BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Microprocessors and Embedded Systems – Project

Bluetooth Metronome

December 4, 2023

Michal L'aš (xlasmi00)

Contents

1 Introduction

The project's objective was to develop a metronome tailored for musicians, with Bluetooth functionality for remote control. The chosen platform for implementation was the ESP32 chip. The metronome was designed to offer adjustable settings for speed, volume, and rhythm. Users can easily control these parameters through a simple HTML interface, leveraging the capabilities of Web Bluetooth technology.

The resulting application was created using the C programming language and the `esp-idf` framework. Primary references for this project were derived from the `esp-idf` documentation [3] and examples available in the esp-idf GitHub repository [2], as well as various online sources. Notably, resources from the websites about web bluetooth technology [1] and insights on Web Bluetooth, Generic ATTribute (GATT), and Generic Access Profile (GAP) [4] played a key role in shaping the implementation.

2 Hardware connection

GPIO pin 14 serves as the control interface for the external buzzer, as illustrated in Figure 1. Simultaneously, the built-in LED, located on GPIO pin 2, is utilized in the system. In the event of a successful client connection to the device, the integrated LED flashes three times as an indicator.

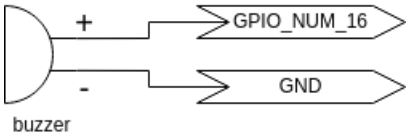

Figure 1: Buzzer connection to ESP32

3 Design

The project can be divided into 2 parts:

1. Metronome Control:

- The metronome's tempo, regulated by a GPTimer (General Purpose Timer), ensures periodic ticking. This timer triggers a buzzer sound event, managed through the `init_metronome_timer()` function.
- The metronome can be activated with `run_metronome()` and halted with `stop_metronome()`. To set the tempo (BPM), use `set_metronome_tempo(bpm)`.
- Before commencing the metronome, it is essential to initialize Pulse Width Modulation (PWM) for GPIO pin 14 (buzzer pin) using the `pwm_init()` function.
- Buzzer triggering and volume are modulated using `set_duty_cycle(duty)`, while the buzzer's frequency or tone is adjusted with `set_buzzer_frequency(frequency)`.

2. Bluetooth Communication:

- The `init_ble()` function kickstarts Bluetooth Low Energy (BLE) settings and initiates a thread for capturing Bluetooth events. Two UUIDs, one for Bluetooth service and another for Bluetooth characteristic, were generated using the UUID generator tool [5].

- `ble_app_advertise()` initiates device advertising. Upon client connection, `ble_gap_event()` captures and signals a successful connection by flashing the built-in LED thrice this function also handles device disconnection. Upon disconnection, the application reinitiates advertising, enabling the establishment of another connection. This ensures continuous accessibility for potential users seeking to connect with the device.
- Messages from the client are intercepted by `ble_gatt_event()`, with processing managed by the `write_client()` function. This establishes comprehensive control over all metronome functions for the client.

This configuration empowers the client to have comprehensive control over all the metronome functions outlined earlier.

Additionally, the implementation features a client comprising a straightforward HTML page (refer to image 2) accompanied by a script, `script.js`. This script is responsible for messaging to manipulate the metronome. Each message starts with the first byte denoting the action code, succeeded by supplementary data. For example, to establish the tempo, the code is `0x30`, followed by the tempo value. This messaging protocol ensures a structured and efficient means for the client to convey instructions to control various metronome parameters.

Codes:

- `0x30`: Set tempo message
- `0x31`: Set rhythm/beat message
- `0x32`: Set volume message
- `0x33`: Start metronome message
- `0x34`: Stop metronome message
- `0x35`: Initialize metronome values (set metronome at connection)

4 Requirements

- ESP32 with bluetooth
- ESP-IDF Release v5.1.2 or newer
- Chrome browser (version 119.0 or newer)

5 Limitations and known bugs

- BLE functionality is operational within secure contexts, necessitating the use of the HTTPS protocol for the client application. Simple HTTP without TLS is not supported. (application can be deployed also on localhost)
- BLE is working only in secure contexts. So it needs to use HTTPS protcol for a client application (it works on localhost as well). Simple HTTP without TLS is not supported.
- The + and − buttons in the client application currently lack an implemented debounce timer. Consequently, rapid button presses may result in collisions, wherein a previously sent message is not processed before the next one arrives (this issue manifests only when the client is actively connected to the device).

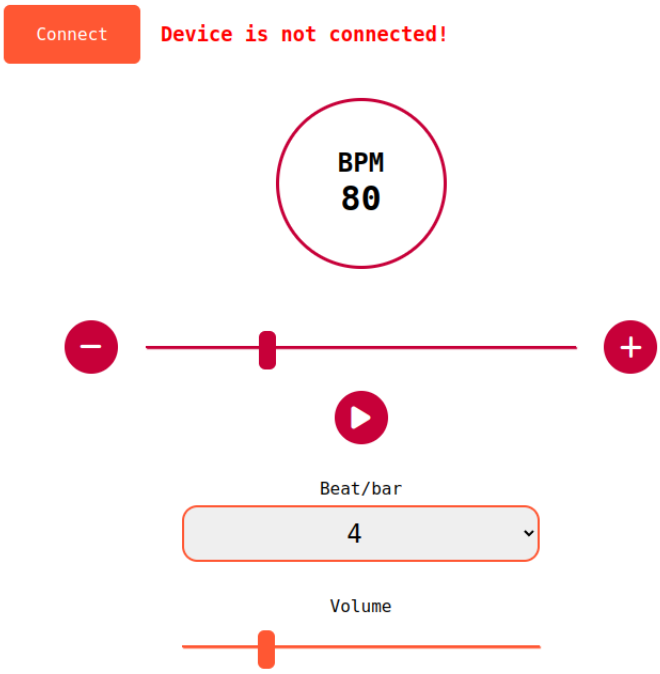

Figure 2: Bluetooth Metronome Client API

6 Conclusion

The project managed to implement all the assignment points.

Literature

[1] BEAUFORT, F. *Communicating with Bluetooth devices over JavaScript* [[online]]. July 2015. [04.04.2023]. [01.12.2023]. Available at: https://web.dev/bluetooth/.

[2] ESPRESSIF SYSTEMS. *esp-idf* [[online]]. [01.12.2023]. Available at: https://github.com/espressif/esp-idf/tree/c8243465e45489835d645bf217a6929fd0c01b7f.

[3] ESPRESSIF SYSTEMS. *ESP-IDF Programming Guide* [[online]]. [01.12.2023]. Available at: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html.

[4] TOWNSEND, K. *Introduction to Bluetooth Low Energy* [[online]]. March 2014. [01.12.2023]. Available at: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/.

[5] TRANSPARENTECH LLC. *Online UUID Generator* [[online]]. Available at: https://www.uuidgenerator.net/.