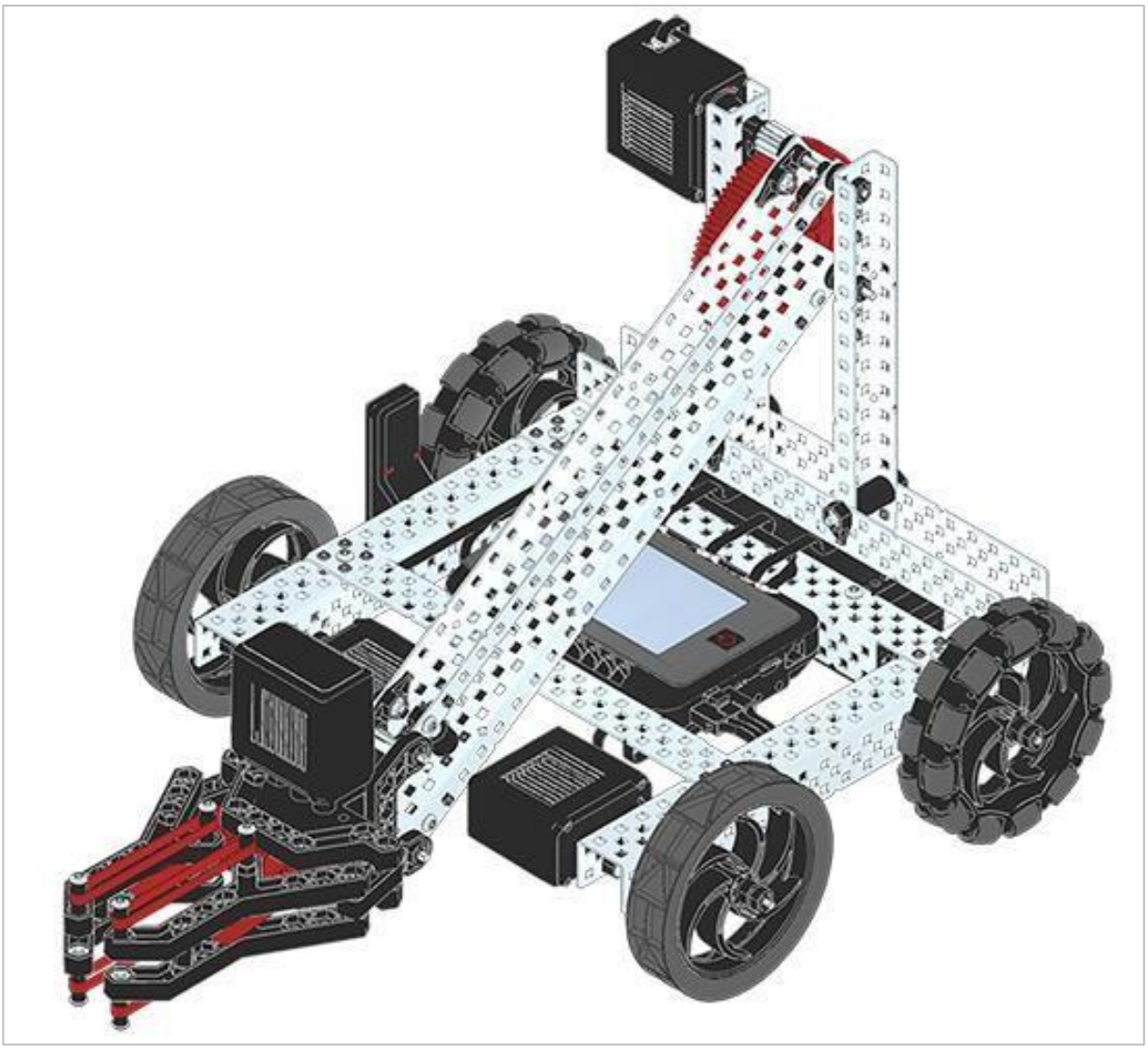
# Pętla, oto ona!

Dowiedz się, jak programować pętle, aby robot mógł bujać się do rytmu!


**Odkryj nowe praktyczne kompilacje i możliwości programowania, aby zrozumieć temat.**

# Kompletny VEX V5 Clawbot



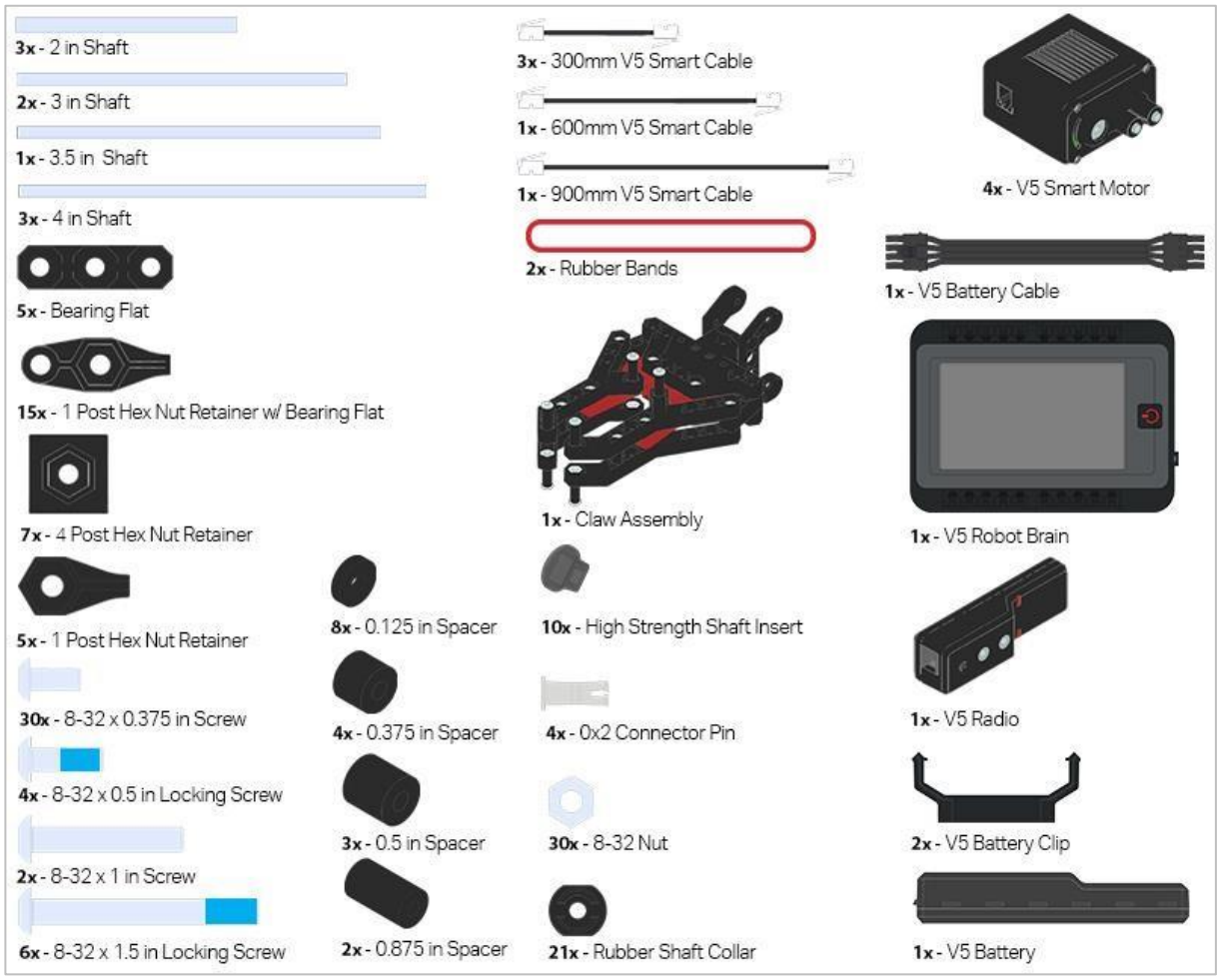
*Kompletny VEX V5 Clawbot*

VEX V5 Clawbot to rozszerzenie VEX Speedbot'a, które można zaprogramować do poruszania się i interakcji z obiektami.

