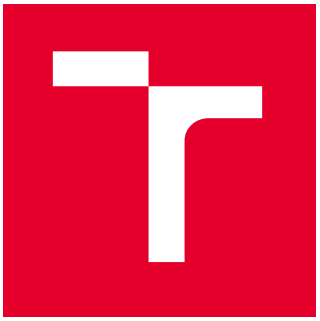


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

MULTIPROTOKOLOVÝ SNIFFER A ANALYZÁTOR

MULTI-PROTOCOL SNIFFER AND ANALYZER

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Vojtěch Lukáš**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Ondřej Krajsa, Ph.D.**

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Vojtěch Lukáš ***ID:*** 211572

Ročník: 2 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Multiprotokolový sniffer a analyzátor

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout multiprotokolový sniffer a analyzátor protokolů ZigBee, LWM a Bluetooth, který bude zachycený provoz zasílat přes Ethernet do programu Wireshark. Součástí bude vhodné konfigurační rozhraní, např. prostřednictvím webové aplikace.

DOPORUČENÁ LITERATURA:

[1] Texas Instruments. CC13xx/CC26xx SimpleLink Core SDK User's Guide [online]. [cit. 2022-11-18]. Dostupné z:
<https://dev.ti.com/tirex/explore/node?node=A__AGhAXIxP6gouUA7QxaW0Lg__com.ti.SIMPLELINK_CC13XX_CC26XX_SDK__BSEc4rl__LATEST&placeholder=true>
[2] SYSEL, Petr. Signálové procesory. Brno: UTKO FEKT VUT, 2015. ISBN 978-80-214-5187-2.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 21.5.2024

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Předmětem této práce je návrh zařízení, které zachytává Bluetooth Low Energy nebo IEEE 802.15.4 rámce a odesílá je pomocí Ethernetové sběrnice do počítače pro následnou analýzu pomocí aplikace Wireshark. Přípravek je postaven na mikročipu CC2652RB výrobce Texas Instruments, který doplňuje Ethernetové rozhraní WIZnet W5500. V práci je navržen software i hardware přípravku a ve stručnosti popsán také navržený dissector do aplikace Wireshark. Součástí teoretické části práce je pak také příručka k vývoji softwaru pro mikroprocesory firmy Texas Instruments.

KLÍČOVÁ SLOVA

Bluetooth Low Energy, Ethernet, IEEE 802.15.4, Wireshark, zachytávání

ABSTRACT

The subject of this thesis is the design of a device that captures Bluetooth Low Energy or IEEE 802.15.4 frames and sends them via Ethernet interface to a computer for further analysis using Wireshark. The device is based on the CC2652RB microchip from Texas Instruments, which is accompanied by WIZnet W5500 Ethernet controller. This thesis describes software and hardware aspects of said device, furthermore it briefly mentions a custom designed Wireshark dissector. In addition, this thesis contains a quick handbook for Texas Instruments microcontrollers software development.

KEYWORDS

Bluetooth Low Energy, Ethernet, IEEE 802.15.4, sniffing, Wireshark

LUKÁŠ, Vojtěch. *Multiprotokolový sniffer a analyzátor*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024, 60 s. Diplomová práce. Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Vojtěch Lukáš
VUT ID autora:	211572
Typ práce:	Diplomová práce
Akademický rok:	2023/24
Téma závěrečné práce:	Multiprotokolový sniffer a analyzátor

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Ondřeji Krajsovi, Ph.D. za trpělivost, ochotu a podnětné připomínky. Velký dík patří také mé rodině, přátelům a kolegům z práce i z divadla.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Bezdrátová komunikace je v současné době odvětví, které se rozvíjí rychlým tempem. Příkladem může být nastupující technologie 5G, v roce 2021 schválený standard Wi-Fi™ 6 nebo stále větší nabídka produktů pro „chytrou domácnost“ a „internet věcí“[1]. Právě IoT zařízením se bude věnovat i tato diplomová práce. Cílem je vytvořit přípravek, který bude přijímat pakety různých bezdrátových protokolů, prostřednictvím kterých spolu tato zařízení komunikují. Zachycené pakety bude pak odesílat do připojeného počítače. Analýza takto zachycených bezdrátových paketů představuje velice užitečný diagnostický nástroj při vývoji vlastního IoT zařízení, prvku chytré domácnosti nebo při analýze existující sítě a jejích uzlů.

První část této práce bude věnována stručné teorii protokolů Bluetooth®a IEEE 802.15.4. V druhé části bude podrobně popsán proces vývoje – od konfigurace IDE[2] až po *Hello World!* program. Třetí část bude pojednávat o návrhu přípravku z hardwarového pohledu, ve čtvrté bude popsán jeho software. V poslední kapitole budou prezentovány výsledky testovacích zachytávání a navrženy různé způsoby využití.

[1] Internet Of Things – internet věcí.

[2] Integrated Development Environment – Integrované vývojové prostředí.

1 Stručný popis relevantních bezdrátových protokolů

Tématem této kapitoly bude velice stručný popis pro tuto práci relevantních bezdrátových protokolů. Jmenovitě půjde o Bluetooth® Low Energy (BLE) a IEEE 802.15.4.

1.1 Bluetooth® Low Energy

BLE je velice robustní protokol pro lokální bezdrátovou komunikaci. Dal by se označit za evoluci protokolu Bluetooth® verze 3.0 (Bluetooth® Classic – BC). Překvapivě však neposkytuje vyšší přenosovou rychlost nebo větší komunikační radius. Při návrhu byl naopak kladen důraz na nízkou spotřebu energie – zařízení využívající BLE mohou být napájena už pouhou „knoflíkovou“ baterií. Architektura byla však natolik změněna, že BLE neposkytuje zpětnou kompatibilitu s BC (tedy Bluetooth® do verze 3.0). Nová BLE zařízení, která chtějí komunikovat se staršími BC zařízeními, musí pracovat v tzv. Dual-Módu.[1] [1]

Stejně jako TCP/IP protokol i BLE je vhodné vnímat po vrstvách. Ta nejnižší, tedy vrstva „nejbližší“ samotnému hardwaru zařízení, je vrstva fyzická. Nad touto vrstvou se nachází vrstva linková, aplikační a další. Vyšší vrstvy nejsou pro tuto práci relevantní, přípravek bude pouze přijímat rámce linkové vrstvy a nebude se pokoušet o jakékoliv složitější procesy.

1.1.1 Fyzická vrstva

Bezdrátová zařízení komunikují prostřednictvím elektromagnetických signálů o různých frekvencích v různých pásmech. V případě BLE se jedná o pásmo 2,4 GHz, které je rozděleno na 40 kanálů. Zmíněné pásmo celosvětově nepodléhá licenčním poplatkům, což má přirozeně nespočet výhod. Ze stejného důvodu je však toto pásmo využíváno i jinými protokoly (například Wi-Fi™ nebo IEEE 802.15.4). Z pohledu BLE je toto pásmo tedy velmi zarušeno. Kvůli tomu disponuje BLE různými systémy, které zajišťují spolehlivý přenos. Pravděpodobně nejvýznamnější je *adaptive frequency hopping*, tedy systém adaptivního frekvenčního skákání. Podstata tohoto systému komunikace tkví, ve stručnosti, v deterministické a pravidelné změně kanálu (nosné frekvence) během komunikace. [1]

Kvůli časté změně komunikačních kanálů je však mnohem složitější komunikaci odposlouchávat („sniffovat“).

[1] Tento projekt nepočítá se zpětnou kompatibilitou s BC zařízeními, soustřeďuje se na BLE.

Ze 40 BLE kanálů je 37 kanálů datových a 3 kanály ohlašovací. Tyto 3 kanály jsou v rámci spektra rozprostřeny s minimálním rozestupem 24 MHz a umístěny strategicky co nejdál od Wi-Fi™ kanálů. [1]

1.1.2 Linková vrstva

Na obr. 1.1 se nachází znázornění BLE rámce. Jsou rozlišovány dva typy těchto rámců – ohlašovací a datové. Ohlašovací rámce jsou určeny k všesměrovému přenosu, měly by tedy být čitelné pro všechna zařízení v dosahu. Jsou taktéž vysílány na ohlašovacích kanálech. Tyto zprávy lze zachytávat a následně analyzovat. Oproti tomu datové rámce si vyměňují pouze dvě zařízení, které jsou navzájem připojené – používají k tomu datové kanály. [1]

Prvních 8 bitů rámce tvoří návěští (*Preamble*). Jedná se o střídající se znaky 0 a 1. [1]

Následuje 32bitové pole *Access Address*. V případě datových rámců se v tomto poli nachází náhodně vygenerovaná hodnota, která splňuje určitá pravidla. Jedná-li se však o rámce ohlašovací, nachází se v tomto poli vždy statická hodnota `0x8E89BED6` z důvodu snazšího rozlišení začátku rámce při souvislém naslouchání. [1]

Podle pole *Header* lze rozlišit typ daného rámce. V kontextu ohlašovacích rámců se jedná o

- `ADV_IND` – obecný ohlašovací rámec,
- `ADV_DIRECT_IND` – rámec ohlašující možnost přímého připojení,
- `ADV_NONCONN_IND` – rámec ohlašující nemožnost přímého připojení,
- `ADV_SCAN_IND` – indikace skenovatelnosti,
- `SCAN_REQ` – skenovací výzva,
- `SCAN_RSP` – odpověď na skenovací výzvu,
- `CONNECT_REQ` – žádost o připojení.

Hlavička datových rámců má odlišnou strukturu a popisuje jiné typy. [1]

Následují pole *Length* vyjadřující délku dat (v bajtech), data samotná a kontrolní součet. [1]

Vzhledem k použitým technologiím je nežádoucí, aby se v BLE rámci vyskytovalo více stejných znaků (0 nebo 1) za sebou. K potlačení tohoto jevu se využívá „whitening“, což je proces, kdy je odchozí zpráva XORována s pseudonáhodnou posloupností, čímž je zvýšena její entropie. Při návrhu zařízení určeného k zachytávání je toto nutné brát v potaz. [1]

1.2 IEEE 802.15.4

Standard IEEE 802.15.4 popisuje, na rozdíl od Bluetooth®, pouze fyzickou a linkovou (MAC) vrstvu. Spíše než jako svébytný protokol je proto vhodnější vnímat tento standard jako *základní stavební kámen* pro jiné protokoly, například ZigBee™. Oproti BLE definuje nejen kanály v pásmu 2,4 GHz, ale také v pásmu 915 MHz (pro oblast USA) a 868 MHz (pro oblast Evropy). [2]

1.2.1 Vrstva PHY a MAC

IEEE 802.15.4 využívá technologie mnohonásobného přístupu CSMA/CA. Zařízení naslouchá na daném kanále po předem nastavený čas. Pokud je kanál prázdný, začne transfer dat, v opačném případě zařízení čeká náhodnou dobu a pokus opakuje. [3]

Síťové topologie podporuje standard dvě: *star* a *peer-to-peer*. Základním prvkem obou těchto topologií je koordinátor sítě – PAN[2] koordinátor. Pro vytvoření složitějších sítí a topologií je třeba spoléhat na protokoly vyšších vrstev. [3]

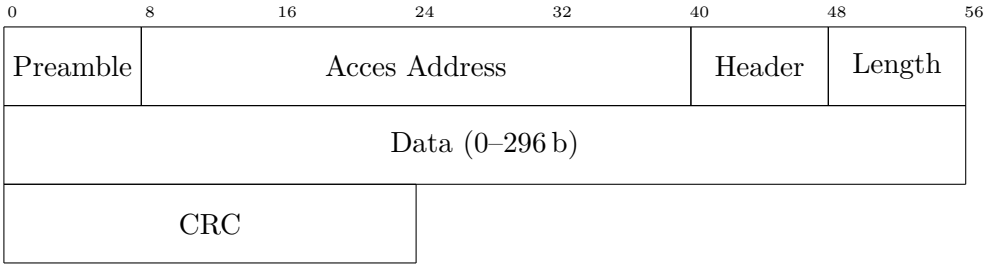
Struktura PHY rámce je zobrazena na obr. 1.2. Po 32bitovém návěští (*Preamble*) následuje pole *Start Frame Delimeter*. Dále pak 7bitové pole *Frame length* – to určuje délku PSDU (*PHY Service Data Unit*). [2, 3]

Obsahem PSDU je MSDU (*MAC Service Data Unit*) zobrazené na obr. 1.3. Podle obsahu pole *Frame Control* rozlišujeme čtyři základní typy rámců:

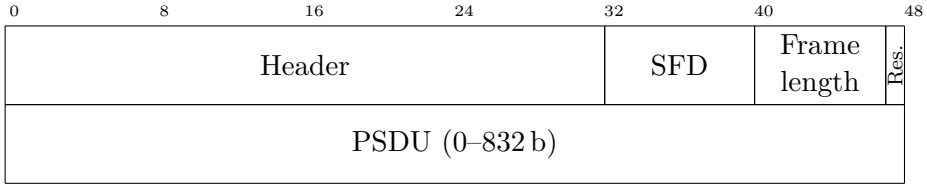
- *Data frame* (aplikační data),
- *Acknowledgement frame* (potvrzení přijetí),
- *Beacon frame* (posílá PAN koordinátor),
- *MAC command frame* (správa MAC vrstvy).

Sequence number se používá k očíslování zpráv typu Beacon a Data. Zprávy Acknowledgement by měly dané číslo zaslat zpět jako potvrzení přijetí. Pole *Address Info* obsahuje buď 16bitové, nebo standardní 64bitové MAC adresy zdroje i cíle. Následuje sekce pro data vyšších protokolů *Payload Data* a kontrolní součet. [2, 3]

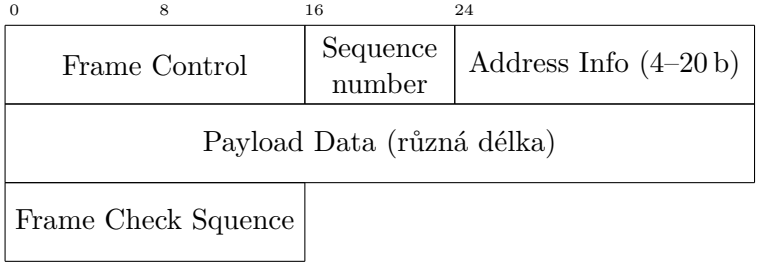
[2]Personal Area Network – zpravidla bezdrátová síť malého rozsahu.



Obr. 1.1: BLE rámec [1]



Obr. 1.2: IEEE 802.15.4 PHY rámec [2]



Obr. 1.3: IEEE 802.15.4 MAC paket [3]

2 Příručka k vývoji TI SimpleLink™ aplikací

Tato část diplomové práce si dává za cíl pomoct vývojáři s vývojem embedded aplikace v Texas Instruments (TI) ekosystému. Bude zde rozebrán každý krok, od výběru platformy přes stažení IDE až po spuštění první *Hello World!* aplikace a následný debugging. Text však zároveň počítá s tím, že vývojář zná základy programování i práci v IDE Eclipse a CCS. Kvůli komplexnosti daného tématu je doporučeno nejprve přečíst celou kapitolu a až následně provádět zmíněné úkony.

2.1 Výběr platformy

I přesto, že TI vydává *většinu* vývojového softwaru pro všechny majoritní platformy (Windows, macOS i Linux), **je autorem důrazně doporučeno zvolit pro vývoj počítač s operačním systémem Windows.** V době psaní této práce jsou všechny knihovny, IDE i ostatní nástroje kompatibilní se systémem Windows 7 a novějšími.

Pokud vývojář nemá přístup k zařízení fungující na OS Windows, ztrácí možnost využít mnoha podpůrných aplikací, které TI vydalo výhradně pro tuto platformu. Je pravděpodobné, že také narazí na větší množství problémů s kompatibilitou.

Následující text bude předpokládat využití počítače s Windows 7.

2.2 Výběr hardwaru

Web Texas Instruments (`ti.com`) je uživatelsky přívětivý, pokud zná vývojář alespoň základní požadavky na konečnou funkcionalitu zařízení, výběr MCU mu nezabere mnoho času. TI nabízí nepřeberné množství mikročipů k nejrůznějším účelům. Poskytuje také bohatou dokumentaci a k určitým mikrokontrolérům také školicí centrum SimpleLink™ Academy.

„SimpleLink™“ je označení pro rodinu mikrokontrolérů určených primárně pro IoT aplikace. Hlavní výhodou této platformy je přenositelnost zdrojových kódů mezi jednotlivými mikročipy. SimpleLink™ zastřešuje ovladače (API pro periferie), TI–RTOS[1], POSIX vrstvu a další prvky.

Tento projekt bude postaven na MCU CC2652RB, který bude blíže rozebrán v kapitole 3.4. Pro pohodlný vývoj je vhodné využít tzv. LaunchPad verzi vybraného čipu. Při zakoupení tohoto produktu vývojář obdrží vývojový kit – PCB s daným MCU a s programátorem (debuggovací sondou), který zajistí pohodlné nahrání zdrojového kódu a diagnostiku programu.

[1] Texas Instruments – Real Time Operating System.

2.3 Instalace podpůrného softwaru

2.3.1 Zjištění aktuální verze SDK

Po výběru MCU je možné přistoupit k instalaci jeho podpůrného softwaru. První krok bývá přirozeně instalace IDE pro danou platformu, v tomto případě by se jednalo o Code Composer Studio (CCS). **Autor práce však doporučuje se stažením a instalací vývojového prostředí počkat a nejprve prostudovat dokumentaci.**

První je vhodné zjistit aktuální verzi dostupného SDK pro daný mikrokontrolér. Například pro **CC2652RB** je v době psaní této práce dostupné aktuální SDK **SIMPLELINK-CC13XX-CC26XX-SDK** ve verzi **6.30.01.03**. Tyto informace jsou snadno k dostání ne webu `ti.com`.

Po zjištění názvu a aktuální verze daného SDK je možné vyhledat dokumentaci, přesněji dokument, který se označuje jako SDK Release Notes. Ten mimo jiné obsahuje seznam podpůrných programů s jejich kompatibilními verzemi.[2].

Dokument *Release Notes* lze najít v modulu *Resource Explorer*.

2.3.2 Modul Resource Explorer

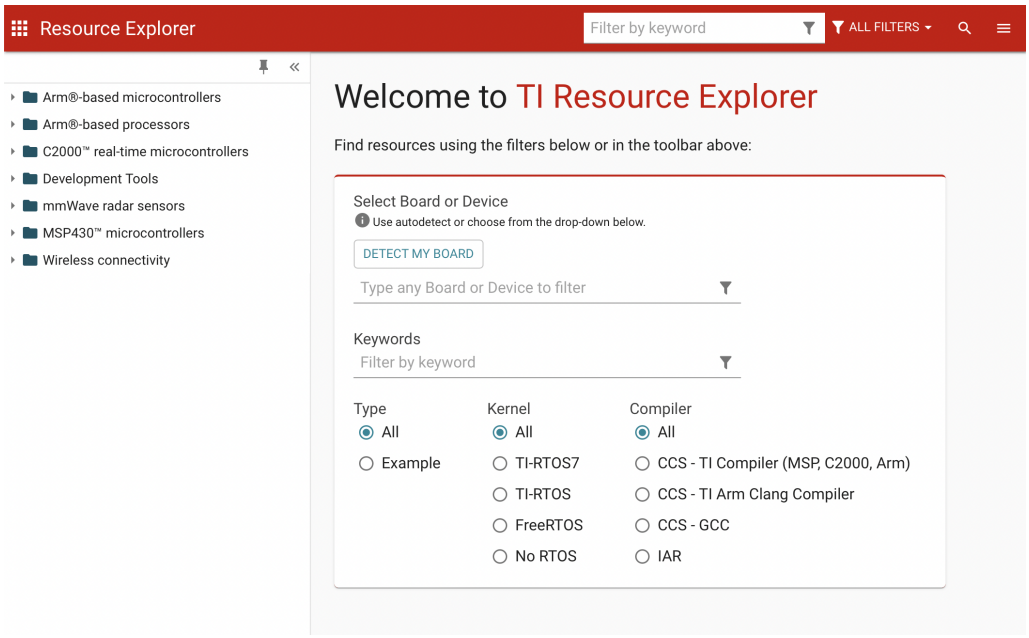
Tento robustní webový nástroj obsahuje velké množství návodů, příkladů, videí, informací. Během vývoje se bude k tomuto „průzkumníkovi“ vývojář pravidelně vracet, je tedy vhodné se v něm přiměřeně zorientovat.

Průzkumník je dostupný na webu[3] `dev.ti.com` pod odkazem *Browse software and examples*, jak je patrné z obr 2.1.

V případě vývoje SimpleLink™ aplikací se budeme pohybovat pouze v rámci kořenové složky *Wireless connectivity*.

[2]Ne vždy je nejnovější verze daného podpůrného programu kompatibilní s nejnovější verzí SDK!

[3]V pozdější fázi bude dostupný také přímo v IDE.



Obr. 2.1: TI Resource Explorer

2.3.3 Instalace IDE podle SDK *Release Notes*

Kýžený dokument se nachází v adresáři

`Wireless connectivity` / `Embedded Software` / `SimpleLink CC13xx CC26xx SDK` / `Documents`.

V odstavci *Dependencies* je již zmíněný seznam podpůrného softwaru. V tuto chvíli lze nainstalovat IDE Code Composer Studio **v té verzi, která je zde uvedená.** Ostatními programy se v tuto chvíli není třeba zabývat, automaticky se nainstalují později.

2.3.4 Instalace SDK

I když lze pomocí instalátoru IDE nainstalovat i dané SDK, je mnohem vhodnější vyřešit tuto instalaci zvlášť. Instalační soubor aktuálního SDK lze opět stáhnout z webu `ti.com`. Jeho instalace je pak přímočará. Cesty pro instalaci IDE i SDK netřeba upravovat.

2.3.5 Automatická instalace kompilátorů a jiných utilit

V tuto chvíli by měl mít vývojář nainstalované navzájem kompatibilní IDE i SDK. Zkontrolovat to lze ve spuštěném CCS po otevření okna `Preferences` z kaskádového menu `Window`. V podokně `Code Composer Studio → Products` se nachází seznam nalezených produktů ve všech nainstalovaných verzích.

Sekundární kontrolu lze provést importováním libovolného projektu z modulu *Resource Explorer*. Ten lze otevřít pomocí kaskádového menu `View → Resource Explorer`. Ukázkové projekty lze nalézt v adresářích

`Wireless connectivity / Embedded Software / SimpleLink CC13xx CC26xx SDK / Examples / Development Tools / CC2652RB LaunchPad /`.

Tuto cestu zjednodušme na symbol „`~`“. Vhodným testovacím příkladem je projekt

`~/ TI Drivers / empty / TI-RTOS7 / TI Clang Compiler / empty`.

Jedná se o jednoduchý příklad, kdy je v rámci hlavní smyčky vytvořeno pouze jedno vlákno aplikace. Toto vlákno pak každou vteřinu přepne vestavěnou LED.

Pro import projektu je třeba kliknout na tři tečky u názvu souboru a zvolit možnost *Import to IDE*. CCS si v tuto chvíli zjistí, zdali je v počítači nainstalováno SDK, kompilátor a další potřebné programy. Pokud nějaký prvek chybí, zobrazí se vývojáři dialogové okno s nabídkou instalace chybějícího softwaru.[4] Po přijetí a instalaci lze již naimportovat `empty` projekt. Následná kompilace a nahrání do LaunchPadu by měla proběhnout bez problémů.

2.4 Konfigurace projektu

2.4.1 Popis významných souborů v projektu

Stažený projekt se nyní objeví v levém panelu CCS. Po jeho rozbalení je vidět, že obsahuje mnoho složek i hlavičkových nebo zdrojových souborů. Ty nejdůležitější jsou:

- `main_tirtos.c` – zdrojový soubor, který obsahuje hlavní funkci programu `main()`. Ta se spustí při startu MCU jako první, nakonfiguruje vše potřebné, vytvoří vlákno a pak zavolá funkci `BIOS_start()`, která toto vlákno spustí.
- `empty.c` – obecný zdrojový soubor, který v případě projektu `empty` obsahuje definici funkce vlákna `mainThread()` s nekonečnou smyčkou. V tomto konkrétním případě se děje pouze to, že vlákno co 1 s přepne `LED0`. V obecném případě se však v těchto zdrojových souborech bude nacházet veškerý užitečný kód programu.
- `empty.syscfg` – velice důležitý soubor, po jehož otevření se zobrazí grafický editor (utilita *SysConfig*) pro přidávání a konfiguraci jednotlivých ovladačů dostupných pro dané MCU. Prostřednictvím tohoto nástroje tak lze „zapnout“

[4] Může se stát, že bude mezi chybějícím softwarem i SDK, které je však již nainstalované. V takovém případě je nutné znovu překontrolovat stránku `Products` podle 2.3.5, zavřít a otevřít *Resource Explorer*, případně celé CCS a akci opakovat.

a konfigurovat jádro pro rádiovou komunikaci, zvolit vstupní a výstupní piny pro sběrnici UART apod. Při kompilaci tato utilita vytvoří hlavičkové soubory jako `ti_drivers_config.h` nebo `ti_radio_config.h`, které je třeba ve zdrojových souborech přiložit pomocí direktivy `#include`.

2.4.2 Konfigurace debuggování

Při vývoji jakéhokoliv programu patří základní výpis do konzole mezi elementární diagnostické nástroje používané takřka při každé příležitosti. Zde je však situace paradoxně docela složitá. Standardní funkce `printf()` se podle [4] nedoporučuje kvůli jejím negativním vlivům na výkon programu. TI ve svých návodech naopak radí používat funkci `System_printf()`, která však netiskne výstup do konzole, ale do utility Runtime Object View (ROV). Tu lze otevřít pomocí kaskádového menu `View → Other → Other → Runtime Object View`.

Aby vše fungovalo správně, je třeba nejprve v souboru `empty.syscfg` najít modul `System` (nachází se v oddílu `TI-RTOS → RUNTIME`) a ujistit se, že je záznam `SystemSupport Module` nastaven na `SysMin`. Dále je nutné ve zdrojovém souboru zahrnout systémovou knihovnu pomocí direktivy `#include`, jak lze vidět ve výpisu 2.2.

Nyní již lze tisknout text nebo diagnostické údaje přímo do vývojového prostředí, přesněji v ROV do okna *SysMin*. Mimo tuto funkcionalitu poskytuje modul ROV další diagnostické informace. Za zmínku stojí okno *Tasks*, které zobrazuje informace o vytvořených vláknech, jejich paměti a současném stavu (*Running, Idle, Blocked* ...). Pro získání aktuálních údajů je nutné modul ROV obnovit (tlačítkem s motivem jedné šipky) nebo zapnout periodické obnovování (tlačítkem s motivem dvou šipek).

Výpis 2.1: Direktiva a funkce pro výpis do modulu ROV

```
1 #include <xdc/runtime/System.h>
2 System_printf("text␣k␣tisku");
```

2.4.3 *Hello World* aplikace

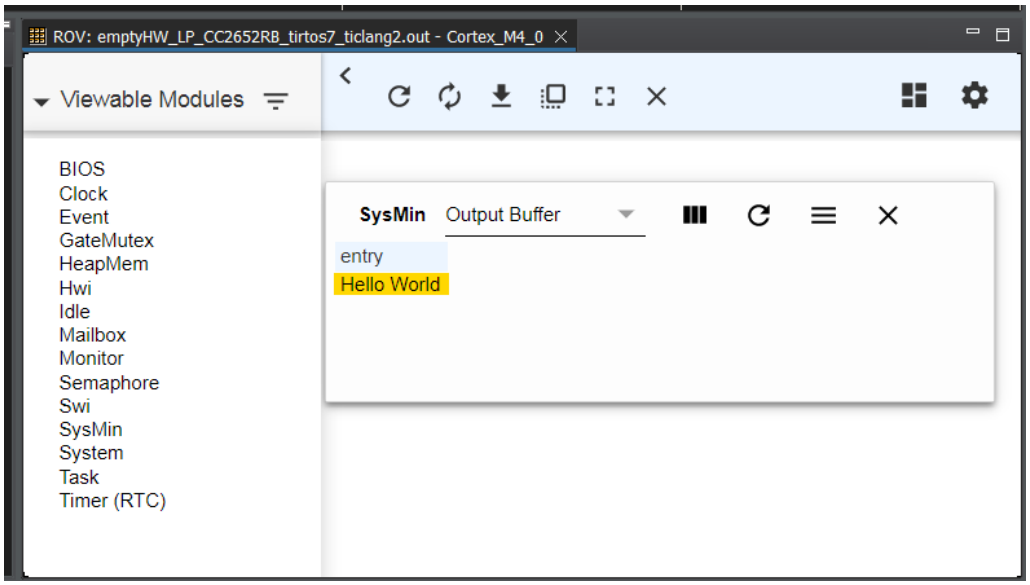
Ještě před zahájením vývoje jakékoliv užitečné aplikace je vhodné vše zkontrolovat a otestovat na jednoduchém programu, který pouze vypíše text do konzole, v tomto případě do ROV. I přes zdánlivou banálnost tohoto programu nám jeho úspěšné spuštění potvrdí správné provedení hned několika zmíněných úkonů (od úspěšné komunikace s programátorem resp. mikročipem, přes instalaci správného SDK až po správné nastavení souboru `.syscfg`).

Výpis 2.2: Soubor `main_tirtos.c` po úpravách

```
1  #include <stdint.h>
2  #include <ti/sysbios/BIOS.h>
3  #include <ti/drivers/Board.h>
4  #include <xdc/runtime/System.h>
5
6  int main(void)
7  {
8      Board_init();
9      System_printf("Hello␣World!\n");
10     BIOS_start();
11     return 0;
12 }
```

Takovýto program jednoduše vytvoříme úpravou kódu v souboru `main_tirtos.c` projektu `empty`. V tuto chvíli lze smazat veškerý kód spojený s rozhraním *POSIX*, tedy objekty `pthread` apod. Pro vytisknutí textu je pak nutné provést úkony nastíněné v sekci 2.4.2. Výsledný kód by mohl odpovídat tomu z výpisu 2.2. Po spuštění programu a otevření utility ROV by měl vývojář vidět totéž, co na obr. 2.2.

Pro vytvoření každého nového projektu **je autorem práce doporučeno využít již zmíněný existující projekt** `empty`, který poslouží jako šablona.



Obr. 2.2: Okno ROV se zápisem *Hello World!*

2.5 Shrnutí

Všechny zmíněné kroky lze shrnout následovně:

1. vývoj ideálně na Windows PC,
2. výběr hardwaru ze stránek `ti.com`,
3. zjištění SDK pro vybraný hardware,
4. podle *Release Notes* aktuální verze SDK (ke stažení na `dev.ti.com` v modulu *Resource Explorer*) instalace kompatibilního IDE,
5. stažení a instalace SDK z `ti.com`,
6. kontrola přítomnosti veškerého podpůrného softwaru,
7. import projektu `empty` z modulu *Resource Explorer*,
8. změna *SysCallback* na *SysMin*,
9. přidání direktivy `#include`,
10. přidání funkce `System_printf()`.

V tuto chvíli může začít vývojář vyvíjet užitečnou aplikaci.

3 Hardware zařízení

Předmětem této kapitoly je popis hardwaru výsledného zařízení. Celková architektura je znázorněna v blokovém schématu na obr. 3.1. Schéma výsledného přípravku je zobrazeno v příloze D.1. Centrálním prvkem je MCU CC2652RB firmy Texas Instruments. Tomuto čipu sekunduje WIZnet W5500, který zprostředkovává síťové rozhraní.

3.1 Napájecí obvody

Jak lze vidět na schématu z obr. 3.1, zařízení je napájeno prostřednictvím mini-USB konektoru. Ten dodává napětí 5 V regulačnímu obvodu, který jej redukuje na 3,3 V, což je napětí vhodné pro všechny zbylé komponenty na desce plošných spojů.

Regulační obvod je tvořen spínaným regulátorem MAX77827. Vstupní napětí tohoto integrovaného obvodu se může pohybovat v rozmezí 1,8 V až 6,6 V. Výstupní napětí je pohodlně nastavitelné v rozsahu 2,3 V až 5,3 V pouze úpravou hodnoty rezistoru R_{SEL} zapojeného mezi terminál SEL a zem. Pro dosažení požadovaného výstupního napětí 3,3 V lze ponechat kontakt SEL nezapojený, popřípadě uzemněný ($R_{\mathrm{SEL}} = 0\,\Omega$). [5]

Toto napájecí řešení dovede poskytnout proud až 1,8 A [5]. Vzhledem k celkovému proudovému odběru přibližně 650 mA (viz tab. 3.1) poskytuje toto řešení dostatečnou rezervu.

Tab. 3.1: Tabulka proudových odběrů jednotlivých integrovaných obvodů a MCU [6, 7, 8, 9, 10]

Integrovaný obvod/MCU	Proudový odběr [mA]
CC2652RB	±10,3
W5500	132,0
BGS12P2L6	0,1
CP2102	500,0
SSD1306	0,8
Celkem	**643,2**

3.2 Sériová linka

Mini-USB konektor lze využít nejen k napájení, ale také k sériové komunikaci se zařízením. Aktuální verze softwaru sice s touto eventualitou nepočítá, DPS je však

na tento způsob komunikace připravena.

Tato příprava je realizována využitím integrovaného obvodu CP2102, který je napojen mezi komunikační linky USB rozhraní a UART RX/TX terminály hlavního MCU [9]. Dokud však nebude sériová komunikace softwarově implementována, bude tento převodník odpojen od napájení i od komunikačních linek, a tím tak nebude reagovat na žádné podněty. Ve schématu (viz příloha D.1) je toto odpojení znázorněno nulovými rezistory R5, R6 a R8.

3.3 OLED displej

Displej není považován za integrální součást výsledného zařízení, proto také není zaznačen ve schématu v příloze D.1. Pro pohodlné zacházení je však vhodné jej k zařízení dodatečně připojit přes kolíkové lišty SV1 nebo SV3.

Pro svou jednoduchost byl vybrán OLED displej SSD1306 ve verzi 128×64 pixelů, který s hlavním MCU komunikuje prostřednictvím protokolu I2C. Jeho úhlopříčka činí 0,96 palce a napájet jej lze prostřednictvím napětí 3,3 V [10]. Jeho ovladač a grafické rozhraní budou podrobněji přiblíženy v sekci 4.6.1.

3.4 Hlavní MCU

Mikročip CC2652RB patří do rodiny SimpleLink™ zařízení výrobce Texas Instruments. Představuje kompletní řešení bezdrátové komunikace pro IoT zařízení. Vyniká svou nízkou energetickou náročností a širokou nabídkou podporovaných bezdrátových protokolů. Využití nalezne například ve spotřební elektronice, sportovních pomůckách, jednotlivých prvcích chytré domácnosti nebo také v průmyslových aplikacích. [4]

Základem tohoto mikrokontroléru je 32bitový procesor s jádrem Arm® Cortex® M4F s taktovací frekvencí až 48 MHz, který poskytuje dostatek výkonu pro zmíněné aplikace při nízké spotřebě energie. Kapacita flash paměti je 352 kB, kapacita paměti SRAM je pak 88 kB. Mikročip disponuje několika komunikačními rozhraními: UART, I2C, SPI i audio rozhraním I2S. [4]

Mikrokontrolér CC2652RB je tzv. SoC[1], v jedné součástce tedy kombinuje více funkčních prvků. Takovýmto samostatným integrovaným prvkem je také rádiové jádro *RF Core*. Úkolem tohoto rádiového koprocesoru je maximálně zjednodušit bezdrátovou komunikaci v pásmu 2,4 GHz. Je založeno na procesoru Arm® Cortex® M0

[1] System-on-chip – integrovaný obvod, který kombinuje více funkčních prvků pro provoz systému na jednom čipu.

s 16 kB paměti SRAM. Toto jádro vývojáři otevírá možnosti komunikace prostřednictvím BLE, IEEE 802.15.4, ZigBee nebo vlastního proprietárního protokolu. [4]

Ve chvílích, kdy je potřeba často přesouvat větší množství dat mezi pamětí, periferiemi a mikroprocesorem, je vhodné využít kontroléru µDMA. Díky využití tohoto 32kanálového řadiče nebude zatěžováno jádro a běh programu bude efektivnější. Modul µDMA je také možné použít k přímému přístupu do paměti během debuggování. [4, 11]

Tento mikrokontrolér byl pro tento projekt vybrán zejména díky bezdrátovým protokolům, se kterými pracuje (BLE a IEEE 802.15.4). Dalším faktorem pro výběr byl i velký počet různých periferií, zejména SPI a I2C. Posledními kritérii byly výkonnost a srozumitelná dokumentace. Vším zmíněným tento MCU disponuje.

Jak lze vidět na schématech (obr. 3.1, resp. příloha D.1), tento čip je centrálním prvkem celého návrhu. Komunikuje s Ethernetovým rozhraním přes SPI, s OLED displejem přes I2C, je k němu také připojena RF anténa a eventuálně bude moct komunikovat přes USB i pomocí UART sběrnice.

3.5 Ethernetové rozhraní

Mikrokontrolér W5500 od výrobce WIZnet byl vybrán díky své jednoduchosti a dostupnosti – i díky tomu se hojně využívá v mnoha jiných IoT zařízeních a je také součástí Ethernetového rozšíření pro zařízení Arduino. [7]

Jakožto SoC Ethernetové rozhraní obsahuje kompletní TCP/IP stack od vrstvy PHY a MAC přes protokol ARP nebo IPv4 až po TCP, UDP, ICMPv4 a další. Fyzická vrstva dovede komunikovat rychlostmi 10 Mbit/s nebo 100 Mbit/s. Podporuje *Auto Negotiation* (automatickou konfiguraci Ethernet komunikace a RJ-45 zásuvek) nebo *Wake on LAN*. Mikročip integruje také 32kB vyrovnávací paměť pro zpracování zpráv. Díky těmto vlastnostem značně uvolňuje zátěž hlavnímu mikrokontroléru, jehož procesorový čas tak může být soustředěn primárně na konstrukci obsahu segmentů nebo datagramů. [7]

Vývojáři tento mikročip poskytuje 8 nezávislých TCP nebo UDP socketů – výsledné zařízení tedy může v rámci jednoho socketu například naslouchat na portu 80 (a fungovat tak jako jednoduchý webový server) a zároveň prostřednictvím jiného socketu neustále odesílat UDP zprávy. [7]

S MCU zařízení komunikuje prostřednictvím rozhraní SPI, a to v maximální rychlosti 80 Mbit/s. V rámci SPI komunikace podporuje W5500 mód *Fixed Data Rate* a *Variable Data Rate*, který je oproti prvnímu jmenovanému mnohem efektivnější. Tyto módy se navzájem liší počtem využitých SPI pinů.

Ze schématu na obr. 3.1 je zřejmé, že je tento čip propojen s konektorem RJ-45. Schéma v příloze D.1 pak toto propojení rozvádí detailněji.

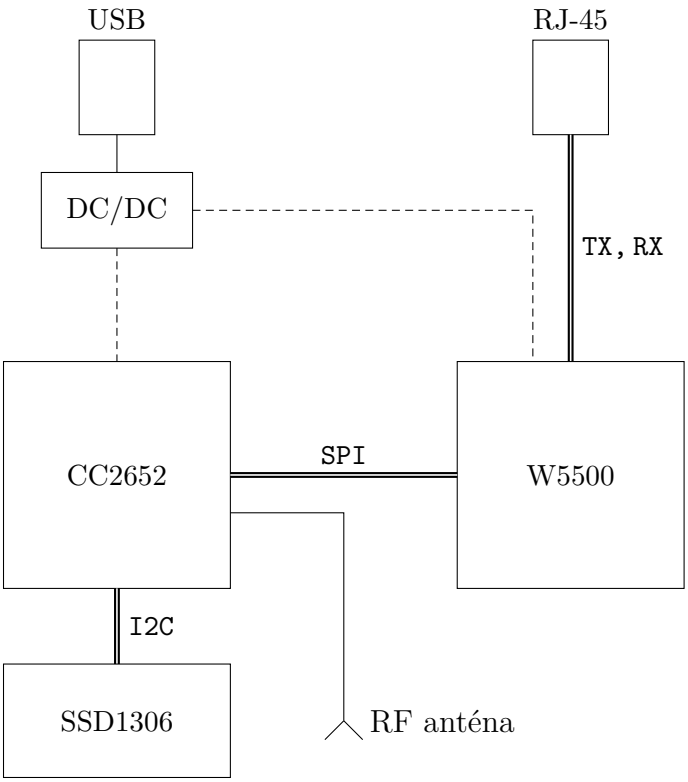
3.6 Programovací rozhraní

Oproti prototypu, který byl vyvíjen v rámci semestrální práce, nedisponuje finální zařízení debuggerem XDS 110. Pro nahrání programu a jeho případnou diagnostiku byl z toho důvodu přípravek vybaven programovacím rozhraním, které komunikuje s *externím* debuggerem.

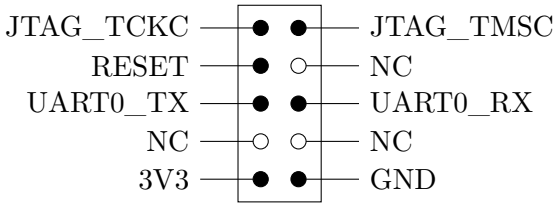
Toto rozhraní má podobu kolíkové lišty, jejíž kontakty jsou propojeny s kontakty procesoru. Podrobné schéma tohoto rozhraní je patrné na obr. 3.2, resp. v příloze D.1.

Procesor CC2652RB používá v implicitním stavu zjednodušené dvouvodičové JTAG rozhraní s názvem cJTAG [12], které obsahuje pouze dva vývody: TMSC (data) a TCK (hodinové impulzy). Kromě cJTAG rozhraní je na kolíkové liště vyvedena také sériová linka (UART0_RX, UART0_TX), jež má za úkol zprostředkovat komunikaci se zavaděčem [12]. Zbývá zmínit kontakt pro přivedení napájecího napětí a země.

Po propojení počítače a debuggeru XDS 110 (lze využít i debugger na evaluačním kitu – LaunchPadu) se zmíněnými kontakty je možné přípravek programovat a diagnostikovat.



Obr. 3.1: Blokové schéma zařízení



Obr. 3.2: Schéma programovacího rozhraní

4 Software zařízení

Program přípravku je pojmenován „multiSniff“. Využívá TI–RTOS7 a je kompilován TI Clang kompilátorem. Byl vyvíjen v prostředí Code Composer Studio v souladu s kapitolou 2. Součástí výsledného zdrojového kódu jsou i knihovny jiných autorů, jejich jména jsou uvedena v hlavičkách jednotlivých zdrojových souborů. Předmětem této kapituly bude mimo program samotný také webové rozhraní, jehož HTML kód se do zařízení nahrává samostatně.

TI–RTOS je operační systém reálného času poskytující robustní základ pro systémy, které mají vysoké požadavky na přesné časování. Využívá preemptivní multitasking, kdy může jádro odebrat řízení úloze, což je oproti kooperativnímu multitaskingu, kdy je předání řízení závislé na úloze samotné, mnohem bezpečnější řešení. TI–RTOS umožňuje vývojáři „rozdělit“ software na více samostatných vláken, která vykonávají různé dílčí funkce celkového programu. [11]

I v tomto případě je software rozdělen na více dílčích vláken, která dohromady vykonávají celistvý program. Jednotlivá vlákna budou v této kapitole blíže rozebrána. Mimo software samotného zařízení bude v této kapitole popsán také HTML a JavaScript kód webového ovládacího panelu, ovladač pro OLED displej a ve stručnosti zmíněn protokol *mSniff* s jeho Wireshark dissectorem.

4.1 Status Vektory

Aby mohla vlákna mezi sebou sdílet informace a adekvátně reagovat na aktuální nastavení, bylo vytvořeno jednoduché datové rozhraní nazvané „Status Vectors“, jehož implementaci lze nalézt v souborech `/source/utils/stv.h` a `stv.c`. Podoba tohoto rozhraní je naznačena v tabulce A.1.

Přednastavené hodnoty této tabulky jsou spolu s objektovým kódem programu nahrány do flash paměti zařízení. V rámci inicializačního vlákna (viz kapitola 4.2) jsou pak překopírovány do paměti RAM, kde mohou být za běhu programu aktualizovány. Za zmínku stojí, že hodnoty v RAM paměti nejsou softwarovým resetem inicializovány – tohoto lze využít kdykoliv je potřeba restem upravit funkcionalitu programu.

4.2 Vlákno Init

Inicializační vlákno neobsahuje nekonečnou smyčku jako zbylá vlákna – je vykonáno pouze jednou a slouží k úvodnímu nastavení komunikačních rozhraní, připojení zařízení do sítě a v neposlední řadě také k vytvoření dalších vláken. Vývojový diagram je vyobrazen na obr. C.1.

V úvodu exekuce dojde k inicializaci periferií (SPI, GPIO, I2C). Následuje procedura k získání IP adresy – zařízení odešle DHCP požadavek a čeká na odpověď, která bude mimo jiné obsahovat i IP adresu pro přiřazení. Pokud tato procedura neskončí úspěšně, je přepnut Status Vektor pro použití DHCP a zařízení je resetováno. Po resetu se pak samo nakonfiguruje za použití statických hodnot (viz tabulka A.1) přednastavených v rámci Status Vektorů.

Jakmile je zařízení nakonfigurováno pro připojení do sítě, je aktualizován OLED diplej a vytvořena zbylá dvě vlákna. Tento průběh tak zaručuje, že přípravek bude mít vždy přiřazenou validní IP adresu a bude jej tedy možné ovládat pes webové rozhraní.

4.3 Vlákno Sniffing

Úkolem tohoto vlákna je ovládat rádiový koprocesor (RF Core) a odesílat zachycené rámce do cílového zařízení. Rádiový koprocesor je svým způsobem nezávislý na hlavním jádru a pracuje autonomně. Zachycené rámce ukládá do připravené fronty, z níž je pak toto vlákno čte.

Hlavní funkce vlákna je implementována v souboru `/sniffing_task.c`. Pomocné funkce zjednodušující práci s rádiovým koprocesorem se nacházejí v souborech `/source/radio_proc/radio_proc.h` a `radio_proc.c`.

4.3.1 Ovládání rádiového koprocesoru

Ovládání zmíněného koprocesoru je zpřístupněno pomocí tzv. „příkazů“. Příkazy jsou ve své podstatě struktury (`struct`) nebo „objekty“[1] s vlastními parametry. Do koprocesoru se posílají pomocí funkcí `RF_runCmd() nebo RF_postCmd()`. [4]

Úvodní inicializaci RF Core lze shrnout do následujících bodů:

1. Otevřít komunikační kanál ke koprocesoru pomocí `RF_open()`. V rámci této funkce se také nastavuje protokol, který bude rádiová komunikace využívat.
2. Naladit frekvenční syntezátor na určitý kanál spektra a nastavit další parametry spolu s frontou pro příchozí pakety.
3. Začít s přijímáním paketů a jejich zápisem do fronty.

Ke všem těmto bodům vygeneruje utilita *SysConfig* zvláštní příkazy, které je vhodné ještě před odesláním do koprocesoru marginálně upravit. Ukázková úprava výchozí hodnoty je nastíněna ve výpisu 4.1.

[1] Některé zdroje označují tyto struktury také jako „objekty“. Nejedná se však o objekty ve stejném smyslu, jako je tomu u objektově orientovaného programování.

Výpis 4.1: Změna výchozí hodnoty `whitening`

```
1 RFCMD_bleGenericRX.whitening.init = 0x65;
```

Úpravou těchto příkazových struktur lze také přiřadit frontu pro příchozí pakety nebo registrovat strukturu, kde se zapisují statistické údaje o přijatých rámcích. Obě tyto úpravy jsou zobrazeny ve výpisu 4.2.

Výpis 4.2: Přiřazení fronty a výstupní statistické struktury

```
1 RFCMD_bleGenericRX.pParams->pRxQ = RadioQueue_getDQpointer();
2 RFCMD_bleGenericRX.pOutput = &statistics;
```

Takto upravené příkazy jsou pak odeslány do koprocesoru buď pomocí funkce `RF_postCmd()`, která zařadí příkaz do fronty rádiových příkazů a nečeká na jeho provedení, nebo funkcí `RF_runCmd()`, která opět zařadí příkaz do fronty, na dokončení však již počká.

Pro zjednodušení a lepší čitelnost kódu byly tyto příkazy seskupeny a zabaleny do funkcí s přehlednějšími názvy, například `Radio_SetUpAndBeginRx()` nebo `Radio_StopRx()`.

RF Core poskytuje velké množství příkazů pro každý protokol, který je na tomto zařízení podporován. V souvislosti s BLE nabízí například příkaz pro skenování spektra nebo pro start BLE MASTER módu. Seznam těchto příkazů je uveden v [4]. Pro tento projekt jsou však klíčové příkazy `CMD_BLE5_GENERIC_RX` a `CMD_IEEE_RX`, které uvedou koprocesor do módu neustálého naslouchání pro dané příchozí pakety.

4.3.2 Fronta pro příchozí rámce

Zdrojový kód fronty je převzatý z firmwaru TI Packet Sniffer 2. Zvláštností na implementaci této fronty může být fakt, že z ní uživatel (programátor) může pouze číst, nikoliv do ní zapisovat. Zapisuje zde pouze jádro RF Core.

Fronta má kapacitu tři bezdrátové rámce. Každý z těchto rámců může mít maximální délku 2046 bajtů. Z hlediska datových struktur je fronta tvořena kruhovým spojovým seznamem, kdy každý záznam obsahuje mimo jiné i odkaz na záznam následující.

Signalizace obsazenosti slotu je prováděna pomocí bajtu `status`, jehož stavy mohou být

- `DATA_ENTRY_PENDING` – volný slot ve frontě,
- `DATA_ENTRY_FINISHED` – plný slot, který je připraven na čtení,
- a další.

Za běhu programu do této fronty rádiový koprocesor neustále zapisuje přijaté rámce a hlavní funkce vlákna (viz sekce 4.3.3) tyto rámce zase neustále čte a vybírá. Pokud

by byl zápis rychlejší než čtení, dojde k zaplnění fronty a vyvolání přerušení, které je obslouženo velice primitivně – pouhým vyprázdněním fronty. Dojde tedy ke ztrátě tří rámců. Během testování však tato událost nenastávala často – toto elementární řešení lze tedy označit za dostačující.

4.3.3 Hlavní funkce vlákna

Vývojový diagram hlavní funkce vlákna je patrný na obr. C.2a. Funkcionalita je velmi jednoduchá – po inicializaci fronty pro příchozí rámce a spuštění zachytávání spadne vlákno do nekonečné smyčky, ve které kontroluje byl-li do fronty rádiovým koprocesorem zapsán nový BLE, nebo IEEE 802.15.4 rámec. Pokud ano, překopíruje se z fronty do lokální proměnné a započne konstrukce UDP datagramu. Tomu je nejprv nastavena cílová IP adresa a port. Poté dojde ke zkopírování přístupové adresy pro „advertising“ kanály `0x8E89BED6` [1] (pouze jsou-li zachytávány BLE rámce). V poslední řadě je do datagramu zkopírován rámec z proměnné a datagram odeslán.

4.4 Vlákno Dashboard

Hlavním úkolem tohoto vlákna je zpřístupnit uživateli dálkové ovládání a konfiguraci přípravku přes jednoduché webové rozhraní. V rámci vlákna běží elementární HTTP server, který odpovídá na požadavky uživatele a také mění nastavení programu.

Hlavní funkci vlákna a funkce pomocné lze nalézt v souboru `/dashboard_task.h` a `dashboard_task.c`. Funkce pro manipulaci s HTML kódem jsou pak implementované v `/source/html/html.h` a `html.c`. HTML šablona se nachází v adresáři `/flashbins/index.html`. Před nahráním do zařízení je potřeba tento soubor konvertovat to binárního formátu a na začátek přidat HTTP hlavičku. O zmíněné kroky se postará Python skript `/flashbins/convert_to_bin.py`. Nahrání samotné obstará skript `/flashbins/load_bin.js`, který se spouští prostřednictvím skriptovací konzole v CCS.

4.4.1 Hlavní funkce vlákna

Vývojový diagram hlavní funkce je zobrazen na obr. C.2b. V úvodu je spuštěno naslouchání na portu 80, který je typický pro nešifrovaný[2] HTTP protokol. Program následně padá do nekonečné smyčky, kdy periodicky kontroluje, zda byl zaregistrován požadavek o připojení na server (TCP SYN). Pokud ano, načte tento požadavek

[2]V tomto případě není nutné komunikaci zabezpečovat pomocí TLS.

do paměti, obslouží případné příkazy (viz sekce 4.4.2) a zašle naformátovanou odpověď (viz sekce 4.4.3). Obsluha některých příkazů může vyžadovat restart zařízení, z tohoto důvodu je po odeslání odpovědi zkontrolován signalizační bajt Status Vektorů. Pokud je restart skutečně vyžádán, v této chvíli už může být proveden. Pokud vyžádán není, program aktualizuje OLED displej a vrátí se na začátek nekonečné smyčky.

4.4.2 Obsluha příkazů

Zařízení lze ovládat pomocí příkazů zasílaných přímo v těle URL jako parametry (viz výpisy 4.3 a 4.4)[3]. Tyto příkazy mají formu „`<klíč>=<hodnota>`“, což úzce souvisí se *slovníkem parametrů*, který bude rozebrán v následující sekci. Od adresy přípravku příkazy odděluje znak „`?`", mezi sebou jsou pak odděleny znakem „`&`“. Seznam všech příkazů, na které zařízení reaguje, lze vyčíst z tabulky B.1 (ve sloupci API s hodnotou „A“).

Příkazy lze do zařízení posílat přímo, tedy jejich manuálním zadáním do adresního pole prohlížeče, nebo prostřednictvím webového formuláře. Preferovaná je druhá varianta – je bezpečnější a uživatelsky přívětivější.

V následujících výpisech jsou nastíněny ukázky ovládání zařízení pomocí dostupných příkazů. IP adresa přípravku je `192.168.5.23`.

Výpis 4.3: Ukázkový příkaz č. 1

```
http://192.168.5.23?t=192.168.5.2&r=1
```

Výpis 4.3 obsahuje příkaz pro nastavení IP adresy cílového zařízení, kam bude přípravek zasílat zachycené bezdrátové rámce (`t=192.168.5.2`) a příkaz pro spuštění zachytávání (`r=1`).

Výpis 4.4: Ukázkový příkaz č. 2

```
http://192.168.5.23?p=1&k=11
```

Druhý ukázkový příkaz ve výpisu 4.4 přenastavuje bezdrátový protokol na IEEE 802.15.4 (`p=1`) a kanál 11 (`k=11`).

Jak již bylo zmíněno, mnohem pohodlnější variantou ovládání je využití webového formuláře. Ten se zobrazí po zadání IP adresy přípravku do adresního pole webového prohlížeče. Ovládání zařízení pomocí webového rozhraní je blíže popsáno v sekci 4.5.

[3]Pro komunikaci je použita výhradně metoda HTTP metoda GET, primárně pro svou jednoduchost.

4.4.3 Konstrukce odpovědi

Šablona webového rozhraní ve flash paměti zařízení z principu nemůže obsahovat – z hlediska běhu programu – aktuální informace. Z tohoto důvodu byly do HTML i JavaScript kódu šablony vloženy zástupné symboly „$“ následované malým písmenem („a“ až „z“) na místa, kde by měl uživatel vidět reálnou a aktuální hodnotu, jak je vyobrazeno ve výpisu 4.5[4].

Výpis 4.5: Část HTML kódu se zástupným znakem

```
<input type="text" id="tgtip" name="t" value="$t">
```

Tyto zástupné znaky se během procesu konstrukce odpovědi nahradí reálnými hodnotami a k uživateli se dostane kód podobný tomu z výpisu 4.6

Výpis 4.6: Část HTML kódu s reálnou hodnotou

```
<input type="text" id="tgtip" name="t" value="192.168.5.2">
```

Zmíněná záměna je umožněna díky implementaci *slovníku parametrů*. Tato datová struktura propojuje klíč s hodnotou, kdy klíčem je unikátní malé písmeno („a“ až „z“)[5] a hodnotou libovolný řetězec s délkou až 18 znaků. Během procesu formátování se slovník naplní aktuálními hodnotami na základě každého klíče (viz tabulka B.1). Například klíči „`t`“ je během aktualizace přířazen řetězec „`192.168.5.2`“.

Po této aktualizaci slovníku je možné přejít k samotné konstrukci výsledného HTML kódu, který je ve finále odeslán klientovi. K tomuto účelu byl v RAM paměti zařízení vyhrazen paměťový „MTU“ sektor, do kterého je po částech kopírován HTML kód z flash paměti. Kopírování probíhá znak po znaku – pokud je však kopírovaným znakem zástupný symbol „$“, je místo něj do MTU sektoru injektována hodnota ze *slovníku parametrů*. Jako klíč je použit znak, který bezprostředně následoval po zástupném symbolu „$“. Kopírování pak pokračuje, dokud není MTU sektor zaplněn, poté je celý odeslán klientovi. V následující iteraci se v rámci HTML kódu pokračuje tam, kde předchozí iterace skončila. Po částech je tak odeslán kód celého webového rozhraní.

Toto řešení má zásadní výhodu v podobě robustnosti a adaptability. HTML kód je zcela nezávislý na zdrojovém kódu, může být libovolně dlouhý a jakákoliv jeho aktualizace není podmíněna rekompilací. Dále je vhodné zmínit, že díky jednoduché indexaci na základě ASCII znaků má proces injektování hodnot ze slovníku parametrů konstantní složitost, což značně zefektivňuje proces konstrukce kódu webového rozhraní.

[4] V komponentě `input` se v rámci šablony vyskytují hodnoty určené i ke čtení, nikoliv jen k zápisu.

[5] Zjednodušeno. Klíčem je číselná hodnota daného znaku z ASCII tabulky.

4.5 Webové rozhraní

Hlavní ovládací panel byl vyvinut s důrazem na jednoduchost a informativnost. Díky spojení více technologií poskytuje bezpečný způsob pro ovládání přípravku. V následujícím textu bude podrobně popsána každá sekce.

4.5.1 Sekce *Network*

Tato sekce poskytuje síťové nastavení zařízení samotného. V první části může uživatel zvolit, zdali si má přípravek vyžádat síťové informace od DHCP serveru, nebo mu budou přiřazeny staticky. Výběr je uskutečněn pomocí přepínače, jehož stav následně reflektují další vstupní pole. Je-li zvolena možnost „DHCP“, jsou všechna další pole znepřístupněna, protože nejsou potřeba. Je-li zvolena možnost „Static“, jsou všechna další pole (kromě pole MAC adresy, které má *pouze* informativní povahu) zpřístupněna a lze do nich vepsat žádanou hodnotu. Tuto funkcionalitu zajišťuje JavaScript.

Všechna vstupní pole pro IP adresy jsou vybavena kontrolou pomocí regulárního výrazu. Ta eliminuje možnost, že by se do zařízení dostal příkaz ve špatném formátu. Pokud tedy uživatel vykoná při zápisu adresy nebo masky překlep, je na to upozorněn již webovým prohlížečem (viz obr. 4.1) a před odesláním je nucen chybu opravit. Celá sekce *Network* je zobrazena na obr. 4.2.

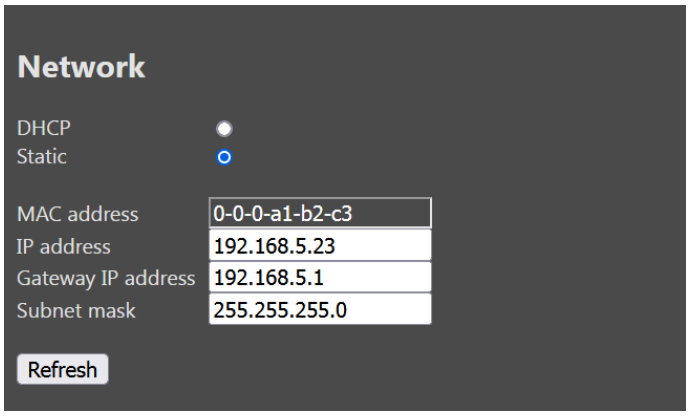
Obr. 4.1: Překlep při nastavování IP adresy

4.5.2 Sekce *Sniffing*

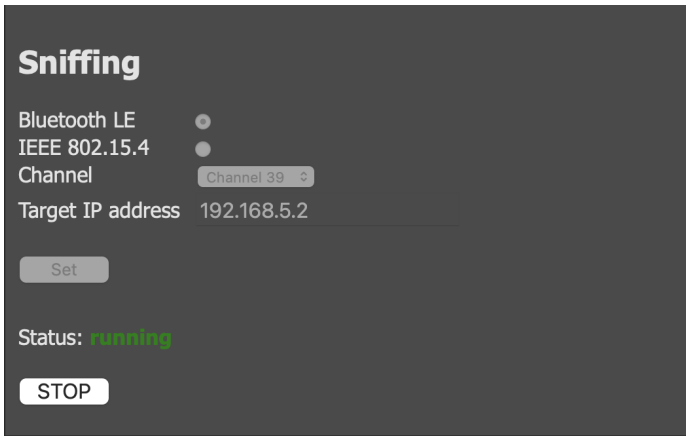
Ovládací prvky této sekce upravují parametry zachytávání a adresu cílového počítače, kam zařízení posílá zachycené bezdrátové rámce. Uživatel má možnost zvolit bezdrátový protokol (BLE, nebo IEEE 802.15.4), kanál, na kterém zařízení naslouchá[6], IP adresu cíle a zachytávání spustit, či zastavit. Opět je zde využit JavaScript, díky kterému jsou HTML prvky dynamické a aktuální (mění barvu i obsah podle současného stavu přípravku).

Tuto sekci lze vidět na obr. 4.3.

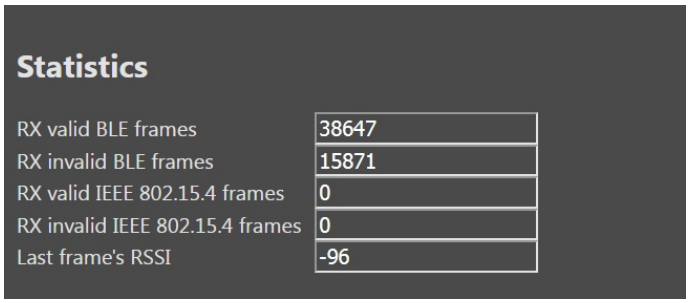
[6]Přepínat mezi protokoly a kanály lze pouze, je-li zachytávání *zastaveno*.



Obr. 4.2: Webové rozhraní: sekce *Netowrk*



Obr. 4.3: Webové rozhraní: sekce *Sniffing*



Obr. 4.4: Webové rozhraní: sekce *Statistics*

4.5.3 Sekce *Statistics*

K informativním a diagnostickým účelům byla do webového rozhraní přidána i sekce obsahující statistická data. Aktualizace probíhá při načítání celé webové stránky, platnost zmíněných dat je tedy značně krátkodobá – vzhledem k jejich povaze to však nepředstavuje velký problém.

Mimo počet úspěšně přijatých rámců je zaznamenáván i počet rámců přijatých s CRC chybou. Dále je pak uchovávána informace o intenzitě příchozího signálu, která se vztahuje vždy k poslednímu příchozímu rámci.

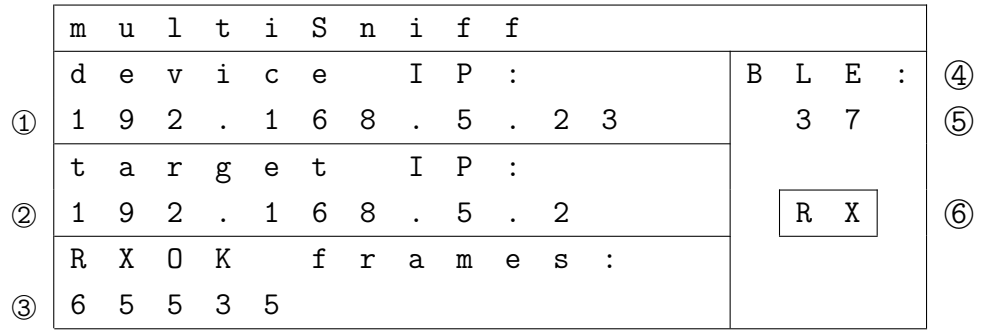
Všechny tyto informace jsou k vidění na obr. 4.4.

4.6 Ovladač OLED displeje

OLED displej představuje velice jednoduchý prostředek pro získání elementárních informací o aktuálním stavu zařízení. Jak je patrno ze schématu na obr. 3.1, byl použit model SSD1306 komunikující přes I2C. Nízkoúrovňový ovladač je implementován v `/source/driverlib/ssd1306.h` a `ssd1306.c`. Komplexní ovladač grafického rozhraní se pak nachází v `/source/oled_gui/gui.h` a `gui.c`.

4.6.1 Grafické rozhraní

Na obr. 4.5 a fotografii E.1 je znázorněno grafické rozhraní, které je za běhu programu zobrazeno na OLED displeji, je-li připojen. Nejdůležitější informací je IP adresa přípravku samotného ①. Po zadání této adresy do adresního pole prohlížeče se zobrazí webové rozhraní a zařízení je možno hlouběji nastavit. Na IP adresu cílového zařízení ② bude přípravek posílat zachycené rámce. Dále je na displeji zobrazen i počet bezchybně zachycených rámců ③, který se vztahuje k právě nastavenému protokolu ④. Nechybí ani informace o nastaveném kanálu, na kterém přípravek naslouchá ⑤ a kontrolka, zdali je zachytávání aktivní ⑥.



Obr. 4.5: Vizualizace grafického rozhraní OLED displeje

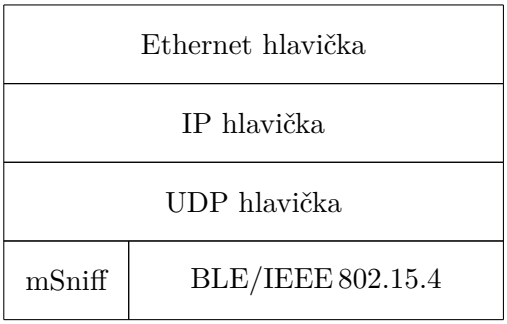
4.7 Wireshark a protokol *mSniff*

Jak již bylo naznačeno v sekci 4.3, přípravek zachytává bezdrátové rámce a vkládá je do datové sekce UDP datagramů, které následně posílá do cílového počítače, kde je analyzuje aplikace Wireshark. Tato aplikace již v čisté instalaci obsahuje dissectory, které dovedou analyzovat BLE i IEEE 802.15.4 rámce a poskytnout uživateli hlubší vhled do probíhající komunikace. Problémem však je, že tyto dissectory nejsou volány automaticky – Wireshark tedy zobrazí zachycené rámce pouze jako „payload“ UDP datagramů.

Pro ošetření tohoto chování byl vyvinut velice jednoduchý protokol a k němu odpovídající dissector. Díky těmto funkčním prvkům dovede Wireshark rozlišit, zdali daný UDP datagram obsahuje BLE či IEEE 802.15.4 rámec a podle toho zařídit další procedury tak, aby byl uživateli prezentováno co nejvíce informací o zachycené bezdrátové komunikaci.

4.7.1 Protokol *mSniff*

Hlavní funkcí tohoto protokolu je rozlišit, zdali daný datagram obsahuje BLE nebo IEEE 802.15.4 rámec. Protokol má podobu pouze jednoho bajtu (viz obr. 4.6), který se nachází za posledním bajtem UDP hlavičky, tudíž jedná se o první datový bajt UDP zprávy. Nabývá pouze dvou hodnot – `0` pro BLE rámec, `1` pro IEEE 802.15.4 rámec. Tato hodnota je zapsána mikrokontrolérem před odesláním datagramu cílovému počítači. Na jejím základě pak dissector rozhodne o dalších procedurách.



Obr. 4.6: Znázornění zapouzdření bezdrátového rámce do mSniff protokolu

4.7.2 Dissector

Dissector, jehož implementaci lze nalézt v `/wireshark/msniff.lua` a který byl vyvinut v jazyce Lua, funguje v tandemu se zmíněným protokolem. Reaguje pouze na UDP datagramy s příchozím portem 2014 – přečte první bajt datové sekce (pole

„mSniff“ z obr. 4.6) a podle jeho hodnoty předá zbytek zprávy dalšímu dissectoru (buď BLE – `btle`, nebo IEEE 802.15.4 – `wpan`). Funguje tedy pouze jako jakýsi „parser“, jeho primárním úkolem je volat další, složitější dissectory. Jeho sekundární funkce pak tkví v označení datagramů obsahujících zachycené bezdrátové rámce. Díky tomu je pak možné proud příchozích a odchozích zpráv ve Wiresharku filtrovat pomocí názvu protokolu `„msniff"`. Vyfiltrovaný proud pak bude obsahovat pouze relevantní zprávy (viz sekce 5.1.2).

5 Zachytávání

Přípravek je možné využít v zásadě dvěma způsoby. Tou standardní možností je lokální zachytávání, tedy použití v rámci lokální sítě, kdy se cílový počítač s nainstalovanou aplikací Wireshark nachází v téže síti jako přípravek samotný. Tou druhou, méně standardní, variantou je přípravek nainstalovat v nějaké lokalitě a nakonfigurovat síťové prvky tak, aby bylo možné zachycené rámce pozorovat vzdáleně.

5.1 Lokální zachytávání

5.1.1 Úvodní konfigurace

Zařízení má v implicitním stavu nastaveno, aby se do sítě připojilo s využitím DHCP. Ještě před tímto připojením je tedy vhodné nastavit DHCP server dané sítě tak, aby zařízení mSniff vždy přiřadil tutéž IP adresu. Toho lze docílit pomocí jednoduchého pravidla, které propojí MAC adresu přípravku s designovanou IP adresou.

Po získání IP adresy ji zařízení zobrazí na svém OLED displeji (viz sekce 4.6.1). Zadáním této adresy do adresního pole libovolného internetového prohlížeče se uživatel dostane k webovému rozhraní (viz sekce 4.5), přes které zařízení nakonfiguruje. Nejprve je nutné nastavit požadovaný protokol a kanál, následně pak IP adresu cílového počítače, na který budou směřovat UDP datagramy se zachycenými bezdrátovými rámci. Po tomto nastavení lze tlačítkem „START“ spustit zachytávání. Pro ukončení sezení lze z webového rozhraní zachytávání taktéž ukončit.

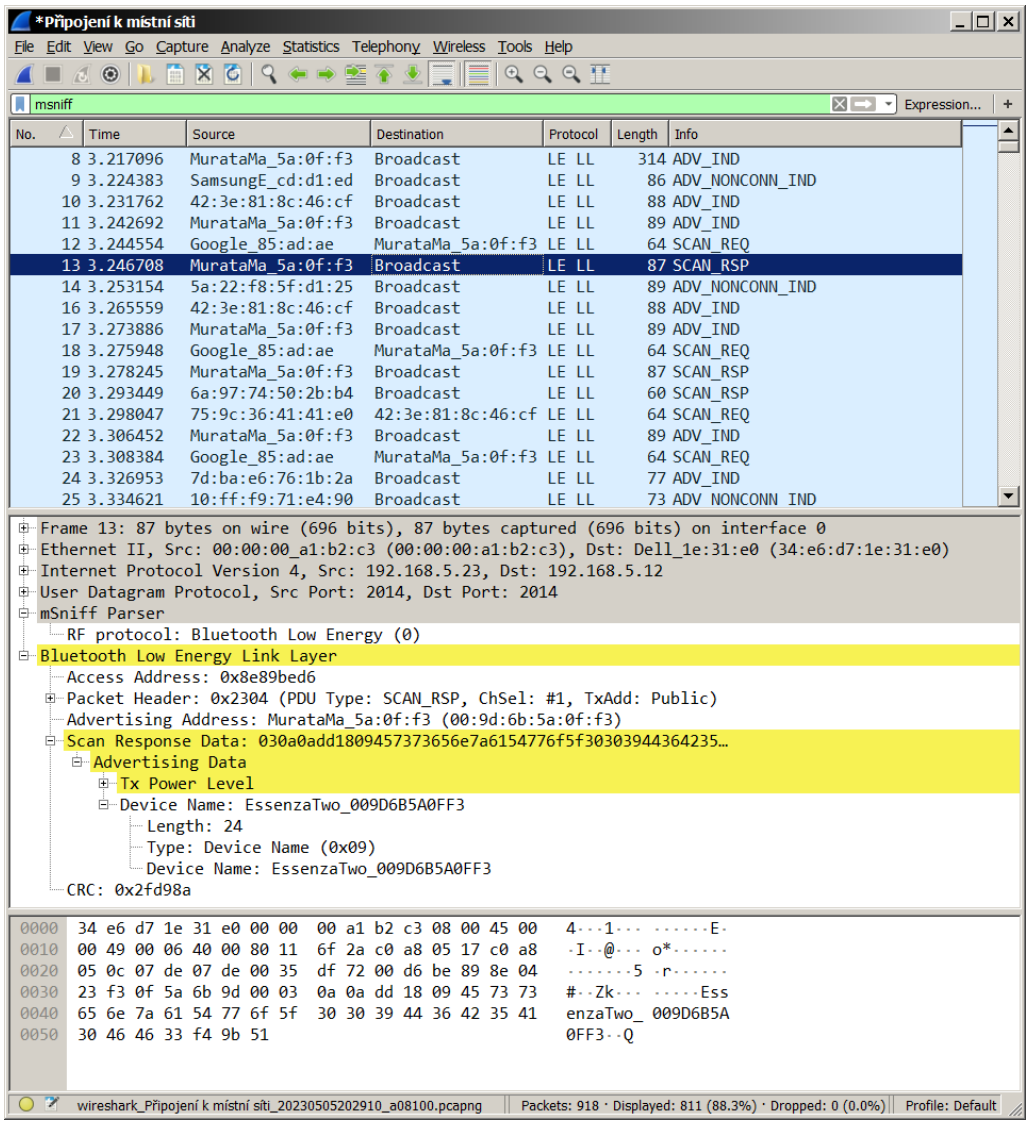
5.1.2 Zobrazení jednotlivých rámců

Po úvodní konfiguraci a spuštění zachytávání začne zařízení na adresu cílového počítače odesílat UDP datagramy obsahující zachycené bezdrátové rámce. Pro jejich zobrazení je nutné spustit aplikaci Wireshark s nainstalovaným dissectorem (`msniff.lua`) a následně zvolit síťové rozhraní, jehož IP adresa byla nastavena během úvodní konfigurace. V této chvíli je vhodné vyfiltrovat proud zpráv pouze na ty relevantní pomocí názvu protokolu („`msniff`“).

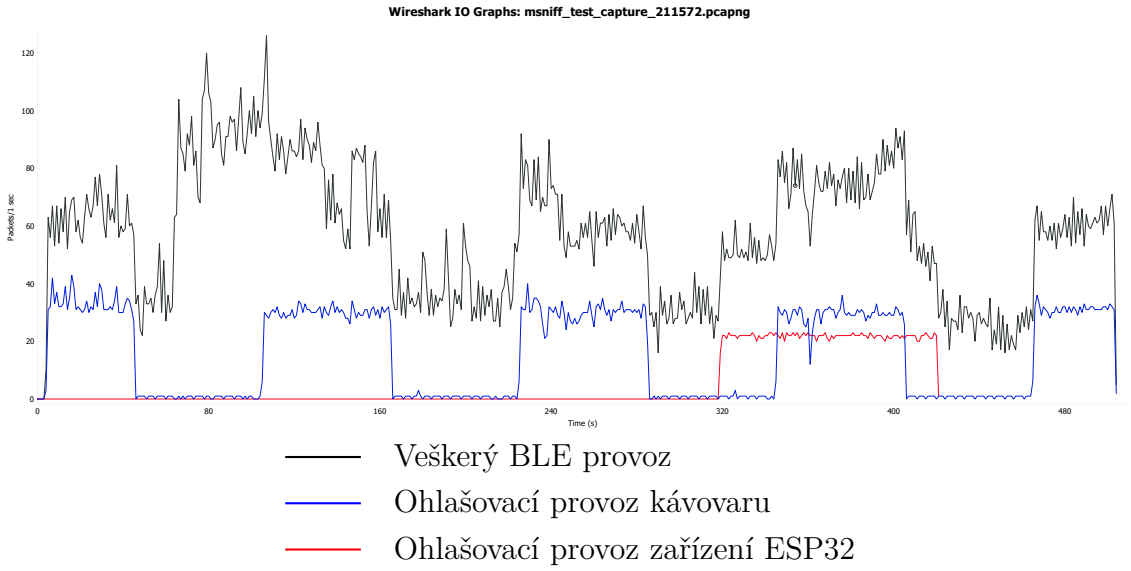
Konfigurace Wiresharku je nyní dokončena a proces zachytávání je spuštěn. Testování proběhlo v lokální síti. Výsledky tohoto testování jsou zachyceny na obr. 5.1

5.1.3 Zobrazení četnosti rámců

Mimo výpis příchozích rámců dovede Wireshark poskytnout i jiné analytické nástroje. Praktický je graf četnosti příchozích rámců (`Statistcs → I/O Graph`), který zobrazuje počet přijatých rámců za jednotku času. Lze tak pozorovat trend zaplnění



Obr. 5.1: Zachytávací prostředí pro BLE rámce



Obr. 5.2: Graf četnosti BLE rámců v průběhu času

daného kanálu (ať už BLE nebo IEEE 802.15.4). Opět je vhodné provoz vyfiltrovat pomocí názvu protokolu („`msniff`“).

Na obr. 5.2 je zmíněný graf zobrazen. Zachytávání trvalo přes 8 minut (přibližně 500 vteřin) a odehrávalo se v domácích podmínkách. Jako generátory BLE provozu byla využita zařízení, která se v domácím prostředí běžně nachází (mobilní telefony, bezdrátová sluchátka, „chytrý“ kávovar apod.). Chování těchto zařízení však nelze deterministicky ovlivnit – z tohoto důvodu byl vytvořen testovací přípravek využívající mikroprocesor ESP32, jehož jediným úkolem bylo periodicky posílat BLE ohlašovací rámce. Přípravek byl uveden do provozu přibližně v 320. vteřině zachytávání, jak lze vidět na zmíněném grafu při pohledu na červenou spojnici. Zapnutí testovacího přípravku se zcela jasně projevilo i na celkovém provozu, jak lze vidět ze spojnice černé. Nejzajímavější je však modrá spojnice, která znázorňuje ohlašovací provoz „chytrého“ kávovaru. Její průběh by se dal označit jako periodický obdélníkový signál s periodou $T \doteq 2\,\mathrm{m}$ a střídou 0,5. Z toho lze usuzovat, že kávovar co minutu střídá ohlašovací a tichý mód. I tento dílčí provoz se výrazně projeví na provozu celkovém, jak lze vidět na černé spojnici.

5.2 Vzdálené zachytávání

5.2.1 Úvodní konfigurace

Schéma typické topologie pro vzdálené zachytávání se nachází na obr. 5.3. Pro správnou funkčnost vzdálené kontroly i zachytávání samotného je nutné provést konfigu-

raci zařízení i hraničních síťových prvků – směrovačů.

Na směrovači v síti u přípravku (směrovač 1) je třeba povolit přesměrování portů (*port forwarding*) tak, aby se příchozí HTTP GET požadavek na portu 80 skutečně dostal až k zařízení. Pravidlo bude mít tuto podobu:

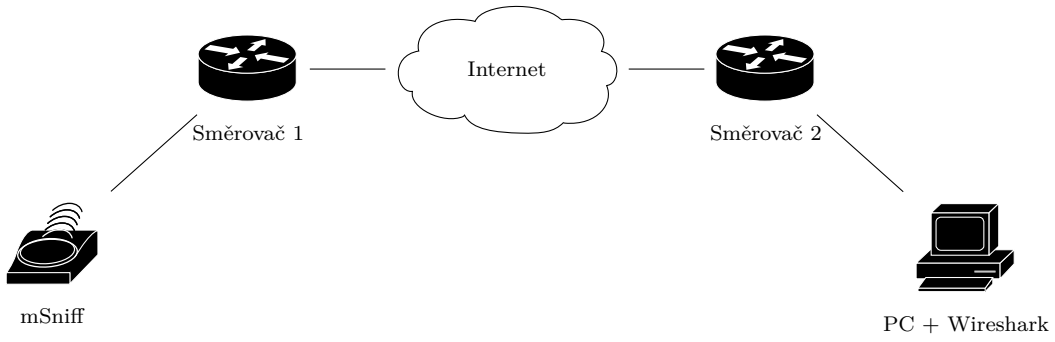
Port	Protokol	IP
80	tcp	lokální_ip_přípravku:80

Tímto pravidlem byla zajištěna funkčnost ovládacího webového rozhraní. Obdobnou konfiguraci bude potřeba provést i na směrovači u cílového počítače (směrovač 2), zde bude také nutné zapnout přesměrování portů s tímto pravidlem:

Port	Protokol	IP
2014	udp	lokální_ip_počítače:2014

Díky této konfiguraci se pak zachycené bezdrátové rámce zapouzdřené v UDP datagramech dostanou přes směrovač až k cílovému počítači.

Do konfiguračního panelu přípravku se uživatel nyní dostane přes veřejnou IP adresu směrovače 1. Do pole „Target IP“ (viz sekce 4.5.2) je pak pochopitelně nutné nastavit veřejnou IP adresu směrovače 2.



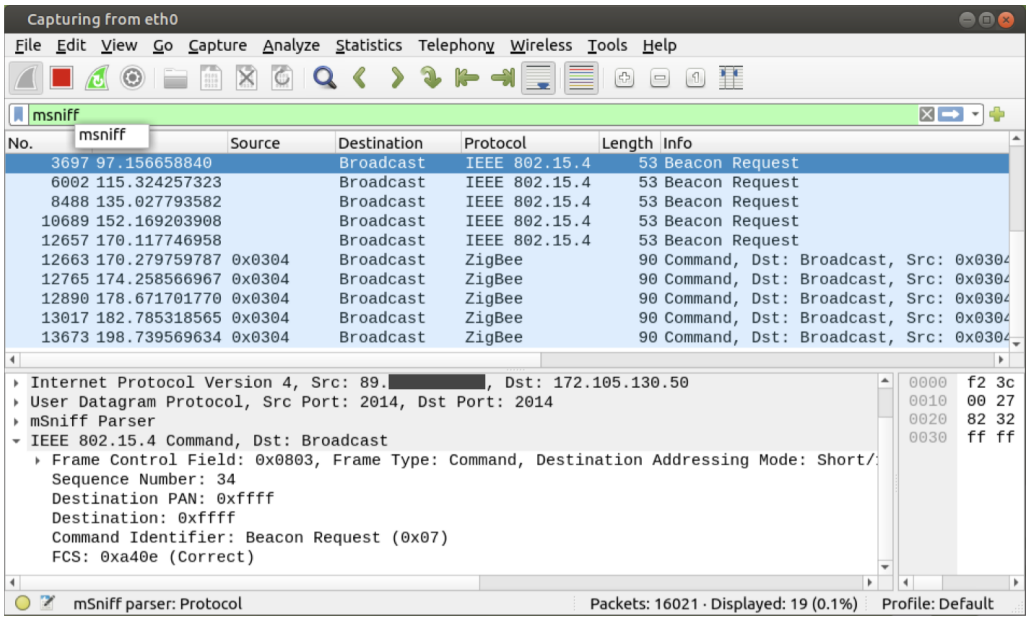
Obr. 5.3: Schéma vzdáleného zachytávání

5.2.2 Zobrazení vzdálených rámců v prostředí Wireshark

Pokud byly úspěšně provedeny všechny konfigurační kroky, je možné v aplikaci Wireshark postupovat stejně jako v sekci 5.1.2. Vzhledem k povaze UDP protokolu však nelze zaručit, že se k uživateli dostanou *všechny* zachycené rámce.

Testování vzdáleného zachytávání proběhlo prostřednictvím virtuálního privátního Linux serveru. Tento server disponoval veřejnou IP adresou, nebylo tedy nutné konfigurovat síťové prvky pro přesměrování portů. K serveru bylo přistupováno přes SSH tunel a VNC klienta.

Výsledky vzdáleného zachytávání jsou k nahlédnutí na obr. 5.4.



Obr. 5.4: Zachytávající aplikace Wireshark běžící na vzdáleném serveru

Závěr

Výstupem této diplomové práce je zařízení, které zachytává bezdrátové rámce protokolů BLE, nebo IEEE 802.15.4 a odesílá je prostřednictvím Ethernetového rozhraní do počítače, kde dochází k jejich analýze za použití aplikace Wireshark. Software zařízení byl vyvinut s důrazem na univerzálnost a modularitu. Mnoho funkcí lze přenastavit, aniž by bylo nutné upravovat a rekompilovat zdrojový kód. Webové ovládací rozhraní je také zcela nezávislé na implementaci webového serveru a snadno jej lze aktualizovat. Aby byl zachycený provoz ve Wiresharku korektně rozpoznán, byl také napsán jednoduchý dissector.

Během vývoje bylo provedeno mnoho testovacích zachytávání, výsledky několika z nich jsou v této práci prezentovány. Testy dokazují, že je program dostatečně spolehlivý a že je schopen reagovat na změny v bezdrátové síti. Zařízení dovede odesílat zachycený provoz na libovolnou IP adresu, což znamená, že použití není omezeno pouze na lokální síť – zařízení lze po konfiguraci hraničních směrovačů používat i vzdáleně.

K dalšímu vývoji se nabízí implementace TLS protokolu, který by zajistil utajení zpráv při vzdáleném zachytávání, a zefektivnění softwaru webového serveru využitím přerušení.

Literatura

[1] HEYDON, Robin. *Bluetooth low energy: the developer's handbook*. Upper Saddle River: Prentice Hall, c2013. ISBN 01-328-8836-X.

[2] LABIOD, H, H. AFIFI a C. DE SANTIS. *Wi-Fi™, Bluetooth™, Zigbee™ and WiMax™*. 1. Springer Dordrecht, 2007. ISBN 978-1-4020-5397-9.

[3] LEA, Perry. *Internet of Things for Architects*. Birmingham: Packt Publishing, 2018. ISBN 978-1-78847-059-9.

[4] TEXAS INSTRUMENTS. *CC13xx/CC26xx SimpleLink Core SDK User's Guide*. Online. Dostupné z: <https://dev.ti.com/tirex/explore/node?node=A__AGhAXIxP6gouUA7QxaW0Lg__com.ti.SIMPLELINK_CC13XX_CC26XX_SDK__BSEc4rl__LATEST&placeholder=true>. [citováno 2022-11-18].

[5] MAXIM INTEGRATED PRODUCTS, INC. *MAX77827*. Online. 2021. Dostupné z: <https://www.analog.com/media/en/technical-documentation/data-sheets/MAX77827.pdf>. [citováno 2024-05-11].

[6] TEXAS INSTRUMENTS. *CC2652RB SimpleLink™ Crystal-less BAW Multiprotocol 2.4 GHz Wireless MCU*. Online. Dostupné z: <https://www.ti.com/lit/ds/symlink/cc2652rb.pdf>. [citováno 2022-11-18].

[7] WIZNET CO., LTD. *W5500 Datasheet*. Online. 2013. Dostupné z: <https://docs.wiznet.io/img/products/w5500/W5500_ds_v110e.pdf>. [citováno 2024-05-11].

[8] INFINEON TECHNOLOGIES AG. *BGS12P2L6*. Online. 2024. Dostupné z: <https://www.infineon.com/dgdl/Infineon-BGS12P2L6-DataSheet-v02_01-EN.pdf?fileId=5546d4626cb27db2016d4487d53603ce>. [citováno 2024-05-11].

[9] SILICON LABORATORIES INC. *CP2102/9*. Online. 2017. Dostupné z: <https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf>. [citováno 2024-05-11].

[10] SOLOMON SYSTECH LIMITED. *SSD1306 Advance Information*. Online. 2008. Dostupné z: <https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf>. [citováno 2024-05-11].

[11] SYSEL, Petr. *Signálové procesory*. Brno: UTKO FEKT VUT, 2015. ISBN 978-80-214-5187-2.

[12] TEXAS INSTRUMENTS. *CC13xx/CC26xx Hardware Configuration and PCB Design Considerations*. Online. 2022. Dostupné z: <https://www.ti.com/lit/an/swra640g/swra640g.pdf>. [citováno 2024-05-11].

Seznam symbolů a zkratek

5G Technologie pro vysokorychlostní bezdrátovou komunikaci

API Application Interface – aplikační rozhraní

ARP Address Resolution Protocol

ASCII American Standard Code for Information Interchange – kódová tabulka definující znaky anglické abecedy

BC Bluetooth® Classic

BLE Bluetooth® Low Energy

cJTAG Compact JTAG

CRC Cyclic Redundancy Check – kontrolní součet

CC2652RB SimpleLink™ MCU pro IoT aplikace

CCS Code Composer Studio – IDE pro vývoj fimrwaru pro TI MCU

CSMA/CA Carrier Sense Multiple Access, Collision Avoidance – technologie mnohonásobného přístupu k bezdrátovému médiu

DHCP Dynamic Host Configuration Protocol – protokol pro dynamickou distribuci IP adresy a dalších informací

DMA Direct Memory Access

DPS Deska plošných spojů

GPIO General Purpose Input, Output

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

I2C Inter-Integrated Circuit – rozhraní pro komunikaci mezi prvky na PCB

ICMP Internet Control message Protocol

IDE Integrated Development Environment – Integrované vývojové prostředí

IEEE Institute of Electrical and Electronics Engineers

IP	Internet Protocol
IoT	Internet Of Things – internet věcí
JTAG	Joint Test Action Group – v kontextu této práce se jedná o programovací a debuggovací rozhraní
LED	Light Emmitting Diode
MAC	Medium Acces Control – obecný pojem, který charakterizuje vrstvu přistupující ke sdílenému médiu [2]
MCU	Microcontroller Unit
MTU	Maximum Transmission Unit
OLED	Organic Light-Emitting Diode – typ LED
OS	Operační Systém
PAN	Personal Area Network – zpravidla bezdrátová síť malého rozsahu
PCB	Printed Circuit Board – deska plošných spojů
PHY	zkratka označující fyzickou vrstvu v ISO/OSI modelu
POSIX	Portable Operating System Interface
RAM	Random Access Memory – Paměť s náhodným přístupem
RF	Radio Frequency
ROV	Runtime Objecet View
SDK	Software Development Kit
SoC	System-on-chip – integrovaný obvod, který kombinuje více funkčních prvků pro provoz systému na jednom čipu
SPI	Serial Peripheral Interface – sériové rozhraní pro komunikaci mezi prvky na PCB
SSD1306	Ovladač pro OLED displej
SSH	Secure Shell – protokol pro zabezpečené vzdálené ovládání serverů
TCP	Transmission Control Protocol

TCP/IP Transfer Control Protocol – protokol mj. zajišťující spolehlivé doručení odeslané zprávy

TI Texas Instruments

TI–RTOS Texas Instruments – Real Time Operating System

TLS Transport Layer Security

UART Universal Asychronous Receiver Transmitter – asynchronní komunikační rozhraní

UDP User Datagram Protocol

URL Uniform Resource Locator

USB Universal Serial Bus

VNC Virtual Network Computing – technologie pro připojení ke vzdálené ploše

W5500 WIZnet W5500

Wi-Fi™ Wireless Fidelity, standard IEEE 802.11

WPAN Wireless Personal Area Network – bezdrátové osobní sítě, sítě velmi malého rozsahu

Seznam příloh

A Status Vektory

Tab. A.1: Status Vektory

Jméno	Adresa (Flash)	Adresa (RAM)	Délka	Hodnota
MAC adresa	0x50000	-	6	např. 0x00 0x00 0x00 0x11 0x22 0x33
Použít DHCP?	0x50006	0x20013500	1	0x59 Ano | 0x3E Ne
IP adresa zařízení	0x50007	0x20013501	4	např. 192 168 5 23
IP adresa výchozí brány	0x5000B	0x20013505	4	např. 192 168 5 1
Maska sítě	0x5000F	0x20013509	4	např. 255 255 255 0
IP adresa cílového zařízení	0x50013	0x2001350D	4	např. 192 168 5 2
RF protokol	0x50017	0x20013511	1	0xB5 BLE | 0x15 IEEE 802.15.4
RF kanál	0x50018	0x20013512	1	např. 11
Aktuální status	0x50019	0x20013513	1	0x52 Zachytávání běží | 0x00 neběží
Vyžádán restart?	-	0x20013514	1	0xAA Ano | 0x00 Ne

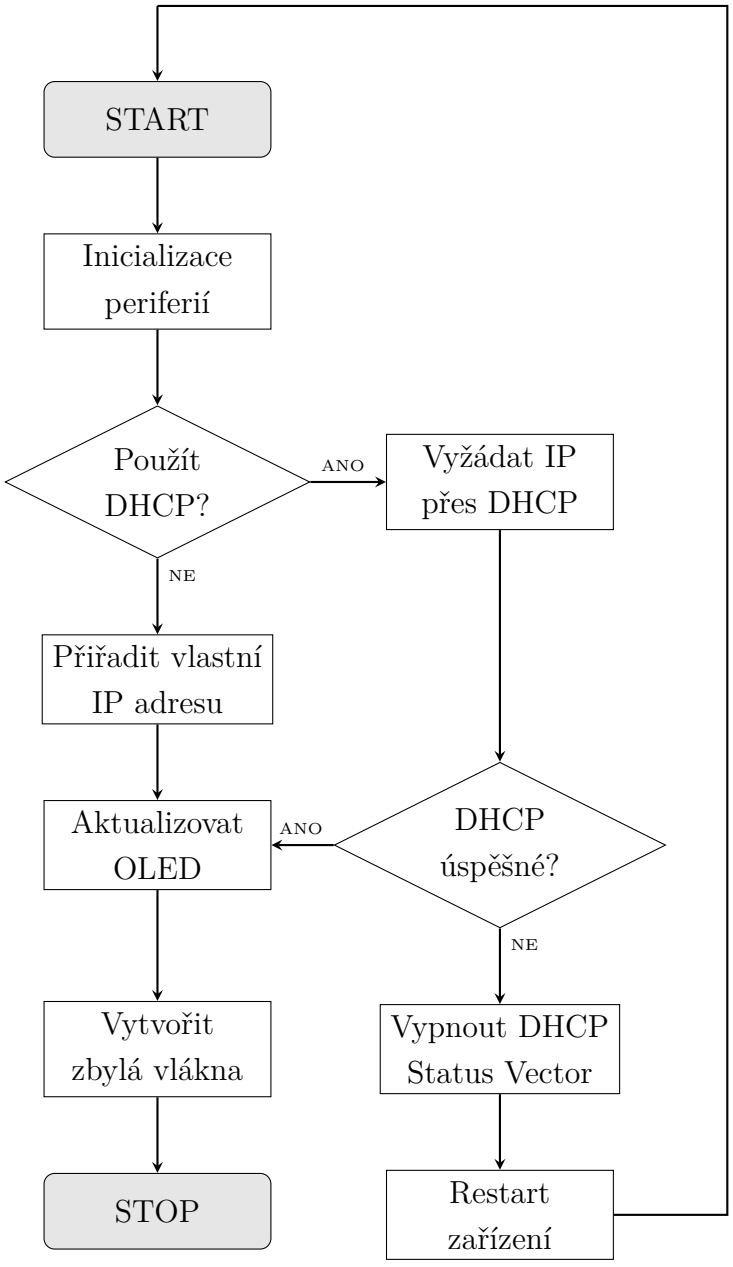
B Slovník parametrů

Tab. B.1: Obsah slovníku parametrů s popisem pro každý klíč

Klíč	Hodnota	Popis	API
`$d`	x.x.x.x	IP adresa zařízení	A
`$g`	x.x.x.x	IP adresa výchozí brány	A
`$h`	0/1	Použití DHCP	A
`$k`	[11..26], 37, 38, 39	Kanál	A
`$m`	xx-xx-xx-xx-xx-xx	MAC adresa zařízení	N
`$p`	0/1	Bezdrátový protokol	A
`$r`	0/1	Zachytávání běží/neběží	A
`$s`	x.x.x.x	Maska sítě	A
`$t`	x.x.x.x	Adresa cílového zařízení	A
`$v`	x	Počet bezchybně přijatých BLE rámců	N
`$w`	x	Počet přijatých BLE rámců s chybou	N
`$x`	x	Počet bezchybně přijatých 802.15.4 rámců	N
`$y`	x	Počet přijatých 802.15.4 rámců s chybou	N
`$z`	x	RSSI posledního přijatého rámce	N

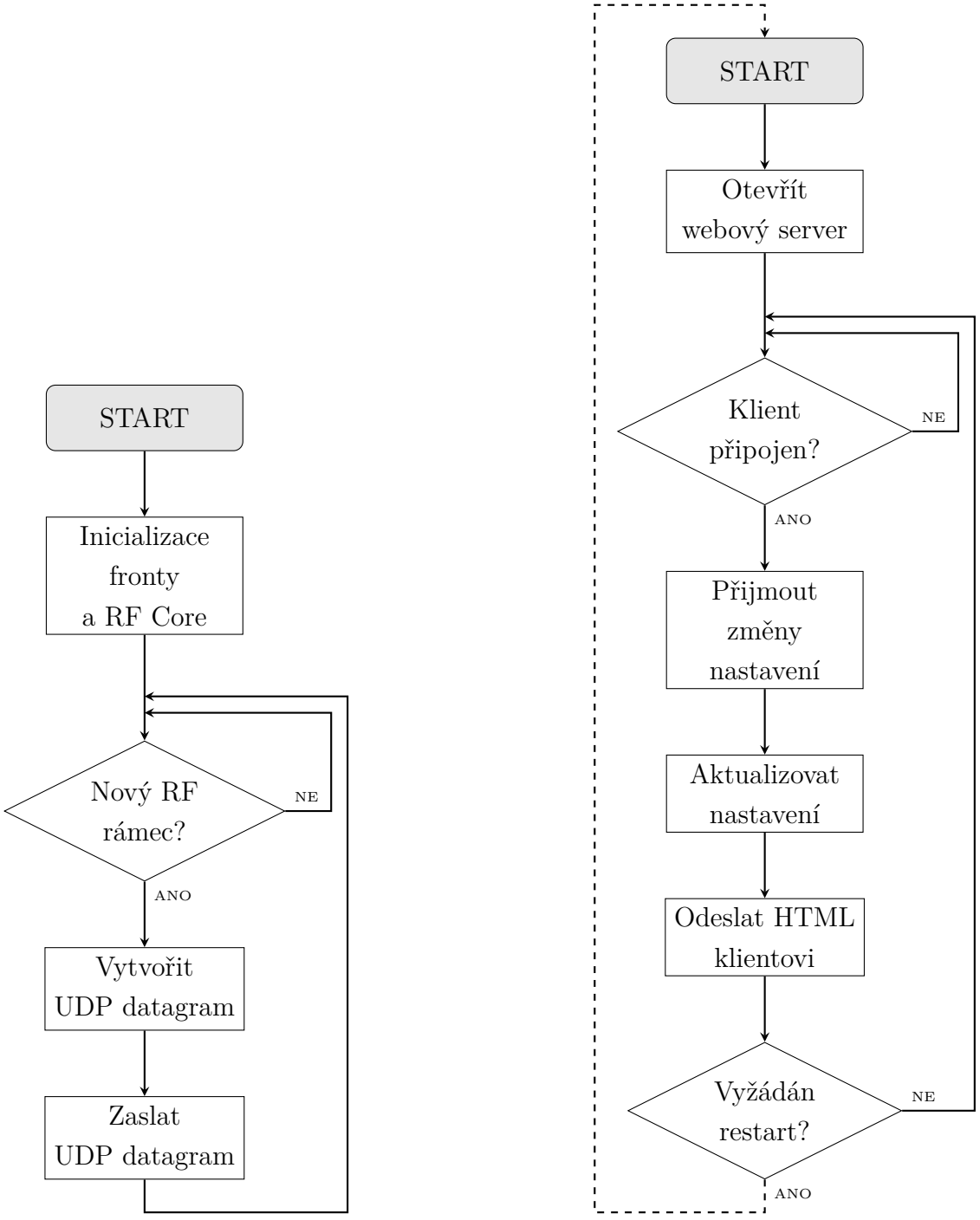
C Vývojové diagramy

C.1 Inicializační vlákno



Obr. C.1: Vývojový diagram inicializačního vlákna

C.2 Vlákno webového serveru a zachytávacího vlákna



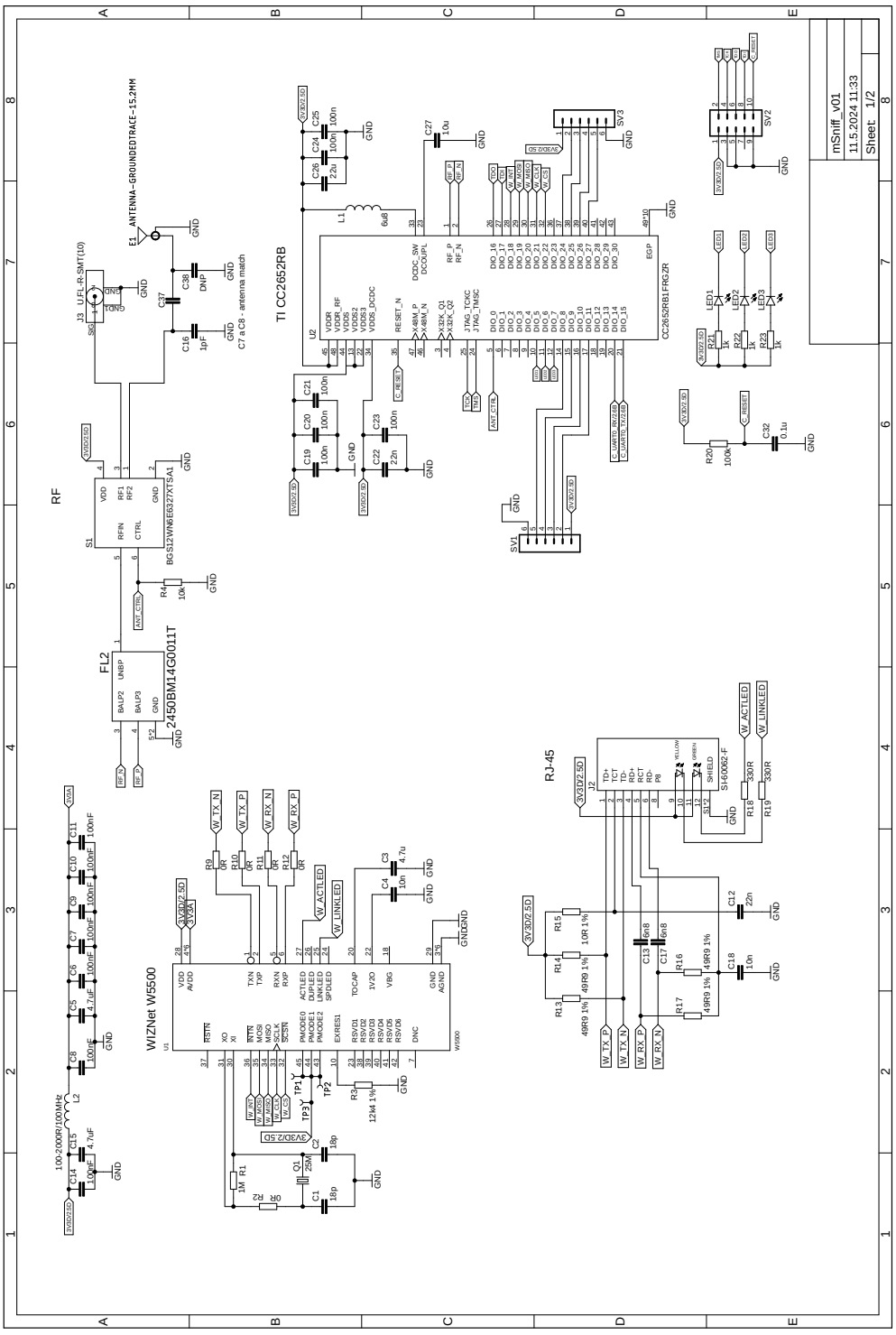
(a) Vývojový diagram zachytávacího vlákna

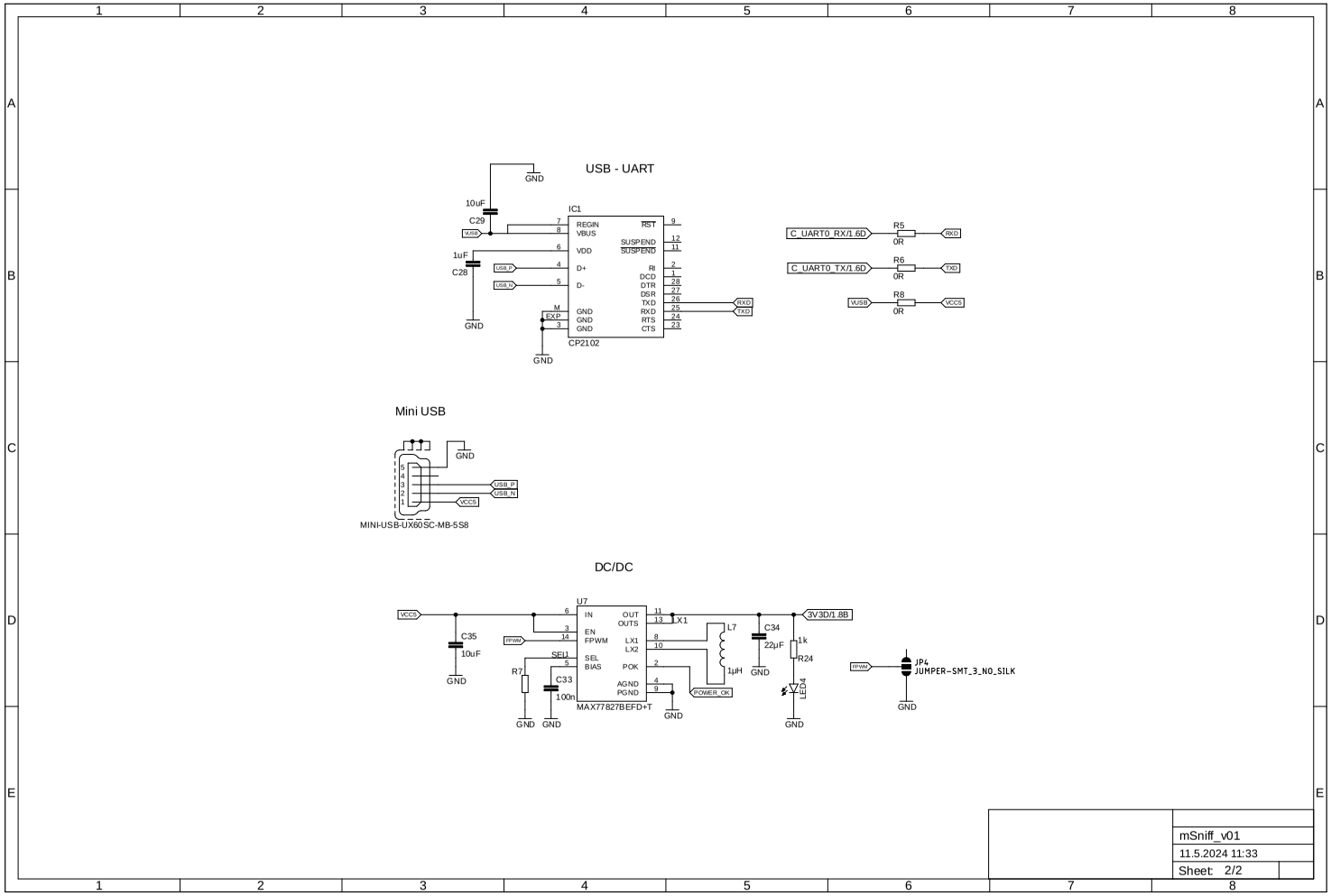
(b) Vývojový diagram vlákna webového serveru

Obr. C.2: Vývojové diagramy vlákna webového serveru a zachytávacího vlákna

D Schémata

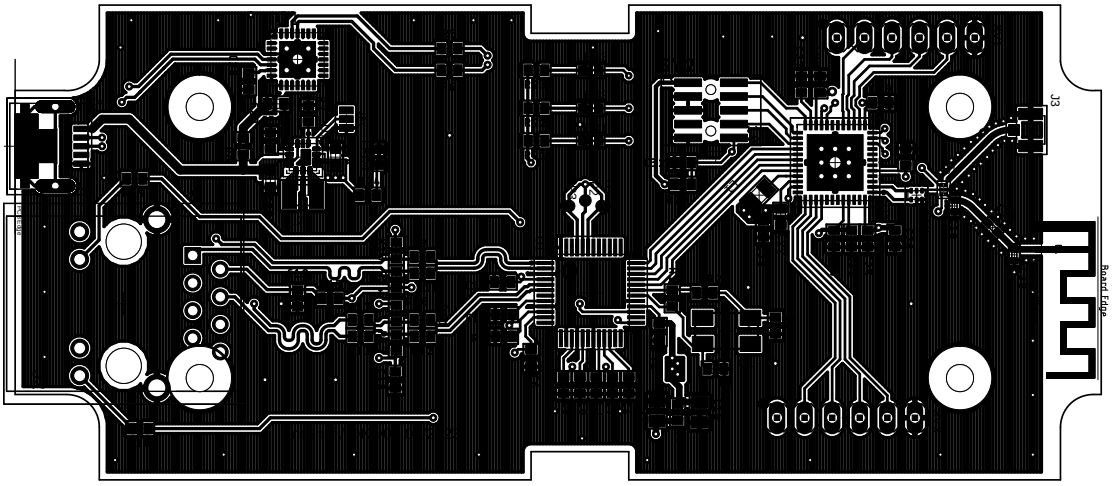
D.1 Blokové schéma



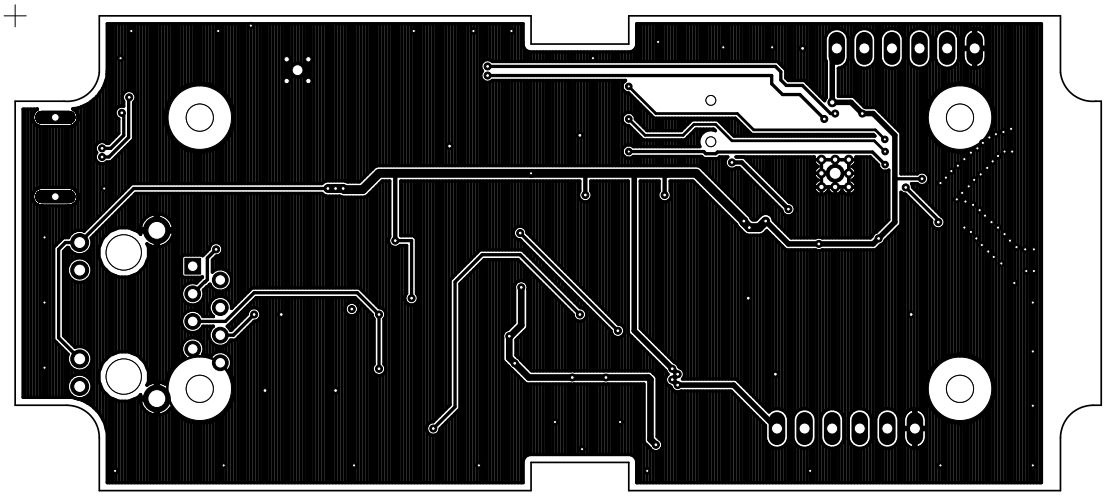

USB - UART
GND
10uF
C29
IC1
REGIN
VBUS
VDD
D+
D-
GND
GND
GND
RST
SUSPEND
SUSPEND
RI
DCD
DTR
DSR
TXD
RXD
RTS
CTS
CP2102
1uF
C28
C_UART0_RX/1.6D
C_UART0_TX/1.6D
R5
R6
R8
0R
0R
0R
RXD
TXD
VCC5
Mini USB
MINI-USB-UX60SC-MB-5S8
DC/DC
U7
MAX77827BEFD+T
C35
10uF
C33
100n
R7
L7
1µH
C34
22µF
R24
1k
LED4
JP4
JUMPER-SMT_3_NO_SILK
3V3D/1.8B
mSniff_v01
11.5.2024 11:33
Sheet: 2/2

D.2 Schéma DPS

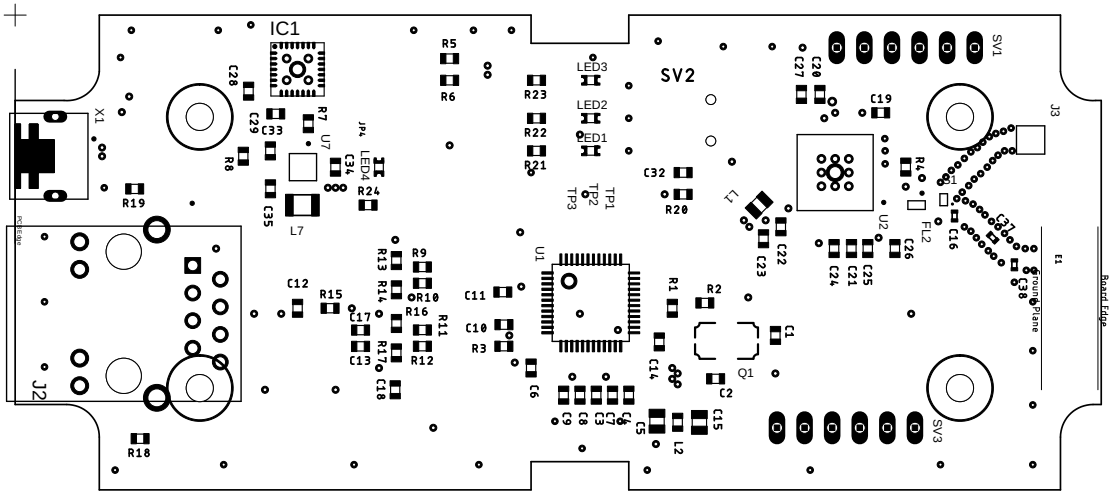
D.2.1 Horní strana DPS



D.2.2 Dolní strana DPS

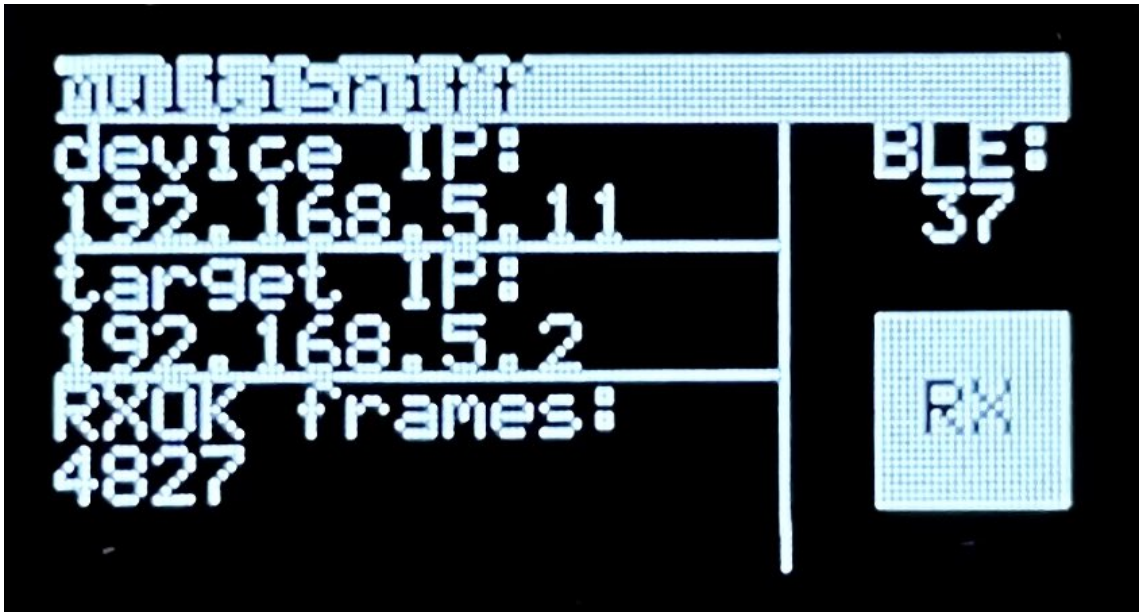


D.2.3 Rozmístění součástek



E Fotografie

E.1 Grafické rozhraní OLED displeje



Obr. E.1: Grafické rozhraní OLED displeje