Manual de Desarrollo para RosRobotController

Configuración, Compilación y Flasheo en Linux

Equipo de Ingeniería C-ROS

19 de febrero de 2026

Índice

1. Introducción

Este documento establece el procedimiento estándar para desarrollar firmware en la tarjeta **RosRobotController V1.1** (STM32F407VET6) utilizando herramientas Open Source en Linux. El flujo de trabajo se divide en:

1. **STM32CubeMX:** Configuración de hardware y generación de estructura.
2. **STM32CubeIDE:** Programación lógica y generación del binario (.hex).
3. **stm32flash:** Carga del firmware mediante terminal.

2. Fase 1: Configuración de Hardware (STM32CubeMX)

Esta fase define los planos del microcontrolador: qué pines hacen qué función y a qué velocidad trabaja el cerebro.

2.1. Selección del Microcontrolador y Configuración

1. Abrir el programa **STM32CubeMX**.
2. En la pantalla principal, hacer clic en el botón **Access to MCU Selector**.
3. En la barra de búsqueda **Commercial Part Number**, teclear **STM32F407VETx**.
4. Seleccionar el microcontrolador en la lista inferior y presionar el botón **Start Project** (arriba a la derecha).
5. **Configuración del Oscilador (RCC):** En el menú lateral izquierdo, desplegar la categoría **System Core** y hacer clic en **RCC**. En el panel central, cambiar la opción **High Speed Clock (HSE)** a **Crystal/Ceramic Resonator**. Esto habilita el uso del cristal de cuarzo físico de la tarjeta.
6. **Configuración del Reloj:** Ir a la pestaña *Clock Configuration*. Aquí se define la velocidad del sistema aprovechando el oscilador externo (HSE) recién configurado.

2.2. Declaración de Pines (Entradas/Salidas)

Ir a la pestaña *Pinout & Configuration*. Aquí se mapea el hardware físico (LEDs, motores, sensores) a los pines del microcontrolador.

1. Hacer **clic izquierdo** sobre el pin deseado en el dibujo del chip.
2. Seleccionar el comportamiento del pin en el menú desplegable:
   - `GPIO_Output`: Para actuar sobre el entorno (encender LEDs, activar motores, buzzer).
   - `GPIO_Input`: Para leer el entorno (botones, sensores simples).

3. Hacer **clic derecho** sobre el mismo pin, seleccionar **Enter User Label** y asignarle un nombre descriptivo (ej. `LED1`, `BTN_START`). Usar nombres claros facilita la programación posterior.

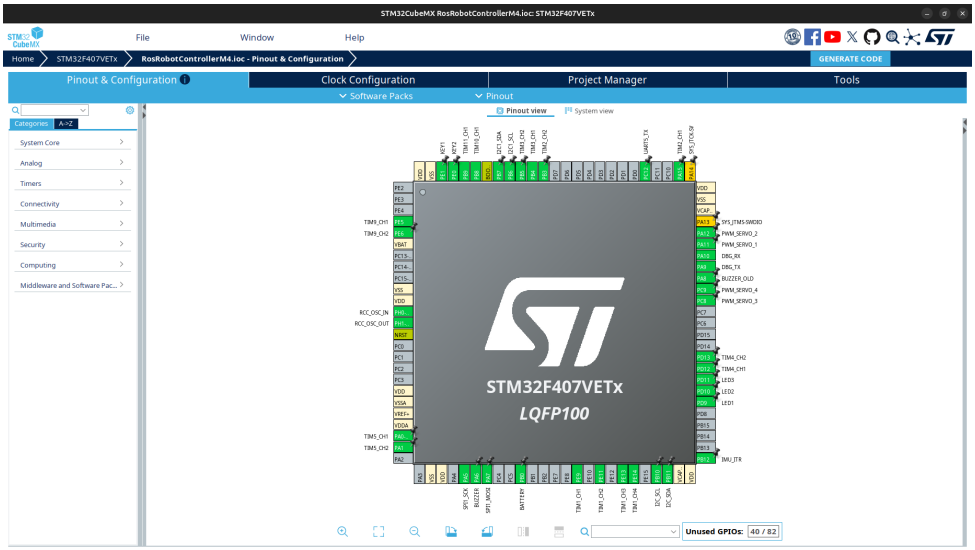

Figura 1: Vista de configuración de pines en CubeMX

2.3. Generación del Código y Opciones de Toolchain

1. Ir a la pestaña **Project Manager**.
2. Asignar un **Project Name** y seleccionar la carpeta de destino.
3. **Selección de Toolchain / IDE:**
   - Seleccionar **STM32CubeIDE** si se usará este entorno para programar.
   - Seleccionar **Makefile** si el equipo prefiere compilar desde la terminal con un compilador externo (ej. GCC de ARM).
4. Hacer clic en el botón **GENERATE CODE**.

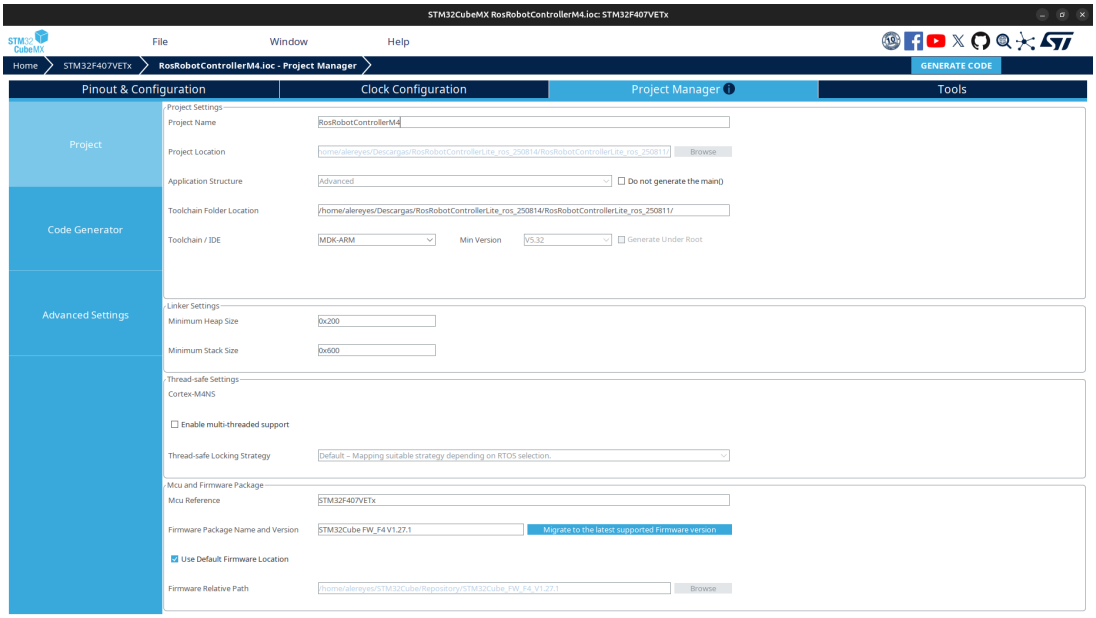

Figura 2: Selección correcta del Toolchain STM32CubeIDE

¿Qué pasa si necesito agregar un pin después?

El flujo de STMicroelectronics está diseñado para ser iterativo. Si más adelante necesitas agregar un motor o cambiar un pin:

- Abre el archivo `.ioc` del proyecto (que vuelve a abrir CubeMX).
- Haz la modificación del pin o periférico.
- Vuelve a presionar **GENERATE CODE**.
- **Nota de seguridad:** El código *C* se actualizará automáticamente incluyendo el nuevo pin. Todo el código que tú hayas escrito se conservará intacto **siempre y cuando** esté escrito dentro de los comentarios delimitadores `/* USER CODE BEGIN ... */` y `/* USER CODE END ... */`.

3. Fase 2: Programación y Compilación (STM32CubeIDE)

3.1. Apertura del Proyecto

Una vez que CubeMX termina de generar el código, aparecerá una ventana emergente.

1. Hacer clic en **Open Project**. Esto abrirá automáticamente el STM32CubeIDE con tu proyecto cargado.
2. *Alternativa manual:* Si se cerró la ventana, abrir STM32CubeIDE, ir a `File` → `Open Projects from File System`, hacer clic en **Directory** y seleccionar la carpeta del proyecto recién creada.

3.2. Habilitar Generación de Hexadecimal

Por defecto, el IDE no siempre genera el archivo `.hex` necesario para flashear en Linux con nuestra herramienta. Se debe configurar manualmente:

1. En el explorador de la izquierda, hacer clic derecho sobre el nombre del proyecto y seleccionar **Properties**.
2. Navegar a `C/C++ Build` → `Settings` → `MCU/MPU Post build outputs`.
3. Marcar la casilla: **Convert to Intel Hex file (-O ihex)**.
4. Clic en **Apply and Close**.

