

Цель: изучить интерфейс и возможности программирования роботов в среде VEX EXP.

Предметные результаты:

- знать принципы подключения и настройки робототехнического набора VEX EXP
- уметь создавать простые программы для управления движением робота.
- Загружать программы на робота и тестировать их.

Используемое оборудование и материалы:

- Набор VEX EXP с робототехническими компонентами.
- Компьютер со средой программирования VEXcode.
- USB-кабель для подключения контроллера к компьютеру.

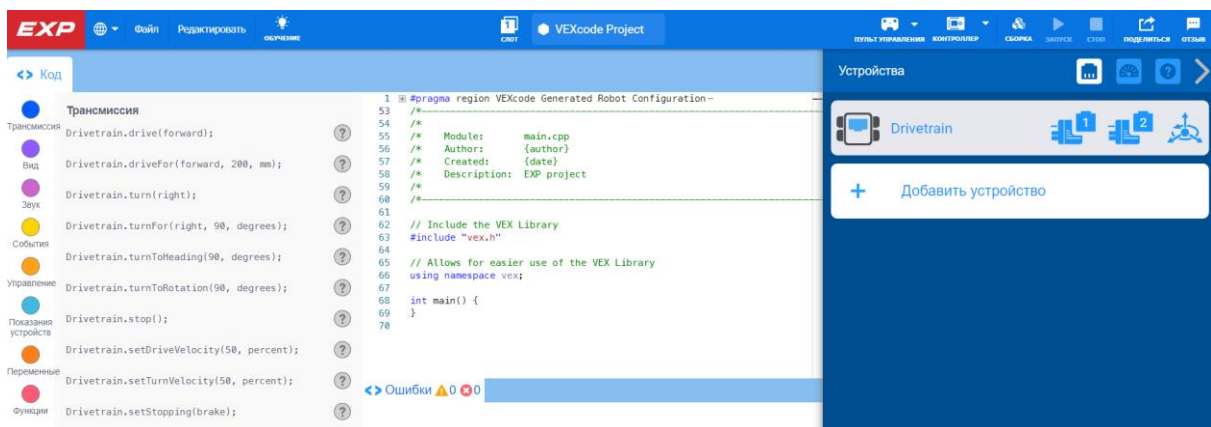


Теоретический материал



Программируемый контроллер для набора VEX EXP – это микроконтроллер, который предоставляет возможность управления робототехническими конструкциями. Контроллер подключается к компьютеру с помощью USB-кабеля для программирования и загрузки кода на робота.

У контроллера есть несколько портов для подключения датчиков, моторов и других устройств. Каждый порт имеет свое назначение и может быть использован для управления различными устройствами в работе.



Для программирования контроллера можно использовать специальное программное обеспечение VEX EXP или веб среду, которое предоставляет удобный интерфейс для создания программ на блочном языке программирования или Python/C++. С помощью этого программного обеспечения можно написать код для управления моторами, обработки сигналов с датчиков, осуществления автономного движения и многое другое.

Настройка контроллера включает в себя выбор правильного порта для подключения устройств, определение типа датчика или мотора, настройку параметров движения и другие действия, необходимые для корректной работы робота.

Практическая работа

Задания:

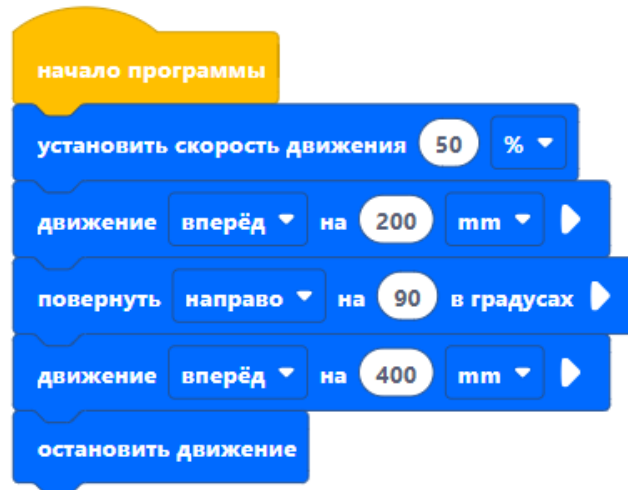
1. Подготовить набор VEX EXP и компьютер с VEXcode.
2. Подключить к компьютеру контроллер робота с помощью USB-кабеля.
3. Запустить среду VEXcode и создать новый проект.
4. Написать программу для движения робота по заранее заданному маршруту.
5. Загрузить программу на контроллер робота и запустить ее.
6. Провести тестирование работы программы на роботе.

Контрольные вопросы

1. Какие языки программирования робота можно использовать в среде VEX EXP?
2. Какие способы управления движением робота можно использовать в среде VEX EXP?

Приложение

Фрагмент кода (блочный язык программирования)



Фрагмент кода (C++)

```
float myVariable;  
  
int whenStarted1() {  
    Drivetrain.setDriveVelocity(50.0, percent);  
    Drivetrain.driveFor(forward, 200.0, mm, true);  
    Drivetrain.turnFor(right, 90.0, degrees, true);  
    Drivetrain.driveFor(forward, 400.0, mm, true);  
    Drivetrain.stop();  
    return 0;  
}  
  
int main() {  
    calibrateDrivetrain();  
    printf("\033[30m");  
    wait(100, msec);  
    whenStarted1();  
}
```

Фрагмент кода (Python)

```
myVariable = 0  
def when_started1():  
    global myVariable  
    drivetrain.set_drive_velocity(50, PERCENT)  
    drivetrain.drive_for(FORWARD, 200, MM)  
    drivetrain.turn_for(RIGHT, 90, DEGREES)  
    drivetrain.drive_for(FORWARD, 400, MM)  
    drivetrain.stop()  
  
calibrate_drivetrain()  
when_started1()
```

Базовый робототехнический набор

Тема: Функция как упрощение процесса создания программного кода или легко читаемый код

Цель: изучить правила создания и использования функций в среде VEX EXP, а также применение их для управления движением робота.

Предметные результаты:

- знать принципы подключения и настройки робототехнического набора VEX EXP
- уметь создавать простые программы для управления движением робота.
- Загружать программы на робота и тестировать их.

Используемое оборудование и материалы:

- Набор VEX EXP с робототехническими компонентами.
- Компьютер со средой программирования VEXcode.
- USB-кабель для подключения контроллера к компьютеру.



Теоретический материал

```
21 // Begin project code
22
23 Drivetrain.driveFor(forward, 100, mm, true)
24 }
```



Пользовательские функции в программировании представляют собой независимые блоки кода, которые разработчики создают для выполнения определенной задачи или действия. Они позволяют улучшить структуру и управляемость кода, разделяя большие задачи на более мелкие и понятные компоненты.

Функция - это именованный блок кода, который выполняет определённую задачу и может быть вызван из любого места в программе.

Когда разработчик создает функцию, он дает ей имя и определяет последовательность команд, которые будут выполнены при вызове этой функции. Функции могут принимать входные данные, которые называются аргументами, и возвращать результат выполнения определенных действий.

Преимущества использования пользовательских функций включают:

- Повторное использование кода: Функции позволяют использовать один и тот же блок кода несколько раз без необходимости его копирования.
- Улучшение читаемости кода: Разделение задач на функции делает код более организованным и понятным для других разработчиков.
- Упрощение отладки: Отдельные функции могут быть протестированы и отлажены независимо от остального кода, что упрощает обнаружение и исправление ошибок.
- Модульность: Функции позволяют создавать модули кода, которые могут быть легко модифицированы и обновлены без воздействия на другие части программы.

Практическая работа

Задания:

1. Подготовить набор VEX EXP и компьютер с VEXcode.
2. Подключить программу к контроллеру робота с помощью USB-кабеля.
3. Запустить среду VEXcode и создать новый проект.
4. Написать программу для движения робота по заданному маршруту.
5. Создание пользовательской функции для управления движением робота (например, функция "moveForward").
6. В функции определить последовательность команд для движения робота вперед.
7. Использовать созданную функцию в программе для управления роботом.
8. Загрузить программу на контроллер робота и запустить ее.

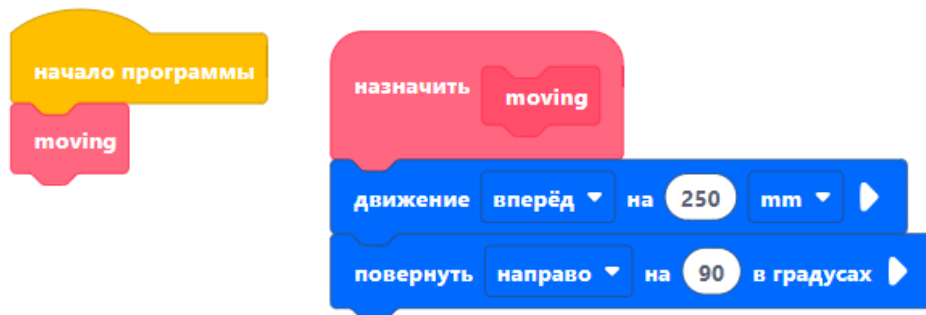
9. Провести тестирование работы программы на роботе.

Контрольные вопросы

1. Что такое пользовательская функция в программировании?
2. Какие преимущества предоставляют функции в коде робота?
3. Почему функции делают код более читаемым и удобным для отладки?

Приложение

Фрагмент кода (блочный язык программирования)



Фрагмент кода (C++)

```
void myblockfunction_moving();
float myVariable;

void myblockfunction_moving() {
    Drivetrain.driveFor(forward, 250.0, mm, true);
    Drivetrain.turnFor(right, 90.0, degrees, true);
}

int whenStarted1() {
    myblockfunction_moving();
    return 0;
}

int main() {
    calibrateDrivetrain();
    printf("\033[30m");
    wait(100, msec);
    whenStarted1();
}
```

Фрагмент кода (Python)

```
myVariable = 0

def moving():
    global myVariable
    drivetrain.drive_for(FORWARD, 250, MM)
    drivetrain.turn_for(RIGHT, 90, DEGREES)

def when_started1():
    global myVariable
    moving()

calibrate_drivetrain()
when_started1()
```

Базовый робототехнический набор

Тема: Движение робота до препятствия с использованием датчика касания

Цель: изучить правила разработки программы для движения робота до препятствия, используя датчик касания.

Предметные результаты:

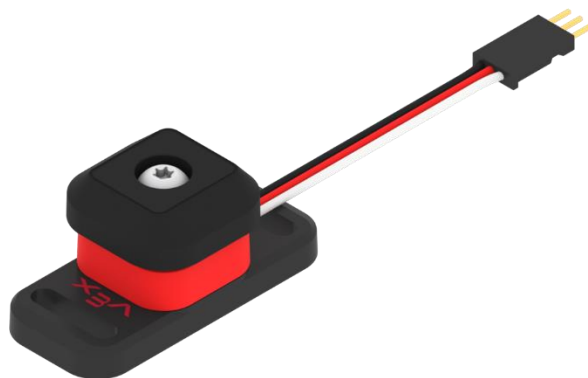
- знать принцип работы датчика касания и его роль в робототехнике
- уметь программировать робота для выполнения задачи остановки перед препятствием
- уметь корректировать свой код для обеспечения правильной работы робота.

Используемое оборудование и материалы:

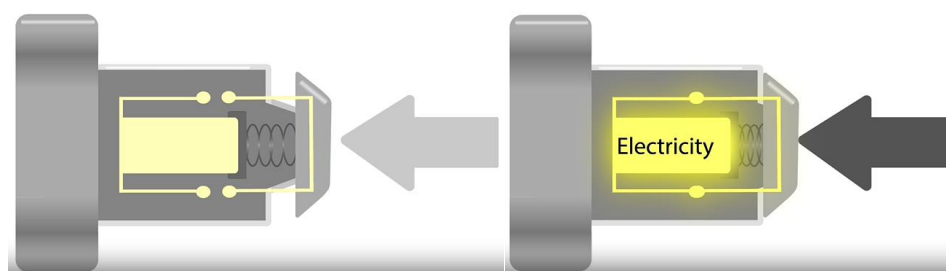
- Набор VEX EXP с робототехническими компонентами.
- Компьютер со средой программирования VEXcode.
- USB-кабель для подключения контроллера к компьютеру.
- Датчик касания.
- Линейка или штангенциркуль для измерений.



Теоретический материал



VEX Bumper Switch основан на наиболее часто используемом электрическом устройстве: переключателе. Переключатель состоит из двух клемм (мест для присоединения провода) и проводного моста для «создания» соединения при нажатии переключателя.

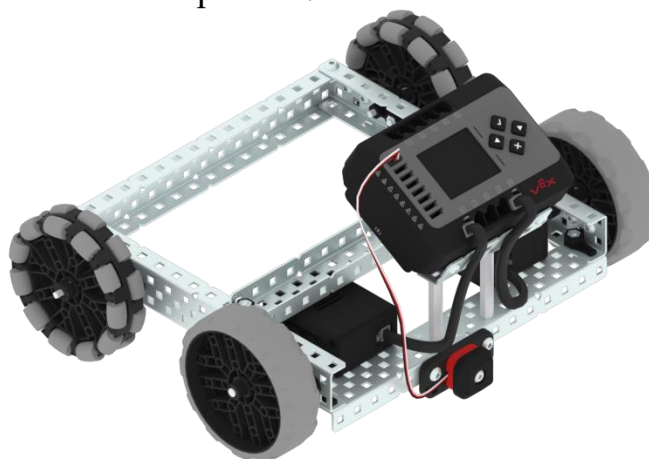


Расположение переключателя Bumper Switch очень важно для получения точных показаний. Убедитесь, что никакая конструкция на роботе не находится перед кнопкой на лицевой стороне датчика. Перед датчиком должен быть свободный путь между любым объектом, на который нажимают, и датчиком.

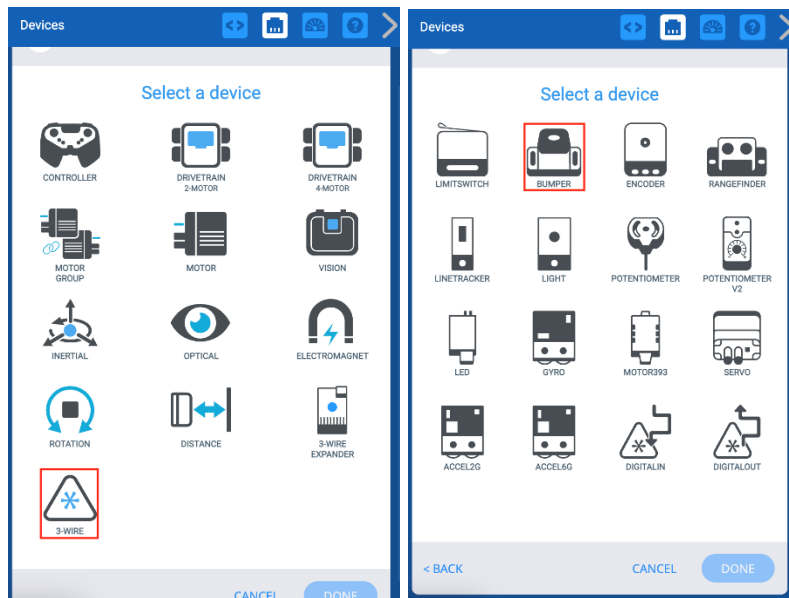
Практическая работа

Задания:

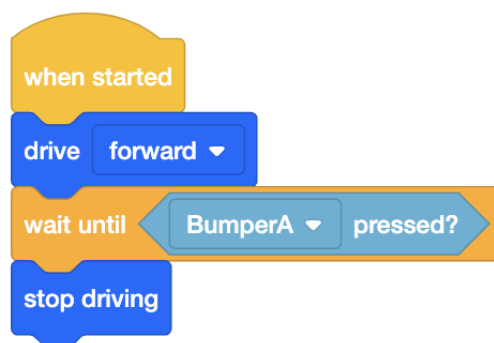
1. Установить датчик касания на робота.



2. Подготовить набор VEX EXP и компьютер с установленным VEXcode.



3. Подключить программу к контроллеру робота с помощью USB-кабеля.
4. Написать программу для движения робота до препятствия с использованием датчика касания.



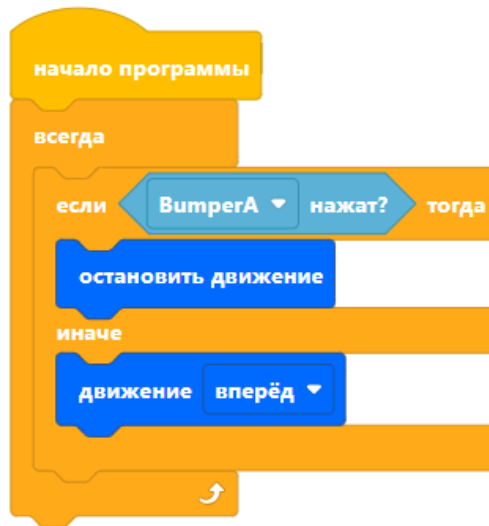
5. Провести тестирование работы программы на роботе.

Контрольные вопросы

1. Как работает датчик касания? Какие физические принципы позволяют ему определять касание?
2. В каких проектах может быть полезен датчик касания?
3. Насколько точно реагирует робот на данные, получаемые с датчиков?
4. Почему использование датчика касания важно для систем автоматизации и робототехники?

Приложение

Фрагмент кода (блочный язык программирования)



Фрагмент кода (C++)

```
float myVariable;

int whenStarted1() {
  while (true) {
    if (BumperA.pressing()) {
      Drivetrain.stop();
    }
    else {
      Drivetrain.drive(forward);
    }
    wait(5, msec);
  }
  return 0;
}

int main
  calibrateDrivetrain();
  printf("\033[30m");
  wait(100, msec);
  whenStarted1();
}
```

Фрагмент кода (Python)

```
myVariable = 0

def when_started1():
  global myVariable
  while True:
    if bumper_a.pressing():
      drivetrain.stop()
    else:
      drivetrain.drive(FORWARD)
      wait(5, MSEC)

calibrate_drivetrain()
when_started1()
```

Цель: освоить основы работы с пультом дистанционного управления роботом и познакомиться с принципами работы ветвления в программе управления.

Предметные результаты:

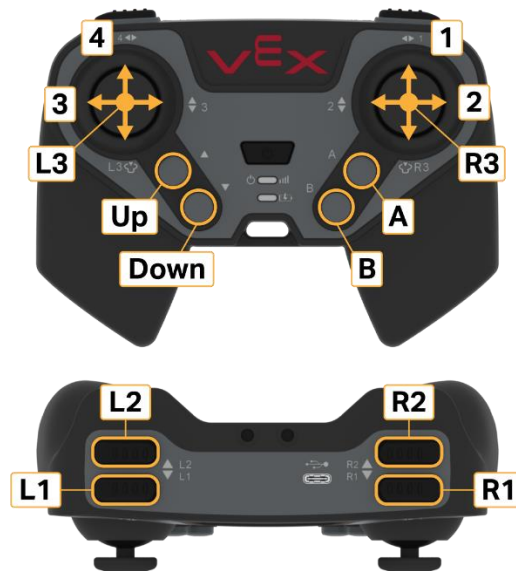
- знать принципы работы дистанционного управления и ветвления в программировании
- уметь программировать робота с использованием пульта (джойстика)

Используемое оборудование и материалы:

- Набор VEX EXP с робототехническими компонентами.
- Компьютер со средой программирования VEXcode.
- USB-кабель для подключения контроллера к компьютеру.
- Пульт дистанционного управления.



Теоретический материал

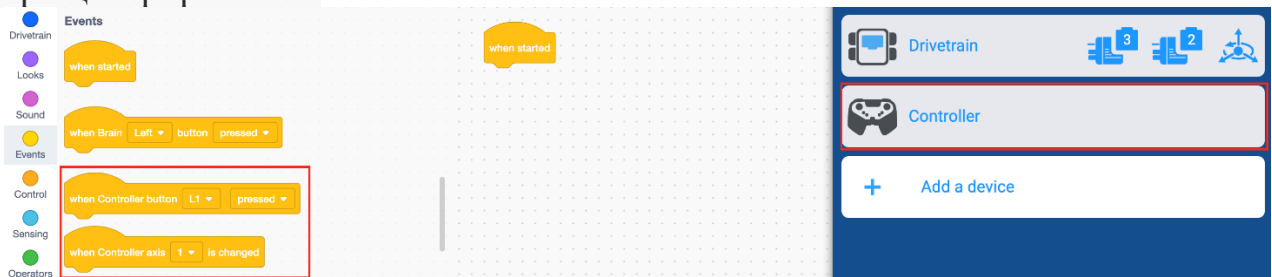


Дистанционное управление роботом является одним из основных способов управления робототехническими устройствами издалека. Это позволяет оператору контролировать и управлять движениями робота без необходимости нахождения непосредственно рядом с ним. Для этого используется специальный пульт дистанционного управления, который передает команды на исполнительные устройства робота через беспроводное или проводное соединение.

