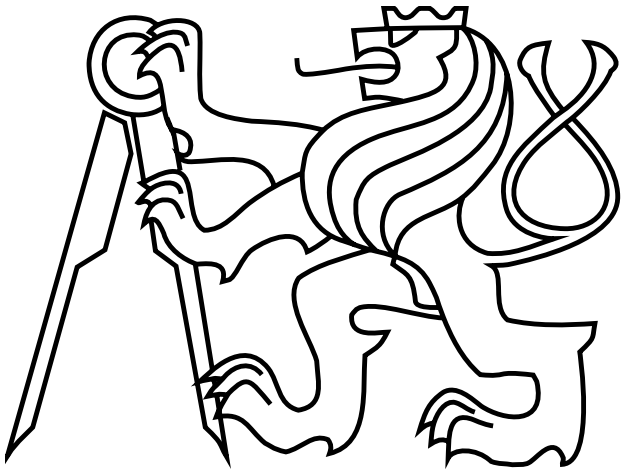
České vysoké učení technické v Praze

Fakulta elektrotechnická



DIPLOMOVÁ PRÁCE

Systém sběru dat s Raspberry Pi pro domovní automatizaci

Raspberry Pi based DAQ for building automation

Autor: Bc. Pavel Hübner

Vedoucí práce: doc. Ing. Jan Fischer, CSc.

Praha, 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Pavel Hübner**

Studijní program: Otevřená informatika (magisterský)
Obor: Počítačové inženýrství

Název tématu: **Systém sběru dat s Raspberry Pi pro domovní automatizaci**

Pokyny pro vypracování:

1. Navrhněte a realizujte vzorové řešení jednoduchého systému sběru dat pro domovní automatizaci s mikropočítačem Rapberry Pi, který umožní vzdálený monitoring a ovládání prostřednictvím rozhraním Ethernet. Systém zajistí sběr dat ze snímačů fyzikálních veličin (teplota, vlhkost, osvětlení, stav vzduchu,..) a dalších informací (přítomnost a pohyb osob ve sledovaném prostoru, ..), které jsou významné v domovní automatizaci.
2. Navrhněte a ověřte způsob připojení senzorových, indikačních, ovládacích, akčních a dalších potřebných prvků k systému pomocí procesorů řady STM32x. Pro připojení vzdálených snímačů využijte rozhraní RS-485.
3. Navrhněte a vytvořte potřebné programové vybavení pro systém.

Seznam odborné literatury:

[1] Yiu, J.: The definitive Guide to the ARM Cortex- M3. Elsevier, 2009
[2] Haasz, V., Roztočil, J., Novák, J.: Číslicové měřicí systémy, monografie ČVUT, 2000
[3] Ripka, P., Ďaďo, S., Kreidl, M., Novák, J.:Senzory a převodníky, ČVUT, 2005, ISBN 80-01-03123-3.

Vedoucí: doc.Ing. Jan Fischer, CSc.

Platnost zadání: do konce letního semestru 2014/2015

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 28. 1. 2014

Čestné prohlášení autora práce

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 10. května 2015 .....

Podpis autora práce

Abstrakt

Tato práce se zabývá návrhem a realizací jednoduchého systému sběru dat pro domovní automatizaci založeném na mikropočítači Raspberry Pi. Raspberry Pi je v systému na pozici centrální jednotky, která sbírá od satelitních jednotek data významná v domovní automatizaci (teplota, relativní vlhkost, intenzita osvětlení, koncentrace plynů, přítomnost a pohyb osob). Satelitní jednotky jsou osazeny mikrokontroléry STM32F050 s jádrem Cortex-M0, ke kterým jsou připojeny senzory výše zmíněných veličin a informací přes různá rozhraní – I^2C, UART, AD převodník. Všechny jednotky spolu komunikují přes sběrnici RS485 pomocí protokolu Modbus.

V práci je také realizováno vyhodnocování přítomnosti a množství osob ve sledovaném prostoru pomocí CMOS obrazového senzoru Raspberry Pi Camera Board. Obraz z kamery je zpracován algoritmem Codebook založeném na metodách modelování pozadí (background subtraction) s využitím knihovny OpenCV.

Abstract

This thesis proposes and implements a simple model solution of data acquisition system for building automation based on Raspberry Pi minicomputer. Raspberry Pi stands as a central unit in the proposed system. The goal of the central unit is to collect data important in building automation (such as the temperature, the relative humidity, the light conditions, the concentrations of gases, the presence and the movement of people). These data are provided by peripheral units which contain 32-bit STM32F050 Cortex-M0 microcontrollers. The sensors are connected to microcontrollers via various interfaces such as I^2C, UART and AD Converter. All units communicate with each other over RS485 bus where Modbus protocol is implemented.

The second part of this thesis deals with a movement detection in the monitored area by using CMOS image sensor Raspberry Pi Camera Board. The image provided by camera is processed by Codebook algorithm which is based on the background subtraction methods. The implemented Codebook algorithm uses OpenCV library.

Obsah

Seznam obrázků

Seznam použitých zkratek

ADC	Analog-to-Digital Converter
ADU	Application Data Unit
AF	alternativní funkce
AIR	Active Infrared Sensor
CO_2	oxid uhličitý
CSI	Camera Serial Interface
DMA	Direct Memory Access
DSI	Display Serial Interface
GP	General Purpose
GPIO	General Purpose Input/Output
HVAC	Heating, Ventilating and Air Conditioning
I^2C	Inter-Integrated Circuit
LED	Light-Emitting Diode
MCU	Microcontroller Unit
MIPI	Mobile Industry Processor Interface
OTG	On-The-Go
PDU	Protocol Data Unit
PIR	Passive Infrared Sensor
PWM	Pulse Width Modulation
RPi	Raspberry Pi
RTU	Remote Terminal Unit
SoC	System on a chip
SPI	The Serial Peripheral Interface
STM	STMicroelectronics
UART	Universal Asynchronous Receiver/Transmitter

1 Úvod

Systémy sběru dat v obytných a kancelářských prostorech jsou v dnešní době nedílnou součástí každé nové stavby. V budovách se tyto systémy využívají ke sledování stavu vnitřního prostředí. V návaznosti na měřicí systém je možné vytvořit systém regulující mikroklima interiéru budov jak proto, aby byla snížena energetická náročnost budovy, a tím byla zaručena návratnost prvotní výraznější investice do technického zázemí budovy, tak především je mikroklima budov sledováno a řízeno proto, aby se lidé uvnitř cítili spokojeně.

Technologie zabývající se těmito požadavky jsou označovány akronymem HVAC (z anglického **H**eating, **V**entilation and **A**ir **C**onditioning) a mezi jejich základní funkce patří udržování optimálního mikroklima interiéru přívodem čerstvého vzduchu, jeho výměnou či filtrací, zvlhčováním a odvlhčováním nebo vytápěním a chlazením. Součástí HVAC k této činnosti musí být systém, který zajistí sběr dat ze snímačů fyzikálních veličin (teplota, vlhkost, osvětlení, kvalita vzduchu atd.) i dalších informací jako je přítomnost a pohyb osob ve sledovaném prostoru. Systém, který dokáže určit stav řízených technologií (např. polohy ventilů a klapek) a který umožní získaná data archivovat a vizualizovat jak pro potřeby obsluhy, tak pro provádění analýz a hledání trendů. To umožní předvídat mikroklima v budově a reagovat preventivně.

Na českém trhu existuje řada systémů sběru dat nejenom v rámci HVAC, ve většině případů jsou řešení poskytovaná komerčními subjekty uzavřená jak ze strany softwaru, tak ze strany hardwaru. Je to způsob, jakým se výrobce zaručuje za deklarované vlastnosti, bezpečnost a spolehlivost svých produktů. Výsledné řešení systému sběru dat je drahé (cena samotného vybavení se pohybuje v řádech desítek až stovek tisíc korun), neuniverzální a neflexibilní (výrobce nabízí několik produktů a senzorů pro standardní situace) a autonomní – systém ve většině případů nedokáže spolupracovat s jiným systémem, který je již v budově přítomen a který např. zajišťuje manuální až částečně automatické řízení dílčí části vnitřního prostředí budovy.

Systém sběru dat pro domovní automatizaci, jenž bude navržen a realizován v této práci, si klade za cíl překonat některé z nedostatků komerčních řešení. V návrhu bude kladen důraz na jednoduchost systému, jeho snadnou škálovatelnost, univerzálnost a flexibilitu za pomoci otevřeného softwaru a hardwaru a dokumentace vytvořené tak, aby systém mohl použít kdokoli. Otevřenost platformy také napomůže spolupráci s dalšími systémy technického zařízení stavby, které regulují mikroklima budovy. V neposlední řadě bude kladen důraz na cenovou nenáročnost jednotlivých komponent systému sběru dat pro domovní automatizaci.

2 Rozbor a návrh systému

Před započetím návrhu systému je nezbytné stanovit základní pojetí a cíle, jejichž naplnění bude požadováno. V této kapitole jsou rozebrány hlavní rysy navrhovaného systému.

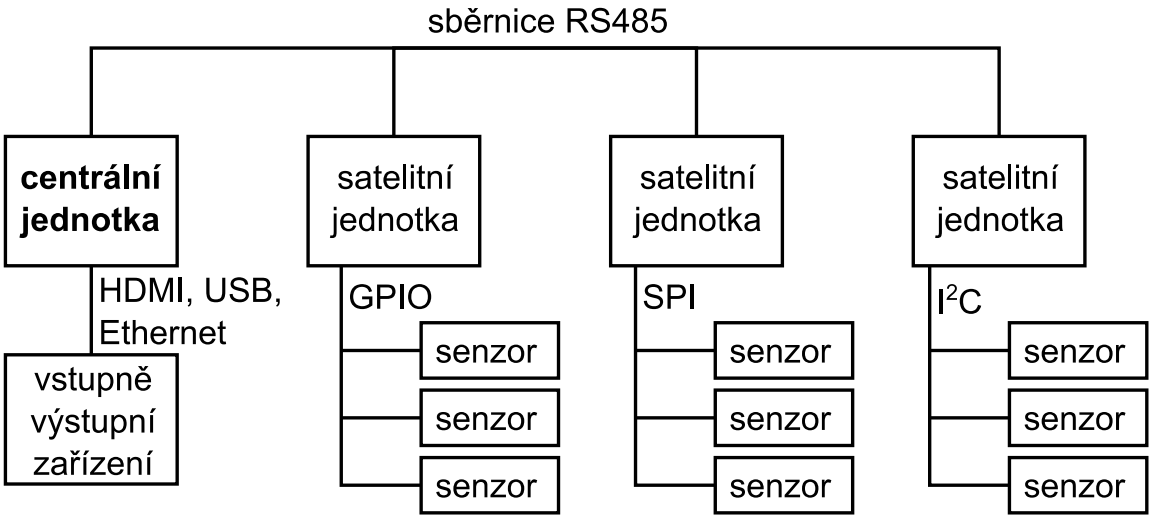
Zásadní funkcí systému sběru dat je schopnost data získávat a archivovat je pro pozdější vizualizaci, analýzu, hledání trendů a periodičností v mikroklimatu budovy. Při nalezení periodičnosti či korelace mezi dvěma a více veličinami je možné předvídat stav vnitřního prostředí a reagovat na situaci předem. Reakce na situaci předem je výhodná z mnoha pohledů – systém se dopředu vyvaruje situací, kdy by pro okamžitou regulaci mikroklimatu bylo vyžadováno více energie oproti regulaci průběžné. Systém také může naplánovat energetické výdaje budovy do časových úseků, kdy je snížena sazba za elektrický proud. Kromě toho se lidé uvnitř budovy nedostanou do situace, kdy by se cítili negativně ovlivněni prostředím.

Jedna centrální jednotka může data snímat, zpracovávat, archivovat, převádět do uživatelsky čitelné podoby a vizualizovat je. Jednotka má mít výpočetní výkon dostatečný k provádění těchto operací a zároveň mít nízkou spotřebu elektrické energie, protože u sledovacích systémů je předpokládán neustálý chod zařízení. Jednotka také musí obsahovat jak periferie vhodné ke komunikaci s vnějším světem (Ethernet, HDMI/DVI/VGA, USB), tak periferie jednodušší uzpůsobené k připojení různorodých senzorů (rozhraní SPI, I^2C, UART, programovatelné GPIO piny). Pro zachování jednoduchosti platformy sběru dat je vhodné, aby byla centrální jednotka vybavena systémovým software, který zajišťuje chod hardware a usnadňuje vývoj dalšího software.

Tyto požadavky splňují miniaturní jednodeskové počítače, které se objevily na trhu v nedávné době. Jedná se o počítače plnohodnotné, vyznačují se relativně vysokým výpočetním výkonem a nízkou pořizovací cenou a jsou přizpůsobeny pro běh operačních systémů typu GNU/Linux. Rovněž tyto počítače mívají rozmanité vstupně výstupní periferie a pro trvalé uložení dat používají flash paměti v podobě paměťových karet nebo USB klíčů. Jedním z těchto jednodeskových počítačů je populární Raspberry Pi, další představitel této kategorie je průmyslověji zaměřený BeagleBone Black.

Raspberry Pi tedy umožňuje získávání, zpracování a prezentaci dat. Nicméně, systémy sběru dat bývají rozsáhlé a není možné připojit veškeré senzory přímo k jednomu počítači, navíc by tak byla výrazně omezena škálovatelnost systému počtem pinů nebo rozhraní, ke kterým je možné snímače připojit. Proto je návrh systému rozšířen o satelitní jednotky, které budou umístěny v různých místnostech budovy, kde budou monitorovat vnitřní prostředí.

Na satelitní jednotku jsou kladeny podobné požadavky jako na jednotku centrální – měla by být vybavena různými rozhraními, která umožňují základní připojení senzorů, a také rozhraními pro propojení s jednotkou centrální (např. UART nebo Ethernet), aby mohla naměřená data odeslat ke zpracování a archivaci. Opět by měla disponovat výpočetním výkonem dostatečným ke sběru a odesílání dat při zachování nízké pořizovací ceny a fyzickými rozměry být malá, aby ji bylo možné spolu se senzory umístit na téměř libovolné místo.



Obr. 2.1: Topologie systému sběru dat

Tomuto popisu vyhovují např. 32-bitové mikrokontroléry STM32F0 společnosti STMicroelectronics s jádrem ARM Cortex-M0, které jsou navíc poměrně rozšířené, cenově velmi příznivé a spolu s podrobnou dokumentací a rozsáhlou podporou různých vývojových prostředí je snadné vytvořit potřebné programové vybavení satelitní jednotky.

Jednotky v systému sběru dat musí být schopny spolu komunikovat a data si vyměňovat. Společné rozhraní, kterým disponuje zmíněná centrální i satelitní jednotka a které umožňuje vzájemné propojení, je UART (**U**niversal **A**synchronous **R**eceiver and **T**ransmitter). Za pomoci sériového rozhraní UART je lze vytvořit propojení dle standardu RS422 nebo RS485 [1], kterým zařízení mohou komunikovat. Sběrnice RS485 zaručuje spolehlivost komunikace do 1200 metrů při dodržení maximální modulační rychlosti a připojení až 32 zařízení – omezení 32 zařízení lze obejít segmentováním sítě izolovaným opakovačem. Zmíněné mikrokontroléry společnosti STM podporují sběrnici RS485 už na úrovni hardware a navíc podporu rozšiřují o jednoduchý a robustní protokol Modbus [2], který byl navržen pro výměnu zpráv na sériové lince v průmyslovém prostředí a který je v linuxovém prostředí podporován ze strany knihoven třetích stran.

Topologie sítě navrhovaného systému sběru dat se standardem RS485 je tedy sběrnice (bus), v tomto pojetí komunikace může docházet ke kolizím během výměny dat mezi prvky, proto je nutné přístup ke sdílenému médiu řídit. Řízení přístupu se může ujmout centrální jednotka, která je v terminologii Modbus protokolu nazývána klientem, satelitní jednotky poskytující naměřená data se označují jako server a samy od sebe data vysílat nemohou. V síti tak bude

jen klient, který se bude serverů dotazovat na data, a ke kolizím během komunikace nebude docházet. K satelitním jednotkám bude připojena řada senzorů komunikujících různorodými rozhraními (GPIO, SPI, I^2C, …), mikrokontroléry data budou střádat a na výzvu centrální jednotky je zašlou pro potřeby archivace a vizualizace po sběrnici RS485 protokolem Modbus.

Vizualizace měřených dat je realizována ve webovém prohlížeči, který může být spuštěn přímo na centrální jednotce a grafický výstup bude poskytnut na obrazovce (obrazovka patří do souboru vstupně výstupních zařízení dle schématu topologie sítě výše) připojené rozhraním HDMI/DVI. Kromě prohlížeče je spuštěn na centrálním prvku webový server a ve spojení s připojením do sítě Internet rozhraním Ethernet nebo Wifi je možné prohlédnout si vizualizaci naměřených dat, popř. vydávat pokyny k řízení systému z jakéhokoli zařízení připojeném přes Internet k centrální jednotce.

Použité senzory budou získávat data dvou druhů. První druh přímo souvisí s mikroklimatem budovy. Určení tohoto stavu zajistí senzory fyzikálních veličin (jako je teplota, vlhkost, intenzita osvětlení, kvalita vzduchu atd.) a čidla dalších informací (např. senzory detekující přítomnost osob jako je PIR nebo kamera). Druhý typ dat je stav zařízení, která regulují vnitřní prostředí (např. polohy ventilů, klapek a jiné).

Raspberry Pi umožňuje připojení kamery rozhraním CSI (**C**amera **S**erial **I**nterface), jako další tedy budou prozkoumány možnosti využití Raspberry Pi k detekci pohybu na základě komplexní informace poskytované obrazovým senzorem.

2.1 Stanovení cílů práce

Jednotlivé kroky návrhu a realizace jednoduché, otevřené, snadno škálovatelné a univerzální platformy sběru dat za účelem sledování mikroklimatu interiéru a případné domovní automatizace jsou následující:

- Seznámení se s počítačem Raspberry Pi, s jeho hardwarovými a softwarovými možnostmi, prozkoumat alternativy tohoto PC (jiné jednodeskové počítače, které mohou mít vhodnější parametry, např. větší výpočetní výkon, nižší spotřebu apod.).
- Vytvoření a zvolení vhodného softwarového vybavení jednodeskového počítače Raspberry Pi počínaje volbou operačního systému, instalací potřebných knihoven a nízkoúrovňových ovladačů pro zpřístupnění periferií, přes konfiguraci webového serveru, který bude zprostředkovávat naměřená data za pomoci vytvořených skriptů, až po napsání programu řídicím komunikaci se satelitními jednotkami, který bude data zaznamenávat.
- Detekování pohybu obrazovým senzorem připojeným k Raspberry Pi. Cílem tohoto bodu je zjistit, zda může Raspberry Pi sloužit jako satelitní jednotka detekující pohyb za využití algoritmu založeného na metodách modelování pozadí.
- Seznámení se s mikrokontroléry společnosti STMicroelectronics, jejich periferiemi, a možnostmi programování. Taktéž je nezbytné vytvořit programové vybavení mikrokontroléru pro realizaci komunikace s centrálním prvkem a se senzory.

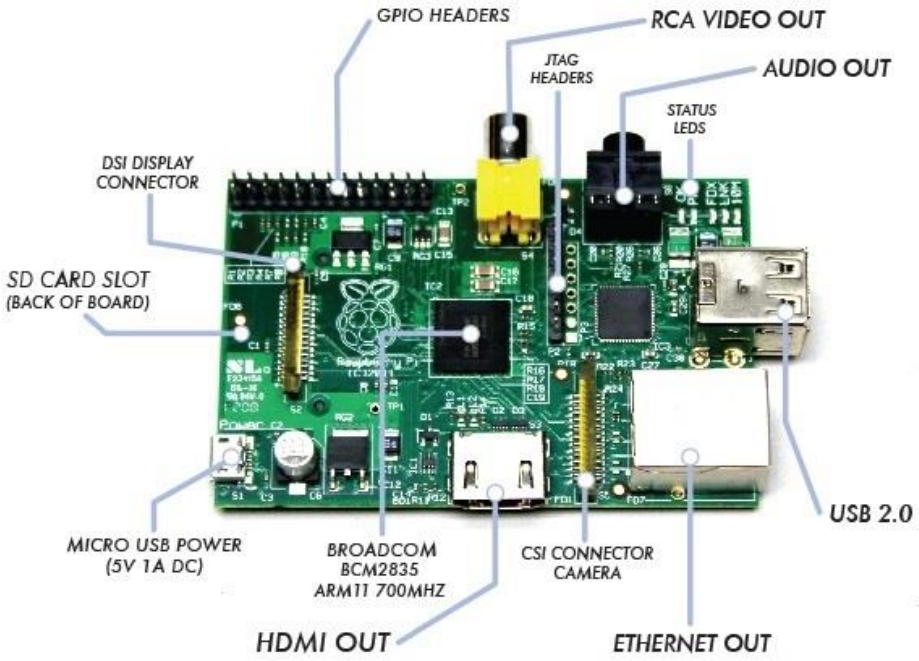
- Prozkoumat a vybrat senzory vhodné ke sběru dat v kancelářských, příp. domovních prostorech, nastudovat jejich možnosti, navrhnout a realizovat zapojení senzorů a jejich pomocných obvodů do systému.
- Nastudování a realizace komunikace na sběrnici RS485 s protokolem Modbus, jenž bude využit k mezijednotkové komunikaci. A nastudování dalších rozhraní k připojení senzorů (SPI, I^2C apod.).

Výsledkem této práce bude univerzální měřicí systém reprezentován demonstrační deskou osazenou jedním centrálním prvkem a satelitními jednotkami s připojenými senzory. Vzájemná komunikace bude probíhat na sběrnici RS485 s využitím protokolu Modbus. Systém bude snímat teplotu ovzduší a relativní vlhkost vzduchu, měřit množství CO_2, kvalitu vzduchu, úroveň osvětlení a detekovat pohyb osob v místnosti.

3 Raspberry Pi

Tato kapitola se věnuje výpočetnímu modulu Raspberry Pi a jeho začlenění do této práce. Kapitola začíná stručným popisem historie RPi, pokračuje výčtem jeho možností a alternativních modulů, kterými může být RPi nahrazen. Dále se zaobírá výběrem operačního systému a jeho prvotním spuštěním. Závěr kapitoly je vyhrazen popisu postavení Raspberry Pi v systému sběru dat.

Raspberry Pi je plnohodnotný, miniaturní, jednodeskový počítač vyvinutý britskou nadací Raspberry Pi Foundation s cílem podpořit výuku informatiky na středních školách [3]. Inženýři nadace navrhovali Raspberry Pi s vizí malého, programovatelného počítače se snadno dostupnými a ovladatelnými periferiemi tak, aby se žáci naučili software nejenom vytvářet, ale aby také pochopili hardware a porozuměli, jak je počítač postaven a jak pracuje. Cíle byly jednoduché - realizovat levný a malý počítač, aby každý žák mohl mít vlastní výpočetní modul a mohl si vše sám vyzkoušet [4].



Obr. 3.1: Raspberry Pi, model B a rozložení periferií

Ve své snaze uspěli – Raspberry Pi, model B (obr. 3.1), počítač o přibližných plošných rozměrech kreditní karty, určený pro běh optimalizovaných operačních systémů na bázi GNU/Linux, se na trhu objevil 29. února 2012 a byl ve své době revoluční především poměrem ceny a výkonu. Za US$35 (přibližně 900 Kč) bylo možné koupit počítač osazený SoC Broadcom BCM2835

integrující procesor ARM11 taktovaný na frekvenci 700 MHz a grafický procesor schopný dekódovat video v FullHD kvalitě v reálném čase. Na desce je k dispozici také operační paměť o velikosti 512 MB a různorodé periferie a sběrnice, které mají za úkol zpopulárnit výuku informatiky. O rok později, 4. března, Raspberry Pi Foundation uvedla na trh model A, oproti předchozímu modelu má poloviční velikost operační paměti, jeden USB port místo dvou a Ethernetové rozhraní nebylo integrováno. Touto redukcí hardwaru se snížila cena modelu na US$25 a klesla energetická spotřeba modulu z 3,5 W na 2,5 W [5].

Mezitím se Raspberry Pi stalo ve světě doopravdy populární, nejenomže splnilo původní záměr v zatraktivnění výuky informatiky, ale příznivci Raspberry Pi využívají tento miniaturní počítač pro všechno možné od domácího serveru, přes řídicí jednotku robota, kvadroptéky či domovní automatizace (a tak budou moci využít systém v této práci navrhovaný) až k metrologickým stanicím umístěným porůznu po světě [6], automatizovaným časosběrným fotoaparátům či multimediálním centrům [7]. Raspberry Pi se dokonce svými rozměry, cenou, univerzálností, možnostmi vývoje a otevřeností linuxového prostředí uplatnilo v průmyslu (reakcí nadace Raspberry Pi Foundation je otevřená platforma Raspberry Pi Compute Module [8] uvedená na trh v dubnu 2014 za US$200, jež návrh a vývoj software a hardware usnadňuje). Další změnou je větší počet GPIO pinů oproti uživatelskému RPi, modul je celkově menší a má 4GB eMMC Flash paměť integrovanou přímo na čipu atd.

Dne 15. července 2014 vydala nadace revizi modelu B s označením model B+. Uživateli nabízí za cenu shodnou s modelem B dva USB porty navíc, patici GPIO rozšířenou o čtrnáct pinů (z 26 na 40), nižší spotřebu a operační systém se nově nahrává z MicroSD karty oproti SD kartě v předchozích dvou verzích. Počet prodaných kusů všech modelů Raspberry Pi ke konci února roku 2015 překonal číslo 4,5 miliónů.

3.1 Parametry Raspberry Pi

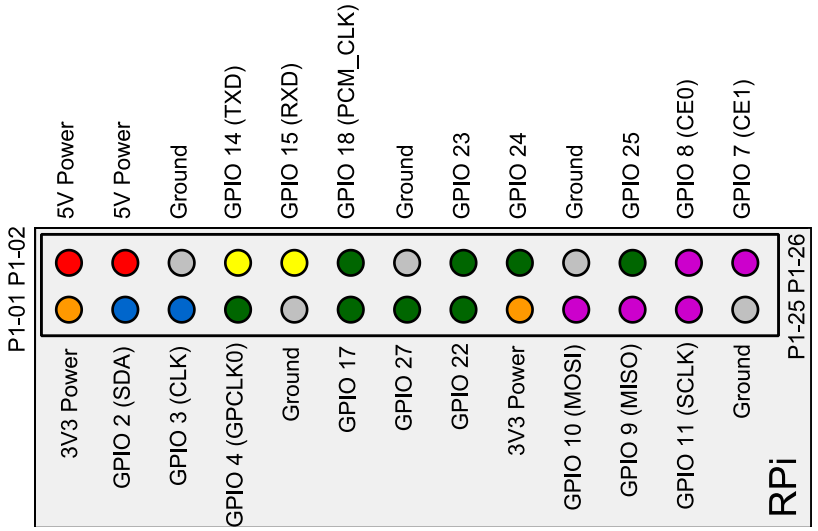
V tabulce níže jsou uvedeny parametry všech modelů Raspberry Pi, v této práci je použit model B. Podrobněji popsané parametry jsou k nalezení na stránkách Embedded Linux Wiki [5].

	Model A	model B	model B+
cena	US$25	US$35	US$35
system-on-a-chip	Broadcom BCM2835 (obsahuje CPU a GPU, SDRAM je odděleně)		
CPU	ARM1176JZF, 700 MHz, ARM11		
GPU	Broadcom VideoCore IV		
SDRAM	256 MB	512 MB	
USB 2.0 porty	1 (integrovaný v SoC)	2 (přes USB hub)	4 (přes USB hub)
video výstupy	Kompozitní RCA, HDMI		RCA vyžaduje adaptér
audio výstupy	3.5 jack, HDMI		
paměť	SD karta		MicroSD karta
síťové rozhraní	žádné	10/100 Ethernet RJ45	
low-level periferie	26 GPIO, SPI, I^2C, I^2S, UART		40 GPIO, navíc I^2C IDC
real-time-clock	žádné		
spotřeba	2,5 W	3,5 W	3,0 W
velikost	85,0 x 56,0 x 15 mm	85,0 x 56,0 x 17 mm	
hmotnost	31g	40g	
další rozhraní	SCI (Camera Serial Interface), DSI (Display Serial Interface)		

Tabulka 3.1: Parametry modelů Raspberry Pi

3.2 GPIO piny Rasberry Pi

Raspberry Pi mimo složitějších rozhraní také disponuje dvaceti šesti programovatelnými GPIO (General-Purpose Input/Output) piny uspořádanými v patici P1 (v obr. 3.1 je označena jako GPIO Headers). Rozložení pinů v patici je 2x13 se standardní roztečí 2,54 mm. GPIO pinů bez alternativní funkce je osm, zbylé piny realizují SPI, I^2C a UART rozhraní, PWM modulaci a napěťové výstupy +3,3 V, +5 V a GND. Pokud programátor nebude využívat alternativní funkce pinů, které popisuje tabulka níže, má celkem k dispozici sedmnáct GPIO pinů. Piny nejsou 5 V tolerantní a jakékoli napětí vyšší než 3,3 V může nenávratně poškodit celé zařízení.



Obr. 3.2: Rozmístění GPIO pinů Raspberry Pi, model B, rev. 2.0, pohled shora

Alternativní funkce GPIO pinů zobrazuje tabulka níže. Zajímavostí je, že pravděpodobně se záměrem usnadnit výuku programování na školách, jsou na pinech určených ke komunikaci I^2C sběrnicí (piny P1-03 a P1-05) přímo připojeny pull-up rezistory o velikosti 1.8 kΩ vyžadované specifikací sběrnice.

Pin	Označení	Poznámka	Alt. funkce	Další alt. funkce
02	5V0	napájené skrz pojistku		
04	5V0	napájené skrz pojistku		
06	GND			
08	GPIO 14	alt 0 aktivní po resetu	UART0_TXD	ALT5 = UART1_TXD
10	GPIO 15	alt 0 aktivní po resetu	UART0_RXD	ALT5 = UART1_RXD
12	GPIO 18		PCM_CLK	ALT4 = SPI1_CE0_N ALT5 = PWM0
14	GND			
16	GPIO 23			ALT3 = SD1_CMD ALT4 = ARM_RTCK
18	GPIO 24			ALT3 = SD1_DAT0 ALT4 = ARM_TDO
20	GND			
22	GPIO 25			ALT3 = SD1_DAT1 ALT4 = ARM_TCK
24	GPIO 8		SPI_CE0_N	
26	GPIO 7		SPI_CE1_N	
01	3.3 V	50 mA max (1&17)		
03	GPIO 2	1K8 pull up rezistor	I2C0_SDA / I2C1_SDA	
05	GPIO 3	1K8 pull up rezistor	I2C0_SCL / I2C1_SCL	
07	GPIO 4		GPCLK0	ALT5 = ARM_TDI
09	GND			
11	GPIO 17			ALT3 = UART0_RTS ALT4 = SPI1_CE1_N ALT5 = UART1_RTS
13	GPIO 27		PCM_DOUT/reserved	[1]
15	GPIO 22			ALT3 = SD1_CLK ALT4 = ARM_TRST
17	3.3 V	50 mA max (1&17)		
19	GPIO 10		SPI0_MOSI	
21	GPIO 9		SPI0_MISO	
23	GPIO 11		SPI0_SCLK	
25	GND			

Tabulka 3.2: Alternativní funkce GPIO, patice P1, RPi, model B, rev. 2.0

Legenda
+5 V
+3.3 V
GND, 0 V
UART
GPIO
SPI
I^2C

Tabulka 3.3: Legenda tabulky funkce pinů RPi

K usnadnění ovládání pinů vytvořila komunita kolem Raspberry Pi několik knihoven v různých programovacích jazycích. Jedna z nejznámějších knihoven zpřístupňující základní funkce GPIO pinů multimediálního procesoru BCM2835 je WiringPi [9] napsaná v jazyce C. Obsáhlejší knihovna, také v jazyce C s pokročilejšími funkcemi dávající programátorovi větší volnost (např. přístup k systémovým časovačům či řízení SPI rozhraní), je pojmenována BCM2835.h [10].

V případě nedostatku GPIO pinů je možné rozšířit počet pinů IO expandéry. Oficiální populární expandér je deska Gertboard osazená mikrokontrolérem ATmega, ten znásobuje množství GPIO pinů a přidává mikrospínače, budiče s otevřeným kolektorem, výstupy pro řízení motorů a AD a DA převodníky. Expandér se připojuje přímo na patici P1.

[1] ALT4 = SPI1_SCLK ALT5 = GPCLK1 / ALT3 = SD1_DAT3 ALT4 = ARM_TMS

3.3 Alternativy k Raspberry Pi

Před uvedením Raspberry Pi na trh existovalo několik jednodeskových počítačů podobných RPi, a však jejich cena byla několikanásobně vyšší. Až spolu s prosazením Raspberry Pi se objevilo na trhu několik miniaturních počítačů se srovnatelným až několikanásobně lepším poměrem cena/výkon schopných rozběhnout operační systém s obdobnými periferiemi. Níže je uvedeno několik jednodeskových počítačů, jejichž cena nepřesáhla US$55, a které by bylo možné v této práci také použít namísto RPi.

3.3.1 Banana Pi

Banana Pi [11] je jednodeskový počítač uvedený na trh v březnu roku 2014 za cenu US$45. Na první pohled je zřejmá jistá podoba s Raspberry Pi, avšak jeho vývoj není přímo napojen na Raspberry Pi Fundation a také hardwarová specifikace je odlišná. Banana Pi je osazen SoC Allwinner A20 (dvoujádrový procesor Cortex A7, 1 GHz, grafický procesor Mali-400MP2) a operační pamětí 1GB DDR3. Mezi rozhraní Banana Pi patří slot pro SD kartu, SATA konektor, HDMI, kompozitní RCA, 3,5 mm jack, mikrofon, RJ45 1GB Ethernet, dva USB 2.0 porty, MicroUSB (OTG), IR přijímač, SCI, DSI, GPIO piny s funkcemi UART, I^2C, SPI, CAN, ADC, PWM, tři neprogramovatelné mikrospínače (jejichž funkce je zapnutí, reset a výběr média pro bootování). Počítač svůj stav indikuje pomocí dvou LED a jedné LED programovatelné.

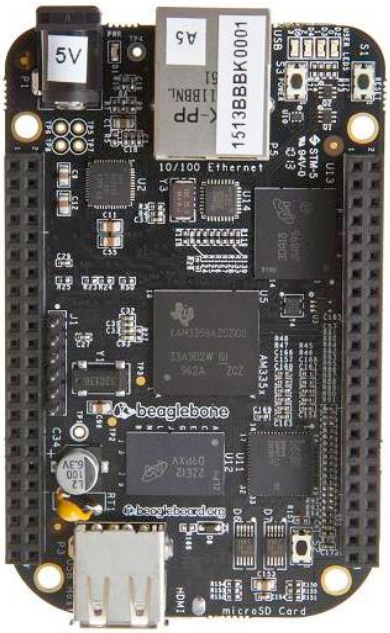
Obr. 3.3: Banana Pi

Oproti Raspberry Pi tak nabízí více už v základu jak po stránce vývoje hardwaru, tak softwaru a zdá se být vhodnější pro výuku informatiky, proti mluví jen jeho vyšší cena. Operačních systémů vhodných pro Banana Pi není mnoho, přesto mezi ně patří zejména Raspbian, Arch Linux, Fedora a pár jiných, s delším výskytem počítače na trhu by se měly objevovat další přizpůsobené OS.

3.3.2 Beaglebone Black

Beaglebone Black [12] byl na trh uveden v dubnu 2013 a nabízí výpočetní výkon srovnatelný s Raspberry Pi, za US$45 je možné zakoupit počítač osazený SoC TI Sitara AM335x (jednojádrový procesor Cortex A8, 1 GHz, grafický procesor PowerVR SGX530), 512 MB DDR3L RAM, flash pamětí 4 GB 8-bit eMMC. Mezi vstupně výstupní rozhraní Beaglebone Black patří Micro-

HDMI, 10/100 Ethernet, USB 2.0 A, a MiniUSB 2.0 B, devadesát dva GPIO pinů (4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I²C, ADC, 2x CAN, 4x časovač), čtyři programovatelné a dvě indikační LED. Rozměry desky počítače jsou 86 x 53 mm.



Obr. 3.4: Beaglebone Black, rev. A5A

Beaglebone Black je vybaven méně výkonným grafickým procesorem oproti Raspberry Pi, zato ho překonává v počtu zpracovaných operací za vteřinu díky vyšší taktovací frekvenci procesoru a jeho novější architektuře i vyšším počtem GPIO pinů (a alternativních funkcí pinů) spolu s výkonnějším procesorem činní z BeagleBone Black počítač vhodnější pro náročnější aplikace, pokud není požadována dekomprese videa v kvalitě 1080p30, která nadále zůstává doménou Raspberry Pi.

Výhodou počítačů BeagleBone z programátorského a vývojářského hlediska je hardwarová otevřenost produktu a podrobná dokumentace, obojím Raspberry Pi strádá (uzavřenost RPi mnohým nevyhovuje a snaží se to změnit – např. v březnu 2014 Broadcom zveřejnil kompletní dokumentaci grafického procesoru Video Core IV).

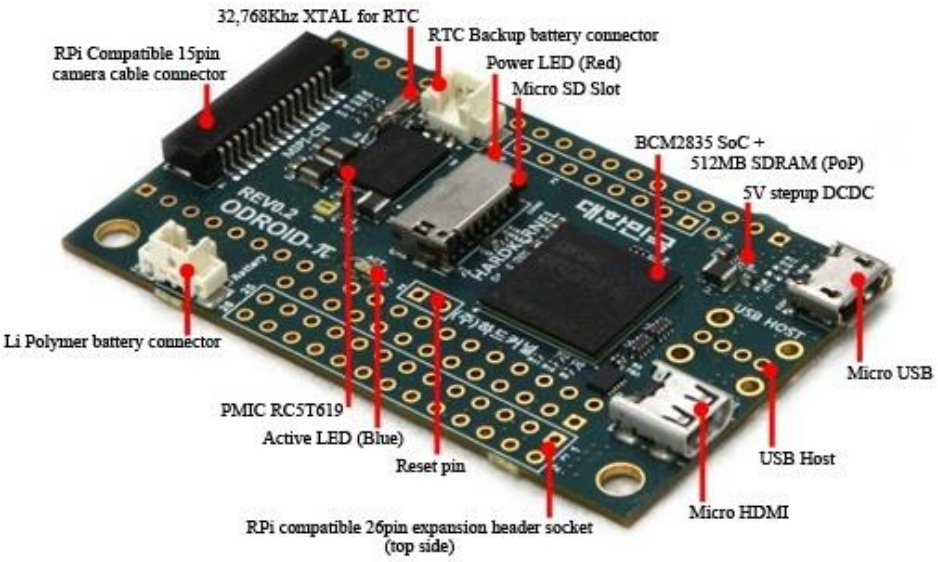
Operační systémy dostupné pro Beaglebone Black jsou Angstrom (distribuce vyvinutá výrobcem), Debian, Android, Ubuntu, Arch Linux, RISC OS a další.

3.3.3 ODROID-W

ODROID-W [13] počítačový modul byl uveden na trh v květnu 2014 za cenu US$30. Výrobce počítač propaguje jako plně softwarově kompatibilní s Raspberry Pi, to zaručuje i stejný SoC a rozložení GPIO pinů. ODROID-W v základu nabízí SoC Broadcom BCM2835 (CPU: ARM11, 700 MHz, GPU: Video Core IV), 512 MB DDR2 SDRAM, slot na MicroSD a eMMC, 1x USB 2.0, Micro-HDMI, třicet dva GPIO pinů (2x ADC oproti RPi), RTC, konektory pro připojení baterií pro RTC i pro napájení celého modulu, MIPI (také RPi kompatibilní). Nevýhodou je chybějící Ethernet rozhraní a celkově nižší hardware vybavenost, výrobce ale nabízí řadu rozšíření. Zajímavé je

zaměření na mobilní počítačový modul, čemuž odpovídají hodiny reálného času integrované na desce a konektor navržený k zapojení Lion baterie. Hmotnost zařízení při rozměrech 60 x 36 mm je 8 g.

Výrobce nabízí výkonnější verzi – miniaturní počítač ODROID-U3 – za US$65, a ten je osazený procesorem 1.7GHz Quad-Core, 2 GB DDR2, grafickým procesorem, třemi USB 2.0 porty, RJ45 Ethernetem, Micro-HDMI, GPIO a dalšími periferiemi a rozhraními.



Obr. 3.5: ODROID-W, PCB rev. 0.2

Na ODROID-W jsou spustitelné stejné operační systémy jako na RPi, jmenovitě Raspbian, Arch Linux, Pidora, RISC OS apod.

3.3.4 A10-OLinuXino-LIME

OlinuXino [14] je řada jednodeskových open-hardware počítačů navrhnutých bulharskou společností OLIMEX Ltd.

Prodej výpočetního modulu A10-OLinuXino-LIME byl zahájen v prosinci 2013 a za US$39 nabízí SoC Allwinner A10 (CPU: Cortex-A8, 1 GHz, GPU: Mali 400), 512 MB DDR3 RAM paměti, SATA, HDMI, 2x USB 2.0, 1x USB 2.0 OTG, RJ45 10/100 Ethernet, MicroSD slot, GPIO LED, LCD,160x GPIO (8x UART, 4x SPI, 3x Two-Wire, 2x PWM) konektor na LiPO baterii, tři programovatelné mikrospínače. Při doplacení US$13 je možné zakoupit verzi s označením A10-OLinuXino-LIME-4GB, která má, jak název napovídá, integrovanou NAND FLASH paměť o velikosti 4 GB. Rozměry desky jsou 84 x 60 mm.

Obr. 3.6: A10-OLinuXino-LIME

Dostupné operační systémy pro A10-OLinuXino-LIME jsou Android a Debian.

3.4 Operační systémy Raspberry Pi

Všechny výše zmíněné počítače jsou určené pro běh operačního systému, většinou na bázi GNU/Linuxu, a na většinu z nich je možné nainstalovat přinejmenším jeden ze systémů zde jmenovaných. Systémy se liší zejména v míře optimalizace pro danou architekturu procesoru. Níže jsou popsány operační systémy spolu s jejich hlavními klady a zápory (OS jsou nabízené distributorem Raspberry Pi a jsou ke stažení na oficiálních stránkách RPi).

NOOBS (**N**ew **O**ut **o**f **B**ox **S**oftware) – nejedná se o skutečný operační systém, ale o zaváděcí program zajišťující snadnou instalaci různých OS a jejich vyzkoušení. Uživatel dostane na výběr ze šesti OS a během každého spuštění Raspberry Pi si uživatel může vybrat jiný operační systém. OS nabízené aplikací NOOBS jsou následující:

- Raspbian
- Pidora
- OpenELEC
- RaspBMC
- RISC OS
- Arch Linux

Raspbian – derivace Debianu, verze Wheezy. Raspbian je nejpoužívanější a doporučovaný nadací Raspberry Pi Fundation, z tohoto důvodu stojí za Raspbianem obrovské množství uživatelů, kteří již vyřešili na veřejných fórech mnoho problémů, na které může uživatel při používání tohoto OS narazit. Podobně – jako Debian – je prezentován jako stabilní systém vhodný pro dlouhodobý provoz. Tento operační systém byl vybrán v této práci pro důvody uvedené výše.

OpenELEC – XBMC Mediacenter distribuce je ořezaný OS vhodný pro použití Raspberry Pi jako multimediálního centra. K přehrávání video souborů v kompresi MPEG-2 nebo VC-1 je nutné zakoupit kodeky, které jsou vázány na sériové číslo Raspberry Pi, a proto nejsou přenositelné.

RISC OS – GUI prostředí optimalizované pro běh při rozlišení 1080p kompilované pro architekturu ARM týmem lidí, kteří původně navrhli ARMové procesory. Nejedná se o derivaci linuxu.

Arch Linux – Distribuce Arch Linux zkompilovaná pro ARM zařízení, je vhodná pro zkušenější uživatele, a těm umožňuje přizpůsobit si systém bez nadbytečných součástí.

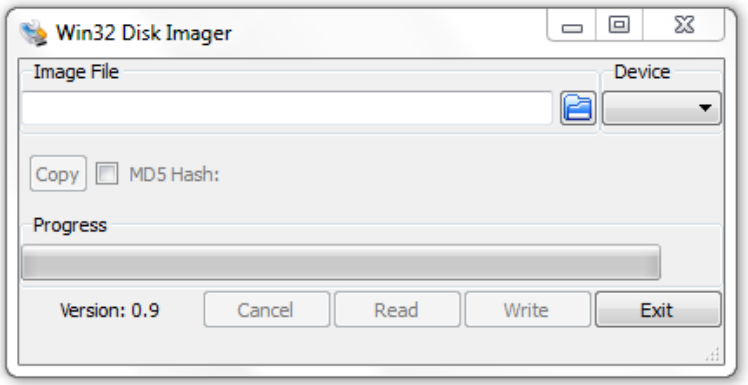
Rozsáhlý seznam operačních systémů přizpůsobených pro Raspberry Pi je k nahlédnutí na stránkách [5].

3.5 Instalace Raspbianu a první spuštění bez monitoru v OS Windows

Po výběru OS (zde Raspbian) je k instalaci systému na Raspberry Pi potřeba následující:

- Raspberry Pi
- Napájecí adaptér s micro-USB konektorem, 5 V, 700 mA – jako zdroj napájení je možné použít USB portu např. laptopu, nicméně ten poskytuje nejvýše 500 mA
- paměťová karta SDHC (doporučený je minimálně třída 4, velikost 4 GB)
- čtečka SDHC paměťových kartet
- síťový kabel (RJ45)
- připojení k Internetu

Po stažení image OS Raspbian ze sekce Download na stránkach http://www.raspberrypi.org/ se systém nainstaluje přímým binárním zápisem image souboru na SD kartu, k tomuto účelu slouží program Win32DiskImager. Po spuštění programu Win32DiskImager s oprávněním administrátora systému a vložení SD karty do čtečky se objeví obr. 3.7.



Obr. 3.7: Win 32 Disk Imager, ver. 0.9

Štítek Image File značí cestu k image souboru s Raspbianem a pod položku Device patří jednotka s SD kartou (např. G:/). Zápis image souboru na kartu se zahájí kliknutím na Write.

Po dokončení zápisu je možné na SD kartě změnit parametry systému Raspbian před spuštěním. Např. přidáním proměnné `ip` do souboru *cmdline.txt* v této formátu

```
ip=192.168.0.3:255.255.255.0
```

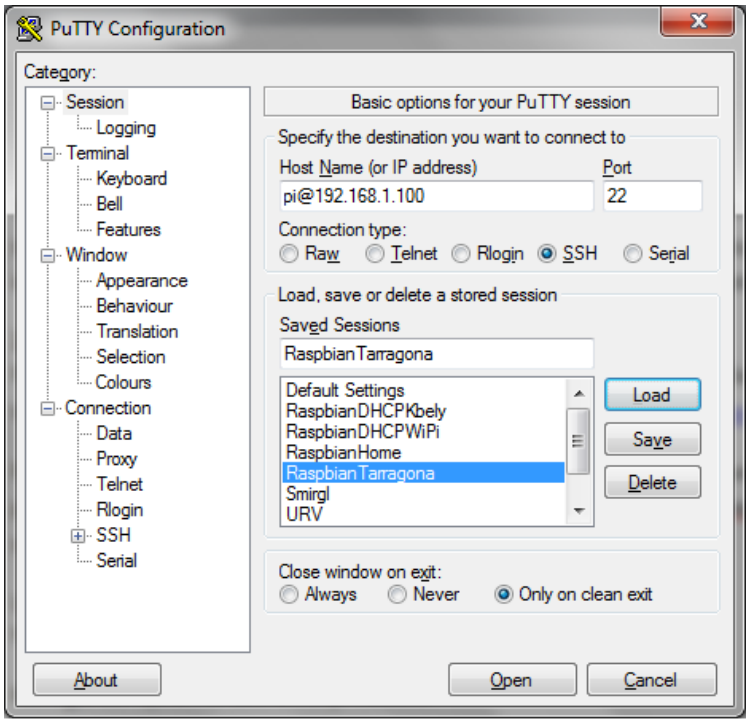
se zařízení přiřadí statická IP adresa, což umožňuje vytvoření LAN sítě mezi laptopem a Raspberry Pi a řízení RPi SSH protokolem. Soubor *config.txt* obsahuje nastavení týkající se grafických výstupů zařízení (např. nucený výstup na HDMI rozhraní, pokud RPi nedetekuje monitor) a je zde možná změnit frekvenci procesoru – pro tuto možnost je lepší nástroj raspi-config, viz níže.

Tímto je vše potřebné k prvnímu spuštění Raspbian na Raspberry dokončeno.

Po vložení SD karty do Raspberry Pi a připojení napájecího USB kabelu se úspěšné načítání systému z karty projevuje rychlým poblikáváním zelené LED diody ACT (ta signalizuje přístup na paměťovou kartu) a snížením jasu červené diody PWR. Diody jsou na obr. 3.1 vyznačené popiskem Status LED.

3.5.1 První spuštění Raspbianu pod OS Windows

Po připojení síťového kabelu a vytvoření lokální sítě se lze připojit k Raspberry Pi přes protokol SSH (**S**ecure **Sh**ell), ten realizuje např. program PuTTY (http://www.putty.org/). Do Host Name patří adresa zařízení (ta je nastavena v souboru *cmdline.txt* na SD kartě nebo je přiřazena DHCP serverem), řetězec `pi` před adresou značí uživatelské jméno, pod kterým se PuTTY identifikuje během navazování spojení. SSH protokol využívá port 22. Spojení se otevře tlačítkem Open.



Obr. 3.8: Program PuTTY

Po přijetí vygenerovaného klíče a přihlášení s následujícími údaji

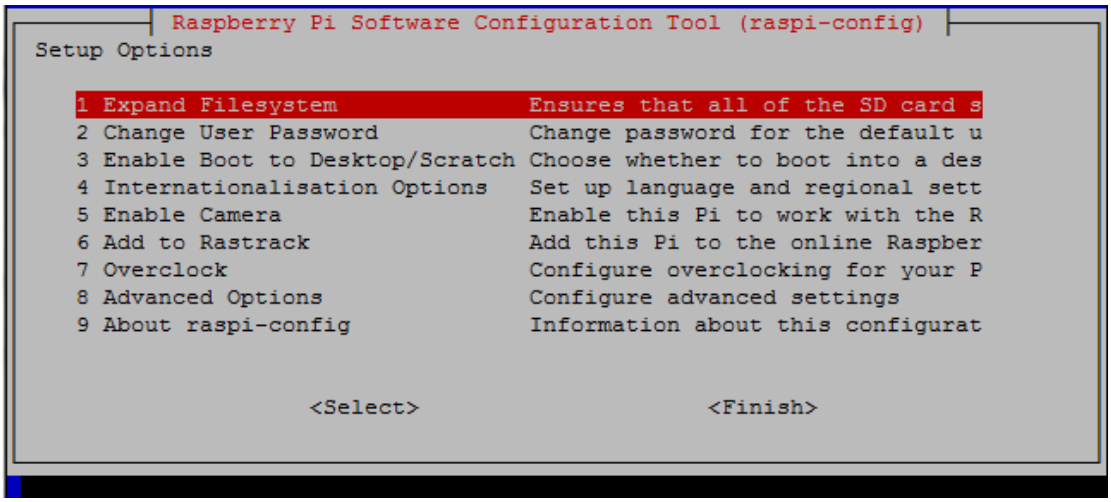
Uživatelské jméno: `pi`
Heslo: `raspberry`

se během prvního spuštění spustí nástroj *raspi-config* (obr. 3.9), který zpřístupňuje základní konfiguraci systému jako je:

- Expandování souborového systému – to je nezbytné pro další používání Raspbianu
- Změnu uživatelského hesla (v této práci je heslo změněno na „**fel**“)
- Povolení automatického spuštění grafického rozhraní během spouštění
- Povolení nebo zakázání kamery
- Přetaktování procesoru (výrobce nabízí možnost přetaktování až na 1 GHz)

Po dokončení konfigurace systém doporučuje restart zařízení. Nástroj *raspi-config* lze opětovně spustit příkazem:

```
>> sudo raspi-config
```



Obr. 3.9: Program raspi-config

3.5.2 Nastavení bezdrátové komunikace

Pokud existuje požadavek, aby Raspberry Pi bylo připojeno k Internetu bezdrátově, tak nejjednodušší cesta je připojit jeden z podporovaných [15] USB WiFi adapérů, pak odpadne starost s instalací ovladačů bez připojení k repozitáři a navázání bezdrátového spojení bude jen otázkou konfigurace síťového adaptéru. Ke konfiguraci je nutné znát parametry spojení (SSID, heslo, typ zabezpečení), poté se konfigurace bezdrátové sítě na RPi vytvoří následovně:

1. Definice a nastavení síťových rozhraní se nachází v souboru */etc/network/interfaces*, soubor se otevře příkazem níže a k tomu je nutné mít oprávnění správce:

   ```
   >> sudo nano /etc/network/interfaces
   ```

2. V souboru je po čerstvé instalaci systému definováno pouze rozhraní *eth0* pro Ethernet a *lo* rozhraní localhosta, definici bezdrátového rozhraní wlan0 se přidá následujícími řádky pro zabezpečení sítě typu WPA/WPA2:

```
auto wlan0
iface wlan0 inet dhcp
wpa-ssid <name of your WiFi network>
wpa-psk <password of your WiFi network>
```

V případě zabezpečení WEP bude výsledný soubor vypadat takto:

```
auto wlan0

iface wlan0 inet dhcp
wireless-essid <name of your WiFi network>
wireless-key <password of your WiFi network>
```

3. Změněný textový soubor se zavře kombinací kláves Ctrl + X, je nutné potvrdit provedené změny. Po restartování Raspberry Pi příkazem `>> sudo reboot` se RPi automaticky připojí k bezdrátové síti tak, jak bylo definováno (příp. je možné restartovat pouze službu obsluhující nastavení připojení příkazem >> `sudo /etc/init.d/networking restart`).

3.5.3 Zpřístupnění rozhraní UART

Rozhraní UART je defaultně nastaveno k ladění systému, pokud je chceme využít k jinému účelu, musíme změnit jejich konfiguraci ve dvou souborech, a to

/boot/cmdline.txt
/etc/inittab

První soubor */boot/cmdline.txt* otevřeme příkazem

```
>> sudo nano /boot/cmdline.txt
```

a vymažeme následující parametry

```
console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
```

Soubor následně zavřeme a uložíme kombinací kláves *Ctrl + X* a potvrzením změny.

Ve druhém souboru zakomentujeme řádek obsahující

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

vložením znaku # na začátek řádku. Soubor opět zavřeme a uložíme kombinací *Ctrl + X*. Po restartování systému bude rozhraní UART přístupné uživatelem.

3.6 Raspberry Pi v systému sběru dat

Jak bylo zmíněno v rozboru, Raspberry Pi zastává v systému sběru dat unikátní pozici prostředníka mezi vnějším světem a senzory. Ke komunikaci s vnějším světem je vybaveno mnoha periferiemi, např.:

- Rozhraním Ethernet, přes které je možno provádět vzdálený monitoring a řízení systému prostřednictvím sítě Internet z libovolného zařízení.

- HDMI rozhraním umožňujícím připojení monitoru a zobrazení právě měřených dat.
- USB rozhraním, které zaručuje možnost lokálního řízení systému při zapojení vstupních zařízení jako je klávesnice nebo myš.

Mimo toho RPi disponuje dalšími rozhraními, na která je možné připojit senzory a měřit data významná pro domovní automatizaci v okolí centrální jednotky (senzory by měly být umístěné do vzdálenosti několika metrů). Mezi tato rozhraní patří:

- I^2C, SPI, UART rozhraní, na která je možné připojit senzor, jež komunikuje digitálně (tato kategorie zahrnuje mnoho inteligentních senzorů a satelitní jednotky v této práci navrhnuté).
- CSI rozhraním, které usnadňuje využití kamery v systému a zjišťovat tak přítomnost osob v místnosti, případně sledovat pohyb každé osoby v rámci celé budovy.

Do vybavení Raspberry Pi patří pouze jedno UART rozhraní, které je použito k připojení centrální jednotky (a satelitních) ke sběrnici RS485 (za pomoci budiče RS485), která zaručuje spolehlivou komunikaci dvou uzlů až na vzdálenost několika stovek metrů při použití vhodného protokolu (zde je použit léty ověřený protokol Modbus).

Z hlediska software poběží na Raspberry Pi kromě operačního systému dva programy vytvořené v této práci – první program je spuštěn na pozadí operačního systému (tento typ programů se v Unixovém prostředí nazývá daemon) a řídí komunikaci se senzory přes rozhraní UART a sběrnici RS485. Řízení komunikace daemonem znamená, že

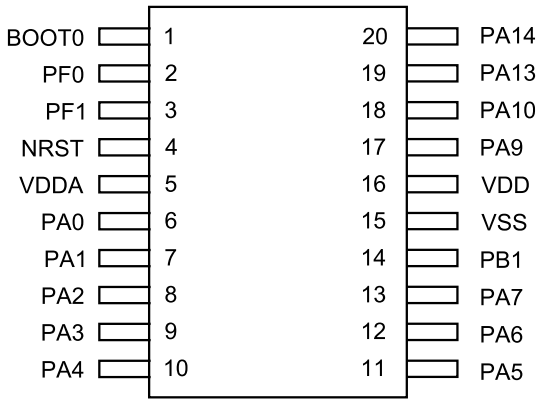
- zasílá požadavky satelitní jednotce na měření dat,
- získává data naměřená senzory připojenými k satelitním jednotkám,
- po získání dat je daemon uloží do paměti centrální jednotky ve formátu CSV.

Data uložená ve formátu CSV jsou zprostředkována PHP skripty, které jsou spouštěny webovým serverem – PHP skripty jsou druhým programem pro RPi vytvořeným v této práci a mají na starosti vizualizaci dat skrz webový prohlížeč. Díky kombinaci webového serveru a připojení k Internetu je možné vzdáleně monitorovat a řídit činnost systému sběru dat z jakéhokoli místa nejenom ve sledované budově, ale i mimo ni.

4 Mikrokontrolér STM32F0

V druhé kapitole byl zvolen za základ satelitních jednotek pro své různorodé periferie 32-bitový mikrokontrolér STM32F0. V této kapitole je představen spolu s jeho základními rysy a je zde popsána konfigurace periferií, které jsou použity k získávání dat ze senzorů. Závěr kapitoly se věnuje postavení mikrokontroléru v systému sběru dat.

Mikrokontroléry STM32F0 od společnosti STMicroelectronics jsou procesory zaměřené na sektor aplikací s požadavky na nízkou pořizovací cenu a nízkou spotřebu energie. Toho výrobce dosáhl nižší taktovací frekvencí CPU, redukovaným počtem periferií a GPIO pinů při zachování dostatečného výpočetního výkonu a bez omezení prostoru programátora (např. zůstala implementována komplexní obsluha přerušení NVIC). Programátor tak může získat třiceti dvou bitové MCU s jádrem ARM Cortex-M0 na frekvenci 48 MHz za cenu 21,55 Kč za jeden kus[2]. V této práci jsou použity mikrokontroléry STM32F050F6P6 uložené v dvaceti pinovém pouzdře TSSOP20, viz obr. 4.1.



Obr. 4.1: Rozmístění pinů mikrokontroléru STM32F050 v pouzdře TSSOP20

V této práci jsou použity mikrokontroléry usazené do modulu navrhnutého panem Čižmárem [16]. Na modulu jsou rozvedeny veškeré piny mikroprocesoru do standardního rastru 2,54 mm, což usnadňuje testování zapojení platformy na bezkontaktním poli a následnou realizaci zapojení na univerzálním plošném spoji. Modul také obsahuje blokovací kondenzátory napájení, červenou LED indikující funkční napájení a programovatelnou červenou LED připojenou na pin procesoru.

[2] v případě množstevní slevy při nákupu nad čtyři tisíce kusů, cena při koupi deseti kusů je 56,8 Kč

4.1 Parametry a periferie STM32F050F6P6

V tabulce níže jsou zachyceny důležité parametry použitého mikrokontroléru, informace jsou čerpány z datasheet výrobce [17].

Jádro	ARM 32-bit Cortex™ M0 CPU, 48 MHz
Paměť	32 kB Flash, 4 kB SRAM (HW kontrola parity)
Časovače	1x Advanced (1x 16-bit), 5x General purpose (4x 16-bit, 1x 32-bit)
Pouzdro	TSSOP20
Napájecí napětí	2,0 až 3,6 V
Kom. rozhraní	1x SPI, 1x I^2C, 1x USART
Program. rozhraní	SWD
Další periferie a I/O	CRC jednotka, 32 kHz oscilátor pro RTC, 1x 12-bit ADC (3x int. + 9x ext. kanálů), 15x GPIO, 2 kanály DMA
Cena za kus	21,55 Kč při nákupu 4000+ kusů
Rozměry modulu	30 x 20 mm (rozměry pouzdra TSSOP20 jsou 6,5 x 6,4 mm)

Tabulka 4.1: Parametry a periferie mikrokontroléru STM32F050

4.1.1 GPIO piny mikrokontroléru

Mikrokontrolér disponuje patnácti programovatelnými GPIO piny, které slouží k nízkoúrovňové komunikaci s hardware. Každý GPIO pin může být nastaven jako výstupní (push-pull nebo open-drain), jako vstupní (s nebo bez pull-up nebo pull-down odporem), do alternativní funkce nebo do analogového módu, a to programově pomocí:

- čtyř konfiguračních registrů GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR a GPIOx_PUPDR
- dvou datových registrů GPIOx_IDR a GPIOx_ODR
- jednoho set/reset registru GPIOx_BSRR
- jednoho uzamykacího registru GPIOx_LCKR
- dvou registrů pro výběr alternativní funkce GPIOx_AFRH a GPIOx_xAFRL

V případě nastavení pinu do alternativní funkce shrnuje výčet funkcí datasheet mikrokontroléru, v tabulce 4.2 níže jsou uvedeny funkce GPIO pinů a některé jejich vlastnosti (např. napěťová tolerance).

Pin	**Alternativní funkce, poznámka**
PA0-PA7	ADC_INx, USART, TIMx, SPI, I^2C , na pin PA5 je připojena uživatelská LED
PA9, PA10	USART, TIMx, I^2C, 5V tolerantní
PA13, PA14	SWD, USART
PB1	ADC_INx, TIMx
PF0, PF1	OSC, 5 V tolerantní

Tabulka 4.2: Výběr mapování alternativních funkcí na GPIO piny

ADC_INx značí pin přímo napojený na AD převodník, TIMx značí vstupy nebo výstupy časovačů a jejich různých kanálů, rozhraní SWD je programovací interface (při nastavení pinu PA14 do alternativní funkce USART1_TX nebude již možné dále ladit mikroprocesor pomocí SWD – pro další programování SWD rozhraním je nutné nastavit bootování z System Memory). OSC slouží pro připojení externího zdroje hodinového signálu a na 5 V tolerantní pin je možné připojit napětí do velikosti 5 V bez rizika poškození čipu, ostatní piny jsou 3,3 V tolerantní.

Ukázka konfigurace GPIO pinu PA3 napojeného na kanál AD převodníku do analogového režimu:

```
1 void GPIOADCInit(void) {
2   GPIOA->MODER |= (GPIO_Mode_AN << 6);//analog input
3   GPIOA->OSPEEDR |= (GPIO_Speed_Level_3 << 6);
4 }
```

Demonstrační funkce je součástí souboru `dp_pins.c`. Po deklaraci funkce je na druhém řádku nastaven třetí pin do analogového režimu (v registru `GPIOx->MODER` jsou vyhrazeny dva bity pro režim pinu, proto je použita bitová rotace o šest bitů doleva), na třetím řádku je nastavena rychlost pinu (opět jsou použity dva bity pro jeden pin). Třetí řádek není potřeba vkládat do kódu, při měření napětí AD převodníkem je v této úloze postačující nejnižší, výchozí frekvence pinu. Nastavení vyšší hodinové frekvence pinu zvyšuje spotřebu zařízení.

4.1.2 DMA

Mezi další periferie mikrokontroléru se řadí pěti kanálový řadič DMA (**D**irect **M**emory **A**ccess) realizující přenosy dat z paměti do paměti, z paměti do periferie a z periferie do paměti bez účasti CPU, tím jsou sníženy požadavky na výpočetní prostor. Přímý přístup do paměti lze například použít při měření napětí AD převodníkem na více kanálech. AD převodník na mikrokontrolérech STM používá k zápisu dat jeden datový registr – převodník po dokončení digitalizace napětí vyvolá přerušení, pokud procesor data neodebere, jsou přepsána při dalším měření. Zde přichází na řadu DMA řadič, který na přerušení zareaguje a data z datového registru AD převodníku vyzvedne a uloží je do paměti. Procesor tak nemusí čekat na dokončení konverze a může mezitím vykonávat jiné části programu a data si vyzvednout později bez obav, že budou přepsána.

Použití DMA periferie tak spočívá v nastavení, na které přerušení má reagovat, které data má odebrat a kam je má zapsat.

Ukázka konfigurace DMA pro odebírání dat z AD převodníku:

```
1 void DMA_ADCInit(uint32_t addressP, uint32_t addressM, uint32_t count) {
2  RCC->AHBENR |= RCC_AHBENR_DMA1EN;   //enable clocks
3  DMA_DeInit(DMA1_Channel1);          //deinit DMA
4  DMA1_Channel1->CCR &= ~DMA_CCR_EN;  //disable DMA
5  DMA1_Channel1->CPAR = addressP;     //the periphery address
6  DMA1_Channel1->CMAR = addressM;     //the memory address
7  DMA1_Channel1->CNDTR = count;       //the count of measurements
8  DMA1_Channel1->CCR |= DMA_CCR_PL | DMA_CCR_MSIZE_0 | DMA_CCR_PSIZE_0 |
```

```
      DMA_CCR_MINC | DMA_CCR_CIRC; //memory increment, memory circular mode
9  DMA1_Channel1->CCR |= DMA_CCR_EN;   //enable the DMA
10 }
```

Funkce výše je použita v souboru `dp_dma.c` v programu mikrokontroléru. Na prvním řádku je deklarace funkce, níže definice. Po deklarování funkce následuje povolení hodin DMA periferie, deinicializace prvního kanálu DMA a zakázání prvního kanálu DMA (nastavení DMA je nastavováno za běhu). Na řádcích 5, 6, 7 se postupně nastaví adresa paměti periferie, z které se mají data odebírat (směr přenosu je určen bitem DIR v `DMA_CCRx` registru), nastavení adresy v paměti, kam se data mají ukládat, a nastavení počtu dat k přenosu řadičem. Po přenesení počtu dat definovaných zde je vyvoláno přerušení DMA řadičem. Nastavení registru `DMA_CCRx` je složitější, zde musí být definována velikost přenášených dat jak na straně periferie, tak na straně paměti, v případě sekvenčního zápisu do paměti je nastaveno inkrementování ukazatele po každém zápisu (bit `DMA_CCR_MINC`). Vhodný je zápis do paměti na způsob kruhového bufferu (po dokončení měření se data opět začnou zapisovat na adresu paměti definovanou při inicializaci DMA, bit `DMA_CCR_CIRC`). Poté už stačí DMA kanál povolit (a provést nastavení DMA na AD převodníku, který má být s DMA propojen).

4.1.3 AD převodník

Obvod mikrokontroléru STM32F050F6P6 má zabudovaný dvanácti bitový aproximační AD převodník o dvanácti multiplexovaných kanálech umožňující měřit napětí z devíti externích a tří interních zdrojů. Externí kanály převodníku jsou vyvedeny na piny modulu, interní jsou připojeny k vnitřním snímačům (teplotní senzor, referenční napětí a V_{BAT} napětí). Konverze hodnot na jednotlivých vstupech AD převodníku může být provedena jednotlivě (single mode) nebo dávkově (scan mode). V dávkovém módu je automaticky změřeno napětí na vybrané skupině kanálů a ve spolupráci s DMA řadičem jsou hodnoty uloženy do paměti MCU.

Vlastnosti AD převodníku:

- Volitelné rozlišení: 12-bit, 10-bit, 8-bit, 6-bit (při nižším rozlišení je čas konverze kratší)
- ADC čas převodu při plném rozlišení: 1,0 µs
- Možnost samokalibrace
- Podpora DMA
- Devět externích analogových vstupů a tři interní
- Volitelné módy převodu: konverze jednoho nebo více kanálů, nepřetržitá nebo jednorázová konverze, konverze spustitelná programově, prioritní konverze vybraných kanálů
- Vyvolání přerušení po dokončení převodu

Ukázka nastavení AD převodníku do scan mode s odebíráním dat DMA řadičem:

```
1 uint16_t ADC_and_DMAInit(uint16_t count, uint16_t channels) {
    //ADC
2   RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;       //enable the clocks
3   ADC_DeInit(ADC1);                         //deinit
```

```
    //DMA
4   DMA_ADCInit((uint32_t) &(ADC1->DR), (uint32_t) ADCCircleBuffer, count);
    //ADC
5   ADC1->CR |= ADC_CR_ADCAL;                  //ADC selfcalibration
6   while((ADC1->CR & ADC_CR_ADCAL) != 0);     //wait until the calbr. is done
    //ADC + DMA
7   ADC1->CFGR1 |= ADC_CFGR1_DMACFG;           //DMA circular mode
8   ADC1->CFGR1 |= ADC_CFGR1_DMAEN;            //enable DMA
    //ADC
9   ADC1->SMPR |= ADC_SMPR1_SMPR;              //sampling time
10  ADC1->CHSELR |= channels;                  //ADC channels
11  ADC1->CR |= ADC_CR_ADEN;                   //enable the ADC
12  while((ADC1->CR & ADC_ISR_ADRDY) == 0);    //wait until the ADC is ready
13  return channels;
14 }
```

Kód opět začíná povolením hodin periferie, deinicializací ADC a zavoláním funkce `void DMA_ADCInit` popsané v podkapitole 4.1.2. Po dokončení inicializace DMA je nastaven bit `ADC_CR_ADCAL` v registru `ADC_CR`, ten vyvolá spuštění samokalibrace převodníku, následně program ve smyčce `while` čeká na dokončení kalibrace. Poté se přistoupí k nastavení spolupráce AD převodníku a DMA řadiče dosazením bitů `ADC_CFGR1_DMACFG` a `ADC_CFGR1_DMAEN` do registru `ADC_CFGR1`, první bit oznamuje, že data jsou zapisována do kruhového bufferu, druhý bit povolí odběr dat DMA řadičem. Na devátém řádku se nastaví čas měření hodnoty na kanálech převodníku, čím je nastaven delší interval, tím spíš se na vstupu kanálů AD převodníku nabije vzorkovací kondenzátor a měření bude přesnější. Po nastavení kanálů, které mají být kvantovány (registr `ADC_CHSELR`), zbývá už jen povolit AD převodník a vyčkat, až bude připraven (jedenáctý a dvanáctý řádek).

AD převodník se používá pro připojení senzorů s napěťovým výstupem.

4.1.4 Čítače

Mikrokontrolér obsahuje šest čítačů, z toho čítač TIM1 patří do skupiny Advanced Control Timers, zbylých pět (TIM2, TIM3, TIM14, TIM16, TIM17) se řadí mezi General Purpose.

Typ	Čítač	Rozlišení	Směr čítání	DMA	Počet kanálů	Komplementární výstup
Advanced Control	TIM1	16-bit	Nahoru, dolů, nahoru/dolů	Ano	4	Ano
General Purpose	TIM2	32-bit	Nahoru, dolů, nahoru/dolů	Ano	4	Ne
	TIM3	16-bit	Nahoru, dolů, nahoru/dolů	Ano	4	Ne
	TIM14	16-bit	Nahoru	Ne	1	Ne
	TIM16, TIM17	16-bit	Nahoru	Ano	1	Ano

Tabulka 4.3: Parametry čítačů mikrokontroléru STM32F0F0

Každý General Purpose čítač může generovat PWM na svém výstupu, pracovat jako časovač (tedy čítat synchronně přicházející impulsy a tím odměřovat úseky času), měřit parametry impulsů, čítat signály z externího zdroje, určit střídu signálu atd.

Čítač TIM1 s pokročilým řízením (Advanced Control) nabízí kromě toho, že se nachází na sběrnici APB2 (na rozdíl od GP čítačů, které jsou na sběrnici APB1, což je nutno vzít v potaz při zavádění hodinového signálu čítače při inicializaci), více možností řízení. Např. komplementární PWM výstup či zpracování kvadraturního signálu (zpracovat kvadraturní signál také zvládne TIM2 a TIM3). Většina čítačů mikrokontroléru má nezávislé kanály pro čtyři základní režimy:

- Input capture – zaznamenání vstupní události (např. náběžné hrany signálu), výstupem je vygenerované přerušení v okamžiku zaznamenání
- Output compare – změna výstupního signálu po uplynutí definovaného úseku času, což dává možnost generovat signál s volitelnou střídou, frekvencí a polaritou
- PWM generation – generování výstupního signálu pulsně šířkovou modulací
- One-pulse mode output – časovač vygeneruje puls po uplynutí definovaného úseku času v odezvě na vstupní událost (např. na náběžnou hranu signálu)

Ukázka nastavení čítače TIM3 do režimu časovače s povolením přerušení při přetečení:

```
1 void TIM3_Init(void) {
2   RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;       //enable TIM3 clocks
3   TIM3->PSC |= TIM3_PRESCALER;              //prescaler
4   TIM3->ARR &= 0;                           //reset and set
5   TIM3->ARR |= TIM3_PERIOD;                 //auto-reload value
6   TIM3->DIER |= TIM_DIER_UIE;               //enable interrupt
7   TIM3->CR1 |= TIM_CR1_CEN;                 //enable timmer
8   NVIC_EnableIRQ(TIM3_IRQn);                //enable interrupt
9 }
```

Nejprve je nutné povolit hodinový signál periferie čítače TIM3, to se provede na druhém řádku. Následně se nastaví dělička časovače (to je hodnota, kterou se dělí počet vstupních pulsů, což zvyšuje rozsah časovače). Poté se nastaví hodnota auto-reload registru. Pokud počet vstupních pulsů dosáhne této hodnoty, časovač přeteče a vyvolá přerušení (vyvolání přerušení je povoleno na řádku šest). Pak už jen zbývá povolit časovač a nastavit přerušení v NVIC.

Výpočet frekvence přerušení časovačem:

K výpočtu periody jednotlivých přerušení vyvolaných časovačem je zapotřebí nejprve znát frekvenci hodin za děličkou PSC, to se vypočítá dle vzorce

$$f_{CK\_CNT} = \frac{f_{CL\_INT}}{\mathrm{PSC}+1},$$

poté se frekvence přerušení vypočítá dle vztahu

$$f_{UI} = \frac{f_{CK\_CNT}}{ARR},$$

kde

f_{CK_INT} je interní frekvence hodin, zde je rovna 48 MHz,

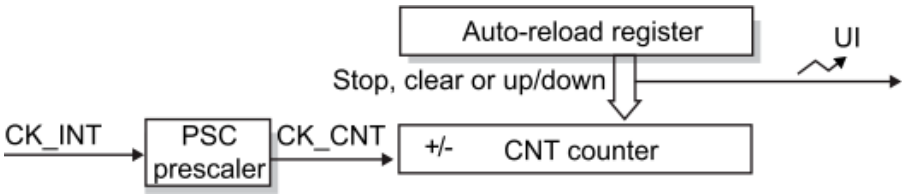
PSC je obsah registru TIM3->PSC (hodnota registru děličky pulsů),

f_{CK_CNT} je frekvence pulsů za dělícím blokem,

ARR je obsah registru TIM3->ARR,

f_{UI} je frekvence přerušení.

V kódu výše je hodnota proměnná TIM3_PRESCALER rovna 1000 a proměnná TIM3_PERIOD rovna 4800, výsledná frekvence přerušení UI je tak 48 MHz/(1k * 4,8k) = 10 Hz. Schéma 4.2 níže zobrazuje část obvodu časovače TIM2 řešící čítání synchronních pulsů CK_INT. PSC prescaler je dělička, v auto-reload registru je hodnota, se kterou se porovnává obsah CNT čítače. Pokud se registr v CNT čítači shoduje s hodnotou uloženou v registru ARR, je vyvoláno přerušení UI, čítač je resetován a celý cyklus začne znova.



Obr. 4.2: Část zapojení general-purpose časovače TIM2 a TIM3, [18]

V práci jsou časovače využity k měření časových úseků a ke generování PWM signálů, které modulují světlo emitované diodou u senzoru optické brány.

4.1.5 USART

Mezi komunikační rozhraní mikrokontroléru patří jedno plně duplexní sériové rozhraní USART (pojmenované USART1) a dokáže komunikovat rychlostí až 6 Mbitps. Obvody STM32F050 podporují řízení toku dat signály CTS, RTS a RS485 DE (poslední jmenovaný signál je podpora RS485 tranceiveru), multiprocesorovou komunikaci, single-wire half-duplex komunikaci, NRZ kódování přenosu. Data mezi periferií USART a vnitřní pamětí mohou být předávána DMA řadičem bez účasti procesoru. Toto rozhraní je také možno použít k nahrání programu do procesoru namísto SWD rozhraní (tzv. bootloader). Mezi další vlastnosti rozhraní patří automatické detekce přenosové rychlosti, volitelný počet datových bitů v jednom rámci (8 nebo 9 bitů) a počet stop bitů (1 bit, 1,5 bitu nebo 2 bity), možnost přenášení dat v pořadí MSB a LSB, kontrola parity, možnost přechodu rozhraní do mute režimu a následné probuzení při rozpoznání své adresy nebo při detekci klidového stavu sběrnice, a podpora komunikace protokolem Modbus.

Ukázka nastavení periferie USART s DMA přenosem dat:

```
1 void USARTInit(void) {
2   GPIOUSARTInit();
3   RCC->APB2ENR |= RCC_APB2ENR_USART1EN;      //enable clocks
4   USART1->BRR |= 0x1A1;                      //115200 baudrate
5   USART1->CR1 |= USART_CR1_MME | USART_CR1_WAKE | USART_CR1_RXNEIE |
     USART_CR1_RTOIE;                          // USART interrupts
```

```
6   USART1->CR2 |= (USART_ADDRESS << 24) | USART_CR2_ADDM7;
7   USART1->RTOR |= 0x23;                       //timeout after last stop bit
8   USART1->CR3 |= USART_CR3_DMAR | USART_CR3_DMAT | USART_CR3_DEM;
9   USART1->CR1 |= USART_CR1_TE | USART_CR1_RE | USART_CR1_UE;
10  USART1->RQR |= USART_RQR_MMRQ;              //mute mode request
11  NVIC_EnableIRQ(USART1_IRQn);                //enable interrupts
12  DMA_USART_TXInit((uint32_t)&(USART1->TDR), (uint32_t)(USARTBuffer));
13  DMA_USART_RXInit((uint32_t)&(USART1->RDR), (uint32_t)(USARTBuffer),
USART_BUFFER_SIZE);
14 }
```

Nastavení rozhraní USART je složitější, protože jsou v úloze využity možnosti periferie jako je hardware podpora sběrnice RS485 a protokolu Modbus.

Kód výše začíná zavoláním funkce, která inicializuje GPIO piny (nastaví piny PA1, PA9 a PA10 do první alternativní konfigurace `USART1_RTS`, `USART1_TX` a `USART1_RX`). Poté kód pokračuje zavedením hodinového signálu na periferii rozhraní a nastavením modulační rychlosti zápisem do registru `USART1->BRR`. Poté se v registru `USART1->CR1` povolí Mute režim (bit `USART_CR1_MME`) vhodný ke komunikaci protokolem Modbus spolu s probuzením po detekování své adresy (`bit USART_CR1_WAKE`). Povolením přerušení v případě příjmu dat nebo chyby vzniklé během příjmu dat se nastaví bitem `USART_CR1_RXNEIE` a povolením přerušení vyvolaném dokončeným přenosem dat bitem `USART_CR1_RTOIE` (tento bit patří do harware podpory protokolu Modbus v RTU režimu ze strany mikrokontroléru, viz podkapitola 6.3.1). Na sedmém řádku se do registru `USART1->CR2` zapíše adresa zařízení a zvolí se používání sedmibitové adresace. V registru `USART1->RTOR` je uložen počet tiků hodin ve smyslu baudrate, po jehož uplynutí po detekování stop bitu je vysílání označeno za ukončené.

V třetím kontrolním registru `USART1->CR3` (desátý řádek) je povoleno DMA pro příjem a odesílání dat spolu s bitem `USART_CR3_DEM`, který značí, že je zapnuta hardwarové řízení směru komunikace protokolem RS485 (signál je vyveden na pin PA1, který je v alternativní funkci `USART1_RTS` – **R**equest **t**o **S**end). Na následujícím řádku je povoleno rozhraní USART spolu s povolením příjmu a odesíláním dat. Poté zbývá uvést rozhraní do mute režimu, povolit v NVIC řadiči přerušení periferie USART1 (řádek jedenáct) a nastavit řadič DMA pro výměnu dat.

Výpočet hodnoty registru `USART1->BRR` dle zadané hodnoty baudrate:

Dle hodnoty USARTDIV uložené v registru `USART1->BRR` je stanovena modulační rychlost rozhraní. Po zvolení požadované baudrate se hodnota registru spočte dle vztahu

$$\mathrm{USARTDIV} = \frac{f_{CLK}}{baudrate},$$

kde f_{CLK} je frekvence jednoho ze čtyř možných zdrojů hodinové frekvence. Ve výchozím nastavení je brána frekvence PCLK, jejíž hodnota je zde rovna 48 MHz. Po dosazení baudrate (115200 Bd) a hodinové frekvence je hodnota USARTDIV rovna 0x1A1 (desetinná část podílu je zaokrouhlena nahoru). Podrobněji je vysvětlen výpočet dle požadované baudrate v referenčním manuálu [18] v kapitole 26.5.4.

4.2 Mikrokontrolér v systému sběru dat

Satelitní jednotky jsou tvořeny mikrokontrolérem ST32F050 a senzory připojenými přes různá rozhraní. Mikrokontroléry mají za úkol data koncentrovat a zasílat je na vyžádání centrální jednotce, na kterou nemůžou být veškeré senzory připojeny přímo. To není možné ze dvou důvodů.

Zaprvé centrální jednotka nedisponuje požadovaným rozhraním (např. Raspberry Pi nemá AD převodník a některé použité senzory mají analogový výstup). Zadruhé má RPi omezený počet rozhraní a nebylo by tedy možné připojit vyšší počet čidel.

Problém získávání dat z větších vzdáleností je vyřešen vytvořením sběrnice RS485 a zajištěním komunikace robustním protokolem Modbus. Toto řešení zaručuje snadné rozšíření platformy v prostoru (je možné připojit novou satelitní jednotku kamkoli) i ve funkčnosti (k mikrokontroléru může být připojen díky jeho univerzálnosti jakýkoli senzor – samozřejmě to vyžaduje i menší úpravu programu MCU).

5 Měřená data a použité senzory

Mezi důležitá data v domovní automatizaci se řadí různé fyzikální veličiny jako je teplota a vlhkost vzduchu, intenzita světla, kvalita vzduchu, koncentrace oxidu uhličitého a jiná data (např. přítomnost a pohyb osob nebo stav řídicích prvků systému jako je pootočení ventilu nebo klapky).

Všechna tato data jsou získávána systémem sběru dat a jsou ukládána do paměti centrální jednotky pro momentální i pozdější použití. Všechna tato data, viz níže, ovlivňují mikroklima interiéru místnosti, které každý člověk vnímá různou měrou a které mají vliv na to, jak se během pobytu v místnosti cítí. V textu níže jsou uvedeny doporučené hodnoty těchto činitelů a možná rizika, která hrozí při nedodržení doporučených rozsahů.

5.1 Teplota a relativní vlhkost vzduchu

Jedním z cílů procesu klimatizování místnosti je upravit teplotu a vlhkost vzduchu v obytných prostorech tak, aby bylo dosáhnuto tzv. tepelné pohody člověka. Tepelnou pohodu, kromě objektivních vlastností mikroklimatu interiéru (což je vnitřní teplota vzduchu, relativní vlhkost, proudění vzduchu, teplota okolních předmětů), ovlivňují také subjektivní faktory (např. druh práce, kterou osoba v prostoru vykonává). Doporučená teplota vzduchu v obytných prostorech je stanovena na 20±2 °C v chladné části roku a na 24±2 °C v teplé části roku [19]. Rozdíl teplot v místě hlavy a chodidel by neměl být větší než 2 °C (v úrovni hlavy může být nižší teplota). Nabízí se tak mít v každé místnosti zapojená dvě čidla teploty, kde první se bude nacházet u podlahy a druhé ve výši hlavy.

Vlhkost vzduchu se běžně vyjadřuje relativní vlhkostí, ta udává poměr mezi aktuálním množstvím vodních par ve vzduchu a množstvím vodních par, které by plně nasycený vzduch obsahoval při téže teplotě [20], proto se relativní vlhkost udává v procentech. Relativní vlhkost vzduchu v letních měsících neměla překročit 65% a v zimní by neměla klesnout pod 30%. Optimální relativní vlhkost se pohybuje kolem 45%.

Přímé zdravotní následky hrozí při nízké relativní vlhkosti vzduchu, např. vysoušení sliznic. Vysoká relativní vlhkost přináší zdravotní rizika nepřímo – v tomto případě dochází ke kondenzaci vodních par na vnitřních stěnách obytných prostor a hrozí výskyt plísní.

5.1.1 Zapojení senzoru vlhkosti a teploty SHT11

Teplotu a relativní vlhkost vzduchu v této práci měří čip SHT11 od společnosti Sensirion. K měření vlhkosti využívá čidlo kapacitního typu a k měření teploty „bandgap“, což je označení pro polovodičový monokrystalický senzor. Každý čip je výrobcem kalibrovaný. Celé zařízení je

uloženo v SMD pájitelném reflow pouzdře (rozměry pouzdra jsou 7,5 x 4,9 mm) a výrobce uvádí, že je vybaven digitálním výstupem I^2C/2 wire interface, nicméně specifikaci I^2C nesplňuje (senzor může být připojen k I^2C sběrnici a komunikovat dle příkazů popsaných v manuálu bez rušení dalších připojených zařízení, avšak ho není možné adresovat I^2C protokolem [21]). Na fotografii 5.1 je zachycen čip SHT11 v SMD pouzdře, v této práci je navíc opatřen adaptérem pro zapojení do standardního rastru 2,54 mm.

Obr. 5.1: Senzor vlhkosti a teploty SHT11

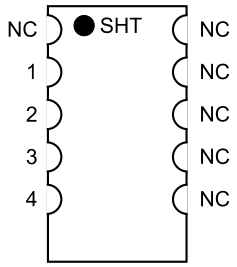
Parametr	Podmínka	Nejméně	Typicky	Nejvýše	Jednotka
napájecí napětí		2,4	3,3	5,5	V
napájecí proud	měření		0,55	1	mA
	jinak		0,3	1,5	µA
log. výstup 0	I_{OL} < 4 mA	0		250	mV
log. výstup 1	R_P < 25 kΩ	90%		100%	VDD
log. vstup 0		0%		20%	VDD
log. vstup 1		80%		100%	VDD

Tabulka 5.1: Elektrické charakteristicky senzoru SHT11

Parametr	Teplotní senzor	Vlhkostní senzor
Rozlišení	12 a 14 bit	8 a 12 bit
Přesnost	±0.4°C	±3% RH
Rozsah	-40°C do 123,8°C	0 – 100% RH
Odezva na změnu	5 – 30 s	8 s
Pokles přesnosti	< 0,04°C za rok	< 0,5% RH za rok
Cena	353 Kč	

Tabulka 5.2: Parametry senzoru SHT11

Senzor vyžaduje napájení mezi 2,4 V až 5,5 V. Dle velikosti napájecího napětí se mění koeficienty, které jsou použity při kvantování dat naměřených senzorem, viz kapitola 8.4. Schéma mikročipu a propojení s piny mikrokontroléru STM32F050 vypadá následovně.



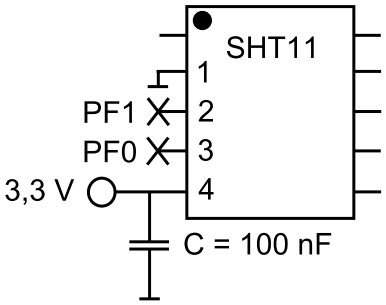
Obr. 5.2: Senzor SHT11, vyvedení pinů

Pin SHT11	Název	Popis	Pin STM32F0xx
1	GND	Země	GND
2	DATA	Sériová data, vstup/výstup	PF0 (interní pull-up)
3	SCK	Hodiny	PF1(interní pull-up)
4	VDD	Napájecí napětí	VDD (+ 3,3 V)
NC	NC	Piny, které musí zůstat nepřipojeny	-

Tabulka 5.3: Zapojení senzoru SHT11 a mikrokontroléru STM32F050

Piny mikrokontroléru PF0 a PF1 jsou nastaveny do výstupního režimu otevřeného kolektoru s pull-up odpory. Pull-up odpor na datové části sběrnice zaručuje, že v klidovém stavu bude v logické 1 a opačně, stažením datové linky na logickou 0 oznamuje senzor nebo mikrokontrolér začátek přenosu dat. Hodinový signál generuje mikrokontrolér, který je v roli mastera. Doporučení z datasheetu senzoru SHT11 zní, že žádný vodič by neměl být delší než 10 cm, jinak dojde k přeslechům a ztrátě dat. Integritu dat získaných ze senzoru je možné ověřit kontrolním součtem CRC, který vytváří senzor.

Doba měření hodnot se liší dle použité přesnosti, pro 14-bitové rozlišení zabere jedna konverze až 320 ms, při použití 12bitového rozlišení 80 ms.



Obr. 5.3: Zapojení senzoru SHT11

5.1.2 Přepočet naměřených dat na teplotu

Čidlo teploty je lineární z principu designu, a tak se teplota spočítá jednoduše dle vzorce

$$T = d_1 + d_2 SO_T,$$

kde T je výsledná teplota okolí, d_x jsou koeficienty ovlivněné napájecím napětím a rozlišením měření a SO_T jsou data získaná ze senzoru.

VDD	d_1 (°C)
5 V	-40,1
4 V	-39,8
3,5 V	-39,7
3 V	-39,6
2,5 V	-39,4

SO_T	d_2 (°C)
14 bit	0,01
12 bit	0,04

Tabulka 5.4:Koeficienty výpočtu teploty

5.1.3 Přepočet naměřených dat na vlhkost

Po získání dat ze senzoru je potřeba je přepočíst na stupně Celsia a relativní vlhkost. V manuálu [21] je uvedena následující rovnicí pro získání relativní vlhkosti ovzduší

$$RH_{linear} = c_1 + c_2 * SO_{RH} + c_3 * SO_{RH}^2 \ (\%RH),$$

kde koeficienty c_x jsou dány použitým rozlišením měření a kde SO_{RH} jsou naměřená data. Do rovnice je zahrnuto potlačení nelineárnosti vlhkostního čidla.

SO_{RH}	c_1	c_2	c_3
12 bit	-2,0468	0,0367	-1,5955E-6
8 bit	-2,0468	0,5872	-4,0845E-4

Tabulka 5.5: Koeficienty pro převod vlhkosti senzorem SHT11

Pokud se teplota výrazně liší od 25 °C, vlhkost vypočítání dle vzorce výše vyžaduje teplotní kompenzaci dle vzorce

$$RH_{true} = (T_{\circ C} - 25)(t_1 + t_2 * SO_{RH}) + RH_{linear},$$

kde koeficienty t_x jsou dány použitým rozlišením měření, $T_{\circ C}$ je naměřená teplota, SO_{HR} jsou naměřená data a RH_{linear} je vypočítaná vlhkost.

SO_{RH}	t_1	t_2
12 bit	0,01	0,00008
8 bit	0,01	0,00128

Tabulka 5.6: Koeficienty výpočtu vlhkosti s ohledem na teplotu

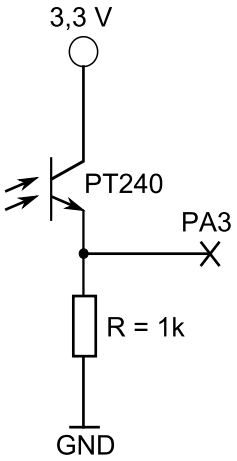
5.2 Intenzita osvětlení

Údaj o osvětlení se v domovní automatizaci využívá například pro zjištění, zda světlo dopadající do místnosti není oslňující, nebo – v kombinaci s detekcí osob – zda se zbytečně neosvětluje prázdná místnost. Systém by tak mohl při nadměrném osvětlení místnosti reagovat vysunutím markýzy, pootočením stínicí plochy žaluzií či spuštěním rolet, a tím upravit světelné podmínky.

5.2.1 Zapojení snímače osvětlení

Snímač osvětlení je realizován jednoduchým zapojením fototranzistoru PT240 se společným kolektorem a rezistoru o velikosti 1 kΩ. Rezistor je zvolen tak, aby se tranzistor nedostal do

saturace a měření dat probíhalo vždy v lineární části charakteristiky. S větším množstvím světla dopadajícího na bázi tranzistoru roste proud procházející tranzistorem, tím stoupá úbytek napětí na rezistoru – tento úbytek napětí je výstupem tohoto senzoru osvětlení. Schéma níže zobrazuje zapojení fototranzistoru se společným kolektorem.



Obr. 5.4: Zapojení snímače osvětlení

5.3 Kvalita vzduchu

Index kvality vzduchu dle Českého hydrometeorologického ústavu zohledňuje vliv emisí na zdravotní stav obyvatelstva a je založen na měření koncentrací oxidu siřičitého, oxidu dusičitého, suspendovaných částic a oxidu uhelnatého. V systémech domovní automatizace se většinou vyhodnocuje pouze podmnožina těchto faktorů, a to plyny přímo vzniklé v důsledku pobytu osob v místnosti, např. se detekuje množství oxidu uhličitého vznikajícího při dýchání člověka či látky obsažené v cigaretovém kouři (sloučeniny vodíku, oxidu uhelnatého atd.).

5.3.1 Zapojení senzoru kvality vzduchu

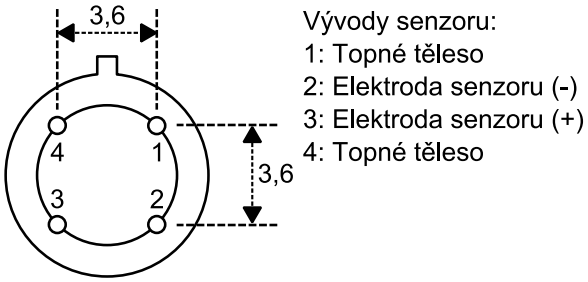
Základním prvkem obvodu pro měření kvality vzduchu je senzor TGS2600 od společnosti Figaro. Elektrochemický senzor reaguje na nízké koncentrace plynů v řádu jednotek až desítek ppm, které obvykle vznikají během pobytu lidí v místnosti. Díky vysoké citlivosti senzoru je možné spustit větrání místnosti před dosažením vnímatelně zhoršené kvality ovzduší. Čidlo reaguje především na přítomnost vodíku, etanolu a izobutanu, méně reaguje na výskyt oxidu uhelnatého a metanu (metan a vodík a některé další plyny vznikají v metabolismu člověka). V návodu [22] výrobce předvádí reakci senzoru v uzavřené místnosti na cigaretový kouř.

Vnitřek snímače je realizován snímací destičkou (odpor R_S ve schématu 5.7), která je zahřívána vnitřním topným tělesem (odpor R_H). Destička zvyšuje svoji vodivost s vyšším množstvím plynů v ovzduší.

Kvalita ovzduší se určuje poměrem odporu destičky R_S k referenčního odporu, který je změřen v čistém ovzduší. Odpor senzoru se měří nepřímo přes napěťový dělič. Při poměru

$$\frac{R_S}{R_{SREF}} = 0.85$$

a nižším je kvalita ovzduší v místnosti označena za sníženou. Referenční odpor R_{SREF} se stanoví jeho změřením v ovzduší, které je považováno za čisté.

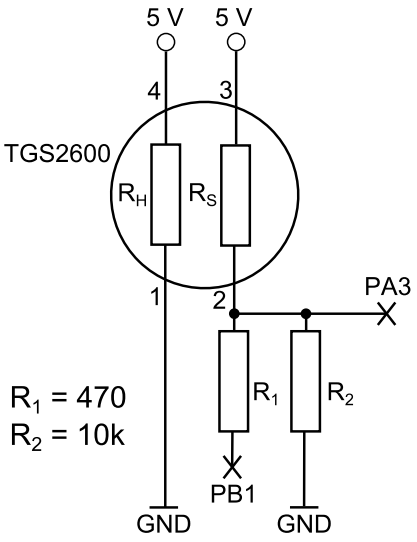


Obr. 5.5: Pohled zespoda, senzor TGS 2600

Senzor TGS2600 má analogový výstup a výrobce doporučuje napětí měřit nepřímo přes napěťový dělič s druhým odporem o velikosti 10 kΩ. Problém je, že vodivost snímací destičky senzoru se mění až o dva řády (z přibližně 20 µS na 2 mS), takže při zvolení velikosti druhého odporu 10 kΩ by se úbytek napětí pohyboval mezi 0,9 V až 4,77 V a na vstup AD převodníku je možné přivést napětí do velikosti 3,3 V.

Při volbě druhého odporu menšího, např. 500 Ω, by se na vstupu neobjevilo napětí vyšší než 3,3 V, ale zase byla výrazně omezena schopnost rozpoznat drobné změny v kvalitě vzduchu – úbytek napětí by se pohyboval v rozmezí 0,05 V a 2,5 V, kde hodnota napětí přesahující 0,06 V značí znečištěné ovzduší (v případě s rezistorem 10 kΩ bylo ovzduší považováno za znečištěné při hodnotě napětí vyšší než 1,5 V). Systém domovní automatizace by tak byl schopen rozlišovat znečistění ovzduší jen ve dvou stavech – obtěžující a neobtěžující.

Proto bylo navrhnuto řešení druhé s programově měnitelnou velikostí odporu:



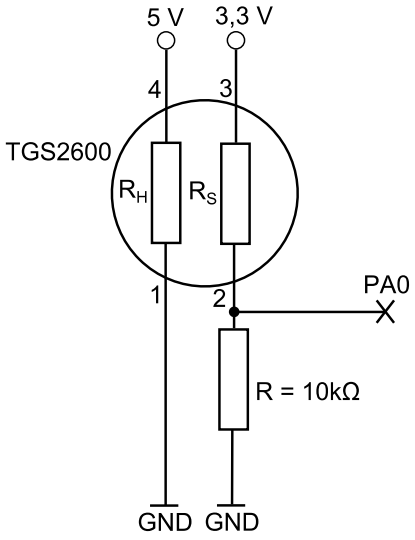
Obr. 5.6: Zapojení senzoru TGS2600 s měnitelnou velikostí odporu

V tomto zapojení probíhá měření kvality ovzduší ve dvou krocích, nejprve je pin PB1 nastaven do výstupního režimu s naprogramovaným výstupem do logické nuly. Při této konfiguraci jsou odpory R_1 a R_2 zapojeny paralelně a výsledný odpor je přibližně 450 Ω. Pokud program mikrokontroléru naměří napětí nižší než

$$U_{MAX} = \frac{5}{\frac{2,7}{3,3} * R_2 + R_1 || R_2} * R_1 || R_2 = 0,26\,V\,,$$

tak to znamená, že při odpojení odporu R_1 se na vstupu ADC převodníku neobjeví napětí vyšší než 3,3 V a je možné měřit kvalitu vzduchu s větší přesností.

Ovšem při použití tohoto algoritmu se změnou odporu, na kterém je měřen úbytek napětí, změnila i velikost odporu R_S až o několik kilo Ohmů pravděpodobně z důvodu poklesu proudu protékajícího snímací destičkou a jejím následným zchlazením (absorpce plynů destičkou je závislá na teplotě, proto je vyhřívána topným tělesem R_H k udržení stále teploty). Z tohoto důvodu bylo navrženo třetí, následné zapojení.



Obr. 5.7: Zapojení senzoru TGS2600

Odlišnost v tomto zapojení je ta, že napájecí napětí na snímací destičce je 3,3 V. Plynocitlivá destička nadále mění svůj odpor, jako v zapojení s 5 V a vzhledem k tomu, že kvalita ovzduší není měřena absolutně, ale poměrně k ustálené velikosti odporu v čistém ovzduší, postup měření je neměnný. Nejvyšší napětí, které se může na vstupu ADC převodníku objevit, je 3,3 V.

Plyn	Plyny znečišťující ovzduší
Rozsah	1 – 30 ppm vodíku
Napájecí napětí	5,0 ± 0,2 V
Odpor rezistoru R_H	83 Ω
Odpor rezistoru R_S	10k – 90k Ω
Reakce na změnu	přibližně 90 vteřin
Zahřívací doba	7 dní
Cena senzoru	509 Kč

Tabulka 5.7: Specifikace senzoru TGS 2600

5.4 Oxid uhličitý (CO_2)

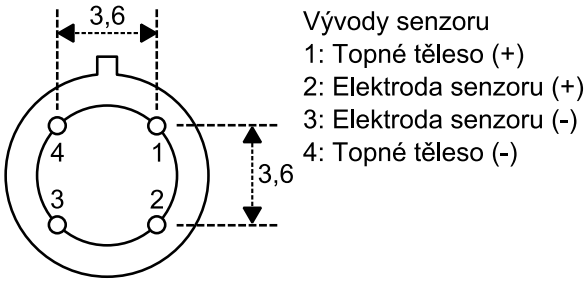
Oxid uhličitý je nejběžnějším nežádoucím plynem v ovzduší obytných budov. Zdrojem tohoto plynu je především člověk sám, popř. také rostliny, které během noci produkují CO_2. Koncentrace CO_2 ve vnějším prostředí se pohybuje okolo 380 ppm (parts per million), za přípustnou hodnotu CO_2 ve vnitřních prostorách se udává množství 1 500 ppm a za bezpečnou hranici koncentrace CO_2 se považuje množství do 5000 ppm. Nad toto číslo hrozí pro člověka zdravotní rizika, viz tabulka níže.

Množství CO_2 [ppm]	**Účinky na lidský organismus, poznámky**
330 – 370	vnější prostředí
450 – 1 000	vhodné množství, příjemný pocit
1 000 – 2 000	pocit ospalosti a horšího vzduchu
2 000 – 5 000	možné bolesti hlavy, nižší schopnost koncentrace, snížená pozornost
> 5 000	pocit těžkého vzduchu a nevolnosti, zvýšený tep
> 15 000	potíže s dýcháním
> 30 000	bolesti hlavy, závratě a nevolnost
> 45 000	letargie a ztráta vědomí
35 000 – 50 000	vydechovaný vzduch dospělého člověka

Tabulka 5.8: Příklady koncentrace CO_2, [23]

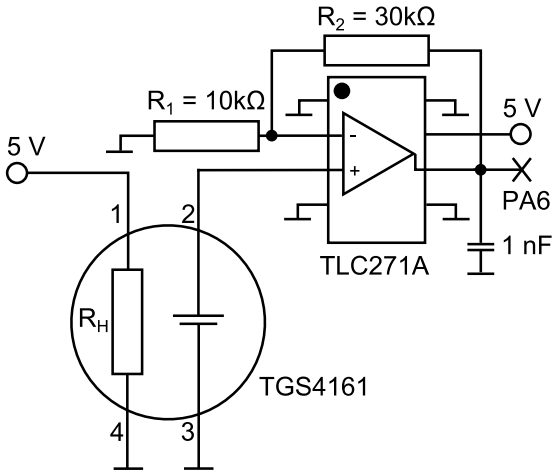
5.4.1 Zapojení senzoru CO_2

V této práci je použit senzor TGS4161 od společnosti Figaro, který je navržen pro měření koncentrace CO_2 v ovzduší v rozsahu 350 – 10 000 ppm. Měřicí rozsah senzoru pro aplikaci domovní automatizace je dostatečný. Plynocitlivá část senzoru je pevný elektrolyt umístěný mezi dvěma elektrodami, senzor je proto velice citlivý na přepólování napětí na vývodech a při špatném zapojení může být senzor nenávratně zničen. Koncentraci CO_2 je možné určit podle změny elektromotorického napětí (EMF) mezi dvěma elektrodami. Závislost mezi změnou napětí a koncentrací plynu je lineární na logaritmické ose [24]. Senzor je stabilní s ohledem na relativní vlhkost vzduchu (při koncentraci CO_2 kolem 1000 ppm a větší dochází k několika milivoltovému poklesu elektromotorického napětí).



Obr. 5.8: Pohled zespoda, senzor TGS 4161

V aplikaci je senzor zapojen dle doporučení datasheetu (viz schéma 5.9) s přidaným kondenzátorem o velikosti 1 nF za účelem stabilizace úbytků napětí, které mohou vznikat při odebírání náboje AD převodníkem.



Obr. 5.9: Zapojení senzoru TGS4161

Operační zesilovač je použit v zapojení neinvertujícího zesilovače. První důvod je, aby bylo napětí na senzoru měřeno obvodem s vysokou impedancí, což je doporučení z datasheetu senzoru. Druhý důvod je, že AD převodníky zanáší do měření chybu kvantování[3], a protože se napětí na výstupu pinu 2 se pohybuje kolem 400 mV a rozsah AD převodníku je 3,3 V, bylo zesíleno dle vztahu

$$U_{výstupní} = \left(1 + \frac{R_2}{R_1}\right) * U_{vstupní} .$$

Napětí je tedy zesíleno na čtyřnásobek své původní velikosti.

[3] Každá naměřená hodnota je kvantována k jedné úrovni, počet úrovní je dán rozlišením převodníku.

Snímací část	pevný elektrolyt
Plyn	oxid uhličitý
Rozsah	350 – 10 000 ppm
Napájecí napětí	5,0 ± 0,2 V
Odpor rezistoru R_H	70 ± 7 Ω
EMF	220 – 490 mV v 350 ppm CO_2
Reakce na změnu	přibližně 90 vteřin
Zahřívací doba	12 hodin
Cena senzoru	699 Kč

Tabulka 5.9: Parametry senzoru TGS 4161.

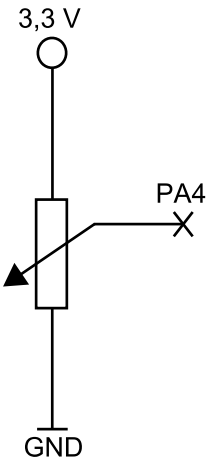
5.5 Měření úhlu otočného mechanismu

Systém umožňuje snímat polohu otočného mechanismu, kupříkladu oken, dveří či žaluzií za pomoci potenciometru. Ve spojení se silovými výstupy na motory by bylo možné měnit úhel otevření okna, a tím ovlivňovat rychlost proudění vzduchu a kvalitu ovzduší v místnosti. Jiné využití potenciometru jsou tzv. lankové snímače, které odměřují vzdálenost na principu lanka odvíjeného z bubínku.

Obr. 5.10: Lankový senzor

5.5.1 Zapojení snímače otočení

V této práci je realizováno čidlo otočení klasickým otočným potenciometrem dle zapojení 5.11. Program dokáže na základě hodnoty naměřené AD převodníkem na pinu PA4 napětí vypočítat úhel otočení lineárního potenciometru.



Obr. 5.11: Zapojení snímače otočení

5.6 Přítomnost osob

Senzory detekující pohyb osob se v automatizaci budov používají především z bezpečnostních a ekonomických důvodů. Z bezpečnostního hlediska se nabízejí dva pohledy – ten první je střežení objektu před vniknutím osob bez oprávnění, druhý pohled bezpečnosti jsou např. automaticky se rozsvěcející orientační světla při detekování pohybu v tmavých prostorech, kde by jinak hrozilo zranění v důsledku zakopnutí apod. Uvedený druhý bezpečností důvod souvisí také s ekonomickým – nebudeme zbytečně neosvětlovat místnost, ve které nikdo není.

Se záměrem zjistit přítomnost osob ve střeženém prostoru se používá několik typů senzorů. Běžně instalovaný typ senzoru je PIR (**P**assive **I**nfra**r**ed Detector), který poskytuje výstupu informaci ve dvou stavech – osoby jsou v pohybu a osoby nejsou v pohybu. Výhodou těchto čidel je jednoduchost instalace a nezávislost na nedostatku světla, nevýhodou je, že jejich činnost (počet falešných detekcí pohybu) ovlivňuje např. proudění vzduchu či teplo přicházející zvenčí a některá tato čidla mohou mít poměrně vysokou spotřebu oproti zařízením, jejichž činnost řídí.

Podobnou informaci o pohybu osob zprostředkovávají AIR detektory (**A**ctive **I**nfra**r**ed Detector, to této skupiny patří optická závora), které jsou méně citlivé na jiné zdroje tepla a světla, a většinou se instalují do průchozích prostor, protože mají výrazně nižší zorný úhel oproti PIR.

Třetí zde jmenovanou skupinou senzorů pohybu jsou fotocitlivé prvky umístěné do matice tvořící obrazový senzor. Obrazové senzory mohou realizovat naprogramovaný algoritmus detekce pohybu, mohou extrahovat popředí, rozpoznávat a identifikovat předměty nebo osoby, které se objeví v jejich zorném poli. Výhodou obrazových senzorů je, že poskytují komplexní vizuální informaci o stavu prostředí a že ve spojení s přizpůsobeným software mohou reagovat na libovolný podmět.

Nevýhoda obrazových senzorů je, že zpracování obrazu klade vyšší požadavky na výpočetní a paměťový prostor obsluhujícího počítače, proto může být algoritmus zpracování obrazu vykonáván na úrovni hardware – tím se sníží čas potřebný na zpracování obrazu, ale projeví se vyšší cena, kterou jsou integrované obvody implementující software doprovázeny.

V tomto systému sběru dat je realizováno detekování pohybu osob optickou závorou a CMOS obrazovým snímačem Raspberry Pi Camera Board (viz kapitola 10), který je oficiálně dodáván k Raspberry Pi. V tomto případě je použito druhé Raspberry Pi, které je na pozici satelitní podřízené jednotky a komunikuje s centrální jednotkou přes sběrnici RS485. Obraz z kamery je zpracován algoritmem Codebook, který je zaměřen na extrahování popředí metodou modelování pozadí a dokáže poměrně spolehlivě rozpoznat pohyb v obraze bez falešných detekcí.

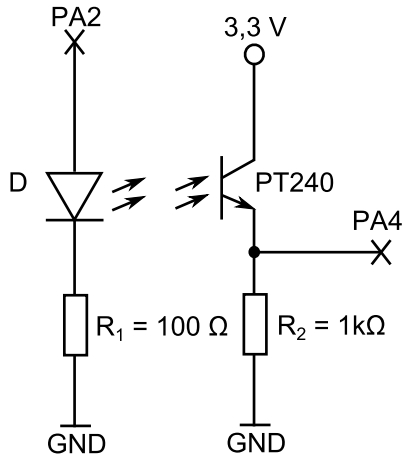
5.6.1 Zapojení optické závory

Optická závora zde realizovaná se skládá ze dvou oddělených obvodů – z vysílače a přijímače. Snímací část je realizována podobně jako senzor pro měření intenzity osvětlení popsaný v kapitole 5.1.2, schéma zapojení optické závory je níže. Princip funkce je následující:

Diodou D emituje paprsek světla na bázi fototranzistor PT240, ten se při přerušení paprsku světla během průchodu osoby uzavře a úbytek napětí na rezistoru R_2 poklesne. Úbytek napětí měří mikrokontrolér na pin PA6, který je připojen na rezistor R_2 a emitor tranzistoru, viz schéma 5.12: Zapojení optické zázvory.

Vysílací část závory je tvořena červenou LED a odporem R_1 o velikosti 100 Ω. Ideální je použít infračervenou diodu, která svou činností neemituje světlo lidským okem viditelné, poté může optická závora fungovat i jako prvek bezpečnostní. Katoda diody D je přes odpor R_1 připojena na zem. Anoda diody je vyvedena na pin mikrokontroléru PA2, která je napojen na výstup čítače TIM3, jenž pracuje v režimu PWM a moduluje světlo vyzařované diodou.

Pulsně šířkovou modulací vyzařovaného světla a nastavením citlivosti přijímače na frekvenci a energii vyzařované PWM modulací se docílí toho, že optická závora je méně náchylná na rušení světlem a teplem dopadajícím z okolí. Zároveň, je-li vysílací prvek zatěžován po velice krátkou dobu, je možné použití většího špičkového proudu, a tím zvýšit dosah závory na několik metrů bez přídavné optiky.



Obr. 5.12: Zapojení optické zázvory

6 Protokol Modbus na sběrnici RS485

Jak bylo navrženo v rozboru, jednotlivé uzly systému sběru dat komunikují jednoduchým a robustním protokolem Modbus na sběrnici RS485. V této kapitole je Modbus rozebrán z pohledu referenčního ISO/OSI modelu na jednotlivých vrstvách a jak jsou tyto vrstvy realizované v jednotkách systému sběru dat.

Modbus je otevřený, jednoduchý, spolehlivý a ověřený protokol vyvinutý v roce 1979 a i přes své starší datum vydání se vzhledem k výše jmenovaným vlastnostem stále používá v domovních a průmyslových prostředí.

Protokol je definovaný na sedmé aplikační vrstvě referenčního ISO/OSI modelu a je navržen pro vzájemnou komunikaci různých zařízení (původně pouze PLC) na různých sběrnicích a sítích (sem patří Ethernet, rádiový přenos, optické vlákno, RS-485). Komunikace mezi klientem (klient žádá o data) a serverem (ten data poskytuje) pracuje na principu předávání zpráv.

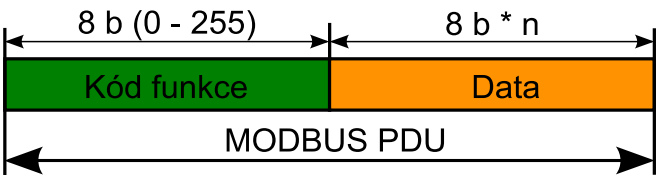
Při implementaci Modbusu na sériové lince se návrh řešení rozrůstá z jedné aplikační vrstvy na celkem tři vrstvy v rámci referenčního modelu. První přidaná vrstva je fyzická, která je zde realizována dle standardu RS485, druhá vrstva je spojová a pracuje dle protokolu Modbus Serial Line Protocol [2].

Vrstva	OSI/ISO model	Realizace
7	Aplikační	MODBUS Application Protocol
6	Prezentační	-
5	Relační	-
4	Transportní	-
3	Síťová	-
2	Spojová	MODBUS Serial Line Protocol
1	Fyzická	EIA/TIA-485 two-wire interface

Tabulka 6.1: MODBUS protokol na sériové lince v OSI/ISO referenčním modelu

6.1 Modbus na aplikační vrstvě

Protokol Modbus na aplikační vrstvě referenčního ISO/OSI modelu vymezuje komunikaci typu klient/server, kde si zařízení vyměňují zprávy/rámce PDU (**P**rotocol **D**ata **U**nit) v binární nebo textové podobě nezávisle na typu použité sběrnice nebo sítě. Základní PDU rámec se skládá z kódu funkce a datové části a teprve dle typu použité sítě jsou k PDU rámci přidána další data.



Obr. 6.1: MODBUS Protocol Data Unit

Kód funkce v obr. 6.1 zprávy PDU značí operaci, kterou má server provést. Různých operací je celkem 128, protože kódy funkcí s MSB v logické jedna (dekadicky to jsou kódy zpráv 128 až 255) jsou vyhrazeny pro chybové odpovědi. Za kódem funkce může následovat datová část, která případně upřesní operaci a podobu odpovědi. Obsahem datové části tak může být například adresa registru, který má být serverem přečten a zaslán klientovi zpět.

V případě úspěšného provedení požadované operace server odpoví opět rámcem PDU. Server do části kód funkce vloží kód vykonané funkce (indikuje správné provedení akce) a do datového pole uloží požadované informace. V případě nedokončení instrukce dosadí do zprávy chybový kód funkce a do datové části zapíše informace upřesňující důvod neprovedení operace.

6.1.1 Implementované Modbus funkce

Z množiny Modbus funkcí jsou v této práci využity dvě základní funkce dle specifikace protokolu, a to

- **funkce 0x03 Read Holding Registers,**
  která slouží ke čtení jednoho nebo více šestnácti bitových registrů satelitní jednotky,
- **funkce 0x06 Write Single Register,**
  tato funkce se používá k zápisu jedné šestnáctibitové proměnné do paměti/registru satelitní jednotky. Kromě toho je v této práci využita k zaslání požadavku na spuštění měření satelitní jednotkou - k tomuto řešení bylo přistoupeno z toho důvodu, aby bylo zamezeno blokovacímu čtení dat z paměti MCU, delší dobu než je nutné.
  K výraznému blokování by např. docházelo při čekání na sejmutí dat ze senzoru SHT11, který při dvanáctibitovém rozlišení vlhkostního, resp. čtrnáctibitového rozlišení teplotního čidla provádí měření přibližně 80 ms, resp. 320 ms. Tento čas může být centrální jednotkou využit k obsluze dalších zařízení.
  Při čekání na data by také mohlo dojít k vypršení času pro odpověď uzlu, tím by byl příjem zprávy přerušen a označen za neúspěšný.

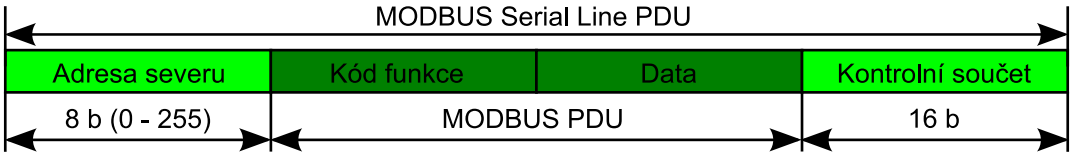
K funkcím protokol Modbus definuje čtyři typy chybových zpráv, které jsou zasílány zpět klientovi, pokud server nedokáže dokončit požadovanou operaci. Chybové odpovědi jsou

- Neplatná funkce – illegal function, kód 0x01
- Neplatná adresa – illegal data address, kód 0x02
- Neplatná hodnota – illegal data value, kód 0x03
- Selhání zařízení – slave device failure, kód 0x04, neimplementována.

6.2 Modbus na spojové vrstvě – sériová linka

MODBUS Serial Line Procokol použitý na spojové vrstvě řídí přístup k médiu na principu Master/Slave, kde může být k sériové lince připojeno pouze jedno Master zařízení a přibližně dvě stě čtyřicet podřízených zařízení. Komunikaci vždy zahajuje master a vždy vyšle jen jednu zprávu, poté čeká na odpověď.

Základní rámec Modbusu z aplikační vrstvy se na sériové lince rozšiřuje o adresu slave zařízení a o pole kontrolního součtu, a tak vzniká Modbus Serial Line PDU.



Obr. 6.2: MODBUS Serial Line Protocol Data Unit

Adresovány jsou pouze podřízené uzly, master zařízení adresu nemá. Adresový prostor je omezen rozsahem osmibitové adresace, tedy na adresy 0 až 255. Z tohoto prostoru je adresa 0 vyhrazena pro broadcastové vysílání (pouze master může vysílat zprávu pro všechny uzly v síti, jednotlivé uzly na zprávu neodpovídají). Adresy 248 až 255 jsou rezervovány. Zbývá tak místo pro dvě stě čtyřicet sedm podřízených zařízení.

Integrita přenášených zpráv je zajištěna šestnáctibitovým kontrolním součtem, ten se liší dle použitého vysílacího módu. V RTU režimu je použita hashovací funkce CRC, v ASCII módu se vysílání LRC kontrolní součet, viz následující kapitola Vysílací módy.

6.2.1 Vysílací módy

Modbus na sériové lince umožňuje dva vysílací režimy, a to ASCII mód a RTU mód. Režim komunikace určuje, v jakém formátu jsou data vysílána a jak je určen začátek a konec vysílání.

MODBUS RTU: V RTU (**R**emote **T**erminal **U**nit) módu se přenáší osm bitů informace v jednom jedenácti bitovém rámci (bity jsou popořadě start bit, osm datových bitů, bit sudé parity a stop bit). Integrita celé zprávy je zaručena CRC (**C**yclic **R**edundancy **C**heck) součtem a pomocí sudého paritního bitu. Vysílání jedné zprávy musí být souvislé, mezery mezi znaky nesmí být širší než jeden a půl znaku. Konec vysílání je dán odmlčením se po dobu vysílání odpovídající tři a půl znaku (přibližně 40 bitů). Výhodou oproti ASCII režimu je vyšší propustnost dat.

MODBUS ASCII: V ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) režimu je osm bitů informace přeneseno ve dvou desetibitových rámcích – každý rámec obsahuje jeden sedmibitový ASCII znak (oproti osmi datovým bitům v RTU je zde použito bitů sedm). Tím se snižuje datová propustnost (např. v RTU módu přeneseme hodnotu 0x5B v jednom rámci, v ASCII režimu odešleme v prvním rámci znak ‘5’, hexadecimálně 0x35, v druhém rámci znak ‘B’, hexadecimálně 0x42). Integrita zprávy je zajištěna LRC (Longitudinal Redundancy Check) součtem. Začátek vysílání zprávy je určen znakem ‘:’, konec zprávy je indikován dvojicí řídicích

znaku CR a LF. Díky odlišnému řízení komunikace je povolená mezera mezi dvěma odesílanými znaky v rámci jedné zprávy až jedna vteřina.

Při komunikaci mezi centrální jednotkou a satelitními jednotkami je implementován RTU režim.

6.3 Modbus na sériové lince – fyzická vrstva

Protokol Modbus je definován na různých médiích, např. na sériových linkách RS232, RS422 a RS485, na optických, rádiových a Ethernetových sítích. V této práci je použit standard RS485 v dvouvodičové poloduplexové verzi.

6.3.1 Standard RS485

Fyzická vrstva protokolu Modbus je realizována sériovou dvouvodičovou poloduplexní verzí sběrnice RS485. Tento průmyslový standard zaručuje vysokou odolnost proti rušení díky zakroucenému páru vodičů a symetrickému vedení, čímž se snižuje vliv elektromagnetického záření přijatého z okolí na přenos dat. Zároveň zakroucení vodičů omezuje množství energie vyzářené.

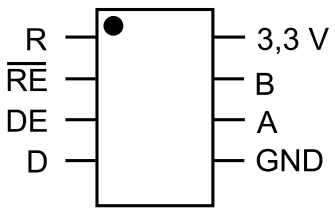
Vedení je tedy symetrické, což znamená, že přenášená data jsou definována rozdílem napětí na vodičích A a B. V klidovém stavu je na vodiči B větší napětí oproti vodiči A, logická stav 1 je daný napětím vyšším o 200 mV na vodiči B, logický stav 0 je určen přesně opačným napětím (na vodiči A napětí větší o 200 mV).

Standard RS485 umožňuje připojení třiceti dvou zařízení na jednom segmentu sítě, sítě je možné propojit izolovaným opakovačem. Délka sběrnice je omezena vysílací komunikační rychlostí, při použití rychlosti 115 200 Baud by neměla vzdálenost mezi nejodlehlejšími uzly přesahovat sedm set metrů [1]. Standard nedovoluje jiné topologie sítě než sběrnici na sdíleném médiu. Z důvodu zamezení odrazů na sběrnici se na obou konci kroucené dvojlinky nachází terminátory o velikosti 120 Ω. Protože je v této práci implementována poloduplexová verze sběrnice, musí být řízen směr komunikace.

Vzhledem k tomu, že Raspberry Pi a mikrokontroléry STM32F050 disponují pouze UART rozhraním, bylo nutné použít budič UART/RS485.

6.3.2 Transceiver ADM3485

Použitým budičem UART/RS485 je obousměrný třístavový transceiver ADM3485 integrovaný v pouzdře DIL-8 od společnosti Analog Devices.



Obr. 6.3: Diferenční transceiver ADM3485

Transceiver vyžaduje ke svému běhu napájení o velikosti 3,3 V (pin 8 na obrázku výše). Piny na pravé straně (piny A a B) jsou vývod na sběrnice RS485.

Pin D (Driver), resp. R (Receiver) slouží pro příjem, resp. odesílání dat na straně UART rozhraní, pin DE (Driver Enable) povoluje v logické 1 odesílání dat po sběrnici RS485, pin RE (Receiver Enable) povoluje příjem dat v logické 0. Piny RE a DE reagují na opačné stavy řídicích signálů, proto je možné měnit směr komunikace poloduplexní verze sběrnice RS485 jedním signálem. Funkci obvodu pro příjem a odesílání dat zachycují pravdivostní tabulky 6.2. Ve výchozím stavu by měl být signál RE v logické 0, aby zařízení mohlo naslouchat příchozí komunikaci.

DRIVER

INPUT D	ENABLE DE	OUTPUTS A	OUTPUTS B
H	H	H	L
L	H	L	H
X	L	Z	Z

RECEIVER

DIFFERENTIAL INPUTS A–B	ENABLE $\overline{RE}$	OUTPUT R
$V_{ID} \geq 0.2$ V	L	H
-0.2 V $< V_{ID} < 0.2$ V	L	?
$V_{ID} \leq -0.2$ V	L	L
X	H	Z
Open	L	?

H = high level, L = low level, ? = indeterminate,
X = irrelevant, Z = high impedance (off)

Tabulky 6.2: Funkčnost obvodu ADM3485

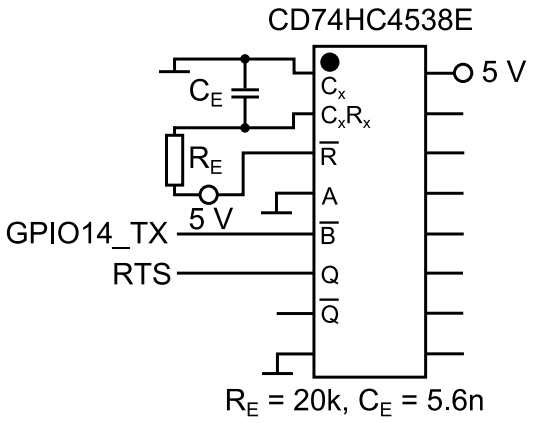
6.3.3 Realizace komunikace na Raspberry Pi

Raspberry Pi oproti mikrokontroléru disponuje jednoduchým full-duplex UART rozhraním bez hardwarové podpory komunikace standardu RS485 a protokolu Modbus, proto bylo nutné řízení směru dat vyřešit (specifikace protokolu Modbus nedovoluje, aby zařízení zároveň naslouchalo datům, která vysílá).

Prvoplánové řešení řízení toku dat softwarovou metodou bylo zavrhnuto, protože komunikující program na straně Raspberry Pi využívá ke komunikaci knihovnu pro Modbus protokol, kterou není žádoucí měnit (v případě instalování novější verze knihovny by bylo nutné měnit danou část kódu v knihovně).

Proto bylo přikročeno k **Hardware řízení směru komunikace:**

Řízení toku komunikace bylo uskutečněno na úrovni hardwaru pomocí monostabilního klopného integrovaného obvodu CD74HC4538E v pouzdře DIL-16. Tento obvod obsahuje dva MKO (**m**onostabilní **k**lopné **o**bvody). Ve schématu 6.4 jsou označeny pouze použité vstupní a výstupní piny jednoho MKO.



Obr. 6.4: Řízení směru komunikace obvodem MKO

Komunikace rozhraním UART s využitím MKO pracuje takto:

Monostabilní klopný obvod má jeden stabilní a jeden nestabilní stav. Stav MKO je vyveden na pin Q. Do nestabilního stavu se obvod dostane po detekování náběžné hrany na pinu A, resp. po detekování sestupné hrany na pinu $\overline{\mathrm{B}}$. Na piny C_x, C_xR_x je zapojen kondenzátor a rezistor, přičemž jejich časová konstanta

$$\tau = RC$$

určuje čas setrvání obvodu v astabilním stavu. Poté se překlopí zpět do stabilního stavu.

Od MKO požadujeme, aby řídil směr komunikace signálem pinu Q, nebo jeho negací, který je připojen na piny DE a $\overline{\mathrm{RE}}$ budiče ADM3485. Předpoklad je, že budič bude v nevysílacím stavu vždy data přijímat a že pomocný obvod bude umístěn mezi RPi a budič.

Aby budič vždy propustil data vyslaná jiným zařízením, musí být signál vyvedený na piny DE a $\overline{\mathrm{RE}}$ v logické 0. Proto na tyto piny vyvedeme signál Q, který je, pokud je MKO ve stabilním stavu, roven logické nule.

Ve výchozím, klidovém stavu je vysílací signál UART rozhraní (GPIO14_TX ve schématu výše) v logické 1, pokud tento signál spojíme s MKO s pinem $\overline{\mathrm{B}}$, který reaguje na sestupnou hranu, bude MKO v nestabilním stavu až po detekování start bitu při začátku vysílání (start bit je definován přechodem z logické 1 do logické 0) a s každou další přenášenou logickou 0 bude MKO resetován a tím je dovoleno vysílání dat.

Rezistor a kondenzátor tvořící časovou konstanta τ jsou zvoleny tak, aby přechod MKO do nestabilního stavu vyvolaném start bitem trval přinejmenším dalších 10 bitů při modulační rychlosti 115 200 Baud.

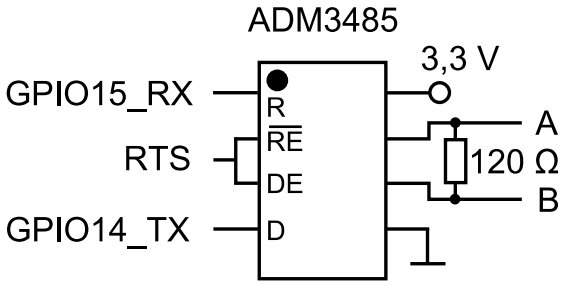
Velikost externím kondenzátoru C_E je rovna 5,6 nF , velikost externího rezistoru R_E je rovna 20 kΩ, časová konstanta τ je poté rovna

$$\tau = 0{,}7 * R_E * C_E = 0{,}7 * 20k * 5{,}6n = 78{,}4\ \mu s\ .$$

Prodleva zanášená do komunikace tímto způsobem řízení komunikace dosahuje velikosti časové konstanty a není tedy v tomto systému sběru dat kritická.

Zapojení Raspberry Pi ke sběrnici RS485

UART rozhraní na Raspberry Pi se nachází na patici P1, pinu 14 slouží k vysílání dat (GPIO14_TX) a pin 15 k příjmu dat (GPIO15_RX), použité zapojení s budičem znázorňuje schéma 6.5. Signál RTS je přiveden z MKO ze zapojení 6.4.



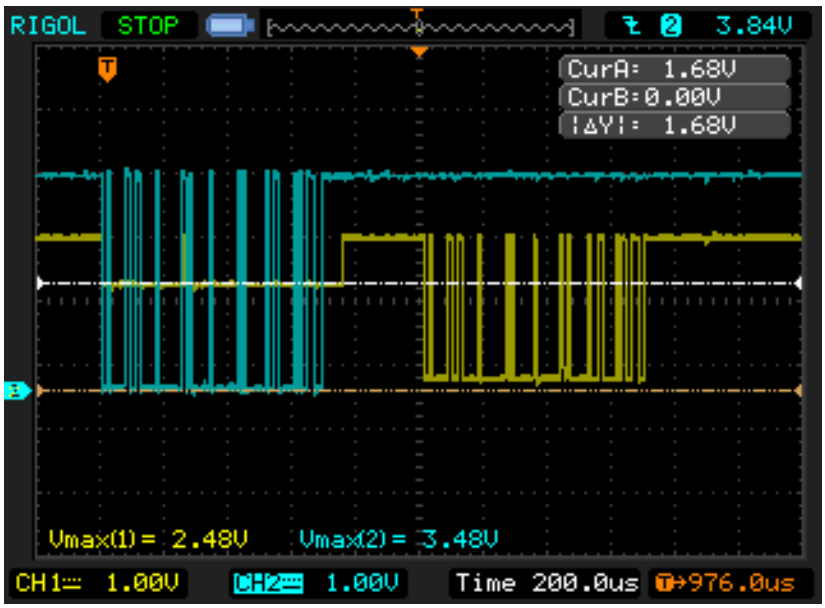
Obr. 6.5: Připojení RPi ke sběrnici RS485

Snímek obrazovky osciloskopu níže zachycuje komunikaci mezi centrální jednotkou a satelitní jednotkou.

Tyrkysovou barvou je vykreslen dotaz centrální jednotky, odpověď satelitní jednotky je zobrazena žlutou barvou.

Mikrokontrolér na žádost odpovídá přibližně po 300 µs, což je doba, která odpovídá mezeře značící konec vysílání zprávy – tři a půl znaku (35 bitů) při komunikační rychlosti 115 200 Baud prodleva τ odpovídá poměrně přesně minimální prodlevě vypočítané vztahem

$$\tau = \frac{1}{Baud\ rate} * mezera * počet\ bitů\ = \frac{1}{115\ 200} * 3{,}5 * 10 = 303{,}8\ \mu s\ .$$



Obr. 6.6: Ukázka komunikace mezi RPi a mikrokontrolérem

Rozdílná úroveň tyrkysového a žlutého napětí (dotaz a odpověď) je zaviněna tím, že v první realizaci platformy sběru dat byl použit budič UART/RS485 pracující na 5 V. Piny na straně MCU jsou 5 V tolerantní a nebylo potřeba velikost napětí přizpůsobovat, na straně Raspberry Pi piny UART rozhraní jsou 3,3 V tolerantní, proto byla velikost napětí snížena napěťovým děličem.

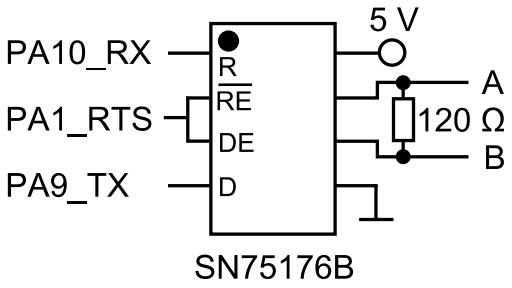
6.3.4 Realizace komunikace na satelitních jednotkách

Satelitní jednotky jsou připojeny ke sběrnici RS485 přes rozhraní USART, dle referenčního manuálu mikrokontroléru STM32F050 podporuje rozhraní USART jak komunikaci standardem RS485, tak protokolem Modbus, proto je možné připojit mikrokontrolér k budiči bez dalších pomocných obvodů.

V podkapitole 4.1.1 GPIO piny mikrokontroléru je poznamenáno, že rozhraní UART na MCU se nachází na pinech PA9 a PA10. Pin PA9 slouží pro vysílání dat (RX) a pin PA10 pro příjem (TX).

Pokud je pin `PA1_RTS` nastaven do první AF[4], tak vysílá signál RTS známý z RS232 komunikace[5], mikrokontrolér tento signál při zapnuté podpoře RS485 používá řízení směru.

Schéma realizace zapojení komunikace rozhraním UART s využitím signálu RTS a budičem UART/RS485 na straně MCU je poté následující:



Obr. 6.7: Zapojení budiče AMD3485 na straně MCU

[4] AF – alternativní funkce GPIO pinu
[5] Vysílací zařízení ve standardu RS232 signálem RTS oznamuje, že má data k odeslání.

7 Softwarové vybavení Raspberry Pi

V teoretickém rozboru systému sběru dat bylo naznačeno, že by na centrální jednotce reprezentované Raspberry Pi měly běžet přinejmenším dva programy, kde

- program první obstará komunikaci na sběrnici RS485 se satelitními jednotkami a archivaci dat,
- program druhý prostřednictvím webového prohlížeče vizualizuje naměřená data.

Následující kapitola je zaměřena na popis, obsluhu, ladění a funkčnost vyjmenovaných programů.

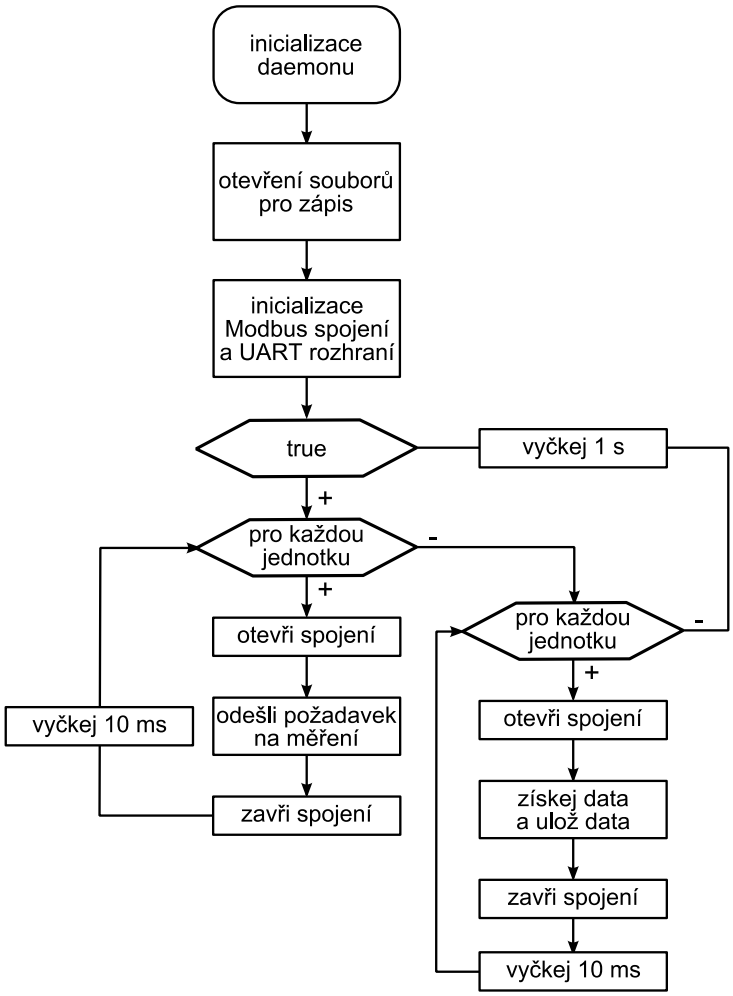
7.1 Program komunikace na sběrnici RS485

Máme k dispozici UART rozhraní na Raspberry Pi, satelitní jednotky připojené ke sběrnici RS485, která je poloduplexní, a proto bude nutné implementovat řízení směru komunikace. Rozhraní UART se nachází na patici P1 na pinech 14 a 15, viz 3.2. Na UART rozhraní je připojen budič UART/RS485. Nyní potřebujeme vytvořit program, který bude získávat naměřená data ze satelitních jednotek a archivovat je ve vhodném formátu.

Program ke své činnosti nevyžaduje interakci s uživatelem a pouze reaguje na události, a tak může běžet na pozadí operačního systému – tato kategorie programů se nazývá v systémech Unix daemon a služba/service pod operačním systémem Windows. Daemon získává data pomocí techniky zvané polling – neustále se každé jednotky dotazuje – a data ukládá do souboru v tabulkovém formátu CSV. V jednom řádku CSV souboru jsou informace o čase získání dat (timestamp), o adrese serveru, který data poskytl, a jednotlivá data. Všechny údaje v jednom řádku jsou odděleny čárkou (proto je formát pojmenován CSV – **C**omma-**s**eparated **v**alues).

Navrhovanou činnost daemonu zachycuje vývojový diagram 7.1: Vývojový diagram daemonu.

K příkazům „otevři spojení“, „získej data“ apod., které jsou zobrazeny v diagramu, musí Raspberry Pi implementovat protokol Modbus na úrovni hardware, nebo daemon software úrovni.



Obr. 7.1: Vývojový diagram daemonu

7.1.1 Protokol Modbus na Raspberry Pi

Je možné několika způsoby dosáhnout toho, aby Raspberry mohlo komunikovat protokolem Modbus. Na rozdíl od mikrokontrolérů řady STM32F0 z pohledu hardware Raspberry Pi nepodporuje jednoduše[6] ani protokol Modbus, ani výměnu dat přes sběrnici RS485, proto zbývají software řešení.

Knihovna libmodbus

Jedno takové řešení je knihovna libmodbus [25], která implementuje základní funkce protokolu Modbus, podporuje přenosy dat v režimech ASCII, RTU. Napsána je v jazyce C.

Knihovna libmodbus není standardní součástí repozitáře Raspbianu, proto je nutné instalovat knihovnu do operačního systému manuálně. Prvním krokem je stáhnutí knihovny z webových stránek knihovny nástrojem wget.

```
>> wget http://libmodbus.org/releases/libmodbus-3.0.6.tar.gz
```

[6] Bylo by možné vytvořit ovladače pro hardware a zavést je do jádra systému, ty by pak plnily tuto úlohu na nižší úrovni.

Po rozbalení knihovny a sestoupení do rozbaleného adresáře knihovny

```
>> tar -xvzf libmodbus-3.0.6.tar.gz
>> cd libmodbus-3.0.6
```

je nutné vygenerovat Makefile, vytvořit binární soubory knihovny a zavést je do systému. To se provede příkazy:

```
>> ./configure
>> make
>> sudo make install
```

Poslední příkaz `make install` vloží soubory knihovny do adresáře `/usr/local/include/modbus`. Tuto cestu poté předložíme kompilátoru jako jeden z parametrů, aby věděl, kde má knihovnu libmodbus hledat.

Program s knihovnou libmodbus

Daemon se nachází v Raspberry v adresáři `/home/pi/Workplace/daemon/`. Skládá se ze dvou souborů `modbuslogger.c` a `modbuslogger.h`. Do programu je vložena knihovna libmodbus direktivou preprocesoru:

```
#include <modbus.h>
```

Nyní máme k dispozici následující funkce knihovny

- `modbus_t *ctx`
  vytvoření proměnné, která obsahuje parametry spojení,
- `ctx = modbus_new_rtu("/dev/ttyAMA0", 115200, 'N', 8, 1);`
  inicializace UART rozhraní, kde první parametr je umístění UART rozhraní, poté následují parametry spojení. Popořadě: Baudrate, parita, počet datových bitů, počet stop bitů
- `rc = modbus_set_slave(ctx, SERVER_ADDRESS);`
  volba satelitní jednotky
- `rc = modbus_connect(ctx);`
  navázání spojení s vybranou satelitní jednotkou

Dvě implementované funkce na straně mikrokontroléru odpovídají funkcím knihovny

- **Write Single Register** odpovídá `rc = modbus_write_register(ctx, 0xF0, 0);`
  kde první parametr je proměnná spojení, druhý je šestnáctibitová adresa registru a třetí je šestnáctibitová hodnota pro zapsání do registru,
- **Read Holding Registers** je funkce `rc = modbus_read_registers(ctx, 0, 7, data);`
  kde druhý parametr je 16b adresa prvního registru, druhý parametr je počet registrů k přečtení a poslední parametr je ukazatel do paměti Raspberry Pi, kam mají být data uložena

7.1.2 Kompilace a spuštění programu

Program je napsán v kompilovaném programovacím jazyce C, pro potřeby vývoje je v adresáři daemonu vytvořen jednoduchý Makefile. Napsaný program se poté přeloží příkazem:

```
>> make
```

Kompilování bez souboru Makefile se provede příkazem:

```
>> gcc -Wall -g modbusloggerd.c -o dp_modbus_loggerd
       -I /usr/local/include/modbus -lmodbus -lm
```

kde parametry

- `-Wall` a `-g` povolují výpis varování kompilátoru při překladu,
- `-I /usr/local/include/modbus` překladači gcc říká, kde má hledat vložený hlavičkový soubor vloženou direktivou preprocesoru `#include`,
- `-lmodbus` a `-lm` oznamují, že program se má nalinkovat se sdílenými knihovnami `<modbus.h>` a `<math.h>`.

Zkompilovaný program se poté spustí příkazem:

```
>> ./dp_modbus_logger
```

7.1.3 Správa daemonu – vypnutí, zapnutí, aktualizování

Pokud chceme vidět spuštěné instance daemonu, můžeme tak učinit kombinací příkazů

```
>> ps aux | grep modbus
```

které vypíší veškeré programy mající řetězec *modbus* v názvu. V případě nalezené běžící instance programu bude výpis v konzoli vypadat přibližně takto:

```
>> root 2149 0.0 0.1 2204 616 ? S 21:42 0:00 /etc/rc2.d/S03dp_modbus_loggerd
>> pi  6773 0.0 0.2 3548 800 pts/1 S+ 21:48 0:00 grep --color=auto modbus
```

První proces spuštěný uživatelem `root` je daemon, jeho PID (identifikátor procesu) je 2149. Druhý proces je nástroj grep, který redukoval výpis nástroje ps. Aktivní daemom se ukončí napsáním:

```
>> sudo kill -9 2149
```

Kde číslo 2149 je PID procesu. Daemon je opět možné spustit příkazem.

```
>> sudo /etc/init.d/dp_modbus_loggerd
```

Pokud byly provedeny úpravy ve zdrojovém kódu programu, je zapotřebí daemon znovu zkompilovat a vložit do adresáře `/etc/init.d/` (zde jsou uloženy veškeré programy spouštěné během startu systému). K přesouvání, resp. kopírování souborů slouží nástroje `mv`, resp. `cp`. Do výše zmíněného adresáře může zapisovat pouze uživatel s oprávněním administrátora systému, proto příkaz začíná řetězcem `sudo`. Celý příkaz pro zkopírování zkompilovaného programu vypadá takto:

```
>> sudo cp dp_modbus_loggerd /etc/init.d/
```

Daemon z adresáře `/etc/init.d/` spustíme příkazem.

```
>> service dp_modbus_loggerd
```

7.2 Program prezentující data

Nyní v systému Raspbian běží daemon, který obstarává data ze satelitních jednotek a ukládá je do souboru ve formátu CSV. Univerzálním řešením prezentace dat tak, aby řešení nebylo přímo závislé na centrální jednotce (např. specifickým programem běžícím pod jedním operačním systémem), je zpřístupnění dat prostřednictvím webového prohlížeče, který je v dnešní době základním vybavením každého počítače.

K této činnosti je nutné nainstalovat webový server, který bude přijímat požadavky od webových prohlížečů a vyřizovat je. V dnešní době je nejpoužívanějším webovým serverem Apache HTTP Server [26], výrazně méně používaným serverem je Internet Information Services vytvořený společností Microsoft.

Použité technologie, které vykreslí data v HTML stránce, jsou:

- **HTML** – značkovací jazyk, který vytváří základní struktury stránky
- **CSS** – kaskádové styly, které graficky formátují obsah
- **PHP** – skriptovací jazyk prováděný na straně serveru
- **JavaScript** – skriptovací jazyk na straně klienta, zajišťuje interakci s uživatelem
- **jQuery** – JavaScriptová knihovna, usnadňuje použití JavaScriptu v prohlížeči
- **D3JS** – JavaScriptová knihovna, která data vykresluje pomocí **SVG**

7.2.1 Příprava prostředí – instalace Apache HTTP Server a jiných balíčků

Webový server Apache se nachází v oficiálním repozitáři Raspbianu, a tak instalace probíhá jednoduše pomocí nástroje apt-get. Prvně je optimální aktualizovat repozitář příkazem:

```
sudo apt-get update
```

Poté se Apache HTTP Server nainstaluje příkazem:

```
sudo apt-get install apache2
```

Po nainstalování serveru a jeho spuštění (příkaz `sudo service apache2 restart`, popř. se spustí po restartu systému) je možné doplnit podporu skriptovacího jazyka PHP. Potřebné balíčky se také nachází v repozitáři a do systému se nainstalují příkazem:

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

Správnou funkčnost nainstalovaného serveru a balíčků lze ověřit vložením HTML stránky do adresáře `/var/www`. Apache server interpretuje všechny soubory umístěné v tomto adresáři a zadáním url adresy `localhost` ve webovém prohlížeči.

7.2.2 Tvorba HTML stránky s využitím jQuery a D3JS

Chceme-li využít javascriptové knihovny jQuery a D3JS při tvorbě stránek, vložíme je do mezi tag <head> ve zdrojovém kódu stránky (řádky 6 a 7) následujícím způsobem.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Systém sběru dat s RPi</title>
5      <meta charset="utf-8">
6      <script src="http://d3js.org/d3.v3.js"></script>
7      <script src="http://code.jquery.com/jquery-1.11.2.min.js"></script>
8      <link rel="stylesheet" type="text/css" href="styles.css">
9    </head>
10   <body>
11   <script> ... </script>
12   <input id="address" type="text" placeholder="address (e.g. 0x81, ...)">
13   <button onclick='AddGraph($("#address").val())'>Add Graph</button>
14   <button onclick='RemoveGraph($("#address").val())'>Remove Graph</button>
15   </body>
16 </html>
```

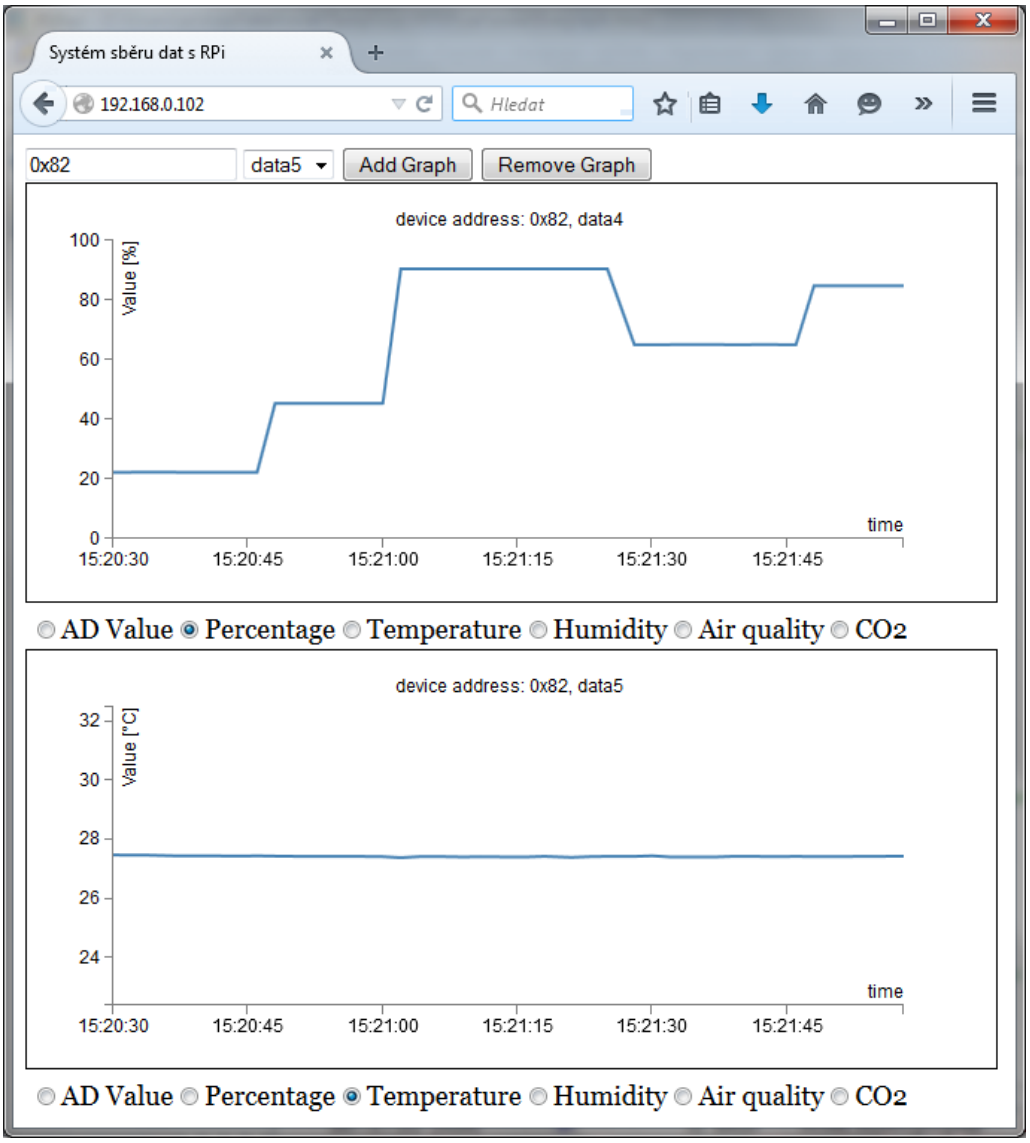
Na řádcích 6 a 7 se vykoná stažení knihoven z daného umístění v Internetu (tedy není možné používat tuto stránku bez připojení k Internetu – je možné to vyřešit stažením knihoven na server). Na řádku 8 proběhne načtení CSS stylů uložených v souboru *styles.css*.

Použití knihovny jQuery je vidět na řádcích 13 a 14, kde se pomocí ID selektoru `$("#address")` a funkce `val()` získá adresa vložená uživatelem do textového pole identifikovaného ID hodnotou „address“. Tato jQuery operace je zavolána po stisknutí tlačítka „Add Graph“ viditelného na stránce. Graf se odebere obdobným způsobem – také je identifikovaný adresou satelitní jednotky, jejíž data vizualizuje.

Stránka generovaná prohlížečem je k vidění níže a je možné přidat na jednu stránku libovolný počet grafů, pokud každý graf zobrazuje jiná data jedné satelitní jednotky (kombinace adresy a zobrazených je použita k identifikaci grafu). Pod grafem je možné zvolit, jak budou data kvantována – ve zdrojovém kódu stránky jsou zahrnuty vzorce pro přepočet hodnoty poskytnuté AD převodníkem na:

- Procenta (vzorec vychází z faktu, že AD převodník je 12bitový, a tak mohou naměřené hodnoty nabývat 4096 hodnot)
- Teploty a relativní vlhkosti ze senzoru SHT11 (vzorce jsou v kapitole 5.1.2)
- Koncentrace CO_2
- Kvalitu vzduchu

Grafy se za pomoci Javasriptu samy aktualizují každou sekundu a vždy zobrazují právě naměřená data.



Obr. 7.2: Stránka vygenerovaná za pomoci jQuery a D3JS knihovny

8 Softwarové vybavení mikrokontroléru

Tato kapitola popisuje vytvořené programové vybavení mikrokontroléru STM32F050, které jsou umístěné v satelitních jednotkách. První část kapitoly je věnována ladění mikrokontroléru (jak vytvořit program, jak ho nahrát do paměti MCU a jak ho ladit). Střední část se zabývá návrhem programu mikrokontroléru a závěr kapitoly implementací programu.

8.1 Vytvoření programu mikrokontroléru

Existuje řada vývojových prostředí pro tvorbu softwaru do mikrokontrolérů, které se liší možnostmi ladění, nástroji usnadňující psaní kódu a také pořizovací cenou. Neplacené verze vývojových prostředí většinou omezují maximální velikost kódu, který přeloží a nahrají do paměti mikroprocesoru. Mezi známá a používaná prostředí s širokými možnosti ladění a nástrojů pro vývoj softwaru patří

- EWARM (IAR Embedded Workbench),
- Atollic TrueSTUDIO (prostředí založené na Eclipse),
- MDK-ARM (Keil).

Všechny tyto programy mají v neplacené verzi omezení kódu na 32 kB. Prostředí zcela bez omezení jsou např.

- Sleepy Cat IDE (vyvíjené na fakultě FEL, ČVUT),
- EM::Blocks (derivace prostředí Code::Blocks).

K tvorbě software mikrokontroléru bylo v této práci použito vývojové prostředí IAR Embedded Workbench for ARM 6.60 v neplacené verzi volně dostupné ke stažení na stránkách výrobce [27].

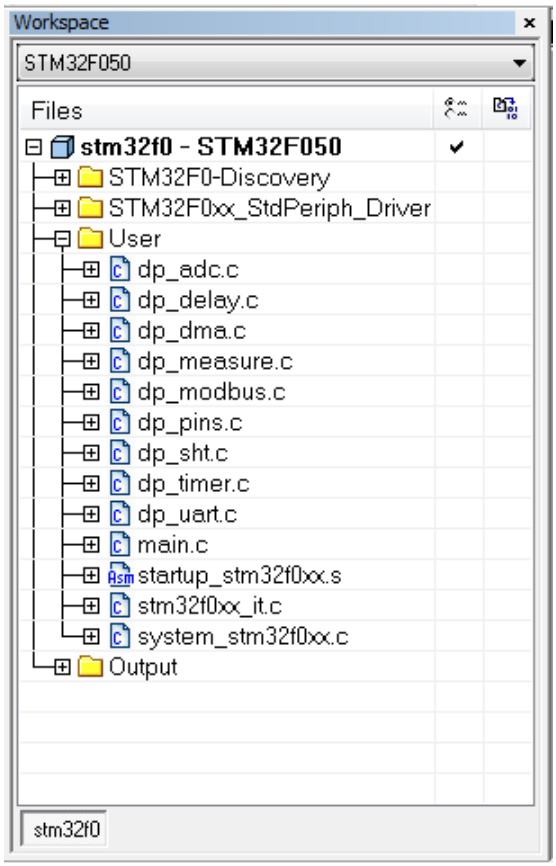
8.2 Spuštění a obsluha programu

Po spuštění vývojového programu IAR a otevření pracovního prostředí (*File -> Open -> Workspace...*, soubor stm32f0.eww v adresáři *stm32f0**Project**stm32f050**EWARM*) vytvořeného v této práci se načte projekt.

V levé části nahoře může programátor přepínat mezi pracovními prostředí (Workspace)[7], viz snímek obrazovky 8.1. V levé části uprostřed vidí programátor strukturu programu rozdělenou do souborů se zdrojovým kódem, viz snímek obrazovky 8.2. Vytvořený program se přeloží a nahraje do paměti mikrokontroléru položkou *Download and Debug* v menu *Project* (klávesová zkratka Ctrl + D).

Obr. 8.1: Přepínání konfigurace v AIR

Po nahrání programu do paměti mikrokontroléru přejde vývojové prostředí do ladícího režimu, v kterém je možné program krokovat či sledovat obsah registrů mikrokontroléru. Postup nahrání programu do paměti MCU z harware hlediska je popsán v následující podkapitole.



Obr. 8.2: Struktura programu mikrokontroléru

[7] V IAR jeden workspace může obsahovat několik konfigurací např. informace pro překlad kódu pro různý hardware, definice preprocesoru, nebo pro různě omezený projekt (např. jedna konfigurace může obsahovat jen určité části kódu). Konfigurace se mění v *Project -> Edit configurations.*

8.2.1 Nahrávání programu do paměti mikrokontroléru

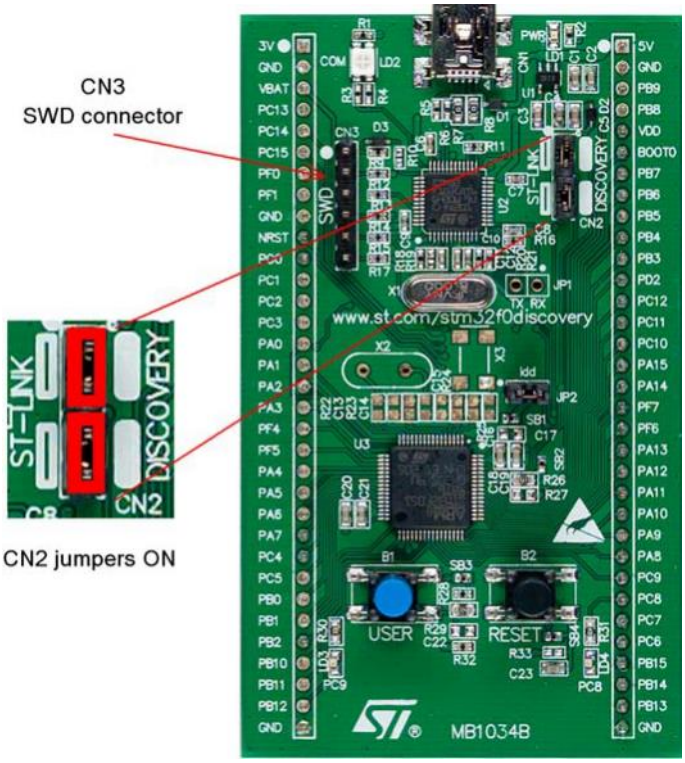
Jedno z programovacích rozhraní mikrokontroléru je rozhraní SWD (**S**erial **W**ire **D**ebug), jedná se o nástupce standardu JTAG a mezi jeho hlavní přednosti patří, že k nahrání programu a jeho ladění za běhu potřebuje pouze dva vodiče (oproti pěti-vodičovému rozhraní JTAG).

Program se do paměti mikroprocesoru nahrává za pomoci ladítka ST-LINK/V2, které existuje samostatně nebo jako součást vývojového kitu, a za pomoci příslušného softwaru (je možné použít STM32 ST-Link Utility nebo některé z výše vyjmenovaných vývojových prostředí).

Zda bude program nahrán do mikroprocesoru přítomného na vývojovém kitu nebo do externího zařízení, je dáno pozicí propojek na konektoru CN2, viz obr. 8.3. Jsou-li obě propojky přítomny, je laděn mikroprocesor na vývojovém kitu, nejsou-li přítomny, je laděn externí mikroprocesor.

Další možností nahrávání programu je pomocí bootloaderu, nicméně ten vyžaduje ke své funkci rozhraní USART, které není na většině dnešních počítačů přítomno a je zapotřebí použít např. USB/USART převodník.

V této práci bylo k nahrávání a ladění programu použito rozhraní SWD.



Obr. 8.3: Ladění mikroprocesoru na vývojovém kitu

Při nahrávání programu do paměti mikroprocesoru rozhraním SWD je nutné dodržet následující:

1. Odstranit propojky z konektoru CN2 (viz obr. 8.3) pro ladění externího zařízení, pro ladění mikroprocesoru na vývojovém kitu propojky naopak zapojit (v tomto případě následuje bod 3)
2. Propojit piny konektoru CN3 SWD s externím zařízením dle tabulky 8.1
3. Připojit vývojový kit k počítači USB kabelem typu „A na mini-B“ z konektoru CN1 přítomném na kitu (po připojení kabelu se rozsvítí LED LD1 PWR a L2 COM)
4. Nyní je možné nahrát program do paměti prostřednictvím aplikace STM32 ST-Link Utility nebo vývojovým prostředím (v případě prostředí IAR poslouží možnost *Project -> Download and Debug*)

Pin ST-LINK/V2 konektoru CN3 na vývojovém kitu	**Pin externího zařízení, pořadí – popis**
1 (označen bílou tečkou)	16 – VDD
2	20 – PA14
3	15 – VSS
4	19 – PA13
5	4 – NRST

Tabulka 8.1: Propojení ST-LINK/V2 a mikrokontroléru STM32F050 v pouzdře TSSOP20

8.3 Rozbor programu mikrokontroléru

Z vyjmenovaných senzorů v kapitole 5 vyplývá, jaké periferie mikrokontroléru jsou potřeba pro jejich připojení.

Senzor teploty a relativní vlhkosti STH11 s digitálním výstupem vyžaduje jakékoli dva GPIO, které jsou schopny pracovat ve vstupním a výstupním režimu.

Senzory intenzity osvětlení, kvality vzduchu, oxidu uhličitého, potenciometr v pozici úhloměru poskytují analogovou informaci o naměřených datech, proto jsou připojeny na kanály AD převodníku.

Optická závora také předává informaci o pohybu osob v analogové podobě, ale navíc je světlo emitováno diodou na vysílací straně modulováno pulsně šířkovou modulací, proto budou potřeba kromě AD převodníku vytvořit kód pro jeden z časovačů, který disponuje PWM.

K připojení ke sběrnici RS485 je potřeba nastavit rozhraní UART mikrokontroléru a využít hardware podporu standardu RS485 a protokolu Modbus, kterou nabízí.

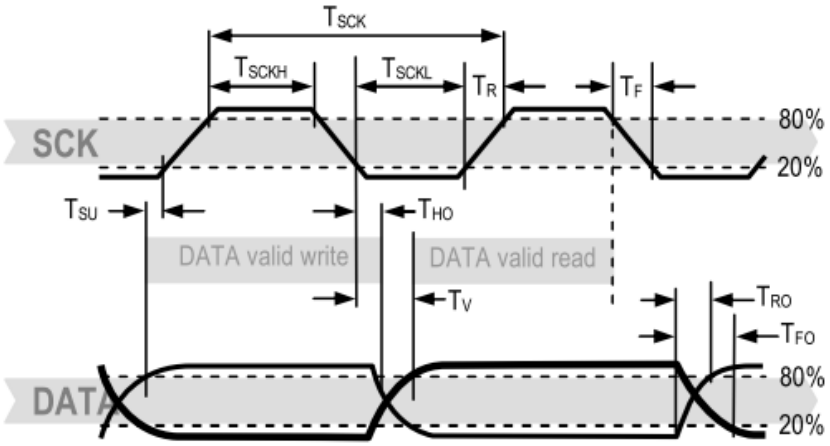
V následujícím textu bude popsáno, jak se jednotlivé periferie nastavují a jak program pracuje.

8.4 Program senzoru SHT11

Zapojení senzoru SHT11 je popsáno v kapitole 5.1.1, ke komunikaci používá dva vodiče, první vodič je označen CLK, který přenáší hodinový signál generovaný nadřazeným zařízením, druhý vodič slouží k obousměrnému přenosu dat. Datový vstup/výstup senzoru je třístavový.

Logika senzoru SHT11 je plně statická, a tak není definována minimální frekvence hodinového signálu. V datasheetu jsou zapsány časy, které je potřeba při komunikaci se senzorem dodržet (např. je tam zapsána dvojice časů T_{SU} a T_{HO} – čas T_{SU} určuje, po jakou dobu má být datový vodič v požadovaném stavu před hodinovým signálem a čas T_{HO} říká, po jakou dobu má vodič zůstat v onom stavu po sestupné hraně hodinové signálu).

Důležité z těchto časů jsou zapsány v tabulce níže. Příslušné časy je možné nalézt v diagramu komunikace nad tabulkou.



Obr. 8.4: Znázornění kritických časů senzoru STH11

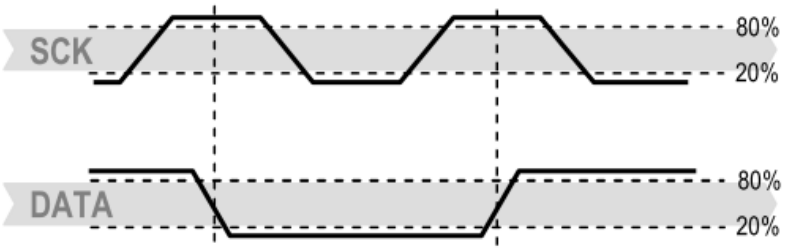
	Parametr	Podmínka	Nejméně	Typicky	Nejvýše	Jednotka
F_{SCK}	SCK frekvence	V_{DD} > 4,5 V	0	0,1	5	MHz
		V_{DD} < 4,5 V	0	0,1	1	MHz
T_{SCKX}	SCK high/low time		100			ns
T_R/T_F	SCK sestupná/ náběžná hrana		1	200		ns
T_{FO}	DATA sestupná hr.		30	40	200	ns
T_v	DATA valid time		200	500		ns
T_{SU}	DATA setup time		100	150		ns
T_{HO}	DATA hold time		10	15		ns

Tabulka 8.2: Kritické časy senzoru STH11

8.4.1 Průběh komunikace

Po připojení napájecího napětí senzor potřebuje 11 ms pro přechod do stavu spánku (neměřící stav s nižším proudovým odběrem). Po tomto čase je možné komunikaci se senzorem zahájit tzv. startovací sekvencí, která obnáší uvedení datového vodiče do logické 0, zatímco je SCK

v logické 1, a následného impulsu na vodiči SCK do logické 0 a zpět do log. 1, viz diagram níže. Teprve poté je povoleno přepnout datový vodič do logické 1 a může být odvysílán první příkaz pro senzor.



Obr. 8.5: Startovací sekvence

Každý příkaz se skládá z osmi bitů, kde první tři bity jsou vyhrazeny pro adresu senzoru (pouze adresa 000 je podporována) a pěti bitů příkazu. Základní příkazy schraňuje tabulka níže.

Příkaz	**Kód**
Měř teplotu	00011
Měř vlhkost	00101
Přečti Status Register	00111
Zapiš do Status Registru	00110
Reset (je nutná prodleva 11 ms před dalším příkazem)	11110

Tabulka 8.3: Příkazy senzoru SHT11

Správné přijetí příkazu senzor potvrdí bitem ACK (stáhne datový vodič do log. 0 po osmém CLK impulsu a po devátém CLK uvolní datový vodič).

Při odeslání příkazu pro měření teploty nebo vlhkosti musí mikrokontrolér počkat na dokončení měření po 20, 80 nebo 320 ms v závislosti na zvoleném rozlišení (8, 12 nebo 14 bitů). Dokončení měření senzor signalizuje stažením datového vodiče do logické 0 (signálme ACK). Senzor naměřená data uschová do doby, než budou přečtena.

Data naměřená senzorem jsou odeslána ve dvou bajtech po osmi bitech s volitelným CRC kódem v bajtu třetím. Každý přijatý bajt musí být potvrzen signálem ACK, všechna data jsou odesílána od nejvýznamnějšího bitu (MSB). Odeslání CRC bajtu se zamezí nepotvrzení přijatých dat ACK signálem.

8.4.2 Softwarová implementace komunikace se senzorem SHT11

Výrobce senzor prezentuje jako schopný komunikovat přes I^2C sběrnici, nicméně v datasheetu upřesňuje, že je tím myšlena schopnost senzoru být připojen do stávající I^2C sběrnice a svojí komunikací nerušit okolní komunikaci.

Proto je protokol senzoru emulován plně softwarově a ke komunikaci mohou být použity libovolné dva GPIO piny mikrokontroléru. Zdrojový kód je umístěn v souborech *dp_sht11.c* a *dp_sht11.h* a vychází z demonstračního kódu výrobce [28].

8.5 Senzory s analogovým výstupem

Senzory s analogovým výstupem jsou připojeny na kanály AD převodníku mikrokontroléru, který dokáže data jimi poskytované digitalizovat.

AD převodník je v kapitole 4.1.3 nakonfigurován tak, že má povolenou spolupráci s řadičem DMA, který po dokončení jedné konverze data odebere, a převodník tak může ihned začít s konverzí dalšího kanálu (DMA ukládá data do kruhového bufferu). Dokončení měření je signalizováno přerušením vyvolaným DMA řadičem, protože právě ten potlačuje přerušení od AD převodníku.

Měření se poté spustí ve funkci

```
void ADCStartMeasurement(void) {
  ADC_StartOfConversion(ADC1);
},
```

po změření všech kanálů AD převodníku DMA řadič nastaví bit TCIF1 v registru `DMAx->ISR`.

V programu je měření každého kanálu provedeno stokrát, mezivýsledky jsou sčítány a na konci měření se zprůměrují.

8.6 Program optické závory

Jak bylo řečeno v kapitole 5.6.1, je vhodné modulovat světlo emitované diodou pulsně šířkovou modulací (PWM), na mikrokontroléru je PWM dostupná na výstupu časovačů.

8.6.1 Pulsně šířková modulace

PWM je diskrétní modulace a přenáší informaci o analogovém signálu ve dvou stavech – zapnuto a vypnuto. Poměr těchto dvou stavů se nazývá střída a přechod mezi těmito stavy perioda modulace.

PWM výstup může sloužit jako náhrada DA převodníku, ve výkonové elektronice se PWM používá k řízení motorů a podobně je možné PWM použít zde k řízení modulování emitovaného světla diodou v optické závoře. Řízení LED pomocí PWM se lidského oku jeví jako pokles jasu LED (např. při střídě 50%), ve skutečnosti ale LED je v první fázi periody plně otevřena a v druhé plně zavřena. Tento princip umožňuje vysílat krátké záblesky světla na určité frekvenci o vyšší energii. Pokud je přijímač nastaven na stejnou frekvenci a jeho citlivost je přizpůsobena vysílané energii, tak se snižuje náchylnost senzoru reagovat na světlo dopadající z okolí a zvyšuje se jeho spolehlivost.

8.6.2 Generování PWM časovačem

Frekvence PWM se nastavuje podobně jako frekvence přerušení vyvolaná čítačem, viz kapitola 4.1.4. Použitý čítač TIM2 v tomto případě čítá vnitřní synchronní pulsy `CK_INT` a je tak nazývám časovačem. Frekvence pulsů `CK_INT` je odvozena od hodinové frekvence sběrnice APB1, která se vydělí hodnotou v registru `TIM2->PSC`. Poté dokáže časovač generovat PWM ve dvou módech. Perioda PWM je shodná s periodou časovače a nastavuje se v registru `TIM2->ARR`.

V první módě (nastaví se trojicí bitů `TIM_OCMode_PWM1` v registru `TIM2->CCMR2`), pokud čítač čítá nahoru, PWM výstup kanálu časovače je aktivní (generuje napětí rovné referenčnímu), dokud je `TIM2_CNT < TIM2_CCR1`, jinak je výstup neaktivní. Mód PWM 2 se nastaví trojicí bitů `TIM_OCMode_PWM2` v tom samém registru a výstup kanálu je komplementární k prvnímu módu.

Kromě PWM módu musíme nastavit střídu PWM modulace (ta se nastaví v registru `TIM2->CCR3`) a bitem `TIM_CCER_CC3E` v registru `TIM2->CCER` povolit výstup časovače na příslušném pinu (pin PA4 v alternativní funkci 2).

8.6.3 Synchronizace vysílače a přijímače optického senzoru

Nyní požadujeme synchronizaci časovače, který řídí LED, a AD převodníku, který měří úbytek napětí na přijímací straně senzoru. K tomu potřebujeme vědět, kdy začne být výstup časovače při generování PWM aktivní.

Dle datasheetu se nastaví bit `TIM_SR_CC3IF` v registru `TIM2->SR` v okamžiku shody registru `TIM2->CNT` a `TIM2->ARR`. To znamená, že pokud zvolíme PWM mód 2, tento bit bude signalizovat začátek emitování světla diodou. Ve zdrojovém kódu programu to vypadá takto:

```
void TIM2_WaitForPWM(void) {
  TIM2->SR &= ~TIM_SR_CC3IF;//reset TIM_SR_CC3IF flag
  while((TIM2->SR & TIM_SR_CC3IF) == 0);//wait for pwm pulse
}
```

8.7 Program protokolu Modbus se standardem RS485

Podpora standardu RS485 a protokolu Modbus mikrokontrolérem STM32F050 spočívá v řízení směru komunikace signálem RTS v případě prvním (viz kapitola 6.3.3, právě jmenovaná kapitola se také zabývá nastavením UART rozhraní). V případě protokolu Modbus je podpora implementována přerušením vyvolaným v okamžiku rozpoznání konce zprávy během příjmu dat.

Řízení směru komunikace povolíme nastavením bitu `DEM` v registru `USARTx->CR3`. Poté lze data odesílat přes UART běžným zápisem bajt po bajtu do datového registru `USARTx->TDR`, nebo za pomoci DMA řadiče, který data v definovaném úseku paměti odešle bez pomoci CPU. Odeslání dat v této situaci spočívá v pouhém určení, kolik slov se má odeslat (parametr funkce `length`), a povolením DMA, viz:

```
void USARTDMASend(uint32_t length) {
  DMA1_Channel2->CNDTR = length;
```

```
  DMA1_Channel2->CCR |= DMA_CCR_EN;
}
```

Podpora protokolu Modbus je dána dvěma body,

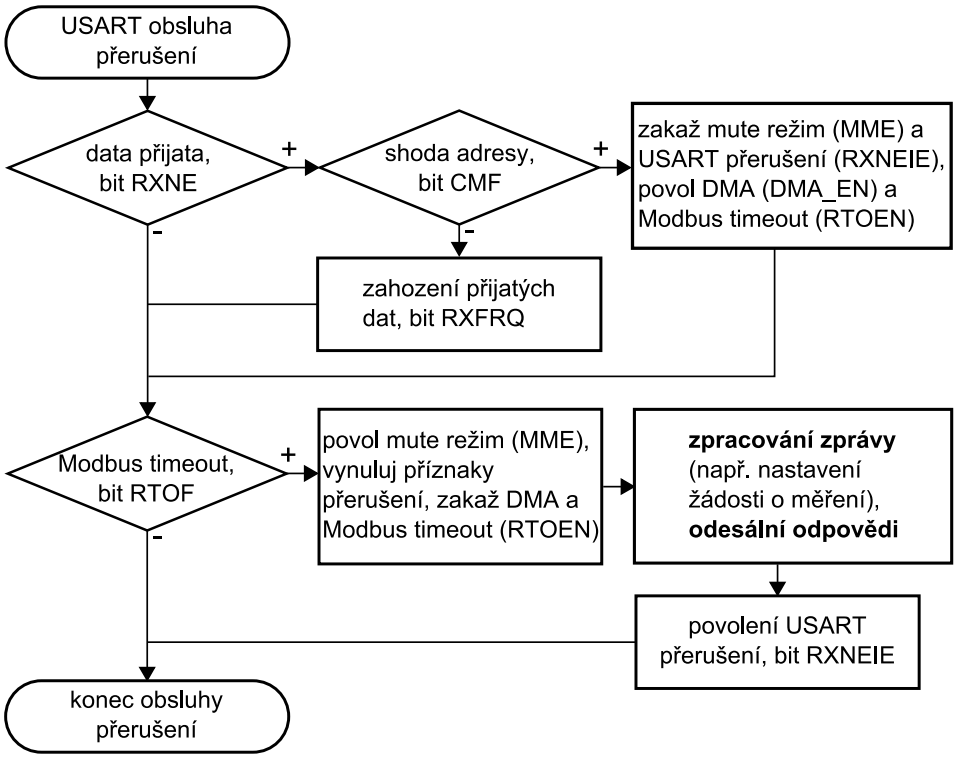
- rozpoznáním adresy zařízení při odposlechu komunikace (každá zpráva vyslaná protokolem Modbus začíná adresou cílového zařízení),
- detekcí konce vysílání zprávy v režimech RTU a ASCII.

Rozpoznání vlastní adresy je vhodné použít proto, že probíhající komunikace nezabírá výpočetní čas procesoru – pokud zařízení pozná, že zpráva není pro něj, uvede se do Mute režimu a dále se vysíláním nezabývá. Do naslouchacího režimu se zařízení opět přepne v okamžiku rozpoznání své adresy. Adresa zařízení se nastavuje v registru `USARTx->CR2`, bit přerušení značící rozpoznání své adresy je CMF v registru `USARTx->ISR`.

Detekce konce zprávy je klíčová vlastnost pro implementaci protokolu Modbus. V případě ASCII režimu mikrokontrolér vyvolá přerušení při příjmu znaku ukončujícím zprávu, právě tak vyvolá MCU to samé přerušení v RTU režimu, pokud rozpozná konec zprávy (tzn., pokud se vysílající zařízení odmlčí na déle než tři a půl znaku, tj. 35 bitů).

Povolení tohoto přerušení se nastaví bitem RTOEN bit v registru `USARTx->CR2` a povolí bitem RTOIE v registru `USARTx->CR1`. Přerušení samotné je potom signalizované bitem RTOF v registru `USARTx->ISR`.

Obsluhu přerušení před zahájením příjmu dat řadičem DMA znázorňuje vývojový diagram níže.



Obr. 8.6: Vývojový diagram obsluhy přerušení programu MCU

8.7.1 Zpracování Modbus zprávy

Z diagramu 8.6 vyplývá, že v obsluze přerušení vyvolané detekováním konce vysílání (Modbus timeout) je přijatá zpráva zpracována. Zpracování zprávy obnáší její rozložení do původní podoby dle kódu funkce.

Proto je nejprve přijatá zpráva přetypována do následující struktury

```
struct sRequest {
 uint8_t deviceAddress;
 uint8_t functionCode;
};
```

tímto příkazem

```
volatile struct sRequest * req = (struct sRequest*) USARTBuffer;,
```

adresa `USARTBuffer` ukazuje do paměti na přijatou zprávu. Umístění CRC kódu také zaslaného ve zprávě, který je potřebný k ověření nezměněnosti zprávy, je vypočteno kódem

```
volatile uint16_t* chcksum = (uint16_t*)(USARTBuffer + length - MDB_CRC_SIZE);,
```

kde proměnná `length` je rovna počtu přijatých bajtů a konstanta `MDB_CRC_SIZE` je rovna velikosti CRC kódu. Po ověření neporušenosti zprávy jsou přijatá data opět přetypována na základě hodnoty proměnné `functionCode` ve struktuře `sRequest`. Pokud je vypočtený a přijatý CRC kód nesouhlasí, zpráva je zahozena a mikrokontrolér se navrátí do stavu naslouchání.

Pokud kód funkce odpovídá Modbus funkci *ReadHoldingRegisters*, bude zpráva přetypována do koncové podoby

```
struct sReadHoldingRegisters {
 uint8_t deviceAddress;
 uint8_t functionCode;
 uint16_t firstRegisterAddress;
 uint16_t totalRegistersNumber;
 uint16_t chcksum;
};.
```

Nyní může mikrokontrolér dle adresy prvního registru a dle celkového počtu registrů odeslat zpět klientovi požadovaná data dle specifikace Modbus.

Podobně je v programu vytvořeno zpracování Modbus zpráv *WriteSingleRegister* a *Exception*. Data přijímána v pořadí Little-endian[8] po dvou bajtech, proto musejí být každé dva bajty zaměněny. Stejný princip platí pro odesílání dat.

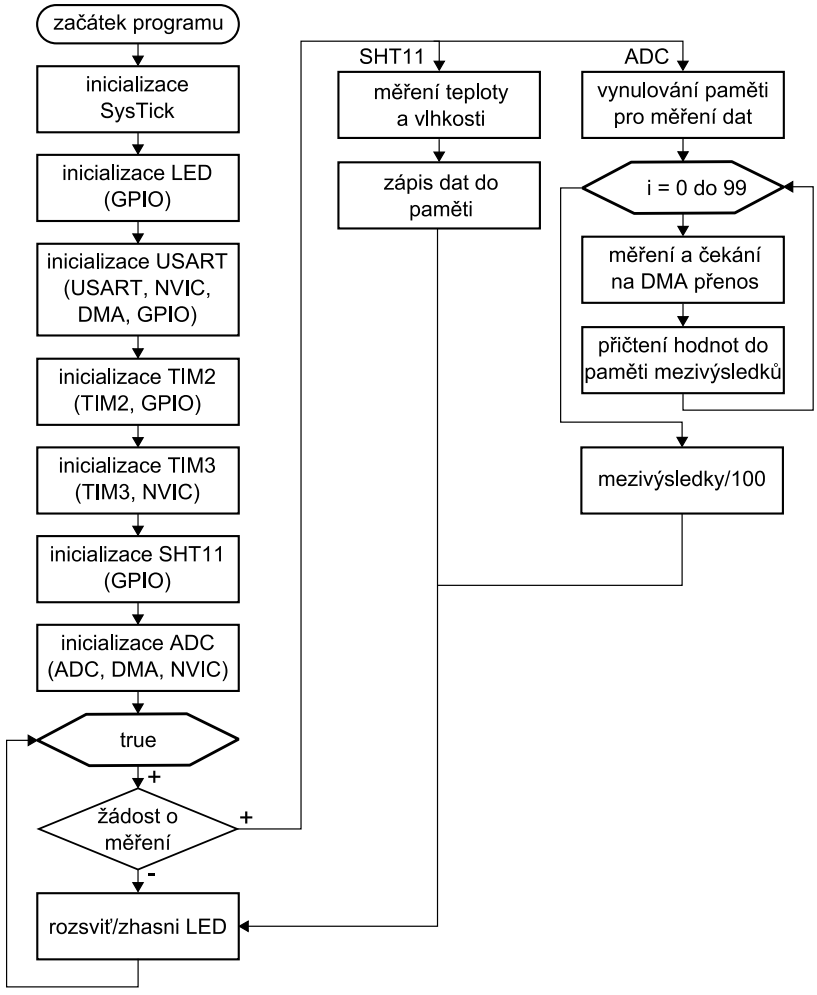
[8] Bajty jsou řazeny sekvenčně do paměti od nejméně významného po nejvíce vyznamný.

8.8 Průběh celého programu

Běh celého programu satelitní jednotky znázorňuje diagram 8.7. Program nejprve provede inicializaci periferií dle připojených senzorů (pokud není senzor připojen, je příslušná část zdrojového kódu zakomentována).

Z doposud nezmíněných periferií je v diagramu vidět jako první je konfigurace systémového časovače SysTick vestavěného přímo do jádra procesoru. Časovač je inicializován tak, aby každých deset milisekund generoval přerušení. Dle generovaného přerušení se v hlavní smyčce programu rozsvěcí a zhasíná LED frekvencí 1 Hz. LED je řízena hlavní smyčkou, aby bylo zřejmé, pracuje-li stále procesor mikrokontroléru.

Kromě rozsvěcení LED hlavní smyčka provádí snímaní dat ze senzorů dle příznaků, které byly nastaveny v přerušení UART rozhraní po zpracování zprávy od centrální jednotky.



Obr. 8.7: Vývojový diagram programu satelitní jednotky

9 Realizace systému sběru dat

V předchozích kapitolách byly postupně navrženy jednotlivé části systému sběru dat z hlediska software a hardware pro centrální jednotku, která je osazena miniaturním počítačem Raspberry Pi, a pro satelitní jednotky osazené mikrokontroléry STM32F050. K satelitním jednotkám jsou připojeny senzory, které snímají data. Naměřená data jsou zasílána centrální jednotce po sběrnici RS485 pomocí protokolu Modbus. Na centrální jednotce jsou data vizualizována ve webovém prohlížeči. Nyní je potřeba navrhnout zapojení jednotlivých modulů, vytvořit sběrnici RS485, rozvést napájení ke všem jednotkám a realizovat navrhované zapojení.

V rámci této práce byly vytvořeny dvě demonstrační desky. První deska reprezentuje případ jedné centrální jednotky a jedné satelitní jednotky, ke které je připojeno více senzorů. Druhá deska byla vytvořena se záměrem ukázat distribuovanost systému, a tak je k jedné centrální jednotce připojeno šest satelitních jednotek, kde ke každé je připojen jeden senzor.

9.1 Napájení jednotek

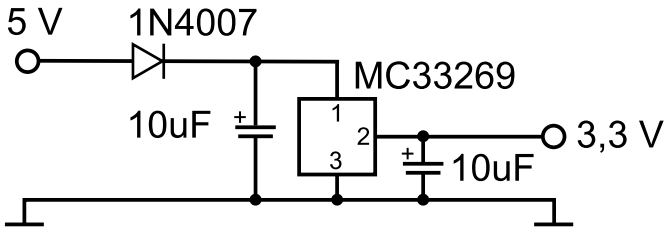
Před vytvořením prvního systému sběru dat musela být vyřešena otázka napájení. Bylo zváženo několik možností.

První možností je nezávislé napájení každé jednotky vlastním napájecím zdrojem. Nevýhodou tohoto řešení jsou mnohonásobné pořizovací náklady jedné jednotky, protože by každá jednotka vyžadovala napájecí adaptér z 230 V na požadovaných 5 V, resp. 3,3 V.

Druhou možností je rozvádět přímo požadovaných 5 V, resp. 3,3 V spolu s vodiči sběrnice RS485. Distribuce napětí 3,3 V byla také zavrhnuta, protože toto napětí slouží nejenom k napájení jednotlivých obvodů (kde by nepatrný úbytek napětí měl marginální dopad na funkčnost obvodů), ale také jako referenční napětí AD převodníku. Při různém referenčním napětí AD převodníku by satelitní jednotka zasílala data vztažená k tomuto napětí a provedené měření by nebylo objektivní. Proto bylo přistoupeno k rozvodu 5 V a následným přizpůsobením napětí na 3,3 V pomocí napěťového regulátoru.

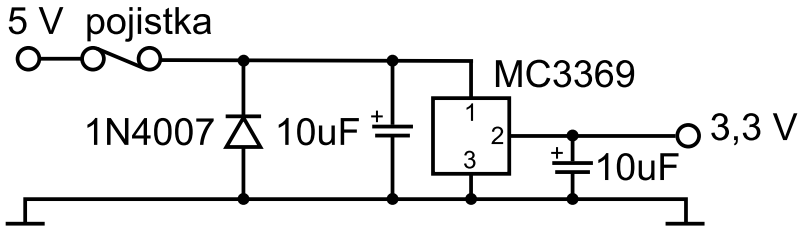
Při převodu napětí z 5 V na 3,3 V musí regulátor patřit do skupiny tzv. „low dropout voltage“, neboli je schopen přizpůsobit výstupní napětí na úroveň, která je velice blízko vstupnímu napětí. Jedním takovým regulátorem je obvod MC33269, který dokáže usměrnit napětí na velikost 3,3 V a má dropout napětí 1 V. Tento dropout dává zároveň prostor umístit do obvodu ochranu proti přepólování prostřednictvím sériově zapojené křemíkové diody 1N4007. Ve schématu napájecího obvodu 9.1 jsou také přítomny blokovací kondenzátory, které jsou

vyžadovány v datasheetu napěťové regulátoru k zaručení správné funkčnosti integrovaného obvodu MC33269.



Obr. 9.1: Napájecí obvod se sériově zapojenou diodou

Některé satelitní jednotky ale požadují také 5 V pro potřeby napájení senzorů (5 V ± 0,2 V), zde by použití obvodu 9.1 nedodávalo potřebné napětí. Proto byla dioda zapojena paralelně spolu s trubičkovou pojistkou, aby v případě přepólování napětí byla zničena pouze pojistka a ne např. některý z dražších senzorů. Schéma zapojení je níže.



Obr. 9.2: Napájecí obvod s paralelně zapojenou diodou

Jedním z vytyčených cílů této práce je vyčíslení ceny celého systému sběru dat. Cena samotných součástek tohoto obvodu je shrnuta v tabulce níže. Ceny v tabulce uvedené jsou uváděny s množstevní slevou deseti kusů. Všechny ceny jsou uvedeny včetně DPH.

Součástka	**Cena za kus**	**Počet**	**Cena**
Dioda 1N4007	0,79 Kč	1	0,79 Kč
El. kond. 22nF	0,55 Kč	2	1,10 Kč
Regulátor MC33269	11,88 Kč	1	11,88 Kč
Pojistka (volitelná)	7,65 Kč	1	7,65 Kč
Celková cena			**21,42 Kč**

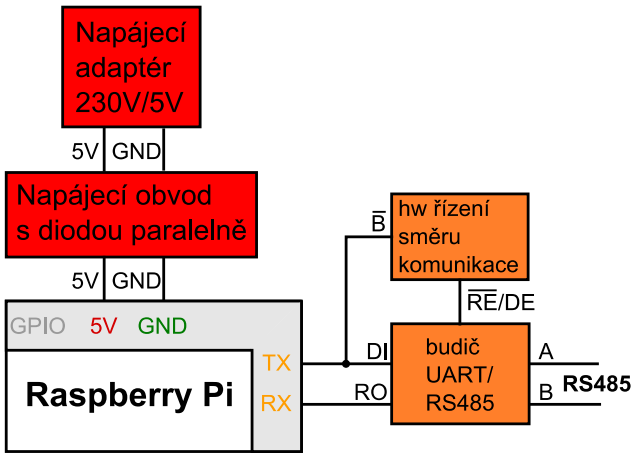
Tabulka 9.1: Vyčíslení ceny napájecího obvodu

Nyní je možné rozvádět napětí 5 V a zem ve dvou vodičích souběžně se dvěma kroucenými vodiči sběrnice RS485.

9.2 Zapojení centrální jednotky

Centrální jednotka obsahuje miniaturní počítač Raspberry Pi, napájecí obvod s paralelně zapojenou diodou popsaný výše a budič UART/RS485 s pomocným obvodem k přepínání směru komunikace. Raspberry Pi je napájeno 5 V z napájecího obvodu přes GPIO piny a ne přes microUSB konektor (viz obr. 3.1 a popis Power), jak bylo výrobcem zamýšleno. Z GPIO patice jsou celkem využity čtyři piny, a to 5 V Power, Ground, GPIO14_TXD a GPIO15_RXD, viz obr. 3.2. Napájecí adaptér z 230 V na 5 V a 2 A je připojen k desce s centrální jednotkou, ale

z principu může být umístěn kdekoli (i mimo desku satelitní jednotky). Rozvod napájení 3,3 V do budiče UART/RS485 a obvodu pro řízení směru komunikace není vyznačeno.



Obr. 9.3: Zapojení centrální jednotky s napájením 230 V

Výčet ceny centrální jednotky je níže. Tabulka je rozdělena na dvě části, v první části se nacházejí součástky nutné ke splnění funkčnosti centrální jednotky, v druhé části jsou volitelné součástky (např. univerzální plošný spoj nebo distanční sloupky), které mohou být zastoupeny jinak či nemusí být přítomny vůbec. Do tabulek nejsou zahrnuty rezistory a kondenzátory.

Součástky nezbytné	**Cena za kus**	**Počet**	**Cena**
Raspberry Pi, mod. B	853,69 Kč	1	853,69 Kč
Napájecí obvod	21,42 Kč	1	21,42 Kč
CD74HCT4358	5,73 Kč	1	5,73 Kč
ADM3485	55,05 Kč	1	55,05 Kč
Dílčí cena			**935,89 Kč**

Součástky volitelné	**Cena za kus**	**Počet**	**Cena**
Samořezný konektor	5,60 Kč	2	11,20 Kč
Plochý kabel, vícežilový	10,00 Kč	1	10,00 Kč
WAGO svorky	22,95 Kč	2	45,90 Kč
Univerzální plošný spoj	110,56 Kč	1	110,56 Kč
Distanční sloupky	5,27 Kč	4	21,08 Kč
Dílčí cena			**198,74 Kč**

Celková cena použitých součástek[9]	**1134,63 Kč**

Senzor	**Cena**
Raspberry Pi Camera Board	557,93 Kč

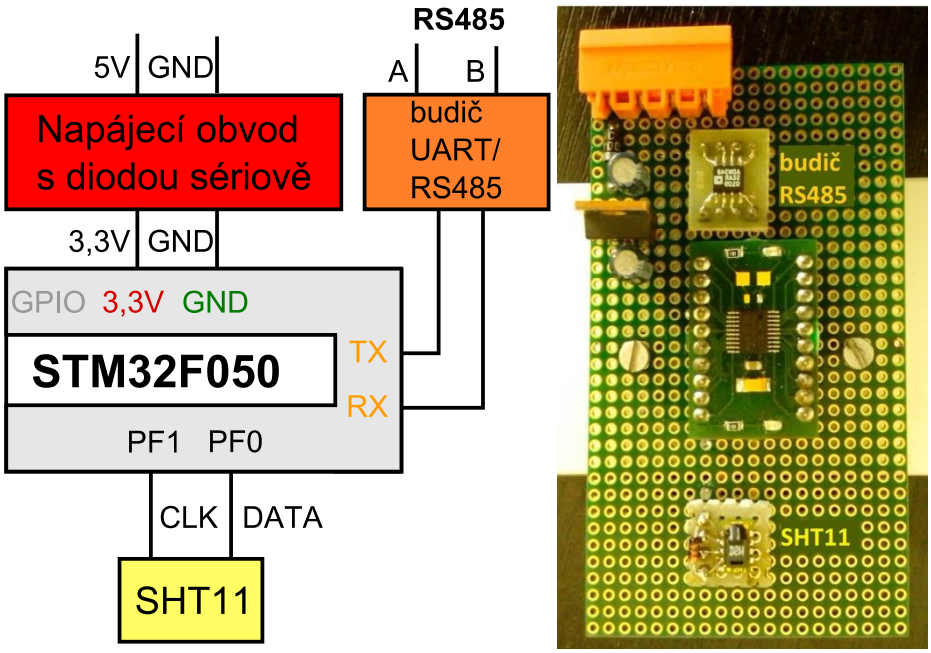
Tabulka 9.2: Vyčíslení ceny centrální jednotky

Do plošného spoje s centrální jednotkou je také zavedeno napájení ze spínaného zdroje SYS1308-2412 (229,68 Kč) přes odpovídající napájecí konektor (4,90 Kč), tím se cena desky s centrální jednotkou zvyšuje o 234,58 Kč. Cena kamery není započítána do ceny centrální jednotky.

[9] Cena bez zdroje napětí, viz vysvětlení pod tabulkou

9.3 Zapojení satelitní jednotky

Satelitní jednotka obsahuje mikrokontrolér STM32F050, napájecí obvod s paralelně nebo sériově zapojenou diodou dle potřeb senzorů, budič UART/RS485 a jeden nebo více senzorů připojených přes různá rozhraní. Ve schématu níže je nakreslené zapojení satelitní jednotky s teplotním a vlhkostním senzorem SHT11.



Obr. 9.4: Schéma zapojení satelitní jednotky se senzorem SHT11 a realizace

Konečnou cenu satelitní jednotky výrazně ovlivňuje použitý senzor. Do levné kategorie senzorů patří fototranzistory, které snímají intenzitu okolního světla nebo paprsek emitovaný diodou v optické závoře. Do dražší kategorie patří senzory plynů a inteligentní senzor teploty a relativní vlhkosti. Tabulka je rozdělena na tři části – součástky nezbytné, součástky volitelné a senzory.

Součástky nezbytné	Cena za kus	Počet	Cena
STM32F050	56,84 Kč	1	56,84 Kč
Napájecí obvod	21,42 Kč	1	21,42 Kč
ADM3485	55,05 Kč	1	55,05 Kč
Dílčí cena			**133,31 Kč**

Součástky volitelné	Cena za kus	Počet	Cena
WAGO svorky	22,95 Kč	2	45,90 Kč
Univerzální plošný spoj	60,00 Kč	1	60,00 Kč
Distanční sloupky	5,27 Kč	4	21,08 Kč
Dílčí cena			**126,98 Kč**

Celková cena použitých součástek	**260,29 Kč**

Senzor	Cena
Fototranzistor PT204	6,71 Kč
Inteligentní senzor STH11	305,50 Kč
TGS2600 – senzor kvality vzduchu	365,97 Kč
TGS4161 – senzor CO_2	775,00 Kč

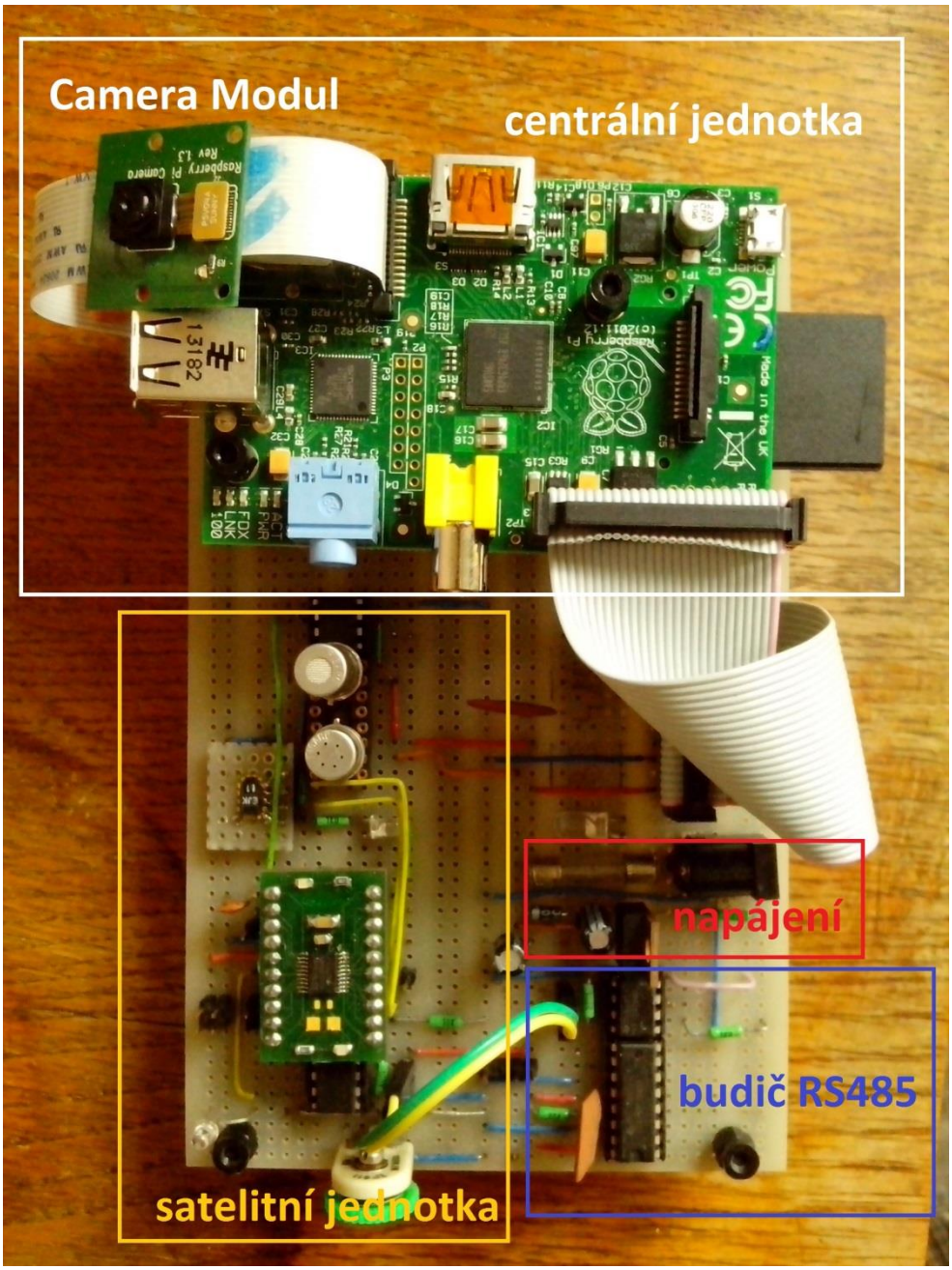
Tabulka 9.3: Vyčíslení ceny satelitní jednotky

9.4 První demonstrační deska

První demonstrační deska se skládá z jedné centrální jednotky a z jedné satelitní jednotky umístěné na desce univerzálního plošného spoje. Na jedné desce je tak Raspberry Pi, mikrokontrolér se senzory a napájecí obvod s paralelně zapojenou diodou. Univerzální plošný spoj je umístěn na distančních sloupcích, aby nedošlo k poškození spojů ze spodní strany desky. Raspberry Pi je také umístěno na distančních sloupcích nad deskou. GPIO piny Raspberry Pi na patici P1 jsou vyvedeny na desku plošného spoje dvaceti šesti žilovým plochým kabelem, viz obr. 9.5.

Demonstrační deska je fyzicky rozdělena do čtyř částí:

- Vpravo dole se nachází budič UART/RS485 spolu s pomocným obvodem pro řízení směru komunikace**.**
- Vpravo uprostřed je napájecí obvod tvořený napěťovým regulátorem, trubičkovou pojistkou, paralelně zapojenou křemíkovou diodou a konektorem pro připojení adaptéru.
- Plocha vlevo dole je vyhrazena mikrokontroléru STM32F050 a senzorům. Na tuto desku jsou připojeny vždy po jednom senzory teploty, vlhkosti, kvality vzduchu, koncentrace CO_2, intenzity osvětlení a potenciometr a optická závora.
- Horní část desky zabírá Raspberry Pi.

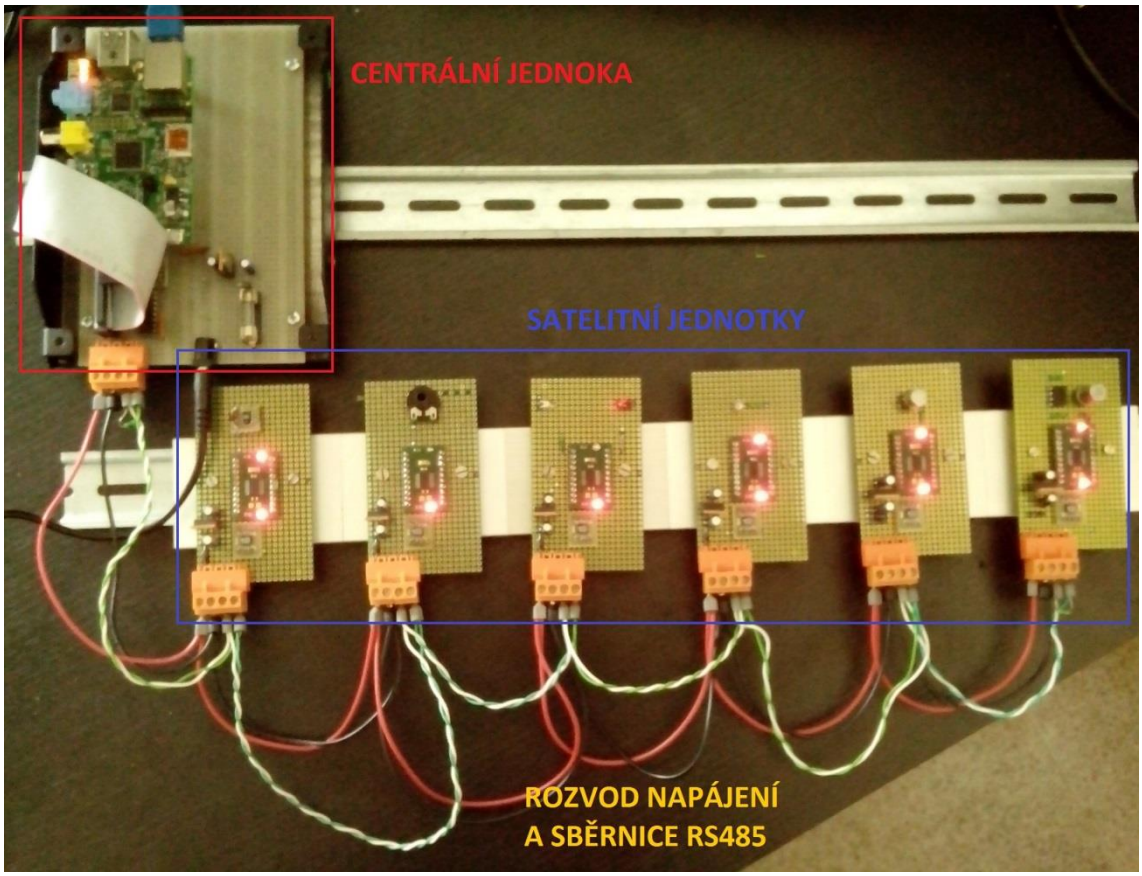


Obr. 9.5: První demonstrační deska systému sběru dat

9.5 Druhá demonstrační deska

Druhá deska, resp. desky byla vytvořena tak, aby ukázalo schopnost systému sběru dat být rozprostřen v prostoru. Toho je dosaženo tak, že celkem bylo vytvořeno sedm desek, na které je umístěna jedna centrální jednotka a šest satelitních jednotek se senzory.

Konstrukční řešení je zde oproti první demonstrační desce mírně odlišné. Jednotlivé moduly jsou připájeny do univerzálních plošných spojů, které jsou připevněny na DIN lištu. Sběrnice RS485 a napájení (5 V a zem) jsou z plošného spoje vyvedeny WAGO svorkovnicí.



Obr. 9.6: Druhá demonstrační deska/desky systému sběru dat

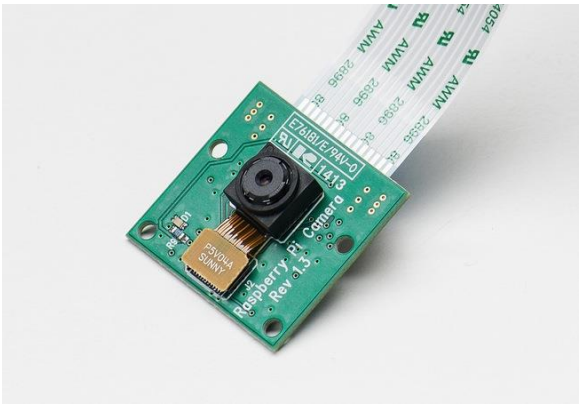
10 Zpracování obrazu na Raspberry Pi

V úvodu je naznačeno, že v této práci budou prozkoumány různé způsoby, jak zjistit přítomnost osob v budovách. Některé z metod již vysvětlených nejsou přímočaré (např. měření koncentrace CO_2 v budově popsané v 5.4) a některé způsoby detekce přítomnosti osob dokáží poskytnout pouze omezené informace (viz optická závora v 5.6). V této kapitole je navrhnuto řešení, jak získat komplexní informace o pohybu a počtu osob ve sledované oblasti za pomoci Raspberry Pi, ke kterému je připojen CMOS obrazový senzor Raspberry Pi Camera Board.

V tomto případě je do systému sběru dat zapojeno další Raspberry Pi, ke kterému je připojen obrazový senzor. Toto Raspberry Pi je na pozici satelitní jednotky a s centrální jednotkou (což je první Raspberry Pi) komunikuje po sběrnici RS485 a na vyzvání zasílá data o pohybu v místnosti.

Na začátku této kapitoly je představen obrazový senzor, který je použit k získání komplexní informace, postup jeho zprovoznění na RPi a příprava prostředí pro implementaci software pro zpracování obrazu. Posléze je popsán princip algoritmů, které detekují pohyb metodami modelování pozadí (background subtraction). Ke konci kapitoly je věnován algoritmu Codebook [29], který je implementován na Raspberry Pi s několika optimalizacemi, jež mají zaručit plynulejší zpracování obrazu. Závěr kapitoly zhodnocuje algoritmus Codebook na RPi.

10.1 Zapojení Raspberry Pi Camera Board



Obr. 10.1 : Obrazový senzor Raspberry Pi Camera Board

Raspberry Pi Camera Board je obrazový senzor dodávaný společností Raspberry Pi Fundation. Camera Board se skládá z CMOS senzoru OmniVision OV5647 o velikosti 3,67 × 2,74 mm, který dokáže snímat obraz v rozlišení 2592 x 1944 pixelů a vytvářet video v kvalitě 1080p30 (v rozlišení 1920 x 1080 pixelů při frekvenci 30 snímků za sekundu). Deska se senzorem se k RPi

připojuje patnáctižilovým plochým kabelem do konektoru CSI, viz popisek „CSI Connector Camera“ v obr. 3.1. Rozhraní CSI je specifikace **M**obile **I**ndustry **P**rocessor **I**nterface (MIPI) Aliance a specifikuje způsob komunikace mezi kamerou a procesorem v mobilních zařízeních.

Parametr	**Hodnota**
Senzor	OmniVision OV5647
Velikost senzoru	3,67 x 2,74 mm
Počet pixelů	2592 x 1944
Velikost pixelu	1,4 x 1,4 um
Zaostření objektivu	1 m až nekonečno
Rozlišení/FPS	FullHD/30, VGA/90
Velikost desky (bez kabelu)	25 x 24 mm

Tabulka 10.1: Parametry Rasbperry Pi Camera Board

10.2 Spuštění ovladačů Raspberry Pi Camera Board

Po připojení kamery do rozhraní CSI musí být do OS Raspbian nainstalovány příslušné ovladače. Ovladače jsou naštěstí součástí repozitáře systému Raspbian, a tak stačí aktualizovat Raspbian (doporučené) a zavést ovladače do kernelu systému příkazy:

```
>> sudo rpi-update
>> sudo modprobe bcm2835-v4l2
```

Po restartování systému můžeme vyzkoušet funkčnost kamery přímo z terminálu. Pokud chceme pořídit video záznam, následujícím příkazem nastavíme kameru pro video záznam v rozlišení Full HD.

```
>> v4l2-ctl --set-fmt-video=width=1920,height=1080,pixelformat=4
```

Samotné vytvoření video sekvence se spustí příkazem

```
>> v4l2-ctl --stream-mmap=3 --stream-count=100 --stream-to=video
```

kde parametr „stream-count“ značí počet snímků, které se mají zachytit, a „stream-to“ je název souboru s videem, který bude vytvořen.

Příkaz

```
>> v4l2-ctl –list
```

vypíše veškerá možná nastavení kamery. Jedno z nastavení kamery je parametr „auto_exposure“, který povoluje nebo zakazuje automatickou úpravu expozice – automatická expozice ovlivňuje detekci popředí tím, že může změnit během několika snímků barevné složky RGB vektorů většiny pixelů a tím zkreslit informaci o snímaném prostředí.

10.3 Instalace knihovny OpenCV

OpenCV [30] (Open **C**omputer **V**ision) je otevřená multiplatformní knihovna zaměřená na počítačové vidění a zpracování digitálního obrazu v reálném čase. OpenCV je napsána v jazyce C++ a poskytuje vývojáři mnoho prostředků pro zpracování obrazu, ať už se jedná o definici proměnné typu Mat, do které je možné uložit obraz, nebo o implementované optimalizované algoritmy, které dokáží provádět morfologické operace, segmentovat obraz a např. sledovat předmět dle barvy nebo tvaru. V této práci je tato knihovna použita k získání obrazu z kamery a jeho uložení do proměnné Mat, se kterou je možné dále v programu pracovat.

Po aktualizování repozitáře je postup instalace knihovny OpenCV je následující. Nejdříve nainstalujeme potřebné balíky dvěma příkazy.

```
>> sudo apt-get -y install build-essential cmake cmake-curses-gui pkg-config
libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev zlib1g-dbg zlib1g
zlib1g-dev pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools
libeigen3-dev
```

```
>> sudo apt-get -y install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-progs
ffmpeg libavcodec-dev libavcodec53 libavformat53 libavformat-dev
libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev libxine1-ffmpeg
libxine-dev libxine1-bin libunicap2 libunicap2-dev swig libv4l-0 libv4l-dev
python-numpy libpython2.6 python-dev python2.6-dev libgtk2.0-dev
```

Zdrojové kódy OpenCV knihovny stáhneme nástrojem `wget`.

```
>> wget -O openCV-2.4.9.zip
http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.9/opencv-
2.4.9.zip/download
```

Po dokončení stahování soubor rozbalíme, sestoupíme dovnitř, vytvoříme soubor „release“ a také do něj sestoupíme.

```
>> unzip openCV-2.4.9.zip
>> cd openCV-2.4.9
>> mkdir release
>> cd release
```

Stažený soubor chceme kompilovat pro Rasbian, nejprve musíme vytvořit konfiguraci OpenCV:

```
>> cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D
BUILD_ZLIB=ON -D WITH_V4L=ON -D WITH_GSTREAMER=ON -D WITH_OPENEXR=ON -D
WITH_UNICAP=ON -D BUILD_PYTHON_SUPPORT=OFF -D INSTALL_C_EXAMPLES=OFF -D
INSTALL_PYTHON_EXAMPLES=OFF -D BUILD_EXAMPLES=OFF ..
```

Program zkompilujeme a nainstalujeme do systému příkazy.

```
>> make
>> sudo make install
```

Nyní můžeme používat knihovnu OpenCV ke zpracování obrazu.

10.4 Úvod do detekce pohybu modelováním pozadí

V počítačovém vidění je modelování pozadí jednou z metod extrahování popředí v sekvenci snímků, jinými slovy se používá k určení pohybu. Další z metod je např. sledování optického toku v obraze, tato metoda poskytuje více informací o pohybu v obraze oproti modelování pozadí, ale mezi její výrazné nevýhody patří velká výpočetní náročnost [31].

Princip algoritmů modelujících pozadí je založen na přístupu, že provedeme-li rozdíl pozadí (snímku bez pohybujících se objektů) a současného snímku (s pohybujícími se objekty), získáme popředí jako rozdíl pixelů, kdy hodnota rozdílu překročí stanovený práh, viz následující rovnice.

$$|sn\acute{i}mek_i - pozad\acute{i}_i| > pr\acute{a}h\,.$$

Zde se objeví hned první problém, které musejí algoritmy modelující pozadí řešit. Jak je možné získat snímek pozadí prostředí bez popředí? Další problémy, kterým algoritmy musí čelit, jsou

- změny osvětlení scény ať už během dne (postupující slunce, mraky na obloze) nebo při rozsvícení světla v místnosti,
- pohyb, který nepatří do popředí – oscilace kamery nebo pohyb listí a větví stromů,
- změny v geometrii pozadí – přidání objektu do scény (např. položení předmětu na stůl) nebo naopak odebrání objektu z pozadí.

Z těchto představených problémů je jasné, že naivní přístup modelování pozadí popsaný výše bude fungovat jen za velice specifických podmínek a bude velmi citlivý na zvolený práh.

Pokročilejší a výpočetně nepříliš náročný algoritmus (požadavek Raspberry Pi), jenž řeší některé z výše uvedených problémů (např. postupně modeluje pozadí, a tak reaguje na změny v pozadí), je Running Gaussian Average [32] (klouzavý Gaussovský průměr). V tomto algoritmu se pro každý pixel v každé iteraci (s každým novým snímkem) vypočítá střední hodnota μ a směrodatná odchylka σ, a poté se určí, zda hodnota tohoto pixelu odpovídá Gaussovské křivce normálního rozložení pravděpodobnosti (μ, σ). Předpokládá se, že parametry pixelů v pozadí se mění v rámci normálního rozdělení, a tak každý pixel, jenž překročí stanovený práh *k* dle rovnice

$$\frac{|I_t - \mu_t|}{\sigma_t} > k\,,$$

patří do popředí a při nesplnění této podmínky patří do pozadí. Střední hodnota μ_t a směrodatná odchylka σ_t se vypočítá na základě předchozího snímku (proto klouzavý průměr). Práh *k* je obvykle roven 2,5 [32]. Pokud je *k* větší než 2,5, algoritmus dokáže rychleji reagovat na měnící se pozadí, nevýhodou většího prahu *k* může být, že popředí rychleji mizí do pozadí.

Klouzavý průměr tohoto algoritmu také umožňuje nízkou paměťovou náročnost během zpracování obrazu.

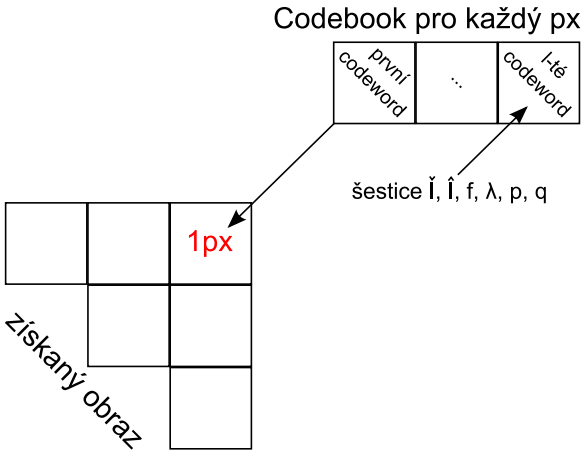
10.5 Modelování pozadí algoritmem Codebook

Dalším algoritmem, který modeluje pozadí, je Codebook [29], jenž je svými tvůrci prezentován jako rychlý algoritmus, který dokáže spolehlivě extrahovat popředí a správně klasifikovat předměty spadající do pozadí (pohybující se větve stromů, pomalé změny osvětlení scény apod.). Těchto vlastností je dosáhnuto tím, že pro každý pixel v obraze je vytvořena tzv. Codebook (řekněme seznam), do kterého se ukládají tzv. codeword, což jsou hodnoty charakterizující pixel.

Hodnoty, které charakterizují pixel, je šestice čísel $\{\check{I}, \hat{I}, f, \lambda, p, q\}$ a RGB vektor, které značí

- $\check{I}, \hat{I}$: minimální, respektive maximální hodnotu jasu, kterých pixel na daných souřadnicích kdy nabyl,
- *f:* frekvence, s jakou se pixel objevuje,
- *λ:* maximální negativní run-length[10] definovaný jako doba, po kterou se pixel neobjevil,
- *p, q:* první, respektive poslední čas objevení pixelu,
- *RGB:* reprezentuje zastoupení tří barev (červená, zelená a modrá) v pixelu. Jejich promíchání tvoří výslednou barvu pixelu, RGB vektor se neukládá do codeword.

Algoritmus poté funguje následovně. Řekneme, že $X = \{x_1, x_2, \ldots, x_N\}$ je trénovací sekvence pro každý pixel a skládá se z N RGB vektorů. Řekněme, že $C = \{c_1, c_2, \ldots, c_L\}$ reprezentuje Codebook pro každý pixel a je tvořen L codeword (pro každý pixel může být Codebook různě dlouhý).



Obr. 10.2: Tvorba struktury Codebook pro každý pixel

Popis algoritmu zobrazuje snímek 10.3 vyňatý z [29]. Zařazení nového pixelu do pozadí nebo do popředí je rozhodnuto na základě podmínky (a), což je barevná vzdálenost, a podmínky (b), která omezuje jas pixelu. Tyto dvě podmínky jsou vyhodnoceny pro každé codeword v Codebook pixelu. Pokud zkoumaný pixel nesplňuje tyto dvě podmínky, tj. není-li v Codebook, znamená to, patří do popředí a je přidán do Codebook.

[10] Run-length se používá např. v bezeztrátové kompresi, kde jsou vstupní data kódována do dvojic znaků, kde druhý z dvojice odpovídá kódovanému znaku a první z dvojice je číslo, které říká, kolikrát v řadě se znak objevil ve vstupní posloupnosti.

Odpovídají-li charakteristiky zkoumaného pixelu některému codeword (slovo) v Codebook, znamená to, že už dříve byl vyhodnocen jako popředí obrazu a nyní patří do pozadí.

Na konci průchodu všech pixelů obrazu se provede filtrace všech codeword dle prahu *λ,* který je stanoven dle potřeb aplikace – čím větší práh *λ,* tím se zvyšuje schopnost algoritmu nedekovat periodický pohyb v pozadí jako popředí, ale snižuje se schopnost algoritmu reagovat na trvalé změny v pozadí jako je odebrání předmětu apod.

Algorithm for Codebook Construction

I. $L \leftarrow 0$ ($\leftarrow$ means assignment), $\mathcal{C} \leftarrow \emptyset$ (empty set)

II. **for** t=1 to N **do**

i. $\mathbf{x}_t = (R, G, B)$, $I \leftarrow R + G + B$

ii. Find the codeword $\mathbf{c}_m$ in $\mathcal{C} = \{\mathbf{c}_i | 1 \leq i \leq L\}$ matching to $\mathbf{x}_t$ based on two conditions (a) and (b).

(a) $colordist(\mathbf{x}_t, \mathbf{v}_m) \leq \epsilon_1$

(b) $brightness(I, \langle \check{I}_m, \hat{I}_m \rangle) =$ **true**

iii. If $\mathcal{C} = \emptyset$ or there is no match, then $L \leftarrow L + 1$. Create a new codeword $\mathbf{c}_L$ by setting

- $\mathbf{v}_L \leftarrow (R, G, B)$
- $\mathbf{aux}_L \leftarrow \langle I, I, 1, t-1, t, t \rangle$.

iv. Otherwise, update the matched codeword $\mathbf{c}_m$, consisting of $\mathbf{v}_m = (\bar{R}_m, \bar{G}_m, \bar{B}_m)$ and $\mathbf{aux}_m = \langle \check{I}_m, \hat{I}_m, f_m, \lambda_m, p_m, q_m \rangle$, by setting

- $\mathbf{v}_m \leftarrow (\frac{f_m \bar{R}_m + R}{f_m + 1}, \frac{f_m \bar{G}_m + G}{f_m + 1}, \frac{f_m \bar{B}_m + B}{f_m + 1})$
- $\mathbf{aux}_m \leftarrow \langle\ min\{I, \check{I}_m\},\ max\{I, \hat{I}_m\},\ f_m + 1,\ max\{\lambda_m, t - q_m\},\ p_m,\ t\ \rangle$.

end for

III. For each codeword $\mathbf{c}_i$, $i = 1 \ldots L$, wrap around λ_i by setting $\lambda_i \leftarrow max\{\lambda_i, (N - q_i + p_i - 1)\}$.

Obr. 10.3: Codebook algoritmus

10.6 Realizace algoritmu Codebook

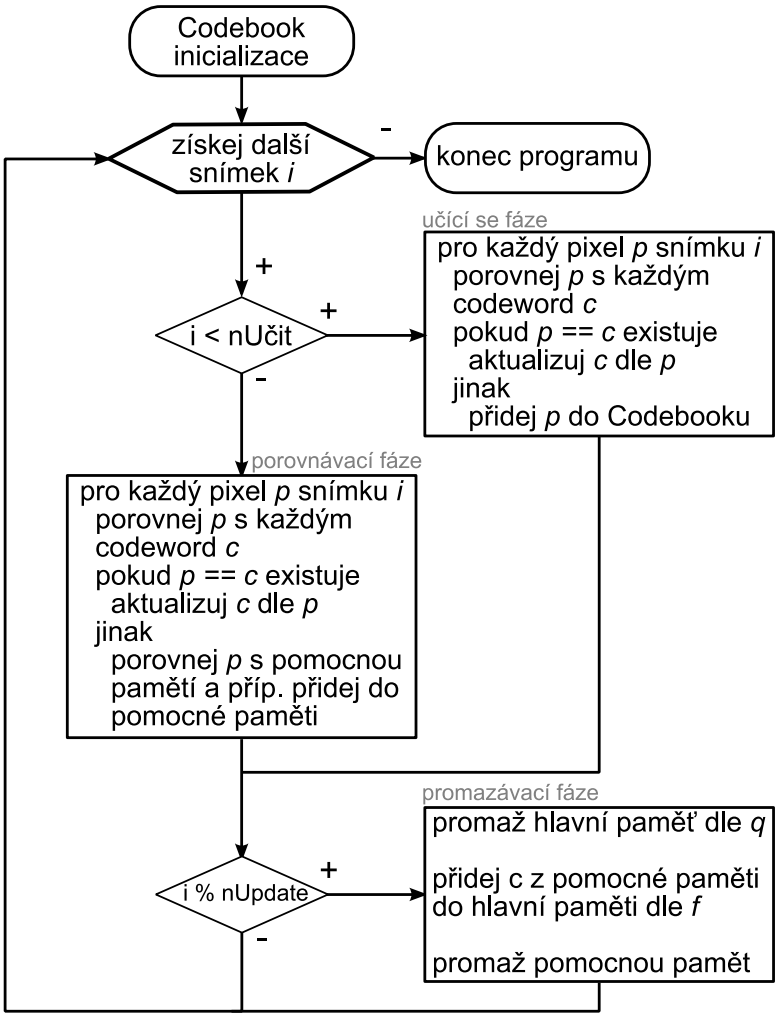
Na Raspberry Pi, které stojí jako satelitní jednotka, je implementován algoritmus Codebook popsaný v minulé kapitole s několika modifikacemi. Provedené modifikace mají zvýšit rychlost zpracování obrazu za cenu snížené spolehlivosti detekce popředí. Jmenovitě se jedná o

- obraz je popsán ve stupních šedi namísto trojrozměrných RGB vektorů,
- podmínka (a) o barevné vzdálenosti (obr. 10.3) je vynechána a je brán v potaz pouze jas pixelu (což je důsledek předchozího bodu)
- nová slova jsou přidána do Codebook po překročení jistého počtu výskytů. Důvodem je snaha, aby detekované popředí v další iteraci nepřešlo do pozadí. Dokud pixely nepřekročí jistý počet výskytů, jsou uloženy v pomocné paměti, která je vytvořena pro každý pixel a pracuje podobně jako hlavní paměť Codebook

- slova z Codebook jsou vyřazována na základě času posledního přístupu, nikoli dle prahu *λ*
- slova jsou mazána z pomocné i z hlavní paměti každý n-tý snímek, nikoliv každý

Průběh implementovaného upraveného algoritmu zobrazuje diagram 10.4, kde

- proměnná `i` značí číslo aktuálního snímku,
- proměnná `nUčit` určuje počet snímků, které jsou použity k úvodnímu modelování pozadí (`nUčit` může být rovno nule)
- proměnná `nUpdate` říká, kolikátý každý snímek se slova přesunou z pomocné do hlavní paměti (jinak řečeno se stanou pozadím). Spolu s tímto přesunem se promažou slova z hlavní i pomocné paměti.



Obr. 10.4: Modifikovaný algoritmus Codebook

Kromě detekce pohybu může algoritmus opatřovat pohybující se objekty v popředí jedinečným číselným identifikátorem (neboli ID), tuto možnost lze vypnout. Kód přidělující ID využívá OpenCV funkci k nalezení kontur v objektech detekovaných Codebook algoritmem. Objektu je přiděleno po vypočtení jeho středu unikátní ID a následně je objekt uložen do paměti. V příští iteraci kód opět spočte středy všech nalezených objektů a postupně naivně páruje objekty

z minulé iterace tak, aby vzdálenost středů objektů v současném a předchozím snímku byla minimální.

Obr. 10.5: Pohybující se člověk extrahovaný algoritmem Codebook

Snímek výše zachycuje extrahování popředí modifikovaným algoritmem Codebook. Vlevo je původní scéna, ve které se pohybuje člověk, jenž byl označen jedinečným číslem a orámován bílým obdélníkem. Vpravo je ten samý snímek, jenž je výstupem algoritmu – popředí ve snímku je vykresleno bíle, pozadí černě. Spodní část extrahovaného popředí tvarově neodpovídá nohám jdoucího člověka, protože do detekovaného pohybu patří také stíny vrhané pohybující se osobou.

10.7 Vyhodnocení algoritmu Codebook na Raspberry Pi

Tato podkapitola shrnuje to, nakolik může Raspberry Pi s obrazovým senzorem sloužit jako satelitní jednotka v systému sběru dat poskytující komplexní informace o pohybu osob a předmětů ve sledovaném prostoru.

Na Raspberry Pi byl spuštěn algoritmus Codebook popsaný v předchozí kapitole. Algoritmus byl upraven tak, že náročnější výpočty byly zjednodušeny, popř. úplně vynechány s ohledem na nižší výpočetní výkon Raspberry Pi (RPi disponuje jednojádrovým ARM procesorem, který tiká na frekvenci 700 MHz. CPU je možné přetaktovat na 1 GHz).

Za referenční algoritmus extrahování popředí byl zvolen Mixture of Gaussians [33] (MoG2)[11], který je implementován v OpenCV knihovně. Codebook algoritmus byl vyhodnocen na dvou počítačích, první je zmíněný Raspberry Pi, druhý počítač je osazen dvoujádrovým procesorem Intel Core 2 Duo na frekvenci 2,4 GHz. Obě zařízení zpracovávaly ten samý video soubor desetkrát za sebou. Výsledné FPS v tabulce níže je rovno aritmetickému průměru jednotlivých měření. V druhé fázi vyhodnocování algoritmu Codebook je popředí detekováno u videa s nižším rozlišením (první má rozlišení 848 pixelů na 480, druhé má 296 pixelů na 200).

[11] Princip tohoto algoritmu je podobný jako u algoritmu popsaného v kapitole 10.4. Hlavní rozdíl spočívá v tom, že MoG2 používá k určení popředí větší počet Gaussovských křivek (zpravidla 3 až 5) místo jedné.

	rozlišení: 848 x 480			**rozlišení: 296 x 200**	
algoritmus	Intel Core	ARM 0.7 GHz	ARM 1 GHz	ARM 0.7 GHz	ARM 1 GHz
MoG2	46,4	1,5	2,3	8,5	12,7
Codebook	31,0	1,1	1,7	6,8	10,2

Tabulka 10.2: Porovnání rychlosti zpracování obrazu algoritmy MoG2 a Codebook

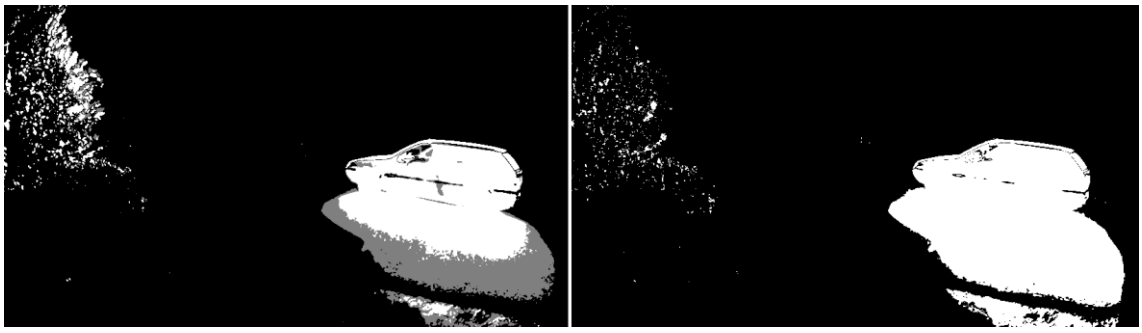
Z tabulky vyplývá, že algoritmus MoG2 pokaždé video zpracuje rychleji, nicméně skutečné popředí není extrahované celé (viz bílá plocha pohybujícího se auta v obr. níže obsahuje černé díry). Navíc algoritmus MoG2 do popředí zahrnuje objekty tam nepatřící – pohybující se větve (v levé části snímku). Kromě popředí dokáže algoritmus MoG2 poměrně úspěšně určit, která část popředí jsou stíny vrhané objektem (stíny jsou vyznačeny šedou barvou, popředí bílou).

Obr. 10.6: Auto a pohybující se větve stromů detekované algoritmem MoG2

Obr. 10.7 zachycuje ten samý obraz zpracovaný algoritmem Codebook. Je vidět, že detekovaný pohybující se vůz je „plnější“, tedy méně části automobilu bylo vyhodnoceno jako pozadí. Také je algoritmus odolnější proti pohybu větví stromů, nicméně i zde je prostor pro zlepšení.

Obr. 10.7: Pohyb auta zpracovaný algoritmem Codebook

Poslední obr. 10.8 zachycuje výsledky předchozích dvou algoritmů vedle sebe (algoritmus MoG2 je vlevo, Codebook vpravo).



Obr. 10.8: Porovnání algoritmu MoG2 (vlevo) a algoritmu Codebook (vpravo)

Z naměřených hodnot vyplývá, že výpočetní nároky algoritmu Codebook neodpovídají výpočetnímu výkonu Raspberry Pi. Výsledky mírně změnilo ve prospěch RPi výrazné snížení rozlišení zpracovávaného obrazu z 0,4 Mpx na 0,06 Mpx. Nevýhoda nižšího rozlišení obrazu ovšem je, že snižuje schopnost určení přesného počtu osob v místnosti, což je důležité např. z hlediska vhánění čerstvého vzduchu do místnosti systémem domovní automatizace. Objem vzduchu je odvozován od množství osob v uzavřeném prostoru.

Jedním z možných řešení je použití výkonnějšího počítače, jenž bude obraz zpracovávat, nicméně tento počítač také musí být schopen připojit se ke sběrnici RS485, která je v systému sběru dat použita k předávání dat mezi jednotkami.

Další řešení je použití méně výpočetně náročného algoritmu, který bude extrahovat popředí, jenže mnohé z této kategorie se potýkají s problémy, jako je změna geometrie pozadí, změny osvětlení v místnosti nebo detekování pohybu, který do popředí nepatří.

11 Závěr

Úkolem této práce, který byl vytyčen v prvních kapitolách, je vytvoření jednoduché a univerzální platformy sběru dat. Tento záměr byl naplněn – systém je otevřený jak z hlediska návrhu a realizace softwarového vybavení, tak z hlediska zapojení hardware. Tyto dvě podmínky zaručují, že systém může uživatel upravit a přizpůsobit vlastnímu použití.

Univerzálnosti systému také napomáhá vhodný výběr komponent, který byl proveden s ohledem na jejich dostupnost. Kupříkladu srdcem systému je miniaturní počítač Raspberry Pi, který je ve světě velmi populární[12] a mnohými lidmi už je používán např. jako domácí server. Na základě této práce budou moci své RPi použít jako základ systému domovní automatizace.

Dalším kritériem při výběru komponent byly jejich možnosti. Satelitní jednotky jsou osazeny 32-bitovými mikrokontroléry STM32F050 s jádrem Cortex-M0, které mají různorodé periferie (I^2C, SPI, AD převodník, PWM, GPIO piny), a tak je možné na tyto periferie připojit téměř libovolný senzor s analogovým nebo digitálním výstupem.

Navrhnutý systém sběru dat tvoří jedna centrální jednotka a mnoho satelitních modulů, které mohou být umístěny v monitorované budově na různých místech, kde mohou provádět požadované měření dat. V této práci mezi data měřená senzory patří fyzikální veličiny (teplota, relativní vlhkost, koncentrace oxidu uhličitého, kvalita vzduchu a intenzita osvětlení) a jiná data (zjišťování přítomnosti osob v místnosti pomocí optické závory a CMOS obrazového senzoru).

Obraz snímaný obrazovým senzorem je zpracován na Raspberry Pi algoritmem Codebook, který je zaměřen na extrahování popředí pomocí metody zvané modelování pozadí (background subtraction). Algoritmus je napsán s pomocí knihovny OpenCV, která je zaměřena na zpracování obrazu. V tomto algoritmu byly provedeny mírné optimalizace s ohledem na nižší výpočetní výkon Raspberry Pi (jednojádrový ARM, 700 MHz), které je v tomto případě na pozici satelitní jednotky komunikující s centrální jednotkou po sběrnici RS485. Nicméně se ukázalo, že Raspberry Pi není schopno zpracovávat obraz algoritmem Codebook v reálném čase a to ani při nižším rozlišení obrazového senzoru. Proto se nabízí otázka, zda nepoužít výkonnější verzi Raspberry Pi, která se objevila v únoru 2015 na trhu (nové RPi je osazeno čtyřjádrovým ARMv7 procesorem na frekvenci 900 MHz).

Dalším úkolem stanoveným v zadání práce bylo umožnit komunikaci jednotek v systému sběru dat na větší vzdálenosti prostřednictvím sběrnice RS485. Toho bylo dosáhnuto, navíc byl

[12] Zprávy z počátku roku 2015 uvádějí, že bylo prodáno přes pět milionů kusů.

implementován protokol Modbus, který zaručuje spolehlivé doručování zpráv na sedmé vrstvě referenčního modelu ISO/OSI za pomoci CRC součtu.

Posledním úkolem v zadání diplomové práce bylo umožnit vzdálený monitoring systému sběru dat prostřednictvím sítě Ethernet. Toho bylo dosáhnuto díky vlastnostem Raspberry Pi, které se v mnohém podobá plnohodnotnému počítači – po zprovoznění komunikace se satelitními jednotkami a vytvoření programu typu daemon, který data od satelitních jednotek získává a ukládá je do interní paměti zařízení, byly na Raspberry Pi vytvořeny skripty v programovacích jazycích Javascript a PHP. Tyto skripty jsou interpretovány webovým serverem Apache, a tak naměřená data mohou být vizualizována ve webovém prohlížeči každého počítače s připojením k Internetu.

12 Reference

[1] V. R. J. N. J. Haasz, Číslicové měřicí systémy, Praha: ČVUT, 2000.

[2] Modbus-IDA, „MODBUS over serial line specification and implementation guide V1.0," 2006. [Online]. Available: http://www.modbus.org/docs/Modbus_over_serial_line_V1.pdf.

[3] „Wikipedia.org," [Online]. Available: http://cs.wikipedia.org/wiki/Raspberry_Pi.

[4] J. Moorhead, „theguardian.com," 9 January 2012. [Online]. Available: http://www.theguardian.com/education/2012/jan/09/raspberry-pi-computer-revolutionise-computing-schools.

[5] B. Traynor, „Embedded Linux Wiki," 30 March 2015. [Online]. Available: http://elinux.org/RPi_Hardware.

[6] T. H. Alyssa Dayan, „AirPi," [Online]. Available: http://airpi.es/.

[7] M. Treacy, „Tree Hugger," 5 May 2013. [Online]. Available: http://www.treehugger.com/slideshows/gadgets/20-awesome-projects-raspberry-pi-microcomputers/.

[8] J. Adams, „Raspberry Pi," 7 April 2014. [Online]. Available: https://www.raspberrypi.org/raspberry-pi-compute-module-new-product/.

[9] „Wiring Pi," [Online]. Available: http://wiringpi.com/.

[10] M. McCauley, „C library for Broadcom BCM 2835 as used in Raspberry Pi," [Online]. Available: http://www.airspayce.com/mikem/bcm2835/.

[11] „Banana Pi," [Online]. Available: http://www.bananapi.org/.

[12] „Beagle Board," [Online]. Available: http://beagleboard.org/BLACK.

[13] „Odroid Platforms," [Online]. Available: http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140610189490.

[14] „Olimex,“ [Online]. Available: https://www.olimex.com/wiki/A10-OLinuXino-LIME.

[15] „RPi USB Wi-Fi Adapters,“ [Online]. Available: http://elinux.org/RPi_USB_Wi-Fi_Adapters.

[16] A. Čižmár, „MODUL STM32F050 v.1,“ ČVUT, Praha, 2013.

[17] STMicroelectronics, „STM32F050x4 STM32F050x6,“ STMicroelectronics, 2012.

[18] STMicroelectronics, „RM0091 Reference manual,“ STMicroelectronics, 2014.

[19] Ministerstva průmyslu a obchodu, „Vyhláška 194/2007 Sb.,“ Ministerstva průmyslu a obchodu, Praha, 2007.

[20] „wikipedie.org,“ 28 Únor 2015. [Online]. Available: http://cs.wikipedia.org/wiki/Vlhkost_vzduchu.

[21] Sensirion, „Datasheet SHT1x,“ Sensirion, 2010.

[22] Figaro, „figarosensors.com,“ 2005. [Online]. Available: http://www.figarosensor.com/products/2600pdf.pdf.

[23] H. Doležílková, „VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, FAKULTA STAVEBNÍ,“ [Online]. Available: http://www.fce.vutbr.cz/veda/JUNIORSTAV2007/pdf/Sekce_1.4/Dolezilova_Hana_CL.pdf.

[24] Figaro, „SOS electronic s.r.o.,“ [Online]. Available: http://www.sos.sk/a_info/resource/c/figaro/tgs4161.pdf.

[25] „libmodbus.org,“ [Online]. Available: http://libmodbus.org/.

[26] „Apache HTTP Server Project,“ [Online]. Available: http://httpd.apache.org/.

[27] „IAR Systems,“ [Online]. Available: https://www.iar.com/iar-embedded-workbench/downloads/.

[28] Sensirion, „www.sensirion.com,“ [Online]. Available: http://www.sensirion.com/nc/en/products/humidity-temperature/download-center/?cid=8575&did=103&sechash=a423b84d.

[29] T. H. C. D. H. L. D. Kyungnam Kim, „BACKGROUND MODELING AND SUBTRACTION BY CODEBOOK CONSTRUCTION,“ University of Maryland, College Park , 2004.

[30] O. S. Community, „OpenCV,“ [Online]. Available: http://opencv.org/.

[31] G. B. Adrian Kaehler, Learning OpenCV, O'Reilly Media, 2013.

[32] C. Wren, „Pfinder: real-time tracking of the human body,“ IEEE, Cambridge, 1997.

[33] W. G. Chris Stauffer, „Adaptive background mixture models for real-time tracking,“ MIT, Cambridge, Cambridge, 2000.

13 Přílohy

Obsah CD

Dokumentace – v této složce je uložena tato práce ve formátu PDF.

Programy – zde je uloženo softwarové vybavení Raspberry Pi a STM32F050. Obsahuje složky:

- **Codebook** – algoritmus Codebook napsaný v jazyce C++ s ukázkovými videi.
- **DAQ** – tento adresář je rozdělen do dvou částí. První část je program centrální jednotky (složka RPi). Druhá část je program satelitní jednotky (STM/stm32f0 - SATELIT).
  Spolu s programem satelitní jednotky je ve složce STM uložen zdrojový kód realizující Virtual Com Port na mikrokontroléru STM32F4 Discovery (STM/stm32f4 - USB VCP) s vysvětlujícím textem.

Reference – tento adresář obsahuje použité elektronické zdroje uvedené v kapitole 12.