

Е. В. Волкова
И. И. Мацаль

Основы программирования в среде **VEXcode IQ**

Учебно-методическое пособие



МОСКВА
2021

УДК 372.8(072)

ББК 74.202.5

В67

Волкова Е. В.

В67 Основы программирования в среде VEXcode IQ: учебно-методическое пособие / Е. В. Волкова, И. И. Мацаль. — М. : Издательство «Экзамен», 2021. — 64 с.

ISBN 978-5-377-16443-2

Данное методическое пособие знакомит с графической средой программирования VEXcode IQ blocks. VEX IQ – это робототехнический модуль для учащихся начальных классов. Структурные части VEX IQ соединяются и разъединяются без использования инструментов, что облегчает работу учащихся, дает возможность быстро собирать и модифицировать робота. VEX IQ включает в себя датчики света, гироскоп, датчик расстояния, которые помогут не только создать больше моделей, но и сконструировать более интересных роботов, которые дают больше возможностей для обучения.

Пособие предназначено для преподавателей и родителей учащихся начальных классов.

УДК 372.8(072)

ББК 74.202.5

Подписано в печать с диапозитивов 14.01.2021.

Формат 60х90/8. Гарнитура «Calibri». Бумага офсетная.

Усл. печ. л. 8. Тираж 1000 экз. Заказ №

ISBN 978-5-377-16443-2

© Волкова Е. В., Мацаль И. И., 2021

© Издательство «**ЭКЗАМЕН**», 2021

© «**ЭКЗАМЕН-ТЕХНОЛАБ**», 2021

Содержание

Урок 1. Основные параметры установки и команды	стр. 5
Урок 2. Готовые проекты	стр. 15
Урок 3. Стандартные маневры: как двигаться на заданное расстояние	стр. 21
Урок 4. Готовые проекты: изменяем скорость движения	стр. 27
Урок 5. Готовые проекты: ускорение	стр. 31
Урок 6. Готовые проекты: выводим значение датчиков на экран	стр. 37
Урок 7. Готовые проекты: распознавание цветов	стр. 43
Урок 8. Готовые проекты: распознавание объектов	стр. 49
Урок 9. Собственные проекты: вертушка Ньютона	стр. 57
Урок 10. Собственные проекты: театр теней	стр. 61



Урок 1.

Основные параметры установки и команды

1. Что такое визуальное (графическое) программирование?
2. Как установить и откуда?
3. Основные блоки команд
4. Первая программа
5. Подключение робота
6. Запуск программы
7. Инструкция по сборке

Языки программирования появились еще в 19 веке. Работа с вычислительными машинами с помощью языков программирования требовала специальных сложных умений. Одновременно с этим вычислительные системы становились сложнее, и к моменту, когда появились первые компьютеры, языков программирования было много, они были разными и требовали долгого обучения работе с ними.

Проекты с использованием цифровых технологий разрастались – команды разработчиков состояли не только из программистов, но и ученых самых разных областей, каждый из которых разбирался только в своей отрасли. Проблема была в том, что судьба проекта сильно зависела от того, как точно все участники проекта поймут друг друга и смогут ли предложить наиболее эффективные решения. Ученые понимали, что необходимо средство, которое даст безусловное единое понимание алгоритмов для всех участников любого проекта. Одним из таких средств стали визуальные языки программирования.

Визуальный язык программирования позволяет вместо написания текста программы использовать графические объекты. Ранние визуальные языки были похожи на блок-схемы. Одним из ярких примеров можно назвать язык ДРАКОН, активно использовавшийся в 80-е годы 20 века для разработки системы управления космического корабля «Буран». Другой пример – более современный LabVIEW, предназначенный для систем сбора и обработки данных.

Сегодня существует много графических сред для обучения программированию в начальной и средней школах. Визуальные языки программирования позволили снизить возраст детей, начинающих заниматься программированием. Программирование роботов стало популярным благодаря упрощенности работы с визуальными языками и наглядности потоков данных внутри программы. Безусловно, упрощенность языка играет и отрицательную роль, например в возможностях создания сложных устройств, но для развития устойчивого интереса к программированию этого не требуется.

Данное методическое пособие предлагает познакомиться с графической средой программирования VEXcode IQ blocks.

VEXcode IQ blocks старается совместить в себе плюсы графического программирования (наглядность потоков данных) и текстового программирования (все основные операторы управления, различные виды циклов, работа с булевой логикой и массивами, обработка событий, сопровождение команд текстом на английском языке).

Для того чтобы установить среду программирования, необходимо скачать установочный пакет с официального сайта (Рисунок 1).

<https://www.vexrobotics.com/vexcode-download> – адрес в Интернете, где можно скачать программное обеспечение VEXcode IQ blocks.



Рисунок 1. Страница скачивания среды программирования VEXcode IQ Blocks

Далее при запуске установочного файла остальной процесс происходит автоматически и не требует особенных умений.






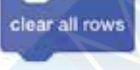











При запуске программы сразу открывается новый файл для создания программы (Рисунок 2).















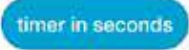

















Рисунок 2. Файл для создания программы в среде VEXcode IQ Blocks




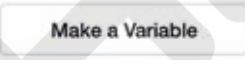






Все основные команды разделены на 9 категорий, подробное описание которых представлено в таблице 1.


Таблица 1. Описание блоков среды программирования VEXcode IQ Blocks

Раздел	Название	Команда	Описание
 Looks	Вывод на экран	Looks 	Вывод на экран
			Установить курсор на строку _ и столбец _
			Установить курсор на следующую строку
			Установить точность вывода
			Очистить строки
			Очистить строку _
 Sound	Звуки	Sound 	Воспроизвести звук
			Воспроизвести ноту
 Events	События	Events 	Начало программы
			Когда кнопка блока нажата/отпущена
			Когда таймер больше _ секунд
			Когда получено сообщение
			Транслировать сообщение
			Транслировать сообщение и ждать

Раздел	Название	Команда	Описание
	Управление	Control 	Ждать заданное количество секунд
			Повторять заданное количество раз
			Повторять бесконечно
			Если <условие>, то ...
			Если <условие>, то ... , иначе ...
			Ждать до тех пор, пока не произойдет заданное условие
			Повторять до тех пор, пока не произойдет заданное условие
			Пока <условие> верно, повторять
			Прерывание цикла

Раздел	Название	Команда	Описание
 Sensing	Измерение	Sensing 	Сбросить таймер
			Установить таймер в секундах
			Столбец курсора
			Строка курсора
			Нажата ли кнопка?
			Процент заряда батареи
 Operators	Операторы	Operators    	Операторы сложения, вычитания, умножения и деления
			Генерация случайного числа в заданном диапазоне
		  	Операторы сравнения: больше, меньше, равно
		  	Логические операции: И, ИЛИ, НЕ
			Округление

Раздел	Название	Команда	Описание
			Функции: - получение модуля, - округление вниз, - округление вверх, - квадратный корень, - синус, - косинус, - тангенс, - котангенс, - арксинус, - арккосинус, - арктангенс, - десятичный логарифм, - логарифм, - e^x , - 10^x
			Остаток от деления
 Variables	Переменные	Variables  	Создать переменную
			Установить значение переменной
			Изменить значение переменной
			Создать переменную логического типа
			Создать список
			Создать двухмерный список

Раздел	Название	Команда	Описание
 My Blocks	Мои блоки	My Blocks 	Создать собственную функцию
 Comments	Комментарии	Comments 	Создать комментарий

Блоки других функциональных групп (например, блоки **Движения (Motion)**) будут подробно рассмотрены в других занятиях. Они появляются при подключении приводов и других устройств.

Составим первую программу вывода на экран приветствия и воспроизведения звука. Для этого после блока **When Started** вытащим и присоединим команду **Print** со значением «Привет!» (Рисунок 3).



Рисунок 3. Вывод на экран слова «Привет!» с помощью команды **Print**

Затем выберем команду **Play Sound** и откроем выпадающий список (Рисунок 4).

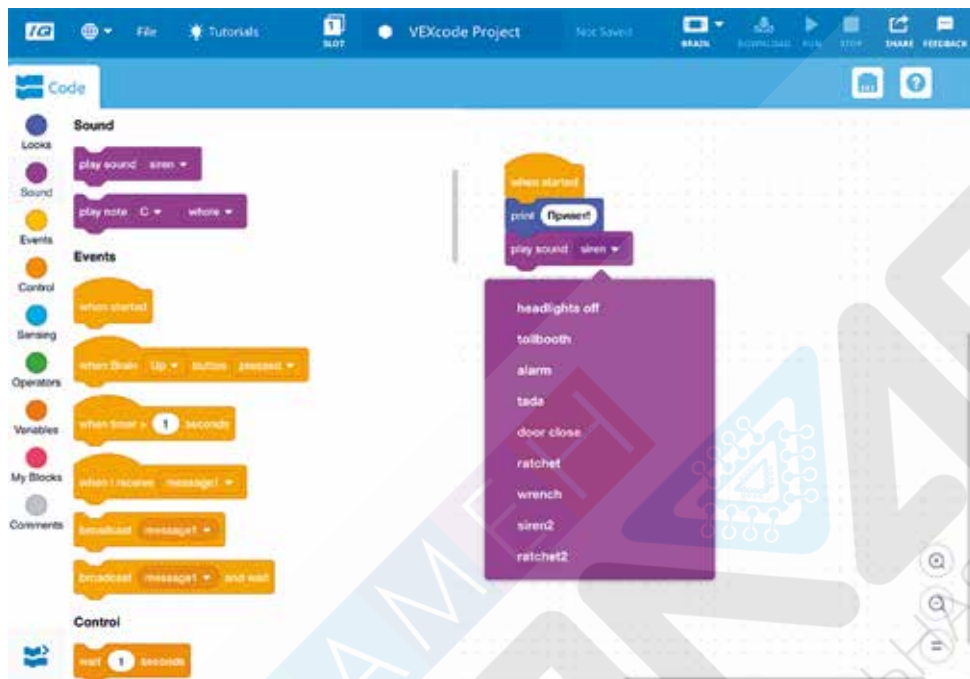


Рисунок 4. Выбор звука для воспроизведения

И после этого определимся, какой звук больше подходит, например «**tada**» (Рисунок 5).

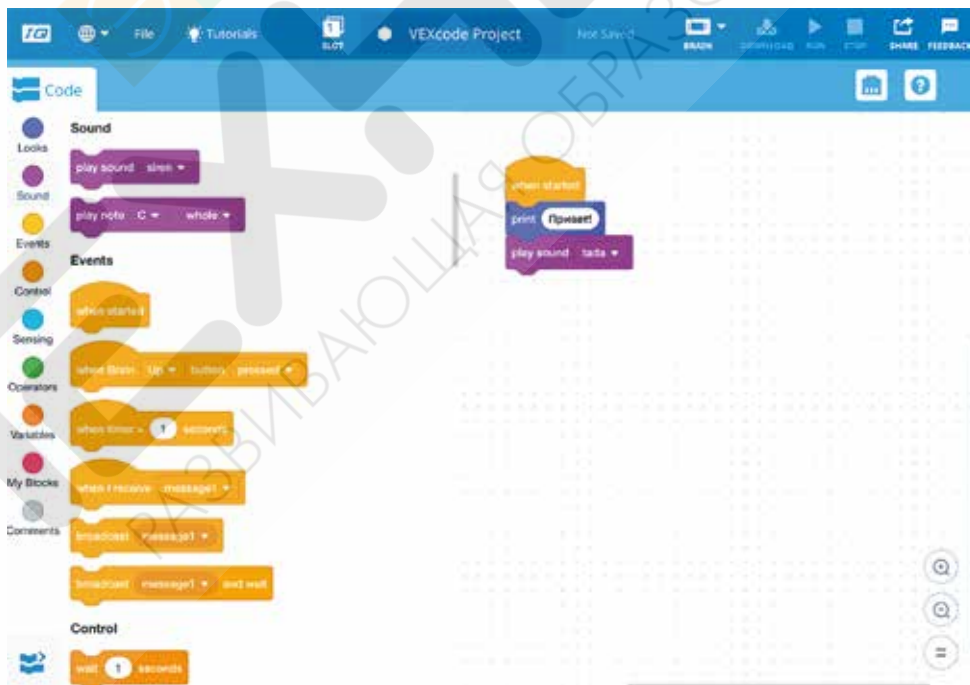


Рисунок 5. Выбор звука «**tada**» для воспроизведения

Программа может быть запущена после подключения блока к компьютеру или планшету. Подробные инструкции по подключению представлены в приложении к данному занятию. После того как блок VEX IQ подключен, значок **BRAIN** будет гореть зеленым цветом (Рисунок 6). После этого достаточно нажать кнопку **RUN** и программа будет запущена.



Рисунок 6. Индикатор подключения блока VEX IQ

Программу можно также загрузить в блок VEX IQ. Для этого достаточно нажать кнопку **DOWNLOAD**. После этого блок можно будет отсоединить от компьютера или планшета и использовать автономно.



Урок 2. Готовые проекты

Знакомство со средой программирования VEXcode IQ Blocks можно продолжить, используя **Tutorials** и **Examples** (Рисунок 7).

