Mikroprocesorové a vestavěné systémy – projektová dokumentace Bluetooth vysílač morseovy abecedy

Vojtěch Dvořák (xdvora3o)

15. prosince 2022

Úvod

Cílem projektu bylo vytvořit zařízení založené na platformě ESP32, které by bylo schopné pomocí bezdrátové technologie Bluetooth přijímat znaky a prostřednictvím bzučáku v kombinaci s morseovou abecedou je převádět do akustické podoby.

Další požadavky zadání také kladly nároky na použití některých datových struktur (fronty) a vyžadovaly přítomnost mechanismu, jenž by umožnila regulovat hlasitost bzučáku. Vstup má být zadáván prostřednictvím webového rozhraní s využitím WebBluetooth.

V této projektové dokumentaci jsou shrnuty nejdůležitější teoretické poznatky pro pochopení fungování řešení, jsou zde popsány některé vybrané části implementace a nakonec je zde přítomen i uživatelský manuál. Ten popisuje veškerou funkcionalitu vytvořeného systému.

Pro řešení projektu byl využit rámec ESP-IDF pro programování platformy v jazyce C/C++ a byli rovněž použity příklady výrobce jakožto základ pro některé funkce a struktury.

1 Teoretická část

Tato kapitola shrnuje nejdůležitější teoretické poznatky potřebné pro implementaci projektu. Kromě informací uvedených níže mi také při vývoji pomohla dokumentace ESP-IDF[1] a oficiální repozitář společnosti Espressif Systems[2], součástí kterého jsou také příklady.

1.1 Bluetooth

Jedním z těžišť toho projektu je zajištění bezdrátové komunikace mezi klientským zařízením vysílajícím data a výsledným zařízením, které je zpracovává.

Bluetooth je standard pro bezdrátovou komunikaci mezi zařízeními na krátkou vzdálenost. Důraz při jeho návrhu byl kladen také na nízkou spotřebu a nízkou cenu hardwaru. Na úspěchu klasického Bluetooth standardu, který je už roky používán v různých oblastech, je postaven nový, nezávislý standard Bluetooth Low Energy (LE nebo BLE), jenž se ještě více snaží snížit příkon při zachování stejné vzdálenosti mezi zařízeními. [3, str. 6].

Jelikož softwarové rozhraní WebBluetooth, které je využito pro implementaci vysílače (klienta) podporuje primárně standard BLE, zaměříme se právě na něj.

Podobně jako klasické síťové technologie, tak i BLE disponuje vrstevnatou architekturou. Z pohledu implementace bude důležitá horní část zásobníku – tedy protokoly ATT, GATT, GAP.

ATT neboli Attribute Protocol definuje role klienta a serveru. Server si udržuje sadu atributů, ke kterým může klient přistupovat prostřednictvím žádostí [2].

GATT neboli Generic Attribute profile definuje způsob jakým si tato zařízení vyměňují data. Server (také GATT server), typicky nějaké zařízení, kterého se jiným zařízením (klient nebo GATT klient) dotazujeme na data obsahuje GATT profil, což je sada služeb, jež tento server poskytuje [5].

Služby jsou identifikovány identifikátory UUID, které jsou dlouhé 16-bitů, v případě oficiálně uznávaných služeb, nebo 128-bitů, v případě služeb uživatelsky definovaných [5].

Služby dále obsahují sadu charakteristik, které obalují již surová data, které klient čte nebo do nich zapisuje. Opět je v jejich případě používán UUID identifikátor s tím rozdílem, že zde si ho můžeme libovolně vybrat [5].

GAP neboli Generic Access Profile definuje, jakým způsobem mohou zařízení podprující BLE navazovat spojení

[1] `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html`
[2] `https://github.com/espressif`

a upozorňovat na sebe zařízení v okolí (provádět tzv. advertising). Periferie (např. myš) rozesílá v daných intervalech (20ms-2s) všem okolním zařízením malé množství dat (31B), které obsahují informace o tomto zařízení a informace potřebné pro navázání spojení s tímto zařízením. Dalším způsobem, jakým tato data šířit je možnost si je od periferie vyžádat prostřednictvím Scan Reponse Reqequst zprávy (volitelné). Zajímavým využitím advertising procesu je možnost BLE broadcastingu [4].

1.2 Pulzně šířková modulace

Realizace analogového výstupu mikrokontroléru je značně problematická. Analogové obvody jsou drahé a trpí nedostatky jako je například šum. Číslicové signály, se kterými mikrokontrolér pracuje, navíc musí být konvertovány na příslušnou analogovou úroveň, k čemuž je zapotřebí další periferie, energie apod.

Efektivním řešením pro řízení analogových obvodů skrze MCU je použití tzv. PWM neboli pulzně šířkové modulace. Jedná se v podstatě o zakódování analogové úrovně napětí do série digitálních impulsů, jejichž integrací v čase získáme stejné množství energie, jako bychom obvod řídili příslušným analogovým signálem. Stále se však jedná o signál digitální, neboť je v každém časovém okamžiku úroveň PWM signálu buď 1 nebo 0 (v ideálním případě) [1].

V kontextu toho projektu použijeme PWM pro regulaci hlasitosti bzučáku. Ačkoliv možná teorie této technologie zní jednoduše, je k její realizaci zapotřebí nějaký čítač, časovač a také komparátor. Časovač na určité frekvenci inkrementuje hodnotu čítače a při dosažení určité hodnoty čítače (která je detekována pomocí komparátoru) je signál přepnut do opačné logické hodnoty, čímž je vytvářen průběh PWM signálu. Střída (tzv. duty cycle) tohoto signálu je regulována skrze hodnotu, se kterou je čítač porovnáván komparátorem.

2 Implementace

Funkcionalita programového řešení přijímače je pomocí modulů rozdělena na tři části – zajištění bezdrátové komunikace, ovládání výstupů a hlavní tělo programu a překlad znaků. Součástí řešení je také ale vysílač založený na rozhraní WebBluetooth pro JavaScript.

2.1 Bezdrátová komunikace

Prvním krokem bylo zprovoznění bezdrátové komunikace na platformě ESP32 (konkrétně se jednalo o vývojovou desku Wemos D1 R32[3]) mezi GATT klientem využívajícím WebBluetooth a GATT serverem v podobě mikrokontroléru ESP32. Protože se jedná o poměrně komplexní záležitost, je implementace GATT serveru založena na příkladech z oficiálního repozitáře výrobce mikrokontroléru (pochopitelně značně upravených pro účely tohoto projektu). Nejdříve je provedeno povolení a inicializace bluetooth modulu. V rámci volání funkce `bluetooth_init`.

V rámci inicializace jsou zaregistrovány obslužné funkce pro příchozí události a to jak pro fázi advertisingu, která se řídí GAP protokolem, tak pro komunikaci klient-server řízenou GATT protokolem. Inicializace advertisingu zahrnuje inicializaci struktury pro advertising pakety a pro scan response pakety (viz Teoretická část) a také registraci funkce `gap_event_handler`. Událostmi, které jsou toutu funkcí obslouženy jsou například změna parametrů spojení nebo začátek a konec advertisingu.

Během inicializace GATT serveru se obdobně zaregistruje obslužná funkce `gatts_profile_morse_code_event_handler`. Ta kromě událostí pro přidání profilů a charakteristik, událostí připojení a odpojení klienta obsluhuje také události pro čtení a zápis charakteristik. Ačkoliv všechny události mají v procesu bezdrátové komunikace a její inicializace svůj význam, zcela zásadní je událost, v rámci které jsou se službou GATT serveru asociovány charakteristiky. Charakteristikami jsou obecně řetězce osmibitových znaků. Délku tohoto řetězce určíme během inicializace GATT serveru.

Zde oproti jiným projektům je situace o něco zjednodušená protože jedinou charakteristikou pro čtení je v rámci toho projektu aktuální úroveň hlasitosti. Vzhledem k citlivosti bzučáku na vstupní napětí bude bohatě stačit pro specifikování úrovně hlasitosti jedno osmibitové číslo. Tudíž i tato charakteristika bude mít velikost 1x8bitů. Tato charakteristika může být klientem jak čtena (např. pro zobrazení úrovně hlasitosti v UI vysílače), tak zapisována. Tyto vlastnosti můžeme charakteristikám nastavit pomocí speciálních příznaků `ESP_GATT_CHAR_PROP_BIT_WRITE` a `ESP_GATT_CHAR_PROP_ BIT_READ`.

Dalšími charakteristikami budou zápis písmen a přerušení vysílání. Do těchto charakteristik však může klient pouze zapisovat. Čtení charakteristik je prováděno knihovní funkcí `esp_ble_gatts_get_attr_value` a odeslání odpovědi klientovi pomocí `esp_ble_gatts_send_response`.

Inicializační funkce má navíc ještě dva parametry, v rámci kterých lze specifikovat funkce, které mají být volány

[3]

při přidání charakteristiky (toho je využito při obnově původní hlasitosti z nevolatilní paměti) a při události zápisu do charakteristiky. Tyto parametry umožňují efektivně napojit funkcionalitu bluetooth na zbytek programu.

Jelikož je projekt koncipován modulárně, funkcionalita bluetooth je koncentrována do modulu `ble_receiver`, který se zbytkem programu komunikuje skrze rozhraní v příslušném hlavičkovém souboru (součástí kterého je také zmíněná inicializační funkce).

Při příchodu události zapisující do charakteristiky zprávy jsou příchozí znaky uloženy po blocích (nyní pouze o velikosti 1 znak) do hlavní fronty `queue`, kterou poskytuje modul pro překlad znaků.

2.2 Překlad znaků

Překlad znaků je realizován modulem `translator`. Jeho stěžejní funkce `tranlate` ve vlastním procesu v nekonečné smyčce vyzvedává příchozí bloky znaků z fronty `queue` a ty pomocí překladové tabulky převádí na sekvenci znaků `.`, `-`, případně `/` (pro ukončení slova mezerou a věty tečkou). Pro větší efektivitu jsou znaky v tabulce seřazeny vzestupně dle rostoucího ASCII kódu, což umožňuje začít vyhledávání přibližně uprostřed tabulky. Vyhledávání poté probíhá sekvenčně příslušným směrem (záleží na ASCII kódu příchozího znaku a znaku přibližně uprostřed překladové tabulky). Na obou koncích tabulky jsou pro efektivní zastavení vyhledávání umístěny zarážky. Oproti triviálnímu sekvenčnímu vyhledávání je díky této modifikaci urychlen překlad přibližně o polovinu.

Přeložené sekvence výše uvedených znaků jsou poté převáděny na sekvence struktur `out_control` a ukládány do druhé fronty poskytované tímto modulem – `out_queue`. Z této fronty jsou jednotlivé struktury vyzvedávány funkcí pro ovládání výstupu.

2.3 Ovládání výstupu

Ovládání výstupních periferií (LED diody a bzučáku) je zodpovědností funkce `out_control_routine`. Tato funkce vyzvedává struktury `out_control` z fronty `out_queue`. Tyto struktury jsou tvořeny třemi čítači – čítačem pro ovládání LED, čítačem pro ovládání bzučáku a čítačem označující prázdné cykly. První dva čítače jsou při každém vyzvednutí struktury z fronty dekrementovány, pokud jsou větší než nula.

V reakci na jejich hodnotu jsou také nastaveny signály na pinech, k nímž jsou přepojeny periferie (viz `BUZZER_GPIO` a `LED_GPIO`). Pokud jsou tyto hodnoty nenulové dojde k přepnutí pinů na úroveň 1. V opačném případě je signál na pinech vypnut. Jestliže jsou oba dva čítače rovny nule, dochází k dekrementování čítače prázdných cyklů, dokud je jeho hodnota opět nenulová. Pokud je alespoň jeden z čítačů ve struktuře nenulový, funkce vrací strukturu zpět do čela fronty a dojde tak k jejímu obsloužení v rámci dalšího zavolání této funkce.

Pomocí počátečních hodnot čítačů ve strukturách můžeme snadno konfigurovat délku pípnutí nebo svitu LED diody pro daný symbol (`- . /`). Tuto počáteční hodnotu určuje funkce `translate` při překladu příchozího znaku. Pomocí čítače prázdných cyklů (`gap`) je implementována časové mezera mezi jednotlivými písmeny, aby nedošlo k jejich splynutí.

Protože musíme zajistit konstantní interval v němž bude `out_control_routine` volána, musíme pro tento účel použít vestavěný časovač, který vyvolává v námi definovaných intervalech přerušení, které je obslouženo právě touto funkcí.

2.4 Vysílač

Vysílač (klientská část programového řešení) je implementován pomocí rozhraní WebBluetooth, které umožňuje přistupovat k bluetooth rozhraní hostitelského zařízení. Funkce, které toto rozhraní poskytuje jsou zpravidla asynchronní, a proto jejich návratové hodnoty musíme zpracovávat v rámci zpětného volání v JS promisech případně pomocí klíčových slov async/await.

Rozhraní je vytvořeno pomocí jednoduché HTML stránky, která obsahuje některé interaktivní prvky, aby mohl uživatel snadno s přijímačem (ESP32) komunikovat. Style stránky je upraven pomocí CSS pro lepší uživatelský zážitek.

3 Výsledné řešení

Řešení projektu by mělo ve všech směrech vyhovovat požadavkům zadání. Byl vytvořen program v jazyce C pro platformu ESP32 s využitím ESP-IDF rámce, který umožňuje přijímat znaky prostřednictvím bezdrátové komunikace BLE. Tyto znaky poté přeloží a převede skrze morseovu abecedu na pípání připojeného bzučáku či svit led diody.

Uživatel může také regulovat hlasitost bzučáku nebo vypípávání zprávy zrušit. Hlasitost je navíc ukládána do perzistentní paměti platformy ESP32, takže mezi jednotlivými spuštěními je hlasitost uchovávána.

Uživatel s tímto zařízením komunikuje prostřednictvím HTML stránky využívají WebBluetooth (je zapotřebí používat prohlížeč, který WebBluetooth podporuje). Stránka obsahuje tlačítka pro připojení k zařízení, odpojení od zařízení, přerušení zprávy a jezdec pro nastavování hlasitosti (ten je zpětně aktualizován při každém úspěšném připojení k zařízení). Stránka také podporuje dva módy vysílání, buďto průběžné vysílání po jednotlivých písmenech, které uživatel mačká na klávesnici, (type-send mode) nebo odeslání celé zprávy (batch mode).

Závěr

Výsledné řešení by mělo plně odpovídat zadání a poskytovat některou funkcionalitu nad jeho rámec (přerušení zprávy, persistence hlasitosti apod.). V rámci řešení projektu jsem se seznámil s programováním platformy ESP32 skrze IDF framework a také s některými teoretickými fakty o fungování bluetooth.

Reference

[1] Barr, M.: Pulse width modulation. *Embedded Systems Programming*, ročník 14, č. 10, 2001: s. 103–104.

[2] Gomez, C.; Oller, J.; Paradells, J.: Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors*, ročník 12, č. 9, 2012: s. 11734–11753, ISSN 1424-8220, doi:10.3390/s120911734.
URL `https://www.mdpi.com/1424-8220/12/9/11734`

[3] Gupta, N. K.: *Inside Bluetooth low energy*. Artech House, 2016.

[4] Kevin, T.: Introduction to Bluetooth Low Energy - GAP. [online], 2014.
URL `https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap`

[5] Kevin, T.: Introduction to Bluetooth Low Energy - GATT. [online], 2014.
URL `https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt`