# **📘 Documento Técnico – Nodo CommunicationNode**

## **🟦 1. Introducción**

El CommunicationNode es el nodo ROS2 que centraliza la conexión entre el robot (iRobot Create 2) y el resto del sistema. Este nodo:

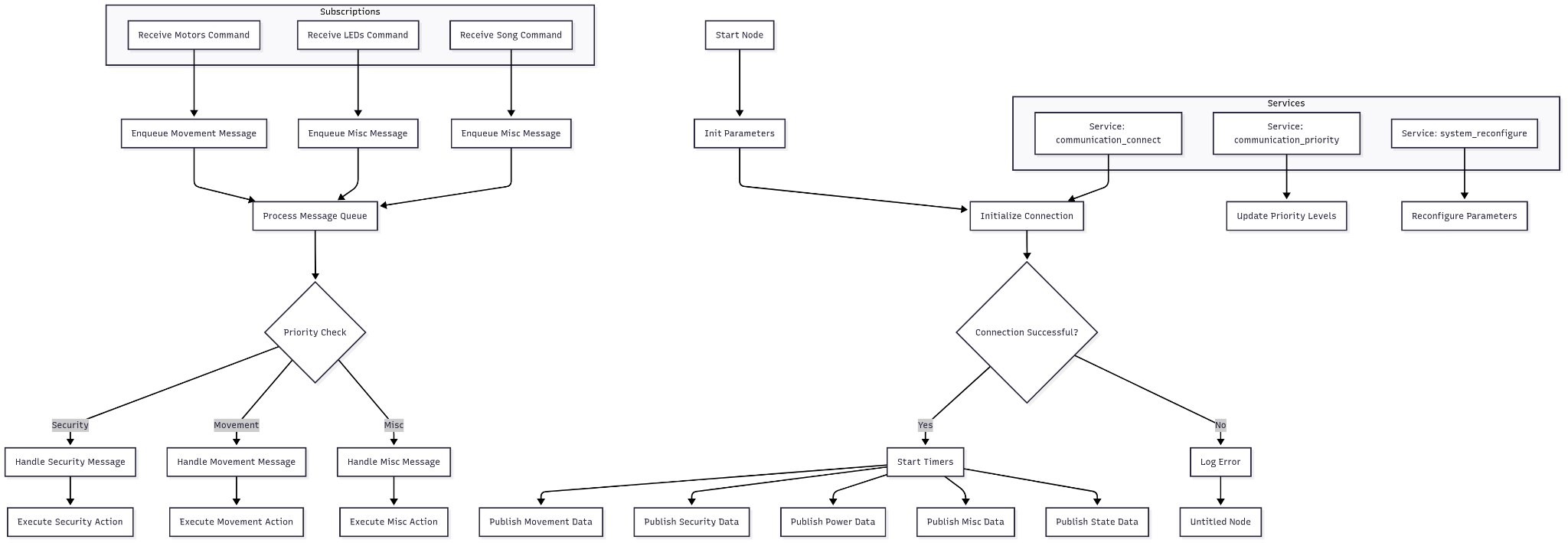
* Se conecta vía puerto serie (/dev/roomba) usando lib\_irobot.
* Publica datos de sensores en tiempo real.
* Recibe comandos de movimiento, LEDs y canciones.
* Procesa estos comandos con un sistema de prioridades.
* Ofrece servicios de reconexión y reconfiguración dinámica.

Es la pieza que garantiza el acceso exclusivo al hardware.

## **🟩 2. Flujo de trabajo general**

El funcionamiento puede describirse así:

1. **Inicialización**
   * Se declaran parámetros ROS2 (puerto serie, frecuencias, filtros).
   * Se inicializa la conexión al robot (connect\_robot()).
2. **Inicio de publicación**
   * Se crean timers que publican:  
     + Movimiento
     + Seguridad
     + Energía
     + Datos misceláneos
     + Estado general
3. **Recepción de comandos**
   * El nodo se suscribe a 3 tópicos de comandos:  
     + /roomba/motion/motors\_command
     + /roomba/misc/leds\_command
     + /roomba/misc/song\_command
   * Cada mensaje recibido se encola con una prioridad.
4. **Procesamiento de comandos**
   * Un timer cada 0.1 segundos extrae la cola de mensajes y los ejecuta en orden de prioridad.
5. **Servicios**
   * Permite:  
     + Reconfigurar frecuencias en tiempo real.
     + Cambiar prioridades.
     + Reconectarse al robot.



## **🟨 3. Funciones principales**

Aquí tienes cada método, con detalle:

| **Función** | **Descripción** |
| --- | --- |
| \_\_init\_\_() | Inicializa el nodo, parámetros, publicadores, suscriptores, servicios y timers. |
| initialize\_connection() | Realiza la conexión inicial al robot y arranca los timers de publicación. |
| start\_timers() | Crea los timers con frecuencias configuradas. |
| publish\_movement\_data() | Lee datos de movimiento (leer\_estado\_movimiento) y publica en /roomba/communication/movement. |
| publish\_security\_data() | Lee sensores de colisión y publica en /roomba/communication/security. |
| publish\_power\_data() | Lee datos de batería y publica en /roomba/communication/power. |
| publish\_misc\_data() | Lee datos misceláneos (cliffs, pared) y publica en /roomba/communication/misc. |
| publish\_state\_data() | Publica estado general (oi\_mode, batería). |
| enqueue\_message(priority, data) | Añade un mensaje a la cola prioritaria. |
| process\_message\_queue() | Procesa mensajes encolados en orden de prioridad. |
| motors\_command\_callback() | Recibe comandos de movimiento y encola mensaje. |
| leds\_command\_callback() | Recibe comandos de LEDs y encola mensaje. |
| song\_command\_callback() | Recibe comandos de música y encola mensaje. |
| handle\_movement\_message() | Ejecuta el comando de movimiento recibido. |
| handle\_misc\_message() | Ejecuta el comando de LEDs o música. |
| handle\_security\_message() | (Placeholder) Procesa mensajes críticos de seguridad. |
| connect\_callback() | Servicio de conexión/reconexión. |
| priority\_callback() | Servicio de cambio de prioridades. |
| reconfigure\_callback() | Servicio de reconfiguración dinámica. |
| destroy\_node() | Detiene el nodo y desconecta el robot. |

## **🟧 4. Servicios**

### **/roomba/communication/connect**

**Función:** Establecer conexión con el robot.  
 **Entrada:**

* port (string): Puerto serie.
* mode (string): FULL o SAFE.  
   **Proceso:**
* Reconecta con connect\_robot().
* Inicia timers de publicación.  
   **Salida:**
* success (bool).
* message (string).

### **/roomba/communication/priority**

**Función:** Cambiar prioridades de procesamiento.  
 **Entrada:**

* message\_type (string): movement, security, misc.
* priority (int).  
   **Proceso:**
* Actualiza el diccionario priority\_levels.  
   **Salida:**
* success (bool).
* message (string).

### **/roomba/system/reconfigure**

**Función:** Cambiar parámetros dinámicos.  
 **Entrada:**

* parameter (string): Nombre del parámetro.
* value (float/int).  
   **Proceso:**
* Usa set\_parameter() para cambiar el valor.  
   **Salida:**
* success (bool).
* message (string).

## **🟦 5. Flujos de datos**

**Publica:**

* /roomba/communication/movement – Movement (distancia, ángulo, encoders).
* /roomba/communication/security – Security (choques, virtual wall).
* /roomba/communication/power – Power (batería, corriente, temperatura).
* /roomba/communication/misc – Misc (cliffs, señales pared).
* /roomba/communication/state – State (modo OI, batería).

**Suscribe:**

* /roomba/motion/motors\_command (comandos de movimiento).
* /roomba/misc/leds\_command (comandos de LEDs).
* /roomba/misc/song\_command (comandos de música).

**Prioridades de mensajes:**

* security – prioridad 1.
* movement – prioridad 2.
* misc – prioridad 3.

## **🟫 6. Mejoras y evoluciones posibles**

Aquí tienes ideas claras de cómo mejorar el nodo:

✅ **Migrar de lib\_irobot a lib\_robot\_roomba:**

* Permite filtros dinámicos reales (sensores activables/desactivables).
* Soporta conexión TCP (socat).

✅ **Completar handle\_security\_message():**

* Implementar paradas de emergencia si hay colisión.

✅ **Persistencia de configuración:**

* Guardar las prioridades y frecuencias en disco al cerrar el nodo.

✅ **QoS y fiabilidad:**

* Configurar perfiles QoS para tolerar desconexiones de red.

✅ **Estadísticas de salud:**

* Publicar métricas de latencia y fallos de sensores.

✅ **Integración con el Nodo Maestro:**

* Añadir servicios /system/status que devuelvan todo el estado consolidado.

✅ **Modo simulación:**

* Permitir arranque sin hardware real.