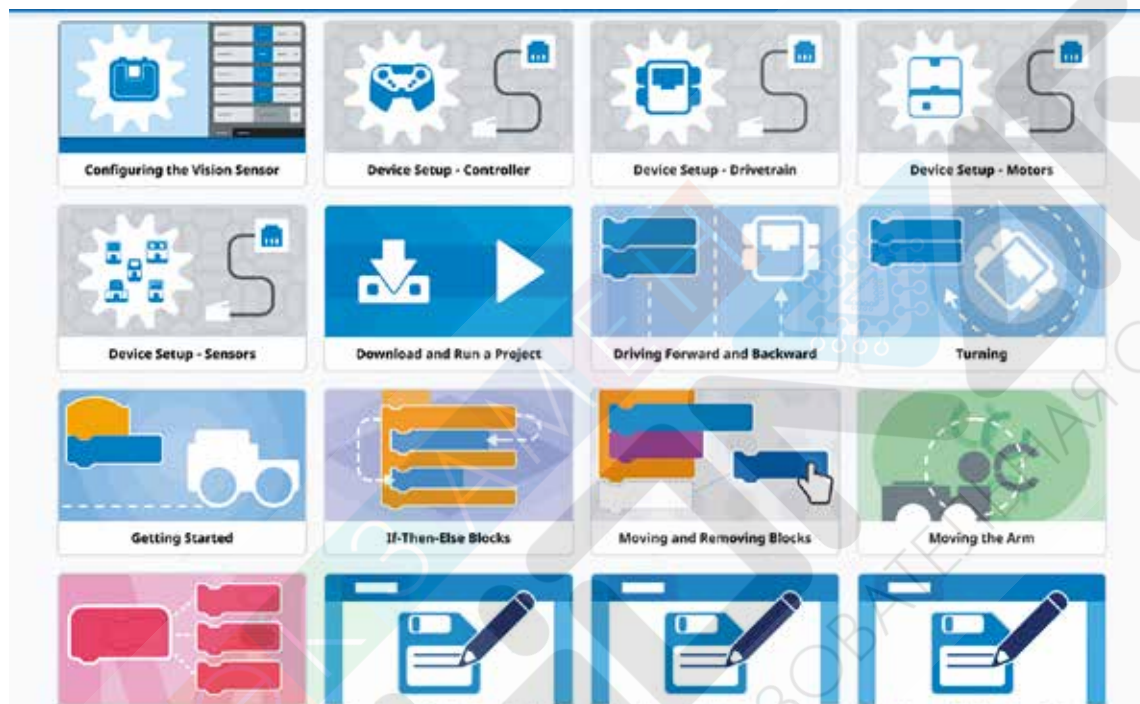


Рисунок 7. Главный вид меню обучающих материалов

Материалы по использованию среды программирования вынесены на главную панель управления. Категории обучающих материалов приведены в таблице 2:

Таблица 2. Категории обучающих материалов

Название категории	Описание категории
Device Setup	Установка датчиков, приводов и настройка блоков
Download and Run a project	Загрузка и запуск программы
Driving	Маневрирование с помощью робота
Managing	Операторы управления, перестановка команд и удаление
My Blocks	Создание собственных функций

Название категории	Описание категории
Naming and Saving	Сохранение программ на разных типах устройств с разными операционными системами
Settings	Настройки сохранения данных, настройки датчиков, обновление ПО робота
Using	Использование примеров и шаблонов, событий, подсказок и циклов
Wireless Downloading	Беспроводное подключение робота

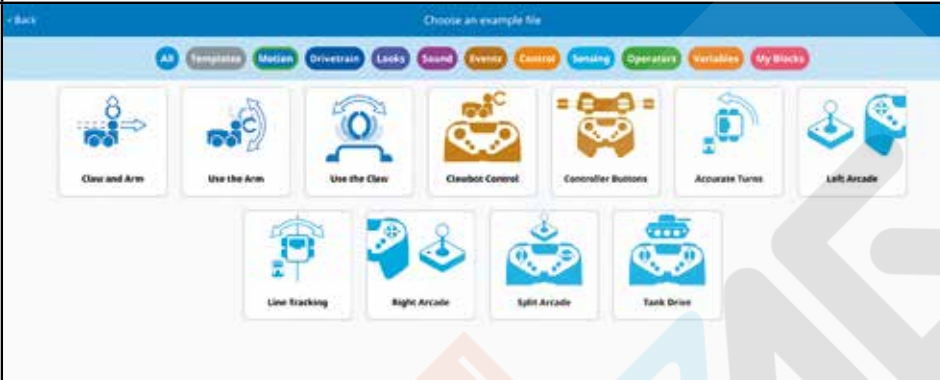

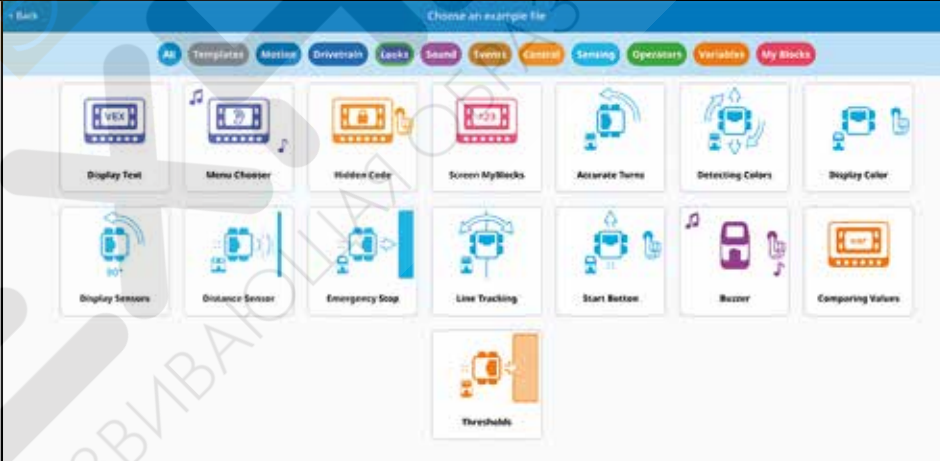

Все обучающие материалы (**Tutorials**) выполнены в форме видео с подробным описанием действий, что не вызывает проблем в использовании, несмотря на то что видео на английском языке (Рисунок 8).









Рисунок 8. Пример видео из раздела обучающих материалов

В меню **File** также можно найти **Examples**. Ниже в таблице 3 представлены категории, на которые разделены все готовые проекты.

Таблица 3. Категории готовых проектов

Раздел	Проекты
Motion (Движение)	
Drivetrain (Привод)	
Looks (Вывод на экран)	
Sound (Звуки)	

Раздел	Проекты
Events (События)	
Control (Управление)	
Sensing (Измерение)	

Раздел	Проекты
Operators (Операторы)	
Variables (Переменные)	
My Blocks (Мои функции)	

Примеры проектов для конструктора VEX IQ содержат готовый код. Проект необходимо открыть и запустить уже готовую программу (Рисунок 9).



Рисунок 9. Пример программы из готового проекта

Раздел **Examples** содержит в себе и шаблоны – **Templates** (Рисунок 10).

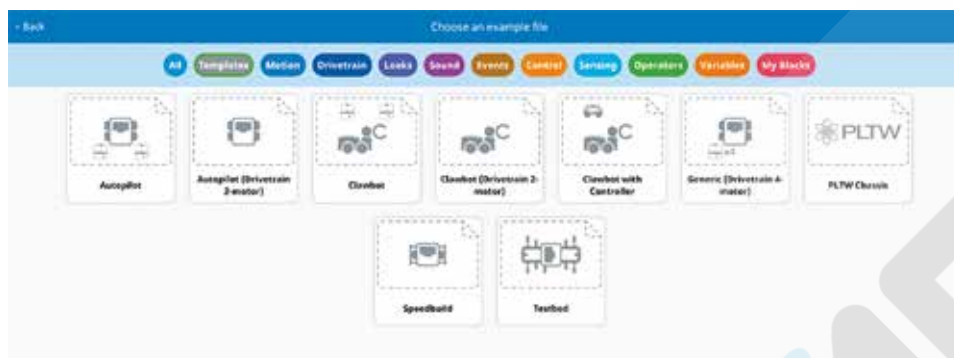


Рисунок 10. Вид главного меню раздела с шаблонами

Шаблоны можно использовать для создания собственных проектов. В них уже подключены все необходимые устройства, например приводы (Рисунок 11).

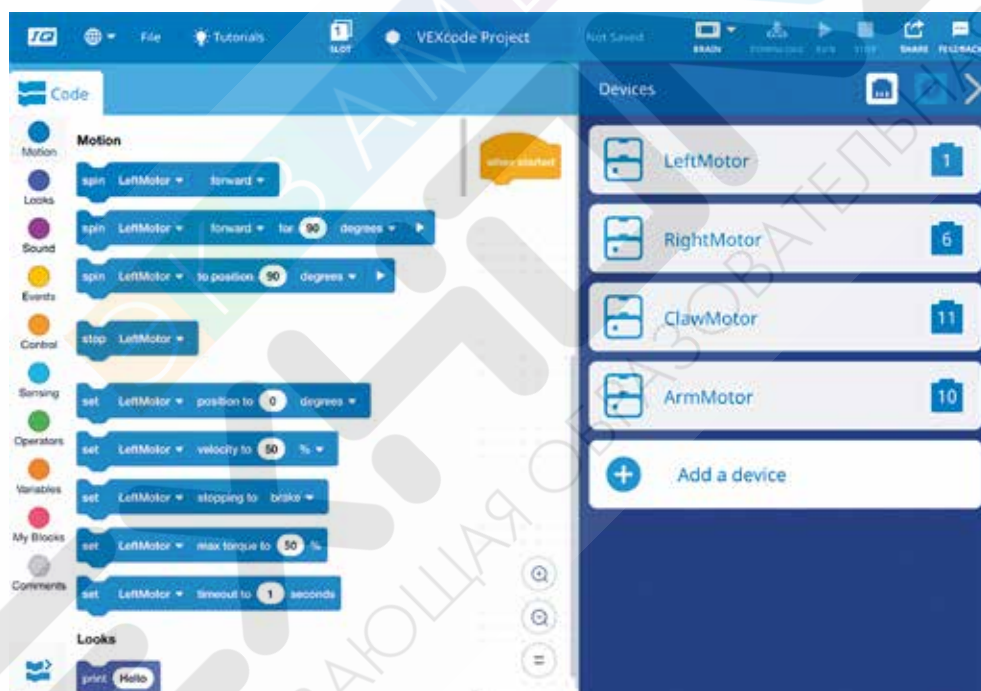


Рисунок 11. Пример подключения устройств в готовом проекте

Следующие занятия будут посвящены методическому сопровождению проектов из раздела **Examples**.

Урок 3.

Стандартные маневры: как двигаться на заданное расстояние

1. Что такое маневрирование?
2. Как подключить трансмиссию?
3. Зачем задавать параметры трансмиссии? Встроенные команды управления колесной базой.
5. Чем отличается обычная программа с маневрами от использования встроенных команд?

Существует набор определенных умений работы с роботами или другими автоматизированными устройствами, принцип работы которых не зависит от используемой платформы. Например, маневрирование, опрос датчиков, работа с такими алгоритмами, как движение по линии и поиск пути.

Остановимся подробнее на маневрах. Любая работа по программированию робота начинается с освоения простых маневров – движение вперед и назад, повороты, движение на заданное расстояние.

VEXcode IQ Blocks предоставляет возможность сделать работу с маневрами более простой для понимания начинающих.


В начале работы над программой необходимо выбрать кнопку  подключения дополнительных устройств в правом верхнем углу экрана (Рисунок 12).



Рисунок 12. Меню подключения дополнительных устройств

Затем выбрать **Add a device** и остановиться на варианте



Drivetrain

(2-motor). Существует возможность работы и с полноприводной трансмиссией, однако для начала остановимся на переднеприводной (Рисунок 13).

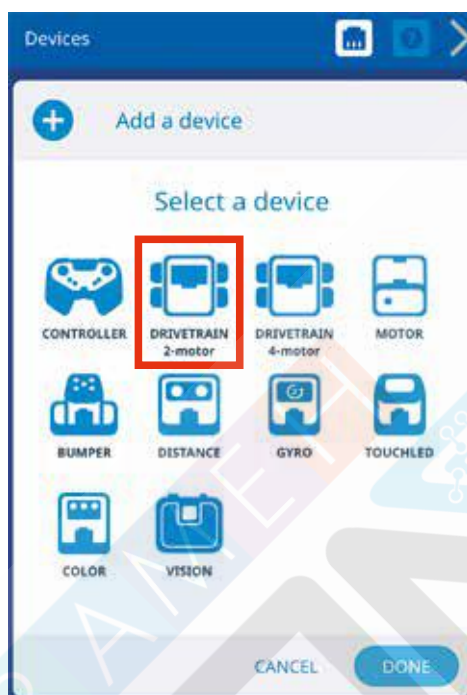


Рисунок 13. Подключение переднеприводной трансмиссии

Далее в меню добавления устройства (**Add a device**) вам будет предложено выбрать порты для приводов (Рисунок 14, а) и определиться с тем, используется ли гироскопический датчик (Рисунок 14, б). Для простых программ с маневрами его подключение не требуется. Более того, это даст обучающимся понимание таких проблем, как неодинаковая работа приводов, где даже внутренняя синхронизация, предусмотренная разработчиками, не может предотвратить некоторые сбои.



Рисунок 14, а. Выбор портов подключения приводов

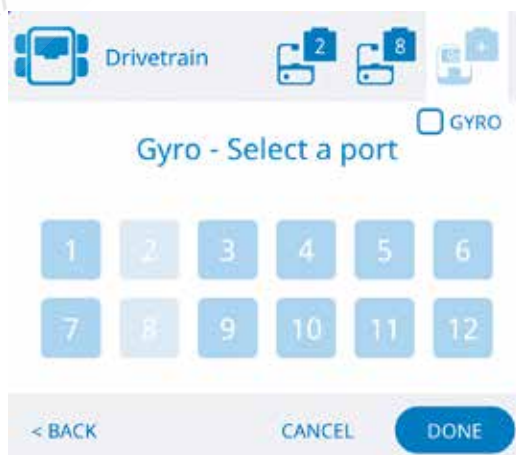


Рисунок 14, б. Выбор порта для подключения гироскопического датчика

Интересным дополнением стала возможность задать параметры колесной базы и размера колес, а также передаточного отношения. **Wheel size** отвечает за длину окружности колес. Ее можно выбрать из нескольких вариантов – 100 мм, 160 мм, 200 мм, 250 мм. **Track Width** – ширина трансмиссии (Рисунок 15).



