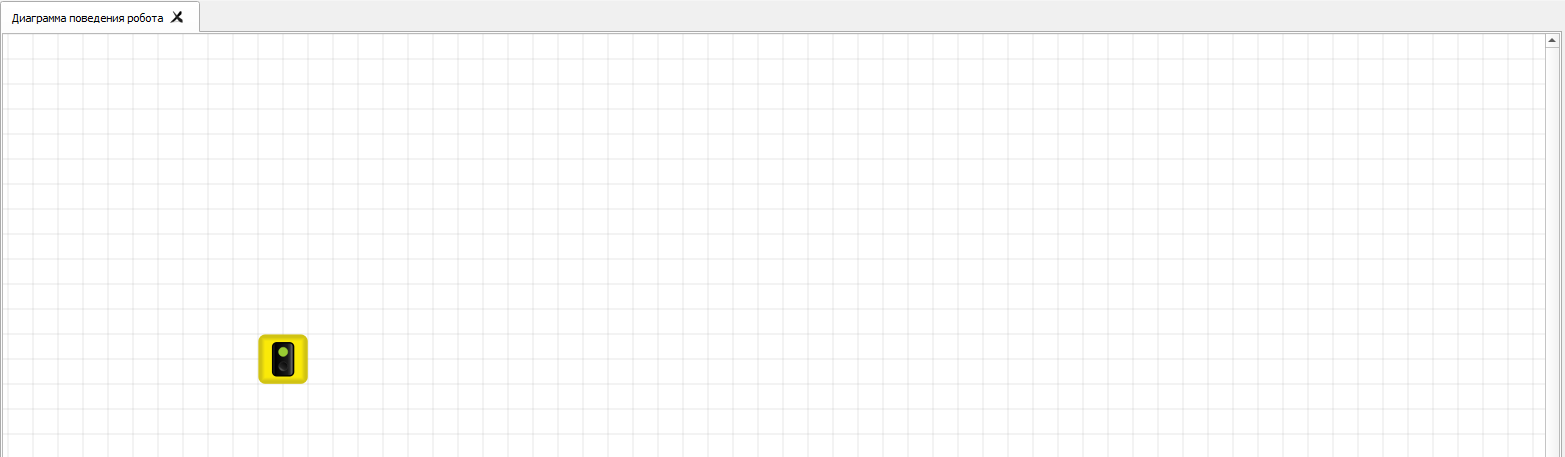
**Визуализированная среда программирования ТРИК и LME. Интерфейс. Режимы. Палитра блоков**

1. **Визуализированная среда программирования ТРИК**

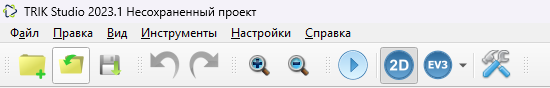
Визуализированная среда программирования ТРИК (Технологии Робототехники и Информатики для Креативного обучения) предназначена для обучения основам программирования и робототехники. Она предоставляет пользователям интуитивно понятный интерфейс и инструменты для создания программ на основе визуального программирования. Давайте рассмотрим основные компоненты и особенности этой среды подробнее.