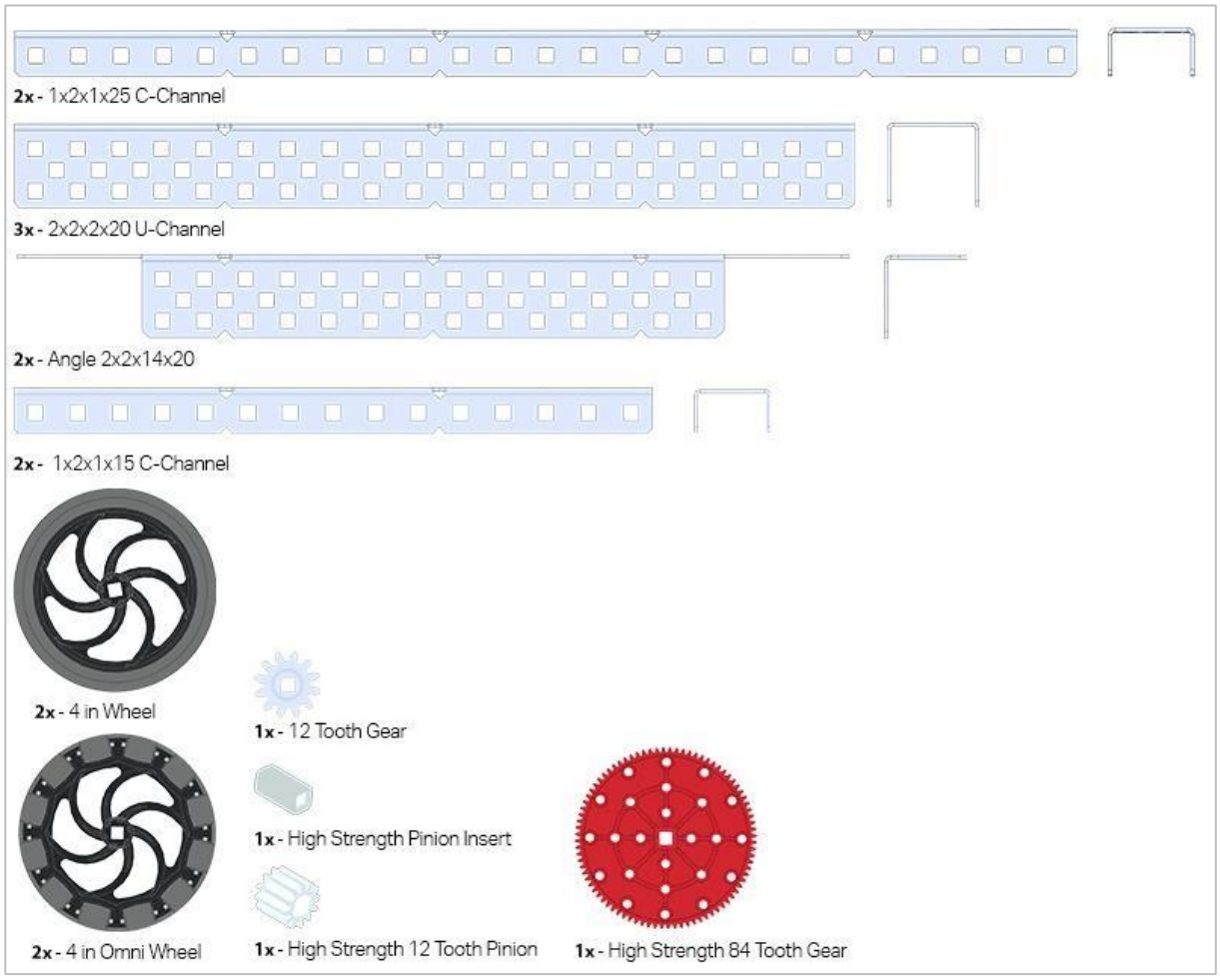
# Potrzebne elementy: część 1

**Może być zbudowany z:**

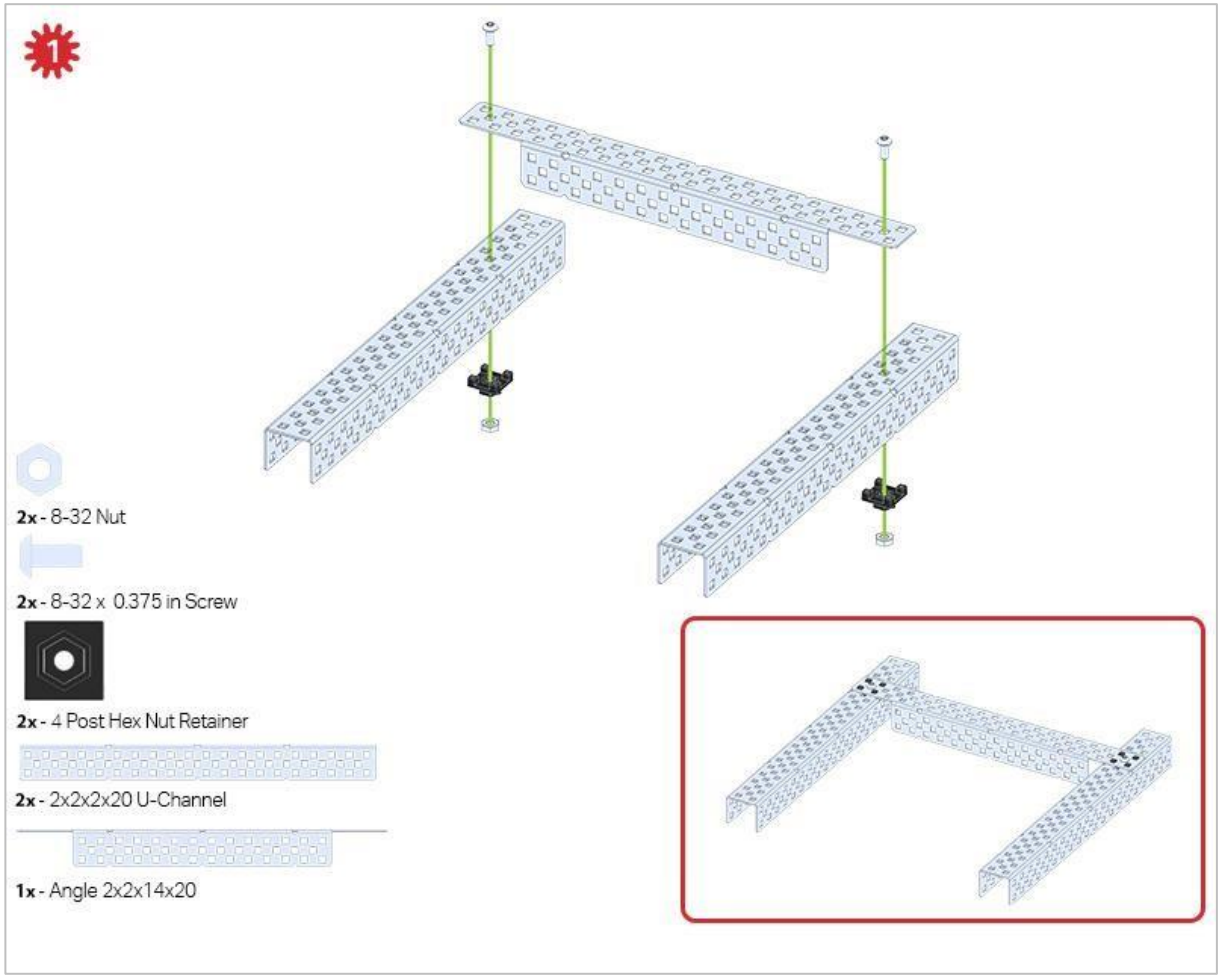
VEX V5 Classroom Starter Kit



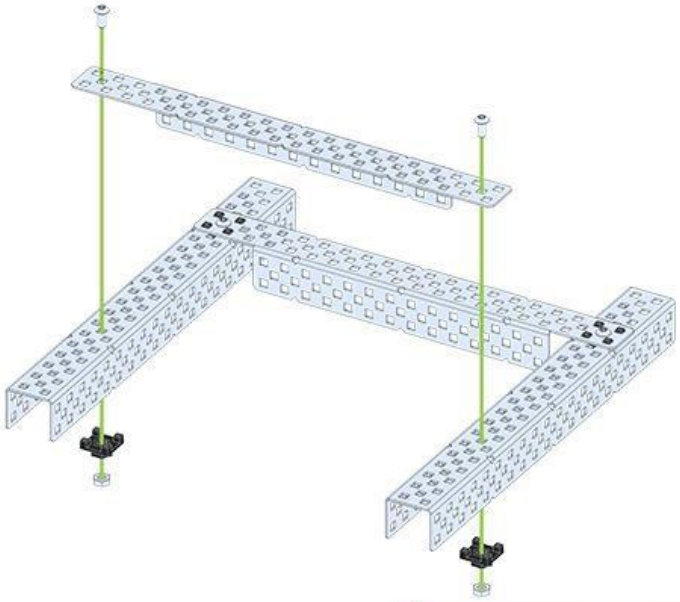
# Potrzebne elementy: część 2



# Instrukcje budowania



2





**2x** - 8-32 Nut

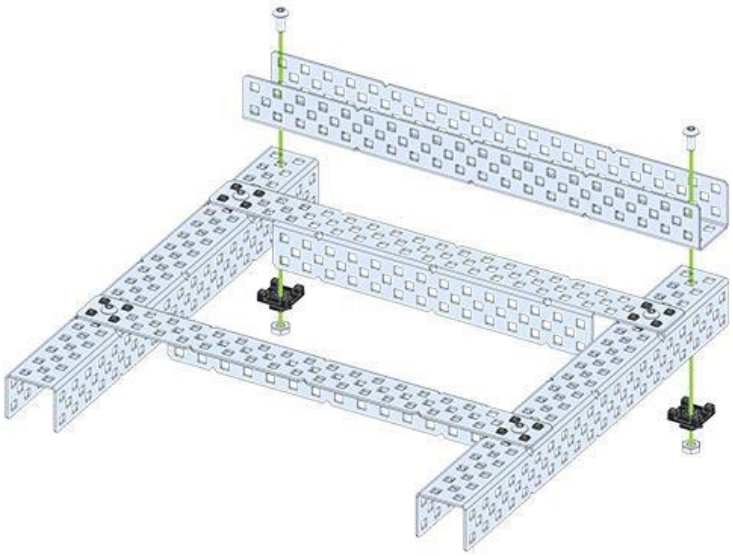


**2x** - 8-32 x 0.375 in Screw

**2x** - 4 Post Hex Nut Retainer

**1x** - Angle 2x2x14x20

3

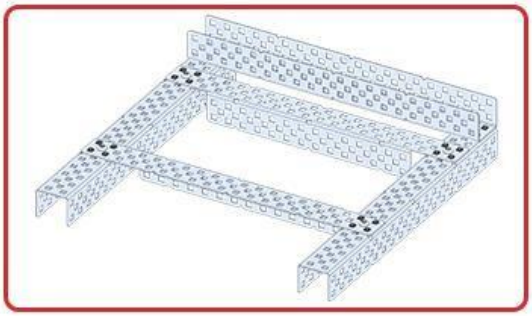


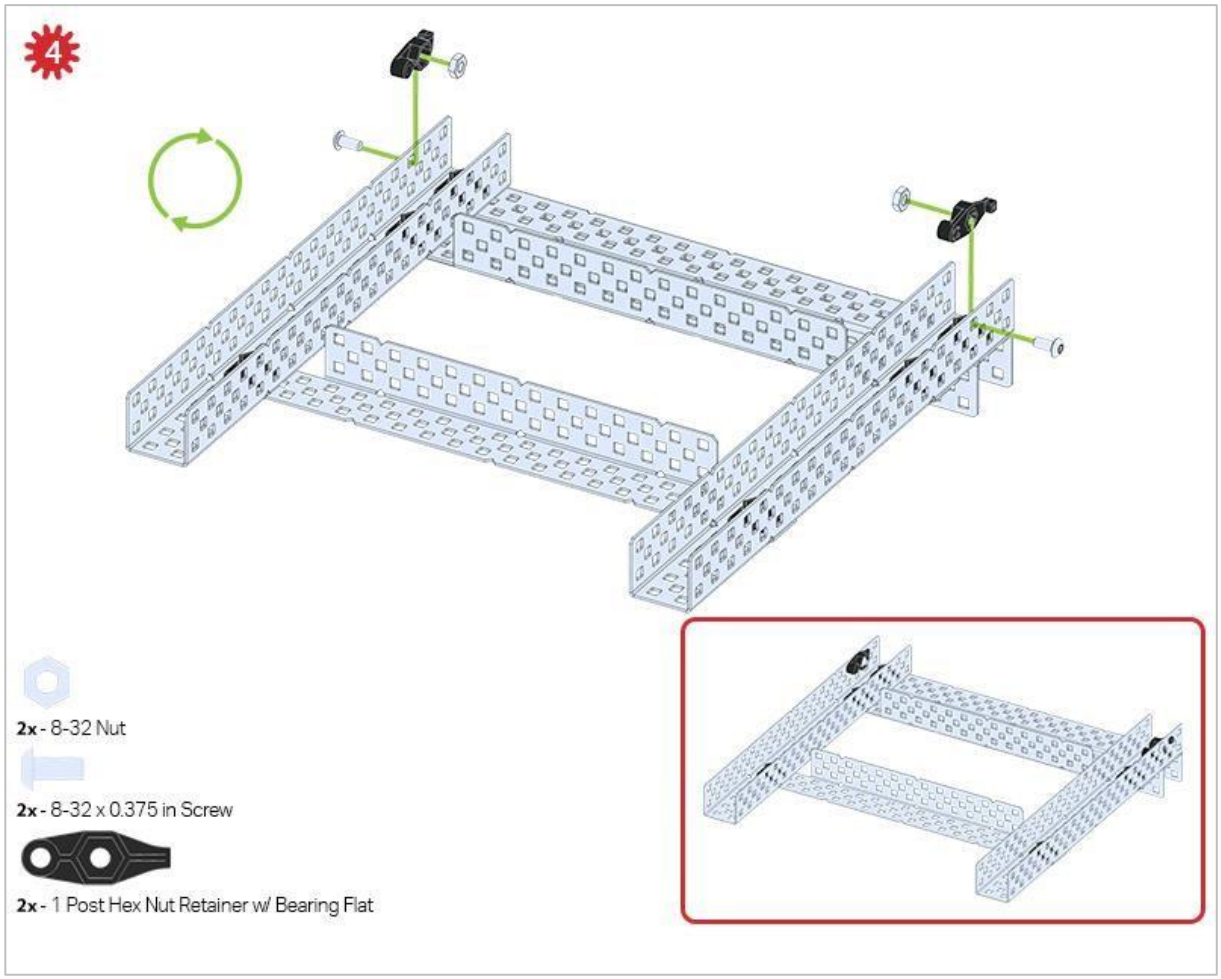
**2x** - 8-32 Nut

**2x** - 8-32 x 0.375 in Screw

**2x** - 4 Post Hex Nut Retainer

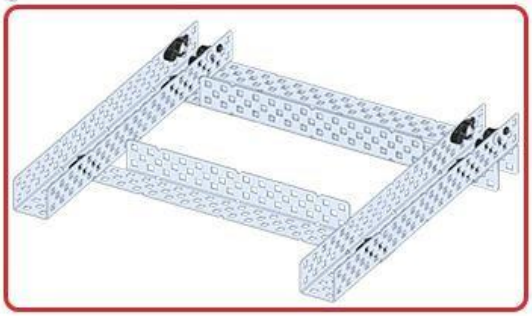
**1x** - 2x2x2x20 U-Channel






*Zielona ikona wskazuje, że konstrukcj musi zostać odwrócona (do góry nogami).*

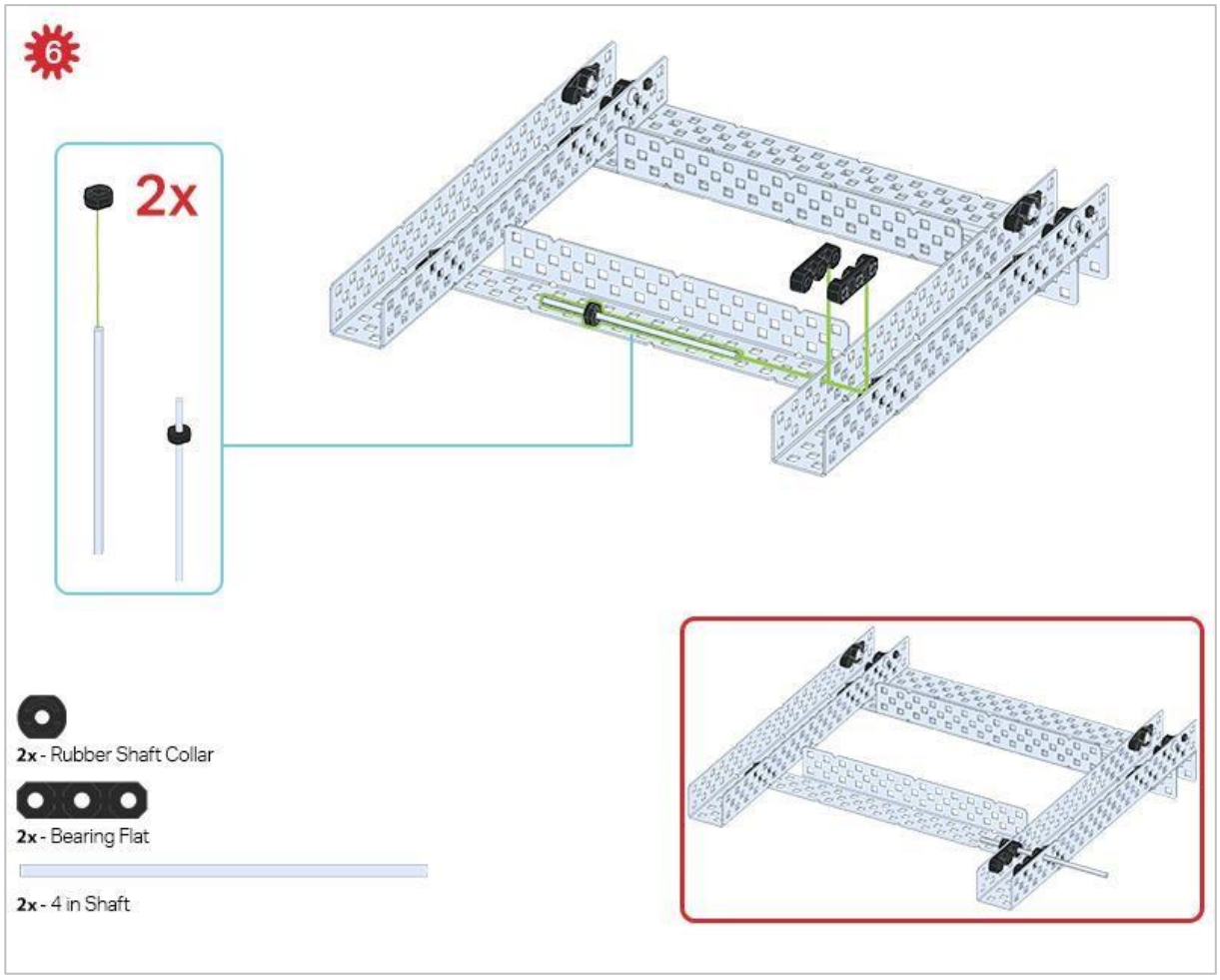
5



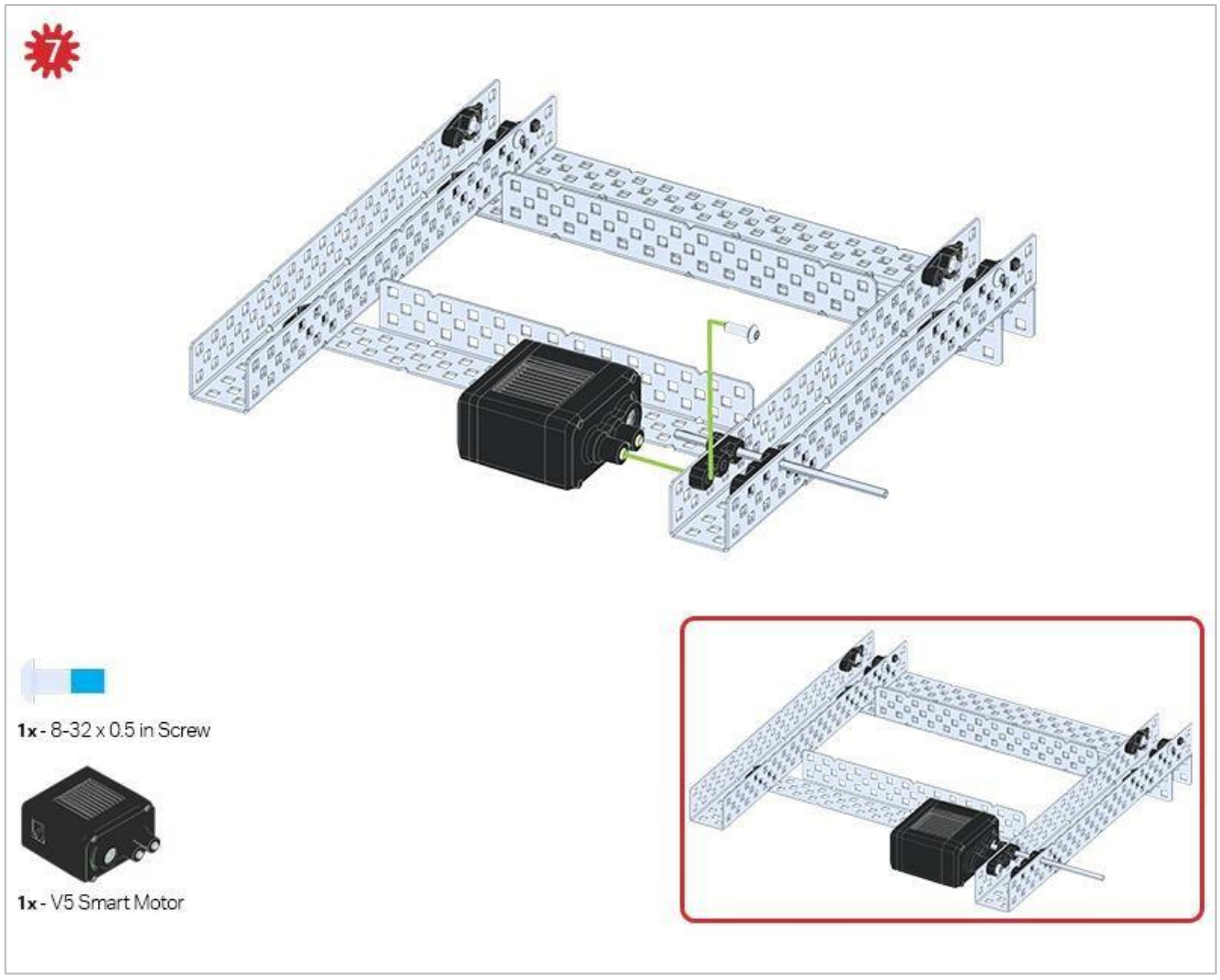
**2x** - 8-32 Nut

**2x** - 8-32 x 0.375 in Screw

**2x** - 1 Post Hex Nut Retainer w/ Bearing Flat

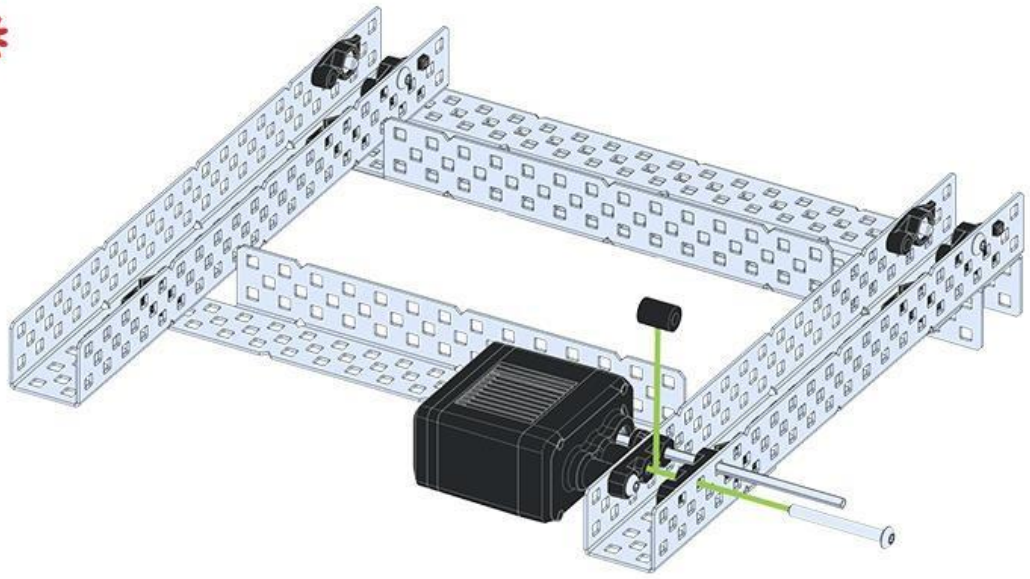



*Obecnie używany jest tylko jeden z dwóch podzespołów wykonanych w tym kroku. Drugi zostanie użyty później w kroku 9.*



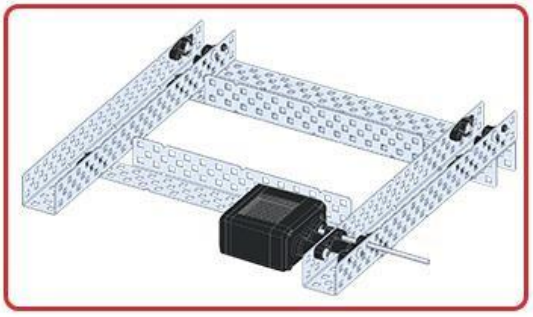

*Upewnij się, że silniki są ustawione we właściwym kierunku (otwory na śruby skierowane na zewnątrz konstrukcji, a otwór wału do wewnątrz).*

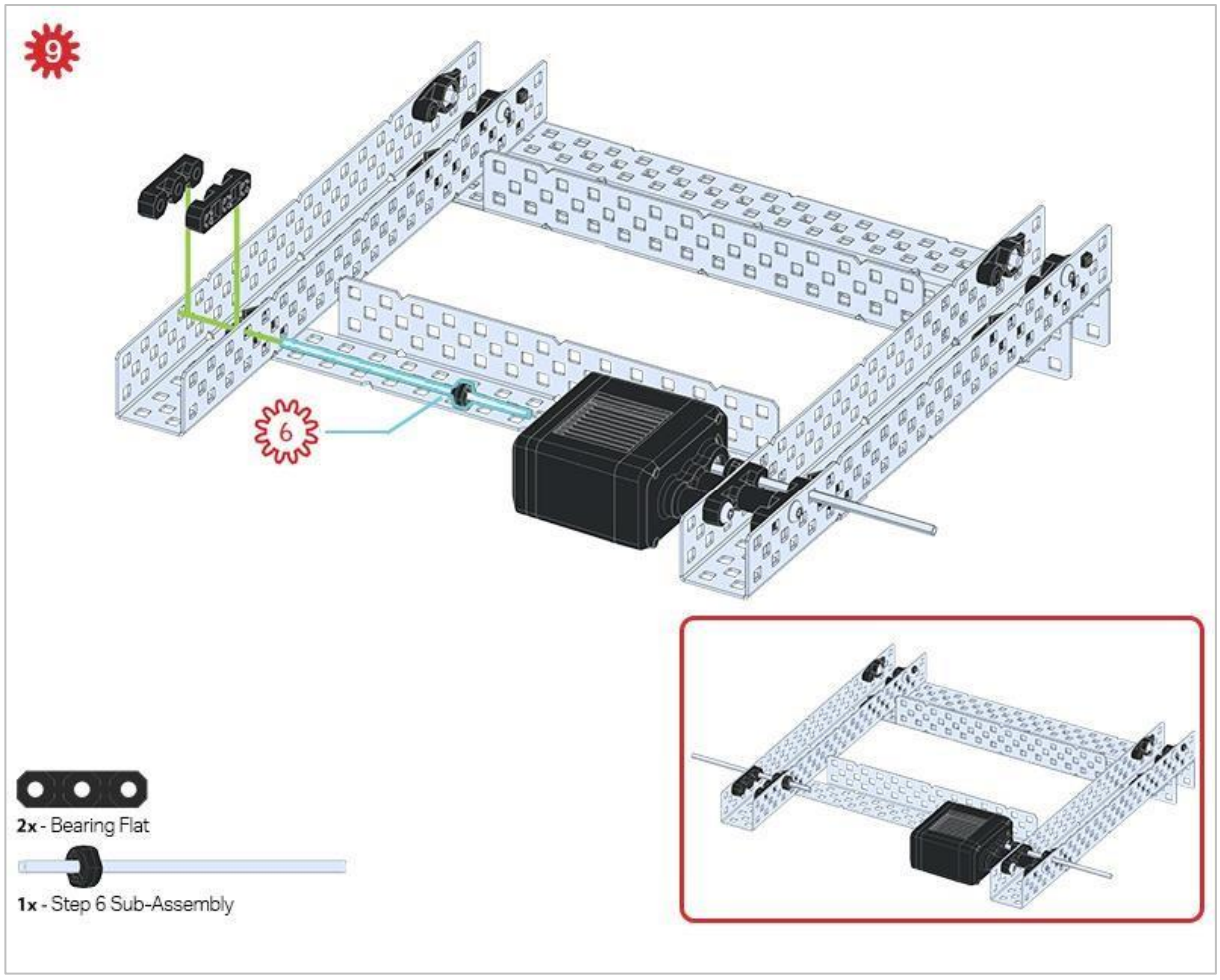
8

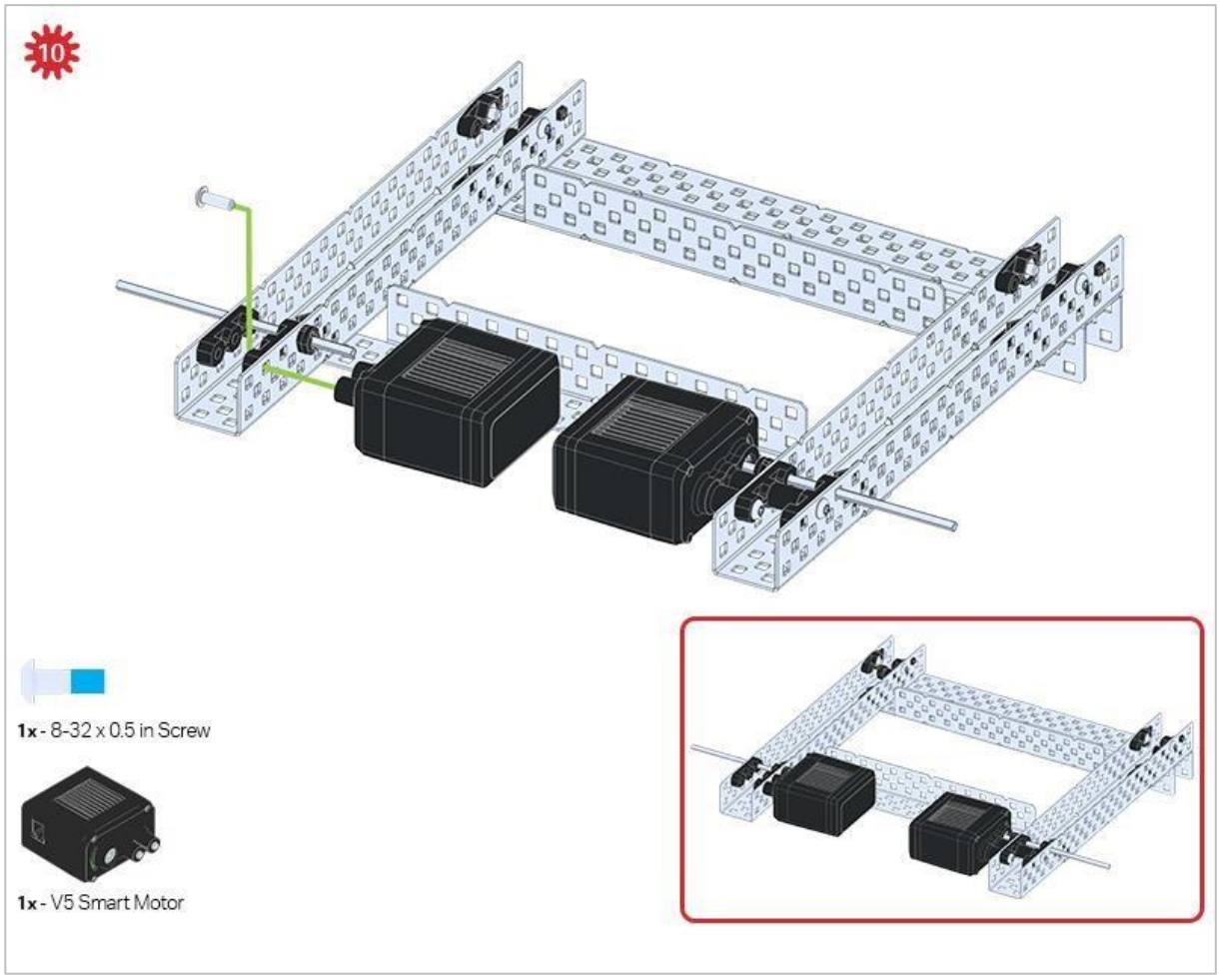


**1x** - 0.5 in Spacer

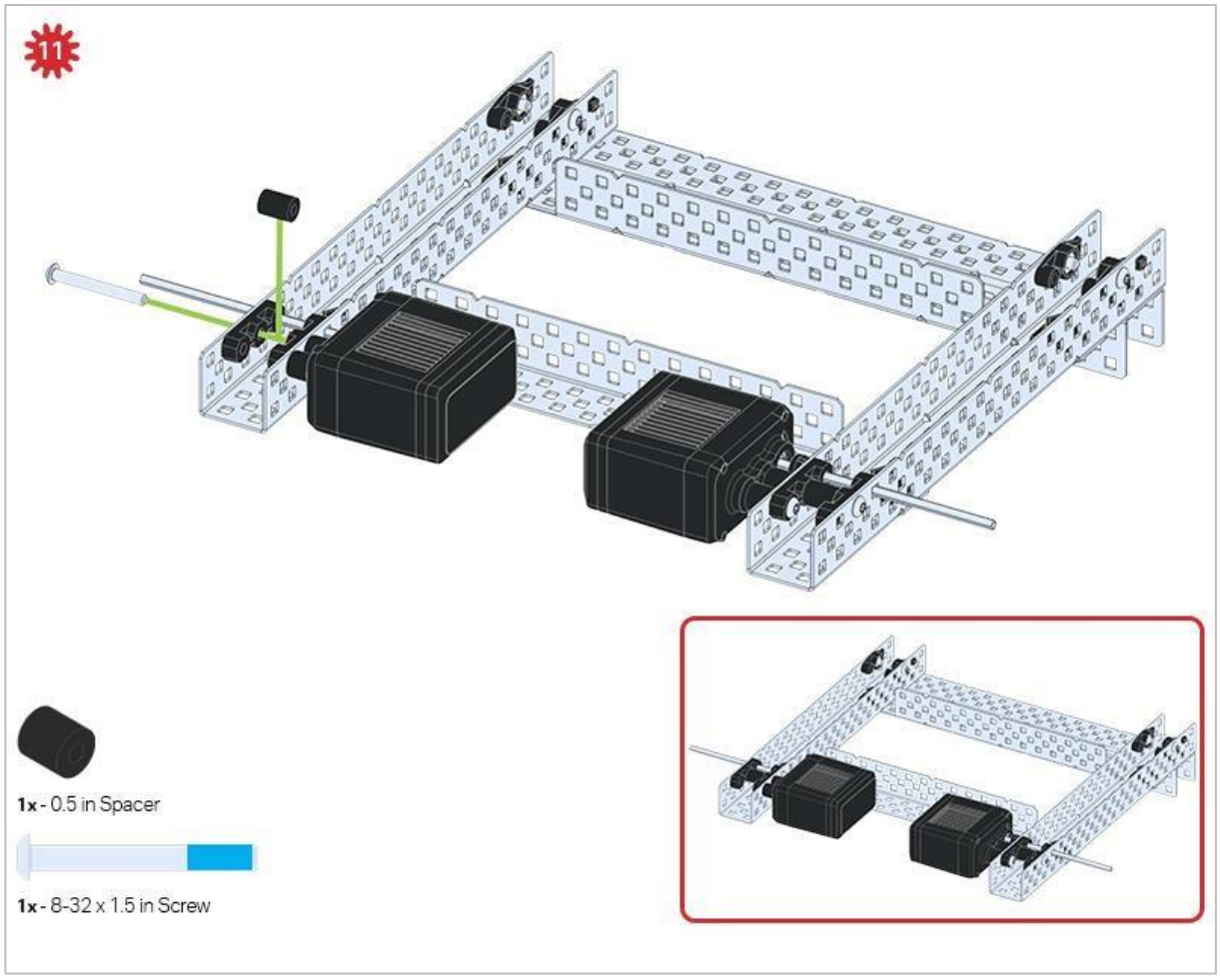
**1x** - 8-32 x 1.5 in Screw



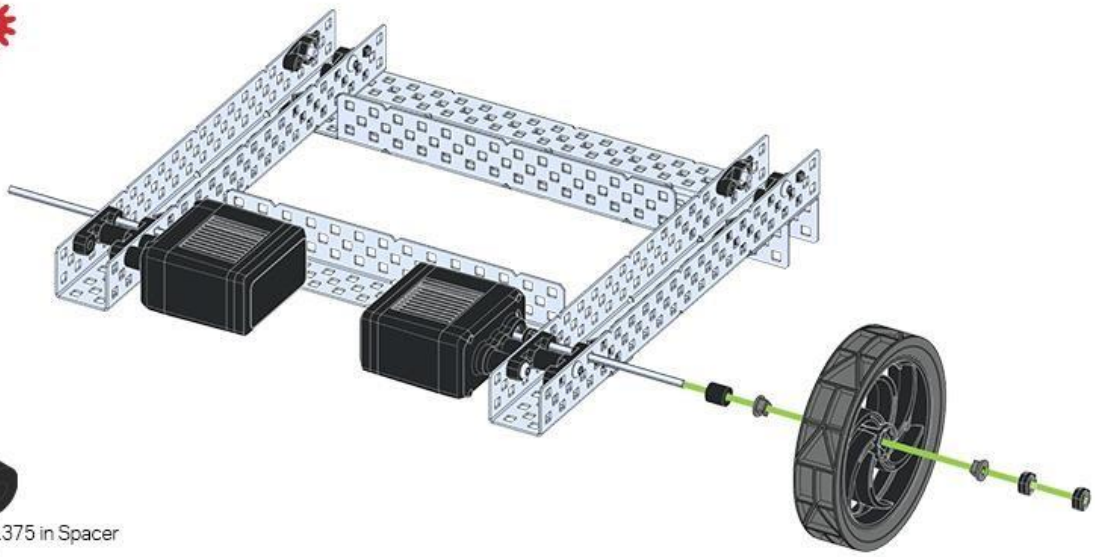

9
6
2x - Bearing Flat
1x - Step 6 Sub-Assembly




*Upewnij się, że silniki są ustawione we właściwym kierunku (otwory na śruby skierowane na zewnątrz konstrukcji, a otwór wału do wewnątrz).*


