FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Katedra informatiky a výpočetní techniky

Komunikační periferie pro emulátor Raspberry Pi Zero

Autor: Patrik Holub

Datum: 14 Ledna 2026

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 14. Ledna 2026 Patrik Holub

Abstract

We designed and implemented a library to facilitate communication between multiple instances of Raspberry Pi Zero emulator. The library is presented transparently as a peripheral device to the emulator, meaning the running program doesn't realize, how the communication to the other device is facilitated. This allows to create a distributed network of emulators, all thinking, they are wired together directly, improving simulation capabalities of said emulator.

Obsah

1 Úvod

Při vývoji softwaru pro specifická fyzická zařízení (tzv. embedded systémy) naráží vývojáři na překážky, které u běžných desktopových aplikací neexistují. Příklady mohou být přístupnost k cílovému hardwaru, šance poškození hardwaru, rychlost testování softwaru (iterační cyklus), nebo třeba cenové problémy, kvůli nutnosti více zařízení.

Typickým příkladem nepřístupnosti cílového hardwaru je situace, kdy je laboratoř s prototypy zařízení v jiném městě, než kde programátor reálně žije nebo pracuje. V takovém případě je efektivní práce bez vzdáleného přístupu nebo alternativního řešení téměř nemožná. Navíc, testovacích zařízení bývá často omezené množství, což brzdí práci v týmu, kdy musí více programátorů čekat na uvolnění jednoho kusu hardwaru. Počet zařízení také může vytvářet omezení pokud je software mířený na distribuované prostředí.

Dalším kritickým aspektem vývoje pro fyzická zařízení je riziko poškození hardwaru chybou v softwaru. Zatímco chyba v běžném programu maximálně způsobí pád aplikace, špatná manipulace s piny nebo napětím na mikrokontroléru může vést k nevratnému zničení součástek.

S rostoucími požadavky na komplexitu dnešních systémů je navíc prakticky nemožné provést stoprocentní analýzu kódu a garantovat jeho správnost ve všech teoretických stavech čistě "na papíře". I při sebelepší kontrole se mohou vyskytnout chyby v místech, kde je programátor vůbec nepředpokládá (tzv. edge cases). Ve fyzickém světě může taková neočekávaná chyba vést k elektrickému zkratu a "odpálení"drahého prototypu.

Právě zde přichází na řadu emulace, která tato rizika eliminuje. Jelikož je emulátor čistě softwarová záležitost, nemůže dojít k žádné fyzické škodě, ať už se program chová jakkoliv nepředvídatelně. Zároveň lze na jednom zařízení spustit více instancí emulátoru, čímž je možné simulovat distribuované systémy.

Právě zde přichází naše práce, jejímž cílem je vytvořit periferii, která je schopná zajistit komunikaci mezi více instancemi emulátoru. Cílem je také zajistit, aby tato komunikace probíhala transparentně, neboli aby programu běžícímu v emulátoru bylo jedno, jestli je opravdu připojena periferie, kterou očekává, nebo jestli periferie běží na jiném stroji. Stroje mohou být odděleny i internetovou sítí a emulátoru by to mělo být jedno. Tato možnost by měla zajistit možnost simulovat silně distribuovaný systém za pomoci výpočetních serverů.

2 Emulátor ZeroMate

Emulátor ZeroMate, v jeho stavu před touto prací, vznikl jako diplomová práce pana Ing. Jakuba Šilhavého [9]. Jde o emulátor, který vznikl pro emulaci operačního systému KIV-RTOS [6], vyvinutého pro zařízení Raspberry Pi Zero W (deska odvozená z BCM2835). Tento mikrokontrér disponuje procesorem, který používá architekturu ARM, specificky ARMv6. ARM je achitektura, spadající do skupiny RISC (Reduced Instruction Set Computer). Jde o architekturu počítače, kdy sada instrukcí je menší a každá instrukce má fixní délku. Jde o jednodušší přístup k výpočetní technice než CISC (Complex Instruction Set Computer). RISC umožňuje vytvořit processor, kde části jako dekodér nebo ALU mají redukovanou komplexitu, což vede k energetickým úsporám. Tyto výhody posunuli ARM na technologii, která pohání 99% mobilních telefonů na celém světě [8].

Emulátor vznikl, kvůli požadavkům na detailní introspekci do běhu programu. Zároveň se soustředí pouze na ARMv6, což vytváří velmi specializovaný nástroj. Tato nutnost bylo spozorována při porovnání emulátorů dostupných v době kdy byla diplomová práce sepsána. Ostatní nástroje buď nemají pokročilé introspekce, nebo ve snaze být systémy, schopné emulovat mnoho jiných architektur, mají problémy replikovat specifické situace vestavěných zařízení. Cíl práce byl hlavně emulovat operační systém KIV-RTOS, který je sám tvořen pro jeden typ hardwaru. Jde o operační systém používaný a vyvíjený v univerzitním prostředí, kde problémy zmíněné v úvodu jsou velmi časté.

2.1 Přehled architektury

2.1.1 Vstup a načítání programu

Vstupem pro emulátor je soubor ve formátu ELF (Executable and Linkage Format). Tento formát byl zvolen pro svou schopnost uchovávat jak spustitelný kód, tak ladicí symboly, což usnadňuje disasemblování a následnou vizualizaci běhu programu. Komponenta *ELF Loader* je zodpovědná za parsování tohoto souboru, kopírování textové sekce do emulované paměti RAM a demangling symbolů pro čitelnější výpis funkcí.

2.1.2 Jádro emulátoru

Jádro emulátoru se skládá z několika klíčových, vzájemně spolupracujících komponent:

- **Systémová sběrnice:** Slouží jako centrální komunikační rozhraní pro zařízení. Zprostředkovává čtení a zápis mezi CPU a pamětí nebo periferiemi. Udržuje seznam všech mapovaných periferií a zajišťuje směrování požadavků na správnou adresu.
- **CPU (ARM1176JZF-S):** Hlavní výpočetní jednotka, která je dále dekomponována na:
  - *ISA Decoder:* Identifikuje typ instrukce z 32-bitového kódu.
  - *ALU a MAC:* Provádí aritmetické a logické operace.
  - *CPU Context:* Uchovává stav registrů (včetně bankovaných registrů pro různé módy CPU) a aktuální mód procesoru.
  - *MMU (Memory Management Unit):* Zajišťuje překlad virtuálních adres na fyzické pomocí jedúrovňové tabulky stránek (v rámci zjednodušení emulátoru).
- **Koprocesory:** Emulátor implementuje coprocesor CP15 pro konfiguraci systému a CP10 pro operace s plovoucí řádovou čárkou (FPU).
- **Paměť (RAM):** Implementována jako souvislý blok paměti o velikosti 512 MB.

2.1.3 Interní periferie BCM2835

Emulátor simuluje vybrané interní periferie mikrokontroléru BCM2835, které jsou mapovány do paměťového prostoru. Mezi nejdůležitější patří:

- **Interrupt Controller:** Spravuje přerušení z různých zdrojů (GPIO, časovače, UART).
- **GPIO Manager:** Řídí stav 54 univerzálních I/O pinů.
- **ARM Timer:** Poskytuje časovací funkce odvozené od taktů emulovaného CPU.
- **UART a I2C (BSC):** Zajišťují sériovou komunikaci.

2.2 Práce s externími periferiemi

Jedním z hlavních cílů návrhu emulátoru ZeroMate byla jeho rozšiřitelnost o uživatelsky definované externí periferie (např. tlačítka, LED diody, displeje), které nejsou součástí samotného čipu SoC, ale jsou k němu připojeny v reálném světě. Tato funkcionalita je klíčová pro věrnou simulaci vestavěných systémů.

2.2.1 Architektura externích periferií

Externí periferie jsou řešeny pomocí samostatné dynamické knihovny (sdílené knihovny .dll na Windows nebo .so na Linuxu). Toto řešení umožňuje:

1. Vývoj periferií nezávisle na jádře emulátoru.
2. Možnost připojení více instancí stejné periferie (např. více LED diod).
3. Konfiguraci zapojení bez nutnosti rekompilace emulátoru.

Implementace externí periferie musí odpovídat definovanému programovému rozhraní (`IExternal_Peripheral`). Jádro emulátoru při startu načte tyto knihovny na základě konfiguračního souboru.

2.2.2 Rozhraní a komunikace

Komunikace mezi emulátorem a externí periferií probíhá primárně přes simulované GPIO piny a systémové hodiny.

2.2.2.1 Napojení na GPIO

Při inicializaci periferie volá emulátor metodu z výše zmíněného rozhraní `Get_GPIO_Subscription`, prostřednictvím které periferie sdělí, které GPIO piny chce sledovat.

Pokud dojde ke změně stavu GPIO pinu (nastavení pinu na HIGH nebo LOW), emulátor iteruje přes všechny připojené externí periferie. Pokud periferie odebírá daný pin, tak je zavolána obslužná metoda. Perieferie musí tuto metodu rychle obsloužit, a vrátit řízení emulátoru. V tomto ohledu se jedná o podobný princip jako u interruptů. Každá periferie si tedy napíše obsluhu změny pinu, kterou požije např. k rozsvícení LED. Komunikace

je obousměrná – externí periferie může také měnit stav pinu (např. stisk virtuálního tlačítka změní stav pinu na LOW), což může vyvolat přerušení v jádře emulátoru.

2.2.2.2 Synchronizace času a hodin

Externí periferie nemají vlastní zdroj hodin v reálném čase, ale jsou synchronizovány s emulovaným časem CPU. Periferie dostávají informaci o počtu proběhlých cyklů CPU po vykonání každé instrukce (nebo bloku instrukcí). Toto je kritické pro periferie, které závisí na časování, jako jsou sériové terminály nebo displeje komunikující přes I2C, kde je nutné dodržet specifické časování protokolu.

2.2.3 Konfigurace zapojení (peripherals.json)

Zapojení periferií je definováno v souboru `peripherals.json`. Tento soubor určuje jak by měl emulátor načíst periferii. Hlavní části jsou:

- **Cesta k knihovně:** Kde se nachází soubor .dll/.so.
- **Název:** Unikátní identifikátor instance.
- **Propojení:** Mapování logických pinů periferie na fyzické GPIO piny emulátoru (např. připojení LED na GPIO pin 47).

2.2.4 Grafické rozhraní externích periferií

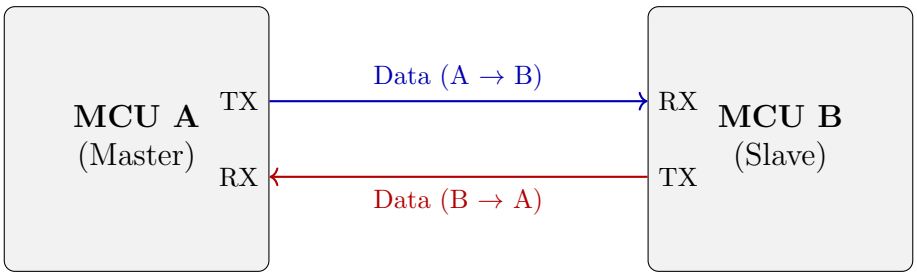
Ačkoliv to není povinné, externí periferie mohou implementovat vlastní grafické rozhraní. Emulátor jim poskytuje kontext pro vykreslování (využívající knihovnu ImGui), což jim umožňuje vizualizovat svůj stav (např. svítící segmenty na 7-segmentovém displeji nebo text na OLED displeji) přímo v okně aplikace ZeroMate.

3 Komunikační protokoly

3.1 UART

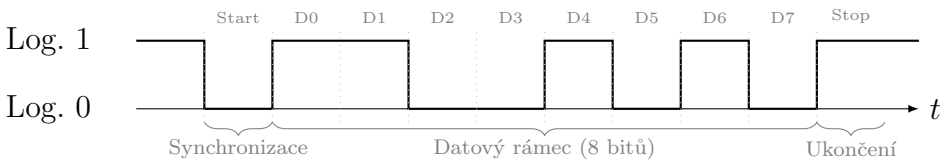
UART (Universal Asynchronous Receiver-Transmitter) představuje jeden z nejzákladnějších komunikačních standardů, který se v oblasti mikrokontrolerů využívá pro sériový přenos dat. Jak již název napovídá, jedná se o asynchronní komunikaci, což znamená, že mezi vysílačem a přijímačem není veden žádný sdílený hodinový signál. Aby byla komunikace úspěšná, musí se obě strany předem dohodnout na přesné přenosové rychlosti, udávané v baudech (Baud Rate). Časování vzorkování signálu je pak odvozováno interně z vlastního zdroje hodin každého zařízení, což klade nároky na přesnost těchto oscilátorů.

Fyzická vrstva je tvořena dvěma datovými vodiči, obvykle označovanými jako RX (příjem) a TX (vysílání), které jsou mezi zařízeními zapojeny křížově. V klidovém stavu, kdy neprobíhá žádný přenos, je linka udržována na logické úrovni jedna (High). [4]



Obrázek 3.1: Schéma zapojení UART

Samotný přenos zprávy, neboli rámce, je vždy zahájen tzv. start bitem. Ten změní úroveň linky na logickou nulu, čímž synchronizuje přijímač a připraví ho na čtení nadcházejících dat. Následuje samotná datová část rámce, která se skládá typicky z osmi bitů, přičemž standardem je odesílání od nejméně významného bitu (LSB) po ten nejvýznamnější. Za daty může následovat volitelný paritní bit, který slouží k základní kontrole integrity přenosu a detekci chyb. Celý rámec je ukončen jedním nebo dvěma stop bity, které vrací linku zpět do klidového stavu logické jedničky. Tato pauza je nezbytná, aby přijímač mohl správně detekovat začátek dalšího případného start bitu. [4]



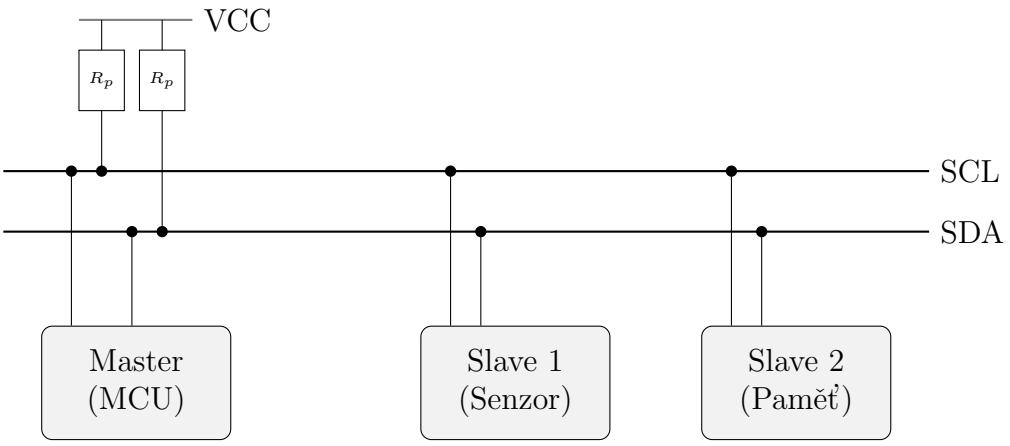
Obrázek 3.2: Časový průběh přenosu znaku 'S' (0x53, LSB first)

Z hlediska hardwarové implementace uvnitř mikrokontroleru je klíčovým prvkem posuvný registr (shift register). Procesor zapíše celá data (například jeden bajt) paralelně do vyrovnávacího registru. Hardware UART periferie následně tato data přesune do posuvného registru, odkud jsou v přesně daném taktu vysouvána na výstupní pin po jednotlivých bitech. Na straně příjmu funguje proces opačně, kdy jsou jednotlivé bity postupně nasouvány do registru, dokud není sestaven celý bajt, který je následně zpřístupněn procesoru ke čtení.

3.2 I2C

Protokol I2C (Inter-Integrated Circuit), je synchronní sériová sběrnice určená pro komunikaci na krátké vzdálenosti. Na rozdíl od UARTu, který spojuje pouze dvě zařízení, je I2C sběrnicí, což umožňuje připojení vícero zařízení na stejné komunikační linky. Architektura je založena na vztahu Master-Slave, kde Master (řídící zařízení) iniciuje veškerou komunikaci a generuje hodinový signál pro synchronizaci přenosu dat. Slave (podřízené zařízení) pouze reaguje na výzvy zařízení Master.

Fyzickou vrstvu tvoří dva vodiče: SDA (Serial Data) pro přenos dat a SCL (Serial Clock) pro hodinový signál. Klíčovou vlastností hardwarového řešení je zapojení výstupů jako tzv. otevřený kolektor (open-drain [2]). To znamená, že zařízení mohou linku pouze stáhnout na logickou nulu (uzemnit), ale nemohou ji aktivně vybudit do logické jedničky. Klidový stav logické jedničky je zajištěn externími pull-up rezistory [2] (v schématu 3.3 označeny jako R_p), které linky drží na napájecím napětí, pokud žádné zařízení nevysílá nulu. Toto zapojení zabraňuje zkratům v případě, že by více zařízení omylem vysílalo opačné hodnoty současně, a umožňuje funkce jako "clock stretching", kdy pomalejší Slave může podržením linky SCL na nule pozastavit komunikaci.



Obrázek 3.3: Topologie sběrnice I2C

Komunikace je zahájena specifickou sekvencí zvanou Start Condition [5], kdy Master stáhne datovou linku SDA do nuly, zatímco hodinová linka SCL je stále v logické jedničce. Následuje vysílání adresy zařízení, se kterým chce Master komunikovat. Adresa je typicky 7-bitová, což umožňuje teoreticky adresovat až 128 zařízení. Osmý bit v prvním bajtu určuje směr komunikace (čtení nebo zápis). Pokud cílové zařízení na sběrnici existuje, potvrdí přijetí adresy stažením linky SDA v devátém hodinovém taktu, což se nazývá ACK [5].

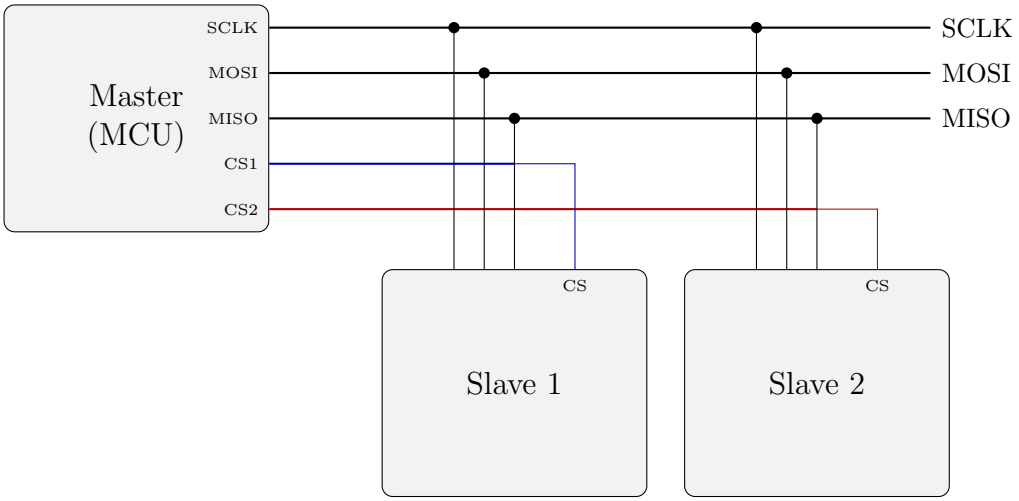
Samotná data jsou přenášena po bajtech, přičemž každý bajt musí být potvrzen přijímací stranou pomocí bitu ACK. Přenos probíhá synchronně s hodinovým signálem na lince SCL, kdy se stav datové linky SDA může měnit pouze v době, kdy je SCL na logické nule. Během vysoké úrovně hodin musí být data stabilní. Ukončení komunikace provede Master sekvencí Stop Condition, při které uvolní linku SDA do logické jedničky v momentě, kdy je SCL také v jedničce.

3.3 SPI

SPI (Serial Peripheral Interface) je vysokorychlostní synchronní sériové rozhraní, které se využívá především pro komunikaci mezi mikrokontrolerem a rychlými periferiemi, jako jsou SD karty, LCD displeje nebo senzory s vysokou vzorkovací frekvencí. Na rozdíl od I2C, SPI nevyužívá systém adresování ani potvrzování přijatých dat (ACK/NACK), což minimalizuje režii a umožňuje

dosahovat mnohem vyšších datových toků. Komunikace probíhá v režimu Full-Duplex [3], což znamená, že data jsou současně vysílána i přijímána.

Populární a velmi časté fyzické zapojení vyžaduje čtyři vodiče [3]. Linka `SCLK` (Serial Clock) přenáší hodinový signál. Hodiny jsou řízené zařízením Master. Data od Master zařízení k Slave zařízení proudí po vodiči `MOSI` (Master Out Slave In), zatímco opačným směrem slouží vodič `MISO` (Master In Slave Out). Pro výběr zařízení na sběrnici se používá dedikovaná linka `CS` nebo `SS` (Chip/Slave Select). Protože SPI nemá adresní systém, musí mít každé Slave zařízení vlastní CS vodič vedoucí k Master zařízení. Aktivací této linky (obvykle stažením do logické nuly) Master dává zařízení najevo, že s ním hodlá komunikovat. [3]

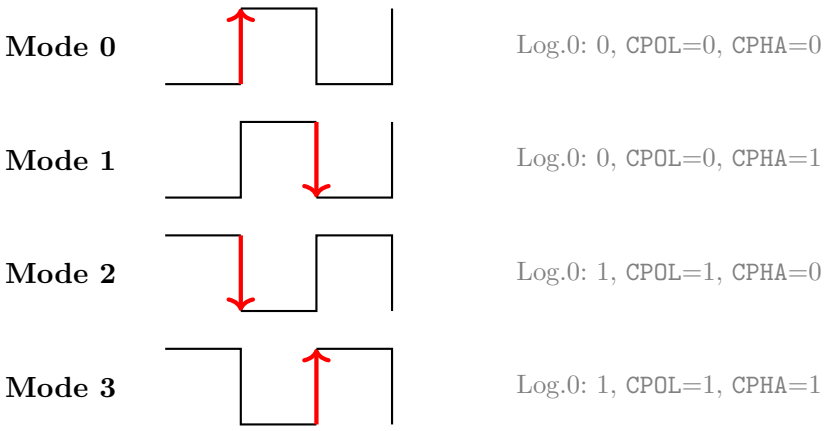


Obrázek 3.4: Topologie SPI

Zpráva u SPI není pevně definována do rámců s pevnou délkou jako u UARTu. Délka přenosu závisí čistě na Master zařízení, které generuje tolik taktů hodin, kolik bitů potřebuje přenést. Hardwarově je SPI realizováno pomocí dvojice posuvných registrů – jeden v Master a jeden ve Slave zařízení. Tyto registry jsou vzájemně propojeny do kruhu přes linky MOSI a MISO. S každým taktem hodin je jeden bit vysunut z Master do Slave a současně je jeden bit vysunut ze Slave do Master [3]. Po osmi taktech (při přenosu bajtu) si tak obě zařízení v podstatě "vymění"obsah svých posuvných registrů.

Důležitým aspektem SPI je konfigurace hodin, která se definuje pomocí polarity (CPOL) a fáze (CPHA). Nutno podotknout, že CPOL a CPHA nejsou fyzické vodiče na sběrnici, ale interní nastavení řadičů v Master i Slave zařízeních, které určují, jak mají interpretovat signály na sdílených linkách.

Polarita (CPOL) určuje klidový stav linky SCLK (0 nebo 1). Fáze (CPHA) definuje, na které hraně (první nebo druhé v rámci cyklu) se vzorkují data. Kombinací vznikají čtyři módy (viz obr. 3.5).



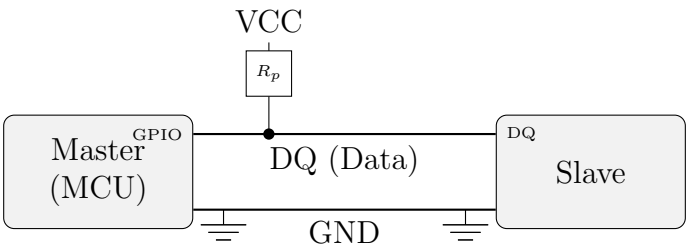
Obrázek 3.5: Čtyři módy SPI

3.4 1-Wire

Protokol 1-Wire, vyvinutý společností Dallas Semiconductor, je unikátní svou minimalistickou fyzickou vrstvou, která pro obousměrnou komunikaci i napájení využívá jediný datový vodič a společnou zem. Tato vlastnost z něj činí ideální řešení pro systémy, kde je kladen důraz na nízkou cenu kabeláže a konektorů, nebo pro zařízení, která jsou od mikrokontroleru značně vzdálená (např. digitální teploměry DS18B20). Podobně jako u I2C, i zde je linka v klidu udržována v logické jedničce pomocí pull-up rezistoru a komunikace probíhá na principu otevřeného kolektoru.

Vzhledem k absenci hodinového signálu je synchronizace mezi Master a Slave zařízeními založena na velmi přesnému časování jednotlivých impulzů. Komunikace začíná resetovacím pulzem, kdy Master stáhne linku na delší dobu k nule. Pokud je na sběrnici přítomno nějaké zařízení, odpoví tzv. prezenčním pulzem (Presence Pulse). Bity jsou přenášeny v definovaných časových slotech. Pro zápis logické nuly Master stáhne linku na delší dobu, zatímco pro zápis jedničky ji uvolní velmi rychle. Čtení probíhá tak, že Master linku krátce stáhne a následně vzorkuje její stav v přesně daném okamžiku, kdy ji

Slave buď drží u země (nula), nebo nechá v jedničce.



Obrázek 3.6: Zapojení 1-Wire sběrnice

Každé 1-Wire zařízení má v sobě z výroby vypálené unikátní 64-bitové identifikační číslo (ROM ID) [1]. Toto číslo slouží jako absolutní adresa, díky které může Master na jediné lince komunikovat s velkým množstvím zařízení, aniž by docházelo ke kolizím. Proces vyhledávání těchto adres na sběrnici (Enumeration) probíhá pomocí binárního stromu, kdy Master postupně eliminuje zařízení, dokud neidentifikuje všechna přítomná ID. To umožňuje dynamické připojování čidel bez nutnosti předchozí konfigurace systému.

Zajímavou vlastností 1-Wire je možnost tzv. parazitního napájení. Zařízení jsou schopna čerpat energii přímo z datové linky v době, kdy je v logické jedničce, a ukládat ji do vnitřního kondenzátoru pro pokrytí spotřeby během krátkých okamžiků, kdy je linka v nule. V mikrokontrolerech se 1-Wire často implementuje softwarově (tzv. bit-banging) pomocí manipulace s GPIO piny, protože vyžaduje velmi precizní zpoždění v řádu mikrosekund. Některé pokročilejší MCU však disponují specializovanými hardwarovými řadiči nebo využívají pro emulaci časování periferii UART.

3.5 Ethernet

Ethernet představuje v kontextu mikrokontrolerů značný skok v komplexitě a možnostech komunikace. Jedná se o rodinu standardů definovanou normou IEEE 802.3, která na rozdíl od předchozích lokálních sběrnic slouží k propojování zařízení na větší vzdálenosti a v rámci rozsáhlejších sítí. Jeho nasazení je vhodné především tam, kde je potřeba zajistit spolehlivý přenos dat mimo hranice jedné desky plošných spojů a integrovat zařízení do existující síťové infrastruktury.

Z hlediska hardwarové implementace je pro běžný mikrokontroler přímá generace signálů pro fyzickou vrstvu nereálná. Proto se v praxi využívá rozdělení kompetencí. Mikrokontroler může obsahovat blok pro řízení přístupu k médiu (`MAC` [7]), ale samotný převod dat na signály fyzické vrstvy přenechává externímu obvodu, označovanému jako `PHY` [7]. Tento přístup umožňuje procesoru pracovat s daty, aniž by musel řešit složitou modulaci signálu pro konkrétní přenosové médium.

V oblasti vestavěných systémů nachází Ethernet využití specificky v roli nadřazených prvků nebo komunikačních bran (gateways). Typickám situace je případ, kdy centrální mikrokontroler sbírá data z lokálních senzorů pomocí jednodušších protokolů, jako jsou I2C nebo SPI, a Ethernet následně využívá k odeslání těchto agregovaných informací do nadřazeného systému. Slouží tedy jako výkonná linka pro připojení lokální elektroniky do větší sítě zařízení.

4 Závěr

V této práci jsme provedli soupis základních konceptů a informací potřebných pro pochopení nadcházejícího textu bakalářské práce. Museli jsme projít diplomovou práci popisující emulátor ZeroMate. Zde bylo třeba vytvořit popis celkové funkčnosti a hlavně popsat systém, kterým emulátor načítá externí periferie. Dále jsou zde obsáhlé popisy základních komunikačních protokolů, které se často objevují v mikrokontrolerech. Ty jsou důležité, pro vývoj komunikačního protokolu komunikační periferie, aby byla schopna tyto protokoly emulovat.

Literatura

[1] B. Linke. „Overview of 1-Wire Technology and Its Use.“ [online], cit. 14. led. 2026. URL: https://www.analog.com/en/resources/technical-articles/guide-to-1wire-communication.html.

[2] T. Instruments, *I^2C Bus Pullup Resistor Calculation*, Dallas, Texas, 2015.

[3] P. Dhaker. „Introduction to SPI Interface.“ [online], cit. 14. led. 2026. URL: https://www.analog.com/en/resources/analog-dialogue/rticles/introduction-to-spi-interface.html.

[4] E. Peňa a M. G. Legaspi. „UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter.“ [online], cit. 14. led. 2026. URL: https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html.

[5] NPX, *I^2C-bus specification and user manual*, High Tech Campus 60, 5656 AG Eindhoven, the Netherlands, 2021.

[6] M. Úbl. „KIV-RTOS - An educational operating system for bare-metal Raspberry Pi Zero W (BCM2835-based board).“ [online], cit. 12. led. 2026. URL: https://github.com/MartinUbl/KIV-RTOS.

[7] Microchip. „Learn Ethernet Products.“ [online], cit. 14. led. 2026. URL: https://developerhelp.microchip.com/xwiki/bin/view/products/interface-connectivity/ethernet.

[8] TEAM a A. Editorial. „The Official History of Arm.“ [online], cit. 12. led. 2026. URL: https://newsroom.arm.com/arm-official-history.

[9] J. Šilhavý a Ú. Martin, „ARMv6 Processor Emulator for Raspberry Pi Environment Emulation,“ dipl. pr., University of West Bohemia, 2024.