**Интерфейс**

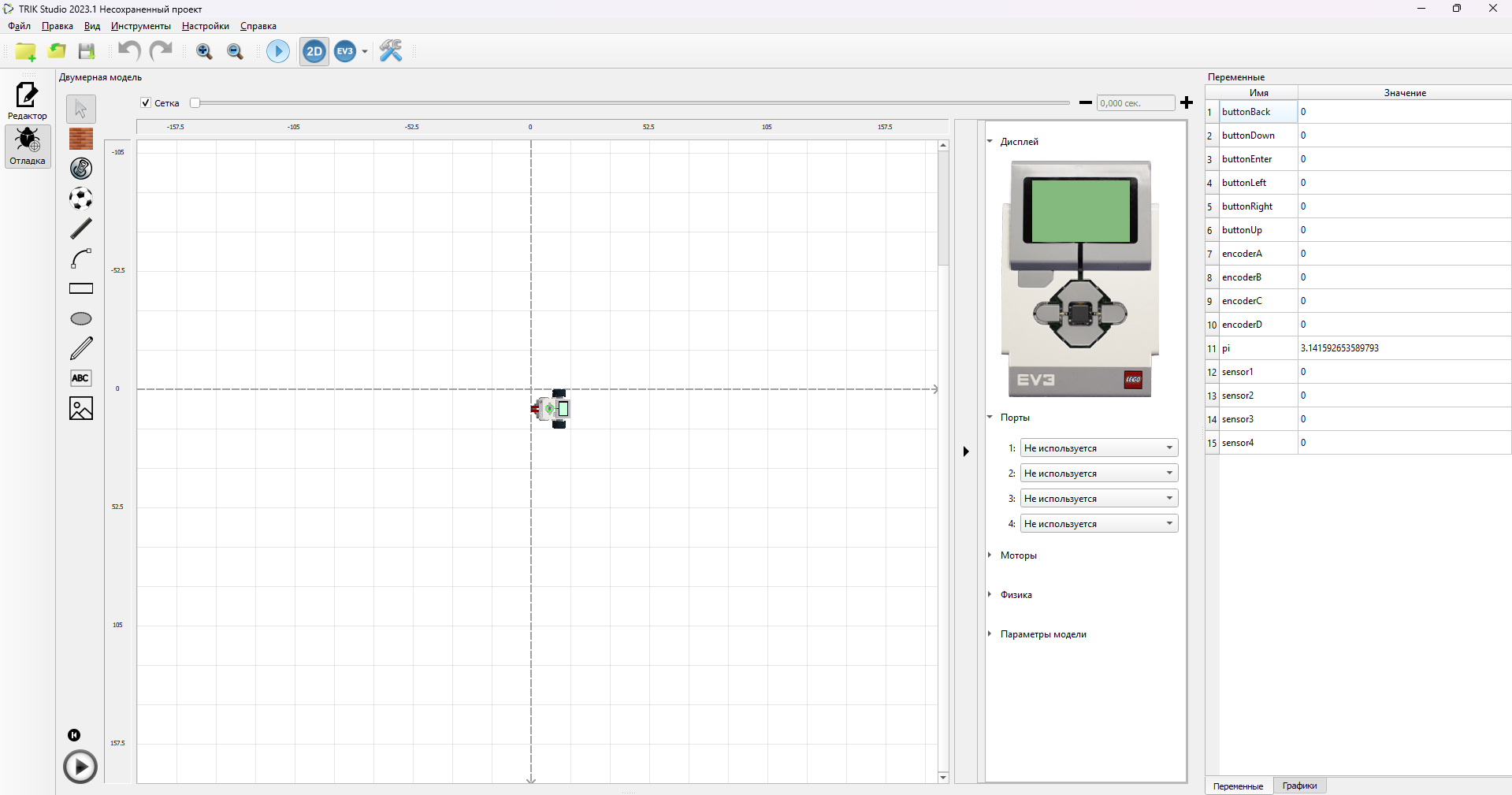
**Главное окно**: Содержит рабочую область для размещения блоков программирования.

****

**Панель инструментов**: Размещена в верхней части и содержит основные функции, такие как создание нового проекта, открытие существующего, сохранение и запуск программы.

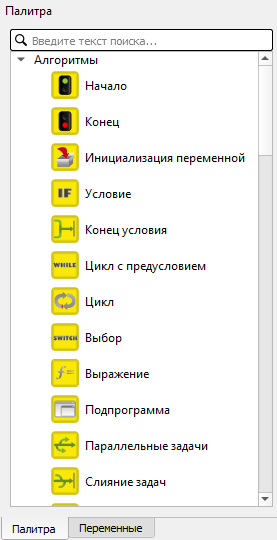
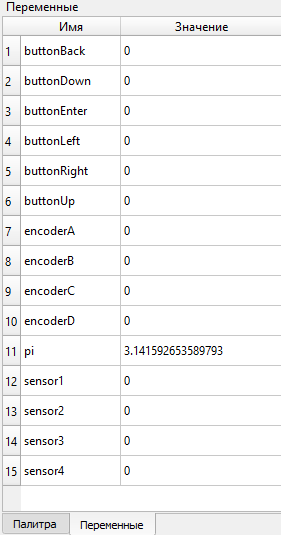


**Режим отладки**: важный инструмент в визуализированной среде программирования ТРИК и LME, который помогает пользователям выявлять и исправлять ошибки в своих программах. Этот режим предоставляет ряд функциональных возможностей, которые делают процесс отладки более эффективным и интуитивно понятным.



