MP3 Player

Author: Necula Mihail

Group: 333CA

Introduction

What does it do?

The project is an **MP3 player** based on the **AVR ATmega2560** architecture. It allows for **audio playback, track navigation and smart brightness adjustment** via an ambient light sensor.

Other key technical feature is the **multi-step speed control** (x0.5, x0.75, x1, x1.25, x1.5). The speed adjustment is available only if the audio files are processed and uploaded to the microSD card using my software. The app pre-encodes the tracks at various speeds, allowing the system to switch between files seamlessly.

Additionally, the system features **persistent memory**. Using the internal EEPROM, the device saves the current volume level and the last played song. This ensures that at restart, the player resumes exactly where it left off.

What is its purpose?

The goal is to provide a convenient, offline and distraction-free listening experience.

What was the initial idea?

The inspiration for this project was nostalgia for the dedicated MP3 player which I used during my childhood.

General description

Block Diagram

Hardware Modules Description

The hardware is divided into four main functional blocks:

- **Central Processing Unit (ATmega 2560):** Acts as the system “Master.” It orchestrates all operations by managing the **I2C** bus for the display and light sensor, the **UART** interface for the MP3 module and **GPIO** pins for user inputs and status feedback.
- **User Interface & Sensing:**
  - **Light Sensor (GY-302 BH1750):** Communicates via **I2C** to provide lux values for the “Smart Brightness” feature.
  - **Rotary Encoder & Buttons:** Provide the primary user interface for navigation, volume control, speed control and playback commands.
- **Audio Subsystem:**
  - **MP3 Decoder (VS1053B):** Offloads the heavy task of MP3 processing from the MCU. It accesses the MicroSD Card via **SPI**.
  - **MicroSD Card:** Stores the mp3 files.
  - **Amplifier (PAM8403):** Takes the low-level analog output from the decoder and drives the Visaton K50 speaker.
  - **Speaker:** The final output transducer of the audio chain. It converts the amplified analog electrical signals from the PAM8403 into audible sound waves.
- **Visual Feedback:**
  - **1.3” OLED (SH1106):** Displays metadata and system status via **I2C**.
  - **Status RGB LED:** Provides immediate visual cues (e.g. playback state or errors) using **PWM**.

Software Modules Description

The software is organized into logical **modules** to ensure maintainability:

- **Track Manager App:** A PC-side utility used to pre-encode audio files at different speeds (x0.5, x0.75, x1, x1.25, x1.5), allowing the hardware to switch playback speeds without digital artifacts.
- **Audio Manager:** When a track is uploaded via the app, the user can select the option to generate five synchronized versions of the audio file at different speeds (x0.5, x0.75, x1, x1.25, x1.5). This allows the hardware to switch playback speed instantly by jumping between files, avoiding the digital artifacts. Moreover, the app includes the functionality to delete a song and manage playlists.
- **Input Manager:** Implements a hybrid input strategy (Interrupts for rotation, Polling for buttons).
- **Display & UI Engine:** Manages OLED rendering and the RGB led.
- **EEPROM Manager:** Ensures persistence by saving the current settings into the MCU's non-volatile memory.

Hardware Design

Bill of Materials

Component	Amout
MEGA 2560 Development Board (16U2)	1
3.5mm 3-Pole Male Jack to Terminal Block Adapter	1
VS1053B MP3 Audio Decoder with MicroSD Card Slot	1

10-Pin Extra Tall Male Header - Straight	1
GY-302 Digital Light Sensor (BH1750, I2C)	1
1.3” White OLED Display (SH1106, I2C)	1
Visaton K50 Speaker (50mm diameter, 8 Ω)	1
PAM8403 Stereo Audio Amplifier Module (2x3W)	1
8GB MicroSD Card	1
MB-102 Breadboard (830 Points)	1
AC/DC Power Adapter (5V, 2A, 5.5×2.5mm)	1
Female DC Barrel Jack to Screw Terminal Adapter	1
Tactile Switch Button (12x12x7.3mm)	3
Tactile Switch Cap (for 12x12x7.3mm button)	3
Encoder Module	1
5mm Common Cathode RGB LED	1
220 kΩ Resistor (1/4 W)	3
1 kΩ Resistor (1/4 W)	2
10 kΩ Resistor (1/4 W)	2
1 µF Electrolytic Capacitor (50V)	2
100 µF Electrolytic Capacitor (25V)	2
1000 µF Electrolytic Capacitor (25V)	1
100 nF Ceramic Capacitor (THT, 50V)	4
Clipband	1
10 cm Male-to-Male Jumper Wire	?
10 cm Female-to-Male Jumper Wire	?
20 cm Male-to-Male Jumper Wire	?

Software Design

Input Strategy: Interrupts vs. Polling

A critical design choice was made to balance CPU responsiveness with input accuracy. The system distinguishes between high-speed dynamic signals and slow mechanical transitions.

- **Encoder Rotation (Hardware Interrupts):**
  - **Why interrupts?:** Fast rotation generates pulses in the millisecond range. If handled via polling, these pulses would be missed whenever the MCU is busy with I2C communication (OLED) or UART (Audio). Interrupts force the MCU to pause its current task and register every “click” of the rotation instantly.
  - **Debouncing:** To avoid using CPU-freezing delays during rotation tracking, we implement a Quadrature State Machine. This software logic only validates a step if the transition between signals follows the correct Gray Code sequence (e.g., 01 → 11 → 10).
- **Tactile Buttons & Encoder Click (Software Polling):**
  - **High-Frequency Sampling:** Although handled via polling, the button is sampled thousands of times per second. Since a human press is very slow (~100ms) compared to the MCU loop speed (< 10ms), it is impossible to miss a press.

- **Why not interrupts?:** Polling is “cheaper” for the CPU. An interrupt forces a full context switch (saving registers, jumping to ISR), which is overkill for a slow button. Mechanical switches bounce violently. Using interrupts would force the MCU to stop and restart dozens of times for a single click, causing audio glitches or UI lag.
- **Debouncing:** We use a time-based “lockout” to ensure a single clean press is registered: if (current_time - last_press_time > DEBOUNCE_THRESHOLD). This ensures that after the first detection, any subsequent bounces are ignored for 50ms, while the MCU continues to drive the OLED and Audio without any freezing delays.

Results obtained

Conclusions

Download

Repository

Journal

- **24.04.2026:** Project Theme Selection and Approval
- **03.05.2026:** Components Order
- **05.05.2026:** Documentation Page Creation

Bibliografy

Hardware

- ATmega2560 Datasheet

Software

Export to PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2026/victor.stoica0203/mihail.necula

Last update: **2026/05/11 15:04**