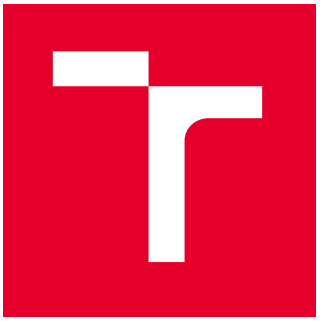


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

POROVNÁNÍ OPERAČNÍCH SYSTÉMŮ PRO EMBEDDED ZAŘÍZENÍ Z OHLEDEM NA JEJICH EFEKTIVITU A NÍZKOODBĚROVOST

COMPARISON OF OPERATING SYSTEMS FOR EMBEDDED DEVICES WITH FOCUS ON EFFECTIVITY AND LOW POWER CONSUMPTION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Filip Kounický**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Michal Mikulášek**

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Filip Kounický ***ID:*** 230597

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Porovnání operačních systémů pro embedded zařízení z ohledem na jejich efektivitu a nízkoodběrovost

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce bude teoreticky porovnat aktuálně dostupné operační systémy pro embedded jednočipové počítače, vybrat vhodné kandidáty a ověřit vhodnost na typické aplikační scénáře IoT segmentu. Hlavní důraz by měl být kladen na efektivitu zařízení s důrazem na minimalizaci energetického odběru celého zařízení pro zajištění dlouhodobého provozu na integrovanou baterii. Dále by se práce měla zaměřit na ověření podpory běžně používaných senzorů v IoT aplikacích (např.: teplotní čidla Dallas, více veličinové senzory Bosch atd). Současně by měla být ověřena softwarová podpora komunikačních modulů technologií GPS, LoRaWAN, NB-IoT, LTE CatM, WiFi, Bluetooth, ZigBee, Z-Wave. Dosažené výstupy budou přehledně zpracovány a diskutovány.

DOPORUČENÁ LITERATURA:

[1] BEAZLEY, David and Brian K. JONES, 2013. Python cookbook: No. 3: Recipes for mastering python. 3rd ed. Sebastopol, CA: O'Reilly Media. ISBN 9781449340377.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: Ing. Michal Mikulášek

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

ABSTRAKT

Tato bakalářská práce se zabývá úvodem do problematiky internetu věcí, embedded systémů a embedded operačních systémů. Představuje vlastní metriky na kvantifikaci embedded operačních systémů s ohledem na aplikaci pro IoT. Je vytvořen modelový scénář, který je implementován pro různé kombinace vybraných embedded operačních systémů a hardwarových platforem. Systematickým měřením odebíraného proudu na základě vlastních metrik je provedeno srovnání a zhodnocení embedded operačních systémů.

KLÍČOVÁ SLOVA

Arduino, Embedded systémy, IoT, Micropython, nízkoodběrovost, operační systémy, RIOT, Zephyr

ABSTRACT

This bachelor thesis deals with an introduction to the Internet of Things, embedded systems and embedded operating systems. It presents custom metrics to quantify embedded operating systems with respect to IoT applications. A model scenario is developed and implemented for different combinations of selected embedded operating systems and hardware platforms. By systematically measuring the current drawn based on custom metrics, a comparison and evaluation of embedded operating systems is performed.

KEYWORDS

Arduino, Embedded systems, IoT, Micropython, low power, operating systems, RIOT, Zephyr

KOUNICKÝ, Filip. *Porovnání operačních systémů pro embedded zařízení z ohledem na jejich efektivitu a nízkoodběrovost*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedoucí práce: Ing. Michal Mikulášek

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Filip Kounický
VUT ID autora:	230597
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Porovnání operačních systémů pro embedded zařízení z ohledem na jejich efektivitu a nízkoodběrovost

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 semestrální práce panu Ing. Michalu Mikuláškovi za odborné vedení, konzultace, vstřícnost, trpělivost a kritickou zpětnou vazbu i podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Na trhu existují desítky embedded operačních systémů a jejich různé varianty. Liší se v mnoha ohledech jako hardwarová podpora, hardwarové nároky, softwarová podpora, cílová aplikace, typ licencování a s tím spojené náklady, velikost komunity či energetická náročnost. Některé jsou v základu vhodné nebo i cílené pro IoT, jiné vyžadují modifikace různých rozsahů. V rámci této práce bude představeno několik embedded operačních systémů, z kterých bude několik vybráno pro komplexnější analýzu.

První kapitola se věnuje úvodu do problematiky internetu věcí, krátkému shrnutí historie IoT, samotnému původu výrazu „IoT“ a bezdrátovým komunikačním technologiím.

Druhá kapitola se zabývá embedded systémy a jejich typy podle složitosti, funkcionality a dynamičnosti. Koncem druhé kapitoly je krátké představení několika operačních systémů cílených na embedded zařízení.

Ve třetí kapitole budou představeny metriky pro zhodnocení vhodností embedded operačních systémů pro použití v IoT zařízeních.

Čtvrtá kapitola hlouběji analyzuje vybrané embedded operační systémy podle metrik zvolených ve třetí kapitole a zabývá se výběrem hardwarové platformy.

Pátá kapitola probírá modelový scénář pro testování.

Šestá kapitola prezentuje výsledky měření.

1 Internet věcí

Pojem IoT (Internet of Things) označuje množinu fyzických zařízení připojených do sítě, skrz kterou komunikují a vyměňují si mezi sebou data bez lidského zásahu. IoT zařízení mohou být vybavena sondami, čidly, senzory – pro sběr dat ze svého okolí, akčními členy – umožňující automatizaci. Tato zařízení lze brát jako podmnožinu embedded zařízení (viz kapitola 2), jelikož vyžadují síťovou konektivitu a tou nedisponují všechna embedded zařízení.

IoT je založeno na telekomunikaci (z řeckého *tele* – vzdálený a latinského *communicare* – sdílet [1]), konkrétně telekomunikaci využívající elektrické signály. Její historie začala vynálezem telegrafu na počátku 19. století, který umožnil přenášet zprávy na velké vzdálenosti pomocí elektrických signálů šířících se jedním vodičem. Telefon, který v roce 1876 představil Alexander Graham Bell, dále změnil komunikaci tím, že umožnil hlasové hovory. Ve 20. století došlo k rozvoji rozhlasového a televizního vysílání, které rozšířilo dosah informací po celém světě. Rozvoj internetu na konci 20. století propojil lidi a stroje po celém světě. Objevily se mobilní telefony a bezdrátové technologie, které umožnily nebývalou mobilitu v komunikaci. Ve 21. století se rozšířily chytré telefony, vysokorychlostní širokopásmové připojení a nástup 5G, které umožňují rychlejší a efektivnější přenos dat.

Jedno z prvních použití IoT bylo na začátku 80. let 20. století na Univerzitě Carnegieho–Mellonových, kde studenti modifikovali a následně připojili nápojový automat na předchůdce internetu – ARPANET, což jim umožnilo vzdáleně sledovat množství limonády a stav jejího vychlazení [2, 3]. V roce 1999 Kevin Ashton představil světu pojem IoT v rámci jeho prezentace o RFID (Radio-Frequency Identification) pro Procter & Gamble s těmito slovy: „*Dnešní počítače – a tedy i internet – jsou takřka zcela závislé na lidech, pokud jde o informace. Téměř všechna ze zhruba 50 petabajtů (petabajt je 1 024 terabajtů) data dostupná na internetu byla nejprve zachycena a vytvořena člověkem – napsáním textu, stisknutím tlačítka záznamu, pořízením digitálního snímku nebo naskenováním čárového kódu. Konvenční schémata internetu zahrnují servery, směrovače a tak dále, ale vynechávají nejpočetnější a nejdůležitější směrovače ze všech: lidi. Problém je v tom, že lidé mají omezený čas, pozornost a přesnost – to vše znamená, že nejsou příliš dobří v zachycování údajů o věcech v reálném světě.*“ [4]

Obecná očekávání od IoT zařízení jsou výdrž na baterii 10 let a více, jenž má za následek morální zestárnutí nežli fyzické, pravidelná komunikace (například zařízení, jenž je součástí senzorické sítě, bude posílat data každých 30 minut pro zachování informační relevantnosti) a deterministické chování s nulovou potřebou fyzické údržby, pouze likvidace na konci jejich životnosti. A správně kalibrované senzory kvantifikujíc realitu [5, 6].

Podle komplexity aplikace lze IoT rozdělit do 4 kategorií [7, 8]:

- Spotřební – automatizace domovů, nositelná elektronika, mobil telefon
- Komerční – automatizace budov po stránce mikroklimatu (klimatizace, topení, otvírání oken, ovládání rolet), kontroly vstupů, hospodárnosti a spotřeby
- Průmyslové – zlepšování a propojování již existujících systémů pro zvýšení efektivity a produkce
- Infrastrukturní – chytrá infrastruktura využívající IoT k zefektivnění, zlevnění provozu na základě analýzy získaných dat a následných vhodných akcích jako včasná údržba či snížení opotřebování.

1.1 Komunikační technologie v IoT

IoT zařízení používají celou škálu technologií a protokolů pro komunikaci. Většina IoT zařízení je mobilní, na vzdálených místech či jsou tak malá a početná, že by ke každému zařízení bylo přespříliš nákladné vést kabelové vedení. Ovšem jsou tu výjimky, kdy je potřeba vyšší propustnost, odolnost proti rušení nebo zařízení je energeticky náročnější. Příkladem mohou být dohledové systémy či IP kamera, která může být zároveň propojena s datovou i s energetickou sítí pomocí jediného kabelu podporujícího PoE (Power over Ethernet[1]). V současnosti převažuje bezdrátová komunikace díky její nízkonákladovosti, pohodlnosti a flexibilitě. A ty se dělí na licenční či bezlicenční podle použitého frekvenčního spektra [9].

Tab. 1.1: Přehled komunikačních technologií v IoT [10, 11, 12, 13, 14, 15, 16, 17, 18].

	Spektrum	Frekvence [MHz]
Buňková rádiová síť 1.1.1	Licenční	300–3000
LTE Cat M 1.1.2	Licenční	800–2100
NB-IoT 1.1.3	Licenční	700–2100
Wi-Fi 1.1.5	Bezlicenční	863–868, 2400, 5000
RFID 1.1.4	Bezlicenční	860–960
BLE 1.1.6	Bezlicenční	2402–2480
Zigbee 1.1.7	Bezlicenční	868, 902–928, 2400
LoRaWAN 1.1.8	Bezlicenční	433, 868, 915
Sigfox 1.1.9	Bezlicenční	868, 915

[1]PoE poskytuje napájení stejnosměrným proudem přes nevyužité kroucené páry vodičů Ethernetového kabelu typu Cat 5 a vyšší

1.1.1 Buňková rádiová síť

Neboli mobilní licencované rádiové sítě jsou téměř všude přítomné a poskytují spolehlivé a širokopásmové připojení. Velkou nevýhodou jsou její vyšší energetické nároky a provozní náklady, proto se nejedná o kompatibilní technologii pro bateriově napájená zařízení. Ovšem pro IoT zařízení napájená z energetické rozvodné sítě je to vhodné řešení. Poloměr jedné buňky se pohybuje od desítek metrů (femtobuňka) až po desítky kilometrů (makrobuňka) [10] .

1.1.2 LTE Cat M

LTE Cat M (Long-Term Evolution Caterogy Machine) je úzkopásmová buňková komunikační technologie typu LPWAN (Low-Power Wide Area Network) zaměřená na IoT. Funguje ve stávajících sítích LTE a poskytuje energeticky efektivní připojení pro zařízení, která vyžadují dlouhou výdrž baterie, rozšířené pokrytí a spolehlivou komunikaci [11].

1.1.3 NB-IoT

NB-IoT (Narrowband Internet of Things) je buňková komunikační technologie typu LPWAN navržená speciálně pro připojení IoT zařízení. Poskytuje efektivní a cenově výhodné připojení tím, že umožňuje zařízením přenášet malé objemy dat prostřednictvím stávajících mobilních sítí. NB-IoT se vyznačuje nízkou spotřebou energie, rozšířeným pokrytím a schopností škálovatelně připojit velké množství zařízení. Je vhodný pro aplikace, kde zařízení vyžadují dlouhou výdrž baterie a spolehlivou komunikaci ve vzdálených nebo náročných prostředích [11].

1.1.4 Radio-Frequency Identification

RFID slouží k identifikaci, sledování a správě předmětů pomocí RFID značek a čteček. Čtečka slouží k získávání dat ze značky. Značky podle zdroje napájení lze rozdělit na aktivní s vnitřním zdrojem, pasivní s napájením formou elektromagnetické indukce rádiových vln vyslaných ze čtečky a poloaktivní, které mají vlastní zdroj napájení pro vnitřní logický obvod a pro vysílání využívají energii indukovanou z rádiových vln. Typ značky má vliv na přenosovou vzdálenost a jedná se o desítky centimetrů až desítky metrů [12].

1.1.5 Wi-Fi

Wi-Fi je označení více protokolů definovaných rodinou standardů 802.11. Podporuje komunikaci na malé až střední vzdálenosti s větší datovou propustností – řádově až 100 Mbit/s. Wi-Fi není designovaná s ohledem na nízkoodběrovost, ale spíše na zpětnou kompatibilitu mezi verzemi Wi-Fi, spolehlivost a zabezpečení [13]. Obecně má Wi-Fi pokrytí do 100 metrů, ale v nezastavěném prostředí za použití směrových antén lze dosah prodloužit na stovky metrů až kilometry. Extrémním případem je spojení za pomocí technologie Long Range Wi-Fi na 50 km [14].

1.1.6 Bluetooth Low Energy

BLE (Bluetooth Low Energy) je komunikační technologie pro nízkovýkonovou komunikaci na malé vzdálenosti – do 100 m. Jedná o energeticky úspornou verzi klasického Bluetooth se zaměřením na malá zařízení s omezenými zdroji, která bývají napájena z baterie. Podporuje frekventovanou komunikaci s nízkým objemem dat. Tyto vlastnosti jsou vhodné pro IoT aplikaci v rámci nositelné elektroniky, zdravotnictví či domácí automatizace [15].

1.1.7 Zigbee

Zigbee je komunikační standard vycházející ze standardu IEEE 802.15.4[2], jenž se stará o fyzickou vrstvu a vrstvu řízení přístupu k médiu podle modelu ISO/OSI, a Zigbee se stará o vyšší vrstvy. Je určen pro sítě s důrazem na spolehlivost, nízkovýkonovými i nízkoodběrovými zařízeními, nízkými přenosovými rychlostmi, uzly v malé vzdálenosti (maximálně 100 m při přímé viditelnosti). Operuje na 27 kanálech, ovšem pouze kanály s indexy 0 (f_c = 868 MHz) a 11 až 26 (f_c = 2,4 GHz) lze použít v Evropě [16].

1.1.8 LoRaWAN

LoRaWAN je LPWAN síťový bezdrátový komunikační protokol s dlouhým dosahem a nízkou spotřebou energie určený pro bateriově napájené zařízení internetu věcí. Umožňuje zařízením přenášet malé objemy dat na vzdálenost několika kilometrů. Díky síťové architektuře LoRaWAN typu hvězda hvězd poskytuje obousměrnou komunikaci, end-to-end zabezpečení, mobilitu a lokalizační služby [17].

[2]IEEE 802.15.4 definuje fungování LR-WPAN (Low-Rate Wireless Personal Area Network)

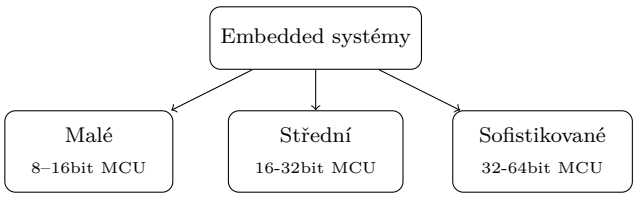
1.1.9 Sigfox

Sigfox je technologie typu LPWAN určená pro IoT. Pracuje v úzkém pásmu a používá proprietární komunikační protokol, který umožňuje připojení IoT zařízení na velké vzdálenosti s nízkou rychlostí přenosu dat. Architektura sítě je centralizovaná, což umožňuje jednoduché nasazení a správu připojených zařízení, a vyznačuje se nízkou spotřebou energie, takže je vhodná pro zařízení s omezenými energetickými zdroji [18].

2 Embedded systémy

Embedded systém je malý počítač vestavěný do většího systému sloužící k jednomu či nízkému množství účelů, který má nízké energetické a finanční nároky za cenu limitovaného množství dostupných prostředků jako je výpočetní výkon a paměť. Skládá se ze zdroje napájení, paměti, výpočetní jednotky ($\approx$ mikroprocesor) a komunikačních rozhraní, přes která probíhá výměna dat mezi mikroprocesorem a periferními zařízeními pomocí různých komunikačních protokolů. Některé systémy využívají MCU (Microcontroller Unit) neboli mikrokontroléry, kde mikroprocesor a paměť jsou součástí jednoho čipu integrovaného obvodu [19, 20].

Embedded systémy lze rozdělit podle jejich komplexnosti na malé, střední a sofistikované. Málo složitý systém obsahuje 8 či 16bitový mikrokontrolér[1], může být napájen z baterie a má nízký výpočetní výkon s minimálními nároky na paměť. Středně komplexní systémy používají větší (16 nebo 32bitové) mikrokontroléry, tedy dosahují větších výpočetních výkonů za cenu složitější konstrukce a vyšších nároků na zdroje. U sofistikovaných systémů se počítá s 32 či 64bitovými mikrokontroléry využívající více paměti, řádově jednotky megabajtů až jednotky gigabajtů, ovšem přesné číslo závisí konkrétní aplikaci. Často je nutné použít konfigurovatelných mikroprocesorů a FPGA (Field Programmable Gate Array) [20, 21].



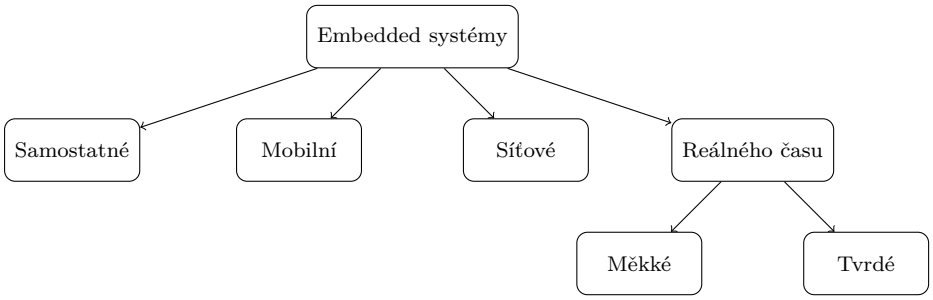
Obr. 2.1: Základní rozdělení embedded systémů podle složitosti. Přepracováno ze stránky [20]

.

Z pohledu funkcionality lze rozlišit 4 typy embedded systémů. Samostatné systémy nejsou závislé na hostitelském systému, dokáží fungovat autonomně. Dokáží přijímat/poskytovat vstupy/výstupy v digitální nebo analogové podobě. Například se jedná o meteorologickou stanici, kávovar, MP3 přehrávač. Mobilní systémy jsou podmnožinou samostatných systémů a vyžadují ještě méně zdrojů, jsou malé a snadno manipulovatelné. Do této kategorie patří například kalkulačka, digitální fotoaparát či také MP3 přehrávač. Síťové systémy jsou drátově či bezdrátově připojené do sítě, přes kterou posílají své výstupy a mohou přijímat vstupy. Může

[1] Mikroprocesor v $n$bitového mikrokontroléru disponuje registry, adresovými a datovými sběrnicemi $n$bitové velikosti, tedy pracuje s daty po $n$bitových částech.

se jednat o čistě lokální síť nebo i Internet. V souvislosti s IoT převažuje bezdrátová konektivita. Mezi síťové systémy lze zařadit bankomaty, platební terminály, IP kamery. Embedded systémy reálného času operují deterministicky, tedy poskytují výstup v určitém časovém rozsahu (termínu). Úlohy vykonávají podle časových priorit a ne metodou FIFO (First In, First Out). Tyto systémy lze dále rozdělit podle úrovně determinismu na měkké a tvrdé. U měkkých systémů nemusí být časové termíny 100 % dodrženy, i při určitých časových odchylkách je výstup validní. Příkladem měkkých systémů jsou nástroj na zaznamenávání teploty a vlhkosti nebo kardiostimulátor. Tvrdé systémy jsou časové termíny striktně dodržovány, výstup se nesmí opozdit a ani nesmí předbíhat, jinak výstup nebude přijat. Do tvrdých systémů možno kategorizovat řídící platformy v letectví či airbagy v automobilech [20, 21, 22].



Obr. 2.2: Rozdělení embedded systémů podle funkcionality. Přepracováno ze stránky [20]

.

Tab. 2.1: Přehled embedded operačních systémů ohledně typu licencování, podporovaných platforem, způsobu distribuce a možnosti vzdálené aktualizace [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33].

OS	Licence	Platformy	Distribuce	OTA
Arduino	LGPL /GPL	Závisí na použitých core knihovnách	open source	✓
Micropython	MIT	ARM, ARM Thumb, x86, x86-64, Xtensa	open source	✓
RIOT	GNU LGPLv2.1	ARM, AVR, MPS430, RISC-V x86,	open source	✓
Apache Mynewt	Apache 2.0	ARM, MIPS32, PIC32, RISC-V	open source	✓
Zephyr	Apache 2.0	ARC, ARM, II, Nios RISC-V, SPARC x86, Xtensa,	open source	✓
Embedded Linux	GPL	ARM, MIPS, RISC-V, x86	open source	✓
QNX Neutrino	proprietární	ARM, x86-64,	closed source	✓
RT-Thread	Apache 2.0	ARC, ARM, C-Sky, DSP, MIPS32, RISC-V, x86	open source	✓

2.1 Embedded operační systémy se statickým aplikačním vybavením

Pro změnu aplikačního vybavení těchto systémů je potřeba přehrát celý operační systém, jelikož aplikační vybavení je jeho součástí.

2.1.1 Assembler

Assembler neboli jazyk symbolických instrukcí je základním nástrojem pro programování na nejnižší úrovni, těsně nad strojovým kódem a s minimální jazykovou abstrakcí, a má úzkou vazbu na hardwarovou architekturou mikroprocesorů či MCU. Díky vazbě typu 1:1 mezi příkazy a instrukcemi strojového kódu nabízí assembler precizní kontrolu nad systémovými prostředky a umožňuje tvorbu vysoce optimalizovaného a efektivního kódu přizpůsobeného konkrétním hardwarovým požadavkům. Přímá manipulace s pamětí a registry umožňuje jemné ladění výkon a efektivní správu zdrojů, což má zásadní význam v pro embedded systémy s omezenými zdroji. Tato přesnost je však za cenu náročnosti vývoje a nepřenositelností, protože výsledný software je přizpůsoben zvláštnostem pouze dané procesorové architektuře.

Navzdory limitacím assembler lze použít v systémech reálného času a ve výkonově kritických aplikacích v rámci embedded systémů. Díky své schopnosti zajistit deterministické chování a splnit přísná časová omezení je nepostradatelný v oblastech, kde je nejdůležitější rychlá odezva a spolehlivost, jako jsou řídicí systémy v automobilovém průmyslu nebo průmyslová automatizace. Assembler navíc umožňuje optimalizaci kódu pro specifická omezení zdrojů, ať už jde o minimalizaci velikosti kódu nebo maximalizaci rychlosti provádění, což přináší významné zvýšení efektivity. [23, 24]

2.1.2 Barebones C/C++

Označuje softwarovou část embedded systému napsanou v programovacím jazyce C, která využívá minimum knihoven pro svoji funkčnost a má minimum závislostí. Toto řešení se používá v systémech se silně limitovanými hardwarovými možnostmi. Může využívat HAL (Hardware Abstraction Layer) pro získání nezávislosti na hardwaru či přistupovat k hardwaru na přímo přes registry, což ale dělá systém nepřenositelným, tedy je svázaný s jednou konkrétní hardwarovou konfigurací [25].

Arduino jako programovací jazyk je podmnožinou C/C++ zaměřenou na prototypování a rychlý vývoj. Oproti C/C++ postrádá standardní knihovnu (STL), zato disponuje specifickými knihovnami pro práci s mikrokontroléry na podporu I/O

operací, obsluhy pinů. Každý mikrokontrolér, jenž má u Arduina podporu, má přidělenou knihovnu Arduino core, která zprostředkovává hardwarovou abstrakci. Dále díky návaznosti na C++ částečně podporuje objektově orientované programování, konkrétně užívaní objektů a tříd [26].

2.1.3 MicroPython

MicroPython je implementace Pythonu 3 napsaná v jazyce C, která disponuje pouze základními knihovnami a je zaměřená na mikrokontroléry a embedded zařízení. Není to operační systém, ale systém umožňující bare metal programování s podporou základních funkcí operačních systémů, jako jsou správa a přepínání vláken, dynamická správa paměti a HAL. Minimální paměťová náročnost je 256 kB a vyžaduje 16 kB volatilní paměti, což je vhodné pro zařízení s omezenými zdroji ale pro ty nejextremnější případy. Zdrojový kód MicroPythonu je open source pod licencí MIT, což umožňuje volné používání, modifikace a distribuci. Tato otevřenost podporuje inovace a spolupráci v globální komunitě vývojářů, kteří neustále přispívají k vylepšování a rozšiřování funkcionality MicroPythonu [27].

2.1.4 RIOT

RIOT je operační systém pro embedded a IoT zařízení s 8, 16 či 32bitovými mikrokontroléry. Disponuje hardwarovou abstrakcí, systémovými knihovnami podporující různé komunikační protokoly, jako jsou 6LoWPAN, IPV6, RPL a CoAP, a kernelovými schopnostmi. Systém podporuje různé platformy jako ARM, AVR, MPS430, RISC-V a x86. Minimální paměťová náročnost je 38,5 kB volatilní a 10 kB nevolatilní, systém nevyžaduje zařízení s MMU (Memory Management Unit) či MPU (Memory Protection Unit). *RIOT native*, jeden z podpůrných nástrojů, umožňuje virtualizaci a emulaci IoT zařízení jako uživatelský proces v hostitelském operačním systému. Distribuce zdrojového kódu je open source pod GNU LGPLv2.1 licencí [28].

2.1.5 Apache Mynewt

Apache Mynewt je modulární RTOS určený pro IoT zařízení s 32bitovými mikrokontroléry, která musí fungovat dlouhou dobu, řádově roky, s limitací na napájení, paměť a úložiště. Jelikož je systém zaměřený na konektivitu, tak je vybaven síťovými zásobníky pro BLE5 (Bluetooth Low Energy 5), Bluetooth Mesh, Wi-Fi, LoRaWAN. Systém je víceplatformní – podpora pro ARM, MIPS32, PIC32 a RISC-V. Distribuce zdrojového kódu je open source pod Apache 2.0 licencí [29].

2.1.6 Zephyr

Zephyr je real-time operační systém (RTOS) navržený pro propojená embedded zařízení s omezenými zdroji, pracující s 32 nebo 64bitovými mikrokontroléry. Podporuje široké spektrum mikrokontrolérů, včetně ARM (Cortex-A, Cortex-R, Cortex-M), Intel x86, ARC, Nios II, Tensilica Xtensa, RISC-V, SPARC a MIPS. S malou paměťovou stopou a možností dosáhnout na 2–3 kB bez časovačů a vícevláknového zpracování, nabízí Zephyr optimální využití omezených prostředků. V plné funkčnosti se paměťové nároky pohybují v rozmezí 7–8 kB. Díky open-source distribuci pod licencí Apache 2.0 je Zephyr volně dostupný a umožňuje vývojářům přizpůsobit a rozšířit jeho funkcionality podle potřeb jejich aplikace [30].

2.2 Embedded operační systémy s dynamickým aplikačním vybavením

Pro změnu aplikačního vybavení těchto systémů není potřeba přehrát celý operační systém, jelikož aplikační vybavení je formou modulů, které lze měnit za chodu systému.

2.2.1 Embedded Linux

Embedded Linux označuje operační systémy založené na Linuxovém jádru pro nízkovýkonové a nízkoodběrové systémy. Jsou vysoce modifikovatelné podle hardwarových potřeb a mají obecně nízké paměťové nároky. Tyto systémy často využívají speciální souborové systémy, jako jsou YAFFS[2] nebo JFFS2[3], a mohou být upraveny na RTOS pro aplikace s časovými omezeními. Příkladem jsou distribuce OpenWrt, Ubuntu, Debian a distribuce vytvořené za pomoci nástrojů Yocto či Buildroot. Kvůli open-source povaze Linuxového jádra jsou systémy na něm postavené také open source [31].

2.2.2 QNX Neutrino

QNX Neutrino je operační systém reálného času (RTOS) podobný Unixu, který je postaven na mikrojádře. Mikrojádro neboli μ-kernel je minimální jádro s elementárními funkcemi, jako je správa paměti, plánování, přepínání vláken a základní

[2]YAFFS (Yet Another Flash File System) je open-source souborový systém určený pro embedded systémy s NAND nebo NOR flash pamětí.

[3]JFFS2 (Journalling Flash File System version 2) je logový souborový systém, který se přímo aplikuje na flash paměť typu NOR ale i NAND.

komunikace mezi procesy (IPC). Ovladače, souborový systém, sady síťových protokolů a další komponenty jsou umístěny v uživatelském prostoru, což umožňuje izolaci jednotlivých aplikací. QNX Neutrino je multiplatformní a podporuje architektury x86-64, ARM32 a ARM64. Systém je proprietární, což znamená, že jeho zdrojový kód není veřejně dostupný. Díky této kombinaci vlastností je QNX Neutrino robustní a široce použitelný operační systém pro vestavná zařízení v různých odvětvích a aplikacích [32].

2.2.3 RT-Thread

RT-Thread je operační systém reálného času (RTOS) pro vestavěné systémy a systémy internetu věcí napsaný v jazyce C s vysokou modularitou díky rozsáhlé sadě komponent a knihoven. Jeho flexibilita umožňuje podporu různých mikroprocesorových architektur, jako jsou ARM, RISC-V, ARC (Argonaut RISC Core), DSP (Digital Signal Processor), MIPS (Microprocessor without Interlocked Pipeline Stages) a x86. RT-Thread se dodává ve dvou verzích: v základní verzi a v upravené verzi nazvané nano. Verze nano je optimalizována pro mikrokontroléry s omezenými zdroji a vyžaduje pouze 3 kB nevolatilní paměti a 1,2 kB volatilní paměti. Distribuce zdrojových kódů pod licencí Apache 2.0 s otevřeným zdrojovým kódem zajišťuje, že RT-Thread je volně dostupný a umožňuje vývojářské komunitě upravovat a rozšiřovat jeho funkčnost podle potřeb a požadavků konkrétních aplikací [33].

3 Metriky pro porovnání operačních systémů

Pro co nejobjektivnější porovnání a zhodnocení embedded operačních systémů je potřeba stanovit vhodné metriky. V souladu se zaměřením této práce je nutno se primárně soustředit na energetickou náročnost systémů, konektivitu a hardwarové požadavky, hlavně ty paměťové, dále nelze opomenout robustnost zabezpečení, schopnost dodržet časové termíny, možnosti modifikace, podporu multithreadingu, případně paralelismu. V neposlední řadě je důležitý typ licencování – open source či close source (proprietární) – a dostupnost dokumentace. V této kapitole budou více rozvedeny následující metriky:

- Konektivita
- Nízkoodběrovost
- Hardwarová kompatibilita
- Typ licenování
- Časovost
- Kritičnost systému.

3.1 Konektivita

Konektivita je základem IoT, proto je důležité, aby systém měl k dispozici co nejefektivnější a nejspolehlivější formu spojení. Komunikační technologie mohou využívat licenčního či bezlicenčního frekvenčního spektra. Mezi výhody licenčního pásma se řadí možnost přenosu většího množství dat a zpráv denně, větší spolehlivost komunikace i lepší zabezpečení za cenu větších nákladů na licenci a komunikačního vybavení. Kdežto užití bezlicenčního pásma není zatíženo licenčními poplatky, hardwarové vybavení je přístupnější na úkor omezení střídy i výkonu vysílání s menší spolehlivostí a zabezpečením. Integritu konektivity lze posílit záložním způsobem komunikace. Dále je vhodné, aby použitá komunikační technologie měla co nejmenší energetické nároky [34].

3.2 Nízkoodběrovost

Systém je vybaven množinou různě energeticky náročných módů jako například: spánek, pasivní a standardní mód. Spánek – nejméně energeticky náročný mód, kdy systém neprovádí žádné činnosti a čeká na signál od interního časovače na probuzení. Pasivní mód, kdy systém čeká na externí signál pro započetí činnosti. Standardní mód – systém je plně funkční a může mít až nominální spotřebu. Důležité je také množství energie spojené s komunikací přes telekomunikační technologie v IoT (viz kapitola 1.1).

3.3 Hardwarová kompatibilita

Tato metrika má vystihovat podporu různých platforem, architektur výpočetních jednotek a běžně používaných senzorů jako jsou teplotní čidla Dallas či více veličinové senzory Bosch. Jaké množství volatilní a nevolatilní paměti je vyžadováno pro správnou funkcionalitu systému. Jestli OS vyžaduje přítomnost hardwarové správy paměti jako MMU, MPU či ne, dále jestli podporuje multithreading, přepínání kontextu, případně paralelismus.

3.4 Typ licencování

Jakým způsobem je systém distribuován, jak je finančně nákladný, jakou má podporu a jak lze rozšiřovat či modifikovat jeho funkcionalitu. Proprietární systémy bývají poskytovány „tak jak jsou“, tedy bez možnosti modifikace kódu. Systém může být formou SaaS Software as a Service s danou délkou podpory, kdy je měsíčně účtováno každé zařízení využívající daný systém [35]. Na druhou stranu open-source systémy bývají distribuovány s plnou volností modifikace a bez finanční zátěže, ovšem podporu mohou mít omezenou nebo zprostředkovanou komunitou vývojářů, která zároveň stojí za vývojem daného systému [30].

3.5 Časovost

Podle schopnosti dodržet časové termíny lze systémy rozdělit na univerzální, měkké a tvrdé. Univerzální buď ani nedokáží přepínat mezi procesy, dokáží zpracovávat jenom jeden proces, nebo nemají časové priority. Měkké a tvrdé přepínají kontext mezi procesy na základě priorit. Měkké se snaží o dodržení časových termínů, ale ne za každou cenu. Tvrdé udělují každému procesu konkrétní čas na vykonání své činnosti, nemůže se stát, že by nějaký proces blokoval ostatní.

3.6 Kritičnost systému

Úroveň kritičnosti systému odpovídá závažnosti nežádoucích následků v případě totálního selhání systému. Existují 4 typy kritických systémů: byznysově, bezpečnostně, úkolově a životně kritické. Chyba byznysově kritického systému může vést k hmatatelné či nehmatatelné finanční ztrátě. U bezpečnostně kritických systémů může dojít k úniku citlivých dat či jejich ztrátě. Nesprávná funkcionalita úkolově kritických systémů může způsobit neschopnost poskytovat požadované výstupy. Pochybení životně kritických systémů může v důsledku způsobit vážné poranění osob až ztrátu na životě či může poškodit životní prostředí [36].

4 Výběr platforem a operačních systémů

4.1 Výběr platformy

Výběr cílové platformy vycházel z dostupnosti, existence dev kitů, podpory senzorů, možnosti konektivity a napájení platforem. Potenciální výběr 7 platforem je vyobrazen v tabulce 4.1. Pro lepší porovnání operačních systémů je třeba, aby všechny platformy byly kompatibilní se všemi testovanými operačními systémy. Finální výběr je v tabulce znázorněn šedě.

Tab. 4.1: Přehled kompatibility hardwarových platforem a kandidátů z embedded operačních systémů [26, 27, 28, 30, 33, 37, 38].

`OS\Platforma`	Chester	ESP32	F. M0	HiFive1	Pi 4	Pico	STM32	pyboard
RIOT	–	✓	✓	✓	–	✓	✓	✓
Micropython	–	✓	✓	–	✓	✓	✓	✓
Zephyr	✓	✓	✓	✓	✓	✓	✓	–
RT-Thread	–	✓	–	✓	✓	✓	✓	–
Win IoT	–	–	–	–	✓	–	–	–
OpenWrt	–	–	–	–	✓	–	–	–
Arduino	–	✓	✓	✓	- ✓	✓	✓	–

Vysvětlení zkratek platforem v tabulce 4.1

- Chester označuje Hardwario Chester, bližší informace na [39]
- ESP označuje ESP32-C3-DevKitM1, bližší informace na [40]
- F. M0 označuje Adafruit Feather M0, bližší informace na [41]
- HiFive1 označuje HiFive1 Rev B, bližší informace na [42]
- Pi 4 označuje Raspberry Pi Model 4 (Cortex-A72), bližší informace na [43]
- Pico označuje Raspberry Pi Pico, bližší informace na [44]
- STM označuje STM32 Nucleo-L152RE, bližší informace na [45]
- pyboard označuje PYBv1.1, bližší informace na [46]

4.2 Hardwarové platformy

4.2.1 ESP32-C3-DevKitM-1

ESP32-C3-DevKitM-1 je univerzální vývojová deska navržená společností Espressif Systems založená na modulu ESP32-C3-MINI-1 s SoC ESP32-C3FN4. Čip ESP32-C3FN4 umožňuje programátorům pracovat s 4 MB flash pamětí, 384 kB ROM pamětí, 400 kB SRAM pamětí a 8 kB SRAM pamětí v nízkoodběrovém režimu.

Modul ESP32-C3-MINI-1 disponuje PCB anténou umožňující rádiovou komunikaci na v bezlicenčním pásmu ISM, konkrétně pomocí standardů Wi-Fi 802.11 b/g/n a BLE5. Deska odhaluje celkově 15 GPIO pinů a podporuje několik periferních rozhraní, jako jsou SPI, I2C, UART, PWM, ADC a DAC, což umožňuje snadno připojit širokou škálu senzorů a dalších periférií. Přítomnost vestavěného převodníku UART-USB spolu s kompatibilitou se standardními vývojovými nástroji jako ESP-IDF a Arduino IDE (Integrated Development Enviroment) zvyšuje její vhodnost pro vývoj a prototypování.

ESP32-C3-DevKitM-1 navržena pro nízkou spotřebu energie a tedy ideální pro aplikace s bateriovým napájením. Obsahuje funkce, jako jsou nízkoodběrové režimy spánku a dynamické škálování vysílacího výkonu, které optimalizují spotřebu energie a prodlužují životnost baterie. Procesor, 32bitové jednojádro RISC-V pracující na frekvenci až 160 MHz, poskytuje vyváženou kombinaci výkonu a energetické účinnosti, vhodnou pro většinu aplikací reálného času.

Vývojová deska obsahuje také základní bezpečnostní funkce, jako je bezpečné spouštění a šifrování flash paměti, které jsou důležité pro ochranu integrity a důvěrnosti softwaru, zejména v kritických aplikacích. SDK (Software Development Kit) pro ESP32-C3 je otevřená, což podporuje transparentnost a flexibilitu při vývoji. Přestože je spolehlivý pro spotřebitelské a průmyslové použití v IoT, není určen pro nejnáročnější systémy s vysokou kritičností, jako jsou lékařské nebo letecké aplikace. ESP32-C3-DevKitM-1 nabízí komplexní platformu pro vývoj široké škály připojených a vestavěných aplikací se středními – bezpečnostními požadavky na kritičnost [40].

4.2.2 Raspberry Pi Pico

Raspberry Pi Pico je deska s mikrokontrolérem založená na čipu RP2040 navrženém společností Raspberry Pi. Je vybavena dvoujádrovým procesorem Arm Cortex-M0+ s frekvencí 133 MHz, 264 kB paměti SRAM a 2 MB vestavěné paměti flash. Pico je vybaven širokou škálou vstupů a výstupů, včetně 26 GPIO pinů, I2C, SPI, UART, PWM a ADC. To z něj činí univerzální a cenově dostupnou platformu nejen pro IoT projekty ale i pro výuku a demonstraci embedded systémů. Ovšem nízká cenová dostupnost je odůvodněna absencí vestavěného UART-USB převodníku.

Jednou z hlavních předností desky Raspberry Pi Pico je její flexibilita a snadné programování. Podporuje vývoj v jazyce C/C++ pomocí oficiální sady Pico SDK a nabízí také podporu jazyka MicroPython, takže je přístupný jak začátečníkům, tak pokročilým uživatelům. Pico lze programovat a ladit prostřednictvím jednoduchého připojení USB, které desku také napájí. Díky svým kompaktním rozměrům a nízké spotřebě energie je navíc vhodná pro embedded systémy, přenosná zařízení a aplikace

napájené z baterií.

Mimo IoT lze Raspberry Pi Pico použít v nesčetných dalších projektech, jako je domácí automatizace, robotika a vlastní periferie. Jeho schopnost zvládat operace v reálném čase a propojení s různými senzory a akčními členy otevírá možnosti v oblastech, jako je monitorování životního prostředí, vzdělávací nástroje a experimentální sestavy. Aktivní komunita a rozsáhlá dokumentace dále zvyšují jeho přitažlivost a poskytují množství zdrojů [44].

4.2.3 STM32 Nucleo-L152RE

STM32 Nucleo-L152RE slouží jako univerzální vývojová platforma, která využívá výkon mikrokontroléru STM32L152RE z rodiny STM32 Nucleo-64 společnosti STMicroelectronics. Díky jádru 32bitovému ARM Cortex-M3 s rychlostí až 32 MHz dosahuje rovnováhy mezi výkonem a energetickou účinností a vyhovuje různým embedded aplikacím. 512 kB flash paměti a 80 kB paměti SRAM poskytují dostatek prostředků pro vývoj složitějších aplikací. Jeho bohatý sortiment periferií, včetně ADC, D/A převodníků, časovačů a komunikačních rozhraní, jako jsou I2C, SPI, USART a USB, navíc usnadňuje integraci s různými senzory a moduly.

Charakteristickým prvkem desek z rodiny STM32 Nucleo-64 je kompatibilita s hlavicemi Arduino Uno V3 umožňující rozšíření pomocí Arduino štítů a kompatibilita s hlavicemi ST Morpho, které umožňují přístup ke všem GPIO pinům a pokročilým funkcím mikrokontroléru. Tato duální kompatibilita poskytuje vysokou flexibilitu při vývoji.

STM32 Nucleo-L152RE se bezproblémově integruje s ekosystémem STM32Cube, včetně nástrojů, jako je STM32CubeMX pro konfiguraci a generování kódu, a softwarového balíčku STM32CubeL1, který poskytuje komplexní knihovnu HAL a komponenty middlewaru. Je kompatibilní s mikrokontrolérovými IDE, jako jsou Keil MDK, IAR EWARM a prostředími založená na GCC. Deska obsahuje integrovaný obvodový debugger a programátor ST-LINK/V2-1, který eliminuje potřebu externích ladicích nástrojů a zjednodušuje vývoj. Díky této kombinaci hardwarových funkcí a robustní softwarové podpory je deska STM32 Nucleo-L152RE vynikající volbou pro rychlé vytváření prototypů a efektivní vývoj projektů v široké škále embedded systémů [45].

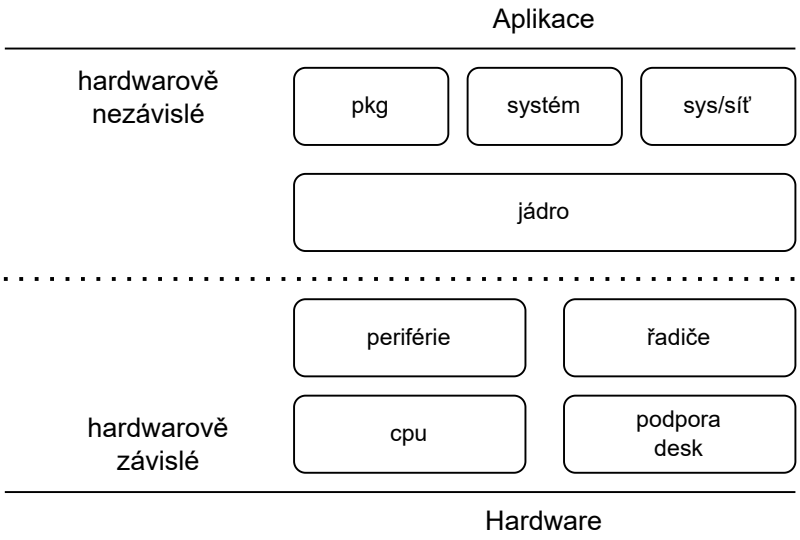
4.3 Výběr operačních systémů

V kapitole 2 bylo představeno několik embedded operačních systémů s různými požadavky, jako jsou minimální nároky na paměti, hardwarová správa paměti či přítomnost RTT časovače. Dané požadavky nemusí splňovat každá cílová platforma, a

vlastnostmi vhodnými jen pro určité IoT aplikace, proto je potřeba provést podrobnější analýzu s ohledem na metriky představené v předchozí kapitole. Win 10 IoT Core (v tabulce 4.1 uveden jako Win IoT) má ve srovnání s ostatními operačními systémy velké paměťové nároky (2 GB nevolatilní a 256 MB volatilní paměti), které z uvedených platforem splňuje pouze Raspberry Pi 4. Stejná situace postihuje i OpenWrt, byť oproti Win 10 IoT Core má 8krát menší paměťové nároky [43, 37, 38].

4.3.1 RIOT

RIOT je open source modulární operační systém založený na minimalistickém jádře. Je cílen na embedded systémy s výrazně limitovanými zdroji. Zaměřuje se na minimalizování použitých zdrojů pamětí (volatilní RAM a nevolatilní ROM), minimalizování spotřeby energie, poskytnutí softwarové platformy se snadným kódováním a schopnosti reálné časovosti. Použitím vrstvy hardwarové abstrakce („hardwarově závislá“ sekce na obrázku 4.1) umožňuje podporu rozsáhlých konfigurací embedded platforem, snížení duplikace kódu mezi platformami a přenositelnost kódu [47].



Obr. 4.1: Znázornění struktury operačního systému RIOT. Přepracováno z práce [47]

.

4.3.2 Micropython

MicroPython je odlehčená open-source implementace programovacího jazyka Python 3 optimalizovaná pro mikrokontroléry. Jeho hlavní předností je vhodnost pro aplikace s nízkou spotřebou energie, takže se dobře hodí pro zařízení s omezenými zdroji. MicroPython se zaměřuje na možnosti práce v reálném čase a umožňuje

rychlý vývoj a spouštění kódu na mikrokontrolérech, což z něj činí univerzální volbu pro embedded systémy. Jeho hardwarová kompatibilita se vztahuje na různé platformy mikrokontrolérů, což podporuje široké možnosti integrace. MicroPython podporuje funkce konektivity, které usnadňují komunikaci s jinými zařízeními prostřednictvím technologií jako LoRa, NB-IoT, LTE Cat M, Wi-Fi, BLE či Zigbee. Je uvolněn pod licencí open-source MIT licencí, což podporuje společný vývoj a široké přijetí. Vzhledem k jeho schopnosti pracovat v prostředí s omezenými zdroji, podpoře reálné časovosti, hardwarové kompatibilitě, funkcím konektivity a open-source povaze je MicroPython klíčový pro vytváření efektivních a pohotových embedded systémů [48].

4.3.3 Zephyr

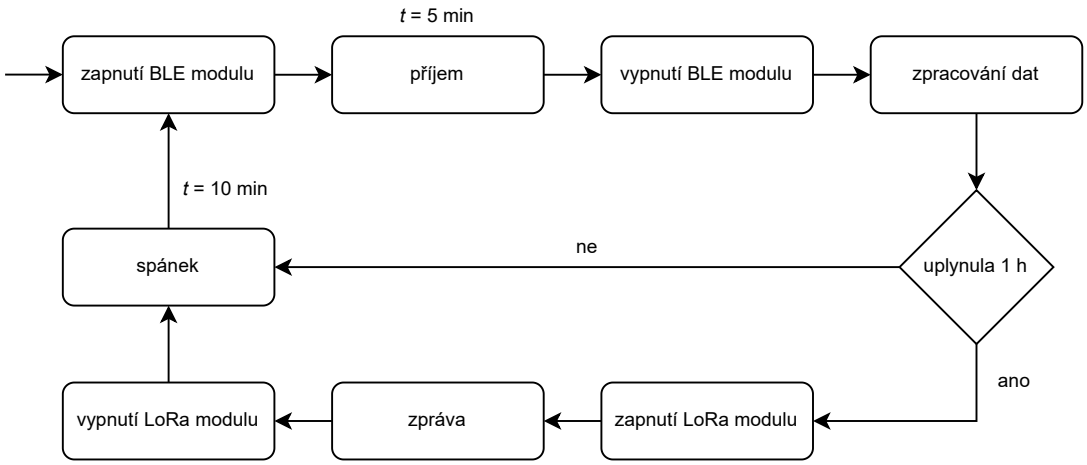
Zephyr je open-source RTOS určený pro embedded systémy s omezenými zdroji, který klade důraz na malý odběr energie a efektivní výkon. Je přizpůsoben pro širokou škálu hardwarových platforem a zajišťuje kompatibilitu s různými mikrokontroléry a procesory. Zephyr vyniká tím, že poskytuje prostředí reálného času, které umožňuje přesné časování a odezvu, jež jsou klíčové pro aplikace, jako jsou IoT zařízení a nositelná zařízení. Pro konektivitu podporuje komunikační protokoly, jako jsou LoRa, NB-IoT, LTE Cat M, Bluetooth, Wi-Fi a Thread, takže je vhodný pro různé síťové scénáře. Zephyr je distribuován s permisivní open-source licencí, která umožňuje flexibilitu při vývoji a komerčním využití. Kritičnost systému podtrhuje jeho schopnost zvládat úlohy s přísnými požadavky na časování, což z něj činí spolehlivou volbu pro aplikace, kde je nejdůležitější rychlá odezva a nízká spotřeba energie, například v průmyslové automatizaci, zdravotnictví a chytrých zařízeních [49].

5 Plán testování a modelové scénáře

Pro co nejobjektivnější otestovávaní je potřeba stanovit jednotný plán testování pro vybrané operační systémy. Testované kombinace (platforma + operační systém) budou disponovat totožným senzorickým a komunikačním vybavením i aplikačním vybavením, bude-li to možné. Energetické nároky testovaných konfigurací budou vypočítány změřením odebíraného proudu při konstantním napětí dodávaného laboratorním zdrojem. Limitace tohoto měření, je to, že oproti baterce dodávané napětí nekolísá a nemění se v závislosti na teplotě baterky a stavu napětí.

5.1 Modelový scénář

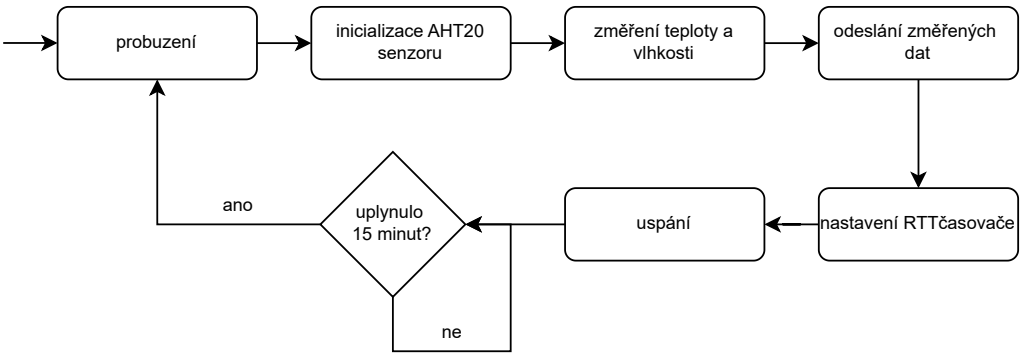
Soustava složená ze zařízení sloužící jako koncentrátor a senzorické sítě po 6 měřicích stanic vybavenými senzory na měření teploty a vlhkosti. Stanice každých 15 minut posílají data o teplotě a vlhkosti pomocí BLE5 koncentrátoru, ten je shromažďuje a následně každou hodinu je pošle na server pomocí LoRaWAN. Koncentrátor opakuje 15 minutový cyklus zakončený 10 minutami v módu hluboký spánek. Koncentrátor pracuje s naměřenými daty jako s 7 bitovými hodnotami, jelikož na vyjádření teploty stačí hodnoty od -63 do 64 a pro vlhkost 0 až 127, tedy sloučením naměřených dat za 1 hodinu se ušetří 1 bajt na měřící stanici.



Obr. 5.1: Blokový diagram chování koncentrátoru pro modelový scénář

5.1.1 Implementace

Modelový scénář pojednává o soustavě celkem 7 zařízení, ale implementace se věnuje pouze jedné měřící stanici. Ze senzorických periférií je stanice vybavena senzorem



Obr. 5.2: Blokový diagram chování měřící stanice pro modelový scénář

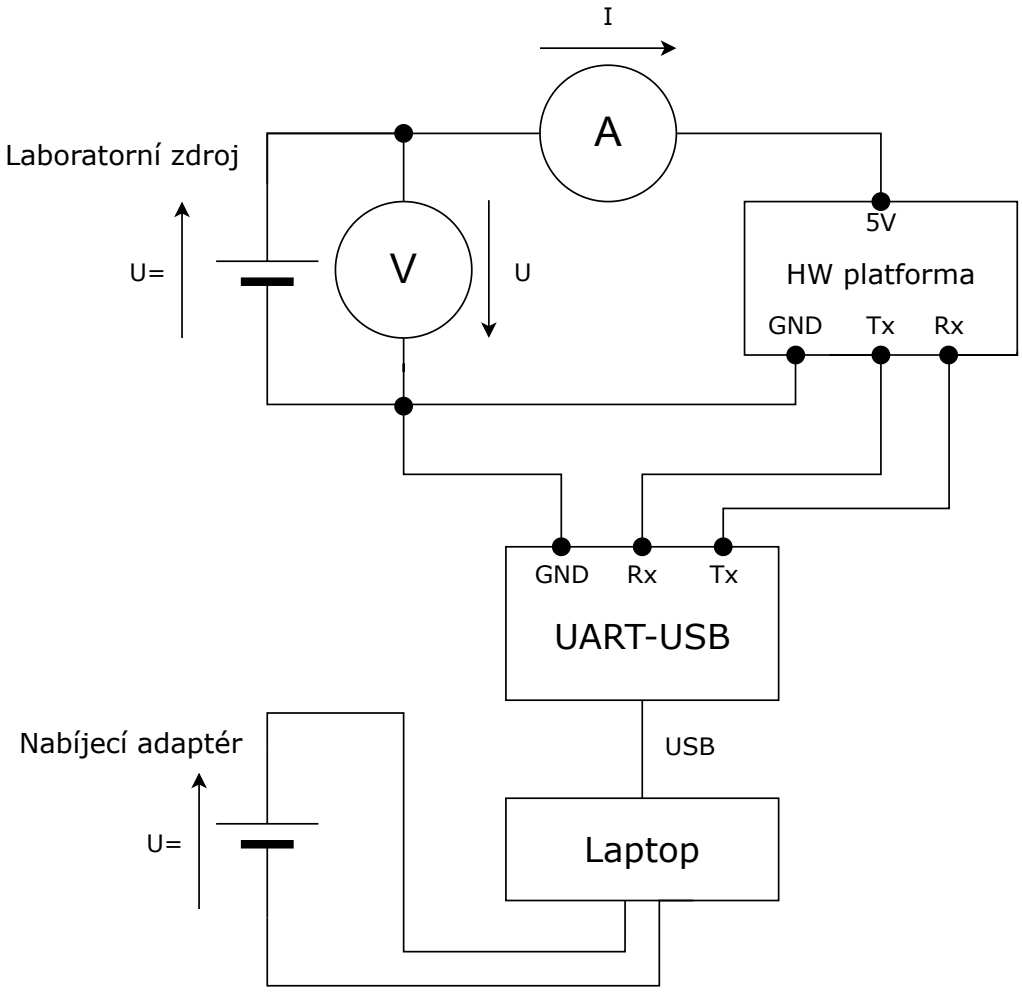
teploty a vlhkosti AHT20 a z komunikačních periférií má UART modul, jenž simuluje BLE5 modul.

5.1.2 Měření

Z pohledu nízkoodběrovosti se v rámci tohoto modelového scénáře měří odběr proudu v různých fázích aktivity zařízení. Za prvé při běžném provozu, kdy zařízení provádí výpočetní operace, ukládá a načítá data do/z paměti, komunikuje s periferiemi. Za druhé při komunikaci, kdy vysílá naměřená data. A za třetí zařízení se nachází v nízkoodběrovém režimu, kdy odstavuje napájení periférií v módu light sleep. V módu deep sleep přestane napájet procesor a paměti. Pro zachování naměřených dat je potřeba je uložit na nevolatilní paměť, napájení je pro změnu spuštěno pro RTT časovač, který má na starosti probuzení systému.

6 Měření

Měření byla prováděna podle zapojení 6.1. Jako voltmetr a ampérmetr posloužila vícekanálová výkonová sonda od Rohde & Schwarz RT-ZVC02A, která byla datovým kabelem připojena k počítači, jenž zpracovával data naměřená sondou o vzorkovací frekvenci $f_{\text{vz}} = 5\,\text{kSa/s}$. Měřené hardwarové platformy byly napájeny laboratorním zdrojem o výstupním napětí $U = 5\,\text{V}$ a limitu na odebíraný proud $I = 200\,\text{mA}$.



Obr. 6.1: Schéma zapojení

V rámci měření vznikali odchylky způsobené propojení vodičů skrze nepájivé pole a hlavně propojením obvodu s elektrickou sítí přes UART-USB převodník připojený k laptopu, jenž byl v době měření napájen ze sítě. Vliv střídavého napětí o frekvenci $f = 50\,\text{Hz}$ lze pozorovat na většině detailních grafů (například A.6, A.12, či A.18).

U každé HW platformy byla snaha o implementaci modelového scénáře měřící stanice vybavené kombinovaným senzorem na měření teploty a vlhkosti AHT20. Komunikace se senzorem probíhala po sběrnici I2C. Měřeny byly následující veličiny:

- $\overline{I}_R$ – průměrný odběr proudu při provozu bez nízkoodběrového módu,
- $\overline{I}_{\mathrm{ds\,+\,a}}$ – průměrný odběr proudu v nízkoodběrovém módu deep sleep s aktivitou každých 20 sekund,
- $\overline{I}_{\mathrm{ls\,+\,a}}$ – průměrný odběr proudu během cyklu v nízkoodběrovém módu light sleep s aktivitou každých 20 sekund,
- $\overline{I}_{\mathrm{ds}}$ – průměrný odběr proudu v nízkoodběrovém módu deep sleep,
- $\overline{I}_{\mathrm{ds}}$ – průměrný odběr proudu v nízkoodběrovém módu light sleep,
- $\overline{t}_{\mathrm{ds}}$ – průměrná doba potřebná na přechod z nízkoodběrového módu deep sleep do aktivního módu,
- $\overline{t}_{\mathrm{ls}}$ – průměrná doba potřebná na přechod z nízkoodběrového módu light sleep do aktivního módu.

Měření každé veličiny probíhalo 60 s při vzorkovací frekvenci $f_{\mathrm{vz}} = 5\,\mathrm{kSa/s}$. Bylo tedy získáno přesně 300000 hodnot využitých pro výpočet aritmetického průměru výstupních veličin. V rámci měření byly zaznamenány 3 pracovní cykly ukončené nízkoodběrovým módem nebo prostým pozastavením procesu vlákna.

Následující podkapitoly jsou věnovány porovnání vybraných operačních systémů na jednotlivým hardwarovým platformám .Grafické výstupy pro každou kombinaci platformy a operačního systému se skládají ze dvou grafů. První graf znázorňuje průběh celého měření a zachytává většinou 3 pracovní cykly. Druhý graf je zaměřený na první přechod z nízkoodběrového módu do aktivního a opětovného uspání, tedy obnovení napájení periférií a případného obnovení paměťového kontextu v případě módu deep sleep.

6.1 ESP32-C3-DevKitM-1

V rámci implementace modelového scénáře a nízkoodběrových módů byla tato platforma takřka bezchybná. Problémový byl pouze Mikropython, u kterého mód deep sleep způsoboval pád systému a následně bylo třeba znovu nahrát kompilátor Mikropythonu.

Na grafu a) obrázku A.1 pro kombinaci ESP/Arduino bez nízkoodběrového módu lze vidět vidět oscilující hodnotu odebíraného proudu od 18 mA po 19,5 mA, což odpovídá průměrné hodnotě $\overline{I}_R$ v tabulce 6.1. Maximální odběr proudu při činnosti byl 29,5 mA, jak je lépe vidět na spodním grafu b). Zároveň jsou spodním grafu hezky vidět proudové špičky, které vznikali v důsledku komunikace po sběrnici I2C, a následná velká špička, jenž vznikla v důsledku poslání dat přes UART.

Na grafu a) obrázku A.2 pro kombinaci ESP/Arduino s módem light sleep lze vidět průměrný odebíraný proud $1,1\,mA$ a maxima v činnosti na 30 mA. A z grafu b) lze vyčíst takřka instantní probuzení z módu light sleep za méně než 1 ms.

Na grafu a) obrázku A.3 pro kombinaci ESP/Arduino s módem deep sleep lze spatřit průměrný odebíraný proud $0,9\,mA$ se špičkami až 35 mA po probuzení. Graf b) v první polovině časové osy zobrazuje akt probuzení, kde se na základě akce RTT (Real Time Timer) časovače během 40 ms obnoví napájení volatilní paměti s procesorem a během dalších 40 ms obnoví napájení i periférií.

Na grafu a) obrázku A.4 pro kombinaci ESP/Mikropython bez nízkoodběrového módu si lze povšimnout neznámých odchylek kolem 18. a 20. sekundy, kdy odebíraný proud nejdříve klesl na 11 mA a následně vystoupal k 27 mA. Běžně hodnota odebíraného proudu osciluje kolem 19 mA a v při inicializaci komunikace přes sběrnice I2C a UART vystoupá odebíraný proud až na 35 mA. Tato hodnota nejvyšší ze všech platforem.

Tab. 6.1: Přehled průměrných proudových odběrů pro hardwarovou platformu ESP32-C3-DevKitM-1

OS\Měření	$\overline{I}_R$ [mA]	$\overline{I}_{\mathrm{ds+a}}$ [mA]	$\overline{I}_{\mathrm{ls+a}}$ [mA]	$\overline{I}_{\mathrm{ds}}$ [mA]	$\overline{I}_{\mathrm{ls}}$ [mA]	$\overline{t}_{\mathrm{ds}}$ [ms]	$\overline{t}_{\mathrm{ls}}$ [ms]
Arduino	18,885	1,006	1,147	0,9	1,1	70	0,8
Micropython	18,430	-	1,208	-	0,9	-	44,8
RIOT	17,144	3,380	2,712	2,175	2,42	14	1,6
Zephyr	18,667	2,408	1,184	0,905	1,17	212,8	1

V nízkoodběrových módů dosáhl nejlepších výsledků systém Arduino. Systém Zephyr měl podobné výsledky v rámci odběru proudu, ale oproti Arduinu potřeboval

trojnásobnou dobu na přechod módu deep sleep do aktivního módu. Systém RIOT sice měl největší odběr proudu, ale zase se probouzel nejrychleji z módu deep sleep a dále měl nejmenší odběr v případě referenčního měření bez nízkoodběrového módu.

6.2 Raspberry Pi Pico

Měření na platformě Raspberry Pi Pico bylo vcelku limitované, jelikož tato hardwarová platforma, byť má širokou softwarovou podporu, nemá ji až tak hlubokou. RIOT nemá implementované všechny funkce platformy Pico, přesněji řečeno funkcionalitu sběrnice I2C, která je vitální pro modelový scénář kvůli odečítání dat ze senzoru AHT20. Arduino, lépe řečeno základní knihovna zprostředkovávající HAL, neposkytuje kontrolu nad nízkoodběrovými režimy, kterými Pico disponuje a v rámci Barebones C programů jsou dostupné. A pro systém Zephyr byla použita aplikace s automatickým spánkem, kde systém se měl implicitně přepínat do nízkoodběrového módu v čase neaktivity, ale na základě výsledků měření je jasné, že tato implementace nebyla úspěšná.

Tab. 6.2: Přehled průměrných proudových odběrů pro hardwarovou platformu Raspberry Pi Pico

OS\Měření	$\overline{I}_R$ [mA]	$\overline{I}_{\mathrm{ds+a}}$ [mA]	$\overline{I}_{\mathrm{ls+a}}$ [mA]	$\overline{I}_{\mathrm{ds}}$ [mA]	$\overline{I}_{\mathrm{ls}}$ [mA]	$\overline{t}_{\mathrm{ds}}$ [ms]	$\overline{t}_{\mathrm{ls}}$ [ms]
Arduino	19.008	-	-	-	-	-	-
Micropython	18,556	1,461	1,318	1,3	1,5	155,2	2,3
RIOT	-	-	-	-	-	-	-
Zephyr	18,901	-	18,944	-	18,8	-	10

Z grafu a) obrázku A.13 lze vyčíst odebíraný proud osciloval kolem hodnoty 18,5 mA a při vykonávání úkonu byl maximální proud až 25 mA. Na grafu b) lze vidět průběh komunikace se senzorem přes sběrnici I2C. Proudové špičky, jenž dosáhli hodnoty kolem 24 mA a trvali cca 2 ms, byly způsobeny zasláním příkazu k měření. Následné špičky odpovídají čekání na odpověď s naměřenými hodnotami od senzoru. Poslední špička je způsobena komunikací přes UART.

Na grafu a) obrázku A.14 lze vidět odebíraný proud během módu light sleep oscilující kolem hodnoty 1,5 mA. Na grafu b) lze vidět navýšení odběru proudu způsobené obnovení napájení periférií při změně módu 26 mA, která trvala 2,3 ms. Po přechodu do aktivního módu je průběh totožný jako v referenčním grafu A.13 pro platformu Pico s Mikropythonem.

Graf a) obrázku A.15 vizualizuje průběh odebíraného proudu při střídání nízkoodběrového módu deep sleep a aktivního módu. Proud osciluje kolem hodnoty 1,3 mA a v aktivním módu vystoupá až na hodnotu 27 mA, jenž je způsobeno opětovným napájením periférií. Na grafu b) lze vidět probuzení z módu deep sleep je ve dvou fázích. V první fázi trvající 52 ms se obnovilo napájení pamětí a procesoru

a v druhé fázi trvající 100 ms bylo obnoveno napájení periférií. Proudové špičky v aktivním módu jsou stejné jako na grafu b) obrázku A.13 pro referenční měření kombinace Pico/Mikropython.

6.3 STM32 Nucleo-L152RE

Na platformě STM pro systém Zephyr byl podobně jako u platformy Pico použit mód automatického spánku, ovšem ani tady nebyla implementace úspěšná. Implementace podle modelového scénáře zbylých systémů proběhly v pořádku.

