

Modul internetového rádia na bázi mikropočítačové vývojové desky ESP32

Daniel Janák

Bakalářská práce
2025



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Daniel Janák
Osobní číslo: A20367
Studijní program: B0613A140020 Softwarové inženýrství
Forma studia: Prezenční
Téma práce: Modul internetového rádia na bázi mikropočítačové vývojové desky ESP32
Téma práce anglicky: Internet Radio Module Based on the ESP32 Microcontroller Development Board

Zásady pro vypracování

- Zpracujte literární rešerši na dané téma.
- Provedte hardwarový návrh modulu internetového rádia na bázi mikropočítačové vývojové desky ESP32.
- Prakticky zrealizujte modul internetového rádia.
- Vytvořte programové vybavení modulu zajišťující základní funkcionality internetového rádia – správa stanic, synchronizace reálného času, funkce budíku.
- Ověřte správnou funkci modulu internetového rádia.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. CAMERON, Neil. *Electronics projects with the ESP8266 and ESP32: building web pages, applications, and Wifi enabled devices*. Berkeley, CA: Apress L. P., 2021, 1 online zdroj (703 stran). Technology in action. Dostupné z: doi:9781484263365
2. Espressif Systems. *ESP32: ESP-IDF Programming Guide*. Rev. 5.3.1. [online]. 2024 [cit. 2024-10-16]. Dostupné z: <https://www.espressif.com/>.
3. JURÁNEK, Antonín a Miroslav HRABOVSKÝ. *EAGLE pro začátečníky /: uživatelská a referenční příručka*.. 2. vydání. Praha: BEN – technická literatura, 2007, 191 s. ISBN 80-730-0213-2.
4. MANN, Burkhard. *C pro mikrokontroléry: ANSI-C, kompilátory C, spojovací programy – linkery, práce s ATMELE AVR a MSC-51, příklady programování v jazyce C, nástroje pro programování, tipy a triky*. Praha: BEN – technická literatura, 2003. μ C & praxe. ISBN 80-7300-077-6.
5. VLSI Solution. *VS1053b Datasheet: VS1053b – Ogg Vorbis/MP3/AAC/WMA/FLAC/ MIDI Audio Codec Circuit*, Rev. 1.32. [online]. 2024 [cit. 2024-10-16]. Dostupné z: <https://www.vlsi.fi>.

Vedoucí bakalářské práce:

Ing. Petr Dostálek, Ph.D.

Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **27. října 2024**

Termín odevzdání bakalářské práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Tato bakalářská práce se zabývá návrhem a realizací jednoduchého internetového rádia využívajícího mikrokontrolér ESP32 a audio dekodér VS1053. Cílem bylo vytvořit zařízení schopné připojení se k Wi-Fi, přehrávání online streamů a zobrazování informací o aktuálně přehrávané stanici na LCD displeji. Ovládání probíhá pomocí tlačítek a součástí je i funkce budíku spouštějícího se ve stanovený čas.

V teoretické části jsou popsány principy streamování audia, používané formáty a síťové protokoly, stejně jako architektura a technické parametry použitých komponentů. Praktická část se věnuje návrhu zapojení, sestavení zařízení a vývoji firmware pomocí PlatformIO ve vývojovém prostředí Visual Studio Code. Zařízení bylo úspěšně otestováno a je vhodné pro technické nadšence či začátečníky se zájmem o embedded systémy.

Klíčová slova: internetové rádio, mikrokontrolér, ESP32, VS1053

ABSTRACT

This bachelor thesis focuses on the design and implementation of a simple internet radio using the ESP32 microcontroller and the VS1053 audio decoder. The aim was to create a device capable of connecting to Wi-Fi, playing online streams, and displaying information about the currently playing station on LCD screen. The system is controlled via buttons and includes an alarm function that activates at a specified time.

The theoretical part describes the principles of audio streaming, the formats and network protocols used, as well as the architecture and technical specifications of the selected components. The practical part focuses on circuit design, device assembly and firmware development using PlatformIO in the Visual Studio Code development environment. The device was successfully tested and is suitable for technical enthusiasts or beginners interested in embedded systems.

Keywords: internet radio, microcontroller, ESP32, VS1053

Rád bych na tomto místě poděkoval všem, kteří mě podporovali během zpracování této bakalářské práce.

V první řadě bych chtěl velmi poděkovat panu Ing. Petru Dostálkovi, Ph. D. za jeho velkou podporu ve všech fázích této práce, za jeho odborné vedení a za trpělivý dohled nad celým pracovním procesem. Jeho rady a připomínky mě významně nasměrovaly a motivovaly k dokončení projektu.

Mé poděkování patří také mé přítelkyni a mé mamce za jejich ochotu a čas, který věnovaly kontrole a úpravě gramatické stránky textu v mé práci. Jejich pomoc mi velmi usnadnila finální úpravy.

V neposlední řadě bych rád poděkoval i paním sekretářkám ze studijního oddělení FAI za jejich neustálou trpělivost a vstřícnost při zodpovídání všech mých dotazů, které jsem jim během studia zasílal.

Na konec bych to jenom zakončil větou jak by řekli mladí: „ej bro, nešlo nic, ale dali jsme to“. Děkuji.

OBSAH

ÚVOD.....	11
TEORETICKÁ ČÁST.....	12
1 INTERNETOVÉ RÁDIO	13
1.1 STREAMING AUDIA PŘES INTERNET	14
1.1.1 Streamování audia přes HTTP a HTTPS	14
1.1.2 HTTP Live Streaming (HLS).....	15
1.1.3 Icecast.....	17
1.1.4 Shoutcast	18
1.2 FORMÁTY AUDIO STREAMŮ	19
1.2.1 MP3	20
1.2.2 AAC	21
1.2.3 Ogg Vorbis	22
1.2.4 Podpora formátů ve VS1053	24
1.3 SYNCHRONIZACE REÁLNÉHO ČASU ZAŘÍZENÍ	24
1.3.1 Význam přesného času pro funkce rádia	25
1.3.2 Síťová synchronizace času (NTP).....	25
1.3.3 Hardwarový RTC modul.....	27
2 MIKROKONTROLÉRY A MODULY ESP32	30
2.1 POPULÁRNÍ MODELY ESP32.....	30
2.2 ESP32-WROOM-32 ARCHITEKTURA A TECHNICKÉ PARAMETRY	32
2.2.1 Paměť	32
2.2.2 Bezdrátová konektivita.....	33
2.2.3 Periferie a rozhraní	33
2.2.4 Výkon a energetická náročnost	33
2.2.5 Bezpečnost	34
2.3 VÝVOJOVÉ DESKY S ESP32.....	34
2.3.1 Espressif ESP32-DevKitC	35
2.3.2 NodeMCU-32S	35
2.3.3 Adafruit HUZZAH32 Feather.....	36
2.3.4 SparkFun ESP32 Thing Plus.....	36
2.3.5 WEMOS/LOLIN D32, D32 Pro	36
2.3.6 M5Stack Core2.....	36
2.4 MOŽNOSTI PROGRAMOVÁNÍ ESP32	37
2.4.1 Další možnosti programování	38
2.5 PŘÍKLADY REÁLNÉHO POUŽITÍ ESP32	39
2.6 SROVNÁNÍ ESP32 S JINÝMI MIKROKONTROLÉRY	40
2.6.1 ESP32 vs STM32	40
2.6.2 ESP32 vs Arduino	41
2.6.3 ESP32 vs Raspberry Pi Pico	42

3	AUDIO MODUL VS1053	43
3.1	FUNKČNÍ BLOKY VS1053 (DEKODÉR, DSP, I/O)	43
3.2	KOMUNIKAČNÍ PROTOKOL MEZI VS1053 A ESP32.....	45
3.3	POROVNÁNÍ S JINÝMI DEKODÉRY/MODULY	47
4	NÁVRH A VÝVOJOVÉ PROSTŘEDÍ.....	49
4.1	EAGLE – NÁVRH PLOŠNÉHO SPOJE	49
4.2	VÝVOJ FIRMWARE VE VISUAL STUDIO CODE	50
4.3	PLATFORMIO.....	51
4.4	UŽIVATELSKÉ ROZHRANÍ ZAŘÍZENÍ	52
	PRAKTICKÁ ČÁST.....	55
5	HARDWAROVÝ NÁVRH MODULU	56
5.1	VÝBĚR A ZAPOJENÍ KOMPONENTŮ	57
5.2	NÁVRH ZAPOJENÍ.....	58
6	REALIZACE MODULU	62
7	PROGRAMOVÉ VYBAVENÍ MODULU	67
7.1	VÝVOJOVÉ PROSTŘEDÍ A POUŽITÉ KNIHOVNY	67
7.2	ARCHITEKTURA PROGRAMU	67
7.3	FUNKCE A JEJICH IMPLEMENTACE.....	68
7.3.1	Zpracování vstupu z tlačítek.....	68
7.3.2	Spuštění přehrávání internetového rádia	69
7.3.3	Zpracování audio streamu	70
7.3.4	Funkce budíku	71
8	OVĚŘENÍ A TESTOVÁNÍ	73
8.1	PRAKTICKÉ OVĚŘENÍ FUNKCIONALITY.....	73
8.2	ZHODNOCENÍ FUNKČNOSTI A NEDOSTATKŮ	74
8.3	MOŽNOSTI DALŠÍHO ROZŠÍŘENÍ	74
	ZÁVĚR.....	76
	SEZNAM POUŽITÉ LITERATURY	77
	SEZNAM OBRÁZKŮ	91
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	92
	SEZNAM PŘÍLOH.....	97

ÚVOD

V posledních desetiletích prochází oblast multimédií zásadními změnami. Tradiční vysílání (AM/FM rádio) je postupně nahrazováno moderními metodami šířenými přes internet. Tento trend je patrný jak u komerčních stanic, tak u jednotlivců, kteří mohou snadno provozovat vlastní online vysílání. Internetová rádia díky flexibilitě, široké nabídce a možnosti personalizace obsahu mění způsob, jakým lidé konzumují audio obsah.

Tato bakalářská práce vznikla jako reakce na tento vývoj zároveň z mého osobního zájmu o oblast embedded systémů a spojení hardwaru se softwarovým řešením. Mikrokontroléry jako ESP32 dnes umožňují vytvářet praktická zařízení v domácích podmínkách, včetně vlastního internetového rádia. Tento projekt spojuje moderní technologie s cílem vytvořit funkční a rozšiřitelný celek.

Cílem práce bude navrhnout a realizovat jednoduché internetové rádio, které se připojí k Wi-Fi síti, načte a přehraje online stream, zobrazí informace o přehrávané stanici a umožní základní ovládání pomocí tlačítek a LCD displeje. Zařízení bude přístupné i pro technické začátečníky, kteří by se chtěli naučit něco nového nebo by chtěli zařízení dále rozvíjet. Součástí implementace bude i budík, který v nastavený čas přehraje uložený zvuk.

Práce bude rozdělena do teoretické a praktické části. V teoretické části budou popsány technologie, na kterých projekt stojí. To jsou streamování audia, přenosové protokoly, audio formáty a parametry použitých mikrokontrolérů a audio dekodérů. Praktická část se zaměří na návrh zařízení, vývoj firmware a otestování výsledného řešení.

Výsledný modul by měl sloužit nejen jako funkční internetové rádio, ale také jako výuková pomůcka, která by mohla pomoci dalším zájemcům pochopit principy fungování mikrokontrolérů, přenosu audia a práce s programovacím prostředím. V době ústupu tradičního rádiového vysílání a rostoucí digitalizace médií bude mít podobný projekt své opodstatnění i význam, ať už z hlediska technického poznání nebo praktické využitelnosti.

Také prohlašuji, že při tvorbě této práce jsem použil nástroj generativního modelu AI ChatGPT (verze GPT-4o) dostupný na adrese <https://chat.openai.com>. Tento nástroj byl využit za účelem vyhledávání odborných zdrojů a sepisování krátké rešerše o daných zdrojích. Při používání nástroje jsem si byl plně vědom jeho možných chyb a prováděl jsem kontrolu a revizi veškerého vygenerovaného obsahu, za který přebírám plnou zodpovědnost.

TEORETICKÁ ČÁST

1 INTERNETOVÉ RÁDIO

Internetové rádio označuje technologii přenosu audio signálu (zvuku – např. rozhlasového vysílání či hudby) prostřednictvím internetu formou datového streamu. První pokusy o streamování audia se objevily již v 90. letech 20. století, kdy byla v roce 1995 představena technologie RealAudio. Formát RealAudio umožnil přehrávat zvuk v reálném čase během stahování (streaming) a díky kompresi nabídl přenos audia v kvalitě srovnatelné nejprve s AM rádiem a později téměř s FM rádiem i na pomalých telefonních linkách o rychlosti 14-56 kb/s. Tato technologie tak poprvé umožnila živé internetové vysílání zvuku širokému publiku. Zvuk byl na straně vysílače digitalizován a komprimován, odeslán po internetu a na straně posluchače dekodován a přehráván téměř v reálném čase. [1]

Od konce 90. let se začaly prosazovat nové standardy a služby streamingu audia. V roce 1998 uvedla společnost Nullsoft platformu Shoutcast, která místo proprietárních kodeků RealAudio využívala komprimaci do formátu MP3. Díky použití MP3 formátu s vysokou mírou komprese a přitom kvalitním zvukem a rostoucí dostupností širšího internetového připojení bylo možné dosáhnout výrazně lepší kvality online audia než dříve. Shoutcast tak fakticky odstartoval éru internetových rádií streamovaných v MP3 a rychle si získal popularitu. Následující rok (1999) vznikl open-source projekt Icecast jako otevřená alternativa k proprietárním streamovacím technologiím (RealAudio, Shoutcast apod.). Icecast umožnil vysílat audio po internetu bez licenčních poplatků a s podporou otevřených formátů (zejména Ogg Vorbis) a od verze 2 (2004) přinesl lepší podporu metadat a kompatibilitu s klienty Shoutcast. [2]

Význam streamovaného audia od 90. let neustále rostl. Oproti klasickému rozhlasu odstranilo internetové rádio geografická omezení, tedy vysílatel může přes internet zasáhnout globální publikum, což dalo rádiu „nový nádech“ a umožnilo oslovit posluchače po celém světě. Zároveň dramaticky klesla vstupní bariéra pro nové vysílatele, neboť odpadla nutnost vlastnit drahé vysílače či přidělené frekvence. K provozování rádia stačí server s připojením k internetu, software a licence k obsahu. Díky snadné dostupnosti internetu a nízkým nákladům se tak demokratizovalo prostředí rozhlasového vysílání. Prakticky kdokoliv s vášní pro obsah může spustit vlastní online rádio a oslovit publikum. Internetová rádia dnes nabízejí obrovskou žánrovou pestrost, ať se už bavíme o klasickém poslouchání muziky, zprávy, specifické diskuze, podcasty atd.. Takováto rozmanitost by v prostoru klasických FM frekvencí nebyla možná. Tradiční rozhlasové stanice rovněž využívají internet k simulcastu svého vysílání a k oslovení posluchačů mimo dosah

pozemního vysílače. Streamované audio se stalo nedílnou součástí moderního mediálního prostoru. Z pohledu rozhlasového vysílání internetové rádio představuje zásadní přínos, rozšiřuje dosah rádia globálně, diverzifikuje obsahovou nabídku a poskytuje posluchačům možnost volby kdykoliv a kdekoliv poslouchat libovolný obsah. [3]

1.1 Streaming audia přes internet

Přenos audia přes internet je realizován formou streamingu, kdy je audio obsah kontinuálně odesílán ze serveru a přijímán (a přehráván) klientem téměř v reálném čase. Existuje několik protokolů a technik, jak audio stream po internetu přenášet. Tato kapitola popisuje hlavní používané metody, jako přenos přes HTTP/HTTPS, streamování pomocí HLS a technologie Icecast/Shoutcast. Včetně popsání principu fungování, jejich výhod a nevýhod, typické použití a posouzení vhodnosti pro platformu ESP32 jako klientské zařízení. [2; 4]

1.1.1 Streamování audia přes HTTP a HTTPS

Protokol HTTP (Hypertext Transfer Protocol) a jeho zabezpečená varianta HTTPS jsou široce využívány pro distribuci multimediálního obsahu, včetně audia, přes internet. V kontextu streamování audia funguje HTTP jako transportní mechanismus, který umožňuje klientovi požadovat a přijímat audio data ve formě kontinuálního datového toku. Na rozdíl od tradičního stahování souborů, kde je obsah nejprve kompletně stažen a poté přehrán, umožňuje HTTP streamování přehrávání audia již během jeho přenosu. To zajišťuje plynulý poslech bez nutnosti čekání na stažení celého souboru. [5; 6]

Při využití HTTPS je komunikace mezi klientem a serverem šifrována pomocí protokolu TLS (Transport Layer Security), což zajišťuje důvěrnost a integritu přenášených dat. To je obzvláště důležité při přenosu citlivého obsahu nebo při ochraně proti neoprávněnému přístupu. [5; 6]

Streamování přes HTTP/HTTPS přináší své výhody. Jsou to standardní protokoly podporované většinou webových serverů a klientských zařízení, což usnadňuje implementaci a zajišťuje širokou dostupnost streamovaného obsahu. Díky široké známosti těchto protokolů je jejich implementace relativně jednoduchá, což je výhodné zejména pro vývojáře a menší projekty. Také díky využití standardních portů (80 pro HTTP a 443 pro HTTPS) je komunikace snadno prostupná skrze firewally a NAT zařízení, což minimalizuje potřebu speciální konfigurace sítě. [5; 6]

Nevýhodou HTTP/HTTPS je například omezená podpora pro adaptivní streamování. Standardní HTTP neumožňuje dynamickou změnu kvality streamu v závislosti na aktuálních podmínkách sítě, což může vést k přerušení přehrávání nebo snížení kvality při kolísání šířky pásma. HTTP/HTTPS mají také vyšší latenci v porovnání se speciálně navrženými protokoly pro streamování, jako je RTSP. To je značnou nevýhodou pro aplikace vyžadující nízké zpoždění. Při použití HTTPS může šifrování a dešifrování dat představovat dodatečnou zátěž pro klientské zařízení, zejména pokud se jedná o zařízení s omezenými výpočetními prostředky, jako je například mikrokontrolér ESP32. [5; 6]

Mikrokontrolér ESP32 je vybaven vestavěnou podporou pro Wi-Fi a nabízí dostatečný výkon pro zpracování HTTP/HTTPS komunikace. Díky dostupným knihovnám, jako je například `esp_http_client` v rámci vývojového prostředí ESP-IDF, je implementace HTTP klienta na ESP32 relativně přímočará. Při streamování audia může ESP32 přijímat data přes HTTP/HTTPS a předávat je specializovanému audio dekodéru, jako je například VS1053, který se stará o dekódování a přehrávání audia. Jak bylo již zmíněno, při použití HTTPS je však třeba počítat s vyššími nároky na paměť a výpočetní výkon, zejména kvůli šifrování a dešifrování dat. ESP32 disponuje dostatečnými prostředky pro zvládnutí těchto operací, avšak je důležité optimalizovat velikost bufferů a správu paměti, aby nedošlo k přetížení systému. [5; 6; 7]

Závěrem lze tvrdit, že streamování audia přes HTTP a HTTPS představuje jednoduché a široce kompatibilní řešení pro distribuci audio obsahu přes internet. Díky své jednoduchosti a široké podpoře je vhodné pro mnoho aplikací, včetně implementace na mikrokontrolérech, jako je ESP32. Je však důležité zvážit specifické požadavky aplikace, zejména pokud jde o latenci, adaptivitu streamu a požadavky na systémové prostředky. Případně by bylo potřeba zvážit alternativní protokoly nebo optimalizaci pro dosažení požadovaného výkonu. [5; 6]

1.1.2 HTTP Live Streaming (HLS)

HLS je streamingový protokol vyvinutý firmou Apple původně pro streaming videa, který se však používá i pro audio. Princip HLS spočívá v rozdělení vysílaného média na malé části (tzv. segmenty) a jejich postupné načítání a přehrávání. Klient nejprve stáhne tzv. manifest (playlist), což je .m3u8 soubor obsahující odkazy na segmenty, následně postupně stahuje jednotlivé segmentové soubory a přehrává je za sebou. Každý segment obvykle reprezentuje několik sekund audio záznamu (typicky 2-10 s). Po stažení jednoho segmentu

se klient automaticky dotazuje na další podle playlistu, který se průběžně aktualizuje na serveru. HLS tedy nepoužívá jedno trvalé spojení, ale sérii krátkých HTTP požadavků na malé soubory. Obsah segmentů jsou nejčastěji MPEG-TS či fMP4 kontejnery s audio/video daty. [6; 8; 9]

Výhody HLS jsou, že přináší robustnost a adaptabilitu. Díky segmentaci je stream odolnější vůči výpadkům, tedy ztráta jednoho segmentu neznamena ukončení celého přenosu. Klient může také snadno adaptivně přepínat mezi různými kvalitami streamu (bitrate) podle dostupné rychlosti sítě. Manifest může obsahovat více variant a přehrávač si zvolí například nižší bitrate, pokud detekuje pomalé spojení. To umožňuje tzv. adaptivní streaming, podobně jako to umožňuje konkurenční protokol MPEG-DASH. Další výhodou je využití běžného HTTP, HLS segmenty jsou cache-ovatelné na CDN serverech, což usnadňuje škálování pro velký počet posluchačů. HLS je také podporován v mnoha moderních zařízeních a prohlížečích (zejména u Apple ekosystému je to nativní standard). Pro poskytovatele obsahu nabízí HLS také možnost šifrování přenášeného obsahu. [6]

Za hlavní nevýhodu HLS se považuje jeho latence a režie spojení. Kvůli segmentaci dochází k určitému zpoždění, kdy přehrávač čeká, než nasbírá pár segmentů dopředu. Běžné nastavení (např. 4 segmenty po 6 s) vede k latenci okolo 20-30 sekund. I při zkrácení segmentů na 2 s a omezení bufferu na 3 segmenty se latence typicky pohybuje kolem 6-10 sekund. To je výrazně více než u nepřerušovaného HTTP streamu, kde latence může být jen pár sekund (dána velikostí bufferu). Pro živé rozhlasové vysílání s důrazem na nízké zpoždění může být HLS nevhodné. Další nevýhodou je komplexita na straně klienta. Je potřeba implementovat logiku stahování playlistu, parsování .m3u8 souboru, výběru segmentů a jejich skládání. To vyžaduje více paměti (nutné ukládat segmenty a jejich části) a výpočetního výkonu. Pro méně výkonná zařízení by mohl být problém i značná režie, každých pár sekund nové HTTP spojení, HTTP hlavičky atd.. [6; 8]

Pro malý mikropočítač jako ESP32 představuje HLS poměrně náročný protokol. Neexistuje jednoduchá vestavěná knihovna, bylo by nutné implementovat načítání a parsování playlistu m3u8 a následné postupné stahování segmentů. To zvyšuje komplexitu kódu i paměťové nároky. ESP32 by muselo uchovávat alespoň jeden segment (velikost desítky kB) v RAM, než jej pošle do dekodéru. Navíc VS1053 dekodér neumí přímo formát TS nebo MP4, takže by to vyžadovalo z proudu segmentu extrahovat samotná komprimovaná audio data a ta teprve posílat do VS1053. Taková funkcionality výrazně přesahuje možnosti jednoduchých existujících knihoven. Z těchto důvodů není HLS pro

běžné projekty na ESP32 příliš vhodné. HLS by připadal v úvahu v případě, že bychom chtěli využít adaptivní bitrate a byli ochotní implementovat potřebnou podporu. [10; 11]

1.1.3 Icecast

Icecast je open-source streamovací server pro audio vyvíjený nadací Xiph.org. Funguje tak, že přijímá od zdrojového klienta (tzv. source klient) audio stream a ten pak nabízí posluchačům jako HTTP stream. Icecast od verze 2 podporuje více formátů včetně Ogg Vorbis, Ogg Opus, WebM, MP3 a ACC. Pro klienty (posluchače) se Icecast tváří velmi podobně jako webový server. Každá stanice má vlastní URL (tzv. mountpoint) a posluchač si ji naladí otevřením této adresy ve svém přehrávači. Icecast je navržen tak, aby byl kompatibilní s protokolem Shoutcast. To znamená, že umí přijímat stream od Shoutcast source klienta a posílat metadata v Shoutcast formátu, díky čemuž jej podporuje široká škála přehrávačů. [2; 12]

Icecast je svobodný software a nevyžaduje žádné licenční poplatky za nasazení. Mezi další jeho výhody patří, že je velmi flexibilní. Umožňuje jedním serverem obsluhovat více nezávislých streamů (mountpointů), nastavit přístupová omezení, autentizaci posluchačů, přesměrování na jiné hosty, řetězení serverů apod.. Podporuje moderní formáty s účinnou kompresí (např. Opus, AAC) i otevřené formáty (Ogg Vorbis). Díky podpoře Ogg může poskytovat i tzv. Icecast metadata přímo v datovém toku (u Ogg streamu se posílají názvy skladeb v rámci bitstreamu), a od verze 2.4 zlepšil i kompatibilitu Shoutcast metadat pro MP3/AAC streamy. Icecast má aktivní komunitu, je průběžně aktualizován a široce používán v internetovém vysílání. Mnoho nezávislých rádií i velkých stanic používá Icecast server pro své online streamy. Z pohledu kvality streamu nemá Icecast žádné vlastní omezení. Je schopený zpracovat tolik současných uživatelů a v takové kvalitě, jak umožňují zdroje serveru. Díky podpoře Opusu a moderních kodeků umožňuje vysílat ve vysoké kvalitě zvuku s nízkým bitrate (např. Opus 64 kb/s může znít lépe než MP3 128 kb/s). [2; 12; 13; 14]

Nevýhoda Icecast, tak jako každé self-hosted technologie je to, že vyžaduje správu serveru. Pro běh je nutné mít vlastní server s dostatečným uploadem pro všechny posluchače nebo využít hosting. Samotný protokol vychází z HTTP, takže nepřináší adaptivní bitrate. Oproti Shoutcastu postrádá jednotný globální katalog stanic. Shoutcast má oficiální adresář, kam se stanice registrují, zatímco Icecast komunita toto řeší různě (existují veřejné katalogy, ale ne jednotný od výrobce). Pro koncové uživatele to ale není

zásadní problém, stanice stejně často zveřejňují přímo odkazy na stream. Výkonově je Icecast velmi efektivní a zvládá i tisíce posluchačů, tam nevýhoda není. Pro uživatele ESP32 není žádná specifická nevýhoda, klient jen přijímá standardní stream. [12; 14]

ESP32 jako klient si s Icecast streamy poradí výborně. Jelikož Icecast používá standardní HTTP stream, stačí na ESP32 otevřít příslušnou URL a začít číst data. Mnoho projektů internetových rádií s ESP32 bylo navrženo právě pro příjem Icecast/Shoutcast streamů. VS1053 dekodér navíc přímo podporuje formáty, které se s Icecastem nejčastěji pojí (MP3, AAC, Ogg Vorbis), takže ESP32 může data přímo přeposílat dekodéru bez nutnosti je zpracovávat. Z tohoto důvodu lze Icecast považovat za jeden z ideálních zdrojů audia pro ESP32 rádio. Navíc umožňuje využití otevřených formátů, takže odpadá starost s licencemi. Celkově vzato, pokud je k dispozici stream přes Icecast, není z pohledu ESP32 žádné výrazné omezení nebo komplikace. [2; 10; 11]