Рисунок 15. Ширина трансмиссии


Знак вопроса  всегда подскажет, какой параметр необходимо измерить, чтобы записать в каждое поле. **Wheelbase** – расстояние между колесами (длина трансмиссии на рисунке 16).



Рисунок 16. Длина трансмиссии

Gear Ratio отвечает за передаточное число, где можно указать входное значение (**Input**) и выходное значение (**Output**) (Рисунок 17).

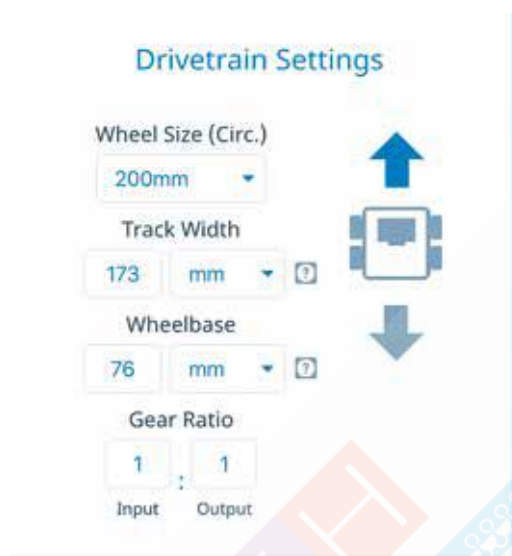


Рисунок 17. Передаточное отношение

Зачем это необходимо? Дело в том, что в среде программирования есть дополнительные встроенные команды движения на заданное количество дюймов/миллиметров, а также поворот на заданное количество градусов.

После добавления переднеприводной трансмиссии в меню с командами появляется новая категория – **Drivetrain**. Программа проезда трансмиссии по заданной траектории (в данном случае квадрат) обычно выглядит, как программа слева: команда управления приводом, а затем время ее работы в секундах или количестве оборотов (Рисунок 18).

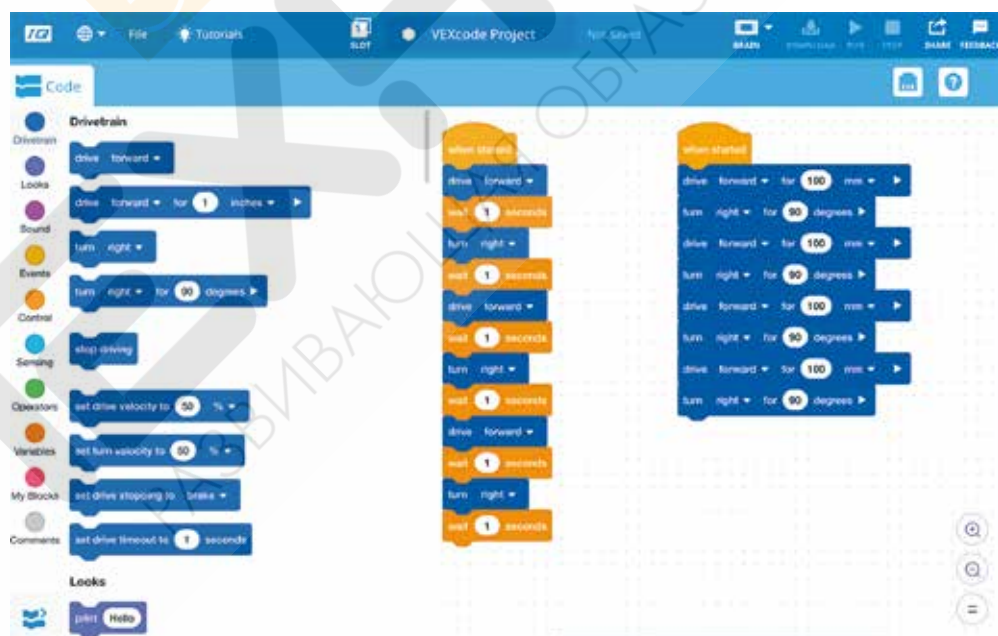












Рисунок 18. Сравнение программ с использованием встроенных команд для движения (справа) и без использования (слева)

Программа, расположенная справа, построена с помощью встроенных команд движения на количество миллиметров и поворота на количество градусов.

Использование встроенных команд поможет освоиться с маневрами быстрее, понять принципы составления линейной программы, но не отменяет необходимости в дальнейшем разобраться в том, как производится расчет пройденного расстояния.

Помимо команд движения вперед-назад и поворотов в разделе **Drivetrain** также присутствуют команды управления параметрами движения (Таблица 4).

Таблица 4. Раздел команд для управления трансмиссией

Раздел	Название	Команда	Описание
 Drivetrain	Привод	Drivetrain  	Движение вперед/назад
		 	Поворот направо/налево
			Остановка движения
			Установить скорость движения
			Установить скорость движения при повороте
			Параметры торможения
			Установка времени ожидания работы привода



Урок 4.

Готовые проекты: изменяем скорость движения

Готовые проекты могут прийти на помощь в начале обучения с учащимися начальных классов, когда процесс от старта работы над роботом до получения первого результата программы не должен занимать много времени, чтобы осталась возможность внести изменения в программу, доработать ее, да и просто немного поиграть с трансмиссией и друг другом.

Одно из удобств готовых проектов – автоматическое подключение приводов и датчиков, в зависимости от выбранного проекта (Рисунок 19).



Рисунок 19. Пример подключения трансмиссии в готовом проекте

При этом для ребят сохраняется необходимость проверки портов для выбранных устройств. Так постепенно они научатся следить за всеми пунктами, которые необходимо проверить перед запуском программы.

Одновременно с этим на экране появляется пример программы, например, в проекте **Adjusting speed** – это изменение скорости во время движения.

Программа представлена для движения в дюймах, которые достаточно легко изменить на миллиметры (Рисунок 20).

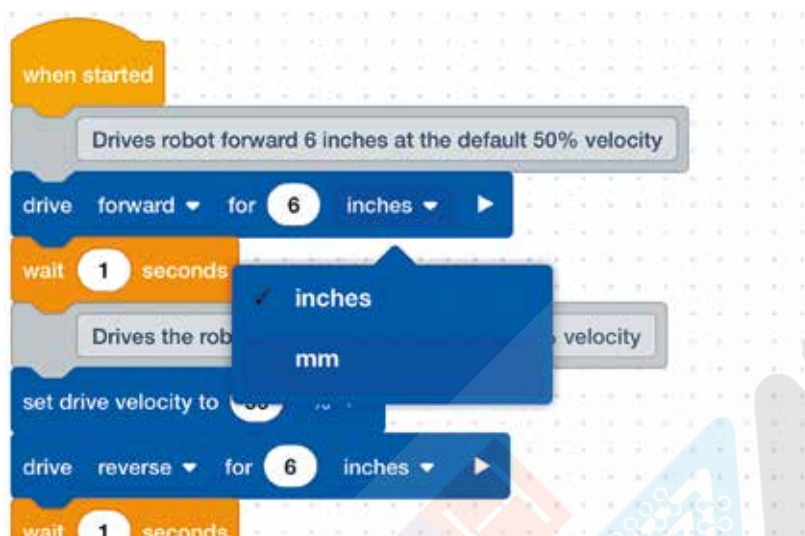


Рисунок 20. Изменение дюймов на миллиметры

Предположим, что роботу необходимо проехать на 5 сантиметров вперед. В таком случае достаточно установить значение 50 миллиметров (Рисунок 21).

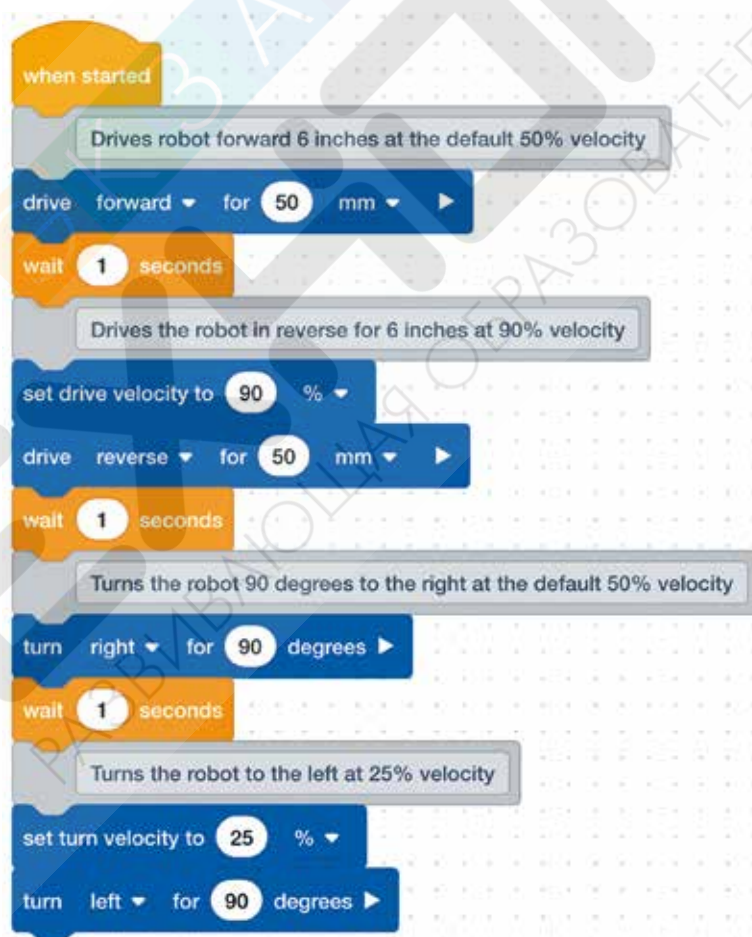


Рисунок 21. Изменение расстояния для движения трансмиссии

Команды серого цвета – это комментарии. Комментарии важны для понимания программы другими программистами, поэтому если ученики уже могут достаточно быстро печатать, то имеет смысл сразу работать с ними в этом направлении.

Серым цветом в программе можно отметить не только комментарий, но и неиспользуемый блок. В программе ниже одна из команд **Set drive velocity** отмечена серым цветом (Рисунок 22). Это значит, что данная команда не будет исполняться.

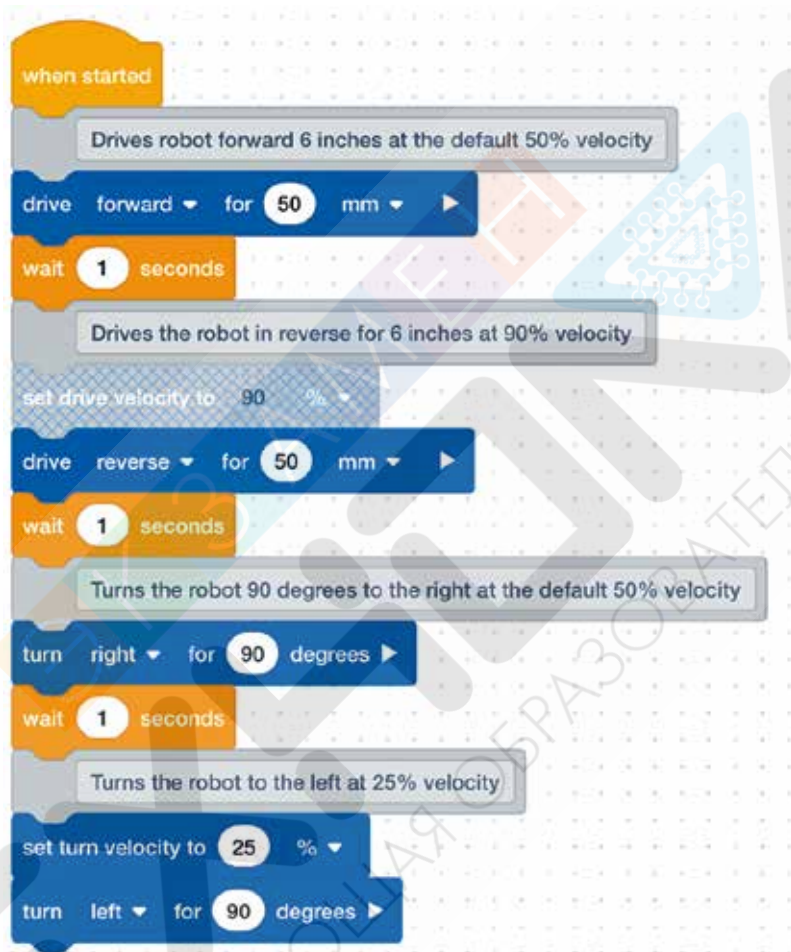


Рисунок 22. Пример неисполняемой команды

Закомментировать или отключить блок можно с помощью команды контекстного меню – **Enable Block** или **Disable block**. Команда удаления или копирования также находится в контекстном меню (Рисунок 23).

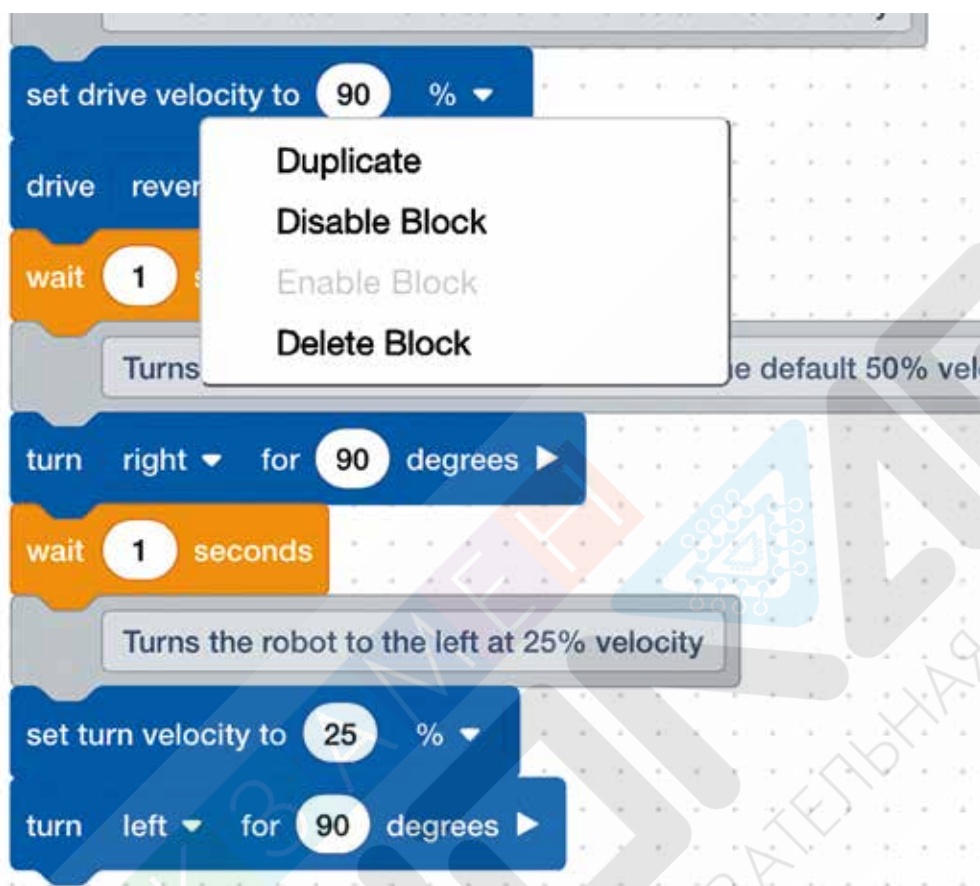


Рисунок 23. Меню включения/выключения блока к исполнению

Урок 5.

Готовые проекты: ускорение

1. Что такое циклы? Циклы **repeat-until** и **while**
2. Что такое переменные?
3. Как ускоряться или замедляться по ходу движения?

Линейные программы, рассмотренные в предыдущих занятиях, позволяют выполнять небольшое количество действий однократно. Представим, что есть необходимость посадить цветы на грядке. Сама грядка состоит из 20 мест. Основываясь на предыдущем опыте, садовод может описать свои действия следующей программой:

- (1) <посадить цветок>
- (2) <посадить цветок>
- (3) <посадить цветок>
- (4) <посадить цветок>
- (5) <посадить цветок>
- (6) <посадить цветок>
- (7) <посадить цветок>
- (8) <посадить цветок>
- (9) <посадить цветок>
- (10) <посадить цветок>
- (11) <посадить цветок>
- (12) <посадить цветок>
- (13) <посадить цветок>
- (14) <посадить цветок>
- (15) <посадить цветок>
- (16) <посадить цветок>
- (17) <посадить цветок>
- (18) <посадить цветок>
- (19) <посадить цветок>
- (20) <посадить цветок>

Программа получилась достаточно длинная. А что будет, если посадить придется не 20 цветов на грядке, а, например, 50 или 100? Такой пошаговый способ уже не будет эффективным. В случае, когда необходимо многократно выполнить одно и то же действие, можно использовать циклы.

Цикл – многократно исполняемая последовательность инструкций.

После организации цикла программа будет выглядеть более компактной и удобной для чтения:

Пока число посаженных цветов не достигло 20:

<посадить цветок>
<посчитать цветы>

Цикл, используемый для решения программы по посадке цветов, называется ци-

клом с предусловием (Рисунок 24). В начале цикла задается и проверяется условие.



Рисунок 24. Цикл с предусловием «Пока»

Если условие выполняется – число посаженных цветов меньше 20 – команды внутри цикла исполняются.

Если условие не выполняется – число посаженных цветов равно 20 – цикл завершает работу.

Похожий принцип работы имеет цикл повтора, пока не наступит условие (Рисунок 25).



Рисунок 25. Цикл с предусловием «Пока не»

В чем отличие? Цикл **while** выполняется, пока условие верно. Цикл **repeat-until** работает наоборот, пока условие не выполняется.

Обязательно внутри таких типов цикла вести счет, в данном случае цветам. Если не указать команду «пересчет цветов» после каждого посаженного, то количество посаженных цветов для программы и для цикла не будет меняться. Даже, если в реальности робот посадит очень большое количество таких цветов, он будет считать, что по-прежнему посадил меньше 20.

Существуют другие виды циклов, например, цикл со счетчиком (Рисунок 26).

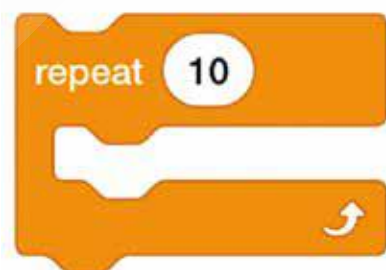


Рисунок 26. Цикл со счетчиком

Условие цикла со счетчиком – заданное количество повторов. Сегодня во многих высокоуровневых (современных) языках программирования цикл самостоятельно следит за тем, чтобы количество выполненных повторов было верным. Но так происходит не во всех языках, поэтому внимательность не повредит.

В VEXcode IQ Blocks цикл со счетчиком может самостоятельно следить за тем, когда нужно закончить работу.

Особенный тип цикла, представленный в среде программирования, – бесконечный цикл (Рисунок 27).

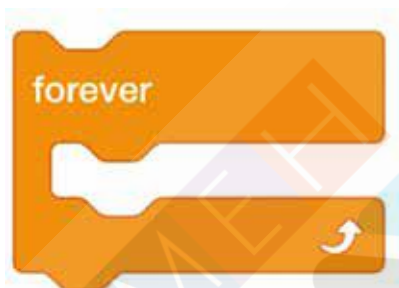


Рисунок 27. Бесконечный цикл

Цикл **forever** выполняется бесконечно, поскольку на входе нет условия, либо условие всегда верно. Другие виды циклов также могут по ошибке становиться бесконечными, если внутри них не производится проверка условия.

При программировании роботов такой тип цикла достаточно часто используется.

Выйти из такого типа цикла все же возможно. Для этого можно использовать команду остановки работы цикла (Рисунок 28).



Рисунок 28. Команда остановки работы цикла

Рассмотрим работу циклов на примере проекта по ускорению во время движения. Представим, что робот проходит полосу препятствий, где вначале идет неровная дорога с плохим покрытием, а затем становится лучше и лучше. Поэтому в начале пути он должен ехать помедленнее, а затем может ускориться.

Как будет выглядеть такая программа (Рисунок 29)? Вначале с помощью команды «установить переменную» выделяется место, где будет храниться информация о скорости, с которой движется робот. Такое место хранения информации и называется **переменной**. Ниже в программе переменная, в которой хранится информация о скорости, называется **velocity**.

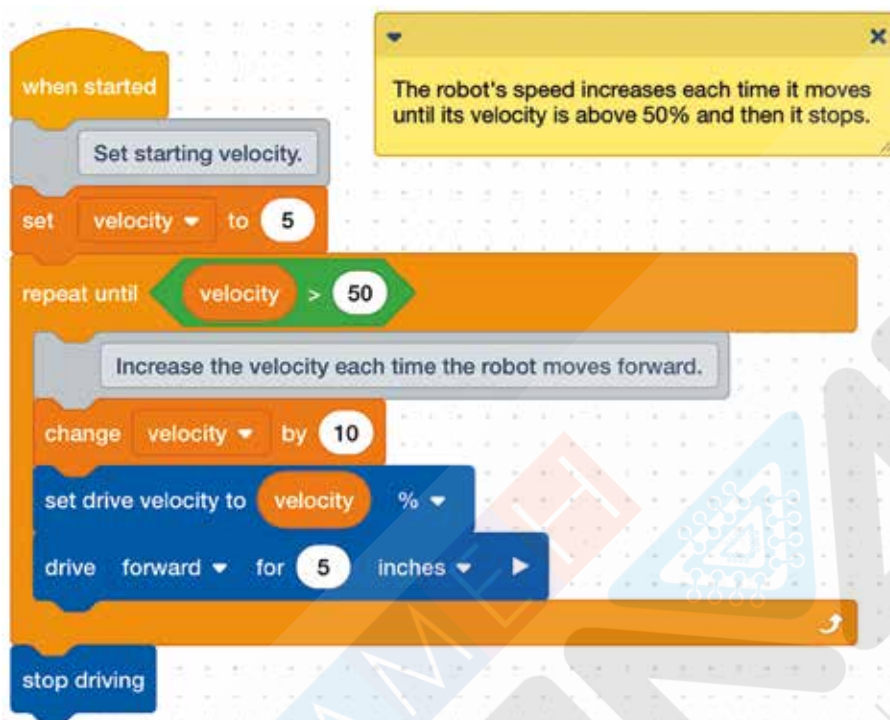


Рисунок 29. Проект по ускорению движения трансмиссии
(исходная программа готового проекта)

Имя переменной можно изменить, выбрав в контекстном меню команду **Rename variable** (Рисунок 30). Однако использовать можно только латинские символы.

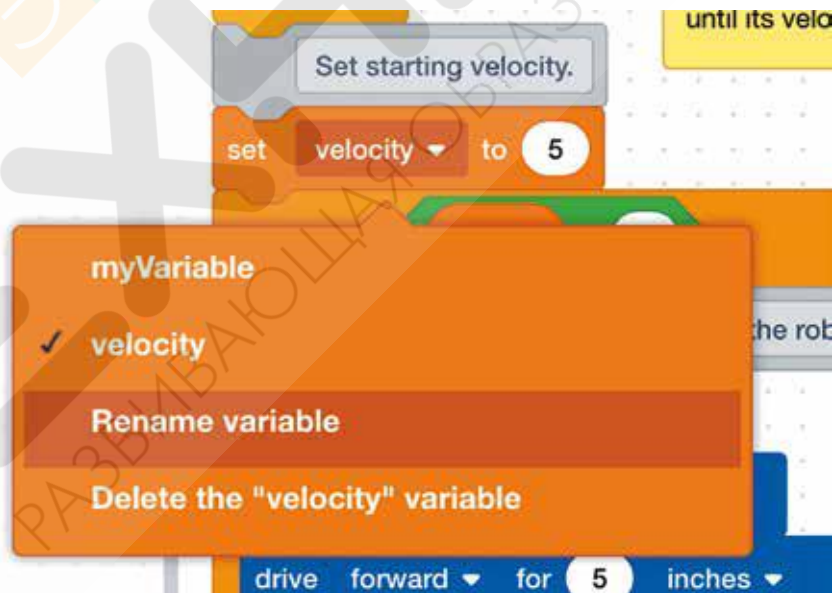


Рисунок 30. Изменение имени переменной

Далее идет уже знакомый цикл повтора, пока не наступит условие. Рассмотрим его отдельно (Рисунок 31).

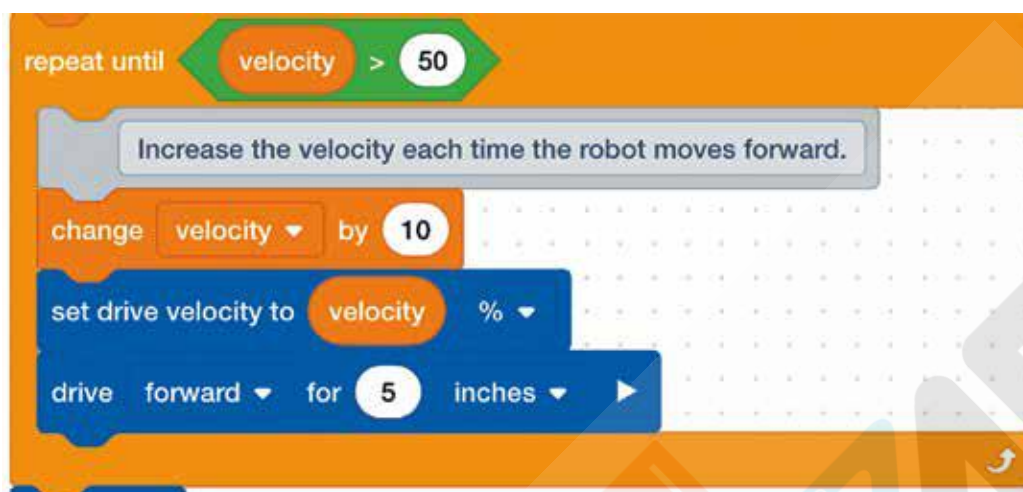


Рисунок 31. Цикл повтора, пока не наступит условие

Цикл будет выполняться до тех пор, пока скорость меньше 50. Как только скорость станет больше 50, цикл прекратит работу.

Далее с помощью команды изменения информации в переменной **Change by** следует изменить скорость на 10 единиц.

Команда установки скорости движения в процентах **Set drive velocity to %** – получается новое значение переменной **velocity**.

И с такой скоростью движется 5 дюймов. В этом случае рекомендуется изменить дюймы на миллиметры для удобства.

Данный набор инструкций повторяется до тех пор, пока скорость не превысит 50 единиц.

После этого цикл остановит работу и выполнит последнюю команду в программе (Рисунок 32).

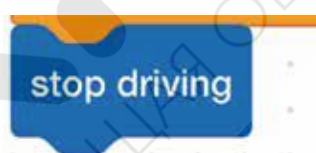


Рисунок 32. Команда остановки приводов

Обратной ситуацией будет постепенное замедление робота. Этот цикл используется, когда необходимо аккуратно подъехать к какому-то объекту, например, чтобы его захватить и перетащить. Для этого необходимо поменять стартовое значение скорости с 5 на 50. Затем поменять условие работы цикла и ждать, пока скорость не станет меньше 5.

Важно также изменять переменную скорости не на 10, а на -10, так как нам нужно постоянно вычитать из 50, а не прибавлять (Рисунок 33). Далее: количество миллиметров можно оставить прежним или изменить по необходимости.

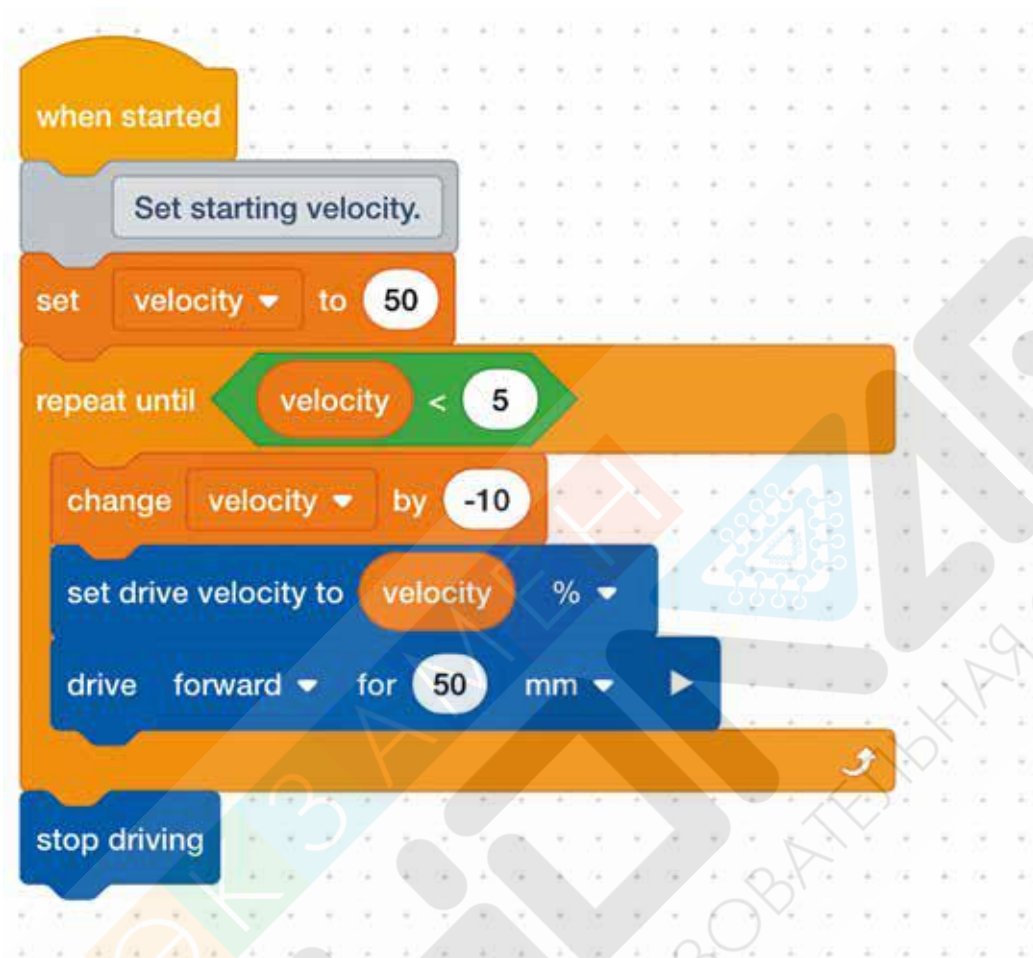


Рисунок 33. Изменение программы для замедления движения

Урок 6.

Готовые проекты: выводим значение датчиков на экран

1. Зачем нужны датчики?
2. Разбираем программу проекта.
3. Как работать с другими датчиками?

В предыдущем занятии робот-садовод смог успешно посадить 20 цветов благодаря использованию циклов в программе. Теперь он столкнулся с новой задачей: ему выделили длинную грядку и не определили количество посадок. Задачу определили так: высаживать, пока грядка не закончится. Что это значит? Как это определить?

Программа робота может собирать информацию от датчиков (или их еще называют сенсорами). Как у человека есть определенные способности восприятия, такие как зрение, слух, осязание, так и у робота есть некоторый набор возможностей:

- п-кодеры могут предоставить информацию о количестве градусов, на которое поворачивается привод;
- датчик касания с LED реагирует на нажатие и может светиться любым цветом;
- датчик цвета дает информацию о том, какой цвет перед ним;
- датчик расстояния может определить расстояние до объекта;
- гироскопический датчик может определить угол, на который его повернули;
- камера может определять различные объекты перед собой.

Предположим, что робот-садовод знает длину грядки в метрах (2 метра), расстояние между цветами (20 см) и может посчитать, на какое количество градусов должен повернуться выходной вал привода, чтобы робот проехал это расстояние. Программист может посчитать идеальное значение, зная диаметр колес, а может сначала попробовать на практике.

Для этого необходимо открыть проект **Display Sensor** из вкладки готовых проектов **Sensing**.

Рассмотрим программу подробнее (Рисунок 34). Вначале две команды печати на экране **Print** выводят значение датчиков перед запуском с помощью команды **Drive heading in degrees**. Затем с помощью команды переноса курсора **Set cursor to next row** переходят на новую строку.

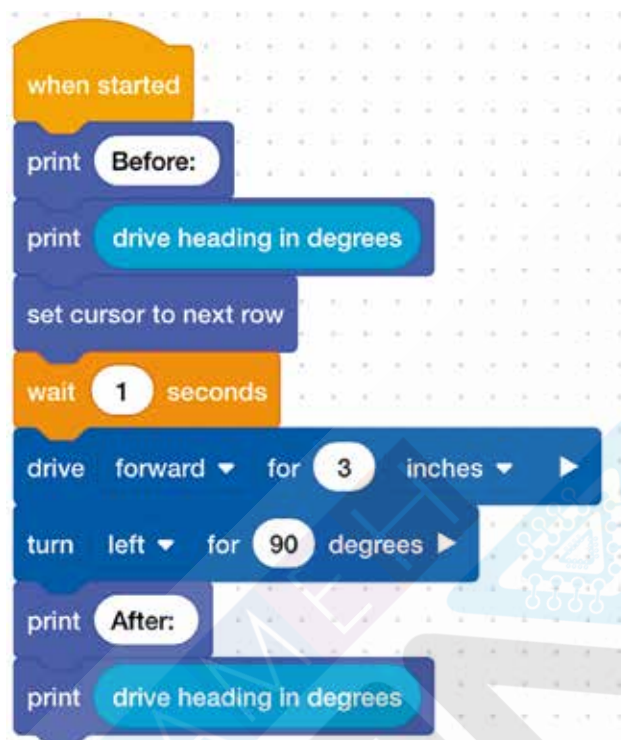


Рисунок 34. Программа вывода на экран показаний п-кодера

После чего робот ждет 1 секунду и начинает движение вперед на 3 дюйма. Заменим дюймы на миллиметры и установим значение 200 (= 20 см). А также уберем следующую команду поворота налево, представим, что роботу-садоводу нет необходимости разворачиваться, чтобы посадить цветок (Рисунок 35).

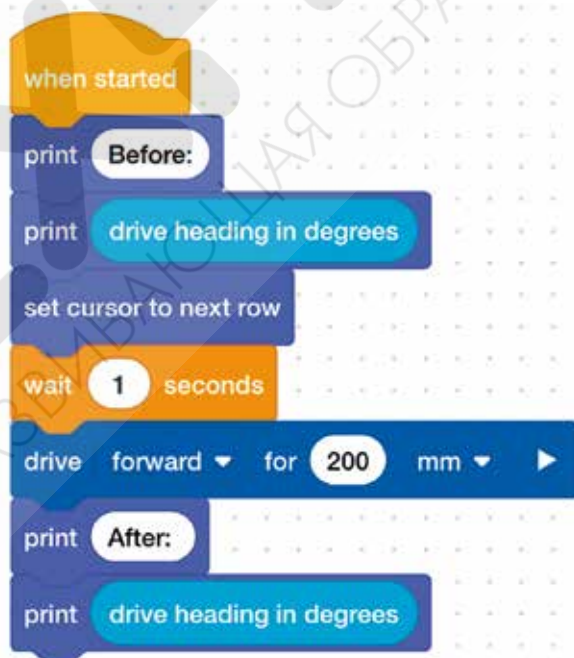


Рисунок 35. Программа движения для посадки цветка роботом-садоводом

Далее команды печати на экран выводят значение п-кодеров после выполнения программы. Теперь программист знает количество градусов, на которое необходимо повернуться приводу чтобы проехать 20 см до места посадки цветка.

Если разделить 2 метра (=200 см) на 20 см, то мы получим количество таких участков для проезда, равное 10. Умножив количество градусов, показанных на экране робота, на 10, программист получит итоговое значение для проезда по всей грядке.

Выводить на экран можно значение не только п-кодеров, но и других датчиков. Для этого необходимо открыть настройки робота и выбрать команду добавления устройства – **Add a device** (Рисунок 36).



Рисунок 36. Меню добавления нового устройства


Затем выбрать необходимый датчик, например, цвета  (Рисунок 37).



Рисунок 37. Выбор необходимого устройства на примере датчика цвета

После среда программирования попросит выбрать порт подключения датчика (Рисунок 38).

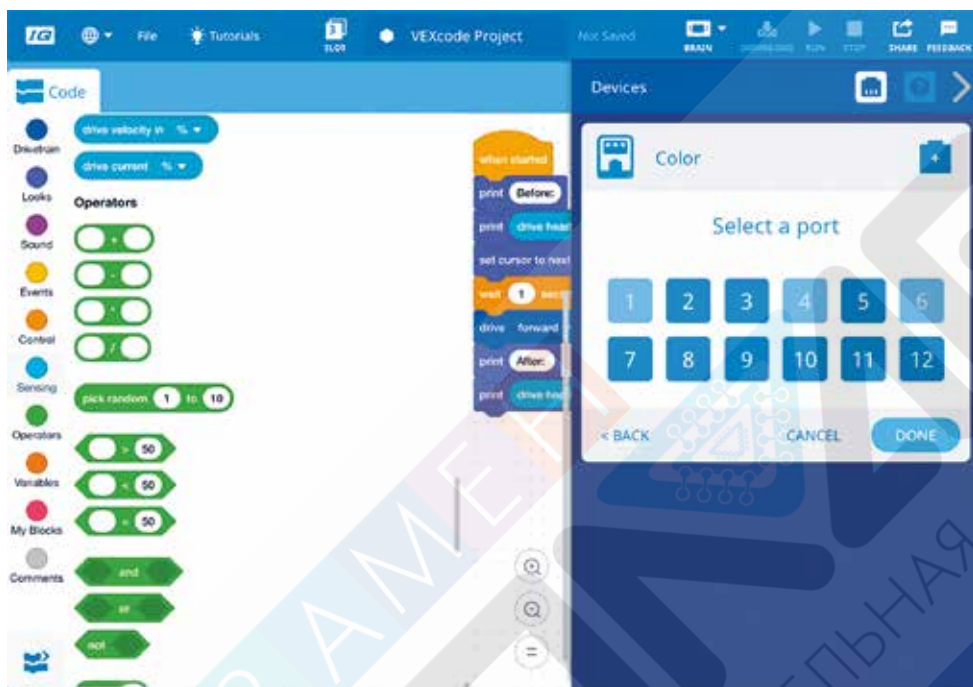


Рисунок 38. Выбор порта для подключения датчика цвета

И можно будет завершить подключение, нажав на кнопку **Done** (Рисунок 39).



Рисунок 39. Завершение подключения нового устройства

Слева в меню команд в разделе Показания объектов (**Sensing**) появятся дополнительные команды для работы с датчиком цвета (Рисунок 40).

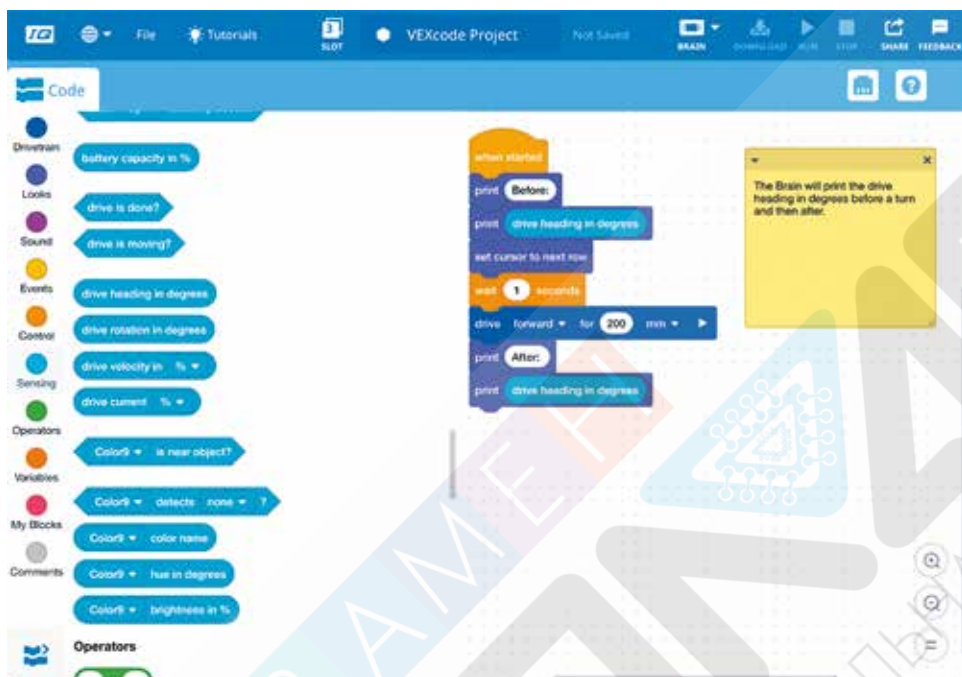


Рисунок 40. Дополнительные команды для работы с датчиком в разделе «Показания объектов»

Используя новые команды, можно заменить вывод на экран градусов вращения привода на имя цвета с помощью команды **Color name** (Рисунок 41).

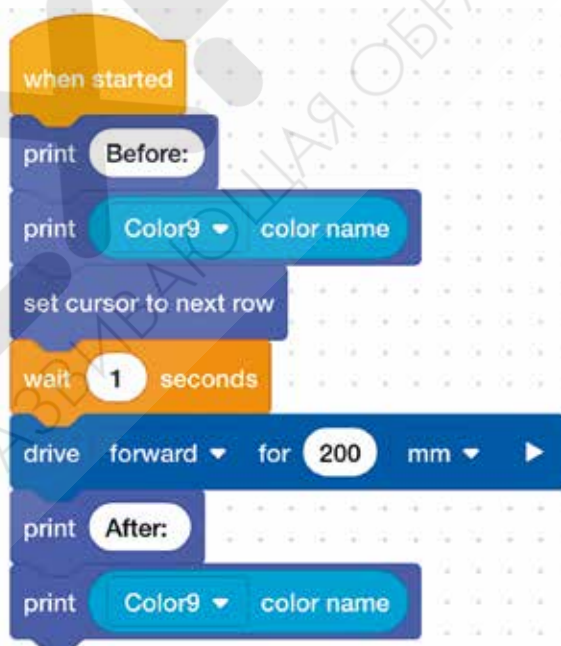


Рисунок 41. Вывод на экран показаний датчика цвета

Вывод значения датчиков на экран полезен также и при отладке программы, когда робот выполняет неправильные действия. Например, он должен распознать белый цвет и остановиться, а он едет дальше. Возможно, датчик видит совсем другой цвет. Если вывести на экран его значение, то можно понять, как исправить ситуацию.



Урок 7.

Готовые проекты: распознавание цветов

1. Как устроена обратная связь?
2. Как опрашивать датчики?
3. Разбираем устройство программы.

Робот-садовод так понравился другим работникам, что они решили дать ему более сложную работу. Теперь его задача – поливать клумбы. Он должен объехать всю территорию сада и полить все желтые цветы. Программист и робот уже знают, что у них есть датчик цвета и с помощью него можно определить, какого цвета объект.

Для того чтобы правильно выполнить задачу, программа робота постоянно должна опрашивать датчик цвета и узнавать, что он видит. И в тот момент, когда он видит желтые цветы, программа должна менять поведение робота и давать команду «полить цветок».

Каждый день человеческий организм выполняет сотни таких действий, можно сказать, что человек в своей обычной жизни постоянно выполняет похожие программы. Например, когда мы кипятим воду для того, чтобы что-то приготовить, мы постоянно смотрим, нет ли пузырьков на поверхности. Как только вода начинает бурлить, мы понимаем, что она закипела и можно бросать туда, например, макароны.

Мы получаем сигнал и меняем свое поведение в зависимости от полученного значения. И не просто меняем, наши действия могут поддерживать процесс, а могут его прекратить.

Более сложный пример: катание на велосипеде. Для того чтобы ехать прямо, необходимо ровно держать руль. Если мы случайно повернули руль в сторону или решили объехать лужу, то после этого нужно повернуть руль в обратную сторону. Если отклонение от курса влево сильное, то и повернуть вправо надо сильно, а если совсем чуть-чуть, то и обратное действие должно быть небольшим. Мы постоянно оцениваем ситуацию и принимаем решения, которые помогают поддерживать процесс движения на велосипеде.

Такой принцип в науке называют обратной связью. **Обратная связь** – влияние на систему таким образом, что отклонение (ошибка) между желаемым состоянием системы и действительным будет уменьшаться или увеличиваться.

Обратная связь, способствующая уменьшению такого отклонения, то есть приводящая действительное состояние системы к желаемому, называется отрицательной обратной связью. Несмотря на то что слово «отрицательный» редко связывают с благоприятными последствиями для того, чтобы получить стабильную работающую систему, нужно использовать именно отрицательную обратную связь.

Как же роботу-садоводу получить такую обратную связь в своей работе?

Откроем готовый проект по распознаванию цветов **Detecting colors** из раздела проектов Показания объектов (**Sensing**) (Рисунок 42).



Рисунок 42. Выбор проекта из меню готовых проектов

Программа проекта достаточно длинная, но если внимательно прочитать команды, то окажется, что она состоит из похожих частей (Рисунок 43).



Рисунок 43. Исходная программа готового проекта по распознаванию цветов

Рассмотрим подробнее каждую из них. В начале программы находится уже знакомый бесконечный цикл **forever** (Рисунок 44).



Рисунок 44. Бесконечный цикл

При работе с обратной связью циклы часто бывают необходимы, ведь для того, чтобы всегда знать, что показывает датчик, ему нужно задавать вопросы с постоянной периодичностью.

Как будут выглядеть эти вопросы?

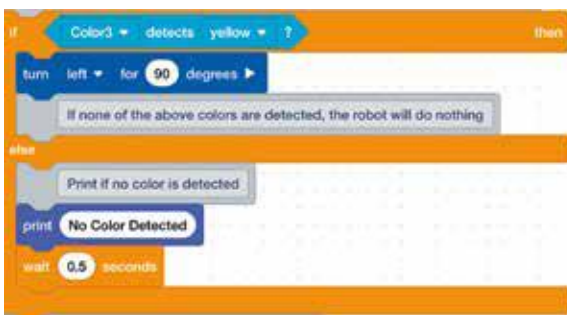

Во-первых, такие вопросы должны быть сформулированы таким образом, чтобы на них можно было ответить только «да» или «нет».

Во-вторых, при ответе на такой вопрос «да» необходимо указать, что в таком случае делать.

В программировании существует такая структура управления, как условный оператор. Рассмотрим примеры таких условий из программы по распознаванию цветов (Таблица 5).

Таблица 5. Примеры условий из программы готового проекта по распознаванию цветов

	<p>Условный оператор if помогает задать правильный вопрос. Условие Датчик цвета распознает (Color detects) из раздела Измерения (Sensing) и позволяет спросить, видит ли датчик цвета красный цвет?</p> <p>Если ответ от датчика на этот вопрос «да», тогда внутри if записывается команда для выполнения – проехать вперед (Drive forward)</p>
	<p>В данном вопросе датчику нас интересует уже не красный, а голубой цвет. Если датчик видит голубой цвет, то программа предлагает двигаться назад (Drive reverse)</p>
	<p>Третий по счету условный оператор задает поведение робота при распознавании зеленого цвета. Ему нужно повернуться направо (Turn right)</p>

	<p>Четвертый условный оператор определяет поведение робота, если датчик видит желтый цвет – это поворот налево (Turn left). Но у него есть не только часть с ответом «да» (Then – «Тогда»), но и часть, если ответ «нет» (Else – «Иначе»), в которой сказано, что нужно вывести на экран сообщение о том, что никакой цвет не распознан при помощи команды печати Print</p>
	<p>После прохода по всем условиям программа предлагает очистить экран (Clear all rows) и вернуть курсор в начало строки (Set cursor to row)</p>

Для того чтобы реализовать идею полива желтых цветов, столько условных операторов не понадобится, можно оставить только для желтого цвета (Рисунок 45).



Рисунок 45. Измененная программа для распознавания желтого цвета для робота-садовода

Определимся также с реакцией, если желтый цвет обнаружен. Роботу необходимо остановиться и подать звук о том, что желтые цветы обнаружены и могут быть по-

литы. Для этого заменим команду поворота на команду остановки **Stop Driving**, добавим команды вывода на экран **Print** и команду воспроизведения звука **Play Sound** (Рисунок 46).



Рисунок 46. Действия робота, когда датчик распознает желтый цвет

Осталось определить, что будет делать робот, когда не видит желтый цвет. Он может, например, просто ехать вперед (**Drive forward**) (Рисунок 47).

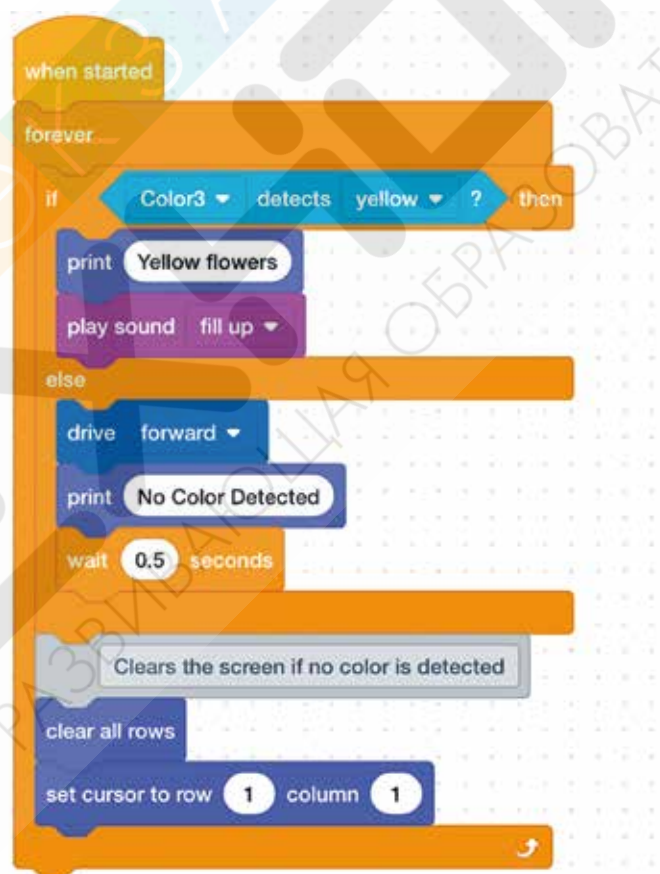


Рисунок 47. Готовая программа с добавлением действий робота при отсутствии цвета

Программа с условным оператором поможет объехать клумбу и узнать, есть ли на ней желтые цветы. Но как быть, если клумб несколько? Как их найти? Рассмотрим эти вопросы на следующем занятии.



Урок 8.

Готовые проекты: распознавание объектов

Территория сада состоит из 4 клумб, расположенных на одной площади при входе в основное здание. С одной клумбой робот-садовод справился: цветы успешно политы. Как отыскать другие?

Что для этого обычно делает человек? Он оглядывается и ищет все, что похоже на клумбу. Задача в том, чтобы объяснить роботу, что же именно похоже на клумбу.

Какое из «чувств» робота поможет объяснить ему, как отличить клумбу? Датчик касания может определить объект, если он близко и датчик его плотно касается, но различий между клумбами и другими предметами, например столбами, он не видит.

Датчик расстояния может определить препятствие перед собой на большом расстоянии, но не делает различий между предметами: они все для него препятствие.

Датчик цвета может отличать объекты друг от друга, если один синий, другой желтый, а третий красный и т.д. Но придется подъехать очень близко к предмету, так как датчик работает на маленьких расстояниях.

Камера – модуль технического зрения – сочетает в себе и распознавание объекта по цвету, и распознавание объектов на расстоянии. Кроме того, модуль технического зрения одновременно может распознавать до 7 цветов (Рисунок 48).

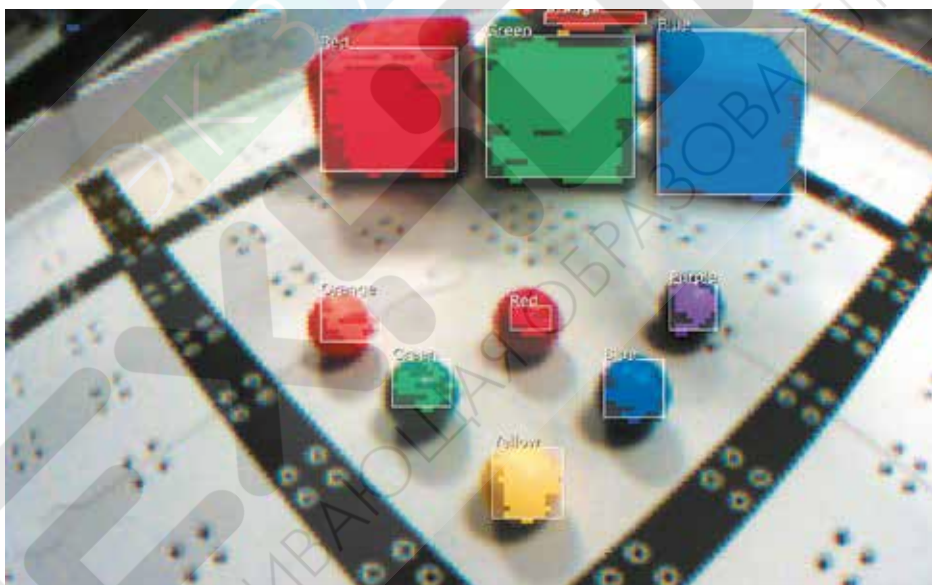


Рисунок 48. Пример распознавания объектов модулем технического зрения

Робот-садовод и программист предложили бортики вокруг клумб покрасить в голубой цвет (отличный от желтых и красных цветов, растущих на клумбах, а также от зелени вокруг). Тогда робот, приезжая на площадь, может с помощью камеры определить клумбу, подъехать к ней и выполнить необходимые действия по поливке. Затем повернуться и поискать соседнюю клумбу, подъехать к ней и так далее.

Выберем проект **Detecting objects** из раздела **Показания устройств (Sensing)**.

