Faculty of Information Technology
Brno University of Technology

IMP
Microprocessors and Embedded Systems

WiFi Metronome

Andrii Bondarenko (xbonda06)

December 8, 2024

Contents

1 Introduction **2**

2 Hardware Setup **2**

3 Preparation **2**

4 Implementation **3**

4.1 Timing Logic 3

4.2 Web Interface 3

5 User Interface **3**

6 Conclusion **3**

7 References **4**

Abstract

The project aims to create a metronome for musicians using the ESP32 development board. This device features a buzzer to produce beats, an integrated web interface for configuration, and the ability to adjust volume, tempo (BPM), and time signature (e.g., 4/4, 3/4, 2/4). The system is implemented using the ESP-IDF framework, and the device operates as a Wi-Fi access point, allowing users to connect and configure the metronome locally.

Video Presentation

A video demonstration of the project's functionality is available at the following link:
`https://www.youtube.com/shorts/ZsZiJpgmlRk`

1 Introduction

This document describes the design and implementation of a metronome for musicians. The primary goal was to develop a user-friendly device capable of producing metronome beats with configurable parameters such as volume, tempo, and rhythm pattern. The project requirements included:

- Use of ESP32 and ESP-IDF framework.
- Device operates as a Wi-Fi access point.
- Web-based interface for real-time configuration.
- Adjustable PWM output for controlling buzzer volume.
- Precise beat timing using internal timers.

2 Hardware Setup

The hardware used in this project includes:

- ESP32 development board.
- Piezo buzzer.
- Breadboard, resistor, and jumper wires for connections.

The buzzer is connected as follows:

- Buzzer positive terminal to GPIO 25.
- Buzzer negative terminal to ground (GND).

The ESP32 board is powered via USB, and all connections are made using a breadboard.

3 Preparation

The project was developed using Visual Studio Code with the PlatformIO plugin. The ESP-IDF framework was chosen for its robustness and low-level control over hardware features.

Key preparation steps included:

- Installing ESP-IDF and PlatformIO.
- Configuring Wi-Fi Access Point settings in the source code.
- Setting up the development environment and testing basic functionality.

4 Implementation

The metronome functionality was implemented using the following key features:

- **PWM Output:** The LEDC driver was used to generate PWM signals for controlling the buzzer volume.
- **Beat Generation:** Internal timers ensure precise timing for beats. A strong beat (higher frequency) is played for the first note in a measure, and weak beats (lower frequency) are used for the rest.
- **Wi-Fi Access Point:** The ESP32 operates as an access point, enabling users to connect directly to the device without external network infrastructure.
- **Web Interface:** A built-in HTTP server allows users to configure tempo (BPM), volume, and time signature through a responsive HTML form.

4.1 Timing Logic

The tempo and time signature are controlled by the variables `bpm` and `time_signature`. The beat timing is calculated dynamically to match the selected tempo.

4.2 Web Interface

The HTML interface includes input fields for:

- Volume (0 to 255).
- BPM (30 to 300).
- Time signature (4/4, 3/4, 2/4).

Styling is added using CSS to enhance usability.

5 User Interface

The web interface is accessible via the ESP32's local IP address (`http://192.168.4.1/control`). Users can configure the metronome parameters in real-time. The interface includes:

- Numeric input for Volume.
- Numeric input for BPM.
- Dropdown for time signature selection.

The changes take effect immediately upon submission.

6 Conclusion

This project successfully implements a configurable metronome using the ESP32. The combination of hardware PWM, web-based configuration, and precise timing makes it a practical tool for musicians. Future enhancements could include storing user presets and adding more rhythm patterns.

7 References

[1] (MDN), M. D. N. *CSS Reference Guide*. 2024. Available at: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference. Accessed: 2024-12-03.

[2] Systems, E. *ESP-IDF Programming Guide*. 2024. Available at: https://docs.espressif.com/projects/esp-idf/en/latest/. Accessed: 2024-12-03.

[3] Team, P. *PlatformIO IDE Documentation*. 2024. Available at: https://platformio.org/. Accessed: 2024-12-03.

[4] (W3C), W. W. W. C. *HTML Forms Documentation*. 2024. Available at: https://www.w3.org/TR/html52/sec-forms.html. Accessed: 2024-12-03.