Tab. 6.3: Přehled průměrných proudových odběrů pro hardwarovou platformu STM32 Nucleo-L152RE

OS\Měření	$\overline{I}_R$ [mA]	$\overline{I}_{\text{ds+a}}$ [mA]	$\overline{I}_{\text{ls+a}}$ [mA]	$\overline{I}_{\text{ds}}$ [mA]	$\overline{I}_{\text{ls}}$ [mA]	$\overline{t}_{\text{ds}}$ [ms]	$\overline{t}_{\text{ls}}$ [ms]
Arduino	55,148	43,903	44,384	44	44,5	0,8	0,8
Micropython	47,759	43,613	43,821	43,5	44	11	6,6
RIOT	48,808	43,875	43,588	43,7	43,8	1,8	0,8
Zephyr	48,432	-	47,189	-	47,3	-	5,8

Měření vývojové desky STM32 Nucleo-L152RE bylo problematické, jelikož deska disponuje červenou LED diodu typu PWM (Pulse Wave Modulation) signalizující externí napájení blikáním o frekvenci $f = 1$ Hz a tedy zvyšovala odběr proudu o nějakých 20 mA. Průměrný odběr proudu při referenčním měření je vyšší až o 7 mA. Nejnižších odebíraných proudů dosáhl Mikropython. V rámci nízkoodběrových módů byly RIOT a Arduino výsledky velice blízko Mikropythonu, dokonce měli značně rychlejší probuzení. Pouze u Zephyru s nešťastnou implementací automatického spánku lze pozorovat nejmenší snížení odebíraného proudu v nízkoodběrovém módu light sleep.

Na grafu a) obrázku A.18 lze vidět jak PWM LED dioda způsobuje kolísající oscilaci na odebíraném proudu. Vliv diody je tak velký, že činnost v aktivním módu jen nepatrně zvyšuje odebíraný proud $\Delta I = 1$ mA, což je stejná hodnota přesahů diody.

Na grafu a) obrázku A.19 lze vidět, že dvakrát nastala časová shoda okna aktivity a vysokého pulzu PWM LED diody, tedy odebíraný proud vystoupal až na 63 mA, což je navýšení $\Delta I = 11$ mA oproti módu light sleep. Na grafu b) jsou pozorovatelné proudové špičky. Ty, které přesáhly hranici 62 mA, jsou způsobeny výčtem veličin ze senzoru a poslední špička, před přechodem do nízkoodběrového módu, je způsobena vysíláním dat přes UART.

Na grafu a) obrázku A.20 lze vidět, že dvakrát nastala časová shoda okna aktivity s vysokým pulzem PWM LED diody a odebíraný proud vystoupal až na 61,7 mA, tedy zvětšil o 12 mA oproti módu deep sleep.

Závěr

Tato bakalářská práce se věnovala embedded operačním systémům s ohledem na jejich efektivitu a nízkoodběrovost. Hodnocení vychází z metrik stanovených ve třetí kapitole. A jsou jimi konektivita, nízkoodběrovost, hardwarová kompatibilita, typ licencování a kritičnost.

Z pohledu kritičnosti Mikropython zaostává za ostatními už jenom kvůli své dynamické povaze, kdy kompiluje kód za běhu, což snižuje šanci na včasné odhalení chyb jako je třeba na nic neukazující pointer (NULL pointer). Ostatní systémy jsou předkompilovány. Arduino používá AVR g++, RIOT a Zephyr nepoužívají samotný kompilátor ale celý řetězec nástrojů, z kterého jsou používány pouze potřebné nástroje potřebné pro konkrétní platformu.

Ve věci časovosti jsou RIOT a Zephyr vyrovnaní, jelikož se v obou případech jedná o operační systémy reálného času s podporou plánovače. Ovšem jejich zaměření na hardwarově limitovaná zařízení znamená, že nemůžou být časově tvrdé. Arduino podporuje funkce RTOS formou komunitní knihovny HeliOS. Mikropython nemá podporu pro dodržení časování ani oficiální ani komunitní.

Hardwarová kompatibilita značí rozsah podporovaných architektur MCU a rozsáhlost hardwarové abstrakční vrstvy. Mikropython podporuje desítky desek, Arduino přes 100, RIOT podporuje více než 200 desek a Zephyr má podporu pro stovky desek. Ovšem hardwarová podpora nemusí hotová nebo zpřístupňovat všechny funkce desky. Konkretně Arduino dokonce má i více core knihoven pro Raspberry Pi Pico, ale ani jedna nepodporuje nízkoodběrové módy. Dále RIOT nemá implementované ovladače pro periférie platformy Raspberry Pi Pico mimo UART.

Všechny čtyři operační systémy jsou open source a mají kolem sebe velkou komunitu. Arduino neoznačuje pouze operační systém/programovací jazyk, ale také hardware, který je taktéž open source, díky čemuž existuje rozsáhlý ekosystém Arduino doplňků.

V rámci nízkoodběrovosti dosáhlo nejnižších odběrů proudu Arduino, ovšem Mikropython si také nevedl špatně. Oba systém měli v módu deep sleep proudový odběr kolem $I = 1$ mA.

RIOT a Zephyr jsou operační systémy zaměřené na IoT, takže v základu jsou vybaveny rozsáhlými komunikačními zásobníky, kdežto Arduino a Mikropython se spoléhají na externí knihovny.

Literatura

[1] HARPER, Douglas. *Origin and meaning of tele–* [online]. 2024-01-31. [cit. 2024-05-20]. Dostupné z: https://www.etymonline.com/word/tele-.

[2] FOOTE, Keith D. *A Brief History of the Internet of Things* [online]. 2022-01-14. [cit. 2023-11-15]. Dostupné z: https://www.dataversity.net/brief-history-internet-things.

[3] *The "Only" Coke Machine on the Internet* [online]. 2005. [cit. 2023-11-15]. Dostupné z: https://www.cs.cmu.edu/~coke/history_long.txt.

[4] ASHTON, Kevin. *That 'Internet of Things' Thing* [online]. 2009. [cit. 2023-11-15]. Dostupné z: https://www.rfidjournal.com/that-internet-of-things-thing.

[5] CALLEBAUT, Gilles; LEENDERS, Guus; VAN MULDERS, Jarne; OTTOY, Geoffrey; DE STRYCKER, Lieven; PERRE, Liesbet Van der. The Art of Designing Remote IoT Devices—Technologies and Strategies for a Long Battery Life. *Sensors*. 2021, roč. 21, č. 3. ISSN 1424-8220. Dostupné z DOI: 10.3390/s21030913.

[6] 3GPP. *Cellular system support for ultra low complexity and low throughput Internet of Things; (release 13), version 13.1.0*. 2015. Tech. zpr., Rep. 45.820. 3GPP.

[7] *Internet of Things: The Five Types of IoT* [online]. 2022. [cit. 2023-11-15]. Dostupné z: https://syntegra.net/internet-of-things-the-five-types-of-iot.

[8] *Co to je IoT?* [online]. 2020. [cit. 2023-11-15]. Dostupné z: https://www.iotport.cz/iot-novinky/ostatni-clanky-o-iot/co-to-je-iot.

[9] *How IoT Devices Connnect To The Internet* [online]. 2023. [cit. 2023-11-15]. Dostupné z: https://robots.net/tech/how-iot-devices-connect-to-the-internet/.

[10] POLÁK, Ladislav. *Mobilní komunikace*. Brno: UREL FEKT VUT v Brně, 2023. Přednášky.

[11] 3GPP. *Release 14 Description; Summary of Rel-14 Work Items; (release 14), version 14.0.0*. 2018. Tech. zpr., Rep. 21.914. 3GPP.

[12] *Understanding RFID and RFID Operating Ranges* [online]. 2017. [cit. 2023-11-15]. Dostupné z: https://blog.acdist.com/understanding-rfid-and-rfid-operating-ranges.

[13] *Discover Wi-Fi: Internet of Things* [online]. c2023. [cit. 2023-11-29]. Dostupné z: https://www.wi-fi.org/discover-wi-fi/internet-things.

[14] LUKAC, Martin; STUBAILO, Igor; GUY, Richard; DAVIS, Paul; PURUHUAYA, Victor; CLAYTON, Robert; ESTRIN, Deborah. First-class metadata: a step towards a highly reliable wireless seismic network in Peru. 2009, s. 3.

[15] *Bluetooth Low Energy (BLE): A Complete Guide* [online]. 2022. [cit. 2023-11-29]. Dostupné z: https://novelbits.io/bluetooth-low-energy-ble-complete-guide/.

[16] *Introduction of ZigBee* [online]. 2023. [cit. 2023-11-29]. Dostupné z: https://www.geeksforgeeks.org/introduction-of-zigbee/.

[17] *What is LoRaWAN® Specification* [online]. c2023. [cit. 2023-12-09]. Dostupné z: https://lora-alliance.org/about-lorawan/.

[18] *What is Sigfox 0G Technology* [online]. c2023. [cit. 2023-12-09]. Dostupné z: https://www.sigfox.com/what-is-sigfox/.

[19] LUTKEVICH, Ben. *What is an Embedded System?* [online]. c2005–2023. [cit. 2023-11-15]. Dostupné z: https://www.techtarget.com/iotagenda/definition/embedded-system.

[20] *Main Types of Embedded Systems Worth Knowing* [online]. 2023. [cit. 2023-11-15]. Dostupné z: https://www.velvetech.com/blog/types-of-embedded-systems.

[21] *Classification of Embedded Systems* [online]. 2020. [cit. 2023-11-15]. Dostupné z: https://www.geeksforgeeks.org/classification-of-embedded-systems.

[22] *An Introduction to Real-Time Embedded Systems* [online]. 2019. [cit. 2023-11-15]. Dostupné z: https://www.totalphase.com/blog/2019/12/an-introduction-to-real-time-embedded-systems/.

[23] *Assembly Embedded Systems: Practical Applications And Techniques* [online]. 2023. [cit. 2023-11-30]. Dostupné z: https://marketsplash.com/tutorials/assembly/assembly-embedded-systems/.

[24] *Assembly language* [online]. 2024. [cit. 2024-03-26]. Dostupné z: https://codedocs.org/what-is/assembly-language#Use_of_assembly_language.

[25] *Bare Bones Programming: The C Language* [online]. 2023. [cit. 2023-11-30]. Dostupné z: https://mathscitech.org/articles/c-for-systems.

[26] *Arduino Documentation* [online]. 2024. [cit. 2024-05-20]. Dostupné z: https://docs.arduino.cc/.

[27] *MicroPython* [online]. c2014–2023. [cit. 2023-11-15]. Dostupné z: https://micropython.org.

[28] BACCELLI, Emmanuel; GÜNDOĞAN, Cenk; HAHM, Oliver; KIETZMANN, Peter; LENDERS, Martine S.; PETERSEN, Hauke; SCHLEISER, Kaspar; SCHMIDT, Thomas C.; WÄHLISCH, Matthias. RIOT: An Open Source Operating System for Low-End Embedded Devices in the IoT. *IEEE Internet of Things Journal*. 2018, roč. 5, s. 4428–4440. Dostupné také z: https://api.semanticscholar.org/CorpusID:3924613.

[29] *Apache mynewt: Introduction* [online]. c2015–2021. [cit. 2023-11-20]. Dostupné z: https://mynewt.apache.org/latest/.

[30] *Zephyr Project Documentation* [online]. c2015–2023. [cit. 2023-11-20]. Dostupné z: https://docs.zephyrproject.org/latest/index.html.

[31] *Embedded Linux – Ubuntu* [online]. c2023. [cit. 2023-11-15]. Dostupné z: https://ubuntu.com/embedded.

[32] *QNX Neutrino Real-Time Operating System (RTOS)* [online]. c2023. [cit. 2023-11-20]. Dostupné z: https://blackberry.qnx.com/en/products/foundation-software/qnx-rtos.

[33] *RT-Thread Introduction* [online]. 2023. [cit. 2023-11-20]. Dostupné z: https://www.rt-thread.io/document/site/tutorial/quick-start/introduction/introduction/.

[34] ISMAIL, Noor Laili; KASSIM, Murizah; ISMAIL, Mahamod; MOHAMAD, Roslina. A review of low power wide area technology in licensed and unlicensed spectrum for IoT use cases. *Bulletin of Electrical Engineering and Informatics*. 2018, roč. 7, č. 2, s. 183–190.

[35] *Ceny za Windows 10 IoT Core Services* [online]. c2023. [cit. 2023-11-25]. Dostupné z: https://azure.microsoft.com/cs-cz/pricing/details/windows-10-iot-core/.

[36] HINCHEY, Mike; COYLE, Lorcan. Evolving critical systems: a research agenda for computer-based systems. 2012. Dostupné také z: https://researchrepository.ul.ie/articles/conference_contribution/Evolving_critical_systems_a_research_agenda_for_computer-based_systems_/19849987.

[37] *Windows for IoT Documentation* [online]. 2023. [cit. 2023-11-15]. Dostupné z: https://learn.microsoft.com/en-us/windows/iot.

[38] *OpenWrt Wiki* [online]. 2023. [cit. 2023-11-15]. Dostupné z: https://openwrt.org.

[39] *Introduction to HARDWARIO CHESTER* [online]. c2024. [cit. 2024-05-28]. Dostupné z: https://docs.hardwario.com/chester/.

[40] *ESP32-C3-DevKitM-1* [online]. c2024. [cit. 2024-05-28]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/hw-reference/esp32c3/user-guide-devkitm-1.html.

[41] *Adafruit Feather M0 Basic Proto - ATSAMD21 Cortex M0* [online]. c2024. [cit. 2024-05-28]. Dostupné z: https://www.adafruit.com/product/2772.

[42] *HiFive1 Rev B* [online]. c2024. [cit. 2024-05-28]. Dostupné z: https://www.sifive.com/boards/hifive1-rev-b.

[43] *Raspberry Pi 4 Tech Specs* [online]. c2024. [cit. 2024-05-28]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/.

[44] *Raspberry Pi Pico and Pico W* [online]. c2024. [cit. 2024-05-28]. Dostupné z: https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html.

[45] *STM32L152RE* [online]. c2024. [cit. 2024-05-28]. Dostupné z: https://www.st.com/en/microcontrollers-microprocessors/stm32l152re.html.

[46] *Quick reference for the pyboard* [online]. c2024. [cit. 2024-05-28]. Dostupné z: https://docs.micropython.org/en/latest/pyboard/quickref.html.

[47] BACCELLI, Emmanuel; GÜNDOĞAN, Cenk; HAHM, Oliver; KIETZMANN, Peter; LENDERS, Martine S.; PETERSEN, Hauke; SCHLEISER, Kaspar; SCHMIDT, Thomas C.; WÄHLISCH, Matthias. RIOT: An Open Source Operating System for Low-End Embedded Devices in the IoT. *IEEE Internet of Things Journal*. 2018, roč. 5, č. 6, s. 4428–4440. Dostupné z DOI: 10.1109/JIOT.2018.2815038.

[48] *MicroPython documentation* [online]. 2023. [cit. 2023-12-03]. Dostupné z: https://docs.micropython.org/en/latest/.

[49] *Zephyr Project Documentation* [online]. 2023. [cit. 2023-12-03]. Dostupné z: https://docs.zephyrproject.org/latest/index.html.

Seznam symbolů a zkratek

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks

6TiSCH IPv6 over the TSCH mode of IEEE 802.15.4e

API Application Programming Interface

ARC Argonaut RISC Core

ARM Advanced RISC Machines

BLE5 Bluetooth Low Energy 5

BLE Bluetooth Low Energy

CMSIS Cortex Microcontroller Software Interface Standard

CoAP Constrained Application Protocol

DSP Digital Signal Processor

FIFO First In, First Out

FPGA Field Programmable Gate Array

HAL Hardware Abstraction Layer

IDE Integrated Development Enviroment

IPC Inter-Process Communication

ISA Instruction Set Architecture

IoT Internet of Things

JFFS2 Journalling Flash File System version 2

LPWAN Low-Power Wide Area Network

LR-WPAN Low-Rate Wireless Personal Area Network

LTE Cat M Long-Term Evolution Caterogy Machine

LoRaWAN Long Range Wide Area Network

MCU Microcontroller Unit

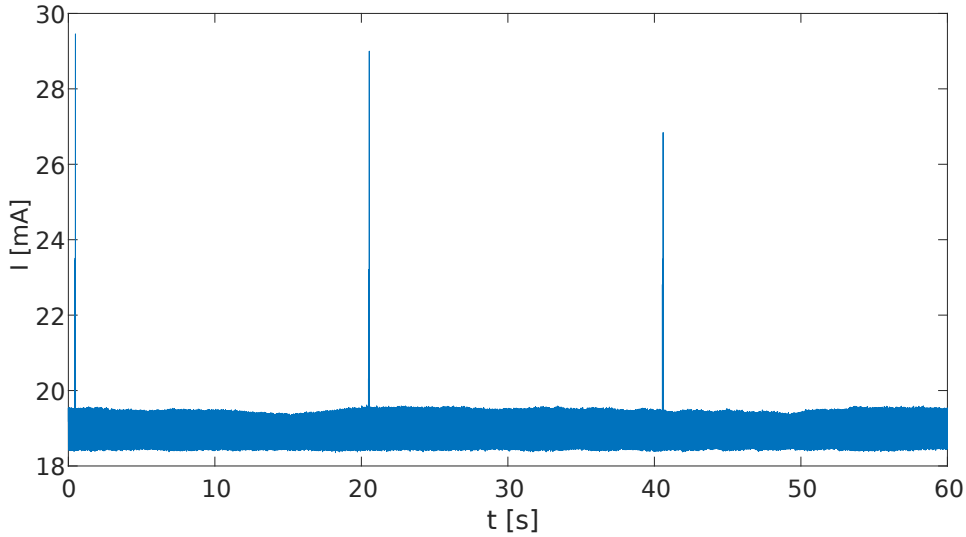
MIPS Microprocessor without Interlocked Pipelined Stages

MMU	Memory Management Unit
MPU	Memory Protection Unit
NB-IoT	Narrowband Internet of Things
OpenWrt	Open Wireless router
OS	Operating System
OTA	Over The Air
POSIX	Portable Operating System Interface
PoE	Power over Ethernet
PWM	Pulse Wave Modulation
RFID	Radio-Frequency Identification
RISC	Reduced Instruction Set Computer
RPL	Routing Protocol for Low Power and Lossy Networks
RTOS	Real-Time Operating System
RTT	Real Time Timer
RTX	Real-Time eXecutive
SaaS	Software as a Service
SDK	Software Development Kit
SoC	System on Chip
UART	Universal Asynchronous Receiver-Transmiter
YAFFS	Yet Another Flash File System

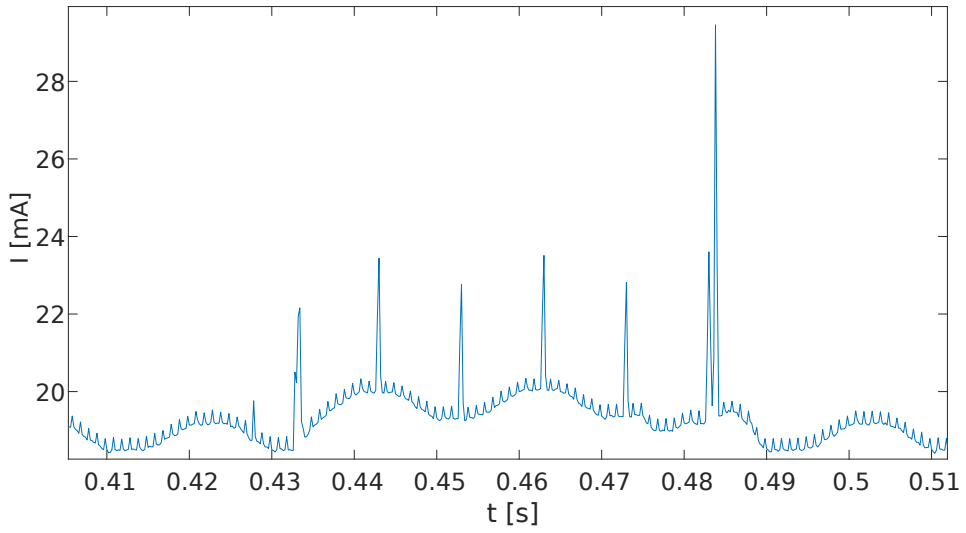
A Přehled grafů podle HW platforem

A.1 ESP32C3 DevKitM1

A.1.1 Arduino

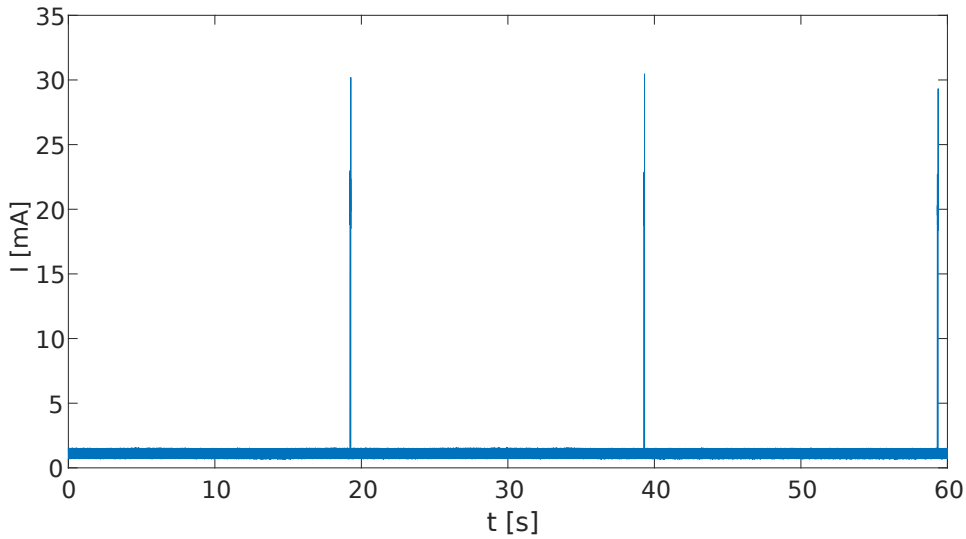


(a) 3 cykly

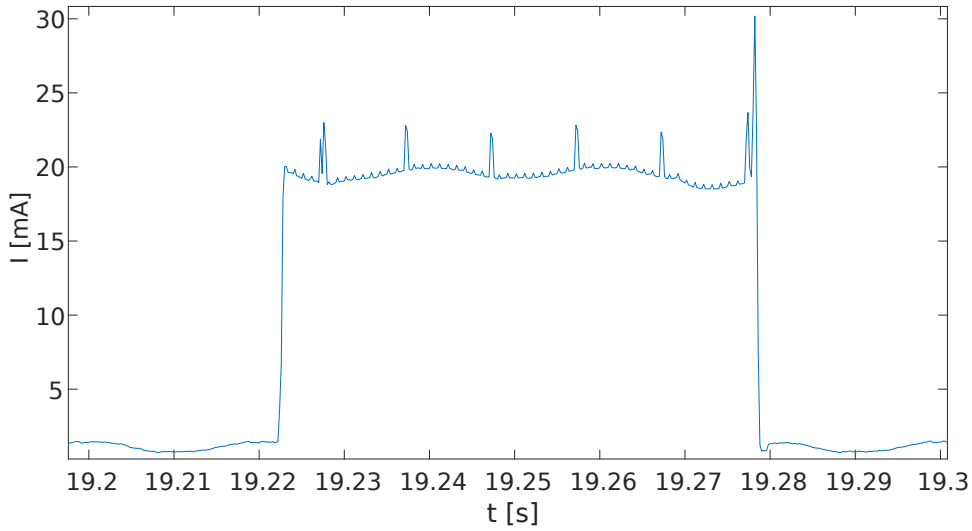


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.1: Arduino na ESP bez nízkoodběrového módu

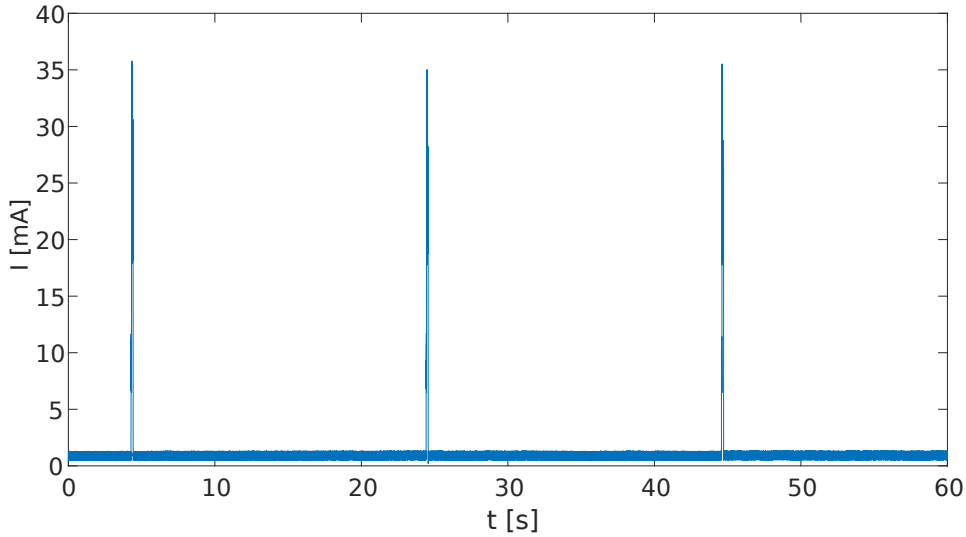


(a) 3 cykly

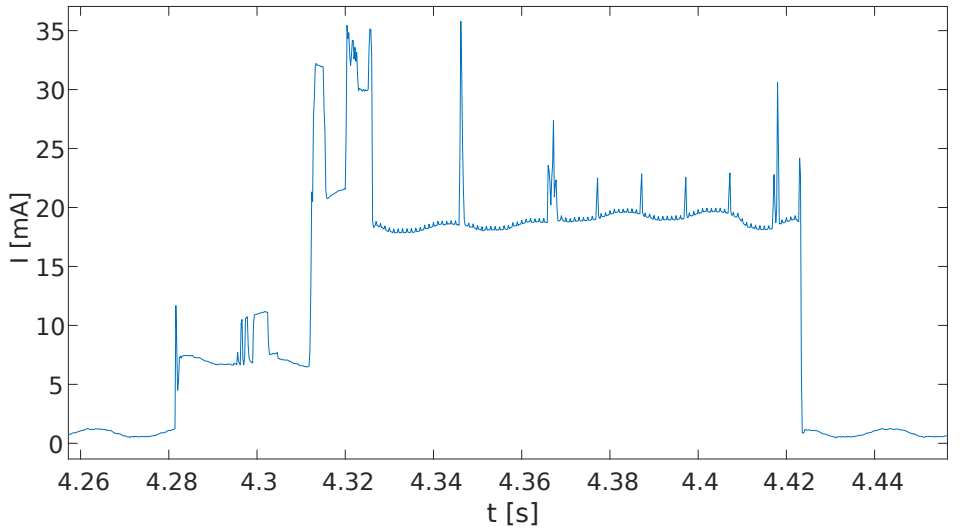


(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

Obr. A.2: Arduino na ESP v módu light sleep



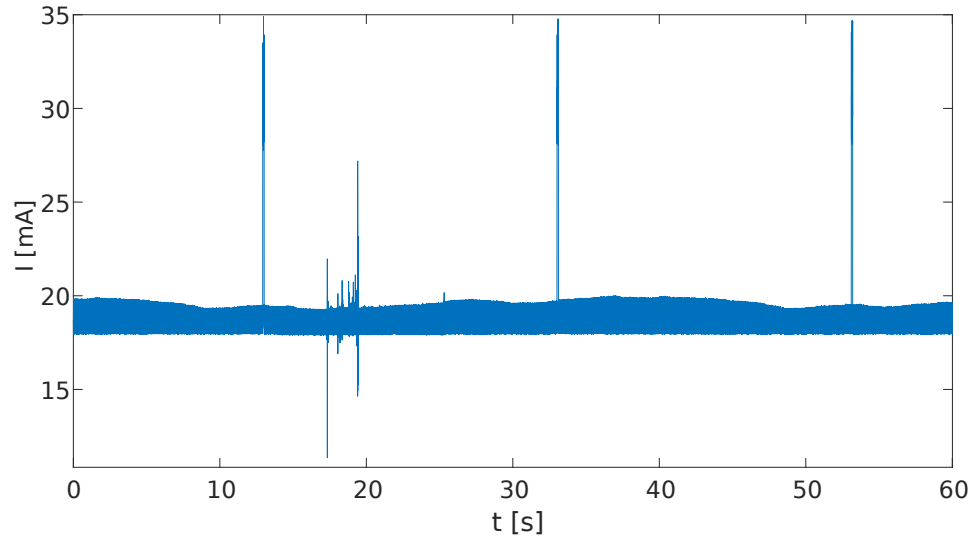
(a) 3 cykly



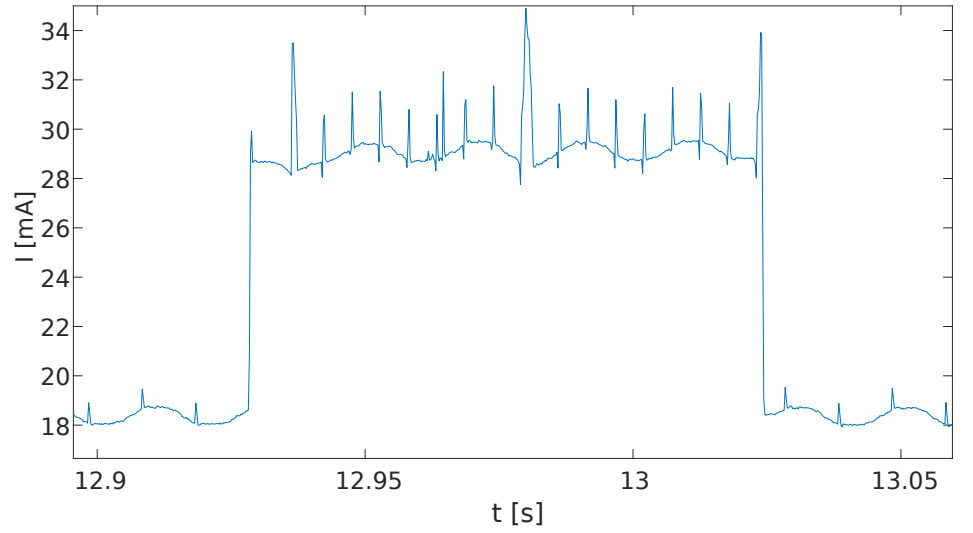
(b) Detail odebíraného proudu při změně módu deep sleep na aktivní a zpět do deep sleep.

Obr. A.3: Arduino na ESP v módu deep sleep

A.1.2 Mikropython

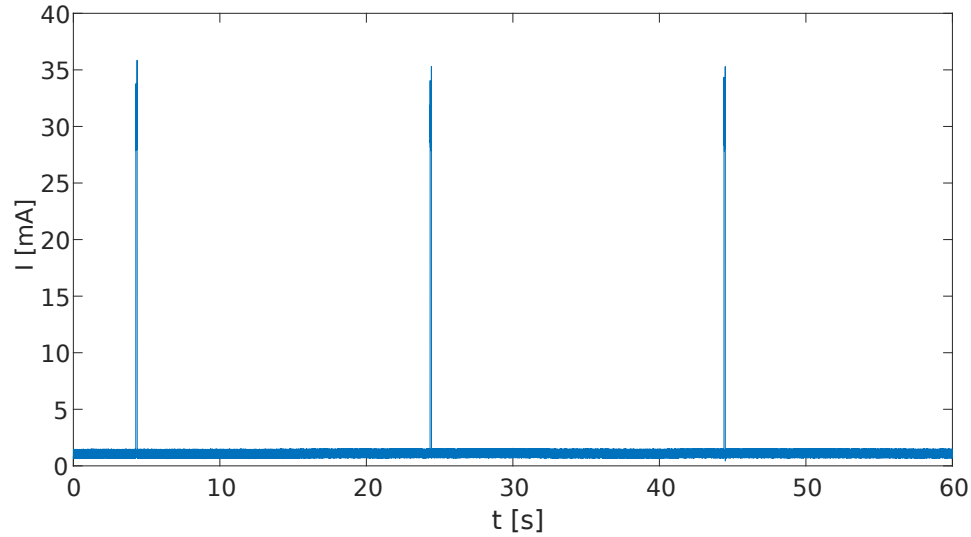


(a) 3 cykly

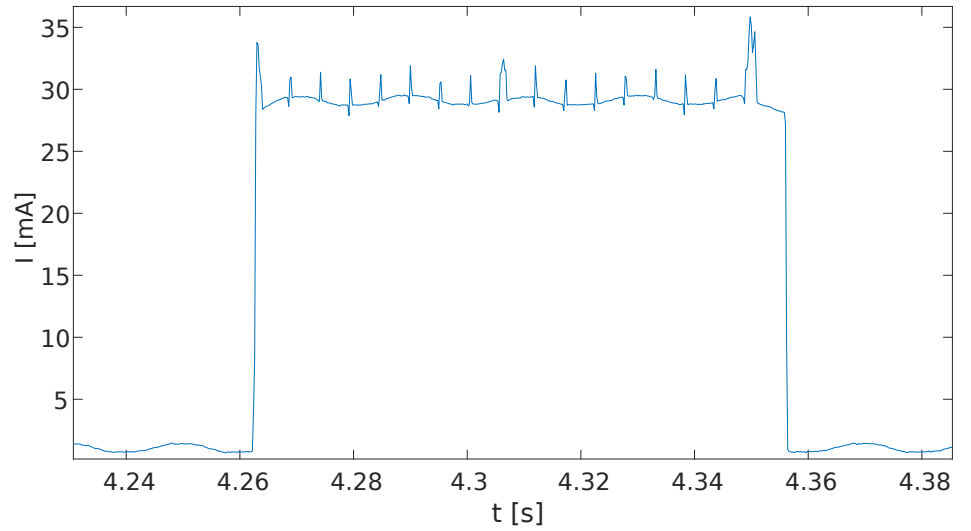


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.4: Mikropython na ESP bez nízkoodběrového módu



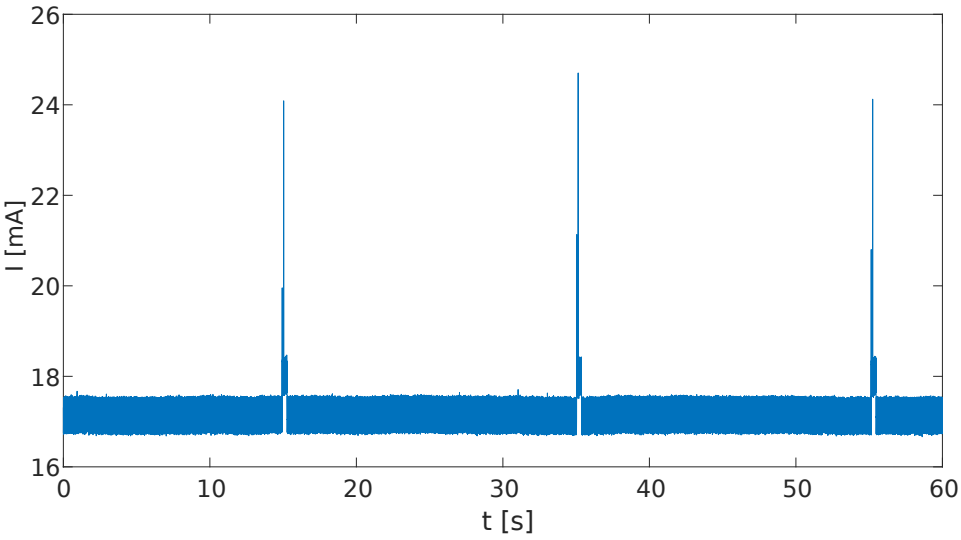
(a) 3 cykly



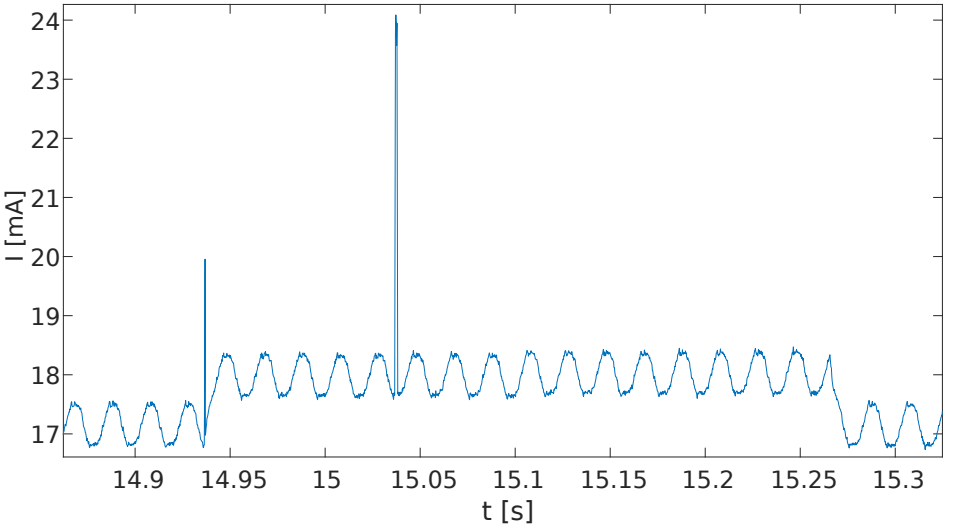
(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

Obr. A.5: Mikropython na ESP v módu light sleep

A.1.3 RIOT

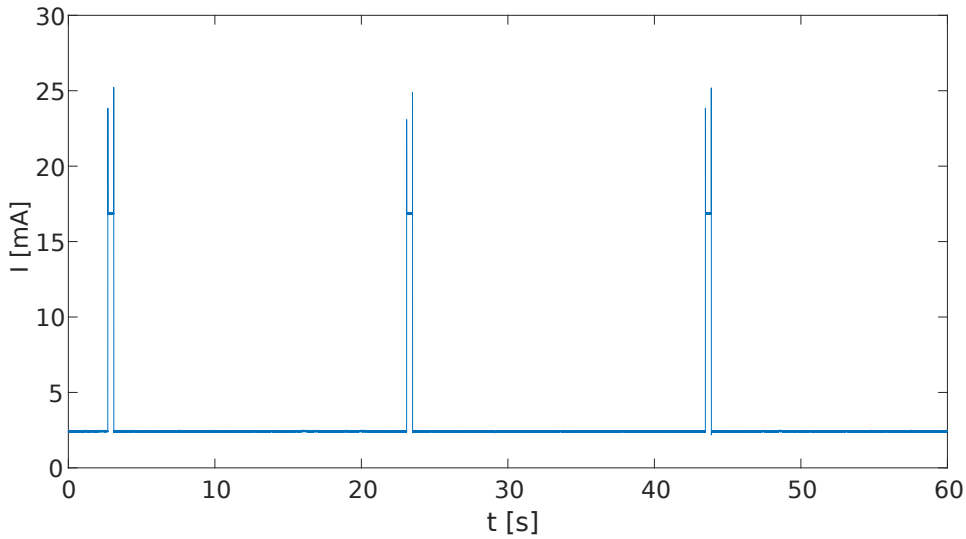


(a) 3 cykly

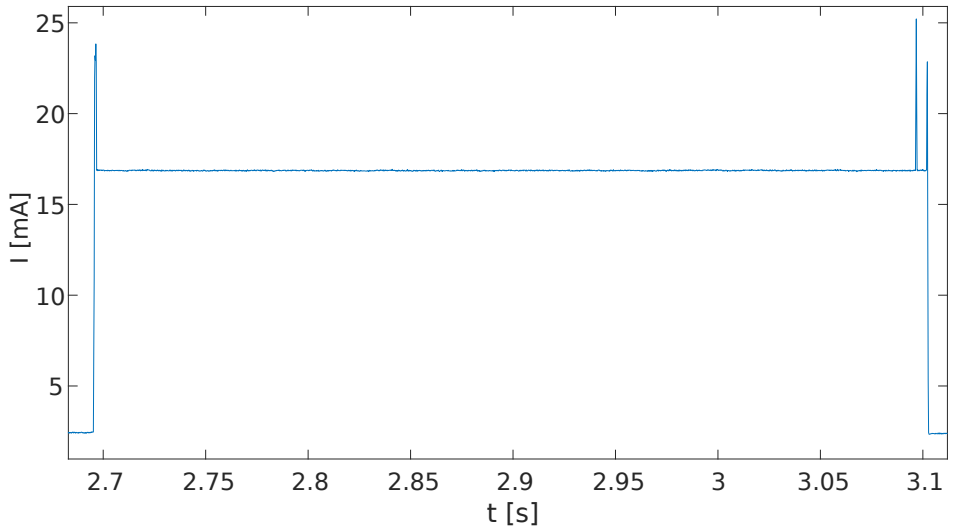


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.6: RIOT na ESP bez nízkoodběrového módu

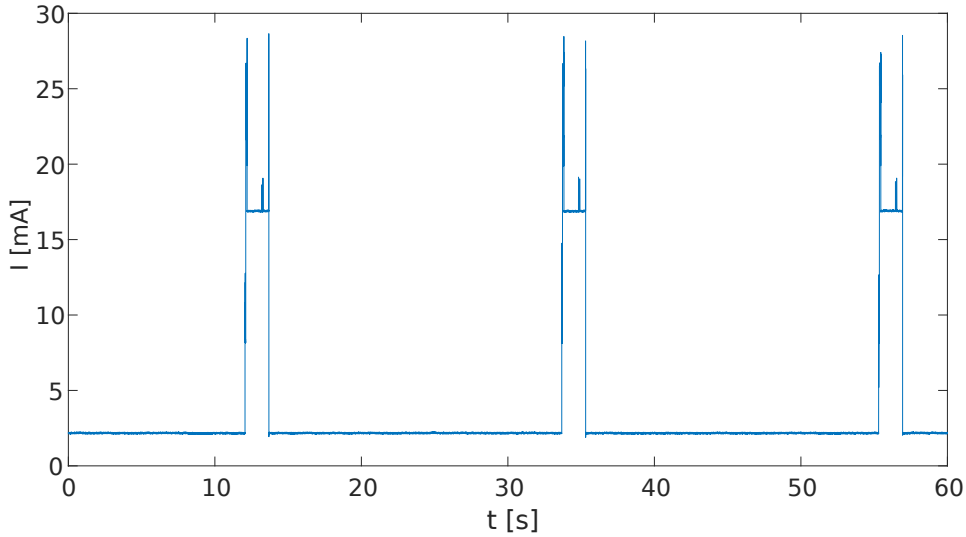


(a) 3 cykly

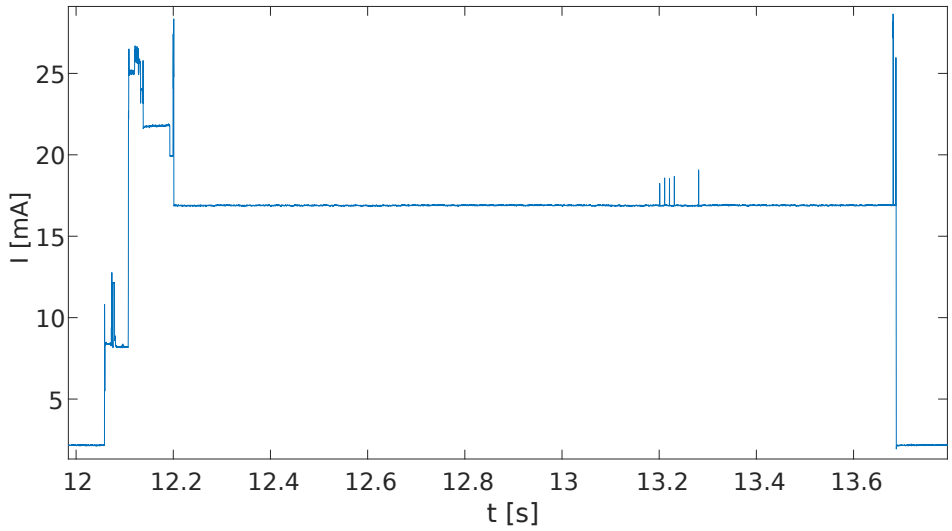


(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

Obr. A.7: RIOT na ESP v módu light sleep



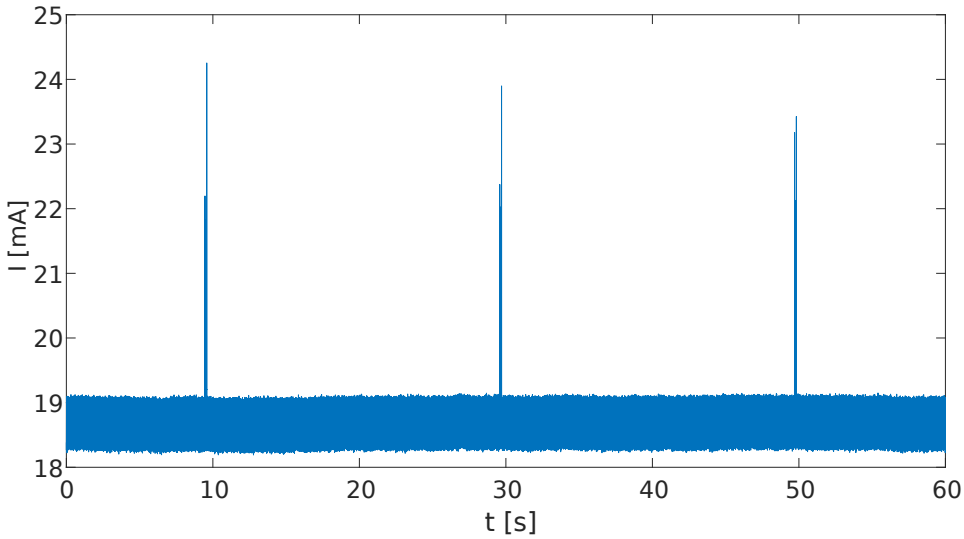
(a) 3 cykly



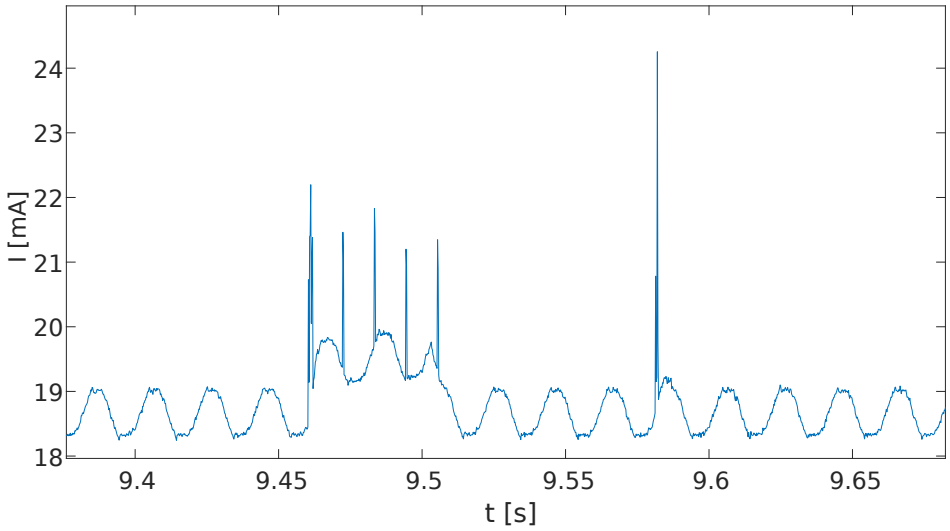
(b) Detail odebíraného proudu při změně módu deep sleep na aktivní a zpět do deep sleep.

Obr. A.8: RIOT na ESP v módu deep sleep

A.1.4 Zephyr

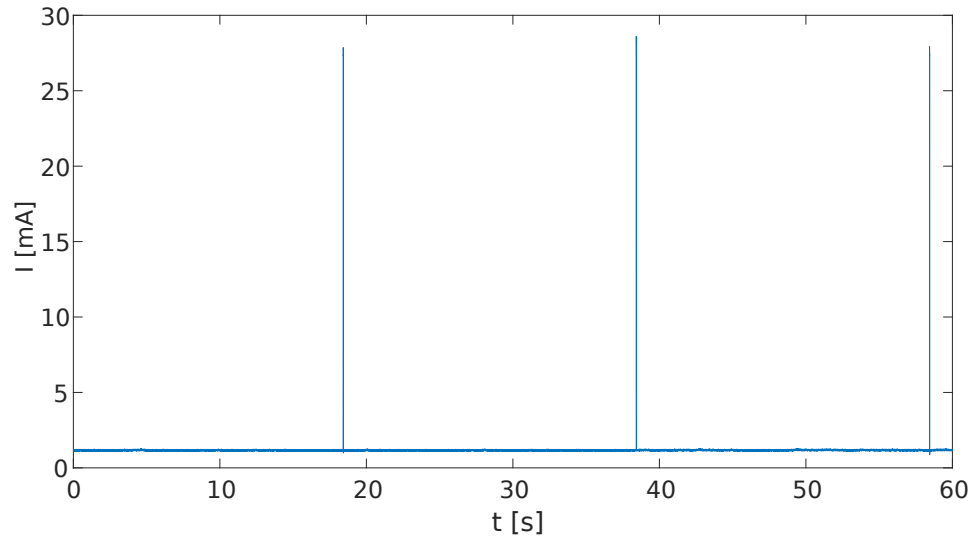


(a) 3 cykly

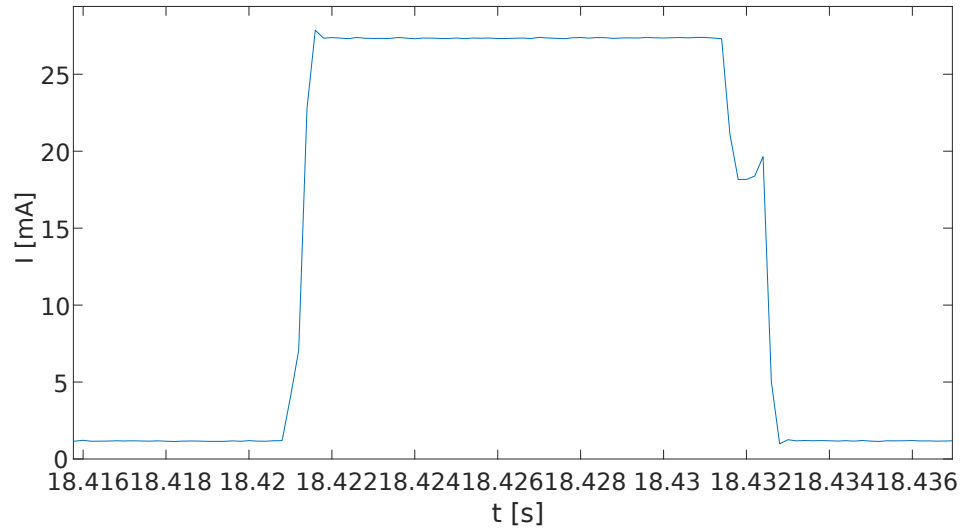


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.9: Zephyr na ESP bez nízkoodběrového módu

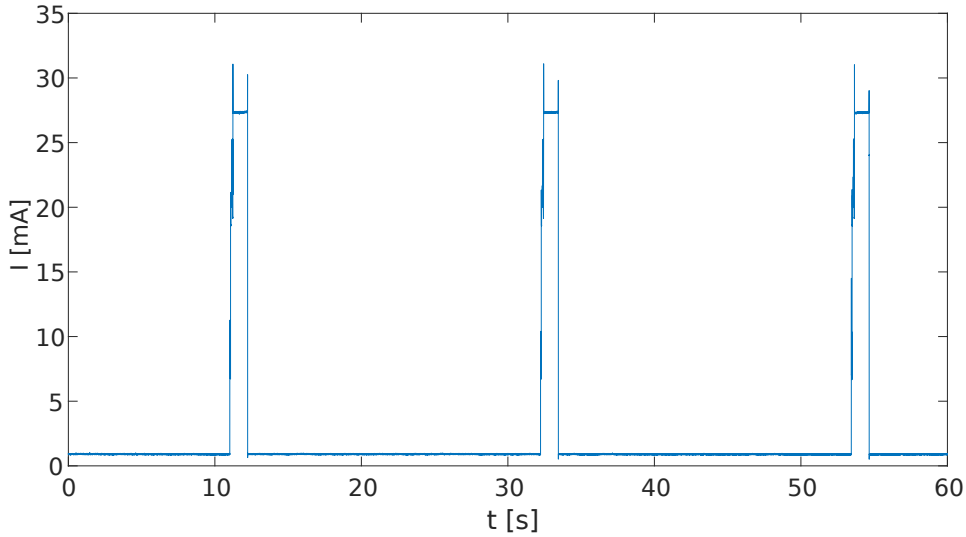


(a) 3 cykly

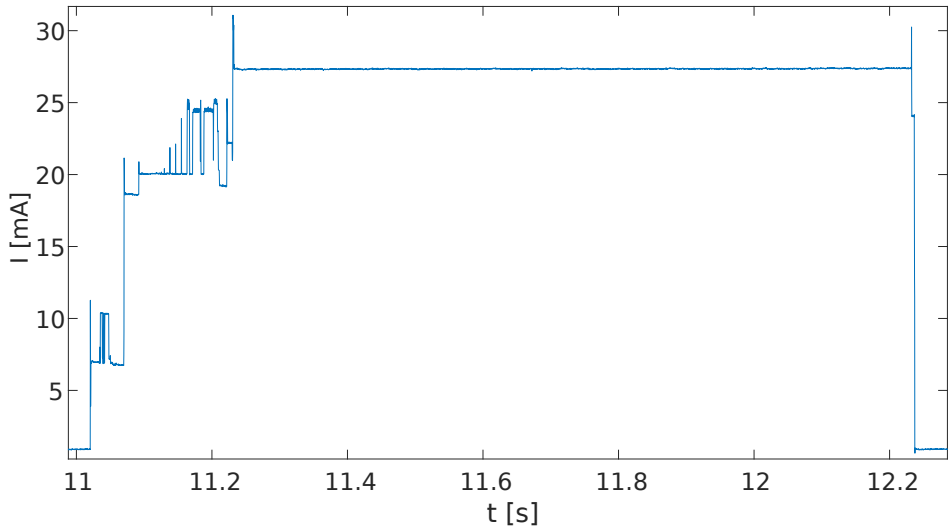


(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

Obr. A.10: Zephyr na ESP v módu light sleep



(a) 3 cykly

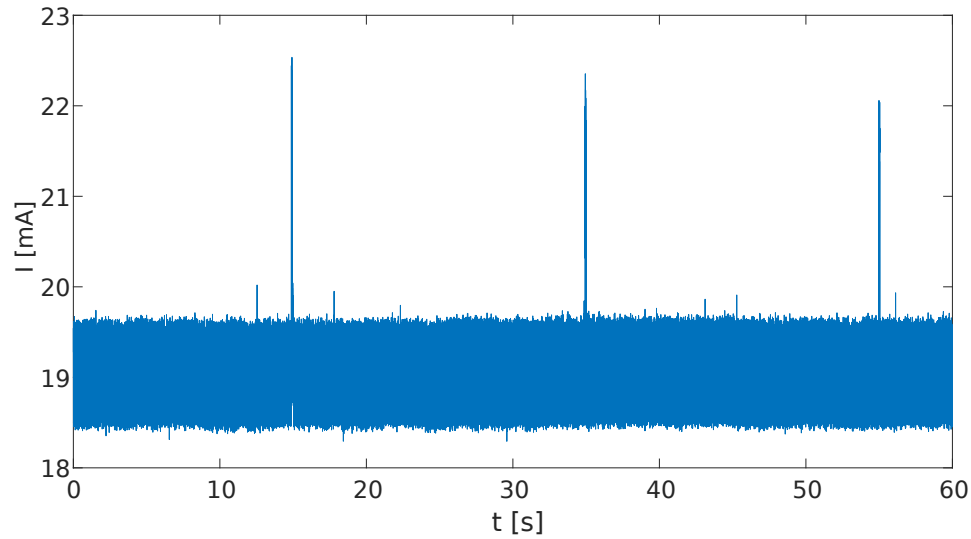


(b) Detail odebíraného proudu při změně módu deep sleep na aktivní a zpět do deep sleep.

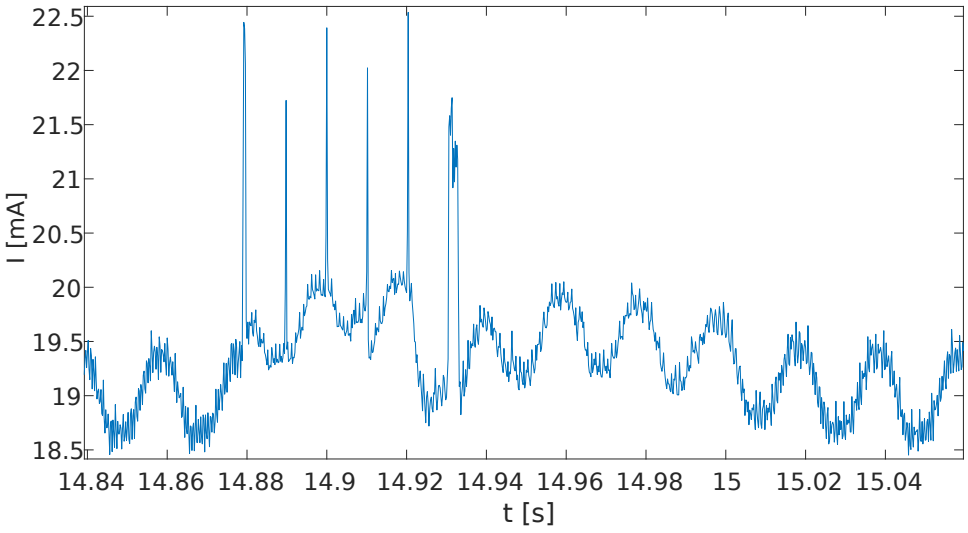
Obr. A.11: Zephyr na ESP v módu deep sleep

A.2 Raspberry Pi Pico

A.2.1 Arduino



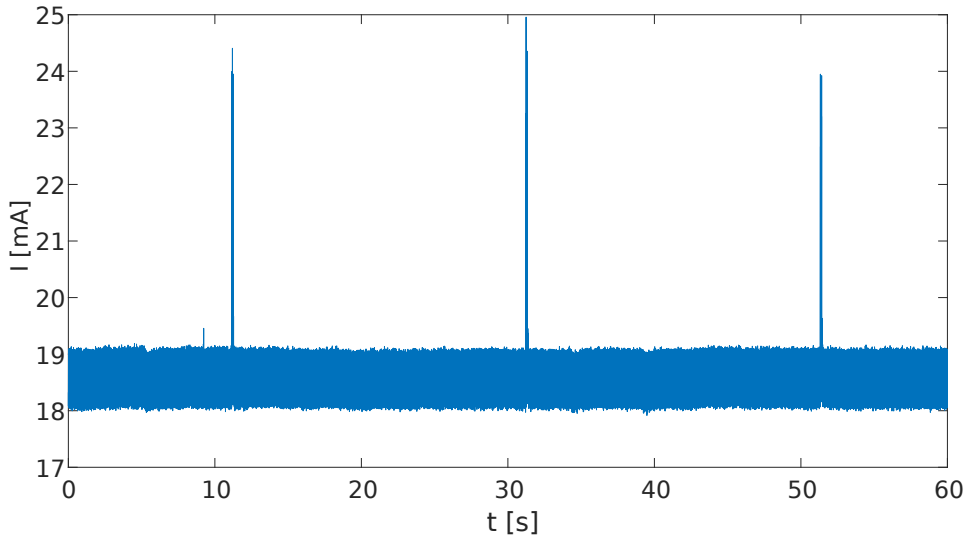
(a) 3 cykly



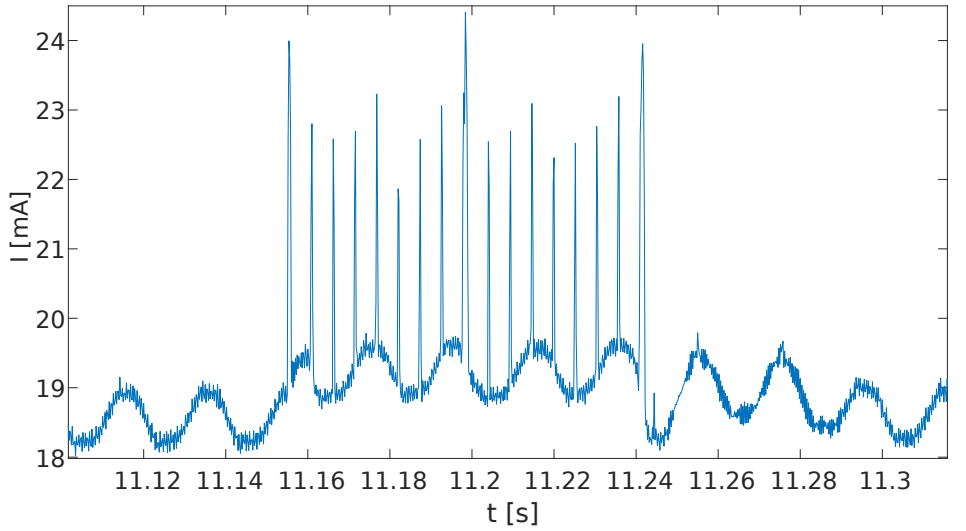
(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.12: Arduino na pico bez nízkoodběrového módu

A.2.2 Mikropython

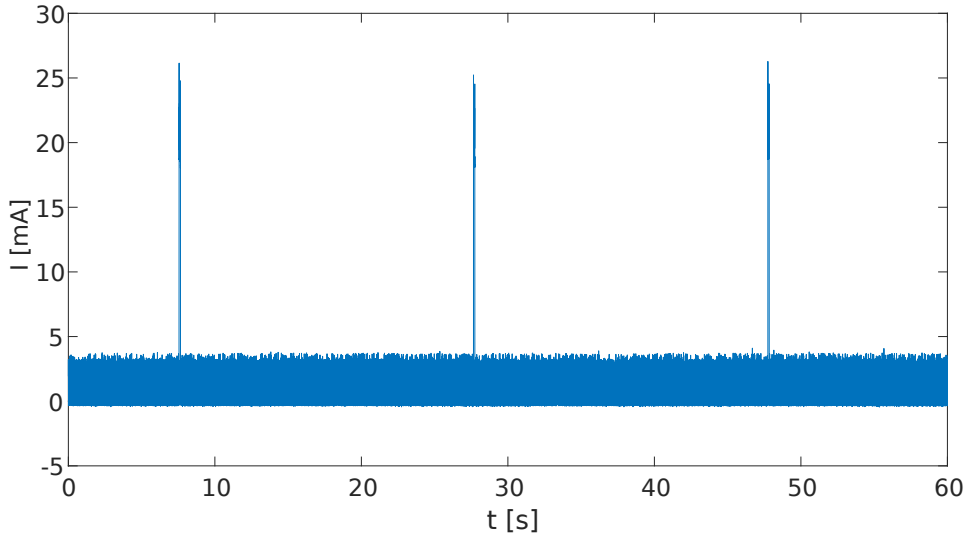


(a) 3 cykly

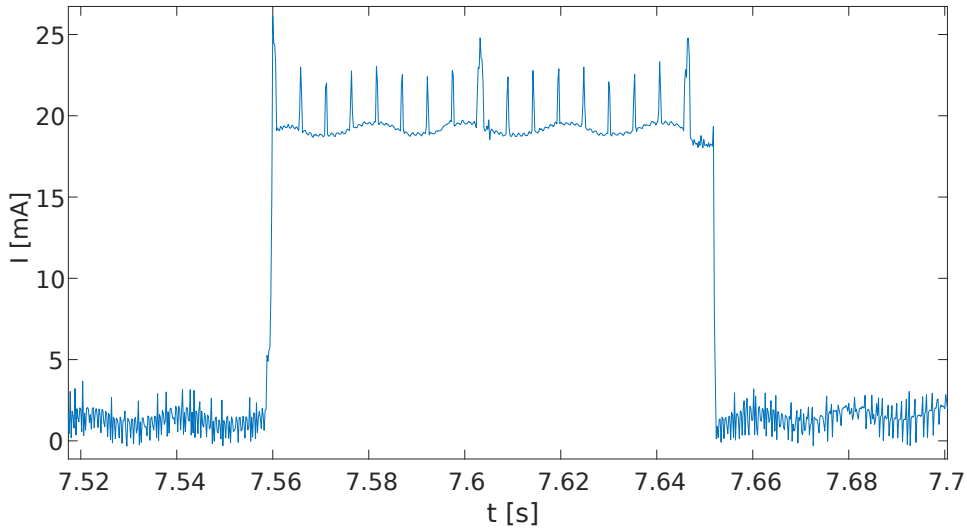


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.13: Mikropython na pico bez nízkoodběrového módu

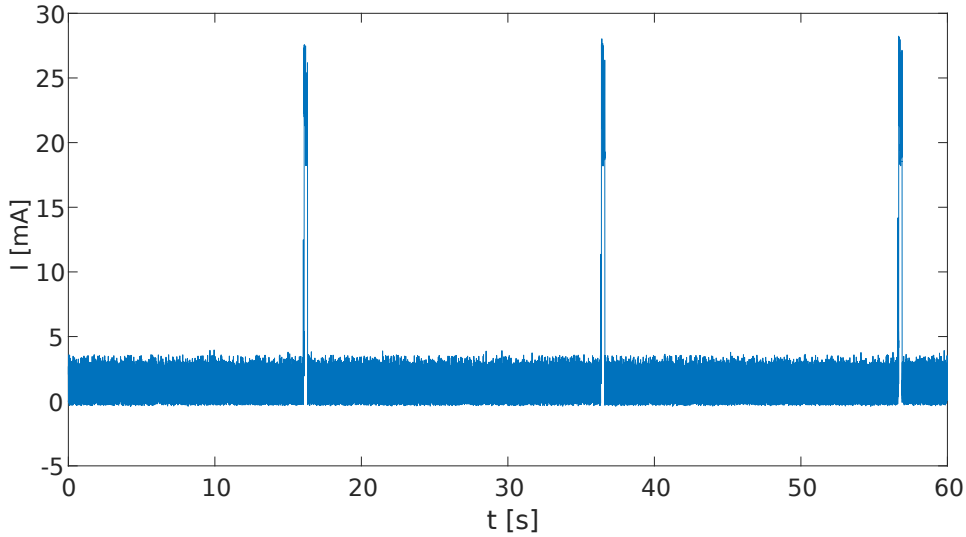


(a) 3 cykly

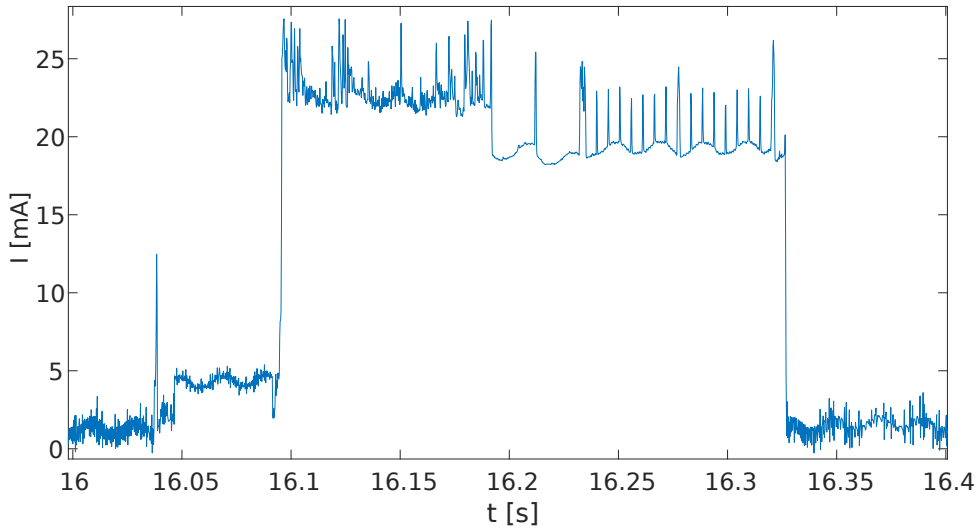


(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

Obr. A.14: Mikropython na pico v módu light sleep



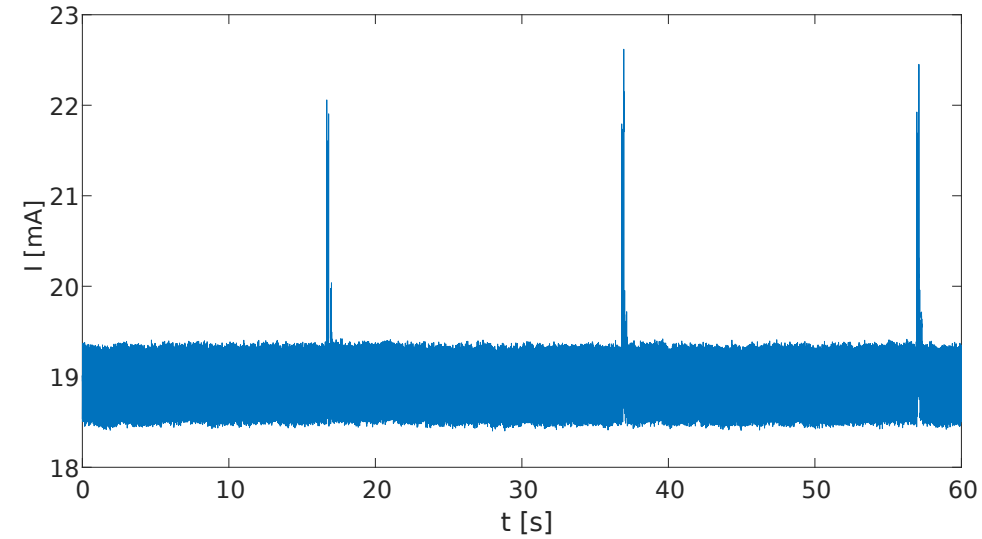
(a) 3 cykly



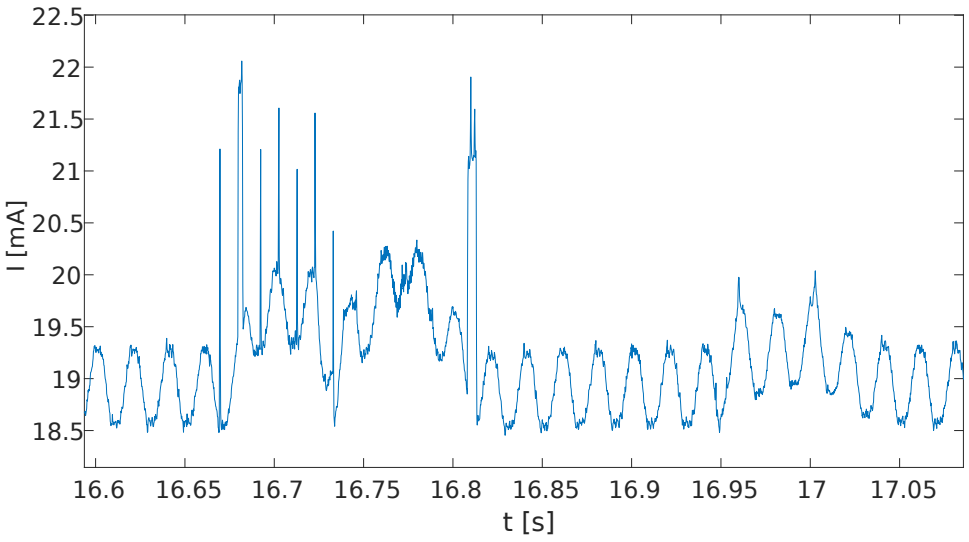
(b) Detail odebíraného proudu při změně módu deep sleep na aktivní a zpět do deep sleep.

Obr. A.15: Mikropython na pico v módu deep sleep

A.2.3 Zephyr

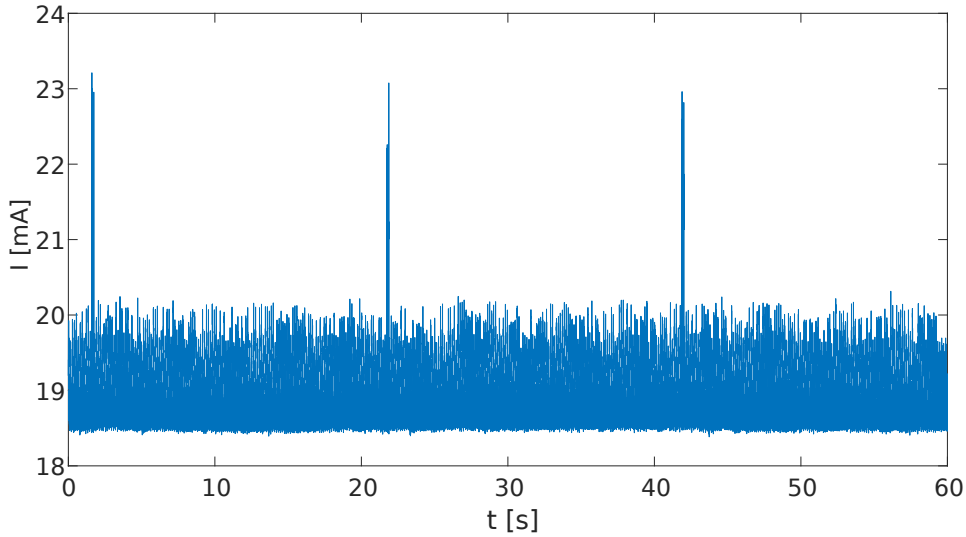


(a) 3 cykly

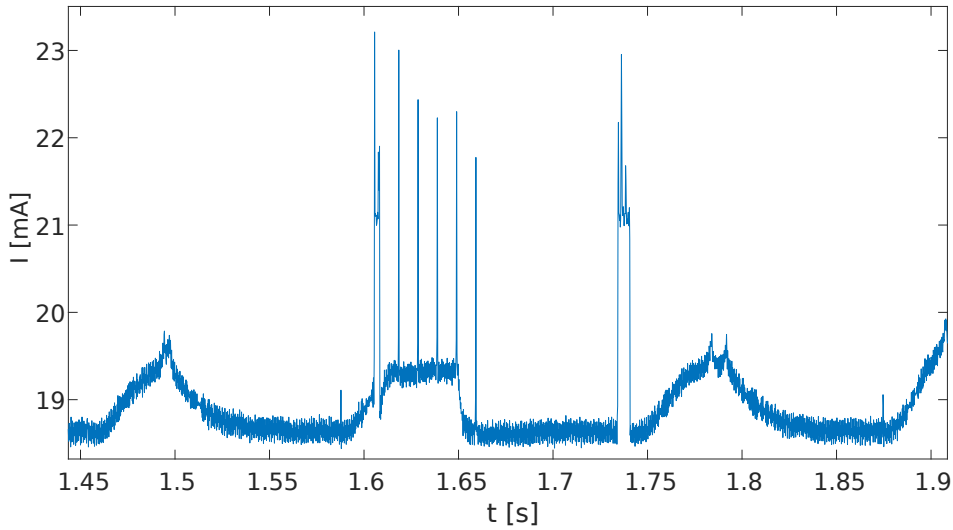


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.16: Zephyr na pico bez nízkoodběrového módu



(a) 3 cykly

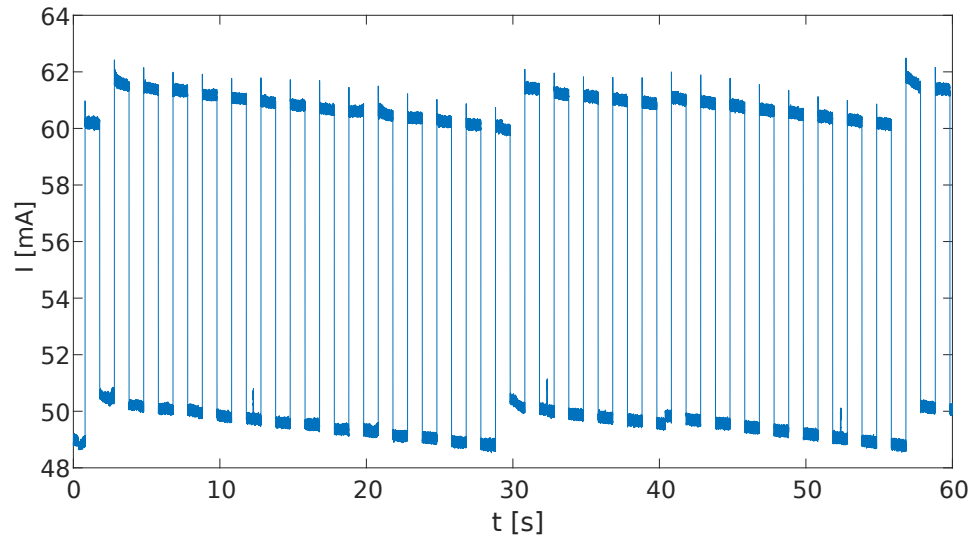


(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

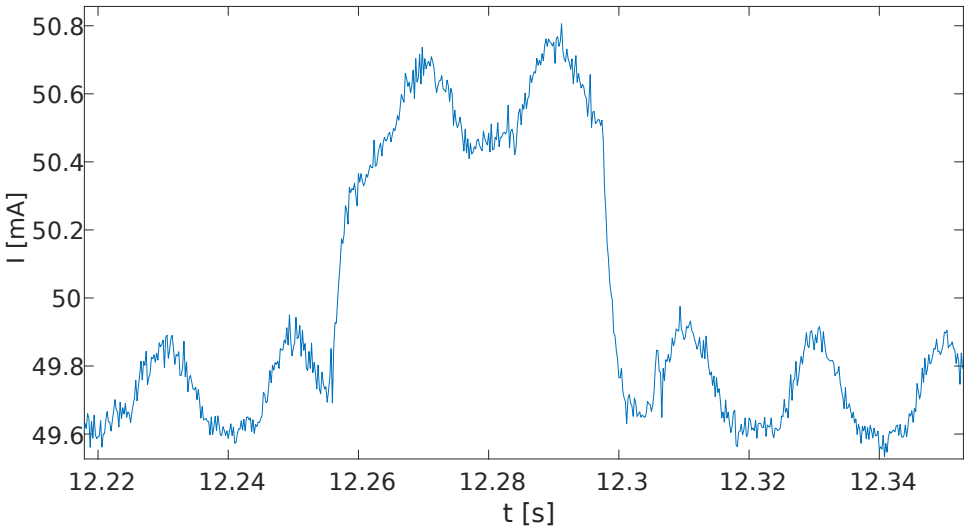
Obr. A.17: Zephyr na pico v módu light sleep

A.3 STM32 Nucleo-L152RE

A.3.1 Arduino

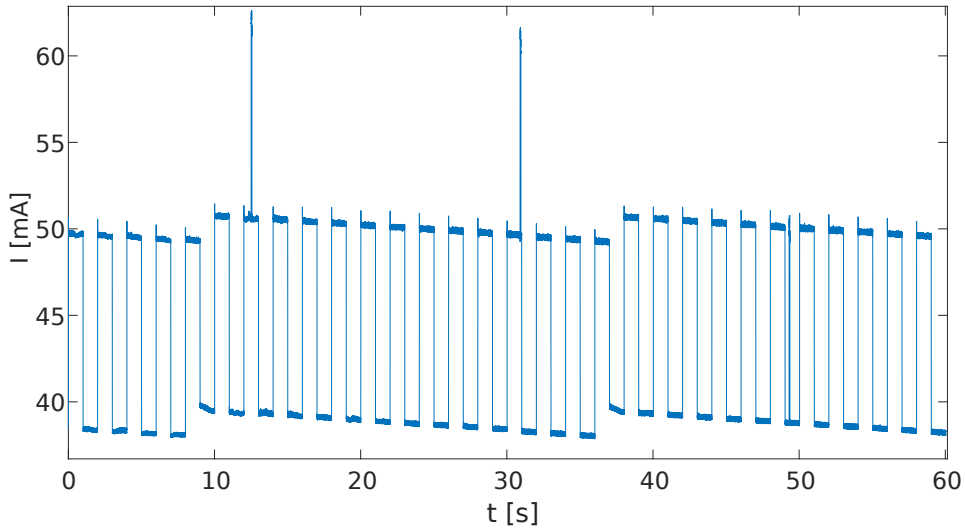


(a) 3 cykly

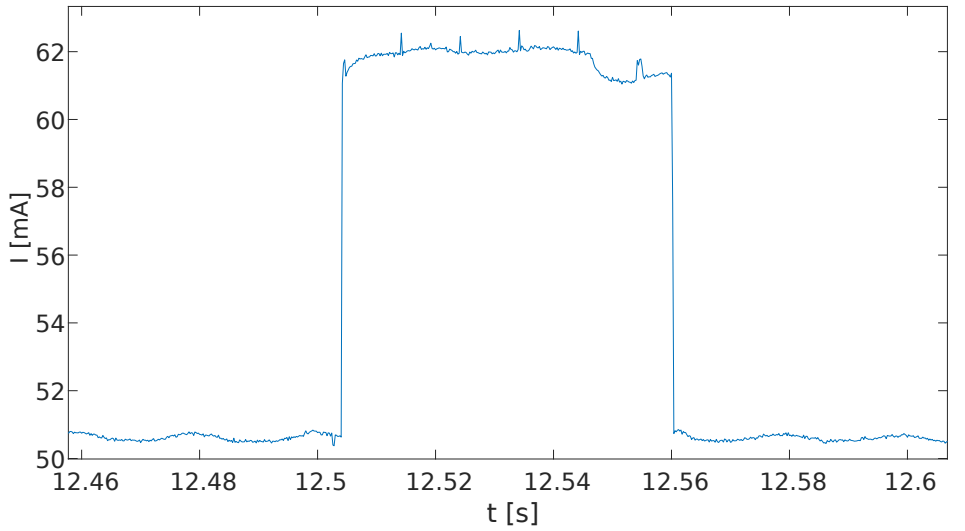


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.18: Arduino na STM bez nízkoodběrového módu

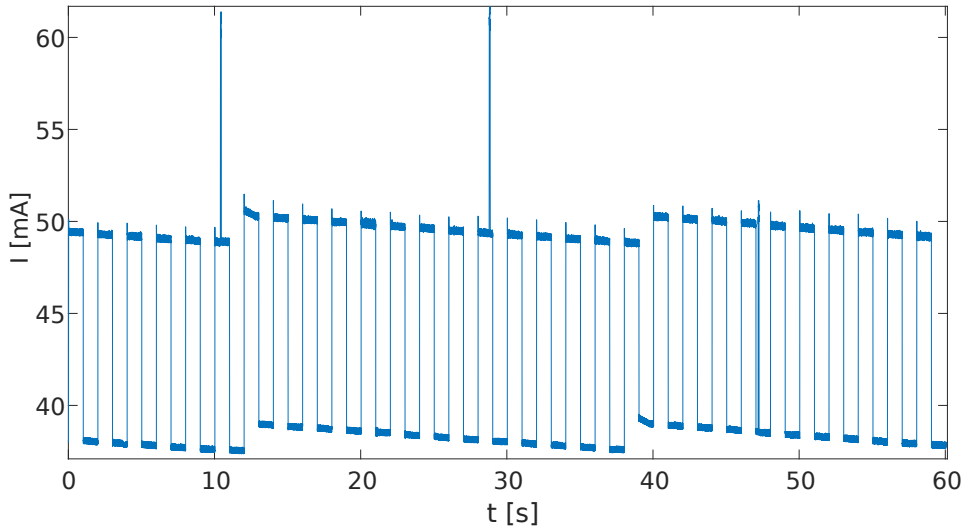


(a) 3 cykly

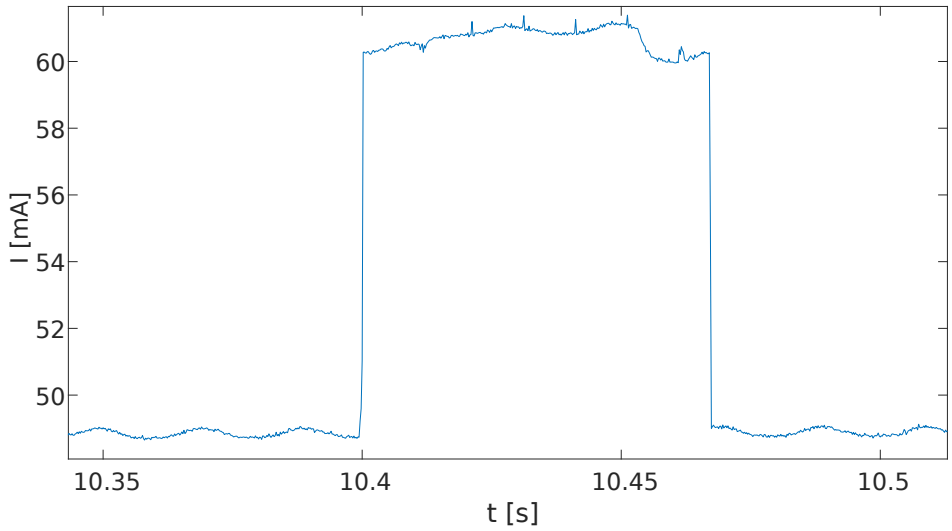


(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

Obr. A.19: Arduino na STM v módu light sleep



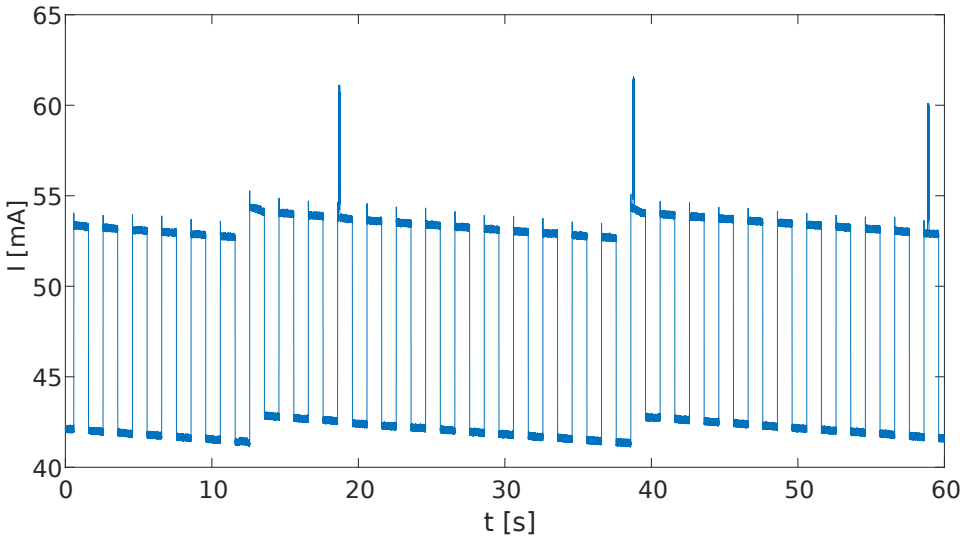
(a) 3 cykly



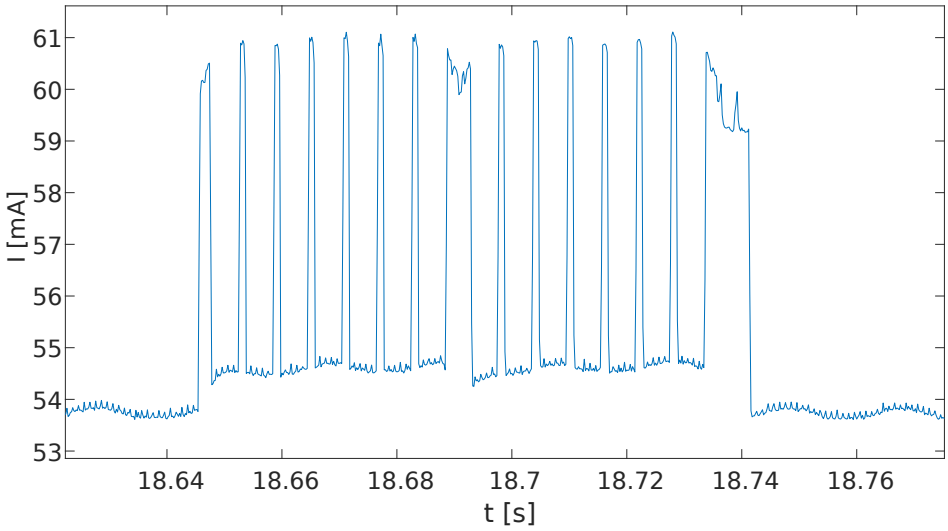
(b) Detail odebíraného proudu při změně módu deep sleep na aktivní a zpět do deep sleep.

Obr. A.20: Arduino na STM v módu deep sleep

A.3.2 Mikropython

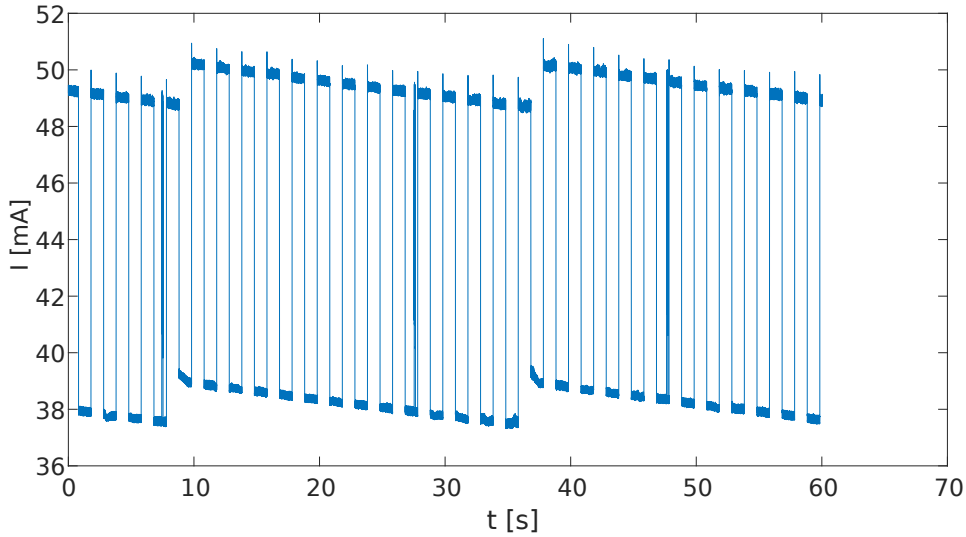


(a) 3 cykly

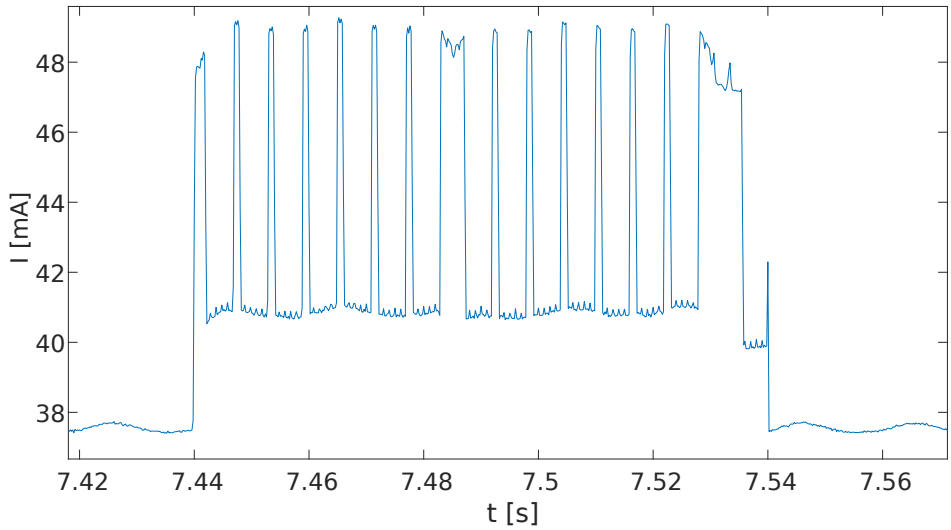


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.21: Mikropython na STM bez nízkoodběrového módu

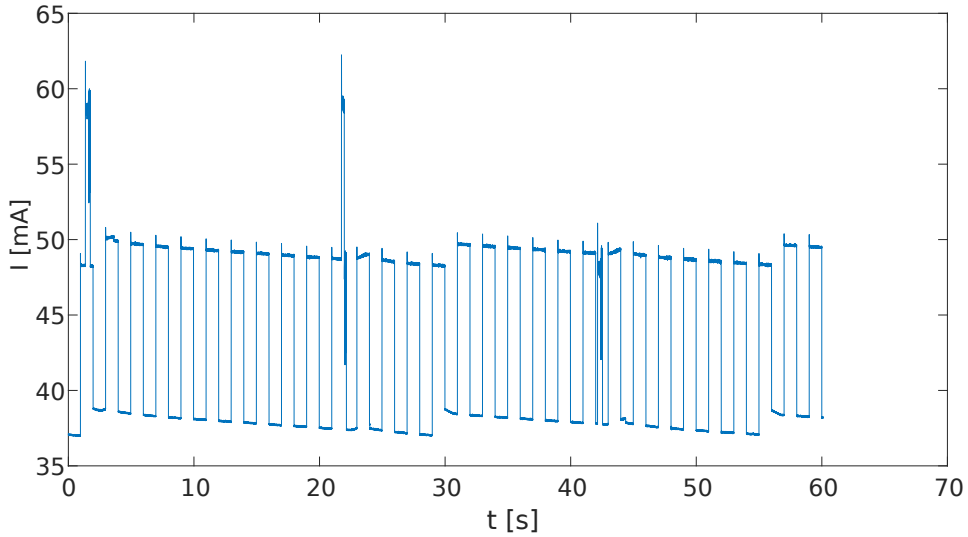


(a) 3 cykly

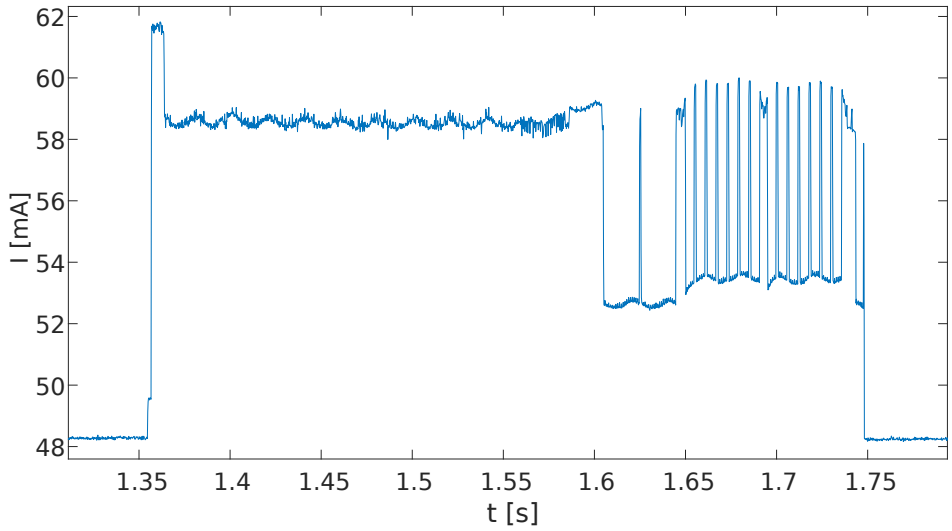


(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

Obr. A.22: Mikropython na STM v módu light sleep



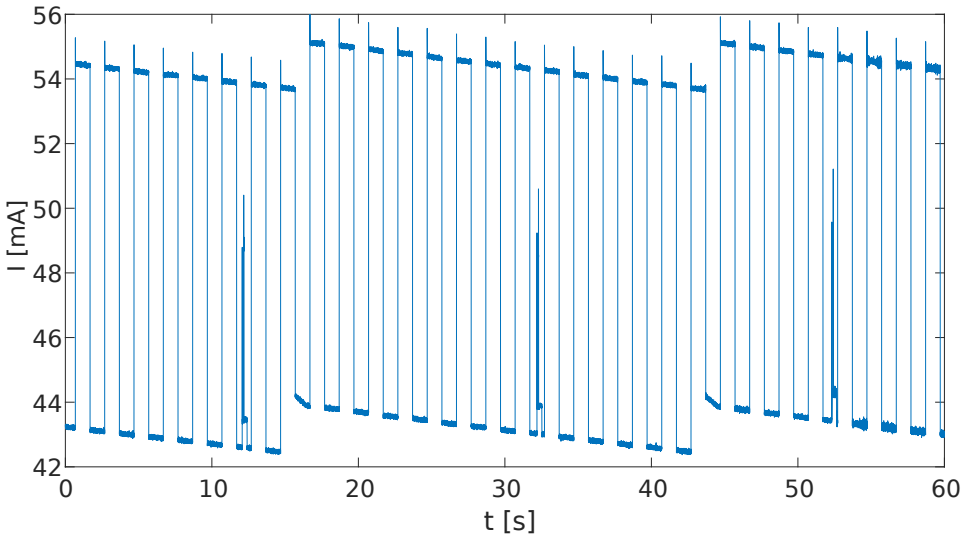
(a) 3 cykly



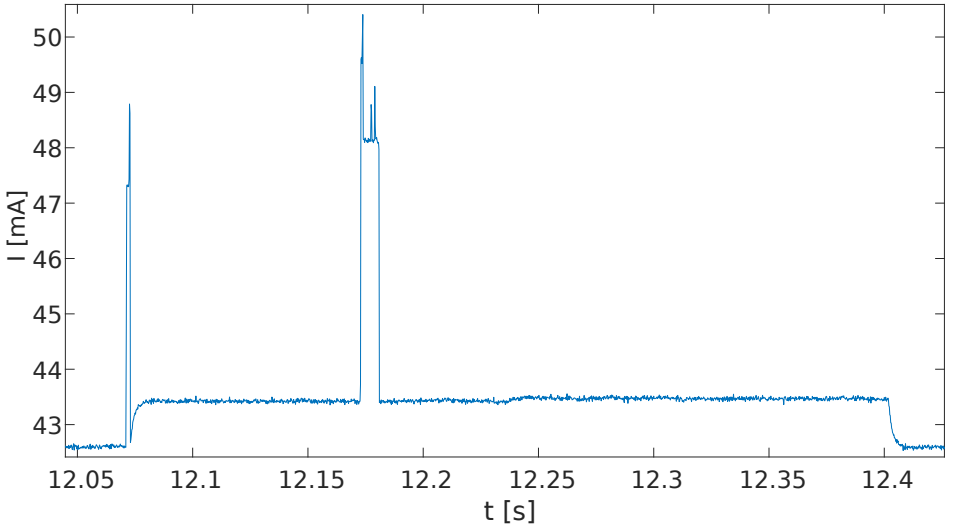
(b) Detail odebíraného proudu při změně módu deep sleep na aktivní a zpět do deep sleep.

Obr. A.23: Mikropython na STM v módu deep sleep

A.3.3 RIOT

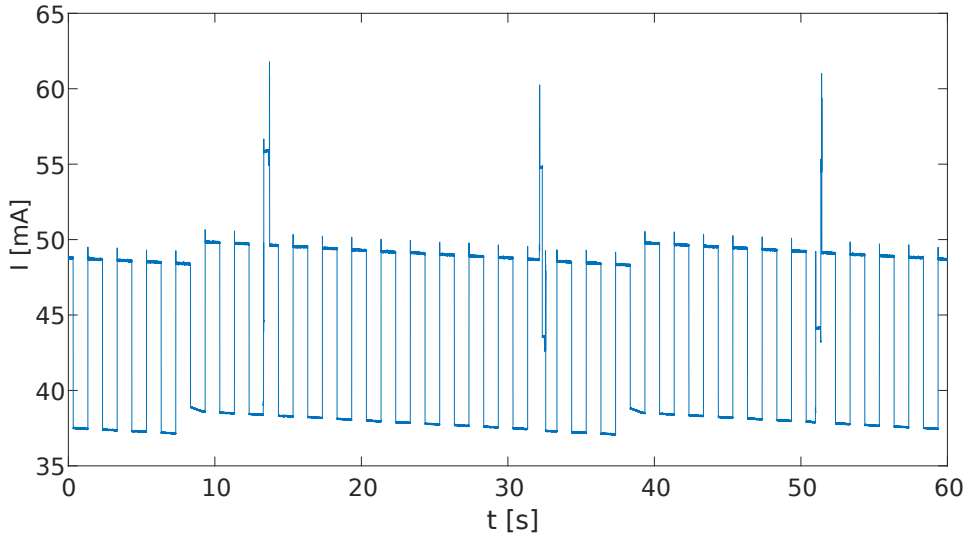


(a) 3 cykly

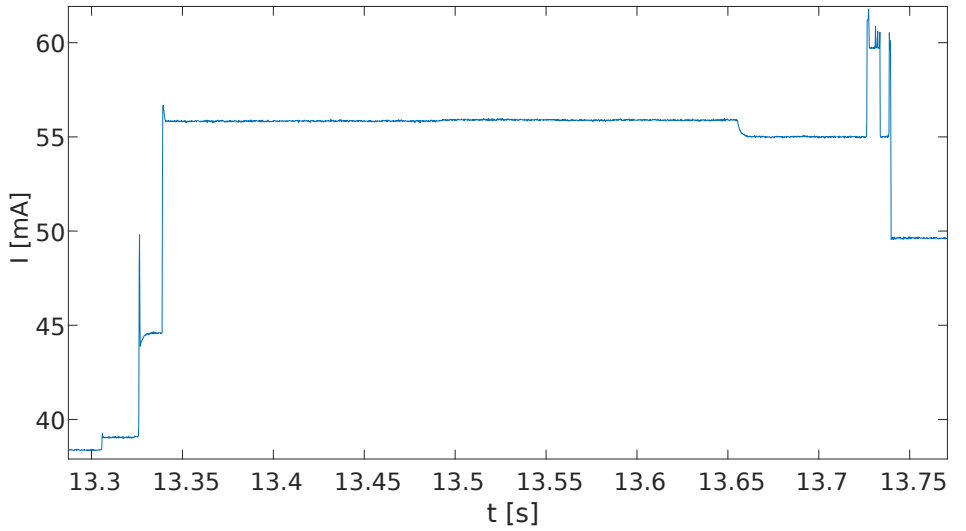


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.24: RIOT na STM bez nízkoodběrového módu

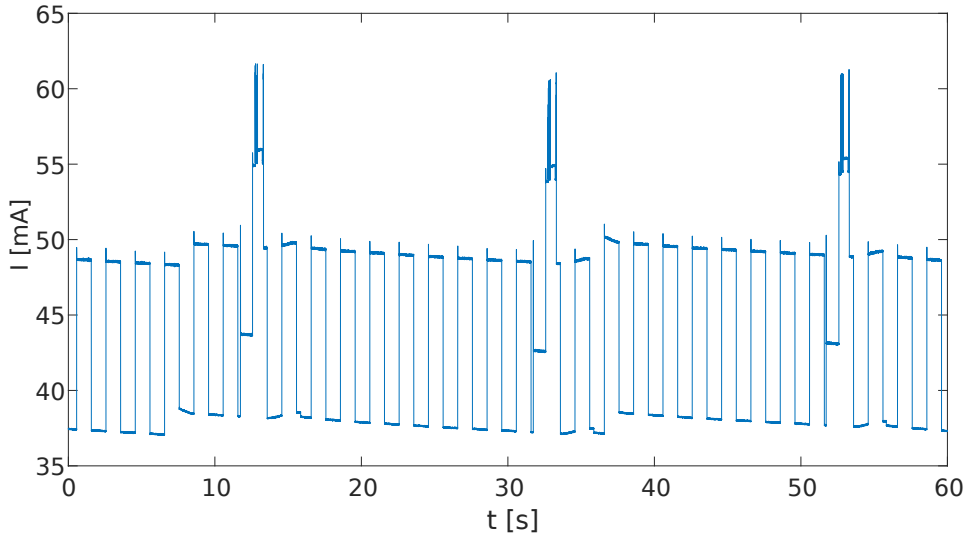


(a) 3 cykly

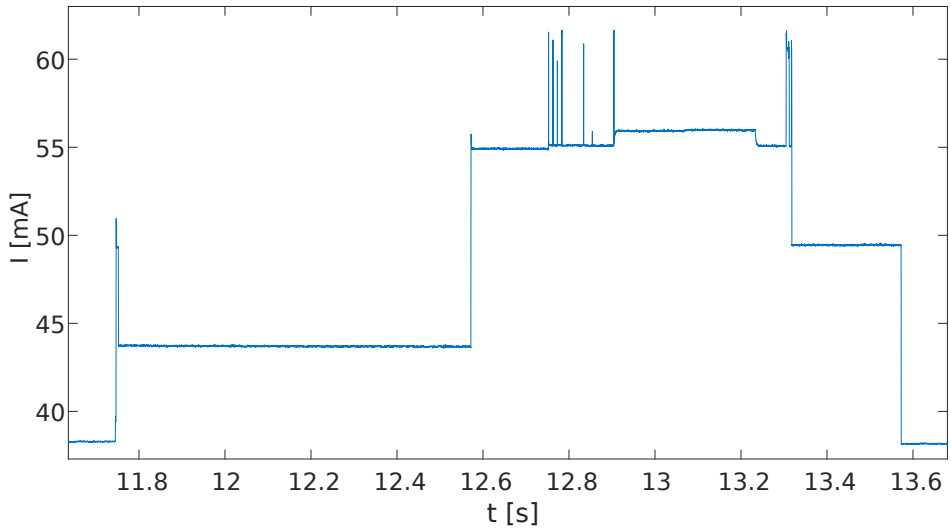


(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

Obr. A.25: RIOT na STM v módu light sleep



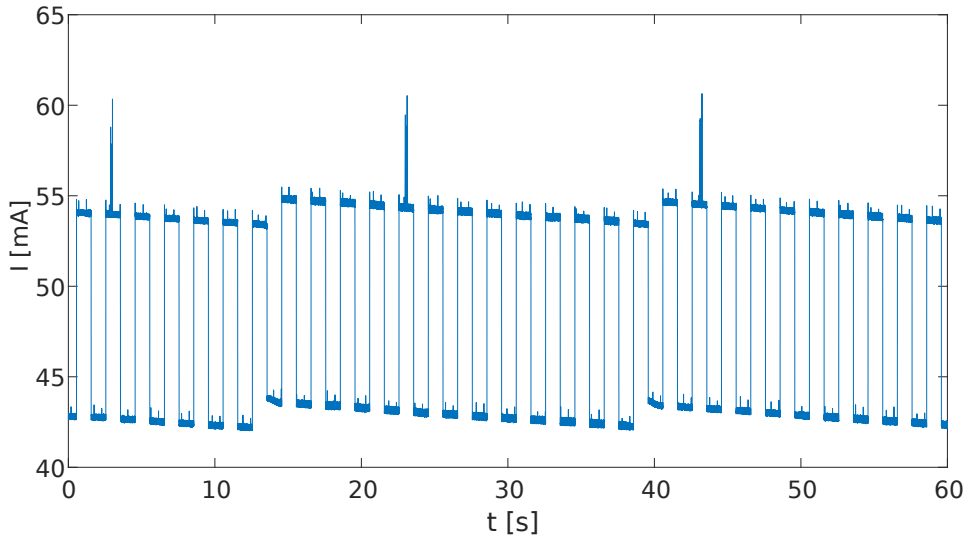
(a) 3 cykly



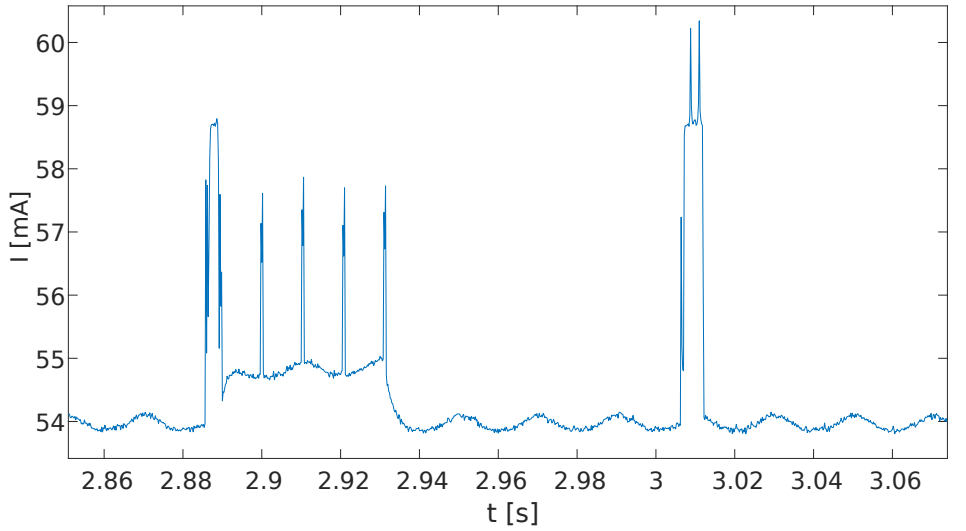
(b) Detail odebíraného proudu při změně módu deep sleep na aktivní a zpět do deep sleep.

Obr. A.26: RIOT na STM v módu deep sleep

A.3.4 Zephyr

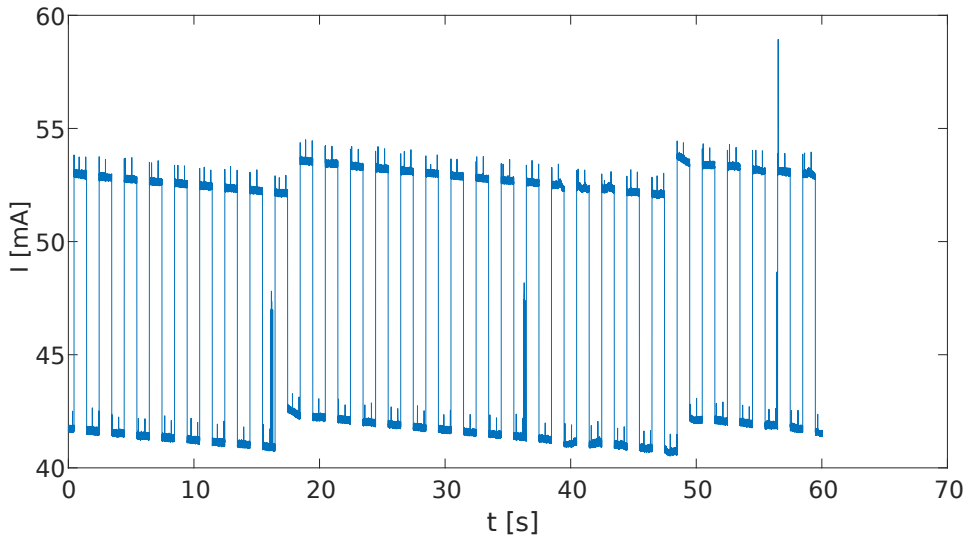


(a) 3 cykly

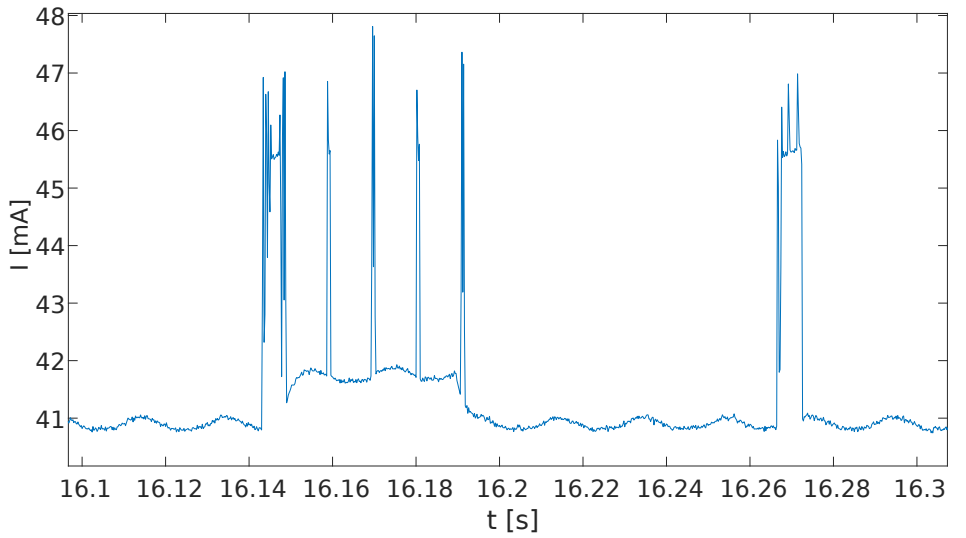


(b) Detail odebíraného proudu při provádění činnosti v aktivním módu.

Obr. A.27: Zephyr na STM bez nízkoodběrového módu



(a) 3 cykly



(b) Detail odebíraného proudu při změně módu light sleep na aktivní a zpět do light sleep.

Obr. A.28: Zephyr na STM v módu light sleep