# **📘 Registro de Versiones – CommunicationNode**

A continuación te hago un recorrido por cada versión, indicando:

* **Cambios clave**
* **Qué aporta respecto a versiones anteriores**
* **Limitaciones o características a tener en cuenta**

## **🟢 v0.0.0 – Prototipo inicial**

**Características principales:**

* Uso directo de la clase Robot simulada.
* Lectura básica de sensores (read\_sensors()).
* Publicación en topics de:  
  + Movement
  + Security
  + Power
  + Misc
  + State
* Timers configurables mediante parámetros ROS2:  
  + movement\_frequency
  + security\_frequency
  + power\_frequency
  + misc\_frequency
  + state\_communication\_frequency
* **No incluye colas de prioridad ni gestión diferida de mensajes.**
* No hay callbacks de suscripción ni comandos.

**Qué aporta:**

* Punto de partida funcional para probar conexión y publicación.

**Limitaciones:**

* No maneja comandos de movimiento ni LEDs.
* Todo es en tiempo real, sin priorización.

## **🟢 v0.0.1 – Primera evolución**

**Características principales:**

* Añade colas de prioridad (queue.PriorityQueue).
* Introduce process\_message\_queue() para procesar mensajes periódicamente.
* Implementa enqueue\_message() que permite insertar mensajes con prioridad.
* Estructura para futuros callbacks (handle\_security\_message, handle\_movement\_message, etc.).
* Inicia la separación de responsabilidades.

**Qué aporta:**

* Primer diseño que contempla gestión de prioridades.
* Base para desacoplar publicación y procesamiento.

**Limitaciones:**

* Los métodos de handle\_\*\_message() solo imprimen logs (no ejecutan acciones reales).

## **🟢 v0.0.2 – Cambio de librería**

**Características principales:**

* Sustituye lib\_irobot por irobot.robots.create2 (Create2).
* Lee sensores y publica directamente desde atributos de Create2.
* Simplifica la inicialización del robot.
* Mantiene la arquitectura básica de timers.
* **No contiene callbacks de suscripción.**

**Qué aporta:**

* Prueba de integración con la librería oficial irobot.

**Limitaciones:**

* Se pierde el sistema de colas de prioridad.
* No gestiona mensajes de comandos.
* Es una versión de exploración.

## **🟢 v0.0.3 – Retorno a lib\_irobot con mejoras**

**Características principales:**

* Retoma lib\_irobot.
* Recupera colas de prioridad y process\_message\_queue().
* Añade filtros de sensores configurables mediante parámetros (movement\_filter, etc.).
* Soporta servicios:  
  + /communication/connect
  + /communication/priority
  + /system/reconfigure
* Cada topic tiene su propio timer con frecuencia parametrizable.
* Implementa callbacks de suscripción:  
  + motors\_command\_callback
  + leds\_command\_callback
  + song\_command\_callback
  + (aunque la lógica real no está desarrollada en handle\_\*\_message()).

**Qué aporta:**

* Vuelve a la librería que ya tienes probada.
* Diseño más modular y extensible.

**Limitaciones:**

* La lógica de manejo de comandos aún es esquelética.
* No gestiona filtros activamente.

## **🟢 v0.0.4 – Consolidación del sistema de colas**

**Características principales:**

* Basado en lib\_irobot.
* **Todos los callbacks de comandos encolan los mensajes** según prioridad:  
  + movement prioridad 2
  + misc prioridad 3
  + security prioridad 1
* process\_message\_queue() procesa la cola cada 0.1 s.
* Los métodos handle\_\*\_message() listos para implementar acciones reales.
* Sistema de timers como en v2.
* Se afianza la separación entre **recepción, encolado y ejecución**.

**Qué aporta:**

* Diseño limpio y consistente con colas de prioridad.
* Base estable para crecimiento.

**Limitaciones:**

* Aún no implementa lógica real en handle\_\*\_message().

## **🟢 v0.0.5 – Versión más madura**

**Características principales:**

* También usa lib\_irobot.
* Incluye:  
  + Sistema de colas robusto.
  + enqueue\_message() y process\_message\_queue().
  + Publicadores configurables.
  + Suscripciones para comandos.
  + Servicios de conexión, reconfiguración y prioridades.
* **Implementa callbacks completos de movimiento, LEDs y música:**
  + motors\_command\_callback:  
    - Divide el string recibido y llama a moverse\_manual().
  + leds\_command\_callback:  
    - Procesa colores e intensidad.
  + song\_command\_callback:  
    - Genera la lista de notas y llama a reproducir\_musica().
* Publica datos en topics y maneja reconexión dinámica.
* Implementa reconfigure\_callback() que actualiza parámetros en tiempo real.

**Qué aporta:**

* Es la versión **más avanzada, robusta y lista para producción**.
* Todas las piezas están conectadas: publicación, recepción, prioridades, acciones.
* La arquitectura coincide con tu **mapa de nodos Fase 2**.

**Limitaciones:**

* Aunque muy completa, sigue usando lib\_irobot, por lo que si quieres filtros dinámicos y conexión TCP necesitarías migrar a lib\_robot\_roomba.

# **📊 Tabla Resumen de Evolución**

| **Versión** | **Librería Principal** | **Colas de Prioridad** | **Filtros de Sensores** | **Manejo de Comandos** | **Servicios Reconfigurables** | **Observaciones** |
| --- | --- | --- | --- | --- | --- | --- |
| v0.0.0 | Robot simulada | No | No | No | No | Prototipo inicial |
| v0.0.1 | Robot simulada | Sí | No | Parcial (estructura) | No | Primer paso hacia modularidad |
| v0.0.2 | Create2 (irobot) | No | No | No | No | Exploración con Create2 |
| v0.0.3 | lib\_irobot | Sí | Sí | Parcial | Sí | Consolidación inicial |
| v0.0.4 | lib\_irobot | Sí | Sí | Parcial (con encolado) | Sí | Diseño robusto con colas |
| v0.0.5 | lib\_irobot | Sí | Sí | Sí (completo) | Sí | Versión madura y lista para expandir |