Принципы ветвления в программировании представляют собой способ организации выполнения программы на основе различных условий и ситуаций. Этот подход позволяет создавать гибкие и адаптивные системы управления, где решения принимаются на основе текущих данных или внешних обстоятельств. К примеру, при использовании ветвления в программе управления роботом, можно предусмотреть различные сценарии действий в зависимости от обнаруженных препятствий, изменений окружающей среды или других факторов.

Комбинирование дистанционного управления роботом с принципами ветвления в программировании позволяет создавать эффективные и гибкие системы управления, способные адаптироваться к различным условиям и ситуациям. Это особенно важно в современной робототехнике, где требуется высокая степень автономности и гибкости в управлении роботами для эффективного выполнения различных задач.

Каждая ось джойстика возвращает значение от -100 до +100 и возвращает значение ноль при центрировании.



При программировании понимание названий кнопок и джойстика, а также их положения на контроллере помогает определить, какой блок следует использовать.

Практическая работа

Задания:

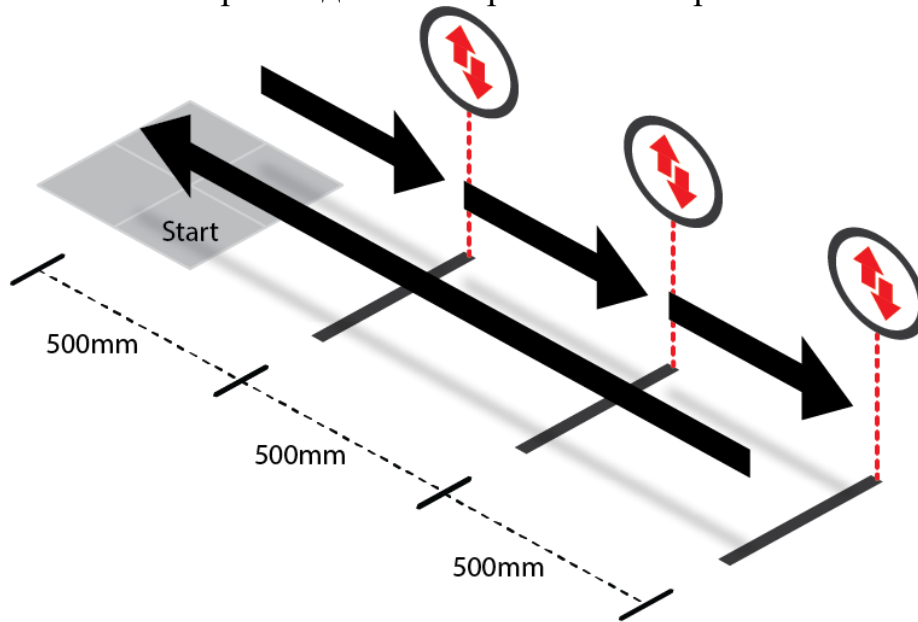
1. Подготовить робота к работе и проверить его функциональность.
2. Установите связь между пультом управления и роботом.



При беспроводном подключении светодиод контроллера и светодиод Power/Link джойстика должны мигать зеленым цветом, показывая, что они подключены.

3. Протестировать работу робота с использованием пульта управления, выполнить базовые команды (например, движение вперед, влево, вправо и т.д.).

Вариант движения роботом на время



Контрольные вопросы

1. Какие возможности предоставляет ветвление в программировании для управления роботом?
2. Какие типичные команды можно передавать роботу с помощью пульта дистанционного управления?

Цель: освоить основы работы с клешней на роботе для перемещения объектов.

Предметные результаты:

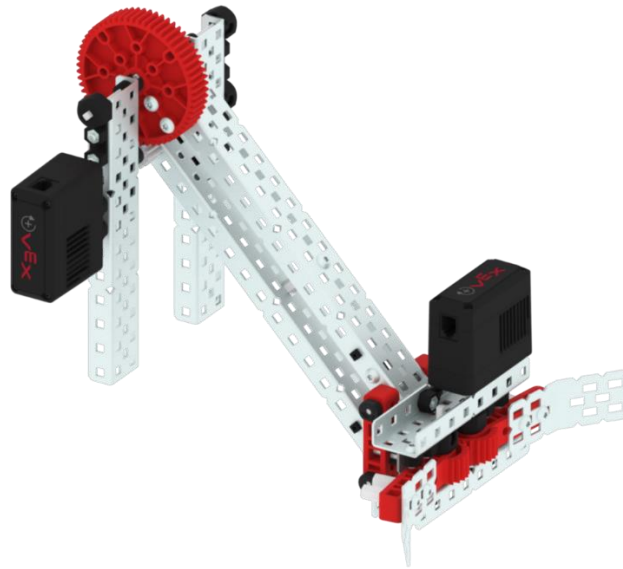
- знать основные принципы механики, включая законы движения и силы
- уметь разрабатывать простые алгоритмы управления клешнями, включая последовательности захвата и отпускания объектов

Используемое оборудование и материалы:

- Набор VEX EXP с робототехническими компонентами.
- Компьютер со средой программирования VEXcode.
- USB-кабель для подключения контроллера к компьютеру.
- Клешня (схват), совместимая с используемым роботом.
- Объект для перемещения (например, кольцо или мяч).
- Датчики (при необходимости): датчик расстояния или касания.
- Линейка или штангенциркуль для измерений.



Теоретический материал



Клешня, или захват, представляет собой механическое устройство, используемое в робототехнике для захвата, удержания и перемещения объектов. Она работает по принципу манипуляции с предметами, обеспечивая возможность выполнения различных задач, таких как сборка, транспортировка и сортировка. Клешни могут иметь разные формы и размеры в зависимости от типа объектов, с которыми они работают.

Клешня обычно состоит из двух или более подвижных частей, которые сходятся в центральной точке. При работе клешни учитываются следующие принципы:

Сила трения: Для надежного захвата объекта необходимо, чтобы клешня могла обеспечить достаточную силу трения между объектом и поверхностью захвата. Это особенно важно, если объект находится в движении.

Геометрия захвата: Форма и угол наклона клешни могут влиять на ее способность захватывать объекты различных форм. Клешни могут быть спроектированы для захвата предметов в виде цилиндров, кубов, а также асимметричных форм.

Принцип действия: Современные клешни могут использовать пневматику (воздушное давление) или сервоприводы для открытия и закрытия захвата. Пневматические системы обеспечивают более плавное и мощное действие, в то время как электрические приводы позволяют более точно контролировать движение.

Управление клешней осуществляется путем программирования контроллера робота. Существуют несколько аспектов, которые необходимо учитывать:

- Использование датчиков (например, датчиков касания или расстояния) позволяет клешне определять, когда захватить предмет и насколько сильно его сжимать. Это важно для предотвращения повреждений как захватываемого объекта, так и клешни.
- Программы могут включать алгоритмы, которые определяют, как и когда клешня должна захватывать и освобождать предметы. Например, использование состояния цикла, где клешня остается в определенном положении до тех пор, пока не будет получен сигнал от датчика.

Практическая работа

Задания:

1. Установить захват на робота.
2. Запустить программное обеспечение и создать базовую программу для движения захвата.

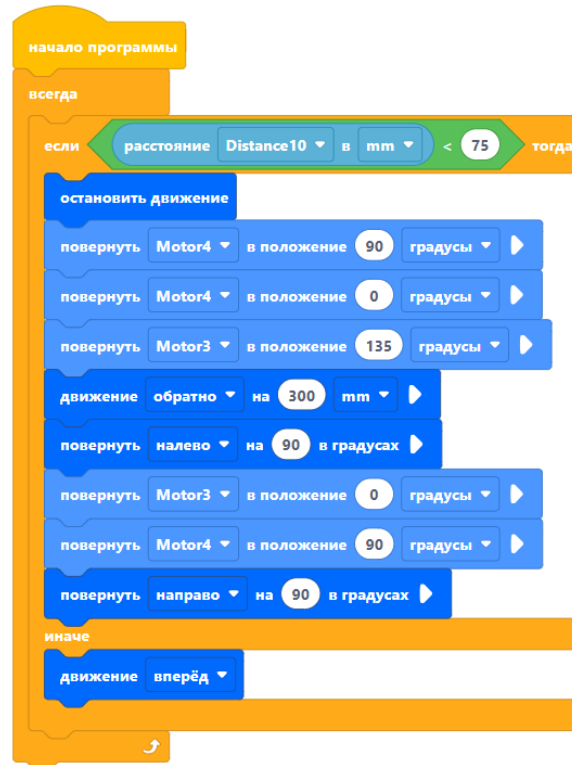
3. Спроектировать логику программы, включая команды для захвата, перемещения и освобождения объекта.
4. Загрузить программу на робота и протестировать захват объекта.
5. Оценить эффективность выполнения программы и провести необходимые корректировки.

Контрольные вопросы

1. Каковы основные функции клешни в робототехнике?
2. Какие факторы влияют на эффективность захвата объекта?
3. Как программа управления влияет на работу клешни?
4. В каком случае может происходить неудачный захват объекта?
5. Какие параметры следует учитывать при создании механизма захвата для различных объектов?

Приложение

Фрагмент кода (блочный язык программирования)



Фрагмент кода (C++)

```
float myVariable;
```

```
int whenStarted1() {
while (true) {
if (Distance10.objectDistance(mm) < 75.0) {
Drivetrain.stop();
Motor4.spinToPosition(90.0, degrees, true);
Motor4.spinToPosition(0.0, degrees, true);
Motor3.spinToPosition(135.0, degrees, true);
Drivetrain.driveFor(reverse, 300.0, mm, true);
Drivetrain.turnFor(left, 90.0, degrees, true);
Motor3.spinToPosition(0.0, degrees, true);
Motor4.spinToPosition(90.0, degrees, true);
Drivetrain.turnFor(right, 90.0, degrees, true);
}
}
```

```
else {
    Drivetrain.drive(forward);
}
wait(5, msec);
}
return 0;
}
```

```
int main() {
    calibrateDrivetrain();
    printf("\033[30m");
    wait(100, msec);
    whenStarted1();
}
```

Фрагмент кода (Python)

```
myVariable = 0
```

```
def when_started1():
    global myVariable
    while True:
        if distance_10.object_distance(MM) < 75:
            drivetrain.stop()
            motor_4.spin_to_position(90, DEGREES)
            motor_4.spin_to_position(0, DEGREES)
            motor_3.spin_to_position(135, DEGREES)
            drivetrain.drive_for(REVERSE, 300, MM)
            drivetrain.turn_for(LEFT, 90, DEGREES)
            motor_3.spin_to_position(0, DEGREES)
            motor_4.spin_to_position(90, DEGREES)
            drivetrain.turn_for(RIGHT, 90, DEGREES)
        else:
            drivetrain.drive(FORWARD)
            wait(5, MSEC)
```

```
calibrate_drivetrain()
when_started1()
```

Цель: освоить работу с датчиком цвета и научиться контролировать движение робота на основе цветowych сигналов.

Предметные результаты:

- знать основные принципы работы оптического датчика
- уметь настраивать датчик для распознавания цветowych меток в различных условиях освещения
- уметь создавать простую программу, которая будет управлять движением робота в зависимости от сигналов, получаемых с датчика цвета

Используемое оборудование и материалы:

- Набор VEX EXP с робототехническими компонентами.
- Компьютер со средой программирования VEXcode.
- USB-кабель для подключения контроллера к компьютеру.
- Датчик цвета.
- Датчики (при необходимости): датчик расстояния или касания.
- Цветные метки (картонные кружки, наклейки и прочие объекты разных цветов).
- Линейка для установки меток.



Теоретический материал



Оптический датчик представляет собой комбинацию следующих датчиков:

- **Датчик окружающего света** сообщает текущее количество окружающего света, которое обнаруживает датчик. Это могут быть уровни яркости в комнате или яркость определенного объекта.
- **Датчик цвета.** Информация о цвете доступна в виде RGB (красный, зеленый, синий), оттенка и насыщенности или оттенков серого. Распознавание цвета работает лучше всего, когда объект находится ближе, чем в 100 миллиметрах (мм).
- **Датчик приближения** измеряет отраженную ИК (инфракрасную) энергию, поступающую от встроенного ИК-светодиода. Таким образом, значения будут меняться в зависимости от окружающего освещения и отражательной способности объекта.

Оптический датчик также содержит белые светодиоды. Эти светодиоды можно включать и выключать или устанавливать на определенный процент яркости. Это обеспечивает постоянный источник света при обнаружении цветов независимо от окружающих условий освещения.

Оптический датчик улавливает световую энергию и преобразует её в электрические сигналы. Внутренняя электроника, называемая аппаратным конечным автоматом, обрабатывает эти сигналы и генерирует выходные сигналы, которые воспринимаются центральным контроллером как входные данные.

Датчик наилучшим образом распознает цвет, когда объект находится на расстоянии менее 10 см.

С помощью разработанной пользовательской программы оптический датчик может выполнять следующие задачи:

- Включать или выключать белые светодиоды датчика.
- Настраивать процент мощности белых светодиодов.
- Обнаруживать объекты.
- Определять цвет.
- Измерять уровень яркости окружающего света.
- Измерять цветовой оттенок в градусах.



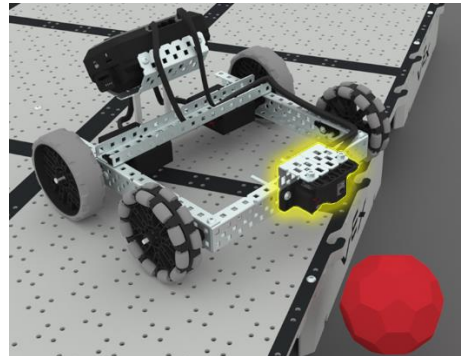
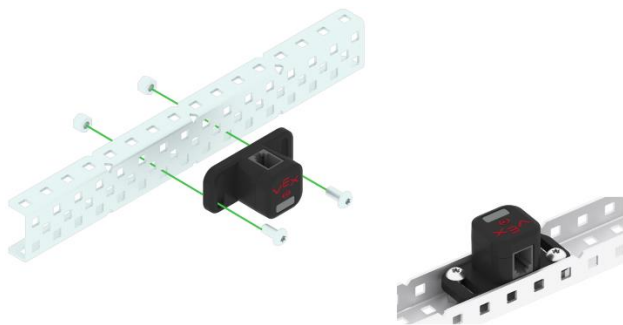
Считывание значений оптического датчика:

- Светодиод: текущий процент яркости светодиодов. 0 — выключен, 100% — полностью включен.
- BRT: процент яркости окружающего света в помещении или объекта.
- PROX: близость объекта, как близкого, так и далекого.
- Hue: значение оттенка в диапазоне от 0 до 359 градусов. Каждое значение оттенка имеет цветовую ассоциацию.

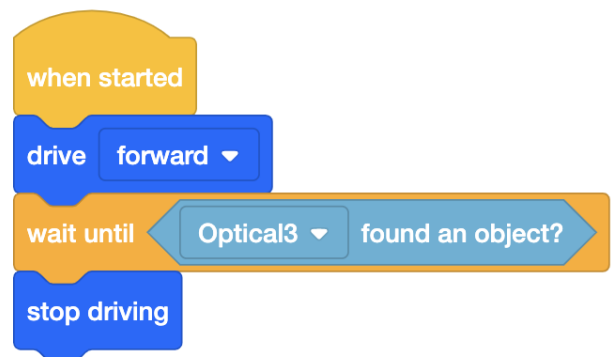
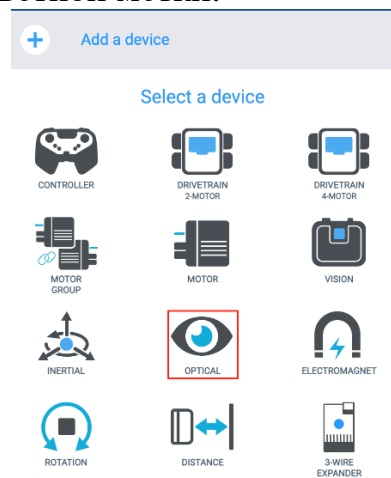
Практическая работа

Задания:

1. Установить оптический датчик на робота.

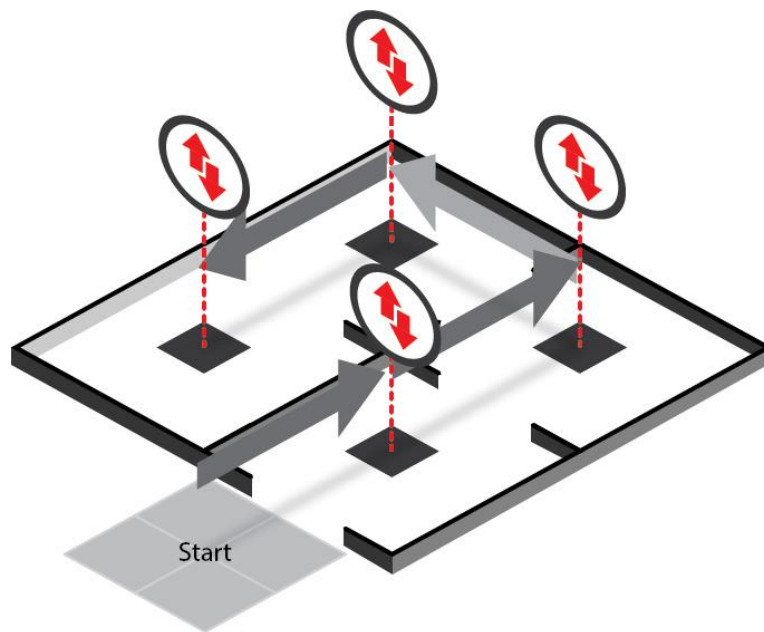


2. Запустить программное обеспечение и создать базовую программу для определения цветной метки.



3. Загрузить на робота созданный код, который будет обрабатывать данные от датчика цвета.
4. Оценить эффективность выполнения программы и провести необходимые корректировки.

Вариант расположения меток



Контрольные вопросы

1. Как работает датчик цвета и какие данные он предоставляет?
2. Какие ошибки могли произойти при тестировании робота?
3. Как можно улучшить программу для более точного считывания цветowych меток?
4. Какие практические применения имеют датчики цвета в робототехнике?

Приложение

Фрагмент кода (блочный язык программирования)



Фрагмент кода (C++)

```
float myVariable;

int whenStarted1() {
    Optical10.gestureDisable();
    while (true) {
```

```

if (Optical10.color() == green) {
  Drivetrain.stop();
  wait(1.0, seconds);
  Brain.playSound(alarm);
  wait(1.0, seconds);
}
else {
  Drivetrain.drive(forward);
}
wait(5, msec);
}
return 0;
}

```

```

int main() {
  calibrateDrivetrain();
  printf("\033[30m");
  wait(100, msec);
  whenStarted1();
}

```

Фрагмент кода (Python)

```
myVariable = 0
```

```

def when_started1():
    global myVariable
    optical_10.gesture_disable()
    while True:
        if optical_10.color() == Color.GREEN:
            drivetrain.stop()
            wait(1, SECONDS)
            brain.play_sound(SoundType.ALARM)
            wait(1, SECONDS)
        else:
            drivetrain.drive(FORWARD)
            wait(5, MSEC)

```

```

calibrate_drivetrain()
when_started1()

```

Цель: освоить работу с датчиком расстояния и научиться контролировать движение робота в лабиринте.

Предметные результаты:

- знать основные принципы работы датчика расстояния
- уметь создавать простую программу, которая будет управлять движением робота в лабиринте

Используемое оборудование и материалы:

- Набор VEX EXP с робототехническими компонентами.
- Компьютер со средой программирования VEXcode.
- USB-кабель для подключения контроллера к компьютеру.
- Датчик расстояния.
- Поле «Лабиринт».
- Линейка для измерений.



Теоретический материал



Датчик расстояния - это устройство, которое используется для измерения расстояния до объекта или препятствия в окружающей среде. Программирование робота с использованием датчика расстояния позволяет реализовать различные функции и повысить его автономность.

Один из наиболее распространенных типов датчиков расстояния - ультразвуковой датчик. Он работает по принципу отправки звукового импульса и измерения времени, за которое отраженный сигнал вернулся обратно. На основании этого времени можно определить расстояние до объекта.

Датчик расстояния имеет следующие возможности:

- Измерение расстояния. Датчик использует импульс безопасного для класса лазерного света для измерения расстояния от передней части датчика до объекта. Расстояние отображается в дюймах или сантиметрах на панели Brain's Sensor Dashboard, а также в дюймах или миллиметрах в VEXcode EXP.
- Обнаружение объекта. Датчик также можно использовать для обнаружения находящегося рядом объекта.
- Определить относительный размер объекта. Датчик также может быть использован для определения относительного размера обнаруженного объекта. Приблизительный размер объекта сообщается как маленький, средний или большой.
- Отчет о скорости объекта. Датчик можно использовать для расчета и сообщения скорости в метрах в секунду для объекта, приближающегося к датчику, или для датчика, приближающегося к объекту.

Программирование робота с использованием датчика расстояния позволяет следующее:

- Избегание столкновений: робот может реагировать на препятствия, измерять расстояние до них и изменять направление движения.
- Навигация по преградам: робот может автоматически обходить препятствия, используя данные с датчика расстояния для определения оптимального маршрута.
- Точное позиционирование: датчик расстояния может использоваться для точного перемещения робота в определенное место или для выхода из сложных местоположений.

Датчик расстояния посылает импульс безопасного для класса лазерного света и измеряет время, необходимое для отражения импульса. Это позволяет рассчитать расстояние.

Лазер класса 1 датчика похож на лазеры, используемые в современных сотовых телефонах для обнаружения головы. Лазер позволяет датчику иметь очень узкое поле зрения, поэтому обнаружение всегда происходит прямо перед датчиком.

Диапазон измерения датчика составляет от 2 см до 200 см.



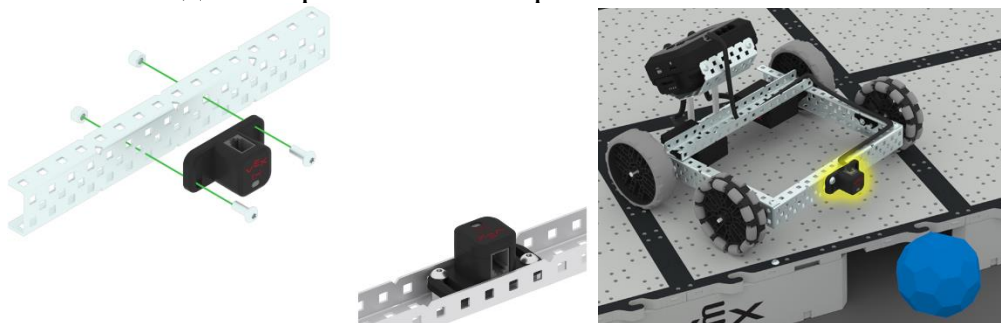
Считывание значений датчика расстояния. На Sensor Dashboard панель Distance Sensor сообщает расстояние до ближайшего объекта в дюймах или сантиметрах. Единицы измерения можно изменить, нажав кнопку Check для переключения между дюймами и сантиметрами.

Использование датчика расстояния в программировании робота позволяет значительно расширить его функционал и возможности в автономном режиме работы. Роботы с датчиками расстояния могут быть эффективно задействованы в различных областях, где требуется точное движение и взаимодействие с окружающей средой.

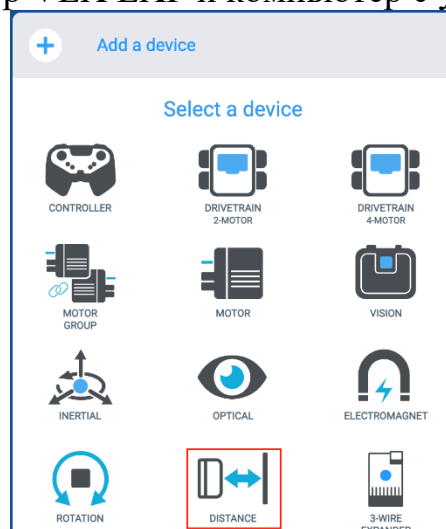
Практическая работа

Задания:

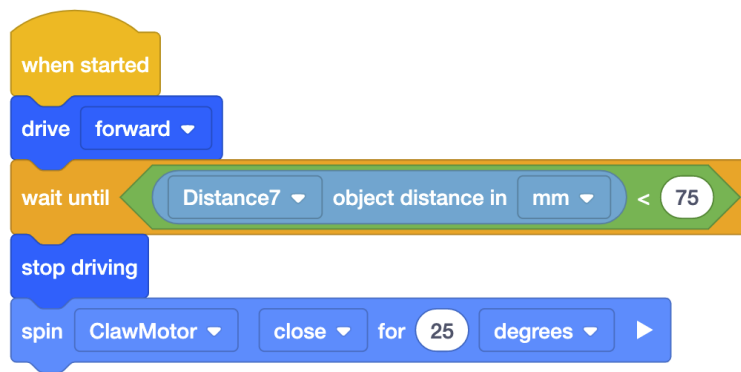
1. Установить датчик расстояния на робота.



2. Подготовить набор VEX EXP и компьютер с установленным VEXcode.



3. Подключить программу к контроллеру робота с помощью USB-кабеля.
4. Написать программу для движения робота движения и выхода из лабиринта с использованием датчика расстояния.



5. Провести тестирование работы программы на работе.

Контрольные вопросы

1. Каким образом можно использовать датчик расстояния в программе для робота?
2. Какие преимущества предоставляет использование датчиков при программировании роботов в VEX EXP?
3. Как может улучшить взаимодействие вашего устройства с окружающей средой объединение датчика расстояния с датчиком касания?

Приложение

Фрагмент кода (блочный язык программирования)



Фрагмент кода (C++)

```
float myVariable;

int whenStarted1() {
  Drivetrain.turnFor(right, 90.0, degrees, true);
  while (true) {
    if (Distance10.objectDistance(mm) > 100.0) {
      Drivetrain.driveFor(forward, 250.0, mm, true);
      Drivetrain.turnFor(right, 90.0, degrees, true);
    }
    if (Distance10.objectDistance(mm) < 100.0) {
      Drivetrain.turnFor(left, 90.0, degrees, true);
    }
  }
}
```

```
}  
wait(5, msec);  
}  
return 0;  
}  
  
int main() {  
    calibrateDrivetrain();  
    printf("\033[30m");  
    wait(100, msec);  
    whenStarted1();  
}
```

Фрагмент кода (Python)

```
myVariable = 0
```

```
def when_started1():  
    global myVariable  
    drivetrain.turn_for(RIGHT, 90, DEGREES)  
    while True:  
        if distance_10.object_distance(MM) > 100:  
            drivetrain.drive_for(FORWARD, 250, MM)  
            drivetrain.turn_for(RIGHT, 90, DEGREES)  
        if distance_10.object_distance(MM) < 100:  
            drivetrain.turn_for(LEFT, 90, DEGREES)  
        wait(5, MSEC)
```

```
calibrate_drivetrain()  
when_started1()
```

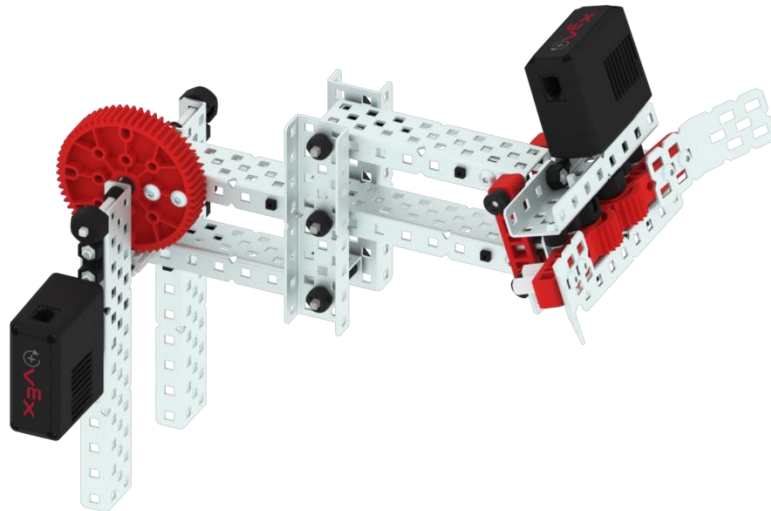
Цель: освоить принципы сборки и функционирования различных вариантов подъемных систем для мобильного робота, а также изучение их влияния на эффективность работы устройства.

Предметные результаты:

- знать принципы работы и управления подъемными механизмами
- уметь осуществлять сборку и настройку подъемных систем для мобильных роботов

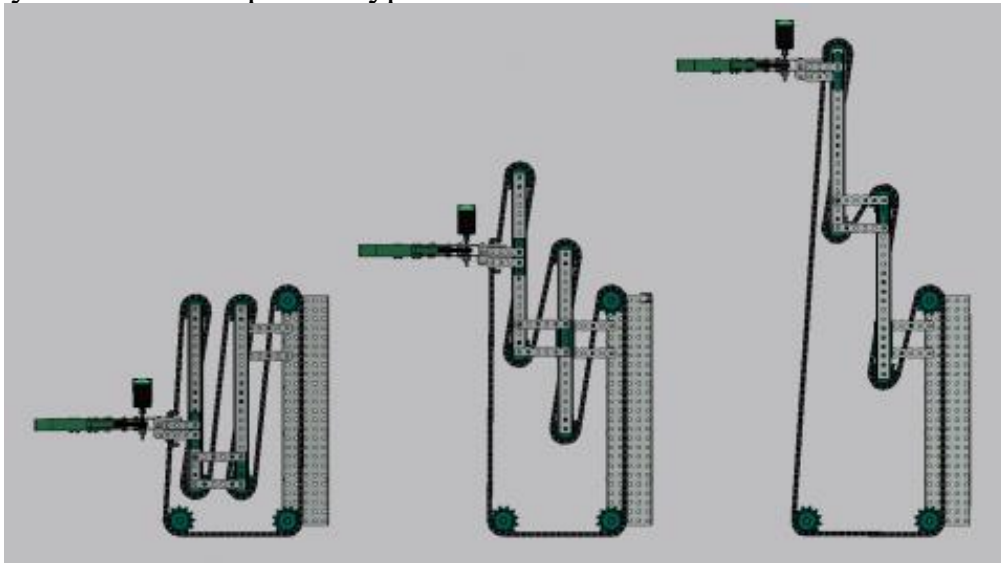
Используемое оборудование и материалы:

- Набор VEX EXP с робототехническими компонентами.
- Компьютер со средой программирования VEXcode.
- USB-кабель для подключения контроллера к компьютеру.
- Датчики (при необходимости).



Теоретический материал

Подъемные системы играют ключевую роль в обеспечении функциональности мобильных роботов, особенно в контекстах, связанных с манипуляцией объектами, их переноской и установкой на разные уровни.



Подъемные системы работают на основе преобразования энергии в механическую работу. Основные принципы:

- **Закон сохранения энергии.** Энергия не создается и не уничтожается, а только преобразуется из одной формы в другую. При подъеме груза подъемный механизм преобразует электрическую энергию в потенциальную.
- **Механические преимущества.** Подъемные механизмы могут использовать механические преимущества для уменьшения силы, необходимой для подъема объекта.
- **Измерение нагрузок.** Использование тензодатчиков позволяет оценивать вес поднимаемого объекта и улучшать управление системой.

Типы подъемных систем:

- **Механические подъемники:** Используют разнообразные механизмы, такие как рычаги, ролики и блоки для обеспечения подъема. Примером является система с двумя или более рычагами, где один рычаг усиленно поднимает другой.
- **Сервоприводы:** Широко применяемые в робототехнике, сервомоторы обеспечивают высокую точность управления углом поворота и могут использоваться для точного подъема тяжелых грузов.
- **Шаговые двигатели:** Используются для задач, требующих точного управления положением и скорости. Шаговые двигатели могут обеспечивать высокую точность управления, что делает их идеальными для систем, где необходима точность.
- **Пневматические и гидравлические механизмы:** Эти системы используют давление газов или жидкостей для производства силы. Они способны поднимать тяжелые грузы, но требуют более сложного оборудования и контроля.

Практическая работа

Задания:

1. Собрать и установить подъемную систему (на выбор) на мобильный робот.
2. Подключить все необходимые датчики и моторы.
3. Программировать контроллер для управления подъемом и опусканием груза.
4. Запустить тестирование различных сценариев работы подъемного механизма.
5. Записать данные о времени подъема, нагрузках и успешности выполнения задач.

Контрольные вопросы

1. Какие факторы влияют на эффективность подъемного механизма?
2. Как можно улучшить работу подъемной системы?
3. В чем различия между различными типами подъемных механизмов?
4. Какие ошибки могут возникнуть при работе с подъемными системами и как их избежать?

Варианты сборок подъемных механизмов

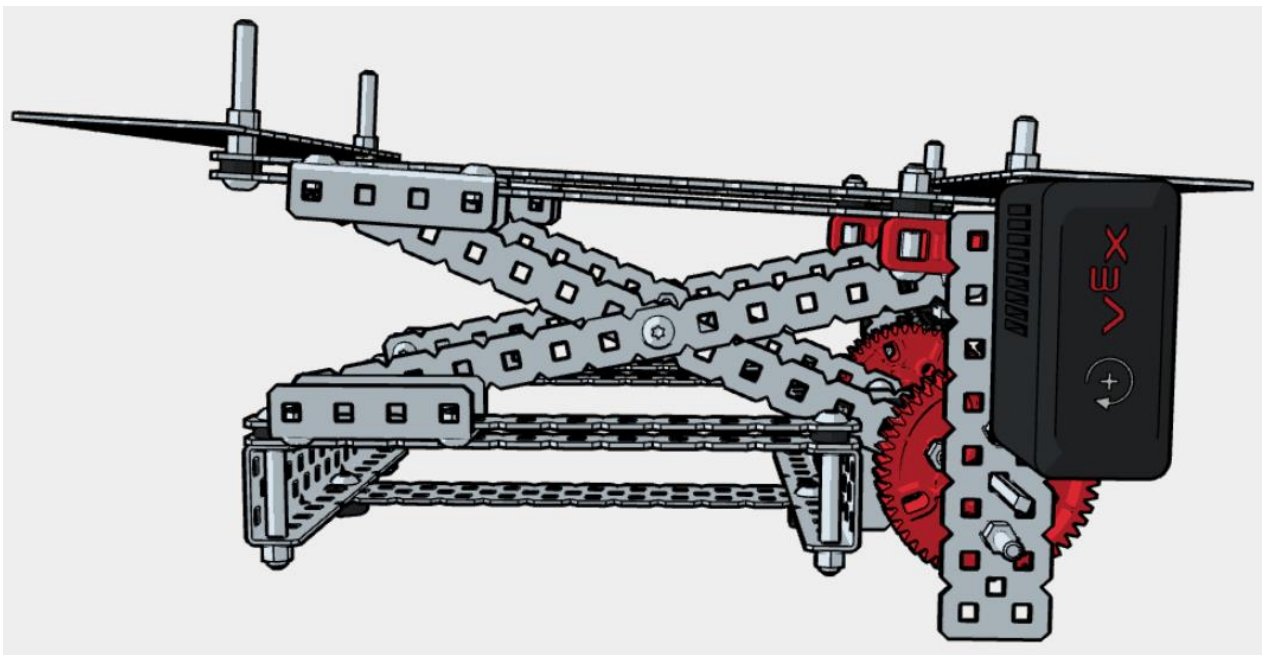
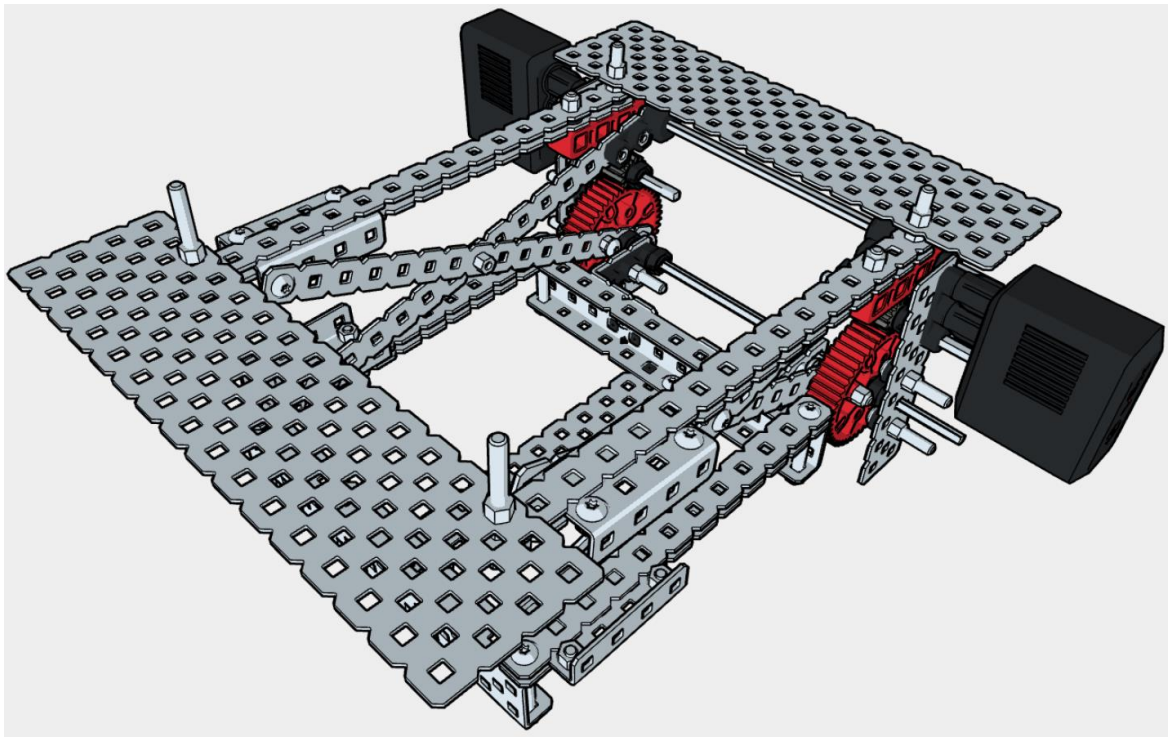
1. Ножничные подъемники

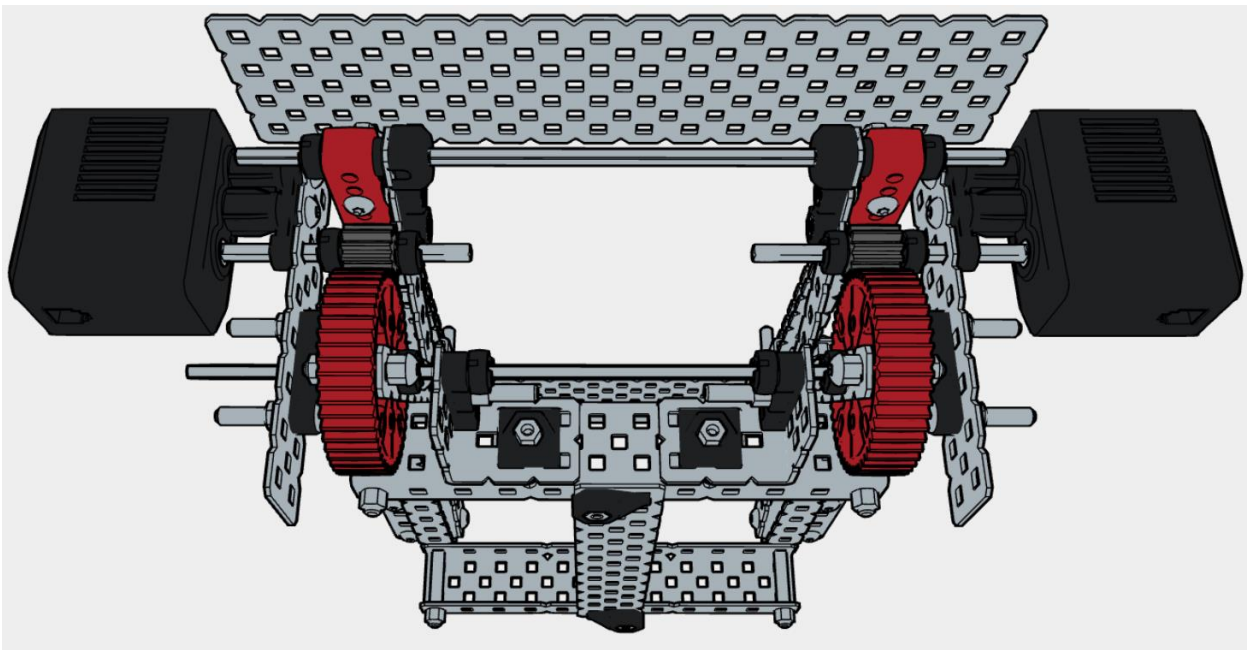
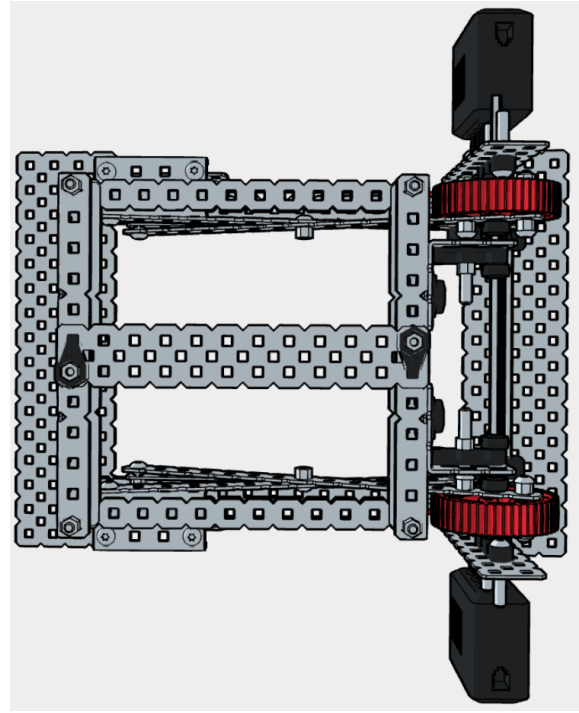
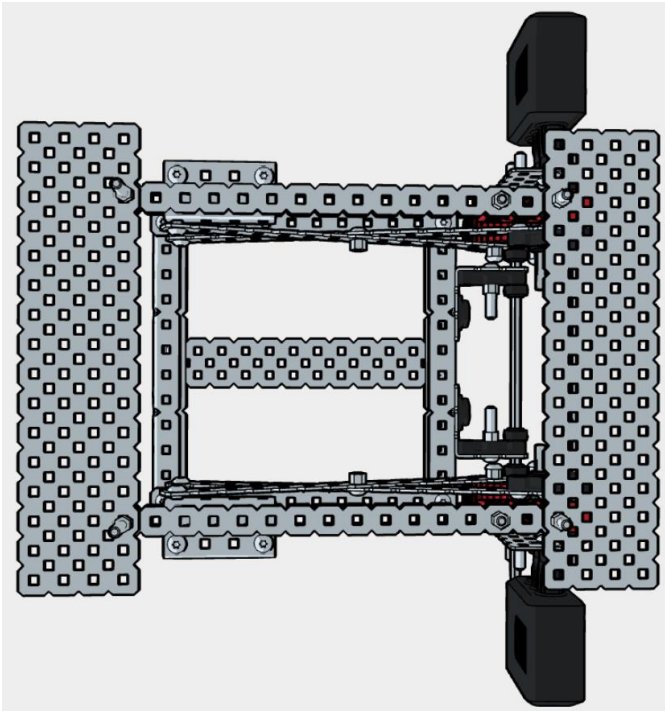
Ножничные подъемники делают, создавая точку поворота в середине двух перекрещивающихся металлических деталей. Обычно один конец одной детали крепится к точке поворота на основе подъемника, а конец другой детали может двигаться к первому концу по этой основе. Это приводит к тому, что подъемник поднимается, сводя детали вместе. Подъемники почти всегда делают в паре, чтобы уравновесить силы.

Верхняя часть подъемника обычно имеет платформу, которая крепится аналогично нижней. Один край платформы прикреплен к точке поворота, а другой может скользить. Подъемник может работать с помощью большой шестерни (84 зубца), которая находится в точке поворота между двумя металлическими деталями. Эта шестерня приводится в движение меньшей шестерней (12 зубцов), которая прикреплена к двигателю. Меньшая шестерня приводит в движение большую шестерню, что поднимает подъемник.

Существует и другой способ поднятия ножничного подъемника: он использует цилиндрическую шестерню на двигателе и реечную передачу, чтобы тянуть одну сторону ножничного механизма к точке поворота. Ножничные подъемники часто нуждаются в дополнительных поперечных опорах для стабильности.

Чем больше секций ножниц сложено друг на друга, тем выше может подняться подъемник, но для этого нужно будет приложить больше усилий, и его сложнее стабилизировать. Когда ножничный подъемник опущен ниже, его сложнее поднять. Из-за множества соединений внутри и различных сил, действующих на подъемник, его создание может быть довольно сложным, чтобы он работал эффективно.



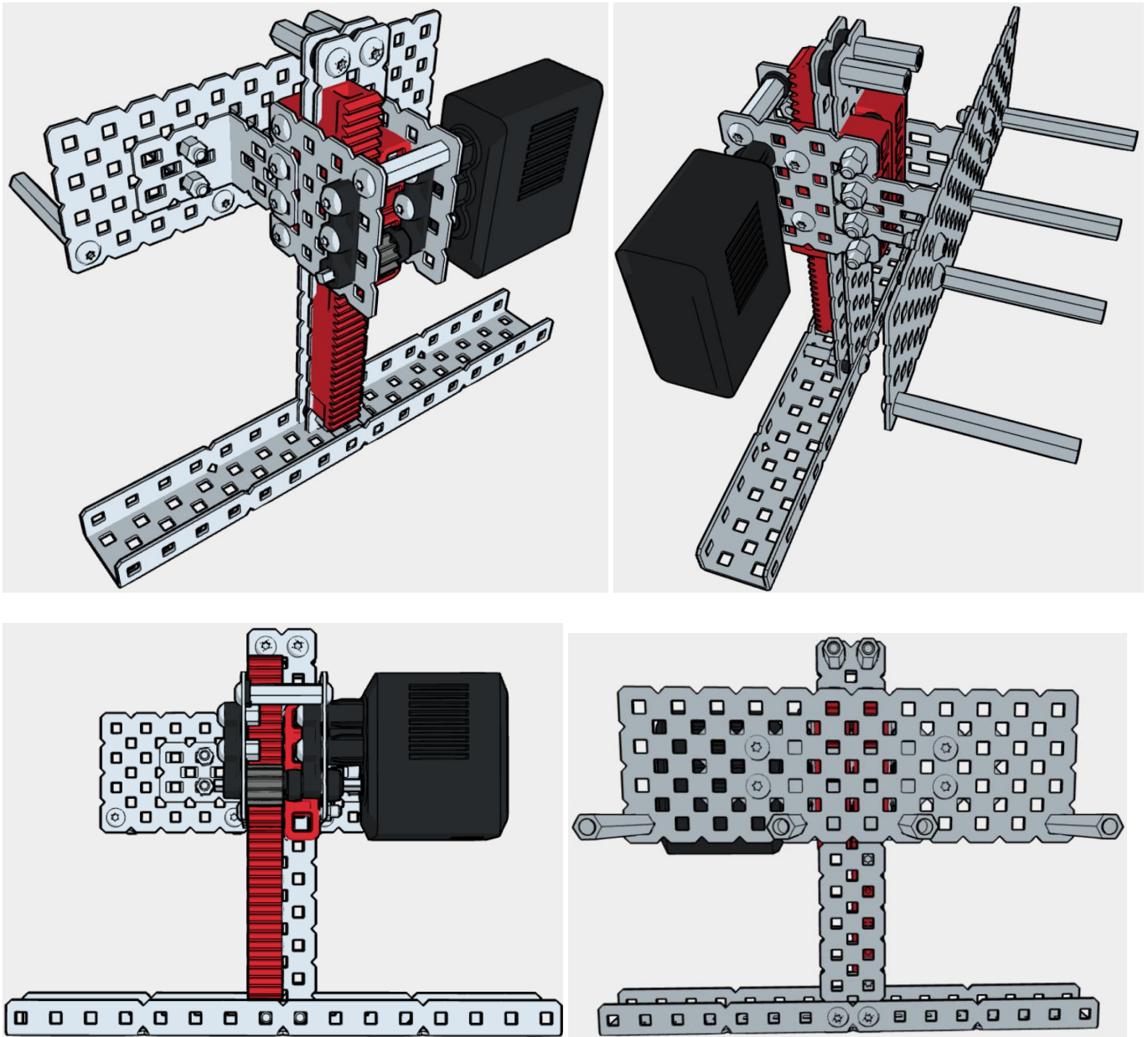


2. Линейные слайды

Линейные направляющие собираются с помощью зубчатых реек и цилиндрических шестерен, которые входят в комплект EXP. Чтобы создать эффективный самодельный комплект для линейного перемещения, нужно использовать детали из этого набора. Зубчатые рейки крепятся на плоской балке, а дополнительные стойки и защелкивающиеся распорки прикрепляют другую плоскую балку сзади первой. Затем соединительные балки помогают поднимать подъемник.

В конце концов, двигатель устанавливается в специальный кронштейн для зубчатой рейки, на валу которого находится цилиндрическая шестерня. Это позволяет кронштейну рейки двигаться вверх и вниз по направляющей, когда цилиндрическая

шестерня движется по зубчатой рейке. Плоские балки обычно фиксируются к основной части конструкции (шасси), а к кронштейну рейки можно присоединить манипулятор, дополнительную зубчатую рейку или платформу.

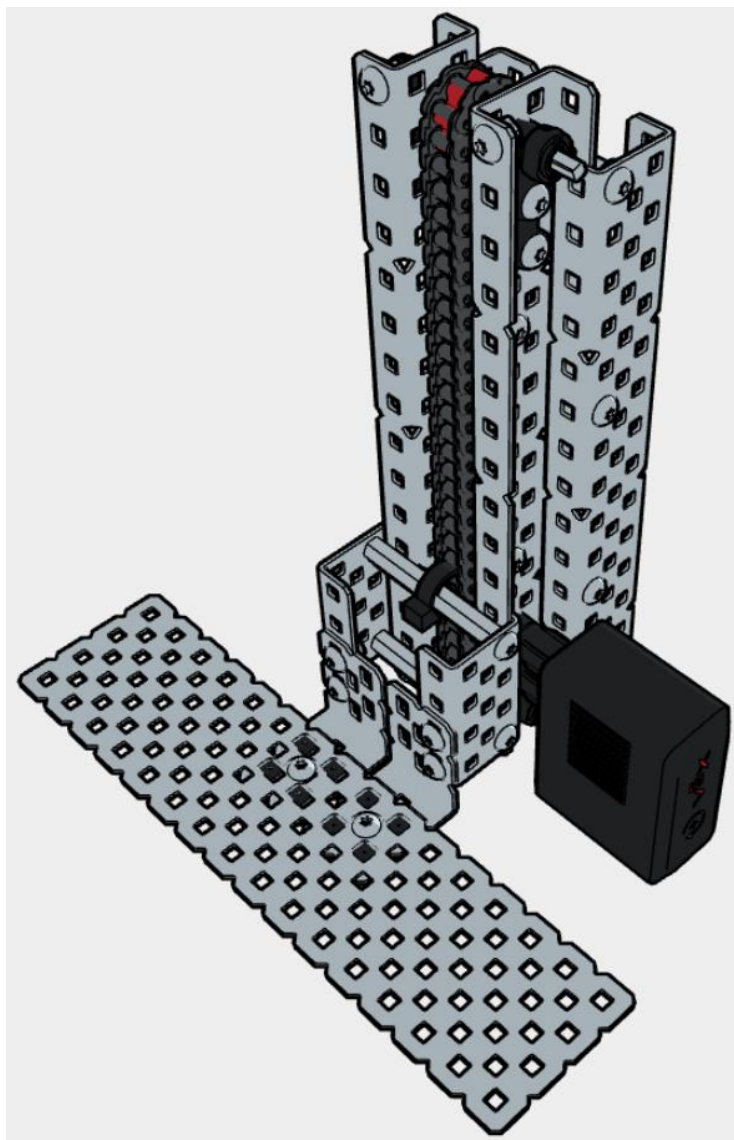


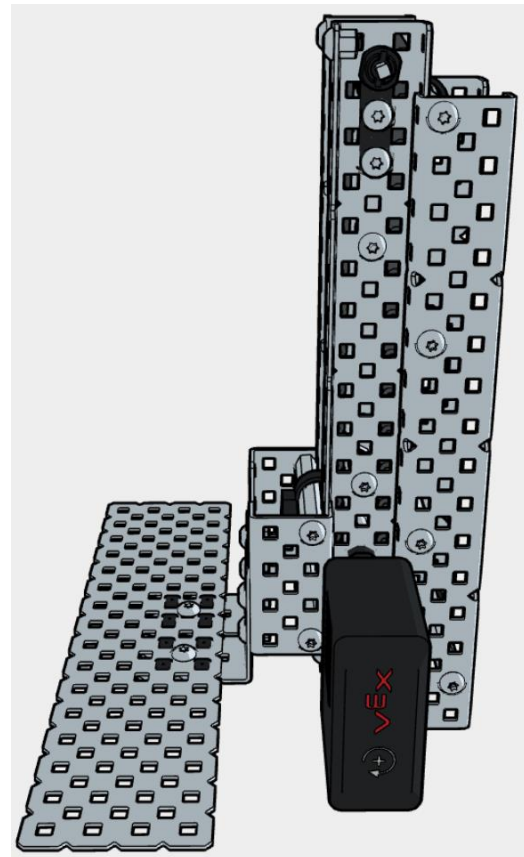
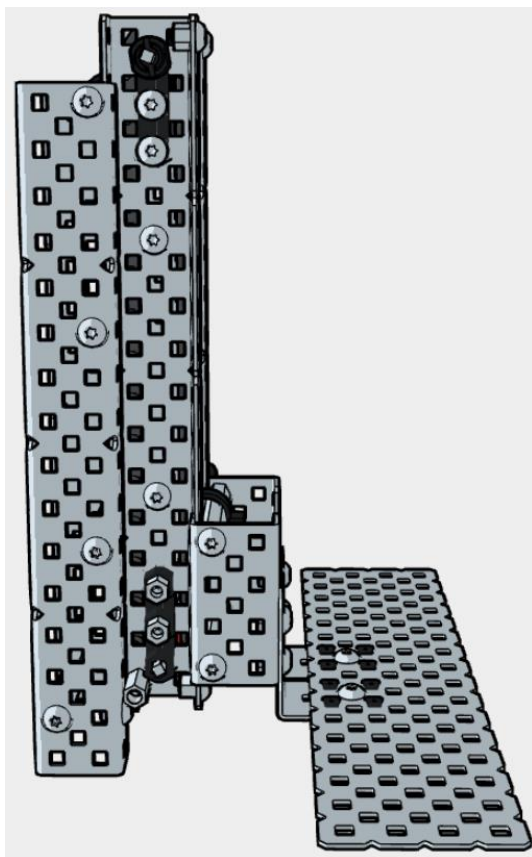
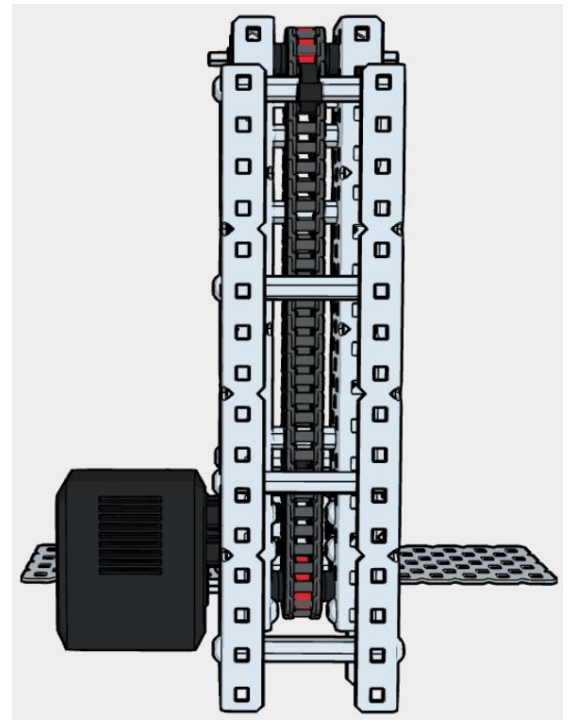
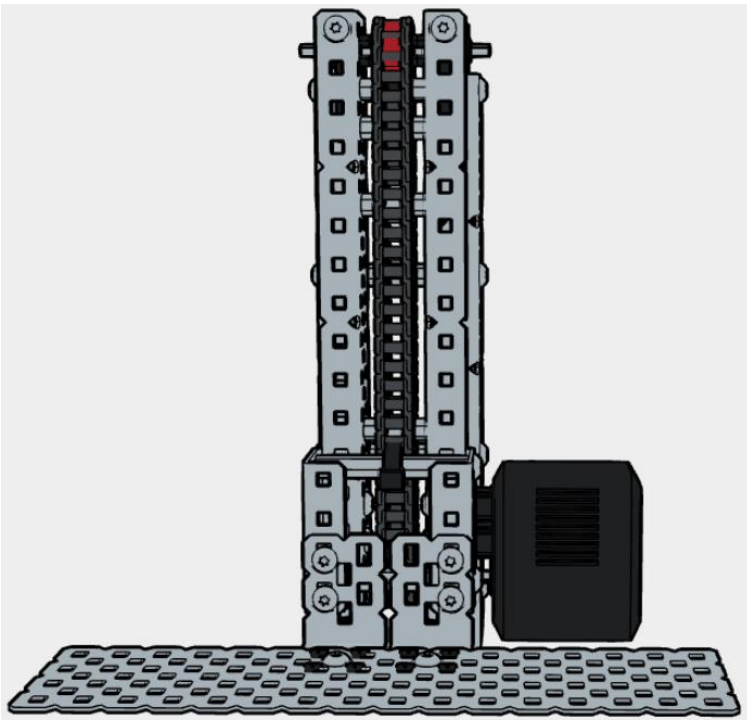
3. Телескопические подъемники

Телескопические подъемники собираются из вложенных С-образных каналов, которые могут вставляться друг в друга. Они похожи на каскадные подъемники, потому что тоже используют скользящие С-образные каналы.

Работа подъемника начинается с того, что штифтовая цепная связь устанавливается в верхней части сборки из С-образных каналов. Основной С-образный канал крепится к шасси (базе) подъемника. Двигатель устанавливают на первой секции подъемника, и он приводит в движение цепь. За счет штифтовой связи возникает сила, которая поднимает подъемник вверх.

Часто в телескопических подъемниках также используются лебедочные механизмы и блоки для упрощения работы. Однако эта конструкция как раз отвечает содержанию комплекта, который используется для создания подъемника. Сборку таких подъемников можно повторять, что позволяет им легко и быстро подниматься на большую высоту.





4. Каскадные лифты

Каскадные подъемники работают похоже на телескопические, так как в версии VEX EXP используются цепи и скользящие С-образные каналы. Иногда их называют цепными подъемниками. Обычно такие подъемники собираются парами, чтобы силы распределялись равномерно.

Сборка каскадных подъемников происходит так: две пары С-образных каналов вставляются друг в друга, и один набор компонент крепится к шасси с помощью системы цепей и звездочек. Затем к этой системе крепится вторая пара С-образных каналов. Когда один или несколько двигателей приводят в движение цепь и звездочки, вторая пара С-образных каналов поднимается вверх по первой благодаря цепной связи, которая надежно закреплена.

Эту конструкцию можно повторять, что позволяет подъемнику подниматься очень высоко. Для того чтобы секции С-образных каналов плотно входили друг в друга и могли двигаться вверх и вниз, необходимы распорки.

