UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2017 Tomáš Vodehnal

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Datalogger

Tomáš Vodehnal

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Vodehnal**
Osobní číslo:	**I14030**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Komunikační a mikroprocesorová technika**
Název tématu:	**Datalogger**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování:

Vytvořte datalogger pro sledování teploty a vlhkosti prostředí.
Systém bude sestávat z vhodného mikrokontroléru, příslušných čidel a případně z paměti a bude napájen z miniaturní baterie.
Nasnímaná data budou stahována do PC připojením dataloggeru přes USB, program pro PC bude zobrazovat nasnímaná data ve formě grafů a zajišťovat zálohováni dat v souborech.

Rozsah grafických prací:

Rozsah pracovní zprávy: **40 stran A4**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

MATOUŠEK, David. Aplikace mikrokontrolérů ATmega644. 1. vyd. Praha: BEN - technická literatura, 2013, ca 200 s. v různém stránkování. ISBN 978-80-7300-492-7.
VÁŇA, Vladimír. Mikrokontroléry ATMEL AVR: programování v jazyce C : popis a práce ve vývojovém prostředí CodeVisionAVR C. 1. vyd. Praha: BEN - technická literatura, 2003, 215 s. ISBN 80-730-0102-0.

Vedoucí bakalářské práce: **Ing. Bc. David Matoušek**
Katedra elektrotechniky

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Jan Pidanič, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2016

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 12. 05. 2017

podpis autora
Tomáš Vodehnal

PODĚKOVÁNÍ

Rád bych tímto poděkoval svému vedoucímu práce Ing. Davidu Matouškovi za věnovaný čas a trpělivost. Dále panu Jiřímu Sládkovi za poskytnutí pájecí stanice a dalších potřebných věcí pro praktickou výrobu. Davidovi Sládkovi děkuji za pomoc při praktické tvorbě výrobku. Mgr. Haně Vavříkové děkuji za pravopisnou kontrolu. Mgr. Kateřině Faltysové za překlad anotace do angličtiny.

ANOTACE

Práce se zabývá návrhem dataloggeru pro měření vlhkosti a teploty prostředí. Datalogger tyto hodnoty naměří a uloží do vnitřní paměti. Následně jsou data vyčtena a zobrazena formou grafů v aplikaci na počítači. Tato data je možné zálohovat v souborech.

KLÍČOVÁ SLOVA

měření teploty, měření vlhkosti, datalogger, ukládání dat, zpracování dat

TITLE

The Datalogger

ANNOTATION

The aim of the thesis is to design a datalogger measuring humidity and temperature of its environment. The datalogger measures and saves these values in the internal memory. The data are subsequently read and displayed in graphs in a computer application. The data can be backed up in files.

KEYWORDS

temperature measuring, humidity measuring, datalogger, store data, data processing

OBSAH

Úvod 12

1 Teoretická část 13

1.1 Datalogging 13

1.1.1 Princip fungování dataloggerů 13

1.2 Senzory 13

1.3 Vlhkost vzduchu 14

1.4 Vlhkoměry 16

1.5 Teplota 16

1.6 Teploměry 16

1.7 Datalogger hardware 17

1.7.1 Blokové schéma a funkce 18

1.8 Paměť 18

1.9 Výstup do PC a USB 19

1.10 Napájení z baterie 20

1.11 Datalogger software pro PC 20

1.12 Zálohování dat v souborech 20

2 Praktická část 22

2.1 Cíl práce 22

2.2 Výběr komponent 22

2.3 Sestavení hardwaru 23

2.3.1 Schéma zapojení 24

2.3.2 Deska plošného spoje 24

2.4 Popis funkce 25

2.5 Firmware 25

2.5.1 Knihovna pro senzor vlhkosti DHT11 25

2.5.2 Knihovna pro komunikaci s pamětí 27

2.5.3 Knihovna pro časovač a spánek 30
2.5.4 Knihovna pro UART 32
2.5.5 Způsob uložení dat v paměti 36
2.5.6 Hlavní smyčka 36
2.6 Aplikace pro PC 37
2.6.1 Struktura komunikace 37
2.6.2 Vykreslení grafu 39
2.6.3 Odeslání časové značky 40
2.6.4 Uložení dat 41
2.6.5 Načtení dat 42
2.6.6 Vzhled okna 43
2.7 Oživení 43
3 ZÁVĚR 46
4 Použitá literatura 47
5 Přílohy 49

SEZNAM ILUSTRACÍ A TABULEK

Obrázek 1 - princip fungování dataloggeru [2] 13
Obrázek 2 - Blokové schéma dataloggeru 18
Obrázek 3 - Blokové schéma převodníku CP2102 [11] 19
Obrázek 4 - Blokové schéma StepUp měniče [12] 20
Obrázek 5 - Formát uložení dat csv [14] 21
Obrázek 6 - StepUp měnič [19] 23
Obrázek 7 - Schéma zapojení 24
Obrázek 8 - Návrh PCB 25
Obrázek 9 - Komunikace DHT11 [15] 26
Obrázek 10 - čtení hodnoty sensoru vlhkosti 27
Obrázek 11 - Zápis do paměti [17] 28
Obrázek 12 - Čtení z paměti [17] 29
Obrázek 13 - Inkrementace paměťových proměnných 34
Obrázek 14 - Způsob uložení dat v paměti 36
Obrázek 15 - Struktura komunikace 38
Obrázek 16 - Pořadí přijatých dat 39
Obrázek 17 - vzhled okna aplikace 43
Obrázek 18 - Výsledný výrobek, zleva verze 2 a verze 1 44
Obrázek 19 - Ukázka naměřených dat 45

Tabulka 1 - Absolutní vlhkost při různých teplotách [5] 15

SEZNAM ZKRATEK A ZNAČEK

PC	Personal computer (osobní počítač)
DHT	Digital humidity temperature (označení senzoru vlhkosti a teploty)
SW	Software
HW	Hardware
CSV	Comma separated value (typ souboru)
IoT	Internet of things (internet věcí)
.NET	Platforma společnosti Microsoft pro vývoj aplikací
AVR	Rodina procesorů od firmy Atmel
RH	Relative humidity (relativní vlhkost)
USB	Universal serial bus (označení pro periferii počítače)
UART	Universal asynchronous reciever transmitter (označení pro sériovou linku)
SDA	Datový vodič I2C sběrnice
SCL	Hodinový vodič I2C sběrnice
I2C	sběrnice definovaná společností Philips
LSB	Least significant bit (bit s nejnižší vahou)
MSB	Most significant bit (bit s nejvyšší vahou)

ÚVOD

Práce se zabývá návrhem a konstrukcí zařízení, které nese název datalogger, včetně naprogramování uživatelské aplikace pro PC.

Teoretická část práce se zabývá vysvětlením pojmu datalogger a datalogging. Následně jsou zde rozebrány principy měření teploty a vlhkosti a také jejich využití v praxi, zejména v meteorologii. Dále se zde popisuje ukládání dat v paměti a komunikace s počítačem za pomoci sériové linky. V poslední části jsou popsány různé možnosti aplikací pro zobrazování naměřených dat v podobě grafů, jejich ukládání a archivace v souborech.

Praktická část práce nejprve popisuje samotný návrh hardwaru zařízení a výběrem komponent. Největší část je věnována programování mikrokontroleru ATtiny2313A v jazyce C. Jsou zde popsány knihovny pro dílčí komponenty dataloggeru, například knihovna pro senzor vlhkosti, paměť, sériovou linku a časovač. Z každé knihovny jsou popsány stěžejní funkce. Následně je popsána hlavní smyčka celého programu. Kromě programu pro mikrokontroler obsahuje praktická část taktéž vývoj aplikace pro PC. Zde je řešena především komunikace mezi hardwarovou a softwarovou částí práce. Zmíněna je také práce s daty a jejich vykreslení v podobě grafu a následné ukládání a zálohování. Poslední část obsahuje fotografie výrobku a práci na oživování.

Práci jsem si vybral z důvodu zájmu o měření neelektrických veličin a meteorologii.

1 TEORETICKÁ ČÁST

Teoretická část představuje výchozí poznatky před samotným praktickým návrhem práce.

1.1 Datalogging

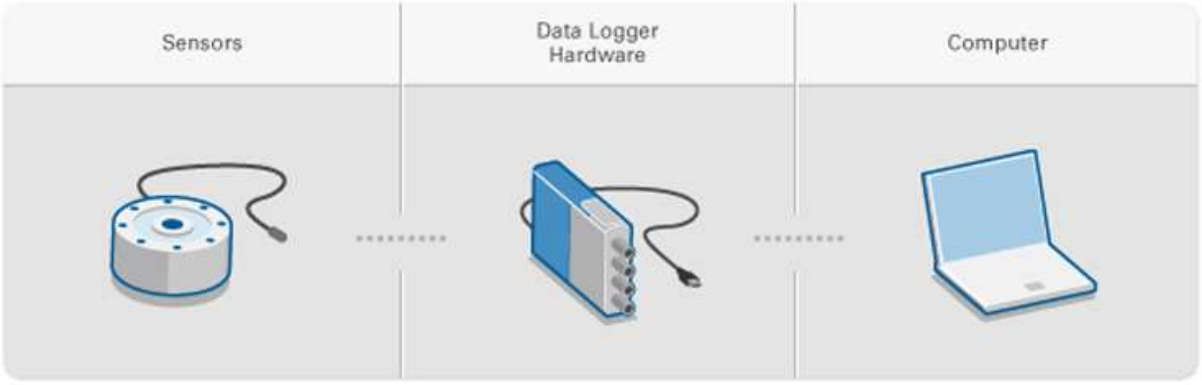
Datalogging je činnost, při které se měří fyzikální veličiny v určitých časových intervalech. Využívá se například ve:

- zdravotnictví
- dopravních prostředcích
- meteorologii

a jiné. Ve zdravotnictví má velké využití , například pro měření srdečního pulsu a životních funkcí člověka. V případě dopravních prostředků, například záznam dráhy a rychlosti jízdy, nebo u železniční dopravy se používá pro zapisování grafikonů. Meteorologie používá datalogging pro vytvoření srážkových map a následných animací. [1][2]

1.1.1 Princip fungování dataloggerů

Následující obrázek stručně ukazuje, jak probíhá proces získávání dat.



Obrázek 1 - princip fungování dataloggeru [2]

Měřená hodnota je snímána senzorem, který je připojen do dataloggeru. Datalogger vyčte měřenou hodnotu ze senzoru vždy, jednou za určenou časovou periodu. Následně Datalogger může data uložit do vnitřní paměti až do doby, kdy budou hodnoty odeslány do počítače ,nebo je může odeslat přímo do počítače k dalšímu zpracování. Toto záleží na typu dataloggeru, o kterých pojednává práce níže. [2]

1.2 Senzory

Senzory jsou součástky, které převádí neelektrickou veličinu na elektrickou. Ekvivalentem slova senzor je snímač. Podle vstupní neelektrické veličiny rozlišujeme senzory:

- teploty
- tlaku
- vlhkosti
- rychlosti proudění.

A jiné. Vzhledem k zadání práce jsou níže rozebrány pouze senzory teploty a vlhkosti.[3]

Senzory jsou rozděleny do dvou hlavních skupin a to:

- pasivní
- aktivní.

Pasivní snímače mění některou svou charakteristickou vlastnost (kapacita, odpor, indukčnost, …) v závislosti na měřené veličině (teplota, vlhkost, …). Pasivní snímače ke své funkci potřebují dodávat elektrickou energii. Například u termistoru, kde se mění odpor v závislosti na teplotě, se elektrická energie vyzařuje v podobě tepla.[3]

Aktivní snímače nepotřebují ke správnému fungování zdroj elektrické energie. Naopak v obvodu jsou samy tímto, byť slabým, zdrojem. Příkladem může být piezoměnič, který při mechanickém namáhání vytvoří na svých svorkách napětí.[4]

Podle výstupního elektrického signálu jsou senzory rozděleny na:

- analogové
- digitální.

Analogové snímače mají na výstupu hodnotu napětí nebo proudu, která odpovídá vstupní veličině. Hlavní výhodou těchto senzorů je spojitost, to znamená, že lze změřit kteroukoliv hodnotu vstupní veličiny.[3][4]

Digitální snímače již obsahují řídicí elektroniku, která převede analogovou hodnotu na digitální. Díky tomu dochází ke kvantizaci a nelze změřit libovolnou hodnotu vstupní veličiny. Výhoda digitálních snímačů spočívá převážně v jednoduchosti implementace. Datalogger pracuje už s digitální informací a není potřeba, aby analogovou informaci převáděl. Navíc je digitální hodnota lépe uchovatelná v paměti. [3][4]

1.3 Vlhkost vzduchu

Vlhkost určuje množství vodní páry v daném prostředí. Toto množství se dá určit různými veličinami. Jednotlivé veličiny jsou vysvětleny na vlhkosti vzduchu.[5]

Absolutní vlhkost vzduchu udává hmotnost vodních par v jednom metru krychlovém vzduchu. Značí se řeckým Φ.[5]

$$\Phi = \frac{m}{V} \quad (1)$$

m – hmotnost vodních par

V – objem vzduchu

Relativní vlhkost vzduchu udává poměr aktuálního nasycení vzduchu k maximálnímu možnému nasycení při dané teplotě. Jedná se o bezrozměrnou veličinu, častěji se ale vynásobí stem a udává se v procentech.[5]

$$\varphi = \frac{\Phi}{\Phi_{max}} 100\% \quad (2)$$

Φ – absolutní vlhkost

Φ_{max} – maximální vlhkost při dané teplotě (tabulková hodnota)

Vlhkost je závislá na teplotě. Obecně platí, že čím vyšší teplota prostředí, tím větší množství vody může pojmout, viz Tabulka 1. Údaje v tabulce jsou absolutní maximální vlhkost prostředí nad vodní, případně ledovou, plochou při určité teplotě. Z těchto údajů se poté vypočítává relativní vlhkost.[5]

Tabulka 1 - Absolutní vlhkost při různých teplotách [5]

Teplota [°C]	Hustota nasycených vodních par nad vodou [g/m^3]	Hustota nasycených vodních par nad ledem [g/m^3]
-40	0,18	0,12
-30	0,45	0,34
-20	1,07	0,88
-10	2,36	2,14
0	4,85	4,85
10	9,40	-
20	17,30	-
30	30,38	-
40	51,19	-

V meteorologii se často užívá údaj pocitová teplota. Tato hodnota silně závisí na vlhkosti. Vysoká pocitová teplota se projevuje jako dusno a bývá často v létě před bouří. Vysokou pocitovou teplotu člověk vnímá, když je ve vzduchu hodně vlhkosti. Kromě vlhkosti závisí také na rychlosti proudění vzduchu, obecně platí, že čím rychleji fouká vítr, tím nižší je pocitová teplota. [5]

1.4 Vlhkoměry

V minulosti se používalo zejména psychometrické měření vlhkosti. To spočívá v měření tzv. suché a mokré teploty. Suchá teplota se změří klasickým teploměrem a mokrá se změří teploměrem, který je ve spodní části omotán mokrým kusem látky. Z těchto dvou hodnot se poté z psychometrických tabulek odečte relativní vlhkost vzduchu.[6]

Další možností změření relativní vlhkosti je použití vlhkoměru. Ten využívá například vlastnosti lidského vlasu a jeho schopnosti pohlcovat vzdušnou vlhkost, díky čemuž mění svou délku.[6]

Dnes se nejčastěji používají elektronické vlhkoměry. Ty můžeme rozdělit do dvou hlavních kategorií

- rezistorové
- kapacitní

Principem kapacitních vlhkoměrů je změna vlastností dielektrika při pohlcování vzdušné vlhkosti. Mezi dvě elektrody je vloženo dielektrikum, které je schopné pohlcovat vlhkost. Toto dielektrikum změní svou relativní permitivitu a tím se změní i kapacita kondenzátoru úměrně k vlhkosti vzduchu, protože relativní vlhkost je závislá na teplotě prostředí. K určení relativní vlhkosti je také nutné měřit teplotu, proto senzory vlhkosti měří kromě vlhkosti také teplotu.[6]

Rezistorové vlhkoměry využívají změnu vodivosti některých materiálů na množství absorbované vody. Materiál se použije buď jako spirála namotaná na nevodivém tělísku, nebo se napařuje na keramický podklad. Velká výhoda tohoto řešení je vysoké rozlišení.[6]

1.5 Teplota

Teplota je nejvíce sledovaný údaj prostředí. Na teplotě závisí mnoho technologických i přírodních procesů. Teplotu okolí velmi dobře vnímá lidské tělo. Teplota je údaj, který popisuje teplotní stav prostředí. V meteorologii se používá pojem pocitová teplota, ta je ale vždy jiná než teplota, kterou lze změřit teploměrem, protože závisí na vlhkosti vzduchu, viz kapitola 1.3.[7]

1.6 Teploměry

Teplota se měří teploměrem. Teploměry můžeme rozdělit do dvou skupin:

- dilatační teploměry
- elektrické teploměry

Dilatační teploměry, jak už z názvu vyplývá, jsou založeny na principu změny určité fyzikální neelektrické veličiny. Například tlaku, objemu nebo délky. Elektrické teploměry fungují na principu změny elektrické veličiny, například odporu, napětí nebo se využívá termoelektrický jev. Vzhledem k zadání práce budou podrobněji popsány pouze elektrické teploměry.[7]

Měření elektrickým teploměrem obnáší měření jiné elektrické veličiny, která se mění v závislosti na teplotě. S touto veličinou se může přímo dále pracovat bez účasti mikrokontroleru analogově, nebo v případě použití mikrokontroleru je potřeba ji převést do binární podoby, za pomoci AD převodníku.[7]

Dnes je možné již zakoupit integrovaný obvod se zabudovaným AD převodníkem a protokolem pro komunikaci (oneWire, I2C, atd). Tyto součástky mají velké výhody v jednoduchosti implementace, protože na výstupu těchto obvodů už je přímo binární číslo, které odpovídá naměřené hodnotě. Další velkou výhodou je široký výběr a cena, která se stále snižuje.[7]

1.7 Datalogger hardware

Klasický datalogger je samostatné zařízení, které měří příslušné veličiny a ukládá je do vnitřní paměti. Tato data musí být později odeslána do počítače k následnému zpracování, dlouhodobému uložení a vizualizaci. Toto řešení má několik výhod:

- malá velikost
- napájení z baterie
- snadná implementace [8]

Dataloggery přímo spojené s počítačem již nejsou samostatné zařízení a ke své funkci potřebují připojení k počítači například pomocí sériové linky, ethernetu, bluetooth nebo wifi. Zde jsou některé výhody, které toto řešení nabízí:

- vizualizace v reálném čase
- připojení k síti
- velký úložný prostor
- možnost použití výkonného procesoru počítače
- okamžitá analýza dat.

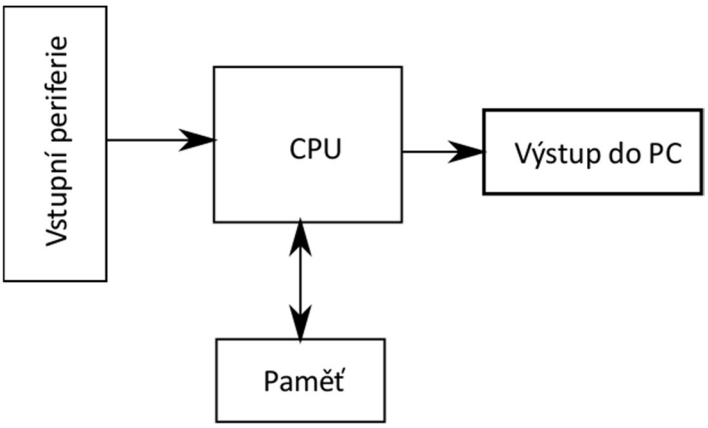
Nevýhodou je o něco složitější implementace. Již je potřeba zajistit napájení pro počítač a jeho propojení s dataloggerem.[8]

Hardware dataloggeru může obsahovat různé množství čidel různých druhů. V závislosti na počtu připojených senzorů dělíme dataloggery na:

- jednokanálové
- vícekanálové. [8]

1.7.1 Blokové schéma a funkce

Následující obrázek ukazuje obecné blokové schéma dataloggeru. Vstupními periferiemi se myslí senzory příslušných měřených veličin. Data ze senzorů zpracovává mikroprocesor a ukládá je do paměti. Při připojení k PC se data z paměti vyčtou a odešlou do PC. Obrázek 2 je mým vlastním dílem, nakresleným na základě znalostí získaných z webu společnosti National Instruments. [8]



Obrázek 2 - Blokové schéma dataloggeru

1.8 Paměť

Paměť slouží pro ukládání dat. Je součástí hardwaru dataloggeru. Obecně je paměť organizována jako matice buněk, do kterých lze ukládat vždy jeden bit. Paměti lze rozdělit následovně:

- ROM
- PROM
- EPROM
- EEPROM
- FLASH
- RAM.

Paměti ROM, PROM a EPROM jsou zařazeny spíše z historických důvodů. Dnes se používá EEPROM, FLASH nebo RAM.[9]

EEPROM znamená Electricaly Erasable Programable Read Only Memory, v překladu elektricky mazatelná programovatelná paměť pouze pro čtení. I přesto, že je v názvu této paměti pouze pro čtení, lze do této paměti zapisovat. Jedná se o statickou a energeticky nezávislou

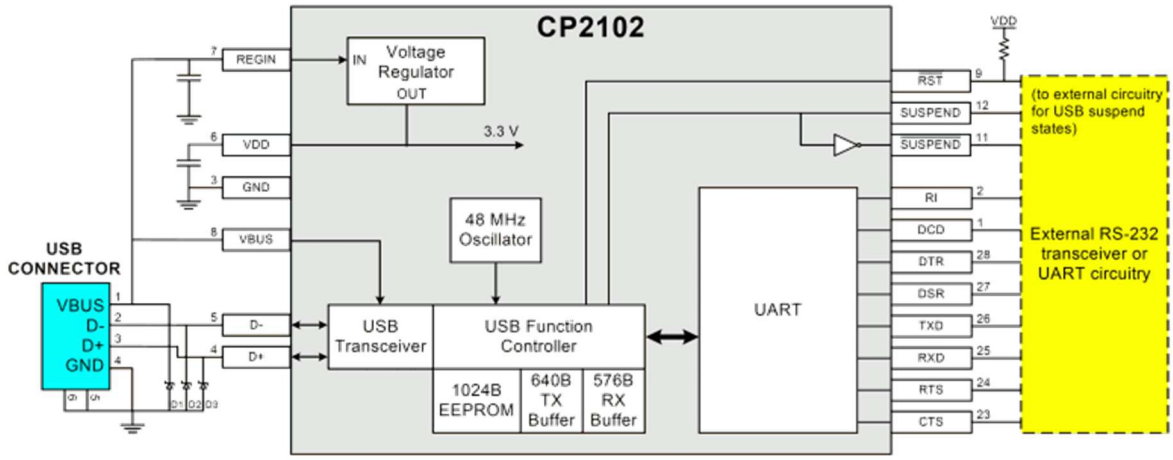
paměť. Statická znamená, že se data nemusí v paměti obnovovat. Energeticky nezávislá znamená, že udržuje data i po odpojení napájení.[9]

Paměti FLASH jsou podobné jako EEPROM. Na rozdíl od EEPROM pamětí probíhá mazání informace po celých blocích. Nikoli buňka po buňce.[9]

Paměti RAM jsou energeticky závislé, tudíž není vhodné je používat pro realizaci dataloggeru, protože při vybití baterie by došlo ke ztrátě dat.[9]

1.9 Výstup do PC a USB

Nejčastěji se periferní zařízení k PC připojují pomocí USB. Většina mikrokontrolerů ovšem nemá USB periferii. Ke komunikaci používá nejčastěji sériovou linku. Starší počítače jsou sériovou linkou vybaveny, toto ale dnes nepatří ke standardům. Vytváří se tedy virtuální porty sériové linky, které využívají USB sběrnici. Pro komunikaci po USB sběrnici se využívají speciální obvody převodníky USB – UART. Blokové schéma je uvedeno na Obrázek 3. [10]



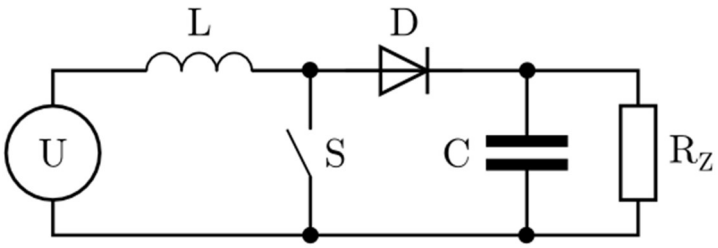
Obrázek 3 - Blokové schéma převodníku CP2102 [11]

Blokové schéma obsahuje 2 hlavní části:“USB function controller a UART.“ USB function controller má na starosti připravit data do formátu, ve kterém budou poslány přes USB rozhraní. Zároveň data, která přijdou přes USB rozhraní zpracuje a předá bloku UART. Blok UART zpracovává data, která přijdou po sériové lince a předává je USB controlleru. Zároveň zpracovává data od USB controlleru a odesílá je na sériovou linku. [10]

Obvod CP2102 je jedním z nejpoužívanějších převodníků USB UART. Obsahuje paměť EEPROM, do které je možné naprogramovat ID výrobce a produktu, krátký popis zařízení, popis napájení, sériové číslo a výrobní číslo výrobku.[10]

1.10 Napájení z baterie

V zadání práce stojí, že datalogger bude napájen miniaturní baterií. Knoflíkové baterie se vyrábí s různými napětími od 1 do 3V. Napájení mikrokontroleru a dalších součástek ale vyžaduje 5V. Ke zvýšení napětí se používají obvody, které se v anglické literatuře označují jako STEP-UP měniče. STEP-UP měniče se dnes používají téměř ve všech zařízeních napájených z baterie. Blokové schéma STEP-UP měniče je na Obrázek 4. [12]



Obrázek 4 - Blokové schéma StepUp měniče [12]

Spínač S je na reálných obvodech realizován pomocí tranzistoru v zapojení jako spínač. Obvod má 2 stavy. Spínač S je sepnutý nebo rozepnutý. V případě, že spínač je sepnutý, proud ze zdroje U prochází přes cívku L, kladná svorka cívky je nyní na levé straně. V cívce L se naakumuluje energie. V případě rozepnutého spínače se polarita na cívce změní a v obvodu se začne chovat jako zdroj. Nyní jsou tedy v obvodu 2 zdroje zapojené v sérii. Jeden zdroj U a druhý zdroj realizovaný cívkou L. Napětí na obou zdrojích se sečte a přes diodu se nabije kondenzátor C na vyšší hodnotu, než je samotné napětí U, tedy napětí baterie. V případě ideálních součástek závisí výstupní napětí na frekvenci spínání. [12]

1.11 Datalogger software pro PC

Software dataloggeru je ve své podstatě pouze zobrazovač dat. Některé umožňují i jejich editaci, přímé výpočty, zálohování, porovnávání, analýzu atd. Trh překypuje obrovským množstvím různých softwarů a to jak komerčních placených aplikací, tak i freeware aplikací. V poslední době s rozvojem IoT (Internet of Things) se softwary pro datalogging orientují tímto směrem. Jako příklad free online softwaru je server thingspeak.com. [13]

1.12 Zálohování dat v souborech

Pro ukládání velkého množství dat se velmi často využívá souborů s koncovkou CSV (comma separated value). Formát uložení dat je následující:

```
field_name,field_name,field_name CRLF
aaa,bbb,ccc CRLF
zzz,yyy,xxx CRLF
```

Obrázek 5 - Formát uložení dat csv [14]

Každá hodnota je oddělena čárkou nebo středníkem. Každý jeden záznam je na vlastním řádku. První řádek s názvy sloupce je volitelný. Poslední řádek již není odřádkován (CRLF). CRLF znamená odřádkování. Tyto soubory je možné pohodlně otevřít v programu Excell i jiných tabulkových procesorech. Platforma .NET má pro práci s těmito soubory vytvořené knihovny. [14]

2 PRAKTICKÁ ČÁST

Praktická část řeší výběr komponent a praktický návrh práce. Poměrně velkou část zaujímá popis firmwaru a softwaru, který byl v rámci práce vytvořen.

2.1 Cíl práce

Cílem je funkční datalogger spolu s aplikací pro PC, která zajistí zálohování a zobrazování dat. Pro tuto práci jsem se rozhodl, protože mě velmi zajímá měření neelektrických veličin a práce s mikrokontroléry AVR. Výhledově bych datalogger chtěl použít v malé domácí meteorologické stanici.

2.2 Výběr komponent

Hardwarové řešení práce není složité. Začal jsem výběrem senzoru. Našel jsem jednoduchý senzor vlhkosti DHT11, který se vyrábí jako modul pro arduino. Měřicí rozsah senzoru vlhkosti závisí na teplotě. Pro 25°C je rozsah 20% - 90%RH (relative humidity). Při měření relativní vlhkosti je nezbytně nutné měřit i teplotu. Měřicí rozsah pro teplotu u tohoto čidla je 0°C – 50°C. Pro použití venku není tento rozsah dostačující, ale tento senzor je plně kompatibilní se senzorem DHT22. Měřicí rozsah senzoru DHT22 je 0%RH – 100%RH a -40°C - +80°C. Další výhodou je ochranné pouzdro, které počítá s použitím ve vnějším prostředí. Do práce jsem tedy použil senzor DHT11, protože mi zbývají některé moduly ještě z dob, kdy jsem se zajímal o arduino. Později je možná záměna se senzorem DHT22. [15][16]

Paměť pro ukládání dat jsem vybíral zejména s ohledem na kapacitu paměti. Ze senzoru DHT11 je každé měření přijato 4B dat. V případě měření každých 15 minut to vychází na 384B dat denně. Vybral jsem paměť AT24C512C, která má kapacitu 512kB. Tato kapacita je více než dostatečná a je schopná uchovávat naměřené informace za 1333 dní. Výhodou je nízká spotřeba, i2c rozhraní a velmi vysoká kapacita. [17]

Baterii jsem vybíral zejména s ohledem na kapacitu a velikost, protože zadání požaduje miniaturní baterii. Vybral jsem baterii CR2032. Tato baterie má napětí 3V a kapacitu 220mAh. Vzhledem k tomu, že procesor je většinu času uspán v režimu snížené spotřeby, tak tato kapacita je dostačující. [18]

Protože senzor DHT11 pracuje při napětí od 3,3V, je nutné použití step-up měniče. Na českém obchodě jsem našel step-up měnič z 0,9V – 5V na 5V s výstupem USB, viz Obrázek 6. K měniči jsem nenašel konkrétní dokumentaci. Obchod uvádí výstupní proud 300mA, což je

více než dostatečné. S tímto měničem nemám dosud zkušenosti a vybral jsem ho z důvodu nízké ceny a snadné dostupnosti na českém obchodě. [19]

Obrázek 6 - StepUp měnič [19]

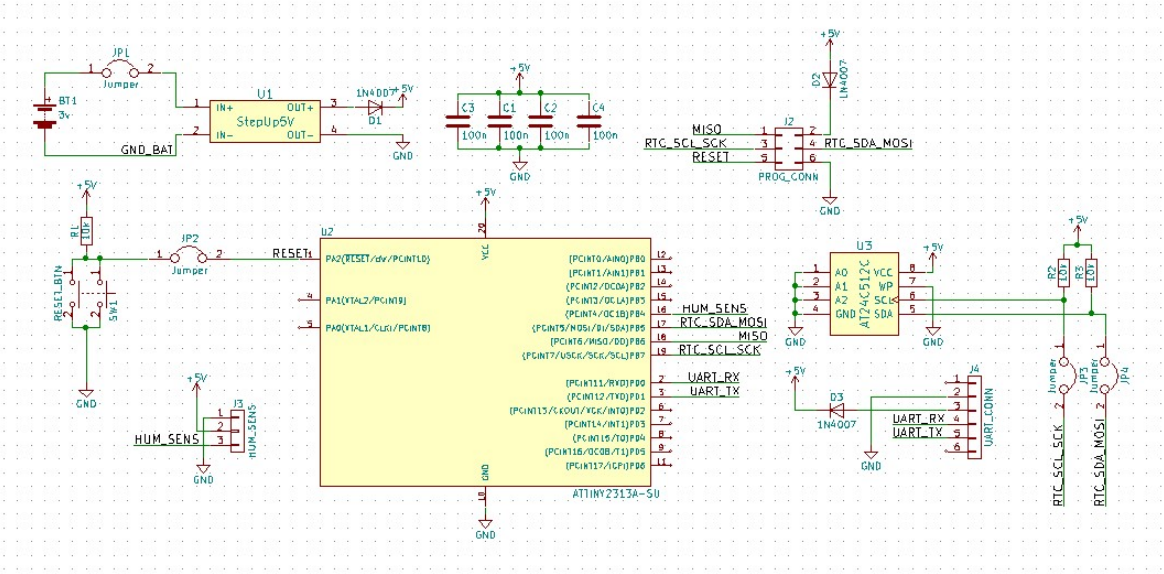
Pro komunikaci s PC jsem zvolil převodník USB UART CP2102, který lze sehnat již jako hotový modul se všemi potřebnými součástmi a USB konektorem. S tímto modulem mám již zkušenost a osvědčil se mi v mnoha různých aplikacích, proto jsem ho zvolil i do této práce. [10]

Klíčovým prvkem v obvodu je mikrokontroler. Vzhledem k napájení z baterie, jsem sáhl do rodiny procesorů ATtiny, které mají obecně nižší spotřebu. Požadavky na vybavenost periferií jsou UART, I2C rozhraní a jeden další pin pro komunikaci se senzorem. Procesor ATtiny 2313A tyto periferie obsahuje a má také dostatek pinů. Procesor jsem vybral z důvodu nízké spotřeby a ceny. [20]

2.3 Sestavení hardwaru

Hardware dataloggeru je poměrně jednoduchý. Většina komponent je realizována pomocí modulů. Tyto moduly se připojují pomocí pinových řad s roztečí 2,54mm. Pro návrh desky plošného spoje jsem použil program KiCAD, což je freeware program pro kreslení plošných spojů.

2.3.1 Schéma zapojení



Obrázek 7 - Schéma zapojení

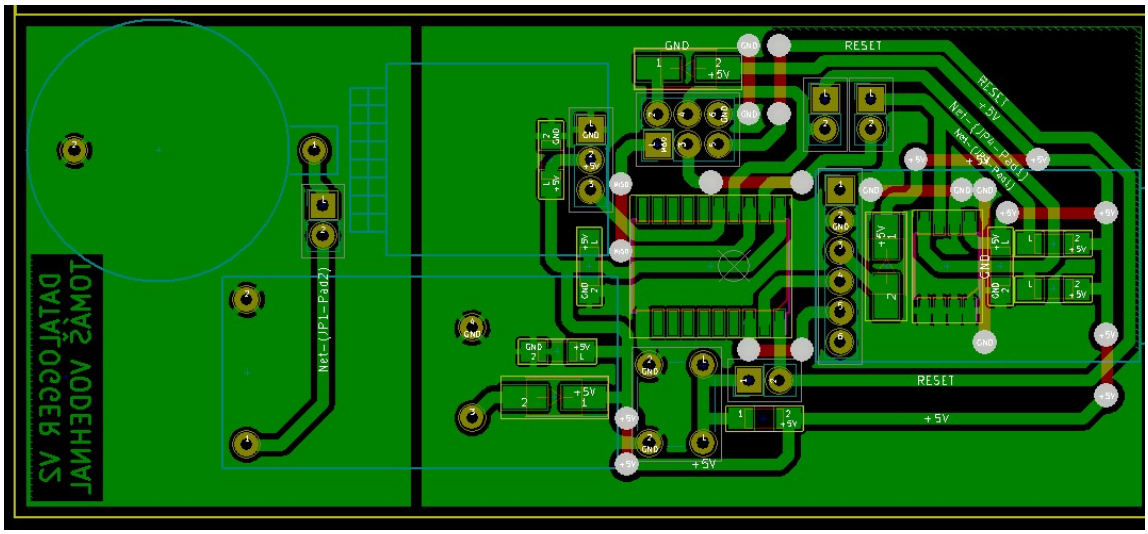
Obrázek 7 zobrazuje schéma zapojení. V levé horní části je zdroj napájení z baterie se step-up měničem. Od zdroje vpravo jsou 4 blokovací kondenzátory. Jeden na výstup step-up měniče, další na napájecí vstup mikrokontroleru a paměti. Poslední na napájecí vstup senzoru. Následuje programovací konektor pro procesory od firmy Atmel. Vlevo od procesoru je zakresleno resetovací tlačítko s pull-up rezistorem. Toto tlačítko je přes jumper připojeno k samotnému mikrokontroleru. Vpravo od mikrokontroleru je nakreslena paměť. Adresní vstupy (A0, A1, A2) paměti jsou připojeny přímo k zemi, tím je nastavena adresa zařízení. Linky SDA a SCL jsou odpojitelné pomocí jumperů. Připojeny jsou také pull-up rezistory. Pod resetovacím tlačítkem je zakreslena pinová řada pro připojení senzoru vlhkosti, pull-up rezistor je již na modulu. Poslední součástka je pinová řada pro připojení převodníku USB-UART.

Deska obsahuje 3 diody. Tyto diody zabraňují toku napájení do periférií, které v danou chvíli nejsou potřeba. V režimu měření není potřeba, aby byl napájen modul USB-UART. V případě režimu komunikace je potřeba odpojit baterku pomocí jumperu a poté připojit datalogger k počítači. V této chvíli je hardware napájen z USB počítače.

2.3.2 Deska plošného spoje

Obrázek 8 zobrazuje výsledný návrh desky plošného spoje (PCB – printed circuit board). Všechny cesty jsou v šíři 1mm z důvodu výrobních limitů. Vrtací otvory taktéž šíře 1mm. Veškeré vývodové součástky jsou myšleny ze strany bez mědi. Veškeré SMD součástky jsou

myšleny ze strany s mědí, proto jsou zrcadlově obráceny. Červené spoje označují umístění drátových propojek.



Obrázek 8 - Návrh PCB

2.4 Popis funkce

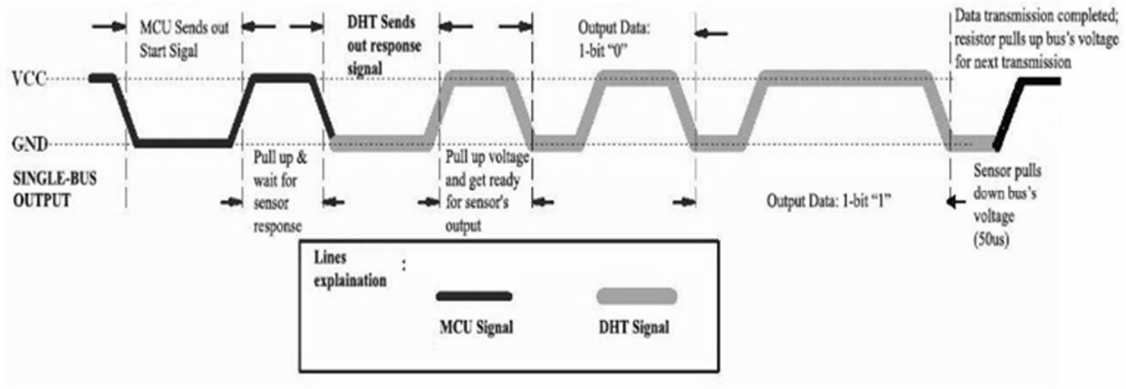
Hlavní smyčka programu se skládá z čekací doby, která určuje periodu měření, samotného měření a ukládání dat. Zároveň je nutné, aby hardware byl schopen rozpoznat, kdy dojde k připojení k PC. Při připojení k PC musí hardware být schopen vyčíst veškerá naměřená data v předem definovaném formátu a odeslat je ve správném pořadí do počítače.

2.5 Firmware

V následující části popisuji jednotlivé bloky programu. Každý odstavec vždy popisuje část dokumentace k dané součástce, ze které jsem vycházel při tvorbě. Následně je uveden seznam funkcí, které daná knihovna obsahuje. Kompletní zdrojové kódy všech knihoven jsou umístěny na CD spolu se spustitelnými projekty. Dále může být v některých případech uveden podrobný popis klíčových funkcí. Popisuji software od nejmenších bloků, k celkovému programu. Všechny knihovny, kromě knihovny pro i2c komunikaci, jsou mým vlastním dílem. Jsou naprogramovány na základě dokumentací k jednotlivým součástkám a znalosti jazyka C. V některých případech jsem při psaní programu čerpal z učebnice jazyka C. [21]

2.5.1 Knihovna pro senzor vlhkosti DHT11

Senzor vlhkosti DHT11 komunikuje definovaným protokolem po jednom vodiči. Každá komunikace má jednotný předem daný průběh, viz Obrázek 9.



Obrázek 9 - Komunikace DHT11 [15]

Mikrokontroler musí poslat start signál a poté uvolnit linku. Připojený pull-up rezistor zajistí návrat na vysokou hodnotu. Senzor poté odpoví stažením linky do nuly a poté ji opět uvolní. Následuje sekvence jednoho bitu. Ta je definovaná nejprve stažením linky po dobu 50us a následné uvolnění. Doba, po kterou je linka následně uvolněna, indikuje logickou 1 nebo 0. Přijímaný bit nabývá hodnoty 0, pokud doba uvolněné linky je 26us – 28us. Bit nabývá hodnoty 1, pokud tato doba je 70us. Senzor vždy pošle všechna naměřená data najednou, včetně kontrolního součtu. To znamená, že po 4 přijatých bitech je komunikace ukončena a je potřeba znovu odeslat startovní podmínku. [15]

Zde je uveden seznam hlaviček a proměnných knihovny.

```
extern char hum_data[5];     //REL_HUM|REL_HUM|TEMP|TEMP|CHECKSUM
extern void hum_sensor();
```

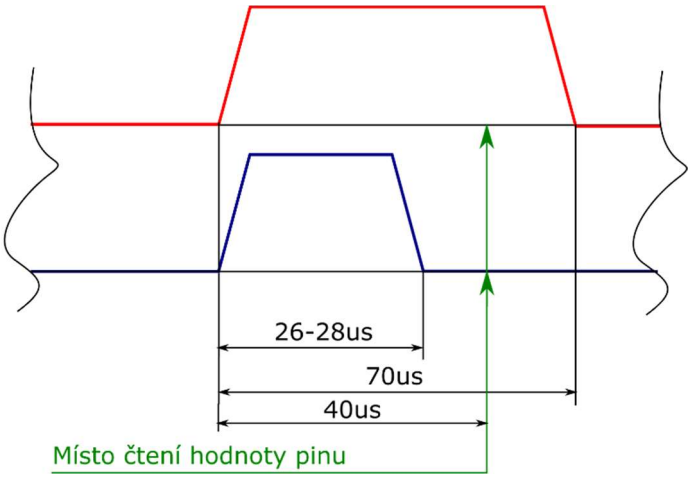
Struktura pole hum_data je naznačena v komentáři. Metoda hum_sensor je hlavní metoda knihovny. Tato metoda nejdříve vyčte data ze senzoru a postupně je uloží do proměnné hum_data na příslušné pozice. Kompletní hlavičkový soubor i soubor knihovny je uveden v příloze.

Stěžejní část této knihovny je vyčítání ze senzoru přesně definovaným způsobem, který je uveden na začátku této kapitoly. Cyklus pro rozeznání logické nuly nebo jedničky je následující:

```
for (char i = 0; i < 5; i++)
{
   for (char j = 0; j < 8; j++)
   {
      while(!(HUMPIN & (1<<HUMPINN)));
      _delay_us(40);
      if (HUMPIN & (1<<HUMPINN))
      {
         hum_data[i] |= (1<<(7-j));
         while(HUMPIN & (1<<HUMPINN))
```

```
            {
               _delay_us(10);
            }
         }
      }
   }
```

První cyklus počítá každý přijatý bajt. Bajtů je celkem 5 , protože struktura dat senzoru DHT11 je 16 bitů vlhkost, 16 bitů teplota, 8 bitů kontrolní součet, tedy dohromady 5 bajtů. Následný vnořený cyklus for probíhá vždy při přijímání každého bitu. Cyklus while počká, dokud senzor neuvolní linku do 1. Poté program počká 40us a následně zjistí, zda ještě stále je na pinu logická jednička. Pokud ano, znamená to dlouhý signál logické 1. Pokud ne, hodnota se neuloží. Protože před spuštěním funkce se globální proměnná hum_data nuluje, na místech, kam nebylo ukládáno, je logická 0. Následující obrázek ukazuje místo čtení hodnoty pinu.



Obrázek 10 - čtení hodnoty sensoru vlhkosti

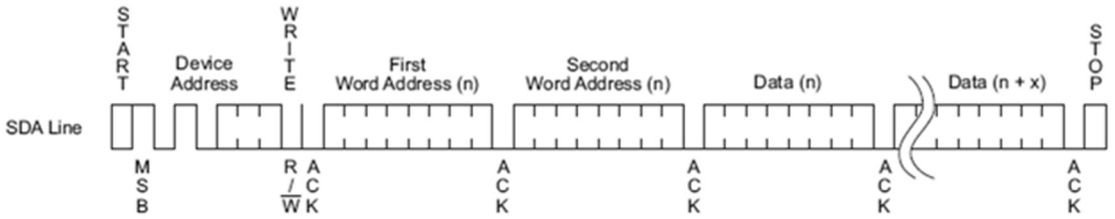
Modrý průběh na Obrázek 10 označuje případ logické nuly, červený logické jedničky. Kompletní kód knihovny je na přiloženém CD.

2.5.2 Knihovna pro komunikaci s pamětí

EEPROM paměť AT24C512C komunikuje po sběrnici I2C. Procesor ATtiny2313A nemá periferii TWI (Two Wire Interface), jako procesory z rodiny mega. Disponuje však periferií USI (Universal Seriál Interface), na které je možné implementovat komunikaci po I2C sběrnici. Toto řešení se mi nepodařilo vytvořit. Našel jsem proto knihovnu pro I2C sběrnici, která celou komunikaci řeší softwarově. V této knihovně stačí nastavit, který pin procesoru bude použit jako SCL (hodinový signál) a který jako SDA (datový signál). [22][17]

V této části je potřeba rozlišovat adresu zařízení a adresu dat. Adresa zařízení slouží pro komunikaci po I2C sběrnici a identifikaci zařízení na sběrnici. Adresa dat je adresa uložených dat v paměti a z hlediska sběrnice se chová jako data.

Struktura pro zápis do paměti je na Obrázek 11.



Obrázek 11 - Zápis do paměti [17]

Komunikace začíná start podmínkou. Následuje adresa zařízení s LSB nastaveným na 0, což znamená zápis. Paměť odpoví potvrzením (ACK). Poté následují dva bajty adresy dat. Nakonec se posílají jednotlivá data. Po každém bajtu odeslaných dat paměť odpoví potvrzením. Je možné odeslat libovolný počet bajtů až do doby, než adresa dat dojde na konec stránky. [17]

Následuje seznam funkcí a proměnných knihovny.

```
extern void ulozit_data(char *data, char pocet, char ad1, char ad2);
extern void vycist_data(char ad1, char ad2);
extern void reset_memory();

extern char mem_data[64];
extern char a1, a2;
```

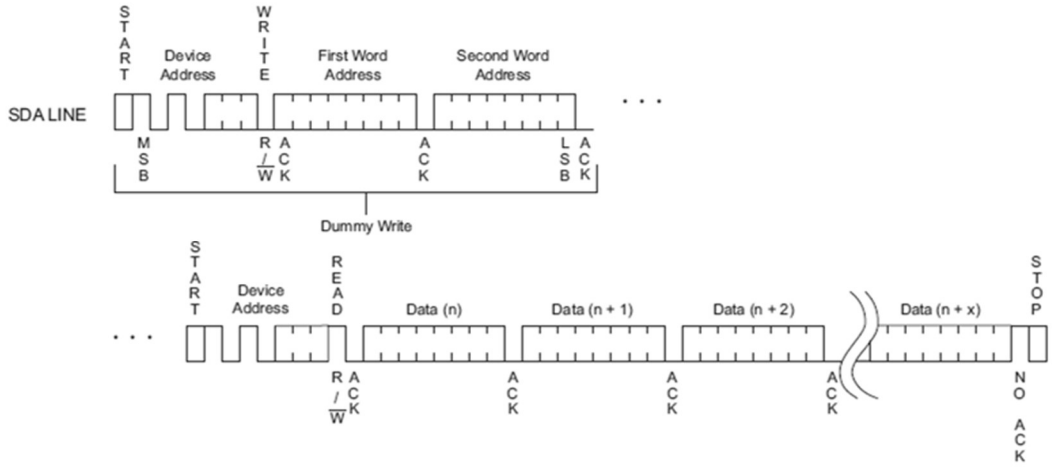
Proměnné a1 a a2 jsou adresní proměnné, které určují místo paměti, se kterým se bude pracovat. Jsou vytvořeny globálně, protože se s nimi pracuje v hlavní smyčce programu, viz kapitola 2.5.6. Proměnná mem_data obsahuje, po provedení funkce vycist_data hodnoty, které byly vyčteny z paměti. Metoda reset_memory není použita nikde v celém běhu programu, zde je pouze z důvodů testování aplikace. Tato metoda uloží do celé paměti na každé místo nulové bity. Dvě stěžejní metody ulozit_data a vycist_data jsou podrobně popsány níže. Jak už jejich názvy vypovídají, jsou použity pro ukládání a vyčítání dat z paměti. Kompletní hlavičkový soubor spolu se souborem knihovny je v příloze.

Programové řešení spolu s využitím knihovny pro I2C komunikaci je následující.

```
void ulozit_data(char *data, char pocet, char ad1, char ad2)
{
   i2c_start(WRITE_ADDRESS);
   i2c_write(ad1);
   i2c_write(ad2);
   for (char i = 0; i < pocet; i++)
   {
      i2c_write(*(data+i));
   }
   i2c_stop();
}
```

WRITE_ADDRESS je makro, které je nahrazeno pamětí zařízení a LSB nastaveným na 0, tj. zápis. Funkce i2c_start je z knihovny i2c_master. Tato funkce odešle start podmínku, adresu zařízení a zkontroluje navrácený ACK. Funkce i2c_write odešle data, v tomto případě adresu ad1 a ad2, a zkontroluje navrácený ACK. Následuje cyklus for, který proběhne tolikrát, kolikrát je počet odesílaných bajtů. Zakončení komunikace je pomocí funkce i2c_stop.

Strukturu pro vyčtení z paměti zobrazuje Obrázek 12.



Obrázek 12 - Čtení z paměti [17]

Nejprve je potřeba určit , od kterého místa v paměti se budou data vyčítat. To je provedeno odesláním příslušných adres na i2c sběrnici. Následuje znovu start podmínka s adresou zařízení s nastaveným LSB na 1, což znamená čtení. Následně paměť začne posílat uložená data. Každý přijatý bajt musí master potvrdit ACK. Poslední bajt již potvrzen není a následuje stop podmínka. [17]

Softwarové řešení s využitím knihovny pro I2C je následující.

```
void vycist_data(char ad1, char ad2)
{
   i2c_start(WRITE_ADDRESS);
   i2c_write(ad1);
   i2c_write(ad2);
   i2c_rep_start(READ_ADDRESS);
   for (char i = 0; i < 63; i++)
   {
      mem_data[i] = i2c_readAck();
      if (mem_data[i] == 0xFF)
      {
         break;
      }
   }
   mem_data[63] = i2c_readNak();
   i2c_stop();
}
```

Nejdříve se odešle adresa zařízení pro zápis a poté adresa místa v paměti. Následuje opakovaná podmínka start s adresou zařízení s nastaveným LSB pro čtení. Cyklus for proběhne 63 krát plus jeden bajt, který se vyčte mimo cyklus. To je polovina jedné stránky paměti. V této metodě se vyčte vždy polovina stránky, protože se v aplikaci nepředpokládá čtení pouze jedné konkrétní hodnoty nebo části paměti. Vždy, když dochází ke čtení, čte se paměť celá. Z důvodu malé kapacity RAM paměti procesoru dochází k vyčítání dat po menších částech, tedy polovinách stránky. Pokud se nalezne v paměti uložená hodnota FFh, znamená to, že dál již nejsou v paměti validní naměřené hodnoty, proto již není nutné dál data číst. Následuje zakončení komunikace podmínkou stop.

Kompletní kód je na přiloženém CD.

2.5.3 Knihovna pro časovač a spánek

Datalogger bude měřit data každých 15 minut, proto je potřeba pomocí časovače těchto 15 minut odměřit. ATtiny2313A disponuje dvěma časovači, jeden 8 bitový a druhý 16 bitový. Protože je potřeba odměřit velmi dlouhý časový údaj, zvolím 16 bitový časovač. Pomocí následujícího vzorce jsem vypočítal výslednou frekvenci, s jakou bude docházet ke shodě s output compare registrem. Tento vzorec vznikl úpravou vzorce z dokumentace k procesoru ATtiny2313A, doplněním o násobení 2. Frekvence se násobí dvěma, protože ke shodě s OCR registrem dochází 2 krát za periodu pinu OCn. [20]

$$f_{OCRA} = 2\frac{f_{clk\_io}}{2N(1 + OCRnA)} \tag{3}$$

kde,

f_{OCRA} – výsledná frekvence shody

f_{clk_io} – frekvence procesoru

N – předdělička (1, 8, 64, 256, 1024)

OCRnA – hodnota output compare registru [20]

Z tohoto vzorce jsem si vyjádřil vztah pro hodnotu OCR registru a dosadil hodnoty. Frekvence procesoru je 1MHz, požadovaná perioda 1 minuta.

$$OCRnA = 2\frac{f_{clk\_io}}{2 * N * f_{OCnA}} - 1 = 2\frac{1000000}{2 * 1024 * \frac{1}{60}} - 1 = 58592{,}75 = 58593_D^{\cdot} = E4E1_H$$

Protože s 16 bitovým časovačem lze dosáhnout maximální periody 134 sekund, což je méně než požadovaných výsledných 15 minut, rozhodl jsem se, že budu vyvolávat přerušení od časovače každou minutu. Toto v cyklu provedu celkem 15 krát. To má velkou výhodu pro případ, že by požadavek na snímání dat byl častější, stačí jednoduchá úprava podmínky cyklu.

Seznam funkcí a proměnných knihovny je následující.

```
extern void sleep_1min();
extern void timer_stop();
```

Metoda sleep_1min je podrobně popsána níže. Obecně tato metoda pouze uspí procesor na 1 minutu. Metoda timer_stop vypne časovač, aby nedocházelo k přerušením od časovače v případě, kdy je datalogger připojen k počítači. Kompletní hlavičkový soubor, spolu se souborem knihovny , je v příloze.

Kód metody sleep_1min je následující.

```
void sleep_1min()
{       //e4, e1
   OCR1AH = 0xE4;
   OCR1AL = 0xE1;
   TIMSK |= (1<<OCIE1A);
   sei();
   set_sleep_mode(SLEEP_MODE_IDLE);
   TCCR1B |= (1<<CS12) | (1<<CS10) | (1<<WGM12);
}
```

První dva řádky nastavují OCR registr podle předem vypočítaných hodnot. Následuje povolení přerušení od časovače a globální povolení přerušení. Nastavení požadovaného módu spánku. Spuštění časovače v CTC režimu s předděličkou 1024.

V hlavní smyčce je poté vidět cyklus, který probíhá 15 krát, a proto odměřuje 15 minut.

```
for (char i = 0; i < 15; i++)
   {
      sleep_1min();
      uart_recive_char();
      sleep_mode();
   }
cli();
```

Nejprve se spustí časovač. Poté se pomocí funkce uart_recive_char aktivuje citlivost na přerušení uartem viz kapitola 2.5.4. Nakonec se zapne režim spánku.

ATtiny2313A má 3 režimy spánku:

- idle mod
- stand by mod
- power down mod.

Tyto 3 režimy se liší citlivostí na určitá přerušení. Idle mod je nejméně spořivý režim uspání, ale reaguje na téměř jakákoliv přerušení. Stand by a power down mod již nereagují na přerušení od časovače, ani na přerušení od uartu, proto se ani jeden nehodí pro čekací smyčku.

Kompletní kód je na přiloženém CD.

2.5.4 Knihovna pro UART

UART (Universal Asynchronous Reciever Transmitter), česky nazýván jako sériová linka. Jde o jeden z nejstarších komunikačních protokolů. Je velmi spolehlivý a ATtiny2313A má pro použití tohoto protokolu vlastní periferii. Komunikace probíhá po rámcích. Každý rámec se skládá z 1 start bitu, 5 až 9 datových bitů, možnost paritního bitu, 1 nebo 2 stop bity. Data se posílají od bitu s nejnižší vahou rychlostí definovanou hodinovým taktem (baudrate). [20]

Hodinový takt se nastavuje v registru UBRR. Protože jsem použil mod dvojnásobné rychlosti, platí pro výpočet hodnoty UBRR registru následující vzorec:

$$UBRR = \frac{f_{osc}}{8 * BAUD} - 1 \tag{4}$$

kde,

UBRR – hodnota UBRR registru

f_{OSC} – frekvence procesoru

BAUD – požadovaná hodnota rychlosti

Požadovaná rychlost je 9600Bd a frekvence procesoru je 1MHz. Proto výsledná hodnota UBRR registru je:

$$UBRR = \frac{1000000}{8 * 9600} - 1 = 12{,}028 \Longrightarrow 12 \tag{5}$$

Seznam všech funkcí a proměnných knihovny je následující.

```
extern char znak;

extern void uart_init();
extern void uart_recive_char();
extern void uart_send_char(char c);
```

Proměnná znak slouží pro uložení příchozího znaku. Tato proměnná je globálního charakteru pro případ, že by s ní bylo potřeba pracovat v hlavní smyčce. Metoda uart_init nastaví správnou hodnotu UBRR registru a zapne periferii uart. Metoda uart_recive_char je podrobně popsána níže. V této funkci je nastavena citlivost na přerušení od uartu. Co se má stát po přijetí znaku

je řešeno v interní funkci uart_char_recived, taktéž je popsána podrobněji níže. Metoda uart_send_char odešle daný znak na sériovou linku, neprobíhá pomocí přerušení.

Nejdříve je potřeba periferii správně nastavit a inicializovat.

```
void uart_init()
{
  UCSRB |= (1<<TXEN)|(1<<RXEN);     //ENABLE RECIVER AND TRANSCIVER
  UCSRA |= (1<<U2X);
  UBRRL |= 12;                  //BAUDRATE 9600
}
```

RXEN a TXEN jsou bity, které zapínají piny procesoru jako komunikační piny seriové linky. U2X je bit, který nastavuje dvojnásobnou rychlost komunikace. Do registru UBRRL, protože je UBRR 16 bitový registr, dělí se na 2 části, zapíšu vypočítanou hodnotu 12.

Metoda pro příjem znaku. Přijetí znaku je řešeno pomocí přerušení.

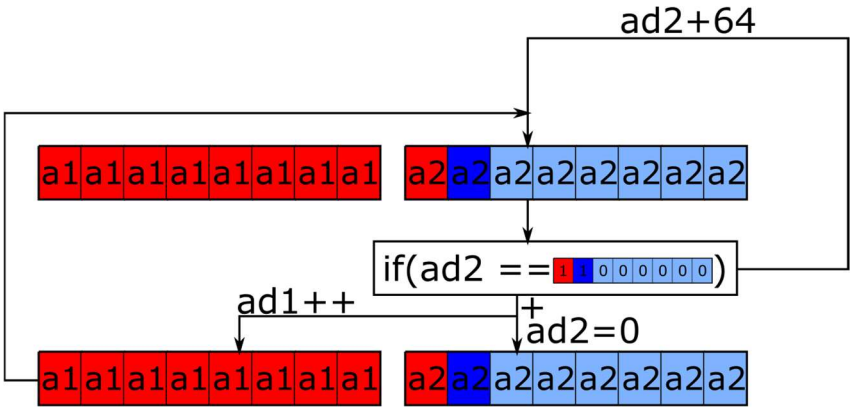
```
void uart_recive_char()
{
  UCSRB |= (1<<RXCIE);
  sei();
}
ISR(USART0_RX_vect)
{
  znak = UDR;
  if (znak == 'v' | znak == 's')
  {
    uart_char_recived();
  }
}
```

Metoda uart_recive_char pouze zapne citlivost na přerušení příchozím znakem uartu. V bloku přerušení se právě přijatý znak uloží do proměnné znak, která je globální a v případě, že znak odpovídá malému v nebo s, je vyvolána metoda uart_char_recived. Tyto dva znaky jsou velmi důležité při komunikaci s aplikací v PC. Následující kód zobrazuje, co se stane v případě, že je přijatý znak v.

```
case 'v':
  for (uint16_t j = 0; j < 1024; j++)
  {
    vycist_data(ad1, ad2);
    for(char i = 0; i < 64; i++)
    {
      uart_send_char(mem_data[i]);
      if (mem_data[i] == 0xFF)
      {
        i = 64;
        j = 1024;
      }
    }
    if (ad2 == 192)
    {
      ad1++;
      ad2 = 0;
```

```
      }
      else
      {
         ad2 += 64;
      }
   }
   prijmout_casovou_znacku();
break;
```

Znak v znamená pro procesor požadavek na vyčtení naměřených dat a odeslání je pomocí sériové linky do počítače. Metoda vycist_data je popsána v kapitole 2.5.2. Adresy jsou zpočátku nastaveny na hodnoty ad1 = 0 a ad2 = 0. Vyčítání probíhá pomocí vnořeného cyklu. První cyklus počítá do 1024, protože každá stránka paměti má 512 řádků, ale vždy každou stránku vyčítám po polovině. Druhý vnořený cyklus počítá do 64, což je polovina stránky paměti. Následující obrázek utváří lepší představu o inkrementaci paměťových proměnných.



Obrázek 13 - Inkrementace paměťových proměnných

Obrázek 13 popisuje vnější cyklus. Vnitřní cyklus vždy vyčte polovinu stránky , tj. 64 hodnot. Po každém čtení poloviny stránky se kontroluje podmínka, zda v dolním bajtu nedošlo již k přetečení adresy. Pokud ano, je přičtena jednička do horního bajtu a dolní bajt je vynulován. Červené bity označují adresu stránky v paměti. Modré bity označují adresu bajtu v paměti. Tmavě modrý bit rozděluje stránku na polovinu, pokud je 0 čte se první polovina, pokud 1 čte se druhá polovina. Světle modré bity představují adresu bajtu v polovině stránky. Po každém vyčtení 64 hodnot se ukazatel v paměti o těchto 64 hodnot posune.

Poslední podmínkou je ukončovací podmínka. Pokud ve vyčtených datech je hodnota 0xFF, tak se cyklus vyčítání ukončí, protože za poslední validní hodnotou dat je vždy uložen ukončovací znak o hodnotě 0xFF. Viz kapitola 2.5.5.

Po vyčtení všech dat se vyvolá metoda prijmout_casovou_znacku.

```
void prijmout_casovou_znacku()
{
   cli();
```

```
   char cas[9] = {0,0,0,0,0,0,0,0,0xFF};
   for (char i = 0; i < 5; i++)
   {
      while (!(UCSRA & (1<<RXC)));
      cas[i] = UDR;
   }
   ulozit_data(cas, 9, 0x00, 0x00);
   _delay_ms(5);
   vycist_data(0x00, 0x00);
   for (char i = 0; i < 8; i++)
   {
      if (cas[i] == mem_data[i])
      {
         if (i == 7)
         {
            uart_send_char(0xAA);
         }
      }
      else
      {
         uart_send_char(0xFF);
         break;
      }
   }
   a1 = 0;
   a2 = 8;
   sei();
}
```

Aplikace v PC musí, po vyčtení všech dat, uložit do paměti novou časovou značku pro začátek dalšího měření. Časová značka zabírá 8 bajtů na začátku paměti. Obsahuje čas zahájení měření, ze kterého se zpětně dopočítají časy jednotlivých měřených hodnot.

V cyklu for se do proměnné cas uloží 5 bajtů, které přijdou sériovou linkou. Následně se celé pole cas uloží na začátek paměti. Metoda obsahuje i zpětnou kontrolu, kdy časovou značku znovu z paměti přečte a pokud se shoduje s hodnotou v poli cas, odešle směrem k aplikaci znak 0xAA. Tento znak je pro aplikaci známkou správného uložení časové značky. Následně se nastaví adresy pro ukládání dat na hodnotu 8, protože na prvních osmi bajtech je právě časová značka. Více o časové značce v kapitole 2.6.3.

V případě, že je přijat znak s, vyvolá se pouze metoda prijmout_casovou_znacku.

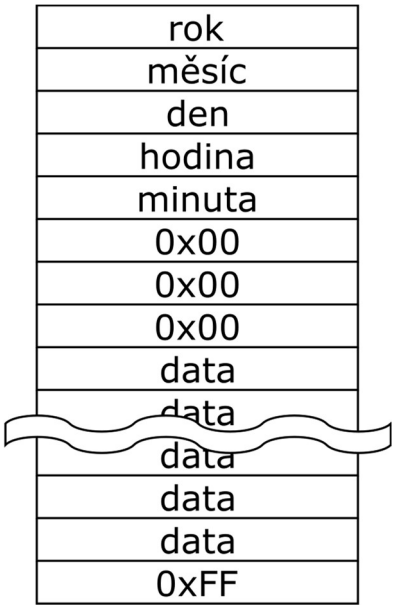
Metoda pro odeslání dat na sériovou linku vypadá následovně:

```
void uart_send_char(char c)
{
   while (!(UCSRA&(1<<UDRE)));
   UDR = c;
}
```

Odeslání probíhá bez přerušení procesoru. Mikrokontroler počká, dokud není registr pro odesílaná data prázdný a následně odešle data.

2.5.5 Způsob uložení dat v paměti

Následující obrázek shrnuje způsob ukládání dat v hlavní smyčce programu.



Obrázek 14 - Způsob uložení dat v paměti

2.5.6 Hlavní smyčka

V hlavní smyčce programu je veškeré měření a následné uspávání procesoru. Cyklus uspávání je již uveden v kapitole 2.5.3. Kód pro ukládání a měření dat je následující:

```
hum_sensor();
if (a2 == 124)
{
   ulozit_data(hum_data, 4, a1, a2);
   _delay_ms(5);
   a2 += 4;
   hum_data[0] = 0xFF;
   ulozit_data(hum_data, 1, a1, a2);
   }
else if (a2 == 252 && a1 != 255)
   {
   ulozit_data(hum_data, 4, a1, a2);
   _delay_ms(5);
   a1++;
   a2 = 0;
   hum_data[0] = 0xFF;
   ulozit_data(hum_data, 1, a1, a2);
}
else if (a1 == 255 && a2 == 252)
{
   ulozit_data(hum_data, 4, a1, a2);
   timer_stop();
   sei();
   set_sleep_mode(SLEEP_MODE_IDLE);
   sleep_mode();
}
else
```

```
{
  hum_data[4] = 0xFF;
  ulozit_data(hum_data, 5, a1, a2);
  a2 += 4;
}
```

Metoda hum_sensor naměří data viz kapitola 2.5.1. Při ukládání se řeší několik situací:

1. a2 = 124

Tento případ značí, že 4 bajty dat se ještě mohou uložit na konec stránky v paměti, ale ukončovací znak se musí uložit již na následující stránku. Adresa a2 se inkrementuje o 4 protože se následujícím uložením do paměti přepíše předchozí ukončovací znak.

2. a2 = 252 a a1 != 255

Tento případ znamená, že data je ještě možné uložit na konec stránky, ale ukončovací znak musí být uložen na následující stránku, jejíž adresa přetekla z dolního bajtu do horního. Proměnná a1 se tedy inkrementuje o 1 a a2 se vynuluje.

3. a2 = 252 a a1 = 255

Tento případ znamená, že je možné uložit poslední naměřená data do paměti na její úplný konec. Dále již není možné další měření ukládat. Paměť je plná. Procesor tedy přejde do spánku a dále již neměří. Jediné přerušení je možné pomocí sériové linky, vyčtením dat nebo zahájením nového měření, viz kapitola 2.5.4.

4. Ostatní případy

V ostatních případech dojde k běžnému uložení dat a inkrementaci ukazatele o 4.

Kompletní kód hlavní smyčky je na přiloženém CD.

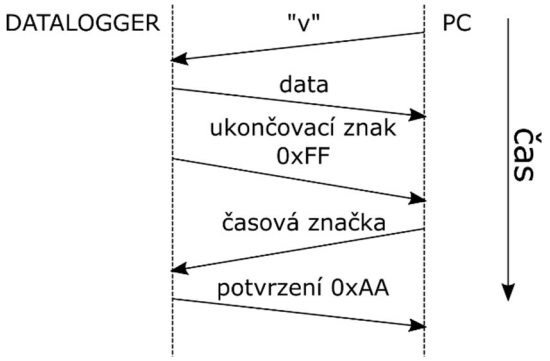
2.6 Aplikace pro PC

Software pro PC musí vyčíst data z paměti dataloggeru, zobrazit je v podobě grafu a uložit do souboru. Aplikace je mým vlastním dílem, vytvořeným na základě znalostí jazyka C# a prostředí .NET.

Kompletní kód aplikace v jazyce C# je na přiloženém CD.

2.6.1 Struktura komunikace

Následující obrázek zobrazuje strukturu, jak probíhá čtení dat z dataloggeru.



Obrázek 15 - Struktura komunikace

Aplikace nejprve vyšle řídicí znak „v“, na který datalogger zareaguje a začne odesílat data uložená v paměti. Za posledním platným naměřeným záznamem je uložen ukončovací znak, který je také odeslán. Ihned po posledním znaku aplikace v PC začne s odesíláním časové značky, kterou zjistí z aktuálního času systému. Hardware časovou značku uloží a potvrdí její přijetí, tím je zahájeno další měření, viz kapitola 2.5.4.

Kromě řídicího znaku „v“ je možné ještě použití řídicího znaku „s“. Hardware je poté připraven přijmout časovou značku a zahájit nové měření, viz kapitola 2.5.4.

Následující část kódu obsahuje softwarové řešení vyčtení dat.

```
serialPort1.Open();
serialPort1.Write("v");
label1.Text = "Čtení";
while (data_prijata != 1) ;
data_do_grafu();
Thread.Sleep(100);
data_prijata = 0;
odesli_casovou_znacku();
serialPort1.Close();
```

Aplikace otevře předem vybraný sériový port. Zapíše řídicí znak „v“. Počká dokud nebudou přijata veškerá data. Metoda data_do_grafu z naměřených dat vykreslí graf, viz kapitola 2.6.2. Dále aplikace odešle časovou značku, viz kapitola 2.6.3, a uzavře sériový port.

Samotné vyčtení dat probíhá v okamžiku, kdy v běhu programu dojde k události prijata_data. Tato událost se spustí v okamžiku, kdy v bufferu sériového portu jsou data, která zatím nebyla vyčtena. Událost prijata_data obsahuje následující kód:

```
while (serialPort1.BytesToRead != 0)
{
      prijataData[pointer] = serialPort1.ReadByte();
      pointer++;
      }
if (prijataData[pointer-1] == 0xFF | prijataData[pointer - 1] == 0xAA)
{
      data_prijata = 1;
```

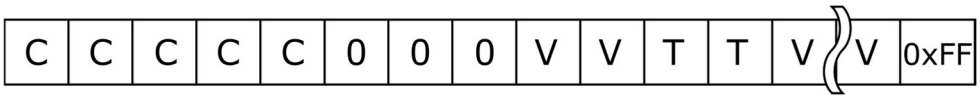
```
}
```

Smyčka while poběží, dokud nebude vyčten celý buffer přijatých znaků. Proměnná prijataData je pole typu int, které obsahuje přijatá data. Jde o proměnnou globální a dále s ní pracuje metoda data_do_grafu, viz kapitola 2.6.2. Podmínka kontroluje, zda není v posledním přijatém bajtu řídicí znak 0xFF pro ukončení komunikace, nebo řídicí znak 0xAA pro potvrzení časové značky.

2.6.2 Vykreslení grafu

Tato kapitola se zabývá metodou data_do_grafu. Tento podprogram má za úkol přijatá data správně roztřídit, dopočítat časy jednotlivých měření a vykreslit graf.

Data přijdou do počítače v následujícím pořadí (Obrázek 16).



Obrázek 16 - Pořadí přijatých dat

C označuje časovou značku, V hodnotu vlhkosti, T hodnotu teploty, 0xFF je ukončovací znak. Následující kód tato data třídí do jednotlivých předem připravených proměnných.

```
for (int i = 0; i < 8; i++)
{
      casovaZnacka[i] = prijataData[i];
}
for (int i = 8; i < pointer-1; i += 4)
{
      vlhkost.Add(double.Parse(prijataData[i].ToString() + "," +
      prijataData[i + 1].ToString()));
      teplota.Add(double.Parse(prijataData[i + 2].ToString() + "," +
      prijataData[i + 3].ToString()));
}
```

První cyklus vyčte prvních 8 bajtů časové značky. Druhý cyklus vyčte data z následujících polí až do konce pole, kromě posledního znaku. Tato data převede do formátu double a uloží do seznamu vlhkost a teplota. Hodnoty vlhkosti a teploty jsou vždy 16 bitová čísla, kde prvních 8 bitů je celá část čísla a spodních 8 bitů je desetinná část čísla.

Další část metody je dopočítání časů k jednotlivým měřením. Její kód je následující:

```
double[] t = teplota.ToArray();
double[] v = vlhkost.ToArray();
DateTime time;
try
{
      time = new DateTime(casovaZnacka[0] + 2000, casovaZnacka[1],
      casovaZnacka[2], casovaZnacka[3], casovaZnacka[4], 0);
}
catch (Exception)
{
```

```
        MessageBox.Show("Nepodařilo se načíst datum! Předpokládá se poslední
        naměřený údaj v " + DateTime.Now.ToLongDateString() + " " +
        DateTime.Now.TimeOfDay);
        time = DateTime.Now.AddMinutes(-1 * t.Length * 15);
}
```

První tři řádky pouze vytváří potřebné proměnné. Následuje blok try. Zde se převede časová značka do formátu datového typu DateTime. Na pozici 0 je vždy uložen rok, jehož číslo je sníženo o 2000, proto zde je nutné tyto 2000 přičíst. Takže pro rok 2017 je na této pozici číslo 17. Na dalších pozicích je hodnota měsíc, den, hodina, minuta. Sekundy se vždy zadávají 0.

V případě, že dojde během tohoto převodu k chybě, je poškozená časová značka, která byla přijata. V takovém případě se počítá s tím, že poslední naměřený údaj byl naměřen v aktuálním čase. Tímto se vždy dopouští chyba, která ale není větší než 15 minut. Časová značka se dopočítá zpětně až k první naměřené hodnotě.

Takto získaná a zpracovaná data zbývá pouze vykreslit do grafu, což dělá následující kód:

```
chart1.Series["Teplota"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.DateTime;
chart1.Series["Vlhkost"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.DateTime;
for (int i = 0; i < t.Length; i++)
      {
      chart1.Series["Teplota"].Points.AddXY(time.AddMinutes(i*15), t[i]);
      chart1.Series["Vlhkost"].Points.AddXY(time.AddMinutes(i*15), v[i]);
      }
chart1.Update();
```

Nejprve se nastaví datový typ osy x pro vlhkost i teplotu na typ DateTime. Následuje cyklus, ve kterém je vložen do grafu údaj o teplotě a vlhkosti a k tomu příslušný čas, ve kterém byl naměřen.

2.6.3 Odeslání časové značky

Časová značka je důležitý údaj, který udává datum, kdy bylo zahájeno měření dataloggeru. Tato značka se odesílá hardwaru po vyčtení předchozích naměřených dat, nebo po odeslaném řídicím znaku „s“. Následující část kódu vytváří novou aktuální časovou značku a odesílá ji hardwaru:

```
char[] cas = { (char)(DateTime.Now.Year - 2000), (char)DateTime.Now.Month,
(char)DateTime.Now.Day, (char)DateTime.Now.Hour, (char)DateTime.Now.Minute
};
for (int i = 0; i < 5; i++)
{
  Thread.Sleep(5);
  serialPort1.Write(cas, i, 1);
}
```

Proměnná cas se nejprve naplní potřebnými daty. Program zjistí aktuální systémový čas, od hodnoty rok odečte 2000, aby bylo možné hodnotu uložit v datovém typu char, a zabírala tak pouze 8 bitů. Tato proměnná se v cyklu postupně posílá hardwaru.

Po odeslání čeká aplikace na potvrzení správně uložené časové značky, viz kapitola 2.5.4. Kód vypadá následovně:

```
while (data_prijata != 1) ;
if (prijataData[pointer - 1] == 0xFF)
{
      data_prijata = 0;
      pomocna++;
      if (pomocna >= 10)
      {
            MessageBox.Show("Opakovaně se nepodařilo zahájit měření.
            Vyndejte a vložte baterii dataloggeru a zkuste to znovu.");
            pomocna = 0;
            serialPort1.Close();
      }
      odesli_casovou_znacku();
}
else if (prijataData[pointer - 1] == 0xAA)
{
      MessageBox.Show("Datalogger měří.");
      pomocna = 0;
      serialPort1.Close();
}
```

Aplikace čeká na příchozí data. V případě, že přijatá odpověď je 0xFF znamená to, že časovou značku hardware neověřil. Aplikace se proto pokusí časovou značku odeslat znovu, maximálně však 10x. Pokud ani po desáté se nepodaří značku uložit, uživatel je vyzván k resetu procesoru. V případě, že přijatá odpověď je 0xAA znamená to, že hardware ověřil uloženou časovou značku a zahájil měření.

Odeslání časové značky probíhá také v případě stisknutí tlačítka zahájení. V tomto případě se odešle řídicí znak „s“ a zavolá se metoda odesli_casovou_znacku, která je popsána výše.

2.6.4 Uložení dat

Načtená data v grafu je možné uložit ve formátu csv na disk. Stěžejní část kódu je následující:

```
FolderBrowserDialog fbd = new FolderBrowserDialog();
fbd.ShowDialog();
if (fbd.SelectedPath != "")
{
      string pth = @"" + fbd.SelectedPath + "\\" + time.Year + "_" +
      time.Month + "_" + time.Day + "_" + time.Hour + "_" + time.Minute +
      "_" + "datalogger.csv";
      using (StreamWriter sw = new StreamWriter(pth))
      {
            double[] uteplota, uvlhkost;
            uteplota = teplota.ToArray();
            uvlhkost = vlhkost.ToArray();
            for (int i = 0; i < uteplota.Length; i++)
            {
                  sw.WriteLine(time.AddMinutes(i * 15).Year + ";"
                  +time.AddMinutes(i * 15).Month + ";" + time.AddMinutes(i
                  * 15).Day + ";" + time.AddMinutes(i * 15).Hour + ";" +
```

```
                    time.AddMinutes(i * 15).Minute + ";" + uteplota[i] + ";"
                    + uvlhkost[i]);
                }
        }
}
```

Nejdříve je vyvolán dialog pro výběr složky, kam se mají data uložit. Název souboru generovaného aplikací má následující tvar: „rok_mesic_den_hodina_minuta_datalogger.csv“ Rozhodující je vždy datum prvního změřeného vzorku. Do proměnných uteplota a uvlhkost se vloží data. Cyklus zapíše řádek po řádku soubor csv. Struktura jednoho řádku je následující: „rok;mesic;den;hodiny;minuty;teplota;vlhkost“

2.6.5 Načtení dat

Aplikace může načíst dříve uložená data a znovu zobrazit ve formě grafu. Stěžejní část kódu je následující:

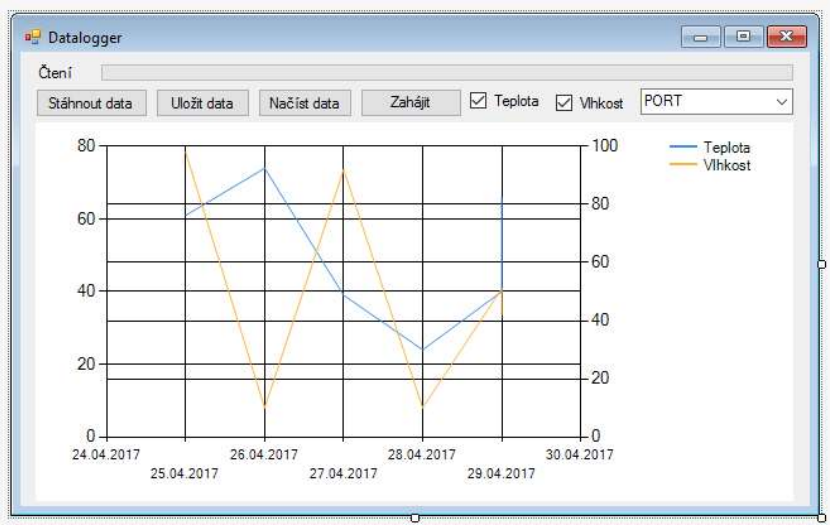
```
using (StreamReader sr = new StreamReader(file))
{
      int rok, mesic, den, hodiny, minuty;
      double t, v;
      string[] value = sr.ReadLine().Split(';');
      rok = int.Parse(value[0]);
      mesic = int.Parse(value[1]);
      den = int.Parse(value[2]);
      hodiny = int.Parse(value[3]);
      minuty = int.Parse(value[4]);
      time = new DateTime(rok, mesic, den, hodiny, minuty, 0);
      t = double.Parse(value[5]);
      v = double.Parse(value[6]);
      chart1.Series["Teplota"].Points.AddXY(time, t);
      chart1.Series["Vlhkost"].Points.AddXY(time, v);
      teplota.Add(t);
      vlhkost.Add(v);while (!sr.EndOfStream)
      {
            string[] value = sr.ReadLine().Split(';');
            rok = int.Parse(value[0]);
            mesic = int.Parse(value[1]);
            den = int.Parse(value[2]);
            hodiny = int.Parse(value[3]);
            minuty = int.Parse(value[4]);
            DateTime cas = new DateTime(rok, mesic, den, hodiny, minuty,
            0);
            t = double.Parse(value[5]);
            v = double.Parse(value[6]);
            chart1.Series["Teplota"].Points.AddXY(cas, t);
            chart1.Series["Vlhkost"].Points.AddXY(cas, v);
            teplota.Add(t);
            vlhkost.Add(v);
      }
}
```

Aplikace otevře StreamReader a přečte soubor řádek po řádku až do konce souboru. Hodnoty v souboru jsou odděleny středníkem a jejich struktura je popsána v kapitole 2.6.4. Samostatně se přečte první řádek, který uloží do globální proměnné time, čas první naměřené veličiny. Tato

hodnota je velmi důležitá, protože se s ní pracuje v dalších metodách. Smyčka vyčte data od 2 řádku, protože první řádek je přečten ještě před vstupem do smyčky a roztřídí data, ze kterých pak vykreslí graf.

2.6.6 Vzhled okna

Obrázek 17 zobrazuje vzhled okna aplikace dataloggeru. Jednotlivé části jsou vysvětleny níže.

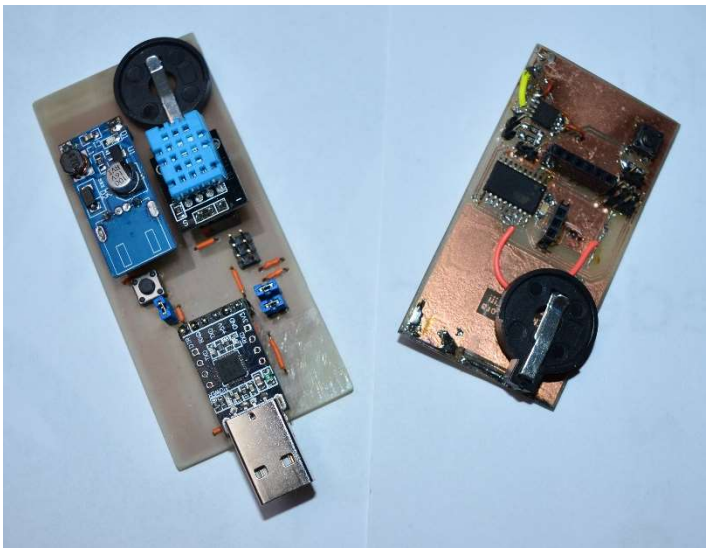


Obrázek 17 - vzhled okna aplikace

Okno obsahuje 4 tlačítka pro práci s daty. Stáhnout data odešle dataloggeru příkaz k vyčtení dat. Uložit data uloží data ve formátu csv na disk. Načíst data načte data ve formátu csv z disku. Zahájit odešle firmwaru příkaz k přijetí časové značky a odešle časovou značku, čímž se zahájí měření. Následné 2 zatrhávací políčka umožňují zobrazit pouze vlhkost nebo teplotu v grafu. Nabídka port umožňuje vybrat port, ke kterému je datalogger připojen. Hlavní část okna zabírá oblast grafu. V horní části je vložen progress bar, který indikuje probíhající čtení z hardwaru.

2.7 Oživení

V první verzi desky bylo během oživování nalezeno několik hardwarových i softwarových chyb. Firmware i software byl opraven. Řešení hardwarových chyb bylo vyřešeno novou verzí desky. Obrázek 18 zobrazuje porovnání dataloggeru verze1 a 2.



Obrázek 18 - Výsledný výrobek, zleva verze 2 a verze 1

Následuje popis některých chyb, které byly zjištěny během oživování.

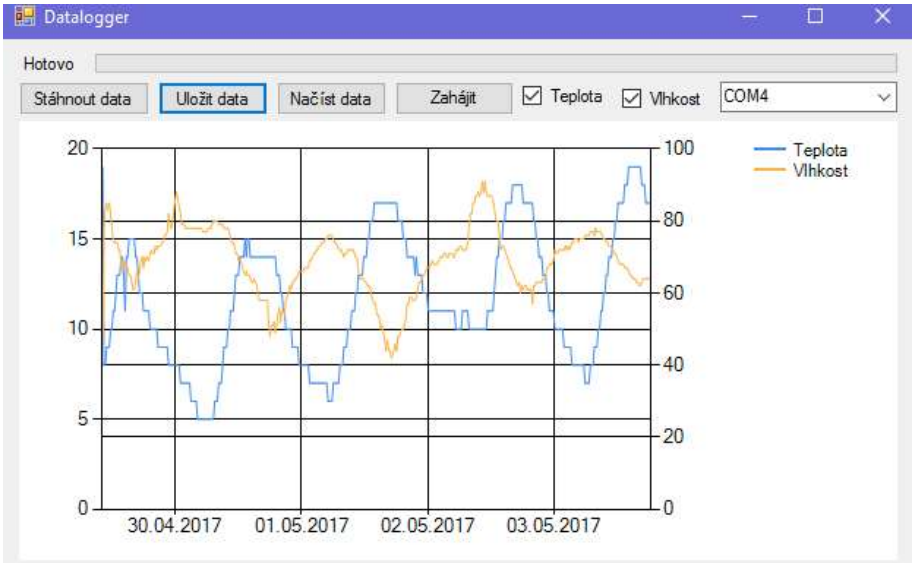
Problém s komunikací hardwaru a PC. Zjistil jsem, že aplikace v počítači odesílá časovou značku příliš brzy po přijetí zakončovacího znaku. Proto jsem odesílání časové značky přesunul až za vykreslení dat do grafu a ještě jsem přidal uspání vlákna na 100ms, viz kód v kapitole 2.6.1. Problém byl následně vyřešen.

Špatné počítání adresy paměti. Ve dvou vnořených cyklech popsaných v kapitole 2.5.4, jsem objevil chybu v konstantách, které určují počet opakování cyklů. Konstanty jsem opravil a problém byl vyřešen.

Hardwarové nedostatky jsou zřejmé z Obrázek 18. Na desku jsem navrhl příliš úzké cesty, které se podleptaly. Příliš malé díry, které nebylo možné vyvrtat. Špatně jsem otočil piny pro senzor vlhkosti. Pájecí plošky pro paměť byly příliš úzké. Tento seznam vyústil v novou verzi desky. Taktéž na Obrázek 18.

Proudový odběr hardwaru a provoz na baterii. Při prvních odhadovaných výpočtech jsem počítal s provozem na 3V knoflíkovou baterii 220mAh cca 1 měsíc. Do tohoto čísla ale nebyl zahrnut odběr step-up měniče, protože k němu není k dispozici dokumentace. Jak se ukázalo, před měničem jsem naměřil proudový odběr 10mA a za měničem je odběr již minimální a běžným multimetrem neměřitelný. Tento problém nebyl dosud vyřešen zcela a poskytuje prostor pro další vylepšení. Na desku jsem přidal některé diody, které popisuji v kapitole 2.3.1.

Obrázek 19 ukazuje naměřená data za dobu 4 dnů od 30.4. do 3.5. Měření jsem provedl na půdě. Je zřetelně vidět, jak se teplota během dne mění a v místě, kdy je teplota nejvyšší je relativní vlhkost nejnižší, viz kapitola 1.3.



Obrázek 19 - Ukázka naměřených dat

3 ZÁVĚR

Zadáním práce bylo vytvořit datalogger pro měření teploty a vlhkosti prostředí. Výsledkem je prototyp, který nabízí prostor pro další vývoj. Výrobek je schopen měřit vlhkost i teplotu a následně je odeslat do počítače pomocí uživatelské aplikace. Tato aplikace vykreslí data v podobě grafů a umožňuje uložení těchto dat. Také umožňuje prohlížet dříve uložená měření.

V zadání práce je požadování napájení z miniaturní baterie. Tento bod se podařilo naplnit pouze z části. Původní představa počítala s využitím devítivoltové baterie spolu se stabilizátorem na 5V. Z mého pohledu, ale 9V baterie se nedá označit jako miniaturní, proto jsem navrhl jiné řešení. Datalogger je napájen z 3V baterie přes step-up měnič, který jsem zakoupil v podobě modulu. Toto řešení se ukázalo jako špatné, protože step-up měnič příliš zatěžuje baterii a její napětí velmi rychle klesá, z důvodu malého vnitřního odporu. Výhodnější by bylo použití nábojové pumpy, která nepředstavuje tak velkou zátěž.

Lepší řešení napájení představuje prostor pro další vývoj. Dále bych rád rozšířil datalogger o měření atmosférického tlaku a rychlosti větru. Jako komunikační rozhraní plánuji wi-fi, nebo jiný bezdrátový přenos. Aplikaci chci rozšířit o možnost online sledování veličin. Pro měření ve venkovních podmínkách je potřeba nahradit senzor DHT11, například kompatibilním senzorem DHT22.

4 POUŽITÁ LITERATURA

[1] Data Loggers. *National Instruments* [online]. Austin: National Instruments, c2017 [cit. 2017-03-07]. Dostupné z: https://www.ni.com/data_logger/

[2] *What is Data Logging?* [online]. Austin: National Instruments, c2017 [cit. 2017-03-07]. Dostupné z: https://www.ni.com/data_logger/whatis.htm

[3] 2. blok Pasivní snímače. *LEARN* [online]. Pardubice: Univerzita Pardubice, c2011 [cit. 2017-03-08]. Dostupné z: http://fei-learn.upceucebny.cz/mod/resource/view.php?id=3682

[4] 3. blok Aktivní snímače. *LEARN* [online]. Pardubice: Univerzita Pardubice, c2011 [cit. 2017-03-08]. Dostupné z: http://fei-learn.upceucebny.cz/mod/resource/view.php?id=3684

[5] Vlhkost vzduchu. *Meteorologie* [online]. Ostrava: Mgr. Antonín Balnar, 2001 [cit. 2017-03-08]. Dostupné z: http://artemis.osu.cz/Gemet/meteo2/vlhkost.htm

[6] Vlhkoměry. *HW.cz* [online]. Praha: Redakce HW serveru, 2004 [cit. 2017-03-09]. Dostupné z: http://vyvoj.hw.cz/teorie-a-praxe/dokumentace/vlhkomery.html-0

[7] 5. blok Měření teploty. *LEARN* [online]. Pardubice: Univerzita Pardubice, c2011 [cit. 2017-03-09]. Dostupné z: http://fei-learn.upceucebny.cz/mod/resource/view.php?id=3688

[8] Advantages of PC-Based Data Logging. *National Instruments* [online]. Austin: National Instruments, c2017 [cit. 2017-03-10]. Dostupné z: https://www.ni.com/data_logger/advantages.htm

[9] Vnitřní paměti. *Fakulta informatiky MU* [online]. Brno: Jaroslav Pelikán, c1998 [cit. 2017-05-03]. Dostupné z: http://www.fi.muni.cz/usr/pelikan/ARCHIT/TEXTY/INTPAM.HTML

[10] CP2102/9. In: *Silicon labs* [online]. Austin: Silicon laboratories, c2017 [cit. 2017-05-03]. Dostupné z: https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf

[11] CP2102. In: *Sparkfun* [online]. Denver: Silicon Laboratories, c2004 [cit. 2017-03-11]. Dostupné z: https://www.sparkfun.com/datasheets/IC/cp2102.pdf

[12] DC/DC měniče. *UArt* [online]. Vlastimil Slinták, 2013 [cit. 2017-03-11]. Dostupné z: http://uart.cz/952/dc-dc-menice/

[13] Learn More about ThingSpeak. *Thingspeak* [online]. Natick: MathWorks, c2017 [cit. 2017-03-12]. Dostupné z: https://thingspeak.com/pages/learn_more

[14] RFC 4180. *Common format and..* [online]. Y. Shafranovich, 2005 [cit. 2017-03-15]. Dostupné z: https://tools.ietf.org/html/rfc4180#section-23

[15] DHT11 Humidity & Temperature Sensor. *Droboticsonline* [online]. D-Robotics UK, 2010 [cit. 2017-05-03]. Dostupné z: http://www.micropik.com/PDF/dht11.pdf

[16] Digital-output relative humidity.. *Sparkfun* [online]. Canada: Aosong, c2017 [cit. 2017-03-15]. Dostupné z: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[17] AT24C512C. *Atmel* [online]. San Jose: Atmel, 2010 [cit. 2017-03-25]. Dostupné z: http://www.atmel.com/images/Atmel-8720-SEEPROM-AT24C512C-Datasheet.pdf

[18] Baterie lithiová knoflíková.. *GM Electronic* [online]. Praha: GM Electronic, c2017 [cit. 2017-03-25]. Dostupné z: https://www.gme.cz/baterie-lithiova-knoflikova-westinghouse-cr2032-3v-lithiova

[19] Step-up měnič z 0.9V.5V na 5V USB. *GM Electronic* [online]. Praha: GM Electronic, c2017 [cit. 2017-03-25]. Dostupné z: https://www.gme.cz/step-up-menic-z-0-9v-5v-na-5v-usb

[20] ATtiny2313A/ATtiny4313. *Atmel* [online]. San José: Atmel Corporation, 2011 [cit. 2017-03-25]. Dostupné z: http://www.atmel.com/images/doc8246.pdf

[21] HEROUT, Pavel. *Učebnice jazyka C*. 6. vyd. České Budějovice: Kopp, 2009. ISBN 978-80-7232-383-8.

[22] I2C Master library. *AVR-GCC Libraries* [online]. Peter Fleury, 2015 [cit. 2017-03-25]. Dostupné z: http://homepage.hispeed.ch/peterfleury/doxygen/avr-gcc-libraries/group__pfleury__ic2master.html

5 PŘÍLOHY

Práce neobsahuje žádné přílohy.