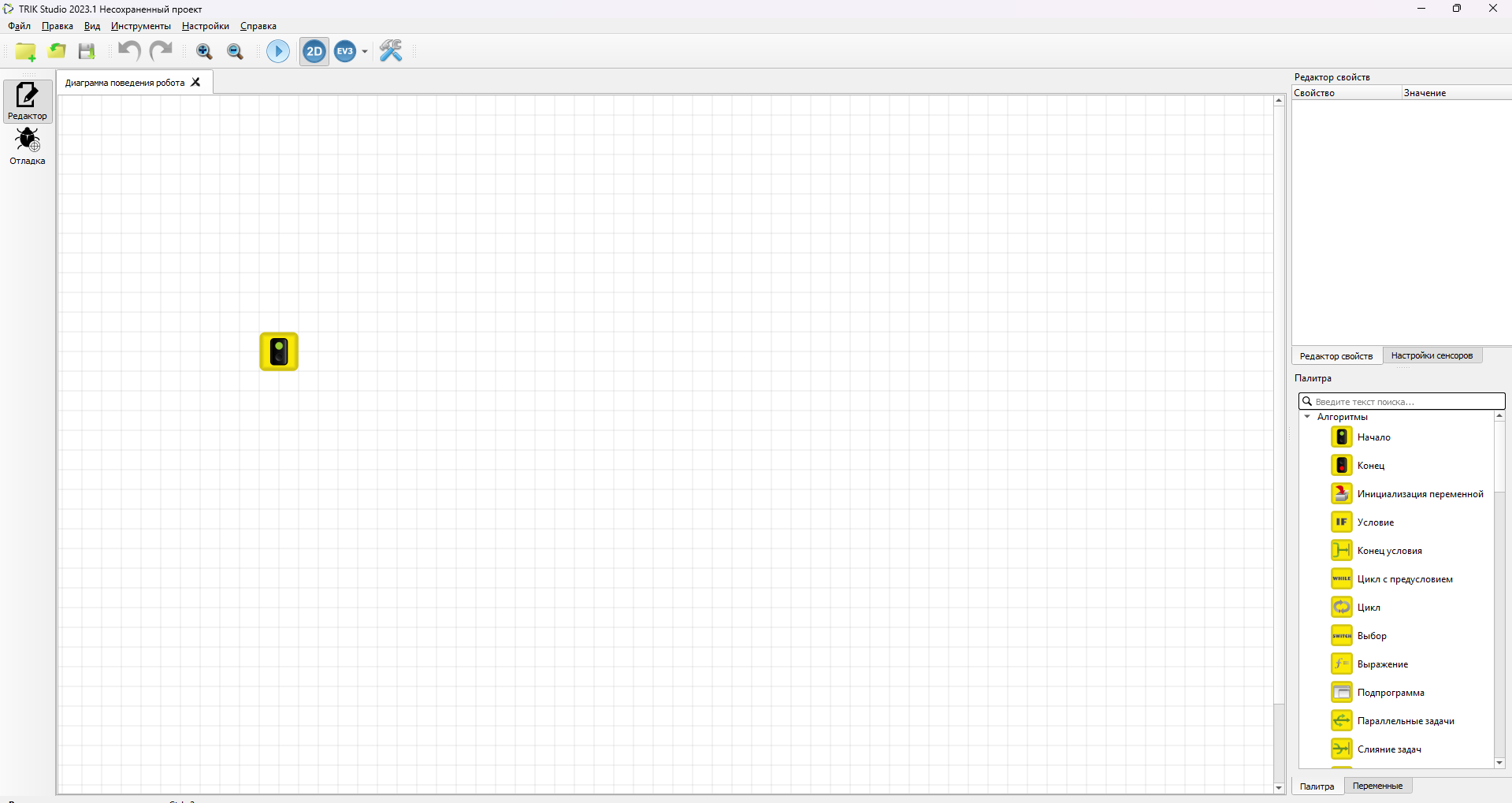
**Панель блоков**: ключевой элемент визуализированной среды программирования, который предоставляет пользователям доступ к различным типам блоков для создания программ. Она организована по категориям, что упрощает поиск нужных элементов.

**Панель переменных** служит для управления данными, которые используются в программе. Она позволяет создавать, изменять и отслеживать значения переменных.

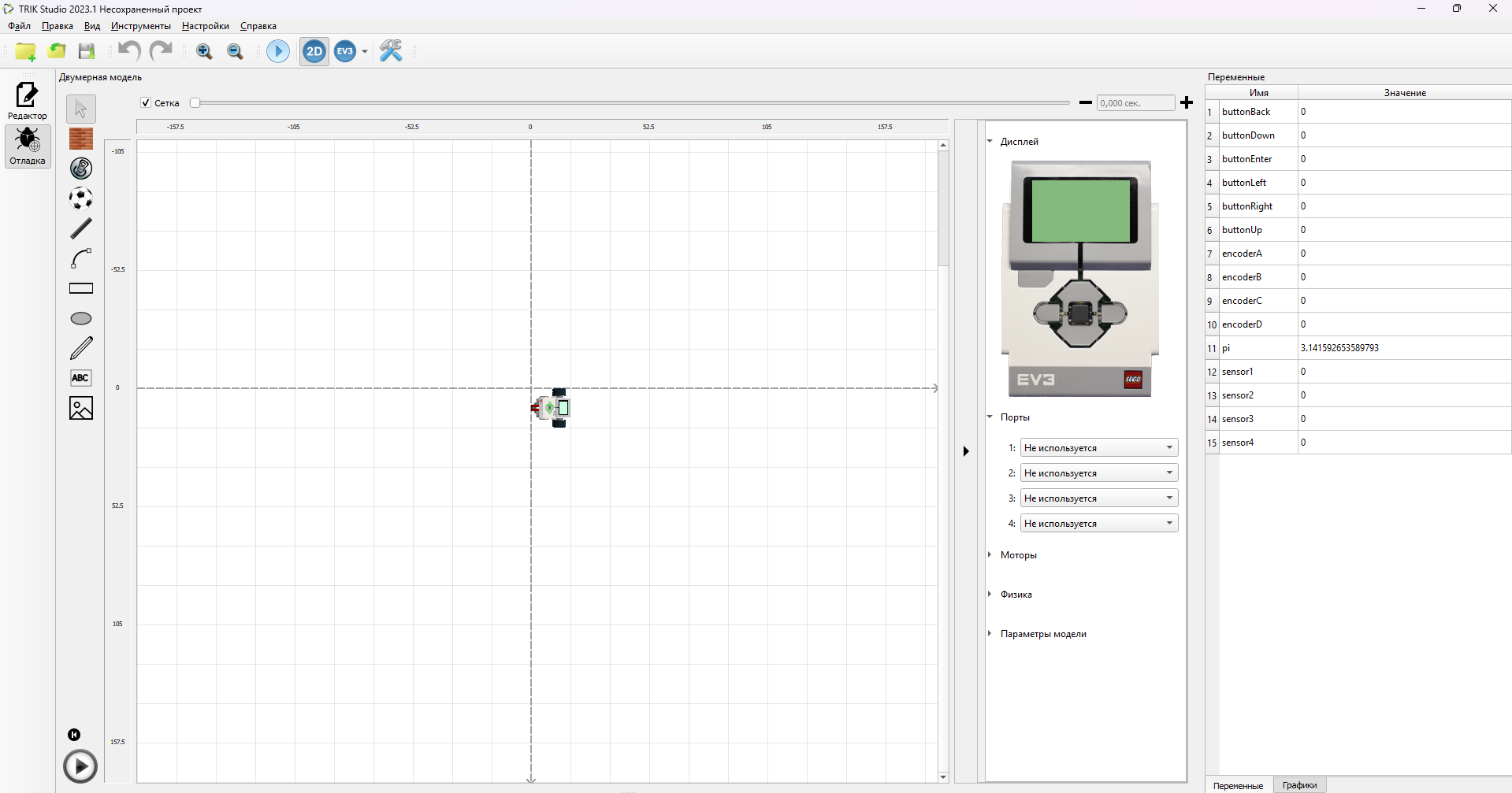


**Режимы**

**Режим программирования**: Позволяет пользователям создавать и редактировать программы, используя блоки визуального программирования.



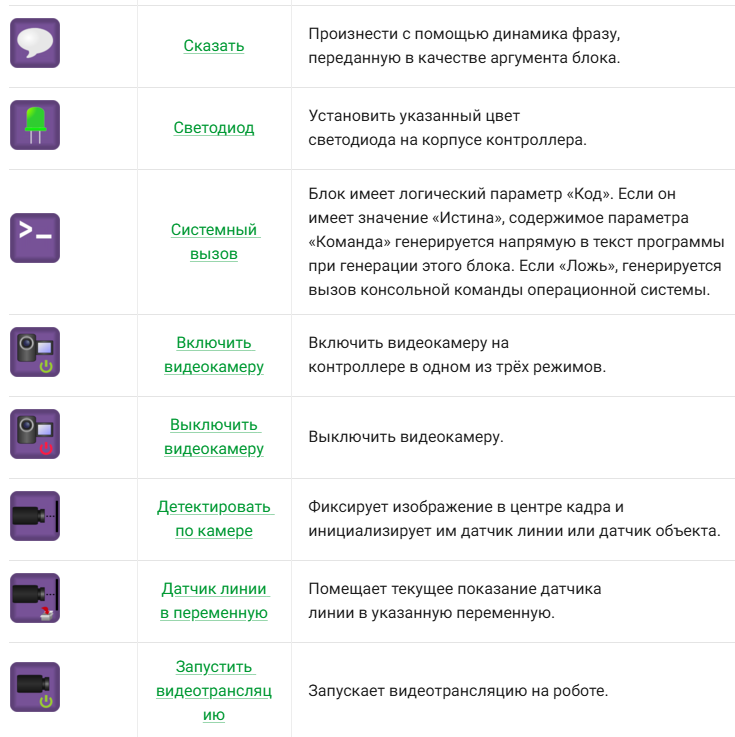
**Режим отладки**: Позволяет выявлять и исправлять ошибки в коде, обеспечивая пошаговое выполнение программы и просмотр значений переменных.

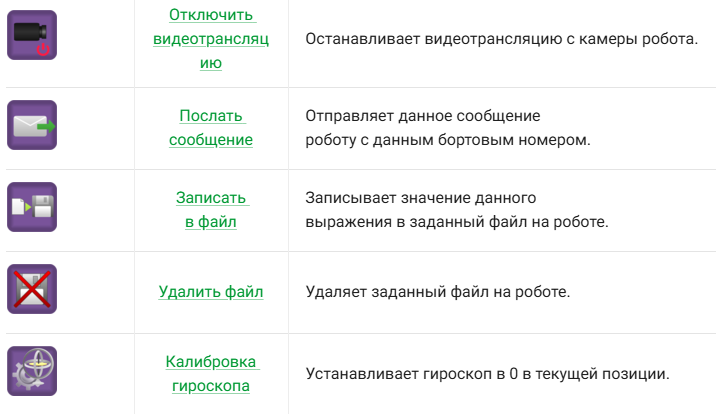


**Палитра блоков**

Блоки действия:

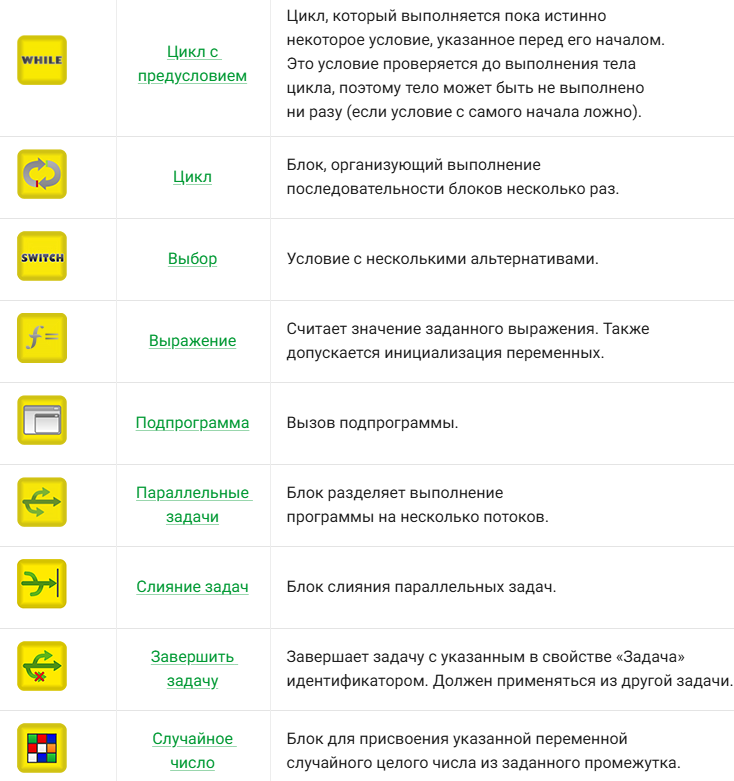
****

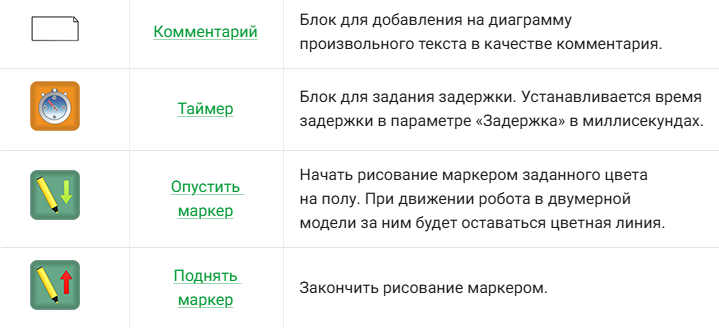
****

****

Все общие блоки

****

****

****

1. **Визуализированная среда программирования LME**

Визуализированная среда программирования LME (Лаборатория Мобильной Энергетики) предназначена для обучения и разработки программ для мобильных и робототехнических систем. Она предлагает интуитивно понятный интерфейс и функционал, позволяющий пользователям легко создавать и тестировать свои проекты.

**Интерфейс**

Главное окно является основным пространством для работы, где пользователи могут собирать свои программы, перетаскивая блоки из палитры в рабочую область. Панель блоков содержит коллекцию блоков, сгруппированных по категориям, таким как управление, действия и сенсоры. Блоки управления включают условные операторы и циклы, блоки действий позволяют выполнять команды, например, движение и управление сенсорами, а блоки сенсоров работают с датчиками и получают данные.

Панель свойств позволяет настраивать параметры выбранного блока, изменяя значения переменных, условия и другие настройки. Строка состояния отображает текущий статус программы, сообщения об ошибках и другую важную информацию о процессе выполнения. Консоль вывода дает возможность видеть вывод данных и сообщений от программы, что полезно для отладки.

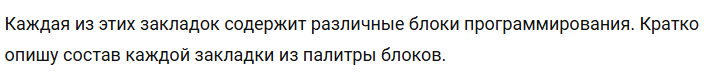
В режиме программирования пользователи могут создавать и редактировать свои программы, используя блоки визуального программирования. Это основной режим, в котором происходит сборка алгоритмов. Режим тестирования позволяет запускать программы на виртуальном моделировании или непосредственно на реальных устройствах, проверяя, как программа взаимодействует с роботами и сенсорами.

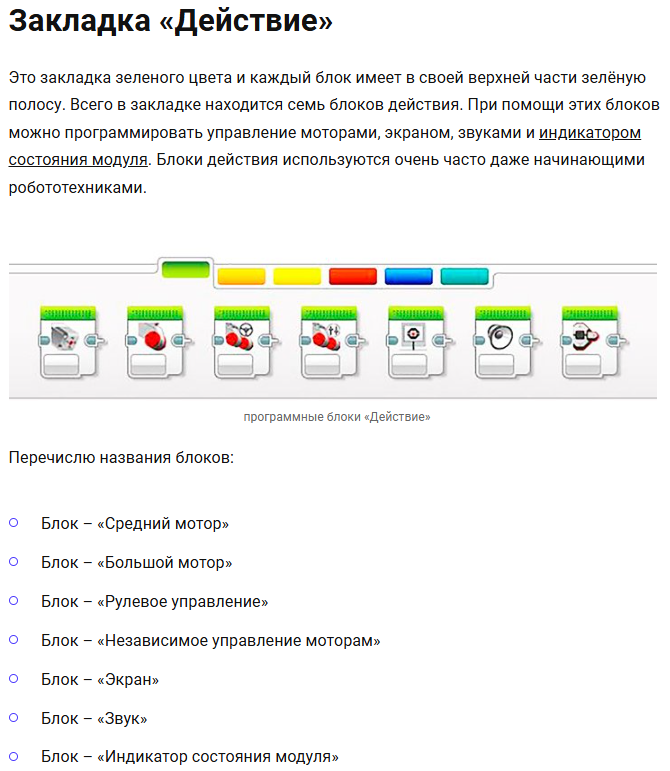
Режим отладки предоставляет возможность анализировать и исправлять ошибки в коде. Пользователи могут выполнять программу пошагово, устанавливать точки останова и отслеживать значения переменных в реальном времени. В режиме симуляции пользователи могут тестировать алгоритмы в виртуальной среде, имитируя поведение робототехнических систем без необходимости использования реальных устройств, что помогает избежать повреждений и ускоряет процесс разработки.

Режим просмотра позволяет пользователям анализировать уже созданные программы, изучая их структуру и логику работы. Это помогает развить навыки программирования и понимание алгоритмов.

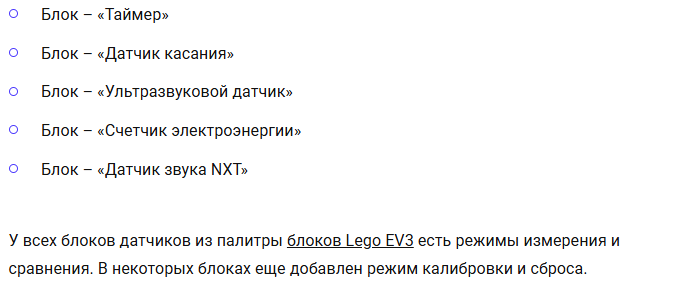
**Палитра блоков:**

****

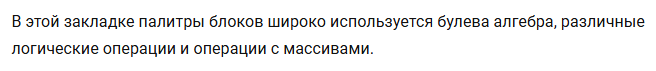
****

****

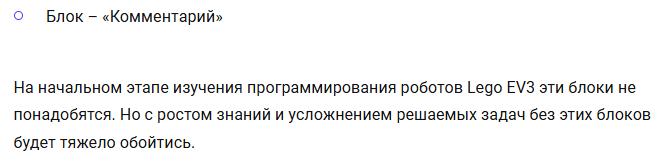
****

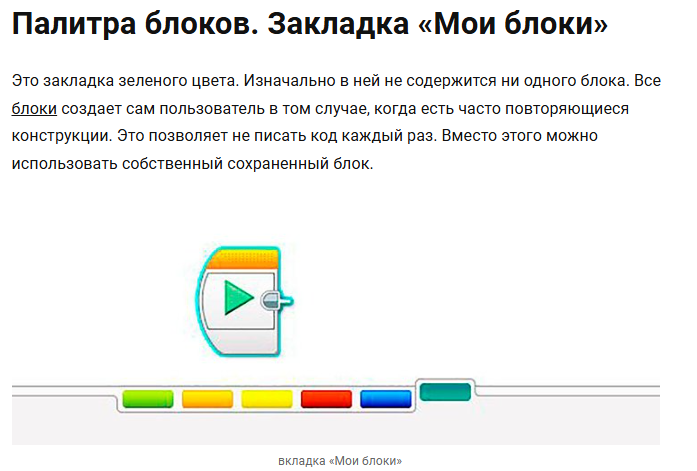
**** ****

****

****

****

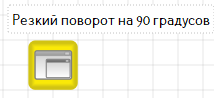
****

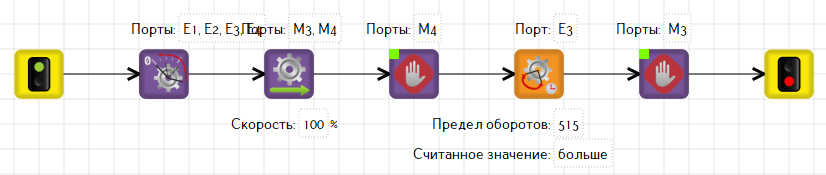
****

**Прохождение известного лабиринта с использованием подпрограмм**

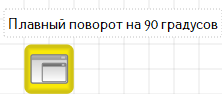
**Подпрограммы**

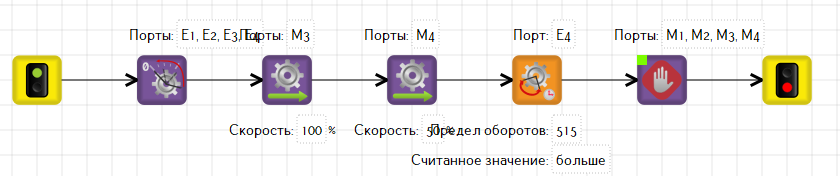
**Резкий поворот на 90 градусов.** Эта подпрограмма предназначена для выполнения резкого поворота робота на 90 градусов. Такой поворот может быть полезен в ситуациях, когда требуется изменить направление движения робота быстро и точно.

****

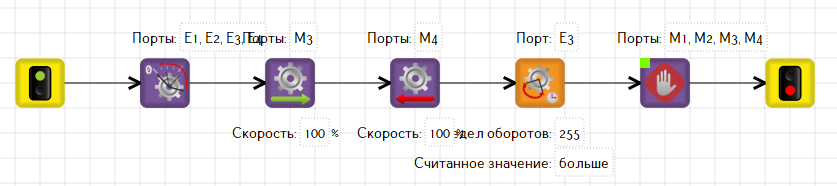
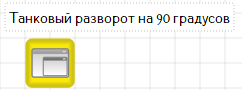
****

**Резкий поворот на 90 градусов.** Эта подпрограмма предназначена для выполнения плавного поворота робота на 90 градусов. Плавный поворот обеспечивает более контролируемое и предсказуемое изменение направления, что особенно важно в узких пространствах или при сложных маневрах

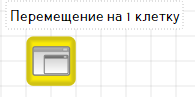
****

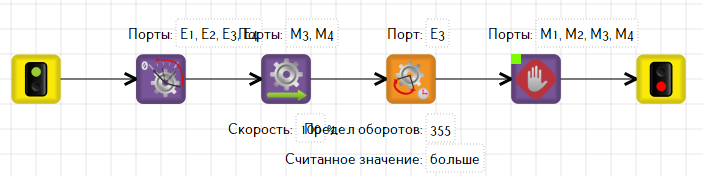
****

**Танковый разворот**. Метод поворота робота, при котором один набор колес (двигателей) движется вперед, а другой — назад. Этот метод позволяет быстро и эффективно изменить направление робота на 90 градусов.

****

**Перемещение на 1 клетку.** Эта подпрограмма предназначена для перемещения робота на одну клетку в заданном направлении. Она может быть использована в различных задачах, таких как навигация по сетке или лабиринту.

****

****

**Алгоритм**

Алгоритм прохождения известного лабиринта с использованием подпрограмм включает несколько этапов и действий, которые обеспечивают навигацию робота в ограниченном пространстве.

Инициализация: Запустите все необходимые сенсоры и подготовьте системы к работе. Убедитесь, что все подпрограммы готовы к выполнению.

Начните цикл прохождения лабиринта. Пока робот не достигнет цели (выхода из лабиринта), выполните следующие шаги:

Запустите подпрограмму "Обнаружение препятствий". Используйте сенсоры для проверки наличия препятствий впереди. Если препятствие обнаружено, перейдите к следующему шагу.

Запустите подпрограмму "Корректировка пути". Определите новое направление, используя логику обхода. Это может быть основано на правилах, например, "правила правой руки".

В зависимости от нового направления выполните повороты, используя подпрограммы для резких или плавных поворотов, или танкового разворота.

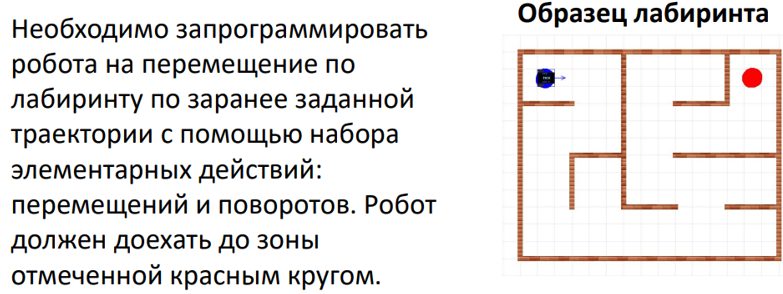
Если препятствий нет, запустите подпрограмму "Перемещение на 1 клетку". Двигайтесь в заданном направлении, перемещая робота на одну клетку вперед.

На каждом шаге проверяйте состояние сенсоров, чтобы убедиться в корректности движения и отсутствии неожиданных препятствий.

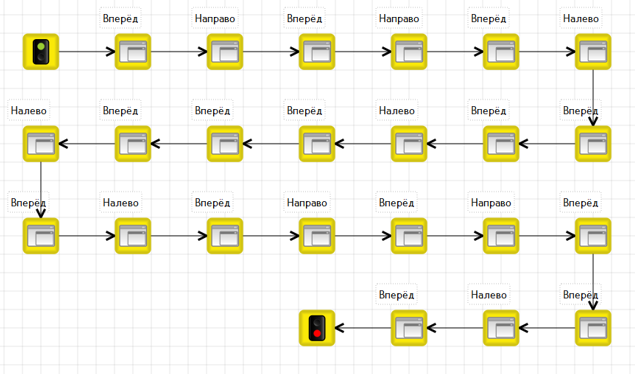
После достижения цели остановите движение и завершите программу.

