VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

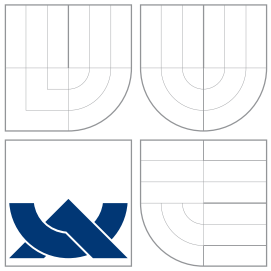
INTELIGENTNÍ PROGRAMOVATELNÉ RAZÍTKO NA BÁZI INKOUSTOVÉHO TISKU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

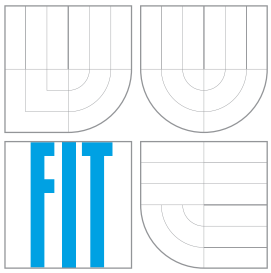
AUTOR PRÁCE
AUTHOR

Bc. ADAM CRHA

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

INTELIGENTNÍ PROGRAMOVATELNÉ RAZÍTKO NA BÁZI INKOUSTOVÉHO TISKU

INTELLIGENT PROGRAMMABLE STAMP BASED ON INKJET PRINT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ADAM CRHA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÁCLAV ŠIMEK

BRNO 2013

Abstrakt

Tato práce se zabývá návrhem a fyzickou realizací inteligentního, elektronicky programovatelného razítka na bázi inkoustového tisku. Razítko ve smyslu malé přenosné tiskárny umožní tisknout libovolný jednoduchý text a je schopno nahradit běžná kancelařská razítka za cílem redukovat náklady na pořízení běžných razítek a jejich počet. V práci je zkoumána technologie inkoustového tisku a principy řízení tiskových hlav. Na základě zjištěných poznatků je navržený koncept. Důležitou součástí práce je fyzická realizace prototypu, který je vytvořen na základě konceptu.

Abstract

This thesis deals with a concept and physical prototype of an intelligent, electronically programmable stamp, based on inkjet print. The stamp is basically a small inkjet printer. The stamp can print a simple custom text and is meant to replace regular office stamps. The benefit of the proposed stamp should include cost reduction and need for multiple stamps. The theoretical concept is followed by a prototype, which is an essential part of this work.

Klíčová slova

Vestavěný systém, razítko, tisk, tisková hlava, mikrokontrolér, enkodér, senzor, hardware, firmware, software.

Keywords

Embedded system, stamp, print, cartridge, microcontroller, encoder, sensor, hardware, firmware, software.

Citace

Adam Crha: Inteligentní programovatelné razítko na bázi inkoustového tisku, diplomová práce, Brno, FIT VUT v Brně, 2013

Inteligentní programovatelné razítko na bázi inkoustového tisku

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně pod vedením pana Ing. Václava Šimka.

.....
Adam Crha
14. května 2013

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Václavu Šimkovi za jeho čas, který věnoval a poskytl odbornou pomoc při řešení této práce.

Obsah

Kapitola 1

Úvod

Obyčejné inkoustové razítko najdeme zcela jistě v každé kanceláři. Ve většině případů je potřeba mnoho různých razítek, což je nepraktické na skladování. V případě změny údajů je nezbytné objednat nové razítko. Tím rostou finanční náklady a časové prodlevy spojené s objednávkou a pořízením. Časová razítka jsou také velmi nepraktická, zapomene-li uživatel změnit datum pomocí nastavovacích koleček. Řešením těchto problémů a nedostatků by mohlo být inteligentní programovatelné razítko[1], dále IPR. IPR je speciální zařízení založené na myšlence vestavěného systému. Jednoúčelová aplikace, která řeší nedostatky již stávajících možností. Inteligentní progamovatelné razítko obsahující základní části jako mikrokontrolér a tiskovou hlavu, by mělo nahradit běžné statické razítko. Díky implementované inteligenci a možnostem mikrokontroléru je umožněno do razítka nahrát více vzorů textů, více písem a v případě využití RTC[2] modulu aktuální datum a čas. Stiskem příslušného tlačítka se zahájí tisk aktuálně zvoleného textu. Pohybem IPR po papíře se postupně tiskne text.

Možnosti použití takového razítka je mnoho - tisk podpisů, iniciálů, tisk výrobních čísel na produkty a balíky, potisk výrobních informací na vajíčka, tisk iniciálů majitele na jednotlivé stránky tištěné knihy, tisk čárových kódů a mnoho dalších aplikací.

Tato práce se zabývá návrhem a realizací inteligentního programovatelného razítka s funkcí nastíněnou v předchozím odstavci. Cílem je navrhnout zařízení IPR, které bude schopné nahradit dnes běžná razítka. Výsledkem práce bude fyzicky realizované zařízení schopné ručního tisku.

Obsahem práce je kompletní návrh koncepce, konstrukce a hardwaru, po kterém následuje fyzická realizace vycházející z návrhu. Po zkonstruování razítka je nutné provést testování, které je jednou z nejdůležitějších fází celé práce.

Práce je členěna do několika kapitol. První čtyři teoreticky zaměřené kapitoly se věnují principu inkoustového tisku, principům a možnostem optického a mechanického snímání pohybu, přehledu dostupných mikrokontrolérů na trhu a možnostem jejich programování a nakonec konceptuálnímu návrhu zařízení. Další kapitoly zmiňují hardwarovou realizaci, implementaci firmwaru IPR a nakonec testování prototypu, možnosti rozšíření a závěr.

[1] IPR - Inteligentní programovatelné razítko - Pracovní název v rámci projektu a vývoje

[2] RTC - Real Time Clock.

Kapitola 2

Principy inkoustového tisku

Tato kapitola obsahuje stručný, ale sjednocený přehled principů inkoustového tisku. Diskutuje základní vlastnosti a typy inkoustových tiskových hlav. Informace o principech a technologiích byly čerpány ze zdrojů [24],[28].

2.1 Základní vlastnosti

Inkoustový tisk během posledních patnácti let prošel obrovským rozvojem a stal se velmi rozšířenou a oblíbenou technologií nejen v domácnostech. Kvalita tisku dosahuje nejlepších výsledků ve srovnání s jinými technologickými principy. Inkoustové technologie nacházejí uplatnění jak v domácnostech tak i ve velkoformátových systémech. Je nutné zmínit také použití inkoustového tisku v průmyslu.

Existuje několik principů inkoustového tisku. Je však důležité upozornit na to, že všechny principy mají jedno společné. Princip tisku je založen na vytváření velmi malých kapiček inkoustu, typicky o objemu 1pl (pikolitr = 10^{-12}). Tyto inkoustové kapky jsou vymršťovány velkou rychlostí, přibližně 50 - 100 kilometrů za hodinu. Technologické principy vymršťování kapek jsou už rozdílné a závislé na použité metodě/technologii.

Následující výčet stručně shrnuje vlastnosti inkoustového tisku:

Výhody inkoustového tisku:

- tichý provoz
- velké rozlišení
- kvalitní fotografický tisk
- barevný tisk
- nízká pořizovací cena

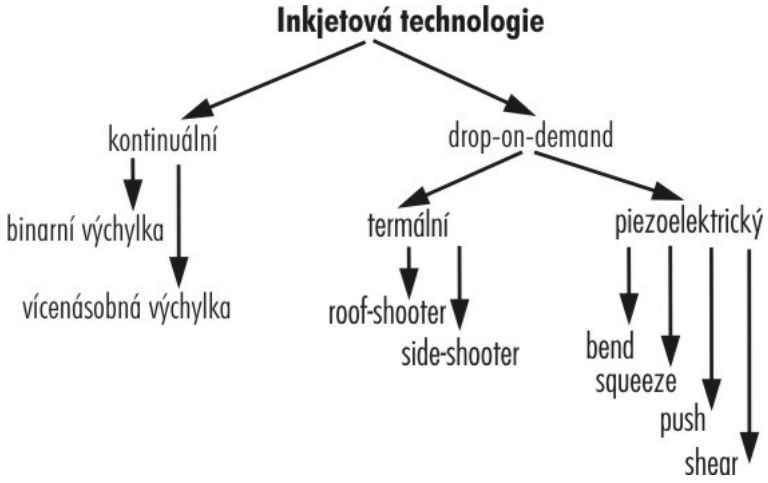
Nevýhody inkoustového tisku:

- Inkoust je velmi drahý
- trysky se často ucpávají zaschlým inkoustem
- inkoustový potisk je rozpustný ve vodě
- inkoust po čase vybledne

2.2 Rozdělení

Technologické postupy a principy inkoustového tisku prošly jistým vývojem a dnes stále probíhá široký rozvoj inkoustového tisku. V současnosti jsou nejrozšířenější tři přístupy a to kontinuální tisk, termální tisk a tisk založený na piezoelektrickém jevu. Termální a piezoelektrický princip jsou si velmi podobné a spadají do takzvaného „on-demand“ tisku. Kontinuální systém se od předchozích dvou systémů velmi liší. Jednotlivé systémy si popíšeme v následujících podkapitolách.

Obrázek 2.1 graficky znázorňuje stromové dělení jednotlivých technologií.



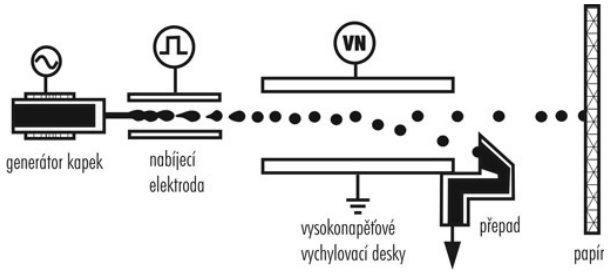
Obrázek 2.1: Technologická mapa [24].

2.2.1 Kontinuální systém

Historicky nejstarší princip inkoustového tisku se nazývá kontinuální systém. Z inkoustového zásobníku je inkoust přiváděn do generátoru kapek, ze kterého je neustále vystřikován inkoust mezi pár elektrod, které jsou nabity elektrický nábojem. Na kapky se přenese část náboje v době průletu mezi elektrodami. V případě systému založeném na binární výchylce kapky putují dále mezi dvě deflekční elektrody. Kapka nabitá nábojem je vlivem elektrického pole vychylována ze svého směru. Nenabité kapky pokračují ve své dráze a dopadají na papír, zatímco nabité kapky se vychýlí do odpadního kanálku a inkoust je poté zrecyklován do zásobníku. Velmi zjednodušeně lze tento systém přirovnat ke kanónu, který neustále vystřeluje koule. Chceme-li, aby koule zasáhla cíl, nesmíme jí ovlivnit. Nechceme-li cíl zasáhnout, můžeme kouli vychýlit pomocí větru do vody. Na obrázku 2.2 je znázorněn princip kontinuálního systému inkoustového tisku.

Vylepšená varianta kontinuálního systému vychylování kapek se nazývá „multiple deflection“ tedy víceúrovňové vychylování kapek. Nenabité kapičky jsou odvedeny do zásobníku inkoustu a jsou zrecyklovány. Kapičky, v tomto vylepšeném systému, je možné nabít na různou velikost náboje. Málo nabité kapičky jsou vychýleny méně, více nabité kapičky jsou vychýleny více. Tím je do jisté míry možné ovlivnit cíl dopadu kapky na potiskovaném médiu.

Oba tyto systémy jsou velmi rychlé, ale velmi složité a drahé z důvodů recyklace inkoustu. Proto se spíše používají v průmyslových a velkoformátových tiskárnách.



Obrázek 2.2: Princip kontinuálního tisku [24].

2.2.2 On-demand systém

V této podkapitole je vysvětlen princip systémů „on-demand“. Tyto systémy jsou v současnosti velmi levné, jednoduché a extrémně rozšířené. Princip je velmi jednoduchý. Jednotlivé kapičky inkoustu jsou z tiskové hlavy vystřikovány jen tehdy, pokud mají dopadnout na potiskované médium. Každé vystříknuté kapce odpovídá jeden konkrétní tiskový bod. Tento systém se dělí na další dva základní podsystémy, které se liší použitou technologií. První ze dvou technologií je nazývána jako „termální systém“. Druhou velmi rozšířenou technologií je „piezoelektrický systém“, který využívá piezoelektrického jevu. Obě tyto technologie budou detailněji popsány dále v této podkapitole.

Piezoelektrický systém

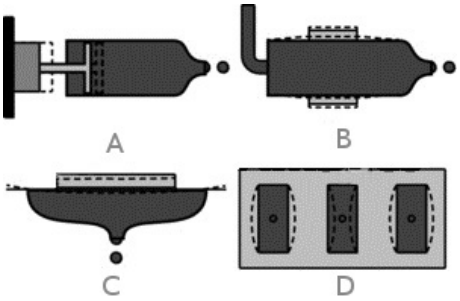
Piezoelektrický jev je úkaz, vyskytující se u monokrystalických látek, jehož schopnost je generovat elektrické napětí při deformaci látky a naopak, kdy se látka vlivem elektrického napětí deformuje.

Piezoelektrické systémy využívají piezoelektrického jevu a hydrodynamického tlaku pro generování jednotlivých kapek. Komůrka pro preparaci inkoustu před vystříknutím je tvořena pružnou membránou, která je mechanicky spojena s piezoelektrickým materiálem. Přiložením napětí na piezoelektrický materiál se membrána začne deformovat. Deformací komůrky se mění objem a tím je inkoust vystříknut na potiskované médium. Podle způsobu pohybu a deformace membrány se piezoelektrické systémy rozdělují na další 4 podsystémy:

- a) Push (pístový mechanizmus)
- b) Squeeze (škrtící mechanizmus)
- c) Bend (prohýbací mechanizmus)
- d) Shear (stříhací mechanizmus)

Na obrázku 2.3 jsou znázorněny jednotlivé piezo systémy. Systém *a) Push* funguje na myšlence pístu, který je stlačován piezoelektrickým prvkem. V systému *b) Squeeze* je komůrka kolem dokola obalená piezomateriálem. Při přiložení napětí se komůrka „zaškrtí“. Systém *c) Bend* má přiložený piezoprvek ke komůrce, při přiložení napětí se vyboulí a zmenší objem komůrky. Systém *d) Shear* má přiloženy na bocích komůrky dva piezoprvky, které sevřou komůrku z obou stran. Princip je podobný lisu.

Piezoelektrické systémy jsou schopné „zpětného chodu“. To znamená, že inkoust může být nasán zpět do trysky. Díky tomuto efektu je také možné eliminovat vznik nežádoucích



Obrázek 2.3: Principy piezoelektrického tisku [24].

mikrokapiček právě nasátím zpět do trysky okamžitě potom, co vymrštěná kapka opustí trysku.

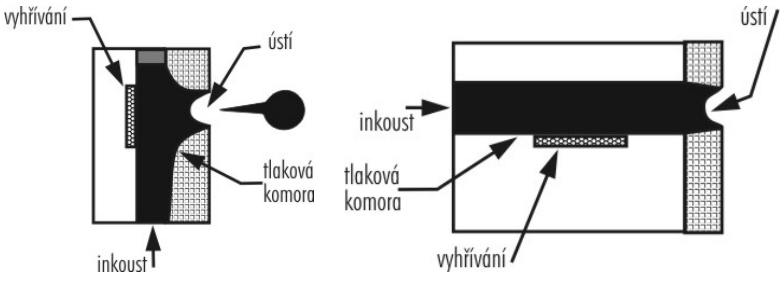
Termální systém

Termální, neboli tepelné systémy pro generování kapek jsou dnes nejrozšířenějším systémem. Technologie využívá pro vytrysknutí inkoustu na potiskované médium teplo. Termální systém taktéž využívá komůrky pro uchování inkoustu stejně jako piezoelektrický systém. Komůrky se však nedeformují. V bezprostřední blízkosti komůrky se nachází topné tělísko, které zahřívá kapalinu, inkoust.

Termální systém má tři základní fáze procesu. V první fázi je komůrka s inkoustem zahřívána. Zahříváním se začne vytvářet bublina, která zvyšuje tlak v komůrce. V druhé fázi tlak přesáhne určitou mez a obsah komůrky, inkoust, je vymrštěn na papír. Při ochlazování komůrky se bublina zmenšuje a tím vzniká podtlak. V poslední fázi se vlivem podtlaku nasaje inkoust ze zásobníku a tryska je připravena k dalšímu tisku [27].

Termální technologie nese jistá omezení v tisku nutností používat inkoust odolný teplu. Používání tepla také vyžaduje potřebu chlazení, proto je potřebné obětovat čas pro zchladnutí topného tělíska.

Termální systém má základní dva podsystémy. Jejich rozdělení je odvozeno z umístění topného tělíska vůči komůrce. Roof-shooter se nazývá systém, který má topné tělísko naproti trysce. Side-shooter se nazývá systém, který má topné tělísko na boční stěně komůrky. Jednoduché náčrty dvou základních podsystémů jsou znázorněny na obrázku 2.4.



Obrázek 2.4: roof-shooter (vlevo) [23], side-shooter (vpravo)[23].

Kapitola 3

Principy snímání polohy

K tomu, aby tisk inteligentním programovatelným razítkem nebyl zdeformovaný v závislosti na rychlosti pohybu, je vhodné, aby byla sledována změna pozice razítka. I běžné inkoustové tiskárny mají systémy pro určení přesné polohy tiskové hlavy. Tím je zajištěna maximální přesnost a rychlost tisku. U inteligentního razítka bude vhodné sledovat, jakou rychlostí uživatel razítkem pohybuje a adaptovat tak rychlost tisku. Bez použití tohoto rozšíření by bylo nutné, aby se uživatel razítka naučil pohybovat razítkem přesnou konstantní rychlostí, aby byl text proporčně stejný a nedeformovaný. V případě, že by uživatel pohyboval razítkem pomalu, text by byl velmi nahuštěný a naopak, kdyby uživatel pohyboval razítkem příliš rychle, text by byl velmi roztažený. Použitím vhodné technologie pro snímání pohybu by se tomuto nepříznivému vlivu dalo, alespoň částečně zabránit.

Následující kapitola se věnuje stručné rešerši použitelných metod pro snímání změn pohybu.

3.1 Optické kamerové snímače

S optickým kamerovým snímačem přicházíme často do styku, aniž bychom si to uvědomovali. Takové snímače se nacházejí v běžných počítačových myších pro ovládání kurzoru. Již před lety byly mechanické kuličkové myši nahrazeny myšmi optickými. Tento stav zapříčinil obrovský pokles ceny relativně složitých zařízení. Optický senzor obsahuje kameru, většinou o rozlišení 16x16 pixelů, logiku pro zpracování obrazu a v neposlední řadě komunikační rozhraní se svým okolím.

3.1.1 Princip optického snímače

Jak již bylo zmíněno v úvodu této sekce, optické kamerové snímače se nacházejí v každé moderní počítačové myši. Jak takové snímání funguje? Princip je popsán dále.

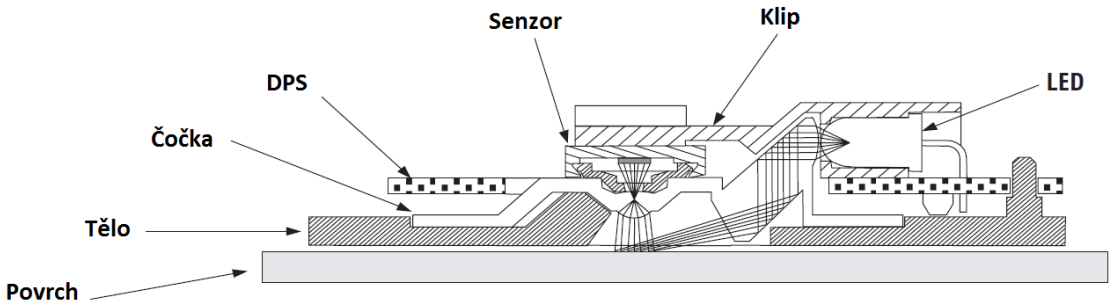
K optickému snímání je nutné několik dílů a součástek. Zcela určitě integrovaný obvod, který zajišťuje snímání (CCD - Charge-coupled device) a komunikaci s rozhraním.

Další součástkou je LED dioda, která osvětluje povrch podložky a zvýrazňuje tak texturu povrchu, která je nutná k detekci pohybu.

Světlo emitované LED diodou je rozptýlené, proto je nutné světlo usměrnit. K tomu slouží další nezbytná součástka - zrcátko, nebo světelný hranol, který emitované světlo nasměruje přímo na podložku.

Protože se jedná o velmi jednoduchou kameru o rozlišení v řádu desítek pixelů, je důležité snímaný obraz transformovat do malého okénka CCD snímače. K tomu poslouží poslední součástka - čočka.

K optickému snímání jsou tedy důležité čtyři součástky, které spolu musí být kompatibilní. Pokud toto nebude zajištěno, optické snímání nebude fungovat.



Obrázek 3.1: Princip snímání optickým kamerovým snímačem [9].

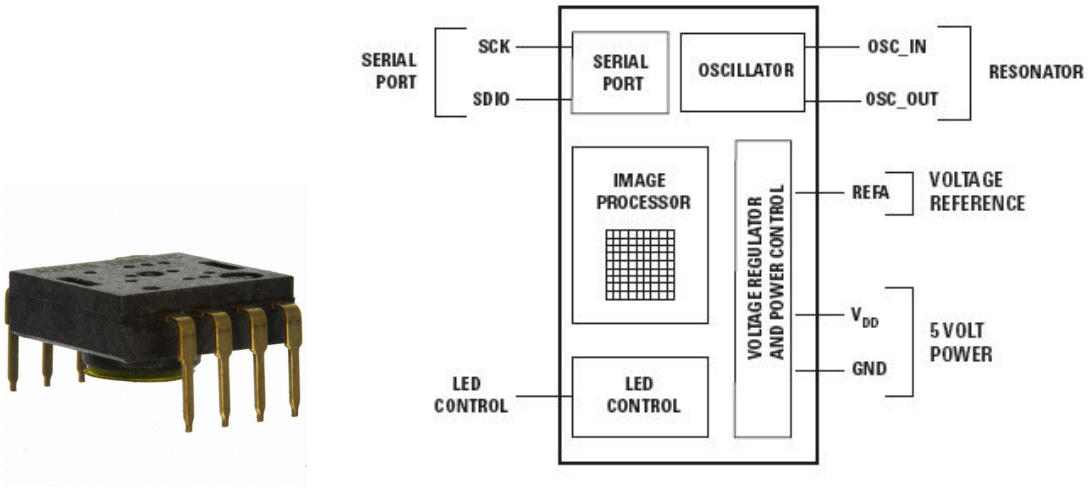
Funkce a princip snímání (viz obrázek 3.1): LED diodou je emitováno světlo do hranolu, nebo zrcátka. Světlo je lámáno pod určitým úhlem a směrováno na snímaný povrch. Používá se červená LED dioda a to z těchto důvodů: červené diody jsou jedny z nejlevnějších, vyžadují menší proud elektrické energie oproti modré nebo bílé diodě a údajně mají pozitivní vliv na zvýšení kontrastu reliéfu podložky [33].

Osvícená podložka je snímána pomocí CCD snímače. Snímač sejme obrázek typicky o velikosti 16x16 pixelů a uloží do paměti. Sejme se další snímek a pomocí algoritmu implementovaného v integrovaném obvodu se vyhodnotí rozdíly mezi dvěma snímky. Na základě těchto rozdílů se dá rekonstruovat směr pohybu. Vypočítaná diference posunu obrazu je zaznamenána do registru. Z toho registru je možné číst přes komunikační rozhraní obvodu USART, SPI nebo dokonce USB [33].

3.1.2 Nejjednodušší optický senzor

Nejjednodušší dostupný optický senzor je vyráběn firmou Avago Technologies [6] a je označen typovým číslem ADNS2610. Existuje také téměř identický senzor od firmy Pixart s označením PAN3101. Oba senzory mají stejné pouzdro, stejný počet vývodů i stejný způsob komunikace, ovšem PAN3101 disponuje širší nabídkou stavových a řídících registrů, například registr informující o detekci pohybu.

Oba dva zástupci velmi jednoduchých optických snímačů disponují pouze dvouvodičovou sériovou linkou pro komunikaci s okolím. Jeden vodič je využit pro data, druhý pro přenos obdélníkového signálu, coby hodiny. Komunikace se senzorem je velmi triviální a je rozebrána v dalších kapitolách. Na obrázku 3.2 je možné vidět fotografii senzoru a jeho blokové schéma.



Obrázek 3.2: Vlevo senzor ADNS2610 [6], vpravo blokové schéma senzoru [6].

3.2 Enkodéry

Tato podkapitola popisuje mechanické snímače pohybu - enkodéry. Enkodér je speciální typ elektronického zařízení, tzv. inteligentní senzor, jehož vstupem je mechanický pohyb a výstupem je elektronický signál. Enkodéry se dnes používají zejména v robotice a ve většině aplikací, kde jsou využity pohony a kde je potřeba znát aktuální polohu natočení hřídele motoru, měřit úhlovou rychlost otáčení nebo zrychlení, aby bylo zajištěno přesné řízení dané aplikace. Jsou-li jako pohon použity krokové motory, poloha a natočení hřídele motoru je známá z principu řízení. U motorů s komutátorem či střídavých motorů je nutné zajistit sledování polohy hřídele jiným způsobem - pomocí zpětné vazby. Motor se řídí nepřesně a poloha hřídele se určuje zpětně, tzn. zpětně se vyhodnocuje o kolik nebo kam se motor otočil [30].

Principů snímání polohy je několik a jednotlivé enkodéry budou popsány v následujících podkapitolách.

Tato podkapitola čerpá informace a obrázky zejména z literatury [30].

Enkodéry mohou být klasifikovány do následujících tří kategorií[1]:

Dle konstrukce:

- Rotační (motory - natočení, pootočení hřídele).
- Posuvné (tiskárny - pozice tiskové hlavy vůči papíru).

Dle principu snímání:

- Enkodéry s optickou závorou.
- Magneticky citlivé enkodéry.

Dle výstupní hodnoty:

[1] Klasifikace je na základě uvážení autora.

- Absolutní enkodéry.
- Inkrementální enkodéry.

Dalším důležitým parametrem je rozlišení - nejmenší velikost jednoho inkrementu na celém intervalu. Volba správného enkodéru se odvíjí od požadavků a použití pro konkrétní aplikaci.

3.2.1 Optické vs. magnetické enkodéry

Kapitola popisuje rozdíl mezi těmito dvěma principy snímání.

Optické enkodéry

Optické enkodéry jsou jedny z nejrozšířenějších enkodérů. Abychom dobře pochopili princip převodníku, vyjmenujeme si základní součásti:

- Kódové kolečko s posloupnosti průhledných a neprůhledných míst nebo odrazných a neodrazných ploch.
- Zdroj světla - typicky LED dioda.
- Přijímač světla - typicky fototranzistor nebo fotodioda.
- Volitelně optika pro zaostření světelného paprsku emitovaného z LED diody.

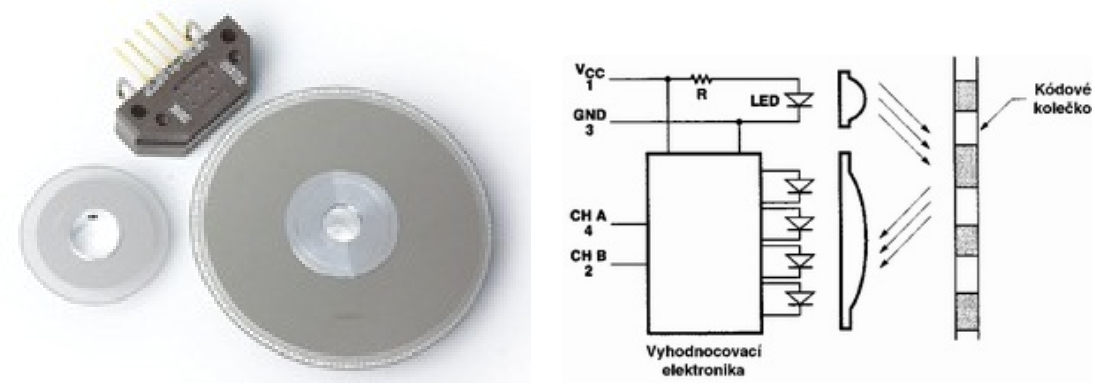
Princip optického enkodéru:

Kódové kolečko je pevně spojeno s hřídelí motoru. Kódové kolečko se tak současně otáčí s hřídelí motoru. Je využíváno takzvaného principu optické závory. Světlo emitované světelným zdrojem prochází skrz průhledná místa v kódovém kolečku, zatímco přes neprůhledná místa je světlo pohlcováno a neprochází tak skrz kódové kolečko. Otáčení hřídelí a současně tak kódovým kolečkem generujeme světelné pulsy, které dopadají na detektor světla. Detektor se stane vodivým, dopadá-li na něj světlo. Je-li detektor zastíněn, elektrický proud nevede. Tím je možné převádět optický signál na elektrický a generovat tak sekvenci elektrických impulsů v závislosti na otáčení hřídelí.

Takto funguje optický rotační enkodér. Posuvný optický enkodér pracuje prakticky stejně, místo kódového kolečka je použit děrný pásek.

Na obrázku 3.3 vlevo je optický enkodér od firmy Avago Technologies a vpravo je zobrazen princip optické závory.

Výhodou těchto enkodérů je jednoduchost a výroba je možná v prakticky amatérských podmínkách, není-li podmínkou vysoké rozlišení. Jako nevýhoda se může jevit princip založený na světle, kdy při přímém nevlastním osvětlení může být enkodér rušen. Rušení může také způsobit znečištění optiky zdroje nebo přijímače světla.



Obrázek 3.3: Enkodér od Avago tech.(vlevo) [6], princip optické závory (vpravo) [30].

Magnetické enkodéry

Magnetické enkodéry jsou principiálně naprosto odlišné snímače od optických enkodérů. Enkodéry využívají Hallova jevu, pracují tedy s magnetickým pólem a indukčním tokem. Hallův jev je proces generování elektrického pole v polovodiči (možno i v kovech, nicméně kovy obsahují vysoké množství vodivostních elektronů, a proto se příliš neuplatňují), za současného působení vnějšího magnetického i elektrického pole. Důsledkem toho se hromadí na jedné straně polovodiče/kovu kladný náboj a na druhé straně záporný. Vzhledem k tomu, že tímto procesem se na obou stranách vytvoří různý potenciál, je po přiložení svorek voltmetru na obě strany možné naměřit Hallovo napětí [32].

Hallova sonda nebo také Hallův článek je elektronická součástka využívající tzv. Hallova jevu. Jedná se o součástku, která se používá pro měření magnetického pole, jak již bylo zmíněno v předchozím odstavci o Hallově jevu.

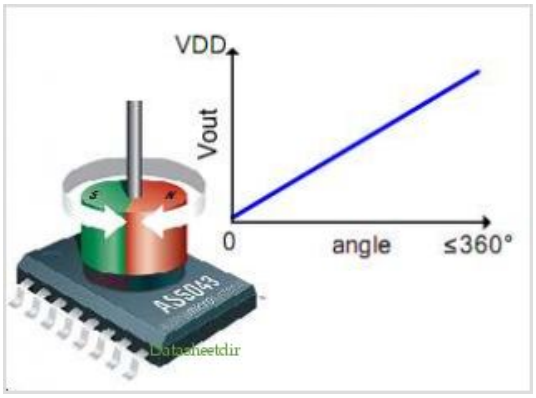
Magnetické enkodéry se skládají ze dvou částí. Z integrovaného obvodu, který obsahuje jednu či několik Hallových sond a vyhodnocovací logiku. Kolmo proti integrovanému obvodu je umístěn permanentní magnet se dvěma póly, severním a jižním. Magnet je pevně připevněn k hřídeli motoru. Otáčí-li se hřídel motoru, otáčí se současně póly magnetu.

Správnému natočení magnetu vůči Hallově sondě je na sondě indukováno Hallovo napětí. Napětí na konkrétní Hallově sondě je přímo úměrné úhlu natočení magnetu. Pomocí analog-digitálního převodníku je vyhodnocena úroveň napětí a tím i zjištěna poloha natočení hřídele. Rozlišení analog-digitálního převodníku určuje rozlišení enkodéru. Po zpracování úrovně napětí je informace distribuována dále přes vhodný komunikační kanál. Princip magnetického enkodéru je možné spatřit na obrázku 3.4.

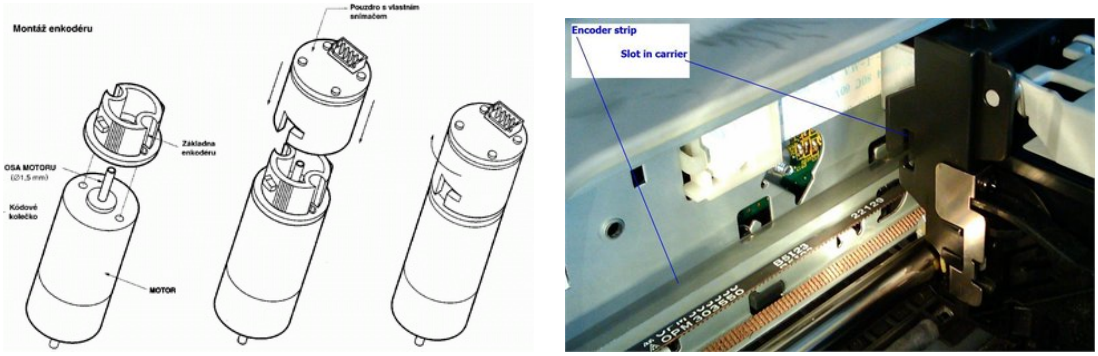
Magnetické enkodéry se zejména používají v chemických a prašných prostředích, v prostředích s vysokou teplotou a jiných nebezpečných zónách, protože poskytují možnost bezkontaktního snímání a elektrické části tak mohou být dobře zaizolovány.

3.2.2 Rotační vs. posuvné enkodéry

Kapitola popisuje rozdíly mezi těmito dvěma typy konstrukce.



Obrázek 3.4: Magnetický enkodér [5].



Obrázek 3.5: Rotační enkodér (vlevo), posuvný enkodér (vpravo) [30].

Rotační enkodéry

Rotační enkodér převádí otáčivý pohyb na elektronický signál. Tyto enkodéry mají většinou na hřídeli součástku, která jasně znázorňuje orientaci hřídele. Takovou součástkou může být kódové kolečko (viz. podsekce 3.2.1), nebo magnet se severním a jižním pólem (viz. podsekce 3.2.1). Další součástkou je zdroj přímého světla a jeho detektor, nebo magneticky citlivý integrovaný obvod.

Použití rotačních enkodéru není omezeno jen na robotiku, můžeme je nalézt v dnes již historických kuličkových myších nebo ve spalovacím motoru dnešního automobilu pro měření otáček.

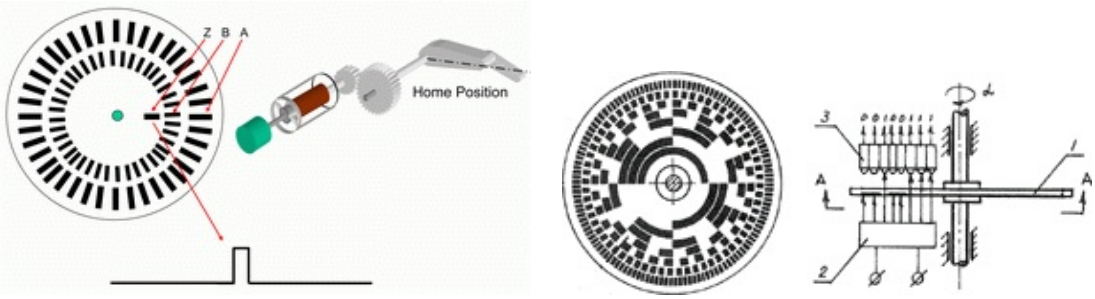
Na obrázku 3.5 vlevo je ukázka aplikace enkodéru připevněného k elektromotoru.

Posuvné enkodéry

Posuvné enkodéry převádí posuvný pohyb na elektrický signál. Tyto enkodéry se používají zejména v tiskárnách a jim podobných aplikacích. Většinou pracují na principu optické závory (viz. podsekce 3.2.1) a děrného pásku. Staticky umístěná optická závora generuje signál v závislosti na pohybu děrného pásku, který se pohybuje současně s tiskovou hlavou. Takto lze jednoduše zjistit polohu tiskové hlavy. Na obrázku 3.5 vpravo je ukázka použití posuvného enkodéru v tiskárně.

3.2.3 Inkrementální vs. absolutní enkodéry

Tato kapitola popisuje rozdíly mezi dvěma možnými výstupy, které enkodéry mohou poskytovat.



Obrázek 3.6: Inkrementální (vlevo), absolutní (vpravo)[30].

Inkrementální enkodéry

Jsou snímače, které poskytují relativní informaci o změně polohy. Výstupem jsou dva fázově posunuty signály impulsů, pomocí kterých je možné sledovat změnu v rotaci a směru pohybu. Základní charakteristikou těchto enkodérů je především rozlišení, tedy počet pulsů na otáčku. Kvalitní enkodéry dosahují až 10 000 pulsů na otáčku. Tyto dva fázově posunuté signály mohou být doplněny ještě o tzv. indexový kanál, který je aktivní pouze jedenkrát za otáčku ve výchozí poloze. Otáčením hřídele tak získáváme obdélníkový signál. Použitím inkrementálních enkodérů však nezjistíme absolutní polohu hřídele, pokud nebude neustále zaznamenáván počet pulsů. Tento nedostatek však řeší absolutní enkodéry.

Absolutní enkodéry

Absolutní enkodéry jsou většinou rotační snímače polohy natočení hřídele, které poskytují aktuální informaci o současné poloze díky binární kombinaci několika signálů z kódového disku. Tato informace je statická, takže je dostupná i po výpadku napájení. Kódové kolečko obsahuje speciální binární kombinaci děr, nebo odrazových ploch ke kódování aktuální pozice. Většinou se využívá grayova, nebo binárního kódu. Vpravo na obrázku 3.6 je znázorněno kódové kolečko s binárním kódem.

Kapitola 4

Dostupné platformy a vývojové nástroje

Jak již bylo popsáno v úvodu, razítko se bude svými vlastnostmi řadit do vestavěných systémů, což jsou jednoúčelové systémy, ve kterých je řídící počítač zcela zabudován do zařízení. Jednoduše se dá vestavěný systém definovat jako počítačové zařízení, o kterém uživatel netuší, že pracuje s počítačem.

Řídící počítač zastává většinou nějaký jednoduchý mikrokontrolér, známý také jako jednočipový počítač. Mikrokontrolér je polovodičová programovatelná součástka obsahující kompletní mikropočítač.

Na trhu dominují 4 největší výrobci mikrokontrolérů. Patří sem firmy Freescale, Atmel, Microchip a ST microelectronic.

4.1 8bitové a 32bitové mikrokontroléry

Mikrokontroléry lze rozlišovat na základě délky operandu v bitech. Základní vlastností mikrokontrolérů, ale i procesorů obecně, je délka slova (operandu). Délka slova značí s kolika bity dokáže procesor pracovat v rámci jednoho kroku. Zjednodušeně lze říci, že např. 8bitový mikrokontrolér umí přímo počítat s čísly 0 až 255, 16bitový s čísly 0 až 65535 atd. Operace s většími čísly musí být provedeny v několika krocích. Tím je dána rychlost zpracování instrukcí [34].

Pro jednoduché aplikace se spíše využívají 8 nebo 16bitové mikrokontroléry pro jejich jednoduchost a dostatek výkonu. Pro složitější zařízení jako jsou mobilní telefony, tablety a PDA se využívají spíše 32bitové mikrokontroléry, typicky ARM[1].

Protože konstruované zařízení nebude obecně výpočetní stroj, ale jednoúčelová jednoduchá aplikace, bude použito 8bitového mikrokontroléru.

4.1.1 Mikrokontroléry Freescale

Freescale Semiconductor, Inc. je americký výrobce polovodičových součástek, který vznikl oddělením polovodičové divize od společnosti Motorola v roce 2004. Hlavní činností je výroba součástek pro automobilový a telekomunikační průmysl, mikrokontrolérů a mikromechanických senzorů. Freescale Semiconductor je jednou z 20 největších firem v polovodičovém průmyslu na světě [14].

[1] ARM - Advanced RISC Machine

Freescale nabízí širokou škálu mikrokontrolérů, od 8bitových po 32bitové. V jejich nabídce také nalezneme i 128bitové procesory. Je tedy z čeho vybírat.

4.1.2 Mikrokontroléry Atmel

Firma Atmel Corporation je výrobce polovodičů a integrovaných obvodů založená roku 1984. Předmětem podnikání firmy Atmel je široký okruh aplikačních segmentů včetně konzumního sektoru, počítačů, počítačových sítí, zdravotnictví, telekomunikačního automobilového, leteckého a vojenského průmyslu.

Mezi jeho známé produkty patří zejména řada mikrokontrolérů AVR a AVR32, dále také rádiové zařízení, EEPROM a Flash paměti a mnoho dalších produktů. Nabízí také „SOP“, tedy „system on chip“ řešení.

4.1.3 Mikrokontroléry Microchip

Microchip Technology je americká firma založená v roce 1989, zabývající se taktéž výrobou polovodičových součástek jako jsou mikrokontroléry, sériové EEPROM paměti, sériové SRAM paměti, rádiové zařízení, obvody pro správu napájení a řízení baterií, signálové procesory, a čipy pro různé interface - USB, Ethernet apod.

Nejznámějším produktem firmy Microchip Technology jsou mikrokontroléry PIC. Jsou založeny na harvardské architektuře a vyznačují se velmi omezenou instrukční sadou. Jsou tak velmi oblíbené u domácích kutilů a amatérů.

4.1.4 Mikrokontroléry STMicroelectronics

STMicroelectronics je franzousko-italská mezinárodní firma se sídlem ve Švýcarsku založená roku 2000. Jedná se o největšího evropského výrobce elektronických a polovodičových součástek. Vyrábí jak mikrokontroléry, tak polovodičové součástky pro mobilní multimediální komunikaci, automobilový průmysl, paměťové elementy a mnoho dalšího. Firma STM se výrazně specializuje na nízkopříkonová řešení,

Poznámka autora: Z osobního pohledu nemám s STMicroelectronics dobré zkušenosti z hlediska dokumentace k mikrokontrolérům. Snad nadejde čas, kdy STM vytvoří kvalitní a sjednocené dokumentace ke svým výrobkům. Jako jediné řešení pro pohodlný vývoj je použit STM ARM s kombinací Real-Time OS.

4.2 Hardwarové nástroje

Každý renomovaný výrobce mikroprocesorů by měl mít ve své nabídce vývojové kity, což jsou již hotové tištěné spoje osazeny daným mikrokontrolérem, které mají vhodně vyvedené kontakty pro snadné připojení dalších periferií. Toto velmi usnadní vývoj.

Některé kity obsahují i debugger, tj. možnost krokování běžícího programu přímo na mikrokontroléru a sledovat tak chování programu za běhu. Debugger však není podporován všemi kity.

Všichni zmínění výrobci nabízejí vývojové kity pro jejich vlastní mikrokontrolery. Díky tomu je velmi dobrá podpora kitů ze strany výrobce. Vývojových kitů je velké množství a je

opět na vývojáři, aby určil, který je pro něj ten pravý. Kity se liší osazeným mikroprocesorem a také počtem a druhy periferií zabudovaných v kitu.

Výhodou vývojových kitů je implantovaný programátor. Je tedy možné programovat mikrokontrolér bez nutnosti drahých a univerzálních programátorů. Po vyvedení potřebných vodičů je možné programovat kontroléry mimo kit.

4.2.1 Arduino

Za zmínku stojí velmi zajímavá platforma Arduino pro vývoj na mikrokontrolérech AVR od Atmelu. Je to open-source vývojová platforma založena na flexibilním a snadno použitelném hardware. Vzhledem k její jednoduchosti je vhodná pro velkou škálu použití.

Arduino nabízí knihovny právě pro své vývojové kity a umožňuje tak jediným příkazem nahradit několik řádků čistého zdrojového kódu. To vysoce zvyšuje efektivitu programátora, protože není nucen hledat v dokumentaci nastavení potřebných registrů a jiných informací [1].

Poznámka autora: Nevýhoda spočívá v možnosti využití arduino knihoven, která zapříčiní posun k abstrakci a programátor snadno „zleniví“, přestane přemýšlet nad tím proč a jak daná věc funguje a to vede často k těžko odhalitelným chybám. Nicméně využití knihoven může ušetřit čas potřebný k vývoji.

4.3 Softwarové nástroje

Konkurence na trhu v rámci mikrokontrolérů je veliká. Ve světě proti sobě bojují více než 4 výrobci mikrokontrolérů a každý z nich touží po tom, aby zákazník sáhl právě po jejich produktu. Nabízejí vývojové kity se svými mikroprocesory, snaží se o to, aby práce s mikrokontroléry byla co nejjednodušší, a mnoho dalších triků, jak vývojáře přesvědčit o koupi jejich výrobku.

Velmi důležitou roli hrají vývojové nástroje. Je možné vyvíjet a psát zdrojové kódy v textovém editoru, jenže takový přístup je dosti neefektivní. Výrobci se snaží vývojářům poskytnout softwarové nástroje, pomocí kterých jejich práce bude mnohem efektivnější, ve formě vývojového prostředí, ať už vlastního či nějakého pluginu do univerzálních programovacích nástrojů.

Vývojový nástroj poskytuje mnoho usnadnění, od napovídání kódu až po WYSIWYG[2] editaci. WYSIWYG není běžně integrovaným usnadněním, ale například vývojové prostředí CodeWarrior od Freescale pomocí WYSIWYG dokáže zapsat inicializační procedury periferií.

Velkou výhodou vývojových prostředí je podpora debugeru, díky kterému je umožněno ladit kód za běhu aplikace v hardwaru.

Atmel poskytuje vývojové prostředí pro své výrobky zvané Atmel Studio. V současnosti je dostupná verze 6 a je po registraci stáhnutelná zdarma.

Freescale má k dispozici vývojový nástroj CodeWarrior. Ten je dostupný v několika verzích, z nichž nejnižší verze je zdarma na vyzkoušení.

Microchip nabízí pro vývoj prostředí MPLAB, které je dostupné zdarma. Nástroj MPLAB je taktéž dostupný v několika verzích určený pro konkrétní způsoby použití.

[2] WYSIWYG - What You See Is What You Get - Co vidíš, to dostaneš

Stmicroelectronics bohužel nemá vlastní vývojové prostředí, lze však použít vývojové nástroje Atollic TrueSTUDIO[4]. TrueSTUDIO je v současné době dostupné zdarma pouze v trial verzi, takže každých 15minut vyskakuje okénko informující verze trial.

4.4 Zhodnocení situace

Závěrem této stručné rešerše o dostupných platformách jsou shrnuty základní poznatky. Na světě je mnoho výrobců mikrokontrolérů, kteří si neustále konkurují. Musí spolu neustále držet krok a jakmile jeden výrobce přestane stačit druhému, lepšímu, na trhu končí. Díky tomuto pravidlu můžeme čekat velmi podobnou nabídku sortimentu od různých výrobců. Pro příklad mikrokontrolér s jistými parametry od výrobce Atmel je možné nalézt s podobnými parametry například u firmy Freescale. Nenajdeme-li mikrokontrolér u konkurence, je volba platformy jasná.

Základní aspekty pro volbu platformy jsou požadavky, které má platforma splňovat. Dalšími důležitými parametry je především cena a podpora výrobku ze strany výrobce. Velmi důležitá je dokumentace, v případě, že je dobře napsaná, usnadní mnoho práce.

Jak již bylo zmíněno, můžeme nalézt u několika výrobců podobné produkty. V tomto případě je možné kontrolér zvolit na základě sympatií vývojáře k dané platformě.

Volba mikrokontroléru a dalších součástek je diskutována v následující kapitole 5.

Kapitola 5

Koncepce programovatelného razítka

Úkolem systému, stručně popsaného v úvodu, je tisk jednoduchého textu. Bude se jednat o vestavěný systém, který bude zastávat funkci běžného kancelářského razítka.

Běžná razítka jsou velmi praktická v případě, že počet různých razítek nepřekročí únosnou mez. Díky dnes dostupným technologiím můžeme nahradit velké množství různých statických razítek jedním elektronickým razítkem. Tam, kde se používají razítka, se většinou používá i běžný personální počítač, který v kanceláři usnadňuje práci. Již přes 15 let se rozvíjí velmi oblíbené rozhraní USB a je snahou, aby všechna externí zařízení komunikovala právě přes USB, protože rozhraní je velmi jednoduché a univerzální. Základní myšlenkou programovatelného razítka je tedy spojení s personálním počítačem přes rozhraní USB a možností programovat paměť textů razítka.

Cílem také je, aby bylo možné razítko ovládat bez počítače. Razítko tedy musí obsahovat drobný displej a malou klávesnici.

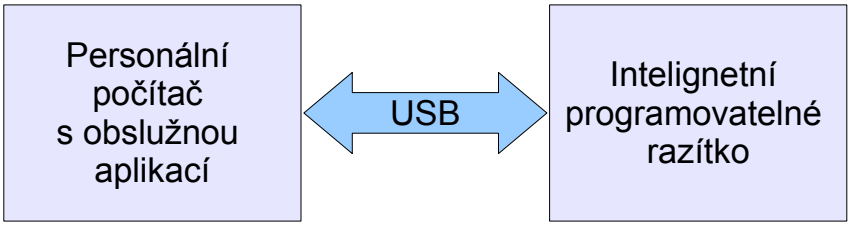
Systém elektronického programovatelného razítka je navrhován jako vestavěný systém. Práce se tedy skládá z několik fází, které jsou typické pro návrh a realizaci vestavěných systémů. Návrh a realizace hardwaru, návrh a implementace firmware řídícího mikrokontroléru, návrh uživatelského rozhraní pro práci s razítkem a testování. V poslední fázi, testování, je předpokládáno, že budou zjištěny nedostatky a celý proces se bude opakovat od konceptuálního návrhu.

5.1 Model situace

Reálný systém bude použit v kanceláři. Razítko by mělo představovat tzv. skříňku, ve které budou zabudovány veškeré periferie a uživatel by neměl mít tušení, že pracuje s počítačem. Veškerá elektronika musí být vestavěná uvnitř krabičky. K interakci uživatele bude použito jako vstupní rozhraní drobná klávesnice s nezbytnými tlačítky pro pohodlnou navigaci v uživatelském rozhraní a jako výstup LCD displej případně tisková hlava. Pro nahrávání nových textů bude použito rozhraní USB.

Model situace bude obsahovat tyto prvky: Inteligentní programovatelné razítko a personální počítač s obslužnou aplikací.

Součástí této práce je návrh inteligentního programovatelného razítka a současně návrh a implementace obslužné aplikace pro osobní počítač. Výsledná aplikace by mohla být

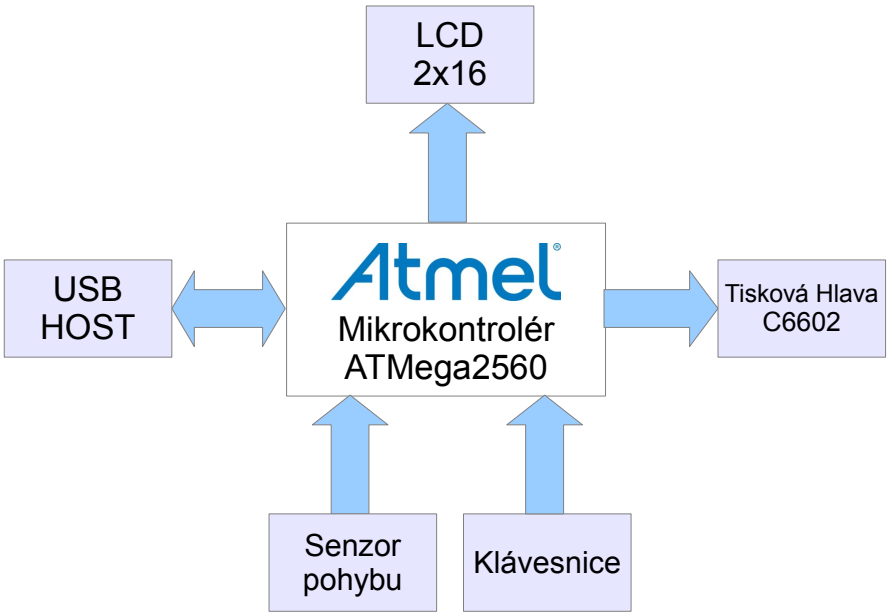


Obrázek 5.1: Model situace.

blokově realizována tak, jak je znázorněno na obrázku 5.1. Je uvažováno, že razítko bude plně použitelné bez osobního počítače. Ovládání ovšem nebude tak komfortní, jako v případě jeho použití.

K navrhovanému systému je vhodné dobře zvolit základní komponenty, abychom jednak nepořídili komponentu, která nebude disponovat potřebnými vlastnostmi, ale také abychom zbytečně nepořizovali komponentu, která by zbytečně měla vlastnosti, které nevyužijeme. Jak již bylo několikrát řečeno, zařízení bude obsahovat mikroprocesor, dále také tiskovou hlavu, LCD displej, klávesnici, enkodér, převodník z USB na USART, napěťový konvertor, a také vhodnou krabičku.

Na obrázku 5.2 je znázorněno blokové schéma inteligentního programovatelného razítka.



Obrázek 5.2: Blokové schéma programovatelného razítka.

5.2 Volba mikrokontroléru

Mikrokontrolérů je na trhu velké množství, jak již bylo řečeno v kapitole 4.1, proto budeme brát ohled především na tyto důležité požadavky: kvalitní IDE (vývojové prostředí), potřebné rozhraní a periferie, dobrou podporu mikrokontroléru výrobcem a také cenu. Požadujeme, aby splňoval zejména tyto vlastnosti:

- 12 (2x8) výstupní pinů - pro řízení tiskové hlavy.
- 8 vstupních pinů podporující přerušení - připojení klávesnice.
- 12 (2x8) vstupně/výstupních pinů - připojení a řízení LCD displeje.
- 8 výstupních pinů pro řízení napájení periferií.
- I2C rozhraní - připojení externí paměti pro uložení textů.
- SPI rozhraní pro připojení rozšiřujících modulů (např. RTC).
- Dostatečně velikou paměť programu $\geq$16 kB.
- Dostatečně velikou paměť dat $\geq$4 kB.
- SMD pouzdro - díky malému pouzdru vznikne úspora místa na DPS $\rightarrow$ menší rozměry na DPS.
- Nízká cena - cena zařízení by neměla být příliš vysoká.

Při výběru byl brán ohled na předchozí požadavky. Nejvíce vyhovující mikrokontroléry nabízely firmy Freescale [14] a Atmel [3]. Mikrokontroléry nabízené firmou Freescale, které vyhovují požadavkům, mají poměrně vysokou pořizovací cenu oproti výrobkům se stejnými parametry od firmy Atmel. Firma Freescale se může pyšnit svým vývojovým prostředím zvaným CodeWarrior, Atmel ji však dohání se svým vývojovým prostředím AVR Studio, které je v současnosti na velmi profesionální úrovni. Klíčová byla možnost pořízení vývojového kitu. Firma Freescale nabízí své vývojové kity s vhodnými mikroprocesory za příliš vysoké částky. Open-source projekt Arduino [1] nabízí rozšířené vývojové kity za přijatelné ceny a obrovskou komunitu amatérských vývojářů, díky kterým je Arduino projekt velmi diskutován. Není tak problém rychle najít řešení nějakého problému.

Na základě těchto požadavků byl vybrán mikrokontrolér řady ATMega2560 vyráběný firmou Atmel, který požadované vlastnosti zcela splňuje. ATMega2560 je mikrokontrolér, jehož instrukční soubor je typu RISC. Tento mikrokontrolér byl také zvolen na základě možnosti pořízení kitu Arduino Mega, který obsahuje právě tento mikrokontrolér pro možnosti vývoje.

Zvolený model ATMega2560 se vyrábí ve dvou různých provedeních, a to v pouzdře TQFP[1] určené pro SMT montáž. Dostupné je také pouzdro CBGA[2], které výrazně redukuje místo na desce plošných spojů. Pro zapájení pouzder BGA je nutná speciální technologie. Tento mikrokontrolér je vybaven 256 kB pamětí programu a poskytuje paměť dat o velikosti 8 KB a elektronicky programovatelnou paměť EEPROM o velikosti 4096 B. Tyto kapacity by měly být pro potřeby projektu dostačující.

Vlastnosti mikrokontroléru ATMega2560:

[1] TQFP - Thin Quad Flat Package
[2] CBGA - Ceramic Ball Grid Array

- 8bitový mikrokontrolér s jádrem AVR a instrukční sadou RISC.
- 2 typy pouzder, TQFP a CBGA 100 pinů.
- Provozní kmitočet 0 - 16 MHz při napájení 4,5 - 5 V.
- Napájecí napětí 1,8 - 5,5 V DC.
- Provozní teplota od $-40\,^{\circ}\mathrm{C}$ do $85\,^{\circ}\mathrm{C}$.
- 256 kB paměť programu, 8 kB interní SRAM paměť dat, 4096 B EEPROM.
- Programovatelný pomocí JTAG.
- 86 programovatelných vstupně výstupních pinů.
- Podpora 6 sleep módů.

Podrobnější specifikace mikrokontroléru ATMega2560 v datasheetu výrobce [21].

5.3 Volba tiskové hlavy

Na základě přehledu technologií tisku zvolíme vhodnou tiskovou hlavu, neboli cartridge. Tato tisková hlava bude vykonávat tisk textu na tiskové médium.

Vzhledem k tomu, že výrobci tiskáren mají vlastní tiskové hlavy a nemají tudíž důvod zveřejňovat dokumentace k řízení tiskových hlav, vzniká problém, jakým způsobem řídit tiskovou hlavu. Tento velký problém ohrožoval celý projekt, neboť zjištění principu řízení tiskové hlavy není jednoduché. Bylo by nutné analyzovat řízení tiskové hlavy z již existující tiskárny. Tento proces je nazýván reverzním inženýrstvím a v některých státech je zákonem zakázaný. Jako autor projektu bych toto riziko nepodstoupil. Naštěstí se však podařilo objevit tiskovou hlavu 51604A od firmy HP, kde je řízení hlavy popsáno v knize Inkjet Applications od autora Matt Gilliland [25]. Autor taktéž zmiňuje, že řízení je totožné s tiskovou hlavou C6602A od HP, která má větší inkoustový zásobník.

5.3.1 Tisková hlava C6602A

Kvůli dostupné literatuře [25], popisující řízení hlavy, byla zvolena tisková hlava C6602. Tisková hlava využívá termální systém pro vstřikování inkoustu. Tento systém je popsán v kapitole 2.2.2.

Tisková hlava disponuje těmito vlastnostmi:

- Rozlišení 96 dpi.
- Počet trysek - 12.
- Rezistivita 65 Ohmů.
- Napájení 20 - 24 V.

Na obrázku 5.3 je fotografie tiskové hlavy C6602 a držáku.

Obrázek 5.3: Tisková hlava C6602 včetně držáku.

5.3.2 Napájení tiskové hlavy

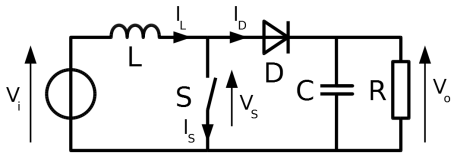
Tisková hlava vyžaduje napájení 20 - 24 V. Takové napájecí napětí není typické pro napájení vestavěných systémů. Vestavěné systémy používají většinou napájecí napětí 5 - 12 V. Vzhledem k tomu, že zařízení by mělo být mobilní, je vyžadováno napájení z baterií. 20V baterie se však nevyrábí v kompaktní velikosti a je tak nutné nalézt jiné řešení.

Otázka zní, jak z nízkého napájecího napětí získat několikanásobně vyšší napětí?

Řešením je tzv. DC-DC měnič neboli konvertor, který dokáže transformovat malé napájecí napětí na napětí větší. Všichni známe transformátory, případně stabilizátory napětí, které napětí snižují. Existují však integrované obvody, které za jistých technik dokáží převádět malá napětí na větší napětí.

Princip funkce takovýchto měničů je založen na vlastnostech cívky a kondenzátoru, které jsou schopné uchovávat energii. Na obrázku 5.4 je zobrazeno schéma velmi jednoduchého měniče, který si v následujících řádcích popíšeme.

1. Je-li spínač S uzavřen, proud protéká skrz cívku L a cívka je nabíjena.
2. Je-li spínač S otevřen, cívka se stane zdrojem zapojeného do série se zdrojem V. Sériové spojení zdrojů zapříčiní součet napěťových úrovní jednotlivých zdrojů.



Obrázek 5.4: Princip DC - DC Step up konvertoru [31].

Ve specializovaných integrovaných obvodech musí být rychlost spínání velmi přesná. Pak by se cívka L nemusela dostatečně nenabít nebo by se mohl kondenzátor C vybít dříve, než by mu byl dodán další náboj.

Na základě tohoto jednoduchého principu fungují speciální integrované obvody a právě jeden takový integrovaný obvod bude použit.

V literatuře [29] je diskutován zdroj energie pro tiskovou hlavu 51604A, která vyžaduje taktéž napájecí napětí 24 V. Autor literatury použil integrovaný obvod LT1930A od Linear Technology. Výrobce v technické dokumentaci k integrovanému obvodu uvádí i příklady zapojení pro konkrétní aplikaci. Nachází se zde i schéma ukázkové zapojení obvodu pro zvýšení napájecího napětí z 5 na 24 V.

K potřebám projektu byl vybrán obvod *LT1930A* výrobce Linear Technology [16], který je schopný zajistit zvýšení napájecího napětí. Tento obvod byl zvolen z důvodů pozitivních referencí a zkušeností. Jeho nasazení bylo realizováno v několika projektech, které jsou prezentovány na síti internet. Použití právě tohoto obvodu bylo dostatečnou zárukou funkce napájení tiskové hlavy C6602A.

5.3.3 Řízení tiskové hlavy

V projektu je použita tisková hlava, označená C6602A, od firmy HP. Výrobce bohužel neposkytuje k tiskové hlavě žádnou dokumentaci, proto bylo nutné najít informace z jiného zdroje.

V knize „Inkjet Applications“, jejímž autorem je Matt Gilliand [25], je velmi detailně popisován princip řízení tiskové hlavy C6602 a zobrazeno schéma zapojení uvnitř tiskové hlavy. Z této knihy bylo v projektu vycházeno při návrhu a realizaci řízení tiskové hlavy.

Autor knihy uvádí, že tisková hlava pracuje na termálním principu (termální princip viz kapitola 2.2.2). To znamená, že trysky jsou realizovány jako malý odpor, který se po přiložení napětí zahřeje na vysokou teplotu, což zapříčiní vymrštění kapky inkoustu. Dle knihy tyto trysky, coby rezistory, vyžadují napájecí napětí 20 - 24 V. Po změření rezistivity jedné trysky byla naměřena hodnota 65 Ohmů. Spočteme-li dle Ohmova zákona proud tekoucí tryskou (rovnice 5.1), dostaneme výsledek o hodnotě 0,37 A. Takový proudový odběr ani napěťovou úroveň není možné zásobovat z pinu mikrokontroléru, proto je nutné najít jiné řešení.

$$I = \frac{U}{R} = \frac{24}{65} = 0,37A \tag{5.1}$$

Řešením tohoto problému bude použití tranzistorového pole ULN2803, který obsahuje 8 tranzistorů typu NPN. Tak bude možné přivádět napětí 24 V na trysky tiskové hlavy malým napětím z mikrokontroléru.

5.4 Volba snímače pohybu

Použití snímače pohybu není nutné, avšak uživatel by se musel naučit pohybovat razítkem přesně danou rychlostí.

Snímače pohybu jsou rozebrány v kapitole 3. Byly zkoumány vlastnosti jednotlivých snímačů, jak optických senzorů tak magnetických i optických enkodérů. Laborováním bylo zjištěno, že optický senzor je citlivý na typ povrchu a jeho odezva není vždy stoprocentní. Oba enkodéry fungují naprosto přesně v závislosti na uražené dráze. Bohužel enkodéry jsou svým fyzickým provedením mohutné a bylo by nutné zvětšovat rozměry razítka. Protože pro potřeby razítka potřebujeme odhadovat přibližnou rychlost pohybu, není vyžadována stoprocentní přesnost. Enkodér, coby senzor pohybu snímá pohyb pouze v jedné ose, zatímco optický kamerový snímač umožňuje snímat pohyb ve dvou osách, osách x a y. Tato vlastnost může být vhodná pro tisk bitmapy, nebo více řádků textu.

I přesto, že optický kamerový snímač není dokonale přesný, vykazuje oproti enkodérům menší rozměry a umožňuje snímat pohyb ve dvou osách. Na základě toho bylo rozhodnuto použití optického kamerového snímače, konkrétně s označením ADNS2610.

5.5 Volba modulu reálného času

Aby funkce razítka byly maximálně využity, nesmí chybět takzvaný RTC[3] modul. RTC modul je speciální obvod, který uchovává informace o aktuálním čase i přes odpojené napájecí napětí, avšak na rozdíl od běžného univerzálního obvodu je taktován krystalem o přesné frekvenci reálného času. Modul sice potřebuje ke svému chodu energii, ta je však dodávána z miniaturní baterie. Energie potřebná k chodu obvodu je tak malá, že miniaturní baterie může v zařízení vydržet až několik let. Přesným krystalem a baterií je zajištěno přesné čítání hodin. Jestliže by byly hodiny taktovány z krystalu mikrokontroléru, čítání reálného času by bylo nepřesné.

Jedním z cílů projektu je náhrada datumových razítek. Často se stává, že uživatel zapomene změnit na datumovém razítku datum. S RTC modulem v inteligentním razítku by tento problém byl eliminován. Datum by se posouvalo automaticky.

Při výběru modulu bylo opět snahou vybrat nejpoužívanější modul, aby byla zaručena vysoká pravděpodobnost funkce. Byl vybrán modul, označen DS1307, který je v síti internet nejvíce diskutovaný.

5.6 Volba vhodného displeje

Displej bude v projektu použit pro možnost obsluhy razítka bez počítače. Na displeji by se měly zobrazovat informace o aktuální poloze v menu a možnosti volby. Je tedy vhodné použít dvouřádkový displej. Aby byla práce v menu příjemná, je dobré zvolit alespoň 16 znaků na řádek, jelikož by se v případě méně znaků muselo textem často rotovat, aby byla zobrazená celá položka.

Displejů je na trhu celá řada, existují grafické displeje, které dokáží zobrazovat libovolné obrazové prvky, nebo alfanumerické, které jsou určeny pro zobrazování znaků. Pro potřeby projektu jsou plně dostačující alfanumerické displeje.

Displeje se dále dělí na displeje bez řadiče, což jsou displeje bez jakékoliv inteligence a je pouze na návrháři, aby zajistil, jak se s displejem bude komunikovat, jak se provede zapsání znaků na displej atd., a nebo na displeje s řadičem, které obsahují mikroprocesor. Tento mikroprocesor plně řeší zápis na obrazovku a návrhář se tímto nemusí zabývat. Mikroprocesor displeje poskytuje komunikační rozhraní, přes které programátor dává displeji instrukce typu „Vytiskni text“, „Smaž displej“.... Protože řadiče velmi usnadní práci, budeme se při výběru zabývat displeji s řadičem.

Byl vybrán dvouřádkový šestnáctiznakový LCD displej s řadičem od firmy MIDAS [13], jehož připojovací konektor je vhodný pro připojení plochého kabelu.

5.7 Volba klávesnice

Klávesnice bude v projektu sloužit jako vstupní rozhraní pro jednoduchou navigaci v menu a obsluhu razítka.

V menu bude potřeba alespoň pěti tlačítek: nahoru, dolů, vpřed, vzad, a tlačítko potvrzení. Dále bude třeba tlačítko pro zahájení tisku (sdílená funkce s tlačítkem potvrzení) a také pro uspání celého elektronického razítka do nízkoodběrového režimu, protože bude napájen z baterie. Celkem tedy 6 tlačítek.

[3] RTC - Real Time Clock - Hodiny reálného času

Cílem bylo nalézt vhodnou klávesničku, která by splňovala požadavky projektu. Bohužel se nepodařilo zajistit klávesnici, která by vyhovovala potřebám tohoto projektu a proto bylo vhodné vyrobit klávesnici vlastní.

5.8 Návrh nabíjení razítka

Razítko má sloužit jako přenosný vestavěný systém, který by neměl být závislý na externím zdroji napětí. Zcela jistě by bylo možné napájet razítko z externího zdroje, ovšem toto řešení by bylo velmi nepohodlné a výrazně by komplikovalo použití razítka. Proto bylo rozhodnuto, že zařízení bude čerpat energii z přenosného zdroje napětí, neboli baterie. Vzniká ovšem otázka, jak zajistit obnovu energie, tedy výměnu baterií nebo nabíjení.

Zařízení je nutné napájet alespoň 6 V, k čemuž je ideální devítivoltová baterie. Nabízí se tedy dvě možnosti, a to: výměna baterie po vybití, což vyžaduje přizpůsobit zařízení k pohodlné výměně baterie, nebo použít nabíjecí baterii, která nebude vyžadovat zásah do zařízení.

Nejlepším řešením je použití nabíjecí baterie, ovšem otázka stále zní, jakým způsobem zajistit nabíjení baterie. Ideálním řešením je integrace inteligentní nabíječky přímo do zařízení a provádět nabíjení například z USB portu. Řešení vestavěné nabíječky vyžaduje dobré zkušenosti a znalosti této problematiky a vzhledem k charakteru projektu bylo nutné věnovat pozornost důležitějším částem projektu.

5.9 Návrh hardwarového řízení spotřeby

Razítko bude napájené z baterie. Je snahou, aby razítko mělo co nejmenší spotřebu co se týče konzumované energie. Bylo potřebné navrhnout řízení spotřeby tak, aby byl odběr co nejmenší.

Nápad je takový: Odpojit napájení periferiím, které nejsou v danou chvíli využívány. Co se ale může při nečinnosti odpojit?

- Ve dne není potřebné podsvícení klávesnice a displeje.
- Netiskne-li se, není potřeba napájet tiskovou hlavu a není potřeba komunikovat s optickým kamerovým snímačem.
- Nekomunikuje-li se s externí pamětí, není důvod paměť napájet.

Můžeme tedy částečně nebo úplně periferii odpojit od napájecího napětí. Typicky se tyto problémy řeší pomocí tranzistorů a řešení v projektu se nebude nijak lišit. Ke spínání periferií bude použito unipolárních tranzistorů typu N a P, z důvodů malé spotřeby, protože jsou řízeny napětím, ne proudem, jak je tomu u bipolárních tranzistorů.

Řízení podsvícení klávesnice a displeje: LED diody se typicky spínají proti GND. To znamená, že anoda je neustále připojena k napájecímu napětí a pomocí tranzistoru je katoda uzemněna. K tomuto použití se používají tranzistory typu N. Aby bylo možné šetřit napájení, když je podsvícení nezbytné, jsou řídící kanály tranzistorů přivedeny na výstup PWM[4], díky kterému je možné měnit střední hodnotu napětí na diodě. Pomocí signálu PWM je možné měnit intenzitu svítivosti a tím šetřit energii.

[4]PWM - Pulse Width Modulation - Pulsně šířková modulace

5.10 Návrh externí datové paměti EEPROM

Externí datová paměť, nazývaná často EEPROM[5], je polovodičová součástka, která slouží k trvalému uchování dat i bez připojeného napájecího napětí. I přesto, že mikrokontrolér obsahuje 4 Kb datové paměti typu EEPROM, je v případě nedostatku paměti navržena externí datová paměť. Taková paměť by mohla být využita v případě uložení bitmapy či více typů písma.

Komunikace s okolím probíhá pomocí I2C rozhraní, což je dvouvodičové sériové komunikační rozhraní vyvinuté firmou Philips. Rozhraní je využíváno k nízkorychlostnímu přenosu dat. Specifikace I2C sběrnice rozděluje zařízení na řídící (Master - zahajuje a ukončuje komunikaci a generuje hodinový signál SCL) a řízené (Slave - čeká na adresaci Masterem a synchronizuje se dle SCL).

Použití paměti prozatím není uvažováno a proto je paměť neosazena. Výrobci pamětí zachovávají stejná pouzdra a v budoucnu je možné paměť zvolit pouze podle ceny a velikosti paměti.

5.11 Návrh komunikace s PC

Bez možnosti ovládat razítko z osobního počítače by bylo razítko málo použitelné. Proto je nutné vyřešit problém, jak komunikovat s počítačem. Možností je několik:

1. Vyvést z inteligentního razítka vodiče pro sériovou linku a připojit se z počítače pomocí sériového portu, případně USB za pomoci převodníku.
2. Vestavět převodník USB - USART do inteligentního razítka a připojovat razítko pomocí USB.
3. Simulovat USB port na pinech mikrokontroléru.

Kvalitním řešením se zdá býti možnost 2 a to vestavět převodník USB - USART do razítka, díky čemuž uživatel nebude mít tušení o tom, že nějaká sériová komunikace probíhá. Razítko se bude připojovat přes USB port. Na trhu je několik integrovaných obvodů realizujících převodník USB - USART. Nejkvalitnější a nejrozšířenější jsou čipy označené FTxxx. K potřebám projektu je zvolen USB - USART převodník FT232R.

[5]EEPROM - Electronically Eraseable Programable Read Only Memory / Elektronicky Mazatelná Programovatelná Paměť Pouze pro Čtení

Kapitola 6

Realizace fyzického provedení

Tato kapitola popisuje fyzickou realizaci razítka, tedy postupy při sestavování prototypového produktu, které využívají rozboru diskutovaného v kapitole 5. Ve zmíněné kapitole bylo rozhodnuto, že zařízení bude jednotné a nebude ke své funkci vyžadovat žádné podpůrné zařízení, např. dokovací stanici. Veškerá elektronika a periferie budou umístěny v jedné krabičce.

Tato část práce byla nejobtížnější částí z celého projektu. Bylo nutné brát ohled na velikost, ergonomii, pohodlí uživatele při manipulaci a v neposlední řadě také design, který je v dnešní době velmi důležitý, chceme-li uspět na trhu.

Fyzická realizace byla realizována v několika krocích:

1. Zjištění rozměrů všech součástek, které budou vestavěny.
2. Náčrt přibližného tvaru a vzhledu razítka.
3. Volba krabičky, které bude představovat tělo razítka.
4. Narýsování výkresu ke strojní obrobě krabičky.
5. Obrobení krabičky (ext. firmou).
6. Vestavba součástek do krabičky.

6.1 Zjištění rozměrů součástek

Razítko bude obsahovat několik součástek. Tyto součástky budou vestavěny uvnitř razítka, nebo namontovány na těle razítka. Ke správnému návrhu je nutné zjistit rozměry součástek. Součástky byly vybrány na základě konceptuálního návrhu rozebíraného v kapitole 5.

Rozměry LCD

Přesné rozměry LCD displeje byly získány z dokumentace, z tzv. datasheetu výrobce [13]. Tyto rozměry byly ověřeny přeměřením.

Bylo zjištěno, že displej je široký 85 mm a vysoký 36 mm. Protože displej je největší součástkou, bude se od těchto hodnot odvíjet velikost celého razítka.

Rozměry klávesnice

Při návrhu klávesnice je vycházeno z dnes zažitých konstrukcí a to takových, že klávesnice by měla být umístěna pod zobrazovací jednotkou, tedy displejem.

Bude-li klávesnice umístěna pod displejem, je k dispozici prostor až 85 mm na šířku. Minimální výšku, která je umožněna kvůli konstrukci tlačítek se nepodařilo dostat na méně než 18 mm.

Na základě těchto myšlenek byl stanoven rozměr klávesnice na 80 x 20 mm, do kterého se musí vtěsnat minimálně 5 tlačítek a konektor pro připojení k základní desce.

Rozměry držáku tiskové hlavy

Držák tiskové hlavy, označený HP Q2347A, bohužel nemá dostupnou dokumentaci. Bylo nutné držák přeměřit pomocí posuvného měřítka. Byly naměřeny tyto hodnoty: š x v x h : 30 x 55 x 21 mm.

Rozměry snímače pohybu

Protože bylo rozhodnuto, že v projektu svoji funkci zastane optický kamerový snímač pohybu, je pro něj nutné navrhnout desku plošných spojů. Tento návrh je podrobně popsán v kapitole realizace hardware 7.3. V dokumentaci snímače [9], výrobce uvádí doporučené rozměry pro desku plošných spojů a potřebných součástek. Ke správné funkci snímače je potřeba čočka HDNS-2100, která disponuje těmito rozměry: šířka x výška : 39,5 mm x 42,0 mm. Vzhledem k tomu, že čočka je největší použitou součástkou, byly tak odvozeny rozměry desky plošných spojů snímače na š x v x h : 40,0 x 50,0 x 15,0 mm. Z těchto rozměrů také vycházel návrh desky plošných spojů pro optický snímač.

6.2 Náčrt a představa konstrukce

Vezmeme-li v úvahu informace o rozměrech z předchozí kapitoly a budeme-li mít představu o propojení periferií, můžeme začít přemýšlet o výsledném tvaru a konstrukci razítka.

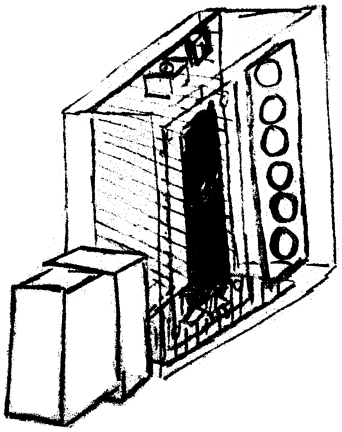
Zcela jistě je nutné dodržet nějaké podmínky, které jsou klíčové k funkci razítka. Například: senzor pohybu musí mít styk s podložkou, trysky tiskové hlavy musí ústit na potiskované médium. Klávesnice by měla být umístěná pod displejem.

Tato část byla spíše kreativního charakteru a vzniklo velmi mnoho různých nápadů a náčrtů. Nebylo však možné říci, který nápad bude nejlepší.

Jak ale bylo postupováno při vytváření náčrtu? Nejprve bylo vzato v úvahu, že razítko musí být dobře uchopitelné do ruky - tedy nějakou krabičku, která bude dobře obejmutelná lidskou rukou. Experimentálně bylo zjištěno, že maximální rozměr, který se dá pohodlně uchopit mezi palec a ostatní prsty, činí 40 mm. Současně bylo bráno v úvahu pohodlné uchopení v případě uživatelského zásahu do razítka pomocí klávesnice. Bylo uvažováno, že klávesnice bude ovládaná palci obou rukou a ostatními prsty bude razítko obejmuto a zajištěno proti vyklouznutí. Z této úvahy plyne rozměr stěny taktéž 40 mm. Velikost boční stěny, na které bude přimontován displej a klávesnice, musela být přizpůsobena velikosti displeje a klávesnice, jejichž rozměry jsou 85 mm na šířku a přibližně 60 mm na výšku obou periferií.

Dále musel být brán ohled na to, že tisková hlava a snímač pohybu musí mít kontakt s podložkou. Aby byla dodržená tato podmínka, musela být tisková hlava umístěná vně krabičky. To je ovšem pozitivní, protože díky tomu bude snadná výměna tiskové hlavy.

Na základě těchto odvození vznikl náčrt, viz. 6.1, ze kterého se vycházelo při sestavování konstrukce.



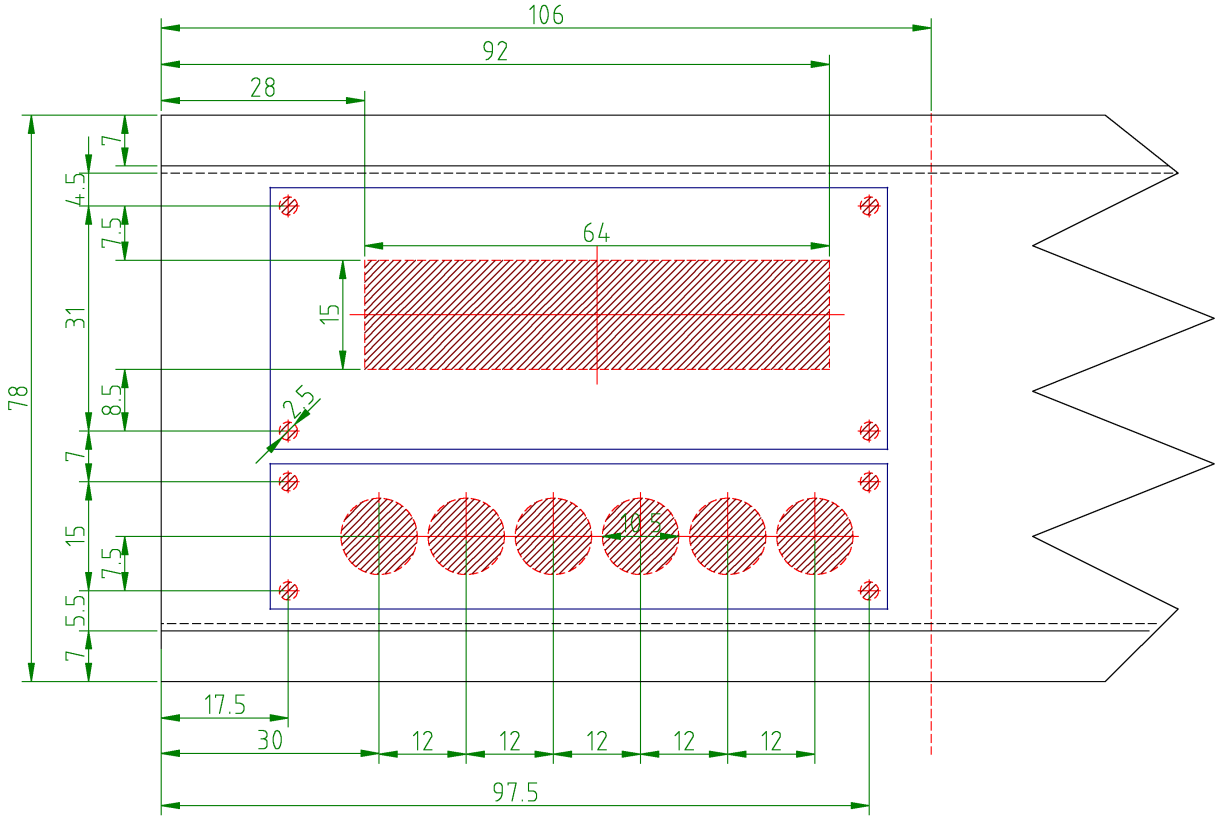
Obrázek 6.1: Jeden z mnoha náčrtů konstrukce, který byl realizován.

6.3 Volba krabičky

Na základě náčrtů a měření z předchozích dvou podkapitol je možné přejít k výběru konkrétní krabičky, která bude představovat tělo razítka. Protože se jedná o prototyp, postačí běžná krabička určená k vestavbě elektroniky. Ideální krabičkou se jevila hliníková krabička s odnímatelnou boční stěnou a dvěmi plastovými bočnicemi. Krabice je od výrobce Hammond, disponuje rozměry š x v x h : 43,0 x 78,0 x 120 mm [15].

Obrázek 6.2: Krabička od firmy Hammond určená k vestavbě razítka [15].

Tato krabička však vyžaduje strojírenské úpravy pro vestavbu klávesnice, displeje a ostatních periferií.



Obrázek 6.3: Náhled výkresu k obrobě krabičky.

6.4 Výkres k obrobě krabičky

Protože krabička je určena pro vestavbu libovolného elektronického systému, je nutné krabičku upravit k potřebám razítka. To znamená vyřezat okénko pro zobrazovací část displeje, díry pro uchycení displeje, otvory pro klávesy a díry k uchycení klávesnice. K tomu, aby bylo možné nechat krabičku strojně obrobit v externí firmě, je vyžadován výkres, jehož náhled je na obrázku 6.3. Tento výkres vychází ze všech předchozích podkapitol.

Obrobení krabičky

Aby krabička k vestavbě byla profesionálně obrobená, bylo nutné ji zaslat včetně narýsovaného výkresu do výroby firmě, která se obráběním zabývá. Obrobení bylo provedeno na počítačem řízené fréze CNC. Cena jednorázového obrobení byla však vysoká, zhruba 500 Kč. U sériové výroby více kusů by se cena pohybovala kolem 50 Kč za kus.

6.5 Vestavba součástek do krabičky

Poslední část fyzického sestavené spočívala ve vestavbě součástek do těla razítka. Vzhledem k tomu, že vše bylo na začátku vyměřené a dobře naplánované, nevyskytl se v průběhu vestavby žádný problém. Obrobená krabička byla obrobena na setinu milimetru přesně a tudíž všechny součástky perfektně zapadly.

Kapitola 7

Realizace obvodového zapojení

Tato kapitola popisuje realizaci konceptuálního návrhu hardwaru. Obsahem je tedy návrh schémat zapojení hlavní desky, klávesnice a desky plošných spojů pro optický snímač. Dále návrh desky plošných spojů (DPS) a seznam potřebných součástek. Navržená schémata, DPS a seznam součástek jsou obsažená v příloze práce a jsou ve formátu návrhového systému DPS EAGLE, viz kapitola 7.1.1.

7.1 Prostředky použité při realizaci hardware

7.1.1 Systém pro návrh desek plošných spojů

Návrhu desek plošných spojů (DPS, angl. PCB) byl proveden v návrhovém systému EAGLE[1] od firmy CadSoft [7] pocházející z Německa. Návrhový systém EAGLE je velmi schopný a výkonný editor DPS, použitelný i na profesionální návrhy. Je dostupný ve free verzi s omezením na velikost DPS (100x80mm). Návrhový systém je dostupný jak pro operační systém Windows, tak i Linux.

Aplikace má v sobě několik podaplikací, které jsou schopné návrhu jednotlivých a individuálních součástek, návrhu elektrických schémat a návrhu desek plošných spojů. Podaplikace v sobě obsahují usnadňující funkce. Například kontrola správně spojených vodičů v schématu. Mezi další významné funkce patří Auto-router. Tato funkce umožňuje automaticky propojit signály na DPS. Strategie propojování vodičů je určena uživatelskými parametry (šířka spoje, izolace, apod). Auto-router však u tohoto projektu nebyl použit. Navržené schémata i DPS je možné exportovat pomocí CAM procesoru ve formátech PNG, PostScript a jiné. Pro návrh byl použit dvouvrstvý návrh DPS.

7.1.2 Osazení desek plošných spojů

Na osazení DPS byla použita technologie SMT[2]. SMT je technologický postup, kdy se součástky pájejí přímo na povrch plošného spoje. Tyto součástky jsou nazývány SMD[3]. Na rozdíl od původní technologie, kdy se musely vývody součástek prostrčit skrz DPS a zapájet na druhé straně, jsou součástky SMD miniaturní. Vzniká tak úspora místa. Dalším kladem těchto součástek je možnost osazení z obou stran DPS. DPS tak může být daleko menší v porovnání DPS řešenou klasickou technologií. Jediná nevýhoda spočívá v tom, že

[1] EAGLE – Easily Applicable Graphical Layout Editor
[2] SMT – Surface Mount Technology
[3] SMD – Surface Mount Device

součástky jsou miniaturní a je obtížné, prakticky až nemožné, využívat technologii SMT pro amatérské návrhy a je třeba speciálního osazovacího zařízení.

V této práci jsou SMT technologií vyrobeny a osazeny DPS hlavní desky, klávesnice a optického snímače pohybu. Použití této technologie vedlo k daleko menší velikosti desek všech modulů. Základní deska tak nabývá rozměrů 75 x 80 mm, modul klávesnice 80 x 20 mm a modul snímače 50 x 40 mm. Na výsledné DPS je i zelená nepájivá maska, která vede ke snadnějšímu osazování a chrání před nechtěným zkratem kovovým předmětem. Pocínování pájecích plošek taktéž usnadňuje práci při osazování. Této pomocné procedury je v práci taktéž využito.

7.2 Realizace hlavní desky

Základní deska vychází z konceptuálního návrhu popsaného v kapitole 5. Protože prototyp obsahoval několik chyb, bylo nutné návrh základní desky opravit. Tato kapitola popisuje již opravený návrh schématu a desky plošných spojů základní desky. Prototyp je však realizován na základě prvního návrhu, který obsahuje chyby.

7.2.1 Popis schématu zapojení hlavní desky

Na obrázku D.3 v příloze D je možné vidět schéma zapojení základní desky. Hlavní částí této základní jednotky je mikrokontrolér ATMega2560. Dále jsou potřebné moduly ke správné funkci vestavěného systému: Stabilizace napájení, převodník USB - sériová linka, tranzistorové pole nezbytné ke spínání trysek, napěťový převodník z 5 na 20 V, konektory pro připojení LCD, klávesnice a tiskové hlavy a také RTC modul. Všechny zmíněné integrované obvody a moduly vyžadují dodatečné drobné součástky, jako rezistory a kapacitory, bez kterých by neplnily správnou funkci.

V následujících podkapitolách si popíšeme jednotlivé části schématu hlavní desky.

Napájení

Elektronika zařízení vyžaduje pro správnou funkci napájecí napětí 5 V. Napětí zdroje proto musí být alespoň 6 V, aby jej bylo možné stabilizovat na požadovaných 5 V, které budou napájet elektroniku základní desky. Svou velikostí i napěťovým potenciálem alespoň ≥ 6 V, je dostačující devítivoltová baterie, i přesto, že kvůli své velikosti má malou kapacitu.

Na obrázku 7.1 je zobrazeno schéma napájecí části. Zařízení bude napájeno přibližně 6 - 9 V. Všechny integrované obvody použity v projektu vyžadují napájecí napětí 5 V. Proto je použit stabilizátor napětí 5 V, označený *7805DT*.

Z dokumentace bylo zjištěno, jaký maximální proud odebírají všechny periferie. Z těchto hodnot byl vypočítám maximální proudový odběr celého zařízení. Výpočet ukázal, že celé zařízení by nemělo přesáhnout proudový odběr větší než 200 mA.

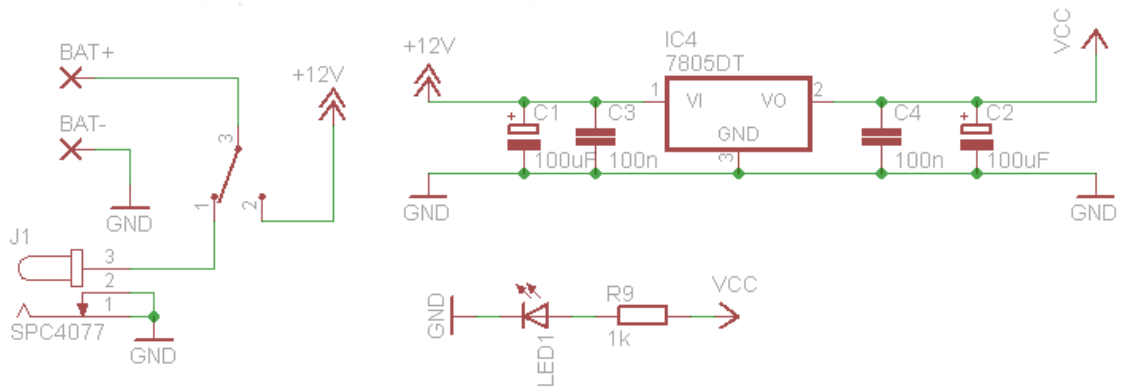
Nabíjecí devítivoltová baterie je tvořena dvěma lithium-iontovými články, kdy každý článek má maximální napětí 4,2 V. Maximální napětí na svorkách baterie je tak 8,4 V max.

Klasický lineární stabilizátor není bohužel ideální součástka, jelikož výkon, který je stabilizován, se přemění na teplo. Nazývá se ztrátový výkon a lze jej vypočítat pomocí vzorce 7.1.

$$P_{ztratovy} = (U_{vstupni} - U_{vystupni}) * I = (8,4 - 5,0) * 0,2 = 0,68\ W \tag{7.1}$$

Výpočtem bylo zjištěno, že maximální ztrátový výkon je 0,68 W. V dokumentaci stabilizátoru [8] je poznamenáno, že ztrátový výkon do 1,3 W nevyžaduje externí chladič. Z tohoto důvodu není nutné na desce plošných spojů řešit chlazení stabilizátoru.

Kondenzátory *C1 - C4* slouží k vyhlazení a filtraci vstupního napětí. Všechny integrované obvody obsahují blízko svých napájecí vodičů tzv. blokovací kondenzátory. jejichž úkolem je zásobovat integrovaný obvod elektrickým proudem při rychlých změnách proudového odběru. *LED1* je informační LED dioda, která signalizuje připojení napájecího napětí.



Obrázek 7.1: Schéma napájecí části.

Nabíjení

Na základě návrhu konceptu je možné razítko nabíjet externí nabíječkou přes nabíjecí konektor. Protože je v zařízení použita lithium-iontová baterie, je nutné nabíjet baterii speciální nabíječkou, určenou k nabíjení lithium-iontových baterií. Použitím nesprávné nabíječky, či špatného nabíjecího režimu hrozí nebezpečí explodování baterie.

Hlavní vypínač přepíná kladnou svorku baterie mezi kladný pól nabíjecího konektoru a svorku kladného napájecího napětí zařízení. Vypínač v poloze OFF propojuje přímo nabíjecí konektor s kontakty baterie a tudíž proces nabíjení není nijak kontrolován. Viz schéma zapojení 7.1.

Mikrokontrolér ATMega2560

Mikrokontrolér ATMega2560 je zapojen dle doporučení výrobce [2]. Pouzdro mikrokontroléru je čtvercového tvaru a jeho vývody jsou vyvedeny do všech stran. Proto je na desce plošných spojů umístěn přibližně do středu desky, aby bylo možné dobře vyvést vodiče k jiným součástkám.

Jako hodiny taktující jádro mikrokontroléru je umístěn krystal o frekvenci 16 MHz, lze však použít interní krystal obvodu. Je doporučeno připojit k externímu krystalu kondenzátory o velikosti 22 pF. Kondenzátory jsou ve schématu označeny *C7-C8*. K programování ATMega2560 je vyveden konektor 2x3 pin header označený *ISP* a využívá standardu JTAG.

Resetovací vodič mikrokontroléru je taktéž zapojen dle doporučení výrobce. Resetovací tlačítko bylo vyloučeno, protože zařízení bude možné resetovat vypnutím.

ATMega má všechny vstupy/výstupy sdruženy po osmi do takzvaných portů. Porty jsou označeny velkým písmenem abecedy. Připojení periferií k jednotlivým vstupům a výstupům je popsáno v každé podkapitole dané periferie.

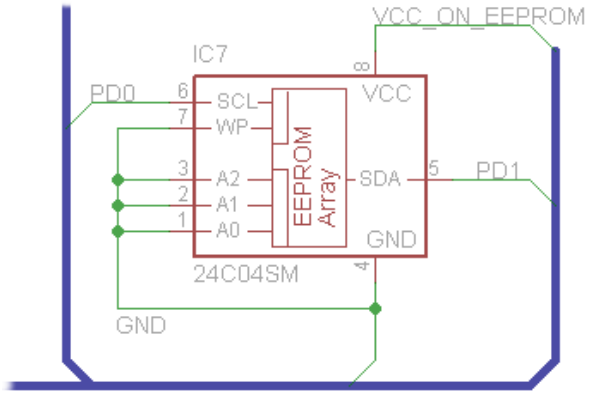
Protože mikrokontrolér obsahuje 100 vývodů, jeho schéma zapojení nabývá velkých rozměrů a proto čtenář může schéma zapojení mikrokontroléru nalézt v příloze D na obrázku D.4.

Externí datová paměť

Na obrázku 7.2 je možné vidět schématko zapojení externí EEPROM paměti. Pozorný čtenář si jistě všiml, že paměť není napájena vodičem *VCC*, ale *VCC_ON_EEPROM*. To je kvůli úspoře napájení, které je řízeno mikrokontrolérem. Více je o této problematice napsáno v podkapitole 7.2.1.

Dále si můžeme všimnout, že ke komunikaci jsou opravdu použity pouze dva vodiče SDA a SCL, připojené k mikrokontroléru na port D k V/V pinům PD0 a PD1.

Cílové zařízení slave se vybírá zasláním adresy. Část adresy je přidělována výrobcem. Nastává ovšem problém, objeví-li se na sběrnici dvě stejná zařízení. Některé periferie umožňují částečnou změnu adresy pomocí vstupů *A0-A2*, čímž se eliminuje tento problém. Protože bude připojena paměť pouze jedna, není adresa nijak modifikována a adresové vodiče jsou připojeny ke GND, jak je možné spatřit na obrázku 7.2.

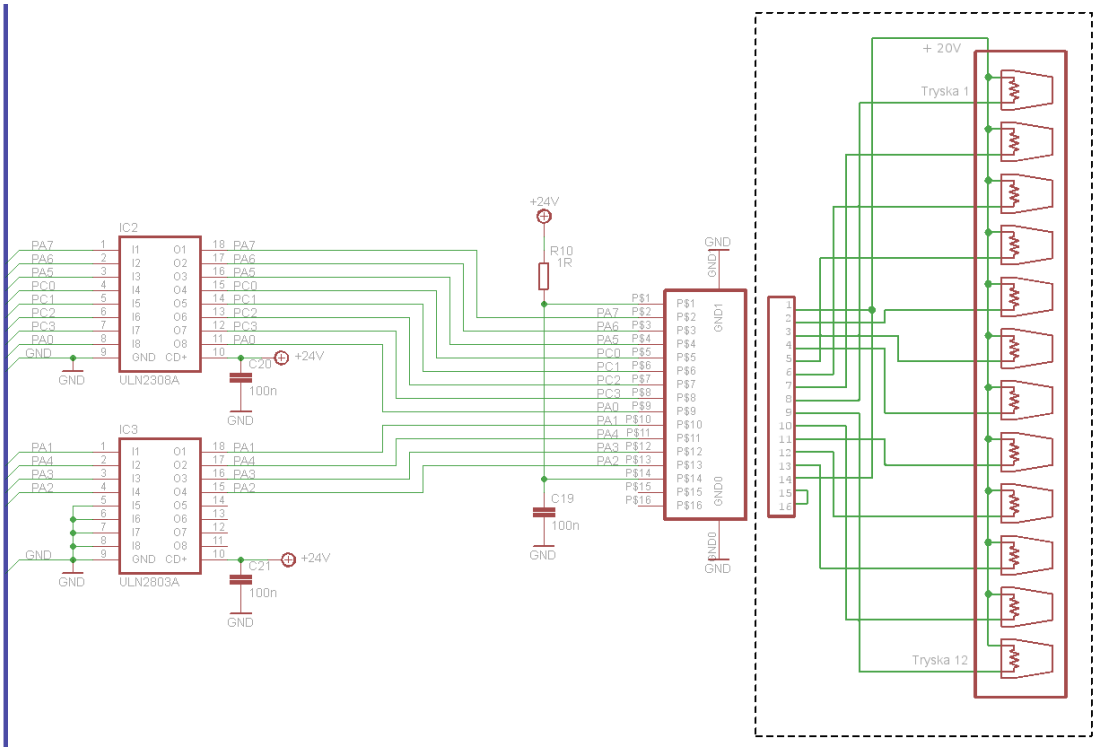


Obrázek 7.2: Schéma zapojení EEPROM.

Ovladač tiskové hlavy

Protože tisková hlava obsahuje 12 trysek, bude nutné použít dvě tranzistorová pole. Na obrázku 7.3 je náhled schématu ovladače tiskové hlavy. Jsou vidět obě tranzistorová pole a signály připojené k mikrokontroléru. Tisková hlava je připojená k mikrokontroléru na celý port A a polovinu portu C. Aby byl návrh desky plošných spojů jednodušší, jsou signály přizpůsobeny zapojení uvnitř tiskové hlavy. To vysvětluje neuspořádané signály ve schématu zapojení.

Princip spínání je takový: Každá tryska tiskové hlavy je trvale připojena k napájecímu napětí 20 - 24 V. Ve výchozím stavu jsou tranzistory, obsažené v integrovaných obvodech



Obrázek 7.3: Schéma zapojení řadiče tiskové hlavy.

IC2 a *IC3*, zavřené. Tento stav zajistí, aby trysky nebyly uzemněné a tím pádem není uzavřen elektrický okruh. Přivedením napětí na jednu z takzvaných *bází* integrovaného obvodu *IC2* a *IC3* se tranzistor otevře, uzemní tak záporný pól trysky a uzavře elektrický okruh. Na trysce se v tu chvíli objeví mezi kladným a záporný pólem napětí 20 - 24 V.

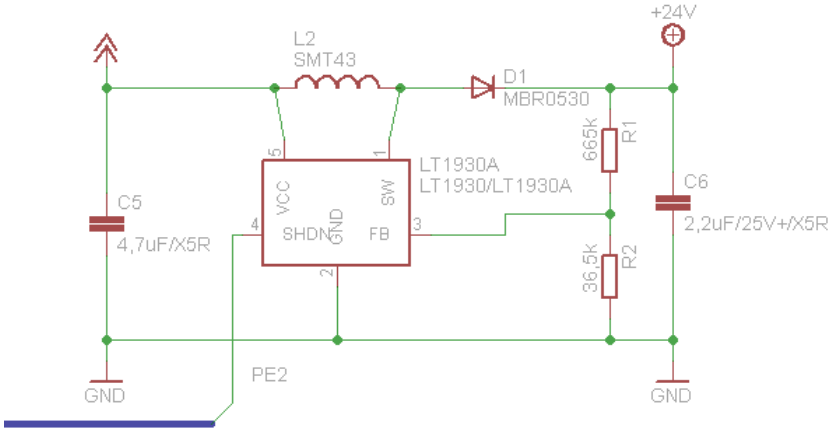
V blízkosti integrovaných obvodů ULN se nacházejí takzvané blokovací kondenzátory *C20 a C21* o hodnotě 100 nF. Rezistor *R10* slouží jako ochrana před zkratem.

DC-DC měnič - Booster

Schéma zapojení měniče je inspirováno doporučeným zapojením z dokumentace výrobce [12]. Na obrázku 7.4 je možné spatřit schéma zapojení DC - DC měniče použitého v aplikaci. Cívka *L2* slouží k uchování energie, tak jak je popsáno v principu funkce měniče. Vývod *SW* měniče realizuje spínač. Rezistory R1 a R2 slouží jako dělič napětí, tedy k úpravě výstupního napětí. Jejich poměr určuje napětí na vstupu *FB*, dle kterého je řízena rychlost spínání. Hodnoty rezistorů jsou spočítány dle vzorce z dokumentace [12].

$$R_1 = R_2 * (\frac{U_{vystup}}{1,225} - 1) \tag{7.2}$$

Signálem PE4 z mikrokontroléru je zapínána/vypínána činnost měniče. Tím je zajištěna úspora energie.



Obrázek 7.4: Schéma zapojení DC-DC měniče z 5 na 20 V.

Hardwarové řízení spotřeby

Řízení přívodu napájení k periferiím je navrženo dle konceptuálního návrhu. Aby byly vyrovnány napěťové úrovně, je nutné mít obvod neustále připojen ke GND. Je žádoucí, aby byl spínán kladný pól periferie a záporný byl stále uzemněn. Pomocí tranzistoru typu N to však není možné. Ke spínání kladného pólu se používají tranzistory s P-kanálem. Řídící vývody tranzistoru jsou připojeny na běžný V/V pin mikrokontroléru, kde není třeba PWM, protože periferie potřebuje ke správné funkci maximální napájení systému, tj. 5 V.

Napájení podsvícení je realizováno pomocí tranzistoru typu N, jehož řídící kanál je připojen na PWM pin mikrokontroléru.

Protože tranzistory jsou umístěny blízko spínaných periferií, ukázka schématu není uvedena.

LCD displej a klávesnice

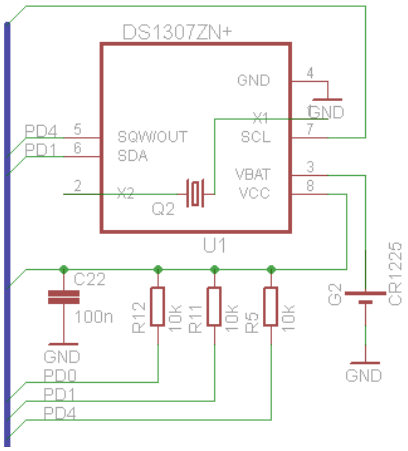
LCD displej je hotová součástka, která je již osazená na vlastní desce plošných spojů a poskytuje vývody sloužící ke komunikaci. Z těchto důvodů nelze s konektorem na displeji nijak manipulovat. Protože displej bude pevně namontován na těle razítka, je potřeba, aby nebyl závislý na hlavní desce. Proto je displej připojen pomocí plochého kabelu, který je možné nalézt například v osobním počítači pro připojení pevných disků.

Na hlavní desce je nezbytné navrhnout připojovací konektor. Ale pozor, konektor displeje je zrcadlený. K tomu, aby se vodiče správně propojily, je nutné zrcadlit konektor i na hlavní desce, byť konektory budou naletovány nezrcadleně. Ano, zní to komplikovaně a návrh také komplikovaný byl.

Klávesnice je osazena na samostatné desce plošných spojů, stejně jako LCD displej. Konektor pro připojení klávesnice je navržen stejně, jako u LCD displeje, aby byl sjednocený princip připojení kabelů.

Návrhu DPS hlavní desky vyžadoval zvýšenou pozornost, aby byly konektory vůči sobě správně umístěny na obou deskách. To znamená aby vývod číslo 1 na hlavní desce odpovídal vývodu číslo 1 na desce klávesnice.

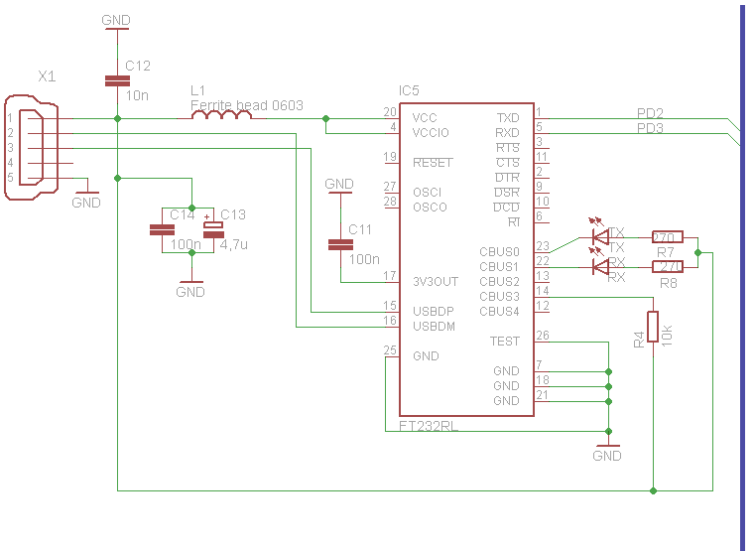
RTC



Obrázek 7.5: Schéma zapojení RTC modulu.

Na obrázku 7.5 je možné vidět schéma zapojení RTC modulu *DS1307* použitého v projektu. Zapojení je realizováno dle doporučení výrobce [10]. Modul komunikuje s mikrokontrolérem pomocí 2 vodičové sériové linky I2C. Kondenzátor ve schématu slouží jako blokovací kondenzátor obvodu a rezistory slouží jako pull-up rezistory na sériovou linku. *G2* je pouzdro pro knoflíkovou baterii CR1225.

Převodník USB - USART



Obrázek 7.6: Schéma zapojení USB - USART převodníku FT232R.

V projektu je použit převodník *FT232R*, který je dostupný v SMD pouzdře. Schéma zapojení 7.6 je opět následováno doporučením výrobce [11]. Pro připojení k PC je použit

mini-USB konektor. Obvod je napájen z USB a má společný zemnící vodič se zbytkem systému. Je možné spatřit blokovací kondenzátory *C14* a *C13* a také informační diody, které signalizují vysílání / příjem. Komunikační vodiče RX a TX jsou připojeny k příslušným pinům mikrokontroléru.

7.2.2 Deska plošných spojů základní desky

Na obrázku E.1 v příloze E je zobrazena deska plošných spojů hlavní desky. Rozměry desky jsou 75 x 80 mm. Návrhu desky plošných spojů se držel doporučení a technik z literatury [35].

7.3 Realizace modulu snímače pohybu

Optický kamerový snímač bylo nutné umístit na samostatnou desku plošných spojů a to z důvodů rozdílného umístění v těle razítka. Propojení se základní deskou je zajištěno pomocí 4 měkkých vodičů. Ke komunikaci se senzorem je opět použita dvouvodičová sériová komunikace, kde jeden vodič přenáší data a druhý se přenáší takt hodin. Výrobce snímače doporučuje schéma zapojení, podle kterého byl senzor zapojen [9].

Na obrázku D.1 v příloze D je vidět zapojení čipu ADNS2610. *K2* je konektor pro připojení k základní desce. Opět jsou vidět blokovací kondenzátory *C2* a *C3*. Obvod je taktován externím keramickým krystalem o frekvenci 24 MHz. Za zmínku stojí tranzistor *T1* a dioda *D1*. Kvůli úspoře energie si obvod řídí intenzitu svítivosti diody. Je-li senzor v nečinnosti, dioda je v úsporném režimu. Tento jev je také možné pozorovat u optických myší.

Na další sadě obrázků E.2 v příloze E je zobrazena horní a spodní strana desky plošných spojů snímače. Deska má rozměry 40 x 50 mm.

7.4 Realizace modulu klávesnice

Klávesnice je důležitým prvkem sloužícím k ovládání razítka. Je umístěna pod displejem a taktéž bude připevněná k tělu razítka. Je tedy nutné navrhnout desku plošných spojů pro klávesnici. Snahou však je zachovat podobné proporce a vlastnosti DPS LCD displeje, aby způsob připojování byl totožný s LCD displejem.

Zapojení klávesnice k mikrokontroléru je velmi jednoduché. Pin mikrokontroléru je nastaven jako vstup a stiskem tlačítka je tento vstup uzemněn. Stejného principu je využito i při zapojení klávesnice. Protože je použito pouze 6 kláves a mikrokontrolér disponuje velkým množstvím vstupních pinů, není třeba používat zapojení kláves do matice.

Na obrázku D.2 v příloze D je možné vidět schéma zapojení klávesnice. Klávesy klávesnice jsou podsvícené. Každá LED dioda má vlastní rezistor k omezení proudu. Je plánováno, že intenzita podsvícení bude řízena pulsně šířkovou modulací.

Cílem bylo navrhnou desku plošných spojů podobnou desce LCD displeje. Z náhledu E.3 v příloze E je patrné, že DPS je proporčně podobná DPS displeje. Tím je zajištěn jednotný princip připojení k hlavní desce.

Kapitola 8

Koncepce firmware

Tato kapitola popisuje použité nástroje a teoretický postup při vytváření firmwaru mikrokontroléru.

8.1 Nástroje použité k implementaci

Aby bylo možné firmware projektu vyvíjet, jsou potřebné určité nástroje. Mezi tyto nástroje patří vývojové prostředí, tzv. programátor potřebný k naprogramování součástky a možnost ladění. V následující řádcích jsou popsány jednotlivé nástroje použité při implementaci.

8.1.1 Vývojové prostředí Atmel Studio

Atmel Studio 6 je integrovaná vývojová platforma pro vývoj a ladění firmwaru mikrokontroléru od firmy Atmel. Atmel Studio 6 poskytuje bezproblémové a snadno použitelné prostředí. Umožňuje psát, vytvářet a ladit aplikace napsané v jazyce C/C++ nebo v assembleru. Atmel studio je dostupné zdarma a je integrované současně s Atmel Software Framework, což je velká knihovna otevřených zdrojových kódů včetně 1600 příkladů pro mikrokontroléry ARM a AVR. Atmel Software Framework posiluje Atmel Studio velkým množstvím low-level kódů potřebných k mnoha projektům.

Se zavedením Atmel Galerie a Atmel Spaces se výrazně zjednodušuje design návrhu firmwaru a díky tomu je snížena doba vývoje a náklady na programátora. Atmel Galerie je online aplikace, která umožňuje online nakupování vývojových nástrojů a designů softwarů pro vestavěný systém. Atmel Spaces je technika pro vývoj v týmu a poskytuje jednotný prostor pro vývoj projektů zaměřených na Atmel mikrokontrolérech.

Stručně řečeno, Atmel Studio je standardní integrované vývojové prostředí vhodné k vytváření nových firmwarů pro mikrokontroléry AVR [3].

8.1.2 Programátor

K naprogramování součástky je potřeba takzvaný programátor. Ten dokáže strojový kód nakopírovat do paměti mikrokontroléru.

K programování byl použit programátor PRESTO od českého výrobce ASIX [20]. Firma Asix byla založena v roce 1991 a zabývá se vývojem a prodejem nástrojů potřebných k vývoji vestavěných systémů.

Programátor Presto je velmi rychlý programátor moderní koncepce. Podporuje programování již osazených součástek - ISP (In System Programming) a programuje velké

množství součástek - mikrokontroléry, CPLD, FPGA, sériové flash, a EEPROM, apod. S počítačem je propojen rozhraním USB.

8.2 Použité periferie

Na tomto místě jsou popsány jednotlivé periferie mikrokontroléru, které byly využity při návrhu a realizaci projektu.

8.2.1 Univerzální komunikační vývody GPIO

Mikrokontroléry rodiny ATMega mají programovatelné vstupy/výstupy jednotlivých portů. Každý V/V pin má svoji primární funkci. Tato funkce je přiřazena na základě toho, ke které periferii port patří (USART, SPI, ADC, ...). Každý port má konfigurační, stavové a kontrolní registry. Funkce pinů lze těmito registry softwarově změnit. Takto lze například periferie USART nakonfigurovat jako univerzální V/V pin.

Typické vlastnosti GPIO[1]:

- Až 3 možné funkce jednoho V/V pinu.
- V/V registry umožňují:
  - Nastavení pinu jako vstup nebo výstup.
  - Zápis výstupních dat.
  - Čtení logické hodnoty z pinu.
  - Nastavení jednotlivých pinů příslušného portu.

8.2.2 Sériové komunikační rozhraní USART

Rozhraní USART[2] je jedno z velmi rozšířených sériových rozhraních. Často používaná pro komunikaci dvou zařízení. Rozhraní umožňuje využívat protokol RS232 pro standardní asynchronní komunikaci. Mikrokontrolér obsahuje 4 programovatelné USART moduly. Moduly mají tyto vlastnosti:

- Plně duplexní komunikace.
- Synchronní nebo asynchronní komunikace.
- Mód vysílače nebo přijímače.
- Programově nastavitelná velikost datových rámců, režim parity a mód stop-bitů.
- Detekce chyb.
- 3 rozdílné vektory přerušení.

Toto rozhraní je v projektu využito pro připojení ladícího terminálu a také pro komunikaci s čipem FT232R, který slouží jako most mezi USART a USB rozhraním.

[1] GPIO - General Purpose Input Output
[2] USART- Universal Asynchronous Receiver/Transmitter

8.2.3 Další periferie

Samozřejmě, že budou použity další periferie, například optický kamerový snímač, displej apod., ovšem tyto periferie nekomunikují přes žádný standardizovaný protokol. V tomto případě je nutné implementovat podporu komunikace softwarově. Tato problematika je podrobněji řešena v kapitole 9.1 ke každé periferii zvlášť.

8.3 Návrh postupu implementace řídícího firmware

Při vytváření nového projektu je zásadní mít vše od začátku promyšlené. Návrh a stanovení přesného cíle je jedna z nejdůležitějších částí práce na projektu. Realizace pak postupuje dle tohoto návrhu a trvá přibližně 20 - 30 % celkového času. Jakmile se ve fázi realizace objeví chyba v návrhu, může se několikanásobně prodloužit doba dokončení projektu.

Na základě předchozí úvahy bylo snahou odhalit problémy v návrhu ještě před realizací prototypu.

1. V první fázi se pořídí/zakoupí prototypová platforma Arduino, na které se implementují knihovny pro jednotlivé periferie. Implementace knihoven proběhne bez využití Arduino knihoven. Tyto knihovny, sloužící k obsluze periferií budou samostatně otestovány. V případě zjištění problému je možné návrh stále měnit bez výrazných komplikací. Tím bude zajištěna korektní funkce každé periferie a současně modularita systému, protože bude možné využít libovolnou knihovnu periferie k jinému projektu.

2. Další fází je fyzická výroba razítka. Vývoj firmwaru je pozastaven.

3. Nyní jsou připravené knihovny ke každé periferii a máme k dispozici plnohodnotný hardware, pro který můžeme začít vyvíjet firmware. V poslední fázi je potřeba implementovat jádro firmwaru a spolupráci jednotlivých periferií při vykonávání určité funkce.

Kapitola 9

Implementace firmware

V této kapitole je popsána implementace firmwaru mikrokontroléru. V první podkapitole 9.1 jsou detailně popsány knihovny jednotlivých periferií, které bylo nutné implementovat. V další kapitole je popsáno jádro firmwaru, které zajišťuje správnou funkčnost razítka a spolupráci jednotlivých periferií.

K implementaci je využit jazyk C a vývojové prostředí Atmel Studio 6. Projekt je tak realizován jako projekt spustitelný v Atmel Studiu. Je však možné projekt přeložit i na systémech Linux bez využití Atmel Studia.

Na úvod je nutné podotknout, že zde popisovaná implementace firmwaru je aktuální pro první prototyp, kdežto kapitola 7.2 popisuje již upravený hardware odproštěný od nedostatků z první verze prototypu. Rozdíly jsou popsány v kapitole 11.3.

9.1 Knihovny periferií

Tato kapitola detailně popisuje implementace knihoven pro komunikaci s jednotlivými periferiemi, které jsou v projektu použity. Implementace firmwaru se drží návrhu postupu popsaného v kapitole 8.3. To znamená, že v první fázi implementace firmwaru jsou implementovány knihovny a dále pak jádro firmwaru.

9.1.1 Knihovna USART

Knihovna pro obsluhu sériové komunikace je obsažena v souborech *usart.c/h*. Všechny základní funkce jsou implementovány v souboru *usart.c*. Tento soubor obsahuje funkce pro inicializaci a obsluhu periferie. V tomto souboru jsou implementovány čtyři základní funkce, jejichž činnost si popíšeme.

Inicializace

Funkcí pro inicializaci USART je
`USARTn_Init(unsigned int baud)`
kde parametrem je index odvozený z tabulky 22-9 v manuálu [2] v závislosti na požadované přenosové rychlosti a taktování mikrokontroléru.

Odeslání bajtu

První funkcí, kterou je vhodné zmínit, je funkce pro odeslání jednoho bajtu. Při zavolání funkce

`void USARTn_Sendbyte( unsigned char data )`
se program dostane do nekonečné smyčky, která čeká na prázdný odesílací buffer (Bit `UDREn`- USART Data Register Empty v registru `UCSRnA`). Jakmile je buffer uvolněn, program do registru `UDRn`- USART Data Register uloží bajt určený k odeslání. Logika mikrokontroléru provede odeslání autonomně.

Odeslání řetězce

Aby bylo možné odeslat řetězec jedním příkazem, je implementována funkce
`void USARTn_SendString(const char *str)`,
jejímž vstupním parametrem je ukazatel na začátek řetězce. Po zavolání funkce se program dostane do nekonečné smyčky, která odesílá řetězec po jednotlivých znacích s využitím funkce pro odeslání bajtu. Jakmile se v řetězci narazí na terminální znak, je cyklus odesílání ukončen.

Příjem bajtu

Aby byla komunikace oboustranná, byla implementována funkce pro příjem jednoho bajtu. Funkce je implementována jako blokující. To znamená, že program je uvězněn v čekací smyčce, dokud nepřijdou data. Podmínka ukončení cyklu spočívá opět v kontrolním registru `UCSRnA`, kde bit `RXCn` značí naplněný datový registr příchozími daty. Funkce nemá žádné vstupní parametry, ovšem návratová hodnota je přijatý bajt.

9.1.2 Knihovna LCD

Pro komunikaci s LCD displejem je potřeba implementovat komunikační protokol dle dokumentace k LCD displeji [13]. LCD displej má v sobě vestavěný řadič, takže komunikace bude probíhat na úrovni příkazů.

Knihovna, jež je implementována v souborech *lcd.c/h*, obsahuje několik základních funkcí. Implementovány byly pouze ty nezbytné k funkci razítka.

Komunikace s LCD displejem využívá tři řídící vodiče a čtyři nebo osm datových. Protože osmibitový přenos je dvakrát rychlejší nežli čtyřbitový a mikrokontrolér má dostatek V/V pinů, je použit osmibitový přenos.

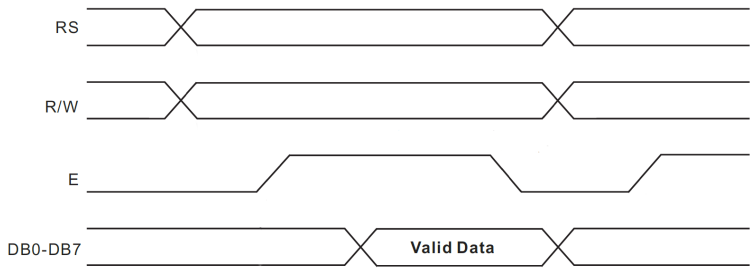
Hlavička knihovny obsahuje definice portů, ke kterým je displej připojen a definice konstant (příkazů). Na obrázku 9.1 je zobrazený princip komunikace s LCD displejem.

Inicializace LCD

Jako každá periferie, i LCD je nutné inicializovat. Inicializace je implementována přesně dle doporučení výrobce a obsahuje pouze nastavení použitých portů jako výstup a následně zaslání sekvence inicializačních bitů.

Odeslání příkazu

Řadič LCD displeje většinou komunikuje pomocí příkazů. Na datový port se vystaví bajt, který určuje příkaz. Následně se nastaví řídící signál E - *Enable* do log.1, tím informuje LCD displej o vystavení příkazu. Jakmile si LCD řadič sejme data z datového portu, je nastaven řídící signál E do výchozí hodnoty. tj. logická 0.



Obrázek 9.1: Diagram komunikace s LCD displejem.

Odeslání znaku

Odeslání znaku na LCD displej je velmi obdobné jako odeslání příkazu, ovšem zápis znaku je proveden do paměti RAM. Zápis do paměti RAM se identifikuje signálem *RS - Register Select* v log. 1. Na datový port se vystaví znak, který se má být na displeji zobrazen. Následně se nastaví oba řídící signály do vysoké hodnoty, čímž informují displej o připravených datech a poté řídící signály nastaví do výchozí hodnoty, tj logická 0.

Nastavení pozice kurzoru

LCD displej využívá kurzor. Kurzor ukazuje, na jakém místě se na obrazovce displeje právě nacházíme a současně je to ukazatel do paměti RAM zobrazovaného textu. Často je potřeba s kurzorem hýbat, proto je implementována funkce
`void LCDGotoXY(uint8_t x, uint8_t y)`.

Vstupními parametry funkce je pozice kurzoru v ose x a y. Funkce pouze realizuje výpočet místa v paměti zobrazovaného textu, kdy paměť je realizována jako 1D paměť. Po spočtení adresy v paměti je adresa odeslána LCD displeji jako příkaz.

9.1.3 Knihovna klávesnice

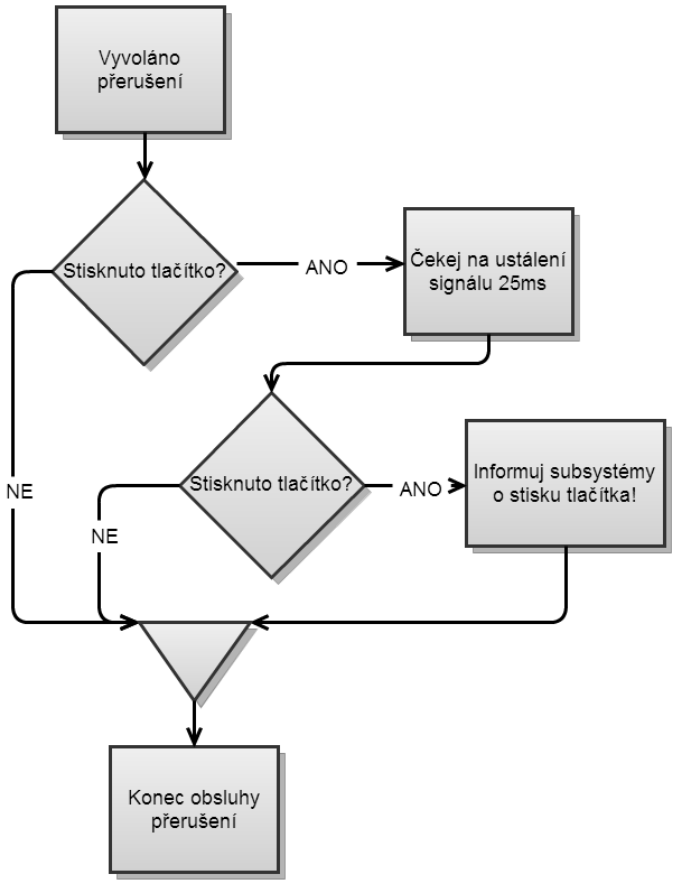
Klávesnice je periferie, která zahajuje akci a je nutné ji okamžitě obsloužit. Mikrokontroléry umožňují použití takzvaných přerušení, které jsou přesně to co k implementaci knihovny pro obsluhu klávesnice potřebujeme.

V hardwarovém návrhu je klávesnice připojena k PCINT[1] pinům. To jsou piny, které spolu sdílí jeden vektor přerušení pro více pinů, typicky pro jeden celý port a přerušení je generováno nástupnou i sestupnou hranou. Je na programátorovi, aby ošetřil nevyžádaná přerušení a obsloužil zařízení (klávesu), která přerušení vyvolala.

Je nezbytné, aby obsluha klávesy netrvala příliš dlouho. Také se musí dbát na to, aby přerušení nezměnilo chování programu v nežádoucí okamžik. Proto jsou v hlavičkovém souboru knihovny klávesnice definovány globální proměnné nesoucí stav každého tlačítka. Vyvolá-li tlačítko přerušení, jeho příslušná globální proměnná se nastaví do nenulové hodnoty.

Nesmíme opomenout ošetření zákmitů tlačítka. Pří stisku nebo uvolnění tlačítka se kvůli mechanickým vlastnostem tlačítka objevují zákmity ve sledovaném signálu. Je tedy nutné počkat na ustálení signálu, angl. takzvaný Debouncing time. Říká se, že 25 ms je dostačující doba na ustálení signálu.

[1] PCINT - Pin Change Interrupt



Obrázek 9.2: Diagram obsluhy klávesnice.

Inicializace

Inicializace klávesnice obsahuje pouze nastavení portů do režimu vstupu a aktivaci pull-up rezistorů. Nakonec je aktivován vektor přerušení od celého portu a povolena přerušení pouze od tlačítek klávesnice.

Obsluha přerušení

Přerušení je generováno jakýmkoliv tlačítkem klávesnice a současně nástupnou i sestupnou hranou, proto je nutné všechny stavy ošetřit v obsluze v přerušení. V první fázi se otestují všechna tlačítka, zda jedno z nich nevyvolalo přerušení. Jestliže se zjistí, že nějaké tlačítko vyvolalo přerušení, je vloženo čekání přibližně 25 ms. Poté je opět ověřen stav tlačítka. Pakliže tlačítko stále vykazuje sepnutý stav, ostatní subsystémy jsou o této události informovány. Jinak je obsluha přerušení ukončena bez zásahu do systému. Tento tok událostí je znázorněn na obrázku 9.2.

9.1.4 Knihovna optického snímače

Optický snímač pohybu je periferie, která komunikuje pomocí dvouvodičové linky. Tato linka obsahuje jeden datový vodič a druhý vodič udávající hodiny, které jsou důležité k syn-

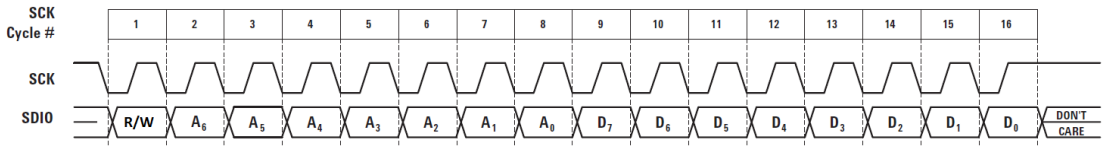
chronizaci. Tento způsob komunikace není definován žádným standardem, ovšem označuje se jako sériová linka, již z principu komunikace. Komunikace je obousměrná, nicméně je zahájena vždy ze strany *mastera*. Snímač nikdy nezahajuje komunikaci. Není k dispozici ani možnost nějakého přerušení.

Protože pro potřeby jádra firmwaru potřebujeme získávat hodnotu uražené vzdálenosti, bylo nutné implementovat knihovnu pro komunikaci se snímačem. Opět je knihovna implementována jako univerzální a přenositelná na jiný typ mikrokontroléru. Programátor však nesmí opomenout změnit v hlavičkovém souboru definice pinů, na které je snímač připojen.

Princip komunikace je následující:

1. Master odešle adresu cílového registru ve snímači.
2. MSB[2] určuje, zda je operace čtení nebo zápis z registru.
3. Jedná-li se o čtení registru, snímač odpoví obsahem registru. Jedná-li se o zápis, je nutné, aby master poslal druhý bajt s obsahem, který se má zapsat do cílového registru.
4. Ukončení komunikace.

Na obrázku 9.3 je možné vidět princip komunikace mezi mikrokontrolérem a optickým snímačem.



Obrázek 9.3: Princip komunikace s optickým senzorem ADNS2610 [9].

Inicializace

Jako každá periferie i senzor pohybu je nutné inicializovat. V tomto případě není inicializace nijak složitá, pouze se synchronizují hodiny signálem SCK

SCK = log.1 ← Počkat 5 μs. ← SCK = log.0 ← Počkat 1 μs. ← SCK = log.1 ← Počkat 1000 ms.

Čtení registru ze snímače

K potřebám razítka stačí pouze číst obsah registru, ve kterém je uložena hodnota dráhy, kterou razítko urazilo od posledního čtení registru.

Funkce je realizována přesně podle obrázku 9.3 v úvodu této podkapitoly, který ukazuje princip komunikace. Implementace je velmi jednoduchá, stačí pouze překlápět logické hodnoty na výstupních pinech mikrokontroléru dle časového diagramu. Z důvodů jednoduchosti zde není funkce zobrazena celá.

[2]MSB - Most significant bit

Za zmínku však stojí jedna zajímavost. Zápis nebo čtení registru obsaženého v ADNS2610 je prováděno sériově a je nutné příchozí data upravit do běžného formátu po osmi bitech. To je provedeno pomocí následující konstrukce, kdy se každý příchozí bit nasune na své místo v cílovém bajtu. Malé r je cílová proměnná pro uložení obsahu čteného registru. Konstrukce provede nasunutí hodnoty na datovém vstupu na správné místo v proměnné r.

```
//Čtení:
for (i=7; i>=0; i--){
  r |= ( ( (UROVEN_NA_SDIO) ? 1 : 0) << i);
}
//Zápis:
for(i=7; i>=0; i--){
  ( (uint8)adresa & (1<<i)) == 0) ? SDIO = 0; SDIO = 1;
}
```

9.1.5 Knihovna pro řízení tiskové hlavy

Aby bylo možné jednoduše tisknout pomocí tiskové hlavy, je opět implementována knihovna, která zajistí vytrysknutí inkoustu a současně zabrání zničení tiskové hlavy z důvodů nedodržení časových limitů kritických pro jednotlivé trysky.

Knihovna obsahuje dvě základní funkce. Opět je přítomna funkce pro inicializaci a dále klíčová funkce nazvaná `spray_ink`. Konstanty, definice připojených pinů tiskové hlavy, jsou definovány v hlavičkovém souboru.

Inicializace

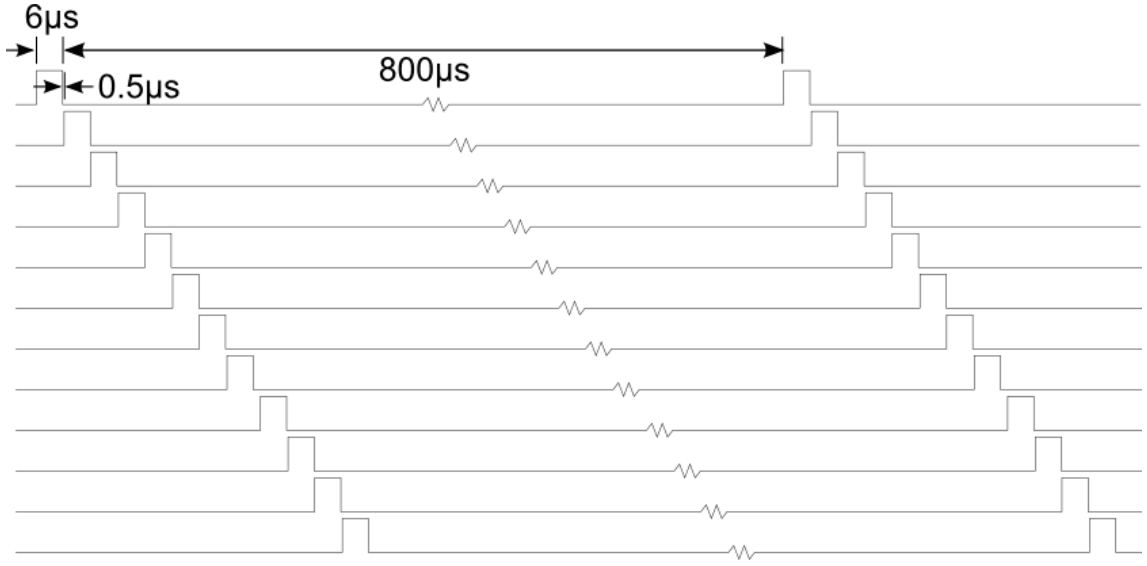
Tisková hlava není periferie, který by vyžadovala složitou inicializaci, jako například LCD displej. Zde postačí pouze nastavit piny jako výstupní a neprodleně je nastavit do logické nuly.

Tisk proužku

Jedinou a nejdůležitější funkcí je funkce pro tisk jednoho svislého proužku. Protože má tisková hlava 12 trysek, nejjednodušší možnost, jak předat proužek k vytištění, je použití neznaménkové proměnné o velikosti 16 bitů. Vstupním parametrem funkce je tedy proměnná obsahující kladné bity na místě, kde má být vytrysknut inkoust, a nulové bity, kde nemá být inkoust vytrysknut. Je počítáno s tím, že nejvyšší 4 bity jsou ignorovány.

Řízení tiskové hlavy vyžaduje obezřetnost v časování. Jakmile se na trysce objeví napětí déle než je předepsáno, tryska se zničí. Proto je sem vložen časový diagram 9.4 ukazující, jak má být časováno řízení tiskové hlavy. Na obrázku si čtenář může všimnout, že v jednom okamžiku může být aktivní pouze jedna tryska. Napětí přiložené na trysce nesmí překročit dobu 10 μs. Experimentálně bylo zjištěno, že ke zničení trysky dojde při 20 μs přiloženého napětí. Dále je třeba počkat mezi vytrysknutím inkoustu z další trysky alespoň 0,5 μs. Trysky potřebují také nějaký čas na vychladnutí. Před dalším výstřikem inkoustu z jedné a té samé trysky se musí počkat ideálně 800 μs. Nebudou-li tyto časové podmínky dodrženy, bude tisková hlava zničena.

Níže je zobrazen kód pro tisk jednoho proužku. Kód by měl být dle předchozího popisu zřejmý. Je možné si všimnout příkazu `cli();` resp. `sei();`, které slouží k deaktivaci



Obrázek 9.4: Časový diagram řízení tiskové hlavy [22].

resp. aktivaci globálního přerušení. Z důvodů ochrany tiskové hlavy je globální přerušení pozastaveno. V případě, že se na trysku přivede napájecí napětí a v tu chvíli je vyvoláno přerušení, které trvá déle než dovolený čas přiloženého napětí na trysce, došlo by ke zničení trysky.

V kódu je možné vidět smyčku `for`. Program postupně vybírá z proměnné strip hodnoty jednotlivých bitů na konkrétní pozici a v případě, že se bit v daném místě rovná 1, přivede odpovídající trysce napětí po dobu 3 μs.

```
void spray_ink(uint16_t strip){
  cli(); //dissable all global interrupts
  for(uint8_t i = 0; i <= NUM_OF_NOZZLES; i++){
    if(strip & 1<<i){
      CARTRIDGE_PORT |=  (1<<i);
      _delay_us(3);
      CARTRIDGE_PORT &= ~(1<<i);
      _delay_us(1);
    }
  }
  sei(); //enable all global interrupts

  //wait to cool down
  _delay_us(800);
}
```

9.1.6 Knihovna pro definici znaků

Protože tisková hlava dokáže v jeden okamžik tisknou pouze 1 x 12 bodů ve svislé orientaci, je nutné vymyslet, jakým způsobem se budou předávat tiskové hlavě informace k tisku

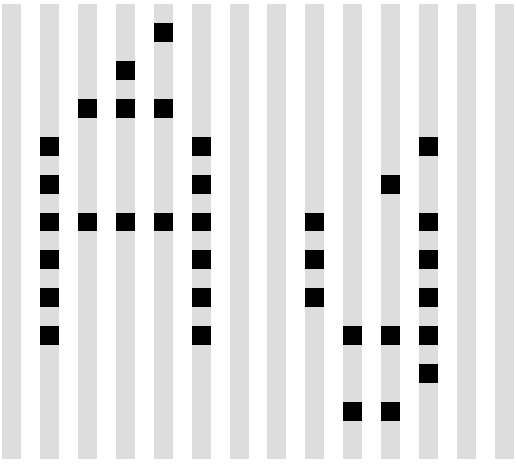
písmena abecedy.

Jedním ze způsobů, jak tisknout daný znak, je rozsekat znak na jednotlivé sloupečky, které se budou postupně předávat k vytištění na potiskované médium.

Funkce k vytištění jednoho proužku má jako vstupní parametr 16-bitovou proměnnou. Z důvodů jednoduchosti není uvažován proporční text, jelikož by se komplikovala funkce, jež převádí znaky na tisknutelné proužky. To znamená, že všechny písmena abecedy budou stejně široká. Není však problém vypracovat podporu proporčního textu později.

Na základě předchozích úvah se jevil jako vhodný nápad vytvořit 2D pole, které bude představovat definici všech znaků. První dimenze obsahuje 256 položek a představuje kódovou tabulku. Druhá dimenze pak obsahuje přesně 7 položek a udává počet proužků pro danou znakovou sadu. Datový typ položek druhé dimenze bude neznaménkový integer o velikosti 16 bitů, který se pak předá funkci k vytištění. Adresace pak může vypadat následovně: `charset[znak v~kódové tabulce][0.-6.~-~index proužku daného znaku]`

Na obrázku 9.5 je zobrazeno rozsekání znaku na jednotlivé proužky v případě velkého dlouhého písmene *Á* a v případě malého dlouhého písmene *ý*



Obrázek 9.5: Ukázka znakové sady vytvořené pro potřeby razítka.

Implementace

Implementace kódové tabulky je realizována v souboru *charset.h*. Kódová tabulka a jednotlivé znaky jsou vytvořeny přesně podle předchozích úvah. Všech 256 znaků kódové tabulky, rozsekaných na 7 sloupečků o velikosti 16 bitů, představuje velké paměťové nároky. Viz rovnice 9.1.

$$\begin{aligned}(\text{Potřebná paměť}) = &\ (\text{počet znaků}) * \\ &(\text{počet sloupců jednoho znaku}) * \\ &(\text{poč.bitů jednoho sloupce}) \ = \\ = &\ 256 * 7 * 16 = \\ = &\ 28672b = \\ = &\ 3584\ B\end{aligned} \tag{9.1}$$

Proto je znaková sada uložená do programového paměťového prostoru, tedy do Flash paměti. Kódová tabulka je kódována dle standardu CP1250. Definici jednoho znaku je možné vidět v ukázce kódu níže.

```
/*0xC1 Á*/ {
  // top <-> bottom of the char
  0b0000000000000000, // |
  0b0000000111111000, // |
  0b0000001001000000, // |
  0b0000011001000000, // | Arrow shows direction
  0b0000101001000000, // |
  0b0000000111111000, // |
  0b0000000000000000  // V
},
```

9.1.7 Knihovna pro řízení spotřeby

Aby bylo možné z jádra programu jednoduše řídit přívod napájení k jednotlivým periferiím, bylo vhodné implementovat knihovnu pro obsluhu řízení napájení.

Tato knihovna je velmi jednoduchá. Její implementace se nachází v souborech *power.c/h* a obsahuje několik drobných funkcí.

Tyto funkce pouze nastaví na daném výstupu úroveň logické 0, či logické 1. Je bráno v potaz, že tranzistory typu PMOS se otevírají logickou 0 a zavírají logickou 1, narozdíl od tranzistorů typu NMOS, jejichž řízení je přesně opačné.

Zde jsou vyjmenované funkce implementované v knihovně pro řízení spotřeby:

- Inicializace: Nastaví řídící piny jako výstup a nastaví jim hodnotu přečtenou z konfigurační paměti.
- Příprava periferií k uspání mikrokontroléru.
- Příprava periferií k probuzení mikrokontroléru.
- Přivedeni napájecího napětí na LCD a ADNS2610.
- Odpojení napájecího napětí z LCD a ADNS2610.
- Překlopení stavu napájení LCD a ADNS2610.
- Přivedení napájecího napětí k podsvícení displeje.
- Odpojení napájecího napětí k podsvícení displeje.
- Překlopení stavu napájení podsvícení displeje.
- Přivedení napájecího napětí k podsvícení klávesnice.
- Odpojení napájecího napětí k podsvícení klávesnice.
- Překlopení stavu napájení podsvícení klávesnice.

- Zapnutí činnosti měniče.
- Vypnutí činnosti měniče.
- Překlopení stavu činnosti měniče.

9.2 Jádro firmwaru

Nyní přichází na řadu popis jádra firmwaru. Toto jádro je část programu mikrokontroléru, který vykonává funkce razítka, a využívá k tomu funkce knihoven, které byly popsány v předchozí kapitole.

9.2.1 Hlavní kód

Hlavní kód programu je uložen v souboru *InkStamp.c*. V tomto souboru je implementována hlavní funkce `main()`, která provede inicializaci veškerých periferií, přerušení a inicializaci jádra. Následně je volán kód jádra `kernel()`.

Hlavičkový soubor *InkStamp.h* obsahuje nalinkování hlavičkových souborů jednotlivých knihoven a definice o použitém mikrokontroléru a frekvenci krystalu.

Po zavolání funkce `kernel()` z hlavního programu se začne vykonávat hlavní smyčka programu, která je ukončena pouze vypnutím nebo resetem zařízení.

Protože razítko bude schopno nabízet uživatelské prostředí, musí se počítat i s implementací menu nabídky. Předpokládá se, že po spuštění jádra bude spuštěno menu, které bude umožňovat interakci s razítkem. Po zvolení položky v menu se vykoná příslušná funkce. Detailně se jednotlivými částmi budeme zabývat v následujících několika řádcích.

9.2.2 Implementace nabídky menu

Vzniklo několik nápadů realizace menu, ovšem všechny nápady byly komplikované nebo neefektivní. Proto byla hledána inspirace z jiných zdrojů, které poskytují kvalitní a odladěné řešení.

Implementace menu byla z velké části inspirována ze zdroje [19], protože se jedná o poměrně kvalitní implementaci. Navigace v menu byla ovšem upravena k potřebám razítka.

Idea tohoto menu spočívá v použití ukazatele na funkci. Hodnota ukazatele na funkci je odvozená z aktuální položky v menu. To znamená, že nacházím-li se v menu na konkrétní položce číslo 1, ukazatel na funkci obsahuje ukazatel funkce číslo 1. Každá položka druhé úrovně v menu má implementovanou vlastní funkci.

Kód zobrazený níže ukazuje nezbytné části implementace.

V první části je definujeme globální proměnnou, která ponese ukazatel na zaměřenou funkci. Pak je nutné definovat strukturu, která ponese aktuální stav menu.

```
typedef void (*FuncPtr)(void);
//function pointer
FuncPtr FPtr;

//current menu and submenu state
struct Menu_State{
```

```
  uint8_t menuNo;//1,2,3,4,5
  uint8_t subMenuNo;//1,2,3
  uint8_t menuLevel;
}MN;
```

Texty jednotlivých položek jsou uložené v programové paměti a jejich názvy jsou odvozeny dle úrovně zanoření v menu.

```
//menu 1
const uint8_t MN100[] PROGMEM=,,1-TISK‘‘;
//menu 1 submenus
const uint8_t MN101[] PROGMEM=,,1.1-Tisk textu‘‘;
const uint8_t MN102[] PROGMEM=,,1.2-Tisk bitmapy‘‘;
//menu 2
const uint8_t MN200[] PROGMEM=,,2-MOZNOSTI TEXTU‘‘;
//menu 2 submenus
const uint8_t MN201[] PROGMEM=,,2.1-Zvolit text‘‘;
const uint8_t MN202[] PROGMEM=,,2.2-Editovat text‘‘;
const uint8_t MN203[] PROGMEM=,,2.3-Smazat text‘‘;
...
... etc.
```

Další částí menu je pole ukazatelů na texty nejvyšší úrovně:

```
//Arrays of pointers to menu strings stored in flash
const uint8_t *MENU[] ={
  MN100, //menu 1 string
  MN200, //menu 2 string
  MN300, //menu 3 string
  MN400, //menu 4 string
  MN500  //menu 5 string
};
```

A následuje pole ukazatelů na texty nižší úrovně:

```
const uint8_t *SUBMENU[] ={
  MN101, MN102,         //submenus of menu 1
  MN201, MN202, MN203, //submenus of menu 2
  MN301, MN302,         //submenus of menu 3
  MN401, MN402, MN403, //submenus of menu 4
  MN501, MN502          //submenus of menu 5
};
```

Nyní je potřeba říci, jak vypadá struktura menu. To znamená kolik položek obsahuje první úroveň a kolik položek obsahuje každá další úroveň zanoření. Je to důležité k výpočtu ukazatele na funkci na základě pozice v menu:

```
const uint8_t MSTR2[] PROGMEM ={
5,  //number of menu items
2,  //number of submenu items of menu item 1
3,  //number of submenu items of menu item 2
2,  //number of submenu items of menu item 3
3,  //number of submenu items of menu item 4
2   //number of submenu items of menu item 5
};
```

Poslední nezbytným polem je pole ukazatelů na funkce, které je zobrazeno pod textem:

```
//Array of function pointers in Flash
const FuncPtr FuncPtrTable[] PROGMEM={
  func000,                      //default function (while(1))
  func101, func102,             //functions for submenus of menu 1
  func201, func202, func203,    //functions for submenus of menu 2
  func301, func302,             //functions for submenus of menu 3
  func401, func402, func403,    //functions for submenus of menu 4
  func501, func502
};
```

Ke správné funkci menu je zapotřebí funkce, která na základě pozice v menu vypočítá správný index do tabulky ukazatelů na funkci:

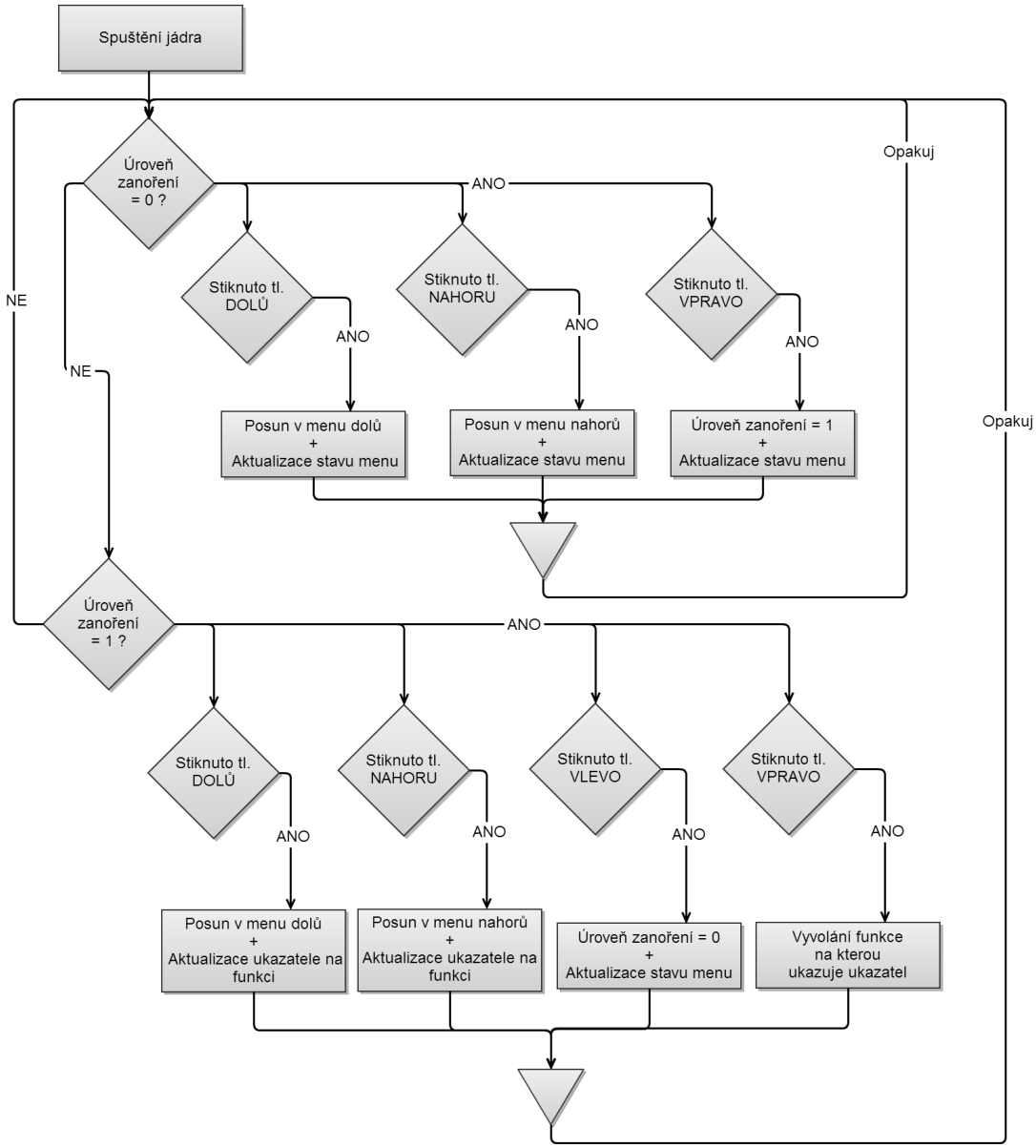
```
uint8_t MFIndex(uint8_t mn, uint8_t sb){
  uint8_t p=0;//points to menu in table of function pointer
  for(uint8_t i=0; i<(mn-1); i++){
    p=p+pgm_read_byte(&MSTR2[i+1]);
  }
  p=p+sb-1;
  return p;
}
```

Nyní je vše připraveno k použití menu. V inicializaci se nastaví všechny položky na výchozí hodnoty a čeká se na akci uživatele. Pro názornost je uveden vývojový diagram 9.6, který graficky znázorňuje použití menu.

Typicky se neustále udržuje a sleduje stav menu. Menu je rozděleno na dvě úrovně zanoření. V první úrovni zanoření je možné tlačítky nahoru a dolů listovat nultou úrovní. Stiskem tlačítka → se menu zanoří do druhé úrovně, která obsahuje další položky. Tyto položky už představují nějakou akci. Opět je mezi položkami možné listovat tlačítky nahoru a dolů, avšak stisk tlačítka vpravo vyvolá funkci.

Jak již bylo řečeno, jsou použity takzvané ukazatele na funkci, které jsou odvozeny z aktuální pozice v menu. Aktualizace ukazatele na funkci probíhá tímto způsobem:

```
FPtr=(FuncPtr)pgm_read_word(
      &FuncPtrTable[MFIndex(MN.menuNo, MN.subMenuNo)+1]     );
```



Obrázek 9.6: Vývojový diagram implementace menu.

`FPtr` je ukazatel na funkci, `pgm_read_word` slouží ke čtení dat z programové paměti, `&FuncPtrTable[]` je tabulka ukazatelů na jednotlivé funkce a `MFIndex(MN.menuNo, MN.subMenuNo)` je funkce, která na základě pozice menu vypočte index do pole ukazatelů na funkci.

9.2.3 Implementace paměťového prostoru datové paměti

Protože mikrokontrolér disponuje 4 kB EEPROM paměti, je možné tuto paměť využít k uložení konfigurace a několika textů, které budou k dispozici i po odpojení napájení

razítka.

Rozdělení paměťového prostoru

Paměť je možné adresovat po bajtech. Velikost 4 kB tak odpovídá adresovému prostoru od `0x000` d `0xFFF`. Protože je paměti dostatek, je možné si dovolit vyhradit větší prostor než je nezbytně nutný a vyvarovat se tak případným nedostatkům paměti v budoucnu. Proto bylo rozhodnuto, že paměť bude rozdělena na bloky o velikosti 256 B. Experimentálně bylo zjištěno, že vhodný počet uložených textů je přibližně 5.

- od `0x000` do `0x0FF` ... Konfigurační prostor
- od `0x100` do `0x1FF` ... Text číslo 1
- od `0x200` do `0x2FF` ... Text číslo 2
- od `0x300` do `0x3FF` ... Text číslo 3
- od `0x400` do `0x4FF` ... Text číslo 4
- od `0x500` do `0x5FF` ... Text číslo 5

Konfigurace razítka bude mít vyhrazený prostor od `0x000` do `0x0FF`, což odpovídá 256B paměti. Prozatím jsou využity 3 bajty k uložení konfigurace. Ano, čtenář by mohl namítat, že se jedná o plýtvání datovým prostorem, ovšem je-li paměti dostatek, je lepší rezervovat větší prostor a vyhnout se tak problémům v případě rozšíření. Cílem také bylo přehledné zarovnání paměťového prostoru.

Význam tří konfiguračních bajtů je vysvětlen zde:

1. `0x000` Index zvoleného textu
2. `0x001` Podsvícení LCD displeje (možnost nastavit intenzitu od 0-255)
3. `0x002` Podsvícení klávesnice (možnost nastavit intenzitu od 0-255)

Čtení a zápis z datové paměti

Čtení a zápis do datové paměti je prováděn pomocí funkcí z knihovny `avr/eeprom.h`, jež jsou součástí vývojového prostředí Atmel Studio.

9.2.4 Implementace funkce tisku

Implementace tisku je klíčovou funkcí celého projektu. Bez této funkce by razítko neplnilo požadovanou funkci. Protože bylo vše od začátku pečlivě promyšleno, implementace se obešla bez větších komplikací.

Nejdříve je nutné získat text, který bude tisknut. Tento text je uložen v EEPROM paměti na dané adrese. Adresa zvoleného textu je uložena v konfigurační struktuře razítka, případně v prvním bajtu konfiguračního bloku paměti EEPROM.

Transformace textu na proužky

Po získání textu, který se má tisknout je pomocí funkce
`getStripsFromText(text)`
provedena transformace textu na proužky. Vstupním parametrem této funkce je ukazatel na tisknutý text a návratová hodnota je ukazatel na začátek pole proužků, které se budou předávat funkci k tisku.

Transformace textu na proužku pracuje s využitím knihovny *charset.h*, o které již bylo hovořeno.

V první fázi je zjištěna délka textového řetězce a zní spočten počet proužků. Výsledné číslo, udávající velikost pole s proužky, je použito k alokaci paměťového prostoru.

V druhé fázi se prochází textový řetězec po znacích zleva doprava. Znak v textovém řetězci je indexem do kódové tabulky v souboru *charset.h*. Položka, indexovaná znakem, je reprezentována jako pole o 7 prvcích, jež jeden prvek obsahuje jeden proužek, je překopírována do připravené paměti pro proužky.

Protože tisková hlava má pouze 12 trysek a datový typ je 16bitový, jsou volně k dispozici 4 bity. Tyto bity budou použity ke zjištění konce tisku. Jakmile funkce pro transformaci textu narazí na konec textového řetězce, vloží poslední proužek se samými jedničkami v horních 4 bitech, čímž bude možné detekovat konec tisku.

Tisk s využitím senzoru pohybu

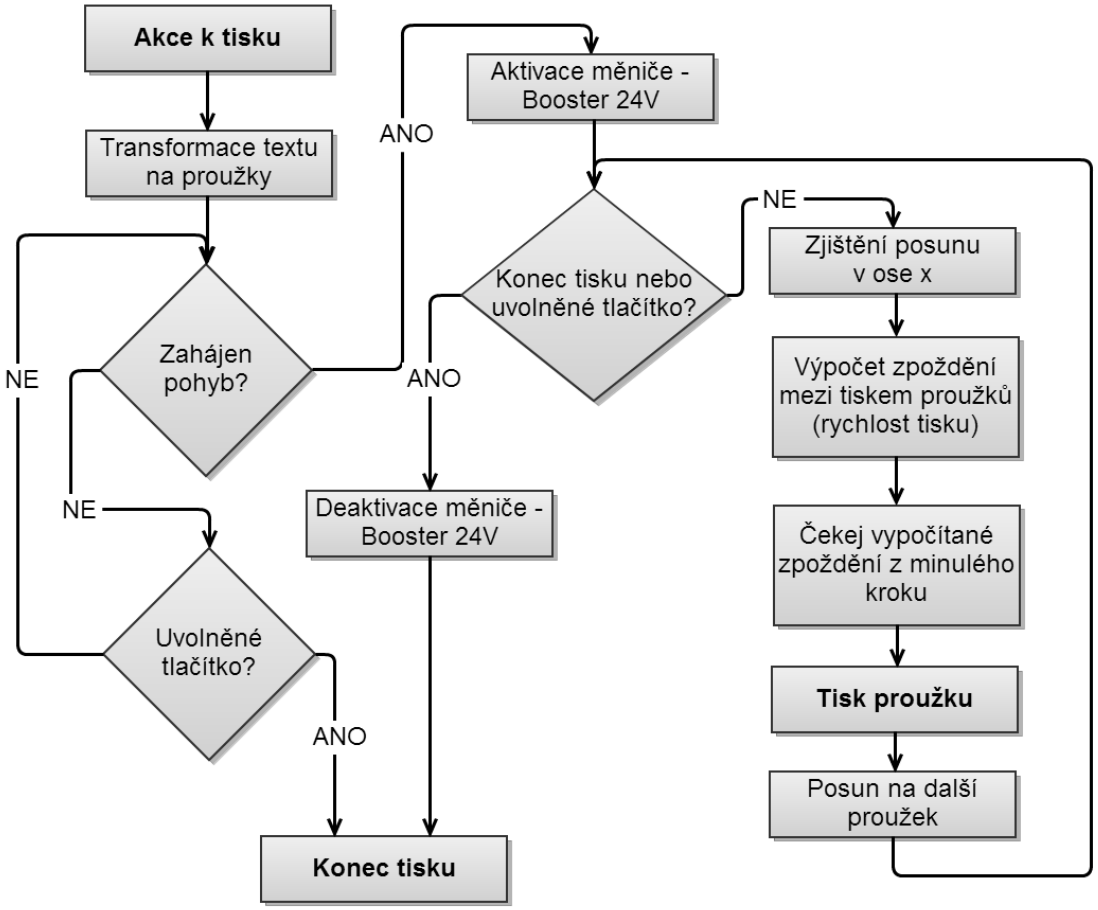
Nyní je připravena sada proužků k tisku a čeká se pouze na zahájení tisku. Tisk je zahájen stiskem tlačítka *OK*. Aby nedošlo k deformaci tištěného textu, je využito dat získaných ze senzoru pohybu.

Po několika experimentech bylo zjištěno, není vhodné zahajovat tisk okamžitě, protože než se s razítkem začne pohybovat, razítko udělá inkoustovou kaňku. Z těchto důvodů byla implementována funkce, která čeká tak dlouho, dokud se s razítkem nebude pohybovat.

Po detekci pohybu se zjistí rychlost pohybu zavoláním funkce
`dx = (signed char)OPTICAM_readRegister(0x03);`,
která vrací dráhu za dobu od posledního čtení registru. Protože senzor nedává příliš přesné výsledky, bylo nutné implementovat velmi jednoduchý filtr, který počítá novou hodnotu rychlosti. Byl navržen filtr, který počítá novou hodnotu rychlost v poměru 3:1 (stará vypočítaná rychlost : aktuální rychlost). Tím jsou špičky ve čtených datech vyfiltrovány. I přes jednoduchost výpočtu by takový výpočet byl na mikrokontroléru výpočetně náročný a výrazně by ovlivnil maximální rychlost tisku, proto je tento výpočet realizován pomocí bitových operací.

Dále je nutné převést informace o rychlosti na zpoždění tisku jednotlivých proužků. Jedná se o nepřímou úměru, jelikož čím větší rychlost tisku, tím menši zpoždění a čím menší rychlost, tím větší zpoždění. Bylo zjištěno, že převod pomocí výpočtu taktéž značně ovlivňuje rychlost tisku, proto byla implementována takzvaná Look-up tabulka, jejímž indexem je rychlost a hodnotou je požadované zpoždění.

Proužek textu je předán funkci `spray_ink(strip)` k vytištění. Ověří se, zda se nejedná o proužek, který ukončuje tisk. Pokud ano, tisk se ukončí. Na obrázku 9.7 je zobrazen diagram procedury tisku.



Obrázek 9.7: Vývojový diagram tiskové procedury.

9.2.5 Implementace volby textu

Razítko vždy pracuje s textem, který je zvolený jako aktivní. To proto, aby se eliminovala akce neustálého výběru textu. K tomu byla implementována funkce k volbě textu, se kterým bude razítko pracovat.

V první fázi se jednoduše načte první text z paměti EEPROM. Na displeji se zobrazí číslo textu a zda je aktivní nebo ne.

Stiskne-li uživatel jedno z tlačítek ↑, nebo ↓, načte se z paměti další text a zobrazí se na displeji včetně čísla textu a jeho aktivity. Současně se posune i kurzor na další text, který je na displeji zobrazený jako podtržené číslo textu. Takto se dá v seznamu textů postupně listovat, dokud nebude stisknuto tlačítko ←, což zapřičiní návrat bez jakékoliv změny, nebo stisknuto tlačítko →, které zajistí změnu aktivity textu zobrazeného pod kurzorem.

Aktivita textu je změněna jak paměti RAM, tak současně ve vyhrazené konfigurační paměti EEPROM, která uchovává volbu i po odpojení napájení.

9.2.6 Implementace editace textu

Funkčnost razítka by byla splněna i bez možnosti editace textu přímo v razítku, ovšem je-li potřeba natisknout nějaký velmi jednoduchý text přímo v terénu, je tato funkce velmi přínosná.

Bohužel razítko neobsahuje plnohodnotnou klávesnici, proto bude třeba klávesnici nějak simulovat. Jsou k dispozici tlačítka ↑, ↓, ←, → a tlačítko *OK*. P Editace textu se ovládá tímto způsobem:

- ← - Posun kurzoru vlevo.
- → - Posun kurzoru vpravo.
- ↑ - Změna znaku +1 (dle ascii tabulky).
- ↓ - Změna znaku -1 (dle ascii tabulky).
- *OK* - Uložení editace do EEPROM.

Narozdíl od předchozí funkce volby textu je tato implementace daleko komplikovanější. Protože máme zvolený aktivní text, je po startu funkce možné načíst text z paměti EEPROM do paměti RAM. Není-li v paměti EEPROM uložen žádný text, je k editaci připraven prázdný řetězec.

Stiskem tlačítek ← nebo → se posouvá kurzor v zobrazovaném řetězci. Je-li potřeba původní řetězec rozšířit, je po stisku tlačítka → přialokováno místo v paměti RAM pro uložení dalšího znaku. Výchozí hodnota nově alokovaného znaku je `0x00`, takže dokud nebude změněna, jedná se o konec řetězce.

Změna znaku se provádí stiskem tlačítka ↑ resp. ↓. Touto akcí je vždy vybraný znak inkrementován resp. dekrementován dle kódové tabulky ASCII. Změna je prozatím aktualizována pouze v paměti RAM a zobrazena na LCD displeji. Protože editace textu tímto způsobem není příliš pohodlná, bylo snahou editaci usnadnit. Dlouhodobý stisk tlačítka simuluje několikanásobné stisknutí. Tím je zajištěno pohodlné listování seznamem znaků. Protože editace textu přímo v razítku je chápáno jako nouzová možnost změny textu, je editace omezena pouze na znaky ASCII.

Stiskem tlačítka *OK* je editovaný text uložený v paměti RAM zapsán do paměti EEPROM. Návrat z funkce bez provedení změn není prozatím implementován.

9.2.7 Implementace smazání textu

Tato část je velmi jednoduchá. Opět je počítáno s tím, že je aktivní nějaký text. Tato funkce přečte číslo textu (adresu v EEPROM paměti) z konfigurační struktury a následně je proveden zápis `0xFF` do paměti EEPROM v celém vyhrazeném bloku, čímž je aktivní text kompletně vymazán.

Funkce je velmi jednoduchá a není nutné ji dále diskutovat.

9.2.8 Implementace komunikace s obslužnou aplikací InkStampPC

Pro komunikaci s počítačem bude použita sériová linka USART. Navenek se však razítko bude připojovat přes USB a převodník FT232R, interní komunikace bude probíhat sériově. Sériová komunikace byla zvolena na základě její jednoduchosti a také proto, že není vyžadována vysoká rychlost přenosu. Budou se přenášet pouze textová data.

Po zavolání funkce, která slouží pro komunikaci s obslužnou aplikací, je aktivován přijímač sériové linky. Následně se program dostane do nekonečné smyčky, která čeká na příchozí data. Smyčka může být ukončena stiskem tlačítka ←. Zpracování dat je detailně popsáno v následujících řádcích.

Protokol komunikace

Základem jakéhokoliv přenosu dat je vhodný formát. To znamená, že přenášená data mají nějaký předem daný tvar a díky tomu je umožněná jednoduchá extrakce informací. V praxi se taková pravidla, která určují pravidla komunikace, nazývají *protokol komunikace*.

Komunikaci zahajuje vždy Master - tedy aplikace InkStampPC. Razítko nikdy neiniciuje komunikaci a pouze čeká na příkazy. Protože přenos bude realizovaný na úrovni bajtů, je žádoucí komunikační protokol taktéž realizovat na úrovni bajtů.

Idea protokolu spočívala v několika otázkách:

- Jak předat příkaz?
- Jak předat číslo textu (adresu v paměti) ?
- Jak předat text?
- Jak poznat konec komunikace?

První informaci, kterou razítko potřebuje znát, je požadovaný příkaz. Tento příkaz bude přenášen v prvním bajtu. Příkaz může vyžadovat zaslání dodatečných informací, například zaslání adresy nebo čísla textu. Dodatečné informace k příkazu budou zasílány bezprostředně po zaslání příkazu. Po přenosu příkazu včetně dodatečných informací se mohou vyskytnout zasílaná data nebo případně terminální znak `0x00` značící konec přenosu. Na obrázku 9.8 je možné vidět formát přenášených dat. Dle charakteru příkazu je položka data nepovinná.

	Příkaz	Adresa	Data	Term.znak
Velikost:	1 B	1 B	0 - 256 B	1 B

Obrázek 9.8: Formát komunikace přes USART..

Prozatím jsou implementovány tři příkazy. Příkaz čtení textu, příkaz k zápisu textu a příkaz smazání textu.

V případě, že příjde příkaz požadující čtení textu, je poslán příkaz čtení, následně pozice v paměti a ukončující znak `0x00`. Razítko příkaz zpracuje a začne s odesíláním zvoleného textu, který je taktéž ukončen terminálním znakem `0x00`.

Je-li požadován zápis, tvar požadavku vypadá následovně:
Příkaz k zápisu, pozice v paměti na kterou má být proveden zápis, textový obsah a konec přenosu je opět identifikován znakem `0x00`.

Požaduje-li uživatel smazání textu, je zaslán příkaz k vymazání, pozice v paměti a ukončující znak `0x00`. Po příjetí těchto tří bajtů razítko provede příkaz smazání textu v daném místě paměti EEPROM.

9.2.9 Implementace nastavení

Další částí jádra razítka jsou možnosti nastavení. Jsou to jednoduché funkce, které pouze změní konfigurační strukturu razítka, případně změnu zapíší do paměti EEPROM. Byly implementovány prozatím tři funkce razítka, které jsou popsány dále.

Nastavení podsvícení klávesnice a LCD

Protože je razítko poháněné z baterií, je snahou co nejvíce omezit spotřebu elektrické energie. Dost výrazný podíl na spotřebě energie má osvětlení. Ve dne, kdy je dostatek venkovního osvětlení, není třeba podsvěcovat ovládací prvky razítka. Funkce prozatím dokáže pouze vypnout podsvícení LCD displeje nebo klávesnice, jelikož ke změně intenzity není přizpůsoben hardware prototypu. Hardware popisovaný v technické zprávě 7 je již připraven pro řízení intenzity podsvícení ovládacích prvku razítka pomocí PWM[3].

Procedura řízení podsvícení pouze využívá funkce z knihovny *power.h*, pomocí kterých řídí tranzistory k daným periferiím, tedy k LED diodám podsvícení. Při změně stavu je změna zapsána do konfigurační paměti EEPROM a konfigurační struktury v paměti RAM. Po restartu razítka tak změny zůstanou zachovány.

Obnovení standardního nastavení

Protože se jedná o prototyp, který prochází testováním, je možný výskyt chyby v programu, které mohou způsobit neočekávaný zápis nesmyslných hodnot do paměti EEPROM. Tato funkce vymaže kompletně celou paměť EEPROM, včetně textů a konfigurace.

9.2.10 Implementace vývojových nástrojů

Protože vývoj razítka neustále pokračuje, jsou pro potřeby vývoje implementovány funkce, které by mohly pomoci při ladění nebo usnadnit vývoj. Pokud by razítko bylo určeno k prodeji, tyto funkce by zřejmě byly odstraněny. Prozatím byly implementovány dvě funkce, které jsou popsány v následujících řádcích.

Načtení výchozích textů

Při vývoji jádra firmwaru se dost často docházelo k porušení dat v paměti EEPROM. Aby se porušené texty nemusely opravovat ručně, byla implementována funkce, která načte do paměti EEPROM nějaké výchozí texty, které slouží především k testování. Výchozí texty se skládají z těchto sad a jsou uloženy v programové paměti:

1. Jméno autora.
2. Kontakt autora.
3. Malá a velká abeceda.
4. Diakritické znaky českého jazyka.
5. Číslice a ostatní znaky.

[3] PWM - Pulse Width Modulation

Test snímače pohybu ADNS2610

Před implementací tisku řízeného daty ze senzoru bylo nutné mít implementované a otestované získávání dat přímo ze senzoru. Proto byla implementována funkce, která zobrazuje na LCD displeji souřadnice v ose x a v ose Y.

Tato funkce pouze získá hodnoty z obou registrů, pomocí funkcí
`(signed char)OPTICAM_readRegister(address);`,
nesoucí přírůstek dráhy v každé ose, který přičte k celkové dráze a výsledek zobrazí na LCD displej.

Tato funkce by mohla být ve finální verzi razítka ponechána.

Kapitola 10

Implementace obslužné aplikace InkStampPC

Tato kapitola se zabývá vývojem obslužné aplikace pro osobní počítač. Aplikace má usnadnit obsluhu razítka a zjednodušit editaci textů. V následujících podkapitolách se diskutují použité nástroje, použité knihovny třetích stran a výsledná implementace.

10.1 Vývojové prostředí

V dnešní době se rozšiřuje operační systém na bázi Linuxu, ovšem většina uživatelů stále používá Windows OS od firmy Microsoft. Cílem však bylo navrhnout aplikaci, která bude přenositelná mezi platformami Windows a Linux a současně bude poskytovat uživatelské rozhraní.

Existuje mnoho knihoven pro práci s okny. Základním požadavek je podpora více platforem a programovací jazyk C/C++. Tímto se výběr velmi zredukoval a proto byla vybrána knihovna Qt [18], od finské Nokie, na základě pozitivních zkušeností autora.

Qt je jedna z nejpopulárnějších multiplatformních knihoven pro vytváření aplikací s grafickým uživatelským rozhraním. Qt toolkit byl vytvořen v roce 1999 společností Trolltech, která jej v roce 2008 prodala firmě Nokia.

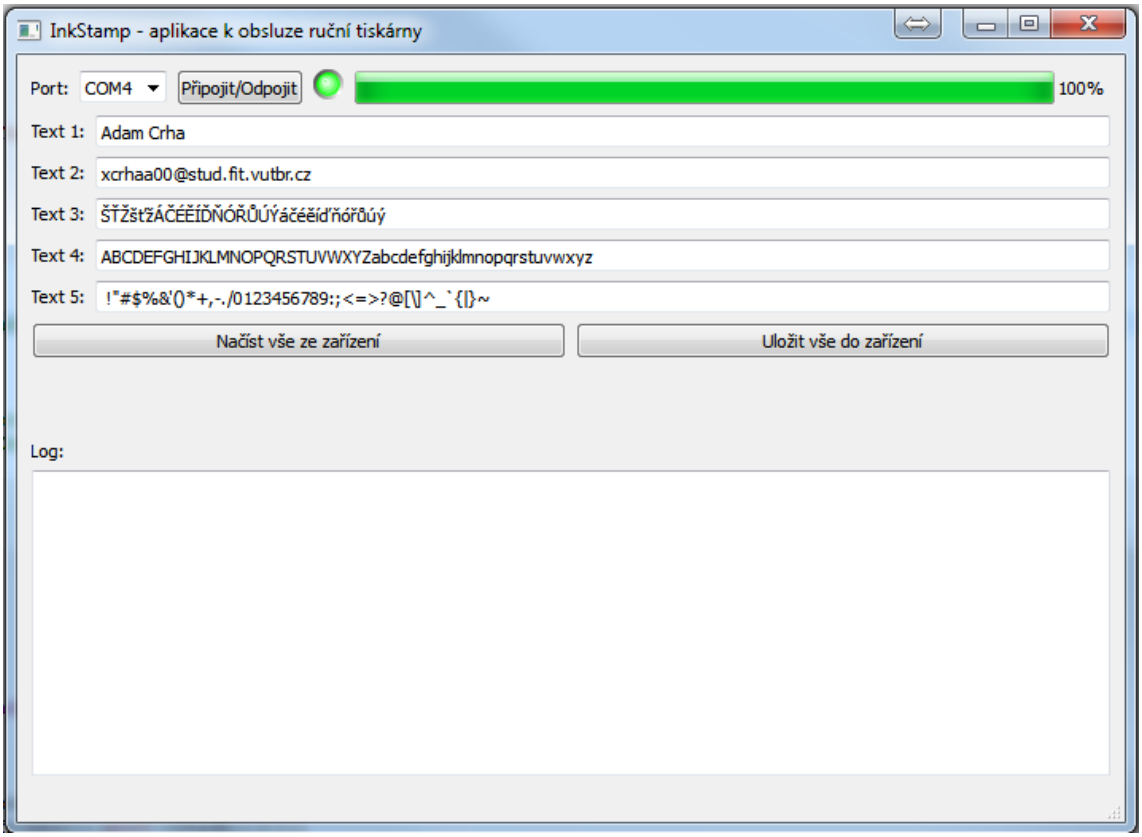
Qt je knihovna programovacího jazyka C++, podporuje lokalizaci aplikací a také SQL, zpracování XML, správu vláken, přístup k souborům, práci s grafikou a multimédii. Velkou výhodou Qt je velmi přehledně zpracovaná dokumentace a také vývojové programy Qt Creator nebo Qt Designer. Aplikace vytvořené pro grafické uživatelské prostředí používají nativní vzhled operačního systému, takže vyvinuté aplikace se vždy přizpůsobí používanému prostředí.

10.2 Tvorba uživatelského rozhraní

Uživatelské rozhraní je vytvořeno v aplikaci Qt Designer. Rozhraní musí být velmi jednoduché a použitelné takzvaně „bez návodu“. Jednoduchost aplikace usnadní vývoj, omezí možnost vzniku chyb a současně se stane efektivním nástrojem ke správě razítka.

Na obrázku je zobrazené okno obslužné aplikace. Vlevo je možné vidět list-box určený k výběru portu, signalizační „světélko“ informující připojený port, tlačítko *připojit/odpojit*.

Nesmí chybět kolonky k editaci textu a tlačítka, které provedou načtení, či odeslání textů. Dole je možné spatřit logovací okénko, které informuje uživatele o provedených akcích.



Obrázek 10.1: Uživatelské rozhraní obslužné aplikace.

10.3 Externí knihovna pro přístup k sériovému portu

Razítko v sobě obsahuje převodník z USB na RS232, tedy na sériovou linku. Razítko tak bude navenek připojené přes USB, ovšem uvnitř razítka je obsažen čip, který převod z USB na RS232 realizuje. Protože celá komunikace bude probíhat přes sériový port (pomocí FT232R čipu), je nutné implementovat ovladač sériového portu. Implementace multiplatformního ovladače sériového portu není triviálním úkolem kvůli specifickým vlastnostem obou operačních systémů a výrazně tak přesahuje rozsah práce. Proto byla použita externí knihovna, která je převzata ze zdroje [17] bez jakýchkoliv úprav. Tato knihovna je distribuována pod licencí MIT a implementuje kompletní multiplatformní ovladač sériového portu, který plně vyhovuje potřebám tohoto projektu.

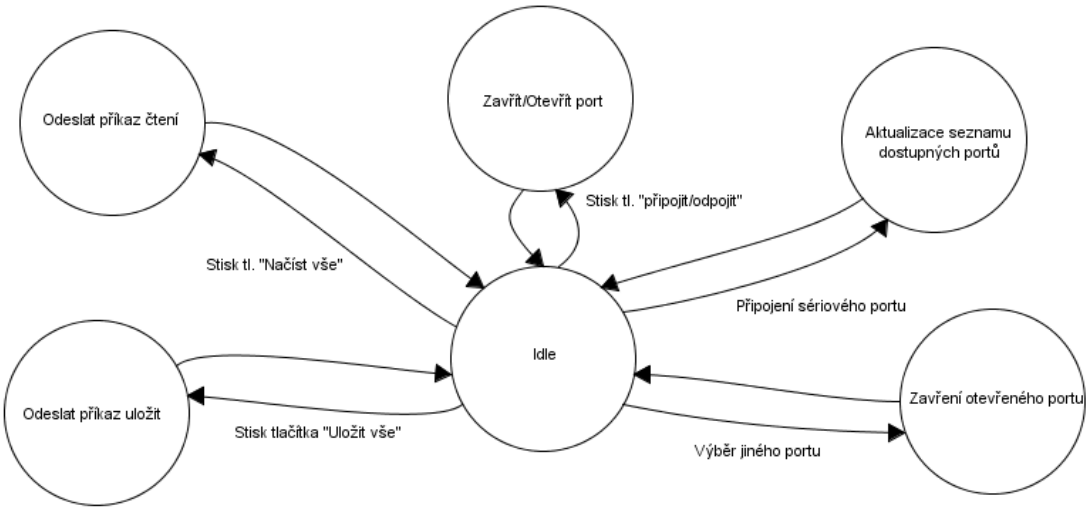
10.4 Implementace

Implementace obslužné aplikace je vytvořena pomocí Qt Creatoru.

V hlavní funkci je zavolán konstruktor při vytvoření instance hlavního okna. Konstruktor obsahuje sestavení uživatelského rozhraní, naplnění list-boxu dostupnými sériovými

porty, nastavení parametrů sériové komunikace, vytvoření instance třídy portu a propojení takzvaných signálů se sloty. Sloty je možné chápat jako funkce, které se vykonají při daném signálu (vyvolání události).

Uvádět implementační detaily je zbytečné, jelikož zdrojový kód je velmi dobře komentovaný a není příliš obsáhlý. Z těchto důvodů je zde uveden stavový diagram 10.2 popisující akce, které mohou být vyvolány určitou událostí.



Obrázek 10.2: Konečný automat zobrazující akce na události aplikace.

Za zmínku však stojí zmínit přístup k multiplatformnímu kódování znaků.

Přenositelnost mezi platformami a kódování znaků

Jak jistě všichni víme, Windows a Linux používají rozdílná kódování znaků. Bylo nutné se rozhodnout, jaké kódování zvolit. Kódování ISO-8859-2 je standardní na systémech Linux, kdežto český Windows pracuje s kódováním CP1250. Protože je předpokládáno, že razítko budou využívat zejména uživatelé v kanceláři, kde v 99% běží operační systém Windows, bylo zvoleno kódování CP1250. Jak se však zachovat, je-li aplikace přeložená a spuštěná na systému Linux?

Tato kolize kódování obou operačních systémů byla vyřešena pomocí tříd *QTextDecoder* a *QTextCoder*, obsažené v knihovně Qt. Tyto třídy vzájemně dokáží překonvertovat libovolně kódovaný text do UTF[1] a naopak.

V aplikaci je tedy text před odesláním do zařízení překonvertován na kódování CP1250 a při čtení z razítka je z CP1250 překonvertován do UTF.

Tím, že se při odeslání a čtení text převede do požadovaného kódování, je zajištěna přenositelnost aplikace mezi více operačními systémy.

[1] UTF - UCS Transformation Format

Kapitola 11

Testování, problémy a další vývoj

Tato kapitola popisuje průběh testování, problémy a možnosti rozšíření.

11.1 Testování

Systém byl v průběhu vývoje podrobně testován. Nejprve byly implementovány knihovny pro práci s jednotlivými periferiemi, které byly testovány samostatně. Protože vestavěný systém nemá žádný standardní výstup, bylo nejdříve zprovozněno rozhraní USART pro připojení k terminálu. Tato funkce posloužila k výpisu textových informací. Jako terminálová aplikace byl použit software Herkules [26] od firmy HW-group. Tento software je plně kompatibilní s Windows 7. Aplikace obsahuje sériový terminál, a mnoho dalších nástrojů.

Dále byla implementována knihovna pro řízení LCD displeje, pomocí kterého bylo možné opět zobrazovat stav programu. LCD displej a rozhraní USART sloužili k ladění prakticky po celou dobu vývoje firmwaru. Po připravení testovacího rozhraní byly implementovány knihovny pro další periferie, například pro snímač pohybu, řízení tiskové hlavy apod., které byly podrobeny detailním testům.

Velmi obtížné bylo ladění ovladače tiskové hlavy, jelikož chyba v implementaci ovladače znamenala zničení tiskové hlavy. Proto byla implementace otestována pomocí osciloskopu bez připojené tiskové hlavy.

Testování jednotlivých funkcí jádra probíhalo podobně jako testování funkčnosti jednotlivých periferií. Každá funkce v jádru byla samostatně otestována bez závislosti na neotestovaných funkcí.

Po dokončení implementace alfaverze byl produkt předán osobě, která s ovládáním razítka nemá žádné zkušenosti. Je vysoká pravděpodobnost, že neznalá osoba způsobí neočekávaný zásah do firmwaru razítka, který není ošetřen.

Tímto způsobem testování se také podařilo získat zpětnou vazbu od uživatele v jaké míře je pohodlná navigace v menu a zda je ovládání intuitivní, tedy zda je možné razítko použít i bez návodu.

11.2 Technické problémy a obtíže při realizaci

V průběhu vývoje se vyskytlo mnoho problémů, které vyšly na povrch až ve fázi realizace a implementace firmware. Jednalo se o drobné i závažné chyby v návrhu a jednotlivé nedostatky jsou popsány v následujících řádcích. Problémy byly z velké části způsobeny

chybou HW návrhu, proto musel být návrh opraven. V technické zprávě 7 je diskutovaný již opravený návrh, avšak výstupní prototyp stále nese tyto chyby a problémy.

11.2.1 Problémy při realizaci hardware

Nejvíce problému bylo způsobeno chybou návrhu hardwaru. Jednotlivé chyby jsou dále zmíněny.

- **Prohozené signály sériového komunikačního rozhraní:** Při implementaci firmware bylo na základě testování zjištěno, že byly z důvodů nepozornosti návrhu prohozené vodiče *RX* a *TX*, sloužící k sériové komunikaci mezi čipem FT232R a mikrokontrolérem ATMega2560. Chyba byla opravena.
- **FT232R - Signalizační diody přenosu:** Opět kvůli nepozornému studiu manuálu k součástce *FT232R*, byla způsobena chyba v návrhu hardwaru. Diody, které signalizují přenos, byly navrženy jako spínané proti zemi, ovšem integrovaný obvod diody spíná proti napájecímu napětí 5V. Tato chyba byla taktéž opravena.
- **Každá periferie - vlastní řízené napájení:** V původním návrhu je navrženo spínání přívodu napájení LCD displeje a optického snímače společné. I přes to, že optický snímač umožňuje režim spánku, stále odebírá malé množství energie. Protože je snahou opravdu co nejvíce omezit spotřebu, je vhodné optický snímač odpojit úplně. Přitom je žádoucí, aby LCD displej neustále byl napájen. Proto se provedlo rozdělení řízení napájení každé periferie zvlášť.
- **Intenzita podsvícení:** Bylo zjištěno, že intenzita podsvícení displeje a klávesnice je příliš vysoká, což je zbytečné a současně je tak spotřebováváno mnoho energie. Žádoucí je, aby intenzita svítivosti měla dostatečnou úroveň k pohodlnému čtení displeje, či rozpoznání symbolů na klávesnici.

  Původní návrh s tímto nepočítal a je tak možné podsvícení pouze zapnout nebo vypnout. Tento nedostatek byl v dalším návrhu opraven a to tak, že řídící kanály tranzistorů byly zapojeny k pinu, který umožňuje PWM.
- **Integrace modulu reálného času:** V původním návrhu bylo plánováno připojení RTC modulu k možnosti tisku aktuálního data a času. Později však bylo zjištěno, že hotové RTC moduly jsou velmi drahé, což znamenalo potřebu vlastního RTC modulu integrovaného v základní desce. Dodatečně byl tedy na desku plošných spojů integrován RTC modul.

11.2.2 Problémy při implementaci firmware

Při implementaci a testování se vyskytl pouze jeden závažný problém, který stojí za zveřejnění. Jeho oprava naštěstí nebyla nijak nákladná.

- **Přemostění napájení periferií:** Tento problém byl zpočátku považován za chybu návrhu hardwaru a nebylo lehké najít příčinu problému.

  Razítko je navrženo s ohledem na úsporu spotřebovávané energie. Proto jsou realizovány funkce, které dokáží odpojit periferii od napájecího napětí pomocí tranzistoru.

  Po implementaci řízení napájení však spínání nepracovalo správně a chyba přisuzována chybě v návrhu hardwaru.

Postupným hledáním problému, který spočíval v odpojení veškerých periferií a postupném připojování, bylo zjištěno, že spínání napájení přestane správně pracovat po připojení datových vodičů LCD displeje. Po detailním prozkoumání problému se zjistilo, že napájení periferií se propaguje přes datové vodiče LCD displeje.

Tento problém byl eliminován tak, že před odpojením napájení každé periferie je nutné v programu mikrokontroléru nastavit datové vodiče do logické 0.

11.3 Rozdíly ve verzích

Jak je zmíněno v předchozí kapitole, výstupní prototyp obsahuje velké množství chyb a nedostatků. Tyto nedostatky však byly opraveny a zde je výčet jednotlivých změn mezi verzemi, které opravy návrhu vyžadovaly.

- Řídící piny LCD displeje přesunuty z *PH4-6* na *PH0-2*
- Datové piny optického snímače přesunuty z *PE2-3* na *PB4-5*
- Přesunuty a přidány piny k řízení napájení:
  - PE2 - Booster.
  - PE3 - PWM podsvícení LCD displeje.
  - PE4 - PWM podsvícení klávesnice.
  - PE5 - Přívod napájení LCD displeje.
  - PE6 - Přívod napájení Optického snímače.
  - PE7 - Přívod napájení externí EEPROM paměti.

11.4 Možnosti rozšíření a plány do budoucna.

Možností rozšíření razítka je hned několik a mnoho dalších nápadů neustále přichází.

V první fázi je potřeba dořešit nedostatky, které použití razítka omezují. Následně je pak možné uvažovat nad rozšiřujícími funkcemi.

Hlavním cílem do budoucna je minimalizace rozměrů razítka.

- Možné rozšíření do budoucna:
  - Zobrazit české znaky na LCD displeji.
  - Zdokonalit komunikační protokol (chybí potvrzující odpověď).
  - Implementovat intenzitu podsvícení pomocí PWM.
  - Implementovat podporu RTC.
  - Integrovat bluetooth modul pro obsluhu z mobilního telefonu/tabletu/notebooku.
  - Tisk bitmapy.

Kapitola 12

Závěr

V této diplomové práci je diskutována problematika vestavěných systémů a možností ručního tisku. Práce je rozdělena na dvě části, teoretickou část a část, která popisuje jednotlivé kroky sestavení razítka od počátku návrhu až po fyzickou realizaci včetně vývoje firmwaru a softwaru obslužné aplikace.

V první části je čtenář seznámen s principy inkoustového tisku, kde jsou popsány nejrozšířenější technologie. Je diskutován princip kontinuálního tisku a princip tisku on-demand, který se dělí na další dvě technologie, termální a piezoelektrickou. Teoretická část pokračuje rešerší způsobů snímání pohybu. Jsou diskutovány mechanické enkodéry na principu optické závory, magnetické enkodéry využívající Hallova jevu a optické kamerové snímače. Posledním blokem teoretické částí je stručný přehled výrobců mikroprocesorů a polovodičových součástek. Jsou představeny a srovnány čtyři dominanty na trhu: Freescale, Atmel, Microchip a STMicroelectronics.

V druhé části, zaměřené na realizaci razítka, je nastíněn koncept inteligentního programovatelného razítka. Jsou popsány jednotlivé komponenty a zdůvodnění jejich volby. Pokračuje kapitola, která se zabývá návrhem konstrukce, kterou následuje kapitola popisující fyzické sestavení razítka. Další kapitola se věnuje návrhu hardwaru a desce plošných spojů. Jsou detailně rozebírány jednotlivé části schémat zapojení.

Třetí část technické zprávy se věnuje návrhu a implementaci firmwaru mikrokontroléru. Jsou zde diskutovány knihovny periferií, které bylo nutné implementovat, i samotné jádro firmwaru. Technická zpráva zmínila i popis návrhu a implementaci obslužné aplikace pro osobní počítač.

V poslední části této práce je popsáno testování, jsou zmíněny problémy, které se vyskytly v průběhu návrhu, a jsou diskutovány možnosti rozšíření do budoucna. Příloha B obsahuje stručný návod k použití razítka.

Výsledkem práce je funkční a otestovaný prototyp inteligentního programovatelného razítka, který splňuje požadavky zadání. Závěrem práce na projektu byla fyzická realizace razítka, po které se ukázaly výhody i nedostatky návrhu. Po odstranění nedostatků se podařilo zkonstruovat amatérské zařízení v ceně zhruba 3500 Kč s DPH (viz příloha C), které v této podobě nebylo prozatím nikým konstruováno.

V příloze A.1 a na přiloženém CD jsou k dispozici fotografie a obrázky zhotoveného inteligentního programovatelného razítka.

Literatura

[1] Arduino. [Online; navštíveno 30.12.2012].
URL http://www.arduino.cc

[2] *ATMEGA2560 Data Sheet - 8-bit Atmel Microcontroller with 256K Bytes In-System Programmable Flash.* [Online; navštíveno 12.10.2012].
URL http://www.atmel.com/Images/doc2549.pdf

[3] Atmel. [Online; navštíveno 30.12.2012].
URL http://atmel.com

[4] Atollic. [Online; navštíveno 30.12.2012].
URL http://atollic.com

[5] Austria micro systems. [Online; navštíveno 30.12.2012].
URL http://www.ams.com

[6] Avago technologies. [Online; navštíveno 30.12.2012].
URL http://www.avagotech.com

[7] CadSoft online. [Online; navštíveno 1.4.2013].
URL http://cadsoft.de

[8] *Datasheet 7805, Stabilizátor napětí.* [Online; navštíveno 30.12.2012].
URL http://www.gme.cz/dokumentace/934/934-051/dsh.934-051.1.pdf

[9] *Datasheet ADNS2610, Optický senzor.* [Online; navštíveno 30.12.2012].
URL http://www.avagotech.com/docs/AV02-1184EN

[10] *Datasheet DS1307, RTC čip.* [Online; navštíveno 30.12.2012].
URL http://www.gme.cz/dokumentace/959/959-324/dsh.959-324.1.pdf

[11] *Datasheet FT232R, USB - USART převodník.* [Online; navštíveno 30.11.2012].
URL
http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[12] *Datasheet k LT1930, DC-DC Měnič.* [Online; navštíveno 30.12.2012].
URL http://cds.linear.com/docs/en/datasheet/1930f.pdf

[13] *Datasheet LCD, LCD MIDAS 16x2, MC21605B6WD-BNMLW.* [Online; navštíveno 20.12.2012].
URL http://www.midascomponents.co.uk

[14] Freescale Semiconductor. [Online; navštíveno 30.12.2012].
URL http://freescale.com

[15] Hammond manufacturing. [Online; navštíveno 1.4.2013].
URL http://www.hammondmfg.com

[16] Linear technology. [Online; navštíveno 30.12.2012].
URL http://www.linear.com

[17] Multiplatformní ovladač sériového portu. [Online; navštíveno 1.3.2013].
URL https://code.google.com/p/qextserialport/

[18] Qt project. [Online; navštíveno 1.3.2013].
URL http://qt-project.org

[19] Tvorba menu pro mikrokontroléry AVR. [Online; navštíveno 1.3.2013].
URL http://winavr.scienceprog.com/example-avr-projects/avr-lcd-menu-routine.html

[20] Asix: [Online; navštíveno 10.10.2012].
URL http://www.asix.cz

[21] Atmel: ATMega2560. [Online; navštíveno 30.12.2012].
URL http://www.atmel.com/devices/atmega2560.aspx

[22] C´Levis, N.: InkShield projekt. [Online; navštíveno 30.12.2012].
URL http://nicholasclewis.com/projects/inkshield

[23] Dzik, P.: Hardware inkoustových tiskáren. 2007, [Online; navštíveno 30.12.2012].
URL http://www.x-pozice.cz/clanky/hardware-inkoustovych-tiskaren.html

[24] Dzik, P.: Technologické principy inkoustového tisku. 2012, [Online; navštíveno 29.12.2012].
URL http://www.chempoint.cz/technologicke-principy-inkoustoveho-tisku

[25] Gilliland, M.: *Inkjet Applications*. Woodglen Press, 2005, iSBN 978-0972015936.

[26] Group, H.: Hercules terminal. [Online; navštíveno 15.2.2013].
URL http://www.hw-group.com/products/hercules/index_cz.html

[27] Kučera, J.: Inkoustový tisk včera, dnes a zítra. 2004, [Online; navštíveno 30.12.2012].
URL http://www.fi.muni.cz/usr/jkucera/pv109/2004/xmacuga.htm

[28] Müller, V.: Princip a historie inkoustových tiskáren. 2012, [Online; navštíveno 29.12.2012].
URL http://www.kopirky.com/princip-a-historie-inkoustovych-tiskaren

[29] Parallax: InkJet Development Kit. [Online; navštíveno 30.12.2012].
URL http://www.parallax.com/dl/docs/prod/robo/InkjetKitDocs-v1.0.pdf

[30] Vojáček, A.: Princip optických enkodérů polohy pro řízení motorů. 2006, [Online; navštíveno 30.12.2012].
URL http://automatizace.hw.cz/clanek/2006022801

[31] Wikipedia: Boost converter. [Online; navštíveno 30.12.2012].
URL http://en.wikipedia.org/wiki/Boost_converter

[32] Wikipedia: Hallův jev. [Online; navštíveno 30.12.2012].
URL http://cs.wikipedia.org/wiki/Hall%C5%AFv_jev

[33] Wikipedia: Optická myš. [Online; navštíveno 30.12.2012].
URL
http://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%A1_my%C5%A1

[34] Wikipedia: Procesor. [Online; navštíveno 30.12.2012].
URL http://cs.wikipedia.org/wiki/Procesor

[35] ZÁHLAVA, V.: *Návrh a konstrukce desek plošných spojů*. Praha, CZ: ben, první vydání, 2010, ISBN 978-80-7300-266-4.

Seznam příloh

A Výsledná podoba realizačního výstupu.

B Návod k použití.

C Přibližné náklady.

D Schémata zapojení.

E Osazení desek plošných spojů.

F Datový nosič se zdrojovými soubory a podklady pro konstrukci. Podrobný obsah je vypsán v souboru README v kořenovém adresáři CD.

Dodatek A

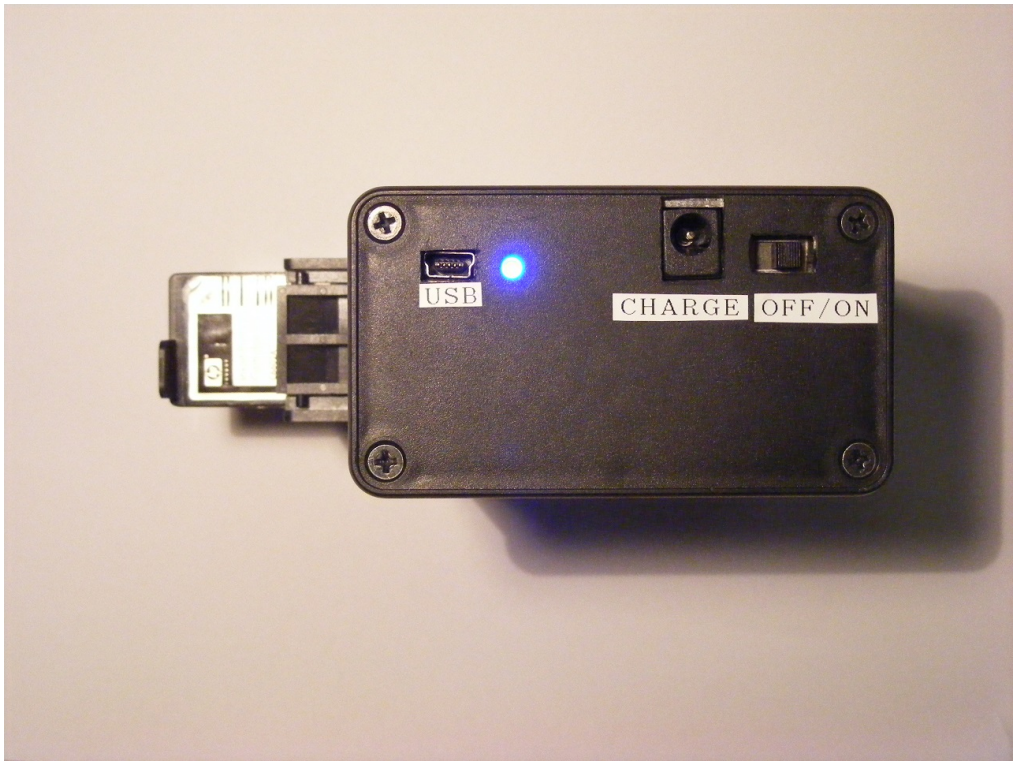
Výsledná podoba realizačního výstupu

Výsledný produkt je vyobrazen na obrázcích v příloze A.1. Nechybí také fotografie výstupu razítka, tedy natištěného textu.

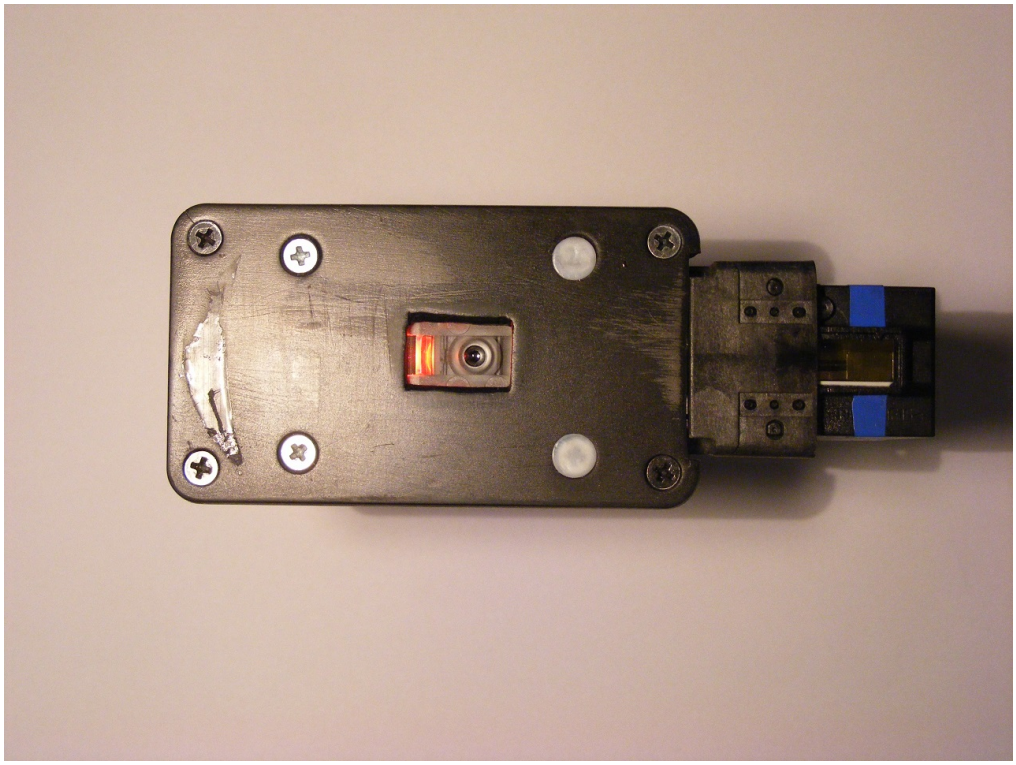
A.1 Fotografie prototypu

Obrázek A.1: Pohled na prototyp razítka zepředu.

Obrázek A.2: Pohled na prototyp razítka zezadu.

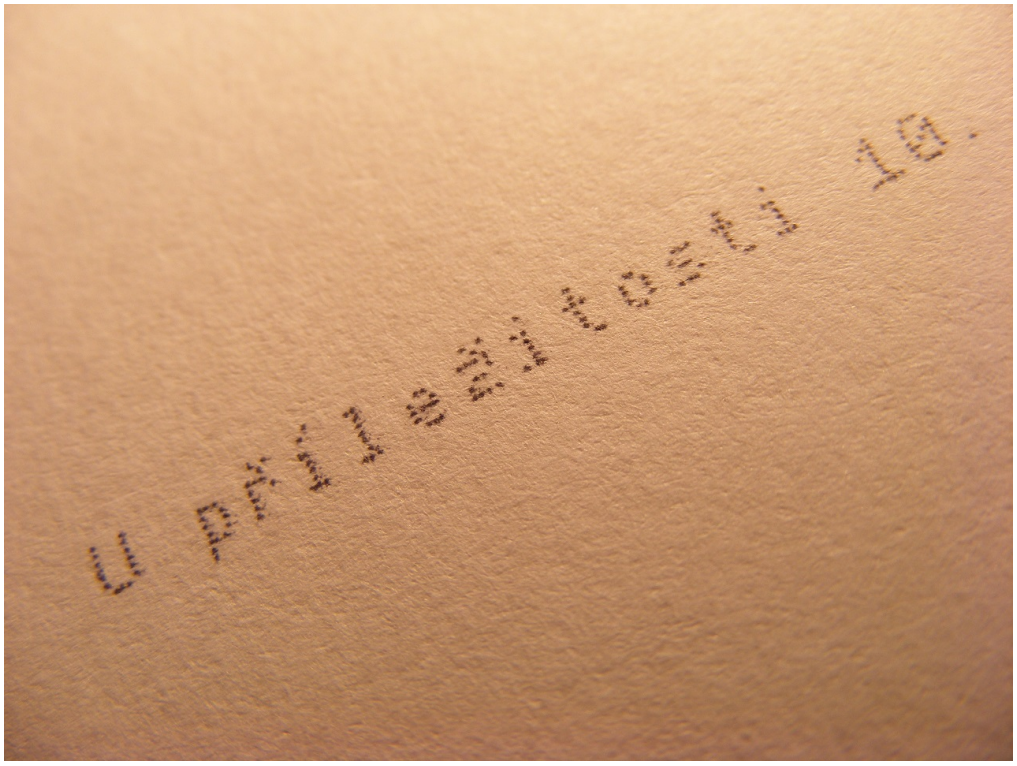


Obrázek A.3: Pohled na prototyp razítka shora.

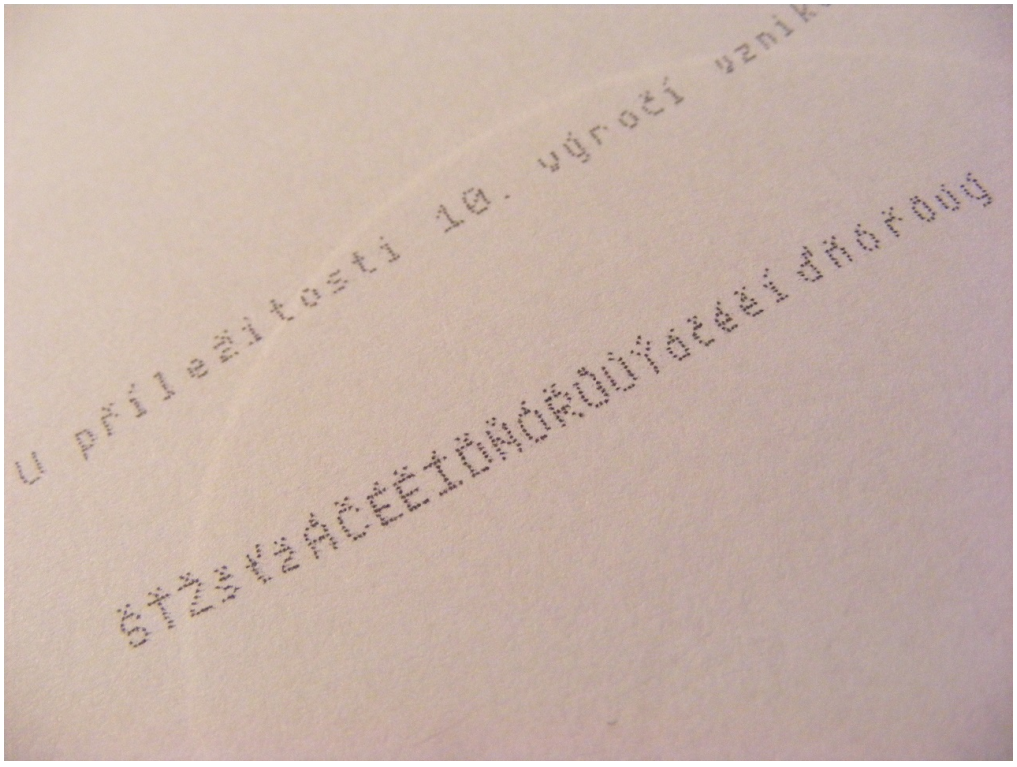


Obrázek A.4: Pohled na prototyp razítka zdola.

Obrázek A.5: Pohled na prototyp razítka.



Obrázek A.6: Makro fotografie vytištěného textu.



Obrázek A.7: Fotografie vytištěného textu.

Dodatek B

Návod k použití

Tato kapitola poskytuje stručný návod k použití razítka. Velmi důležitý je návod k použití, i přes to, že bylo snahou navrhnout ovládání takzvaně „bez návodu“.

B.1 Navigace v menu a funkce razítka

Na obrázku B.1 je možné vidět mapu menu sloužící k jednodušší orientaci.

Navigace v menu probíhá pomocí šipek ←, →, ↑ a ↓. Tlačítka ↑ a ↓ slouží k listování v dané úrovni menu. Tlačítko → slouží k zanoření do položky menu, případně k volbě funkce! Tlačítko ← slouží často k vynoření nebo pro návrat z funkce.

Funkce, které jsou ovládány odlišně, jsou popsány dále:

B.1.1 Tisk

K samotnému tisku je třeba určitá zručnost. Přiložte razítko k podložce a stiskněte tlačítko *OK*. Za stálého držení tlačítka *OK* začněte s razítkem pohybovat zprava doleva konstantní rychlostí. Tisk je ukončen uvolněním tlačítka *OK*, případně dokončením tisku.

B.1.2 Editace textu

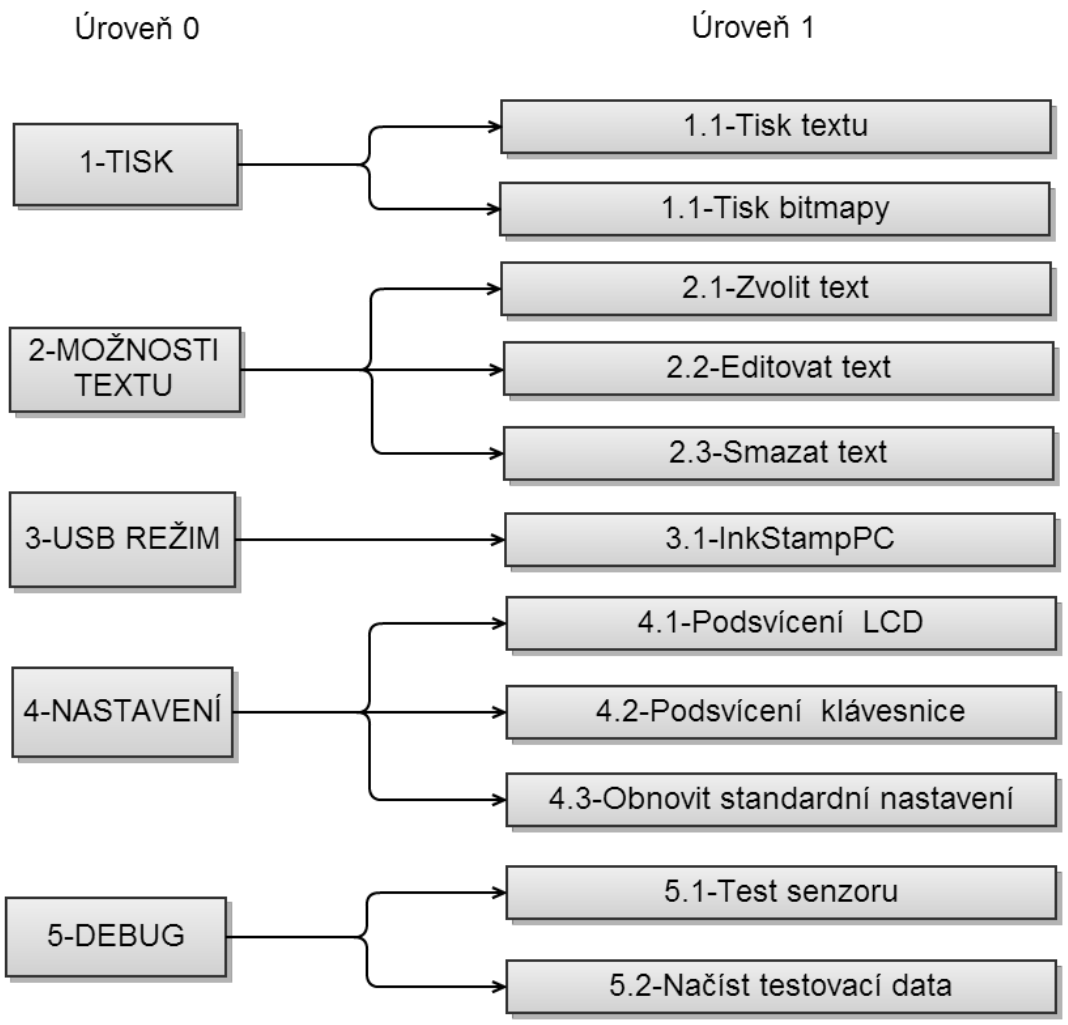
Editace textu má mírně odlišné ovládání. Tlačítky ← a → se posunuje kurzor v rámci editovaného textu. Tlačítka ↑ a → slouží ke změně znaku. Tlačítkem *OK* se text uloží a editace se ukončí.

B.2 Výměna tiskové hlavy

Výměna tiskové hlavy je velmi jednoduchá. Vypněte zařízení hlavním vypínačem. Nyní tahem za zobáček tiskové hlavy uvolníte tiskovou hlavu z držáku a je možné hlavu vyjmout. Vložení nové tiskové hlavy se provede opačným postupem. Nejdříve se uloží spodní strana tiskové hlavy do držáku a poté se zatlačí na zobáček, čímž dojde k zajištění hlavy v držáku.

B.3 Nabíjení

Jakmile razítko přestane vykazovat známky funkce - tedy nejde zapnout, je pravděpodobně vybitá baterie.



Obrázek B.1: Navigační mapa menu.

Hlavní vypínač uveďte do polohy *OFF*. Připojte nabíjecí konektor. Zahajte nabíjení speciální nabíječkou určenou k nabíjení Lithium-Iontových akumulátorů.

Vypínač v poloze OFF propojuje přímo nabíjecí konektor s kontakty baterie a tudíž proces nabíjení není nijak kontrolován. Viz 7.1

Dodatek C

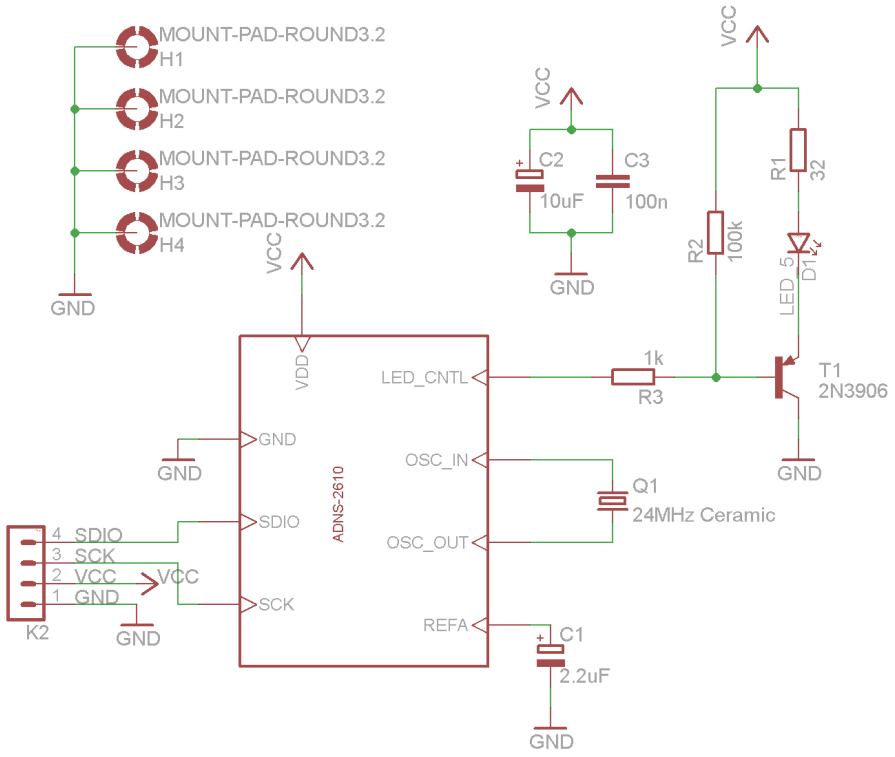
Přibližné náklady

V této příloze je uveden přibližný výčet nákladů na sestavení razítka platný k datu duben 2013:

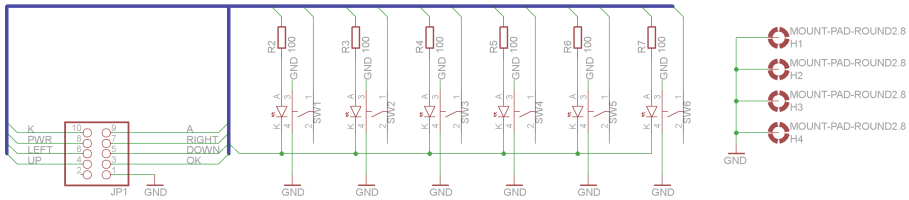
Mikrokontrolér ATmega2560	450 Kč
LCD displej 2x16	250 Kč
Klávesnice (6 tl.)	250 Kč
DC-DC měnič	150 Kč
FT232R	150 Kč
Držák tiskové hlavy	600 Kč
Tisková hlava	350 Kč
Krabička	250 Kč
Obrobení krabičky	500 Kč
Drobné součástky	500 Kč
Celkem	**3450 Kč**

Dodatek D

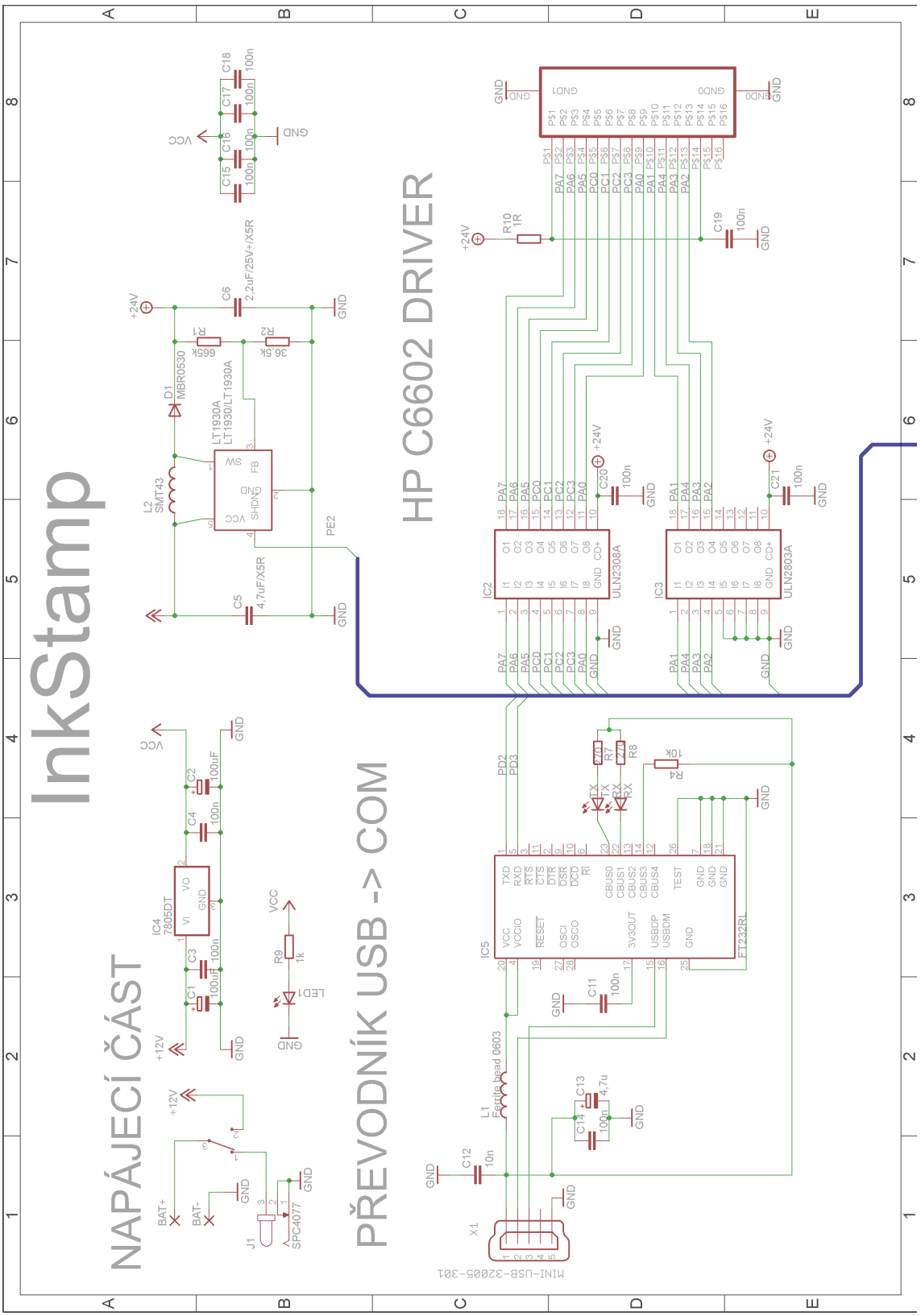
Schémata zapojení



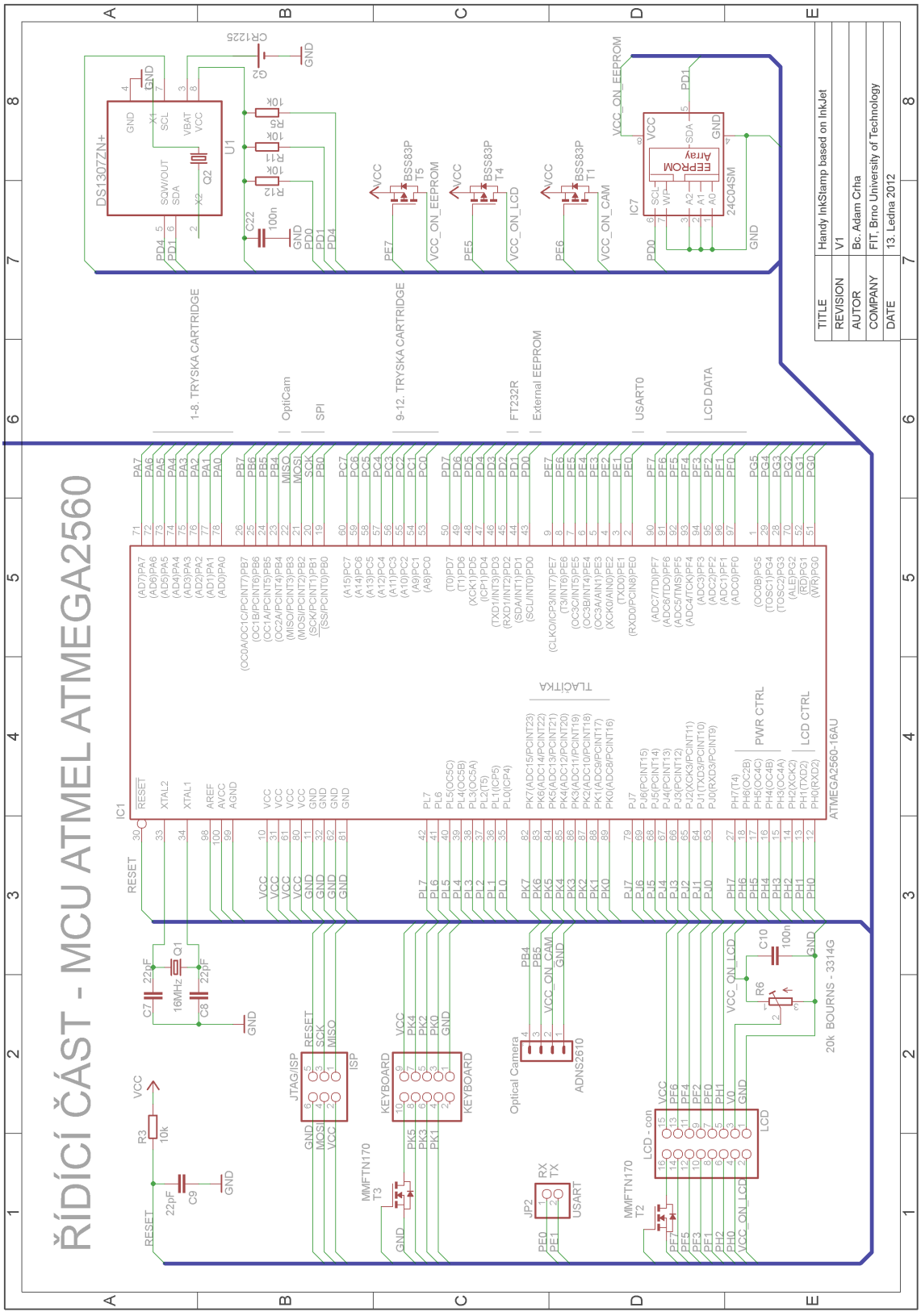
Obrázek D.1: Schéma DPS optického senzoru.



Obrázek D.2: Schéma klávesnice.



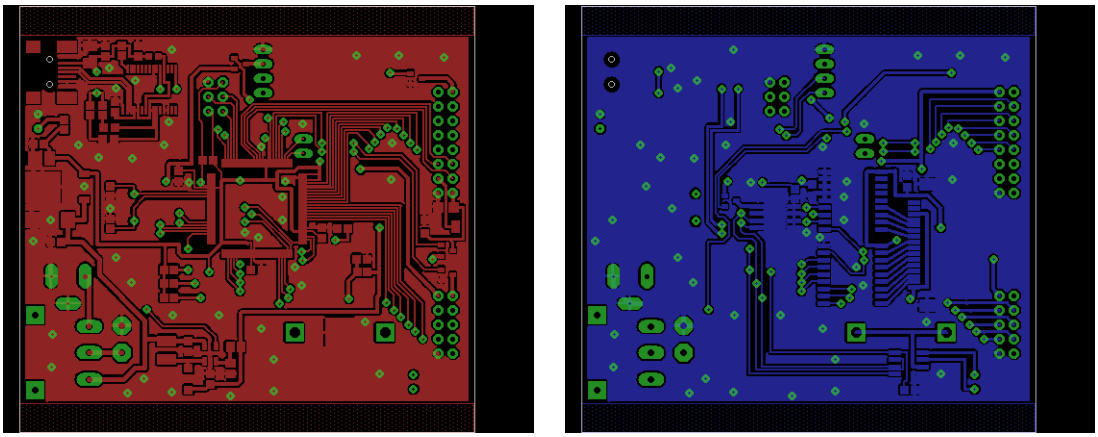
Obrázek D.3: Schéma základní desky, část A.



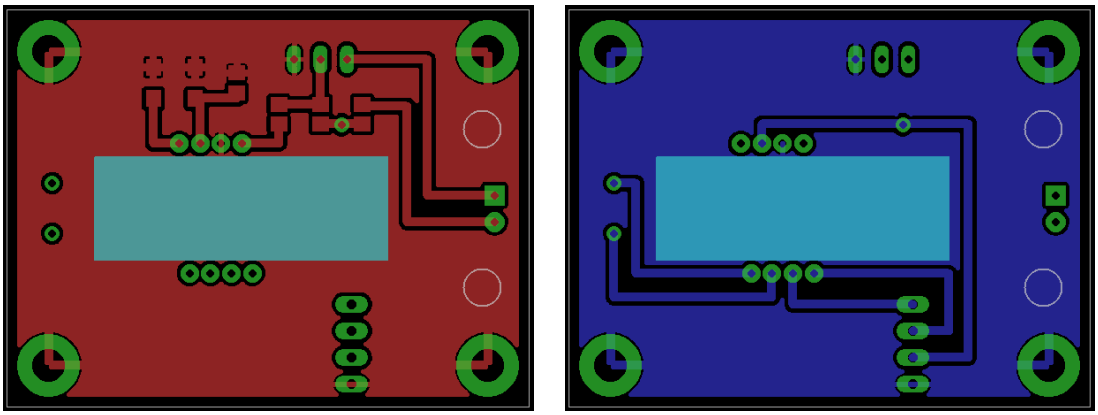
Obrázek D.4: Schéma základní desky, část B.

Dodatek E

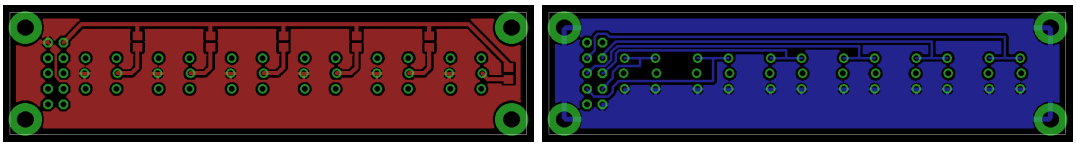
Desky plošných spojů



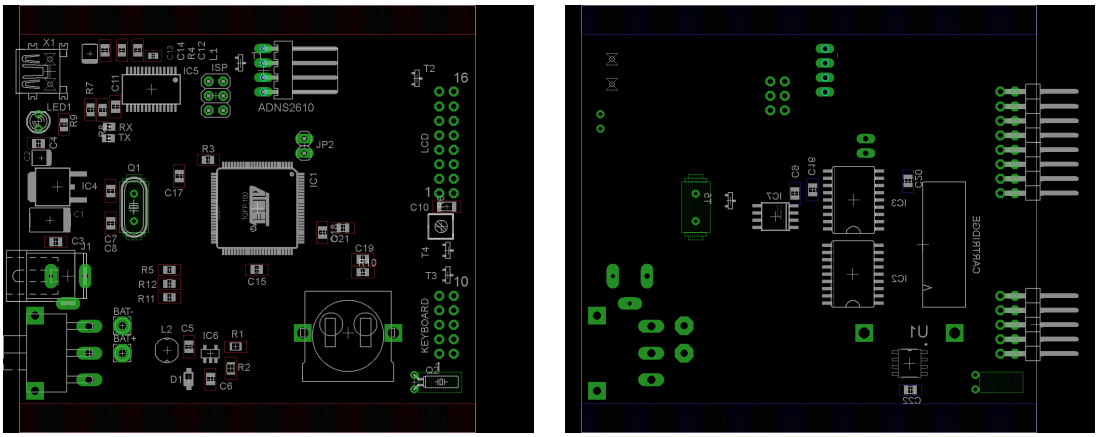
Obrázek E.1: DPS základní desky - horní strana vlevo, spodní strana vpravo.



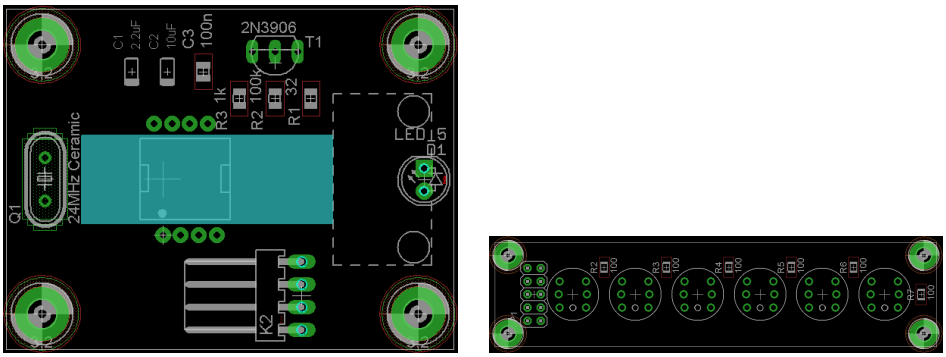
Obrázek E.2: DPS snímače - horní strana vlevo, spodní strana vpravo.



Obrázek E.3: DPS klávesnice - horní strana vlevo, spodní strana vpravo.



Obrázek E.4: Osazení horní a spodní strany hlavní desky plošných spojů.



Obrázek E.5: Osazení desky plošných spojů optického senzoru (vlevo) a klávesnice (vpravo). Pouze horní strana.