Многопоточное программирование EV3

Автор: Трофимов Александр Артемович

МБУДО «ГЦДТТ им. В.П. Чкалова» г. Казани

2019 год

Синхронная программная модель

- Это программная модель, когда потоку назначается одна задача и начинается выполнение. Когда завершено выполнение задачи тогда появляется возможность заняться другой задачей. В этой модели невозможно останавливать выполнение задачи чтобы в промежутке выполнить другую задачу.

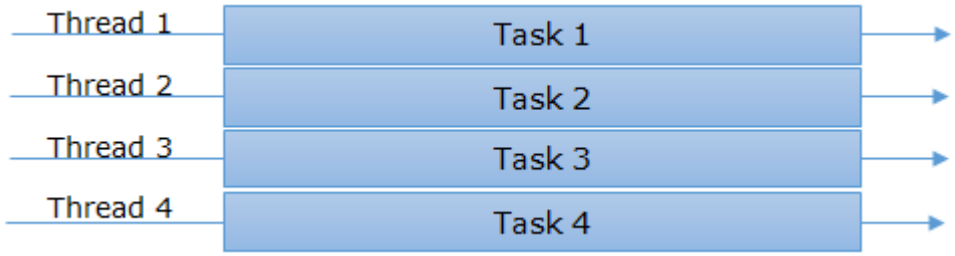
Однопоточность

- Если мы имеем несколько задач, которые надлежит выполнить, и текущая система предоставляет один поток, который может работать со всеми задачами, то он берет задачи поочередно одну за другой и процесс выглядит так:

- Здесь мы видим, что мы имеем Поток 1 и 4 задачи, которые необходимо выполнить. Поток начинает выполнять поочередно одну задачу за другой и выполняет их все.

Многопоточность

- В этом сценарии, мы использовали много потоков, которые могут брать задачи и приступать к работе с ними. Итак, программа может работать вот так:



- Здесь мы можем видеть, что у нас есть 4 потока и столько же задач для выполнения, и каждый поток начинает работать с ними. Это идеальный сценарий, но в обычных условиях мы используем большее количество задач чем количество доступных потоков, таким образом освободившийся поток получает другое задание. Как уже говорилось, создание нового потока не происходит каждый раз, потому что для этого требуются системные ресурсы, такие как процессор, память, а начальное количество потоков должно быть определенным.

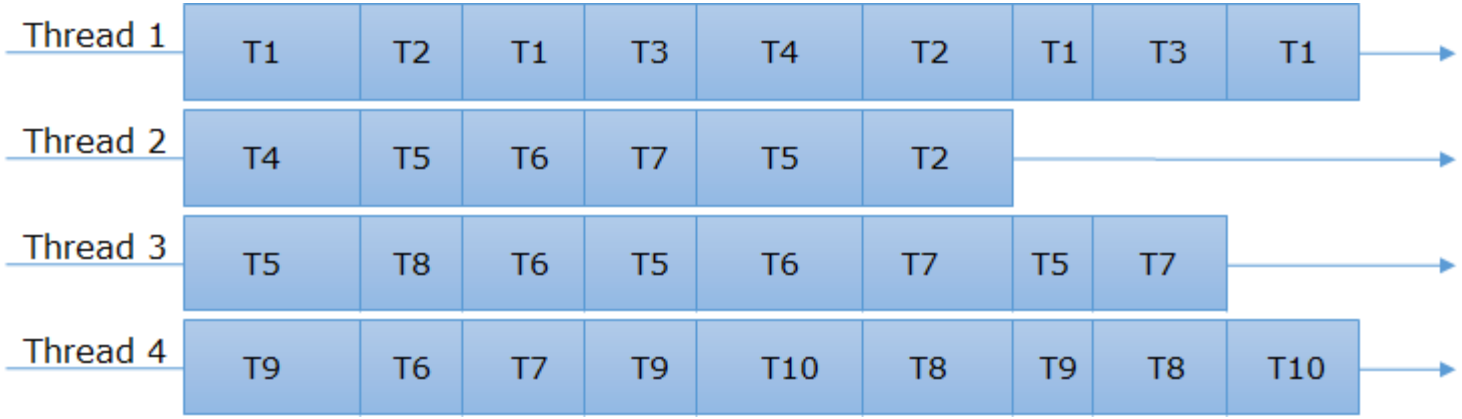
Асинхронная модель программ

- В отличии от синхронной программной модели, здесь поток однажды начав выполнение задачи может приостановить выполнение сохранив текущее состояние и между тем начать выполнение другой задачи.

- Здесь мы можем видеть, что один поток отвечает за выполнение всех задач и задачи чередуются друг за другом, но в зависимости от приоритета и прихода других задач, предыдущая задача останавливается и продолжает работу, когда поток освободиться.

Асинхронная многопоточная модель

- Если наша система способна иметь много потоков, тогда все потоки могут работать в асинхронной модели как показано ниже:



- Здесь мы можем видеть, что одна и таже задача скажем T4, T5, T6 обрабатывается несколькими потоками. Это красота данного сценария – использование потоков по максимуму.

4 модели программ

- Синхронная однопоточная
- Синхронная многопоточная
- Асинхронная однопоточная
- Асинхронная многопоточная

- Какую же модель программ используют роботы EV3?

Принцип работы робота EV3

- Робот EV3 – это полноценный компьютер. В нем есть всё, что есть в обычном ПК: центральный процессор, память оперативная и постоянная, устройства ввода и вывода, а главное, в нем есть операционная система.
- Операционная система основана на многопоточном выполнении десятков приложений. Отсюда мы можем сказать, что робот работает в многопоточном режиме, однако наша программа занимает всегда только один поток.
- Хоть мы и можем запрограммировать робота таким образом, чтобы он выполнял блоки параллельно, однако, это просто пример асинхронного программирования.

Многопоточности в EV3 нет

- Спасибо за внимание!

Хотя… Еще немного

- Хоть настоящей многопоточности в EV3 нет, однако для обычного пользователя та же асинхронная модель выглядит точно так же.
- Рассмотрим несколько примеров программ…

Однопоточная программа для управления двигателем

- В ЛЕГО постарались сделать так, чтобы детям было легко управлять роботом, ведь движение робота – первое что привлекает детей.
- В следующем примере стоит блок начала программы и блок рулевого управления двигателями с параметрами:
  - «Поворачивай немного вправо»
  - «Со скоростью 50%»
  - «Как сделаешь 3 оборота двигателями»
  - «Остановись»

Разбитие блока в явную асинхронную модель

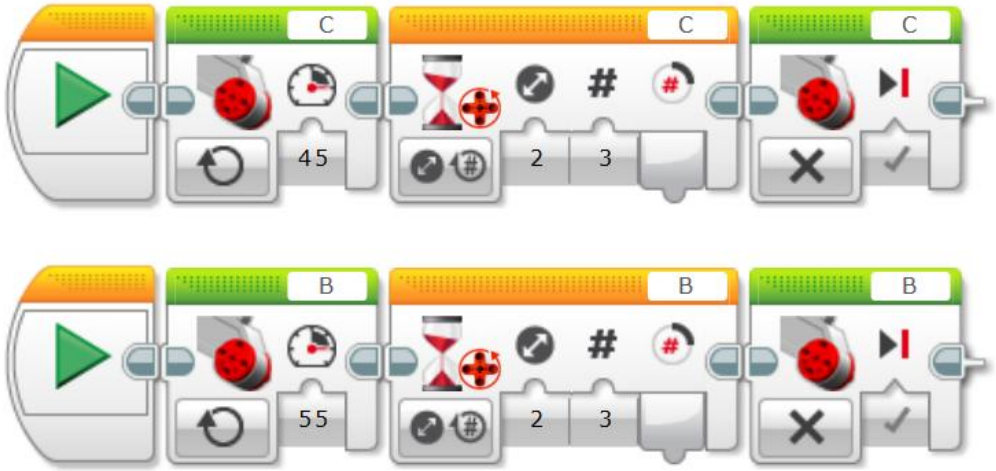
- Красным обведена программа, ничем не отличающаяся от предыдущей, однако каждое действие стоит в отдельном блоке:
  - Запусти двигатель 1
  - Запусти двигатель 2
  - Жди 3 оборота
  - Останови двигатель 1
  - Останови двигатель 2

- Именно так и выглядит со стороны робота этот блок, однако он стал в разы сложнее для пользователя.

Мнимая многопоточность

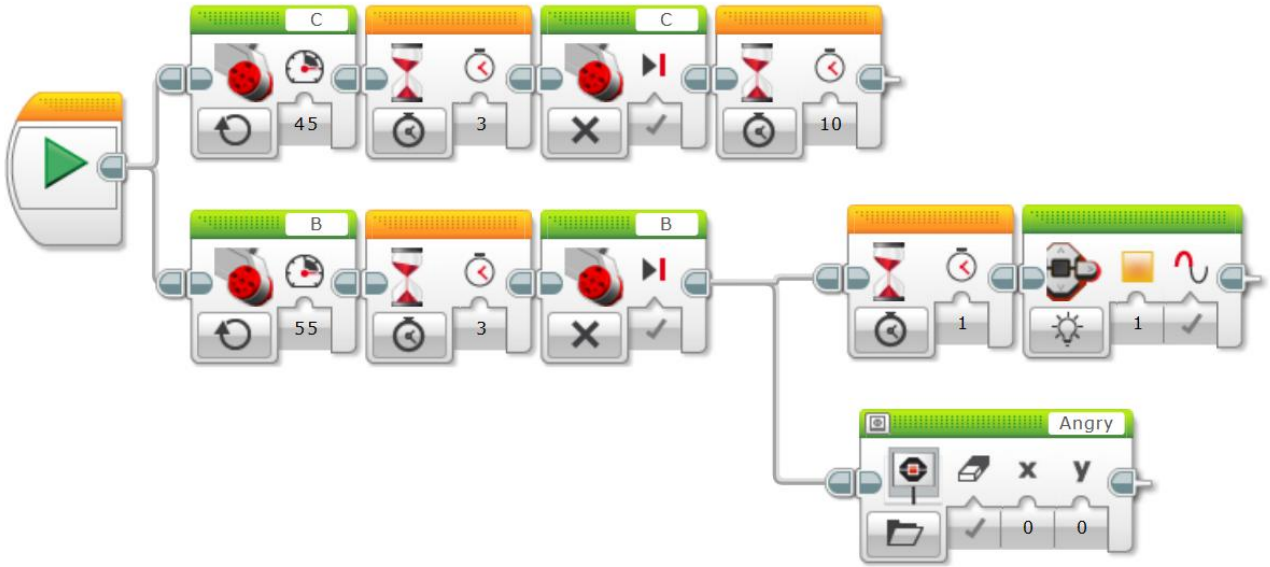
- Мы можем в одну программу поставить два начальных блока. Внешне будет казаться, что эти два «начала» стартуют одновременно, а нам и внешнего сходства с многопоточностью хватит.

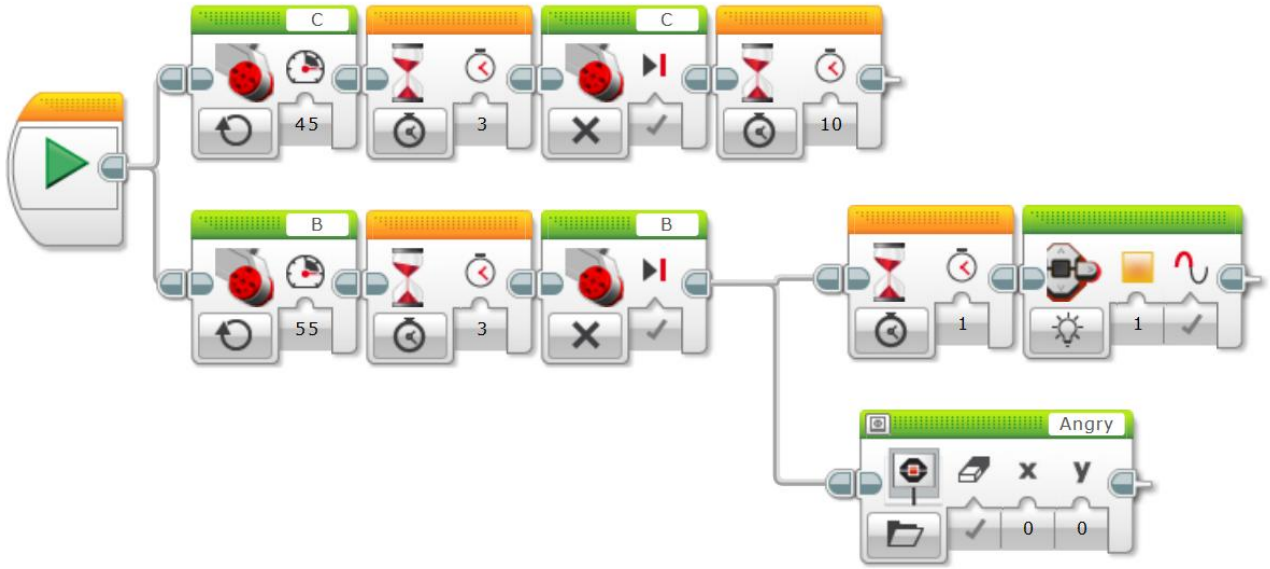
В программе допущена ошибка, так как мотор C будет ехать медленнее, но на то же расстояние – в конце программы робот начнет резко разворачиваться налево. Чтобы этого избежать, внутри блока рулевого управления стоят корректировки длительности работы.



Второй способ создать дополнительный поток

- Дополнительный «мнимый» поток можно создать просто подтянув язычок к нужному блоку. На следующем изображении можно увидеть предыдущую программу с некоторыми дополнениями:
  - Блоки ожидания оборотов заменены на блоки ожидания времени для исправления ошибки.
  - Добавлены дополнительные блоки.

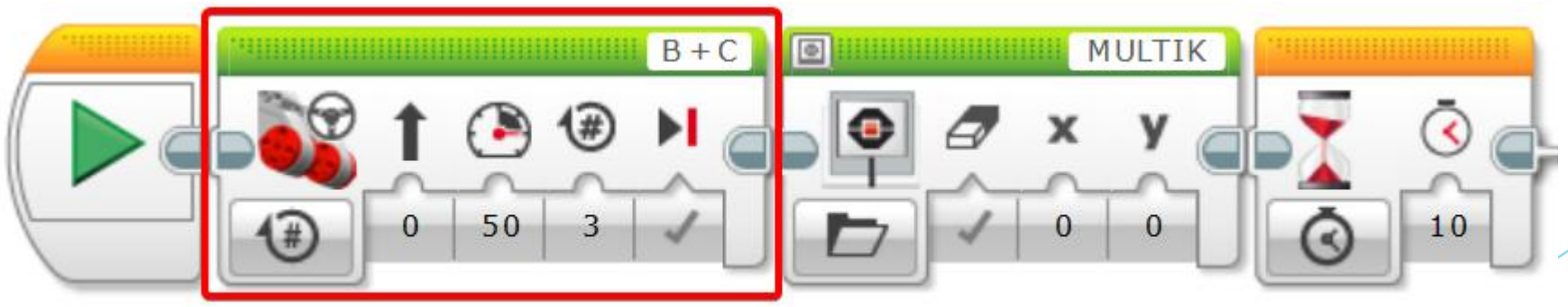




- Начни:
  - После поворота мотором C 3 секунды – жди 10 секунд
  - После поворота мотором B 3 секунды:
    - Жди секунду и включи желтую индикацию
    - Включи рожицу «злой»
- При этом после поворота сначала покажется рожица, затем с задержкой в секунду сменится индикация на желтую, а через еще 9 секунд робот завершит работу программы.

Блокирующие и неблокирующие блоки

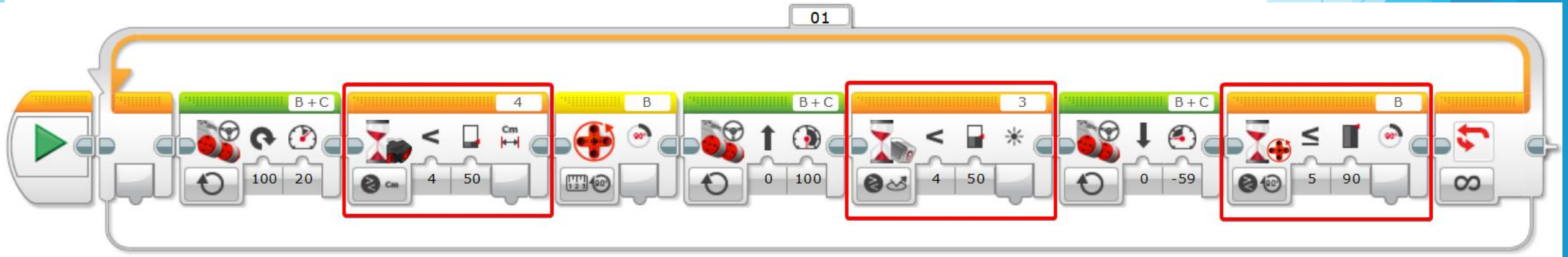
- В EV3 есть два вида блоков:
  - Блоки, которые выполняются мгновенно
  - Блоки, которые выполняются некоторое время, либо до определенного события
- Например, блок двигателя. Если вы насильно остановите робота и не дадите его колесам крутиться – он зависнет на выделенном блоке, так и не показав нам мультик (MULTIK). При этом сам блок «Экран» здесь – неблокирующий и по этому, если не поставить ожидание в конце, то программа сразу завершится и мы максимум, что успеем увидеть – это моргнувший экран.



Ходила байка, что если на «Электронике» с «Ну, погоди!» набрать 1000 очков – покажут мультик. Здесь это реально – просто отпусти робота.

Разбор «кегельринга»

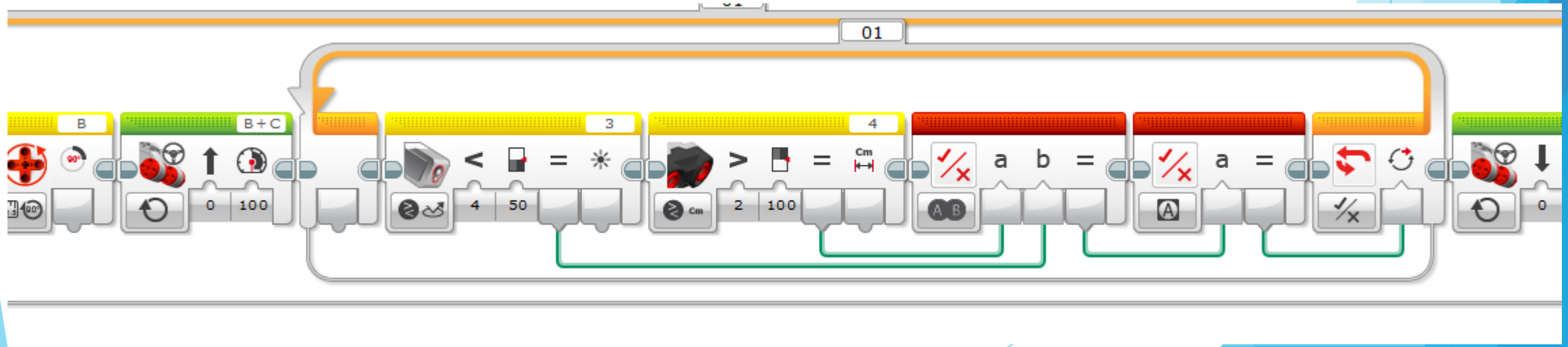
- В кегельринге нам требуется выбить кегли за пределы ринга. Для этого есть несколько тактик, как с датчиками, так и без. Мы рассмотрим первый вариант – выбивание кегель каждый раз из центра:
  - Мы вращаемся, пока не увидим кеглю датчиком расстояния
  - Мы сбрасываем датчик оборотов и едем вперед, пока не увидим черную линию датчиком цвета
  - Мы едем назад, пока не увидим ноль или меньше на датчике оборотов
  - GOTO 10
- Блоки отмеченные красным – блокируют выполнение программы до определенного события. Однако в данной программе нам это не мешает.



Разбор «сумо»

- Сумо от кегельринга отличается двумя вещами:
  - Конструкция робота
  - Противник не стоит на месте

- Минус предыдущей программы в том, что противник может уехать за то время, пока мы едем на него. Чтобы этого не произошло блок ожидания черной линии требуется заменить на цикл, который будет выполняться, либо пока не увидим черную линию, либо пока не увидим, что противник пропал из поля зрения. Цикл получается достаточно сложный.

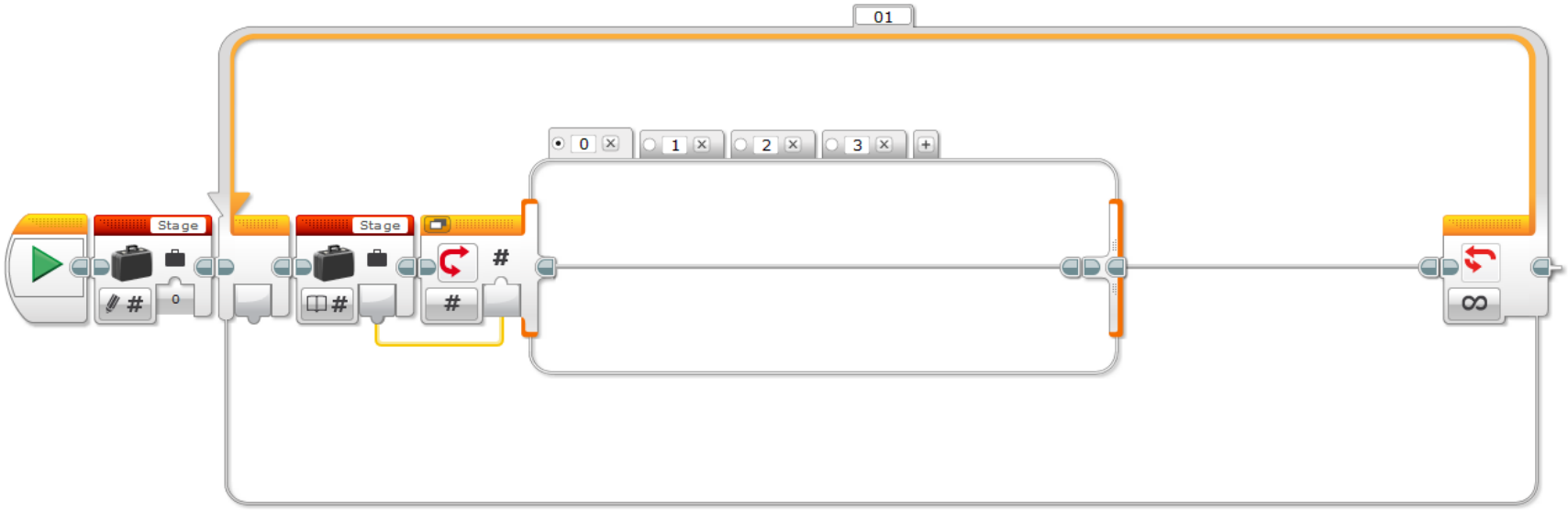


Стадийная модель работы робота

- В идеале, в сумо используются 2 пары датчиков (спереди и сзади). Однако «классическая» программа для такого робота будет настолько запутанной, что мне лень ее писать для этой презентации. Давайте лучше познакомимся с тем, что я назвал «стадийная модель работы».
- Она позволяет опрашивать столько датчиков одновременно, сколько вам требуется, при этом не загромождая экран кучей связанных непонятных блоков. Также она позволяет достаточно просто и четко выстроить алгоритм работы робота используя ветвления зависящие от датчиков.

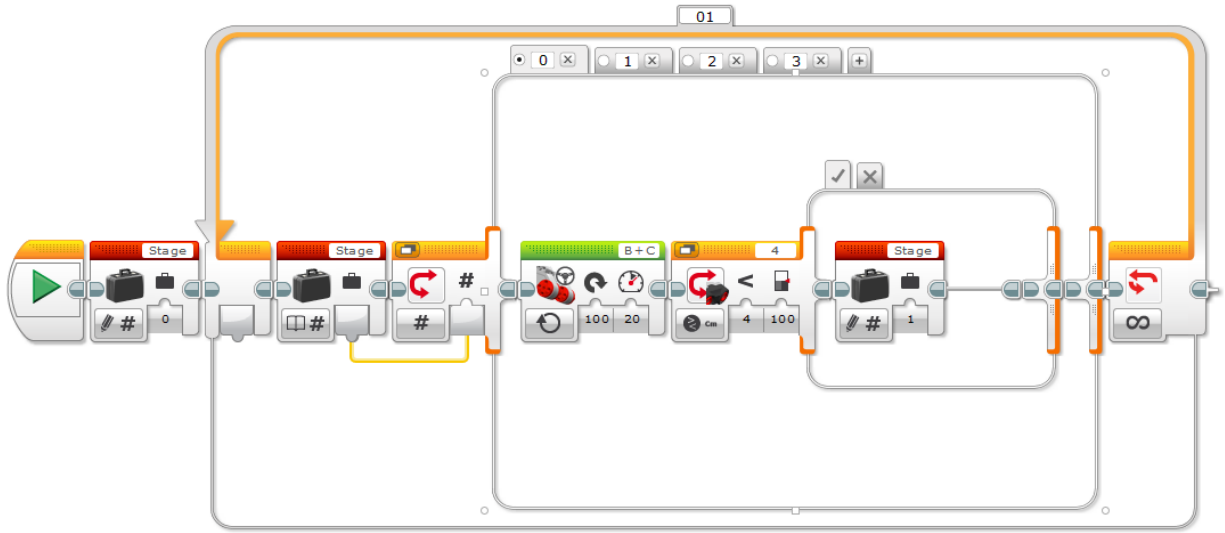
Основа

- Основа у этой модели всегда одинаковая:
  - У нас есть переменная, которая отвечает за выбор стадии
  - Бесконечный цикл с переключателем стадий
- При использовании этой модели есть правила:
  - Не используйте блокирующие блоки, если не уверены, что они вам нужны.
  - По одному выполняющемуся неблокирующему блоку управления двигателями в одной стадии.



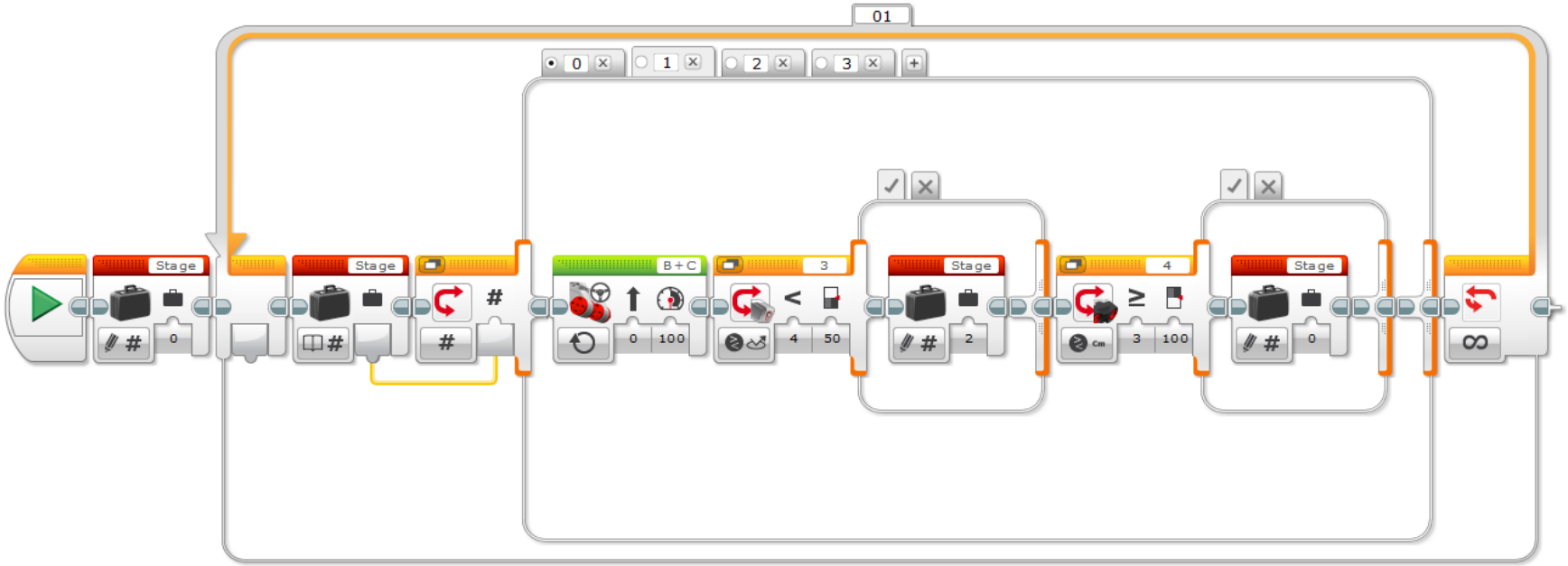
Разбор сумо в стадийной модели

- Стадия 0:
  - Вращайся
  - Если видишь препятствие до 100см – переключись на стадию 1.
  - Повтори
- Эти два блока не блокируют поток, но в данной стадии нам это особо не поможет, однако поможет в стадии 1.
- Вариант «ложь» в датчиках этой модели почти всегда пустой.



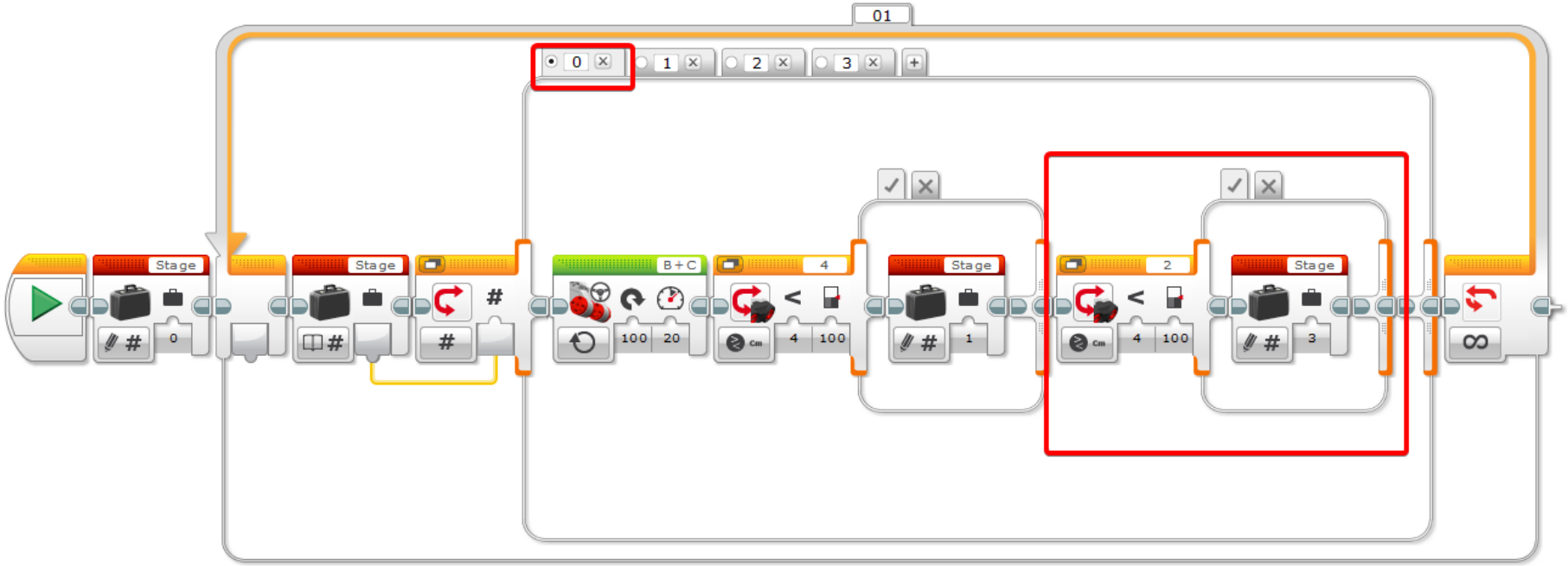
Разбор сумо в стадийной модели

- Стадия 1:
  - Едем вперед
  - Если доехал до черной линии – переключись на стадию 2, мы выбили врага.
  - Если потерял противника из виду – переключись на стадию 0, чтобы вновь начать его искать.
  - Повтори



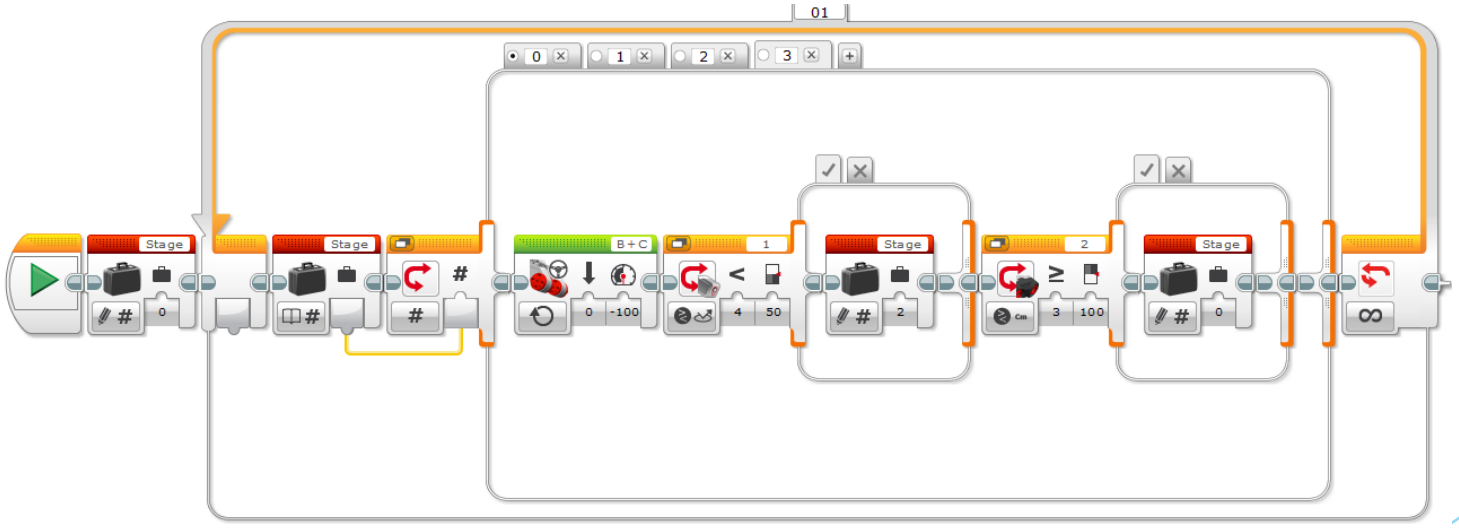
Разбор сумо в стадийной модели

- Помните, я говорил, что эта модель куда проще для понимания? Так вот, добавление второй пары датчиков осуществляется также, как и добавление первой…
- Стадия 0 (дополнение):
  - Если видишь препятствие вторым датчиком – переключись на стадию 3.



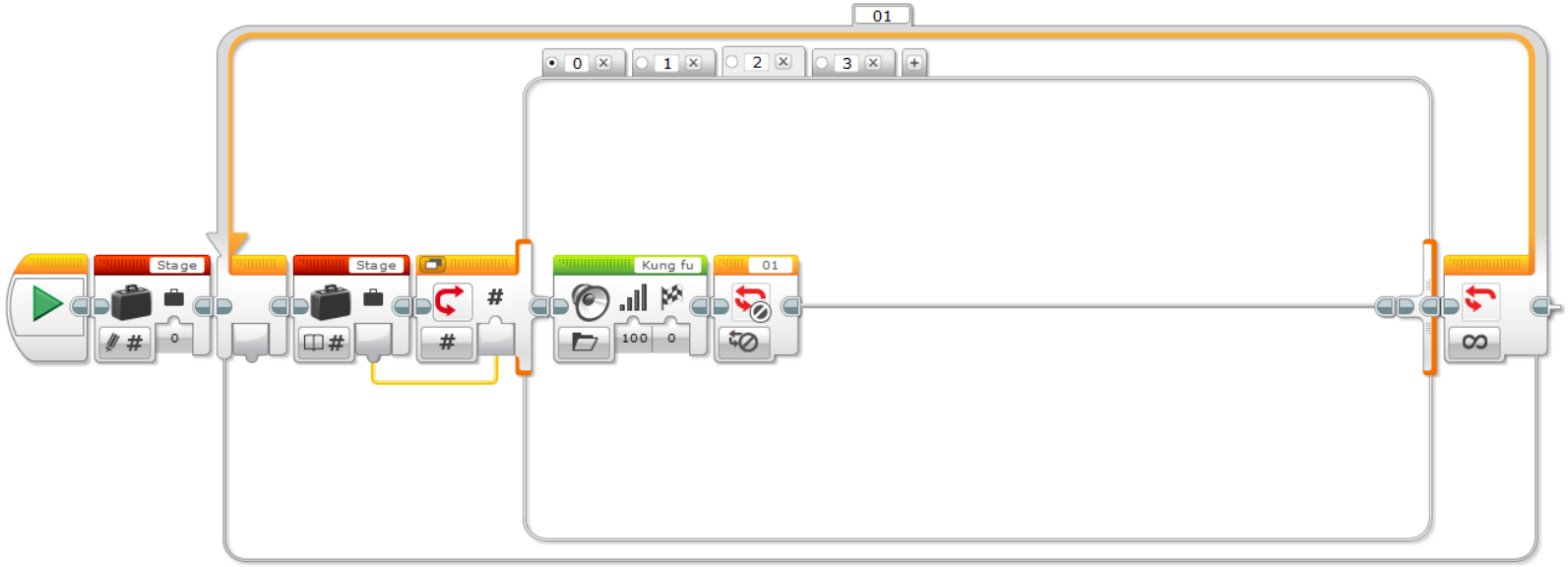
Разбор сумо в стадийной модели

- Стадия 3:
  - Едем назад
  - Если доехал до черной линии – переключись на стадию 2, мы выбили врага.
  - Если потерял противника из виду – переключись на стадию 0, чтобы вновь начать его искать.
  - Повтори
- Отличия от стадии 1:
  - Едем назад, так как сработал датчик расстояния, стоящий сзади.
  - Используем задние датчики (на других портах).



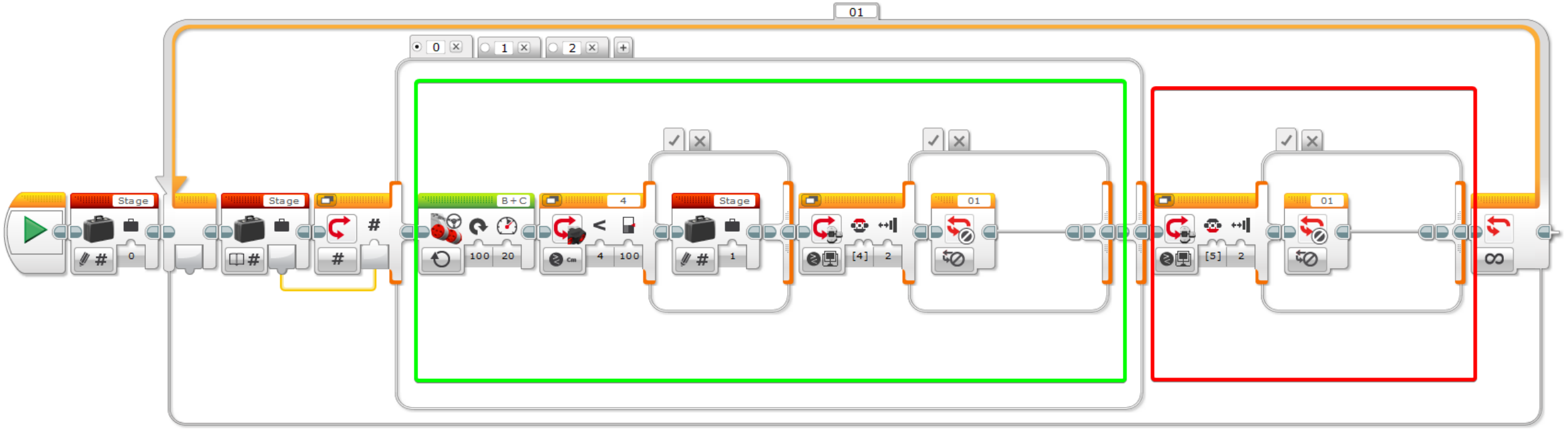
Разбор сумо в стадийной модели

- Ну, а в стадии 2, мы можем сделать все что угодно, например бесконечно играть мелодию или просто прервать цикл, завершив тем самым программу.



Стадийная модель - области

- Стадийная модель имеет две области проверки датчиков:
  - Локальная, которая находится в каждой отдельной стадии – зеленая
  - Глобальная, которая находится за переключателем стадий – красная
- Например, так программа будет выключаться всегда при нажатии кнопки вниз, и будет выключаться только во время стадии 0 при нажатии кнопки вверх:



Стадийная модель - итоги

- Позволяет легко и просто одновременно опрашивать нужное количество датчиков в нужный момент времени.
- Позволяет построить легко отслеживаемый алгоритм работы.
- Имеет глобальную область проверки датчиков.
- Имеет возможность отделить исполнительную часть от проверочной.

- Чуть сложнее в освоении, нежели «классическая» модель.

Спасибо за внимание!

- Автор:
  - Трофимов Александр Артемович
  - +79047606909
  - alex.trofimov@chkalovc.ru

- Использовались материалы с сайта Хабр:
  - https://habr.com/ru/post/337528/