Этот алгоритм обеспечивает эффективное и безопасное прохождение лабиринта, используя заранее определенные подпрограммы для маневрирования и навигации.

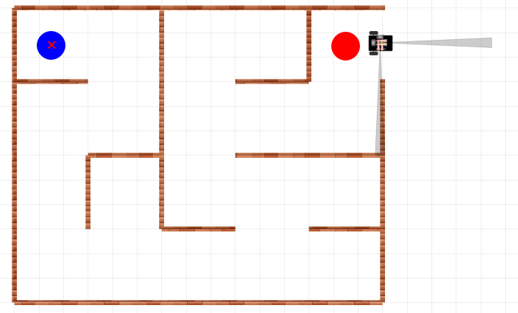
**Пример передвижения по лабиринту:**

****

Составляем программу из подпрограмм:

****

Проверим работоспособность

****

**«Нормализации» датчика цвета**

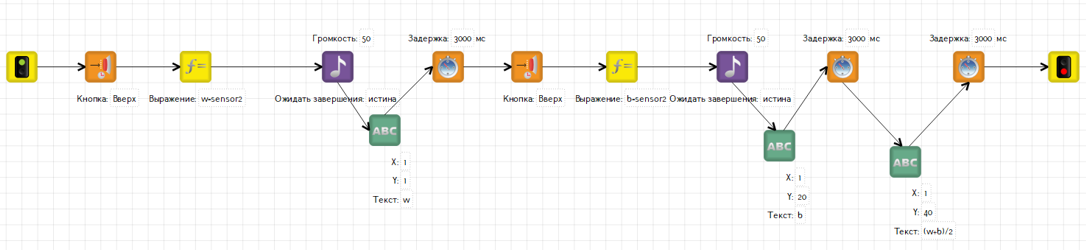
Программа "Нормализация" датчика цвета предназначена для определения среднего значения серого цвета с использованием датчика на роботе.

Процесс начинается с запуска программы на роботе, что подготавливает систему к работе с датчиком цвета. После этого вы устанавливаете датчик на белый цвет. Этот шаг необходим для получения значения отражаемого света от белого объекта. Нажимаете на кнопку, чтобы зафиксировать измерение. Программа считывает значение отраженного света и сохраняет его как первое значение.

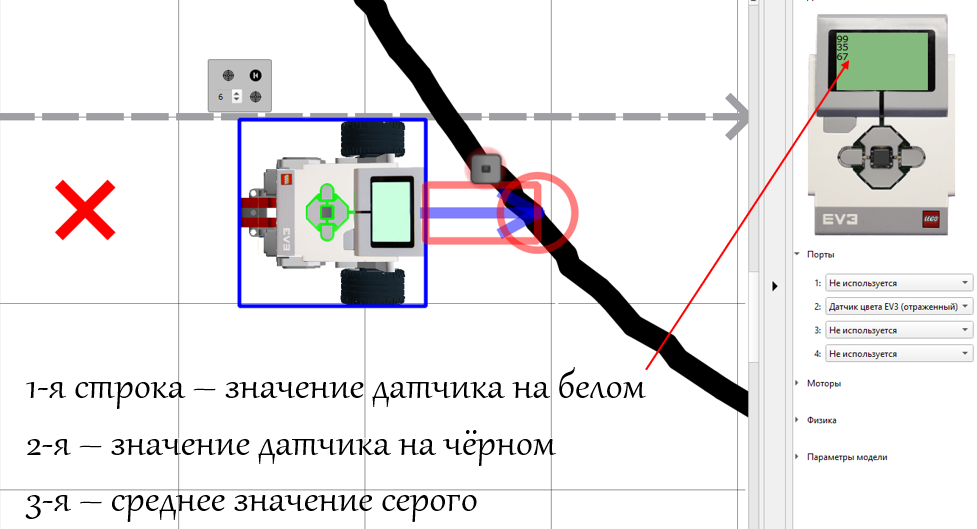
Далее вы повторяете процесс, устанавливая датчик на черный цвет. Это значение отражает минимальное количество света, которое может быть зарегистрировано датчиком. Снова нажимаете на кнопку, чтобы зафиксировать измерение. Программа сохраняет это значение как второе.

После того как оба значения зафиксированы, программа выполняет вычисления. Она складывает два полученных значения — значение на белом цвете и значение на черном цвете. Затем программа делит сумму на два, чтобы получить среднее значение серого цвета.

В завершение программа выводит рассчитанное среднее значение серого цвета. Это значение может быть использовано для дальнейших действий или анализа в задачах, связанных с распознаванием цвета и робототехникой.



Реализация программы в режиме отладки:

****