

Алгоритмические структуры и элементарные действия



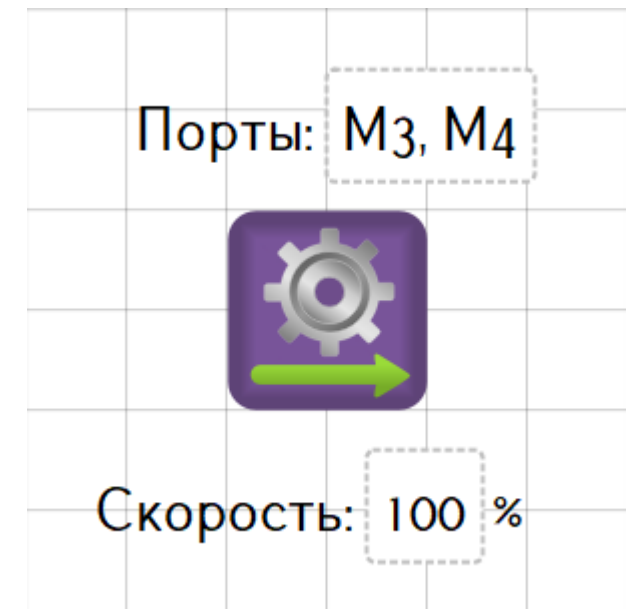
- Ознакомление с алгоритмами для элементарных действий мобильного робота и с базовыми алгоритмическими структурами

Движение вперед базовой тележки задается подачей на левый и правый мотор одинаковой скорости

В TRIK Studio для подачи мощности на мотор существует отдельный блок «**Моторы вперед**»

У этого блока два свойства:

1. Порты
2. Скорость



Движение вперед

TRIK

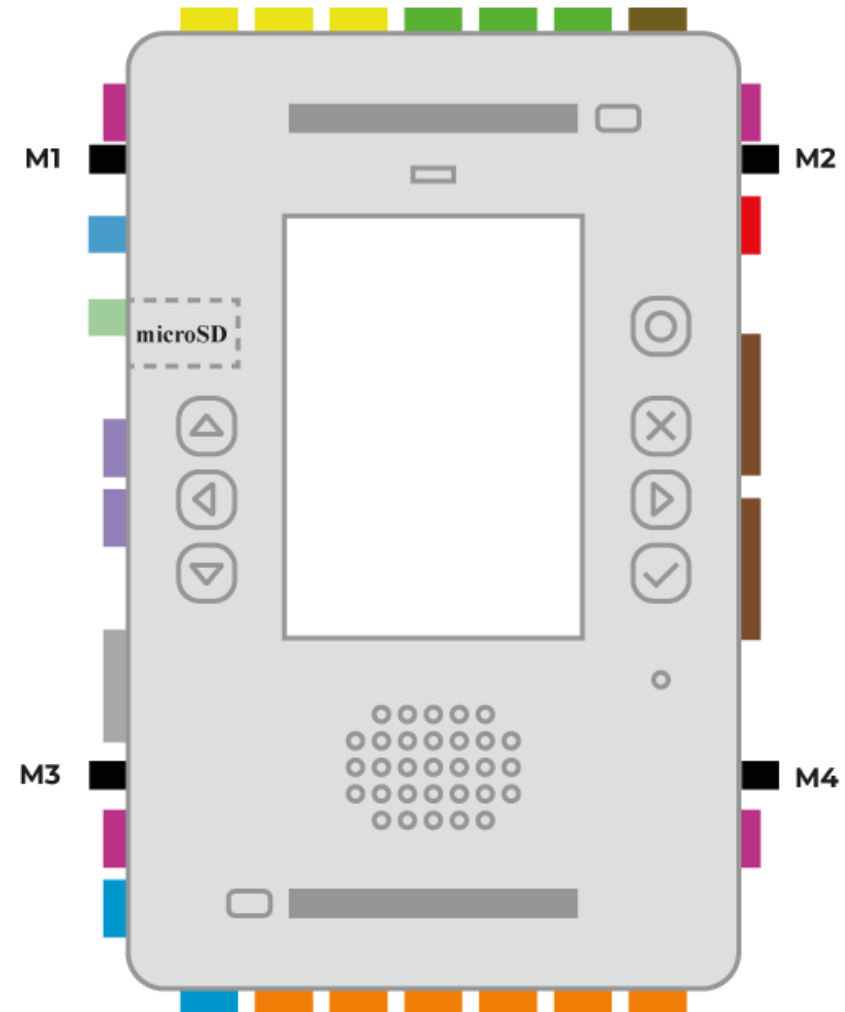


Подключение моторов

TRIK

У контроллера ТРИК
четыре порта для
подключения силовых
моторов:

M1, M2, M3 и M4



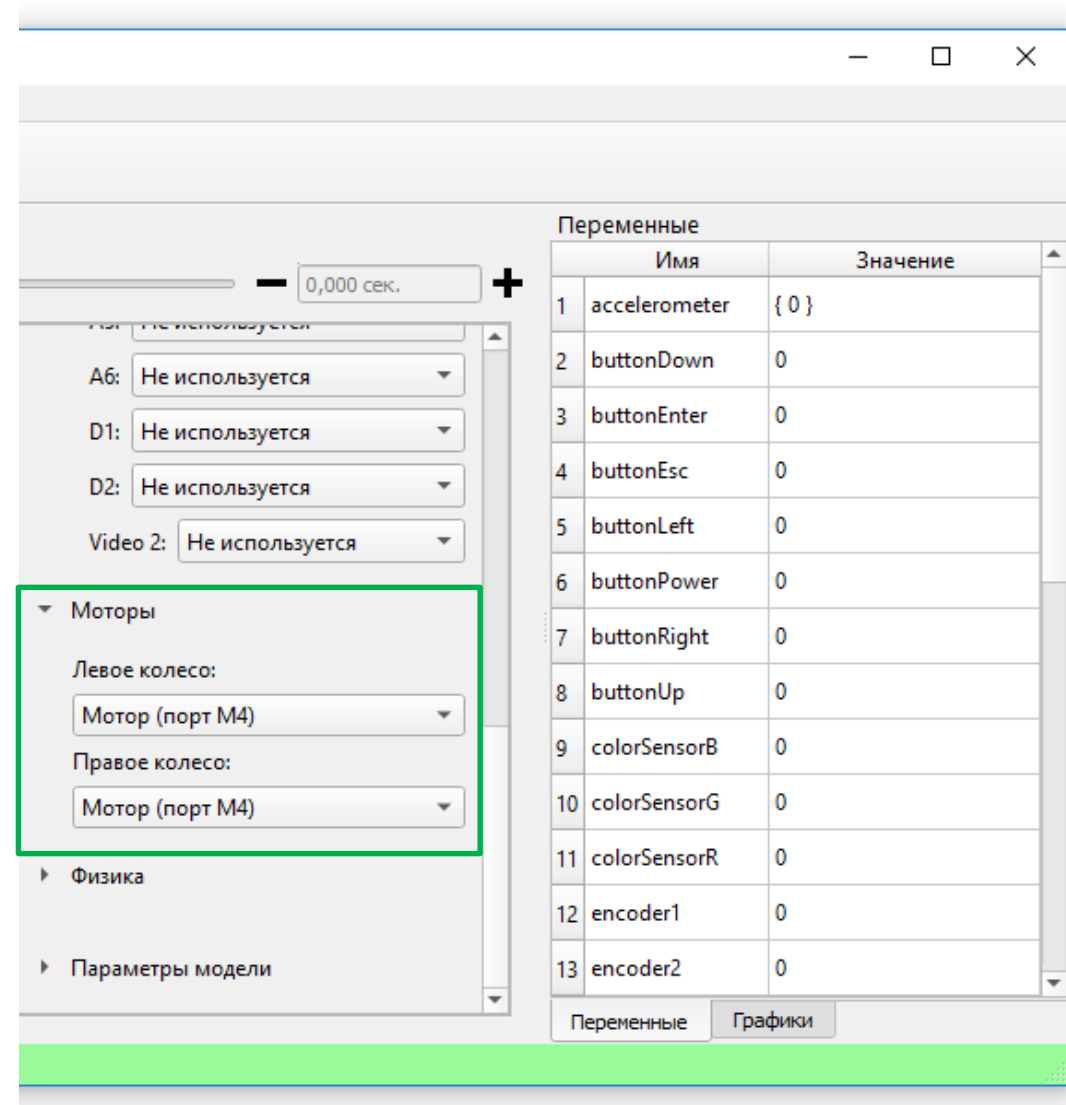
Подключение моторов



Подключение моторов в 2D-модели по умолчанию:

- левый — к порту **M3**
- правый — к порту **M4**

Настройку подключения моторов можно изменить в режиме отладки на центральной панели в разделе «Моторы»



Подключение моторов

TRIK

The screenshot shows the TRIK Studio 2019.8 interface. The main window displays a 2D model of a robot on a grid. The robot is positioned at the center of the grid. The interface includes a menu bar (Файл, Правка, Вид, Инструменты, Настройки, Справка), a toolbar with various icons, and a left sidebar with icons for Редактор and Отладка. The main workspace is titled "Двумерная модель" and contains a grid with a robot icon. A green box highlights the "Моторы" (Motors) section in the right-hand configuration panel, which shows two motor slots, both set to "Мотор (порт M4)".

TRIK Studio 2019.8 Несохраненный проект [изменён]

Файл Правка Вид Инструменты Настройки Справка

Двумерная модель

Сетка 0,000 сек.

Моторы

- Левое колесо: Мотор (порт M4)
- Правое колесо: Мотор (порт M4)

Физика

Параметры модели

Переменные

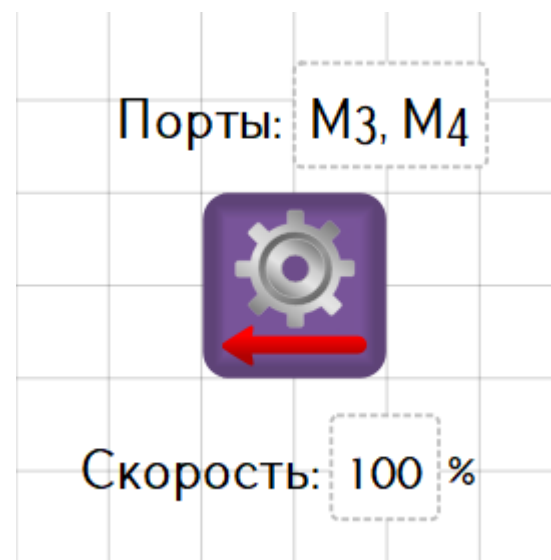
Имя	Значение
1 accelerometer	{ 0 }
2 buttonDown	0
3 buttonEnter	0
4 buttonEsc	0
5 buttonLeft	0
6 buttonPower	0
7 buttonRight	0
8 buttonUp	0
9 colorSensorB	0
10 colorSensorG	0
11 colorSensorR	0
12 encoder1	0
13 encoder2	0

Режим отладки - нажмите Ctrl+1 или кликните здесь для переключения в режим редактирования

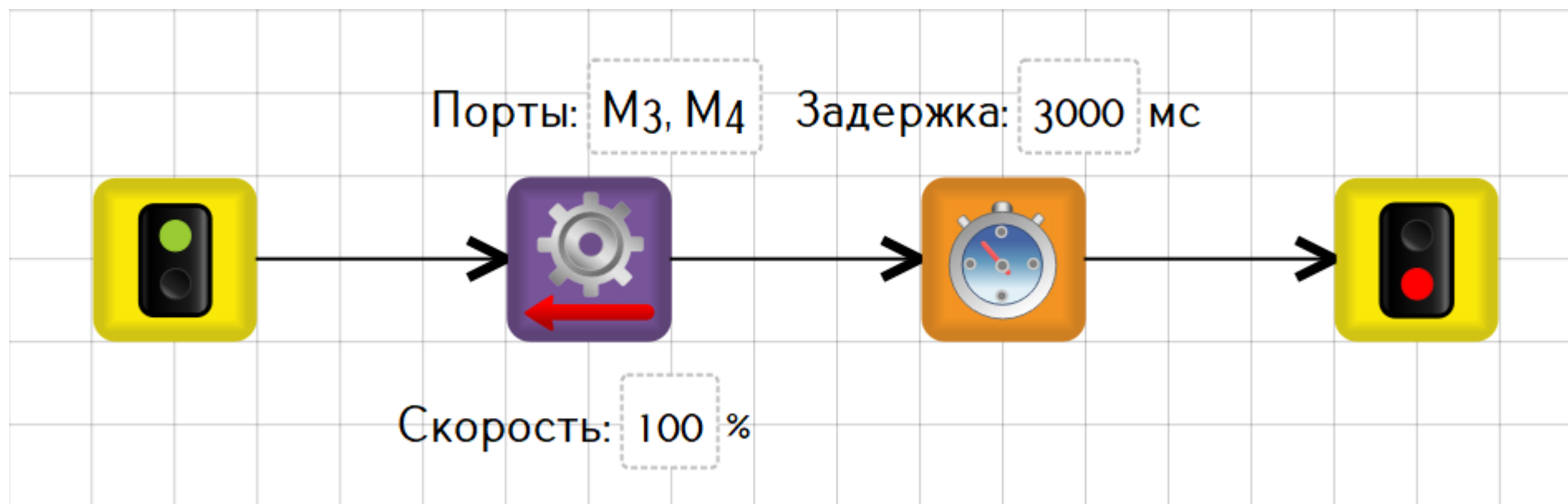


Движение назад выполняется аналогично

Используем блок
«**Моторы назад**»



Движение назад



Но! Диапазон подаваемой мощности: от -100 до 100 %



То есть для движения назад можно использовать и блок «Моторы вперед», подав мощность -100 %

Движение назад

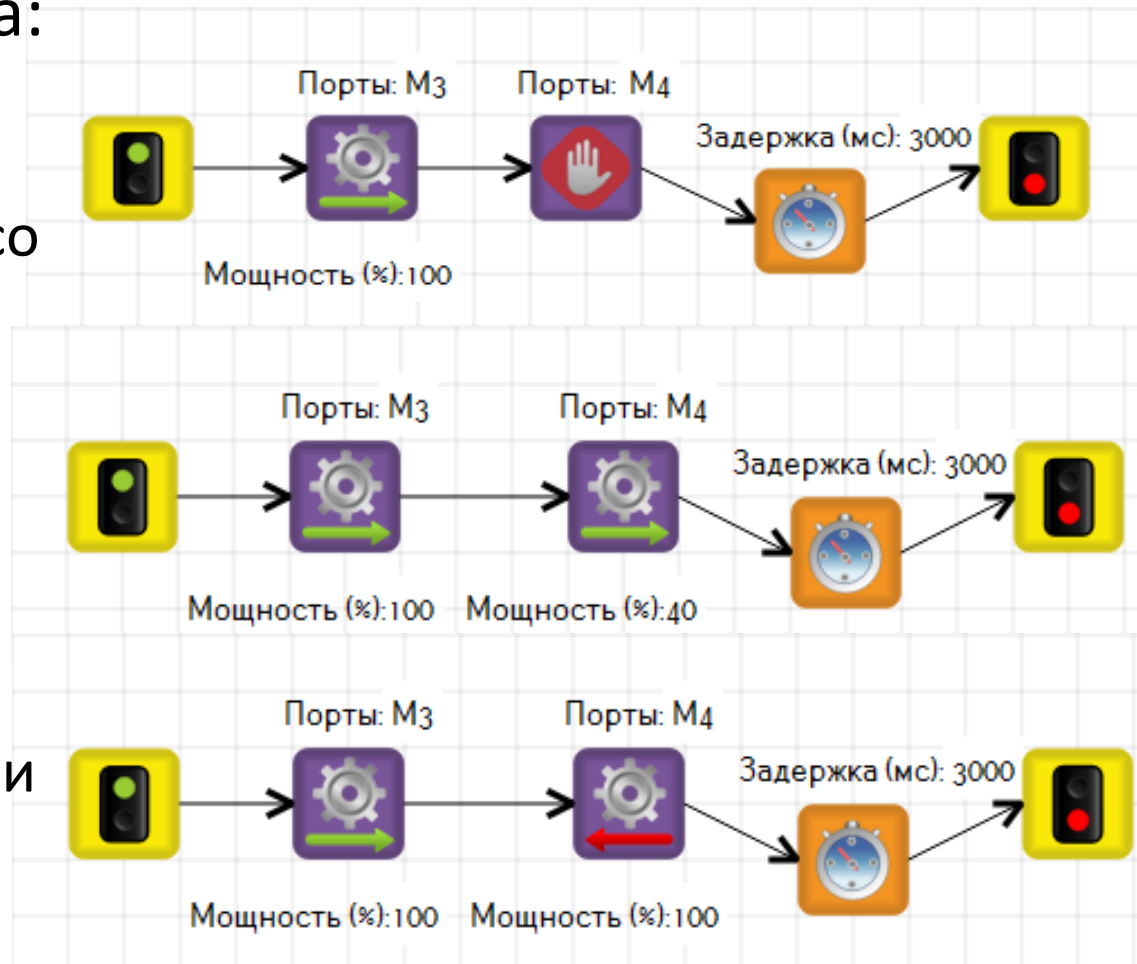
Но! Диапазон подаваемой мощности: от -100 до 100 %



То есть для движения назад можно использовать и блок «Моторы вперед», подав мощность -100 %

Повороты можно разделить на 3 типа:

- **резкий поворот**
мощность подается только на одно колесо
- **плавный поворот**
мощность подается на два колеса,
но на одно больше
- **поворот на месте**
одинаковая мощность с разными знаками
на два колеса



Модели алгоритмов

Представленные выше алгоритмы – **тайм-модели**. Движение осуществляется по таймеру. Это «плохой» подход, так как в этом случае выполняемое действие зависит от заряда аккумулятора

Правильно будет использовать **ожидание значения энкодеров**

В этом случае перед элементарным действием необходимо сбросить значения энкодеров



Остальные элементарные действия (движение назад, повороты) реализуются аналогично

Модели алгоритмов



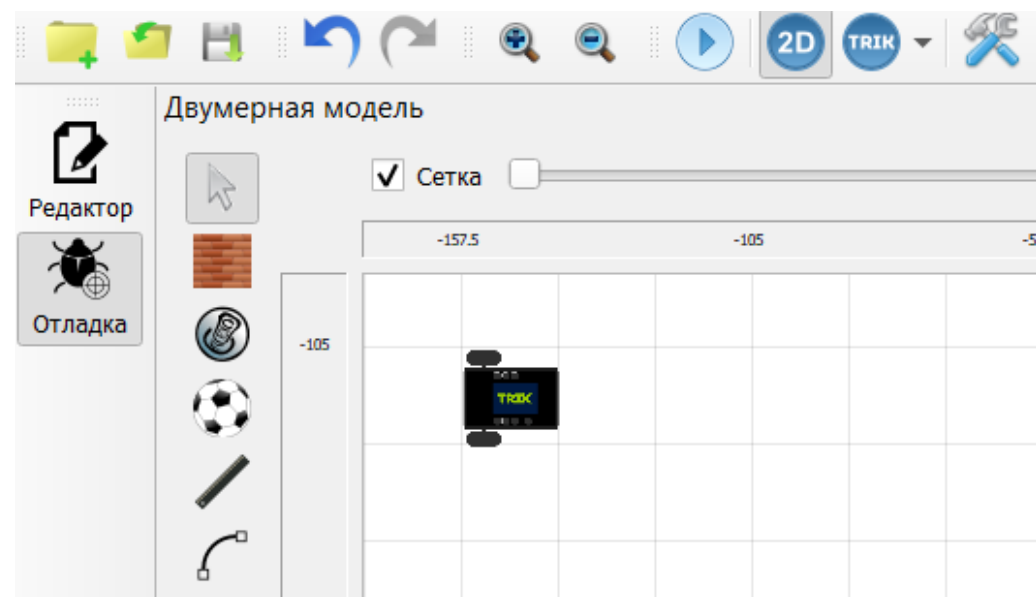
Ждать до того момента,
когда показание энкодера
станет **больше 360**

Точные перемещения

TRIK

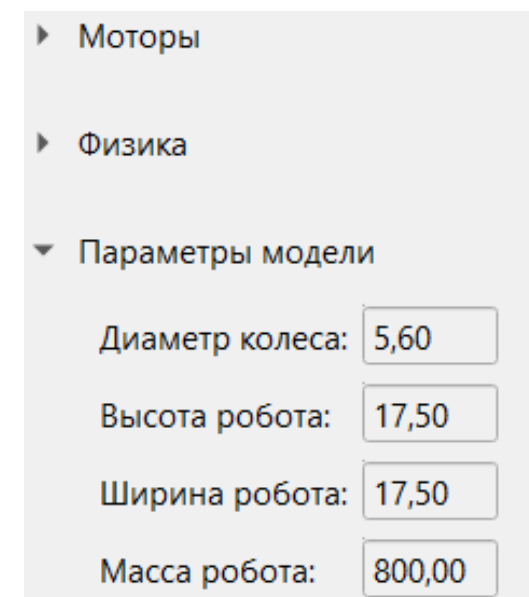
Поставьте галочку «Сетка». Теперь вы можете отслеживать точные перемещения модели

1 клетка = 17,5 см



Также, в режиме «отладка» всегда можно посмотреть параметры визуальной модели

Для удобства длина и размер базы робота совпадают с размером клетки (17,5 см)



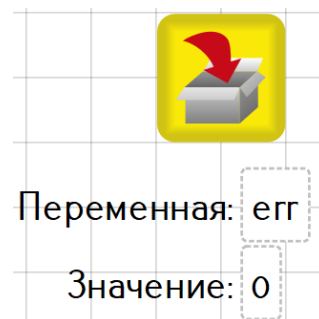
Переменные

Переменная — поименованная область памяти

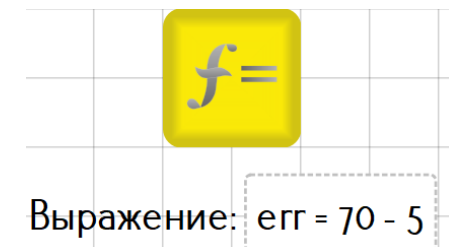


В TRIK Studio можно ввести свои переменные, используя блоки:

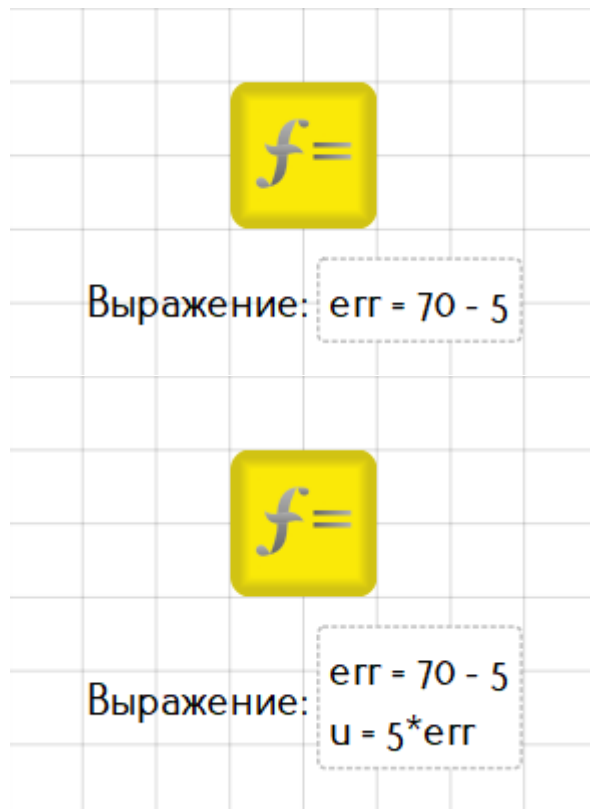
«Инициализация
переменной»



«Выражение»



В блоке «**Выражение**» можно как создавать новые переменные, так и записывать выражения

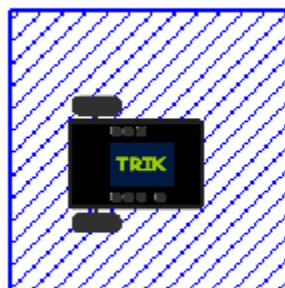


- Нецелые числа пишутся через точку.
Например: **1.75**
- Для перехода на новую строку используйте «**Shift**» + «**Enter**»
- Созданным переменным можно присваивать другие переменные, если последние были объявлены и инициализированы ранее.
Например: **u = 5*err**

Энкодеры. Задача



Задача 2.1.1 Робот находится в синей зоне старта. Робот должен проехать вперед, развернуться на 180° между зонами старта и финиша, проехать задом и остановиться в зеленой зоне финиша. Использовать энкодерную модель



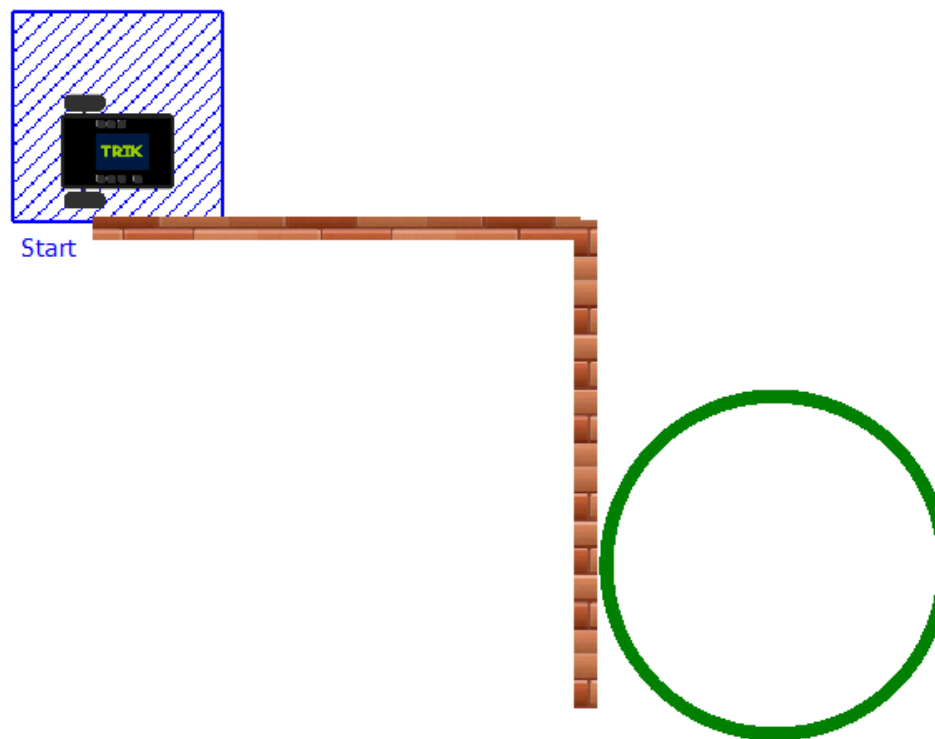
Start



Энкодеры. Задача

TRIK

Задача 2.1.2. Обогнуть угол. Робот должен проехать вперед со скоростью 60, повернуть на 90° , проехать вперед с максимальной скоростью и остановиться в зеленом круге. Использовать энкодерную модель



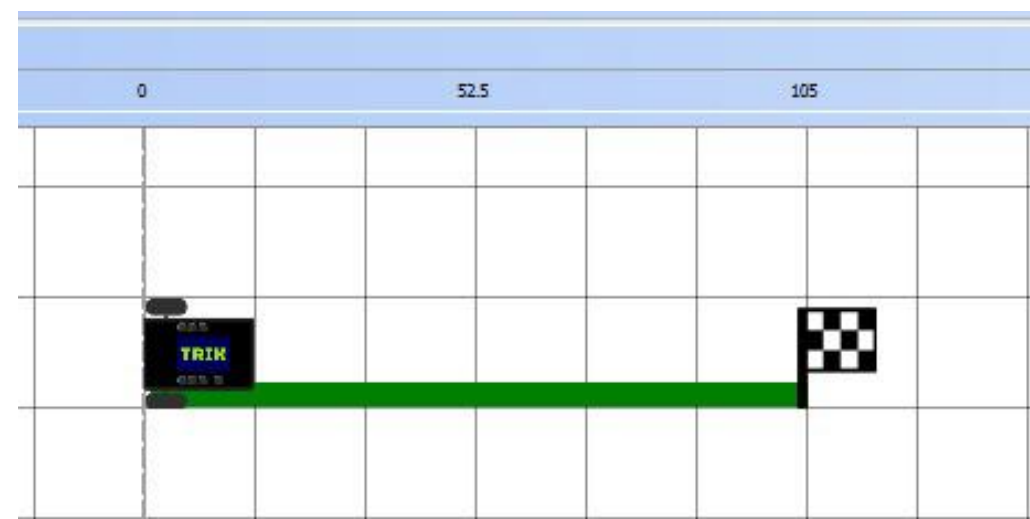
Точные перемещения. Задача

TRIK

Задача 2.1.3. Проехать вперед ровно на 1 метр и 5 сантиметров.
Использовать энкодерную модель

Вам пригодятся следующие параметры:

- $d = 5.6$ см (диаметр колеса)
- $CPR = 360$ (полный оборот колеса)



Точные перемещения. Решение



Решение.