На экране появится готовая программа (Рисунок 49).

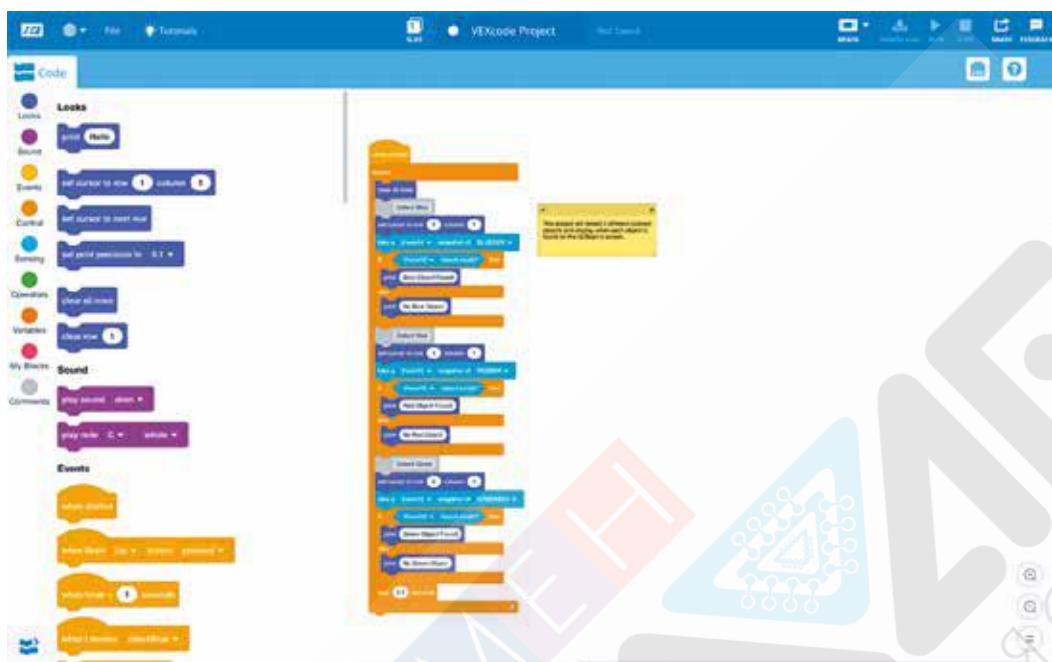


Рисунок 49. Программа готового проекта по распознаванию объектов

Рассмотрим отдельные ее части подробнее. Начинается программа с бесконечно-го цикла **Forever (Всегда)** (Рисунок 50). При необходимости его можно заменить на любой другой с условием. Далее внутри идут команды очистки экрана и установки курсора. После чего начинается часть программы, отвечающая за распознавание объектов – **Take a snapshot (Сделать снимок)**.

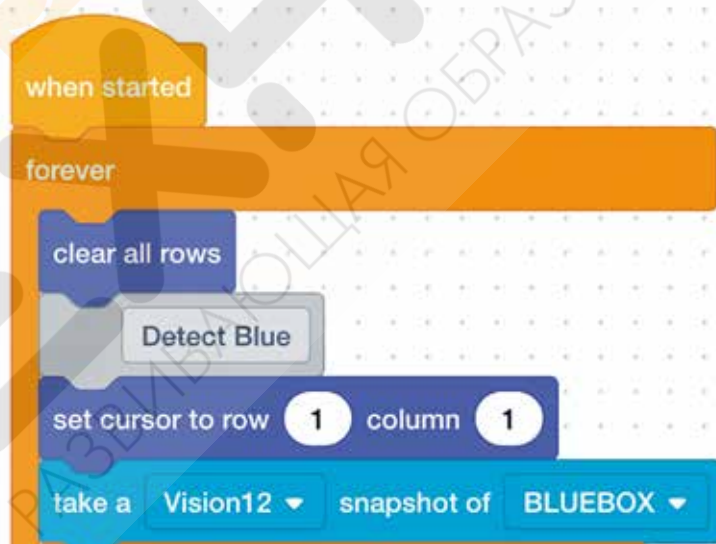


Рисунок 50. Начало программы с бесконечным циклом

В этом примере команде камере предлагается сделать снимок объекта **BLUEBOX**. Что это значит? Если открыть раздел устройств и выбрать датчик Vision, то появится меню настроек (Рисунок 51).

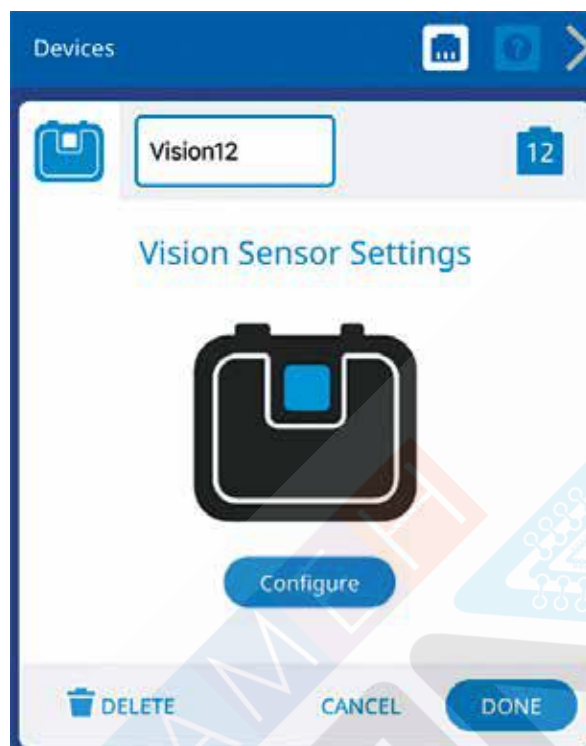


Рисунок 51. Меню настроек модуля технического зрения

При нажатии на кнопку **Configure** (Конфигурация) откроется другое меню – калибровки работы камеры (Рисунок 52).



Рисунок 52. Меню для калибровки модуля

Одновременно камера может распознавать 7 объектов разных цветов. Стандартные объекты голубого, красного и зеленого цветов названы **BLUEBOX**, **REDBOX** и **GREENBOX** соответственно. При желании названия объектов можно изменить. Рядом с каждым объектом есть окно **Clear** (Очистить значение), а рядом с ним двойная стрелка (Рисунок 53).

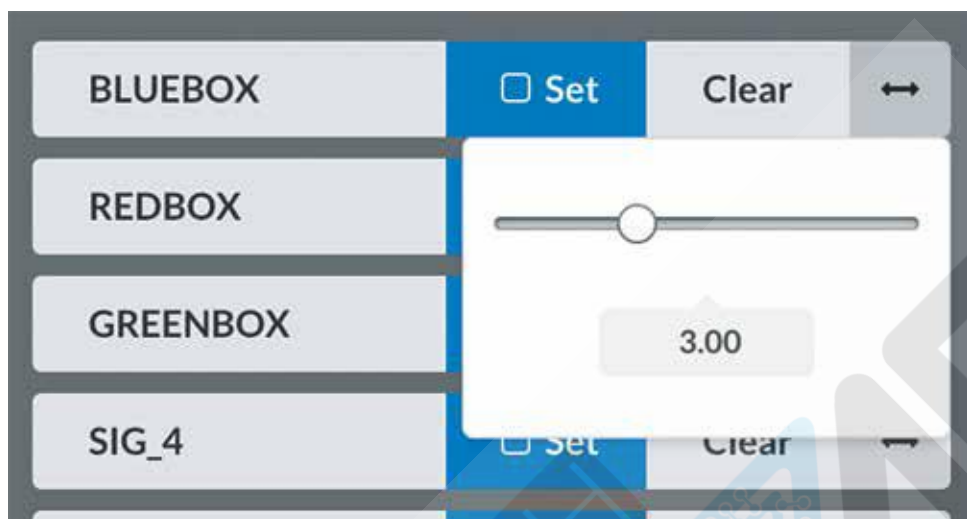


Рисунок 53. Настройка цветов

При нажатии на стрелку можно отрегулировать уровень определения голубого цвета. Автоматическое значение не всегда позволяет корректно распознать объект, ведь освещение и глубина цвета объекта всегда разные. Уровень определения можно настроить вручную с помощью этого инструмента.

BLUEBOX обозначает, что команде было дано задание сделать снимок объекта голубого цвета.

После того как снимок сделан, можно распознавать, есть ли объект голубого цвета на снимке с помощью условного оператора **if (если)** (Рисунок 54).



Рисунок 54. Распознавание объекта голубого цвета

Если объект существует, то на экран будет выведено сообщение об этом. Иначе будет выведено сообщение о том, что объект не найден. Для того чтобы сделать программу для робота-садовода, необходимо будет заменить или дополнить сообщение на экране командами движения. Для этого нужно будет в меню устройств добавить **Drivetrain (Трансмиссия)** (Рисунок 55).

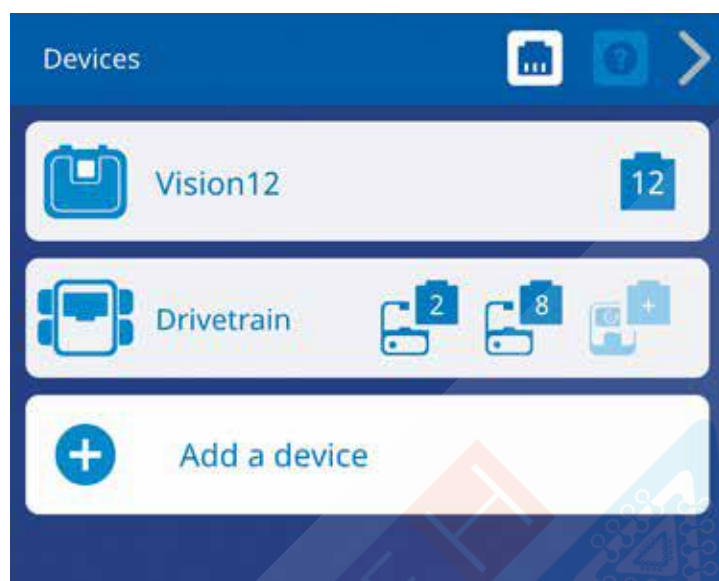


Рисунок 55. Подключение трансмиссии

После этого в меню команд слева появится раздел **Drivetrain (Трансмиссия)**. Добавим команду движения вперед на 30 см, если объект распознан. Если объект не найден, будем поворачиваться на угол 90 градусов, чтобы поискать клумбу в другом месте (Рисунок 56).

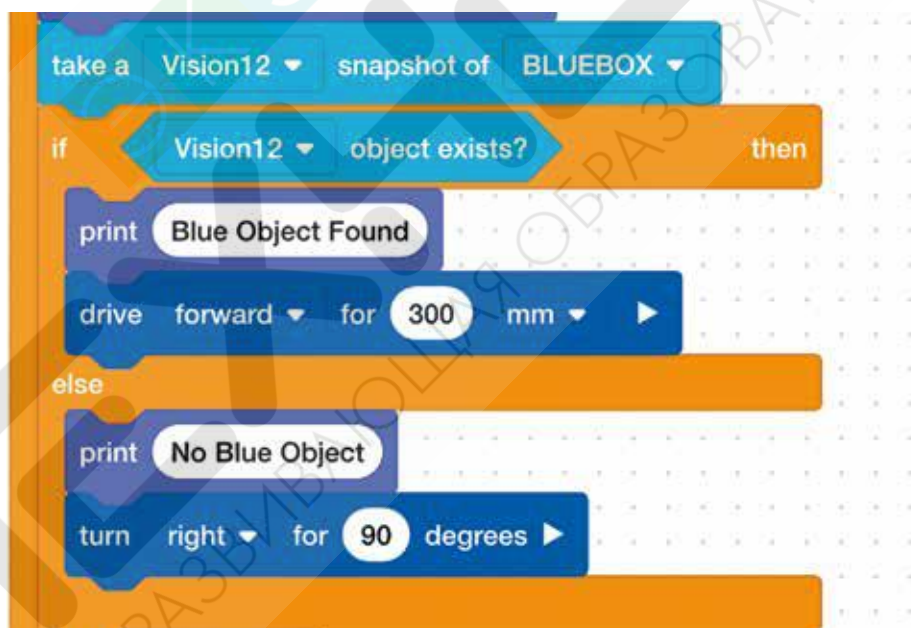


Рисунок 56. Действия робота при распознавании объекта-клумбы

Остальные части программы позволяют похожим образом распознать красный и зеленый объекты. Их можно убрать, но оставить в конце цикла ожидание 0,1 секунды для корректной работы цикла. Итоговая программа будет выглядеть таким образом (Рисунок 57).

