UNIVERSIDAD DE TARAPACÁ

ESCUELA UNIVERSITARIA DE INGENIERÍA INDUSTRIAL, INFORMÁTICA Y DE SISTEMAS

EUIIIS

Área de Ingeniería en Computación e Informática

Ingeniería Computación e Informática

Avance Proyecto FInal
DANTRON

Autor(es):
Angelo Coriza
David Orellana
Nicolas Vargas
Asignatura: Proyecto 1
Profesor(es): Diego Aracena
Ricardo Valdivia

ARICA, 7 de diciembre del 2018

Historial de Cambios

Fecha	Versión	Descripción	Autor(es)
16/08/2018	1.0	Creación y formulación de proyecto	Angelo Coriza David Orellana Nicolás Vargas
16/08/2018	1.1	Complementación de ideas existentes , se añadió nuevas tareas y se asignaron nuevos roles debido a la cantidad limitada de integrantes	Nicolás Vargas
06/09/2018	1.2	Estudio de posibles riesgos que puedan existir a lo largo del proyecto , soluciones de estos , completación de contenidos faltantes	Nicolás Vargas
15/11/2018	1.3	Arquitectura de software propuestas y requerimiento	Angelo Coriza David Orellana Nicolás Vargas
21/11/2018	1.4	Requerimiento y diseño de interfaz	Angelo Coriza David Orellana Nicolás Vargas
22/11/2018	1.5	implementación, estado actual del proyecto y anexo de código	Angelo Coriza David Orellana Nicolás Vargas
05/12/2018	1.6	Corrección de errores del informe anterior y añadir los nuevos puntos	David Orellana Angelo Coriza

Tabla de Contenidos

9. Referencias (estándar IEEE)

Anexos

Anexo A: Código de los programas implementados

Anexo B: Robot (diagrama de construcción, componentes principales)

1.1. Introducción

Lego Mindstorms es la tercera generación de bloques EV3 la cual posee de ciertos puertos de comunicación RJ12 los cuales nos permiten realizar comunicación entre EV3 y el dispositivo ya sea sensor o motor ,para así lograr una tarea determinada , con este modelo de lego se pretenderá ayudar a una persona a resolver el famoso cubo de Rubik de 3x3 el cual consta de 5 colores distintos
La finalidad de este informe es dar a conocer el proyecto “Dantron ” que consiste en el armado de un Robot Mindstorms de Lego el cual mediante unos algoritmos codificados en lenguaje de programación Python y escrito en el IDE (Entorno de Desarrollo Integrado) Visual Studio Code será capaz de ayudar a una persona a armar un cubo de Rubik’s (mediante un algoritmo escogido por los miembros del equipo) en un lapso de tiempo determinado adicionalmente en el presente informe se detallaran los miembros del equipo a los cuales se les ha asignado una/s tareas específicas para contribuir con el desarrollo del proyecto y este sea todo un éxito , así mismo se informará de los recursos y gastos necesarios para realizar el proyecto , analizar los posibles riesgos que puedan existir mediante el transcurso de su desarrollo y las posibles soluciones para poder resolverlos de manera adecuada y sin detener el avance de dicho proyecto .

1.2. Objetivo General

1.2.A. Armar un robot Lego Mindstorms que puede ejecutar algoritmos de movimiento para ayudar al armado de un cubo de Rubik de 3x3x3.

1.3. Objetivos Específicos

1.3.A. Crear robot a partir de instrucciones de armado del robot Lego Mindstorms EV3

1.3.B. Diseñar y codificar algoritmos de movimientos de cubo rubik 3x3x3 a un lenguaje de programación

1.3.C. Establecer una comunicación exitosa entre el robot y el dispositivo (PC, Teléfono celular) mediante Bluetooth para enviar algoritmos de movimiento los cuales debieran ayudar al armado del cubo 3x3x3 de Rubik's .

1.4. Restricciones:

Para el exitoso desarrollo del proyecto como restricciones tenemos los siguientes puntos a considerar.

- **Tiempo**: El tiempo es un factor clave para el proyecto ya que es el tiempo en el cual el proyecto se debe realizar sí o sí , ya que no disponemos de ningún mes , día o semana de más debido al fin del semestre académico.

- **Cantidad de Integrantes**: Otra restricción ya que algunos miembros del equipo deberán realizar más de alguna tarea debido que solo somos 3 integrantes en comparación a otros equipos de la asignatura.

- **Lego Mindstorms:** Kit del robot que se nos ha dado para trabajar (Lego Mindstorms) no tenemos otra opción para trabajar.

- **Programación:** Para codificar los algoritmos del cubo Rubik sólo podemos utilizar el lenguaje de programación Python por lo cual también nos vemos limitado.

- **Sistema Operativo:** Solo podemos elegir entre las 2 imágenes ISO entregadas por el profesor en clases EV3D.

1.5. Entregables

- **Bitácoras:** Al finalizar la clase del día jueves se deberá subir una Bitácora con los avances realizados en la semana, los problemas surgidos las posibles soluciones a estos y la planificación de las actividades a realizar en la semana siguiente.

- **Presentaciones:** Pueden ser realizadas en formato Power Point, estás presentaciones sirven para promocionar el proyecto realizado indicando detalles, costes, avances, resolución de problemas surgidos, etc.

- **Producto Final:** EL producto final será entregado con un manual de usuario en el cual se detalla cómo funciona el producto, sus características su funcionamiento y una pequeña demostración que garantice su funcionamiento.

- **Wiki:** Promoción del proyecto en él se detallan avance actual del proyecto, presupuesto , fotografías del proceso , etc.

2.1. Descripción de Roles

- **Líder del equipo:** Encargado de verificar el cumplimiento de los avances del proyecto en las fechas establecidas , velar por el trabajo en equipo , generar buenas relaciones entre los miembros del equipo .

- **Programador:** Persona que se dedicara previamente a estudiar el lenguaje de programación Python para posteriormente codificar algoritmos de armado de cubo Rubik a dicho lenguaje de programación.

- **Armador:** Persona encargada de arma el robot, conseguir piezas faltantes (en caso de de no estén dichas piezas) si están piezas no logran ser conseguidas se tratará de adaptar un mecanismo con las piezas que ya poseemos.

- **Secretario:** Persona encargada de subir las Bitácoras y/o avances del proyecto.

- **Wiki :** Persona encargada de subir fotografías con avances para promocionar el proyecto que se está realizando .

2.2. Personal que cumplirá los Roles

- **Líder** : Nicolás Vargas
- **Programador:** Angelo Coriza / Nicolás Vargas
- **Armador:** Nicolás Vargas /David Orellana / Angelo Coriza
- **Secretario :** David Orellana / Angelo Coriza / Nicolás Vargas
- **Wiki :** David Orellana

2.3. Mecanismos de Comunicación

El mecanismo de comunicación entre los miembros del equipo es en el aula de clases en las cuales se conversara de avances, problemas surgidos, resolución de estos este medio de comunicación es bastante limitado adicionalmente hemos creado Google Drive (para realizar más rápido los entregables del proyecto) poseemos grupo de WhatsApp y grupo en Facebook ante cualquier problema que pudiera surgir en determinado momento. Otro medio es REDMINE en el cual subimos los avances del proyecto y tenemos acceso a nuestra carta Gantt.

Planificación del Proyecto

3.1. Actividades: Dentro del proyecto de momento tenemos contempladas las siguientes actividades :

- **Armado del robot:** Construir el Robot a partir de un plano modificado
  Responsables:- Nicolas Vargas
  Seguidores: -Angelo Coriza
  Producto: Armado completo del Robot mediante las instrucciones de Lego

- **Aprender Python:** Aprender lenguaje Python para la codificación de algoritmos de movimiento del robot
  Responsable: -Nicolas Vargas
  Producto: Conocimiento del lenguaje de programación Python

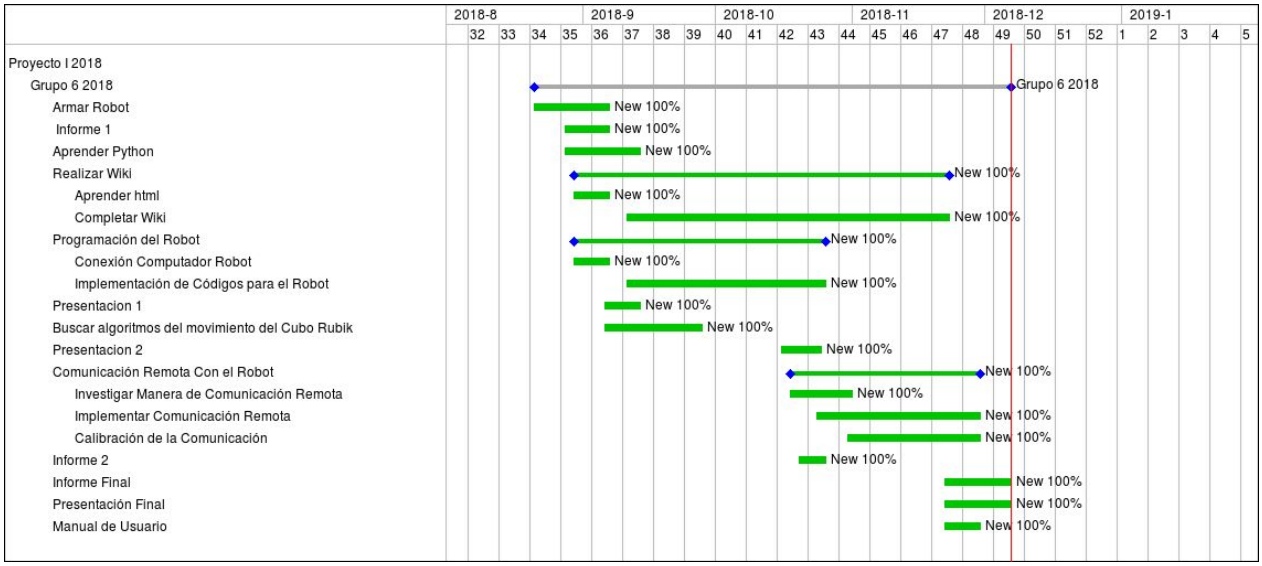
- **Realizar Wiki:** Crear y actualizar una wiki en la plataforma Redmine
  1. Aprender HTML: Aprender lenguaje HTML para la creación de la wiki
  2. Completar wiki: Actualizar la wiki
  Responsable:-David Orellana
  Seguidores:---
  Producto: Wiki en la cual se promociona y se muestran avances realizados en el proyecto.

- **Programación del robot :** Programar mediante el lenguaje Python poder mover el robot.
  1. Conexión computadora robot: Conexión vía SSH del Brick con el PC para poder enviar los archivos ejecutables los cuales permitirán que el Robot realice ciertos movimientos los cuales ayudaran a resolver un cubo de Rubik's.
  2. Probar código del robot: Se probarán los códigos o ajustarán para que realicen la tarea predeterminada de manera óptima.
  Responsable: - Nicolas Vargas
  P**roducto**: Código del robot
- **Informe 1:** Crear informe de planificación del proyecto
  Responsable:- Nicolas Vargas
  Seguidores: Angelo Coriza y David Orellana
  Producto: Planificación del proyecto en formato Word.
- **Presentación 1:** Crear presentación de la planificación del proyecto
  Responsable: Nicolas Vargas
  Seguidores: -Angelo Coriza, -David Orellana
  Producto: Planificación del proyecto en formato Power Point
- **Buscar algoritmos del movimiento del Cubo:** Hallar algoritmos de movimiento de cubo Rubik's de 3x3 y adaptarlo al Lenguaje de Programación Python.
  Responsable:-David Orellana
  Producto: Conocimiento de algoritmos y su codificación
- **Comunicación Remota Con el Robot:** Buscar e implementar comunicación remota con el robot
  1. Investigar Manera de Comunicación Remota: Investigar múltiples formas de

conectarse remotamente con el robot
2. Implementar Comunicación Remota: Implementar una forma de comunicación previamente investigada.
3. Calibración de la Comunicación: Resolver problemas en la comunicación entre el robot y el dispositivo.
Responsable:-Angelo Coriza
Seguidores:---
Producto: comunicación remota con el robot

- **Informe 2:** Crear informe del avance del proyecto
  Responsable:-David Orellana
  Seguidores: - Nicolas Vargas , Angelo Coriza
  Producto: Informe con los avances realizados y la corrección del informe entregado anteriormente en formato WORD
- **Presentación 2:** Crear presentación del avance del proyecto
  Responsable:-David Orellana
  Seguidores: - Nicolas Vargas, Angelo Coriza
  Producto: Presentación en formato Power Point

3.2. Asignación de tiempo:



3.3. Personal-rol asignado:

Actividad	Descripción	Responsable	Seguidores
Líder	Encargado de verificar el cumplimiento de los avances del proyecto en las fechas establecidas , velar por el trabajo en equipo , generar buenas relaciones entre los miembros del equipo .	Nicolas Vargas	---
Programadores	Persona que se dedicara previamente a estudiar el lenguaje de programación Python para posteriormente codificar algoritmos de armado de cubo Rubik a dicho lenguaje de programación.	Nicolas Vargas	Angelo Coriza
Armadores	Persona encargada de arma el robot, conseguir piezas faltantes (en caso de de no estén dichas piezas) si están piezas no logran ser conseguidas se tratará de adaptar un mecanismo con las piezas que ya poseemos.	Nicolas Vargas	David Orellana – Angelo Coriza
Secretario	Persona encargada de subir las Bitácoras y/o avances del proyecto.	David Orellana	Nicolas Vargas – Angelo Coriza
Encargado de la wiki	Persona encargada de subir fotografías con avances para promocionar el proyecto que se está realizando .	David Orellana	---

3.3. Gestión de Riesgos:

Riesgos	Probabilidad de Ocurrencia	Nivel de Impacto	Acción Remedial
Falte tiempo para el desarrollo del proyecto	70%	3	Se tratará de distribuir de manera mejor las actividades restantes o algún miembro del equipo deberá realizar más de alguna tarea con el fin de agilizar el avance del proyecto
Persona ajena al equipo manipule el Robot y lo dañe o use incorrectamente	60%	5	Dejamos escondido en la sala del ayudando el robot del equipo con la finalidad de protegerlo de la manipulación de terceras personas ajenas al proyecto.
Algún miembro tenga problemas o abandone el equipo	90%	6	Mantener una comunicación estable en el equipo.
Daño o pérdida de Tarjeta SD o Adaptador Wifi	40%	2	Las tarjetas SD suelen ser muy frágiles a múltiples factores en caso de daño se harán copias de seguridad constantemente y se reemplazará el dispositivo dañado
Piezas faltantes en el armado del robot	30%	4	Se deberá adaptar la pieza faltante los las piezas ya tenidas en el kit de Lego

4.1. Recursos Hardware-Software requeridos

- **Hardware:** Para el exitoso desarrollo del proyecto se necesita de Un Kit de lego MindStorms con todos sus accesorios, un PC para codificar los algoritmos en Python , una tarjeta SD, un adaptador WIFI y un cubo de Rubik's

- **Software:** Se requiere del sistema operativo (para cargador en el robot y enviar los códigos) el cual fue entregado en clases, el IDE Visual Studio Code , el lenguaje de Programación Python el cual puede ser descargado de su sitio WEB

4.2. Estimación de Costos (Hardware, Software, Recursos Humanos)

- **Hardware:** 1 notebook o PC para realizar la programación de algoritmos de armado de cubo Rubik's , 1 kit de Lego Mindstorm, cubo rubik 3x3, tarjeta sd, dongle wi-fi .

- **Software:** Los software utilizados en este proyecto no poseen costo ya que puede ser descargado de la web sin ningún costo .

- **Recursos Humanos:** El conocimiento de cada miembro es esencial para el desarrollo de este proyecto ya que cada uno aporta su conocimiento para potenciar el éxito que tendrá en proyecto.
    - Valor Hora:$9.500
    - Horas de trabajo(en la semana):12 horas
    - Semanas de trabajo: 18
    - Total de costos(Por persona):$2.052.000

Productos	**Costos(CLP)**
Robot EV3	$490.209
MicroSD	$6.000
Adaptador	$9.000
Cubo Rubik	$8.000
Software	$0
Valor equipo(3 personas)	$6.156.00
Total	**$6.669.209**

5. Análisis-diseño

5.1 Especificación de Requerimientos (incluyendo método/algoritmos considerados para resolver el cubo Rubik)

Requerimientos:

Funcional:	**No Funcional:**
El robot ejecuta movimientos básicos y algunos de mayor complejidad.Los cuales son: **Basicos:**R ,R',L,L',U,U',D,D',F,F' **Complejos:**Sexy move, Cruz, Sexy move en la cara izquierda, Cambio Aristas	El robot ejecutará los movimientos básicos en un tiempo de 20 segundos y los complejos en 5 minutos
La conexión entre los dispositivos (Robot - PC) solo debe ser por WIFI ya que por otro método no es posible realizar dicha funcional	Se requiere una conexión WIFI estable para un uso más cómodo y no perder el vínculo entre los dispositivos.
Interfaz cómoda y entendible para el usuario	Se recomienda el uso de una computadora con sistema operativo Windows.
Los algoritmos disponibles son suficientes para que el usuario arme el cubo rubik	El robot solo armara un cubo rubik 3x3x3.
Los imágenes de movimientos son los botones que moverán al robot	El sistema operativo del robot debe ser EV3dev Stretch

5.2 Arquitectura Propuesta (incluyendo aspectos de comunicación)

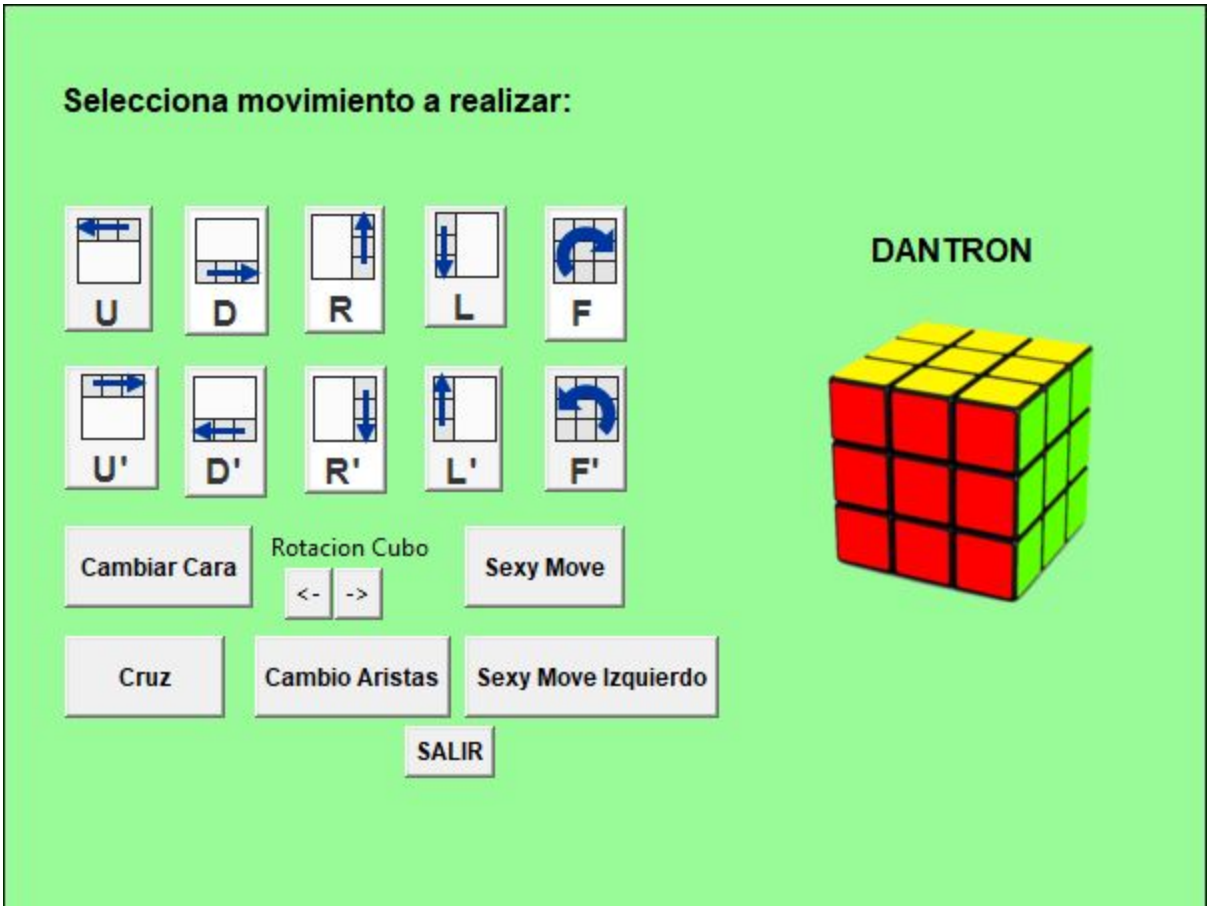
- **Computador:** El computador será el cliente ,ya que contiene la interfaz que solicita la ejecución de las funciones que solicite el usuario.

- **Usuario:** El usuario podrá hacer uso de la interfaz en la cual le aparecerán los movimientos que pueda realizar , dichos movimientos le ayudarán en el armado del cubo Rubik's.

- **Robot Lego Mindstorms:** Será el encargado de realizar los movimientos que elija el usuario en la interfaz de usuario estará conectado mediante la aplicación ambos están vinculados mediante conexión WIFI.El robot será el servidor ,ya que contiene los algoritmos de movimiento en lenguaje Python y luego serán pedidos por el cliente

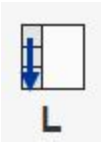
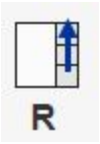
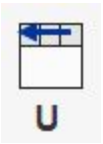
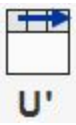
- **WIFI :** Sería ideal que la conexión sea estable para no perder dirección IP y no entorpecer la comunicación entre los dispositivos (Computador - Robot) .

5.3 Diseño de la Interfaz Usuario

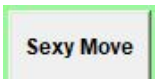
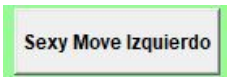
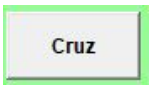


Consiste en una ventana en la cual existen 18 botones , 10 movimientos básicos de cubo rubik 3x3x3, 4 movimientos complejos, 3 movimientos que ayudan al usuario para analizar el cubo y uno para salir de la aplicación .

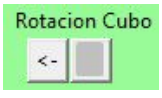
Movimientos básicos:

 Movimiento down	 Movimiento down primo
 Movimiento front	 Movimiento front primo
 Movimiento left	 Movimiento left primo
 Movimiento right	 Movimiento right primo
 Movimiento up	 Movimiento up primo

Movimientos complejos:

 Ejecuta Sexy move	 Ejecuta Sexy move en la cara izquierda
 Ejecuta Cruz	 Ejecuta Cambio Aristas

movimientos para analizar el cubo y salir:

 Cambia la cara del cubo	 Rota el cubo completo en sentido horario
 Rota el cubo completo en sentido antihorario	 Botón para cerrar la aplicación

6. Implementación

6.1 Descripción de los programas implementados (entradas, salidas, procesos)

Los movimientos básicos creados para la implementación de de los movimientos del cubo son los siguientes :

1)**CambiarCara()**: Este consiste en el movimiento del brazo para trasladar la cara frontal hacia arriba.

2)**def BaseAH():** Mueve la base donde está el cubo 90° en sentido anti horario.

3)**def BaseNH():** Mueve la base donde está el cubo 90° en sentido horario.

4)**def Descanso():** Retrae el brazo del robot para un movimiento de la libre de la base .

5)**def Accion():** El brazo vuelve a estar sobre el cubo.

Para realizar los movimientos del cubo:

6)**def Dn():**Realiza movimiento Down normal.

7)**def Dp():** Realiza movimiento Down prima.

8)def Un(): Realiza movimiento Up normal.

9)def Up(): Realiza movimiento Up prima.

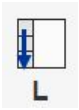
10)def Fn(): Realiza movimiento Frontal normal.

11)**def Fp():** Realiza movimiento Frontal prima.

12)def Rn(): Realiza movimiento Right normal.

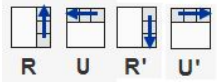
13)def Rp(): Realiza movimiento Right prima.

14)def Ln(): Realiza movimiento Left normal.

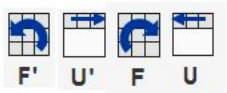


15)def Lp(): Realiza movimiento Left prima.

