Mapeado del entorno utilizando un robot móvil teleoperado con Lego Mindstorm

Robots Móviles
Máster en Robótica y Automatización

Abderrahmane Rahiche <arahiche@ing.ucm3.es>
Pablo Marín Plaza <100073801@alumnos.uc3m.es>
Verónica González Pérez <100289866@alumnos.uc3m.es>
Raúl Pérula Martínez <raul.perula@uc3m.es>

Contenido

1. Introducción
2. Modelo diseñado
3. Algoritmo implementado
4. Resultados
5. Conclusiones
6. Futuras mejoras
7. Referencias

INTRODUCCIÓN

Introducción

- Para realizar un robot móvil que pueda realizar un mapeado y la navegación en un entorno no conocido a priori. Se ha usado dos tecnologías bastante utilizadas en estos momentos.

- LEGO Mindstorms:
  - El LEGO Mindstorms es un kit de bajo coste y programable para robótica educativa.
  - Usando dicho kit de desarrollo educativo, se ha realizado el diseño y la construcción del robot móvil.

Introducción

- ROS:
  - ROS (Robot Operating System) proporciona un conjunto de librerías y herramientas para ayudar a los desarrolladores para crear aplicaciones para robots.
  - Proporciona abstracción de hardware, drivers de dispositivos, librerías, simuladores gráficos, paso de mensajes, etc.
  - Usando ROS, se pueden desarrollar algoritmos para el mapeado de entornos con el robot NXT.
  - Además, se puede controlar el robot para una teleoperación del mismo.

DISEÑO DEL MODELO

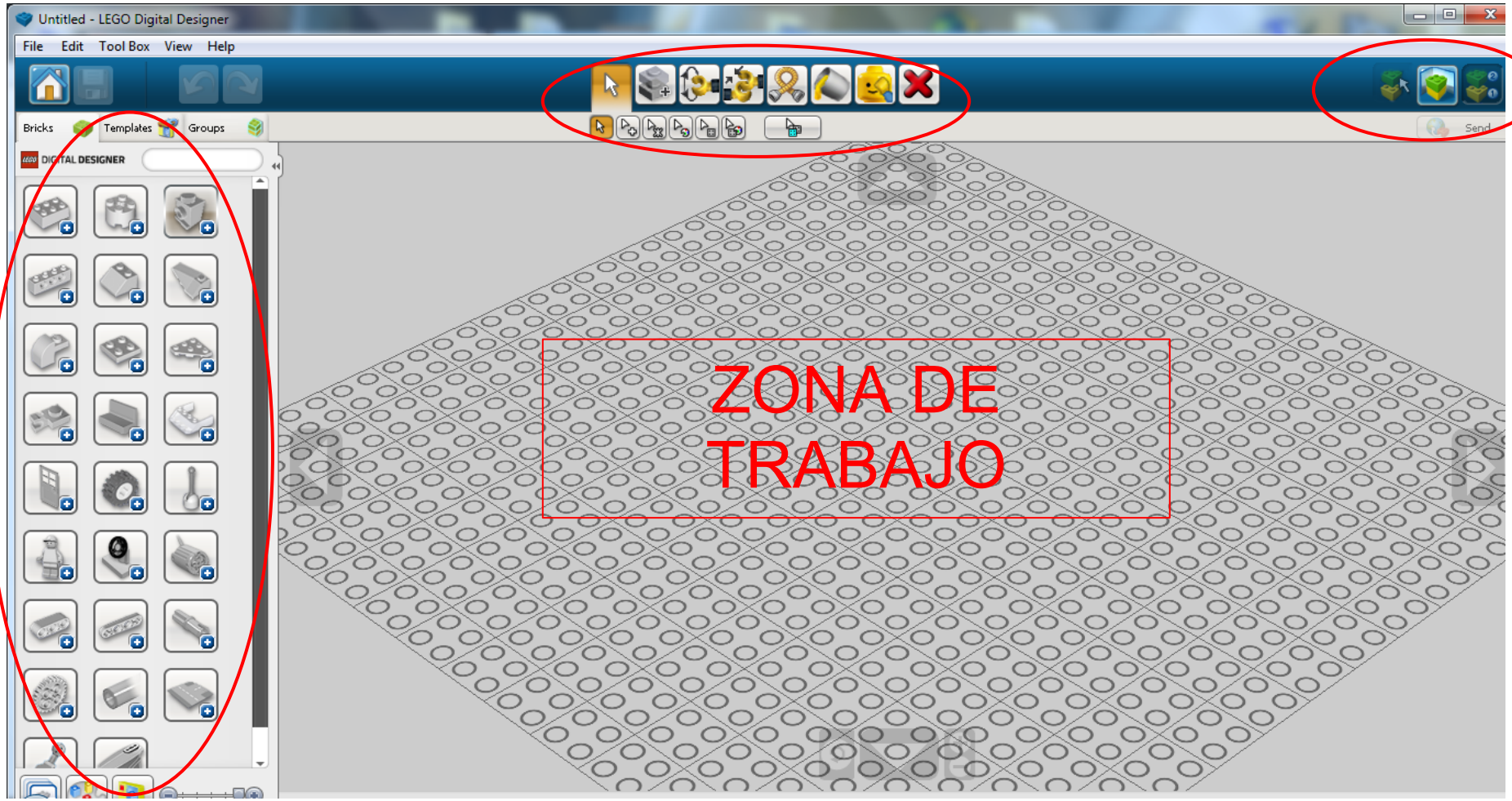
Diseño del Modelo

LEGO DIGITAL DESIGNER

- MODELOS 3D
- INTUITIVO
- REALISTA
- GUÍA DE MONTAJE
- COMPARTIR

Diseño del Modelo

LEGO DIGITAL DESIGNER



Diseño del Modelo

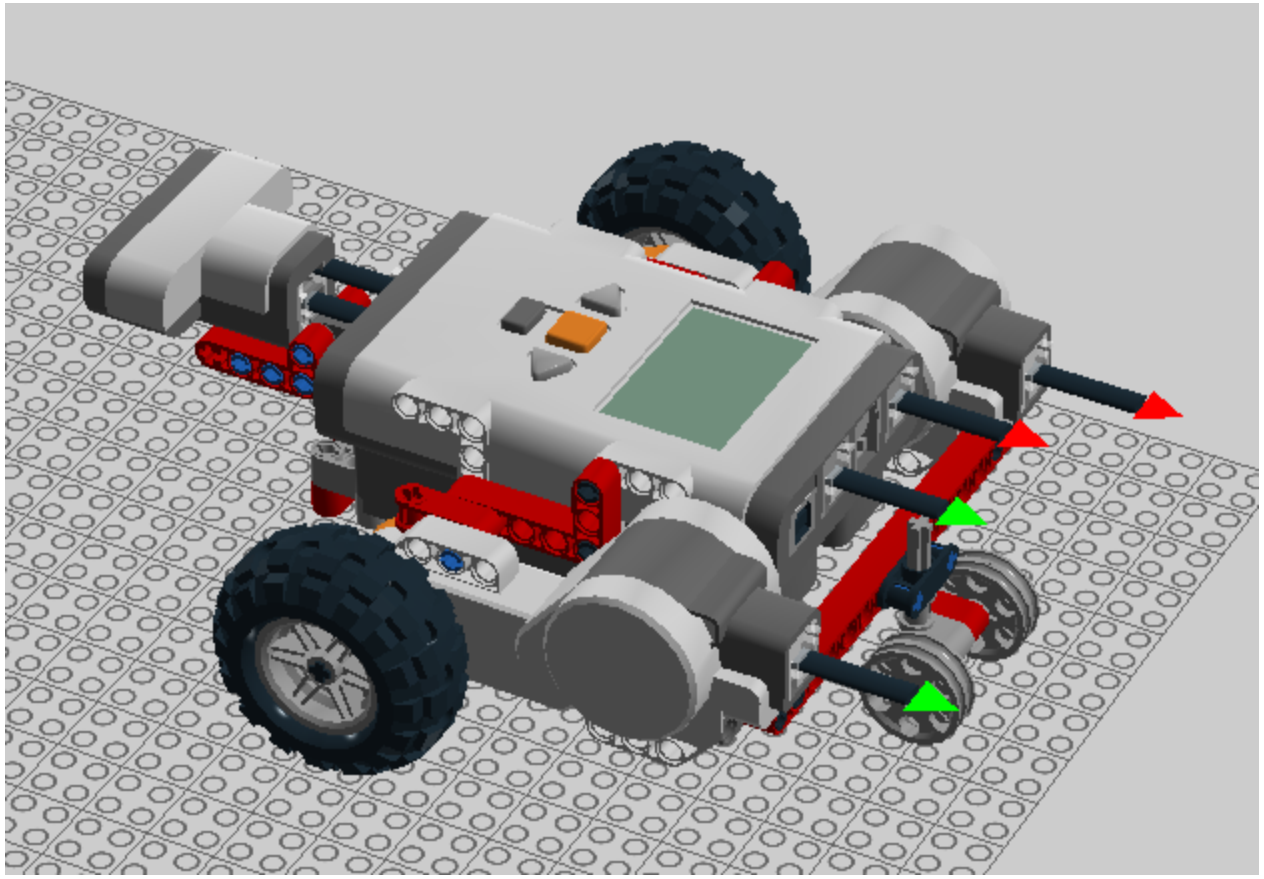
Tipos:

- 3 Ruedas:
  - Diferencial
  - Direccional (steering)
  - Tracción + Direccional
  - SynchroDrive (todas giran a la vez)

