Středoškolská technika 2015

Setkání a prezentace prací středoškolských studentů na ČVUT

EEPROM Programmer

Vladislav Mlejnecký

Střední odborná škola a Střední odborné učiliště Nymburk
V Kolonii 1804

Obsah

Úvod

Můj projekt, EEPROM Programmer, vznikl jako reakce na potřebu programování paralelních EEPROM pamětí 28C256 a jim podobných. V dnešní době člověk najde na internetu velké množství různých programátorů pamětí, většinou jsou ale určeny pro programování sériových pamětí. Má to jednoduchý důvod, paralelní paměti už málokdo potřebuje. Jelikož se ve mně ale zrodil zájem, věnovat se starší mikroprocesorové technice, počínaje mikroprocesorem Intel 8080A, jsou pro mě paměti EEPROM v paralelním provedení nezbytné.

Původní myšlenka byla tedy taková, vyrobit velice jednoduchý a co nejuniverzálnější programátor pamětí, který nebude moc stát a ideálně půjde postavit z tzv. šuplíkových zásob. Nechtěl jsem programátor, který budu někde publikovat, chtěl jsem obyčejný pracovní nástroj.

Verze č. 1

Idea

Když jsem plánoval první verzi, rozhodoval jsem se jak ji koncipovat. Rozhodl jsem se, že programátor bude s počítačem komunikovat pomocí LPT portu. Můj osobní počítač je vybaven PCI kartou se dvěma LPT porty, a již tehdy jsem je hojně využíval, například pro programátory JTAG, logickým analyzátorem a dalšími jednoúčelovými zařízeními. Port LPT se zdál být jasnou volbou.

Dále bylo nutné rozhodnout se, jak bude fungovat hardware programátoru. Použití mikrokontroléru jsem zavrhl rovnou, neboť by pak bylo vhodnější použít sériový přenos dat namísto paralelního. Druhá možnost by byla použití menšího CPLD, například od firmy Xilinx, se kterými jsem se již setkal a pár menších vzorových úloh si vyzkoušel. Možnost by to byla rozumná, plošný spoj by byl estetický a jednoduchý. Ovšem odporovalo to myšlence levného programátoru ze šuplíkových zásob.

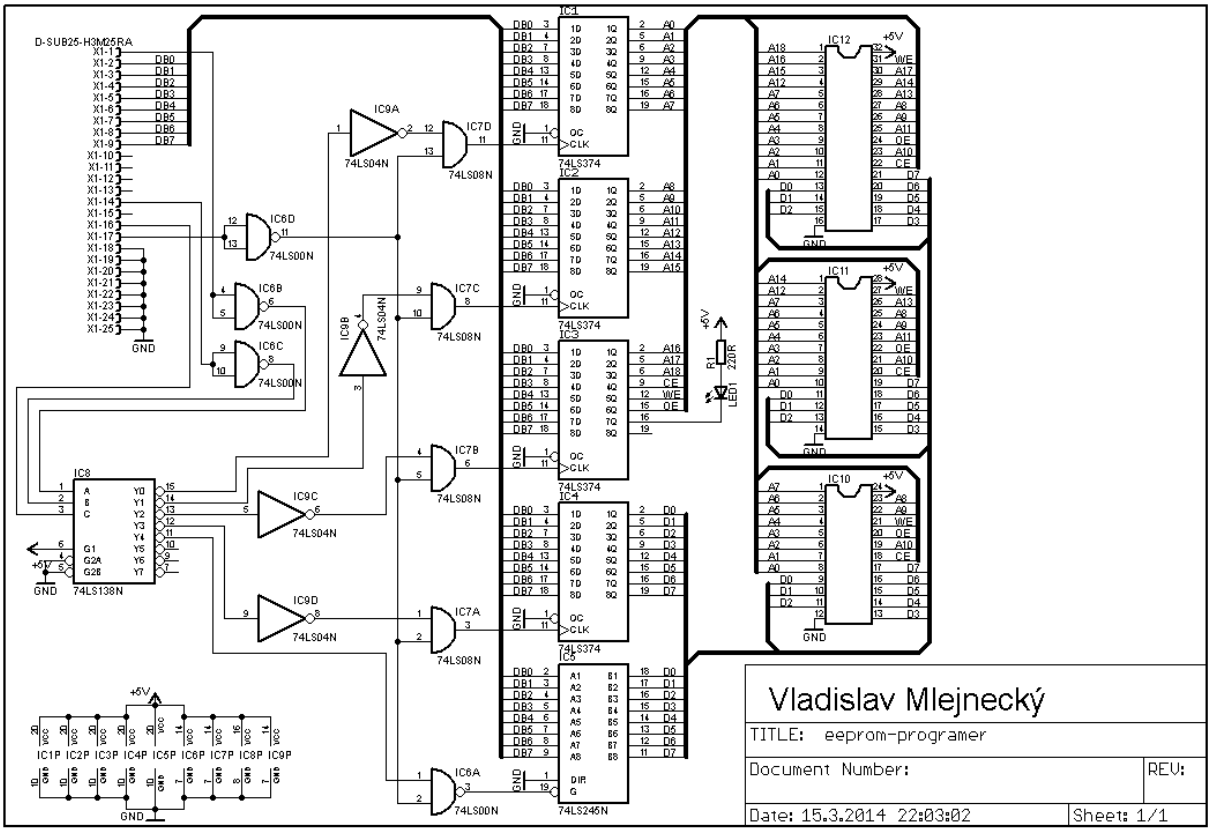
Nakonec jsem se rozhodl pro tu asi nejkrkolomnější cestu, kterou jsem mohl. Hardware programátoru postavím z integrovaných obvodů řady 7400. Všechny potřebné součástky se dají doma najít, nebude mě to tedy stát skoro nic, a se složitějším spojem si nějak poradím při návrhu.

Ovládací software, měl být naprogramován celý v programovacím jazyce Python a určený primárně pro operační systém Linux. Návrh počítal s tím, že program bude tvořen dvěma hlavními částmi – editorem a programátorem samotným. Editor měl umožňovat zobrazit obsah souboru Intel Hex, který by se nahrával do paměti, zobrazení dat přečtených z paměti a ovládání programátoru. Druhá část – programátor – měl sloužit k ovládání hardwaru, připojeného na LPT port počítače a samotné komunikaci s pamětí.

Hardware

Jádro hardwaru spočívalo ve čtyřech integrovaných obvodech typu 74LS374, jde o osmibitový paralelní registr s reakcí na náběžnou hranu a s třístavovými výstupy. Jeden z nich byl zapojen paralelně s integrovaným obvodem 74LS245, osmibitovém transceiveru s třístavovými výstupy, který umožňoval čtení dat z paměti. Dále se v zapojení nacházely tři programovací patice ve velikosti DIP24, DIP28 a DIP32, pro programované paměti, dále pak LPT port, a několik nutných hradel okolo zajišťujících funkci hardwaru. Rozpracované schéma celé první verze je na obrázku č. 1 níže.

Schéma bylo vytvořeno za pomoci programu Eagle, se kterým mám již bahaté zkušenosti a využívám jej při každém větším projektu.

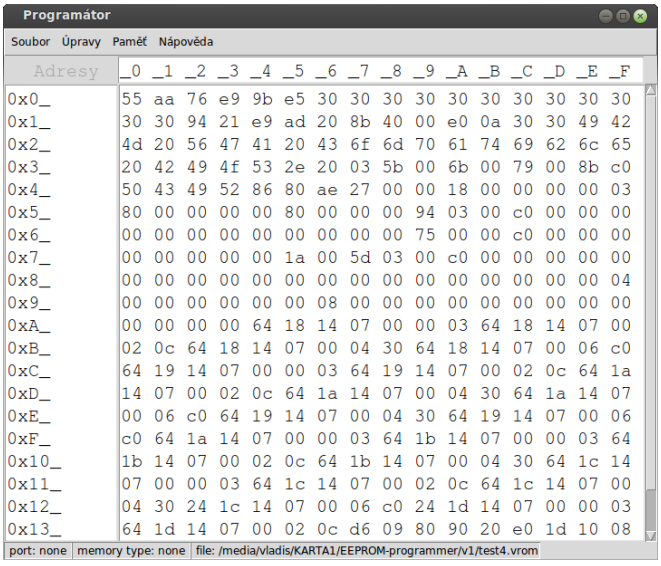


Obr. č. 1 Rozpracované schéma první verze programátoru.

Software

Zvolil jsem si programovací jazyk Python, pro jeho jednoduchost a rychlost programování v něm. Vznikla jen první část softwaru, editor. Snímek okna editoru můžete vidět na obrázku č. 2, napravo. Pro vykreslování GUI byl použit modul Tkinter, který funguje jak pod operačním systémem Linux, tak i pod operačním systémem Windows a výsledný program je tedy multiplatformní.

Program měl 381 řádků, což na to co uměl, bylo zkrátka moc, a byl poněkud těžkopádný. Bylo to dáno tím, že jsem v době jeho vzniku ještě moc neuměl používat modul Tkinter a toto byl prakticky můj první pokus o realizaci GUI.



Obr. č. 2 Snímek okna editoru.

Smysluplnost verze

Bylo zapotřebí, aby hardware programátoru nesl tři patice, neboť konstrukce s elektronickým přepínáním mezi napájením a daty (odlišné umístění vývodů u pamětí v menším pouzdře než u pamětí v pouzdře větším), by byla poměrně dost složitá a výrazně by zvětšovala, už tak dosti veliký, plošný spoj. Ve spojení s relativně vysokým počtem integrovaných obvodů by to mělo za následek, že výsledný plošný spoj by byl poměrně veliký.

Dále pak po stránce softwaru, se čím dál více ukazovalo, jak moc zbytečný vlastně editor je. A že pro plnou funkci bohatě stačí, když program bude mít o hodně jednoduší GUI, které bude umožňovat jen ovládat hardware, na základě požadavku nahrání souboru do paměti, přečtení paměti případně jiné operace.

Rozhodl jsem se tedy, že vývoj první verze pozastavím, poučím se z chyb, a přejdu k verzi druhé, která tyto chyby bude odstraňovat.

Verze č. 2

Idea

Jak jsem již zmínil, druhá verze se měla poučit z chyb verze první a být konečně použitelným zařízením. Opět jsem tedy stál u rozhodování na jakém principu navrhnout celý projekt. Používat hradla se ukázalo jako slepá cesta. Ostatně, používat LPT se taktéž ukázalo jako nepraktické. Bylo potřeba použít modernější rozhraní, nějaké, které bude i za pár let stále dostupné a použitelné. Tady nebylo moc na výběr, zvolil jsem USB. Je téměř všudypřítomné, levné a jednoduché na implementaci, taktéž je zde veliká pravděpodobnost, že i za pět let bude stále bez problému dostupné na nových zařízeních a vyhnu se tak potížím s nedostupností LPT portu.

Výborně, rozhraní bylo vybráno. Když ale USB tak na desce programátoru bude muset být přítomen nějaký jednočipový mikrokontrolér. Volba padla na osmi-bitový AVR od firmy Atmel. Sice firma nabízí i typy s integrovaným řadičem USB, ale jejich dostupnost je horší. Na desce tedy bude muset být i převodník mezi rozhraním USB a UART.

Software bude na počítači číst zdrojový soubor s daty, a pomocí USB bude zadávat jednoduché příkazy programátoru, který je bude plnit. Alespoň taková byla myšlenka verze dvě.

Představu jsem si tedy udělal a mohl jsem se pustit do samotné realizace.

Hardware

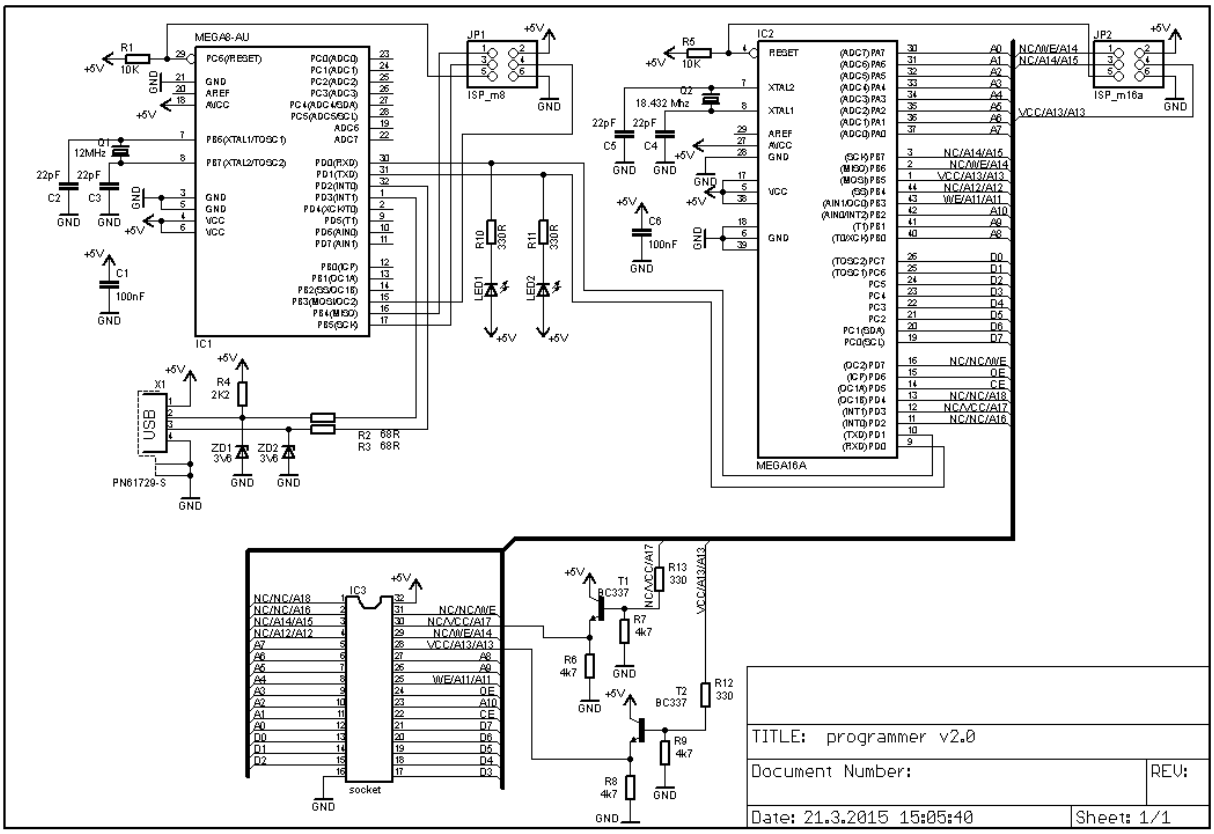
Návrh

Měl jsem jasnou představu výsledného programátoru. Co možná nejmenší rozměr desky plošného spoje, co nejnižší cena a s tím související využití šuplíkových zásob, jen jedna patice pro všechny typy EEPROM pamětí, dvě indikační led diody pro informaci o přenosu dat a možnost vyrobit plošný spoj doma. Pro návrh jsem použil opět program EAGLE a stvořil schéma, které můžete vidět na obr. č. 3.

Rozhodl jsem se nakonec, jít nestandardní cestou a místo obvykle používaného převodníku USB/UART od firmy FTDI, jsem zvolil konstrukci s druhým jednočipovým mikrokontrolérem. Použil jsem mikrokontrolér ATmega8A, který je levný a snadno dostupný ale nemá integrovaný USB řadič. Ovšem již před delší dobou jsem na internetu narazil na projekt AVR-CDC, který používá osmibitové mikroprocesory AVR pro softwarovou emulaci USB sběrnice. Dokáže se připojit k počítači a pomocí třídy CDC emulovat sériový port. Pod operačním systémem Linux, takto emulovaný port funguje zcela normálně a okamžitě po zapojení programátoru do USB portu. Výsledek je o něco pomalejší, co se týká přenosu dat, oproti řešení od firmy FTDI, ale cena, 40 korun za AVR oproti 150 korunám za FT232RL je docela velký rozdíl a programátor stejně nebude využívaný pro programování většího množství pamětí, nýbrž jen pro občasně naprogramování jedné či dvou pamětí.

Druhým mikrokontrolérem na desce programátoru je, taktéž osmibitové AVR, ATmega16A, které má za úkol plnit příkazy přicházející z počítače a programovat tak tedy připojenou paměť. Typ ATmega16A, jsem zvolil hlavně kvůli počtu jeho GPIO, které přesně

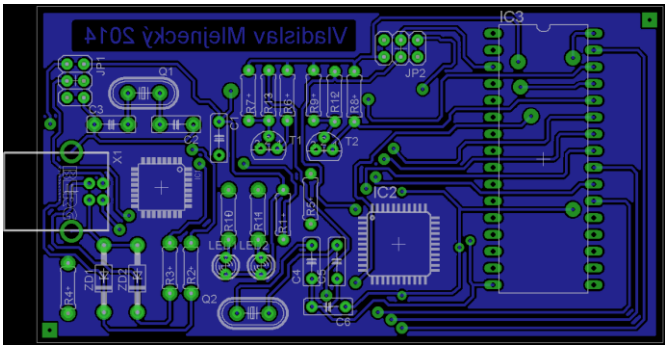
odpovídá potřebnému počtu pro obsloužení paměti. Tento mikrokontrolér si nese 1kB interní paměti SRAM a 16kB paměti FLASH, což by mělo být pro celý firmware dostatečné. Pokud by se náhodou ukázalo, v průběhu vývoje firmware, že to dostatečné není, je možné jej nahradit typem ATmega32A, který je pinově kompatibilní a paměti má dvojnásobek.



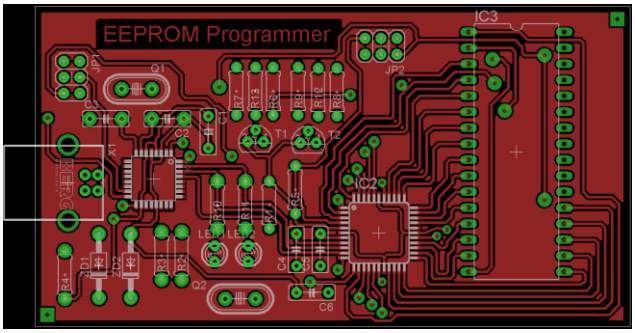
Obr. č. 3 Schéma druhé verze programátoru.

Dalším krokem bylo vytvořit návrh desky plošného spoje. Program EAGLE, toto umožňuje taktéž a proto jsem volil jej. Jak jsem již psal v odstavci Idea, chtěl jsem co nejmenší plošný spoj a přesto vyrobitelný doma. Pokud jsem chtěl malý plošný spoj, musel jsem jej navrhnout oboustranně, pokud jsem jej ale chtěl vyrobit doma, oboustranný plošný spoj byl spíše překážkou.

Nakonec jsem se odhodlal navrhnout oboustranný plošný spoj, s tím že jej zkusím vyrobit doma. Otvory prokovené nebudou, použiji pokud možno vývod součástky, pokud to nebude možné tak holt udělám propojku z drátku. V rozích jsem si pak umístil otvory pro slícování obou stran plošného spoje. Výsledný návrh plošného spoje je vidět na obrázku č. 4 a obrázku č. 5. Na obrázku č. 4 je vidět spodní strana plošného spoje, na obrázku č. 5 pak strana vrchní.



Obr. č. 4 Spodní strana plošného spoje.



Obr. č. 5 Vrchní strana plošného spoje.

Celý plošný spoj jsem naroutoval na

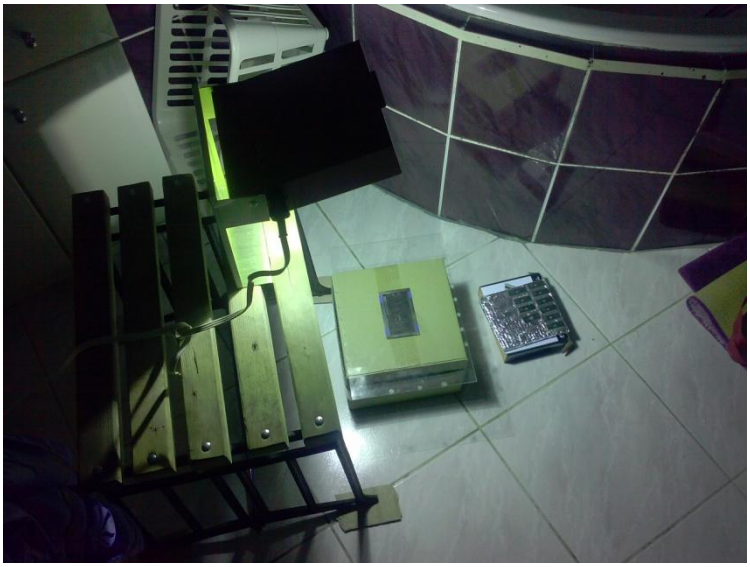
svém notebooku s procesorem Intel Atom N270, displejem o úhlopříčce 10,1“ a rozlišení 1024x600px. Byla to docela výzva a opravdu silná zkouška mého odhodlání a trpělivosti.

Realizace

Vzhledem k faktu, že letos maturuji, byla realizace hardware pomalá a stále jsem na ni nemohl najít čas. Vždy jsem kousek práce odvedl o některých prázdninách, a postupně jsem se tedy realizací prokousal až do finále.

Výroba plošného spoje

Disponuji doma veškerým vybavením potřebným pro výrobu plošných spojů fotocestou a s jejich výrobou, touto metodou, mám již bohaté zkušenosti. Novinkou pro mě ovšem bylo vytvořit plošný spoj oboustranný. Zatím jsem vyráběl pouze jednostranné plošné spoje. Nakonec se ukázalo, že ani domácí výroba oboustranného plošného spoje není nic složitého, čeho by se musel amatér v domácích podmínkách bát.

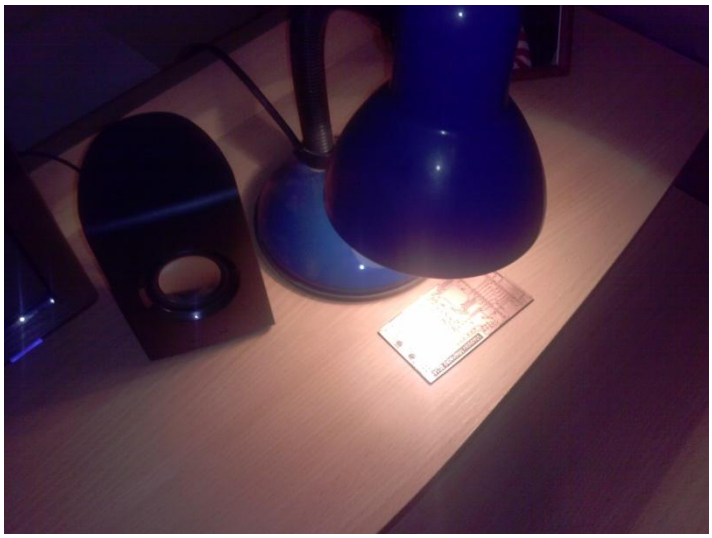


Obr. č. 6 Expozice plošného spoje a mazání EPROM pamětí.

Ze všeho nejdříve jsem nechal v tiskárně zhotovit masku plošného spoje. Laserovou tiskárnou vytištěný motiv plošného spoje na průhledné fólii. Tyto motivy jsem ostřihl pro lepší manipulaci.

Dalším krokem bylo připravit si cuprextit. Doma jsem našel starší odstřižek oboustranného cuprextitu ideální velikosti. Ten jsem očistil, obrousil, odmastil a poté z jedné strany nalakoval fotocitlivým lakem. Nalakovaný cuprextit jsem nechal zaschnout.

Když byl lak suchý, osvítil jsem přes masku motiv na plošný spoj. Jako osvitka dobře posloužilo staré horské sluníčko, a jako temná místnost koupelna, se zabedněným oknem. Při osvitu jsem zrovna smazal EPROM paměti. Na obrázku č. 6 je vidět expozice plošného spoje.



Obr. č. 7 Trik se stolní lampičkou.

Po uplynutí zhruba osmi minut, mám odzkoušeno, že toto je ideální doba expozice, jsem cuprextit ponořil do vývojky, respektive do rozpuštěného hydroxidu sodného. Během okamžiku se osvícený lak uvolnil a já před sebou viděl krásně přenesený motiv plošného spoje. Plošný spoj jsem omyl horkou vodou a opatrně osušil.

Abych zabránil odleptání druhé strany plošného spoje, zakryl jsem ji důkladně lihovým fixem. Zkoušel jsem experimentovat i s lakem na nehty ale ten se neukázal jako vhodný. Plošný spoj poté putoval do leptací lázně, tedy do

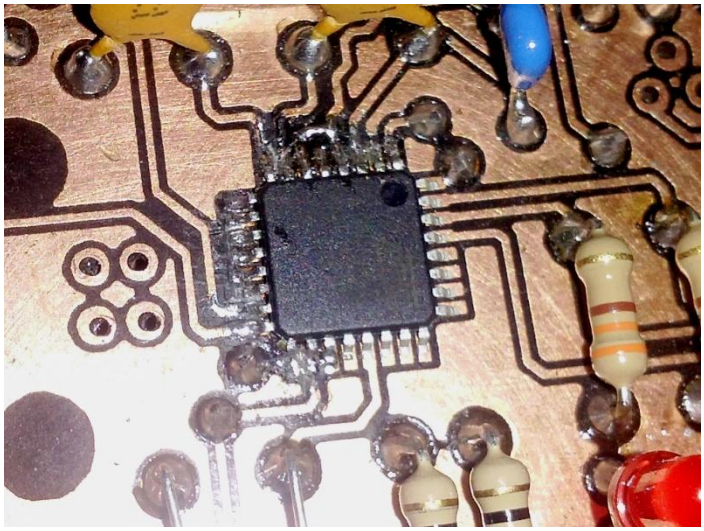
chloridu železitého, a brzy jsem měl první stranu plošného spoje hotovou.

Dalším krokem bylo vytvoření druhé strany, již při návrhu jsem na toto myslel a do rohů plošného spoje jsem umístil otvory, abych obě strany náležitě slícoval. Teď je stačilo jen provrtat a celý postup zopakovat. Jediným rozdílem bylo, že při umisťování masky jsem musel být velice pečlivý, aby byly kontrolní otvory v zákrytu.

Kompletně vyleptaný plošný spoj jsem vyvrtal, očistil, z obou stran nalakoval a nechal zaschnout. Pro zasychání používám malý trik se stolní lampičkou a 60W žárovkou pro urychlení schnutí, detail na obrázku č. 7.

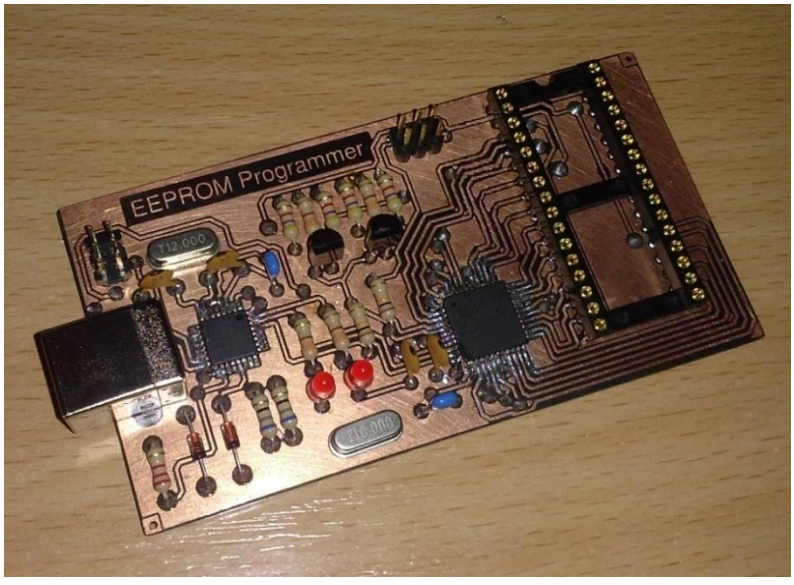
Osazení součástkami

Když byl plošný spoj hotový a já našel opět chvilku na práci, mohl jsem jej konečně osadit součástkami. Používám mikropájku, kterou jsem si sám vyrobil, když jsem se elektronice začínal věnovat. Jednoduchá spolehlivá konstrukce umí za práci vzít, ovšem jak se ukázalo, hrot pájky je už mírně opotřebovaný a tak výsledek pájení SMD pouzder není zcela vynikající. Funkční to sice je, ale není to tak hezké, jak jsem chtěl. Detail na obrázku č. 8.



Obr. č. 8 Detail pájení SMD.

Jelikož byl plošný spoj oboustranný a nebylo zde prokovů, musel jsem všechny vývodové součástky zapájet z obou stran. Menší problém byl se zapájením patice, u ní, ačkoliv jsem volil precizní verzi, bylo málo místa pod plastem, dostat se tam hrotem pájky a neponičit při tom tělo patice bylo docela náročné. Tam kde měli být obyčejné propojky mezi vrchní a spodní vrstvou, použil jsem odstřižků z vývodů rezistorů, což bylo taktéž poměrně náročné, neboť když jsem zapájel drátek z jedné strany, a chystal se pájet druhou stranu, měla strana první tendenci se opět uvolnit.



Obr. č. 9 Osazený plošný spoj.

Bohužel jsem si uvědomil, že jsem při návrhu udělal drobnou chybu při umisťování USB konektoru. Zvolil jsem totiž verzi bez stínění a tak mi editor nepřipravil do masky otvory pro jeho uchycení. Při vrtání plošného spoje jsem tento nedostatek přešel s myšlenkou, že ony 4 vývody USB konektoru jej spolehlivě udrží na místě. Jak se ale ukázalo po osazení, mýlil jsem se. Konektor má poměrně velkou vůli a viklá se. Při manipulaci se s ním proto musí zacházet opatrně.

Celý osazený plošný spoj je vidět na obrázku č. 9.

Oživení

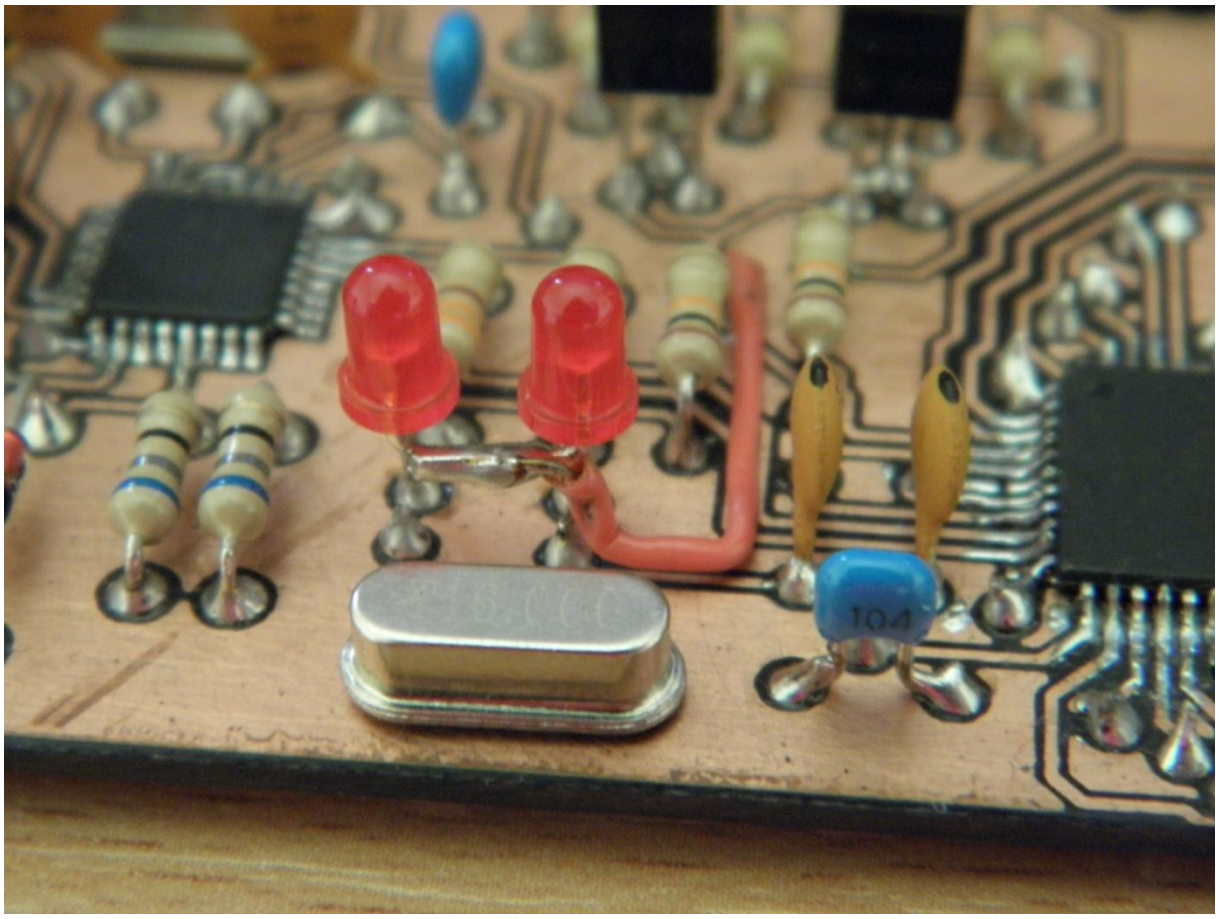
Poté co jsem měl na stole osazený plošný spoj, čekala mě další část, a sice oživení a základní odzkoušení funkčnosti. Oživení spočívalo v nahrání příslušného firmware do obou mikroprocesorů. Pro nahrávání firmware do mikroprocesorů slouží dva malé, šesti-pinové konektory.

Pomocí programátoru USBasp jsem nahrál do menšího mikrokontroléru, ATmega8A, aktuální a odpovídající firmware z projektu AVR-CDC. Dále jsem si připravil jednoduchý testovací program pro větší mikrokontrolér, ATmega16A, který vytvořil jednoduchou smyčku na rozhraní UART, která odeslala zpátky to, co přijala. Tento program jsem zkompiloval a opět pomocí programátoru USBasp nahrál do mikrokontroléru.

Takto připravený programátor jsem zapojil do USB portu počítače, zjistil, jakou cestu mu Linux přiřadil a zkusil otevřít sériový terminál. Vložil jsem do něj krátkou sekvenci znaků, kterou jsem odeslal. Stejná sekvence se mi i vrátila, což značilo malý úspěch, neb převodník USB-UART realizovaný pomocí ATmega8A opravdu funguje.

Nicméně jeden zádrhel se přeci jen našel. Nevěřícně jsem sledoval LED diody, které mají signalizovat přenos dat, že svítí, když se data nepřenáší a jen když nějaká data začnou proudit tak zhasnou. To bylo přesně naopak, než jsem chtěl. Po krátké úvaze mi to došlo. Led diody jsem nechtěně do schématu nakreslil tak, jako by UART měl být v klidu v log. 0. Jenže tomu tak není a v klidu se UART nachází v log. 1. Zapojení jsem tedy předělal a schéma i návrh plošného spoje patřičně upravil. Zároveň jsem upravil ve výkresu USB konektor tak, aby byla použita verze se stíněním a toto stínění bylo možné zapájet do plošného spoje.

Opravené zapojení indikačních LED diod je vidět na obrázku č. 10, stačilo je jen otočit tak aby byli katodou na datovém vodiči a k anodám přivést +5V, toto napětí se naštěstí vyskytuje na vývodu jednoho rezistoru nedaleko.



Obr. č. 10 Oprava zapojení indikačních LED diod.

USB firmware AVR-CDC

Jak jsem se již zmínil, převodník USB na UART je tvořený pomocí osmibitového AVR mikrokontroléru ATmega8A. Celá tato část je převzata z projektu AVR-CDC[1].

Tento projekt využívá implementaci V-USB od Object Deveopment's a třídu CDC k vytvoření virtuálního sériového portu. Komunikace probíhá pomocí staršího USB 1.1 ve verzi low speed. Výsledný port je schopný přenášet data rychlostí až 38400bps, taktéž umožňuje řízení toku dat, ačkoliv jsem tuto možnost v projektu nevyužil.

Firmware a software

Firmware pro hlavní mikrokontrolér, ATmega16A, stejně tak jako software pro ovládání programátoru, je v dobu psaní této práce stále ještě ve vývoji vzhledem k rozsahu projektu a nedostatku času.

Celý firmware bude jakýmsi interpretem příkazů přicházejících z řídícího počítače, mikrokontrolér bude přijímat pomocí UART příkazy a data, ukládat si je do bufferu, a postupně zpracovávat. Příkazy budou jednoduché, ve stylu „nastavit adresu XXXXX“ „sekvenčně zapsat následujících 256 bajtů dat“ apod. V tuto chvíli mám naprogramované základní funkce pro obsluhování paměti, v nejbližší době se je budu pokoušet dát dohromady tak aby bylo možné alespoň některé druhy pamětí číst.

Ovládací software pro počítač bude s největší pravděpodobností konzolový a velice jednoduchý, bude umět provádět s pamětí základní operace jako čtení, mazání, zápis, zamknutí apod. Realizovat jej budu v programovacím jazyce C a bude primárně určený pro operační systém Linux.

Závěr

Tento projekt patří rozsahem a náročností mezi ty největší, do kterých jsem se kdy zatím pustil. Potřeba jednoduchého programátoru pár typů EEPROM pamětí nakonec přerostla ve stavbu poměrně komplexního a složitého programátoru. Na druhou stranu jsem ale rád, neboť mi takový projekt může přinést spousty zkušeností a nových znalostí.

Až projekt dokončím, zveřejním na své osobní internetové stránce[2] kompletní dokumentaci a zdrojové kódy pod open sourcovou licencí. Takže každý případný zájemce o tento programátor si jej bude moci doma sám postavit.

Možná bych se jednou mohl zamyslet i nad verzí číslo 3. Ta by ovšem už musela mít o hodně lepší vlastnosti, nejlépe mikrokontrolér ARM, s integrovaným plnohodnotným řadičem USB. Dále pak najít vhodný a levný pin driver, který by umožňoval ovládat 40 pinů s možností volby digitální vstup či výstup, +5V, +3,3V, +12V a jeden zdroj volitelného napětí. Poté by bylo možné zkonstruovat programátor velice univerzální, který by mohl programovat vše možné, od pamětí PROM, přes některé MCU až po programovatelná logická pole. Otázkou ovšem je, jak drahý by takový programátor byl, a jestli by jeho vývoj byl vůbec smysluplný. Neboť takovéto programátory na trhu dostupné jsou.

Seznam odkazů

[1] http://www.recursion.jp/avrcdc/
[2] http://vladis.moxo.cz/