1.1.4 Shoutcast

Shoutcast je proprietární protokol a software pro internetové rádio, původně vyvinutý firmou Nullsoft v roce 1998. Historicky sehrál klíčovou roli, jelikož umožnil domácím tvůrcům i profesionálním stanicím snadno vysílat audio ve formátu MP3 po internetu, čímž konkuroval tehdejší dominanci RealAudio. Shoutcast server (známý jako DNAS) přijímá od zdrojového programu audio (zpočátku pouze MP3, později AAC) a distribuuje jej posluchačům. Protokol Shoutcast je velmi podobný HTTP odpovědi. Rozdílem je jen speciální označení „ICY“ místo „HTTP“ v hlavičkách a přidání Shoutcast specifických metadat prostřednictvím tzv. „ICY headers“. Díky této podobnosti jsou Shoutcast streamy kompatibilní s většinou přehrávačů podporujících HTTP streaming. [2; 15]

V době vzniku přinesl Shoutcast zásadní výhodu, využil populárního formátu MP3 místo proprietárních kodeků RealAudia. MP3 dekodéry byly tehdy již rozšířené, takže posluchači mohli snadno poslouchat (např. ve Winampu). Shoutcast rovněž poskytuje globální adresář stanic (shoutcast.com), kde se vysílatelé mohou registrovat, což zvyšuje objevitelnost jejich rádia. Tento katalog byl historicky velmi populární a integrace s Winampem znamenala pro stanice výrazné rozšíření zásahu publika. Pro vysílatele byl Shoutcast relativně nenáročný, software DNAS byl k dispozici zdarma pro nekomerční použití a konfigurace byla jednoduchá. V současné verzi (Shoutcast 2) přibyla podpora moderních formátů (HE-AAC) a některé pokročilé funkce. Shoutcast protokol je optimalizovaný pro velký počet posluchačů a v testech vykazuje velmi nízkou režii CPU i

při stovkách až tisících klientů. Pro posluchače je výhodou široká kompatibilita, jelikož prakticky každý audio přehrávač umí přehrát Shoutcast stream. [2; 15]

Nevýhodou Shoutcastu je jeho uzavřenost a v posledních letech i obchodní model. Samotný protokol sice klienti mohou implementovat, ale serverový software DNAS je proprietární. Novější verze Shoutcast serveru (od v2.5+) zavedly pro komerční stanice licenční poplatky nebo povinné vkládání reklam (provozovatel Shoutcast.com monetizuje platformu). To vedlo řadu internetových rádií k odklonu k Icecastu. Technicky Shoutcast nepodporuje některé otevřené formáty (např. Ogg Vorbis). Metadata jsou omezena, posílají se pouze jako textové informace o skladbě každých „n“ bytů dat, což je méně pružné než u Icecast Ogg streamu. Z pohledu kvality zvuku Shoutcast žádné omezení nemá, kvalita závisí na zvoleném kodeku a bitrate a zdrojích serveru. Nicméně výběr kodeků je omezen na MP3 a AAC, nelze využít modernější Opus. [2; 15; 16]

Pro klienta na platformě ESP32 nepředstavuje příjem Shoutcastu žádný problém. Z pohledu implementace jde opět o HTTP spojení s průběžnými daty. Je jen potřeba ošetřit specifikum metadat. Shoutcast posílá metadata v tzv. ICY protokolu každých „n“ bytů audio dat. Výhodou je, že dostupné knihovny pro Arduino/ESP32 s tím počítají, nebo lze metadata vypnout (server je neposílá, pokud o ně klient nepožádá v požadavku). Po dekódovací stránce VS1053 opět pokrývá formáty, které Shoutcast používá (MP3, AAC). Celkově je tedy Shoutcast stream pro ESP32 stejně dobře použitelný jako Icecast. Vývojář si jen musí dát pozor na správnou komunikaci, tedy že Shoutcast server při navazování spojení očekává specifickou hlavičku „icy-metadata: 1“ od klienta pokud chce přijímat i metadata. [10; 11; 15]

1.2 Formáty audio streamů

Pro internetové rádio jsou klíčové podporované formáty komprese audia. Ty určují kvalitu zvuku při daném datovém toku (bitrate), náročnost dekódování i případné licenční požadavky. V projektu s ESP32 a dekodérem VS1053 přicházejí v úvahu zejména formáty MP3, AAC a Ogg Vorbis, neboť právě tyto tři patří k nejrozšířenějším v internetovém vysílání a VS1053 je umí dekódovat. Následuje popis jednotlivých formátů, jejich technických vlastností a licenčních aspektů. Na závěr je uvedena podpora těchto formátů v použitém dekodéru VS1053.

1.2.1 MP3

MP3 (MPEG-2 Audio Layer III) je pravděpodobně nejznámější audio formát vůbec, který v devadesátých letech znamenal revoluci v distribuci digitální formy muziky. MP3 je ztrátový kompresní formát, který využívá psychoakustický model k odstranění složek zvuku, které lidské ucho méně postřehne a dosahuje tak výrazného zmenšení datového objemu. Typicky MP3 dokáže zmenšit audio soubor v CD kvalitě (44,1 kHz, 16 bit stereo) kolem 11x při zachování přijatelné kvality. Například 1 minuta nekomprimovaného audia (~10 MB) se v MP3 ve velmi dobré kvalitě vejde do cca 1 MB. [17; 18; 19]

MP3 kóduje zvuk po malých blocích (rámcích) o délce 1152 vzorků. Používá MDCT transformaci s hybridním filtrováním, dynamické přidělování bitů frekvenčním koeficientům a tzv. bit reservoir (umožňuje vypůjčit bitrate mezi rámci pro složitější úseky). Kvalita MP3 výrazně závisí na zvoleném datovém toku. Tradičně se uvádí, že „near-CD“ kvality je dosaženo u 128 kb/s stereo, ale to je spíše minimum. Při 128 kb/s jsou slyšet artefakty komprese. Dnešní standard pro velmi dobrou kvalitu je 192 kb/s a více. Empiricky platí, že pro běžný poslech je MP3 od ~192 kb/s prakticky transparentní (tj. nerozeznatelný od originálu), zatímco 128 kb/s může způsobovat slyšitelné ztráty. Modernější kodeky jako AAC dokáží dosáhnout srovnatelné kvality při nižším bitrate než MP3 (např. AAC). Latence u MP3 streamu není velká, dekodér potřebuje vyrovnávací paměť na několik rámců, typicky v řádu desítek milisekund, což je zanedbatelné ve srovnání se sítí. MP3 podporuje vzorkovací frekvence 32, 44,1 a 48 kHz (MPEG-1) plus poloviční a čtvrtinové vzorkovací frekvence při MPEG-2 a MPEG 2.5, až 2 kanály (stereo) a pro vícecestné audio není určen. [17; 18; 20]

Formát MP3 byl dlouho zatížen patenty. Technologie vyvinuté Fraunhoferovým institutem a dalšími byly patentovány v mnoha zemích. Kdokoliv implementoval enkodér nebo dekodér MP3, měl oficiálně platit licenční poplatky (zpravovala je společnost Technicolor a Fraunhofer). To komplikovalo open-source implementace. Situace se však změnila v dubnu roku 2017, kdy vypršel poslední klíčový patent a licenční program MP3 byl ukončen. Dnes je tedy MP3 formát považován za volně použitelný poté co Fraunhofer oficiálně ukončil vybírání poplatků. Je třeba podotknout, že stále mohou existovat okrajové patenty (např. určité optimalizace dekodérů), ale v praxi je možné MP3 libovolně používat bez licenčních poplatků. To je důvod, proč v posledních letech např. distribuce Linuxu mohly začít přímo zahrnovat MP3 podporu. Z hlediska internetového rádia tedy MP3 nepřináší licenční zátěž. [21; 22]

Lze shrnout, že MP3 je nyní univerzálně podporovaný formát, jelikož prakticky všechny přehrávače, zařízení a prohlížeče jej umí přehrát. Díky tomu je často „jistotou“ kompatibility, zejména pro starší nebo jednodušší zařízení. Na druhou stranu, z hlediska efektivity komprese jde o více než 25 let starý formát, takže novější kodeky (AAC, Vorbis, Opus) dokáží poskytnout lepší kvalitu zvuku při stejném či nižším bitrate. Pro internetová rádia je MP3 stále velmi populární volbou, mnoho stanic vysílá v MP3 128 kb/s pro zajištění kompatibility. S ohledem na patentovou situaci již není třeba se vyhýbat MP3 kvůli licencím, takže jeho používání je bez omezení i v open-source projektech. [17; 23]

1.2.2 AAC

AAC (Advanced Audio Coding) je ztrátový audio formát vyvinutý jako nástupce MP3 v rámci standardů MPEG-2 a MPEG-4. Původní MPEG-2 AAC vzniklo v roce 1997, MPEG-4 AAC pak přineslo další vylepšení kolem roku 1999. AAC využívá pokročilejší algoritmy komprese než MP3, má efektivnější psychoakustický model, podporuje více kanálů, větší rozsah vzorkovacích frekvencí a další nástroje (např. tzv. temporal noise shaping). Existuje několik profilů AAC: nejrozšířenější AAC-LC (Low Complexity) pro vysokou kvalitu, dále HE-AAC (High Efficiency) verze v1 a v2 pro nízké bitrate (přidávají SBR a Parametric Stereo technologii) vhodné např. pro streaming na mobilních sítích a také AAC-LD/ELD (Low Delay) pro nízkolatenční aplikace. [24; 25]

AAC typicky pracuje s bloky 1024 vzorků, případně 960 vzorků u HE-AAC. Má lepší frekvenční rozlišení než MP3 a volitelně může použít delší či kratší bloky podle signálu. Ve srovnání s MP3 dosahuje AAC vyšší účinnosti, obecně AAC poskytuje lepší zvukovou kvalitu než MP3 při stejném datovém toku, zejména u nižších bitrate. Například AAC-LC na 128 kb/s obvykle převyšuje kvalitou MP3 128 kb/s a blíží se k 192 kb/s MP3. To dělá z AAC výhodný formát pro internetové vysílání, protože při polovičním datovém toku je schopný dosáhnout podobné kvality jako MP3, což šetří přenosové pásmo. HE-AAC v2 je optimalizován pro velmi nízké bitrate kolem 32-64 kb/s. V těchto podmínkách dramaticky překonává kvalitu MP3, a proto se stal populárním pro mobilní streamy. AAC podporuje více zvukových kanálů (až 48) a je vhodný i pro prostorový zvuk nebo vícestopé nahrávky. Z hlediska latence je standardní AAC-LC podobně náročné jako MP3, u HE-AAC může dojít k většímu zpoždění dekódování kvůli zpracování SBR, ale stále se jedná maximálně o stovky ms. AAC vyžaduje vyšší výpočetní výkon na enkodéru i dekodéru, ale moderní HW i SW tyto nároky bez problému zvládají. [17; 18; 24; 25; 26; 27]

AAC je zatížen patenty řady společností (Fraunhofer, Dolby, AT&T a další). Patentové pooly spravuje například organizace Via Licensing. Na rozdíl od MP3 ještě všechny patenty AAC nevypršely, proto implementace AAC enkodéru/dekodéru oficiálně vyžaduje licenci. Nicméně pro koncové uživatele a vysílatele platí, že streamování a distribuce obsahu v AAC není zpoplatněna, licenční poplatky se týkají výrobců softwaru a hardwaru. V praxi to znamená, že např. výrobce čipů (jako je VS1053) si musel obstarat licenci na AAC dekódování, aby mohl prodat čip podporující AAC. Běžný uživatel nebo autor internetového rádia nemusí platit za to, že vysílá nebo přehrává AAC stream. Z pohledu open-source je situace složitější. Existují otevřené implementace (např. Fraunhofer FDK AAC pod licencí kompatibilní s Androidem, nebo ffmpeg má volitelný AAC), ale kvůli patentům je např. v Linuxových distribucích AAC enkodér někdy zvlášť (non-free repozitář). Avšak i přes tyto licenční omezení je dekódování AAC dnes natolik rozšířené, že jej podporuje i běžný hardware jako mobilní čipy. Každopádně na platformě ESP32 se zpravidla AAC dekóduje přímo pomocí čipu VS1053, který licenci má. [18; 28; 29; 30; 31]

Obecně lze AAC považovat za modernější náhradu MP3 s lepší účinností. Proto ho upřednostňují streamingové služby a moderní platformy. Například Apple má AAC jako standard pro iPhone, iTunes a iPad a YouTube jej používá také. Pro internetová rádia je AAC výhodné, má-li cílové publikum novější zařízení či aplikace, které nebudou mít problém se zpracováním tohoto formátu. Nevýhodou může být u nejkonzervativnějších posluchačů či starších autorádií nedostatečná podpora (zatímco u MP3 by tento problém nebyl). Dnes je však AAC velmi rozšířen a jeho kompatibilita je vynikající napříč platformami (Windows, iOS, Android, YouTube a další). Internetová rádia často nabízí AAC stream jako alternativu k MP3, buď pro nižší bitrate nebo i pro hlavní vysoko-kvalitní stream. Z pohledu ESP32/VS1053 je AAC podporován, takže pokud je k dispozici AAC stream, lze jej využít. Celkově AAC představuje efektivní formát pro streaming. [11; 17; 27]

1.2.3 Ogg Vorbis

Ogg Vorbis je ztrátový audio formát, vyvinutý jako otevřená alternativa k patentovaným formátům typu MP3 a AAC. Vývoj vedla nadace Xiph.org (hlavní autor kodeku Christopher Montgomery) a první stabilní verze formátu byla vydána v roce 2000. „Ogg“ označuje kontejner (formát zapouzdření), zatímco „Vorbis“ je název samotného kompresního algoritmu pro audio. Vorbis je nabízen pod licencí BSD a byl navržen tak,

aby se vyhnul všem patentovaným technikám. Jedná se o zcela otevřený, neproprietární a bezlicenční (royalty-free) formát. To znamená, že kdokoliv může Vorbis používat či implementovat bez obav z porušení patentů. [32; 33]

Vorbis je svými schopnostmi obdobou AAC. Pracuje s MDCT transformací na blocích o velikosti standardně 1024 a 128 vzorků (tzv. long/short blocks pro různé komplexní signály). Má pokročilý psychoakustický model a umí efektivně kódovat stereo signál. Kvalitou při daném bitrate Vorbis překonává starší MP3 a je plně srovnatelný s AAC. Oproti AAC-LC byl Vorbis dlouho považován za přibližně rovnocenný, záleželo na enkodéru a testovaném materiálu. Dnes už AAC prošlo větším vývojem (HE-AAC apod.), zatímco Vorbis je ve své původní podobě, takže pro extrémně nízké bitrate je AAC vhodnějším kandidátem. Při vyšších bitrate 128-320 kb/s má však Vorbis srovnatelné výsledky v kvalitě. Vorbis podporuje širokou škálu vzorkovacích frekvencí (8-192 kHz) a 2-256 kanálů. Při kódování se typicky nevolí přímo bitrate, ale „kvalita“ a enkodér Vorbis dynamicky přizpůsobuje bitrate podle obsahu. Pro streaming lze ale nastavit i konstantní bitrate režim. Z hlediska výpočetní náročnosti je Vorbis dekodování o něco složitější než MP3, ale dobře optimalizované implementace umožňují běh i na slabých zařízeních. Vorbis nemá žádné umělé omezení kvality, lze jít až na 512 kb/s na 48 kHz stereo, čímž se blíží bezztrátové kvalitě. Ve streaming praxi se používají bitrate okolo 128 kb/s pro kvalitu blízkou CD. [32; 34; 35; 36; 37; 38]

Hlavním rysem Ogg Vorbisu je absence patentových omezení. Xiph.org deklaroval Vorbis jako patent-free od jeho počátku. V době jeho vzniku to byla revoluční věc, zatímco MP3 bylo zatíženo licencemi, Vorbis nabízel svobodnou alternativu. Toho využila řada open-source projektů (například hry a aplikace na Linuxu používaly Vorbis místo MP3). Použití Vorbisu tedy nevyžaduje žádné poplatky či povolení. Například firma Mozilla (Firefox jej implementoval nativně dříve než AAC) a ostatní populární prohlížeče také podporují Ogg Vorbis. V posledních letech Vorbis částečně ustoupil novějšímu formátu Opus (také od Xiph.org), který jej překonává při nižších bitrate. Nicméně Opus je mladší (standardizován 2012) a mnoho starších zařízení ho nepodporuje. [33; 39; 40; 41; 42; 43]

Vorbis představuje kvalitní a svobodný formát pro audio streaming. Z hlediska kvality vs bitrate je plně konkurence schopný vůči patentovaným formátům. Výhodou je, že komunita jej může využívat bez právních komplikací, což zvyšuje jeho popularitu, například přehrávače Windows Media Player a VLC jej podporují již v základu. Nevýhodou může být jeho menší rozšíření než populární MP3 formát. Dnes se ale situace

zlepšila a Vorbis přehraje většina softwarových přehrávačů i některé novější hardware zařízení. Pro internetové rádio je Vorbis atraktivní volbou, pokud se cílí na kvalitu a zároveň open-source řešení. V kontextu ESP32 je Vorbis podporován dekodérem VS1053, takže technicky není problém takový stream přijímat. [11; 35; 36]

1.2.4 Podpora formátů ve VS1053

Specializovaný dekodér VLSI VS1053B, použitý v tomto projektu, je multifunkční audio kodek, který podporuje řadu formátů. Konkrétně podle dokumentace umožňuje dekódovat MP3, AAC, Ogg Vorbis, WMA, MIDI, FLAC a nekomprimované WAV (PCM i ADPCM). To znamená, že všechny výše probírané formáty (MP3, AAC, Vorbis) jsou přímo hardwarově podporovány. VS1053 umí dokonce i moderní bezztrátový formát FLAC a starší formát WMA od Microsoftu. Pro účely internetového rádia jsou klíčové MP3/AAC/Vorbis, jejichž podpora ve VS1053 umožňuje přijímat jakýkoliv stream v těchto formátech a převádět jej na analogový zvuk. ESP32 samo o sobě tedy nemusí řešit dekódování, pouze zprostředkovává data z internetu do dekodéru. Licenční otázky za použití těchto formátů řeší výrobce čipu (VLSI Solution zajistilo licence AAC), takže koncový uživatel ani vývojář se tím nemusí zabývat. Omezení může nastat u formátu Opus, který VS1053 nepodporuje (nebyl v době vzniku čipu rozšířený). Nicméně většina stanic vysílá v podporovaných formátech. VS1053 tedy poskytuje širokou kompatibilitu a flexibilitu pro internetové rádio projekty, jelikož umožňuje přehrávat jak klasické MP3 streamy, tak modernější AAC i otevřené Ogg Vorbis streamy, a dokonce si poradí i s případnými FLAC streamy. [11]

1.3 Synchronizace reálného času zařízení

Internetové rádio na bázi ESP32 typicky není jen jednoduchý přehrávač streamu, ale může plnit i další funkce, pro které je důležitý přesný čas. Například mnoho implementací zahrnuje funkci budíku. Uživatel si může nastavit čas, kdy se má rádio samo zapnout či přehrát určitou stanici. Dále může jít o plánování záznamů pořadů, synchronizaci denního/nočního režimu displeje, případně logování přehrávaných skladeb s časovým razítkem. To vše vyžaduje, aby zařízení mělo k dispozici reálný čas (RTC – Real Time Clock) s dostatečnou přesností. Na rozdíl od osobních počítačů, které udržují čas i po jejich vypnutí, musí IoT zařízení jako ESP32 řešit čas autonomně, tedy při spuštění zjistit aktuální čas a dále jej udržovat. Tato kapitola rozebírá dvě hlavní metody zajištění přesného času, tedy synchronizaci času ze sítě (protokolem NTP) a použití hardwarového

RTC modulu. Budou zde porovnány výhody a nevýhody obou přístupů a zmíněná jejich realizace na ESP32. [44]

1.3.1 Význam přesného času pro funkce rádia

Přesný čas je kritický zejména pro funkci budíku a časového spínání. Pokud má rádio každé ráno v 7:00 zapnout stream zpráv, musí mít spolehlivě nastavený čas, jinak by se stream nezapínal správně. Podobně u plánovaného záznamu pořadu je nutné zahájit nahrávání ve správnou dobu. Také pro zobrazení digitálních hodin na displeji rádia (přístroj slouží jako hodiny pro uživatele) je vyžadováno, aby čas zůstal přesný i při delším provozu. V neposlední řadě mohou hrát roli i časová razítka u síťových protokolů, některé streamovací servery vyžadují správně nastavený čas kvůli ověření klienta, HTTPS certifikáty jsou platné jen při správném čase a datu. Je tedy žádoucí, aby zařízení ESP32 vědělo reálný čas s přesností na sekundy a dlouhodobě jej udržovalo. Samotný mikrokontrolér ESP32 obsahuje interní hodiny reálného času využitelné v deep sleep režimu, ty by však nemusely poskytovat dostatečnou přesnost. Proto je možné použít buďto pravidelné aktualizace času z internetu, nebo lze připojit externí RTC čip. [44; 45; 46]

1.3.2 Síťová synchronizace času (NTP)

Standardním způsobem, jak zjistit přesný čas v zařízení připojeném k internetu, je využití NTP (Network Time Protocol). Jde o internetový protokol sloužící k synchronizaci času zařízení podle časových serverů. NTP funguje nad UDP (port 123) a umožňuje počítačům získat od specializovaných časových serverů aktuální čas UTC s vysokou přesností. NTP je schopno dosáhnout přesnosti cca 1 ms v LAN sítích a desítky ms ve WAN sítích. Provoz NTP je hierarchický (tzv. stratum), na vrcholu (stratum 0) jsou etalonové zdroje času jako atomové hodiny a GPS, stratum 1 jsou time servery přímo synchronizované s nimi a stratum 2 se synchronizují se stratum 1 atd.. Běžný klient (např. ESP32) se dotazuje veřejných NTP serverů (stratum 2-3), kterých je po světě mnoho (např. projekt pool.ntp.org poskytující volně dostupné NTP servery). [44; 45; 47]

Espressif SDK pro ESP32 obsahuje podporu NTP ve formě modulu LwIP SNTP. V arduino frameworku lze jednoduše použít funkci „configTime()“ ze <time.h>, která interně spustí SNTP klienta. Ten se připojí k zadanému NTP serveru (např. „pool.ntp.org“) a jednorázově synchronizuje systémový čas ESP32. Typicky se po zavolání configTime() naváže spojení na NTP server, získá se časové razítko UTC a nastaví se v interních RTC

hodinách ESP32. Lze nastavit i časovou zónu a případně DST offset (letní/zimní čas). Espressif implementace umí i periodicitu, ve výchozím nastavení se čas synchronizuje každou hodinu automaticky, aby se eliminoval drift interních hodin. Případně si může programátor vyžádat aktualizaci ručně (např. jednou za den). Existuje i funkce, která umožňuje čekat na dokončení synchronizace (funkce `sntp_sync_wait()`). Po synchronizaci si ESP32 udržuje čas samo, pomocí kombinace interního vysokofrekvenčního timeru a RTC timeru, také díky opakovaným NTP dotazům se průběžně koriguje chyba v přesnosti času. Přesnost výsledného času se typicky pohybuje v jednotkách až desítkách ms, což poskytuje dostatečnou přesnost pro budík či zobrazování hodin. [44; 45; 46; 47]

Hlavní výhoda NTP řešení je, že odpadá nutnost dalšího hardware. Výsledné řešení je velmi přesné, jak bylo zmíněno, tak NTP dokáže synchronizovat čas s přesností na desítky milisekund přes internet. Pro účely projektu (budík s přesností na sekundy) je to více než dostatečné. Další výhodou je automatická korekce, i když interní hodiny ESP32 mají mírný drift, SNTP klient je každou hodinu dorovná, takže akumulovaná chyba je dlouhodobě „nulová“. Synchronizace NTP je také poměrně rychlá a nenáročná, typický dotaz a odpověď proběhnou během zlomku sekundy, zátěž na síť i CPU je minimální. Pro platformu s Wi-Fi konektivitou, která je stejně využita pro streaming, je logické využít internet i pro synchronizaci času. Implementace NTP je jednoduchá, v podstatě jedna funkce `configTime()` a o zbytek se stará ESP-IDF knihovna. Lze také získat informaci o časovém pásmu a letním čase (ESP32 umí aplikovat posun podle zadané zóny). NTP také umožňuje bezpečnou synchronizaci, pokud by byl požadavek přes TLS, dalo by se zajistit, že se čas nevezme od podvrženého serveru. V praxi se NTP servery dostatečně ověřují tím, že jsou to známé pooly. [44; 45; 47]

U NTP řešení je podmínkou, že zařízení má připojení k internetu. Pokud bychom chtěli, aby ESP32 rádio fungovalo i off-line (bez Wi-Fi), pak by po odpojení nemělo jak zjistit aktuální čas (mohlo by dále číst své interní hodiny, ale ty se časem odchýlí a nebudou přesné). Pro čistě online zařízení to nevádí, předpokládá se, že rádio ke streamování stejně internet potřebuje. Další drobná nevýhoda je závislost na dostupnosti NTP serverů, nicméně veřejné pooly jsou robustní a výpadek synchronizace o pár hodin by neznamenal kritický problém. Bezpečnostní aspekt NTP protokolu je, že není šifrovaný ani autentizovaný. Teoreticky by útočník mohl podvrhnout odpověď a nastavit špatný čas. To by mohlo ovlivnit například platnost TLS certifikátů (při velkém posunu času). V praxi je ale riziko malé a pro budík není podvržený čas zásadní problém. Každopádně pro kritické

aplikace existuje NTP autentizace nebo protokol NTS (Network Time Security), ale to je nad rámec běžného IoT použití. Z hlediska spotřeby energie může být nevýhoda, že ESP32 kvůli synchronizaci musí být periodicky probouzeno a komunikovat. Pokud by rádio bylo napájené z baterie a většinu času spalo, pak by pravidelné probouzení kvůli NTP mohlo zvyšovat spotřebu. V našem případě se však předpokládá napájení ze sítě. [44; 45; 47; 48]

Lze shrnout, že síťová synchronizace času pomocí NTP je pro ESP32 jednoduché, levné a efektivní řešení, pokud je k dispozici připojení k internetu, tak je výsledný čas velmi přesný. Řada projektů vyřešila synchronizaci času tak, že ESP32 se při startu připojí k Wi-Fi, zavolá NTP synchronizaci a poté po dobu provozu má ESP32 již k dispozici reálný čas. Udržování času během chodu zajistí kombinace interních časovačů, pokud by však ESP32 pracovalo dlouhodobě bez synchronizace, čas by se začal rozcházet. Při pravidelné synchronizaci, ale může ESP32 dlouhodobě pracovat s prakticky nulovou chybou, proto je metoda NTP ideální řešení pro zařízení, která mají trvalé připojení k Wi-Fi. [44; 45]

1.3.3 Hardwarový RTC modul

Druhou možností, jak zajistit přesný čas v zařízení, je použití specializovaného RTC modulu - malého externího obvodu, který udržuje čas nezávisle na hlavním mikrokontroléru. RTC moduly (např. DS3231, DS1307, PCF8523 a další) obsahují vlastní krystalový oscilátor, často kompenzovaný pro co nejvyšší přesnost a záložní baterii, která udržuje hodiny v chodu i při vypnutí hlavního napájení. Tyto moduly komunikují s ESP32 typicky přes sběrnici I²C (DS3231, PCF8523) a umožňují přečíst aktuální čas. [46; 49]