Для решения этой задачи необходимо вспомнить элементарные формулы из курса школьной математики: расчет длины окружности и угла поворота

Введем следующие переменные:

d — диаметр колеса робота

dist — расстояние, которое необходимо проехать роботу

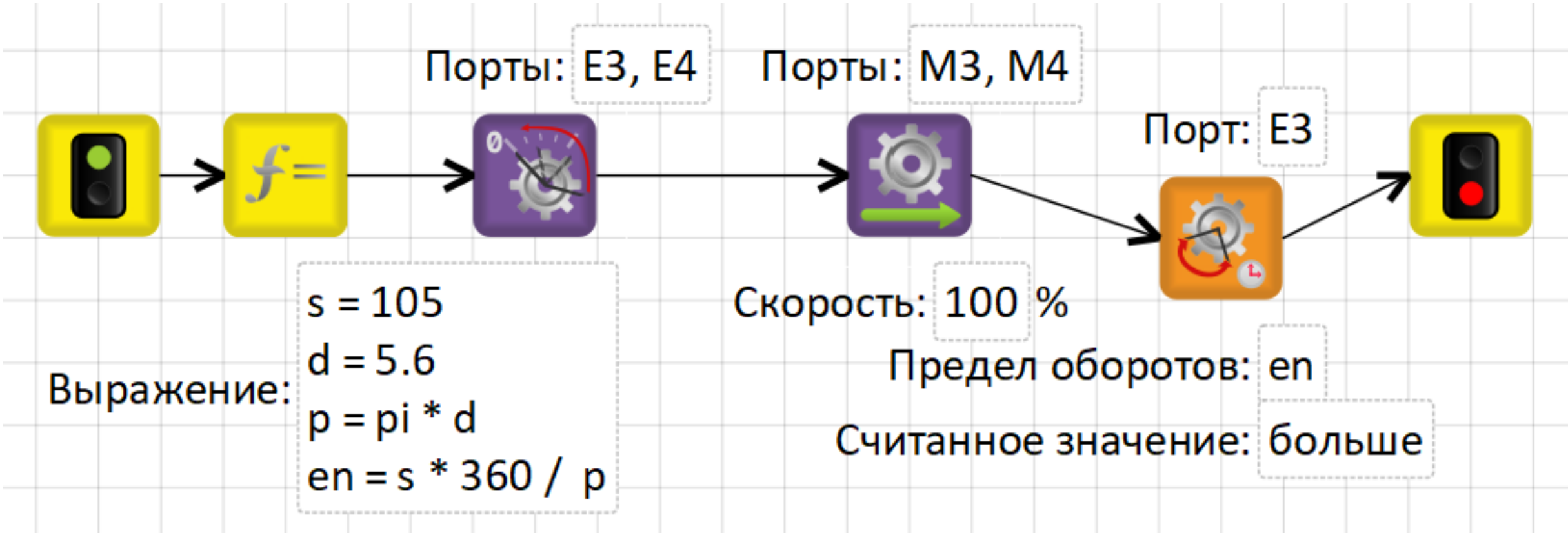
сpr — один оборот колеса в градусах (количество сигналов на оборот)

p — длина окружности

en — количество энкодеров



Точные перемещения. Решение



Точные перемещения. Задача

TRIK

Задача 2.1.4. (самостоятельно) Развернуться на месте ровно на 90 градусов.
Использовать энкодерную модель

Для решения вам понадобится дополнительный параметр:

- $b = 15.4$ см (ширина колеи робота)

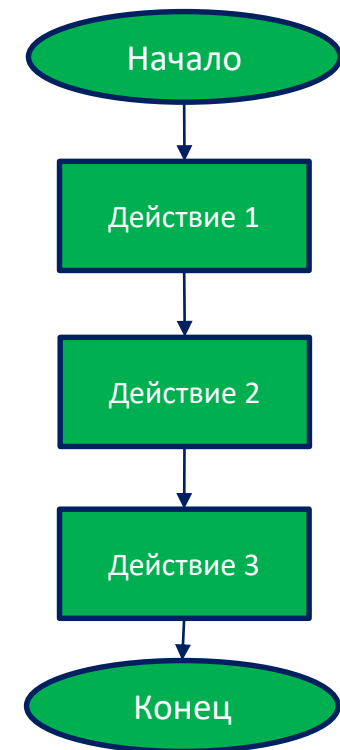


Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное число действий, при любом наборе исходных данных

Исполнитель: робот или любое другое устройство

Инструкции: включить мотор, ждать 3 секунды, повернуть серводвигатель на 80 градусов, включить диод и т.д.

Блок-схема — распространенный тип **схем** (графических моделей), описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде **блоков** различной формы, соединенных между собой линиями, указывающими направление последовательности



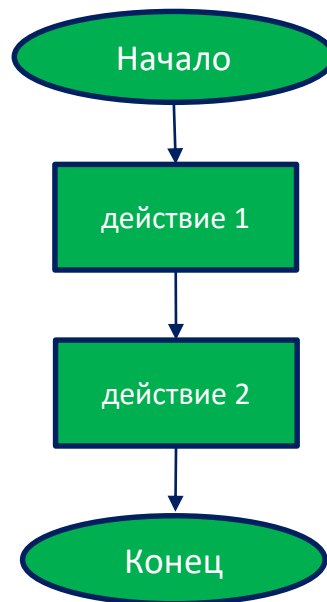
Следование (последовательность) — однократное выполнение операций в том порядке, в котором они записаны в тексте программы

Ветвление — однократное выполнение одной из двух или более операций, в зависимости от выполнения заданного условия

Цикл — многократное исполнение одной и той же операции до тех пор, пока выполняется заданное условие

Следование (последовательность) — однократное выполнение операций в том порядке, в котором они записаны в тексте программы

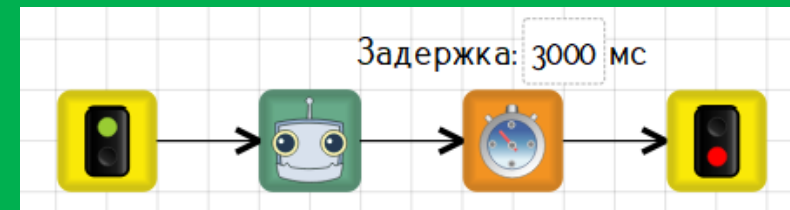
Блок-схема



Псевдокод

```
speed=-100;  
brick.motor(M2).setPower(100)  
brick.motor.(M3).setPower(speed)  
script.wait(2000)
```

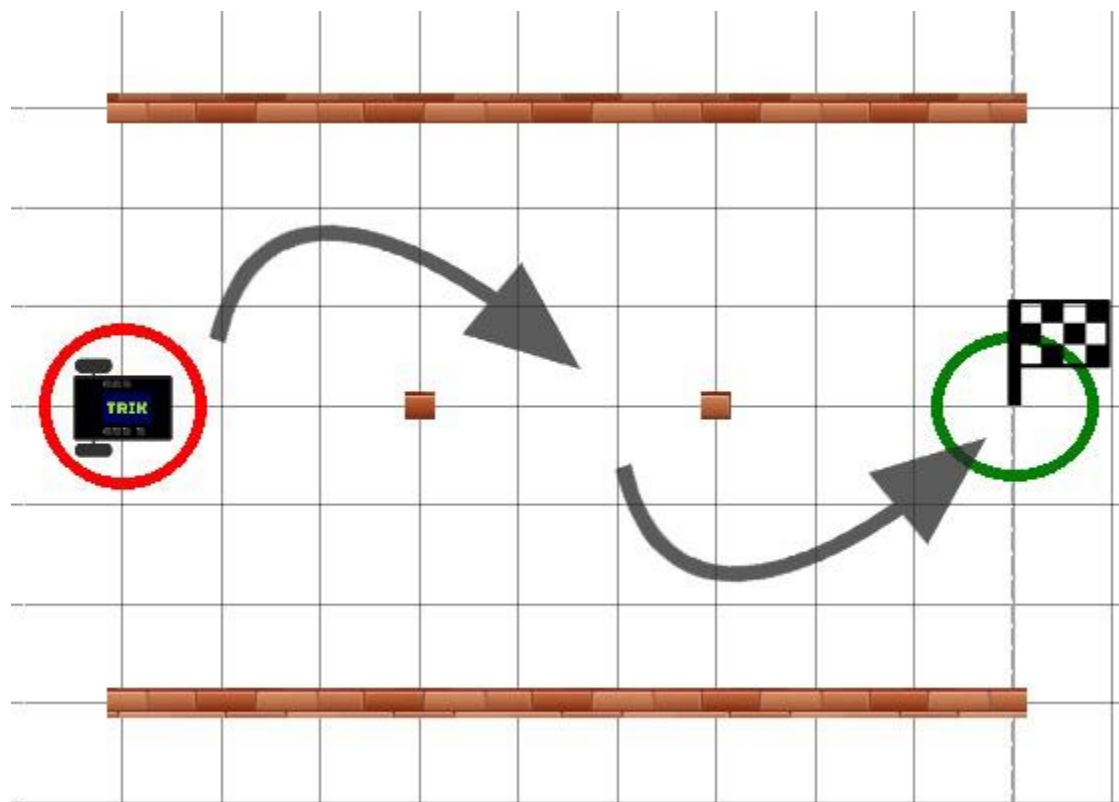
Пример в TRIK Studio



Следование. Задача

TRIK

Задача 2.1.5. (самостоятельно) Написать алгоритм движения модели «змейкой». Использовать энкодерную модель



Выполнение программы идет по одной из двух, нескольких или множества ветвей. Выбор ветви зависит от условия на входе ветвления и поступивших сюда данных

Существует две основные формы условной инструкции, встречающиеся в реальных языках программирования:

- условный оператор (оператор **if**)
- оператор многозначного выбора (оператор **switch**)



Ветвление. Условный оператор



Условный оператор реализует выполнение одной последовательности (ветви) команд при условии, что некоторое логическое выражение (условие) принимает значение «истина», и другой последовательности (ветви), если выражение "ложно". Любая из этих последовательностей может быть "пустой", т.е. не выполнять никаких действий

Встречаются следующие формы условного оператора:

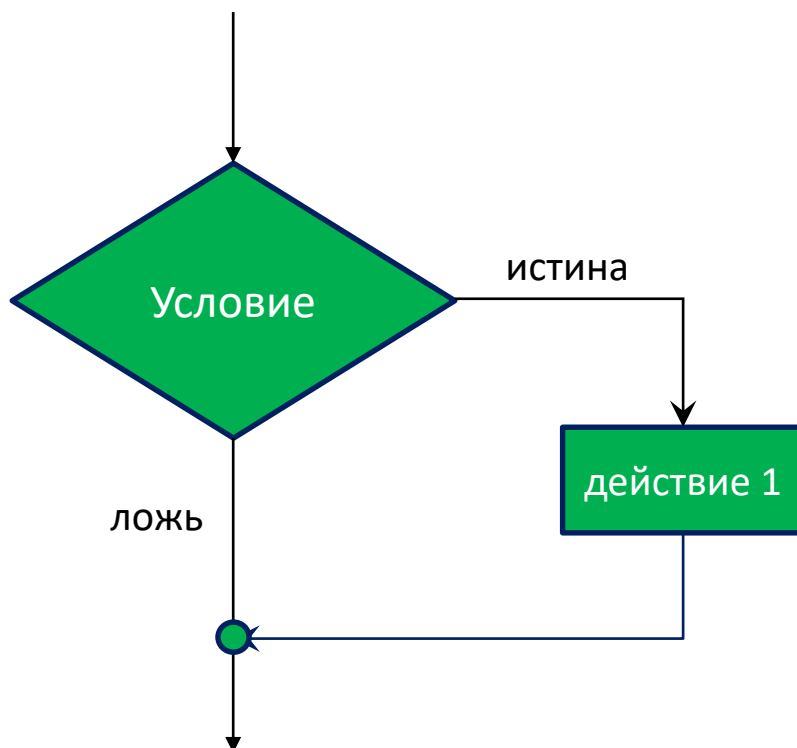
- Условный оператор с одной ветвью
- Условный оператор с двумя ветвями
- Условный оператор с несколькими условиями



Условный оператор с 1 ветвью

TRIK

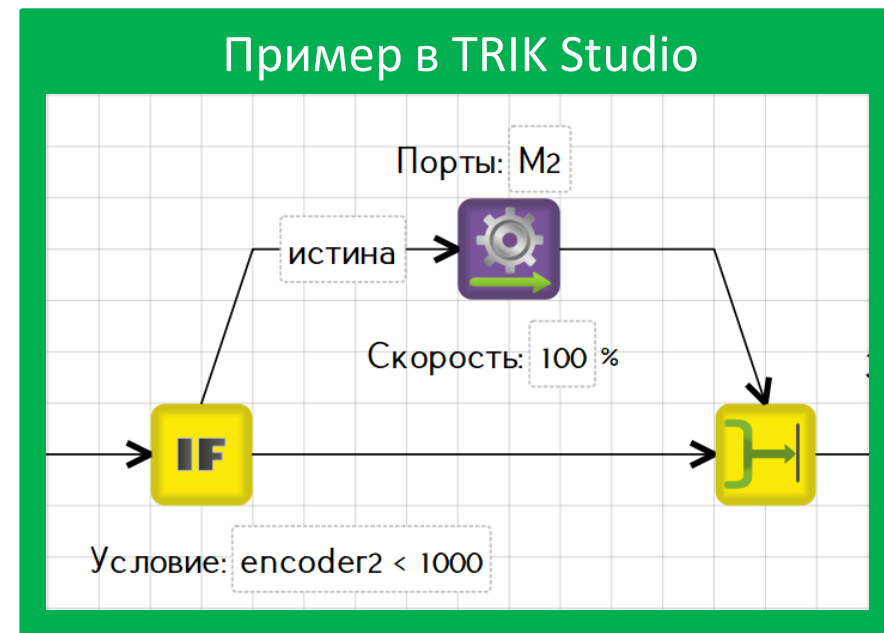
Блок-схема



Псевдокод

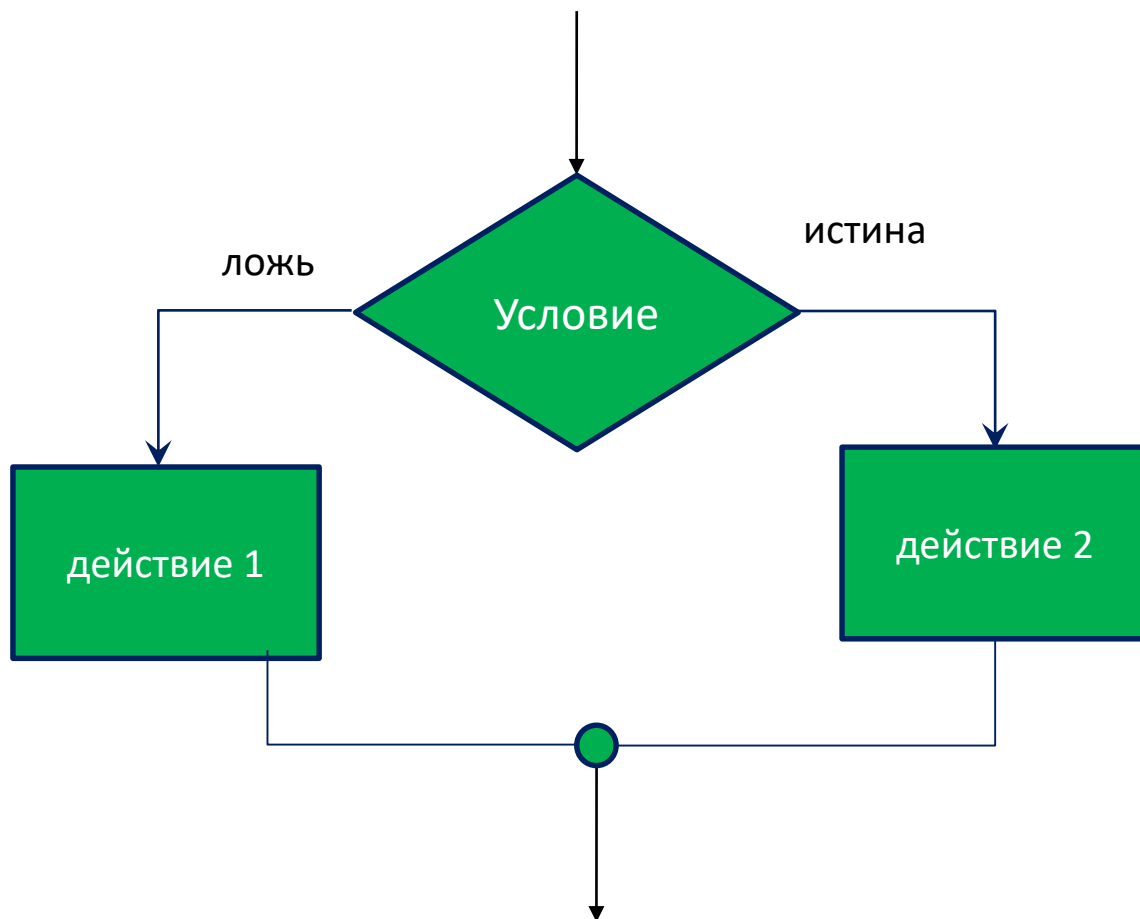
```
if (brick.encoder(E2).read() < 1000):  
    brick.motor(M2).setPower(100)
```

Пример в TRIK Studio



Условный оператор с 2 ветвями

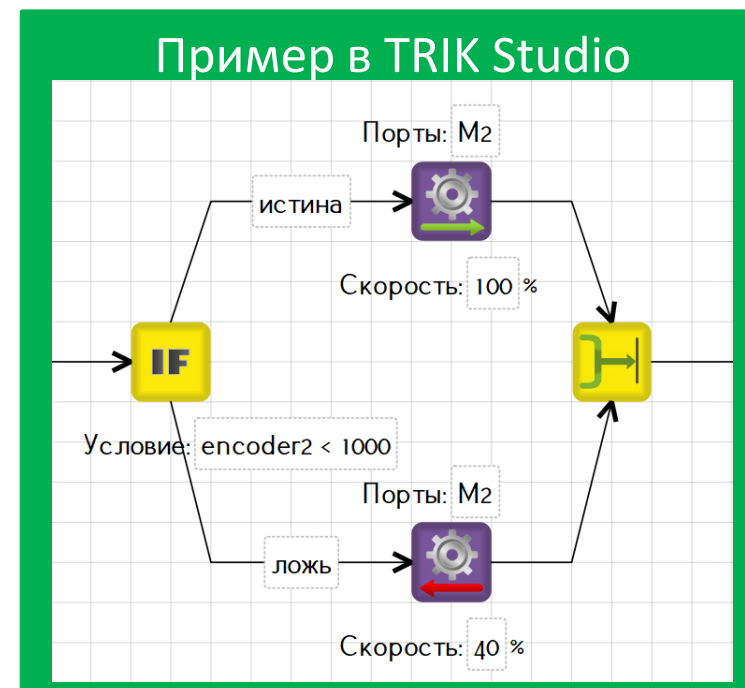
Блок-схема



Псевдокод

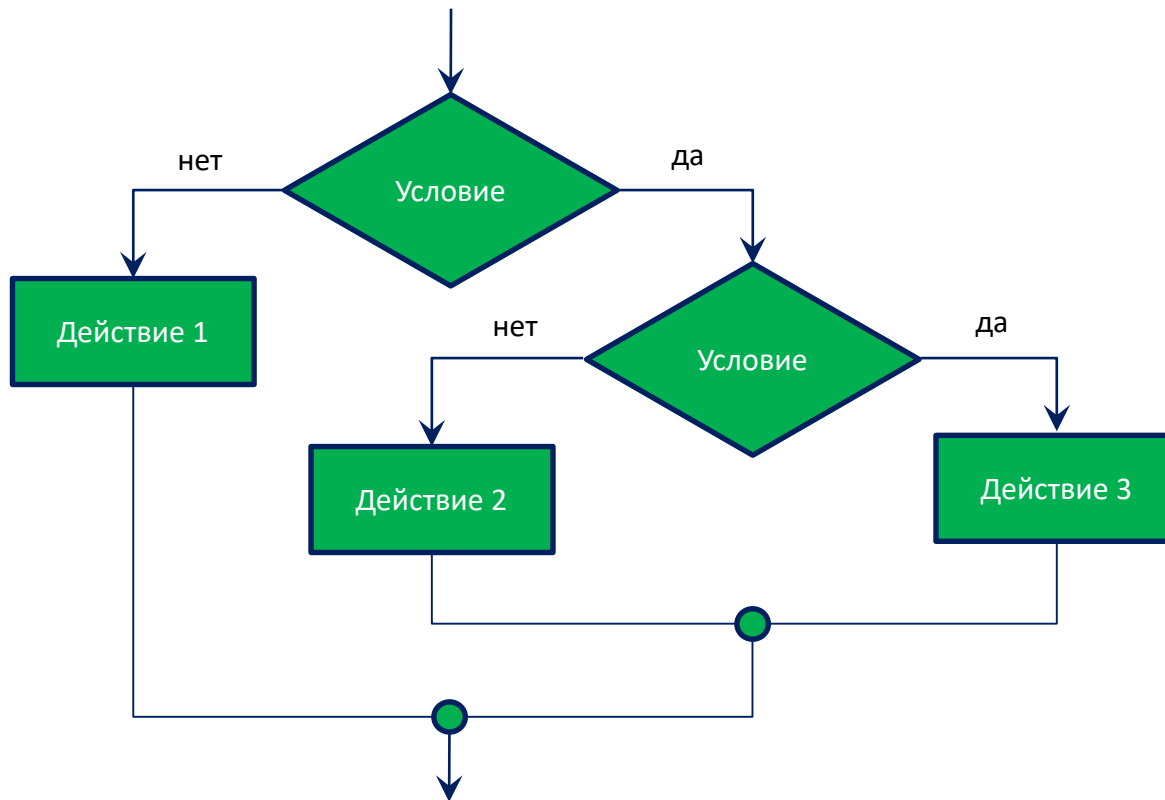
```
if (brick.encoder(E2).read() < 1000):  
    brick.motor(M2).setPower(100)  
else:  
    brick.motor(M2).setPower(-40)
```

Пример в TRIK Studio



Несколько условий

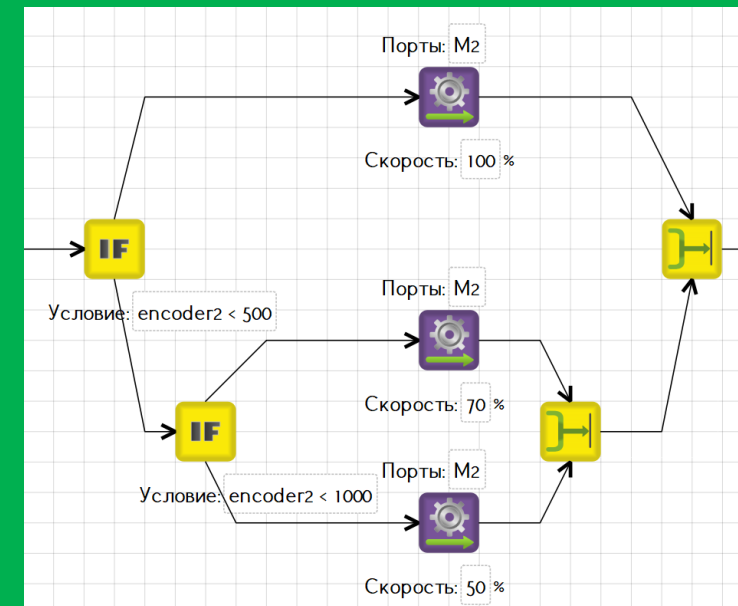
Блок-схема



Псевдокод

```
if (brick.encoder(E2).read() < 500):  
    brick.motor(M2).setPower(100)  
elif (brick.encoder(E2).read() < 1000):  
    brick.motor(M2).setPower(70)  
else:  
    brick.motor(M1).setPower(50)  
script.wait(2000)
```

Пример в TRIK Studio



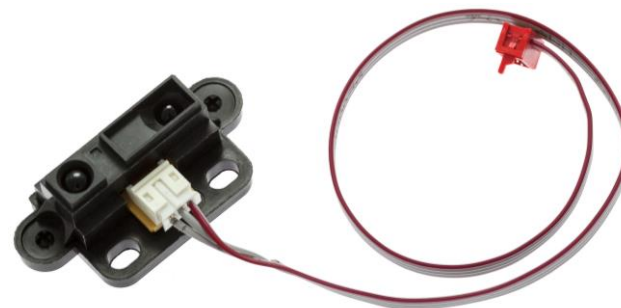
Ветвление. Задача

TRIK

Задача 2.1.6. вывести на экран грустный смайлик, если робот далеко от стены, и веселый, если близко, на 3 секунды или дольше. За границу считать значение 50 ИК датчика

Инфракрасный датчик расстояния — аналоговый датчик для измерения расстояния.

Рабочий диапазон: 10–80 см



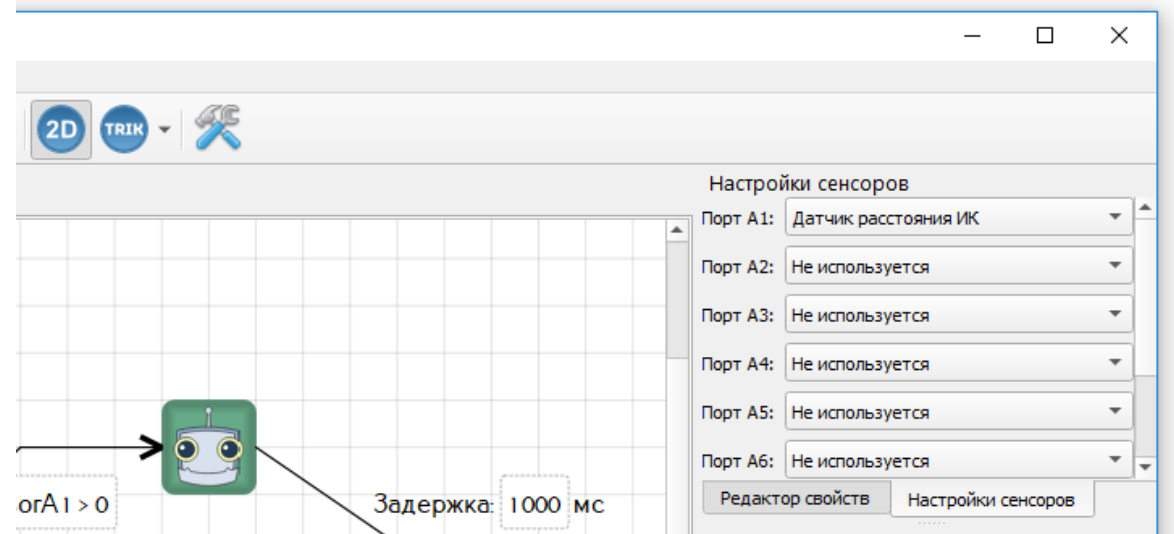
В TRIK Studio все датчики подключаются на панели **«Настройка сенсоров»**

Ветвление. Задача

TRIK

Задача 2.1.6. вывести на экран грустный смайлик, если робот далеко от стены, и веселый, если близко, на 3 секунды или дольше. За границу считать значение 50 ИК датчика

В TRIK Studio все датчики подключаются на панели «Настройка сенсоров»

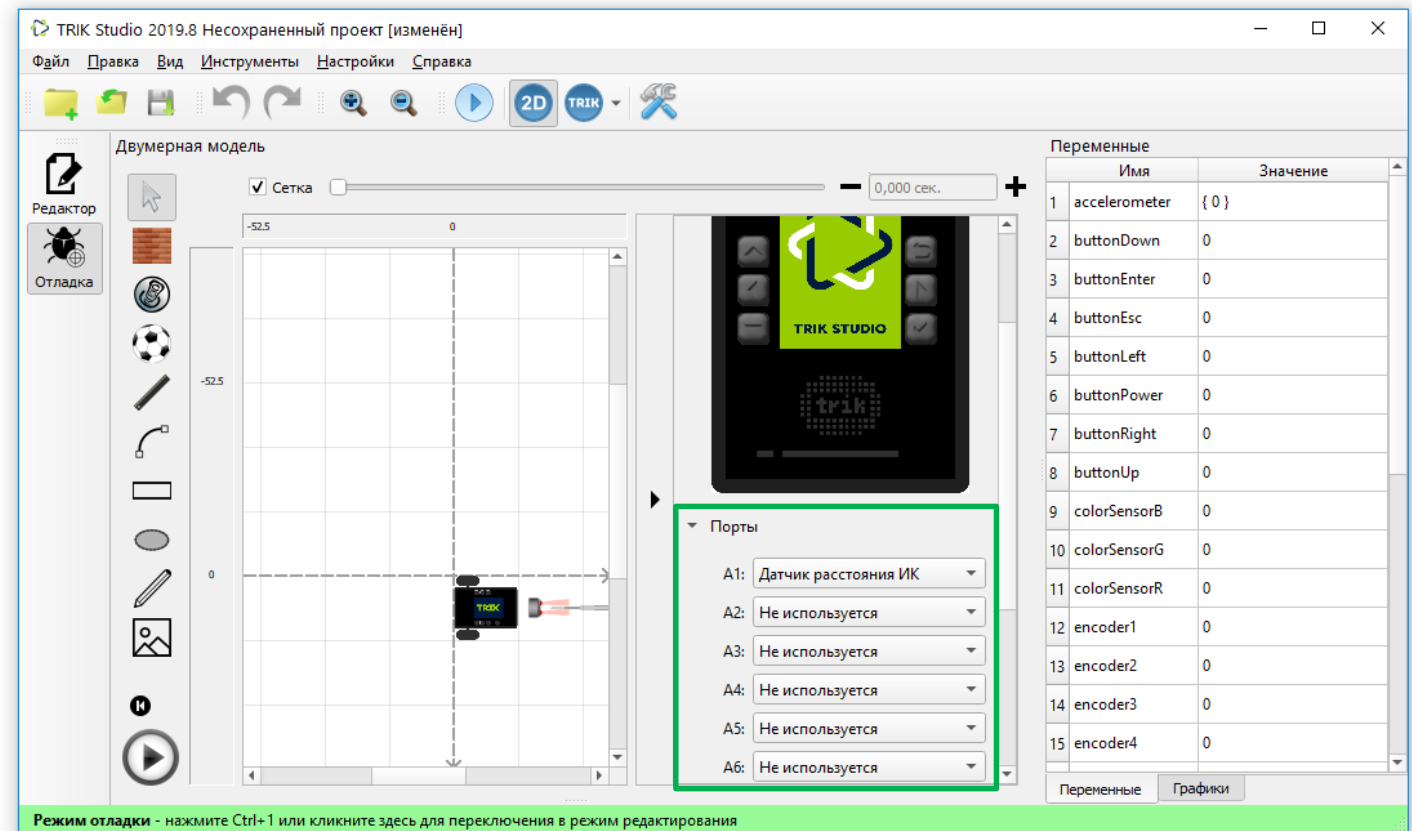


Ветвление. Задача



Задача 2.1.6. вывести на экран грустный смайлик, если робот далеко от стены, и веселый, если близко, на 3 секунды или дольше. За границу считать значение 50 ИК датчика

Или в режиме 2D-
модели на панели
«Порты»



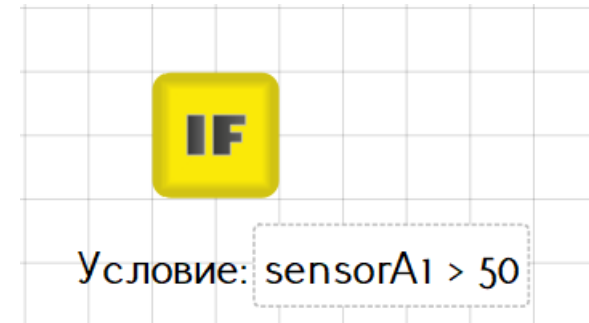
Ветвление. Задача

TRIK

Задача 2.1.6. вывести на экран грустный смайлик, если робот далеко от стены, и веселый, если близко, на 3 секунды или дольше. За границу считать значение 50 ИК датчика

Для ветвления в TRIK Studio используется блок **«Условие»**, у которого имеется только одно свойство — само условие

Использование значений датчика осуществляется в TRIK Studio через **сенсорные переменные**



При подключении датчика:

- к порту A1 используется сенсорная переменная **sensorA1**
- к порту A2 — **sensorA2**
- и т.д.

Сенсорной переменной нельзя присвоить значение. В нее записывается регулярно показание с датчика

Для задания различных условий роботу необходимы операторы сравнения и логические операторы

Операторы сравнения

Оператор	Синтаксис	Пример
равенство	<code>==</code>	<code>enterButton == 1</code>
неравенство	<code>!=</code>	<code>rightButton != 0</code>
больше	<code>></code>	<code>sensorA1 > 50</code>
меньше	<code><</code>	<code>sensorA2 < 30</code>
больше или равно	<code>>=</code>	<code>sensorA3 >= 50</code>
меньше или равно	<code><=</code>	<code>sensorA4 <= 50</code>

Для задания различных условий роботу необходимы операторы сравнения и логические операторы

Логические операторы

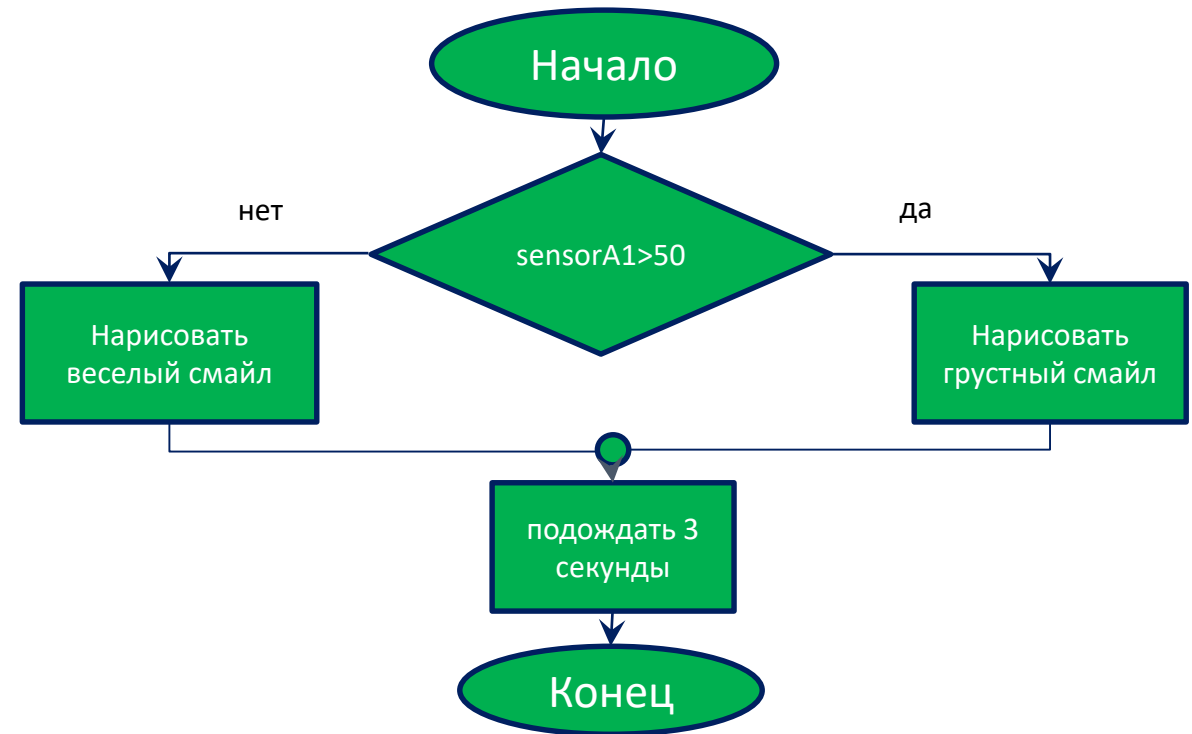
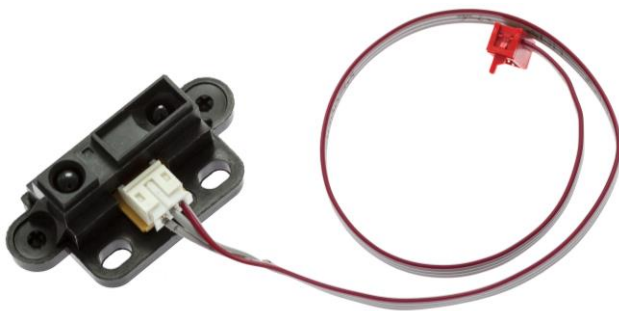
Оператор	Синтаксис	Пример
логическое отрицание, НЕ	!	<code>!flag</code>
логическое умножение, И	&& или and	<code>(sensorA1>20) and (sensorA1<60)</code>
логическое сложение, ИЛИ	 или or	<code>(sensorA1<30) or (sensorA1>70)</code>

Ветвление. Задача

Задача 2.1.6. вывести на экран грустный смайлик, если робот далеко от стены, и веселый, если близко, на 3 секунды или дольше. За границу считать значение 50 ИК датчика

Инфракрасный датчик расстояния
— аналоговый датчик для измерения расстояния

Рабочий диапазон: 10–80 см



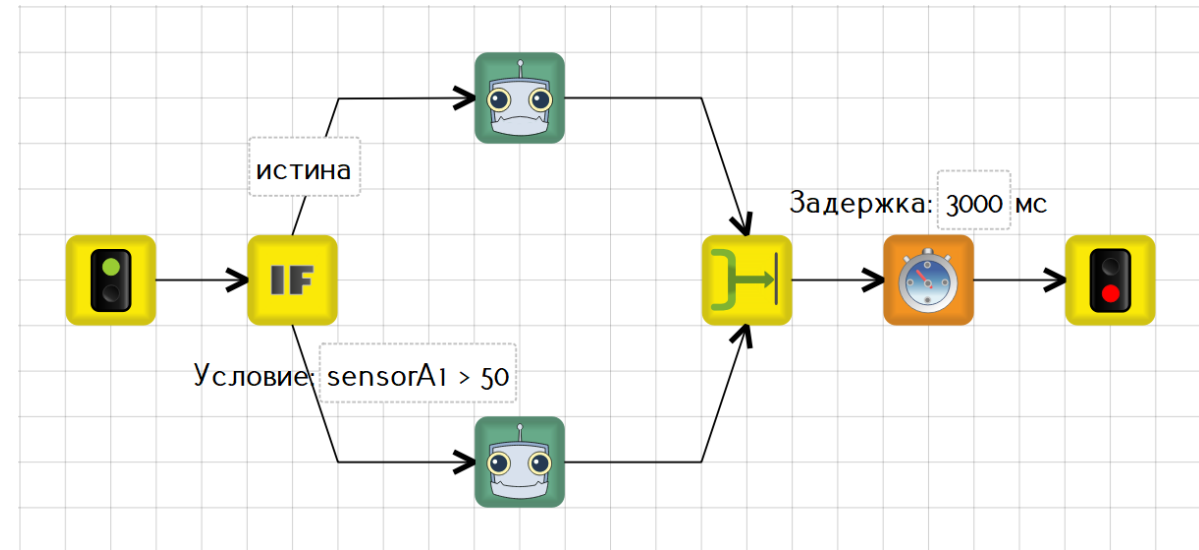
Ветвление. Задача

Задача 2.1.6. вывести на экран грустный смайлик, если робот далеко от стены, и веселый, если близко, на 3 секунды или дольше. За границу считать значение 50 ИК датчика.

Псевдокод

```
if (brick.sensor(A1).read() > 50):  
    brick.sadSmile()  
else:  
    brick.smile()  
script.wait(3000)
```

Решение в TRIK Studio

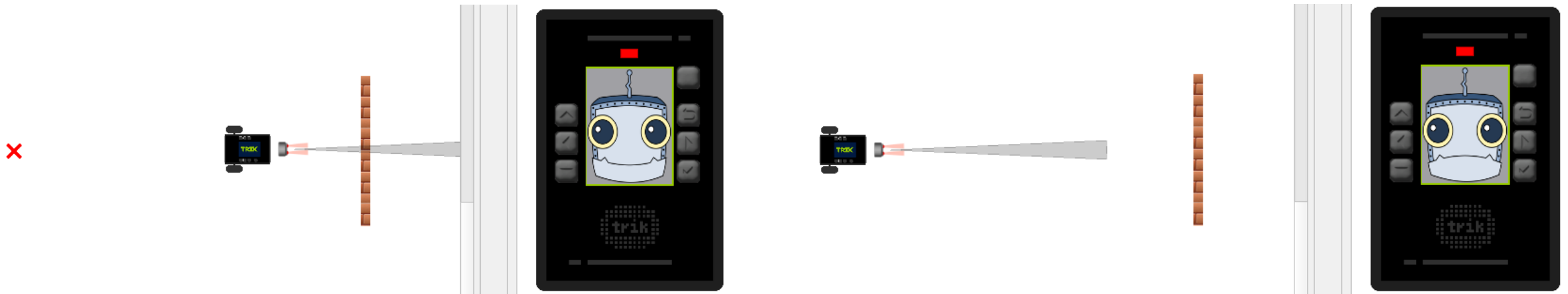


На связях, идущих от условия, указывается в свойствах **ИСТИНА** и **ЛОЖЬ** для определения дальнейших действий, когда условие верно, и когда — нет.

Ветвление. Задача

TRIK

Для проверки задачи используйте 2 разных поля: на одном стена близко к роботу, на другом - далеко.



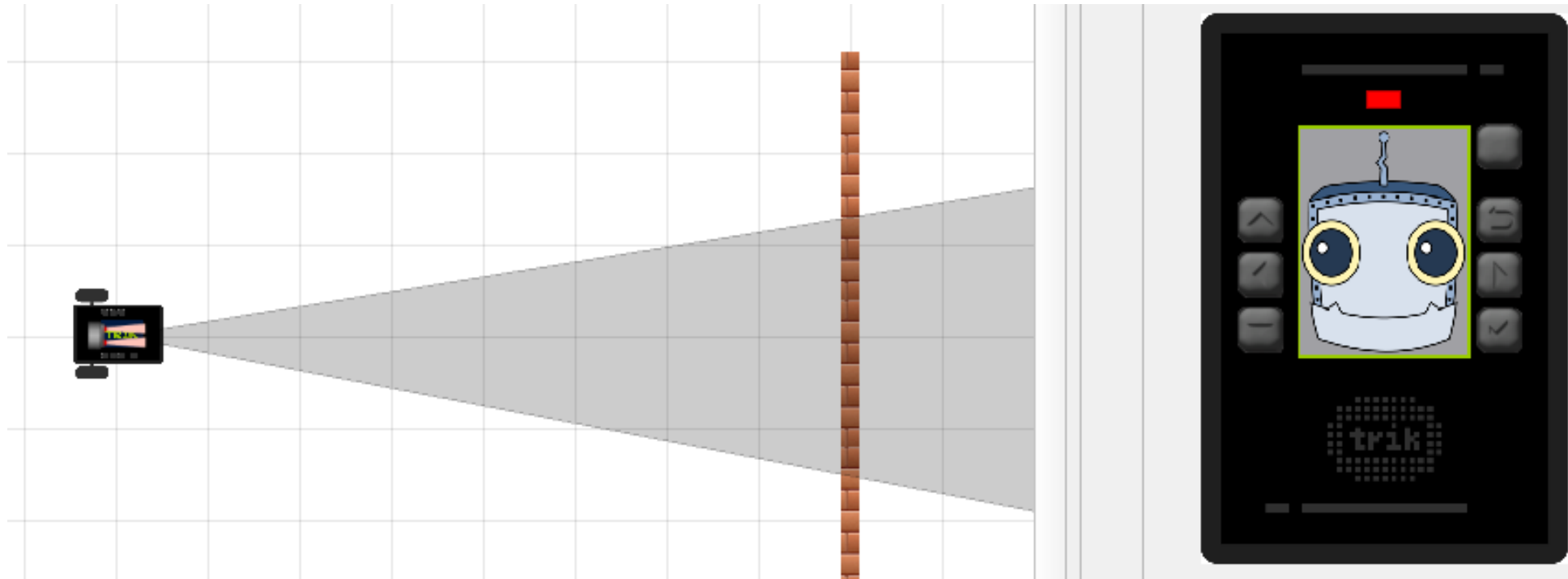
Для проверки можно использовать поля **2.1.6 – field1.xml**, **2.1.6 – field2.xml**

Ветвление. Задача

TRIK

Задача 2.1.7 (самостоятельно) Вывести на экран:

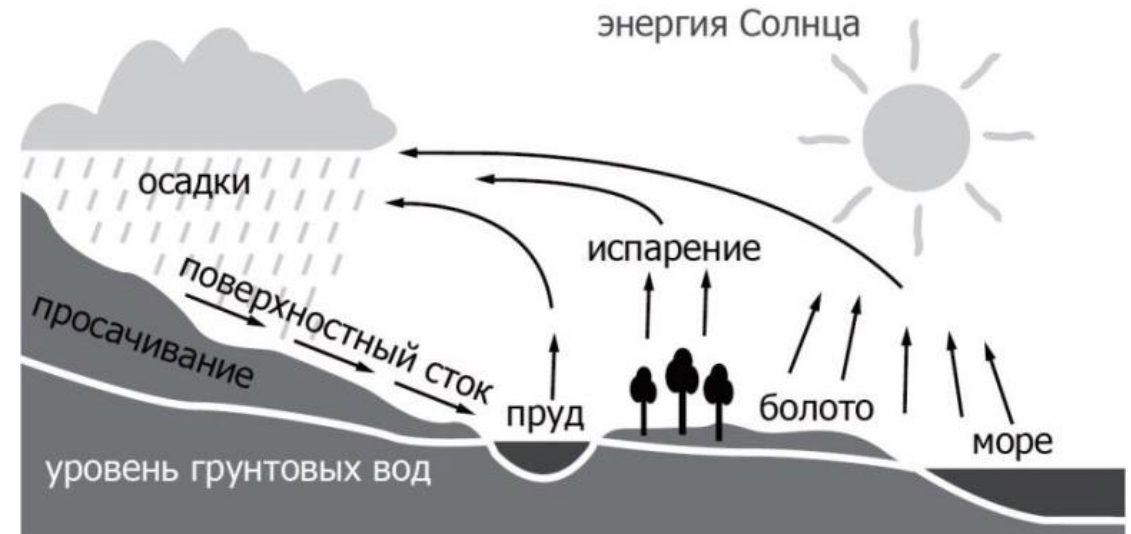
- Веселый смайлик, если ИК датчик выдает до 40.
- Слово «неопределенность», если ИК датчик выдает от 40 до 60.
- Грустный смайлик — в противном случае.



Цикл — управляющая конструкция в языках программирования для организации многократного выполнения набора инструкций.

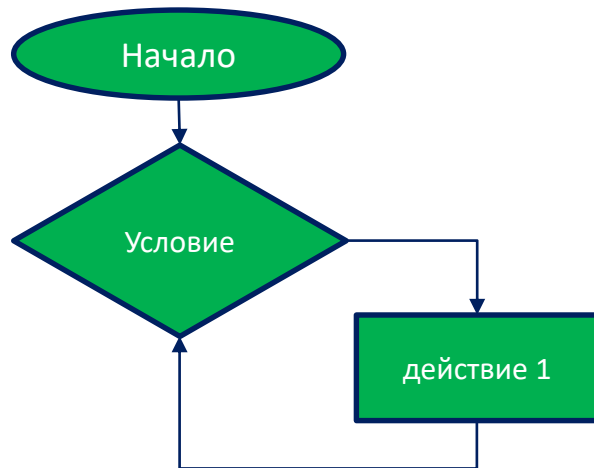
Существует 4 основных вида циклов:

- Безусловные циклы
- Цикл с предусловием
- Цикл с постусловием
- Цикл со счетчиком



Цикл безусловный

Блок-схема

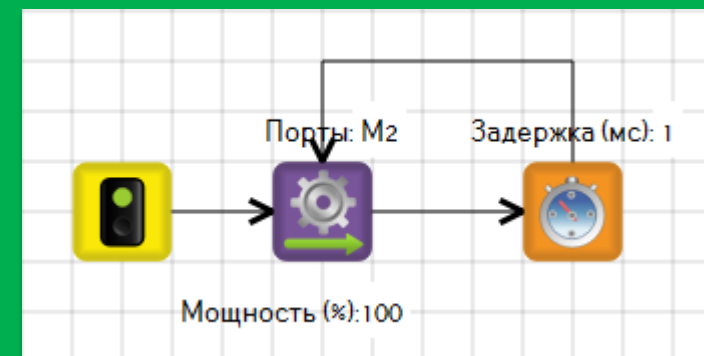


В этом случае у программы может не быть конца!

Псевдокод

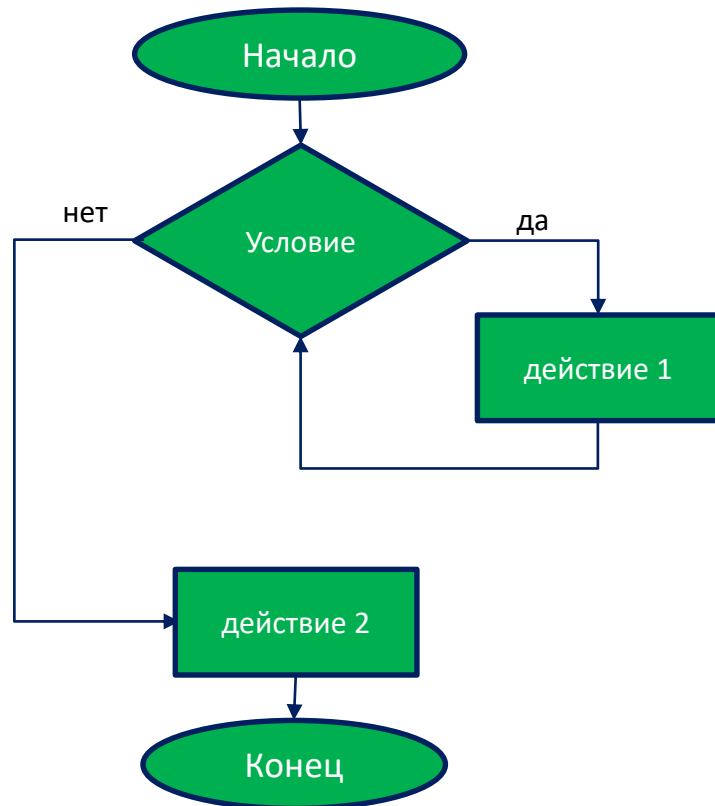
```
while True:  
    brick.motor(M2).setPower(100)
```

Пример в TRIK Studio



Цикл с предусловием

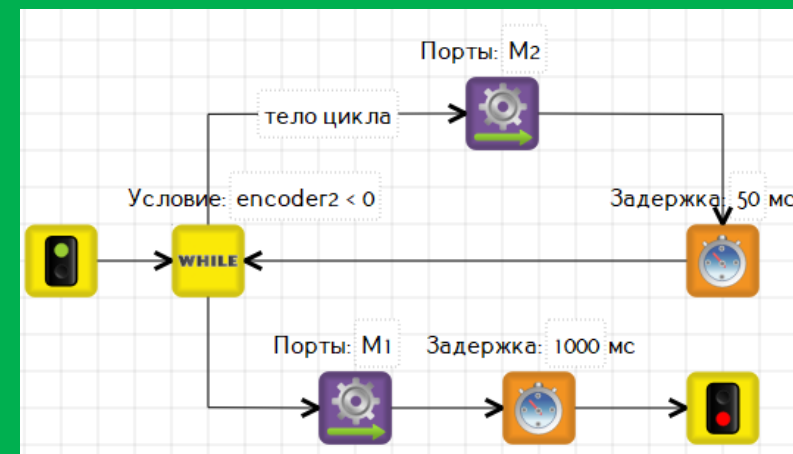
Блок-схема



Псевдокод

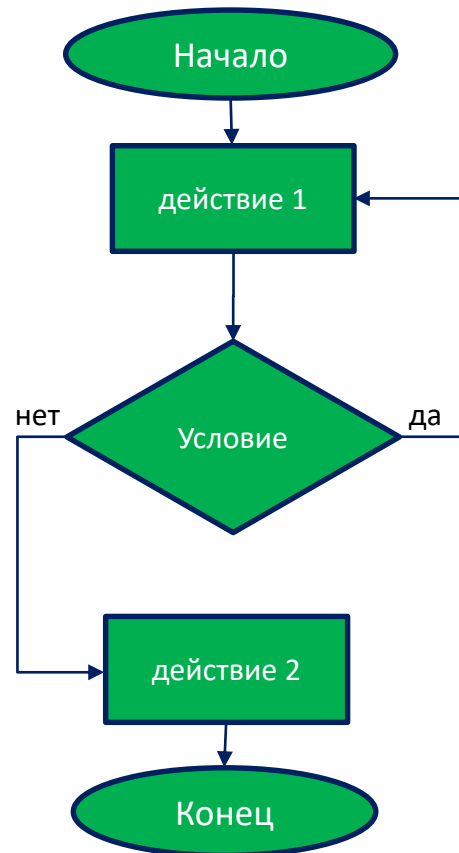
```
while brick.encoder(E2).read() < 500:  
    brick.motor(M2).setPower(100)  
    brick.motor(M1).setPower(100)
```

Пример в TRIK Studio



Цикл с постусловием

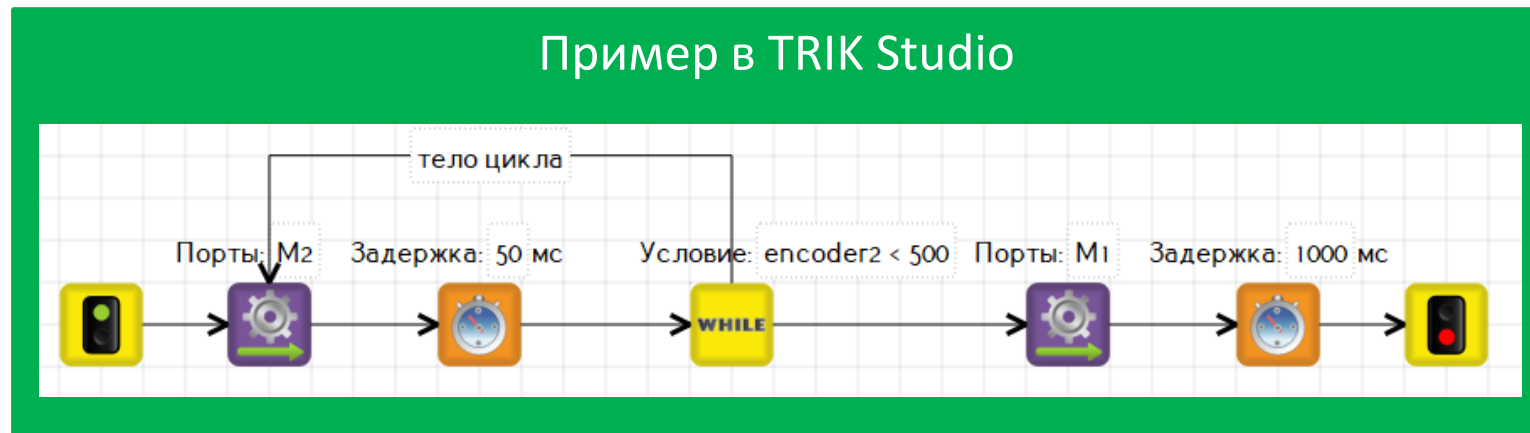
Блок-схема



Псевдокод

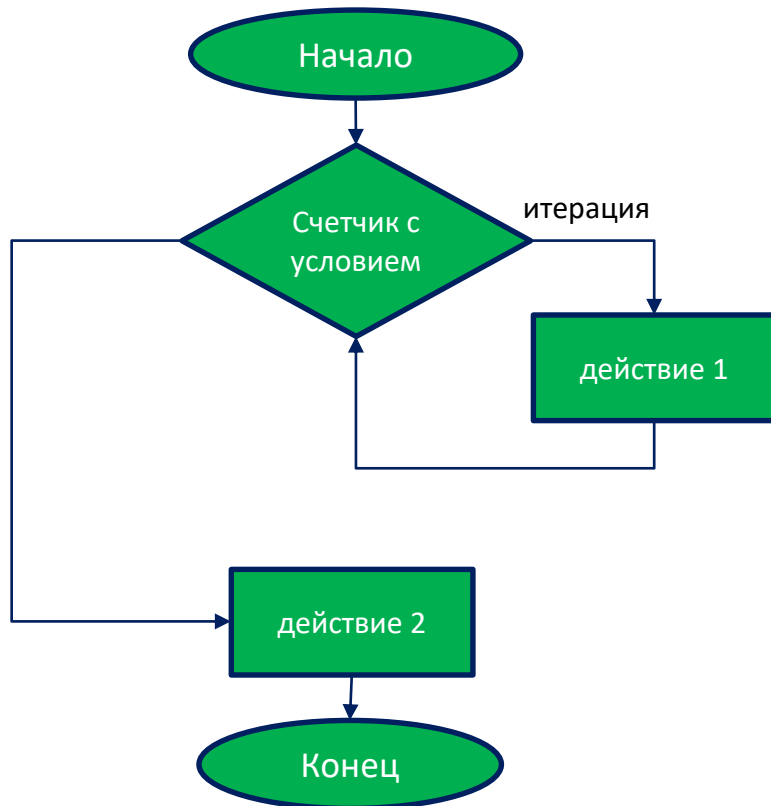
```
do {  
    brick.motor(M2).setPower(100)  
    script.wait(1)  
} while (brick.encoder(E2).read() < 500)  
brick.motor(M1).setPower(100)
```

Пример в TRIK Studio



Цикл с итерациями

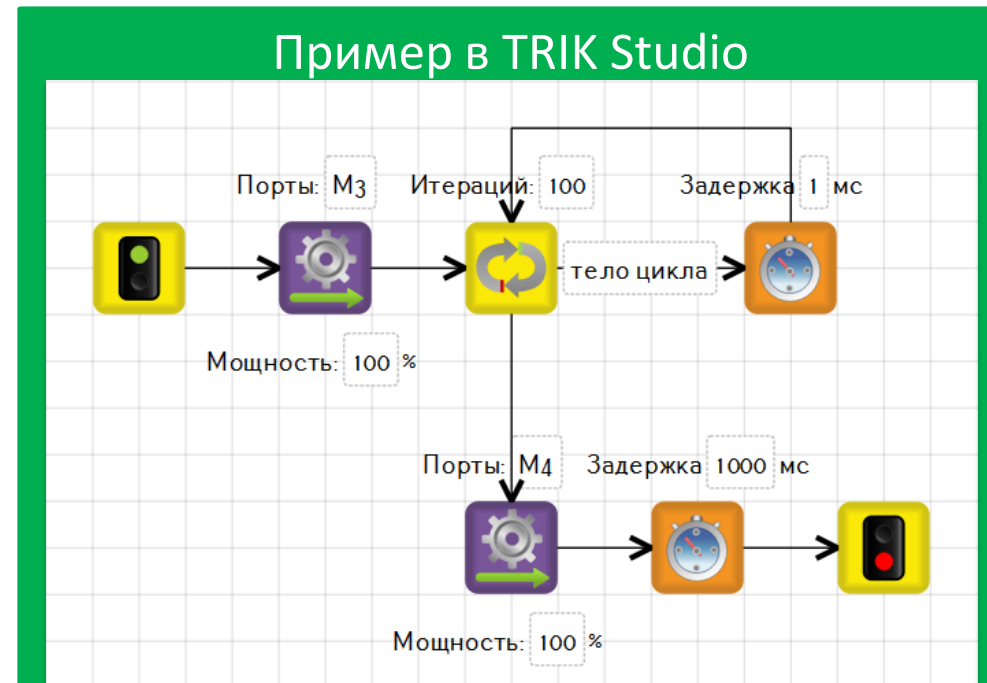
Блок-схема



Псевдокод

```
robot.motor.[M3].setPower(100);  
for (i = 0; i < 1000; i++)  
    robot.wait(1);  
robot.motor.[M4].setPower(100);
```

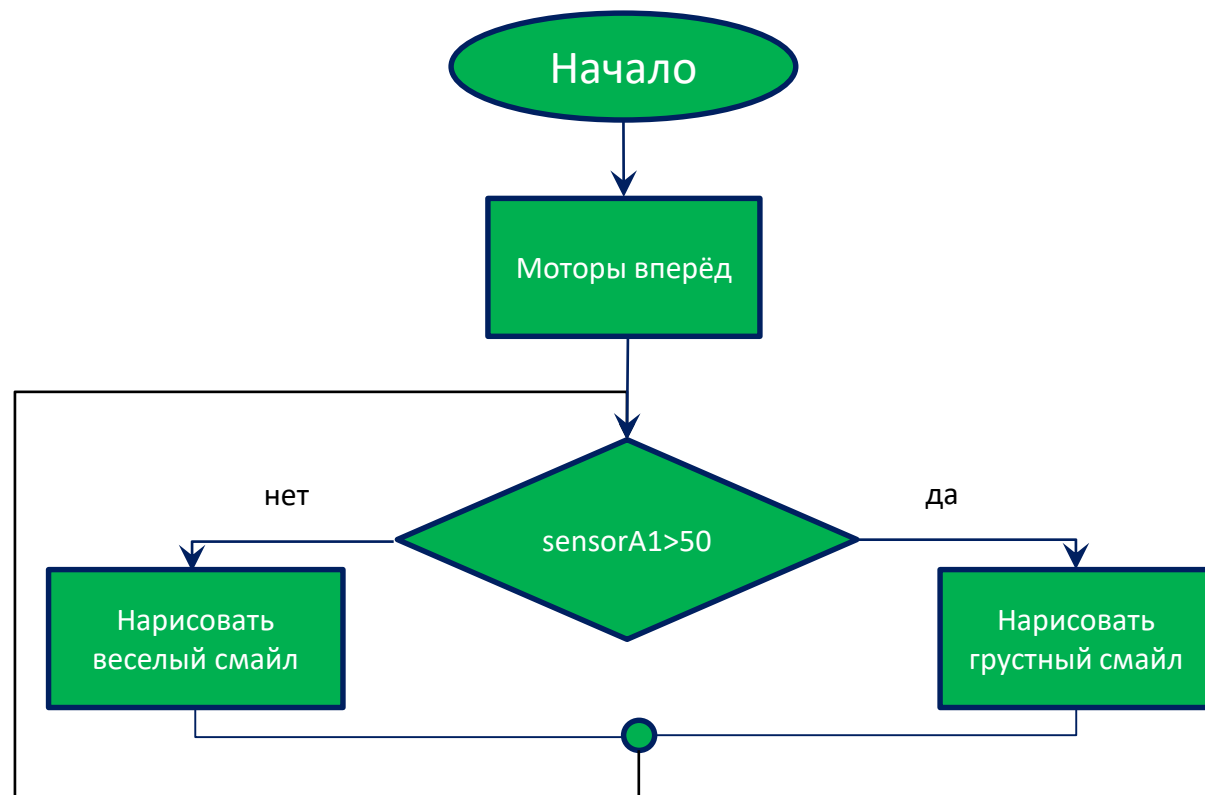
Пример в TRIK Studio



Цикл. Задача

Задача 2.1.10. «Настройка робота» Робот движется прямо через черные и белые поля. Непрерывно выводить на экран веселый смайлик, если робот на черном поле, и грустный, если на белом. За границу считать значение 50 датчика света.

Датчик освещенности – аналоговый датчик для измерения освещенности. Выдает значение от 0 до 100.



Цикл. Задача

TRIK

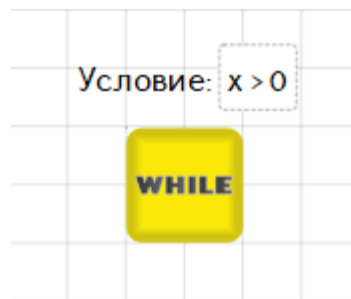
Задача 2.1.10. «Настройка робота» Робот движется прямо через черные и белые поля. Непрерывно выводить на экран веселый смайлик, если робот на черном поле, и грустный, если на белом. За границу считать значение 50 датчика света.



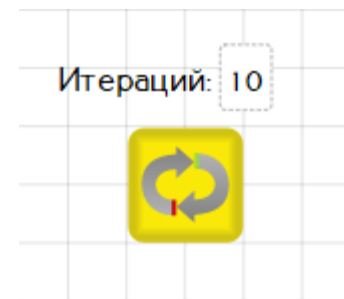
Цикл. Задача

Задача 2.1.10. «Настройка робота» Робот двигается прямо через черные и белые поля. Непрерывно выводить на экран веселый смайлик, если робот на черном поле, и грустный, если на белом. За границу считать значение 50 датчика света.

Для циклов с условиями в TRIK Studio используется блок **«Цикл с предусловием»**.



А с итерациями — блок **«Цикл»**.



Бесконечные циклы реализуются путем соединения одного из блоков с каким-либо предыдущим.

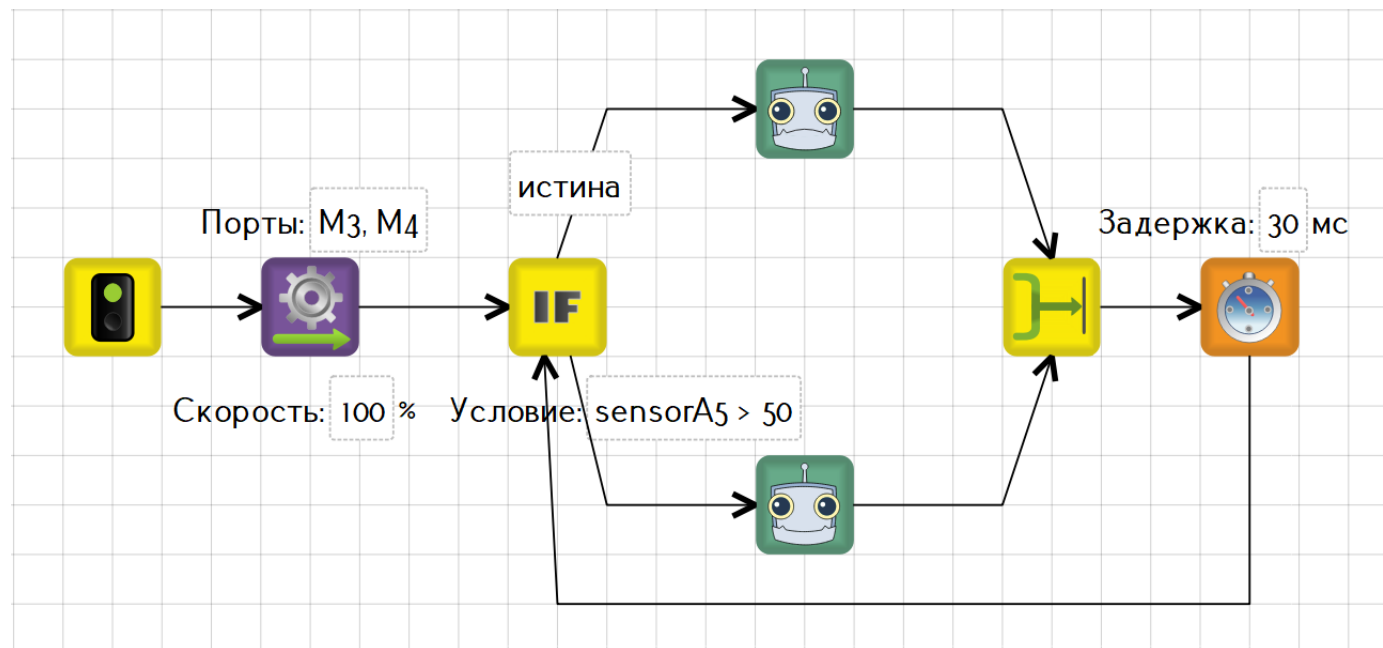
Цикл. Задача

Задача 2.1.10. «Настройка робота» Робот двигается прямо через черные и белые поля. Непрерывно выводить на экран веселый смайлик, если робот на черном поле, и грустный, если на белом. За границу считать значение 50 датчика света.

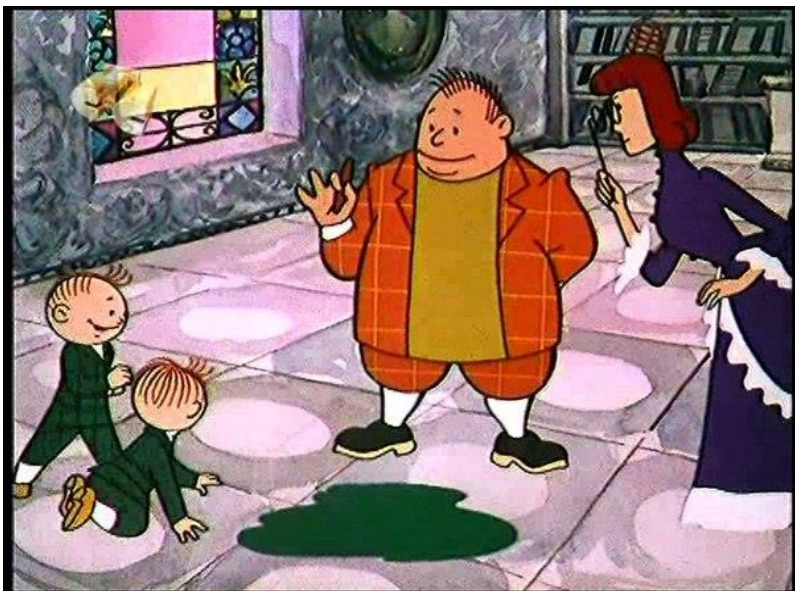
Псевдокод

```
brick.motor(M3).setPower(100)
brick.motor(M4).setPower(100)
while True:
    if (brick.sensor(A5).read() > 50):
        brick.smile()
    else:
        brick.sadSmile()
    script.wait(30)
```

Решение в TRIK Studio



Задача 2.1.11: «Кентервильское привидение».



Кентервильский робот: привидение рисует каждую ночь лужи красной краской. Убедившись, что лужа красная, он довольный скрывается из виду. Когда красная краска заканчивается, он рисует лужи зеленым и расстроенный отключается.

Научите робота определять цвет лужи и выключаться, если лужа зеленая. В первый раз робот всегда в приподнятом настроении.

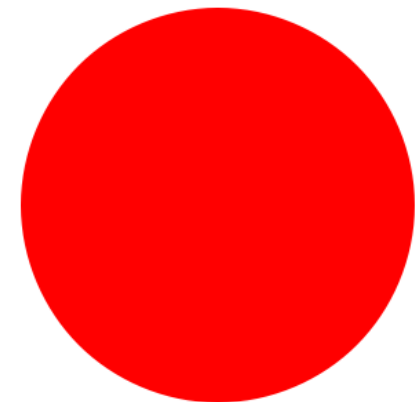
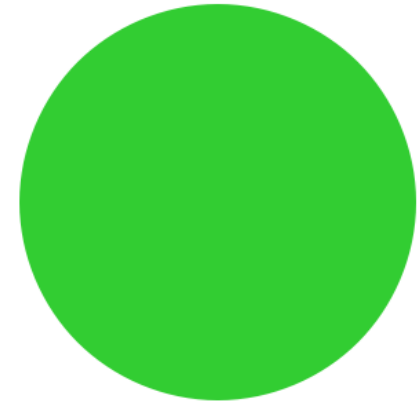
Задача 2.1.11: «Кентервильское привидение»

Пояснение.

Выводить на экран:

- Веселый смайлик, если робот видит красную лужу или пустой пол
- Грустный смайлик (в течение 3 секунд) в противном случае (зеленая лужа). И закончить выполнение программы.

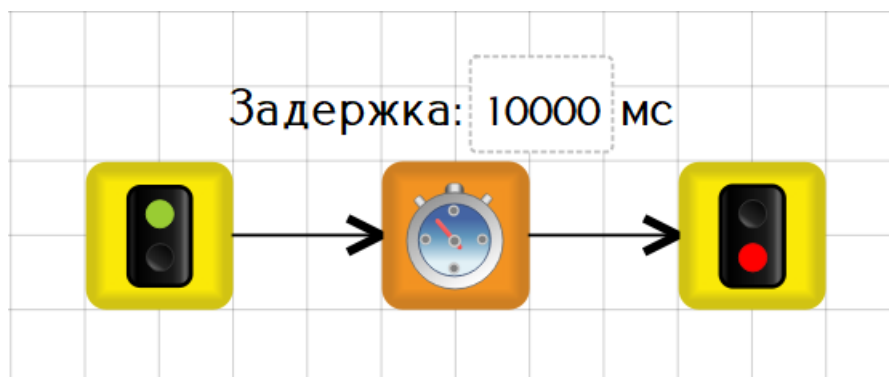
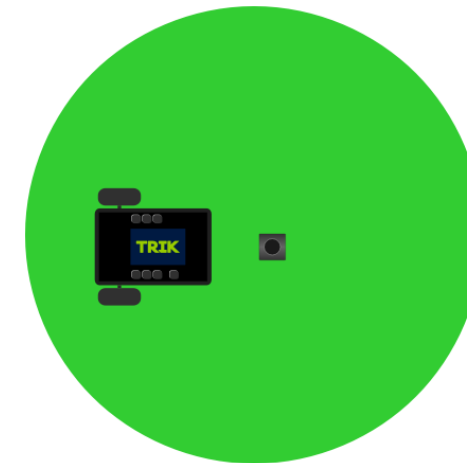
Считывать новое значение с датчика **каждую секунду**. Использовать блок **«Цикл с условием»**



Задача 2.1.11: «Кентервильское привидение»

Как узнать значения цветов?

Подключить нужный датчик, запустить программу и посмотреть, что выдает датчик в 2D на панели **Переменные**.



Красный – 78,
пустой пол – 0,
Зелёный – 37.

21	sensorA1	0
22	sensorA2	0
23	sensorA3	0
24	sensorA4	0
25	sensorA5	37
26	sensorA6	0

Задача 2.1.11: «Кентервильское привидение»

В виртуальном мире показания **всегда точные и одинаковые** на одном цвете. При составлении алгоритма нужно учитывать, что потом он будет запущен на реальном устройстве. и там показания при наведении на поверхность могут отличаться при разных запусках. В этом случае лучше сразу проверку делать в диапазоне. Например:

Красный – больше 75,
пустой пол – меньше 5,
Зелёный – больше 35 и меньше 39.

Решите задачу до конца, используя циклы.

Задача на реальном роботе: Определение цветов

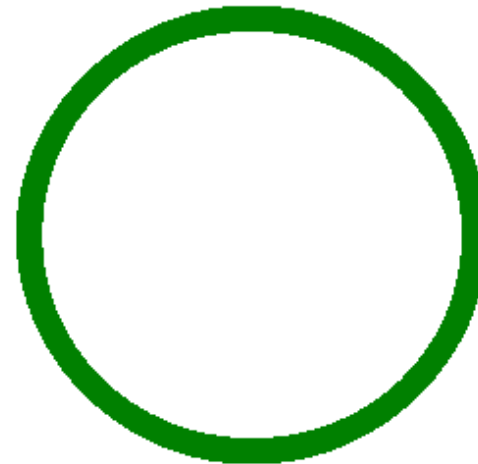
Закрепите на роботе датчик освещенности таким образом, чтобы он был на фиксированном расстоянии от поверхности (8-20 мм). Выберите 2 поверхности с разной освещенностью (цвета отражают по-разному). Например, цвет парты и цвет листа тетради.

Решите задачу.

Когда робот находится в руках (датчик освещенности ничего не отражает) или стоит на белом листе на экран непрерывно выводится весёлый смайлик. Когда робота ставим на парту на экран выводится грустный смайлик на 3 секунды и работа программы завершается.

Задача 2.1.12 (самостоятельно) «Разгон и торможение»

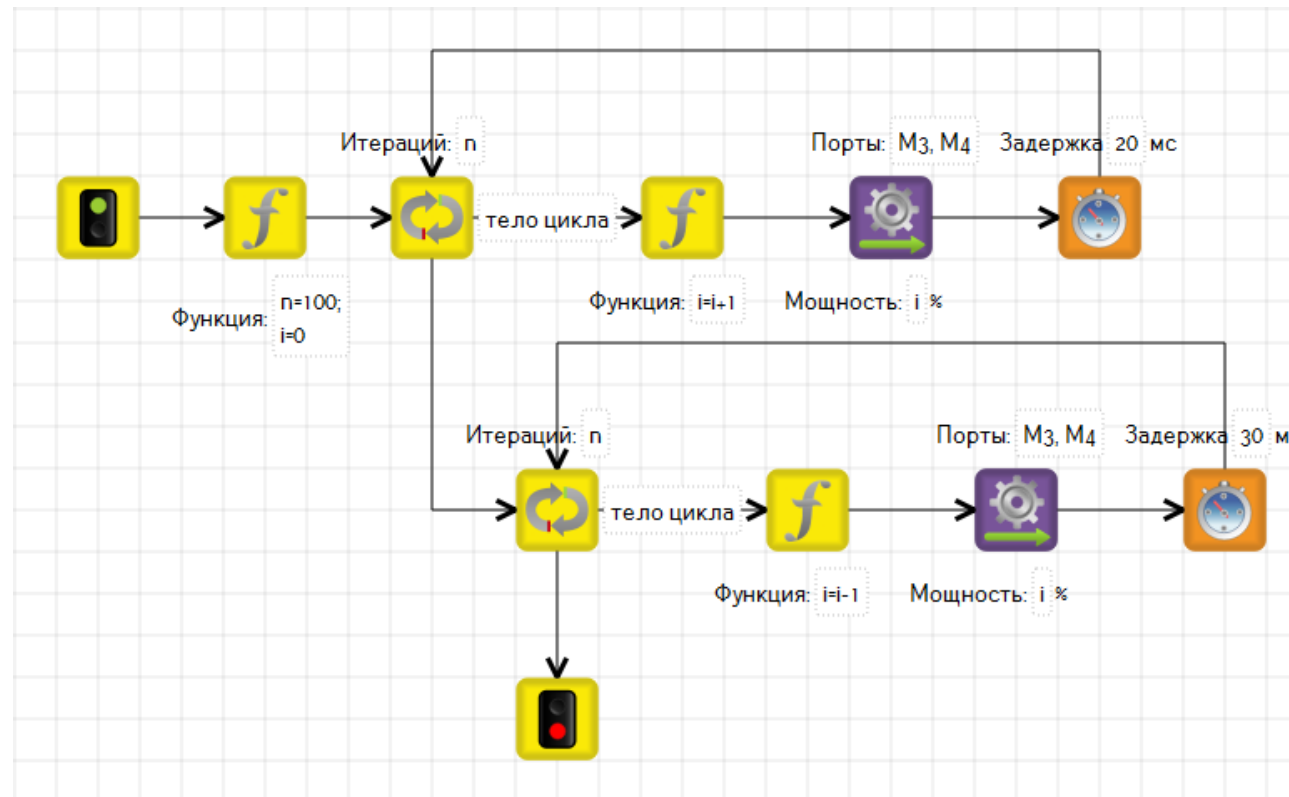
Напишите программу: плавный разгон робота от 0 до 100 в течение 2 секунд, а затем плавное торможение от 100 до 0 в течение 3 секунд. Используйте блок «Цикл»



Цикл. Задача

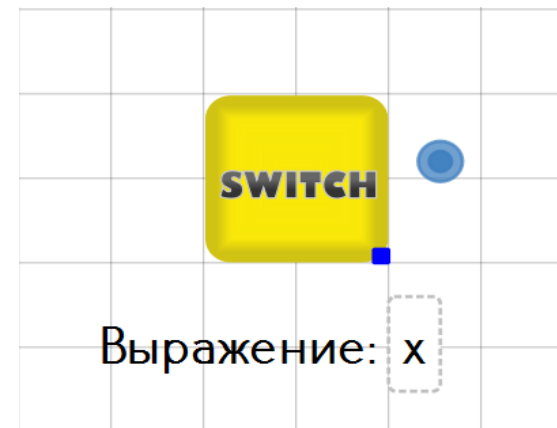
Задача 2.1.12 (самостоятельно) «Разгон и торможение»

Напишите программу: плавный разгон работа от 0 до 100 в течение 2 секунд, а затем плавное торможение от 100 до 0 в течение 3 секунд. Используйте блок «Цикл»



Представляет собой структуру, построенную по принципу меню, и содержит все возможные варианты условий и инструкции, которые следует выполнить в каждом конкретном случае.

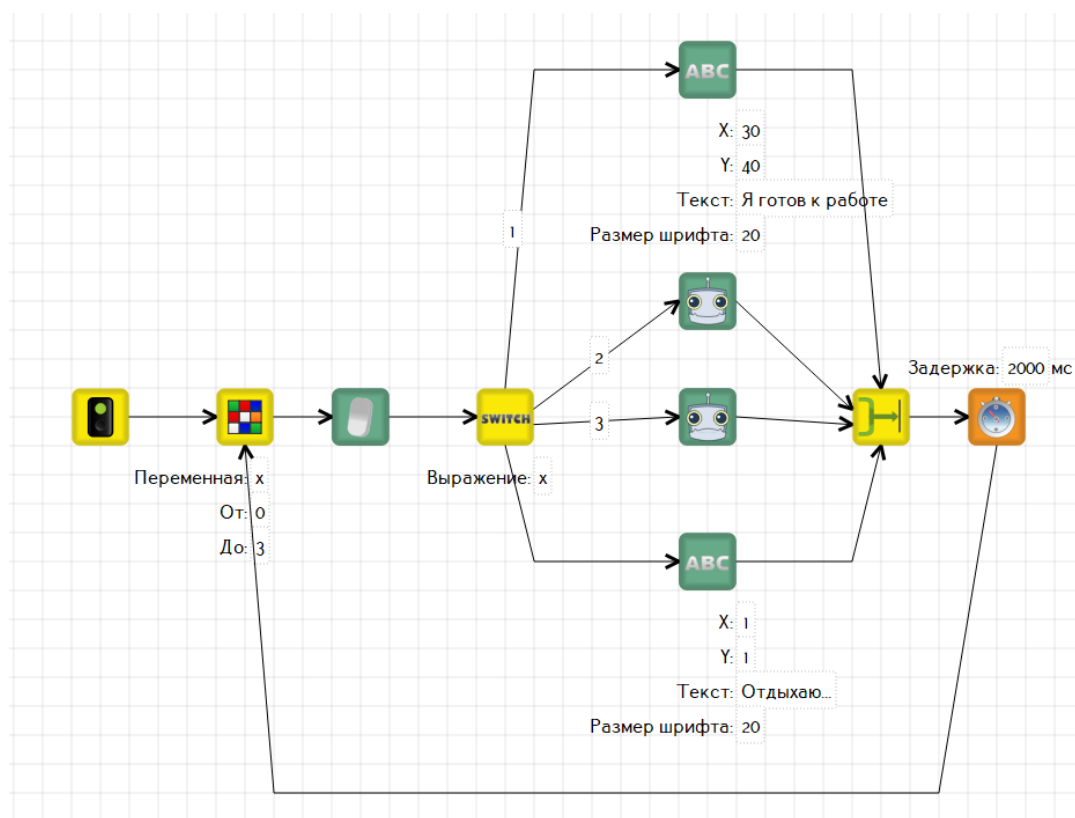
В TRIK Studio реализуется с помощью одноименного блока:



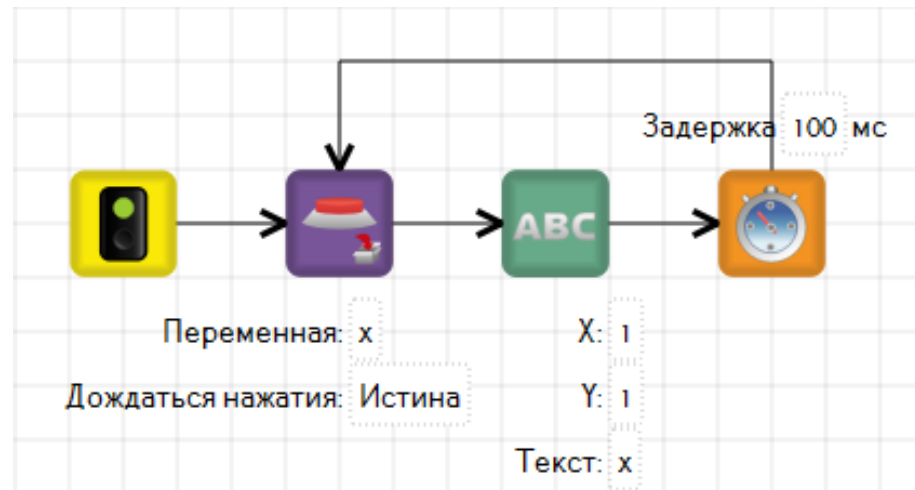
Блок проверяет выражение. От блока отводятся связи, на которых указываются возможные значения этого выражения (например, переменной). Одна связь **обязательно** должна быть пустая (“default”) - по ней алгоритм будет двигаться, если не выполнено ни одно из условий.

Switch

Данный пример демонстрирует случайный выбор одного из четырех состояний робота: «Я готов к работе», «Улыбаюсь», «Грущу», «Отдыхаю...»



Задача 2.1.8. Выводить в цикле с задержкой минимум в 100 мс на экран робота в 2D модели коды кнопок контроллера TRIK, по нажатию на них.



В TRIK Studio имеется блок **«Получить код кнопки»**, который записывает код нажатой кнопки в переменную. Все коды кнопок представлены в кодировке ASCII.

Зная коды кнопок, с помощью **switch** можно написать своё меню.

Задача 2.1.9. (самостоятельно)

Выполнять в цикле действия по нажатию клавиш:

«вверх» (103) — крутить моторами вперед

«вниз» (108) — крутить моторами назад

«влево» (105) — поворачивать влево

«вправо» (106) — поворачивать вправо

«ввод» (28) — улыбаться и говорить «Привет»

«Esc» (1) — выход из программы

Задача 2.1.10. (самостоятельно)

Реализуйте меню из 3х элементов: смайл, сказать, выход. Текущий элемент обведен прямоугольником. С помощью стрелок на контроллер реализовать переход между элементами меню. Выбор реализовать с помощью нажатия на галочку:

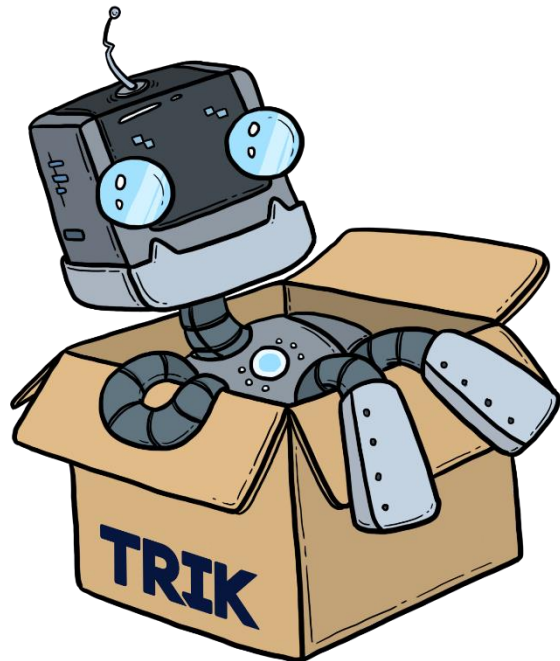
1. Вывести смайл на 3 секунды, вернуться в меню
2. Сказать «Привет», вернуться в меню
3. Завершить выполнение программы



Информация и контакты

TRIK

trikset.com



Поддержка TRIK:
support@trikset.com

Справочный центр TRIK:
help.trikset.com



Распространяется по лицензии
[Creative Commons BY-NC-SA](https://creativecommons.org/licenses/by-nc-sa/4.0/)

ООО «КиберТех»
Санкт-Петербург, 2023