Ярмолинский Леонид Маркович

УРОК 6. ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ

ВВЕДЕНИЕ

В этой статье мы рассмотрим возможности связи контроллера NXT по беспроводному каналу Bluetooth. Попробуем различные варианты дистанционного управления и способы передачи данных между устройствами.

СВЯЗЬ МЕЖДУ ДВУМЯ КОНТРОЛЛЕРАМИ NXT

Контроллер NXT поддерживает три одновременных соединения с другими контроллерами NXT. В нашем примере мы ограничимся связью двух NXT контроллеров, поэтому обозначим задачу как передачу какой-либо информации между двумя контроллерами NXT по каналу Bluetooth.

Рис. 1

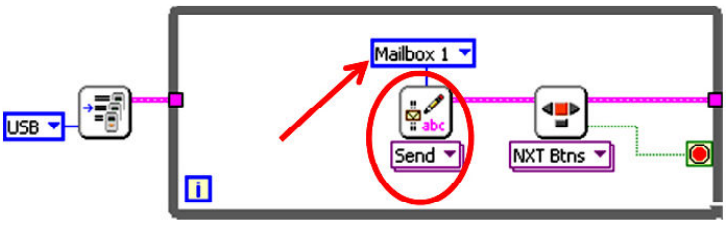


Рис. 2

Начнем с создания шаблона, для программы «Передатчик» (рис. 1).

Также не забываем переключить режим привязки программы к NXT на панель редактирования диаграмм, в главном меню выбираем **NXT file → Target to NXT.**

ПРОГРАММА «ПЕРЕДАТЧИК»

Добавляем функцию передачи сообщения **NXT I/O → Mail**  и в меню выбираем **Send to client → String → Connection 1** – «Передать клиенту строку».

У NXT есть 10 буферов или ячеек памяти для хранения принимаемой и передаваемой информации, в **Labview** данные ячейки названы **Mailbox**, и для передачи информации необходимо указать, в какой **Mailbox** мы планируем произвести запись. Для этого у верхнего входа функции **Send client Mail 1** создаем константу и выбираем у нее значение **Mailbox 1** (рис. 2).

В качестве передаваемой информации возьмем текущий заряд аккумулятора контроллера NXT «Передатчика», для этого добавим функцию **Complete → Sensors → Read Battery**  . Данная функция на выходе выдает уровень напряжения аккумулятора

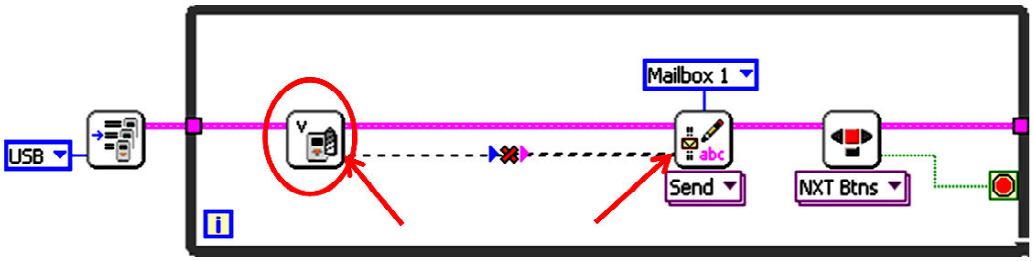


Рис. 3

NXT в милливольтах, но появляется проблема – данный выход имеет числовой тип, а мы собираемся передавать на второй NXT строку. Прямое соединение в нашем случае невозможно из-за различного типа данных.

Из сложившейся ситуации есть два выхода: первый, более простой,– заменить тип передаваемых данных на **Numeric**: в меню функции **Send client Mail 1** выбираем **Send to client → Whole number → Connection 1**

– передать целое число или **Send to client → Decimal number → Connection 1** – передать дробное число (рис. 4).

Для простого случая, когда нам требуется передать только одно число, данный способ подходит, но, когда нам требуется передавать большое количество значений за один раз и при этом разных типов, использование этой функции неудобно.

Мы будем пользоваться более сложной передачей значения заряда аккумулятора, через строку. Для этого в меню функции **Send client Mail 1** возвращаем **Send to client → String → Connection 1** и добавляем функцию преобразования числа в строку **NXT Programming → String → String/Number Conversion → Number to String**

. Подключаем ко входу значение с функции **Read Battery**, а выход передаем на функцию **Send client Mail 1**. Добавим в цикл временную задержку в 30 миллисекунд для облегчения работы программы (рис. 5).

Программа готова, сохраняем ее и записываем на первый NXT.

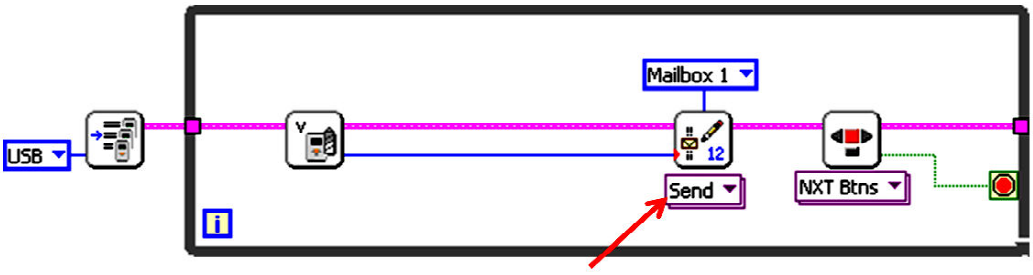


Рис. 4

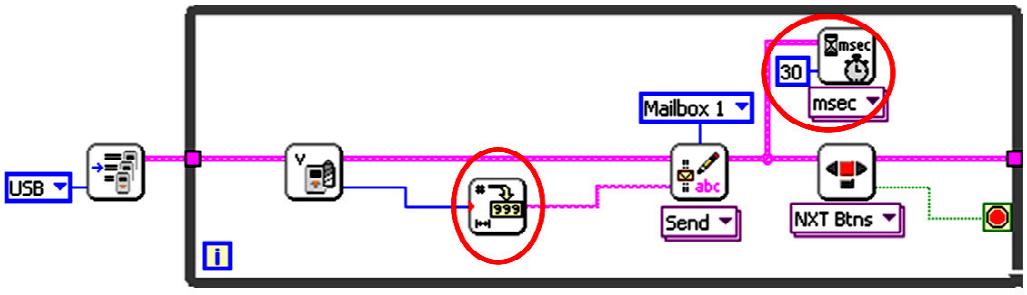


Рис. 5

ПРОГРАММА «ПРИЕМНИК»

За основу для написания программы «Приемник» возьмем программу «Передатчик» и удалим те функции, которые нам не понадобятся (рис. 6).

У функции **Send client Mail 1** в меню выбираем **Read NXT Mailbox → String** 

. Теперь прочитанную строку нужно «распаковать», то есть вытащить из нее значение заряда аккумулятора первого NXT. До «распаковки» надо учитывать следующую особенность: программы на двух NXT не синхронизированы и работают в разном темпе. Может получиться, что «передатчик» один раз послал значение, а приемник за это время произвел два считывания из **Mailbox 1**. В этом случае достоверное значение в программу приемника попадет только при первом считывании, а при втором – прочитается пустая строка. Чтобы избежать чтения недостоверных данных, добавим проверку чтения пустой строки (структура **Case()**), на терминал селектор которой подключен результат сравнения (**NXT programming → Comparison → Equal?** 
) прочитанной строки и пустой строковой константы (**NXT Programming → String → String constant** 
) (рис.7).

Во вкладке **True** не производим никаких действий, а во вкладке **False** добавляем функцию выделения числа из строки **NXT Programming → String → String/Number Conversion → String to Number** 
и результат выводим на дисплей NXT приемника: **NXT I/O → Display Control** в меню **Whrite integer** 

. Можно было бы вывести на дисплей строку напрямую без выделения числового значения, но тогда бы мы пропустили функцию **String to Number** 
, которая нам понадобится в дальнейшем.

Наша программа «Приемник» готова (рис. 8).

Сохраняем программу и записываем ее на NXT-приемник.

Организуем связь (**Pairing**) между NXT-приемником и NXT-передатчиком, так чтобы инициализация связи исходила от NXT-передатчика. После успешного соединения запускаем поочередно программы на одном и на другом NXT. На дисплее NXT-приемника должно появиться значение заряда аккумулятора NXT-передатчика.

Рис. 6

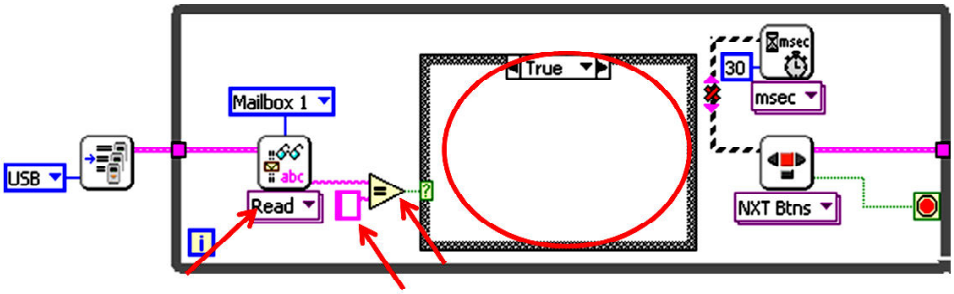


Рис. 7

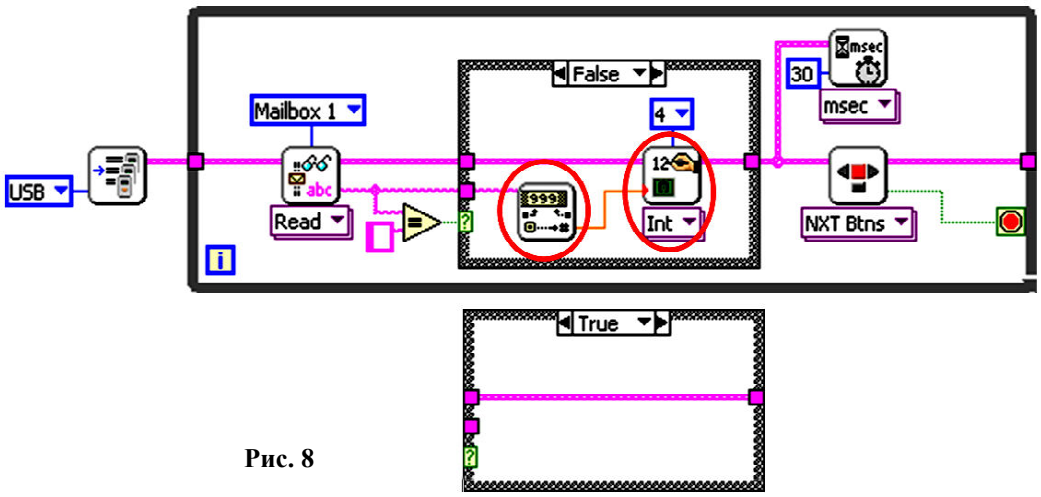


Рис. 8

Рассмотренная программа не имеет прикладного значения, однако она дает понять каким образом можно организовать передачу информации между двумя NXT.

СВЯЗЬ МЕЖДУ КОМПЬЮТЕРОМ И КОНТРОЛЛЕРОМ NXT

В данном случае инициализацию связи будет производить компьютер, и для этого заранее нужно произвести **Paring** между компьютером и используемым NXT. Нужно отметить, что аппаратный адаптер Bluetooth у контроллера NXT довольно «капризный» и совместим с ограниченным списком Bluetooth адаптеров.

Перейдем к нашей задаче. На этот раз нам нужно написать программный комплекс из двух программ, который обеспечит возможность ручного управления роботом, а также перевод робота в автоматический режим следования по линии.

Как и в предыдущей задаче, начнем с программы «передатчика», только теперь она будет называться «Пульт управления» и выполняться на компьютере, а не на NXT.

Переключаем режим привязки программы к компьютеру на панель редактирования диаграмм, в главном меню выбираем к **NXT file → Target to Computer**.

Создаем шаблон для программы в виде машины состояний, рассмотренной в предыдущей статье (рис. 9).

Для создания элемента управления **stop**

на входе терминала завершения цикла кликаем правой кнопкой мышки и в выпадающем меню выбираем **Create control**.

Шаблон готов, теперь перейдем к инициализации связи с контроллером NXT.

В инициализацию добавляем следующие функции: **NXT robotics → NXT I/O → NXT Direct I/O → Connection → Find NXT**

– поиск NXT по имени;

NXT robotics → NXT I/O → NXT Direct I/O → Connection → Create NXT Object

– создание сессии для связи с NXT.

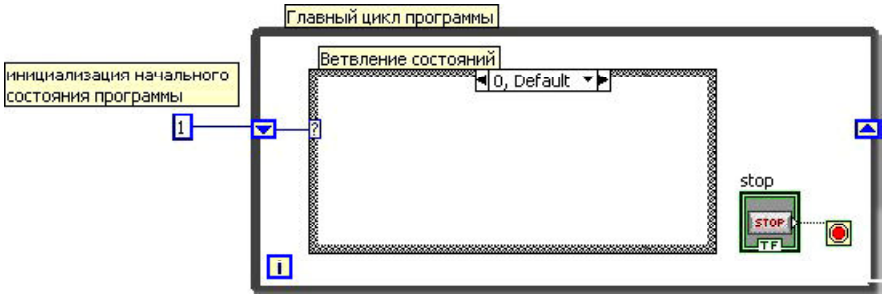


Рис. 9

Соединяем их следующим образом (рис. 10).

У функции **Find NXT** на входах **NXT name** и **Connection type** константы с именем нашего NXT и типом соединения **Bluetooth**. У функции **Create NXT Object** на входе **Bluethooth passkey** создаем константу с паролем Bluetooth соединения компьютера и NXT, который был задан во время процедуры **Paring** (рис. 11).

Добавим защиту от неправильной работы программы в случае отсутствия связи с NXT-контроллером. Вокруг главного цикла программы создадим структуру **Case** и на вход терминала селектора заведем выходной кластер ошибки функции **Create NXT Object**. Рамка структуры **Case** поменяет окраску на зеленый цвет, а в ветвлениях появятся два варианта **No error** и **Error** – действия, выполняемые в случае отсутствия или наличия ошибки на входе (рис. 12).

Если связь с NXT не инициализировалась, то на вход добавленной структуры **Case** придет значение **error** и главный цикл программы не начнет выполняться.

Инициализация соединения и проверка на его успешность готова. Теперь перейдем к описанию состояний в структуре «ветвление состояний». Обозначим следующие состояния нашего пульта управления:

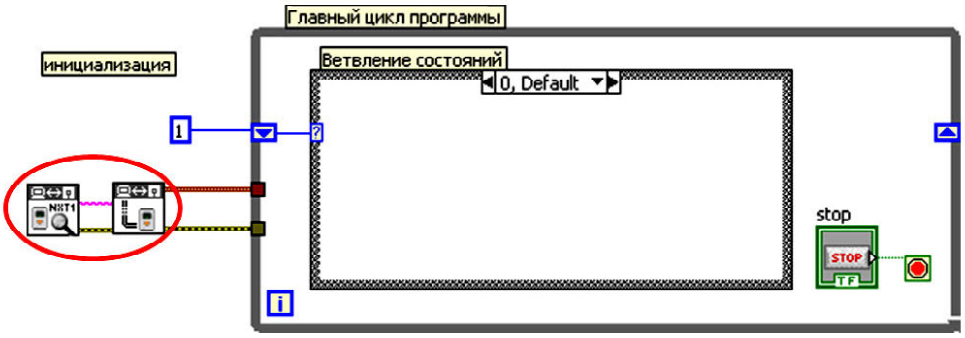


Рис. 10

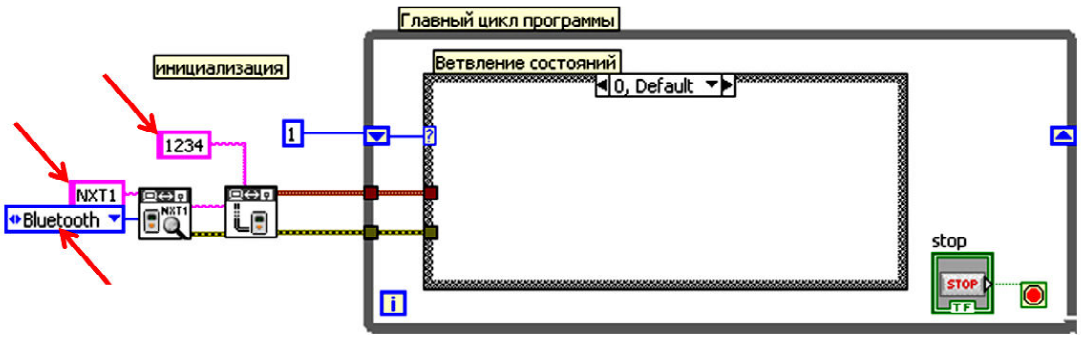


Рис. 11

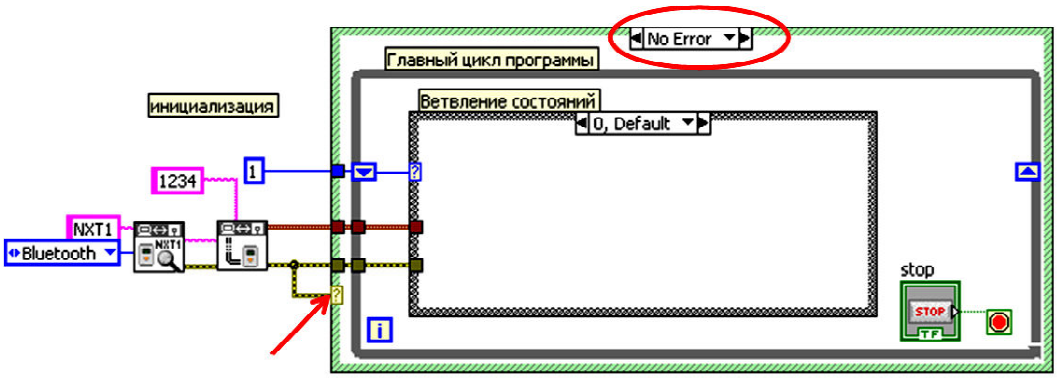


Рис. 12

1 – Прямое управление роботом (возможно при незапущенной программе на роботе).

2 – Управление роботом при помощи отправки команд в исполняемую программу на роботе.

3 – Перевод робота в автоматический режим следования по линии.

ПРЯМОЕ УПРАВЛЕНИЕ РОБОТОМ

У нас установлено прямое соединение с NXT (**Direct control**), поэтому мы можем не только передавать и получать посылки, но и напрямую читать информацию с датчиков контроллера а также управлять двигателями, подключенными к NXT. Используя эти возможности, напишем подпрограмму, позволяющую управлять роботом при помощи стрелок на клавиатуре.

В структуре «ветвление состояний» во вкладке «1» создадим структуру событий **Programming → Structures → Event structure** (рис. 13).

Этот тип структуры похож на структуру **Case**, но вместо терминала селектора для этой структуры могут использоваться различные события, например, движение мышки в определенном месте на экране, нажатие клавиш клавиатуры, кнопок мыши или сворачивание окна программы. В общем, различных событий большое множество, и то событие, для которого описано действие и которое произошло раньше, инициирует выполнение соответствующей ветви данной структуры.

В левом верхнем углу структуры есть изображение песочных часов, у этого входа создаем константу со значением «100». Это

Рис. 13

...если ничего не произошло, то выполнится ветвь Timeout.

означает, что, дойдя до этой структуры, программа будет ждать 100 мс, пока произойдет какое-либо из описанных в структуре событий, и если ничего не произошло, то выполнится ветвь **Timeout**. Если к этому входу подсоединить значение «–1», то эта структура будет бесконечно ждать описанное событие.

Кликаем на полосатую рамку правой кнопкой мышки и выбираем **Add event case** (рис. 14).

В открывшемся окне в среднем поле Event Sources (источник события) выбираем **<This VI>**, а в правом поле **Events** (события) выбираем **Key Repeat** (событие по удержанию клавиши клавиатуры) (рис. 15).

Добавим еще одно событие, для этого нажимаем кнопку **Add Event** 

, и выбираем **<This VI>**

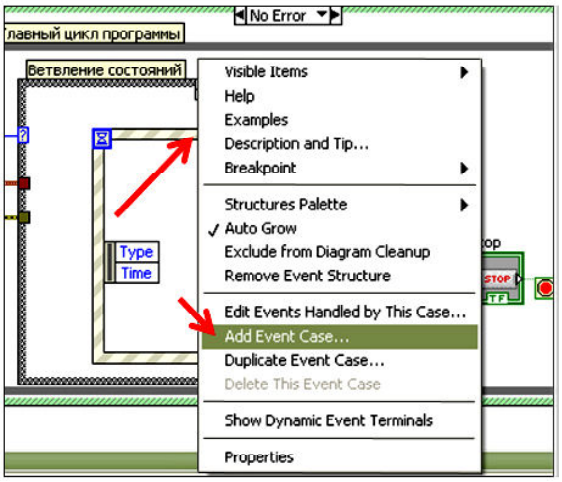


Рис. 14

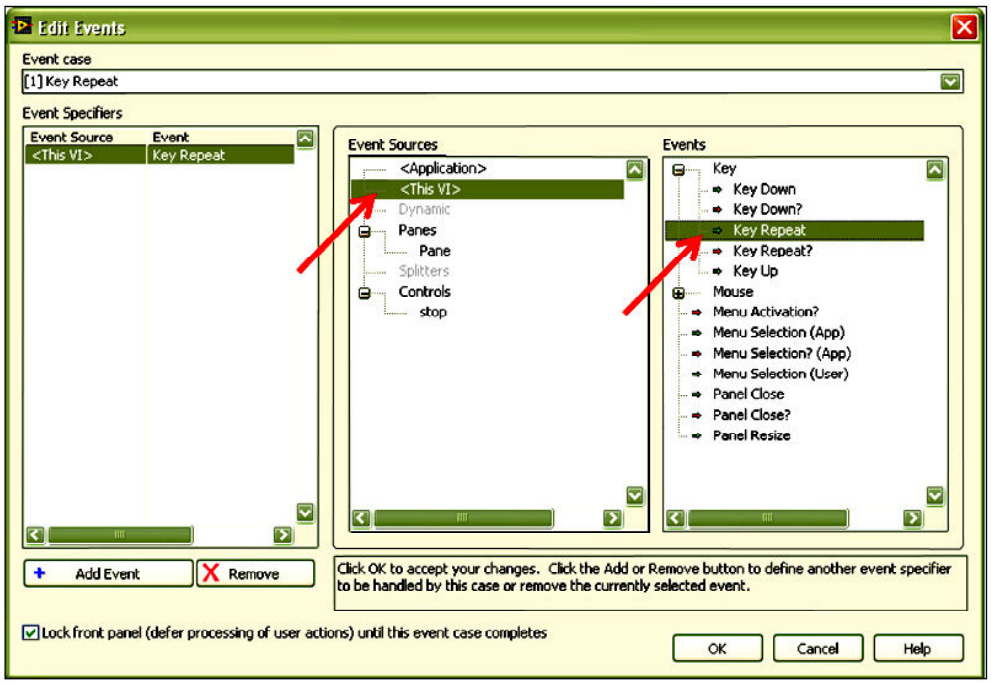


Рис. 15

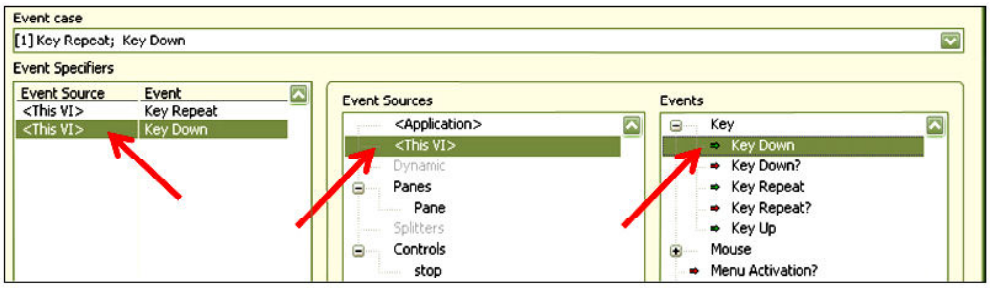


Рис. 16

→ **Key Down** (событие по нажатию клавиши клавиатуры) (рис. 16).

Нажимаем «ОК», и в нашей структуре событий добавилась ветвь, выполняемая когда мы либо нажимаем кнопку на клавиатуре, либо продолжаем держать уже нажатую клавишу.

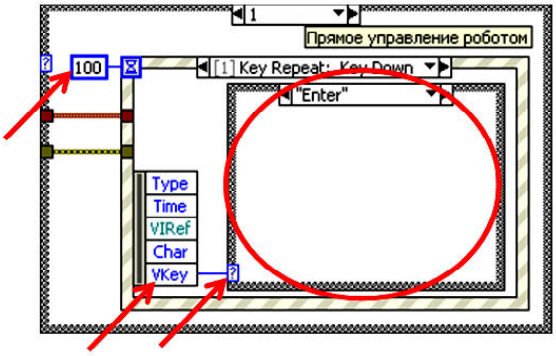


Рис. 17

Внутри этой ветви добавляем структуру **Case**, задачей которой будет определять, какую клавишу мы нажали. На вход терминала селектора заводим параметр события **Vkey**. Нажимаем правой кнопкой мышки на рамке структуры **Case** и выбираем пункт **Add case for every value** (рис. 17).

Теперь, открыв список ветвлений, получим перечень ветвей для всех типов клавиш клавиатуры. Нас интересуют ветки **Up**, **Down**, **Left** и **Right**, эти ветви соответствуют событиям при нажатии клавиш стрелок.

При помощи функции **NXT robotics** → **NXT I/O** → **NXT Direct I/O** → **Output** →

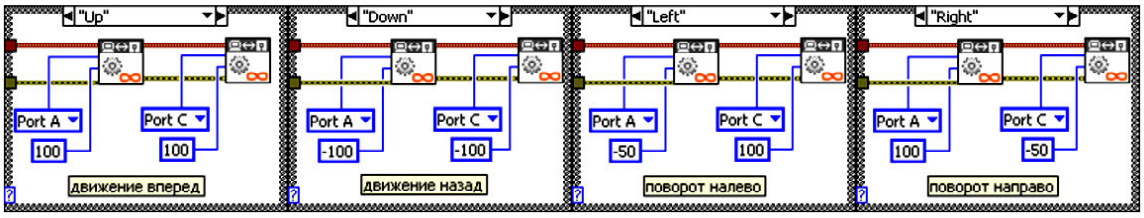


Рис. 18

Motor unlimited (прямое управление мотором) для вышеуказанных ветвей создаем следующие действия (рис. 18).

Нам нужно добавить еще одно ветвление в структуру событий. Снова кликаем на полосатую рамку правой кнопкой мышки и выбираем **Add event case**. Далее выбираем **<This VI> → Key Up** (событие по отпусканию клавиши клавиатуры).

По этому событию, независимо от типа отпущенной кнопки, мы будем выдавать команды на остановку моторов **NXT robotics → NXT I/O → NXT Direct I/O → Output → Motor stop** (рис. 19).

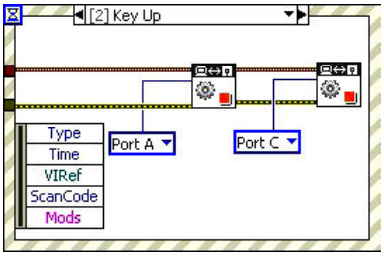
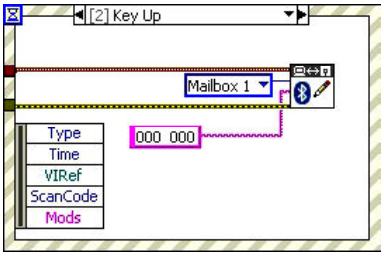


Рис. 19

УПРАВЛЕНИЕ РОБОТОМ ПРИ ПОМОЩИ ОТПРАВКИ КОМАНД В ИСПОЛНЯЕМУЮ ПРОГРАММУ НА РОБОТЕ

Для реализации данной программы нужна будет функция **NXT robotics → NXT I/O → NXT Direct I/O → BT Messages → Write message** .



Данная программа будет выглядеть практически так же, как предыдущая, только при нажатии и удерживании клавиш клавиатуры ветви для кнопок **Up**, **Down**, **Left** и **Right** а также действия при отпускании клавиш будут иметь следующий вид (рис. 20).

Следует обратить внимание на то, что в передаваемых строках одно число от другого отделяют два пробела.

За основу программы для робота возьмем программу «Приемник» из первого примера. С изменениями она будет выглядеть так (рис. 21).

Здесь у функций **String to number** задействован вход **offset** (у верхней функции значение 0, у нижней – 5), который указывает на номер символа исходной строки, начиная с которого функция будет искать число. Для

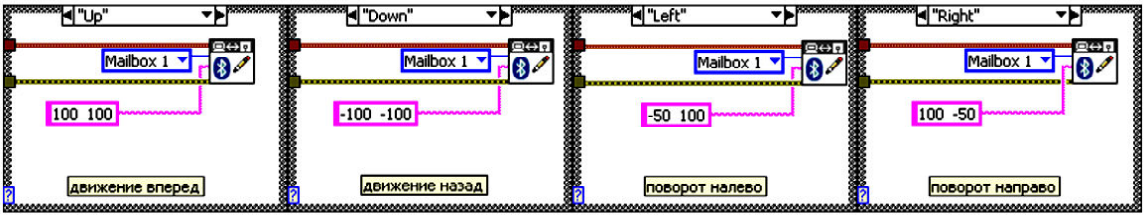


Рис. 20

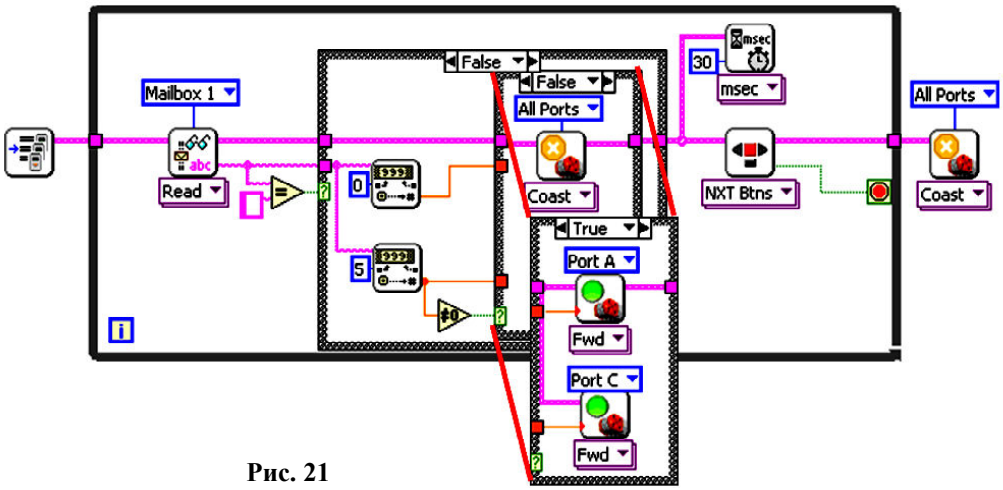


Рис. 21

этого важно было в посылках вставлять по два пробела между значениями.

ПЕРЕВОД РОБОТА В АВТОМАТИЧЕСКИЙ РЕЖИМ СЛЕДОВАНИЯ ПО ЛИНИИ

Для решения этой подзадачи мы воспользуемся уже написанной ранее программой движения по линии. Назовем ее **Line_follower** и запишем ее на контроллер NXT.

В программе «Пульт управления» выбираем ветвь с номером «3» в «ветвлении состояний» и добавляем в нее следующие функции:

NXT robotics → NXT I/O → NXT Direct I/O → Program execution → Stop program

 – остановка программы, которая в данный момент выполняется на NXT.

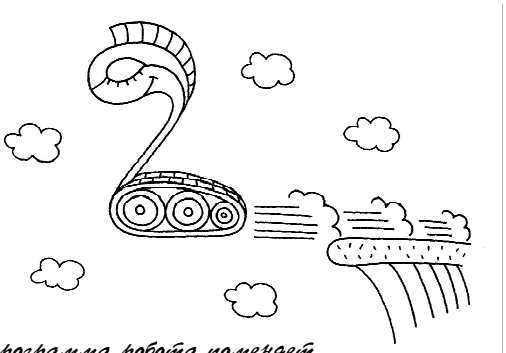
NXT robotics → NXT I/O → NXT Direct I/O → Program execution → Start program

 – запуск программы на контроллере NXT. Необходимо указать имя запускаемой программы, в нашем случае, это программа «Line_follower» (рис. 22).

Рис. 22

Эту задачу также можно решить через отправку специального сообщения на NXT, по которому программа робота поменяет свое состояние на «движение по линии», и он будет ехать по ней, пока не получит другого сообщения от компьютера. Все инструменты для реализации подобного решения уже были рассмотрены ранее, поэтому для практики предлагается написать подобную подпрограмму самостоятельно.

В завершение программы «Пульт управления» осталось добавить переключение между состояниями и корректное завершение работы программы. Для этого числовую константу в разделе инициализации заменим на элемент управления, кликаем по рамке константы правой кнопкой мыши и выбира-



...программа робота поменяет свое состояние на «движение по линии», и он будет ехать по ней, пока не получит другого сообщения от компьютера.

ем **Change to control**. Меняем имя у появившегося терминала с **Numeric** на **State**. Снова кликаем по нему правой кнопкой мыши и в меню выбираем **Create → Local variable**, у нас в «руке» появился синий прямоугольник с черным домиком и надписью **State** – это дополнительный терминал для чтения или записи значений в элемент индикации **State** 

. В **Labview** любой элемент управления или индикации является переменной, а терминал **Local variable** – один из способов получить доступ к значению этой переменной. Устанавливаем терминал переменной **State** после ветвления состояний, затем кликаем на нем правой кнопкой мыши и меняем его тип с «записи» на «чтение» – **Change to read**. Независимо от типа элемента на лицевой панели, к которому привязан терминал локальной переменной, в программе мы можем как читать, так и записывать значения в этот элемент управления.

После главного цикла добавляем функцию удаления сессии связи с NXT: **NXT robotics → NXT I/O → NXT Direct I/O → Connection → Destroy NXT Object** 
(рис. 23).

Программа «Пульт управления» готова! Как обычно, не забываем сохранить ее и, конечно, проверить работу. Будьте внимательны: в третьем состоянии (запуск движения по линии) умышленно упущен один момент, который не позволит роботу адекватно ехать по линии. Нахождение недочета, а также его устранение, остаются вам как домашнее задание.

ЗАКЛЮЧЕНИЕ

Хочется отметить, что набор приведенных в статье примеров носит общий, собирательный характер. Основной задачей было показать как можно большее количество инструментов и методов, которые помогут вам в решении множества задач по робототехнике. Остается только пожелать успехов в освоении робототехники!

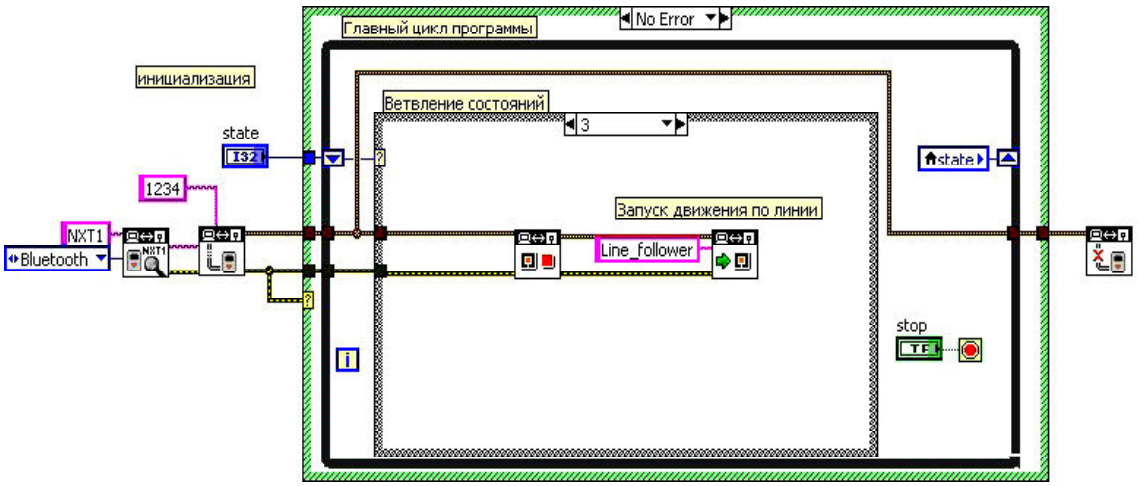


Рис. 23

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