Připojení RTC čipu k ESP32 je přímočaré, například modul DS3231 se připojí na piny I²C (GPIO21 SDA, GPIO22 SCL na ESP32 Devkit). V softwaru lze využít existující knihovny (<RTCLib.h> od Adafruit podporuje DS3231 i další) nebo napsat jednoduché rutiny pro komunikaci. Při startu ESP32 může načíst čas z RTC modulu a nastavit si jej jako systémový. Dále může ESP32 číst RTC pro zobrazení. DS3231 má také alarm, lze si v nich nastavit budík, který v určitou dobu vyvolá přerušení. To je užitečné například v deep sleep režimu ESP32, kdy RTC modul může probudit ESP32 externím signálem v nastavený čas. Zkrátka, integrace RTC modulu je poměrně snadná a dobře podporovaná. Existují příklady i knihovny, stačí modul zakoupit. [50; 51; 52]

Největší výhodou externího RTC je nezávislost na připojení k síti a velmi dobrá dlouhodobá stabilita času. Kvalitní RTC jako DS3231 mají zabudovaný TCXO (Teplotně

kompenzovaný krystalový oscilátor), který zaručuje přesnost plus mínus 2 minuty za rok v běžných teplotách. To odpovídá chybě 0,0003%, tedy méně než 0.2 sekundy za 24 hodin. To je výborný výsledek, pro srovnání obyčejný krystal v hodinkách (DS1307 s externím krystalem) mívá drift kolem 1 minuty za měsíc (závisí na teplotách). DS3231 tak nabízí řádově lepší stabilitu. Další výhodou RTC modulu je, že běží, i když je hlavní zařízení vypnuté nebo resetované, z toho důvodu má vlastní záložní baterii. To znamená, že po vypnutí a odpojení rádio od napájení by RTC modul stále držel správný čas. Odpadá nutnost nastavovat čas ručně nebo čekat na NTP při startu. Z hlediska funkčnosti budíku je to ideální, protože s RTC modulem by mohl budík fungovat bez časových driftů i offline. RTC moduly mají také velmi nízkou spotřebu (řádově μA z baterie), takže mohou běžet roky na knoflíkovou baterii. Pro ESP32 v deep sleep by to znamenalo, že může spát s odpojenou Wi-Fi a probudit se v okamžiku, kdy RTC vyvolá přerušení. Každopádně externí RTC poskytuje spolehlivý zdroj přesného času, nezávislého na Wi-Fi připojení a serverech. [52; 53]

Očividnou nevýhodou je nutnost doplňkového hardware. RTC modul zabírá další místo, vyžaduje zapojení a konfiguraci. Zvyšuje se tím složitost zařízení (byť mírně) a také cena. Pro malá IoT zařízení by mohla vadit i velikost modulu. Dalším aspektem je, že RTC modul sám o sobě neřeší automatické nastavení času. Při úplně prvním použití nebo po vybití baterie neví, kolik je hodin a je nutné jej nastavit (ručně z kódu nebo synchronizací s NTP při prvním připojení). Čili často se kombinuje RTC modul s NTP pro nastavení reálného času a zapsání této hodnoty jako výchozí. Pak již běží RTC modul samostatně. Pokud by rádio bylo v provozu delší období, je třeba počítat s korekcí letního času (RTC čip sám o sobě neřeší). Další nevýhodou je omezená přesnost při dlouhých obdobích bez kalibrace (čip je přesný, ale může se posunout o cca 2 minuty za rok). Použití RTC modulu však nevylučuje synchronizaci času přes NTP ani připojení k Wi-Fi. [52; 53]

Z tohoto popisu lze usoudit, že RTC modul je skvělým řešením pro zařízení, které nemají garantované připojení k internetu nebo musí udržovat čas i při odpojení (např. hodiny s ESP32, které se jen občas synchronizují). Pro internetové rádio, které je většinou online, není RTC nutný, ale může zvýšit spolehlivost budíku (např. kdyby vypadla Wi-Fi nebo by došlo k restartu). V projektech, kde se klade důraz na vysokou přesnost a stabilitu času offline, je RTC modul DS3231 vhodným kandidátem díky své vysoké přesnosti. Alternativně levnější DS1307 poskytuje přesnost s odchylkou cca 1 minuta za měsíc. Pro bateriové aplikace s deep sleep se používají i ultranízkonapěťové RTC (např. RV-8803-C7

se spotřebou pár stovek nA). V kontextu ESP32 DevKit modulů je vhodné zmínit, že mnohé mají vývody pro připojení 32,768 kHz krystalu k internímu RTC ESP32. Pokud by se tento krystal osadil, zlepšila by se přesnost interních hodin v deep sleep. Interní RC oscilátor by se nahradil krystalem s průměrným driftem 2 sekund na den za cenu mírně zvýšené spotřeby. Celkově tedy, pro co největší spolehlivost a zároveň přesnost času je externí RTC modul vhodným doplňkem. [45; 50; 53; 54; 55]

2 MIKROKONTROLÉRY A MODULY ESP32

Mikrokontroléry ESP32 od firmy Espressif Systems tvoří širokou rodinu čipů pro vestavěné systémy s bezdrátovou konektivitou Wi-Fi a Bluetooth. První čip ESP32 od této firmy byl na trh uveden roku 2016 jako nástupce populárního ESP8266. Z původního modelu vzniklo několik variant a následných generací, každá se zaměřením na konkrétní aplikace. Mezi hlavní představitele patří původní dvoujádrové ESP32 a dále novější modely například: ESP32-S2, ESP32-S3 a ESP32-C3. Tyto modely byly na trh uvedeny roku 2020. [56]

2.1 Populární modely ESP32

ESP32 (ESP32 Series) je původní verze, která obsahuje dvoujádrový 32-bitový procesor Xtensa LX6 s frekvencí až 240MHz a celkovým výkonem okolo 600 DMIPS (Dhrystone Millions Instructions Per Second). K dispozici má 520 kB interní SRAM a 448 kB ROM. Integrovaná rádiová část podporuje Wi-Fi 4 (802.11 b/g/n 2,4 GHz) a Bluetooth v4.2 (Bluetooth Classic i Low Energy). Čip nabízí bohatou škálu periférií – dvě 12-bitová ADC (až 18 kanálů), dvě 8-bitová DAC, 34 GPIO pinů, rozhraní UART, SPI, I²C, I²S, SDIO, PWM, kapacitní dotykové senzory (10 kanálů), hallův senzor, rozhraní CAN (TWAI) a další. Typická pracovní frekvence procesoru je 240 MHz a čip dosahuje výborného poměru výkonu vzhledem k ceně. Velikost čipu (pouzdro QFN48) je přibližně 5x5mm. Původní ESP32 obsahuje technologii Ultra-Low-Power (ULP), to je pomocný ultra-nízkonapěťový koprocessor, který může běžet v režimu hlubokého spánku a monitorovat vstupy při spotřebě v řádu desítek μ A. [10; 56; 57; 58]

ESP32-S2 je nástupce z roku 2020 zaměřený na nižší cenu a spotřebu pro IoT senzory. Obsahuje jednojádrový procesor Xtensa LX7, který je také taktovaný na 240 MHz, ale s výkonem 300 DMIPS na jedno jádro a celkovou nižší spotřebou je tento procesor efektivnější než jeho předchůdce. Oproti původnímu ESP32 má ESP32-S2 pouze Wi-Fi a neobsahuje Bluetooth bezdrátové komunikační rozhraní. Paměť SRAM má kapacitu 320 kB a paměť ROM 128 kB. Výhodou ESP32-S2 je integrovaný USB OTG (podpora USB 2.0 rozhraní) a více GPIO pinů (43) v porovnání s ESP32. Zachovává 12-bitové ADC (rozšířené na 20 kanálů) a 8-bitové DAC, a přidává například až 14 dotykových senzorů. Bezpečnostní funkce (secure boot, šifrování flash) jsou vylepšeny. ESP32-S2 nabízí rychlejší/bezpečnější varianty oproti první generaci. ESP32-S2 je vyráběn v pouzdru QFN56 o velikosti 7x7mm. [56]

ESP32-S3 je výkonnější varianta navazující na ESP32-S2, ale opět s dvoujádrovým procesorem Xtensa LX7 o pracovní frekvenci 240 Mhz a u tohoto modelu je také návrat Bluetooth. ESP32-S3 disponuje 512 kB SRAM a 384 kB ROM. Oproti S2 má S3 vyšší počet GPIO pinů (45) a podporuje obdobnou sadu periférií včetně USB OTG, RGB LCD rozhraní, kamerového rozhraní (jednotka pro přímé připojení kamery) a vylepšeného AI akcelerátoru (ve formě vektorových instrukcí Xtensa pro rychlejší zpracování např. neuronových sítí). Bluetooth je podporován ve verzi 5 (BLE) a Wi-Fi zůstává 2,4 GHz 802.11n jako u ostatních. Bezpečnost je na úrovni S2 – hardwarové akcelerátory AES-256, SHA-512, RSA-4096, podpora HMAC, digitálních podpisů atd. Pouzdro čipu je také QFN56 (7x7mm). [56]

ESP32-C3 – Speciální varianta představená v roce 2020 cílicí na levné a bezpečné zařízení. Odlišuje se architekturou CPU – obsahuje jedno jádro RISC-V 32-bit namísto architektury Xtensa. Taktován je na 160 MHz a nabízí 400 kB SRAM a 384 kB ROM. ESP32-C3 podporuje Wi-Fi 4 (2,4 GHz) a Bluetooth 5 LE (s podporou BLE Mesh), ale nemá Bluetooth Classic. Některé parametry jsou zredukovány – např. pouze 22 GPIO pinů, 6 kanálů 12-bitového ADC a chybí DAC převodníky či dotykové senzory. Výhodou je však malé pouzdro QFN32 o velikosti 5x5mm a velmi nízká spotřeba v úsporných režimech. Díky RISC-V jádru má ESP32-C3 otevřený ekosystém nástrojů a jednodušší podporu pro některé moderní jazyky. Bezpečnostně nabízí podobné funkce (secure boot 3. generace, flash i PSRAM encryption) jako modely S2 a S3. [56]

Všechny zmíněné modely ESP32 integrují Wi-Fi 802.11b/g/n (2,4 GHz) s podporou režimů AP, STA i P2P a rychlostí PHY až 150 Mb/s. Kromě ESP32-S2 a C3, které nemají Bluetooth Classic, disponují ostatní varianty Bluetooth Dual-Mode (kombinace klasického BR/EDR a BLE) či alespoň BLE 5.0 v novějších čipech. Všechny rovněž podporují hardwarově akcelerované šifrování (AES, SHA, RSA atd.) a funkcionality jako secure boot a šifrování flash paměti, přičemž novější modely vylepšují rychlost a bezpečnost těchto funkcí. [10; 56]

Pro obecné shrnutí, původní ESP32 nabízí nejuniverzálnější výbavu (Wi-Fi + Bluetooth Classic/LE, dvoujádrový procesor) a vysoký výkon ~600 MIPS, hodí se pro náročnější úlohy (např. paralelní zpracování signálu, audio). Model ESP32-S2 je zjednodušený (jednojádrový procesor, bez Bluetooth) a optimalizovaný pro levná IoT zařízení se senzory, kde stačí Wi-Fi konektivita. ESP32-S3 kombinuje výhody S2 (vylepšené periferie, USB) s výkonem dvou jader a podporou BLE 5. Cílí tak např. na

aplikace s AI, zpracování obrazu či zvuku, kde je potřeba vyšší výkon CPU i periferie jako kamera či LCD. ESP32-C3 pak představuje úsporné řešení s RISC-V jádrem, vhodné pro jednodušší úlohy, senzory a zařízení na baterie, kde využije nízkou spotřebu a dostatečnou konektivitu (Wi-Fi + BLE) při nižší ceně a menší velikosti. Vývoj rodiny pokračuje i dále, existují např. nové modely ESP32-C5 a C6, které již poskytují např. Wi-Fi 6, IEEE 802.15.4 a protokoly Thread/Zigbee, což však přesahuje rámec tohoto přehledu. [56; 59]

2.2 ESP32-WROOM-32 architektura a technické parametry

ESP32-WROOM-32 je nejrozšířenější modul s čipem ESP32 uvedený na trh v roce 2016, který integruje samotný SoC ESP32 spolu s potřebnou pamětí flash a anténou do jednoho pouzdra. Jde o obdélníkový modul o rozměrech cca 18x25,5 mm s plošnými vývody po stranách. Srdcem modulu je čip ESP32-D0WDQ6, který využívá dvoujádrový 32-bitový procesor Tensilica Xtensa LX6 s maximální taktovací frekvencí až 240MHz. Jádra jsou založena na moderní RISC architektuře s možností výběru 5 nebo 7-stupňové pipeline a podporou pro paralelní instrukční zpracování. Procesor podporuje rozšířené DSP instrukce a může být vybaven volitelnou jednotkou FPU (Floating-Point Unit) kompatibilní se standardem IEEE 754 pro jednoduchou i dvojitou přesnost, což umožňuje efektivní zpracování výpočtů s plovoucí desetinnou čárkou. Architektura je typu Harvard, ta odděluje paměť pro instrukce a data, a čip disponuje oddělenými instrukčními a datovými cache s až čtyřcestnou asociativitou. Díky dvoujádrovému provedení je možné v rámci FreeRTOS provozovat skutečný paralelismus, například jedno jádro lze dedikovat aplikační logice, zatímco druhé může obsluhovat síťový stack (Wi-Fi/Bluetooth). Výpočetní výkon procesoru je uváděn kolem 600 DMIPS. [10; 56; 60]

2.2.1 Paměť

Uvnitř ESP32-WROOM-32 je integrována externí SPI flash paměť o kapacitě 4 MB pro uložení programu. Samotný čip ESP32 má 520 kB SRAM pro data a instrukce a 448 kB interní ROM paměť s knihovnami a firmwarem pro bootování. Také disponuje pamětí RTC fast a RTC slow, každá o velikosti 8 kB. Tyto paměti jsou připojené k RTC doméně a lze je využívat v režimech hlubokého spánku, například pro monitorování vstupů nebo čidel. Modul dále obsahuje interní krystal 40 MHz pro taktování rádiové části i CPU, takže nevyžaduje externí oscilátor. Nepřítomná je však externí RAM – pokud by aplikace vyžadovala více operační paměti, existuje varianta modulu ESP32-WROVER se 4 MB SPI PSRAM, avšak standardní ESP32-WROOM-32 tuto PSRAM neobsahuje. [10; 61]

2.2.2 Bezdrátová konektivita

Modul ESP32-WROOM-32 poskytuje plnohodnotné Wi-Fi rozhraní v pásmu 2,4 GHz (802.11b/g/n, 20 MHz i 40 MHz kanály). Podporuje režimy AP (Access Point), STA (Station) i P2P (Peer-to-Peer) s datovými toky až 150 Mbps (802.11n 40 MHz). Součástí je integrovaná anténa (na PCB modulu) optimalizovaná pro 2,4 GHz. Citlivost přijímače Wi-Fi dosahuje cca 97 dBm a vysílač má výkon typicky +18,5 dBm. Kromě Wi-Fi čip podporuje Bluetooth 4.2 Dual Mode, tedy jak klasické BR/EDR, tak úsporné Bluetooth Low Energy. Lze jej využít pro komunikaci se staršími BT periferiemi i moderními BLE senzory. Podporováno je mimo jiné také rozšíření Bluetooth AFH (adaptivní přeskokování kmitočtů) a základní audio kodeky CVSD/SBC v hardwaru. [10; 59]

2.2.3 Periferie a rozhraní

ESP32-WROOM-32 přivádí na své piny bohatou sadu rozhraní čipu ESP32. K dispozici jsou například: 3x univerzální UART, 3x SPI (1x plně duplexní pro hosta, 2x poloduplexní), 2x I²C sběrnice, 2x 8-bitové DAC, 2x 12-bitové ADC převodníky, 16-kanálový PWM generátor (pro řízení LED, serv apod.), 8-kanálové čítače pulsů, 8-kanálový dálkový ovladač (Remote Control Peripheral pro generování přesných časových pulsů), 10 kapacitních dotykových senzorů, rozhraní SDIO pro SD karty nebo jako slave, Ethernet MAC (pro připojení externího PHY), SPI slave atd.. Nechybí ani periferní jednotka I²S (2 kanály, 8-48 bit, použitelná pro audio vstup/výstup) a kamerové rozhraní (8-bit paralelní, pro snímače CMOS kamer). Díky této výbavě je ESP32 velmi univerzální. Zvládá připojit různé senzory (analogové či digitální), řídit motory, LED displeje, sbírat data z mikrofونů či kamer a komunikovat po sběrnících se zařízeními typu čtečky SD karet, externí DAC/ADC, atd.. [10; 59; 62]

2.2.4 Výkon a energetická náročnost

Při maximální frekvenci 240 MHz a aktivní Wi-Fi dosahuje ESP32 vysokého výpočetního výkonu, kromě zmíněných 600 DMIPS má např. v benchmarku CoreMark skóre cca 1080 (539 na jedno jádro). To jej řadí výrazně nad běžné 8-bitové mikrokontroléry, například mikrokontrolér Atmega328P, běžně používaný v Arduino Uno dosahuje cca 16 DMIPS při frekvenci 16 MHz. Nevýhodou je však vyšší spotřeba v aktivním režimu, podle dokumentace činí průměrný proud cca 80 mA při aktivní komunikaci. Při plném vysílacím výkonu Wi-Fi se ale může odběr krátkodobě zvyšovat až na 200 mA, proto se doporučuje zdroj schopný dodat alespoň 500 mA. Naopak v úsporných režimech dokáže ESP32

výrazně šetřit energii. V režimu Modem-sleep (Wi-Fi modem se periodicky probouzí) klesá odběr na jednotky mA, v Light-sleep (uspání CPU, zachování RTC paměti) na stovky μA a v nejhlubším Deep-sleep s ULP koprocесorem může klesnout spotřeba pod 10 μA . To vše činí ESP32 vhodným i pro bateriově napájené aplikace s dlouhou výdrží, pokud se chytře využijí spánkové režimy. Podmínkou je ovšem kvalitní návrh hardware, modul sám má velmi nízkou spotřebu ve spánku, ale ostatní části desky mohou odběr navyšovat. [10; 58; 63]

2.2.5 Bezpečnost

ESP32-WROOM-32 obsahuje vestavěné bezpečnostní prvky. Podporuje Secure Boot (ověřování digitálního podpisu firmware při spuštění) a Flash Encryption (šifrování obsahu externí flash paměti za běhu). Dále má jedinečnou 1024-bitovou eFUSE paměť (OTP) pro uložení klíčů či identifikátorů. Hardwarový akcelerátor poskytuje šifrování AES-128/192/256, výpočet hashů SHA-2 (256/512 atd.), RSA až 4096 bitů i generátor náhodných čísel. Novější modely S2/S3/C3 tato zabezpečení dále vylepšují (větší OTP, podpora HMAC a ECC, rychlejší šifrování PSRAM apod.), avšak i původní ESP32 je dostatečně zabezpečen pro běžné IoT použití. Pro kritické aplikace (platební systémy, trezory apod.) může být výhodou přechod na S2/S3 s pokročilejším zabezpečením. [10; 56; 58]

Z uvedených informací vyplývá, že modul ESP32-WROOM-32 představuje výkonný a univerzální mikropočítačový modul pro bezdrátové aplikace. V kompaktním provedení integruje dvoujádrový procesor Xtensa LX6 taktovaný na 240 MHz, Wi-Fi/Bluetooth konektivitu a bohatou paletu analogových i digitálních periférií. Díky tomu zvládne je schopný zvládnout vše od sběru dat z čidel, přes zpracování zvuku a obrazu, až po přímou komunikaci po internetu. To vše je dostupné při zachování relativně nízké ceny a možnosti provozu v úsporných režimech s minimální spotřebou. ESP32-WROOM-32 je díky své jednoduché integraci a napájení 3,3 V široce využívaným modulem pro různá IoT zařízení s využitím Wi-Fi/Bluetooth. [10; 57]

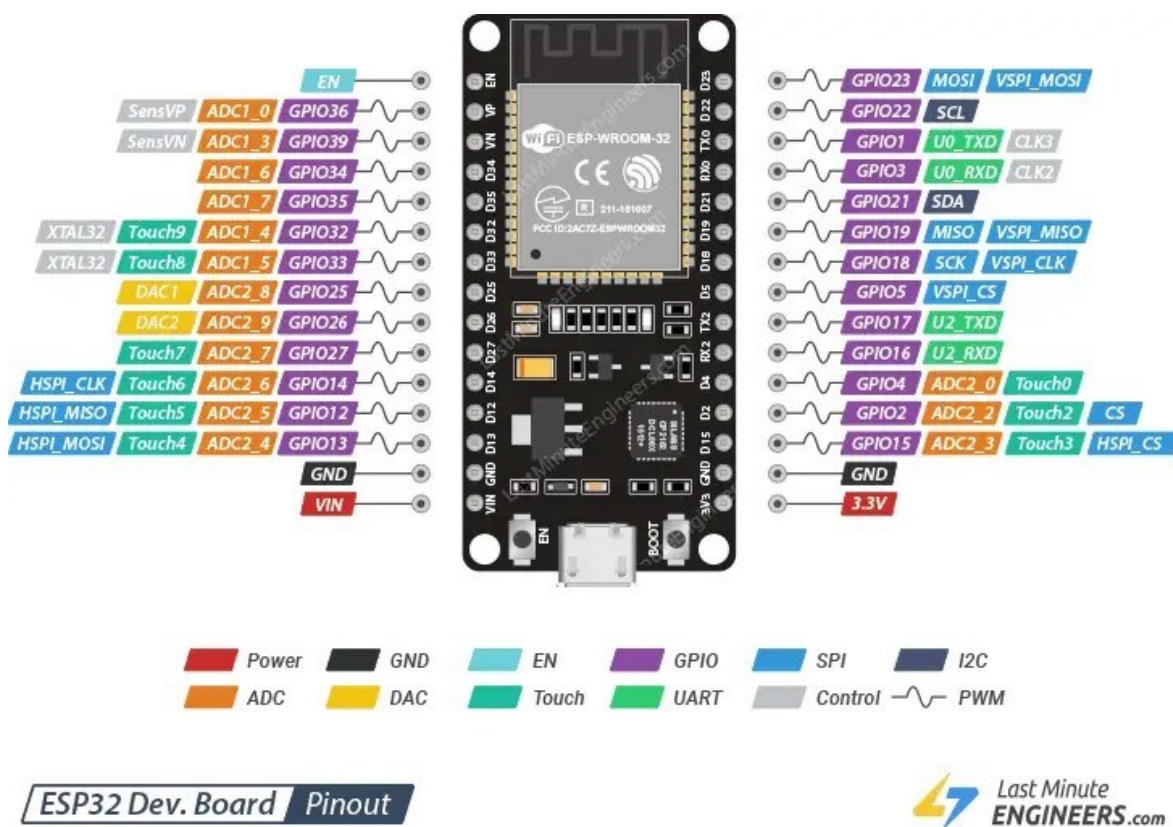
2.3 Vývojové desky s ESP32

Pro usnadnění práce s modulem ESP32-WROOM-32 vzniklo množství vývojových desek. Tyto desky obvykle obsahují samotný modul ESP32, USB-UART převodník pro programování, stabilizátor napětí a případně další doplňky (tlačítka RESET/BOOT,

konektor pro baterii, atd.). Níže je uveden přehled několika populárních vývojových desek. [64]

2.3.1 Espressif ESP32-DevKitC

Oficiální jednoduchá deska přímo od výrobce čipu použitá v této bakalářské práci. Jedná se o malý modul (tzv. minimal system), kde je na plošném spoji osazen ESP32-WROOM-32 a vyvedeny jsou všechny I/O piny ESP32 na boční header piny v rastru 2,54 mm. Deska obsahuje USB rozhraní (typicky CP2102 nebo CH340), tlačítko Reset a Boot a jednu status LED. Neobsahuje naopak žádné přídavné senzory ani periférie. K dispozici jsou varianty s 4 MB nebo 8 MB flash, případně také osazeným modulem ESP32-WROVER (s PSRAM). Výhodou je nízká cena a oficiální podpora. Jedná se o základní desku, která je díky svým malým rozměrům vhodná i pro vestavbu do prototypů. [54; 64]



Obrázek 1 ESP32_DevKitC pinout

(zdroj: [129])

2.3.2 NodeMCU-32S

Populární vývojová deska od komunitního projektu NodeMCU. Nabízí obdobný koncept jako DevKitC: modul ESP32, USB-UART, napájecí část a piny od rastru 2,54 mm. Layout

je inspirován starší NodeMCU pro ESP8266, má šířku 30 pinů a lze jej zasunout do nepájivého pole. NodeMCU-32S se často prodává jako DoIT DevKitV1. Funkčně je velmi podobný DevKitC, liší se jen drobnostmi (např. je použit čip CP2102) a díky hromadné produkci je snadno dostupný. [65; 66]

2.3.3 Adafruit HUZZAH32 Feather

Vývojová deska formátu Feather od firmy Adafruit. Obsahuje ESP32-WROOM modul a kromě standardních prvků (USB-UART CP2104, napájení atd.) přidává integrovaný Li-Pol nabíječ a konektor pro připojení Li-Pol baterie. Deska je navržena pro mobilní IoT projekty. Lze ji napájet z USB nebo baterie a má obvody pro nabíjení. Rozložení pinů odpovídá ekosystému Feather (Adafruit standard pro rozložení pinů). HUZZAH32 má 4 MB flash a 520 kB RAM ve standardní verzi. Novější verze Feather V2 dokonce integruje modul s 8 MB flash a 2 MB PSRAM pro náročnější aplikace. Jde o profesionálně provedenou desku s bohatou dokumentací a podporou v Arduino IDE i MicroPython. [67; 68]

2.3.4 SparkFun ESP32 Thing Plus

Tato varianta vývojové desky od SparkFun má formát Feather s USB-C konektorem. Zajímavostí je, že SparkFun osazuje moduly s větší flash pamětí např. Thing Plus má 16 MB flash (oproti běžným 4 MB), což umožňuje např. použití MicroPythonu či větších aplikací. Samozřejmostí je Li-Pol nabíjecí obvod a Qwiic konektor pro I²C periférie. Tyto desky se vyznačují kvalitním zpracováním a jsou zaměřeny na prototypování IoT projektů s důrazem na snadné rozšíření (Feather kompatibilita, Qwiic). [69]

2.3.5 WEMOS/LOLIN D32, D32 Pro

Další populární řada nízkonákladových vývojových desek od firmy VEMOS (Lolin). Verze D32 je podobná NodeMCU, verze D32 Pro pak nabízí slot pro SD kartu a 4 MB PSRAM navíc. Tyto desky často volí hobby komunita pro projekty díky své dobré ceně. [70; 71]

2.3.6 M5Stack Core2

Zajímavý koncept miniaturní krabičky s ESP32. Základní modul M5Stack Core2 obsahuje ESP32, displej, baterii a tlačítka v kompaktním pouzdře. K němu lze připojovat další

stackovatelné moduly (GPS, RFID, atd.). Je určen pro rychlé prototypování IoT gadgetů s uživatelským rozhraním. [72]

Existuje řada dalších různých desek s ESP32, hlavní rozdíly spočívají ve formátu (velikosti, pinout), použití dodatkového hardware (displeje, čidla, bateriové obvody) a někdy v osazeném typu modulu. Z hlediska výkonu samotného ESP32 však tyto desky nabízí stejný čip, tedy výkonnostní parametry se neliší. Při výběru desky pro projekt je tak klíčové zohlednit požadované rozhraní (např. potřeba Li-Pol nabíjení favorizuje Feather/Thing Plus), fyzické rozměry a dostupnost podpory. ESP32-DevKitC je ideální pro začátek díky jednoduchosti a množství oficiální dokumentace a podpory. Zatímco Feather nebo Thing Plus se hodí pro pokročilejší vestavby s využitím baterie a M5Stack Core2 pro rychlé vytvoření funkčního prototypu s minimem zapojování.

2.4 Možnosti programování ESP32

Ekosystém ESP32 podporuje různé programovací nástroje a jazyky, což značně zvyšuje přitažlivost platformy pro vývojáře. Základním oficiálním prostředkem je ESP-IDF (Espressif IoT Development Framework), vývojový framework v C/C++ poskytovaný přímo Espressifem. ESP-IDF umožňuje nízkourovňový přístup ke všem funkcím čipu, obsahuje ovladače periférií, knihovny pro Wi-Fi, Bluetooth, FreeRTOS a množství vzorových kódů. Vývoj probíhá primárně v prostředí Cmake a Makefile s možností integrace do IDE (Visual Studio Code, Eclipse atd.). Pro odbornější vývoj (např. ladění výkonu) je ESP-IDF nejmocnějším nástrojem a Espressif jej používá i interně. [73; 74]

Pro rychlejší vývoj a pro komunitu makerů je k dispozici oficiální podpora Arduino frameworku pro ESP32. Jedná se o port Arduino API nad hardwarem ESP32. Implementace Arduino-ESP32 interně využívá ESP-IDF, Arduino core v podstatě volá funkce ESP-IDF, ale zabalí je do známých jednoduchých Arduino tříd a funkcí. Díky tomu může vývojář programovat ESP32 podobně jako Arduino Uno (např. funkce pinMode(), digitalWrite(), knihoven WiFi.h, atd.), avšak s tím rozdílem, že uvnitř běží FreeRTOS a plný Espressif stack. Výhodou je snadnost, mimo jiné proto, že mnoho existujících Arduino knihoven (např. pro senzory, displeje) funguje na ESP32. Arduino IDE také zjednodušuje kompilaci a nahrávání. Nicméně nevýhodou naopak může být menší kontrola nad detaily a mírná režie. Pro časově kritické či paměťově náročné části může být potřeba sáhnout k nativnímu ESP-IDF. Každopádně, Arduino podpora činí ESP32 přístupné i pro

začátečníky a silná komunita přispěla řadou knihoven a návodů, které jsou dostupné pro kohokoliv. [73; 74; 75]

Další populární možností je MicroPython, tj. interpret jazyk Python optimalizovaný pro mikrořadiče. ESP32 je natolik výkonný, že zvládne provozovat MicroPython firmware přímo v modulu (typicky se MicroPython nahraje do flash paměti jako celek a následně interpretuje Python scripty). MicroPython na ESP32 umožňuje velmi rychlé prototypování. Kód lze psát a spouštět interaktivně (přes sériovou konzoli či Wi-Fi telnet/WebREPL). Podporuje mnoho hardwarových modulů (GPIO, ADC, I²C, atd.) prostřednictvím vlastní knihovny „machine“. Výhodou je jednoduchost a produktivita, tedy i méně zkušení programátoři mohou zprovoznit IoT program bez nutnosti kompilace C kódu. Nicméně daní za tuto „jednoduchost“ je výrazně nižší výkonnost běhu programu, neboť Python je interpretovaný. Studie ukazují, že kód v MicroPythonu na ESP32 může běžet až několik set krát pomaleji než ekvivalent v jazyce C. Například experimentální měření algoritmů (CRC, šifrování, filtrace signálu) ukázalo, že MicroPython měl nejhorší časy, řádově 1000x delší než C, Rust či TinyGo implementace těch stejných úloh. Na druhou stranu, vývoj v MicroPythonu je velmi rychlý a kód je výrazně kratší. Pro méně náročné skripty nebo výuku je to vynikající nástroj. Lze jej doporučit např. pro sběr dat ze senzorů a jejich odesílání na web, kde rychlost není kritická a oceníme rychlé iterace vývoje. [76; 77; 78]

2.4.1 Další možnosti programování

Kromě výše zmíněných hlavních cest existují i další možnosti jak naprogramovat ESP32.

LUA vznikl původně pro ESP8266 jako NodeMCU firmware s LUA interpretem. Ten byl portován i na ESP32, ale vzhledem k existenci MicroPythonu jeho popularita klesla. [79]

ESP-AT firmware – Espressif poskytuje firmware ESP-AT, který umožní používat ESP32 jako modem ovládaný AT příkazy. To je užitečné, pokud ESP32 funguje jako Wi-Fi/Bluetooth modul připojený k jinému mikrokontroléru. [80]

ESP-ADF (Audio Development Framework) slouží pro audio aplikace (internetová rádia, BT reproduktory). Je to specifický framework s knihovnami pro přehrávání, DSP efekty apod.. [81]

FreeRTOS – Přímě v ESP-IDF lze psát vícevláknové aplikace využívající FreeRTOS (který běží v ESP32 out-of-the-box). Lze tak realizovat komplikované real-time úlohy.

ESP-IDF také podporuje spolupráci více jader, synchronizaci vláken (mutexy, semafore, atd.). [82]

Zephyr OS je open-source RTOS podporovaný i na ESP32 navržený s důrazem na modularitu, malou paměťovou zátěž a přenositelnost napříč architekturami. Podporuje široké spektrum zařízení od jednoduchých senzorů až po složité vestavěné řídicí jednotky a bezdrátové IoT aplikace. [83]

ESP32 je natolik populární, že vznikly porty různých jazyků, experimentálně např. Espruino (JavaScript engine), LuaRTOS či běh Kotlinu a dalších. Ty však mají menší komunitní podporu. [84; 85]

Shrneme-li to, vývojář má na výběr od nízkoúrovňového jazyka C (ESP-IDF) přes jednodušší Arduino C++ abstrakci až po skriptovací jazyky jako Python. Tato flexibilita je jedním z důvodů masového rozšíření ESP32. Začátečníci mohou začít v Arduino IDE nebo MicroPythonem a postupem času přejít až k ESP-IDF pro plné možnosti využití čipu. Je také možnost tyto nástroje kombinovat, například kritické části napsat v C a pro zbytek použít MicroPython jako vrstvu pro uživatelské úpravy. Espressif navíc udržuje Arduino core jako open source projekt na GitHubu, který je interně synchronizován s novými verzemi ESP-IDF, což zajišťuje, že i Arduino prostředí postupně podporuje nové čipy (ESP32-S2/S3/C3) a jejich funkce. [75; 86]

2.5 Příklady reálného použití ESP32

Díky svým schopnostem našel ESP32 uplatnění v širokém spektru zařízení IoT a vestavěných systému. Uvádím zde několik případů jeho použití:

Chytrá domácnost – ESP32 je široce využíván v chytrých domácích zařízeních díky své bezdrátové konektivitě. Například Shelly Plus 1 integruje ESP32 pro spolehlivé ovládání osvětlení a dalších zařízení přes Wi-Fi. Sonoff POW Elite využívá ESP32 k monitorování spotřeby energie a odesílání dat do cloudu. [87; 88]

Senzorové sítě a IoT čidla – Podpora protokolu ESP-NOW umožňuje ESP32 zařízením přímou komunikaci mezi sebou pomocí Wi-Fi. To může být ideálním řešením pro senzorové sítě s nízkou latencí. Typickým příkladem jsou dataloggery, které sbírají data z čidel a odesílají je do centrály. [89]

ESP32 je vhodný pro audio aplikace díky své schopnosti zpracovávat signály v reálném čase. Deska ESP32-LyraT je navržena pro vývoj chytrých reproduktorů a podporuje streamování přes Bluetooth i Wi-Fi. [90]

Kamera a zpracování obrazu/video – Modul ESP32-CAM kombinuje ESP32 s kamerou OV2640, což umožňuje pořizování a přenos obrazu přes Wi-Fi. Je vhodný pro aplikace jako bezpečnostní kamery nebo detekce obličejů. [91]

ESP32 nachází uplatnění také v nositelných zařízeních díky své nízké spotřebě a bezdrátové konektivitě. Například chytré hodinky Spectra využívají ESP32-S3 pro sledování fitness aktivit a komunikace přes Wi-Fi a Bluetooth. [92]

Lze říci, že ESP32 je univerzální platforma pro IoT. V případech, kdy je třeba bezdrátová konektivita a slušný výpočetní výkon při zachování nízké ceny, je ESP32 častou volbou. Mnohé projekty i hobby vývojáři volí ESP32 pro své prototypy i finální produkty, jak dokládají výše zmíněné příklady z komerční sféry (Shelly Plus, POW Elite, hodinky Spectra). Vývojáři oceňují, že za dostupnou cenu získají plnohodnotný mikropočítač s jednoduchým programováním a aktivní komunitou a mohou se tak soustředit na aplikační logiku místo řešení základní konektivity.

2.6 Srovnání ESP32 s jinými mikrokontroléry

Pro zasazení do kontextu porovnejme ESP32 s jinými používanými mikrokontroléry v embedded projektech. Zmíním se zde o mikrokontrolérech z rodiny STM32, o mikročipech Arduino a také moderním Raspberry Pi Pico.

2.6.1 ESP32 vs STM32

Produkty firmy STMicroelectronics představují širokou rodinu 32-bitových MCU založených na architektuře ARM Cortex-M (M0/M3/M4/M7). Oproti ESP32, který je navržen primárně pro IoT a obsahuje integrované bezdrátové technologie, se STM32 typicky zaměřují na průmyslové a obecné embedded aplikace. Nemají vestavěné Wi-Fi/Bluetooth (s výjimkou specializovaných STM32WB s BLE), ale vynikají v jiných ohledech. Například výkonnější STM32 (řady H7) mohou mít jádra taktovaná i přes 400 MHz a dosahovat vyššího výpočetního výkonu než ESP32. Navíc jsou k dispozici varianty s matematickým koprocесорem s dvojitou přesností (double FPU), většími paměťmi (desítky MB externí SDRAM) a bohatou analogovou výbavou (ADC 16 bitů, DAC 12 bitů apod.). STM32 rovněž nabízí mnohem více periférií a rozhraní v určitých kategoriích,

například rozhraní Ethernet, USB (USB-C, 3.0), SDMMC, kamera, LCD, audio kodeky u specifických modelů. Silnou stránkou STM32 je také velký počet GPIO pinů a různé velikosti pouzder až do 144 pinů a více, což je vhodné pro komplexní systémy vyžadující připojení mnoha zařízení. Na druhou stranu, cena pokročilých STM32 MCU bývá podstatně vyšší a dostupnost v době křemíkových krizí kolísala. Pro začátečníka je vývoj na STM32 také složitější. Prostředí není jednotné jako Arduino, je třeba učit se s HAL ovladači nebo nízkourovňovými registry a práci s nástroji jako CubeMX, což představuje strmější křivku učení. Lze říci, že ESP32 vítězí v IoT scénářích svou nízkou cenou, integrovaným Wi-Fi/BLE a jednoduchým vývojem. STM32 zase vede v aplikacích vyžadující vyšší spolehlivost, více periférií a robustní reálný čas (automotive, průmysl), kde často bezdrátová konektivita není třeba nebo je řešena separátně a klíčová je stabilita a dlouhodobá dostupnost. Obě platformy se tedy spíše doplňují. Například v profesionálním návrhu by STM32 mohlo řídit systém a ESP32 sloužit jako komunikační modul. [10; 93; 94]

2.6.2 ESP32 vs Arduino

Klasické Arduino desky (Uno, Nano) jsou založené na 8-bitových mikrokontrolérech Atmega328 taktovaných na 16 MHz s 2 kB RAM. Oproti ESP32 jde o velmi jednoduchý čip, neobsahující bezdrátovou technologii, s omezenou pamětí a výkonem cca 16 MIPS. ESP32 jej překonává v podstatě ve všech směrech – pracuje na patnáctkrát vyšším taktu (240 MHz vs 16 MHz), má 250x více RAM (520 kB vs 2 kB), 125x více flash paměti (4 MB vs 32 kB) a hlavně integruje Wi-Fi/Bluetooth. Na druhou stranu Arduino má velmi nízkou spotřebu při běhu v nižších frekvencích a power save sleep módu (lze dosáhnout $<1 \mu\text{A}$ v power-down režimu) a jeho jednoduchost může být výhodou pro triviální úlohy. Pro moderní IoT projekty je však 8-bit Arduino již nedostačující a často se používá spíše jako výuková platforma. Existuje však také například deska Arduino Uno WiFi Rev.2 s mikrokontrolérem Atmega4809, která integruje modul u-blox NINA-W102 poskytující Wi-Fi a Bluetooth připojení. Novější Nano ESP32 obsahuje modul NORA-W106, který má ESP32-S3 mikročip uvnitř. Tento modul od firmy u-blox poskytuje Wi-Fi a Bluetooth připojení, poskytující Arduinu možnosti implementace v IoT oblasti. [95; 96; 97; 98; 99; 100]

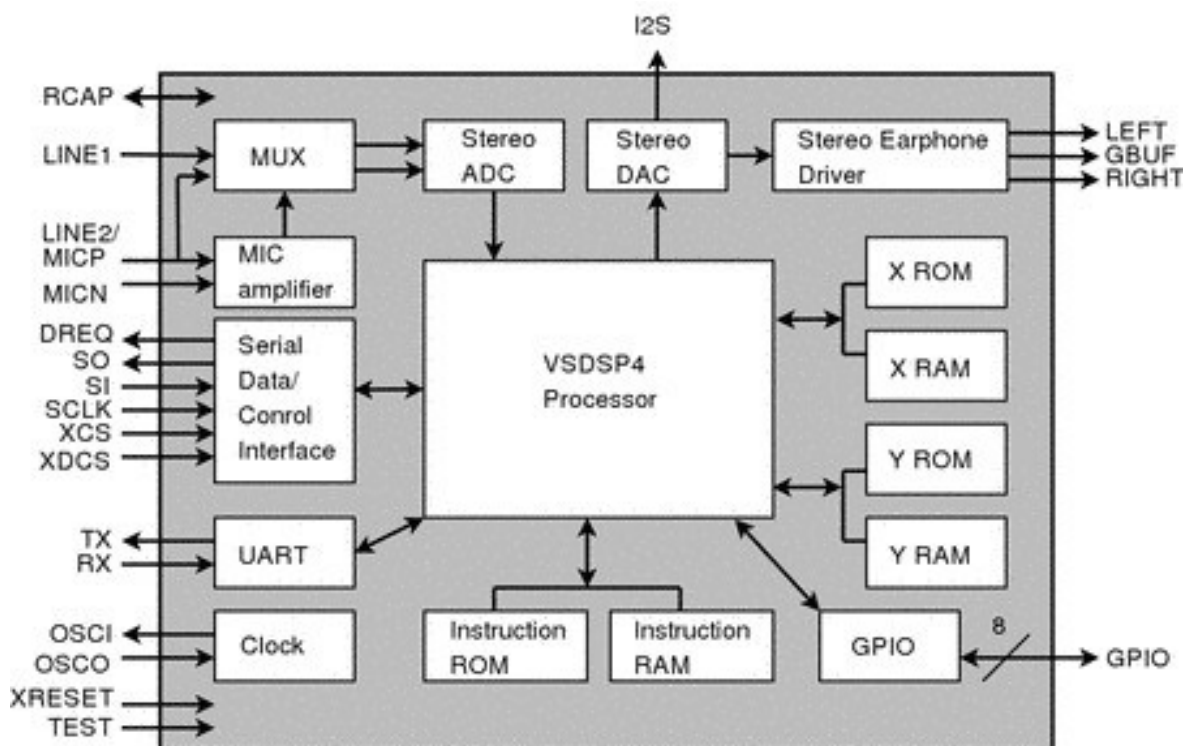
2.6.3 ESP32 vs Raspberry Pi Pico

Raspberry Pi Pico je vývojová deska uvedená v roce 2021, osazená mikročipem RP2040 vlastní konstrukce Raspberry Pi Foundation. RP2040 obsahuje dvoujádrový procesor Arm Cortex-M0+ o taktovací frekvenci 133 MHz a 264 kB SRAM. Nemá integrovanou Wi-Fi ani Bluetooth. Z hlediska výkonu je RP2040 srovnatelný s ESP32-C3, jednotlivé M0+ jádro je méně výkonné než jedno LX6 jádro ESP32-C3 (nemá DSP instrukce, FPU), ale dvě jádra a podobném taktu mu dávají určitý náskok. Klíčový rozdíl se nachází ve výbavě, kde RP2040 nabízí unikátní PIO (programovatelný I/O subsystém) pro generování libovolných protokolů a časování. To je silná stránka pro hobbyisty (např. snadná emulace VGA výstupu, ovládání LED panelů, či přesné čtení vstupních pulsů). Oproti tomu ESP32 má pevný soubor periférií, byť dost široký. Ve spotřebě je RP2040 poměrně úsporný v aktivním režimu, podobně jako ESP32, ale chybí mu extrémně úsporný deep sleep, nelze dosáhnout tak nízkých hodnot jako 10 μ A u ESP32. Moduly Pico jsou velmi levné, samotný čip je o více než polovinu levnější než samotný čip ESP32. Pokud tedy projekt nepotřebuje bezdrátovou komunikaci a vystačí si s menšími zdroji, je RP2040 ekonomickou volbou. Avšak v momentě, kdy by nastal požadavek na Wi-Fi či Bluetooth, musí se k Pico připojit externí rádiový modul, čímž se ztrácí cenová i prostorová výhoda. Z toho důvodu Raspberry Pi roku 2022 vydala variantu Pico W s přidaným Wi-Fi i Bluetooth (modul CYW43439). Lze tedy shrnout, že ESP32 je lepší pro IoT a síťové aplikace, kde dojde k využití jeho integrovaného Wi-Fi a Bluetooth možností, zatímco RP2040 je zajímavou volbou pro lokální úlohy a tam, kde chceme dosáhnout co nejnižší ceny nebo využít PIO. [101; 102]

Závěrem k porovnání lze tvrdit, že ESP32 se svými vlastnostmi (bezdrátová konektivita, vysoká integrace, výkon) vyplňuje mezeru mezi jednoduchými mikrokontroléry a plnohodnotnými Linuxovými počítači (jako Raspberry Pi Zero W). Ve srovnání s Raspberry Pi Zero W je ESP32 méně výkonný a nemá operační systém Linux, ale za to má výrazně nižší spotřebu a startuje okamžitě, což je vhodné pro real-time úlohy. Oproti mikrokontrolérům bez Wi-Fi/Bluetooth konektivity (STM32, Arduino, Pico) nabízí all-in-one řešení pro IoT projekty. Lze tedy konstatovat, že ESP32 vyniká v aplikacích, kde je potřeba bezdrátová komunikace, vícejádrový výkon a nízké náklady, zatímco v extrémních požadavcích (např. velmi precizní analogové měření nebo nutnost specializovaných periférií jako vysokorychlostní sběrnice) mohou mít jiné MCU navrch. [103]

3 AUDIO MODUL VS1053

Internetové rádio s mikrokontrolérem ESP32 využívá specializovaný dekodér zvukových formátů VS1053 (ID zařízení je VS1053b). Tento integrovaný audio kodek od firmy VLSI Solution představuje všestranný MP3 dekodér schopný hardwarově zpracovat komprimované audio formáty. Čip zvládá dekódování běžných audio formátů (MP3, AAC, Ogg Vorbis, WMA, FLAC) a obsahuje také volitelnou podporu pro MIDI syntézu či záznam zvuku. Na čipu VS1053 je integrován výkonný nízkopříkonový signálový procesor s jádrem VSDSP4, paměti a periferie, které umožňují nejen dekódování zvukových dat, ale i jejich přehrávání přes vestavěný D/A převodník a sluchátkový zesilovač. VS1053 tak funguje jako audio dekodér i audio výstupní jednotka v jednom. Po přijetí datového streamu dekóduje komprimovaný zvuk a výsledný analogový signál poskytuje na stereofonních výstupech pro další zesílení nebo přímý poslech. [11; 104]



Obrázek 2 VS1053 Blokové schéma interní architektury

(zdroj: [130])

3.1 Funkční bloky VS1053 (dekodér, DSP, I/O)

Architektura čipu VS1053 je modulární a sestává z několika hlavních bloků. Vnitřního digitálního signálového procesoru (DSP jádra) s paměťovými bloky, samotného dekodéru audio formátů (realizovaného firmwarem běžícím na DSP) a vstupně-výstupních obvodů

pro komunikaci a audio signály. Dekodér je implementován softwarově na proprietárním jádře VSDSP4 a podporuje širokou paletu audio formátů, od standardního MPEG Layer III (MP3) včetně VBR, až po moderní formáty jako AAC-LC/HE-AAC v2, WMA, či otevřený Ogg Vorbis. Dále dokáže dekódovat bezztrátový formát FLAC a nekomprimované formáty WAV/PCM. Firmware čipu umožňuje i opačnou funkci, může také enkódovat zvuk do komprimovaného formátu. VS1053 umí zaznamenávat audio (např. z mikrofону nebo linky) do formátu PCM (WAV) nebo Ogg Vorbis a to po nahrání speciálního softwarového pluginu. [11; 104; 105]

Srdcem VS1053 je DSP procesor VSDSP4 s interní pamětí programu a dat. Tento signálový procesor je optimalizován pro zpracování audia v reálném čase a přímo realizuje dekódovací algoritmy pro jednotlivé formáty. Čip obsahuje celkem 16 kB instrukční RAM a několik bloků datové RAM (X, Y RAM) pro potřeby dekodéru i případných uživatelských rozšíření. Zároveň má integrované ROM paměti, které obsahují výchozí firmware dekodérů a podpůrných rutin. Součástí DSP firmware jsou také doplňkové efekty a funkce, například prostorový efekt EarSpeaker pro sluchátka (simulace reproduktorů v místnosti) lze aktivovat pomocí příslušných bitů v registru ve třech úrovních intenzity. Dále jsou v čipu implementovány digitální basy a výšky, jednoduchý ekvalizér umožňuje zesílit či potlačit nízké a vysoké frekvence. Konfigurace probíhá zápisem do registru SCI_BASS, kde lze nastavit zesílení basů (v krocích po 1 dB) pod určitou mezní frekvenci a zesílení výšek (po 1,5 dB) nad zadanou frekvenci. Například hodnota registru 0x00F6 by znamenala posílení basů o +15 dB pod frekvencí 60 Hz. Podobně je možné ovlivnit i hlasitost, digitální regulace hlasitosti ve VS1053 má rozlišení 0,5dB na krok. Hlasitost se nastavuje 16-bitovým registrem SCI_VOL, jehož horní byte ovládá levý kanál a dolní byte pravý kanál. Čím vyšší binární hodnota, tím větší útlum (0x0000 = 0 dB, 0xFEFE = -127 dB). Nastavením SCI_VOL na 0xFFFF se analogová část úplně ztiší (power-down výstupu). Pro plynulé změny hlasitosti využívá VS1053 detekci průchodu audio signálu nulou, změna úrovně se softwarově odloží k nejbližšímu průchodu audio signálu nulou a eliminuje tak slyšitelné lupnutí. Kromě toho je implementována i ochrana sluchu, čip například podporuje postupné zesílení při zapnutí a ztišení při vypnutí (tzv. quiet-power-on/off) a umí detekovat tiché pasáže signálu. [11; 104]

VS1053 také zahrnuje různé I/O bloky pro komunikaci a připojení analogových obvodů. Pro vstup audio signálu má čip integrovaný stereofonní ADC (analogově digitální převodník) s odpovídajícím analogovým multiplexorem a zesilovačem pro mikrofonní

vstup. Uživatel tak může do VS1053 přivést buď linky LINE1/LINE2 (stereofonní analogový vstup) nebo mikrofon. Volba mezi analogovým line-in a mikrofonem se provádí přes vstupní multiplexer a konfiguruje bitem SM_LINE1 v registru SCI_MODE. ADC převádí analogový vstup do digitální podoby pro případné nahrávání. Pro výstup zvuku je na čipu integrován kvalitní stereo DAC (18-bit sigma-delta převodník). Tato analogová výstupní část poskytuje stereo signál na pinech LEFT a RIGHT vůči společné pomocné výstupní referenci GBUF. Kromě analogových vstupů/výstupů má VS1053 i digitální audio výstup, obsahuje podporu rozhraní I²S (Inter-IC Sound), které umí vyvést digitální zvuková data pro případné externí audio DAC. Pro řízení a komunikaci s hostitelským mikrokontrolérem slouží sériové rozhraní SPI ve dvou oddělených logických kanálech SDI a SCI pro zápisy/čtení řídicích registrů. Tyto komunikační linky, stejně jako několik volných GPIO pinů a sériový UART port, tvoří vstupně/výstupní vybavu čipu. UART rozhraní se využívá např. ve speciálním MIDI módu dekodéru, kdy čip může fungovat jako samostatný MIDI syntetizér. Při bootování do tzv. MIDI firmware režimu přijímá UART port MIDI příkazy a VS1053 generuje odpovídající tóny a bicí zvuky z interní banky instrumentů. Celkově je tedy VS1053 koncipován jako slave audio procesor s řadou periférií, který od hostitele (ESP32) přebírá datový audio stream a zajišťuje veškeré zpracování a výstup zvuku. [11; 104]

3.2 Komunikační protokol mezi VS1053 a ESP32

Pro spojení mikrokontroléru ESP32 s audio dekodérem VS1053 se využívá sériová synchronní sběrnice SPI. VS1053 funguje jako podřízené zařízení (SPI slave) a očekává od hostitele dva typy datových přenosů. SDI (Serial Data Interface) pro stream komprimovaných dat a SCI (Serial Control Interface) pro zápisy/čtení řídicích registrů. K tomu čip disponuje dvěma oddělenými signály pro výběr čipu (chip select signály). Pin označený XDCS (Data Chip Select) slouží k výběru režimu zápisu audio dat do dekodéru a pin XCS (Control Chip Select) slouží k zápisu/čtení ovládacích registrů čipu. Oba tyto vstupy jsou aktivní v logické nule. Díky samostatnému signálu pro příkazy lze odlišit, zda posílané byty po SPI představují datový paket hudby (MP3 frame apod.), nebo adresu a obsah řídicího registru. Komunikační rychlost lze volit v řádu jednotek až desítek MHz, po resetu je obvykle používána nižší frekvence SPI (CLKI/7) do doby, než čip povolí interní PLL násobič taktů. Poté je možné pracovat s vyšší pracovní frekvencí, maximální rychlost je dána interní frekvencí čipu (CLKI 54 MHz), pro čtení registrů je tato rychlost CLKI/7 a pro zápis dat CLKI/4. [11; 104]

