Two-Player Reaction Time Game

Introduction

A two-player reaction time game inspired by the Romanian TV show “Ce spun romanii” and by the start lights of Formula 1. Three red LEDs light up one by one, then turn off simultaneously after a random delay — this is the start signal. The first player to press their button wins: their blue LED lights up and an OLED display shows the winner and their reaction time in milliseconds. The goal of the project is to create a simple yet engaging interactive game that tests the reflexes of two players. The idea started from TV shows where contestants must react as fast as possible to a visual signal. The project is useful as a practical demonstration of hardware interrupts, timers and I2C communication on an AVR microcontroller, while also being an accessible example of an interactive embedded system.

General Description

The project is structured around the ATmega328p XMINI microcontroller, which coordinates all hardware and software modules.

Hardware modules:

- 3× Red LEDs — connected to digital output pins, light up sequentially during the countdown phase, then turn off simultaneously as the start signal
- 2× Player buttons — connected to external interrupt pins INT0 (PD2) and INT1 (PD3), each assigned to one player
- 2× Blue LEDs — connected to output pins, one lights up to indicate the winning player
- OLED display SSD1306 — communicates with the microcontroller via the I2C protocol (SDA/SCL), displays the winner and reaction time in milliseconds

Software modules:

- Interrupts (Lab 2) — button presses are handled via hardware interrupts, ensuring the fastest possible response time detection
- Timers (Lab 3) — Timer1 measures the reaction time in milliseconds; Timer0 generates the random delay between the countdown and the GO signal
- I2C communication (Lab 6) — the TWI hardware module sends data to the OLED display to render text

Hardware Design

Bill of Materials

Component	Quantity	Details
ATmega328p XMINI	1	Main development board
OLED display	1	0.96” SSD1306, I2C, 128x64px
LED red	3	5mm, countdown signal
LED blue	2	5mm, winner indicator
Resistor 220Ω	5	Current limiting for LEDs
Push button	2	12x12mm tactile with round cap
Breadboard	1	830 points
Jumper wires	19	Male-to-male

Pin Connections

Pin	Component	Role
PB0	LED1 red + 220Ω	Countdown LED 1
PB1	LED2 red + 220Ω	Countdown LED 2
PB2	LED3 red + 220Ω	Countdown LED 3
PB3	LED4 blue + 220Ω	Winner indicator Player 1
PB4	LED5 blue + 220Ω	Winner indicator Player 2
PB7	SW0 (on-board)	Start / restart game
PD2	Button Player 1	External interrupt INT0
PD3	Button Player 2	External interrupt INT1
PC4 (SDA)	OLED display	I2C data
PC5 (SCL)	OLED display	I2C clock
VCC	OLED display	3.3V - 5V power
GND	All components	Common ground

Hardware Schematic Diagram

Signal diagram

The game follows a fixed sequence of states:

1. **IDLE** — waiting for start button (PB7)
2. **COUNTDOWN** — LED1, LED2, LED3 light up one by one, 1 second apart. Player button presses during this phase trigger a false start (INT0/INT1).
3. **RANDOM DELAY** — all 3 LEDs remain on for a random interval between 1 and 3 seconds.

4. **GO** — all LEDs turn off simultaneously. Timer1 starts measuring reaction time.
5. **RESULT** — first player to press wins. Winner LED lights up, OLED displays both reaction times. If a false start occurred, the offending player is disqualified.

Software Design

Development Environment

The firmware was developed using **PlatformIO** integrated into **Visual Studio Code**, targeting the ATmega328P with the Arduino framework (used only as a build system — all peripheral control is done via direct AVR register access, without any Arduino library functions).

Upload is done via the **xplainedmini** protocol using avrdude, over USB.

Third-party Libraries and Sources

No external libraries were used. All drivers were implemented from scratch using AVR-libc headers only:

- `<avr/io.h>` — register definitions
- `<avr/interrupt.h>` — ISR macro and sei()/cli()
- `<util/delay.h>` — _delay_ms() for debouncing

The SSD1306 OLED driver, TWI/I2C driver, USART driver, and timer modules were written entirely by hand, inspired by the laboratory implementations provided during the PM course.

File Structure

File	Role
`src/main.cpp`	Game logic, state machine, ISR definitions
`src/uptime.cpp`	Timer2 CTC — 1ms system tick counter
`src/timers.cpp`	Timer1 CTC — reaction time measurement
`src/twi.cpp`	I2C/TWI driver using AVR hardware TWI module
`src/ssd1306.cpp`	SSD1306 OLED driver with 5×7 bitmap font
`src/usart.cpp`	UART driver for serial debug output
`include/uptime.h`	uptime_ms() declaration
`include/timers.h`	Timer1_init/start/stop/get_ms declarations
`include/twi.h`	TWI function declarations and frequency config
`include/ssd1306.h`	SSD1306 function declarations
`include/usart.h`	USART function declarations

Algorithms and Data Structures

Game state machine

The game runs as a sequential state machine inside the main loop:

1. `IDLE` — display shows “READY”, waiting for SW0 (PB7) press
2. `COUNTDOWN` — LED1, LED2, LED3 light up one by one at 1s intervals using non-blocking timing via uptime_ms(). External interrupts INT0/INT1 are active during this phase to detect false starts.
3. `RANDOM DELAY` — all 3 LEDs remain on for a random interval (1–3 seconds), generated via rand() seeded with uptime_ms()
4. `GO` — LEDs turn off, Timer1 starts counting reaction time
5. `RESULT` — first player to press wins. If a false start was detected, the offending player is disqualified. Results are shown on OLED and serial.

Non-blocking timing

All delays during the countdown and random delay phases use the pattern:

```
uint32_t t = uptime_ms();
while ((uptime_ms() - t) < 1000 && !false_start_p1 && !false_start_p2);
```

This allows the main loop to remain responsive to false start interrupts even while waiting.

False start detection

External interrupts INT0 (PD2) and INT1 (PD3) are configured for falling edge detection. ISRs set volatile flags only if `game_started == 0`:

```
ISR(INT0_vect) { if (!game_started) false_start_p1 = 1; }
ISR(INT1_vect) { if (!game_started) false_start_p2 = 1; }
```

Reaction time measurement

Timer1 is initialized in CTC mode at 16MHz with prescaler 8, generating an interrupt every 1ms (OCR1A = 1999). A volatile counter `reaction_ticks` increments in the ISR. Timer1_start() resets and enables it; Timer1_get_ms() reads the counter atomically using cli()/sei().

I2C communication

The TWI hardware module is configured for 100kHz SCL frequency. The SSD1306 driver sends initialization commands on startup, then writes character bitmaps page by page using twi_start/twi_write/twi_stop sequences.

OLED font rendering

Characters are stored as a 5×7 bitmap font in Flash memory (PROGMEM) for 58 ASCII characters (space through Z). Each character is 5 bytes wide plus 1 byte spacing. ssd1306_print_str() renders a string at a specified page row (0–7).

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF