Движение по черной линии с сохранением маршрута

Введение

На этом занятии вы:

1. Познакомитесь с устройством датчика освещенности;
2. Напишете программу движения робота с одним датчиком освещенности, по черной линии с использованием пропорционального регулятора;
3. Научитесь использовать массивы для сохранения набора данных.
4. Познакомитесь с операцией логического сложения (операция «или»)

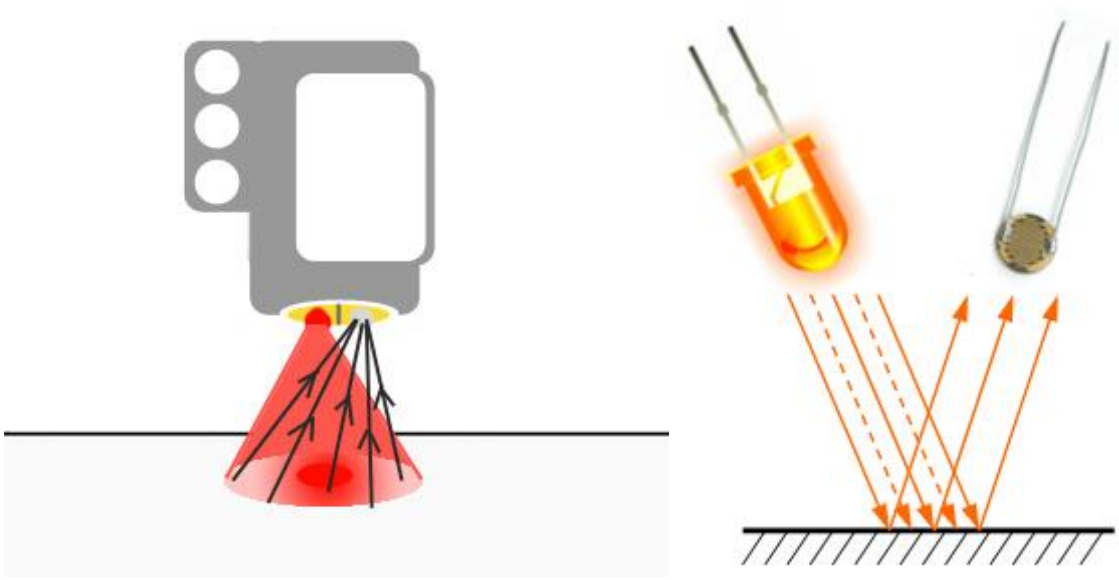
Датчик освещенности

Датчик освещенности из набора Lego Mindstorms NXT, один из наиболее используемых сенсоров при конструировании и программировании Lego-роботов.

Основным элементом в нем является светочувствительный элемент (фоторезистор или фототранзистор).

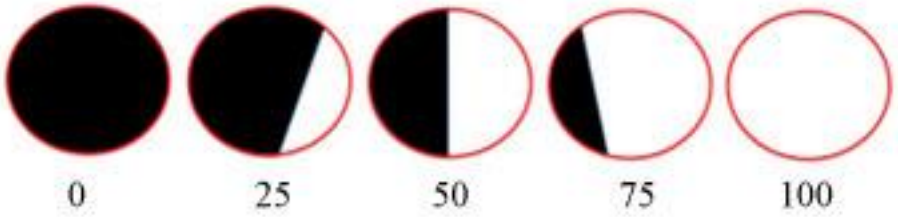
В режиме измерения окружающей освещенности, количество света, попавшее на светочувствительный элемент, преобразуется в цифровое значение, которое уже используется в программе. Например, с датчиком, работающем в этом режиме, можно собрать робота, который ищет самое освещенное место в комнате.

В режиме измерения отраженного цвета, помимо светочувствительного элемента, активируется светоиспускающий элемент (светодиод). Свет, выпущенный этим элементом, отражается от какой-нибудь поверхности и попадает обратно в светочувствительный элемент.



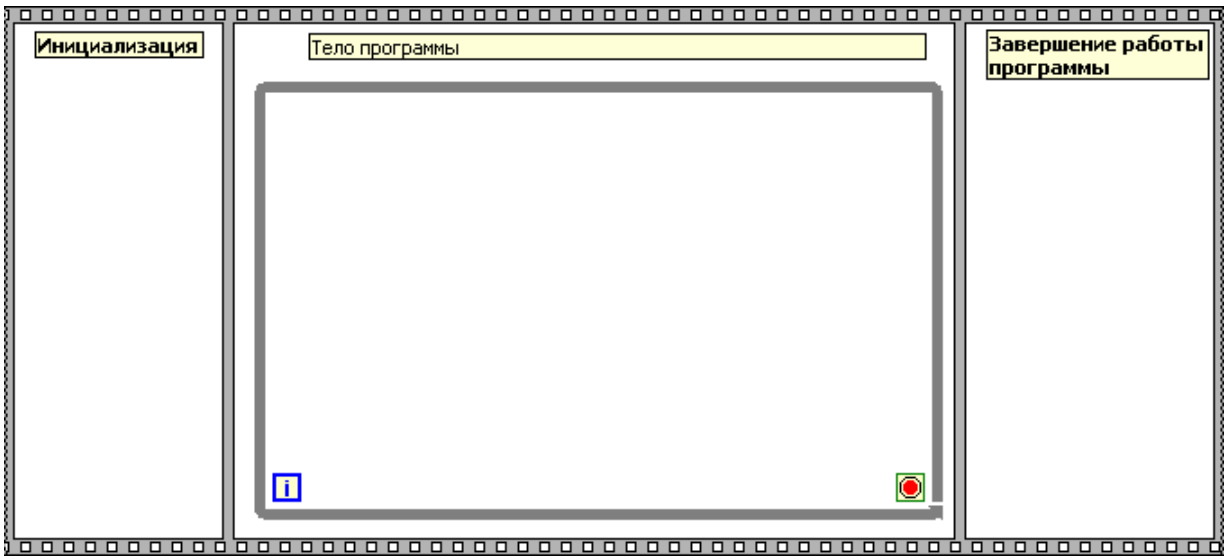
В зависимости от того насколько светлая отражающая поверхность, в светочувствительный элемент приходит больше света. Это количество света преобразуется в цифровое значение и передается в программу. Чем темнее поверхность, тем меньше света приходит – в программу приходят маленькие значения; чем светлее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

Ниже приведен пример различных значений датчика освещенности соответствующих различным вариантам области видимости датчика.



Движение по линии с 1 датчиком освещенности

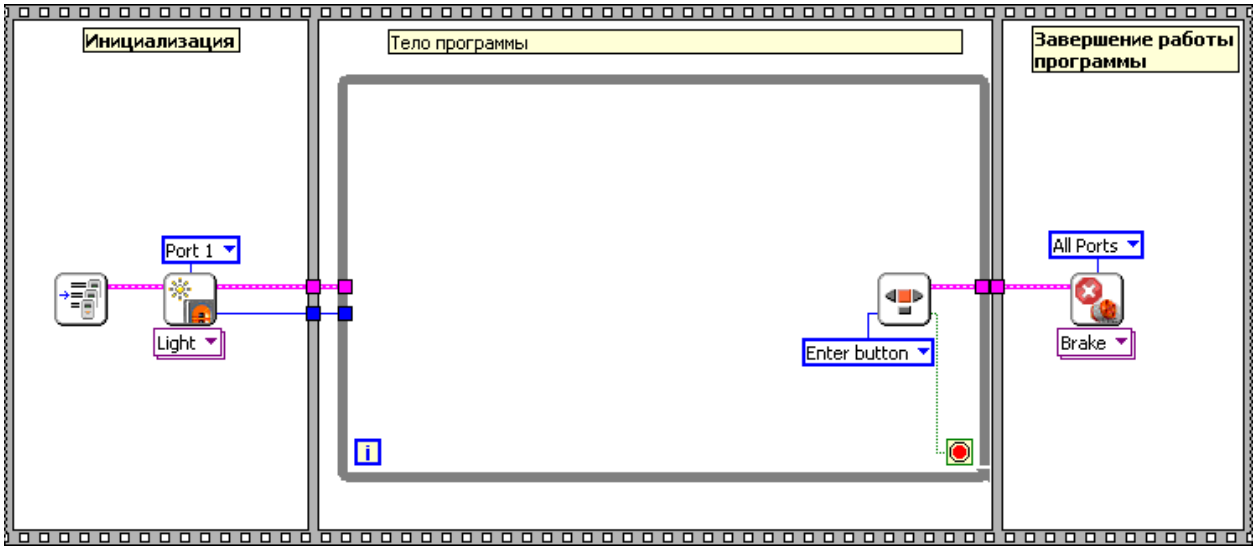
Создаем шаблон программы



В самом начале программы нам нужно провести калибровку датчика освещенности. Калибровкой датчика называется установление зависимости между его показаниями (значением датчика) и измеряемой (входной) величиной. В нашем случае нам нужно зафиксировать значение датчика освещенности соответствующее границе черной линии.

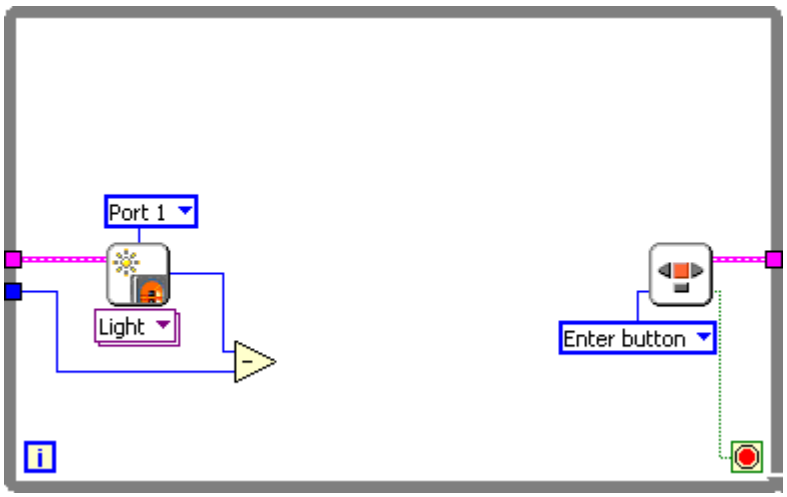
Добавляем функцию чтения датчика **NXT I/O -> Read Sensor,** в меню выбираем **Read Light -> LED on.** Размещаем иконку функции в кадре инициализации после **Specify NXT**  **.**

Также сразу мы добавляем проверку нажатия кнопки enter для завершения основного цикла, и команду выключения моторов в кадр завершения работы.

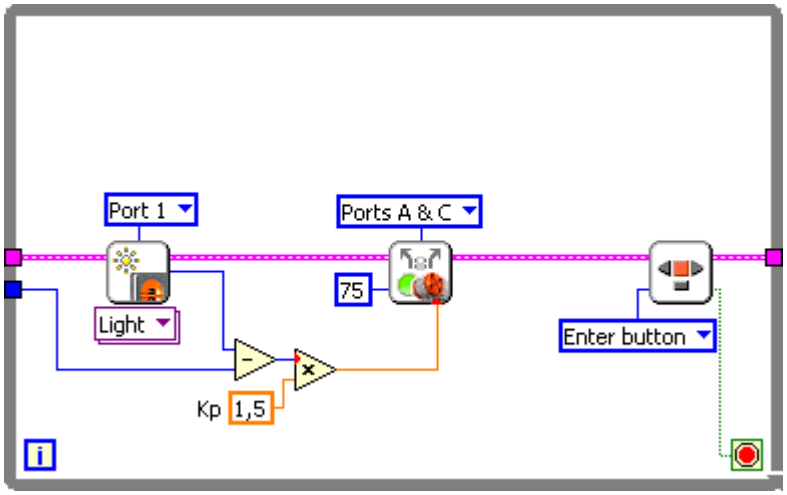


Нашу основную задачу, движение по линии, мы будем рассматривать как задачу позиционирования, только вместо значения енкодера мотора у нас значение датчика освещенности, а задание по положению это значение датчика освещенности на границе линии.

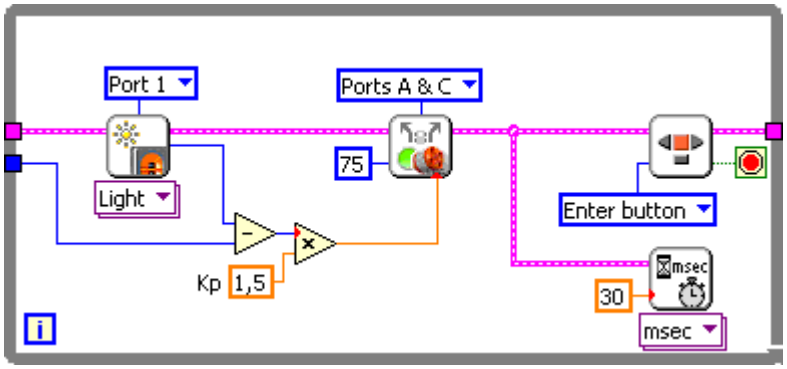
Опрашиваем датчик освещенности в цикле и из текущего значения освещенности вычитаем значение соответствующее границе линии. Разность текущего значения и задания, как мы проходили раньше, называется ошибкой.

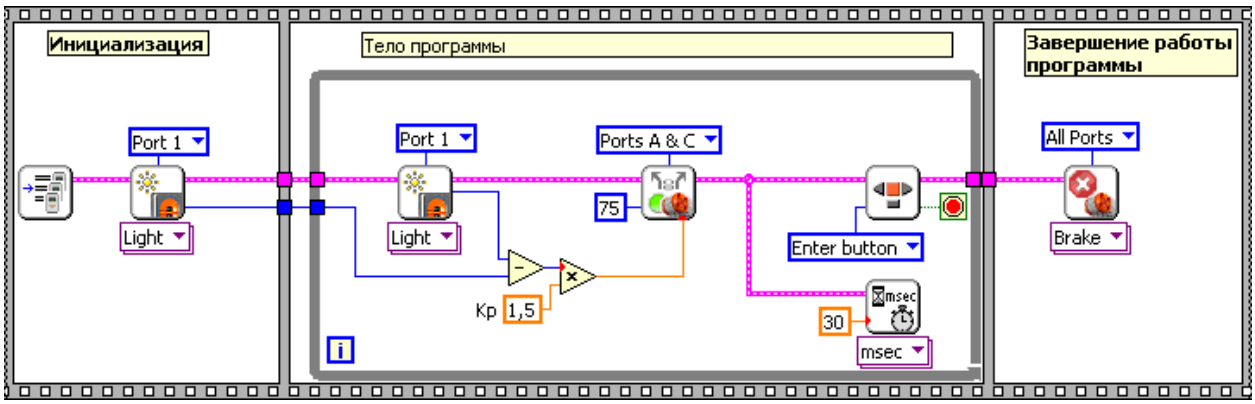


Умножаем ошибку на пропорциональный коэффициент усиления Kp равный 1.5, и результат подаем на вход функции steering on  , **NXT Functions -> NXT I/O -> Complete -> Motors -> Steering On.** Создаем на входах выбора моторов и задания мощности, константы.



Добавим в цикл временную задержку в 30 миллисекунд для облегчения работы программы.





Сохраняем программу.

Записываем ее на NXT и проверяем работу.

ВНИМАНИЕ! Перед запуском программы, робота следует поставить так, чтобы область видимости датчика освещенности находилась на границе черной линии!

Далее необходимо подобрать значение пропорционального коэффициента усиления Kp (текущее значение 1.5) и мощности (текущее значение 75) для обеспечения гладкого движения робота по черной линии.

ВНИМАНИЕ! Если вы подобрали выше указанные значения для гладкого перемещения по линии, то нужно понимать что если вы будете использовать робота с другой конструкцией, то значения Kp и мощности возможно придется подбирать заново.

Запоминание маршрута

Добавим в нашу программу сохранение маршрута движения робота по линии, чтобы он мог повторить траекторию движения без линии.

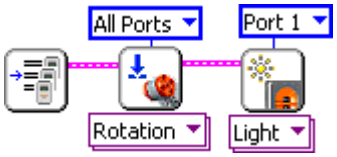
Берем написанную программу, и сохраняем под новым именем.

Будем вносить изменения последовательно в каждый кадр программы.

Кадр инициализации

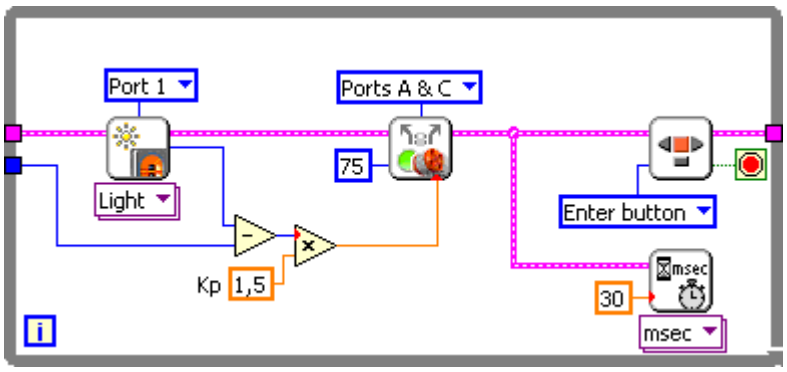
Нам нужно обнулить значения энкодеров моторов. Именно при помощи энкодеров мы будем запоминать положение робота на маршруте и стартовое положение робота будет соответствовать нулю обоих энкодеров.

Добавляем в кадр инициализации функцию чтения датчика **NXT I/O -> Read Sensor,** выбираем в меню под иконкой функции **Reset Motor,** и указываем что нам нужно обнулить энкодеры по всем портам**.**



Кадр тело программы

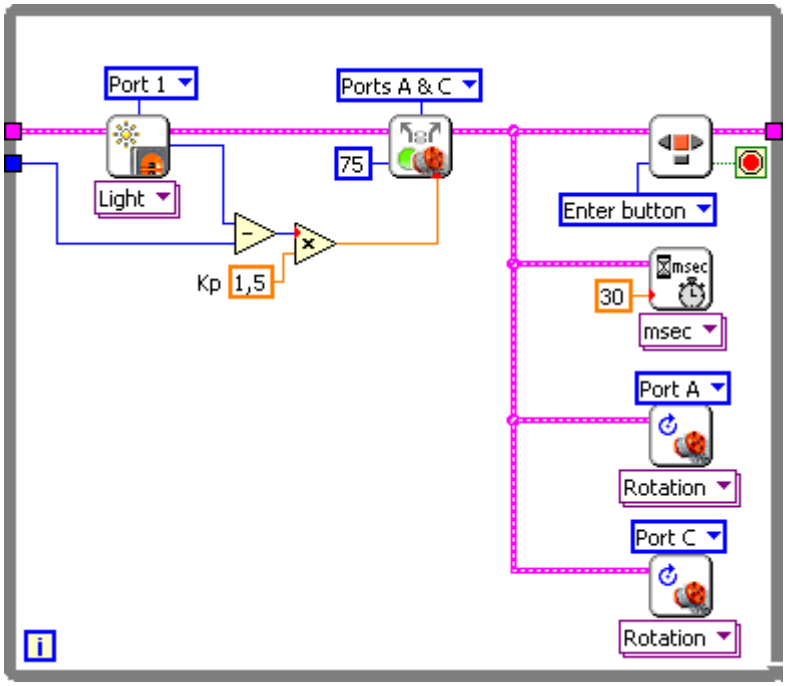
Сейчас главный цикл нашей программы должен выглядеть так:



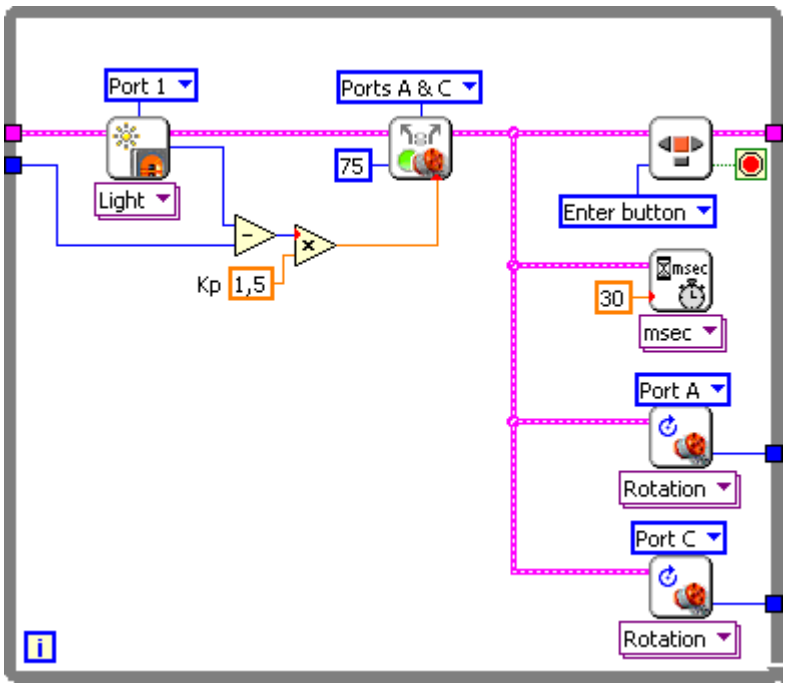
Тело нашей новой программы будет состояить из 2 этапов,:

1. Тело программы;
    a. Проезд по линии и запоминание маршрута;
    b. Повторение маршрута без линии;

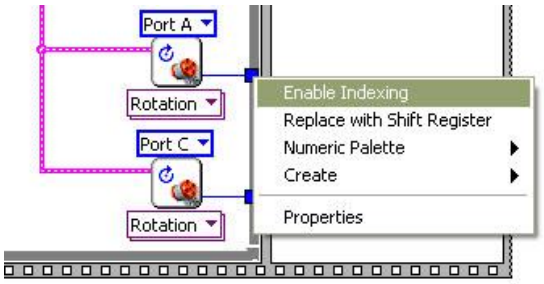
Добавляем опрос значений энкодеров моторов **NXT I/O -> Read Sensor-> Read Rotation** при этом подключаем их не последовательно остальным функциям цикла а параллельно временной задержке и опросу кнопки Enter.



Выходы с текущими значениями функций **Read Rotation** подключаем на выход из цикла через туннель.



Кликаем правой кнопкой мышки на одном из выходных туннелей и выбираем **Enable Indexing** (включить индексацию).



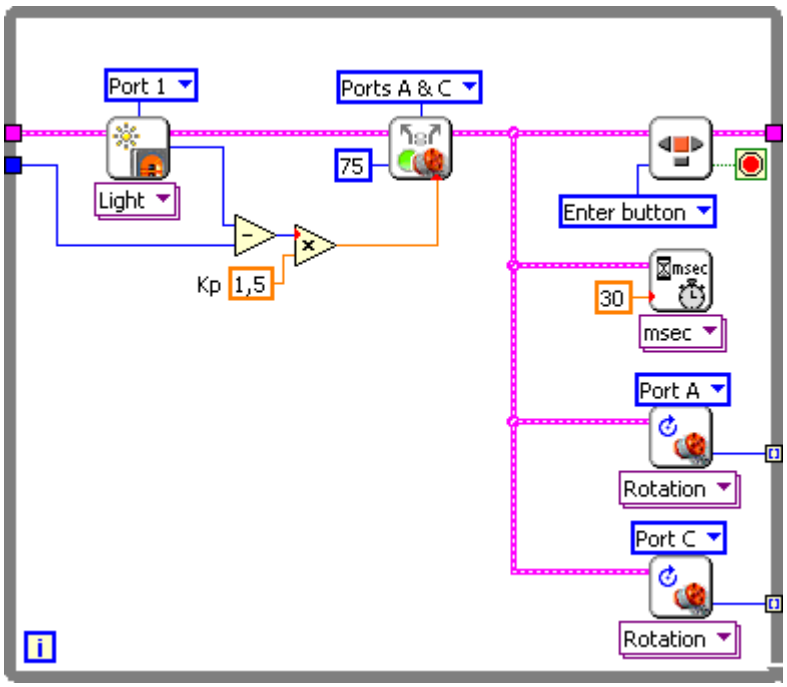
Туннель поменяет свой внешний вид с ■ на ▣

Теперь на выходе этого туннеля мы получим не одно значение, а массив значений, каждое из которых будет соответствовать определенной итерации цикла.

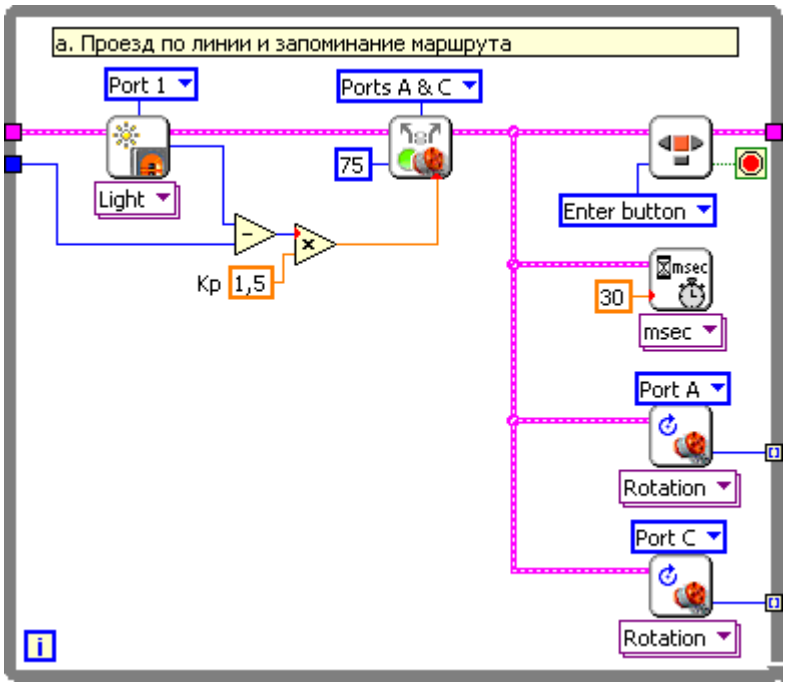
Массивом называется именованный набор однотипных данных, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексу (порядковому номеру).

К примеру, если, пока робот ехал по черной линии, цикл повторился 10 000 раз, то на выходе этого туннеля будет массив из 10 000 значений.

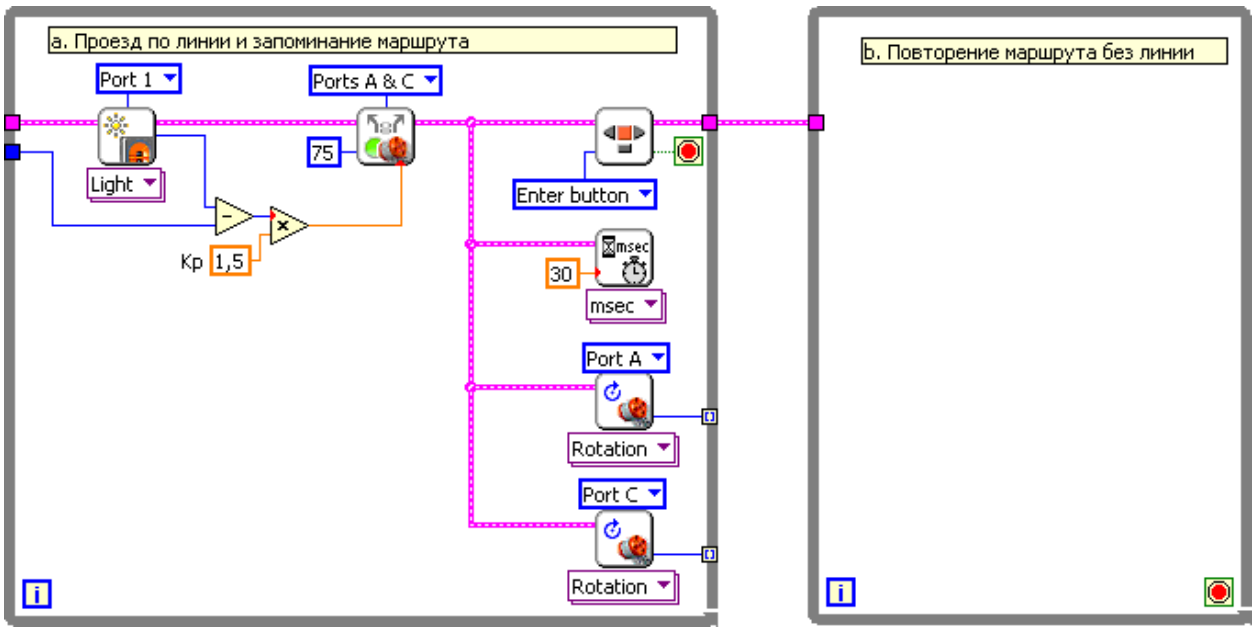
У второго туннеля также включаем индексацию.



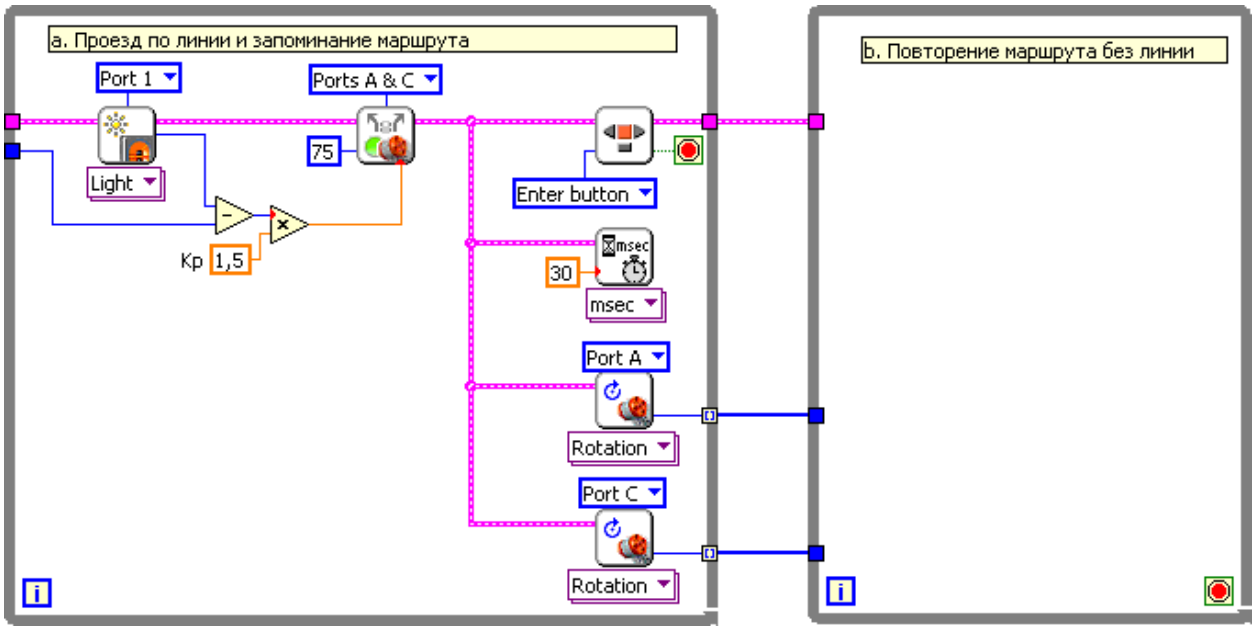
Назовем цикл «a. Проезд по линии и запоминание маршрута», далее будем называть его цикл «a».



Создадим в теле программы еще один цикл, последовательно за циклом «a», и назовем его «b. Повторение маршрута без линии» (далее цикл «b»).

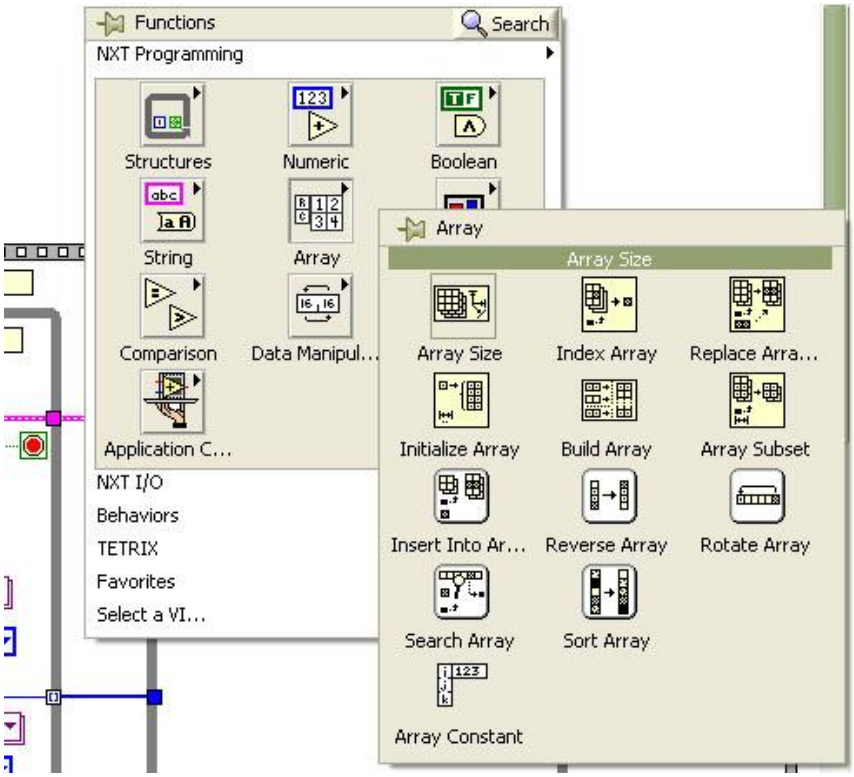


Подключаем выходы туннелей из цикла «a» на вход цикла «b»

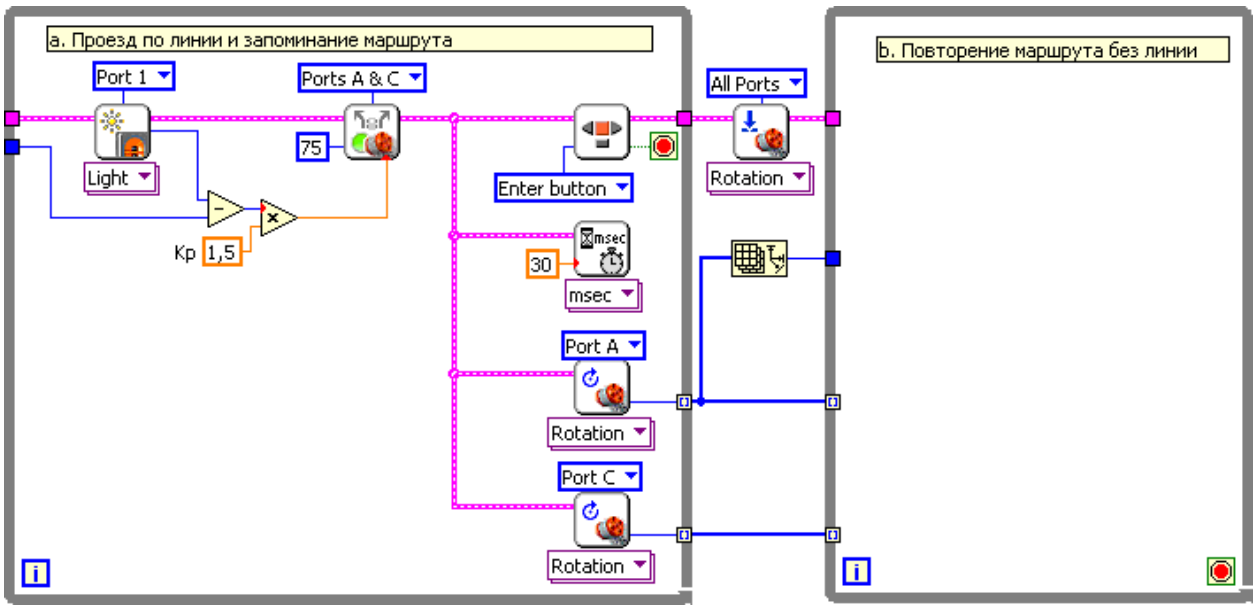


У входных туннелей цикла «b», включаем индексацию, их вид также должен поменяется с ■ на ▣. Таким образом значение на выходе входного туннеля будет соответствовать значению из массива с индексом текущей итерации. Например, если возьмем тот же массив из 10 000 значений, и в данный момент будет выполнятся 341 итерация цикла «b», то на выходе индексированного входного туннеля будет 341-ое значение из массива.

Между циклами добавляем обнуление значений энкодеров и функцию **Array size**  **NXT Programming -> Array-> Array size.**

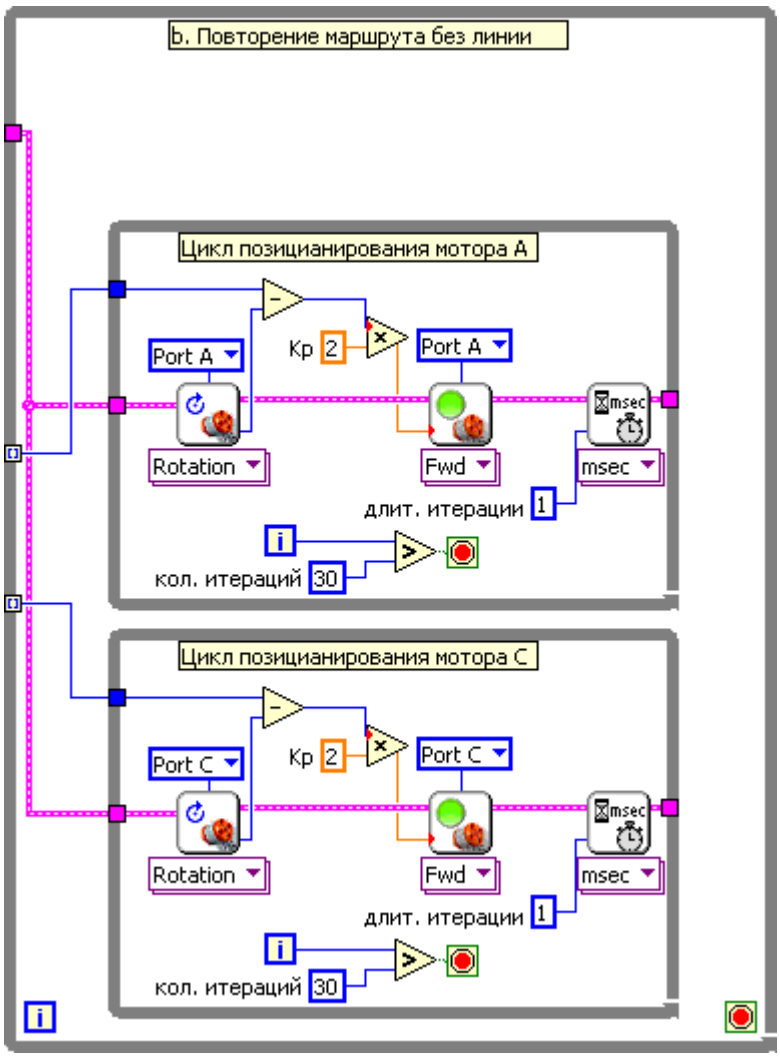


Ко входу этой функции подключаем один из массивов со значениями энкодера, а выход заводим через туннель в цикл «b».



На выходе функции **Array size**, мы получим количество значений содержащихся в массиве который подключен ко входу функции.

Внутри цикла «b» создаем цикл позиционирования мотора A и цикл позиционирования мотора C.



В качестве задании по положению подключаем значения из массивов соответствующие текущей итерации.

Каждый цикл позиционирования должен иметь одинаковые параметры, за исключением портов подключения моторов:

- Коэффициент усиления Kp равен 2;
- Количество итераций каждого цикла позиционирования равно 30;
- Длительность итерации 1 миллисекунда.

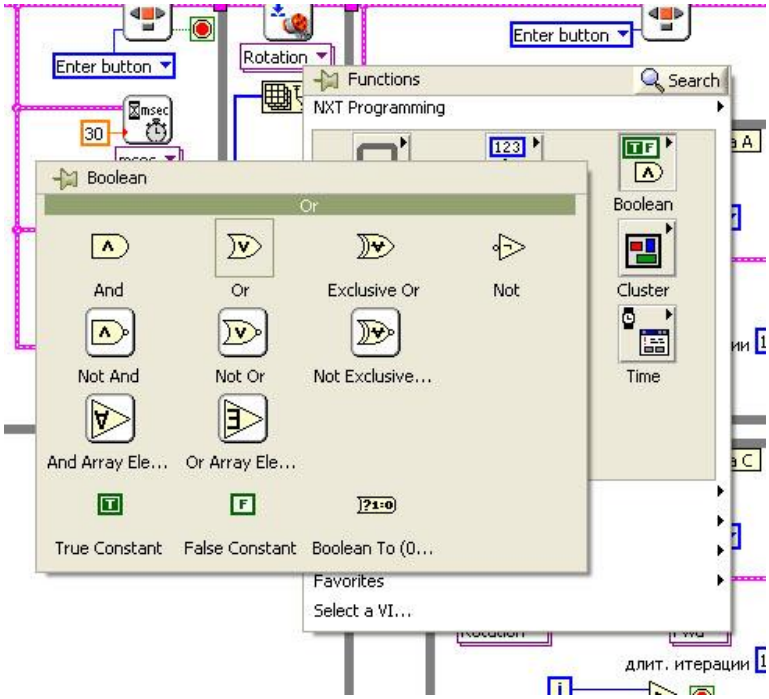
Осталось определить условия завершения цикла «b».

Условий будет два, первое по нажатию кнопки Enter, а второе по завершению маршрута, то есть когда закончатся все значения в массивах.

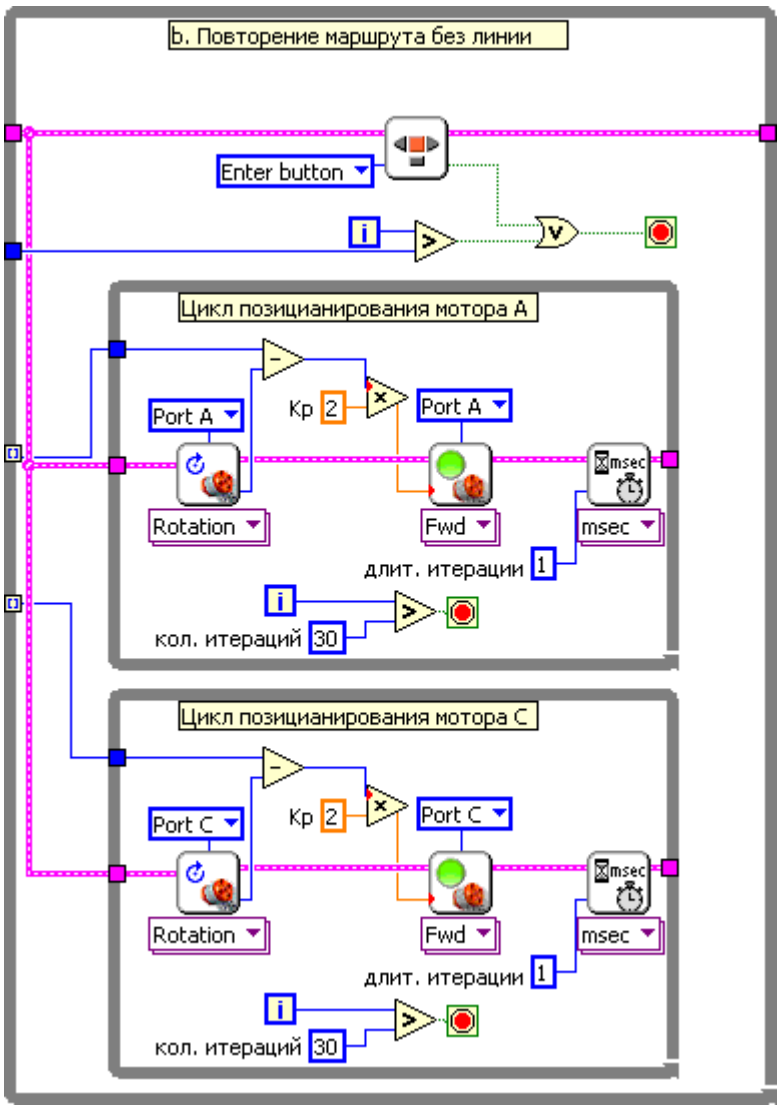
Добавляем опрос кнопки Enter **NXT I/O -> Complete -> Sensors -> Read NXT Buttons** и сравнение  **greather?**(больше?) **Functions -> comparison-> greather?.**

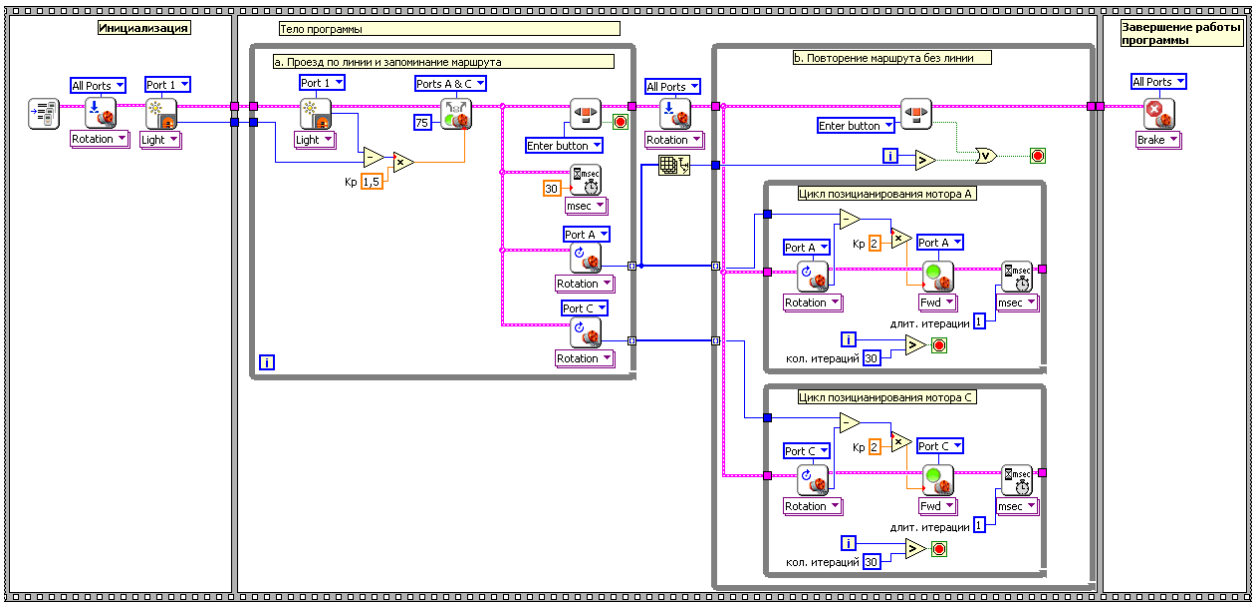
К верхнему входу сравнения **greather?** Подключаем номер текущей итерации, а к нижнему входу туннель от функции **array size.**

Далее нужно объеденить оба условия, для этого воспользуемся функцией логического сложения, операция **Or** **(или) NXT programming -> Boolean -> Or**



Данная функция на выходе имеет значение **True**, если хотя бы одно из входных значений имеет **True.**





Программа готова! Проверяем работу программы.