Řídicí rozhraní SCI využívá standardní 16-bitové registry mapované do adresního prostoru VS1053. Čtení a zápis registrů probíhá zasláním 8-bitového operačního kódu (READ = 0x03 nebo WRITE = 0x02) a 8-bitové adresy registru, následovaným 16-bitovou datovou položkou. Tyto operace se provádějí, když je aktivní CS na pinu XCS (tj. XCS LOW). Data se posílají po vodiči MOSI (Master Out, Slave In) a případná odpověď při čtení je vrácena po MISO (Master In, Slave Out). Během komunikace platí, že data jsou vzorkována na vzestupné hraně SCK, byty se posílají MSB (nejvyšší bit) jako první. Po ukončení přenosu je nutné uvolnit příslušný CS signál do log. 1, čímž se sběrnice uvede do klidového stavu). Data pro dekodování (audio bitstream) se posílají podobně přes SDI rozhraní, s tím rozdílem, že během posílání streamu je aktivní linka XDCS. Samotný proces nahrávání dat do VS1053 je řízen pomocí signálu DREQ (Data Request). Jde o výstup z dekodéru, kterým čip oznamuje připravenost přijímat další data. Pin DREQ se nachází v log. 1 (HIGH), pokud má interní FIFO buffer VS1053 volnou kapacitu alespoň 32 B. V tu chvíli může ESP32 odesílat další blok dat. Logickou 0 (LOW) pin DREQ indikuje, že přijímací 2048-bytová FIFO paměť je téměř plná, nebo probíhá zpracování příkazu a hostitel by měl počkat s dalším zápisem dat. DREQ se o vyprázdnění bufferu opět automaticky nastaví do HIGH, takže typická obsluha na straně ESP32 vypadá tak, že čeká na DREQ == HIGH a pak pošle další 32-bytový blok datového streamu a opět vyčká až bude DREQ znovu HIGH. Tato „handshake“ metoda zajišťuje, že dekodér nikdy nepřijde o žádná data ani není přeplněn. VS1053 si sám říká o data tempem, jakým stíhá dekodovat. DREQ také krátce zůstane v 0 během zápisu do ovládacích SCI registrů, jelikož vnitřní kontrolér dočasně pozastaví plnění FIFO. [11; 104]

Pro uvedení VS1053 do výchozího stavu slouží vstup XRESET (asynchronní reset, aktivní v 0). ESP32 by měl při startu systému držet VS1053 v resetu (XRESET=0) minimálně po dobu 2 cyklů krystalu a teprve poté jej nastavit do 1. Po uvolnění z resetu si čip provede interní inicializaci (cca 22 000 taktů XTALI) a teprve jakmile VS1053 dokončí init rutiny, nastaví DREQ=HIGH, předtím nelze odesílat žádná data či příkazy. Správná posloupnost při zapínání dekodéru tedy je: aktivovat napájení, nastavit XRESET=HIGH a čekat na nastavení DREQ. Poté je čip připraven přijímat příkazy přes SCI. Typicky je vhodné nejdříve inicializovat několik základních konfiguračních registrů (mode, clock, volume apod.) než se začnou posílat data. Kromě hardwarového resetu lze VS1053 kdykoliv zresetovat i softwarově, pomocí nastavení bitu SM_RESET na HIGH

v registru SCI_MODE (bit se automaticky vynuluje). Softwarový reset se využívá například při přechodu na nový datový stream nebo při změně formátu. [11; 104]

3.3 Porovnání s jinými dekodéry/moduly

Při návrhu audio výstupu internetového rádia s ESP32 přichází v úvahu více řešení, z nichž VS1053 představuje jednu z variant (dedikovaný hardwarový dekodér s analogovým výstupem). Jednou alternativou je využít výkonu samotného ESP32 a realizovat dekódování softwarově, zatímco audio výstup zajistí jednoduchý externí D/A převodník připojený přes I²S. ESP32 je poměrně výkonný mikrokontrolér (dvoujádrový 32-bit, 240 MHz) a dokáže v reálném čase dekódovat streamy MP3 či AAC o běžném datovém toku (128-192 kbps) pomocí vhodné knihovny. Řešení založené na I²S DAC čipu (např. stereo převodník PCM5102A nebo integrovaný zesilovač MAX98357A) má výhodu v menší ceně a jednodušším hardware, vyžaduje však, aby veškerou práci dekódování zastal ESP32. To klade nároky na jeho výpočetní výkon a může omezit další běh programů (obsluha displeje, komunikace apod.). V případě internetového rádia se však ukazuje, že ESP32 dekódování zvládá. Existují projekty využívající přímo I²S port ESP32 a modul PCM5102. ESP32 v nich zpracovává většinu běžných formátů internetových stanic (MP3, AAC, WAV) a přes I²S posílá audio data do externího DAC. Kvalita výstupu pak závisí na parametrech použitého DAC, nicméně pro komprimované streamy je přínos zanedbatelný. Hlavní rozdíl je tedy v zatížení mikrokontroléru, použití VS1053 odlehčuje CPU a umožňuje i méně výkonným jednotkám přehrávat MP3. Samotný protokol I²S je navíc poněkud citlivý na časování (DMA atd.), takže volba dekodéru typu VS1053 zjednodušuje vývoj na straně softwaru. [10; 62; 106; 107; 108]

Další srovnání lze provést s jinými hardwarovými dekodéry audio formátu MP3, které byly na trhu před VS1053. Příkladem je obvod STA013 od STMicroelectronics, ten byl populární dříve jak MP3 dekodér k mikrokontrolerům. STA013 však představuje pouze dekodér dat s digitálním výstupem a vyžaduje externí DAC pro převod na výstupní analogový signál. Komunikace s STA013 je navíc řešena kombinací dvou sběrnic, pro příkazy a nastavení používá I²C, zatímco komprimovaná MP3 data se posílají zvlášť po SPI kompatibilním sériovém vstupu. Podstatný rozdíl je i v inicializaci, STA013 nemá interní ROM se vším potřebným firmware, takže při startu je nutné do něj nahrát tzv. konfigurační patch (textový soubor s cca 2000 registry), bez něhož dekodér vůbec nezačne fungovat. Oproti tomu VS1053 obsahuje potřebný firmware v sobě a po resetu je připraven

ihned dekodovat. VS1053 je také více formátový (umí navíc Ogg Vorbis, WMA, AAC atd.), kdežto STA013 byl omezen na MPEG 1 Layer III. Dalším příbuzným dekodérem je rodina čipů od VLSI Solution, například starší VS1003 a VS1033 dekodéry. Ty jsou funkčně podobné (také kombinují dekodér a DAC), ale VS1053 je překonává podporou novějších formátů a možností záznamu. Existuje i VS1063, což je následník VS1053 rozšířený o přímé nahrávání do formátu MP3 a další vylepšení. V režimu dekódování je ale prakticky shodný s VS1053. Lze ještě zmínit i specializovaný modul jako například DFPlayer Mini, který integruje mikrořadič s MP3 firmwarem a slotem na microSD. Tento modul se ale spíše hodí pro lokální přehrávání souborů z karty a nelze jej snadno použít pro streamovaná data z internetu. [109; 110; 111; 112; 113]

Z praktického hlediska vývoje jsou velmi užitečné takzvané VS1053 moduly či shields, které integrují dekodér VS1053 spolu s nutnými obvody a konektory. Například existují Arduino shiedly s VS1053, jež kromě samotného čipu nesou i slot pro microSD kartu, audio konektory (linkový vstup, mikrofon, sluchátkový výstup) a případně zesilovač. Tím získá vývojář kompletní audio přehrávač modul, stačí do něj poslat data z ESP32 a modul obstará jak dekódování, tak ukládání/čtení souborů z SD karty a výstup zvuku. Adafruit nabízí breakout board VS1053, který kombinuje dekodér s microSD slotem a podporuje nahrávání přes vestavěný elektretový mikrofon nebo linkový vstup. Tyto moduly často obsahují i drobné úpravy pro usnadnění použití. Další varianta zmíněného Adafruit VS1053 board již přímo zahrnuje velké 220 μ F vazební kondenzátory na výstupech, takže jeho „line-out“ lze rovnou připojit na externí zesilovač nebo hi-fi soustavu. Další producenti (SparkFun, Seeed a další) nabízejí obdobné moduly, někdy i s integrovaným zesilovačem pro malé reproduktory. Z pohledu parametrů zvuku jsou všechny tyto varianty s VS1053 prakticky srovnatelné, neboť všechny využívají tentýž čip. Rozdíly jsou hlavně v příslušenství a jednoduchosti integrace do projektu. Moderní trend v hobby konstrukcích směřuje spíše k využití schopností výkonných MCU (jako ESP32) pro dekódování, avšak dedikované dekodéry typu VS1053 si stále drží místo díky své spolehlivosti a odlehčení hlavního procesoru. VS1053 tak představuje ověřené řešení audio dekódování pro embedded projekty, které ob stojí ve srovnání s alternativami zejména tehdy, je-li požadována podpora většího množství formátů, analogový výstup a minimální zátěž mikrokontroléru. [114; 115]

4 NÁVRH A VÝVOJOVÉ PROSTŘEDÍ

Tato kapitola navazuje na předchozí teoretické části práce, které se věnovaly principům internetového rádia, vlastnostem mikrokontroléru ESP32 a audio dekodéru VS1053. Zaměřuje se na praktickou stránku návrhu vlastního zařízení, konkrétně na použité softwarové nástroje, návrh desky plošných spojů a volbu ovládacích prvků. Cílem této části je popsat, jakými prostředky byl realizován vývoj zařízení z hlediska návrhu desky plošných spojů, implementace firmwaru a konstrukce jednoduchého uživatelského rozhraní.

V první části je popsáno využití návrhového programu Autodesk Fusion Eagle, ve kterém byla vytvořena vlastní deska s rozmístěním všech klíčových komponentů. Následuje popis prostředí Visual Studio Code s rozšířením PlatformIO IDE, které bylo zvoleno pro programování mikrokontroléru a organizaci projektu. Kapitola se dále věnuje způsobu správy knihoven a závislostí pomocí PlatformIO, což usnadnilo vývoj a údržbu firmware. V závěru jsou popsány hardwarové prvky uživatelského rozhraní, LCD displej a ovládací tlačítka a jejich zapojení do systému.

4.1 Eagle – Návrh plošného spoje

Pro návrh plošného spoje zařízení byl použit program Autodesk Fusion Eagle (verze Fusion 2601.1.34). Eagle představuje EDA (Electronic Design Automation) nástroj pro návrh desek plošných spojů, který propojuje tvorbu schématu, umístění součástek, routování spojů a využití rozsáhlých knihoven komponentů v jednotném prostředí. Tato integrace v rámci platformy Autodesk Fusion 360 umožňuje plynulý přechod od návrhu elektronického obvodu k mechanickému designu zařízení. [116; 117]

Hlavním důvodem pro vytvoření vlastního návrhu PCB (Printed Circuit Board) byla optimalizace rozmístění prvků z hlediska ergonomie a přizpůsobení konkrétní konstrukci zařízení. Vlastní PCB umožňuje navrhnout tvar a velikost desky tak, aby vyhovovala rozměrům a tvarům použitých součástek a současně vhodně umístit ovládací prvky a konektory pro pohodlné užívání. [118]

Na výsledném PCB jsou integrovány všechny klíčové komponenty zařízení. Jako řídicí jednotka slouží mikrokontrolér ESP32. Dále je na desce osazen audio dekodér VS1053, který slouží pro dekódování streamovaného zvuku a převádí digitální audio data na analogový výstup. Návrh desky zahrnuje také napájecí konektor typu DC jack (2,1

mm), ke kterému je připojen vhodně dimenzovaný napěťový regulační obvod zajišťující stabilní napájení celého zařízení. Pro ovládání displeje byl integrován I²C převodník pro řadič HD44780, který umožňuje připojení standardního znakového LCD displeje 16x02 s minimálním počtem vodičů. Z důvodu rozdílných logických úrovní mezi jednotlivými moduly byl na desce použit také převodník logických úrovní typu HW-024, který zajišťuje spolehlivou komunikaci mezi ESP32 a I²C převodníkem. Nechybí ani sada uživatelských uživatelských tlačítek, jež slouží k interakci s grafickým rozhraním a ovládání funkcí zařízení. [119; 120; 121]

Při návrhu byly použity dostupné knihovny součástek jež Eagle nabízí pro vložení běžných součástek jako jsou kondenzátory, diody a tlačítka. Pro ESP32 modul, VS1053 dekodér, HW-024 převodník a LCD modul s I²C však bylo nutné vytvořit vlastní knihovnu. V rámci Eagle to znamená navrhnout nový device s odpovídajícím schématickým symbolem a footprintem. Tvorba vlastních součástek klade důraz na přesné nastudování parametrů z datasheetu a pečlivé umístění vývodů a padů tak, aby výsledný plošný spoj odpovídal fyzické součástce. Díky tomu má navržené PCB všechny potřebné komponenty ve správném tvaru a rozměru. [116; 122]

4.2 Vývoj firmware ve Visual Studio Code

Pro vývoj firmware bylo použito prostředí Visual Studio Code rozšířené o plugin PlatformIO IDE. Toto prostředí přináší oproti klasickému Arduino IDE výrazné výhody zejména u větších projektů. VS Code je moderní multiplatformní editor podporující mnoho programovacích jazyků a díky rozšíření PlatformIO se z něj stává plnohodnotné vývojové IDE pro embedded zařízení. Na rozdíl od jednoduchého Arduino IDE (zaměřeného spíše na menší projekty) nabízí PlatformIO ve VS Code širší podporu mikroprocesorových platforem (kromě ESP32 také Arduino, STM32, AVR a další) a mnoho pokročilých funkcí, jako je správa knihoven, jednotkové testování či debugging. Prostředí je tak vhodné pro komplexnější projekty, kde jsou potřeba robustnější nástroje. [123; 124]

Významnou předností kombinace VS Code + PlatformIO je přehledná organizace projektu. PlatformIO automaticky generuje strukturu složek, například adresář „src/“ pro zdrojové kódy, „include/“ pro hlavičkové soubory, „lib/“ pro případné vlastní knihovny a také konfigurační soubor platformio.ini, který se nachází v kořenovém adresáři projektu. Tato struktura udržuje projekt modulární a přehledný, což je u větších projektů nezbytné. Veškeré nastavení projektu se soustředí do zmíněného souboru platformio.ini, kde je

specifikována cílová platforma, deska, framework a další závislosti a nastavení. Například v této práci je v konfiguraci uvedeno použití frameworku Arduino pro ESP32, což PlatformIO interně zajistí načtením potřebných nástrojů a knihoven pro Arduino Core ESP32. Dále jsou v souboru deklarovány potřebné knihovny, konkrétně ESP_VS1053_Library (pro obsluhu audio dekodéru VS1053) a LiquidCrystal_I2C_ESP32 (pro řízení LCD displeje přes I²C), aby je PlatformIO mohlo automaticky stáhnout a zahrnout při kompilaci. Díky této centralizované konfiguraci odpadají manuální zásahy do nastavení překladače či ruční správa knihoven, což urychluje práci na firmware. [123; 125; 126]

Další výhodou je komfortní nahrávání kódu a možnost ladění přímo z prostředí VS Code. PlatformIO dokáže automaticky detekovat připojené vývojové desky a při spuštění akce „upload“ přeloží firmware a nahraje jej do zařízení přes sériový převodník bez nutnosti opouštět editor. Oproti Arduino IDE, které postrádá pokročilé ladící nástroje, nabízí PlatformIO i debugger. Při použití debuggeru lze program i krokovat, používat breakpointy a sledovat hodnoty promenných v reálném čase. Tyto funkce usnadňují odhalování chyb a podrobné testování firmware. Integrovaný serial monitor umožňuje rovněž pohodlně sledovat ladící výstupy ze zařízení. Celkově tak spojení VS Code a PlatformIO poskytuje moderní vývojové prostředí, které zvyšuje produktivitu a kvalitu vývoje firmware v porovnání s tradičním Arduino IDE. [123; 124]

4.3 PlatformIO

PlatformIO vyniká nad standardním Arduino ekosystémem především propracovanou správou závislostí projektu, flexibilní volbou cílové platformy a výkonným build systémem. V embedded projektech je běžné, že firmware využívá řadu externích knihoven. Ruční správa těchto závislostí však bývá náchylná k chybám. PlatformIO Library Manager tento problém řeší centralizovaně, umožňuje deklarovat knihovny potřebné pro projekt a automaticky se stará o jejich stažení, aktualizace i udržování správných verzí. V praxi to znamená, že autor v konfiguraci platform.ini uvede v sekci lib_deps seznam požadovaných knihoven a PlatformIO je před překladem automaticky stáhne z PlatformIO Registry nebo z GitHub repozitáře. Takto byly do projektu přidány i zmíněné knihovny ESP_VS1053_Library a LiquidCrystal_I2C_ESP32. Při prvním build procesu je PlatformIO samo dohledalo online a instalovalo do izolovaného adresáře projektu. Díky izolaci závislostí pro každý projekt nehrozí konflikt verzí knihoven mezi různými projekty.

Aktualizace knihovny/knihoven v jednom projektu neovlivní funkčnost jiného projektu. Tento vestavěný mechanismus správy závislostí patří k typickým rysům moderních softwarových ekosystémů a PlatformIO jej jako jedno z mála embedded IDE plně podporuje. [123]

Konfigurační soubor PlatformIO rovněž jednoznačně určuje cílovou platformu projektu. V našem případě je platformou espressif32 a board nastaven na konkrétní model vývojové desky s ESP32. PlatformIO na základě těchto údajů automaticky volí příslušný překladač, kompilátory a linker skripty, které odpovídají danému čipu a zahrnuje potřebné frameworky (Arduino Core pro ESP32). To vývojáři značně usnadňuje práci, podobně jako s knihovnami lze odpadá ruční konfigurace nástrojového řetězce pro každou platformu. Současně je možné v jednom projektu definovat více environment (prostředí) pro různé desky či varianty firmware a PlatformIO pak umožňuje přeložit kód pro každou z nich jen přepnutím cílového prostředí. Tato flexibilita kontrastuje s Arduino IDE, které je tradičně vázáno primárně na desky Arduino a vyžaduje dodatečné instalace core balíčků pro jiné platformy. Naproti tomu PlatformIO od základu podporuje širokou škálu architektur od AVR přes ESP až po STM32 atd.. [123; 124]

4.4 Uživatelské rozhraní zařízení

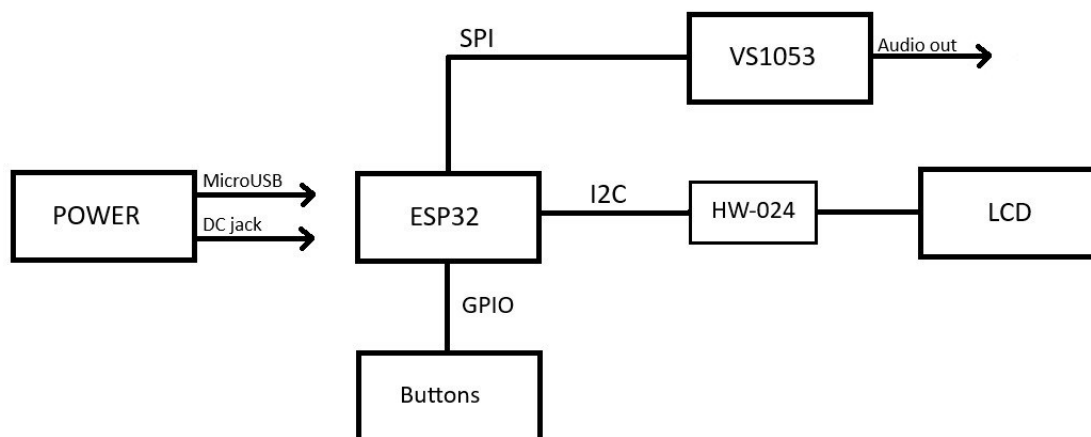
Pro interakci s uživatelem byl v zařízení implementován jednoduchý znakový LCD displej a pětice tlačítek. Displej typu LCD1602A (16 znaků a 2 řádcích) zobrazuje základní informace, například zobrazení menu, čas nebo aktuálně přehrávanou stanici. Tento modul využívá převodník I²C s ovladačem HD44780 a byl zvolen zejména pro svou nenáročnost a dobrou čitelnost v různých podmínkách. Pro úsporu pinů mikrokontroléru je LCD připojen přes sériové rozhraní I²C pomocí převodníku s čipem PCF8574. Místo přímého paralelního propojení, které by zabralo minimálně 6 GPIO pinů tak stačí pouze dvojice vodičů I²C sběrnice (SDA a SCL) pro ovládání všech datových i řídicích linek displeje. Čip PCF 8574 funguje jako 8-bitový I/O expandér, převádí povely z I²C linky na změny stavu svých 8 výstupních pinů, které jsou připojeny na vstupy HD44780 řadiče. Displej je tak provozován v 4-bitovém módu přes I²C expander, což výrazně zjednodušuje zapojení. Další výhodou tohoto řešení je existence hotových knihoven, které komunikaci s displejem přes expander podporují. Programování z pohledu vývojáře poté zůstává velmi podobné, jako by ovládal LCD přímo, avšak fyzicky využívá I²C protokol. [120; 121; 127]

Pro ovládání zařízení je použito 5 tlačítek, rozmístěných do kříže (směry nahoru, dolů, vlevo, vpravo a středové tlačítko OK). Tato sada odpovídá rozložení známému z běžných LCD keypad shieldů pro Arduino a poskytuje základní možnosti navigace a ovládání. Uživatel pomocí šipkových tlačítek prochází položky nabídky na displeji a tlačítko OK potvrzuje. Takové uspořádání umožňuje implementovat intuitivní menu systém, ve kterém se lze jednoduše přepínat mezi položkami či obrazovkami. V návrhu hardware jsou všechna tlačítka připojena na digitální vstupy ESP32 přes interní pull-up rezistory. Firmware následně čte stav těchto pinů a vyhodnocuje stisky. Ošetřeno je také odstraňování zákmitů tlačítek v kódu (debounce ochrana) a případné opakované posílání povelů při delším stisku. [54; 128]

Zvolená kombinace znakového LCD displeje a pěti tlačítek představuje osvědčené uživatelské rozhraní pro jednoduchá embedded zařízení. Výhodou je nízká cena a snadná implementace. LCD1602A s I²C rozhraním neklade nároky na piny mikrokontroléru a mechanická tlačítka umožňují přesné ovládání bez nutnosti dotykového displeje či dalších hardwarových komponent. Toto rozhraní postačuje pro základní interakci uživatele s internetovým rádiem, při zachování potřebné funkcionality a uživatelského komfortu.

PRAKTICKÁ ČÁST

5 HARDWAROVÝ NÁVRH MODULU



Obrázek 3 Blokové schéma internetového rádia

(zdroj: vlastní)

Na obrázku č. 3 je znázorněno blokové schéma navrženého modulu internetového rádia. Z něj je patrné, jakým způsobem jednotlivé komponenty vzájemně komunikují a jakou funkci v systému plní.

Základní řídicí jednotkou celého systému je vývojová deska ESP32, která zajišťuje veškeré řízení logiky, zpracování dat a připojení k bezdrátové síti Wi-Fi. Prostřednictvím rozhraní SPI komunikuje s audio dekodérem VS1053, kterému předává streamovaná data pro dekódování, jenž je následně přehraje na zvukovém výstupu. Pro zobrazení informací o aktuálním stavu systému je použit LCD displej o rozměrech 16x2 znaků, které je připojeno přes I²C sběrnici. Jelikož displej vyžaduje logické úrovně 5 V, zatímco výstupy z ESP32 pracují s úrovněmi 3,3 V, je mezi nimi použit napěťový převodník typu HW-024.

Ovládání uživatelského rozhraní je realizováno pomocí pěti tlačítek, která jsou připojena přímo na digitální vstupy ESP32. Ta umožňují pohyb v menu, výběr rádiové stanice a další funkce systému. Napájení celého systému je řešeno buďto přes microUSB umístěném na ESP32 modulu, nebo je k dispozici DC Jack 2,1 mm s napěťovým regulátorem, který zajišťuje stabilní napětí vhodné pro jednotlivé komponenty. Kromě toho je ve schématu vyznačen také sluchátkový výstup, do kterého je vyveden dekódovaný audio signál z modulu VS1053.

5.1 Výběr a zapojení komponentů

Při návrhu modulu internetového rádia byl kladen důraz na dostupnost komponent, jednoduchost integrace a podporu potřebných funkcí. Následující část popisuje jednotlivé zvolené součásti a jejich konkrétní zapojení do systému.

Řídící jednotkou celého zařízení je vývojová deska ESP32 DevKit V1 osazená modulem ESP32-WROOM-32. Tato vývojová deska byla zvolena díky své vysoké integraci, podpoře bezdrátového připojení přes Wi-Fi, dostatečnému počtu vstupně-výstupních pinů a široké podpoře v komunitě. Kromě toho nabízí dostatečný výpočetní výkon pro řízení uživatelského rozhraní, obsluhu připojených periférií a zpracování streamovaných dat.

Pro dekódování internetového streamu byl zvolen breakout modul VS1053 s čipem VS1053b od firmy VLSI Solution. Tento čip umožňuje hardwarové dekódování zvukových formátů jako jsou MP3, AAC, Ogg Vorbis nebo WAV, čímž významně snižuje zátěž hlavního mikrokontroleru. Mezi jeho výhody patří integrovaný D/A převodník s analogovým výstupem a možnost přímého připojení sluchátek. Komunikace mezi ESP32 a modulem VS1053 probíhá pomocí protokolu SPI, konkrétně s následujícím přiřazením pinů:

- D5 – CS
- D19 – MISO
- D23 – MOSI
- D18 – SCK
- D27 – XCS (Command Select)
- D26 – XDCS (Data Select)
- D33 – XRESET (reset čipu)
- D34 – DREQ (Data Request)

Uživatelské prostředí je tvořeno LCD displejem 1602A, který využívá rozhraní I²C pro minimalizaci počtu potřebných vodičů. Pro převod signálů mezi 3,3 V logikou ESP32 a 5 V logikou převodníku I²C byl použit konvertor úrovní typu HW-024. Zapojení signálových vodičů je následující:

- D21 (BSDA) – SDA (ASDA)

- D22 (BSCL) – SCL (ASCL)

Uživatelské ovládání je realizováno pomocí pěti tlačítek, která slouží pro navigaci v menu a potvrzení výběru. Tlačítka jsou zapojena přímo na digitální vstupy mikrokontroleru a využívají interní pull-up rezistory. Přehled zapojení:

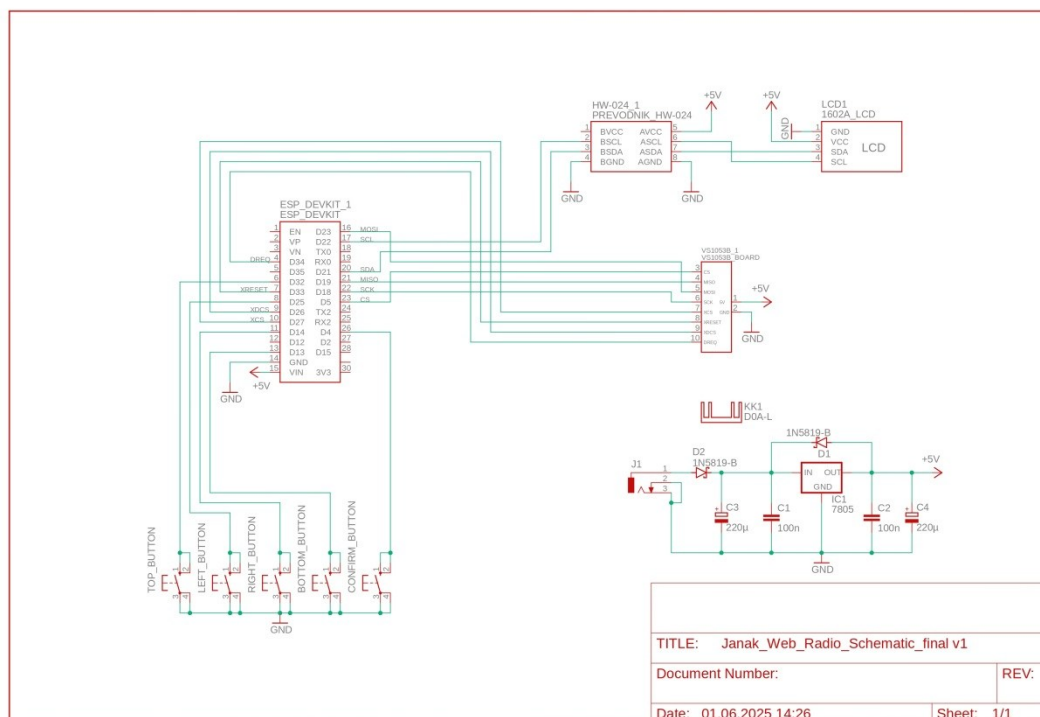
- D32 – tlačítko Nahoru
- D13 – tlačítko Dolů
- D25 – tlačítko Vlevo
- D14 – tlačítko Vpravo
- D4 – tlačítko OK

Napájení zařízení je možné dvěma způsoby. Prvním je připojení přes microUSB konektor na vývojové desce ESP32, druhým pak pomocí externího napájecího adaptéru přes DC konektor typu 2,1 mm. V tomto případě je napětí stabilizováno pomocí lineárního regulátoru typu 7805, který snižuje vstupní napětí na hodnotu 5 V, vhodnou pro napájení připojených periférií.

Zvolená kombinace komponentů zajišťuje optimální poměr mezi funkčností, jednoduchostí zapojení a cenovou dostupností. Všechny prvky jsou běžně dostupné a vhodné pro vývoj funkčního prototypu internetového rádia.

5.2 Návrh zapojení

Návrh zapojení celého zařízení byl realizován v softwaru Autodesk Eagle, který umožňuje vytvářet jak schémata zapojení, tak návrhy desek plošných spojů (DPS). Výsledkem práce jsou dva základní dokumenty, to je schéma zapojení (schematic) a dvouvrstvý návrh DPS (top a bottom vrstvy).

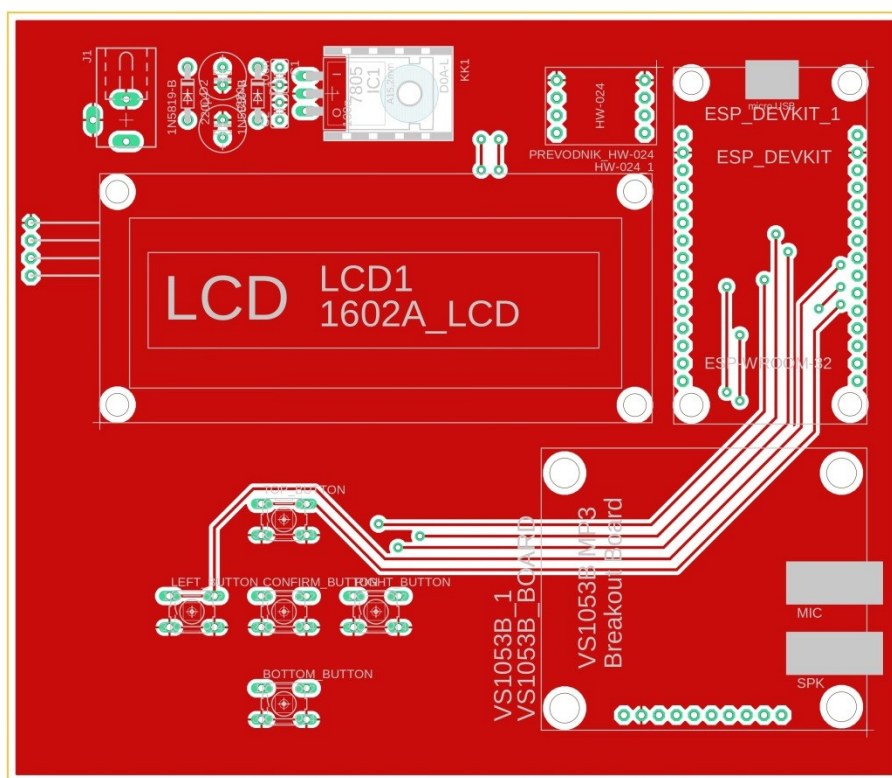


Obrázek 4 Eagle Schematic

(zdroj: vlastní)

Na obrázku č. 4 je zobrazeno kompletní schéma zapojení celého systému. Schéma obsahuje všechny základní komponenty použité v zařízení. Řídící jednotku ESP32, zvukový dekodér VS1053, LCD displej s převodníkem HW-024, pět ovládacích tlačítek a napájecí obvod s regulátorem napětí typu 7805. Na schématu lze vidět, že komunikace mezi ESP32 a modulem VS1053 probíhá po sběrnici SPI, zatímco pro připojení LCD je využito I²C rozhraní s převodníkem úrovní. Tlačítka jsou zapojena jako spínané vstupy vůči zemi a využívají interní pull-up rezistory mikrokontroleru. Schéma také obsahuje část napájecího obvodu, včetně ochranných diod, filtračních kondenzátorů a samotného lineárního stabilizátoru.

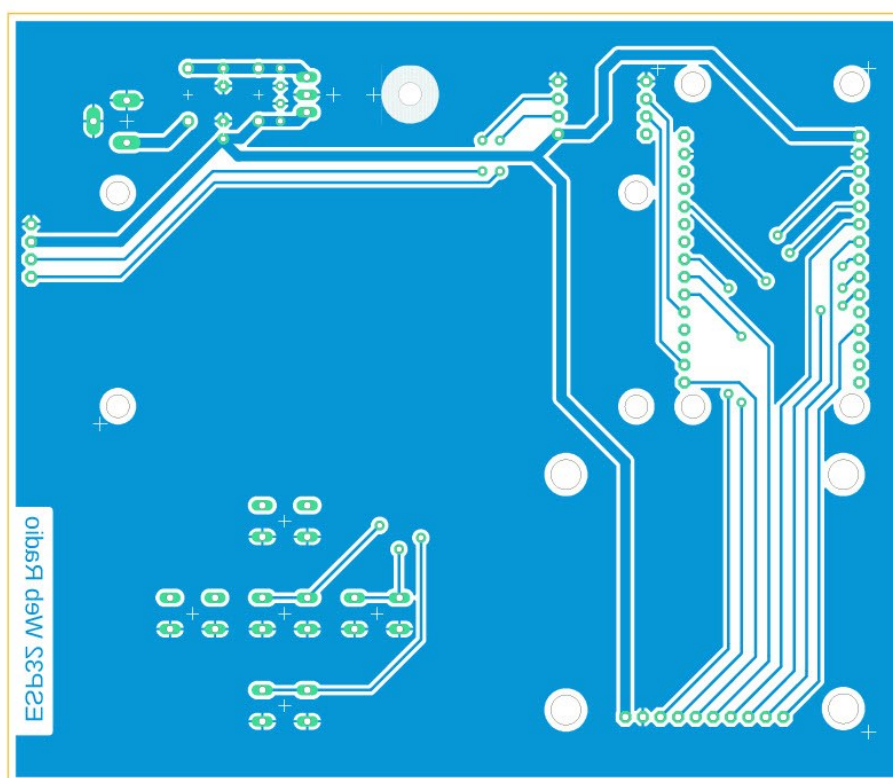
Pro účely sestavení funkčního prototypu byla navržena a vyrobena dvouvrstvá deska plošných spojů, jejíž návrh je zobrazen a popsán na dalších dvou obrázcích.



Obrázek 5 Eagle Board Top Layer

(zdroj: vlastní)

Vrchní strana DPS obsahuje všechny součástky, které tento návrh obsahuje. Vedení signálových a napájecích cest je rozvrženo s ohledem na přehlednost a minimalizaci délky kritických spojů. Popisy součástek jsou kvůli prostorovým omezením místy překryté, avšak na samotnou funkci návrhu to nemá vliv.



Obrázek 6 Eagle Board Bottom Layer

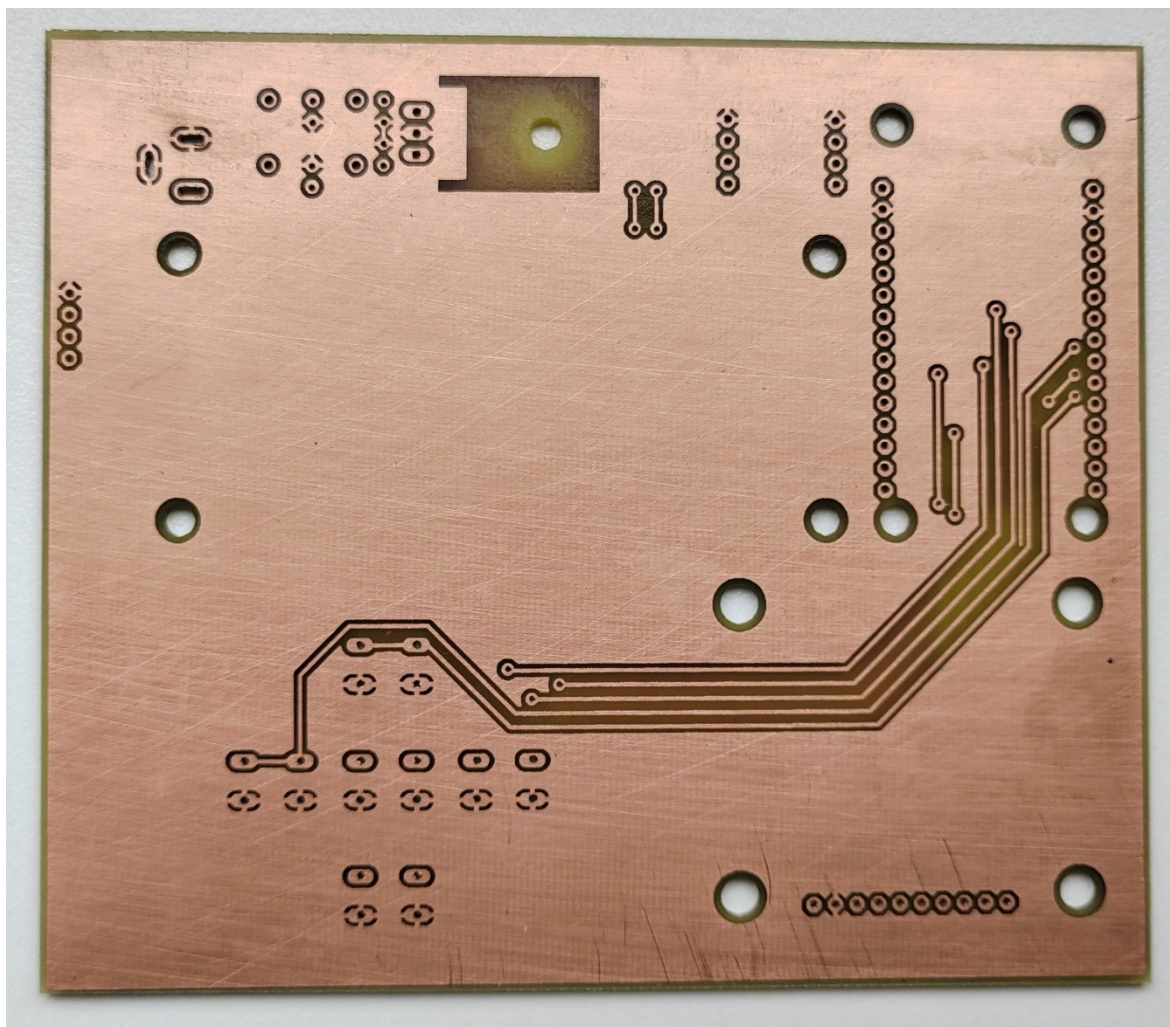
(zdroj: vlastní)

Spodní strana DPS slouží zejména pro vedení doplňkových signálových cest a přemostění míst, kde by se cesty na jedné vrstvě křížily. K tomu byly využity průchozí otvory (vias), které umožňují přechod mezi jednotlivými vrstvami. Celková koncepce rozložení součástek a vedení součástek byla navržena s důrazem na praktickou realizovatelnost a správnou funkci. Díky použití dvouvrstvého návrhu bylo možné dosáhnout kompaktnějšího rozvržení a zkrácení délky některých kritických signálových spojů.

Navržená deska byla následně vyrobena a použita při sestavení funkčního prototypu internetového rádia. Ověření správné funkce jednotlivých spojů proběhlo při osazování a testování zařízení, jak bude popsáno v následujících kapitolách.

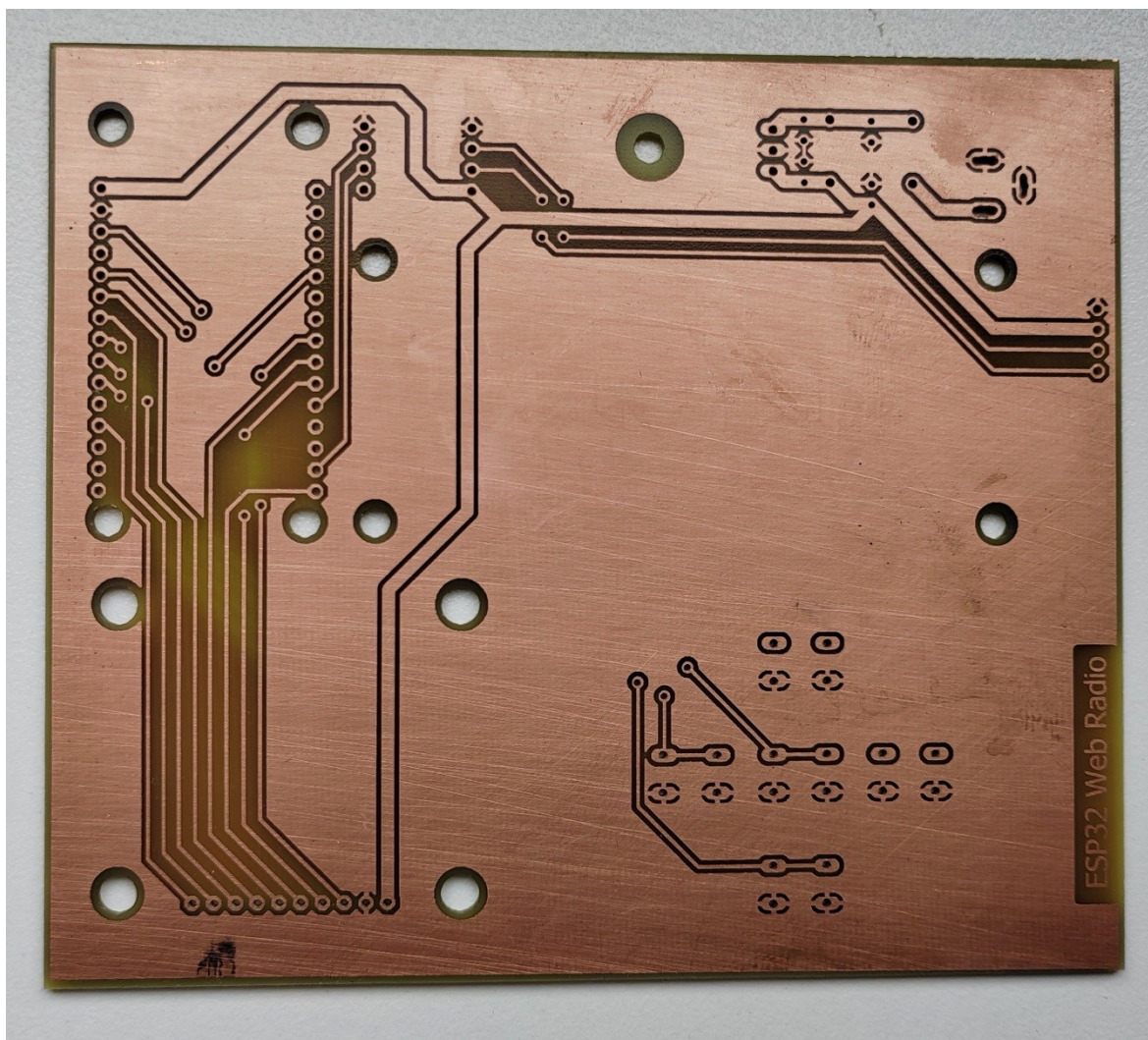
6 REALIZACE MODULU

Po dokončení návrhu zapojení a DPS bylo přistoupeno k samotné realizaci fyzického modulu. Návrh byl exportován a předán k výrobě ve formě dvouvrstvé desky s měděnými spoji, jak je patrné z přiložených fotografií. První dva snímky zachycují horní a spodní stranu vyrobené DPS ve stavu před osazením.



Obrázek 7 DPS Top

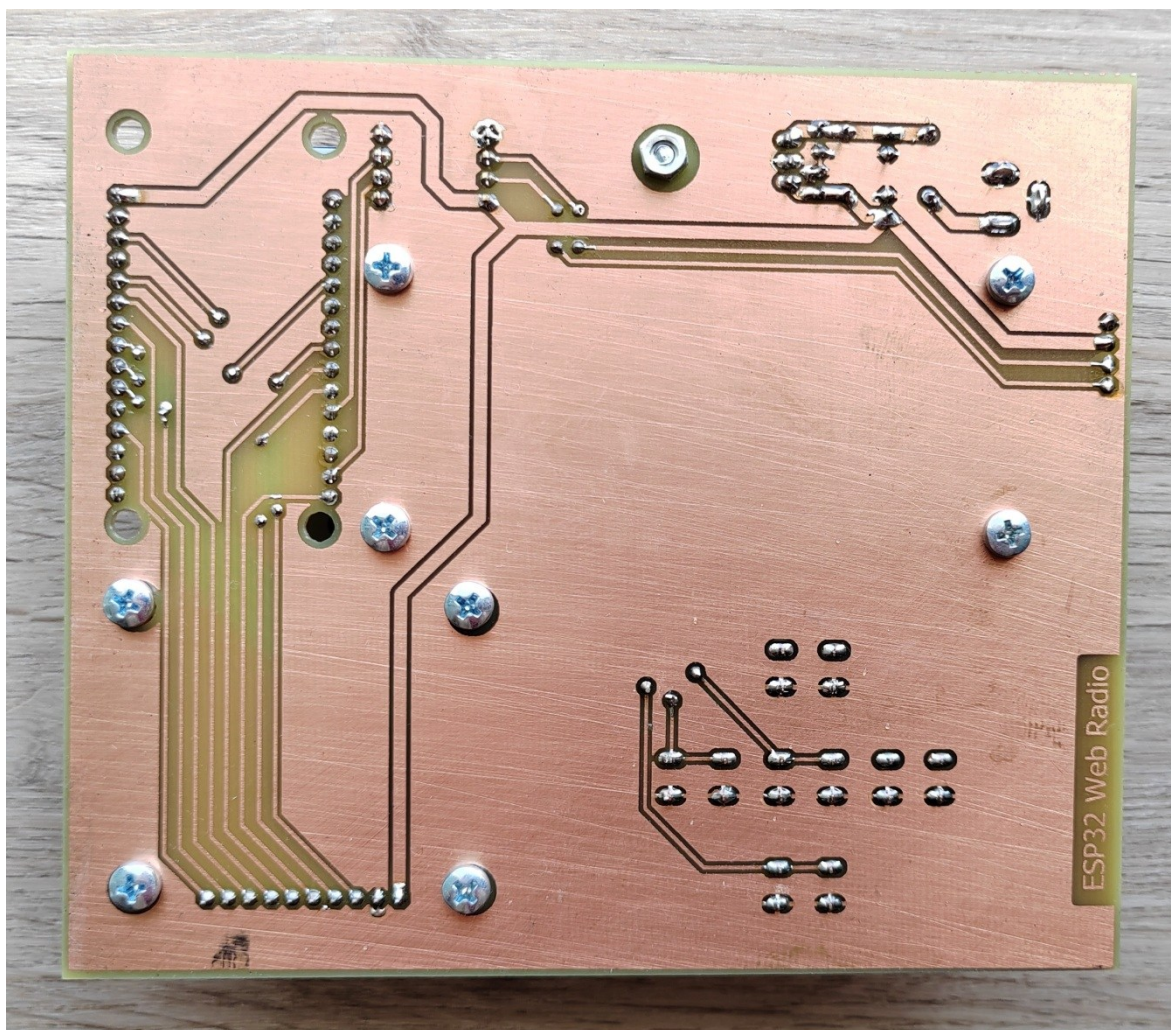
(zdroj: vlastní)



Obrázek 8 DPS Bottom

(zdroj: vlastní)

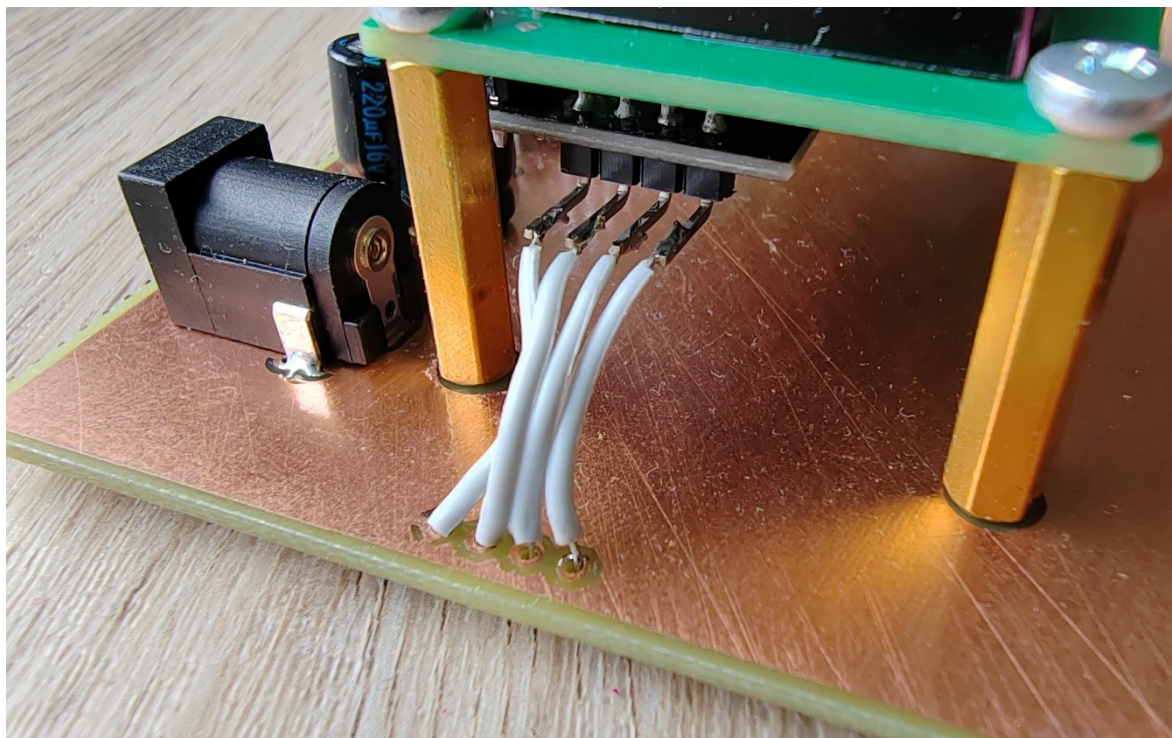
Osazení součástek na desku bylo provedeno ručně pomocí klasické pájecí stanice a ruční bodové páječky. Během procesu byly jednotlivé komponenty umísťovány na své pozice dle návrhového výkresu (viz. příloha). Pájení probíhalo standardním způsobem, přičemž byly dodrženy zásady elektroinstalace a bezpečnosti práce. Uzemnění, vstupní napájecí větve a signálové spoje byly po dokončení pájení ověřeny pomocí digitálního multimetru. Kontrolovala se především vodivost, přítomnost zkratu a správná polarita u napájecího konektoru a stabilizátoru.



Obrázek 10 Internetové rádio bottom

(zdroj: vlastní)

Na výsledných fotografiích je možné vidět kompletně osazenou desku, která obsahuje všechny komponenty uvedené v návrhu.



Obrázek 11 Propojení I²C převodníku pomocí vodičů

(zdroj: vlastní)

Při osazování bylo nutné vyřešit jednu drobnou komplikaci, s kterou se ale počítalo již při návrhu desky samotné. Piny převodníku I²C jsou příliš krátké pro přímé zapájení do desky. Tento problém byl vyřešen připojením převodníku pomocí vodičů, které byly připájeny z jeho pinů přímo do připravených otvorů v DPS. Tímto způsobem byla zachována funkčnost obvodu i bez mechanického připevnění modulu k desce.

Konstrukce celého zařízení byla navržena s ohledem na kompaktní rozmístění komponentů a mechanickou stabilitu. Deska byla opatřena montážními otvory pro přišroubování modulů, čímž byla zajištěna jejich pevná fixace. Výsledná sestava je plně funkční a připravena k dalšímu programování a testování, kterým se věnují následující kapitoly práce.

7 PROGRAMOVÉ VYBAVENÍ MODULU

7.1 Vývojové prostředí a použité knihovny

Vývoj programového vybavení modulu internetového rádia probíhal ve vývojovém prostředí Visual Studio Code s rozšířením PlatformIO, které umožňuje pohodlnou správu projektu, knihoven a cílových platforem. Jako cílová platforma byla zvolena Espressif32. Pro práci s jednotlivými periferiemi a funkcemi systému byly využity následující knihovny:

- ESP_VS1053_Library ve verzi 1.1.4 od autora Baldram sloužící pro komunikaci s dekodérem VS1053.
- LiquidCrystal_I2C_ESP32 ve verzi 1.1.6 od autora Iakop sloužící pro ovládání I²C LCD displeje s čipem PCF8574.
- Interní knihovny WiFi.h, WiFiClient.h a time.h použité pro zajištění síťové konektivity, přenosu dat a synchronizace času pomocí SNTP.
- LittleFS.h pro práci s interním souborovým systémem ESP32.
- vector pro správu dynamických seznamů položek ve složkách.

Toto prostředí poskytovalo dostatečný komfort pro strukturovaný vývoj, ladění i testování vestavěného firmwaru zařízení.

7.2 Architektura programu

Celý program je strukturován na základě stavového řízení, kde se zařízení nachází v jednom z několika základních stavů aplikace (AppState), reprezentovaných výčtovým typem. Mezi tyto stavy patří:

- STATE_MENU – hlavní menu programu
- STATE_RADIO – přístup ke složkám a souborům se stanicemi
- STATE_TIME – zobrazení hodin a nastavení budíku
- STATE_ALARM – režim aktivního budíku

Řízení je implementováno skrze globální proměnnou currentState, která se přepíná podle uživatelského vstupu a okolností v programu. V hlavní smyčce loop() se podle aktuálního stavu volají příslušné funkce pro obsluhu tlačítek (handleStateMenuButtons,

handleStateRadioButtons atd.) a vykonávají se specifické akce, například zobrazení času, streamování nebo přehrávání alarmu.

Kód je rozdělen do logických bloků a využívá modulární přístup pomocí funkcí, z nichž některé stojí za bližší zmínku:

- processButton() – komplexní obsluha vstupních tlačítek s detekcí krátkého a dlouhého stisku a implementací debounce ochrany.
- handleItemTypeSelection() – obsluha výběru položky a případně zavolání funkce pro spuštění streamu.
- startRadio() – zajištění připojení ke streamu, s pomocí dalších funkcí.
- processStream() – zpracování příchozích audio dat a předávání do dekodéru.
- updateIdleDisplay() – cyklické zobrazování informací na LCD displeji.
- iniTime() – synchronizace času pomocí SNTP a NTP serveru.
- toggleFavourite() – přidávání nebo odebrání stanic do složky Favourites.

Nastavení jednotlivých komponent probíhá v rámci funkce setup(), kde se provádí inicializace sériové komunikace, displeje, VS1053 modulu, připojení k Wi-Fi, synchronizace času a připojení souborového systému. Vše je podmíněno kontrolní proměnnou initOk, která v případě chyby pozastaví běh zařízení a zobrazí chybové hlášení na LCD.

7.3 Funkce a jejich implementace

Tato kapitola podrobně rozebírá jednotlivé funkce, které tvoří programové jádro modulu internetového rádia. Každá funkcionálita byla navržena s ohledem na omezené zdroje vestavěného systému a současně s důrazem na uživatelskou přívětivost a stabilitu provozu.

7.3.1 Zpracování vstupu z tlačítek

Základním prvkem ovládání zařízení je pětice mechanických tlačítek (Nahoru, Dolů, Vlevo, Vpravo, OK), které slouží k navigaci v menu, výběru stanic a obsluhy dalších funkcí. Pro zpracování jejich vstupu byla vytvořena funkce processButton(), která detekuje jak krátké kliknutí (event CLICK), tak dlouhý stisk (event HOLD).

Tato funkce funguje na principu časového měření stisknutí tlačítka:

- V okamžiku detekce stisku se uloží čas pomocí millis().

- Následně se kontinuálně vyhodnocuje, zda je tlačítko stále drženo.
- Pokud je tlačítko uvolněno dříve, než uplyne definovaný práh (`HOLD_THRESHOLD = 750 ms`), je vyhodnocen krátký stisk.
- Pokud je překročen časový práh, jedná se o dlouhý stisk, což umožňuje přidat další vrstvu ovládání.

Součástí funkce je také ochrana proti zákmitům kontaktů (debounce), která eliminuje falešné stisky způsobené mechanickým chvěním kontaktu při zmáčknutí. Díky tomu je ovládání stabilní a dobře reaguje i při opakovaném mačkání.

Výsledek této funkce je vrácen ve formě datové struktury `ButtonEvent`, která nese informaci o typu události pro dané tlačítko. Tato struktura je následně zpracovávána v hlavní smyčce programu v závislosti na aktuálním stavu systému (`currentState`). Funkce `processButton()` tak tvoří klíčový prvek celé logiky programu, jelikož skrze ni je zajištěno veškeré ovládání systému od uživatele.

7.3.2 Spuštění přehrávání internetového rádia

Klíčovým bodem softwarové části je také spuštění přehrávání internetového streamu. Tuto činnost zajišťuje funkce `startRadio(const char* url)`, která s pomocí funkce `readRadioInfo()` obdrží URL adresu cílového streamu (např. ve formátu `http://uk7.internet-radio.com:8040/`). Funkce se následně postará o kompletní síťovou komunikaci, vyhodnocení odpovědi serveru, detekci metadat a případné řešení přesměrování.

1. Detekce playlistu (.pls)

Pokud zadaná URL adresa končí příponou `.pls`, jedná se o tzv. playlistový soubor (formát používaný zejména v systému Shoucast/Icecast). V tomto případě je pomocí podpůrné funkce `fetchPlsStream()` provedeno jeho zpracování a následné zjištění skutečné adresy streamu, kterou obsahuje. Funkce se následovně rekurzivně volá znovu s touto novou adresou.

2. Parsování URL adresy

Po případném zpracování playlistu přistupuje funkce `startRadio()` ke klasickému rozboru zadané adresy URL. Ze zadaného řetězce se extrahuje:

- komunikační protokol (např. `http`)
- název hostitele (`host`) a port (standardně `80`)

- cesta (path) k požadovanému streamovacímu zdroji

3. Navázání spojení a HTTP požadavek

Po rozboru adresy se ESP32 pokusí navázat TCP spojení se vzdáleným serverem. Pokud je spojení úspěšné, odešle zařízení vlastní HTTP požadavek typu GET. V tomto požadavku jsou zahrnuty důležité hlavičky:

- Host – identifikace serveru
- User-Agent – označení klienta (zde jako EPS32)
- Icy-MetaData:1 – požadavek na posílání metadat
- Connection: close

4. Čtení hlaviček a interpretace odpovědi

Po odeslání požadavku přechází ESP32 do fáze čtení http hlaviček. Zde funkce:

- kontroluje stavový kód odpovědi (např. 200 OK, 302 Redirect, apod.)
- extrahuje název stanice (icy-name)
- zjišťuje bitrate (icy-br) a interval metadat (icy-metaint)
- v případě potřeby detekuje přesměrování (např. 302 Found + hlavička Location) a zpracuje jej opětovným GET požadavkem

Ve zvláštním případě, kdy server vrací Content-Type: text/html, což značí, že nebyl nalezen platný stream (ale např. chybová schránka) funkce provede tzv. mount-point fallback a pokusí se o alternativní přístup přes cestu „/;?sid=1“ běžně používanou u Shoutcast serverů.

5. Zahájení přehrávání

Pokud je vše úspěšně navázáno a server odpovídá očekávaným způsobem, funkce nastaví stavovou proměnnou isStreaming na hodnotu true, čímž oznamuje hlavní smyčce programu, že je možné začít přijímat a předávat data do dekodéru VS1053.

7.3.3 Zpracování audio streamu

Po úspěšném navázání spojení se streamovacím serverem se mění proměnná isStreaming na hodnotu true. Tato změna způsobuje, že hlavní smyčka loop() opakovaně volá funkci processStream(). Tato funkce představuje jádro běhu internetového rádia, zajišťuje

přijímání audio dat, jejich odesílání do dekodéru VS1053 a současně zpracovává metadata zasílaná serverem.

Funkce nejprve zjišťuje, zda je stále aktivní spojení se serverem (`wifiClient.connected()`). Pokud dojde k ukončení spojení, zobrazí se uživateli upozornění na LCD displeji a stream je ukončen pomocí `stopStream()`. Následně je vypsána položka (název stanice), která byla naposledy zvolena v menu.

Hlavní část funkce je tvořena smyčkou `while`, která běží dokud:

- jsou dostupná data na síťovém klientovi (`wifiClient.available()`)
- a dokud je dekodér VS1053 připraven přijímat další data (DREQ pin je v logické úrovni HIGH)

V každé iteraci se zjišťuje, kolik bytů lze přijmout, aniž by byla narušena integrita metadat (streamy často vkládají metadata v pravidelných odstupech, tzv. *icy-metadata*). Maximální velikost načteného „chunku“ je navíc omezena na 256 bytů, protože větší bloky způsobují zpoždění ve zpracování, což by mohlo mít negativní vliv na plynulost přehrávání.

Pokud dojde na bod v rámci streamu, kde začínají meta, je ze streamu přečten jeden byte určující jejich délku (v násobcích 16). Pokud jsou metadata přítomna, načtou se a jsou předána funkci `handleMetadata()`.

Audio data jsou z bufferu odesílána na dekodér VS1053 funkcí `musicDecoder.playChunk()`, která zajišťuje zapsání dat do dekodéru FIFO. Proměnná `metaBytesRead` sleduje, kolik bytů již bylo od začátku aktuálního metadatového bloku přeneseno a podle toho se určuje, kdy očekávat další metadata.

7.3.4 Funkce budíku

Součástí návrhu internetového rádia je také implementace funkce budíku, která umožňuje v definovaný čas automaticky spustit zvukový alarm. Tato funkcionality je integrovaná přímo do hlavního běhu programu. Využívá interní hodiny systému a stavovou logiku k rozpoznání a řízení průběhu budíku.

Po nastavení času budíku a jeho aktivaci, která probíhá v rámci stavu `STATE_TIME` (konkrétně `TIME_ALARM`) se proměnná `alarmEnabled` změní na `true`, čímž se značí, že je připraven ke spuštění. V hlavní smyčce programu `loop()` je každou minutu kontrolováno,

zda aktuální čas odpovídá nastavenému budíku. Při shodě aktuálního času a času nastaveného budíku se systém přepne do stavu STATE ALARM. Do proměnné alarmStartMs se zaznamená aktuální čas (kvůli budoucímu ukončení) a na LCD displeji se zobrazí zpráva upozorňující uživatele, že budík byl spuštěn a spustí se zvuk budíku.

Po aktivaci budíku je v každém cyklu volána funkce processAlarm(), která řídí jeho chování. Pokud v momentě spuštění alarmu právě běží nějaký stream, ten je nejprve ukončen funkcí stopStream(). Následně se provede jednorázové otevření předem uloženého zvukového souboru (/Alarm/alarmBeep.mp3) z interní paměti pomocí LittleFS. Pokud není soubor dostupný, dojde k předčasnému ukončení budíku a návratu do hlavního menu.

Jakmile je zvukový soubor otevřen, funkce se opakovaně pokouší přehrávat jeho data. Pokud je dekodér VS1053 připraven (pin DREQ je HIGH) a v souboru zbývají data, načte se blok o maximální velikosti 256 bytů a je předán dekodéru funkcí musicDecoder.playChunk(). Po dosažení konce souboru se pomocí alarmFile.seek(0) přehrávání restartuje od začátku, čímž vzniká efekt „smyčky“.

Tento mechanismus zajistí, že budík bude vydávat zvuk neustále, dokud nenastane jedna ze dvou ukončujících podmínek. To je buďto automatické ukončení budíku po uplynutí definovaného časového intervalu (ALARM_DURATION) 60 sekund. Druhou možností je, že uživatel stiskne tlačítko OK pro ruční ukončení budíku předčasně. Poté se soubor zavře, proměnné pro aktivaci budíku se změní zpět na false a program se přepne do hlavního menu.

8 OVĚŘENÍ A TESTOVÁNÍ

V této části práce budou popsány testovací scénáře, které ověřují funkčnost jednotlivých komponent a částí systému internetového rádia. Každý scénář odpovídá reálnému kroku uživatele a je doložen fotografií zařízení v daném stavu. Testování probíhalo na finálně sestaveném zařízení a ověřovalo, že jednotlivé části programu i hardware pracují dle očekávání. Součástí práce je také stručný návod k obsluze zařízení. Tento dokument je přiložen v příloze.

8.1 Praktické ověření funkcionality

Postupně bude ověřena funkčnost prototypu internetového rádia na testovacích scénářích. Všechny testovací případy byly úspěšně dokončeny a zachyceny fotograficky, aby byla doložena reálná funkčnost zařízení.

Test 1 – Spuštění zařízení a inicializace

Po připojení zařízení k napájení probíhá inicializace komponent (sériové komunikace, Wi-Fi, paměťového systému, displeje, VS1053 dekodéru, tlačítek a času). Na displeji se zobrazí úvodní načítací obrazovka. Po úspěšné inicializaci všech částí systému se zobrazí hlavní menu programu.

Test 2 – Ovládání menu pomocí tlačítek

Tlačítka umožňují uživateli listovat v menu (nahoru/dolů), a také se přesouvat potvrzením výběru nebo se vracet zpět (vpravo/vlevo). Testovány byly všechny stavy (MENU, RADIO, TIME) a jejich odpovídající podnabídky. Reakce na stisk tlačítek odpovídá očekávanému chování. Při vybrání položky se program přesune do odpovídajícího stavu/podsložky.

Test 3 – Výběr souboru a přehrávání stanice

Po potvrzení výběru souboru v adresáři se zavolá funkce `startRadio()` a spustí se audio stream vybrané stanice. Začne se přehrávat audio stream do sluchátkového výstupu na dekodéru VS1053 a budou načtena metadata, které daný stream posílá. Na LCD displeji bude zobrazen název aktuálně přehrávané stanice a pomocí funkce `updateIdleDisplay()` budou periodicky zobrazována další data v druhém řádku (název stanice, název skladby, datum, čas).

Test 4 – Zobrazení a aktualizace času

Po připojení k síti se ESP32 synchronizuje s veřejným NTP serverem. Získaný čas se ukládá do interních RTC hodin. Je ověřeno, že čas běží správně a ESP32 tak může fungovat pro uživatele jako hodiny.

Test 5 – Nastavení a spuštění budíku

Ve stavu TIME -> ALARM bylo ověřeno, že při aktivaci budíku pomocí tlačítka OK se proměnná alarmEnabled změní na true. Když je budík aktivován, tak v pravém horním rohu zobrazuje „*“ jako indikace aktivovaného budíku. V čase shodujícím se s nastavením se systém přepne do stavu ALARM a probíhá zpracování aktivovaného budíku. Po jeho ukončení je načtena první položka v hlavním menu zařízení.

Test 6 – Reakce na ztrátu připojení

Simulací výpadku Wi-Fi během streamování bylo ověřeno, že systém správně detekuje odpojení, zastavení přehrávání a informuje uživatele na displeji. Tím se zajišťuje robustnost při reálném používání.

8.2 Zhodnocení funkčnosti a nedostatků

Systém splňuje všechny požadované funkce a během testování se choval stabilně. Uživatelské rozhraní je intuitivní a všechny klíčové části (streamování, budík, zobrazení informací) byly plně funkční.

Nicméně byly zaznamenány následující nedostatky:

- Dlouhá doba startu rádia – Při zapnutí zařízení trvá několik sekund, než se inicializují všechny periferie a naváže se Wi-Fi spojení.
- Nutnost internetového připojení – V aktuální implementaci není možné zařízení používat offline. Všechny klíčové funkce (včetně zobrazení času a výběru stanic) vyžadují aktivní připojení k síti.
- Zpoždění reakce na stisk tlačítka – Při přehrávání streamu může dojít k uživatelskému diskomfortu kvůli zpomalení odezvy tlačítek. Toto zpomalení je způsobeno zejména blokující I²C komunikací během aktualizací displeje.

Tyto nedostatky však nebrání základnímu používání zařízení.

8.3 Možnosti dalšího rozšíření

Během vývoje byly identifikovány možnosti budoucího rozšíření:

- Offline režim – Umožnit používání zařízení bez Wi-Fi připojení. Zařízení by stále mohlo zobrazovat čas a fungovat jako budík.
- Bluetooth audio výstup – Díky možnostem čipu ESP32 by se dalo připojit k externím Bluetooth reproduktorům, což by mohlo dodat uživatelům lepší hudební požitek.
- Ovládání přes mobilní zařízení – ESP32 umožňuje možnost fungování i jako Bluetooth přijímač. Bylo by možné například skrze mobilní aplikaci ovládat zařízení vzdáleně.
- Větší displej – Použití většího LCD nebo grafického displeje by umožnilo zobrazovat více informací (atraktivnější vzhled menu, propracovanější efekty při přepínání stavů/položek a další...). Větší nebo náročnější displeje by však mohli na ESP32 vyvíjet příliš velké hardwarové nároky.

Přestože nejsou tato rozšíření součástí stávající verze projektu, jsou technicky realizovatelná a mohou být předmětem dalšího vývoje.

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout, sestavit a naprogramovat funkční prototyp jednoduchého internetového rádia založeného na mikrokontroléru ESP32 a zvukovém dekodéru VS1053. Projekt měl sloužit nejen jako funkční zařízení pro každodenní použití, ale také jako výukový prostředek pro zájemce o oblast embedded systémů, kteří si chtějí osvojit základy návrhu elektroniky a vývoje firmwaru.

V teoretické části práce byly rozebrány principy streamování audia přes internet, přehled nejpoužívanějších audio formátů a jejich podpora ve vybraných hardwarových dekodérech. Dále byly popsány vlastnosti platformy ESP32, vývojová prostředí, přenosové protokoly a možnosti synchronizace reálného času, které jsou pro podobná zařízení klíčové.

V praktické části byl navržen a vyroben plošný spoj zařízení, následovalo osazení komponentů, jejich oživení a tvorba programového vybavení. Firmware zařízení byl navržen s důrazem na modularitu a přehlednost. Systém podporuje výběr a přehrávání online streamů z vnitřní paměti, zobrazení informací na LCD displeji a ovládání pomocí pěti tlačítek. Kromě samotného přehrávání bylo implementováno i nastavení času a budík, který přehrává uložený soubor ve stanovený čas.

Během testování byla ověřena správná funkčnost všech částí systémů, včetně připojení k Wi-Fi, přehrávání internetových stanic, práce s tlačítky, zobrazování informací a aktivace budíku. Mezi identifikované limity patří nutnost připojení k internetu, delší doba startu způsobená inicializací periférií a mírné zpomalení reakce na stisk tlačítek při přehrávání streamu. Přesto se projekt ukázal jako plně funkční a stabilní.

Zařízení splnilo všechny stanovené cíle a vytvořený systém může sloužit jako výchozí bod pro další rozvoj. Například by se jednalo o rozšíření o offline režim, zapojení funkcionality Bluetooth pro vzdálené ovládání či připojení externích reproduktorů nebo větší displej pro uživatelský komfort. Projekt tak může dále růst jak z hlediska funkcí, tak edukační hodnoty.

SEZNAM POUŽITÉ LITERATURY

- [1] RICKERT, Benjamin D. *Internet audio streaming and the emerging model of radio*. Online, diplomová práce. Michigan: Michigan State University, Telecommunication, Information Studies and Media, 2011. Dostupné z: <https://doi.org/doi:10.25335/1gr1-mh46>. [cit. 2025-04-23].
- [2] ROCKET BROADCASTER. *Icecast vs. Shoutcast: Which is Better?*. Online, článek. In: Rocket Broadcaster. 2019-01-29. Dostupné z: <https://www.rocketbroadcaster.com/help/icecast-vs-shoutcast>. [cit. 2025-04-23].
- [3] IVYPANDA. *The Internet Radio: A Critical Discussion Research Paper*. Online, článek. In: IvyPanda. 2023-12-27. Dostupné z: <https://ivypanda.com/essays/the-internet-radio-a-critical-discussion-research-paper/>. [cit. 2025-04-23].
- [4] CLOUDFLARE. *What is HTTP live streaming (HLS)?*. Online, článek. In: Cloudflare. Dostupné z: <https://www.cloudflare.com/learning/video/what-is-http-live-streaming/>. [cit. 2025-04-24].
- [5] PANTRY, Sheila. *Overview of streaming Audio and Video*. In: *Building Community Information Networks: Strategies and Experiences*. Online. London: Library Association Publishing, 1999, s. 8/1-8/15. IBSN 1-85604-337-1. Dostupné z: <https://www.dr-chuck.com/citoolkit/content/book/chapter8.pdf>. [cit. 2025-04-24].
- [6] PUBNUB. *What is HTTP Streaming?*. Online, článek. In: PubNub. 2023-009-17. Dostupné z: <https://www.pubnub.com/guides/http-streaming/>. [cit. 2025-04-24].
- [7] ESPRESSIF. *ESP HTTP Client*. Online. In: Průvodce programováním ESP-IDF. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/protocols/esp_http_client.html. [cit. 2025-04-24].
- [8] SCALEENGINE. *HTTPS Live Streaming*. Online, dokumentace. In: Scale engine. Dostupné z: <https://www.scaleengine.com/docs/playback/hls-over-https>. [cit. 2025-04-25].
- [9] SCALEENGINE. *About HLS live streaming*. Online, dokumentace. In: Scale engine. Dostupné z: https://www.scaleengine.com/docs/live_streaming/about_hls. [cit. 2025-04-25].
- [10] ESPRESSIF. *ESP32-WROOM-32: Datasheet Version 3.5*. Online, dokumentace. In: Espressif. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. [cit. 2025-04-25].

- [11] VLSI. *VS1053b Datasheet: VS1053b – Ogg Vorbis/MP3/AAC/WMA/FLAC/MIDI AUDIO CODEC CIRCUIT*. Online, dokumentace. In: VLSI Solution. 2025-04-04. Dostupné z: <https://www.vlsi.fi/fileadmin/datasheets/vs1053.pdf.com>. [cit. 2025-04-25].
- [12] THE XIPH OPEN SOURCE COMMUNITY. *FAQ*. Online. © 2023. Dostupné z: <https://icecast.org/faq/.com>. [cit. 2025-04-25].
- [13] INCECAST DOCS. *Icecast 2.4.1 Documentation*. Online, dokumentace. In: Icecast Docs. Dostupné z: <https://www.icecast.org/docs/icecast-trunk/>. [cit. 2025-04-25].
- [14] THE XIPH OPEN SOURCE COMMUNITY. *Icecast 2.4.1.Docs – Config File*. Online, dokumentace. In: Hiph.Org. Dostupné z: <https://icecast.org/docs/icecast-2.4.1/config-file.html#limits>. [cit. 2025-04-25].
- [15] SEIBEL, Peter. 28. *Practical: A Shoucast Server*. In: *Practical Common Lisp*. Online. New York: Apress, 2005, s. 401-410. IBSN 978-1-4302-0017-8. Dostupné z: <https://gigamonkeys.com/book/practical-a-shoutcast-server.html>. [cit. 2025-04-26].
- [16] SHOUTCAST. *Serverová software Shoutcast: Dohody*. Online, licenční smlouva. In: Shoutcast. 2018-08-30. Dostupné z: <https://www.shoutcast.com/legal/agreements/dnas.com>. [cit. 2025-04-26].
- [17] CHERUKURI, Anshu. *MP3 vs AAC: Which audio format works for you*. Online, článek. In: FastPix. 2025-02-28. Dostupné z: <https://www.fastpix.io/blog/mp3-vs-aac-which-audio-format-works-for-you>. [cit. 2025-04-26].
- [18] HERRE, Jurgen a DICK, Sascha. *Psychoacoustic Models for Perceptual Audio Coding: A Tutorial Review*. Online. *MDPI Open Access Journals*. 2019, vol. 9, no. 14. Dostupné z: <https://doi.org/10.3390/app9142854>. [cit. 2025-04-27].
- [19] STANFORD ENGINEERING. *MP3*. Online. In: Stanford. Dostupné z: <https://cs.stanford.edu/people/eroberts/cs201/projects/1999-00/dmca-2k/mp3.html>. [cit. 2025-04-27].
- [20] SRIPADA, Praveen. *MP3 DECODER in Theory and Practice*. Online, diplomová práce. Sweden: Blekinge Institute of Technology, Signal Processing and Telecommunications, 2006. Dostupné z: <https://www.diva-portal.org/smash/get/diva2%3A830195/FULLTEXT01.pdf.com>. [cit. 2025-04-27].
- [21] FRAUNHOFER. *Alive and Kicking: mp3 Software, Patents and Licensec*. Online, článek. In: Fraunhofer Audio Blog. 2017-05-18. Dostupné z:

<https://www.audioblog.iis.fraunhofer.com/mp3-software-patents-licenses>. [cit. 2025-04-27].

[22] HOFFMAN, Chriss. *Why Ubuntu Doesn't Come With Support for MP3s, Flash, and Other Multimedia Formats*. Online, článek. In: How To Geek. 2013-04-10. Dostupné z: <https://www.howtogeek.com/144083/why-ubuntu-doesnt-come-with-support-for-mp3s-flash-and-other-multimedia-formats/>. [cit. 2025-04-27].

[23] TODD, Nick. *Choosing the Best Radio Station Bitrate*. Online, článek. In: Radio.co. 2024-07-16. Dostupné z: <https://radio.co/blog/best-radio-station-bitrate.com>. [cit. 2025-04-27].

[24] WOOTTON, Cliff. *Standards: Part 18 - High Efficiency And Other Advanced Audio Codecs*. Online, článek. In: The Broadcast bridge. 2024-10-4. Dostupné z: <https://www.thebroadcastbridge.com/content/entry/20851/standards-part-18-high-efficiency-and-other-advanced-audio-codecs.com>. [cit. 2025-04-28].

[25] OPENLEARN. *Perceptual source – coding and lossy compression: MPEG-4 AAC (advanced audio coding)*. Online. In: The Open University OpenLearn. Dostupné z: <https://www.open.edu/openlearn/digital-computing/exploring-communications-technology/content-section-3.7>. [cit. 2025-04-28].

[26] MULTIMEDIA WIKI. *Understanding AAC*. Online. In: Multimedia Wiki. 2009-08-13. Dostupné z: https://wiki.multimedia.cx/index.php/Understanding_AAC.com. [cit. 2025-04-28].

[27] NAIR, Uma. *AAC vs MP3: Which audio format is right for you?*. Online, článek. In: Gumlet. 2024-05-19. Dostupné z: <https://www.gumlet.com/learn/aac-vs-mp3/>. [cit. 2025-04-28].

[28] FRAUNHOFER IIS. *Patent Licensing*. Online, dokumentace. In: Fraunhofer Institute for Integrated Circuits IIS. Online. 2025. Dostupné z: <https://www.iis.fraunhofer.de/en/ff/amm/lizenz/patent.html>. [cit. 2025-04-28].

[29] VIA LICENSING ALLIANCE. *AAC Licensors*. Online. 2023. Dostupné z: <https://www.via-la.com/licensing-2/aac/aac-licensors/>. [cit. 2025-04-28].

[30] VIA LICENSING ALLIANCE. *AAC: Frequently asked questions*. Online. In: Via licensing alliance. Dostupné z: <https://www.via-la.com/licensing-2/aac/aac-faqs/>. [cit. 2025-04-28].

- [31] SPDX. *Licenses: Fraunhofer FDK AAC Codec Library*. Online. 2018. Dostupné z: <https://spdx.org/licenses/FDK-AAC.html.com>. [cit. 2025-04-28].
- [32] THE XIPH OPEN SOURCE COMMUNITY. *Vorbis audio compression*. Online. Dostupné z: <https://xiph.org/vorbis/>. [cit. 2025-04-28].
- [33] MOFFITT, Jack. *Ogg Vorbis--Open, Free Audio--Set Your Media Free: Ogg Vorbis is the Open-Source Community's hot alternative to MP3*. Online. Dostupné z: <https://doi/fullHtml/10.5555/364682.364691>. [cit. 2025-04-28].
- [34] THE XIPH OPEN SOURCE COMMUNITY. *Vorbis I specification*. Online. 2020-07-04. Dostupné z: https://xiph.org/vorbis/doc/Vorbis_I_spec.html. [cit. 2025-04-29].
- [35] NAIR, Uma. *OGG vs MP3: Which is the Best Audio Format?*. Online, článek. In: Gumlet. 2024-07-07. Dostupné z: <https://www.gumlet.com/learn/ogg-vs-mp3/>. [cit. 2025-04-29].
- [36] HENRY, Dave. *AAC vs Ogg: Which Is Better in 2023?*. Online, článek. In: ViWizard. 2023-06-20. Dostupné z: <https://www.viwizard.com/drm-basics/aac-vs-ogg.html>. [cit. 2025-04-29].
- [37] AUDIOKINETIC. *Vorbis Encoder Parameters*. Online. 2024. Dostupné z: https://www.audiokinetic.com/en/public-library/2024.1.5_8803/?source=Help&id=vorbis_encoder_parameters. [cit. 2025-04-29].
- [38] NLP. *OGG Vorbis*. Online. In: Centrum zpracovávání přirozeného jazyka. Dostupné z: <https://nlp.fi.muni.cz/cs/OggVorbis>. [cit. 2025-04-29].
- [39] XIPH.ORG. *Vorbis Software Players*. Online. In: xiph.org. 2016-04-05. Dostupné z: https://wiki.xiph.org/Vorbis_Software_Players. [cit. 2025-04-29].
- [40] XIPH.ORG. *Games that use Vorbis*. Online. In: xiph.org. 2018-03-03. Dostupné z: https://wiki.xiph.org/Games_that_use_Vorbis. [cit. 2025-04-30].
- [41] LAMBDATEST. *Ogg Vorbis audio format Browser Compatibility On Mozilla Firefox*. Online. In: LAMBDATEST. 2025-03-16. Dostupné z: <https://www.lambdatest.com/web-technologies/ogg-vorbis-firefox>. [cit. 2025-04-30].
- [42] LAMBDATEST. *AAC audio file format*. Online. In: LAMBDATEST. 2025-03-16. Dostupné z: <https://www.lambdatest.com/web-technologies/aac>. [cit. 2025-04-30].

- [43] THE XIPH OPEN SOURCE COMMUNITY. *OpusFAQ*. Online. Dostupné z: <https://wiki.xiph.org/OpusFAQ>. [cit. 2025-04-30].
- [44] RANDOM NERD TUTORIAL. *ESP32 NTP Client-Server: Get Date and Time (Arduino IDE)*. Online. In: Random nerd tutorial. Dostupné z: <https://randomnerdtutorials.com/esp32-date-time-ntp-client-server-arduino/>. [cit. 2025-04-30].
- [45] ESPRESSIF. *System Time*. Online, dokumentace. In: Espressif Systems. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/system_time.html. [cit. 2025-05-30].
- [46] RANDOM NERD TUTORIAL. *ESP32: Guide for DS1307 Real Time Clock (RTC) Module (Arduino IDE)*. Online. In: Random nerd tutorial. Dostupné z: <https://randomnerdtutorials.com/esp32-ds1307-real-time-clock-rtc-arduino/>. [cit. 2025-04-30].
- [47] YANGYANG, Gao. *What Is NTP?*. Online. In: Huawei Technologies. 2021-09-30. Dostupné z: <https://info.support.huawei.com/info-finder/encyclopedia/en/NTP.html>. [cit. 2025-04-30].
- [48] DENISOV, Anatoly. *Time Synchronization: Ensuring Trust in Secure Communications with Certificates*. Online, článek. In: LinkedIn. 2023-10-10. Dostupné z: <https://www.linkedin.com/pulse/time-synchronization-ensuring-trust-secure-anatoly-denisov-ms>. [cit. 2025-04-30].
- [49] MEDIUM. *A Complete Guide to Checking RTC Accuracy on the ESP32*. Online, článek. In: Medium. 2024-10-24. Dostupné z: <https://medium.com/%40raypcb/a-complete-guide-to-checking-rtc-accuracy-on-the-esp32-3f83a5c70ec7>. [cit. 2025-04-30].
- [50] ESP BOARDS. *ESP32 DS3231 / AT24C32 Real-Time Clock (RTC)*. Online. In: ESP Boards. Dostupné z: <https://www.espboards.dev/sensors/ds3231/>. [cit. 2025-04-30].
- [51] MIND BLOG. *ESP32 použití RTC*. Online, článek. In: Voxcafe. 2022-12-30. Dostupné z: <https://www.voxcafe.cz/mindblog/clanky/arduino/esp32-pouziti-rtc.html>. [cit. 2025-04-30].
- [52] MAXIM INTEGRATED. *DS3231: Extremely Accurate I2C-Integrated: RTC/TCXO/Crystal*. Online. 2015. Dostupné z:

<https://www.analog.com/media/en/technical-documentation/data-sheets/DS3231.pdf>. [cit. 2025-04-30].

[53] HEYPETE. *A look inside the DS3231 real-time clock*. Online. In: HeyPete.com Blog. 2020-09-24. Dostupné z: <https://blog.heypete.com/2017/07/29/a-look-inside-the-ds3231-real-time-clock/>. [cit. 2025-04-30].

[54] ESPRESSIF. *ESP32: esp-dev-kits Documentation*. Online, dokumentace. 2025-05-28. Dostupné z: <https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32/esp-dev-kits-en-master-esp32.pdf>. [cit. 2025-05-01].

[55] MICRO CRYSTAL SWITZERLAND. *Application Manual: RV-8803-C7: DTCXO Temp. Compensated Real-Time Clock Module with I2C-Bus Interface*. Online, aplikační manuál. 2019. Dostupné z: https://www.microcrystal.com/fileadmin/Media/Products/RTC/App.Manual/RV-8803-C7_App-Manual.pdf. [cit. 2025-05-01].

[56] ESPRESSIF. *Chip Series Comparison*. Online, dokumentace. In: Espressif Systems. Dostupné z: <https://docs.espressif.com/projects/esp-idf/en/v5.0/esp32s3/hw-reference/chip-series-comparison.html>. [cit. 2025-05-01].

[57] NEVSEMI. *ESP32 vs. STM32*. Online, článek. In: Nevsemi.com. 2023-12-04. Dostupné z: <https://www.nevsemi.com/blog/esp32-vs-stm32>. [cit. 2025-05-01].

[58] ESPRESSIF. *ESP32 Series: Datasheet Version 4.9: 2.4 GHz Wi-Fi + Bluetooth® + Bluetooth LE SoC*. Online. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [cit. 2025-05-02].

[59] ESPRESSIF. *Modules*. Online, dokumentace. In: Espressif Systems. Dostupné z: <https://www.espressif.com/en/products/modules>. [cit. 2025-05-02].

[60] CADENCE. *Cadence Xtensa LX6 datasheet*. Online, dokumentace. In: Electronic Specifier.

2015-01-15. Dostupné z: <https://www.electronicspecifier.com/attachment/view/54b79cb9e34e242f2f1aee34/Cadence%20Xtensa%20LX6%20datasheet>. [cit. 2025-05-02].

- [61] ESPRESSIF. *ESP32WROVER: Datasheet*. Online, dokumentace. 2023. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf. [cit. 2025-05-02].
- [62] ESPRESSIF. *ESP32: Technical Reference Manual Version 5.4*. Online, dokumentace. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf. [cit. 2025-05-02].
- [63] ATMEL. *ATmega328P: 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*. Online, dokumentace. 2015. Dostupné z: https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. [cit. 2025-05-03].
- [64] ESPRESSIF. *Development Boards*. Online, dokumentace. In: Espressif Systems. Dostupné z: <https://www.espressif.com/en/products/devkits>. [cit. 2025-05-03].
- [65] ESBBOARDS. *NodeMCU-32S Development Board*. Online. In: ESB Boards. Dostupné z: <https://www.espboards.dev/esp32/nodemcu-32s/>. [cit. 2025-05-03].
- [66] AI-THINKER. *Nodemcu-32s Datasheet: Version V1*. Online, dokumentace. 2019. Dostupné z: https://docs.ai-thinker.com/_media/esp32/docs/nodemcu-32s_product_specification.pdf. [cit. 2025-05-03].
- [67] ADAFRUIT. *Adafruit HUZZAH32 – ESP32 Feather Board*. Online. In: Adamfruit Industries. Dostupné z: <https://www.adafruit.com/product/3405>. [cit. 2025-05-04].
- [68] ADAFRUIT. *Adafruit ESP32 Feather V2 - 8MB Flash + 2 MB PSRAM - STEMMA QT*. Online. In: Adamfruit Industries. Dostupné z: <https://www.adafruit.com/product/5400>. [cit. 2025-05-04].
- [69] SPARKFUN. *SparkFun Thing Plus - ESP32 WROOM (Micro-B)*. Online. In: SparkFun Electronics. Dostupné z: <https://www.sparkfun.com/sparkfun-thing-plus-esp32-wroom-micro-b.html>. [cit. 2025-05-04].
- [70] WEMOS. *D32*. Online. In: <https://www.wemos.cc/en/latest/d32/d32.html>. [cit. 2025-05-06].

- [71] SCHORCHT, Gunar. *Wemos LOLIN D32*. Online. In: Riot. Dostupné z: https://doc.riot-os.org/group_boards_esp32_wemos-lolin-d32-pro.html. [cit. 2025-05-07].
- [72] M5STACK. *Constrrollers.Core: Core2*. Online, dokumentace. In: M5Stack Developer Center. Dostupné z: <https://docs.m5stack.com/en/core/core2>. [cit. 2025-05-07].
- [73] ESPRESSIF. *ESP-IDF Programming Guide*. Online. In: Espressif Systems. Dostupné z: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html>. [cit. 2025-05-07].
- [74] ČERMÁK, David. *Espressif IoT Development Framework*. Online. In: GitHub. Dostupné z: <https://github.com/espressif/esp-idf>. [cit. 2025-05-09].
- [75] PLATFORMIO COMMUNITY. *Using esp-idf library within the Arduino Framework (ESP32)*. Online, diskuse. In: Community platformio. 2020-2022. Dostupné z: <https://community.platformio.org/t/using-esp-idf-library-within-the-arduino-framework-esp32/17226>. [cit. 2025-05-09].
- [76] MICROPYTHON. *Quick reference for the ESP32*. Online, dokumentace. In: MicroPython. Dostupné z: <https://docs.micropython.org/en/latest/esp32/quickref.html>. [cit. 2025-05-09].
- [77] MICROPYTHON. *ESP322: WROOM*. Online. In: MicroPython. Dostupné z: https://micropython.org/download/ESP32_GENERIC/. [cit. 2025-05-09].
- [78] PLAUSKA, Ignas; LIUTKEVIČIUS, Agnius a JANAVIČIUTĖ, Audronė. Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller. Online, článek. *MDPI Open Access Journals*. 2022, vol. 12, no. 1. Dostupné z: <https://doi.org/10.3390/electronics12010143>. [cit. 2025-05-10].
- [79] GITHUB. *NodeMCU 3.0.0*. Online. In: GitHub. Dostupné z: <https://github.com/nodemcu/nodemcu-firmware>. [cit. 2025-05-10].
- [80] ESPRESSIF. *ESP-AT User Guide*. Online. In: Espressif Systems. Dostupné z: <https://docs.espressif.com/projects/esp-at/en/latest/esp32/>. [cit. 2025-05-10].
- [81] ESPRESSIF. *Espressif Advanced Development Framework Guide*. Online. In: Espressif Systems. Dostupné z: <https://docs.espressif.com/projects/esp-adf/en/latest/>. [cit. 2025-05-10].

- [82] ESPRESSIF. *FreeRTOS Overview*. Online. In: Espressif Systems. Dostupné z: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html>. [cit. 2025-05-10].
- [83] ZEPHYR. *Introduction*. Online. In: Zephyr Project Documentation. Dostupné z: <https://docs.zephyrproject.org/latest/introduction/index.html>. [cit. 2025-05-10].
- [84] HORNER, Thomas. *What's Lua RTOS?*. Online, článek. In: GitHub. Dostupné z: <https://github.com/whitecatboard/Lua-RTOS-ESP32>. [cit. 2025-05-11].
- [85] ESPURINO. *Espruino on ESP32*. Online. In: Espruino. Dostupné z: <https://www.espruino.com/ESP32>. [cit. 2025-05-11].
- [86] GITHUB. *Arduino core for the ESP32, ESP32-C3, ESP32-C6, ESP32-H2, ESP32-P4, ESP32-S2 and ESP32-S3*. Online. In: GitHub. Dostupné z: <https://github.com/espressif/arduino-esp32>. [cit. 2025-05-11].
- [87] MY SMART HOME. *Shelly Plus 1 - spínací modul 1x 16A (WiFi, Bluetooth)*. Online. In: My Smart Home. Dostupné z: <https://www.mysmarthome.cz/zbozi/shelly-plus-1-spinaci-modul-1x-16a-wifi--bluetooth-/>. [cit. 2025-05-11],
- [88] SONOFF. *SONOFF Smart Power Meter Switch: POW Elite*. Online. ©2025. Dostupné z: <https://sonoff.tech/product/diy-smart-switches/pow-elite/>. [cit. 2025-05-14].
- [89] DRÁTEK NÁVODY. *Bezdrátová komunikace ESP-Now s ESP32*. Online, článek. In: Drátek návody. Dostupné z: <https://navody.dratek.cz/navody-k-produktum/bezdratova-komunikace-esp-now-s-esp32.html>. [cit. 2025-05-15].
- [90] ESPRESSIF. *ESP32-LyraT*. Online. In: Espressif Systems. Dostupné z: <https://www.espressif.com/en/products/devkits/esp32-lyrat>. [cit. 2025-05-16].
- [91] RANDOM NERD TUTORIAL. *ESP32-CAM Video Streaming and Face Recognition with Arduino IDE*. Online. In: Random nerd tutorial. Dostupné z: <https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/>. [cit. 2025-05-16].
- [92] LISZEWSKI, Andrew. *You can easily disassemble this repairable smartwatch using just a Phillips screwdriver: Basic tools are all you need to repair or upgrade the Pocuter Spectra*. Online, článek. In: The Verge. 2024-11-22. Dostupné z: <https://www.theverge.com/2024/11/22/24303386/pocuter-spectra-smartwatch-repairable-hackable-wearable>. [cit. 2025-05-16].

- [93] STMICROELECTRONIC. *Releasing your creativity: Discover the STM32 family of microcontrollers & microprocessors*. Online, prezentace. Dostupné z: https://www.st.com/resource/en/product_presentation/microcontrollers-stm32-family-overview.pdf. [cit. 2025-05-17].
- [94] STMICROELECTRONIC. *STM32H745xI/G: Dual 32-bit Arm® Cortex®-M7 up to 480MHz and -M4 MCUs, up to 2MB flash, 1MB RAM, 46 com. and analog interfaces, SMPS*. Online. 2023. Dostupné z: <https://www.st.com/resource/en/datasheet/stm32h745zg.pdf>. [cit. 2025-05-17].
- [95] FARNELL. *Arduino Uno*. Online, dokumentace. In: Farnell an avnet company. Dostupné z: <https://www.farnell.com/datasheets/1682209.pdf>. [cit. 2025-05-18].
- [96] ARDUINO. *User Manual: SKU: A000066*. Online, dokumentace. 2025. Dostupné z: <https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>. [cit. 2025-05-18].
- [97] ARDUINO. *User Manual: SKU: A000005*. Online, dokumentace. 2025. Dostupné z: <https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf>. [cit. 2025-05-18].
- [98] GITHUB. *ATmega328-Power-Save-Sleep-Solved*. Online, článek. In: GitHubb. Dostupné z: <https://github.com/IowaDave/ATmega328-Power-Save-Sleep-Solved>. [cit. 2025-05-19].
- [99] ARDUINO DOCS. *UNO WiFi Rev2*. Online. In: Arduino Docs. Dostupné z: <https://docs.arduino.cc/hardware/uno-wifi-rev2/>. [cit. 2025-05-19].
- [100] ARDUINO DOCS. *Nano ESP32*. Online. In: Arduino Docs. Dostupné z: <https://docs.arduino.cc/hardware/nano-esp32/>. [cit. 2025-05-19].
- [101] RASPBERRY, Pi. *RP2040 Datasheet: A microcontroller*. Online, dokumentace. 2025-02-20. Dostupné z: <https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf>. [cit. 2025-05-19].
- [102] RASPBERRY, Pi. *An RP2040 – based microcontroller board with wireless*. Online, dokumentace. 2024-10-15. Dostupné z: <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>. [cit. 2025-05-20].
- [103] RPISHOP. *Raspberry Pi Zero W*. Online. In: RPishop. Dostupné z: <https://rpishop.cz/raspberry-pi-zero/647-raspberry-pi-zero-w.html>. [cit. 2025-05-20].

- [NEN9] MICRO CRYSTAL SWITZERLAND. *Application Manual: RV-8803-C7: DTCXO Temp. Compensated Real-Time Clock Module with I2C-Bus Interface*. Online, dokumentace. 2019. Dostupné z: https://www.microcrystal.com/fileadmin/Media/Products/RTC/App.Manual/RV-8803-C7_App-Manual.pdf. [cit. 2025-05-21].
- [104] VLSI SOLUTION. *VS1053 - Ogg Vorbis / MP3 / AAC / WMA / FLAC / MIDI Audio Codec Chip*. Online. In: VLSI Solution. Dostupné z: <https://www.vlsi.fi/en/products/vs1053.html>. [cit. 2025-05-21].
- [105] ADAFRUIT. *VS1053 Codec + MicroSD Breakout - MP3/WAV/MIDI/OGG Play + Record - v4*. Online. In: Adafruit Industries. Dostupné z: <https://www.adafruit.com/product/1381>. [cit. 2025-05-21].
- [106] ESP REGISTRY. *Esp audio codec: Espressif audio encoder and decoder*. Online. In: ESP Registry. Dostupné z: https://components.espressif.com/components/espressif/esp_audio_codec/versions/2.0.0?language=en. [cit. 2025-05-21].
- [107] SIKORA, Ernst. *ESP32 Audio Project - Part I: Internet Radio with I²S DAC*. Online, článek. In: Hackster.io. 2022-02-05. Dostupné z: https://www.hackster.io/esikora/esp32-audio-project-part-i-internet-radio-with-i-s-dac-a5515c?utm_source=chatgpt.com. [cit. 2025-05-21].
- [108] GUPTA, Sourav. *ESP32 Based Internet Radio using MAX98357A I2S Amplifier Board*. Online, článek. In: Circuit Digest. 2021-05-24. Dostupné z: <https://circuitdigest.com/microcontroller-projects/esp32-based-internet-radio-using-max98357a-i2s-amplifier-board>. [cit. 2025-05-21].
- [109] COMPUPHASE. *The STA013 and STA015 MP3 decoders*. Online, dokumentace. In: CompuPhase. 2024-11-24. Dostupné z: <https://www.compuphase.com/mp3/sta013.htm>. [cit. 2025-05-21].
- [110] VLSI SOLUTION. *VS1003 - MP3 / WMA / MIDI Audio Codec Circuit*. Online. In: VLSI Solution. Dostupné z: <https://www.vlsi.fi/en/products/vs1003.html>. [cit. 2025-05-21].

- [111] VLSI SOLUTION. *VS1033 - MP3 / WMA / LC-AAC / MIDI Audio Codec Circuit (Discontinued)*. Online. In: VLSI Solution. Dostupné z: <https://www.vlsi.fi/en/products/vs1033.html>. [cit. 2025-05-21].
- [112] VLSI SOLUTION. *VS1063 - Audio Decoder, WAV Codec, and WAV / MP3 / Ogg Vorbis Encoder Circuit*. Online. In: VLSI Solution. Dostupné z: <https://www.vlsi.fi/en/products/vs1063.html>. [cit. 2025-05-21].
- [113] PICAXE. *DFPLayer Mini*. Online, dokumentace. Dostupné z: <https://picaxe.com/docs/spe033.pdf>. [cit. 2025-05-21].
- [114] BOTNROLL. *VS1053 MP3 Player Shield*. Online. In: Botnroll. Dostupné z: <https://www.botnroll.com/en/sound-image/4303-vs1053-mp3-player-shield.html>. [cit. 2025-05-21].
- [115] ADAFRUIT. *VS1053 Codec + MicroSD Breakout - MP3/WAV/MIDI/OGG Play + Record - v4*. Online. In: Adafruit Industries. Dostupné z: <https://www.adafruit.com/product/1381>. [cit. 2025-05-21].
- [116] JURÁNEK, Antonín a HRABOVSKÝ, Miroslav. *EAGLE pro začátečníky: návrhový systém pro plošné spoje: uživatelská a referenční příručka*. 2. vyd. Praha: BEN – technická literatura, 2007. IBSN: 9788073002138.
- [117] AUTODESK. *Autodesk Eagle: PCB design made easy for every engineer*. Online. In: Autodesk. Dostupné z: <https://www.autodesk.com/products/eagle/overview>. [cit. 2025-05-22].
- [118] PCB TRACE. *All About Irregularly Shaped PCBs*. Online. In: PCB TRACE Technologies Inc. 2024-02-05. Dostupné z: <https://pcbtrace.com/all-about-irregularly-shaped-pcbs/>. [cit. 2025-05-22].
- [119] DRÁTEK NÁVODY. *Převodník úrovní I2C 5V na 3V*. Online. In: Drátek návody. Dostupné z: <https://dratek.cz/arduino/999-prevodnik-urovni-i2c-5v-na-3v.html>. [cit. 2025-05-22].
- [120] BOTLAND. *Převodník I2C pro LCD displej HD44780*. Online. In: Botland. Dostupné z: <https://botland.cz/ostatni-prevadece/2352-prevodnik-i2c-pro-lcd-displej-hd44780-5903351248693.html>. [cit. 2025-05-22].

- [121] CRYSTALFONTZ AMERICA INC. *LCD1602A Datasheet (PDF) - Crystalfontz America, Inc.*. Online, dokumentace. Dostupné z: <https://www.alldatasheet.com/datasheet-pdf/view/1574134/CRYSTAIFONTZ/LCD1602A.html>. [cit. 2025-05-22].
- [122] SATTEL, Sam. *Library Basics Part 1: Creating Your First Package in Autodesk EAGLE*. Online, článek. In: Autodesk. 2017-04-03. Dostupné z: <https://www.autodesk.com/products/fusion-360/blog/library-basics-part-1-creating-first-package-autodesk-eagle/>. [cit. 2025-05-23].
- [123] PLATFORMIO. *Your Gateway to Embedded Software Development Excellence*. Online, dokumentace. Dostupné z: <https://docs.platformio.org/en/latest/index.html>. [cit. 2025-05-25].
- [124] STACKSHARE. *Arduino vs PlatformIO*. Online. In: Stackshare. Dostupné z: <https://stackshare.io/stackups/arduino-vs-platformio>. [cit. 2025-05-23].
- [125] SZALOMSKI, Marcin. *VS1053 library*. Online, článek. In: GitHub. Dostupné z: https://github.com/baldrum/ESP_VS1053_Library. [cit. 2025-05-24].
- [126] PEDERSEN, Jacob. *LiquidCrystal_I2C_ESP32*. Online. In: GitHub. Dostupné z: https://github.com/iakop/LiquidCrystal_I2C_ESP32?utm_source=platformio&utm_medium=piohome. [cit. 2025-05-24].
- [127] TEXAS INSTRUMENTS. *PCF8574 Remote 8-Bit I/O Expander for I2C Bus*. Online, dokumentace. 2024. Dostupné z: <https://www.ti.com/lit/ds/symlink/pcf8574.pdf>. [cit. 2025-05-24].
- [128] PROTOSUPPLIES. *LCD1602 16×2 Blue LCD with Keypad Shield*. Online. Dostupné z: <https://protosupplies.com/product/lcd1602-16x2-blue-lcd-shield-with-keypad/>. [cit. 2025-05-24].
- [129] VLSI SOLUTION. *VS1053 - Ogg Vorbis / MP3 / AAC / WMA / FLAC / MIDI Audio Codec Chip*. Online, obrázek. In: VLSI Solution. Dostupné z: https://www.vlsi.fi/typo3temp/pics/v_58d24a706c.png. [cit. 2025-05-25].

[130] LAST MINUTE ENGINEERS. *ESP32 Pinout Reference*. Online, obrázek. In: Last Minute Engineers. Dostupné z: <https://lastminuteengineers.com/wp-content/uploads/iot/ESP32-Pinout.png>. [cit. 2025-05-25].

SEZNAM OBRÁZKŮ

Obrázek 1 ESP32_DevKitC pinout	35
Obrázek 2 VS1053 Blokové schéma interní architektury	43
Obrázek 3 Blokové schéma internetového rádia	56
Obrázek 4 Eagle Schematic	59
Obrázek 5 Eagle Board Top Layer	60
Obrázek 6 Eagle Board Bottom Layer.....	61
Obrázek 7 DPS Top	62
Obrázek 8 DPS Bottom.....	63
Obrázek 9 Prototyp internetového rádia top	64
Obrázek 10 Internetové rádio bottom	65
Obrázek 11 Propojení I ² C převodníku pomocí vodičů	66

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AAC Advanced Audio Coding

ADC Analog-to-Digital Converter

ADPCM Adaptive Differential Pulse Code Modulation

AES Advanced Encryption Standard

AI Artificial Intelligence

AM Amplitude Modulation

AP Access Point

API Application Programming Interface

BLE Bluetooth Low Energy

Bluetooth AFH Bluetooth Adaptive Frequency Hopping

Bluetooth BR Bluetooth Basic Rate

Bluetooth EDR Bluetooth Enhanced Data Rate

TWAI Two-Wire Automotive Interface

CDN Content Delivery Network

CLKI Internal Clock Frequency (VS1053)

CMOS Complementary Metal Oxide Semiconductor

CPU Central Processor Unit

CRC Cyclic Redundancy Check

CVSD Continuously Variable Slope Delta

DAC Digital-to-Analog Converter

DC Direct Current

DMA Direct Memory Access

DMIPS Dhrystone Million Instructions Per Second

DNAS Dynamic Network Authentication System

DPS Desky Plošných Spojů

DREQ Data Request

DSP Digital Signal Processing

DST Daylight Saving Time

ECC Error Correction Code

EDA Electronic Design Automation

ESP-ADF ESP – Advanced Development Framework

ESP-IDF ESP – IoT Development Framework

ESP-NOW ESP – Network on Wireless

FIFO First In First Out

FLAC Free Lossless Audio Codec

FM Frequency Modulation

FPU Floating-Point Unit

FreeRTOS Open source Real Time Operating System

GPIO General Purpose Input/Output

GPS Global Positioning System

HAL Hardware Abstraction Layer

Hi-Fi High-fidelity audio

HLS HTTP Live Streaming

HMAC Hash-Based Message Authentication Code

HTTP Hypertext Transfer Protocol

HTTPS HTTP Secure

HW Hardware

I²C Inter-Integrated Circuit

I²S Inter-Integrated Circuit Sound

ICY I Can Yell

IDE Integrated Development Environment

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

LAN Local Area Network

LCD Liquid Crystal Display

Li-Pol Lithium-ion Polymer

MAC Media Access Control

MCU Microcontroller Unit

MDCT Multi-Detector Computed Tomography

MIDI Musical Instrument Digital Interface

MIPS Million Instructions Per Second

MISO Master In Slave Out

MOSI Master Out Slave In

MPEG Moving Picture Experts Group

MSB Most Significant Bit

NAT Network Address Translation

NTP Network Time Protocol

NTS Network Time Security

OTP One True Password

P2P Peer-to-Peer

PCM Pulse Code Modulation

PHY Physical Layer

PIO Programmable Input/Output

PLL Phase-Locked Loop

PSRAM Pseudo-Static Random Access Memory

PWM Pulse Width Modulation

RAM Random Access Memory

RC	Resistor-Capacitor
RFID	Radio Frequency Identification
ROM	Read Only Memory
RSA	Rivest Shamir Adleman
RTC	Real Time Clock
RTSP	Real Time Streaming Protocol
SBC	Subband Coding
SBR	Spectral Band Replication
SCI	Serial Communications Interface
SCK	Serial Clock
SCL	Serial Clock Line
SD	Secure Digital
SDA	Software Defined Access
SDI	Serial Digital Interface
SDIO	Secure Digital Input/Output
SDMMC	Secure Digital Multimedia Card
SDRAM	Synchronous Dynamic Random Access Memory
SHA	Secure Hash Algorithm
SNTP	Simple Network Time Protocol
SoC	System on Chip
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
STA	Station
SW	Software
TCXO	Temperature Compensated Crystal Oscillators
TLS	Transport Layer Security

UART Universal Asynchronous Receiver/Transmitter

UDP User Datagram Protocol

ULP Ultra-Low Power

URL Uniform Resource Locator

USB Universal Serial Bus

USB OTG USB On-The-Go

UTC Universal Time Coordinated

VBR Variable Bit Rate

VGA Video Graphics Array

WAN Wide Area Network

WAV Waveform Audio File Format

WMA Windows Media Audio

XCS Control Chip Select

XDCS Data Chip Select

XTALI Input Clock Frequency (VS1053)

SEZNAM PŘÍLOH

Příloha P 1: Janak_Web_Radio_Schematic.sch

Příloha P 2: Janak_Web_Radio_Board.brd

Příloha P 3: Janak_Web_Radio_library.lbr

Příloha P 4: Janak_Web_Radio_Board_Top.pdf

Příloha P 5: Janak_Web_Radio_Board_Bottom.pdf

Příloha P 6: Janak_BP_Návod_k_zařízení.docx

Příloha P 7: main.cpp

Příloha P 8: Osazovací plán desky TOP.png

Příloha P 9: Osazovací plán desky BOTTOM.png

Příloha P 10: Seznam součástek – tabulka.jpg

