VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2019 Bc. Michal Ciprys

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SYSTÉM PRO SBĚR DAT S RASPBERRY PI
SYSTEM FOR DATA ACQUIRE WITH RASPBERRY PI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Michal Ciprys

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Tomáš Macho, Ph.D.

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Michal Ciprys ***ID:*** 76534
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Systém pro sběr dat s Raspberry Pi

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s jednodeskovým mikropočítačem Raspberry Pi a OS Raspbian.
2. Vyhledejte vhodné databázové systémy a webové servery, které lze provozovat pod OS Raspbian. Jednotlivé databázové systémy a webové servery porovnejte. Do databáze by měly být ukládány naměřená data a webový server by měl umožňovat data zobrazovat uživateli.
3. Rozeberte možnosti připojení A/D převodníku k mikropočítači Raspberry Pi. Uvažujte 10-ti a 12-ti bitové převodníky a možnost sběru data z 8 analogových kanálů. Porovnejte jednotlivá řešení zejména z hlediska rychlosti sběru dat.
4. Systém pro sběr dat implementujte. Řešte zabezpečení přístupu k datům přes webové rozhraní.
5. Ověřte funkčnost systému a diskutujte dosažené výsledky.

DOPORUČENÁ LITERATURA:

Raspberry Pi 2. RASPI [online]. 2015 [cit. 2015-12-30]. Dostupné z:

http://www.raspi.cz/2015/02/raspberry-pi-2-nova-turbomalina/

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 13.5.2019

Vedoucí práce: Ing. Tomáš Macho, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

UPOZORNĚNÍ:
Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Tato práce se zabývá sběrem dat z analogových senzorů, jejich ukládání a zobrazováním pomocí mikropočítače Raspberry Pi. Podrobněji se zabývá výběrem vhodného analogově digitálního převodníku, výběrem vhodného úložiště a databázového serveru, webového serveru a aplikace zajišťující zobrazení naměřených dat.

KLÍČOVÁ SLOVA

Sběr dat, analogově digitální převodník, Raspberry Pi, databáze, webový server

ABSTRACT

This work deals with the collection of data from analog sensors, their storage and display using the Raspberry Pi microcomputer. In more detail it deals with selecting the appropriate analog-to-digital converter, selecting the appropriate storage and database server, web server and application to display the measured data.

KEYWORDS

Data collection, analog-to-digital converter, Raspberry Pi, Database, web server

CIPRYS, Michal. *Systém pro sběr dat s Raspberry Pi*. Brno, 2019, 76 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Tomáš Macho, Ph.D.

Vysázeno pomocí balíčku `thesis` verze 2.63; `http://latex.feec.vutbr.cz`

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Systém pro sběr dat s Raspberry Pi“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Tomáši Macho, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V praxi je řada aplikací ve kterých je potřeba sbírat naměřená data. Jedná se například o sběr dat o využívání energií, informace o čistotě vzduchu, informace průtoku řek nebo sledování teploty teplotně citlivých potravin během přepravy. Tato data je nutné automatizovaně zaznamenávat a zpřístupňovat k další analýze.

Existuje mnoho zařízení určených k zaznamenávání hodnot, a to od jednokanálových jednoúčelových (např. určených k monitorování teploty), které se pro naše účely nehodí, protože nemáme zadané měřené veličiny, po vícekanálové univerzální, které je potřeba osadit požadovanými senzory.

Cílem této práce je sestrojit zařízení s použitím jednodeskového mikropočítače Raspberry Pi, které bude umožňovat sběr dat, ukládat data do databáze a zobrazovat data ve webovém prohlížeči.

1 ROZBOR ZADÁNÍ

Cílem je navrhnout a realizovat zařízení umožňující sběr analogových dat na minimálně osmi kanálech, ukládání dat do databáze a prezentace naměřených dat uživateli prostřednictvím webového rozhraní. Je požadována přesnost měření s rozlišením 10 až 12 bitů. Předpokládá se vstupní rozsah měření vstupních signálů řádově jednotky voltů. Předpokládá se také maximální rychlost vzorkování desítky vzorků za sekundu.

Cílem práce bude stanovit maximální možnou rychlost sběru dat s ohledem na ukládání do databáze. Součástí práce také bude ověření maximální rychlosti sběru dat.

Vzhledem k tomu, že zařízení může být připojeno do veřejné sítě, je třeba řešit zabezpečený přístup k datům.

Dle zadání má být zařízení založeno na jednodeskovém počítač Raspberry Pi, který je vybaven 64 bitovým procesorem architektury ARM, který umožňuje použít Operační systém (OS), a dále obsahuje ethernetové rozhraní.

Protože jednodeskový mikropočítač není vybaven Analogově digitální (A/D) převodníky a zadání vyžaduje měření analogových signálů, je třeba systém doplnit A/D převodníkem, protože je požadováno měření minimálně osmi kanálů, je potřeba použít analogový multiplexor.

Data získaná z A/D převodníku je požadováno ukládat do databáze, proto součásti práce musí být i výběr vhodného databázového systému. Při výběru musí být brán ohled i na relativně malou operační paměť mikropočítače (1 GB Random Access Memory (RAM)).

Stejně tak musí být proveden výběr vhodného webového serveru. Předpokládá se připojení malého počtu uživatelů (max. 5), obvykle bude připojen pouze jeden uživatel. Vzhledem k omezeným systémovým zdrojům mikropočítače je třeba vybrat webový server s nízkými nároky na systémové zdroje. Webový server musí podporovat zabezpečené spojení pomocí Hypertext Transfer Protocol Secure (https). Stejně tak musí být řešeno ověřování uživatele.

2 PRŮZKUM TRHU - VYRÁBĚNÁ ZAŘÍZENÍ PRO SBĚR DAT

Na trhu je celá řada zařízení zabývajících se záznamem hodnot ze senzorů. Ve většině případů se jedná o zařízení určené pro jednu měřenou veličinu a s velmi malým počtem měřených kanálů. Dále tato zařízení jsou často uzpůsobena pro provoz v měřeném prostředí (odolnost proti vlhkosti, nedostupnost napájení). V tab. č. 2.1 je porovnání několika hotových produktů nalezených na internetu.

Výrobce	Pico Technology [1]	Sensus Metering Systems [2]	COMET SYSTEM, s.r.o. [3]
Produkt	ADC-24	CDL - 4U (obr. č. 2.1)	LOGGER S6021
Počet kanálů	8 diferenční / 16 jednokoncový	4 (analogové / impulsní)	2
Vzorkovací frekvence	až 60ms	0,1s. ... 1den	10s - 24h (20 možností)
Velikost A/D převodníku	24bitů	12bitů	7 900kroků
Rozsah	±2500, ±1250, ±625, ±312, ±156, ±78, ±39mV	Pouze snímače výrobce	0-20mA ss
Připojovací rozhraní	USB	RS 232	RS232, USB
Cena [Kč]	7 000	30 250	5 965

Tab. 2.1: Srovnávací tabulka dostupných zařízení.

Z tab. č. 2.1 je zřejmé, že hotové zařízení, které by splňovalo zadání této práce by bylo finančně náročné a nenabízelo by pokročilé možnosti konfigurace.

Obr. 2.1: Sensus Metering Systems CDL - 4U. [2]

3 JEDNODESKOVÝ MIKROPOČÍTAČ RASPBERRY PI

Raspberry Pi je minipočítač původně zkonstruovaný pro výuku do škol, ale vzhledem k nízké spotřebě elektrické energie a mnoha možnostem připojení komponentů je oblíben pro provozování nenáročných aplikací. Např. webové a souborové servery, přehrávače video obsahu a hudby a v neposlední řadě jednoduché řízení a regulace. Verze Raspberry Pi 3 model B (obr. č. 3.1) již disponuje dostatečným výkonem pro využití jako domácí počítač sloužící převážně k prohlížení internetu.

Obr. 3.1: Raspberry Pi 3 Model B. [4]

Datum vydání	Únor 2016
SoC	Broadcom BCM2837 (CPU, GPU, DSP, SDRAM, jeden USB port)
Procesor (CPU)	1.2GHz 64-bitový čtyřjádrový ARM Cortex-A53
Video (GPU)	Broadcom VideoCore IV @ 400MHz / 300MHz, OpenGL ES 2.0 (24GFLOPS), MPEG-2 a VC-1 (s licencí), 1080p30 H.264/MPEG-4 AVC dekodér a kodér s vysokým profilem
Paměť (SDRAM)	1GB (sdílená s GPU)
USB 2.0 porty	4 (přes zabudovaný pětiportový USB hub; jeden USB port vnitřně propojen s ethernet portem)
Video výstup	HDMI (rev 1.3 & 1.4), 14 HDMI rozlišení od 640×350 do 1920×1200 plus různé PAL a NTSC standardy, kompozitní video (PAL a NTSC) via 3.5mm TRRS jack sdílený s výstupem zvuku
Video vstup	15-pinový MIPI konektor kamerového rozhraní (CSI)
Zvukový výstup	Analogový via 3,5mm jack; digitální via HDMI a I^2C
Zvukový vstup	Prostřednictvím I^2C rozhraní GPIO
Integrovaná síť	10/100Mbit/s Ethernet + WiFi a Bluetooth
Nízkoúrovňové periferie	17 x GPIO plus tytéž specifické funkce a HAT ID sběrnice
Interní paměť	MicroSDHC slot
Rozměry	85,60mm × 56,5mm (bez konektorů)
Váha	45g

Tab. 3.1: Specifikace Raspberry Pi 3 Model B 64-bit 1GB RAM. [4]

3.1 Podporované sběrnice

Jak je ze specifikace popsané v kapitole č. 3 zřejmé, tak Raspberry Pi 3 Model B nedisponuje vestavěným A/D převodníkem. Z tohoto důvodu je nutné použít externí

moduly nebo čipy, které komunikují na sběrnici Inter-Integrated Circuit (I^2C) nebo po sběrnici Serial Peripheral Interface (SPI).

Tyto sběrnice jsou připojeny na vývody general-purpose input/output (GPIO), které přímo připojíme k našemu zařízení. Jednotlivé vývody je možné adresovat podle umístnění na desce např. pin č. 7 nebo označením výstupu z procesoru se předchozí pin označuje jako `GPIO4`. Označení jednotlivých výstupů je možné zjistit zadáním příkazu `pinout` do terminálu Raspberry Pi (tab. č. 3.2).

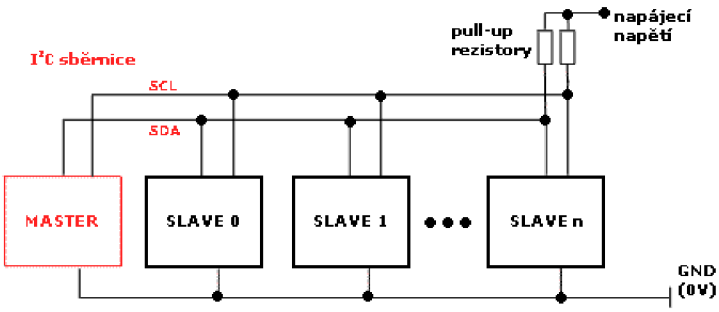
Označení výstupu z procesoru	Čísla vývodů GPIO		Označení výstupu z procesoru
2V3	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3V3	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
GPIO0	27	28	GPIO1
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Tab. 3.2: Označení GPIO výstupů Raspberry Pi.

3.1.1 Sběrnice I^2C

Sběrnice I^2C je často využívána ke komunikaci modulů určených pro Raspberry Pi. Tato sběrnice využívá dvou datových vodičů, ale v praxi je nutné dalším vodičem propojit sdílenou zem. Vodič označovaný Serial Data Line (SDA) slouží k odesílání a přijímání dat a vodič označený Serial Clock Line (SCL) slouží k odesílání synchronizačního hodinového signálu. Synchronizační signál odesílá výhradně master zařízení

a všechna slave zařízení se tímto signálem synchronizují. Blokové schéma zapojení sběrnice je na obrázku č. 3.2. Velikosti pull-up rezistorů R_p se stanovují s ohledem na komunikační frekvenci.

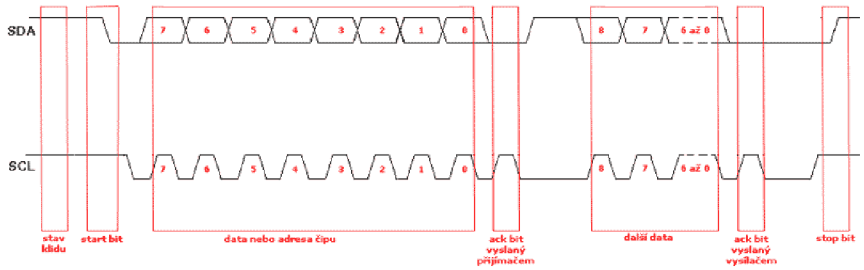


Obr. 3.2: Blokové schema zapojení sběrnice I^2C. [13]

Každé slave zařízení musí mít unikátní 7-mi bitovou nebo 10-ti bitovou adresu. Tato adresa se dá buďto nastavit při zapínání sběrnice, nebo u některých zařízení, většinou u čipů, je možné propojením různých vstupů zvolit z několika přednastavených adres. Některá jednodušší zařízení neumožňují měnit adresu vůbec, a proto je provoz více těchto zařízení na stejné I^2C sběrnici vyloučen.

V klidovém režimu jsou na vodičích SDA i SCL logické jedničky. Pokud chce master zařízení komunikovat, tak na vodiči SDA přepne logickou nulu a na SCL ponechá logickou jedničku. Tomuto signálu se říká start bit a doba ponechání logické jedničky je závislá na přenosové rychlosti. Po tomto start bitu následuje sedmibitová adresa a poté následuje bit určující zda master zařízení odesílá data na slave zařízení (zápis - logická 0), nebo data přijímá (čtení - logická 1). Po přečtení těchto osmi bitů všechna zařízení na síti provedou porovnání s vlastní nastavenou adresou a zařízení se stejnou adresou jako je požadovaná (pokud existuje a je připraveno přijímat data) přepne signál SDA do logické 0. Tomuto poslednímu bitu se říká potvrzovací bit. Po navázání komunikace zařízení nastavené jako odesílací, začne odesílat požadovaná data a zařízení nastavené pro příjem dat začne data přijímat. Po každých osmi přijatých bitech zařízení, které přijímalo data potvrdí jejich příjem zasláním potvrzovacího bitu. Pokud nedojde k odeslání potvrzovacího bitu, tak je možné ukončit komunikaci zasláním stop bitu ve formě změny signálu na vodiči SDA z logické 0 do logické 1, přičemž signál na SCL je v hodnotě logické 1.

Adresace sedmibitovou adresou má nevýhodu v tom, že umožňuje na sběrnici provozovat pouze necelých 128 zařízení (některé adresy mají speciální význam). Proto je možné použít desetibytovou verzi adresy, která umožňuje připojit až 1024 zařízení. Tato adresa se posílá ve dvou bajtech, přičemž v prvním bajtu se přenáší dva nejvyšší bity. Tento bajt má tvar `11110??0` (symboly `??` znamenají přenášené bity



Obr. 3.3: Schema průběhu komunikace po sběrnici I^{2}C při použití sedmibitové adresy. [13]

adresy) a zároveň zařízení slave přepíná posledním bitem do režimu čtení. V dalším bajtu je přenášeno ostatních osm bitů adresy. Pokud se posílají data z master zařízení do slave zařízení, tak jsou přímo odeslána dříve popsaným způsobem. Pokud se mají data posílat ze slave zařízení do master zařízení, tak je nutné resetovat spojení zasláním start bitu a prvním bajtem jako při navazování spojení, ale tento bajt má nyní na poslední pozici hodnotu logickou 1 pro přepnutí do režimu čtení. Adresaci pomocí desetibitové adresy nepodporují všechna zařízení.

Raspberry Pi obsahuje dvě sběrnice I^{2}C. První sběrnice je dostupná na GPIO pinech `GPIO2` pro SDA a `GPIO3` pro SCL. Druhá sběrnice je určena pro programování pamětí Electrically Erasable Programmable Read-Only Memory (EEPROM) a tato sběrnice je dostupná na GPIO pinech `GPIO0` pro SDA a `GPIO1` pro SCL.

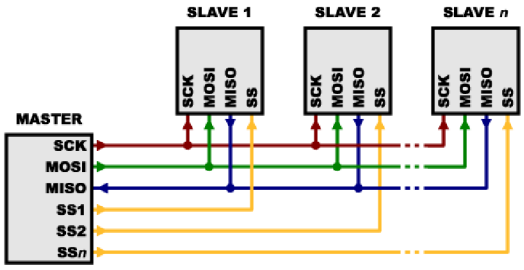
Podporu sběrnice v OS Raspbian můžeme ručně zapnout zadáním příkazu `sudo raspi-config` do terminálu, dále ve spuštěném programu zvolíme pátou hodnotu s názvem `5 Interfacing Options`, potom vybereme volbu `P5 I2C` a na všechny následující dotazy odpovíme `yes`. Nakonec program ukončíme zadáním volby `<Finish>` a na žádost programu zařízení restartujeme. Výše popsané kroky lze provést i přímou editací konfiguračních voleb souboru `/boot/config.txt`, čehož můžeme využít při automatizovaném zapnutí sběrnice I^{2}C. Po restartu zařízení by již měla být sběrnice I^{2}C dostupná jako blokové zařízení a v jádře by měl být zaveden příslušný modul umožňující její využití. Přítomnost blokového zařízení si můžeme ověřit zadáním příkazu `ls /dev/*i2c*` do terminálu.

K ověření správné funkčnosti sběrnice můžeme využít nástroje z instalačního balíku s názvem `i2c-tools`, který si musíme v případě potřeby doinstalovat. Obzvláště užitečný nástroj je `i2cdetect`, který nám projde všechny adresy zvolené sběrnice a vypíše nám ty, které jsou obsazené nějakým zařízením. Tímto způsobem můžeme detekovat připojená zařízení u kterých neznáme adresu.

Sběrnice I^{2}C je v programovacích jazycích podporována formou knihoven. V jazyce C++ můžeme použít knihovnu `wiringPiI2C.h` a v jazyce Python `smbus`.

3.1.2 Sběrnice SPI

Sběrnici SPI často využívají integrované obvody pro svou jednodušší implementaci než sběrnice I^2C. Tato sběrnice umožňuje propojit jedno master zařízení s několika slave zařazeními. K propojení využívá vodič Serial Clock (SCK), který přenáší hodinový signál generovaný master zařízením a dále vodič Master Out, Slave In (MOSI), který přenáší data z master zařízení do slave zařízení a vodič Master In, Slave Out (MISO), který přenáší data v opačném směru. Vzhledem k tomu, že sběrnice SPI využívá oddělené vodiče pro odesílání a přijímání dat, tak je možný ve stejnou chvíli obousměrný provoz. Master zařízení provádí volbu slave zařízení vodičem Slave Select (SS). Pro každé slave zařízení je proto nutný zvláštní vodič (obr. č. 3.4).

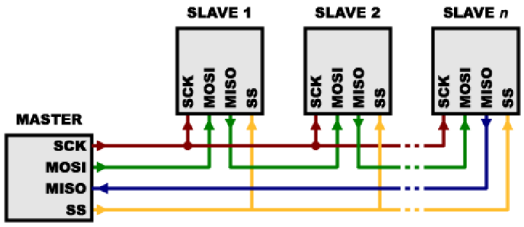


Obr. 3.4: Blokové schema zapojení sběrnice SPI. [14]

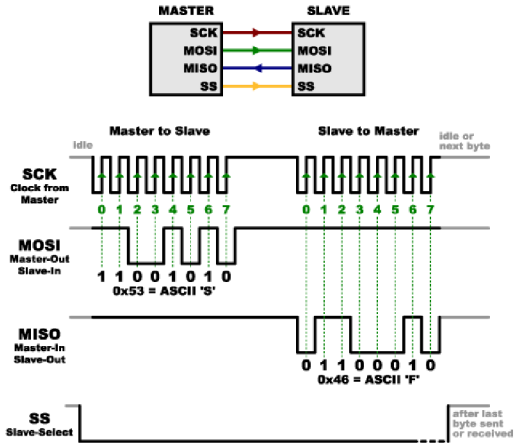
Sběrnice SPI umožňuje i jiné zapojení a to zapojení zřetězené (obr. č. 3.5). V tomto zapojení je hodinový signál SCK a signál volby slave zařízení SS přiveden do všech zařízení paralelně, ovšem signály MOSI a MISO jsou za sebou zřetězeny tak, aby výstup z jednoho zařízení byl zapojen do vstupu druhého zařízení. Takto jsou vodiče zapojeny do kruhu a tvoří posuvný registr. Pokud by žádné zařízení data nezměnilo, tak by se po $8 \times n$ cyklech stejná data vrátila zpět do master zařízení. Toto propojení má nevýhodu v tom, že při odpojení libovolného slave zařízení dojde k přerušení datového okruhu a výpadku celé sběrnice. Další nevýhodou tohoto zapojení je nutnost adresace zařízení a tím snížení množství přenášených bitů. Výhodou je snížení počtu vodičů SS.

Komunikace po sběrnici je možné konfigurovat dle potřeby, proto mnoho zařízení obsahuje konfigurační register, kterým je možné komunikaci nastavit. Je možné nastavit, zda bude hodinový signál v klidovém režimu v logické hodnotě 0 nebo 1, nebo zda bude aktivace čtení bitů probíhat na sestupnou nebo vzestupnou hranu hodinového signálu. Na obr. č. 3.6 je zobrazena komunikace, která probíhá na vzestupnou hranu při klidové hodnotě signálu SCK v logické 1.

Raspberry Pi obsahuje dvě sběrnice SPI. První sběrnice je dostupná na GPIO pinech `GPIO10` pro MOSI, `GPIO9` pro MISO, `GPIO11` pro SCK, `GPIO8` pro SS0 a



Obr. 3.5: Blokové schema zřetězeného zapojení sběrnice SPI. [14]



Obr. 3.6: Schema průběhu komunikace po sběrnici SPI. [14]

`GPIO7` pro SS1. Druhá sběrnice je dostupná na GPIO pinech `GPIO20` pro MOSI, `GPIO19` pro MISO, `GPIO21` pro SCL, `GPIO18` pro SS0, `GPIO17` pro SS1 a `GPIO16` pro SS2.

Podporu sběrnice v systému můžeme zapnout podobně jako sběrnici I^2C pouze s rozdílem, že ve druhém menu vybereme volbu `P4 SPI` a ověření přítomnosti blokového zařízení provedeme příkazem `ls /dev/*spi*` zadaným do terminálu.

Podporu sběrnice SPI v jazyce C++ získáme použitím knihovny `wiringPiSPI.h`, zatímco v jazyce Python můžeme použít knihovnu `spidev`.

3.2 Operační systém

Jednodeskový počítač Raspberry Pi není vybaven žádným operačním systémem nebo firmwarem určeným ke spuštění po startu zařízení, a proto je nutné tento systém nejdříve nainstalovat. Instalace systému se provádí zkopírováním obrazu disku OS staženého z internetu na Micro Secure Digital (MicroSD) kartu, která se vkládá do příslušného slotu na zařízení.

Na oficiálních stránkách projektu Raspberry Pi jsou k dispozici podporované operační systémy i s instalačními obrazy. Instalace se nejsnadněji provádí pomocí speciálního OS s názvem NOOBS, který se instaluje jednoduchým zkopírováním stažených souborů do hlavní složky MicroSD karty zformátované na souborový systém FAT32. Tento speciální OS nám po spuštění umožní snadný výběr, automatické stažení z internetu a snadnou instalaci několika vybraných systémů.

Na zařízení Raspberry Pi můžeme provozovat univerzální OS Raspbian založený na známé a oblíbené distribuci Debian vytvořené v roce 1993. Pro tuto distribuci existuje velké množství aplikací a návodů jak aplikace nastavit. Podle potřeby je možné tuto distribuci nainstalovat s grafickým prostředím pro jednodušší ovládání, ale pro serverové a automatizační využití je lepší použít verzi bez grafického prostředí, které je méně náročné na hardwarové prostředky.

Dalším zajímavým systémem je OS Pidora. Tento systém vychází z linuxové distribuce Fedora, a je vyvíjený společností Red Hat, Inc. Systém je svojí kvalitou velmi podobný distribuci Raspbian, ale jeho nevýhodou je absence v instalátoru NOOBS.

Pokud plánujeme využít Raspberry Pi jako stolní počítač můžeme použít kromě dříve popsaného Raspbianu i OS Ubuntu MATE. Tento OS je oblíben u začínajících uživatelů OS Linux, ale pro nasazení v systému pro sběr dat se nehodí zejména z důvodu nároků na systémové prostředky.

Dalším OS, který můžeme nainstalovat je RISC OS. Tento OS nevychází z žádné linuxové distribuce, proto se na měj nedají nainstalovat klasické softwarové balíčky. Toto bohužel negativně ovlivňuje množství dostupného softwaru k instalaci.

OS Windows 10 IoT Core je sestaven pro použití jako zařízení „internet věcí“ a nemá nainstalované grafické prostředí a proto pokud vyžadujeme grafické prostředí např. na displeji, tak si jej musíme vytvořit sami. K vývoji je nutné použít další nástroje z dílny společnosti Microsoft, jedná se zejména o nástroj Visual Studio pro vytváření programů. Správa zařízení se provádí pomocí webového rozhraní, nebo pomocí PowerShellu.

Dalším způsobem využití zařízení Raspberry Pi je jako multimediální centrum. Z této skupiny můžeme jmenovat systém OSMC nebo systém LIBREELEC, který je nástupce známého multimediálního centra KODI. Tyto systémy se vyznačují zaměřením na zobrazování multimediálních souborů a podobají se více chytrým televizím, než systémům určeným pro počítače.

Více informací o podporovaných OS na zařízeních Raspberry Pi je možné získat na webové stránce. [5]

Samotný OS je uložen na vložené MicroSD kartě, a proto musí být značně optimalizovaný nejen na rychlost, ale i velikost. Doporučená minimální velikost MicroSD karty je 8 GB.

Jako nejvhodnější z hlediska účelu a využití systémových prostředků je OS Raspbian bez nainstalovaného grafického prostředí. Dále doporučuji na systém nainstalovat službu Secure Shell (SSH), která nám umožní správu zařízení po síti bez nutnosti fyzického přístupu k zařízení.

4 SBĚR ANALOGOVÝCH HODNOT

A/D převodníky převádí analogovou spojitou hodnotu veličiny na hodnotu digitální tzn. hodnotu reprezentovanou číslem v daném okamžiku. Spojitý signál je nutné nejdříve na vzorkovat. Nejvyšší vzorkovací frekvence měřené veličiny se volí nejméně dvojnásobná proti nejvyšší frekvenci analogového signálu. Důležitou vlastností A/D převodníku je doba převodu, která nám stanovuje maximální vzorkovací frekvenci. Nakonec je potřeba navzorkovanou hodnotu správně kvantizovat a výstup kvantizace předat na výstup A/D převodníku. Kvantizace je velmi důležitá pro přesnost A/D převodníku a udává se v bitech. Např. 10-ti bitový A/D převodník rozlišuje $2^{10} = 1024$ úrovní, což při referenčním napětí 3,3V odpovídá rozlišení 0,003V na jednu číselnou úroveň. Výsledná naměřená hodnota je předána záznamovému zařízení předem známým způsobem (sběrnice SPI, I^2C, ...).

K dispozici máme následující typy převodníků:

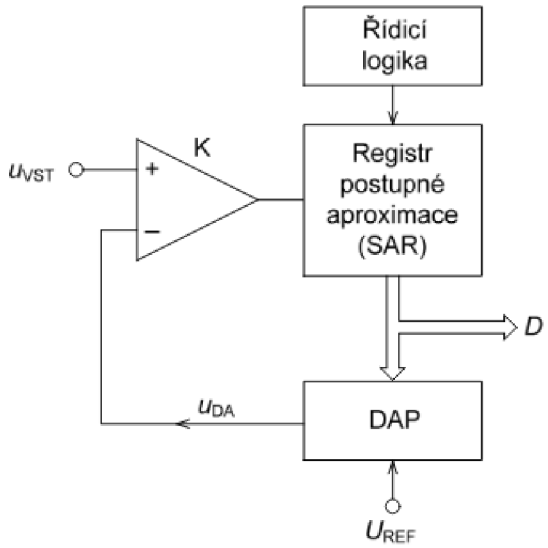
Paralelní A/D převodník je nejrychlejším typem A/D převodníků, protože celý převod trvá pouze jediný krok. Ovšem velikou nevýhodou je velká konstrukční náročnost. Pro každý převodník potřebujeme $2^n - 1$ komparátoru a 2^n vysoce přesných rezistorů (tj. pro 10-ti bitový převodník 1023 komparátorů a 1024 rezistorů). Z tohoto důvodu se tyto A/D převodníky obvykle vyrábí s menším počtem bitů než je námi požadovaný a bývají velmi drahé. Pro naši aplikaci se příliš nehodí zejména z důvodu vysoké ceny a rychlosti, kterou v našem zařízení nevyužijeme.

Kaskádní A/D převodník tento převodník, který bývá někdy nazýván A/D převodník s postupnou aproximací se používá ke snížení počtu komparátorů u paralelního převodníku. Principem je zapojení několika paralelních A/D převodníků za sebe, kdy výstup z prvního převodníku odpovídá nejvýznamnějším bitům, tento výstup je dále pomocí Digitálně analogový (D/A) převodníku převeden zpět na analogový signál, který je odečten od vstupního napětí a předán dalšímu A/D převodníku k převedení hodnoty. Tento postup se opakuje až k poslednímu A/D převodníku, který odpovídá nejméně významným bitům. Vzhledem k tomu, že měření trvá delší dobu než u paralelního A/D převodníku, tak se dopouštíme různých chyb měření, které musíme kompenzovat. Tento převodník se pro naši aplikaci příliš nehodí ze stejných důvodů jako paralelní A/D převodník.

Integrační A/D převodníky využívají integrátor s definovanou počáteční hodnotou k převodu na časový interval pomocí kmitočtu nebo pomocí šířky pulzu a tuto hodnotu dále přepočítávají na námi měřené vstupní napětí. Buďto můžeme použít A/D převodník s dvojsklonnou integrací, nebo můžeme použít A/D převodník s přepočtem na kmitočet. Využití těchto A/D převodníků je

nejčastěji pro svoji přesnost v laboratorních přístrojích, ale pro použití v naší aplikaci se nehodí zejména z důvodu vyšší ceny.

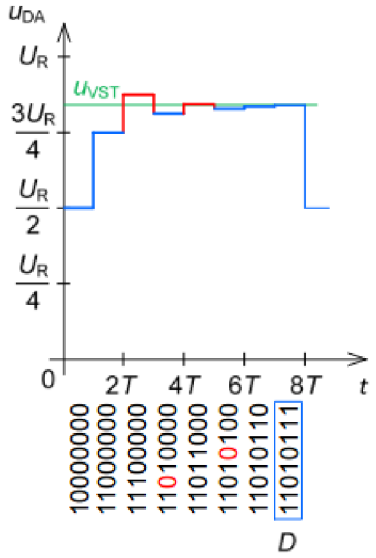
Kompenzační A/D převodník s postupnou aproximací pracuje na principu porovnávání vstupní hodnoty s hodnotou generovanou D/A převodníkem (obr. č. 4.1). Při převodu se nastavuje bit s nejvyšším významem na logickou 1 a výstup z D/A převodníku se porovnává se vstupním napětím u_{VST}. Pokud je u_{VST} vyšší než výstupní hodnota, tak se ponechává logická 1 a pokud je nižší, tak se bitu vrátí původní hodnota logická 0. Dále se provádí předchozí kroky s dalšími významovými bity až po nejméně významný bit. Průběh převodu s ukázkou nastavování bitů je zobrazen na obr. č. 4.2. Z popisu je zřejmé, že převod je hotový po `n` krocích, kde `n` odpovídá počtu bitů A/D převodníku. Tento A/D převodník je pro naši aplikaci nejvhodnější zejména z důvodu nízké ceny, dostatečné přesnosti a dostatečné rychlosti převodu.



Obr. 4.1: Schema kompenzačního A/D převodníku s postupnou aproximací. [15]

4.1 Připojovací rozhraní A/D převodníků

A/D převodníky mají ve svém pouzdru implementováno rozhraní pro komunikaci s okolními elektronickými obvody. Převodníky určené pro komunikaci s mikrokontrolérem implementují rozhraní SPI nebo I^2C, ale některá pouzdra jsou vybavena přímo výstupem pro sedmisegmentový displej (často zobrazují s přesností 3,5 DIG). Nejjednodušším typem komunikace je řádek led diod, které se rozsvěcují v množství dle změřené hodnoty (např. čip LM3914 s analogovým vstupem 0-5 V a deseti diodami rozlišuje pouze deset nenulových úrovní).



Obr. 4.2: Průběh převodu kompenzačního A/D převodníku s postupnou aproximací. [15]

Pro naše potřeby je nutné vybírat z A/D převodníků implementujících rozhraní pro sběrnice SPI a I^2C.

4.2 A/D převodníky dostupné na trhu

Na českém trhu je celá řada A/D převodníků, nebo modulů pro Raspberry Pi obsahujících A/D převodník, které je možné připojit k Raspberry Pi.

Například modul A/D převodníku ADC Pi je osazen dvěma čipy MCP3424 (každý může převádět 4 analogové kanály, každý má svoji vlastní adresu na sběrnici I^2C a jejich adresy je možné nastavit na osm přednastavených pomocí propojovacích jumperů).

V tab. č. 4.1 je uveden základní výčet A/D převodníků. Z tabulky je zřejmé, že osmikanálové převodníky se vyrábí nejčastěji ve 12-ti bitovém provedení. Do tabulky jsem zařadil i hotové moduly k Raspberry Pi.

Pro použití jsem vybral A/D převodník MCP3208-BI/P DIP16 MICROCHIP, který má dostatečnou vzorkovací frekvenci, splňuje podmínku přesnosti a je dostupný na českém trhu za rozumnou cenu. Jedná se o osmikanálový 12-ti bitový kompenzační A/D převodník s postupnou aproximací a rozhraním SPI. A/D převodník dokáže měřit i rozdílné napětí mezi dvěma vývody tzv. diferenciální zapojení, které je možné softwarově zapnout. Převodník MCP3208 disponuje čtyřmi diferenciálními kanály. Udávaná vzorkovací frekvence je 50 kSPS při použití napájecího

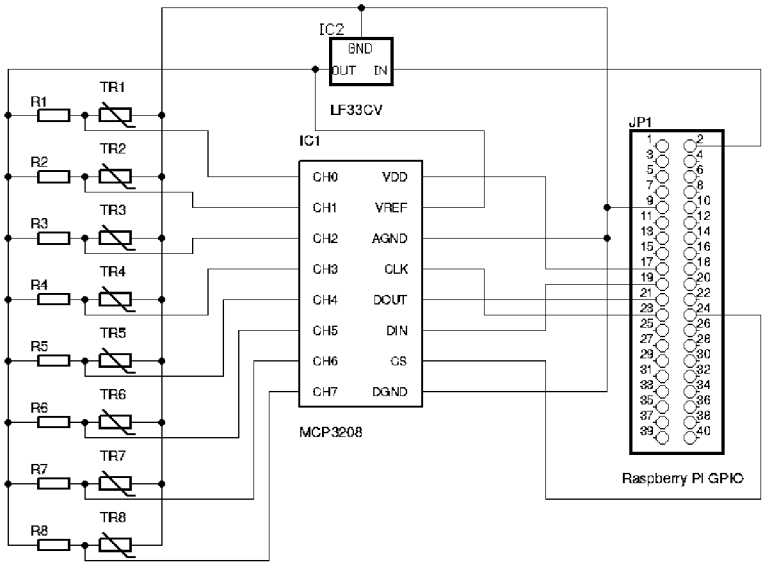
Produkt	Vzorkovací frekvence [SPS]	Velikost A/D převodníku [bit]	Maximální napětí [V]	Připojovací rozhraní	Cena [Kč]
ADC Pi Plus Kit [6]	60	13	5	I^2C	666
ADC Differential Pi Kit [7]	240	12	2,048	I^2C	666
ADC Pi [18]	240	11	5,06	I^2C	518
MCP3208-BI/P DIP16 [16]	50 k	12	5,5	SPI	254
AD7997BRUZ-0 TSSOP20 [17]	188 k	10	5,5	I^2C	69
AD7298BCPZ [19]	1 M	12	3,6	SPI	220
MAX147BCPP+ [19]	133 k	12	5,25	SPI	508
ADS7844E [19]	200 k	12	5,25	SPI	234
AD7888ARZ [19]	125 k	12	5,25	SPI	260

Tab. 4.1: Srovnávací tabulka osmi kanálových A/D převodníků kompatibilních s Raspberry Pi a dostupných na českém trhu.

napěti 2,7 V. Čip A/D převodníku je napájen 3,3 V přímo z Raspberry Pi a je nutné toto napětí dodržet z důvodu 3,3 V logiky Raspberry Pi. Tato rychlost je dostačující s ohledem na rychlost zápisu do databázového systému změřené v kapitole č. 5.1. Dalším důvodem výběru tohoto převodníku bylo jeho pouzdro typu DIP16 určené k montáži typu Through-hole technology (THT), která umožňuje snadnou výrobu plošného spoje.

4.3 Schema připojení A/D převodníku k Raspberry Pi

Schema připojení A/D převodníku MCP3208 k Raspberry Pi je zobrazeno na obr. č. 4.3. K připojení je využita sběrnice SPI a napájecí napětí 3,3 V. Pro testovací účely jsou ve schematu připojeny termistory pro měření teploty. Tyto termistory jsou připojeny ke stabilizovanému napětí pomocí lineárního regulátoru napětí LF33CV TO220 STM/HOMPSON, který má na výstupu napětí 3,3 V a maximální výstupní proud 0,5 A. Tento regulátor napětí byl vybrán s ohledem na dostatečný výkon i při použití výkonově náročnějších aplikací. Regulátor napětí je připojen ke GPIO pin č. 2, který je připojen k napájení Raspberry Pi 5 V a může, s ohledem na výkon zdroje napájení, dodávat proud až 1,5 A.



Obr. 4.3: Připojení A/D převodníku k Raspberry Pi a teplotních čidel k jednotlivým kanálům.

5 SOFTWAROVÉ ZPRACOVÁNÍ DAT

Další důležitou částí realizace systému pro sběr dat je získávání hodnot z A/D převodníku, jejich ukládání do databáze a následné zobrazování dle požadavku uživatele.

Prvním krokem v této kapitole bude vybrat vhodný databázový systém a webový server, dále si stanovíme způsob instalace potřebných programů a knihoven a jejich konfigurace. Také si stanovíme konfigurační parametry důležité pro software pro ukládání dat do databáze a aplikaci zajišťující zobrazování dat uživateli.

Druhým krokem bude vytvořit software pro ukládání dat do databáze. K dispozici mám hodnotu změřenou A/D převodníkem. Tuto hodnotu je potřeba zpřístupnit v programovacím prostředí a požadavkem je ukládat změřené hodnoty do databázové tabulky, jejichž první hodnota bude časová značka vytvoření záznamu a následovaná bude osmi hodnotami odpovídajícími měřené veličině.

Třetím krokem této části bude zobrazování dat uživateli. Pro vyřešení tohoto kroku již musíme mít vyřešen předchozí krok s ukládáním dat do databáze, což bude vstupem tohoto kroku a výstupem bude graf získaný z měřených hodnot, tabulka hodnot odpovídající grafu a možnost stáhnutí dat do počítače uživatele. Je požadována uživatelská volba časového intervalu, který se bude uživateli zobrazovat. Dále je potřeba provést omezení hodnot zobrazených v grafu z důvodu přehlednosti při volbě intervalu s velkým počtem vzorků. Tyto vybrané vzorky musí být rovnoměrně rozprostřeny v celém požadovaném intervalu.

5.1 Databázové systémy

K ukládání strukturovaných dat, jejich filtrování a vypisování je nejvhodnější použít databázový systém, tj. program, kterému předáme naše data do úschovy, ten si je uloží do souboru, označí je pro jednodušší a rychlejší vyhledávání a dále nám vystupuje jako prostředník při práci s těmito daty. Databázové programy jsou optimalizovány pro práci s velkým množstvím dat, pro rychlost práce s daty (zejména vyhledávání) a pro co nejnižší zatížení systému.

Pro práci s databázemi na bázi tabulek se používá jazyk Structured Query Language (SQL). Jedná se o jazyk vyvinutý pro dotazování do databáze, aby se co nejvíce podobal přirozenému anglickému jazyku. Mimo samotné dotazování obsahuje příkazy i pro ukládání a změnu dat, správu samotné databáze a práce se strukturou tabulek.

Existují i tzv. non SQL (NoSQL) databáze. Tyto databáze jsou velmi škálovatelné a většinou mají jinou strukturu než tabulku (např. typ klíč - hodnota). Tyto

databáze se pro naše požadavky nehodí, protože budeme ukládat vysoce strukturovaná data, která můžeme vyhledávat i podle jiných hodnot, než je klíč.

Základní seznam databázových programů:

MarriaDB - tato databáze je známá pod názvem MySQL, ale z důvodu prodeje společnosti Oracle Corporation a změně v politice vydávání nových verzí se skupina původních vývojářů rozhodla odštěpit a na poslední veřejné verzi postavit nový databázový systém pojmenovaný MarriaDB. Nynější verze MarriaDB je mnohem aktivněji vyvíjená než současná verze MySQL. Jedná se o klasickou databázi často používanou webovými aplikacemi.

SQLite - tato databáze bývá označována za odlehčenou databázi, ale pro své jednoduché nastavení bývá často používána. V podstatě se jedná pouze o sadu knihoven a modulů přepsanou do různých programovacích jazyků a zkompilovanou pro různé operační systémy. Tyto moduly a knihovny není vždy nutné instalovat do systému, ale můžou být k programu pouze přiloženy nebo pro rychlost dokonce do programu přímo vloženy. Pomocí těchto knihoven se připojujeme přímo k souboru s daty (databáze), a dále pracujeme s tabulkami a daty pomocí jazyka SQL. Odlehčení spočívá v tom, že nemáme spuštěný program (službu), který by zpracovával naše požadavky, ale změny si provádíme sami pomocí knihovny. Dále nejsou v rámci SQLite řešeny oprávnění pro přístup k datům. Tyto oprávnění jsou řešeny na úrovni operačního systému. Databázový systém SQLite používá celá řada aplikací z nich nejznámější je Mozilla Firefox.

PostgreSQL - jedná se o obdobný databázový systém jako MarriaDB, ale tato databáze je lépe konfigurovatelná a nastavitelná. Tuto výhodu využívají převážně náročnější aplikace, které s touto databází dosahují lepších výkonů.

Všechny zmíněné databáze mají zveřejněny své zdrojové kódy pod svobodnou licencí, a proto je možné si je mimo jiné upravit přímo pro vlastní potřeby. My využijeme softwarové balíčky ze standardních repositářů distribuce Raspbian.

Protože jsem se rozhodl použít jednu z těchto tří databází, tak bylo nutné provést srovnání rychlosti zápisu do databáze. Předpokládám, že tento parametr bude nejvíce ovlivňovat rychlost zařízení na sběr dat. Za tímto účelem jsem si vytvořil test, který se skládal ze zápisu předpřipravené kolekce vzorku dat do jednotlivých databázových systémů, přičemž jsem měřil dobu zápisu těchto vzorků. Dobu zápisu vzorků jsem jsem měřil pomocí systémového času, kdy jsem si po přípravě databáze a tabulek uložil systémový čas a po provedení testu jsem od aktuálního systémového času odečetl čas začátku testování databáze. Tímto měřením času jsem dosáhl přesnosti měření na deseti tisícinu sekundy, ale rozdíly mezi testy byly v rozmezí desetin sekund až stovek sekund.

Ovladače pro práci s databázemi MarriaDB a SQLite požadují po programu, aby

prováděl ruční ukládání dat do souboru, proto jsem testoval i optimální nastavení počtu vzorků po kterých je vhodné toto ukládání provádět.

Pro tento test jsem si vytvořil skript v jazyce Python (umístěn na přiloženém CD ve složce /test_databazi/test_databases.py.j2). V tomto skriptu jsem si definoval funkce provádějící test jednotlivých databází. Tyto funkce jsou téměř identické a liší se pouze specifiky jednotlivých databázových systémů.

Nejdříve provádí připojení k databázi ovladačem nainstalovaným na zařízení, dále provádí vytvoření testovací tabulky a uložení startovací časové značky. Dále již skript provádí testování databáze tak, že ve smyčce odesílá SQLpříkaz pro uložení záznamu do ovladače databáze a v případě potřeby odesílá ovladači i příkaz k uložení databázového souboru. Po zápisu celé sady testovacích dat skript provede odečtení času začátku měření od aktuálního času a tím získá dobu měření. Po uložení času měření funkce již pouze smaže testovací tabulku, odpojí se od databáze a jako návratovou hodnotu odešle dobu měření zápisu do databáze.

Funkce pro měření databáze PostgreSQL nepřijímá žádné parametry a přesně provádí testování pomocí výše popsaného postupu. Databáze PostgreSQL si ukládání databázového souboru řídí sama.

Funkce pro měření databáze MarriaDB přijímá jako parametr počet vzorků po kterých se má provádět ukládání databázového souboru. Tato funkce je implementovaná pomocí interního počítadla provádějícího počítání vzorků uložených do databáze a využívá vyhodnocení pomocí podmínky.

Funkce pro měření databáze SQLite přijímá jako první parametr počet vzorků po kterých se má provádět ukládání databázového souboru a jako druhý nepovinný parametr cestu k databázovému souboru. Tato funkce se od funkce pro měření databáze MarriaDB liší pouze v možnosti definovat umístění databázového souboru, čehož využijeme při měření ukládání dat na externí úložiště.

Pro účel testování jsem na svém počítači vytvořil kolekci dat tvořenou tabulkou o osmi náhodných číslech v řádku nahrazující osm kanálů A/D převodníku a 10 000 řádcích simulujících počet měření.

Dále jsem si zvolil, že databázové soubory budu ukládat postupně po každém záznamu, dále po stu, pěti stech a tisíci záznamech. Všechny databázové systémy ukládaly soubory na MicroSD kartu se systémem a databázový systém SQLite jsem navíc testoval i zápisem na externí plotnový disk připojený přes rozhraní Universal Serial Bus (USB) 2.0.

Pro instalaci a konfiguraci zařízení Raspberry Pi pro měření rychlostí databází jsem využil instalační nástroj Ansible jehož instalační skript je uložen na přiloženém CD ve složce /test_databazi/databases.yml.

Výsledky ze skriptu jsou zapsány v tab. č. 5.1. Z této tabulky je zřejmé, že množství záznamů po kterých se ukládá databázový soubor velmi ovlivňuje rychlost

zápisu do databáze. Pokud záznamy odesíláme po větších blocích, tak méně vytěžujeme zařízení a zapisujeme rychleji. Databázový server PostgreSQL si řídí ukládání dat vlastním interním mechanismem, který se snaží ukládat data do permanentního úložiště co nejdříve, a proto tento server není pro nasazení v naší aplikaci příliš vhodný. Dále měření prokázalo, že zápis na externí disk bylo výrazně pomalejší, proto se externí disk nehodí pro záznam s vyšší vzorkovací frekvencí, ale při použití nižších vzorkovacích frekvenci pro rychlosti vzorkování pod 0,5 Hz nám externí disk zajistí vyšší spolehlivost datového nosiče.

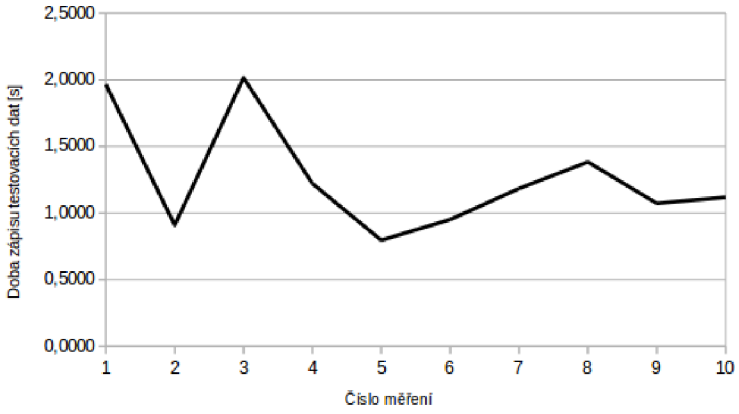
Databázový systém	Počet vzorků, po kterých se ukládá databázový soubor	Průměr doby zápisu 10 000 vzorků [s]	Průměrný počet uložených záznamů za 1 s
PostgreSQL	Interní mechanismus	59,9798	167
SQLite (MicroSD karta)	1	285,7351	35
SQLite (MicroSD karta)	100	4,5311	2207
SQLite (MicroSD karta)	500	1,2618	7925
SQLite (MicroSD karta)	1000	0,8245	12128
SQLite (externi disk)	1	2543,9562	4
SQLite (externi disk)	100	25,0843	399
SQLite (externi disk)	500	5,6655	1765
SQLite (externi disk)	1000	2,9839	3351
MarriaDB	1	143,1533	70
MarriaDB	100	7,4796	1337
MarriaDB	500	6,4938	1540
MarriaDB	1000	6,2642	1596

Tab. 5.1: Srovnání rychlosti zápisu do databáze.

Další věc, kterou nám měření odhalilo, je rozdíl v dobách trvání opakovaných testů se stejnými parametry. Tyto rozdíly jsou zaznamenány v tab. č. 5.2, a pro databázový systém SQLite jsem pro názornost vykreslil i graf (obr. č. 5.1) jednotlivých naměřených hodnot. S touto vlastností musíme při návrhu měřícího obvodu počítat a nechat dostatečně velikou rezervu pro čas potřebný k uložení dat.

Databázový systém	Nastavení proměnné *commit_every*	Minimální doba trvání testu [s]	Maximální doba trvání testu [s]	Rozdíl mezi minimální a maximální dobou trvání testu [s]
PostgreSQL	Interní mechanismus	53,3226	96,4440	43,1214
SQLite (MicroSD karta)	1	274,2748	298,8109	24,5361
SQLite (MicroSD karta)	100	3,9687	5,1357	1,1670
SQLite (MicroSD karta)	500	0,7965	2,0152	1,2187
SQLite (MicroSD karta)	1000	0,6077	1,2964	0,6886
SQLite (externi disk)	1	2291,5926	2626,7715	335,1789
SQLite (externi disk)	100	22,1219	26,6809	4,5590
SQLite (externi disk)	500	5,0326	6,0977	1,0651
SQLite (externi disk)	1000	2,6966	3,3512	0,6546
MarriaDB	1	138,5382	153,5909	15,0527
MarriaDB	100	6,5411	8,1469	1,6057
MarriaDB	500	5,3320	8,2952	2,9631
MarriaDB	1000	5,6607	7,1066	1,4459

Tab. 5.2: Rozdíly v dobách trvání jednotlivých testů databází.



Obr. 5.1: Graf jednotlivých měření zápisu do databáze SQLite na interní MicroSD kartu s ukládáním po 500 záznamech.

Jako nejrychlejší se jeví databázový server SQLite, proto jsem jej vybral pro nasazení v mém zařízení. Server SQLite disponuje teoretickým maximem řádků v tabulce 2^{64} nebo maximální velikostí databázového souboru 140 TB přičemž aplikováno je omezení, která nastane dříve. Další omezení jako třeba na maximální počet sloupců v tabulce (2000) nejsou pro naši aplikaci omezující a jsou k dispozici na manuálových stránkách programu. Další výhodou kromě rychlosti je možnost pracovat přímo s databázovým souborem, a proto nemusíme provozovat, žádnou službu, která by zpomalovala zařízení.

S databází budeme komunikovat pomocí knihoven dostupných pro jednotlivé programovací jazyky.

5.2 Webové servery

Webové servery jsou programy, které odpovídají na Hypertext Transfer Protocol (http) požadavky a vrácením odpovídajícího dokumentu (textového souboru, obrázku nebo jakéhokoli jiného souboru) zadaného adresou. Odpověď serveru může být i generovaná externím programem jehož výstup (tzv. dynamický obsah) je předán místo statického souboru. Ke generování dynamického obsahu se nejčastěji používá skriptovací jazyk Hypertext Preprocessor (PHP), ale může být generován jakýmkoli překládaným jazykem (C, C++, ...) nebo skriptovacím jazykem (Ruby, Python, ...). Webový server musí umět tento program spustit a jeho výstup odeslat uživateli.

Základní seznam webových serverů:

Apache je velmi rozšířený webový server. Tento server je možné rozšířit velkým množstvím modulů. Obsahuje speciální moduly pro zabezpečenou komuni-

kaci, podporu skriptovacích jazyků, modul pro úpravu \adresy (s jeho pomocí můžeme vytvářet tzv. hezké adresy) a spoustu dalších modulů. V případě generování dynamického obsahu spouští skript při každém požadavku, ale je možné nastavit připojení k běžící systémové službě a její návratovou hodnotu odeslat uživateli.

Nginx je webový server s reverzní proxy a load managmentem, který se zaměřuje na vysoký výkon a nízké nároky na paměť. Jeho další výhodou je odolnost vůči některým útokům na webové servery a proto se využívá u velmi vytížených webových stránek jako reverzní proxy pro webový server Apache, který proti těmto útokům není tak dobře ochráněn. Pro tvorbu dynamického obsahu vyžaduje spuštění systémové služby generující obsah odpovědi.

Pro porovnání jednotlivých webových serverů jsem vytvořil jednoduchý test, při kterém jsem na čistý operační systém ručně nainstaloval testovaný webový server, k němu jsem nainstaloval skriptovací jazyk PHP, jako zástupce jazyků generujících dynamický obsah a knihovny nezbytné pro komunikaci s databází a zjistil paměťovou náročnost této soustavy programů a jméno uživatele, pod kterým jsou dané procesy spuštěny. Rychlost odezvy webových serverů nemělo smysl testovat, protože zpoždění vytvořené skriptovacím jazykem bude mnohem vyšší něž zpoždění vyvolané samotným webovým serverem.

Nginx			Apache		
Uživatel	Využitá paměť RAM [KiB]	Název procesu	Uživatel	Využitá paměť RAM [KiB]	Název procesu
www-data	47256	nginx	www-data	123620	apache2
www-data	47256	nginx	www-data	123580	apache2
www-data	47256	nginx	www-data	123580	apache2
www-data	47256	nginx	www-data	123580	apache2
root	122172	php-fpm7.0	www-data	123580	apache2
www-data	122544	php-fpm7.0			
www-data	123492	php-fpm7.0			
Součet	604300		Součet	741488	

Tab. 5.3: Využití paměti RAM jednotlivými webovými servery (získáno použitím programu `top`).

Z tabulky č. 5.3 je zřejmé, že přestože Apache server má spuštěno pouze pět instancí, jsou tyto instance z důvodu spuštěného PHP modulu náročnější na paměť RAM než server Nginx, který má spuštěn čtyři instance a další tři instance má spuštěna služba PHP, která je pro provoz webového rozhraní nezbytná.

Vhodným webovým serverem pro nasazení je Nginx zejména pro jeho jednoduchost, stabilitu, odolnost vůči útokům a i nižší paměťovou náročnost.

Konfigurací tohoto serveru (v příloze na přiloženém CD ve složce /instalace_systemu_pro_sber_dat/ngnixVhost.j2) jsem se inspiroval na stránce [21].

V konfiguraci nejdříve provedeme nastavení nezabezpečeného serveru komunikujícího po protokolu http tak, aby prováděl přesměrování na zabezpečený server využívající protokol https. Toto přesměrování označujeme jako dočasné (stavový kód č. 302) z důvodu možné změny adresy zařízení a tím nastalých potíží, pokud by se na tuto adresu zapojilo zařízení, které protokol https nepodporuje. V tomto konfiguračním souboru je i neaktivní nastavení pro trvalé přesměrování, které se dá v případě potřeby zapnout.

Další nastavení v tomto souboru upravuje zabezpečený server. V tomto nastavení nejdříve definuji standardní komunikační porty 443 pro protokol https, dále definují umístnění certifikátů určených pro šifrovanou komunikaci a hlavní veřejný adresář webové aplikace. Další nastavení provádí zablokování přístupu ke skrytým, konfiguračním a souborům obsahujícím chybová hlášení, které obvykle obsahují citlivé informace. Dále v konfiguraci upravuji přístup k obrázkům a uloženým skriptům, kterým nastavuji maximální dobu uložení na vyrovnávacích serverech, protože jejich obsah se nemění.

Dále definuji jakým způsobem se mají zpracovávat skripty uložené v PHP souborech. Tyto soubory se odesílají službě PHP přes soket, což je soubor implementující rozhraní služby a umožňuje její komunikaci.

Poslední část konfigurace již jen provádí předání části adresy webové stránky jako parametr PHP skriptu. Toto nastavení umožňuje používat „hezké“ \adresy.

Systémová služba webového serveru `nginx` i systémová služba PHP `php7.0-fpm` je spouštěna pomocí inicializačního systému `systemd` při spouštění zařízení. Konfigurace automatického spouštění je součástí jednotlivých systémových instalačních balíčků.

5.3 Konfigurační a instalační nástroje

K instalaci a konfiguraci jsem se rozhodl využít nástroj, ve kterém bych vytvořil popis konfigurace celého zařízení a jeho spuštěním bych získal funkční zařízení s nainstalovanými, nastavenými a spuštěnými službami. Měl by umožňovat lokální instalaci a spouštění popisu, měl by umožňovat vzdálené spuštění konfigurace pomocí klienta, neměl by být náročný na systémové prostředky, nemusí provádět automatickou kontrolu konzistence nastavení a nástroj může požadovat prvotní instalaci operačního systému a nastavení počítačové sítě s přístupem na internet.

Konfigurační a instalační nástroje:

Ansible používá k popisu konfigurace jazyk YAML Ain't Markup Language (YAML), který je jednoduše čitelný nejen pro stroj, ale i pro člověka. Nástroj můžeme

nainstalovat přímo na Raspberry Pi (nemá spuštěnou žádnou službu a proto pouze zabírá místo na MicroSD kartě se systémem) a provádět konfiguraci lokálně, nebo můžeme nainstalovat nástroj na svůj počítač a správu provádět vzdáleně přes službu SSH. Nástroj nevyužívá svého agenta pro kontrolu konzistence, a proto není nutné mít spuštěnou žádnou další službu a zatěžovat zařízení.

Puppet používá k popisu konfigurace jazyk podobný jazyku Ruby. Tento nástroj potřebuje mít neustále spuštěný server kterému klienti automaticky odesílají svůj stav (defaultně každých 30 min) a server jim odpoví tzv. katalogem, což je požadovaný popis stavu. Dále klienti zkontrolují stav podle získaného katalogu a provedou změny v systému, pokud najdou rozdíl mezi stavem a katalogem tak, aby popis v katalogu odpovídal skutečnosti. Výslednou zprávu poté zašlou zpátky na server. Tento nástroj je možné používat lokálně, ale je nutné mít stále spuštěné služby pro master i klientskou aplikaci. Nástroj Puppet je vytvořen v jazyce Ruby.

Chef tento nástroj vychází z nástroje Puppet a proto je velmi podobný. Základním rozdílem je, že nástroj Puppet provádí zpracování úkolů paralelně a je nutné definovat jejich pořadí a závislosti, zatím co nástroj Chef provádí úkoly postupně jak má uvedeny v kuchařce.

Z výše popsaných nástrojů je zřejmé, že nejvíce nám vyhovuje nástroj Ansible, který jsem se rozhodl použít k automatizaci celé instalace a nastavení.

Ansible předpokládá předem nainstalovaný OS s nakonfigurovaným síťovým rozhraním a přístupem k internetu. V průběhu instalace budou instalovány programy přes balíčkovací systém OS Raspbian, které se budou stahovat přes internet z oficiálních repositářů.

Nástroj Ansible je včetně modulů vytvořen v jazyce Python. Moduly, které Ansible používá k instalaci a konfiguraci se v případě vzdálené instalace (Ansible je nainstalován na počítači a k Raspberry Pi se připojuje přes službu SSH) nakopíruje na vzdálené zařízení, spustí a po té co je modul ukončen a zpráva o provedené úloze odeslána zpět, tak je modul ze vzdáleného zařízení smazán. Tento přístup klade malé nároky paměť na MicroSD kartě.

5.4 Nastavení systému pro sběr dat

Po té co jsem si stanovil potřebné programy a zjistil jejich požadavky jsem definoval hlavní konfigurační proměnné, které budou dále použity v konfiguračním nástroji Ansible v částí play booku `vars`, ve které provádíme nastavení proměnných.

Konfigurační proměnné definující hlavní nastavení zařízení pro sběr dat, nastavení aplikace pro sběr dat a nastavení zobrazování uložených

dat:

vhost_name_0 musí obsahovat IP adresu, nebo DNS jméno zařízení. Na tuto adresu se přesměrovává při přechodu z protokolu http na protokol https.

commit_every nám říká po kolika uložených vzorcích se má provést zápis do databáze a tím permanentně uložit naměřená data do souboru. Pro vysoké vzorkovací frekvence je vhodné tuto hodnotu nastavit na 1000 (viz. tabulka č. 5.1, ale pro vzorkovací frekvence menší než 1Hz je vhodné nastavit hodnotu 1, která nám zajistí postupné ukládání každého vzorku.

maxSamples definuje kolik vzorku se bude zobrazovat ve webovém grafu. Příliš velká hodnota může graf příliš znepřehlednit a navíc velmi zatíží server, který pak nemusí zvládat ukládání naměřených hodnot z A/D převodníku při vyšších rychlostech. Pokud je v nastaveném časovém rozmezí uložený menší počet záznamů než je požadovaný, tak vracíme všechny dostupné záznamy.

samplingRate vzorkovací frekvence pro čtení hodnot z A/D převodníku udávaná v Hz. Tuto hodnotu je možné zadávat i ve formě výpočtu např. `1/(60*60*6)` znamená vzorkování po šesti hodinnách.

samplingTime odpovídá době po kterou má být prováděno vzorkování. Zejména velmi rychlé jevy, které se vzorkuji vysokou frekvencí není nutné touto frekvencí vzorkovat po neomezenou dobu. Potom je vhodné touto proměnnou omezit dobu měření. Proměnnou je možné taky zadávat formou výpočtu.

definition_functions tato proměnná definuje vzorce pro přepočet hodnoty získané z ovladače A/D převodníku na hodnotu měřené veličiny. Způsob přepočtu je podrobně popsán v kapitole č. 5.5.5.

web_users seznam uživatelů a hesel, kteří budou mít přístup k naměřeným hodnotám. Hesla se pro jednoduchost ukládají na serveru v otevřené podobě, proto by se neměla používat stejná hesla jako mají uživatelé nastavená do jiných služeb a serverů.

Tyto důležité volby jsem v play booku pro přehlednost a snadnou orientaci sloučil do skupiny `USER_configuration` a také jsem je opatřil jednoduchými vysvětlujícími komentáři přímo v kódu.

Ostatní nastavené proměnné definují různé cesty pro konfigurační soubory, nastavení služeb a popis certifikátů použitých pro šifrovanou komunikaci https. Tyto hodnoty není většinou nutné měnit.

5.5 Aplikace pro sběr dat

Aplikace pro sběr dat má za úkol získávat naměřené hodnoty z A/D převodníku a ukládat je do databáze. K tvorbě programu jsem si vybral programovací jazyk Python z důvodů největších zkušeností s tímto programovacím jazykem. Pro splnění

úkolu máme vybraný A/D převodník MCP3208 ke kterému přistupujeme pomocí ovladače a vybranou databázi SQLite přístupnou pomocí ovladače v knihovně.

Aplikace pro sběr dat je nejdůležitější částí projektu sběru dat na Raspberry Pi a jsou na tuto aplikaci kladeny nejvyšší nároky. V režimu rychlého ukládání vzorků je nutné zajistit co nejvyšší vzorkovací frekvenci, přičemž se předpokládá, že tato aplikace poběží po omezenou dobu (max. jednotky hodin). Naopak v režimu s nižšími vzorkovacími frekvencemi je kladen důraz na přesnost vzorkování. U této aplikace se předpokládá, že poběží i výrazně delší dobu (např. při vzorkování 1 vzorek za 15 min. poběží řádově měsíce až roky).

5.5.1 Práce s databází

Pro práci s databází SQLite máme v programovacím jazyce Python dostupnou knihovnu `sqlite3`. Tato knihovna je dostupná v balíčkovém systému operačního systému a instaluje se zadáním příkazu `aptitude install sqlite3` (instaluji automaticky nástrojem Ansible). Knihovna je psaná v jazyce C a v jazyce Python je dostupná prostřednictvím modulu.

Tento modu zavedeme klasickým příkazem `sqlite3`. Po zavedení modulu se již můžeme připojit do databáze zadáním příkazu `conn = sqlite3.connect(database_name)`. Parametr funkce connect odpovídá databázovému souboru a pokud soubor neexistuje, tak je soubor automaticky vytvořen. Je nezbytné, aby systémový uživatel měl k databázovému souboru práva ke čtení a zápisu a zároveň se jedná o jediné zabezpečení dat v databázi, proto by k tomuto souboru neměli mít práva všichni uživatelé, ale jen nezbytná skupina uživatelů.

Speciálním případem je název souboru `:memory:`, který vytvoří databázi pouze v operační paměti zařízení. Tato databáze je velmi rychlá, ale po ukončení spojení je automaticky vymazána, a proto se pro naši aplikaci nehodí.

Dalším důležitým příkazem je `cursor = conn.cursor()`, kterým se vytvoří kurzor pro zadávání SQL příkazů. Těchto kurzorů můžeme mít klidně více a navzájem se nijak neovlivňují.

Nyní se již dostáváme k příkazu, pomocí kterého můžeme pokládat SQL dotazy a dávat SQL příkazy. Jedná se o příkaz `cursor.execute("SQL", parametry_SQL_dotazu)` jehož prvním parametrem je SQL příkaz a druhým parametrem je seznam proměnných, které se pozičně vkládají do SQL příkazu místo otazníků (můžeme vkládat i na jméno). Např. dotaz `surnames = cursor.execute("SELECT surname FROM users WHERE name = ?", "Milada")` doplní místo otazníku do dotazu jméno Milada. Tento přístup je bezpečnější než sestavovat přímo SQL dotaz i s parametry, protože parametry se vkládají až po sestavení příkazu což zabraňuje nežádoucí modifikaci odesílaného SQL dotazu.

Návratovou hodnotou bude objekt tzv. iterátor, který bude obsahovat všechna příjmení uživatelek s křestním jménem Milada. Tento objekt můžeme použít ve smyčce tímto způsobem `for surname in surnames:`, nebo si můžeme převést celý výsledek na pole polí příkazem `data = surnames.fetchall()`.

Databáze SQLite je relační a pokud vložíme nějaká data do tabulky příkazem `cursor.execute("INSERT INTO...)`, tak se data do databázového souboru neuloží dokud tuto relaci neuzavřeme příkazem `conn.commit()`. Dle manuálu můžeme nastavit i automatické uzavírání relace po každém příkazu (`con.isolation_level = None`), ale toto nastavení je nevýhodné obzvláště při velkém množství dat určených k zápisu, která takto můžeme ukládat hromadně.

Posledním důležitým příkazem je `conn.close()`, který ukončí připojení k databázi a odstraní veškeré zámky databáze, které si ovladač vytvořil. V případě násilného ukončení aplikace někdy tyto zámky databáze zůstanou, a potom je nutné, je ručně nebo nějakým automatickým skriptem smazat.

5.5.2 Schema databáze

Databázově systém SQLite ukládá data do tabulek, u kterých je předem známo, jaký typ budeme budeme ukládat. My budeme do tabulky ukládat časovou značku uloženého záznamu, a dále osmkrát přepočítané hodnoty jednotlivých kanálů A/D převodníku.

SQLite nemá vestavěn datový typ pro datum a čas, ale má vestavěné funkce v následujících datových typech [22]:

TEXT ukládá datum a čas podle ISO8601 tj. data ve formátu `YYYY-MM-DD HH:MM:SS.SSS`. Tento formát je pro nás nejvhodnější, protože umožňuje čitelný a srozumitelný zápis. Zápis je prováděn s přesností na tisíciny sekundy, ale to pro naši aplikaci nemusí být dostatečně přesné, proto vytvářím časové značky v programu (jazyk Python vytváří časové značky s přesností na miliontinu sekundy) a databázi předávám již hotovou časovou značku.

REAL tento datový typ provádí ukládání v Juliánském kalendáři. Tento typ s přesností na jeden den je vhodný na ukládání dlouhých časových období a historických událostí, ale pro naše přesné měření se nehodí.

INTEGER zápis data a času podle Unix Time. Tento formát definuje počet sekund od 1970-01-01 00:00:00 UTC. K ukládání časových značek se s ohledem na požadovanou přesnost při rychlém vzorkování nehodí.

Z výše jmenovaných datových typů jsem pro ukládání časové značky použil typ `TEXT`. Pro ukládání hodnot jsem použil datový typ s pohyblivou desetinnou čárkou `REAL`. Databázové schema se vytvoří SQL příkazem `CREATE TABLE samples`

```
(sample_time TEXT, ch1 REAL, ch2 REAL, ch3 REAL, ch4 REAL, ch5 REAL,
ch6 REAL, ch7 REAL, ch8 REAL);
```

Schema tabulky je nutné vytvořit před prvním spuštěním programu a o vytvoření se stará instalační a konfigurační nastroj Ansible.

5.5.3 Komunikační protokol A/D převodníku MCP3208 po sběrnici SPI

Pro svoji aplikaci jsem vybral A/D převodník MCP3208 který umožňuje připojení k Raspberry Pi pomocí sběrnice SPI. Komunikace probíhá pomocí tří bajtů (tab. č. 5.4).

Bajty odeslané z master zařízení Raspberry Pi do slave zařízení A/D převodník:

Bajt č. 0 bity č. 7 až 2 mají vždy tvar `000001` a znamenají zahájení A/D převodu. Další bit č. 1 (symbol `M`) nastavuje A/D převodník do unipolárního modu (hodnota 1) nebo do diferenčního modu (hodnota 0). Poslední bit č. 0 má význam horního bitu čísla kanálu (symbol `C`).

Bajt č. 1 horní bity č. 7 a 6 obsahují spodní bity čísla kanálu. Bity č. 5 až 0 nemají žádný význam (symbol `x`).

Bajt č. 2 bity nemají žádný význam.

Bajty odeslané z A/D převodníku do Raspberry Pi:

Bajt č. 0 bity nemají žádný význam. Dle datasheetu A/D převodníku je tato hodnota nastavena na napětí v logické 1 - napětí v logické 0.

Bajt č. 1 bity č. 7 až 5 nemají žádný význam. Bit č. 4 má význam startovací sekvence odesílání výsledku převodu a má vždy hodnotu `0`. Bity č. 3 až 0 mají význam horních čtyřech bitů výsledku převodu (symbol `R`).

Bajt č. 2 obsahuje dolních osm bitů výsledku převodu.

Byte	0	1	2
Tx	000001MC	CCxxxxxx	xxxxxxxx
Rx	xxxxxxxx	xxx0RRRR	RRRRRRRR

Tab. 5.4: Komunikační schema s A/D převodníkem po sběrnici SPI.[26]

Význam bitů označených jako `M` a `C` jsem shrnul v tab. č. 5.5. V tabulce jsou v diferenčním nastavení jednotlivé dvojice kanálů označeny `IN+` pro kladnou hodnotu

měřené veličiny a `IN-` pro zápornou hodnotu měřené veličiny, která se odečítá od hodnoty kladné.

Řídící bity				Konfigurace	Výběr
M	C_2	C_1	C_0	vstupů	kanálů
1	0	0	0	unipolární	kanál 0
1	0	0	1	unipolární	kanál 1
1	0	1	0	unipolární	kanál 2
1	0	1	1	unipolární	kanál 3
1	1	0	0	unipolární	kanál 4
1	1	0	1	unipolární	kanál 5
1	1	1	0	unipolární	kanál 6
1	1	1	1	unipolární	kanál 7
0	0	0	0	diferenční	kanál 0 = IN+ kanál 1 = IN-
0	0	0	1	diferenční	kanál 0 = IN- kanál 1 = IN+
0	0	1	0	diferenční	kanál 2 = IN+ kanál 3 = IN-
0	0	1	1	diferenční	kanál 2 = IN- kanál 3 = IN+
0	1	0	0	diferenční	kanál 4 = IN+ kanál 5 = IN-
0	1	0	1	diferenční	kanál 4 = IN- kanál 5 = IN+
0	1	1	0	diferenční	kanál 6 = IN+ kanál 7 = IN-
0	1	1	1	diferenční	kanál 6 = IN- kanál 7 = IN+

Tab. 5.5: Významy řídících bitů v komunikaci. [16]

5.5.4 Vyčítání dat z A/D převodníku

Protokol popsaný v kapitole č. 5.5.3 je implementován v knihovně `gpiozero`[26], kterou využívám.

Knihovnu `gpiozero` je možné do systému nainstalovat příkazem `aptitude install python3-gpiozero` (instaluji automaticky nástrojem Ansible).

Knihovna `gpiozero` je psaná v jazyce Python a umožňuje manipulovat s GPIO vstupy a výstupy.

Knihovna má speciální funkce pro používání LED diod, podpora tlačítek umožňuje připojení typu pull up (v klidovém stavu je na vstupu logická 1 a tlačítkem

připojujeme logickou 0) i typu pull down (opačné než pull up), ovládání motorů pomocí integrovaných obvodů SN754410, připojení různých čidel, např. světelný senzor, senzor vzdálenosti, pohybový senzor a A/Dpřevodníky podporující rozhraní SPI.

Ovladač pro A/D převodník MCP3208, je umístěn v objektu s názvem `MCP3208` (hierarchie objektů podporovaných A/D převodníků je na ob. č. 5.2), který při svém vytváření přijímá následující parametry.

Parametr `channel=0`, který definuje použitý kanál A/D převodníku z rozsahu nula až sedm. V objektu `MCP3208` se tato hodnota pouze zkontroluje, zda je ve správném rozsahu a předá se rodiči objektu, aby byla nakonec předána objektu `MCP3xxx`, ve kterém je tato proměnná použita v metodě `_int_to_words()`, která definuje protokol vyčítání dat z A/D převodníku.

Dalším parametrem je `differential=False` s výchozí volbou vypnuto. Tato volba je také zpracována v objektu `MCP3xxx` a použita v komunikačním protokolu. Volba zapíná pseudo diferenční měření mezi dvěma vstupy A/D převodníku. Protože tento typ měření využívá dva vstupy (např. kanál č. 0 odpovídá plusové hodnotě a od něj se odečte kanál č.1, ...), tak se maximální počet měřených kanálů sníží na čtyři.

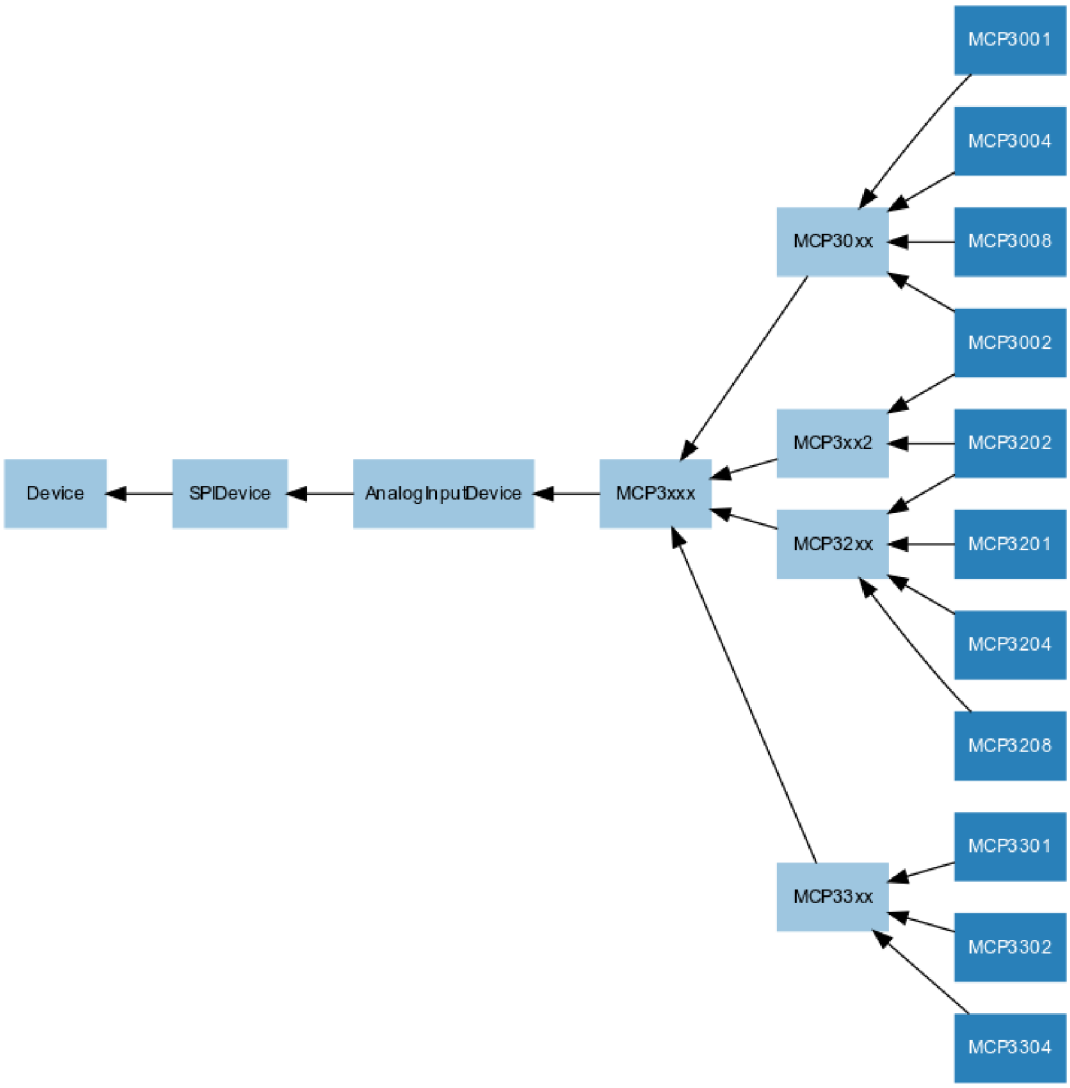
Další parametr `max_voltage=3.3` s výchozí hodnotou 3,3 V definuje maximální napětí U_{REF}. Tento parametr je zpracován v objektu `AnalogInputDevice` a slouží pouze k výpočtu měřeného napětí z relativní hodnoty napětí (relativní rozsah je v rozmezí nula až jedna.) v metodě `voltage()`.

Posledním parametrem, `**spi_args`, který zpracovává objekt `SPIDevice` a slouží k definici vstupních a výstupních vývodů GPIO. Ukázkové nastavení tohoto parametru, které je uvedeno v dokumentaci je `clock_pin=11, mosi_pin=10, miso_pin=9, select_pin=8` [26].

5.5.5 Ukázka přepočtu hodnoty přečtené z A/D převodníku na požadovanou veličinu

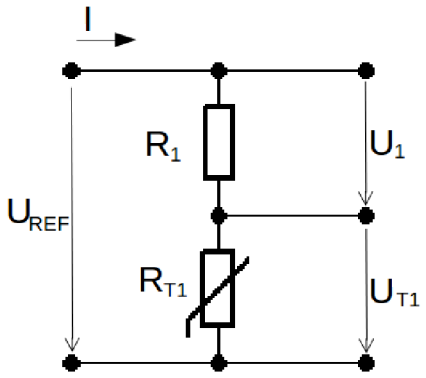
Pro demonstraci funkčnosti jsem použil snímač teploty `Pt100`, který jsem připojil ke kanálu č. 0 A/D převodníku. Před uložením do databáze je nutné hodnotu získanou z A/D převodníku přepočítat na měřenou veličinu. A/D převodník měří napětí na vstupu vzhledem k referenčnímu napětí a referenční zemi (na obr. č. 4.3 vstupy označené `VREF` a `AGND`). Pro vysvětlení postupu výpočtu teploty pomocí teplotního čidla `Pt100` si zjednodušíme zapojení dle obr. č. 5.3.

Z obr. č. 5.3 známe napětí $U_{REF} = 3,3\,V$, velikost odporu R_1, kterou si volíme při návrhu a velikost napětí U_{T1}, které získáme z A/D převodníku. Neznáme a potřebujeme zjistit velikost odporu R_{T1}, kterou potřebujeme do vzorce na přepočet odporu teplotního čidla na teplotu.



Obr. 5.2: Hierarchie tříd objektů A/D převodníků.[26]

K určování velikosti odporu teplotního čidla využijeme 1. Kirchhoffův zákon, který nám říká, že součet všech proudů, které do uzlu vtékají se musí rovnat součtu všech proudů, které z uzlu vytékají. Vzhledem k tomuto zákonu můžeme prohlásit, že proud I je v celém obvodu stejný a jeho velikost spočítáme pomocí Ohmova zákona dle vzorce č. 5.1. Ve vzorci č. 5.1 máme neznámou hodnotu U_1, kterou můžeme spočítat pomocí 2. Kirchhoffova zákona, který říká, že součet všech napětí zdrojů se rovná součtu všech úbytků napětí na všech spotřebičích. Potřebné napětí U_1 spočítáme dle vzorce č. 5.2. Nyní již spočítáme požadovanou velikost odporu teplotního čidla pomocí Ohmova zákona (vzorec č. 5.3). Po sloučení vzorců 5.1, 5.2 a 5.3 získáme výsledný vzorec pro výpočet odporu teplotního čidla (vzorec č. 5.4), který budeme potřebovat později.



Obr. 5.3: Schema připojení teplotního čidla.

$$I = \frac{U_1}{R_1} \tag{5.1}$$

$$U_1 = U_{REF} - U_{T1} \tag{5.2}$$

$$R_{T1} = \frac{U_{T1}}{I} \tag{5.3}$$

$$R_{t1} = \frac{R_1 U_{T1}}{U_{REF} - U_{T1}} \tag{5.4}$$

K výpočtu měřené teploty jsem se pro testovací účely rozhodl použít transformaci průběhu kalibrační křivky na přímku. Tímto zjednodušením jsem se dopustil maximální chyby 1,14 °C proti kalibrační tabulce [20].

$$t_1 = \frac{R_{t1} - R_{01}}{R_{MAX1} - R_{01}} \cdot t_{MAX} \tag{5.5}$$

Přepočet je prováděn pomocí vzorce č. 5.5, ve kterém hodnota R_{t1} odpovídá výsledku vzorce č. 5.4, hodnota R_{01} odpovídá odporu čidla při 0 °C, tj. 100 Ω pro námi zvolené čidlo Pt100, hodnotu t_{MAX} získáme z kalibrační tabulky příslušného senzoru [20] a odpovídá maximální námi požadované měřené teplotě a hodnota R_{MAX1} odpovídá odporu z kalibrační tabulky pro příslušné t_{MAX}.

Spojením vzorců č. 5.4 a 5.5 a převedením vzorce do programovacího jazyka Python, získáme konfigurační hodnotu `definition_functions` pro daný kanál A/D převodníku. Hodnotu U_{T1} je třeba nastavit na speciální symbol `x`, který bude v programu nahrazen za hodnotu získanou z A/D převodníku a hodnotu U_{REF} je nezbytné nastavit na 1 V z důvodu normalizace výstupní hodnoty ovladačem A/D převodníku na číslo v rozsahu 0-1. Ostatní hodnoty jsou nastaveny takto: $R_{01} = 100\,\Omega$, $t_{MAX} = 175$ °C, $R_{MAX1} = 166,63\,\Omega$, $R_1 = 200\,\Omega$. Nakonec ještě celý výpočet zaokrouhlíme na pět desetinných míst a ošetříme proti chybě dělení nulou.

Výsledný vzorec tedy je `round(((200*x/(1.0-x))-100)/(166.63-100)*175,5) if x != 1 else -1`.

Chybu dělení nulou jsem ošetřil vrácením chybové hodnoty -1, ale pro řadu aplikací toto může být nepřípustné a potom je nutné zvolit jinou hodnotu jako chybovou (pravděpodobně došlo k odpojení čidla a na vstupu A/D převodníku měříme referenční napětí). Podobným způsobem je možné provádět i složitější přepočty. Například teplotní čidlo `Pt100` má jiný vzorec přepočtu pro teploty nad nulou a pod nulou. Protože testovací měření teploty jsem prováděl v obytné místnosti, kde jsem nepředpokládal teplotu menší než 0 °C, tak jsem výpočet teploty pod nulou mohl zanedbat.

5.5.6 Aplikace pro sběr dat

Aplikace pro sběr dat má za úkol připojit se k ovladači A/D převodníku, vyčíst data jednotlivých kanálů, přepočítat je na námi měřenou veličinu a tu uložit do databáze.

V rámci vývoje jsem vytvořil aplikaci, která provádí své příkazy sériově a pro porovnání výkonu aplikace jsem vytvořil i aplikaci, která provádí své příkazy ve více vláknech.

Sériová aplikace

Aplikace pro sběr dat vytváří objekty zaštiťující práci s ovladači A/D převodníku a databáze. Schema objektů programu, jejich metod, vlastností a toku dat je pro sériovou aplikaci v příloze č. B, diagram programu je v příloze č. C a zdrojový kód je na přiloženém CD ve složce /instalace_systemu_pro_sber_dat/PythonService/readADconverter.py.

Základem aplikace pro sběr dat je vyčítání dat z A/D převodníku. Pro práci s kanálem A/D převodníku jsem si vytvořil objekt s názvem `ADChannel`, který přijímá jako parametr číslo kanálu A/D převodníku tak, jak je definováno v ovladači `gpiozero.MCP3208`, a funkci zajišťující přepočet hodnoty získané z A/Dpřevodníku na námi požadovanou hodnotu. Při svém vytvoření objekt provede vytvoření připojení k A/D převodníku s použitím ovladače a také vytvoří speciální funkci pro přepočet hodnoty získané z ovladače A/D převodníku na měřenou veličinu.

Objekt `ADChannel` implementuje jedinou metodu s názvem `return_value`, která nepřijímá žádné parametry a jejichž návratová hodnota je hodnota měřené veličiny.

Abych nemusel ručně vytvářet osm objektů pro osmikanálový A/D převodník a potom se každého ptát na jeho měřenou hodnotu, tak jsem si vytvořil objekt s názvem `ADConverter`, jehož úkolem je zaštítit celý A/D převodník. Při vytváření objekt požaduje při vytváření pouze pole funkcí k přepočtu hodnoty, které odesílá s číslem kanálu jednotlivým objektům `ADChannel` a ukládá si jejich adresy.

Objekt implementuje jedinou metodu s názvem `load_data`, která nepřijímá žádné parametry a má za úkol vytvořit pole, kde jako první položku vytvoří časovou značku vytvoření záznamu a jako další položky vkládá výstupy z objektu `ADChannel` metody `return_value` postupně od nejnižšího kanálu po nejvyšší. Výsledné pole vrátí v návratové hodnotě metody.

Objekt zaštiťující práci s databázovým ovladačem `sqlite3` se jmenuje `Database` a pro své vytvoření požaduje cestu k souboru s databázovým souborem, ve kterém již musí být vytvořena tabulka na zaznamenávání dat, dále požaduje jméno tabulky do které se mají data ukládat a také potřebuje číslo, po kolika vzorcích se mají data fyzicky ukládat do souboru. V rámci vytvoření objekt rovnou provede připojení k databázovému ovladači v knihovně `sqlite3` a jeho propojení s námi požadovaným souborem. Po navázání spojení si ještě uloží kurzor, kterému objekt bude předávat vytvořené SQL příkazy.

Další metodou, kterou objekt `Database` implementuje se jmenuje `insert`, která jako parametr požaduje pole o devíti proměnných obsahující na prvním místě časovou značku a na dalších místech postupně hodnoty veličin jednotlivých kanálů. Ze získaných dat metoda sestaví SQL příkaz pro vložení záznamu do tabulky databáze a předá jej ovladači `sqlite3` k provedení. Dalším úkolem této metody je zkontrolovat, jestli již byl odeslán požadovaný počet záznamů a má být provedeno jejich uložení do souboru.

O ukládání dat do souboru se stará samostatná metoda objektu nazvaná `commit`, která v případě selhání o této skutečnosti informuje návratovou hodnotou.

Hlavní program pro sběr dat nejdříve načte potřebné knihovny (`gpiozero.MCP-3208`, `sqlite3`, `time.sleep`, `datetime`, `math` - tuto v programu přímo nepotřebujeme, ale implementuje funkce, které můžeme potřebovat při přepočítávání výstupu z A/D převodníku na měřenou veličinu), dále načte konfiguraci programu ze souboru `/etc/readADconverter.conf` pomocí funkce `load_config`, která nepřijímá žádné parametry, ale vrací slovník obsahující požadované nastavení.

Nyní si vytvoříme jednotlivé objekty pro komunikaci s periferiemi `Database` a `ADConverter`. Definujeme si proměnnou obsahující periodu vzorkování a spočítáme si čas ukončení vzorkování, pokud jsou tyto funkce požadovány a definujeme si řídící proměnnou s názvem `run`, která nám v hodnotě `True` definuje průběh vzorkování a v hodnotě `False` definuje požadavek na ukončení vzorkování a celého programu.

Dále v programu spustíme smyčku s podmínkou na stav proměnné `run` ve které načítáme data z A/D převodníku příkazem `adata = converter.load_data()`, data dále předáváme objektu pro práci s databází příkazem `database.insert(adata)`, provádíme kontrolu splnění podmínky ukončení a nakonec smyčky pozastavujeme program metodou `sleep` na dobu stanovenou vzorkovací periodou.

Po splnění podmínky na ukončení vzorkování provedeme nastavení řídící pro-

měnné `run` na hodnotu `False`, dojde k ukončení výše popsané smyčky a nakonec provádíme řízené uvolnění paměti obsazené objekty. Uvolněním objektu `Database` zajistíme uložení veškerých dat do databázového souboru a odpojení ovladače databáze od souboru. Nekorektní ukončení spojení může mít za následek ztrátu dat, nebo ponechání databázového souboru v zamknutém stavu, které by zapříčinilo nemožnost opětovného spuštění programu. Dále je důležité správným způsobem uvolnit kanály A/D převodníku. V ovladači je vývojářům nahlášená chyba, která zapříčiňuje výjimku o nekorektním uvolnění objektu a pád aplikace. Definováním přesného postupu uvolňování paměti ovladač tuto výjimku nevyvolá a program je korektně ukončen.

Tato aplikace má velikou výhodu ve své rychlosti, protože nedochází k synchronizaci vláken programu, ale i nevýhodu v podobě synchronizece zápisu dat na MicroSD kartu, která zajišťuje trvalé uložení dat. Toto se projevuje výpadkem ve vzorkování na dobu nezbytně nutnou k uložení dat. Frekvence výpadků se dá ovlivnit nastavením konfigurační proměnné `commit_every` v konfiguračním souboru aplikace, nebo odpovídající volbou v play booku nástroje Ansible.

Vzhledem ke své vysoké rychlosti jsem vytvořil i verzi jednovláknové aplikace bez možnosti nastavení vzorkovací frekvence. Podrobnější srovnání bude provedeno v kapitole č. 5.5.7.

Vícevláknová aplikace

Schema objektů programu, jejich metod, vlastností a toku dat je pro vícevláknovou aplikaci v příloze č. D diagram aplikace je v příloze č. E a zdrojový kód je na přiloženém CD ve složce /instalace_systemu_pro_sber_dat/PythonService/readADconverterThreads.py.

Vícevláknová aplikace využívá vícevláknového zpracování dat. V jazyce Python máme na výběr z několika knihoven, které implementují vícevláknový přístup. Já jsem vybral knihovnu `asyncio`, která byla vytvořena pro asynchronní zpracování vstupně výstupních signálů.

Pro komunikaci s ovladačem A/D převodníku jsem použil identické objekty `ADChannel` a `ADConverter` jako u sériové aplikace, pouze v objektu `Database` jsem udělal drobnou změnu, která spočívá v odstranění podmínky pro uložení dat do databázového souboru z metody `insert` a její uložení do vlastní metody nazvané `conditioned_commit`. Tato metoda nemá žádné vstupní parametry ani návratovou hodnotu, pouze zkontroluje, zda vnitřní počítadlo odeslaných záznamů překročilo hodnotu `number_commits_items` nastavenou v konfiguračním souboru a pokud ano, tak spustí metodu `commit` pro uložení dat do souboru. Tuto změnu jsem provedl,

aby nedocházelo k ukládání dat do databázového souboru dokud není zásobník vyprázdněn.

Pro práci s vlákny jsem vytvořil objekt s názvem `Threads`, který při vytváření požaduje již vytvořené objekty `Database` a `ADConverter`. Požaduje také hodnoty vzorkovací frekvence pro výpočet periody vzorkování a požadovanou dobu vzorkování. Objekt si také vytvoří frontu k předávání záznamů určených k uložení do databáze (realizováno polem, do kterého se budou přidávat nové prvky), synchronizační zámek zajišťující konzistenci předávaných dat a řídící proměnou `run`. Posledními kroky při vytváření objektu jsou vytvoření nekonečné smyčky spouštějící vlákna provedené vytvořením objektu mudulu `asyncio` a samotné spuštění vláken modulem `asyncio`.

V aplikaci jsem vytvořil dvě vlákna. První zajišťuje načítání dat z objektu `ADConverter`, a poté uzamkne přístup do sdílené proměnné, uloží do ní načtená data a proměnnou zase odemkne pro přístup druhého vlákna. Nyní vlákno příkazem `await asyncio.sleep(self.samplingPeriod)` uspíme na dobu nastavené periody a o probuzení se postará modul `asyncio`. Načítání dat je obdobně jako u sériové aplikace umístěno ve smyčce s podmínkou na stav v proměnné `run`. Funkce tohoto vlákna má vliv vzorkování měřených dat přičemž nekontrolované zpoždění vzorkování může způsobit pouze druhé vlákno, které si ve stejný okamžik přebírá data ze sdílené proměnné, nebo plánovač operačního systému, který nejsme schopni ovlivnit.

Druhé vlákno aplikace zajišťuje ukládání dat do databáze. tato funkce je také umístěna do smyčky s podmínkou na stav v proměnné `run`, ale v tomto případě musíme i zajistit, že budou načtena a uložena veškerá data uložená do sdílené proměnné. Z tohoto důvodu jsem ukončovací podmínku smyčky doplnil i o test, zda je sdílená proměnná prázdná.

Dalším krokem je test, zda máme ukončit vzorkování a pokud ano, tak nastavíme pomocnou proměnou `run` na hodnotu `False`. Nyní uzamkneme pomocnou sdílenou proměnnou proti změnám, proměnnou si lokálně zkopírujeme, vymažeme její sdílenou verzi a opět ji odemkneme, aby nedocházelo k přílišnému zdržení vlákna zajišťujícího načítání dat z objektu `ADConverter`. Nyní, když je sdílená proměnná uvolněna, tak vlákno může provádět ukládání dat do databáze pomocí objektu `Database` metody `insert` a je-li splněna podmínka na synchronizaci dat do databázového souboru, tak provedeme i tuto operaci.

Vlákno zajišťující ukládání dat do databáze je pokaždé uspáno na dobu jedné sekundy, což nám zajišťuje, že změřené vzorky budou do jedné sekundy odeslány databázovému ovladači.

Při požadavku na ukončení aplikace je nutné nejdříve ukončit vlákno provádějící čtení dat z A/D převodníku, dále je nutné ukončit vlákno provádějící ukládání dat, aby nedošlo k jejich ztrátě a nakonec můžeme provést uvolnění všech objektů stejně

jako v sériové aplikaci.

5.5.7 Srovnání verzí aplikací k zaznamenávání naměřených dat

V průběhu vývoje aplikace pro sběr dat jsem vytvořil verzi pracující sériově a vícevlánknovou aplikaci. Ke zjištění rychlosti a přesnosti vzorkování jsem vytvořil test, který prováděl vzorkování po dobu třiceti minut. Z takto naměřených dat jsem spočítal počet vytvořených vzorků a směrodatnou odchylku.

	Max. Rychlost			Vzorkovani 1 Hz	
	Sériová aplikace	Sériová aplikace bez vzorkovací frekvence	Vláknová aplikace	Sériová aplikace	Vláknová aplikace
Doba ukládání dat	0:29:59,998474	0:29:59,957752	0:30:00,820674	0:30:00,606316	0:30:00,172871
Počet vzorků	604 788	605 999	516 342	1 788	1 785
Průměrná doba vytváření vzorku [s]	0,00297	0,00297	0,003488	1,00705	1,0085
Rozptyl [s^2]	$9{,}1358 \cdot 10^{-06}$	$1{,}0595 \cdot 10^{-05}$	$1{,}0339 \cdot 10^{-05}$	$1{,}1927 \cdot 10^{-06}$	$6{,}305 \cdot 10^{-08}$
Směrodatná odchylka [s]	0,003023	0,003255	0,003215	0,001092	0,000251
Průměrná vzorkovací frekvence [Hz]	336,0215	336,7003	286,6972	0,993	0,9916

Tab. 5.6: Srovnávací tabulka verzí aplikace pro sběr dat.

Z tab. č. 5.6 jsem zjistil, že nejrychlejší aplikace je sériová, bez možností volby vzorkovací frekvence s průměrnou rychlostí vzorkování 336,7003 Hz a směrodatnou odchylkou 0,003255 s. Výsledek byl zapříčiněn nejmenší režií potřebnou na uložení jednoho záznamu.

Při nastavení vzorkování na 1 Hz byly rozdíly ve vzorkovací frekvenci minimální, ale naopak se výrazně snížila směrodatná odchylka pro vícevláknovou verzi aplikace pro sběr dat. Důvodem tohoto výsledku je oddělení vytváření vzorku do zvláštního vlákna, které není zdržováno ukládáním naměřených dat do databáze a dalšími obslužnými funkcemi.

Z výše popsaných důvodů jsem se rozhodl použít sériovou aplikaci bez možnosti nastavení vzorkovací frekvence pro nasazení, ve kterém potřebujeme vzorkovat maximální rychlostí i vícevláknovou aplikaci, pokud potřebujeme vzorkovat stanovenou vzorkovací frekvencí. Ansible role dokáže sama vyhodnotit, zda je nastavena vzorkovací frekvence a v případě, že je požadavek na maximální vzorkovací frekvenci, tak je nainstalována sériová aplikace bez možnosti nastavení frekvence a v případě nastavení frekvence, tak je automaticky nainstalovaná aplikace vícevláknová.

5.5.8 Spouštění aplikace pro sběr dat

Je požadováno, aby aplikace pro sběr dat byla spouštěna automaticky při spuštění zařízení, proto jsem se rozhodl využít službu `systemd`. Tato služba provádí spouštění

shellových skriptů umístěných ve složce `/etc/init.d/`.

Spouštěcí skript aplikace pro sběr dat je umístěn na přiloženém CD ve složce /instalace_systemu_pro_sber_dat/Templates/readADconverter.j2 a je založen na skriptu uvedeném v [27].

Skriptovacímu souboru je při spuštění předán jeden parametr, který může být jeden z řetězců: start, stop, restart a status. Jakékoli jiné řetězce, nebo úplné vynechání parametru vyvolají vypsání nápovědy.

Ve skriptu nejdříve definuji funkci, která mi na základě souboru s číslem procesu služby ověří zda je daný proces služby skutečně spuštěn.

Volba start nejdříve ověří zda je služba skutečně spuštěna a pokud ano, tak vypíše informační zprávu. Pokud spuštěna není, tak provede její spuštění startovacím příkazem, a poté dojde k uložení čísla procesu služby do odpovídajícího souboru. Nakonec se ještě ověří, že ke spuštění skutečně došlo a skript skončí s návratovou hodnotou 1 pokud se spouštění nezdařilo, nebo návratovou hodnotou 0. pokud bylo spouštění úspěšné.

Volba stop nejdříve zkontroluje zda není již služba zastavena, pokud není, tak spustí příkaz k jejímu ukončení a vyčká na skutečné ukončení služby. Pokud k ukončení nedojde do deseti sekund, tak skript skončí s návratovou hodnotou 1. V případě úspěšného ukončení procesu služby dojde ještě ke smazání souboru s číslem služby a zámku databáze pokud existuje.

Volba restart nejdříve provede své rekurzivní spuštění s volbou stop a pokud je služba ukončena, tak opět provede své rekurzivní spuštění tentokrát s volbou start.

Volba status pouze provede ověření, zda je proces služby spuštěn a výsledek vrátí v návratové hodnotě.

Služba může být ve třech stavech:

1. Služba byla spuštěna (byl vytvořen soubor s číslem služby) a proces služby běží. V tomto případě běží vzorkování a zařízení funguje správně.
2. Služba byla spuštěna, ale proces služby neběží. Tento stav znamená, že uplynul požadovaný čas vzorkování, aplikace již splnila svůj úkol a ukončila se. Spuštění aplikace je možné restartováním služby, nebo celého zařízení.
3. Služba byla ukončena (soubor s číslem služby byl smazán) a proces neběží. Aplikace byla ručně nebo systémem v případě vypínání OS ukončena.

5.6 Webová aplikace pro zobrazování a stahování dat

Úkolem webové aplikace je zpřístupnit data v databázi uživateli. Data je požadováno zobrazovat ve formě grafu průběhů jednotlivých měřených hodnot, tato data zobrazit

v tabulce a také umožnit jejich stažení do počítače uživatele k další podrobnější analýze. Data je nutno zabezpečit uživatelským jménem a heslem.

Webová aplikace má data k dispozici v databázi SQLite a odesílat data uživateli budeme přes webový server Nginx. Tento webový server nám umožňuje aplikaci vytvořit v mnoha jazycích (např. C, C++, PHP, Ruby, Python, ...).

5.6.1 Frameworky webových aplikací

Z důvodu zvýšení bezpečnosti a urychlení vývoje jsem se rozhodl vývoj provádět ve webovým frameworku. Tímto rozhodnutím jsem získal možnost využít řadu funkcí (např. šablonovací systém, funkce pro práci s databázemi, funkce omezující oprávnění přístupu k webovým stránkám, atd.), které již jsou ve frameworku dostupné.

Webowé frameworky:

Nette Framework je opensource webový framework psaný v jazyce PHP zaměřený na bezpečnost a snadnost psaní aplikací. Framework se skládá z řady komponent a je plně škálovatelný.

Zend Framework je framework psaný v jazyce PHP a obsahuje obdobnou funkcionalitu jako Nette Framework.

Symfony2 je framework psaný v jazyce PHP a jeho vývoj byl inspirován frameworky Ruby on Rails, Django a Spring.

Django je framework psaný v jazyce Python. Výhodou frameworku je snadnost psaní aplikace. Po definici databázového modelu si framework umí sám vytvořit tabulky a zpravovat vazby mezi jednotlivými tabulkami.

Ruby on Rails framework je psaný v jazyce Ruby a od ostatních frameworků se liší tím, že šetří práci používáním globálního nastavení a pokud chce programátor nějakou vlastnost frameworku blíže specifikovat, tak ji musí ručně nastavit.

Rozdíly mezi webovými frameworky jsou minimální, proto jsem vybral Nette Framework [23], se kterým jsem se setkal již dříve a proto s ním mám zkušenosti.

Framework využívá návrhový vzor Model-View-Presenter, kde modelová vrstva jako jediná komunikuje s databází a provádí zápis a načítání dat dle požadavku presenteru. Presenter přijatá data zpracuje a předá vrstvě view, která zajistí pomocí šablonovacího systému Latte konečné vykreslení webové stránky.

Součástí Nette Frameworku je i několik nástrojů usnadňující vývoj aplikace, jedná se zejména o nástroj Tracy, který nám zajišťuje zobrazování a ukládání chybových hlášení. Tato hlášení se ukládají do složky `log` vždy, když k nějaké chybě dojde, ale k zobrazení chybových hlášek dojde pouze v případě, že je nástroj Tracy přepnut do vývojového režimu. Pokud ve vývojovém režimu přepnut není, tak se uživateli zobrazí obecná chyba např. 500 Internal server error, znamenající chybu aplikace.

Dalším velmi užitečným nástrojem je šablonovací systém Latte. Tento nástroj umožňuje efektivní a bezpečné vytváření šablon webových stránek. Všechny vkládané proměnné automaticky ošetřuje správným způsobem (jinak při vkládání proměnné do zdrojového kódu PHP a jinak při vkládání do zdrojového kódu JavaScriptu) a pokud chci funkci automatického ošetřování vypnout, tak musím toto úmyslně zakázat.

Nette Framework obsahuje i jednoduchou metodu autentizace a autorizace, kterou využijeme k zabezpečení dat naší webové aplikace.

Daní za použití frameworku je mírně pomalejší odezva webového rozhraní daná režií frameworku.

5.6.2 Zobrazování grafu

Pro vykreslování grafu je možné použít několik nástrojů. Požadavkem je, aby nástroj umožňoval zobrazování několik průběhů (potřebujeme zobrazit průběhy osmi kanálů A/D převodníku do jednoho grafu), umožňoval vypínání a zapínání jednotlivých průběhů (každý kanál může zobrazovat hodnoty v jiném rozmezí hodnot a graf proto může být nepřehledný). Dalším požadavkem je i klást důraz na nízkou zátěž zařízení a raději práci přesunout do prohlížeče uživatele.

Nástroje pro zobrazení grafu:

PHP Chart tento nástroj napsaný v jazyce PHP umožňuje zobrazování různých typů grafů. Tyto grafy jsou generovány na straně serveru, tzn. na zařízení Raspberry Pi.

Chart.js je nástroj napsaný v jazyce JavaScript. Chart.js umožňuje zobrazování různých druhů grafů, umožňuje jednotlivé průběhy grafu skrývat a opět zobrazovat.

Vybral jsem si Chart.js z důvodu běhu ve webovém prohlížeči uživatele, proto nezatěžuje Raspberry Pi, které se může věnovat sběru dat, a také jeho funkcím, které usnadňují prohlížení průběhů měřených veličin.

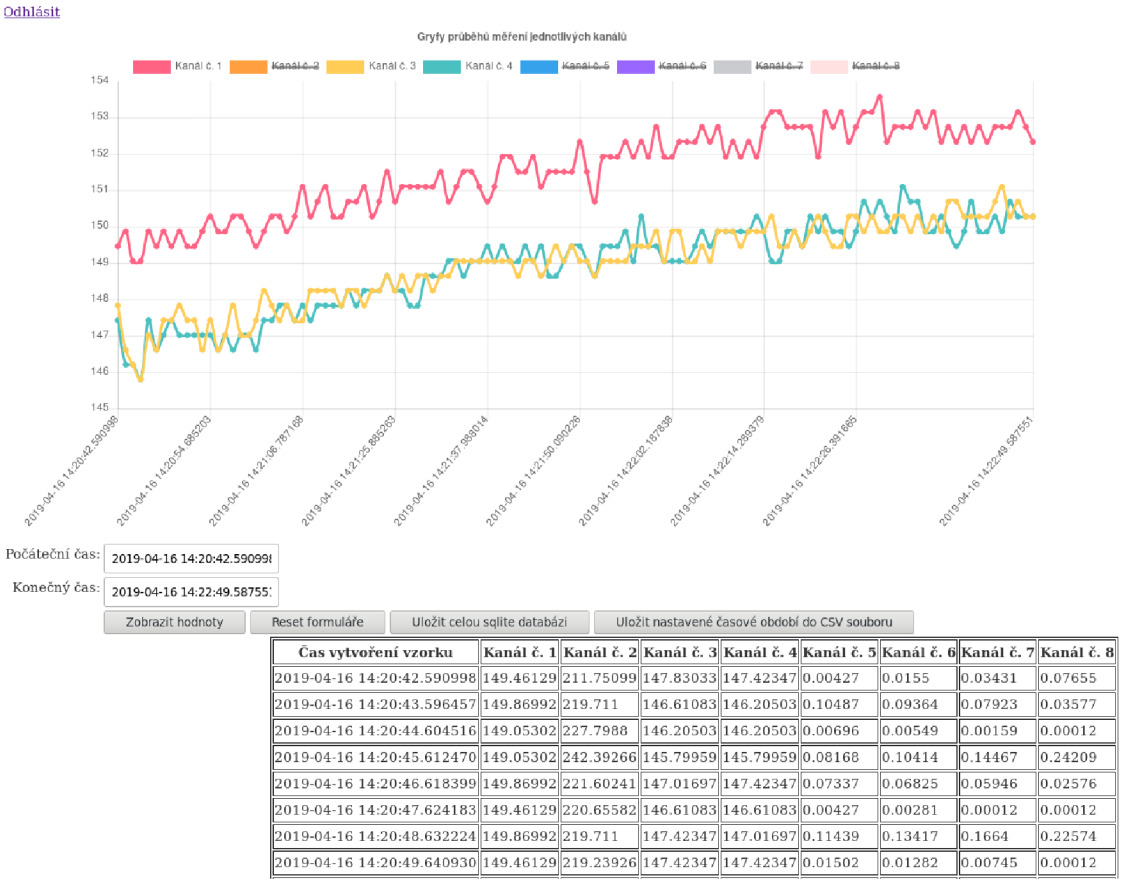
Chart.js [24] je opensource nástroj pro vykreslování grafů psaný v jazyce JavaScript. Tento nástroj dokáže vykreslovat celou řadu grafů. Např. sloupcový graf, čárový graf nebo koláčový graf. Pro naše nasazení je výhodné použít čárový graf, kdy do jednoho grafu vykreslíme všech osm kanálů oddělených barvami. Zobrazení těchto kanálů je možné zapínat a vypínat dle potřeby.

5.6.3 Webové prostředí

Úkolem webové aplikace je zprostředkovat naměřená data uživateli. Přičemž data musí nejen zobrazit v tabulce, ale i vykreslit grafy průběhů jednotlivých kanálů

a umožnit stáhnout požadovaný interval dat v textovém souboru typu Comma-separated values (csv) nebo celou SQLite databázi v binární podobě. Přístup k datům musí být chráněn jménem a heslem.

K dispozici máme data uložená v SQLite databázi, dále jsme si vybrali webový framework Nette, který nám usnadní vývoj a nástroj Chart.js, pomocí kterého budeme vykreslovat požadované grafy.



Čas vytvoření vzorku	Kanál č. 1	Kanál č. 2	Kanál č. 3	Kanál č. 4	Kanál č. 5	Kanál č. 6	Kanál č. 7	Kanál č. 8
2019-04-16 14:20:42.590998	149.46129	211.75099	147.83033	147.42347	0.00427	0.0155	0.03431	0.07655
2019-04-16 14:20:43.596457	149.86992	219.711	146.61083	146.20503	0.10487	0.09364	0.07923	0.03577
2019-04-16 14:20:44.604516	149.05302	227.7988	146.20503	146.20503	0.00696	0.00549	0.00159	0.00012
2019-04-16 14:20:45.612470	149.05302	242.39266	145.79959	145.79959	0.08168	0.10414	0.14467	0.24209
2019-04-16 14:20:46.618399	149.86992	221.60241	147.01697	147.42347	0.07337	0.06825	0.05946	0.02576
2019-04-16 14:20:47.624183	149.46129	220.65582	146.61083	146.61083	0.00427	0.00281	0.00012	0.00012
2019-04-16 14:20:48.632224	149.86992	219.711	147.42347	147.01697	0.11439	0.13417	0.1664	0.22574
2019-04-16 14:20:49.640930	149.46129	219.23926	147.42347	147.42347	0.01502	0.01282	0.00745	0.00012

Obr. 5.4: Webová aplikace pro zobrazování a stahování dat.

Přihlášení do webové aplikace probíhá pomocí jednoduchého autentifikátoru implementovaného v Nette Frameworku. Tento autentifikátor načítá jména a hesla uložená v konfiguračním souboru webové aplikace a na jejich základě provádí ověřování uživatelů. Autentifikátor vyžaduje hesla v textové podobě, proto by uživatelé neměli volit hesla, která používají k jiným službám. Protože předpokládáme, že přístup správce k zařízení bude mít jenom velmi malý počet uživatelů (nejspíše pouze jeden nebo dva), a protože není žádoucí, aby si uživatelé zřizovali na zařízení účet sami, tak je vhodné, aby správu uživatelů a v tomto případě i jejich hesel prováděli pouze tito určení správci zařízení, kteří by měli volit dostatečně silná a bezpečná hesla.

Webová aplikace po přihlášení je zobrazená na obr. č. 5.4. Na vrch webové stránky jsem vložil graf průběhů jednotlivých kanálů, dále jsou vložena dvě textová pole pro stanovení začátku a konce intervalu, který se bude zobrazovat. Tento interval není nutno nastavovat s přesností, která je předvyplněna, ale stačí jej nastavit s přesností např. na minuty.

Pod těmito textovými poli jsou umístěna postupně tlačítka pro odeslání nastaveného intervalu, resetování formuláře na maximální dostupný interval zobrazování, uložení SQLite databáze a posledním tlačítkem uložíme nastavený rozsah měřených hodnot do csv souboru.

V poslední části je zobrazena tabulka s naměřenými daty odpovídajícími zobrazenému grafu.

Zobrazení uložených dat

Zobrazování uložených dat provádíme dvojím způsobem. Ke grafickému zobrazení využíváme nástroj Chart.js popsaný v kapitole č. 5.6.2 a dále vypisujeme naměřená data do tabulky přímo ve webovém prohlížeči.

V případě nízkých vzorkovací frekvencí dojde velmi rychle k nárůstu počtu uložených vzorků, a proto bylo nutné provést omezení zobrazovaných záznamů. Počet záznamů, které se zobrazí nastavujeme buďto konfigurační volbou `maxSamples` v playbooku nástroje Ansible, nebo stejnou volbou v konfiguračním souboru webové aplikace. Při vykreslování si nejdříve zjistíme nejmenší a nejvyšší identifikační číslo záznamu (tato čísla si program SQLite vytváří interně sám), dále si zjistíme krok po kolika záznamech budeme vykreslovat a pokud je krok menší než jedna, tak vrátíme všechny dostupné záznamy, ale pokud je krok vyšší než jedna, tak sestavíme SQL dotaz, který nám vybere pouze požadovaný počet záznamů rovnoměrně z celého zvoleného intervalu. Tento krok je z celého vykreslování záznamů nejnáročnější na výkon zařízení, a proto není vhodné zobrazovat naměřená data při velmi rychlém vzorkování, ale až po ukončení měření.

Data zobrazená v tabulce pod grafem odpovídají zobrazenému grafu, a proto pokud je omezen počet záznamů v grafu, tak je stejným způsobem omezen i počet záznamů v tabulce. Pokud potřebujeme získat všechna naměřená data, tak musíme provést export dat do csv souboru, nebo stáhnout celou SQLite databázi.

Export dat

Export dat je důležitá součást aplikace, protože umožňuje data archivovat a dále zpracovávat k dalším výpočtům.

Prvním způsobem exportu je export do csv souboru. Při vytváření tohoto exportu jsem se inspiroval na webových stránkách [25].

Soubor má formát, ve kterém se textové řetězce umísťují do uvozovek a jednotlivé záznamy v řádku jsou oddělené čárkou. První řádek obsahuje nadpisy sloupců tak, jak jsou zobrazeny v tabulce na obr. č. 5.4. Na dalších řádcích jsou již vypsány časové značky jednotlivých měření a naměřené hodnoty. K vytvoření csv souboru jsem použil vestavěnou funkci PHP `fputcsv`, která formátuje vstupní data do požadované podoby, tato data dále zapisuje do výstupního streamu jazyka PHP (vzniká otevřením souboru `php://output` pro zápis) a data jsou dále předána webovému serveru k odeslání uživateli. Výhodou csv souboru je zejména možnost stanovit časový rozsah výstupu a podpora většiny matematických a statistických softwarů importování dat z csv souboru k další analýze.

Druhou možností exportu dat je stažení datového souboru SQLite databáze. Tento soubor si nejdříve zkopírujeme do dočasné složky, aby jsme omezili nekonzistence dat způsobené zápisem nově vytvořených vzorků, dále odešleme dočasný soubor pomocí interní funkce PHP `readfile`. Po odeslání tento zkopírovaný dočasný soubor smažeme. Výhoda stažení celé SQLite databáze je možnost pokládání složitějších SQL dotazů, např. výběr jen některých kanálů, hledání maximální a minimální změřené hodnoty kanálu, nebo přesnější výběr rozsahu vypsaných hodnot (aplikace umožňuje pouze určit rozsah pouze podle časových značek, ale muže nás zajímat rozsah určený změřenou hodnotou).

5.6.4 Testování výkonu webové aplikace

Výhon webové aplikace má vliv na uživatelský komfort a při vyšším počtu uložených dat může i negativně ovlivňovat vzorkovací frekvenci aplikace pro vyčítání dat z A/D převodníku. Důvodem tohoto negativního ovlivnění je zejména skript PHP, který provádí získání dat za databáze SQLite a sestavení webové stránky a využívá velké množství systémových prostředků, zejména procesorového času.

K měření rychlostí webové aplikace jsem se rozhodl využil nástroj Nette Frameworku s názvem Tracy Bar (obr. č. 5.5). Panel Tracy Bar, který je v produkčním prostředí vypnut zobrazuje v pravém dolním rohu webové stránky informace o načtené stránce. Zleva je nejdříve zobrazena celková doba běhu skriptu generujícího webovou stránku, dále je ikona, která po najetí myši zobrazuje okno se seznamem objektů webové aplikace, které byly načteny. Panel pokračuje informací o načítané webové stránce, její interní adresa (název presenteru). Poté následuje informace o komunikaci s databází, jaké byly prováděny SQL příkazy a jak dlouho trvalo tyto příkazy provést. Poslední informace se týká uživatele. Zda je přihlášen a jaké má role.

Webová aplikace byla po celou dobu testu nastavena, aby v grafu a v zobrazované tabulce hodnot zobrazovala nejvýše 100 záznamů (konfigurační volba s názvem

Obr. 5.5: Nástroj Tracy Bar.

`maxSamples`). Pro náš test bude z výše popsaných hodnot zobrazených v panelu důležitá doba běhu celého skriptu vykreslování stránky a celková doba zpracování SQL dotazů.

Výsledkem testu byla závislost počtu záznamů uložených v tabulce na velikosti databázového souboru a na době vytváření webové stránky.

Před započetím testování jsem si na stolním počítači vytvořil testovací data a dále jsem provedl klasickou instalaci zařízení Raspberry Pi pro sběr dat. Po instalaci jsem zastavil aplikaci pro sběr dat a smazal jsem databázový soubor. Dále jsem na zařízení nakopíroval skript (k dispozici na přiloženém CD ve složce /test_zatizeni_webu/test_web-create_database.py) a předem vygenerovaná data.

Test probíhal tak, že jsem si nejdříve vytvořil databázový soubor a v něm vytvořil prázdnou tabulku. Hodnoty získané načtením této stránky odpovídají výkonu aplikace bez uložených záznamů. Dále jsem opakovaně ručně spouštěl skript, který do databáze ukládal 1 500 000 záznamů a následně se ukončil. Po každém ukončení skriptu jsem provedl obnovení stránky tlačítkem Reset formuláře, které nejdříve smaže zapamatované výsledky SQL dotazů a poté provede opětovné načtení stránky do webového prohlížeče. Po načtení stránky jsem z panelu odečetl dobu sestavování webové stránky a celkovou dobu zpracování SQL dotazů. Pro každé množství vzorků jsem toto načtení stránky provedl čtyřikrát a výslednou hodnotu zprůměroval. Úplné výsledky měření jsou k dispozicina přiloženém CD ve složce /test_zatizeni_webu/vysledky.csv.

Hodnotu velikosti databázového souboru jsem získal příkazem `ls -lh`, který jsem spustil v shellu ve složce s databázovým souborem.

Z testu (tab.č. 5.7) jsem zjistil, že zpracování 1 500 000 záznamů a vypsání 100 záznamů do tabulky webové stránky trvá 2,4 s. Velikost databázového souboru uloženého na MicroSD kartě byla 84 MB.

Dále jsem výpočtem zjistil, že aplikaci pro sběr dat, která by vzorkovala frekvencí 300 SPS by 1 500 000 záznamů vytvořila za 1,4 hod.

Testováním jsem dále zjistil, že webová aplikace nedokáže zpracovat více než 15 000 000 záznamů a ukončí se z důvodu překročení maximální dovolené doby běhu PHP skriptu, která byla v době provádění testu nastavena na hodnotu 240 s. Tuto maximální dobu běhu je možné nastavovat v konfiguračním souboru PHP, nebo využít konfigurační nástroj Ansible, ve kterém jsem v bloku `php_configuration` vytvořil proměnnou `max_execution_time`.

Počet záznamů v tabulce	Velikost databázového souboru [B]	Doba běhu skriptu sestavování webové stránky [ms]	Rychlost zpracování SQL dotazů [ms]	Maximální doba po kterou lze vyčítat data z AD převodníku při vzorkovaci frekvenci 300 SPS [hod.]
0	8 K	34,5	1,4	0,0
1500000	84 M	2352,8	1117,8	1,4
3000000	169 M	4612,4	2216,6	2,8
4500000	254 M	6905,5	3337,1	4,2
6000000	339 M	9193,9	4458,8	5,6
7500000	425 M	11387,7	5540,6	6,9
9000000	510 M	13592,0	6622,4	8,3
10500000	595 M	15925,9	7744,7	9,7
12000000	680 M	18211,3	8845,4	11,1
13500000	765 M	20448,4	9975,5	12,5
15000000	850 M	27197,3	15432,7	13,9

Tab. 5.7: Rychlost načítání webové aplikace v závislosti na počtu vzorků v tabulce.

Výpočtem jsem zjistil, že při vzorkovací frekvenci 300 SPS by vytvoření 15 000 000 záznamů trvalo 13,9 hod.

Dále jsem v rámci testu prováděl i zjišťování velikosti databázového souboru. Testem jsem zjistil, že nejvyšší velikost souboru, kterou je zařízení je 850 MB. Pro instalaci OS Raspbian, naši aplikaci i uložená data je doporučovaná velikost MicroSD karty 8 GB dostatečná.

5.7 Konfigurace systému pro sběr dat nástrojem Ansible

Nyní již byly stanoveny základní konfigurační parametry systému pro sběr dat, vybral jsem vhodný webový a databázový server a vytvořil jsem aplikaci provádějící sběr dat z A/D převodníku a aplikaci která data zobrazuje uživateli. Dále je třeba vytvořit soubor s popisem stavu zařízení v jazyce YAML pro instalační a konfigurační nástroj Ansible a vytvořit šablony konfiguračních souborů formátovaných v jazyce Jinja2 [28], který je implementován v nástroji Ansible.

Adresářová struktura instalace (k dispozici na přiloženém CD ve složce /instalace_systemu_pro_sber_dat) se skládá ze složky `Templates`, která obsahuje šablony konfiguračních souborů formátované jazyce Jinja2 pro programy a služby: Webová aplikace, Aplikace pro sběr dat, PHP a webový server.

Další složka s názvem `PythonService` obsahuje obě verze aplikace aplikace pro sběr dat a poslední složka s názvem `wwwData` obsahuje webovou aplikaci. Webová aplikace je zde uložena bez webového frameworku a externích knihoven, které jsou stahovány a instalovány z originálních úložišť umístěných na internetu.

Soubor s popisem stavu zařízení tzv. playbook (v příloze na přiloženém CD ve složce /instalace_systemu_pro_sber_dat/AnsiblePlaybook.yml) se skládá ze čtyř hlavních částí. První část nastavuje nástroj Ansible. Zde nastavujeme způsob povýšení oprávnění pomocí nástroje `sudo`. Heslo pro povýšení práv je možno zadat při spuštění playbooku použitím parametru `--ask-become-pass`.

V druhé části se definují proměnné, které nám umožňují provádět veškerá nastavení z jednoho místa. Tyto proměnné jsou dále dostupné v celém playbooku a všech šablonách.

Nejdříve jsem definoval proměnné popsané v kapitole 5.4. Další definované proměnné není obvykle nutné měnit a určují například umístnění databázového souboru, které je nutné předat aplikaci pro sběr dat i webové aplikaci, která data zobrazuje. Dále je zde definovaná systémová cesta k adresáři s webovou aplikací, nebo jsou zde definovány parametry a umístnění šifrovacích certifikátů určených pro zabezpečenou komunikaci přes protokol https, a mnoho dalších nastavení.

Třetí část playbooku obsahuje popis stavu popsaný formou úkolů, které se mají provést. Nejdříve požadujeme plně aktualizovaný systém. Pro tento úkol jsem zvolil modul s názvem `APT`, kterému jsem předal požadavek k plné aktualizaci systému (`upgrade: dist`).

Dále požadujeme mít nainstalované různé programy knihovny funkcí. K tomuto úkolu jsem opět využil modul `APT`, ale tentokrát jsem mu předáním parametrů nastavil, že požaduji mít nainstalované všechny systémové balíčky ze seznamu, který jsem mu předal. Tento modul nejdříve ověří, zda jsou tyto balíčky nainstalované a pokud zjistí, že některý z nich není, tak jej doinstaluje.

Dalším důležitým krokem je zapnutí sběrnic SPI a I^2C, ktré jsou nezbytné ke komunikaci s A/D převodníkem. K tomuto úkolu jsem použil modul `replace`, který na základě regulárních výrazů provádí nahrazování řetězců v konfiguračním souboru dle požadavku. V případě, že dojde ke změně konfiguračního souboru, tak je ještě nutné provést restart zařízení Raspberry Pi, aby se změny projevily. Z tohoto důvodu zaregistrujeme požadavek na restartování zařízení, který bude proveden po provedení všech změn.

Dále požadujeme po zařízení, aby mělo nastaveno interpreter PHP dle našich požadavků. K tomuto úkolu jsem použil šablonu s požadovanou konfigurací. Tuto šablonu interpreteru PHP jsem vytvořil tak, že jsem si zkopíroval soubor s výchozím nastavením a v něm upravil výchozí nastavené hodnoty na příkazy pro vložení proměnných. Tyto proměnné musí být definovány v části `vars`. K tomuto požadavku jsem použil modul `template`, který nejdříve vygeneruje požadovaný tvar konfiguračního souboru a poté jej porovná s aktuálním nastavením. Pokud se nastavení změnilo, tak se provede nahrazení souboru za nový a zaregistruje se požadavek na restartování služby PHP.

Další kroky provádí nastavení webového serveru, generování certifikátu a vytvoření potřebných složek.

Dále požadujeme mít nainstalovaný instalační nástroj pro PHP nástroje s názvem Composer [29]. Tento nástroj potřebujeme k instalaci webového frameworku. K jeho instalaci jsem se rozhodl použít oficiální postup uvedený na stránkách projektu, který spočívá v několika krocích. Nejdříve je potřeba provést test, zda je již nainstalován a pokud není tak se provádí stažení instalačního souboru do dočasné složky a spuštění instalace nástroje. Nakonec se ještě provádí kontrola, zda je nainstalovaná aktuální verze.

Dalšími kroky se provádí zkopírování webové aplikace ze složky `wwwData` do požadované složky na zařízení Raspberry Pi, nastavení požadovaných přístupových práv k celé aplikaci, nastavení přístupových práv pro konkrétní složky a soubory, kopírování konfiguračního souboru webové aplikace, nainstalování webového frameworku a požadovaných javascriptových knihoven (jQuery [30], Chart.js). Součástí tohoto bloku je i vytvoření databázového souboru a tabulky určené pro ukládání vzorků naměřených A/D převodníkem.

Dále jsou kroky zajišťující instalaci a konfiguraci aplikace pro sběr dat. Nejdříve vytvářím konfigurační soubor a dále vybírám verzi samotné aplikace. Pokud je požadováno provádět vzorkování s maximální frekvencí, tak se nainstaluje aplikace jednovláknová, ale pokud potřebujeme provádět vzorkování o frekvenci nastavené v proměnné `samplingRate`, tak se automaticky nainstaluje aplikace vícevláknová. Dalším krokem se nainstaluje spouštěcí skript aplikace pro sběr dat.

Posledním úkolem je zajištění spouštění webového serveru, služby PHP a aplikace pro sběr dat při každém spouštění OS. Spouštění webové služby a služby PHP bývá zajištěno instalátorem balíčku operačního systému a proto se ve většině případů pouze zkontroluje nastavení. Spouštění aplikace pro sběr dat je tímto krokem při první instalaci potřeba pokaždé nastavit, ale protože ještě nejsou zapnuty sběrnice SPI a I^2C, tak služba není spuštěna a spustí se automaticky po restartu zařízení. V případě, že se nejedná o prvotní instalaci a sběrnice jsou již zapnuty, tak dojde i ke spuštění aplikace pro sběr dat.

Poslední část playbooku je věnována akcím, které jsou vyvolány při změně konfiguračních souborů. Tyto akce zajišťují restartování jednotlivých služeb z důvodu nového načtení konfiguračních souborů.

Jedinou výjimku tvoří požadavek restartování celého zařízení, které vyvolává zapnutí sběrnic SPI a I^2C. Tento restart je nutné provést se zpožděním, aby mohlo být dokončeno zpracování playbooku. Minimální doba na kterou je možné zpoždění nastavit je jedna minuta.

Nakonec jsem ještě vytvořil skript (v příloze na přiloženém CD ve složce /instalace_systemu_pro_sber_dat/install.sh) zajišťující instalaci nejnovější verze ná-

stroje Ansible na místní zařízení a provádějící lokální spuštění přiloženého play booku.

6 DOSAŽENÉ VÝSLEDKY

Hlavním hlediskem systému pro sběr dat je frekvence vzorkování a udává se buďto v jednotkách SPS označující počet vzorků za sekundu, nebo Hz jako frekvence vytváření vzorků, kterou je zařízení schopno zaznamenat a uložit. Tato rychlost je ovlivněna faktory popsanými níže.

Rychlost A/D převodníku - rychlost vybraného převodníku MCP3208 je dle parametrů z datasheetu 50 kSPS.

Rychlost databáze při ukládání na záznamové médium - dle tab. č. 5.1 jsme nejvyšší rychlosti dosáhli použitím databáze SQLite při ukládání po skupině obsahující tisíc záznamů a data se ukládala na interní MicroSD kartu.

Za těchto podmínek jsme dosáhli rychlosti ukládání dat 12 128 SPS.

Před začátkem testování rychlosti databázových systémů jsem si na stolním počítači vytvořil testovací data, která se skládala z osmi hodnot, která představovala osm změřených hodnot A/D převodníku. Tato data jsem uložil pomocí modulu Pythonu s názvem `pickle` do speciálního binárního souboru a tento soubor jsem nahrál do zařízení.

K testu databází jsem použil skript napsaný v jazyce Python (přiložený na CD /test_databazi/test_databases.py.j2). V tomto skriptu jsem si nejdříve načetl předpřipravený soubor s testovacími daty a dále jsem vytvořil tři funkce, které odpovídají třem testovaným databázím (SQLite3, MrriaDB a PostgreSQL). Tyto funkce se nejdříve připojí ke svému databázovému serveru, dále vytvoří potřebnou testovací tabulku a uloží si systémový čas začátku testu.

Dále jsem vytvořil samotný test, který spočívá v ukládání připravených dat do databáze a průběžné ukládání databázového souboru. Po kolika záznamech budeme ukládat datový soubor definujeme v parametru funkce a je součástí testování (toto neplatí pro databázi PostgreSQL, která si řídí ukládání do souboru sama).

Po uložení všech záznamů jsem si od aktuálního systémového času odečetl čas začátku testování a tím jsem získal dobu trvání testu. Nakonec jsem už pouze smazal testovací tabulku, ukončil spojení s databázovým serverem a jako návratovou hodnotu funkce vrátil dobu provádění testu.

Spouštění funkcí s testy provádí hlavní program, kterému předáváme číselné pole s počty po kolika odeslaných záznamech máme ukládat databázový soubor (tuto hodnotu definovanou v konfiguračním souboru jako `commit_every` jsem v testu postupně nastavoval na hodnoty 1 - ukládání každého vzorku zvlášť, 100, 500 a 1000). Dalším parametrem programu je kolikrát se mají jednotlivé testy opakovat a posledním je cesta k externímu úložišti, které slouží k testování. Pokud místo cesty k externímu úložišti zadáme prázdný řetězec, tak se tento test přeskočí.

Hlavní program nejdříve vytvoří proměnné, určené k uchovávání naměřených

hodnot, dále provede testování databází dle nastavení testovacích požadavků, provede výpočet průměrné doby zápisu do databáze a nakonec všechny výsledky vypíše do terminálu.

Zpoždění způsobené zatížením zařízení Raspberry Pi, operačním systémem a jeho plánování procesů, vstupně výstupními operacemi, vnitřní režií jazyka Python a dalšími spuštěnými aplikacemi - tato zpoždění jsou nepředvídatelná a nelze je výrazně ovlivnit. Jediné zatížení, které můžeme ovlivnit je práce webové aplikace tím, že po dobu měření nebudeme zatěžovat zařízení prohlížením uložených hodnot, ale jejich stažení provedeme až po ukončení měření.

Rychlost aplikace pro sběr dat - tuto rychlost zjistíme z tab. č. 5.6 a to z řádku Průměrná vzorkovací frekvence. Tato hodnota zahrnuje i všechny předchozí rychlosti a zpoždění, které se nedají z této rychlosti oddělit. Nejvyšší hodnotu jsme získali při použití sériové aplikace bez nastavení vzorkovací frekvence s hodnotou 336,7 SPS.

Testování aplikace pro sběr dat probíhalo spuštěním vzorkovací aplikace s nastavením vzorkovací doby na třicet minut. Toto testování probíhalo s různými verzemi aplikace pro sběr dat. Testoval jsem sériovou aplikaci s nastavením na co nejvyšší vzorkovací frekvenci a s nastaveným vzorkováním na 1 SPS. Stejnou sérii testů jsem provedl i pro aplikaci paralelní.

Výstupem těchto testů byly datové soubory databáze s uloženými záznamy, které jsem analyzoval skriptem napsaným v jazyce Python (přiložen na přiloženém CD ve složce /test_rychlost_aplikaci/statika.py). Tento skript si nejdříve zjistil z databáze ID nejmenšího a nejvyššího vzorku, dále si k nim doplnil odpovídající časové značky. Z těchto získaných dat jsme již schopni spočítat průměrnou vzorkovací frekvenci, ale pro vyhodnocení kvality programu potřebujeme znát ještě s jakou přesností vzorkování probíhalo tj. směrodatnou odchylku vytváření vzorků v sekundách. Tato směrodatná odchylka nám vypovídá o přesnosti vzorkování a jeho náchylnosti na zatížení zařízení.

Při vzorkování spojitého signálu a požadavku na jeho následnou rekonstrukci musíme splnit Vzorkovací teorém. Tento teorém nám říká, že pro přesnou rekonstrukci spojitého signálu musíme použít alespoň dvojnásobnou vzorkovací frekvenci než je nejvyšší harmonická složka obsažená v signálu. Z výše popsané rychlosti aplikace pro sběr dat (336,7 SPS) je zřejmé, že pro bezpečnou rekonstrukci spojitého signálu můžeme vzorkovat signál s nejvyšší harmonickou složkou 150 Hz.

Dalším hlediskem sběru dat je přesnost vzorkování. Toto hledisko u jednovláknové aplikace bez nastavení vzorkovací frekvence nemůžeme ovlivnit, protože aplikace je konstruovaná na maximální rychlost a každé výraznější zatížení zařízení tuto přesnost negativně ovlivňuje. U této aplikace jsem naměřil směrodatnou odchylku vytváření vzorků 0,003255 s. Pokud však nepotřebujeme vzorkovat s maximální rychlostí, tak můžeme od vzorkovací frekvence 250 SPS použít vícevláknovou aplikaci,

která je odolnější vůči zatížení zařízení. Směrodatná odchylka u této aplikace byla při vzorkování 1 SPS 0,000251 s.

Zařízení zobrazuje změřené hodnoty ve webové aplikaci, dostupné přes protokol https. Webový server jsem zvolil Nginx z důvodu nižší paměťové náročnosti a stabilitě. Webový server automaticky přesměrovává komunikaci z nešifrovaného protokolu http na šifrovaný protokol https. Přístup do aplikace je chráněn jménem a heslem, které do konfiguračního souboru webové aplikace systému zadává správce systému a uživatel si je nemůže měnit.

Data ze zvoleného rozsahu jsou zobrazována v grafu a zobrazena v tabulce, dále je možné data stáhnout do počítače uživatele ve formátu csv, nebo je možné stáhnout celý databázový soubor aplikace SQLite.

Webovou aplikaci jsem podrobil zátěžovému testu. Do databázového souboru jsem opakovaně nahrával 1 500 000 záznamů a sledoval jsem dobu běhu skriptu, který generuje webovou stránku a celkovou dobu zpracování SQL dotazů. Výsledky testu jsou zobrazeny v tab. č. 5.7.

Z testu jsem zjistil, že aplikace s nastavenou maximální dobou běhu na 240 s nedokáže zobrazit více než 15 000 000 záznamů. V případě vzorkování nejvyšší doporučovanou vzorkovací frekvencí 300 SPS dosáhneme 15 000 000 záznamů za 13,9 hod.

Dále jsme z testu zjistili, že databázový soubor uložený na MicroSD kartě měl při uložených 15 000 000 záznamech velikost 850 MB.

7 ZÁVĚR

Cílem této práce bylo navrhnout, realizovat a ověřit funkčnost zařízení pro sběr dat založeného na jednodeskovém počítači Raspberry Pi.

Protože jednodeskový počítač Raspberry Pi nedisponuje A/D převodníkem bylo třeba zbolit vhodný A/D převodník. Pro danou aplikaci se mi jevil jako nejvhodnější MCP3208-BI/P, který jde připojit po sběrnici SPI. Převodník obsahuje osm kanálů, má rozlišovací schopnost dvanáct bitů, rozsah vstupního měřeného napětí 0 až 5,5 V a rychlost převodníku je 50 kSPS je pro naši aplikaci dostatečná.

Dále jsem navrhl software pro vyčítání dat z A/D převodníku. K vyčítání dat jsem využil v jazyce Python knihovnu `gpiozero`. Pro ukládání naměřených vzorků jsem vybíral ze třech databázových systémů a na základě testu ve kterém jsem testoval rychlost zápisu 10 000 vzorků jsem se rozhodl použít databázový systém SQLite s ukládáním databázového souboru po tisíci záznamech. V případě vzorkování nižší frekvencí je možné ukládat databázový soubor po každém vzorku nastavením hodnoty jedna.

Tento software je spouštěn při startu zařízení pomocí služby `systemd`. Spouštěcí skript umístěný ve složce `/etc/init.d` umožňuje spuštění aplikace, její ukončení, restartování a dokáže vypsat stav, zda je aplikace spuštěna nebo ne.

Pro vzdálený přístup k datům v databázi jsem použil webový server Nginx, který jsem zvolil pro svou jednoduchost a paměťovou nenáročnost. Další výhodou je odolnost proti některým útokům odepření služby (např. zahlcením). Tento server je nakonfigurován k automatickému přesměrování z nezabezpečeného protokolu http na zabezpečený šifrovaný protokol https. Pro účely šifrování jsem v automatické instalaci zvolil certifikát podepsaný sám sebou, protože zařízení může být připojeno na neveřejné síti a certifikační autorita pro vydání certifikátu by mohla být nedostupná. V případě potřeby je možné certifikát vydaný certifikační autoritou nainstalovat ručně.

Pro grafické zobrazování hodnot přes webové rozhraní jsem použil Nette Framework, který mi umožnil díky svým nástrojům velmi rychlý a efektivní vývoj. Hlavní důvod výběru tohoto frameworku byl důraz na bezpečnost, protože automaticky ošetřuje data vkládaná uživatelem tak, aby nemohla sloužit k napadení webové aplikace.

Dále jsem z Nette Frameworku použil implementovaný autentifikátor k ověřování uživatelů. Tento autentifikátor se zapíná v konfiguračním souboru a v tomto souboru také definujeme uživatelská jména a hesla, která jsou v textové podobě. Autentifikátor jsem vybral z důvodu jednoduchosti implementace a předpokládaného malého počtu uživatelů.

Při návrhu webové aplikace mi velmi pomáhal i nástroj Tracy, který je součásti

Nette Frameworku. Tento nástroj slouží k přehlednému zobrazování chybových hlášek a v produkční verzi je přepnut do modu, ve kterém tyto hlášky nezobrazuje, ale pouze ukládá do speciální složky.

Webová aplikace umožňuje zobrazování naměřených vzorků v grafu a data zobrazená v grafu vypisuje i v tabulce pod formulářem. Dále aplikace umožňuje volbu rozsahu vypisování vzorků a stažení naměřených dat. Tato data je možné buďto stáhnout v souboru typu csv, nebo je možné stáhnout celý databázový soubor SQLite.

Zátěžovým testem webové aplikace jsem zjistil, že nejvyšší počet záznamů v databázové tabulce, které je webová aplikace ještě schopna zobrazit je 15 000 000. V případě, že je tato hranice překročena, tak generování stránky překročí maximální dobu zpracování a je ukončeno. V případě potřeby je možné tuto hranici zvýšit v konfiguračním souboru PHP, nebo v playbooku aplikace Ansible.

Funkčnost zařízení byla ověřena při ukládání naměřených hodnot z teplotních senzorů `Pt100`. Bylo dosaženo maximální vzorkovací frekvence 336,7 SPS se směrodatnou odchylkou vytváření vzorků 0,003255 s. Při použití nižších vzorkovacích frekvenci je možné dosáhnout i nižší směrodatné odchylky. Test prováděný pro vzorkovací frekvenci 1 SPS dosáhl směrodatné odchylky 0,000251 s.

V případě požadavku na rekonstrukci spojitého signálu je nutno splnit vzorkovací teorém. Tento teorém nám říká, že pro přesnou rekonstrukci spojitého signálu musíme použít alespoň dvojnásobnou vzorkovací frekvenci než je nejvyšší harmonická složka obsažená v signálu. Z tohoto důvodu můžeme zařízení pro sběr dat použít pro měření jevů, jejichž nejvyšší harmonická složka je 150 Hz.

LITERATURA

[1] Pico Technology: *High-resolution data acquisition* [online]. [cit. 30. 12. 2017]. Dostupné z URL: <https://www.picotech.com/data-logger/adc-20-adc-24/adc-20-adc-24-specifications>.

[2] Sensus: *CZ - CDL Data Sheet.pdf* [online]. 2004, poslední aktualizace 23. 5. 2004 [cit. 30. 12. 2017]. Dostupné z URL: <https://sensus.webdamdb.com/bp/#/search/22970927?q=%22Cosmos-%20Data%20Logger%20(CDL)%22>.

[3] COMET SYSTEM, s.r.o.: *Návod k použití - i-log-s6021* [online]. [cit. 30. 12. 2017]. Dostupné z URL: <http://www.cometsystem.cz/userfiles/file/Manuals-Czech/Dataloggery/i-log-s6021.pdf?time=1514734312&FixForIE=.pdf>.

[4] RPishop.cz: *Raspberry Pi 3 Model B 64-bit 1GB RAM* [online]. 2016, [cit. 30. 12. 2017]. Dostupné z URL: <http://rpishop.cz/kategorie/283-raspberry-pi-3-model-b-64-bit.html?src=raspberrypi>.

[5] Raspberry Pi Foundation: *Raspberry Pi Downloads - Software for the Raspberry Pi* [online]. [cit. 30. 12. 2017]. Dostupné z URL: <https://www.raspberrypi.org/downloads/>.

[6] RPishop.cz: *ADC Pi Plus Kit* [online]. [cit. 2. 1. 2018]. Dostupné z URL: <http://rpishop.cz/raspberry-pi-prislusenstvi/279-adc-pi-plus.html>.

[7] RPishop.cz: *ADC Differential Pi Kit* [online]. [cit. 2. 1. 2018]. Dostupné z URL: <http://rpishop.cz/vstupvystup-io/391-adc-differential-pi.html>.

[8] Texas Instruments: *12-Bit, 8-Channel Sampling ANALOG-TO-DIGITAL CONVERTER with I2C Interface (Rev. C) - ads7828* [online]. 2001, poslední aktualizace 3. 2011 [cit. 2. 1. 2018]. Dostupné z URL: <http://www.ti.com/lit/ds/symlink/ads7828.pdf>.

[9] Red Hat, Inc.: *Ansible Documentation* [online]. poslední aktualizace 10. 10. 2017 [cit. 3. 1. 2018]. Dostupné z URL: <http://docs.ansible.com/ansible/latest/index.html>.

[10] The Raspberry Pi Foundation: *GPIO - Raspberry Pi Documentation* [online]. [cit. 23. 3. 2019]. Dostupné z URL: <https://www.raspberrypi.org/documentation/usage/gpio/>.

[11] Root.cz: *Komunikace po sériové sběrnici I2C* [online]. poslední aktualizace 8. 1. 2009 [cit. 21. 3. 2019]. Dostupné z URL: <https://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc>.

[12] Root.cz: *Externí sériové sběrnice SPI a* I^2C [online]. poslední aktualizace 30. 12. 2008 [cit. 21. 3. 2019]. Dostupné z URL: <https://www.root.cz/clanky/externi-seriove-sbernice-spi-a-i2c>.

[13] HW server s.r.o.: *Stručný popis sběrnice I2C a její praktické využití k připojení externí eeprom 24LC256 k mikrokontroléru PIC16F877 | Vývoj.HW.cz* [online]. poslední aktualizace 20. 5. 2000 [cit. 22. 3. 2019]. Dostupné z URL: <https://vyvoj.hw.cz/navrh-obvodu/strucny-popis-sbernice-i2c-a-jeji-prakticke-vyuziti-k-pripojeni-externi-eeprom-24lc256>.

[14] SparkFun Electronics: *Serial Peripheral Interface (SPI) - learn.sparkfun.com* [online]. poslední aktualizace 30. 12. 2008 [cit. 23. 3. 2019]. Dostupné z URL: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/slave-select-ss>.

[15] Prof. Ing. Kamil Vrba, CSc., Ing. David Kubánek, Ph.D.: *A/D a D/A převodníky pro integrovanou výuku VUT a VŠB-TUO* [online]. 1. vydání, 2014, 116 s. ISBN 978-80-214-5116-2. Dostupné z URL: <https://vut-vsb.cz/home/get-file?file=468>.

[16] Microchip Technology Incorporated: *DATASHEET SEARCH SITE | WWW.ALLDATASHEET.COM - dsh.320-065.1.pdf* [online]. 2007, poslední aktualizace 8. 12. 2006 [cit. 2. 4. 2019]. Dostupné z URL: <https://www.gme.cz/data/attachments/dsh.320-065.1.pdf>.

[17] Analog Devices: *AD7998/AD7997 8-Channel, 10- and 12-Bit ADCs with I2C-Compatible Interface in 20-Lead TSSOP Data Sheet (REV. 0) - dsh.931-035.1.pdf* [online]. 2004 [cit. 2. 4. 2019]. Dostupné z URL: <https://www.gme.cz/data/attachments/dsh.931-035.1.pdf>.

[18] RPishop.cz: *ADC Pi - RPishop.cz* [online]. [cit. 2. 1. 2018]. Dostupné z URL: <http://rpishop.cz/ab-electronics/768-adc-pi-zero.html>.

[19] TME Czech Republic, s.r.o.: *Elektronické díly. Distributor a obchod online - Transfer Multisort Elektronik* [online]. [cit. 2. 1. 2018]. Dostupné z URL: <https://www.tme.eu/cz/>.

[20] CYNTEC CO., LTD.: *dsh.530-109.1.pdf* [online]. [cit. 2. 4. 2019]. Dostupné z URL: <https://www.gme.cz/data/attachments/dsh.530-109.1.pdf>.

[21] Jiří Pudil: *Configuring Nginx and PHP-FPM (and Nette) – Blog – Jiří Pudil* [online]. [cit. 8. 4. 2018]. Dostupné z URL: <https://jiripudil.cz/blog/configuring-nginx-php-fpm-nette>.

[22] Hipp, Wyrick & Company, Inc.: *Datatypes In SQLite Version 3* [online]. [cit. 8. 4. 2018]. Dostupné z URL: <https://www.sqlite.org/datatype3.html>.

[23] Nette Foundation: *Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework* [online]. [cit. 8. 4. 2018]. Dostupné z URL: <https://nette.org>.

[24] *Chart.js | Open source HTML5 Charts for your website* [online]. [cit. 10. 4. 2018]. Dostupné z URL: <https://www.chartjs.org/>.

[25] Kate Rose Morley *Creating downloadable CSV files using PHP* [online]. [cit. 10. 4. 2018]. Dostupné z URL: <http://code.iamkate.com/php/creating-downloadable-csv-files/>.

[26] Ben Nuttall: *gpiozero - Gpiozero 1.5.0 Documentation* [online]. [cit. 11. 4. 2018]. Dostupné z URL: <https://gpiozero.readthedocs.io/en/stable/>.

[27] Felix H. Dahlke: *init-script-template/template at master · fhd/init-script-template · GitHub* [online]. [cit. 11. 4. 2018]. Dostupné z URL: <https://github.com/fhd/init-script-template/blob/master/template>.

[28] Armin Ronacher: *Welcome | Jinja2 (The Python Template Engine)* [online]. [cit. 20. 4. 2018]. Dostupné z URL: <http://jinja.pocoo.org/>.

[29] Nils Adermann, Jordi Boggiano and many community contributions: *Composer* [online]. [cit. 20. 4. 2018]. Dostupné z URL: <https://getcomposer.org/>.

[30] The jQuery Foundation: *jQuery* [online]. [cit. 20. 4. 2018]. Dostupné z URL: <https://jquery.com/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

A/D	Analogově digitální
D/A	Digitálně analogový
MicroSD	Micro Secure Digital
USB	Universal Serial Bus
I^2C	Inter-Integrated Circuit
OS	Operační systém
SPI	Serial Peripheral Interface
SSH	Secure Shell
YAML	YAML Ain't Markup Language
SQL	Structured Query Language
NoSQL	non SQL
RAM	Random Access Memory
http	Hypertext Transfer Protocol
https	Hypertext Transfer Protocol Secure
PHP	Hypertext Preprocessor
URL	Uniform Resource Locator
SDA	Serial Data Line
SCL	Serial Clock Line
MISO	Master In, Slave Out
MOSI	Master Out, Slave In
SS	Slave Select
SCK	Serial Clock
GPIO	general-purpose input/output
EEPROM	Electrically Erasable Programmable Read-Only Memory
U_{VST}	Vstupní napětí
U_{REF}	Referenční napětí
THT	Through-hole technology
csv	Comma-separated values

SEZNAM PŘÍLOH

A POSTUP INSTALACE

Instalaci a konfiguraci Raspberry Pi pro sběr dat doporučuji provádět na novou instalaci OS Raspbian.

Doporučený postup instalace:

1. Stažení a zkopírování obrazu OS Raspbian na MicroSD kartu. K instalaci postačuje verze systému bez grafického prostředí, která je méně náročná na systémové prostředky.
2. Provedení instalace OS s pomocí instalátoru, nastavení sítě a v případě požadavku na vzdálenou správu i instalace SSH serveru.
3. Příprava Ansible playbooku. Provedení konfigurace v části `USER_configuration`, ve které je nutné nastavit zejména požadované parametry měření (vzorkovací frekvence a doba měření), přepočítávací funkce, parametr `vhost_name_0`, pokud je požadovaná adresa zařízení jiná, než aktuálně nastavená a přihlašovací údaje uživatelů, kteří budou mít přístup k naměřeným datům.
4. Při vypnutém Raspberry Pi hardwarově připojit A/D převodník dle obr. č. 4.3 a připojit požadovaná měřící čidla.
5. Zkopírování Ansible playbooku se všemi přiloženými soubory na Raspberry Pi. Ten je možné zkopírovat pomocí USB flashky, pomocí čtečky karet nakopírovat přímo na MicroSD kartu se systémem, nebo také vzdálené po počítačové síti pomocí SSH serveru.
6. Spuštění skriptu `install.sh` přiloženého u Ansible playbooku. Tento krok vyžaduje připojení k internetu z důvodu instalace programů a stahování potřebných knihoven.
7. Po restartu zařízení způsobeném Ansiblem již dojde ke spuštění aplikace pro sběr dat a celé zařízení začne vykonávat svoji činnost.

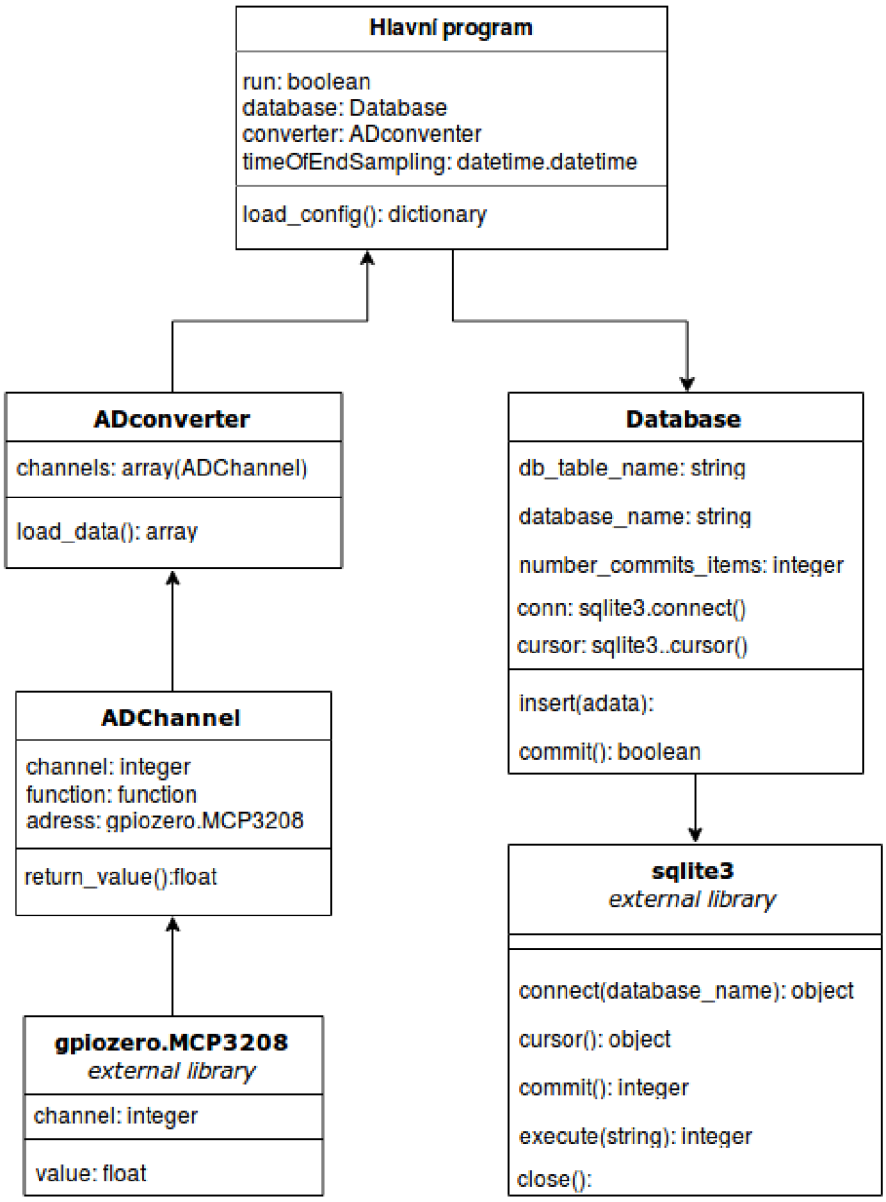
V případě požadavku na změnu konfigurace zařízení doporučuji neprovádět nastavení přímo v konfiguračních souborech, ale doporučuji opět použít nástroj Ansible z důvodu zajištění konzistence nastavení. Dále by také mohlo dojít k přepsání konfigurace do původního stavu, pokud by někdo spustil playbook s neaktuální konfigurací (např. z důvodu vytvoření nové čisté databáze).

Postup změny nastavení zařízení:

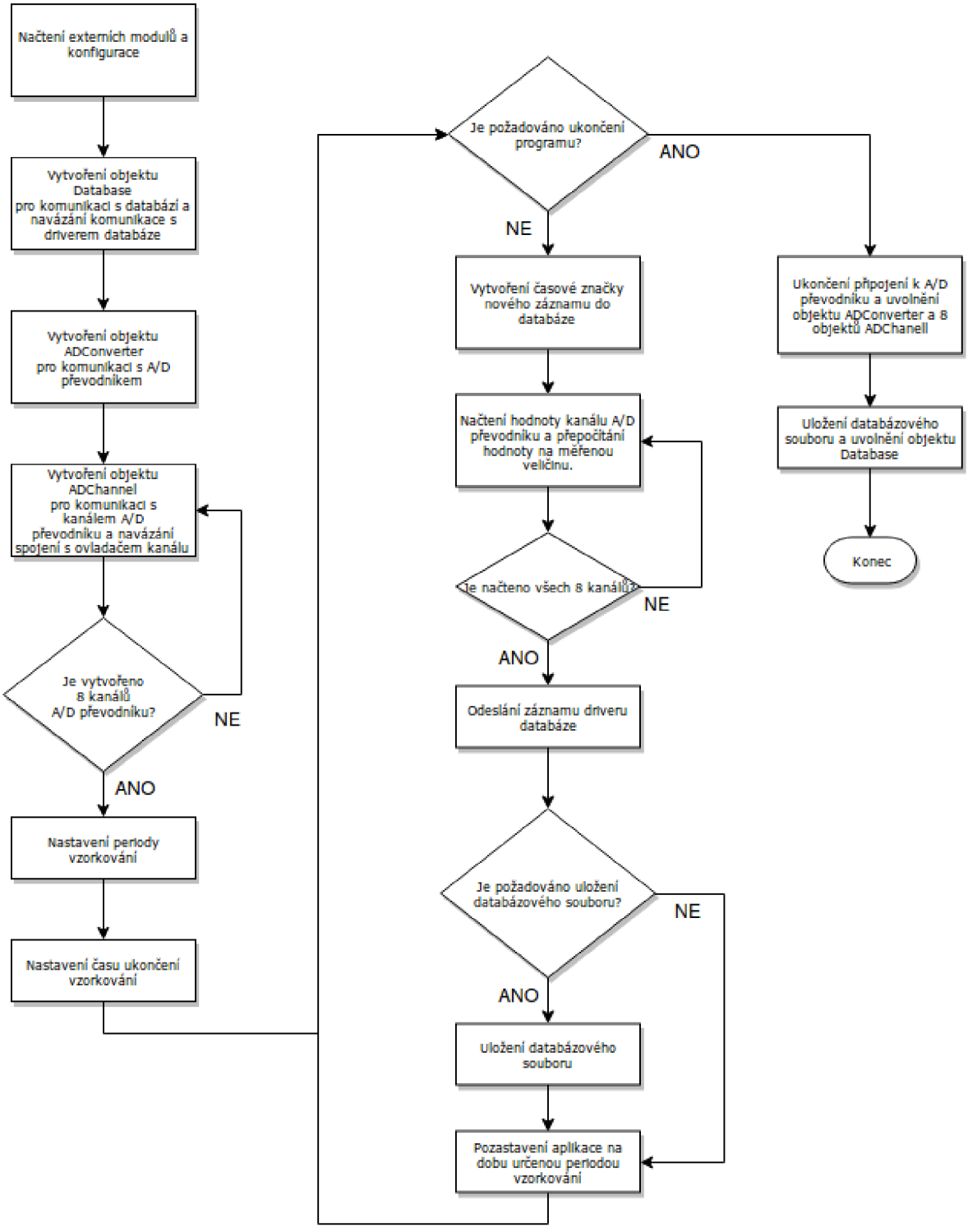
1. Zkopírování Ansible playbooku na zařízení pokud byl playbook po instalaci odstraněn.
2. Provedení požadovaných změn v konfiguraci playbooku.
3. Opětovné spuštění skriptu `install.sh`.
4. Změny se projeví ihned po ukončení skriptu. Restart celého zařízení není v tomto případě již potřeba, protože restart je nutný pouze k aktivaci podpory sběrnic SPI a I^2C, které jsou již zapnuté z prvotní instalace.

Skript `install.sh` můžeme spouštět opakovaně kolikrát chceme, protože vždy dojde pouze k aplikaci změn v konfiguraci a pokud ke změně v konfiguraci nedošlo, tak se žádné úkoly neprovádí.

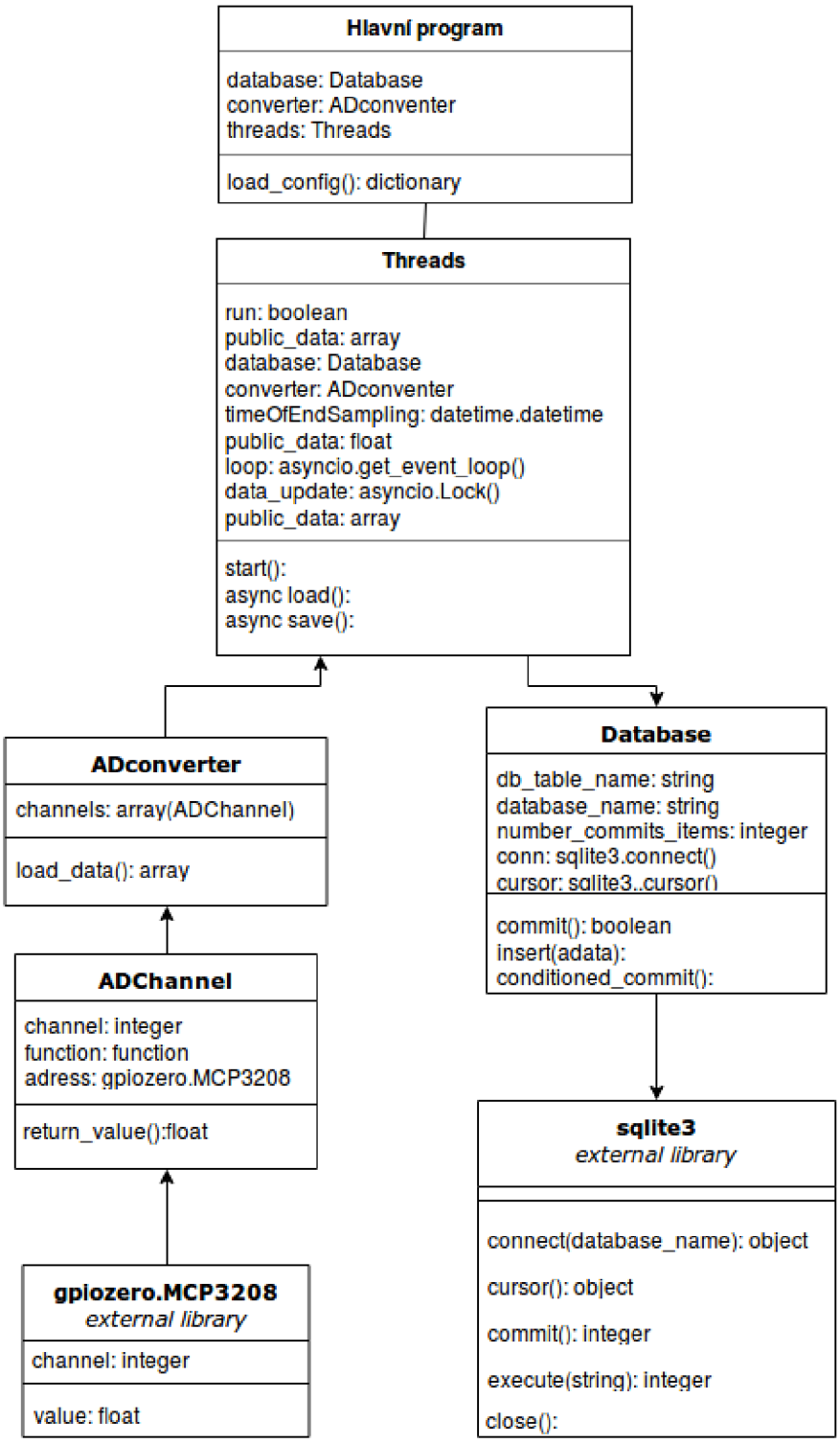
B SCHEMA SÉRIOVÉHO PROGRAMU



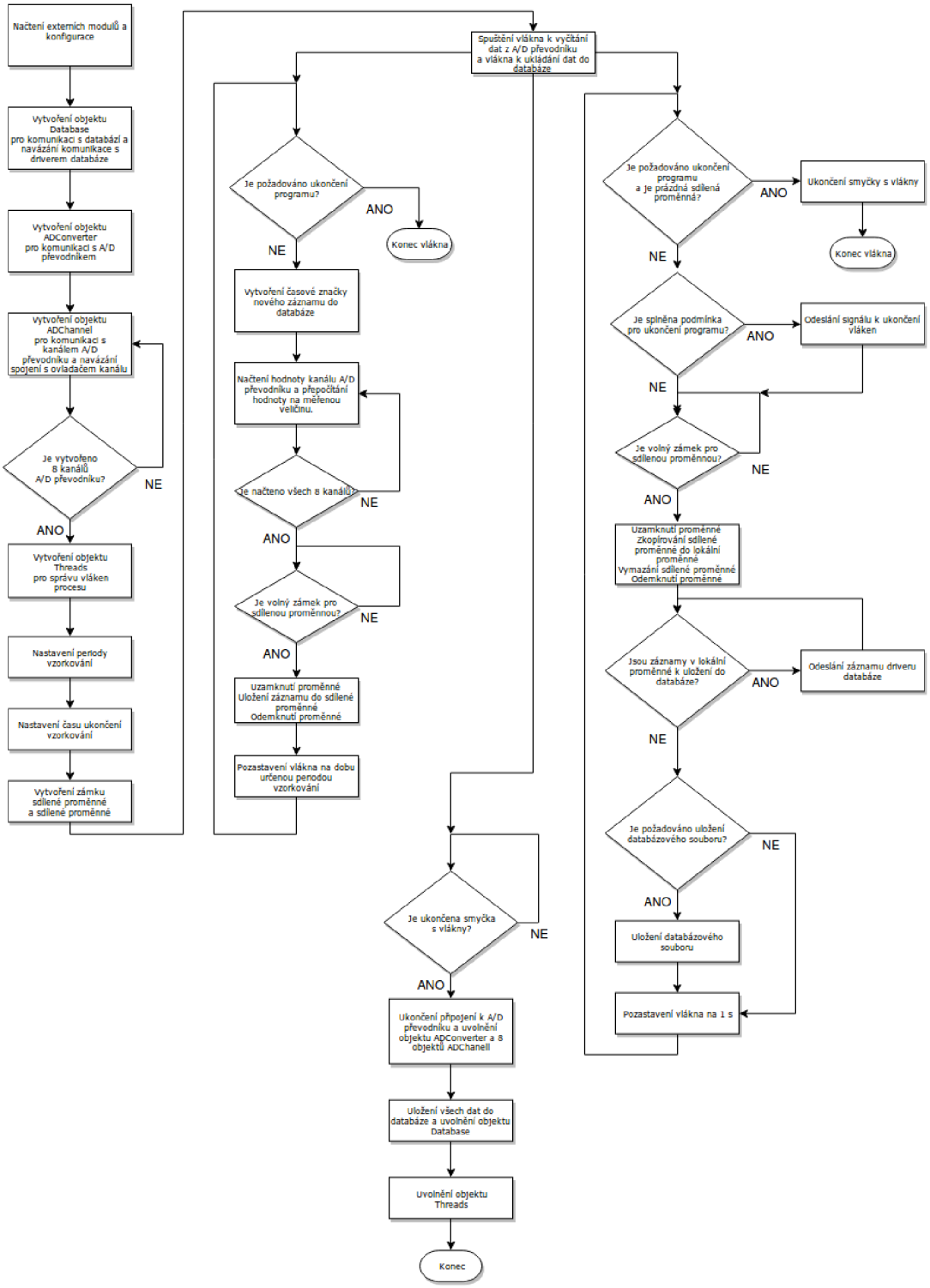
C DIAGRAM SÉRIOVÉHO PROGRAMU



D SCHEMA VLÁKNOVÉHO PROGRAMU

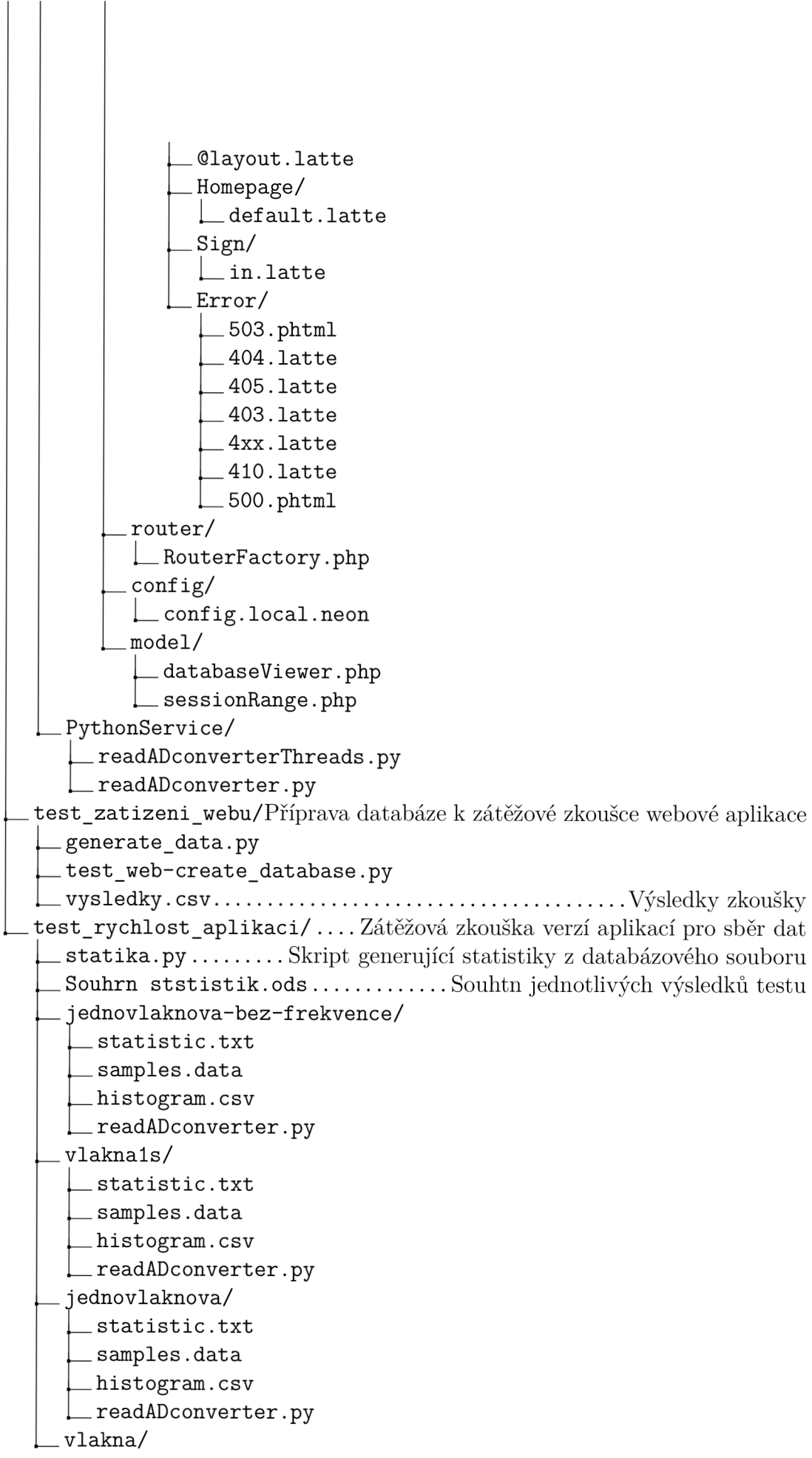


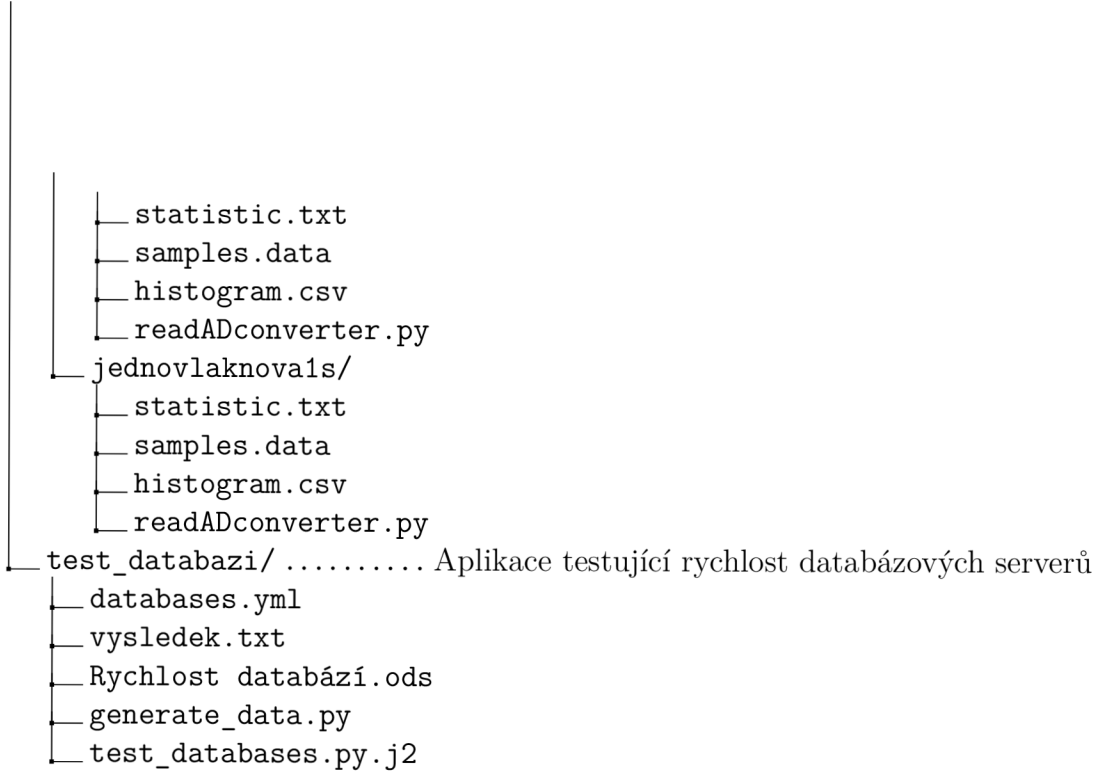
E DIAGRAM VLÁKNOVÉHO PROGRAMU



F OBSAH PŘILOŽENÉHO CD

- `/` ..... kořenový adresář přiloženého CD
  - `Diplomova_prace_Ciprys_Michal.pdf` ..... Text diplomové práce
  - `instalace_systemu_pro_sber_dat/` .. Instalační skripty a soubory zařízení
    - `AnsiblePlaybook.yml` .. Playbook aplikace Ansible pro instalaci zařízení
    - `install.sh` ..... Instalační skript zajišťující lokální instamaci zařízení
    - `Templates/`
      - `php.ini.j2`
      - `config.neon.j2`
      - `openssl.cnf.j2`
      - `ngnixVhost.j2`
      - `readADconverter.conf.j2`
      - `readADconverter.j2`
    - `wwwData/`
      - `composer.lock`
      - `composer.json`
      - `log/`
        - `web.config`
        - `.htaccess`
      - `temp/`
        - `web.config`
        - `.htaccess`
      - `www/`
        - `web.config`
        - `.htaccess`
        - `favicon.ico`
        - `robots.txt`
        - `index.php`
        - `images/`
          - `spinner.gif`
        - `css/`
          - `style.css`
        - `js/`
          - `main.js`
      - `app/`
        - `bootstrap.php`
        - `web.config`
        - `.htaccess`
        - `presenters/`
          - `Error4xxPresenter.php`
          - `SignPresenter.php`
          - `HomepagePresenter.php`
          - `ErrorPresenter.php`
          - `BasePresenter.php`
          - `templates/`


@layout.latte
Homepage/
default.latte
Sign/
in.latte
Error/
503.phtml
404.latte
405.latte
403.latte
4xx.latte
410.latte
500.phtml
router/
RouterFactory.php
config/
config.local.neon
model/
databaseViewer.php
sessionRange.php
PythonService/
readADconverterThreads.py
readADconverter.py
test_zatizeni_webu/Příprava databáze k zátěžové zkoušce webové aplikace
generate_data.py
test_web-create_database.py
vysledky.csv.....Výsledky zkoušky
test_rychlost_aplikaci/ ....Zátěžová zkouška verzí aplikací pro sběr dat
statika.py.....Skript generující statistiky z databázového souboru
Souhrn ststistik.ods.....Souhtn jednotlivých výsledků testu
jednovlaknova-bez-frekvence/
statistic.txt
samples.data
histogram.csv
readADconverter.py
vlakna1s/
statistic.txt
samples.data
histogram.csv
readADconverter.py
jednovlaknova/
statistic.txt
samples.data
histogram.csv
readADconverter.py
vlakna/


statistic.txt
samples.data
histogram.csv
readADconverter.py
jednovlaknova1s/
statistic.txt
samples.data
histogram.csv
readADconverter.py
test_databazi/ ..... Aplikace testující rychlost databázových serverů
databases.yml
vysledek.txt
Rychlost databází.ods
generate_data.py
test_databases.py.j2