

Komunikace se senzory po sériových sběrnících

Adam Huspenina



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Adam Huspenina
Osobní číslo: A20713
Studijní program: B0613A140020 Softwarové inženýrství
Forma studia: Kombinovaná
Téma práce: Komunikace se senzory po sériových sběrnících
Téma práce anglicky: Communication with Sensors Over Serial Buses

Zásady pro vypracování

- Vypracujte literární resersi na dané téma.
- Popište principy komunikace po sériových sběrnících SPI a I2C.
- Vytvořte sadu utilit pro komunikaci s vybranými senzory.
- Navrhněte způsob ukládání a vizualizace dat.
- Proveďte srovnání a testy pro různé scénáře použití.
- Zhodnotte výsledky a proveďte diskuzi.

Seznam doporučené literatury:

1. *TMS320x281x Serial Peripheral Interface Reference Guide* [online]. Texas Instruments, INC. 2009 [cit. 2024-10-23]. Dostupné z: <https://www.ti.com/lit/ug/spru059e/spru059e.pdf>
2. *SPI Block Guide V03.06* [online]. Motorola, Inc. 2003 [cit. 2024-10-23]. Dostupné z: <http://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/teare/ee308l/datasheets/S12SPIV3.pdf>
3. *UM10204 I2C-bus specification and user manual* [online]. NXP Semiconductors. 2021 [cit. 2024-10-23]. Dostupné z: <https://web.archive.org/web/20221006073143/http://www.nxp.com/docs/en/user-guide/UM10204.pdf>
4. COOK, Jeremy. *Understanding Sensor Interfaces – UART, I2C, SPI and CAN* [online]. 2024 [cit. 2024-10-23]. Dostupné z: <https://dev.blues.io/blog/blues-university-understanding-sensor-interfaces-uart-i2c-spi-can>
5. TIŠNOVSKÝ, Pavel. *Externí sériové sběrnice SPI a I²C*. Online. Root.cz. 2008 [cit. 2024-10-23]. Dostupné z: <https://www.root.cz/clanky/externi-seriove-sbernice-spi-a-i2c/>.
6. TIŠNOVSKÝ, Pavel. *Komunikace po sériové sběrnici I²C*. Online. Root.cz. 2009 [cit. 2024-10-23]. Dostupné z: <https://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/>.

Vedoucí bakalářské práce:

doc. Ing. Martin Sysel, Ph.D.

Ústav počítačových a komunikačních systémů

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

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



L.S.

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

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

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

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

Prohlašuji, že

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

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Snaha o zvýšení výkonu v moderních embedded systémech často zahrnuje překračování provozních parametrů komunikačních protokolů, jako je I2C a SPI. Tato práce zkoumá integritu přenosu dat přes tyto sběrnice při provozu na zvýšených rychlostech, se zaměřením zejména na fenomén zastarávání nebo přerušení přenosu dat. Byly vytvořeny benchmarkové utility pro systematické čtení dat ze senzorů LSM303D a L3GD20H připojených jednotlivě k Raspberry Pi 3 přes sběrnice SPI a I2C. Zvýšením taktovacích frekvencí sběrnice byly pozorovány kritické prahové hodnoty, při kterých dynamické hodnoty senzorů buď zamrzly, což indikovalo zastavení aktualizací nových dat, nebo se staly zastaralými, což odráželo opakované získávání zastaralých hodnot. Tato zjištění kvantifikují praktická rychlostní omezení těchto rozhraní v kontextu embedded systémů a zdůrazňují důležitost výběru rychlosti sběrnice pro zajištění věrnosti dat ve vysoce výkonných senzorových aplikacích.

Klíčová slova: SPI, I2C, sběrnice, Raspberry Pi

ABSTRACT

The effort for performance enhancement in modern embedded systems often involves exceeding the operating parameters of communication protocols such as I2C and SPI. This work investigates the integrity of data transmission over these buses when operating at increased speeds, with a particular focus on the phenomenon of data aging or interruption. Benchmark utilities were created to systematically read data from a LSM303D and a L3GD20H sensor connected individually to a Raspberry Pi 3 via the SPI and I2C buses. By increasing the bus clock frequencies, critical thresholds were observed at which the dynamic sensor values either froze, indicating a halt in new data, or became stale, reflecting repeated stale values. These findings quantify the practical speed limitations of these interfaces in the context of embedded systems and highlights the importance of bus speed selection to ensure data fidelity in high-performance sensor applications.

Keywords: SPI, I2C, bus, Raspberry Pi

Při tvorbě tohoto textu byl využit nástroj ChatGPT (GPT-4o), a to pro pomoc při sestavení struktury kapitol, ke korektuře gramatiky a úpravě stylistiky.

Contents

ÚVOD	13
I TEORETICKÁ ČÁST	14
1 LITERÁRNÍ REŠERŠE	15
1.1 PRINCIPY KOMUNIKACE	15
1.2 TECHNICKÉ VLASTNOSTI	16
1.3 VÝHODY A NEVÝHODY	16
1.4 VYUŽITÍ	17
1.5 VYHODNOCENÍ	17
2 SPI - SÉRIOVÉ PERIFERNÍ ROZHRANÍ	18
2.1 PRINCIP KOMUNIKACE	18
2.2 FYZICKÁ VRSTVA	19
2.3 FÁZE A POLARITA HODINOVÉHO SIGNÁLU	20
2.4 TYPY PŘENOSU DAT	21
2.5 BAUDRATE	22
2.6 TOPOLOGIE SBĚRNICE	23
3 I²C - INTER-INTEGRATED CIRCUIT	26
3.1 PRINCIP KOMUNIKACE	26
3.2 FYZICKÁ VRSTVA	28
3.3 PŘENOSOVÉ RYCHLOSTI	30
3.4 BAUDRATE	32
3.5 TOPOLOGIE SBĚRNICE	32
II PRAKTICKÁ ČÁST	34
4 KOMUNIKACE PŘES I²C	35
4.1 NASTAVENÍ HARDWARU	35
4.2 SOFTWAREOVÉ PROSTŘEDÍ	37
4.3 KNIHOVNA PIGPIO	39

4.4	NÁSTROJ NUITKA	40
5	BENCHMARK UTILITA PRO I^2C	42
5.1	INICIALIZACE KOMUNIKACE I^2C	42
5.2	INICIALIZACE SENZORU A KONFIGURACE REGISTRŮ	42
5.3	MĚŘICÍ SMYČKA A ČASOVÁNÍ	44
5.4	ČTENÍ DAT ZE SENZORU	45
5.5	ZÁZNAM A VYHODNOCENÍ	46
6	KOMUNIKACE PŘES SPI	48
6.1	NASTAVENÍ HARDWARU	48
6.2	SOFTWAREVÉ PROSTŘEDÍ	49
6.3	KNIHOVNA SPIDEV	51
7	BENCHMARK UTILITA PRO SPI	52
7.1	INICIALIZACE KOMUNIKACE SPI	52
7.2	INICIALIZACE SENZORU A KONFIGURACE REGISTRŮ	53
7.3	MĚŘICÍ SMYČKA A ČASOVÁNÍ	54
7.4	ČTENÍ DAT ZE SENZORU	54
7.5	ZÁZNAM A VYHODNOCENÍ	56
8	NÁVRH UKLÁDÁNÍ DAT A VIZUALIZACE	57
8.1	SBĚR DAT A VZORKOVÁNÍ	57
8.2	FORMÁTY UKLÁDÁNÍ DAT	58
8.3	VIZUALIZACE	59
9	ANALÝZA VÝSLEDKŮ	60
9.1	STAV SENZORU V KLIDU	60
9.2	STAV SENZORU V POHYBU	61
10	ZÁVĚR	64
	SEZNAM POUŽITÉ LITERATURY	65
	SEZNAM OBRÁZKŮ	67
	SEZNAM TABULEK	68

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	69
--	-----------

ÚVOD

V dnešní době se senzory stávají nepostradatelnou součástí moderních technologických systémů. Nacházejí uplatnění v široké škále aplikací, od průmyslové automatizace přes chytrou domácnost až po internet věcí (*IoT*). Efektivní komunikace mezi senzory a řídicími jednotkami je klíčová pro správné fungování těchto systémů, přičemž dvě nejčastěji používané sériové sběrnice pro tento účel jsou *SPI* (Serial Peripheral Interface) a *I²C* (Inter-Integrated Circuit).

Tyto sběrnice umožňují přenos dat mezi mikrokontroléry a periferními zařízeními, jako jsou senzory, paměťová úložiště nebo displeje. Každá z nich má své specifické vlastnosti, které určují jejich vhodnost pro různé typy aplikací. Zatímco *SPI* nabízí vyšší rychlost přenosu dat a jednoduchou implementaci, *I²C* umožňuje propojení více zařízení na jedné sběrnici s minimálním počtem vodičů.

Cílem této práce je analyzovat komunikaci se senzory prostřednictvím *SPI* a *I²C*, především z pohledu shromažďování dat, jejich zpracování a vizualizace. Bude zde popsána teoretická stránka obou sběrnic, jejich výhody a nevýhody, a také praktická implementace sběru dat v konkrétním prostředí. Součástí práce bude také demonstrace způsobů zpracování a vizualizace získaných dat, což je důležité pro jejich následnou analýzu a využití v praxi.

I TEORETICKÁ ČÁST

1 LITERÁRNÍ REŠERŠE

Spolehlivá komunikace mezi integrovanými obvody (*IO* angl. Integrated circuits) je důležitou součástí při návrhu vestavěných a elektronických systémů. Dva z nejběžněji používaných protokolů sériové komunikace pro krátkodosahová propojení na úrovni desek plošných spojů jsou I^2C (angl. Inter-Integrated Circuit) a *SPI* (angl. Serial Peripheral Interface). Ačkoli oba protokoly umožňují komunikaci mezi komponentami, výrazně se liší svou architekturou, principy komunikace a typickými případy použití, což vyžaduje pečlivé zhodnocení při návrhu elektronických systémů. [1][2]

1.1 Principy komunikace

Komunikační sběrnice definují pravidla a fyzické struktury nezbytné pro usnadnění výměny dat, včetně počtu komunikačních linek, směrovosti toku dat a mechanismů koordinace zařízení.

I^2C byla vyvinuta společností Philips Semiconductors (nyní NXP Semiconductors) v roce 1982, aby poskytla jednoduchý a efektivní způsob propojení více integrovaných obvodů pomocí pouhých dvou linek [1]. Využívá sériovou hodinovou linku *SCL* a sériovou datovou linku *SDA*, které obě vyžadují externí pull-up rezistory, které při nečinnosti zvedají dle normy napětí na úroveň logické jedničky [3]. Zařízení komunikují poloduplexním způsobem, přičemž určený master zahajuje komunikaci vydáním spouštěcí podmínky, následované adresou cílového zařízení a instrukcí pro čtení/zápis. Přenos dat je bajtově orientovaný, přičemž každý bajt následuje potvrzovací bit *ACK*. I^2C podporuje více masterů na stejné sběrnici a nabízí vestavěnou arbitráž a protahování hodinového signálu, aby se předešlo konfliktům a efektivně se řídil přístup ke sběrnici. [4]

SPI bylo navrženo společností Motorola, klade důraz na jednoduchost a rychlost. Pracuje s minimálně čtyřmi linkami: sériovým hodinovým signálem *SCLK*, vstupním signálem Master Output Slave Input *MOSI*, výstupním signálem Master Input Slave Output *MISO* a linkou pro výběr podřízeného zařízení *SS* pro každé podřízené zařízení. Komunikace je plně duplexní, což umožňuje simultánní přenos a příjem dat. Na rozdíl od I^2C *SPI* nezahrnuje adresování; místo toho je každé podřízené zařízení individuálně vybráno tak, že se jeho vyhrazená linka *SS* přivede na nízkou úroveň (Analog Devices, n.d.). Tento přístup zjednodušuje komunikaci, ale omezuje škálovatelnost, pokud se nepoužívá další hardware

(například demultiplexery). [2][5]

1.2 Technické vlastnosti

Technické vlastnosti I^2C a SPI významně ovlivňují jejich praktické nasazení a ovlivňují datové rychlosti, složitost hardwarového návrhu a škálovatelnost systému.

I^2C je velmi atraktivní díky nízkému počtu pinů, protože vyžaduje pouze dva vodiče bez ohledu na počet zařízení. Podle společnosti Texas Instruments ([4]) standardní I^2C podporuje datové rychlosti až 100 kb/s, rychlý režim až 400 kb/s a vysokorychlostní režim až 3,4 Mb/s. Jak je však uvedeno ve specifikaci I^2C ([1]), sdílená povaha komunikačních linek přináší problémy s kapacitou sběrnice, což omezuje maximální rychlost a délku kabelu. Konfigurace s open-drain navíc umožňuje systémy s více mastery a odolnost proti chybám, ale vede k pomalejším dobám náběhu a poklesu ve srovnání s push-pull budiči [4].

SPI na druhou stranu nabízí výrazně vyšší rychlosti přenosu dat. Podle společnosti Analog Devices ([5]) se rychlosti sběrnice SPI obvykle pohybují od několika megahertzů do desítek megahertzů, přičemž optimalizované systémy dosahují ještě vyšších rychlostí. Plně duplexní komunikace umožňuje efektivní a simultánní přenos a příjem dat. Elektrické propojení point-to-point v SPI navíc zlepšuje integritu signálu a umožňuje delší přenosové vzdálenosti ve srovnání s I^2C [6]. Každé další podřízené zařízení však vyžaduje samostatnou linku SS, což komplikuje zapojení systému a zvyšuje potřebu dostupných pinů mikrokontroléru [7].

1.3 Výhody a nevýhody

Srovnání silných stránek a omezení sběrnic I^2C a SPI je nezbytné pro výběr vhodného protokolu pro dané řešení.

I^2C vyniká v aplikacích, kde je minimální zapojení zásadní, například v systémech s omezenými I/O zdroji mikrokontroléru. Jeho multi-masterová schopnost, adresovatelná zařízení a vestavěné potvrzovací mechanismy zvyšují jeho flexibilitu a spolehlivost [1]. Jeho nižší maximální datové rychlosti a vyšší komunikační režie v důsledku potřeby adresování a handshakingu však omezují jeho účinnost ve vysokorychlostních aplikacích [4]. Pro udržení stability komunikace při vyšších rychlostech je navíc nutné pečlivé uspořádání a správa kapacity sběrnice.

SPI je preferováno pro vysokopropustnou komunikaci díky jednodušší struktuře protokolu a plně duplexnímu provozu [6]. Nízká režie *SPI* vede k vyšším přenosovým rychlostem, což ho činí ideálním pro datově náročné aplikace, jako je přístup k paměťovým kartám nebo ovládání grafického displeje. Nutnost vyhrazené linky SS pro každé zařízení se nicméně stává úzkým hrdlem škálovatelnosti v komplexních systémech s mnoha periferiemi (Digi-Key Electronics, n.d.). *SPI* také postrádá standardní mechanismy pro detekci chyb, arbitráž více masterů nebo řízení toku, které je v případě potřeby nutné implementovat externě.[3][2]

1.4 Využití

Volba mezi *I²C* a *SPI* často závisí na specifických systémových požadavcích, jako je rychlost, počet zařízení a dostupné piny.

I²C se běžně používá v senzorových sítích, konfiguračních *EEPROM*, teplotních senzorech a hodinách reálného času, kde jsou komunikační rychlosti střední a primárním problémem je jednoduchost zapojení. Jeho schopnost propojit více zařízení přes dva vodiče s minimální režii je obzvláště vhodná pro mobilní zařízení, vestavěné systémy a nízkopříkonové konstrukce.[4]

SPI nachází své primární uplatnění v systémech vyžadujících rychlou výměnu dat, jako je komunikace s SD kartami, *ADC* a *DAC* s vysokým rozlišením a zobrazovací moduly. Díky své plně duplexní schopnosti a vysokým taktovacím frekvencím je *SPI* často preferovanou volbou v aplikacích vyžadujících přenos velkých objemů nebo časově kritických dat. [5]

1.5 Vyhodnocení

Ačkoli *I²C* i *SPI* slouží k umožnění komunikace mezi integrovanými obvody, jsou optimalizovány pro různé systémové priority. *I²C* je nejvhodnější pro systémy vyžadující minimální zapojení a více nízkorychlostních periferií, zatímco *SPI* je výhodné v aplikacích vyžadujících vysokorychlostní, plně duplexní přenosy dat s jednoduššími komunikačními protokoly. Výběr vhodného protokolu vyžaduje pečlivou analýzu specifických konstrukčních požadavků, včetně rychlosti, složitosti, škálovatelnosti a nákladů. [1][2][4][6]

2 SPI - SÉRIOVÉ PERIFERNÍ ROZHRAŇÍ

Použití *SPI* (angl. *Serial peripheral interface*) bylo zavedeno společností Motorola, ale dnes je de facto standardem s mnoha implementacemi. Proto je důležité odkazovat na datasheet použitého čipu, aby byla zajištěna správnost detailů. Co se týče původu, *SPI* se poprvé objevilo pravděpodobně v roce 1983, nicméně existují diskuse, že bylo ve skutečnosti představeno již v roce 1979. Bez ohledu na přesné datum vzniku má *SPI* více než 40 letou historii přenosu dat a je dnes široce využíváno. [8]

Sériové periferní rozhraní je vysokorychlostní synchronní sériový vstup/výstup (*I/O*), který umožňuje sériový bitový tok naprogramované délky o rozsahu jeden až šestnáct bitů, který má být posunut do a nebo ze zařízení naprogramovanou přenosovou rychlostí bitů. *SPI* se obecně používá pro komunikaci mezi *DSP* kontrolerem a externím periferním zařízením nebo jiným ovladačem. Typické aplikace zahrnují externí *I/O* nebo periferie rozšíření pomocí zařízení, jako jsou posuvné registry, ovladače displeje a analogově-digitální převodníky *ADC*. [6]

2.1 Princip komunikace

Sběrnice *SPI* představuje jednu z forem sériových externích sběrnic sloužících pro vzájemné propojení dvou či více komunikujících uzlů. Každý *SPI* systém se skládá z jednoho *master* uzlu (tzv. řadiče sběrnice) a jednoho resp. více *slave* uzlů, kde *master* je definován jako mikropočítač, který obsahuje generátor *SPI* hodinového signálu rozváděného do ostatních uzlů, a *slave* je jakýkoli integrovaný obvod (*IC*), který přijímá *SPI* hodinový signál od *master* uzlu. Je možné mít systém, kde může být více jak jeden *IC* v roli *master*, ale v daném okamžiku může být *master* pouze jeden. [9][7]

V nejjednodušším případě obsahují uzly dva registry (viz. 2.1. Prvním je datový záchytný registr (Serial Input Buffer – *SSPBUF*). Do posuvného registru (Shift Register – *SSPSR*) je zapsán bajt (celá osmice bitů), který byl korektně přijat, ale ještě nebyl zpracován, tj. mikrořadič si jeho obsah nepřčetl. Tento registr tedy slouží jako jednoprvková fronta zabezpečující, že při korektní obsluze nedojde ke ztrátě dat. Posuvný registr *SSPSR* slouží současně k vysílání i příjmu jednoho bitu z celé osmice přenášených bitů – každý posun obsahu tohoto registru doprava znamená, že se vysunutý bit pošle na pin *SDO* (v případě uzlu typu *master* se jedná o vodič *MOSI*) a naopak logická hodnota přečtená na pinu *SDI* (tedy

vodič *MISO*) je zapsána do nejnižšího bitu posuvného registru. [7] Během přenosu *SPI* jsou data přenášena (sériově posunuta) a přijímána (sériově posunuta) současně. Sériové hodiny *SCK* synchronizují posun a vzorkování informací na dvou sériových datových vodičích. [2]

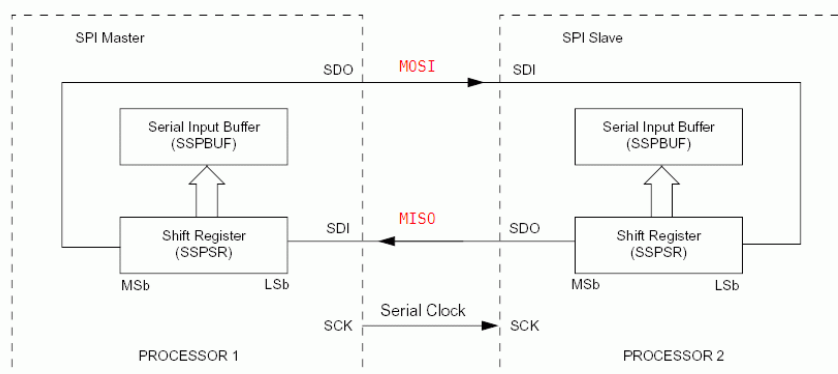


Figure 2.1 Znázornění přenosu dat mezi *master* (vlevo) a *slave* (vpravo) pomocí posuvných registrů.

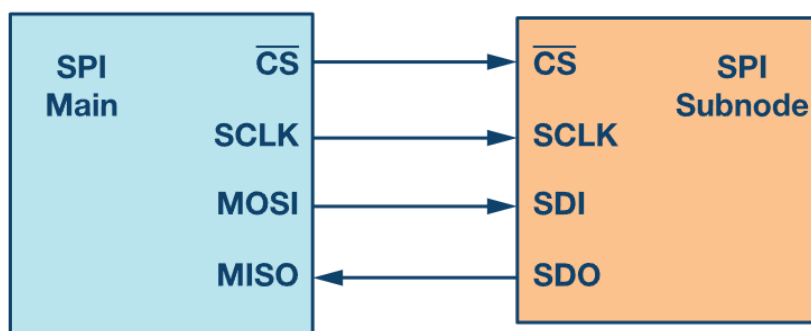
(zdroj: 2)

2.2 Fyzická vrstva

SPI používá k přenosu dat celkem čtyři vodiče (viz. obr. 4.1). Zároveň se jedná o poměrně vysokorychlostní protokol v dvoumístném rozsahu Mbps. Nabízí také plně duplexní provoz mezi jedním *master* a pouze jedním *emphslave* zařízením současně. [8] Typy vodičů jsou následující:

- *MOSI - Master-out/Slave-in* Tento pin se používá k přenosu dat z modulu *SPI*, když je nakonfigurován jako *master*, a k příjmu dat když je nakonfigurován jako *slave*.
- *MISO - Master-in/Slave-out* Tento pin se používá k přenosu dat z modulu *SPI*, když je nakonfigurován jako *slave*, a k příjmu dat když je nakonfigurován jako *master*.
- *CS/SS - Cable select/ Slave select* Tento pin se používá pro výstup selektivního signálu z modulu *SPI* na další periférii, se kterou jsou data přenos se má uskutečnit, když je nakonfigurován jako *master* a je použit jako vstup pro příjem volby *slave* signál, když je *SPI* nakonfigurováno jako *slave*. *CS* je držen na vysokém napětí, aby se odpojil *slave* od sběrnice, a snížen *masterem* pro aktivaci konkrétního *slave*, (tj. active low).
- *SCLK - SPI Clock* Tento pin se používá pro výstup hodin, s ohledem na které *SPI* přenáší data nebo v případě přijímá hodiny *slave*.

[2]

Figure 2.2 Základní schéma zapojení zařízení *SPI*.

(zdroj: 5)

Hlavním prvkem systému *SPI* je registr dat *SPI*. 8-bitový datový registr v *masteru* a 8-bitový datový registr v *slave* jsou propojeny piny *MOSI* a *MISO* a tvoří distribuovaný 16-bitový registr. Když se provádí operace přenosu dat, tento 16-bitový registr je sériově posunut o osm bitových pozic pomocí hodinového signálu z *masteru*, takže dochází k výměně dat mezi *masterem* a *slave*. Data zapsaná do hlavního *SPI* datového registru se stanou výstupními daty pro *slave* a data načtená z hlavního *SPI* datového registru po operaci přenosu jsou vstupními daty z *slave*. [2]

Čtení *SPI* posuvného registru s *SPTEF* nastaveným na logickou jedničku následované zápisem do *SPI* data registru vloží data do registru přenášených dat. Po dokončení přenosu se přijatá data přesunou do registru přijímaných dat. Data lze číst z tohoto systému s dvojitou vyrovnávací pamětí kdykoli před dokončením dalšího přenosu. Tento 8-bitový datový registr funguje jako registr *SPI* pro příjem dat pro čtení a jako registr pro přenos dat *SPI* pro zápis. Jediná adresa registru *SPI* se používá pro čtení dat z vyrovnávací paměti načtených dat a pro zápis dat do registru vysílaných dat.

2.3 Fáze a polarita hodinového signálu

Pomocí kombinace dvou bitů v řídicím registru *SPI* software vybere jednu ze čtyř kombinací fáze a polarity hodinového signálu. Bit pro řízení polarity taktu *CPOL* určuje aktivitu při vysoké nebo nízké hladině hodinového signálu a nemá na ně žádný významný vliv formát přenosu. Bit pro řízení fáze hodinového signálu *CPHA* vybírá jeden ze dvou zásadně odlišných přenosových formátů. Fáze a polarita hodinového signálu by měly být identické

pro hlavní *SPI* zařízení a komunikující *slave* zařízení. V některých případech se fáze a polarita mezi přenosy mění, aby bylo umožněno hlavnímu zařízení komunikovat s periferními *slave* jednotkami s různými požadavky. [2]

- *CPOL - Clock polarity* Pokud je bit polarity hodinového signálu vymazán a data se nepřenášejí, na pinu *SCK* na *master* zařízení se vytvoří nízká hodnota v ustáleném stavu. Naopak, pokud je tento bit nastaven, bude pin *SCK* nečinný. Tento bit se také používá ve spojení s řídicím bitem fáze hodinového signálu pro vytvoření požadovaného vztahu hodinových dat mezi *master* a *slave* zařízením. [9]
- *CPHA - Clock phase* Bit určující fázi hodinového signálu ve spojení s bitem *CPOL* řídí vztah mezi signálem a přenášenými daty mezi *master* a *slave*. Bit *CPOL* si lze představit jako jednoduché vložení měniče do série s vedením *SCK*. Bit *CPHA* vybírá jeden ze dvou zásadně odlišných protokolů taktování. Když je *CPHA* rovna 0, fáze signálu jsou logický *OR SCK* s *SS*. Jakmile *SS* klesne na nízkou úroveň, transakce začne a první hrana *SCK* zahrnuje první vzorek dat. Když je *CPHA* rovna 1, *SS* pin lze považovat za jednoduché ovládání aktivace výstupu. [9]

2.4 Typy přenosu dat

Typ přenosu dat představuje způsob jakým jsou data přenášena s ohledem na generování hodinového signálu. Polarita a fáze hodinového signálu jsou důležité pro datové režimy. Podle polarity signálu lze pochopit úroveň signálu, která může být v klidovém stavu nízká a bude začínat vzestupnou hranou při přenosu dat, nebo může být vysoká v klidovém stavu a začne klesat při výměně dat. V závislosti na fázi hodinového signálu jsou data generována nebo vzorkována s ohledem na signál v kanálu (tj. na vzestupné nebo sestupné hraně) jak je možné vidět na obrázku níže. [10]

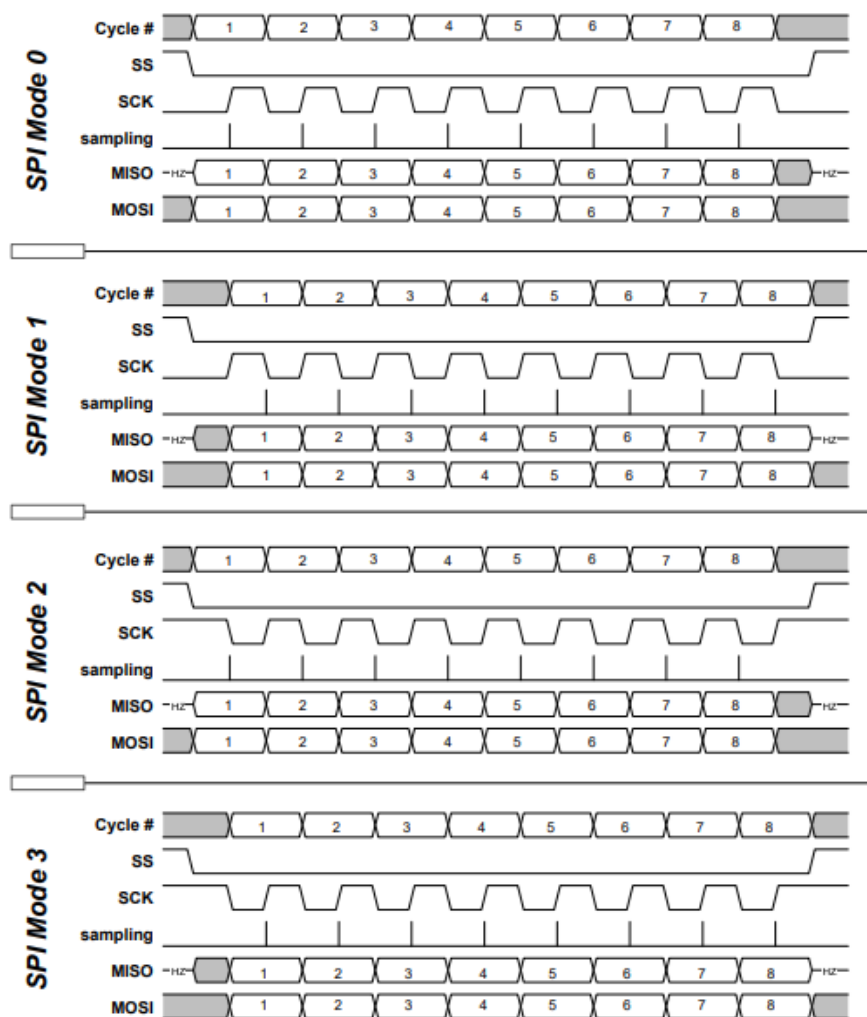


Figure 2.3 Módy přenosů dat v *SPI*. Jak *master*, tak *slave* zařízení musí být nakonfigurováno stejným způsobem, aby jedno mohlo správně dekodovat to, co zakódovalo druhé.

(zdroj: 10)

2.5 Baudrate

Modul *SPI* podporuje 125 různých přenosových rychlostí (*Baudrate*) a čtyři různá schémata časování. V závislosti na tom, zda jsou hodiny *SPI* v režimu *slave* nebo *master*, může pin *SCLK* přijímat externí hodinový signál *SPI* nebo poskytovat hodinový signál *SPI*. [6]

- V režimu *slave* jsou hodiny *SPI* přijímány na pin *SPICLK* z externího zdroje a nemohou být větší než frekvence *LSPCLK* dělená 4.
- V hlavním režimu jsou hodiny *SPI* generovány *SPI* a jsou vyvedeny na pin *SPICLK*

a nemohou být větší než frekvence $LSPCLK$ dělená čtyřmi.

Pro výpočet hodnoty SPI Baudrate, kdy registr nabývá hodnot mezi 3 až 127:

$$SPIBaudrate = LSPCLK / (SPIBRR + 1)$$

Pro výpočet hodnoty SPI Baudrate, kdy registr nabývá hodnot mezi 0, 1 nebo 2:

$$SPIBaudrate = LSPCLK / 4$$

2.6 Topologie sběrnice

Sběrnice SPI je založena na principu komunikace *master - slave* (viz. 2.2) kdy musí vždy existovat jeden uzel v roli *master* a jeden či více uzlů v roli *slave*. Následně budou popsány možnosti zapojení jednotlivých variant:

Master - Slave

Jedná se o základní zapojení sběrnice. V tomto případě je připojeno jen jedno zařízení.



Figure 2.4 Schéma zapojení jednoho zařízení na sběrnici SPI . V tomto případě je signál CS volitelný a není potřeba.

(zdroj: 11)

Master - Slaves (CS)

V tomto schématu je každé *slave* zařízení ovládáno pomocí jeho vodiče pro výběr čipu (CS - *Chip select*). Když je SS (viz. 2.2) deaktivován, výstup dat ze *slave* zařízení přejde do stavu vysoké impedance, takže neinterferuje s aktuálně vybraným *slave* zařízením a vstup dat *slave* zařízení je ignorován.

Výhodou tohoto zapojení je možnost ovládat každé zařízení zvlášť oproti metodě *daisy chain*. Tento přístup umožňuje připojení SPI zařízení, která vyžadují různá schémata časování (viz. 2.5).

Pokud *slave* zařízení vyžaduje nastavení různých časování (tzv. hrany/stavy nečinnosti), je možné přeprogramovat hlavní *SPI* hardwarový modul před povolením konkrétního *CS* tak, aby do něj každé *slave* zařízení mělo zasílané správné signály.

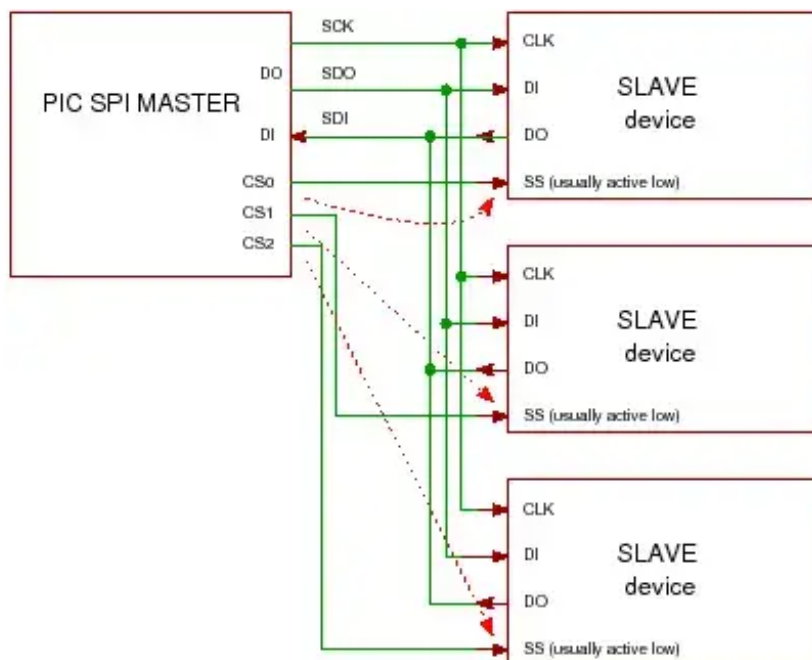


Figure 2.5 Schéma zapojení více zařízení na sběrnici *SPI* pomocí *CS*. Červené šipky znázorňují kontrolní vodiče.

(zdroj: 11)

Master - Slaves (Daisy chain)

V tomto schématu jsou všechna data odesílaná *master* zařízením přesunuta do všech zařízení a všechna data odeslaná z každého zařízení jsou posunuta na další (viz. obr. 2.6). Při použití metody *Daisy chain*, musí být zaručeno, že všechny *slave* zařízení používají stejnou hranu hodin a stav nečinných hodin a musí být získán správný počet bitů, což vede k vyšší sofwarové náročnosti. Výhodou metody *Daisy chain* je ukládání signálu CS pro každé podřízené *SPI* zařízení.[11]

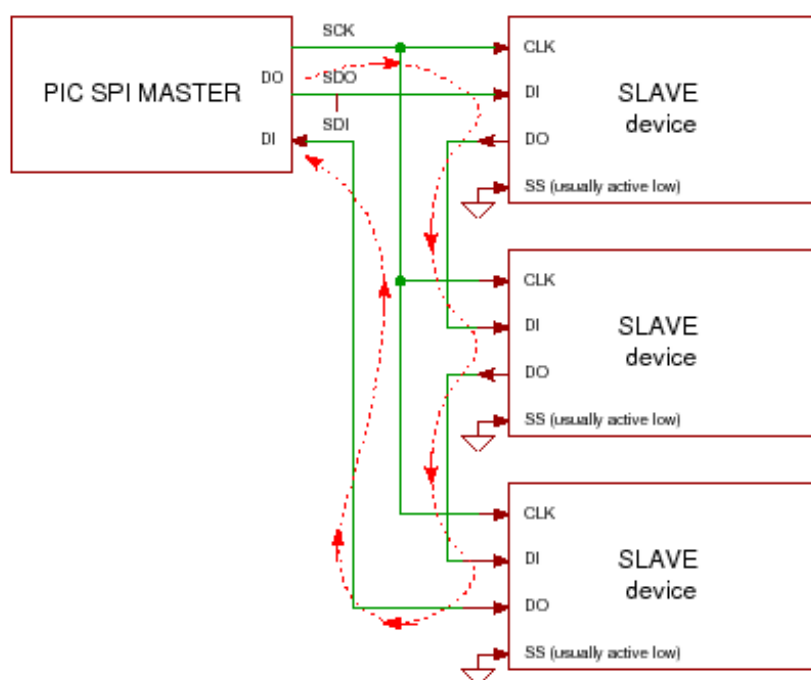


Figure 2.6 Schéma zapojení více zařízení na sběrnici *SPI* pomocí *Daisy chain*. Červené šipky znázorňují kontrolní vodiče.

(zdroj: 11)

3 I^2C - INTER-INTEGRATED CIRCUIT

Sběrnice I^2C (Inter-Integrated Circuit) byla představena v roce 1982 společností Philips Semiconductor a nyní je volně dostupná pro kohokoli, kdo ji může používat a dokonce ji může dle potřeby upravovat [1]. Původní specifikace umožňovala komunikaci pouze na 100 kHz a poskytovala pouze 7-bitové adresy, čímž omezovala počet zařízení na sběrnici na 112 (existuje několik rezervovaných adres, které nebudou nikdy použity pro platné adresy I^2C). V roce 1992 byla publikována první veřejná specifikace, která přidala 400kHz rychlý režim a také rozšířený 10-bitový adresní prostor. Kromě originální sběrnice I^2C představil Intel v roce 1995 variantu s názvem *SMBus* - System Management Bus. *SMBus* je přísněji kontrolovaný formát, jehož cílem je maximalizovat předvídatelnost komunikace mezi podpůrnými integrovanými obvody na základních deskách počítačů. [12]

Jedná se o sériový komunikační protokol široce používaný pro připojení nízkorychlostních periférií k mikrokontrolérům v embedded systémech. Je navržena pro efektivní, nenáročnou komunikaci na krátkou vzdálenost mezi více integrovanými obvody s využitím pouze dvou obousměrných linek. Tato jednoduchost v kombinaci s možností připojení více periferních zařízení na stejné sběrnici přispěla k širokému přijetí I^2C v průmyslových, spotřebitelských a automobilových aplikacích. [1]

Základní filozofie I^2C spočívá v modelu *master-slave* a arbitraci sdílené sběrnice. I když nabízí relativně nízké datové rychlosti, kompenzuje to minimálními požadavky na zapojení a umožňuje přímočaré adresování zařízení. I^2C tak dosahuje rovnováhy mezi jednoduchostí designu a flexibilitou komunikace, což z něj činí nezbytnou součást moderního návrhu embedded systémů. [4]

3.1 Princip komunikace

Sběrnice I^2C je standardní obousměrné rozhraní, které ve své podstatě funguje jako synchronní komunikační protokol s více *master*y a více *slave* zařízeními. Přenos dat je zajištěn pomocí dvojice linek, která je tvořena sériovou hodinovou linkou *SCL* a sériovou datovou linkou *SDA*. Hlavní zařízení, které zahajuje komunikaci, je zodpovědné za generování hodinového signálu, zatímco podřízená zařízení reagují podle svých jedinečných adres. [1]

Slave zařízení tedy nesmí přenášet data, pokud nebyla adresována *masterem*. Každé

zařízení na sběrnici I^2C má specifickou adresu zařízení, která rozlišuje mezi ostatními zařízeními, která jsou na stejné sběrnici I^2C . Mnoho *slave* zařízení vyžaduje konfiguraci při spuštění, aby se nastavilo jejich chování. To se obvykle provádí, když *master* přistupuje k interním mapám registrů *slave* zařízení, které mají jedinečné adresy registrů. Zařízení může mít jeden nebo více registrů, kde se data ukládají, zapisují nebo čtou. [4]

Komunikace začíná podmínkou START, která je definována přechodem z vysokého na nízký na *SDA*, zatímco *SCL* zůstává vysoký. Následuje 7-bitový nebo 10-bitový adresní rámec, bit pro čtení/zápis a sekvence 8-bitových datových rámců. Po každém datovém bajtu následuje bit potvrzení *ACK* nebo nepotvrzení *NACK*, který indikuje stav komunikace. Komunikace je ukončena podmínkou STOP, která je označena přechodem z nízké hladiny na vysokou na *SDA*, zatímco *SCL* je na vysoké hladině.

Detekce chyb je nedílnou součástí protokolu prostřednictvím mechanismu *ACK/NACK* a arbitráže, umožňuje koexistenci více *master*ů porovnáváním bitů přenášených na *SDA* s pozorovanými bity. Pokud dojde ke konfliktu, *master*, který detekuje nesrovnalost, ukončí komunikaci. Dále je povoleno prodloužení hodinového signálu, kdy *slave* zařízení může udržovat linku *SCL* na nízké úrovni, pokud vyžaduje delší čas ke zpracování dat, čímž efektivně pozastaví komunikaci. [1]

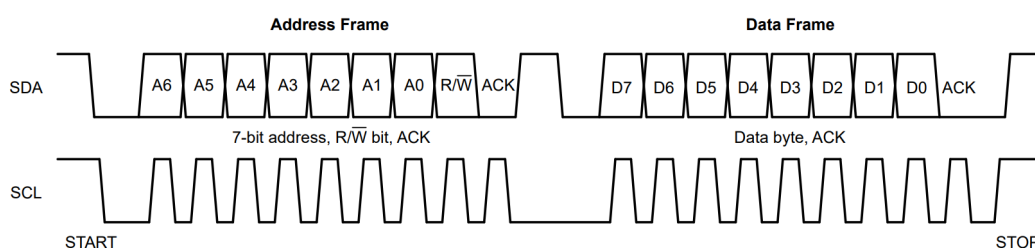


Figure 3.1 Schéma transakce I^2C zobrazující START, bit adresy a R/W, *ACK/NACK*, datové rámce a podmínku STOP.

(zdroj: 4)

Po vydání podmínky START odešle *master* v prvním bajtu adresu zamýšleného *slave* zařízení. Tento adresní rámec se skládá z [13]:

- 7 nejvýznamnějších bitů (nebo prvních 10 bitů v rozšířeném režimu) identifikujících adresu *slave* zařízení.

- Nejméně významného bitu (bit R/W), který určuje směr přenosu dat. A to pro 0: *master* bude zapisovat do podřízeného zařízení nebo 1: *master* bude číst ze *slave* zařízení.

Adresované *slave* zařízení musí potvrdit příjem snížením signálu na linku *SDA* během hodinového impulsu *ACK*. Pokud žádné zařízení neodpoví potvrzením *ACK*, hlavní zařízení dojde k závěru, že adresa je neplatná nebo že zařízení nereaguje. Tento mechanismus zajišťuje, že na hlavní zařízení reaguje v daném okamžiku pouze jedno zařízení, a tvoří základ pro komunikaci mezi více zařízeními.

Kromě samotného postupu přenosu dat popsaného výše používá I^2C specifický adresovací mechanismus, který umožňuje selektivní komunikaci se *slave* zařízeními. Každé *slave* zařízení připojené ke sběrnici I^2C musí mít jedinečnou adresu. Existují dva formáty [3]:

- 7-bitové adresování (nejběžnější): Umožňuje až 112 použitelných adres ze 128 kombinací, přičemž některé jsou rezervovány pro speciální účely.
- 10-bitové adresování: Zavedeno později pro rozšíření počtu adresovatelných zařízení. Používá speciální prefix následovaný dvěma bajty pro adresování.

3.2 Fyzická vrstva

Fyzická vrstva sběrnice I^2C je definována svou jednoduchostí a efektivitou v komunikaci na krátkou vzdálenost, v rámci desky nebo mezi čipy. Spoléhá na dvou vodičovou konfiguraci složenou ze sériové datové linky *SDA* a sériové hodinové linky *SCL*. Obě linky jsou obousměrné a sdílené mezi všemi zařízeními na sběrnici. *SDA* i *SCL* jsou linky s open-drain (nebo open-collector). Zařízení mohou linku stahovat pouze do stavu LOW; k dosažení stavu HIGH se používají externí pull-up rezistory. [1]

Správné dimenzování pull-up rezistorů je klíčové a závisí na kapacitě sběrnice a požadované komunikační rychlosti. Hodnoty odporu se obvykle pohybují od 1 k Ω do 10 k Ω . Nižší odpor zajišťuje rychlejší doby náběhu, vhodné pro vysokorychlostní provoz, zatímco vyšší odpor snižuje spotřebu energie a je přijatelný pro nízkorychlostní aplikace. [4]

Celková kapacita sběrnice, určená délkou vodičů a počtem připojených zařízení, musí zůstat v mezích specifikací (obvykle pod 400 pF), aby se zachovala integrita signálu.

Nadměrná kapacita vede ke zhoršeným dobám náběhu a poklesu, což zvyšuje riziko komunikačních chyb. [12]

Fyzickou vrstvu sběrnice I^2C představují následující typy vodičů:

- *SDA* (Serial Data Line) přenáší data a adresy mezi *masterem* a *slave*. Musí zůstat stabilní během doby HIGH linky hodin *SCL* přechody jsou povoleny pouze tehdy, když je *SCL* LOW. [1]
- *SCL* (Serial Clock Line) poskytuje hodinový signál generovaný *masterem* pro synchronizaci komunikace. Má také otevřený odtok a potřebuje pull-up rezistor. *Slave* zařízení mohou dočasně udržovat tuto linku v nízkém stavu LOW, aby zpozdila komunikaci (roztahování hodin). [4]

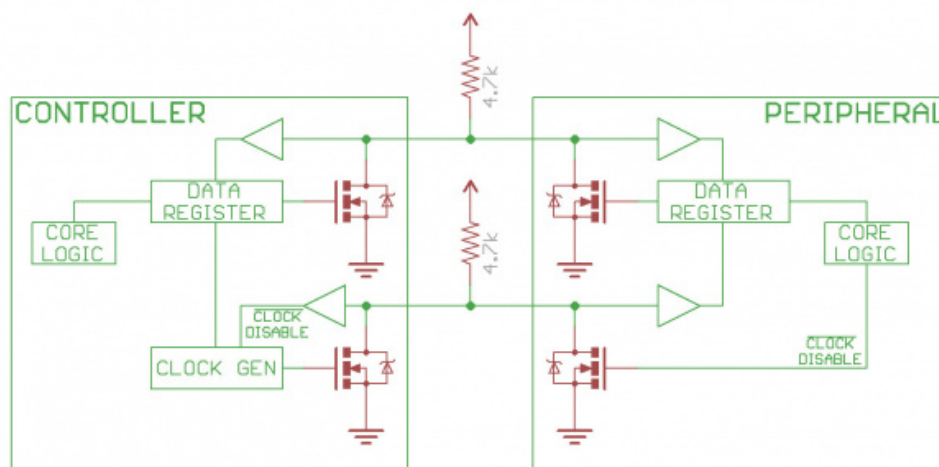


Figure 3.2 Schéma zapojení I^2C zobrazující fyzické zapojení jednoho periferního zařízení s využitím pull-up rezistorů.

(zdroj: 12)

Výše zmíněné vlastnosti zajišťují bezpečný provoz více zařízení, protože signalizace s open-drain zabraňuje soupeření sběrnice, i když se více zařízení pokouší řídit stejnou linku. Kompromisem pro tuto jednoduchost je pomalejší sběrnice ve srovnání s diferenciálními nebo push-pull systémy, díky čemuž je I^2C nejvhodnější pro nízkorychlostní komunikaci na krátkou vzdálenost mezi integrovanými obvody na stejné desce nebo přes krátké kabely.

3.3 Přenosové rychlosti

Protokol sběrnice I^2C se opírá o dobře definovaný časový vztah mezi sériovou datovou linkou SDA a sériovou hodinovou linkou SCL . Komunikace I^2C je ve své jádru synchronní, přičemž hlavní zařízení generuje hodinový signál SCL a tím definuje časování pro přenos dat. Polarita signálů SDA i SCL je aktivní nízká, což znamená, že logická nízká úroveň (0) je reprezentována sníženou úrovní napětí (obvykle blízko 0 V), zatímco logická vysoká úroveň (1) je definována pasivním vysokým napětím (obvykle 3,3 V nebo 5 V) udržovaným pull-up rezistory. [1]

Protokol I^2C definuje specifické řídicí podmínky, které signalizují začátek a konec komunikace. Tyto podmínky jsou rozpoznávány všemi zařízeními na sběrnici a musí striktně dodržovat pravidla časování signálu [1]:

- Stav START je indikován sestupnou hranou na SDA , když je SCL na úrovni HIGH. To signalizuje, že *master* zařízení má v úmyslu zahájit komunikaci. Jakmile je detekován stav START, všechna ostatní zařízení monitorují sběrnici a zjišťují informace o adresování.
- Stav STOP je indikován vzestupnou hranou na SDA , když je SCL na úrovni HIGH. To označuje konec komunikační sekvence, uvolňuje sběrnici a vrací ji do klidového stavu (jak SDA , tak SCL na úrovni HIGH).

Tyto podmínky jsou kódovány v časování signálu, nikoli v konkrétních datových bajtech, takže přesná interpretace vyžaduje, aby zařízení nepřetržitě monitorovala obě linky. Každý režim má svou vlastní sadu minimálních period pro vysoké nebo nízké úrovně hodnoty hodin a doby nastavení nebo udržení dat. Různé rychlostní režimy ukládají přísnější časová omezení [3]:

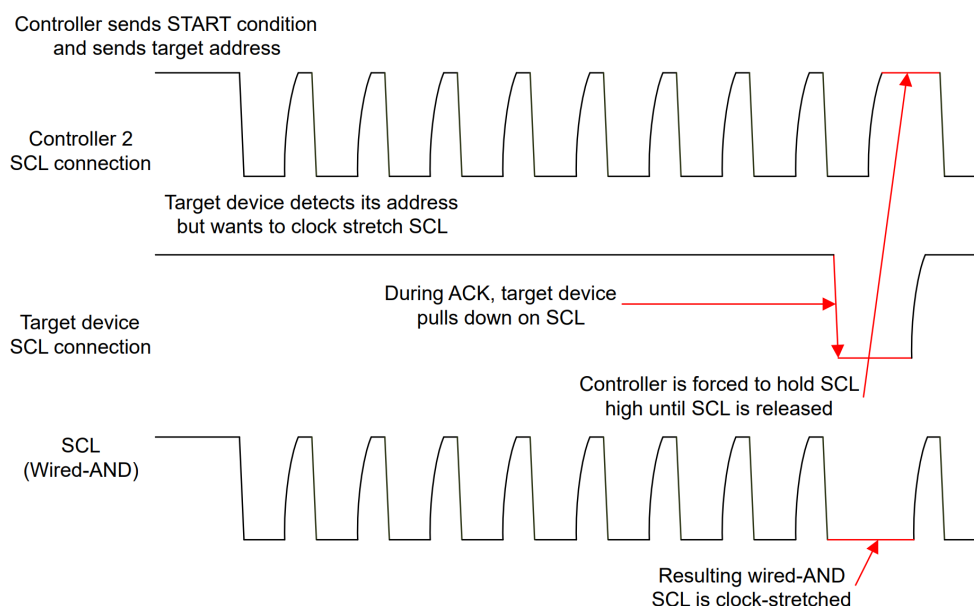
Table 3.1 Režimy rychlostí sběrnice I^2C .

Označení	Přenosová rychlost
Standardní režim	$\leq 100 \text{ kbit/s}$
Rychlý režim	$\leq 400 \text{ kbit/s}$
Rychlý režim Plus	$\leq 1 \text{ Mbit/s}$
Vysokorychlostní režim	$\leq 3,4 \text{ Mbit/s}$

(zdroj: [3])

Protahování hodinového impulsu

Protahování hodinového impulsu je funkce protokolu I^2C , která umožňuje *slave* zařízením zpoždit komunikaci tím, že drží linku *SCL* na nízké úrovni poté, co ji hlavní zařízení uvolní. To je obzvláště užitečné, když podřízené zařízení potřebuje více času na zpracování dat nebo přípravu odpovědi. [4]

Figure 3.3 Diagram průběhu komunikace na sběrnici I^2C se znázorněním protažení hodinového signálu k přenosu dat.

(zdroj: [3])

Během protahování hodinového impulsu podřízené zařízení aktivně stahuje linku *SCL* na nízkou úroveň po náběžné hraně předchozího hodinového impulsu. *master* musí monitorovat

SCL a čekat, až se uvolní (tj. vrátí se na vysokou úroveň), než vygeneruje další hodinový impuls. Tento mechanismus umožňuje asynchronní koordinaci mezi zařízeními s různými výpočetními schopnostmi, aniž by došlo k porušení pravidel časování protokolu. [13]

Ne všechny implementace hlavních zařízení I^2C podporují protahování hodinového impulsu a některé rychlé nebo vysokorychlostní návrhy jej mohou z důvodu výkonu zakázat. V embedded aplikacích, zejména při komunikaci se senzory nebo EEPROM, protahování hodinového impulsu zajišťuje zachování integrity dat i při různých latencích zařízení. [4]

3.4 Baudrate

Přenosová rychlost v I^2C se vztahuje k rychlosti přenosu dat a je do značné míry určena *master* zařízením, které řídí linku *SCL*. I^2C podporuje několik standardních režimů přenosové rychlosti, od 100 kbit/s ve standardním režimu až po 3,4 Mbit/s ve vysokorychlostním režimu. Výběr přenosové rychlosti ovlivňuje nejen rychlost přenosu dat, ale také přípustné hodnoty kapacity sběrnice a pull-up rezistoru. [1]

Skutečná dosažená přenosová rychlost v praxi závisí na době náběhu linek, která je ovlivněna hodnotami rezistorů a kapacit. Aby bylo spolehlivě dosaženo požadované přenosové rychlosti, musí konstruktéři zajistit, aby časová konstanta RC sběrnice umožňovala linkám ustálit se v rámci přiděleného bitového času. [12] V hlavních sběrniciích jsou hodiny generovány s pevnými časovými intervaly, ale ve *slave* zařízeních, zejména během protahování hodin, se efektivní přenosová rychlost může dynamicky měnit. [1]

3.5 Topologie sběrnice

Přestože I^2C využívá logickou strukturu sběrnice, její fyzické topologie se mohou lišit v závislosti na umístění zařízení a návrhu systému. Všechna zařízení sdílejí stejné dva vodiče *SDA* a *SCL*. Topologie musí zachovat správnou integritu signálu a časovací výkon. Typické topologie zahrnují [1]:

- Lineární (řetězec): Jednoduché zapojení zařízení za sebou.
- Hvězdicová topologie (nedoporučuje se): Může způsobit odrazy signálu a časové neshody.
- Vícebodová: Kde je připojeno více *master* a *slave* zařízení. Arbitráž zajišťuje, že

sběrnici v daném okamžiku ovládá pouze jeden *master*.

Počet připojených zařízení je omezen adresním prostorem (zejména u 7-bitového adresování) a celkovou kapacitou sběrnice. Ve větších systémech lze použít opakovače nebo multiplexory I^2C k izolaci kapacity a rozšíření rozsahu adresování. [1] Zvláštní aspekty platí v systémech se smíšeným napětím. Pokud některá zařízení pracují na různých logických úrovních (např. 3,3 V a 5 V), jsou potřebné posunovače úrovní. Komunikace na dlouhé vzdálenosti navíc vyžaduje aktivní ukládání do vyrovnávací paměti nebo přepnutí na robustnější fyzické vrstvy (např. SMBus nebo PMBus). [12]

II PRAKTICKÁ ČÁST

4 KOMUNIKACE PŘES I^2C

Moderní vestavěné systémy často vyžadují sběr dat v reálném čase z různých senzorů. Sběrnice I^2C (Inter-Integrated Circuit) je široce používaný komunikační protokol díky své jednoduchosti, nízkému počtu pinů a možnosti připojení více zařízení na stejnou sběrnici. Tato kapitola popisuje komplexní nastavení propojení se dvěma specifickými senzory I^2C tříosým gyroskopem L3GD20H a 3D akcelerometrem a magnetometrem LSM303D, s využitím Raspberry Pi 3 Model B. Dále představuje utility Pythonu pro interakci s těmito senzory prostřednictvím knihovny pigpio a zkoumá použití nástroje Nuitka ke kompilaci skriptů Pythonu do optimalizovaných spustitelných souborů. Cílem kapitoly je poskytnout praktický návod od zapojení a nastavení hardwaru až po spuštění utilit.

4.1 Nastavení hardwaru

V této kapitole jsou popsány veškeré potřebné hardwarové komponenty, které byly použity pro komunikaci na sběrnici I^2C , a také způsob zapojení mezi deskou Raspberry Pi a nepájivým prostředím.

Použité komponenty

Pro replikaci experimentů popsaných v této kapitole je vyžadován následující hardware:

- Raspberry Pi 3 Model B s funkční instalací operačního systému Raspbian
- Gyroskopický senzor L3GD20H (verze kompatibilní s I^2C)
- Akcelerometr a magnetometr LSM303D (verze kompatibilní s I^2C)
- Nepájivé pole a propojovací vodiče pro prototypování
- Stabilní napájecí zdroj 5V 2,5A

Zapojení obvodů

Každý senzor se připojuje k Raspberry Pi přes sběrnici I^2C , což umožňuje více zařízením sdílet dvě komunikační linky SDA (data) a SCL (hodiny). L3GD20H[14] a LSM303D[15] mají továrně nastavené adresy I^2C , které nekolidují (např. 0x6B a 0x1D), což umožňuje jejich případné současné použití. Zapojení by mělo být provedeno pečlivě, aby se zabránilo

poškození součástek. Schémata znázorňující fyzická připojení by měla obsahovat označené piny *GPIO*, konektory senzorů a barevně odlišené vodiče. Zapojení je popsáno v následující tabulce:

Table 4.1 Zapojení senzoru k desce.

Pin senzoru	Pin <i>GPIO</i>	Funkce
VIN	Pin 1/2 (3,3 V, 5 V)	Napájení (3,3 V nebo 5 V)
GND	Pin 6 (<i>GND</i>)	Zem
SDA	Pin 3 (<i>GPIO2</i>)	I^2C Data
SCL	Pin 5 (<i>GPIO3</i>)	I^2C Hodiny

(zdroj: vlastní)

Při zapojení je důležité zajistit, aby napájení odpovídalo požadavkům na napětí senzoru. L3GD20H i LSM303D podporují provoz minimálně na 2,5 V a maximálně na 5,5 V, což je v souladu s logickými úrovněmi *GPIO* Raspberry Pi. Využití piny je možné dohledat v následujícím diagramu:

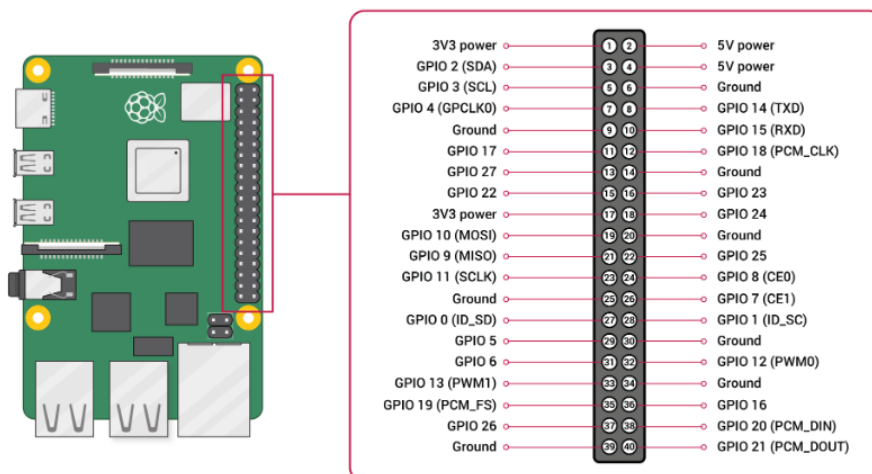


Figure 4.1 Schéma *GPIO* pro Raspberry PI 3 model B.

(zdroj: 16)

4.2 Softwareové prostředí

Před komunikací se senzory musí být Raspberry Pi nakonfigurováno tak, aby povolil a používal rozhraní I^2C .

Povolení I^2C

Systém Raspbian má ve výchozím nastavení sběrnici I^2C zakázanou. Pro povolení je nutné použít nástroj `raspi-config`, který je dostupný v grafickém rozhraní v nastavení a nebo v případě absence grafického rozhraní je jej možné spustit z terminálu následovně:

```
1 sudo raspi-config
```

Výpis 4.1 Příkaz ke spuštění `raspi-config` (zdroj: vlastní)

Zde v sekci „Interfacing Options“ stačí vybrat položku obsahující název sběrnice I^2C , která rozhraní aktivuje. Následně je potřeba provést restart systému a moduly jádra I^2C by měly být automaticky načteny. To lze ověřit kontrolou přítomnosti souboru `/dev/i2c-1`.

Alternativou je úprava přímo konfiguračního souboru `/boot/config.txt`, ve kterém je potřeba odkomentovat řádek začínající řetězcem `"dtparam=i2c_arm=on"`. K editaci souboru lze využít předinstalovaný textový editor `nano`. Provádění změn vyžaduje privilegovaný přístup.

```
1 sudo nano /boot/config.txt
```

Výpis 4.2 Příkaz k editaci konfiguračního souboru systému. (zdroj: vlastní)

Tento soubor také slouží k nastavení rychlosti sběrnice (viz. 3.3). Ve výchozí konfiguraci není explicitně parametr upravující rychlost uváděn. Hodnota rychlosti je uváděna v HZ a jeho výchozí hodnota odpovídá rychlosti *Standard mode*, tedy 100000. Dle rychlosti, kterou podporuje připojený senzor, je možné upravit hodnotu na 400000 (*Fast mode*) nebo případně 3400000 (*Ultra-fast mode*). Konfigurační řádek pro sběrnici I^2C by měl po úpravě vypadat následovně:

```
1 dtparam=i2c_arm=on,i2c_arm_baudrate=3400000
```

Výpis 4.3 Konfigurace sběrnice I^2C nastavující její rychlost na *Ultra-fast mode*. (zdroj: vlastní)

Ověření připojení zařízení

Naskenování sběrnice I^2C a nalezení připojených zařízení je nutné k ověření, zda je zařízení správně připojeno a také ke zjištění adresy, s kterou bude sběrnice komunikovat. Výstup by měl ve výchozím stavu obsahovat adresu 0x6B pro L3GD20H a 0x1D pro LSM303D. Pro naskenování sběrnice slouží následující příkaz:

```
1 i2cdetect -y 1
```

Výpis 4.4 Příkaz k naskenování sběrnice I^2C (zdroj: vlastní)

Tyto adresy mohou být změněny dvěma způsoby. Prvním je přidání dalšího vodiče propojujícího pin *SDO* senzoru s pinem *GND* nebo *VCC*. Toto zapojení zaručí trvalou změnu adresy. Tento přístup zvýší resp. sníží adresu senzoru podle pinu, ke kterému bylo *SDO* připojeno (viz. 4.2).

Table 4.2 Příklad změny adresy senzoru při přivedení *GND* či *VCC* na pin senzoru *SDO*.

Pin <i>GPIO</i>	Původní adresa	Nová adresa
Pin 1/2 (3,3 V, 5 V)	0x1D	0x1C
Pin 6 (<i>GND</i>)	0x1D	0x1E

(zdroj: vlastní)

Druhou možností je softwarově zapsat přímo do registru senzoru využitím funkce *write_byte_data()* a tím přepsat jeho adresu. Tato změna není trvalá a je vrácena na výchozí hodnotu po ztrátě napětí. Ne všechny senzory podporují tento přístup.

```
1 import smbus
2 bus = smbus.SMBus(1)
3 default_addr = 0x1D
4 new_addr = 0x1E
5 bus.write_byte_data(default_addr, 0x8A, new_addr << 1)
```

Výpis 4.5 Příklad softwarové změny adresy senzoru připojeného na sběrnici I^2C . (zdroj: vlastní)

Instalace závislostí

Pro podporu komunikace a správy systému I^2C založené na Pythonu je nutné nainstalovat následující balíčky:

```
1 sudo apt update
2 sudo apt install -y python3-pip python3-dev pigpio i2c-tools
3 pip3 install pigpio
```

Výpis 4.6 Příkazy k instalaci balíčků (zdroj: vlastní)

Tyto příkazy instalují daemona pigpio a klientskou knihovnu, stejně jako i2c-tools pro nízkourovňové ladění a ověřování.

4.3 Knihovna pigpio

Knihovna pigpio je navržena pro nízkourovňové řízení *GPIO* na Raspberry Pi. Je obzvláště užitečná pro I^2C aplikace vyžadující přesné časování nebo souběžný přístup z více vláken či procesů. Na rozdíl od standardního smbus nebo *RPi.GPIO* funguje pigpio jako daemon spuštěný na pozadí. Tato architektura se vyhýbá blokování procesů a zjednodušuje vícevláknové aplikace. Podporuje bit-banged a hardwarové I^2C režimy a je vhodná pro prototypování a dotazování senzorů v reálném čase.

Knihovna pigpio používá model klient-server. Skript se připojuje k daemonu pigpio a používá metody specifické pro I^2C , jako například *i2c_open*, *i2c_write_byte_data* a *i2c_read_byte_data*, k interakci se zařízeními. Po dokončení sběru dat se spojení explicitně ukončí.

Knihovna pigpio vyžaduje, aby její daemon pigpiod běžel na pozadí. Daemon umožňuje skriptům přístup k rozhraní *GPIO* a I^2C s přesným načasováním a souběžností bezpečnou pro vlákna. Spuštění daemona lze dosáhnout následujícím příkazem:

```
1 sudo pigpiod
```

Výpis 4.7 Příkaz ke spuštění pigpiod daemonu (zdroj: vlastní)

4.4 Nástroj Nuitka

Python je interpretovaný jazyk a není vždy vhodný pro nasazení v produkčním prostředí, zejména v prostředí s omezenými zdroji nebo citlivými na bezpečnost. Nuitka je kompilátor Pythonu, který převádí skripty Pythonu do samostatných spustitelných souborů. Na rozdíl od interpretů nebo balíčků bajtkódu Nuitka překládá kód Pythonu do C a poté jej kompiluje do optimalizovaného binárního kódu. To nabízí několik výhod jako ochranu zdrojového kódu, běh bez nutnosti prostředí Pythonu, ale především vyšší rychlost exekuce kódu.

Instalace

Nejjednodušší možností jak nainstalovat nástroj Nuitka je pomocí manažera balíčků pro Python *pip*. Instalace Nuitka a požadovaných nástrojů pro sestavení:

```
1 pip3 install nuitka
2 sudo apt install build-essential
```

Výpis 4.8 Příkaz k instalaci nástrojů pro nuitka (zdroj: vlastní)

Kompilace a spuštění

Kompilace se provádí pomocí příkazu *nuitka* s argumentem *--follow-imports*, který rekurzivně zkompiluje hlavní soubor včetně jeho závislostí do jednoho souboru. Další volby mohou doladit výkon a diagnostický výstup. Finální spustitelný soubor se nachází v podadresáři pojmenovaném podle původního kompilovaného skriptu. Výsledný spustitelný soubor obsahuje všechny potřebné závislosti. Utilitu Pythonu lze zkompilovat následovně:

```
1 nuitka3 --follow-imports <nazev_utility>.py
```

Výpis 4.9 Příkaz ke kompilaci utility nástrojem nuitka (zdroj: vlastní)

Po kompilaci lze binární soubor spustit přímo z příkazového řádku. Chová se identicky

jako původní skript v Pythonu, ale má rychlejší spouštění a provádění. Při spuštění je nezbytně nutné, aby byl daemon pigpio aktivní na pozadí, protože kompilovaný binární soubor stále závisí na rozhraní pigpio.

5 BENCHMARK UTILITA PRO I^2C

Tato kapitola popisuje návrh benchmarkovacího nástroje pro senzory I^2C na Raspberry Pi. Utilita je určena k měření efektivní rychlosti, s jakou senzor poskytuje nová data, a toho, jak se tato rychlost chová při různých frekvencích dotazování. I když se senzory použité v experimentech liší z hlediska adresního prostoru a inicializačních registrů, struktura nástroje zůstává napříč moduly stejná. Tato utilita poskytuje výsledky díky nimž je možné pousoudit reálné limity výkonu senzorů připojených ke sběrnici I^2C na Raspberry Pi 3.

Utilita postupně navyšuje frekvenci dotazování senzoru procházením předem definované sady frekvencí. Na každé z nich zaznamenává celkový počet přečtených vzorků, počet detekovaných změn a časové charakteristiky. Výsledky poskytují informace o komunikační latenci i načasování dostupnosti dat senzoru.

5.1 Inicializace komunikace I^2C

Pro komunikaci se zařízeními I^2C v Pythonu na Raspberry Pi se používá knihovna `pigpio`. Tento přístup umožňuje přesné řízení I^2C na bitové úrovni, což je důležité pro konzistenci v aplikacích kritických pro načasování. Komunikace začíná spuštěním daemona `pigpio` a připojením k němu pomocí rozhraní `pigpio.pi()`.

```
1 pi = pigpio.pi()
2 handle = pi.i2c_open(1, 0x1d)
```

Výpis 5.1 Příklad připojení k daemonu `pigpio` s následným otevřením komunikace s daným senzorem. (zdroj: vlastní)

- `pi` reprezentuje objekt, který představuje rozhraní daemona `pigpio`.
- `pi.i2c_open(1, 0x1d)` zahájí komunikaci na sběrnici I^2C (typicky hodnota 1 na Raspberry Pi) se zařízením na adrese 0x1d (v tomto případě LSM303D).
- `handle` (reprezentován celým číslem) je použit pro budoucí transakce I^2C se zařízením.

5.2 Inicializace senzoru a konfigurace registrů

Před čtením dat ze senzoru je nutné jej nakonfigurovat do aktivního provozního režimu. To se provádí zápisem specifických bitových vzorů do konfiguračních registrů, které se liší podle

typu senzoru. Tyto registry obvykle řídí, které osy jsou aktivní, rychlost výstupních dat *ODR* a režim konverze.

Akcelerometr LSM303D

Akcelerometr LSM303D vyžaduje povolení subsystému akcelerometru a nastavení jeho výstupní datové rychlosti pomocí řídicího registru. Konfigurace používá bitovou masku k povolení všech os a výběru nejvyšší dostupné datové rychlosti. Inicializace je možná provést následovně:

```
1 pi.i2c_write_byte_data(h, 0x20, 0b01110111)
2 pi.i2c_write_byte_data(h, 0x21, 0b00000000)
```

Výpis 5.2 Příklad nastavení řídicích registrů pro konfiguraci akcelerometru. (zdroj: vlastní)

- Registr 0x20: hodnota 0b01110111 povoluje osy X, Y, Z a nastavuje *ODR* na 800 Hz.
- Registr 0x21: nastavuje filtrování a škálování blokových dat.

Magnetometr LSM303D

Magnetometr se nachází na stejném zařízení LSM303D, avšak je inicializován samostatně. Magnetometr je aktivován prostřednictvím samostatného řídicího registru. Typické nastavení pro aktivaci režimu kontinuálního převodu a výstupu s vysokým rozlišením může být inicializováno následovně:

```
1 pi.i2c_write_byte_data(handle, 0x24, 0b1111000)
2 pi.i2c_write_byte_data(handle, 0x26, 0b00000000)
```

Výpis 5.3 Příklad nastavení řídicích registrů pro konfiguraci magnetometru. (zdroj: vlastní)

- Registr 0x24: nastavuje režim s vysokým rozlišením a nastavuje magnetometr *ODR* na 100 Hz.
- Registr 0x26: nastavuje režim kontinuální konverze.

Gyroskop L3GD20H

Gyroskop L3GD20H se inicializuje prostřednictvím bitové masky na kontrolním registru. Typické nastavení s povolením všech tří os a nastavením výstupní frekvence na 760 Hz je možné provést následovně:

```
1 pi.i2c_write_byte_data(handle, 0x20, 0b11111111)
2 pi.i2c_write_byte_data(handle, 0x23, 0b00000000)
```

Výpis 5.4 Příklad nastavení řídicích registrů pro konfiguraci gyroskopu. (zdroj: vlastní)

- Registr 0x20: Povoluje napájení senzoru a nastavuje čtení všech tří os.
- Registr 0x23: Filtry a rozsah ponechává ve výchozím nastavení.

5.3 Měřicí smyčka a časování

Pro porovnání odezvy senzoru napříč různými frekvencemi se používá předdefinovaný seznam dotazovacích frekvencí. Na každé frekvenci se ze senzoru načte pevný počet vzorků a zaznamená se počet změn hodnot. Cílem je určit, jak často senzor dodává nová data při různých dotazovacích frekvencích.

Každý cyklus čtení musí dodržovat konzistentní interval, aby se dosáhlo požadované vzorkovací frekvence. Protože standardní funkce *time.sleep()* v Pythonu nemusí poskytovat dostatečné rozlišení nebo konzistenci, používá se pro časování s vysokým rozlišením *time.perf_counter()*. Interval pro každou frekvenci se vypočítá jako převrácená hodnota cílové frekvence. Hlavní smyčka v těchto intervalech dotazuje senzor a zahrnuje horní časový limit *timeout*, aby se zabránilo zablokování, pokud zařízení nereaguje nebo pokud systémové zpoždění překročí předpokládaný časový limit.

```
1 interval = 1.0 / frequency
2 next_time = time.perf_counter()
```

Výpis 5.5 Příklad inicializace časování pro měřicí smyčku. (zdroj: vlastní)

- *interval* nastavuje požadované zpoždění mezi po sobě jdoucími čteními, vypočítané jako inverzní hodnota cílové frekvence.

- *next_time* uchovává čas, ve kterém by se mělo provést další čtení.

Funkce *time.perf_counter()* vrací hodnotu s plovoucí desetinnou čárkou představující čas v sekundách s rozlišením submikrosekund. Je monotónní, což ji činí vhodnou pro časové intervaly. Uvnitř měřicí smyčky může být časování řízeno následovně:

```
1 now = time.perf_counter()
2 diff = next_time - now
3 if diff > 0:
4     time.sleep(diff)
5 next_time += interval
```

Výpis 5.6 Příklad řízení časování uvnitř měřicí smyčky. (zdroj: vlastní)

- *now* ukládá aktuální čas.
- *diff* představuje zbývajícím čas do plánovaného čtení. Pokud je kladný, program se přepne do aktivního čekání, aby se přizpůsobil nadcházejícímu čtení.
- *next_time* je zvýšen o interval pro naplánování dalšího čtení.

Tento přístup udržuje konzistentní tempo dotazování senzoru a umožňuje po spuštění vyhodnocení, zda senzor při každém pokusu odpověděl novými daty.

5.4 Čtení dat ze senzoru

Každý vzorek je čten v jednom blokovém přenosu I^2C se zapnutým příznakem automatického inkrementace, který umožňuje čtení více sousedních datových registrů. U senzorů ukládajících 16-bitová celá čísla se znaménkem v po sobě jdoucích registrech program čte hodnoty pomocí bitových operací. Data ze senzorů jsou přístupná prostřednictvím čtení bloků I^2C počínaje základním registrem. LSM303D umožňuje čtení více bajtů s automatickým zvyšováním adresy, když je nastaven nejvyšší bit adresy registru. Při čtení je použita bitová konverze:

```
1 x = (data[1] << 8) | data[0]
2 y = (data[3] << 8) | data[2]
3 z = (data[5] << 8) | data[4]
```

Výpis 5.7 Příklad bitové konverze. (zdroj: vlastní)

- data jsou ve formátu little-endian; nižší bajt je první.
- Bitové posuny a operace OR rekonstruuji 16-bitová celá čísla se znaménkem ze dvou 8-bitových hodnot.

Akcelerometr LSM303D

Čtení dat zahrnuje přístup k registrům počínaje od adresy 0x28 s nastaveným bitem automatického inkrementace (tj. *MSB* adresy registru):

```
1 data = pi.i2c_read_i2c_block_data(handle, 0x28 | 0x80, 6)
```

Výpis 5.8 Příklad čtení dat z akcelerometru. (zdroj: vlastní)

Magnetometr LSM303D

Datové registry pro magnetometr začínají na adrese 0x08. Podobně jako u akcelerometru se načítá 6 bajtů s povoleným automatickým přírůstkem:

```
1 data = pi.i2c_read_i2c_block_data(handle, 0x08 | 0x80, 6)
```

Výpis 5.9 Příklad čtení dat z magnetometru. (zdroj: vlastní)

Gyroskop L3GD20H

Data gyroskopu se načítají z registrů začínajících na adrese 0x28, podobně jako u akcelerometru LSM303D. Proces čtení a převodu je co do formy identický, ale může se lišit v zarovnání os a škálování citlivosti.

5.5 Záznam a vyhodnocení

Každý frekvenční cyklus zaznamenává počet načtených vzorků, počet vzorků, které se změnily vzhledem k předchozímu naměřenému údaji, průměrný čas mezi jednotlivými naměřenými údaji a dosaženou frekvenci. Tyto hodnoty se zapisují do souboru CSV, aby bylo možné je následně zpracovat nebo vizualizovat. Protokol se aktualizuje po každém cyklu a položky jsou formátovány jako:

- Požadovaná frekvence

- Počet měřených vzorků
- Počet lišících se vzorků
- Doba měření
- Dosažená frekvence

To umožňuje uživateli odvodit, na kterých frekvencích senzor nadále poskytuje nová data a v jakém bodě čtecí smyčka začíná převzorkovat zastaralá data.

6 KOMUNIKACE PŘES *SPI*

Sběrnice *SPI* je běžně používaný protokol pro vysokorychlostní, plně duplexní komunikaci mezi mikrokontrolérem a periferními zařízeními. Na rozdíl od *I²C*, které podporuje adresování více zařízení na sdílené dvou vodičové sběrnici, *SPI* komunikuje přes čtyři vyhrazené linky a pro správu více podřízených zařízení se spoléhá na signály pro výběr jednotlivých čipů. To může vést ke složitějšímu zapojení, ale má za následek výrazně vyšší datovou propustnost a jednodušší logiku protokolu.

V kontextu komunikace senzorů se *SPI* často volí pro aplikace, které vyžadují přesné načasování, minimální latenci nebo rychlý přenos dat, které jsou obzvláště výhodné při čtení dat souvisejících s pohybem ze senzorů. Tato kapitola popisuje kompletní návod na nastavení, propojení a komunikaci se dvěma takovými senzory, gyroskopem L3GD20H a akcelerometrem-magnetometrem LSM303D, přes *SPI* pomocí Pythonu, s využitím Raspberry Pi 3 Model B.

Na rozdíl od verze založené na *I²C*, která byla představena dříve, utility pro sběrnici *SPI* využívají pro komunikaci nativní knihovnu *spidev*. Knihovna *pigpio*, která byla použita pro nízkoúrovňový časově citlivý přístup k *I²C*, zde není nutná, protože ovladač Linux *SPI* již poskytuje stabilní a výkonný přístup prostřednictvím rozhraní `/dev/spidev<bus>.<device>`. Navíc jsou utility zkompileovány pomocí nástroje *nuitka*, aby byla zvýšena efektivita provádění a flexibilita nasazení.

6.1 Nastavení hardwaru

V této kapitole jsou popsány veškeré potřebné hardwarové komponenty, které byly použity pro komunikaci na sběrnici *SPI*, a také způsob zapojení mezi deskou Raspberry Pi a nepajivým prostředím.

Použité komponenty

Pro replikaci experimentů popsaných v této kapitole je vyžadován následující hardware:

- Raspberry Pi 3 Model B s funkční instalací operačního systému Raspbian
- Gyroskopický senzor L3GD20H (verze kompatibilní s *SPI*)
- Akcelerometr a magnetometr LSM303D (verze kompatibilní s *SPI*)

- Nepájivá pole a propojovací vodiče pro prototypování
- Stabilní napájecí zdroj 5V 2,5A

Zapojení obvodů

Spojení přes sběrnici *SPI* vyžaduje čtyři signální vodiče. Vodič *MOSI* (Master Out Slave In), který slouží k odesílání dat z Raspberry Pi do senzoru. Vodič *MISO* (Master In Slave Out) přijímá data vrácená ze senzoru do Raspberry Pi. Dále *SCLK* (Serial Clock), který přenáší signál hodin generovaný Raspberry Pi. A nakonec *CS* (Chip Select) vedoucí signál používaný k povolení komunikace s konkrétním senzorem. Raspberry podporuje *SPI* na svém *GPIO* konektoru prostřednictvím vyhrazených pinů:

Table 6.1 Zapojení senzoru k desce.

Pin senzoru	Pin <i>GPIO</i>	Funkce
<i>VIN</i>	Pin 1/2 (3,3 V / 5 V)	Napájení (3,3 V nebo 5 V)
<i>GND</i>	Pin 6 (GND)	Zem
<i>SA0</i>	Pin 21 (GPIO21)	<i>MISO</i>
<i>SDA</i>	Pin 19 (GPIO19)	<i>MOSI</i>
<i>SCL</i>	Pin 23 (GPIO23)	<i>SPI</i> Hodiny
<i>CS</i>	Pin 24 (GPIO24)	<i>CS</i>

(zdroj: vlastní)

Každý senzor musí být připojen k vlastnímu pinu pro výběr čipu *CS*, pokud mají být používány současně. V tomto případě je připojen pouze jeden senzor, aby se předešlo konfliktům adres. Napájení je dodáváno přes 3,3 V (pin 1 nebo 17) a *GND* (např. pin 6). *SPI* režim senzoru se aktivuje změnou hladiny pinu *SDO/SA0* na nízkou nebo vysokou úroveň dle příslušných pokynů v datasheetu.

6.2 Softwarové prostředí

Před komunikací se senzory musí být Raspberry Pi nakonfigurováno tak, aby povolil a používal rozhraní *SPI*. Pro komunikaci přes sběrnici *SPI* není vyžadována instalace žádných speciálních

systémových balíčků.

Povolení *SPI*

Stejně jako v případě sběrnice *I²C* systém Raspbian má ve výchozím nastavení sběrnici *SPI* zakázanou. Pro povolení je nutné použít nástroj `raspi-config`, který je dostupný jak v grafickém rozhraní v nastavení tak v případě absence grafického rozhraní je možné jej spustit z terminálu (viz. 4.2).

Zde v sekci „Interfacing Options“ stačí vybrat položku obsahující název sběrnice *SPI*, která danou sběrnici povolí. Tím se načtou moduly jádra *SPI* a zpřístupní se adresáře `/dev/spidev0.0` pro *CS0* a `/dev/spidev0.1` pro *CS1*. Po provedení této změny může být nutný restart systému.

Obdobně jako u sběrnice *I²C* je možné použít alternativní přístup a upravit konfigurační soubor `/boot/config.txt`. V tomto případě je ovšem nutné odkomentovat řádek obsahující následující řetězec:

```
1 dtparam=spi_arm=on
```

Výpis 6.1 Příklad části konfiguračního souboru povolující sběrnici *SPI*. (zdroj: vlastní)

Ověření připojení zařízení

Ověřit, zda je sběrnice *SPI* skutečně povolena, je možné zjištěním zpřístupnění adresářů `/dev/spidev<bus>.<device>`. K ověření lze například využít příkaz k vylistování adresáře:

```
1 ls -l /dev/spidev*
```

Výpis 6.2 Příkaz k ověření povolení sběrnice *SPI*. (zdroj: vlastní)

Instalace závislostí

Pro podporu komunikace a správy systému *SPI* založené na Pythonu je nutné nainstalovat následující balíčky (společné nástroje a knihovny pro *I²C* jsou popsány výše):

```
1 sudo apt update
2 sudo apt install -y python3-pip python3-dev build-essential
3 pip3 install spidev
4 pip3 install nuitka
```

Výpis 6.3 Příkazy k instalaci balíčků (zdroj: vlastní)

6.3 Knihovna spidev

Knihovna *spidev* nabízí nízkoúrovňové funkce pro přenos dat přes *SPI*, včetně podpory pro nastavení taktovací frekvence, výběru režimu *SPI* (polarita a fáze) a provádění plně duplexní výměny dat. Zatímco *pigpio* byl využit pro komunikaci *I²C* díky své schopnosti poskytovat přesné softwarové časování, komunikace *SPI* v Linuxu těží z vestavěné podpory jádra. Rozhraní */dev/spidev<bus>.<device>* umožňují uživatelským aplikacím komunikovat se zařízeními *SPI* bez nutnosti dalších knihoven. To znamená, že *spidev* může spolehlivě pracovat při vysokých rychlostech s nízkou latencí a bez problémů s časováním, které mohou ovlivňovat přepínání bitů na sběrnici *I²C*.

V tomto kontextu je *pigpio* pro sběrnici *SPI* proto zbytečný. Navíc knihovna *spidev* nabízí přímou a detailní kontrolu nad sběrnicí *SPI*, což je ideální pro rychlé prototypování i pro utility na produkční úrovni.

Objekt *spidev* je inicializován číslem sběrnice a číslem zařízení odpovídajícím */dev/spidev<bus>.<device>*. Přenosy se provádějí pomocí metod *xfer2()* nebo *readbytes()* a *writebytes()*. Komunikace často zahrnuje zápis příkazového bajtu následovaného fiktivními bajty pro načtení odpovědi z podřízeného zařízení. Čtení a zápisy registrů musí respektovat protokol senzoru ve většině případů nastavení *MSB* na 1 indikuje čtení a nastavení bitu 6 umožňuje automatické zvyšování při čtení více registrů.

7 BENCHMARK UTILITA PRO SPI

Tato benchmarkingová utilita byla vyvinuta za účelem vyhodnocení komunikačního výkonu sběrnice *SPI* mezi Raspberry Pi 3 Model B a senzory L3GD20H a LSM303D. Utilita je navržena tak, aby posoudila výkon *SPI* komunikace v celém rozsahu frekvencí sběrnice tím, že co nejrychleji zachytí pevný počet po sobě jdoucích hodnot ze senzorů. Návrh vyvažuje abstrakci a detaily specifické pro hardware, čímž zajišťuje reprodukovatelnost a adaptabilitu na podobné senzory.

Pro každou frekvenci *SPI* shromažďuje utilita pevný počet vzorků ze senzorů tak rychle, jak to hardware umožňuje. Utilita zaznamenává každý vzorek s přiřazeným časovým razítkem, vypočítává intervaly vzorkování, odhaduje efektivní vzorkovací frekvenci a sleduje počet unikátních hodnot pro vyhodnocení variability signálu. Všechna měření se zapisují do strukturovaných výstupních souborů CSV pro další zpracování a vizualizaci.

7.1 Inicializace komunikace SPI

Komunikace přes *SPI* na Raspberry Pi je řešena modulem `spidev`, který poskytuje nízkoúrovňové rozhraní k hardwarovému řadiči *SPI*. Toto nastavení používá výchozí sběrnici *SPI* a konfiguruje režim *SPI*, což odpovídá požadavkům senzorů. Rychlost hodin *SPI* `spi.max_speed_hz` se nastavuje dynamicky před každou iterací testu, což umožňuje nástroji porovnat více frekvencí v jednom spuštění. Sběrnice může být inicializována následovně:

```
1 spi = spidev.SpiDev()  
2 spi.open(SPI_BUS, SPI_DEVICE)  
3 spi.mode = 0b00
```

Výpis 7.1 Příklad inicializace komunikace na sběrnici *SPI*. (zdroj: vlastní)

- `spi` reprezentuje objekt sloužící jako komunikační rozhraní na sběrnici.
- `SPI_BUS` reprezentuje výchozí sběrnici *SPI* (typicky 0).
- `SPI_DEVICE` reprezentuje výchozí zařízení připojené na sběrnici *SPI* (typicky 0).
- `spi.mode` nastavuje režim sběrnice *SPI* na 0b00 (polarita hodin = 0, fáze hodin = 0).

Každá *SPI* transakce se provádí pomocí *spi.xfer2()*, která odesílá a přijímá bajty současně. To umožňuje jak zápis do registrů senzorů, tak i čtení z nich pomocí plně duplexních přenosů.

7.2 Inicializace senzoru a konfigurace registrů

Každý senzor musí být před zahájením měření inicializován zápisem příslušných řídicích hodnot do jeho interních registrů přes *SPI*. Proces inicializace zajišťuje správný provozní režim, škálování výstupu a formát dat. Hodnoty registrů se odesílají pomocí transakcí zápisu *SPI*.

Akcelerometr LSM303D

Pro povolení části senzoru LSM303D obsahující akcelerometr je provedeno následujícím nastavením registrů:

```
1 spi.xfer2([0x20, 0x97])
2 spi.xfer2([0x25, 0x00])
```

Výpis 7.2 Příklad inicializace akcelerometru. (zdroj: vlastní)

- Registr 0x20: povolí všechny tři osy a nastaví výstupní datovou rychlost na 1600 Hz.
- Registr 0x25: vybere rozsah +/- 2 g v plném rozsahu, který poskytuje nejvyšší rozlišení.

Magnetometr LSM303D

Pro povolení magnetometru v LSM303D jsou řídicí registry konfigurovány následovně:

```
1 spi.xfer2([0x24, 0x14])
2 spi.xfer2([0x26, 0x00])
```

Výpis 7.3 Příklad inicializace magnetometru. (zdroj: vlastní)

- Registr 0x24: povolí režim s vysokým rozlišením a výstupní frekvenci 100 Hz.
- Registr 0x26: vybere režim kontinuální konverze pro kontinuální čtení dat.

Gyroskop L3GD20H

Gyroskop L3GD20H vyžaduje před čtením dat explicitní konfiguraci. Senzor L3GD20H se inicializuje zápisem do jeho řídicích registrů. Tento krok zajišťuje, že senzor začne vysílat gyroskopická data na všech třech osách v režimu nepřetržité aktualizace.

```
1 spi.xfer2([0x20, 0xf7])  
2 spi.xfer2([0x23, 0x00])
```

Výpis 7.4 Příklad inicializace gyroskopu. (zdroj: vlastní)

- Registr 0x20: bitová maska 0x0f (binárně 0b00001111) povoluje senzor a nastavuje čtení všech tří os.
- Registr 0x23: nastavuje škálování, kontinuální přenos dat a čtení dat ve formátu little-endian.

7.3 Měřicí smyčka a časování

Pro vyhodnocení komunikačního výkonu napříč různými frekvencemi je předdefinován seznam dotazovacích frekvencí. Pro každou takovou frekvenci je následně co nejrychleji zachycen pevný počet vzorků ze senzoru. Současně utilita sleduje, o kolik naměřených hodnot se liší od předchozího vzorku. Pro každou iteraci je aktuální časové razítko zaznamenáno pomocí monotónního časovače s vysokým rozlišením *time.perf_counter_ns()*. Cílem je získat pevný počet kompletních vzorků dat ze senzoru s minimálním zpožděním mezi vzorky a přesně zaznamenat čas každého snímání.

Časování je zaznamenáváno pomocí funkce *time.perf_counter_ns()* v jazyce Python, která poskytuje monotónní hodiny s vysokým rozlišením v nanosekundách. Čas každého vzorku je zaznamenáván vzhledem k začátku vzorkovací relace. To zajišťuje spolehlivé časování i za přítomnosti jitteru plánování CPU nebo jiné systémové aktivity. Časová razítka jsou převedena na mikrosekundy, aby byla zajištěna přesnost a zároveň zůstala čitelná pro člověka.

7.4 Čtení dat ze senzoru

Kromě hrubé rychlosti je užitečné sledovat, jak často se mění hodnoty senzorů, aby se zajistilo, že data nejsou jen opakovaným šumem nebo statickými hodnotami. Počet

změněných hodnot poskytuje vzhled do dynamiky senzorů i konzistence čtení.

Čtení dat ze senzorů je prováděno pomocí vícebajtového *SPI* přenosu, který začíná v počátečním registru. Sensory podporují sekvenční čtení všech šesti datových bajtů představujících v případě senzoru L3GD20H úhlové rychlosti X, Y a Z (každý ve formátu little-endian). To umožňuje přesnou intervalovou analýzu mezi po sobě jdoucími vzorky. Data se zaznamenávají v následujícím formátu: *sample, timestamp, x, y, z*. Příslušné volání čtení je strukturováno následovně:

```
1 spi.xfer2([READ_FLAG | MULTI_BYTE_FLAG | OUT] + [0x00]*6)
```

Výpis 7.5 Příklad čtení dat ze senzoru. (zdroj: vlastní)

Tento příkaz zahájí čtení počínaje adresou *OUT* s použitím příznaků čtení *READ_FLAG* a automatického inkrementace *MULTI_BYTE_FLAG* pro načtení celkem šesti bajtů. Funkce *spi.xfer2()* odešle všechny bajty najednou a vrátí odpověď senzoru. První bajt v odpovědi je potvrzení čtení a dalších šest bajtů obsahuje nezpracovaná data pro osy X, Y a Z v pořadí little-endian.

```
1 x = int.from_bytes(response[1:3], "little", signed=True)
2 y = int.from_bytes(response[3:5], "little", signed=True)
3 z = int.from_bytes(response[5:7], "little", signed=True)
```

Výpis 7.6 Příklad přečtení naměřených hodnot. (zdroj: vlastní)

Každý senzor používá mírně odlišnou adresu registru *SPI*, ale logika sběru dat je strukturálně stejná.

Akcelerometr LSM303D

Pro čtení dat z akcelerometru ze senzoru LSM303D může vypadat implementace následovně:

```
1 response = spi.xfer2([0x80 | 0x40 | 0x28] + [0x00]*6)
```

Výpis 7.7 Příklad přečtení dat z akcelerometru. (zdroj: vlastní)

- 0x80: Nastavuje bit pro čtení.

- 0x40: Aktivuje automatickou inkrementaci pro čtení více registrů v jedné transakci.
- 0x28: Počáteční adresa dolního bajtu výstupu osy X.

0x00 *6: Fiktivní bajty potřebné k odposlechu odezvy senzoru.

Magnetometr LSM303D

Magnetometr v LSM303D začíná na jiné adrese, ale používá stejný mechanismus čtení *SPI*:

```
response = spi.xfer2([0x80 | 0x40 | 0x08] + [0x00]*6)
```

Výpis 7.8 Příklad přečtení dat z magnetometru. (zdroj: vlastní)

Výstup magnetometru začíná v registru 0x08, který je specifický pro data magnetometru LSM303D. Dodržuje se stejný vzorec kombinování bitů pro čtení a automatické inkrementování se základní adresou. Odpověď opět obsahuje šest bajtů (X, Y, Z), které jsou analyzovány identicky.

Gyroskop L3GD20H

Přestože L3GD20H používá odlišnou vnitřní strukturu, jeho akcelerometr sdílí stejnou základní adresu výstupního registru 0x28 jako LSM303D, takže příkaz *SPI* je co do formy identický. Výstupní hodnoty jsou opět načítány jako 16-bitová celá čísla se znaménkem s použitím kódování little-endian.

7.5 Záznam a vyhodnocení

Utilita dodržuje identický formát zpracování naměřených vzorků a to co do formátu výstupního souboru, tak i samotných výstupních hodnot určených k následné vizualizaci či analýze (viz. 5.5).

8 NÁVRH UKLÁDÁNÍ DAT A VIZUALIZACE

Tato kapitola se zbývá implementačním návrhem nástroje pro zpracování a vizualizaci dat ze senzorů, vyvinutého konkrétně pro použití se senzory pohybu a orientace – gyroskopem L3GD20H a akcelerometrem/magnetometrem LSM303D. Ačkoliv oba senzory systém podporuje, senzory byly připojeny a používány samostatně během každého experimentu. Při návrhu softwarových komponent byl kladen důraz na rozdělení úkolů do dvou fází. Data ze senzorů jsou získávána pomocí utility skriptů, které upřednostňují přesnost načasování a nepřerušovaný sběr dat, zatímco vizualizace a je prováděna samostatně offline na naměřených datech.

Jedním z hlavních cílů tohoto nástroje je poskytnout rozšiřitelný mechanismus pro nepřetržitý sběr dat z více os, jejich efektivní ukládání ve strukturovaném formátu a vizualizaci pro následnou vizualizaci. Implementace podporuje možnosti vzorkování založené na časových razítkech i indexech. Kromě toho bylo zváženo více formátů serializace dat a konečný návrh odráží rovnováhu mezi výkonem, čitelností a kompatibilitou s běžnými vědeckými nástroji.

8.1 Sběr dat a vzorkování

K senzorům se přistupuje prostřednictvím příslušných digitálních rozhraní *SPI* a *I²C*. Komunikace se zařízením *SPI* je řešena prostřednictvím standardní knihovny *spidev*, která poskytuje přímý přístup k hardwarovému rozhraní *SPI* Raspberry Pi. Pro komunikaci *I²C* se používá knihovna *pigpio*. Tato knihovna slouží k dosažení přesnějšího řízení a konzistentního časování při čtení ze sběrnice *I²C*, zejména v situacích, kdy je kritické přesné řazení čtení registrů.

Bez ohledu na rozhraní je sběr dat součástí utility v Pythonu, který je přizpůsoben aktivnímu senzoru. Utilita konfiguruje senzor při spuštění a poté vstupuje do vysokofrekvenční vzorkovací smyčky, během níž se výstupní registry nepřetržitě čtou. Aby se předešlo narušení a zachovala se konzistence, během aktivního čtení vzorků se neprovádí žádné protokolování ani vstupně-výstupní operace se soubory. Místo toho jsou měření senzorů po dobu trvání relace ukládány do vyrovnávací paměti v paměti.

Každý vzorek dat může být označen buď s přesným časovým razítkem pomocí Pythonského *time.perf_counter()* pro monotónní časování s vysokým rozlišením, a nebo

s jednoduchým indexem vzorku, který se s každou iterací smyčky zvyšuje. Tyto dva přístupy nabízejí různé přístupy:

- Vzorkování založené na časovém razítku umožňuje přesnou rekonstrukci časových intervalů a je nezbytné v experimentech, kde je nutné zachytit nebo analyzovat variace časování. Je odolné vůči drobným zpožděním způsobeným plánovačem systému nebo proměnnou latencí komunikace.
- Vzorkování založené na indexu předpokládá konstantní vzorkovací frekvenci a může zjednodušit ukládání a následné zpracování, zejména v kontrolovaných experimentech nebo tam, kde je dostatečné relativní uspořádání.

Podporou komunikace *SPI* i *I²C* umožňuje použití s řadou typů senzorů. Poskytnutím možností vzorkování založených na časových razítkách a nebo indexech umožňuje uživatelům přizpůsobit proces záznamu svým potřebám z hlediska přesnosti. Použití vyrovnávací paměti v paměti a zpožděného zápisu souborů zajišťuje, že časově citlivý sběr dat není ohrožena zpožděním *I/O*.

8.2 Formáty ukládání dat

Po ukončení sběru dat, obvykle spouštěného ručně nebo po dosažení požadovaného počtu vzorků, se uložená data zapíše do souboru na disku. Návrh klade důraz na čitelnost pro člověka a interoperabilitu, a proto je použit standardní formát *CSV* (angl. Comma-Separated Values). Každý řádek souboru obsahuje buď časové razítko, nebo index vzorku následovaný daty z víceosých senzorů. Například:

```
1 timestamp, accel_x, accel_y, accel_z
2 0.00001, -0.03, 9.82, 0.12
3 0.02001, -0.02, 9.81, 0.10
```

Výpis 8.1 Příklad formátu uložených dat. (zdroj: vlastní)

Formát *CSV* je široce podporován a snadno je možné jej parsovat pomocí knihoven, jako jsou *pandas*, *numpy* nebo dokonce tabulkový software, což z něj činí vynikající volbu. Byly však zvažovány i jiné formáty ukládání, každý s odlišnými výhodami a omezeními:

- *JSON* je čitelný pro člověka a umožňuje vyjadřovat složité vnořené struktury. Může být užitečný pro zahrnutí dalších metadat (např. konfigurace senzorů, experimentální

podmínky). Pro rozsáhlá numerická data je však neefektivní kvůli své rozsáhlosti a nedostatku nativní podpory pro tabulkové struktury.

- Binární formáty (např. HDF5, NumPy) jsou optimalizovány pro výkon a kompaktnost. Podporují kompresi a umožňují rychlý náhodný přístup k podmnožinám dat. HDF5 podporuje zejména hierarchické datové sady a metadata, takže je vhodný pro komplexní systémy multisenzorového protokolování. Jejich interpretace však vyžaduje specializovaný software, což omezuje přístupnost pro člověka.
- Jednoduchý binární soubor je nejefektivnější variantou z hlediska prostoru. Neupravené binární formáty jsou extrémně kompaktní a rychlé, ale pro uživatele téměř zcela nečitelné bez definovaného schématu. Nejlépe se hodí pro integrované protokolování, kde je úložná kapacita značně omezená.

Volba CSV jako výchozího formátu pro ukládání dat odráží preferenci snadného použití a kompatibility, zejména v akademickém nebo výzkumném prostředí. Software je však navržen tak, aby umožňoval modulární nahrazení výstupní funkce, což by mohlo v případě potřeby usnadnit případnou budoucí migraci na výkonnější formát.

8.3 Vizualizace

Vizualizace je implementována jako samostatný skript v Pythonu, který funguje nezávisle na utilitách senzorů pro sběr dat. Tento skript slouží ke čtení uložených dat ve formátu CSV a vykresluje je pomocí knihovny matplotlib.

Logika vykreslování je navržena tak, aby vyhovovala různým typům senzorů a osám. Například gyroskopický výstup z L3GD20H je obvykle zobrazen jako úhlová rychlost v závislosti na čase pro každou ze tří prostorových os, zatímco data z akcelerometru LSM303D jsou vykreslena tak, aby zobrazovala změny lineárního zrychlení nebo posunu orientace. Popisky os a legendy se generují automaticky pro usnadnění interpretace.

Tato modulární architektura, kde je vizualizace oddělena od sběru dat, má několik výhod. Zaprvé, zabráňuje výpočetní režii během citlivé fáze vzorkování. Zadruhé, umožňuje snadné experimentování s různými styly vykreslování, technikami filtrování nebo odvozenými metrikami. A zatřetí, podporuje opětovné použití zaznamenaných datových sad bez nutnosti opětovného sběru dat, což usnadňuje zpětnou aplikaci nových analytických metod.

9 ANALÝZA VÝSLEDKŮ

Tato kapitola poskytuje přehled výsledků získaných z provedených experimentů s využitím benchmarkovacích utilit na připojených senzorech na sběrnici I^2C a SPI , a to za statických i dynamických podmínek. Důraz je kladen na pochopení vztahu mezi frekvencí sběrnice, výstupní datovou rychlostí senzoru ODR a pozorovanou variabilitou dat. Cílem je odvodit, jak rychlost vzorkování v porovnání s frekvencí aktualizace dat senzoru ovlivňuje kvalitu dat a jak se tyto vlivy mohou lišit při použití komunikačního protokolu SPI .

9.1 Stav senzoru v klidu

Ve statickém experimentu byl senzor umístěn na několikrát přeloženou tlustou látku pro docílení co největší absorpce vibrací z vnějších zdrojů. Po zvýšení výstupní datové rychlosti ODR senzoru na maximální hodnotu 1600 Hz (tj. jeden nový vzorek každých 625 mikrosekund) byla měření provedena při několika rostoucích frekvencích sběrnice I^2C .

Při nižších frekvencích I^2C , jako je 100 Hz až 1000 Hz, bylo všech 1000 shromážděných vzorků označeno jako „změněné“, což naznačuje, že každá operace čtení produkovala data odlišná od předchozí. To odráží skutečnost, že při nízkých rychlostech sběrnice je každá transakce I^2C relativně pomalá, což umožňuje dostatek času mezi po sobě jdoucími čteními, aby senzor aktualizoval své interní registry novými hodnotami a to i bez pohybu.

Jak se frekvence I^2C zvyšovala nad určitou prahovou hodnotu, objevil se zřetelný vzorec, kdy počet změněných vzorků se začal snižovat, obvykle se s každým krokem ve frekvenci sběrnice snižoval přibližně na polovinu předchozího vzorku. Toto chování lze vysvětlit vztahem mezi interním intervalem aktualizace dat senzoru a načasováním operací čtení iniciovaných hostitelem. Při vysokých rychlostech I^2C se čtecí smyčka stává natolik rychlou, že se často dotazuje senzoru ještě předtím, než aktualizuje své výstupní registry. V důsledku toho se zachycují redundantní hodnoty a počet odlišných přečtených dat (tj. změn) klesá.

Vyhodnocení

Tyto výsledky podtrhují vliv saturace frekvence dotazování. Když smyčka pro čtení dat překročí rychlost aktualizace dat na senzoru, nástroj načte duplicitní hodnoty. Ve statických podmínkách, kde se očekává, že hodnoty zrychlení zůstanou konstantní, je tento efekt dále zesílen absencí významných změn prostředí. Na počet změn tak působí pouze šum a jitter.

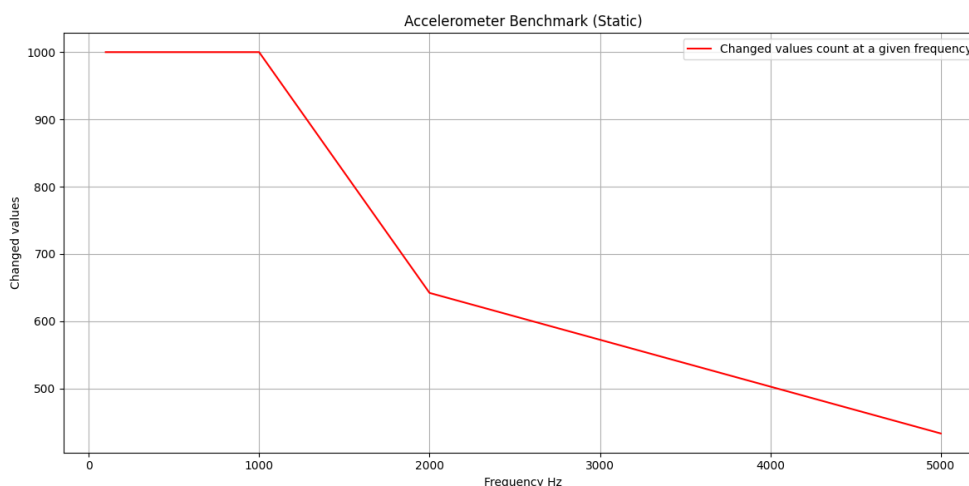


Figure 9.1 Graf vývoje přechtených změn v závislosti na zvyšující se frekvenci ve stavu kdy je senzor v klidu.

(zdroj: vlastní)

Tento experiment také zdůrazňuje důležitý metodologický bod, kdy vysoká frekvence sběrnice I^2C nemusí nutně vést k užitečnějším datům, pokud výstupní registry senzoru nejsou obnovovány stejnou rychlostí.

Porovnání se sběrnicí SPI

Při provedení stejného experimentu přes sběrnici *SPI* nastalo několik rozdílů. Protože *SPI* podporuje vyšší propustnost a obecně nižší latenci než sběrnice I^2C , systém může provádět operace čtení rychleji a s menší režií. Základní problém však zůstává nezměněn. Stále byly pozorovány opakované hodnoty, jakmile rychlost čtení *SPI* překročila *ODR* senzoru. I když je tedy *SPI* schopna umožnit dosažení limitu vzorkování při vyšší prahové frekvenci, stejné asymptotické chování, tedy kdy se počet změn ustálí nebo sníží, nakonec nastává, jakmile dojde k *bottleneck* aktualizace dat senzoru.

9.2 Stav senzoru v pohybu

Při dynamickém experimentu bylo senzorem během sběru dat záměrně pohybováno. Experimentální nastavení a frekvenční rozmítání I^2C byly identické se statickým případem, přičemž senzor byl konfigurován na *ODR* 1600 Hz.

Výsledky překvapivě odrážely výsledky ze statického stavu. U nejnižších tří frekvencí I^2C bylo všech 1000 vzorků označeno jako změněných. Při vyšších frekvencích se počet

změněných hodnot opět progresivně snižoval, přičemž byl pozorován přibližně stejný vzorec poloviční délky, jaký byl pozorován ve statickém experimentu.

Tato podobnost naznačuje, že schopnost systému zachytit nová data související s pohybem není zásadně omezena přítomností nebo nepřítomností fyzického pohybu, ale načasováním transakcí I^2C vzhledem k internímu mechanismu aktualizace senzoru. Když frekvence dotazování I^2C překročí rychlost produkce dat senzoru, je nemožné detekovat každou událost pohybu v reálném čase a vzorky se stávají redundantními.

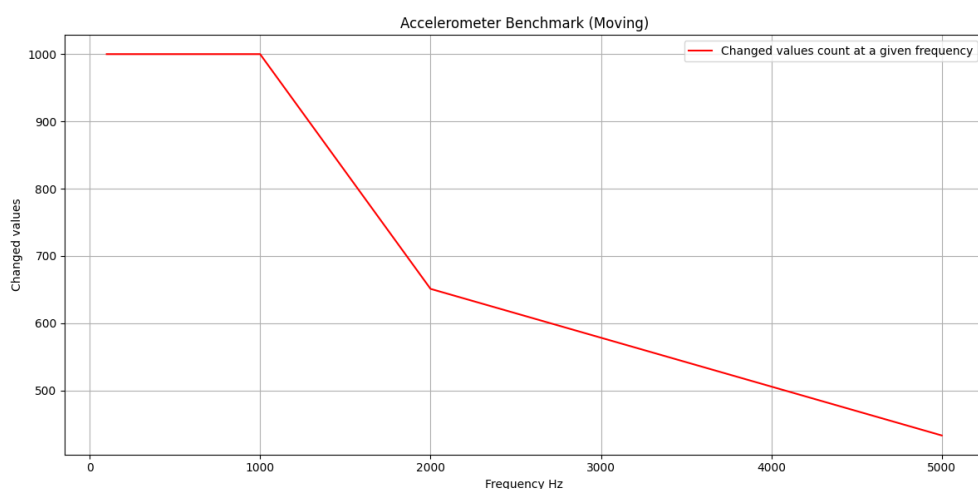


Figure 9.2 Graf vývoje přechtených změn v závislosti na zvyšující se frekvenci kdy je senzor v pohybu.

(zdroj: vlastní)

Vyhodnocení

Klíčovým zjištěním je fakt, kdy zvyšující se pohyb senzoru nekompenzuje nesoulad mezi frekvencí dotazování na straně hostitele a interní periodou aktualizace dat senzoru. I za aktivních podmínek vede dotazování rychlejší, než senzor dokáže aktualizovat své výstupy, k opakovanému čtení již přijatých dat.

Toto pozorování zpochybňuje intuitivní předpoklad, že větší pohyb by měl přirozeně vést k větší variabilitě vzorkovaných dat. V systémech založených na dotazování je odezva omezena nejen dynamikou senzoru, ale také schopností komunikačního protokolu zachytit aktualizované hodnoty včas.

Porovnání se sběrníci *SPI*

Při opakování tohoto dynamického experimentu s použitím *SPI* zvýšení komunikační propustnosti umožňuje systému udržet vysokou frekvenci dotazování s minimální režii protokolu. V důsledku toho by se dalo očekávat, že pokles počtu změn začne při vyšší frekvenci dotazování, čímž se rozšíří rozsah, ve kterém lze pozorovat plnou variabilitu dat. Nicméně základní omezení, tedy latence aktualizace registrů senzoru, zůstává. Jakmile frekvence dotazování *SPI* překročí schopnost senzoru obnovovat data, jsou přehlédnuty i změny vyvolané pohybem a opakované hodnoty se objevují znovu.

10 ZÁVĚR

Tato práce zkoumá komunikační výkon mezi Raspberry Pi 3 Model B a senzory přes sběrnice *I²C* a *SPI*, se zvláštním zaměřením na akcelerometr a magnetometr LSM303D a také na gyroskop L3GD20H. Primárním cílem bylo navrhnout a implementovat benchmarkingové utility schopné vyhodnotit efektivní vzorkovací frekvenci, variabilitu časování a integritu dat při čtení výstupů senzorů při vysokých frekvencích.

Pro každou konfiguraci senzoru a sběrnice byla vyvinuta sada Pythonových utilit. Tyto programy byly navrženy tak, aby sekvenčně testovaly rozsah komunikačních rychlostí, zachytily pevně daný počet po sobě jdoucích vzorků při každé rychlosti a uložily jak naměřená data, tak souhrnné statistiky ve formátu CSV. Zaznamenaná data byla strukturována pro kompatibilitu pro navrhovaný vizualizační nástroj, což usnadňuje další analýzu.

Každá utilita zachycuje časová razítka pomocí čítačů s vysokým rozlišením, aby určila průměrné intervaly vzorkování a identifikovala, jak často probíhaly změny hodnot senzorů. Tento mechanismus detekce změn se ukázal jako účinný ukazatel pro pochopení, zda je frekvence dotazování na straně hostitele v souladu s interní obnovovací frekvencí senzoru.

Provedením experimentů bylo zjištěno několik klíčových poznatků. Jak ve statickém stavu, tak i dynamických podmínkách vykazovaly senzory charakteristický pokles v počtu změřených hodnot, když se frekvence sběrnice zvýšila nad kritickou hranici. Toto zjištění odráží nesoulad mezi rychlostí dotazování dat a schopností senzoru produkovat nové vzorky. I ve scénářích s vysokým výskytem pohybu vedlo nadměrné dotazování k duplicitním hodnotám, což potvrdilo důležitost sladění frekvence odečtu s ODR senzoru.

Při porovnání *I²C* a *SPI* se ukázalo, že ačkoli *SPI* nabízí výrazně vyšší komunikační propustnost, zásadně nemění strop vzorkování daný interní obnovovací frekvencí dat senzoru. Místo toho *SPI* umožňuje dosáhnout tohoto limitu efektivněji a s nižší variabilitou, což z něj činí robustnější volbu v aplikacích vyžadujících konzistentní načasování a vysokorychlostní sběr dat.

Lze tedy říci, že vyvinuté nástroje poskytly potenciální experimentální základ pro posouzení komunikačních limitů senzorů a pomohly přiblížit zásadní vztah mezi rychlostí sběrnice, logikou dotazování a konfigurací senzorů.

REFERENCES

- [1] SEMICONDUCTORS, NXP, 2021. *UM10204 I2C-bus specification and user manual*. Online. Dostupné z: <https://web.archive.org/web/20221006073143/http://www.nxp.com/docs/en/user-guide/UM10204.pdf>. [cit. 2024-10-23].
- [2] MOTOROLA, Inc., 2003. *SPI Block Guide V03.06*. Online. Dostupné z: <https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee3081/datasheets/S12SPIV3.pdf>. [cit. 2024-10-23].
- [3] TIŠNOVSKÝ, Pavel, 2009. *Komunikace po sériové sběrnici I²C*. Online. Dostupné z: <https://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/>. [cit. 2024-10-23].
- [4] TEXAS INSTRUMENTS, INC., 2015. *Understanding the I2C Bus*. Online. Dostupné z: <https://www.ti.com/lit/an/slva704/slva704.pdf>. [cit. 2025-03-17].
- [5] DHAKER, Piyu, 2018. *Introduction to SPI Interface*. Online. Dostupné z: <https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html>. [cit. 2025-02-11].
- [6] TEXAS INSTRUMENTS, INC., 2009. *TMS320x281x Serial Peripheral Interface Reference Guide*. Online. Dostupné z: <https://www.ti.com/lit/ug/spru059e/spru059e.pdf>. [cit. 2024-10-23].
- [7] TIŠNOVSKÝ, Pavel, 2008. *Externí sériové sběrnice SPI a I²C*. Online. Dostupné z: <https://www.root.cz/clanky/externi-seriove-sbernice-spi-a-i2c/>. [cit. 2024-10-23].
- [8] COOK, Jeremy, 2024. *Understanding Sensor Interfaces - UART, I2C, SPI and CAN*. Online. Dostupné z: <https://dev.blues.io/blog/blues-university-understanding-sensor-interfaces-uart-i2c-spi-can/>. [cit. 2025-02-11].
- [9] FREESCALE SEMICONDUCTOR, Inc., 2004. *Using the Serial Peripheral Interface to Communicate Between Multiple Microcomputers*. Online. Dostupné z: <https://www.nxp.com/docs/en/application-note/AN991.pdf>. [cit. 2025-02-11].

- [10] ALIN STOICESCU, Microchip Technology Inc., 2018. *TB3215 Getting Started with SPI*. Online. Dostupné z: <https://ww1.microchip.com/downloads/en/Appnotes/TB3215-Getting-Started-with-SPI-90003215A.pdf>. [cit. 2025-02-11].
- [11] BEST-MICROCONTROLLER-PROJECTS. *SPI interface Tutorial*. Online. Dostupné z: <https://www.fpga4fun.com/SPI1.html>. [cit. 2025-02-11].
- [12] SFUPTOWNMAKER, 2013. *I2C*. Online. Dostupné z: <https://learn.sparkfun.com/tutorials/i2c>. [cit. 2025-03-11].
- [13] TEXAS INSTRUMENTS, INC., 2022. *A Basic Guide to I2C*. Online. Dostupné z: <https://www.ti.com/lit/an/sbaa565/sbaa565.pdf?ts=1747530537627>. [cit. 2025-03-17].
- [14] CORPORATION, Pololu, 2025. *L3GD20H 3-Axis Gyro Carrier with Voltage Regulator*. Online. Dostupné z: <https://www.pololu.com/product/2129>. [cit. 2025-04-17].
- [15] CORPORATION, Pololu, 2025. *LSM303D 3D Compass and Accelerometer Carrier with Voltage Regulator*. Online. Dostupné z: <https://www.pololu.com/product/2127>. [cit. 2025-04-17].
- [16] PI, Raspberry, 2025. *Raspberry Pi hardware*. Online. Dostupné z: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#gpio>. [cit. 2025-04-17].

List of Figures

Obrázek 2.1	Diagram komunikace po sběrnici <i>SPI</i>	19
Obrázek 2.2	Schéma zapojení <i>SPI</i>	20
Obrázek 2.3	Módy přenosů dat v <i>SPI</i>	22
Obrázek 2.4	Schéma zapojení <i>SPI</i> v módu <i>Master - Slave</i>	23
Obrázek 2.5	Schéma zapojení více zařízení na sběrnici <i>SPI</i> pomocí <i>CS</i>	24
Obrázek 2.6	Schéma zapojení více zařízení na sběrnici <i>SPI</i> pomocí <i>Daisy chain</i>	25
Obrázek 3.1	Diagram komunikace po sběrnici <i>I²C</i>	27
Obrázek 3.2	Diagram zapojení sběrnice <i>I²C</i>	29
Obrázek 3.3	Diagram časového signálu sběrnice <i>I²C</i>	31
Obrázek 4.1	Schéma zapojení <i>SPI</i>	36
Obrázek 9.1	Graf experimentu v klidu	61
Obrázek 9.2	Graf experimentu v pohybu	62

List of Tables

Tabulka 3.1	Režimy rychlostí sběrnice I^2C	31
Tabulka 4.1	Zapojení senzoru k desce.....	36
Tabulka 4.2	Příklad změny adresy senzoru při přivedení <i>GND</i> či <i>VCC</i> na pin senzoru <i>SDO</i>	38
Tabulka 6.1	Zapojení senzoru k desce.....	49

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

<i>SPI</i>	Serial Periphetal Interface
<i>I²C</i>	Inter-Integrated Circuit
<i>IoT</i>	Internet of Things
<i>MSB</i>	Most Significant Bit
<i>ODR</i>	Output Data Rate
<i>MISO</i>	Master-In Slave-Out
<i>MOSI</i>	Master-Out Slave-In
<i>GPIO</i>	General-purpose input/output
<i>GND</i>	Ground
<i>IC</i>	Integrated Circuit
<i>I/O</i>	Input/Output
<i>CS</i>	Cable select
<i>SS</i>	Slave select
<i>SMBus</i>	System Management Bus

<i>ACK</i>	Acknowledge
<i>NACK</i>	Not Acknowledge
<i>SCL</i>	Serial Clock Line
<i>SDA</i>	Serial Data Line
<i>SDI</i>	Serial Data In
<i>SDO</i>	Serial Data Out
<i>CPOL</i>	Clock Polarity
<i>CPHA</i>	Clock Phase
<i>CLK</i>	Clock
<i>R/W</i>	Read/Write
<i>SDA</i>	Serial Data
<i>CPU</i>	Central Processing Unit
<i>VCC</i>	Voltage Common Collector