TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

BAKALÁŘSKÁ PRÁCE

Liberec 2011 Roman Halow

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Roman HALOW**
Osobní číslo:	**M07000014**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Demonstrační příklady a možnosti rozšíření přípravku pro výuku předmětu PMP**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s existujícím výukovým přípravkem pro předmět PMP.
2. Vytvořte sadu demonstračních aplikací pro ukázku všech jeho možností.
3. Seznamte se s možnostmi rozšíření tohoto přípravku a navrhněte několik variant jak rozšířit přípravek o další periferie.
4. Realizujte jednu z navržených variant.

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika

Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Demonstrační příklady a možnosti rozšíření přípravku pro výuku předmětu PMP

Demonstration examples and possible extensions of education module for subject PMP

Autor:	**Roman Halow**
Vedoucí práce:	Ing. Tomáš Martinec, Ph.D.
Konzultant:	Ing. Josef Grosman

V Liberci 20. 5. 2011

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **souhlasím** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

V Liberci dne 20. 5. 2011 .....

Poděkování

Děkuji především vedoucímu bakalářské práce Ing. Tomášovi Martincovi, Ph.D za odbornou pomoc, obětavé vedení a další cenné rady. Dále bych chtěl poděkovat také mé rodině a přátelům za jejich trpělivost a neustálou podporu.

Abstrakt

Prezentovaná bakalářská práce se zabývá programováním aplikací a návrhem hardwarových rozšíření pro výukový přípravek Technické univerzity v Liberci. Teoretická část práce řeší problematiku programování mikroprocesorů a možnosti výukového přípravku. Praktická část se zabývá konkrétními aplikacemi pro výukový přípravek a realizací hardwarového rozšíření, které ukazuje jednu z možností, jak lze přípravek rozšířit.

Vytvořené demonstrační aplikace realizují digitální hodiny s využitím RTC obvodu, tester všech součástí přípravku, hru hádání čísel a aplikaci pro řízení jasu žárovky. Realizované hardwarové rozšíření je samostatný obvod, který umožňuje ovládání krokových motorů.

Klíčová slova: mikroprocesory, výukový přípravek TUL, krokové motory, RTC obvod, programovací jazyk C

Abstract

The presented Bachelor thesis deals with programming of applications and designing hardware extensions for educational module of the Technical University in Liberec. Theoretical part is concerned with problematic of programming of microprocessors and capabilities of the educational module. Practical part contains applications for the educational module and an implementation of a hardware extension, that’s presents one of many possibilities of extending the module.

The created demonstrational applications implement digital clock with the use of RTC circuit, a tester of all peripheries of the module, a number guessing game and an application for controlling bulb brightness. Realized hardware extension shows a possibility of extending the module with a circuit, enabling a control of a stepper motor.

Key words: microprocessors, learning module of TUL, stepper motors, RTC circuit, programming language C

Obsah

Seznam použitých zkratek

PMP [Počítače a mikropočítače]
Zkratka předmětu vyučovaného na Technické univerzitě v Liberci.

PHS [Počítačový hardware a rozhraní]
Zkratka předmětu vyučovaného na Technické univerzitě v Liberci.

RTC [Real Time Clock]
Integrovaný obvod realizující hodiny reálného času.

USB [Universal Serial Bus]
Univerzální sériová sběrnice.

CAN [Controller Area Network]
Sériová datová sběrnice používaná především v automobilovém průmyslu.

I^2C [Inter-Integrated Circuit]
Sériová sběrnice vyvinutá firmou Philips.

UART [Universal Synchronous Asynchronous Receiver Transmitter]
Univerzální synchronní i asynchronní rozhraní.

SDCC [Small Device C Compiler]
Překladač jazyka C pro mikroprocesory.

LCD [Liquid Crystal Display]
Displej z tekutých krystalů.

LED [Light-Emitting Diode]
Dioda emitující světlo.

ACK [Acknowledge]
Bit potvrzující přenos dat.

PWM [Pulse Weight Modulation]
Pulzní šířková modulace.

IR [Infra Red]
Infračervené záření.

ROM [Read-Only Memory]
Paměť určená pouze ke čtení.

Seznam obrázků

Seznam tabulek

1 Úvod

Hlavní částí této práce je vytvoření aplikací a realizace hardwarového rozšíření pro výukový přípravek Technické univerzity v Liberci. Ten je určený především pro výuku předmětů PMP a PHS. Vytvořené aplikace a rozšíření mohou být pomůckou při výuce, nebo základem budoucích prací na přípravku. Součástí práce je seznámení se s výukovým přípravkem a všemi jeho součástmi, jako například použitého mikroprocesoru řady 8051 nebo displeje.

Mikroprocesory s jádrem 80C51 nalezneme v mnoha aplikacích, od řízení různých zařízení až po sběr dat. Díky jejich rozšířenosti je jedna z verzí tohoto mikroprocesoru umístěna právě na přípravku.

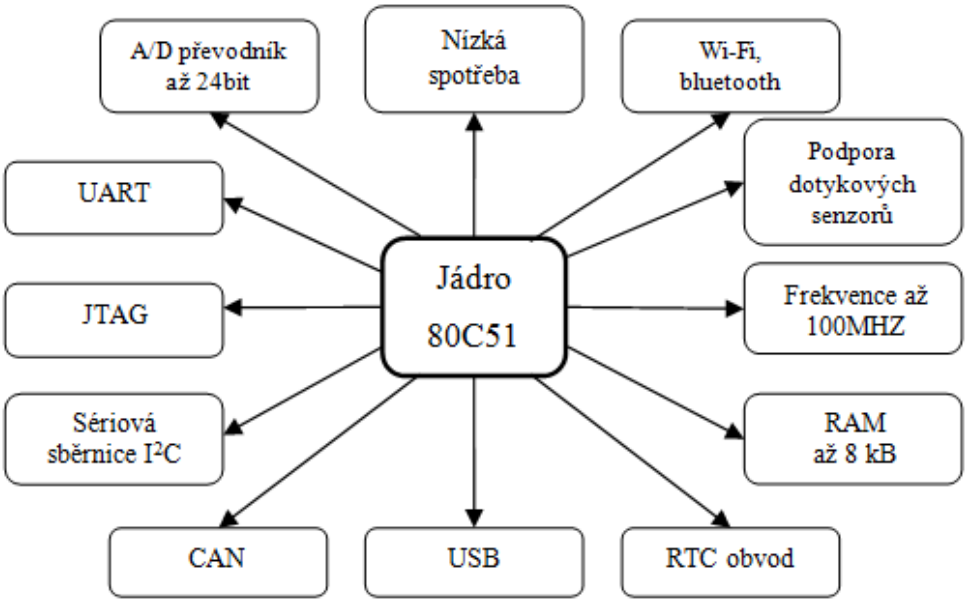
Práce je rozdělena na teoretickou a praktickou část. Teoretická část se zabývá problematikou programování mikroprocesorů, použitým programovacím jazykem C a jednotlivými součástmi výukového přípravku. Nejvíce popisované součásti jsou mikroprocesor, RTC obvod a displej.

Praktická část se zabývá programováním několika aplikací, jako digitální hodiny, tester přípravku nebo řízení jasu žárovky pomocí pulzní šířkové modulace. Praktická část obsahuje dva teoretické návrhy hardwarových rozšíření, které popisují požadavky a přibližný postup jejich řešení. Třetí návrh hardwarového rozšíření byl vybrán k realizaci a je rozpracován detailně, od teoretické přípravy, až do výsledné realizace a odzkoušení.

2 Teoretická rešerše

2.1 Mikroprocesory řady 8051

Mikroprocesor 8051 byl navržen v roce 1980 a jedná se tedy o relativně starý procesor. Díky tomu, že se stal u návrhářů velmi oblíben, používá se s různými rozšířeními až do současnosti. Má mnoho vývojových nástrojů a výrobou těchto mikroprocesorů se zabývá velké množství výrobců integrovaných obvodů (např. Atmel a Silicon Labs) Mikroprocesory, jsou obvykle rozšířeny o periferie, jako například I^2C sběrnici nebo podporu CAN. Obvyklá rozšíření jsou na obrázku 2.1 [1].



Obrázek 2.1: Obvyklá rozšíření základního jádra mikroprocesoru 8051

Většina mikroprocesorů, vyrobených společností Silicon Labs, má integrovány dva A/D převodníky. Jeden, který má více multiplexovaných vstupů a druhý, který má obvykle vyšší rozlišení a rychlost. V současné době, se většina výrobců orientuje hlavně na snižování spotřeby a velikost pouzdra. Komponenty, o které jsou mikroprocesory rozšiřovány, vycházejí především z aktuální poptávky.

2.2 Programovací jazyk C

Byl vyvinut v letech 1969 – 1973 v Bellových laboratořích Denisem Ritchiem. Původní myšlenkou bylo vytvořit jazyk pro použití s operačním systémem Unix. Jazyk C se stal velmi oblíbeným a v současné době jedním z nejrozšířenějších jazyků.

Jazyk C je známý hlavně díky své všeobecné použitelnosti. Programy napsané v tomto jazyce jsou rychlé, poměrně krátké a díky tomu i snadno čitelné. Hlavní výhodou je, že dobře napsaný program v C je obvykle stejně rychlý, jako program napsaný ve strojovém kódu [2].

Právě díky těmto dobrým vlastnostem byl jazyk C zvolen k realizaci bakalářské práce. Další možností byl jazyk symbolických adres, tedy assembler. Ten se ovšem častěji používá pro jednodušší aplikace, protože při programování složitějších aplikací se kód napsaný v assembleru stává velmi dlouhým, a tedy i nepřehledným. Dalším důvodem, proč byl zvolen právě jazyk C, jsou dobré předchozí zkušenosti.

2.3 Překladač SDCC

Překladač SDCC (Small Device C Compiler) je překladač pro 8 bitové mikroprocesory s otevřeným zdrojovým kódem, který je navržený tak, aby mohl generovat kód pro více platforem. Je distribuovaný pod licencí GNU General Public License (všeobecná veřejná licence GNU), tedy licencí napsanou pro svobodný software.

