VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

INFORMAČNÍ PORTÁL S WWW ROZHRANÍM

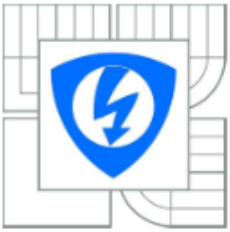
INFO-PORTAL WITH WWW

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. JIŘÍ PILNÝ
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. ZDENĚK BRADÁČ, Ph.D.
SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Jiří Pilný	***ID:***	44419
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Informační portál s WWW rozhraním

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh a realizace informačního portálu, který umožní zobrazovat data zasílaná z centrální stanice na velkoplošném displeji BUSE.

1. Zpracujte literární rešerši obdobných systémů.
2. Navrhněte systém s rozhraním Ethernet, s funkčností základní meteorologické stanice a s měřením prostorových parametrů v interiéru.
3. Popište návrh a navrhněte elektrické schéma.
4. Navrhněte elektroniku, realizujte DPS, osaďte a oživte.
5. Napište programové vybavení pro řídicí systém i PC.
6. Ověřte, otestujte a demonstrujte funkčnost. Vyhodnoťte výsledky.

DOPORUČENÁ LITERATURA:

Pavel Herout: Učebnice jazyka C, KOPP, 2004, IV. přepracované vydání, ISBN 80-7232-220-6
Dle pokynů vedoucího práce.

Termín zadání:	10.2.2014	***Termín odevzdání:***	19.5.2014

Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.
Konzultanti diplomové práce:

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

UPOZORNĚNÍ:

Abstrakt

Práce se zabývá návrhem a realizací informačního portálu, který je ovládán pomocí rozhraní Ethernet. Elektronika se skládá ze tří částí: server, řídící elektronika a samotný zobrazovací panel. Zprávy k zobrazení jsou v rámci místní sítě zasílány na server nebo z něj naopak načítány. Informační portál je vybaven čidly k měření parametrů interiéru (teplota, vlhkost, tlak).

Klíčová slova

Informační portál, teplota, vlhkost, atmosférický tlak, Raspberry Pi, Linux, Apache2, MySQL5, PHP5, AT Mega32, BUSE BS110

Abstract

This thesis deals with conception of info-portal controlled by Ethernet connection. The system is divided into three parts: web server, control electronics and flip-dot panel. The information to display is sent to or received from the web server. The info-portal is equiped by sensors for measurement of interior parameters (temperature, humidity and atmosferic pressure).

Keywords

Info-portal, temperature, humidity, atmosferic pressure, Raspberry Pi, Linux, Apache2, MySQL5, PHP5, AT Mega32, BUSE BS110

Bibliografická citace:

PILNÝ, J. *Informační portál s WWW rozhraním.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 74 s. Vedoucí diplomové práce doc. Ing. Zdeněk Bradáč, Ph.D..

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Informační portál s WWW rozhraním jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 19. května 2014

……
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce doc. Ing. Zdeňku Bradáčovi, Ph.D. za metodickou, pedagogickou a odbornou pomoc při zpracování mé diplomové práce.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

Na základě zadání této práce se budu zabývat sestavením informačního portálu. Systém musí být možno připojit přes rozhraní Ethernet do počítačové sítě a ovládat jej například pomocí WWW prohlížeče z jiného zařízení na téže síti. Vybraná data budou následně zobrazena na panelu BUSE 110. Elektronika infomačního portálu bude vybavena funkcí jednoduché meteostanice s možností měřit základní parametry interiéru. Pro tyto účely využijeme čidla pro měření teploty, vlhkosti vzduchu a atmosférického tlaku.

Informační panely zahrnují obecně velkou škálu zobrazovacích zařízení s různým provedením a funkcí. Taková zařízení vídáme doslova na každém kroku na nádražích, v nákupních centrech, turistických oblastech, atd. Základní funkcí informačního panelu je zobrazit informaci, která je na zařízení zaslána. Může jít o jednoduché statické zobrazení, kdy na panelu je zobrazen jednoduchý text, či piktogram. Na druhé straně jsou běžné i portály s velkými barevnými obrazovkami, kde se informace zobrazují jako videosekvence mnohdy i se zvukovým doprovodem. To je ovšem doména hlavně reklamní oblasti.

Zobrazovací panel BUSE 110, který bude hlavním zobrazovacím prvkem v popisovaném informačním portálu je zařízení původně určené k zobrazení čísla autobusu a názvu stanice. Panel bývá instalován ve vozech MHD a je tak využíván k zobrazení krátké textové nebo číselné zprávy. Pro naši potřebu plně vyhovuje, protože i naše informace budou především textového nebo číselného charakteru. Přesto bude naším cílem ovládat jednotlivé pixely panelu samostatně, takže by v případě potřeby neměl být problém se zobrazením ani jednoduché grafiky.

Z obecného pohledu je možností využití panelu nepřeberné množství. Vedle periodického zobrazování základních informací například o datu, času, teplotě, vlhkosti vzduchu a atmosférickém tlaku, může panel zobrazovat data zaslána v reálném čase z nějakého zařízení přes www rozhraní. Řídící systém panelu může data k zobrazení získávat automaticky například i z internetu, čímž získáme obrovskou škálu dalšího využití. Na druhou stranu je třeba vnímat i jasná omezení, která využití tohoto typu panelu přináší. Panel BUSE 110 má poměrně malé rozlišení 140x19 bodů. Princip jeho zobrazení, jak bude popsáno dále je založen na otáčení jednotlivých bodů pomocí magnetického pole. Každý bod může být směrem k pozorovateli otočen stranou opatřenou černým povrchem nebo stranou s reflexní žlutou barvou. Samotný zápis na panel probíhá bod po bodu a je z principu poměrně pomalý.

Jak vyplývá z přechozího odstavce, informační portál bude nejvhodnější použít na zobrazení krátkých zpráv a jednoduchých obrazových informací s relativně nízkou frekvencí změn. Velkou výhodou panelu pak může být jeho nízká spotřeba elektrické energie v případě, že informace je staticky zobrazena na panelu. V takovém případě je

možné panel odpojit od napájení a informace zůstane zachována. I když to není pro naše účely zásadní, jedná se o zajímavou vlastnost.

Celý systém portálu se bude skládat ze čtyř hlavních částí:

- Server (mikropočítač s Ethernet rozhraním)
- Senzory (měření teploty, tlaku a vlhkosti vzduchu)
- Řídící elektronika (převod informací ze serveru na panel)
- Panel BUSE110

Cílem návrhu je funkční portál s co nejuniverzálnější možností použití. Zadání práce záměrně neuvádí typ zpráv, konkrétní využití a umístění panelu. Při návrhu a realizaci bychom měli tedy mít na paměti, že celý informační portál by sice měl být hotovým a funkčním celkem, měl by ale informovanému uživateli poskytnout jednoduchou možnost, jak funkce panelu pozměnit, či doplnit. Obojí jak po sofwarové, tak harwarové stránce. Během návrhu a realizace panelu se tedy zaměřím na využití standartních rozhraní a programových funkcí tak, aby případné rozšíření funkčnosti portálu bylo otázkou připojení dalšího zařízení pomocí standartní sběrnice a přidání několika bloků kódu do řídícího programu.

2 TEORETICKÝ ROZBOR

2.1 PANEL BUSE110

Pro zobrazení dat informačního portálu máme k dispozici panel BUSE BS110. Výrobcem zařízení je společnost BUSE s.r.o., Blansko, Česká Republika.

Obr.1: panel BUSE BS110

Jedná se o zobrazovací část informačního systému pro městskou hromadnou dopravu. Panel je určen k zabudování do kabiny autobusu, kde slouží k zobrazení informací o lince spoje a stanici. Použitá zobrazovací technologie je realizována pomocí elektromagnetických „štripsů“ DOT. Každý pixel obsahuje otočný kruhový magnetický disk o průměru 10 mm. Každý disk má dvě stabilní polohy. Jedna plocha disku je v matné černé barvě a druhá je opatřena žlutou reflexní fólií. Výhodou je velmi dobrá čitelnost, která není příliš závislá na intenzitě osvětlení. K překlopení terče dochází na základě krátkého proudového impulzu, na jehož konci se terč překlopí do požadované polohy, v níž může setrvat neomezeně dlouhou dobu. To, na kterou stranu se disk otočí je dáno směrem protékajícího proudu. Zápis a mazání bodu tedy provádíme přepólováním napájecího napětí na kontakty cívky „štripsu“.

Výraz „štrips DOT“ jsem nalezl v materiálech firmy BUSE s.r.o [1] a nikdy předtím jsem se s ním nesetkal. V zahraničních materiálech bývá tato technologie popsána termínem FLIP-DISC nebo FLIP-DOT. Technologie jak taková bývá využívána již řadu let, a to na informačních panelech instalovaných na nádržích, či přímo autobusech. V minulosti se jednalo o jednu z mála technologií použitelných pro velkoplošná zobrazení dat (burzy, dálniční informační panely, ceníky čerpacích stanic, atd.).

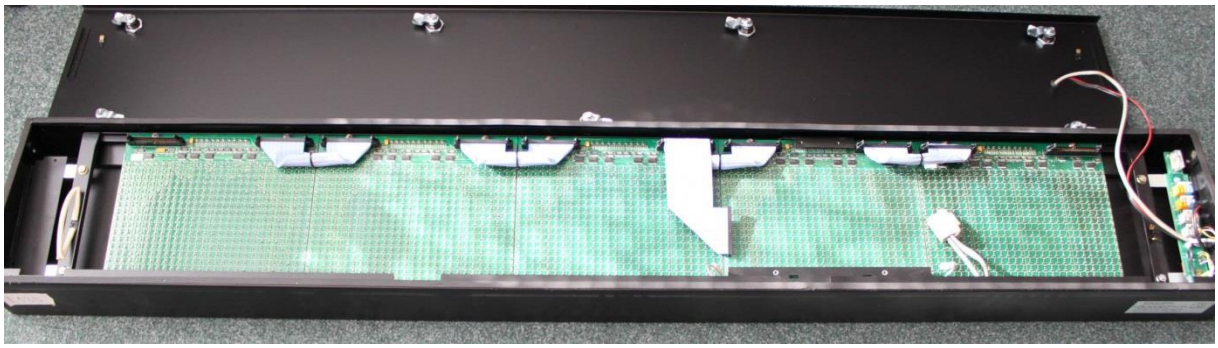
Řídící systém panelu BUSE BS110 počítá s použitím právě jen jako informačního portálu městského autobusu. Elektronika zařízení neumožňuje ovládat jednotlivé pixely a zobrazení dat je závislé na pevně definované znakové sadě. Komunikace a zadávání dat se děje po sériové sběrnici systémem IBIS. IBIS je evropský informační systém, který popisuje konkrétní datové věty určené pro řídící systém panelu. Přenosová rychlost je 1200Bd, data jsou odesílána po 7mi bitech se sudou paritou a organizována do vět s proměnnou délkou. Data jsou přijata do BUFFERU a postupně vypisována na panelu.

Na základě zadání projektu nelze stávající řídící elektroniku panelu BUSE BS110 použít. Nemáme k dispozici žádné další zařízení systému IBIS a projekt počítá pouze s využitím zobrazovacího panelu. Je tedy třeba navrhnout nový řídící systém a rozhraní mezi zobrazovacím panelem a novým systémem. Předpokládáme řízení panelu na úrovni jednotlivých pixelů. V původním panelu zůstane tedy jen ovládací elektronika elektromagnetických disků a bude navrženo nové rozhraní. Nový systém by měl umožňovat i kontrolu a testování jednotlivých „štripsů“, neboť se jedná o mechanickou součástku, jejíž spolehlivost je jistě omezena.

2.2 ELEKTRONIKA PANELU BUSE 110

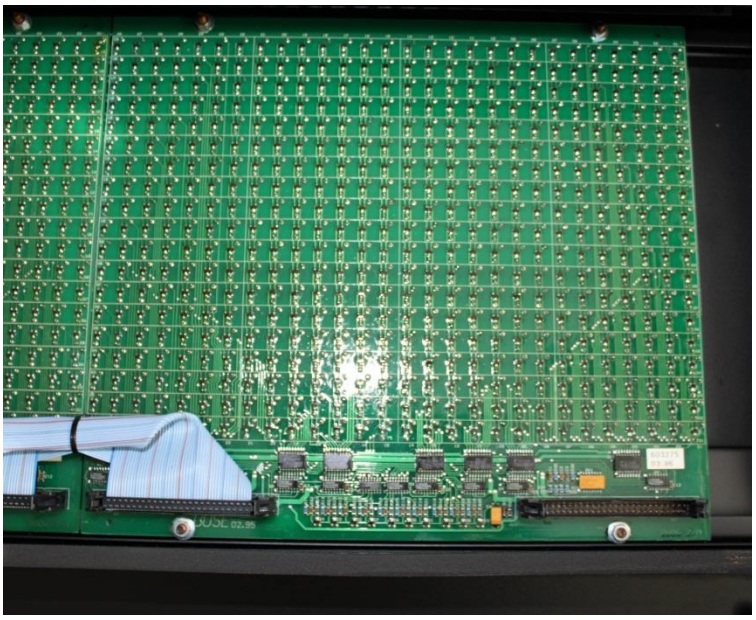
Jak již bylo zmíněno v předchozí kapitole, panel BUSE BS110 disponuje vlastním řídícím systémem, který odpovídá požadavkům na zobrazování informací o linkách a stanicích autobusu, ale není dostatečně flexibilní, aby vyhověl zadání. Hlavním úkolem projektu je tedy vedle naprogramování webového rozhraní také propojení stávajícího elektronického vybavení panelu BUSE BS110 s novým řídícím modulem.

Panel BUSE se skládá z pěti zobrazovacích segmentů s rozlišením 28x19 bodů. Celkově tedy panely pokrývají plochu s rozlišením 140x19 bodů, což činí 2660 pixelů.



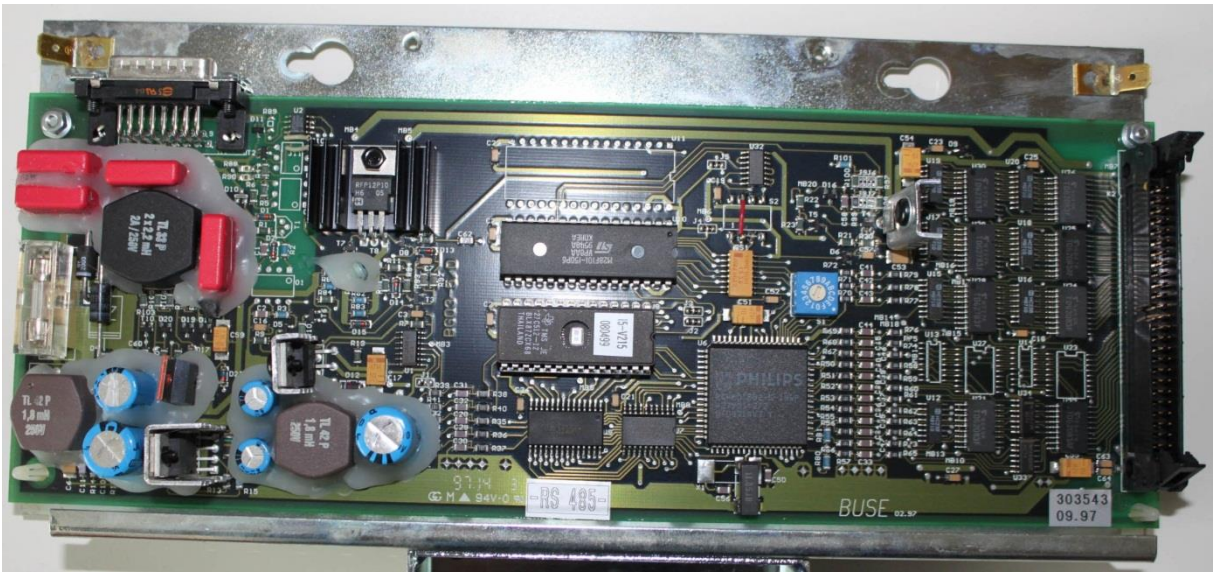
Obr. 2: panel BUSE je sestaven z pěti identických panelů

„Nahazování“ a shazování jednotlivých pixelů se děje doslova bod po bodu. K zápisu nebo smazání konkrétního bodu je třeba poměrně značný proudový impuls 350 mA, proto není současný zápis několika bodů možný. Obvody elektroniky panelu totiž obsahují tranzistorová pole s maximálním výstupním proudem 500 mA. Zadání konkrétního sloupce probíhá v multiplexním režimu. Sloupců je 140 a řídící procesor nemůže obsloužit jednotlivě všechny sloupce. Volí se tedy panel, skupina (osmice) sloupců na panelu a dále konkrétní sloupec. Detail panelu je na obrázku 3.



Obr.3: detail segmentu 28x19 bodů

Původní řídící elektronika panelu BUSE obsahuje 8-bitový mikroprocesor PHILIPS PCF80C552, paměť CMOS EPROM 512K (64Kx8), flash paměť 1Mbit (128Kx8) a paměť SRAM 64K (8192wordsx8bits). Původně je tedy systém realizován tak, že po zapnutí se spustí program uložený v EPROM paměti, který dále reaguje na instrukce a zpracovává data uložená ve flash paměti. V té jsou uloženy znakové sady, informace o zastávkách a linkách autobusu, piktogramy, atd. Buffer displeje pak tvoří paměť SRAM, kde se do fronty zapisují znaky, které jsou postupně vykreslovány na zobrazovači. Detail elektroniky panelu je na obr. 4.



Obr. 4: detail řídící elektroniky BUSE BS110

Řídící elektronika a panely jsou spojeny 50ti žilovým kabelem, který tvoří sběrnici se zobrazovanými daty. Proudový impuls pro překlopení jednotlivých segmentů dodávají obvody TD62783F a TD62083F (transistorová pole v bipolárním zapojení). Volbu segmentů, řádků a bodů pak zajišťují klopné obvody PHILIPS 74HC238D (demultiplexor s osmi výstupy).

Komunikace řídící jednotky s panelem je následující:

Řádky panelu se adresují přímo a nezávisle. Na komunikační sběrnici, realizované 50ti žilovým kabelem je prvních 19 pinů vyhrazeno právě řádkům. Volba řádku je uskutečněna pomocí obvodu TD62783F nebo TD62083F . Výstupy obou těchto obvodů jsou připojeny na sběrnici a volbou obvodu je uskutečněno buď připojení 24 VDC nebo 0 VDC, podle toho, jakým směrem má protékat proud cívkou jednotlivých magnetických disků.

Volba konkrétního sloupce panelu je komplikovanější. V panelu je namontováno 5 panelů s 19 ti řádky a 28 sloupci. Volba jednotlivého segmentu je uskutečněna pomocí tří bitů na pinech 36-38. Pin 36 náleží bitu 0, pin 37 bitu 1 a pin 38 bitu 2. Volbu panelu provedeme tedy zápisem binárního čísla v rozsahu 0-4 (respektive 0b000 – 0b100) na příslušné piny sběrnice. V závislosti na použité kombinaci dojde k tomu, že „hlavní“ demultiplexer 74HC238D na příslušném panelu je odblokován shozením signálu E2 do logické 0. To zajistí obvod 74HC85D (4 bitový komparátor) na panelu.

Na výstupy 0-7 „hlavního“ demultiplexoru je připojeno dalších 8 obvodů stejného typu. Hlavní demultiplexor je adresován 3 bity na pinech 33-35. Pomocí těchto bitů je vybrán konkrétní demultiplexor připojený k určitým 8 sloupcům panelu. Každá osmice sloupců je obsluhována dvěma typy obvodů. Opět to jsou obvody typu TD62783F a TD62083F. V rámci panelu je 28 sloupců, tedy 4 skupiny sloupců po osmi sloupcích (poslední skupina má jen 4 sloupce). Na každou skupinu připadají dva multiplexory. Jeden ovládá výstupy obvodu TD62783F, to když chceme na druhý pól „štripsu“ přivést 24VDC, druhý ovládá výstupy obvodu TD62083F, to když chceme na druhý pól „štripsu“ přivést 0 VDC. „Hlavním“ demultiplexorem tedy aktivujeme konkrétní obvod, který následně vybere pin sloupce.

Po aktivaci konkrétního z osmice multiplexorů, provedeme adresaci konkrétního sloupce v dané osmici, která přísluší tomuto obvodu. To se udělá nastavením tří bitů na pinech 30-32. Máme tedy vybrány řádky, které chceme „otočit“ a nyní i konkrétní sloupec. Proudový impuls k překlopení „štripsů“ dáme nahozením signálu enable na pinu 38. Stejně jako všechny signály na pinech 30-39 je i tento signál ošetřen pull-up rezistory.

Popis zapojení jednotlivých signálů sběrnice panelu BUSE 110 je v tabulce 1. Protože v této kapitole popisuji stávající situaci elektroniky panelu zatím bez návrhu nového systému, je na pinech 43,44 a 46 uvedeno napětí 24 V. Jedná se o napětí, které bude přivedeno na cívky jednotlivý bodů v případě výmazu bodu (překlopení matné černé strany směrem ven). Protože princip funkce magnetického disku umožňuje i

použití nižšího napětí, používal jsem pro testovací účely napětí 12 V. K tomu mne ze začátku vedla obava z možnosti poškození elektroniky panelu, protože konkrétní dokumentaci původního zapojení jsem k dispozici neměl. Protože funkce překlopení jednotlivých disků byla takto ovlivněna „pouze“ nutností o něco prodloužit délku proudového impulzu, zůstal jsem u použití napětí 12 V. Problematika trvání proudového impulzu bude zmíněna v dalších kapitolách.

Tabulka 1: Popis pinů sběrnice panelu BUSE 110

řádek 1	1	2	řádek 2
řádek 3	3	4	řádek 4
řádek 5	5	6	řádek 6
řádek 7	7	8	řádek 8
řádek 9	9	10	řádek 10
řádek 11	11	12	řádek 12
řádek 13	13	14	řádek 14
řádek 15	15	16	řádek 16
řádek 17	17	18	řádek 18
řádek 19	19	20	
	21	22	
	23	24	
	25	26	
	27	28	
GND	29	30	bit1 skupiny sloupců
Bit2 skupiny sloupců	31	32	bit0 adresy sloupce
bit1 adresy sloupce	33	34	bit2 adresy sloupce
bit0 adresy panelu	35	36	bit1 adresy panelu
bit2 adresy panelu	37	38	enable
bit0 skupiny sloupců	39	40	
	41	42	
24VDC	43	44	24VDC
GND	45	46	24VDC
12VDC	47	48	GND
GND	49	50	5VDC

Ačkoliv slovní popis adresace sloupce displeje uvedený v předchozích odstavcích je věcně správný, může být poměrně matoucí a vyznívá značně komplikovaně. Skutečná adresace je v praxi ale v podstatě jednoduchá. Jak víme, máme panel složený z pěti segmentů s 28 sloupci bodů. Když pro začátek vyjdeme z použití pouze prvního panelu (piny 35-37 nastavíme na logickou „0“ = 0 V), tak konkrétní sloupec vybereme zapsáním pětibitové informace na piny 30-34. Váha bitů se zvyšuje s vyšším číslem pinu. Konkrétně, když zapíšeme „0“ na pin 30, „1“ na pin 31, „0“ na pin 32, „1“ na pin 33 a „0“ na pin 34, zapsali jsme číslo 0b01010, což odpovídá dekadické hodnotě 10. Vybrali jsme tedy 11. sloupec z leva při pohledu na panel z čelní strany. Sloupce jsou totiž adresovány čísly 0-27.

Abychom mohli provést například zápis prvního bodu od spodu ve sloupci 11 musíme dále udělat ještě několik kroků. Na pin 39 přivedeme signál logické „0“ (tímto krokem volíme zápis), čímž jsme na společný kontakt cívek všech bodů ve sloupci 11 připojili zem – pin 45. Na pin 1 (společný kontakt cívek všech bodů v řádku 1) přivedeme 12 V. Zápis ovšem ještě neprobíhá, to v konečné fázi provedeme nahozením pinu 38 na dobu 1 ms. Proudový impulz a následně vytvořené magnetické pole překlopí disk žlutou reflexní stranou ven.

Pokud budeme chtít stejný bod naopak vymazat, změníme jen signál na pinu 39 na logickou „1“ (5V) což bude znamenat, že na společný kontakt cívek všech bodů ve sloupci 11 připojíme 12V (resp. 24) – pin 46. Na pin 1 (společný kontakt cívek všech bodů v řádku 1) přivedeme zem a nahozením pinu 38 provedeme otočení bodu černou stranou ven.

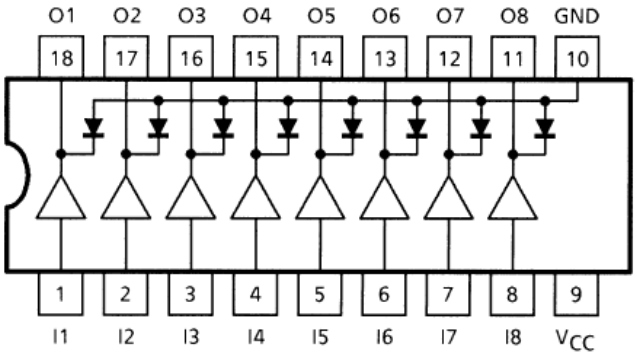
2.3 NÁVRH ŘÍZENÍ PANELU

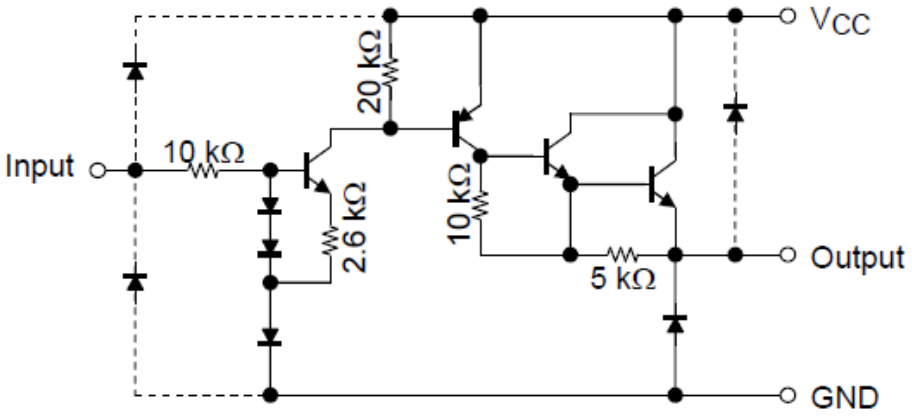
Protože volba sloupce a připojení napětí respektive uzemnění na požadovaný řádek jsou obsluhovány původní elektronikou panelu, bude naším úkolem navrhnout nový způsob řízení panelu s využitím výše popsaného postupu.

Základem této řídící elektroniky budou obvody řady TD62783 a ULN2803. V obou případech se jedná o tranzistorová pole (v jednom obvodu je 8 tranzistorů).

Stručný popis obvodu TD62783 :

Maximální výstupní proud: 500 mA
Maximální spínané napětí: 50V
Logické úrovně: TTL, 5V

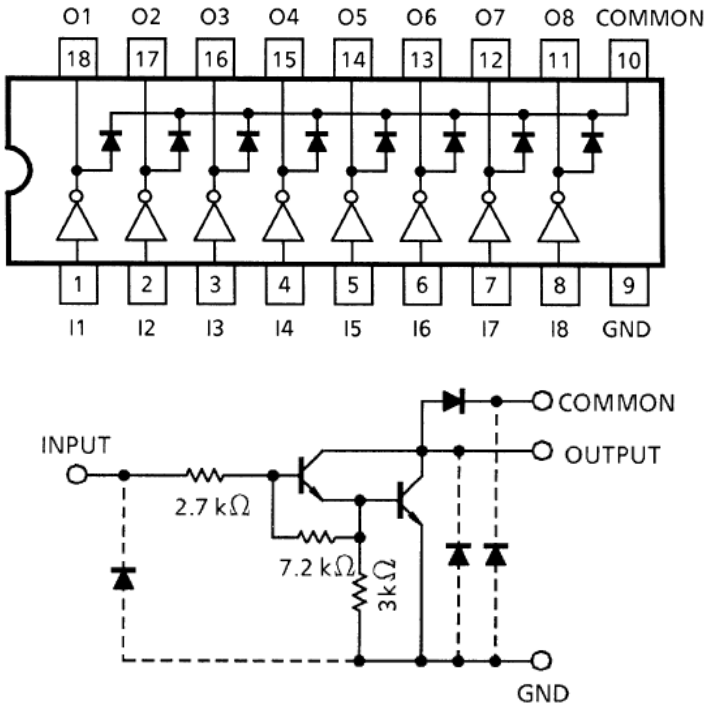




Obr. 5 – zapojení obvodu TD62783

Stručný popis obvodu ULN2803 :

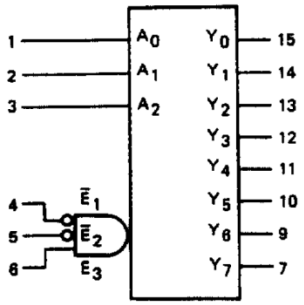
Maximální výstupní proud: -500 mA
Maximální spínané napětí: 50V
Logické úrovně: TTL, 5V



Obr. 6 – zapojení obvodu ULN2803

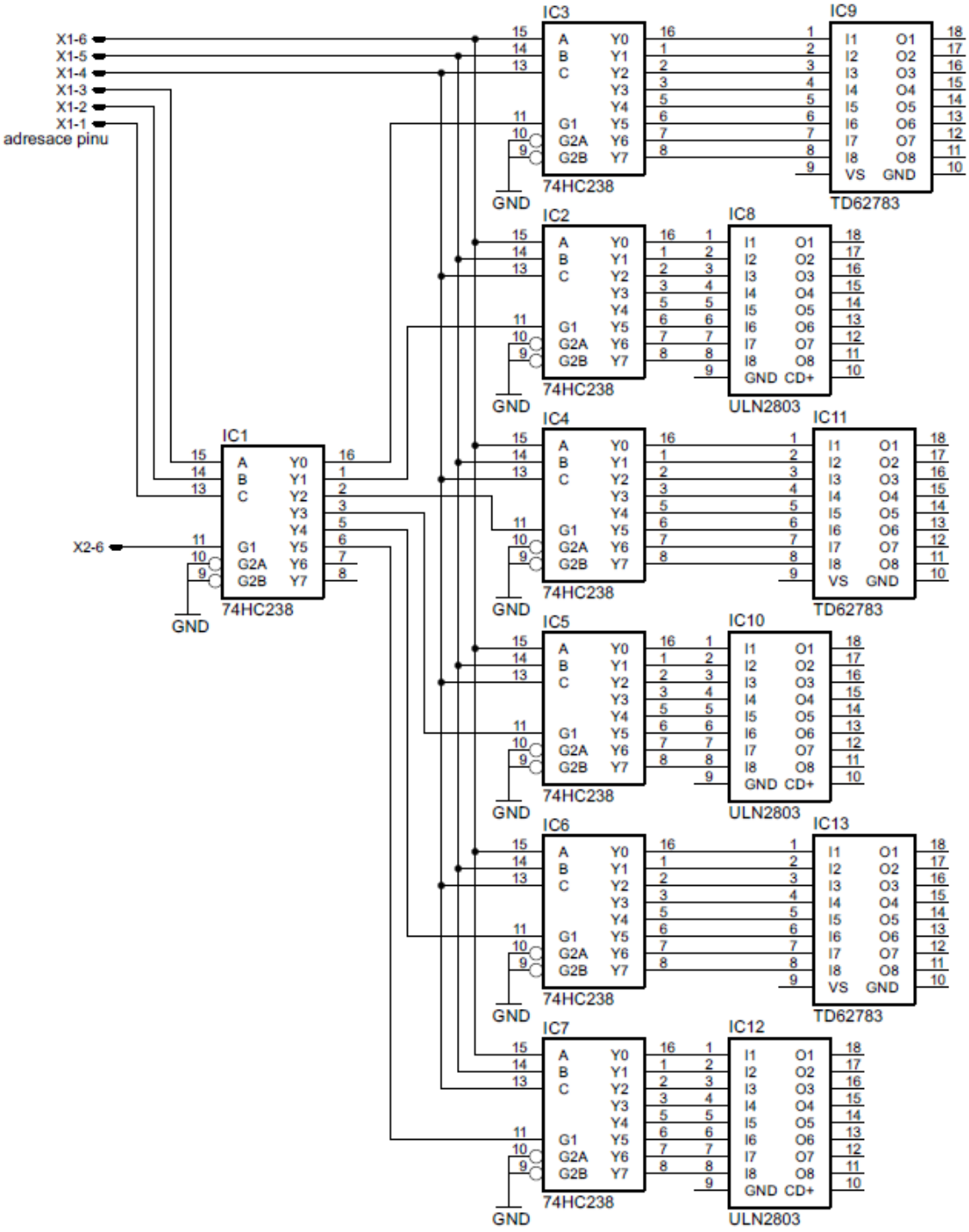
Jak vyplývá z uvedených informací obvody slouží ke spínání zátěže na které může být přivedeno buď kladné napětí až 50 V (obvod TD62783) nebo zem (obvod ULN2803) a snesou proudy až do 500 mA na jeden kanál. Kanály (výstupy) lze samozřejmě paralelně spojovat a navýšit tak maximální proud, což v našem případě není nutné.

Protože uvedené obvody budeme využívat k ovládání společných elektrod na řádcích displeje, kterých je 19, budou nám stačit od každého obvodu tři kusy (dohromady 24 výstupů jednoho typu obvodu). Pro ovládání 19 kontaktů řádku displeje s možností připojit 12-24 V nebo zem musíme mít možnost nezávislého řízení 38 (2 x 19) signálů. To je poměrně velký počet a nese sebou nutnost mít řídící obvod s dostatečným počtem výstupů. Můžeme se ovšem inspirovat provedením samotné původní elektroniky panelu, kde jsou použity multiplexory řady 74HC238, které pomocí tříbitové adresace spínají jeden ze svých osmi výstupů.



Obr. 7 – zapojení obvodu 74HC238

Obvody 74HC238 jsou vybaveny nejen vstupy, které slouží k výběru konkrétního výstupu, ale také celkem třemi signály, které slouží k jeho odblokování. Dva z těchto vstupů (E1, E2) odblokují obvod, pokud je na ně přivedena logická „0“ a vstup E3 funguje stejně při přivedení logické „1“. Toho můžeme využít k zapojení obvodů do „kaskády“. Jeden obvod 74HC238 bude svými výstupy připojen na E3 signál ostatních obvodů (až 8 ks) a ty budou používat společné 3 signály pro adresaci jejich výstupu. Pomocí 6 bitů (3 bity adresace výstupů a 3 bity adresace signálu E3 z hlavního obvodu) tak můžeme adresovat až 64 výstupů (8 obvodů s 8mi výstupy na každém).



Obr. 8 – ideový návrh adresace řádků panelu

Na obrázku 7 můžeme vidět zatím pouhý návrh adresace řádků. Pro úplnost dodávám, že obvody IC9 až IC13 ve schématu musí být připojeny svými vývody 9 a 10 k napájecímu napětí 12-24V, aby celý obvod fungoval. Schéma není úplné kvůli přehlednosti a možnostem zobrazení na stránce A4, což snad nyní ve fázi teoretického rozboru není na závadu. Jak ze schématu vidíme, můžeme řídit všech šest zamýšlených obvodů TD62783 a ULN2803 (každý po třech kusech) pomocí sedmi multiplexorů 74HC238. Na výběr konkrétního výstupu nám stačí 6 samostatných signálů. Signál pro odblokování hlavního multiplexoru (na schématu X2-6) můžeme ovládat sedmým signálem nebo nastálo připojit na logickou „1“.

Na každých 8 řádků displeje bude připojeno 8 výstupů obvodu TD62783 a také 8 výstupů obvodu ULN2803. Jak víme z předchozích odstavců, na každý řádek musíme být schopni připojit 12-24V v případě zápisu bodu nebo zem v případě jeho mazání. Schéma z obrázku 7 pracuje tak, že první pin od shora (X1-6) slouží k výběru, zda budeme zapisovat nebo mazat. Úplně přesně řečeno určuje, zda bude aktivní lichá skupina obvodů nebo sudá. Lichá znamená obvody TD62783, a tedy připojení 12-24V, sudá pak obvody ULN2803, a tedy přizemnění vybraného řádku. To, jaký řádek bude nakonec vybrán k zápisu nebo výmazu určíme dalšími pěti piny konektoru adresace. Váha bitu jednotlivých vstupů na se zvyšuje od X1-5 do X1-1. Protože nám stačí adresovat 19 pinů, ponecháme ty navíc nezapojené.

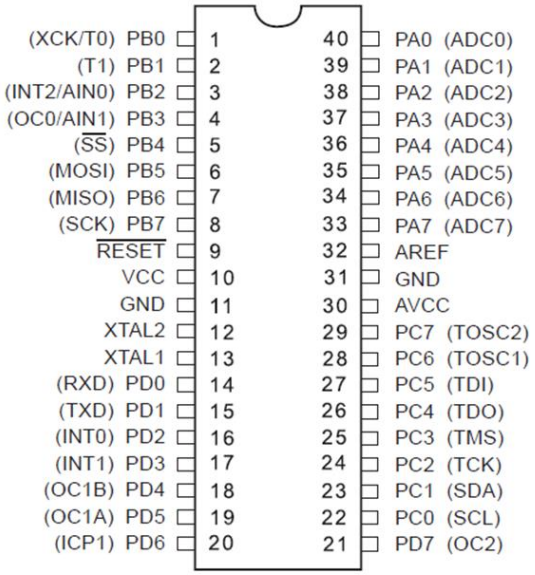
Z výše uvedeného rozboru vyplývá, že pro zápis a výmaz všech bodů panelu BUSE 110 budeme potřebovat řídit deset signálů na sběrnici 30 – 39 k výběru konkrétního sloupce a dále 7 signálů k přivedení požadovaného napětí (12-24V nebo 0V) na vybraný řádek. Celkem tedy potřebujeme mikrokontrolér, který svými výstupy obslouží 17 signálů. I když bychom mohli teoreticky některé signály i sloučit, zdá se mi výhodné mít možnost samostatného řízení, už kvůli časování a návaznosti signálů.

Ke zvolené variantě bych rád připojil poznámku. V průběhu realizace řídících obvodů jsem zvažoval variantu nahradit multiplexory ve formě fyzických integrovaných obvodů sítí naprogramovanou do některého obvodu CPLD. Protože v následujících kapitolách budu navrhovat také řízení těchto signálů na základě povelů odesílaných na informační panel, nabízela se možnost použití obvodu CPLD s naprogramovaným komunikačním rozhraním. V úvahu přicházela možnost využití I2C nebo UART. Takové řešení je technicky možné a poměrně elegantní. Jednoduchý mikrokontrolér s požadovaným počtem vstupů stojí kolem 100 Kč a 7 obvodů 74HC238 pak dohromady asi 35 Kč, což je v porovnání například s cenou obvodů Xilinx při požadovaném počtu vstupů výstupů alespoň 50 (38 pinů pro řádky, 10 řídících signálů pro sloupce a minimálně 2 piny pro komunikaci) cena srovnatelná nebo jen mírně nižší. Pro potřeby výroby prototypu a testování je ale varianta integrovaných obvodů lepší i díky použitým pouzdrům DIL. Navíc jsem se v průběhu projektu chtěl držet řešení realizované v programovacím jazyku C na což ostatně odkazuje i doporučená literatura uvedená v zadání projektu.

2.4 NÁVRH ROZHRANÍ PANELU

V předchozí kapitole jsem si nastínili rozvedení signálů pro zápis a výmaz jednotlivých bodů panelu BUSE 110. Z dosud provedeného rozboru výme, že hardwarovou část panelu můžeme plně ovládat 17 vstupy úrovně TTL. To nám dává dostatečnou informaci pro výběr rozhraní.

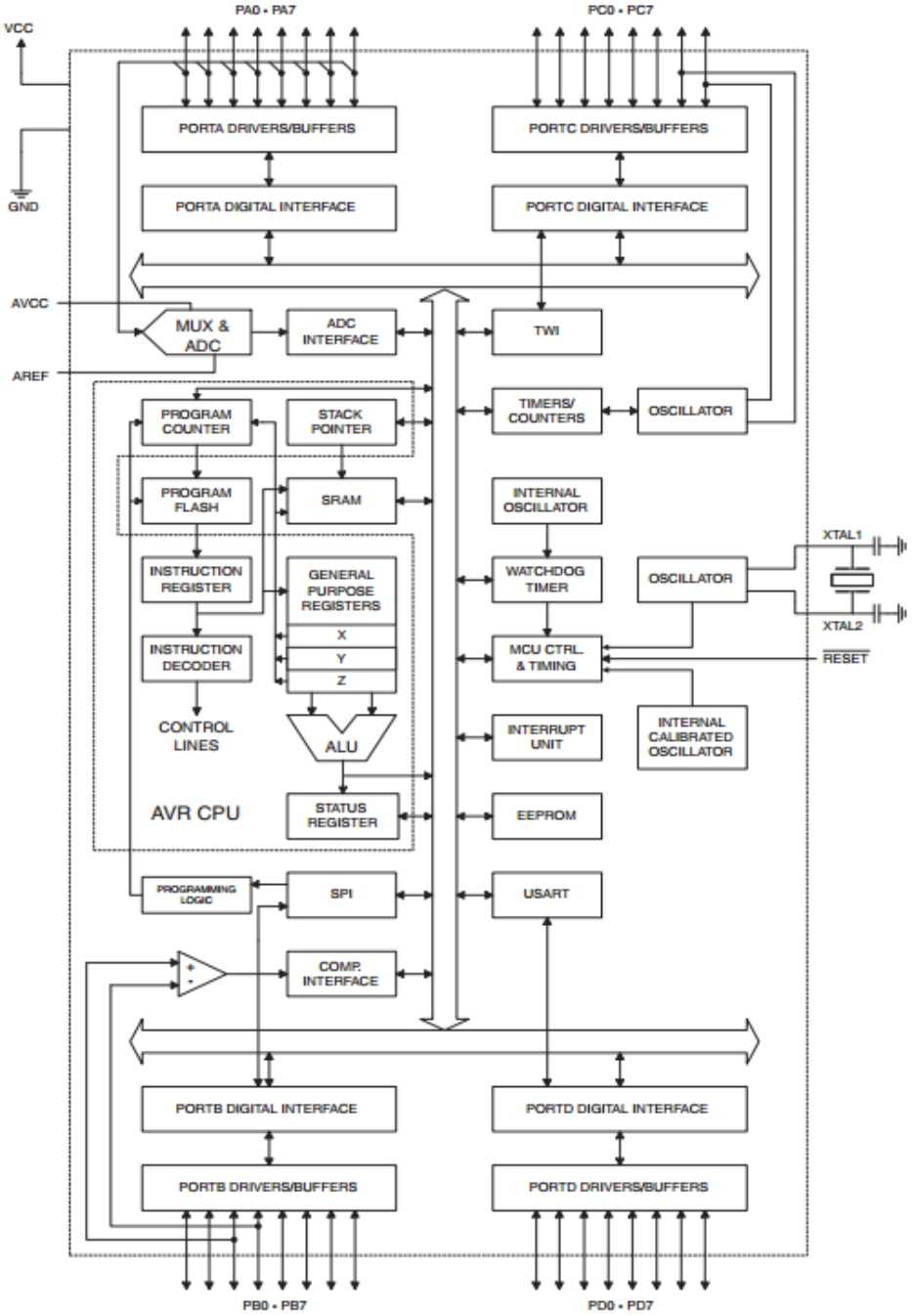
Na základě svých zkušeností s jinými projekty jsem zvolil mikrokontrolér Atmel AVR ATmega32. Mikrokotrolér má dostatek vstupů a výstupů, protože disponuje celkem čtyřmi osmibitovými porty PA-PD, přičemž každý port může být nakonfigurován jako vstupní nebo výstupní.



Obr. 9 –zapojení mikrokontroléru ATmega32

Mikroprocesory ATMega32 je 8 - bitové a používá harvardskou architekturu. To znamená, že má oddělenou paměť programu a dat. ATmega32 má 32kB programové a 2kB datové paměti. Nabízí komunikaci pomocí USART, I2C (TWI), ISP a umožňuje na portu A použít ADC převodník. Obsahuje jeden 16 bitový a dva 8 bitové čítač/časovače s oddělenou děličkou frekvence. Napájecí napětí je 5V.

Mikroprocesor tedy disponuje dostatečným počtem vstupů a výstupů i s malou rezervou na pozdější rozšíření. I při zapojení PA0-PA7, PC0-PC7 a PD7 do obvodu elektroniky panelu, zůstane nám zcela volný port B s programovacím rozhraním ISP a 6 pinů portu D na němž využijeme UART komunikaci. Pro programování mikrokontroléru budeme potřebovat vývojové prostředí AtmelStudio, které je možno získat zdarma ze stránek výrobce procesoru a ISP programátor. Během realizace projektu jsem pracoval v AtmelStudiu 6.1 a použil programátor AVR ISP mkII.



Obr. 10 –blokové schéma mikrokontroléru ATmega32

Zvolená koncepce nízkoúrovňového řízení panelu je tedy následující. Po sběrnici UART přijme mikrokontrolér ATmega32 souřadnice bodu spolu s informací, zda má být bod zaspán nebo smazán. Na základě této informace mikrokontrolér nastaví příslušný řádek a sloupec v požadovaném režimu a dá signál ke spuštění proudového impulsu ke změně stavu bodu. Takové řešení je poměrně jednoduché i robustní. Odpovídá i zadání, aby displej bylo možno řídit graficky. Na druhou stranu to sebou přináší nároky na softwarové vybavení nadřazeného systému, který bude s mikrokontrolérem komunikovat. Zejména s ohledem na tvorbu znaku jako takového, protože na nejnižší úrovni řízení panelu k dispozici znakové sady nemáme.

2.4 WWW ROZHRANÍ

Teoreticky máme vytýčen způsob zápisu a výmazu bodů na panelu BUSE110. Vzhledem k zadání projektu je nutné, aby řídící počítač informačního panelu splňoval dvě podmínky: ethernetové rozhraní a možnost běhu webového serveru. Na doporučení vedoucího semestrální práce jsem hledal mezi mikropočítači s architekturou ARM, kde by byla možnost běhu plnohodnotného operačního systému. Tyto počítače jsou v dnešní době poměrně levné a jejich možnosti použití jsou obrovské. Po určitých pokusech s mikropočítačem Beaglebone jsem zvolil deskový mikropočítač Raspberry Pi. K tomuto rozhodnutí mne vedla jednak příznivá cena (cca 1.000,- Kč) a také poměrně rozsáhlá komunita, která kolem tohoto typu počítače vznikla. Tím je dána i velmi dobrá podpora a dostupnost technických informací.

Obr. 11 – mikropočítač Raspberry Pi

Hardware Raspberry Pi:

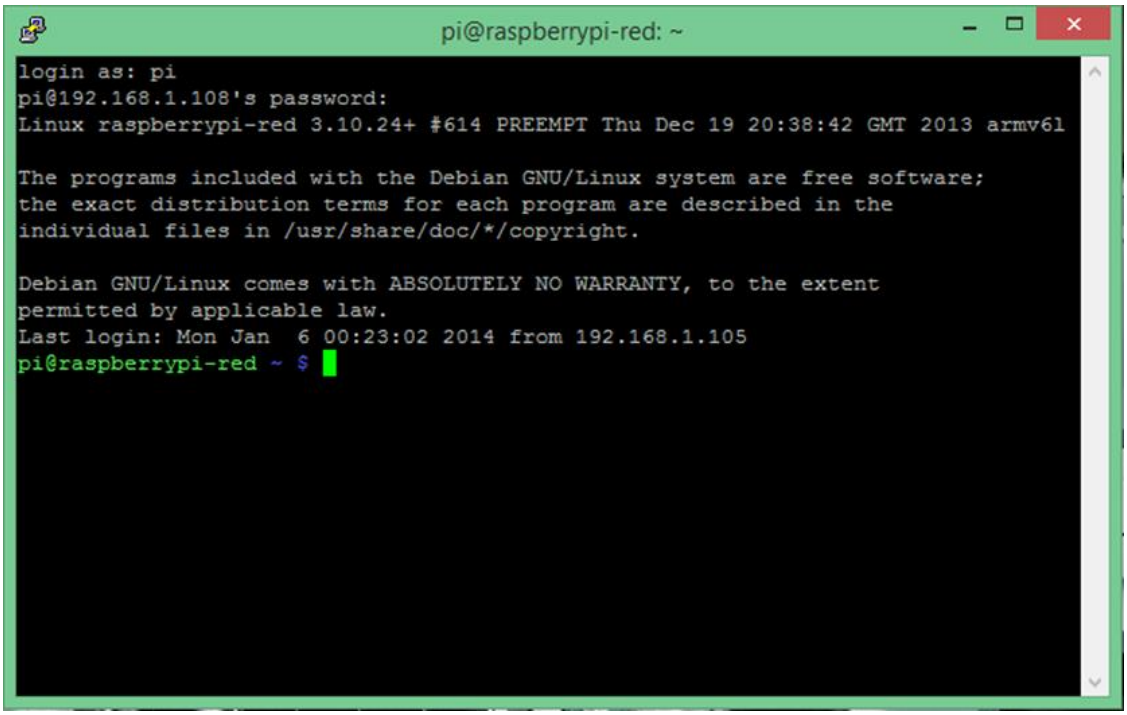
- procesor ARM1176JZF-S z rodiny ARM11 taktovaný na 700 Mhz
- GP VideoCore IV, podporující OpenGL ES 2.0, 1080p30,MPEG-4
- 512 MB RAM sdílených s grafickou kartou
- dva USB porty
- Obrazový výstup Composite RCA, HDMI, DSI
- Zvukový výstup přes 3,5 mm konektor, HDMI
- slot pro SD nebo MMC kartu
- ethernetový adaptér 10/100 s konektorem RJ45
- 17×GPIO(UART, I²C, sběrnice SPI)

Vzhledem k přítomnosti sběrnice UART máme teoreticky vyřešenu komunikaci s panelem resp. mikrokontrolérem ATmega32. Díky možnosti využití I2C a také volným GPIO získáváme velkou volnost pro připojení dalších zařízení. Toho využijeme k připojení čidel pro měření parametrů interiéru.

Protože je Raspberry Pi díky platformě ARM schopen spuštění linuxového jádra, máme k dispozici „dospělý“ počítač. Na mikropočítači budeme používat specializovanou distribuci Raspbian, která je derivací populární distribuce Debian.

Výběr je ale rozhodně širší a v současné době je možné volit mezi několika OS s linuxovým jádrem určeným přímo pro Raspberry Pi. Raspbian je ale rozhodně vhodnou volbou s širokými možnostmi. Protože jde o plnohodnotný linuxový systém, zcela nám odpadnou starosti s vývojovým prostředím a volbou programovacího jazyka. Ačkoliv je komunita kolem Raspberry Foundation hodně zaměřena na použití programovacího jazyka Python, obsahuje Raspbian GNU kompilátor GCC verze 4.6.3 a můžeme tedy ihned psát a kompilovat programy v programovacím jazyku C.

Protože počítáme s instalací celé elektroniky dovnitř panelu BUSE 110, nepředpokládám, že bude Raspberry Pi používán s připojeným monitorem a klávesnicí. I když to umožňuje a většina uživatelů ho tak určitě používá, my se zaměříme na jeho ovládání přes SSH z počítače připojeného na stejné místní síti s protokolem Ethernet. Na vzdálenou správu Raspberry Pi jsem použil program PuTTY, který je Open Source (šířen pod licencí MIT). Pro připojení k Raspberry stačí znát IP adresu, která je Raspberry přidělena routerem.

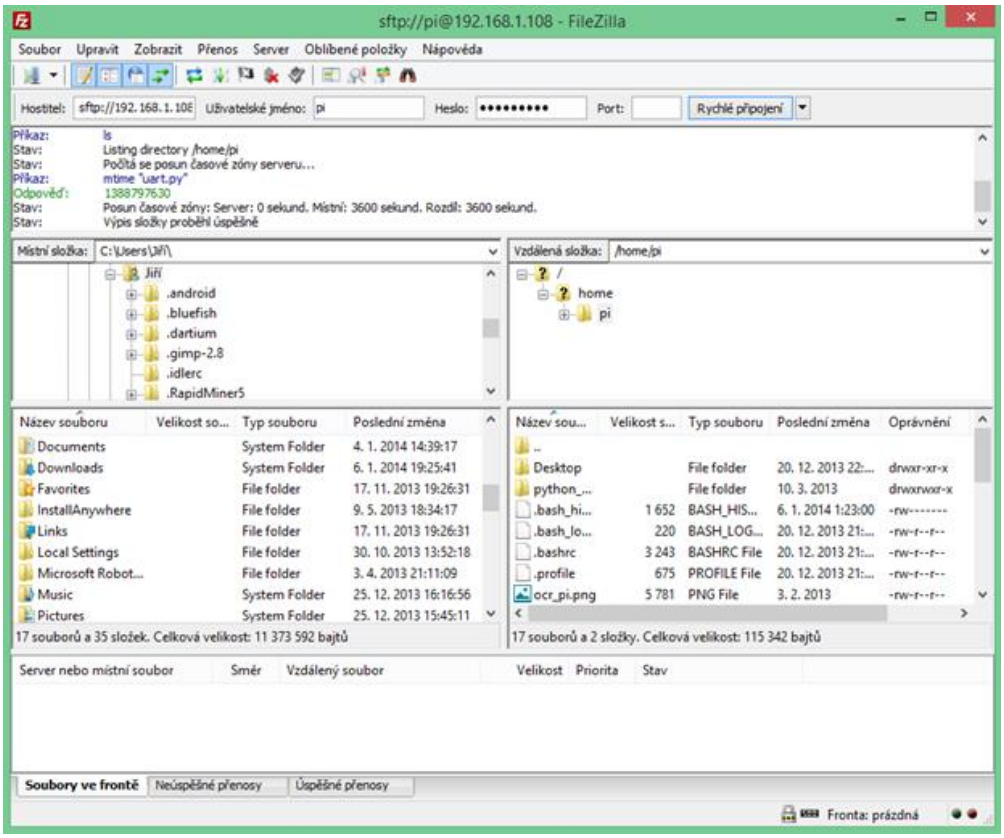


Obr. 12 spuštěný program PUTTY s obrazovkou systému Raspbian v textovém režimu.

V příkladu uvedeném na obrázku 12 byla Raspberry Pi přidělena IP adresa 192.168.1.108. V originální konfiguraci je v Raspbianu nastaven uživatel *pi* s přihlašovacím heslem *raspberry*. Po vytvoření SSH spojení s těmito údaji máme plnou kontrolu nad mikropočítačem Raspberry Pi. Můžeme se samozřejmě rovnou připojit se superuživatelským právy. Pak místo uživatele *pi* zadáme *root*. Heslo je v obou

případech stejné. Nicméně téhož dosáhneme spuštěním příkazu su v příkazové řádce a asi není úplně vhodné podnikat pokusy s právy superuživatele.

Pro pohodlnější vzdálenou práci s počítačem Raspberry Pi můžeme dále využít instalovaného ftp serveru a například open source programu Filezilla pro přenášení souborů mezi Raspberry Pi a naším místním počítačem. Připojovací údaje jsou stejné a i zde můžeme zvolit superuživatelská práva. Vzhledem k možnosti přímého přístupu do složky //var/www/ pro editaci to dává o něco větší smysl.



Obr.13: spuštěný program Filezilla s připojeným souborovým systémem Raspberry Pi

Jedním z hlavních důvodů využití mikropočítače na platformě ARM11 s linuxovým jádrem je možnost spuštění a provozu síťového serveru. Tato funkce není u Rasperry Pi s instalovaným systémem Raspbian splněna automaticky, nicméně je možné snadno doinstalovat potřebný software. Pro provozování potřebných funkcí existuje hned několik variant, například http server Lighttpd s SQL databází Sqlite. Já jsem nakonec zvolil řešení, které je v linuxovém prostředí známé pod výrazem LAMP, neboli Linux, Apache, MySQL a PHP. Linuxové jádro už přirozeně instalované máme a další tři nástroje nainstalujeme ze standardního repozitáře Raspbianu. Ke konkrétní instalaci a zprovoznění softwarových nástrojů se vrátíme v kapitolách věnovaných sestavení a zprovoznění portálu.

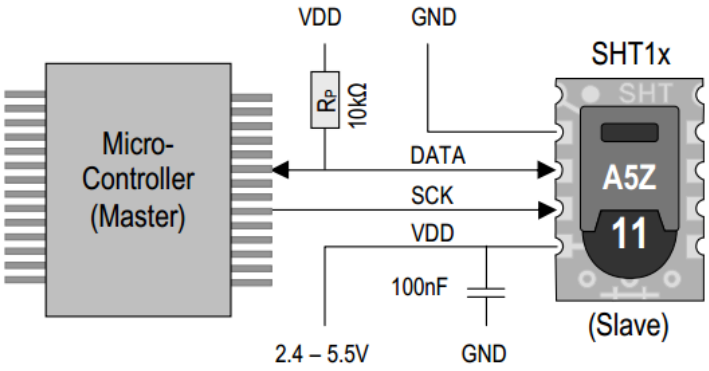
2.5 FUNKCE METEOSTANICE

Moderní meteostanice dnes nabízejí mnoho funkcí. Vedle zobrazení venkovní a vnitřní teploty a vlhkosti vzduchu, umožňují i ukládání dat a zobrazení trendů. Na základě vývoje atmosférického tlaku je možná i prognóza vývoje počasí na následujících několik hodin.

Vzhledem k zadání, kde je výslovně uvedena funkce snímání parametrů interiéru, nemá funkce měření atmosférického tlaku velký význam. Přesto tuto funkci pro zajímavost zprovozníme. Logování údajů z interiéru není příliš zajímavé z hlediska prognózy počasí, ale systém by měl takovou funkci umožňovat. Pro účely projektu postačí vybavit systém portálu jednoduchým čidlem pro měření teploty a relativní vlhkosti. Toto zadání splňuje například čidlo SHT11 od společnosti Sensirion (viz. Obr.1).

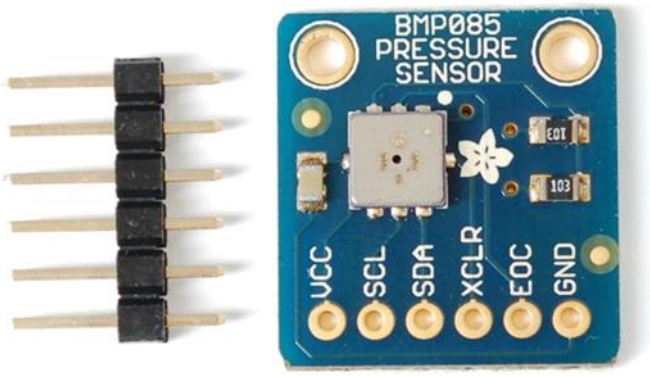
Obr. 14: Čidlo SHT11

Toto čidlo umožňuje měření relativní vlhkosti v rozsahu 0 až 100% s přesností +/- 3% a teploty v rozsahu -40 až +125 °C s přesností +/- 0,4 °C. Ke komunikaci je použito dvouvodičové digitální připojení. Rozhraní je podobné I2C, ale není kompatibilní. Se sběrnicí I2C je možné sdílet hodinový signál SCK, ale signál DATA vyžaduje samostatný vstup. Přesněji řečeno, čidlo může být s mikrokontrolérem spojeno přes I2C port, ale bez interference s jiným zařízením na téže sběrnici I2C. My ho budeme používat s připojením k Raspberry Pi přes některý z GPIO, kterých máme k dispozici dostatek. Energetická spotřeba čidla je 90μW (12bit, 3V, 1 měření / s) a dynamická odezva čidla relativní vlhkosti je 8 sekund (při 63% τ).



Obr. 15: doporučené zapojení čidla SHT11

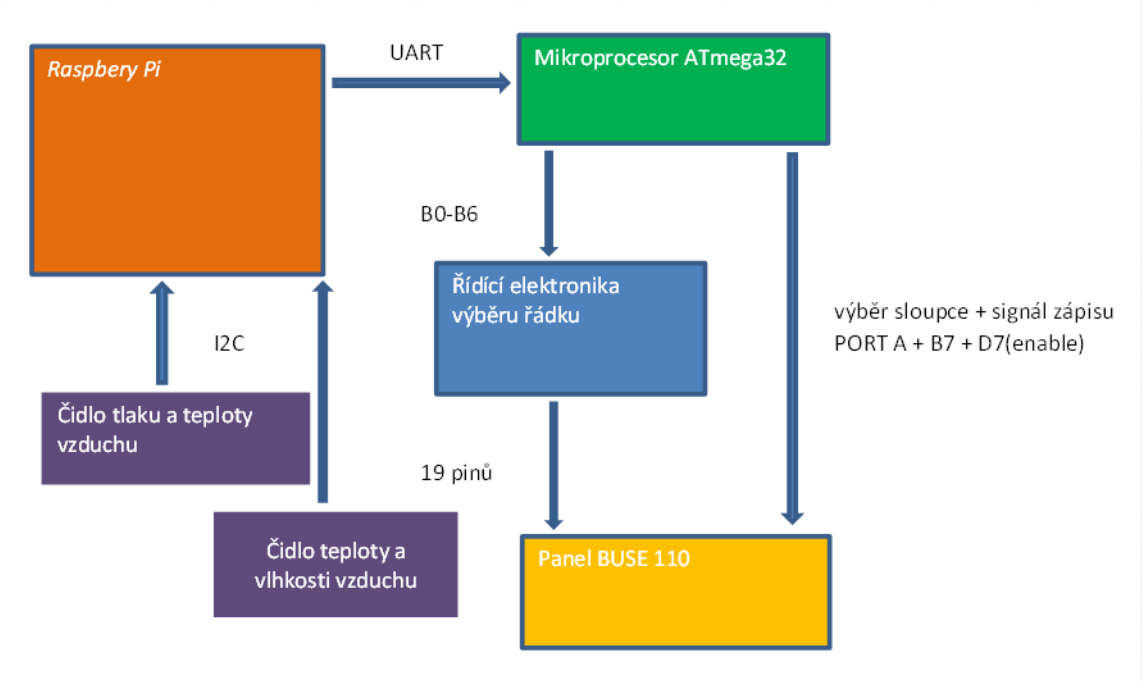
Pro možnost vyzkoušení kompatibility a všech funkcí sběrnice I2C jsem se rozhodl využít i čidlo BMP085. BMP085 měří teplotu a tlak (300-1100hPa) a je plně kompatibilní se sběrnicí I2C. Pro testovací účely jsem zatím používal pouze toto čidlo.



Obr. 16: Čidlo BMP085 (Bosch)

Čidlo BMP085 měří teplotu v rozsahu 0 - 65 °C s přesností +/-1 °C. To je podstatně horší přesnost, než máme u SHT11, proto použijeme toto čidlo pouze pro měření atmosférického tlaku, kde v rozsahu 300 – 1100 hPa při teplotě 0 - 65 °C dosahuje typické absolutní přesnosti +/- 1 hPa.

2.6 KONCEPCE



Obr.17: Blokové schéma informačního portálu

Na obrázku 17 je zobrazen celý návrh informačního portálu z pohledu propojení funkčních celků a komunikačních rozhraní. Zatímco mikropočítač Raspberry Pi sbírá data přes Two-Wire interface a I2C, následně je pak v grafické podobě posílá k vykreslení přes UART mikrokontroléru ATmega32, mikrokontrolér samotný po přijmutí dat už vše obsluhuje pomocí signálů úrovně TTL na řídících pinech.

2.7 POPIS ROZHRANÍ ETHERNET

Jedná se o celý soubor technologií pro LAN (lokální počítačové sítě) pro komunikaci přenosovými rychlostmi od 10Mbit/s do 10Gbit/s. Problematiku popisuje standard IEEE 802.3. Pro fyzické propojení se používají kabely s kroucenou dvoulinkou nebo i optické kabely. V referenčním modelu OSI realizuje Ethernet fyzickou a linkovou vrstvu. Na sítích typu Ethernet je tedy možno provozovat více protokolů síťové vrstvy z nichž nejvíce se používají protokoly IP a IPv6, které jsou využívány pro síť Internet.

Již v devadesátých letech minulého století získal Ethernet pozici nejpoužívanější technologie pro lokální sítě. Rozhraní Ethernet prochází neustálým vývojem (přechod od koaxiálního kabelu ke kroucené dvoulince) a neustálým zvyšováním přenosové rychlosti:

- 1 Mbit/s –první varianty Ethernetu pro kroucenou dvoulinku standardizované jako IEEE 802.3e v roce 1986
- 10 Mbit/s – klasický Ethernet –definován pro koaxiální kabel, kroucenou dvojlinku a optické vlákno.
- 100 Mbit/s – Fast Ethernet – rychlejší verze s přenosovou rychlostí 100 Mbit/s definovaná standardem IEEE 802.3u. Převzala maximum prvků z původního Ethernetu (formát rámce, algoritmus CSMA/CD apod.), aby se usnadnil, urychlil a zlevnil vývoj. V současnosti ji lze považovat za základní verzi Ethernetu. Je k dispozici pro kroucenou dvojlinku a optická vlákna.
- 1 Gbit/s – Gigabitový Ethernet – zvýšil přenosovou rychlost na 1 Gbit/s. V praxi je gigabitový Ethernet provozován přepínaně s plným duplexem. Původně pouze pro optická vlákna (IEEE 802.3z), později byla doplněna i varianta pro kroucenou dvojlinku (IEEE 802.3ab).
- 10 Gbit/s – Desetigigabitový Ethernet –zatím poslední standardizovaná verze. Standard IEEE 802.3ae přijat v roce 2003. Přenosová rychlost činí 10 Gbit/s, jako médium slouží hlavně optická vlákna. Opět používá stejný formát rámce. Algoritmus CSMA/CD byl opuštěn, tato verze pracuje vždy plně

Pro přístup ke sdílenému přenosovému médiu (sběrnici) se používá metoda CSMA/CD (Carrier Sense with Multiple Access and Collision Detection) neboli metoda mnohonásobného přístupu s nasloucháním nosné a detekcí kolizí. Stanice, která se chystá vysílat, sleduje situaci na přenosovém médiu. Stanice začne vysílat, pokud na přenosovém médiu nevysílá žádná další stanice. Pokud dvě stanice začnou vysílat ve stejný okamžik, jejich signály budou v kolizi. Vysílající stanice v takovém případě zjistí příchod cizího signálu. Stanice, která detekuje kolizi, vyšle krátký signál (jam o 32 bitech). Na to se všechny vysílající stanice odmlčí a po uplynutí náhodné doby učiní nový pokus o vysílání. Náhodně vybíraná čekací doba se během prvních deseti pokusů vždy zdvojnásobí. Zvyšuje se tak pravděpodobnost, že se o sdílené médium stanice

úspěšně podělí s ostatními. Pokud se ani během šestnácti pokusů nepodaří rámec odvysílat, stanice své snažení ukončí a ohlásí nadřízené vrstvě neúspěch.

Formát rámce se popisuje pomocí oktetů, což je osmice bitů. Důvodem je přesnost definice, protože některé počítače mohou pracovat s jinou základní délkou bajtu (např. 4 nebo 10 bitů), což by v počítačových sítích způsobovalo nekompatibility. Níže uvedená tabulka popisuje rámec Ethernet II a 802.3, které se liší využitím jednoho pole pro typ nebo pro délku (vysvětlení je pod tabulkou).

Tabulka 2: Ethernetový rámec (wikipedia)

Preambule	SFD	MAC cíle	MAC zdroje	Typ/délka	Data a výplň	CRC32	Mezera mezi rámci
7× oktet 10101010	1× oktet 10101011	6 oktetů	6 oktetů	2 oktety	46-1500 oktetů	4 oktety	12 oktetů
		64-1518 oktetů					
72-1526 oktetů							

V současné době je naprostá většina běžných spotřebních zařízení, která lze připojit na drátovou síť (počítače, notebooky, routery, herní konzole, atd) vybavena konektorem P8C (RJ-45) pro síť Ethernet

Obr.18 konektor RJ-45

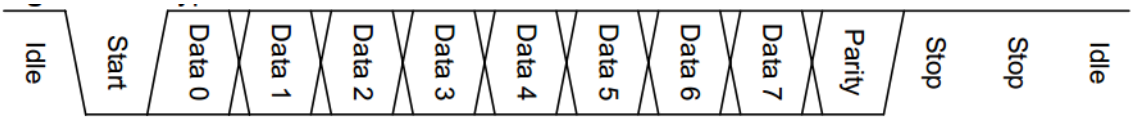
Tímto konektorem (RJ-45) je vybaven i mikropočítač Raspberry Pi. Mikropočítač zároveň umožňuje komunikaci po síti Ethernet až rychlostí 100Mbit/s (Fast Ethernet).

2.8 POPIS ROZHRANÍ UART

UART znamená univerzální asynchronní přijímač/vysílač. Je to hardwarové zařízení, které pomocí dvou pinů (označovaných jako RX a TX) odesílá a přijímá data po dvouvodičové lince. Protože se jedná o asynchronní způsob komunikace, obsahuje přijímač i vysílač vlastní generátor hodinového signálu. A jelikož je UART univerzální, je také možné rychlost těchto hodin řídit (není stanovena), stejně jako velikost jednoho bajtu, počet stop bitů, paritní bit, atd.

Často můžeme se používá také termín UART. Jedná se v podstatě o to samé, pouze s tím rozdílem, že je USART obsahuje i synchronní režim.

Jak vyplývá z výše uvedeného, UART se používá pro sériový přenos dat. Komunikační rámec obsahuje vždy jeden start bit, dále se přenese 5-8 datových bitů (typicky 8), parita (lze nastavit žádná, sudá, lichá) a stop bity (může být jeden nebo dva).



Obr.19: průběh UART rámce (Atmel)

Protože budeme UART rozhraní používat pro komunikaci mezi Raspberry Pi a AVR ATmega32, musíme použít stejné nastavení pro obě zařízení.

V případě ATmega32 může konfigurace zařízení vypadat takto:

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

// nastaveni rychlosti v BAUDech a vypocet prescale hodnoty pro registr UBRR
#define USART_BAUDRATE 9600
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)

/*************************************************************************
        Inicializace USART nahozeni prijimani i odesilani
*************************************************************************/
void _init_usart() {
        UBRRL = BAUD_PRESCALE;/ * nastaveni nizsich 8mi bitu hodnoty
BAUD_PRESCALE v nizsim bytu UBRR registeru */
        UBRRH = (BAUD_PRESCALE >> 8); /* nastaveni vyssich 8mi bitu hodnoty
BAUD_PRESCALE ve vyssim bytu UBRR registeru */
        UCSRB = 0x18; // zvolen transmitter i reciever
        UCSRC = 0x86; // 8 bitova data, zadny parity bit a 1 stop bit.
}
```

Protože použijeme externí hodinový signál řízený krystalem 8MHz, je důležité nastavit správně rychlost CPU (F_CPU). Nastavení rychlosti UART rozhraní se

provede zápisem 16bitového čísla do UBRR registru. Toto 16 bitové číslo získáme výpočtem vzorce:

UBRR=((F_CPU/(USART_BAUDRATE*16))-1)

V našem případě to tedy bude ((8000000/(9600*16))-1) = 51. Je poměrně důležité, že nám vyšlo celé číslo. Pro některé rychlosti F_CPU a zvolené rychlosti UART může vyjít hodnota uložená do registru UBRR a sloužící k vytvoření synchronizačního signálu UART z hodinového signálu CPU jak desetinné číslo. To je pak z principu zaokrouhleno a mezi zvoleným a skutečným synchronizačním signálem UARTvznikne rozdíl. To má samozřejmě negativní vliv na chybovost provozu komunikačního rozhraní .

Další nastavení komunikačního rámce můžeme provést nastavením registrů UCSRB a UCSRC.

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Obr.20 : registr UCSRB (Atmel)

Popis registru UCSRB:

- RXCIE = povolení přerušení v případě kompletního příjmu dat
- TXCIE = povolení přerušení v případě kompletního odvysílání dat
- UDRIE = povolení přerušení v případě prázdného datového registru
- RXEN povolen pro příjem z UART
- TXEN = povolen pro zápis na UART
- UCSZ2= v kombinaci z UCSZ1 aUCSZ0 z registru UCSRC určuje počet datových bitů

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

Obr.21 : registr UCSRC (Atmel)

Popis registru UCSRC:

- UMSEL = volba mezi synchronním a asynchronním režimem
- UPM1:0 = nastavení zda je použita parita a jaká („00“= bez parity)
- USBS = určuje počet stop bitů
- UCSZ1:0 = v kombinaci z UCSZ2 z registru UCSRB určuje počet datových bitů
- UCPOL = určuje polaritu hodinového signálu v synchronním režimu

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Obr.22 : registr UCSRA (Atmel)

Popis registru UCSRC:

- RXC: stav dokončení příjmu
- TXC: stav dokončení vysílání
- UDRE: datový registr je prázdný
- FE: chyba rámce
- DOR: přetečení
- PE: chyba parity
- U2X: v asynchronním módu zdvojnásobí rychlost UART
- MPCM: povolí více zařízení, zprávy bez adresy zařízení budou ignorovány

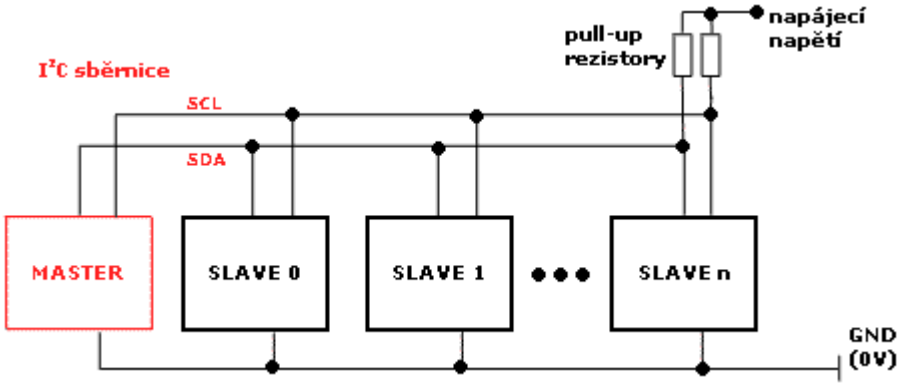
Do registru UCSRA jsme nezapisovali, jedná se z větší části o stavový registr a jeho jedinné dva konfigurační bity k ničemu nepotřebujeme.

Do registru UCSRB jsme zapsali hodnotu 0x18 = 0b00011000, takže jsme port iniciovali pro čtení a zápis.

Do registru UCSRC jsme zapsali hodnotu 0x86 = 0b10000110, takže jsme port iniciovali pro přenos 8 datových bitů, žádnou kontrolu parity a jeden stop bit. Stejně musíme postupovat při konfiguraci sériového portu na mikropočítači Raspberry Pi. Na Raspberry Pi použijeme také jazyk C, ale využijeme strukturu TERMIOS. Termios znamená Terminal Input Output Interfaces. Příklad programového kódu jsem uvedl v kapitole 3.3

2.9 POPIS ROZHRANÍ I2C

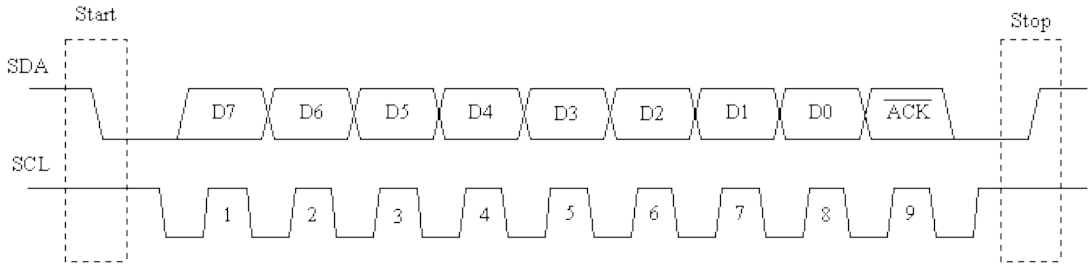
Název I2C vznikl z pozměněného zápisu zkratky IIC , neboli Internal Integrated Circuit Bus. I2C vyvinula společnost Philips zhruba před 20 lety. Jak název napovídá, rozhraní je zpravidla integrované na čipu a slouží pro komunikaci mezi moduly, a to většinou v rámci jednoho zařízení. V současnosti se dají sehnat součástky, které rozhraním disponují a pokrývají celou škálu použití. Jedná se o senzory, rozšiřující vstupy a výstupy, paměťové obvody, A/D a D/A převodníky, obvody reálného času, ale také displeje. Většinou je na jedné sběrnici jedno master zařízení, ale může fungova i s více mastery. Každé zařízení má svou individuální adresu. Adresy jsou určeny 7 bitovým číslem, což znamená, že na sběrnici může být současně až 128 zařízení. Existuje možnost v rozšířené podobě adresovat až 10 bity. Pak vzroste počet připojitelných zařízení až na 1024.



Obr. 23: zapojení sběrnice I2C

I2C využívá linku SDA pro přenos dat a linku SCL pro synchronizaci hodinovým signálem. Obě linky musí být vybaveny pull-up rezistorem. Řízení sběrnice probíhá na základě přesně definovaných situací, které určují začátek a konec přenosu a probíhající vysílání.

- SDA i SCL jsou v úrovni „HIGH“ – klidový stav, sběrnice je neaktivní
- SDA je LOW, SCL je HIGH – SDA byl „stažen“ masterem, což indikuje start
- SDA přejde z LOW na HIGH, SCL je HIGH – přenos dat, vysílač posílá 8 datových bitů, přenos začíná bitem s největší vahou. Bity se „posouvají“ s hodinovým impulsem na SCL.
- SDA je LOW (přijímač potvrzuje příjem) dokud master nevyšle devátý impuls na SCL

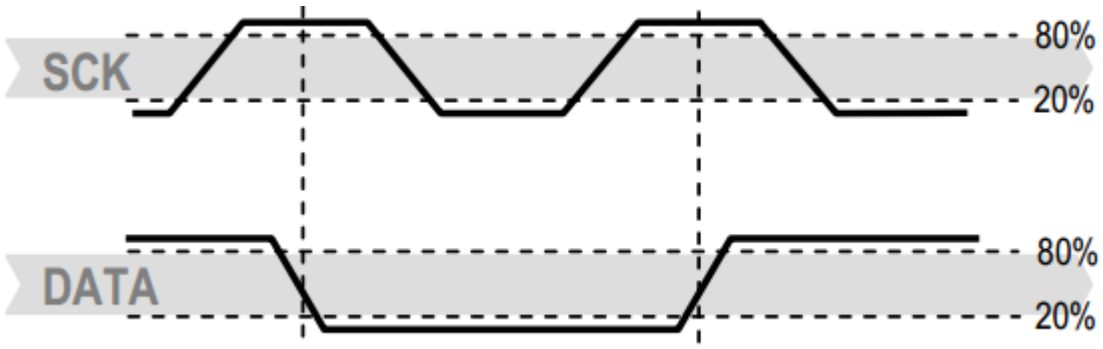


Obr. 24: průběh signálů na sběrnici I2C

V našem případě bude masterem mikropočítač Raspberry Pi a na sběrnici I2C budeme mít připojeno jedinné slave zařízení, číslo Bosch BMP085. Adresa čidla je 0x77. V kapitole 3.2 se budeme věnovat zprovoznění sběrnice na mikropočítači Raspberry Pi a uveden bude i krátký příklad kódu.

2.10 POPIS ROZHRANÍ SHT11 (two wire)

Čidlo SHT11, které v projektu rovněž použijeme není kompatibilní se sběrnicí I2C, ale používá podobnou digitální sběrnici označovanou jako two-wire. Sběrnice je svou funkcí I2C podobná, ale má omezení. Začátek vysílání je stejný, jako v případě I2C, datový signál přejde z HIGH do LOW, zatímco hodinový signál zůstává HIGH. Následuje vysílání 8 bitů, synchronizovaných hodinovým signálem. První tři bity obsahují adresu zařízení (v případě čidla Sensirion SHT-11 je adresa „000“ a další bity nesou informaci o zaslaném příkazu.



Obr. 25: začátek komunikace na sběrnici Two-wire (Sensirion)

Tabulka 3: seznam příkazů pro čidlo Sensirion SHT-11

Command	Code
Reserved	0000x
Measure Temperature	**00011**
Measure Relative Humidity	**00101**
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values. Wait minimum 11 ms before next command	**11110**

Z tabulky 3 vidíme, že seznam příkazů je velmi jednoduchý. Nás samozřejmě nejvíc zajímají pokyny pro zaslání dat o naměřené teplotě a vlhkosti. Pro komunikaci se senzorem použijeme volně dostupné knihovny funkcí RPi_SHT1x.c a RPi_SHT1x.h, stejně jako ovladač pro GPIO piny mikropočítače Raspberry Pi bcm2835.h.

3 REALIZACE

3.1 INSTALACE SYSTÉMU RASPBERRY PI

Mikropočítač Raspberry Pi je v základní verzi dodáván bez operačního systému. Jakkoliv je jeho zprovoznění triviální, pro naše použití bude základní nastavení mikropočítače vyžadovat doinstalování softwaru a nakonfigurování některých systémových služeb.

Samotný mikropočítač od prodejce dorazí bez paměťové karty a bez jakéhokoliv programového vybavení. Stačí však jakákoliv paměťová karta formátu SD velikosti alespoň 4GB (doporučená je 8 GB) a třídy 4 (rychlost zápisu nejméně 4 MB/s), PC s internetovým připojením a příslušnou čtečkou karet Secure Digital k tomu, abychom Raspberry Pi zprovoznili. Na stránkách www.raspberrypi.org je v sekci Download odkaz na obraz disku nejnovější verze systému Raspbian. Tento obraz ve formátu *.iso stáhneme do našeho počítače a dle návodu si v prostředí Windows stáhneme program Win32DiskImager. Program je šířen pod licení GNU GPL. Po spuštění programu vybereme obraz disku z místa, kam jsme ho uložili a zvolíme kartu k jeho zápisu. V prostředí Linuxu můžeme použít ImageWriter nebo utilitu dd.

Po zapsání obrazu na kartu můžeme kartu zasunout do slotu Raspberry Pi, připojit Raspberry k síti ethernet a k napájecímu napětí (microUSB konektor). Systém za několik sekund naběhne. Pro prvotní spuštění může dávat smysl připojit Raspberry Pi k monitoru (hdmi) a klávesnici a prvotní nastavení provést v GUI linuxu. Na druhou stranu poslední verze Raspbianu vydána 7.1.2014 fungovala ihned s možností připojení přes SSH.

3.2 ZPROVOZNĚNÍ I2C NA RASPBERRY PI

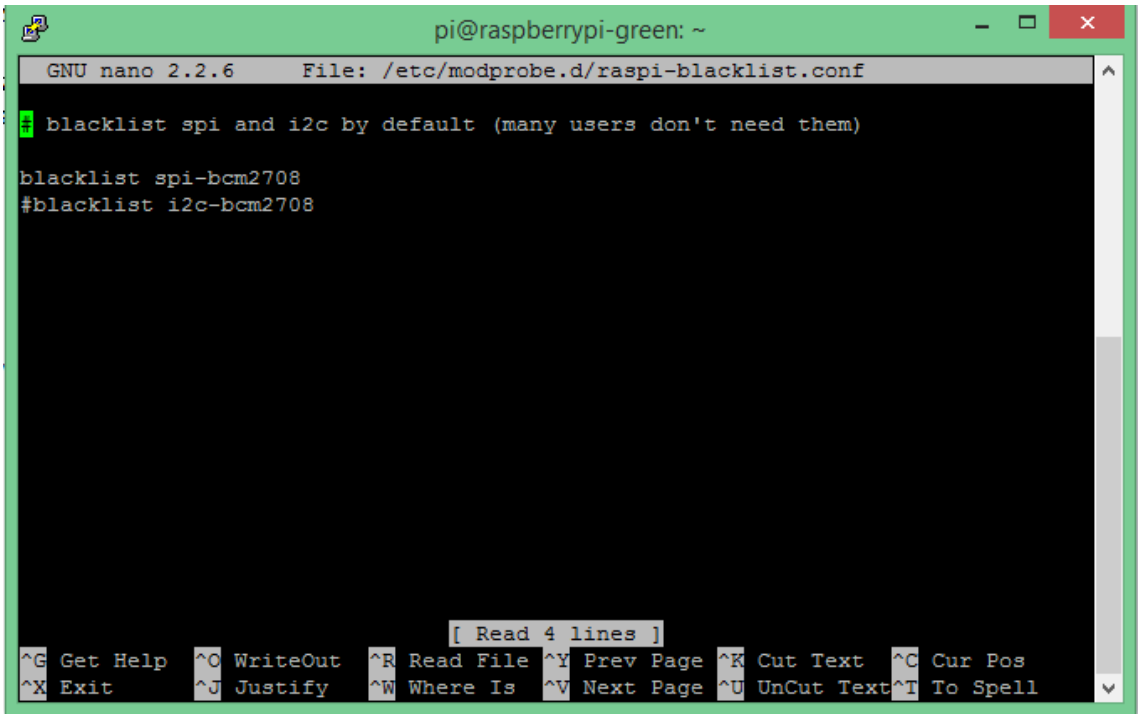
Po čisté instalaci systému je rozhraní I2C na mikropočítači Raspberry Pi blokováno. Příslušný ovladač systému je uveden v textovém konfiguračním souboru **/etc/modprobe.d/raspi-blacklist.conf**. Toto změníme editací souboru příkazem:

sudo nano /etc/modprobe.d/raspi-blacklist.conf

Výpis souboru vypadá takto:

```
# blacklist spi and i2c by default (many users don't need them)

blacklist spi-bcm2708
blacklist i2c-bcm2708
```

Zprovoznění I2C dosáhneme zakomentováním řádku s I2C tak, aby výsledek vypadal jako na obrázku 26. Editaci ukončíme stiskem kláves CTRL-X a následným potvrzení zápisu klávesou Y.

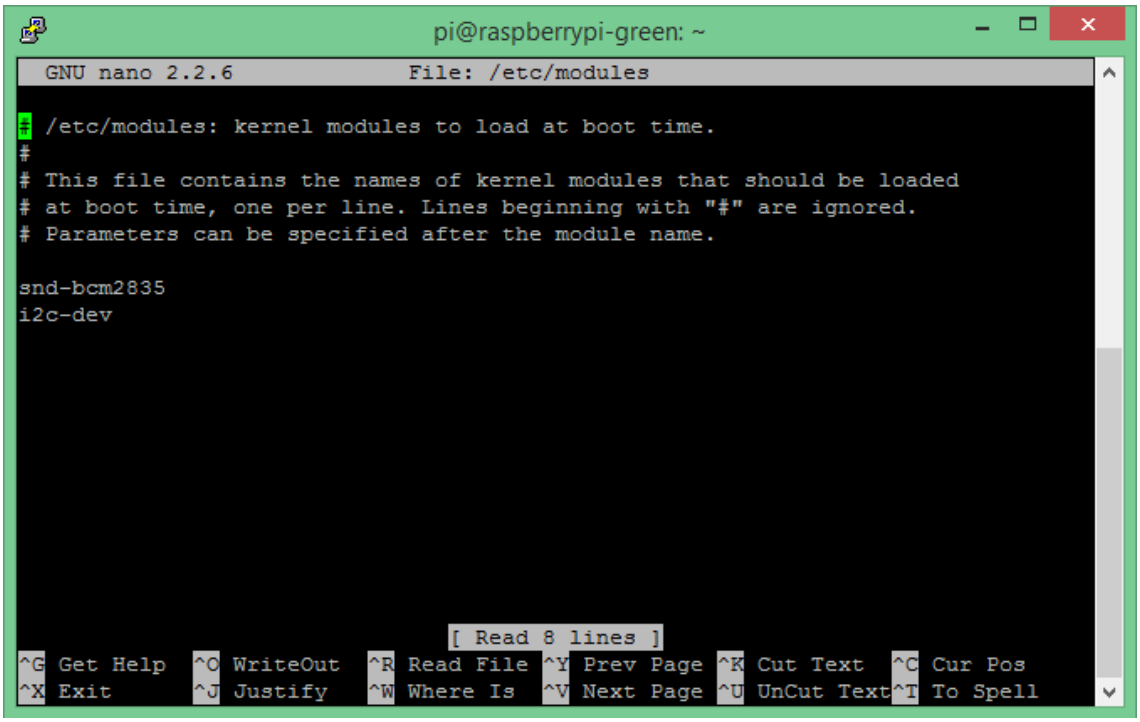


Obr.26: žádoucí obsah souboru /etc/modprobe.d/raspi-blacklist.conf

V dalším kroku zavedeme ovladač sběrnice I2C do jádra systému. Tento krok provedeme zanesením modulu do konfiguračního souboru. Jeho editaci provedeme příkazem:

sudo nano /etc/modules

Na konec souboru zapíšeme řetězec i2c-dev a opět editaci ukončíme a potvrdíme zápis.



Obr.27 žádoucí obsah souboru /etc/modules

Po těchto krocích ještě doinstalujeme softwarové nástroje pro práci s rozhraním:

sudo apt-get install i2c-tools
sudo apt-get install python-smbus
sudo adduser pi i2c

modul python-smbus pro nás sice není zásadní, protože se zaměříme na programování v jazyku C, může být ale šikovným nástrojem pro testování rozhraní.

Následně systém restartujeme:

sudo reboot

Nyní jsme připraveni otestovat, zda rozhraní I2C funguje. Protože máme k dispozici čidlo BMP085, jehož adresa je dle dokumentace 0x77, zkusíme ho připojit. Rozložení pinů čidla můžeme vidět na obrázku 16. Pro připojení využijeme piny VCC, SCL, SDA a GND. Protože napájecí napětí čidla je dle dokumentace v rozmezí 1,8 – 3,6 V, využijeme pro napájení pin 1 a pin 6 na Raspberry Pi. Jak vidíme pin 1 slouží k napájení napětím 3,3 V a pin 6 je zem. Kontakty SDA a SCL připojíme na příslušné piny tedy 3 a 5. Důležitou informací je, že čidlo má již integrovány pull-up rezistory na obou signálech sběrnice. Celý rozpis hlavního 26ti pinového konektoru Raspberry Pi můžeme vidět v tabulce 4.

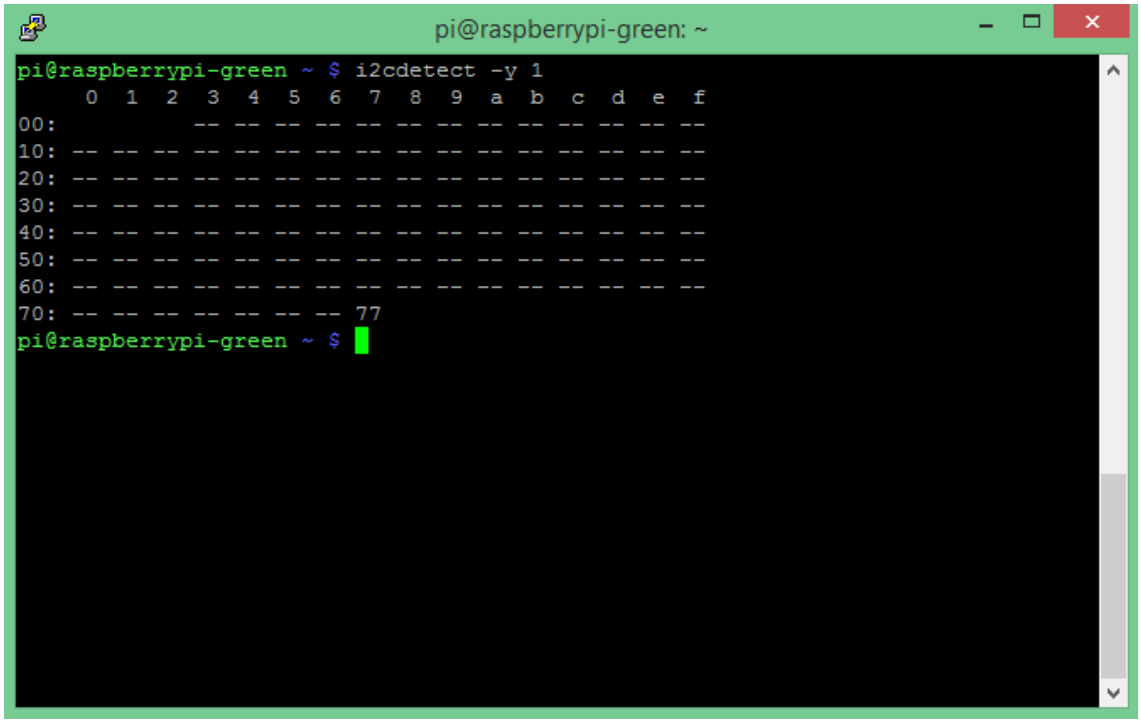
Tabulka 4: Popis pinů hlavního konektoru mikropočítače Raspberry Pi

3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

Po připojení čidla můžeme hned otestovat, zda ho systém vidí. Do příkazového řádku zapíšeme:

i2cdetect -y 1

Výsledek by měl vypadat jako na obrázku 28. Tímto způsobem jsme nastavili a zprovoznili rozhraní I2C mikropočítače Raspberry Pi.



Obr.28: výpis zařízení na sběrnici I2C

Protože programové vybavení chceme realizovat v jazyce C, je pro nás zajímavější získat přístup k portu I2C pomocí programu napsaném v C. Pro komunikaci se senzorem BMP085 na I2C použijme soubory smbus.h a smbus.c . Připojené knihovny funkcí nám umožní pracovat se sběrnicí i2c velmi intuitivním způsobem. Oba soubory lze najít například zde:

http://www.lm-sensors.org/browser/i2c-tools/trunk/lib/smbus.c

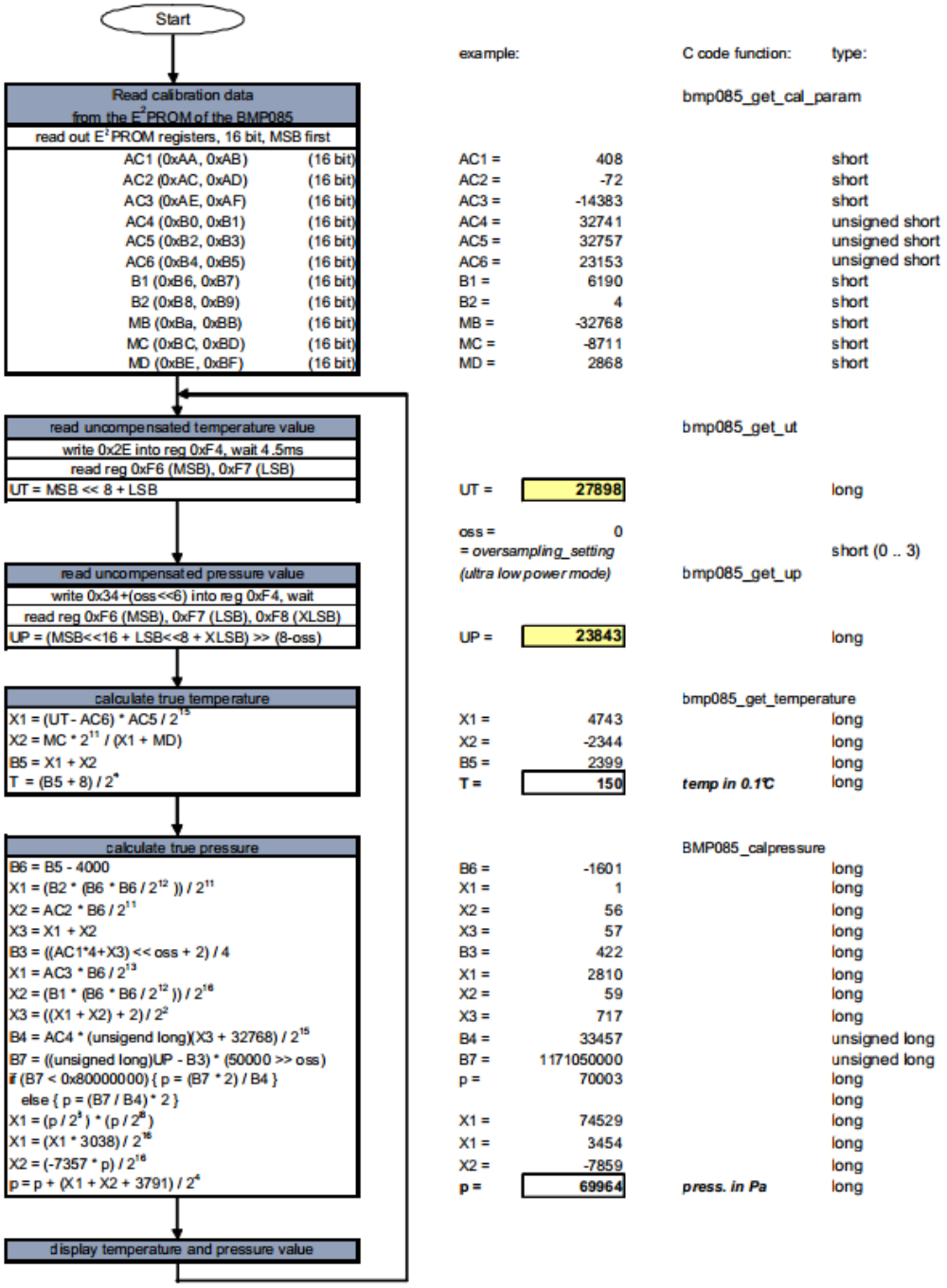
http://www.lm-sensors.org/browser/i2c-tools/trunk/include/i2c/smbus.h

Příklad konfigurace a otevření komunikace se senzorem BMP085:

```
#define BMP085_I2C_ADDRESS 0x77

int bmp085_i2c_Begin()
{
	int fd;
	char *fileName = "/dev/i2c-1";

	// otevře port pro čtení a zápis
	if ((fd = open(fileName, O_RDWR)) < 0)
		exit(1);

	// Nastaví vlastnosti portu a adresu zařízení
	if (ioctl(fd, I2C_SLAVE, BMP085_I2C_ADDRESS) < 0) {

		close(fd);
		exit(1);
	}

	return fd;
}
```

V našem případě je důležitá definice adresy 0x77 a název I2C zařízení /dev/i2c-1.

Program pro načtení hodnot teploty a tlaku ze senzoru BMP085 je poměrně komplikovaný. Nejprve je nutné načíst kalibrační data, která jsou v každém kusu jiná a uložit je do příslušných proměnných. Následně je naměřená nekalibrovaná hodnota a tak je pomocí načtených konstant přepočtena na skutečnou hodnotu. Příklad a popis celé procedury je možné vidět na obrázku 22, což je výňatek z originální dokumentace od firmy Bosch.



Obr.28: načtení kalibračních hodnot a výpočet reálné hodnoty teploty a tlaku z BMP085

3.3 ZPROVOZNĚNÍ UART NA RASPBERRY PI

Stejně jako v případě I2C je třeba i UART (sériový port) na Raspberry Pi nejprve nakonfigurovat. Ovladač je v systému zaveden, ale sériový port vyvedený na piny 8 a10 hlavního konektoru (viz. obrázek 20) je v základním nastavení rezervován pro konzoli.

Příkazem *sudo nano /boot/cmdline.txt* otevřeme soubor cmdline.txt k editaci. Při jeho výpisu bychom měli vidět toto:

dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwai

Ze souboru musíme vymazat část, ktrá se týká zařízení ttyAMA0, takže po zásahu bychom měli vidět takovouto podobu souboru:

dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

Dále bychom měli otevřít soubor /etc/inittab a zakomnetovat znakem # všechny řádky, kde se vyskytuje zařízení ttyAMA0. K nalezení takových řádků můžeme použít v editoru Nano příkaz CTRL+W a následně zadat řetězec ttAMA0 k vyhledání.

Po následném znovuzavedení systému příkazem *sudo reboot* je již sériový port na pinech 8 a10 hlavního konektoru připraven k použití, jeho název v systému je /dev/ttyAMA0.

Uveďme si příklad funkcí pro konfiguraci a otevření portu. Po doplnění kódu o hlavní funkci main(), ve které zavoláme funkce open_port a configure_port provede program otevření a nakonfigurování portu. O úspěchu nebo neúspěchu pak podá hlášení výpisem textu.

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
#include <time.h>


int open_port(void)
{
        int fd; // uloží atributy sériového portu

        fd = open(„/dev/ttyAMA0“, O_RDWR | O_NOCTTY | O_NDELAY);

        if(fd == -1) // pokud nelze otevřít port
        {
                printf(„open_port: Nelze otevřít /dev/ttyAMAO. \n“);
        }
```

```
        else
        {
                fcntl(fd, F_SETFL, 0);
                printf(„port je otevřen.\n“);
        }

        return(fd);
} //

int configure_port(int fd)      // konfigurace portu
{
        struct termios port_settings;

        cfsetispeed(&port_settings, B9600);    // nastavení rychlosti v baudech
        cfsetospeed(&port_settings, B9600);

        port_settings.c_cflag &= ~PARENB;    // bez kontroly parity
        port_settings.c_cflag &= ~CSTOPB;  // bez stop bitů
        port_settings.c_cflag &= ~CSIZE;     //
        port_settings.c_cflag |= CS8;                    // přenos 8mi bitů

        tcsetattr(fd, TCSANOW, &port_settings);    // použití nastavení portu
        return(fd);

}
```

Na port pak můžeme zapisovat byty například příkazem:

```
unsigned char data_odeslat[] = { 201, 202, 203, 204, 205, 206, 207, 208};
write(fd, data_odeslat, 8;  //pošli data
```

3.4 RASPBERRY PI JAKO SERVER

Protože v našem projektu budeme zobrazovat data z Raspberry Pi pomocí WWW rozhraní, potřebujeme na našem mikropočítači zprovoznit funkci webového serveru. V teoretickém rozboru (kapitola 2.4) jsme se rozhodli pro řešení pomocí programu Apache 2. Tato aplikace je velmi rozšířeným serverovým řešení nejen na linuxové platformě a na mikropočítači Raspberry Pi ji zprovozníme velmi jednoduchou doinstalací a editací webových stránek v příslušném adresáři.

V příkazovém řádku zadáme pokyn pro instalaci sérií příkazů:

46ud osu
apt-get update && apt-get upgrade
apt-get install apache2 php5 mysql-client mysql-server tomcat6 vsftpd

Prvním příkazem si nastavíme práva superuživatele, abychom měli přístup do systémových adresářů. Druhým složeným příkazem obnovíme seznam softwaru v repozitářích a následně provedeme upgrade všech instalovaných aplikací na nejnovější verzi. Potom už nainstalujeme balík aplikací:

Apache2 – aplikace pro provoz webového serveru
PHP5 – programovací jazyk k vytváření webových aplikací
MySQL – databázový server založený na SQL
Tomcat6 – aplikační server založený na jazyku Java
VSFTPD – FTP server

Všechny tyto aplikace jsou šířeny jako Open Source.

Během instalace budeme dotázáni na heslo pro uživatele root databázového serveru MySQL. Já jsem zvolil heslo „coating“. Pro pohodlnější práci se systémem MySQL je rozhodně vhodně doinstalovat rozšíření phpMyAdmin. Tento nástroj nám umožní zobrazovat a editovat databáze pomocí webového rozhraní, což je jistě mnohem pohodlnější, než jejich editace v příkazovém řádku pomocí jazyka SQL.

Apt-get install phpmyadmin

V průběhu instalace zvolíme jako webový server Apache2

Následně v příkazovém řádku zadáme příkaz k editaci:

nano /etc/apache2/apache2.conf

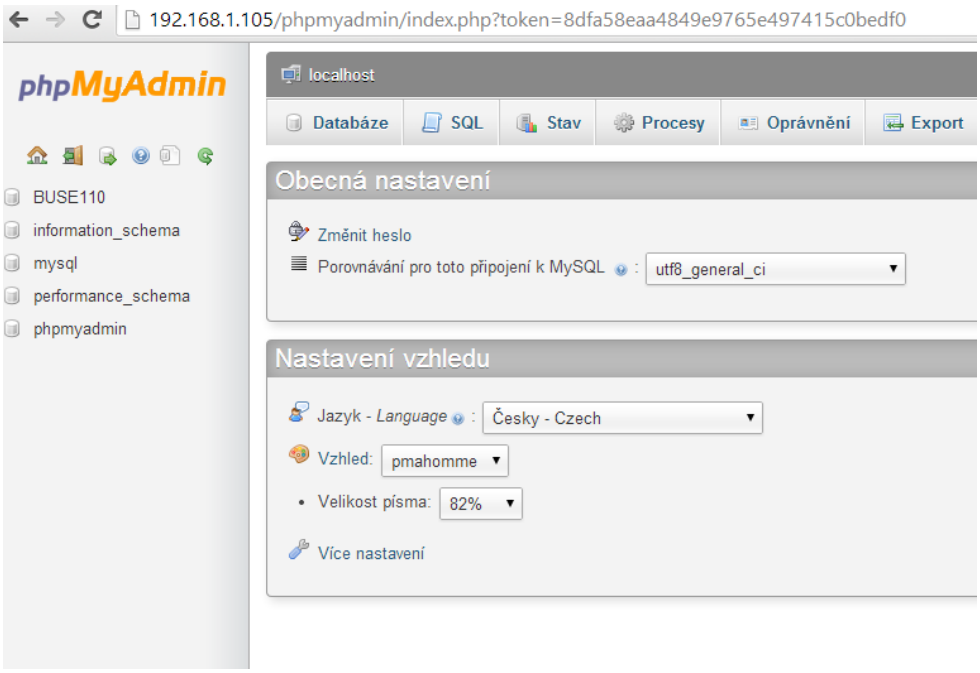
Na konec souboru dopíšeme text Include /etc/phpmyadmin/apache.conf a soubor uložíme.

Nakonec webový server zrestartujeme příkazem

```
/etc/init.d/apache2 restart
```

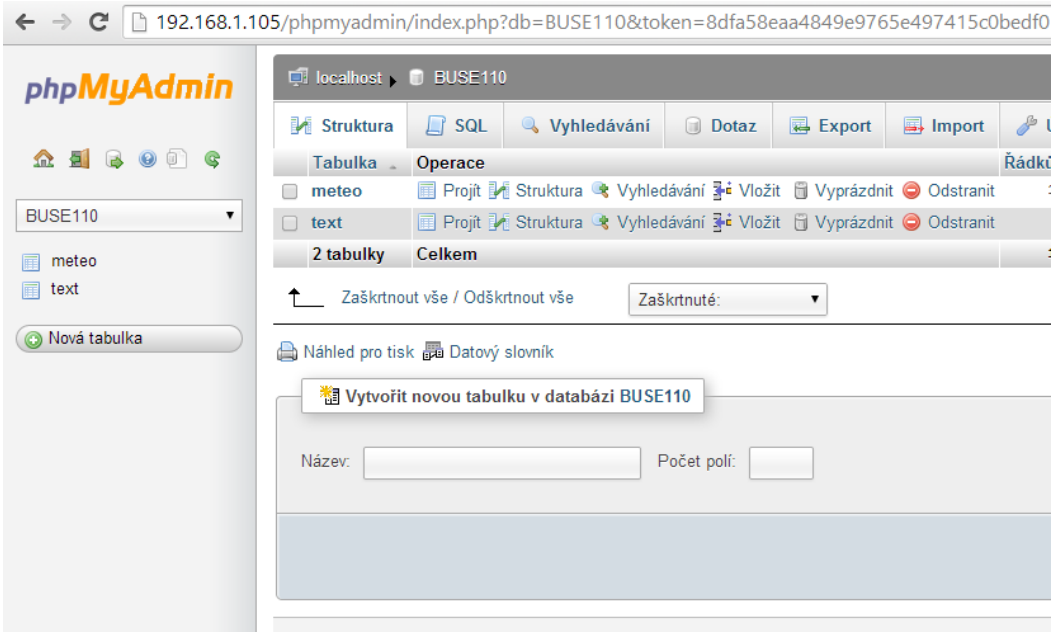
Pokud jsme vše provedli správně, tak nyní můžeme otevřít webový prohlížeč a zadat do něj IP adresu mikropočítače Raspberry Pi. Při správném běhu čerstvě nainstalovaného webového serveru Apache2 uvidíme skoro prázdnou webovou stránku s velkým nápisem“ It Works!“. Jelikož v době, kdy vzniká tento text je již webová stránka na mikropočítači Raspberry Pi jiná, nemohu sem dát konkrétní obrázek ☺

Pro otestování funkce phpMyAdmin, zadáme IP adresu mikropočítače Raspberry Pi a doplníme o text /phpmyadmin. Uvítá nás obrazovka s volbou jazyka a možností přihlášení do MySQL. Já zadám uživatele root a heslo coating (viz. předchozí odstavec o instalaci MySQL).

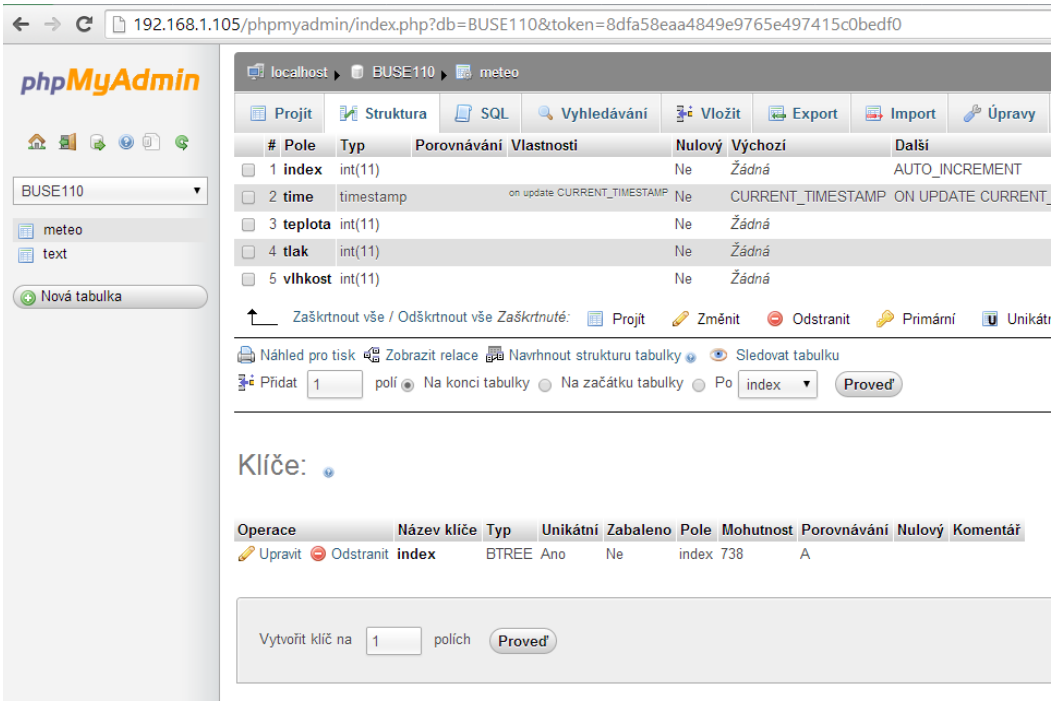


Obr. 30: prostředí MySQL v phpMyAdmin

Na obrázku 30 můžeme vidět, že v levé části nabídky už máme vytvořenou databázi BUSE110. Nebudu zde uvádět konkrétní kroky k obecnému vytvoření databáze v MySQL, ale popíšeme si mnou vytvořenou databázi BUSE110. Databáze obsahuje dvě tabulky. Jedna má název meteo a druhá text. Znalost názvů a položek je klíčová pro přístup do databáze pomocí jazyk PHP nebo C. Do tabulky meteo budeme automaticky zapisovat naměřená data z čidel, abychom je mohli následně zobrazit na webových stránkách. Jednoduchá tabulka text bude zase sloužit k zadání textového řetězce, který následně zašleme k zobrazení na panel BUSE110.



Obr. 31: zobrazení databáze BUSE110

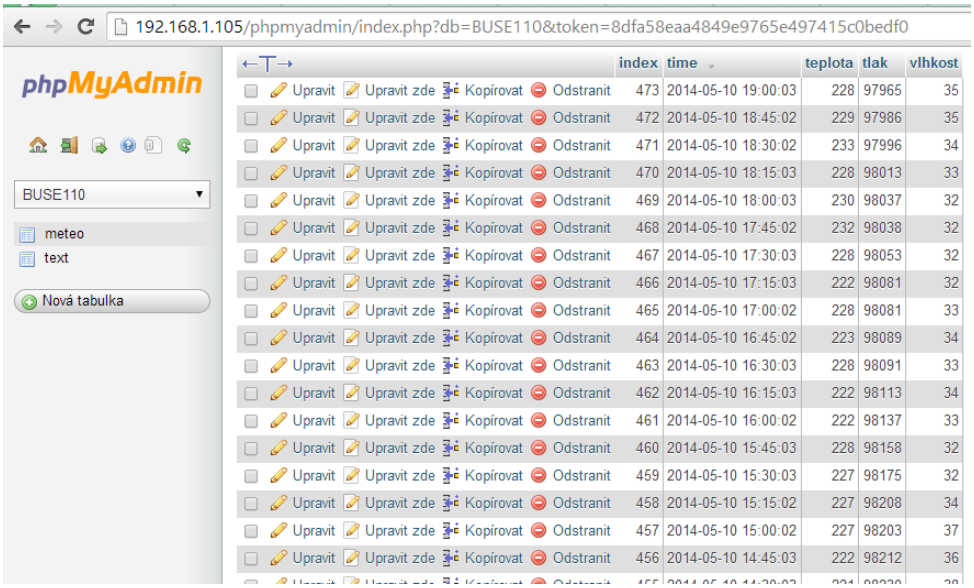


Obr. 32: detail tabulky meteo databáze BUSE110

Jak můžeme vidět na obrázku 32, tabulka meteo obsahuje celkem pět prvků:

- index - automaticky zvyšovaný unikátní údaj o počtu a pořadí měření
- time – automaticky přidaná informace o datu a čase zápisu naměřených hodnot
- teplota – sem vložíme údaj o naměřené teplotě
- tlak – sem vložíme údaj o naměřeném tlaku
- vlkost – sem vložíme údaj o naměřené vlhkosti vzduchu

Důležitým aspektem, který musíme mít na paměti je, že údaje jsou ty **int** , což znamená, že neobsahují desetinnou čárku. Jazyk SQL umožňuje správu dat ve formátu float , ale pro účely čtení údají pomocí API různých programovacích prostředí mi připadá praktičtější řešení, kdy teplotu uložíme ve formátu celých čísel a pro získání relevantního údaje vydělíme číslo 10, čímž získáme údaj s rozlišením na jedno desetinné místo. Přesněji vzato jsem měli stejný údaj i předtím, jen jsem ho teď správně vyjádřili ve stupních Celsia. U tlaku budeme hodnou dělit 100, takže výsledná hodnota bude obsahovat dvě desetinná místa. Nejjednodušší je situace u vlhkosti vzduchu. Čidlo Sensirion SHT11 sice posílá údaj s rozlišením na jedno destinné místo, ale vzhledem k jeho absolutní přesnosti +/- 1% takové rozlišení postrádá smysl. Proto údaj ukládám pouze s rozlišením na celá procenta.



	index	time	teplota	tlak	vlhkost
Upravit Upravit zde Kopírovat Odstranit	473	2014-05-10 19:00:03	228	97965	35
Upravit Upravit zde Kopírovat Odstranit	472	2014-05-10 18:45:02	229	97986	35
Upravit Upravit zde Kopírovat Odstranit	471	2014-05-10 18:30:02	233	97996	34
Upravit Upravit zde Kopírovat Odstranit	470	2014-05-10 18:15:03	228	98013	33
Upravit Upravit zde Kopírovat Odstranit	469	2014-05-10 18:00:03	230	98037	32
Upravit Upravit zde Kopírovat Odstranit	468	2014-05-10 17:45:02	232	98038	32
Upravit Upravit zde Kopírovat Odstranit	467	2014-05-10 17:30:03	228	98053	32
Upravit Upravit zde Kopírovat Odstranit	466	2014-05-10 17:15:03	222	98081	32
Upravit Upravit zde Kopírovat Odstranit	465	2014-05-10 17:00:02	228	98081	33
Upravit Upravit zde Kopírovat Odstranit	464	2014-05-10 16:45:02	223	98089	34
Upravit Upravit zde Kopírovat Odstranit	463	2014-05-10 16:30:03	228	98091	33
Upravit Upravit zde Kopírovat Odstranit	462	2014-05-10 16:15:03	222	98113	34
Upravit Upravit zde Kopírovat Odstranit	461	2014-05-10 16:00:02	222	98137	33
Upravit Upravit zde Kopírovat Odstranit	460	2014-05-10 15:45:03	228	98158	32
Upravit Upravit zde Kopírovat Odstranit	459	2014-05-10 15:30:03	227	98175	32
Upravit Upravit zde Kopírovat Odstranit	458	2014-05-10 15:15:02	227	98208	34
Upravit Upravit zde Kopírovat Odstranit	457	2014-05-10 15:00:02	227	98203	37
Upravit Upravit zde Kopírovat Odstranit	456	2014-05-10 14:45:03	222	98212	36

Obr. 33: detail tabulky meteo databáze BUSE110

Na obrázku 33 vidíme konkrétní hodnoty z 10.5.2014. Data byla zapisována každých 15 minut. Když už máme v tomto případě data v tabulce načtená, obrátíme trochu směr toku informací a zaměříme s v této kapitole na rozhraní MySQL-PHP. Programovací jazyky PHP disponuje dostatečným počtem nástrojů pro správu sat v databázích MySQL. Pomocí webových stránek můžeme informace v databázových tabulkách obsažené zobrazovat, editovat, vytvářet i mazat. Když si uvědomíme, že námi před chvilkou používaný klient phpMyAdmin je napsán právě v jazyku PHP, uděláme si asi o možnostech jazyka PHP v této oblasti sami obrázek.

Stránky, které nám webový sever běžící na mikropočítači Raspberry Pi zasílá jsou v souborovém systému uloženy v adresáři /var/www/ . Pokud do tohoto adresáře nahrajeme webovou stránku s názvem index.html nebo index.php (jsou-li v php napsány) bude to právě tato stránka, která se nám zobrazí v prohlížeči při zadání IP adresy Raspberry Pi.

Nyní si sestavíme jednoduché webové stránky v jazyku PHP, které nám zobrazí tabulku naměřených dat uložených v databázi BUSE110 v tabulce meteo.

```
<!DOCTYPE html>
<html>
 <head>
  <title>RaspberryPi-green MySQL pro BUSE110 zobrazení</title>
 </head>
<body>

<br>

<?php

$con=mysqli_connect("localhost","root","coating","BUSE110");
// kontrola připojení
if (mysqli_connect_errno()) {
  echo "Chyba připojení k MySQL: " . mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM meteo ORDER BY time DESC LIMIT 30");

echo "<table border='1'>
<tr>
<th>Index</th>
<th>Datum a čas</th>
<th>Teplota</th>
<th>Tlak</th>
<th>Vlhkost</th>
</tr>";

while($row = mysqli_fetch_array($result)) {
  echo "<tr>";
  echo "<td>" . $row['index'] . "</td>";
  echo "<td>" . $row['time'] . "</td>";
  echo "<td>" . $row['teplota'] . "</td>";
  echo "<td>" . $row['tlak'] . "</td>";
  echo "<td>" . $row['vlhkost'] . "</td>";
  echo "</tr>";
}
echo "</table>";
mysqli_close($con);
?>
</body>
```

Zobrazený kód je kompletní a po připojení k databázi, k čemuž musíme znát přihlašovací údaje, název databáze a tabulky, kterou chceme načíst. Začínáme běžným formátování v jazyku html, mezi značkami <?php a ?> jse nachází kód v jazyku PHP. Samotný přístup do databáze využívá rozhraní mysqli. Text:

```
"SELECT * FROM meteo ORDER BY time DESC LIMIT 30"
```

je psaný v jazyku SQL a jedná se o složený příkaz, který načte prvních 30 údajů všech položek tabulky meteo seřazených podle času sestupně. Následuje html kód, který získané údaje zformátuje tak, že je zpřehlední a zapíše do tabulky. Pokud tento soubor uložíme s názvem index.php do adresáře /var/www mikropočítače Raspberry Pi, pak to bude právě tato stránka, která se nám zobrazí po zadání IP adresy Raspberry Pi do prohlížeče WWW stránek.

Index	Datum a čas	Teplota	Tlak	Vlhkost
1043	2014-05-18 18:15:02	225	98228	50
1042	2014-05-18 18:00:03	225	98239	50
1041	2014-05-18 17:45:02	225	98239	50
1040	2014-05-18 17:30:02	225	98229	50
1039	2014-05-18 17:15:03	225	98244	50
1038	2014-05-18 17:00:03	225	98249	50
1037	2014-05-18 16:45:02	220	98252	50
1036	2014-05-18 16:30:02	219	98260	50
1035	2014-05-18 16:15:03	219	98266	49
1034	2014-05-18 16:00:03	219	98279	50
1033	2014-05-18 15:45:02	219	98274	49
1032	2014-05-18 15:30:02	219	98268	49

Obr. 34: zobrazení souboru index.php ve webovém prohlížeči

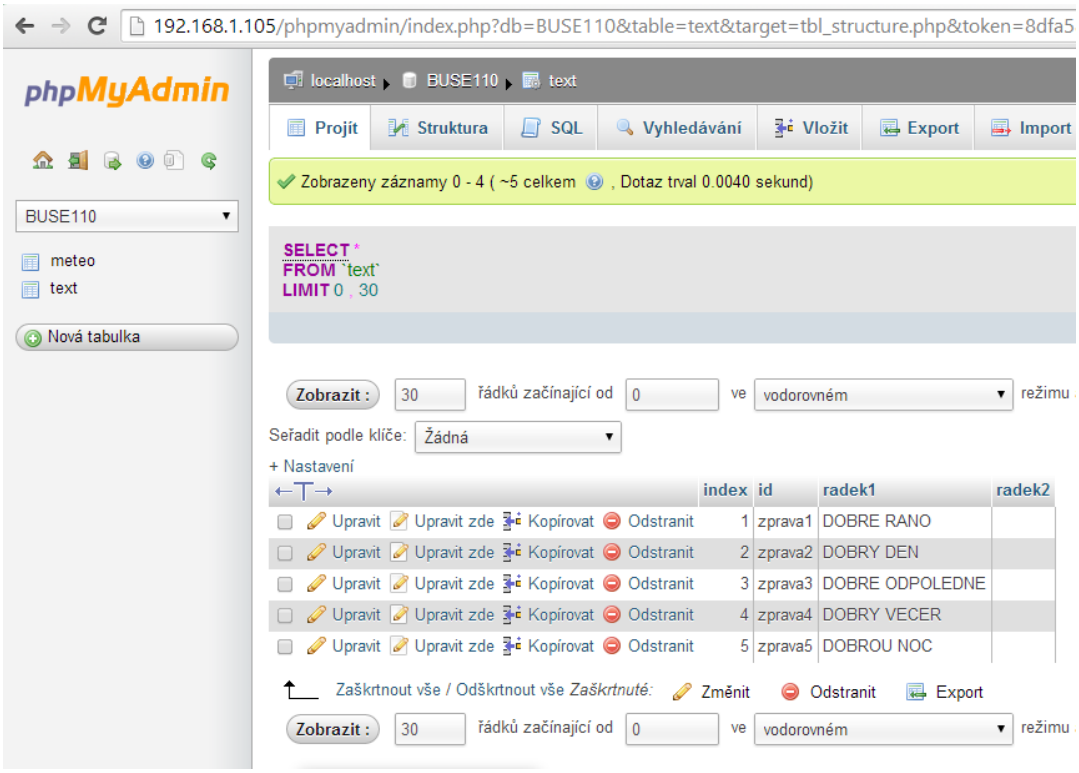
Obnovení zobrazení této stránky bude mít za následek zaktualizování hodnot zobrazených na stránce. Naměřená data jsou dostupná všem zařízením připojeným ve stejné místní síti. Data si můžeme prohlédnout na tabletu, telefonu, či na jakémkoliv dalším zařízení, stačí nám k tomu je připojení na stejný síť (router) a znalost IP adresy Raspberry Pi.

Nyní si ukážeme, jak můžeme přes webové rozhraní mikropočítači Raspberry Pi data naopak posílat a následně je ukládat do databáze k dalšímu použití. Řekněme, že chceme zadat a uložit text nějakého znění pro pozdější použití. Velkou výhodou celého návrhu našeho rozhraní je, že data uložená v MySQL databázi jsou takto přístupná pro další možné aplikace a to nejenom k zobrazení.

V databázi BUSE110 si vytvoříme další tabulku, která bude obsahovat celkem pět záznamů s údaji:

- index – pořadí záznamů v databázi
- id – textový popis pořadí zpráv (int formát hodnoty v indexu není vždy vhodný pro zpracování, takto máme k rozlišení záznamů k dispozici i textový řetězec
- radek1 – první řádek zprávy
- radek2 – druhý řádek zprávy

Ačkoliv je to zcela na rozhodnutí uživatele, předpokládejme nyní, že zpráv bude vždycky pouze pět a budeme je různě měnit. Nebudeme tedy ukládat nové záznamy, ale měnit ty stávající.



Obr. 35: náhled tabulky text databáze BUSE110

Jak vidíme na obrázku 35, tabulka text obsahuje pět zpráv u kterých je vyplněna pouze položka radek1. Původní soubor index.php si uložíme (vytvoříme si kopii) do adresáře /var/www/php . Děláme to čistě jen proto, že hlavní stránku chceme mít přístupnou pouze pro zobrazení dat.

Zkopírovaný soubor nazveme třeba mysqlBUSE110.php a kód rozšíříme o nové funkce:

```
<!DOCTYPE html>
<html>
 <head>
  <title>RaspberryPi-green MySQL pro BUSE110 zobrazení a editace</title>
 </head>
<body>

<br>

  <form method='POST' action='mysqlBUSE110.php'>
  <select name='cislo'>
  <option value=zprava1>Zpráva1</option>
  <option value=zprava2>Zpráva2</option>
  <option value=zprava3>Zpráva3</option>
  <option value=zprava4>Zpráva4</option>
  <option value=zprava5>Zpráva5</option>
  </select>
  Řádek1: <input type='text' name='zprava1'>
  Řádek2: <input type='text' name='zprava2'>
  <input type='submit' name='Odeslat' value='Pošli!'>
  </form>
<br>
<br>

<?php
$con=mysqli_connect("localhost","root","coating","BUSE110");
// kontrola připojení
if (mysqli_connect_errno()) {
  echo "Chyba připojení k MySQL: " . mysqli_connect_error();
}
if (isset($_POST['Odeslat'])) {

$radek1 = $_POST['zprava1'];
$radek2 = $_POST['zprava2'];
 $index =  $_POST['cislo'];

mysqli_query($con,"UPDATE text SET radek1='$radek1', radek2='$radek2' WHERE
id='$index'");
}
$result = mysqli_query($con,"SELECT * FROM text");

echo "<table border='1'>
<tr>
<th>Číslo zprávy</th>
<th>Řádek1</th>
<th>Řádek2</th>
</tr>";

while($row = mysqli_fetch_array($result)) {
  echo "<tr>";
```

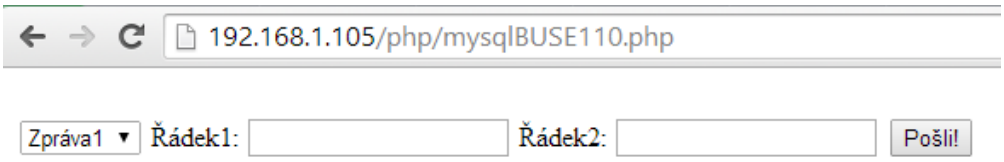
```
  echo "<td>" . $row['id'] . "</td>";
  echo "<td>" . $row['radek1'] . "</td>";
  echo "<td>" . $row['radek2'] . "</td>";
  echo "</tr>";
}

echo "</table>";

echo "<br>";
echo "<br>";

$result = mysqli_query($con,"SELECT * FROM meteo ORDER BY time DESC LIMIT 12");

echo "<table border='1'>
<tr>
<th>Index</th>
<th>Datum a čas</th>
<th>Teplota</th>
<th>Tlak</th>
<th>Vlhkost</th>
</tr>";

while($row = mysqli_fetch_array($result)) {
  echo "<tr>";
  echo "<td>" . $row['index'] . "</td>";
  echo "<td>" . $row['time'] . "</td>";
  echo "<td>" . $row['teplota'] . "</td>";
  echo "<td>" . $row['tlak'] . "</td>";
  echo "<td>" . $row['vlhkost'] . "</td>";
  echo "</tr>";
}

echo "</table>";

mysqli_close($con);

?>

</body>
```

Kód funguje následovně: po kliknutí na tlačítko „Pošli!“ provede načtení vyplněných dat a zvoleného čísla zprávy z příslušných vstupních prvků (textové pole a výběr položek). Následně provede změnu příslušné položky v databázi BUSE110 v tabulce text a obnoví stránku. Nejdůležitější částí kódu je tedy řádek:

```
mysqli_query($con,"UPDATE text SET radek1='$radek1', radek2='$radek2' WHERE id='$index'");
```

Zde probíhá změna položky v databázi na základě načtených proměnných, což jsou zadané textové řetězce pro řádek 1 případně řádek 2 a vybraná položka (jedna z pěti možných). Takovýmto způsobem můžeme měnit data v databázi. Zbytek kódu je

už velmi podobný předchozímu případu, pouze jsme navíc vypsali obsah tabulky text a zkrátili seznam zobrazených naměřených dat z tabulky meteo na 12. Jde jen o velmi jednoduchý příklad, ale opět platí, že změnu tabulky můžeme udělat z jakéhokoliv zařízení na stejné místní síti s Raspberry Pi a s možností zobrazit webové stránky.



Číslo zprávy	Řádek1	Řádek2
zprava1	DOBRE RANO	
zprava2	DOBRY DEN	
zprava3	DOBRE ODPOLEDNE	
zprava4	DOBRY VECER	
zprava5	DOBROU NOC	

Index	Datum a čas	Teplota	Tlak	Vlhkost
1046	2014-05-18 19:00:03	226	98215	50
1045	2014-05-18 18:45:03	226	98224	50
1044	2014-05-18 18:30:02	225	98224	50
1043	2014-05-18 18:15:02	225	98228	50
1042	2014-05-18 18:00:03	225	98239	50
1041	2014-05-18 17:45:02	225	98239	50
1040	2014-05-18 17:30:02	225	98229	50
1039	2014-05-18 17:15:03	225	98244	50
1038	2014-05-18 17:00:03	225	98249	50
1037	2014-05-18 16:45:02	220	98252	50
1036	2014-05-18 16:30:02	219	98260	50
1035	2014-05-18 16:15:03	219	98266	49

Obr. 36: výsledná podoba souboru mysqlBUSE110.php

Důležitou poznámkou je, že umístění této nové stránky je nyní v adresáři /php/ pod názvem mysqlBUSE110. Jak se ke stránce dostaneme je dobře vidět v adresovém řádku prohlížeče na obrázku 36.

3.5 PŘÍSTUP DO MYSQL V PROGRAMOVACÍM JAZYKU C

Protože v předchozí kapitole jsme si vytvořili databáze k ukládání dat a zprovoznili způsob, jakým je budeme zobrazovat a editovat přes WWW rozhraní, je nejvyšší čas zabývat se způsobem, jak získáme data z čidel a zapíšeme je do databáze. V teoretické části projektu jsme si stanovili měřit teplotu, tlak a vlhkost vzduchu v interiéru pomocí čidel Sensirion SHT11 a Bosch BMP085. Čidlo SHT11 použijeme pro měření teploty a vlhkosti vzduchu, BMP085 pak použijeme k měření tlaku vzduchu.

Pro získání dat z čidel a jejich zápis do databáze jsem sestavil program meteo_sql.c . Program samotný je poměrně obsáhlý zejména v části věnované senzorům. Ke správné obsluze portu I2C, na kterém je připojeno čidlo BMP085 a portu Two-wire ke kterému je připojeno čidlo SHT11, navíc používám externí volně šiřitelné knihovny. Program meteo_sql.c ale obsahuje i část, kde naměřená data zasíláme do databáze a tyto funkce si nyní popíšeme.

Abychom mohli v programu přistupovat k databázím v MySQL musíme si do našeho systému doinstalovat vývojové nástroje. To provedeme příkazem:

```
sudo apt-get install libmysqlclient-dev
```

Následně můžeme využít API v jazyku C pro přístup k databázím. Opět potřebujeme dopředu znát přihlašovací údaje do MySQL a základní informace o příslušné databázi (pokud nechceme programem tvořit databázi novou).

```
int main(int argc, char **argv)
{
   // volání bmp085
        bmp085_Calibration();
        temperature = bmp085_GetTemperature(bmp085_ReadUT());
        pressure = bmp085_GetPressure(bmp085_ReadUP());

   // volání SHT11

        if(!bcm2835_init())
                return 1;
   int temperatureSHT11;
        int humiditySHT11;

   temperatureSHT11 = (int) (SHT11Temp());
        humiditySHT11 = (int) (SHT11Humi());

        printf("Temperature1\t%0.1f%cC\n", SHT11Temp(),0x00B0);
        printf("Humidity\t%0.2f%\n", SHT11Humi());
        printf("Temperature2\t%0.1f%cC\n", ((double)temperature)/10,0x00B0);
        printf("Pressure\t%0.2fhPa\n", ((double)pressure)/100);


        //část s mysql API
```

```
        int teplota = (temperatureSHT11*10);
        int tlak = pressure;
        int vlhkost = humiditySHT11;
        char mysqlprikaz[512];

        snprintf(mysqlprikaz, 512, "INSERT INTO meteo VALUES (DEFAULT, DEFAULT, %i, %i, %i)",
teplota, tlak, vlhkost);

        MYSQL *con = mysql_init(NULL);

  if (con == NULL)
  {
    fprintf(stderr, "mysql_init() failed\n");
    exit(1);
  }

  if (mysql_real_connect(con, "localhost", "root", "coating", "BUSE110", 0, NULL, 0) == NULL)
  {
    finish_with_error(con);
  }
        if (mysql_query(con, mysqlprikaz)) {
  finish_with_error(con);
        }

        mysql_close(con);

        //konec mysql API
        return 0;
}
```

Ve výše uvedeném výpisu vidíme celou funkci main programu meteo_sql.c . Program je v mikropočítači Raspberry Pi uložen ve formě zdrojového kódu i zkompilovaný v adresáři ~/meteo_sql/ . Je samozřejmě součástí elektronické přílohy této dilomové práce. Protože jsme se v této kapitole zaměřili na komunikaci s MySQL z programu jazyka C zaměříme se na část okomentovanou jako //část s mysql API.

Na začátku si pro přehlednost vytvoříme proměnné se stejným názvem, jako používáme v tabulce meteo. Protože program v případě zavolání z příkazové řádky vypíše hodnoty i na obrazovku, pro co jsme již naměřené hodnoty naformátovali, vynásobíme hodnotu změřené teploty z SHT11 deseti, abychom získali číslo ve formátu **int**. Nadefinujeme si řetězec mysqlprikaz s velikostí do 512 znaků. Do tohoto řetězce pak příkazem :

```
snprintf(mysqlprikaz, 512, "INSERT INTO meteo VALUES (DEFAULT, DEFAULT, %i, %i, %i)", teplota,
tlak, vlhkost);
```

uložíme řetězec, který je úplným příkazem v jazyku SQL, a který následně po otevření připojení do databáze odešleme do MySQL. V tomto okamžiku dojde k vytvoření nového záznamu v tabulce meteo databáze BUSE110, kde položku index a time doplní automaticky systém samotný a položky teplota, tlak a vlhkost jsme definovali sami. Při kompilaci programu, kde využíváme funkce pro přístup do MySQL musíme (alespoň na

Raspberry Pi) zadat umístění příslušných knihoven a konfigurace MySQL. Například takto:

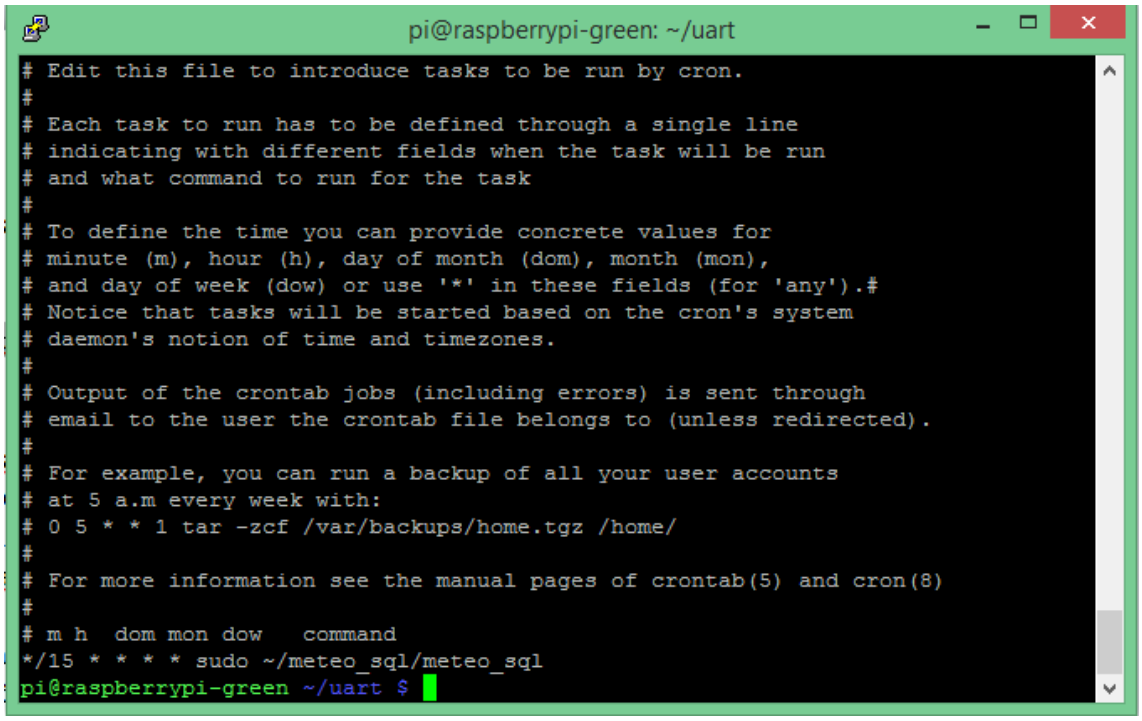
sudo gcc -o meteo_sql $(mysql_config --cflags) meteo_sql.c $(mysql_config --libs)

Je zřejmé, že ke správnému zápisu data a času je nutné, aby systémový čas mikropočítače Raspberry Pi byl korektně nastaven. Pokud je v síti, do které je Raspberry Pi připojeno, k dispozici připojení k Internetu, což celou dobu předpokládáme, neboť jsme veškerý software instalovali ze vzdálených repozitářů, tak se systém Raspbian postará o nastavení systémového času sám. Pokud toto připojení k dispozici není a mikropočítač Raspberry Pi restartujeme, bude systémový čas nesprávný. To můžeme opravit buď ručním nastavením nebo vyřešit zakoupením modulu reálného času. V každém případě jsou záznamy zapsané do tabulky meteo vždy opatřeny jedinečným indexem (pořadové číslo měření), které nás může minimálně upozornit, že záznam času není korektní. V případě, že se začnou objevovat měření se starším datem a vyšším indexem.

Abychom zajistili opakované měření a zapisování dat v pravidelném určeném čase, můžeme využít linuxového programu cron. Editací seznamu úloh tohoto programu příkazem:

crontab –e

Nastavíme například spuštění programu meteo_sql každých 15 minut. To provedeme zápisem řádku dle obrázku 37.



Obr. 37: zanesená úloha spouštět každých 15 minut program meteo_sql

3.6 OVLÁDÁNÍ PANELU BUSE110

V teoretické části práce jsme zvolili k použití mikrokontrolér Atmel AVR ATmega32 a obvody TD62783, ULN2803 a 74HC238 k řízení příslušných vstupů na sběrnici panelu BUSE110. Ideový návrh jsem v praxi navrhl v programu Eagle6.5.0 a vyrobil prototypové desky. Desky tištěných spojů jsem vyrobil doma z polotovarů opatřených fotocitlivým lakem. Desky jsem nasvítil UV diodami přes předlohy vytištěné na laserové tiskárně a zprůhledněné přípravkem TRANSPARENT 21ve spreji. Fotocitlivý lak jsem vyvolal (osvícenou část rozpustil) v 1,5% roztoku Hydroxidu Sodného a následně desky vyleptal v roztoku Persíranu Sodného (250g na 1l vody). Po osazení a zapájení jsem desky propojil a oživil. Prototypové desky zapojení jsem koncipoval jako moduly k odzkoušení

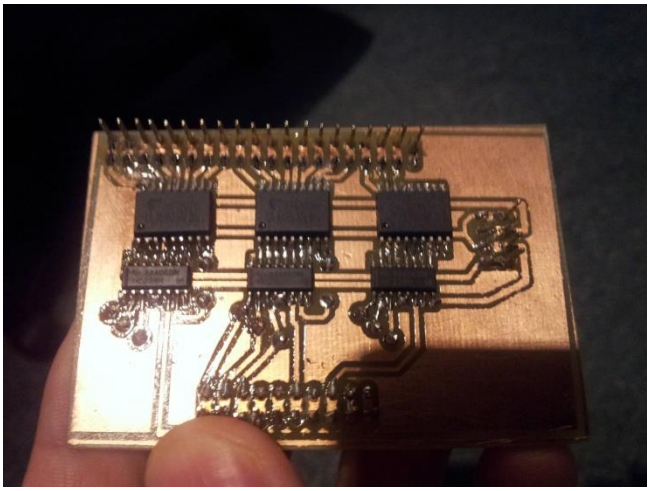
Musím uvést, že jsem od začátku počítal s výrobou desek svépomocí a v případě úspěšného odzkoušení desek jsem chtěl provést návrh ze stejného (funkčního) schématu, ale v podobě pro profesionální vyhotovení desek. Protože první generace prototypů se mi příliš nepovedla, návrh jsem opakoval a v současné podobě systém funguje obstojně. Výrobu profesionálně zpracovaných desek plošných spojů jsem ale do termínu odevzdání diplomové práce nestihl. Na druhou stranu modulární provedení má své výhody v další možnosti dopracování

Celkem jsem navrhl a sestavil 5 modulů:

- modul s plochým konektorem k připojení na sběrnici BUSE110 a rozvedení kontaktů na konektory Wago
- modul s procesorem AVR ATmega32 s vývody portů na konektory Wago
- modul s obvody 74HC238 a TD62783 pro zápis bodů na panelu
- modul s obvody 74HC238 a ULN2803 pro výmaz bodů na panelu
- Napájecí modul se vstupem 24V a stabilizaci napětí na hodnotu 12V a 5V pomocí obvodů řady L7805 respektive L7812.

Obr. 38: improvizovaná osvitová jednotka 99ti UV LED (napájení 12V)

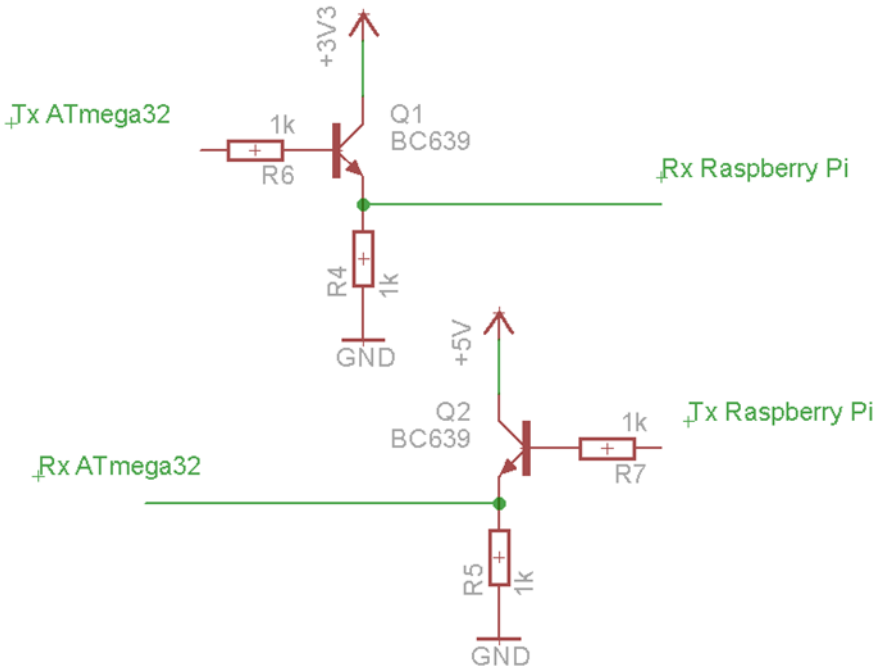
Obr. 39: předloha k osvícení desky s fotocitlivým lakem



Obr. 40: hotový modul s obvody 74HC238 a TD62783 pro zápis bodů na panel

Návrh obvodů a desek plošných spojů je uveden mezi přílohami diplomové práce. Celá koncepce zůstala stejná, jak byla nastíněna v teoretickém úvodu. Na mikrokontroléru AVR ATmega32 používáme porty A a C a pin 7 portu D. Dohromady máme tedy 17 signálů k obsluze panelu. Osm výstupů (PA0-PA7) používáme k adresaci sloupců na panelu, tedy piny 30 - 37 na sběrnici panelu. Výstup PC7 slouží k nastavení, zda se provádí zápis nebo mazání, výstup PD7 je signál k proudovému impulzu pro zápis nebo výmaz bodu. Zbylé výstupy PC0-PC6 slouží k ovládání multiplexorů 74HC238 zcela identicky podle návrhu z obrázku 8 v teoretické části práce. Piny PD0 a PD1 slouží jak UART rozhraní.

Pro propojení Raspberry Pi s ATmega32 musíme přizpůsobit úroveň signálů. Logika Raspberry Pi totiž pracuje s napětím 3,3V zatímco ATmega32 pracuje s úrovní 5V, už kvůli komunikaci s elektronikou panelu BUSE110. Přizpůsobení obou signálů provedeme jednoduše pomocí spínacích tranzistorů BC639.



Obr. 41 přizpůsobení napěťových úrovní signálů

Jednoduchý konvertor úrovní signálů dle obrázku 41 je již součástí modulu s mikrokontrolérem ATmega32. Protože příklad zprovoznění UART na Raspberry Pi včetně příkladu programového kódu jsme si uvedli již v kapitole 3.3, provedeme nyní totéž na příkladu pro ATmega32. V kapitole 2.8 jsme již teoretický rozbor práce s rozhraním popsali. Pokud shrneme důležité informace, tak nastavení portu bude následující:

Rychlost 9600 BAUDŮ, 8 datových bitů, žádná kontrola parity a 1 stop bit.

Ukázka výpisu kódu s nastavením a s definovanými funkcemi pro čtení a zápis do datového registru UART.

```
/*
 * BUSEtestadresace.c
 *
 * Created: 12. 5. 2014 22:01:03
 *  Author: Jiří Pilný
 */


#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

// nastaveni rychlosti v BAUDech a vypocet prescale hodnoty pro registr UBRR
#define USART_BAUDRATE 9600
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)
```

```
/*************************************************************************
        Inicializace USART nahozeni prijimani i odesilani
*************************************************************************/
void _init_usart() {
        UBRRL = BAUD_PRESCALE;// nastaveni nizsich 8mi bitu hodnoty BAUD_PRESCALE
v nizsim bytu UBRR registeru
        UBRRH = (BAUD_PRESCALE >> 8); // nastaveni vyssich 8mi bitu hodnoty
BAUD_PRESCALE ve vyssim bytu UBRR registeru
        UCSRB = 0x18; // zvolen transmitter i reciever
        UCSRC = 0x86; // 8 bitova data, zadny parity bit a 1 stop bit.
}

/*************************************************************************
        Prijem bytu, ktery dorazi na Rx
*************************************************************************/
unsigned char usart_receive( void ) {
        unsigned char data;
        while ((UCSRA & 0x80) == 0x00); // ceka dokud neprijdou data
        data=UDR; // jakmile data prijdou jsou prectena z UDR
        return data;
}

/*************************************************************************
        Odeslani dat, ktera jsou vlozena jako argument funkce
*************************************************************************/
void usart_transmit( unsigned char data ) {
while ((UCSRA & 0x20) == 0x00); // ceka dokud není buffer prazdny
UDR = data; // jakmile je buffer prazny, uloz data do UDR
}
```

Pokud ve funkci main zavoláme funkci pro příjem bytu například takto:

```
znak = usart_receive(); // prijmi znak z usart
```

bude funkce usart_receive čekat, než přijdou data a následně je předá proměnné znak ve funkci main(). A to je smírnou nadsázkou všechno, co od komunikace mezi Atmega32 a Raspberry Pi požadujeme. Raspberry Pi bude vysílat povely k řízení panelu a ATmega32 bude na základě došlých příkazů nastavovat svoje výstupní porty.

Pro ATmega32 jsem zvolil velmi jednoduchou koncepci, protože na UART budou přicházet jednotlivé byty, bylo by dobré, pokud by jejich vyhodnocení nebylo závislé na tom, v jaké sekvenci za sebou následovaly. Zkrátka, pokud by z hodnoty bytu šlo okamžitě poznat, co má Raspberry Pi vykonat. Protože máme na panelu 140 sloupců a 19 řádků, které mohou být buď nahozeny nebo smazány, můžeme řídící povely stanovit například takto:

Znak 0x01 (1) až 0x8C (140) – ATmega nastaví PA0-PA7 na příslušnou hodnotu pro výběr konkrétního sloupce na elektronice panelu.

Znak 0xC9 (201) až 0xDB (219) – ATmega přivede přes pomocí výstupů PC0-PC6 na obvody 74HC238 a TD62783 signál tak, aby na příslušný řádek (odpovídá hodnotě došlého bytu – 200) bylo připojeno napětí 24V, pak nastaví pin PC7 na zápis („0“) a provede spuštění proudového impulsu pinem PD7 (po dobu alespoň 500 mikrosekund).

Znak 0xE7 (231) až 0xF9 (239) – ATmega přivede přes pomocí výstupů PC0-PC6 na obvody 74HC238 a TDULN2803 signál tak, aby na příslušný řádek (odpovídá hodnotě došlého bytu – 230) byla připojena zem, pak nastaví pin PC7 na výmaz („1“) a provede spuštění proudového impulsu pinem PD7 (po dobu alespoň 500 mikrosekund).

Znak 0x96 (150) – ATmega32 postupně zapíše všechny body na panelu

Znak 0xA0 (160) – ATmega32 postupně smaže všechny body na panelu

Celou tuto proceduru vykonává program BUSEadresace.c napsaný pro ATmega32 v prostředí Atmel Studio 6.1. Výhodou celého postupu je, že ATmega nemusí čekat na příjem více bytů, z každého došlého bytu lze jednoznačně určit, co má dělat. Zároveň nám zbylo poměrně dost místa na další příkazy, pokud bychom chtěli doplnit nové funkce. Nevýhodou takového systému je, že jsme sice velmi zjednodušili softwarový návrh na straně ATmega32, ale o to náročnější nás čeká práce při tvorbě znaků a symbolů na straně Raspberry Pi.

3.7 ZASÍLÁNÍ ZNAKŮ NA PANEL

Elektroniku panelu máme připravenou, funkci můžeme velmi jednoduše ověřit zasláním znaků pro výmaz nebo nahození panelu. Například si můžeme napsat jednoduchý program, který pouze zašle byte s dekadickou hodnotou 160 na sériový port. Pokud zasílané číslo bude mít dekadickou hodnotu 150, všechny body na panelu se nahodí (celý panel bude žlutý).

```
//////////////////////////////////////////////////////////////////////
// program zašle číslo 160 na sériový port     //
//////////////////////////////////////////////////////////////////////

#include <stdio.h> // standard input / output functions
#include <string.h> // string function definitions
#include <unistd.h> // UNIX standard function definitions
#include <fcntl.h> // File control definitions
#include <errno.h> // Error number definitions
#include <termios.h> // POSIX terminal control definitions
#include <time.h>   // time calls

int open_port(void)
{
        int fd; // uloží atributy sériového portu

        fd = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY | O_NDELAY);

        if(fd == -1) // pokud nelze otevřít port
        {
                printf("open_port: Nelze otevřít /dev/ttyAMAO. \n");
        }
        else
        {
                fcntl(fd, F_SETFL, 0);
                printf("port je otevřen.\n");
        }

        return(fd);
} //

int configure_port(int fd)      // konfigurace portu
{
        struct termios port_settings;

        cfsetispeed(&port_settings, B9600);    // nastavení rychlosti v baudech
        cfsetospeed(&port_settings, B9600);

        port_settings.c_cflag &= ~PARENB;    // bez kontroly parity
        port_settings.c_cflag &= ~CSTOPB;       // bez stop bitů
        port_settings.c_cflag &= ~CSIZE;     //
        port_settings.c_cflag |= CS8;                   // přenos 8mi bitů

        tcsetattr(fd, TCSANOW, &port_settings);    // použití nastavení portu
        return(fd);

}

main()
{
```

```
int fd = open_port();
configure_port(fd);
unsigned char vymaz[] = { 160};
        write(fd, vymaz, 1);
}
```

Pokud bychom zapsali za sebou například číslo 1 a 201, tak se v levém dolním rohu panelu nahodí příslušný bod na pozici prvního řádku, prvního sloupce. Po zaslání hodnoty 231 bod zase zmizí (příslušný sloupec je stále aktivní, není třeba opět zasílat jeho hodnotu). Jakkoliv můžeme mít z takové funkce a jednoduchého ověření funkčnosti bodů na panelu radost, tak celá práce vlastně teprve začíná. Pokud totiž chceme zapsat na panel textovou informaci, musíme mít k dispozici nějakou znakovou sadu k tomu, abychom na panel mohli vypisovat písmena.

Znaky jsou v základní podobě u mikrokontrolérů definovány pomocí sekvence hexadecimálních čísel. Jedno číslo představuje například v síti 8x5 bodů pět po sobě jdoucích dvojciferných hexadecimálních čísel. Je to vlastně pět sloupců kde hexadecimální hodnoty rozepsané binárně od nejnižšího bitu po nejvyšší směrem od zdola nahoru určují které body mají být viditelné, jsou v logické „1“ a která mají být „zhasnuty“. Například písmeno A, vypadá v takové síti následovně:

	D	E	F	G	H	I	J	K
	0	1	1	1	0	0	0	0
	1	0	0	0	1	0	0	0
	1	0	0	0	1	0	0	0
	1	0	0	0	1	0	0	0
	1	1	1	1	1	0	0	0
	1	0	0	0	1	0	0	0
	1	0	0	0	1	0	0	0
	1	0	0	0	1	0	0	0
ın	1	2	3	4	5	6	7	8
	7F	88	88	88	7F	00	00	00

Obr. 42: zobrazení znaku A v matici 8x8

Bohužel nám v takovém případě nezbývá nic jiného, než napsat všechny potřebné znaky s jejich bitovou reprezentací do programu a následně řetězce znak po znaku a znaky bod po bodu vykreslovat na bodový displej. Můžeme porovnat A z obrázku 40 se znakem z nápisu AHOJ, který jsem zapsal pomocí programu znakbuse.c. Program zná (zatím) pouze znaky velké abecedy bez diakritiky a čísla od nuly do devíti. Po zapsání příkazu:

./znakbuse AHOJ 100 2

Provede program odeslání dat na sériový port k vykreslení nápisu AHOJ na pozici 100. sloupce zleva a ve druhém řádku od spodu.

Obr. 43: zapsání textového řetězce na panel BUSE110

Protože program znakbuse.c čte zadání z příkazové řádky, nelze zadat řetězec s mezerou. To lze udělat zápisem slov zvlášť nebo je v programu prozatím přidána detekce znaku * , který funguje při výpisu jako mezera. Po zadání příkazu:

*./znakbuse DOBRY*DEN 0 2*

bychom měli vidět řetězec jako na obrázku 42.

Obr. 44: zapsání řetězce s mezerou

Na obrázku 44 je v případě písmene D vidět, že při vytváření znakové sady a její interpretace do námi zvoleného způsobu adresace panelu BUSE snadno dojde k drobné chybě. Tu ale není těžké odhalit a opravit. Výhoda grafického zápisu je zřejmá hlavně v tom, že když si dáme tu práci, můžeme implementovat prakticky jakýkoliv druh písma, samozřejmě v případě, že to poměrně nízké rozlišení panelu dovolí. Protože v rámci vnitřní funkce programu, převádíme hexadecimální čísla na posloupnost příkazů pro panel, můžeme využít definovanou znakovou sadu, která bude vyjádřena stejným způsobem, jako znak A na obrázku 39.

Takovéto využití programu znakbuse.c je sice efektní, ale přeci jen ne příliš praktické. Ačkoliv i zápis z příkazové řádky, pomocí protokolu SSH je zadáváním dat přes WWW rozhraní, uplatnění kód najde až ve formě vnitřní funkce programu, který bude požadovaná data načítat z jiného zdroje a následně je posílat této funkci k zobrazení. Využít proto můžeme databázi MySQL a programy popsané v kapitole 3.5

Pomocí programu ctenimysql.c , který načte poslední naměřená data z tabulky meteo získáme naměřené hodnoty, a ty pak funkcí znakbuse vypsat na panel BUSE110. Pro každé použití je ale třeba vzít v úvahu, zda není nutné doplnit znakovou sadu, protože zatím je nekompletní

4 ZÁVĚR

Celý informační portál jsem navrhnul, sestavil a odzkoušel. Jednotlivé části projektu pracují uspokojivě a plní svou funkci. Vývoj hardwarové části byl poměrně obtížný, protože se do určité míry jednalo o reverzní inženýrství. Jakkoliv jsem na projektu strávil mnoho času, pro kompletní funkci je třeba stále ještě vytvořit a nastavit přívětivější uživatelské rozhraní, aby i méně poučený uživatel mohl portál používat. Pokud rozdělíme problematiku projektu na dvě části, tak měření parametrů interiéru se podařilo realizovat zcela. Zapojení je funkční a přístup pomocí sítě Ethernet z jiných zařízení funguje díky použitým webovým technologiím bez problémů. Samotný zobrazovací panel je po hardwarové stránce navržen a realizován. V omezené míře je možné na panel vypisovat textové a číselné informace, ale stále se jedná o fázi testování a celou funkci je třeba odladit.

Další práce na zařízení k plnému splnění požadované funkce spočívá v návrhu softwarového vybavení. Nejlépe knihovny funkcí pro snadný a intuitivní přístup ke grafickým možnostem panelu. Tuto část práce jsem dokončil jen z části, i když se domnívám, že potřebné principy a postupy jsem popsal a programový kód, který jsem pro portál vytvořil je dále rozšiřitelný a použitelný. Jako další námět pro rozšíření funkcí panelu se nabízí automatické monitorování některé z webových aplikací, například výpis zpráv z portálu Twitter, atd. V dokončení celého projektu a jeho rozšíření budu dále pokračovat, protože se jedná o zajímavou problematiku s širokým záběrem napříč technologiemi.

LITERATURA

[1] BUSE S. R. O. Technický popis panelu BUSE BS 110. Blansko, Česká Republika, 1997.

[2] UPTON, Eben a Gareth HALFACREE. Raspberry Pi user guide. Chichester, England: John Wiley, c2012, xiv, 248 p. ISBN 11-184-6446-X.

[3] HEROUT, Pavel. Učebnice jazyka C. 4., přeprac. vyd. České Budějovice: Kopp, 2004, 271, viii s. ISBN 80-723-2220-6.

[4] GM Electronic. GM electronic, spol. s r. o. [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://www.gme.cz

[5] Xilinx.com. Xilinx Inc: All programable technologies [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://www.xilinx.com

[6] Raspberry Pi Wiki. RPi Hub [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://elinux.org/RPi_Hub

[7] Alldatasheet. Electronic component datasheets search engine [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://www.alldatasheet.com

[8] NXP Semiconductors. Products by function [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://www.nxp.com

[9] Raspberry Pi. The Raspberry Pi Foundation [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://www.raspberrypi.org/

[10] Adafruit Industries. Raspberry Pi [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://www.adafruit.com/category/105

[11] Linuxsoft.cz. HORÁČEK, Petr. Raspberry Pi - seriál [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1937

[12] PUTTY: A Free Telnet/SSH Client. TATHAM, Simon. [online]. [cit. 2014-01-06]. Dostupné z: http://www.chiark.greenend.org.uk/~sgtatham/putty/

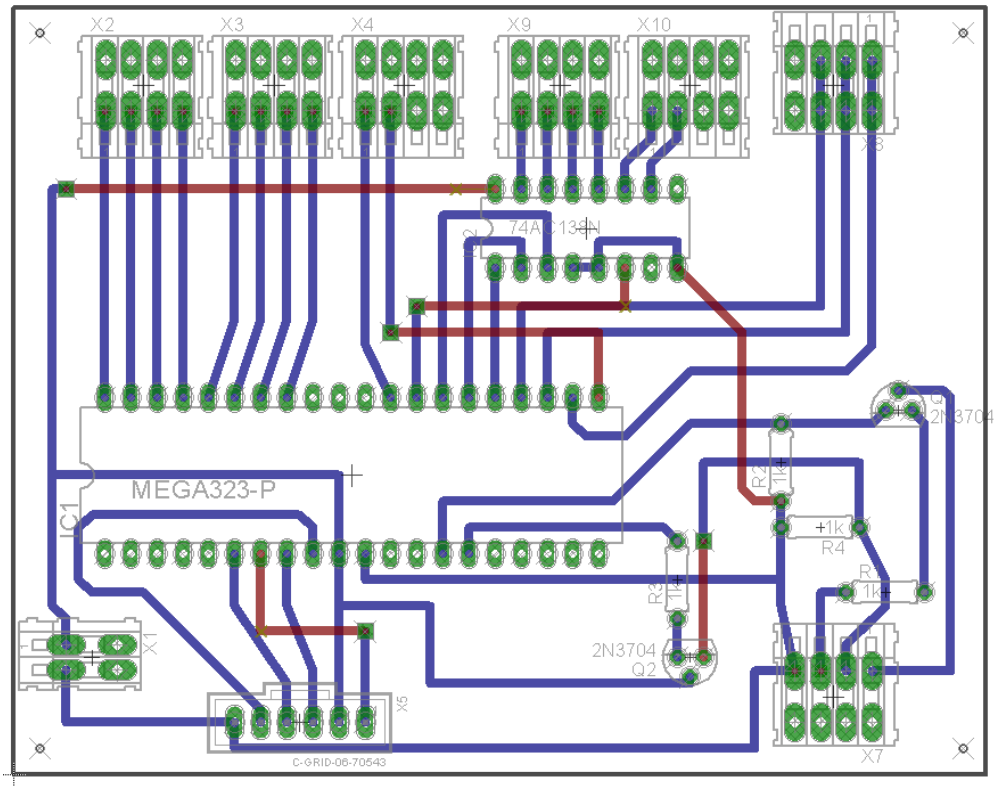
[13] Http://www.debian.org/social_contract: Debian - společenská smlouva. SPI AND OTHERS.[online].[cit. 2014-01-06]. Dostupné z: http://www.debian.org/social_contract Filezilla: the free FTP solution [online]. [cit. 2014-01-06]. Dostupné z: https://filezilla-project.org/

[14] FREE SOFTWARE FOUNDATION, Inc. GCC, the GNU Compiler Collection [online]. [cit. 2014-01-06]. Dostupné z: http://gcc.gnu.org/

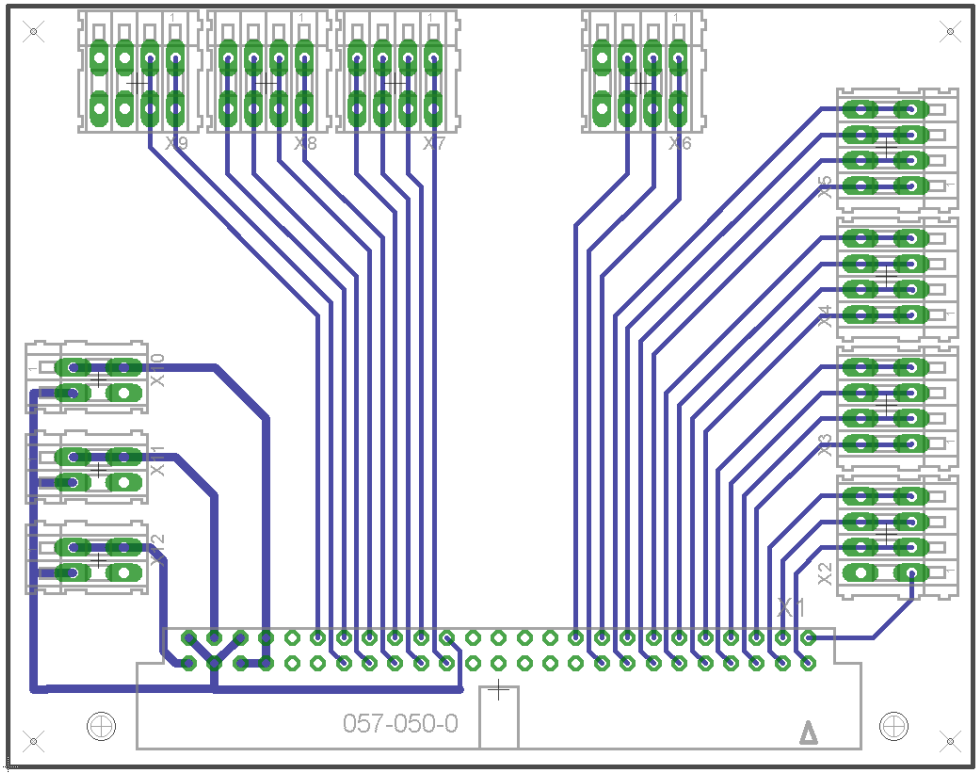
[15] ATmega32: Microcontroller with 32KBytes In-System Programmable Flash. In: [online]. [cit. 2014-01-06]. Dostupné z: http://www.atmel.com/Images/doc2503.pdf

[16] Bosch BMP085 and Raspberry Pi. [online]. 2012 [cit. 2014-01-06]. Dostupné z: http://www.john.geek.nz/2012/08/reading-data-from-a-bosch-bmp085-with-a-raspberry-pi/

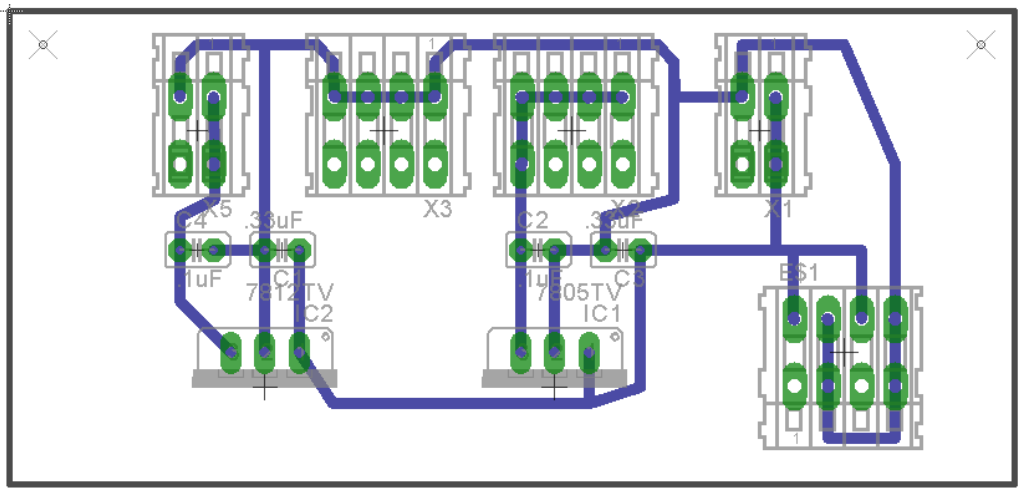
PŘÍLOHY



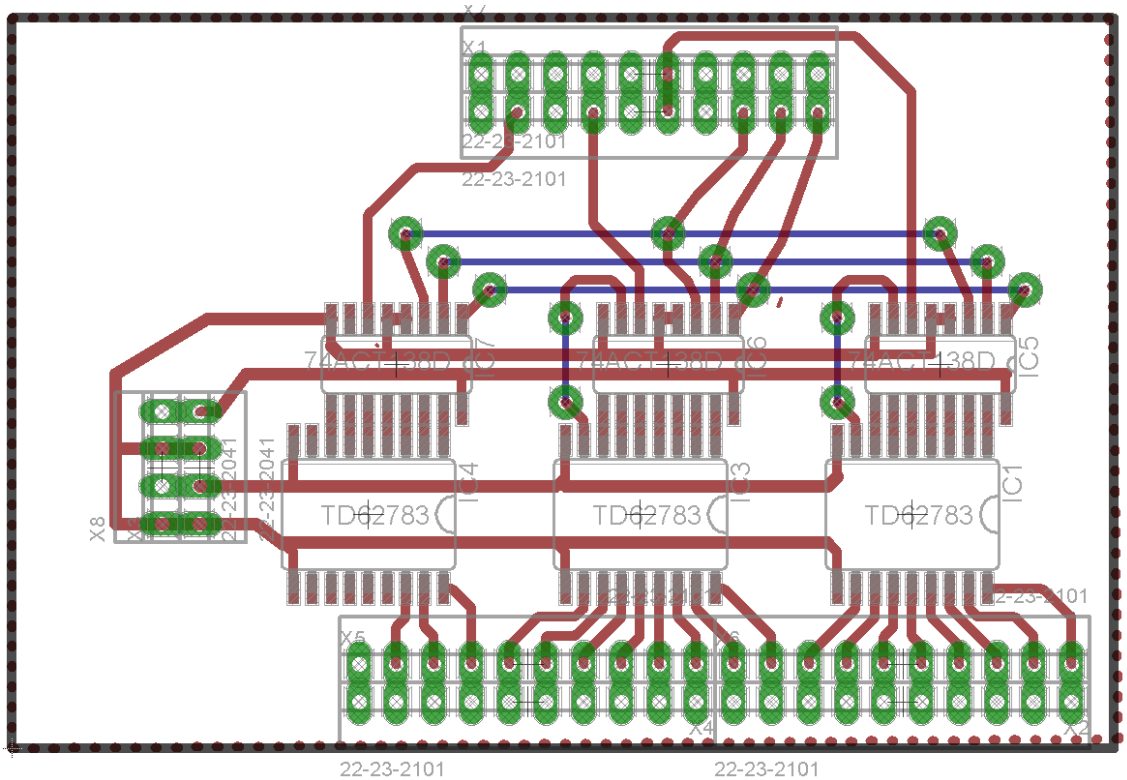
Příloha1: návrh DPS modulu ATmega32



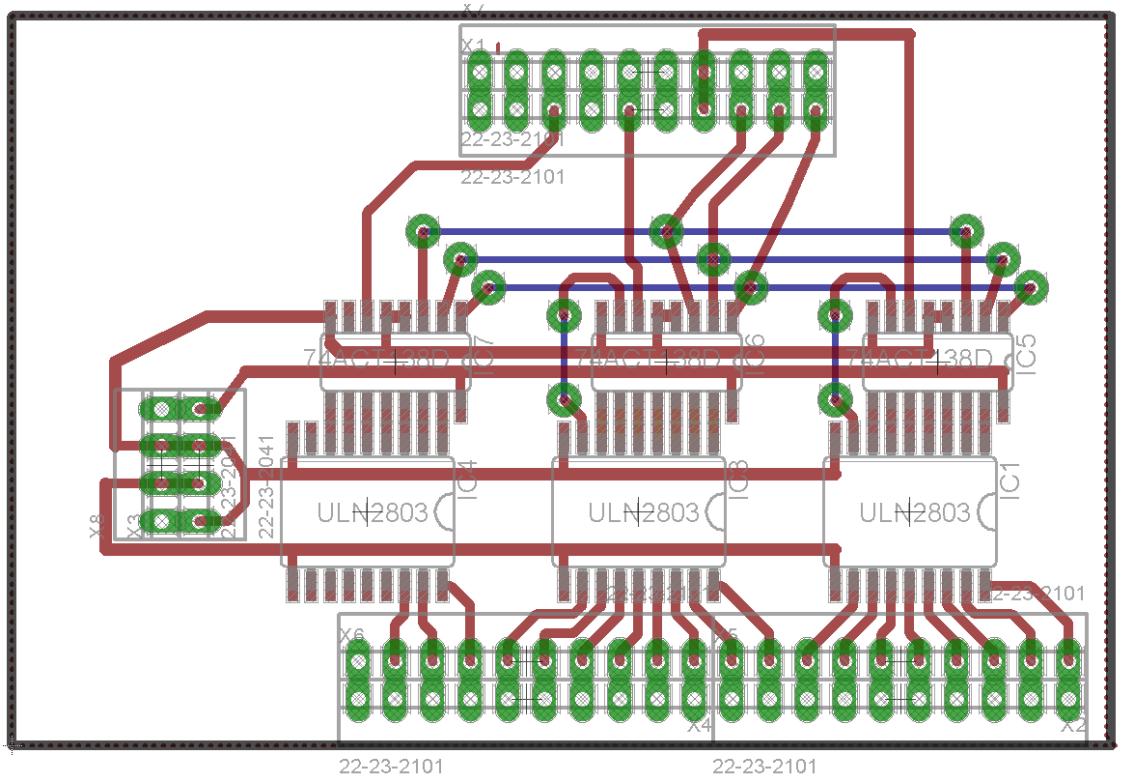
Příloha2: návrh DPS konektoru sběrnice BUSE110



Příloha3: návrh DPS modulu napájení



Příloha4: návrh DPS modulu logiky 1



Příloha5: návrh DPS modulu logiky 2

OBSAH PŘILOŽENÉHO CD

1/ dokumenty:
- 2014_DP_Pilny_Jiri.docx
- 2014_DP_Pilny_Jiri.pdf

2/ ATmega32:
- BUSE_ATmega32.c – zdrojový kód hlavního programu

3/ DPS:
- Podklady pro výrobu DPS

4/ RaspberryPI:
- Webové stránky
- meteo_sql.c – zdrojový kód programu pro měření a zápis dat
- zdrojové kódy programů pro interakci s panelem BUSE