует подключить NXT Brick к компьютеру с помощью USB кабеля и выбрать параметры, изображенные на рис. 1.

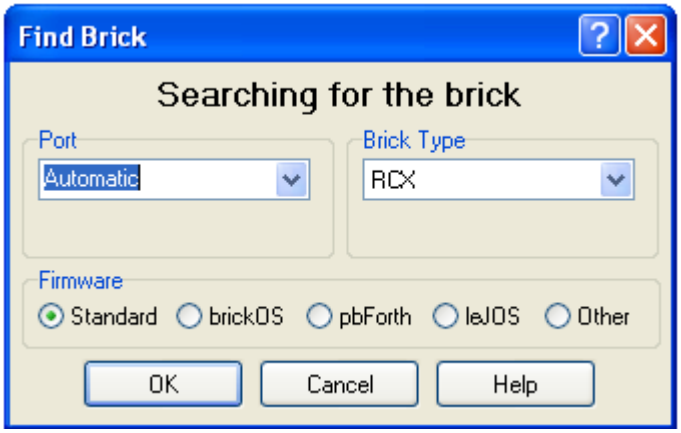

Рисунок 1 — Окно подключения к NXT Brick.

После подключения NXT Brick появится интерфейс пользователя, изображенный на рис. 2.

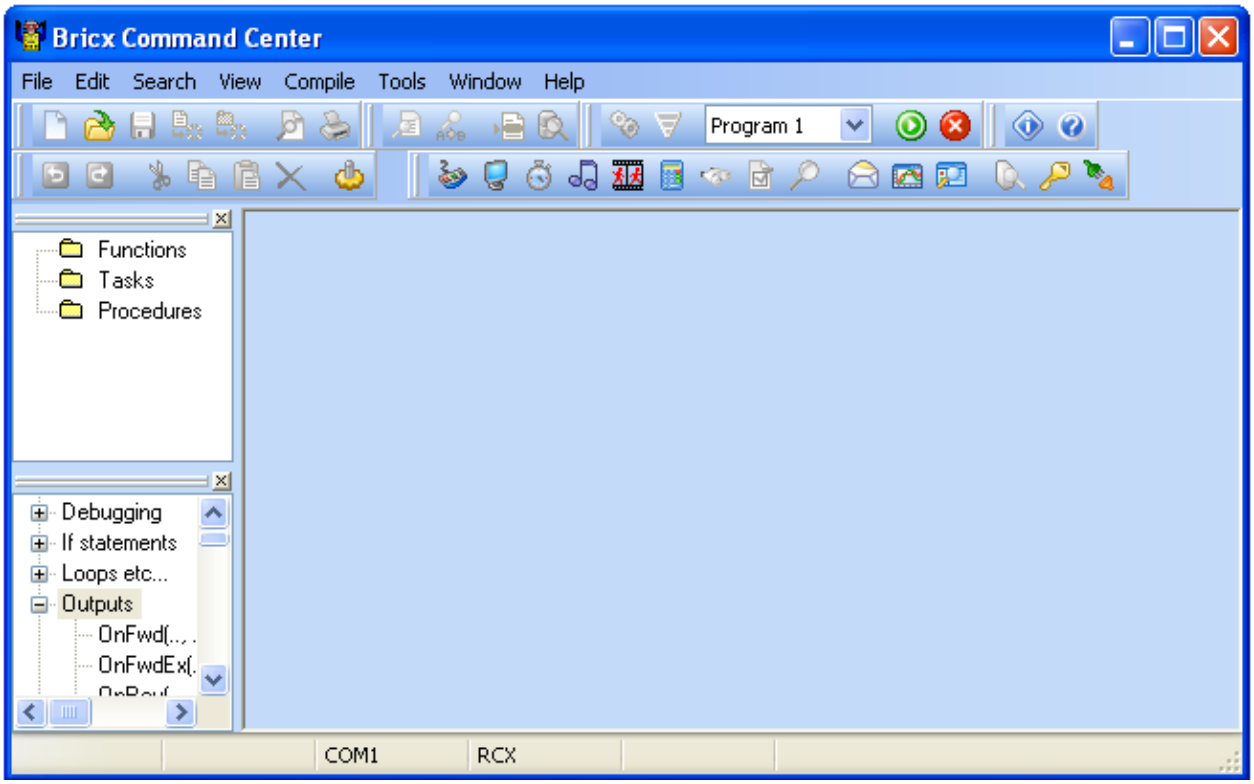

Рисунок 2 — Интерфейс пользователя Bricx CC.

Интерфейс выглядит как стандартный текстовый редактор с обычным меню, кнопками для открытия и сохранения файлов, печати, редактирования и т.п. Однако среди них есть несколько специальных меню для компилирования программ и загрузки их в робота, а также для получения отладочной информации. Для написания программы, нужно нажать на кнопку "New File", появится новое окно с текстовым редактором.

После написания программы необходимо её скомпилировать (превратить её в двоичные коды, которые робот может разобрать и выполнить) и переслать полученные двоичные коды в робота, используя USB-кабель.

На этой панели, изображенной на рис. 3, расположены кнопки, которые позволяют (слева направо) компилировать программу, загружать её в робота, запускать выполнение программы на роботе и останавливать его.

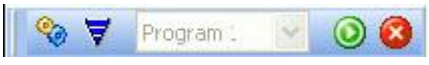

Рисунок 3 — Панель управления компиляцией и загрузки программ.

Для запуска программы на выполнение, нужно зайти в меню "My Files" на NXT Brick, "Software files", и запустить ("run") программу. Файлы программ в файловой системе NXT имеют такие же имена, как и исходные NXC-файлы.

Кроме того, после загрузки программы на робота, при нажатии на зелёную кнопку, она запустится.