Aktuální verze SDCC je zaměřena na mikroprocesory založené na architektuře Intel MCS51 (8031, 8032, 8051, 8052 a další), Maxim 80DS390, Zilog Z80 a Motorola 68HC08.

SDCC provádí při překladu všechny standardní optimalizace kódu jako:

- Eliminace společných podvýrazů (global sub expression elimination)
- Optimalizace cyklů (loop optimizations)
- Optimalizace konstantních výrazů (constant folding & propagation)
- Nahrazení výrazů jejich hodnotami (copy propagation)
- Odstranění bezvýznamného kódu (dead code elimination)
- Optimalizace skoků přepínače (jump tables for switch statements)

Další výhodou SDCC je, že lze v programu, psaném v C použít také kód napsaný v assembleru a vložit jej do kterékoliv funkce.

2.3.1 Datové typy v SDCC

V SDCC jsou k dispozici všechny standardní datové typy, např. int, long aj. Tyto typy jsou tzv. znaménkové („signed“). Kromě standardních datových typů jsou ještě k dispozici datové typy odvozené, tzv. neznaménkové („unsigned“). Rozdíl mezi těmito typy spočívá v tom, zda ukládají znaménko (signed), nebo zda znaménko neukládají (unsigned) [3]. Všechny podporované datové typy jsou vypsány v tabulce 2.1.

Typ	Velikost	Výchozí rozsah	Signed	Unsigned
Bool	1 bit	Unsigned	-	0, 1
Char	1 bajt	Signed	-128; +127	0; +255
Short	2 bajty	Signed	-32.768; +32.767	0; +65.535
Int	2 bajty	Signed	-32.768; +32.767	0; +65.535
Long	4 bajty	Signed	-2.147.483.648; +2.147.483.647	0; +4.294.967.295
Float	4 bajty	Signed		1.175494351E-38; 3.402823466E+38
Pointer	1, 2, 3 nebo 4 bajty	Generic (obecný)		

Tabulka 2.1: Podporované datové typy [4]

Datové typy *bool*, *short* a *int* jsou celočíselné. *Float* a *double* jsou určeny k ukládání reálných čísel, někdy se také nazývají datové typy s plovoucí desetinou čárkou.

Datový typ *char* je určen pro ukládání znaků v kódování ASCII, znaky se zapisují mezi apostrofy. Pokud je zapotřebí zadat speciální znaky (například apostrof, konec řádku aj.), vkládá se před ně zpětné lomítko. Tyto posloupnosti se nazývají „escape sekvence“ [2]. Základní přehled těchto sekvencí je v tabulce 2.2. Datový typ char lze také použít pro uložení celého čísla v rozsahu 8 bitů.

Jméno	Posloupnost	Pořadové číslo v ASCII
Otazník	' \? '	63
Procenta	' \% '	37
Konec řádku	' \n '	10
Tabulátor	' \t '	9

Tabulka 2.2: Základní přehled „escape sekvencí“

3 Popis výukového přípravku

Aplikace a rozšíření jsou realizovány pro výukový přípravek Technické Univerzity v Liberci, vytvořený v předchozích letech. Výukový přípravek je určen pro výuku předmětů PMP a PHS. Jedná se o důmyslný přípravek, který může být různými způsoby upravován a rozšiřován o další komponenty.

Skládá se ze základní desky a desky s mikroprocesorem, kterou lze odpojit a nahradit za jinou. Od základní desky lze odpojit také LCD displej a klávesnici.

Celý přípravek je napájen napětím 5V z USB portu, který je schopen dodávat dostatečně velký proud. Pro komponenty vyžadující napětí 3,3V obsahuje přípravek regulátor napětí LM3940. Napájení z USB portu je možné odpojit a nahradit například adaptérem. Tím je zajištěna přenositelnost přípravku i na místa, kde není napájení z USB portu k dispozici. Fotografie výukového přípravku je na obrázku 3.1.

Obrázek 3.1 Fotografie výukového přípravku pro předměty PMP a PHS

Základní deska přípravku obsahuje:

- LED-bar
- Maticovou klávesnici
- LCD displej s řadičem HD44780
- 3 led diody (zelenou, žlutou a červenou)
- Teplotní čidlo SMT160
- Žárovku pro ohřev teplotního čidla
- RTC (Real time clock) obvod
- Piezoměnič
- Rozhraní RS-232
- USB převodník na RS-232 v TTL úrovních

Deska s mikroprocesorem obsahuje:

- Mikroprocesor AT89C51CC03
- Sběrnici CAN
- Dva posuvné registry pro připojení LED-baru

K připojení externích rozšíření slouží konektory SV1 a SV2, které jsou umístěny na základní desce (viz příloha A). Tyto dva konektory umožňují odpojení pinů mikroprocesoru od některých komponentů (např. LED-diody, teplotní čidlo aj.) a připojení těchto pinů k externím zařízením.

Schéma zapojení základní desky výukového přípravku je v příloze A, schéma zapojení desky s mikroprocesorem je v příloze B.

3.1 Mikroprocesor AT89C51CC03

Na přípravku je osazen mikroprocesor AT89C51CC03, tedy procesor s jádrem řady 89C51 od společnosti Atmel. Mikroprocesor lze napájet napětím v rozsahu 3 až 5,5V.

Hlavními výhodami použitého mikroprocesoru jsou [5]:

- Plná podpora pro síť CAN
- Podpora I^2C sběrnice
- 256B RAM
- 2048B ERAM
- 64kB flash paměti
- 2kB flash paměti pro zavaděč

3.2 A/D převodník

A/D převodník je zařízení, které slouží k převodu analogového signálu na číslicový. Nejčastějším vstupním signálem je napětí, výstupem je pak datové slovo o určitém počtu bitů v závislosti na typu převodníku [6].

Mikroprocesor AT89C51CC03 má integrovaný jeden 10 bitový aproximační A/D převodník, který má 8 vstupních kanálů. Přepínání mezi jednotlivými kanály je zajištěno pomocí multiplexoru.

3.3 LCD displej s řadičem HD44780

Na přípravku je umístěn LCD displej v provedení 2x8 znaků s řadičem HD44780 a LED podsvícením. Displej může pracovat ve 4 bitovém, nebo 8 bitovém režimu. Rozdíl mezi režimy spočívá v počtu pinů, potřebných pro zápis dat na displej. V paměti typu ROM je přednastaveno 240 různých znaků a dalších osm znaků může být nadefinováno uživatelem. Uživatelské znaky je zapotřebí při každém spuštění programu nadefinovat znovu. Displej lze napájet napětím v rozsahu 2,7 až 5,5V.

Displej má nadefinované tyto základní instrukce:

- Vymazání displeje
- Nastavení kurzoru na začátek řádku
- Zapnutí / Vypnutí displeje
- Zapnutí / Vypnutí kurzoru
- Blikání kurzoru
- Posun kurzoru
- Posun zobrazení na displeji

Řadič HD44780 se v současnosti stal standardem pro všechny běžné znakové displeje [7].

3.4 RTC obvod

Na přípravku je osazen RTC obvod DS1338C. Jedná se o RTC obvod s integrovaným krystalem, který je plně digitální a připojený pomocí sběrnice I^2C. Kromě integrovaného krystalu, je další hlavní výhodou možnost připojení záložního napájení (např. baterie). RTC obvod automaticky rozpozná ztrátu hlavního napájení a přepne na záložní. [8]

Hlavní výhody obvodu DS1338C jsou:

- RTC obvod počítá sekundy, minuty, hodiny, měsíce, dny a roky
- Ošetřený přestupný rok
- Integrovaný krystal
- 56 bajtů uživatelské paměti
- Připojení přes I^2C sběrnici
- Programovatelný výstupní signál
- Automatická detekce ztráty napájení

3.4.1 Sběrnice I^2C

Sběrnici I^2C vyvinula společnost Philips za účelem vzájemného propojení menších zařízení na jediné desce. Komunikaci mezi zařízeními zajišťují pouze dva vodiče, jejichž délka je omezena maximální kapacitou 400pF. Vodiče se nazývají SCL (hodinový signál) a SDA (datový signál). Každé zařízení má svou přidělenou 7 bitovou adresu, z toho plyne, že maximální počet připojených zařízení je teoreticky 127.

Výhody použití sběrnice I^2C:

- Sběrnice je pouze dvouvodičová
- Všechna připojená zařízení jsou softwarově adresovatelná
- Detekce kolize v případě, že přenos dat zahájí více zařízení současně
- Obousměrná komunikace dosahuje ve standardním módu rychlosti 100 kbit/s, v rychlém módu až 400 kbit/s [9]

3.5 Teplotní čidlo

Přípravek obsahuje teplotní čidlo SMT 160-30. Jedná se o polovodičové čidlo, které generuje digitální výstup. Výhodou digitálního výstupu je snadné připojení k různým druhům mikroprocesorů bez nutnosti použít A/D převodník [10].

Hlavní výhody čidla SMT 160-30:

- Rozsah teplot -45 °C až 130 °C
- Přesnost až ±0,7°C
- Kompatibilita s technologiemi TTL a CMOS
- Nízká spotřeba (méně než 1mW)
- Připojení přímo na datový pin procesoru, bez použití A/D převodníku

4 Demonstrační aplikace

Při návrhu demonstračních aplikací bylo používáno základních pravidel pro programování mikroprocesorů. Byl kladen důraz především na efektivní zdrojový kód s rychlou odezvou, minimálním využitím paměti a současně s plným využitím veškerých možností přípravku.

4.1 Aplikace realizující digitální hodiny

Hodiny skutečného času zajišťuje RTC obvod. Součástí digitálních hodin je datum, minutka a budík. Aplikace poskytuje jednoduché a přehledné menu pro nastavení času, data a dalších funkcí. K ovládání se používá výhradně maticové klávesnice. Po prvním spuštění je aplikace na základní obrazovce, kde zobrazuje aktuální čas, datum, stav budíku a minutky.

Funkce *main* obsahuje inicializaci periferií (displej, RTC obvod, přerušení) a hlavní cyklus programu. Ten slouží především k přepínání mezi jednotlivými stavy a k volání funkcí pro čtení dat z RTC. Obsahuje konstrukci „switch-case“, která obsluhuje přepínání mezi třemi zobrazeními – čas, budík a minutka. Součástí hlavního cyklu je dále kontrola stavu budíku, minutky, piezo-měniče a LED-baru. Ke kontrole budíku a minutky cyklus přistupuje pouze, jsou-li aktivní.

Konstrukce „switch-case“ mezi jednotlivými obrazovkami přepíná na základě stisknutí klávesy „A“. V případě že dojde ke stisknutí klávesy „B“ přepne program do režimu nastavení položky, která je aktuálně zobrazena na displeji. Například pokud jsou na displeji zobrazeny hodiny, zobrazí se menu pro jejich nastavení. Všechna menu, která slouží k nastavení, lze kdykoliv opustit opětovným stisknutím klávesy „B“.

Čtení dat z RTC zajišťuje především funkce s názvem *DS1338_get()*, ta na základě získaného parametru přečte požadovanou položku (minuty, hodiny, sekundy atd.). Vyžádané hodnoty jsou ukládány do příslušných proměnných – *sec*, *min*, *hour*, *date*, *month* a *year*. Všechny tyto proměnné jsou typu *short int* a jsou definovány jako globální.

Pro výpis času a data na displej slouží funkce *print_time()* a *print_date()*. Pokud je čas nebo datum pouze jednociferná číslice, automaticky se před ní přidá nula. Tím je vždy zachován stejný počet digitů. Při výpisu času se mezi hodiny, minuty a sekundy

vkládá dvojtečka. Při výpisu data se vkládá mezi den, měsíc a rok pravé lomítko. Protože RTC obvod ukládá z celého roku pouze poslední dvojčíslí, vkládají se před položku rok ještě číslice 20 popř. 200.

Funkce budíku je založena na porovnávání uloženého času budíku s aktuálním časem v proměnných *sec, min* a *hour*. Nastavení budíku je uloženo v proměnných *alarm_hour (hodiny), alarm_min (minuty)* a *alarm_stat (stav budíku – zapnuto, vypnuto).* V případě budíku se tedy nepředpokládá potřeba ukládat sekundy. Nastavení budíku je současně uloženo v uživatelské paměti RTC obvodu, která má dostačující velikost 56 bajtů. Výhodou takového řešení je zachování nastavení i v případě ztráty napájení. Proměnné *alarm_min, alarm_hour* a *alarm_stat* jsou tedy synchronizovány s vyhrazenými bajty v uživatelské paměti RTC. K čtení této uživatelské paměti slouží funkce *DS1338_getalarm().* K zápisu slouží funkce *DS1338_setalarm().* Čas budíku je v paměti uložen stejným způsobem, jako čas aktuální viz kapitola 4.1.3. Čtení stavu budíku je zajištěno funkcí *DS1338_getalarm()*. Zápis stavu budíku je zajištěn funkcí *DS1338_setalarmstat().* Pokud je budík ve stavu zapnuto, zobrazuje se na displeji malé písmeno „a“. Podmínka pro porovnávání času budíku s aktuálním časem je obsažena v hlavním cyklu. Pokud je tato podmínka platná nastaví se hodnota proměnné *beeper_status* na 1. Pokud je tato proměnná rovna 1, piezo-měnič je aktivní.

Minutka přepočítává zadaný čas na sekundy a v případě potřeby zobrazení na displeji zase zpět. Sekundy jsou uloženy v proměnné *timer,* která je typu unsigned int. Pokud minutka běží, dekrementuje se každou sekundu proměnná *timer*. Dekrementace je prováděna v hlavním cyklu. Je-li hodnota proměnné *timer* nulová, zapíše se do proměnné *beeper_status* hodnota 1. Stav minutky (zapnuto, vypnuto) je na displeji vyjádřen malým písmenem „t“.

Nastavení času provádí funkce *setting_time().* Ta ihned po zavolání smaže celý displej, vypíše aktuální čas a zapne blikající kurzor. V dalším kroku vstoupí funkce do cyklu while. Cyklus obsahuje strukturu „switch-case“, která určuje, jaký digit je aktuálně nastavován. Při stisknutí číselné klávesy se hodnota klávesy uloží a aktuálně nastavovaný digit se změní na digit následující. Tento proces se neustále opakuje, dokud není stisknuta klávesa „B“, která cyklus ukončí. Po skončení cyklu proběhne kontrola, zda jsou vložené hodnoty platné. Pokud ne, nastaví se nejvyšší možná

hodnota. Například pokud je vložená hodnota 25 hodin, automaticky se změní na 23. Pomocí funkce *DS1338_settime()* se nové hodnoty nastaví do RTC.

Nastavení data zajišťuje funkce *setting_date()*, která je zavolána bezprostředně po nastavení času. Princip nastavení data je shodný s principem pro nastavení času s výjimkou kontroly vložených hodnot. U data je navíc ošetřena situace, kdy má například každý měsíc odlišný počet dní nebo případ přestupného roku. Nastavení budíku a minutky je principielně shodné s nastavením času.

4.1.1 Ovládání piezo-měniče

K ovládání piezo-měniče slouží funkce *Beep()*, jedná se o funkci, která vygeneruje na piezo-měniči jedno krátké pípnutí. Generování pípnutí je založeno na rychlém střídání log. 0 a log. 1 na výstupu mikroprocesoru, který je připojen na vstup piezo-měniče. Frekvence střídání odpovídá frekvenci generovaného zvuku.

Na výstupu je log. 0 a log. 1 vystřídána 200x, mezi každou změnou je generováno zpoždění. Aby byla zajištěna rychlá odezva na ukončení činnosti piezo-měniče, je součástí cyklu kontrola proměnné *beeper_status*. Pokud je proměnná rovna nule, cyklus se přeruší. Piezo-měnič lze tedy vypnout i během generování písknutí a není potřeba čekat na dokončení.

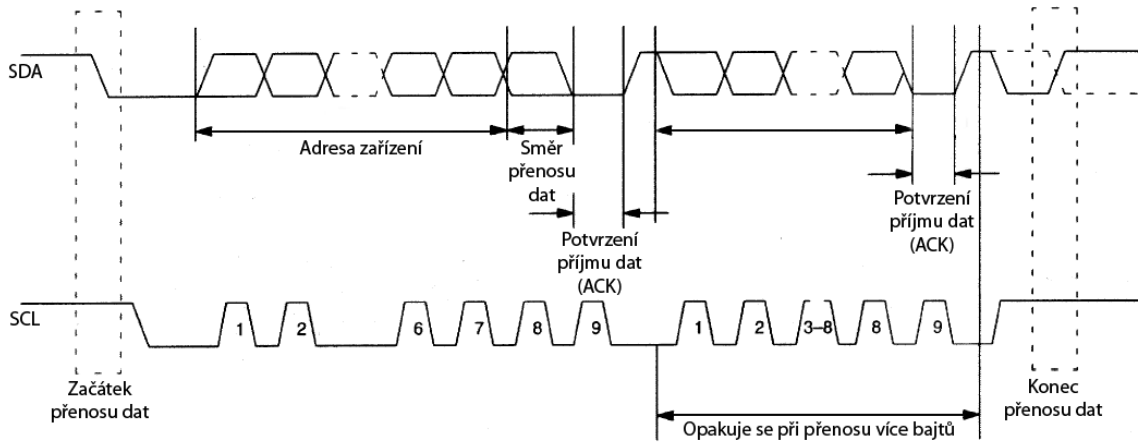
4.1.2 Komunikace po sběrnici I^2C

Po sběrnici může komunikovat vždy pouze jedno zařízení, sběrnice I^2C tedy nepodporuje tzv. duplexní přenos dat. Zařízení, které řídí data na sběrnici je nazýváno master. Zařízení, která jsou masterem řízena, se nazývají slave. Master vysílá na sběrnici hodinový signál, řídí přístup ostatních zařízení ke sběrnici a generuje podmínky pro začátek a konec přenosu dat. Jestliže je hodinový signál v log. 1, musí mít datový signál ustálenou hodnotu. Je-li hodinový signál při změně datového signálu v log. 1, je signál považován za signál řídící [8].

Na sběrnici může nastat 5 různých stavů:

- **Sběrnice je volná** – SDA i SCL jsou v log. 1
- **Začátek přenosu dat (inicializace sběrnice)** – přechod SDA z log. 1 do log. 0, když SCL je v log. 1
- **Konec přenosu dat** – přechod SDA z log. 0 do log. 1, když SCL je v log. 1
- **Přenos dat** – stav na datové sběrnici představuje přenášená data. Na jednu periodu hodinového signálu připadá jeden bit dat. Stav SDA může být změněn pouze, je-li SCL v log. 0.
- **Potvrzení příjmu dat (ACK)** – zařízení potvrzující příjem dat musí SDA převést na log. 0 v okamžiku, kdy zařízení master vyšle potvrzující hodinový signál

Přenos dat může být zahájen pouze, je-li sběrnice volná (SDA i SCL je v log. 1). Pokud je tato podmínka splněna, nastaví vysílající zařízení hodnoty SDA a SCL na stav, který zahajuje přenos dat. Všechna ostatní zařízení na sběrnici naslouchají. Vysílač pošle na sběrnici sedmibitovou adresu v sériové podobě a zařízení, které má odpovídající adresu odešle bit ACK. K odesílané adrese je ještě přidán osmý bit, který určuje směr toku dat, tedy zda bude zařízení data odesílat, nebo přijímat. Přijme-li vysílač ACK bit, začne odesílat/přijímat data. Po odeslání/příjmu 8 bitů opět odešle zařízení potvrzovací bit ACK. Tento proces se opakuje, dokud nejsou přenesena všechna požadovaná data. Po přenosu všech dat nastaví vysílač na sběrnici stav pro ukončení přenosu [8]. Časový diagram je na obrázku 4.1.



Obrázek 4.1: Časový diagram přenosu dat po sběrnici

Detekce kolize je založená na porovnávání vysílaných bitů se skutečným stavem sběrnice. Jestli-že zařízení zjistí, že stav sběrnice neodpovídá očekávanému stavu, indikuje kolizi. Zařízení, které vysílá na SDA log. 1 a při kontrole zjistí, že úroveň SDA je v log. 0, musí okamžitě přenos ukončit [9].

4.1.3 Obsluha RTC

Všechny funkce pro obsluhu RTC využívají funkce z knihovny pro komunikaci přes I^2C sběrnici. Při každém zápisu, popř. čtení z RTC, je nutné nastavit „register pointer“ na registr, se kterým budeme pracovat. Před použitím je potřeba RTC obvod správně inicializovat. K tomu byla napsána funkce s názvem *DS1338_Init()*. Základem inicializace je správné nastavení bitu CH (povoluje/zakazuje hodiny) a OSF (povoluje/zakazuje oscilátor). Oba zmiňované bity se musí rovnat log 0.

Adresa	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Funkce	Rozsah
0h	CH	10 - sekund			1 - sekund				sekundy	00-59
01h	0	10 - minut			1 - minut				minuty	00-59
02h	0	$12/\overline{24}$	$\overline{AM}$/PM 10-hodin	10-hodin	1 - hodin				hodiny	1-12 +AM/PM 00-23
03h	0	0	0	0	0	den v týdnu			den v týdnu	1-7
04h	0	0	10 - dnů		1 - dnů				dny	01-31
05h	0		0	10 - měsíců	měsíc				měsíce	01-12
06h	10 - roků				rok				roky	00-99
07h	OUT	0	OSF	SQWE	0	0	RS1	RS0	ovládání	
08-3Fh									56 x 8	00H-FFH

Tabulka 4.1: Rozložení paměti v RTC, kde 10 – vyjadřuje desítky a 1 - jednotky

Z tabulky 4.1 je patrné, že bit CH je ve stejném registru jako sekundy. Proto je při jeho zápisu potřeba dbát na zachování ostatních bitů v původní podobě.

Postup inicializace je následující:

- Připojení k RTC pro čtení
- Nastavení registr pointeru na 0x00
- Přečtení registru
- Maskování registru maskou 0b01111111
- Zápis namaskovaného registru zpět do RTC
- Zápis hodnoty 0b10010011 do ovládacího registru

Jak již bylo zmíněno v kapitole 4.1, ke čtení dat z RTC obvodu slouží funkce *DS1338_get()*. Funkci je předán parametr, na základě kterého je nastaven „register pointer“ na příslušnou adresu (adresa hodin, minut atd.). Předávané parametry jsou definovány v tabulce 4.2.

Parametr:	SEC	MIN	HOUR	DATE	MONTH	YEAR
Adresa:	0x00	0x01	0x02	0x04	0x05	0x06

Tabulka 4.2: Předávané parametry

Přečtená data se uloží do lokální proměnné *value* typu unsigned char. Z tabulky 4.1 je patrné, že jednotlivé položky času a data jsou v paměti rozděleny na desítky a jednotky. Získaná hodnota v proměnné *value* je tedy dále zpracována a vrácena jako datový typ short int. Zápis do RTC je prováděn obdobným způsobem jako čtení.

Příklad přenosu dat z RTC:

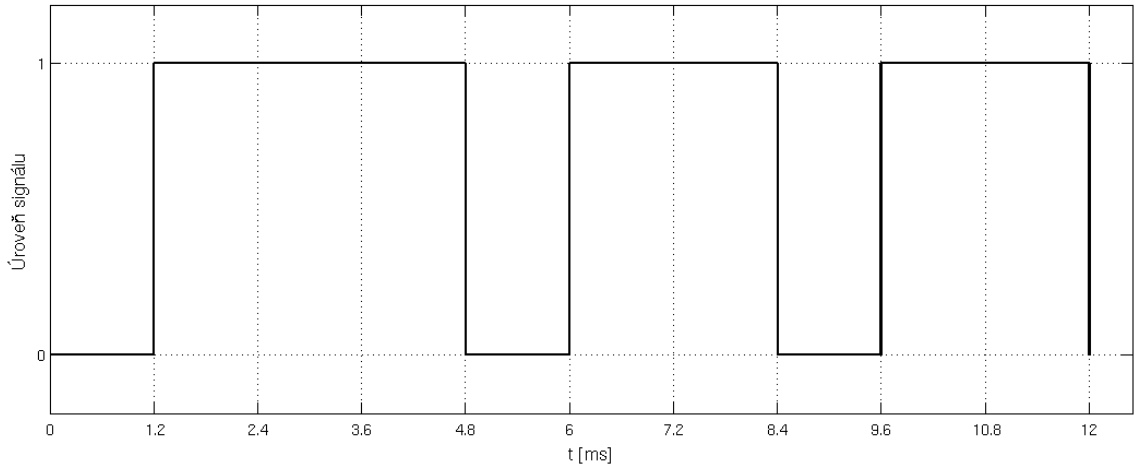
- I2C_start() – inicializace sběrnice
- I2C_write(DS1338_ID) – připojení k RTC, kde DS1338_ID je přidělená adresa RTC
- I2C_write(0x00) – nastavení „register pointeru“ v RTC
- I2C_start() – opětovná inicializace
- I2C_write(DS1338_ID+1) – připojení k RTC pro čtení
- I2C_read() – přečtení požadovaných dat
- I2C_stop() – ukončení přenosu dat

4.2 Aplikace pro řízení jasu žárovky

Cílem aplikace je ukázat možnost řízení jasu žárovky pomocí pulzní šířkové modulace (PWM). Změny mezi jednotlivými stavy žárovky jsou prováděny velmi vysokou rychlostí, proto je lidské oko není schopno rozeznat. Dalším důvodem, který znemožňuje pozorovatelnost změn, je vlákno žárovky, které mění svůj stav relativně pomalu.

Pulzní šířkovou modulaci obsluhuje přerušení (nastavení přerušení je popsáno v kapitole 4.5). Z toho plyne, že jednotlivé pulzy jsou dlouhé 1,4 ms. Základem je proměnná *PWM* typu short int. V proměnné je uložena konstanta, podle které se mění

poměr času, kdy žárovka svítí, nebo nesvítí, tedy střída. Na začátku každého přerušení je stav žárovky změněn na „zapnuto“. Proměnná *PWM* je přičítána do proměnné *temp*, a pokud obsah *temp* je větší, než konstanta přednastavená na hodnotu 1000, změní se stav žárovky na „vypnuto“. Žárovka v tomto stavu setrvá až do zavolání dalšího přerušení. Tímto způsobem je zajištěno pokrytí celého rozsahu od stavu, kdy žárovka svítí, až do stavu, kdy nesvítí vůbec. Předpokládaný průběh jedné periody PWM je na obrázku 4.2, hodnota proměnné *PWM* je 300. K nastavení hodnoty v proměnné *PWM* je využíván A/D převodník a potenciometr v rozsahu 0 - 1000. Použití A/D převodníku a potenciometru je popsáno v kapitole 4.4.1.



Obrázek 4.2: Jedna perioda generované PWM při konstantě 300

4.3 Aplikace realizující hru hádání čísel

Hra hádání čísel spočívá v uhodnutí náhodně vygenerovaného čísla v rozsahu od 0 do 100. Uživatel zadá číslo, které tipuje. Poté se na displeji se zobrazí, zda tipované číslo je větší, nebo menší než číslo, které bylo vygenerováno. Uživatel tipuje čísla, dokud se jeho tip neshoduje s vygenerovaným číslem. Po uhodnutí se na displeji zobrazí počet pokusů, které uživatel potřeboval.

Pro generování náhodných čísel jsou používány funkce *rand()* a *srand()*. Funkce *rand()* vrací pseudonáhodná čísla od 0 do maximální zadané hodnoty. Pseudonáhodná čísla jsou čísla z řady, která je vygenerována na základě složitých matematických podmínek. Aby čísla nebyla po spuštění programu vždy stejná, je zapotřebí přidat náhodný faktor. K tomu slouží funkce *srand(),* která určí počátek číselné řady na základě předaného parametru. U běžných počítačů je parametr obvykle získáván z aktuálního data a času.

Protože u mikroprocesorů tato možnost není, bylo navrženo řešení, které využívá k získání parametru čítač/časovač 0. Čítač/časovač 0 běží ihned po spuštění a je nastaven v 16 bitovém módu. Ke spuštění první hry musí uživatel stisknout libovolnou klávesu. Ta ještě před začátkem hry zavolá funkci *srand()* a jako vstupní parametr jí předá obsah bitů TH0 a TL0 z čítače/časovače 0. Náhodný faktor je tedy vytvořen uživatelem, na základě okamžiku, kdy je spuštěna první hra.

Protože při frekvenci krystalu 20MHz se obsah bitů TH0 a TL0 změní přibližně 1666667x za 1 sekundu, je pravděpodobnost, že uživatel stiskne klávesu dvakrát za sebou ve stejný okamžik, velmi nízká.

Během vývoje byly vyzkoušeny i jiné způsoby pro získání náhodného faktoru. Například čtení dat z A/D převodníku, nebo klávesa, kterou uživatel stiskne při prvním spuštění. Tyto způsoby se ovšem neprojevily jako dostačující.

4.4 Aplikace pro testování přípravku

Tato aplikace slouží k otestování všech součástí přípravku. Např. displej, RTC obvod, A/D převodník, klávesnici, LED diody a další. Aplikace disponuje jednoduchým a přehledným uživatelským rozhraním, které využívá displej a klávesnici. Po spuštění se na základní obrazovce zobrazuje posunující se text, který uživateli poskytuje jednoduchý návod na ovládání. Funkce *main* obsahuje strukturu „switch-case“, která na základě stisknuté klávesy volá příslušný podprogram testování. Jednotlivé podprogramy je možné opustit stiskem libovolné klávesy. V následujících třech podkapitolách jsou popsány nejdůležitější části aplikace.

4.4.1 Test A/D převodníku

A/D převodník je možné použít dvěma způsoby. První se nazývá precizní a druhý standardní.

Precizní režim převodníku využívá přerušení a dočasné „uspání“ (pseudo-idle mode) procesoru, kdy samotný procesor neběží, ale jeho periferie ano. V tomto režimu se snižuje podíl digitálního šumu a převod analogové hodnoty je tedy přesnější. Pro opětovné „probuzení“ procesoru je zapotřebí přerušení, které je vyvoláno koncem převodu analogové hodnoty.

Standardní režim převodníku nevyužívá dočasné „uspání“ procesoru. To snižuje přesnost převodu, ale zároveň velice zjednodušuje jeho použití [5].

V aplikaci je A/D převodník využíván pro čtení hodnoty potenciometru. K tomuto použití postačuje standardní režim.

Ukázka použití A/D převodníku ve standardním režimu:

- ADCF=0x1 – nastavení portu, na kterém jsou jednotlivé kanály
- ADCON = 0x20 – povolení A/D převodníku
- ADCON &= 0xF8 – vyčištění SCH2-0
- ADCON |= 0x0 – nastavení kanálu
- ADCON |= 0x08 – start převodu
- while((ADCON & 0x01) != 0x01) – čekání na dokončení převodu
- ADCON &= 0xEF - vyčištění příznaku konce převodu
- value_converted = (ADDH << 2)+(ADDL & 0x03) – přečtení převedené hodnoty

Zpracovaná hodnota z A/D převodníku (pro tento případ stav potenciometru) je uložena do proměnné typu short int a zobrazena na displeji.

4.4.2 Test teplotního čidla

Čidlo převádí teplotu na střídu s lineární odezvou na měřitelný rozsah podle vztahu D.C. = 0.320 + 0.00470t, kde „t“ je teplota a „D.C.“ střída. Z toho plyne výsledný vzorec pro teplotu 4.1.

$$t = \frac{D.C. - 0.320}{0{,}0047}$$

(Vzorec 4.1)

Měření střídy je možné provádět tak, že se v náhodném čase odměří větší počet vzorků a na základě celkového počtu vzorků a počtu vzorků, které obsahují hodnotu 1, je vypočítána střída. Tuto metodu lze jednoduše realizovat, ale měření bohužel není příliš přesné a je časově náročné, protože jsou zapotřebí řádově stovky vzorků.

Další metodou, jak střídu změřit, je metoda přímého měření času, kdy je výstup čidla na hodnotě log. 1 a kdy na log. 0. Hlavní výhodou této metody je vysoká rychlost, protože k přesnému určení střídy postačí pouze jedna perioda signálu. Nevýhodou je

složitější realizace, která je způsobena velikostí frekvence na výstupu čidla. Ta se pohybuje v rozsahu 1 – 4kHz. Při měření musí být na mikroprocesoru zakázána všechna přerušení, aby nedošlo k chybnému měření. K měření času je využíván čítač/časovač 0. Měří-li se doba, po kterou je signál v horní úrovni, mikroprocesor počká na náběžnou hranu signálu, ihned vynuluje čítač/časovač 0 a počká na sestupnou hranu. Obsah bitů čítače/časovače TH0 a TL0 se uloží do proměnných a vynuluje. Měření času dolní úrovně signálu probíhá obdobným způsobem.

Výsledná střída se vypočítá z doby trvání jedné periody (time0 + time1) a z doby, kdy je signál v horní úrovni (time1). Viz vzorec 4.2. Teplotu lze na základě změřené střídy vypočítat podle vzorce 4.1.

$$D.C. = \frac{time1}{(time0 + time1)}$$

(Vzorec 4.2)

4.4.3 Test posuvného registru a LED-baru

Na přípravku je umístěn LED-bar s deseti LED diodami. Protože připojení jednotlivých LED přímo na piny mikroprocesoru by bylo z různých důvodů nepraktické, je LED-bar připojen přes posuvný registr 74HC595D. To zajišťuje značné snížení počtu použitých pinů mikroprocesoru. Jedná se o sériový posuvný registr s 8 výstupy. Jelikož LED-bar má LED diod deset, je potřeba použít dva sériově zapojené posuvné registry.

Aplikace provede na LED-baru postupné rozsvícení a zhasnutí jednotlivých diod. Tato sekvence se opakuje, dokud uživatel nestiskne libovolnou klávesu.

4.5 Sdílené součásti aplikací

Všechny aplikace mají sdílené funkce ovládání maticové klávesnice, displeje a zpoždění. K ovládání displeje slouží funkce *LCD_Init()* (inicializace displeje), *LCD_SendCmd()* (pro zasílání ovládacích příkazů) a *LCD_SendData()* (pro zasílání dat). Pro výpis na displej slouží funkce *putchar()* (předávání znaků z funkce printf na displej), *LCD_goto()* (pro skok na danou pozici displeje) a *scroolmessage()* (pro automatické posouvání delšího textu na displeji). Klávesnici obsluhují funkce *HandleMatrix()* (generuje číslo klávesy na základě stavu příslušných portů) a *GetKey()* (ukládá na základě vygenerovaného čísla klávesy příslušný znak do proměnné *znak*).

Všechny aplikace mají zavedené přerušení z čítače/časovače 0, který je nastaven v 16 bitovém módu. Čítač/časovač 0 je ručně přeplňován při každém přerušení na hodnoty TL0=0 a TH0=248, což odpovídá při frekvenci krystalu 20MHz přibližně 1,4ms. K nastavení čítačů/časovačů slouží registr TMOD, ten je nastaven na hodnotu 0x01. Přerušení je nastaveno v registru IE na hodnotu 0x91.

Výpočet čítače/časovače:

Jedna perioda:

$$T = \frac{12}{f} = \frac{12}{20 \cdot 10^6} = 0{,}6 \cdot 10^{-6} s$$

(Vzorec 4.3)

Doba přetečení čítače:

$$t_0 = T \cdot (256 - TH0) \cdot 256 + 256 - TL0$$

(Vzorec 4.4)

$$t_0 = 0{,}6 \cdot 10^{-6} \cdot (256 - 248) \cdot 256 = 0{,}001382\ s \cong 1{,}4\ ms$$

Nastaveni čítače/časovače:

TMOD=0x01

TCON=0x50

Přeplňování:

TL0=0

TH0=248

Nastavení přerušení:

IE=0x92

V každém přerušení dojde k přečtení stavu klávesnice pomocí zmiňované funkce *HandleMatrix().* V případě, že je stisknuta libovolná klávesa, je zavolána funkce *GetKey()*.

Další sdílenou součástí je funkce *Delay()* realizující zpoždění. Tato funkce využívá proměnné *TIMER* typu word, která je dekrementována v každém cyklu přerušení. Proměnná *TIMER* tedy přeteče přibližné každých 10 ms.

Funkce *LCD_goto* využívá možnosti displeje přejít kurzorem na jakýkoliv segment. Každý segment má příslušnou paměťovou buňku. Při zavolání funkce jsou

předávány dva parametry – řádek a pozice. Z těchto dvou parametrů je vypočítána adresa požadované buňky, která je předána displeji. Při dalším použití funkce printf se odeslané znaky zapisují od požadované buňky.

5 Návrh hardwarových rozšíření

Součástí zadání bylo také navrhnout několik rozšíření a jedno z nich realizovat. Rozšíření se k přípravku připojují přes konektory SV1 a SV2 (viz kapitola 3). Byla navržena tři rozšíření, dvě jsou popsána v kapitolách 5.1 a 5.2. Třetí rozšíření bylo vybráno k realizaci, a proto je podrobně popsáno v kapitole 6.

5.1 Rozšíření o světelné senzory

První rozšíření by spočívalo v propojení přípravku s několika světlenými senzory. Rozšíření by mohlo sloužit například k automatickému rozsvěcení pouličního osvětlení. Senzory by byly rozmístěny na různých místech v obci. V případě, že by většina senzorů indikovala tmu, osvětlení by bylo aktivováno. Větší počet senzorů by zajišťoval eliminaci případné chyby. Chyba by mohla být způsobena například dočasným zastíněním senzoru (mraky na obloze, zaprášení senzoru aj.).

Analogová hodnota senzorů by byla převáděna na digitální signál pomocí A/D převodníku a dále softwarově zpracována. Mikroprocesor na přípravku teoreticky umožňuje připojení až 8 senzorů. Na konektor SV1 jsou ovšem přivedeny pouze 3 kanály z A/D převodníku, což omezuje počet senzorů na 3. V případě potřeby připojení více senzorů by bylo nutné přepínání mezi jednotlivými senzory realizovat hardwarově.

Obslužný software by měl umožňovat ruční zapnutí a vypnutí osvětlení, nastavení úrovně, při které má být pouliční osvětlení zapnuto, a další potřebná nastavení.

5.2 Rozšíření o optickou bránu

Druhé rozšíření by realizovalo jednoduchou optickou bránu propojenou s přípravkem. Rozšíření by mohlo sloužit například k realizaci počítadla průchodů nebo jako blokovací prvek automatické závory.

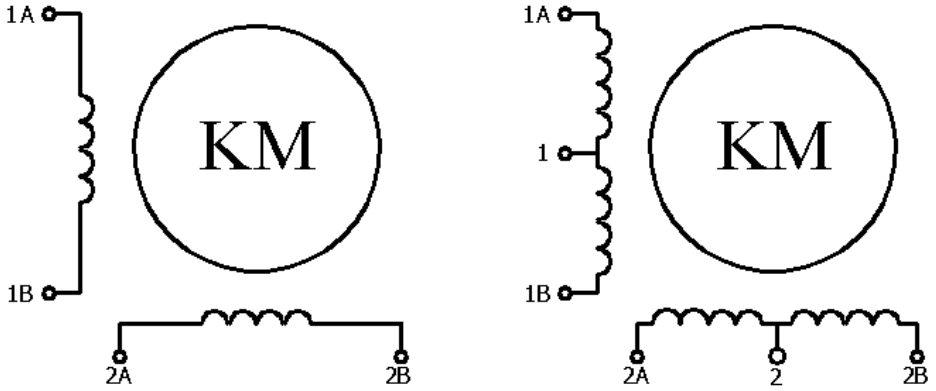
Základem by byla IR dioda a IR přijímač. Paprsek z IR diody by bylo zapotřebí modulovat na vyšší frekvenci (přibližně 35kHz), aby nedocházelo k interakci s běžným denním světlem. V závislosti na frekvenci paprsku by byl vybrán nejvhodnější IR přijímač. K modulaci paprsku lze využít například obvod 555, který má všechny potřebné vlastnosti. Výstup z rozšíření by měl dva stavy – log. 0 a log. 1. V případě přerušení paprsku by stav byl log. 0.

6 Realizace hardwarového rozšíření

Třetí rozšíření umožňuje řídit bipolární krokový motor. Hlavním cílem rozšíření bylo prozkoumat možnosti řízení krokových motorů pomocí přípravku. Využití by bylo především v různých složitějších soustrojích a jako základ pro budoucí rozšiřování výukového přípravku.

„Krokový motor (KM) je impulsně napájený motor, jehož funkční pohyb je nespojitý a děje se po jednotlivých úsecích (krocích). K řízení krokového motoru slouží ovladač krokového motoru.“(citace, [11], s. 2).

Krokové motory se podle konstrukce rozdělují na motory s pasivním rotorem (rotor je z feromagnetického materiálu) a na krokové motory s aktivním rotorem (rotor obsahuje permanentní magnet). Krokové motory lze ještě rozdělit podle toho, zda jsou unipolární, nebo bipolární. Unipolární motory mají vyvedené středy cívek. Bipolární motory středy cívek vyvedené nemají. Náhledy zapojení obou typů motorů jsou na obrázku 6.1.



Obrázek 6.1: Náhled zapojení krokového motoru - vlevo bipolární, vpravo unipolární

Ovladač krokového motoru je zařízení, které generuje posloupnost impulzů a obsahuje výkonové prvky pro buzení jednotlivých cívek motoru. Z posloupnosti impulzů vychází směr otáčení a jeho rychlost. V případě rozšíření je ovladač výukový přípravek, ke kterému je pomocí konektoru SV1 připojena výkonová část.

Výhoda krokových motorů spočívá v tom, že při řízení je známa poloha natočení hřídele. Proto se často používají například v robotice nebo automobilovém průmyslu.

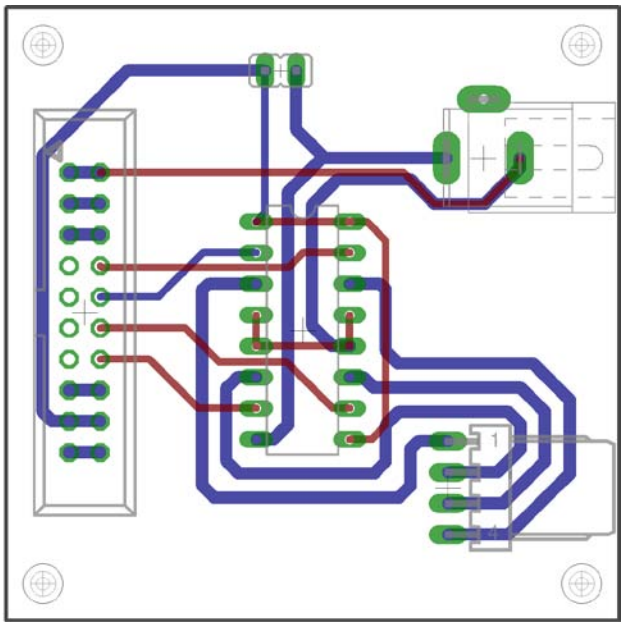
Princip otáčení krokového motoru je založený na vhodném zapojování jednotlivých cívek statoru tak, že vytvoří točivé magnetické pole.

6.1 Návrh zapojení

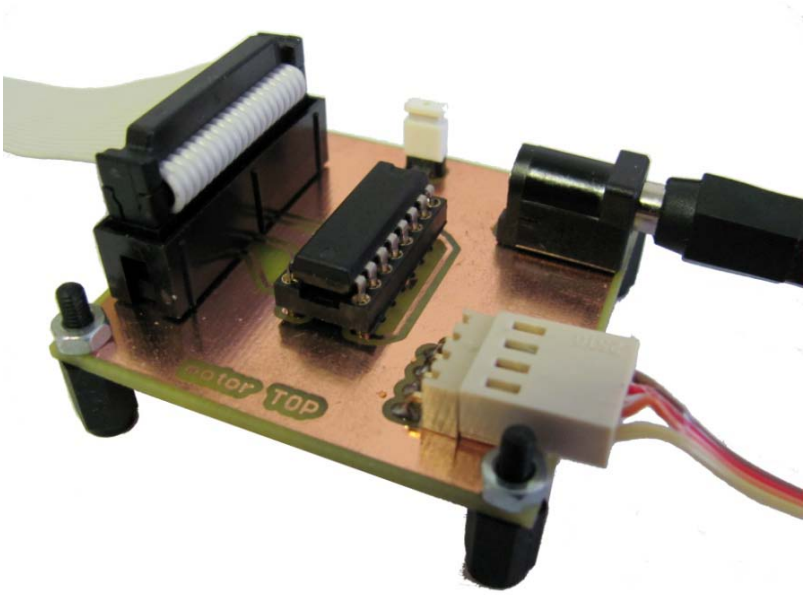
V rozšíření bylo použito krokového motoru, který je běžně dostupný v 3,5 palcových disketových mechanikách, kde zajišťuje pohyb čtecí hlavou. Jedná se o bipolární krokový motor s 10 póly, který je ovládán pomocí čtyř vodičů a napájen napětím 5V. Počet pólů byl vizuálně ověřen na rozebraném motoru shodného typu.

Základem zapojení je H-můstek L293D, ten může být napájen napětím v rozsahu 4,5 až 36V s maximálním proudem 600mA. Tyto parametry jsou pro většinu menších krokových motorů dostačující. H-můstek L293D byl vybrán především proto, že má na výstupech integrované diody pro potlačení indukčního přechodového jevu. Tím se zapojení výrazně zjednoduší.

K napájení rozšíření je používán externí adaptér připojený přes napájecí jack s napětím 5V. Napájecí napětí z externího adaptéru je současně přiváděno na výukový přípravek přes jumper a použitý konektor SV. V případě, že chceme pro krokový motor použít vyšší napětí než 5V, je zapotřebí jumper rozpojit, aby na přípravku nedošlo k přepětí. Rozšíření lze také napájet zpětně přes jumper přímo z přípravku. Tuto možnost lze ovšem použít jen u počítačů, které jsou schopné dodávat na sběrnici USB dostatečně velký proud. Schéma zapojení je v příloze C. Návrh desky plošného spoje je na obrázku 6.2. Zrealizované rozšíření je na obrázku 6.3.



Obrázek 6.2: Návrh desky rozšíření



Obrázek 6.3: Fotografie zrealizovaného rozšíření

6.2 Aplikace pro řízení

Hlavním úkolem aplikace pro řízení je demonstrovat funkčnost zapojení a základní principy řízení krokového motoru. Aplikace umožňuje měnit rychlost a směr otáčení.

Na displeji jsou vždy zobrazeny informace o aktuálním pohybu motoru. K ovládání slouží maticová klávesnice a potenciometr. Klávesnicí je ovládáno spuštění, zastavení a změna směru otáčení. Potenciometrem je ovládána rychlost otáčení.

Protože se jedná o lineární potenciometr, je jeho výstup přepočítáván do logaritmického rozsahu od 0 do 100. Logaritmický rozsah zajišťuje přesnější řízení při vyšších otáčkách.

Při řízení krokového motoru je potřeba zajistit, aby čas mezi jednotlivými kroky byl vždy stejný. K tomu je používáno přerušením z čítače/časovače 0, které má nejvyšší prioritu. Čítač/časovač 0 je v 16 bitovém módu a je přeplňován na hodnoty TH1=247 a TL1=60. Při těchto hodnotách přeteče čítač/časovač 0 přibližně každých 1,5ms. Krokový motor je řízen výhradně z uvedeného přerušení.

Obsluha přerušení obsahuje konstrukci „switch-case“, která přepíná mezi jednotlivými stavy cívek motoru podle toho, kterým směrem se má motor otočit. Celá konstrukce je uzavřena v jednoduché podmínce, která je platná každé n-té přerušení. Například pokud je podmínka platná každé druhé přerušení, je doba mezi jednotlivými kroky motoru 3ms. Změnou doby mezi jednotlivými kroky je možné řídit rychlost otáčení motoru.

Sekvence přepínání cívek, která provede na krokovém motoru jeden krok vpravo je v tabulce 6.1, krok vlevo vyjadřuje tabulka 6.2. Pro jeden krok jsou zapotřebí 4 stavy cívek.

	M1A	M1B	M2A	M2B
Stav 1	1	1	0	0
Stav 2	0	1	1	0
Stav 3	0	0	1	1
Stav 4	1	0	0	1

Tabulka 6.1: Sekvence jednoho kroku vpravo

	M1A	M1B	M2A	M2B
Stav 1	0	0	1	1
Stav 2	0	1	1	0
Stav 3	1	1	0	0
Stav 4	1	0	0	1

Tabulka 6.2: Sekvence jednoho kroku vlevo

6.3 Výpočet rychlosti otáčení

U použitého krokového motoru bylo experimentálně zjištěno, že při nulovém zatížení jsou maximální otáčky přibližně 2000 ot/min, tyto otáčky jsou také maximem, které lze v aplikaci nastavit. Otáčky motoru byly vypočítány na základě vzorce 6.2 [11]. Vzorec vychází z velikosti úhlu, o který se hřídel otočí za jeden krok a frekvence střídání jednotlivých kroků. Úhel kroku je vypočítán z počtu kroků na jednu otáčku, tedy 360º / 4 = 72º. Frekvence je vypočítána z doby přetečení čítače/časovače 0 (t_0) a z počtu potřebných přerušení k provedení jedné změny stavu cívek (N) podle vzorce 6.1.

$$f_k = \frac{1}{t_0 \cdot 4 \cdot N} \ [Hz]$$

(Vzorec 6.1)

$$n = \frac{60 \cdot f_k \cdot \alpha}{360^{\circ}} \ [ot/\mathrm{min}]$$

(Vzorec 6.2)

Výpočet maximálních otáček, které lze v aplikaci nastavit**:**

$$f_k = \frac{1}{0{,}0015 \cdot 4 \cdot 1} = 166,\overline{6}\ Hz$$

$$n_{max} = \frac{60 \cdot 166,\overline{6} \cdot 72}{360^{\circ}} = 2000\ ot/min$$

Výpočet minimálních otáček, které lze v aplikaci nastavit**:**

$$f_k = \frac{1}{0{,}0015 \cdot 4 \cdot 100} = 1,\overline{6}\ Hz$$

$$n_{min} = \frac{60 \cdot 1,\overline{6} \cdot 72}{360^{\circ}} = 20\ ot/min$$

Minimální otáčky, které umožňuje aplikace nastavit, jsou podle výpočtu výše 20 ot/min. Protože při těchto nízkých otáčkách lze pozorovat jednotlivé otáčky samostatně, byl postup výpočtu vizuálně ověřen.

7 Závěr

Cílem bakalářské práce bylo seznámení se s již dříve realizovaným přípravkem pro výuku předmětů PMP a PHS na Technické univerzitě v Liberci. Dále vytvoření sady demonstračních aplikací pro tento přípravek a zrealizování hardwarového rozšíření.

Sada demonstračních aplikací byla vytvořena pomocí programovacího jazyka C se snahou o co nejefektivnější využití všech součástí přípravku. Zrealizované aplikace jsou digitální hodiny s obvodem RTC, řízení jasu žárovky pomocí pulzní šířkové modulace, hra hádání čísel a tester všech komponentů na přípravku. Protože aplikace ukazují, jak používat jednotlivé součásti přípravku, mohou se stát základem pro budoucí vývoj dalších aplikací na přípravku.

Všechny aplikace byly otestovány postupně na větším počtů přípravků. Zajímavostí bylo pozorování reakcí jednotlivých displejů. Přesto, že se technicky jedná o naprosto stejné typy, byl požadavek na nastavení časových zpoždění mezi jednotlivými zápisy na displej rozdílný. Tento rozdíl může být způsoben drobnými odlišnostmi v řadiči displeje jednotlivých výrobců.

Navržená hardwarová rozšíření mohou být v budoucnu rozpracována do realizovatelné podoby a využita pro další práce na přípravku. Hardwarové rozšíření, které umožňuje řízení krokového motoru, může být použito jako součást složitějších aplikací, nebo jako ukázka jedné z možností, jak přípravek rozšířit. K ovládání krokového motoru byla napsána demonstrační aplikace, která ukazuje, jak lze motor řídit. Při testování řízení krokového motoru bylo zajímavé pozorovat ztrátu momentu na hřídeli, v závislosti na otáčkách.

Seznam použité literatury

[1] Doc. Ing. SKALICKÝ, Petr, CSc. *Mikroprocesory řady 8051.* Praha 1997-1998.

[2] ŠALOUN, Petr. *Programovací jazyk C pro zelenáče.* Praha 2003

[3] Ing. MATOUŠEK, David. *C pro mikrokontroléry Atmel AT89S52.* Praha 2007.

[4] SDCC 3.0.0. *SDCC Compiler User Guide*. SDCC manual.pdf, 19. 10. 2010, [CD-ROM].

[5] Atmel. *Datasheet: AT89C51CC03*. AT89C51CC03.pdf, [CD-ROM].

[6] Prof. Ing. NOVÁK, Ondřej, CSc. Prof. Ing. NOUZA, Jan, CSc. Doc. Ing. DOLEŽAL, Ivan, CSc. Ing. KOLÁŘ, Milan, CSc. *Elektronika*. Liberec 2004.

[7] Hitachi. *Datasheet: HD44780U.* hd44780.pdf, [CD-ROM].

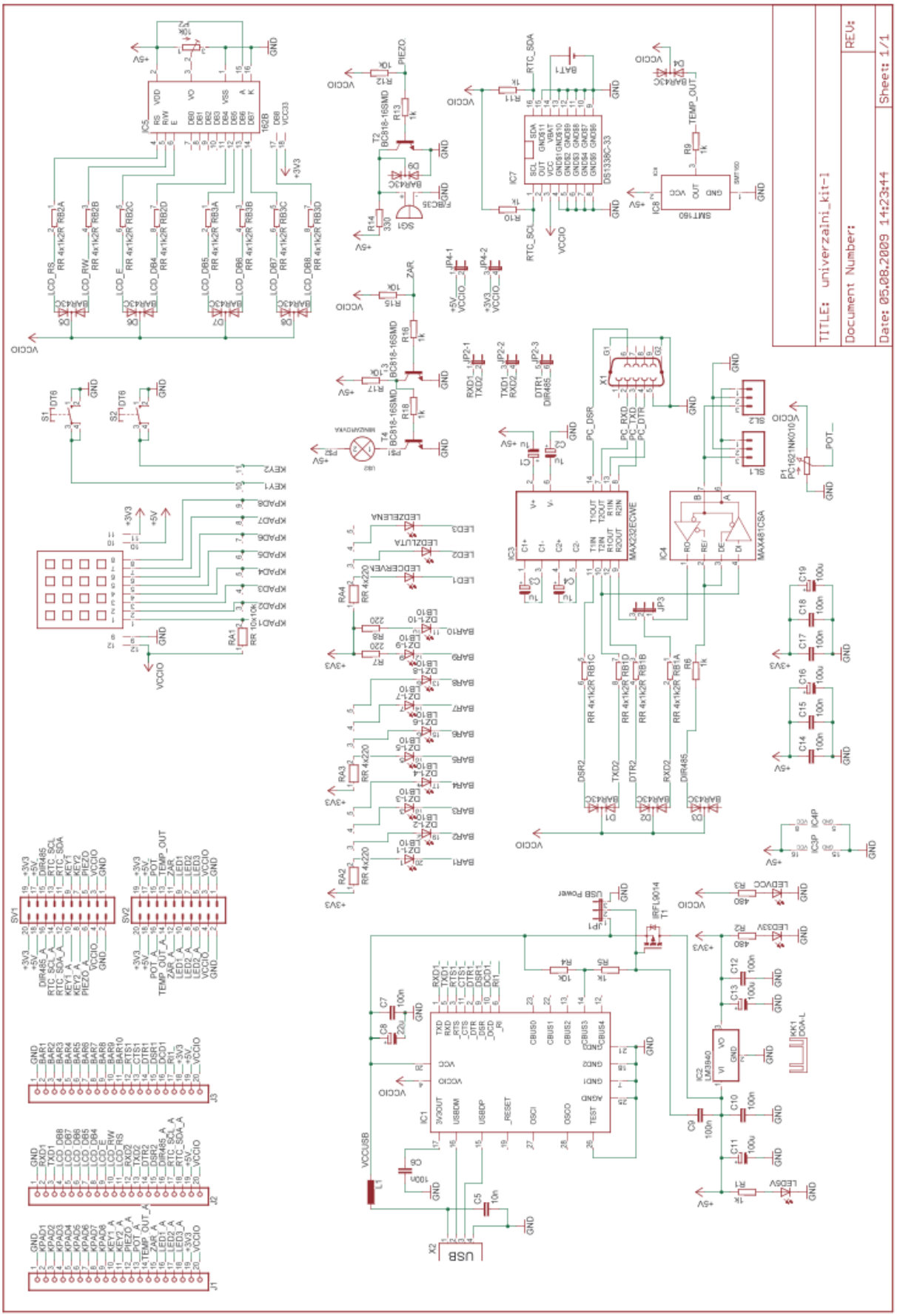
[8] Maxim. *Datasheet: DS1338C.* DS1338-DS1338Z.pdf, [CD-ROM].

[9] Philips Semiconductors. *I^2C Manual*. I2C.pdf. 2003, [CD-ROM].

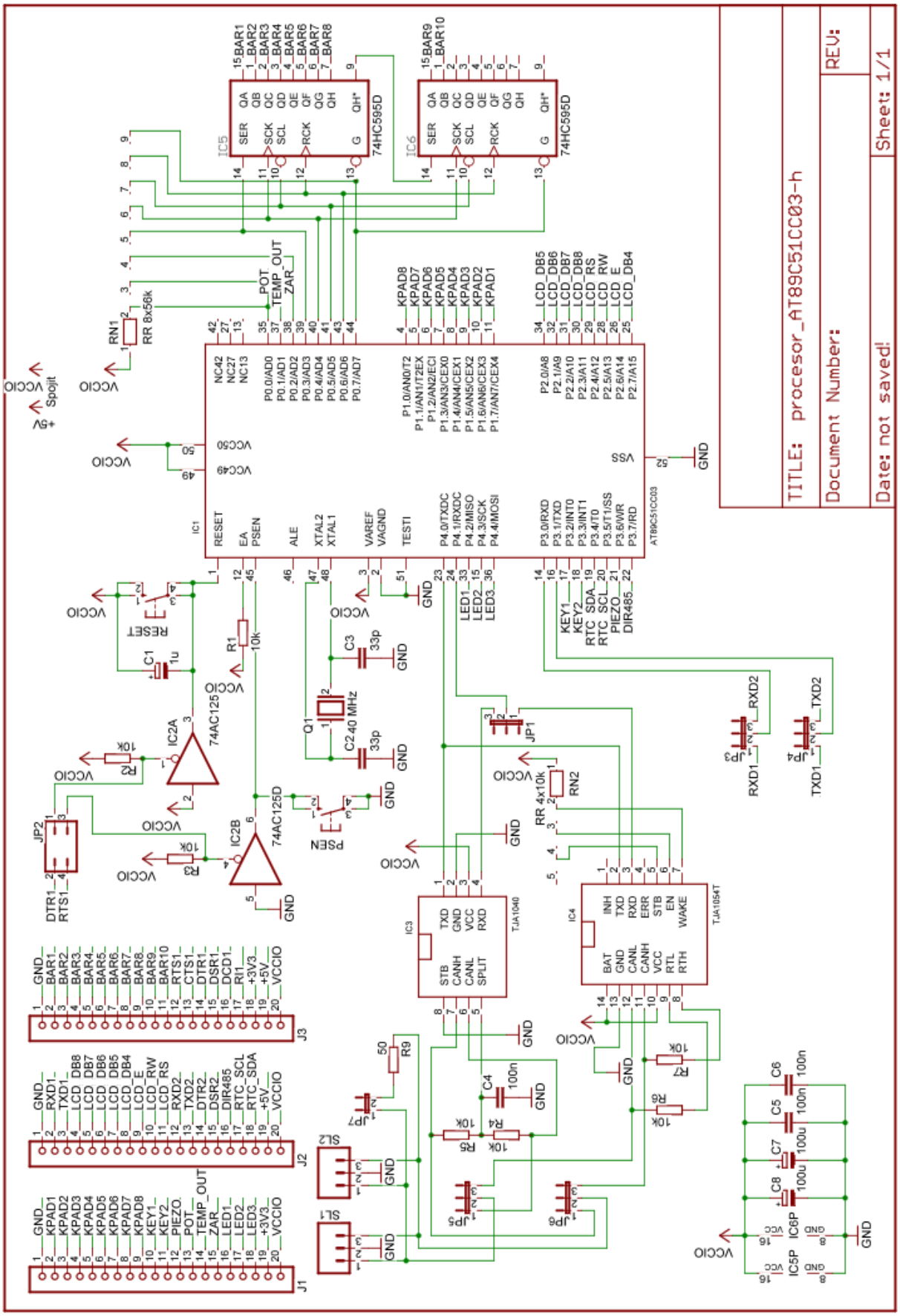
[10] Hy-Line. *Datasheet: SMT 160-30.* smt160.pdf, [CD-ROM].

[11] doc. Ing. Pavel Rydlo, Ph.D. *Krokové motory a jejich řízení.* Liberec 2000.

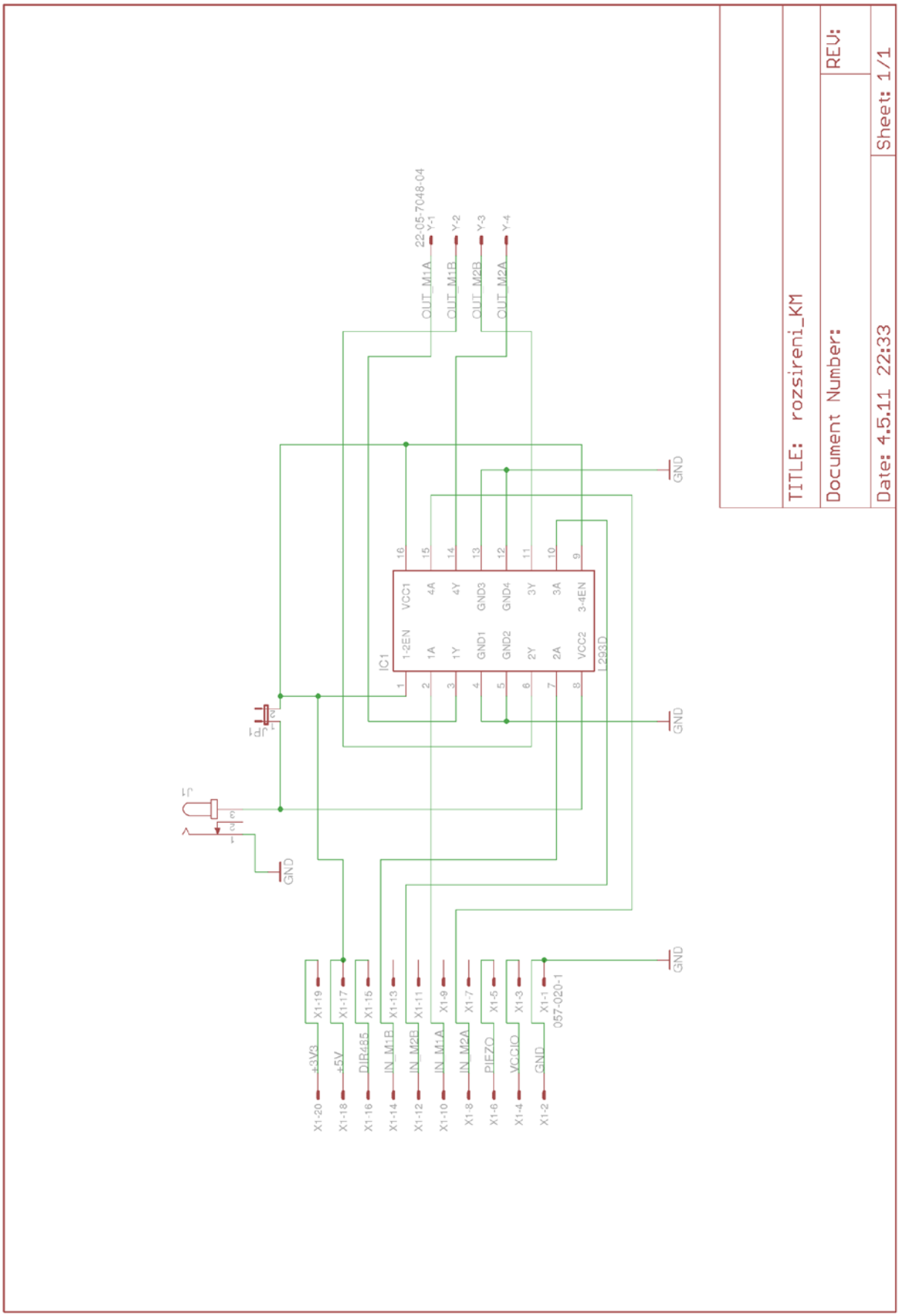
Příloha A – Schéma zapojení základní desky použitého přípravku



Příloha B – Schéma zapojení použité desky s mikroprocesorem



Příloha C – Schéma zapojení navrženého rozšíření



Příloha D – Návody k používání jednotlivých aplikací

Návod k aplikaci realizující digitální hodiny:

Základní obrazovka:

Základní obrazovka zobrazuje čas a datum. Písmeno „t“ indikuje stav minutky. Písmeno „a“ indikuje stav budíku. K přepínání mezi jednotlivými obrazovkami použijte klávesu A. Indikátory se zobrazují na všech obrazovkách

Nastavení času:

K nastavení času a data stiskněte klávesu B. Postupně zadejte celý čas a zadání potvrďte stiskem klávesy B. Dále zadejte celé datum a opět potvrďte klávesou B.

Nastavení data:

Pokud jsou zadány neplatné hodnoty, nastaví se automaticky nejvyšší možné.

Obrazovka minutky:

Pomocí klávesy A se přepněte na obrazovku minutky. Zde je zobrazen aktuální odpočet nastaveného času. Pro zastavení, nebo spuštění odpočtu stiskněte klávesu C.

Nastavení minutky:

Pro nastavení minutky se přepněte na obrazovku minutky a stiskněte klávesu B, dále postupujte stejně jako při nastavení času.

Obrazovka budíku:

Dalším stiskem klávesy A se přepněte na obrazovku budíku. Pro jeho zapnutí nebo vypnutí stiskněte klávesu C.

Nastavení budíku:

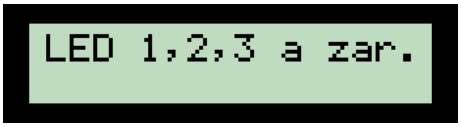
Pro nastavení budíku se přepněte na jeho obrazovku a stiskněte klávesu B, dále postupujte stejně jako při nastavení času nebo minutky.

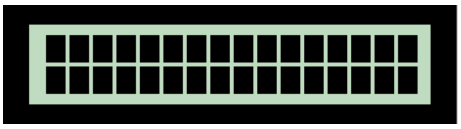
Návod k aplikaci pro testování přípravku:

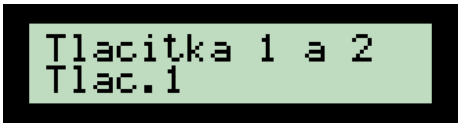
Základní obrazovka:

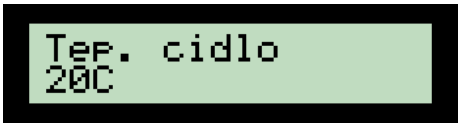
K testování jednotlivých součástí přípravku použijte klávesy 1 – 8. Na displeji se vždy zobrazí, která část je právě testována.

Obrazovky všech testovaných částí:









Seznam kláves:

1 – Test LED-diod a žárovky

2 – Test LED-baru

3 – Test LCD displeje

4 – Test potenciometru a A/D převodníku

5 – Test RTC obvodu

6 – Test piezo-měniče

7 – Test tlačítka 1 a 2

8 – Test teplotního čidla

Pro ukončení testování konkrétní součásti stiskněte libovolnou klávesu.

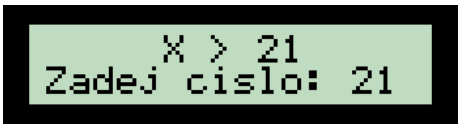
Pro otestování kláves, které nejsou použity pro ovládání, stiskněte postupně jednotlivé klávesy. Na displeji se vždy zobrazí informace, která klávesa je právě stisknuta.

Návod k aplikaci realizující hru hádání čísel:

Základní obrazovka:

Pro spuštění hry stiskněte libovolnou klávesu. Dále zadávejte čísla od 0 do 99. Zadané číslo musí být vždy vloženo jako dvouciferné. Čísla, menší než deset, musí tedy být uvozena nulou (např. 05). Na displeji bude zobrazována informace o tom, zda je hledané číslo větší, nebo menší než číslo tipované.

Obrazovka po zadání čísla:



Obrazovka po uhodnutí čísla:

Po uhodnutí čísla se na displeji zobrazí nápis „Uhodnuto!!“ a celkový počet pokusů, potřebných k uhodnutí.

Návod k aplikaci pro řízení jasu žárovky:

Základní obrazovka:

Pro změnu jasu žárovky použijte potenciometr. Aktuální hodnota je zobrazována na displeji.

Návod k aplikaci pro ovládání krokového motoru:

Základní obrazovka:

Pro otáčení motoru doprava stiskněte klávesu 1, pro otáčení motoru doleva stiskněte klávesu 3. K zastavení motoru použijte klávesu 2.

Obrazovka pro otáčení doprava:

Pro nastavení rychlosti otáčení použijte potenciometr.

Obrazovka pro otáčení doleva: