# **Robot Asistente Doméstico Basado en iRobot Create 2**

## **1. Introducción**

El presente proyecto tiene como objetivo transformar un robot de limpieza **iRobot Create 2** en una base móvil para un robot asistente doméstico autónomo. Este sistema estará diseñado para ejecutar tareas avanzadas, como navegación autónoma, mapeo del entorno mediante un sensor **LIDAR**, reconocimiento de estancias, y desplazamiento eficiente entre ellas.

Se implementará una arquitectura basada en **ROS 2**, donde cada nodo cumplirá una función específica, asegurando modularidad, escalabilidad y una interacción eficiente con el hardware del robot. El proyecto incluye un nodo centralizado para la gestión de la conexión con el robot y la coordinación de los diferentes nodos funcionales.

## **2. Arquitectura General del Sistema**

El sistema se organiza en varios nodos ROS 2, cada uno con una responsabilidad bien definida. Los nodos interactúan entre sí a través de **tópicos**, **servicios**, y **acciones**, con un nodo maestro actuando como coordinador global.

El **nodo de comunicación serial** es responsable de interactuar exclusivamente con el hardware del robot a través del puerto serie. Los demás nodos obtienen datos y envían comandos al robot mediante este nodo, evitando conflictos de acceso concurrente al puerto serie.

### **Componentes Principales**

**Nodo Maestro (**master\_node**)**:

* 1. Coordina la interacción entre nodos.
  2. Gestiona la conexión con el robot.
  3. Supervisa el estado general del sistema.

**Nodo de Comunicación Serial (**communication\_node**)**:

* 1. Gestiona el acceso exclusivo al puerto serie.
  2. Publica datos del robot y recibe comandos de otros nodos.

**Nodo de Sensores (**sensor\_node**)**:

* 1. Proporciona información del entorno obtenida de los sensores del robot.

**Nodo de Cotrol de Movimiento (**motion\_control\_node**)**:

* 1. Controla los motores del robot mediante comandos de velocidad.

**Nodo de Odometría (**odometry\_node**)**:

* 1. Calcula y publica la posición y orientación del robot.

**Nodo de Gestión de Carga (**charging\_node**)**:

* 1. Supervisa el nivel de batería y gestiona el retorno a la base de carga.

**Nodo de Navegación (**navigation\_node**)**:

* 1. Utiliza un sensor **LIDAR** para generar un mapa y planificar rutas autónomas.

## **3. Definición de los Nodos**

### **3.1 Nodo Maestro (**master\_node**)**

#### **Descripción**

El nodo maestro es el coordinador global del sistema. Supervisa y controla los demás nodos, administra la conexión con el robot, y proporciona servicios para tareas de alto nivel.

#### **Funciones**

* Gestionar la conexión con el robot a través del nodo de comunicación serial. Debe ejecutar el script remoto para despertar el robot antes de la conexión.
* Monitorear el estado general del sistema. Debe solicitar al resto de Nodos se estado para crear un estado global del robot.
* Gestiona los parámetros dinámicos del resto de nodos centralizando así la configuración del robot en un solo nodo.

#### **Interfaces**

* **Parámetros**:
  + connection\_method: Método de conexión (usb o tcp).
    - [Modo, Datos]
    - Modo: USB / TCP
    - Datos: IP:Puerto / Puerto Serie
  + connection\_time: Fecha de Inicio:
* **Servicios**:
  + /roomba/system/connect: Establece conexión con el robot. /roomba/system/status: Devuelve el estado general del sistema.

#### **Interacción**

* Envía comandos de conexión al nodo de comunicación.
* Solicita estados al resto de nodos para crear un estado global del robot.

### **3.2 Nodo de** Comunicación **Serial (**communication\_node**)**

#### **Descripción**

Este nodo centraliza el acceso al puerto serie del robot, garantizando que solo un nodo interactúe directamente con el hardware. Publica los datos obtenidos del robot y recibe comandos de otros nodos para enviarlos al robot.

#### **Funciones**

* Gestionar el acceso exclusivo al puerto serie.
* Publicar datos de sensores y estado del robot.
* Recibir comandos de movimiento y enviarlos al robot.
* Valorar la prioridades de los mensajes para priorizar la en la comunicacion con el robot mediante el puerto serie.

#### **Interfaces**

* **Parametros**
  + connection\_seria: Define el nombre del puerto serie el cual se utilizará para conectarse al robot
  + connection\_mode: Define el modo de conexion, Full como acceso completo, Safe como modo seguridad, en caso de que un senson de seguridad se active el robot no ctuará (FULL|SAFE)
  + movement\_frecuency: Define la frecuencia con la que publicará los datos de movimiento.
  + security\_frecuency: Define la frecuencia con la que publicará los datos de los sensores.
  + power\_frecuency: Define la frecuencia con la que publicará los datos de los energia.
  + misc\_frecuency: Define la frecuencia con la que publicará los datos de los variados.
  + movement\_filter: Define la lista de sensores de movimiento.
  + security\_filter: Define la lista de sensores de sensores.
  + power\_filter: Define la lista de sensores de energía.
  + misc\_filter: Define la lista de otros sensores, como botones leds…
* **Tópicos**:
  + **Publica**:
    - /roomba/security: Datos de los sensores de caida, choque, distancia…
    - /roomba/movement: Datos de los encoder, distancia recorrida, angulo recorrido obtenidos del robot.
    - /roomba/power: Datos de energia
    - /roomba/misc: Datos de canciones, led, botones..
  + Suscribe:
    - /roomba/motion\_command: Datos de movimiento de motores.
    - /roomba/display\_command: Datos del disply y leds..
    - /roomba/song\_command: Datos de sonidos.
* **Servicios**:
  + - /roomba/system/connect: Establece conexión con el robot.
* **Interacción**:
  + Proporciona datos al nodo de sensores, odometría, control de movimiento, energía y miscelanea.
  + Recibe comandos del nodo de control de movimiento y miscelanea.

### **3.3 Nodo de Se**gurida **(**security\_node**)**

#### **Descripción**

Este nodo procesa y publica datos específicos de los sensores del robot, como sensores de acantilados, colisiones, y proximidad y define la seguridad del robot evitando que problemas al robot o detectando colisiones.

#### **Funciones**

* Leer datos de sensores publicados por el nodo de comunicación.
* Publicar información procesada para otros nodos, como colisiones, objetos cercanos, robot volcado..

#### **Interfaces**

* **Parametros**
  + fit\_cliff\_left: Señal mínima aceptable para considerar que no hay riesgo de caída del sensor izquierdo de acantilado
  + fit\_cliff\_front\_left:Señal mínima aceptable para considerar que no hay riesgo de caída del sensor frontal izquierdo de acantilado
  + fit\_cliff\_right: Señal mínima aceptable para considerar que no hay riesgo de caída del sensor derecho de acantilado
  + fit\_cliff\_front\_right: Señal mínima aceptable para considerar que no hay riesgo de caídal del sensor frontal derecho de acantilado
  + safe\_zone\_radius: Define un radio virtual alrededor del robot dentro del cual no se deben detectar obstáculos.
  + wall\_proximity\_threshold: Define la distancia mínima (valor de señal) para considerar que una pared está demasiado cerca.
  + virtual\_wall\_enabled: Activa o desactiva el uso de paredes virtuales.
  + safety\_check\_rate: Frecuencia en Hz a la que el nodo verifica los datos de los sensores y toma decisiones.
  + publish\_emergency\_events: Activa o desactiva la publicación de eventos de emergencia.
  + state\_security\_frecuency: **Frecuencia a la que se publican los datos de estado de seguridad.**
* **Tópicos**:
  + **Publica**:
  + /roomba/state\_security: Publica eventos de estado

Ejemplo de estados:

* SAFE: Todo está bien.
* OBSTACLE:Detalles Se detectó un obstáculo (choque, pared, etc.).
* CLIFF:Detalles Riesgo de caída detectado.
* DIRT: Se detecto un nivel de suciedad alto.
* EMERGENCY:Detalles Estado de emergencia, se ha solicitado hacer una parada
* COLLISION:Detalles Se ha producido una colisión
* HILLY:Detalles Acidentado
* **Servicios**:
  + /roomba/emergency\_stop: solicita eventos de emergencia

**Interacción**

* Suscribe a los datos del nodo de comunicación.
* Proporciona información al nodo maestro y al nodo de movimiento.

### **3.4 Nodo de Control de Movimiento (**motion\_control\_node**)**

#### **Descripción**

Este nodo gestiona los comandos de velocidad lineal y angular del robot. Traduce los comandos en acciones que el robot puede ejecutar. También controla los motores de los cepillos, el nodo calcula y publica la posición y orientación del robot en el espacio. Genera las transformaciones necesarias para la navegación.

#### **Funciones**

* Procesar comandos de velocidad enviados por el nodo de navegación o el nodo maestro.
* Enviar comandos de movimiento al nodo de comunicación.
* Procesar comandos de controles de cepillos de limpieza
* calcula y publica los datos de odometria

#### Interfaces

* **Parametros**
  + **max\_linear\_velocity: Límite superior para la velocidad lineal (m/s).**
  + **max\_angular\_velocity: Límite superior para la velocidad angular (rad/s).**
  + **odom**\_frecuency**: Frecuencia a la que se publican los datos de odometría (en Hz).**
  + **brush\_motor\_power: Potencia predeterminada de los motores de los cepillos (0-255 o porcentual).**
  + **vacuum\_power: Potencia de succión del aspirador.**
  + **emergency\_stop\_timeout: Tiempo de espera para que el nodo deshabilite los motores después de una emergencia.**
  + **base\_frame\_id: Identificador del marco base del robot (p. ej., "base\_link").**
  + **odom\_frame\_id: Identificador del marco de odometría (p. ej., "odom").**

### linear\_tolerance **Este parámetro se refiere al margen de error permitido en distancia lineal. Es decir, si el robot tiene que moverse hacia un punto específico en el espacio, linear\_tolerance define cuánto puede desviarse en línea recta del objetivo antes de que se considere que llegó correctamente.**

### angular\_tolerance **Este parámetro define el margen de error permitido en la orientación angular del robot.**

* + estate\_motion\_frecuency: **Frecuencia a la que se publican los datos de estado de movimiento**
  + **motion\_command\_frecuency: Frecuencia a la que se publican los datos de estado de movimiento al nodo comunicaciones**
* **Tópicos**:
  + **Suscribe**:
    - /roomba/movement: Datos de movimiento del nodo comunicación.
* **Publica**:
  + - /roomba/odom: Datos de odometría.
    - /roomba/motion\_command: Comandos de movimientos del robot
    - /roomba/state\_motion: Estado de los motores
* **Servicios**:
  + /roomba/emergency\_stop: Detener los motores en caso de recibir un eventos de emergencia
  + /roomba/wheels\_motors: Datos de movimiento del robot, mensajes cmd\_vel
  + /roomba/set\_motors: Datos de los cepillos de limpieza

#### **Interacción**

* Envía comandos de movimiento al nodo de comunicación.
* Recibe datos de movimiento del nodo master.
* Envia datos de odometria al nodo master
* Envia estado de los motores al nodo master
* recibe paradas de emergencias

### **3.6 Nodo de Gestión de Carga (**charging\_node**)**

#### **Descripción**

Este nodo gestiona todos los aspectos relacionados con la energía, como el estado de la batería, la monitorización del voltaje, las corrientes y las sobrecargas. También maneja interacciones relacionadas con fuentes de carga y sensores infrarrojos aplicados a energía..

#### **Funciones**

* Monitorizar el nivel de batería.
* Ordenar al robot regresar a la base de carga.

#### **Interfaces**

* **Tópicos**:
  + **Suscribe**:
    - /roomba/movement: Datos de movimiento del nodo comunicación.
* **Publica**:
  + - **/roomba/power/state/**

**battery\_charge**: Publica la carga actual de la batería (en %).

**battery\_capacity**: Publica la capacidad máxima de la batería.

**voltage**: Publica el voltaje actual de la batería.

**current**: Publica la corriente consumida por el sistema.

**temperature**: Publica la temperatura actual de la batería.

**power\_state**: Publica el estado general de la batería (p. ej., cargando,

* + - **/roomba/power/ir/**

**ir\_char\_omni: Publica la señal infrarroja omnidireccional relacionada con la base de carga.**

**ir\_char\_left: Publica la señal infrarroja detectada a la izquierda.**

**ir\_char\_right: Publica la señal infrarroja detectada a la derecha.**

* + - **/roomba/movement/overcurrent/**

**left\_wheel: Publica si hay sobrecorriente en la rueda izquierda.**

**right\_wheel: Publica si hay sobrecorriente en la rueda derecha.**

**main\_brush: Publica si hay sobrecorriente en el cepillo principal.**

**side\_brush: Publica si hay sobrecorriente en el cepillo lateral.**

* **Servicios**:
  + **/roomba/power/**

**enable\_energy\_saving**: Activa un modo de ahorro de energía.

**disable\_energy\_saving**: Desactiva el modo de ahorro de energía.

**check\_status**: Devuelve el estado completo de la batería (carga, voltaje, temperatura, etc.).

* + **/roomba/emergency\_stop/**

Servicio que detiene todos los motores y dispositivos en caso de sobrecarga o sobrecalentamiento de la batería.

Debe publicar el estado, temperatura, carga de la bateria... voltage y corriente

#### **Interacción**

* Consulta el nivel de batería al nodo de comunicación.
* Coordina con el nodo de navegación para planificar el retorno.

### **3.7 Nodo de Navegación (**navigation\_node**)**

#### **Descripción**

El nodo de navegación utiliza un sensor **LIDAR** y datos de odometría para generar un mapa del entorno y planificar rutas autónomas.

#### **Funciones**

* Crear y actualizar un mapa del entorno.
* Planificar rutas hacia puntos específicos.

#### **Interacción**

* Utiliza datos de odometría y sensores para planificar rutas.
* Envía comandos al nodo de control de movimiento.

### **4. Flujo de Información**

El flujo de información en el sistema sigue una arquitectura modular, donde cada nodo se comunica a través de **tópicos**, **servicios**, y en algunos casos mediante **acciones**, asegurando una interacción eficiente y ordenada. El nodo maestro coordina la comunicación general, mientras que el nodo de comunicación centraliza la interacción con el hardware del robot. A continuación, se describe en detalle el flujo de información entre los nodos.

### **Diagrama del Flujo de Información**

**Nodo Maestro (**master\_node**)**:

* 1. **Solicita y supervisa** la conexión con el robot al **nodo de comunicación serial**.
  2. **Coordina** la ejecución de comandos de navegación, carga y estado general.
  3. **Consulta** el estado del sistema, consolidando información de todos los nodos.

**Nodo de Comunicación Serial (**communication\_node**)**:

* 1. **Envía y recibe** datos directamente del hardware del robot (puerto serie).
  2. **Publica** los datos obtenidos del robot, como sensores y estado general.
  3. **Recibe comandos** de movimiento y otros comandos específicos de nodos como el de control de movimiento.

**Nodo de Sensores (**sensor\_node**)**:

* 1. **Recibe datos crudos** del nodo de comunicación y los procesa.
  2. **Publica información procesada** sobre sensores críticos como batería, acantilados y colisiones.

**Nodo de Control de Movimiento (**motion\_control\_node**)**:

* 1. **Recibe comandos de velocidad** desde el nodo de navegación o el nodo maestro.
  2. **Envía comandos de movimiento** al nodo de comunicación para ejecutar.

**Nodo de Odometría (**odometry\_node**)**:

* 1. **Obtiene datos de movimiento** del nodo de comunicación.
  2. **Calcula y publica** datos de posición y orientación del robot.
  3. **Proporciona transformaciones TF** necesarias para la navegación.

**Nodo de Gestión de Carga (**charging\_node**)**:

* 1. **Consulta el nivel de batería** al nodo de comunicación.
  2. **Envía solicitudes de navegación** al nodo de navegación para retornar a la base.
  3. El Robot tiene un proceso para regresar a la base de forma automatica, añadimos esta opcion tambien
  4. **Informa al nodo maestro** sobre el estado de carga.

**Nodo de Navegación (**navigation\_node**)**:

* 1. **Recibe datos de odometría y sensores** para planificar rutas.
  2. **Envía comandos de velocidad** al nodo de control de movimiento para ejecutar las rutas planificadas.
  + **Coordina con el nodo de carga** para manejar situaciones de batería baja.

**Áreas de Mejora**:

* Falta un nodo dedicado a la interacción con el usuario
* No se especifica el manejo de errores y recuperación
* La gestión del estado global podría estar mejor definida
* Falta un sistema de logging centralizado

**Nuevos Nodos Propuestos**:

1. **User Interface Node** (ui\_node):

* Gestiona la interfaz con el usuario (web, app móvil, etc.)
* Proporciona feedback sobre el estado del robot
* Permite configurar parámetros y programar tareas
* Expone servicios para comandos de voz/texto

1. **Task Manager Node** (task\_manager\_node):

* Coordina la ejecución de tareas de alto nivel
* Mantiene una cola de tareas prioritarias
* Gestiona interrupciones y reanudación de tareas
* Coordina con el navigation\_node para completar objetivos

1. **Diagnostic Node** (diagnostic\_node):

* Monitoriza el estado de todos los nodos
* Registra errores y warnings
* Implementa estrategias de recuperación
* Mantiene métricas de rendimiento

1. **Optimización de Recursos**

A. **Gestión de Memoria**:

* Implementar buffers circulares para datos de sensores
* Limitar el tamaño de los logs
* Usar mensajes ROS 2 comprimidos para datos grandes

B. **Optimización de CPU**:

* Ajustar frecuencias de publicación según necesidad
* Implementar filtrado de datos redundantes
* Usar QoS profiles apropiados para cada tipo de mensaje