

Nefertari

Electronics trainer

Nerfi 1 product and user guide

Circuit building, measurement and programming on one learning platform.

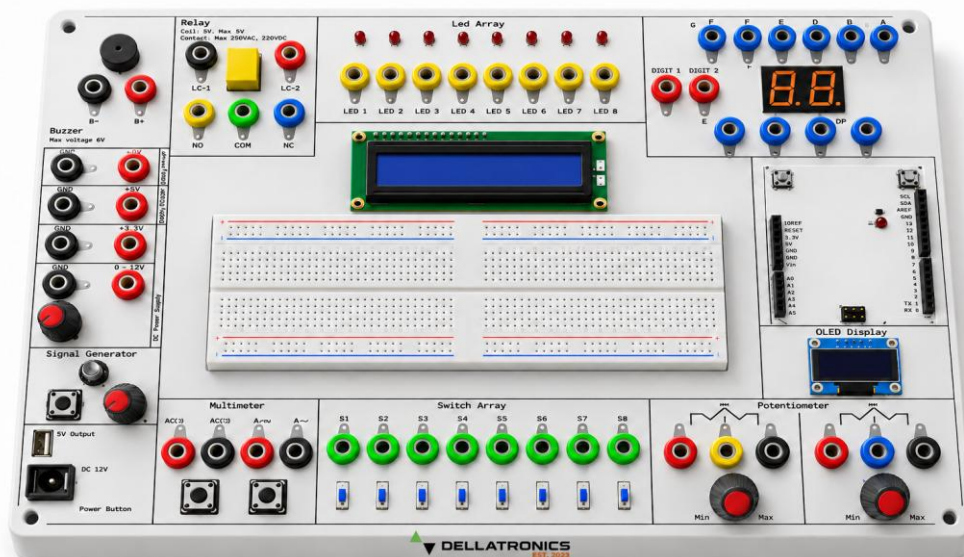


Figure 1 Nefertari trainer with a simple white base

Dr. Della Tamin

DIGITAL LOGIC • PRACTICAL ELECTRONICS • ARDUINO

Cover illustration is a product rendering. Use the actual board labels and verified hardware documentation for electrical connections.

1 Introduction

The Nerfi 1 Digital Acquisition Trainer (DAT) is a multifunctional educational platform for electronics teaching in classrooms and through distance learning. It supports learning from introductory levels through university, with a modular board that combines practical circuit building, measurement and programming.

This guide introduces the trainer, explains its components and handling, and provides practical exercises and Arduino code examples. Read the safety and setup sections before connecting power or assembling an experiment.

Learning support

The learning resources described for the trainer include electronic projects with circuit diagrams, explanatory tutorial videos, guidance adapted to primary, secondary and tertiary learning stages, and learning applications.

Skills developed

- Logic gates, truth tables and Boolean operations
- Combinational circuits including adders, multiplexers and decoders
- Sequential circuits including flip-flops, counters and shift registers
- Timing, clock-based circuits, mechatronics and systems automation

Guide contents

- 2 System description and integrated subunits
- 3 Safety handling of the board
- 4 Getting started with Nerfi 1
- 5 Components and functions
- 6 Troubleshooting guide
- 7 Technical specifications
- 8 Key applications
- 9 Learning summary
- 10 Practical experiments
- 11 Coding examples

2 System description

The trainer uses a modular printed circuit board. Subunits connect directly or through designated connection points, while the integrated breadboard provides space for practical circuits. An external adapter supplies the board; USB supports computer connection. Confirm which modules can operate from USB on the supplied board revision.

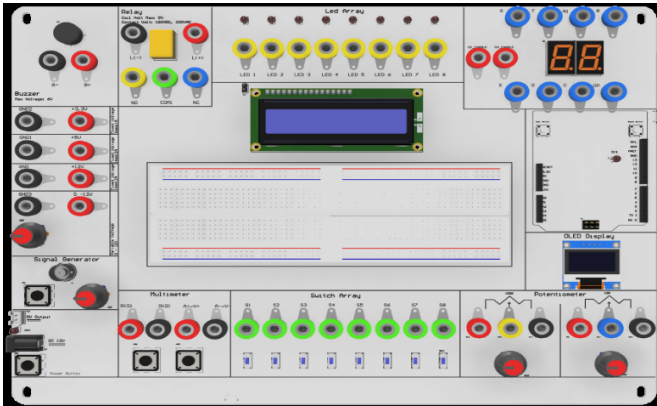


Figure 2 Original Nerfi 1 board layout

Table 1 Integrated subunits and typical uses

Subunit	Function and typical use
Arduino Leonardo	ATmega32u4 microcontroller platform for USB programming, digital interfacing, automation and sensor integration.
Two seven segment displays	Numeric and simple character output for counters, clocks and results.
Relay module	Load switching and automation demonstrations. Verify fitted relay type and ratings before use.
Buzzer	Audible alarms, status signals and feedback.
LCD and OLED displays	Text, menus and measurement displays; OLED supports compact graphical output.
Power supply and control	Regulated power distribution, module switching and power management.
Signal generator	Test signals for timing, analogue and digital circuits, and sequential logic.
Multimeter	Measurement of electrical quantities for diagnostics and experiment verification.
Two potentiometers	Adjustable inputs for voltage dividers, control and analogue conversion.

3 Safety handling of the board

Use the approved power supply

Use the supplied adapter and follow its label, polarity and electrical ratings. The guide describes a 220 V AC to 18 V DC adapter; confirm suitability for the local mains supply before connection.

Handle integrated circuits carefully

Use anti-static precautions when inserting or removing ICs. Switch off and disconnect power before changing components or wiring.

Prevent incompatible connections

Do not short outputs or apply voltages beyond the rating of a module. Check connections before power is restored.

Keep the trainer dry

Protect the board from liquids, spills and damp work surfaces.

Use instructor-approved low-voltage loads for student experiments. Relay contact markings do not establish that exposed trainer terminals are suitable for mains-voltage work.

Commented [ER1]: The source describes a solid-state relay, while its illustration shows a relay with NO/COM/NC contacts. Confirm the fitted device, isolation and permissible loads before publication.

4 Getting started with Nerfi 1

1 Unpack and inspect

Check the board, accessories and supplied adapter for damage. Confirm that the trainer is switched off.

2 Confirm the power arrangement

Verify the input rating on the actual board. The source also describes two 9 V batteries in series and a USB connection; use these only when approved for this board revision.

3 Power the trainer

Connect the approved supply, switch on and verify the Power ON LED.

4 Run an initial check

Under instructor guidance, connect the clock output to a compatible LED indicator with the required interface. Use a suitably low frequency to observe the indication.

Power input requires confirmation: the guide states 18 V, while the original board image labels a DC input as 12 V. Resolve this before powering the trainer.

5 Components and functions

Table 2 Main functional elements

Component	Function
Power supply	Regulated outputs described as ± 12 V DC, +3 V DC and +5 V DC. The specifications section instead lists 3.3 V; verify the fitted rail.
Logic inputs	DIP switches select logic 0 or 1.
Indicators	Red and green LEDs indicate digital output states.
Function generator	Adjustable signal frequency with a manual step control for experiments.
Breadboard	Prototyping area for temporary circuit connections.
Seven segment displays	Numeric indication for digital output experiments.

6 Troubleshooting guide

Table 3 Initial checks

Symptom	Check
No power	Check the approved adapter connection and power switch. Inspect for loose connections before reconnecting power.
LED does not light	Check polarity and wiring. Confirm that the signal is compatible with the indicator.
Clock activity is not visible	Check the frequency setting. An LED cannot visually resolve a high-frequency clock; use a suitable measurement method.

Before changing connections

Turn off and disconnect the trainer. Inspect the circuit, confirm the intended supply rail and check the experiment wiring. Reconnect power only after the cause of the problem has been addressed.

7 Technical specifications

Table 4 Specifications recorded in the guide

Feature	Recorded specification
Supply outputs	± 12 V, 3.3 V and 5 V regulated; 1 A maximum stated. Per-rail versus total current limit requires confirmation.
Logic inputs	8 DIP switches.
Indicators	8 logic LEDs and 1 power LED.
Function generator	1 MHz to 25 MHz with 0.1 Hz resolution stated; confirm range and any separate low-frequency clock function.
Numeric displays	Two seven segment LED displays.
Breadboard	430 tie-points stated; confirm against the fitted breadboard.
OLED display	Graphical and numeric presentation of signals and measurements.
Arduino Leonardo	Native USB; 20 digital I/O, 7 PWM outputs and 12 analogue inputs as recorded in the guide. Supports mouse or keyboard emulation.

8 Key applications

Digital logic learning

Build truth tables and test gates, adders, multiplexers, decoders, flip-flops, counters and shift registers.

Timing and automation

Explore clock pulses, timing sequences and synchronization. Combine sensor inputs, PWM and relay control in supervised low-voltage projects.

Prototyping and research

Use the breadboard, microcontroller and displays for classroom demonstrations, laboratory work and student research.

Distance learning

Carry out guided hardware experiments with online tutorials and learning support appropriate to the student's level.

9 Learning summary

The Nerfi 1 DAT connects digital electronics theory with practical circuit construction and programming. Learners can observe logic states, measure signals and build increasingly integrated projects on one platform.

10 Practical experiments

10.1 Digital logic experiments without Arduino

- Use switches to create binary inputs and LEDs to display logic results.
- Construct AND, OR and XOR functions and record their truth tables.
- Build a four-bit adder or a binary-to-decimal display circuit.

10.2 Arduino based experiments

- Read switch inputs and display a count on the two seven segment displays.
- Control a compatible relay module from a button or switch.
- Create a timer or counter with numeric output.
- Use a compatible function-generator signal as an input to a counter program.

10.3 Integration experiments

- Combine Arduino and a relay to control an instructor-approved low-voltage load.
- Display temperature or light sensor values on numeric displays.
- Investigate signal counting and frequency measurement with a compatible input interface.

11 Coding examples

The following listings are editable reproductions of the source examples. Check their pin mapping, display polarity, interface circuitry and input levels against the actual board before use. These listings have not been hardware-validated.

Use current-limiting resistors and suitable digit drivers where required. Do not assume an Arduino pin can directly power a complete display digit or relay coil. The final project counts incoming pulses; its serial output is a cumulative count rather than a calibrated frequency measurement.

11.1 Two digit counter

This example counts from 00 to 99. The source calls it a binary counter, although the displayed sequence is decimal.

Review before use: the source's common-cathode comment, segment encoding, bit order and digit-enable polarity need to be reconciled with the actual display driver.

```
// Example 1: Two 7-Segment Counter
// Common Cathode 7-seg assumed (invert logic if Common Anode)
```

```
int segPins[7] = {2,3,4,5,6,7,8}; // Segments A-G
int digitPins[2] = {9,10}; // Control two digits (multiplexing)
```

```
// 7-seg digit encoding for 0-9
```

```
byte numbers[10] = {
  B0000001, // 0
  B1001111, // 1
  B0010010, // 2
  B0000110, // 3
  B1001100, // 4
  B0100100, // 5
  B0100000, // 6
  B0001111, // 7
  B0000000, // 8
  B0000100 // 9
};
```

```
void displayDigit(int digit, int value) {
  digitalWrite(digitPins[digit], LOW); // Enable digit
  for(int i=0; i<7; i++) {
    digitalWrite(segPins[i], bitRead(numbers[value], i));
  }
  delay(5);
  digitalWrite(digitPins[digit], HIGH); // Disable digit
}
```

```
void setup() {
  for(int i=0; i<7; i++) pinMode(segPins[i], OUTPUT);
  for(int i=0; i<2; i++) pinMode(digitPins[i], OUTPUT);
}
```

```
void loop() {
  for(int num=0; num<100; num++) {
    unsigned long start = millis();
    while(millis() - start < 500) { // Show each number for 0.5 sec
      int tens = num / 10;
      int ones = num % 10;
      displayDigit(0, tens);
      displayDigit(1, ones);
    }
  }
}
```

11 2 Relay control with a switch

This example controls a relay from a switch connected to pin 2. With the assumed active-high relay interface, a LOW switch input turns the relay on; releasing the switch turns it off.

```
// Example 2: Relay Control
const int switchPin = 2; // Input switch
const int relayPin = 8; // Relay module

void setup() {
  pinMode(switchPin, INPUT_PULLUP);
  pinMode(relayPin, OUTPUT);
}

void loop() {
  int state = digitalRead(switchPin);
  if(state == LOW) {
    digitalWrite(relayPin, HIGH); // Relay ON
  } else {
    digitalWrite(relayPin, LOW); // Relay OFF
  }
}
```

Connection checks

The sketch assumes a switch between pin 2 and ground, using the internal pull-up, and an active-high relay-module input on pin 8. Confirm that the board's relay input has an appropriate driver and that the relay is off during startup.

11 3 Function generator counter with LED control

The final example combines pulse counting, a multiplexed two-digit display and four switch-controlled LEDs. The listing continues on the following pages.

Pin conflict to resolve: pins 5 and 6 appear in both segPins and switchPins. The program reconfigures these shared pins as inputs. Allocate distinct, suitable pins and verify the board connections before running this example.

Read the shared pulseCount variable atomically on the Leonardo when taking a multi-byte snapshot. Verify pulse amplitude and allowable counting rate; the advertised generator range must not be assumed compatible with this interrupt-based sketch.

Commented [ER2]: Source code retained rather than silently reassigned. Verify all actual board wiring; the comment header also disagrees with the arrays used below.

Function generator counter setup

Listing 3 Part 1 of 2 Source pin definitions and initialization

```

/*
Nefertari Trainer - Function Generator Counter with LED Control
Arduino Leonardo
- Function generator input on pin 2
- 7-segment display: 2 digits
- LED outputs on pins 8-11
- Switch inputs on pins 4-7
*/

// ----- Pin Configuration -----
const int segPins[7] = {3, 5, 6, 9, 10, 11, 12}; // A-G segments
const int digitPins[2] = {7, 8}; // Two 7-seg digits

const int fgInput = 2; // Function generator input
const int ledPins[4] = {A0, A1, A2, A3}; // LEDs
const int switchPins[4] = {4, 5, 6, 13}; // Switch inputs

volatile unsigned long pulseCount = 0; // Counter incremented by ISR
// 7-segment digit encoding (Common Cathode)
byte numbers[10] = {
  B00111111, // 0
  B00000110, // 1
  B01011011, // 2
  B01001111, // 3
  B01100110, // 4
  B01101101, // 5
  B01111101, // 6
  B00000111, // 7
  B01111111, // 8
  B01101111 // 9
};

// ----- Function Generator ISR -----
void fgPulse() {
  pulseCount++; // Increment count on rising edge
}

// ----- Setup -----
void setup() {
  // 7-segment pins
  for(int i=0; i<7; i++) pinMode(segPins[i], OUTPUT);
  for(int i=0; i<2; i++) pinMode(digitPins[i], OUTPUT);

  // LED pins
  for(int i=0; i<4; i++) pinMode(ledPins[i], OUTPUT);

  // Switch pins
  for(int i=0; i<4; i++) pinMode(switchPins[i], INPUT_PULLUP);

  // Function generator input interrupt
  attachInterrupt(digitalPinToInterrupt(fgInput), fgPulse, RISING);
  Serial.begin(9600);
}

```

Function generator counter loop

Listing 3 Part 2 of 2 Display and main loop

```
// ----- Display Digit -----
void displayDigit(int digit, int value) {
  digitalWrite(digitPins[digit], LOW); // Enable digit
  for(int i=0; i<7; i++) {
    digitalWrite(segPins[i], bitRead(numbers[value], i));
  }
  delay(5); // Short delay for multiplexing
  digitalWrite(digitPins[digit], HIGH); // Disable digit
}

// ----- Loop -----
void loop() {
  // Read switches and control LEDs
  for(int i=0; i<4; i++){
    int switchState = digitalRead(switchPins[i]);
    digitalWrite(ledPins[i], switchState == LOW ? HIGH : LOW); // Active LOW switches
  }

  // Display pulse count on 7-segment (max 99)
  int count = pulseCount % 100;
  int tens = count / 10;
  int ones = count % 10;

  // Multiplex two digits rapidly
  displayDigit(0, tens);
  displayDigit(1, ones);

  // Optional: Print count to Serial
  static unsigned long lastPrint = 0;
  if(millis() - lastPrint > 500){
    Serial.print("Pulse Count: ");
    Serial.println(pulseCount);
    lastPrint = millis();
  }
}
```