- 4 Ruedas:
  - Diferencial
  - Direccional

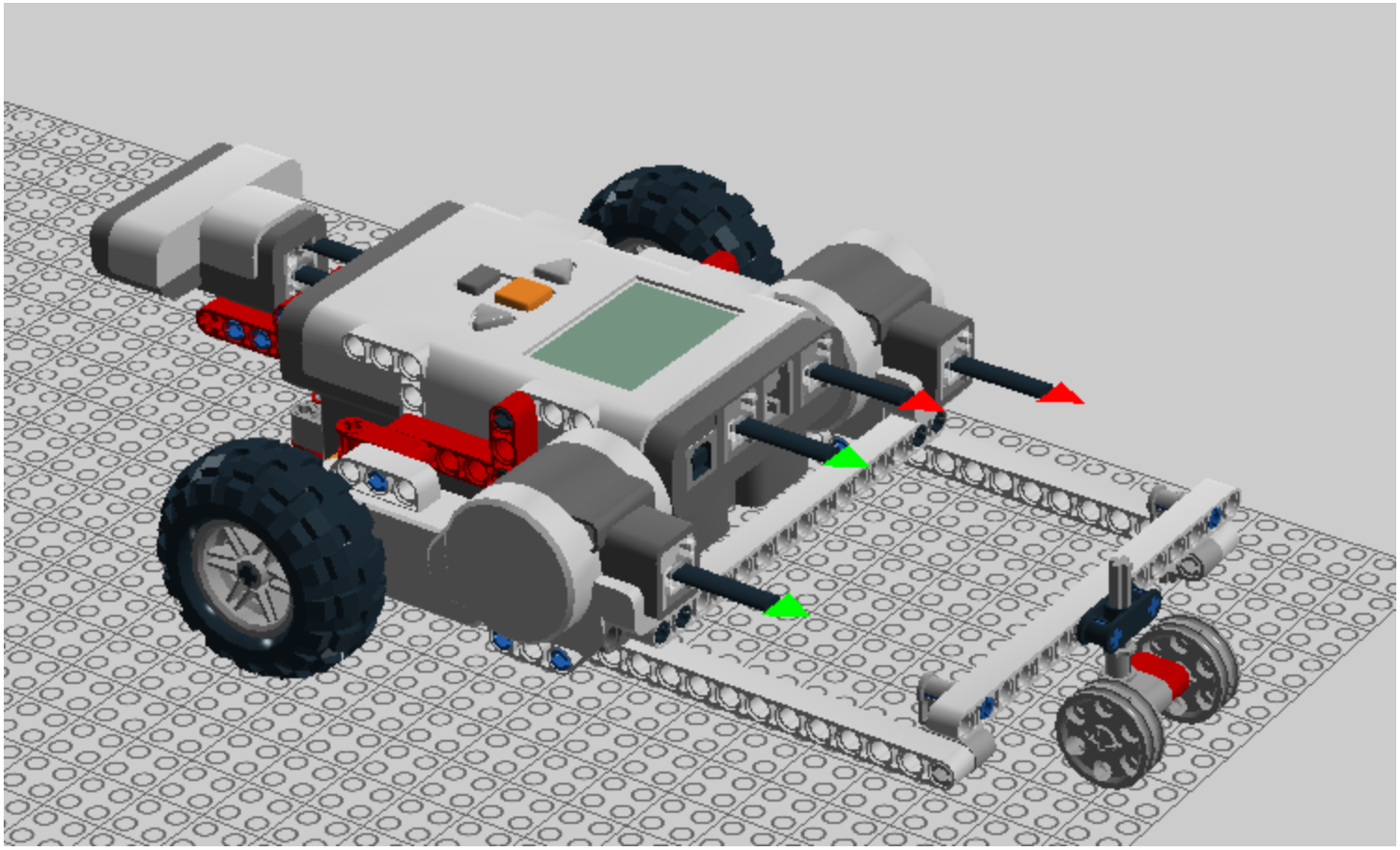
Diseño del Modelo

Modelo 3 Ruedas: Primer Boceto



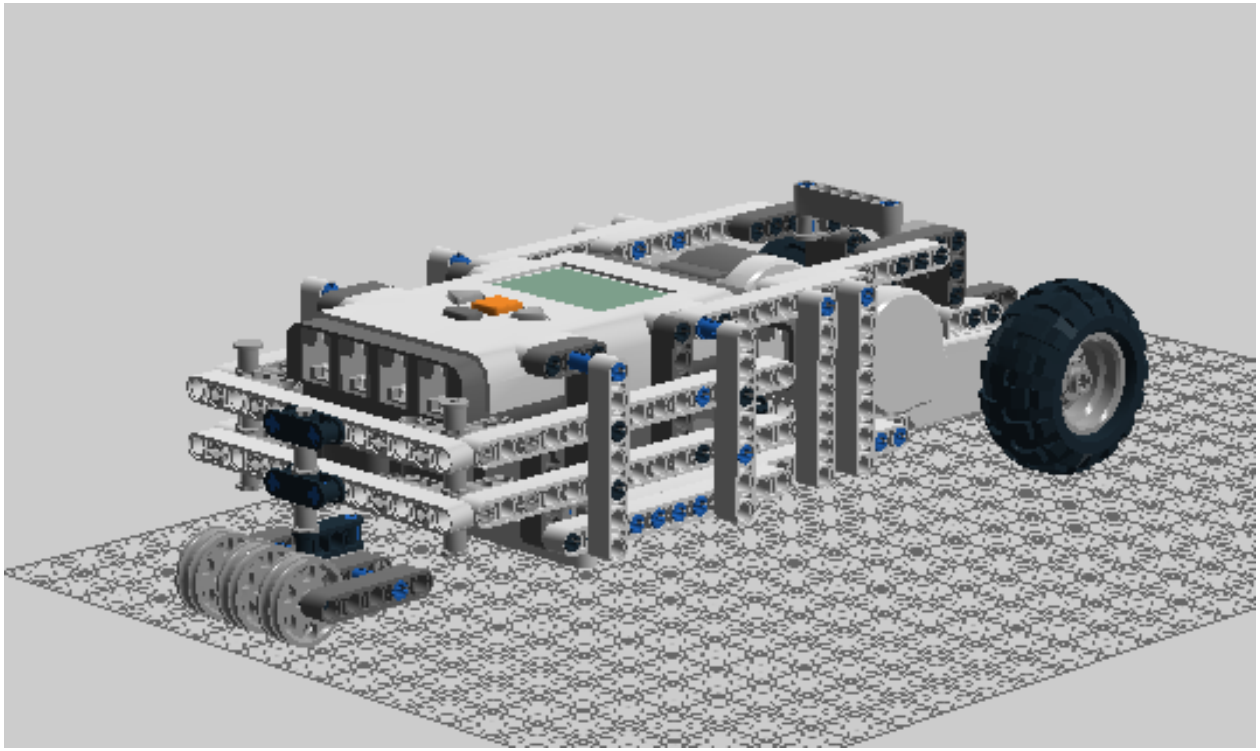
Diseño del Modelo

Modelo 3 Ruedas: Segundo Boceto



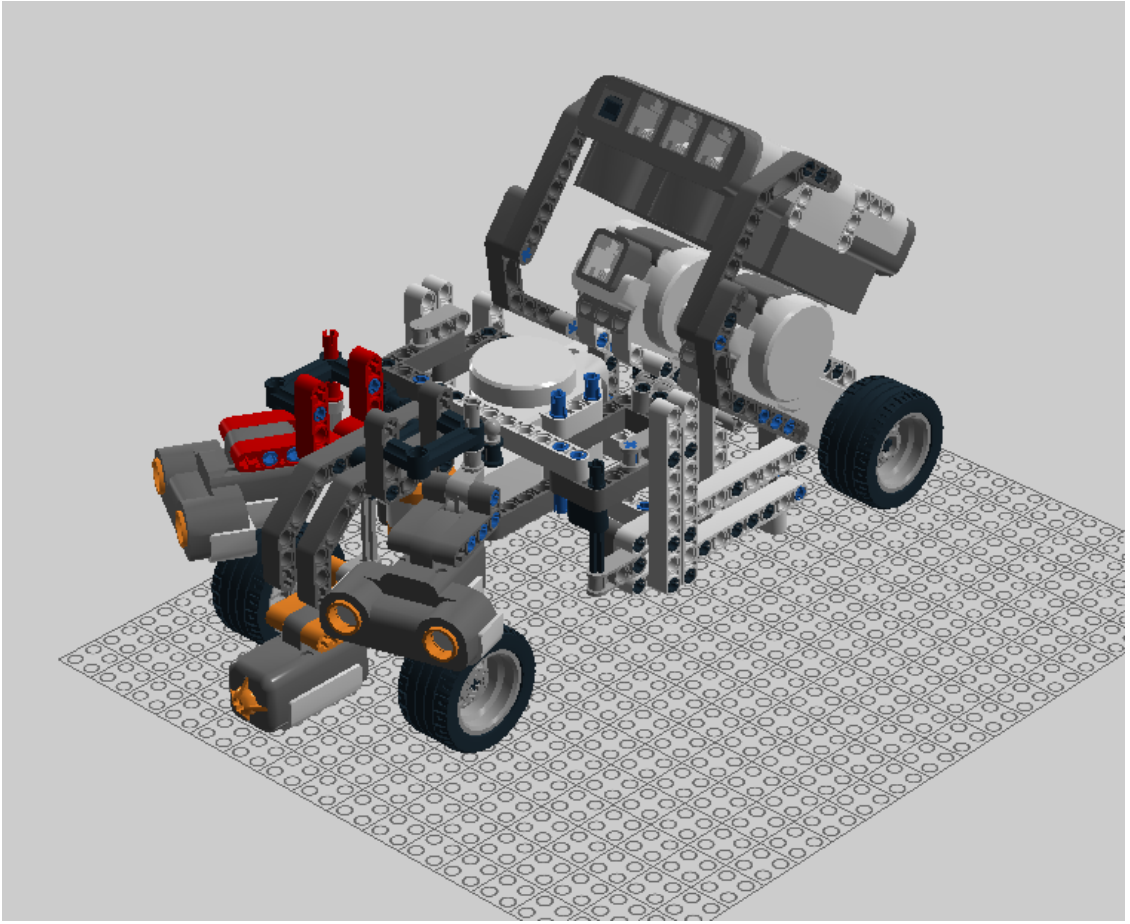
Diseño del Modelo

Modelo 3 Ruedas: Tercer Boceto

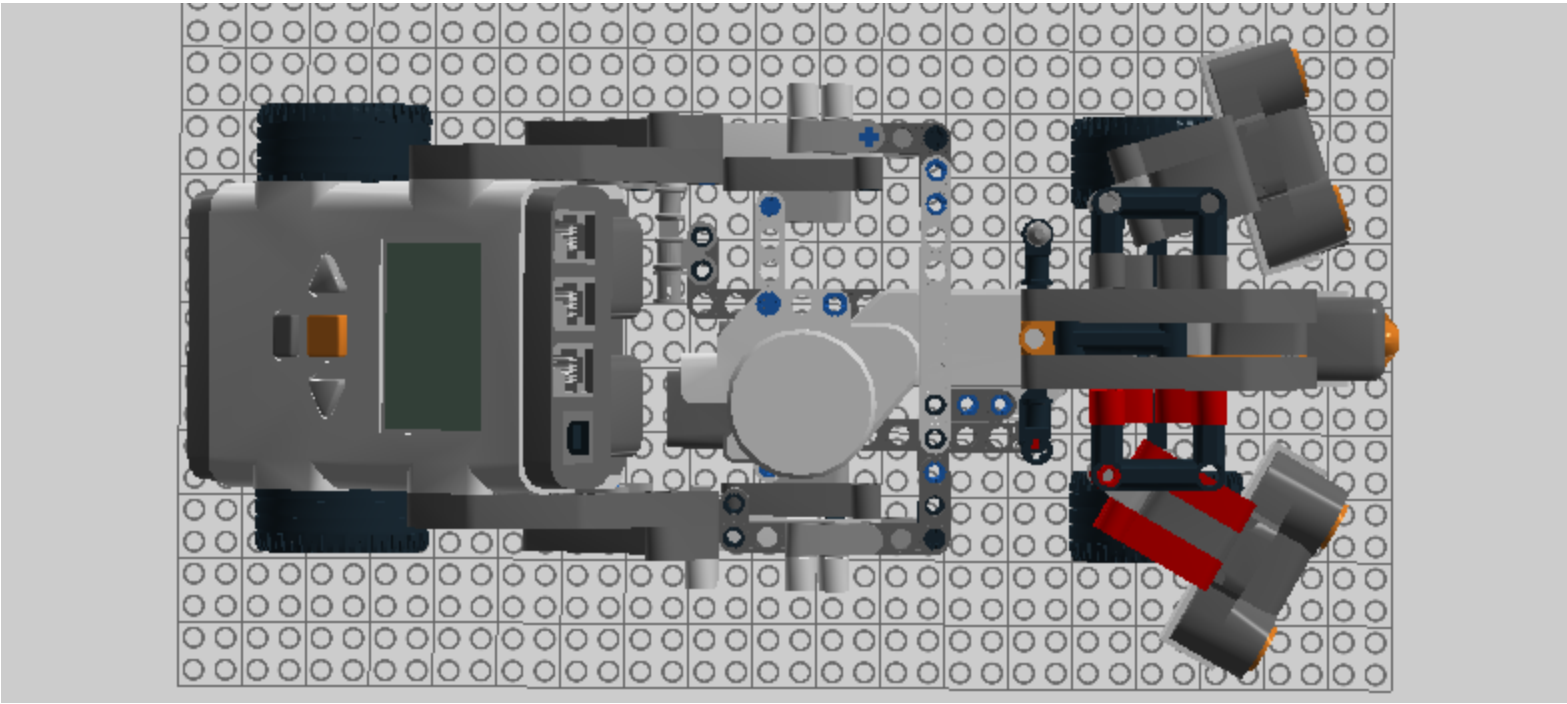


Diseño del Modelo

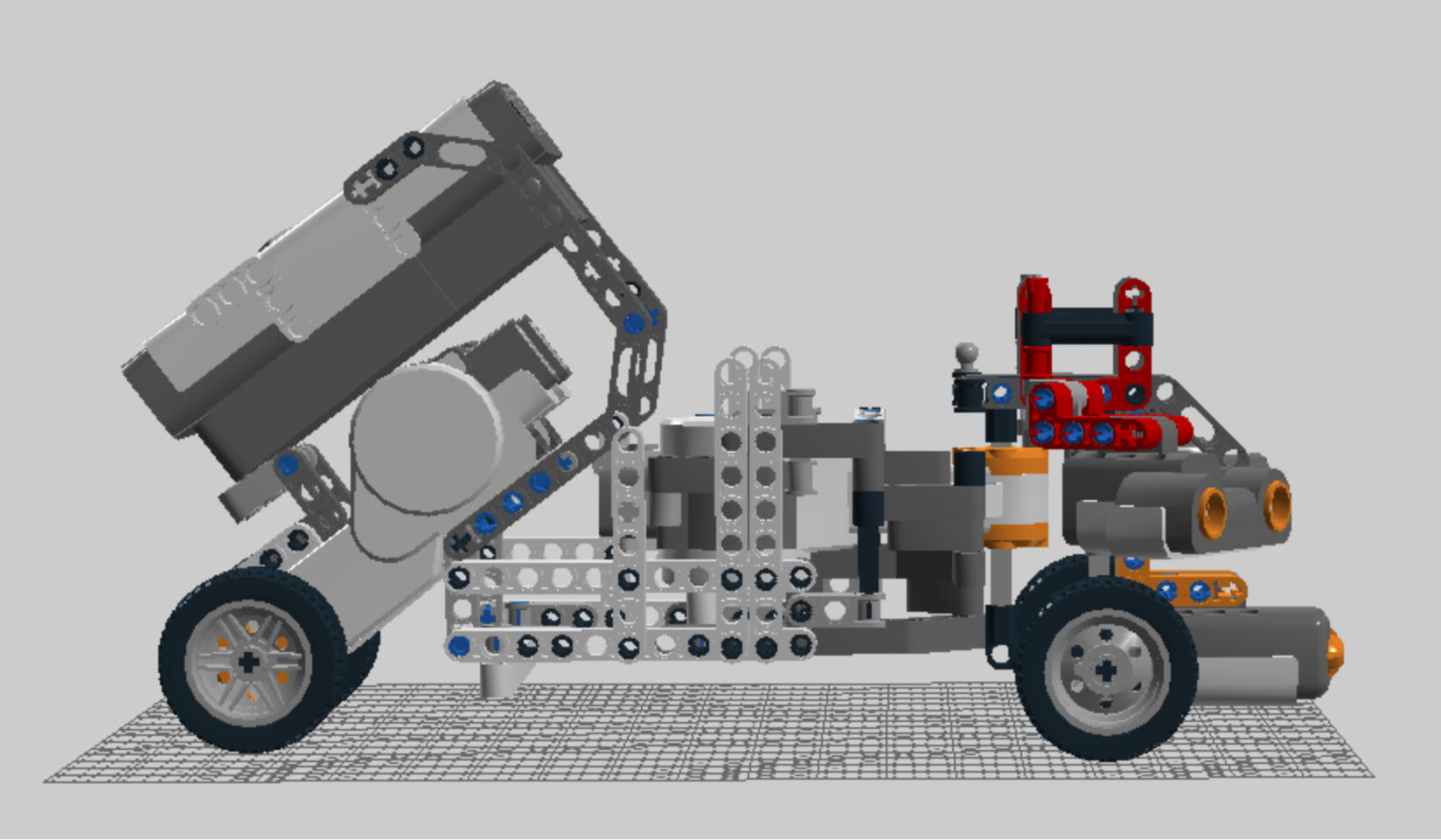
Modelo 4 Ruedas: Primer boceto



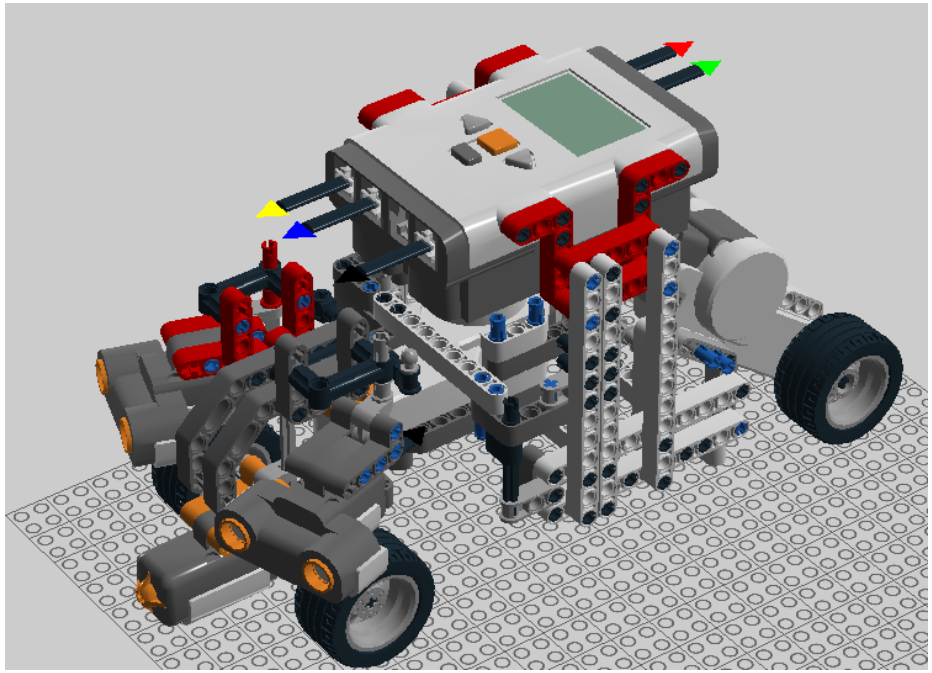
Diseño del Modelo

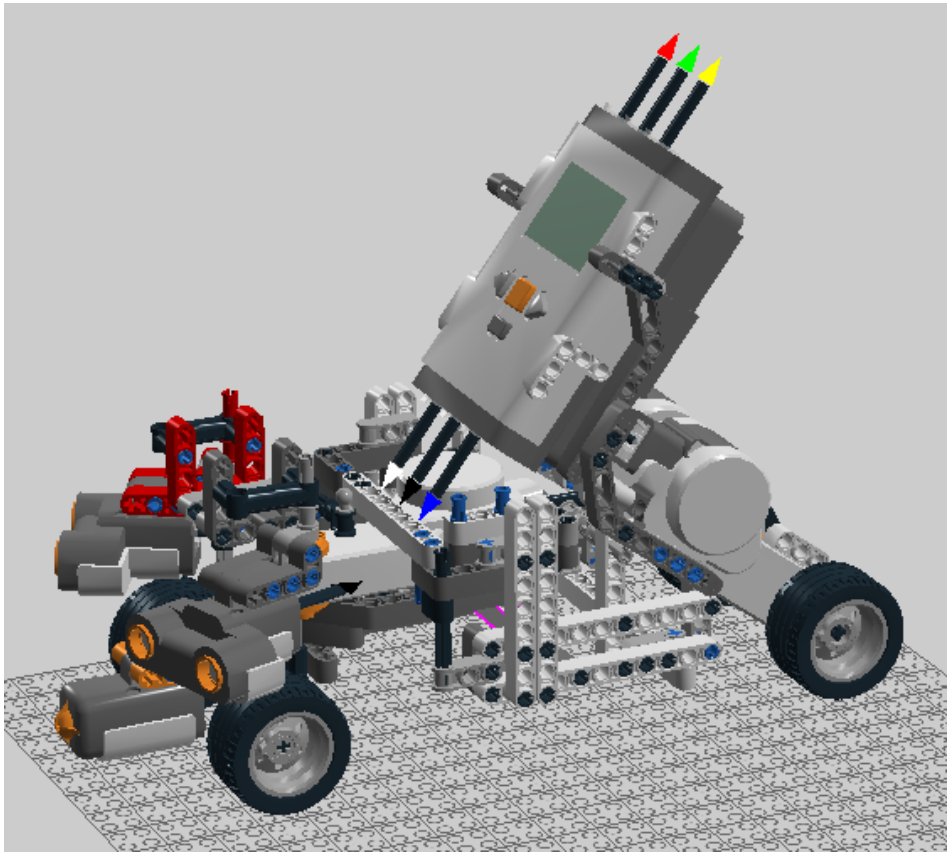


Diseño del Modelo



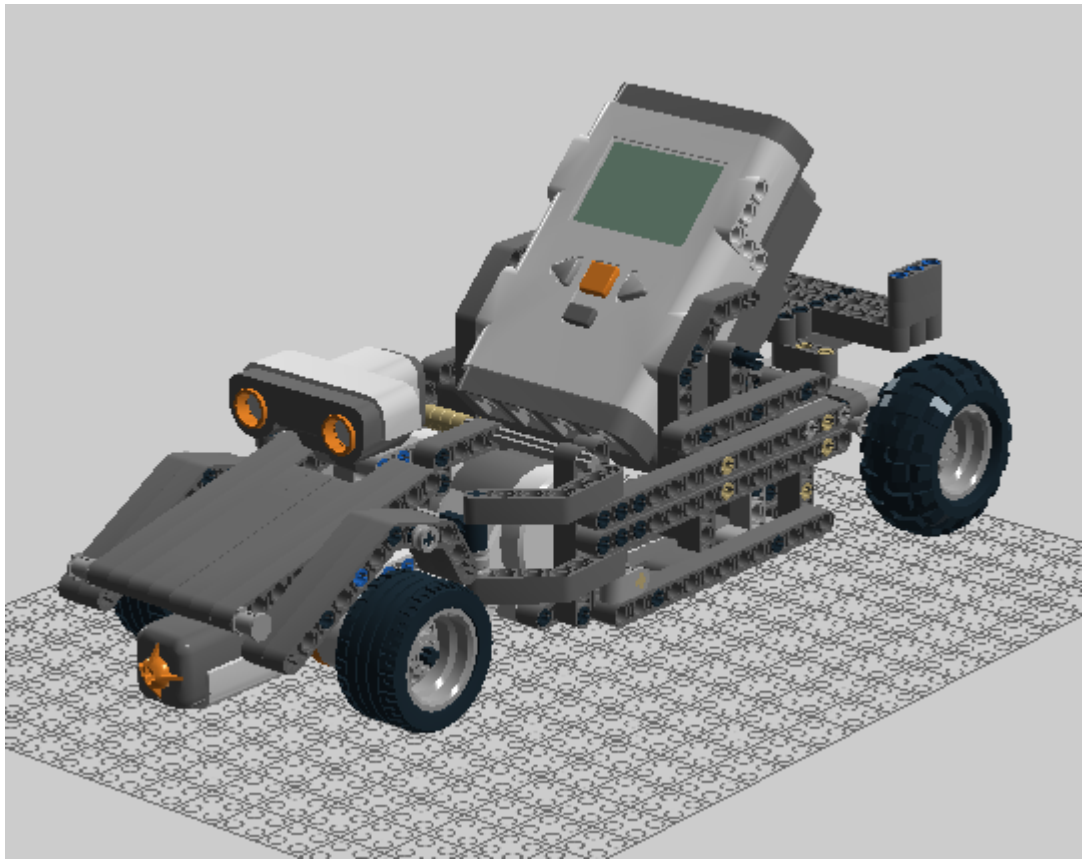
Diseño del Modelo



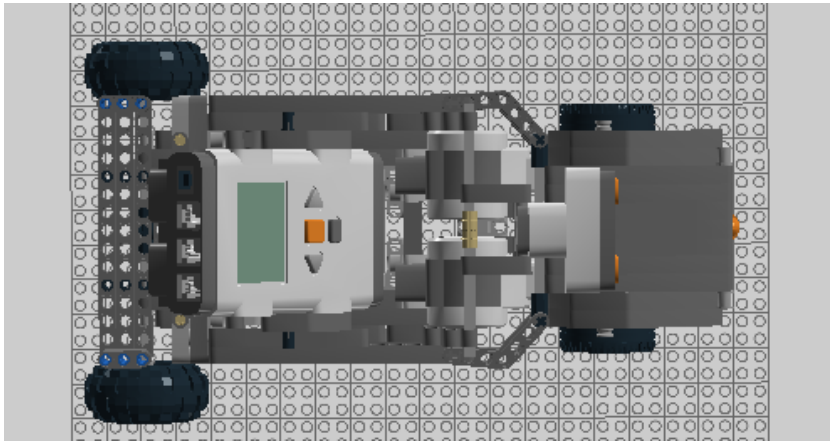


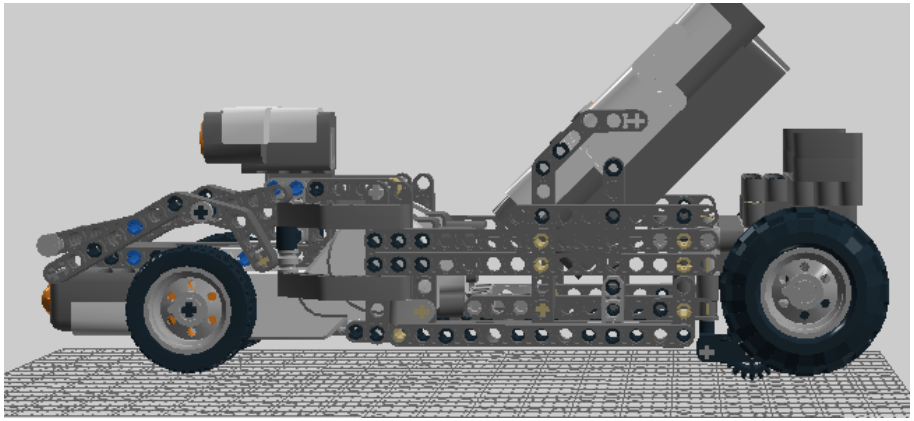
Diseño del Modelo

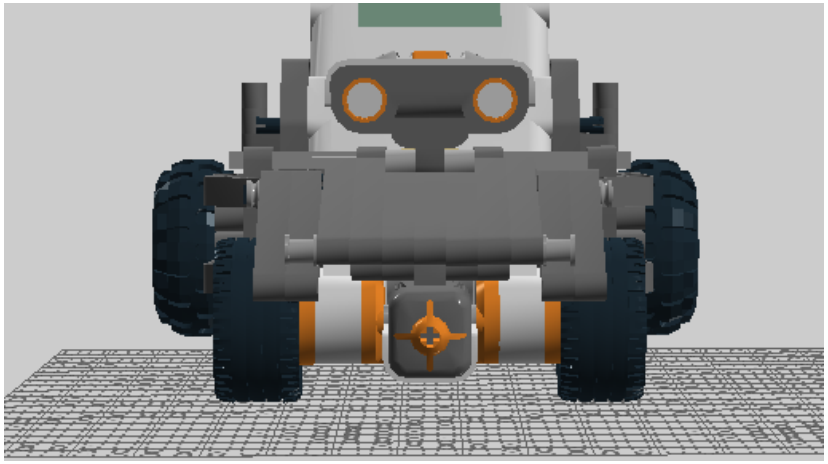
Modelo 3 ruedas: Cuarto boceto

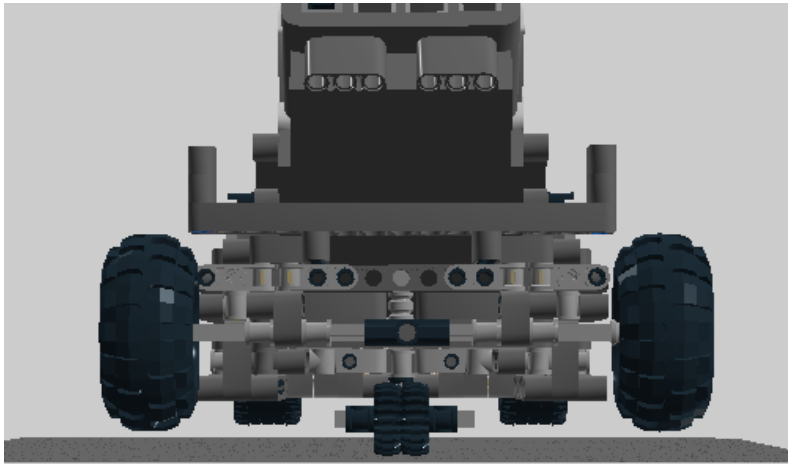


Diseño del Modelo



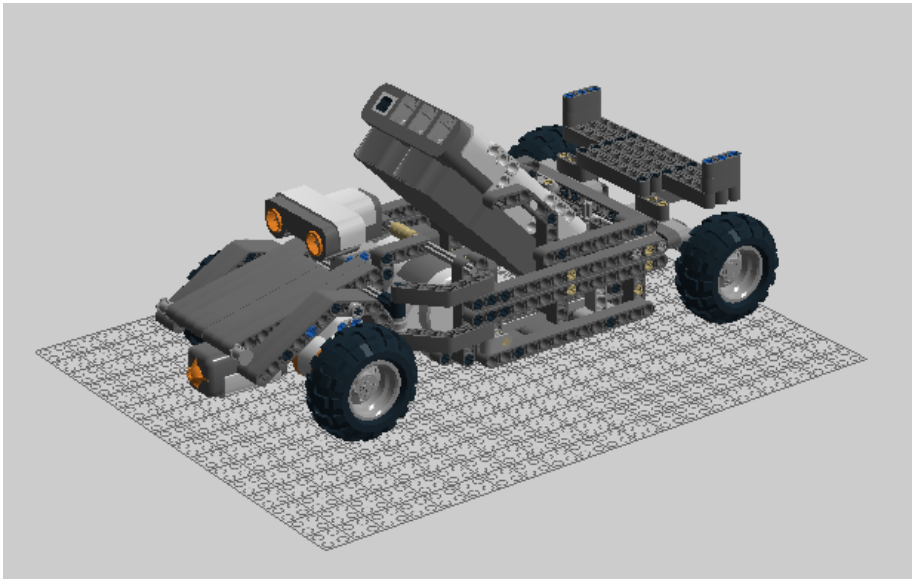


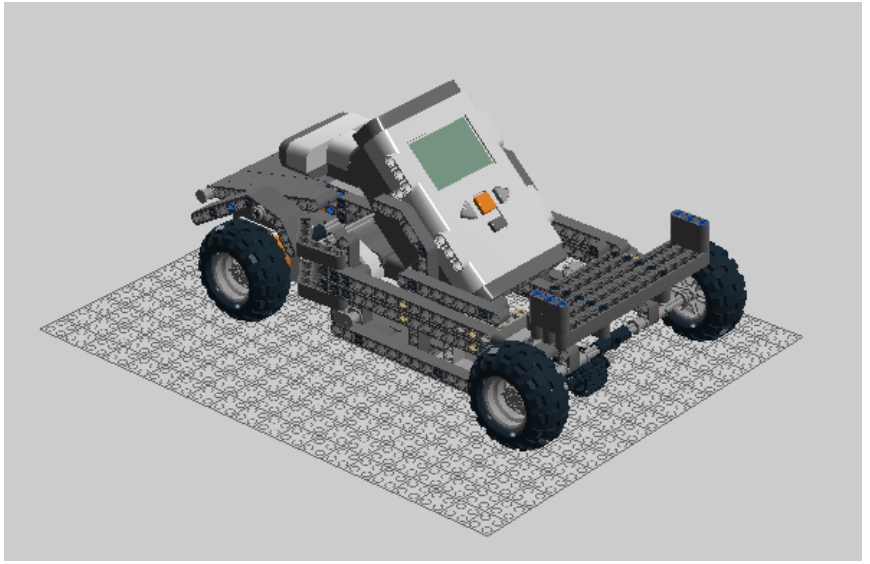




Diseño del Modelo

Modelo de 3 ruedas: Boceto definitivo





ALGORITMO IMPLEMENTADO

Algoritmo. Configuración del robot

robot.yaml

```yaml
nxt_robot:
  - type: motor
    name: r_wheel_joint
    port: PORT_A
    desired_frequency: 20.0

  - type: motor
    name: l_wheel_joint
    port: PORT_B
    desired_frequency: 20.0

  - type: touch
    frame_id: r_touch_link
    name: touch_sensor
    port: PORT_3

  - type: ultrasonic
    frame_id: ultrasonic_link
    name: ultrasonic_sensor
    port: PORT_2
    spread_angle: 0.2
    min_range: 0.01
    max_range: 2.5
    desired_frequency: 5.0
```

Algoritmo.
Puesta en marcha del robot

- Carga del modelo:

```
<param name="robot_description" textfile="$(find nxt_robot_sensor_car)/robot.urdf"/>
```

- Carga de la configuración

```
<node pkg="nxt_ros" type="nxt_ros.py" name="nxt_ros" output="screen" respawn="true">
  <rosparam command="load" file="$(find nxt_mobile_robot)/conf/robot.yaml" />
</node>
```

Algoritmo.
Puesta en marcha del robot

- **Odometría:**

X = Distance * cos(Alpha)
Y = Distance * sin(Alpha)
Alpha = (DR - DL) / 2
Distance = (DR + DL) / 2

- **Sensores de ultrasonido:**

El sensor de ultrasonidos es capaz de detectar y calcular la distancia de los objetos que se encuentran enfrente de él.

- **Filtro extendido de Kalman:**

Los datos recibidos por la odometría y los sensores de ultrasonidos se combinan por un filtro de Kalman para localizar el robot y estimar la posición de los obstáculos en el entorno.

Algoritmo.
Puesta en marcha del robot

- Controlador, Odometría, Filtro de Kalman:

```
<!-- base controller -->
<node pkg="nxt_controllers" type="base_controller.py" name="base_controller" output="screen"/>

<!-- base odometry -->
<node pkg="nxt_controllers" type="base_odometry.py" name="base_odometry" output="screen"/>

<!-- robot pose ekf -->
<node pkg="robot_pose_ekf" type="robot_pose_ekf" name="robot_pose_ekf">
  <remap from="imu_data" to="gyro_imu" />
</node>

<!-- fixed transform -->
<node pkg="tf" type="static_transform_publisher" name="base_footprint_fixed_publisher" args="0 0 0 0 0 0
base_footprint base_link 100"/>
```

Ejecución (en una terminal):

```
$ roslaunch nxt_mobile_robot robot.launch
```

Algoritmo. Teleoperación y Reconocimiento de obstáculos

- **Obtención de los puntos**

```
<!-- point cloud converter -->
<node pkg="nxt_assisted_teleop" type="range_to_pointcloud.py" name="range_to_pointcloud">
  <remap from="range_topic" to="ultrasonic_sensor" />
  <remap from="cloud_topic" to="point_cloud" />
</node>
```

- **Teleoperación del robot**

```
<!-- keyboard teleop node -->
<node pkg="nxt_teleop" type="nxt_teleop_key" name="nxt_teleop"  output="screen">
  <param name="scale_linear" value="1" type="double"/>
  <param name="scale_angular" value="1" type="double"/>
  <remap from="cmd_vel" to="teleop_cmd_vel"/>
</node>

<node pkg="nxt_assisted_teleop" name="assisted_teleop" type="assisted_teleop" respawn="false" output=
"screen">
  <rosparam file="$(find nxt_assisted_teleop)/launch/sensor_car/assisted_teleop.yaml" command="load" />
</node>
```

Algoritmo. Teleoperación y Reconocimiento de obstáculos

Ejecución (en una nueva terminal):

```
$ roslaunch nxt_mobile_robot assisted_teleop_keyboard.launch
```

Algoritmo. Simulación en RVIZ

Ejecución (en una nueva terminal):

```
$ rosrun rviz rviz
```

Configuración de RVIZ:

- Habilitar **nxt_rviz_plugins** en el menú de plugins.
- Añadir **NTXUltrasonicDisplay** y especificar el topic "**/ultrasonic_sensor**".
- En "**Global Options**" poner el "**Fixed Frame**" a "**/odom_combined**" y el "**Target Frame**" a **<Fixed Frame>**.
- Activar la visualización del "**Grid**".
- Añadir del menú plugins un elemento "**Grid Cells**" y poner el topic a "**assisted_teleop/costmap/obstacles**".
- Añadir otro "**Grid Cells**" y poner el topic a "**assisted_teleop/costmap/inflated_obstacles**". Cambiar el color de este a azul.

RESULTADOS

Resultados (RVIZ)

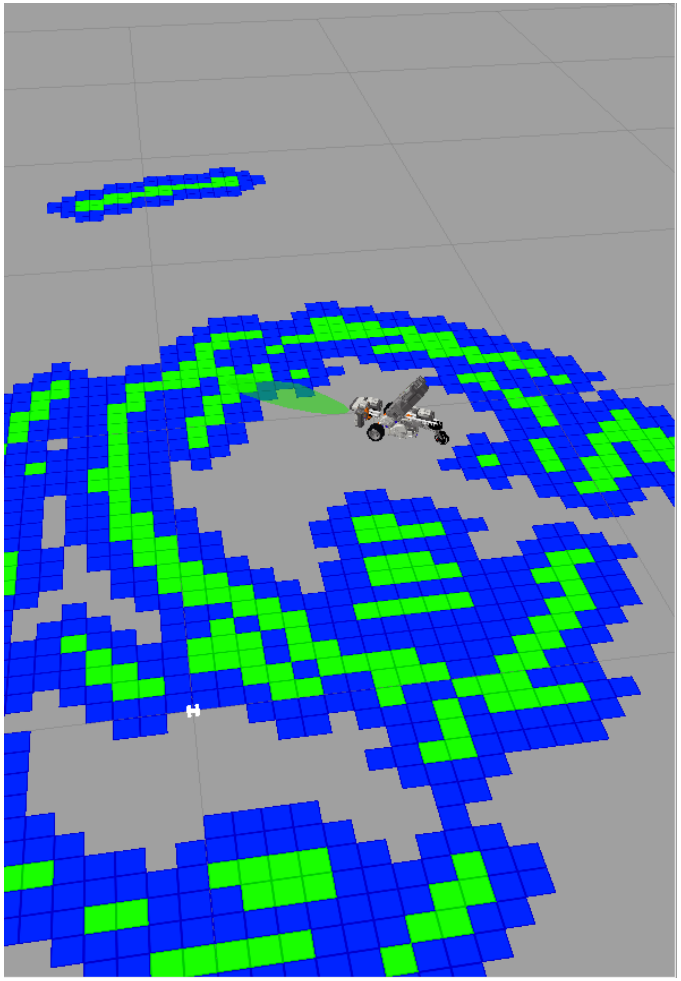
Configuración

Displays

.Global Options	
Background Color	(160,160,160)
Fixed Frame	/odom_combined
Target Frame	<Fixed Frame>
.Global Status: OK	
01. Grid (Grid)	☑
02. Robot Model (Robot Model)	☑
Status: Error	
Visual Enabled	☑
Collision Enabled	☐
Update Interval	0
Alpha	1
Robot Description	robot_description
TF Prefix	
Links	
03. NXTUltrasonicDisplay (NXTUltrasonicDis	☑
Status: OK	
Topic	/ultrasonic_sensor
Color	(25,255,0)
Alpha	0,5
04. Grid Cells (Grid Cells)	☑
Status: OK	
Color	(25,255,0)
Alpha	1
Topic	/assisted_teleop/costmap/obstacles
05. Grid Cells2 (Grid Cells)	☑
Status: OK	
Color	(0,36,255)
Alpha	1
Topic	/assisted_teleop/costmap/inflated_obstacles

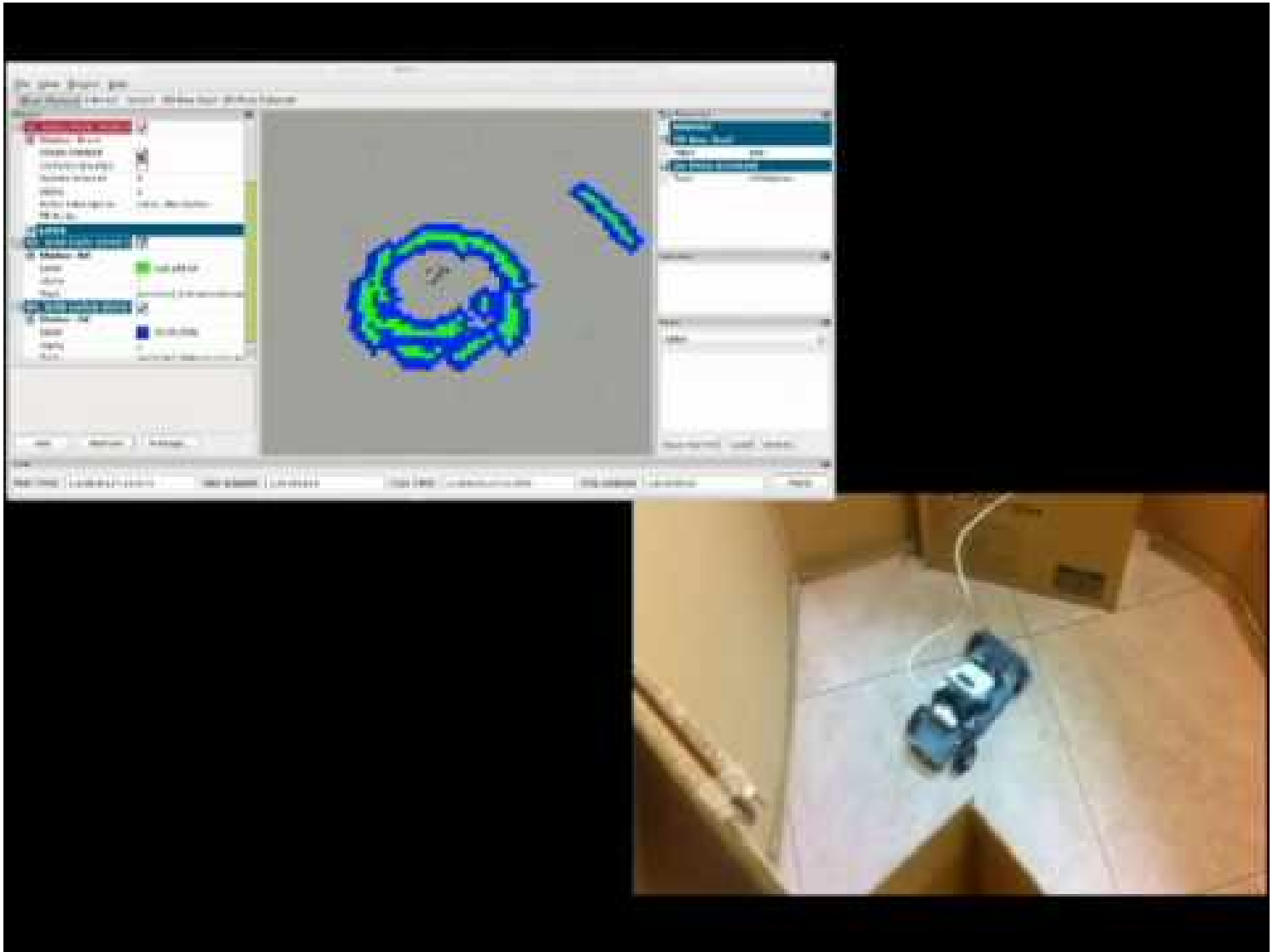
Topic
nav_msgs::GridCells topic to subscribe to.

Add Remove Manage...



Visualización

Resultados.
Vídeo (parte real, parte simulada)



CONCLUSIONES

Conclusiones

- DISEÑO
- LEGO DIGITAL DESIGNER
- ROS
- MAPA

FUTURAS MEJORAS

Futuras Mejoras

- Crear una mayor estabilidad para el robot.
  - Peso del brick.
  - Modificación de rueda loca.
- Poder cargar modelos propios (en RVIZ).
- Poder hacer un mapeado visual (kinect o similar).
- Poder navegar de forma autónoma.

Referencias

1. "ROS NXT". ROS Wiki. Última visita: Mayo-2012. Enlace: http://www.ros.org/wiki/Robots/NXT.

2. "Lego Digital Designer". Última visita: Mayo-2012. Enlace: http://ldd.lego.com/

3. "A SLAM implementation using LEGO Mindstorms NXT and leJOS." PenemuNXT - A Lego Mindstorms NXT project. Última visita: Mayo-2012.
   Enlace http://penemunxt.blogspot.com.es/

¿ ? ¿ ? ¿ ? ¿

PREGUNTAS

MUCHAS GRACIAS