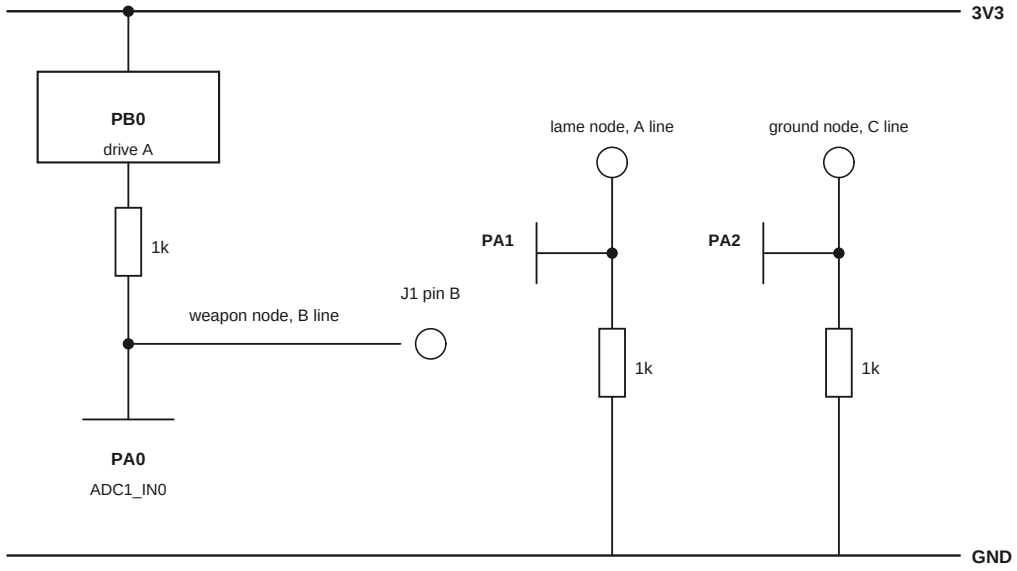
Three weapon fencing scoring box	Schematic
STM32F401 front end	Doc 10 Rev A 2026-08-28

Front end, one channel of seven

Every fencing line gets one 1 k resistor. The five sense lines are pulled permanently to ground. The two weapon lines are pulled from a GPIO instead of from the rail, which is the change that makes blade-to-blade contact visible.



How a touch is recognised

A weapon line pulled to 3V3 through 1 k, joined to a sense line pulled to ground through 1 k, sits at half rail. Both pins read about 2048 at the same instant. Nothing else in the system produces that pattern: an untouched weapon reads 4095 and an untouched sense line reads near zero.

The box therefore knows not just that contact happened but what was touched, because it can see which two nodes moved together. Contact resistance on a damp blade moves both readings the same way, which is why the mid band is wide.

Two phase drive

Phase	PB0 drive A	PB1 drive B	What is measurable
A	HIGH, pull up	LOW, pull down	Everything fencer A's blade touches, including fencer B's blade
B	LOW, pull down	HIGH, pull up	The mirror image

> Without this, both weapon lines sit at the same rail and blade-to-blade contact is invisible. That is why no hobby box implements sabre whip-over. Alternating the drive costs two GPIO pins and nothing else.

Complete connection list

Net	MCU pin	ADC ch	Resistor
Fencer A weapon, B line	PA0	IN0	1k to PB0
Fencer A lame, A line	PA1	IN1	1k to GND
Fencer A ground, C line	PA2	IN2	1k to GND
Fencer B weapon, B line	PA3	IN3	1k to PB1
Fencer B lame, A line	PA4	IN4	1k to GND
Fencer B ground, C line	PA5	IN5	1k to GND
Piste	PA6	IN6	1k to GND
Drive A	PB0	-	-
Drive B	PB1	-	-

Indicators and controls

Function	Pin	Part	Series R
On target A, green	PB12	5 mm LED	220R
Off target A, white	PB13	5 mm LED	220R
Off target B, white	PB14	5 mm LED	220R
On target B, red	PB15	5 mm LED	220R
Fault A	PA8	5 mm LED	220R
Fault B	PA9	5 mm LED	220R
Mode foil / epee / sabre	PB3 / PB4 / PB5	5 mm LED	220R each
Buzzer	PB10	Active piezo	-
Mode button	PB6	Momentary NO	internal pull-up
Reset button	PB7	Momentary NO	internal pull-up
Heartbeat	PC13	on-board LED	-

> Avoid PA13 and PA14, they are the SWD debug pins, and PB2, which is BOOT1. Analog pins on the F401 are not 5 V tolerant, so nothing on the fencing side may ever be driven above 3.3 V.

Bill of materials

Qty	Part	Note
1	WeAct Black Pill, STM32F401CCU6	Buy two, they are cheap
1	ST-Link V2 clone	Backup to USB DFU
7	3-pin fencing socket, or 4 mm banana jacks	6 fencer + 1 piste terminal
1	Binding post or 4 mm terminal	Piste ground
7	1 k resistor, 1 percent	The sense network. Match these
9	220 R resistor	LED limiting
2	5 mm LED, white	Off target
1	5 mm LED, green	On target A
1	5 mm LED, red	On target B
5	5 mm LED, assorted	3 mode, 2 fault
1	Active piezo buzzer	5 V or 3.3 V type
2	Momentary pushbutton, panel	Mode, reset
1	Perfboard, 0.1 inch	
1	Heavy alligator clip lead	Piste ground
-	22 AWG stranded wire	

For the test rig, add an Arduino Nano, an eight channel USB logic analyzer, and seven 2.2 k resistors. Those three items are what let you verify the timings instead of trusting them.

Three weapon fencing scoring box	Firmware
STM32F401 front end	Doc 11 Rev B 2026-08-29

Files

File	Runs on	What it is
scoring_logic.h / .cpp	both	All scoring decisions. Plain C++, no Arduino dependency, so it can be tested on a PC
test_scoring_logic.cpp	PC	25 unit tests over the state machine
scoring_box_f401.ino	Black Pill	Plumbing: ADC, drive phases, lamps
hit_simulator_nano.ino	Nano	Generates contacts of exact duration

Putting the decisions in a plain C++ file is the point. Fencing logic is all about microsecond timing, and timing bugs are close to impossible to find by waving a sword at a box. On a PC the same code runs against synthetic contacts of exact length, in a second, repeatably.

Build and run the tests with: `g++ -std=c++17 -O2 -o test test_scoring_logic.cpp scoring_logic.cpp && ./test`
No g++ installed? The zig toolchain from pip works identically: `pip install ziglang` then `python -m ziglang c++ -std=c++17 -O2 -o test test_scoring_logic.cpp scoring_logic.cpp`

Sabre wiring reality — the Rev B correction

Rev A treated a sabre weapon line joined to its own C line as an equipment fault and as a reason to suppress hits. That was wrong, and it made sabre mode unusable with a real weapon. The FIE Material Rules settle it:

> “The two sockets of the bodywire plug must be in direct contact with the body of the guard, making a closed electrical circuit through the bodywire, the spool and the cable connecting the spool to the scoring apparatus.” — FIE Material Rules, m.24.4 (sabre guard)

So on a healthy sabre the B and C lines are permanently commoned through the guard, at *both* weapons. Three consequences, all in the Rev B code:

Change	Why
Sabre fault is now a **floating** weapon line	W at the rail means the cord, socket or weapon is disconnected. W joined to own C is the normal resting state, never a fault.
toOwnGround and toOppGround no longer veto sabre hits	Own ground is always joined; blade-to-blade contact joins the opponent's B *and* C at once. Cuts delivered during blade contact are decided by the whip-over window, not a blanket veto.
LEVEL_LOW_MAX lowered from 900 to 600	The whip-over pile-up (drive + own C + opponent's B, C and lamé) is one pull-up against four pulldowns = 1/5 rail ≈ 819 counts, which the old 900 floor read as “not joined”. Untouched sense lines read near zero, far below 600.

What the tests establish — 25 checks, all passing

Test	Result
Mid-band sanity: sabre pile-up level (819) joined; near-ground (300) not	2 checks pass
Foil 13 ms on-target contact / 16 ms contact	rejected / registers
Foil floor hit, hit on opponent guard	no light at all
Foil point on insulating cloth / 8 ms cloth slide onto lamé	off target / ON target only
Epee 1 ms contact / 4 ms contact	rejected / registers
Epee floor hit on a metal strip	no light
Epee second hit at 30 ms / at 60 ms	double / locked out
Sabre 1.2 ms lamé contact, with B-C commoned throughout	registers
Sabre at rest (W joined own C) 4 s / floating weapon line 4 s	no fault, no light / fault indicated
Sabre whip-over at 8 ms / whip landing 1 ms after blades part / cut after 25 ms	suppressed / suppressed / still scores
Sabre hit on opponent guard, blade on piste	no light
Foil / epee / sabre lockout gap vs spec (300 / 45 / 170 ms)	error 0 µs, all three
Epee stuck point held 4 s	fault indicated

Three deliberate departures from the common designs

First, the piste and both C lines are read. In the widely copied Arduino sketch the ground pins are declared and wired but never sampled, so every epee hit on a metal strip scores. Here a floor hit produces nothing in all three weapons.

Second, lockout is measured from the moment a hit registers rather than from the start of contact. Measuring from contact start shortens the window by the contact time, which for foil is 14 ms of a 300 ms budget.

Third, nothing blocks. The reference sits in delay() for 3.1 seconds after every hit and is deaf during that time; this latches the lamps and keeps scanning.

Whip-over: mostly confirmed, one residual interpretation

The Material Rules (sabre annexe) confirm what Rev A only assumed: hits through the blade register between 0 and 4 ms of blade contact and are prevented between 4 and 15 ms (+5 ms), **measured from the start of blade contact** — exactly how the code measures it. The one remaining interpretation is the 2 ms tail that keeps rejecting a whip landing just after the blades part. Compare against a known good box before trusting it.

Bring-up order

- [ ] Run the PC tests. They should report 25 passed before you touch hardware.
- [ ] Flash the F401 with nothing connected. Watch the serial scan rate; expect tens of thousands per second. If it reads a few thousand, the fast ADC path is not working.
- [ ] Short a weapon node to a sense node with a jumper and confirm both read mid rail on the serial output before trusting any weapon logic.
- [ ] Wire the Nano simulator through its 2.2 k resistors (share grounds), hold the box's mode button while powering it up to freeze the drive phase, then run the foil suite from the simulator and watch the lamps.
- [ ] Run the sabre suite. The first test holds the commoned B-C resting state for three seconds: no lamp and no fault may appear.
- [ ] Put the logic analyzer on the lamp pins and the simulator trigger pin, and measure the real lockout. It should match the numbers above to within one scan period.
- [ ] Only then connect real weapons.

Why and how

The base design's 5 mm indicator LEDs run a few milliamps from GPIO pins - fine on a desk, invisible across a gym. Real scoring boxes have lamp *panels*. This upgrade keeps the exact same four GPIO pins and firmware, but each pin now switches a logic-level MOSFET driving a segment of 5 V LED strip: a glowing zone per lamp, running hundreds of milliamps from the 5 V supply instead of the microcontroller.

> **Zero firmware changes.** The pins still just go HIGH; the MOSFET does the lifting. Deliberately NOT addressable (WS2812/NeoPixel) strips: updating those disables interrupts for a millisecond or more per refresh, which would blow the 10-30 µs scan loop that all the timing verification rests on. Dumb strips + MOSFETs cost the loop nothing.

Lamp zones

Zone	Pin (unchanged)	Strip colour	Suggested size	Approx. current
On target A	PB12	Green	10-15 cm (6-9 LEDs, 5050)	~120-180 mA
Off target A	PB13	White	8-10 cm (5-6 LEDs)	~250-300 mA
Off target B	PB14	White	8-10 cm (5-6 LEDs)	~250-300 mA
On target B	PB15	Red	10-15 cm (6-9 LEDs, 5050)	~120-180 mA

Mode and fault stay on their little 5 mm LEDs - they are for the operator at the table, not the gym. Keep (or drop) the original four 5 mm lamps as table-side repeaters; both can share the pins.

Driver circuit, one per zone

From	To	Note
GPIO pin (PB12..PB15)	MOSFET gate via 220 R	Use four of the 220 R spares already in the BOM
MOSFET gate	GND via 10 k	Pulldown so lamps stay dark during boot
MOSFET source	GND	Common ground with the Black Pill
MOSFET drain	Strip segment negative (-)	Low-side switch
5 V supply	Strip segment positive (+)	Straight to 5 V, not through the MCU

> **Power budget:** all four zones lit together (lamp test) is ~0.8-1 A plus the Black Pill. Feed the box from a 5 V / 2 A USB supply or power bank into the Black Pill's 5 V pin and the strips in parallel. Do not power strips from the 3V3 regulator. Note some power banks auto-off below ~100 mA - the box idles well under that, so prefer a wall adapter or a bank with an always-on mode.

Bill of materials (adds to Doc 10)

Qty	Part	Notes
4	IRLZ44N logic-level MOSFET, TO-220	Mouser 942-IRLZ44NPBF, ~$1.70. Massive overkill at these currents - which means it never gets warm. Gate is happy at 3.3 V for strip loads.
4	10 k resistor (gate pulldown)	CFR-25JB-52-10K
4	220 R resistor (gate series)	Already spares in the Doc 10 BOM
1	5 V LED strip, white, 60 LED/m	Amazon: "5V USB LED strip white". One metre makes many boxes' white zones
1	5 V LED strip, red + green	Amazon: "5V LED strip red" / green, or one RGB strip wired per-colour (R and G channels used as the two zones, B unused)
1	5 V / 2 A USB supply or power bank	Powers Black Pill 5 V pin + strips
-	Diffuser	A strip of white acrylic or even paper over each zone turns points of light into a solid glowing bar - this is most of the "real box" look

An RGB-strip variant worth considering

A single analog RGB strip (four wires: +5 V, R, G, B) can serve both ON-target zones with only two MOSFETs: the R channel is fencer B's red lamp, the G channel is fencer A's green - mounted as a split bar with an opaque divider in the middle. White zones still want their own dedicated white strip; RGB "white" is dimmer and bluish.

Build and verify

- [ ] Bench-test one driver first: gate to 3.3 V by hand, strip segment lights hard; gate open, dark.
- [ ] Power the strips and the Black Pill from the same 5 V rail, grounds common.
- [ ] Power-up lamp test (already in the firmware) now sweeps the big zones - confirm all four fire in order and the supply holds 5 V while they do.
- [ ] Confirm the lamps are DARK during boot (that is the 10 k pulldowns doing their job).
- [ ] Run a bout on epee mode and check a double touch lights both big zones simultaneously with no supply sag - that is the worst realistic case.
- [ ] If a zone glows faintly when off, the gate pulldown is missing or the wrong pin is wired.

> Brightness control, if ever wanted: PB12-PB15 are timer-capable pins, so PWM dimming is a firmware option later - but the whole point of this upgrade is that no firmware change is needed today.

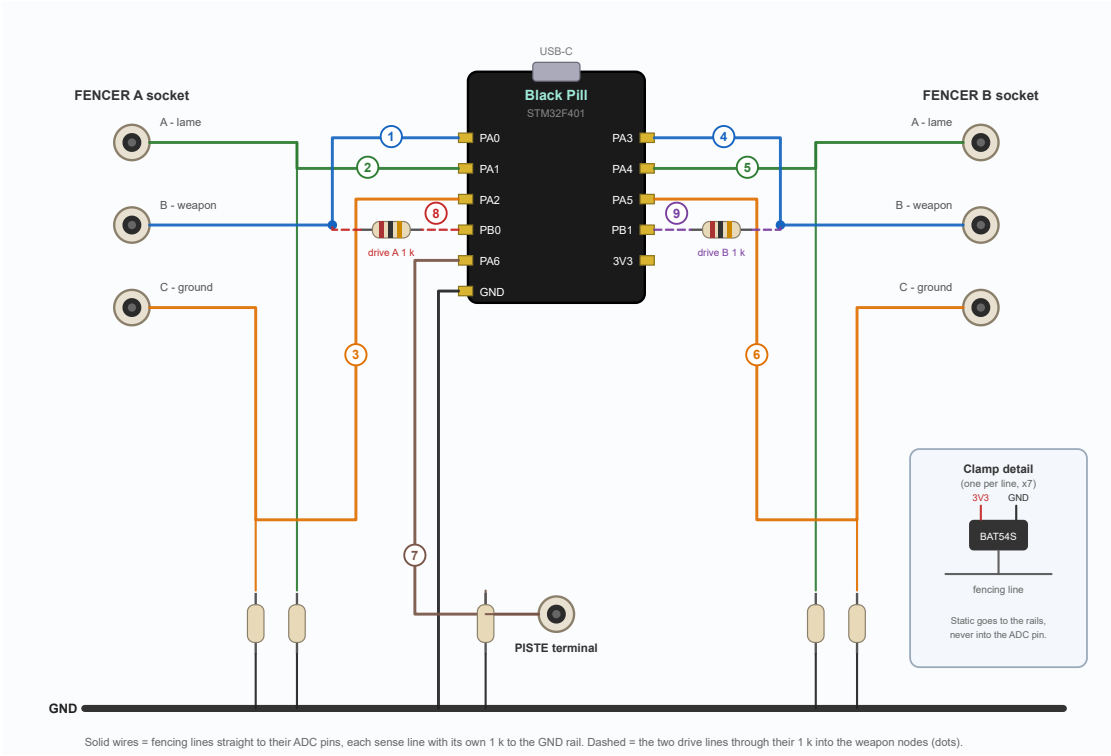
Three weapon fencing scoring box

STM32F401 front end

Pictorial wiring diagram

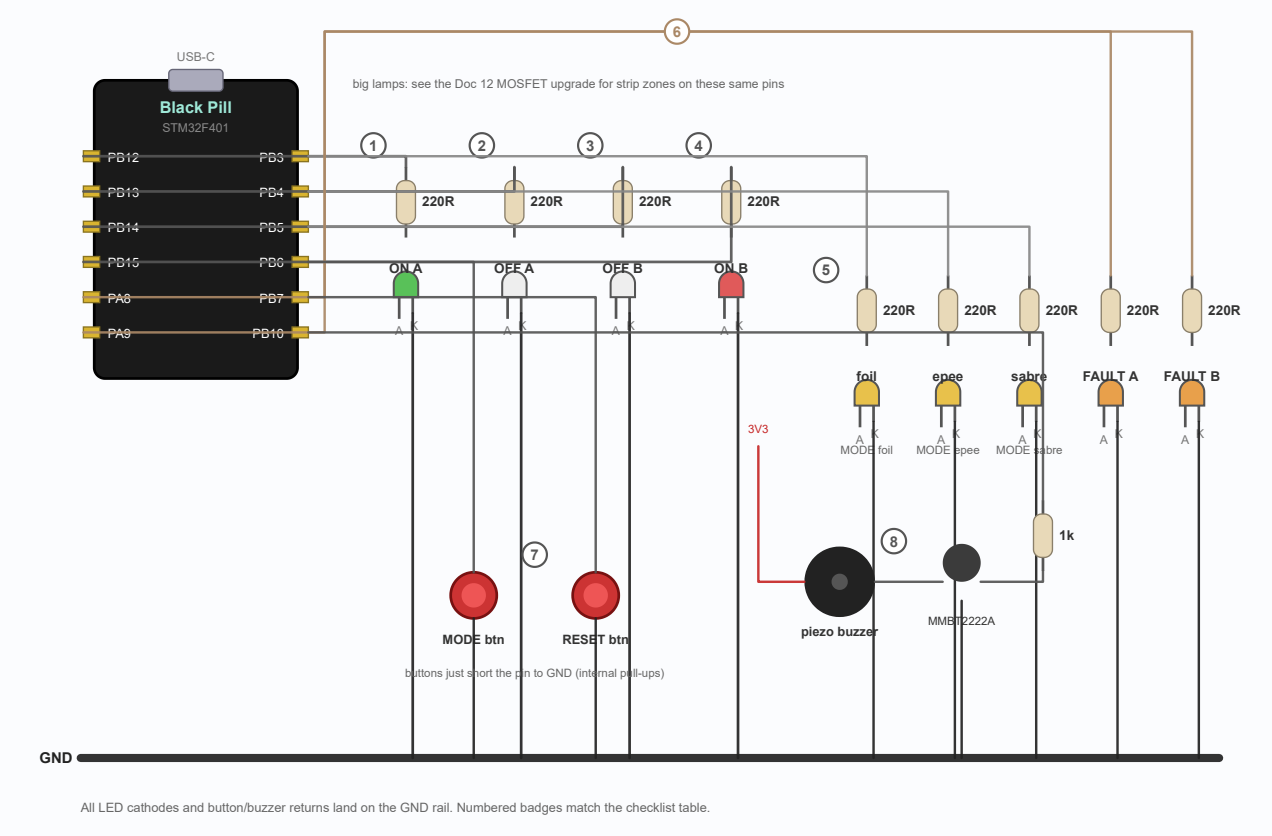
Doc 13 Rev A 2026-08-30

Sheet 1 of 2: the fencing front end. Solid coloured wires run each line straight to its ADC pin; every sense line also hangs a 1 k resistor to ground, and the two dashed drive lines feed the weapon nodes through their own 1 k from PB0 / PB1.



#	From	To	Wire
1	Fencer A socket, B (weapon) line	PA0 directly	blue
2	Fencer A socket, A (lame) line	PA1 directly, plus its own 1 k down to GND	green
3	Fencer A socket, C (ground) line	PA2 directly, plus 1 k to GND	orange
4	Fencer B socket, B line	PA3 directly	blue
5	Fencer B socket, A line	PA4 directly, plus 1 k to GND	green
6	Fencer B socket, C line	PA5 directly, plus 1 k to GND	orange
7	Piste terminal	PA6 directly, plus 1 k to GND	brown
8	PB0 (drive A)	through 1 k to the PA0 node	red, dashed
9	PB1 (drive B)	through 1 k to the PA3 node	purple, dashed
-	One BAT54S per fencing line (7 total)	middle pin on the line, ends to 3V3 and GND	see inset

Sheet 2 of 2: indicators and controls. Everything here is simple: pin, resistor, LED, ground - the only two subtleties are the buttons (no resistor, the chip's internal pull-ups do it) and the buzzer's little transistor.



#	Chain	What it is
1	PB12 → 220 R → green LED → GND	ON target A
2	PB13 → 220 R → white LED → GND	OFF target A
3	PB14 → 220 R → white LED → GND	OFF target B
4	PB15 → 220 R → red LED → GND	ON target B (Doc 12 puts MOSFET strip zones on these same four pins)
5	PB3 / PB4 / PB5 → 220 R → yellow LEDs → GND	Mode: foil / epee / sabre
6	PA8 / PA9 → 220 R → LEDs → GND	Fault A / Fault B
7	PB6 and PB7 each to one button, other side to GND	Mode + reset; pins use internal pull-ups, no resistor needed
8	PB10 → 1 k → MMBT2222A base; buzzer between 3V3 and collector; emitter to GND	Buzzer never loads the pin directly