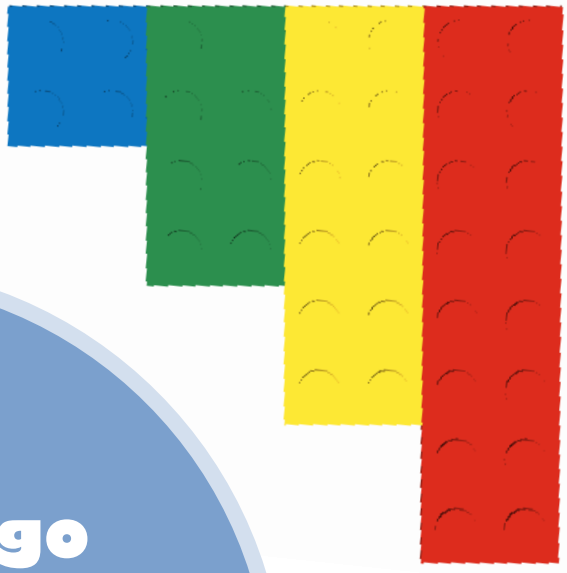
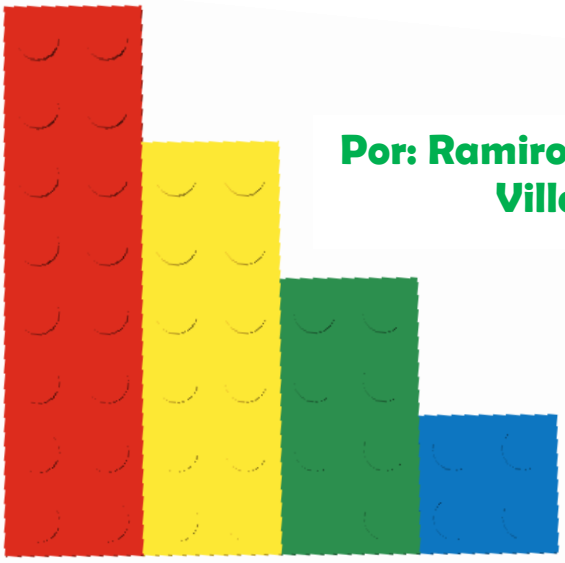
Guía de Ensamble y Programación



Robots Lego RCX y NXT programados en NQC y NXC



Por: Ramiro Robles Villanueva

Tabla de contenido

Prefacio

La presente guía se elaboro como parte de las acciones necesarias para llevar a cabo, diferentes prácticas en diversas áreas con robots Lego, programados en lenguaje C. Cabe señalar que estos robots han facilitado el aprendizaje a estudiantes de diferentes grados de estudio, y de esta manera se integran a los diversos instrumentos educativos que existen actualmente, posicionándose rápidamente como una de las herramientas más importantes en el ámbito educativo y tomando uno de los primeros lugares en el gusto de las personas que los han manipulado, debido a su fácil manejo en motores, servomotores y sensores, además de su divertida programación.

En la actualidad estos kits vienen con un software de programación visual, lamentablemente es una limitante ya que el código generado es muy grande y en ocasiones no se puede cargar en el Brick, haciendo uso de un lenguaje de programación como C o Java, se pueden realizar infinidad de tareas complejas con estos robots. NQC escrito por Dave Baum, es un compilador de C con el que se programa el robot con un lenguaje de alto nivel y de una manera muy simple.

Se recomienda bajar los manuales de programación de las siguientes direcciones:

http://bricxcc.sourceforge.net/nbc/

http://www.euskalnet.net/kolaskoaga/programz/nqc_c.htm

Aquí se muestran tres construcciones con su respectiva programación, guiando al educando paso a paso, para que posteriormente el construya y programe sus propias creaciones.

Léame antes de iniciar

El software y material necesario para utilizar esta guía se menciona en el siguiente párrafo:

- Kit Lego Mindstorms NXT.
- Kit Lego Mindstorms RCX con firmware actualizado a 2.0.
- Algunas piezas adicionales que se muestran en el apartado de la maquina de Turing.
- Preferentemente, los manuales de programación *NXC Guide* de *John Hansen* y *Programming Lego Robots using NQC* de *Mark Overmars.*

Se recomienda comprar el libro de Lego Masterpieces de Giulio Ferrari que contiene construcciones con código.

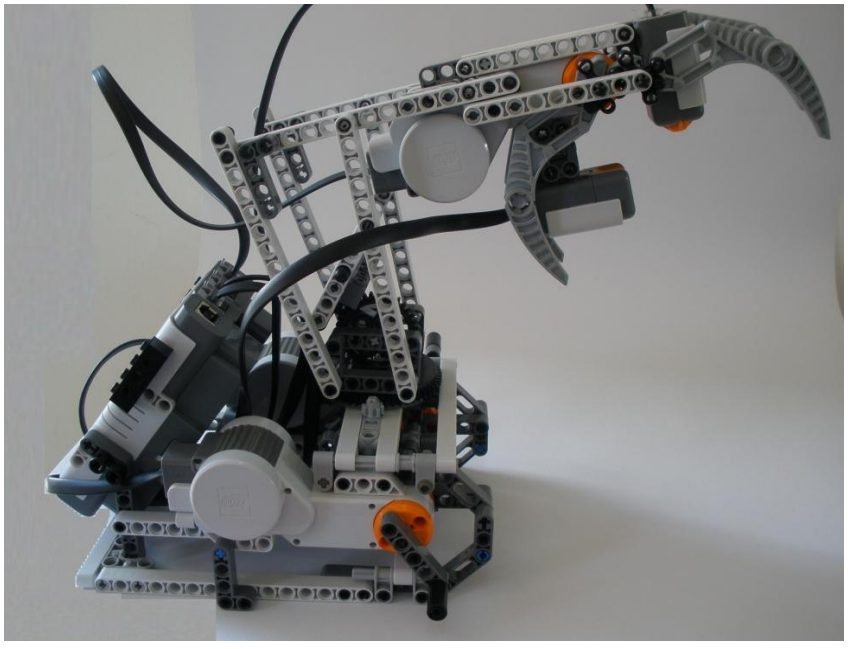
Precauciones y Sugerencias Generales

Nota Si se realiza la construcción sin observar la información, se puede dañar el equipo o no cumplir con el objetivo deseado.

Sugerencia Se aportan algunas sugerencias para llevarse a la práctica.

Robot 1

RoboArm T-56



Introducción

Los robots son imprescindibles en la actualidad. El uso de ellos en la industria ha beneficiado notablemente al humano. Cargar, cortar, soldar, pintar, perforar, son muchas de las tareas que desempeñan en el ramo automotriz, minero, textil, refresquero, etc. Además de facilitar tareas, brinda la seguridad necesaria a la integridad física del humano.

Uno de los robots más populares en la industria, son los que simulan brazos. Con el principal objetivo de brindar seguridad a las personas, además de lograr tareas donde se involucra como actor principal la precisión.

En este documento se muestra un brazo robot original de Lego Mindstorms, el RoboArm T-56. Un mecanismo con tres ejes de movimiento. Existen diversos modelos de brazos robóticos como por ejemplo el CyberArm IV, construido con su antecesor el brick RCX. Se eligió esta construcción debido a su diseño, ya que cuando es demasiado el peso en la mano o el propio simulador del brazo puede ocasionar algún daño a los servomotores. Además que en este documento se pretende hacer uso del compilador NXC. Y no propiamente resaltar nuevos diseños en construcciones.

Historia de los robots en la industria

El inicio de la robótica actual puede fijarse en la industria textil del siglo XVIII, cuando Joseph Jacquard inventa en 1801 una máquina textil programable mediante tarjetas perforadas. La revolución industrial impulsó el desarrollo de estos agentes mecánicos, entre los cuales se destacaron el torno mecánico motorizado de Babbitt (1892) y el

mecanismo programable para pintar con spray de Pollard y Roselund (1939). Además de esto durante los siglos XVII y XVIII en Europa fueron construidos muñecos mecánicos muy ingeniosos que tenían algunas características de robots. Jacques de Vauncansos construyó varios músicos de tamaño humano a mediados del siglo XVIII. Esencialmente se trataba de robots mecánicos diseñados para un propósito específico: la diversión. En 1805, Henri Maillardert construyó una muñeca mecánica que era capaz de hacer dibujos. Una serie de levas se utilizaban como ' el programa ' para el dispositivo en el proceso de escribir y dibujar. Estas creaciones mecánicas de forma humana deben considerarse como inversiones aisladas que reflejan el genio de hombres que se anticiparon a su época.

La definición del robot industrial, como una maquina que puede efectuar un número diverso de trabajos, automáticamente, mediante la programación previa, no es valida, por que existen bastantes maquinas de control numérico que cumplen esos requisitos. Una peculiaridad de los robots es su estructura de brazo mecánico y otra su adaptabilidad a diferentes aprehensores o herramientas. Otra característica especifica del robot, es la posibilidad de llevar a cabo trabajos completamente diferentes e, incluso, tomar decisiones según la información procedente del mundo exterior, mediante el adecuado programa operativo en su sistema informático.

Se pueden distinguir cinco fases relevantes en el desarrollo de la Robótica Industrial:

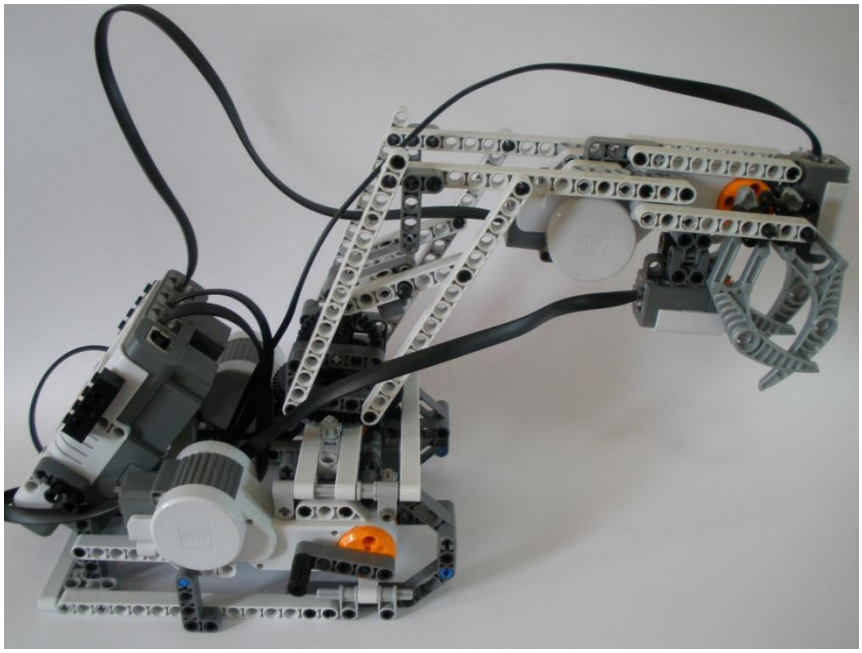
1. El laboratorio ARGONNE diseña, en 1950, manipuladores amo-esclavo para manejar material radioactivo.
2. Unimation, fundada en 1958 por Engelberger y hoy absorbida por

3. Whestinghouse, realiza los primeros proyectos de robots a principios de la década de los sesentas de nuestro siglo, instalando el primero en 1961 y posteriormente, en 1967, un conjunto de ellos en una factoría de general motors. Tres años después, se inicia la implantación de los robots en Europa, especialmente en el área de fabricación de automóviles. Japón comienza a implementar esta tecnología hasta 1968.
4. Los laboratorios de la Universidad de Stanford y del MIT acometen, en 1970, la tarea de controlar un robot mediante computador.
5. En el año de 1975, la aplicación del microprocesador, transforma la imagen y las características del robot, hasta entonces grande y costoso.

A partir de 1980, el fuerte impulso en la investigación, por parte de las empresas fabricantes de robots, otros auxiliares y diversos departamentos de Universidades de todo el mundo, sobre la informática aplicada y la experimentación de los sensores, cada vez mas perfeccionados, potencian la configuración del robot inteligente capaz de adaptarse al ambiente y tomar decisiones en tiempo real, adecuarlas para cada situación.

RoboArm T-56 Diseño y Construcción

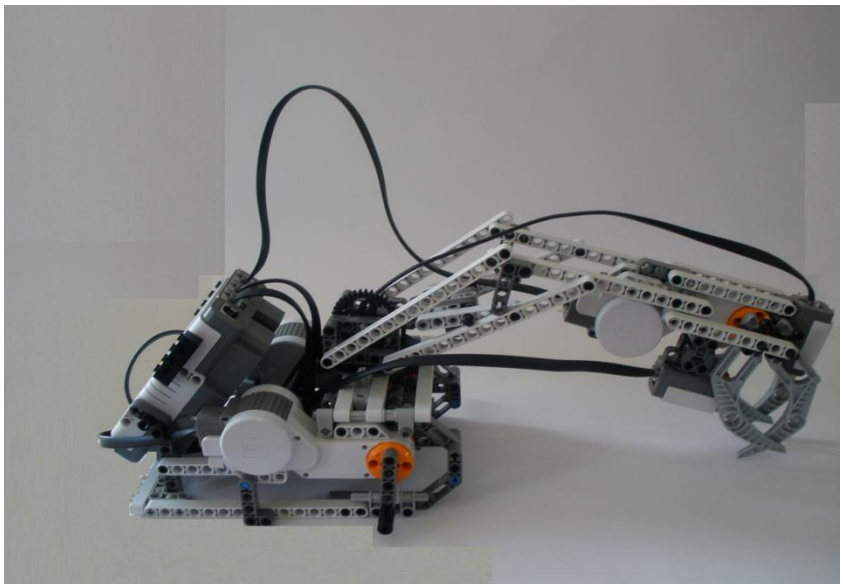
A continuación en este manual solo se muestra el robot que fue programado, para guiarse en la construcción de este, solo necesitan el software del Labview de Lego Mindstorms, en el cual viene cuatro diseños diferentes simulando vehículos, maquinas, animales y humanoides. Como se comento en los párrafos anteriores se hizo uso de esta construcción ya que no se necesitan piezas adicionales para su construcción, se puede elaborar diversas construcciones pero incluso algunas de ellas requieren sistemas neumáticos de Lego. Son necesarios los tres servomotores, sensor de luz, sensor de tacto y el brick NXT. Además de diversas piezas Lego que vienen en el kit de Lego Mindstorms.



Vista del RoboArm T-56 de costado

Brick del RoboArm T-56

Sensor de tacto y de luz



Brazo extendido del RoboArm T-56

Programando el RoboArm T-56 con NXC

A continuación se muestra la tarea que ejecuta el robot con las bolas de colores que vienen incluidas en el kit de Lego Mindstorms NXT.

Nota:

Antes de iniciar recuerde que es necesario tener instalado el software del BricxCC Command Center, haber inicializado con el puerto USB y el Brick NXT.

El objetivo de este brazo es tomar una bola de color azul o roja, dependiendo del color las coloca en el lado derecho las rojas y el izquierdo las azules. Un servomotor se ocupa de abrir y cerrar la mano. Otro se encarga de girar el brazo a la izquierda y derecha. El ultimo de bajar y subir la mano. El sensor de tacto valida que realmente se ha tomado la bola. Y el de luz indica su color.

Los movimientos del brazo son los siguientes:

- La mano se encuentra originalmente abajo.
- Al correr el programa, la mano se abre.
- El brazo se mueve hacia arriba por ocho segundos levantando la mano abierta.
- Hace un ligero movimiento hacia el lado derecho por tres segundos.
- Abre la mano y toma la bola roja o azul. Utilizando los sensores de tacto para validar que tomo un objeto y el de luz para indicar el color y el movimiento que tiene que realizar.
- Si es azul, se mueve el brazo hacia la izquierda por nueve segundos.
- Si es roja se desplaza hacia la derecha.

- Una vez que ejecuta su movimiento según sea el caso del color de la bola. Baja la mano por ocho segundos y la abre, soltando la bola.

A continuación se muestra el código:

```
/*Codigo en NXC para RoboArm T-56
Selecciona objetos de color rojo o azul y los
Desplaza a la derecha o izquierda según sea el caso
Creado por: Ramiro Robles Villanueva*/

#define ma OUT_A  // Motor de mano
#define mb OUT_B  // Motor sube y baja mano
#define mc OUT_C  // Motor que mueve a la derecha o izquierda
#define BUTTON IN_1
#define LIGHT IN_2
#define until(c) while(!(c))

task main()
{
short light_value; //Inicializo variable de tipo corto
/*Apago motores*/
Off(ma);
Off(mb);
```

```
Off(mc);
/*Inicializo sensores en su respectivo puerto*/
SetSensorTouch(IN_1); //Sensor de tacto puerto uno

SetSensorLight(IN_2); //Sensor de luz puerto dos
while (1)  //Ciclo que hace que se ejecute el programa infinitamente
{
OnRev(ma,100); Wait(500);     //Abre mano
OnRev(mb,100); Wait(7000); Off(mb); //Levanta mano
OnRev(mc,100); Wait(4000); Off(mc); //Mueve hacia la derecha
OnFwd(ma,100); Wait(500);    //Cierra mano
until (IN_1 == 0); //Valida el sensor de tacto que realmente hay un objeto
  {
    light_value =  Sensor(IN_2); //Captura el sensor de luz el reflejo del objeto
    NumOut(20, IN_2, light_value); // El valor recibido se despliega en el display
    if (light_value>45) //Si la luz es mayor que 45 es roja y se va de lado derecho
    {
     OnRev(mc,100); Wait(4000); Off(mc); //Mueve hacia la derecha
     OnFwd(mb,100); Wait(8000); Off(mb); //Baja mano
     OnRev(ma,100); Wait(500); //Abre mano
     OnRev(mb,100); Wait(7000); Off(mb); // Sube mano

     OnFwd(mc,100); Wait(6000); Off(mc); // Mueve brazo a la izquierda
```

```
    OnFwd(ma,100); Wait(500);     //Cierra mano
    OnFwd(mb,100); Wait(7000); Off(mb); //Baja mano
    }
  else  //Sino, es azul y se dirige hacia la izquierda
    {
     OnFwd(mc,100); Wait(9000); Off(mc); //Mueve hacia la izquierda
     OnFwd(mb,100); Wait(7000); Off(mb); //Baja mano
     OnRev(ma,100); Wait(500); //Abre mano
     OnRev(mb,100); Wait(7000); Off(mb); // Sube mano
     OnRev(mc,100); Wait(7000); Off(mc); //Mueve brazo hacia la derecha
     OnFwd(ma,100); Wait(500);     //Cierra mano
     OnFwd(mb,100); Wait(7000); Off(mb); //Baja mano
     }
   }
 }
}
```

Sugerencias:

Con esta construcción y el código presentado se pueden realizar diversas prácticas como por ejemplo:

- **Hacer que el brazo se desplace de izquierda a derecha en busca de un objeto, una vez localizado, tomarlo y moverlo a una posición especifica.**

- **Realizar movimientos de precisión utilizando los dos servomotores que mueven el brazo de izquierda a derecha y de arriba abajo.**

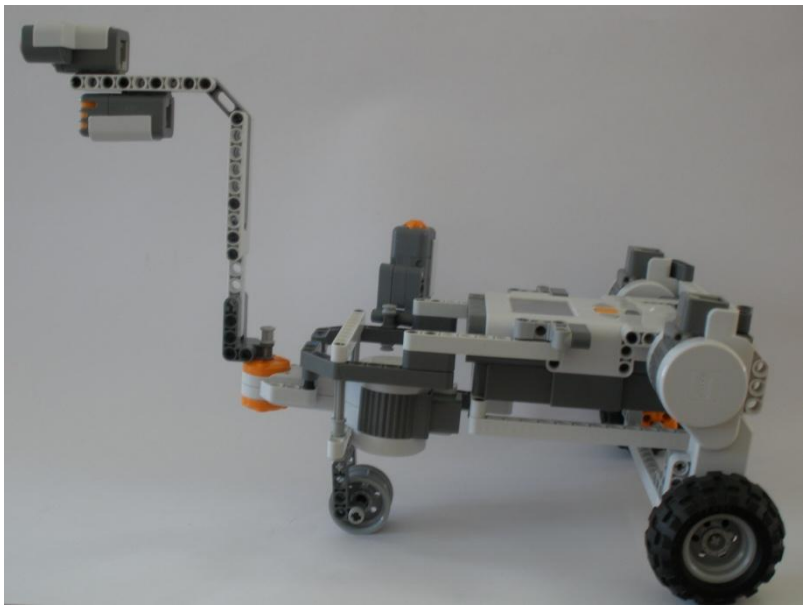
Resumen

Este es un ejercicio simple para conocer y practicar el código necesario para hacer funcionar los servomotores. Además de trabajar con los sensores de luz y tacto. Como se inicializar servomotores y sensores.

El objetivo principal es adentrarnos a la programación de NXC, ya que al basarse en un lenguaje de programación como C, aun tiene algunas palabras reservadas que no se manejan en el lenguaje común. Al mismo tiempo se pretende dar a conocer la precisión que se necesita para que un brazo ejecute acciones con una precisión absoluta. Como podemos observar aun se puede pulir algunos detalles en la programación para hacerlo mas preciso.

Al conocer el manejo básico de servomotores y sensores se pueden realizar diversas construcciones propias para un objetivo definido.

ChalanBot



Introducción

Facilitar las tareas de la personas es el objetivo esencial de un robot. ChalanBot es un prototipo orientado atender las necesidades de las personas que requieren herramienta al alcance de la mano. Imaginemos un mecánico, recibe un carro pero su herramienta se encuentra hasta el otro extremo del taller, el mecánico enciende un dispositivo que emite un sonido constante, Chalanbot se dirige hacia el. El potencial de ChalanBot es ilimitado ya que se puede aplicar en diversas áreas, carpintería, mecánica, electricidad, etc. Para contribuir a que ChalanBot desempeñe sus funciones de manera correcta se equipa con un sensor ultrasonico que evitan que pueda chocar y provocar algún daño al robot o al equipo que transporta. ChalanBot se convierte en uno de los primeros robots serviciales de la familia Lego.

Robots Serviciales

Existen diversas empresas que perfeccionan e innovan robots serviciales, Toyota es una de ellas que presentó uno de ellos, pensados para ayudar a personas con problemas de movilidad. El “Mobility Robot” o Mobiro es una moderna silla de ruedas, que puede desplazarse sobre superficies irregulares y subir pendientes de hasta 10 grados de inclinación, manteniendo siempre al ocupante en una posición estable.

Esto lo consigue gracias al diseño de sus ruedas, cada una de las cuales puede girar y ajustar su altura independientemente para adaptarse al terreno. Va equipado con sensores de proximidad que le permiten evitar obstáculos, y también sabe cómo volver a casa de forma automática después de un paseo, respondiendo a la voz de su dueño.

Pesa 150 kilos y mide 1 metro de altura. Con sus baterías cargadas, tiene una autonomía de 20 kilómetros, moviéndose a una velocidad constante de 6 Km/h. Aunque Toyota afirma que sirve perfectamente para transportar bultos y equipaje, lo primero que van a hacer es probarlo con ancianos y personas convalecientes en hospitales japoneses, a partir del año que viene.

Instituciones educativas tienen diversos proyectos de robots serviciales como ejemplo el Instituto Tecnológico de la Universidad de Georgia que creo el El-E. Donde se señala algo con un puntero láser y lo trae. Este brazo cibernético con ojos y ruedas, está pensado en realidad como asistente para problemas de movilidad reducida.

Primero se señala el objeto deseado con el puntero láser durante unos segundos. Cuando El-E lo recoge, se señala otro punto (nosotros mismos, por ejemplo) para que el robot lo transporte hasta nuestra ubicación. El robot consigue esto gracias, en primer lugar, a una cámara que escanea automáticamente todos sus alrededores en busca del láser.

Suena sencillo, pero no lo es. El-E debe ser capaz tanto de distinguir bien los bordes del objeto para agarrarlo, como a la persona que le da las ordenes para entregárselo. Para ello cuenta con otras dos cámaras colocadas en la misma posición que los ojos humanos, con las que consigue una perspectiva tridimensional y va equipada con software especial para distinguir texturas y rostros.

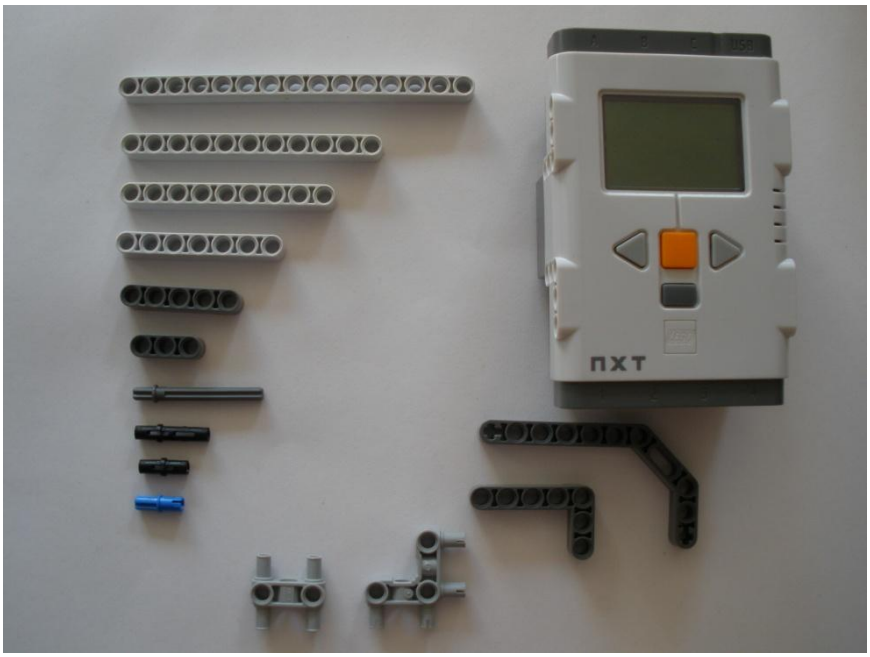
A esto hay que añadirle toda la maquinaria necesaria para que el robot se mueva, baje el brazo, agarre y lo alce. En definitiva, uno de esos robots serviciales orientados a facilitar las cosas tanto al paciente, que consigue más independencia, como a sus cuidadores,

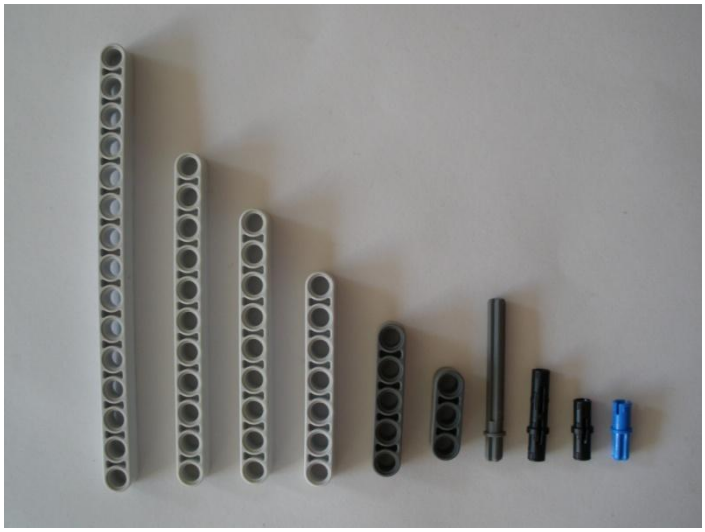
La idea es que, en un futuro próximo, el uso de estos robots se vaya generalizando y que los veamos poco a poco en cada vez más sitios.

ChalanBot Diseño y Construcción

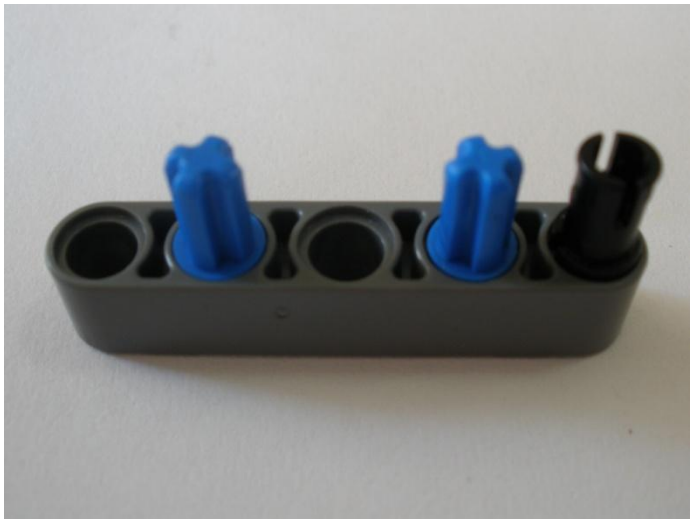
El diseño y construcción de ChalanBot es original, fue diseñado para desplazarse en lugares donde la superficie es firme. Cuenta con dos servomotores en la parte trasera que hacen virar al robot de izquierda a derecha o hacia delante. Un sensor ultrasónico que permite detectar objetos hacia donde se dirige y detenerse, antes que impacte con ellos. Lo que permite al robot dirigirse hacia el sonido, es un sensor de sonido. Para apagarlo al llegar a su destino, se utiliza un sensor de tacto. No fueron necesarias piezas adicionales, con el Kit de Lego Mindstorms de NXT se construyo el ChalanBot en su totalidad. A continuación se muestran los pasos para el ensamble:

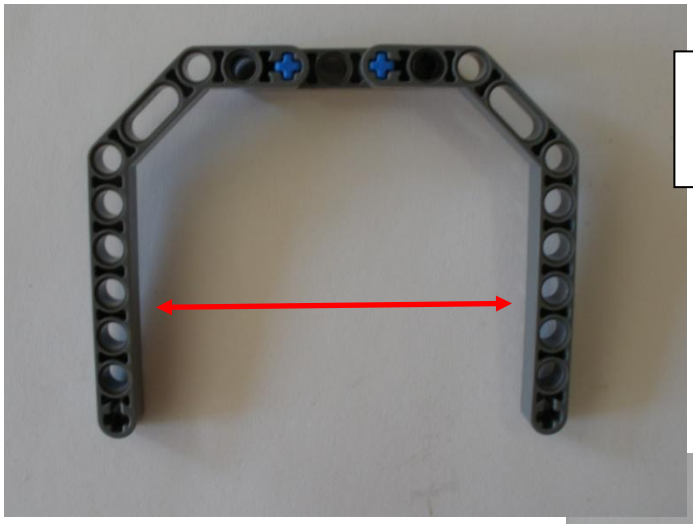
Paso 1: Brick



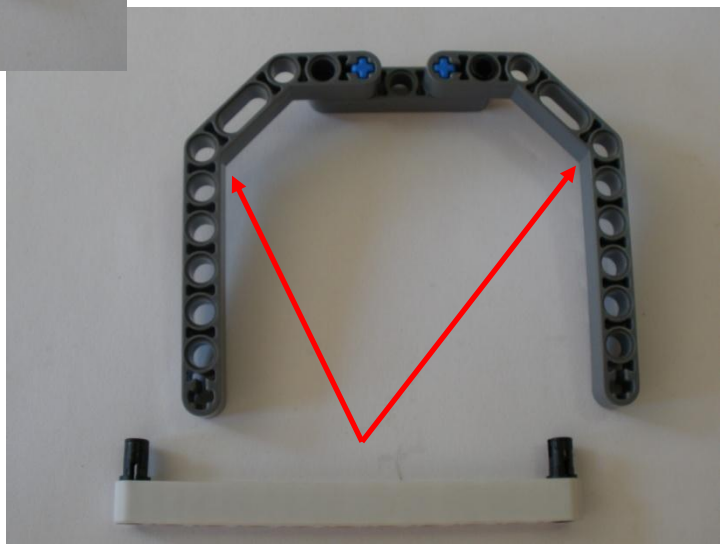


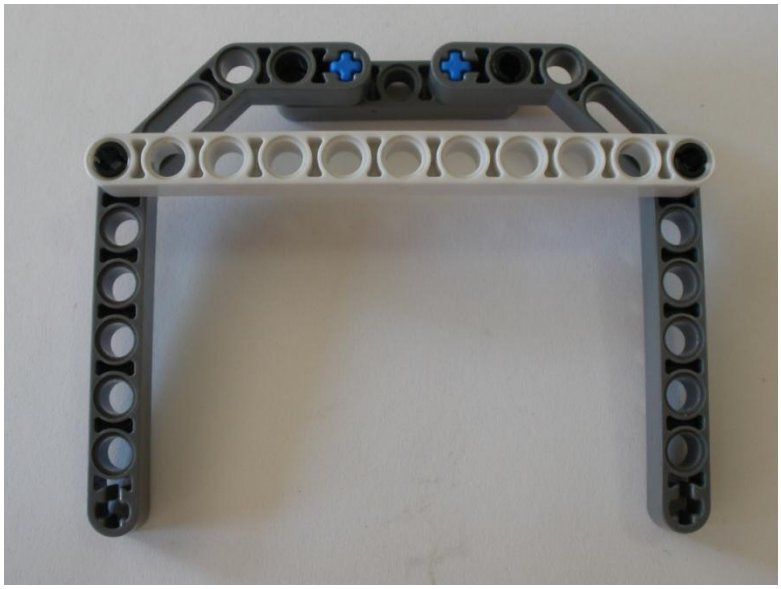

A
B
C
USB
NXT
1
2
3
4

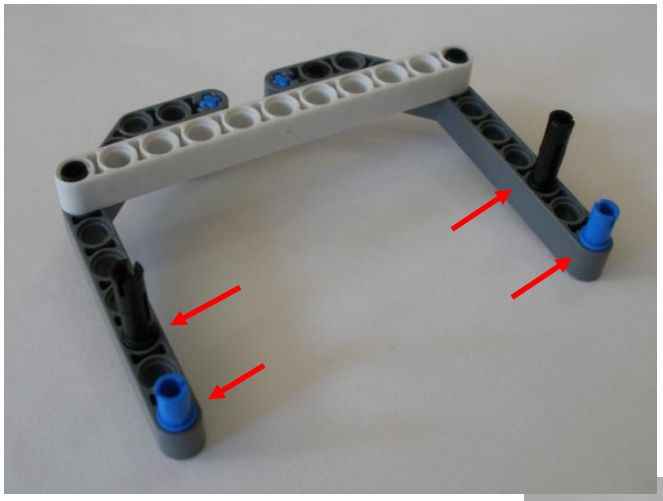


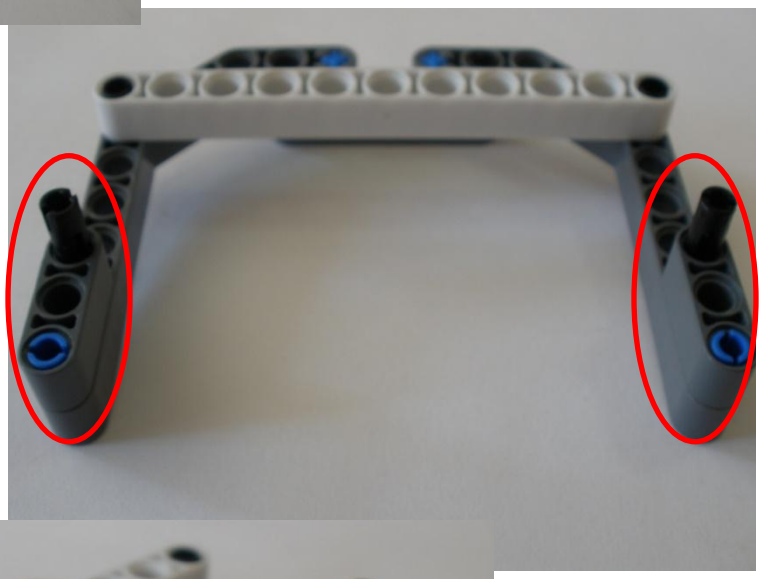


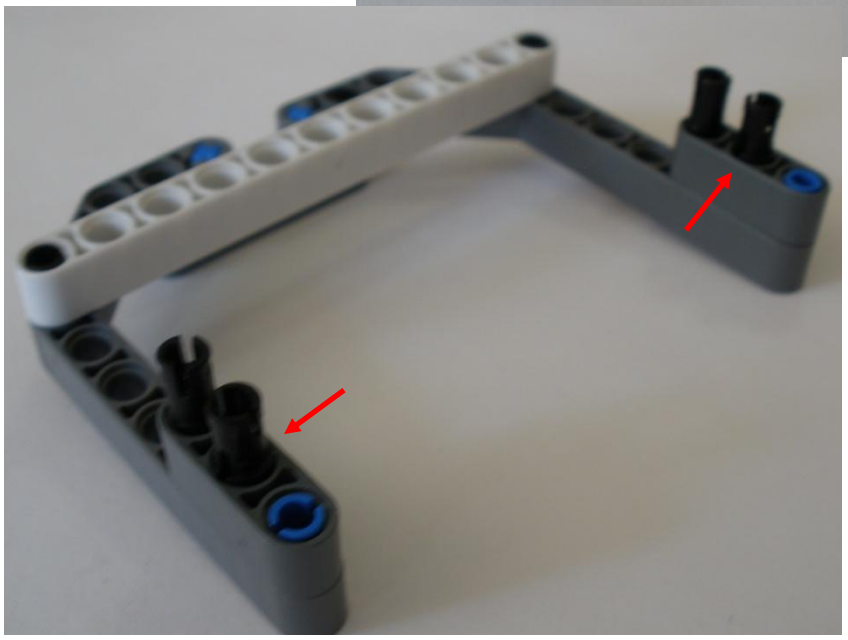
Ensamblamos estas dos piezas en la última pieza armada

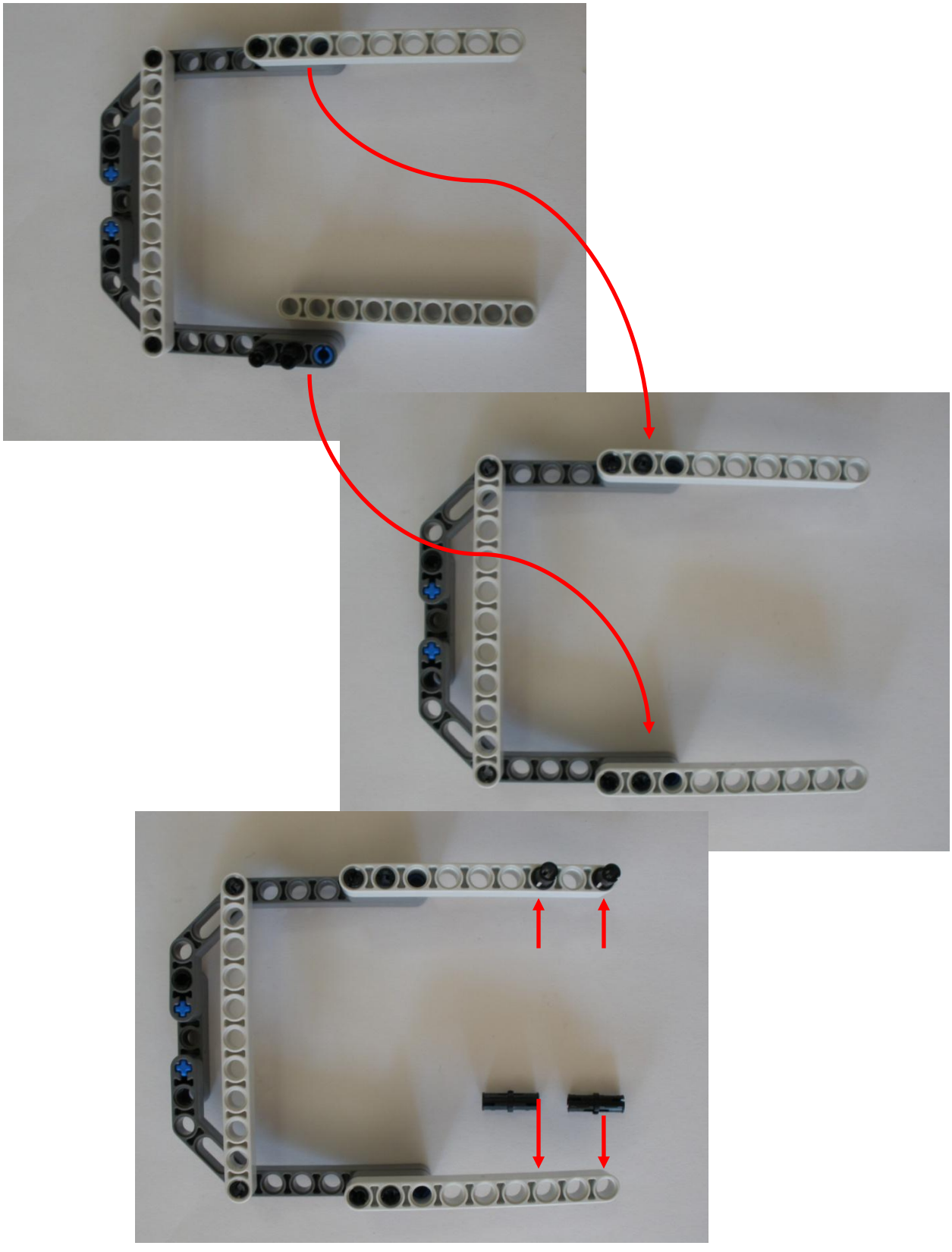


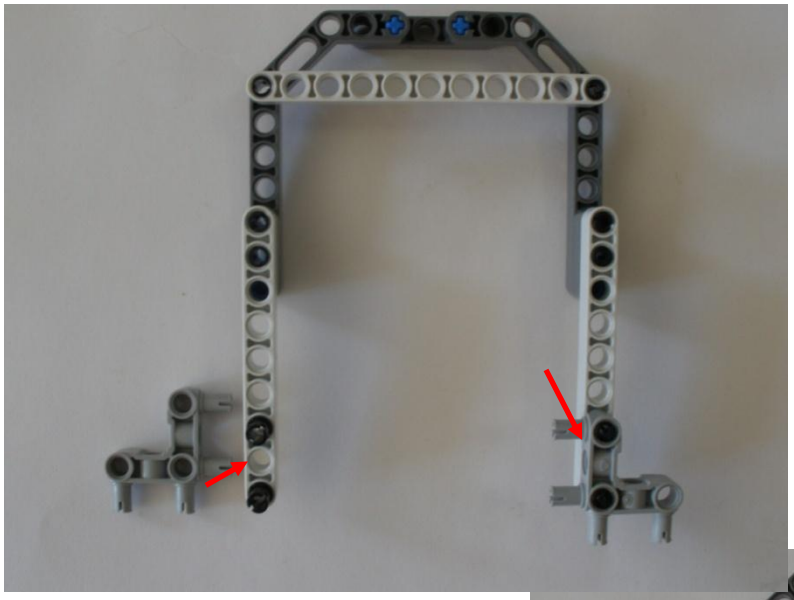


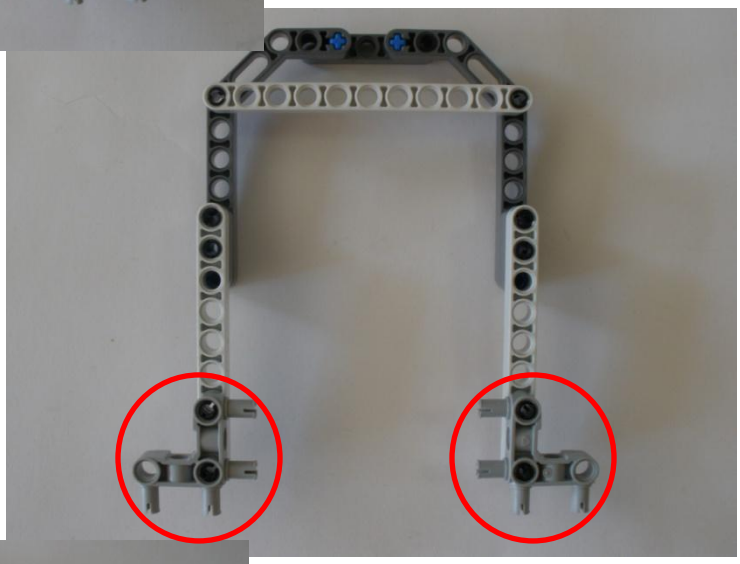


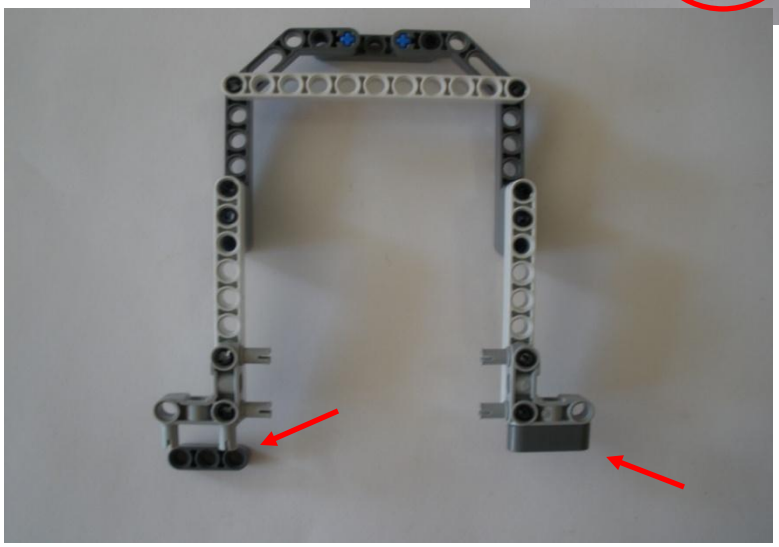


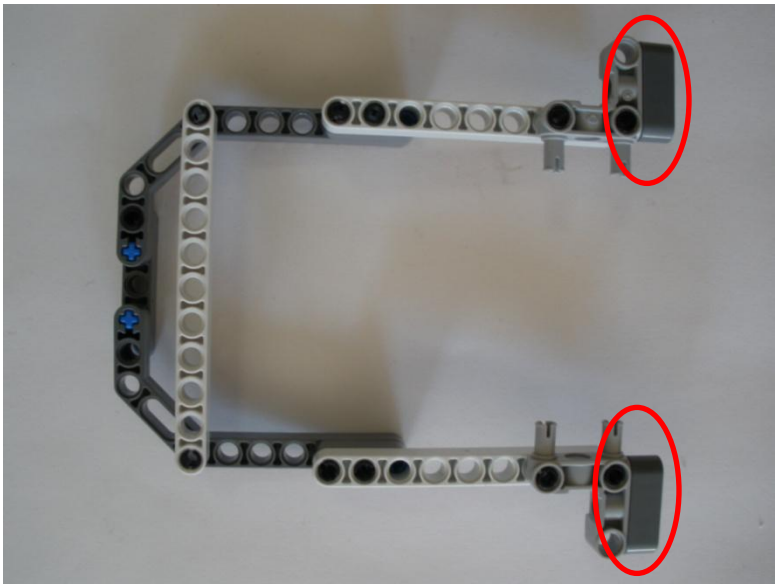


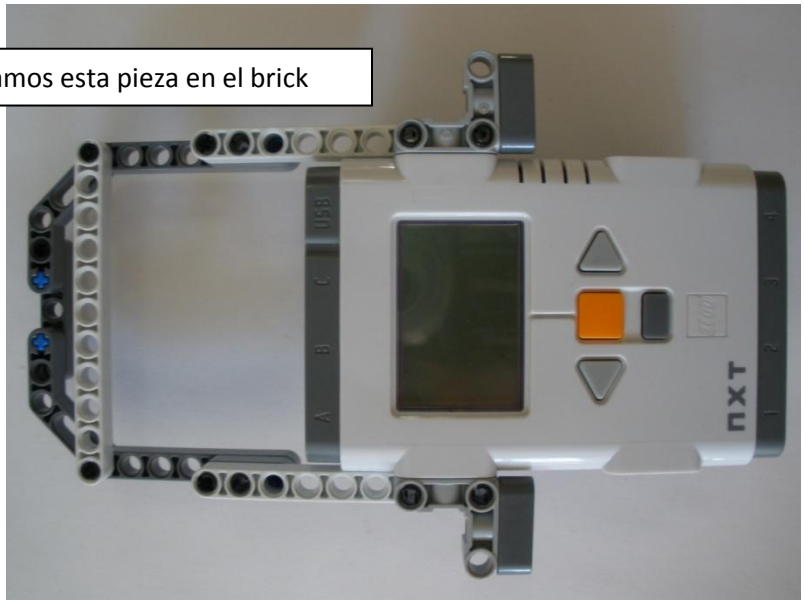




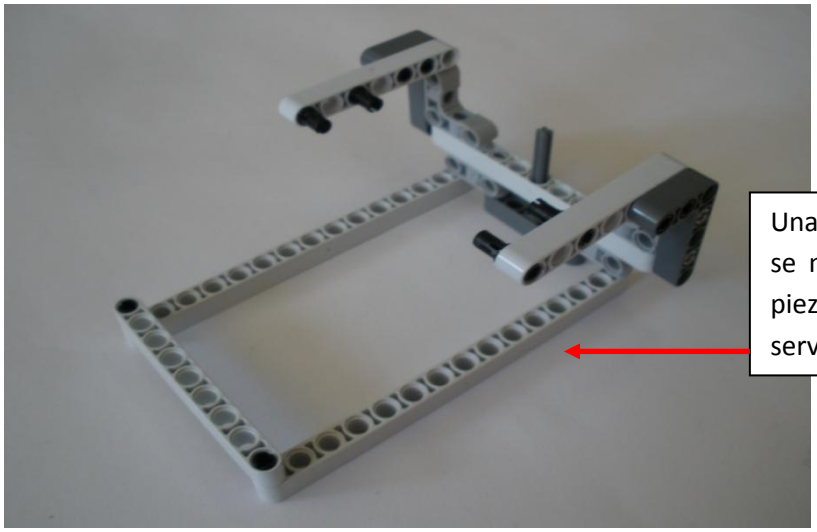




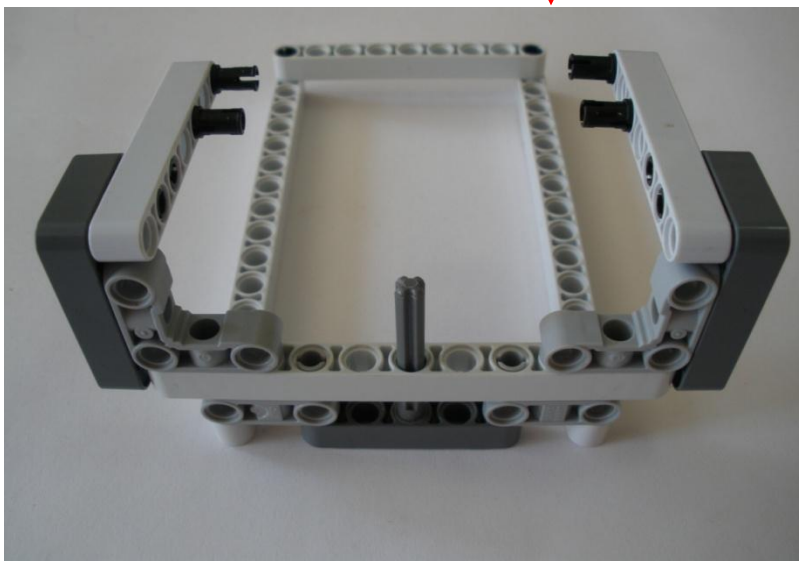


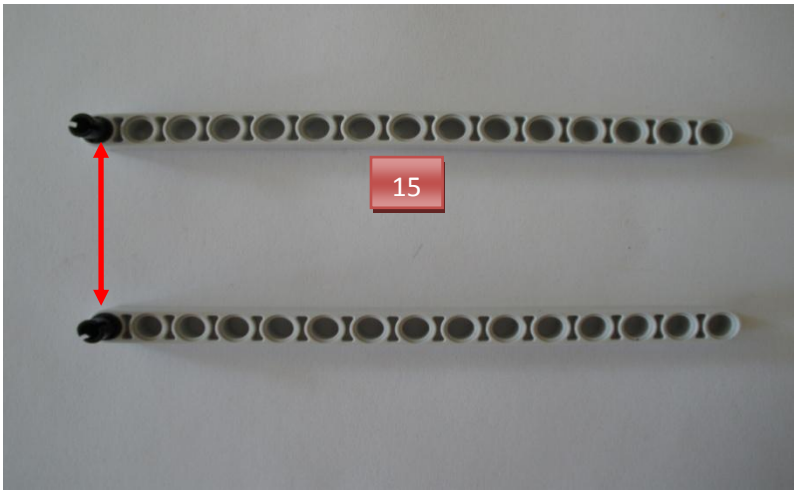


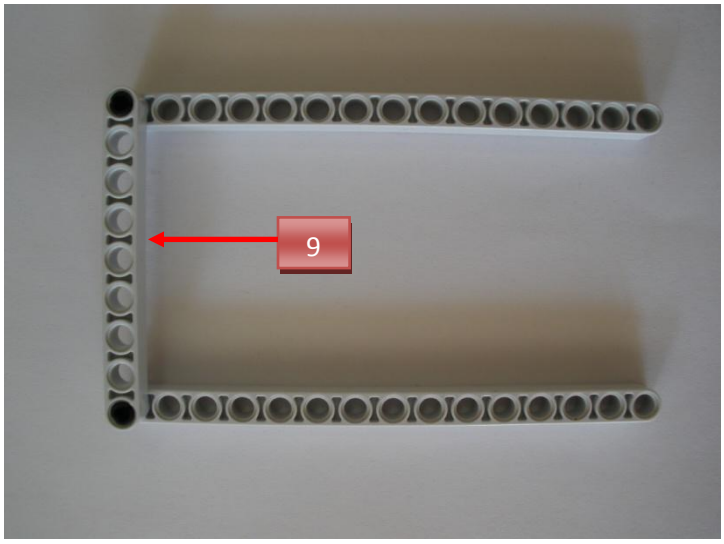
Así quedaría la parte delantera del brick. Donde posteriormente se implementa el cuello. A continuación se ensambla la parte trasera donde se montaran los servomotores.

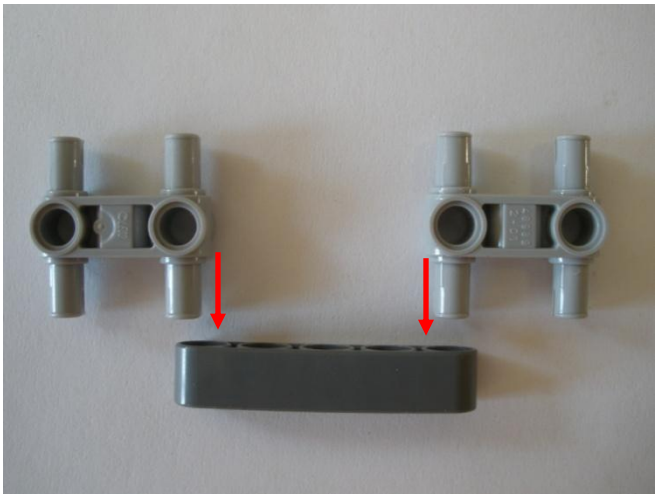


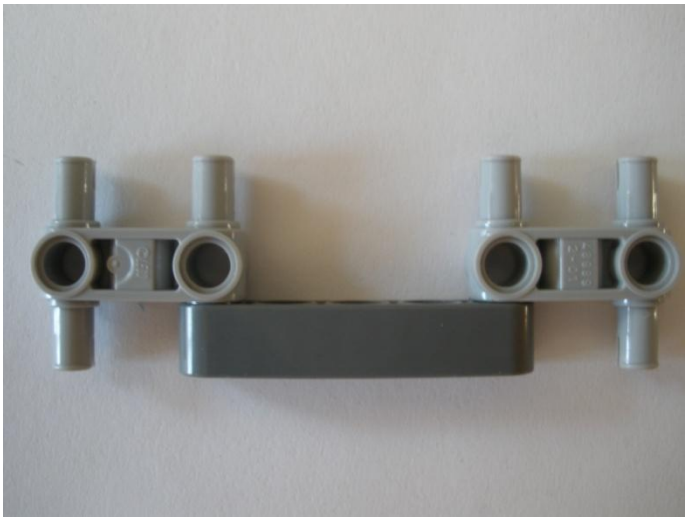
Una vez armada la pieza donde se monta el cuello, armamos la pieza que van a soportar los servomotores con las llantas.

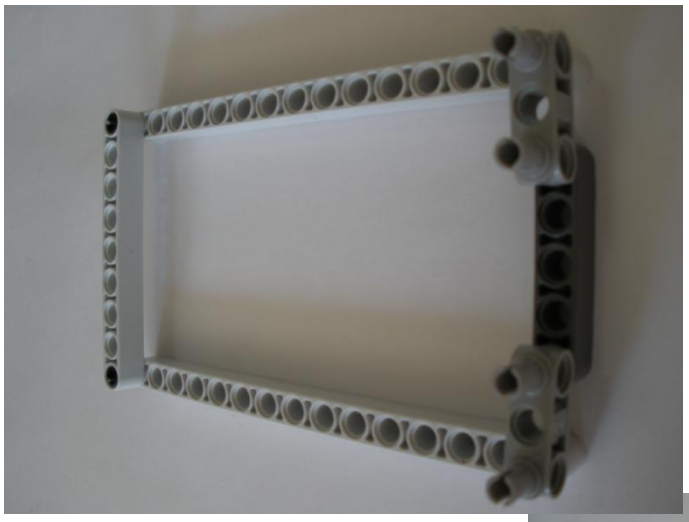


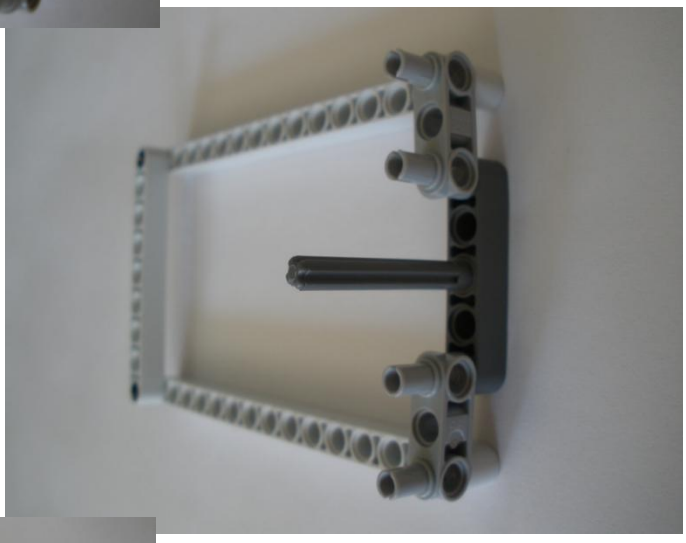


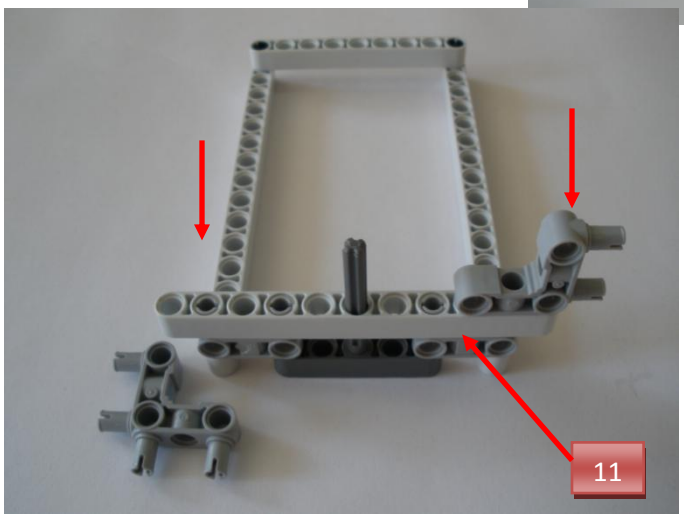

9

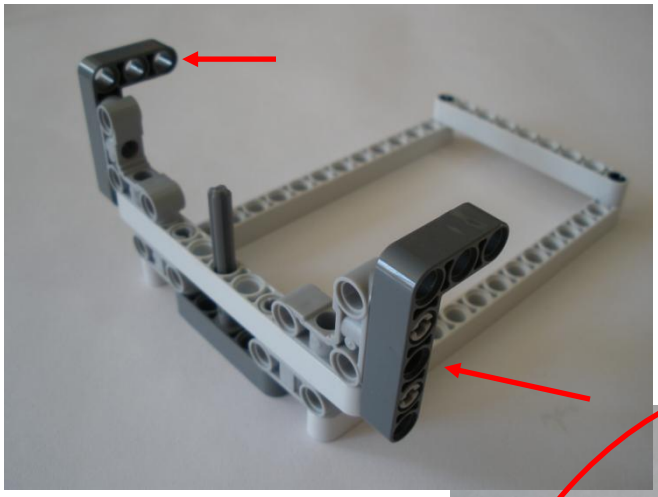


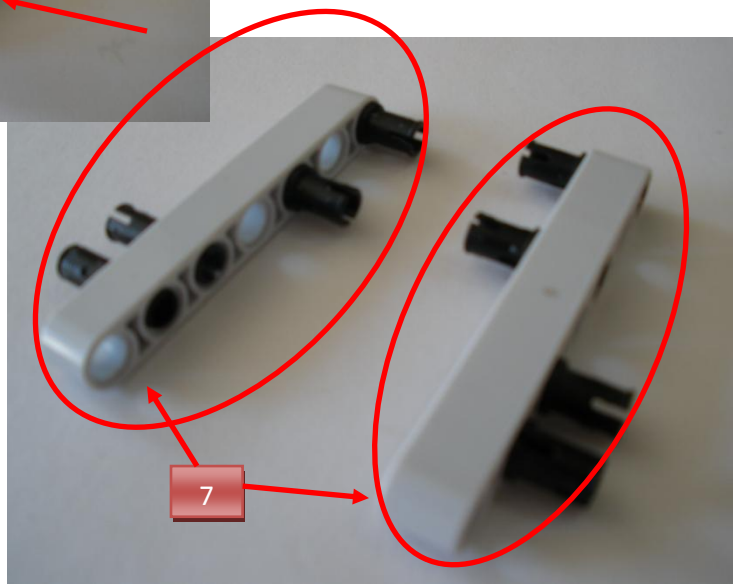


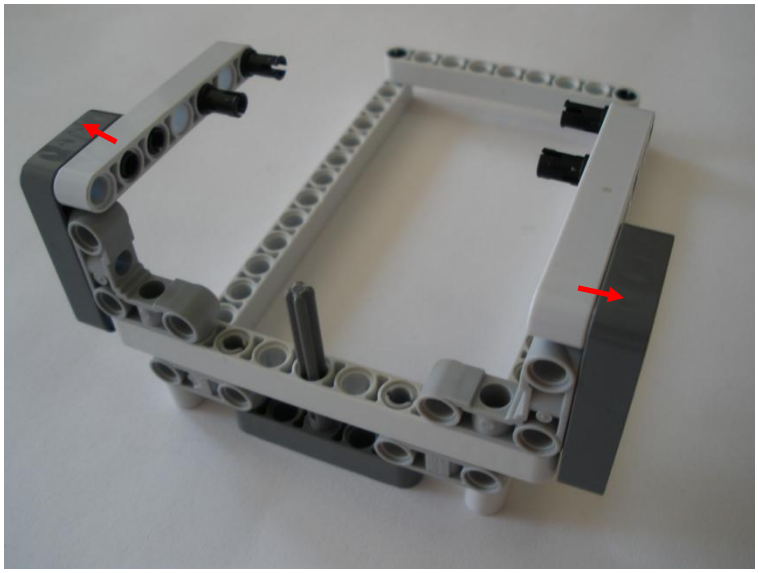





11

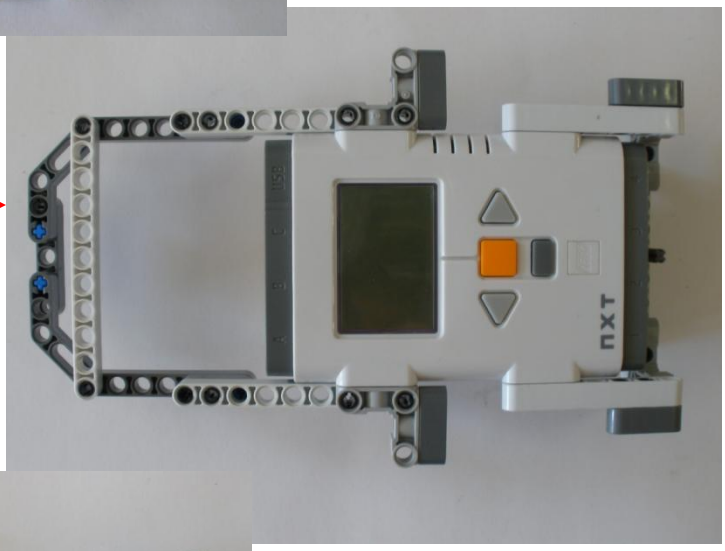


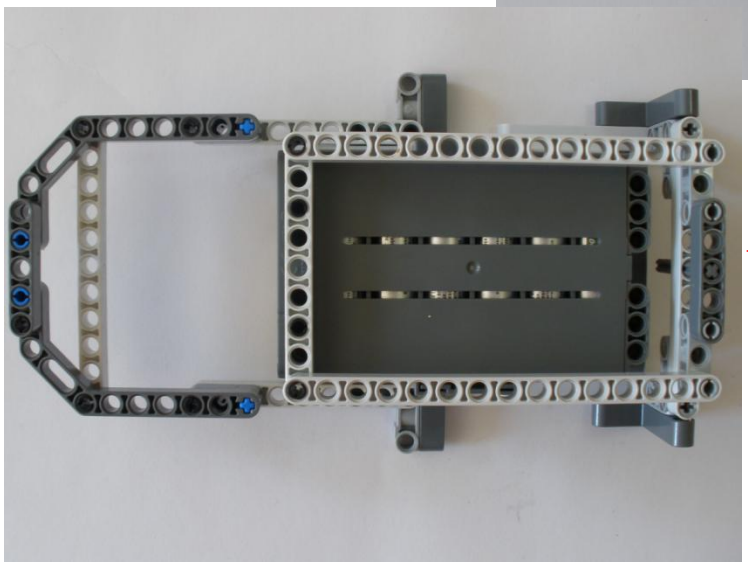

7



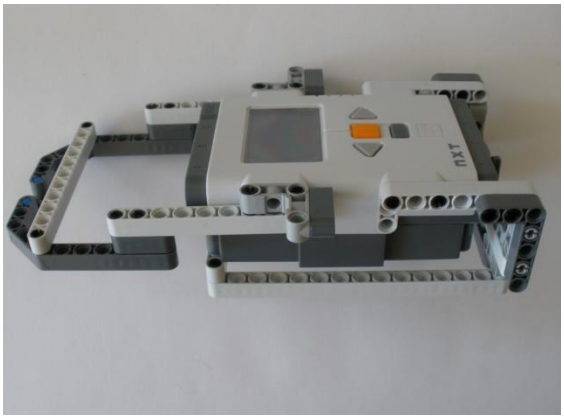
La segunda pieza se acopla en el brick.

Así quedaría visto desde arriba.

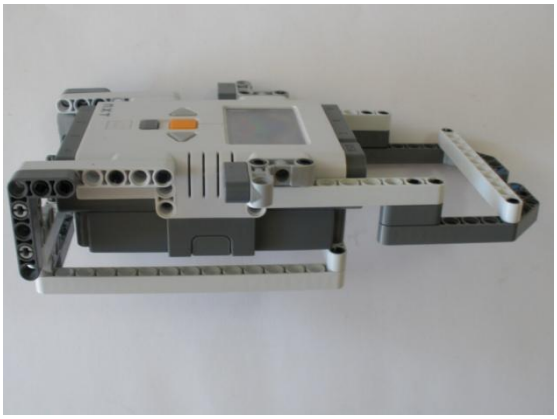




Visto desde abajo.



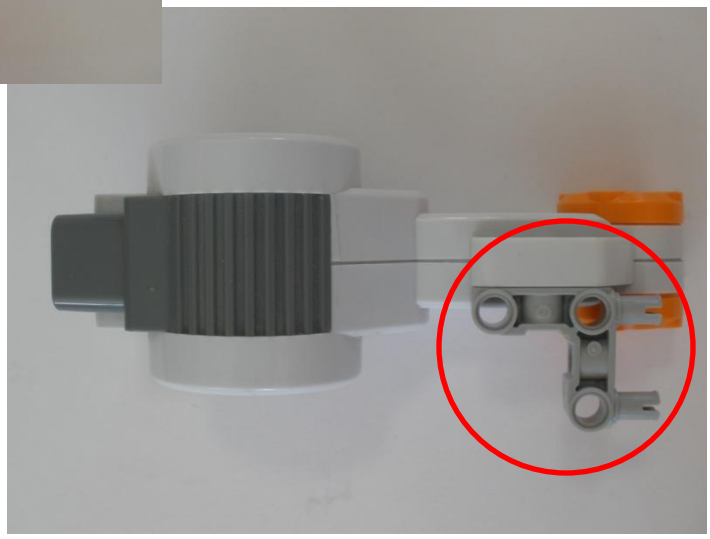
Lado izquierdo.

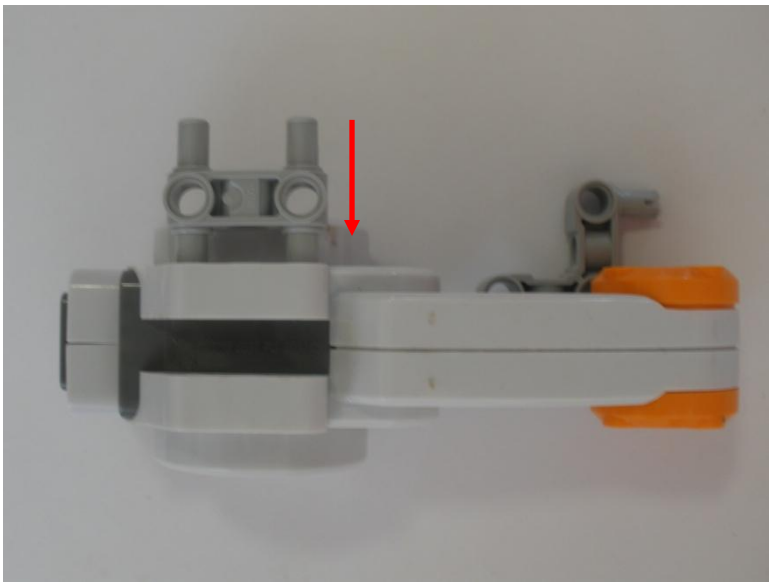


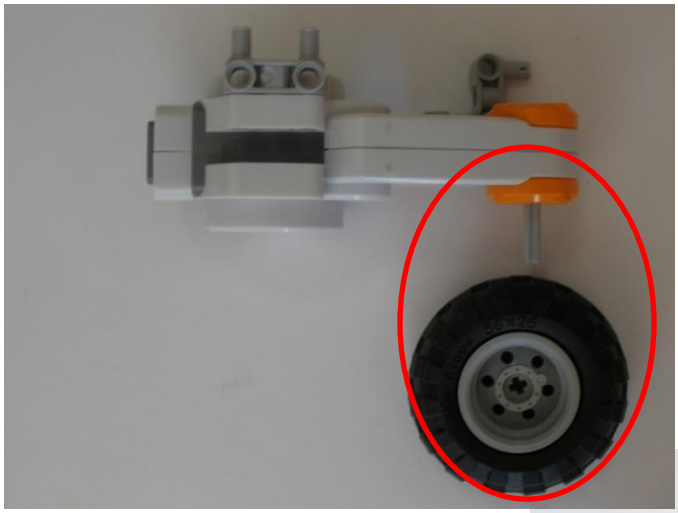
Lado derecho.

Paso 2: Llantas.

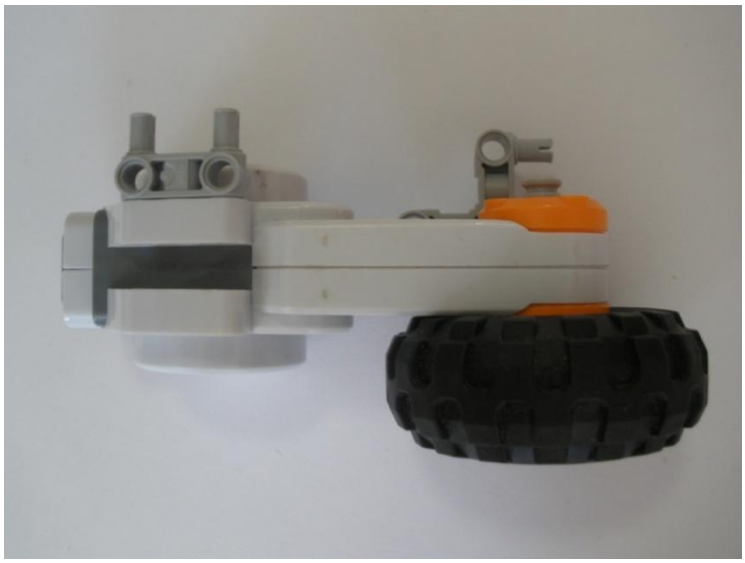
Piezas a utilizar:



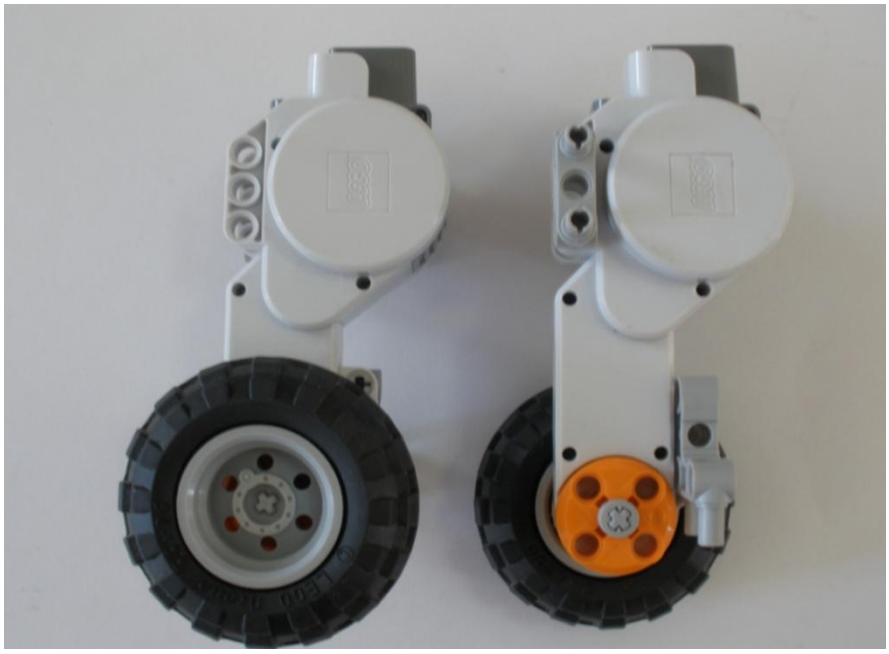





LEGO Group 56 x 26

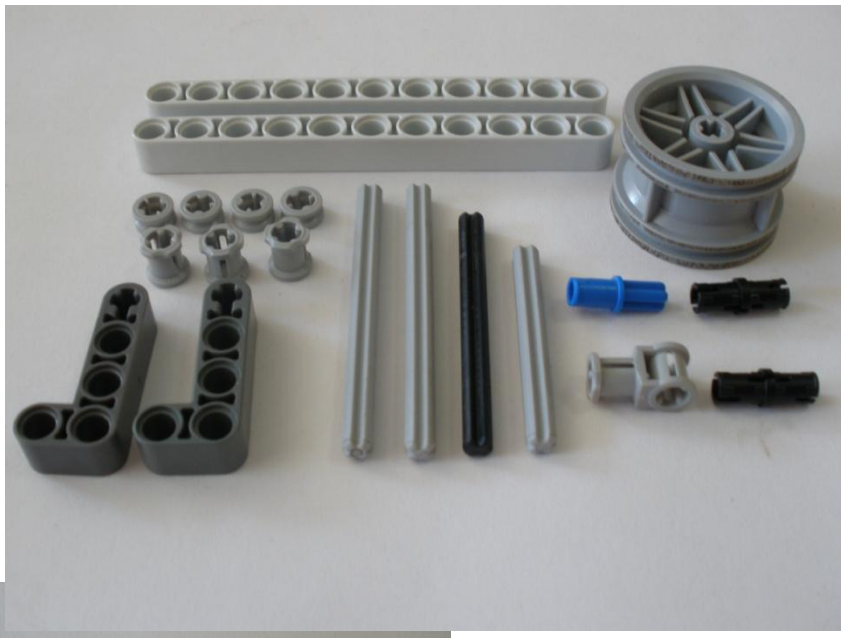


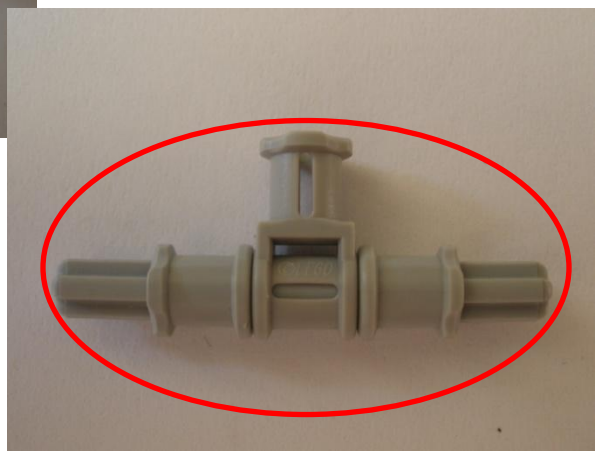
Solo se muestra el ensamble de una llanta, la del otro lado se ensambla similar y así terminaríamos con nuestro par de llantas.

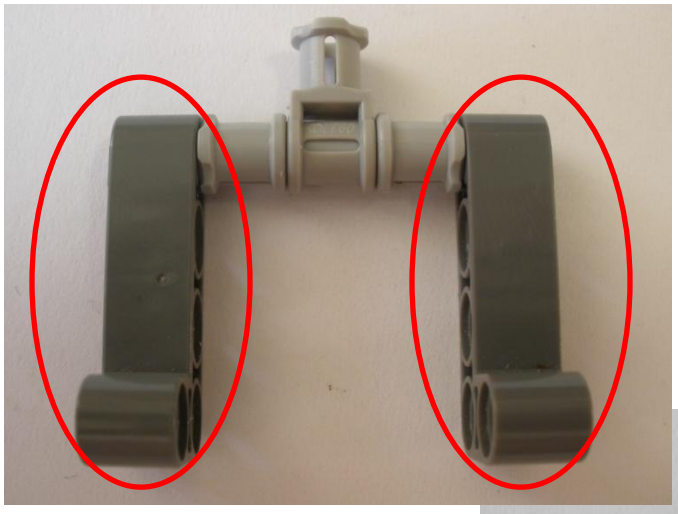


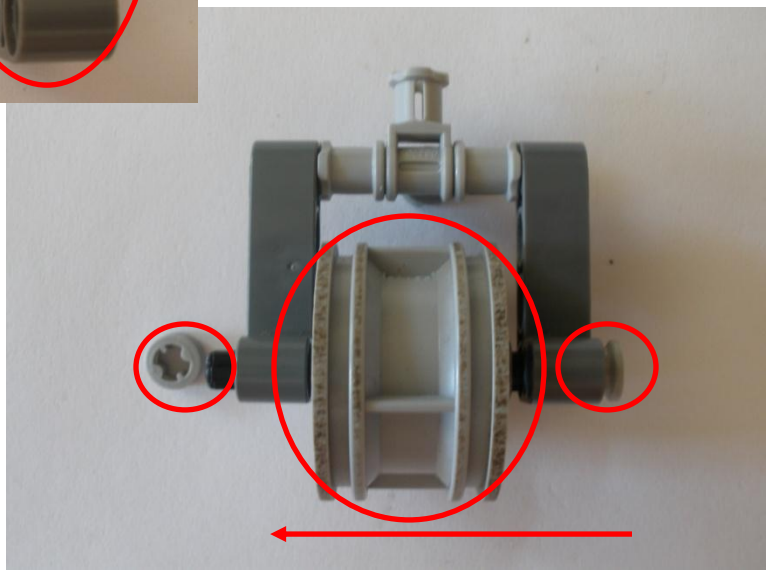
Paso 3: Rueda delantera

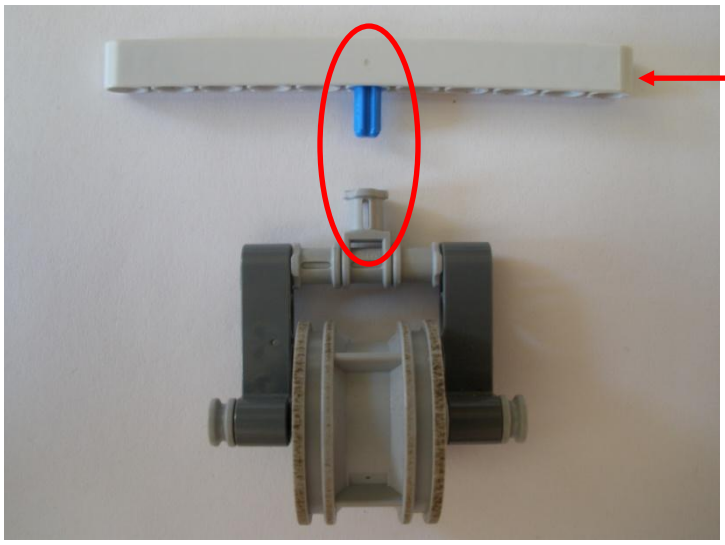
Una vez terminadas las ruedas traseras proseguimos con la rueda delantera. Estas son las piezas necesarias.

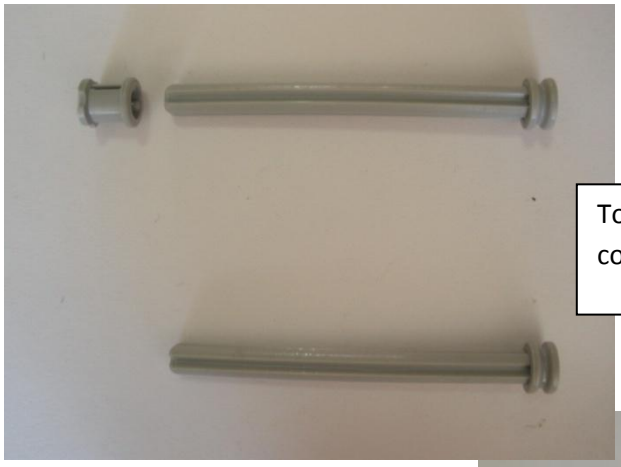




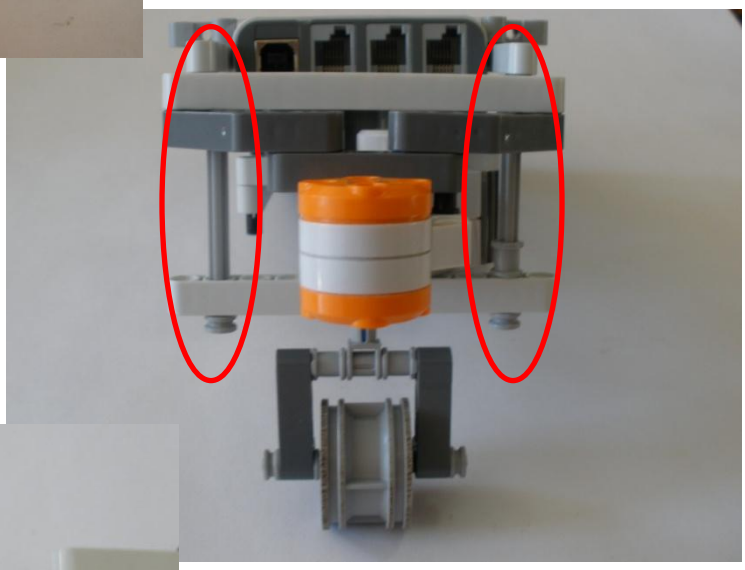


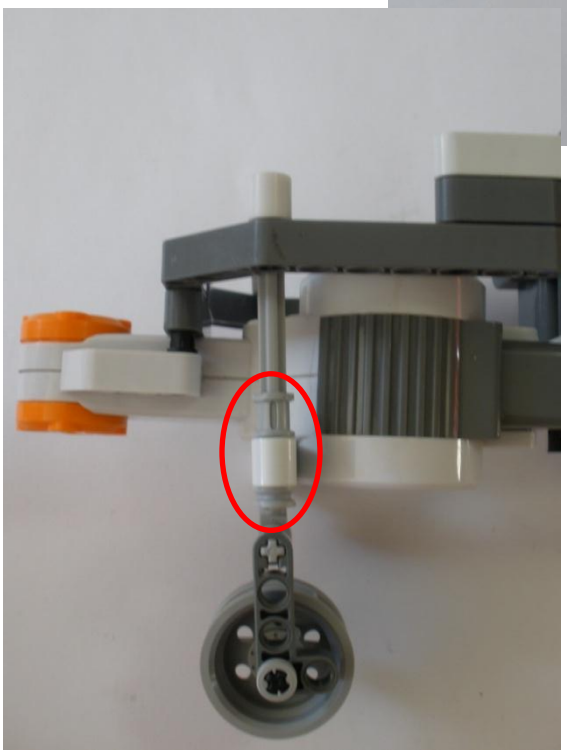


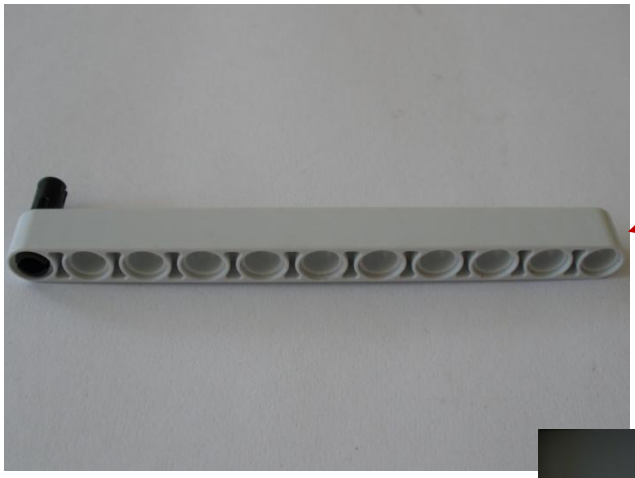

11

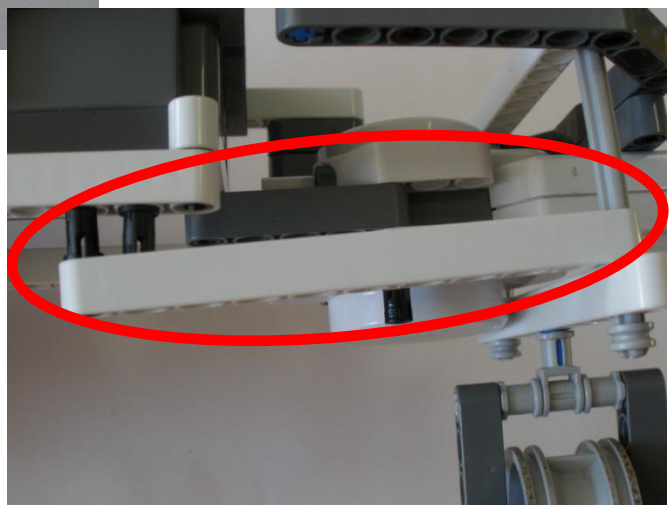


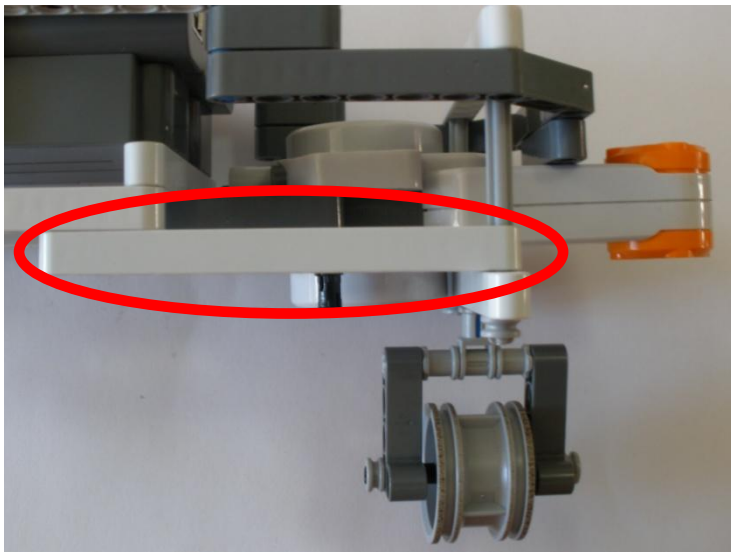
Tomamos este par de piezas y las colocamos como se muestra en la figura.



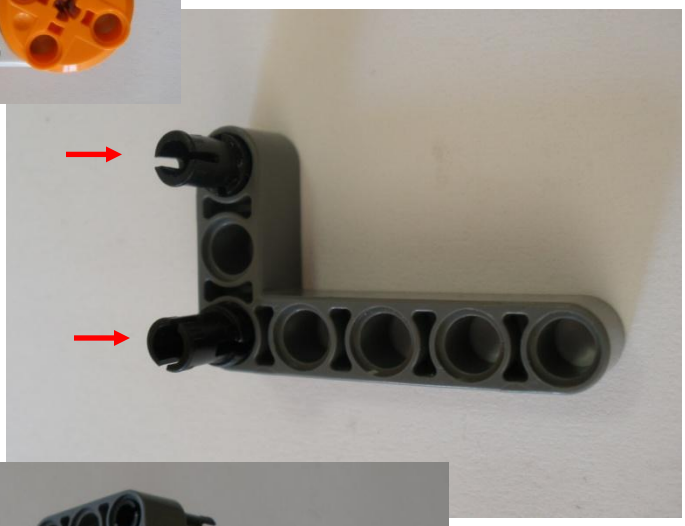


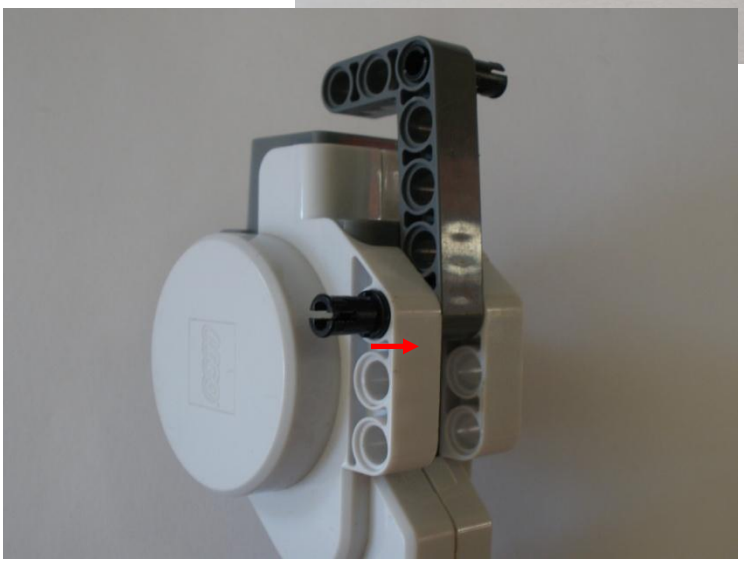

11

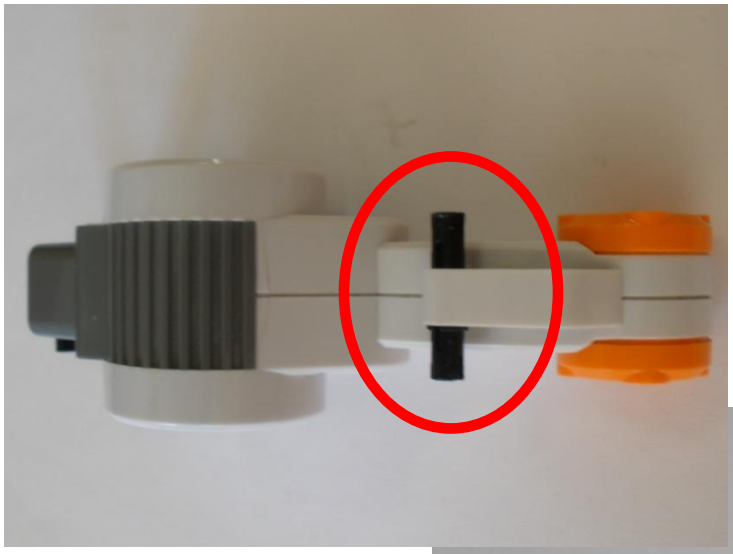


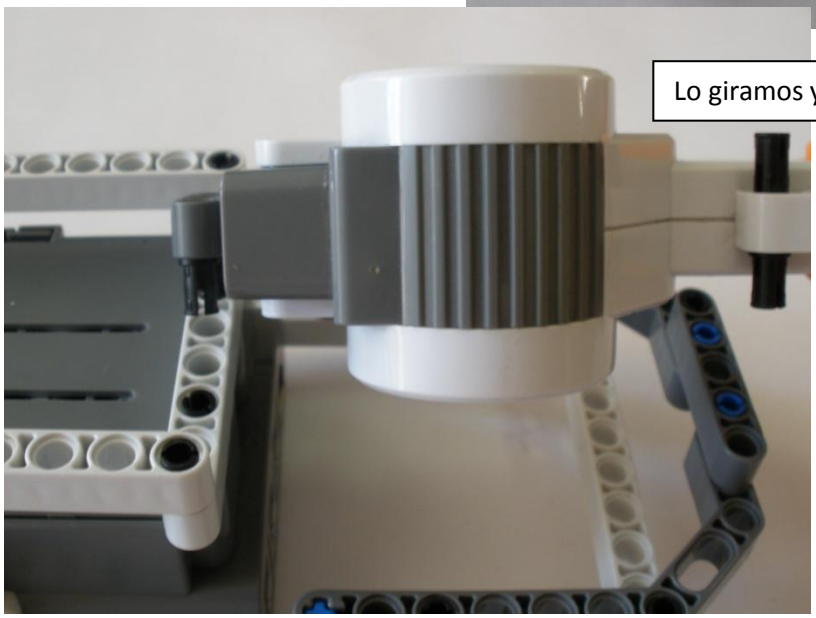


Paso 4: Cuello.

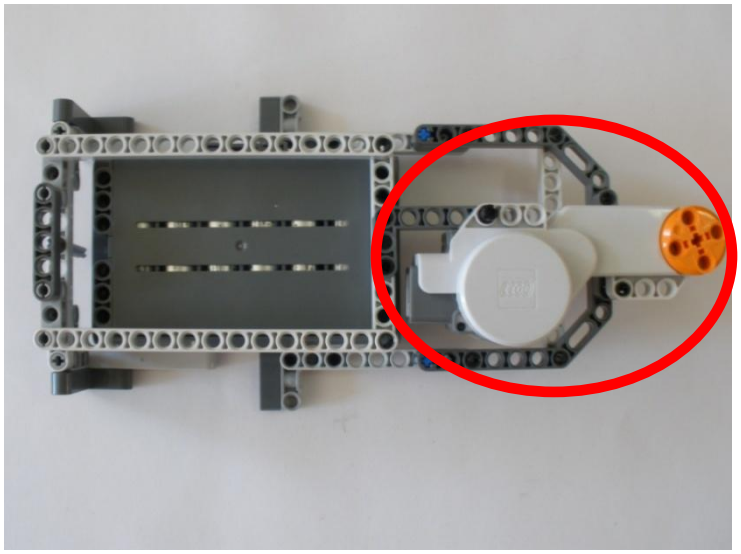




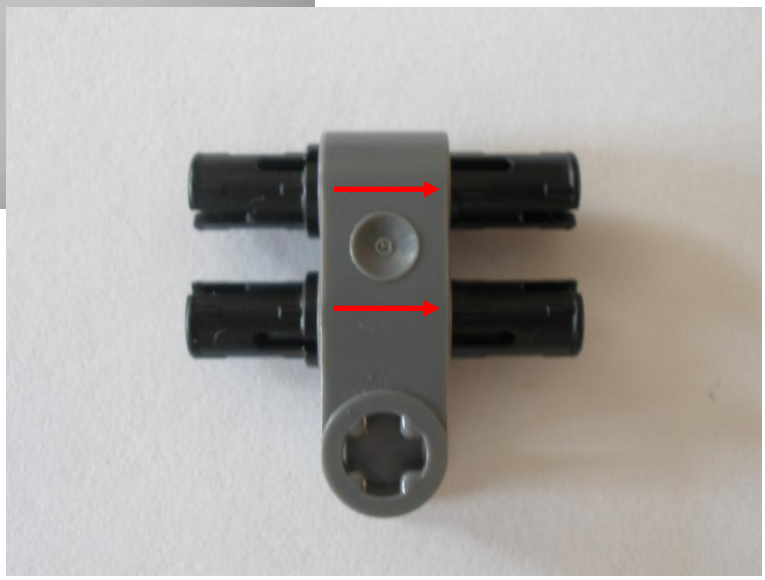


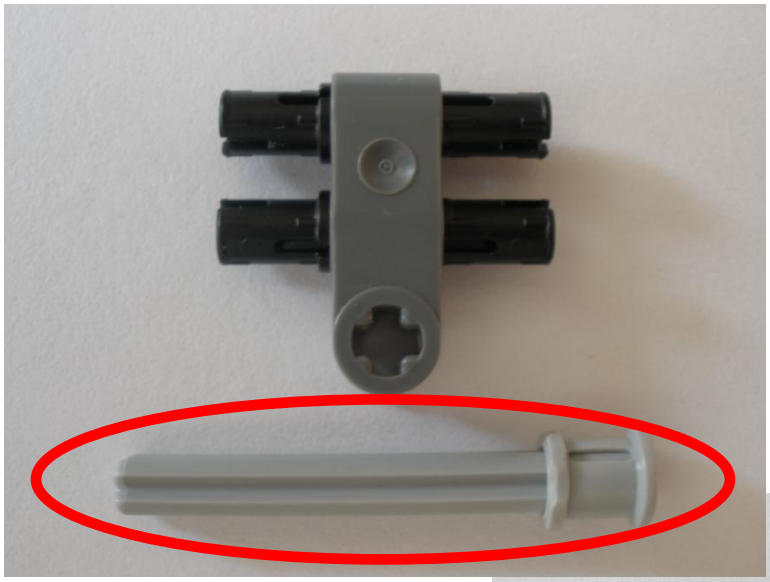


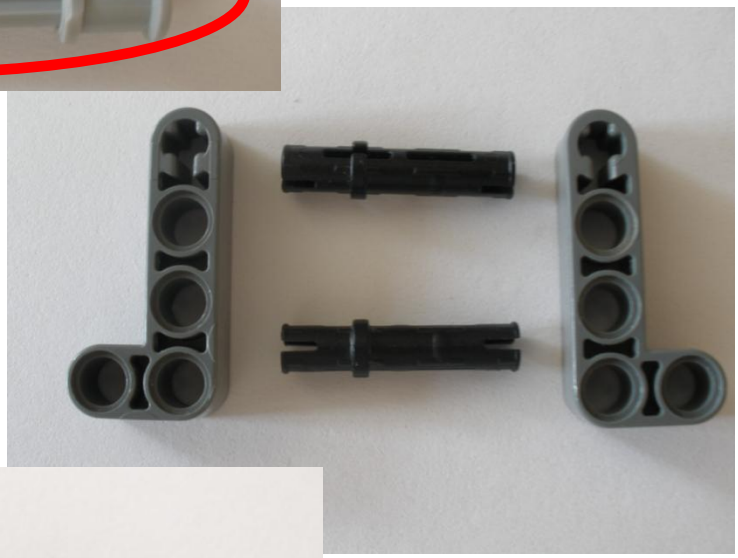
Lo giramos y lo colocamos en el brick

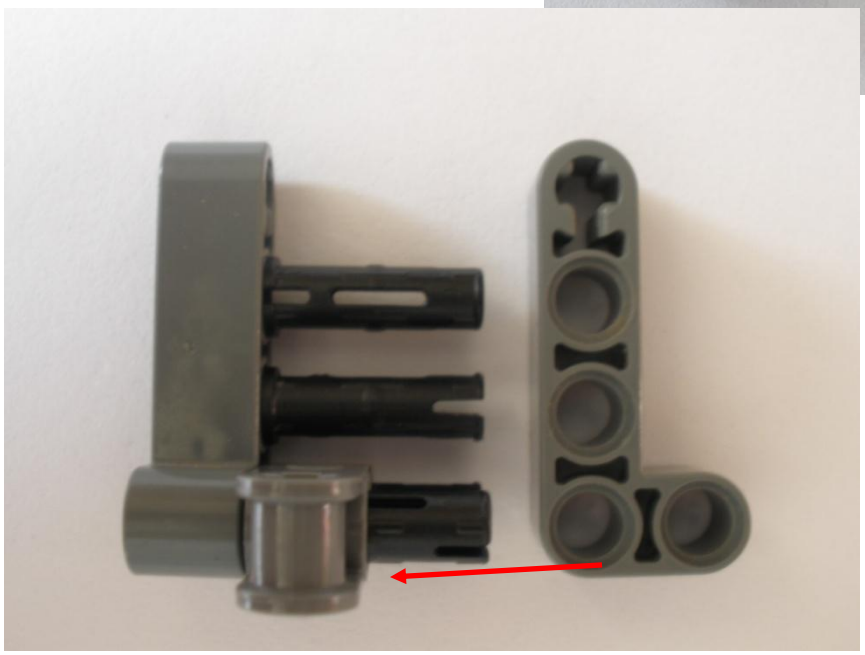


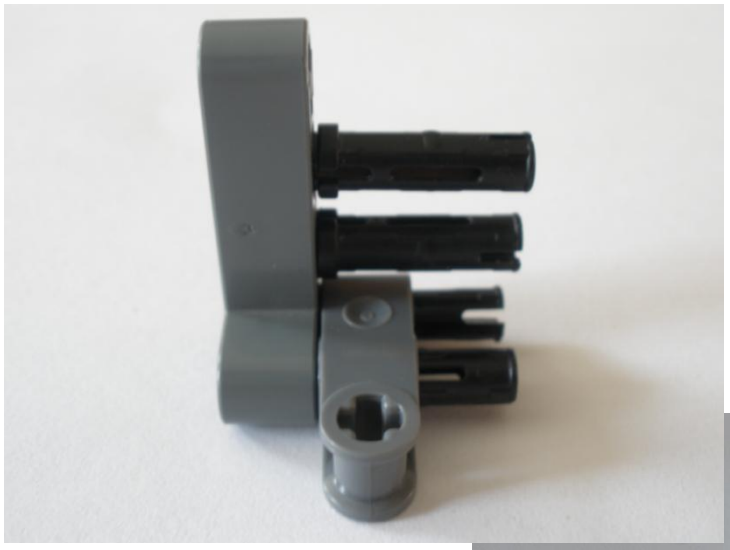
Una vez terminado con el servomotor del cuello proseguimos con la cabeza y cuello. Para formar la figura que se muestra abajo.

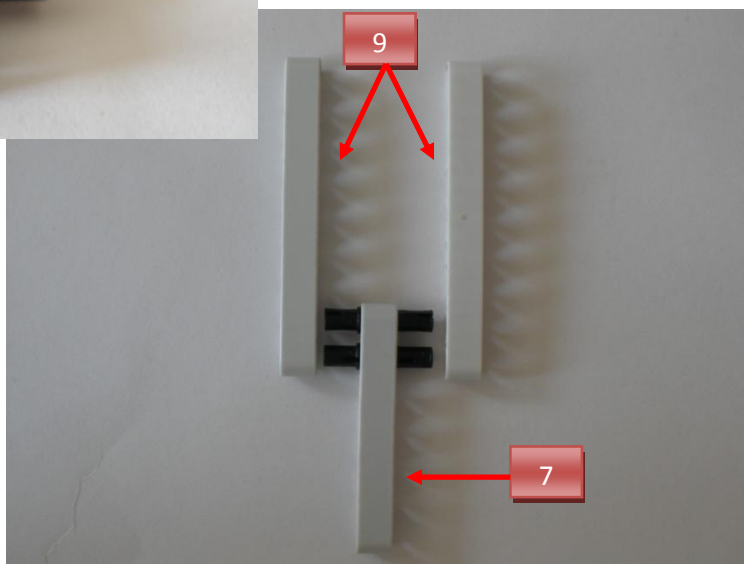


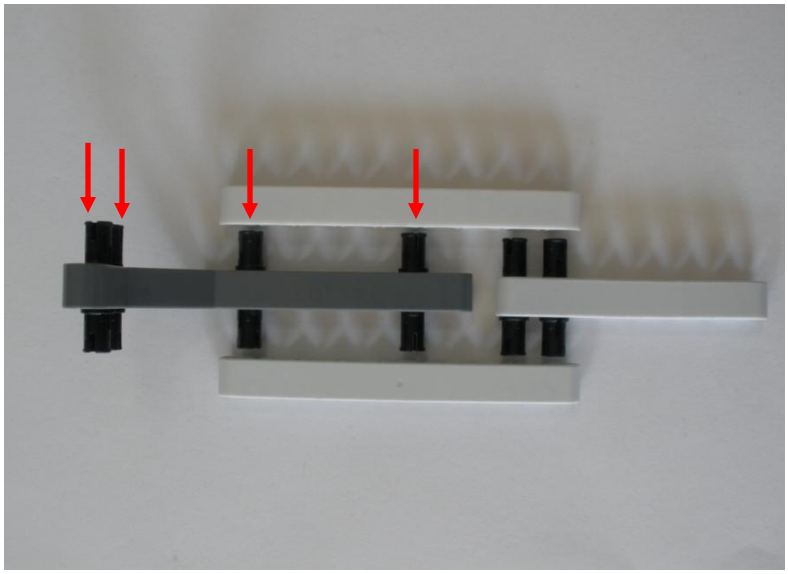


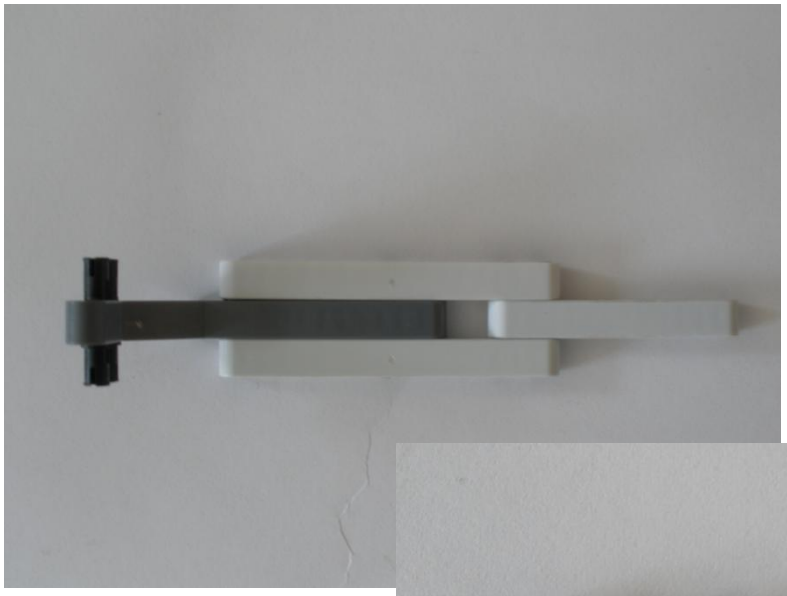


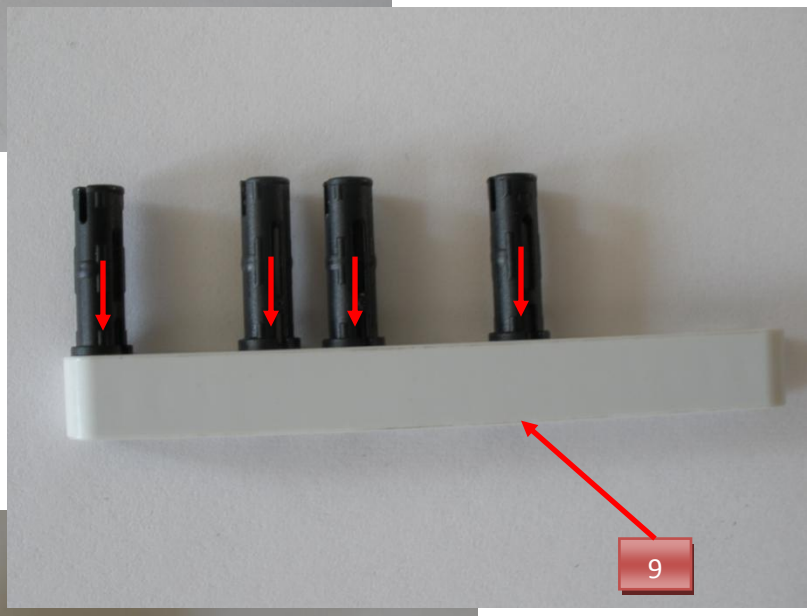


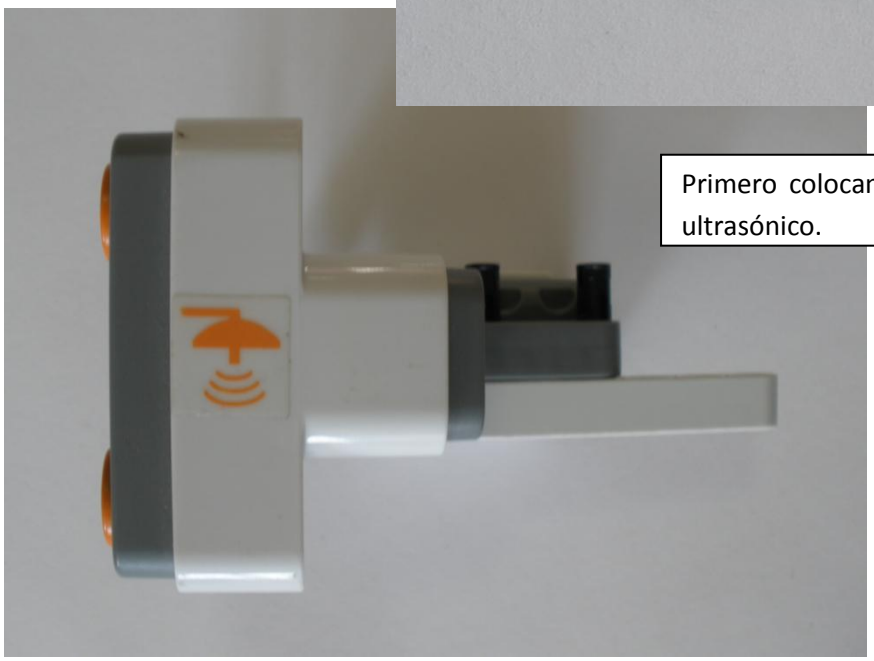



9
7

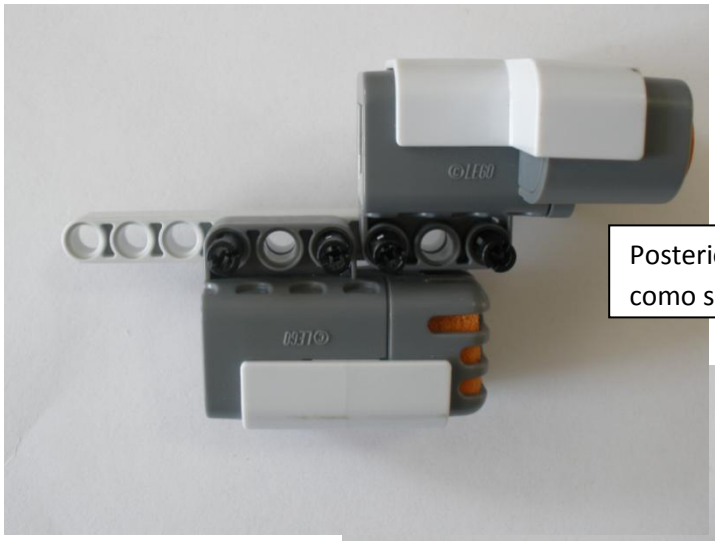




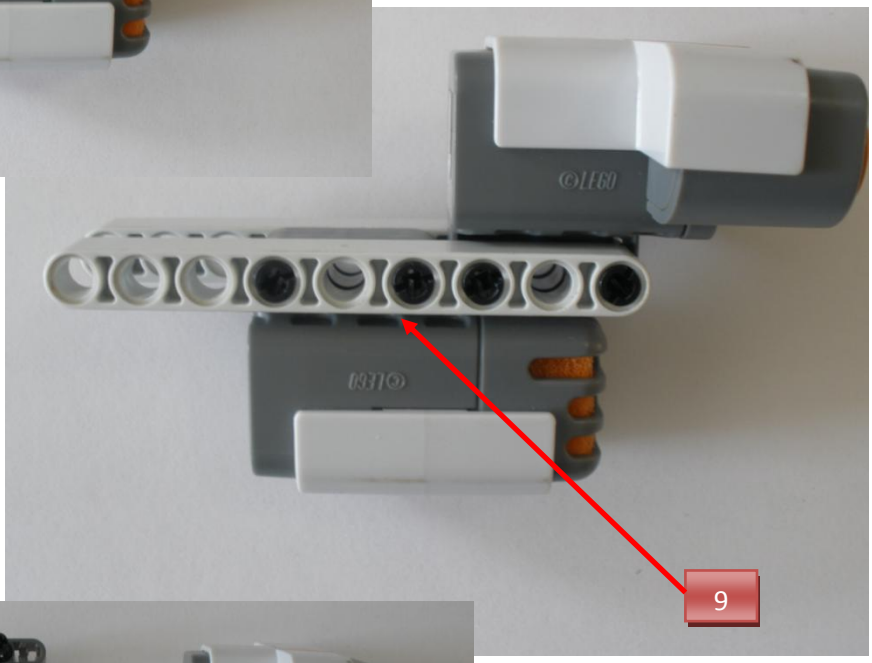


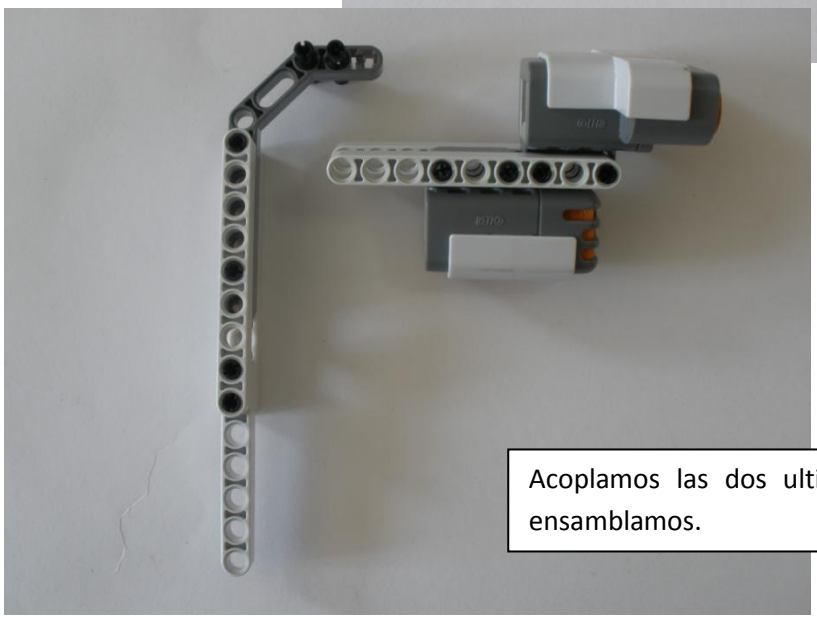


Primero colocamos nuestro sensor ultrasónico.

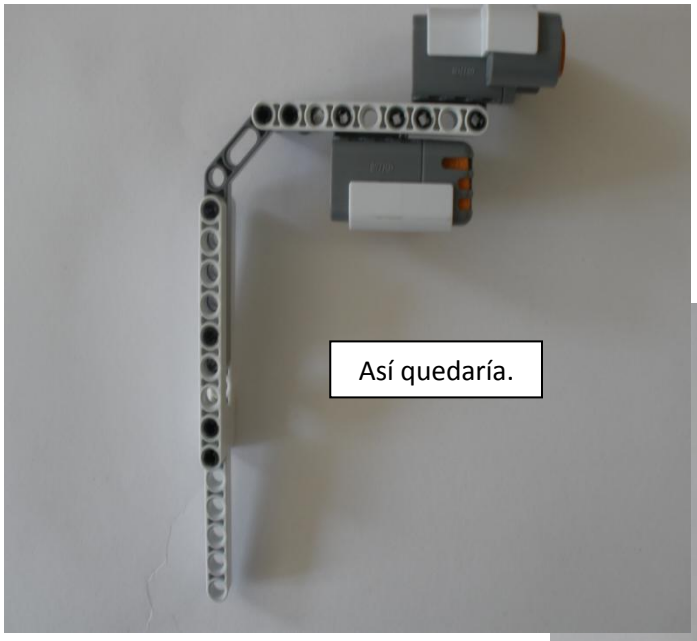


Posteriorment6e el sensor de sonido, tal como se muestra en la figura.





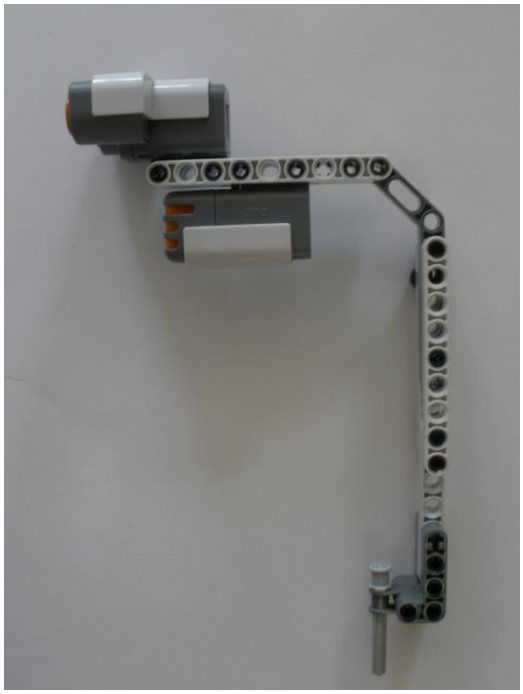
Acoplamos las dos ultimas piezas que ensamblamos.



Así quedaría.

Agregamos esta pieza que previamente habíamos ensamblado.

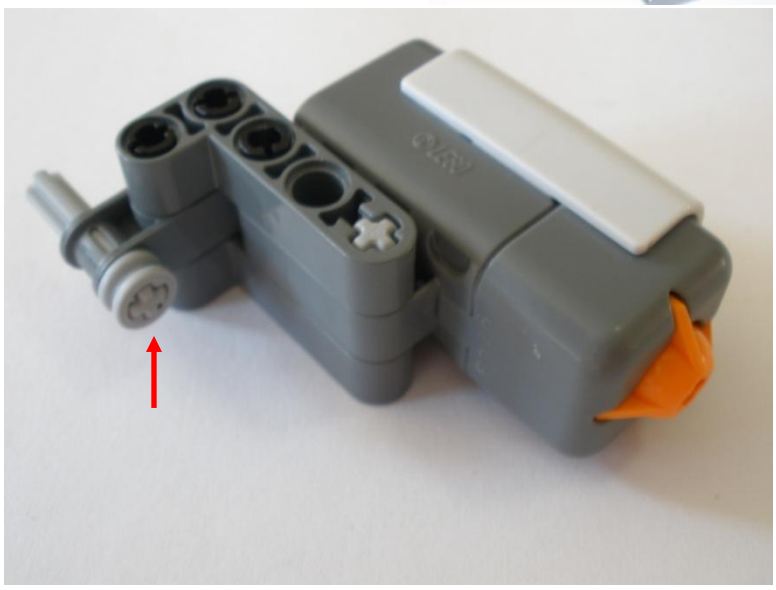
Y con estas dos terminamos la cabeza y el cuello.



Al terminar la cabeza y cuello se ensambla el sensor de tacto que es el que va apagar al robot al llegar a su destino.

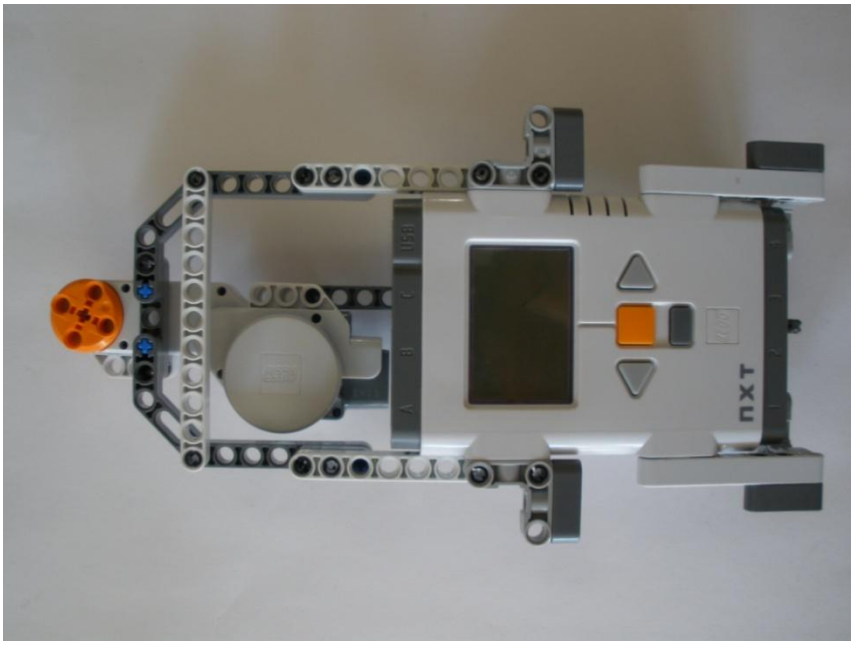
Paso 5: Sensor de tacto.

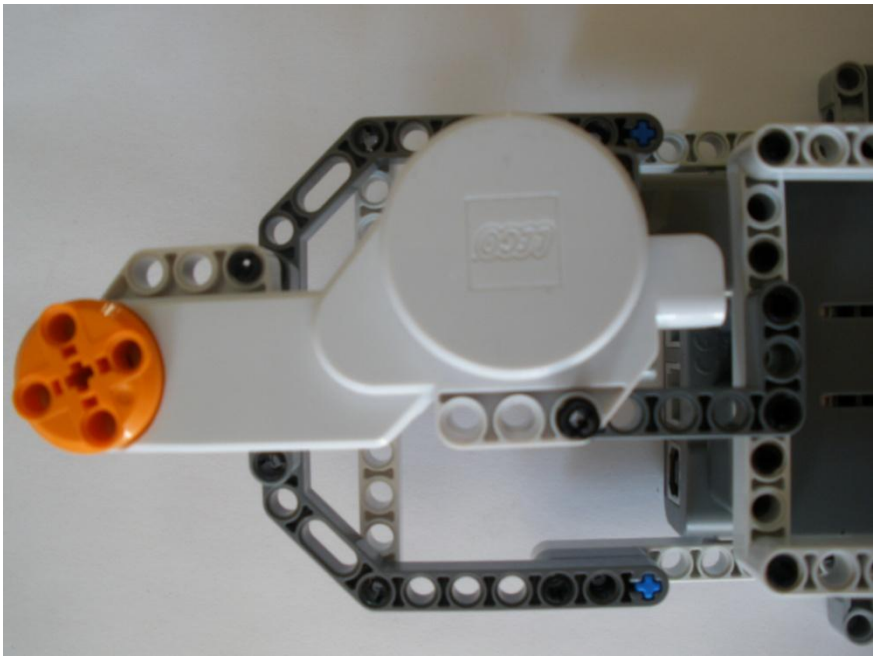
Colocamos el sensor de tacto.

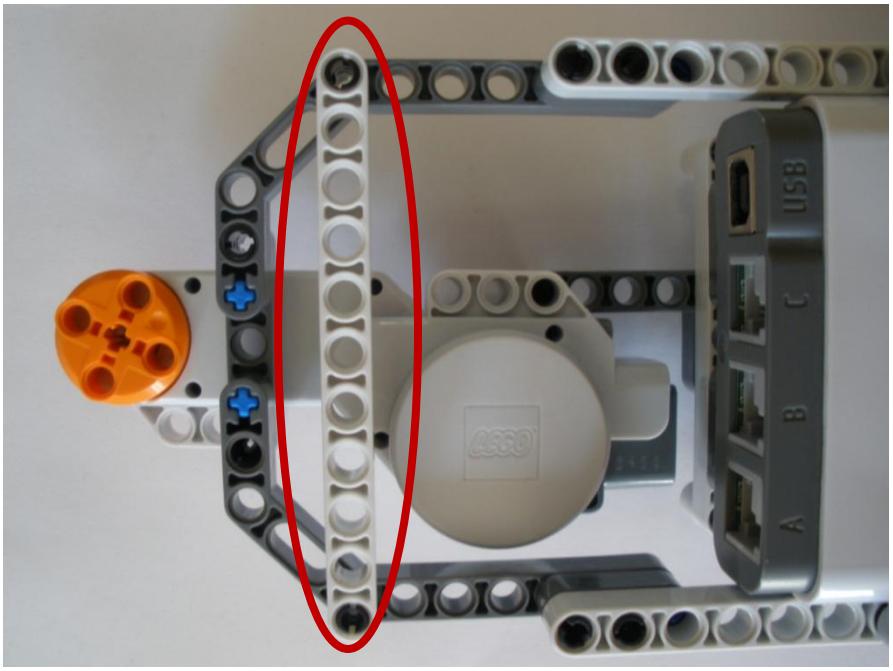


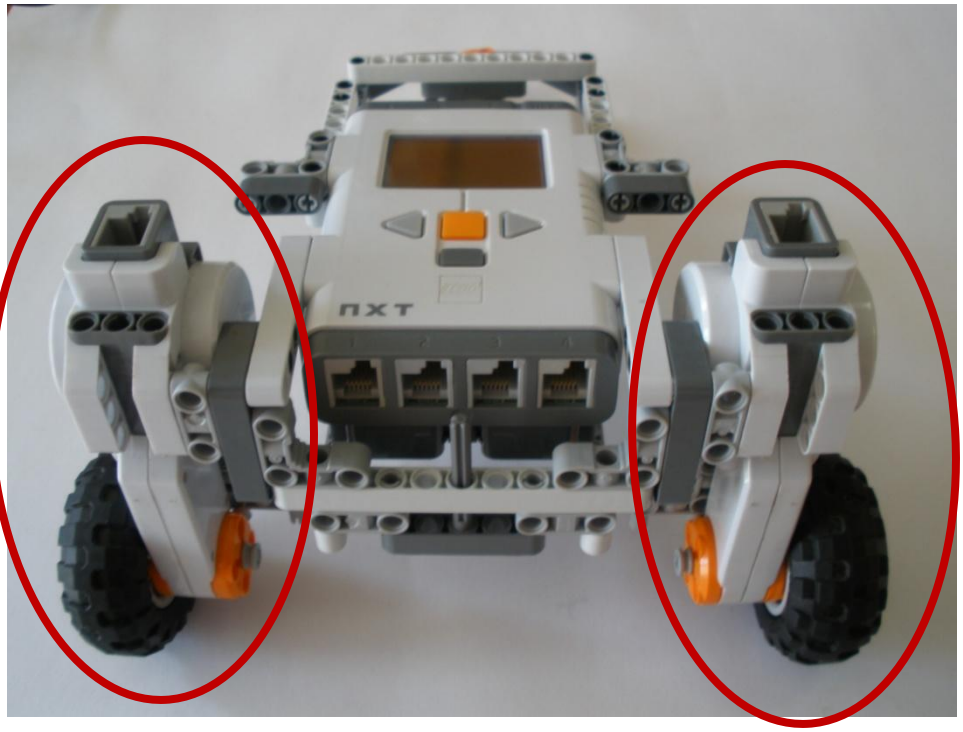
Paso 6: Armado completo.

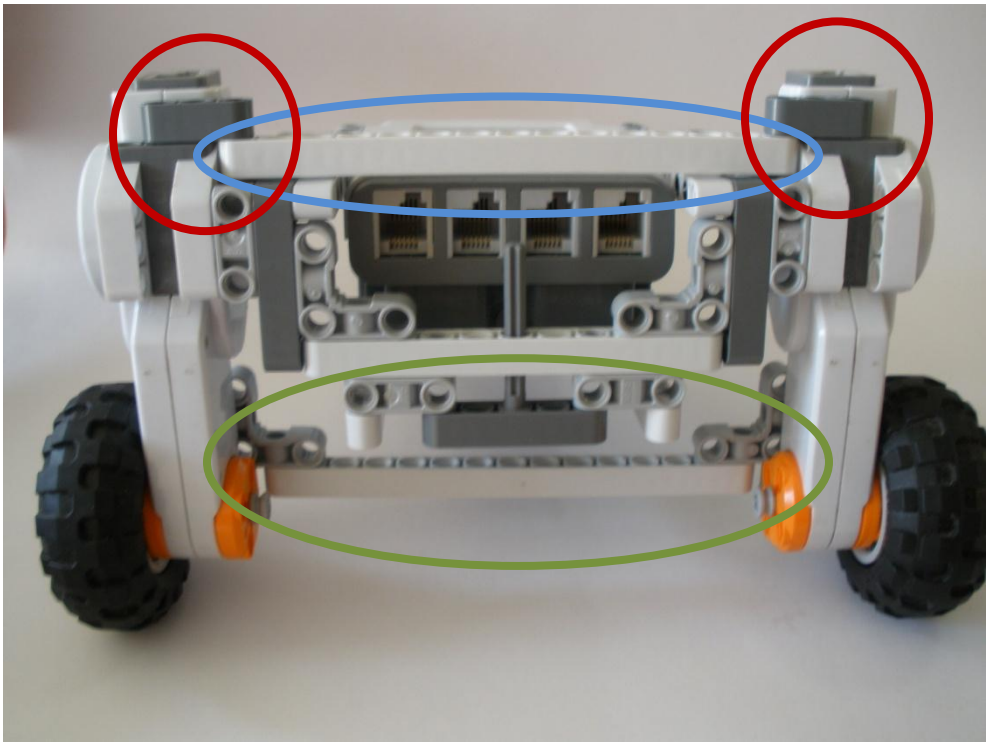
Al terminar con este sensor unimos todas las piezas de la siguiente forma.

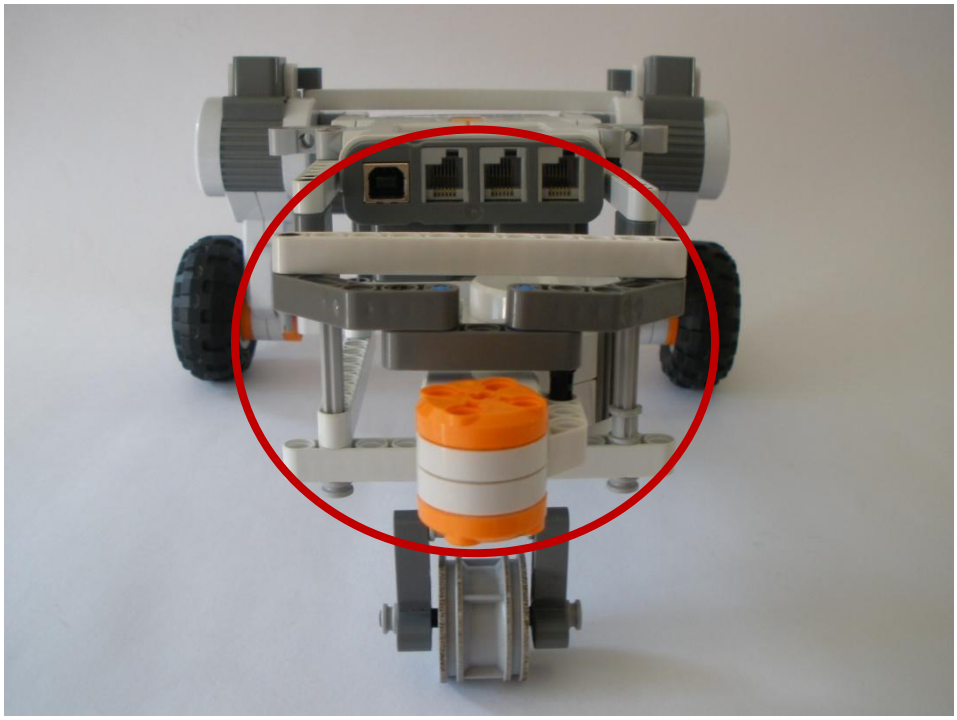


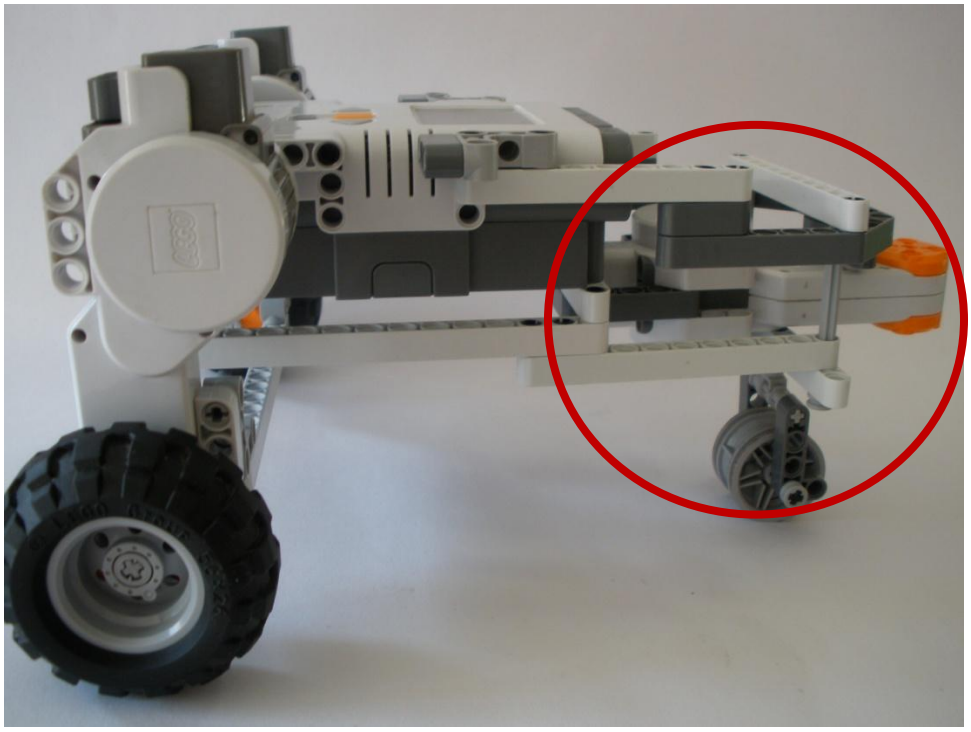


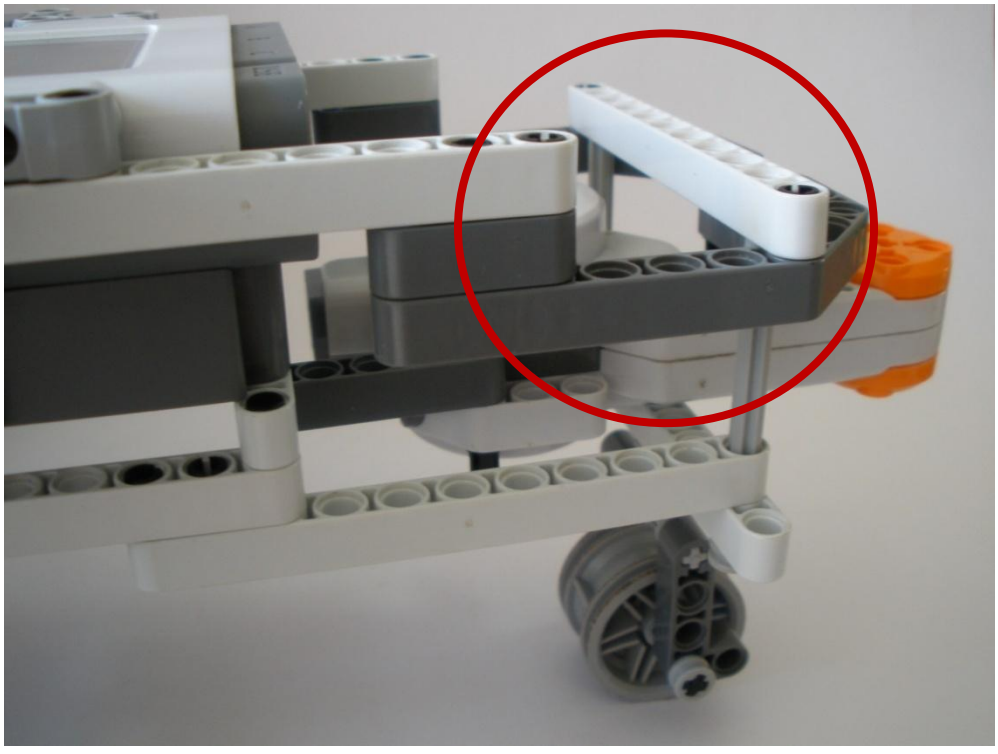

USB
C
B
A

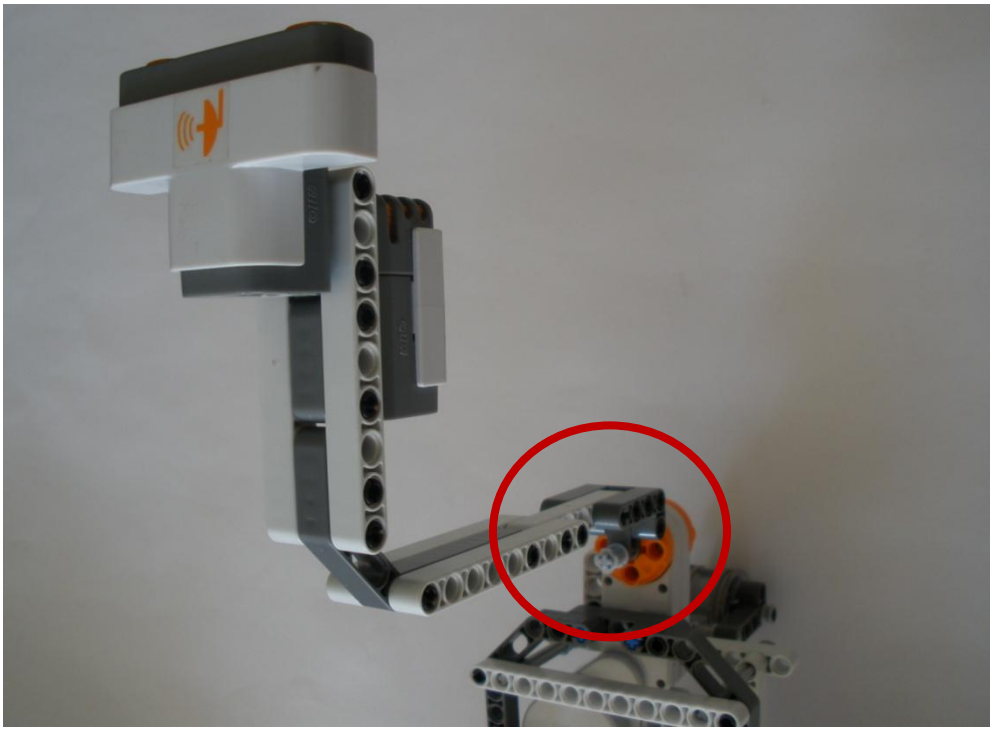

NXT

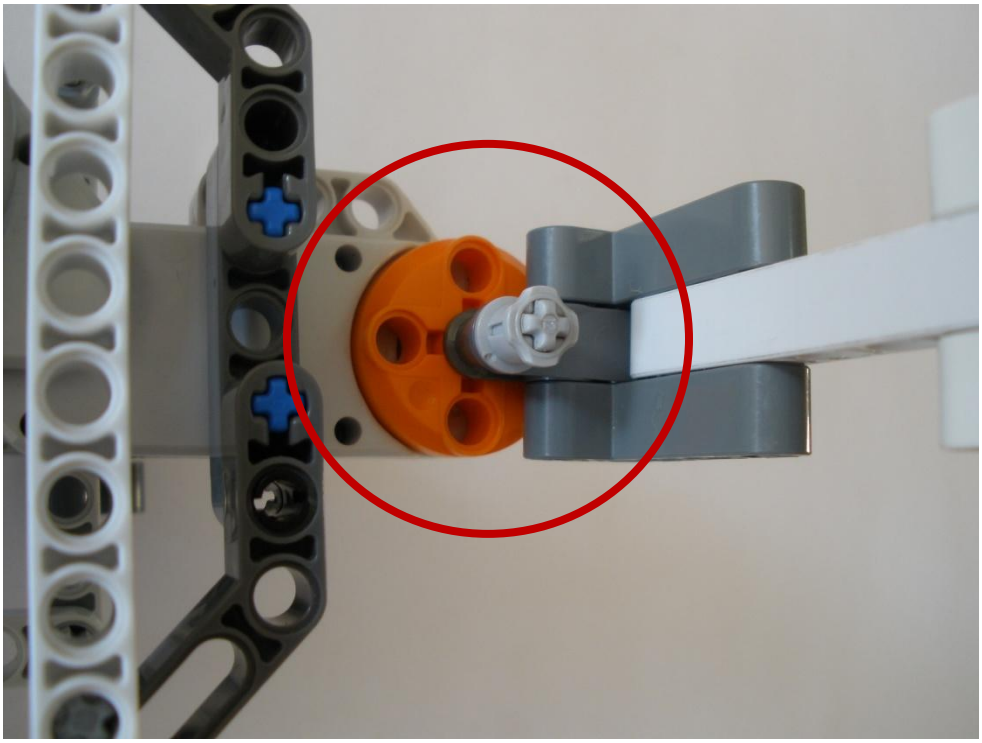


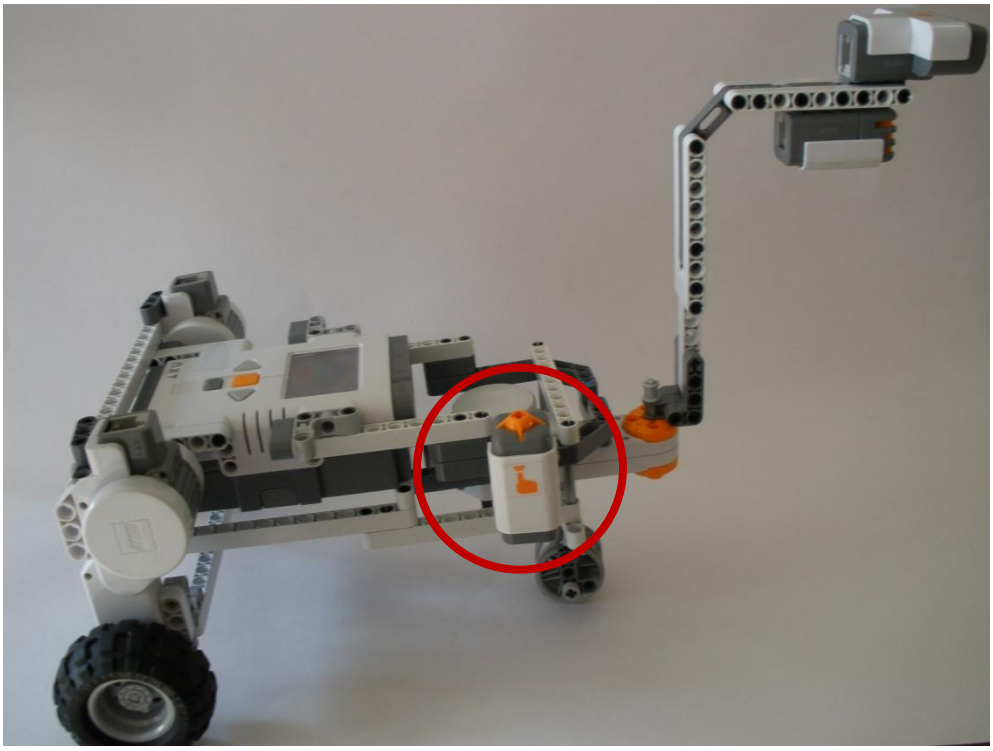


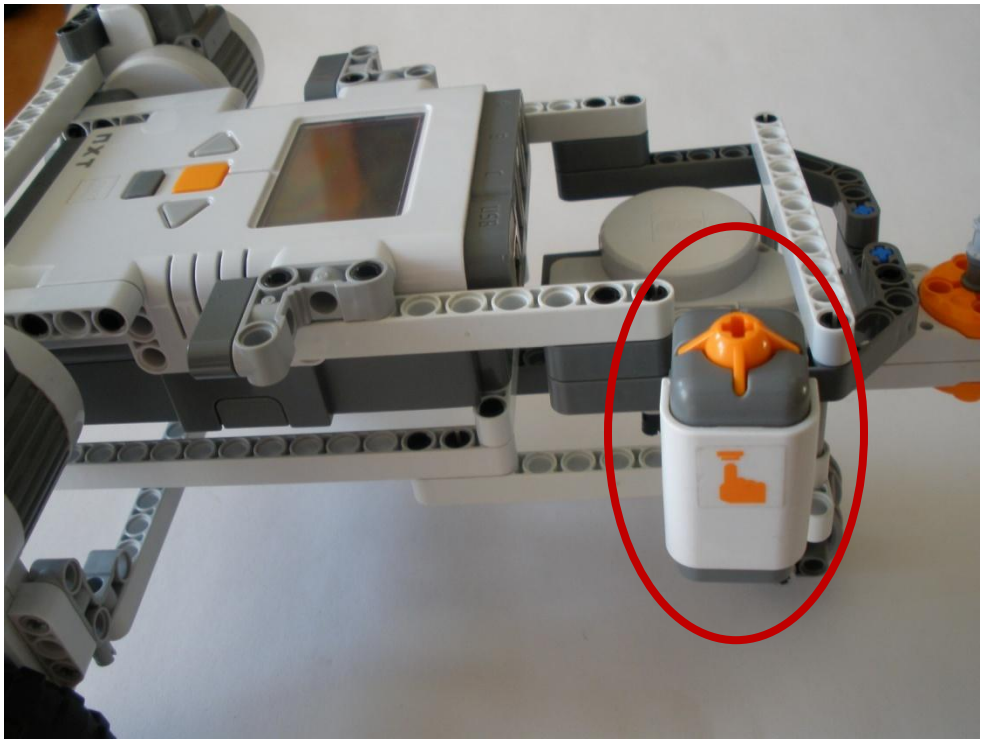




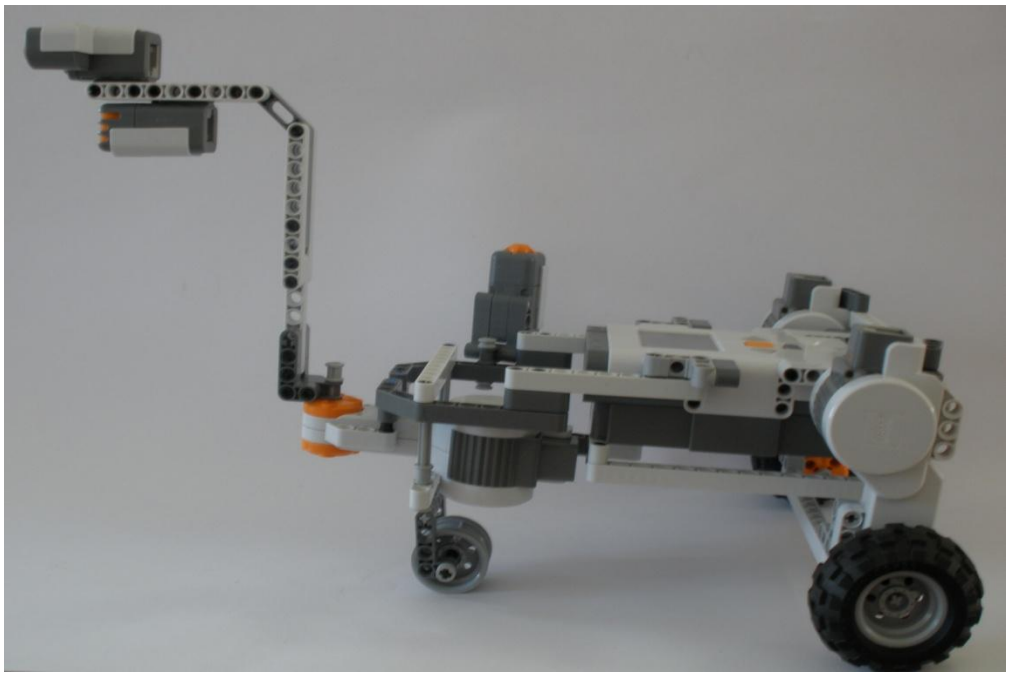


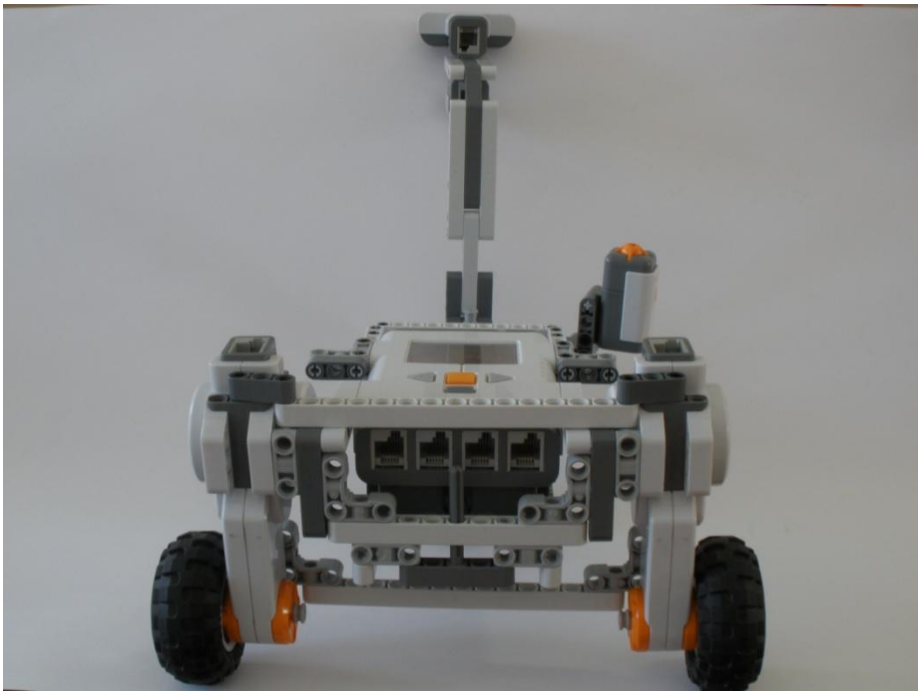




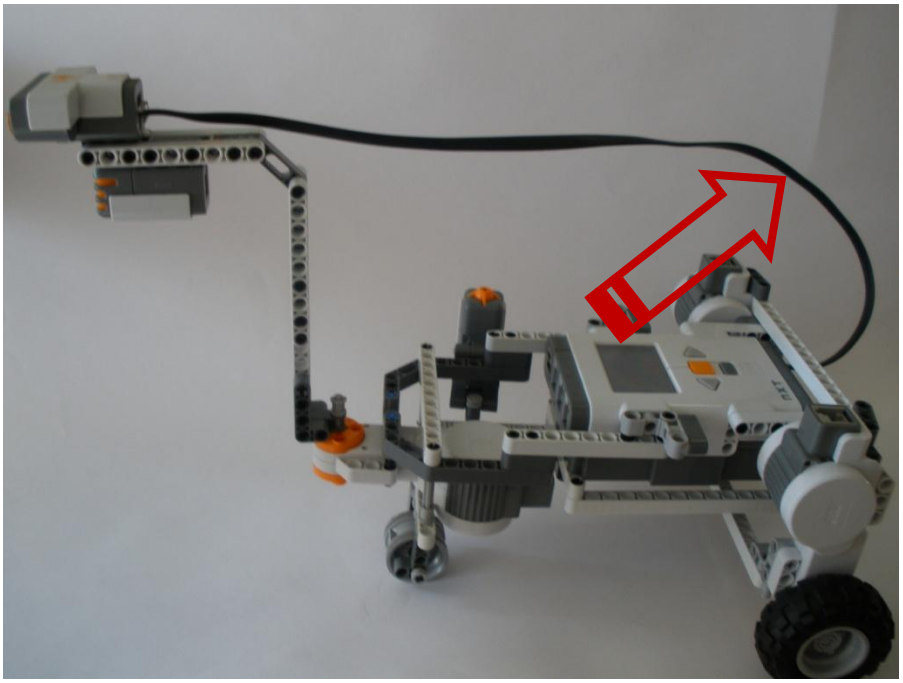


Así quedaría nuestro robot.

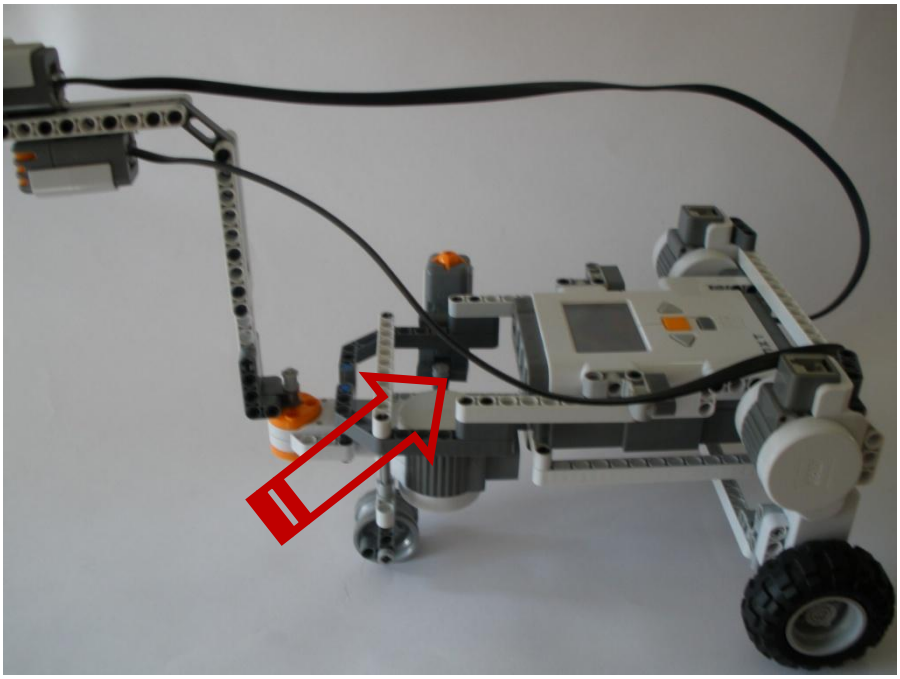




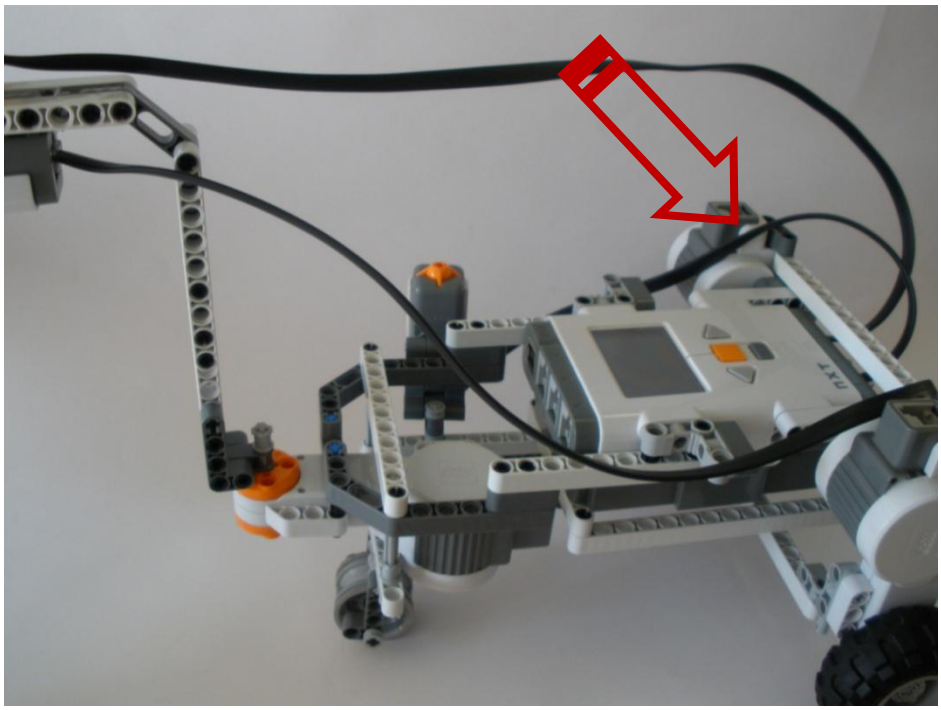
Al terminar con el sensor de tacto, instalamos los cables. El cable del sensor ultrasónico, lo conectamos en el puerto 4.



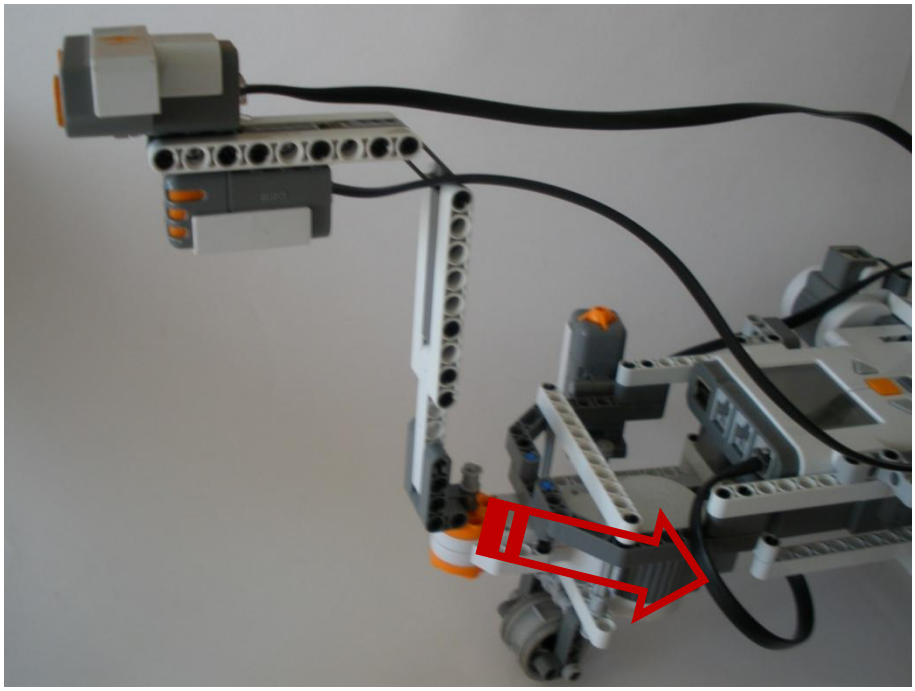
El sensor de sonido en el puerto 1.



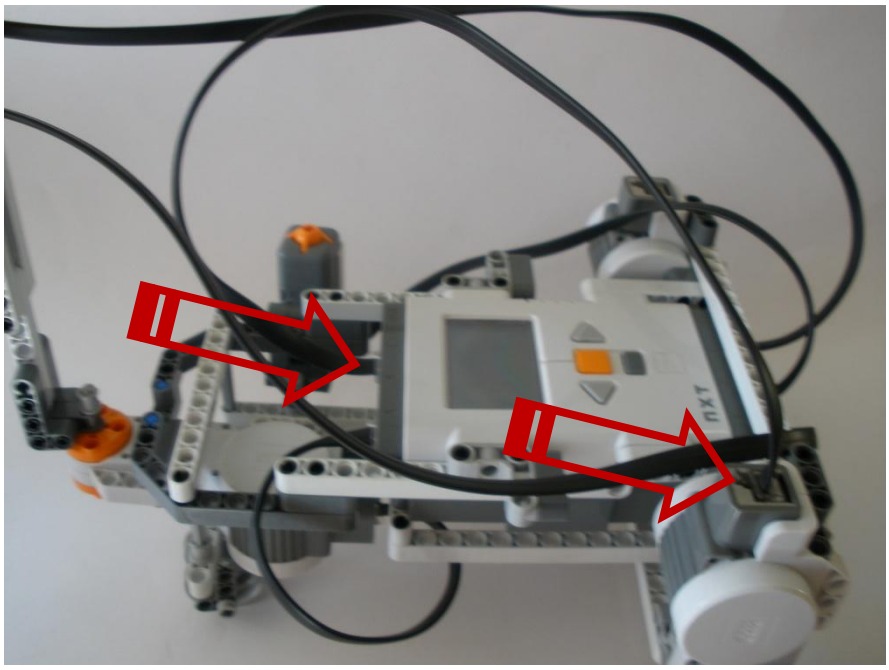
Sensor de tacto en el puerto 2.



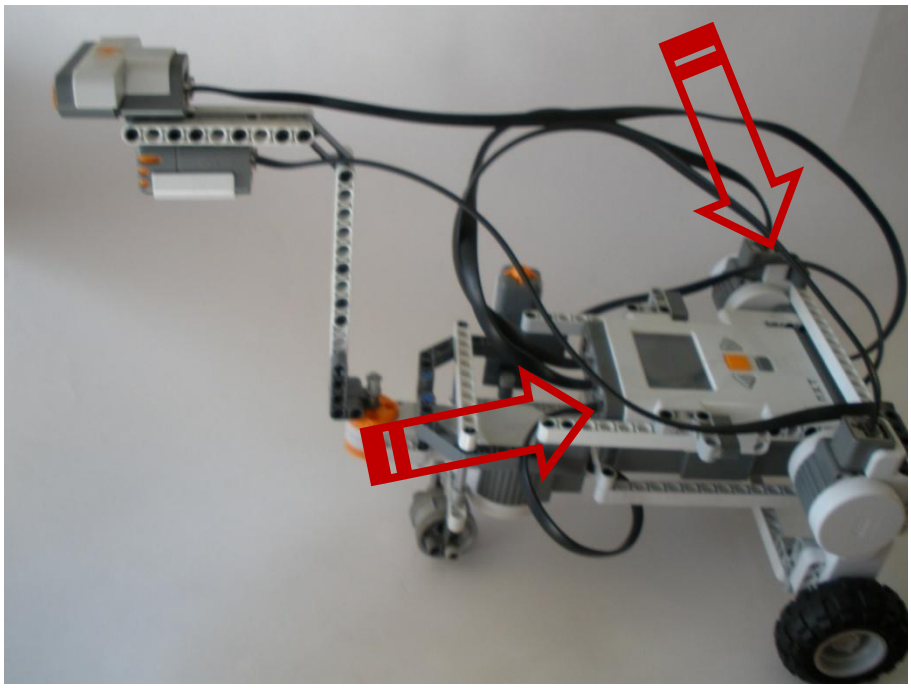
Motor del cuello, en el puerto “A”.

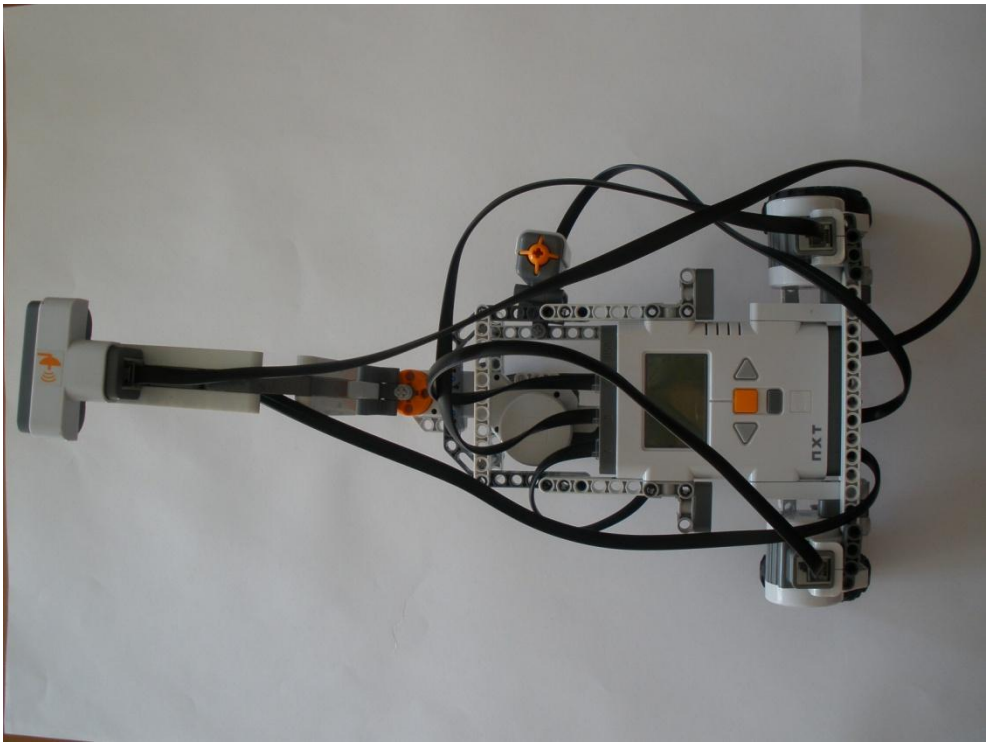


Motor izquierdo del brick en el puerto “C”.



Motor derecho del brick en el puerto “B”.



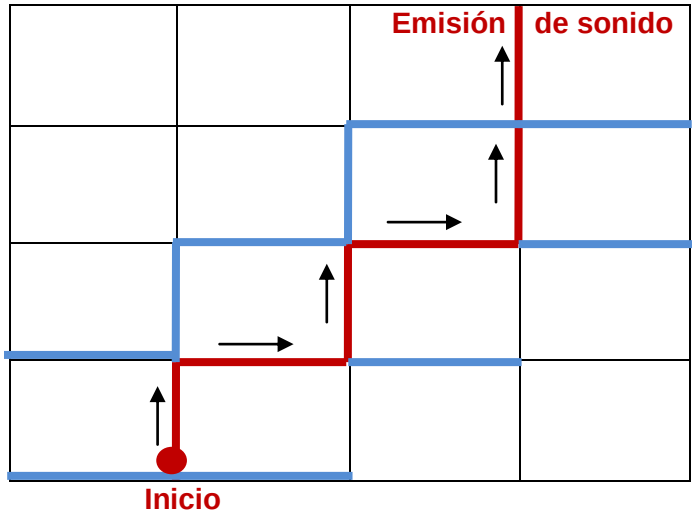


Con esto finalizamos la construcción de nuestro robot.

Programando el ChalanBot con NXC

ChalanBot pretende ser el ayudante ideal. Llevando herramienta de cualquier tipo a quien lo necesite. El funcionamiento de este robot es limitado debido a las piezas con las que cuenta el kit de Lego y el numero de sensores y servomotores que puede manejar el Brick NXT. Sin embargo no deja de cumplir el objetivo de trasladar cosas de un lugar a otro. Para llevar a cabo sus tareas se instalaron los siguientes sensores: sensor ultrasónico, para detectar objetos antes de que ChalanBot avance, evitando así el impacto con algún objeto, sensor de sonido que percibe el sonido del entorno en tres direcciones derecha izquierda y frente, una vez medidos estos valores determina cual es el mayor y se dirige hacia el, un sensor de tacto solo para apagar el funcionamiento del Robot. Se utilizan los tres servomotores disponibles uno para el girar el cuello del robot en las direcciones

comentadas con anterioridad, un servomotor que hace girar la rueda izquierda y el ultimo para hacer girara la derecha. A continuación se muestra el algoritmo de ChalanBot:



En la figura se muestra como se desplaza el ChalanBot, las líneas rojas indican el camino que sigue hasta el objetivo, las azules muestran hacia donde vira el cuello y captura los decibeles del sonido almacenando en variables para posteriormente verificar cual es el mayor y que los servomotores funcionen dependiendo de este valor. Se explica con mas detalle en las siguientes líneas.

- El robot parte del círculo de inicio.
- Gira el cuello hacia el lado derecho 90^{o}, almacena en una variable “d” el valor de los decibeles capturados con el sensor de sonido.
- De su posición actual derecha gira hacia el lado izquierdo 180^{o} y captura el numero de decibeles en otra variable “i”.

- Posicionado el cuello del lado izquierdo regresa a su posición original el frente girando grados al lado derecho, captura el sonido y guarda su valor en una variable llamada “f”.
- Una vez que tiene las variables “i, “”d” y “f” con un valor almacenado. Compara su valor, si i>d
  i>f gira a la izquierda,
  sino
  si d>f
  gira a la derecha
  sino avanza hacia al frente
  Antes de cada giro valida que no se encuentren objetos en frente, si no hay avanza.
- El ciclo sigue hasta que se presione el botón de tacto.

Nota:

El sonido emitido tiene que ser similar a un bip constante, de lo contrario puede variar la captura de decibeles haciendo que el robot se desplace de manera incorrecta.

A continuación se muestra el código:

```
// Chalanbot
// Robot que sigue el sonido, acompañado con un sensor ultrasonico para detenerse si encuentra un obstáculo y un sensor de tacto para apagarlo.
// L.I. Ramiro Robles Villanueva
// 18-Abril-2009

#define mc OUT_C
#define mb OUT_B
#define ma OUT_A
#define SENSOR_TYPE_SOUND_DBA IN_1
#define BUTTON IN_2
#define UMBRAL_US 20


// Procedimiento que gira el cuello del robot al lado derecho.
void gira_ad()
```

```
{
   Off(mc);
   Off(mb);
   RotateMotor (ma,75,90);  //gira a la derecha cuello
   Wait(2000);
}

// Procedimiento que gira el cuello del robot al lado izquierdo.
void gira_ai()
{
   Off(mc);
   Off(mb);
   RotateMotor (ma,75,-180);  //gira a la izq cuello
   Wait(2000);
}

// Procedimiento que gira el cuello del robot al frente.
void gira_af()
{
   Off(mc);
   Off(mb);
   RotateMotor (ma,75,90);  //gira al frente cuello con la cabeza en posicion volteada hacia
la izquierda
   Wait(2000);
}




// Procedimiento que detiene al robot.
void detiene ()
{
 Off(mc);
 Off(mb);
}

// Procedimiento sensa el sonido, pero no fue utilizado, se muestra el código para fines
// didácticos.
void sonido()
 {
  int x;
  SetSensorSound(IN_1);
  x=Sensor(IN_1);
   // Inicia detectando sonido, avanza hacia adelante
  if (x>30)
  {

    OnFwd(mb,75);
    OnFwd(mc,75);
```

```
   Wait(1000);
   }
  else
  {
  detiene ();
   }
  }

 void girari()
{
  RotateMotor (mb,75,560);
  Wait (1000);
  detiene();
 }

 void girard()
{
  RotateMotor (mc,75,560);
  Wait (1000);
  detiene();
 }

// Procedimiento que detecta objetos frente al robot.
void objeto ()
 {
     //Defino variable que tendra el valor del sensor US
      int obj;
      // Defino el sensor en la entrada 4 ( Ultrasonidos entrada 4)
       SetSensorLowspeed( IN_4);
      while (1)
      {

        //Capturo el valor del sensor en la variable obj
         obj = SensorUS(IN_4);
         // Escribo en pantalla su valor
        // TextOut(2,0, "S2:",FALSE) ;  NumOut(20,2, obj, FALSE);
         if( obj >15)
             {
                OnFwd(mc,75);
                OnFwd(mb,75);
                Wait (1000);
            }
            else
            {
            detiene();
            OnRev(mc,75);
            OnRev(mb,75);
            Wait(1000);

            }
    }
```

```
}

// Procedimiento que hace avanzar al robot hacia delante.
void avanza()
  {
    objeto();
    OnFwd(mb,75);
    OnFwd(mc,75);
    Wait(1000);
    }

// Procedimiento que detecta sonido y posteriormente hace que los servomotores trabajen
// dependiendo del resultado.
void detecta()
{
 int f, d, i,obj;
 SetSensorLowspeed( IN_4);
 SetSensorSound(IN_1);
 obj = SensorUS(IN_4);
 TextOut(2,0, "S2:",FALSE) ;  NumOut(20,2, obj, FALSE);
 gira_ad();
 i= Sensor(IN_1);
 gira_ai();
 d=Sensor(IN_1);
 gira_af();
 f=Sensor(IN_1);
   {
   if (i>d)
     if (i>f)
       girari();
     else
        if(obj>15)
            {
             OnFwd(mc,75);
             OnFwd(mb,75);
             Wait (1000);
            }
         else
            {
            OnRev(mc,75);
            OnRev(mb,75);
            Wait(1000);
            }
   else
if (d>f)
        girard();



else
        if( obj>15)
```

```
          {
            OnFwd(mc,75);
            OnFwd(mb,75);
            Wait (1000);
           }
         else
           {
           //detiene();
           OnRev(mc,75);
           OnRev(mb,75);
           Wait(1000);
           }
 }
}

// Este procedimiento no se utilizo sin embargo se muestra para fines didácticos.
void explora()
{

 while(1)
 {
  gira_ad();
  Wait (1500);
  gira_ai();
  Wait (1500);
  gira_af();
  Wait (1500);
  avanza();
  Wait (2000);
 }
}



// Principal, como se observa solo se manda llamar un procedimiento detecta.
task main()
{
 int w;
 SetSensorTouch(IN_2); // Estas líneas hacen que funcione el programa hasta que se
 while(!Sensor(IN_2))      //presione el botón de tacto.
   {
    detecta();
   }
}
```

 **Sugerencias:**

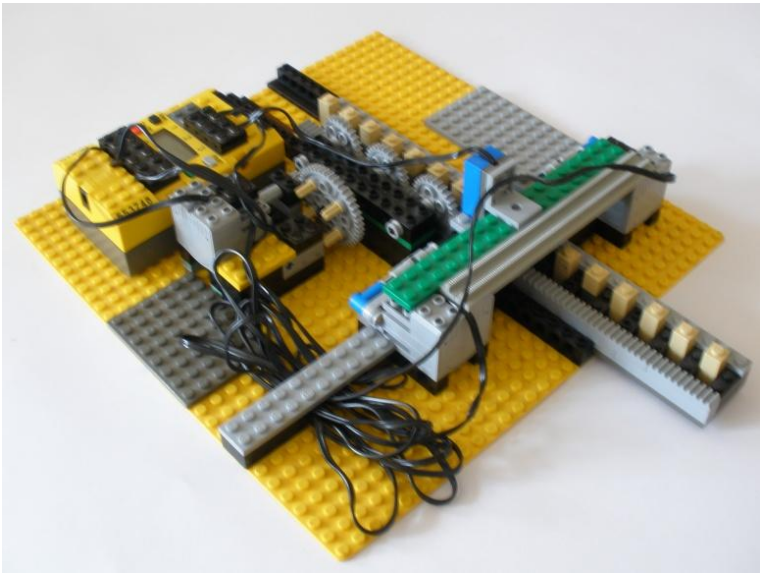
Con esta construcción y el código presentado se pueden realizar diversas prácticas como por ejemplo:

- **Pulir los movimientos del robot.**
- **Realizar prácticas de navegación de IA. Hacerlo que se dirija de un punto a otro evadiendo obstáculos.**

Resumen

ChalanBot es un robot muy didáctico desde su diseño y construcción. Debido a su particular imagen y su función principal de ser un robot servicial, es ideal para hacer prácticas en NXC. Es muy probable que las personas que hagan uso de este robot, sea el detonante para crear nuevas ideas.

Maquina de Turing



Introducción

El abuelo de las computadoras, podríamos nombrar así a este dispositivo tan importante llamado Maquina de Turing (MT). Considero que uno de los dispositivos que marco la era de la computación fue el modelo matemático de Alan Turing, capaz de realizar cualquier calculo. La Máquina Turing es una clase muy primitiva de computadora, muy simple, sin embargo capaz de realizar cualquier cálculo, y un antepasado de las computadoras modernas que usamos todos los días.

El robot que se muestra en el siguiente manual no es una construcción propia, la idea original es de Giulio Ferrari, quien hizo posible la idea de hacer tangible un modelo imaginario en físico. Esta construcción aparece en el libro llamado *Lego Masterpieces* en el cual hay más construcciones con el RCX de diferentes autores, todos programados en NQC y el cual es muy recomendado para hacer trabajos muy profesionales con piezas Lego. A continuación se muestran los datos del libro que es recomendado adquirir *Lego Masterpieces por Giulio Ferrari. Editorial Syngress. ISBN: 1-931836-75-2. Año 2003.*

Para comprender el mejor funcionamiento de la maquina es necesario tener conocimientos en Teoría de la Computación, para obtener el mejor provecho de ella, y poder realizar infinidad de ejercicios de computabilidad. Esta construcción muestra el código de un ejercicio que realiza la maquina: La suma de dos números enteros, programa realizado por Giulio Ferrari. Es importante señalar que se agrego esta construcción Lego para observar el potencial de lo que se puede crear con un kit Lego. Y con esto despertar el interés de diversos profesores y alumnos a nivel universitario, que elaboren proyectos a este nivel.

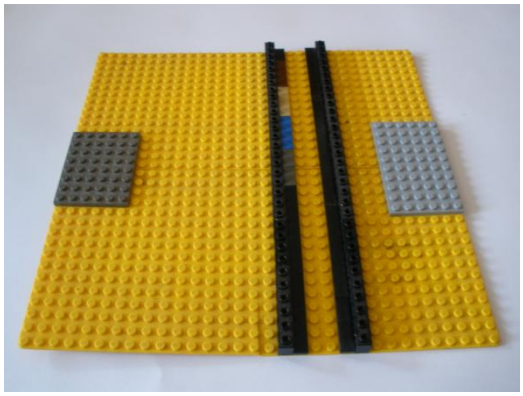
Historia de la Maquina de Turing

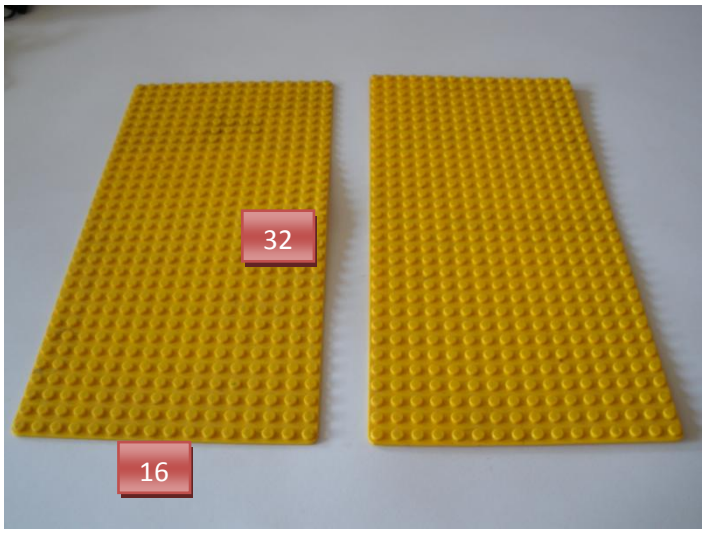
En 1936, Alan Turing contestó al entscheidungsproblem, la cuestión planteada por David Hilbert sobre si las matemáticas son decidibles, es decir, si hay un método definido que pueda aplicarse a cualquier sentencia matemática y que nos diga si esa sentencia es cierta o no. En el artículo On Computable Numbers, Turing construyó un modelo formal de computador, la Máquina de Turing, y demostró que había problemas tales que una máquina no podía resolver. La máquina de Turing es el primer modelo teórico de lo que luego sería un computador programable. Con el tiempo a este tipo de máquina se la conoció como máquina de estado finito, debido a que en cada etapa de un cálculo, la siguiente acción de la máquina se contrastaba con una lista finita de instrucciones de estado posibles.

Maquina de Turing Diseño y Construcción

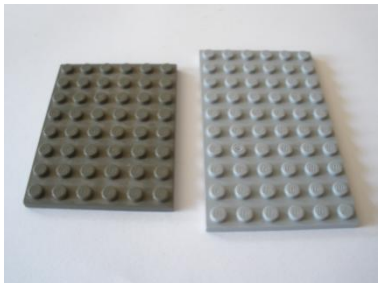
El siguiente diseño es una idea original de Giulio Ferrari, a continuación se muestra su construcción paso a paso.

Paso 1: La base

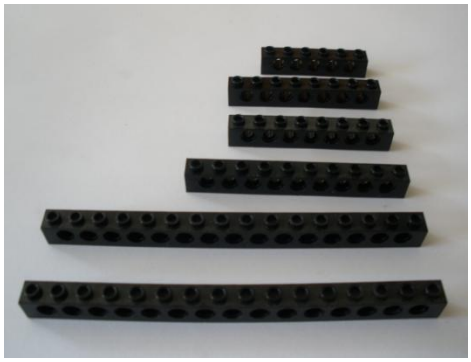




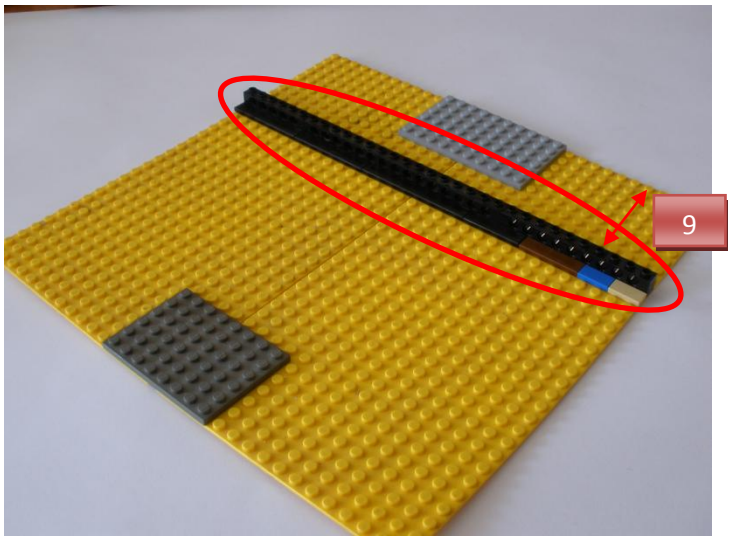
Conseguimos estas piezas para unir las tabletas de 16x32:



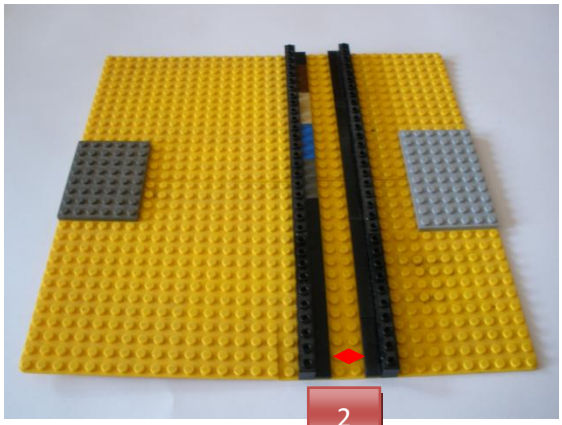
También requerimos las siguientes piezas para completar la base:



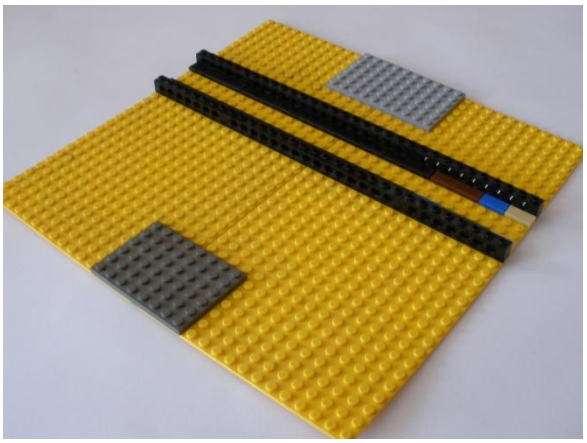
Las piezas pueden variar, como se muestra no hacen algo muy indispensable, solo las piezas que van debajo donde se desplaza la cinta.



Dejamos dos espacios y colocamos en frente las piezas similares.

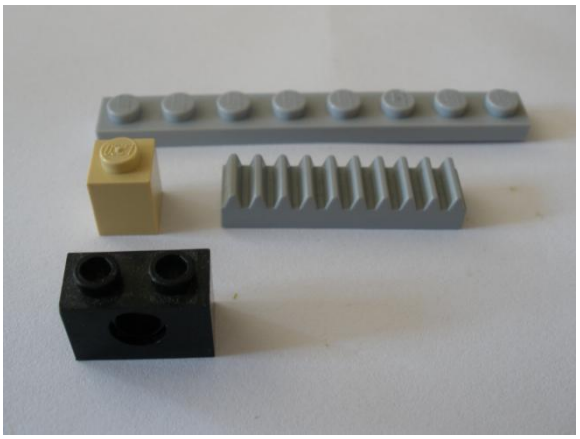


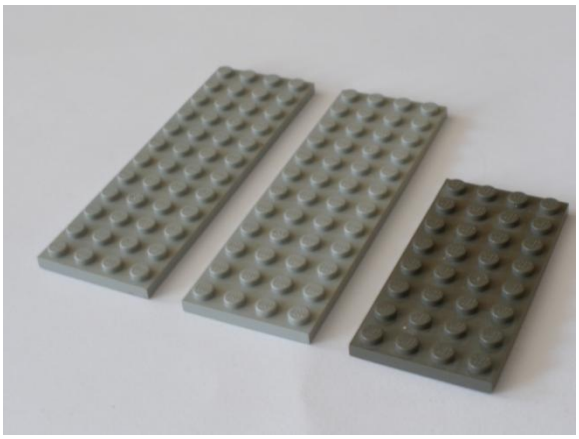
Esta es nuestra base terminada.



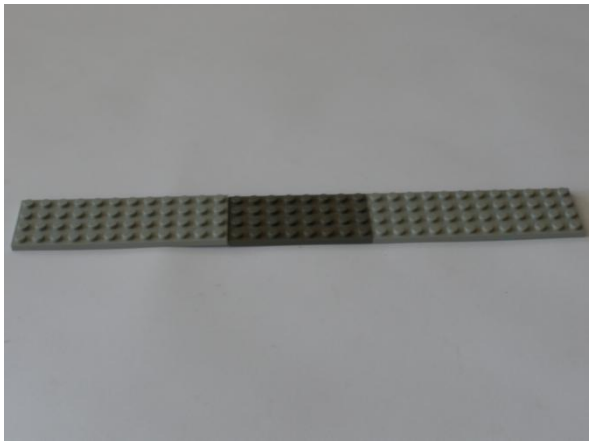
Paso 2: La cinta

Material necesario.

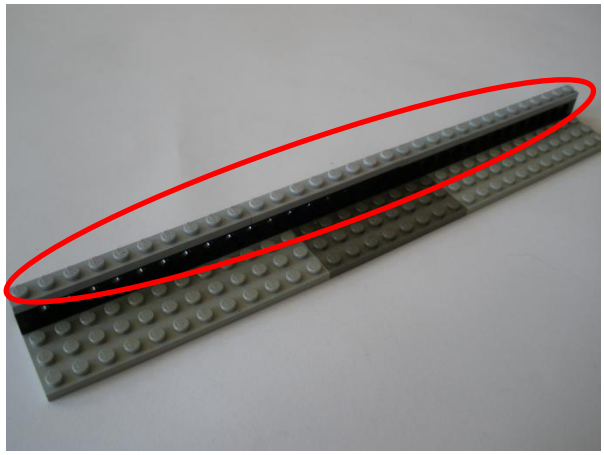


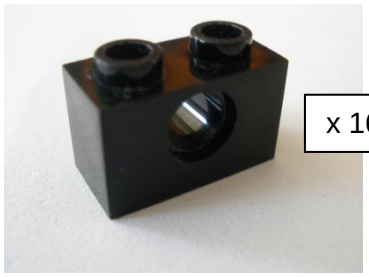


Las piezas se ensamblan de la siguiente manera:



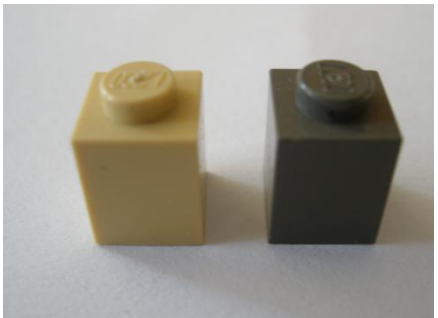
A continuación se unen las siguientes piezas en la cinta.

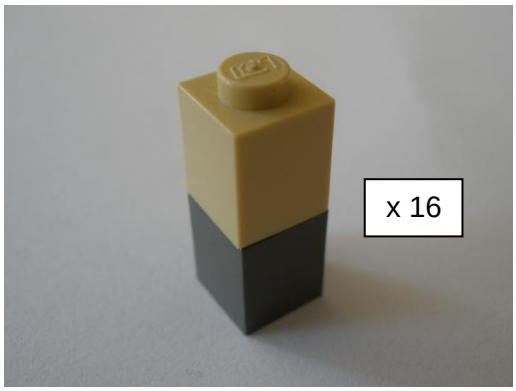




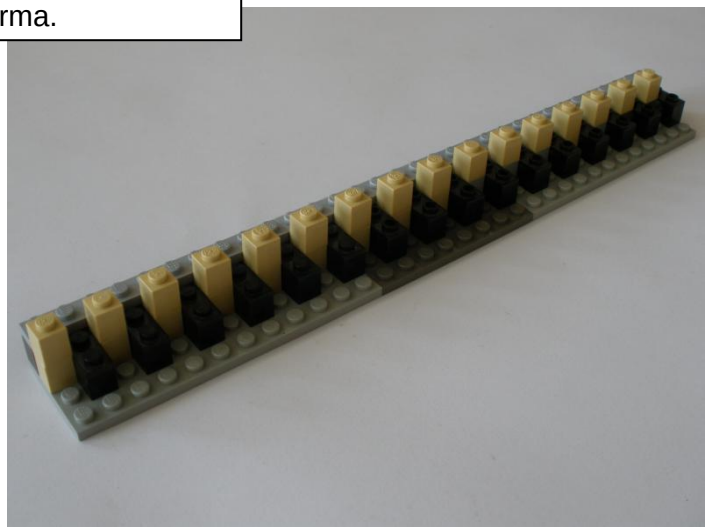
x 16

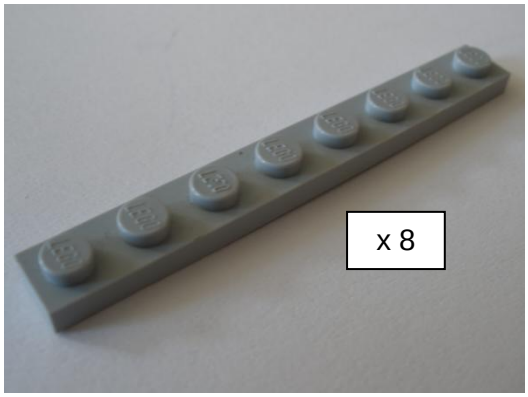
Insertamos las piezas en la cinta intercalándolas con un espacio





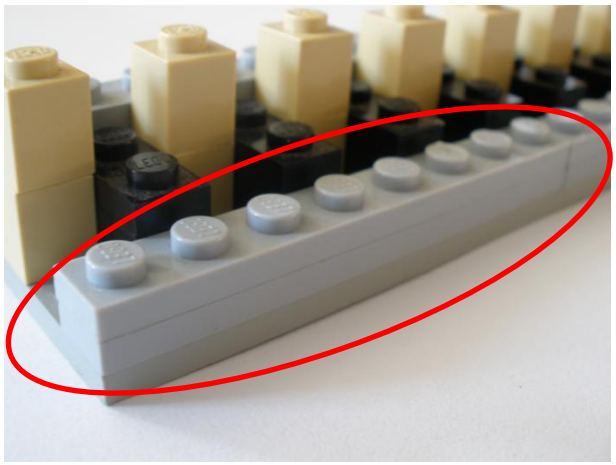
Quedaría de la siguiente forma.



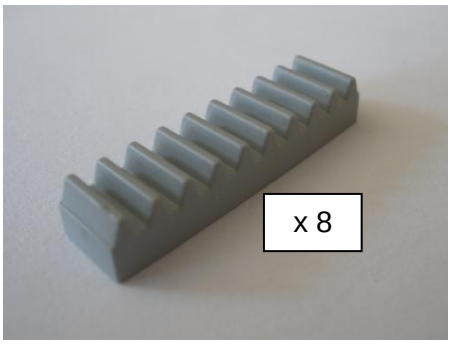


La colocamos en la otra orilla.

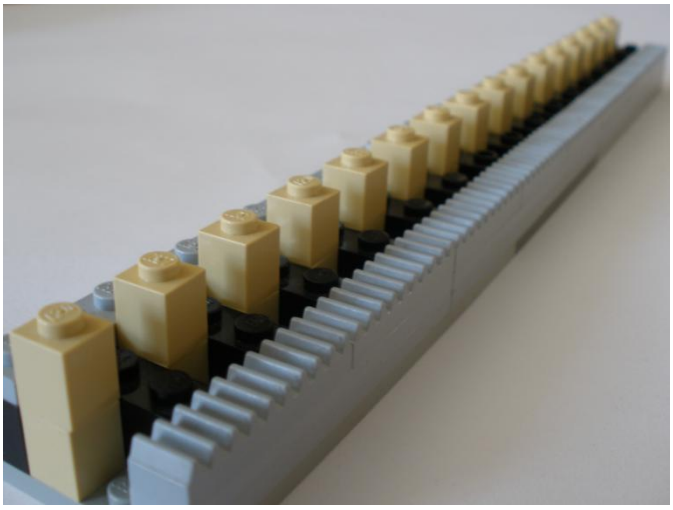
Colocamos nuevamente otra fila. Encima de la última que agregamos para que los engranes la puedan manipular.



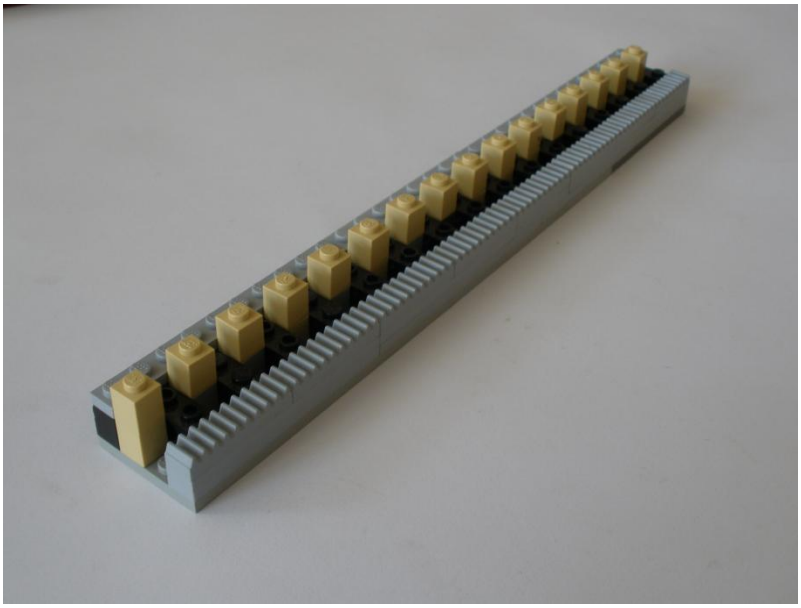
Tomamos la siguiente pieza que servirá como cremallera.



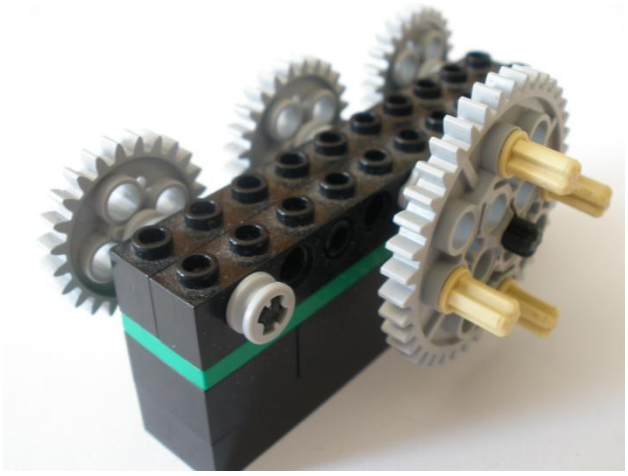
Las colocamos a lo largo de la última fila que agregamos.



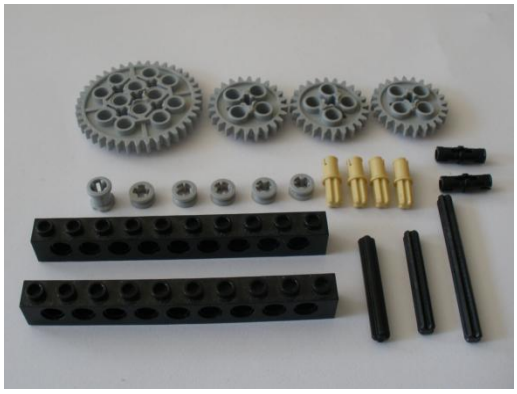
Así quedaría nuestra cinta ya terminada. Esta cinta tiene algunas variantes de la original de Giulio Ferrari, ya que se requieren piezas adicionales a un kit RCX.

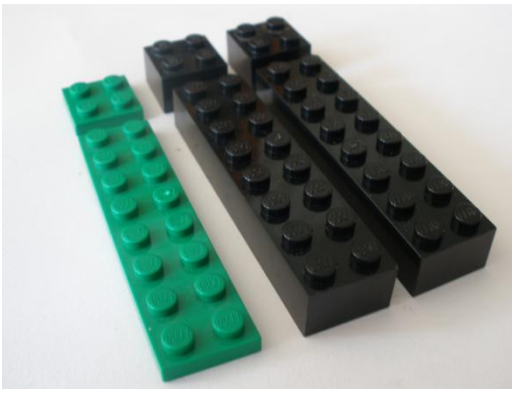


Paso 3: Control de Dirección

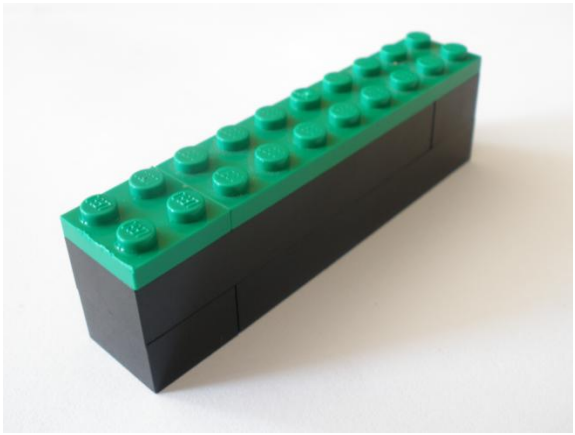


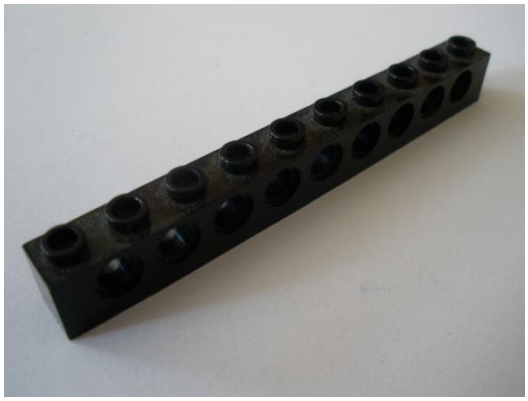
Piezas a utilizar:



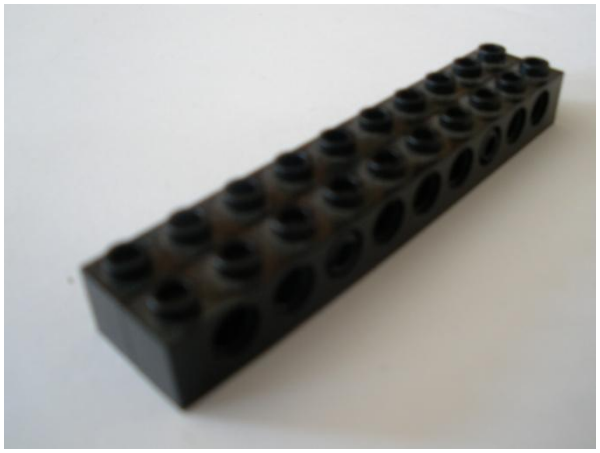


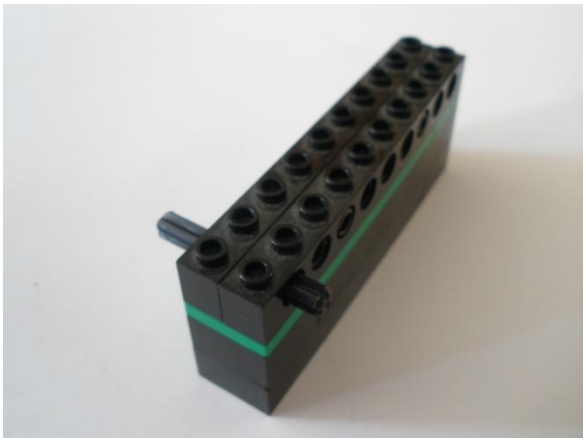
Con las últimas piezas mostradas armamos la siguiente figura:

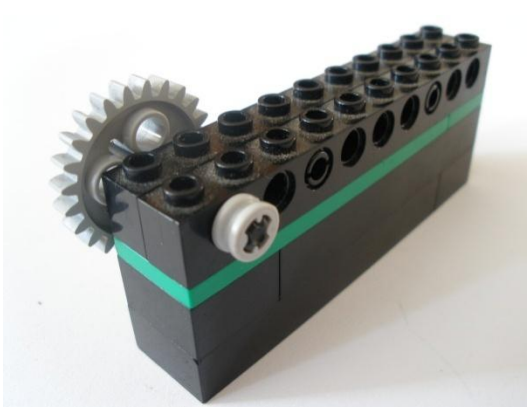


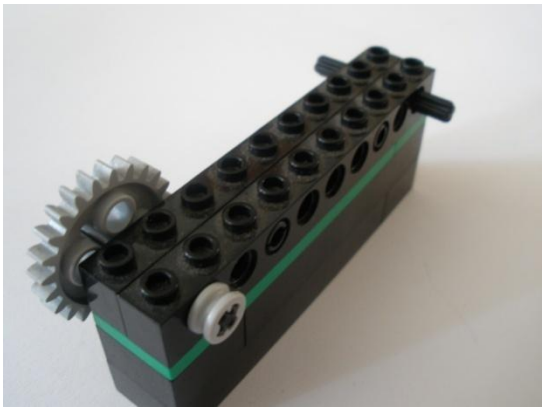


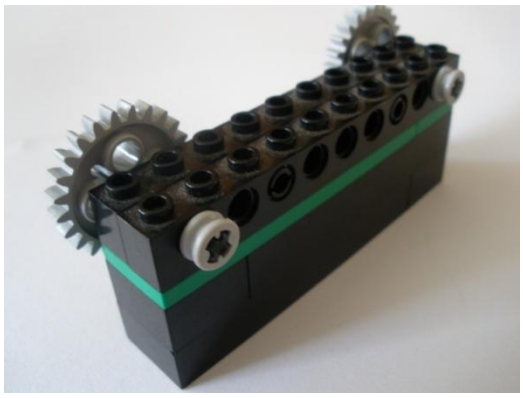
Insertamos estas dos piezas.

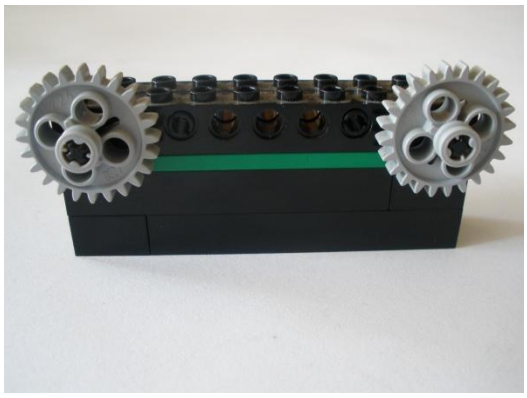


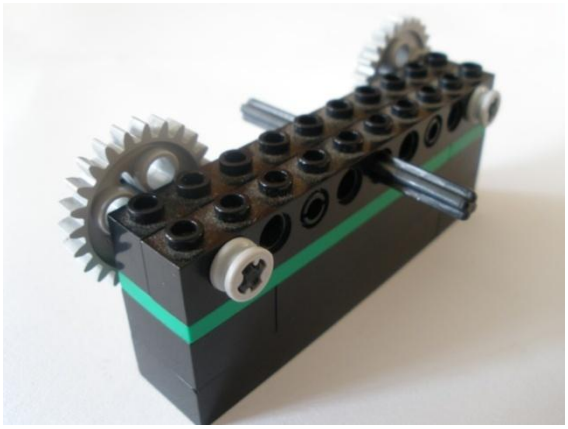


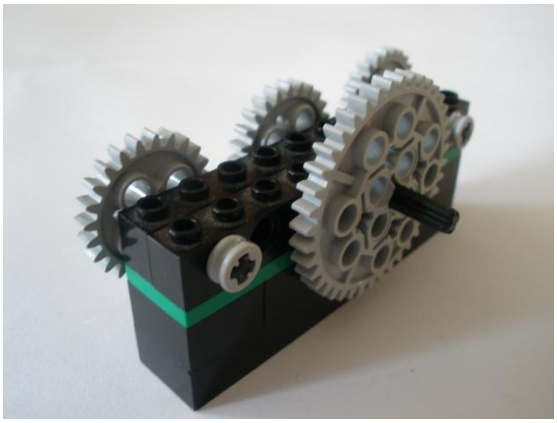




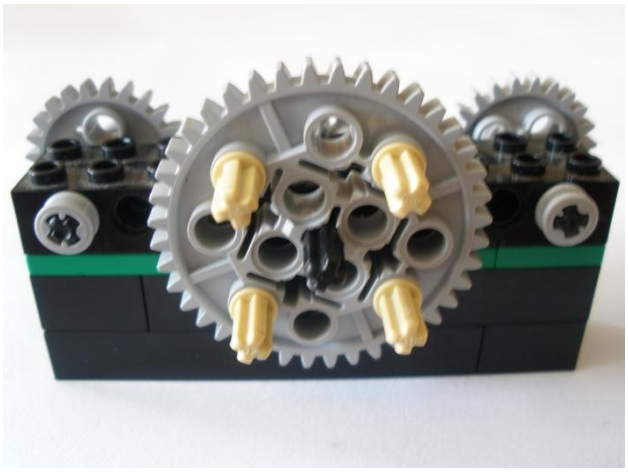




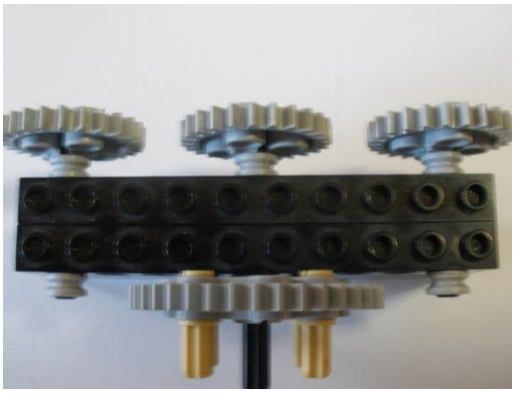




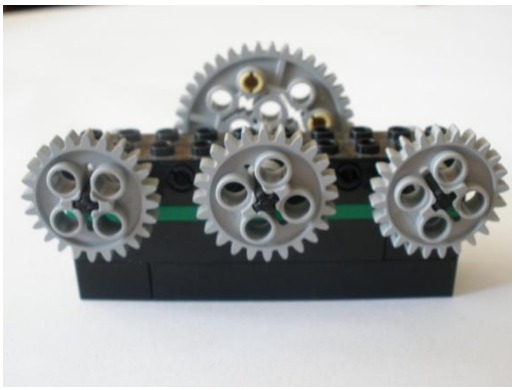
Control de dirección terminado.



Vista superior

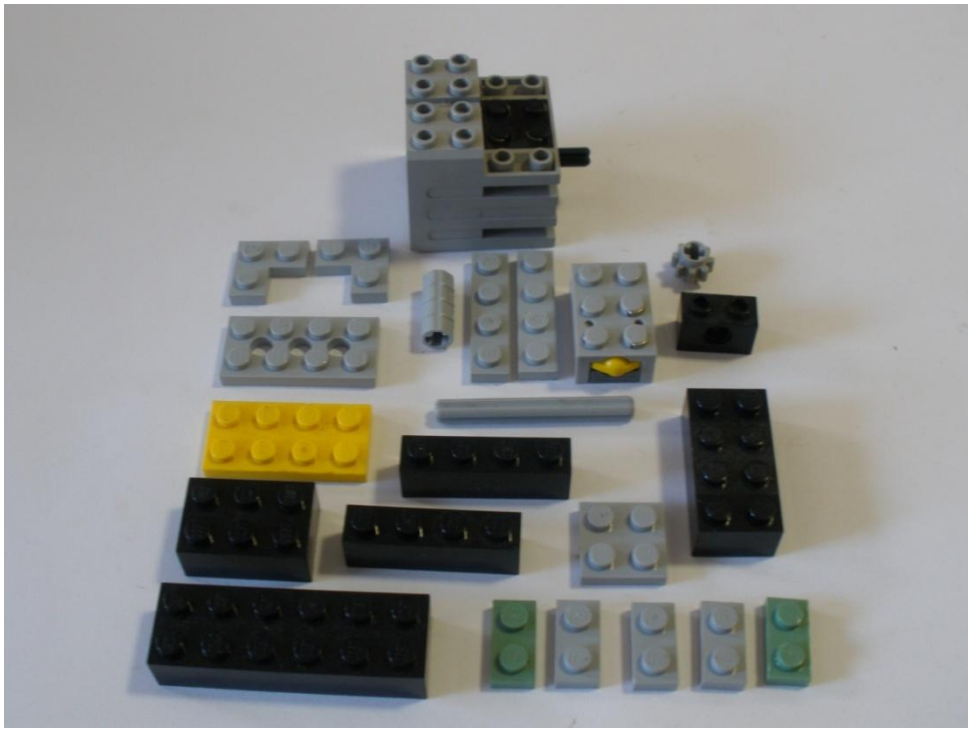


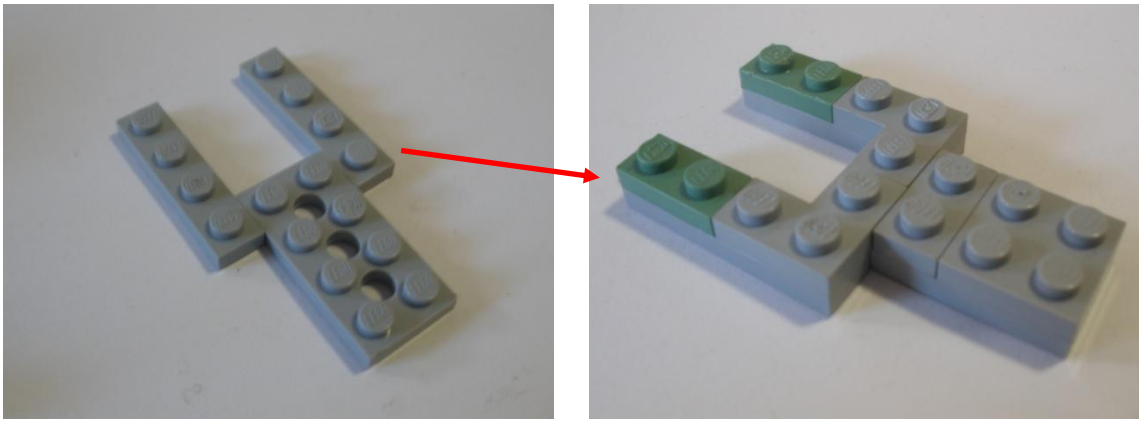
Vista posterior

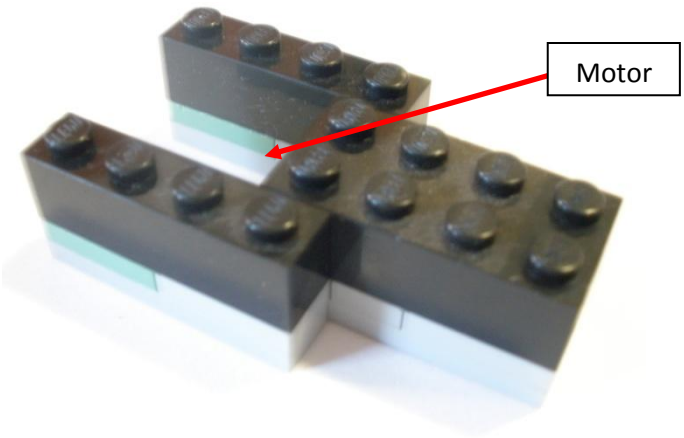


Paso 4: Control de dirección II

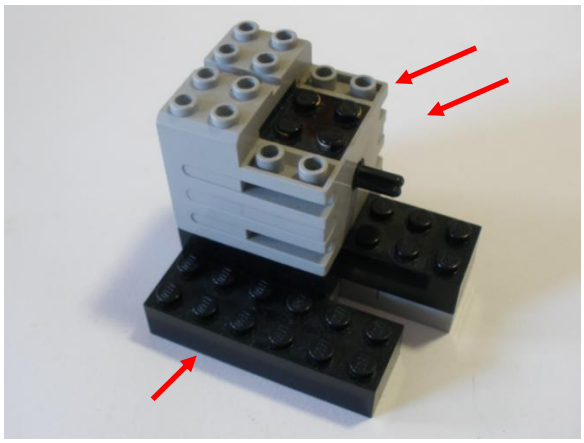
Piezas a utilizar para ensamblar el control de dirección II.



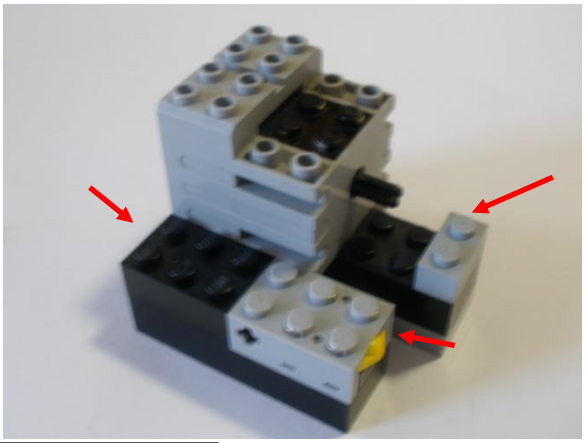




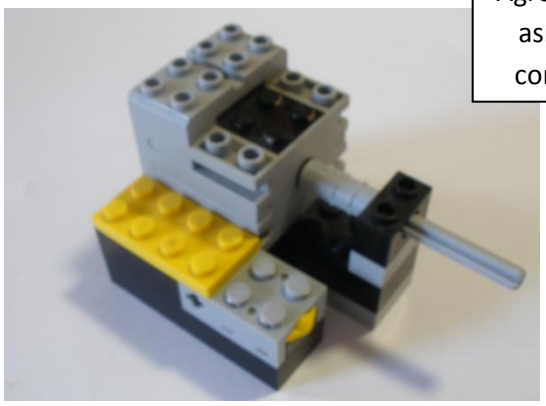
En esta pieza se monta el motor y se coloca otra del lado izquierdo.



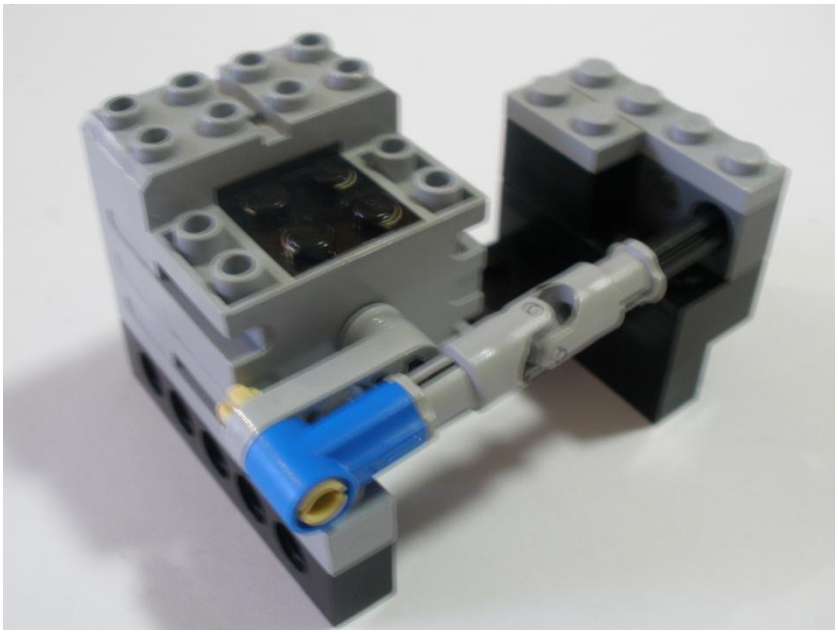
Se instala el sensor de tacto



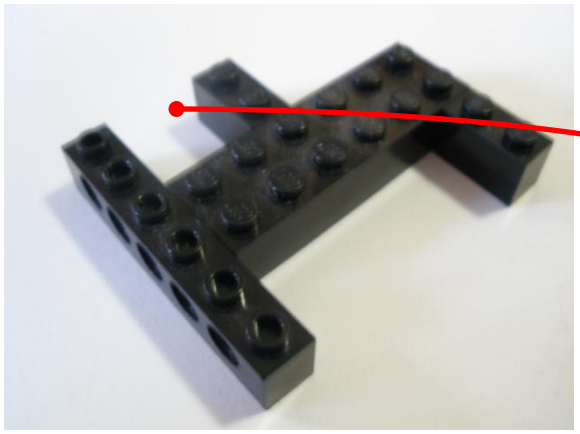
Agregamos un engrane y así concluimos con el control de dirección II

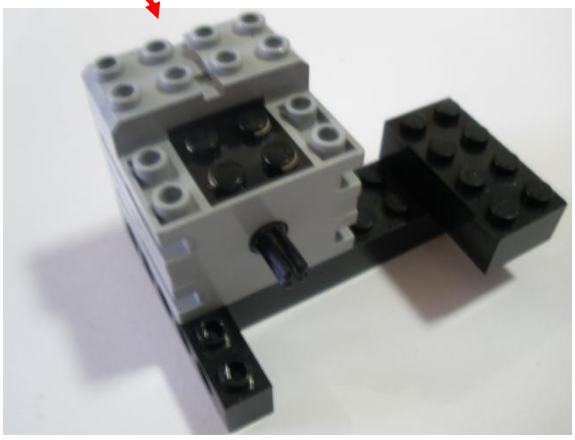


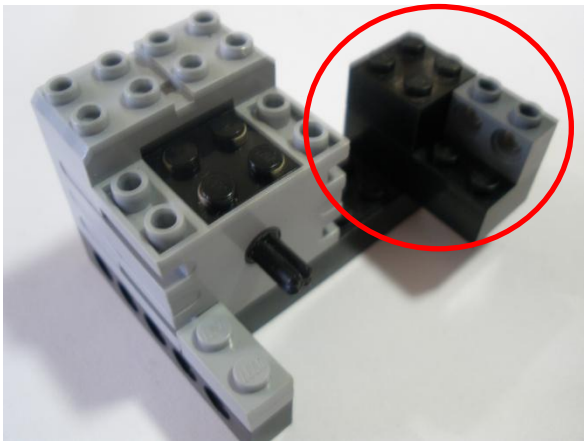
Paso 5: Switch de borrado

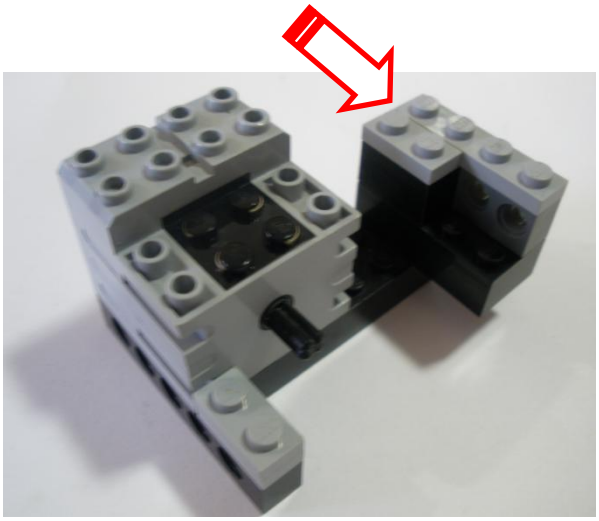


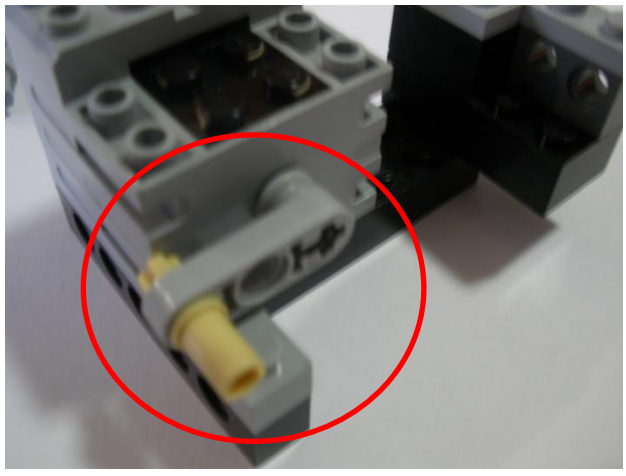
Piezas a utilizar:

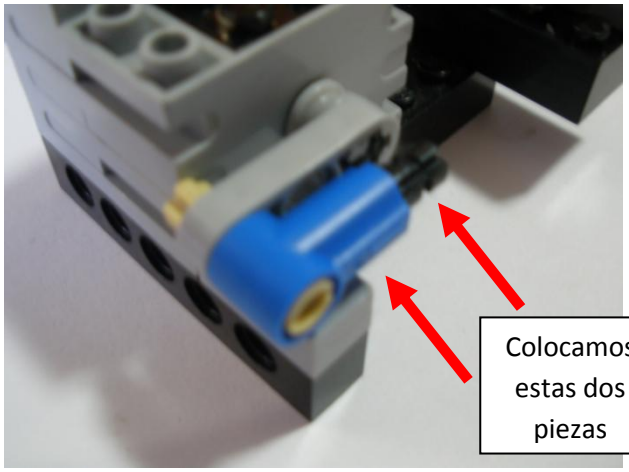




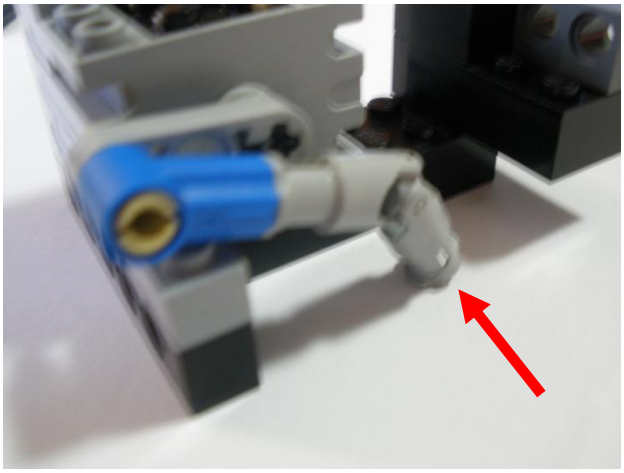


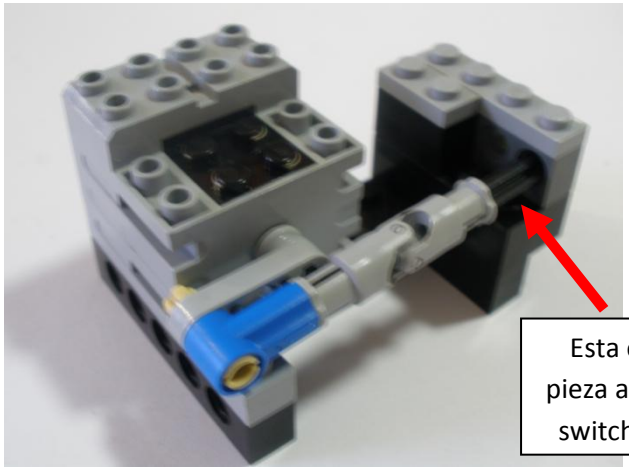






Colocamos estas dos piezas

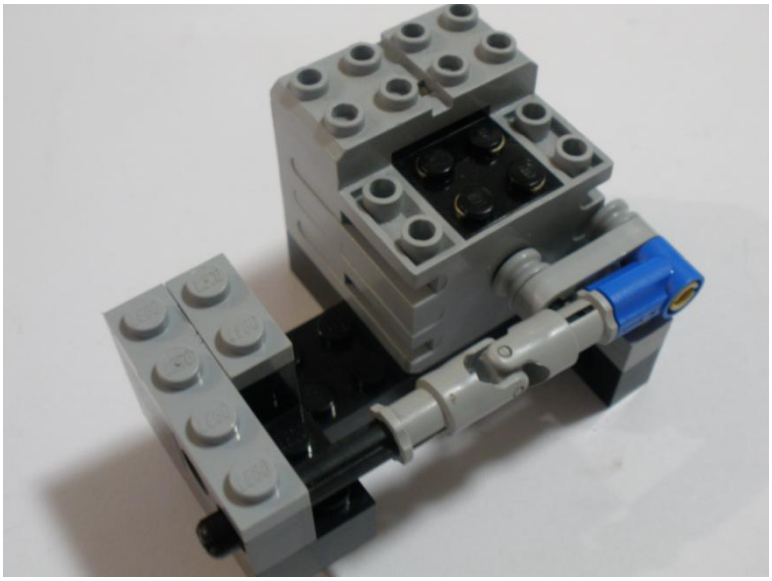




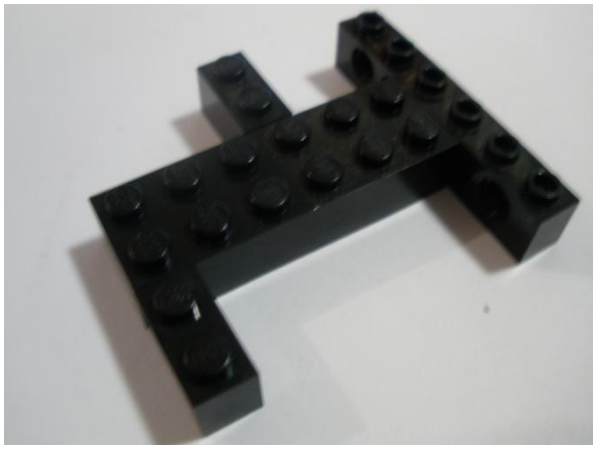
Esta es la ultima pieza a colocar en el switch de borrado

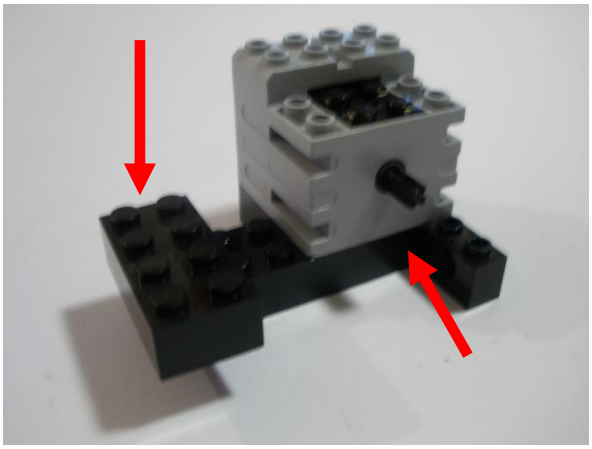
Paso 6: Switch de escritura

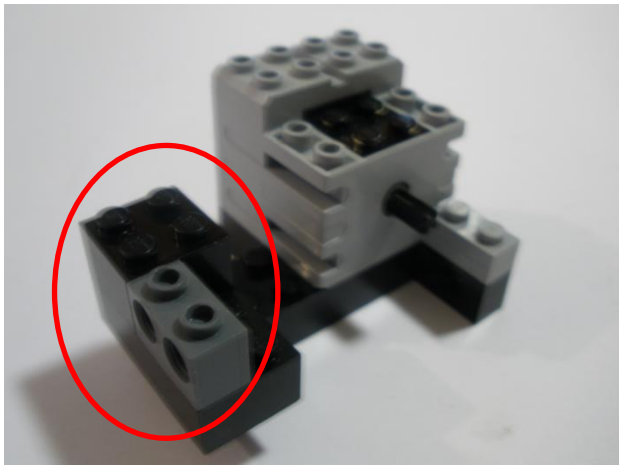
A continuación se muestra la pieza a ensamblar:

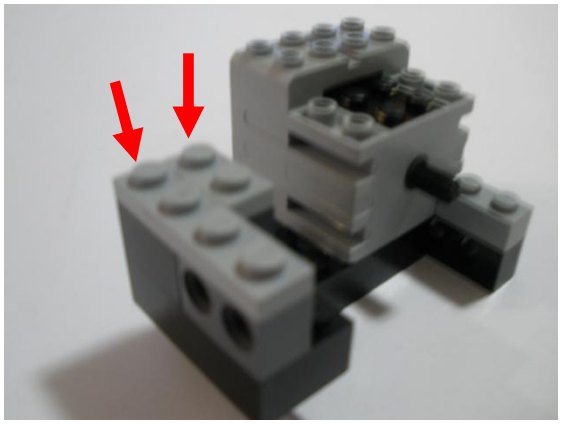


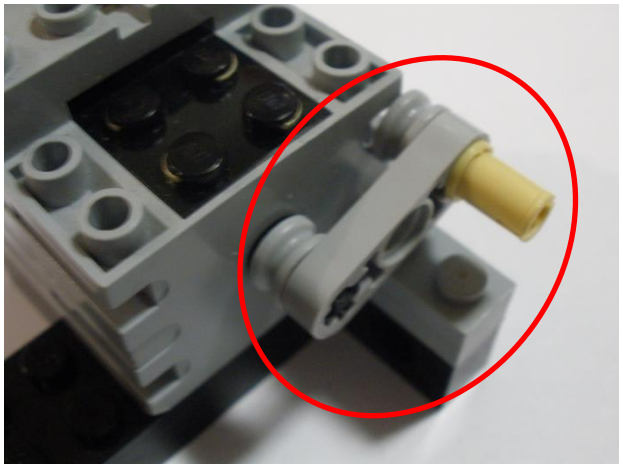
Piezas necesarias:

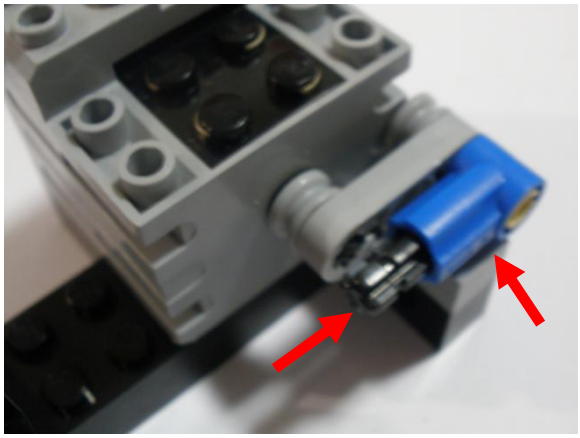


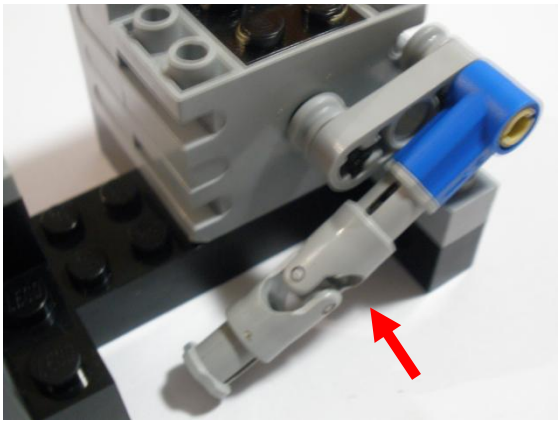




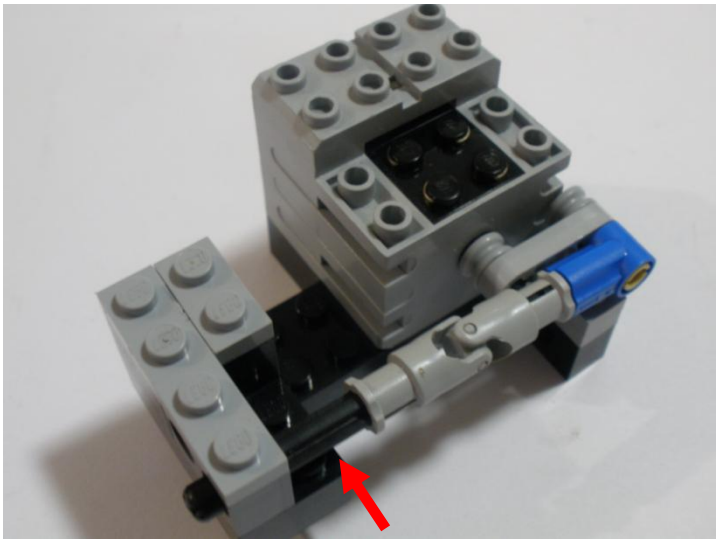






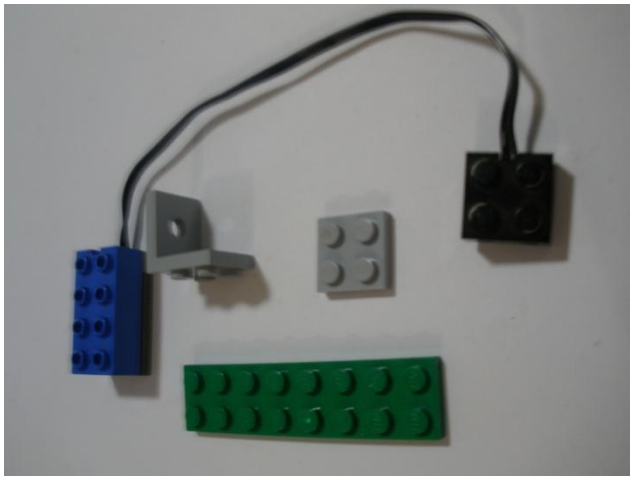


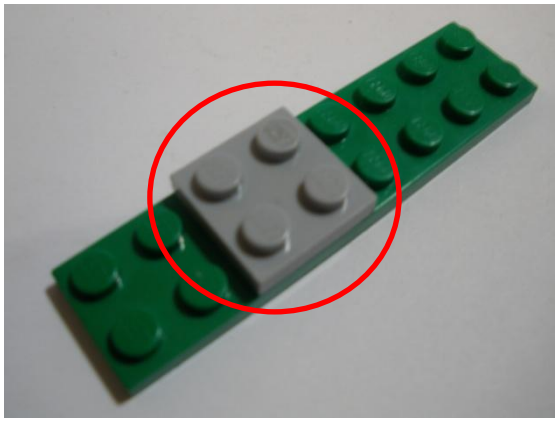
Con esta última pieza concluimos el ensamble del switch de escritura.

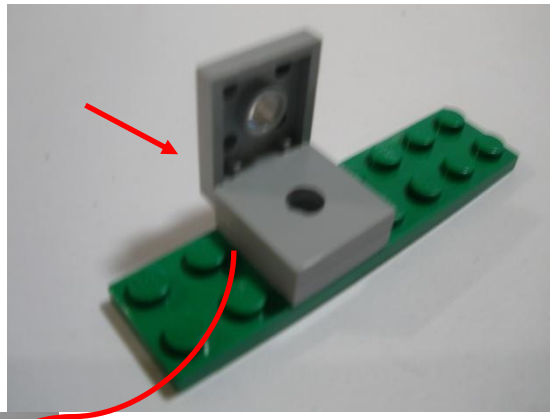


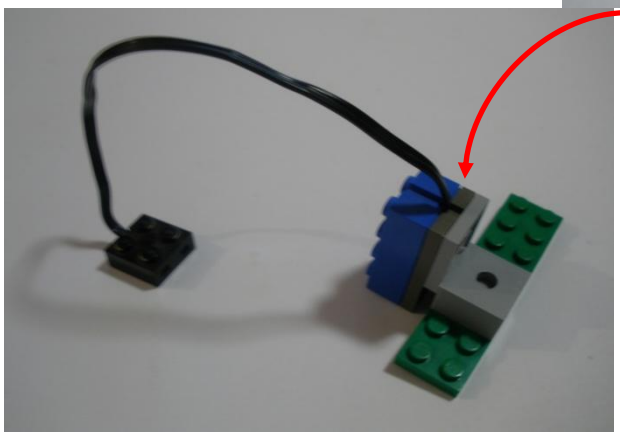
Paso 7: Sensor de luz

Piezas para armar el sensor de luz.

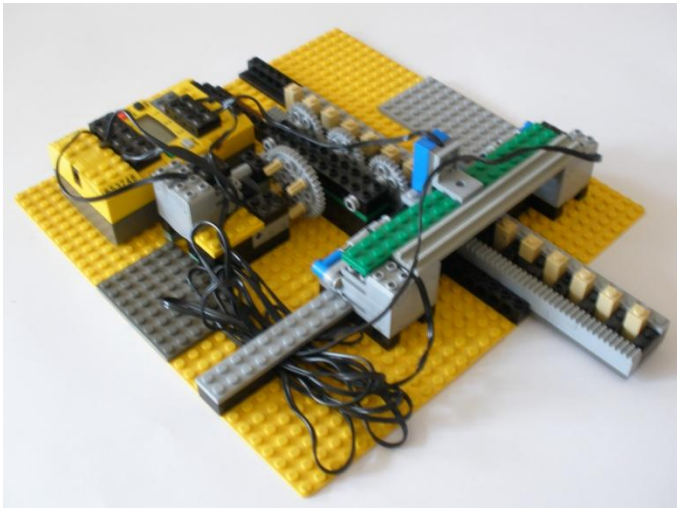




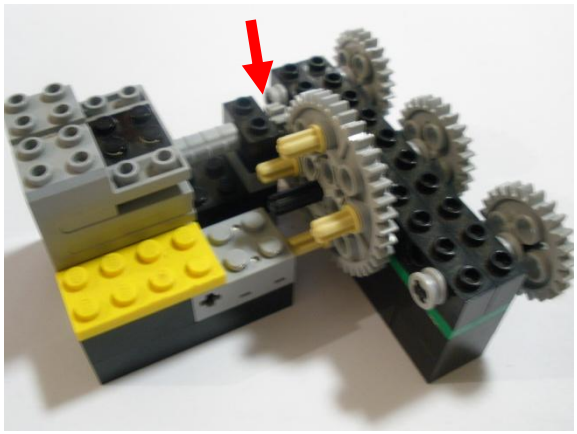




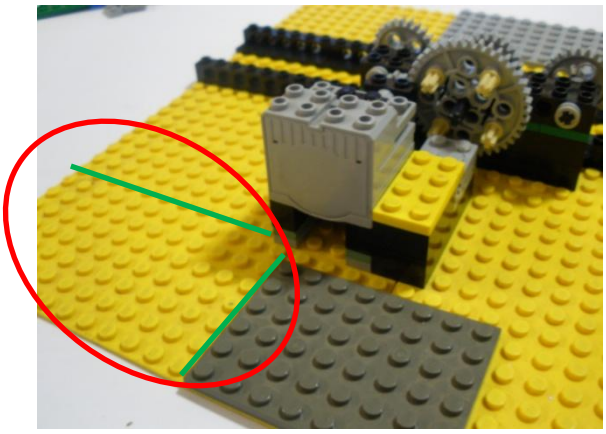
Paso 8: Ensamble de todos los componentes

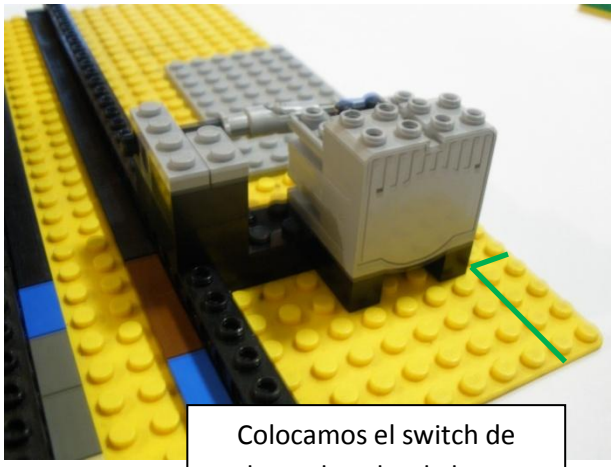


Ensamblamos los dos controles de dirección de la siguiente manera:

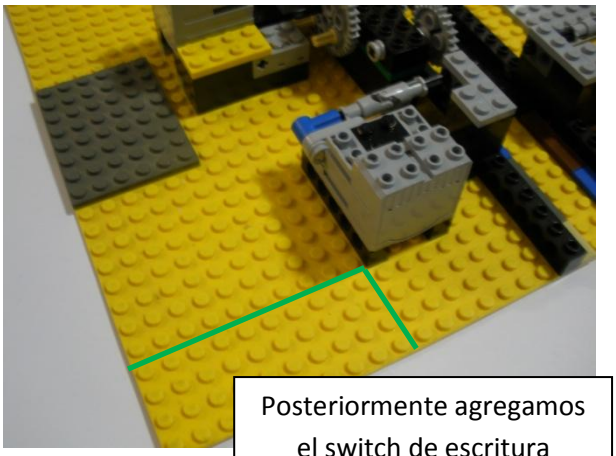


Colocamos los controles de dirección en la base, tomando en cuenta la posición vista en la figura.

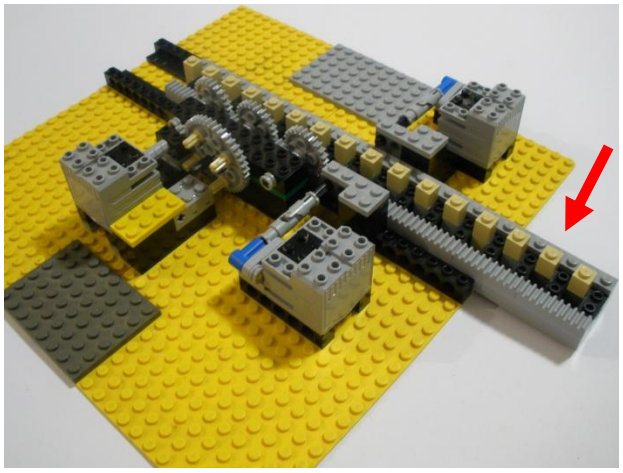


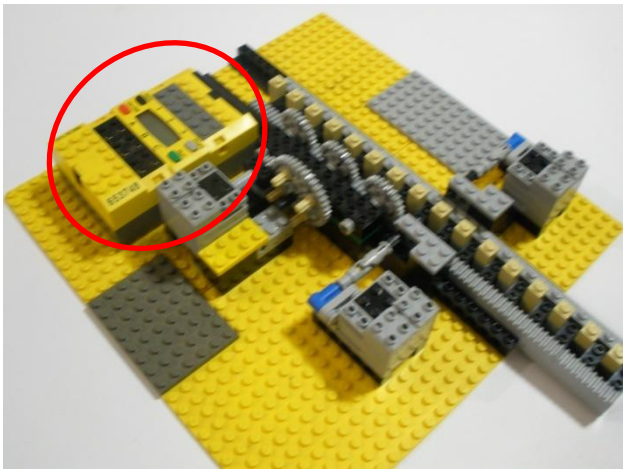


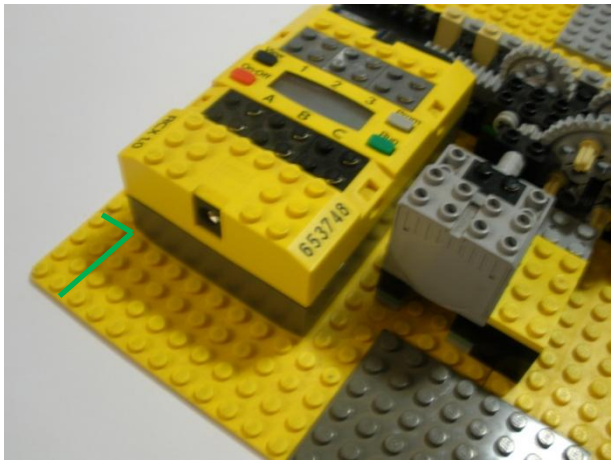
Colocamos el switch de borrado sobre la base



Posteriormente agregamos el switch de escritura

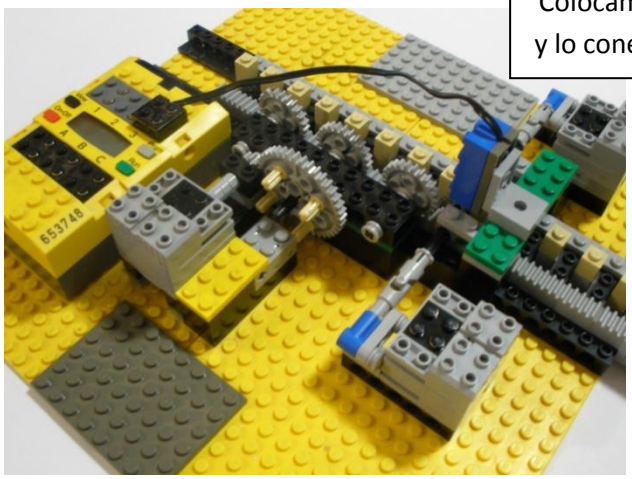




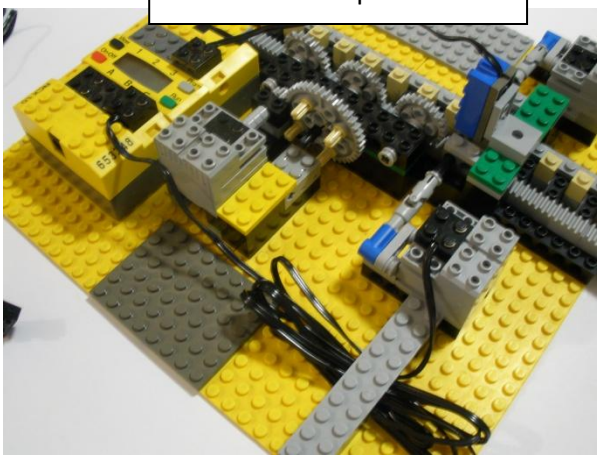


Colocamos el sensor de luz y lo conectamos al “puerto 3”

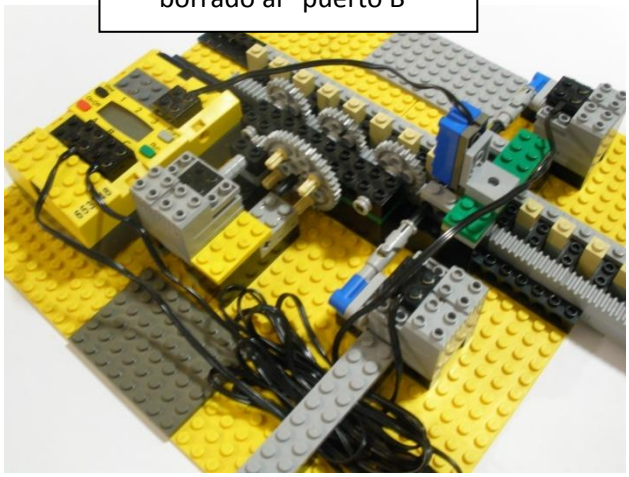
Colocamos el sensor de luz y lo conectamos al puerto 3



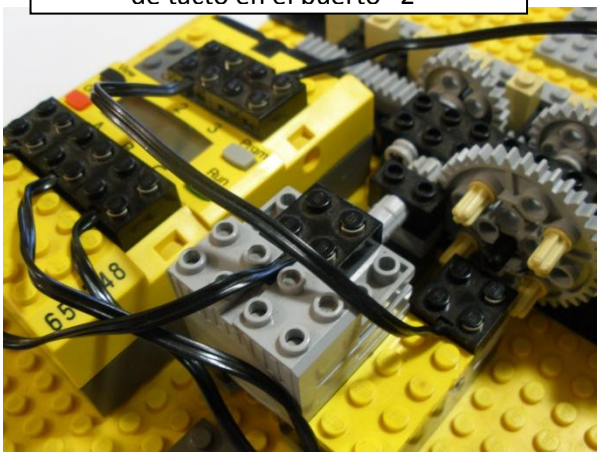
Conectamos el switch de escritura al “puerto C”

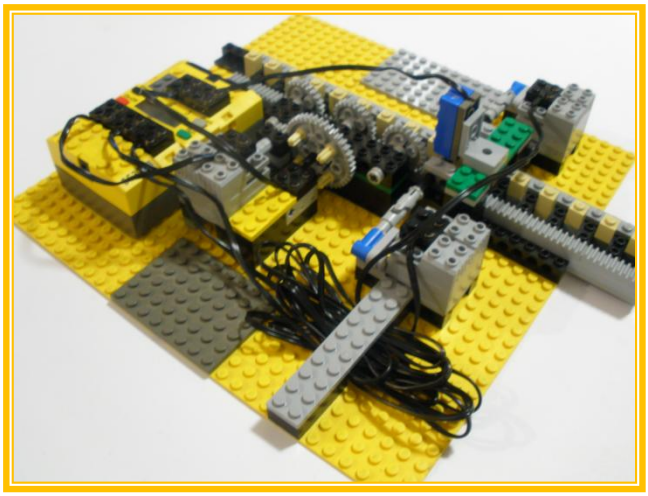


Conectamos el switch de borrado al “puerto B”



Por ultimo conectamos el control de dirección en el puerto “A” y el sensor de tacto en el puerto “2”





Con esto concluimos la construcción de nuestra maquina y quedaría así.

Maquina de Turing Original

El examinar toda y cada parte de la Maquina de Turing original nos ayudara a conocer el papel preciso en la estructura.

La primer y principal parte de la Maquina de Turing original consiste en una cinta de longitud infinita, dividida en celdas. Cada celda tiene espacio suficiente para escribir un símbolo. Una cabeza que lee y escribe en la cinta que puede moverse por una celda a la vez, moviéndose de izquierda derecha. Puede ver, que la maquina de Turing es una maquina primitiva que puede manejar operaciones muy básicas como mover y escribir símbolos.

El dispositivo por sí mismo está siempre de uno a un número finito de estados. Estos estados determinan la acción que desempeñara la maquina en una forma especial de programación, llamada tabla de estados de transición (o tabla de acción). La tabla de transición dice a la maquina de turing que hacer cuando se encuentre en cierto estado, y encuentra un símbolo especifico, similar como cuando el estado de su mente le dice que hacer en cierta situación para responder a acontecimientos específicos.

La máquina puede realizar cuatro acciones:

- Ir a la izquierda de una celda
- Ir a la derecha de una celda
- Escribir un símbolo
- Borrar un símbolo

En síntesis, usted puede pensar en la Máquina Turing como “una caja negra” que es capaz de leer, escribir, y borrar símbolos en una cinta larga dividida en celdas. Observe que mientras la cinta sea infinita, los estados y los símbolos son finitos. Puede ver

fácilmente que no hay otro lugar para almacenar datos que la cinta, por lo tanto para operaciones más complejas (como multiplicar dos números) tiene que agregar placeholders en la cinta sumando los números mismos. De hecho, tiene que guardar información de lo que esta pasando, que va a pasar y datos temporales, para cada simple operación muy simple, los estados crecen a un numero grande rápidamente.

La tabla de transición

Por conveniencia, la maquina empieza en el primer estado de la primera celda mas lejana a la izquierda que contiene el símbolo. Entonces busca en la cinta moviéndose de derecha a izquierda, y moviendo los datos para leer, escribir y borrar los símbolos. La tabla le dice a la maquina que el equivalente a “si usted está en cierto estado y ve cierto símbolo, haga una acción específica”, se mueva de celda y cambie de estado. Veamos este proceso en una secuencia.

1. La máquina lee contenido de una celda, es decir el símbolo escrito en la cinta.
2. La máquina usa la tabla de estado de transición para trazar un mapa, de entrada (símbolo + estado) a la salida (varias acciones).
3. La máquina cambia el símbolo en la cinta, lo borra, o lo deja como esta.
4. La máquina mueve la cinta de izquierda a derecha.
5. La máquina puede o no cambiar su estado interno a un nuevo estado.
6. Si la maquina encuentra el estado "de parada", la máquina deja de operar; sino, el proceso inicia nuevamente en el Paso 1.

Todas las decisiones están basadas en su estado actual, es decir, en la tabla de estado de transiciones, la cual determina el comportamiento de la maquina. La primer parte

indica la situación del dispositivo, es el estado y el símbolo que determina el estado actual de la celda. La segunda parte de la tabla contiene para las operaciones de la maquina incluyendo que se escribe (si hay algo), hacia donde ir y que cambiar.

Este es un esquema del contenido de una línea de la tabla (la Figura 3.1):

Figura 3.1

State	Read Symbol	Operation	Move Direction	Next State

Como puede observar, el programar una maquina Turing es un poco difícil porque la posibilidad de abstracción es mínima, que hace las funciones complejas muy difícil de implementar.

Suponga que quiere sumar dos números enteros. Esto se puede hacer dependiendo de las restricciones en el número de símbolos y los estados que están permitidos utilizar, en numerosos caminos diferentes. Uno de los métodos más simples es representar un número entero en la cinta (que esta vacía) por el número de celdas que esta ocupada por un símbolo, en este caso un “o”. Una celda vacía significa que esta en blanco, es decir, no hay símbolos escritos en ella.

La Figura 3.2 nos muestra la representación del número 3 en la cinta:

Escribimos en la cinta los dos símbolos que queremos sumar (por ejemplo, los números 3 y 4), separado por una celda vacía (la Figura 3.3).

Figura 3.3 Sumando tres y cuatro.

Nuestro objetivo es, los posibles pasos, el resultado es la suma (3+4=7), como puede ser visto en la Figura 3.4.

La figura 3.4 el Resultado de la suma Tres y Cuatro

La maquina debe comenzar en el primer símbolo en la izquierda, suma los dos enteros, y se detiene después de mostrar la respuesta requerida. La tabla 3.1 muestra como la tabla de estado de transición podría estar hecha.

Tabla 3.1. Tabla para sumar dos enteros.

Input		Output		
State	Read Symbol	Operation	Move Direction	Next State
0	o		Right	
0	<Empty>	Write: o	Right	1
1	o		Right	
1	<Empty>		Left	2
2	o	Erase	Stop	

De este modo hemos creado una maquina que es capaz de sumar cualquier numero entero, no importa cual sea el tamaño (suponiendo que Ud. tiene una cinta infinita). La entrada es determinada por la combinación del estado actual (indicado como un número) y el símbolo. En este caso solo dos “símbolos” son utilizados una celda puede estar vacía o llena. Note que cuando la celda de la tabla esta vacía significa que no hay acción a tomar, significa que un símbolo no esta reescribe si ya esta ahí. Del mismo modo, si el siguiente estado no esta especificado, que no necesita cambios, y la maquina simplemente se queda en el estado actual. Es interesante ver como el proceso entero trabaja, por examinar la cinta en cada paso (ver Tabla 3.2). Se indica la posición actual de

cabeza de la maquina con una celda en gris.

Tabla 3.2 Tabla para sumar dos enteros

State	Tape										
0		O	O	O		O	O	O	O		
0		O	O	O		O	O	O	O		
0		O	O	O		O	O	O	O		
0		O	O	O		O	O	O	O		
1		O	O	O	O	O	O	O	O		
1		O	O	O	O	O	O	O	O		
1		O	O	O	O	O	O	O	O		
1		O	O	O	O	O	O	O	O		
1		O	O	O	O	O	O	O	O		
2		O	O	O	O	O	O	O	O		
stop		O	O	O	O	O	O	O			

Es muy interesante entender como la tabla de estado de transiciones (el programa) trabaja, y examina el comportamiento de la maquina paso a paso. ¿Usted cree que podría ser capaz de modificar la tabla para hacer que la maquina regrese a su celda original después de sumar los dos números? Puedo asegurarle que esto no es difícil. Un tip rápido, Actualmente tiene que cambiar el estado 2 y agregar otro estado para dejar que la maquina vaya hacia la izquierda hasta que encuentre otro blanco.

Como puede ver, ahora que nos enfocamos más, este dispositivo es realmente mas como un programa de computadora (una especie de software) más bien que una computadora

(hardware), la clave parte de un principio de algoritmo de programación definido en la acción de la tabla.

Programando la Maquina de Turing con NQC

 **Nota:**

Antes de iniciar observamos la versión del RCX, si es 1.0 es necesario bajar la actualización del Firmware a 2.0, buscando un archivo en internet llamado *firm0328.lgo* y actualizándolo con el software BricxCC Command, en el menú de Tools en la opción Download Firmware. De lo contrario no se podrá compilar correctamente el programa y no funcionara de manera adecuada la MT.

Se iniciara la practica de programación explicando por separado los procedimientos. Examinados uno por uno. Hay múltiples procedimientos identificados por la palabra reservada *void* cuando no devuelve ningún valor y el programa principal llamado *task main()*.

Primero definimos dos variables:

- Una para almacenar el estado de la máquina, indicado por un número.
- Otra usado por el programa para indicar cuando ha terminado su trabajo y detenerlo (el autor decidió llamar a esta "done" porque "stop" es una palabra reservada en NQC).

En NQC se manejan variables booleanas para manejar un valor Verdadero/Falso, entonces se declaran como *int* y ambas son iniciadas en cero.

int state = 0;

int done = 0;

En el RCX que usaremos dos sensores: uno de tacto para censar los movimientos paso a paso, y un sensor de luz para leer el símbolo dentro de la celda. La energía de los motores la ponemos al máximo, y finalmente aprovechamos una nueva característica del firmware RIS 2.0 (en su versión 0328) mostrando el valor de la variable *state* en el display del RCX. Mientras se ejecuta la maquina podemos observar en que estado se encuentra.

```
void setup()
{
SetSensorType(SENSOR_2, SENSOR_TYPE_TOUCH);
SetSensorMode(SENSOR_2, SENSOR_MODE_RAW);
SetSensorType(SENSOR_3, SENSOR_TYPE_LIGHT);
SetSensorMode(SENSOR_3, SENSOR_MODE_RAW);
SetPower(OUT_A+OUT_B+OUT_C, OUT_FULL);
SetUserDisplay(state,0);
}
```

Ahora se escribe un procedimiento para controlar el movimiento de la cinta una celda a la vez. El motor de control de dirección podría ser activado en cualquier dirección, entonces tiene que ejecutarse hasta que la clavija gris desengrane el sensor de tacto y otra clavija gris engrane de nuevo. Para hacer esto se han creado dos constantes al inicio del programa:

```
#define TOUCH_LOW_THRESHOLD 500
#define TOUCH_HIGH_THRESHOLD 1000
```

El motor debe detenerse sólo después de leer un valor más alto que *el TOUCH_HIGH_THRESHOLD* y un valor más bajo que *el TOUCH_LOW_THRESHOLD*. El autor define estos dos valores como constantes, son simples y fácilmente modificables, para determinar los valores que son aceptables para su sensor. (Recuerde que los motores y sensores pueden ser diferentes unos de otros, por lo tanto se necesita de algún modo ajustar la lectura). Para este particular programa, el autor decidió hacer que el

sensor de tacto usara valores RAW en lugar del modo Booleano. Por dos razones se eligió esta opción. Primero, cada sensor es diferente de otros, y el sensor de tacto no es la excepción. Además el sensor de tacto no maneja solo dos posiciones, on/off, sino una serie de valores intermedios. Al manejar los movimientos estos tienen que ser muy precisos, así como el movimiento exacto de una celda de la cinta, se tiene que ser capaz de controlar la lectura de un modo muy preciso.

Nótese que los valores 500 y 1000 son solo ejemplos para dos valores que podrían ser una parte alta y una parte baja de todas las posibles lecturas del sensor de tacto, (esto usualmente se encuentra en un rango entre 250 y 1020). Aquí están los procedimientos para mover una celda de derecha a izquierda:

Estos podrían estar bien, pero hay que calibrarlos en su sistema con ayuda del botón View de su RCX.

Aquí están los procedimientos para mover una celda de derecha a izquierda:

```
void move_right()
{
// run motor until the touch sensor engages a new peg
OnFwd(OUT_A);
while (SENSOR_2 < TOUCH_HIGH_THRESHOLD) {};
while (SENSOR_2 > TOUCH_LOW_THRESHOLD) {};
Off(OUT_A);
}
void move_left()
{
// run motor until the touch sensor engages a new peg


OnRev(OUT_A);
while (SENSOR_2 < TOUCH_HIGH_THRESHOLD) {};
while (SENSOR_2 > TOUCH_LOW_THRESHOLD) {};
Off(OUT_A);
```

```
}
```

Note que cuando observe a la maquina con el RCX cerca de usted, suponiendo que esta frente a ella, si usted ejecuta el procedimiento *move_right*, la cinta se mueve hacia la izquierda y viceversa. Por lo tanto se mantenemos que el concepto de “move right” significa “ mover una celda a la derecha”. Como se observa la cinta sub-ensamblada esta compuesta con múltiples celdas que contienen dos ligeros brazos adyacentes como switch. Consideramos que si el brazo ligero esta apagado, la celda esta vacía, si este se enciende la celda contiene un símbolo. Usando el encendido y apagado de los switch podemos controlar el borrado y la escritura de las celdas en la cinta. Un impulso muy corto para el motor es necesario para que los brazos ligeros pasen de un lado al otro. Y otro impulso en dirección opuesta devolverá el brazo a su posición original.

```
void write()
{
OnRev(OUT_C);
Wait(5);
OnFwd(OUT_C);
Wait(5);
Off(OUT_C);
}
void erase()
{
OnFwd(OUT_B);


Wait(5);
OnRev(OUT_B);
Wait(5);
Off(OUT_B);
```

}

La parte *main()* del programa incluye la tarea de coordinar las otras funciones y poner en practica la tabla de estado de transición que es la clave de operación de la maquina de Turing. En NQC , no hay mejor manera de manejar la tabla dentro del código usando las sentencias *if y switch…case*. La estructura del *switch* checa el estado de la maquina y la sentencia *if* checa el símbolo a leer en la cinta. Un modo conveniente de manejar el sensor de lectura es definiendo una nueva constante: *#define LIGHT_THRESHOLD 750.*
Si la lectura del sensor de luz es mayor que *LIGHT_THRESHOLD entonces el brazo, esta en su mayor posición y la celda contiene un símbolo, de otra manera la celda esta vacía. Probablemente usted tendrá que calibrar el valor de umbral para su configuración y las condiciones de luz. Use el botón View para descubrir la lectura mas alta (con el brazo abajo del sensor) y la más baja (con el brazo en la otra posición), así encontrara el valor entre los dos: este es su umbral. Mi sensor lee de 720 a 780, entonces se calcula:* (720 + 780) / 2 = 750.

```
task main()
{
setup();
do
{
switch(state)
{
case 0:



if (SENSOR_3 < LIGHT_THRESHOLD)
{
write();
state = 1;
```

```
}
move_right();
break;
case 1:
if (SENSOR_3 < LIGHT_THRESHOLD)
{
move_left();
state = 2;
}
else
move_right();
break;
case 2:
erase();
done = 1;
break;
}
Wait(100);
}
while (done != 1);
Float(OUT_A);
PlaySound(SOUND_UP);
}
```

Despues de definir la instalación de la maquina, el programa empezara usando el estado de la tabla de transición y la lectura del sensor para decidir que hacer (cambio de estado,

mover, escribir símbolo), hasta que haya terminado el trabajo y la variable *done* se ponga en 1 (verdadero).

Ahora que el programa ha terminado. usted puede probar la maquina de turing que ha creado. Como podemos ver la programación no fue tan difícil. Puede ajustar el

comportamiento del robot modificando las tres constantes que definió. Si usted desea hacer un cambio en la tabla de transiciones, tendrá que reprogramar la sentencia *switch…case del task main(), y dejar el resto como estaba.*

A continuación se muestra el código para sumar dos enteros:

```
/*
  Lego Turing Machine
  a program for the Mindstorms Masters book
  by Giulio Ferrari
  (C) 2003 by Syngress Publishing, Inc.
*/

/* Add two integers */

#define TOUCH_LOW_THRESHOLD 500
#define TOUCH_HIGH_THRESHOLD 1000
#define LIGHT_THRESHOLD 750

int state = 0;
int done = 0;

void setup()
{


  SetSensorType(SENSOR_2, SENSOR_TYPE_TOUCH);
  SetSensorMode(SENSOR_2, SENSOR_MODE_RAW);
  SetSensorType(SENSOR_3, SENSOR_TYPE_LIGHT);
  SetSensorMode(SENSOR_3, SENSOR_MODE_RAW);
```

```
  SetPower(OUT_A+OUT_B+OUT_C, OUT_FULL);
  SetUserDisplay(state,0);
}

void move_right()
{
  // run motor until the touch sensor engages a new peg
  OnFwd(OUT_A);
  while (SENSOR_2 < TOUCH_HIGH_THRESHOLD) {};
  while (SENSOR_2 > TOUCH_LOW_THRESHOLD) {};
  Off(OUT_A);
}

void move_left()
{
  // run motor until the touch sensor engages a new peg
  OnRev(OUT_A);
  while (SENSOR_2 < TOUCH_HIGH_THRESHOLD) {};
  while (SENSOR_2 > TOUCH_LOW_THRESHOLD) {};
  Off(OUT_A);
}



void write()
{
  OnRev(OUT_C);
  Wait(5);
  OnFwd(OUT_C);
```

```
  Wait(5);
  Off(OUT_C);
}

void erase()
{
  OnFwd(OUT_B);
  Wait(5);
  OnRev(OUT_B);
  Wait(5);
  Off(OUT_B);
}

task main()
{
  setup();
  do
  {
    switch(state)
    {
      case 0:
        if (SENSOR_3 < LIGHT_THRESHOLD)
        {

          write();
          state = 1;
        }
        move_right();
        break;
```

```
      case 1:
        if (SENSOR_3 < LIGHT_THRESHOLD)
        {
          move_left();
          state = 2;
        }
        else
          move_right();
        break;
      case 2:
        erase();
        done = 1;
        break;
    }
  Wait(100);
  }
  while (done != 1);
  Float(OUT_A);
  PlaySound(SOUND_UP);
}
```

 **Sugerencias:**

Es importante recalcar que se pueden realizar diversos ejercicios con la MT, sin embargo esta se encuentra limitada al alfabeto de 0´s y 1´s, es labor del instructor verificar que otras tareas se pueden hacer con este alfabeto, sin embargo se muestran unos ejemplos simples en el siguiente párrafo.

- **Verificar el final de la cinta y detenerse.**
- **Realizar operaciones de resta.**

Resumen

La Maquina de Turing es un modelo matemático muy complejo, pero con el cual podemos conocer el inicio de la computabilidad, y así valorar el increíble mundo de la computación. Este material es de gran apoyo a personas que imparten materias de Teoría de autómatas y lenguajes formales y teoría de la computación.