SP1 Mk1 Firmware - Documentación Técnica

# Introducción

El SP1 Mk1 es el módulo físico principal del sistema Tangible MIDI Controller. Está basado en un Arduino Micro y se encarga de escanear 108 controles físicos (91 potenciómetros y 17 interruptores), enviar sus valores como mensajes MIDI, y comunicarse con otros módulos como el SP1 Mk2 (ESP32-S3) y el futuro Mk3 (Teensy).

# Arquitectura del Firmware

El firmware está organizado de forma modular en archivos .h/.cpp independientes que gestionan cada parte del sistema:

* config.h — Configuración global de pines, UART y flags de depuración
* control\_map.h / control\_map.cpp — Tabla de controles en PROGMEM
* hardware.h / hardware.cpp — Escaneo físico, detección de cambios
* midi.h / midi.cpp — Envío y recepción MIDI (USB, DIN, UART)
* routing.h / routing.cpp — Gestión de comunicación con Mk2
* mux\_input\_pins.h / mux\_input\_pins.cpp — Pines EN de multiplexores
* FIRMWARE\_SP1\_Mk1\_MICRO\_ARDUINO.ino — Loop principal y enrutamiento

# Esquema de conexiones físicas

El firmware está organizado de forma modular en archivos .h/.cpp independientes que gestionan cada parte del sistema:

* config.h — Configuración global de pines, UART y flags de depuración
* control\_map.h / control\_map.cpp — Tabla de controles en PROGMEM
* hardware.h / hardware.cpp — Escaneo físico, detección de cambios
* midi.h / midi.cpp — Envío y recepción MIDI (USB, DIN, UART)
* routing.h / routing.cpp — Gestión de comunicación con Mk2
* mux\_input\_pins.h / mux\_input\_pins.cpp — Pines EN de multiplexores
* FIRMWARE\_SP1\_Mk1\_MICRO\_ARDUINO.ino — Loop principal y enrutamiento

# Funciones clave del firmware

* setup() — Inicializa el sistema y envía una nota de test
* loop() — Ejecuta el escaneo, detección de cambios y reenvío MIDI
* scanControls() — Detecta cambios físicos en los controles
* sendCC() — Envía mensajes Control Change por USB y DIN
* updateMIDI() — Procesa comandos UART entrantes (Mk2 → Mk1)
* processMIDIMessage() — Reenvía mensajes MIDI a todos los puertos
* checkMk2Connection() — Verifica si Mk2 está conectado
* checkIncomingUSBMIDI() — Detecta y reenvía mensajes USB entrantes

# Modo autónomo

Si el Mk1 no detecta conexión con Mk2, funciona de forma independiente. Se aplica un filtro de valores mínimos/máximos y se envían mensajes CC directamente desde los potenciómetros.

# Comandos UART recibidos desde Mk2

* #SET:<idx>:<val>:<chn> — Asigna valor a control y lo envía como CC
* #CHANNEL:<idx>:<chn> — Asigna un nuevo canal MIDI a un control
* #GROUPCH:<grupo>:<chn> — Cambia canal MIDI de un grupo completo
* #MUTE / #UNMUTE — Silencia o activa la salida MIDI
* #ID? — Responde con SP1\_MK1\_V1.1

# Estado actual del proyecto

* Código completo estructurado y modular
* Compila sin errores ni advertencias
* ControlMap en PROGMEM con acceso robusto
* Envío y ruteo MIDI funcionando (USB, DIN, UART)
* UART con Mk2 operativa
* Sistema de muteo global (muteOutput)
* README.md actualizado
* Documento .docx completo y verificado

Pruebas realizadas en modo autónomo y esclavo

Guía de Conexiones y Construcción del SP1 Mk1 (Arduino Micro)

# Introducción

Este documento sirve como guía práctica para construir correctamente el sistema de control SP1 Mk1 basado en Arduino Micro, utilizando el esquema "Esquema completo Conexiones MICRO.pdf". Abarca la interpretación del esquema, la asignación de pines, las conexiones físicas de multiplexores y controles, y consejos útiles para evitar errores.

# Arquitectura General

El SP1 Mk1 se basa en un Arduino Micro y nueve multiplexores CD74HC4067, organizados para escanear hasta 108 entradas analógicas. Los multiplexores comparten las líneas de selección S0–S3, pero tienen pines EN independientes, permitiendo la selección secuencial.

- 9 multiplexores CD74HC4067  
- 91 potenciómetros (knobs)  
- 17 interruptores (switches)  
- Cada multiplexor permite hasta 16 entradas (C0–C15).  
- Las salidas de todos los MUX se unifican mediante una línea SIG común conectada a la entrada analógica A0 del Arduino.

# 2. Pines del Arduino Micro

|  |  |  |
| --- | --- | --- |
| Función | Pin Arduino | Uso |
| MUX\_S0 | D2 | Selector de canal - bit 0 |
| MUX\_S1 | D3 | Selector de canal - bit 1 |
| MUX\_S2 | D4 | Selector de canal - bit 2 |
| MUX\_S3 | D5 | Selector de canal - bit 3 |
| MUX\_ANALOG\_SIGNAL | A0 | Lectura de la señal analógica de todos los MUX |
| EN MUX 0 | D7 | Enable para muxA0 |
| EN MUX 1 | D8 | Enable para muxA1 |
| EN MUX 2 | D9 | Enable para muxA2 |
| EN MUX 3 | D10 | Enable para muxA3 |
| EN MUX 4 | D11 | Enable para muxA4 |
| EN MUX 5 | D12 | Enable para muxA5 |
| EN MUX 6 | D13 | Enable para muxA6 |
| EN MUX 7 | D14 | Enable para muxA7 |
| EN MUX 8 | D15 | Enable para muxA8 |
| UART RX Mk2 | D16 | Comunicación serie con el módulo SP1 Mk2 |
| UART TX Mk2 | D17 | Comunicación serie con el módulo SP1 Mk2 |

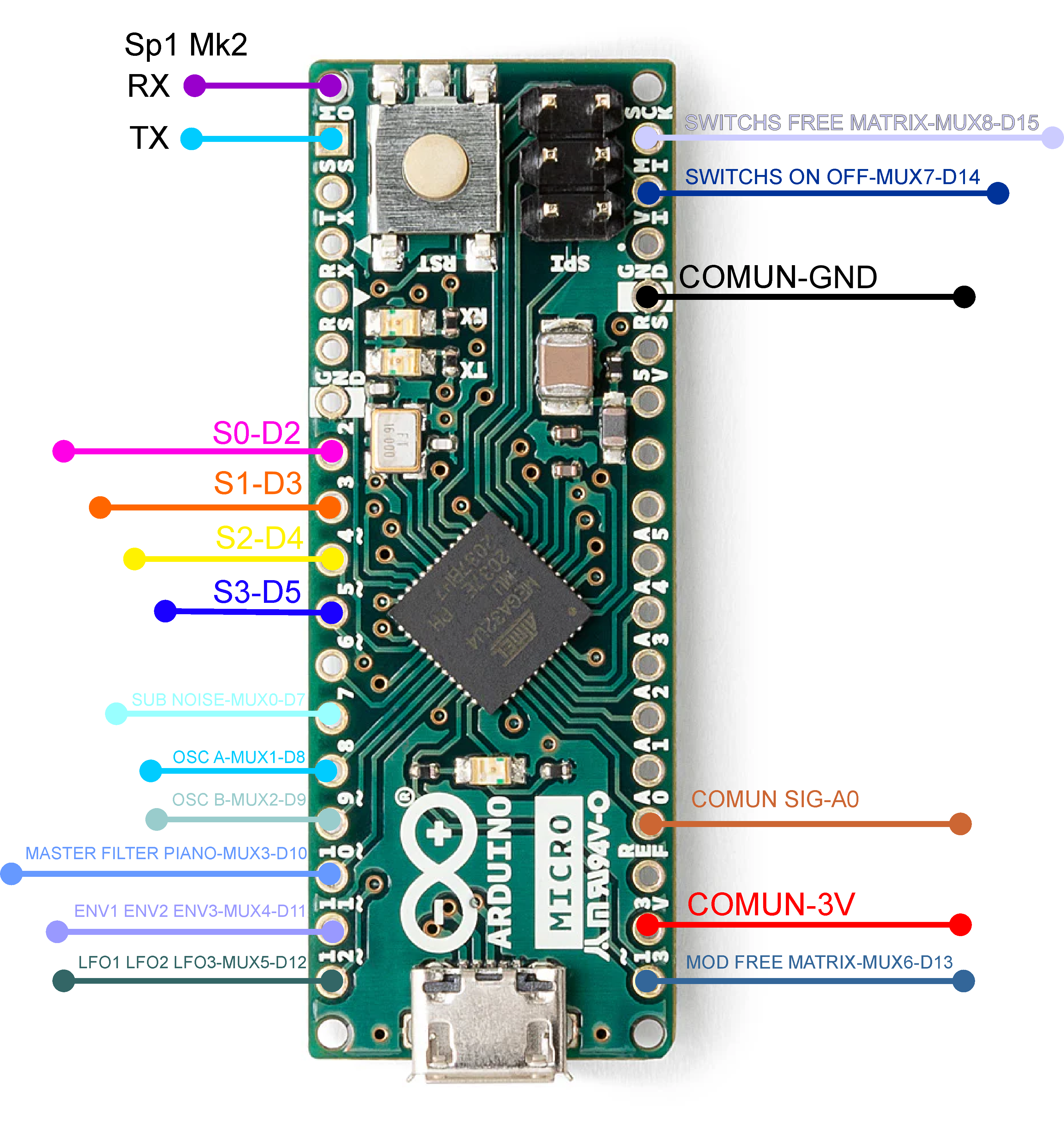


Imagen: Pinout del Arduino Micro

# Conexión de los Multiplexores

Cada multiplexor CD74HC4067 tiene 16 entradas que se escanean secuencialmente. Las líneas S0–S3 están conectadas en paralelo a todos los MUX. Cada MUX tiene su pin EN conectado a un pin dedicado del Arduino (D7–D15), lo cual permite activar uno por vez.

Todos los multiplexores CD74HC4067 comparten las siguientes líneas comunes conectadas al Arduino Micro:  
- Líneas de selección S0–S3 (D2–D5)  
- Línea de señal común SIG → A0  
- Alimentación COMUN-3V  
- Tierra COMUN-GND  
  
Cada multiplexor tiene su propia línea de habilitación (EN) que debe ir conectada a un pin digital exclusivo entre D7 y D15.  
  
⚠️ Solo debe estar habilitado un multiplexor a la vez mediante un nivel lógico LOW en su línea EN. Los demás deben permanecer en HIGH para evitar interferencias.

# Agrupación functional

Los controles están organizados en 10 grupos funcionales: SUB, OSC1, OSC2, MIX, ENV1, ENV2, LFO1, LFO2, FX y SWITCHES. Cada grupo puede ser dirigido a un canal MIDI distinto desde el Mk2.

|  |  |  |
| --- | --- | --- |
| MUX | Grupo Funcional | Pin EN (Arduino) |
| MUX0 | SUB + NOISE | D7 |
| MUX1 | OSC A | D8 |
| MUX2 | OSC B | D9 |
| MUX3 | MASTER, FILTER, PIANO | D10 |
| MUX4 | ENV1, ENV2, ENV3 | D11 |
| MUX5 | LFO1, LFO2, LFO3 | D12 |
| MUX6 | MACROS, FREE MATRIX | D13 |
| MUX7 | SWITCHES ON/OFF | D14 |
| MUX8 | SWITCHES MATRIX | D15 |

# Etiquetado y Documentación

Se recomienda etiquetar cada cable y control físico siguiendo el orden del archivo control\_map.h. Los nombres como 'OSC1\_CUTOFF' o 'ENV2\_ATTACK' ayudan a identificar fácilmente cada control en el firmware y en las herramientas de edición externa.

Cada control físico (knob o switch) debe etiquetarse con:  
- El canal que ocupa en el multiplexor (C0 a C15)  
- El nombre funcional (ej. Filter CutOff)  
- El índice de firmware (ej. Knob-35 o Switch-91)  
  
⚠️ El esquema PDF dibuja un solo componente por multiplexor como ejemplo visual, pero todos los controles están correctamente referenciados mediante etiquetas.  
Estas etiquetas permiten relacionar cada entrada del mux con su descripción funcional y canal MIDI en el firmware.

# Consejos prácticos de construcción

1. Utiliza regletas, conectores hembra y cables dupont para evitar soldaduras fijas.  
2. Etiqueta cada grupo de cables por MUX (ej. MUX0 = amarillo, MUX1 = rojo, etc.)  
3. Usa cables cortos para SIG y líneas S0–S3 para evitar ruido.  
4. Añade condensadores de desacoplo de 100nF entre VCC y GND cerca de cada multiplexor.  
5. Verifica continuidad y cortos con multímetro antes de alimentar el sistema.  
6. No uses los pines D0 y D1 (reservados para USB/Serial).  
7. Para depuración, usa Serial por USB (Serial.print) y UART independiente con Mk2 (Serial1).