Ярмолинский Леонид Маркович

УРОК 4. ВОСПРОИЗВЕДЕНИЕ ПРОЙДЕННОЙ ТРАЕКТОРИИ

ВВЕДЕНИЕ

На этом занятии мы разберем популярную задачу – движение двухколесного робота по черной линии. Также мы научим нашего робота запоминать пройденную траекторию и повторять ее без линии.

Для этого занятия нам понадобиться робот с двумя датчиками освещенности и кабель для записи программы на NXT.

Начнем с написания программы для робота с одним датчиком освещенности.

ДВИЖЕНИЕ ПО ЛИНИИ С 1 ДАТЧИКОМ ОСВЕЩЕННОСТИ

Создаем шаблон программы (рис. 1), состоящий из трех этапов:

– инициализация,
– тело программы,
– завершение работы программы.

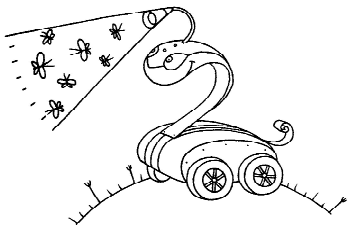
В дальнейшем каждый этап программы будем называть кадром.

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 (рис. 2).

Рис. 1



... нам нужно провести калибровку датчика освещённости...

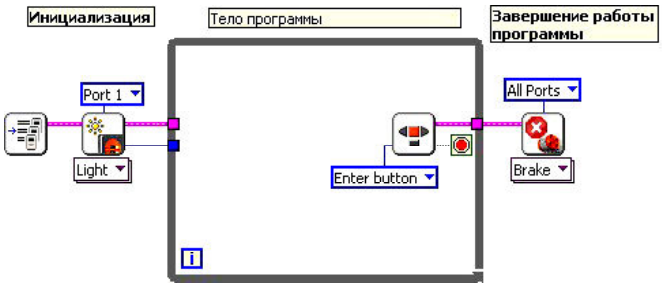


Рис. 2

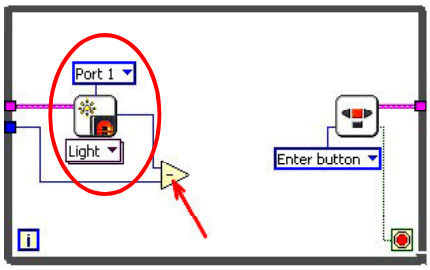


Рис. 3

Нашу основную задачу – движение по линии – мы будем рассматривать как задачу позиционирования, только вместо значения энкодера 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 (рис. 3).

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 (рис. 4).

Добавим в цикл временную задержку в 30 миллисекунд для облегчения работы программы (рис. 5, 6).

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

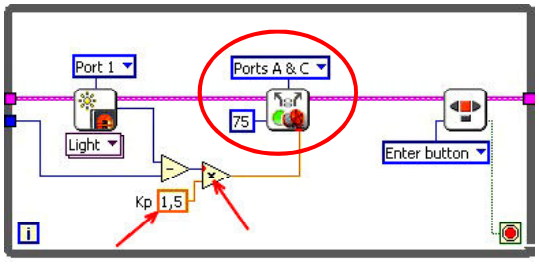


Рис. 4

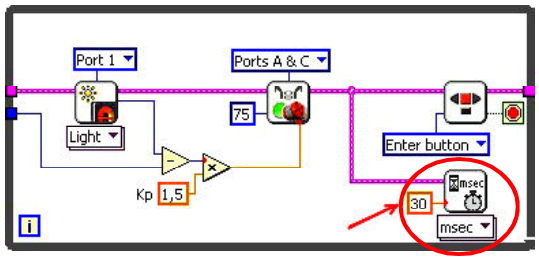


Рис. 5

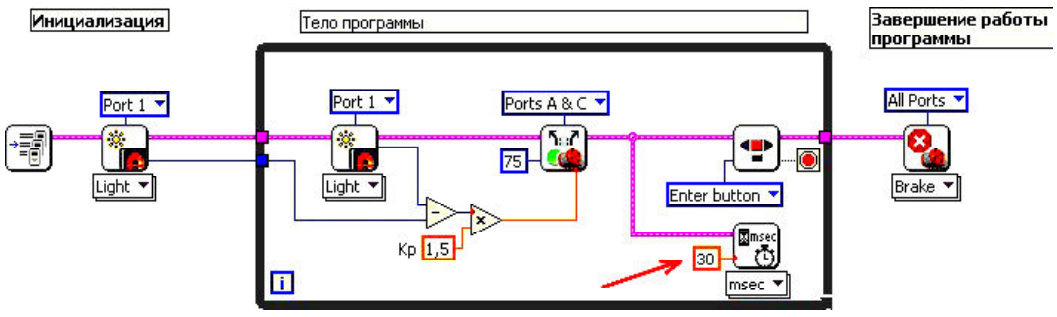


Рис. 6

Kp (текущее значение 1.5) и мощности (текущее значение 75) для обеспечения гладкого движения робота по черной линии.

ВНИМАНИЕ! Если вы подобрали вышеуказанные 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 (рис. 7).

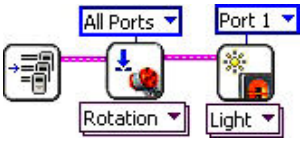


Рис. 7

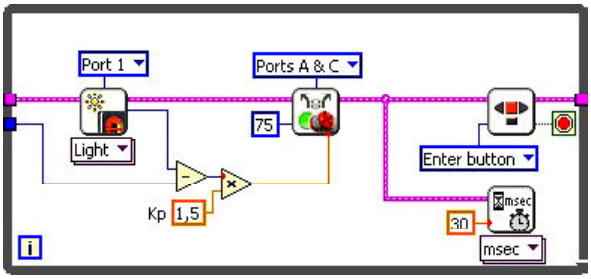


Рис. 8

Кадр тело программы

Сейчас главный цикл нашей программы должен выглядеть так (рис. 8).

Тело нашей новой программы будет состоять из 2 частей:

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 (рис. 9).

Кликаем на рамку цикла справа от функции **Read Rotation port A** и создаем сдвиговый регистр, выбрав пункт меню **Add shift register** (рис. 10).

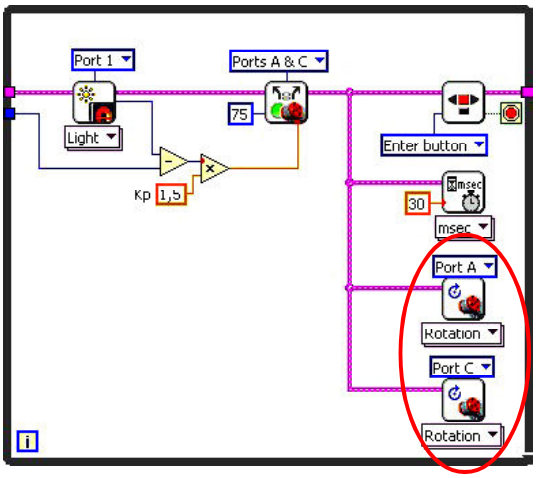


Рис. 9

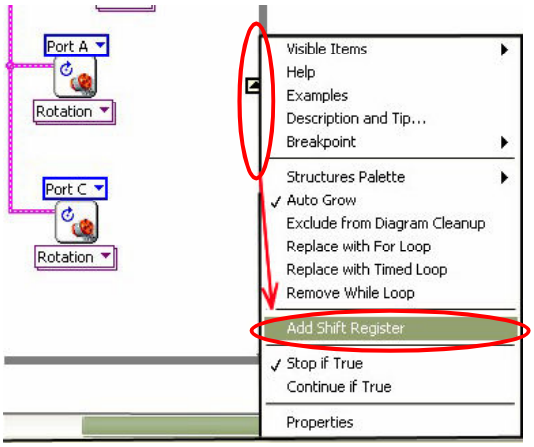
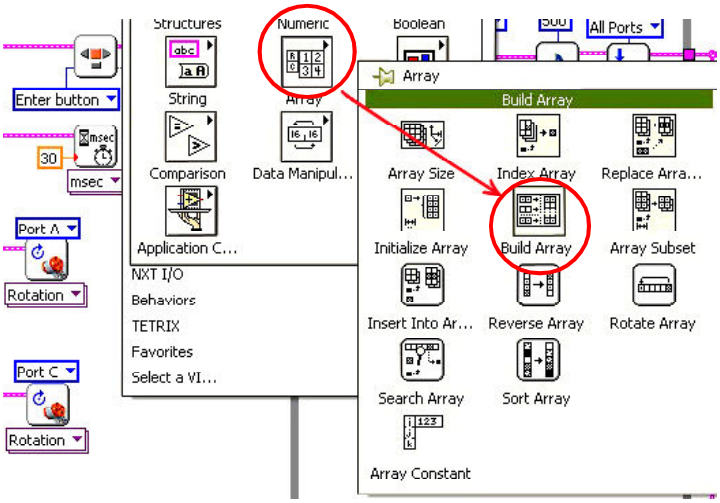
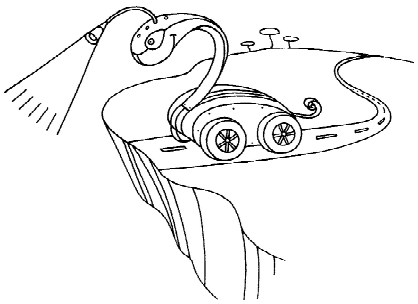


Рис. 10





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

Рис. 11

Сдвиговый регистр позволяет передавать записанные в него значения между итерациями цикла.

Теперь добавляем функцию построения массива **Array → Build array** (рис. 11).

Курсором мыши захватываем нижний край функции **Build array** и растягиваем ее на 2 входа → . Выход функции присоединяем на вход сдвигового регистра, к нижнему входу подключаем значение датчика с функции **Read Rotation port A**, а к верхнему входу значение сдвигового регистра предыдущей иттерации цикла (рис. 12).

Таким образом, на выходном терминале сдвигового регистра (правый терминал со стороны выхода из цикла), мы получили массив со значениями положения мотора, записанными в течение движения робота. К примеру, если, пока робот ехал по черной линии, цикл повторился 10 000 раз, то на выходе правого терминала сдвигового регистра будет массив из 10 000 значений.

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

Кликаем правой кнопкой мышки на входной терминал сдвигового регистра (левый терминал, со стороны входа в цикл) и создаем константу , это константа типа числового массива определения начального значения, записанного в сдвиговый регистр для первой итерации цикла. Серое значение внутри рамки означает, что наш массив не содержит никаких данных.

Далее повторяем описанный выше набор функций и связей для функции **Read Rotation port C** (рис. 13).

Назовем цикл «a. Проезд по линии и запоминание маршрута», далее будем называть его цикл «a» (рис. 14).

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

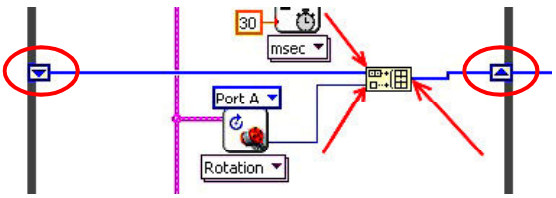


Рис. 12

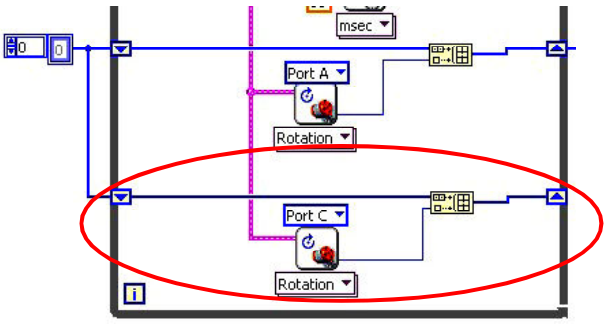


Рис. 13

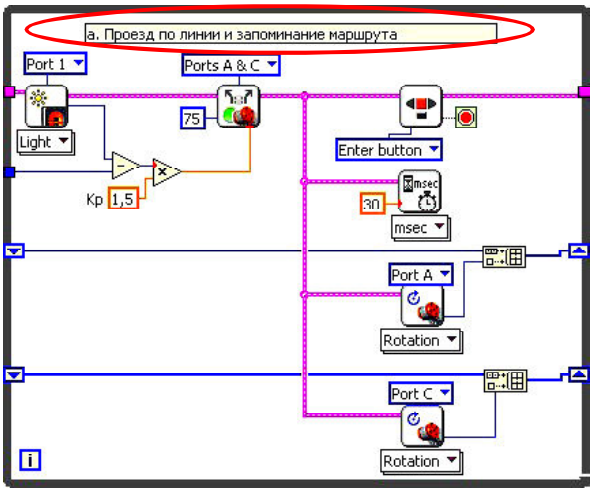


Рис. 14

говых регистров цикла «a» на вход цикла «b».

Также между циклами добавляем звуковую индикацию перехода от одной части программы к другой, при помощи функций **NXT I/O → Sound Control**  и **NXT I/O →**

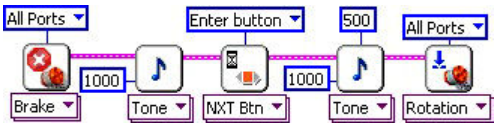


Рис. 15

Wait  **→ Wait for NXT Button**  (рис. 15).

Таким образом, после остановки цикла «a», робот остановит моторы, далее одну секунду будет звучать звуковой сигнал. Выполнение цикла «b» начнется после нажатия оранжевой кнопки «Enter» на NXT и еще одного звукового сигнала в течение одной секунды (рис. 16).

Между циклами добавляем функцию **Array size**  **NXT Programming → Array → Array size** (рис. 17).

К входу этой функции подключаем один из массивов со значениями энкодера, а выход заводим через туннель в цикл «b» (рис. 18).

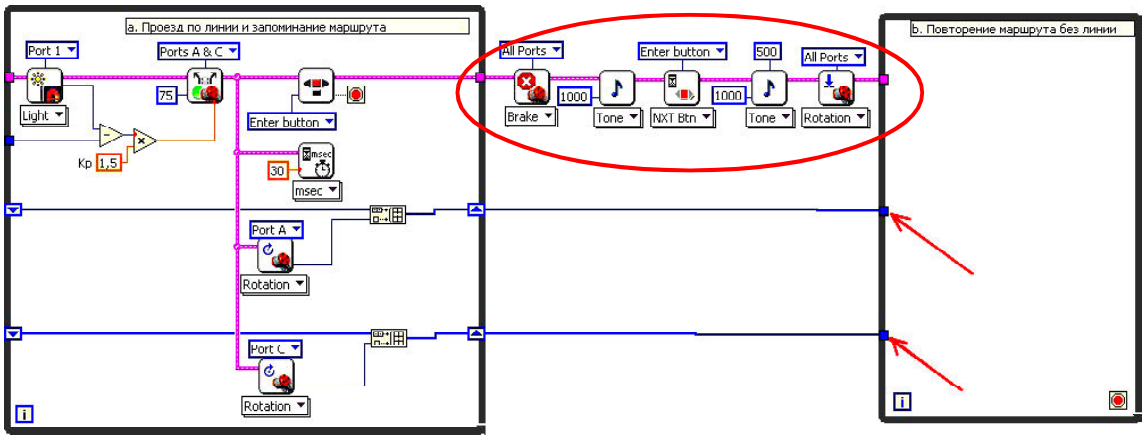


Рис. 16

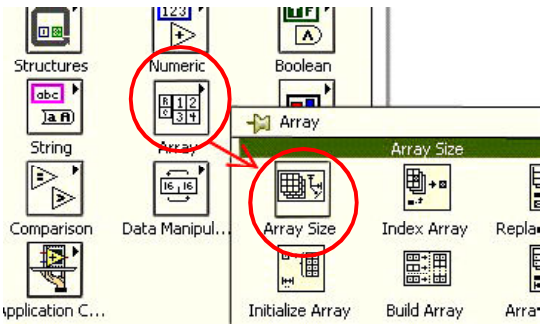


Рис. 17

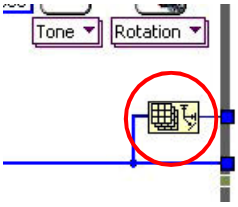


Рис. 18

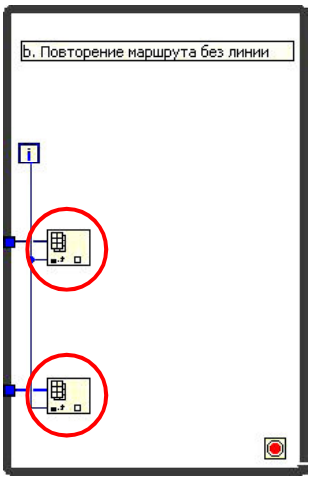


Рис. 19

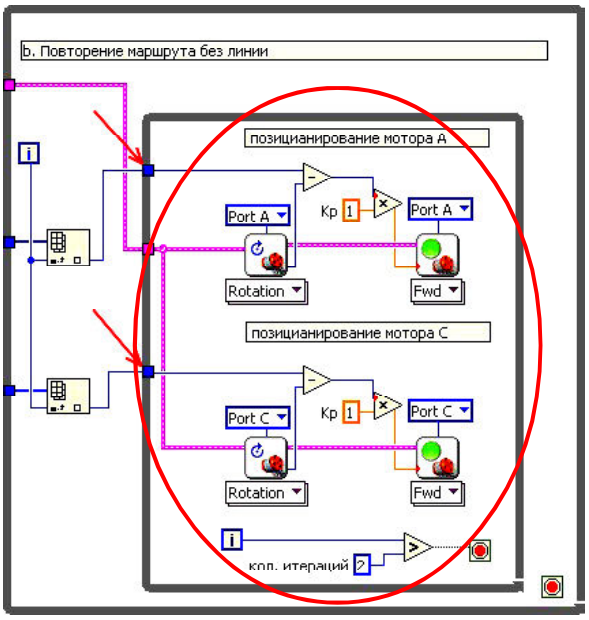


Рис. 20

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

Внутри цикла «b» создаем две функции чтения значения из массива **Array → Index array**  . На верхний вход функций подключаем массивы со значениями энкодеров моторов, а на нижний – значение текущей итерации цикла «b» (рис. 19).

Далее внутри цикла «b» создаем цикл позиционирования моторов A и мотора C (задача позиционирования мотора была рассмотрена в предыдущем номере журнала). В качестве заданий по положению подключаем значения из массивов, соответствующих текущей итерации (рис. 20).

Осталось определить условия завершения цикла «b».

Перейдем к определению условий завершения цикла «b». Условий будет два, первое по нажатию кнопки Enter на NXT, 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** (рис. 21).

Данная функция на выходе имеет значение **True**, если хотя бы одно из входных значений имеет **True** (рис. 22).

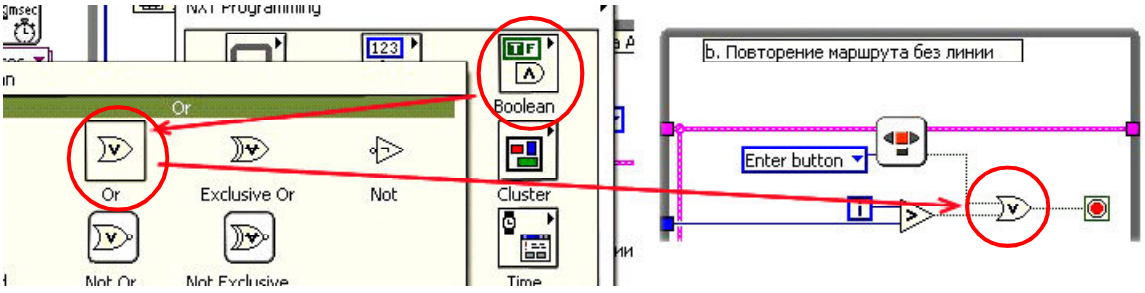


Рис. 21

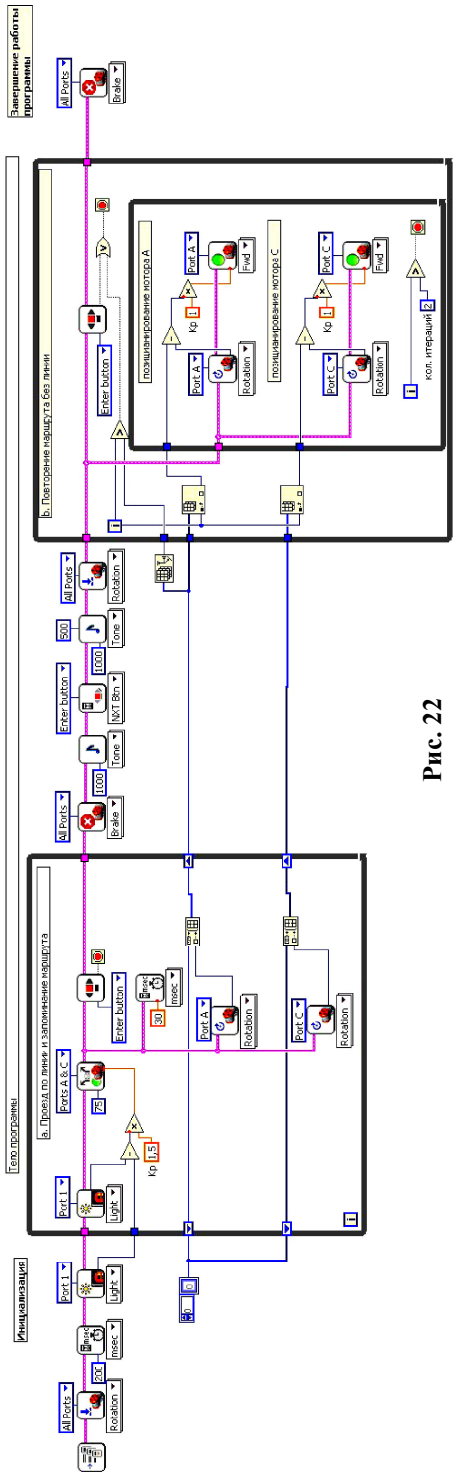


Рис. 22

Программа готова! Сохраняем и проверяем работу программы. Подбирая коэффициенты *Kp* и количество итераций на позиционирование, можно добиться более точной отработки маршрута, а также изменить скорость его воспроизведения. На этом наше занятие заканчивается, успехов!

Ярмолинский Леонид Маркович,
ведущий инженер-программист
ООО «ПромАвтоматика»,
педагог дополнительного образования
ГБОУ СОШ № 255
(кружок робототехники).