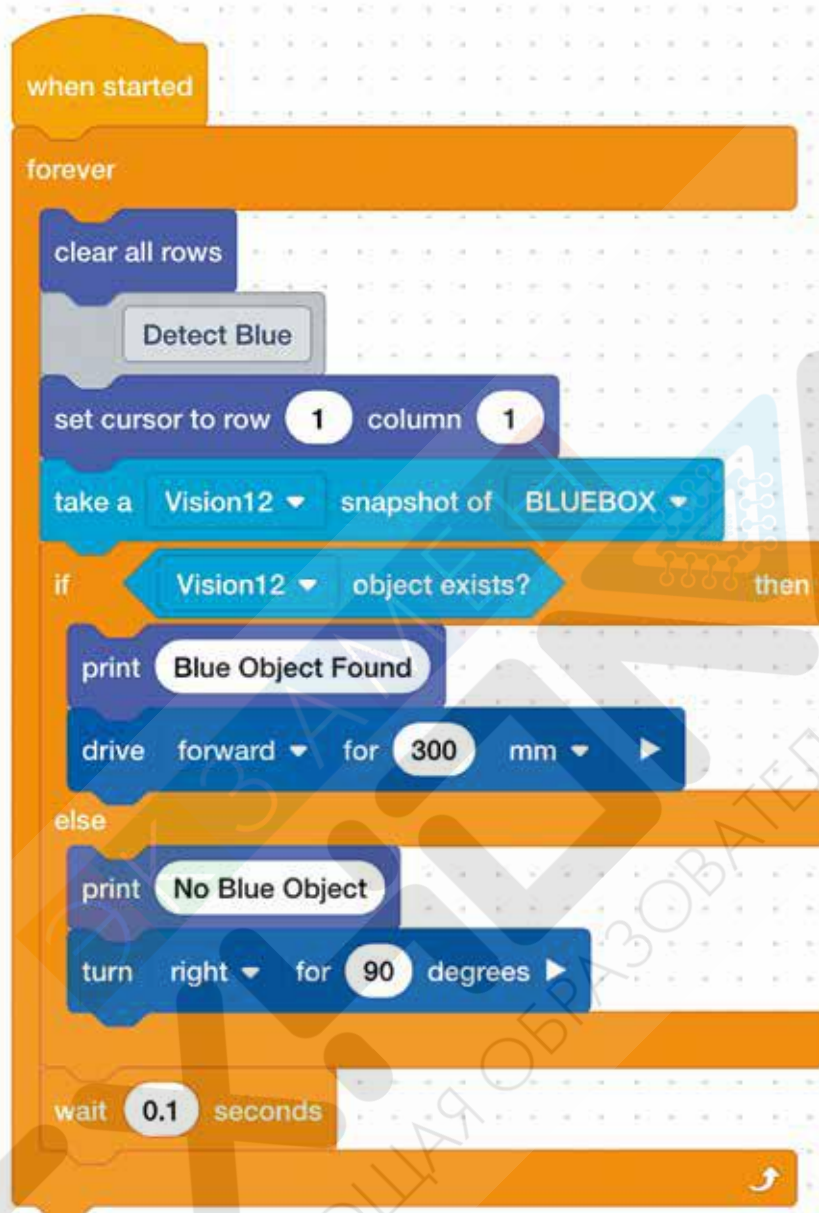


Рисунок 57. Готовая программа

Можно также добавить поиск другой клумбы после того, как подъехали к первой и выполнили полив (Рисунок 58).



Урок 9.

Собственные проекты: вертушка Ньютона

Предыдущие занятия были основаны на использовании готовой логики и изменении уже написанных программ из готовых проектов. Но не все шаблоны могут подойти для решения собственной идеи. Как поступить в таком случае? С чего начать работу?

В первую очередь необходимо словесно описать идею для робота (или другого устройства). Рассмотрим процесс работы над собственной идеей на примере эксперимента со светом «Вертушка Ньютона».

<https://www.youtube.com/watch?v=KiwpVtQ-4gc&t=1s> – адрес в Интернете, где можно посмотреть эксперимент «Вертушка Ньютона».

Словесно устройство можно описать так: цветовой круг, разделенный на 7 цветов, вращается с высокой скоростью. Все цвета на нем сливаются в один – белый. Этот процесс демонстрирует разделение и слияние спектра в единый белый свет. Дополнительно можно добавить распознавание цвета роботом – сможет ли он определить, что это белый?

Далее преобразуем словесное описание в схему работы.

Запуск программы

|

Мотор вращается с прикрепленным к нему цветовым кругом.

|

Датчик цвета определяет цвет и выводит его на экран.

|

Отключение программы

Такое небольшое техническое задание описывает круг работы по конструированию и программированию. Конструкция должна будет учитывать положение мотора, способ крепления на него бумажного цветового круга, а также расположение повышающей передачи для увеличения скорости вращения привода. Программа будет состоять из команды вращения привода.

Примерная конструкция с повышающей передачей представлена в инструкции по сборке. Ее также можно усовершенствовать самостоятельно, увеличив передаточное отношение.

При подключении робота ранее использовалась **2-motor Drivetrain (2-моторная трансмиссия)**. В данной конструкции будет использован только один привод, поэтому вместо трансмиссии выберем его (Рисунок 59).

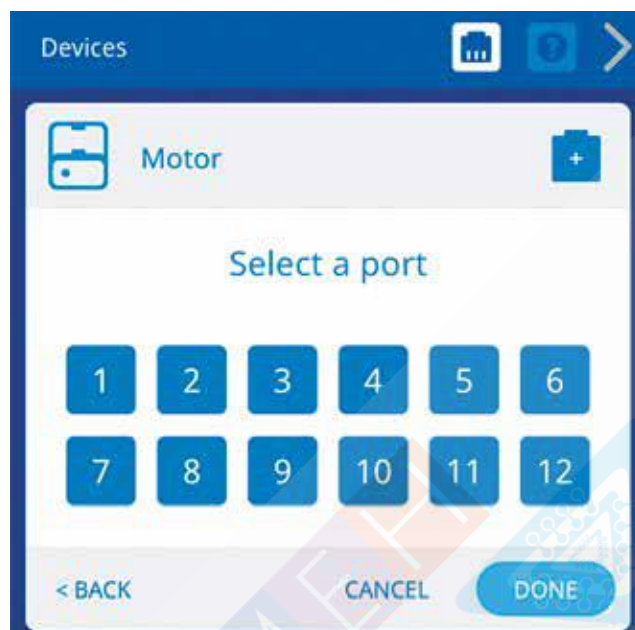


Рисунок 59. Подключение привода











В настройках можно указать не только порт, но и направление вращения – обычное или реверсивное (Рисунок 60).



Рисунок 60. Настройки привода

При подключении одного мотора в меню появляется новый раздел **Motion** (Движение) (Таблица 6).

Таблица 6: Раздел «Движение»

Раздел	Название	Команда	Описание
 Motion	Движение	Motion 	Вращать привод вперед/назад
			Вращать привод на заданное количество градусов
			Вращать привод до позиции в градусах
			Остановить привод
			Установить привод на позицию в градусах
			Установить скорость вращения привода в процентах
			Установить удержание привода
			Установить максимальный момент вращения в процентах
			Установить работу привода в секундах

Все готово к созданию программы.

Все действия робот должен будет выполнять сразу после запуска и останавливаться только при выключении программы, значит, можно использовать бесконечный цикл (Рисунок 61).



Рисунок 61. Бесконечный цикл

Запустим работу привода (Рисунок 62).



Рисунок 62. Вращение привода

Дополнительно можно менять начальную скорость вращения и установить ее на 100% (Рисунок 63).



Рисунок 63. Изменение скорости вращения привода

Далее работа над собственным проектом подразумевает развитие идеи. Данный проект можно развивать как конструкционно (например, менять передаточное число и пытаться добиться идеального белого цвета), так и программно (например, попробовать распознать вращающийся объект как белый с помощью камеры или датчика цвета).

Урок 10.

Собственные проекты: Театр теней

1. Что такое Театр теней?
2. Особенности конструкции: жесткость и прочность.
3. Освещение сцены с помощью TouchLED.

Идеи для собственного проекта далеко не всегда должны исходить из научных экспериментов. Отличным примером может стать создание конструкции для Театра теней.

Сцена театра устроена очень просто: источник света и белый экран, а между ними находятся бумажные кукольные герои или сами актеры (Рисунок 64).



Рисунок 64. Вид сцены для Театра теней сбоку

Начать работу можно с конструкции для экрана. Его можно сделать прямоугольным или необычной формы многоугольника.

При создании конструкции важно помнить о том, что есть фигуры жесткие, а есть, наоборот, легко меняющие свою форму. К первым можно отнести треугольник, а ко вторым – квадрат (Рисунок 65).



Рисунок 65. Треугольник и квадрат

Любой квадрат можно сделать жесткой фигурой, достаточно добавить диагональ. Получится, что квадрат стал разделен на два треугольника (Рисунок 66).

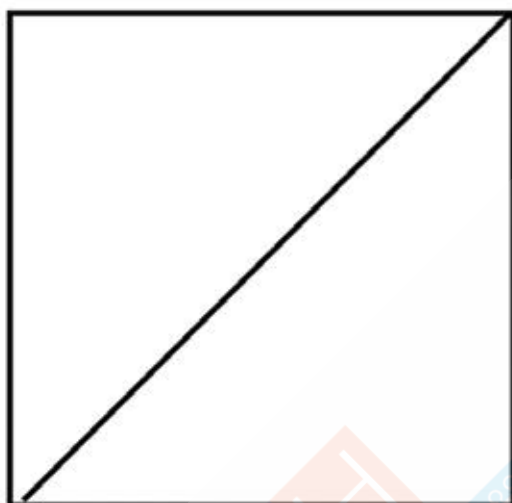


Рисунок 66. Диагональ помогает сделать квадрат жесткой фигурой

Одним из наглядных примеров можно назвать конструкции высотных подъемных кранов.

При создании конструкции для поддержки экрана стоит помнить о том, чтобы соединения были крепкими и жесткими.

Подачу света можно интересно обыграть с помощью TouchLED. Даже один датчик светит достаточно ярко для темного помещения. Еще один плюс использования TouchLED для Театра теней в том, что каждую сцену легко можно подсветить разным цветом и подчеркнуть смысл происходящего.

При подключении датчика появляются новые возможности (Таблица 7).

Таблица 7: Изменения в разделе вывода на экран при подключении TouchLED

Раздел	Название	Команда	Описание
Looks	Вывод на экран	set TouchLED3 color to none	Установить цвет
		set TouchLED3 fade to slow	Угасать быстро/медленно
		set TouchLED3 brightness to 50 %	Установить яркость свечения

Используя данные блоки и блок ожидания в секундах, можно составить программу в зависимости от длительности и настроения спектакля. Пример программы представлен ниже на рисунке 67.

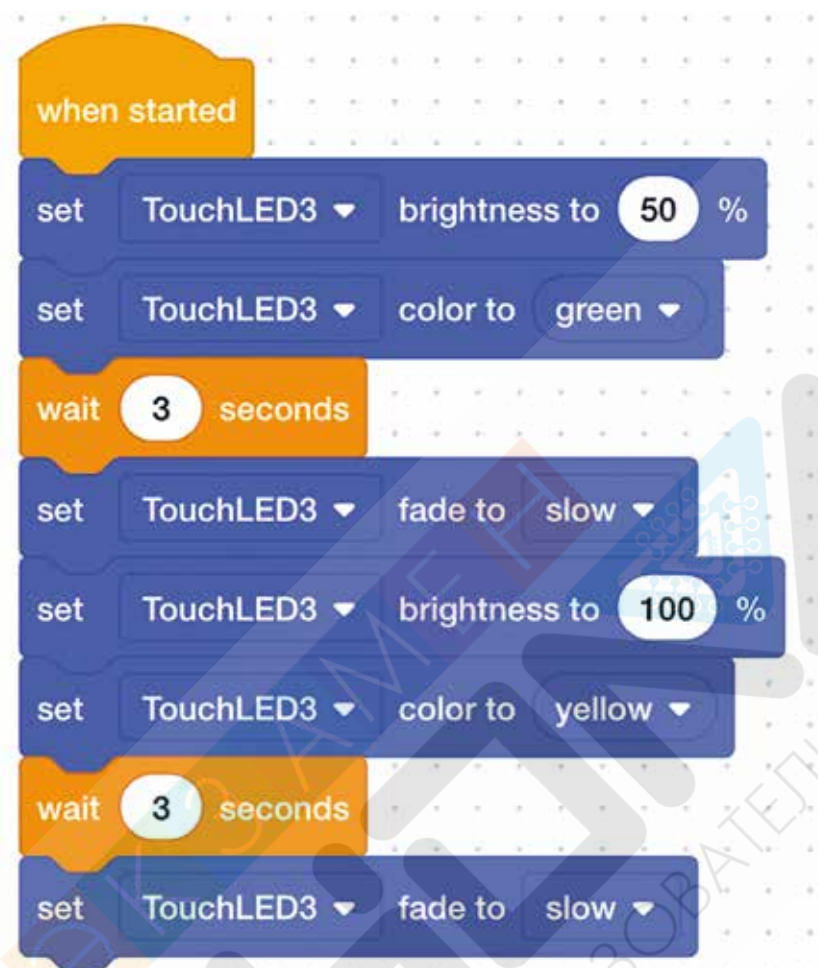


Рисунок 67. Пример программы для работы Театра теней

Если собрать большую сцену, то использовать можно несколько датчиков TouchLED. Расставить их за сценой и каждому назначить свой цвет и время. Возможно и запрограммировать управление с пульта, где за работу каждого датчика будет отвечать своя кнопка. Любую творческую идею можно сопроводить интересным техническим решением с помощью робототехнического конструктора.

Учебно-методическое издание

**Волкова Екатерина Вадимовна
Мацаль Игнатий Игнатьевич**

Основы программирования в среде VEXcode IQ

Издательство «ЭКЗАМЕН»
«ЭКЗАМЕН-ТЕХНОЛАБ»

Гигиенический сертификат
№ РОСС RU C-RU.AK01.H.04670/19 с 23.07.2019 г.

Главный редактор *Л. Д. Лаппо*
Корректоры *О. Ю. Казаньева, В. В. Кожуткина*
Дизайн обложки
и компьютерная верстка *А. А. Винокуров*

107045, Россия, Москва, Луков пер., д. 8.
E-mail: по общим вопросам: robo@examen-technolab.ru;
www.examen-technolab.ru
по вопросам реализации: sale@examen-technolab.ru
тел. +7 (495) 641-00-23

Общероссийский классификатор продукции
ОК 034-2014; 58.11.1 – книги печатные

Отпечатано в соответствии с предоставленными материалами
в ООО «ИПК Парето-Принт», 170546, Россия, г. Тверь, www.pareto-print.ru