VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY

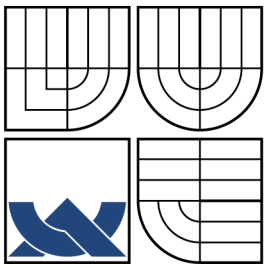
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

TERMINÁLY PRO OVLÁDÁNÍ ZAŘÍZENÍ V RODINNÉM DOMĚ

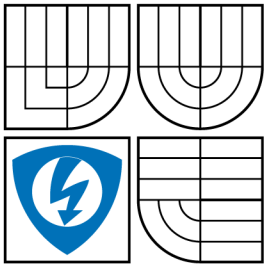
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ MACHÁČEK
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

TERMINÁLY PRO OVLÁDÁNÍ ZAŘÍZENÍ V RODINNÉM DOMĚ

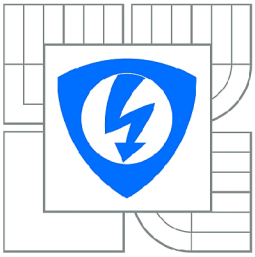
TERMINALS FOR DEVICE CONTROLL IN FAMILY HOUSE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ MACHÁČEK
AUTHOR

VEDOUCÍ PRÁCE — ING. TOMÁŠ MACHO, PH.D.
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Jiří Macháček	***ID:***	106608
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Terminály pro ovládání zařízení v rodinném domě

POKYNY PRO VYPRACOVÁNÍ:

1. Navrhněte koncepci sítě terminálů pro ovládání zařízení a spotřebičů v rodinném domě.
2. Vyberte vhodný mikroprocesor pro realizaci terminálu.
3. Navrhněte a realizujte terminál včetně desek plošných spojů. Vytvořte výrobní dokumentaci.
4. Pro terminál vytvořte softwarové vybavení a odlaďte jej.

DOPORUČENÁ LITERATURA:

MATOUŠEK, David. Práce s mikrokontroléry Atmel AT89C20051. 2. vyd. Praha: BEN, 2006. 248. s. ISBN 80-7300-094-6.

Termín zadání:	8.2.2010	***Termín odevzdání:***	31.5.2010

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

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá návrhem hardware i software síťového terminálu pro ovládání spotřebičů v rodinném domě. Je kladen důraz na jednoduchost celého zařízení, na spolehlivost a odolnost proti výpadkům napájení. Zařízení by mělo umožnit textovou komunikaci v objektu mezi terminály i s PC vybaveným vhodným programem.

KLÍČOVÁ SLOVA

terminál, ATmega, ATmega128, PIC18F67J60, Ethernet, ATM12864, AT89C2051

ABSTRACT

This work deals with concept of hardware and software of site terminal for device controll in family house. The device should be simple, reliable and resistant against power failures. The device should also allow users to chat between Terminals or with PC equipped with suitable program.

KEYWORDS

terminal, ATmega, ATmega128, PIC18F67J60, Ethernet, ATM12864, AT89C2051

MACHÁČEK, J. *Terminály pro ovládání zařízení v rodinném domě*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky, 2010. 75 s. Vedoucí práce: Ing. Tomáš MACHO, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Terminály pro ovládání zařízení v rodinném domě“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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 § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Tomáši Machovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

1 Úvod **9**

2 Základní koncepce **10**

2.1 Blokové schéma sítě 10

2.2 Blokové schéma terminálů a koncových zařízení 11

2.2.1 Blokové schéma terminálu 11

2.2.2 Blokové schéma koncového zařízení 14

3 Obvodový návrh terminálu **15**

3.1 Hlavní procesor terminálové jednotky 15

3.2 Displej 17

3.3 Klávesnice 18

3.4 Analogové vstupy a výstupy 22

3.4.1 Vstupy 22

3.4.2 Výstupy 24

3.4.3 Převodník USART/Ethernet 26

3.5 Napájecí zdroj 27

4 Výrobní dokumentace a oživení **31**

4.1 Návrh plošného spoje a jeho realizace 31

4.2 Oživení 31

5 Softwarové vybavení terminálů **33**

5.1 Řídící software klávesnice 33

5.2 Řídící software PIC18F67J60 34

5.2.1 Komunikace mezi mikrokontroléry 35

5.2.2 Komunikace PIC18F67J60 s koncovým zařízením 37

5.2.3 Multitasking v MCU 38

5.2.4 Stavový diagram aplikace 39

5.3 Software ATmega128-16PU 40

5.3.1 Ovládání grafického displeje 40

5.3.2 Komunikace ATmega přes UART 44

5.3.3 Grafické rozhraní 46
5.3.4 Obsluha lokálních senzorů 49
5.3.5 Další funkce terminálu 52
5.3.6 Členění kódu 52

6 Simulátor Koncového zařízení **54**
6.1 Grafické rozhraní 54
6.2 Pracovní jádro programu 55
6.3 Komunikace programu s okolím 55

7 Závěr **57**

Reference **58**

Seznam symbolů, veličin a zkratek **60**

Seznam příloh **62**

A Kompletní schéma terminálu **63**

B Výrobní dokumentace **65**

C Fotografie výrobku **72**

SEZNAM OBRÁZKŮ

2.1 Blokové schéma sítě . . . 10
2.2 Blokové schéma zařízení v síti . . . 12
3.1 Fotografie použitého displeje . . . 17
3.2 Tlačítkový blok . . . 19
3.3 Obvodové schéma modulu klávesnice . . . 20
3.4 Schéma analogových snímačů . . . 22
3.5 Schéma analogových výstupů . . . 25
3.6 Schéma zapojení převodníku na Ethernet . . . 26
3.7 Schéma zdroje napájení . . . 28
5.1 Pracovní cyklus klávesnice . . . 34
5.2 Stavový diagram pracovního programu MCU PIC . . . 38
5.3 Stavový diagram aplikace v PIC . . . 39
5.4 Pracovní cyklus displeje . . . 41
5.5 Znak 'A' . . . 42
5.6 Kreslení ploch . . . 43
5.7 Kreslení úseček . . . 44
5.8 Stromová struktura menu . . . 48
5.9 Stavový diagram aplikace . . . 49
5.10 Grafická závislost odporu termistoru na teplotě s vykreslenou aproximační funkcí . . . 51
6.1 Rozložení komponent v aplikaci Simulátor koncového zařízení . . . 55
6.2 Program Simulátor koncového zařízení za běhu . . . 56
A.1 Obvodové schéma klávesnicového modulu . . . 63
A.2 Kompletní schéma terminálu . . . 64
B.1 Horní strana desky terminálu. . . . 69
B.2 Dolní strana desky terminálu. . . . 70
B.3 Deska klávesnicového modulu . . . 71
C.1 Fotografie hotového výrobku . . . 73
C.2 Fotografie vnitřního uspořádání . . . 74

SEZNAM TABULEK

5.1 Přehled příkazů pro ovládání PIC18F67J60 přes UART 35
5.2 Přehled zpráv odesílaných PIC18F67J60 přes UART 36
5.3 Přehled zpráv odesílaných PIC18F67J60 přes ETHERNET 37
5.4 Přehled zpráv pro MCU ATmega128 45
5.5 Přehled druhů výstupů 45
5.6 Zprávy pro ovládání koncového zařízení 46
5.7 Závislost konstanty *k* na nastavení *KeyType* 48
5.8 Tabulka funkčních kláves 49
5.9 Závislost odporu termistoru na teplotě 50
5.10 Tabulka knihoven 53

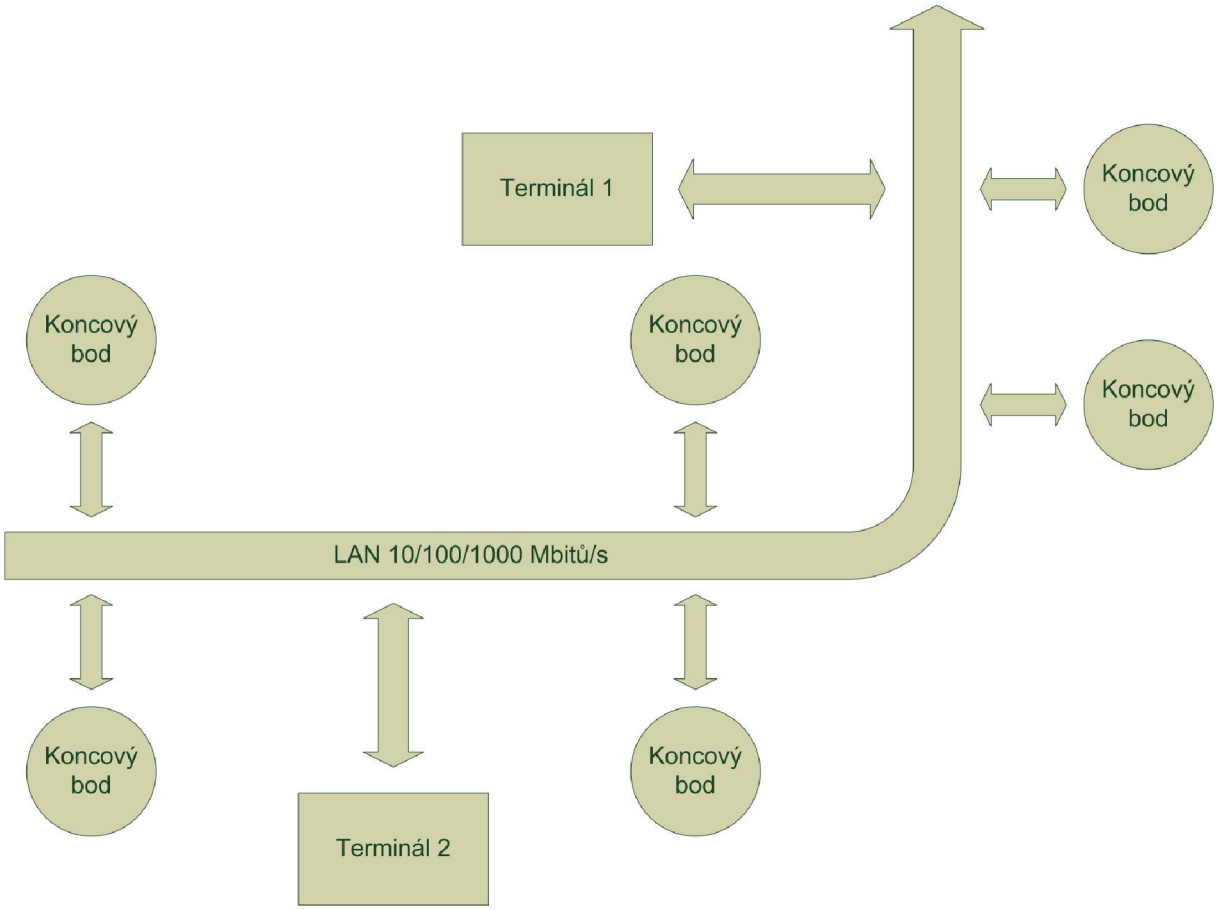
1 ÚVOD

Předložená bakalářská práce se zabývá návrhem a realizací zařízení, jež by umožnilo svému uživateli ovládat koncová zařízení v rodinném domě na dálku. Koncovým zařízením je myšlen ovladač zámku dveří, zásuvkového panelu, případně ovladač světel či klimatizace. Protože tvorba konkrétního koncového zařízení není součástí zadání, bude vytvořen simulátor koncového zařízení v PC, jenž jej bude pro účely demontrace funkce nahrazovat. Zařízení by mělo být jednoduché na obsluhu a montáž a univerzální natolik, aby bylo možné jeden terminál nahradit jiným bez potřeby složitější konfigurace.

Výrobek by dále měl umožnit textovou komunikaci po celém domě a zajistit tak jeho obyvatelům komfortní dorozumívání.

2 ZÁKLADNÍ KONCEPCE

2.1 Blokové schéma sítě



Obrázek 2.1: Blokové schéma sítě

Schéma navržené koncepce celé sítě je uvedeno na obrázku 2.1.

Při volbě komunikačního rozhraní jsem vycházel z předpokladu, že zařízení budeme používat v prostředí s možným výskytem rušení. Bylo tedy nutné vybrat vhodné komunikační rozhraní. Za dobré kandidáty jsem považoval sběrnice RS422, RS423, RS485, CAN, TWI a RS232 nebo Ethernet (Více informací o sběrnicích lze nalézt např. v [3]). Průmyslová sběrnice by byla optimální, pokud bychom zařízení umisťovali do továrny nebo jiného velmi náročného prostředí. Protože ale počítáme s nasazením zařízení do rodinného domu, není toto řešení příliš šťastné. V moderních domech je totiž v dnešní době umístěn rozvod Ethernetu, proto se jeví poměrně zbytečné vést po domě další vodiče,

například s RS-485. Výhodou Ethernetu je snadné použití v PC i v mikrokontroléru. Jeho další výhodou je možnost nasazení protokolů z rodiny TCP/IP a tedy i velice široké možnosti při tvorbě zařízení (HTTP, SMTP, FTP apod.).

Dále bylo třeba rozhodnout, jaká konkrétní zařízení budeme do sítě připojovat. Zpočátku jsem předpokládal, že terminál bude obsahovat vstupně výstupní porty pro komunikaci s vnějšími zařízeními (obsluha zásuvek a světel), ukázalo se ale, že tak stoupá složitost terminálu a tím klesá jeho spolehlivost.

Za nejlepší řešení považuji síť složenou z terminálových jednotek a koncových zařízení. Terminálová jednotka by vlastně byla kontrolním bodem, který by měl být schopen kontaktovat koncové zařízení obsluhující konkrétní zásuvku, klimatizační jednotku, popřípadě ventil topení. Uživatel by díky terminálové jednotce měl být schopen zjistit stav těchto zařízení a nastavit jeho jednotlivé atributy. Takto koncipovaná síť také nabízí možnost textové komunikace mezi dvěma terminály popřípadě mezi terminálovou jednotkou a PC.

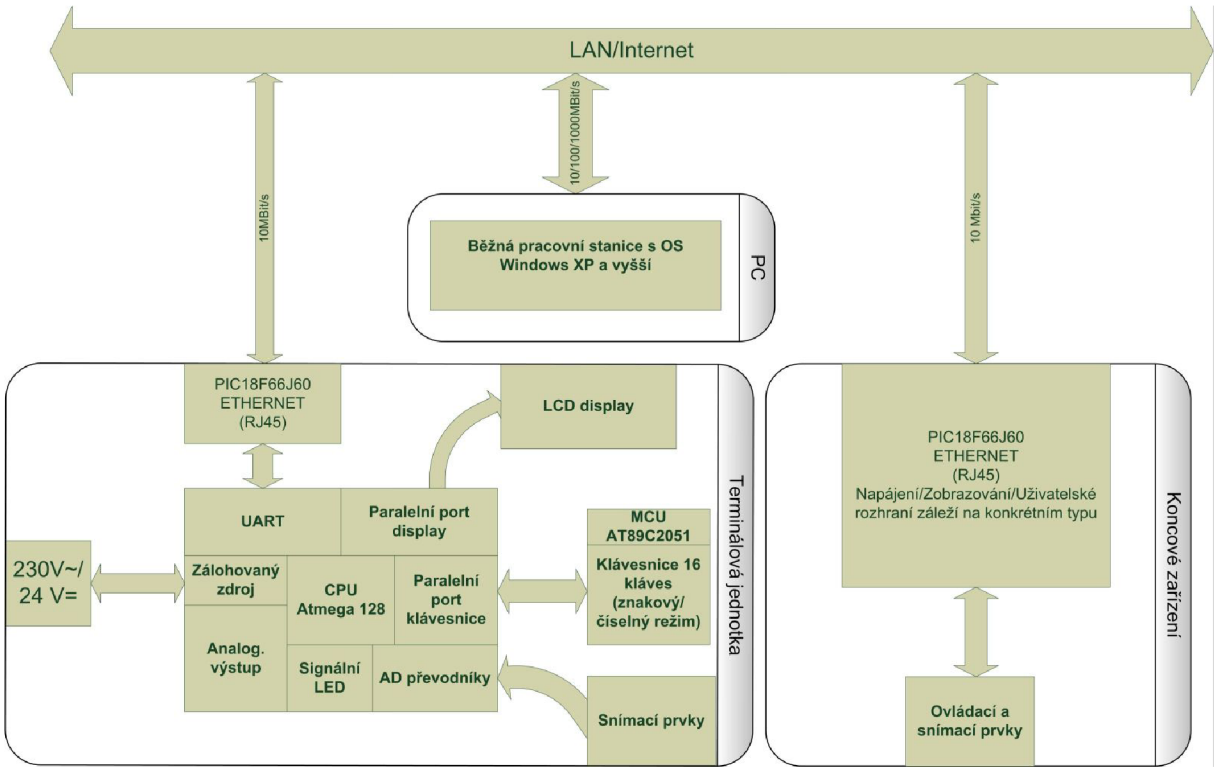
Další výhodou řešení je fakt, že terminálová jednotka bude univerzální, lze ji kdykoli vyměnit za jinou. Pokud bychom použili terminál s vestavěnými periferiemi, tato akce by se značně zkomplikovala. Navíc, pokud je koncový bod oddělený od terminálových jednotek, je jeho výroba jednodušší. Stačí vytvořit vhodný komunikační protokol a poté již na konkrétním konstrukčním řešení prakticky nezáleží.

Tvorba koncového zařízení není součástí mého úkolu, pro účely testování tedy vytvořím jednoduchý simulátor v PC.

2.2 Blokové schéma terminálů a koncových zařízení

2.2.1 Blokové schéma terminálu

Blokové schéma terminálu je zobrazeno na obrázku 2.2. Z obrázku je patrné, že jako jádro celého terminálu je zvolen mikrokontrolér firmy Atmel ATmega 128−16 PU. Byl zvolen proto, že nabízí dostatečný výpočetní výkon a velké množství periferií při zachování malých rozměrů a nízké ceně. Jeho součástí je též mnoho výstupů pro různé periferie, umožňuje nám tedy použít paralelní komunikaci i běžnou sériovou komunikaci přes rozhraní TWI a UART. Výhodou tohoto mikrokontroléru je také fakt, že obsahuje 10−ti bitový osmikanálový AD převodník a PWM výstupy pro analogovou komunikaci.



Obrázek 2.2: Blokové schéma zařízení v síti

Konkrétní typ ATmega 128 byl vybrán pro svou velkou FLASH paměť a schopnost uložit až 4 kB dat do paměti EEPROM. Bližší informace o mikrokontroléru lze nalézt v [5], odkud také budu čerpat v následujícím textu.

Pro komunikaci s uživatelem byla zvolena maticová klávesnice, která je k mikrokontroléru připojena přes paralelní port. Je schopna pracovat ve dvou režimech, v režimu číselném a znakovém, které přepíná uživatel/mikrokontrolér podle potřeby. Při návrhu klávesnice jsem se inspiroval mobilním telefonem, na jehož používání si v dnešní době již zvykla většina lidí. Srdcem klávesnice by měl mikrokontrolér AT89C2051 firmy Atmel. Ačkoli na trhu existuje velké množství mikrokontrolérů s obdobnými vlastnostmi, padla volba právě na Atmel, s nímž mám již dobré zkušenosti.

Druhou částí uživatelského rozhraní je LCD displej. Než jsem začal vybírat konkrétní model, rozhodoval jsem se mezi dvouřádkovým LCD displejem o celkovém prostoru 32 znaků, čtyřřádkovým displejem o prostoru 64 znaků a grafickým displejem s rozlišením 128x64 obrazových bodů. Dále jsem od displeje požadoval podsvětlení a možnost ovládání přes paralelní port. Výhodou znakových displejů se standardním řadičem (HD44780 firmy

Hitachi a kompatibilní náhrady) je znaková sada, kterou obsahují naprogramovanou z výroby. Jejich nevýhodou je poměrně malý paměťový prostor pro českou diakritiku a samozřejmě neschopnost displeje zobrazit obrázky. Nakonec jsem se v rámci uživatelského komfortu rozhodl zvolit displej grafický, s rozlišením 128x64 bodů. Je to z toho důvodu, že pokud si programátor vytvoří vlastní znakovou sadu s velikostí znaku 6x8 bodů, je možné na displej umístit až 168 znaků. Další předností tohoto typu displeje je možnost vykreslení jednoduchých obrázků a obohatit tak uživatelské rozhraní. Při vybírání konkrétního typu padla volba na model ATM12864 s řadičem KS0108. Jeho předností byla nízká cena, jeho nevýhodou absence znakové sady v řadiči displeje (na rozdíl od znakových zobrazovačů, viz [8]). Vzhledem k tomu, že naší prioritou je uživatelský komfort a ne jednoduché programování, je tento problém poměrně zanedbatelný.

Pro monitorování aktulání situace kolem terminálové jednotky jsem se rozhodl využít zabudovaný A/D převodník. Vzhledem k tomu, že má rozlišení 10 bitů a je schopen obsloužit až 8 kanálů, je možné jej využít například jako senzor teploty nebo senzor osvětlení

Nevýhodou mikrokontroléru ATmega128 je jeho neschopnost komunikovat přes Ethernet. Pro tento účel bylo tedy nezbytné užít jiný mikrokontrolér a zajistit mezi oběma MCU komunikaci. Za nejvhodnější považuji komunikaci přes UART nebo I2C. Existovalo několik mikrokontrolérů, mezi kterými jsem se rozhodoval, mezi hlavní patřil mikrokontrolér Atmel AT91SAM7, dále mikrokontroléry Freescale Coldfire a Microchip řady PIC18F. Nakonec jsem se rozhodl zvolit PIC18F67J60, důvodem byla implementovaná fyzická vrstva, ostatní zmíněná MCU obsahují pouze MAC, vyžadují tak doplnění o další integrovaný obvod s PHY. Bližší informace o MCU lze nalézt v [11].

Ne nepodstatným problémem, který bylo potřeba vyřešit, byla otázka napájení. Napájení z baterií jako primárního zdroje není vhodné, neboť předpokládám, že zařízení budou v provozu 24 hodin denně. Mikrokontrolér a displej, které využívám, vyžadují stabilní napájení 5V, je tedy nutné využít v napájecím zdroji stabilizátor. Tvorbě schématu zapojení zdroje a návrhu konkrétních součástek se budu věnovat později. Má-li stabilizátor dostatečně dobře pracovat, je třeba použít jeho napájecí napětí o něco větší, než výše uvedených 5V. Výstup ze zdroje musí být bezpečný podle Vyhlášky č. 50/1978 Sb.[4], což znamená, že stejnosměrné napětí na živých částech elektrického zařízení v prostorách normálních a nebezpečných nesmí překročit 60V. Ve svém zařízení předpokládám

napájecí napětí 17−24V z adaptéru s napájením ze sítě. Některé senzory a periferie mohou vyžadovat vyšší napětí, proto se chystám využít třístupňovou stabilizaci, nejprve na 15V, poté na 5V a na 3,3V. Pro zvýšení spolehlivosti celé sítě jsem se rozhodl doplnit zdroj záložní baterií/akumulátorem, který by v případě výpadku napájení zajistil možnost provádět s terminálem základní operace po dobu alespoň několika minut (v závislosti na kapacitě užitých baterií). Terminál nebude obsahovat dobíjení, baterie tedy bude nutné po každém použití dobít v externím nabíječi.

2.2.2 Blokové schéma koncového zařízení

Jak již bylo řečeno v úvodu, hardwarové řešení koncového zařízení záleží na konkrétní aplikaci. Je pouze důležité, aby byl splněn komunikační protokol, který bude řešen v dalším textu.

Obecně je možné říci, že koncové zařízení je prakticky převodník mezi Ethernetem a výstupy pro ovládání zásuvek, ventilů a vstupy pro snímání stavu senzorů.

3 OBVODOVÝ NÁVRH TERMINÁLU

Základní funkční bloky zařízení jsou tyto: jádro terminálu obsahující MCU a nejnutnější pracovní obvody, displej, klávesnice, převodník USART/Ethernet, datová rozhraní pro připojení periferních zařízení, analogové vstupy a výstupy a samozřejmě napájecí zdroj.

3.1 Hlavní procesor terminálové jednotky

Protože jsem navrhl poměrně rozsáhlé schéma, není vhodné je umisťovat do textu celé. Celkové schéma je uvedeno v příloze A a v dalším textu budu popisovat pouze jednotlivé funkční bloky.

Jak již bylo řečeno v popisu koncepce zařízení, jako hlavní mikroprocesor byl zvolen mikrokontrolér firmy Atmel ATmega128.

K činnosti samotného mikrokontroléru je potřeba několik součástek, jejichž použití je nutné nebo doporučené. Budou popsány v této podkapitole.

Aby procesor dosahoval odpovídajícího výpočetního výkonu, je nezbytné, aby byl správně taktován. Ačkoli mikrokontrolér umožňuje užití vnitřního zdroje hodinového signálu s frekvencí max. 8MHz, viz [5], není jeho použití vhodné, neboť předpokládám, že navržený firmware bude vyžadovat vyšší výpočetní výkon. Proto je k hodinovým vstupům připojen hodinový krystal Q_1 s kmitočtem 16 MHz. Tato hodnota je maximální doporučená, s vyšším kmitočtem se již stává program nestabilním. To je v jakékoli aplikaci nežádoucí okolnost. Dle manuálu výrobce musí být krystal doplněn o keramické kondenzátory C_1 a C_2, jejich doporučená kapacita je $C_{1/2} = 20pF$. Aby mohl software terminálu obsahovat funkci měření času, je praktické užít asynchronní hodinový krystal s frekvencí 32 768Hz. Na schématu je označen jako součástka Q_2. Zde podle výrobce není užití vnějších kondenzátorů potřeba.

Napájení terminálu je řešeno v dalších podkapitolách, za zmínku pouze stojí zajímavost, že má mikrokontrolér čtyři napájecí vstupy, dva GND a dva Vcc. Abychom zamezili problémům při zapnutí napájení, kdy se napájecí napětí stabilizuje, je k mikrokontroléru připojen také watchdog MCP 101 (zapojený a dle datasheetu [13] doplněný o kondenzátor $C_6 = 100nF$), který zajišťuje, že program bude rozběhnut až ve chvíli, kdy je napájení stabilní. Tím je zabráněno vzniku jevu, se kterým mám osobní zkušenost, a sice že se

nesprávně vynulují příslušné registry a program ”běží”z náhodného místa a vykonávají se nesmyslné operace. Externí Watchdog je možné odpojit odstraněním propojky JP_6.

Firmware bude do mikrokontroléru nahráván přes JTAG rozhraní, přes toto rozhraní jej také budu ladit. Proto je na desce umístěna patice SV_5 s vyvedenými příslušnými piny. ATmega128 také obsahuje pin pro povolení programování (PEN). Jumperem JP_1 lze zvolit, bude-li pin nastaven do log. 0 nebo 1. S ohledem na budoucí možnost rozšiřování výrobku je vyvedena patice SV_2 pro SPI rozhraní. Ze stejného důvodu je vyveden i I^2C konektor X_{10} s pull−up rezistory $R_{11} = R_{12} = 4700\Omega$.

Dále je k mikrokontroléru připojeno tlačítko SW_1. Tlačítko je připojeno na vstup přerušení INT6, je mu tedy možné přiřadit funkci, kterou je třeba vykonat bezodkladně, například přerušit probíhající akci apod.

K indikaci stavů mikrokontroléru jsou osazeny také LED_1, LED_2 a LED_3 diody, kterými bude indikována například přijatá nepřečtená zpráva, aktivní spojení, numerický mód klávesnice (viz níže), porucha atd. Ke každé diodě je připojen rezistor (R_{38}, R_{39}, R_{40}) omezující proud diodami I_d. Jeho velikost vypočteme z Ohmova zákona. Volím proud diodami $I_d = 10mA$, typický úbytek napětí na LED diodách je $U_d = 1,8V$. Celkové napětí bude rovno napětí výstupu v log. 1, tedy $U = 5V$. Velikost odporů R_{38}, R_{39} a R_{40} tedy bude:

$$R_{38} = R_{39} = R_{40} = \frac{U - U_d}{I_d} = \frac{5 - 1,8}{10^{-2}} = 320\Omega \tag{3.1}$$

Z řady E12 volím rezistor o velikosti 33OΩ.

Další součástky jsou nutné pro činnost A/D převodníku. Stabilita napájení je zajištěna cívkou $L_1 = 10\mu H$ a kondenzátorem $C_5 = 100nF$. Pro stabilizaci referenčního napětí jsem se rozhodl užít Zenerovu diodu D_{21}, její $U_{zd} = U_{ref} = 4,7V$. Proud tekoucí diodou volím $I_{D21} = 500\mu A$, hodnotu R_{33} určíme:

$$R_{33} = \frac{U_{zd}}{I_{D21}} = \frac{4,7}{5.10^{-4}} = 9400\Omega \tag{3.2}$$

Z řady E12 je zvolen rezistor $R_{33} = 10k\Omega$

Obrázek 3.1: Fotografie použitého displeje

3.2 Displej

V zařízení je použit displej ATM12864 (obrázek 3.1). Jeho velkou předností je velká zobrazovací plocha, nevýhodou je absence znakové sady v řadiči displeje viz [8].

Komunikace s displejem probíhá přes osmibitový vstupně výstupní port, je řízena vstupy displeje CS1, CS2, R/W, CLK, I/D. Datové piny jsou připojeny na PORTC, řídící piny na PORTA. Nastavení kontrastu displeje probíhá přivedením napětí na řídící vstup V0 v rozmezí $(-12V; +5V)$. Displej obsahuje generátor záporného napětí $U_g = -12V$ s výstupem na pin V_{out}, proto jsem vytvořil napěťový dělič tvořený potenciometrem $R_1 = 1k\Omega$, kterým lze napětí plynule nastavit v celém intervalu. Napětí bylo také možné nastavit pevným děličem, ale při kolísání napájecího napětí je častá situace, že je displej nečitelný.

Displej vyžaduje napájení $U = 5V$, jsou vyvedeny zvláštní napájecí piny pro podsvětlení a pro vnitřní logiku.

Zařízení je dále vybaveno spínacím okruhem pro podsvětlení displeje. Ten je tvořen tranzistorem T_1 typu BC337, který pracuje ve spínacím režimu s tzv. ”vnucenou betou”. Protože verze datasheetu displeje [8], kterou jsem měl k dispozici, neuvádí proudový

odběr podsvětlení, bylo potřeba jej změřit. Zjistil jsem, že při $U_z = 5V$ je proudový odběr podsvícení přibližně $I_C = 130mA$. Bylo třeba se ujistit, že tranzistor vydrží trvale spínat tento proud, výrobce udává $I_{Cmax} = 0,8A$ v [9]. Postup výpočtu je na základě informací z [2].

Výpočet potřebných rezistorů R_2 a R_3 je následující. Nejprve zvolíme $\beta = 260$, potom platí:

$$I_b = \frac{I_c}{\beta} = \frac{130.10^{-3}}{260} = 5.10^{-4}A = 0,5mA \tag{3.3}$$

Tímto proudem budeme budit vlastní tranzistor. Rezistor R_3 zde slouží jako ochrana tranzistoru, pokud by došlo k odpojení řídícího pinu. Báze tranzistoru tak bude trvale uzemněna a součástka bude vypnuta. Nebude tak hrozit její samovolné zničení zbytkovým proudem. Další funkce odporu R_3 je taková, že rezistor urychlí vypínací děj. Jeho správná volba je obvykle otázkou inženýrského citu. Je důležité, aby příliš nezatěžoval výstup z mikrokontroléru a zároveň nebyl příliš velký, čímž by ztratil význam. Volím tedy $R_3 = 3k\Omega$. Platí:

$$I_{R3} = \frac{U_{be}}{R_3} = \frac{0,9}{3000} = 0,3mA \tag{3.4}$$

kde hodnota U_{be} je zjištěna z datasheetu příslušného tranzistoru [9]. Proud rezistorem R_2 bude tedy součet proudů I_{R3} a I_b, napětí U_{R2} bude rozdíl U a U_{be}. Hodnotu rezistoru tedy vypočteme:

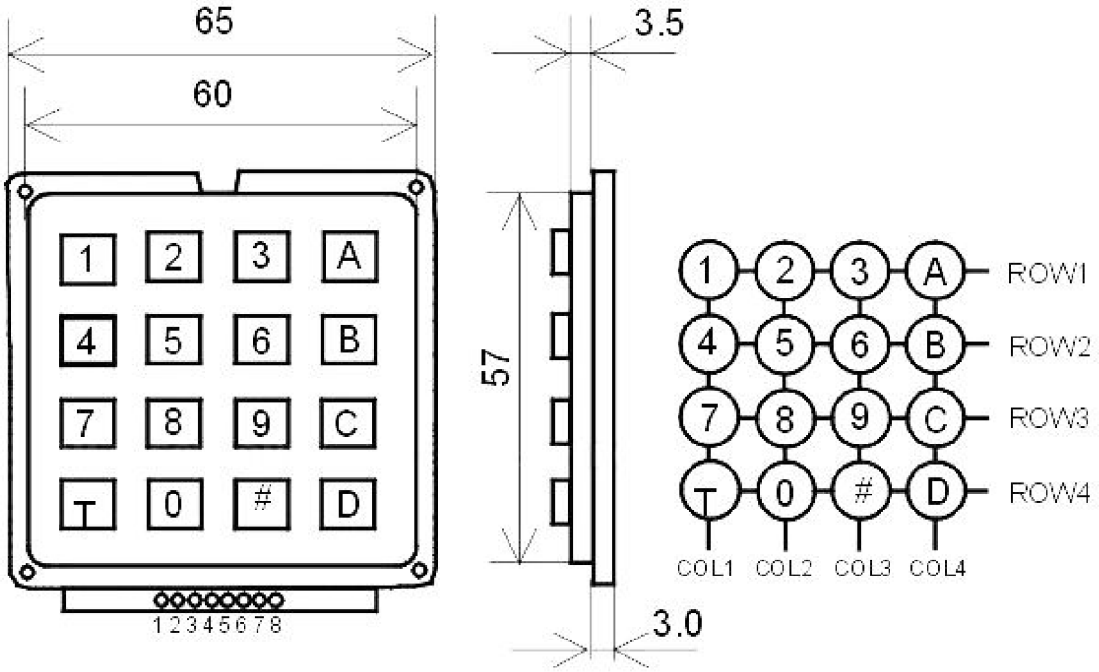
$$R_2 = \frac{U - U_{be}}{I_{R3} + I_b} = \frac{5 - 0,9}{(0,3 + 0,5).10^{-3}} = 5125\Omega \tag{3.5}$$

Odpor R_2 slouží k omezení bázového proudu. Z řady E12 bude na výsledné desce umístěn odpor $R_2 = 4700\Omega$.

Propojení displeje s deskou plošného spoje je realizováno pomocí patice typu MLW20 s dvaceti výstupními piny, což bylo optimální, protože displej disponuje 20 piny. Patice je označena jako *DISP*.

3.3 Klávesnice

Vlastní klávesnice je hotový tlačítkový blok zapojený do matice o rozměrech 4x4 tlačítka. Princip čtení dat z takovéto klávesnice je následovný (viz obrázek 3.2 převzatý z [19]):



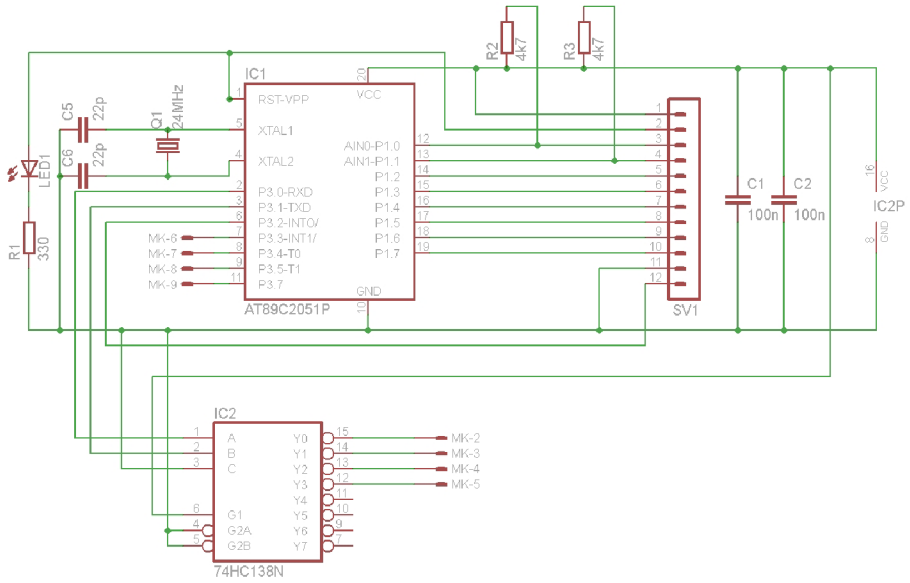
Obrázek 3.2: Tlačítkový blok

Stisknutím klávesy dojde ke galvanickému propojení vývodů příslušného řádku a sloupce. Algoritmus je pak takový, že na všechny sloupcové vstupy přivedeme log. 1 a poté vynulujeme první sloupcový vstup. Na vstupu mikrokontroléru nastavíme všechny řádkové vstupy do log. 1 a zjišťujeme, který vstup je ve stavu log. 0. Takto získáme souřadnice stisknutého tlačítka. Je logické, že je nutné mít k ovládání tlačítkového bloku o velikosti 4x4 tlačítka k dispozici 4 vstupy a 4 výstupy.

Informaci o stisknutém tlačítku je vhodné zjišťovat co nejčastěji a tento postup čtení je poměrně zdlouhavý. Proto bylo vhodné vytvořit samostatnou desku plošného spoje s mikrokontrolérem, který se bude starat o čtení a zpracování dat z tlačítkového bloku a odešle do ATmega pouze finální data. Výhodou je i možnost implementace algoritmu, který bude napodobovat funkci klávesnice mobilního telefonu.

Funkčním jádrem klávesnice je mikrokontrolér firmy Atmel, konkrétně typ AT89C2051. Tento mikrokontrolér obsahuje dva vstupně/výstupní porty, dále nám umožňuje komunikaci přes sériovou linku a zpracování analogového signálu s pomocí AD převodníku. Pro nás je ovšem klíčová možnost paralelní komunikace s pomocí I/O portů a také nízká cena pohybující se okolo 30 Kč. Mikrokontrolér v sobě obsahuje 2kB programové

FLASH paměti a malé množství EEPROM, které ale také v této aplikaci není využito [1]. Tento mikrokontrolér vyžaduje vnější hodinový krystal Q_1, maximální frekvence je 24MHz. Dle datasheetu je potřeba užít externí kondenzátory $C_5 = C_6 = 22pF$.



Obrázek 3.3: Obvodové schéma modulu klávesnice

Datová komunikace s ATmega128 bude probíhat přes P1, komunikační protokol bude řešen v dalších kapitolách. Předpokládám, že dolních 6 bitů portu bude využito pro vystavení kódu klávesy, lze tak rozlišit 64 kódů kláves. Dva horní bity budou užity jako vstupy a budou sloužit k potvrzení převzetí informace hlavním MCU a nastavení znakového nebo číslicového režimu.

Dále je výhodné využít jeden bit z P3 pro vyvolání přerušení MCU ATmega. K tlačítkovému bloku je MCU připojeno přes P3. Bylo řečeno, že jsou potřeba 4 vstupy a 4 výstupy. Jako vstupy použijeme dolní 4 bity P3.

Jelikož ale na výstupu P3.6 je připojen výstup komparátoru (viz [1]) a pin není fyzicky k dispozici, bylo nutné nějak rozšířit výstupy mikrokontroléru. Výstupy jsou rozšířeny s pomocí binárního dekodéru 1 z N, konkrétně verze s negovanými výstupy. Vybral jsem model 74HC138N. Má tři adresové vstupy, nám stačí dva (potřebujeme 4 výstupy), vstup C tedy trvale uzemníme. Tak získáme možnost volit mezi prvními čtyřmi výstupy. Tyto výstupy jsou připojeny na sloupce tlačítkového bloku. Celé zapojení dekodéru je reali-

zováno na základě [18].

K indikaci aktivního RESET signálu je užita LED dioda označená jako LED_1. K diodě je připojen rezistor omezující proud diodou I_d. Jeho velikost vypočteme z Ohmova zákona. Volím proud diodou $I_d = 10mA$ a typické pracovní napětí $U_d = 1,8V$. Celkové napětí bude rovno napětí výstupu v log. 1, tedy $U = 5V$. Velikost odporu R_1 tedy bude:

$$R_1 = \frac{U - U_d}{I_d} = \frac{5 - 1,8}{10^{-2}} = 320\Omega \tag{3.6}$$

Z řady E12 lze zvolit velikost 300Ω nebo 330Ω, zvolil jsem velikost 330Ω.

Důležité je i užití rezistorů R_2 a R_3. Kvůli vnitřní konstrukci mikrokontroléru totiž chybí vnitřní pull−up rezistory na prvních dvou pinech portu P1, nebylo by tedy možné vybudit je do log. 1. Vnitřní pull− up je řešen tranzistorovým zdrojem proudu, viz [6] s konstantním proudovým výstupem $I_p = 0,1mA$. Považoval jsem za vhodné užít pull−up o takové velikosti, aby zde při $U = 5V$ tekl stejný nebo o něco větší proud. Výpočet tedy je:

$$R_2 = R_3 = \frac{U}{I_p} = \frac{5}{10^{-3}} = 5000\Omega \tag{3.7}$$

Z řady E12 lze zvolit rezistory s hodnotou 5k1 nebo 4k7. Zde považuji za výhodnější užít menší hodnotu, zajistí nám to ”tvrdší”log. 1 na výstupu.

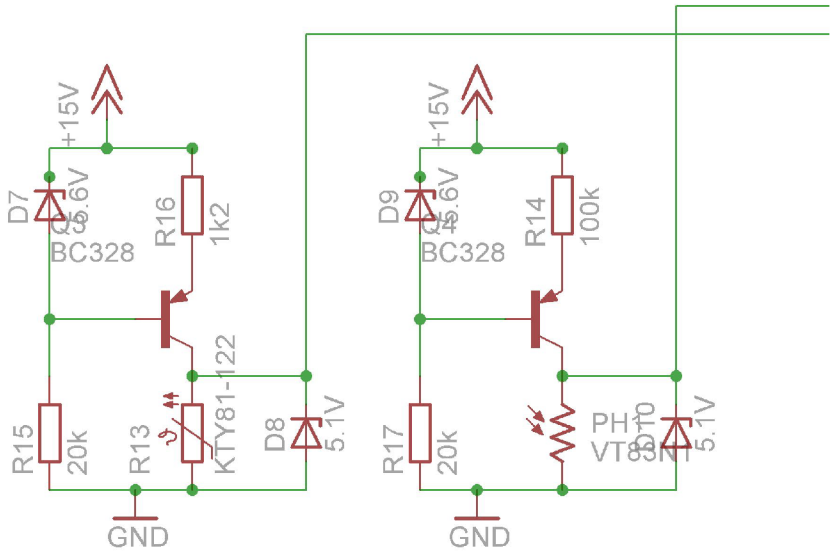
Celý obvod je napájen ze společného zdroje terminálu. Velikost napájecího napětí obou integrovaných obvodů je $U_c = 5V$. Výrobce sice povoluje tuto hodnotu lehce překročit (absolutně maximální hodnota MCU je $U_{cmax} = 6V$ [6]), z dlouhodobého hlediska se ale toto překročení nedoporučuje, protože dochází ke zvýšené zátěži MCU, a tedy ke snížení životnosti celého výrobku. Pro dodatečné vyhlazení napájecího napětí je v blízkosti napájecích pinů umístěn filtrační keramický kondenzátor o velikosti $C_v = 100nF$. Hodnota je zvolena empiricky, a protože se nejedná o klíčovou součástku, je teoreticky možné ji vypustit.

Blok klávesnice je připojen k základní desce terminálu přes lámací lištu, která je označena ve schématu jako prvek SV_1, tlačítkový blok je propojen s AT89C2051 přes konektor MLW10 označený jako MK.

3.4 Analogové vstupy a výstupy

Pro větší komfort uživatelské obsluhy jsem se rozhodl využít jak zabudovaný A/D převodník, tak PWM výstup. Na vstup převodníku jsou připojeny dva senzory pro snímání osvětlení a teploty. PWM výstupy umožňují nastavit na výstup stejnosměrné napětí v rozmezí $(2,3;13,4V)$. Rozmezí je dáno omezeními použitého operačního zesilovače, viz dále. Za pomoci akčního členu s analogovým vstupem (například výkonovým tranzistorem) by bylo možné řídit například osvětlení nebo teplotu v místnosti (ať už na dálku ze serveru nebo lokálně).

3.4.1 Vstupy



Obrázek 3.4: Schéma analogových snímačů

Schéma navržených senzorů a snímačů je zobrazeno na obrázku 3.4. K měření zvolených veličin lze použít několik metod. Nejjednodušší by bylo použití průmyslově vyráběných senzorů například s PWM nebo analogovým výstupem. Toto řešení jsem ovšem považoval za zbytečně finančně náročné. U takového senzoru by sice byla zaručena odpovídající přesnost, ta ale není požadavkem a senzory mají určené veličiny měřit pouze orientačně.

Proto jsem se rozhodl pro odporová čidla - termistor a fotorezistor. U těchto čidel měříme odpor. Pro měření odporu jsem se rozhodl užít Ohmovu metodu. Měřeným rezistorem bude protékat předem definovaný konstantní proud a pomocí A/D převodníku odečteme napětí. Elektrický odpor poté vypočteme dle Ohmova zákona:

$$R = \frac{U}{I} \tag{3.8}$$

Nejprve se budu zabývat návrhem senzoru teploty. Jako zdroj konstantního proudu je užit tranzistor BC328 (součástka Q_3), jako zdroj referenčního napětí v něm byla zvolena Zenerova dioda D_7 s hodnotou $U_{zd} = 5,6V$. Napětí na diodě vytvoříme uzemněním její anody přes velký rezistor R_{15}, který definuje proud diodou. Volím proud $I_d = 0,5mA$, pull−down rezistor má tedy hodnotu přibližně $R_{15} = 20k\Omega$. Proud diodou není klíčový, není třeba jej přesně řešit. Protože výrobce neudává přesnou hodnotu napětí na přechodu báze−emitor tranzistoru a datasheet [10] pouze uvádí jeho maximální hodnotu $U_{BE_{MAX}} = 1,2V$, stanovil jsem napětí $U_{BE} = 0,6V$, což je standardní hodnota. Proud, který náš zdroj bude dodávat jsem vypočetl v závislosti na teplotě, kterou se chystám měřit. Jako zvolený termistor jsem zvolil pozistor KTY81−120. Maximální měřenou teplotu předpokládám okolo 50°C. Odpor pro tuto teplotu jsem odečetl z datasheetu [16], $R_{50} = 1209\Omega$. A/D převodník měří maximální napětí $U_{ref} = 4,7V$, což je referenční napětí dané diodou D_{21}. Proud, který při této teplotě a napětí 4,7V poteče termistorem, bude:

$$I_{50} = \frac{U_{ref}}{R_{50}} = \frac{4,7}{1209} = 3,8mA \tag{3.9}$$

Známe tedy proud, který musí dodávat náš zdroj konstantního proudu. Výpočet je již podle Ohmova zákona snadný:

$$R_{16} = \frac{U}{I} = \frac{U_{zd} - U_{BE}}{I_{50}} = \frac{5,6 - 0,6}{3,8.10^{-3}} = 1315\Omega \tag{3.10}$$

Z řady E12 volím součástku s odporem $R = 1k2$, užijeme tedy tuto, změní se tak nepatrně proud dodávaný zdrojem a maximální napětí na termistoru. Tím se změní i maximální měřitelná teplota.

Nyní vypočtu proud, který ve skutečnosti poteče termistorem. Proud I_t bude:

$$I_t = \frac{U_{ZD} - U_{BE}}{R} = \frac{5,6 - 0,6}{1200} = 0,00416A = 4,16mA \tag{3.11}$$

Odtud lze s pomocí datasheetu zpětně získat maximální měřitelnou teplotu. Vypočtemeli, jaký odpor bude mít součástka při napětí $U = 5V$ a proudu $I_t = 4,16mA$:

$$R_{tmax} = \frac{U}{I_t} = \frac{5}{4,16.10^{-3}} = 1201\Omega \tag{3.12}$$

tomuto odporu odpovídá přibližně teplota $t_{max} = 48°C$.

Protože při překročení maximální teploty by se zvýšilo i napětí na termistoru nad měřitelnou mez a hrozilo by poškození mikrokontroléru, bylo nutné zapojit paralelně s termistorem ještě Zenerovu diodu D_8, která případně stabilizuje příliš vysoké napětí na maximální hodnotě 5,1V.

Při výpočtu senzoru osvětlení jsem užil stejný postup, pouze jsem užil hodnoty odporu součástky z [17]. Jako zdroj proudu je zde užit tranzistor Q_4, jako zdroj referenčního napětí je užita dioda D_9, $U_{zd} = 5,6V$. Pro omezení výstupního napětí je užita dioda D_{10}. Proud diodou D_9 je určen rezistorem $R_{17} = 20k$. Z datasheetu výrobce fotorezistoru je zřejmé, že při denním světle je jeho odpor přibližně $R_{fs} = 10k$, ve tmě až $R_{ft} = 100k$. Je tedy třeba řešit, jaký proud I_t může téct senzorem při úplné tmě, aby napětí na něm bylo maximálně $U_{ref} = 5V$.

$$I_t = \frac{U_{ref}}{R_{50}} = \frac{4,7}{100000} = 47\mu A \tag{3.13}$$

Rezistor R_{14} tedy musí mít hodnotu

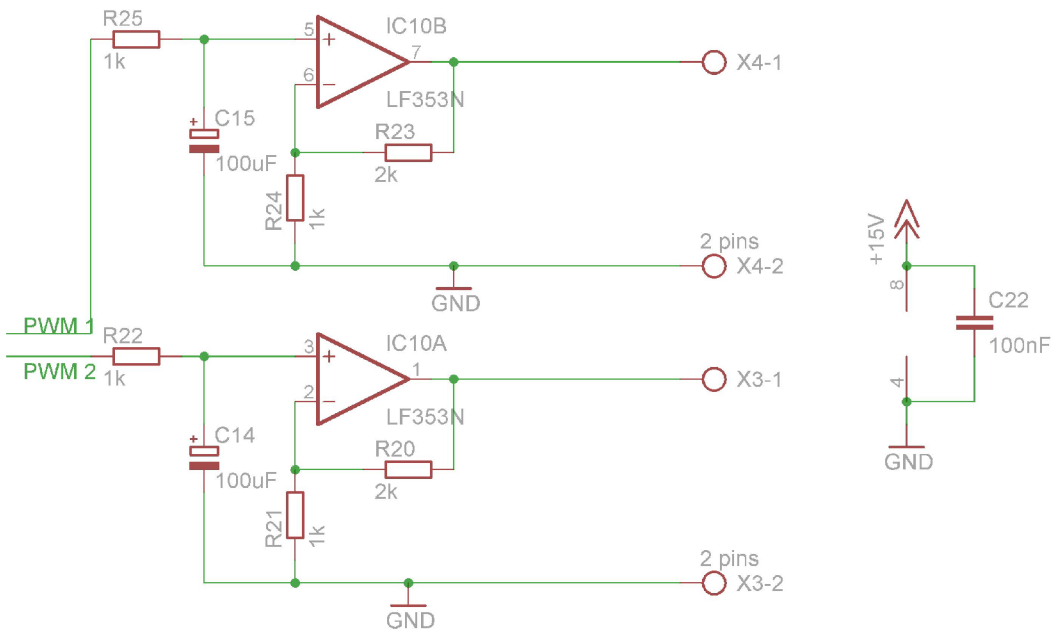
$$R_{14} = \frac{U}{I} = \frac{U_{zd} - U_{BE}}{I_t} = \frac{5,6 - 0,6}{47.10^{-3}} = 106,3k\Omega \tag{3.14}$$

Z řady E12 opět volím rezistor o velikosti $R_{14} = 100k\Omega$.

3.4.2 Výstupy

Schéma zapojení analogových výstupů je zobrazeno na obrázku 3.5. Analogové výstupy jsou řízeny PWM modulátorem mikrokontroléru, vstupní signály jsou filtrovány RC články (R_{25}-C_{15} a R_{22}-C_{14}) s hodnotou rezistoru $R = 1k$ a kapacitou $C = 100\mu F$, čímž je zajištěno, že se na výstupu operačního zesilovače objeví stejnosměrné vyhlazené napětí. Výstup z RC článku je připojen na vstup operačního zesilovače v neinvertujícím režimu se zesílením daným obecným vzorcem

$$K = (1 + \frac{R_a}{R_b}) \tag{3.15}$$



Obrázek 3.5: Schéma analogových výstupů

Tím je zajištěno, že na výstupu zesilovače se objeví výstupní napětí s hodnotou K-krát větší, než na vstupu, přičemž jeho maximální hodnota bude přibližně 15V.[1] Konkrétně jsem použil integrovaný obvod IC_{10} LF353, jehož výhodou je, že obsahuje dva nezávislé operační zesilovače v jednom pouzdře se společným napájením, bližší specifikaci obvodu lze nalézt v [12].

Pro výpočet odporů v konkrétním zapojení větve $IC_{10}B$ platí $R_a = R_{23}$ a $R_b = R_{24}$. Optimální zesílení je zde $K = 3$, protože při této hodnotě lze teoreticky při vstupním napětí $U_{PWM} \in < 0V; 5V >$ získat výstupní napětí v intervalu $U_{out} \in (0V; 15V)$. Zvolíme-li hodnotu odporu $R_{24} = 1k\Omega$, snadno dopočteme R_{23} dle 3.15:

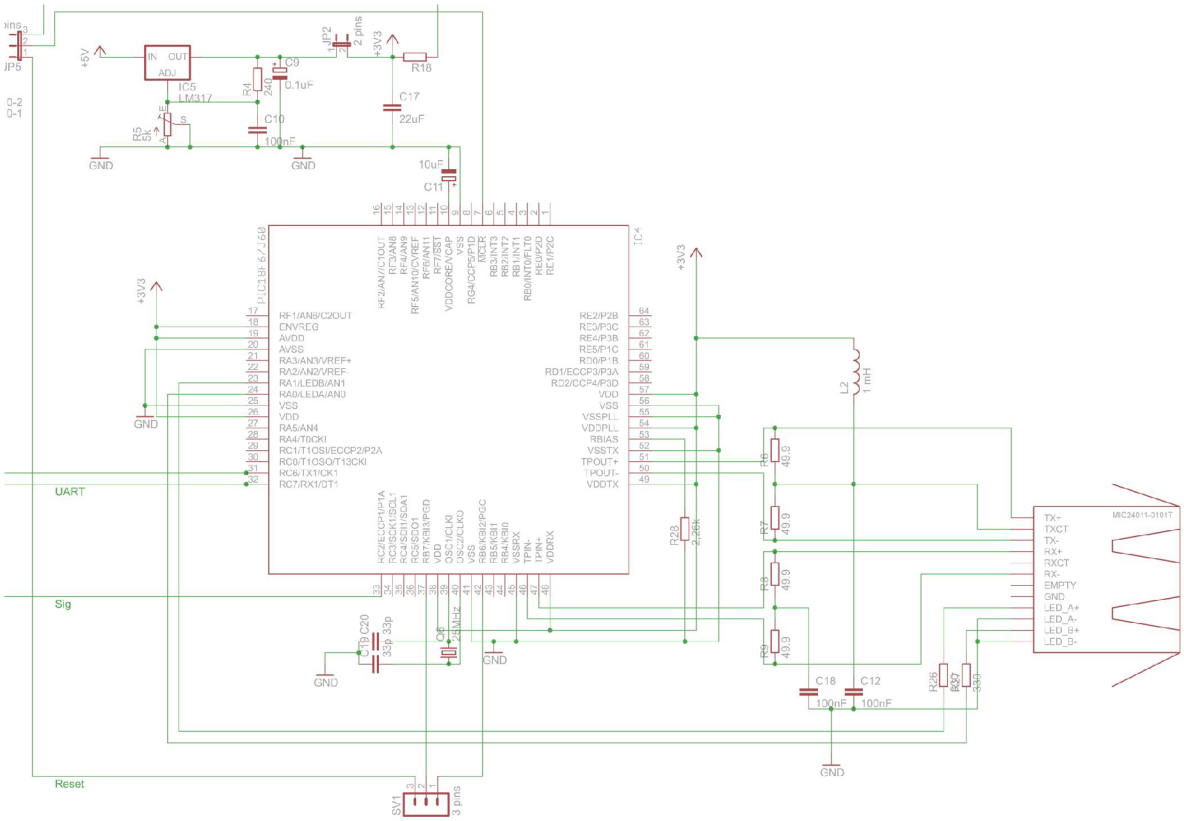
$$R_a = R_{23} = (K-1)R_b = (3-1).1 = 2k\Omega \tag{3.16}$$

Výpočet pro $IC_{10}A$ je naprosto stejný.

Výstupy z operačního zesilovače jsou připojeny ke konektorům X_3 a X_4. Zapojení s operačním zesilovačem má tu výhodu, že výstupy lze zkratovat na libovolně dlouhou dobu bez hrozícího poškození součástek.

[1] 15V je zde napájecí napětí OZ. Výstupní napětí se mu bude limitně blížit v závislosti na odebíraném proudu. Stejně tak kvůli nesymetrickému napájení nikdy nezískáme výstupní napětí 0V.

3.4.3 Převodník USART/Ethernet



Obrázek 3.6: Schéma zapojení převodníku na Ethernet

Jak již bylo výše zmíněno, jako komunikační převodník jsem zvolil mikrokontrolér PIC18F67J60. Důvodem byla implementovaná fyzická vrstva i MAC vrstva Ethernetu (např. AT91SAM7 vyžadoval doplnění fyzické vrstvy), postačilo doplnění několika externích součástek, jako oddělovacího transformátoru integrovaného v zásuvce $MIC24011-0101T$, tlumivky L_2, rezistorů $R_6 - R_9$, kondenzátorů C_{12} a C_{18}. Ethernetová zásuvka také obsahuje signalizační diody doplněné rezistory $R_{26} = R_{27} = 330\Omega$. Poslední součástkou nezbytnou pro správnou funkci Ethernetového modulu je rezistor s odporem $R_{28} = 2,26k\Omega$.

Zapojení PIC je realizováno přesně dle doporučení datasheetu [11]. Pro napájení logických obvodů jádra je použit vnitřní stabilizátor, kterým MCU disponuje. Pro jeho správnou funkci je zapotřebí kondenzátor C_{11}. Jako zdroj hodinového signálu je užit krystal Q_6 s frekvencí 25MHz doplněný kondenzátory s kapacitou $C_{19} = C_{20} = 33pF$.

Komunikační rozhraní je tvořeno výstupy UART a jednou signalizační linkou. Při

propojování bylo třeba zajistit, aby vstupy byly 5V tolerantní. To je u tohoto integrovaného obvodu zajištěno u všech čistě digitálních vstupů, analogové bohužel tuto přednost nemají, signalizační linku není tedy možné připojit na RC0, což by bylo výhodné. Musel jsem zvolit RC2, což zlehka komplikuje tvorbu plošného spoje.

Napájení MCU je řešeno s pomocí stabilizátoru IC_5. Stabilizátor je napájen z 5V větve a jeho výstup je od MCU PIC možné odpojit rozpojením propojky JP_2. Jeho výrobce pro správnou funkci nařizuje použít součástky $R_4 = 240\Omega$ a potenciometr pro nastavení výstupního napětí R_5. Pro zlepšení výstupních parametrů jsou použity kondenzátory C_9 a C_{10}. Pro vyrovnání případných proudových nárazů je užit kapacitor $C_{17} = 22\mu F$.

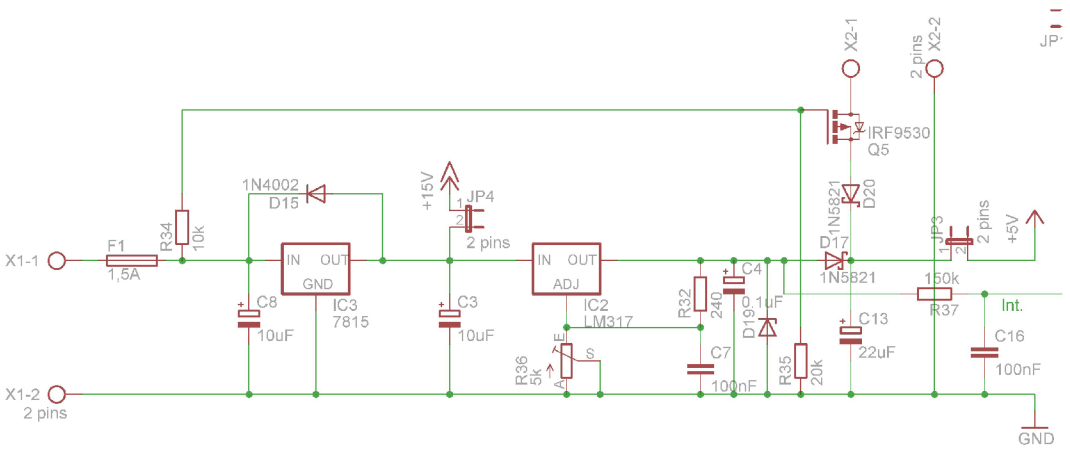
Resetovací obvod MCU lze připojit ”napevno”přes odpor $R_{18} = 10k$ k napájecímu napětí, mikrokontrolér je tak stabilně v aktivním stavu. Druhou možností je řídit tento vstup programátorem z vývojového prostředí (v mém případě konkrétně programátor PIC Kit 2 a prostředí MPLAB IDE). Volbu lze provést propojkou JP_5.

3.5 Napájecí zdroj

Další klíčovou součástí terminálu je napájecí zdroj. Bylo nutné, aby poskytoval dostatečný výkon nejen pro samotné terminálové jádro (mikrokontrolér, klávesnice, displej atd.), ale také pro periferie, které nemají vlastní napájení. Prvním krokem při návrhu bylo přibližné odhadnutí odběru jednotlivých součástí. Spotřeba mikrokontroléru je při plném zatížení přibližně 200-400 mA [5], displej se zapnutým podsvětlením vyžaduje přibližně 200 mA. Klávesnicový modul má předpokládanou spotřebu maximálně 150 mA, převodník pro Ethernet asi 200 mA. Spotřeba senzorů se bude pohybovat kolem hodnoty 10 mA. Z toho vyplývá, že maximální předpokládaný odběr by ve špičkách neměl přesáhnout proud 800 mA. Průměrnou spotřebu ale předpokládám do 300 - 400 mA. Je tedy nutné využít výkonové součástky, které budou schopny tento proud dlouhodobě vydržet.

Schéma zapojení zdroje je na obrázku 3.7. Zdroj lze rozdělit na dvě větve - větev hlavní, připojenou přes adaptér 220V/24V, a větev bateriovou, kde předpokládám využití bateriového zdroje s napětím maximálně 6V (pravděpodobně 4 AA baterie).

Za zmínku stojí, že spotřebu energie bude schopen ovlivnit i programátor nastavením vhodných intervalů pro zhasínání displeje.



Obrázek 3.7: Schéma zdroje napájení

Hlavní větev je tvořena dvěma sériově zapojenými regulátory napětí. První z nich (IC_3) stabilizuje vstupní napětí na 15V, což je vstupní napětí analogových senzorů a napájecí napětí pro operační zesilovače. Stabilizátor bylo třeba podle doporučení výrobce v [7] doplnit i vyrovnávací kondenzátory C_8 a C_3 s kapacitami $C_8 = 10\mu F$ a $C_3 = 10\mu F$. Doporučeno je také užití diody D_{15} pro případ zpětného rázu od aplikace.

Druhý stabilizátor je typu LM317, ve schématu značený jako IC_2 doplněný o diskrétní součástky. Rezistory R_{32} a R_{36} slouží jako zpětná vazba pro regulaci výstupního napětí v rozmezí od 1,2V do 35V. Na výstupu stabilizátoru je připojena Schottkyho dioda, jejímž úkolem je zabránit při napájení z bateriové větve zpětnému průtoku proudu z baterie do stabilizátoru. Tato dioda způsobuje pokles napětí na výstupu o $0{,}5-1{,}1$V, přesná hodnota závisí na konkrétní součástce (Tento stabilizátor je užit právě kvůli této diodě. Lze s ním totiž přesně kompenzovat napěťovou ztrátu způsobenou diodou a vyladit výstupní napětí přesně na 5V). Dále bylo třeba užít kondenzátory C_4 a C_7 s kapacitami $C_4 = 0,1\mu F$ a $C_7 = 100nF$. Pro ochranu proti nechtěnému překročení napětí stabilizátorem je užita Zenerova dioda D_{19} s napětím 5,6V. Při překročení napětí "ořízne"dioda přepětí a v extrémním případě dojde k přepálení pojistky F_1. Oba stabilizátory by měly být schopny vydržet dodávat proud až 1,5A (viz [7] a [15]), což je pro mé zapojení ideální.

Bateriová větev je připojena paralelně k primární větvi. Aby nedocházelo k vybíjení baterie, když to není potřeba, je v obvodu umístěn spínací tranzistor MOSFET, typ IRF9530 označený jako Q_5. Jeho předností je zanedbatelný spínací proud v řádu pikoampér a

typický spínací čas 30ns dle [14]. K uzavření tranzistoru dojde při přivedení kladného napětí na elektrodu Gate, otevření proběhne při jejím uzemnění. Přepínací okruh je tvořen rezistory R_{34} a R_{35}. Hodnota $R_{35} = 20k\Omega$ je volena "inženýrským citem", hodnotu R_{34} vypočteme obdobně jako spínací okruh osvětlení displeje. Na oba rezistory lze pohlížet jako na odporový dělič, protože proud tekoucí do Gate tranzistoru je zanedbatelný. Tranzistor tedy v podstatě řídíme přiloženým napětím, tedy napětím na rezistoru R_{35}. Tranzistor bude uzavřen při napětí $U_u = 12V$, předpokládáme-li tedy napájecí napětí děliče $U_{in} = 18V$, napětí na rezistoru R_{34} tedy bude $U_x = U_{in} - U_u = 6V$. Rezistor R_{34} bude mít velikost:

$$R_{34} = R_{35}\frac{U_x}{U_{in} - U_x} = 20\frac{6}{18-6} = 10k\Omega \tag{3.17}$$

Dále je v obvodu umístěna dioda D_{20} zabraňující přepólování bateriového zdroje.

Došel jsem k závěru, že větev není třeba stabilizovat, protože při použití 4 tužkových baterií dosáhnu v ideálním případě napětí maximálně $U_{max} = 6V$. Je vhodné, aby terminál pracoval po odpojení od sítě co nejdéle a na stabilizátoru by vznikaly zbytečné ztráty (pokles napětí a tepelné ztráty). Na vstup mikrokontroléru se tak při použití spínacího MOSFET tranzistoru a přepólovací diody dostane maximální výstupní napětí 5,3V, což není pro užité součástky nebezpečná hodnota.

Aby měl mikrokontrolér k dispozici informaci o současné situaci napájení, je informace o napájení ze sítě přivedena na vstup pinu vnějšího přerušení. Pro vyhlazení vstupního napětí je vstupu přerušení předřazen RC článek. Jeho konstantu jsem vypočetl následovně:

Přeji-li si eliminovat při napájení z baterie zákmity kratší než $t = 5ms$ při napětí $U = 5V$ a stav vstupu se překlápí z L na H při $u_c = 1,4V$, lze vyjít z následující rovnice:

$$u_c(t) = U(1 - e^{-\frac{t}{\tau}}) \tag{3.18}$$

Vyjádříme-li z rovnice konstantu τ, dostaneme rovnici:

$$\tau = \frac{-t}{ln(1 - \frac{u_c}{U})} = \frac{-0,005}{ln(1 - \frac{1,4}{5})} = 0,015s \tag{3.19}$$

Takovéto časové konstanty dosáhneme vhodnou volbou kapacity. Volím například kapacitu $C = 100nF$, odpor je tedy $R = 150k\Omega$. [2]

[2] Hodnoty součástek nejsou kritické, lze je měnit v jistém rozmezí, protože není kritická doba t.

Napájecí konektor hlavního zdroje je označen jako X_1, konektor umožňuje připojení vodičů utažením příslušného šroubu. Stejným typem konektoru je označen i přívod bateriové větve označený jako X_2.

4 VÝROBNÍ DOKUMENTACE A OŽIVENÍ

Návrh desky plošného spoje (DPS) jsem prováděl v programu Eagle 5.7 Light. Uvádím to zde, protože tato verze nabízí pouze omezenou velikost vytvářené desky, maximální rozměry jsou přibližně 8x10cm, bylo tedy nutné přizpůsobit této velikosti velikost výsledného zařízení. Jsou zde tedy užity SMD součástky, velikost SMD1206. Protože je deska relativně složitá, nebylo možné ji realizovat za pomoci jedné vrstvy. Hlavní deska je tedy navrhována dvouvrstvě, což je dle mého názoru optimální. Vícevrstvé prokovené desky jsou mnohem náročnější na výrobu a jejich cena prudce stoupá s počtem vrstev. Druhou deskou, kterou bylo třeba navrhnout, byl klávesnicový modul. Ten je velice jednoduchý, vyskytují se na něm pouze 2 integrované obvody v DIL pouzdrech a několik periferních součástek. Jeho výroba je tedy možná i v amatérských podmínkách za použití jedné vrstvy.

4.1 Návrh plošného spoje a jeho realizace

Při návrhu jsem se snažil řídit základními pravidly pro tvorbu plošného spoje, jako zachování minimálních rozestupů mezi součástkami, pravoúhlost. Dále jsem se snažil minimalizovat počet prokovů, které snižují spolehlivost desky.

Pro lepší vlastnosti zařízení (odolnost vůči rušení apod.) je užit na desce polygon vyplňující prázdná místa, polygon je svázán se signálem GND.

Navržené desky s rozmístěním součástek jsou zobrazeny na obrázku v příloze B, v elektronické podobě jsou přiloženy k této práci.

4.2 Oživení

Zařízení jsem oživoval během sestavování, postupně po jednotlivých částech. Usnadnilo se tím hledání chyb, ať už hardwarových, tak softwarových.

Mezi tyto chyby patří například prvotní absence pull−up rezistorů R_2 a R_3 na klávesnicovém modulu, kdy jsem původně plánoval jejich funkci nahradit užitím vnitřních pull−upů v ATmega128. Tento plán se bohužel ukázal jako v praxi nepoužitelný, při přenosu stisknutých kláves totiž docházelo k velkému množství chyb.

Jako další nedostatek se projevila absence cívky připojené k středovému výstupu oddělovacího transformátoru. Ačkoli výrobce v datasheetu dovoluje tuto součástku vypustit, je její přítomnost pro komunikaci přes Ethernet naprosto nezbytná.

Další změnou, kterou jsem při oživování provedl, byla úprava resetovacího obvodu MCU PIC18F67J60. Původně jsem měl v plánu řídit resetovací vstup programově z ATmega, tato funkcionalita se ale ukázala být zbytečná a nespolehlivá. Proto je v režimu normální funkce reset připojen přes rezistor na napájecí napětí 3,3V.

5 SOFTWAROVÉ VYBAVENÍ TERMINÁLŮ

Aby terminál správně pracoval, bylo třeba vytvořit řídící software mikrokontrolerů. Vzhledem k tomu, že celé schéma obsahuje 3 mikrokontroléry, je kapitola členěna do tří logických bloků. První z nich se zabývá návrhem firmware pro klávesnici, v další části popíšu návrh firmware pro PIC18F67J60 a poslední podkapitola se bude zabývat návrhem firmware pro MCU ATmega128.

5.1 Řídící software klávesnice

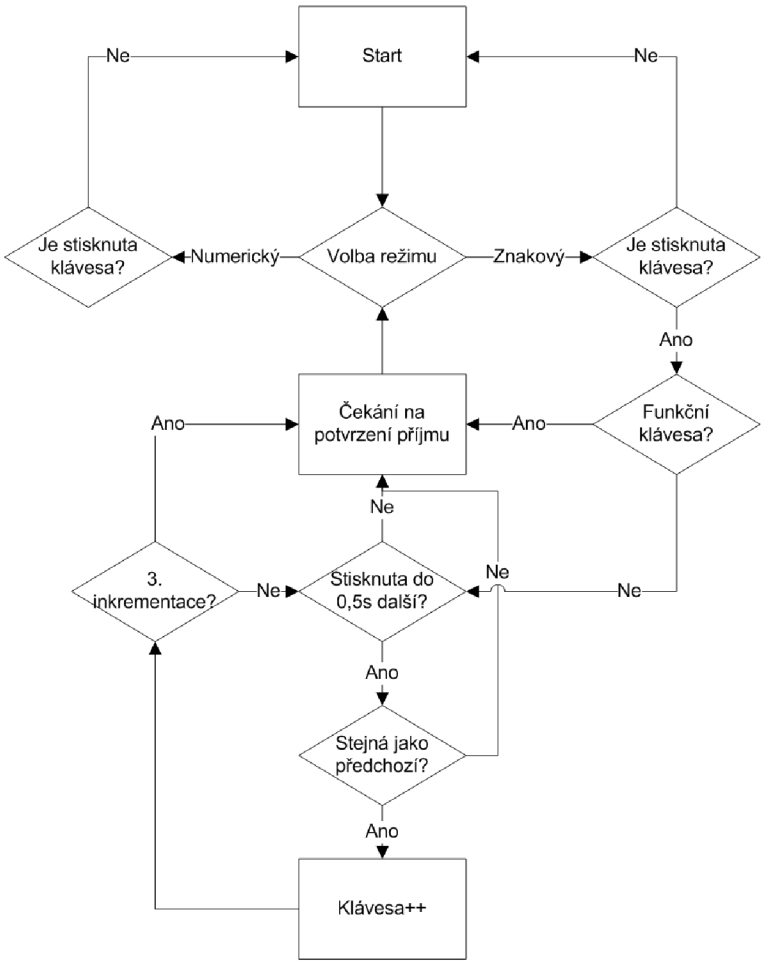
Program v klávesnicovém modulu je poměrně jednoduchý. Rozhodl jsem se jej tvořit v Assembleru z několika důvodů. Prvním z nich bylo, že klávesnicový modul musí pracovat opravdu svižně, aby nevznikaly dlouhé reakční časy při stisku kláves. Dalším (hlavním) důvodem je, že AT89C2051 je velice problematické programovat v jazyce C už proto, že jeho architektura není přizpůsobena vyššímu programovacímu jazyku.

Program běží v jednoduché smyčce, jejíž blokové schéma je uvedeno na obrázku 5.1. Slovně jej lze popsat takto: Na začátku cyklu program kontroluje, je-li nastaven logický bit p1.6. Je-li tomu tak, je požadován numerický režim a postupuje se větví číselného režimu. V opačném případě pracujeme ve znakovém režimu.

V obou režimech je volán podprogram ”fyzické vrstvy”. Tento podprogram nastaví na vstupech multiplexoru adresu požadovaného sloupce, jeden jeho výstup je tak na logické nule. Pokud je stisknuta klávesa, na příslušném řádku se musí objevit log. 0. Čteme tedy piny p3.7, p3.5, p3.4 a p3.3. Je-li některý z nich nulový, známe pozici klávesy. Je-li jich stisknuto víc, ignorujeme klávesy s nižší pozicí. Kód klávesy je uložen do akumulátoru.

Pokud získáme nenulový kód, čteme znovu. Pokud získáme stejný výsledek, je vše v pořádku, nejedná se o zákmit. Čekáme tedy na uvolnění klávesy. V opačném případě vracíme 0.

V číselném režimu je cyklus u konce, čekáme na převzetí klávesy ATmegou a potvrzení na pinu p1.7. Impuls musí být dostatečně dlouhý, aby došlo k jeho zachycení. Ve znakovém režimu cyklicky kontrolujeme ještě cca 0,5 s, jestli je stejná klávesa stisknuta znovu. Stane-li se tak, inkrementujeme kód klávesy a opět čekáme (maximálně 3x), pokud je stisknuta jiná nebo čas vyprší, vracíme původní hodnotu. Protože k zobrazení kódu klávesy



Obrázek 5.1: Pracovní cyklus klávesnice

máme k dispozici pouze 6 bitů, musíme hodnotu klávesy zmenšit. Jako zmenšovací konstantu jsem zvolil číslo 56. V ASCII tato hodnota odpovídá znaku ’8’ a výstupní bitová hodnota je poté při použitých interních kódech menší než 63, datová komunikace se tedy vejde do 6 bitů.

Kompletní program je přiložen k práci.

5.2 Řídící software PIC18F67J60

Protože rodina protokolů TCP/IP je velice rozsáhlá a komplikovaná, rozhodl jsem se neimplementovat ji ručně. Jednak by to bylo velice neefektivní vzhledem k náročnosti na odborné znalosti, jednak protože společnost Microchip nabízí volně ke stažení sadu

knihoven TCP/IP stack. [1].

Tato sada knihoven zaručuje potřebnou stabilitu a spolehlivost síťové komunikace, přičemž nabízí možnost jednoduché konfigurace s pomocí konfigurátoru pro OS Windows. Její výhodou je možnost užití kooperativního multitaskingu, pro který je navržena, lze tedy v zařízení provozovat například jednoduchý HTTP server, případně poštovního klienta.

Tyto funkcionality jsem se prozatím rozhodl nevyužít, prototyp terminálu je schopen navázat spojení k serveru přes TCP protokol, a to podle potřeby udržovat a konfigurovat jeho parametry jako například cílovou IP adresu apod.

5.2.1 Komunikace mezi mikrokontroléry

Komunikace ATmega128 a PIC16F84 probíhá ve dvou režimech. Ty jsou přepínány vystavením log. 1 popř. log. 0 na RC3. Při log. 0 pracuje PIC16F84 jako most UART/Ethernet, kdy veškerá data, která přijme z Ethernetu na portu 65, přepošle na UART ke zpracování MCU ATmega a data z UART přepošle na server.

Příkazy zasílané MCU PIC18F67J60 jsou uvedeny v tabulce 5.1, definované odpovědi jsou uvedeny v tabulce 5.2. Znak "X"v příkazu značí parametr v rozmezí $< 0x00; 0xFF >$.

Tabulka 5.1: Přehled příkazů pro ovládání PIC18F67J60 přes UART

Příkaz	**Popis funkce**
<IXXXX	Nastaví IP adresu terminálu.
<GXXXX	Nastaví výchozí bránu terminálu.
<MXXXX	Nastaví masku sítě
<DXXXX	Nastaví IP adresu primárního DNS serveru
<EXXXX	Nastaví IP adresu sekundárního DNS serveru
<SXXXX	Nastaví IP adresu obslužného serveru
<i	Odešle přes UART aktuálně nastavenou lokální IP adresu
<g	Odešle přes UART aktuálně nastavenou adresu brány
<m	Odešle přes UART aktuálně nastavenou masku podsítě

[1] Download:<http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026439>

Tabulka 5.2: Přehled zpráv odesílaných PIC18F67J60 přes UART

Příkaz	Popis funkce
<S##	Informace o připojení ke koncovému zařízení
<s##	Informace o odpojení od koncového zařízení
<IXXXX##	Vrací aktuálně nastavenou lokální IP adresu
<GXXXX##	Vrací aktuálně nastavenou IP adresu brány
<MXXXX##	Vrací aktuálně nastavenou masku podsítě

Aby nedošlo ke konfliktu v komunikaci mezi jednotkami UART, obsahují pracovní registry jednotek UART v obou MCU stavový bit, který jednoznačně určuje, jestli jednotka přijímá data či nikoli. Pokud je jednotka "volná", teprve poté se začne vysílat. Neexistuje tak důvod, aby se zprávy ztrácely, pokud před vysíláním tento bit zkontrolujeme.

Komunikace probíhá přes rozhraní UART při následujících parametrech spojení: 19200Bd, 1 stop bit, žádná parita. Spojení s vyšší rychlostí se mi bohužel nepodařilo navázat, problém je pravděpodobně v synchronizaci jednotek UART. Rychlost spojení se v obou MCU konfiguruje za pomoci registru SPBRG podle vzorce:

$$SPRBG = \frac{F_o}{64 * BR} - 1 \tag{5.1}$$

kde F_o je pracovní frekvence MCU a BR je požadovaná rychlost. Naše MCU pracuje na frekvenci $F_o = 41,667MHz$. Požadujeme-li tedy komunikační rychlost například $BR = 19200Bd$, je výpočet následující:

$$SPRBG = \frac{41667000}{64 * 19200} - 1 = 33,9086 \tag{5.2}$$

Do registru SPBRG je ale možné zapsat pouze celé číslo, zaokrouhlíme tedy $SPBRG = 34$. Pokud nyní vypočteme zpětně rychlost BR, dostaneme skutečnou komunikační rychlost:

$$BR_s = \frac{F_o}{64 * (SPBRG + 1)} = \frac{41667000}{64 * 34} = 19148Bd. \tag{5.3}$$

Relativní chyba BR je tedy:

$$\delta_{BR} = 100 * \frac{BR - BR_s}{BR} = 100 * \frac{19200 - 19148}{19200} = 0,3\% \tag{5.4}$$

Při rychlosti 19200Bd tedy není rozdíl kritický. Pokud ovšem požadujeme vyšší rychlost, stoupá i δ_{BR}. Dochází tak k přeslechům a při hodnotě například 115200Bd již není komunikace možná. Aby bylo možné komunikovat při této rychlosti, bylo by nutné užít krystaly MCU například o hodnotě 7,768MHz. S ním by ale nepracoval správně ethernetový modul.

5.2.2 Komunikace PIC18F67J60 s koncovým zařízením

Jak již bylo v úvodu podkapitoly řečeno, ethernetová komunikace je řešena implementací sady knihoven TCP/IP Stack, konkrétně verzí 5.1. Při navazování spojení využívá terminálová jednotka na koncovém zařízení port 65. Na tomto portu sama naslouchá, čímž je umožněn příjem zpráv od koncových zařízení, se kterými aktuálně sama aktivně nekomunikuje.

K realizaci spojení je užit TCP protokol, vybral jsem jej pro jeho vlastnosti, mezi které patří především spolehlivý přenos dat a snadné použití nejen v MCU ale i v PC. Ačkoli je velká část řízení toku dat zajištěna knihovnami, bylo nutné implementovat dvě zprávy řízení spojení. Obě jsou uvedeny v tabulce 5.3. První z nich je zpráva příjemci obsahující indentifikační řetězec, podle něhož bude koncové zařízení zjišťovat totožnost připojeného objektu. Rozhodl jsem se další autentifikaci neimplementovat, neboť podle mého názoru není nutná. Další nezbytnou zprávou je zpráva o ukončení spojení. Dokud nebyla implementována, nebylo možné korektním způsobem ukončit spojení v případě překonfigurování terminálové jednotky.

Tabulka 5.3: Přehled zpráv odesílaných PIC18F67J60 přes ETHERNET

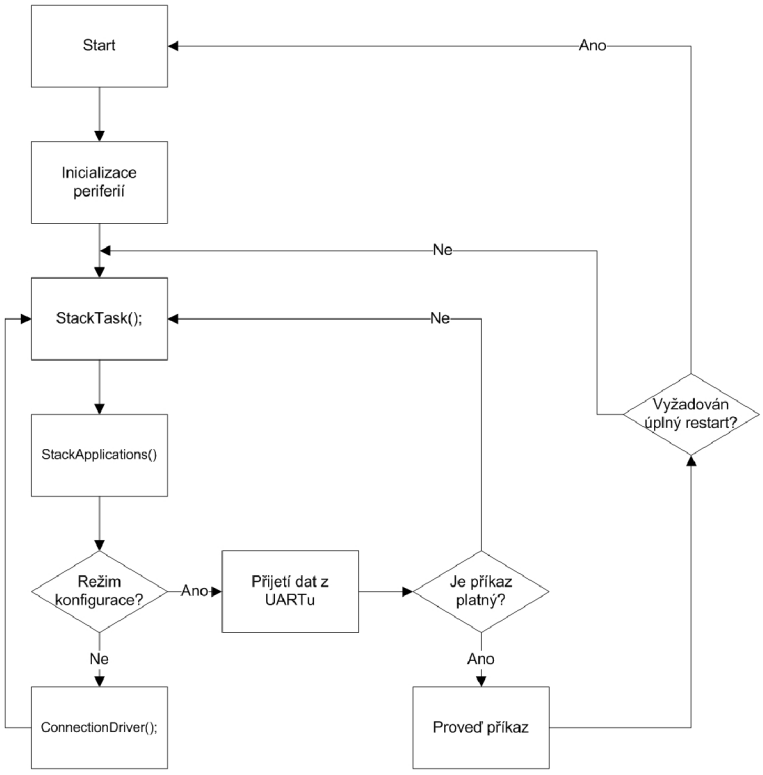
Příkaz	**Popis funkce**
HomeTerm 1.0	Identifikační zpráva po navázání spojení
$$$	Příkaz vzdálené straně k ukončení spojení

Bohužel tímto způsobem není ošetřen problém při úplném výpadku napájení. Spojení tak zůstane ”viset”na straně koncového zařízení a je nezbytné tento problém ošetřit například ukončením spojení po určitém čase.

Po přijetí konfigurační zprávy proběhne měkký reset MCU tak, že s pomocí příkazu ”goto start”se přesuneme na návěští ”start”a program běží od začátku cyklu. Výhodou tohoto řešení je, že při resetu nedochází k fyzickému resetu procesoru, obsah paměti RAM tak růstane zachován.

5.2.3 Multitasking v MCU

Díky kooperativnímu multitaskingu je možné využít v MCU více ”aplikací”, které pracují ”najednou”. Při programování takové ”aplikace”je nezbytné, aby ve funkci nemohlo dojít k dead-locku, a tedy zamrznutí programu. Aplikace při jednom cyklu také nesmí blokovat prostředky příliš dlouho, protože by zpomalila celkový běh MCU.



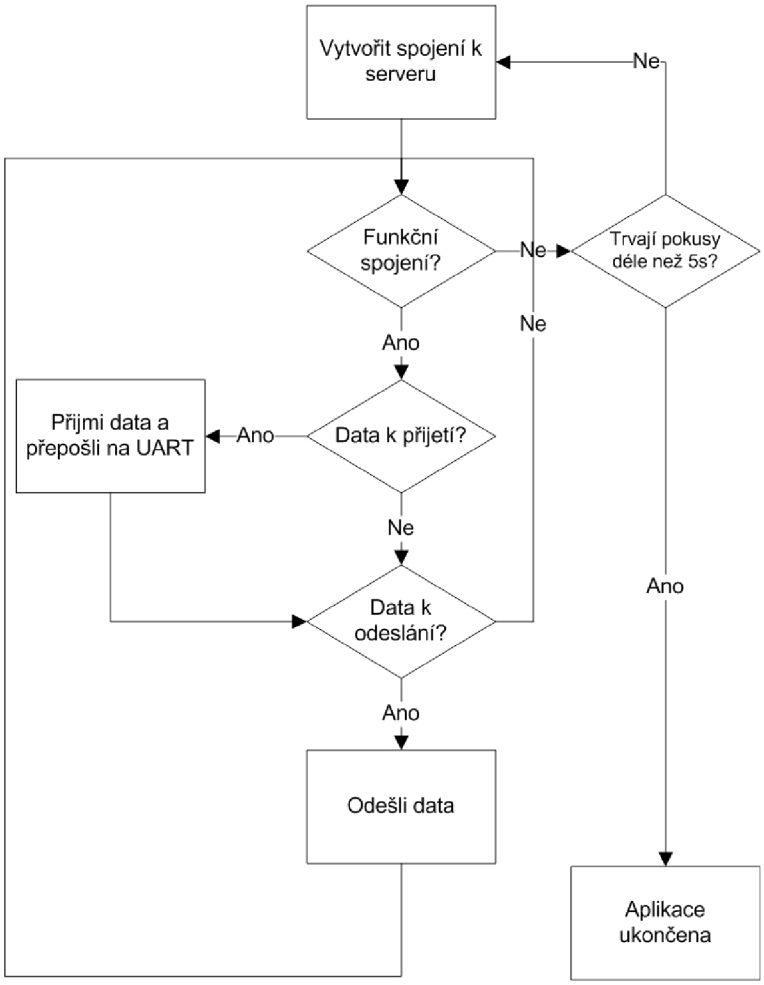
Obrázek 5.2: Stavový diagram pracovního programu MCU PIC

Každá z aplikací pracuje jako stavový automat, který je cyklicky volán při každém cyklu programu, viz obrázek 5.2. Díky tomu je nezbytné užít statické lokální stavové proměnné, které jsou inicializovány pouze jednou při spuštění. Je tak zajištěno, že se aplikace při každém cyklu vrátí do stejného stavu, v jakém byla naposled, a může pokračovat

ve své činnosti.

5.2.4 Stavový diagram aplikace

Jak již bylo řečeno v předchozím textu, každá aplikace pracuje jako stavový automat. Hlavní aplikací, kterou jsem vytvářel pro PIC, byl podprogram na udržování spojení a zajištění funkce mostu mezi UART a Ethernet. Druhá aplikace zajišťuje naslouchání na portu 65 a přeposílání dat z Ethernetu na UART. K tvorbě jsem použil šablonu, která je volně k dispozici na stránkách společnosti Microchip, odkaz je uveden výše. Úkolem první aplikace (nazvané ServerConnection) je po startu procesoru navázat připojení k serveru a to udržovat. Stavový diagram aplikace je zobrazen na obrázku. 5.3



Obrázek 5.3: Stavový diagram aplikace v PIC

Úkolem druhé aplikace (ClientListener) je naslouchat na portu 65 a případným klientům umožnit jednostrannou komunikaci s terminálovou jednotkou.

Považuji za vhodné zmínit, že v aplikaci ServerConnection() je jako vyrovnávací paměť pro příjem dat z UARTu užit buffer o velikosti 250 bajtů. Důvodem byla snaha o co nejmenší fragmentaci zpráv a tím i o zjednodušení programu. Pokud bychom totiž data přijímali po částech, v konfiguračním režimu by se významně zhoršilo jejich zpracování. Museli bychom totiž hledat jejich konec v přijatých datech a následně zprávy spojovat do původní podoby.

V aplikaci ClientListener() je použit podobný buffer, ovšem s velikostí 200 bajtů.

Podrobně je celá implementace firmwaru okomentována ve zdrojovém kódu na přiloženém CD.

5.3 Software ATmega128-16PU

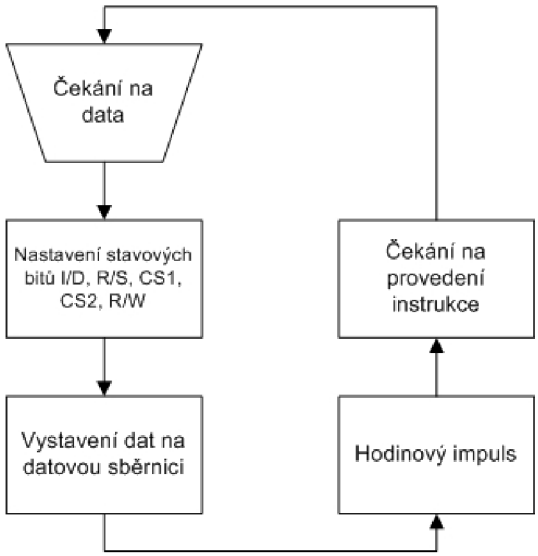
Tato část tvorby softwaru byla pravděpodobně nejkomplikovanější, neboť jsem začínal tzv. od nuly. Bylo tedy nutné nejprve vytvořit ovládací knihovnu pro displej a poté ovládací knihovny pro další periferie. Nakonec bylo nutné sestavit uživatelské rozhraní.

5.3.1 Ovládání grafického displeje

Komunikace s displejem probíhá přes paralelní port. Jednotlivé příkazy lze nalézt v datasheetu[8]. Stavový diagram obecného příkazu je uveden na následujícím obrázku.

Základem celé práce bylo vytvořit nejprve elementární funkce jako funkci generující hodinový impuls, dále čekací funkci apod., poté funkce pro kreslení znakové sady a nakonec i funkce pro vykreslování úseček a dalšíc tvarů. Za tímto účelem jsem byl nucen vytvořit tabulku s definicí znakové sady. Protože je velice obsáhlá, uvedu zde na jednom příkladě kreslení znaku. Kresleme například znak 'A'.

Díky vnitřní konstrukci je displej rozdělen horizontálně na 8 řádků, každý má výšku 8 bodů. Informace do těchto osmi bodů je vždy zapisována současně, nelze tedy vykreslit pouze jeden bod, ale vždy celou osmici. Chceme-li vykreslit pouze jeden bod, musíme nejprve zjistit stav okolních bodů v příslušné osmici. Použijeme k tomu funkci vyčítání, kterou displej podporuje. Vertikálně je plocha rozdělena na 128 sloupečků. Před vykreslováním



Obrázek 5.4: Pracovní cyklus displeje

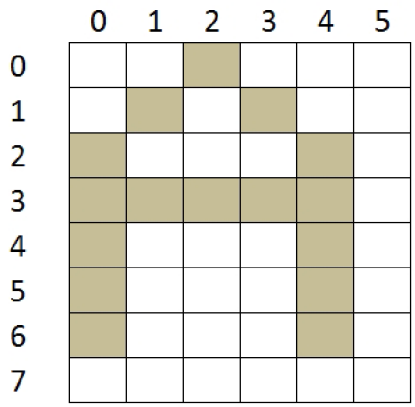
je nezbytné nastavit správnou souřadnici X a souřadnici řádku Y (viz příkazy odesílané displeji).

Po odeslání dat je automaticky inkrementováno počítadlo pozice X, chceme-li tedy kreslit vpravo od poslední pozice, není nutné odesílat znovu souřadnice. Dojdeme-li na konec řádku, počítadlo X přeteče a začíná se kreslit na stejném řádku od začátku.

Každý znak má rozměry 8x6 bodů, z programátorského hlediska je výhodné uložit si data mapy znaků do dvourozměrného pole o rozměrech 107x5 bajtů, neboť chceme mapovat 107 znaků. Každý znak vyžaduje znalost pěti bajtů (pěti sloupečků), šestý bajt je 0x00, což je mezera mezi znaky.

Mazání displeje probíhá podobným způsobem jako kreslení. V podstatě zapisujeme nulové bajty (popřípadě bity) na příslušné adresy.

Zde stojí za to zmínit, že pokud není vykreslovací rutina dostatečně optimalizována, může kreslení větší plochy trvat velice dlouho. Tento jev je možné pozorovat, pokud kreslíme černé obdélníky (případně mažeme obdélníky). Můžeme totiž kreslit buď jednotlivé body, nebo rutinu optimalizovat a části ploch vykreslit po celých osmicích. Jsme tak ušetřeni nutnosti vyčítat data z jednotlivých sloupečků. Celá situace je znázorněna na obrázku 5.6. Kreslíme-li obdélník o rozměrech 7x13 bodů s počátečním bodem $[0;5]$, je zeleně vyznačena oblast, kterou lze vykreslit po sloupcích bez nutnosti vyčítání, modré



Obrázek 5.5: Znak 'A'

jsou oblasti, které je nutné nejprve načíst.

Kreslení úseček je vlastně rozšíření funkcí pro kreslení bodu. Vycházím zde z rovnice přímky známé z analytické geometrie:

$$y = kx + q \tag{5.5}$$

Zadá-li programátor vykreslení bodu z pozice $[x_1; y_1]$ do pozice $[x_2; y_2]$, dojde k výpočtu parametrů k a q následujícím postupem:

Nejprve se zjistí, jestli má úsečka větší rozdíl X-ových nebo Y-ových souřadnic. Je totiž potřeba zajistit, aby koeficient $k <= 1$, tedy aby úhel mezi osou X a úsečkou byl maximálně 45 stupňů. Jedině tehdy lze v celočíselné aritmetice vykreslit spojitou úsečku. Celý problém je zachycen na obrázku 5.7.

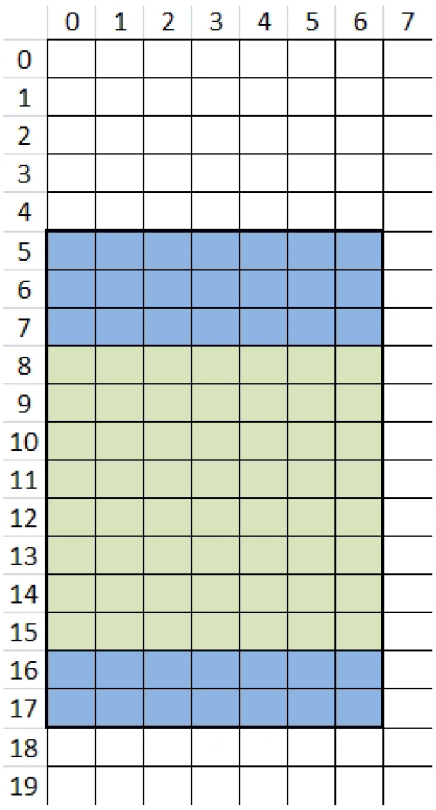
Sklon $k_1 = 1$, sklon $k_2 < 1$ a $k_3 > 1$. Situaci lze vyřešit inverzí funkce, dostaneme tedy funkci

$$x = ky + q \tag{5.6}$$

V praxi to znamená, že obrázek pomyslně otočíme o 90 stupňů. Dostaneme tak pro úhly větší než 45 stupňů opět $k <= 1$. Dostáváme tak dvě sady rovnic, jednu pro úhel $\alpha <= 45\circ$

$$y_1 = kx_1 + q \tag{5.7}$$

$$y_2 = kx_2 + q \tag{5.8}$$



Obrázek 5.6: Kreslení ploch

tedy

$$k = \frac{y_2 - y_1}{x_2 - x_1} \tag{5.9}$$

$$q = y_1 - kx_1 \tag{5.10}$$

Další pro úhel $\alpha > 45\circ$

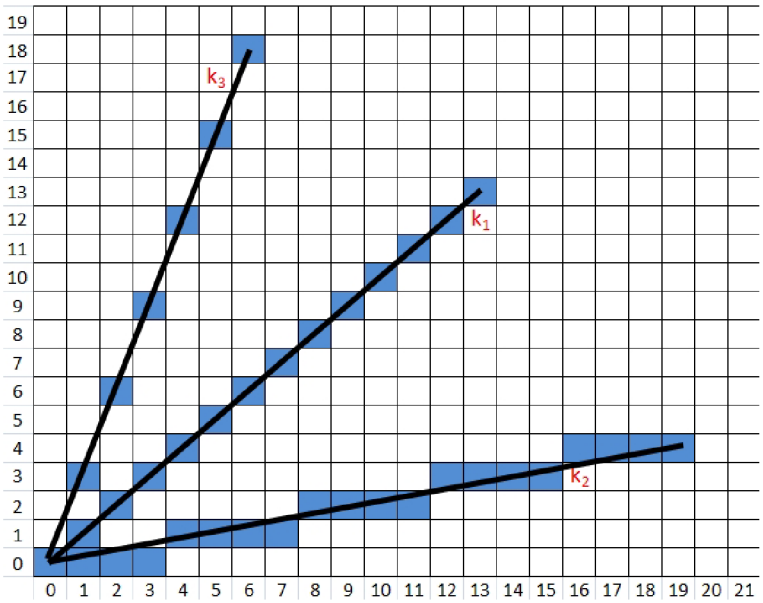
$$x_1 = ky_1 + q \tag{5.11}$$

$$x_2 = ky_2 + q \tag{5.12}$$

tedy

$$k = \frac{x_2 - x_1}{y_2 - y_1} \tag{5.13}$$

$$q = x_1 - ky_1 \tag{5.14}$$



Obrázek 5.7: Kreslení úseček

Protože pracujeme v celočíselné aritmetice, je nezbytné tomu výpočet přizpůsobit. Protože pracujeme s $k <= 1$, ztratili bychom při zaokrouhlení desetinnou část, dostali bychom tedy v každém případě $k = 0$. Před uložením jsem tedy vynásobil koeficient $k_D = k * 100$. To je poté nutné zohlednit ve výpočtech, tedy

$$y = x\frac{k_D}{100_1} + q \tag{5.15}$$

$$x = y\frac{k_D}{100_1} + q \tag{5.16}$$

Na závěr bych rád zmínil, že knihovna je psána tak, aby bylo možné pouhou změnou definic vývodů využít kód při jiném zapojení pinů displeje k MCU.

5.3.2 Komunikace ATmega přes UART

Ovládání jednotky UART je řešeno podle doporučení v datasheetu [5]. Každá zpráva, kterou MCU ATmega dostává musí začínat znakem $<$. Důvodem je snažší oddělování jednotlivých zpráv a jejich zpracování. Přijatá zpráva musí končit znaky ##. Po přijetí bajtu je vyvoláno přerušení, ve kterém je zkontrolováno,jestli je aktuálně přijímána zpráva (byl přijat úvodní bajt). Pokud ano, jsou data uložena do bufferu. Poté je zkontrolováno,

Tabulka 5.4: Přehled zpráv pro MCU ATmega128

Příkaz	**Popis funkce**
<S##	Informace o připojení ke koncovému zařízení
<s##	Informace o odpojení od koncového zařízení
<IXXXX##	Uloží do bufferu lokální IP MCU PIC
<GXXXX##	Uloží do bufferu adresu brány nastavenou v PIC
<MXXXX##	Uloží do bufferu masku podsítě nastavenou v PIC
<Z...##	Uloží text "..."do paměti přijatých textových zpráv
<V##	Požadavek na odeslání teploty a osvětlení okolí terminálu
<Cn$M_1N_1..M_nN_n$##	Nakonfiguruje rozhraní dálkového ovládání KZ
<cn$M_1N_1..M_nN_n$##	Informace o aktuálním stavu n vstupů KZ
<A##	Informace, že odeslaná zpráva byla přijata
<a##	Informace, že odeslaná zpráva nebyla přijata

Tabulka 5.5: Přehled druhů výstupů

Označení M	**Popis funkce**	**Možné stavy N**
1	binární	$1-$ vypnuto; $2-$ zapnuto
2	analogový	rozmezí $<0\%;100\%>$

jestli se v bufferu nachází platná zpráva (začíná a končí správnými znaky). Je-li tomu tak, je dále zpracována.

V následující tabulce 5.4je uveden seznam zpráv, které software v MCU dokáže zpracovat:

Písmena 'X' značí ve zprávě parametr v rozsahu $<0x00;0xFF>$. Ve zprávách od Koncového zařízení značí písmeno 'n' počet výstupů/vstupů, o kterých zpráva informuje. Bajt M_i obsahuje informaci o typu i-tého vstupu/výstupu, bajt N_i obsahuje informaci o jeho aktuálním nastavení zvětšenou o 1. Je tak v případě nulového stavu analogového výstupu zamezeno odeslání znaku bajtu 0, který ukončuje string.

V tabulce 5.6 jsou uvedeny typy výstupů a jejich možné stavy.

Zprávy pro ovládání koncového zařízení jsou shrnuty v tabulce 5.7. Všechny musí začínat znakem $. Je to opět proto, aby se od sebe snáze odlišovaly začátky jednotlivých zpráv.

Tabulka 5.6: Zprávy pro ovládání koncového zařízení

Příkaz	Popis funkce
$c##	Koncové zařízení vrátí aktuální stav vstupů
CnN_n$##	Příkaz k nastavení n-tého výstupu na hodnotu N

Vznikne-li požadavek uživatele na nastavení konkrétního výstupu, je tato informace odeslána přes UART ve formátu $CnX, kde n je pořadové číslo výstupu a X je hodnota výstupu zvýšená o jedna (jedná-li se o analogový výstup), případně X = 1 nebo X = 2, jedná-li se o digitální výstup.

Tvorbě bufferu jsem se nevyhnul ani při řešení odchozí komunikace. Jelikož je nutné odesílat zprávy z několika míst softwaru (odesílání textových zpráv, konfigurační zprávy pro PIC, informace o teplotě apod.), vzniklo nebezpečí, že by se při odesílání mohly zprávy ”křížit”. Dobrým příkladem této situace je okamžik, kdy uživatel odesílá textovou zprávu a ve stejném okamžiku dorazí v přerušení do MCU dotaz na aktuální teplotu. Pak se začne v přerušení odesílat další zpráva, která se odešle během odesílání první. Obě zprávy jsou tak znehodnoceny.

Za tímto účelem jsem byl nucen vytvořit knihovnu ”buffer.c”, jejíž jediným posláním je shromažďovat do pole ukazatele na text, který se má odeslat. Tento zásobník je poté vyprazdňován vždy v přerušení od časovače (2x za sekundu).

Knihovna USART.h je opět psána univerzálně, aby ji bylo možné v případě potřeby použít i na jiném MCU ATmega.

5.3.3 Grafické rozhraní

Před tvorbou samotného grafického rozhraní bylo nutné si ujasnit, jakým způsobem budou v paměti organizovány jednotlivé položky v menu a jakým způsobem budou spouštěny jednotlivé aplikace.

Oba problémy jsem vyřešil najednou tvorbou univerzální struktury pro položku menu. Její definice je uvedena níže:

```
typedef struct TMenuItem
{
```

```
    char* name;
    char type;
    char icon;
    int (*start)();
    struct TMenuItem *subitems;
    char subcount;
}TMenuItem;
```

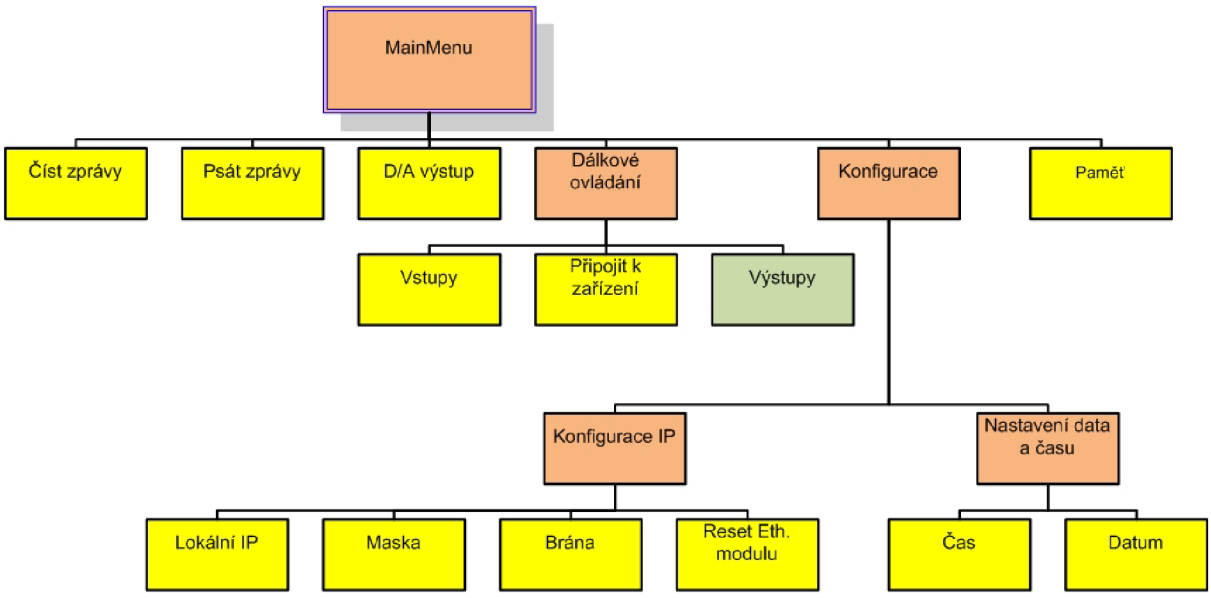
Tato struktura může v proměnné *subitems* obsahovat odkaz na další podmenu, přičemž v proměnné *subcount* je uložen počet podpoložek. Pokud uživatel tuto položku stiskem Enter spustí, zavolá se funkce *void GUI_menu(TMenuItem data)*, kdy vstupním parametrem je zde nadřezená položka požadovaného podmenu. Další možností je uložit do proměnné *(start)()* ukazatel na funkci, která se má spustit při stisknutí klávesy Enter. Rozlišení, zda obsahuje položka další podpoložky je za pomoci proměnné *type*. $type = 0$ značí větvení, $type = 1$ značí spustitelnou položku.

Tímto způsobem jsem vytvořil celý systém menu (inicializovaný *int GUI_MainMenuInit(struct TMenuItem* MainMenu)*). Ten je na obrázku 5.8. Oranžově jsou vyznačeny položky, které se větví, žluté jsou spustitelné. Zeleně vyznačená položka se také větví, její podpoložky jsou ale konfigurovány koncovým zařízením prostřednictvím konfigurační zprávy. Jednotlivé aplikace (včetně aplikace na prohlížení menu) jsou řešeny jednoduchou nekonečnou smyčkou, ve které se vykonávají akce po stisknutí kláves. Jednoduchý stavový diagram takové smyčky je uveden v obrázku 5.9.

Dále jsem nadefinoval jednoduchá informační okna viz například funkce *void GUI_IPopUp(char* $\times$ *text,int length, int del)*, jejímž úkolem je vypsat na displej informační text předaný parametrem *text* a tuto zprávu na displeji ponechat zobrazenou *length* sekund. Poslední parametr udává, zdali se po vypršení času má vymazat displej či nikoli.

Nedílnou součástí uživatelského rozhraní, kterou bylo nezbytné vyřešit, byl způsob vložení informace o stisknuté klávese do programových smyček. Tento problém bylo možné ošetřit dvěma cestami.

První způsob byl následovný: kontrolujeme informační bit výstupu klávesnice ve smyčce a čekáme na jeho nastavení do log. 1. Tento bit je připojen na PINE.5. Protože paralelní datový port klávesnicového modulu je připojen po částech na dva porty MCU,



Obrázek 5.8: Stromová struktura menu

Tabulka 5.7: Závislost konstanty *k* na nastavení *KeyType*

KeyType	Režim	k
0	Velká písmena	56
1	Malá písmena	88
2	Číslice	48

je nezbytné provést několik logických operací pro získání správného kódu klávesy.

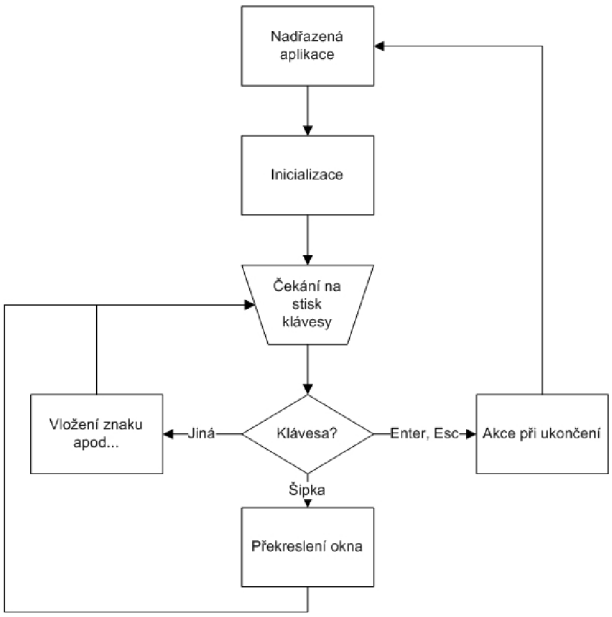
$$KeyPressed = (((PIND >> 4)|((PINB << 4)\&(0b00110000)))); \tag{5.17}$$

Následně je ještě nezbytné přičíst k získanému kódu konstantu *k*, jejíž velikost závisí na aktuálně nastavené proměnné *KeyType*. Je tak rozlišeno, je-li klávesnice v režimu malých nebo velkých písmen, případně číslic. Vše je shrnuto v tabulce 5.7.

Hodnota proměnné KeyPressed bude poté:

$$KeyPressed = KeyPressed + k; \tag{5.18}$$

Dále musíme zkontrolovat, nejedná-li se o stisknutou funkční klávesu. Ta má totiž specifický kód. Pokud je tomu tak, je proměnná *KeyPressed* opět modifikována, aby odpovídala konkrétním stisknutým funkčním klávesám s kódy dle tabulky 5.8.



Obrázek 5.9: Stavový diagram aplikace

Tabulka 5.8: Tabulka funkčních kláves

Původní KeyPressed	Význam	KeyPressed po korekci
91	Backspace	25
93	Šipka vlevo	22
94	Šipka vpravo	23
95	Enter	24
96	Escape	27

Druhý způsob načítání kláves je dle mého názoru mnohem výhodnější, využíváme totiž pro jeho načtení funkci přerušení. Není tak nutné blokovat běh programu a čekat na stisk klávesy aktivně, program může běžet cyklicky ve smyčce a dle potřeby kontrolovat, jaký je aktuální kód klávesy. Kromě toho, že je vyvoláno přerušení a klávesa je načtena v přerušení, je ale postup zpracování informace o klávese stejný jako v předchozím případě.

5.3.4 Obsluha lokálních senzorů

Popis funkce senzorů z elektrického hlediska je popsán v předchozích kapitolách. Nyní se budu zabývat popisem z hlediska softwaru. Senzor teploty je připojen ke kanálu 2, senzor osvětlení ke kanálu 3. Postup zjištění měřené hodnoty je následující:

Tabulka 5.9: Závislost odporu termistoru na teplotě

$t[°C]$	-20	-10	0	10	20	25	30	40	50
$R[\Omega]$	684	747	815	886	961	1000	1040	1122	1209

1. Zjištění hodnoty změřené A/D převodníkem
2. Převod hodnoty na napětí
3. Převod napětí na měřenou veličinu

Fyzikální princip měření teploty je popsán v kapitole věnující se návrhu hardware.

Měřená veličina je v případě senzoru teploty ve $[t] = °C$, v případě senzoru osvětlení v $[l] = \%$. Měření je spouštěno vždy 2x za sekundu, interval je řízen časovačem Timer0. Referenční napětí A/D převodníku $U_{ref} = 4,92V$, což je napájecí napětí MCU.

Zjištění napětí na vstupu A/D převodníku pokud měřená hodnota je x, lze provést za pomoci trojčlenky:

$$U = \frac{U_{ref}x}{1024} \tag{5.19}$$

Opět se zde setkáme s problémem celočíselné aritmetiky, je tedy výhodnější počítat napětí U v $[U] = mV$.

Nyní se budu zabývat výpočtem teploty z naměřeného napětí. Známe-li napětí 5.19 a proud (určený rovnicí 3.11), lze vypočíst aktuální odpor R_t termistoru podle

$$R_t = \frac{U}{I_t} \tag{5.20}$$

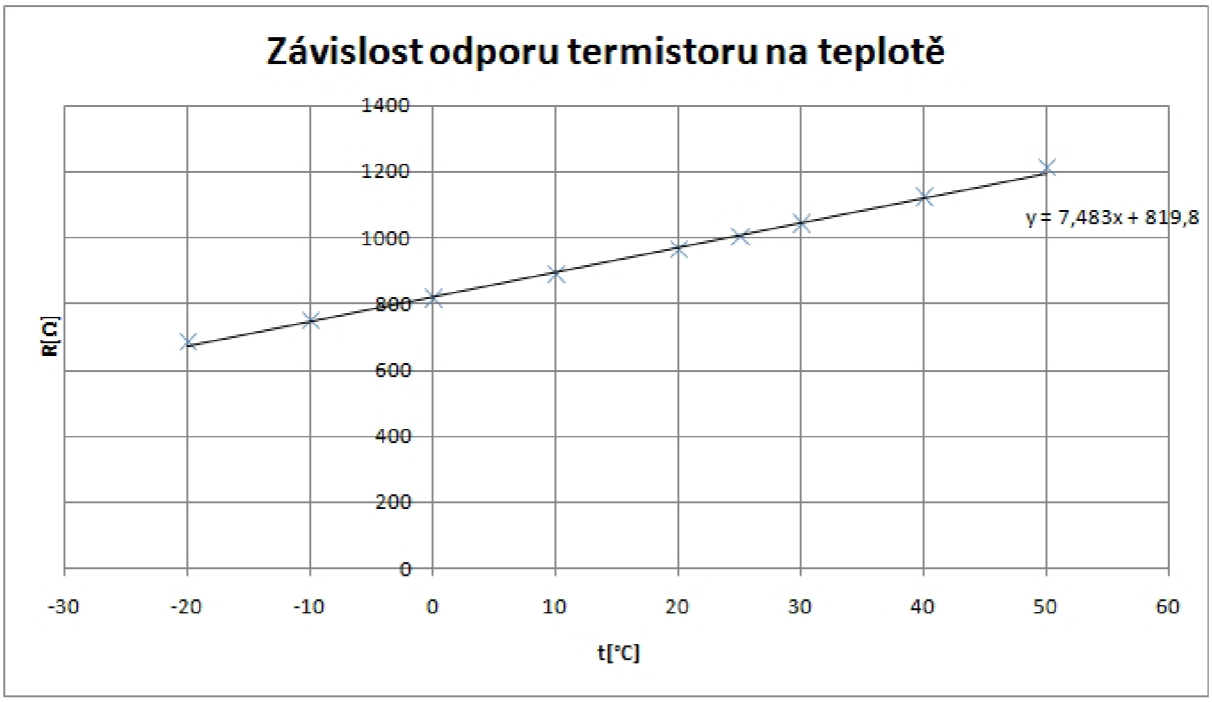
Jelikož není charakteristika závislosti odporu na teplotě lineární, byl jsem nucen vytvořit vhodnou aproximaci. K tomu jsem použil hodnoty závislosti z datasheetu termistoru. Zajímalo mne zejména teplotní rozmezí $t \in < -20°C; 50°C >$. Tyto body jsem tedy zapsal do tabulky 5.9 a v programu Microsoft Excel je vykreslil do grafu. Poté jsem hledal vhodnou aproximační funkci.

Vykreslený graf je uveden na obrázku 5.10. Vidíme v něm i rovnici proložené přímky, která byla vypočtena metodou nejmenších čtverců.

$$y = 7,483x - 819,8 \tag{5.21}$$

Abychom mohli vypočíst z odporu teplotu, musíme vyjádřit z 5.21 neznámou x. Tedy:

$$x = 0,133y - 109 \tag{5.22}$$



Obrázek 5.10: Grafická závislost odporu termistoru na teplotě s vykreslenou aproximační funkcí

Nyní již lze ze získaných údajů z rovnic 5.22 a 5.20 vypočíst teplotu.

$$t = 0,133R_t - 109 = 0,133\frac{U}{I_t} - 109 \tag{5.23}$$

Problémem je opět celočíselná aritmetika, je vhodné počítat v mV a mA stejně jako při výpočtech koeficientu k přímky za pomoci rovnic 5.15 a 5.16. Dále je vhodné poznamenat, že z fyzikálního hlediska by bylo vhodnější prokládat závislost odporu na teplotě exponenciálnou, ovšem při výpočtech v MCU je mnohem efektivnější přímka. Z grafu je evidentní, že chyba výpočtu je stále zanedbatelná.

Výpočet aktuální hodnoty osvětlení probíhá podobně, je ale jednodušší, neboť není potřeba aproximovat žádnou závislost. Z uvedeného také vyplývá, že výsledek je pouze orientační, slouží k získání informace, jestli je kolem terminálu světlo nebo tma (případně šero).

Nejprve vypočteme aktuální hodnotu napětí na fotorezistoru podle 5.19. Experimentálně jsem zjistil, že při maximálním odporu je na fotorezistoru maximální napětí $U_{max} = 2,8V$. hodnota l tedy bude:

$$l = 100\frac{U}{U_{max}} \tag{5.24}$$

5.3.5 Další funkce terminálu

Software terminálu obsahuje několik funkcionalit, které je vhodné krátce shrnout, ale nevyžadují dle mého názoru podrobnější popis, protože vše je podrobně okomentováno v přiloženém zdrojovém kódu.

První funkcionalitou je funkce hodin. Jelikož je k ATmega128 připojen hodinový krystal s pracovní frekvencí 32 768Hz, bylo vhodné využít jej k měření času. Díky nízké frekvenci (při nastavení vnitřní předděličky na hodotu 1024 a OCR0 = 16) je totiž možné zaručit, že bude přerušení od časovače vyvoláno 2x za vteřinu, nebude tedy hlavní program příliš zpomalován a zaručí nám to přesné časování některých aktivit jako zhasínání displeje po 5 vteřinách, popřípadě odeslání zpráv z bufferu. Další časovanou událostí je zjišťování teploty a osvětlení.

Druhou funkcionalitou je digitálně/analogový převodník. Mým cílem bylo užít režim převodníku takový, aby nebylo třeba vyvolat po každém cyklu přerušení. Rozhodl jsem se užít tzv. Rychlý PWM režim s předděličkou nastavenou na hodnotu 1.

5.3.6 Členění kódu

Protože firmware ATmega128 je poměrně rozsáhlý, byl jsem nucen jej logicky rozdělit do několika knihoven, kdy každá knihovna sdružuje funkce podobného účelu. Knihovny a jejich funkce jsou přehledně shrnuty v tabulce5.10

Tabulka 5.10: Tabulka knihoven

	Hlavičkový soubor	Obsah
1	LCD_lib.h	Vstupní bod firmware s funkcí main
2	GLCD_lib.h	Knihovna pro ovládání displeje
3	GUI.h	Knihovna obsahující funkce grafického rozhraní a menu
4	USART.h	Knihovna pro práci s jednotkou UART
5	Timer0.h	Knihovna pro práci s časovačem 0
6	Buffer.h	Textový buffer
7	TCP.h	Knihovna s funkcemi pro ovládání MCU PIC
8	extra.h	Knihovna obsahující zejména funkce pro práci s řetězci
9	AD.h	Knihovna pro práci s A/D převodníkem
10	PWM.h	Knihovna pro práci s analogovým výstupem

6 SIMULÁTOR KONCOVÉHO ZAŘÍZENÍ

Jelikož zadáním mé bakalářské práce je vytvoření terminálové jednotky, rozhodl jsem se hlouběji nezabývat návrhem hardware koncového zařízení. Ten totiž závisí na konkrétním druhu koncového zařízení a terminál je navržen univerzálně, aby bylo možné ovládat jakékoli koncové zařízení podporující příslušný komunikační protokol.

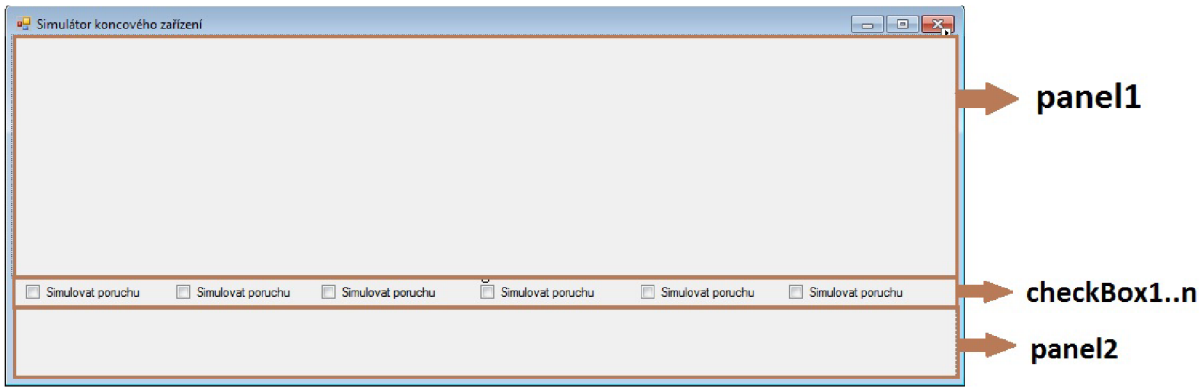
Abych mohl terminál testovat, bylo nutné vytvořit simulátor koncového zařízení. K tomuto účelu posloužil program, který jsem tvořil v jazyce C#, v grafickém prostředí Windows Forms na bázi .NET Framework 3.5. Výhodou tohoto prostředí je velké množství hotových komponent, které stačí stylem drag&drop vložit do formuláře a pouze ”pospojovat”.

Celá aplikace je realizována pomocí dvou podčástí, první z nich je grafické rozhraní, druhá z nich je pracovní jádro programu.

6.1 Grafické rozhraní

Grafické rozhraní programu je uvedeno na obrázku 6.1. Obsahuje tyto komponenty:

- **form1:** Kontejner na ostatní komponenty.
- **panel1:** Vykreslovací plocha, na níž jsou zobrazeny zásuvky a ventilátory. Kvůli blikání při překreslování bylo nezbytné vytvořit zděděnou třídu třídy *Panel* s názvem *DoubleBufferPanel*, která má přepsanou metodu *protected override void OnPaintBackground(PaintEventArgs e)*, což zajistí, že se nebude při každém překreslování znovu vykreslovat pozadí.
- **panel2:** Komponenta slouží k vypisování stavu připojení a k vypisování přijatých zpráv. Je stejného typu jako *panel1*.
- **checkBox1...6:** Tyto komponenty slouží uživateli k simulaci poruchy zařízení. Informace o poruše je ihned zaslána aktuálně připojené terminálové jednotce.
- **timer1:**Tato komponenta není ve formuláři viditelná, stará se o časování cyklicky volaných událostí jako odpovídání na přijaté zprávy, změnu polohy větráků a podobně.



Obrázek 6.1: Rozložení komponent v aplikaci Simulátor koncového zařízení

Vykreslování komponent je realizováno metodami *private void panel1_Paint(object sender, PaintEventArgs e)* a *private void panel2_Paint(object sender, PaintEventArgs e)*, kdy první se stará o vykreslování panelu se zásuvkami a větráky, druhá se stará o vypisování informačních a přijatých zpráv.

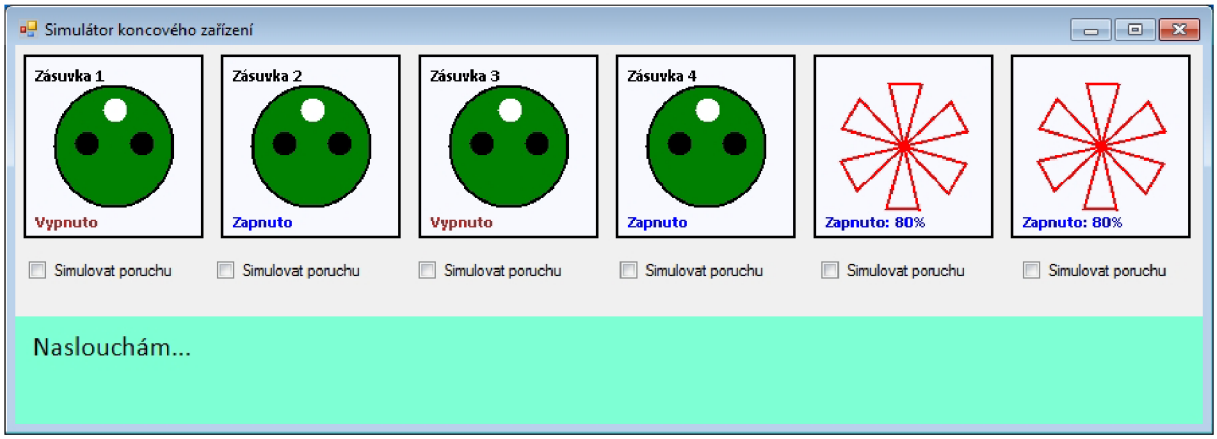
6.2 Pracovní jádro programu

Pracovním jádrem se rozumí podproces běžící v odděleném vlákně tvořený metodou *void WaitForData()*, který se stará o připojené klienty a přijímá od nich data . Tato data jsou buď ihned zpracována nebo jsou uložena do bufferu *string data*, odkud jsou později vyzvednuta při události Tiknutí časovače timer1. Zde je poté zpracován zbytek dat.

Další částí jádra programu jsou třídy *Zasuvka.cs* a *Vetrak.cs*. Tyto třídy v jednotlivých instancích reprezentují jednotlivé zobrazené zásuvky a větráky. Třída *Vetrak.cs* v sobě navíc obsahuje metodu pro změnu aktuálního natočení větráku v závislosti na aktuální pozici a rychlosti.

6.3 Komunikace programu s okolím

Téměř veškerá komunikace programu s okolím probíhá přes protokol TCP. V praxi to znamená, že po spuštění programu již uživatel nemůže program nijak konfigurovat. Jedinou výjimku tvoří políčka checkBox, kterými lze simulovat poruchu vybraného spotřebiče. Screen shot programu je na obrázku 6.2.



Obrázek 6.2: Program Simulátor koncového zařízení za běhu

7 ZÁVĚR

V předložené bakalářské práci je uveden návrh koncepce sítě terminálů pro rodinný dům či jiný objekt neprůmyslové koncepce. Terminály jsou schopny komunikovat s koncovými zařízeními v domě a umožňují tak ovládání spotřebičů různého druhu přes síť Ethernet. Umožňují také textovou komunikaci v objektu. Protože součástí zadání bakalářské práce není výroba koncového zařízení, realizoval jsem Simulátor Koncového zařízení v PC, který umožňuje přístroj náležitě otestovat.

Hlavním procesorem výrobku je mikrokontrolér ATmega128, který se stará o komunikaci terminálu s uživatelem. Jako komunikační rozhraní byl zvolen grafický LCD displej doplněný 16-ti tlačítkovou klávesnicí, která je schopna pracovat v režimu znakovém i numerickém, podobně jako klávesnice mobilního telefonu. Klávesnice také slouží k pohybu uživatele v menu. Výrobek obsahuje signalizační diody pro indikaci zapnutého napájení, indikaci nové zprávy a indikaci připojení k serveru. S pomocí menu lze konfigurovat síťové adresy (lokální, adresu brány apod), datum a čas. Adresy zůstávají uloženy i po odpojení výrobku od napájení v paměti EEPROM a po restartu jsou obnoveny. Výrobek je vybaven také analogovým napěťovým výstupem, který je možné ovládat v rozmezí od 2,3V do 13V.

Spojení přes Ethernet zajišťuje MCU PIC18F67J60. Základem jeho softwarového vybavení je TCP/IP Stack v5.1 modifikovaný pro potřeby této bakalářské práce. Komunikace mezi procesory je realizována přes rozhraní UART za pomoci příkazů definovaných v předchozích kapitolách. Pevně definované jsou i zprávy pro ovládání koncových zařízení a zasílání textových zpráv. Výrobek naslouchá na portu 65 a všechna spojení realizuje také na tomto portu, nemělo by tedy docházet ke konfliktu se žádnou běžnou aplikací. Maximální rychlost připojení je 10MBit/s.

Dvouprocesorové řešení jsem zvolil, protože TCP/IP Stack je relativně náročný na výkon, navíc zabírá poměrně velké množství RAM (Asi 2,2kB), nezbyl by tak dostatečný pracovní prostor pro grafické rozhraní a přijímané textové zprávy. Zařízení je vybaveno funkcí pro zjištění volné operační paměti.

Fotografie hotového výrobku jsou umístěny v příloze a kompletně okomentovaný software je přiložen k této práci.

REFERENCE

[1] MATOUŠEK, David. *Práce s mikrokontroléry Atmel AT89C2051*. 2. vyd. Praha : BEN - technická literatura, 2006. 248 s., 1 CD-ROM. ISBN 80-7300-094-6.

[2] PATOČKA, Miroslav, VOREL, Pavel. *Řídící elektronika - Aktivní obvody*. Skriptum FEKT VUT Brno, 2004. 154 s.

[3] ZEZULKA, F. *Prostředky průmyslové automatizace*. Brno: VUTIUM, 2004. 176 s. ISBN: 80-214-2610-1.

[4] KALÁB, P.; STEINBAUER, M.; VESELÝ, M. *Bezpečnost v elektrotechnice*. Brno: Ing. Zdeněk Novotný, CSc, Ondráčkova 105, 628 00 Brno, 2007. 78 s. ISBN: 978-80-214-3509-4.

[5] *ATmega 128, 8-bit Microcontroller with 128K Bytes In-System Programmable Flash*. Rev. 2467S–AVR–07. [b.m.]: Atmel, 2009 [cit. 2009-11-14]. 386 s. [online]: <www.atmel.com/atmel/acrobat/doc2467.pdf>.

[6] *AT89C2051, 8-bit Microcontroller with 2K Bytes Flash*. Rev. 0368H–MICRO–6. [b.m.]: Atmel, 2008 [cit. 2009-11-14]. 19 s. [online]: <www.atmel.com/atmel/acrobat/doc0368.pdf>.

[7] *L7800 series*. Rev. 12. [b.m.]: STMICROELECTRONICS, November 2004 [cit. 2009-11-14]. 34 s. [online]: <http://www.gme.cz/_dokumentace/dokumenty/330/330-149/dsh.330-149.1.pdf>.

[8] *ATM12864D: Liquid Crystal Display Module*. [b.m.]: [b.j.]. 21 s. [online]: <http://www.hebeiltd.com.cn/lcm.datasheet/ATM12864D.pdf>. [cit. 2009-11-14].

[9] *BC337; BC338: Small signal transistors (NPN)*. [b.m.]: GENERAL SEMICONDUCTOR, 4/98 [cit. 2009-11-14]. 5 s. [online]: <http://www.gme.cz/_dokumentace/dokumenty/210/210-019/dsh.210-019.1.pdf>.

[10] *BC327, BC328*. [b.m.]: Motorola, 1996 [cit. 2009-11-14]. 5 s. [online]: <http://www.datasheetcatalog.org/datasheet/motorola/BC328.pdf>.

[11] *PIC18F97J60 Family Data Sheet*. Rev. 7; 02/08. [b.m.]: Microchip, 2008 [cit. 2010-03-14]. 480 s. [online]: <http://ww1.microchip.com/downloads/en/DeviceDoc/39762d.pdf>.

[12] *LF153 - LF253 - LF353: WIDE BANDWIDTH DUAL J-FET OPERATIONAL AMPLIFIERS*. Rev. December 2003. [b.m.]: National Semiconductor Corporation, 2003 [cit. 2009-12-11]. 14 s. [online]: <www.national.com/ds/LF/LF353.pdf>.

[13] *MCP100101*. Rev. DS11187D. USA: Microchip Technology Inc, 1999 [cit. 2009-12-11]. 8 s. [online]: <http://www.datasheetcatalog.com/datasheets_pdf/M/C/P/1/MCP101.shtml>.

[14] *IRF9530, RF1S9530SM*. Rev. July 1999. [b.m.]: Intersil Corporation, 1999 [cit. 2009-12-11]. 8 s. [online]: <http://www.datasheetcatalog.org/datasheets/70/352213_DS.pdf>.

[15] *LM117/LM317A/LM317 - 3-Terminal Adjustable Regulator*. Rev. May 1996. USA: National Semiconductor, 1996 [cit. 2009-12-11]. 17 s. [online]: <http://www.datasheetcatalog.org/datasheet/nationalsemiconductor/DS009063.PDF>.

[16] *KTY81-1 series - Silicon temperature sensors*. Rev. 2000 Aug 25. Holandsko: Philips Electronics, 2000 [cit. 2009-12-11]. 17 s. [online]: <http://www.datasheetcatalog.org/datasheet/philips/KTY81-1SERIES_3.pdf>.

[17] *Advanced sensor technologies for today's breakthrough applications*. Rev. 2002. [b.m.]: PerkinElmer Optoelectronics, 2002 [cit. 2009-12-11]. 50 s. [online]: <http://www.digchip.com/datasheets/parts/datasheet/633/VT83N1-pdf.php>.

[18] *74HCHCT138 3-to-8 line decoderdemultiplexer; inverting*. Rev. September 1993. [b.m.]: Philips Semiconductors, 1993 [cit. 2009-12-11]. 9 s. [online]: <http://www.datasheetcatalog.org/datasheet2/1/02lg3k6oaoxssd44fhxlu3xzricy.pdf>.

[19] *GM Electronic* [online]. 2010 [cit. 2010-05-24]. F-KV16KEY BLACK. Dostupné z WWW: <http://www.gme.cz/cz/f-kv16key-black-p637-091.html>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

LED Light Emiting Diode

LCD Liquid crystal display

MCU Mikrokontrolér

CPU Central Processor Unit

RAM Random Access Memory

EEPROM Electricaly Erasable Programable Read Only Memory

FLASH Programová paměť

USART Universal Synchronous and Asynchronous serial Receiver and Transmitter

CAN Controller area network

RS – 485 Modifikace sériového rozhraní

TWI Two Wired Interface

SPI Serial Peripheral Interface

TCP/IP Rodina komunikačních protokolů

MAC Vrstva ISO/OSI modelu

PHY Fyzická vrstva ISO/OSI modelu

t Teplota

R Elektrický odpor

C Kapacita

L Indukčnost

f Frekvence

U Napětí

I Proud

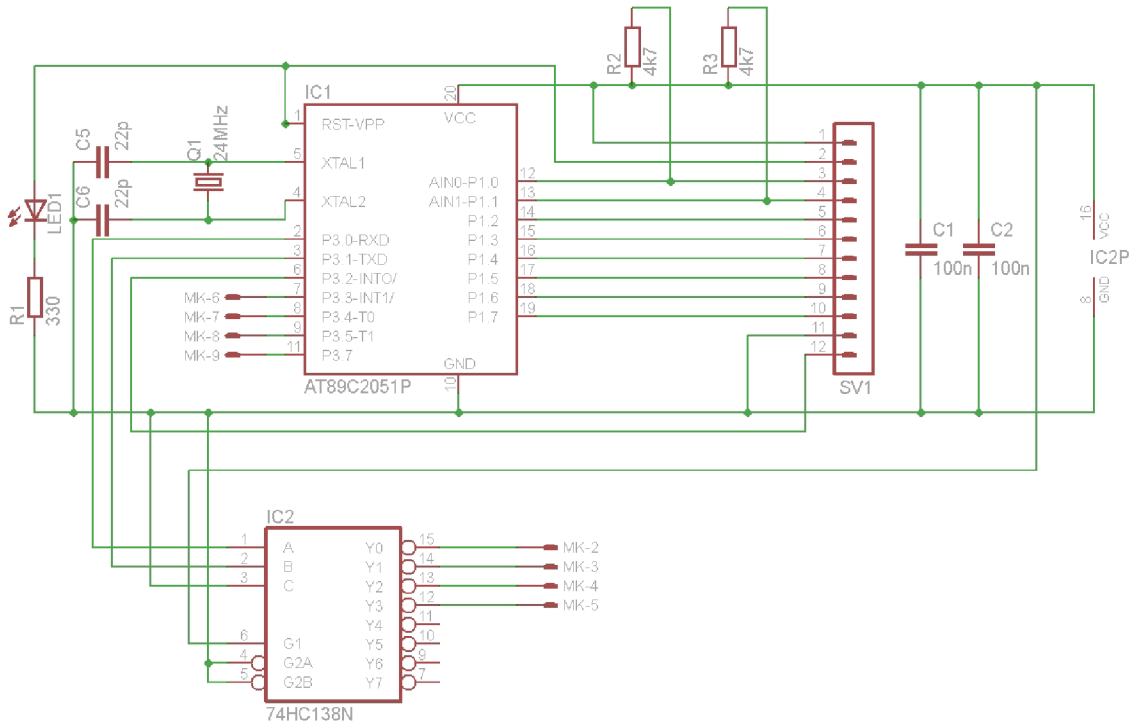
SEZNAM PŘÍLOH

A Kompletní schéma terminálu **63**

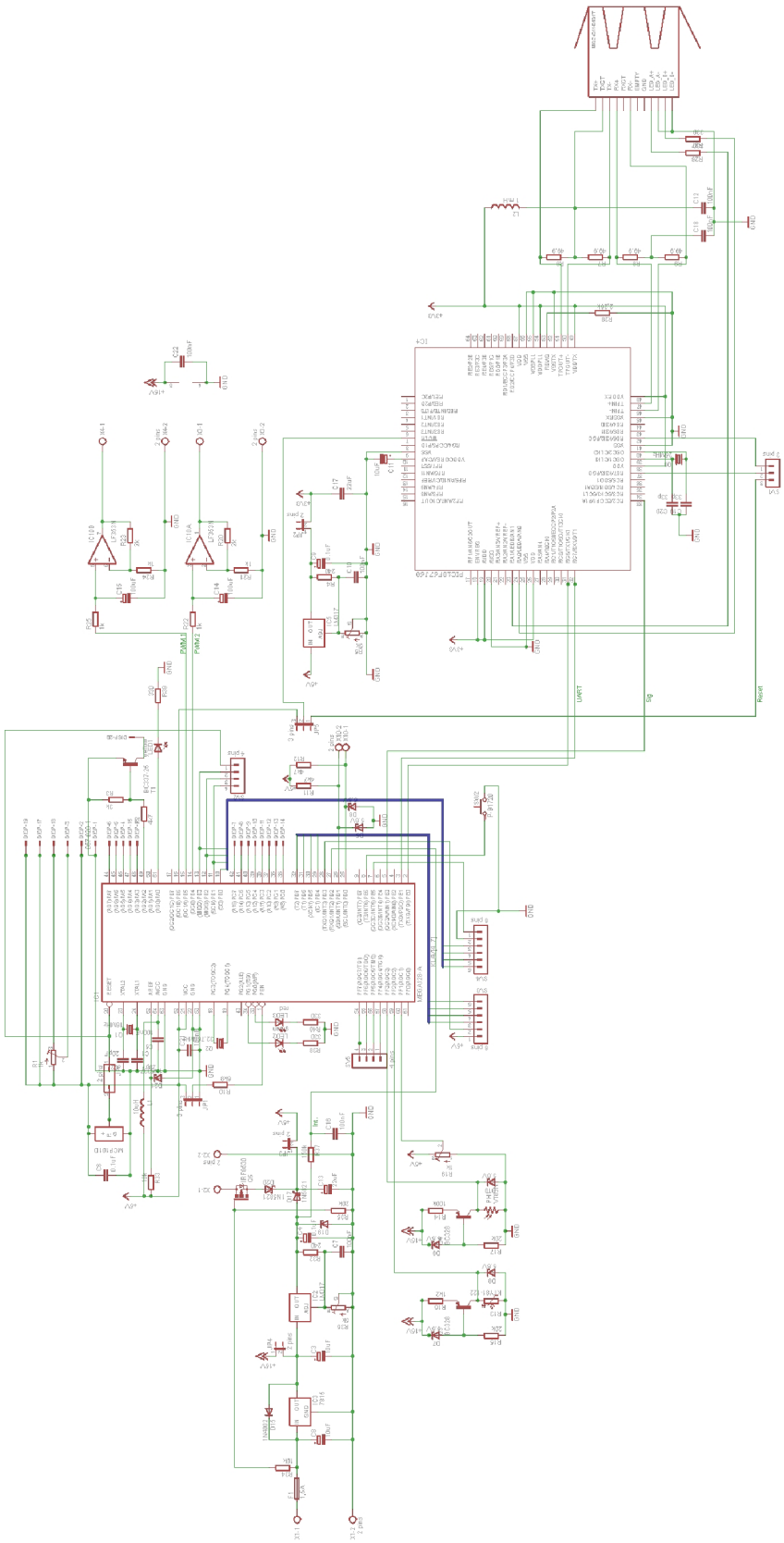
B Výrobní dokumentace **65**

C Fotografie výrobku **72**

A KOMPLETNÍ SCHÉMA TERMINÁLU



Obrázek A.1: Obvodové schéma klávesnicového modulu



Obrázek A.2: Kompletní schéma terminálu

B VÝROBNÍ DOKUMENTACE

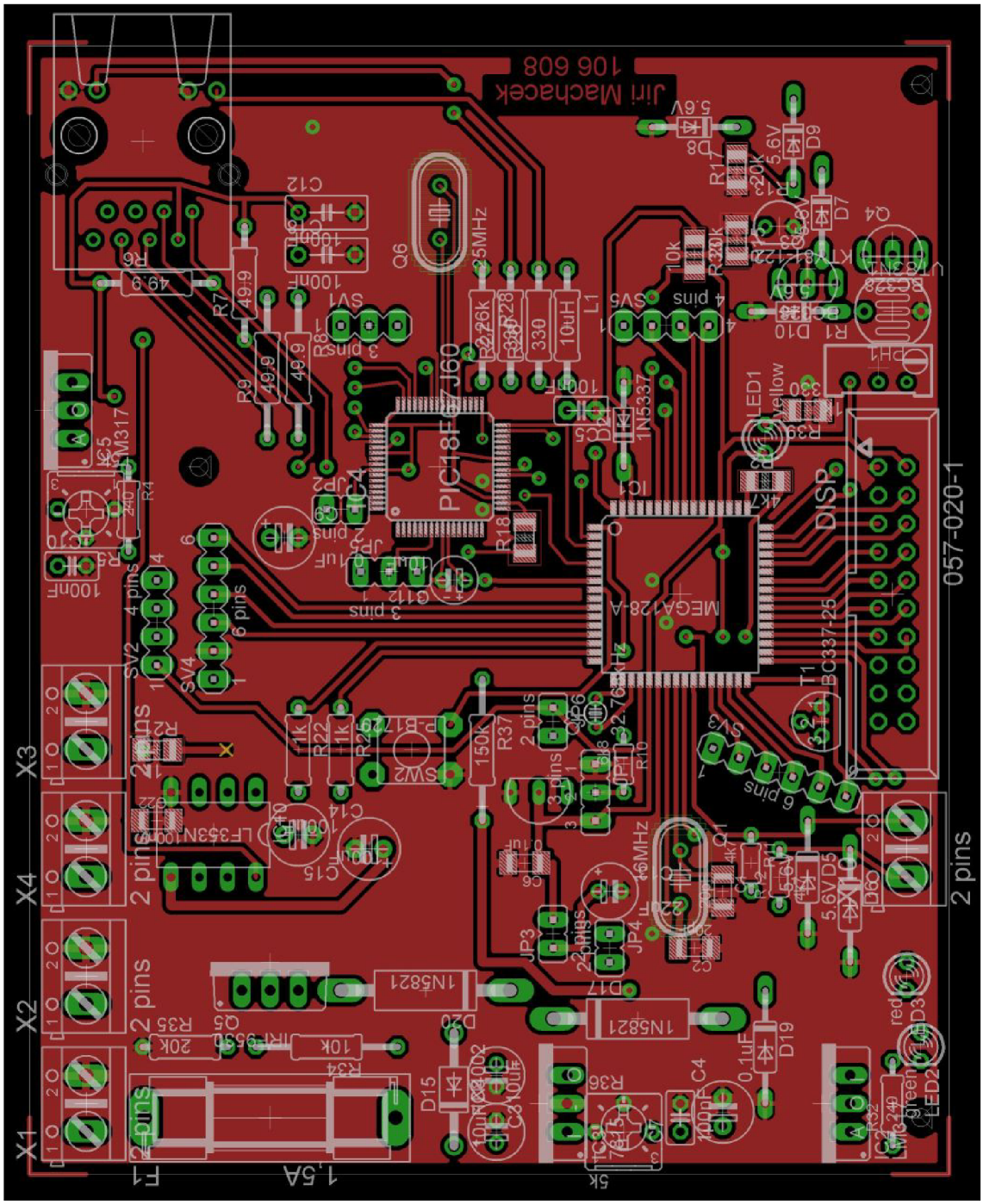
Soupis součástek Terminálové jednotky

Hodnota	Pozn.	Kusy	Název v katalogu	Dodavatel
Kondenzátory keramické				
100nF		7	CK 100N X7R	GM Electronic
22pF	1206SMD	2	CK1206 22P/50V NPO	GM Electronic
33pF	1206SMD	2	CK1206 33P/50V NPO	GM Electronic
100nF	1206SMD	4	CK1206 100N/50V X7R	GM Electronic
Kondenzátory elektrolytické				
22uF/25V		1	E22M/25VTS	GM Electronic
10uF/100V		3	E10M/100V	GM Electronic
100n/50V		2	E0.1M/50V	GM Electronic
100uF/16V		2	E100M/16V	GM Electronic
Cívka				
1mH		1	09P-102J MAT	GM Electronic
10uH		1	TL.10µH-M	GM Electronic
Rezistory				
5k	trimr	2	64W5k	GM Electronic
1k	potenciometr	2	PC1221NK001	GM Electronic
4k7		3	MPR 4K7	GM Electronic
6k8		1	MPR 6K8	GM Electronic
240		2	MPR 240R	GM Electronic
1k		2	MPR 1K	GM Electronic
330		2	MPR 330R	GM Electronic
10k		1	MPR 10K	GM Electronic
20k		1	MPR 20K	GM Electronic
150k		1	MPR 150K	GM Electronic
2k26		1	MPR 2K2	GM Electronic
330	1206SMD	3	R1206 330R 1%	GM Electronic
3k	1206SMD	1	R1206 3k 1%	GM Electronic
100K	1206SMD	1	R1206 100K 1%	GM Electronic
20k	1206SMD	2	R1206 20K 1%	GM Electronic
1k2	1206SMD	1	R1206 1K2 1%	GM Electronic
2k	1206SMD	1	R1206 2K 1%	GM Electronic
1k	1206SMD	2	R1206 1K 1%	GM Electronic
10k	1206SMD	1	R1206 10K 1%	GM Electronic
4k7	1206SMD	1	R1206 4K7 1%	GM Electronic
KTY81-122	termistor	1	KTY81-122	GM Electronic
VT83N1	fotorezistor	1	VT83N1	GM Electronic
Lámací lišty				
Piny		33	ASS11520G	GM Electronic
Jumperové propojky		6	JUMP-SW MINI	GM Electronic
Svorkovnice				
AK500/2		5	AK500/2	GM Electronic
MLW20		1	MLW20	GM Electronic
Pojistky				
Držák		1	KS20-01	GM Electronic
Trubičková pojistka 1A		1	FSBF01	GM Electronic

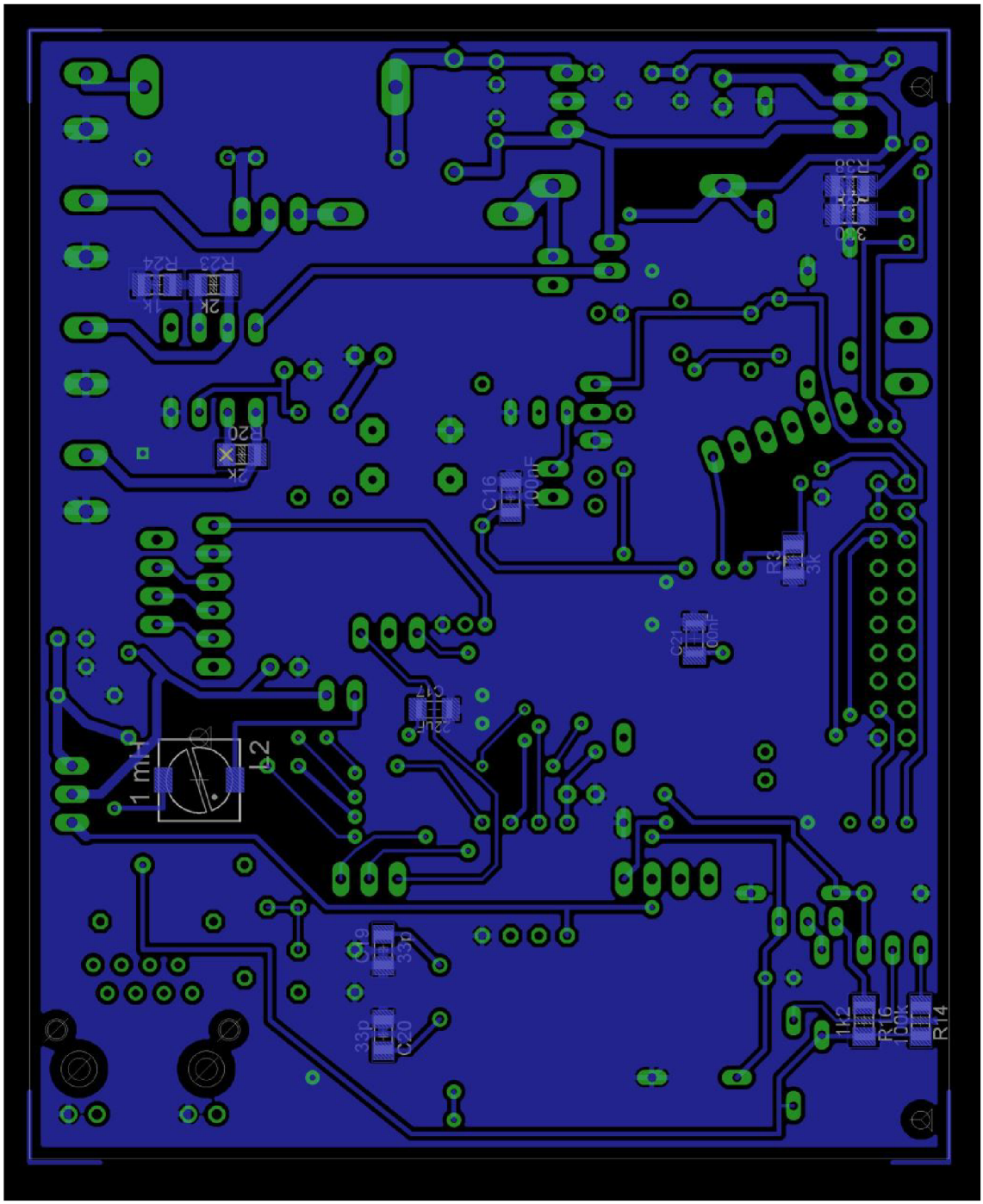
Tranzistory				
BC337-25		2	BC337-25	GM Electronic
BC328-16		3	BC328-16	GM Electronic
IRF9530		1	IRF9530N	GM Electronic
Krystaly				
25MHz		1	Q 25MHZ	GM Electronic
32.768kHz		1	Q 32.768KHZ	GM Electronic
16Mhz		1	Q 16MHZ	GM Electronic
Stabilizátory				
LM317	1,5A	2	LM317T	GM Electronic
7815	1,5A	1	7815	GM Electronic
Operační zesilovače				
LF353N		1	LF353N	GM Electronic
Diody				
3mm LED		4	HLMP-K150	GM Electronic
BZX55CV6		12	BZX55CV6	GM Electronic
1N4002		1	1N4002	GM Electronic
1N5821		2	1N5821	GM Electronic
1N5337		1	1N5337	GM Electronic
Mikrokontroléry				
ATmega128 - 16Pu		1	ATmega128-16PU	GM Electronic
PIC18F67J60		1	PIC18F67J60 - I/P	TME Electronic
Ostatní				
MIC24011-0101T	Zásuvka RJ45	1	MIC24011-0101T	TME Electronic
ATM12864	Displej	1	ATM12864D-FL-YBW	GM Electronic

Rozpis součástek klávesnicového modulu

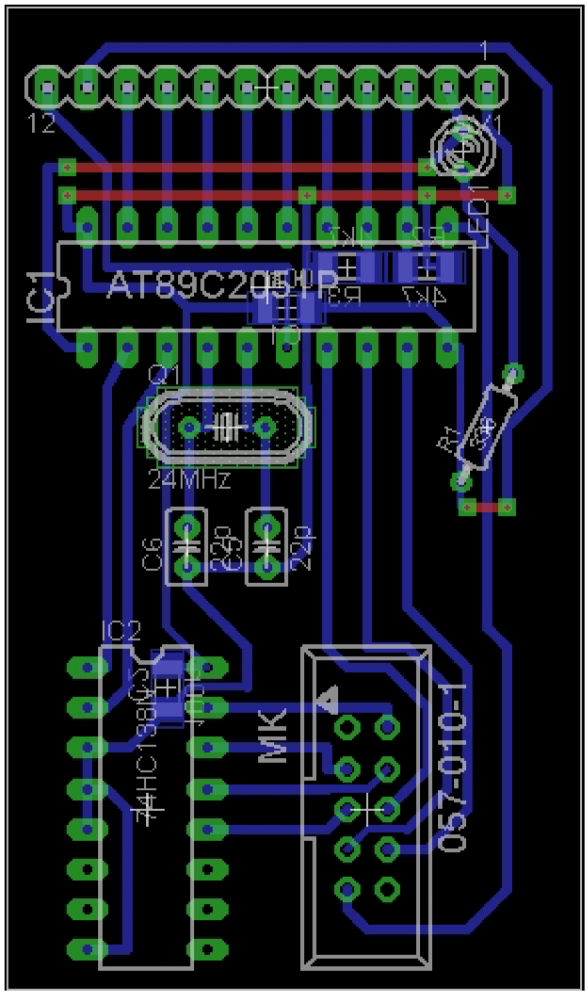
Hodnota	Pozn.	Kusy	Název v katalogu	Dodavatel
Integrované obvody				
AT89C2051		1	89C2051-24PU	GM Electronic
74HC138N		1	74HC138N	GM Electronic
Rezistory				
4k7	SMD1206	2	R1206 4K7 1%	GM Electronic
330		1	MPR 330R	GM Electronic
Kondenzátory keramické				
100nF		2	CK 100N X7R	GM Electronic
22pF	SMD1206	2	CK1206 22P/50V NPO	GM Electronic
Krystaly				
24MHz		1	Q 24MHZ	GM Electronic
Diody				
LED 3mm		1	HLMP-K150	GM Electronic
Konektory				
MLW10		1	MLW10	GM Electronic
Lámací lišta - piny		12	ASS11520G	GM Electronic
Ostatní				
F-KV16KEY BLACK	klávesnice	1	F-KV16KEY BLACK	GM Electronic



Obrázek B.1: Horní strana desky terminálu.



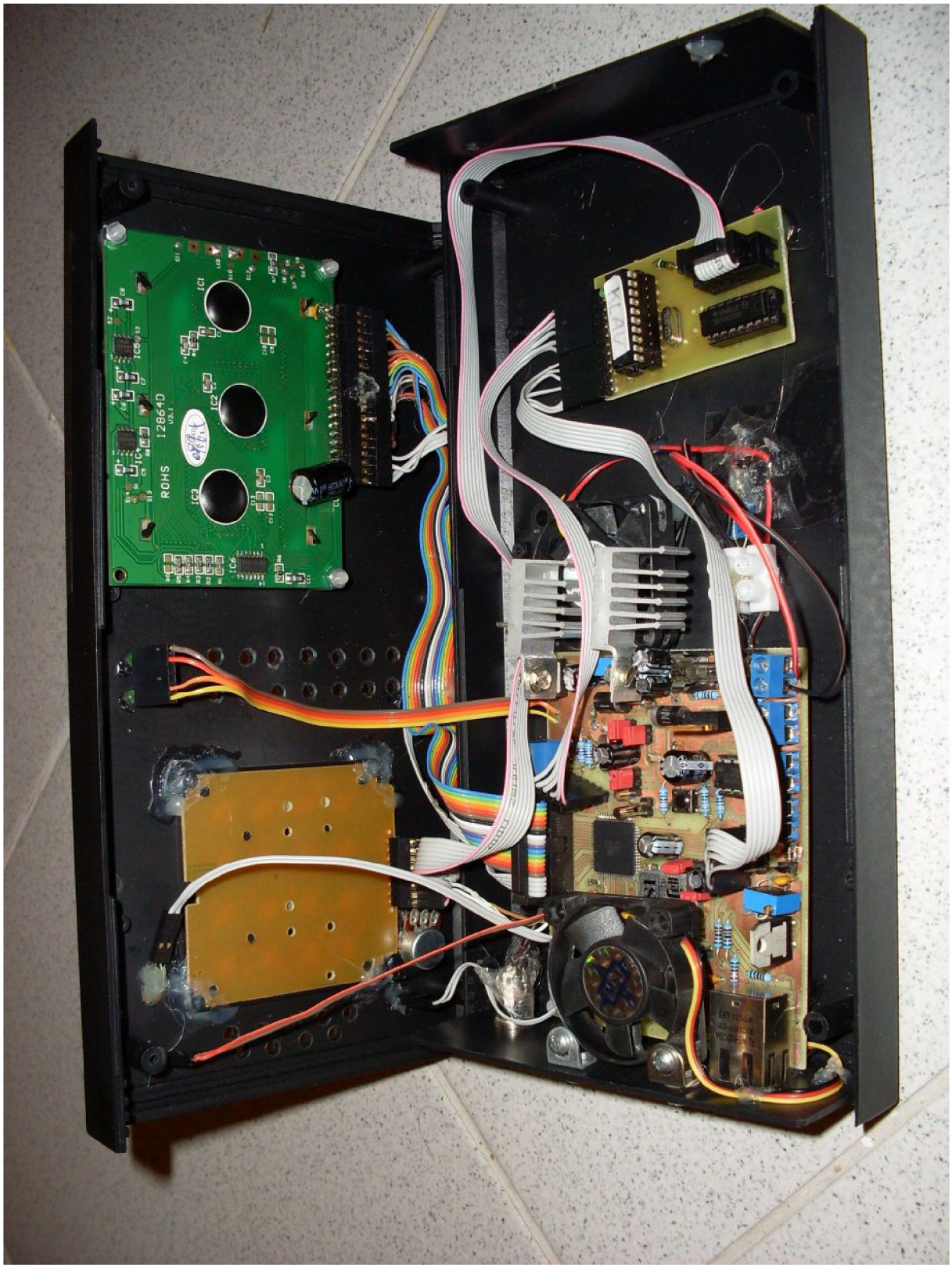
Obrázek B.2: Dolní strana desky terminálu.



Obrázek B.3: Deska klávesnicového modulu

C FOTOGRAFIE VÝROBKU

Obrázek C.1: Fotografie hotového výrobku



Obrázek C.2: Fotografie vnitřního uspořádání