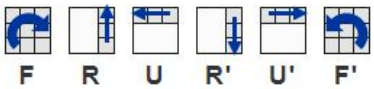
16)def sexym(): relisa el algoritmo de sexy move



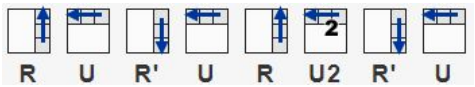
17)**def sexymizq():** relisa el algoritmo de sexy move en la cara izquierda



18)def cruz(): realiza el algoritmo de cruz



19)def cambioari(): realiza el algoritmo cambio de arista



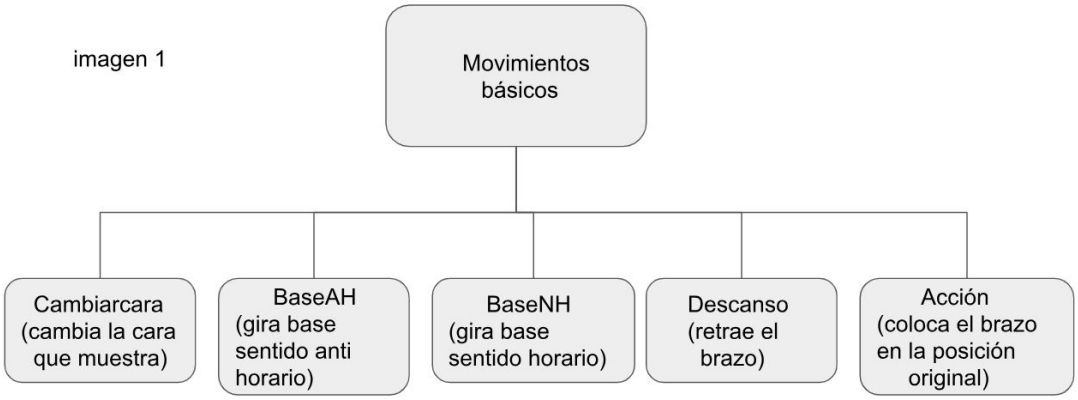
20)def Rotarizq(): Retrae el brazo, rota hacia la izquierda la base del robot y luego lo coloca en posición para ejecutar movimientos

21)def Rotarder(): Retrae el brazo, rota hacia la derecha la base del robot y luego lo coloca en posición para ejecutar movimientos

6.2 Diagrama de interacción entre programas

Movimientos base :

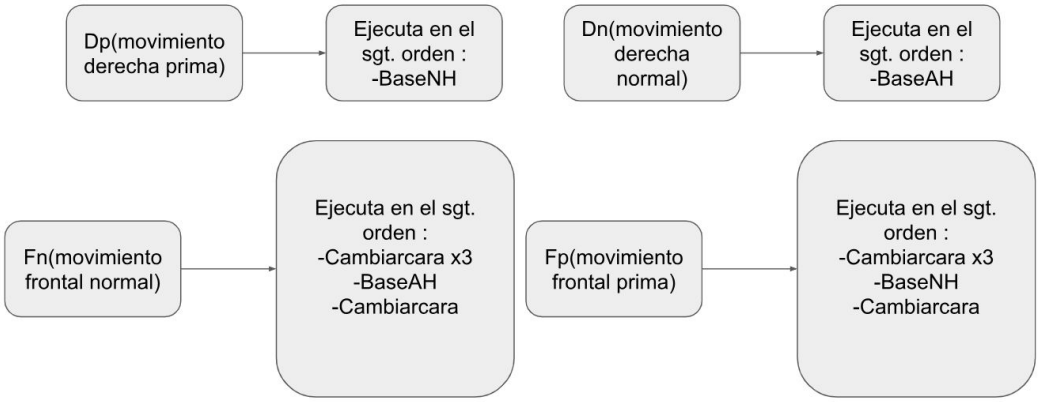
imagen 1



En la imagen 1 se muestra los movimientos básicos que hace el robot para los movimientos de cubo 3x3x3

Movimiento de cubo 3x3x3 :

imagen 2



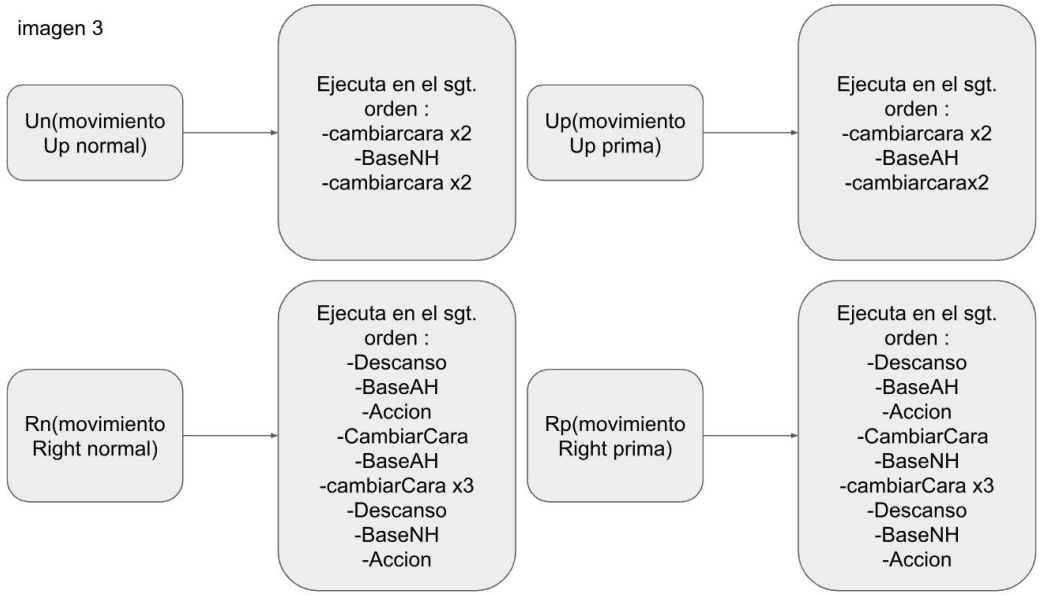
En la imagen 2 se muestra los movimientos :

Dp: Rota en sentido horario la base del cubo, este está formado por BaseNH

Dn: Rota en sentido antihorario la base del cubo, este está formado por BaseAH

Fp: Rota en sentido horario la cara frontal del cubo, este está formado por CambiarCara 3 veces - BaseNH - CambiarCara

Fn: rota en sentido antihorario la cara frontal del cubo, este está formado por CambiarCara 3 veces - BaseAH - CambiarCara



En la imagen 3 se muestra los movimientos :

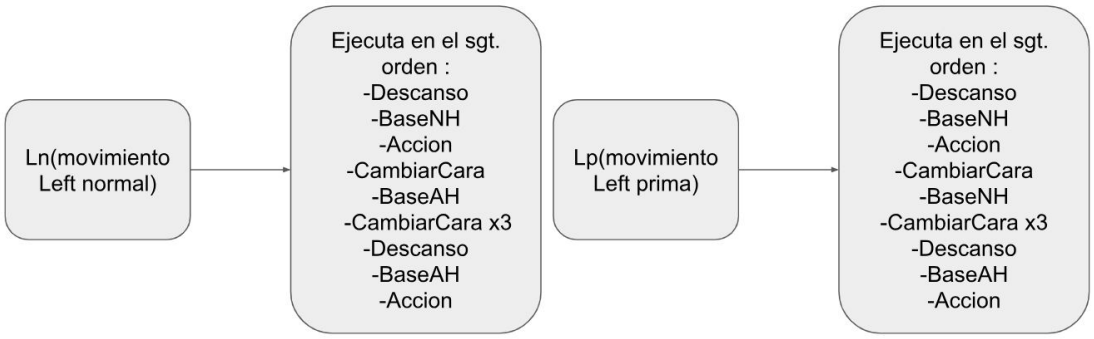
Up: Rota en sentido horario la parte superior del cubo, este está formado por CambiarCara 2 veces - BaseAH - CambiarCara 2 veces

Un: Rota en sentido antihorario la parte superior del cubo, este está formado por CambiarCara 2 veces - BaseNH - CambiarCara 2 veces

Rp: Rota en sentido horario el costado derecho del cubo, este está formado por Descanso - BaseAH - Accion - CambiarCara - BaseNH - CambiarCara 3 veces - Descanso - BaseNH - Accion

Rn: Rota en sentido antihorario el costado derecho del cubo, este está formado por Descanso - BaseAH - Accion - CambiarCara - BaseAH - CambiarCara 3 veces - Descanso - BaseNH - Accion

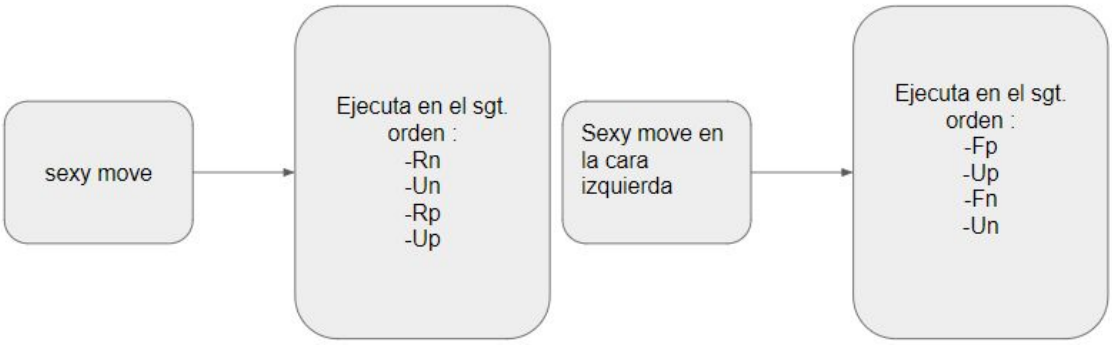
imagen 4



En la imagen 4 se muestra los movimientos :
Lp: Rota en sentido horario el costado izquierdo del cubo, este está formado por Descanso - BaseNH - Accion - CambiarCara - BaseNH - CambiarCara 3 veces - Descanso - BaseAH - Accion
Ln: Rota en sentido antihorario el costado izquierdo del cubo, este está formado por Descanso - BaseNH - Accion - CambiarCara - BaseAH - CambiarCara 3 veces - Descanso - BaseAH - Accion

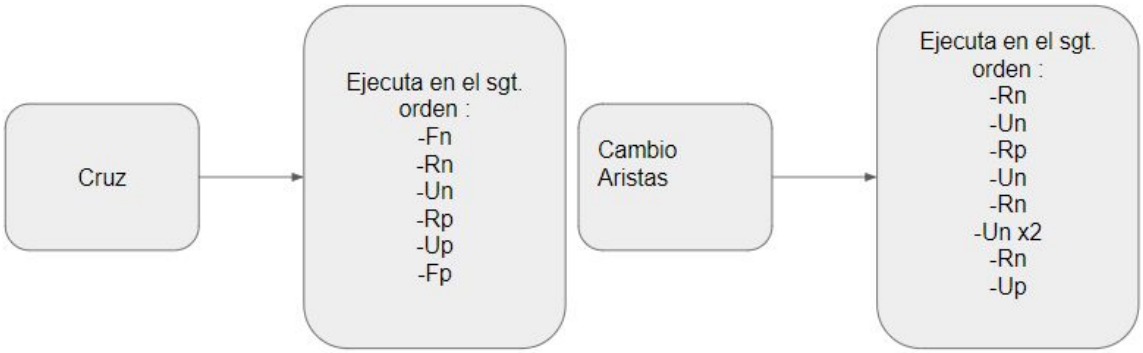
Movimientos complejos:

imagen 5



En la imagen 5 se muestra los movimientos :
Sexy move: Sigue la secuencia de movimientos de cubo
Rn - Un - Rp - Up
Sexy move en la cara izquierda: Sigue la secuencia de movimientos de cubo
Fp - Up - Fn - Un

imagen 6



En la imagen 6 se muestra los movimientos :

Cruz: Sigue la secuencia de movimientos de cubo

Fn - Rn - Un - Rp - Up - Fp

Cambio aristas: Sigue la secuencia de movimientos de cubo

Rn - Un - Rp - Un - Rn - Un - Un - Rn - Up

7. Pruebas

7.1 Descripción de las pruebas realizadas

Conexión: Verificar que el brick (servidor) y la computadora(cliente) está conectado mediante una conexión wi-fi estable

Interfaz: Teniendo la interfaz lograda debíamos lograr conectarla con el servidor

Movimientos: Ejecutar movimientos mediante la aplicación

Movimientos complejos: Esta prueba consiste en ejecutar los movimientos complejos

7.2 Resultados de las pruebas

Conexión: La conexión mediante wi-fi es estable ya que ambos deben estar conectadas a la misma red, esto debido a que se debe ingresar la ip de el robot en el código

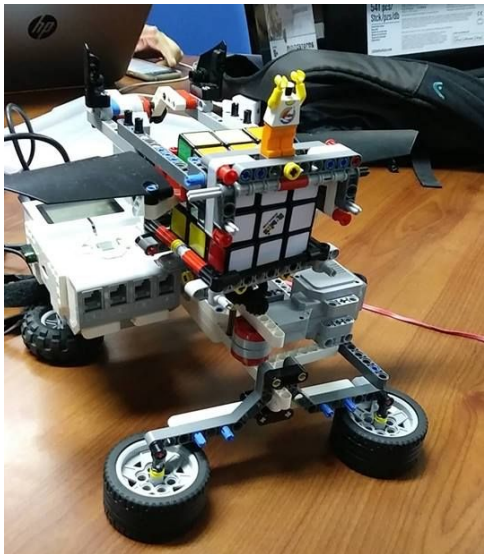
Interfaz: Logramos hacer que la interfaz interactuara con el robot haciéndolo hablar inicialmente para luego hacer los movimientos del cubo

Movimientos: El movimiento se ejecuta bien pero persiste el descalibre

Movimientos complejos: El movimiento constante de el brazo y la base hace que el cubo se descoloque de la base y debido a esto fuerce los motores

8. Resultados

8.1 Estado final



8.2 Conclusiones

A lo largo de este proyecto se nos presentaron varias dificultades tanto prácticas ,falta de piezas , el tiempo dedicado al los avances de los entregables y la programación del robot ,sin embargo estas dificultades fueron superadas para la fecha límite para lograr un cumplimiento con la entrega.

Además hicimos aplicación de ramos hechos con anterioridad “como Fundamento de lenguajes de programación”(FLP) el cual nos ayudó bastante para adaptarnos al lenguaje que fue propuesto por los Profesores para el proyecto, incluso adquirimos experiencia al programar y en la búsqueda de información para lograr interfaz de usuario usando Python.

Otro punto a destacar es lograr una comunicación a través de IP usando Wifi lo cual nunca aplicamos con anterioridad ,pero con ayuda de los Profesores y el ayudante tanto en la parte práctica como con la información que se nos entregó la cual fue de gran ayuda para lograr la conección entre el robot y computador

8.3 Trabajo a Futuro

Como equipo logramos finalizar el proyecto logrando que el robot ayude al armado del cubo rubik, pero nos gustaría mejorarlo y hacer que el robot lo arme de manera independiente usando un sensor y logre armarlo solo

9. Referencias (estándar IEEE)

- **Sitio web de Lego Mindstorms :** Explora una galería de robots épicos de LEGO y LEGO MINDSTORMS creados por fanáticos, y haz clic para obtener todos los detalles, instrucciones de construcción y misiones de programación.

  https://www.lego.com/en-us/mindstorms/build-a-robot

- **Python :** Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos

  https://www.python.org/

- **Visual Code Studio :** Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

  https://code.visualstudio.com/

- **Sitio de EV3DEV :** Ev3dev es un sistema operativo basado en Debian Linux que se ejecuta en varias plataformas compatibles con LEGO® MINDSTORMS, incluyendo LEGO® MINDSTORMS EV3 y BrickPi impulsado por Raspberry Pi.

  https://www.ev3dev.org/

Anexo

Códigos

Código de moviento

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
from time import sleep
def CambiarCara():
    mA = LargeMotor('outA')
    i=1
    if i == 1:
       mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
       time.sleep(1)
    i=i+1
    if i == 2:
       mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
       time.sleep(1)

def BaseAH():
    mB = LargeMotor('outB')
    mB.run_to_rel_pos(position_sp=-285, speed_sp=900, stop_action="hold")
    time.sleep(1)

def BaseNH():
    mB = LargeMotor('outB')
```

```
    mB.run_to_rel_pos(position_sp=285, speed_sp=900, stop_action="hold")
    time.sleep(1)


def Descanso():
    m = LargeMotor('outA')
    m.run_to_rel_pos(position_sp=-100, speed_sp=500, stop_action="hold")
    sleep(1)


def Accion():
    m = LargeMotor('outA')
    m.run_to_rel_pos(position_sp=100, speed_sp=900, stop_action="hold")
    sleep(1)


def Dp():
    BaseNH()


def Dn():
    BaseAH()


def Fn():
    i=0
    while i!=3:
      CambiarCara()
      i=i+1
```

```
    BaseAH()
    CambiarCara()


def Fp():
    i=0
    while i!=3:
      CambiarCara()
      i=i+1
    BaseNH()
    CambiarCara()


def Un():
    i=0
    while i!=2:
      CambiarCara()
      i=i+1
    BaseNH()
    i=0
    while i!=2:
      CambiarCara()
      i=i+1


def Up():
    i=0
```

```
        while i!=2:
            CambiarCara()
            i=i+1
        BaseAH()
        i=0
        while i!=2:
            CambiarCara()
            i=i+1

    def Rn():
        Descanso()
        BaseAH()
        Accion()
        CambiarCara()
        BaseAH()
        i=0
        while i!=3:
            CambiarCara()
            i=i+1
        Descanso()
        BaseNH()
        Accion()

    def Rp():
```

```
        Descanso()
        BaseAH()
        Accion()
        CambiarCara()
        BaseNH()
        i=0
        while i!=3:
            CambiarCara()
            i=i+1
        Descanso()
        BaseNH()
        Accion()

    def Ln():
        Descanso()
        BaseNH()
        Accion()
        CambiarCara()
        BaseAH()
        i=0
        while i!=3:
            CambiarCara()
            i=i+1
        Descanso()
```

```
    BaseAH()
    Accion()


def Lp():
    Descanso()
    BaseNH()
    Accion()
    CambiarCara()
    BaseNH()
    i=0
    while i!=3:
      CambiarCara()
      i=i+1
    Descanso()
    BaseAH()
    Accion()


def sexym():
    Rn()
    Un()
    Rp()
    Up()
def sexymizq():
    Fp()
```

```
    Up()
    Fn()
    Un()

def cruz():
    Fn()
    Rn()
    Un()
    Rp()
    Up()
    Fp()

def cambioari():
    Rn()
    Un()
    Rp()
    Un()
    Rn()
    Un()
    Un()
    Rp()
     Un()

def Rotarizq():
```

```
    Descanso()
    BaseNH()
    Accion()

def Rotarder():
    Descanso()
    BaseAH()
    Accion()

sleep(1)
```

Código Servidor:

```
from ev3dev.ev3 import *
from time import sleep
import ev3dev.ev3 as ev3
import rpyc
from PIL import Image


class MyService(rpyc.Service):

  def exposed_line_cambiarcara(self):
    mA = LargeMotor('outA')
    j=1
```

```
    if j == 1:
        mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
        time.sleep(1)
    j=j+1
    if i == 2:
        mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
        time.sleep(1)

def exposed_line_baseah(self):
    mB = LargeMotor('outB')
    mB.run_to_rel_pos(position_sp=-285, speed_sp=900, stop_action="hold")
    time.sleep(1)


def exposed_line_basenh(self):
    mB = LargeMotor('outB')
    mB.run_to_rel_pos(position_sp=285, speed_sp=900, stop_action="hold")
    time.sleep(1)


def exposed_line_descanso(self):
    m = LargeMotor('outA')
    m.run_to_rel_pos(position_sp=-100, speed_sp=500, stop_action="hold")
    sleep(1)
```

```
def exposed_line_accion(self):
    m = LargeMotor('outA')
    m.run_to_rel_pos(position_sp=100, speed_sp=900, stop_action="hold")
    sleep(1)

def exposed_line_dp(self):
    mB = LargeMotor('outB')
    mB.run_to_rel_pos(position_sp=285, speed_sp=900, stop_action="hold")
    time.sleep(1)

def exposed_line_dn(self):
    mB = LargeMotor('outB')
    mB.run_to_rel_pos(position_sp=-285, speed_sp=900, stop_action="hold")
    time.sleep(1)

def exposed_line_fn(self):
    mA = LargeMotor('outA')
    mB = LargeMotor('outB')
    j=0
    while j!=3:
        i=1
        if i == 1:
            mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
```

```
            time.sleep(1)
        i=i+1
        if i == 2:
            mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
            time.sleep(1)
        j=j+1
    mB.run_to_rel_pos(position_sp=-285, speed_sp=900, stop_action="hold")
    i=1
    if i == 1:
        mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
        time.sleep(1)
    i=i+1
    if i == 2:
        mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
        time.sleep(1)


def exposed_line_fp(self):
    mA = LargeMotor('outA')
    mB = LargeMotor('outB')
    j=0
    while j!=3:
        #Inicio
        i=1
        if i == 1:
```

```
            mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
            time.sleep(1)
        i=i+1
        if i == 2:
            mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
            time.sleep(1)
        #FIn
        j=j+1
    #Inicio
    mB.run_to_rel_pos(position_sp=285, speed_sp=900, stop_action="hold")
    time.sleep(1)
    #Fin
    #inicio
    i=1
    if i == 1:
        mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
        time.sleep(1)
    i=i+1
    if i == 2:
        mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
        time.sleep(1)
    #FIn
def exposed_line_un(self):
    mA = LargeMotor('outA')
```

```
mB = LargeMotor('outB')
i=0
while i!=2:
  #iNICIO cambiar cara
  j=1
  if j == 1:
    mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
    time.sleep(1)
  j=j+1
  if i == 2:
    mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
    time.sleep(1)
  #FIn
  i=i+1
#Inicio
mB.run_to_rel_pos(position_sp=285, speed_sp=900, stop_action="hold")
time.sleep(1)
#FIn
i=0
while i!=2:
  #inicio
  j=1
  if j == 1:
    mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
```

```
            time.sleep(1)
        j=j+1
        if i == 2:
            mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
            time.sleep(1)
        #Fin
        i=i+1


def exposed_line_up(self):
    mA = LargeMotor('outA')
    mB = LargeMotor('outB')
    i=0
    while i!=2:
        #iNICIO cambiar cara
        j=1
        if j == 1:
            mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
            time.sleep(1)
        j=j+1
        if i == 2:
            mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
            time.sleep(1)
        #FIn
        i=i+1
```

```
    #Inicio
    mB.run_to_rel_pos(position_sp=-285, speed_sp=900, stop_action="hold")
    time.sleep(1)
    #FIn
    i=0
    while i!=2:
      #inicio
      j=1
      if j == 1:
        mA.run_to_rel_pos(position_sp=70, speed_sp=750, stop_action="hold")
        time.sleep(1)
      j=j+1
      if i == 2:
        mA.run_to_rel_pos(position_sp=-70, speed_sp=750, stop_action="hold")
        time.sleep(1)
      #Fin
      i=i+1


def exposed_line_rn(self):
    Descanso()
    BaseAH()
    Accion()
    CambiarCara()
    BaseAH()
```

```
    i=0
    while i!=3:
     CambiarCara()
     i=i+1
     Descanso()
     BaseNH()
     Accion()



def exposed_line_rp(self):
    Descanso()
    BaseAH()
    Accion()
    CambiarCara()
    BaseNH()
    i=0
    while i!=3:
       CambiarCara()
       i=i+1
    Descanso()
    BaseNH()
    Accion()
```

```
def exposed_line_ln(self):
    Descanso()
    BaseNH()
    Accion()
    CambiarCara()
    BaseAH()
    i=0
    while i!=3:
        CambiarCara()
        i=i+1
    Descanso()
    BaseAH()
    Accion()

def exposed_line_lp(self):
    Descanso()
    BaseNH()
    Accion()
    CambiarCara()
    BaseNH()
    i=0
    while i!=3:
        CambiarCara()
        i=i+1
```

```
    Descanso()
    BaseAH()
    Accion()
   #Lo nuevo
def exposed_line_sexym(self):
    Rn()
    Un()
    Rp()
    Up()
def exposed_line_sexymizq(self):
    Fp()
    Up()
    Fn()
    Un()

def exposed_line_cruz(self):
    Fn()
    Rn()
    Un()
    Rp()
    Up()
    Fp()
def exposed_line_cambioari(self):
    Rn()
```

```
    Un()
    Rp()
    Un()
    Rn()
    Un()
    Un()
    Rp()
    Un()

def exposed_line_rotaizquierda(self):
    m = LargeMotor('outA')
    m.run_to_rel_pos(position_sp=-100, speed_sp=500, stop_action="hold")
    sleep(1)
    mB = LargeMotor('outB')
    mB.run_to_rel_pos(position_sp=285, speed_sp=900, stop_action="hold")
    time.sleep(1)
    m = LargeMotor('outA')
    m.run_to_rel_pos(position_sp=-100, speed_sp=500, stop_action="hold")
    sleep(1)

def exposed_line_rotaderecha(self):
    m = LargeMotor('outA')
    m.run_to_rel_pos(position_sp=-100, speed_sp=500, stop_action="hold")
    sleep(1)
```

```
    mB = LargeMotor('outB')
    mB.run_to_rel_pos(position_sp=-285, speed_sp=900, stop_action="hold")
    time.sleep(1)
    m = LargeMotor('outA')
    m.run_to_rel_pos(position_sp=-100, speed_sp=500, stop_action="hold")
    sleep(1)
    sleep(1)


from rpyc.utils.server import ThreadedServer
t = ThreadedServer(MyService, port = 16800)
t.start()
```

Código interfaz:

```
#!/usr/bin/env python
# Foundations of Python Network Programming - Chapter 18 - rpyc_client.py
# RPyC client
import rpyc
from tkinter import *
from tkinter import font



proxy = rpyc.connect('192.168.0.20', 16800, config={'allow_public_attrs': True})
```

```
def exit():
        ventana.quit()
def codern():
    pass

def coderp():
        pass
def codeln():
        pass
def codelp():
        pass
def codefn():
    proxy.root.linr_fn()

def codefp():
    proxy.root.line_fp()

def codeun():
    proxy.root.un()

def codecara():
```

```
        proxy.root.line_cambiarcara()


def codeup():
            proxy.root.line_up()
def codedn():
        proxy.root.line_dn()


def codedp():
        proxy.root.line_dp()


def cambiarc():
            pass
def sexy():
            pass
def sexyizq():
            pass
def cruz():
            pass
def cambioari():
            pass
def cambioizquierda():
        proxy.root.line_rotaizquierda()


def cambioderecha():
```

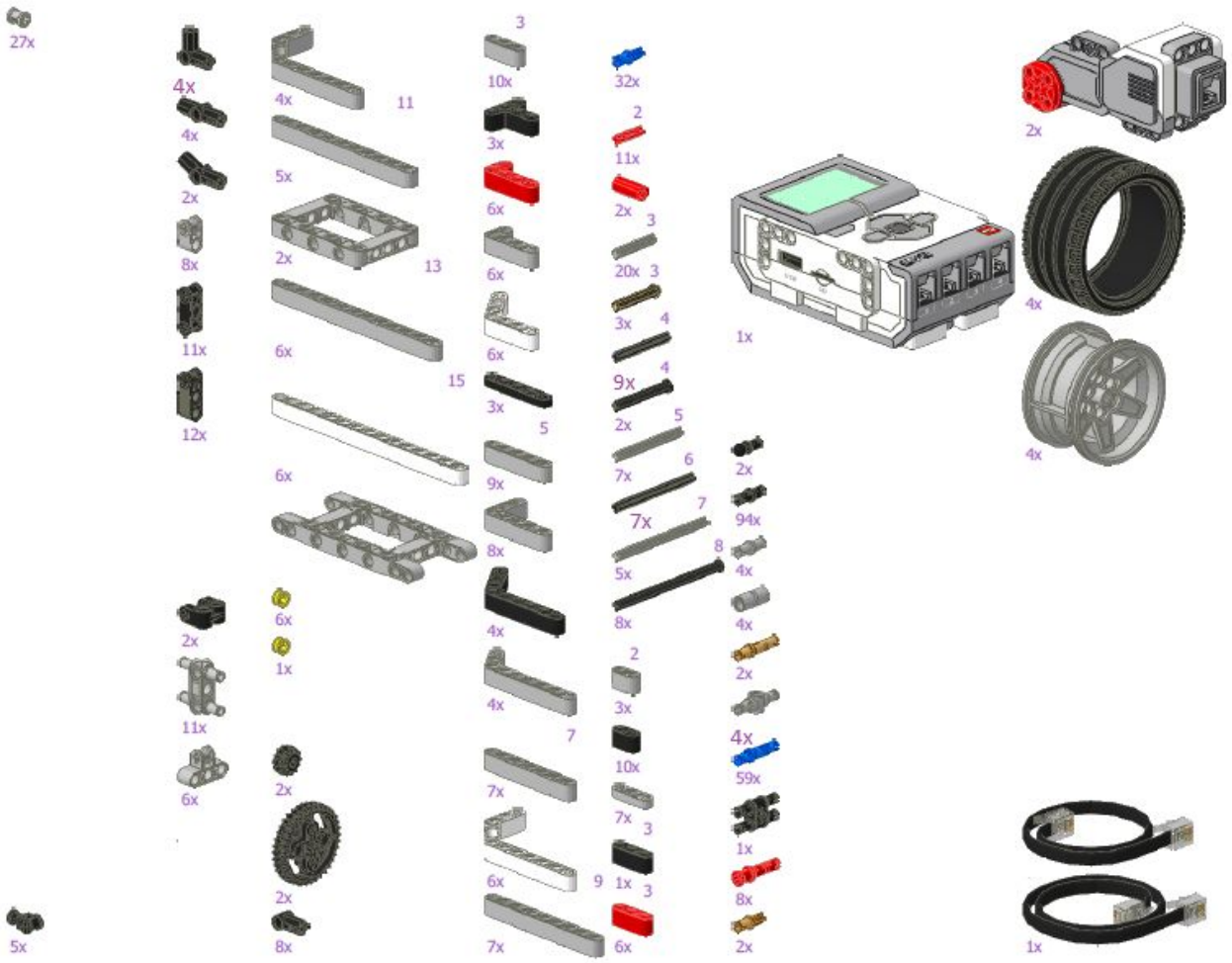
```
            proxy.root.line_rotaderecha()



        #Fin Funciones Botones

        #Inicio Ventana

        ventana = Tk()

        ventana.title("Dantron")

        ventana.geometry("600x450")

        ventana.resizable(width=False, height=False)

        ventana.iconbitmap("teria.ico")

        ventana.config(bg="PaleGreen")


        #Fin ventana

        #Fuente

        Negrita1 = font.Font(family="Negrita", size=12, weight="bold")

        Negrita2 = font.Font(family="Negrita", size=9, weight="bold")


        #Fin Fuente

        etiqueta= Label(ventana, text="Selecciona movimiento a
realizar:",bg="PaleGreen",font=Negrita1).place(x=25 , y=35)



        #Imagenes

        cubo=PhotoImage(file="cub.png")
```

```
un=PhotoImage(file="un.png")

up=PhotoImage(file="up.png")

dn=PhotoImage(file="dn.png")

dp=PhotoImage(file="dp.png")

rn=PhotoImage(file="rn.png")

rp=PhotoImage(file="rp.png")

ln=PhotoImage(file="ln.png")

lp=PhotoImage(file="lp.png")

fn=PhotoImage(file="fn.png")

fp=PhotoImage(file="fp.png")


#Fin Imagenes

label1=Label(ventana, image=cubo,bg="PaleGreen").place(x=400,y=150)

#Inicio Botones


boton1 = Button(ventana, image=un,command=codeun).place( x =30  , y=100)

boton2 = Button(ventana, image=up, command=codeup).place( x =30  , y=180)


boton3 = Button(ventana, image=dn, command=codedn).place( x =90  , y=100)

boton4 = Button(ventana, image=dp, command=codedp).place( x =90  , y=180)


boton5 = Button(ventana, image=rn, command=codern).place( x =150  , y=100)

boton6 = Button(ventana, image=rp, command=coderp).place( x =150  , y=180)
```

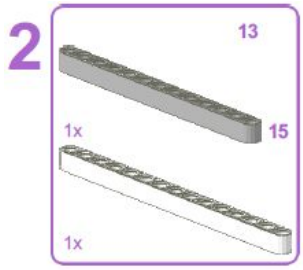
```
        boton7 = Button(ventana, image=ln, command=codeln).place( x =210  , y=100)

        boton8 = Button(ventana, image=lp, command=codelp).place( x =210  , y=180)


        boton9 = Button(ventana, image=fn, command=codefn).place( x =270  , y=100)

        boton10 = Button(ventana, image=fp, command=codefp).place( x =270  , y=180)

        boton11 = Button(ventana, text="Cambiar Cara",
command=codecara,font=Negrita2,height=2,width=12).place( x =30  , y=260)

        #Rotar

        label2=Label(ventana, text="Rotacion Cubo",bg="PaleGreen",fg="Black").place(x=130,y=260)

        boton12 = Button(ventana, text="->", command=cambioderecha).place( x =165  , y=281)

        boton13 = Button(ventana, text="<-", command=cambioizquierda).place( x =140  , y=281)

        #Fin Rotar

        boton14 = Button(ventana, text="Sexy Move",
command=sexy,font=Negrita2,height=2,width=10).place( x =230  , y=260)

        boton15 = Button(ventana, text="Sexy Move Izquierdo",
command=sexyizq,font=Negrita2,height=2).place( x =230  , y=315)

        boton16 = Button(ventana, text="Cruz",
command=cruz,font=Negrita2,height=2,width=10).place( x =30  , y=315)

        boton17 = Button(ventana, text="Cambio Aristas",
command=cambioari,font=Negrita2,height=2).place( x =125  , y=315)

        #Fin botones

        boton20 = Button(ventana, text = "SALIR",font=Negrita2, command=exit).place( x =200  ,
y=360)


        ventana.mainloop()
```

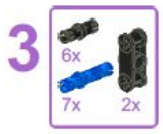
Pasos de armado del robot:

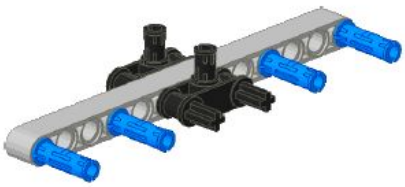
Piezas



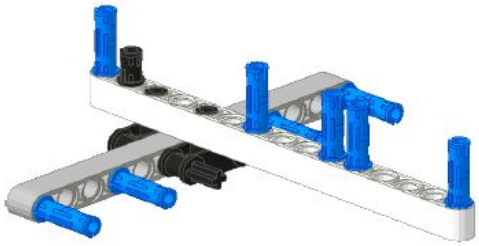
esqueleto

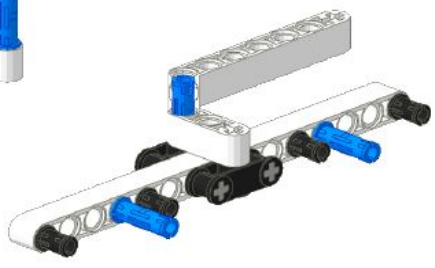


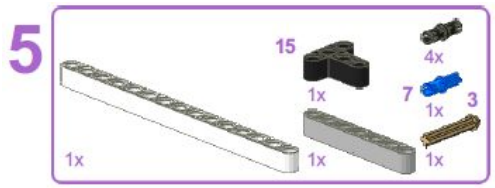

3
6x
7x
2x

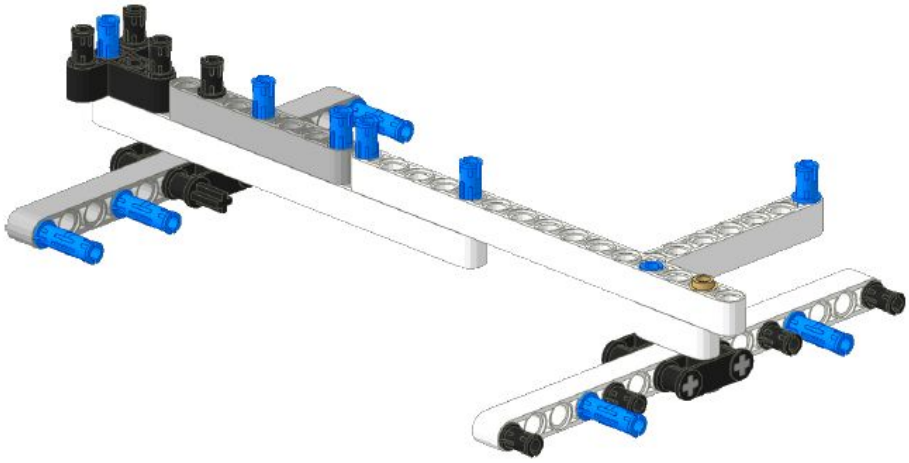


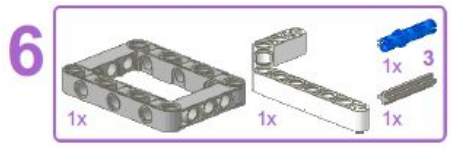

4
1x
15
5x
1x
1x

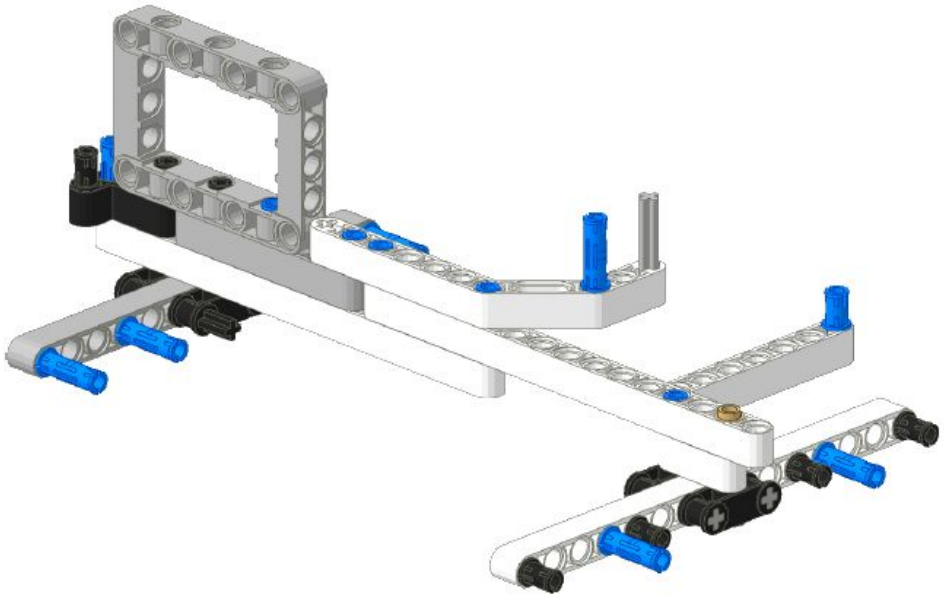


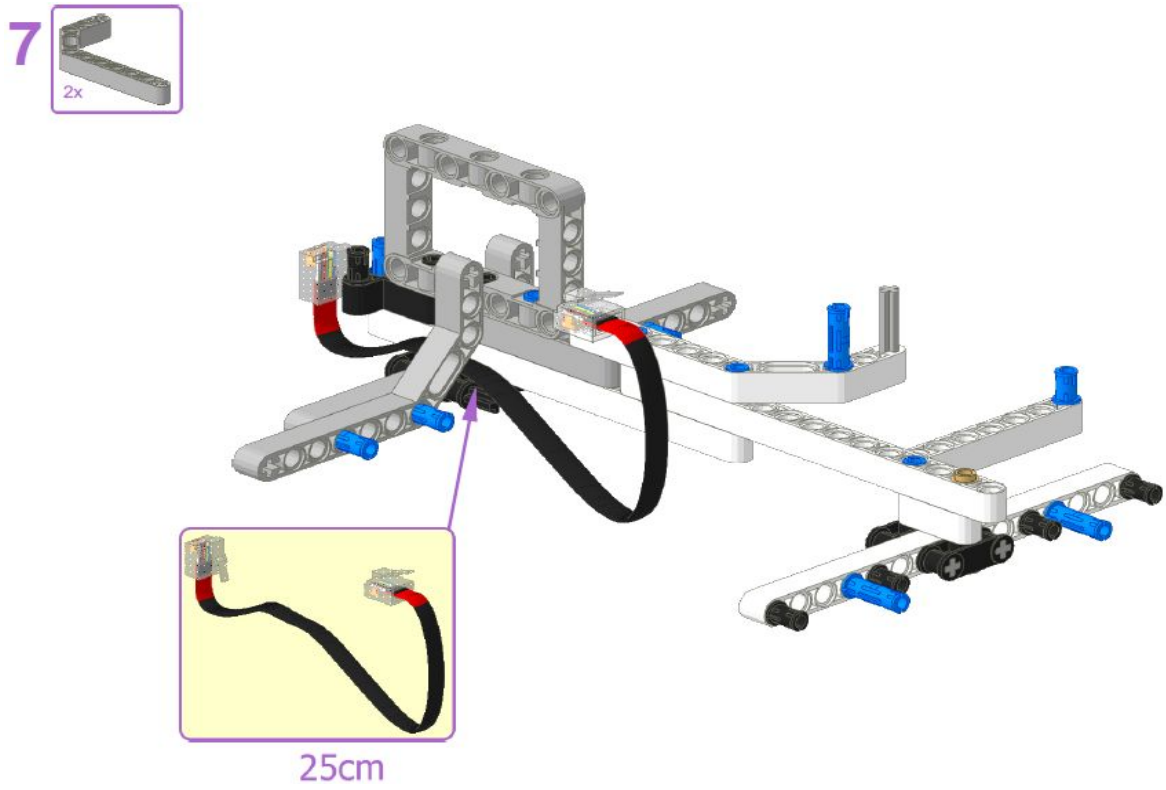


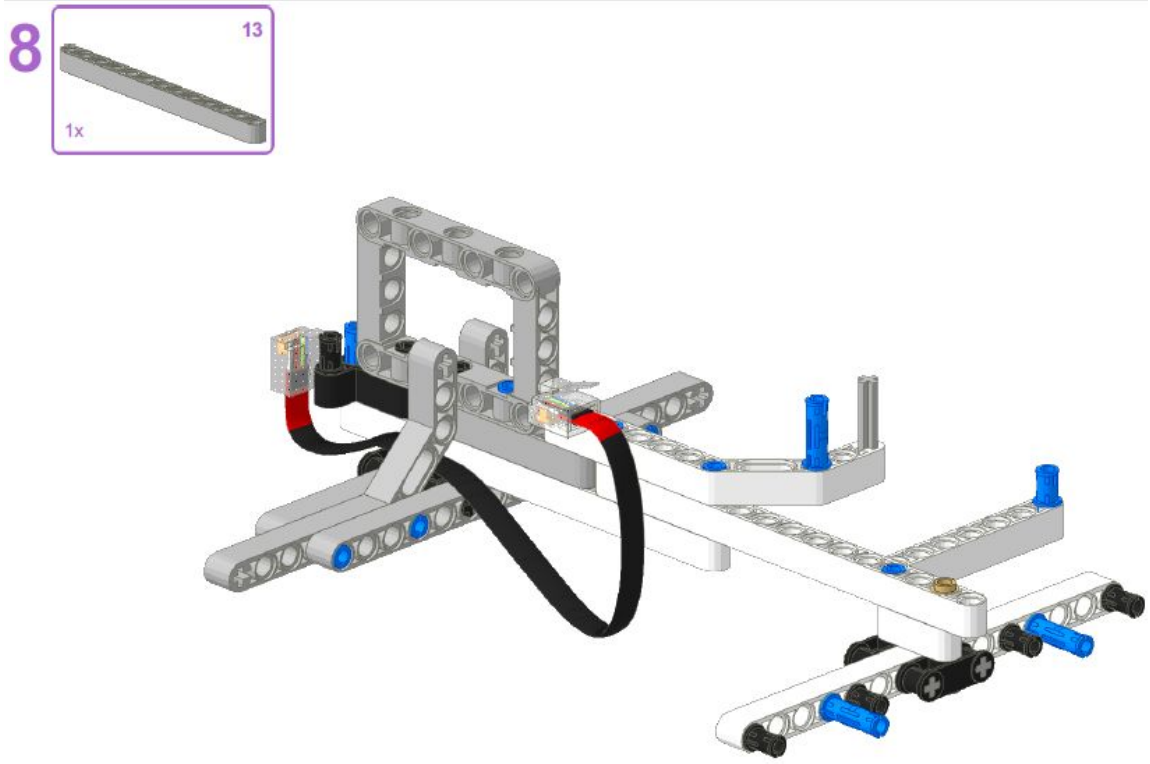

5
15
1x
4x
7
1x
3
1x
1x
1x
1x

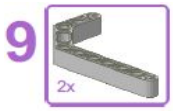


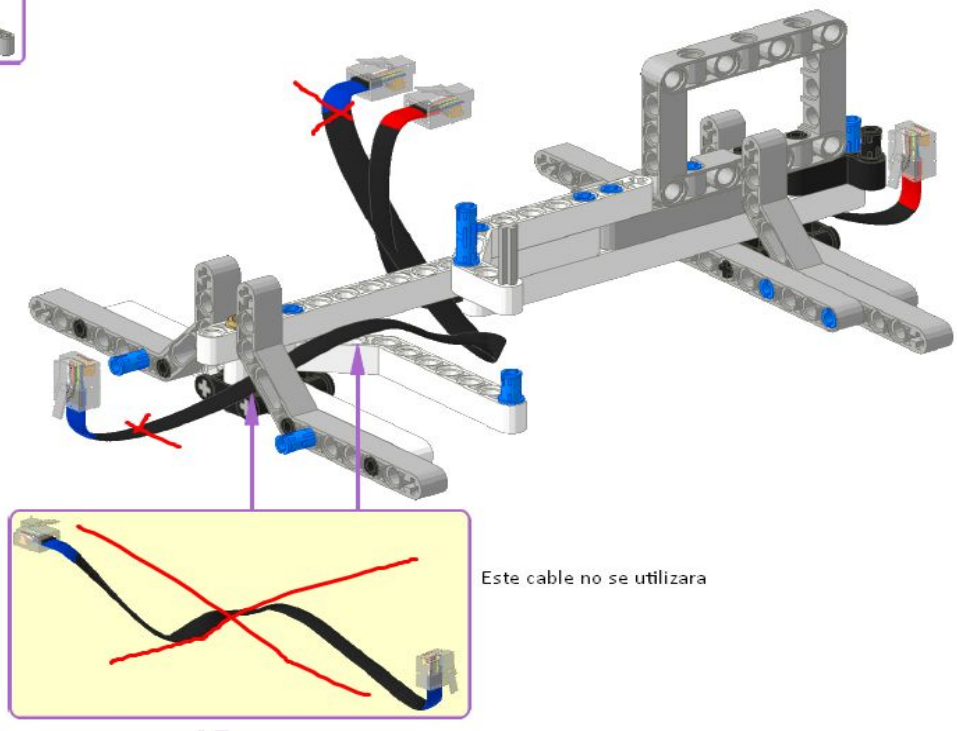

6
1x
3
1x
1x
1x



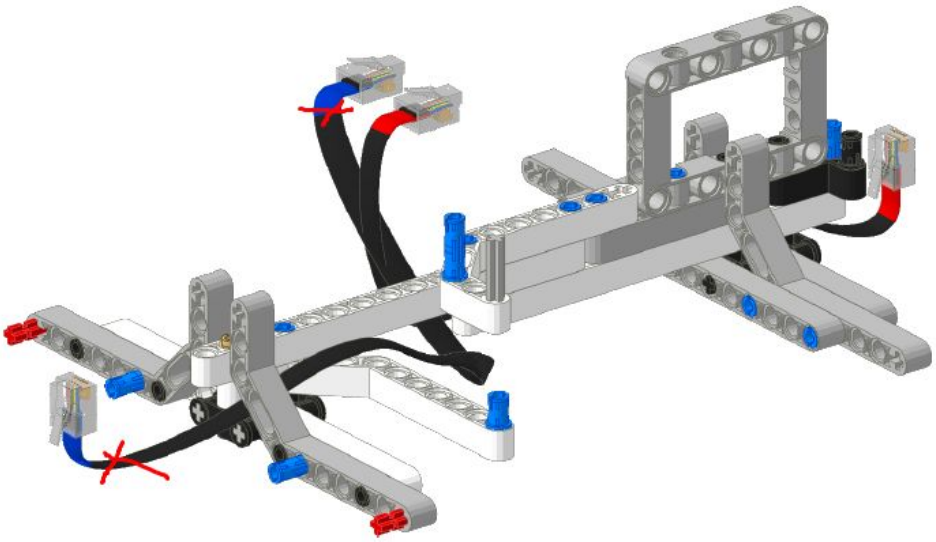

7
2x
25cm


8
13
1x

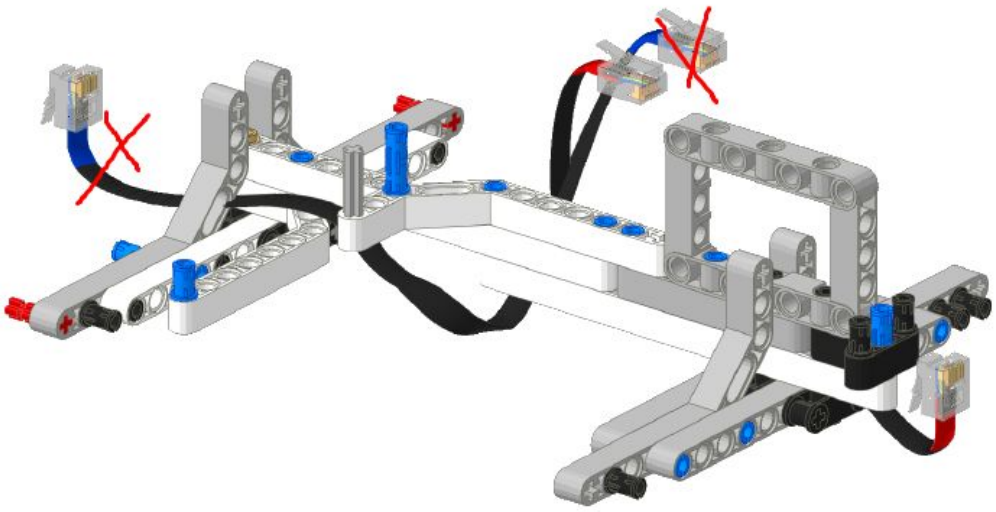

9
2x


Este cable no se utilizara

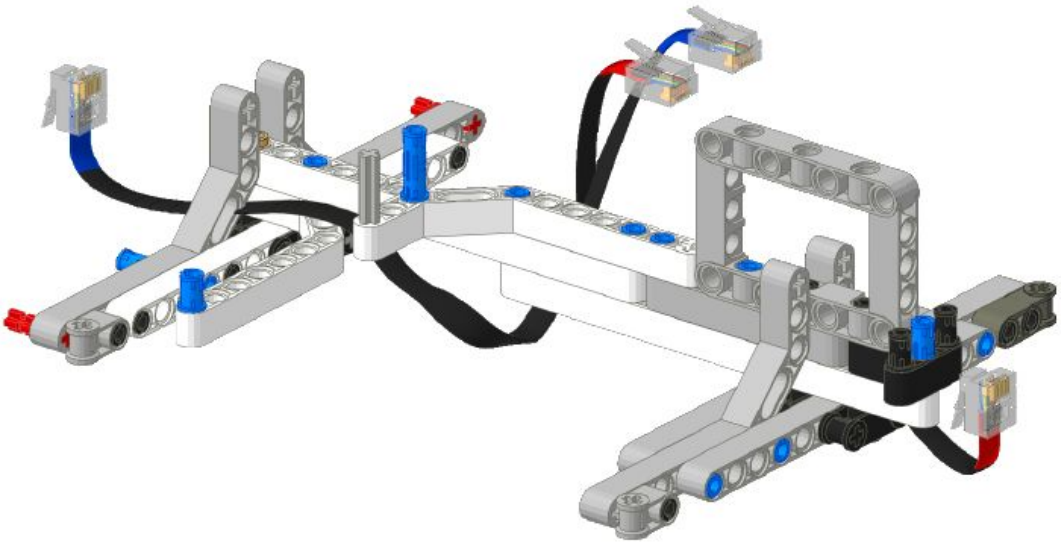

10
2
2x

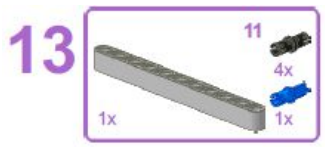


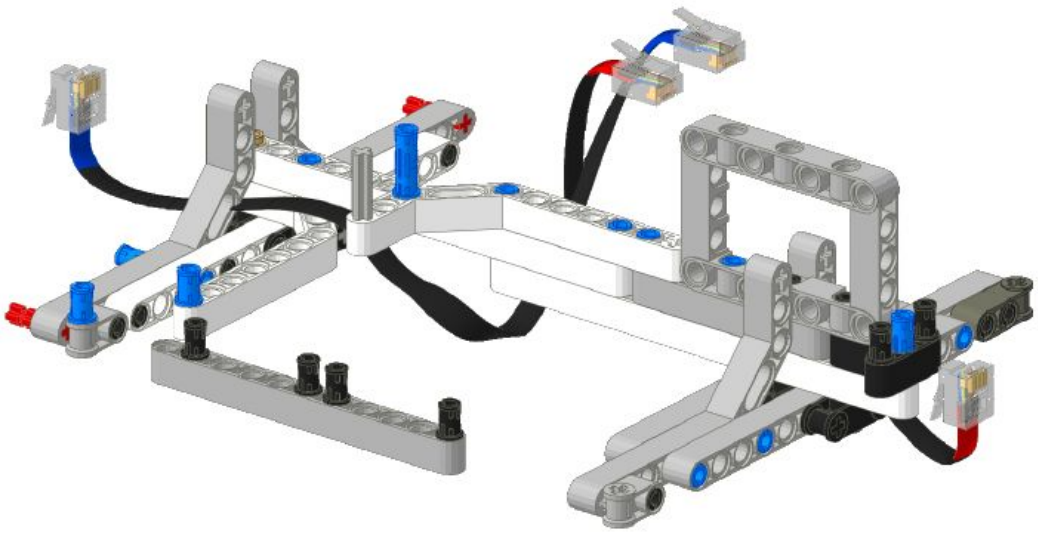
11 4x

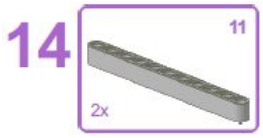


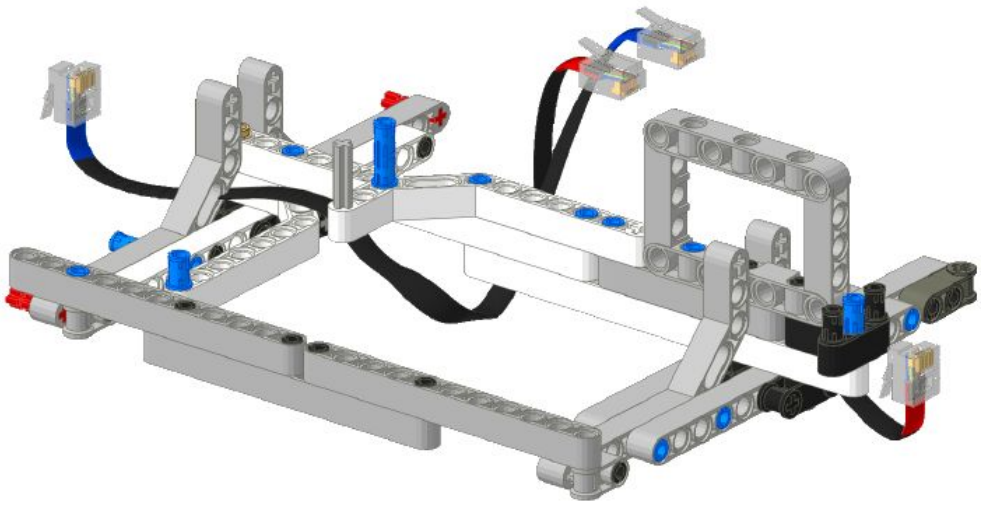
12 1x 2x



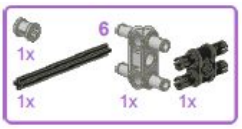

13
11
4x
1x
1x

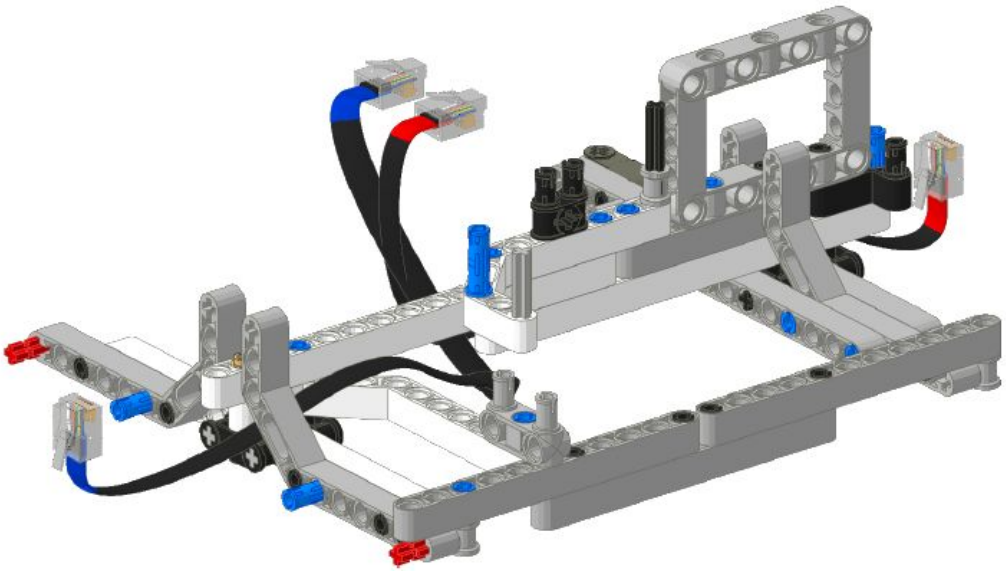



14
11
2x

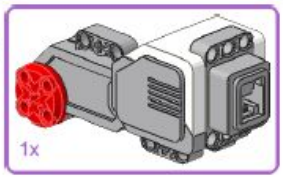


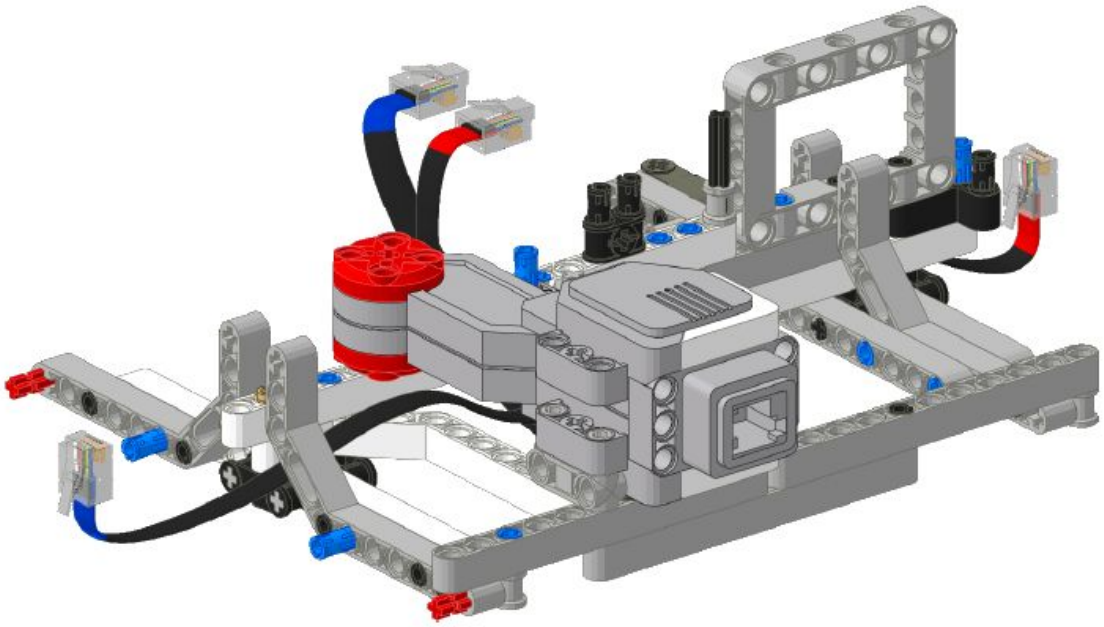
15



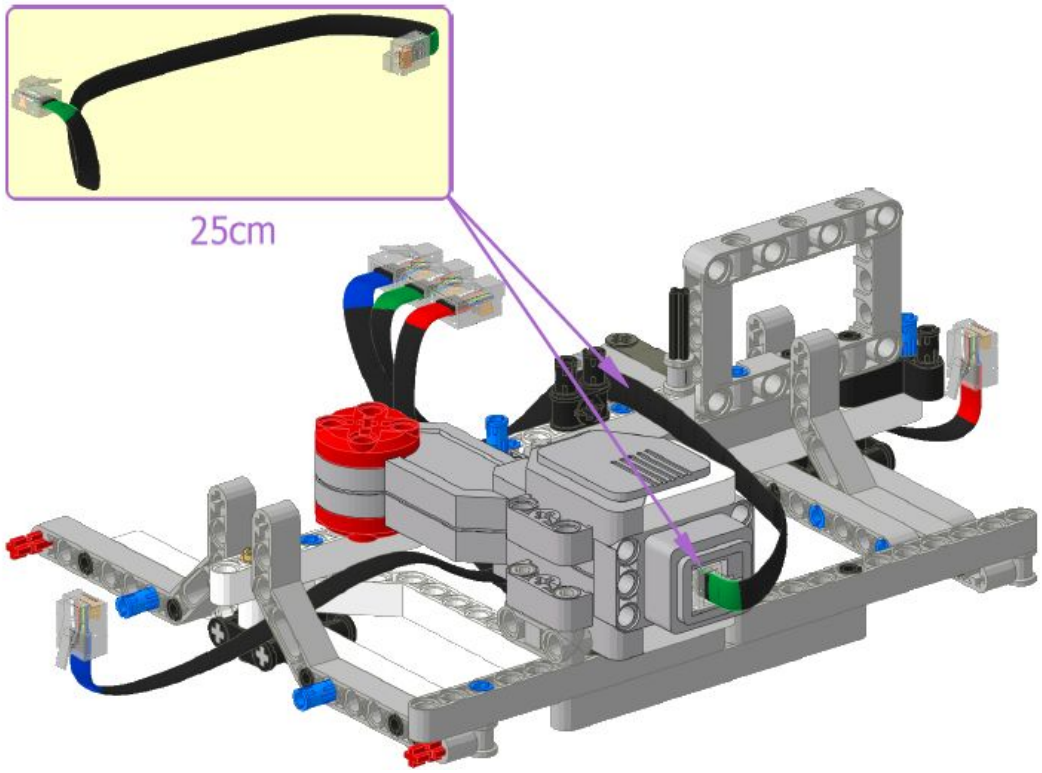


16

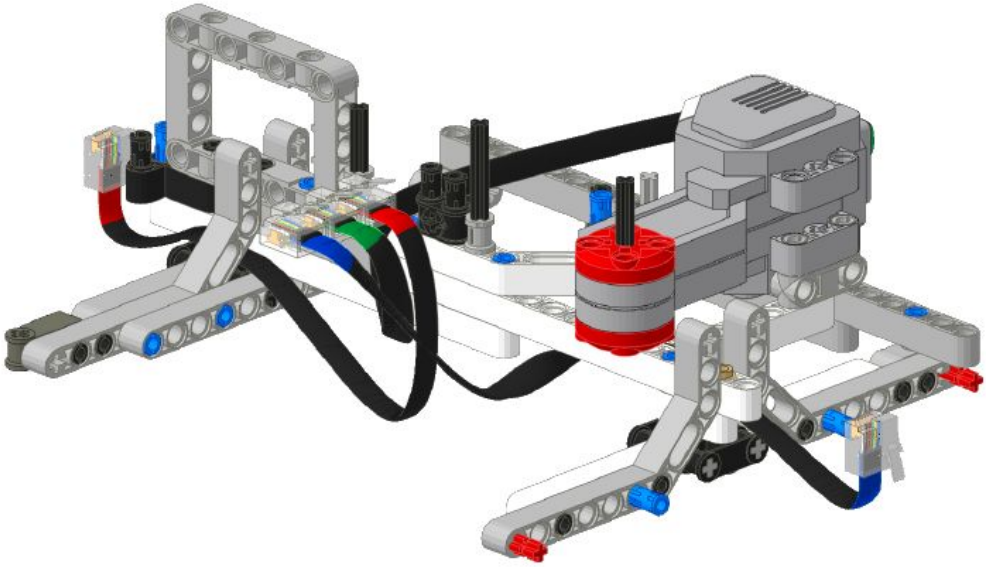


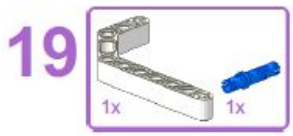


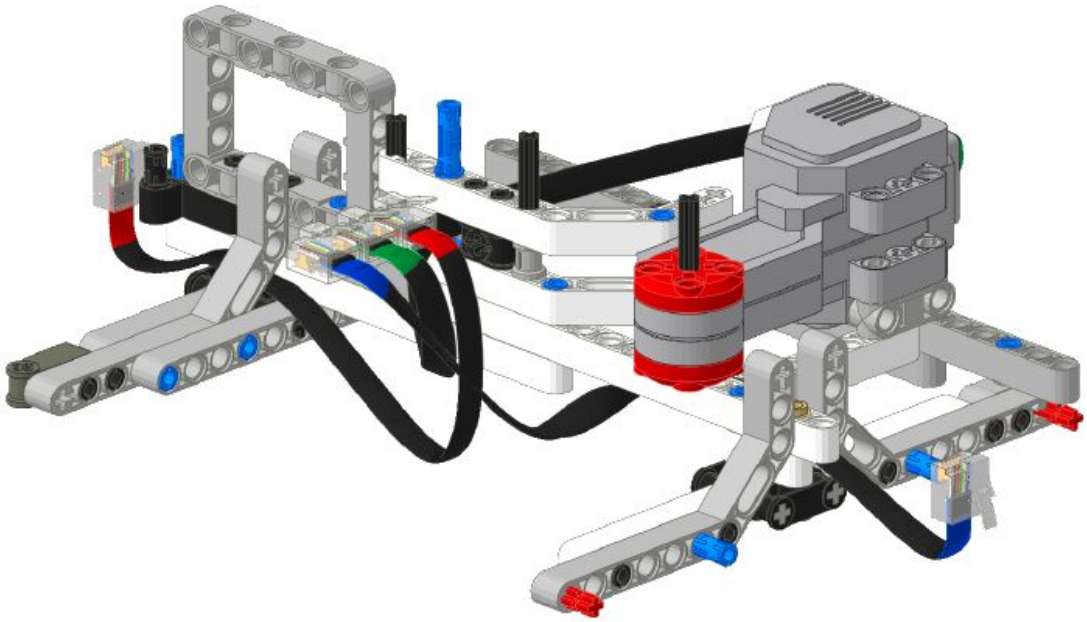
17

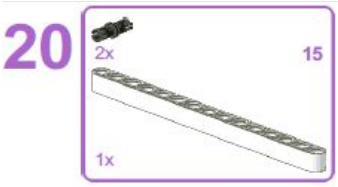


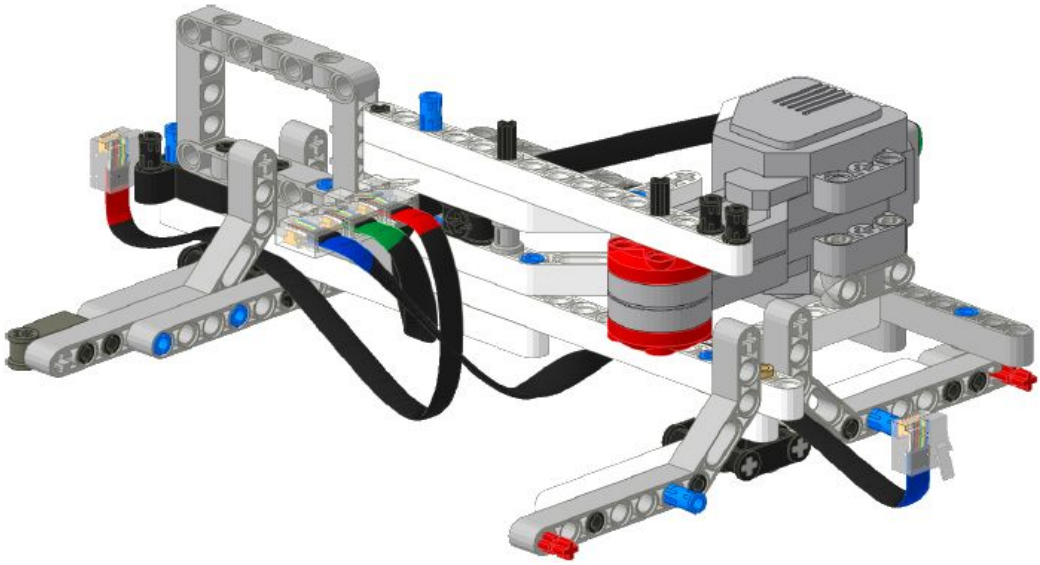
18



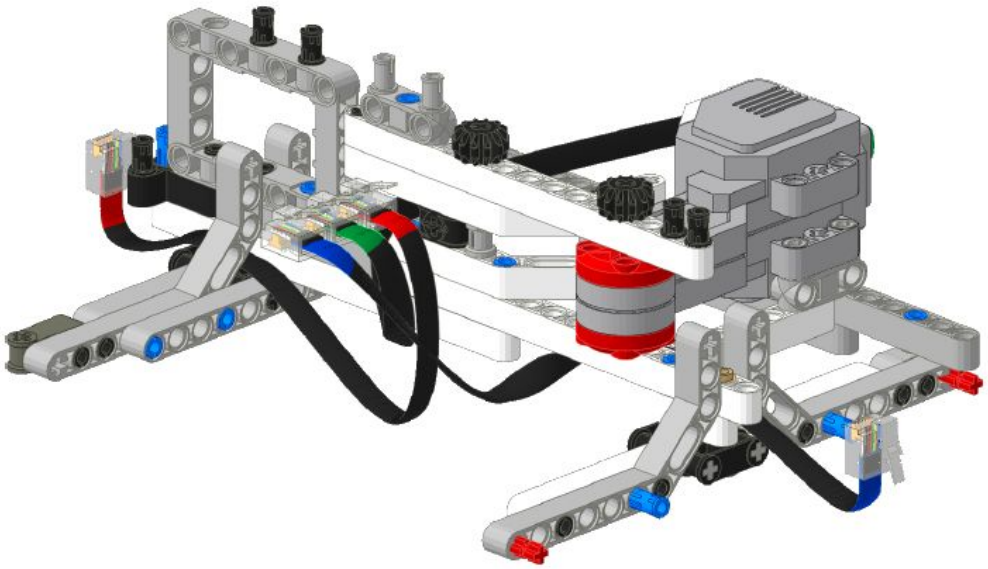

19
1x
1x

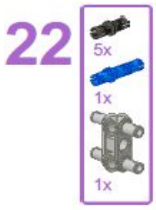


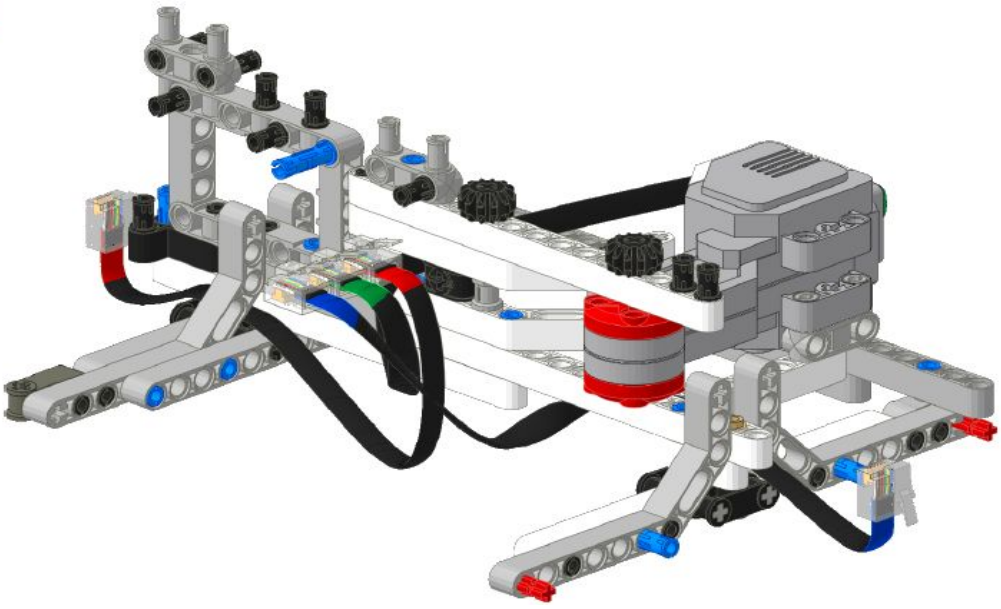

20
2x
15
1x

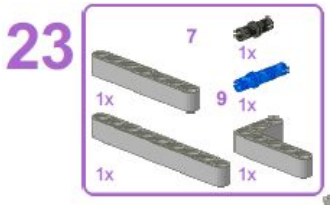


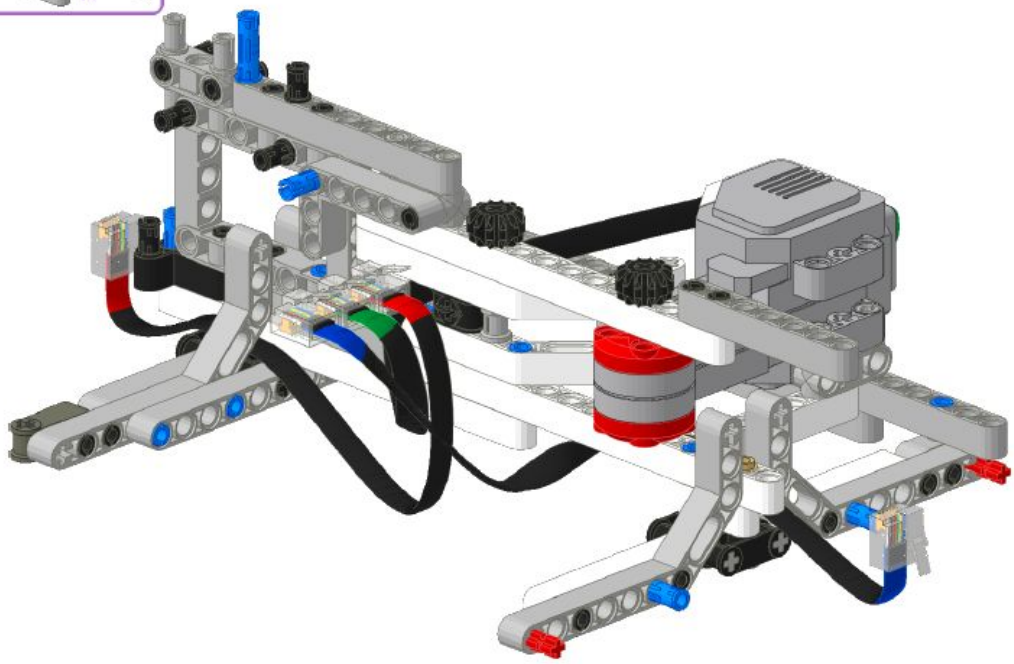

21
2x
1x
2x



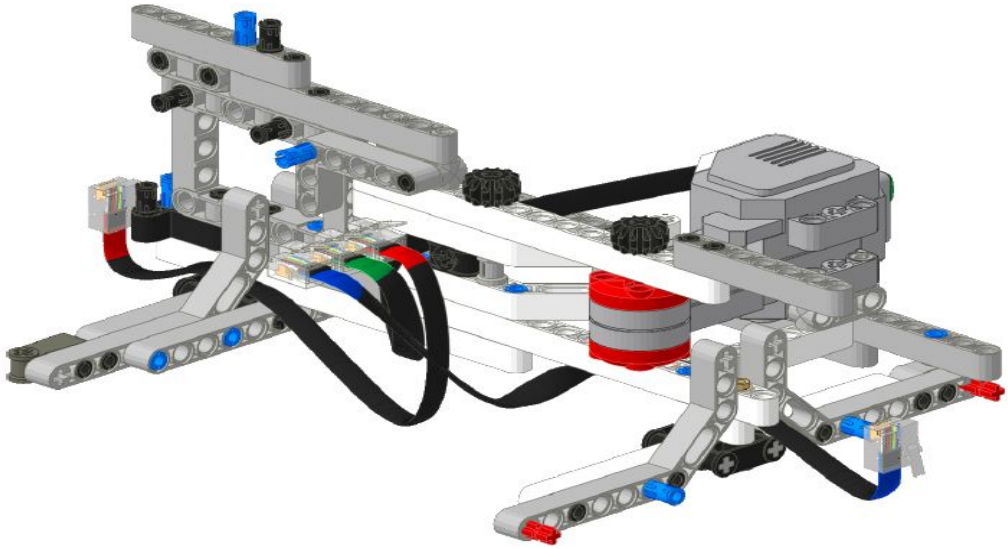

22
5x
1x
1x



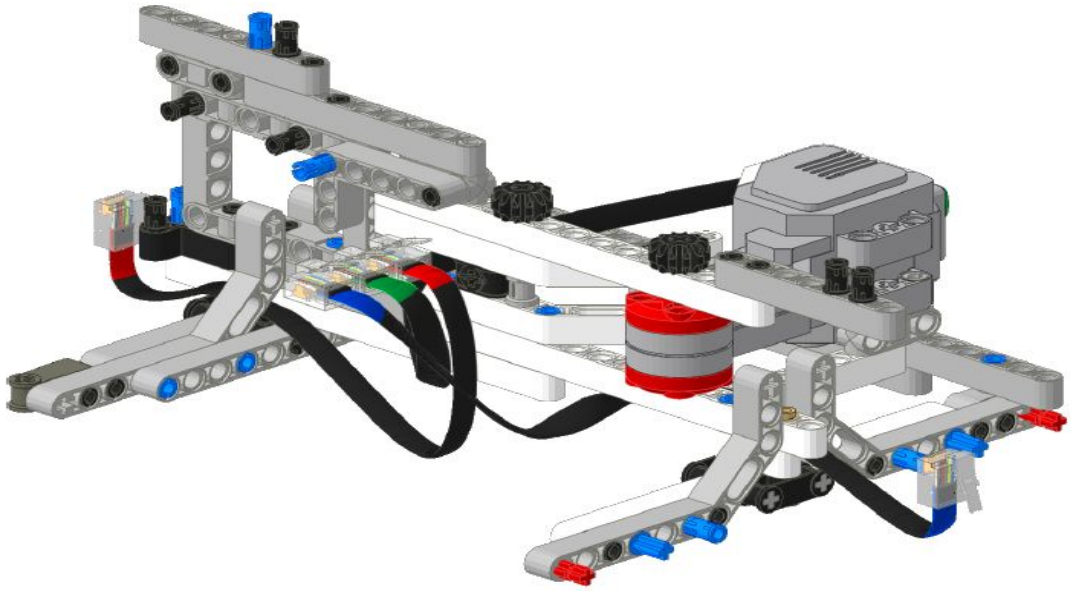

23
7
1x
1x
9
1x
1x
1x

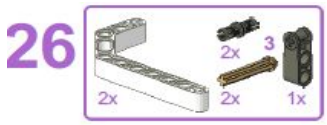


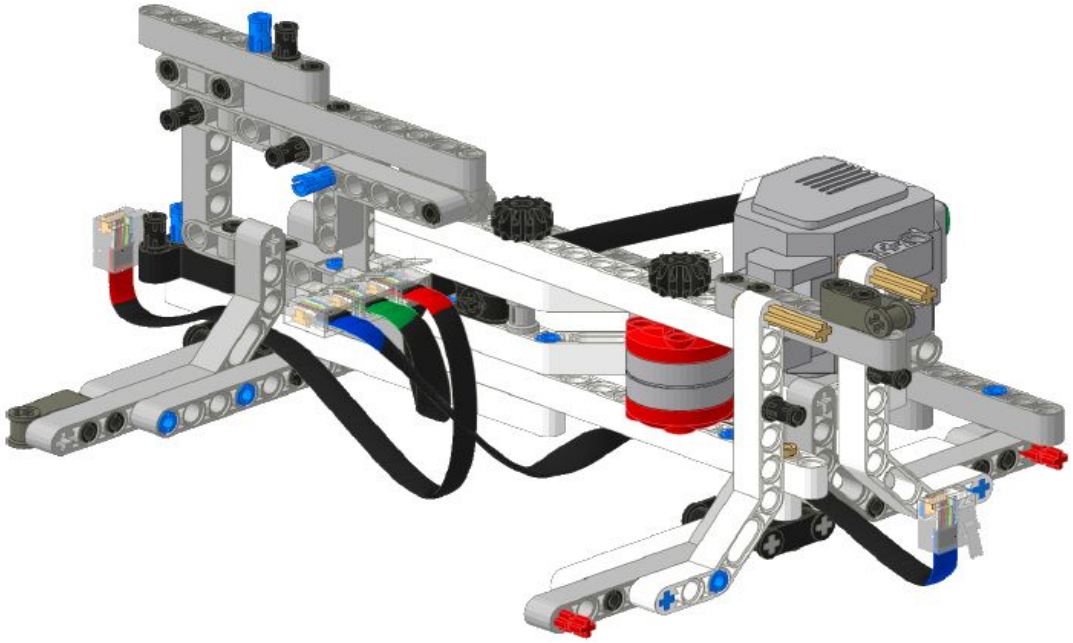

24
1x
7
1x



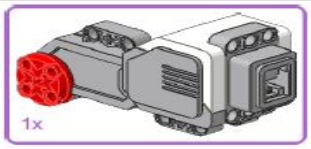

25
2x
2x

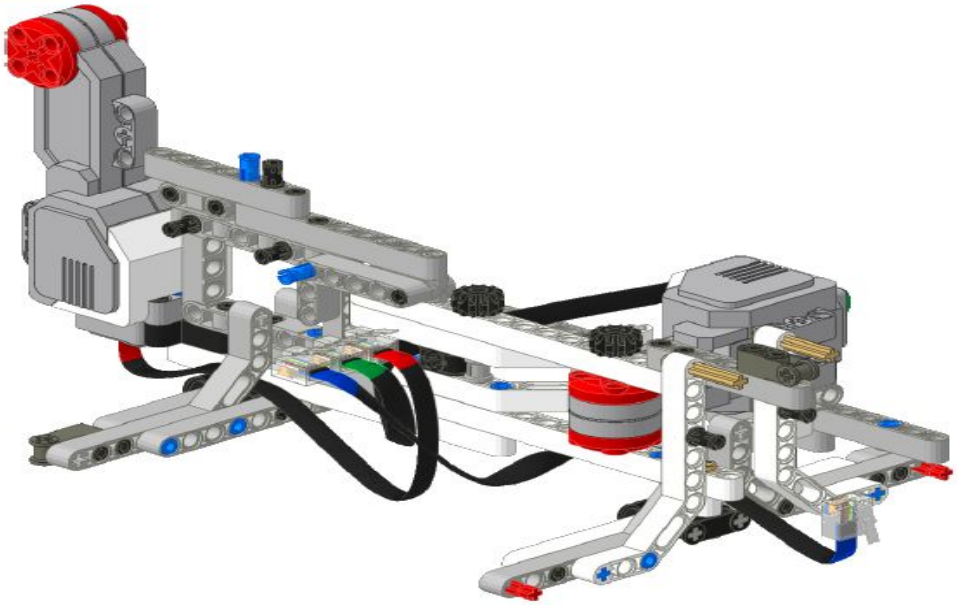



26
2x
2x
3
2x
1x

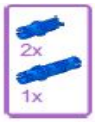


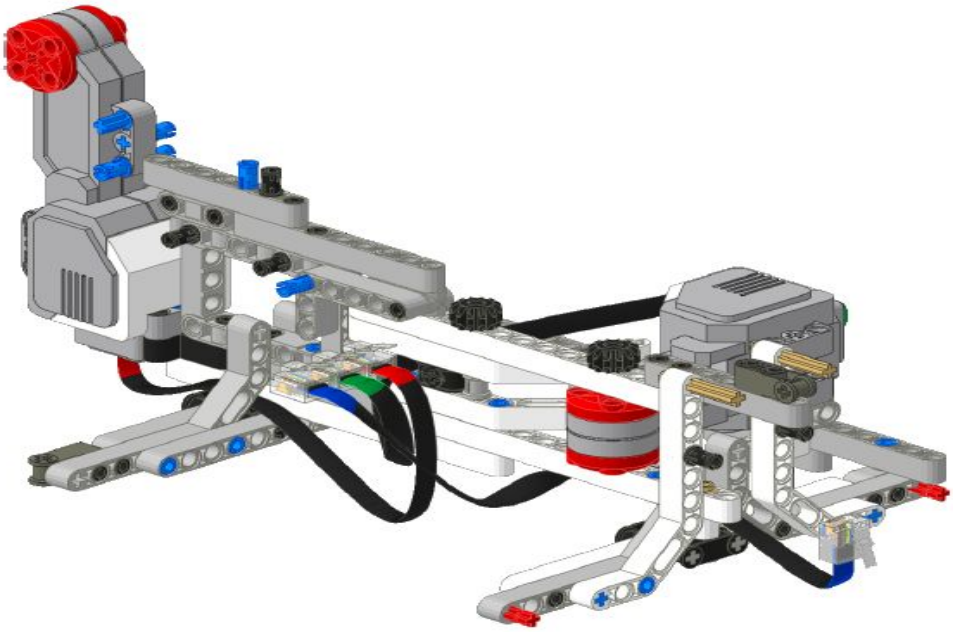
27

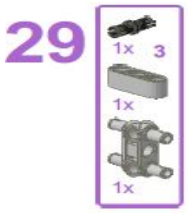


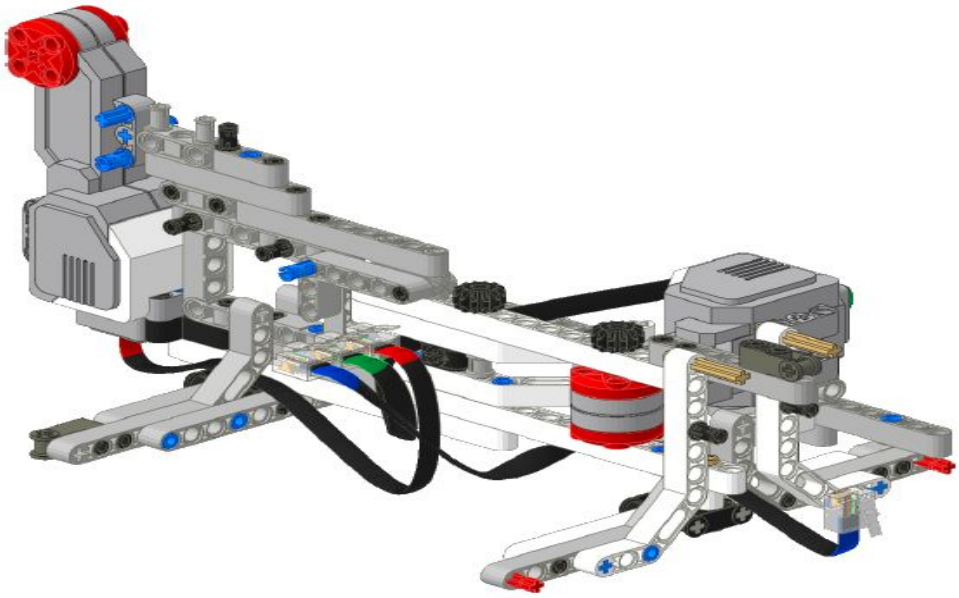


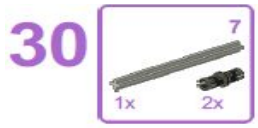
28

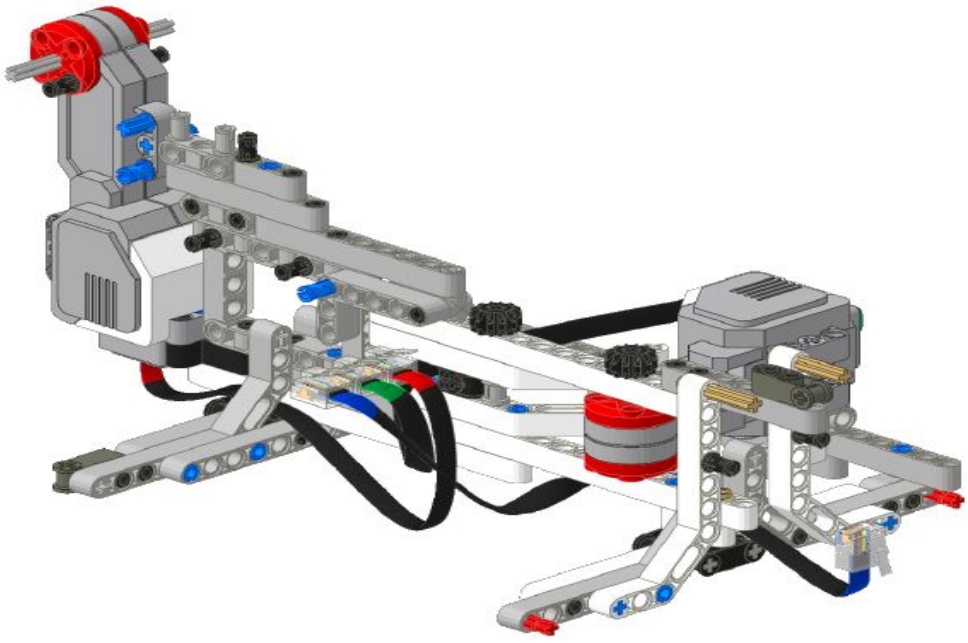




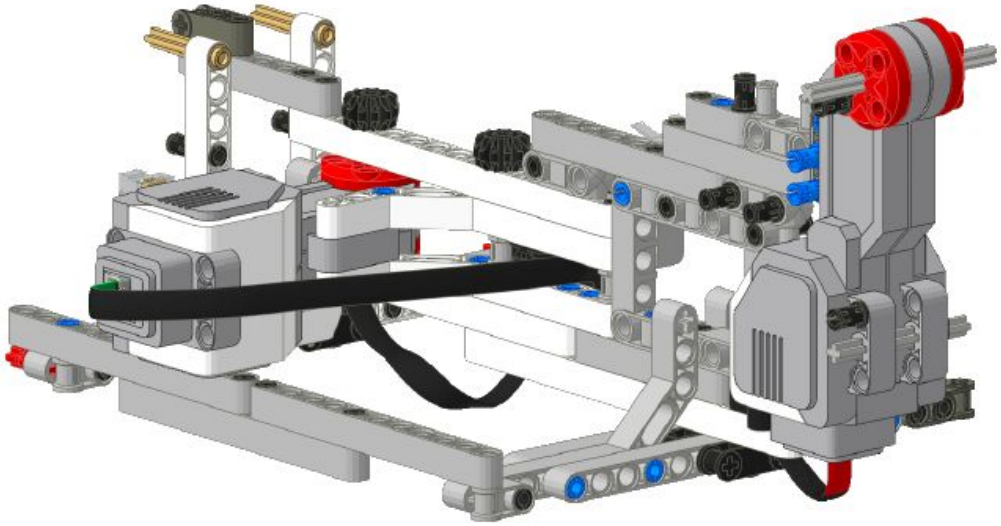

29
1x 3
1x
1x

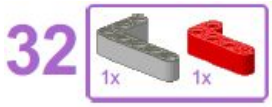


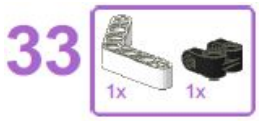

30
7
1x
2x

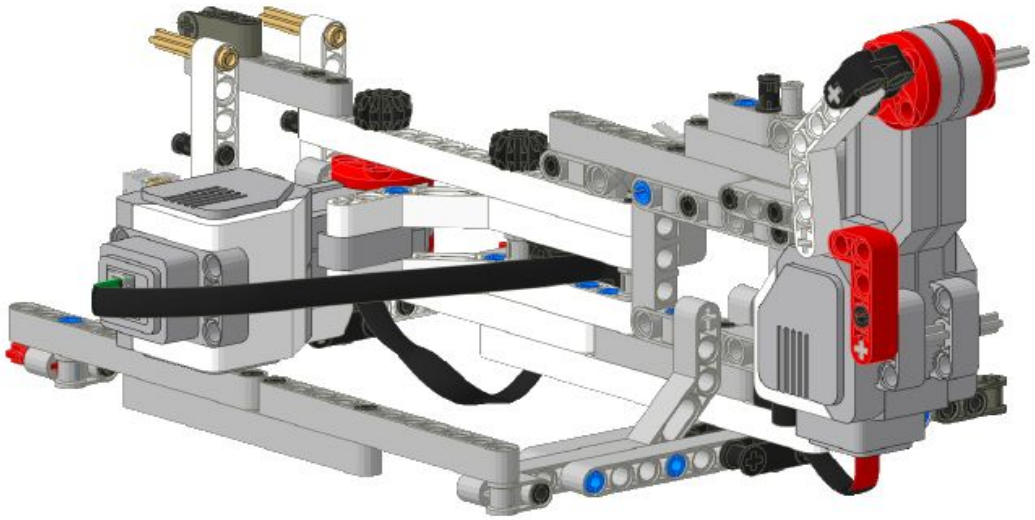



31
5
1x
1x

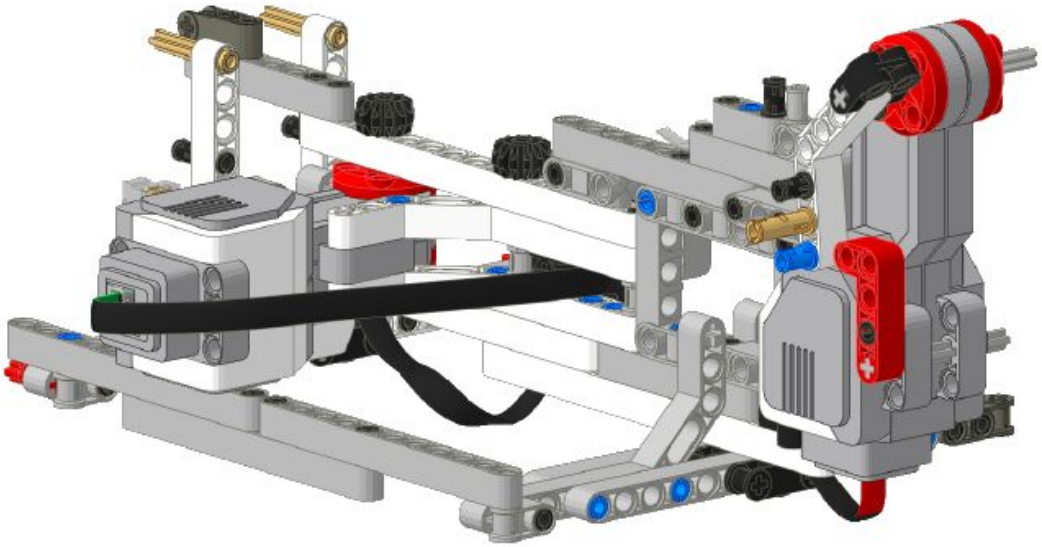


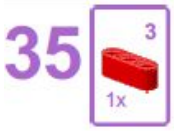

32
1x
1x

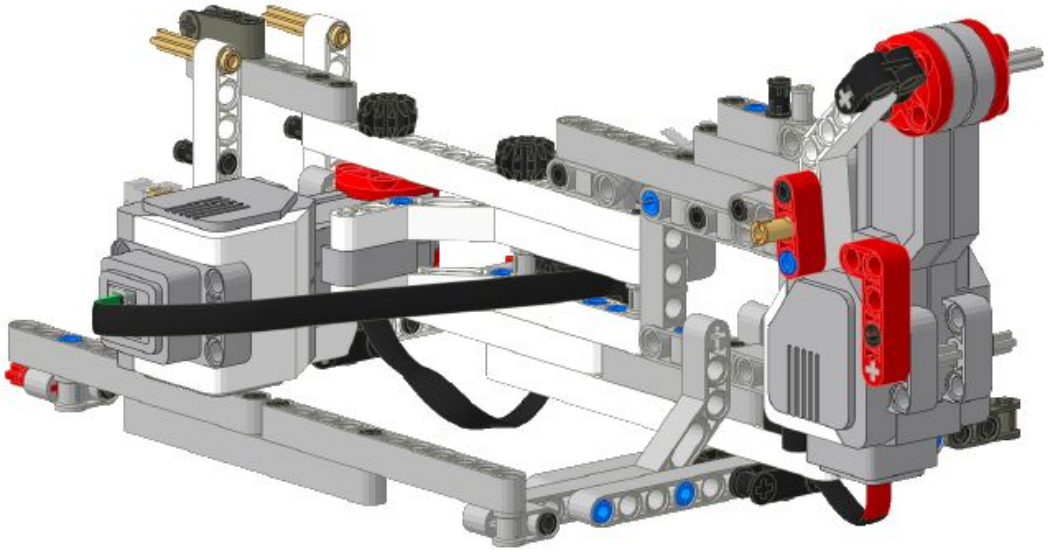

33
1x
1x

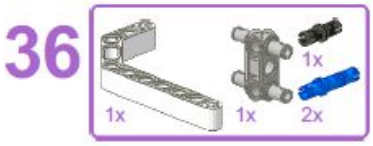


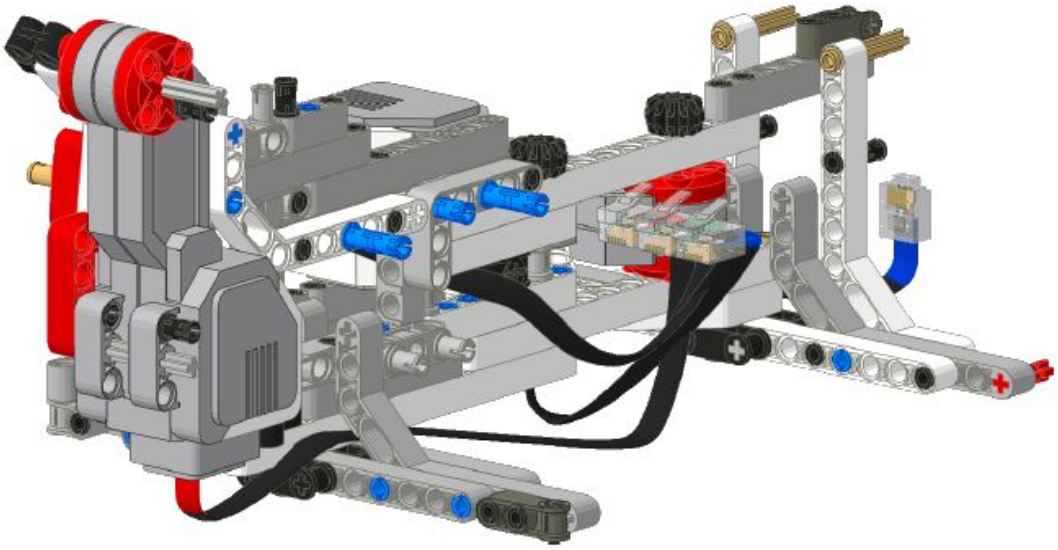

34
1x
1x
1x

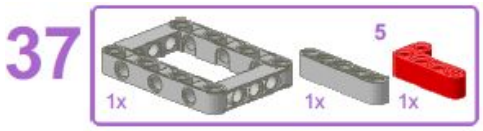


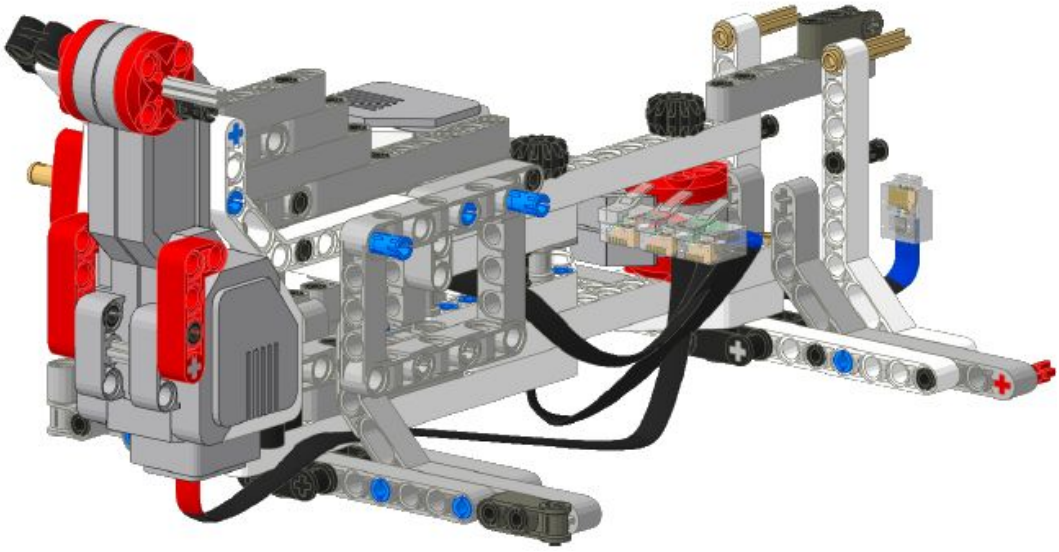

35
3
1x

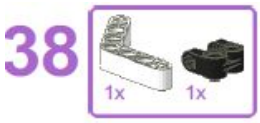


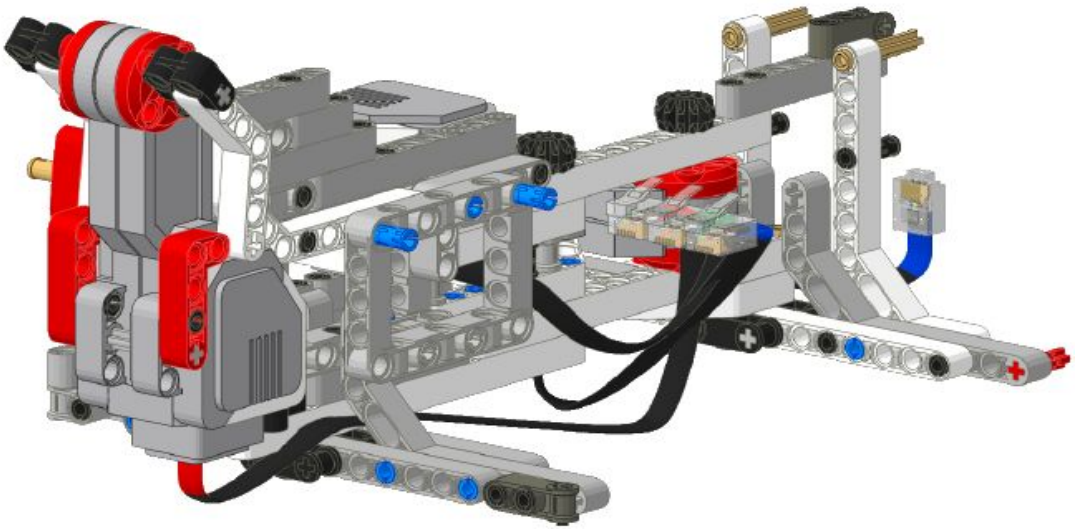

36
1x
1x
1x
2x



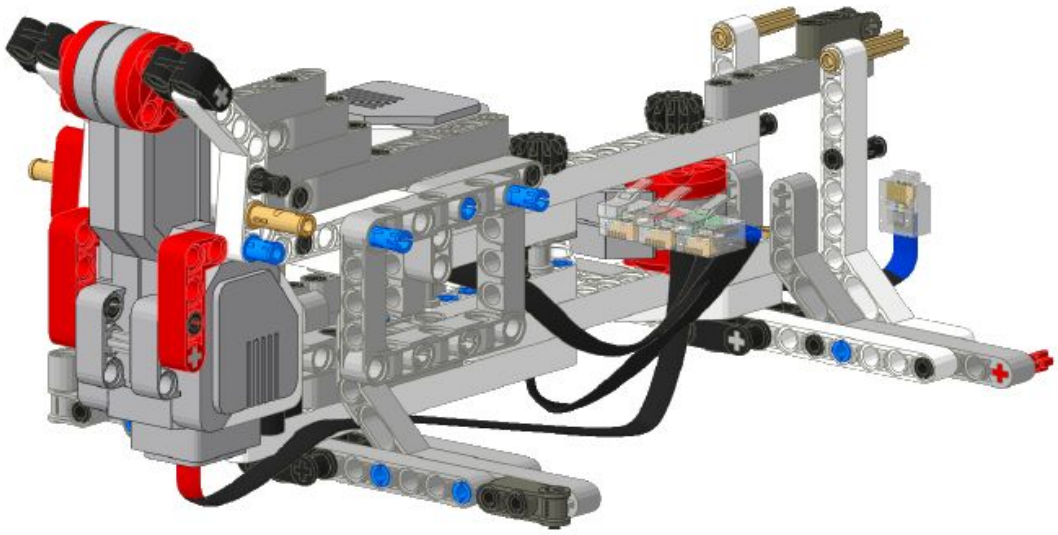

37
5
1x
1x
1x

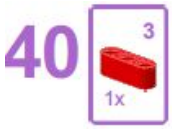


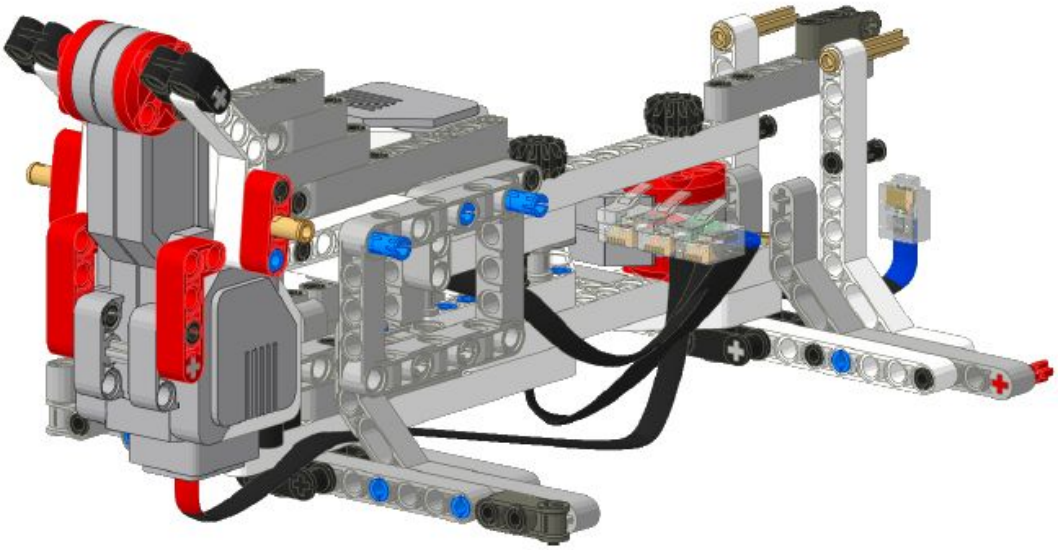

38
1x
1x



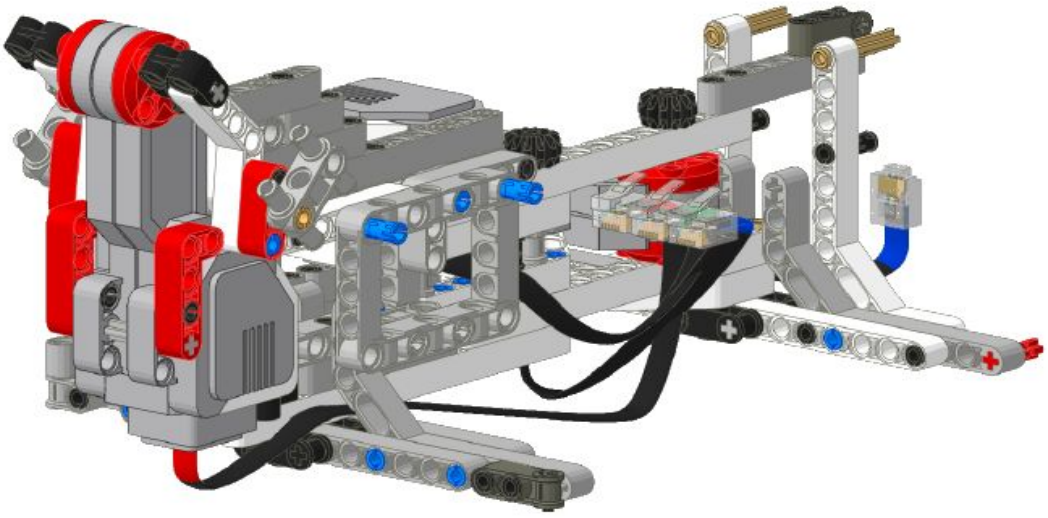

39
1x
1x
1x



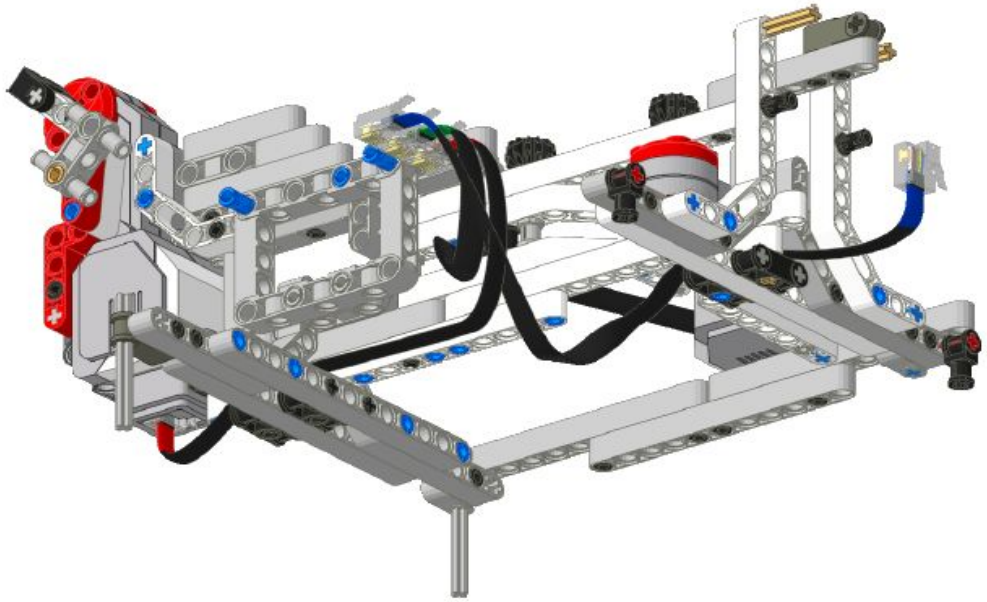

40
3
1x

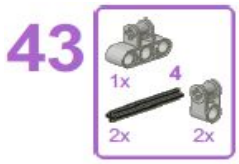


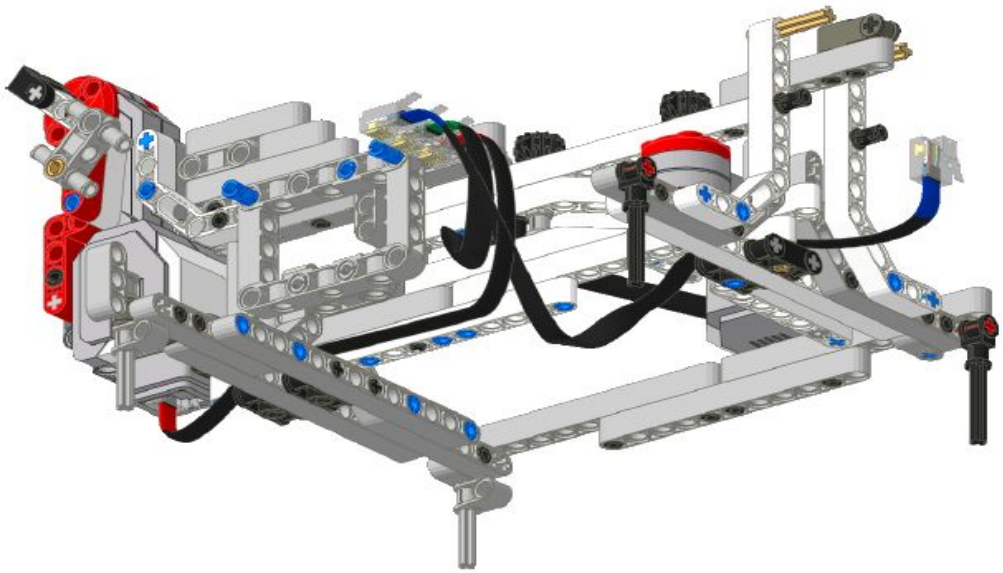

41
2x



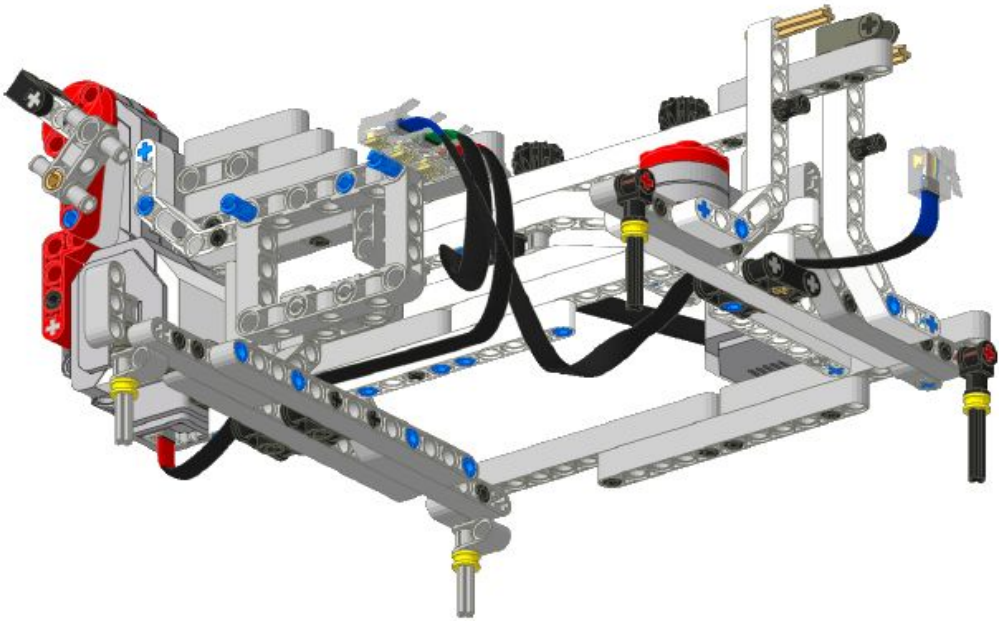

42
2x
5
2x

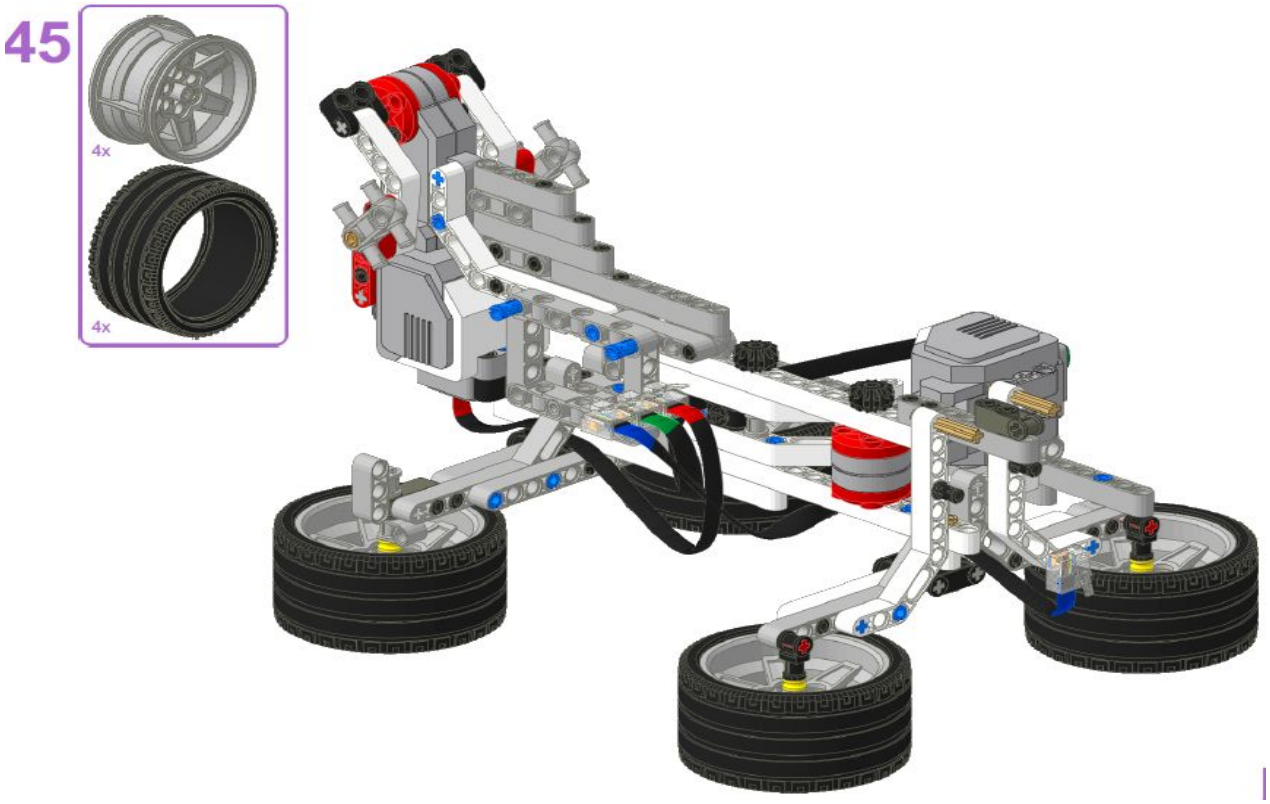



43
1x
4
2x
2x

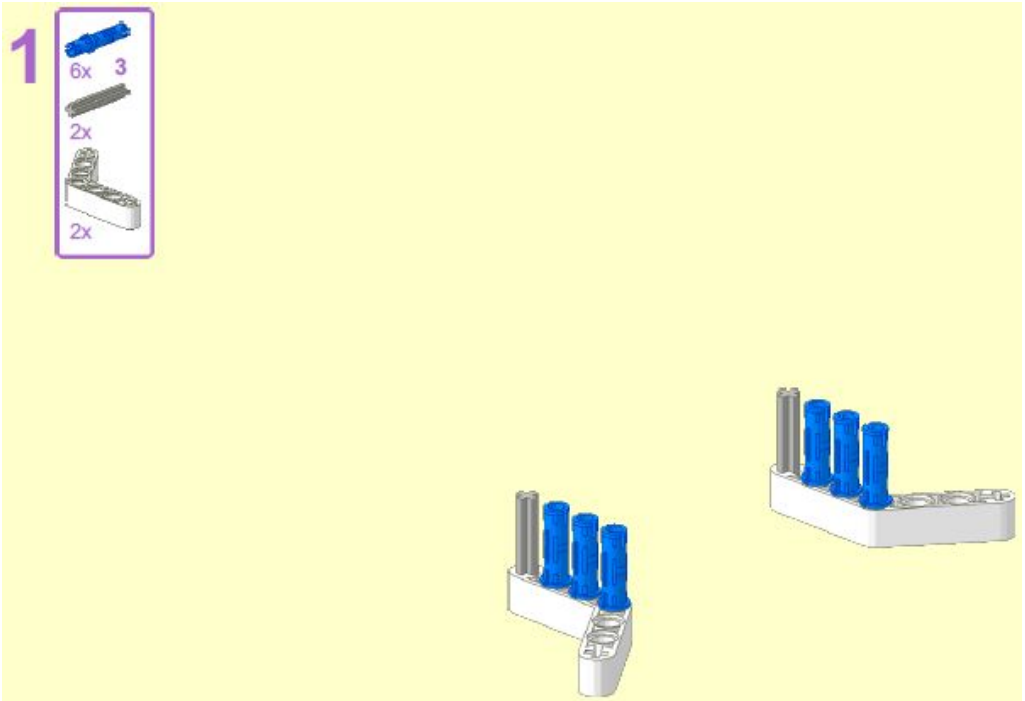


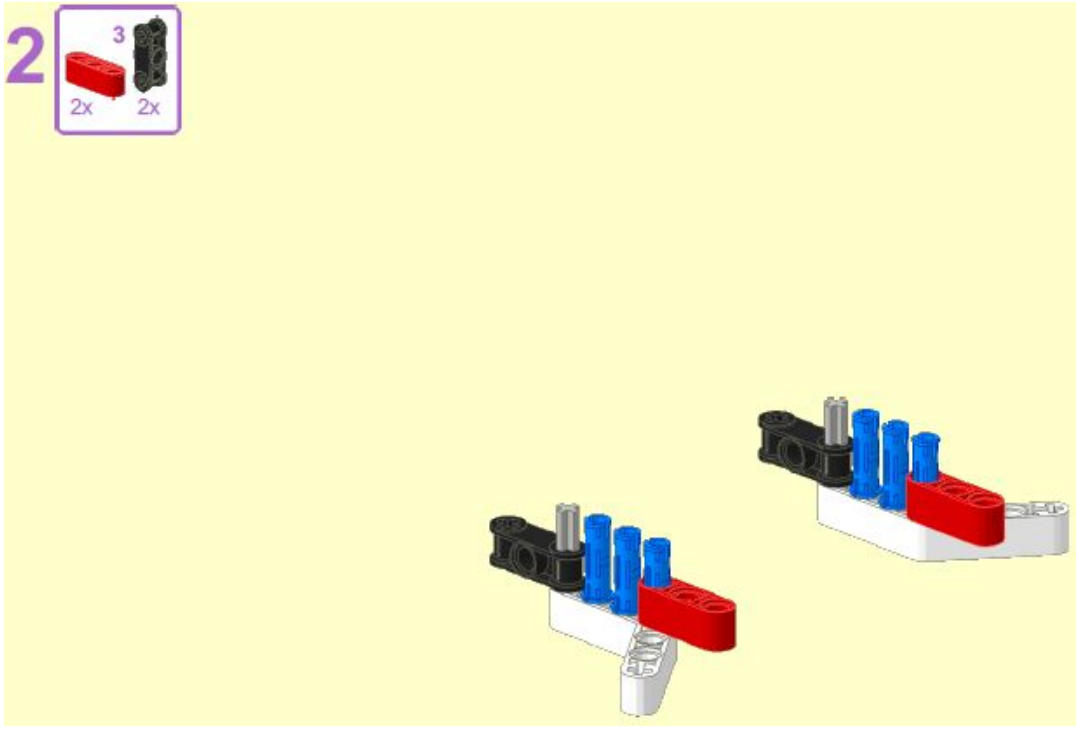

44
4x

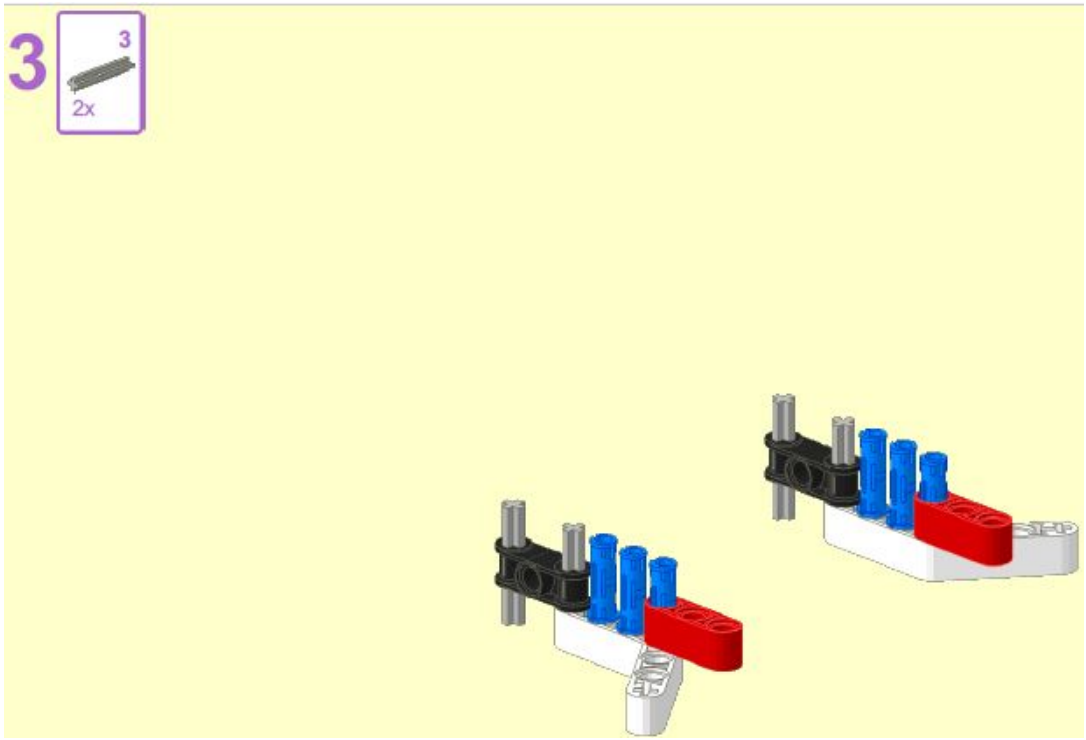


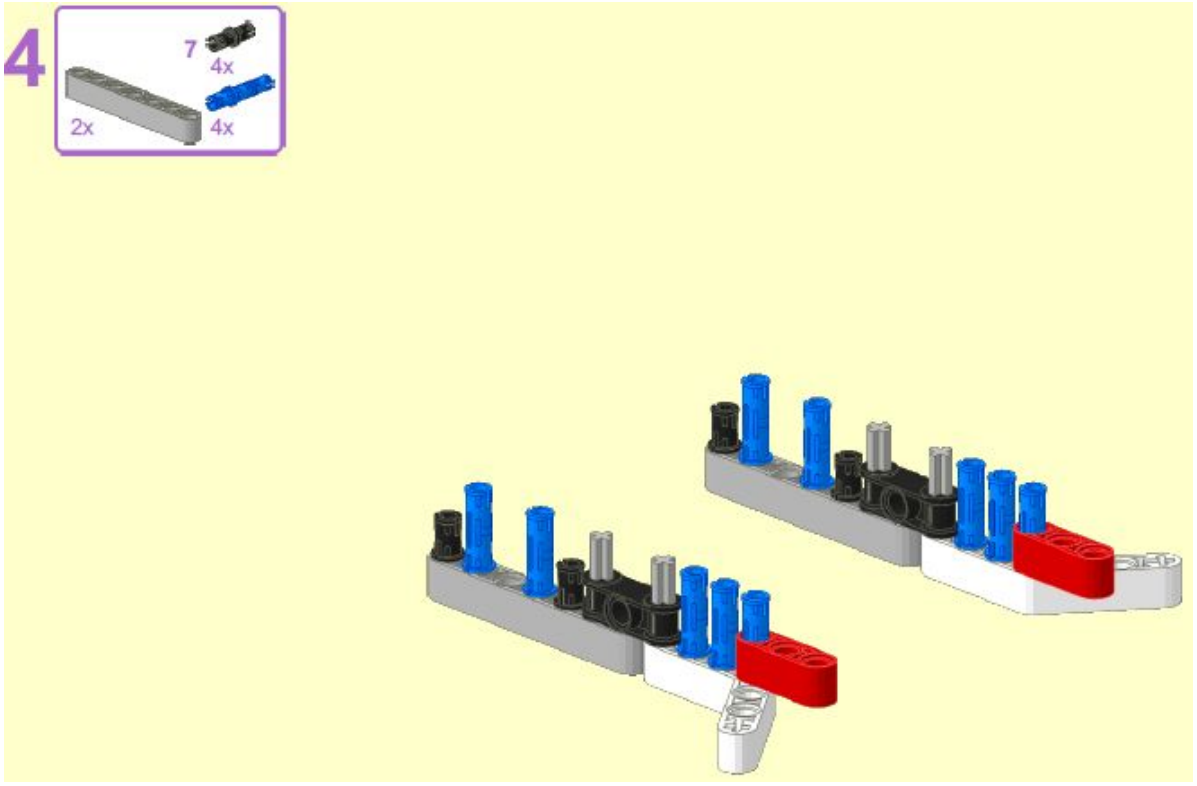


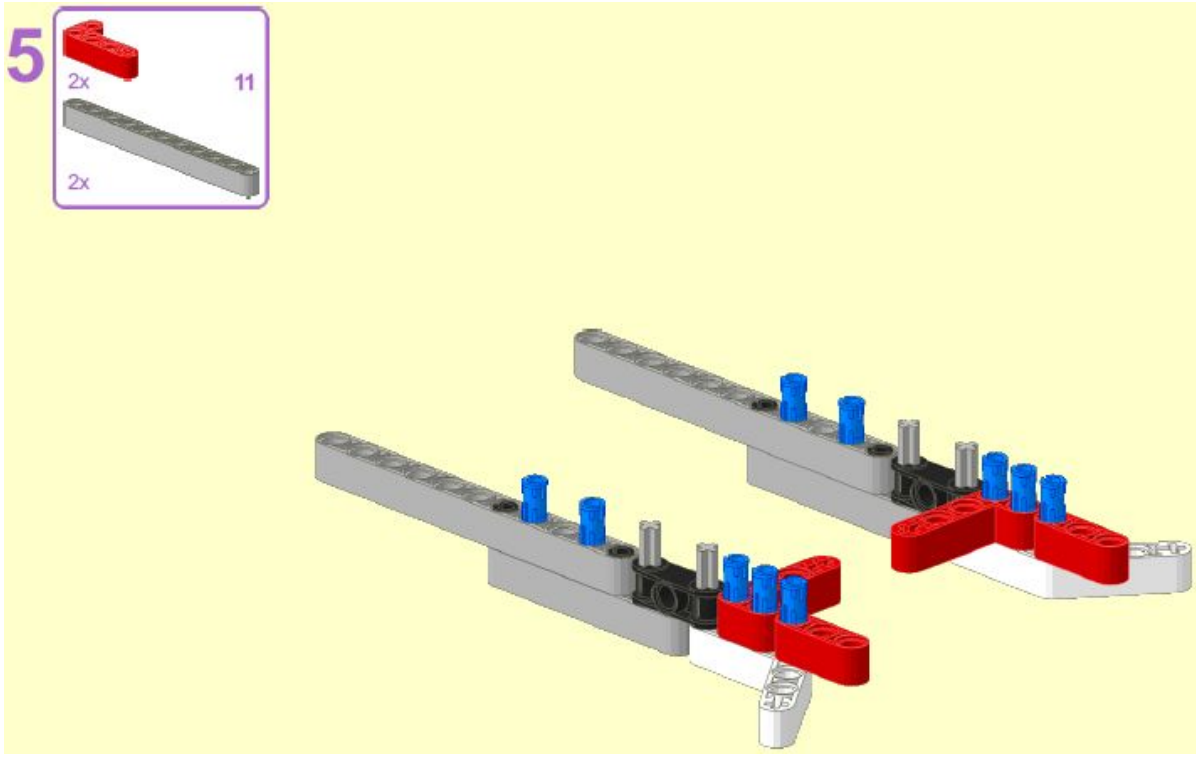
brazo:

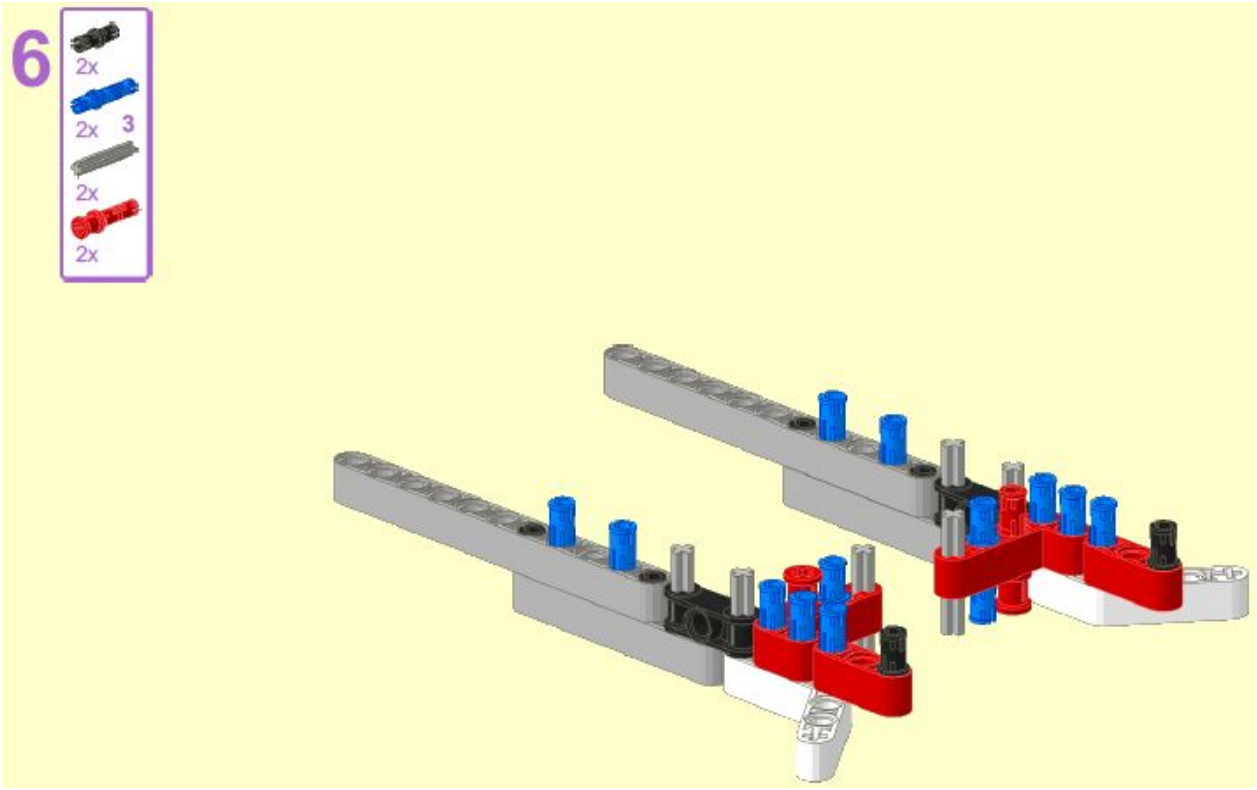


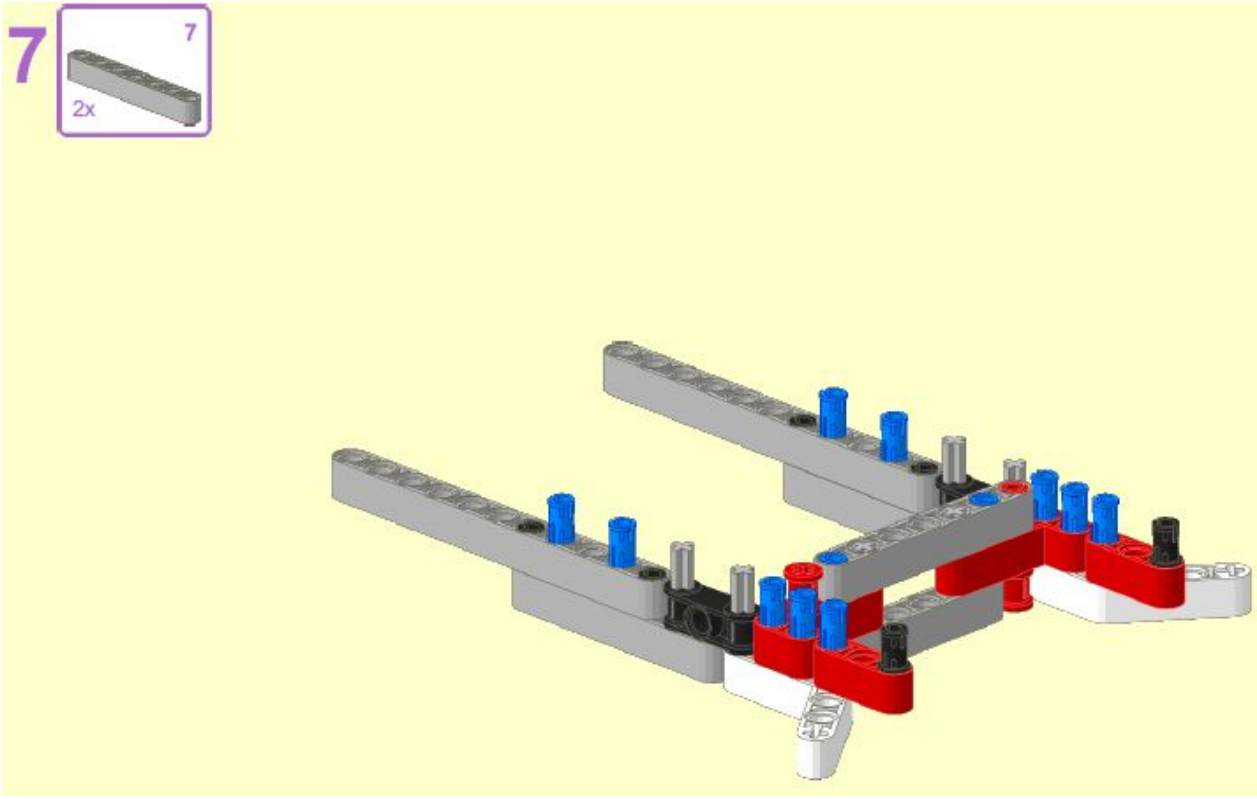

2
3
2x
2x

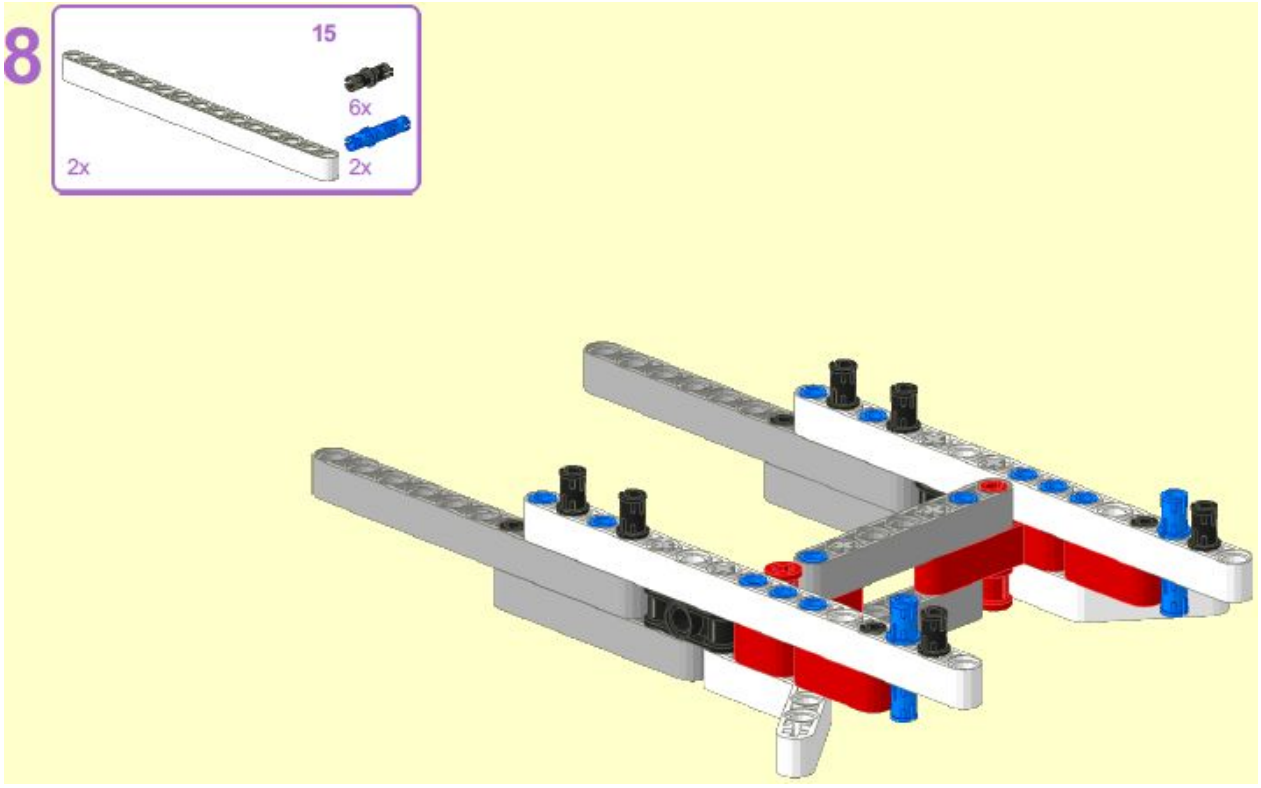

3
3
2x

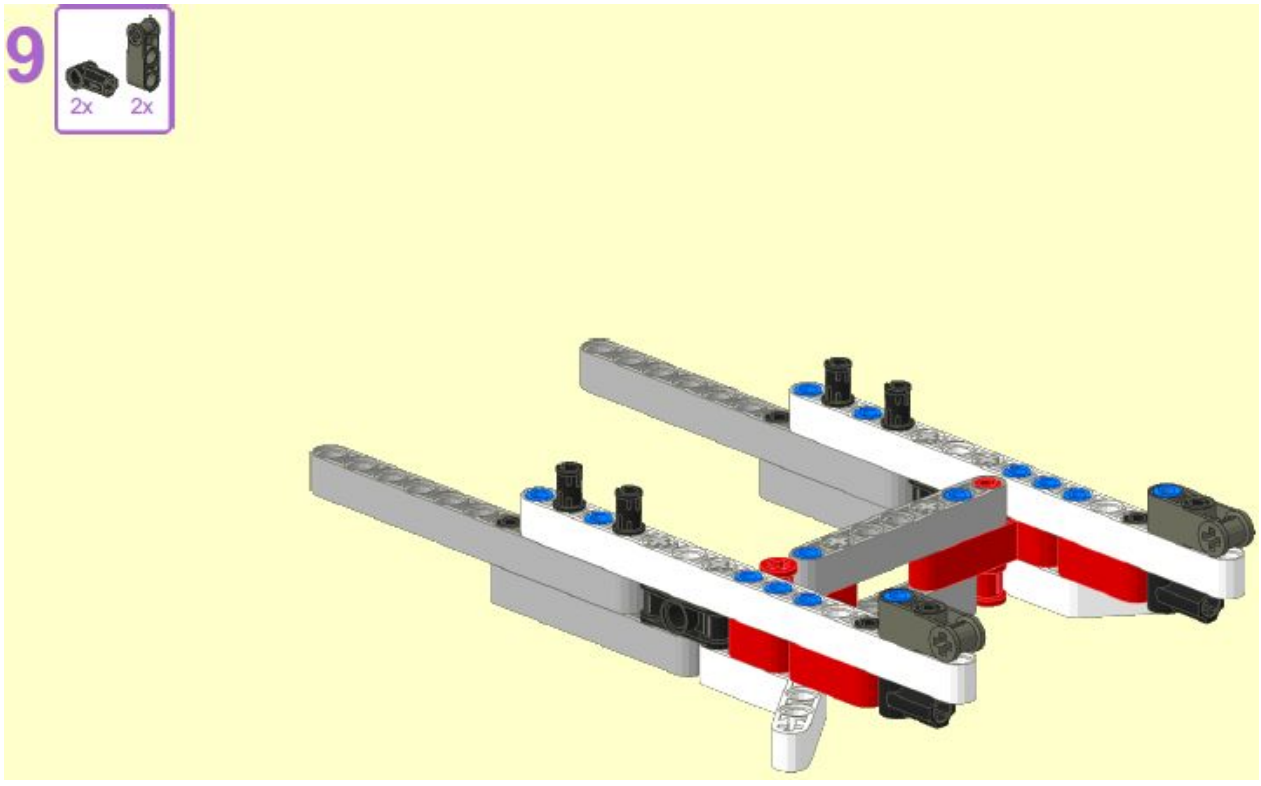

4
7
4x
2x
4x

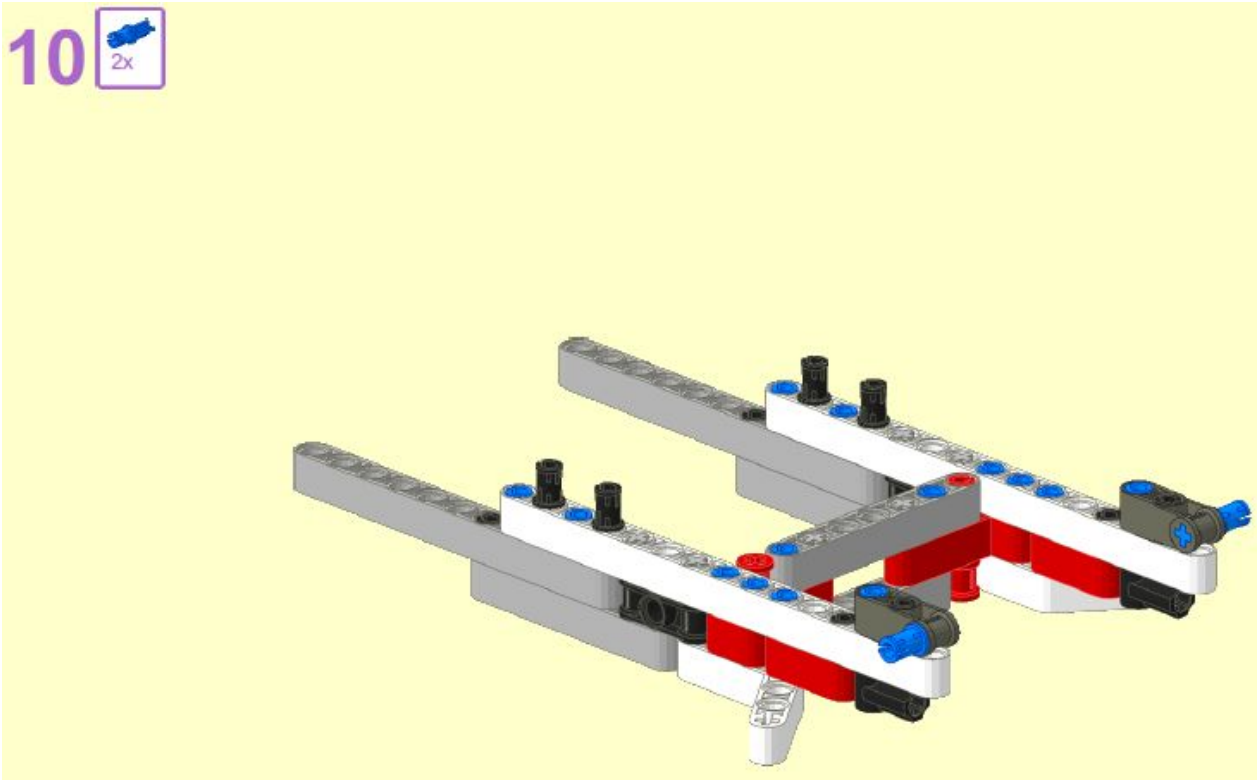

5
2x
11
2x

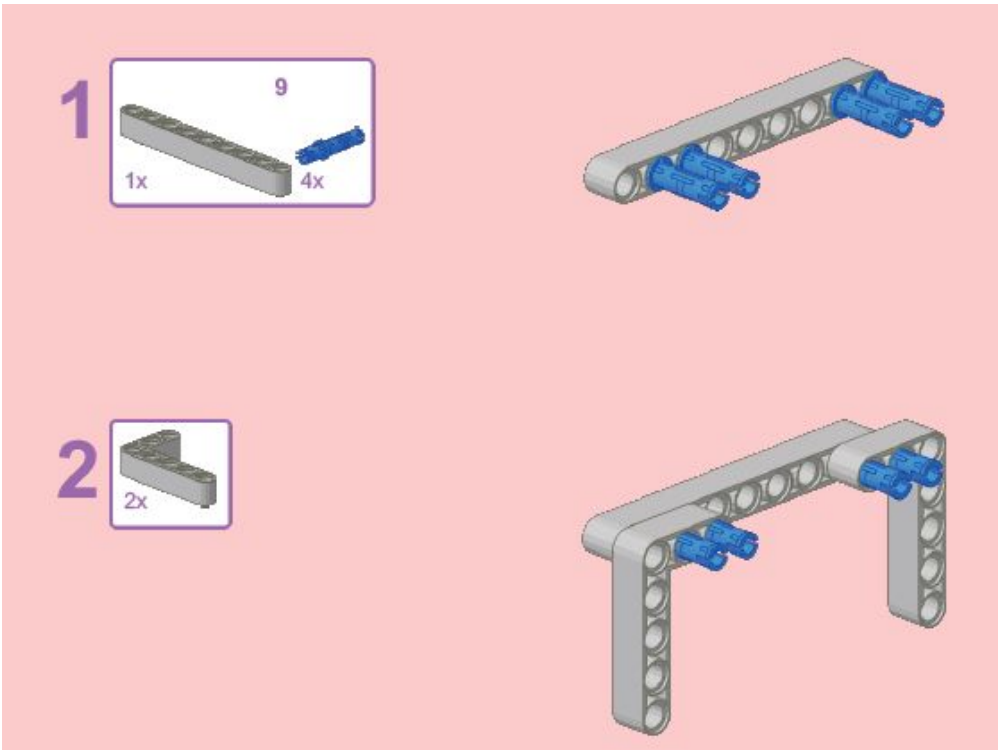

6
2x
2x
3
2x
2x

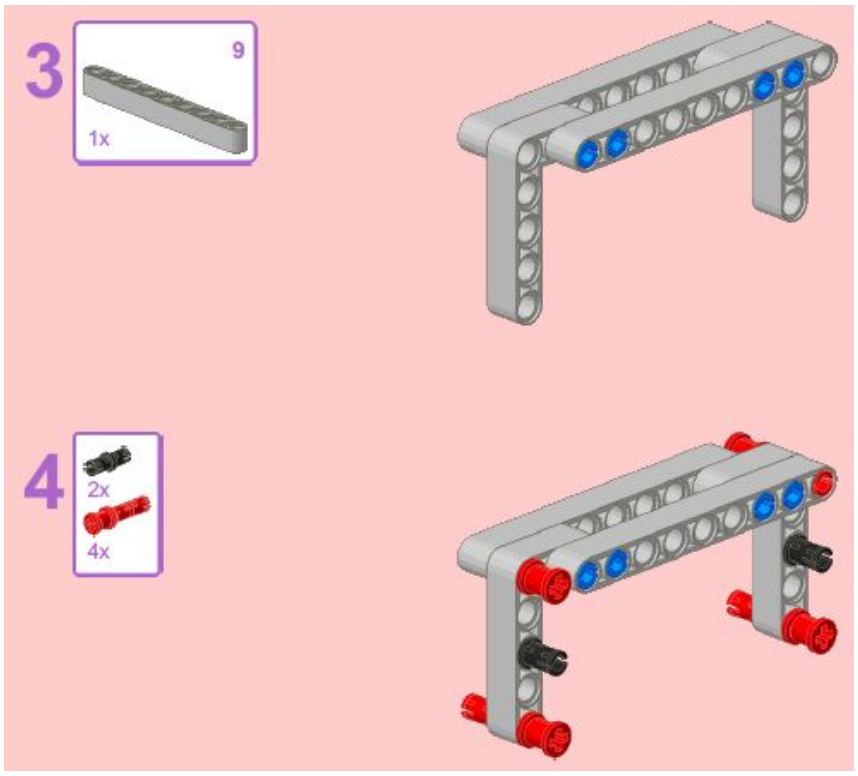

7
2x
7

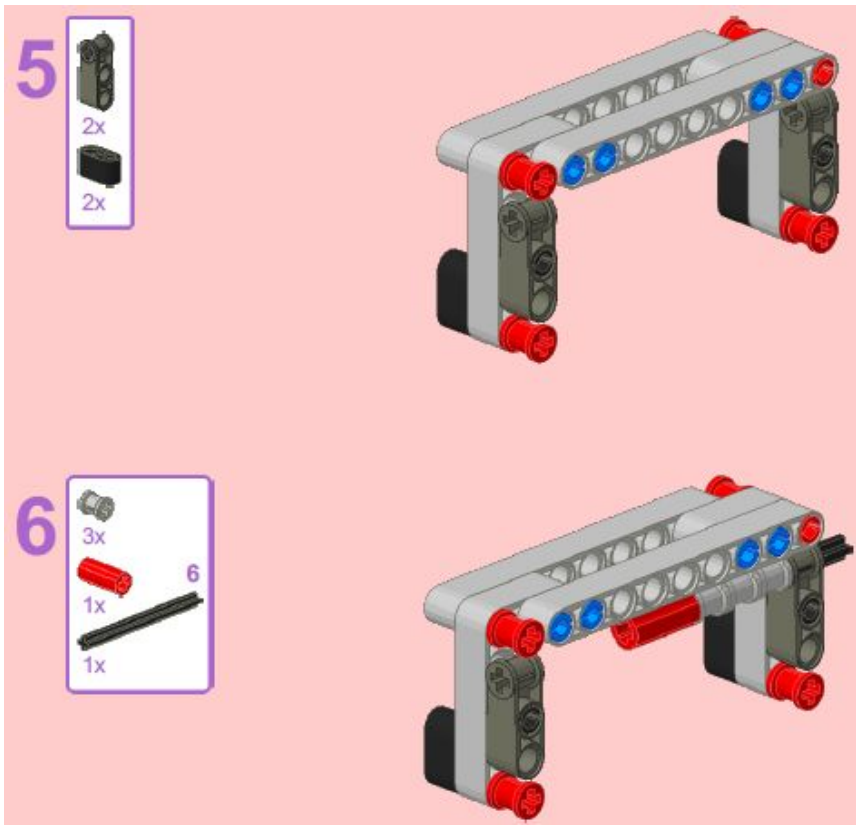

8
15
6x
2x
2x

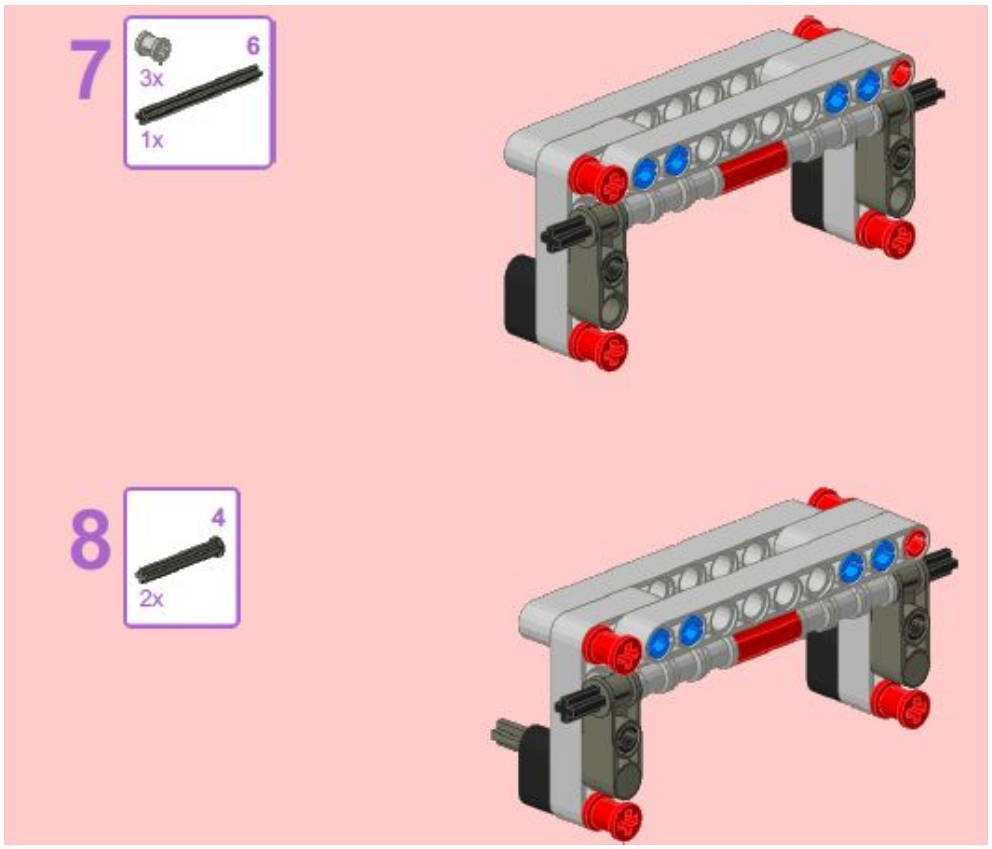

9
2x
2x

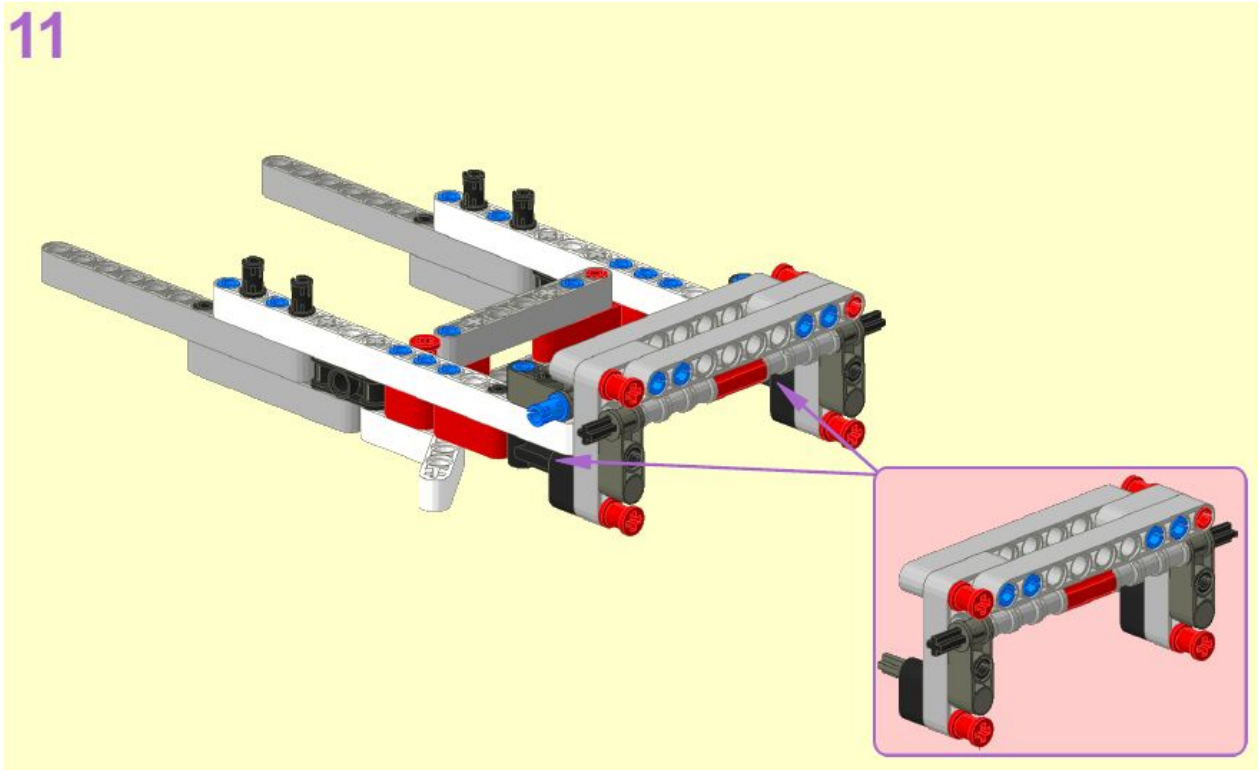

10
2x

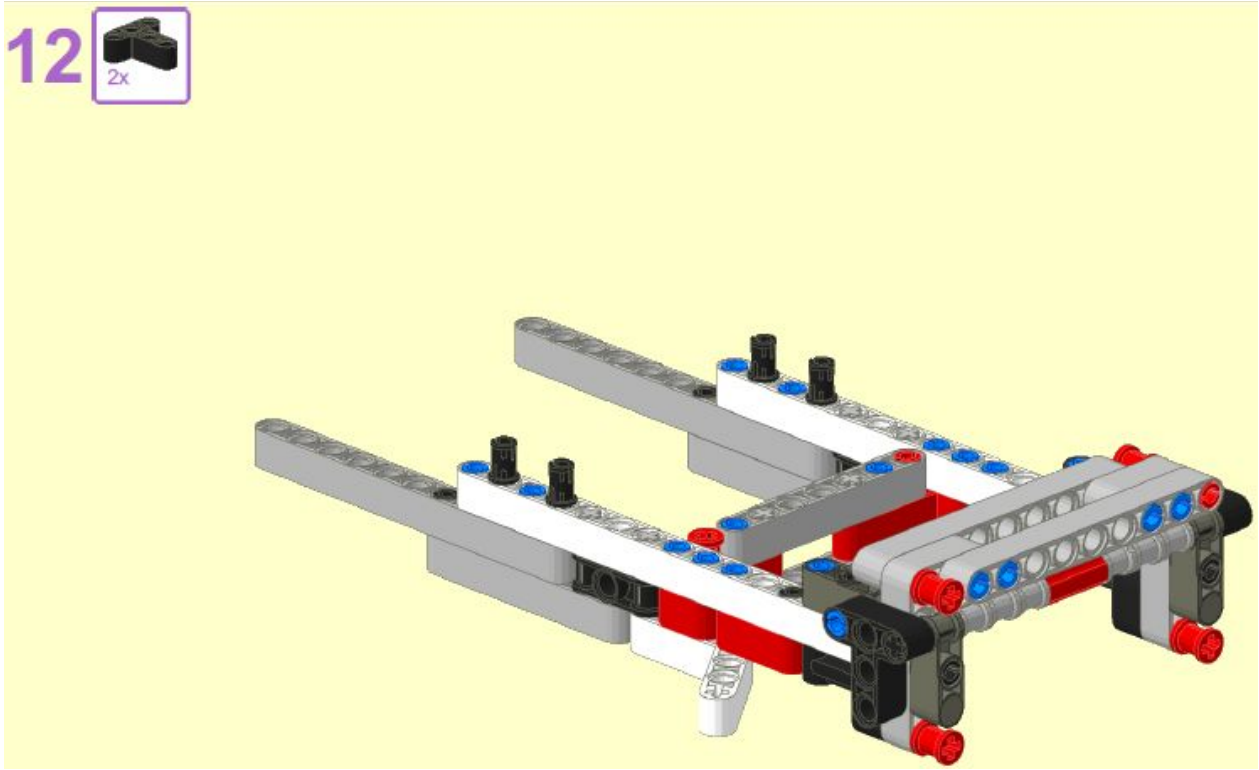

1
9
1x
4x
2
2x

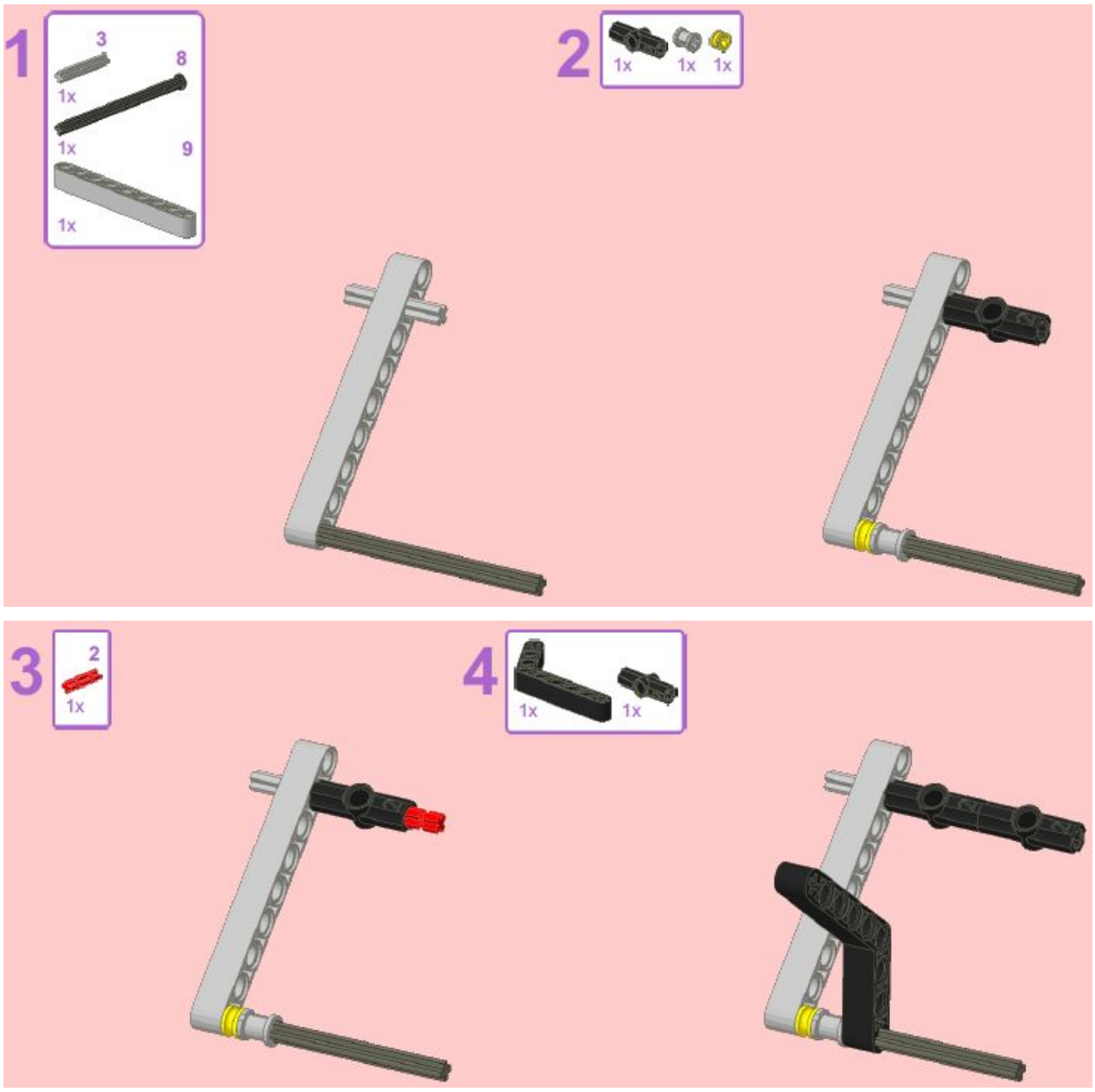

3
9
1x
4
2x
4x

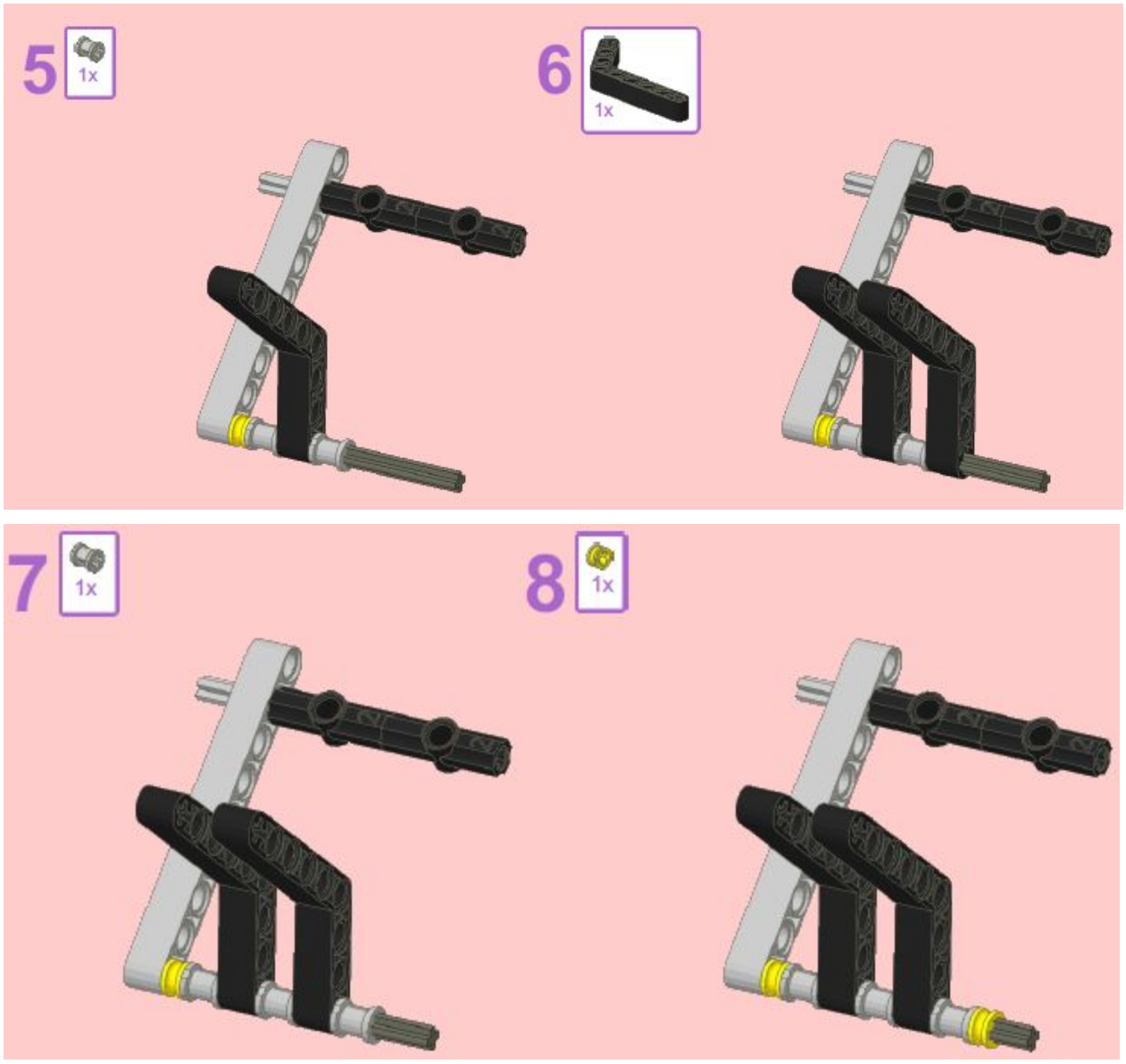

5
2x
2x
6
3x
6
1x
1x

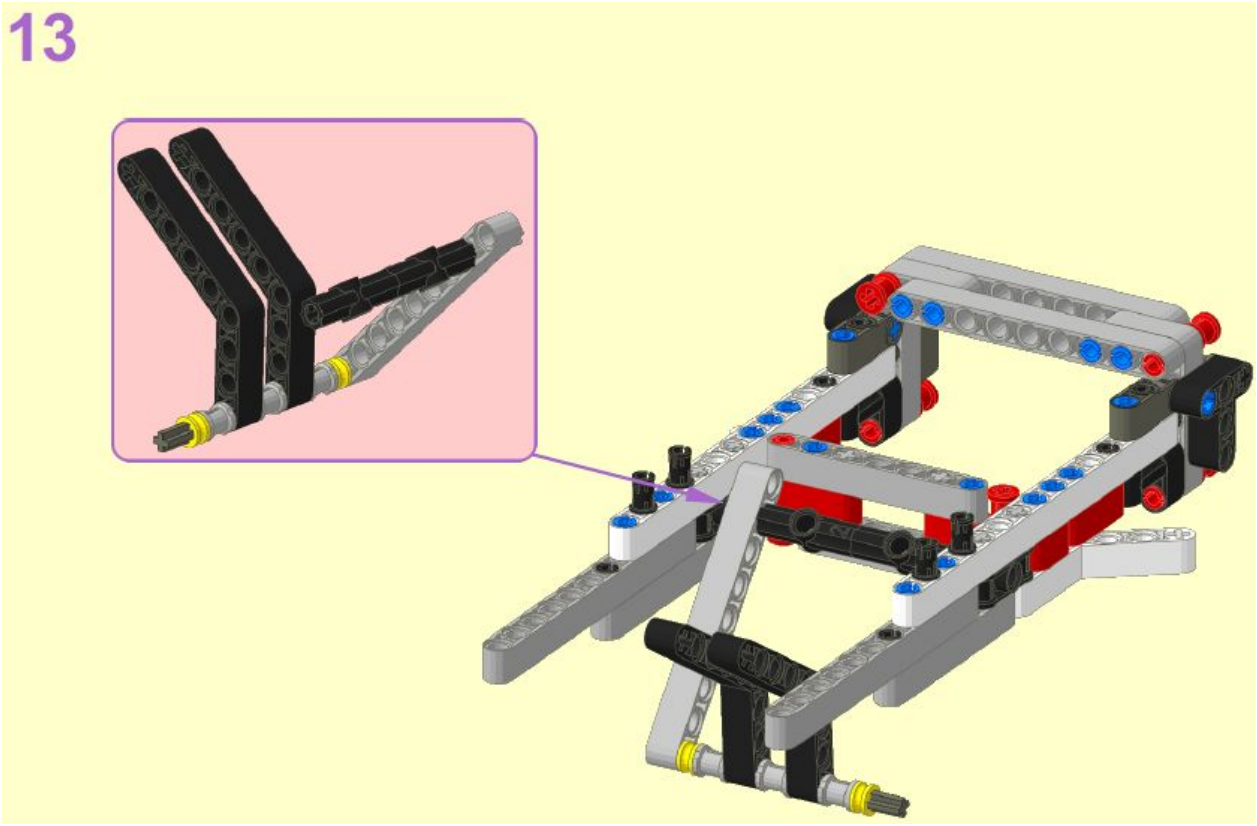

7
6
3x
1x
8
4
2x

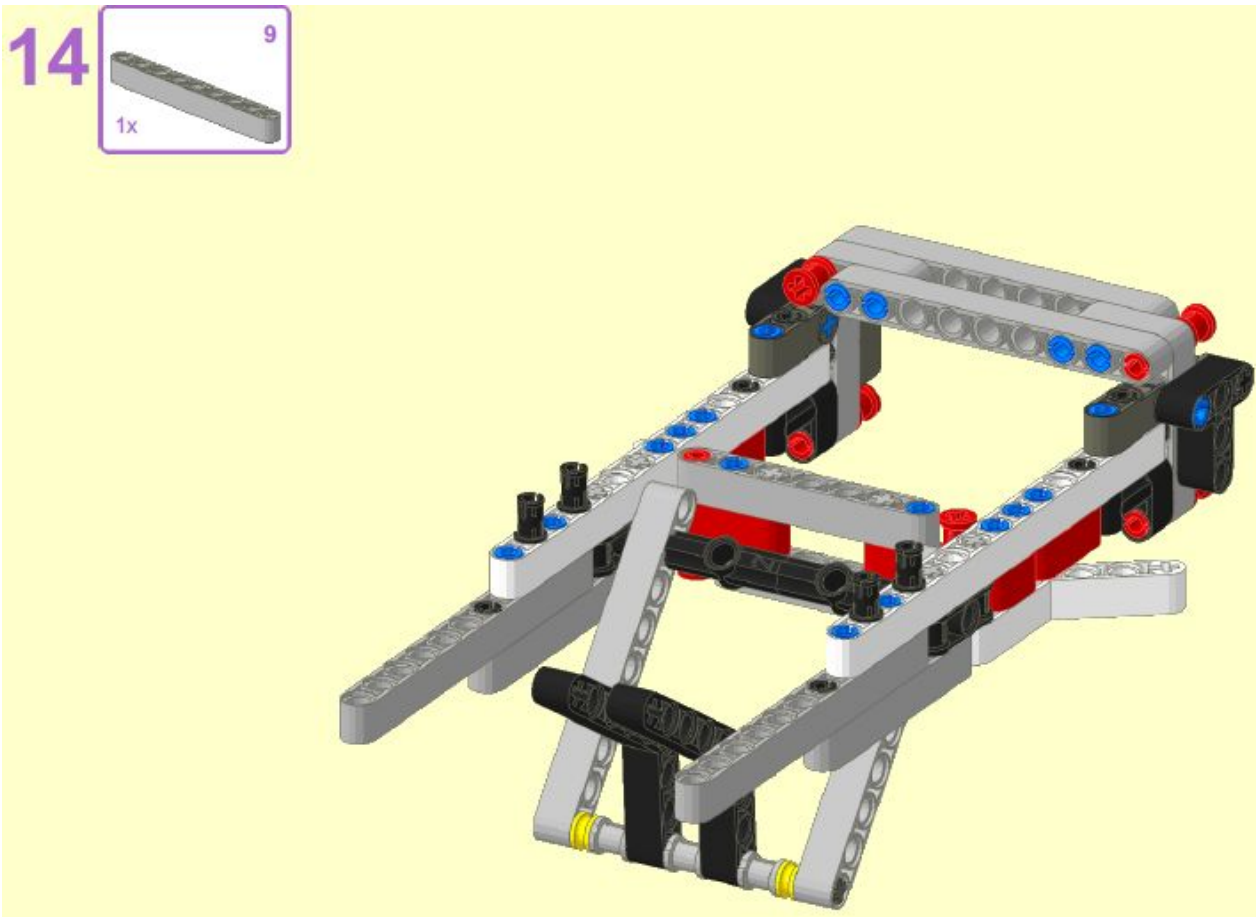

11

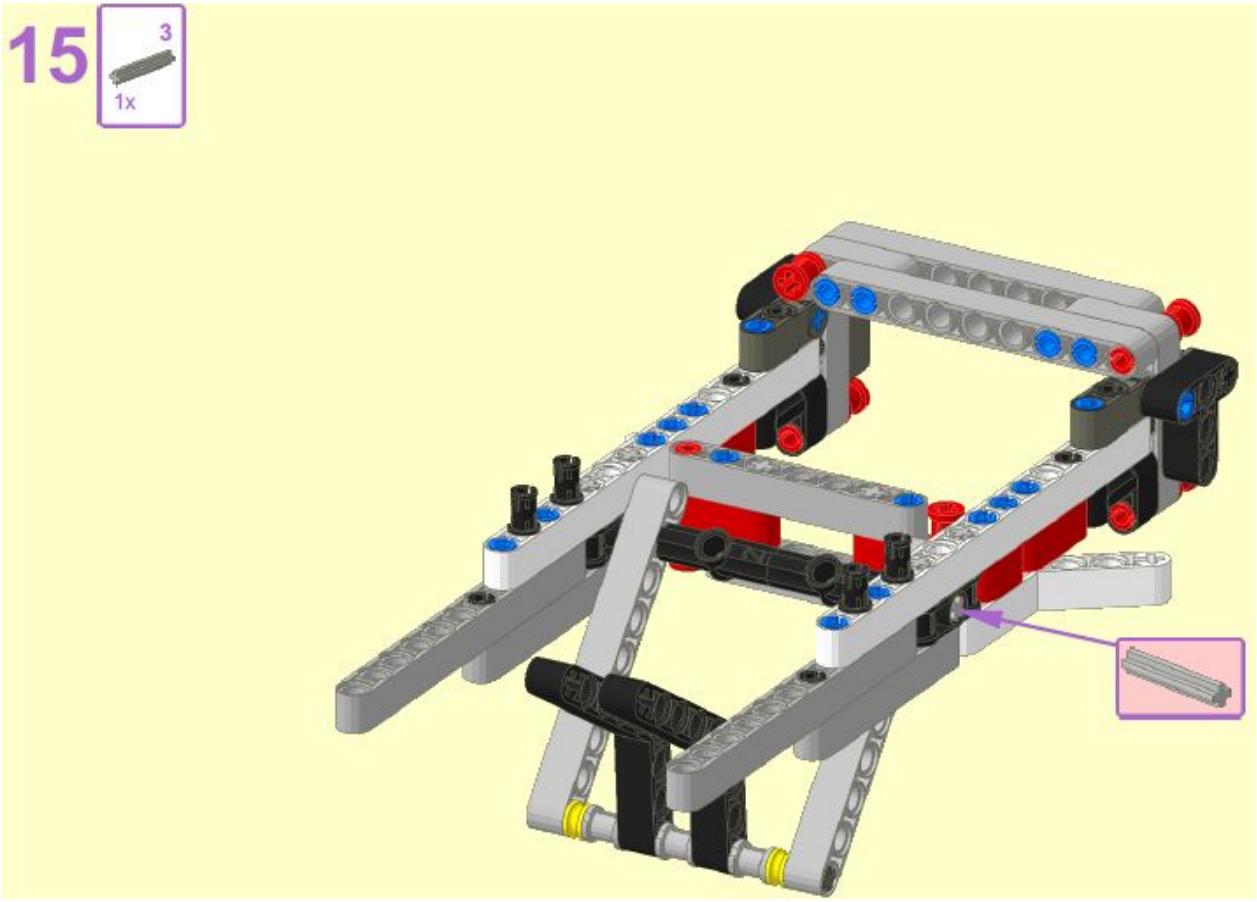

12
2x

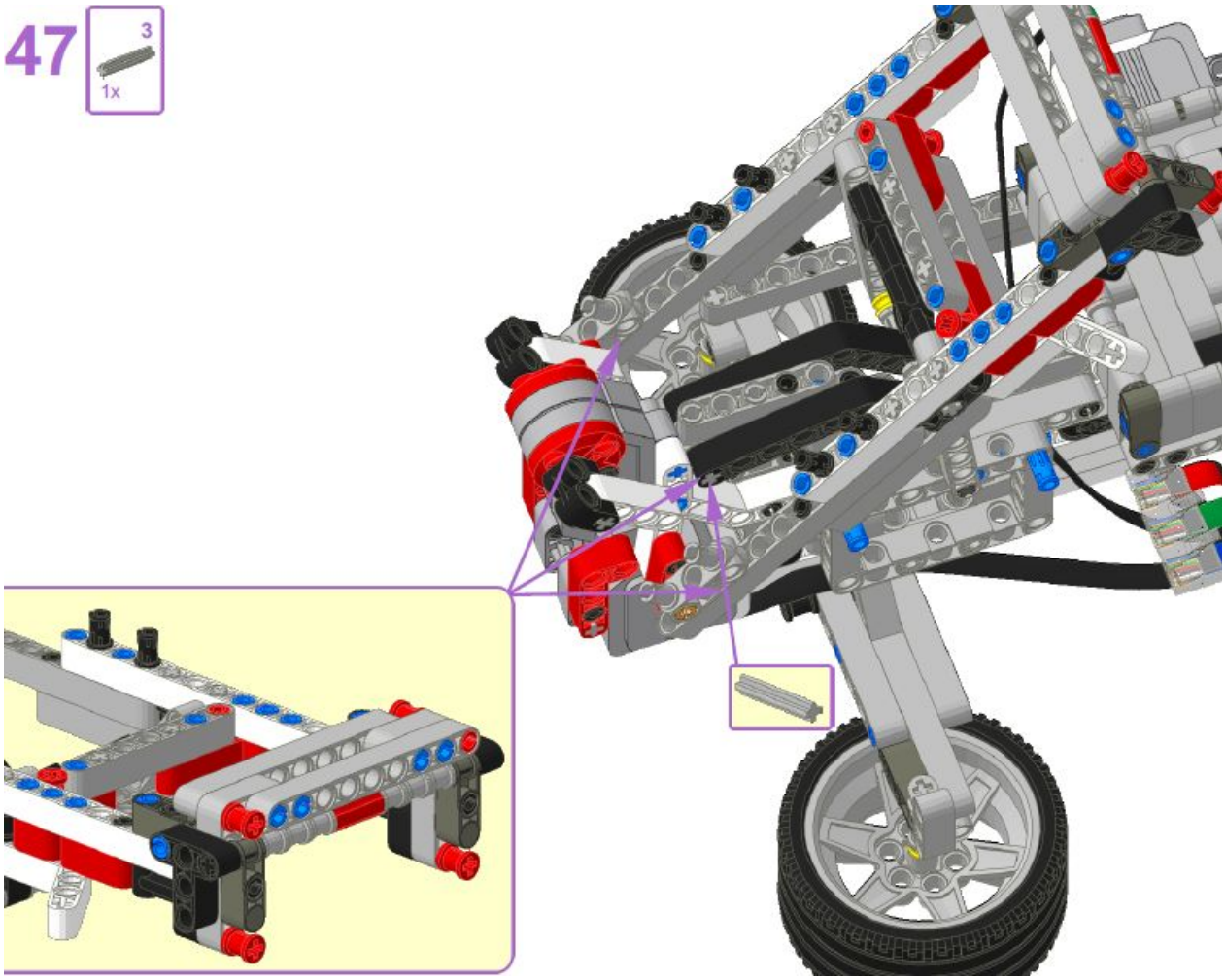

1
3
1x
8
1x
9
1x
2
1x
1x
1x
3
2
1x
4
1x
1x


5
1x
6
1x
7
1x
8
1x


13


14
9
1x


15
3
1x


47
3
1x

base: