SiccaLab

Test & Measurement, Simple

Turn your Arduino into a professional test & measurement instrument — no coding required.

User Guide

Version 1.0

www.siccalab.com

Contents

1. What is SiccaLab? 3
- 1.1 Key capabilities 3
- 1.2 Supported hardware 3

2. Installation & First Connection 4
- 2.1 System requirements 4
- 2.2 Flashing the firmware 4
- 2.3 Connecting a board 5

3. Main Window 6

4. Instrument Panels 7
- 4.1 Digital I/O 7
- 4.2 Analog Input 8
- 4.3 PWM Output 10
- 4.4 Square Wave 12
- 4.5 Data Acquisition (DAQ) 13
- 4.6 SPI Interface 14
- 4.7 I2C Interface 16
- 4.8 UART Interface 18

5. Multi-Board Support 19

6. Board View 20
- 6.1 Pin State Legend 20
- 6.2 Pin table 21

7. CSV & PDF Export 22
- 7.1 CSV 22
- 7.2 PDF 22

8. Configuration Files 23
- 8.1 Session state — automatic 23
- 8.2 Panel configuration — user-managed 23

9. Tips & Best Practices 24

10. Notes & Limitations 25

11. Support & Contact 26

1. What is SiccaLab?

SiccaLab transforms a supported Arduino board into a test and measurement instrument, controlled entirely through a modern Windows desktop interface. No firmware writing, no serial terminal, no custom sketches. Flash the firmware once, connect over USB, and every hardware function is immediately available through a purpose-built panel.

It is designed for engineers, labs, students, and hobbyists who need lab-grade instrument control without the cost of dedicated hardware adapters. A commercial I2C/SPI host adapter runs around €400. An Arduino Uno costs under €5.

1.1 Key capabilities

- Digital I/O — configure, read, write, and monitor pins in real time.
- Analog Input — multi-channel voltage acquisition with configurable VREF and engineering-unit formulas.
- PWM Output — hardware PWM with duty cycle control and live waveform preview.
- Square Wave — precision frequency generator with real-time waveform display.
- DAQ — binary acquisition up to 100 Hz, one or two channels, with live oscilloscope-style graph.
- SPI — Master and Slave modes, multi-device support, full transaction log.
- I2C — Master and Slave modes, bus scan, Write / Read / Write+Read with per-device logs.
- UART — hardware serial channels with ASCII/HEX TX and RX, command history, and CSV export.

1.2 Supported hardware

SiccaLab v1.0 supports four Arduino boards out of the box. The board-specific differences are handled transparently — the same panels work across all of them, with feature availability adjusting automatically.

Capability	Uno R3	Nano	Mega 2560	Micro
Digital I/O	14 pins	14 pins	54 pins	18 pins
Analog inputs	6 channels	8 channels	16 channels	6 channels
PWM outputs	6 pins	6 pins	15 pins	7 pins
Signal generation	✓	✓	✓	✓
SPI (Master & Slave)	✓	✓	✓	✓
I²C (Master & Slave)	✓	✓	✓	✓
UART channels	—	—	3 channels	1 channel
Multi-board support	✓	✓	✓	✓
Data logging up to 100 Hz	✓	✓	✓	✓

More boards currently in development — Arduino UNO R4, ESP32, and wireless support on the way.

2. Installation & First Connection

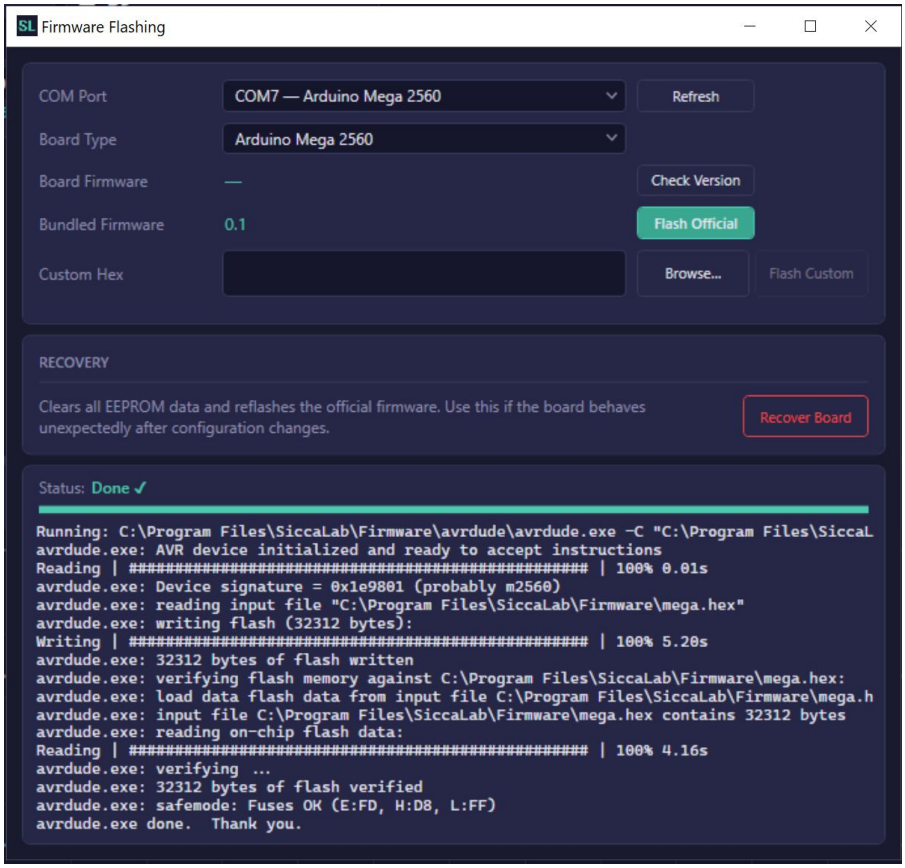
2.1 System requirements

- Windows 10 or Windows 11 (64-bit).
- One of the supported Arduino boards (see §1.2).
- USB cable appropriate for the board (Type-B for Uno/Mega, Mini-B for Nano, Micro-USB for Micro).
- No additional drivers required — Arduino boards are detected automatically.

2.2 Flashing the firmware

The SiccaLab firmware must be flashed to your Arduino. This is done directly from within the application via Tools → Firmware Flashing. No external tools or the Arduino IDE are required.

- Open Tools → Firmware Flashing.
- Select your COM port and board type from the dropdowns.
- Click Check Version to verify any existing firmware on the board.
- Click Flash Official to install the SiccaLab firmware.
- Wait for Status: Done — the board is ready to use.

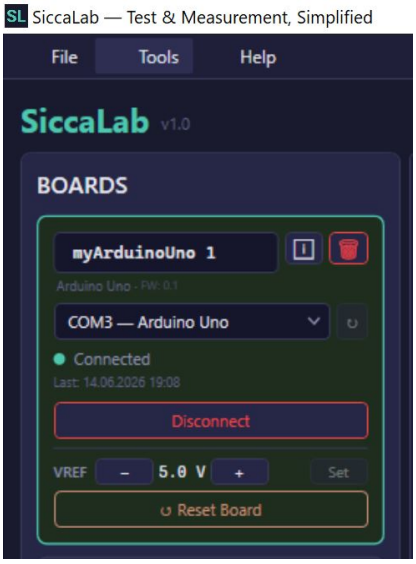


Firmware Flashing window — flashing complete (Status: Done ✓)

2.3 Connecting a board

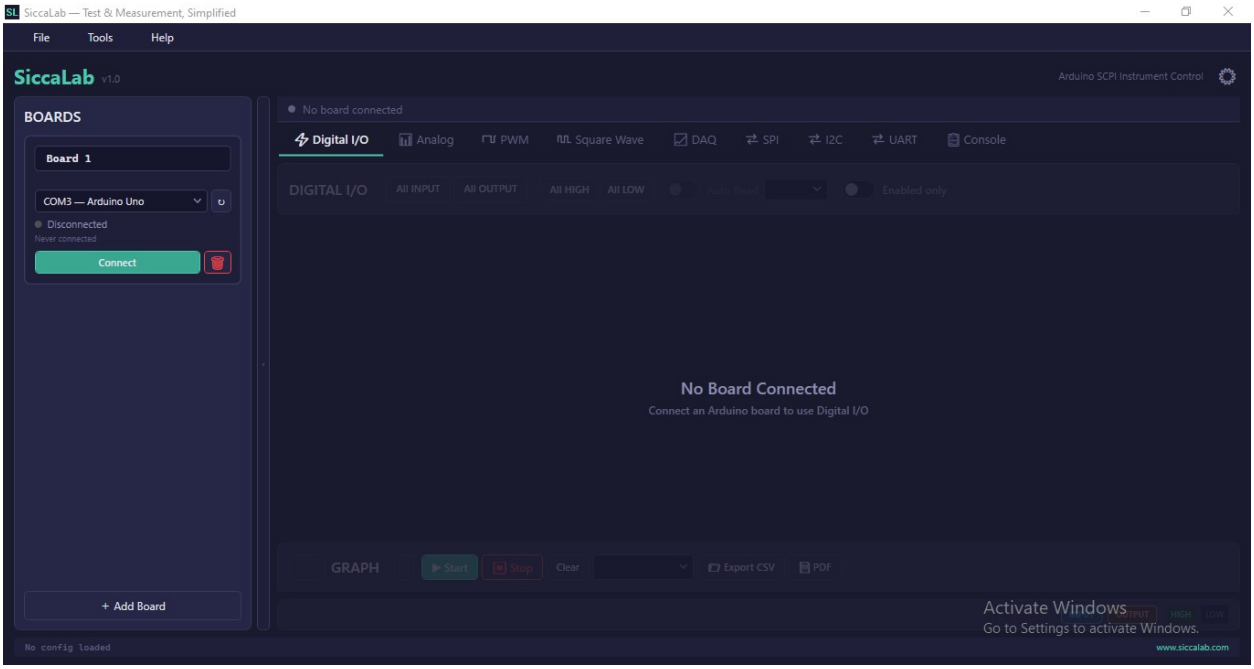
After flashing, return to the main window. The BOARDS sidebar on the left shows your board slot. Select your COM port from the dropdown, optionally give the board a label, and click Connect. SiccaLab identifies the board type and firmware version automatically. All panel tabs become active once the connection handshake completes.

> *If the board is not detected automatically, click the refresh icon next to the COM port dropdown to rescan available ports.*



Boards sidebar — board connected, ready to use

3. Main Window



SiccaLab main window — no board connected

The main window is divided into three areas: the BOARDS sidebar on the left, the active board identity strip below the title bar, and the instrument panel area on the right.

- **BOARDS sidebar —** add, connect, label, and reset boards. Supports up to three simultaneous connections.
- **Identity strip —** shows the active board's label, type, and COM port at all times, even when the sidebar is collapsed.
- **Tab bar —** Digital I/O, Analog, PWM, Square Wave, DAQ, SPI, I2C, UART, Console.
- **Status bar —** firmware version, board type, and current panel status shown at the bottom.
- **File menu —** New, Open, Save, Save As for panel configuration files.
- **Tools menu —** Firmware Flashing and Licensing.

4. Instrument Panels

4.1 Digital I/O

The Digital I/O panel provides full control of every digital pin on the connected board. Each pin can be individually enabled, configured as Input or Output, labeled, and read or written. Bulk operations (All INPUT / All OUTPUT / All HIGH / All LOW) are available in the toolbar. Auto Read polls all active pins at a configurable interval. The built-in graph records up to six channels simultaneously as a logic-style swimlane display.

Digital I/O panel — 3 active pins with live graph (Push Button, Red LED, Green LED)

- Per-pin enable, direction, label, read, and write.
- HIGH / LOW pill badge shows live pin state at a glance.
- Auto Read at 100 ms, 500 ms, 1 s, 2 s, or 5 s intervals.
- Graph records up to 6 channels, 50 000 points per channel.
- Export graph data to CSV or PDF.

Pin counts adjust automatically according to the connected board.

4.2 Analog Input

The Analog panel reads voltage from any ADC channel. VREF is configurable (default 5.0 V). Each channel can be labeled and optionally mapped through a custom formula for engineering unit conversion (for example, voltage to temperature, pressure, or current). The graph records all active channels over time with wall-clock timestamps.

Analog Input panel — channel A3 labeled Potentiometer, live voltage graph running

- Up to 6 channels on Uno and Micro, 8 on Nano, 16 on Mega 2560.
- Configurable VREF from 0.1 V to 5.0 V.
- Per-channel custom label and formula (variable: v).
- See 4.2.1 Formula Reference for the full formula syntax — supported operators, functions, constants, and worked examples.
- Full Scale or Autoscale Y-axis modes.
- Graph: 50 000 points buffer, wall-clock X axis (HH:mm:ss).
- Export to CSV (raw + converted columns) or PDF.

Setting VREF below 5.0 V requires an external reference voltage on the AREF pin. Without it, analog readings will be incorrect.

4.2.1 Formula Reference

Each analog channel can convert its measured voltage into an engineering unit using a custom formula. In the formula, the variable v stands for the channel reading in volts. The result replaces the displayed value, drives the graph when its mode is set to Expression, and is written to the converted column of CSV exports. The unit label beside the formula is text only and does not affect the calculation.

Formulas are validated as you type. While a formula is invalid, the channel keeps showing the raw voltage and a short message indicates the problem. An empty formula simply means no conversion is applied.

Operators

The operators below are listed from highest to lowest priority:

Operator	Meaning and priority
`( )`	Grouping (highest priority).
`^`	Power. Right-associative: `2^3^2` = 512.
`- (unary)`	Negation. Binds tighter than ^, so `-5^2` = 25.
`* /`	Multiplication and division.
`+ -`	Addition and subtraction (lowest priority).

Multiplication is never implicit: write `2*v`, not `2v`. Modulo, comparison, bitwise and logical operators are not supported.

Constants

Two constants are available: `pi` (also `PI`) for π, and `e` for Euler's number. Note that `e` is a constant, not a function — to raise e to a power, use `exp(x)`.

Functions

Function names are case-insensitive. Trigonometric functions use radians; to work in degrees, convert first, e.g. `sin(v * pi / 180)`.

Function	Description
`abs(x)`	Absolute value.
`sqrt(x)`	Square root. A negative input yields no result.
`exp(x)`	e raised to the power x.
`log(x)`	Natural logarithm (base e).
`log(x, base)`	Logarithm of x to the given base.
`log10(x)`	Base-10 logarithm.
`log2(x)`	Base-2 logarithm.
`sin(x), cos(x), tan(x)`	Trigonometric functions. Angle x in radians.
`asin(x), acos(x), atan(x)`	Inverse trigonometric functions. Result in radians.
`round(x)`	Round to nearest integer, half-to-even (`round(2.5)` = 2).
`floor(x)`	Round down to the nearest integer.
`ceil(x)`	Round up to the nearest integer.
`pow(x, y)`	x raised to the power y (same as `x^y`).

Numbers

Numbers use a dot as the decimal separator. Decimals (`3.14`), leading-dot values (`.5`) and scientific notation (`1e3`, `2.5E-4`) are all accepted. A comma is treated as a function-argument separator, not a decimal point.

Examples

Typical sensor conversions. The unit label is shown next to the value and in the CSV header.

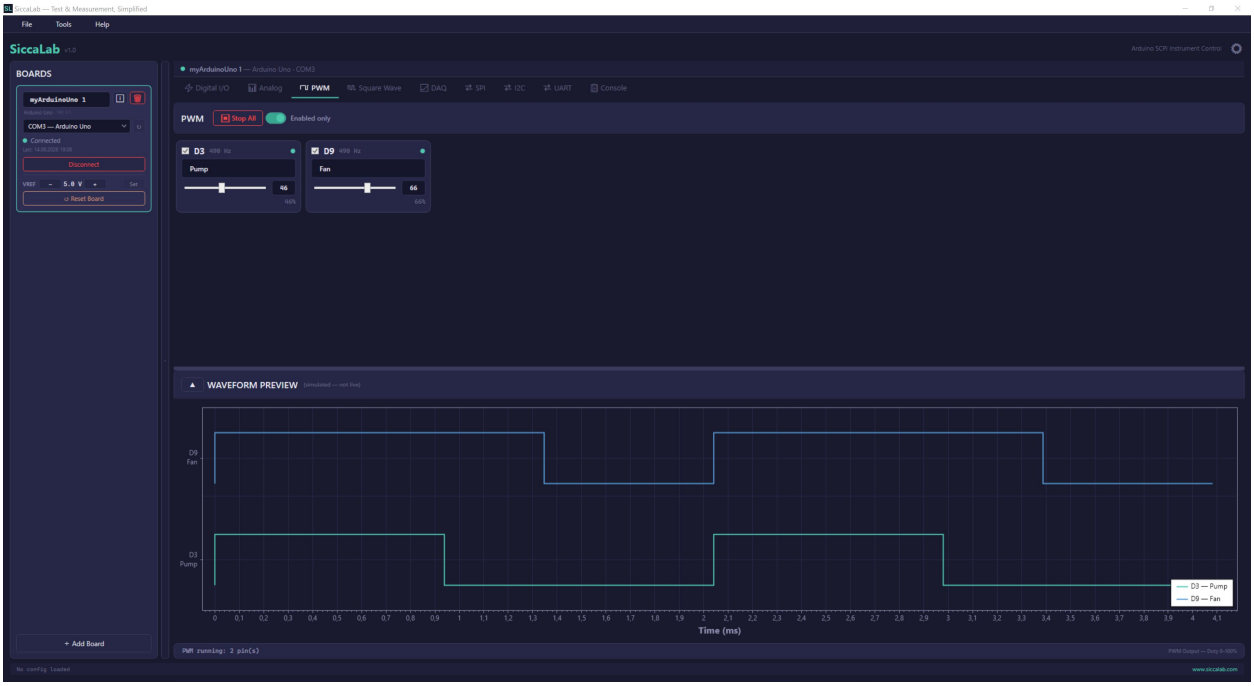
Sensor or use	Formula	Unit label
LM35 temperature	`v * 100`	°C
TMP36 temperature	`(v - 0.5) * 100`	°C
2:1 voltage divider	`v * 2`	V
11:1 divider (0–55 V)	`v * 11`	V
4–20 mA, 250 Ω sense	`v * 4`	mA
4–20 mA scaled 0–100%	`((v * 4 - 4) / 16) * 100`	%
Ratiometric pressure 0.5–4.5 V	`((v - 0.5) / 4) * 100`	psi
Percent of 5 V full scale	`(v / 5) * 100`	%
Decibel (ref 1 V)	`20 * log10(v)`	dB

Notes

- A formula that cannot be computed for a given reading — division by zero, the square root or logarithm of a negative number, and similar — produces no converted value for that sample. Check the formula and the sensor wiring if the converted column is unexpectedly blank.
- Unary minus binds tighter than the power operator, so `-5^2` evaluates to `25`, and the power operator is right-associative, so `2^3^2` evaluates to `512`. Use parentheses when in doubt.
- round() rounds half values to the nearest even integer (`round(2.5) = 2`, `round(3.5) = 4`). Use `floor(x + 0.5)` for conventional half-up rounding.

4.3 PWM Output

The PWM panel controls hardware PWM on all PWM-capable pins of the connected board. Duty cycle is set per pin with a slider (0–100%). The Waveform Preview shows a simulated representation of the active PWM signals. Pin conflicts with SPI or Square Wave are detected and reported automatically.

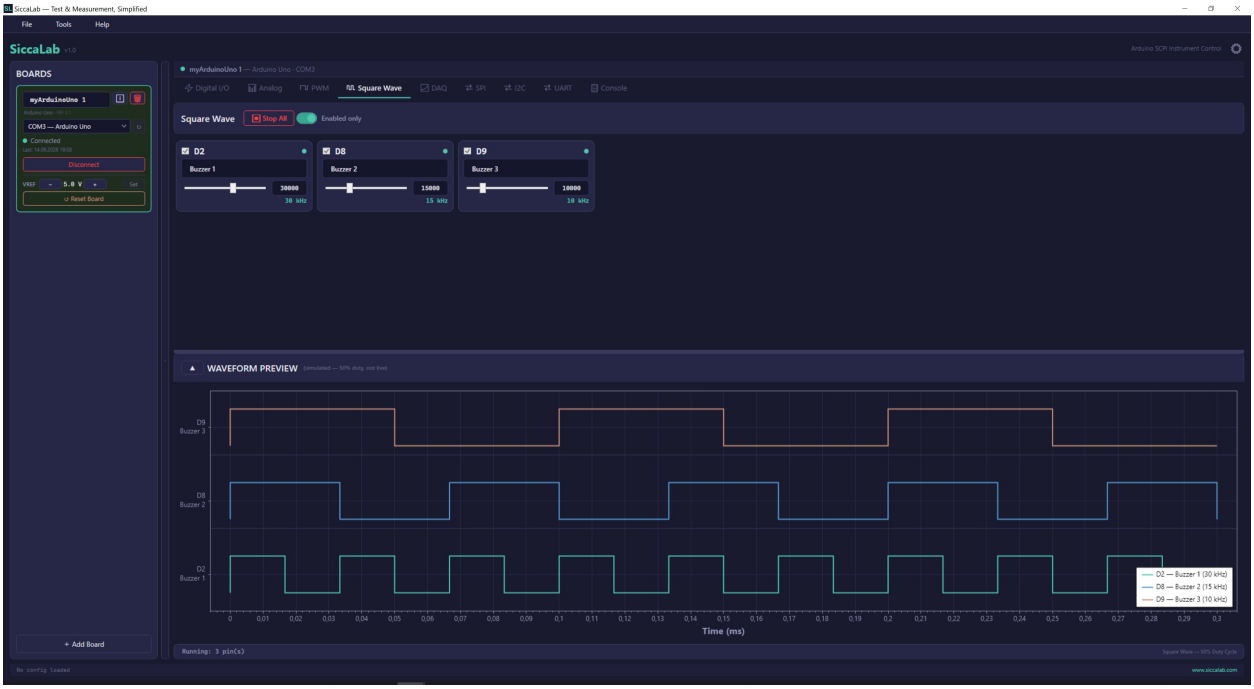


PWM panel — D3 (Scope, 75%) and D5 (Buzzer, 50%) running with waveform preview

- PWM-capable pins are automatically recognised.
- Per-pin duty cycle slider (0–100%) and numeric display.
- Simulated waveform preview.
- Pin conflict detection with SPI hardware pins and Square Wave timer.

4.4 Square Wave

The Square Wave panel generates precise frequency outputs using the Arduino tone() function. Each pin has an independent frequency slider and numeric input. The Waveform Preview updates live. Timer conflicts with PWM are detected and flagged inline per pin.

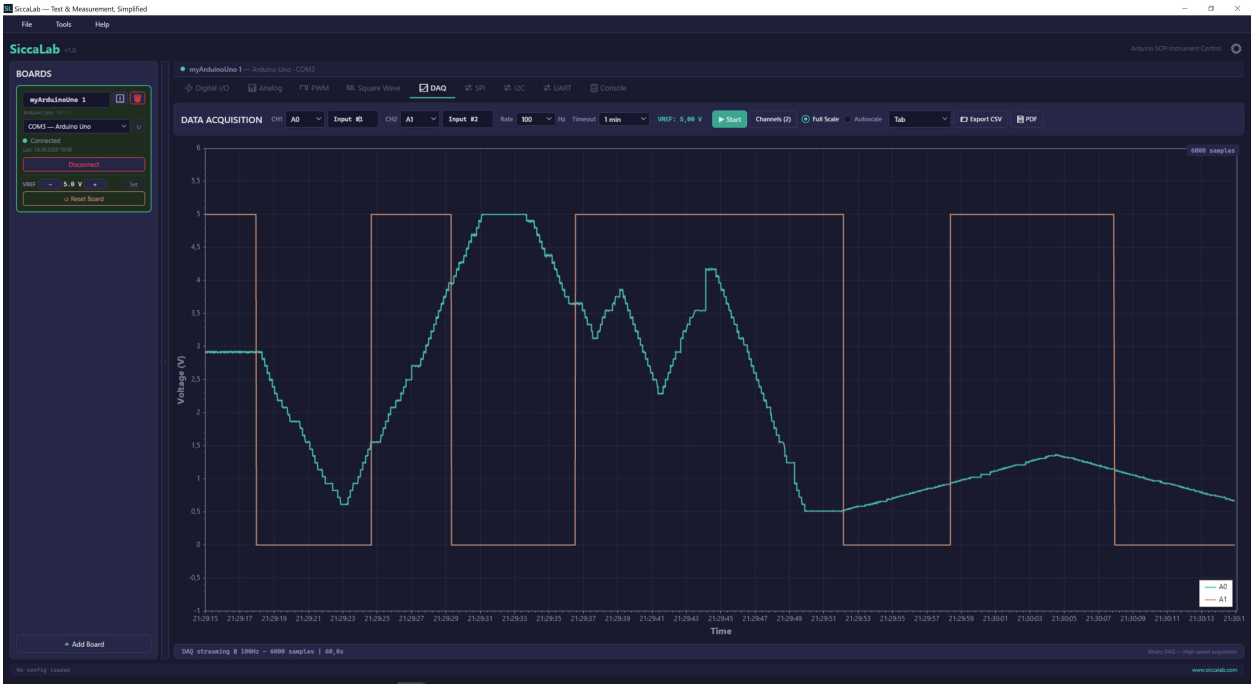


Square Wave panel — D4 running at 20 kHz with waveform preview

- Frequency range: 50 Hz – 50 000 Hz.
- Per-pin frequency slider and direct numeric input.
- Timer conflict warnings shown inline on affected pins.
- Live waveform preview at correct frequency scale.

4.5 Data Acquisition (DAQ)

The DAQ panel provides binary streaming acquisition. Up to two analog channels can be sampled simultaneously at rates from 1 Hz to 100 Hz. Data is transmitted as compact 6-byte binary packets and displayed in real time on an oscilloscope-style graph. A configurable timeout auto-stops the acquisition.



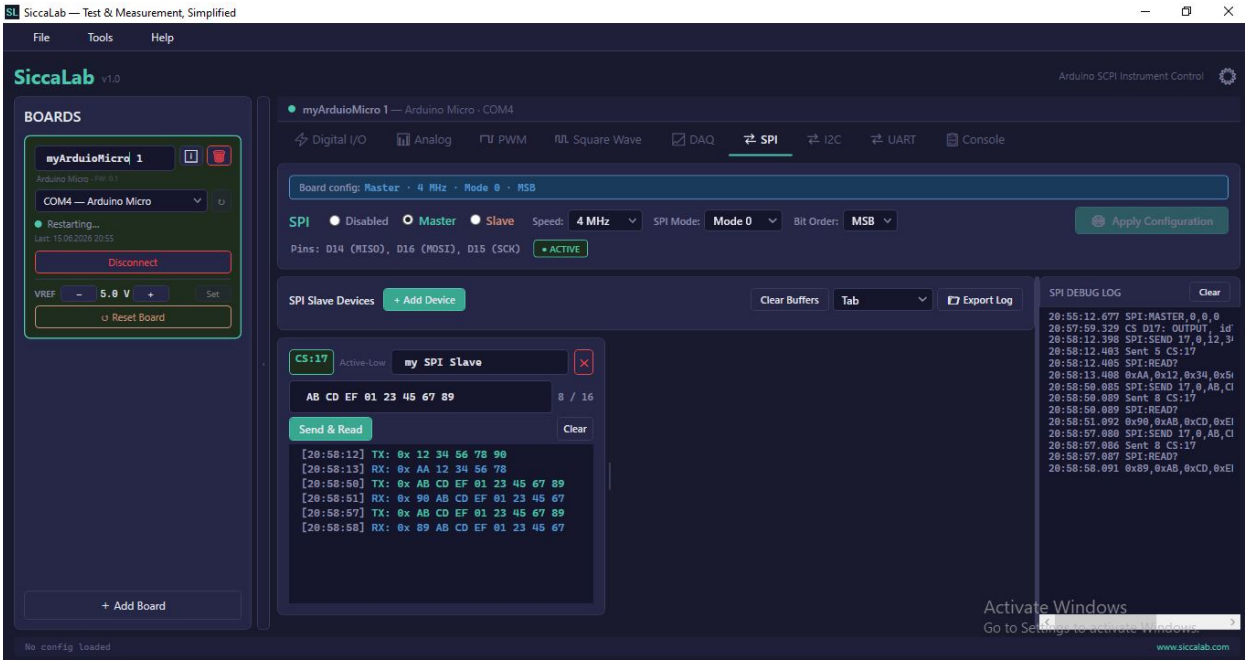
DAQ panel — dual channel capture (A0 and A3) at 50 Hz, auto-stopped on timeout

- One or two analog channels simultaneously, 1–100 Hz sample rate.
- Binary protocol — minimal serial overhead for high throughput.
- Live graph with wall-clock X axis and live sample counter.
- Auto-stop on configurable timeout (1–1800 s, default 1800 s).
- Export to CSV (timestamps at HH:mm:ss.fff precision) or PDF.
- Warning shown if Analog or DIO Auto Read is active before starting.

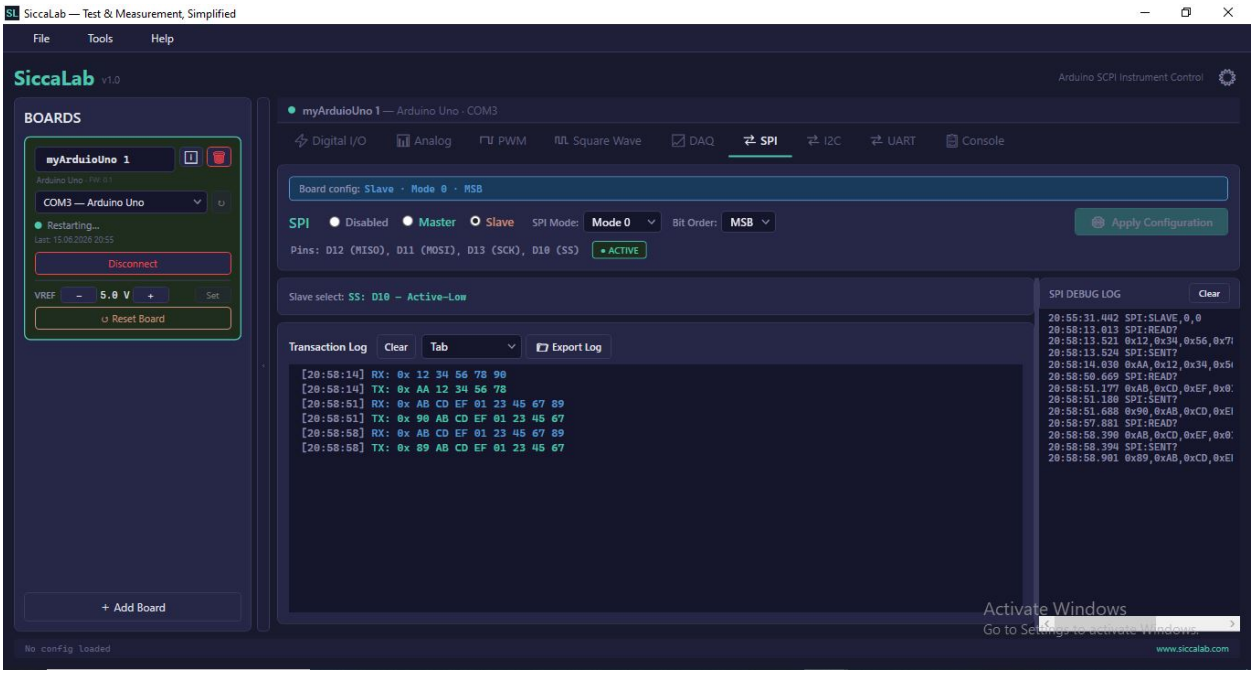
Starting DAQ automatically pauses DIO and Analog Auto Read jobs. They do not resume automatically when DAQ stops — re-enable them manually if needed.

4.6 SPI Interface

The SPI panel supports Master and Slave modes. Configuration (speed, mode, bit order) is stored in EEPROM and restored automatically on power-up. In Master mode, multiple slave devices can be added with individual CS pins, active levels, and labels. Each device has its own TX/RX log. In Slave mode, the Arduino echoes received bytes back on the next transaction.



SPI Master mode – Sending and receiving data with transaction log and SPI Debug Log

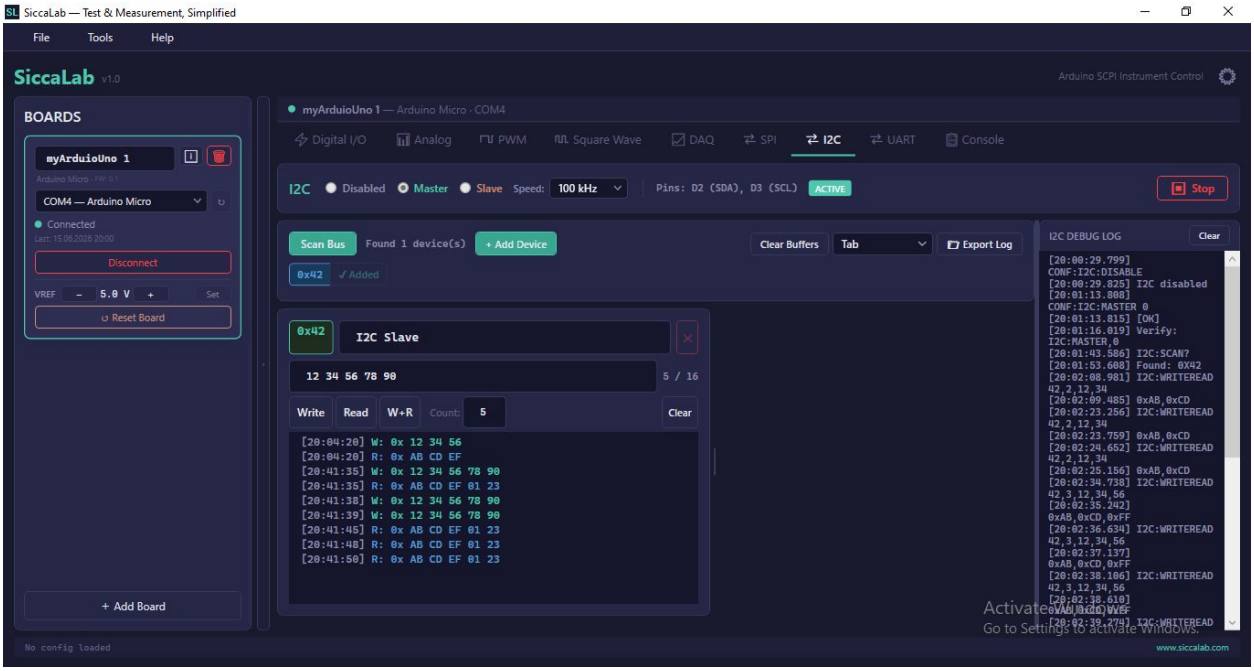


SPI Slave mode — receiving data with transaction log and SPI Debug Log

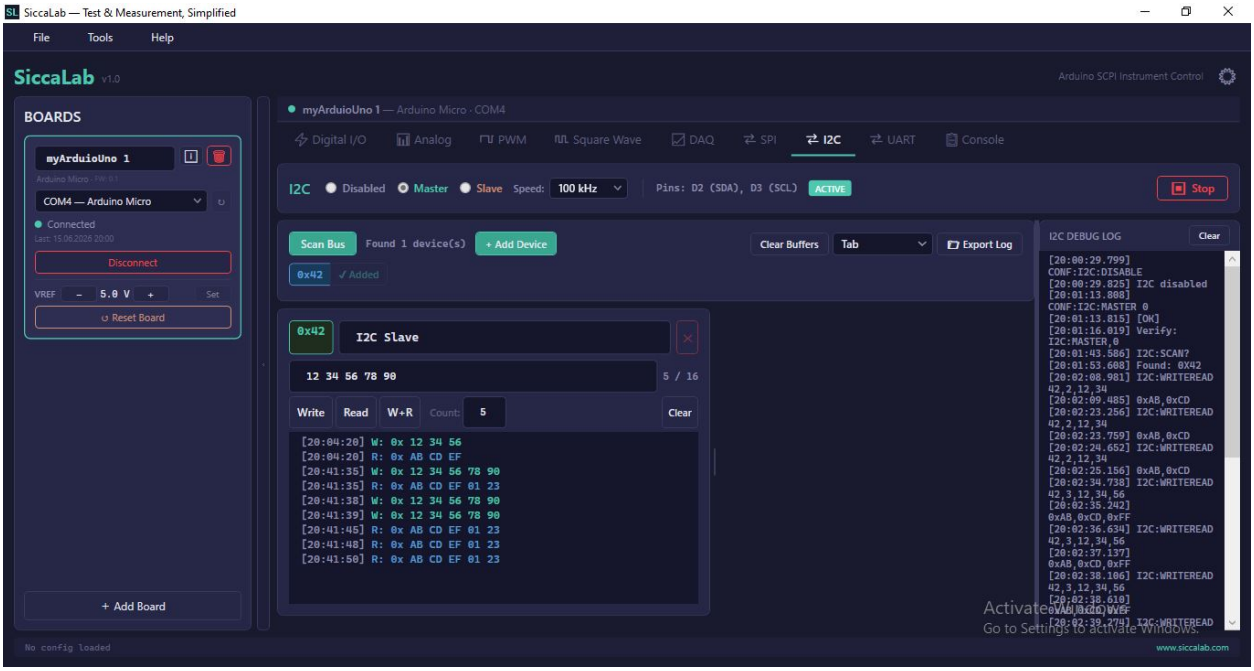
- Master mode: 6 clock speeds (125 kHz – 4 MHz), 4 SPI modes (0–3), MSB or LSB first.
- Per-device CS pin, active level (Active-Low or Active-High), and optional label.
- Send & Read per device — TX and RX logged with timestamps.
- Slave mode: loopback echo, auto-polled at 50 ms.
- SPI Debug Log panel shows full raw serial command trace.
- MISO, MOSI, SCK pins claimed automatically on connect — other panels cannot use them.
- SPI configuration is saved to EEPROM and restored on every power-up.
- A board restart is required after changing SPI configuration (the panel offers a one-click restart).

4.7 I2C Interface

The I2C panel supports Master and Slave modes. In Master mode, a single click scans the bus for all connected devices. Found devices can be added as individual cards, each supporting Write, Read, and combined Write+Read (register access) operations with per-device logs. In Slave mode, a TX buffer is preloaded and the panel auto-polls for data received from a master.



I2C Master mode

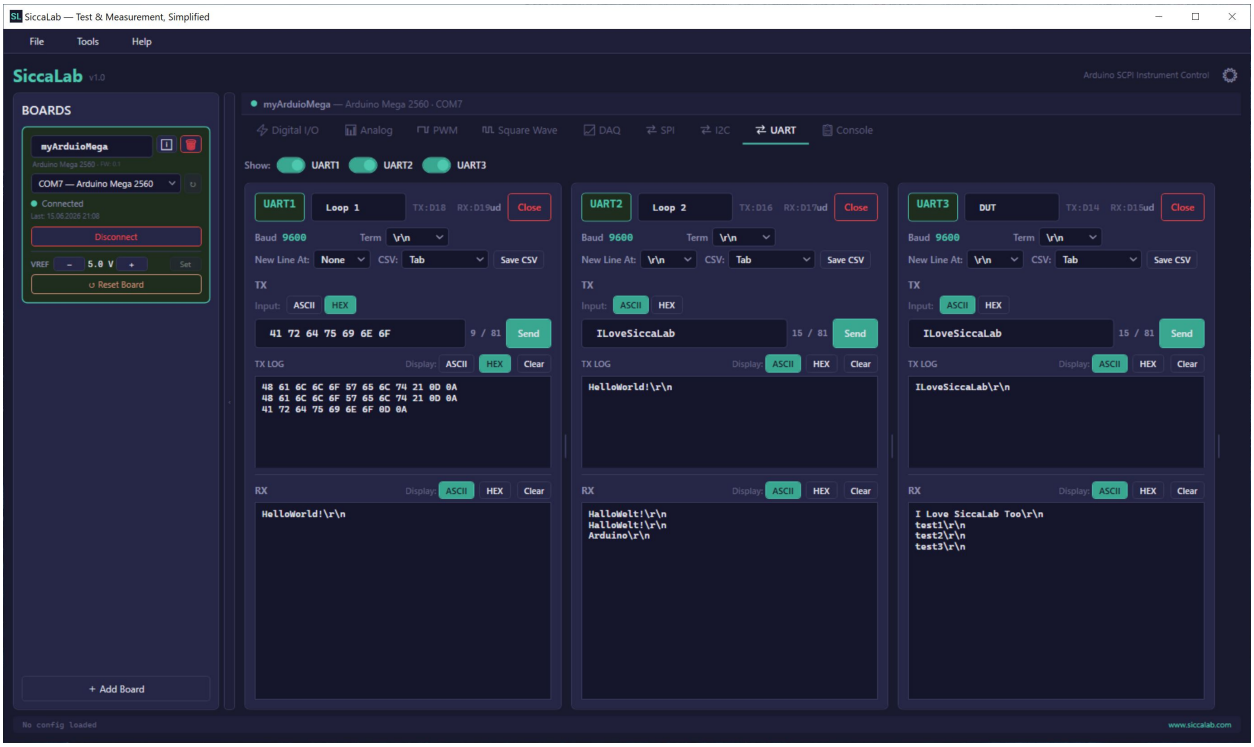


I2C Slave mode — address 0x60, TX buffer loaded, RX and TX transactions logged

- Master: 100 kHz (Standard) or 400 kHz (Fast) bus speed.
- Bus scan with one click — found devices shown and immediately ready to add.
- Per-device Write, Read, and W+R (combined write-then-read for register access).
- Slave: configurable 7-bit address (1–127), TX buffer preload, auto-poll at 50 ms.
- I2C Debug Log panel — full raw serial command trace.
- I2C configuration is NOT saved to EEPROM — always starts Disabled on power-up.
- SDA and SCL pins are claimed when I2C is started: A4/A5 on Uno and Nano, D2/D3 on Micro, D20/D21 on Mega 2560.

4.8 UART Interface

The UART panel provides hardware serial communication on boards that have spare hardware UART ports. Each channel is independently configurable with baud rate, line termination, and display mode. TX and RX are logged separately, with ASCII and HEX display modes for both directions.



UART panel (Mega 2560) — UART1, UART2 and UART3

Board availability:

Board	UART channels available
Arduino Uno	None (the only hardware UART is used for serial).
Arduino Nano	None (same hardware family as Uno).
Arduino Micro	UART1 (one channel).
Arduino Mega 2560	UART1, UART2, UART3 (three channels).

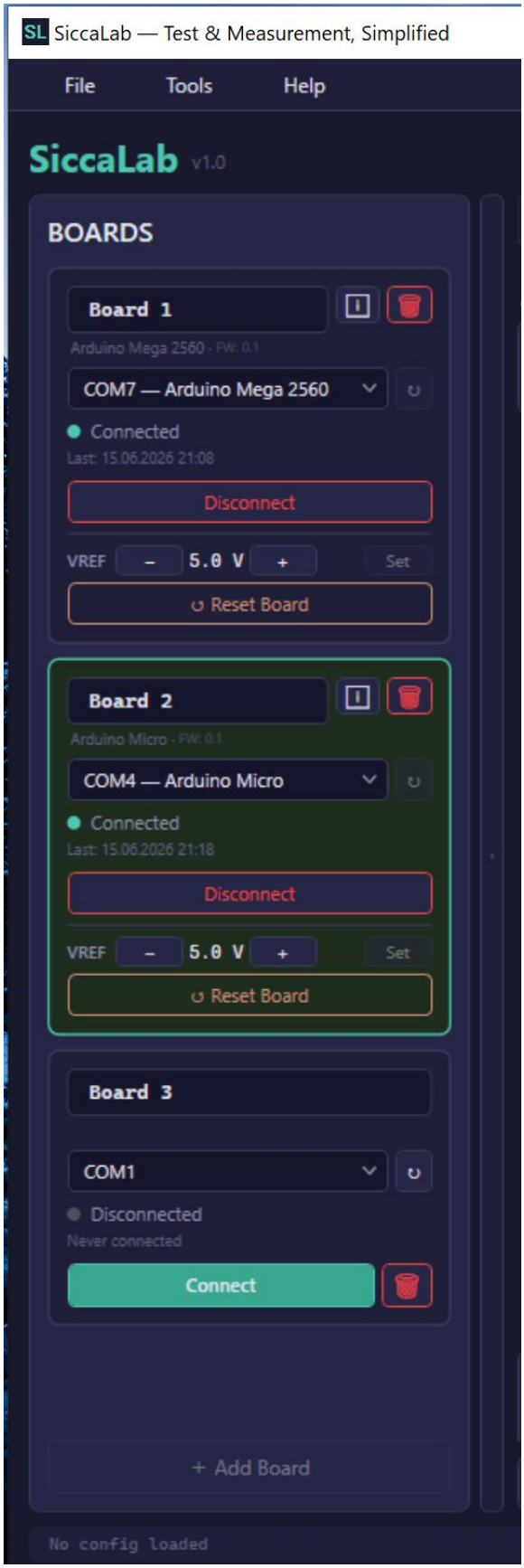
Features:

- Baud rates: 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- TX: ASCII or HEX input, configurable line termination (`None`, `\r`, `\n`, `\r\n`).
- RX: ASCII or HEX display, configurable newline delimiter for message splitting.
- Command history: up/down arrow keys, 50 entries, bash-style editing.
- Byte counter per channel — blocks send if payload exceeds the hardware buffer limit.
- Export TX/RX log to CSV with timestamps.

5. Multi-Board Support

SiccaLab supports up to three Arduino boards connected simultaneously. Each board has its own independent panel state, serial scheduler, and connection. Switching the active board is instant — no state is lost and no running jobs are interrupted. Click any board card to make it active.

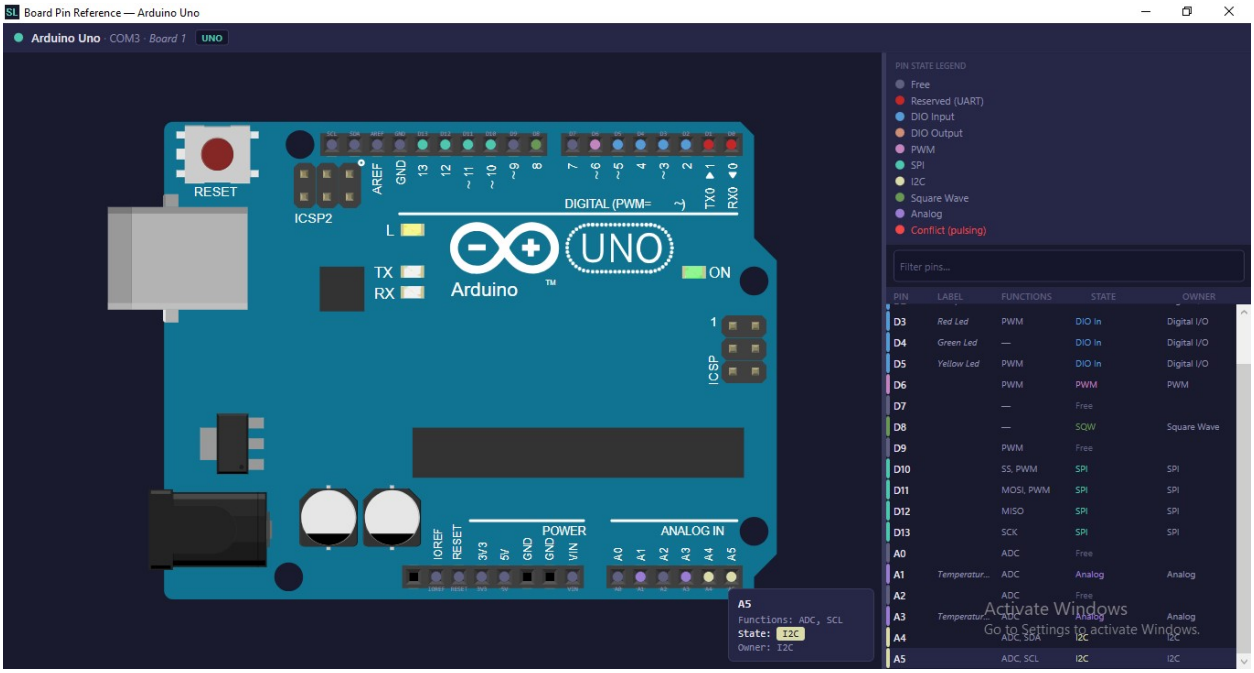
- Up to three simultaneous connections — any combination of Uno, Nano, Micro, and Mega 2560.
- Each board has fully independent state across all eight panels.
- Switching active board does not reset any panel or interrupt any running job.
- Each board can be labeled, independently reset, or disconnected without affecting the others.



6. Board View

Board View is a Board Pin Reference window that shows a live, colour-coded map of every pin on the connected board, alongside a table of each pin's functions, state, and owner. It updates in real time as the instrument panels claim and release pins.

Open it from the info (**i**) button on a connected board's card in the BOARDS sidebar. The window is non-modal, so you can keep it open beside the main window; there is one window per board, and it is available for all four supported boards. The title shows the board name — for example, Board Pin Reference — Arduino Uno.



Board View — Board Pin Reference for a connected Arduino Uno, pins colour-coded by current function

6.1 Pin State Legend

Each pin is colour-coded by what it is currently doing:

State	Meaning
Free	Unclaimed and available.
Reserved (UART)	Held by the firmware for the serial link (pins 0 and 1 on Uno and Mega).
DIO Input	Configured as a digital input in the Digital I/O panel.
DIO Output	Configured as a digital output in the Digital I/O panel (= HIGH / = LOW shown in the tooltip).
PWM	Driven by the PWM panel.
SPI	Claimed by the SPI panel (MISO, MOSI, SCK, SS).
I2C	Claimed by the I2C panel (SDA, SCL).
Square Wave	Generating a square wave (shown as SQW in the table).

State	Meaning
Analog	An analog channel enabled in the Analog panel.
Conflict (pulsing)	Two functions are competing for the same pin or timer.

6.2 Pin table

The table lists every pin with its identifier (PIN), your custom LABEL, hardware FUNCTIONS, current STATE (colour-coded to the legend), and OWNER (the panel holding it). The filter box narrows the table by pin, label, function, state, or owner. Hovering a pin on the graphic shows the same details as a tooltip and highlights its row. A pin pulses red when one panel requests a pin or timer that another already holds — release it in the owning panel to clear the conflict.

7. CSV & PDF Export

Most panels can export their data. The graphing panels — Digital I/O, Analog, and DAQ — export both a CSV and a PDF; the others export CSV only. Export is always triggered from a button on the panel itself, not from the File menu.

Panel	CSV	PDF
Digital I/O	Yes	Yes
Analog	Yes (raw + converted)	Yes
DAQ	Yes	Yes
UART	Yes	—
SPI	Yes (Master & Slave logs)	—
I2C	Yes (Master & Slave logs)	—
Console	Yes	—

Graph exports (Digital I/O, Analog, DAQ) become available once the capture is stopped and has data.

7.1 CSV

CSV exports always contain the full captured data, using the separator you choose on the panel (Tab, Semicolon by default, or Comma; the Console always uses a semicolon). By panel:

- **Digital I/O, Analog, DAQ —** timestamped samples, one column per pin or channel. Analog adds a converted column when a channel has a formula.
- **UART —** Timestamp, Direction, and Content, one file per channel (ASCII or HEX per the display mode).
- **SPI, I2C —** the Master and Slave logs, with device, CS pin or address, direction, and data.
- **Console —** Timestamp, Direction, and Content.

7.2 PDF

Digital I/O, Analog, and DAQ also export an A4-landscape PDF report: a title, the capture metadata (board, timing, sample counts, and VREF where relevant), and the graph image — not a data table. The image is a render of the current view, so whatever you have zoomed or scrolled to is what appears; the CSV always holds the full data.

8. Configuration Files

SiccaLab separates two types of persistence: automatic session state and user-managed panel configuration.

8.1 Session state — automatic

Board connections (COM port, board type, label) and the theme preference are saved automatically on every change and restored silently on the next startup. No user action required.

8.2 Panel configuration — user-managed

All panel settings — DIO pin directions and labels, Analog channels, VREF, and formulas, PWM duty cycles, SPI slave devices, I2C mode and devices, UART channel settings — are saved and loaded explicitly via the File menu. They are never auto-saved.

- `File → New` (Ctrl+N) — clear all panel configuration.
- `File → Open` — load a previously saved configuration file (.json).
- `File → Save` (Ctrl+S) — save current configuration to the open file.
- `File → Save As` (Ctrl+Shift+S) — save to a new file.

Configuration files are plain JSON. They can be shared between setups, version-controlled, or kept alongside lab notebooks.

9. Tips & Best Practices

A few practical notes that will save you time.

Flash firmware before connecting

SiccaLab will connect to any Arduino board, but panels will not work correctly without the SiccaLab firmware. Flash it first via Tools → Firmware Flashing.

Use labels on everything

Pin labels, channel labels, and device labels persist in your configuration file. Set them up once and your setup becomes self-documenting.

Save your configuration file early

Panel configuration is not auto-saved. Save after setting up your pins and devices so you can restore your setup quickly after a board reset or restart.

Start simple before going deep

Test Digital I/O and Analog first to confirm the board is working correctly before moving to SPI, I2C, or DAQ.

Use Auto Read for monitoring

Digital I/O and Analog both support Auto Read with configurable intervals. Enable it for passive monitoring without manual polling.

DAQ pauses other auto-reads

When DAQ starts, DIO and Analog Auto Read are paused automatically. Re-enable them manually after DAQ stops if needed.

Board Reset is non-destructive to your config

Reset Board clears the Arduino hardware state but does not touch your loaded configuration file. You can re-apply the configuration immediately after reset.

SPI requires a restart after configuration changes

SPI configuration is stored in EEPROM. After changing speed, mode, or bit order, the board must be restarted for the new settings to take effect. The SPI panel offers a one-click restart for this purpose.

I2C is reconfigured after every connection

I2C settings are not persisted on the board. After every connect or restart, I2C starts Disabled — open the I2C panel to reconfigure it.

10. Notes & Limitations

Behaviours that are intentional but worth knowing.

Topic	Detail
UART availability	UART panels are active only on boards with spare hardware UARTs: one channel on Arduino Micro, three on Arduino Mega 2560. On Uno and Nano the tab shows a Not Available banner because the single hardware UART is used for serial commands.
DAQ pauses Auto Read	Digital I/O and Analog Auto Read pause automatically when DAQ starts. They do not resume automatically when DAQ stops — re-enable manually if needed.
SPI restart after config	Changing SPI configuration (speed, mode, bit order) requires a board restart for the new settings to take effect. The SPI panel handles this with one click.
I2C not persisted	I2C configuration always starts Disabled on power-up. Reconfigure it after every connect or board reset.
DAQ maximum rate	The DAQ binary protocol is capped at 100 Hz due to Arduino serial throughput. Higher rates are not currently supported.
No auto-reconnect	If a board is disconnected unexpectedly (for example, the USB cable is pulled), it must be reconnected manually. This is consistent with the behaviour of professional benchtop instruments.
Panel configuration is manual	Panel configuration must be saved explicitly via File → Save. Session state (COM ports, labels, theme) is saved automatically and does not need user action.

11. Support & Contact

Reach out for product questions, bug reports, feature requests, or licensing matters.

Website
`www.siccalab.com`

Email
`contact@siccalab.com`

When reporting an issue, please include:

- Your board type and firmware version (visible in the BOARDS sidebar).
- SiccaLab application version (visible in Help → About).
- The panel you were using and the exact action that triggered the issue.
- A screenshot if anything visual is involved.

Thank you for using SiccaLab.