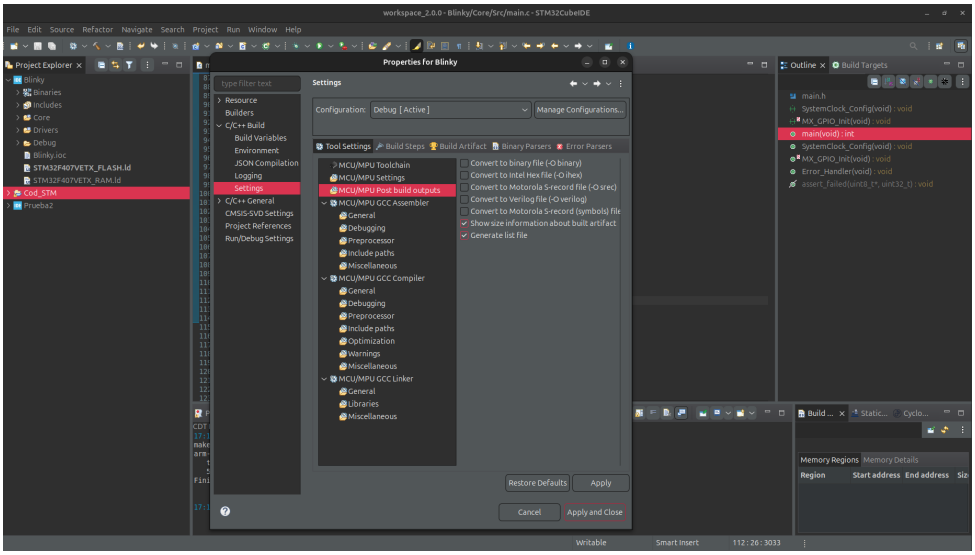

Figura 3: Habilitación de salida .hex en las propiedades del proyecto

3.3. Ubicación y Desarrollo del Código

En el explorador del proyecto a la izquierda, expandir las carpetas y navegar a: `Core` → `Src` → `main.c`. Haremos doble clic para abrirlo.

Buscar el bucle infinito `while (1)` y escribir la lógica de control. Recordar siempre escribir dentro de las etiquetas de usuario.

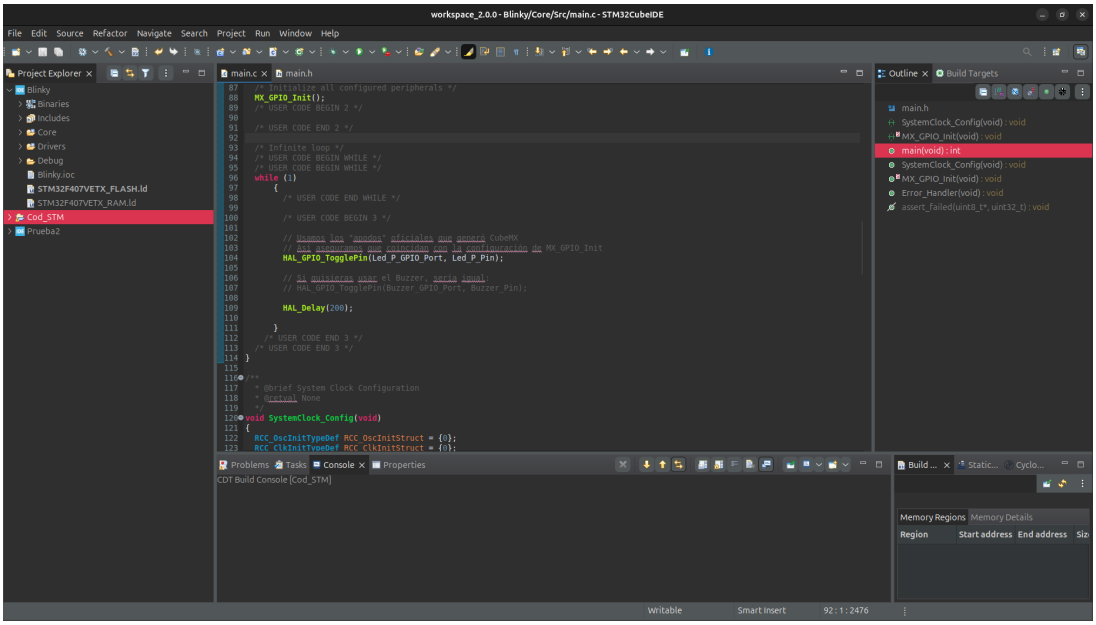

Figura 4: Ejemplo de estructura de código en main.c

3.4. Compilación

Para convertir nuestro código *C* en lenguaje máquina:

1. Hacer clic en el icono del **Martillo** (Build Debug) en la barra de herramientas superior.

2. Verificar en la consola inferior (*Console*) que el proceso finalice con el mensaje: `Build Finished. 0 errors`.

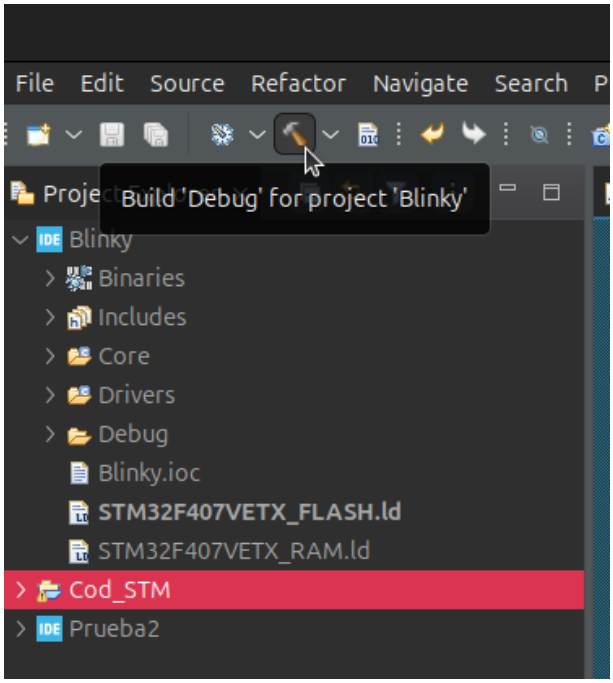

Figura 5: Compilación exitosa del proyecto

4. Fase 3: Carga del Firmware (Terminal Linux)

4.1. Ubicación del Archivo

El archivo `.hex` generado se encuentra dentro de la subcarpeta `Debug` del directorio de tu proyecto.

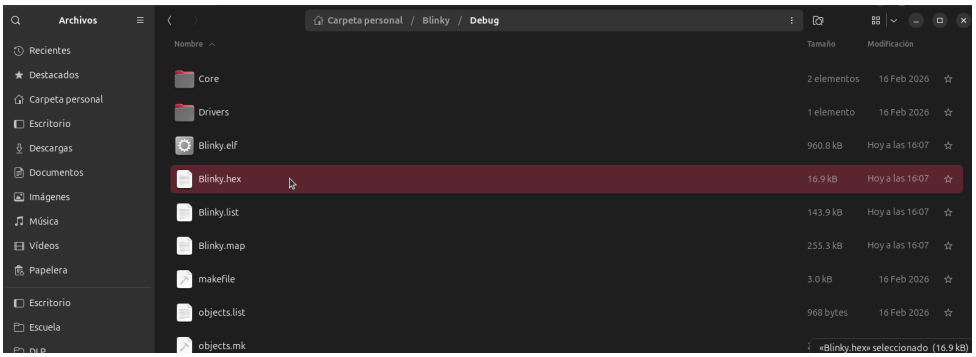

Figura 6: Archivo .hex generado listo para carga

4.2. Procedimiento de Flasheo

1. **Conexión:** Conectar el robot vía USB (puerto central).
2. **Modo Bootloader (Hardware):**
   - Mantener presionado el botón **Custom button,** (o también llamado BOOT).
   - Presionar y soltar **RESET**.
3. **Ejecutar Comando:** Abrir una terminal en la carpeta `Debug` y ejecutar:
4. Soltar **Custom button** una vez cargado el programa.

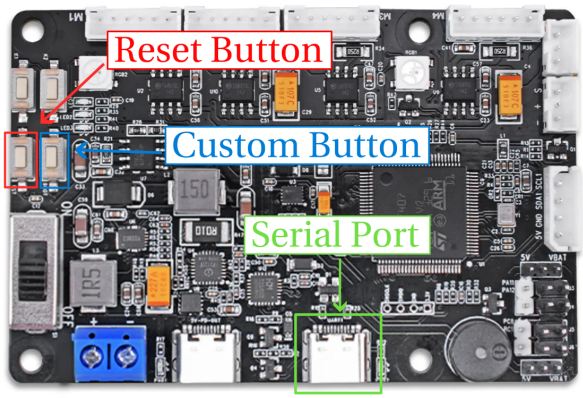

Figura 7: Componentes usados para el flasheo.

```
stm32flash -w NombreArchivo.hex -v -g 0x0 -R /dev/ttyACM0
```

Desglose del comando:

- `-w`: Escribir el archivo especificado.
- `-v`: Verificar la escritura.
- `-g 0x0`: Iniciar ejecución desde la dirección 0 de memoria.
- `-R`: Reiniciar el microcontrolador al terminar.
- `/dev/ttyACM0`: Puerto serial del robot (verificar con `ls /dev/tty*`).

4.3. Nota de Desbloqueo (Chips Nuevos)

Si el proceso falla con error de lectura (`Read Protected`) o se detiene al 4 %, ejecutar primero el comando de desbloqueo total para formatear el chip:

```
sudo stm32flash -k /dev/ttyACM0
```