

Л.Ю. Овсяницкая, Д.Н. Овсяницкий, А.Д. Овсяницкий

**Курс программирования робота EV3
в среде Lego Mindstorms EV3**

Издание второе, переработанное и дополненное

**Москва
2016**

УДК 004.42+004.896

ББК 32.81

О-34

Овсяницкая, Л.Ю. Курс программирования робота Lego Mindstorms EV3 в среде EV3: изд. второе, перераб. и допол. / Л.Ю. Овсяницкая, Д.Н. Овсяницкий, А.Д. Овсяницкий. – М.: «Перо», 2016. – 296 с.

ISBN 978-5-906862-76-1

Книга посвящена программированию робота EV3 в среде Lego Mindstorms EV3. Работа является результатом многолетнего опыта непосредственного участия авторов в региональных, всероссийских и международных состязаниях по робототехнике и педагогической деятельности, направленной на подготовку учителей, преподавателей и тренеров по данной тематике.

Книга будет полезна педагогам начального, среднего, высшего и дополнительного образования, учащимся, студентам и всем, интересующимся вопросами робототехники.

Рецензент:

доктор физико-математических наук,
профессор А.Ф. Шориков.

ISBN 978-5-906862-76-1

© Л.Ю. Овсяницкая, Д.Н. Овсяницкий, А.Д. Овсяницкий, 2016

Оглавление

Введение	6
Глава 1. Характеристики робота. Создание и запуск первого проекта	7
1.1. Краткая характеристика роботизированных платформ. Обзор среды программирования Lego Mindstorms EV3	7
1.2. Способы подключения робота к компьютеру. Обновление прошивки блока EV3. Загрузка программ в блок EV3	23
Глава 2. Программирование робота	30
2.1. Моторы. Программирование движений по различным траекториям	30
2.2. Работа с подсветкой, экраном и звуком	45
2.2.1. Работа с экраном	45
2.2.2. Работа с подсветкой кнопок на блоке EV3	63
2.2.3. Работа со звуком	66
2.3. Программные структуры	76
2.3.1. Структура Ожидание	76
2.3.2. Структура Цикл	77
2.3.3. Структура Переключатель	85
2.4. Работа с данными	92
2.4.1. Типы данных. Проводники	92
2.4.2. Переменные и константы	95
2.4.3. Математические операции с данными	105
2.4.4. Другие блоки работы с данными	111
2.4.5. Работа с массивами	116
2.4.6. Логические операции с данными	125
2.5. Работа с датчиками	129
2.5.1. Датчик касания	130
2.5.2. Датчик цвета	137
2.5.3. Гироскопический датчик	156
2.5.4. Ультразвуковой датчик	166
2.5.5. Инфракрасный датчик и маяк	174
2.5.6. Датчик Вращение мотора (определение угла/количества оборотов и мощности мотора)	185
2.5.7. Кнопки управления модулем	188
2.6. Работа с файлами	194

2.7. Совместная работа нескольких роботов	200
2.7.1. Соединение роботов кабелем USB	200
2.7.2. Связь роботов с помощью Bluetooth-соединения	207
2.8. Полезные блоки и инструменты	213
2.8.1. Блок «Поддерживать в активном состоянии»	213
2.8.2. Блок «Остановить программу».....	215
2.8.3. Создание подпрограмм	216
2.8.4. Запись комментариев.....	227
2.8.5. Использование проводного ввода порта	229
Глава 3. Основные виды соревнований и элементы заданий	232
3.1. Соревнования Сумо	232
3.2. Кегельринг	236
3.3. Слалом (объезд препятствий).....	243
3.4. Программирование движения по линии	247
3.4.1. Алгоритм движения по линии «Зигзаг» с одним и двумя датчиками цвета	251
3.4.2. Алгоритм «Волна»	255
3.4.3. Алгоритм автоматической калибровки датчика цвета	258
3.5. Пропорциональное линейное управление	261
3.5.1. Движение по линии на основе пропорционального управления.....	261
3.5.2. Поиск и подсчёт перекрёстков при пропорциональном управлении движением по линии	270
3.5.3. Проезд инверсии	273
3.5.4. Движение робота вдоль стены.....	276
3.6. Поиск цели в лабиринте.....	282
Глава 4. Обновление встроенного ПО и перезапуск блока EV3	286
Глава 5. Использование сторонних датчиков.....	290
5.1. Работа с NiTech датчиком цвета	291
5.2. Использование других датчиков.....	296
Заключение.....	298

Перечень проектов

Проект «Верная собачка»	90
Проект «Спортивное табло»	98
Проект «Автофиниш»	102
Проект «60 секунд»	109
Проект «Запись и считывание цветного штрих-кода»	120
Проект «Сортировка массива методом пузырька»	123
Проект «Умный дом»	153
Проект «Упрямый робот»	160
Проект «Робот с дистанционным управлением»	182
Проект Мультипликационная игра на экране блока EV3 «Поймай снежок»	191
Проект «Построение 3D карты поверхности»	197
Проект «EV3 – музыкальный синтезатор»	203

изображение катушки (рис. 1.1.11б). Блок станет активным (ярким) (рис. 1.1.11в).

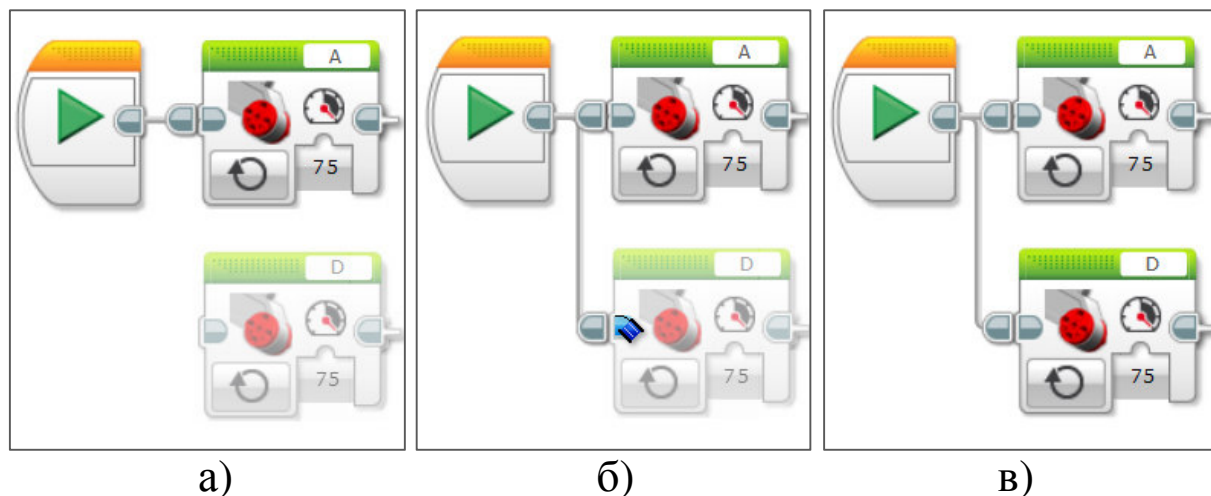


Рисунок 1.1.11. Присоединение параллельной ветки программы

В среде Lego Mindstorms EV3 появилась возможность запуска нескольких программ, которые будут выполняться одновременно. Для этого нужно поместить на поле дополнительные кнопки *Старт* (рис. 1.1.12).

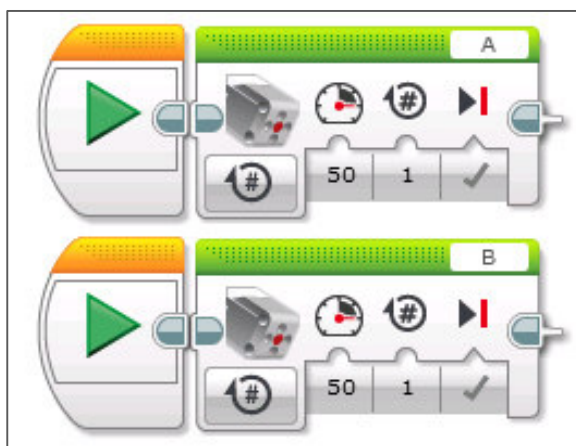



Рисунок 1.1.12. Параллельные программы

Для масштабирования изображений (рис. 1.1.13 а,б) используются стандартные для MSWindows сочетания клавиши *Ctrl* и колеса прокрутки мыши или значки в правом верхнем углу окна: . Масштабирование используется при навигации в больших программах, копировании определённых блоков и многом другом.

Блоки Большой мотор и Средний мотор

Первый блок палитры называется *Средний мотор*, второй – *Большой мотор*. Блоки служат для управления одним мотором и имеют одинаковый функционал. Рассмотрим структуру блоков на примере блока большого мотора (рис. 2.1.4).

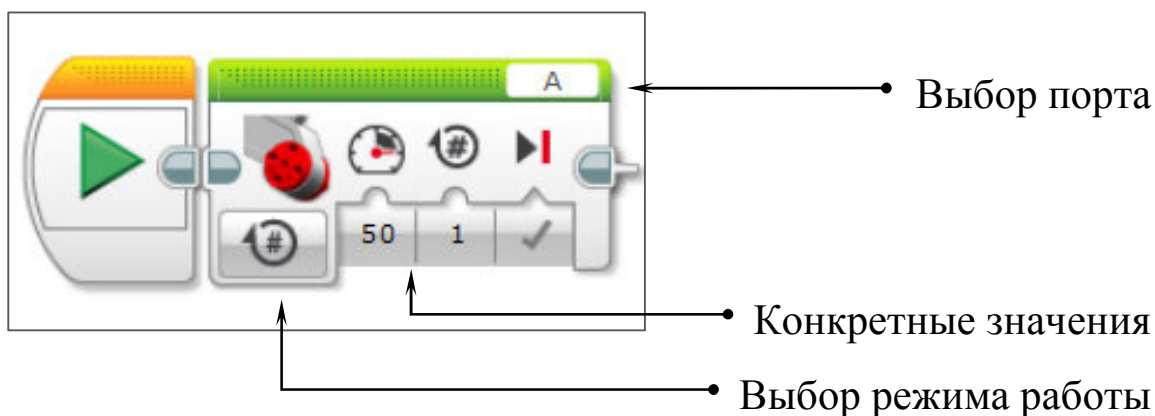


Рисунок 2.1.4. Блок управления большим мотором

Прежде всего щёлкните по букве, обозначающей название порта, и выберите название порта, к которому подключён мотор.

Рассмотрим подробнее каждый управляющий элемент.

1. Выбор режима работы:

а) **включить** (рис. 2.1.5);

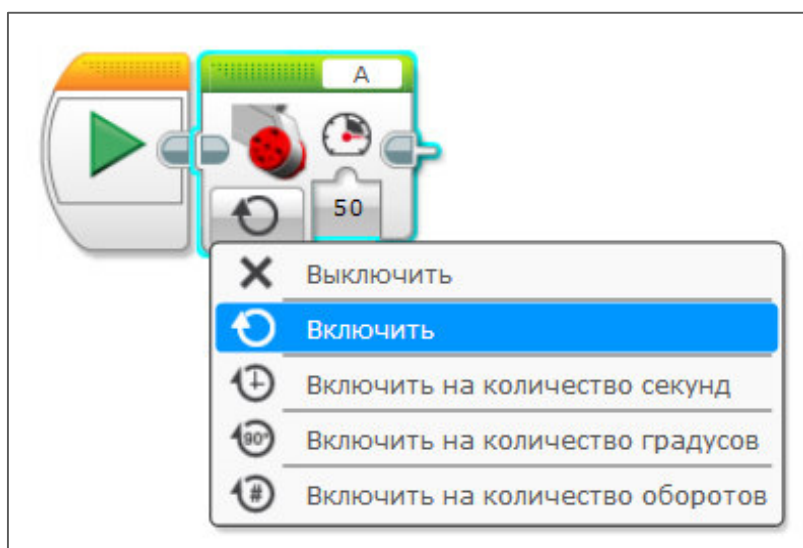


Рисунок 2.1.5.
Режим Включить

Когда мы хотим, чтобы робот ничего не выполнял в случае определённого условия, допустимо оставить данную ветку незаполненной.

 **Важно!**

В структуре Переключатель один из вариантов всегда устанавливается работающим «по умолчанию». Этот вариант обозначается чёрной точкой слева от значения.

Принцип работы варианта «по умолчанию» следующий: если ни одно из условий не совпадает ни с одним из указанных вариантов, то выполняется вариант, имеющий отметку «по умолчанию».

Например, если в условии (рис. 2.3.3.2) задано число 5 (а у нас описаны только три варианта, соответствующие числам 1, 2 и 3), то выполнение программы продолжится по ветке 1.

Подробнее познакомимся со структурой *Переключатель* на конкретных примерах.



Упражнение

Задача. Составьте программу, которая, в случае нажатия на датчик касания, проговаривает «Yes», иначе – «No».

Решение:

1. Так как робот должен осуществлять непрерывную проверку состояния нажатия кнопок на блоке, то перенесём в рабочее окно структуру *Цикл* и установим условие выхода из цикла – *Неограниченный*.
2. Поместим внутрь цикла структуру *Переключатель* и выберем в качестве условия *Датчик касания – Сравнение – Состояние*.

каждую секунду угол поворота стрелки должен составлять 6 градусов.

4. Числовые значения секунд должны отображаться в диапазоне от 0 до 60. Для этого в каждый шаг выполнения цикла будем вычислять остаток от деления числа выполненных итераций (шагов) на 60.

5. Вставляем программный блок большого мотора (А) в режиме *Включить на количество градусов* для реализации вращения секундной стрелки. В каждый шаг цикла, который длится 1 секунду, мотор должен поворачивать вал на 6 градусов.

(Можно использовать большой или средний моторы. При использовании среднего мотора движения стрелки будут более чёткими, без «отскоков»).

6. Вставляем блок *Ожидание* со значением 1 секунда.

На рис. 2.4.3.3 показано изображение экрана блока EV3 в момент выполнения программы и подсоединение «секундной стрелки» к мотору.

На рис. 2.4.3.4 представлена программа, реализующая описанный алгоритм.

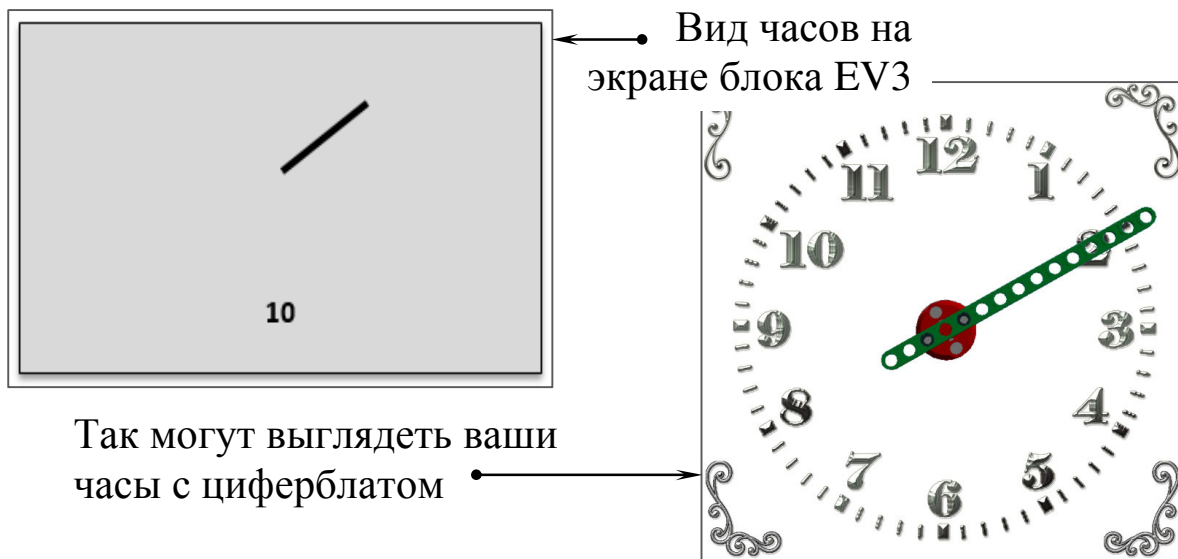


Рисунок 2.4.3.3. Изображение экрана блока EV3 в момент выполнения программы и подсоединение секундной стрелки к мотору

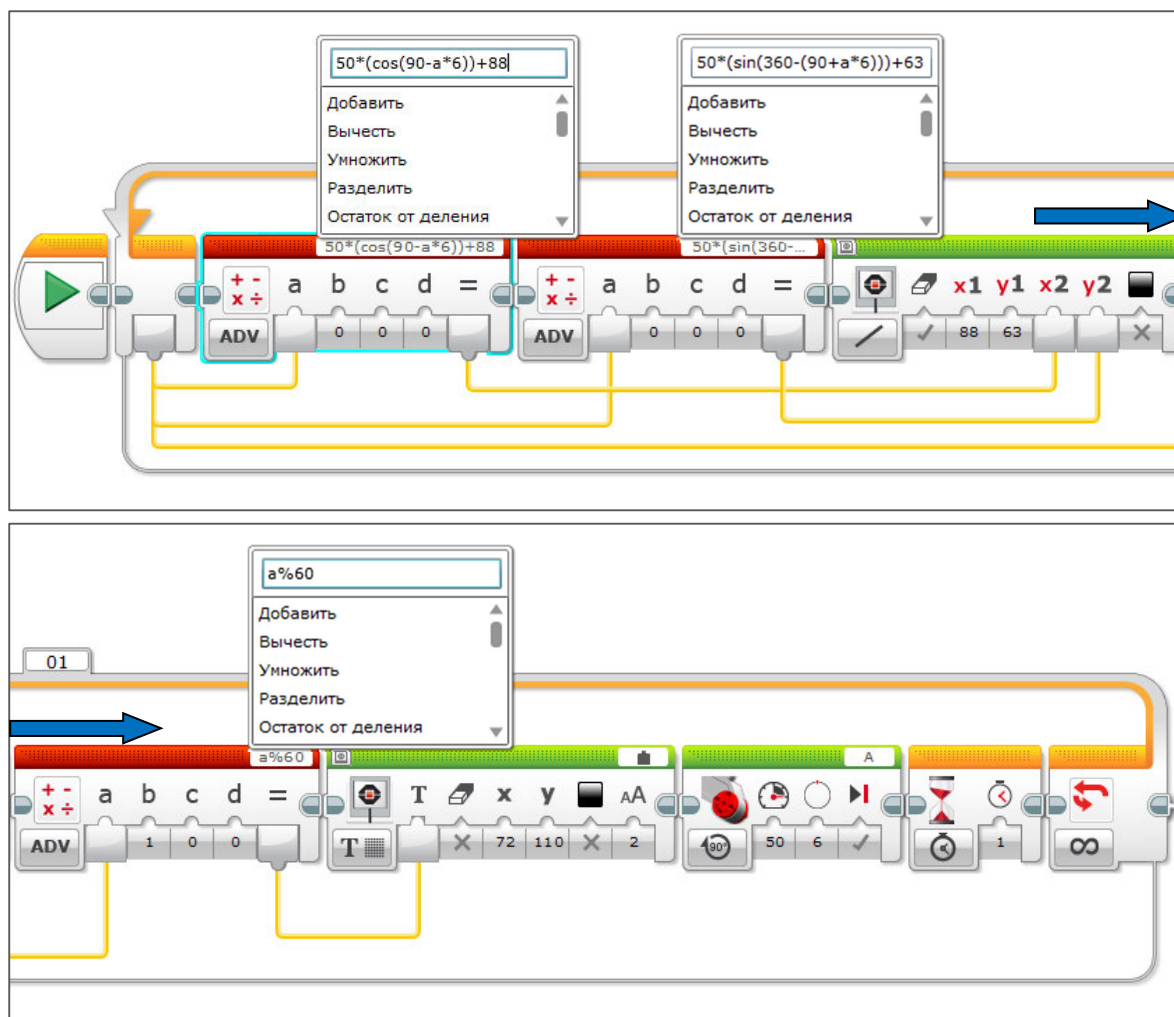


Рисунок 2.4.3.4. Программа реализации проекта «60 секунд»

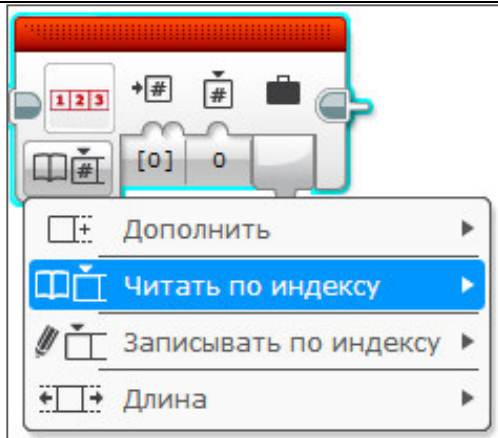
Задания для самостоятельной работы

1. Добавьте на экран изображение циферблата часов.
2. Добавьте по аналогии минутную и часовую стрелки.

2.4.4. Другие блоки работы с данными

Рассмотрим особенности работы с блоками, которые используются в программировании реже, чем блоки арифметики, константы или переменной. Однако их применение позволит решить некоторые задачи профессионально и оптимально.

В табл. 2.4.4.1 приведём пример использования блока *Округление* с разными типами округлений.



Блок *Операции над массивом* позволяет определять длину массива, читать, записывать и удалять требуемые элементы массива.

😊👍 **Важно!**

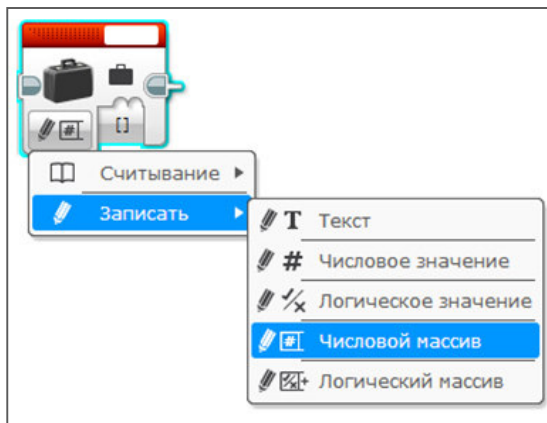
*Прежде чем начать работу с массивами, необходимо их инициализировать, т.е. указать тип (числовой или логический) и присвоить имя. Данные в массив можно вносить в ручном или автоматическом режиме (считывая показания с датчиков). Для создания массива необходимо использовать блок *Переменная*.*

Создание массива. Запись массива в переменную

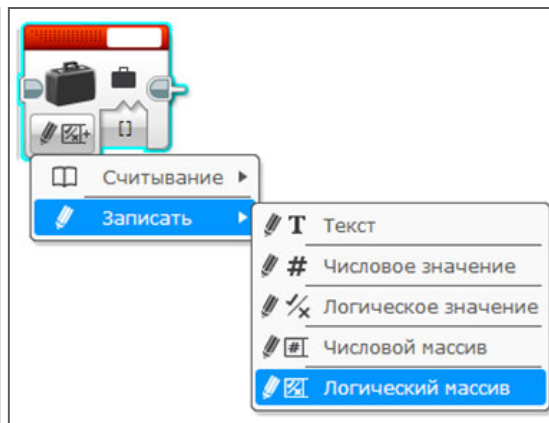
Для того чтобы создать и заполнить массив, необходимо:

(а) перенести на рабочее поле блок *Переменная* и определить её режим (*Записать*) и тип (*числовой или логический массив*);

Формирование числового массива



Формирование логического массива



а)

Обратите внимание, что в блоке *Звук* установлен режим воспроизведения *Воспроизвести один раз*. Если мы установим режим *Ожидать завершения*, то робот может проехать зелёную полосу, поскольку в этот момент будет ожидать завершения проговаривания слова *Go*.

Задача 2. Направить датчик цвета вниз. На поле наклеены полосы разных цветов. Роботу необходимо пересекать полосы и остановиться на пятой синей полосе.



• Направление движения робота

Решение:

1. Создаём цикл 01, условие завершения – подсчёт (устанавливаем значение 5).
2. Робот пересекает цветные линии. Движение создаётся моторами, подключёнными к портам В и С с мощностью 50 единиц.
3. Робот движется до тех пор, пока датчик цвета не обнаружит синий цвет (блок *Ожидание*, режим *Датчик цвета – Сравнение – Цвет*).
4. После обнаружения синей линии робот проговаривает *Blue* (блок *Звук* в режиме *Воспроизвести один раз*. Если мы установим режим блока звука *Ожидать завершения*, то робот может проехать следующую линию, ожидая полного проговаривания слова *Blue*).
5. Создаём цикл 02, условие завершения – *Время – 0,2 сек.*

значение угла не остаётся постоянным, а ошибочно меняется или дрейфует. Поэтому, чем больше времени проходит от начала первого обращения к гироскопическому датчику до чтения показаний, тем менее точными становятся результаты за счёт систематического накопления ошибки. Поэтому перед началом каждого измерения всегда необходимо производить обнуление угла при помощи режима *Сброс*.

Рассмотрим подробнее работу гироскопического датчика. Датчик работает в режимах измерения, сравнения и позволяет делать сброс значений (рис. 2.5.3.3).

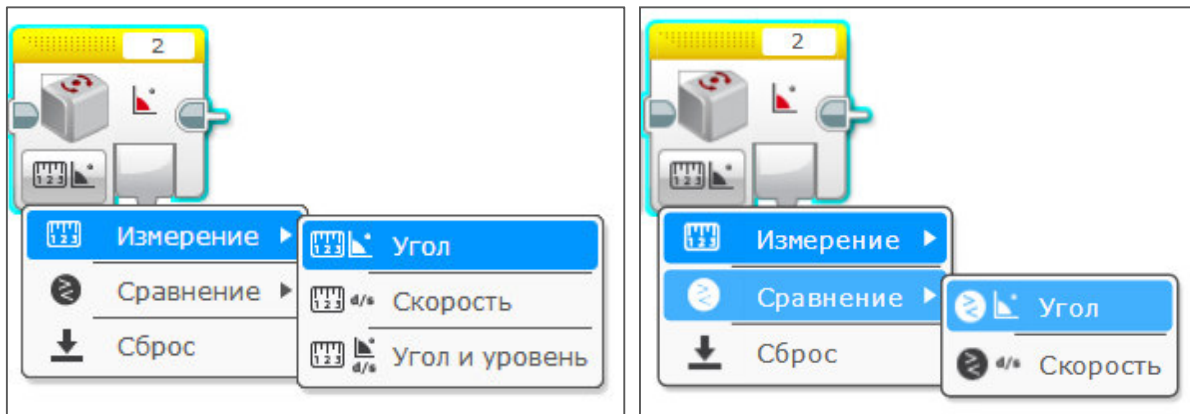
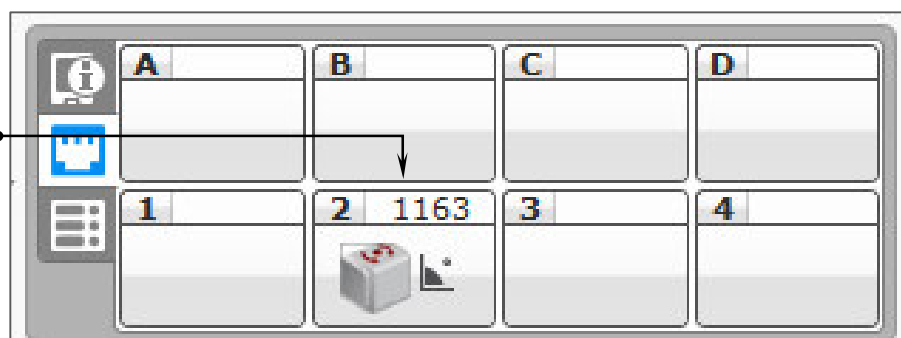


Рисунок 2.5.3.3. Режимы работы гироскопического датчика

😊👍 **Важно!**

Иногда (достаточно часто!) при работе с гироскопическим датчиком можно наблюдать следующее: при запущенной программе робот находится в **неподвижном состоянии**, а значение угла **постоянно увеличивается (дрифт)**, скорость увеличения может составить более 1 градуса в секунду!

Наращение значений датчика



Если маяк находится очень далеко (дальше 1 м), значение измерения будет 100, если очень близко (минимум 1 см) – 0. Промежуточные результаты также *не соответствуют сантиметрам*.

В том случае, когда маяк расположен *прямо перед датчиком*, относительный результат измерения угла будет равен 0, максимальное расположение маяка слева, против часовой стрелки -25 (максимальный определяемый угол отклонения приблизительно 100 градусов), справа, по часовой стрелке 25 (рис. 2.5.5.5).

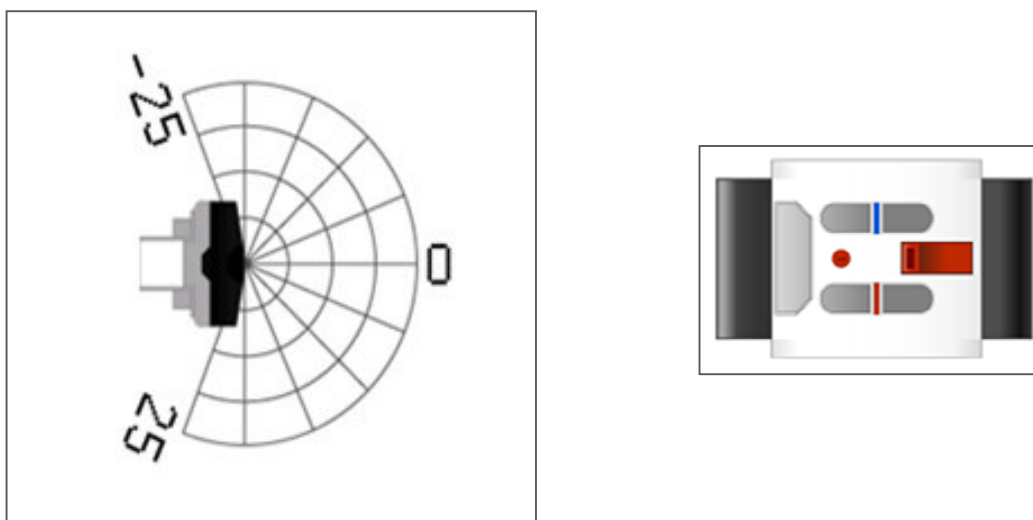


Рисунок 2.5.5.5. Положение ИК-маяка относительно ИК-датчика

Рассмотрим примеры программ. Расположите маяк перед роботом, включите его и направьте в сторону робота. Светодиодный индикатор включится и будет гореть. Маяк будет непрерывно передавать сигнал. На блоке инфракрасного датчика установите тот же канал, который установили на маяке. Датчик будет обнаруживать маяк только на том канале, который вы укажете в своей программе.

Маяк выключается, если не используется в течение часа. На рис. 2.5.5.6 показан выбор режима работы с маяком.

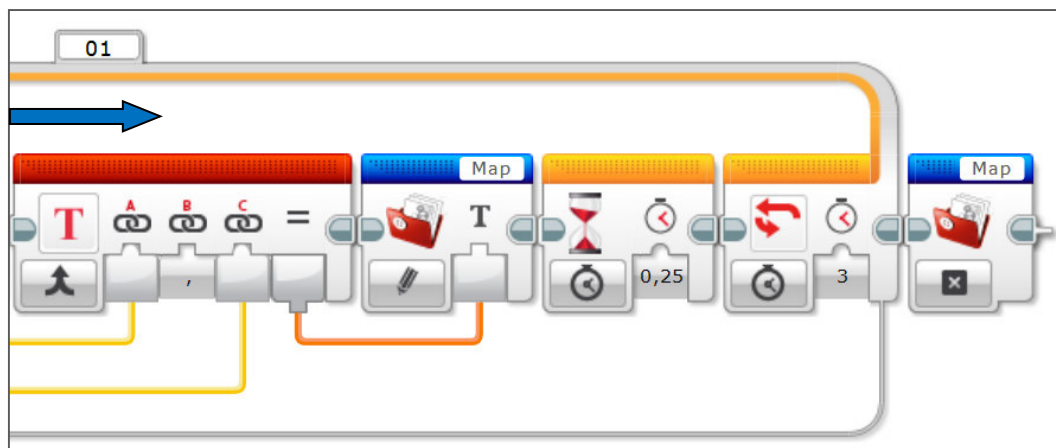
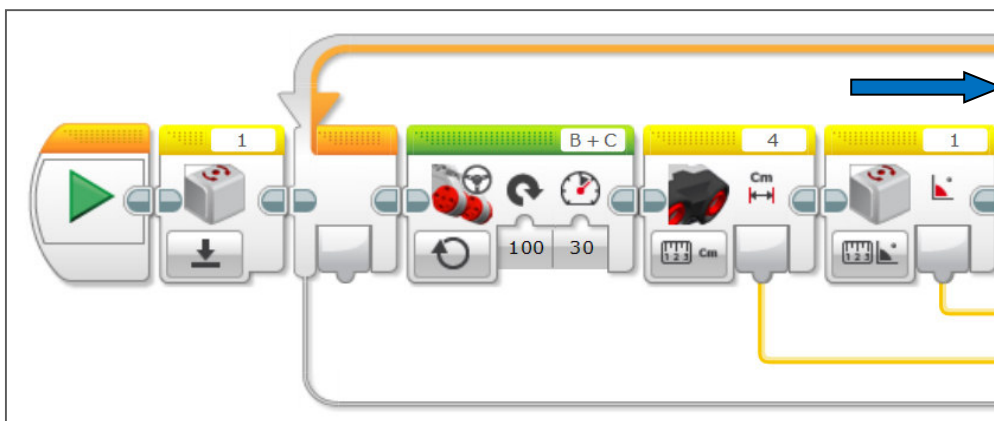
использованием программы MS Excel.

Робот вращается и в каждый момент времени записывает значение угла поворота и расстояние до поверхности.

Решение:

1. Производим сброс значений датчика гироскопа. Вставляем цикл 01, условие завершения – время (3 сек.).
2. В цикле робот вращается и считывает информацию с ультразвукового и гироскопического датчиков. Показания объединяются в программном блоке *Текст*, разделяясь запятой.
3. Результат измерений в каждом шаге цикла записывается в файл *Map*.
4. Устанавливаем паузу 0,25 сек. После окончания цикла закрываем файл.

Внимание! Используя гироскопический датчик, при включении обращайте внимание на наличие дрейфа показаний (см. п. 2.5.3 для удаления дрейфа).



движения и остановке подавайте последовательно каждому вагончику.

2. «Вокально-инструментальный ансамбль»

Задача – исполнение музыкального произведения с ансамблем. Первый робот EV3 – дирижёр, который раздаёт команды по Bluetooth остальным роботам-музыкантам и роботам-певцам, когда воспроизводить их музыкальные партии. Робота-дирижёра можно оснастить дирижёрской палочкой,двигающейся вверх-вниз и поворачивающейся в сторону робота, который начинает воспроизведение. Робот-дирижёр по совместительству может, например, солировать.

3. «Танцевальный ансамбль»

Задача – создание робо-ансамбля. Первый робот, раздающий команды по Bluetooth – солист. Остальные роботы отрабатывают команды. Прографируйте различные виды танцев – хоровод роботов («Паровозик»), медленные и быстрые танцы.

4. «Утренняя гимнастика»

Задача – одновременное выполнение по команде первого робота гимнастических упражнений.

2.8. Полезные блоки и инструменты

2.8.1. Блок «Поддерживать в активном состоянии»

По истечении определённого времени и в случае, когда мы не обращаемся к роботу, а робот не выполняет никаких операций, он выключается (в терминах EV3 – переходит в спящий режим). Это вызывает неудобство при работе с программами, рассчитанными на длительные ожидания каких-либо процессов. Мы можем установить время перехода в спящий режим непосредственно на блоке (существует возможность установить время до выключения: 2 мин,

создать несколько вариантов программ и выбирать тактику поединка. Например, если перед нами мощный, но медленный робот, можно запустить программу, по которой наш робот будет атаковать соперника быстро и сбоку; если робот соперника в поисках нашего робота всегда поворачивается направо, нужно запускать программу, объезжающую и атакующую его слева.

Робот может иметь один или два датчика ультразвука, чтобы определять положение противника без лишних поворотов. Особенно интересными получаются раунды, в которых соревнуются примерно равные по силе или скорости роботы, в этом случае исход решают миллиметры и секунды!

Победителем становится тот участник, который смог собрать крепкую и надёжную конструкцию, написал грамотную программу (или программы) и выбрал правильную стратегию. Именно сочетание этих факторов делают процесс подготовки к состязаниям увлекательным, а сами соревнования очень зрелищными и захватывающими!

Приведём пример алгоритма программы для робота-сумоиста.

1. Робот ждёт три секунды, далее со скоростью 75 единиц едет до тех пор, пока датчик цвета не увидит чёрную линию. Остановка робота.
2. Робот поворачивается до тех пор, пока не увидит датчиком ультразвука робота соперника (пока значение датчика не станет меньше 100 см), что соответствует углу поворота 120-180 градусов. Остановка робота.
3. Создаём цикл 01, условие завершения – *Неограниченный*.
4. В цикл 01 вставляем цикл 02, условие завершения которого – логическое значение: цикл будет выполняться, пока на вход *Условие завершения* не будет подано значение *Истина*.

$$\frac{100}{V1} = \frac{25 + 18}{25}$$

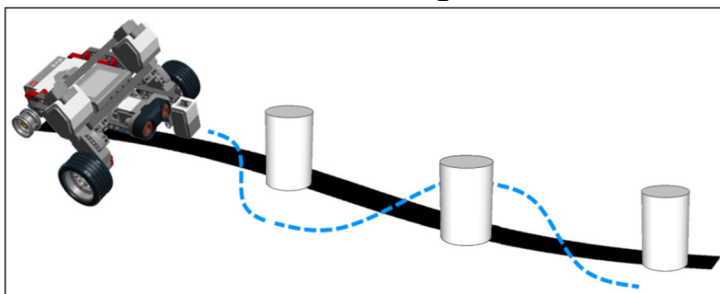
Найдём скорость левого колеса: $V1=58$.

Для реализации алгоритма установите впереди робота по центру ультразвуковой датчик и подключите его в порт 4. Вниз направьте датчик цвета, расположите его *слева от линии* и подключите в порт 2. На рис. 3.3.3 представлена программа объезда препятствия. Обратите внимание, что после обнаружения препятствия робот останавливается и резко поворачивает направо для того, чтобы съехать с линии, перпендикулярной препятствию, и объехать предмет по заданному радиусу.

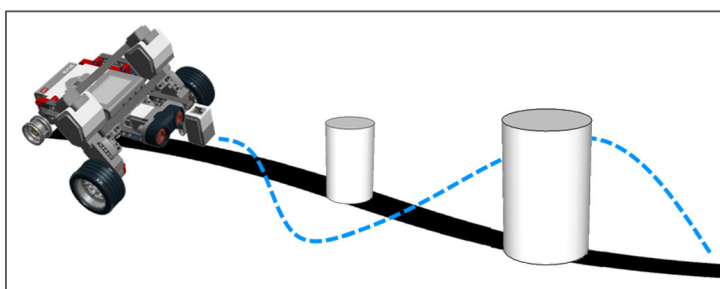
Задания для самостоятельной работы

Запрограммируйте траектории:

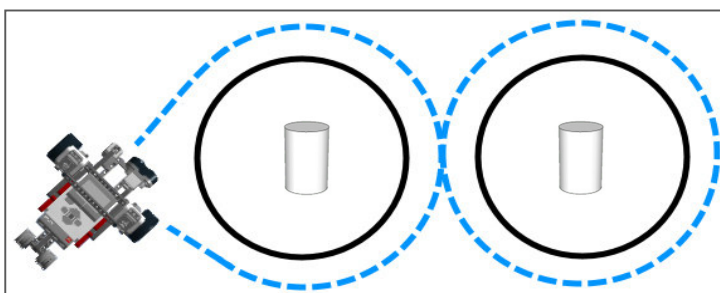
а) объезд нескольких препятствий с одним радиусом;



б) объезд препятствий с разными радиусами;



в) езда «восьмёркой».



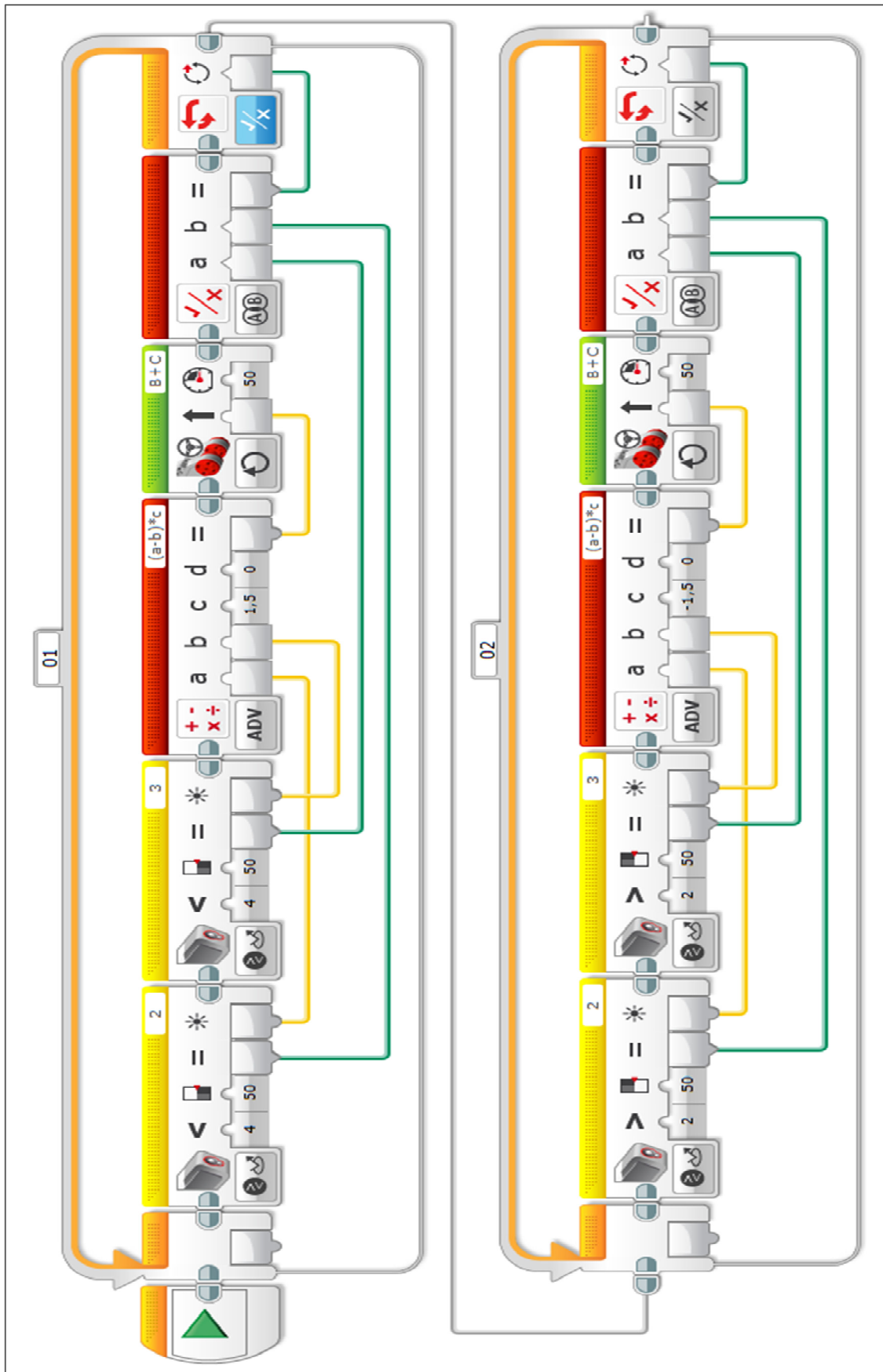


Рисунок 3.5.3.2. Программа проезда инверсной траектории движения

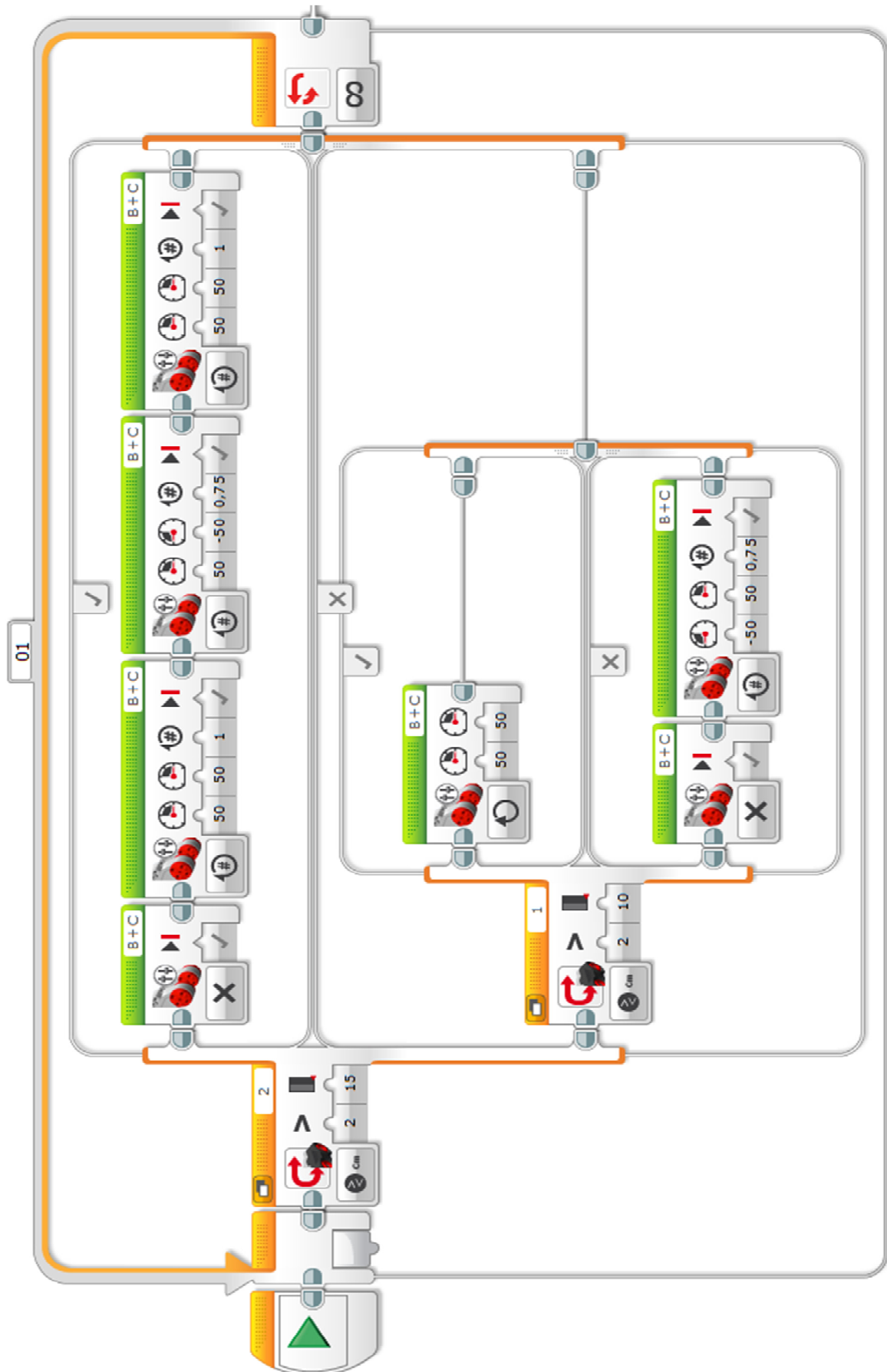


Рисунок 3.6.5. Программа поиска цели в лабиринте

ГЛАВА 5. ИСПОЛЬЗОВАНИЕ СТОРОННИХ ДАТЧИКОВ

Помимо датчиков, входящих в домашний или образовательный комплекты Lego Mindstorms EV3, существует возможность дополнительного приобретения датчиков Lego и сертифицированных Lego датчиков компании HiTechnic.

Компания HiTechnic производит большое количество датчиков для LEGO Mindstorms, большинство из них сертифицированы компанией LEGO, подтверждая полную совместимость, высокие стандарты качества и безопасности. Также немаловажным фактором, позволяющим использовать эти датчики при работе с детьми, является наличие сертификата RoHS (Restriction of Hazardous Substances), подтверждающего отсутствие использования в электрическом и электронном оборудовании веществ: свинец, ртуть, кадмий, олово, шестивалентный хром, некоторые бромидные соединения. Актуальный список сертифицированных для Lego датчиков можно найти на сайте www.hitechnic.com/sensors.

В настоящее время доступны: датчик угла поворота; силы, приложенной поперечно к оси; компас; акселерометр; гироскоп; детектор магнитных полей; инфракрасный датчик; инфракрасный датчик движения, позволяющий определять наличие в помещении людей или животных аналогично датчикам, используемым в охранных системах; барометр, определяющий атмосферное давление и температуру; электрооптический датчик расстояния, точно определяющий небольшие объекты и малые изменения в дальности до них, но на расстоянии не более ~20 см; датчик цвета.