Gila Monster
Pulse generator v1.0

To meet the need for a pulse generator capable of triggering external instruments in neurostimulation protocols with multiple frequency patterns, the Gila Monster v1.0 was developed. Although numerous DIY solutions are available in online repositories, this project intentionally adopts a minimalist hardware design, centered on an Arduino Uno built around the ATmega328P microcontroller. The design prioritizes the timing engine at the hardware and firmware levels, ensuring that time-critical pulse generation is isolated from user interface operations. Encoder polling and LCD updates are handled via interrupts and non-blocking routines so that UI-related tasks do not introduce latency or timing jitter into the output signal

1. Firmware Specifications

- **Time Base:** Microsecond (μs) resolution derived from the 16 MHz system clock.
- **Output Control:** Direct port manipulation using **PORTD, bit 7 (PD7)** to minimize software overhead.
- **Switching Latency:** ≈ 125 ns (theoretical minimum, limited by the ATmega328P instruction cycle at 16 MHz).
- **External Triggering & Synchronization:** Integrated hardware interrupt support via Pin D3 (INT1), allowing synchronization with external lab equipment. The system supports user-selectable Rising or Falling edge detection with a sub-microsecond response latency.
- **Frequency Range & Resolution:** Programmable output frequency from **0.1 Hz to 500 Hz**, with **0.01 Hz resolution**.
- **Output Logic Level:** 5 V TTL-compatible digital output.
- **Single-Shot Test:** Dedicated hardware-level manual pulse capability via **Pin D6**. This function is governed by a software-based debounce routine and a logical interlock that prevents manual firing while the primary stimulation engine is active (State: ON).
- **Parameter Storage:** Non-volatile EEPROM used to store 13 user-configurable operational parameters, preserved across power cycles.
- **Serial Communication:** UART-based serial interface for **logging and reporting of operational and experimental parameters**, enabling external data acquisition, monitoring, and reproducibility of stimulation protocols.

---

2. Operating Engines

2.1 Deterministic Engine (Continuous [Periodic] / Burst Modes)

The deterministic timing engine computes the output period (**T**) as a direct function of the programmed frequency (**f**), according to:

$$\mathrm{T}\mu\mathrm{s} = \frac{1{,}000{,}000}{f\mathrm{Hz}}$$

In **Continuous Mode**, pulses are generated at a fixed period with cycle-accurate timing.

In **Burst Mode**, pulse delivery is governed by:

- a **pulse counter** defining the number of pulses per train, and a secondary **silent-state timer (Gap)** that enforces the inter-train interval (ITI).

2.2 Non-Periodic Stimulation (NPS) Engine

The Non-Periodic Stimulation (NPS) engine implements a constrained uniform distribution algorithm to generate temporally irregular pulse trains while preserving overall pulse density. Although the underlying random selection is uniform within constrained temporal bounds, the sequential dependency between events produces a heavy-tailed distribution of inter-stimulus intervals (ISI). As a result, shorter ISIs occur more frequently while longer intervals remain possible, yielding a power-law-like temporal structure rather than a purely Poisson or periodic pattern.

- **Algorithm:**

$$t_n = \text{Random}\,(t_{n-1} + \text{Min},\ \text{Remaining Window Space})$$

- **Window Size:** Fixed **1,000 ms** temporal window.
- **Pulse Density:** Constant **N pulses per window** (Pulse Density Modulation).
- **Temporal Structure:** Variable inter-stimulus intervals (ISI) exhibiting a **power-law-like distribution**, increasing temporal unpredictability.

The NPS engine operates independently of the deterministic timing core, ensuring that the event scheduling does not compromise microsecond-level pulse timing accuracy

References by Cota, et al.
https://doi.org/10.1016/j.yebeh.2019.106609
https://doi.org/10.1016/j.yebeh.2008.09.006

---

3. Parameter Matrix & Data Mapping

ID	Parameter	Type	Logic / Range	Resolution
00	Mode	Option	Manual (0) / Triggered (1)	N/A
01	Trigger Edge	Option	Falling (0) / Rising (1)	N/A
02	Output Type	Option	Continuous (0) / Burst (1) / NPS (2)	N/A
04	Frequency	Decimal	0.10 Hz – 500.00 Hz *(Cont. and Burst mode only)*	0.01 Hz
05	Pulse Count	Integer	1 – 999 pulses *(Burst and NPSmode only)*	1 pulse
06	Gap (ITI)	Integer	0 – 999,999 ms *(Burst mode only)*	1 ms
07	Minimum ITI	Decimal	0.00 – 999.99 ms *(NPS mode only)*	0.01 ms
09	Pulse Width	Decimal	Fixed (μs) or Duty Cycle (%)	0.01 ms / %
10	Timer	Integer	0 (Disabled) – 999,999 ms	1 ms

4. Hardware Interface (Pinout)

Digital & Analog Pin Assignment

MCU Pin	Arduino Pin	Direction	Pull-Up	Function	Notes
PD7	D7	Output	Disabled	Pulse Output	High-speed TTL output. i.e. use 220 Ω series resistor for opto-isolation or long cables.
PD3	D3	Input	Disabled	Hardware Trigger (INT1)	External interrupt input. **Sub-*μ*s latency**
PD2	D2	Input	Enabled	Encoder Channel A (CLK)	Quadrature decoding. Internal pull-up reduces noise and wiring complexity.
PD5	D5	Input	Enabled	Encoder Channel B (DT)	Quadrature decoding. Matched with Channel A.
PD4	D4	Input	Enabled	Encoder Push Button (SW)	Active-low user input. Debounced in firmware.
PD6	D6	Input	Enable	Single Shot Button	Manual pulse fire. Debounced (200ms) and interlocked with master state
PC4	A4	I/O	Enabled	I²C SDA	Shared data line. Standard-mode **100 kHz**. External pull-ups optional if LCD provides them.
PC5	A5	Output	Enabled	I²C SCL	Shared clock line. Standard-mode **100 kHz**.

---

5. Serial Communication & Data Validation

The **COMM [12]** function operates as a **diagnostic and validation interface**, providing access to internal configuration and timing behavior via the serial port.

1. **Protocol:** UART serial communication at **9600 baud, 8-N-1**.
2. **Current state-machine configuration and active timing parameters**.
3. **Monte Carlo Validation (NPS Mode):** When operating in **Non-Periodic Stimulation (NPS)** mode, the firmware executes an internal **Monte Carlo–style simulation** over the programmed timer duration. The resulting **inter-trial interval (ITI) sequence** enable offline statistical analysis and verification of ISI variability.

Notes for open-hardware users

- Simulation and data export are decoupled from the real-time timing engine and do not affect pulse generation.

---

6. Operational Constraints

- **Pulse Width Constraint:** If the programmed pulse width (**PW**) exceeds the pulse period (**T**), the firmware automatically clamps **PW** to **T – 10 *μ*s**, ensuring a mandatory low phase between pulses and preventing a constant-high (latched) output state.

- **NPS Timing Constraint:** In **Non-Periodic Stimulation (NPS)** mode, the cumulative minimum timing constraint must satisfy

$$N \times ITImin < 1000\ ms$$

where **N** is the requested number of pulses per window. If this condition is violated, the engine enforces a **fail-safe prioritization of the minimum inter-stimulus interval**, automatically reducing the effective pulse count within the window to preserve temporal validity.

- **Single Shot - Manual Pulse Logic:** The Single-Shot button (D6) is only operational when the device is in a non-generating state (State: OFF). Upon detection of a valid transition (HIGH to LOW), the firmware bypasses the menu-based timing core to deliver an immediate pulse. The duration of this manual pulse is fixed at **10 ms** for diagnostic and calibration purposes.

---

7. Operation Modes

7.1 Continuous Mode (Periodic)

Generates a constant flow of pulses at a fixed frequency.

- **Example Configuration:** 20Hz stimulation with 5ms pulse width.
    - Type: Cont.
    - Freq: 020.00 Hz
    - Width: Fixed Time
    - Pulse: 005.00 ms
    - State: ON

7.2 Burst Mode (Sequenced Trains)

Generates groups of pulses (trains) separated by a long silence interval (**Gap**).

- **Example Configuration:** 10 pulses at 50Hz with 2ms pulse width, repeating every 2 seconds.
    - Type: Burst
    - Freq: 050.00 Hz
    - Count: 010
    - Gap: 001500 ms (Note: Gap + Burst duration = Cycle time)
    - Pulse: 002.00 ms
    - State: ON

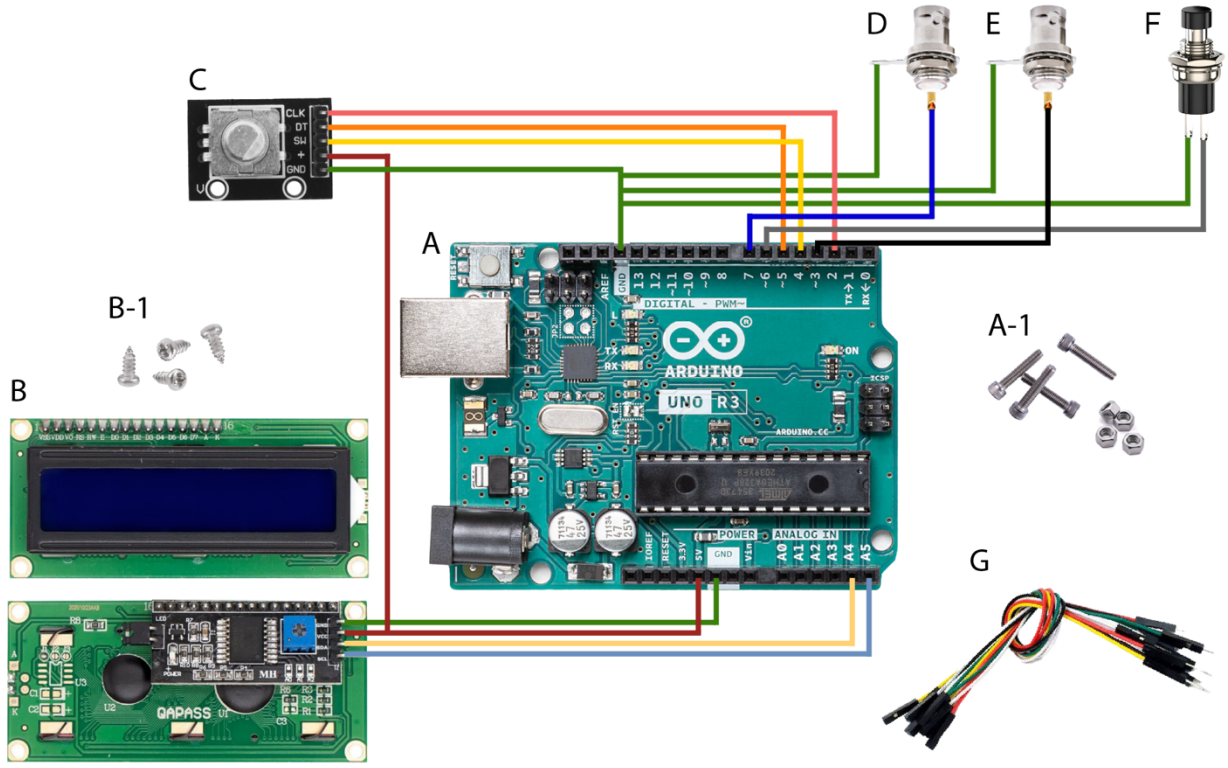
7.3 NPS Mode (Non-Periodic Stimulation)

Generates a specific number of pulses per second at randomized intervals.

- **Example Configuration:** 4 random pulses per second with 100 μs pulse width at least 20ms between them.
    - Type: NPS
    - Count: 004
    - ITIMin: 020.00 ms
    - Pulse: 000.10 mobs
    - State: ON

8. Assembly Instructions

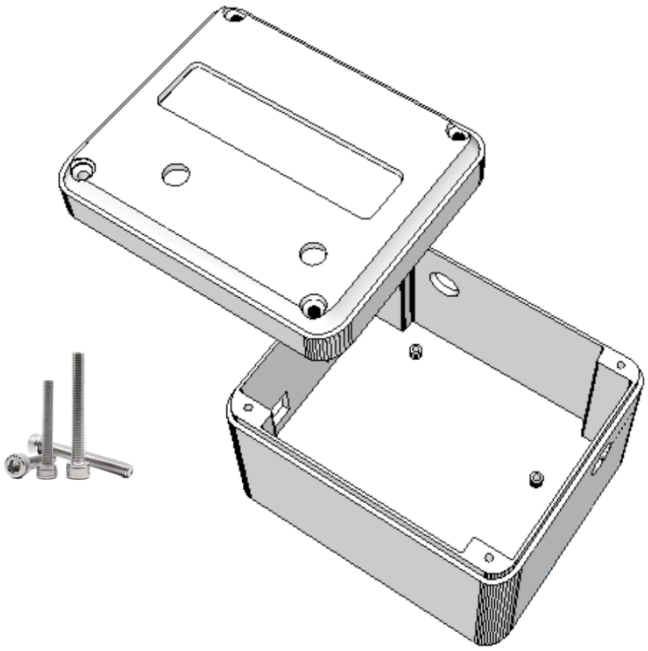
8.1 Electronics



	Materials	Quantity	Link	$
A	Arduino Uno R3	1	Mouser electronics	25.87
A-1	M2 x 12 mm Allen Bolt Socket Cap Screws M2 Hex Nut for Arduino	4	-	-
B	LCD module 16x2, (WxHxD): 80x36x13 mm I2C LCD display module	1	Amazon	10
B-1	Round Head Tap Screw for LCD Ø 2-2.5 mm, L = 3-6 mm)	4	-	-
C	360 Rotatory encoder module KY-040	1	Amazon	~7 (2 un)
D, E	Female BNC panel-mount	2	-	-
F	Push Button RS PBS 110	1	Amazon	~.60
G	Cables, different colors (e.g. Ø 2.5 mm, cross section 0,5 mm², length 10-15 cm)	~15	Amazon	~5 (40 un)

8.2 3D parts

- PLA case (WxHxD): 7.5 x 6 x 8.5 cm
- 4 M2 x 20 mm Allen Bolt Socket Cap Screws



This project was developed by an enthusiastic punk. I know that generating pulses with an Arduino is something even 7-year-old kids learn at school to blink LEDs. So be patient. [A.C.A.B]