11
1x - 0.5 in Spacer
1x - 8-32 x 1.5 in Screw

12

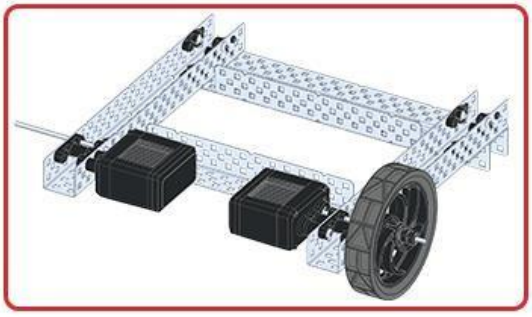


**1x** - 0.375 in Spacer

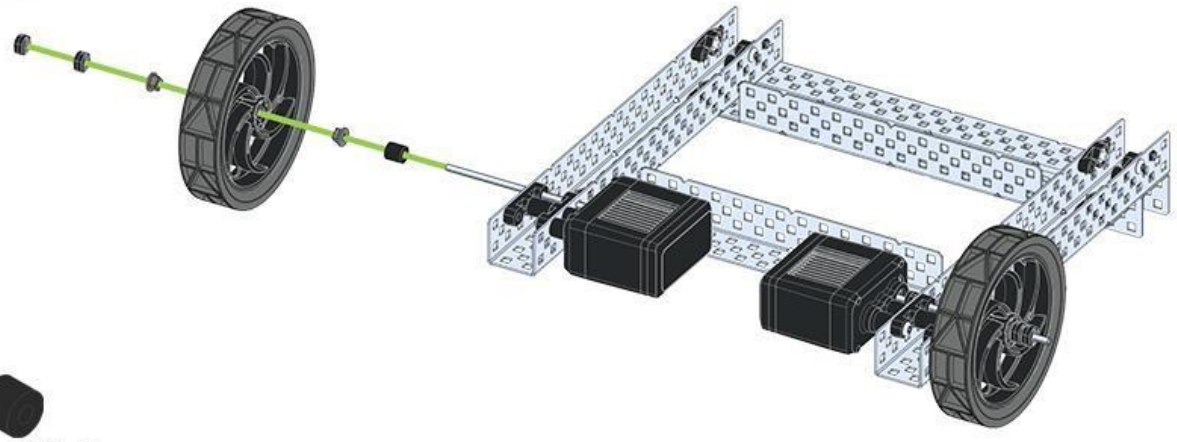
**2x** - Rubber Shaft Collar

**1x** - 4 in Wheel

**2x** - High Strength Shaft Insert



13



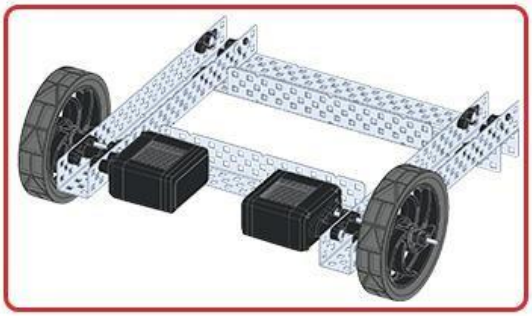


**1x** - 0.375 in Spacer

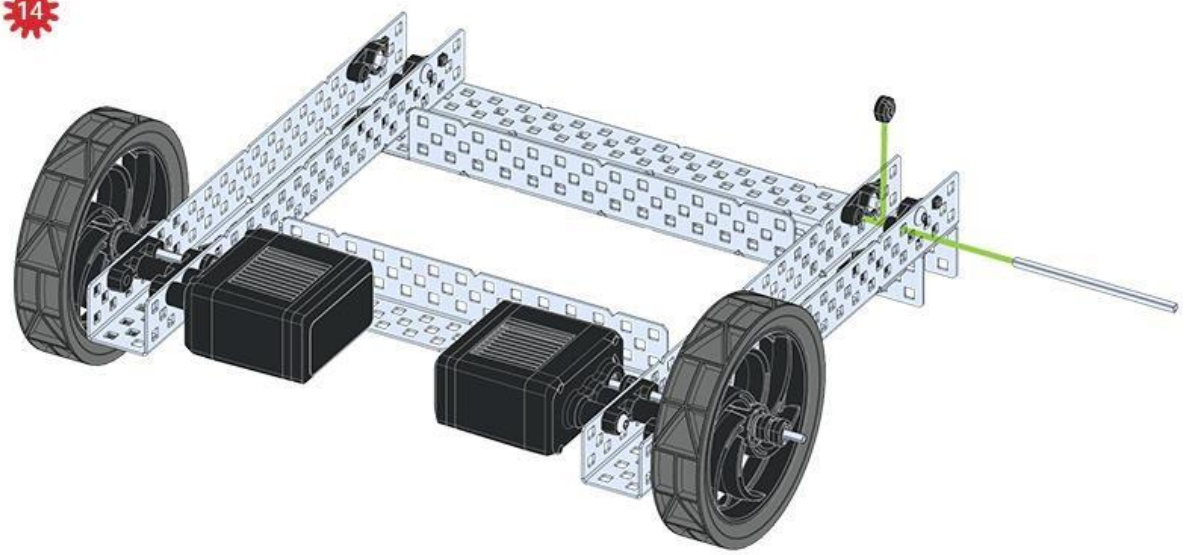
**2x** - Rubber Shaft Collar

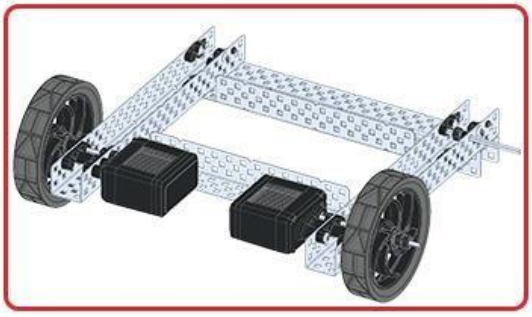
**1x** - 4 in Wheel

**2x** - High Strength Shaft Insert



14

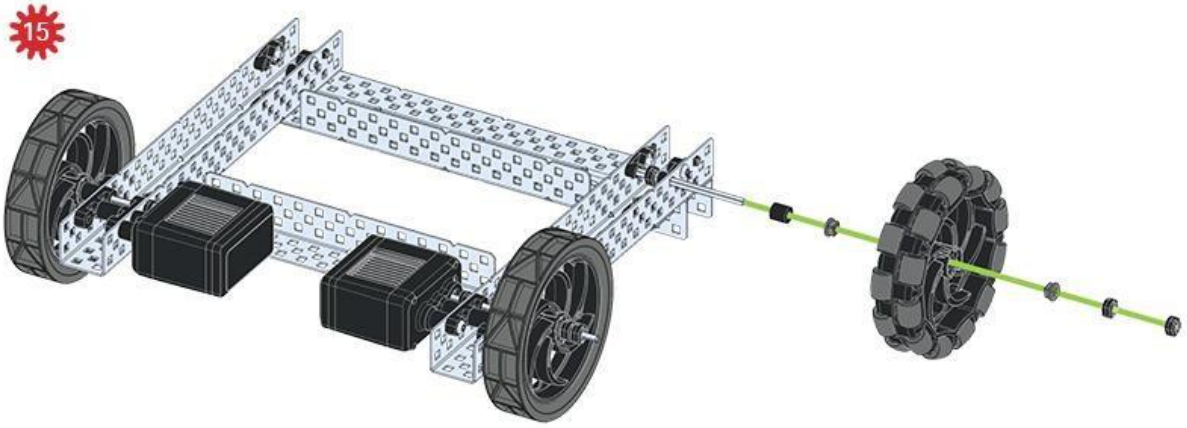




**1x** - Rubber Shaft Collar

**1x** - 3 in Shaft

15

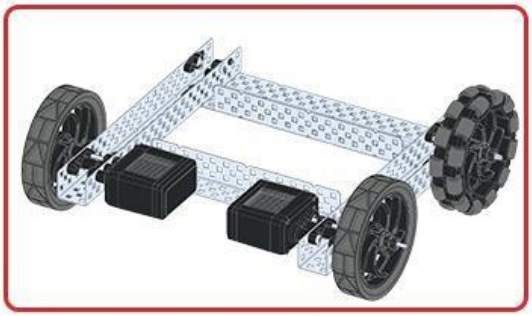


**1x** - 0.375 in Spacer

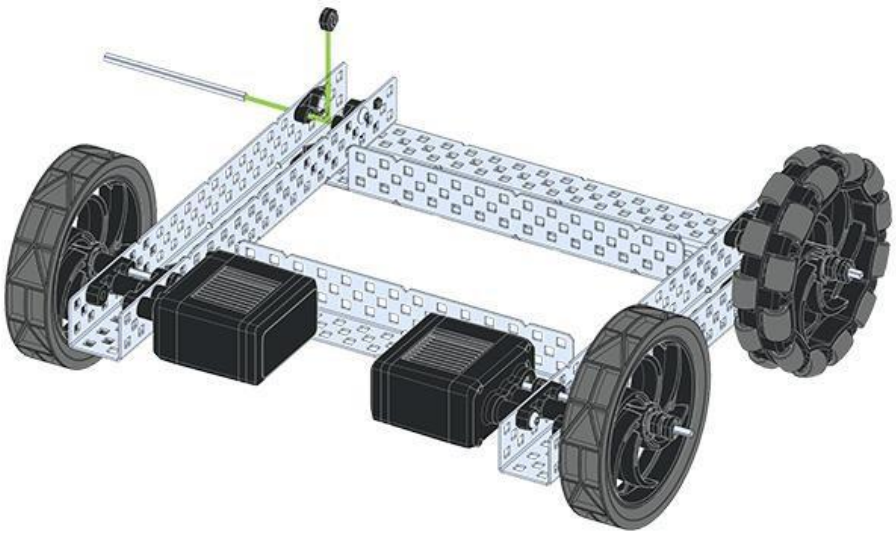
**2x** - Rubber Shaft Collar

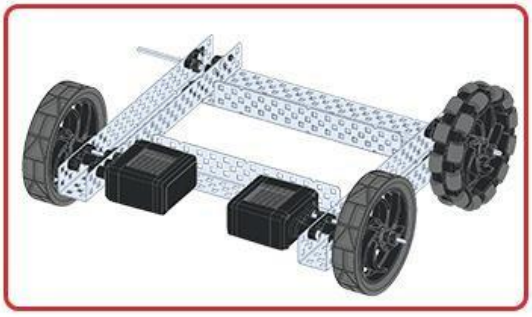
**1x** - 4 in Omni Wheel

**2x** - High Strength Shaft Insert



16

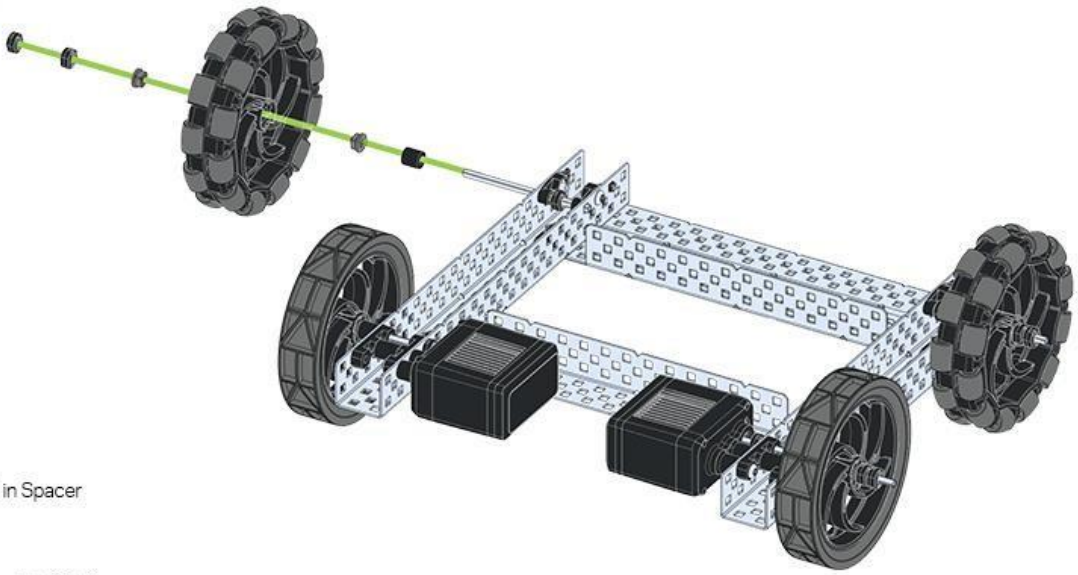




**1x** - Rubber Shaft Collar

**1x** - 3 in Shaft

17

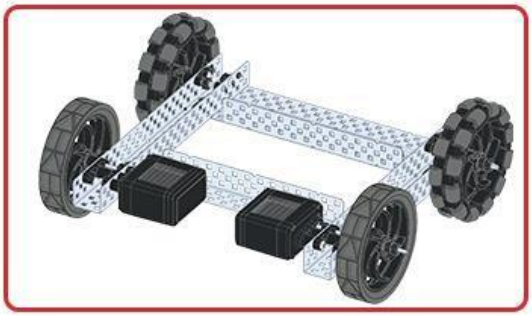


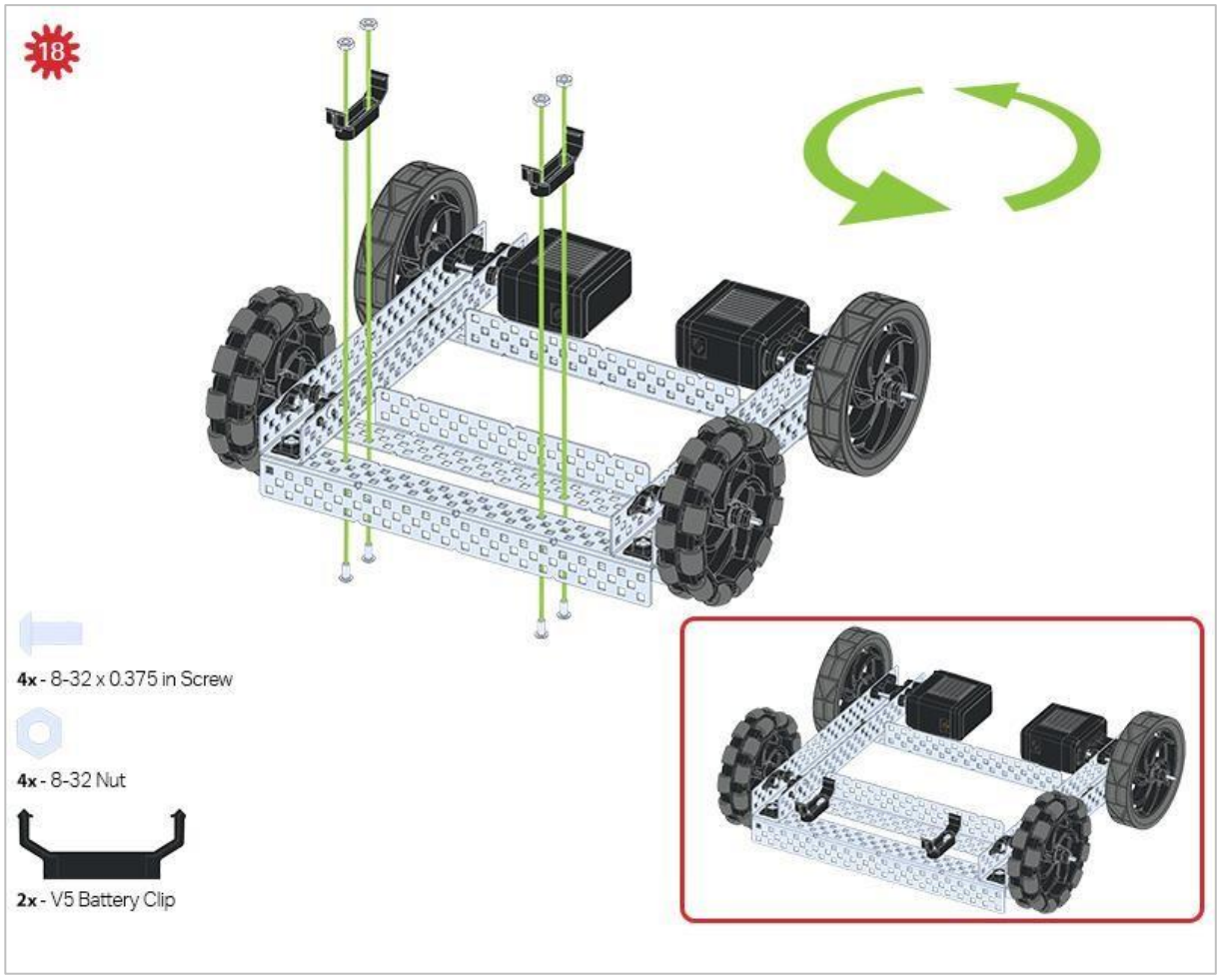
**1x** - 0.375 in Spacer

**2x** - Rubber Shaft Collar

**1x** - 4 in Omni Wheel

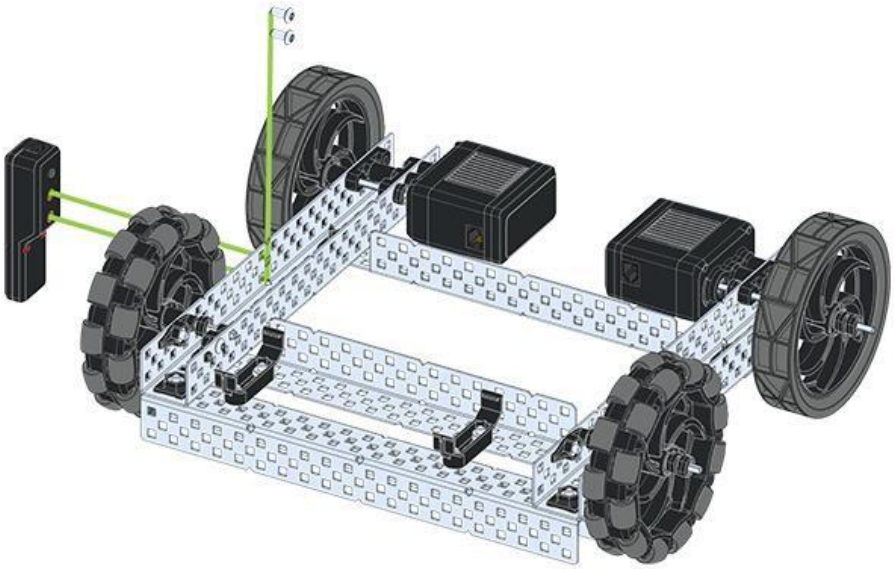
**2x** - High Strength Shaft Insert






*Zielona ikona wskazuje, że konstrukcję należy obrócić (o 180 stopni).*

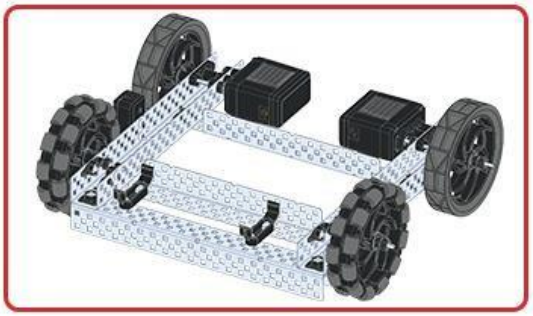
19

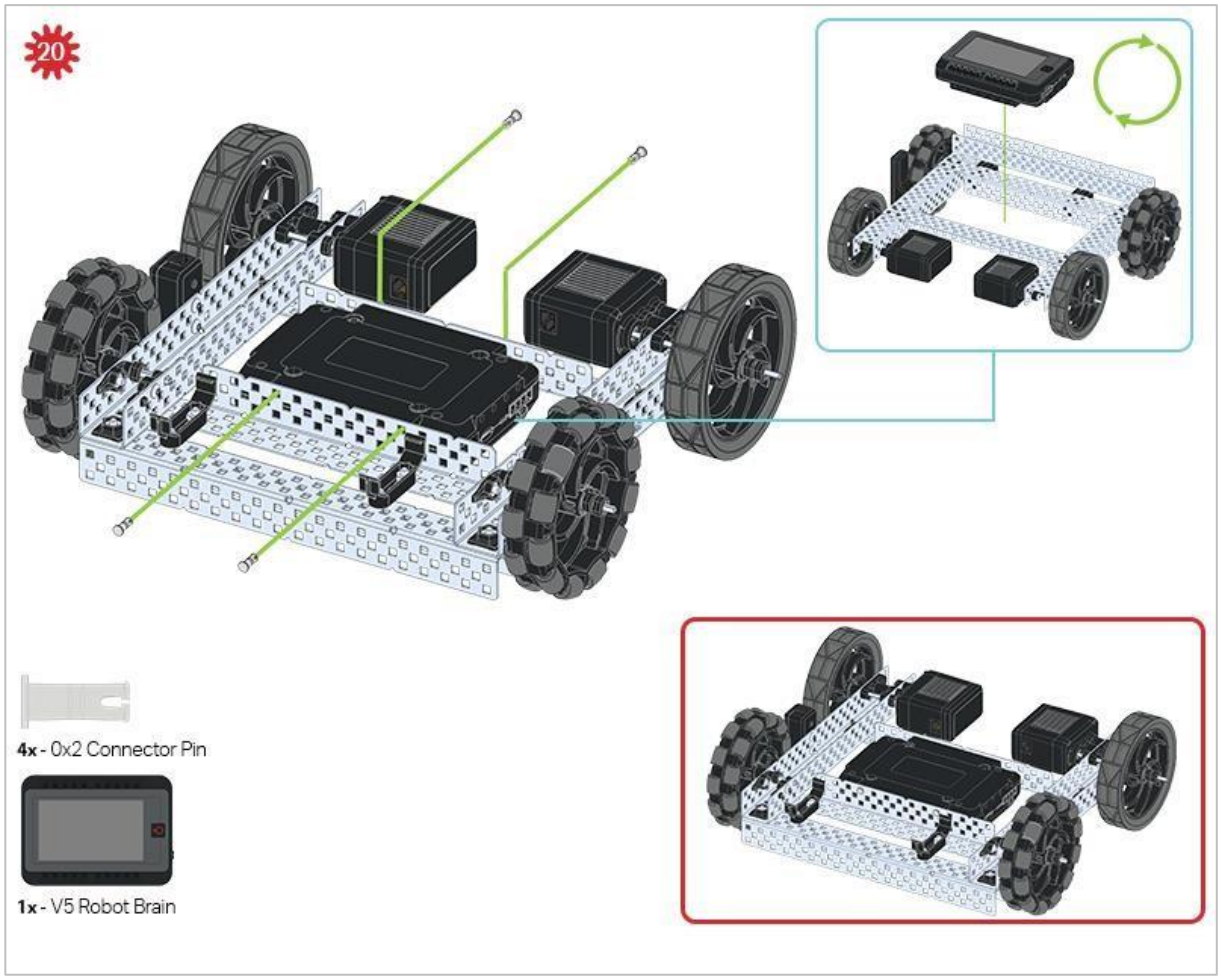


**1x** - V5 Radio

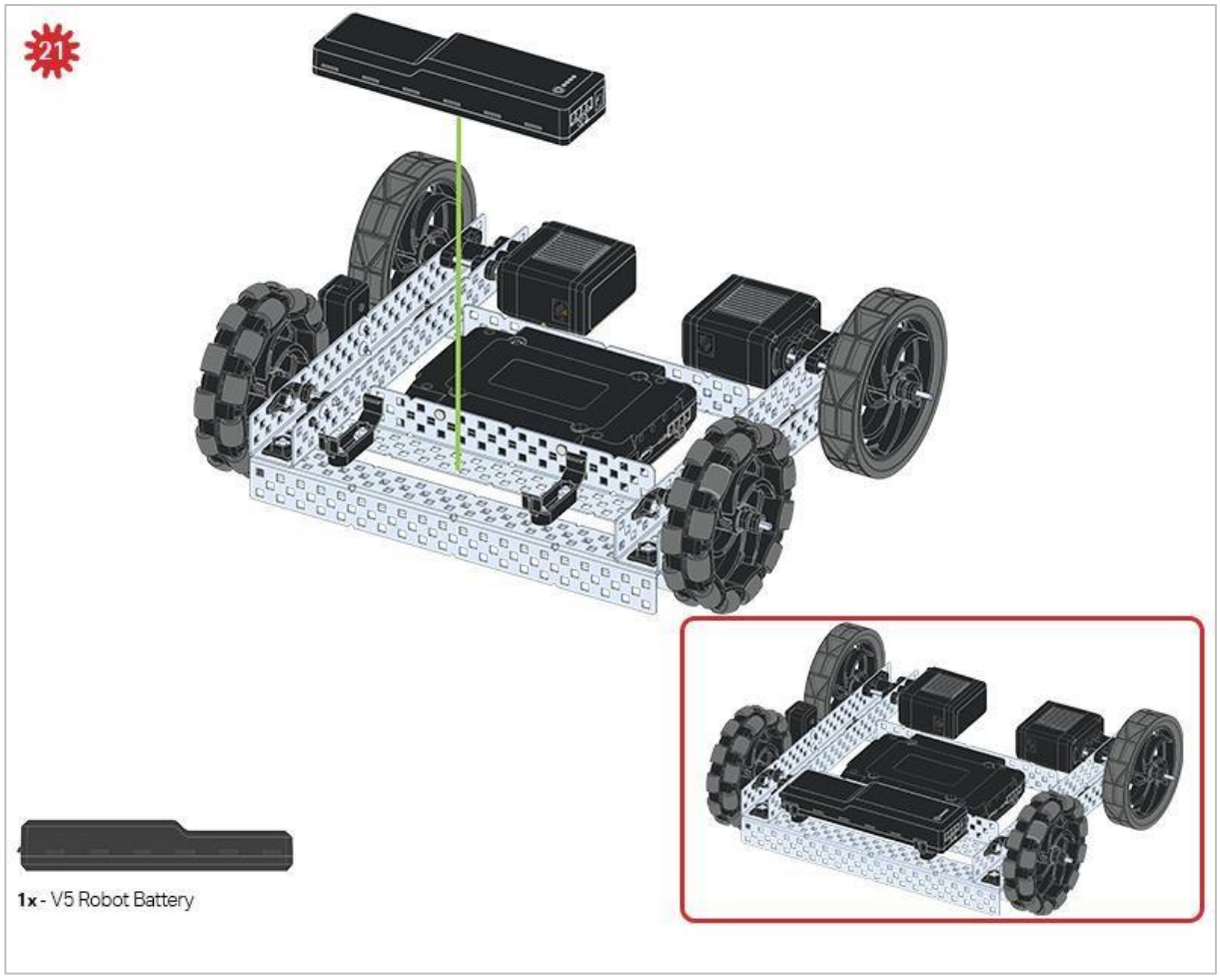


**2x** - 8-32 x 0.375 in Screw

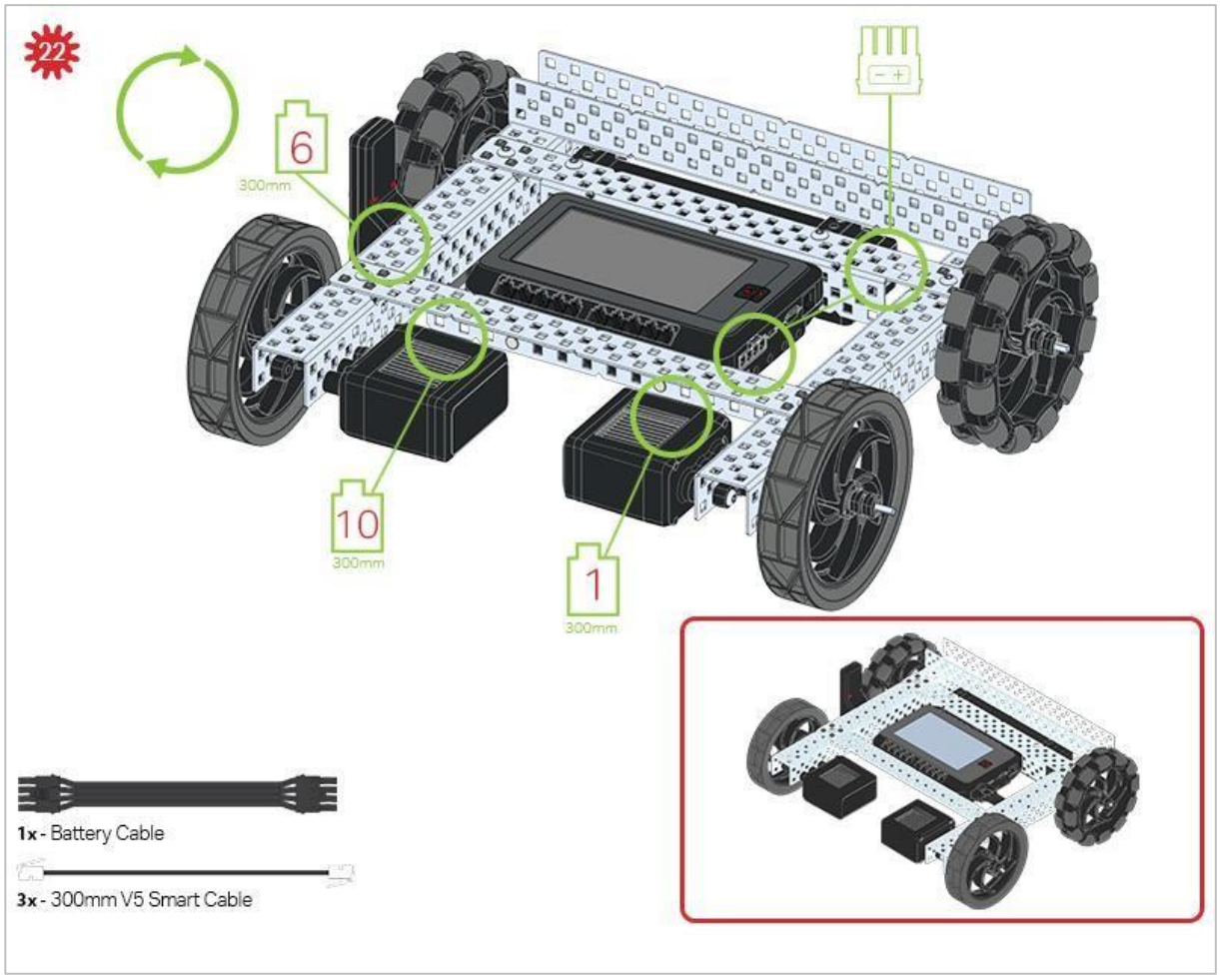




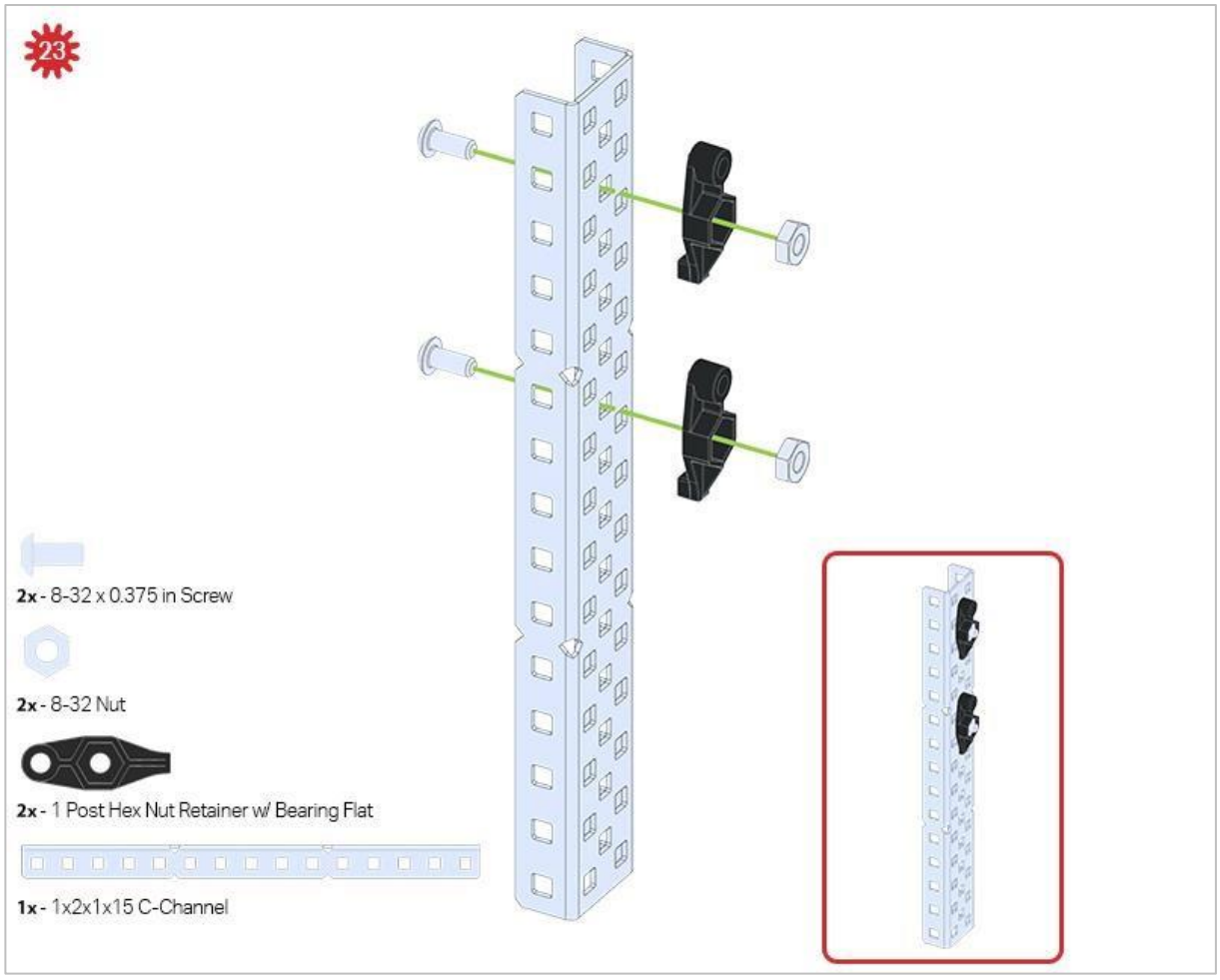

*Niebieskie objaśnieniee pokazuje, jaka powinna być orientacja Mózgu robota, jeśli konstrukcja została odwrócona prawą stroną do góry. Upewnij się, że 3-przewodowe porty w mózgu robota są skierowane w stronę Radia V5!*



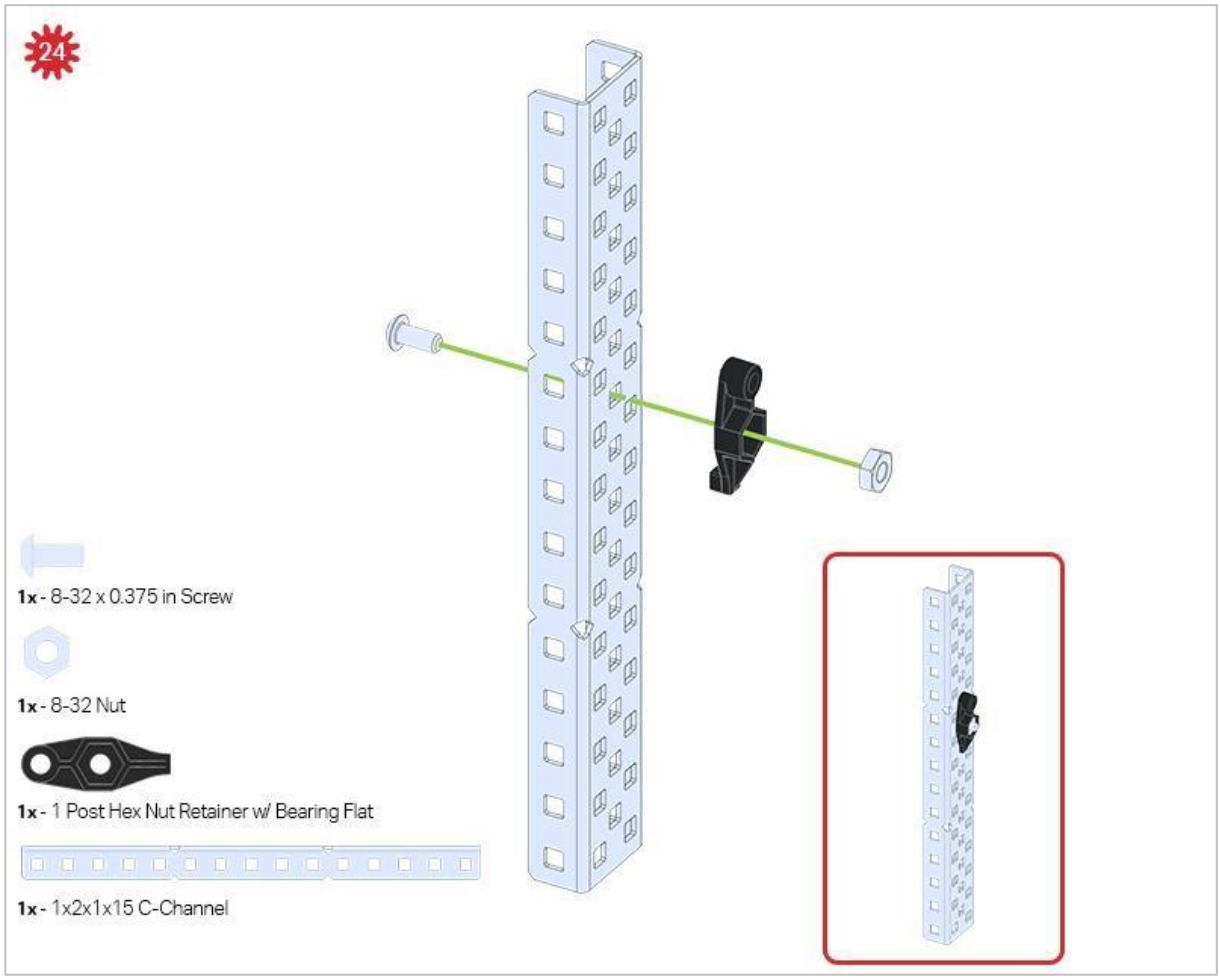

**1x** - V5 Robot Battery




*Zielone wobjasnienia wskazują, do którego portu w Mózgu robota należy podłączyć każde urządzenie za pomocą odpowiedniego kabla.*


23
2x - 8-32 x 0.375 in Screw
2x - 8-32 Nut
2x - 1 Post Hex Nut Retainer w/ Bearing Flat
1x - 1x2x1x15 C-Channel

24



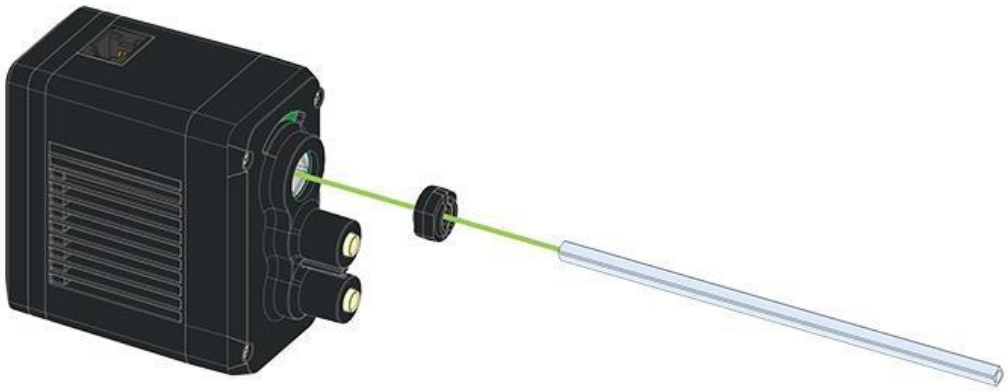
**1x** - 8-32 x 0.375 in Screw

**1x** - 8-32 Nut

**1x** - 1 Post Hex Nut Retainer w/ Bearing Flat

**1x** - 1x2x1x15 C-Channel

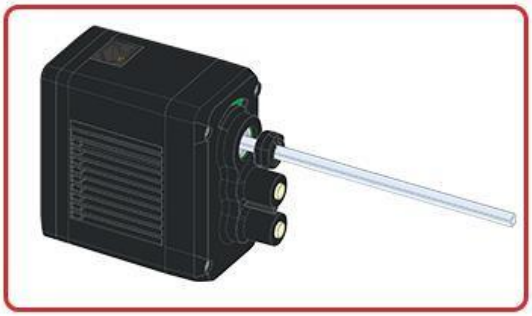
25

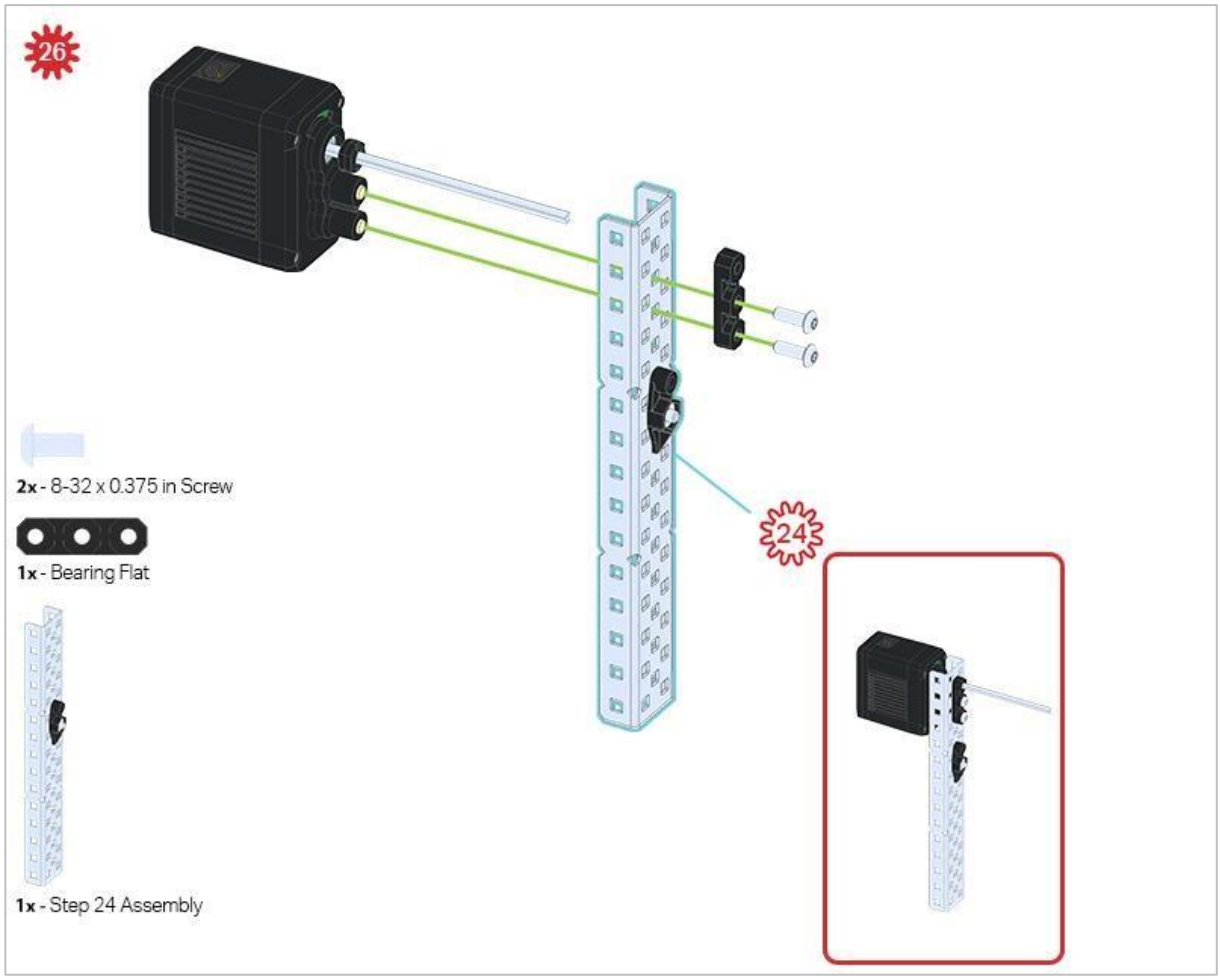


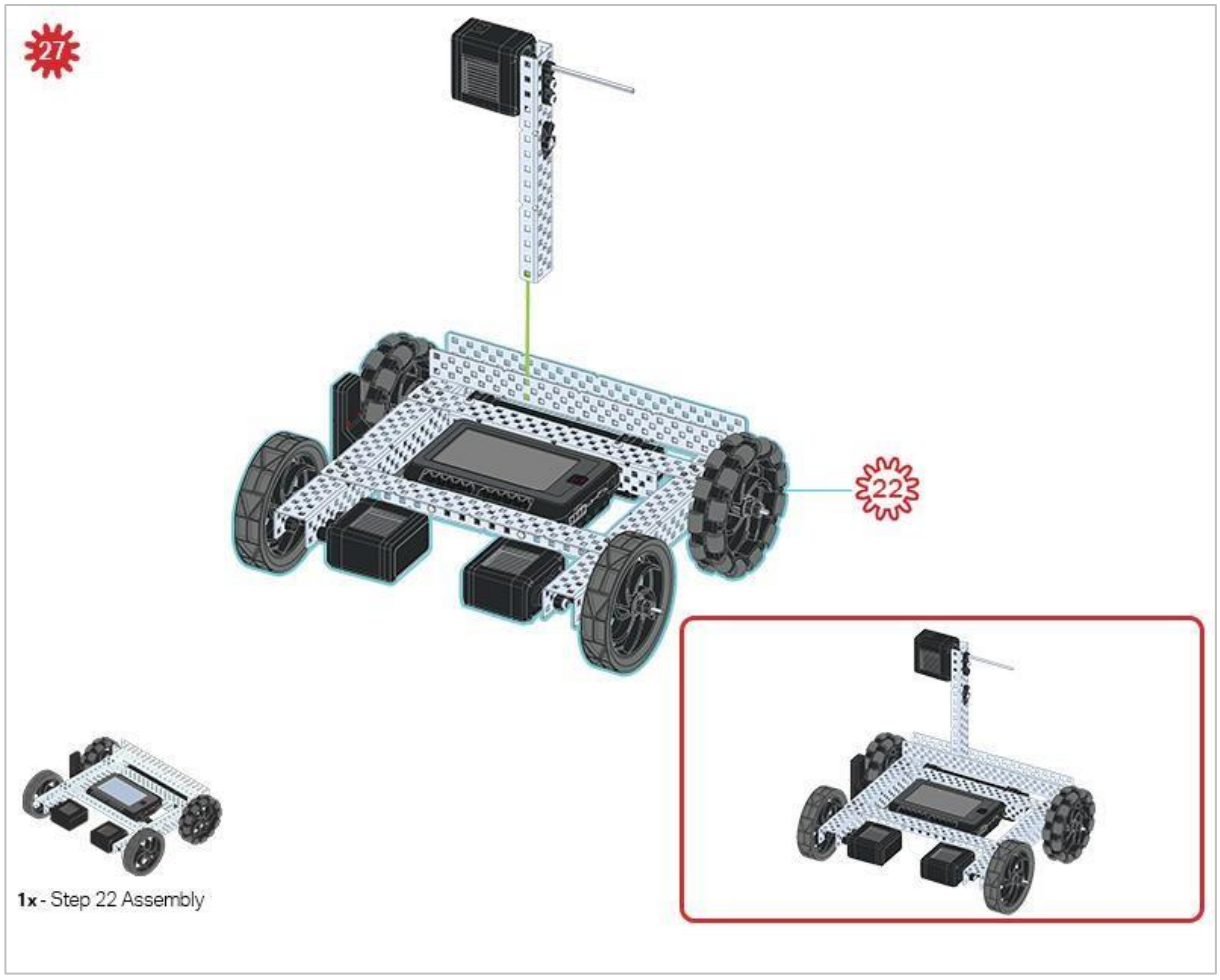
**1x** - Rubber Shaft Collar

**1x** - V5 Smart Motor

**1x** - 4 in Shaft




26
2x - 8-32 x 0.375 in Screw
1x - Bearing Flat
1x - Step 24 Assembly
24


27
22
1x - Step 22 Assembly

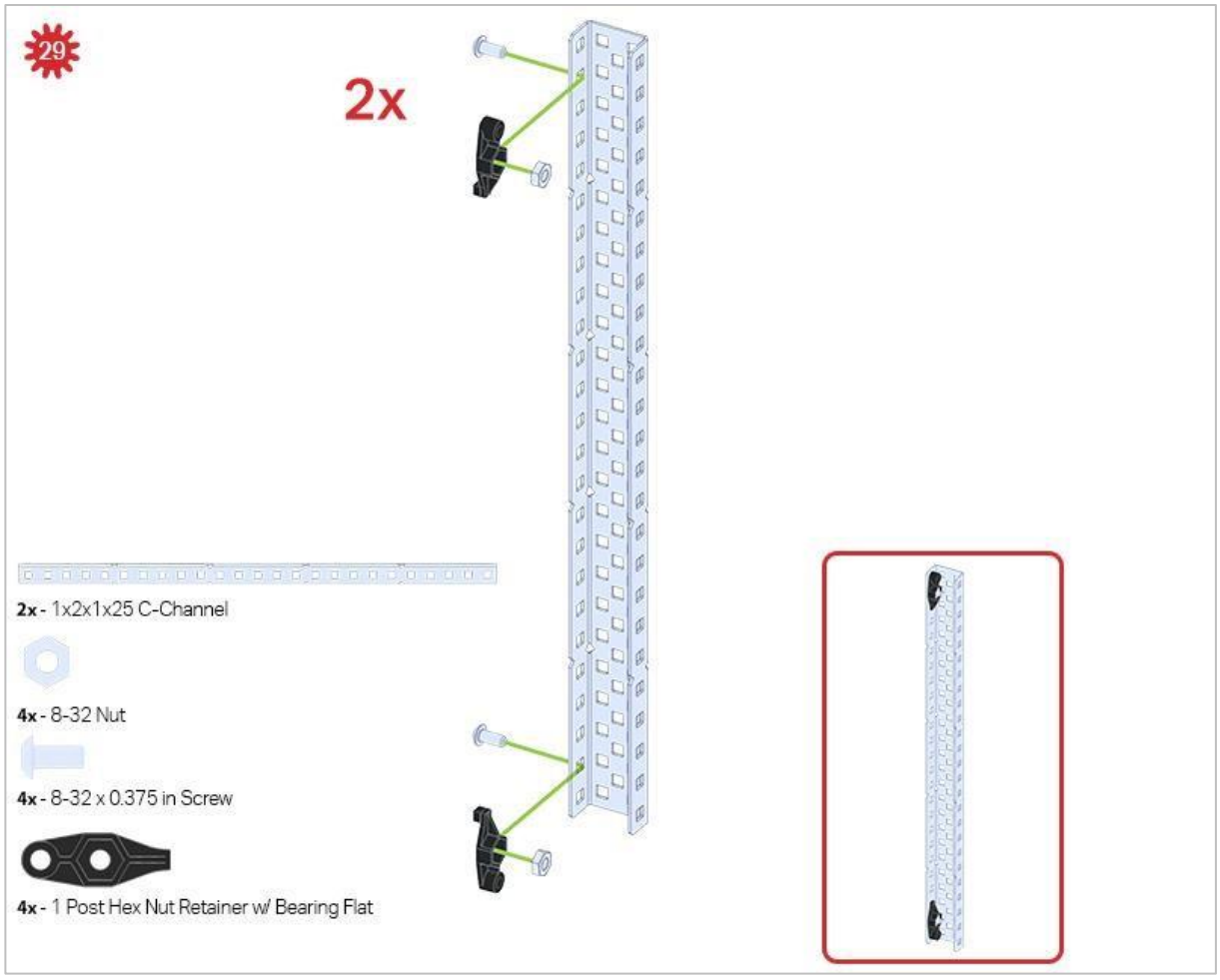
28

**1x** - 8-32 Nut

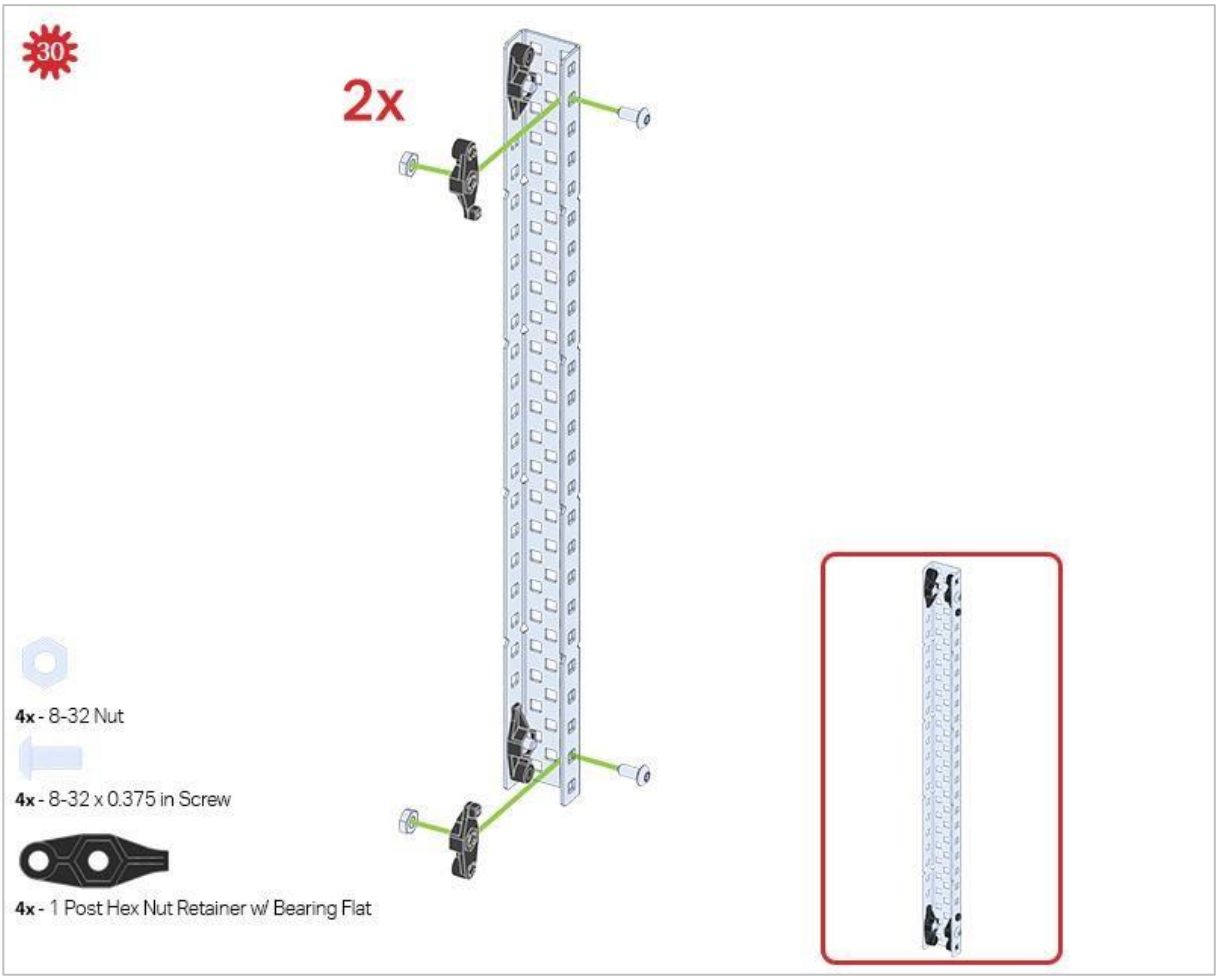
**1x** - 0.875 in Spacer

**2x** - 1 Post Hex Nut Retainer

**1x** - 8-32 x 1.5 in Screw

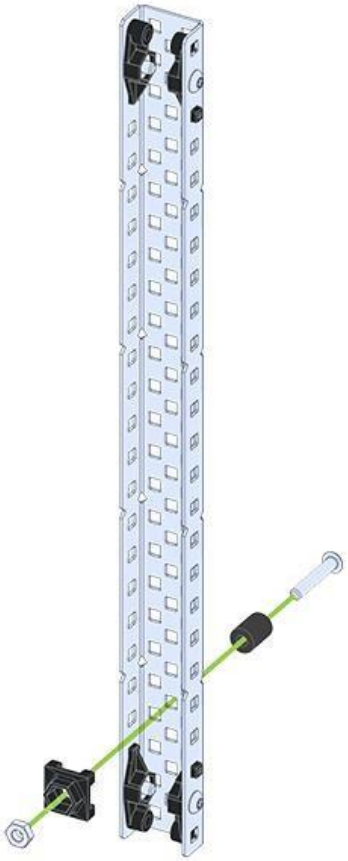



*Upewnij się, że w tym kroku wykonałeś dwa elementy!*




*Ten krok dodaje element do dwóch konstrukcji z kroku 29.*

31



**1x** - 0.5 in Spacer

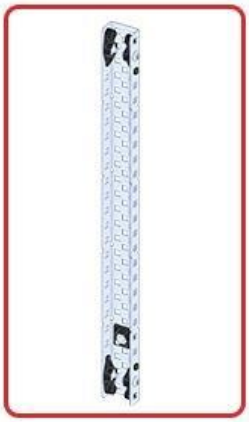


**1x** - 8-32 Nut

**1x** - 4 Post Hex Nut Retainer



**1x** - 8-32 x 1 in Screw



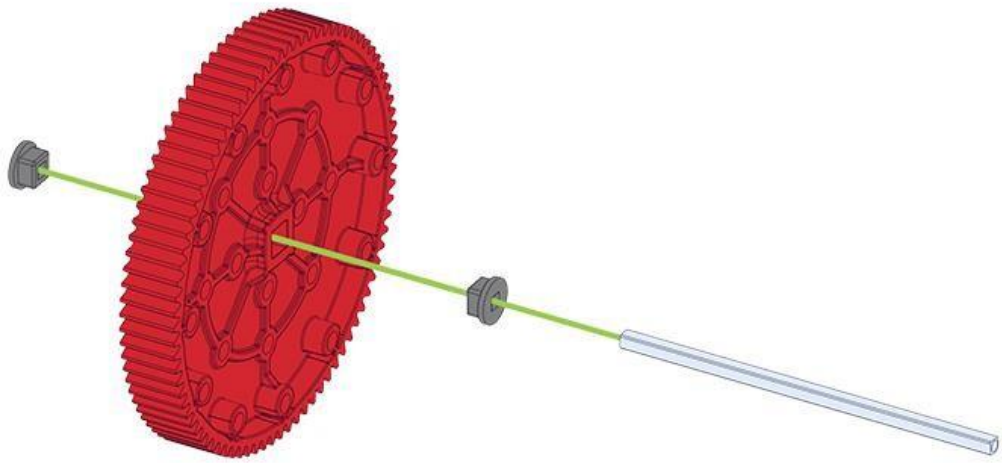
32



**1x** - 8-32 Nut

**1x** - 1 Post Hex Nut Retainer

33

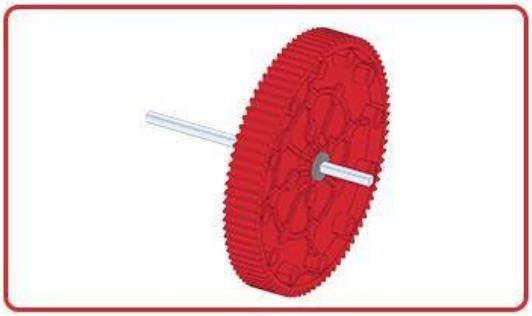


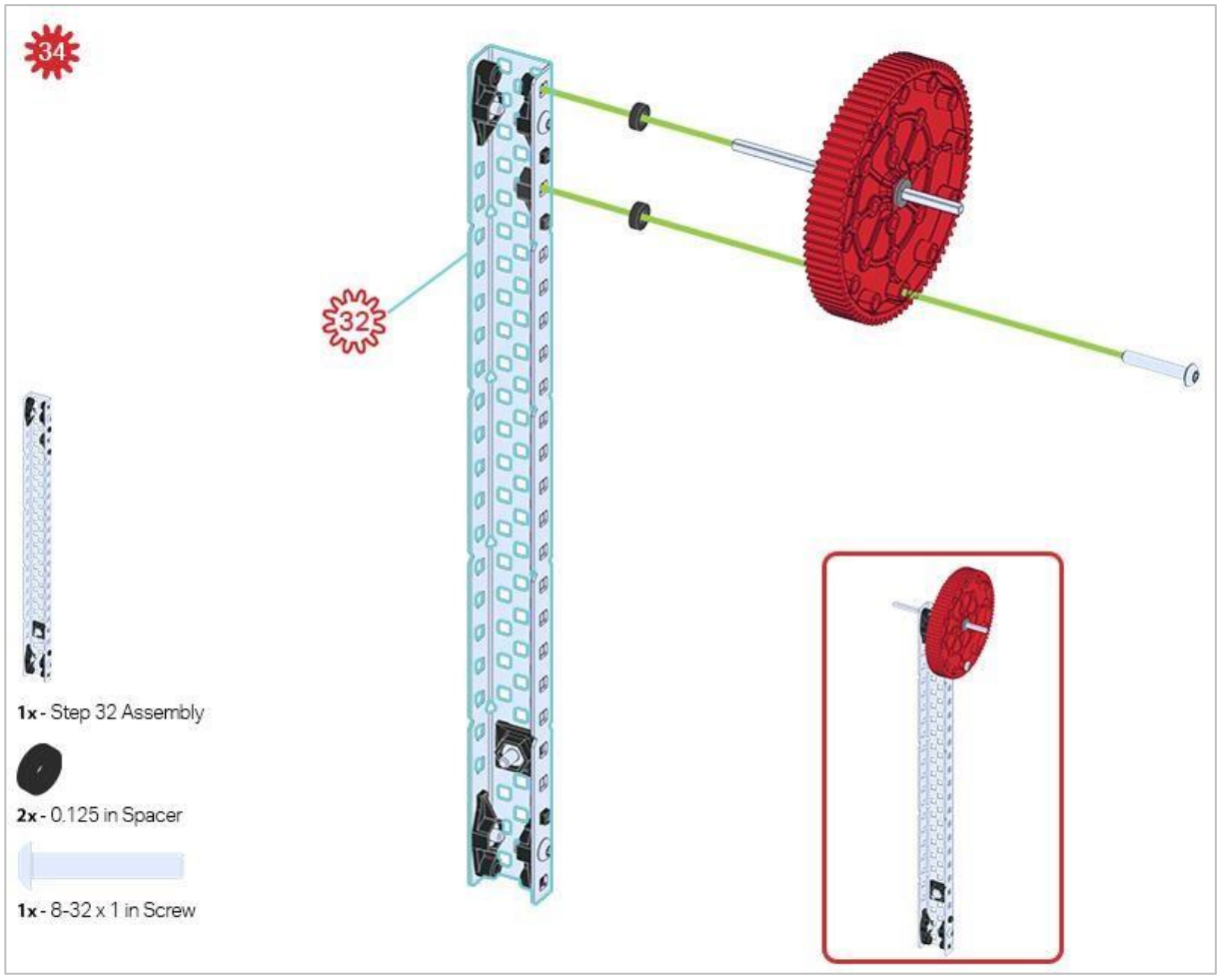
**1x** - High Strength 84 Tooth Gear

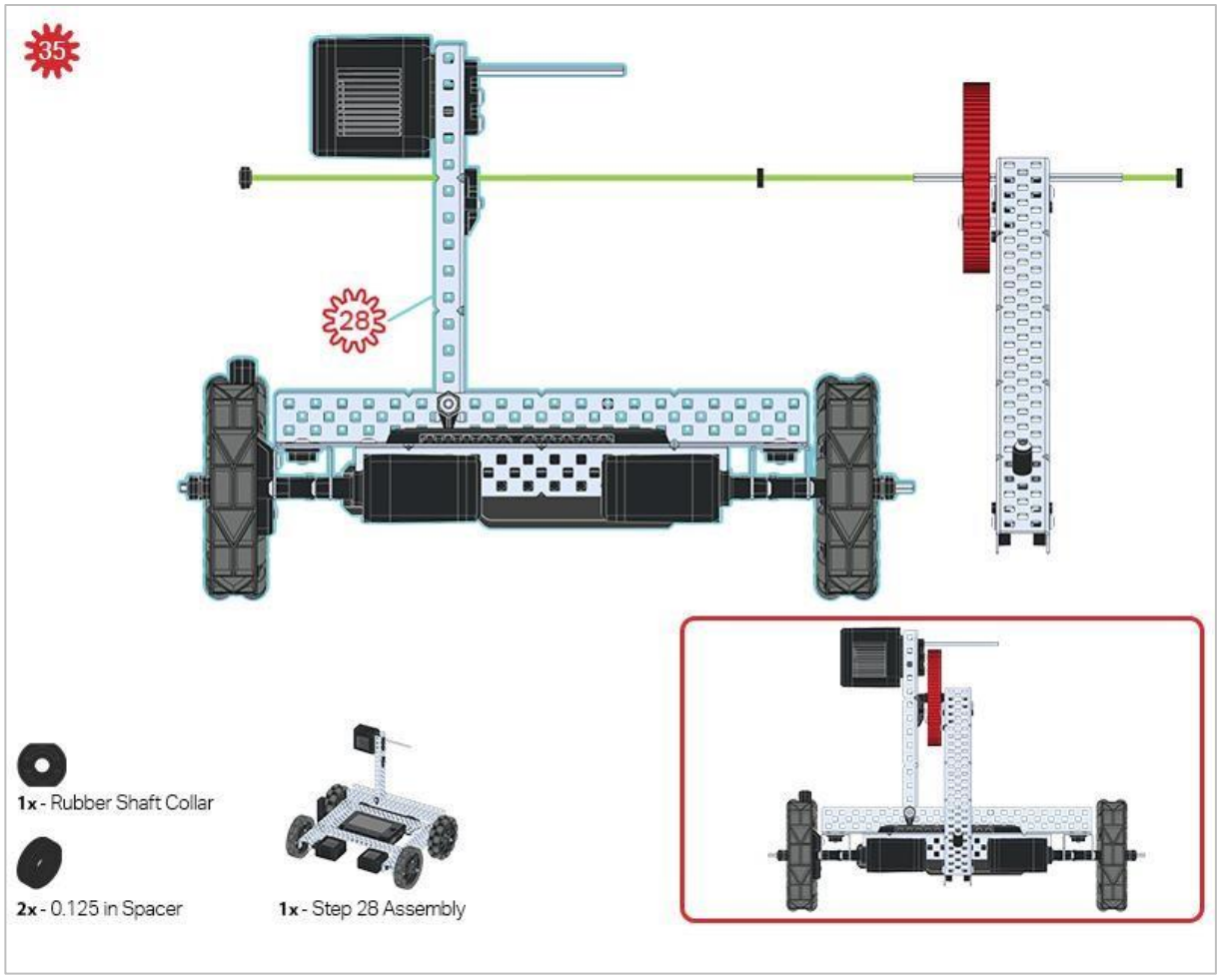
**2x** - High Strength Shaft Insert



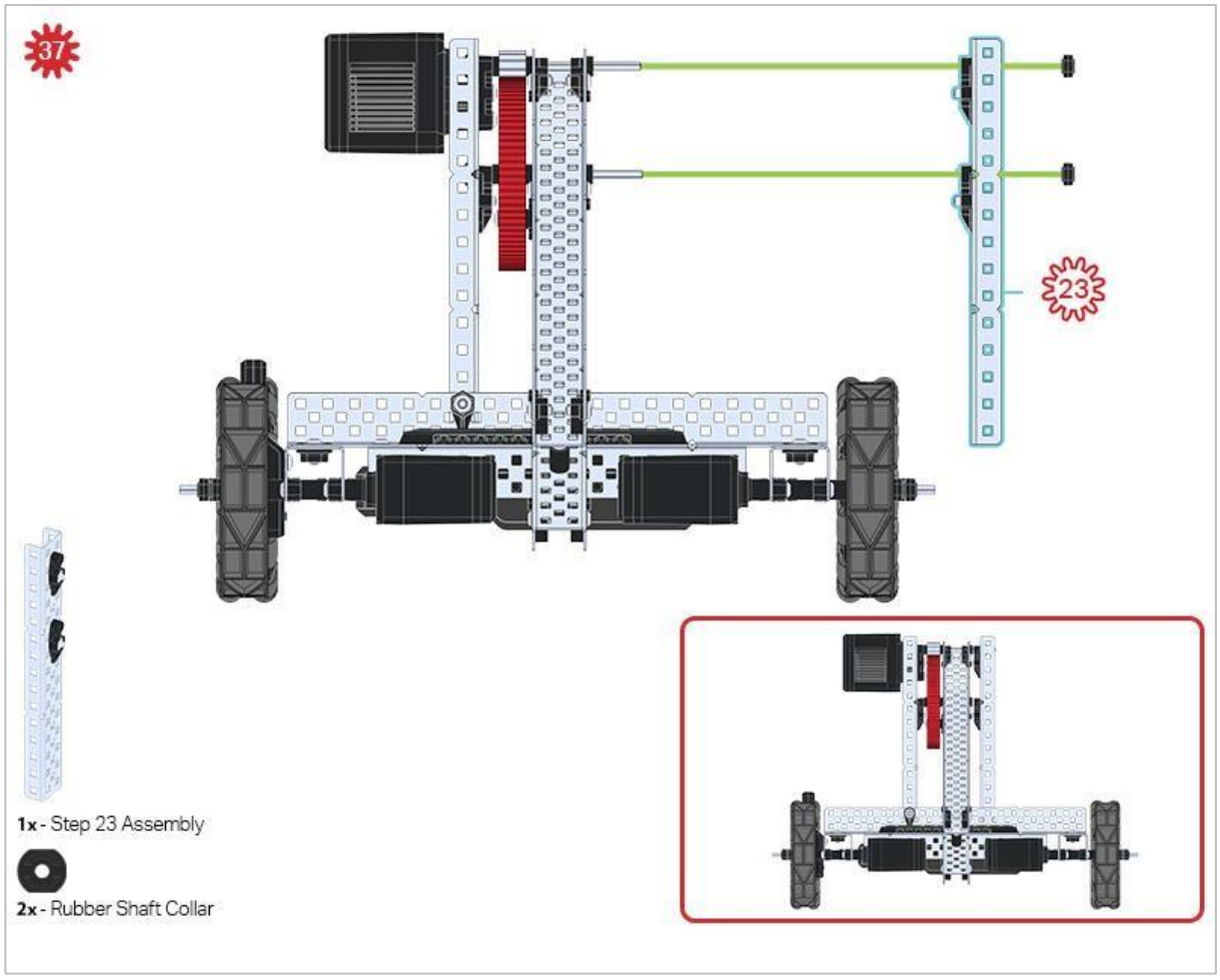
**1x** - 3.5 in Shaft




34
32
1x - Step 32 Assembly
2x - 0.125 in Spacer
1x - 8-32 x 1 in Screw


35
28
1x - Rubber Shaft Collar
2x - 0.125 in Spacer
1x - Step 28 Assembly


36
30
3x - 0.125 in Spacer
1x - High Strength 12 Tooth Pinion
1x - High Strength Pinion Insert
1x - Step 30 Assembly




**1x** - Step 23 Assembly

**2x** - Rubber Shaft Collar

38

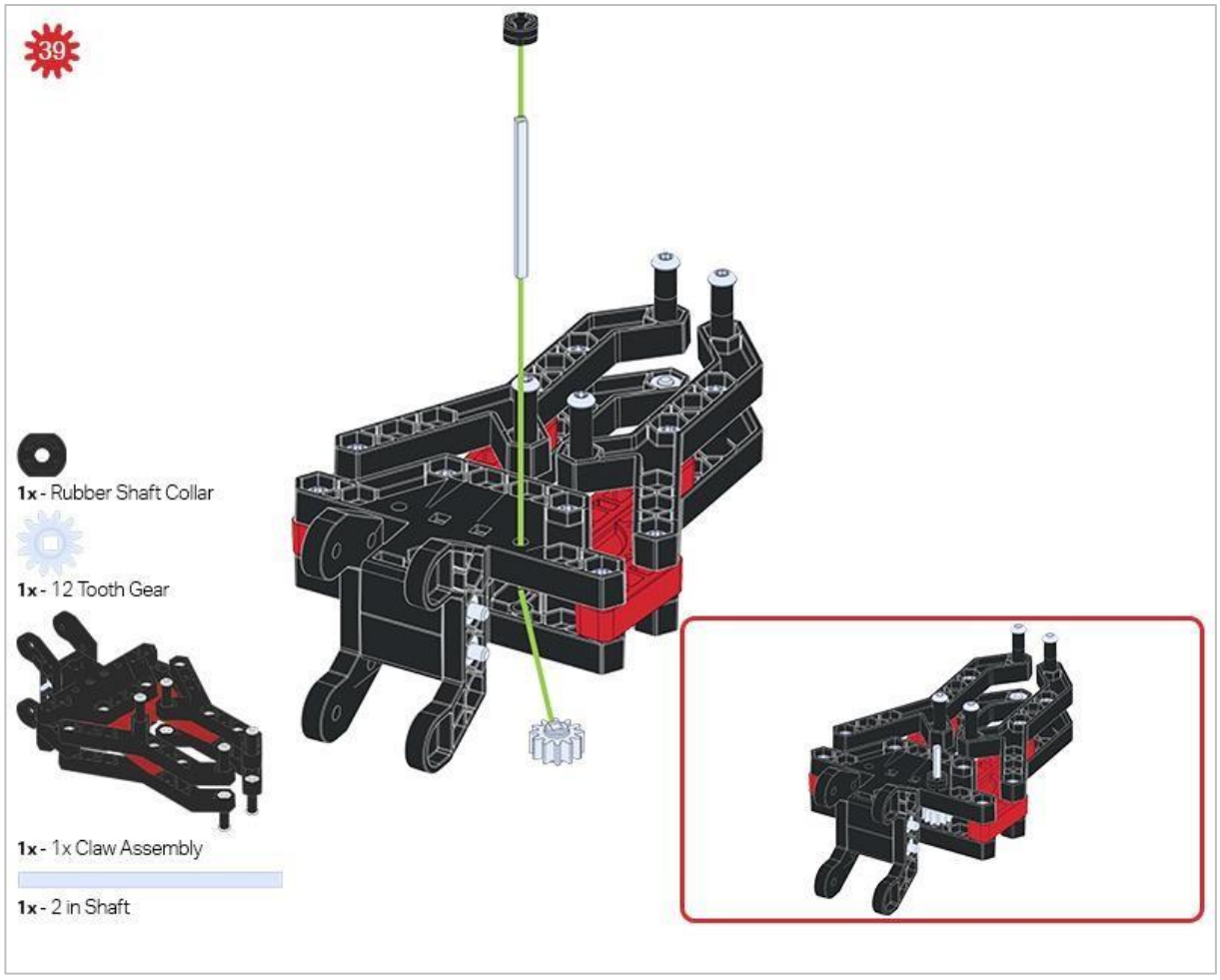
**1x** - 0.875 in Spacer



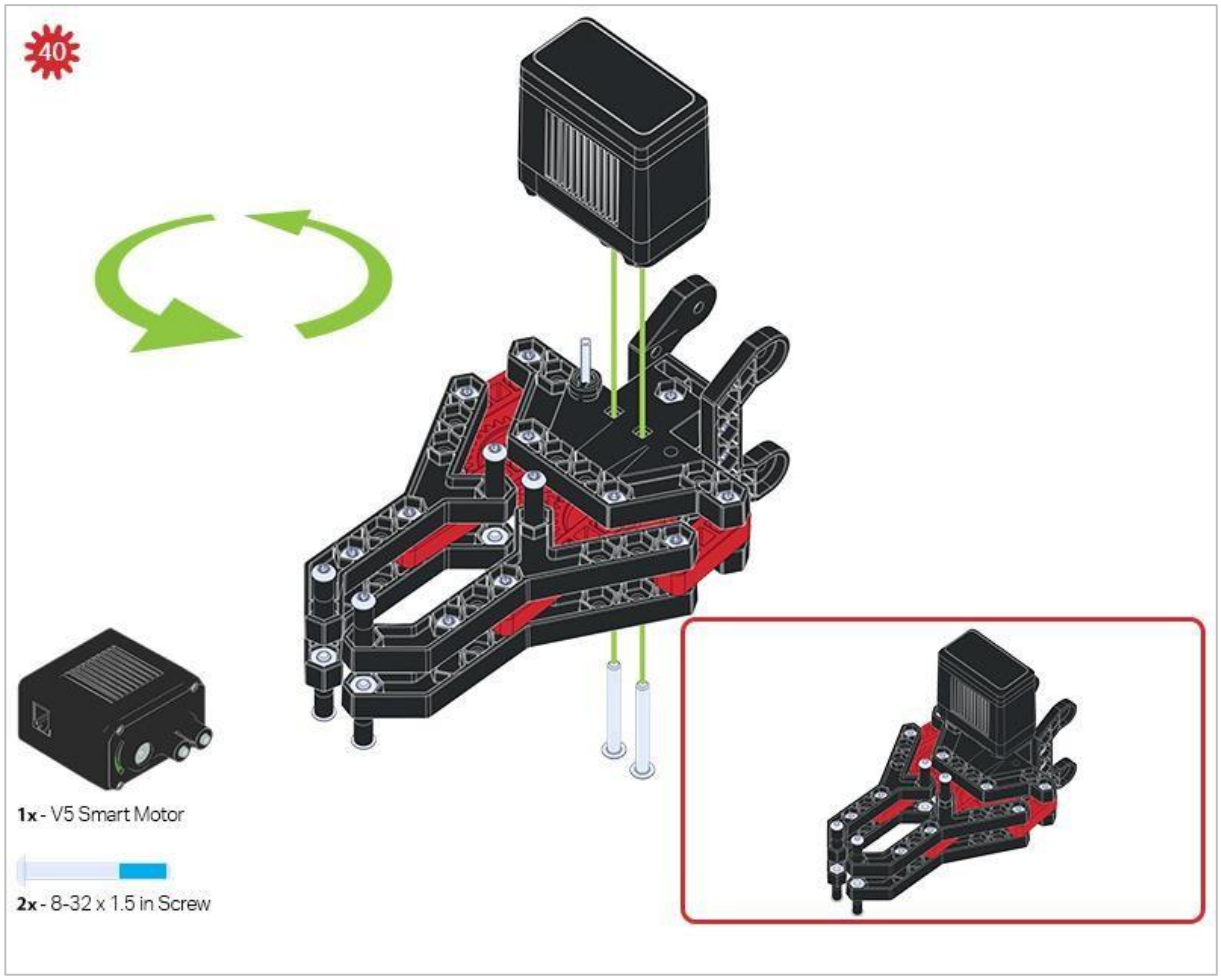
**1x** - 8-32 Nut

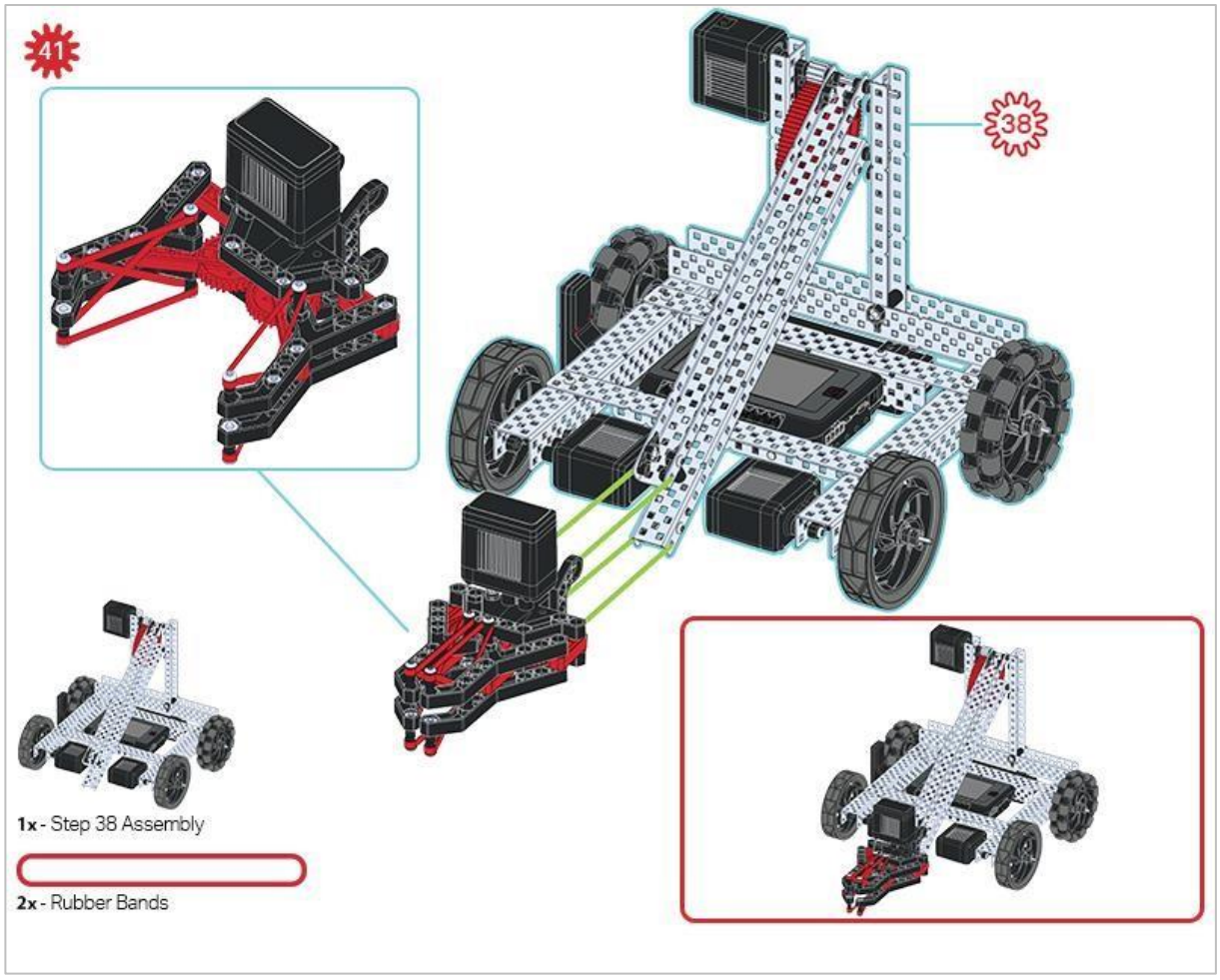
**2x** - 1 Post Hex Nut Retainer

**1x** - 8-32 x 1.5 in Screw



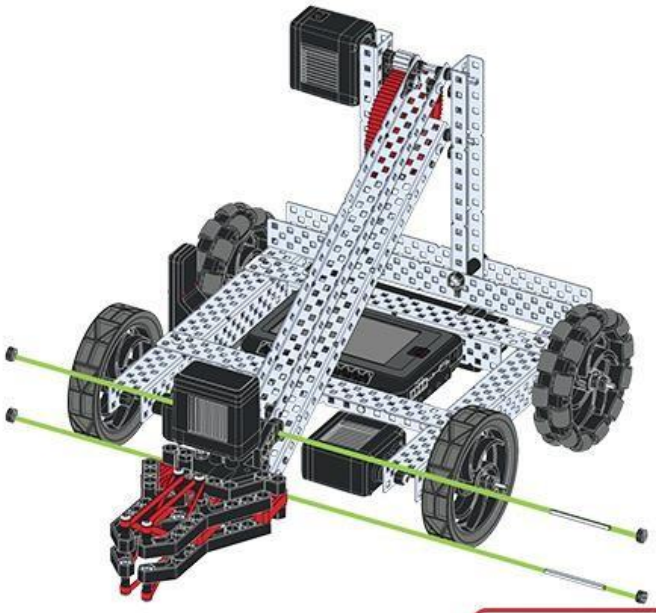

*Upewnij się, że 12-zębowe koło zębate jest zainstalowane po prawej stronie kleszczy.*


40
1x - V5 Smart Motor
2x - 8-32 x 1.5 in Screw



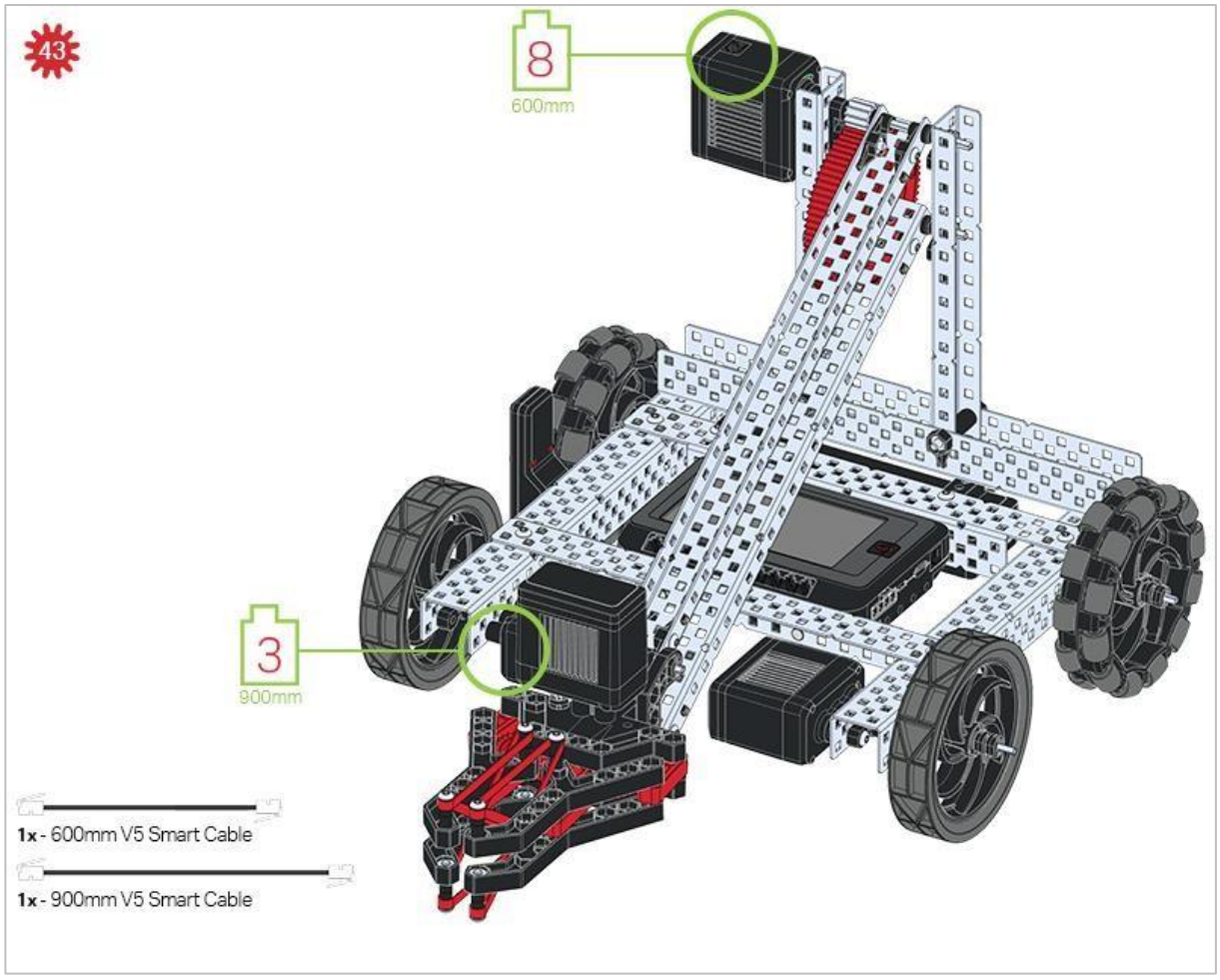

*Upewnij się, że port na silniku jest skierowany w prawą stronę robota, gdy kleszcze są zainstalowane (po tej samej stronie co Radio V5).*

42



**4x** - Rubber Shaft Collar

**2x** - 2 in Shaft




## Wskazówki

Sprawdź Dodatek, aby uzyskać informacje o tym, jak używać Hex Nut Retainers.

# Analiza

Po zakończeniu budowy sprawdź, co robi robot. Zbadaj konstrukcję, a następnie odpowiedz na te pytania w notatniku technicznym.

- Gdzie znajduje się punkt obrotu (punkt, wokół którego obraca się robot) w tej konstrukcji?
- Dokonaj prognozy, a następnie ręcznie przesuń 4-calowe koło z jednej strony do przodu, jednocześnie przesuwając 4-calowe koło po przeciwnej stronie do tyłu w tym samym tempie.

Po ręcznym przesunięciu obu kół opisz teraz, gdzie znajduje się punkt obrotu. Jak zmieniłby się punkt obrotu robota, gdyby projekt został zmieniony, tak aby oba 4-calowe koła Omni (kroki 14-17) były również napędzane silnikami, dzięki czemu wszystkie 4 koła były napędzane?

- Dokonaj prognozy, a następnie ręcznie przesuń oba koła po jednej stronie do przodu, jednocześnie przesuwając oba koła po przeciwnej stronie do tyłu w tym samym tempie.

Po ponownym ręcznym przesunięciu kół opisz teraz, gdzie znajduje się punkt obrotu. Wyjaśnij, jak punkt obrotu robota może zmienić jego zachowanie.

SPARK
PLAY

**Przetestuj swojego robota, obserwuj, jak działa, i wzmocnij swoją logikę i umiejętności rozumowania poprzez pomysłową, kreatywną zabawę.**

# Powtarzalność robotów

*Roboty na linii montażowej muszą wielokrotnie wykonywać te same czynności*

## Używanie pętli do powtarzania działań robotów

Roboty i komputery są bardzo dobre w wielokrotnym wykonywaniu czynności. Komputery używają powtarzania do wykonywania milionów obliczeń na sekundę z niesamowitą konsekwencją. Ponieważ roboty są zbudowane tak, aby wchodzić w interakcję z otoczeniem i precyzyjnie wykonywać zadania, można je skutecznie wykorzystywać do powtarzania zadań.

Zachowania, które mają być powtarzane, są pogrupowane w strukturach programowania zwanych pętlami. Liczba i szybkość powtarzania zależą od wielu czynników, które może określić programista.

Oto kilka przykładów, w których powtórzenia mogą być przydatne:

- Wielokrotne wykonywanie rutynowych zadań
- Wielokrotne sprawdzanie określonych warunków w celu sprawdzenia zachodzących zmian

# Programming Loops - VEXcode V5 Blocks

**The V5 Clawbot is ready to move!**

You can use the Help information inside of VEXcode V5 Blocks to learn about the blocks. For guidance in using the Help feature, see the Using Help tutorial.

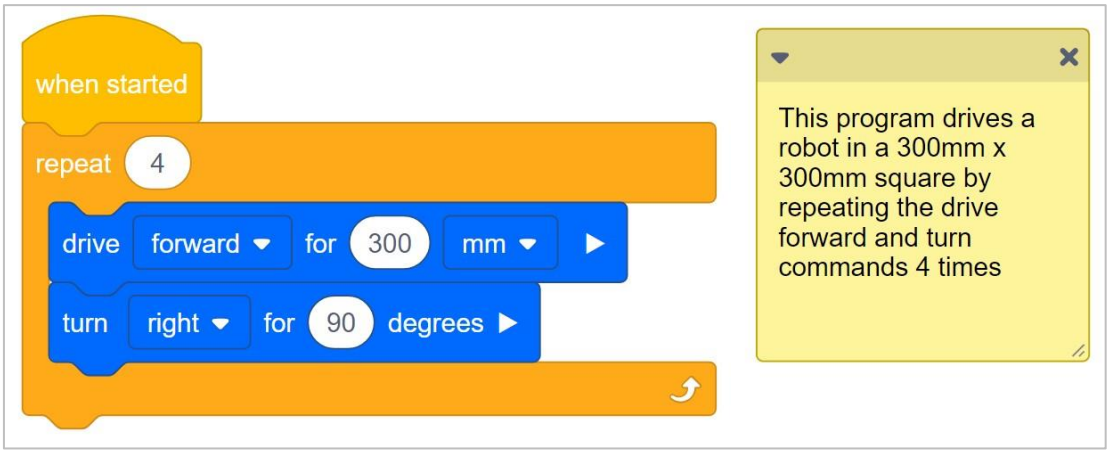

**Hardware/Software Required:**

| Quantity | Hardware/Other Items |
|---|---|
| 1 | VEX V5 Classroom Starter Kit (with up-to-date firmware) |
| 1 | VEXcode V5 Blocks (latest version, Windows, MacOS, Chromebook) |
| 1 | Engineering Notebook |
| 1 | Using Loops (Tutorial) |
| 1 | Repeating Actions (No Gyro) example project |

## 1. Let's start programming with loops.

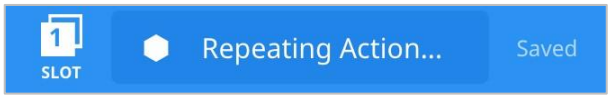
- Start by watching the **Using Loops** tutorial video.
- Open the **Repeating Actions (No Gyro)** example project.


- View the opened example project.




Wykonaj następujące zadania w notatniku technicznym.

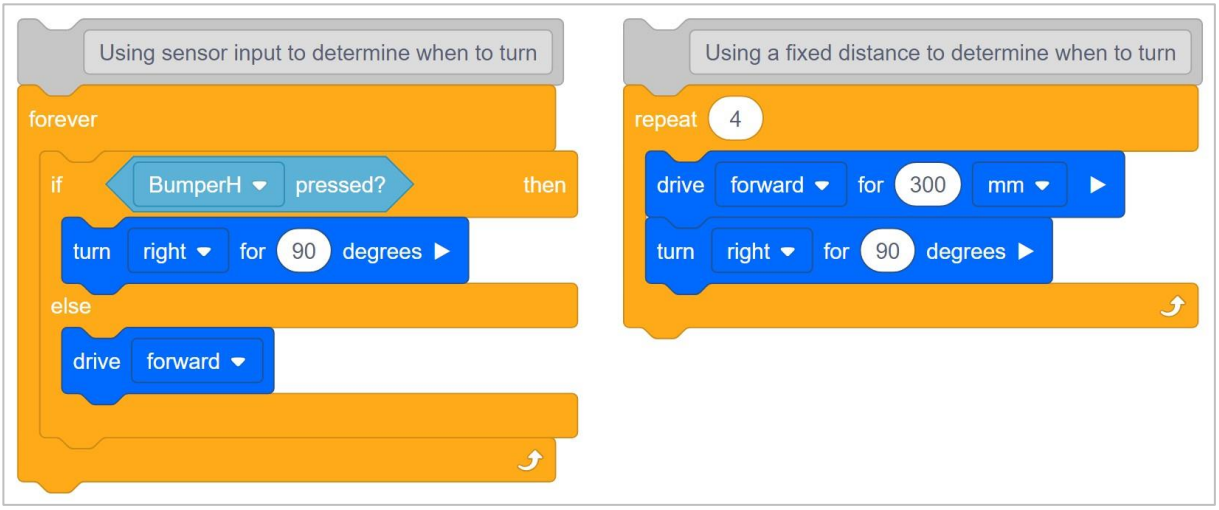
- Zastanów się, co zrobi projekt Clawbot. Wyjaśnij więcej niż fakt, że projekt znajdują się powtórzenia. Co się powtarza? Co robi Clawbot?
- Napisz swoją prognozę, ale nie dziel krótkiego projektu na więcej niż dwie części.
- Zapisz, pobierz i uruchom przykładowy projekt powtarzających się działań (bez żyroskopu).




- Aby uzyskać pomoc, zobacz samouczek dotyczący bloków VEXcode V5, który wyjaśnia, jak pobrać i uruchomić projekt.


- Sprawdź wyjaśnienia projektu w notatniku inżyniera i dodaj notatki, aby w razie potrzeby poprawić swoje przewidywania.

## 2. Uruchom projekt i obserwuj robota.



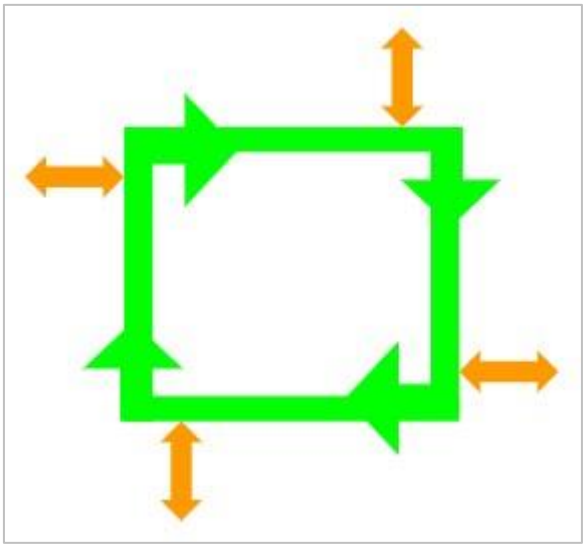

Spójrz ponownie na projekt. Zostanie w nim powtórzona czterokrotnie akcja jazdy do przodu. Blok powtórz jest używany, gdy chcesz użyć zestawu zachowań określoną liczbę razy. Jeśli blok powtórz zostanie zastąpiony blokiem na zawsze, robot będzie powtarzał ruch do przodu, a następnie będzie obracał się bez końca.

W projekcie po lewej, wejście czujnika służy do określenia, kiedy zacząć skręcać. Projekt po prawej używa stałej odległości *Drivetrain*, aby określić, kiedy zacząć skręcać.

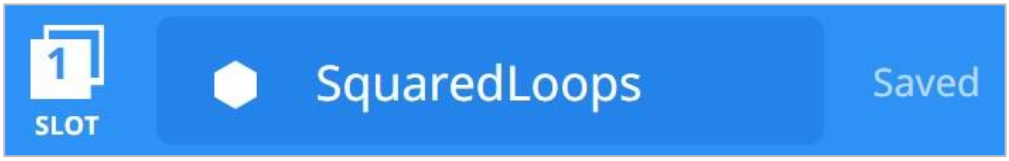
W celu ciągłego sprawdzania wejścia czujnika używany jest blok *jeżeli/w przeciwnym razie* z blokiem *zawsze*. W projekcie po lewej stronie robot skręci w prawo po naciśnięciu czujnika *BumperH*, w przeciwnym razie robot będzie jechał do przodu w nieskończoność, jeśli czujnik *BumperH* nie zostanie naciśnięty. Aby w sposób ciągły sprawdzać wartość czujnika *BumperH*, blok „*jeśli / to*” znajduje się w obrębie bloku na zawsze.

Powyższy projekt po lewej to praktyczny przypadek użycia struktury, która powtarza się w nieskończoność - używając jednocześnie bloku *na zawsze* i *jeżeli / wtedy*. Wyobraź sobie samojezdny odkurzacz, który jedzie do przodu, aż wpadnie na ścianę lub obiekt, a następnie skręca, zanim ruszy dalej.

## 3. Wyzwanie kwadratowej pętli!



- Niech twój Clawbot jedzie w kwadracie.
- Przed każdym obrotem kleszcze muszą być otwarte i zamknięte, a ramię podniesione i opuszczone.
- Clawbot może przejechać po każdej ścianie kwadratu tylko raz.
- Możesz użyć przykładowego projektu Powtarzające się działania (bez żyroskopu) jako punktu wyjścia, ale przed dokonaniem jakichkolwiek zmian zapisz go jako **SquaredLoops**.




W notatniku inżynieryjnym zaplanuj:

- Zaplanuj swoje rozwiązanie i przewiduj, co zrobi Clawbot w każdym bloku projektu.
- Pobierz i uruchom projekt, aby go przetestować go przed przesłaniem.
- W razie potrzeby wprowadź zmiany w projekcie i zanotuj je podczas testowania.

# Tekstowe programowanie pętli - VEXcode V5

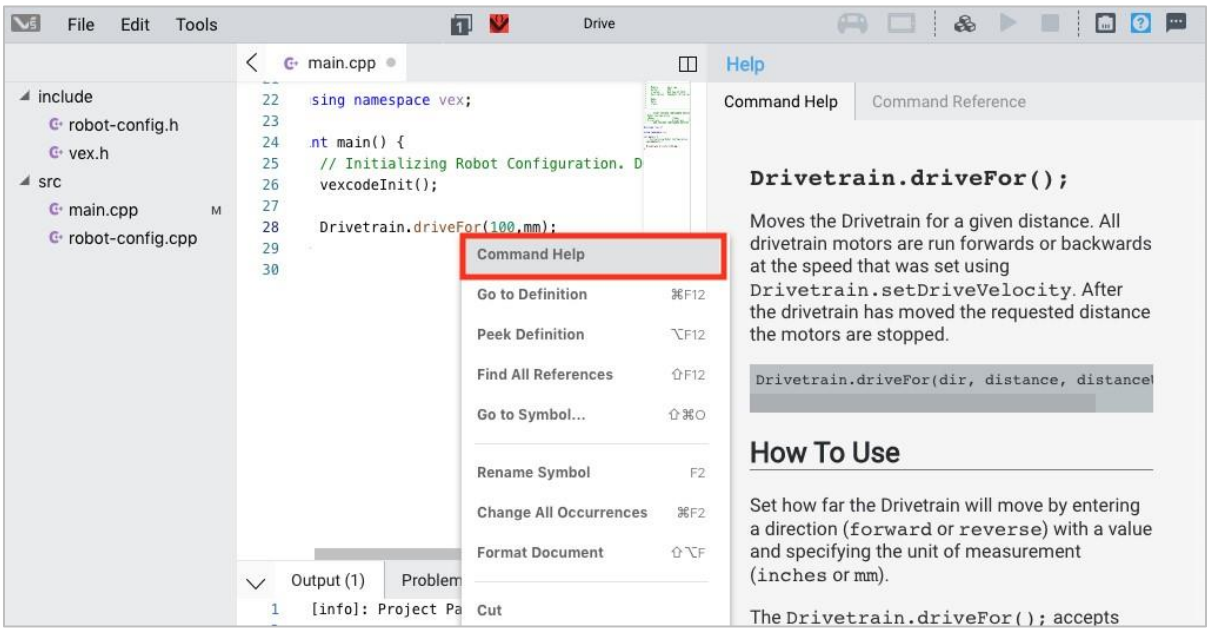
**Clawbot V5 jest gotowy do działania!**

Ta lekcja da ci narzędzia, dzięki którym będziesz mógł zacząć tworzyć projekty wykorzystujące pętle.

- Instrukcje tekstowe, które będą używane w tej eksploracji:

  - Drivetrain.driveFor (przód, 300, mm);
  - Drivetrain.turnFor (prawo, 90, stopnie);
  - ClawMotor.spinFor (do tyłu, 70, stopnie);
  - ArmMotor.spinFor (do przodu, 360 stopni);
  - while (true) {}
  - powtórz (4) {}
  - czekaj (5 sekund);

Aby uzyskać dostęp do dodatkowych informacji, kliknij prawym przyciskiem myszy nazwę polecenia w obszarze roboczym, aby wyświetlić pomoc.



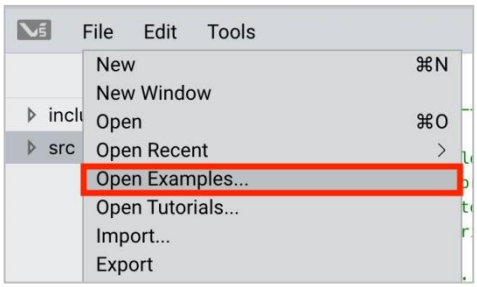

Upewnij się, że masz wymagany sprzęt, notes inżynieryjny i program do programowania tekstowego VEXcode V5 pobrany i gotowy.

**Wymagany sprzęt/oprogramowanie:**

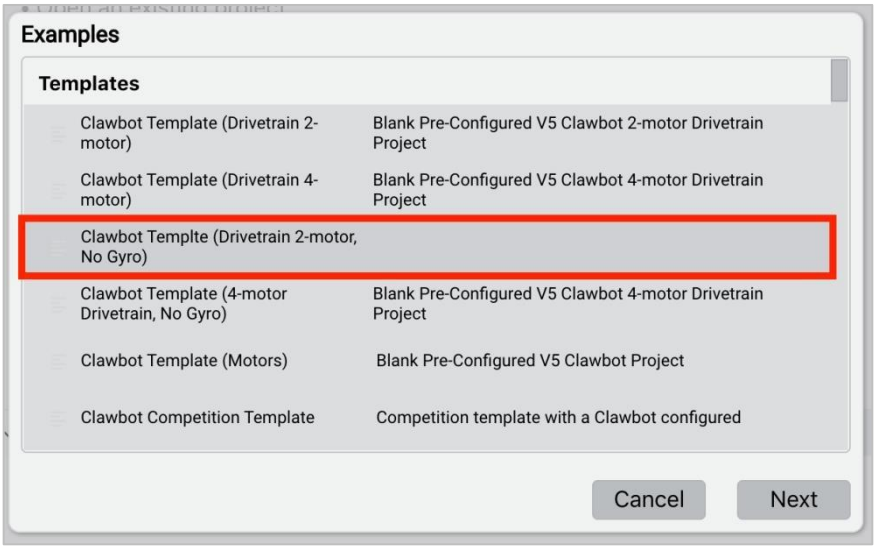
| Ilość | Hardware/Other Items |
|---|---|
| 1 | VEX V5 Classroom Starter Kit (z aktualnym oprogramowaniem) |
| 1 | VEXcode V5 Text (najnowsza wersja) |
| 1 | Notes inżyniera |
| 1 | Przykładowy projekt szblonu Clawbot Template (Drivetrain 2-motor, No Gyro) |

## 1. Zacznijmy programowanie z pętlami.

- Przed rozpoczęciem projektu wybierz prawidłowy szablon. Przykładowy projekt Clawbot Template (Drivetrain 2-motor, No Gyro) zawiera konfigurację silnika Clawbota. Jeśli szablon nie zostanie użyty, twój robot nie uruchomi projektu poprawnie.




- Wybierz **plik** i otwórz **przykłady.**
- Przewiń różne przykładowe **projekty**. Demonstrują one różnorodne działania, które może wykonać twój Clawbot. Wybierz i otwórz przykładowy projekt *Clawbot Template* (Drivetrain 2-motor, No Gyro).




Nazwij projekt RepeatingActions i wybierz opcję Utwórz.


Wpisz następujący kod:

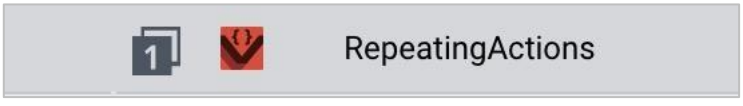
```
using namespace vex;

int main() {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();

  // drives forward and turns 90 degrees for 4 iterations
  repeat(4){
    Drivetrain.driveFor(forward, 300,mm);
    Drivetrain.turnFor(right, 90, degrees);
    wait(5, seconds);
  }
}
```

Wykonaj następujące zadania w notatniku technicznym.

- Zastanów się, co zrobi projekt Clawbot. Wyjaśnij więcej niż fakt, że projekt znajdują się powtórzenia. Co się powtarza? Co robi Clawbot?
- Napisz swoją prognozę, ale nie dziel krótkiego projektu na więcej niż dwie części.
- Zapisz, pobierz i uruchom przykładowy projekt powtarzających się działań (bez żyroskopu).




- Sprawdź wyjaśnienia projektu w notatniku inżyniera i dodaj notatki, aby w razie potrzeby poprawić swoje przewidywania.

## 2. Uruchom projekt i obserwuj robota.

**Using sensor input to determine when to turn**

```
#include "vex.h"

using namespace vex;

int main() {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();
// The robot turns right 90 degrees if the Bumper Switch is pressed
// Otherwise, the robot drives forward
while(true){
  if(BumperB.pressing()){
    Drivetrain.turnFor(right, 90, degrees);
  }
  else{
    Drivetrain.drive(forward);
    }
  }
}
```

**Using a fixed distance to determine when to turn**

```
#include "vex.h"

using namespace vex;

int main() {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();

  //drives forward and turns 90 degrees for 4 iterations
  repeat(4){
    Drivetrain.driveFor(forward, 300, mm);
    Drivetrain.turnFor(right, 90, degrees);
    wait(5, seconds);
  }
}
```

Spójrz ponownie na projekt po prawej. Zostanie w nim powtórzona czterokrotnie akcja jazdy do przodu. Akcja *powtórz* jest używana, gdy chcesz użyć zestawu zachowań określoną liczbę razy.

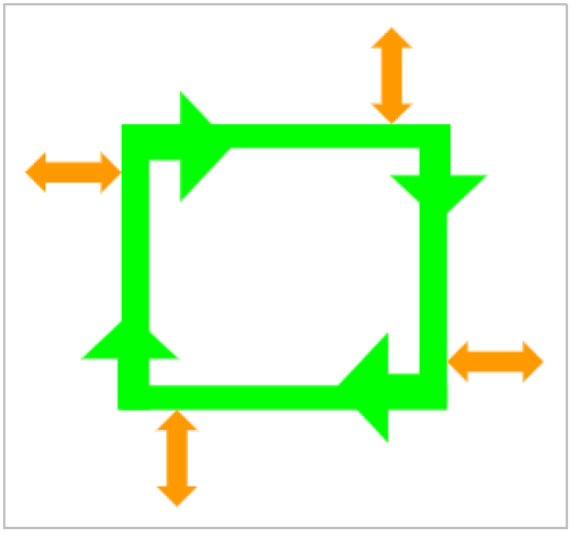
Jeśli struktura zostanie zastąpiona strukturą pętli *„kiedy” - "while",* robot powtórzy jazdę do przodu, a następnie obrót, „*kiedy”* warunek jest prawdziwy. Możesz również ustawić warunek na „*prawda*”, aby pętla „*kiedy*” była kontynuowana w nieskończoność.

W projekcie po lewej, wejście czujnika służy do określenia, kiedy zacząć skręcać. Projekt po prawej używa stałej odległości Drivetrain, aby określić, kiedy zacząć skręcać.

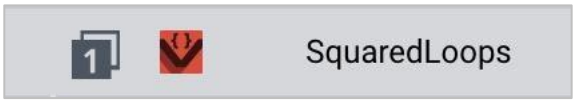
W celu ciągłego sprawdzania wejścia czujnika używany jest blok jeżeli/w przeciwnym razie z komendą *zawsze*. W projekcie po lewej stronie robot skręci w prawo po naciśnięciu czujnika *BumperB,* w przeciwnym razie robot będzie jechał do przodu w nieskończoność, jeśli czujnik *BumperB* nie zostanie naciśnięty. Aby w sposób ciągły sprawdzać wartość czujnika *BumperB*, instrukcja *„if” - ,,jeżeli"* znajduje się w pętli *„while true” ("kiedy prawda").*

Powyższy projekt po lewej to praktyczny przypadek użycia struktury, która powtarza się w nieskończoność - używając jednocześnie komendę *na zawsze i jeżeli / wtedy*. Wyobraź sobie samojezdny odkurzacz, który jedzie do przodu, aż wpadnie na ścianę lub obiekt, a następnie skręca, zanim ruszy dalej.

## 3. Wyzwanie kwadratowej pętli



- Niech twój Clawbot jedzie w kwadracie.
- Przed każdym obrotem kleszcze muszą być otwarte i zamknięte, a ramię podniesione i opuszczone.
- Clawbot może przejechać po każdej ścianie kwadratu tylko raz.
- Możesz użyć przykładowego projektu Powtarzające się działania (bez żyroskopu) jako punktu wyjścia, ale przed dokonaniem jakichkolwiek zmian zapisz go jako *SquaredLoops.*




W notatniku inżynieryjnym zaplanuj:

- Zaplanuj swoje rozwiązanie i przewiduj, co zrobi Clawbot w każdym bloku projektu.
- Pobierz i uruchom projekt, aby go przetestować go przed przesłaniem.
- W razie potrzeby wprowadź zmiany w projekcie i zanotuj je podczas testowania.


**Rozwiąż inne problemy XXI wieku,**
**stosując podstawowe umiejętności**
**i koncepcje, których się nauczyłeś.**

# Roboty na produkcji

*Roboty pracujące w zakładzie produkcyjnym.*

## Roboty fabryczne

Fabryki po raz pierwszy zaczęły używać nowoczesnych robotów przemysłowych na początku lat 60. Mogą one wykonywać brudne, nudne i niebezpieczne prace, które wcześniej wykonywali ludzie. Od tego czasu fabryki na całym świecie zainwestowały miliony dolarów w rozwój i budowanie robotów, które mogą szybko i wydajnie wytwarzać produkty.

Roboty fabryczne są zawsze ulepszane wraz z rozwojem nowych technologii. Nowe metale i materiały pozwalają na stosowanie robotów w środowiskach o wysokim ciśnieniu lub wysokiej temperaturze. Zwykle są odseparowane, aby zapewnić pracownikom bezpieczeństwo w razie wypadku, roboty fabryczne są wykonywane z nowych „miękkich” materiałów. Materiały te, takie jak guma i plastik, mogą pomóc zmniejszyć obrażenia w przypadku zderzenia robota z człowiekiem. Dzięki wprowadzeniu sztucznej inteligencji i czujników roboty fabryczne można „nauczyć” nowych sposobów tworzenia produktów z dnia na dzień i dostosowywania ich ruchów w czasie rzeczywistym. Pozwala to na większą produktywność i precyzję.

Roboty fabryczne są wykorzystywane do produkcji wielu produktów, ale trzy najważniejsze ich role w produkcji to:

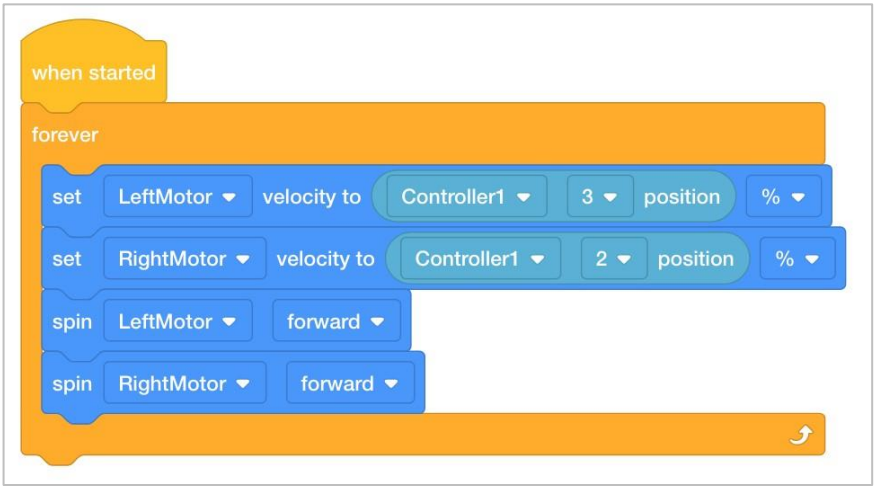
- Wiercenie
- Spawanie
- Malowanie i uszczelnianie

# Kontrolery i pętle - bloki VEXcode V5

## Kontrolery i pętle

Podczas zawodów zespoły używają kontrolerów do bezprzewodowego sterowania robotami. Sterownik jest zaprogramowany do aktualizacji robota na podstawie danych wprowadzonych przez użytkownika. W projekcie używane są pętle, dzięki czemu robot wielokrotnie sprawdza dostępność zaktualizowanych informacji wejściowych. Pętle pozwalają projektowi szybko sprawdzić, które przyciski zostały naciśnięte lub jak daleko wciśnięto joysticki. Po sprawdzeniu informacje te są szybko przekazywane do robota, aby szybko reagował na polecenia kontrolera.

Poniższy obrazek przedstawia przykładowy projekt *Tank Drive* w VEXcode V5. Pętla typu *na zawsze*, w tym projekcie, sprawdza pozycje osi 2 i 3 w celu ustawienia prędkości silników.




*Przykładowy projekt Tank Drive z VEXcode V5 z blokami*

Pętle są ważne nawet w przypadku programowania autonomicznego bez kontrolera. Pomaga uprościć i uporządkować powtarzane polecenia w projekcie.

# Kontrolery i pętle – programowanie tekstowe VEXcode V5

## Kontrolery i pętle

Podczas zawodów zespoły używają kontrolerów do bezprzewodowego sterowania robotami. Sterownik jest zaprogramowany do aktualizacji robota na podstawie danych wprowadzonych przez użytkownika. W projekcie używane są pętle, dzięki czemu robot wielokrotnie sprawdza dostępność zaktualizowanych informacji wejściowych. Pętle pozwalają projektowi szybko sprawdzić, które przyciski zostały naciśnięte lub jak daleko wciśnięto joysticki. Po sprawdzeniu informacje te są szybko przekazywane do robota, aby szybko reagował na polecenia kontrolera.

Poniższy obrazek przedstawia przykładowy projekt *Tank Drive* w VEXcode V5. Pętla typu *na zawsze*, w tym projekcie, sprawdza pozycje osi 2 i 3 w celu ustawienia prędkości silników.

```
int main() {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();

  // Deadband stops the motors when Axis values are close to zero.
  int deadband = 5;
  while (true) {
    // Get the velocity percentage of the left motor. (Axis3)
    int leftMotorSpeed = Controller1.Axis3.position();
    // Get the velocity percentage of the right motor. (Axis2)
    int rightMotorSpeed = Controller1.Axis2.position();

    // Set the speed of the left motor. If the value is less than the deadband,
    // set it to zero.
    if (abs(leftMotorSpeed) < deadband) {
      // Set the speed to zero.
      LeftMotor.setVelocity(0, percent);
    } else {
      // Set the speed to leftMotorSpeed
      LeftMotor.setVelocity(leftMotorSpeed, percent);
    }
    // Set the speed of the right motor. If the value is less than the deadband,
    // set it to zero.
    if (abs(rightMotorSpeed) < deadband) {
      // Set the speed to zero
      RightMotor.setVelocity(0, percent);
    } else {
      // Set the speed to rightMotorSpeed
      RightMotor.setVelocity(rightMotorSpeed, percent);
    }
    // Spin both motors in the forward direction.
    LeftMotor.spin(forward);
    RightMotor.spin(forward);
    wait(25, msec);
  }
}
```

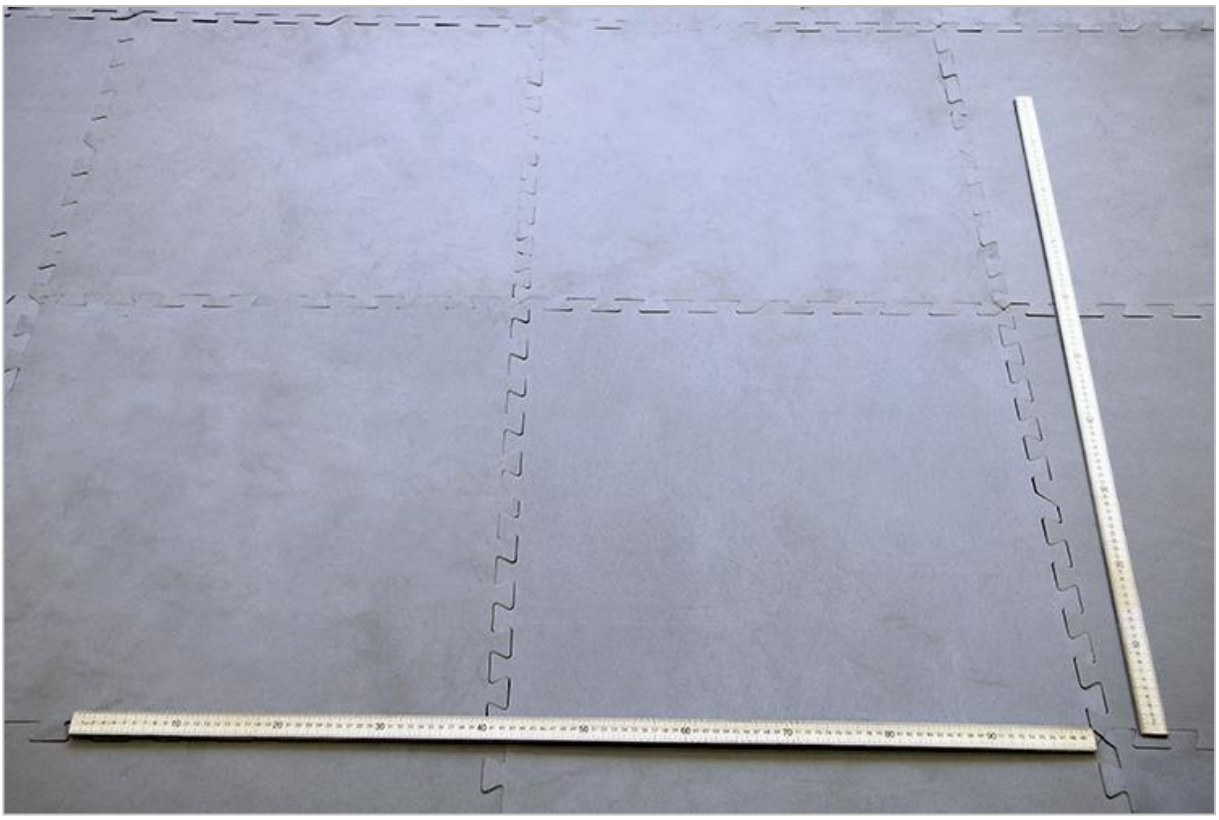
*Przykładowy projekt Tank Drive z VEXcode V5 – programowanie tekstowe*

Pętle są ważne nawet w przypadku programowania autonomicznego bez kontrolera. Pomaga uprościć i uporządkować powtarzane polecenia w projekcie.

SPARK
RETHINK

**Czy istnieje skuteczniejszy sposób, aby dojść do tego samego wniosku? zastanów się nad tym czego się nauczyłeś i spróbuj to ulepszyć.**

# Przygotuj się na wyzwanie maszyny bujającej się do rytmu



*Rozłożenie parkietu*

## Przygotuj się na wyzwanie maszyny bujającej się do rytmu

W zadaniu maszyny bujającej się do rytmu, podzielicie się na grupy i zaprogramujecie robota tak, aby przeszedł przez układ taneczny, korzystając z wiedzy o pętlach.

Aby wykonać wyzwanie, będziesz musiał zwolnić miejsce na podłodze wystarczająco duże, aby Clawbot IQ mógł poruszać się bez wpadania na nic. Zaleca się obszar to 1x1 metra, aby każdy Clawbot miał odpowiednią przestrzeń do poruszania się.

# Projektuj, rozwijaj i powtarzaj swój projekt - bloki VEXcode V5

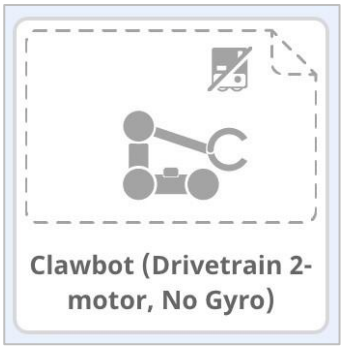
Podczas projektowania odpowiedz na poniższe pytania w swoim notatniku technicznym.

- Jaki rodzaj tańca robotów stworzysz? Wyjaśnij szczegółowo.
- Jakich rodzajów pętli będziesz używać i dlaczego?
- Jakie kroki podejmiesz, aby przetestować taniec? Wyjaśnij szczegółowo.

Aby pomóc Ci w planowaniu, kliknij tutaj, aby zobaczyć kilka przykładowych ruchów tanecznych, które możesz uwzględnić w tańcu Clawbota.

Podczas tworzenia projektu wykonaj poniższe czynności:

- Zaplanuj taniec za pomocą rysunków i pseudokodu.
- Użyj utworzonego pseudokodu, aby opracować projekt przy użyciu bloków VEXcode V5.
- Otwórz przykładowy projekt Clawbot (Drivetrain 2-motor, No Gyro).

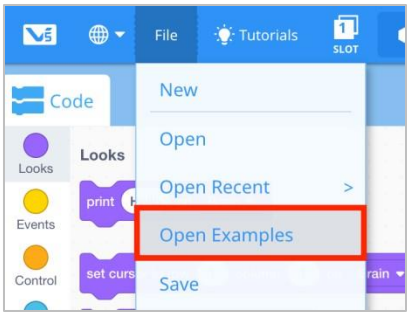



- Zapisz swój projekt jako *GrooveMachine*.


- Uruchom swój projekt, aby często go testować i iteruj go, korzystając z tego, czego nauczyłeś się podczas testów.
- Podziel się końcowym projektem z nauczycielem.

Jeśli masz problemy z rozpoczęciem, zapoznaj się z poniższymi informacjami w VEXcode V5:

- Przykładowe projekty




- Samouczek: korzystanie z pętli


- Poprzednie wersje Twojego projektu
- Funkcja pomocy, aby dowiedzieć się więcej o blokach

# Wyzwanie maszyny bujającej się do rytmu

*V5 Clawbot z podniesionym ramieniem I otwartymi kleszczami*

## Wyzwanie maszyny bujającej się do rytmu

W tym wyzwaniu podzielicie się na zespoły i zaprogramujecie robota tak, aby wykonał układ taneczny, korzystając z wiedzy o pętlach. Twój nauczyciel wyznaczy limit czasu na rozwijanie / testowanie tańca oraz limit czasu na długość tańca. Wszyscy, którzy nie są w rywalizujących drużynach tanecznych, będą oceniać je i głosować na drużynę, która według nich jest najlepsza.

**Zasady:**

- Każdy Clawbot będzie tańczył sam na obszarze 1x1 metra.
- Taniec trwa do momentu naciśnięcia przycisku Stop na ekranie Mózgu, aby zatrzymać działanie projektu.
- Ramię należy podnieść i opuścić.
- Kleszcze muszą się otwierać i zamykać.
- Clawbot musi skręcić w lewo i w prawo.
- Clawbot musi jechać do przodu i do tyłu.
- Projekt musi zostać natychmiast zatrzymany, jeśli Clawbot zderzy się z czymś lub się przewróci. Taniec jest wtedy unieważniony.

# Projektuj, rozwijaj i powtarzaj swój projekt – programowanie tekstowe

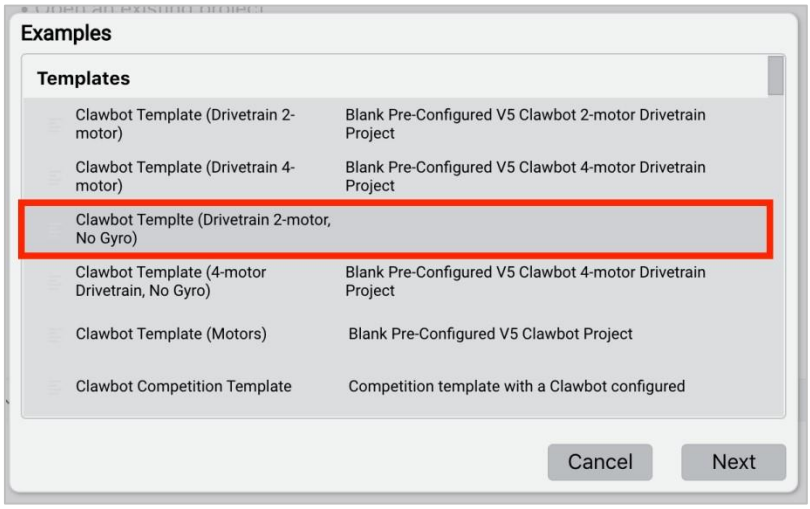
Podczas projektowania odpowiedz na poniższe pytania w swoim notatniku technicznym.

- Jaki rodzaj tańca robotów stworzysz? Wyjaśnij szczegółowo.
- Jakich rodzajów pętli będziesz używać i dlaczego?
- Jakie kroki podejmiesz, aby przetestować taniec? Wyjaśnij szczegółowo.

Aby pomóc Ci w planowaniu, kliknij tutaj, aby zobaczyć kilka przykładowych ruchów tanecznych, które możesz uwzględnić w tańcu Clawbota.

Podczas tworzenia projektu wykonaj poniższe czynności:

- Zaplanuj taniec za pomocą rysunków i pseudokodu.
- Użyj utworzonego pseudokodu, aby opracować projekt używając programowania tekstowego VEXcode V5.
- Otwórz przykładowy projekt Clawbot (Drivetrain 2-motor, No Gyro).

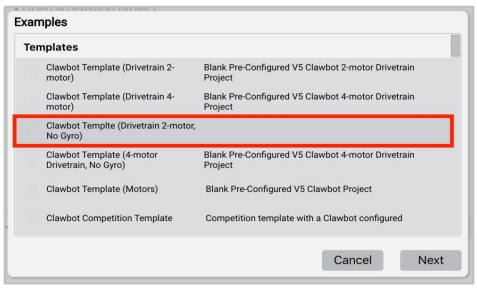



- Zapisz swój projekt jako GrooveMachine.

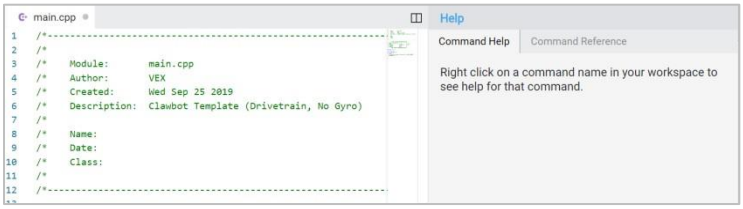

- Uruchom swój projekt, aby często go testować i iteruj go, korzystając z tego, czego nauczyłeś się podczas testów.
- Podziel się końcowym projektem z nauczycielem.

Jeśli masz problemy z rozpoczęciem, zapoznaj się z poniższymi informacjami w VEXcode V5:

- Przykładowe projekty:

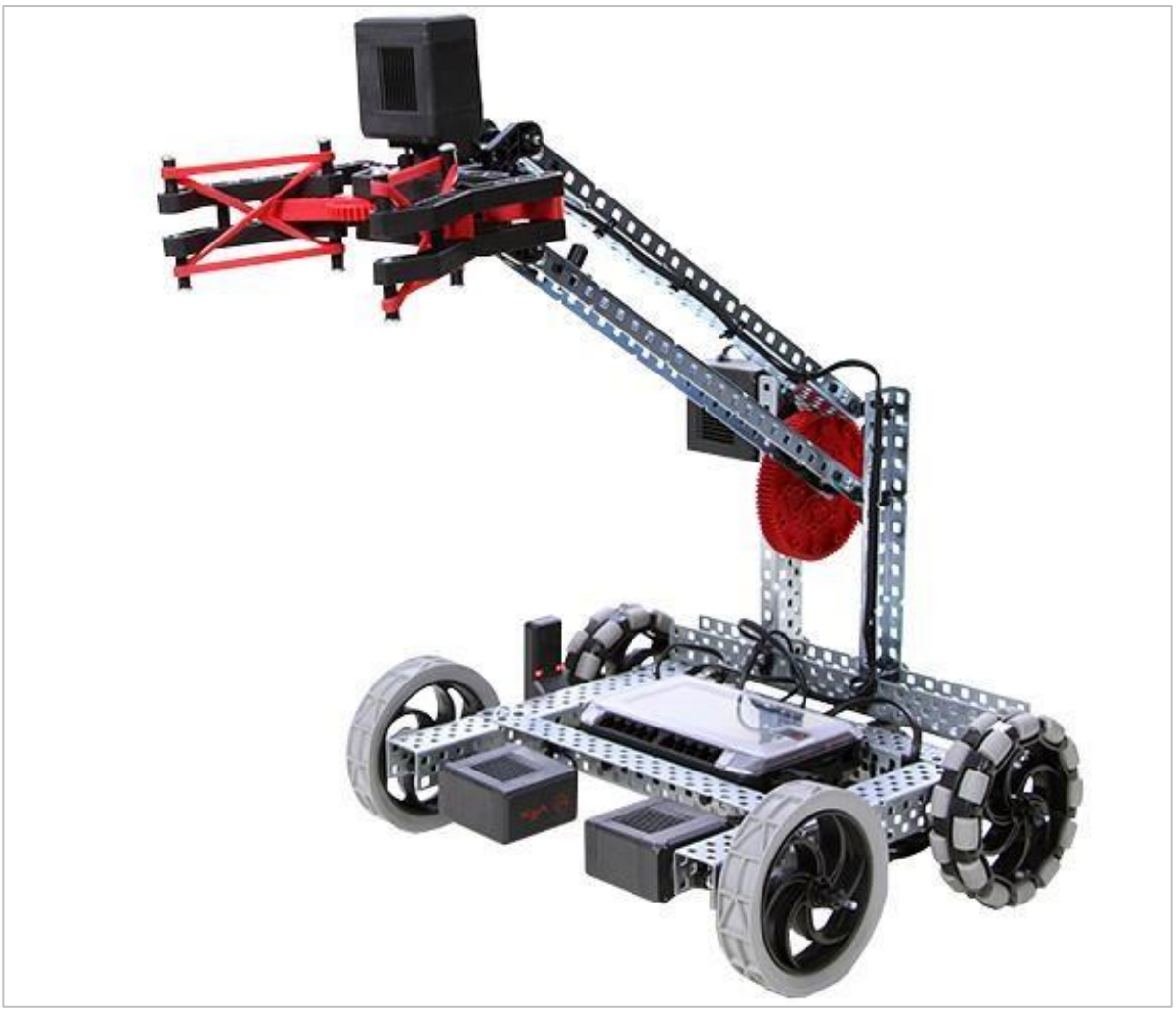



- Aby uzyskać dostęp do dodatkowych informacji podczas tworzenia programu, kliknij prawym przyciskiem myszy na instrukcję w swojej przestrzeni roboczej, aby wyświetlić dodatkowe informacje na jej temat.




- Przejrzyj poprzednie wersje projektu RepeatingActions, aby stworzyć nowy.

# Wyzwanie maszyny bujającej się do rytmu- VEXcode V5 programowanie tekstowe



*V5 Clawbot z podniesionym ramieniem I otwartymi kleszczami*

## Wyzwanie maszyny bujającej się do rytmu

W tym wyzwaniu podzielicie się na zespoły i zaprogramujecie robota tak, aby wykonał układ taneczny, korzystając z wiedzy o pętlach. Twój nauczyciel wyznaczy limit czasu na rozwijanie / testowanie tańca oraz limit czasu na długość tańca. Wszyscy, którzy nie są w rywalizujących drużynach tanecznych, będą oceniać je i głosować na drużynę, która według nich jest najlepsza.

**Zasady:**

- Każdy Clawbot będzie tańczył sam na obszarze 1x1 metra.
- Taniec trwa do momentu naciśnięcia przycisku Stop na ekranie Mózgu, aby zatrzymać działanie projektu.
- Ramię należy podnieść i opuścić.
- Kleszcze muszą się otwierać i zamykać.
- Clawbot musi skręcić w lewo i w prawo.
- Clawbot musi jechać do przodu i do tyłu.
- Projekt musi zostać natychmiast zatrzymany, jeśli Clawbot zderzy się z czymś lub się przewróci. Taniec jest wtedy unieważniony.

SPARK
KNOW

**Zrozum podstawowe pojęcia i dowiedz się, jak zastosować je w różnych sytuacjach. Ten proces powtórki będzie motywował do nauki.**

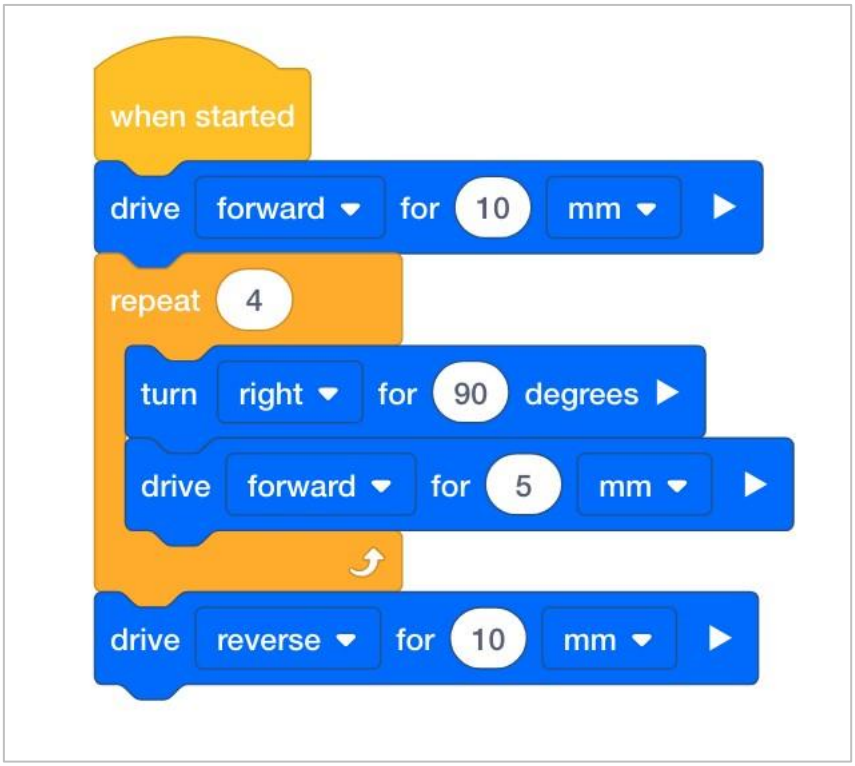
# Powtórzenie – bloki VEXcode V5

1. **Prawda czy fałsz: Kontrolery używają pętli w celu sprawdzenia odpowiedzi wejściowych z przycisków i joysticków.**
   - Prawda
   - Fałsz

2. **Pętle to _____.**
   - dokumentacje mówiące o dostępnych typach danych
   - warunkowe wykonywanie grup instrukcji
   - instrukcje obsługi występujących błędów
   - struktury programistyczne, które powtarzają zachowania

3. **Prawda czy fałsz: W VEXcode V5 bloki powtórz i pętla na zawsze powtarzają wszystkie instrukcje wewnątrz nich, przez określoną liczbę razy. Instrukcje wewnątrz bloku powtórz i pętli na zawsze są uruchamiane w kolejności od góry do dołu.**
   - Prawda
   - Fałsz

4. **W poniższym projekcie, ile razy robot skręca w prawo?**



- Robot skręci w prawo 9 razy, ponieważ jest ustawiony na obrót o 90 stopni.
- Robot skręci raz w prawo, ponieważ jest tylko jeden blok obróć w projekcie.
- Robot skręci w prawo 4 razy, ponieważ blok powtórz ma parametr 4 razy.
- Robot skręci w prawo 5 razy, ponieważ blok jedź o przesuwa go o 5 mm.

**5. Prawda czy fałsz: Roboty nie radzą sobie dobrze z powtarzalnymi zadaniami, tak jak ludzie, ponieważ każdy drobny błąd w ich programowaniu zwiększa się z każdym powtórzeniem..**

- Prawda
- Fałsz

# Powtórzenie – VEXcode V5 programowanie tekstowe

6. **Prawda czy fałsz: Kontrolery używają pętli w celu sprawdzenia odpowiedzi wejściowych z przycisków i joysticków.**
   - Prawda
   - Fałsz

7. **Pętle to _____.**
   - dokumentacje mówiące o dostępnych typach danych
   - warunkowe wykonywanie grup instrukcji
   - instrukcje obsługi występujących błędów
   - struktury programistyczne, które powtarzają zachowania

8. **Prawda czy fałsz: W tekście VEXcode V5 struktury powtórz i pętle na zawsze powtarzają wszystkie instrukcje wewnątrz nich, przez określoną liczbę razy. Instrukcje wewnątrz struktury powtórz i pętli na zawsze są uruchamiane w kolejności od góry do dołu.**
   - Prawda
   - Fałsz

9. **W poniższym projekcie, ile razy robot skręca w prawo?**
   - Robot skręci w prawo 9 razy, ponieważ jest ustawiony na obrót o 90 stopni.
   - Robot skręci raz w prawo, ponieważ jest tylko jedna instrukcja obróć w projekcie.
   - Robot skręci w prawo 4 razy, ponieważ struktura powtórz posiada parametr 4 razy.
   - Robot skręci w prawo 5 razy, ponieważ instrukcja jedź o przesuwa go o 5 mm.

```
22  #include "vex.h"
23
24  using namespace vex;
25
26  int main() {
27    // Initializing Robot Configuration. DO NOT REMOVE!
28    vexcodeInit();
29
30    Drivetrain.driveFor(forward, 100, mm);
31    repeat(4){
32      Drivetrain.turnFor(right, 90, degrees);
33      Drivetrain.driveFor(forward, 50, mm);
34      wait(5, msec);
35    }
36    Drivetrain.driveFor(reverse, 100, mm);
37  }
```

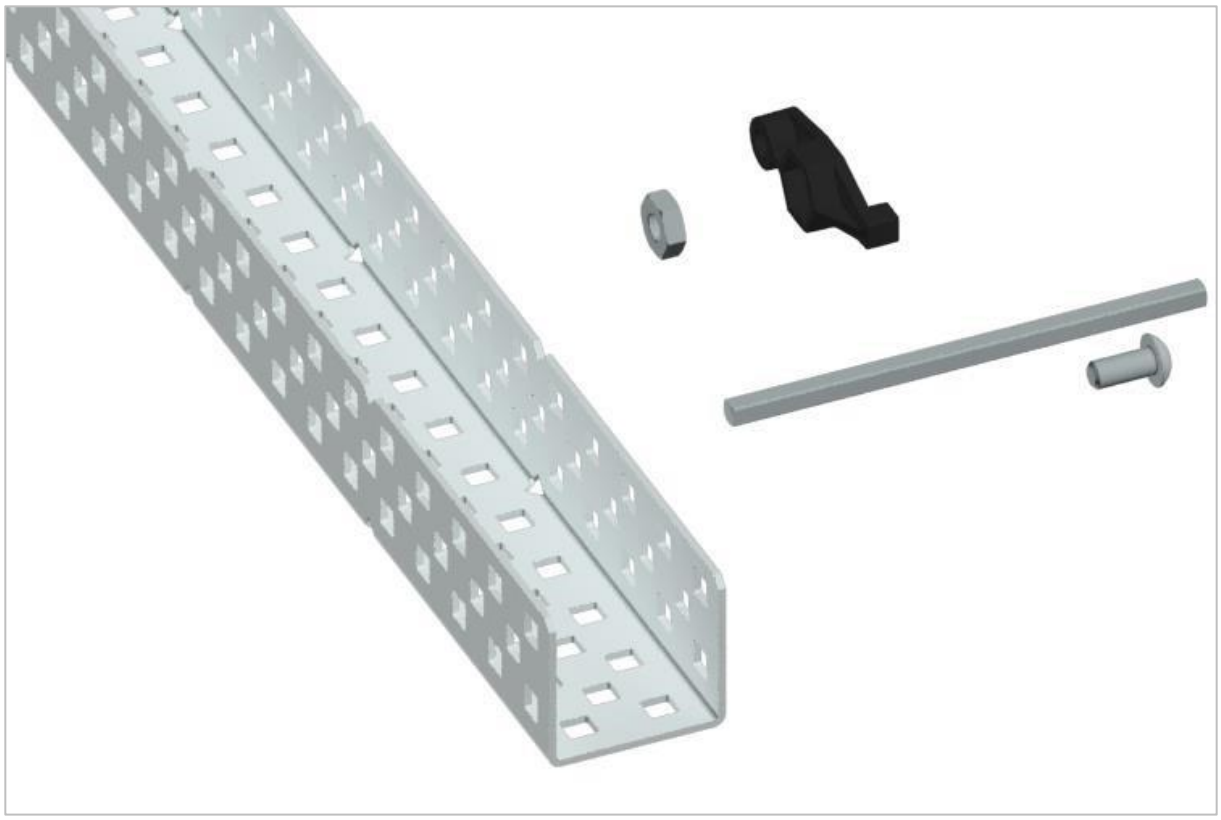
10. **Prawda czy fałsz: Roboty nie radzą sobie dobrze z powtarzalnymi zadaniami, tak jak ludzie, ponieważ każdy drobny błąd w ich programowaniu zwiększa się z każdym powtórzeniem.**

    - Prawda
    - Fałsz

# APPENDIX

## Dodatkowe informacje, zasoby i materiały.

# Używanie 1 Post Hex Nut Retainer w/ Bearing Flat



*1 Post Hex Nut Retainer w/ Bearing Flat*

## Używanie 1 Post Hex Nut Retainer w/ Bearing Flat

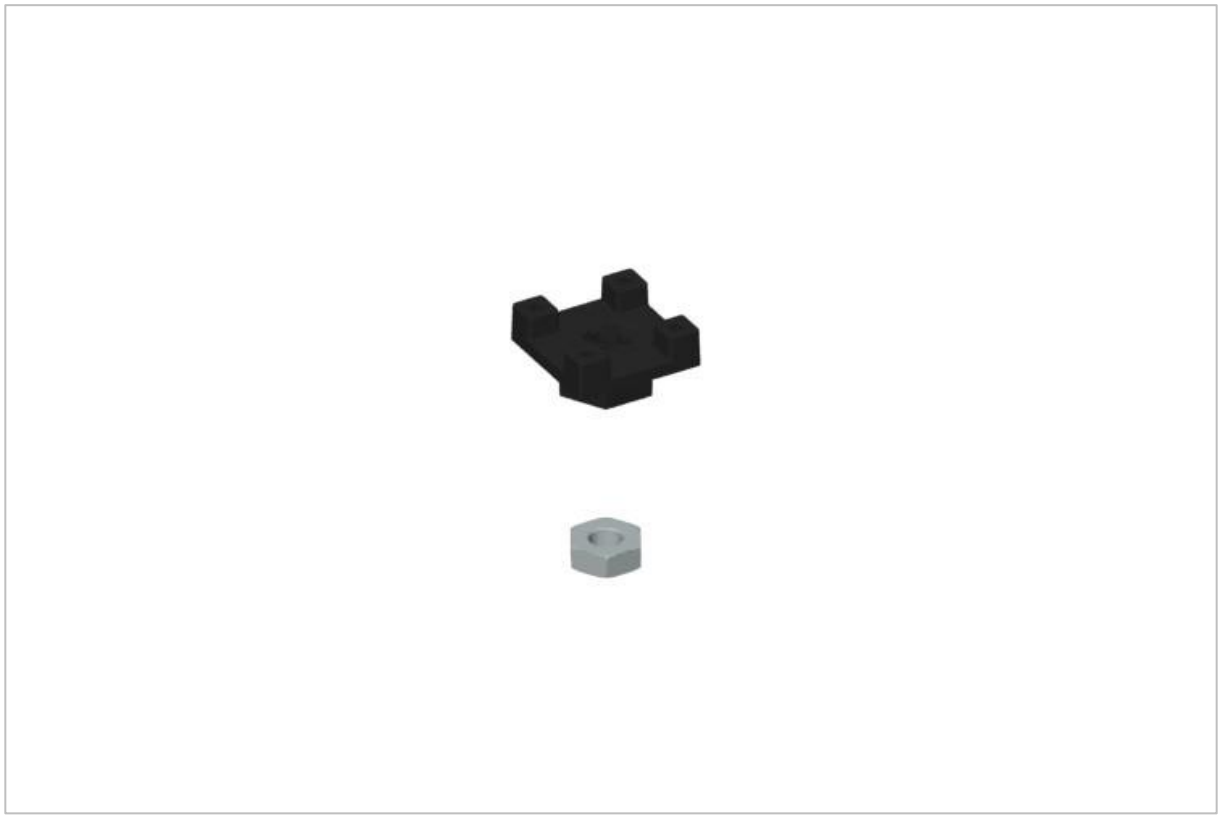
1 Post Hex Nut Retainer w/ Bearing Flat umożliwia płynne obracanie się wałów przez otwory w elementach konstrukcyjnych. Po zamontowaniu zapewnia dwa punkty styku na elementach konstrukcyjnych w celu zapewnienia stabilności. Na jednym końcu elementu znajduje się słupek o wymiarach umożliwiających bezpieczne dopasowanie do kwadratowego otworu elementu konstrukcyjnego. Środkowy otwór elementu ma taki rozmiar i szczelinę, aby bezpiecznie dopasować nakrętkę sześciokątną, umożliwiając łatwe dokręcenie śruby 8-32 bez użycia klucza lub kombinerek. Otwór na końcu elementu jest przeznaczony do przechodzenia wałków lub śrub.

Aby skorzystać z retainera:

- Wyrównaj go na elemencie konstrukcyjnym VEX tak, aby otwór końcowy znajdował się w żądanym miejscu, a sekcja środkowa i końcowa również były podparte przez element konstrukcyjny.

- Włóż kwadratowy słupek wystający z elementu do elementu konstrukcyjnego, aby pomóc go utrzymać na miejscu.
- Włożyć nakrętkę sześciokątną w środkową część retainera tak aby zrównała się z resztą elementu.
- W stosownych przypadkach wyrównaj wszelkie dodatkowe elementy konstrukcyjne z tyłu głównego elementu konstrukcyjnego.
- Użyj śruby 8-32 odpowiedniej długości, aby przymocować elementy konstrukcyjne do retainera przez środkowy otwór i nakrętkę sześciokątną.

# Używanie 4 Post Hex Nut Retainer



*4 Post Hex Nut Retainer*

## Używanie 4 Post Hex Nut Retainer

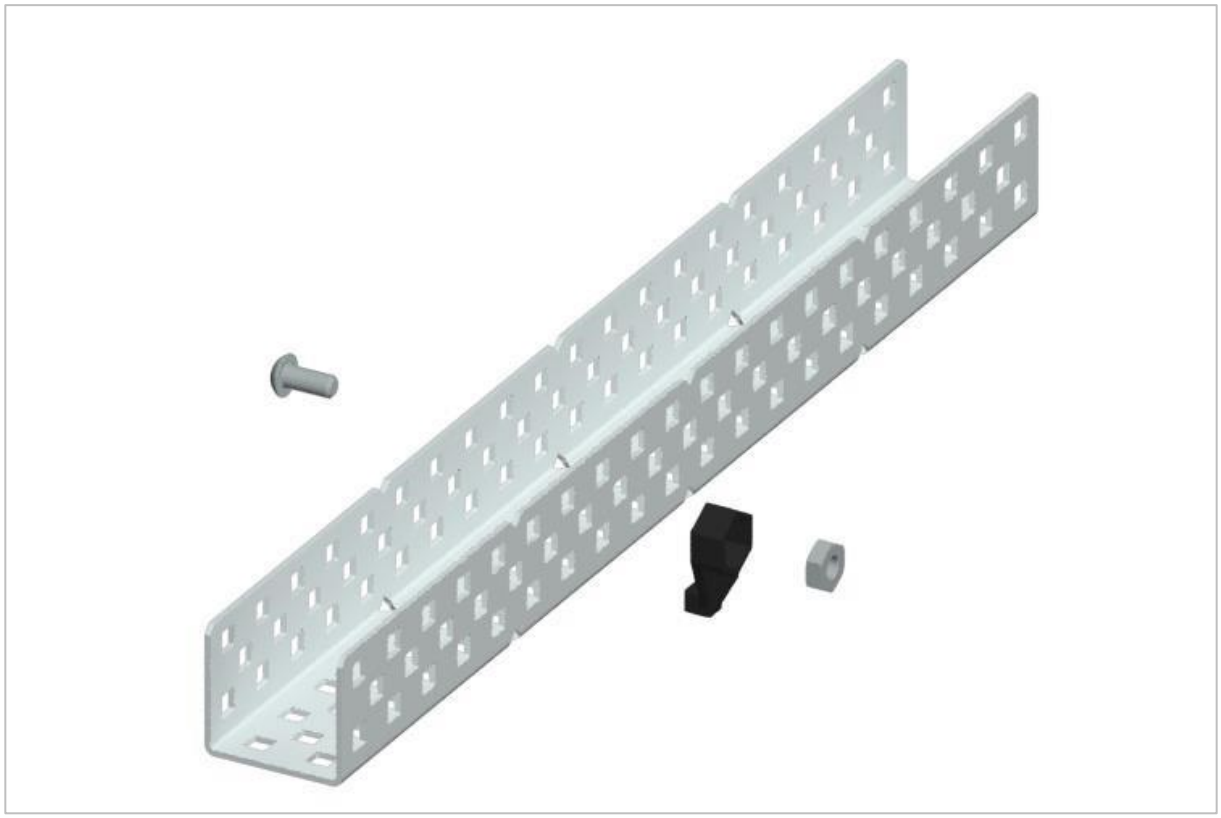
4 Post Hex Nut Retainer zapewnia pięć punktów styku do tworzenia wytrzymałego połączenia między dwoma elementami konstrukcyjnymi za pomocą jednej śruby i nakrętki. Każdy narożnik retainera zawiera słupek o rozmiarze umożliwiającym bezpieczne dopasowanie go do kwadratowego otworu w elemencie konstrukcyjnym. Środek elementu ma taki rozmiar i szczelinę, aby bezpiecznie dopasować nakrętkę sześciokątną, umożliwiając łatwe dokręcenie śruby 8-32 bez użycia klucza lub kombinerek.

Aby skorzystać z retainera:

- Wyrównaj go na elemencie konstrukcyjnym VEX tak, aby środkowy otwór znajdował się w żądanym miejscu, a każdy narożnik był wsparty elementem konstrukcyjnym.
- Włóż kwadratowy słupek wystający z elementu do elementu konstrukcyjnego, aby pomóc go utrzymać na miejscu.
- Włożyć nakrętkę sześciokątną w środkową część retainera tak aby zrównała się z resztą elementu.

- W stosownych przypadkach wyrównaj wszelkie dodatkowe elementy konstrukcyjne z tyłu głównego elementu konstrukcyjnego.
- Użyj śruby 8-32 odpowiedniej długości, aby przymocować elementy konstrukcyjne do retainera przez środkowy otwór i nakrętkę sześciokątną.

# Używanie 1 Post Hex Nut Retainer



*1 Post Hex Nut Retainer*

## Używanie 1 Post Hex Nut Retainer

1 Post Hex Nut Retainer zapewnia dwa punkty styku do łączenia elementu konstrukcyjnego z innym elementem za pomocą jednej śruby i nakrętki. Jeden koniec elementu zawiera słupek o wymiarach umożliwiających bezpieczne dopasowanie do kwadratowego otworu elementu konstrukcyjnego. Drugi koniec retainera ma taki rozmiar i szczelinę, aby bezpiecznie dopasować nakrętkę sześciokątną, umożliwiając łatwe dokręcenie śruby 8-32 bez użycia klucza lub kombinerek.

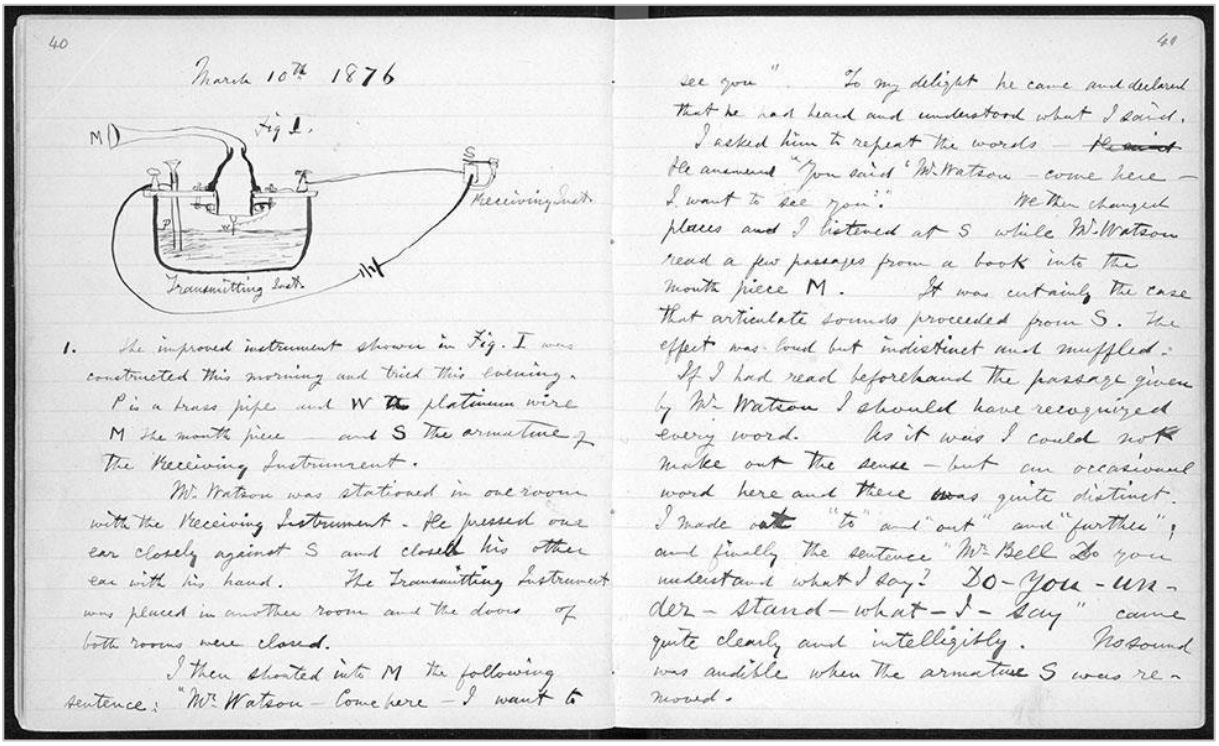
Aby skorzystać z retainera:

- Wyrównaj go na elemencie konstrukcyjnym VEX tak, aby oba końce były podparte przez element konstrukcyjny i ustawione tak, aby zamocować drugi element.
- Włóż kwadratowy słupek wystający z retainera do elementu konstrukcyjnego, aby pomóc go utrzymać na miejscu.
- Jeśli element jest używany do mocowania dwóch elementów konstrukcyjnych, włóż nakrętkę sześciokątną do drugiego końca retainera, tak aby był wyrównany z resztą

elementu. W przypadku użycia do zamocowania innego rodzaju elementu, takiego jak np.: element dystansowy, może być właściwe włożenie śruby przez tę stronę.

- Jeśli jest to niezbędne, wyrównaj wszelkie dodatkowe komponenty z tyłu głównego elementu konstrukcyjnego.
- Jeśli retainer jest używany do łączenia dwóch elementów konstrukcyjnych, użyj śruby 8-32 o odpowiedniej długości, aby zabezpieczyć elementy konstrukcyjne przez otwór i nakrętkę sześciokątną. Jeśli jest używany do łączenia innego typu elementu, takiego jak np.: element dystansowy, zabezpiecz go bezpośrednio lub za pomocą nakrętki sześciokątnej.

# Notes inżyniera



March 10th 1876

Fig 1.

M

S

Receiving Instr.

Transmitting Instr.

1. The improved instrument shown in Fig. I was constructed this morning and tried this evening. P is a brass pipe and W the platinum wire M the mouth piece — and S the armature of the Receiving Instrument.

Mr. Watson was stationed in one room with the Receiving Instrument. He pressed one ear closely against S and closed his other ear with his hand. The Transmitting Instrument was placed in another room and the doors of both rooms were closed.

I then shouted into M the following sentence: "Mr. Watson — Come here — I want to see you." To my delight he came and declared that he had heard and understood what I said.

I asked him to repeat the words — He said the answered "You said 'Mr. Watson — come here — I want to see you.'" We then changed places and I listened at S while Mr. Watson read a few passages from a book into the mouth piece M. It was certainly the case that articulate sounds proceeded from S. The effect was loud but indistinct and muffled.

If I had read beforehand the passage given by Mr. Watson I should have recognized every word. As it was I could not make out the sense — but an occasional word here and there was quite distinct. I made out "to" and "out" and "further", and finally the sentence "Mr. Bell Do you understand what I say? DO-you-un-der-stand-what-I-say" came quite clearly and intelligibly. No sound was audible when the armature S was removed.

*Notatki Alexandra Grahama Bell'a z udanego eksperymentu z jego pierwszym telefonem*

## Notatnik inżyniera dokumentuje Twoją pracę

Nie tylko używasz notatnika do organizowania i dokumentowania swojej pracy, ale jest to także miejscem do refleksji nad działaniami i projektami. Podczas pracy w zespole każdy członek zespołu będzie prowadził własny dziennik, aby ułatwić współpracę w grupie.

Twój notatnik inżyniera powinien mieć następujące elementy:

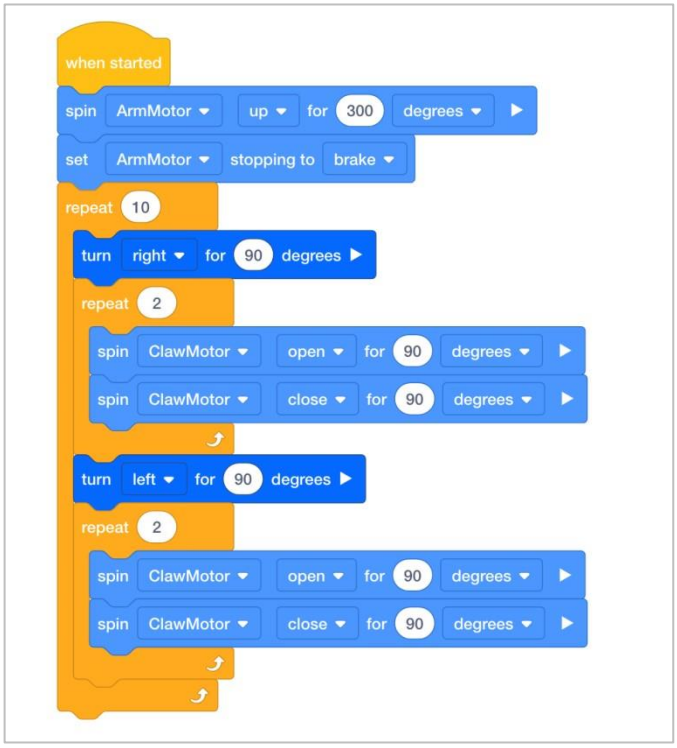
- Wpis na każdy dzień lub sesję, w której pracowałeś nad rozwiązaniem
- Wpisy chronologiczne, z datą każdego wpisu
- Jasne, schludne i zwięzłe notatki, dobrze zorganizowane
- Etykiety, aby czytelnik zrozumiał wszystkie Twoje notatki i ich dopasowanie do iteracyjnego procesu projektowania

Wpis może obejmować:

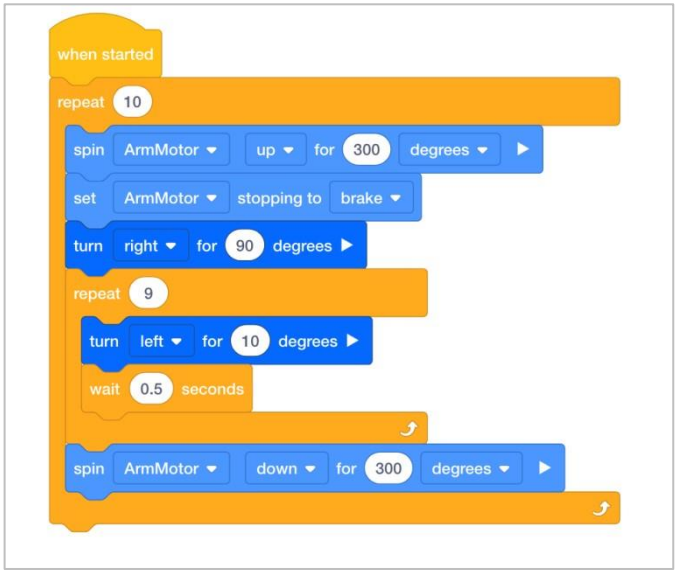
- Burzę mózgów
- Szkice lub zdjęcia prototypów
- Pseudokod i schematy blokowe planowania
- Wszelkie zastosowane obliczenia lub algorytmy
- Odpowiedzi na pytania przewodnie
- Uwagi dotyczące obserwacji i / lub przeprowadzonych testów
- Notatki i refleksje na temat różnych iteracji

# Przykładowe ruchy taneczne - bloki VEXcode V5

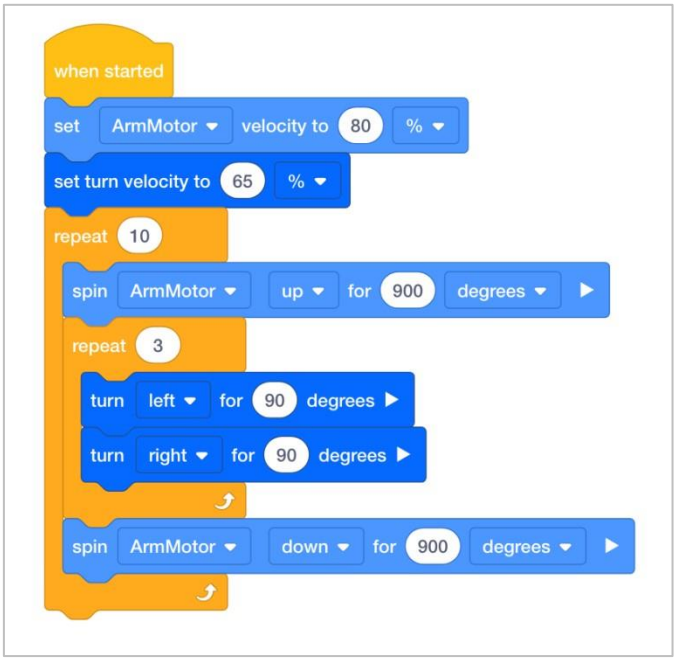
**Klaszcz!**




**Zraszacz**



## Ręce w górę!


when started
set ArmMotor velocity to 80 %
set turn velocity to 65 %
repeat 10
spin ArmMotor up for 900 degrees
repeat 3
turn left for 90 degrees
turn right for 90 degrees
spin ArmMotor down for 900 degrees

# Przykładowe ruchy taneczne - programowanie tekstowe

**Klaszcz!**

```
#include "vex.h"

using namespace vex;

int main() {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();
  ArmMotor.spinFor(forward, 300, degrees);
  ArmMotor.setStopping(brake);
  repeat(10) {
    Drivetrain.turnFor(right, 90, degrees);
    repeat(2) {
      ClawMotor.spinFor(forward, 90, degrees);
      ClawMotor.spinFor(reverse, 90, degrees);
      wait(5, msec);
    }
    Drivetrain.turnFor(left, 90, degrees);
    repeat(2) {
      ClawMotor.spinFor(forward, 90, degrees);
      ClawMotor.spinFor(reverse, 90, degrees);
      wait(5, msec);
    }
  }
}
```

## Zraszacz

```
#include "vex.h"

using namespace vex;

int main() {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();
  repeat(10) {
    ArmMotor.spinFor(forward, 300, degrees);
    ArmMotor.setStopping(brake);
    Drivetrain.turnFor(right, 90, degrees);
    repeat(9) {
      Drivetrain.turnFor(left, 10, degrees);
      wait(0.5, seconds);
    }
    ArmMotor.spinFor(reverse, 300, degrees);
    wait(5, msec);
  }
}

```

## Ręce w górę!

```
#include "vex.h"

using namespace vex;

int main() {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();
    ArmMotor.setVelocity(80, percent);
    Drivetrain.setTurnVelocity(50, percent);

    repeat(10) {
      ArmMotor.spinFor(forward, 500, degrees);
      repeat(3) {
        Drivetrain.turnFor(left, 90, degrees);
        Drivetrain.turnFor(right, 90, degrees);
        wait(5, msec);
      }
      ArmMotor.spinFor(reverse, 500, degrees);
      wait(5, msec);
      }
```