Mini Game Console

- **Author: Căzan Mihnea-Andrei**
- **Group: 335CA**

Introduction

This project consists of building a retro mini game console capable of running classic arcade games (such as Snake and Breakout).

- **The main goal** is to explore and practically apply the interfacing of an 8-bit microcontroller (ATmega328P) with various input and output peripherals.
- **The idea** originated from the desire to recreate the experience of 80s/90s arcade games on a minimal hardware platform, demonstrating how limited memory resources (32KB Flash, 2KB RAM) can provide a complete interactive experience.
- **This project is highly useful** from an educational standpoint, as it deepens the understanding of fundamental concepts (I2C communication protocols, ADCs, Timers, PWM, and Hardware Interrupts), while also resulting in a functional and entertaining gadget.

General Description

The console is structured into three main blocks: input, processing, and output.

- The **ATmega328P** microcontroller acts as the central processing unit. It continuously reads analog signals from the joystick using the internal ADC (to determine movement direction) and monitors hardware interrupts or digital pin states for the action buttons.
- Based on the implemented game logic (Game Loop), the microcontroller updates the system state and sends graphical data packets to the **OLED display** via the I2C hardware interface.
- Simultaneously, the system uses internal Timers to generate PWM (Pulse Width Modulation) signals to a **passive buzzer** (to play specific sound effects) and controls an **RGB LED** to provide quick visual feedback regarding the current state of the console (e.g., green for active gameplay, red for Game Over, blue for menu).

System Block Diagram:

Hardware Design

List of components used for the project (Bill of Materials):

- **ATmega328P Xplained Mini** Development Board (8-bit AVR Microcontroller)
- **0.96” OLED Display** (SSD1306 Controller, I2C Interface, 128×64 resolution)
- **Biaxial Analog Joystick Module** (KY-023 type) with integrated push button (SW)
- **Passive Buzzer 3.3V** (for generating sound effects)
- **5mm RGB LED** (Common Cathode, for visual state indication)
- **2 x Tact Switch Buttons 6x6mm** (For START and PAUSE functions)
- **4 x 220Ω Resistors** (3 for RGB LED current limiting, 1 for buzzer protection)
- **830 Tie-Points Breadboard** (for solderless assembly)
- **Dupont Wire Set** (Male-to-Male and Female-to-Male)

Electrical Scheme:

Connection Table

Component	Component Pin	ATmega328P Pin	Details / Functionality
RGB LED	Red Anode	PD6 (D6)	PWM controlled (Timer0 OC0A)
	Green Anode	PD5 (D5)	PWM controlled (Timer0 OC0B)
	Blue Anode	PB3 (D11)	PWM controlled (Timer2 OC2A)
	Cathode	GND	Common Ground
Joystick	VRx	PC0 (ADC0)	Analog read (X-axis)
	VRy	PC1 (ADC1)	Analog read (Y-axis)
	SW	PD2 (D2)	Hardware Interrupt (INT0, Internal Pull-up)
	VCC	5V	Power Supply
	GND	GND	Ground
OLED Display	SDA	PC4 (A4)	I2C Data line (TWI)
	SCL / SCK	PC5 (A5)	I2C Clock line (Fast Mode 400kHz)
	VCC	5V	Power Supply
	GND	GND	Ground
Buzzer	Pin (+)	PB1 (D9)	Through 220Ω resistor (Timer1 OC1A)
	Pin (-)	GND	Ground
Buttons	START	PD3 (D3)	Hardware Interrupt (INT1, Internal Pull-up)
	PAUSE	PB0 (D8)	Pin Change Int (PCINT0, Internal Pull-up)

Communication Protocols and Hardware Peripherals

To seamlessly manage the display, analog input, and visual/audio effects concurrently without blocking the main loop, the system heavily utilizes the microcontroller's hardware peripherals:

- **I2C (TWI):** Enabled on pins PC4/PC5. Configured in **Fast Mode (400 kHz)** to ensure rapid transfer of the frame buffer to the OLED display (maintaining ~30 FPS).
- **ADC:** ADC0 and ADC1 channels are used for smooth joystick reading, utilizing a 128 prescaler for signal stability.
- **PWM Signals:** Timers 0 and 2 generate hardware PWM signals on 3 independent pins to finely

control the brightness and color mixing of the RGB LED.
- **Timer1 (CTC Mode):** Generates precise hardware frequencies for musical notes sent to the buzzer, offloading the CPU from using blocking delay functions.

Software Design

Descrierea codului aplicaţiei (firmware):

- mediu de dezvoltare (if any) (e.g. AVR Studio, CodeVisionAVR)
- librării şi surse 3rd-party (e.g. Procyon AVRlib)
- algoritmi şi structuri pe care plănuiţi să le implementaţi
- (etapa 3) surse şi funcţii implementate

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