При написании программы, существует вероятность допустить ошибки. Компилятор сообщает обо всех обнаруженных им ошибках. Он автоматически выделяет первую ошибку. Так же в BricxCC имеется подсветка синтаксиса, помогающая избежать ошибок при разработке.

Лабораторное занятие № 1 " Программирование микроконтроллера NXT Brick"

Введение

«Сердцем» комплекта LEGO Mindstorms NXT является микрокомпьютер NXT Brick. Он изображен на рис. 1.1.

Рисунок 1.1 — Внешний вид NXT Brick.

Технические характеристики NXT Brick

Основной процессор: Atmel® 32-bit ARM® processor, AT91SAM7S256

- 256 KB FLASH
- 64 KB RAM
- 48 MHz

Сопроцессор: Atmel® 8-bit AVR processor, ATmega48

- 4 KB FLASH

- 512 Byte RAM
- 8 MHz

Модуль Bluetooth CSR BlueCoreTM 4 v2.0 +EDR System

- Поддержка профиля последовательного порта (SPP)
- Внутренний 47 KByte RAM
- Внешний 8 MBit FLASH
- 26 MHz

Порты

- USB 2.0 порт (12 Mbit/s)
- 4 входных порта, поддерживающих цифровой и аналоговый интерфейс
- 1 высокоскоростной порт, IEC 61158 Type 4/EN 50170 compliant
- 3 выходных портов, поддерживающих энкодеры

Дисплей 100 x 64 пикселей LCD чёрно-белый графический

- Размеры: 26 X 40.6 мм

Динамик

- Поддержка частоты дискретизации 2-16 кГц

4 кнопки управления Brick

6 Батареек типа AA или перезаряжаемый литиевый аккумулятор, емкостью 1400 мАч

Внешний вид микрокомпьютера NXT Brick приведен на рис. 1.1. Питание осуществляется от 6 батарей формата AA. Блок оснащен ЖК дисплеем с возможностью отображения текста и графики. Для перемещения по разделам меню служат клавиши в виде треугольников, оранжевая клавиша служит для выбора, темно-серая – для отмены. Блок может воспроизводить звуки как из заранее записанных файлов, так и различной тональности. Подключение к NXT Brick осуществляется по интерфейсу USB, либо с помощью Bluetooth модуля. NXT Brick имеет три порта для приводов помеченных буквами A, B и C (для движения обычно используются B и C, A - для манипулятора), четыре порта для датчиков, помеченных цифрами.

С помощью языка NXC легко программировать стандартные функции прошивки NXT BRick, такие как вывод информации на дисплей, воспроизведение звуков и прочее.

Пример со звуком:

```
task main()
{
     PlayTone(1000, 300);
     Wait(300);
}
```

Эта маленькая программа будет воспроизводить звук частотой 1кГц в течение 300мс. Функция `Wait(300)`, вызываемая в последней строке не дает программе завершиться до завершения воспроизведения звука.

Пример с выводом информации на дисплей:

```
task main()
{
     TextOut(10, 10, "Hello");
     Wait(5000);
}
```

Эта программа выведет на экран надпись «Hello». Функция Wait(5000), вызываемая в последней строке не дает программе завершиться 5 секунд, в результате чего, все это время можно будет наблюдать надпись «Hello».

Цель занятия

Познакомиться с основными возможностями робототехнического микрокомпьютера NXT Brick, его блоками управления, системой меню, познакомиться со средой программирования BricxCC, произвести программирование базовых функций: вывод на экран, воспроизведение звуков.

Задание

- Изучить органы управления NXT Brick
- Изучить систему меню NXT Brick

- Изучить назначение портов NXT Brick
- Запустить среду программирования BricxCC
- Написать следующие программы:
    - издает одиночный звук определенной тональности
    - издает однократно серию звуков из файла
    - издает однократно серию звуков различной тональности (программируем мелодию)
    - издает непрерывно серию звуков различной тональности (блок "Цикл")
    - выводит на экран изображение из файла
    - выводит на экран текст в одной строке
    - выводит на экран текст в трех строках
- Подготовить отчет с текстами написанных программ.

Ход выполнения работы

1. Запустить среду BricxCC.
2. Создать новую программу с названием lab01.nxc
3. Выполнить задание
4. Подготовить отчет с текстами созданных программ

Контрольные вопросы

1. Сколько микропроцессоров используется в NXT BRick?
2. Какая функция позволяет воспроизводить звуки?
3. Какая функция позволяет вывести на экран текст?
4. Каково назначение портов обозначенных цифрами?

Лабораторное занятие № 2 "Изучение сенсорных датчиков Mindstroms NXT"

Введение

В комплект Lego Mindstorms входит датчик касания, изображенный на рис. 2.1, позволяющий роботу реагировать на прикосновения или на их отсутствие.

Рисунок 2.1 — Датчик касания.

Датчик касания NXT фактически представляет собой кнопку. С помощью датчика касания можно решать, например, такие задачи, как детектор столкновений, два датчика могут помочь определить размер объекта. Датчик подсоединяется к любому из портов, обозначенных цифрами.

Для инициализации работы с датчиками их необходимо «привязать» к определенным портам микрокомпьютера NXT Brick. Сделать это можно следующим образом:

```
task main()
{
     SetSensorTouch(S1); //привязка датчика касания к порту 1
     while (true) //бесконечный цикл
     {
          int touch1 = SENSOR_1;
          if (touch1 == 1) {
               PlayTone(1000, 300);
               Wait(300);
```

```
        }
    }
}
```

Написанная программа «ждет» нажатия кнопки, подключенной к порту 1 и воспроизводит звук частотой 1кГц и длительностью 300 мс. При нажатии кнопки, сенсор возвращает значение 1, пока кнопка не нажата – значение 0.

Также в комплект входит ультразвуковой дальномер, изображенный на рис. 2.2.

Рисунок 2.2 — Ультразвуковой дальномер.

Он работает как сонар. Он посылает ультразвуковые волны и замеряет время, которое потребовалось им, чтобы отразиться от объектов в поле зрения и вернуться к датчику. Это цифровой сенсор, что означает, что в него встроено устройство для анализа и обработки данных. С этим сенсором можно построить робота, который будет избегать препятствия, а не обнаруживать их путём столкновения.

Ультразвуковой дальномер может подключаться к любому порту NXT Brick, по умолчанию к порту 4. Особенностью работы с данным сенсором является его низкая скорость работы, поэтому подключение датчика к порту и считывание его значений происходит несколько иначе, чем в случае с кнопкой или датчиком освещенности. Датчик возвращает значения в диапазоне от 0 до 250 см.

Пример использования ультразвукового дальномера:

```
task main()
{
    SetSensorLowspeed(S4);
```

```
      while (true)
      {
            int x = SENSOR_4;
            if (x < 15)
            {
                  PlayTone(1000, 300);
                  Wait(300);
            }
      }
}
```

Написанная программа «ждет» пока расстояние между ультразвуковым дальномером, подключенным к 4 порту, и препятствием не станет меньше 15 см, после чего воспроизводит звук частотой 1кГц и длительностью 300 мс.

В комплекте имеется датчик цвета. Он изображен на рис. 2.3.

Рисунок 2.3 —Датчик цвета.

Это многофункциональный датчик. Его можно использовать как датчик освещенности и как датчик цвета.

В режиме измерения окружающей освещенности, количество света, попавшее на светочувствительный элемент, преобразуется в цифровое значение, которое уже используется в программе. Например, с датчиком, работающем в этом режиме, можно собрать робота, который ищет самое освещенное место в комнате.

В режиме измерения отраженного цвета, помимо светочувствительного элемента, активируется светодиод. Свет, выпущенный этим элементом,

отражается от какой-нибудь поверхности и попадает обратно в светочувствительный элемент.

В зависимости от того насколько светлая отражающая поверхность, в светочувствительный элемент приходит больше света. Это количество света преобразуется в цифровое значение и передается в программу. Чем темнее поверхность, тем меньше света приходит – в программу приходят маленькие значения; чем светлее поверхность, тем больше света приходит – программа оперирует с большими значениями.

В режиме Light Sensor датчик может работать с тремя различными типами подсветки – красным, синим или зеленым. Это может быть полезно для лучшего детектирования объектов различных цветов и в разных условиях освещенности. Значения, возвращаемые датчиком от 0 до 255.

Пример использования датчика цвета в режиме Light Sensor:

```
task main()
{
     SetSensorColorRed(S1);
     while (true)
     {
          int color = SENSOR_1;
          NumOut(10, 5, color);
          Wait(300);
     }
}
```

Написанная программа включит красную подсветку датчика цвета и будет выводить на экран количество отраженного света в диапазоне от 0 до 255.

В режиме Color Sensor датчик может определять цвета предметов. Каждому цвету соответствует число, возвращаемое функцией `SetSensorColorFull`:

- Черный – 1
- Синий – 2
- Зеленый – 3

- Желтый – 4
- Красный – 5
- Белый – 6

Пример использования датчика цвета в режиме Color Sensor:

```
task main()
{
     SetSensorColorFull(S1);
     while (true)
     {
          int color = SENSOR_1;
          NumOut(10, 5, color);
          Wait(1000);
     }
}
```

Написанная программа будет выводить на экран цвета объектов, поднесенных к датчику.

Цель занятия

Познакомиться с сенсорами комплекта NXT, освоить способы их подключения, диагностики, освоить практические навыки программирования робота с использованием сенсоров.

Задание

Исследование работы с датчиками касания

- Написать программу, реагирующую на нажатие датчика выводом сообщения на дисплей «Pressed».
- Написать программу, реагирующую на нажатие датчика проигрыванием звукового файла.

Работа с датчиком расстояния

- Написать программу, которая реагирует на приближение объекта на расстояние меньше 30см звуковым сигналом

- Написать программу, которая выводит на экран текущее расстояние до объекта
- Построить с помощью предыдущей программы график зависимости показаний датчика от реального расстояния до объекта.

Работа с датчиком цвета в режиме Light Sensor

- Написать программу, которая использует красную подсветку и выводит на экран текущее значение датчика цвета в режиме Light Sensor.
- Построить с помощью предыдущей программы зависимость показаний датчика в режиме красной подсветки от расстояния (от 0 до 5см с дискретностью 2 мм) до белого и черного предметов.
- Написать программу, которая использует зеленую подсветку и выводит на экран текущее значение датчика цвета в режиме Light Sensor.
- Построить с помощью предыдущей программы зависимость показаний датчика в режиме зеленой подсветки от расстояния (от 0 до 5см с дискретностью 2 мм) до белого и черного предметов.
- Написать программу, которая использует синюю подсветку и выводит на экран текущее значение датчика цвета в режиме Light Sensor.
- Построить с помощью предыдущей программы зависимость показаний датчика в режиме синей подсветки от расстояния (от 0 до 5см с дискретностью 2 мм) до белого и черного предметов.

Работа с датчиком цвета в режиме Color Sensor

- Написать программу, которая будет определять цвет предмета и выводить на экран надпись с названием этого цвета.
- Определить на каком расстоянии более эффективно использование датчика цвета.

Подготовить отчет с текстами написанных программ.

Ход выполнения работы

1. Запустить среду BricxCC.
2. Создать новую программу с названием lab02.nxc.
3. Подключить датчики к портам.
4. Выполнить задание.
5. Подготовить отчет с текстами созданных программ.

Контрольные вопросы

1. Какая функция используется для инициализации ультразвукового дальномера?
2. Какая функция используется для инициализации датчика касания?
3. В каких режимах может работать датчик цвета?
4. Какие значения возвратит функция `SetSensorColorFull` при определении датчиком цвета красного, синего, черного цветов?
5. Сколько и какие режимы подсветки имеются у датчика цвета?

Лабораторное занятие № 3 " Основные приемы управлениядвижением мобильного робота"

Введение

В комплект Lego Mindstorms входят сервомоторы, позволяющие роботу двигаться и совершать какие-либо манипуляции. В сервомоторе, изображенном на рис. 3.1, имеется встроенный датчик вращения. Что дает возможность роботу lego mindstorm двигаться точно в заданном направлении. Этот датчик измеряет обороты мотора в градусах (точность +/- 1 градус). Один полный оборот – это, как известно, 360 градусов, поэтому если установить поворот на 180 градусов, то выходная ось сделает пол оборота. Скорость вращения привода зависит от условной мощности, подающейся на сервомотор. Мощность варьируется от -100 до 100. Отрицательные значения движут привод в противоположном направлении. В данном случае под «мощностью» понимается формируемый NXT Brick ШИМ-сигнал, таким образом «мощность» зависит от скважности ШИМ.

Ниже приведен пример использования сервомоторов для движения робота.

```
task main()
{
     OnFwd(OUT_BC, 50); //движение вперед
     Wait(5000); //5 секунд
     Off(OUT_BC);   //остановка
     OnFwd(OUT_BC, -25); //движение назад
     Wait(5000); //5 секунд
     Off(OUT_BC); // остановка
     RotateMotorEx(OUT_BC, 75, 1800, 50, true, true); //поворот
по дуге
     Off(OUT_BC); // остановка
}
```

Рисунок 3.1 — Сервомотор.

Цель занятия

Освоить практические навыки построения мобильного робота, освоить практические навыки программирования различных видов движения мобильного робота.

Задание

- На основе предложенной схемы собрать конструкцию мобильного робота
- Написать следующие программы:
- Прямолинейное движение вперед на расстояние 1 метр
    - Движение по "квадрату"
    - Движение по "окружности"
    - Движение по "восьмерке"
- Подготовить отчет с текстами написанных программ

Ход выполнения работы

1. Собрать робота, изображенного на рисунках 3.2 – 3.4.

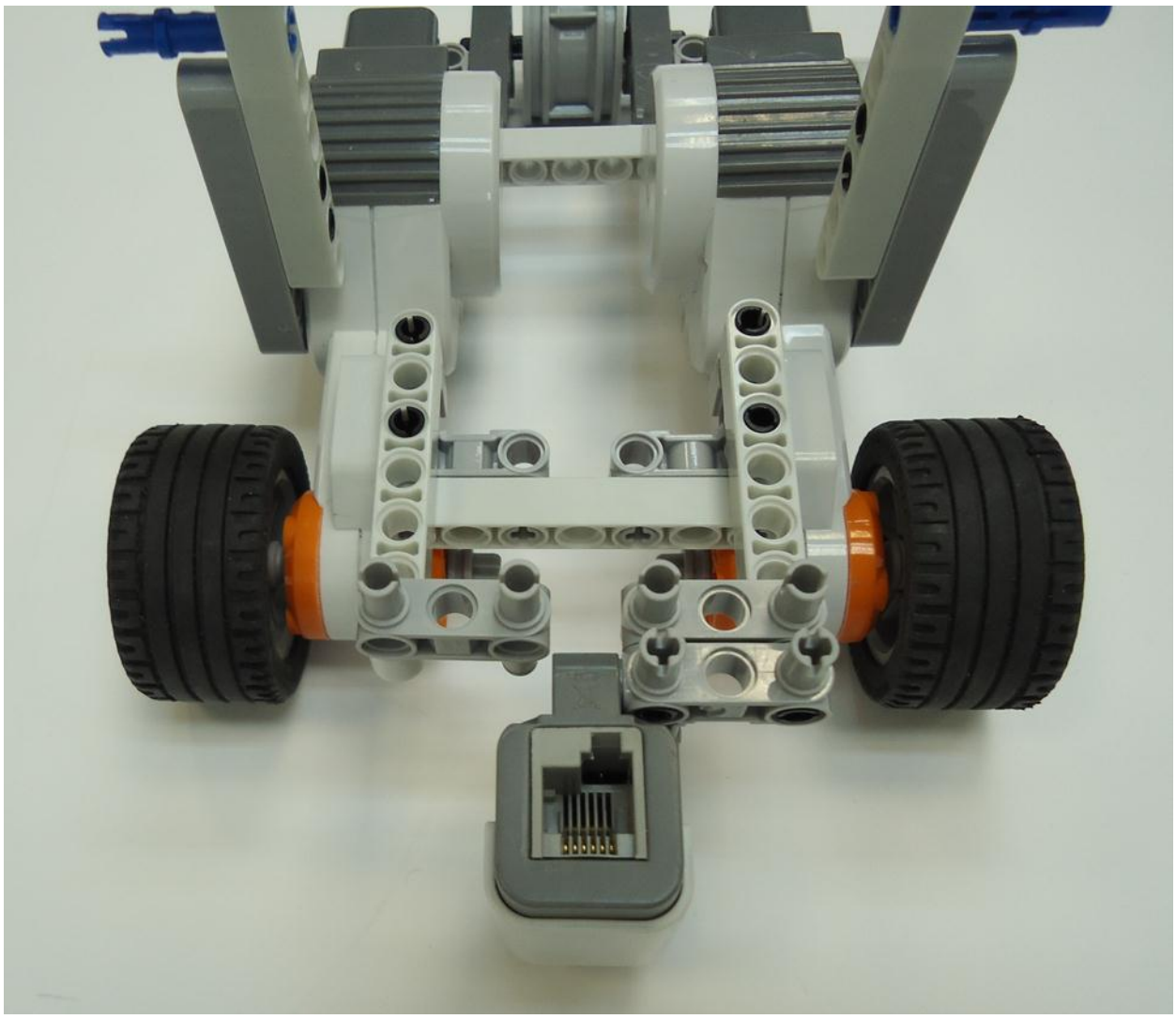

Рисунок 3.2 — Передняя часть робота со снятым NXT Brick.

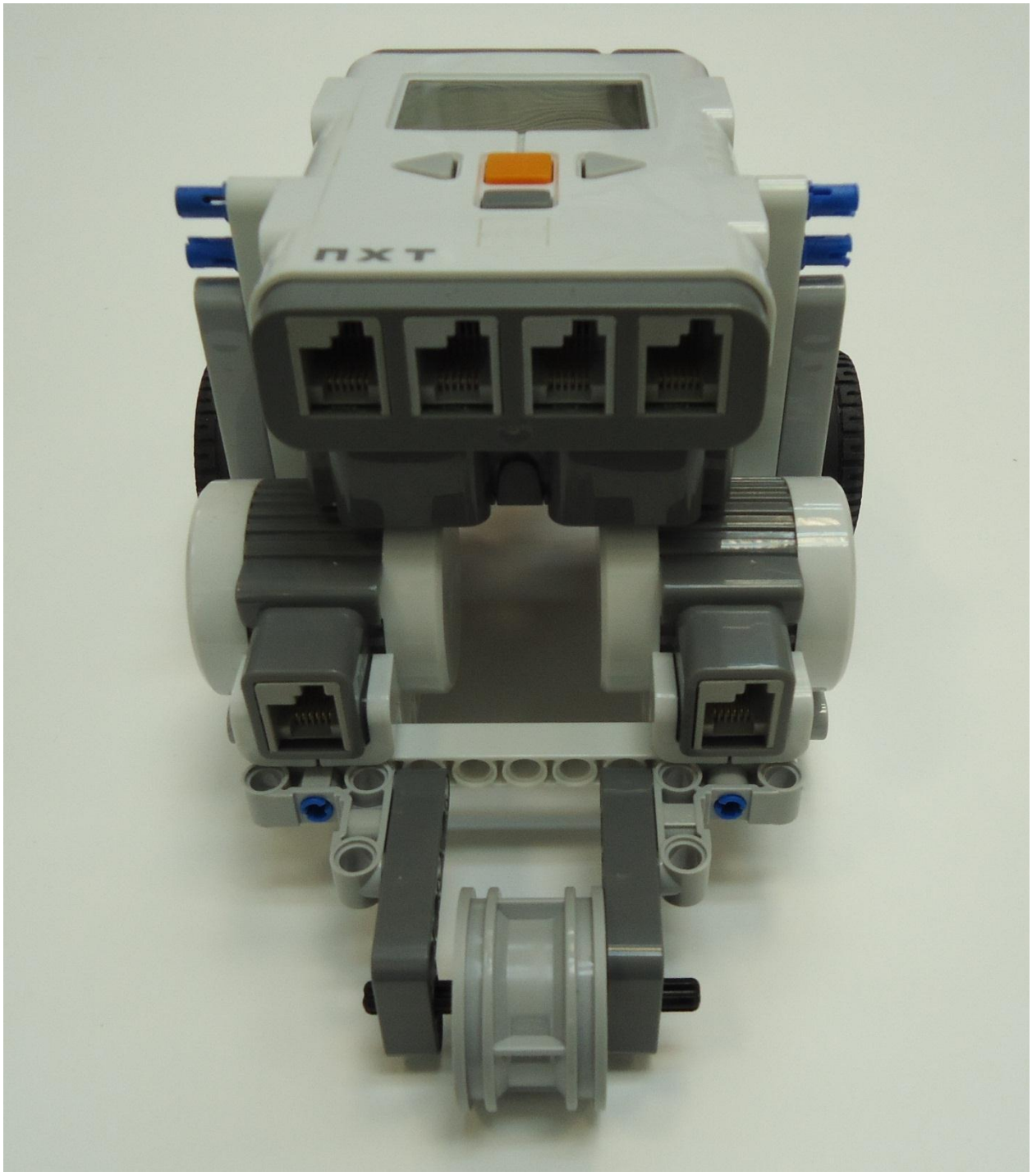

Рисунок 3.3 — Задняя часть робота.

Рисунок 3.4 — Вид с боку на робота в сборе.

1. Подключить сервомоторы и датчик цвета к портам NXT Brick.
2. Запустить среду BricxCC.
3. Создать новую программу с названием lab03.nxc.
4. Выполнить задание.
5. Подготовить отчет с текстами созданных программ

Контрольные вопросы

1. Какая функция используется для движения робота вперед?
2. Какая функция используется для остановки моторов?
3. В чем отличие функций `RotateMotor` и `RotateMotorEx`?
4. В каких пределах варьируется мощность подаваемая на моторы?

Лабораторное занятие № 4 "Движение мобильного робота по черной линии с помощью одного датчика цвета"

Введение

Движение по линии можно осуществлять несколькими способами, один из простейших — движение по границе черной линии и поля с использованием одного цветового датчика в режиме фиксации отраженного света. Сенсор содержит свето- и фотодиод. Светодиод освещает участок поверхности, а фото-диод принимает отраженный свет. По интенсивности полученного света можно судить о том, на каком участке контрастной линии робот находится. Задача заключается в том, чтобы, управляя скоростями двух колес, перемещать робота вдоль линии. Одним из способов решения этой задачи является удерживание сенсора света на кромке линии. Для этого необходимо замерить значение уровня освещенности на границе линии, этот уровень будет принят эталонным. Всякое отклонение от этого уровня (ошибку) нужно компенсировать ускорением одного из колёс и торможением другого. Ниже представлена простейшая программа движения по линии на языке NXC:

```
task main()
{
     int offset = 25; //эталонное значение освещенности на
граница черной линии и поля
     SetSensorColorRed(S3); //инициализация датчика цвета
     while (true) //бесконечный цикл
          {
          int light = SENSOR_3; //постоянно получаемое значение
освещенности с датчика
          if (light >= offset) OnFwd(OUT_B, 75);
          else
          OnFwd(OUT_C, 75);
          }
```

```
}
```

При изменении внешних условий, в частности освещенности, эталонное значение на границе линии и поля придется замерить заново. Меняя эталонное значение, мы можем «загонять» робота либо больше на линию, либо на белое поле. Этот прием может помочь при прохождении резких поворотов, когда робот между двумя измерениями не успевает скорректировать мощность двигателей.

Цель занятия

Освоить практические навыки программирования движения мобильного робота по линии с помощью одного датчика цвета.

Задание

- Написать следующие программы:
  - Движение вдоль черной линии используя "классический алгоритм".
  - Сделать движение робота более плавным.
- Подготовить отчет с текстами написанных программ

Ход выполнения работы

1. Запустить среду BricxCC.
2. Создать новую программу с названием lab04.nxc.
3. Выполнить задание.
4. Подготовить отчет с текстами созданных программ.

Контрольные вопросы

1. В каком режиме работает датчик цвета при движении робота по линии?
2. Как робот «видит» черную линию?

Лабораторное занятие № 5 "Движение мобильного робота по черной линии с помощью одного датчика цвета, используя ПИД регулятор"

Введение

Пропорционально-интегрально-дифференциальный (ПИД) регулятор используется в системах автоматического управления для формирования управляющего сигнала с целью получения необходимых точности и качества переходного процесса. ПИД-регулятор формирует управляющий сигнал, являющийся суммой трёх слагаемых, первое из которых пропорционально разности входного сигнала и сигнала обратной связи (сигнал рассогласования), второе — интеграл сигнала рассогласования, третье — производная сигнала рассогласования.

Если какие-то из составляющих не используются, то регулятор называют пропорционально-интегральным, пропорционально-дифференциальным, пропорциональным и т. п.

Пропорциональная составляющая вырабатывает выходной сигнал, противодействующий отклонению регулируемой величины от заданного значения, наблюдаемому в данный момент времени. Он тем больше, чем больше это отклонение. Если входной сигнал равен заданному значению, то выходной равен нулю. Однако при использовании только пропорционального регулятора значение регулируемой величины никогда не стабилизируется на заданном значении. Существует так называемая статическая ошибка, которая равна такому отклонению регулируемой величины, которое обеспечивает выходной сигнал, стабилизирующий выходную величину именно на этом значении. Например, в регуляторе температуры выходной сигнал (мощность нагревателя) постепенно уменьшается при приближении температуры к заданной, и система стабилизируется при мощности равной тепловым потерям. Температура не

может достичь заданного значения, так как в этом случае мощность нагревателя станет равна нулю, и он начнёт остывать.

Чем больше коэффициент пропорциональности между входным и выходным сигналом (коэффициент усиления), тем меньше статическая ошибка, однако при слишком большом коэффициенте усиления, при наличии задержек в системе, могут начаться автоколебания, а при дальнейшем увеличении коэффициента система может потерять устойчивость.

Интегральная составляющая пропорциональна интегралу от отклонения регулируемой величины. Её используют для устранения статической ошибки. Она позволяет регулятору со временем учесть статическую ошибку. Если система не испытывает внешних возмущений, то через некоторое время регулируемая величина стабилизируется на заданном значении, сигнал пропорциональной составляющей будет равен нулю, а выходной сигнал будет полностью обеспечивать интегральная составляющая. Тем не менее, интегральная составляющая также может приводить к автоколебаниям.

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

$$U(t) = P + I + D = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt}, \qquad (5.1)$$

где K_p, K_i, K_d — коэффициенты усиления пропорциональной, интегральной и дифференциальной составляющих регулятора, соответственно;

$e = (x_0 - x)$ – называется невязкой или рассогласованием;

x_0 – заданное значение;

x – некоторое измененное значение.

П-регулятор – это частный вид ПИД-регулятора без учета интегральной и дифференциальной составляющих.

Пример использования П регулятора:

```
task main()
{
     int offset = 25; //эталонное значение освещенности на
граница черной линии и поля
     int error = 0;
     int k = 2; //коэффициент усиления
     int speed = 50; //мощность, подаваемая на двигатели
     int left = 0; //мощность, подаваемая на левый двигатель, с
учетом ошибки
     int right = 0; // мощность, подаваемая на правый двигатель,
с учетом ошибки
     SetSensorColorRed(S3); //инициализация датчика цвета
     while (true) //бесконечный цикл
          {
          int light = SENSOR_3; //постоянно получаемое значение
освещенности с датчика
          error = offset - light; //постоянное вычисление ошибки
          left = speed + k * error;
          right = speed – k * error;
          OnFwd(OUT_B, left); //мощность, подаваемая на левый
мотор
          OnFwd(OUT_C, right); //мощность, подаваемая на правый
мотор

          }
}
```

При движении на мощностях близких к максимальным (максимум равен 100) иногда возникают ситуации такого рода, что на двигатели подается мощность больше 100. Чтобы избежать этого, можно искусственно ограничить мощность, подаваемую двигателям. Для этого необходимо написать следующую функцию:

```
void Move(int left, int right) {
     if (left < -100) left = -100;
```

```
    if (right < -100) right = -100;
    if (left > 100) left = 100;
    if (right > 100) right = 100;
    OnFwd(OUT_B, left);
    OnFwd(OUT_C, right);
}
```

На вход подаются значения мощности для левого и правого двигателя, и если они больше пороговых значений, то на двигатели поступают только пороговые значения.

Задание

- Написать следующие программы:
  - Движение вдоль черной линии используя П-регулятор
  - Движение вдоль черной линии используя ПИ-регулятор
  - Движение вдоль черной линии используя ПИД-регулятор
- Подготовить отчет с текстами написанных программ

Ход выполнения работы

1. Запустить среду BricxCC.
2. Создать новую программу с названием lab05.nxc.
3. Выполнить задание.
4. Подготовить отчет с текстами созданных программ.

Контрольные вопросы

1. Для чего используется ПИД регулятор?
2. В чем преимущества ПИ регулятора по сравнению с П регулятором?

Лабораторное занятие № 6 "Движение мобильного робота по черной линии с помощью двух датчиков цвета, используя ПИД регулятор "

Введение

Движение по линии с использованием двух датчиков освещенности осуществляется путем размещения датчиков с двух сторон от линии. За ошибку в этом случае будем считать разницу освещенности на датчиках. Когда робот едет строго по линии, она равна нулю. При заезде на линию одним из датчиков разность освещенности будет существенно меняться. Чтоб выровнять робота, нужно подать ускорение на одно из колёс и торможение на другое в зависимости от знака ошибки.

Ниже представлена простейшая программа движения по линии с помощью двух датчиков цвета на языке NXC:

```
task main ()
{
     int error;
     int speed = 40;
     SetSensorColorRed(S3);
     SetSensorColorRed(S2);
     while(true)
          {
          int lightright = SENSOR_3;
          int lightleft = SENSOR_2;
          int error = lightleft - lightright;
          int speedl = speed + error;
          int speedr = speed - error;
          OnFwd(OUT_B, speedl);
          OnFwd(OUT_C, speedr);
  }
}
```

Задание

- На основе предложенной схемы собрать конструкцию мобильного робота
- Написать следующие программы:
    - Движение вдоль черной линии используя "классический алгоритм"
    - Движение вдоль черной линии используя П-регулятор
    - Движение вдоль черной линии используя ПИ-регулятор
    - Движение вдоль черной линии используя ПИД-регулятор
- Подготовить отчет с текстами написанных программ

Ход выполнения работы

1. Собрать робота, изображенного на рисунках 6.1 – 6.3.

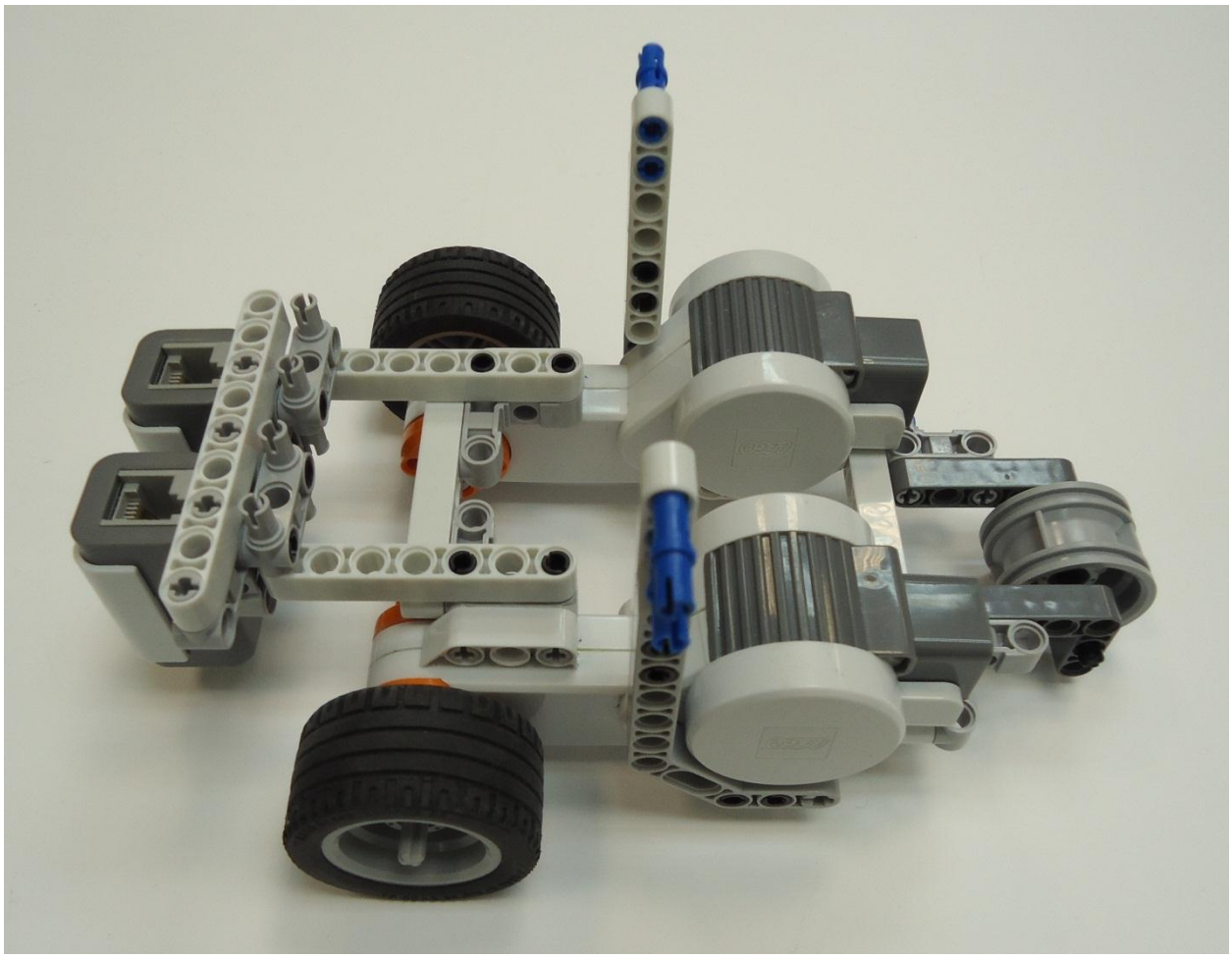

Рисунок 6.1 —Робот со снятым NXT Brick.

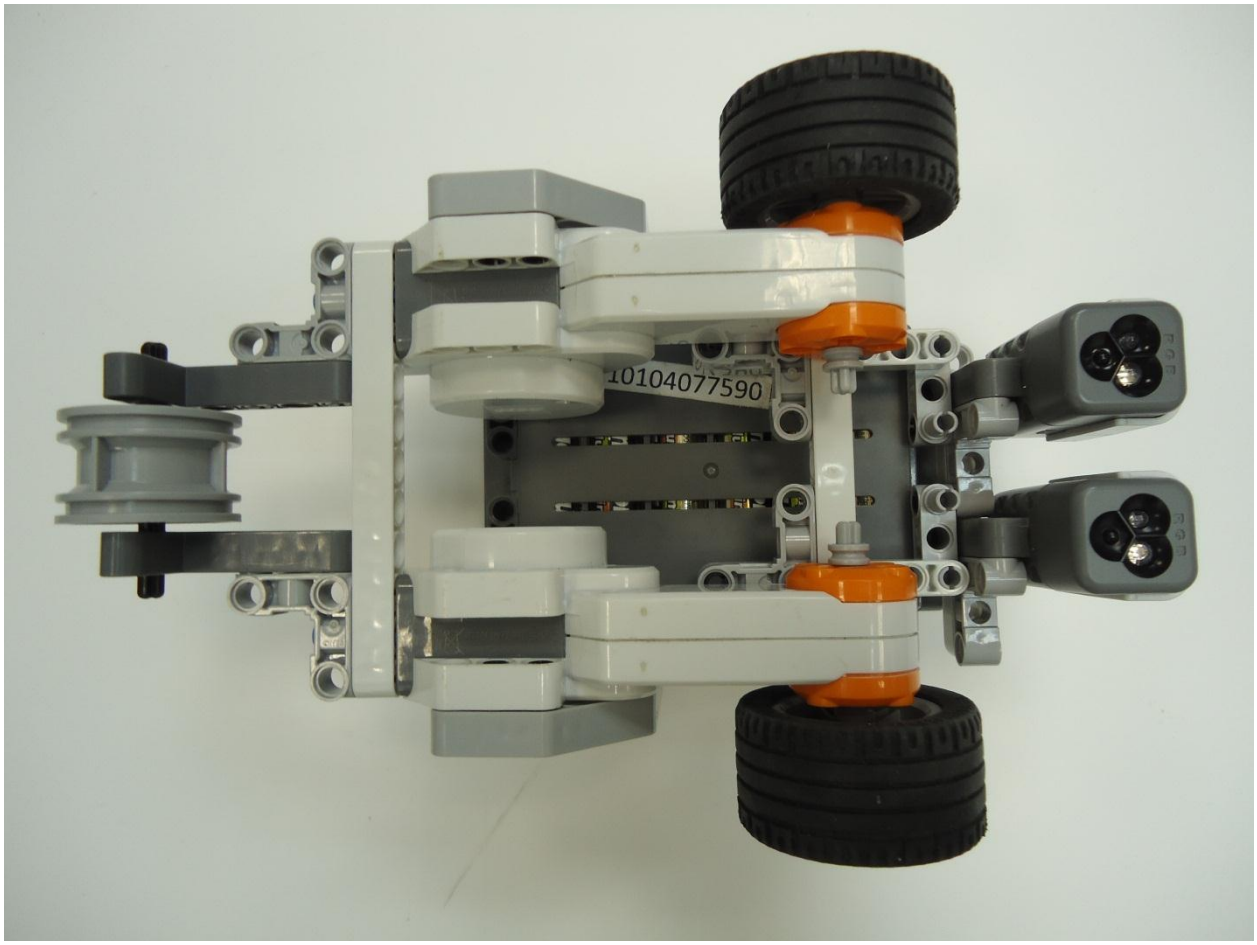

Рисунок 6.2 —Робот со снятым NXT Brick.(вид снизу)

Рисунок 6.1 — Вид с боку на робота в сборе.

2. Подключить сервомоторы и датчики цвета к портам NXT Brick.
3. Запустить среду BricxCC.
4. Создать новую программу с названием lab06.nxc.
5. Выполнить задание.
6. Подготовить отчет с текстами созданных программ

Контрольные вопросы

1. В чем преимущества движения по двум датчикам цвета, по сравнению с движением по одному датчику цвета?
2. Что берут за «ошибку» при движении по двум датчикам цвета?

Лабораторное занятие № 7 "Ориентирование мобильного робота в «городе» с помощью двух датчиков цвета."

Введение

Часто встречаются задачи: детектирования и проезда перекрестков, поворотов на перекрестке. При проезде перекрестка четырех дорог мобильный робот практически не сбивается с трассы, так как разница значений на датчиках не меняется. Но при проезде перекрестка в виде лежачей буквы «Т» робот очень часто сбивается с трассы, так как разница значений на датчиках меняется очень резко.

Задание

- Написать следующие программы:
  - Движение вдоль черной линии, при обнаружении перекрестка остановка.
  - Движение вдоль черной линии, при обнаружении перекрестка подать звуковой сигнал и продолжить движение.
  - Движение вдоль черной линии, на каждом втором перекрестке поворачивать направо.
- Подготовить отчет с текстами написанных программ

Ход выполнения работы

1. Запустить среду BricxCC.
2. Создать новую программу с названием lab07.nxc.
3. Выполнить задание.
4. Подготовить отчет с текстами созданных программ

Контрольные вопросы

1. Возможен ли проезд перекрестков с использованием одного датчика цвета?