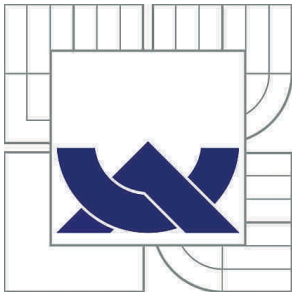
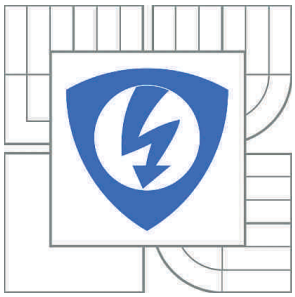


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

LABORATORNÍ VÝUKOVÝ SYSTÉM S MIKROKONTROLÉREM

MICROCONTROLLER BASED LABORATORY KIT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

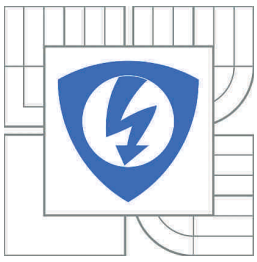
AUTOR PRÁCE — Bc. ROMAN ZACH

AUTHOR

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

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Roman Zach	***ID:***	115313
Ročník:	2	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Laboratorní výukový systém s mikrokontrolérem

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte koncepci komplexního laboratorního výukového kitu vybaveného mikrokontrolérem a periferiemi. Zaměřte se na co nejširší spektrum periferií (čítače, časovače, AD, DA, displeje, klávesnice, I2C, SPI, UART, Ethernet atd.). Systém navrhněte, realizujte, osaďte a oživte. Vybavte programovým vybavením pro PC i pro mikrokontrolér a demonstrujte správnou funkci. Vytvořte pro jednotlivé periferie demonstrační úlohu včetně vzorového řešení.

DOPORUČENÁ LITERATURA:

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

Termín zadání:	11.2.2013	***Termín odevzdání:***	20.5.2013

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

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

UPOZORNĚNÍ:

Abstrakt

Tato práce prezentuje návrh a konstrukci laboratorního výukového systému s mikrokontrolérem. Hardwarová část popisuje zapojení jednotlivých bloků laboratorního výukového systému s mikrokontrolérem. Softwarová část popisuje funkci jednotlivých programů.

Klíčová slova

ARM, STM32F107VCT6, ST-LINK, CoIDE, GCC, HD44780, SSD1289, UART, RS232, RS485, USB, SPI, EEPROM, ETHERNET, FLASH, I2C, AD, DA, RTC

Abstract

This thesis presents the design and construction of microcontroller based laboratory kit. Hardware part describes the wiring of microcontroller based laboratory kit. Software section describes the function of each program.

Keywords

ARM, STM32F107VCT6, ST-LINK, CoIDE, GCC, HD44780, SSD1289, UART, RS232, RS485, USB, SPI, EEPROM, ETHERNET, FLASH, I2C, AD, DA, RTC

Bibliografická citace:

ZACH, R. *Laboratorní výukový systém s mikrokontrolérem.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 138s. Vedoucí diplomové práce doc. Ing. Zdeněk Bradáč, Ph.D..

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Laboratorní výukový systém s mikrokontrolérem jsem vypracoval samostatně pod vedením vedoucího semestrální práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne: ……

podpis autora

Poděkování

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

V Brně dne: ……

podpis autora

Obsah

1 ÚVOD

Cílem této práce je provést návrh vývojového kitu určeného pro výuku programování mikrokontrolérů. Na základě návrhu pak vývojový kit vyrobit, osadit, oživit a vytvořit ukázkové úlohy.

V první části práce byla provedena rešerše dostupných řešení jak vývojových kitů tak vhodných JTAG programátorů. Do rešerše byli vybráni zástupci majoritních výrobců mikrokontrolérů (zaměřil jsem se na mikrokontroléry s jádrem ARM). V druhé části práce byl proveden návrh hardwaru vývojového kitu, přičemž byl kladen důraz na co nejširší spektrum periferií. A poslední část se zabývá vytvořením softwaru pro jednotlivé periferie.

Po provedení rešerše byl vybrán mikrokontrolér STM32F107VCT6, tento mikrokontrolér byl vybrán, protože obsahuje ethernet a poměrně výkonné jádro CORTEX-M3, které je určeno pro embedded aplikace. Další předností tohoto mikrokontroléru je dostatečný počet I/O pinů a periferií.

V druhé části práce je proveden návrh vlastního vývojového kitu, který byl vybaven třemi displeji (klasickým alfanumerickým s řadičem HD44780, maticovým displejem a grafickým displejem s dotykovou folií), maticovou klávesnicí, klasickými tlačítky, LED diodami, stejnosměrným motorem, krokovým motorem, I2C periferiemi (AD a DA převodník, hodiny reálného času, EEPROM a expandéry), SPI flash pamětí, inkrementálním snímačem otáček, rozhraními připojenými pomocí UARTu (RS232, RS485 a USB) a ethernetem.

V poslední části práce byly pro jednotlivé periferie vytvořeny úlohy a jejich vzorové řešení. Pro vývoj softwaru bylo použito nekomerční vývojové prostředí CoIDE, které je založeno na prostředí Eclipse. K překladu zdrojových kódů byl použit GCC překladač pro ARM.

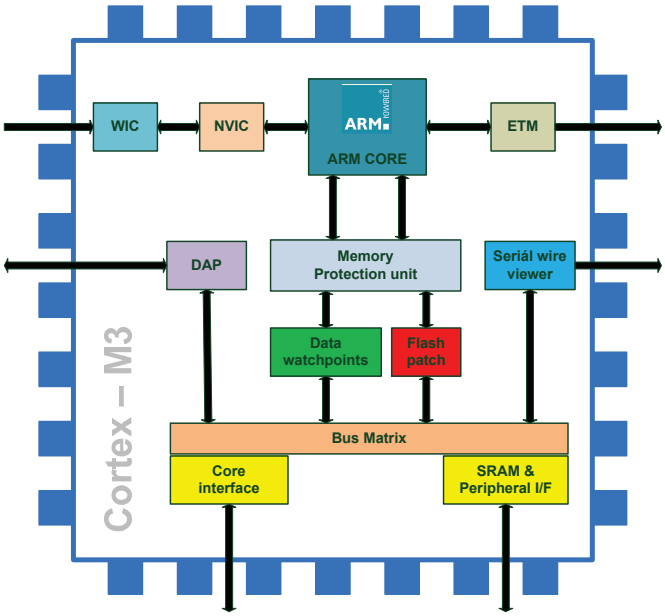
2 ZHODNOCENÍ DOSTUPNÝCH ŘEŠENÍ

V této části práce budou popsány vyráběné vývojové kity. Na trhu je v dnešní době spousta výrobců mikrokontrolérů a vývojových kitů. Mezi přední výrobce patří: STM, ATMEL, NXP, FREESCALE a spousta dalších. V poslední době se na vrchol dostaly mikrokontroléry s jádrem ARM, které nabízí velký výpočetní výkon s relativně malou spotřebou. Mikrokontroléry ARM se dostávají do všech odvětví techniky, a proto není žádným překvapením, že si své místo našly i v oblasti řídicí techniky. V těchto mikrokontrolérech je velký potenciál, jelikož jejich jádro je otevřené a na vývoji mikrokontolerů založených na jádře ARM pracují desítky firem na světě.

Pro vývoj řídicích systémů jsou určeny mikrokontroléry s jádrem ARMv7 označované jako CORTEX. Existují v několika verzích CORTEX M (určené pro všeobecné použití) a CORTEX R (určené pro real-time systémy). Tato jádra následně výrobci doplňují periferiemi, takže na trhu je nepřeberné množství variant mikrokontrolérů. V této práci se zaměřím na jedno z nejrozšířenějších jader CORTEX M3, které pro řídící účely nabízí dostatečný výkon (až 100MHz) a přitom je cenově dostupné.

2.1 Popis jádra ARM CORTEX – M3

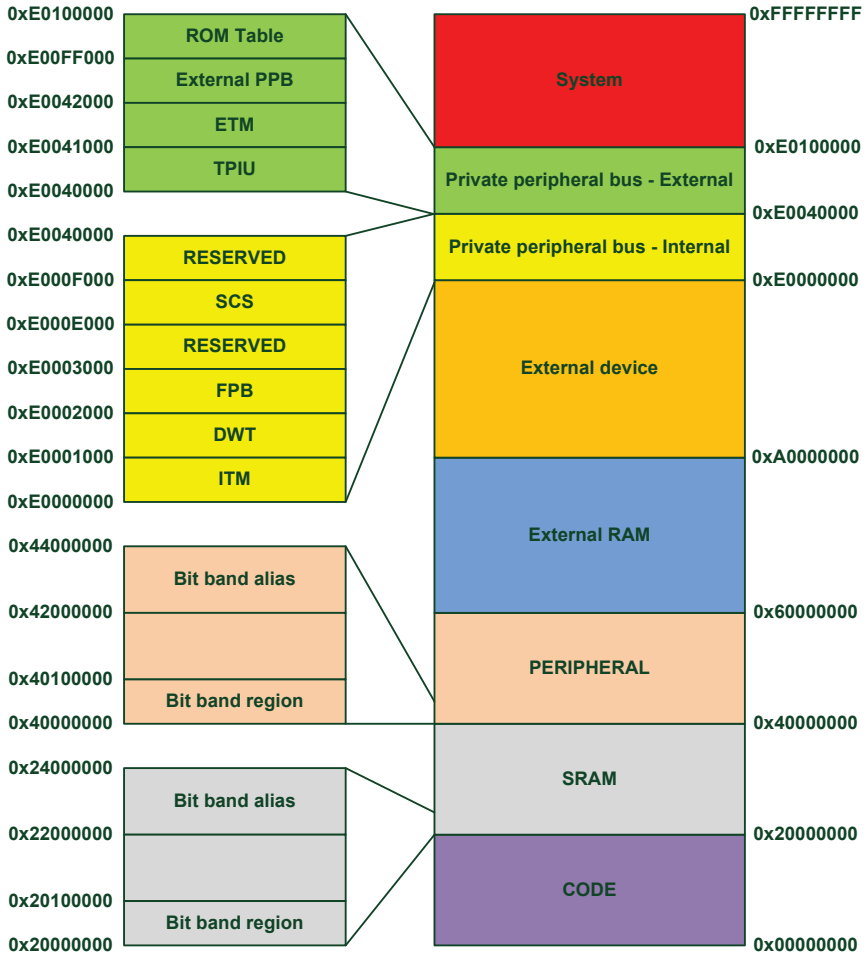
Architektura jádra mikrokontroléru ARM CORTEX M3 je naznačena na obrázku 1. Jedná se o 32 bitovou architekturu s redukovanou instrukční sadou (RISC). Díky redukované instrukční sadě je hardware ARM jádra jednoduší a tím i levnější.



Obrázek 1. ARM CORTEX - M3 [1]

ARM CORTEX - M3 používá instrukční sadu thumb2, díky které je dosahováno o 70% větší účinnosti než u starších verzí ARM jader s instrukční sadou thumb. Výpočetní výkon je 1.25MIPS/MHz, přičemž energetická účinnost je až 0,047mW/MHz, to je dosaženo také díky technologii výroby s přesností 0,13µm. Na vysokém výkonu se také podílí harvardská architektura, na které je toto jádro postaveno. Jelikož dochází zároveň ke čtení dat a instrukcí, tak je v jádru implementován třístupňový pipeline, který v prvním kroku vyzvedne instrukci z paměti, ve druhém kroku provede její dekódování a v posledním kroku je provedeno její zpracování.[1]

Adresová a datová sběrnice je u ARM jádra 32 bitová, z čehož vyplývá, že maximální adresní prostor je 4GB (organizace paměti je naznačena na obrázku 2). Jádro obsahuje 13 registrů pro všeobecné použití, dva ukazatele zásobníku, programový čítač, stavový registr a speciální funkční registry.[1]

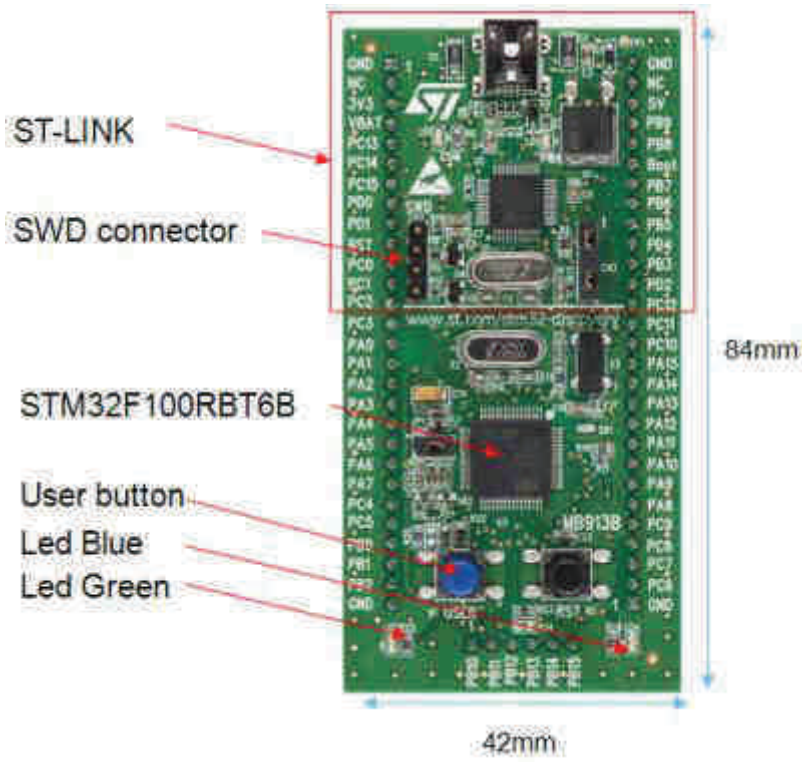


Obrázek 2. Adresovatelný paměťový prostor ARM CORTEX M3 [1]

Podrobnější údaje o jádru ARM CORTEX M3 jsou v referenčním manuálu výrobce [1].

2.2 Vývojový kit STM32VLDISCOVERY

Jedná se o vývojový kit firmy STM určený pro širokou veřejnost, cena tohoto kitu je velmi příznivá a proto si jej může dovolit každý.

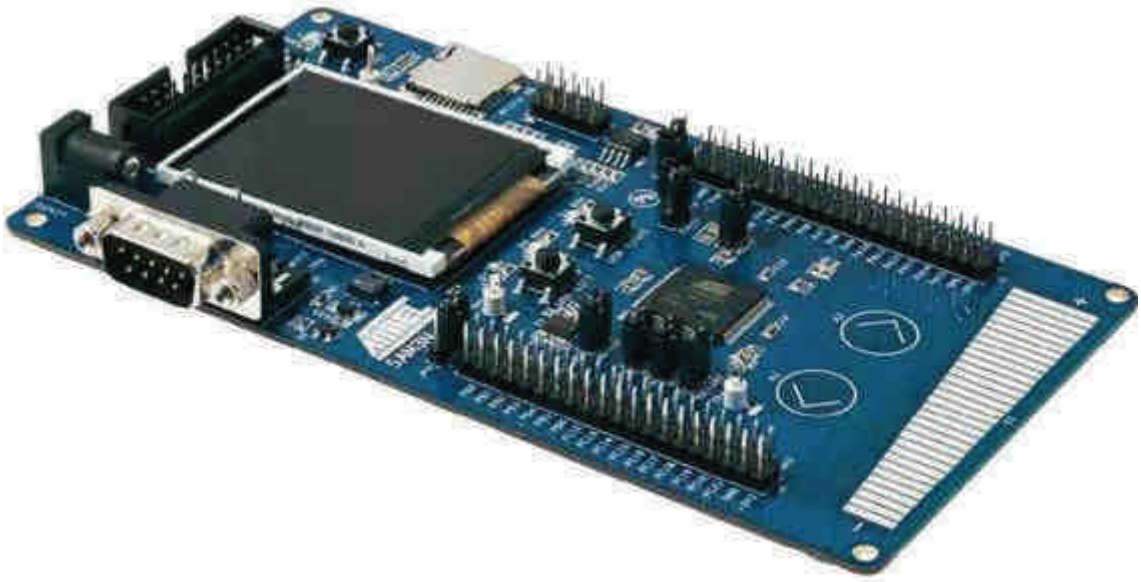


Obrázek 3. Vývojový kit STM32VLDISCOVERY [2]

Tento kit obsahuje mikrokontrolér STM32F100RBT6B, který disponuje 128kB paměti, 8kB SRAM, hodinami reálného času RTC, 7 kanálovým DMA, 12 – bitovým AD převodníkem s dobou převodu 1,2us, dvěma 12 – bitovými DA převodníky, 12 časovači, dvěma I2C rozhraními, třemi USART rozhraními a dvěma SPI rozhraními s maximální rychlostí 12Mbit/s. Součástí kitu je i programátor ST-LINK, který lze použít i k programování jiných mikrokontolerů STM32. Kit dále obsahuje dvě led diody a jedno uživatelské tlačítko. Nevyužité piny mikrokontroléru jsou vyvedeny na pinheadové lišty. Použitý mikrokontrolér je taktován na maximální frekvenci 24MHz, jedná se o 32bitový mikrokontrolér, který je určen jako náhrada starších 8bitových mikrokontrolérů.[2][4]

2.3 Vývojový kit ATMEL SAM3N Evaluation kit

Jedná se již o poměrně vybavený kit, který je osazen výkonným mikrokontrolérem ATMEL SAM3N4C.



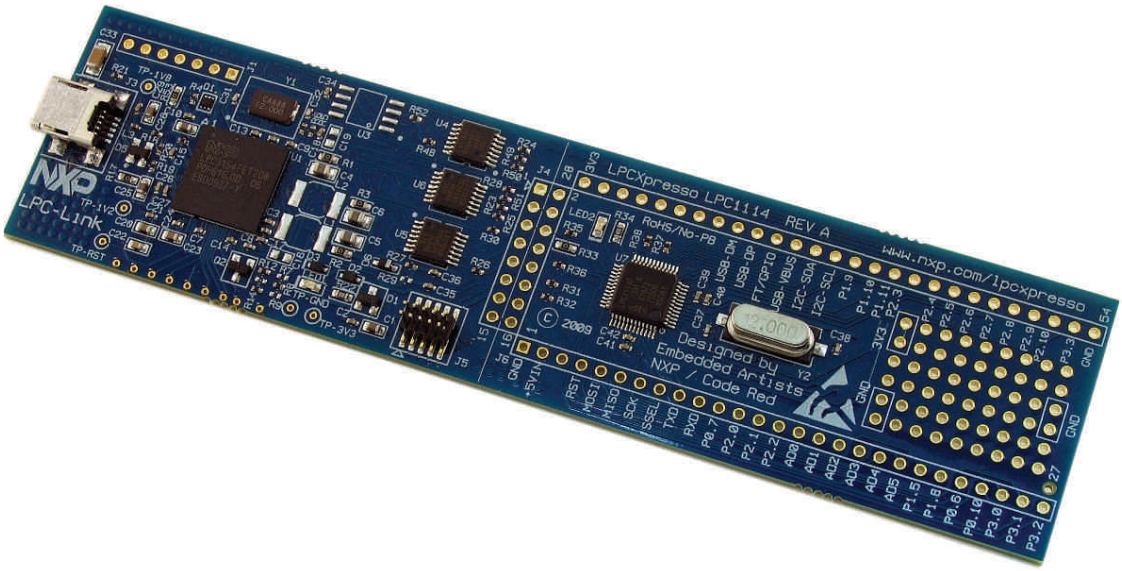
Obrázek 4. Vývojový kit SAM3N Evaluation kit [5]

Tento kit disponuje čtečkou SD karet, převodníkem RS232 na UART, 2,0“ barevným TFT displejem, tlačítky v různém provedení a dvěma led diodami. Mikrokontrolér má 256kB flash pamět, 24kB SRAM, 16KB ROM pamět s bootloaderem, který používá UART. Další periferie mikrokontroléru jsou: 2xUSART, 2xUART, 2xTWI (obdoba I2C), 3xSPI, 10bitový AD převodník, 10bitový DA převodník, hodiny reálného času RTC a 6 časovačů. Kit je osazen mikrokontrolérem se 100pinovým pouzdrem LQFP.[5][6]

Nevýhodou tohoto vývojového kitu je vysoká cena (téměř 100$), absence jtag programátoru (musí se koupit samostatně), a v neposlední řadě u kitu v této cenové relaci mi chybí ethernet.

2.4 Vývojový kit NXP LPCXpresso

Jedná se o vývojový kit firmy NXP, který se prodává v několika variantách, které se liší použitým mikrokontrolérem.



Obrázek 5. Vývojový kit NXP LPCXpresso [7]

Tento vývojový kit je velmi podobný kitu od STM, ovšem firma NXP se neomezuje na variantu s jedním typem mikrokontroléru, ale má v nabídce hned několik variant. Já se zaměřím na variantu s nejvýkonnějším mikrokontrolérem LPC1700. Vývojový kit je složen z programátoru a destičky s mikrokontrolérem, která obsahuje jen to nejnutnější (vlastní mikrokontrolér a krystal pro taktování mikrokontroléru), periferie si každý musí k mikrokontroléru připojit sám nebo si dokoupit desku s periferiemi. Co se týká mikrokontroléru, tak LPC1700 disponuje: 512kB paměti flash, 96kB SRAM, 4kB EEPROM, LCD řadičem pro displeje do velikosti 1024x768 pixelů, ETHERNET MAC vrstvou s interfacem MII a RMII (fyzická vrstva se musí dodat externě), USB 2.0, 3xI2C, 3xSPI, 5xUART, 1xI2S, 12-bitovým AD převodníkem, 10-bitovým DA převodníkem, až 165 vstupně/výstupními piny a je možné jej provozovat na frekvenci 120MHz.[7][8]

Výrobce k tomuto kitu nabízí také jejich vývojové prostředí LPCXpresso IDE, které je pro nekomerční použití do velikosti kódu 128kB zdarma.

2.5 JTAG programátor J-LINK V8

Jedním z předních výrobců JTAG/SWD programátorů pro mikrokontroléry ARM je SEGGER, který podporuje většinu ARM mikrokontrolérů všech výrobců a je kompatibilní s většinou vývojových prostředí. Nevýhodou je vysoká cena pohybující se nad hranicí 250€.

Obrázek 6. JTAG programátor J-LINK V8 [9]

2.6 JTAG programátor ST-LINK V2

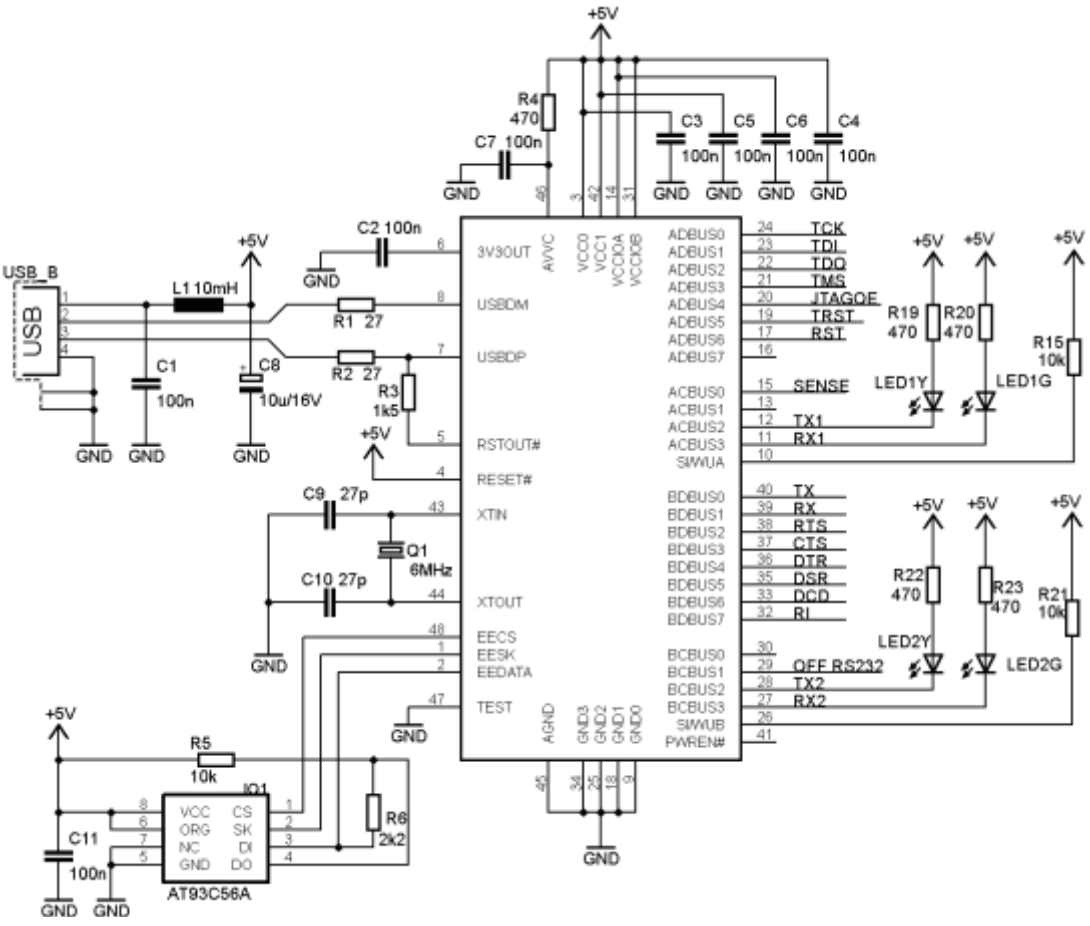
Tento programátor je již omezen na mikrokontroléry STM, ale velmi zajímavá je jeho cena, která se pohybuje pod 30€. Programátor lze použít s většinou komerčních i nekomerčních vývojových prostředí a podporuje JTAG i SWD.

Obrázek 7. JTAG programátor ST-LINK V2

2.7 JTAG programátor založený na obvodu FT2232

V poslední době se rozšířily i nekomerční programátory založené na integrovaném obvodu FT2232 firmy FTDI, který podporuje synchronní paralelní komunikaci. Nevýhoda je, že tento programátor není podporován tolika vývojovými prostředími a zprovoznění komunikace je také poměrně komplikované. Komunikace je většinou založena na projektu OpenOCD.

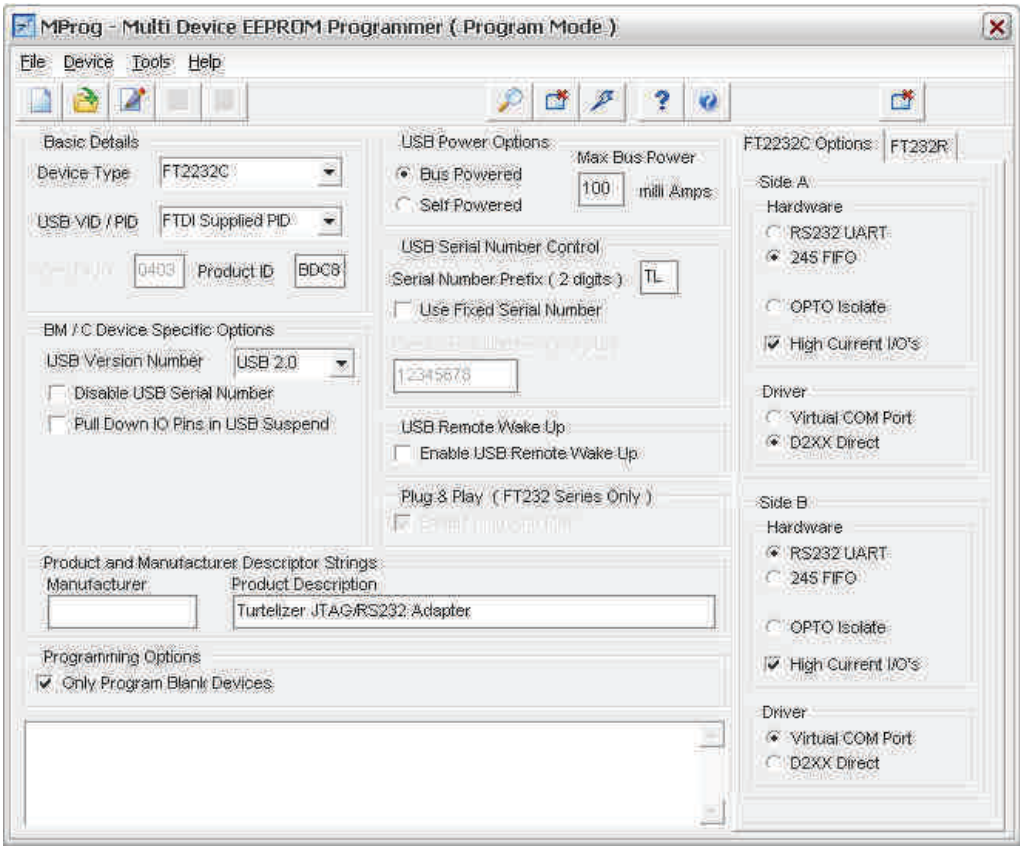
Velmi pěkný je návrh programátoru Turtelizer 2 [11], u tohoto programátoru jsou volně dostupné ovladače a návrh hardwaru [1]. Tento hardware jsem následně upravil, aby se vešel do krabičky KG 22M (72mm x 50mm x 22mm) a aby byla jednoduchá jeho výroba (jednostranný plošný spoj).



Obrázek 8. JTAG Programátor Turtelizer 2 [11]

Srdcem programátoru je zmiňovaný FT2232D, jedná se o převodník z USB na sériovou nebo paralelní synchronní (i asynchronní) komunikaci. Tento obvod obsahuje dva komunikační kanály, které kromě standardního UARTU podporují i MPSSE (Multi-Protocol Synchronous Serial Engine), jedná se o přímou podporu synchronních sériových komunikačních protokolů, jako jsou například JTAG, I2C a SPI.

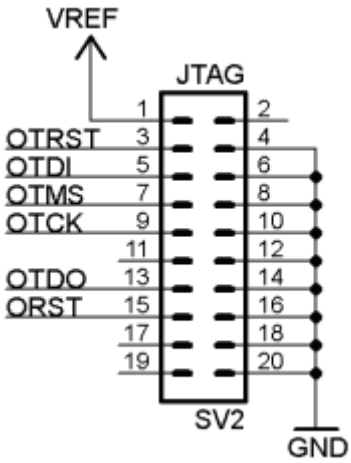
Převodník FT2232D je doplněn EEPROM pamětí AT93C56A, do které je pomocí konfiguračního programu MPROG nahrána konfigurace. Důležité je uvést ID produktu BDC8, jedná se o číslo, pomocí kterého se identifikuje hardware jako Turtelizer 2. Dále je potřeba kanál A přepnout na 245 FIFO a ovladače D2XX (komunikace je pak zajištěna pomocí driverů, které jsou ke stažení na stránkách autora [11]). Kanál B necháme přepnutý na RS232 UART a ovladače Virtual COM Port (kanál se pak tváří jako virtuální COM port). Celá konfigurace je na následujícím obrázku:



Obrázek 9. Konfigurace FT2232D

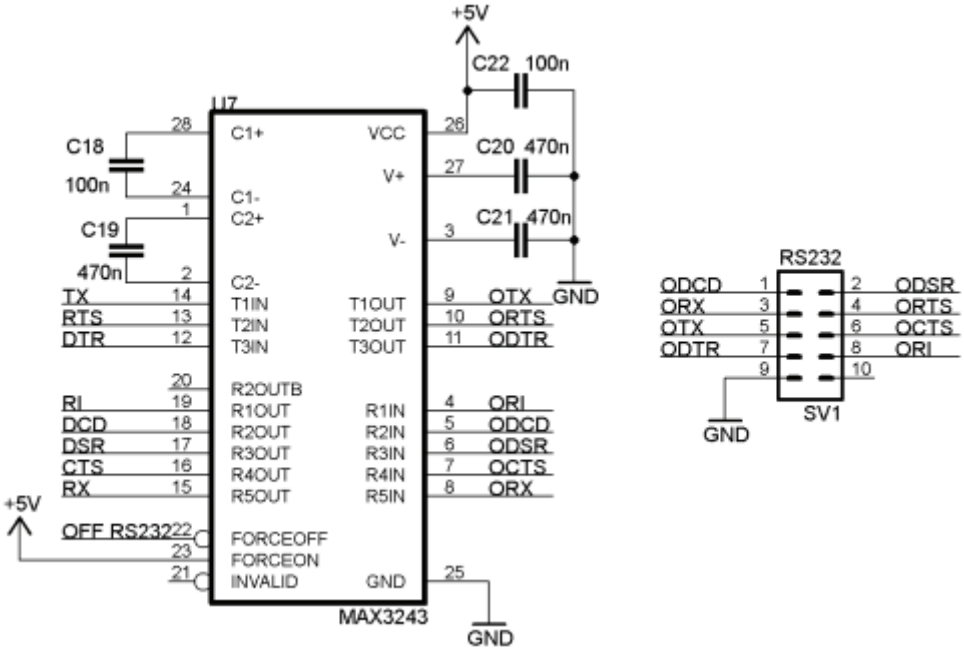
Aby bylo možné programovat ARM mikrokontroléry pracující s různým napájecím napětím, tak na vstupy a výstupy FT2232D jsou připojeny třístavové budiče sběrnice, které jsou schopny pracovat s napětím v rozsahu 1,8V – 5.5V. Budiče jsou zapojeny tak, aby převáděly napěťové úrovně, na kterých komunikuje mikrokontrolér, na úrovně, se kterými pracuje převodník FT2232D (5V logika).

Všechny signály potřebné pro JTAG komunikaci jsou vyvedeny na standardní dvaceti-pinový JTAG konektor.



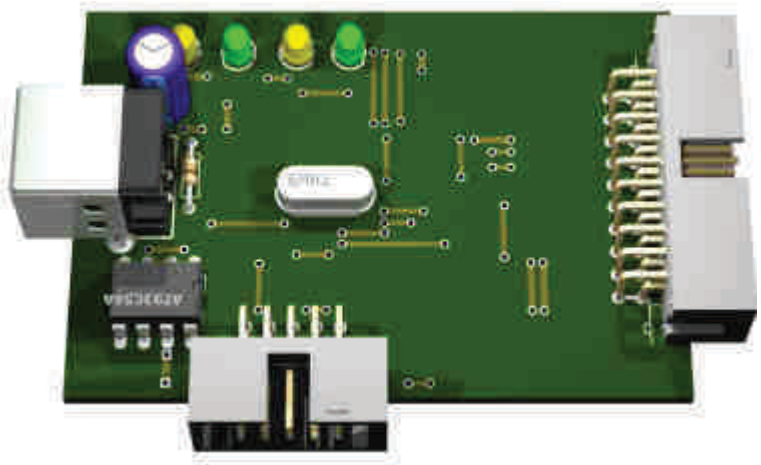
Obrázek 10. JTAG konektor programátoru Turtelizer 2

Druhý kanál převodníku FT2232D je použit pro převod USB komunikace na RS232. Jelikož FT2232D je schopen komunikovat v logických úrovních 0-5V a RS232 komunikuje v úrovních -15V a 15V, proto je potřeba použít převodník MAX3243, který řeší převod napěťových úrovní.

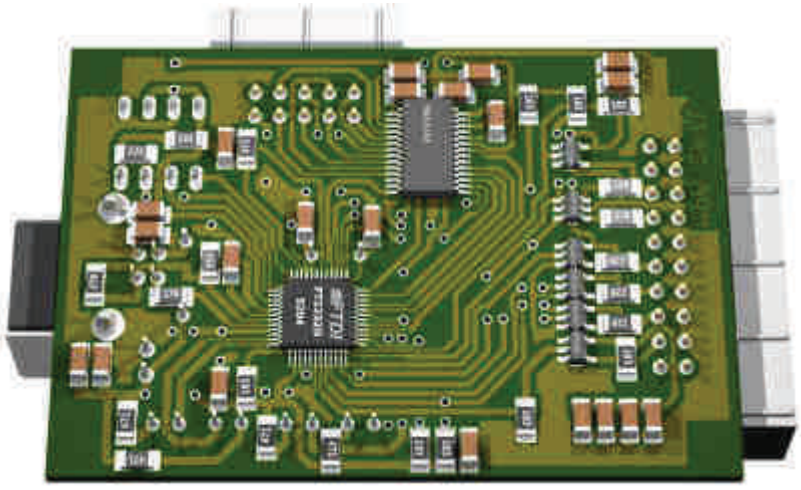


Obrázek 11. Převodník UART na RS232

Plošný spoj USB JTAG programátoru s převodníkem na RS232 byl navržen jako jednovrstvý s několika propojkami, přičemž musel být dodržen rozměr 68mm x 45mm. Pohled na DPS je na následujících dvou obrázcích:



Obrázek 12. JTAG programátor Turtelizer 2 pohled top

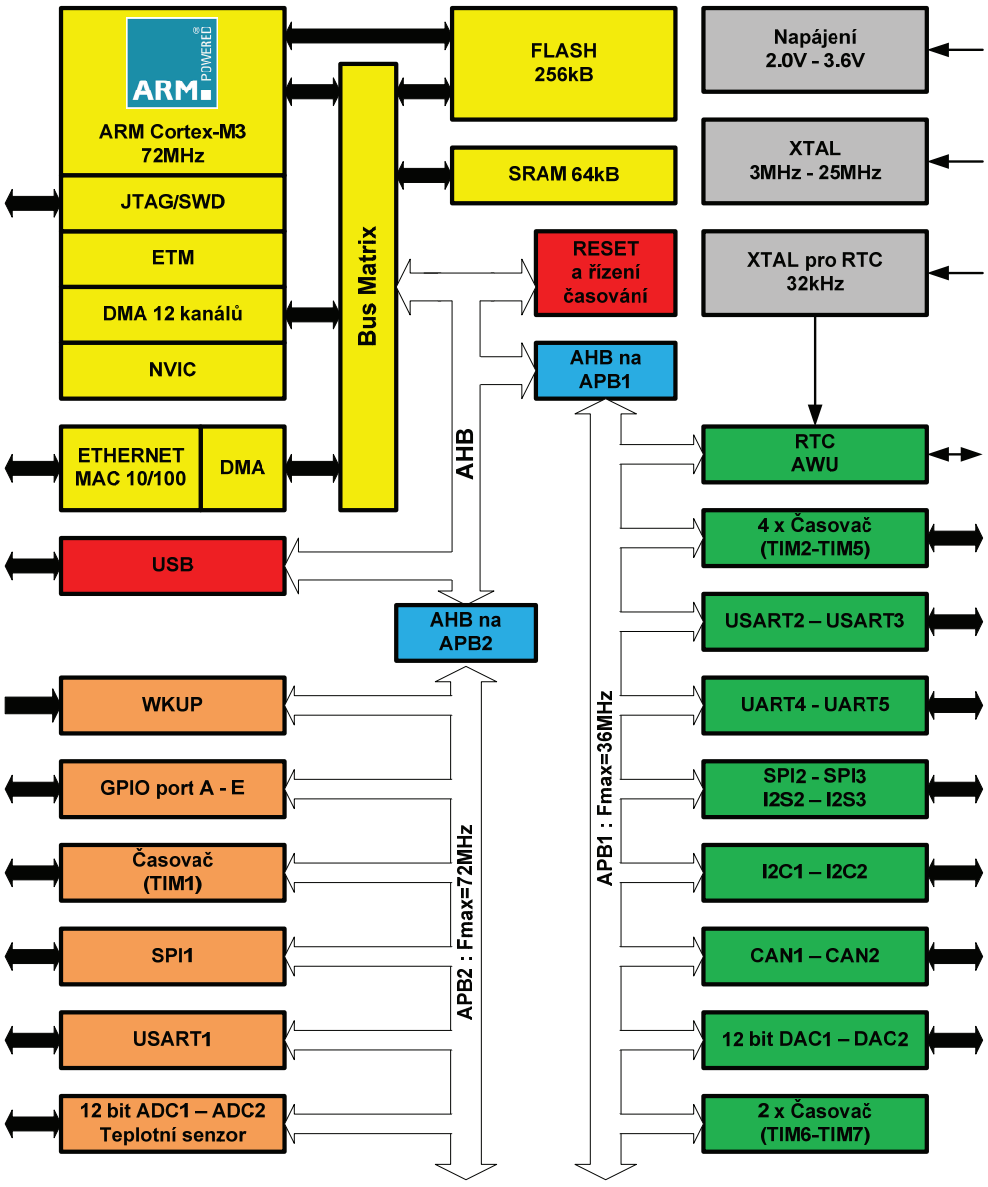


Obrázek 13. JTAG programátor Turtelizer 2 pohled bottom

2.8 Výběr JTAG programátoru a volba mikrokontroléru

Pokud požadujeme, aby JTAG programátor podporoval komunikaci přes USB, aby jeho cena byla co nejnižší a mohl být volně použit i pro komerční účely, tak máme na výběr pouze dvě varianty. První varianta je ST-LINK V2 a druhá je Turtelizer 2 nebo jakýkoliv jiný programátor založený na obvodu FT2232. Cenově obě varianty vychází téměř stejně, proto pokud se omezíme na mikrokontroléry firmy STM, tak ST-LINK V2 je pro nás jasná volba.

Jelikož jsem se přiklonil k použití ST-LINK V2 (podporuje JTAG a SWD) programátoru, tak jsem omezen na STM mikrokontroléry. Hlavním požadavkem na mikrokontrolér je, aby obsahoval ethernet. Jedním z mála vhodných mikrokontolerů je STM32F107VCT6. Struktura mikrokontroléru je na následujícím obrázku:



Obrázek 14. Vnitřní struktura mikrokontroléru STM32F107VCT6 [12]

Mikrokontrolér STM32F107VCT6 obsahuje jádro ARM Cortex – M3 taktované na frekvenci 72MHz, jádro podporuje programování přes JTAG a SWD a až 255 úrovní přerušení (NVIC). Mikrokontrolér disponuje 256kB FLASH paměti, 64kB SRAM, dvanácti kanálovým DMA a MAC vrstvou ethernetu s vlastním DMA. Všechny doteď uvedené součásti mikrokontroléru jsou připojeny přímo k jádru přes sběrnicový přepínač (Bus Matrix). Součástí mikrokontroléru je vysokorychlostní periferní sběrnice

AHB, ke které je připojeno USB a dvě sběrnice s periferiemi (připojeno k AHB pomocí dvou mostů). [12]

Sběrnice AHB2 je vysokorychlostní (až 72MHz) a sběrnice AHB1 disponuje poloviční rychlostí (až 36MHz). Ke sběrnici APB2 jsou připojeny vstupně/výstupní porty (GPIO A-E), vstup pro probouzení mikrokontroléru (WKUP), jeden časovač (TIM1), SPI rozhraní (SPI1), univerzální synchronní/asynchronní rozhraní (USART1), teplotní senzor a dva dvanácti-bitové AD převodníky (ADC1,ADC2). [12]

Sběrnice APB1 pracuje na nižších rychlostech, ale zase nabízí více periferií, které tuto sběrnici sdílejí. Ke sběrnici APB1 jsou připojeny hodiny reálného času (RTC), šest šestnácti bitových časovačů (TIM2-TIM7), dvě univerzální synchronní/asynchronní rozhraní (USART2,USART3), dvě univerzální asynchronní rozhraní (UART4,UART5), dvě sériové periferní rozhraní (SPI2,SPI3), dva kanály I2S pro audio vstup a výstup (I2S2,I2S3), dvě I2C sběrnice pro připojování periferií (I2C1,I2C2), dvě CAN sběrnice (CAN1,CAN2) a dva dvanácti-bitové DA převodníky (DAC1, DAC2). [12]

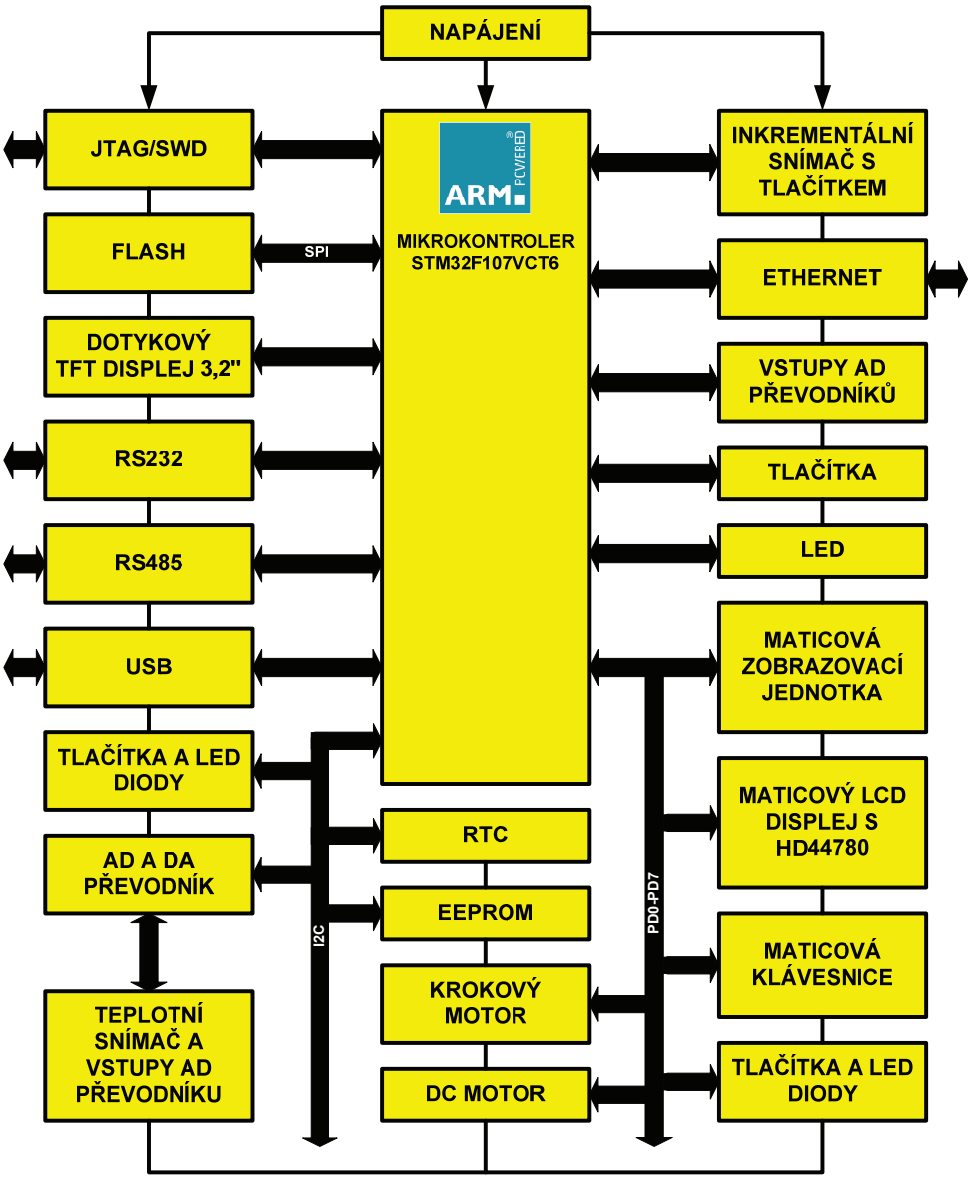
Mikrokontrolér je napájen napětím 2.0V – 3,6V, je taktován interním RC oscilátorem nebo externím krystalovým oscilátorem (3MHz - 25MHz). Pro RTC je potřeba připojit externí krystal 32,768kHz. [12]

3 HARDWARE

V této části práce je popsán návrh hardwaru zařízení. Celý hardware je postaven na mikrokontroléru ARM STM32F107VCT6. Při návrhu byl kladen důraz na co největší množství periferií.

3.1 Blokové schéma konstrukce

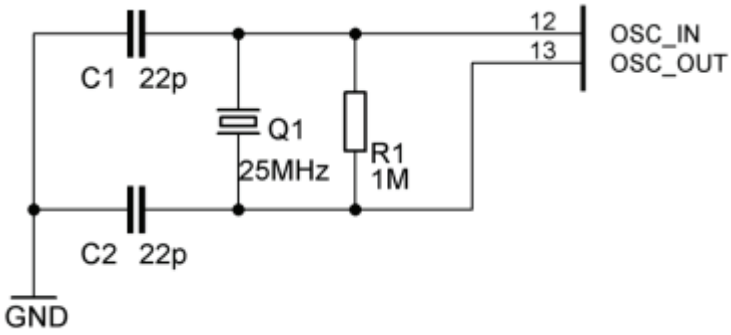
Celá konstrukce se skládá ze dvaceti čtyř částí. Srdcem celé konstrukce je řídicí prvek (mikrokontrolér STM32F107VCT6). Dále je zde napájecí část, I2C sběrnice s množstvím periferií, dotykový displej, ethernet, SPI FLASH paměť atd..



Obrázek 15. Blokové schéma zapojení

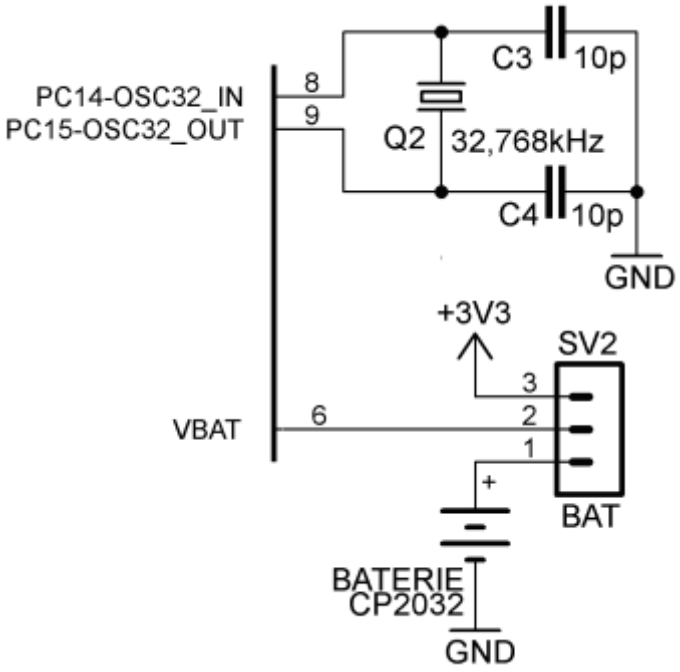
3.2 Mikrokontrolér STM32F107VCT6

Časování mikrokontroléru je realizováno pomocí krystalu Q1 (25MHz), který je připojen k vývodům OSC_IN a OSC_OUT (vstup a výstup invertujícího zesilovače) a přes blokační kondenzátory C1 a C2 je připojen na zem. Velikost kondenzátorů C1 a C2 se volí v rozsahu 30pF±10pF. [12]



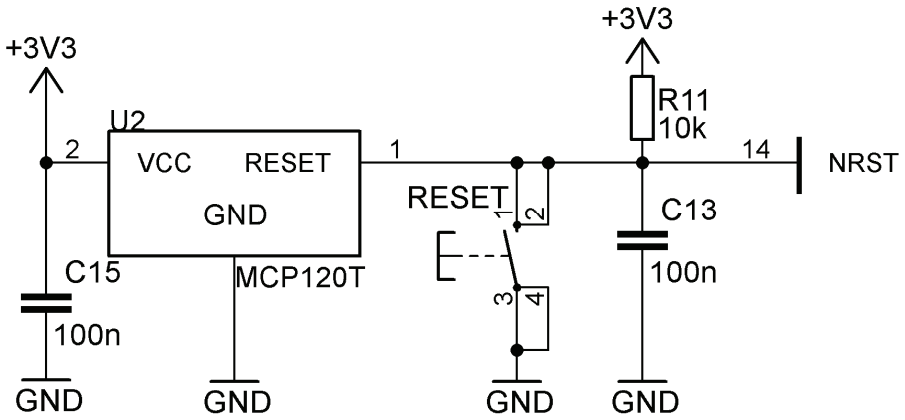
Obrázek 16. Mikrokontrolér - připojení krystalu 25MHz [12]

Jelikož mikrokontrolér obsahuje hodiny reálného času, tak je nutné připojit externí krystal, který slouží pro taktování RTC obvodu. Při odpojení mikrokontroléru se o napájení RTC obvodu stará externí baterie. [12]



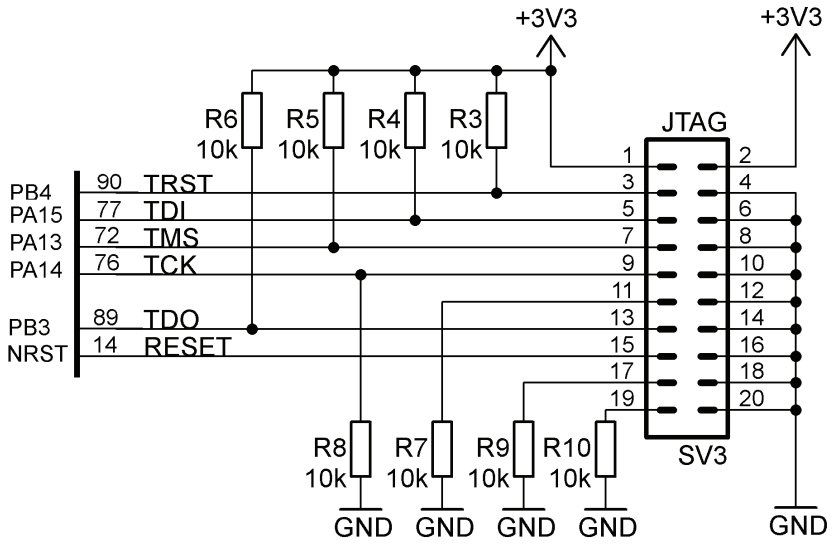
Obrázek 17. Mikrokontrolér - krystal a napájení RTC [12]

Po připojení napájecího napětí k mikrokontroléru je mikrokontrolér resetován pomocí resetovacího obvodu MCP120T (výstup MCP120T je držen v log. 0), jakmile dojde k ustálení napájecího napětí, tak se resetovací obvod přepne (výstup MCP120T je držen v log. 1). Na pin NRST mikrokontroléru je připojen také JTAG programátor a tlačítko RESET (slouží pro ruční reset mikrokontroléru). [14]



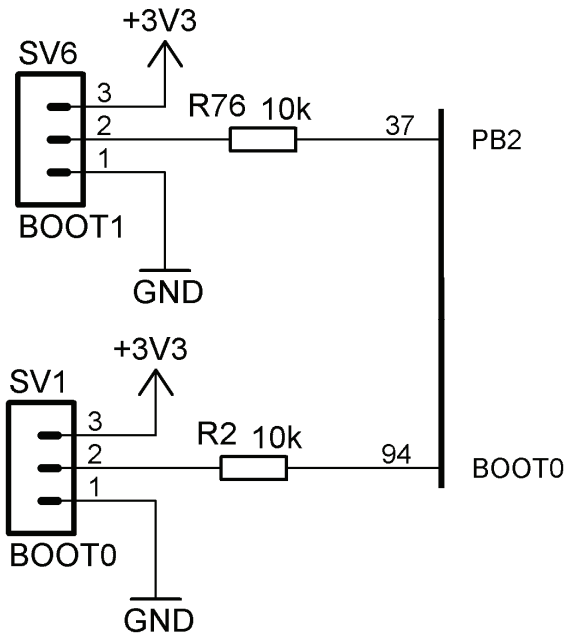
Obrázek 18. Mikrokontrolér - resetovací obvod [14]

K programování a krokování programu v mikrokontroléru jsou vyvedeny signály JTAGu, popřípadě je možno použít SWD, jehož signály jsou vyvedeny na stejné piny mikrokontroléru jako JTAG. [12][15]



Obrázek 19. Mikrokontrolér – JTAG [12][15]

Poslední zajímavou věcí v řídicí části je bootloader mikrokontroléru STM32F107VCT6. Bootloader je schopen nahrát program do mikrokontroléru přes USB, USART1, USART2 nebo přes CAN2. [13]



Obrázek 20. Mikrokontrolér - Aktivace bootloaderu [12][13]

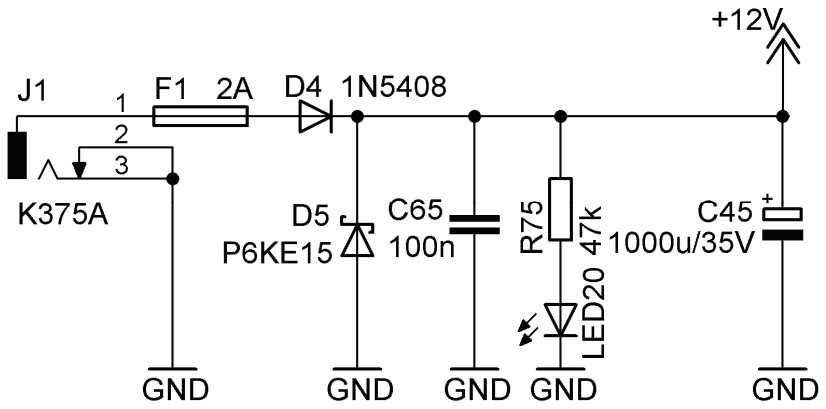
Bootloader se aktivuje tím, že při resetu mikrokontroléru je BOOT1=0 a BOOT0=1. Pokud chceme, aby se po resetu začal zpracovávat program, uložený ve flash, tak musí být BOOT0=0 a na logické úrovni BOOT1 nezáleží. [13]

3.3 Napájení

V této části je řešeno napájení celé aplikace. K napájení mikrokontroléru je potřeba napětí 3,3V a pro některé periferie 5V. V této části byly vybrány DC/DC měniče podle předpokládané zátěže.

3.3.1 Vstupní obvody

Celá aplikace bude napájena 12V externím zdrojem. Na vstupu napájecí části vývojového kitu je použita ochrana proti přepólování (dioda D4). Dále je na vstup připojen transil (D5 P6KE15), který slouží jako ochrana před napěťovými špičkami (pokud na transil přijde napěťová špička, tak se transil zkratuje). Nakonec je na vstupu přidán kondenzátor C45 pro pokrytí odběrových špiček. Zapojení je doplněno menším kondenzátorem C65, který odfiltruje vysoké kmitočty. Pro indikaci napájení je na vstup přidána indikační LED dioda (LED20). Pro ochranu napájecího adaptéru je použita pojistka F1, která v případě zkratu na desce chrání napájecí adaptér před zničením.



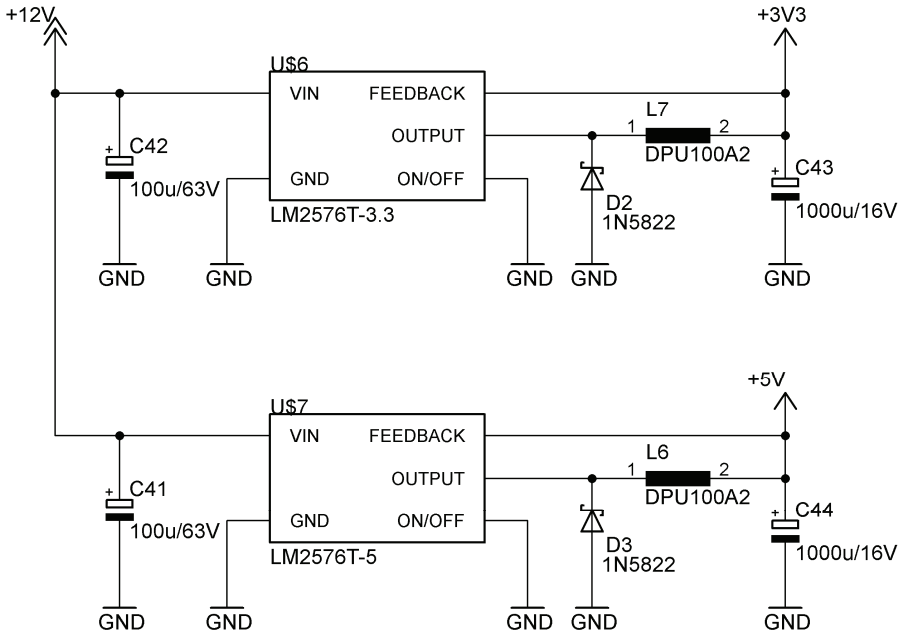
Obrázek 21. Napájení - Vstupní obvody

3.3.2 Měnič napětí z 12V na 3,3V a 5V

Pro napájení mikrokontroléru a některých periferií je potřeba napájecí napětí 3,3V, proto je vstupní napětí 12V pomocí DC/DC měniče LM2576T-3.3 sníženo na požadovaných 3,3V. Maximální proudový odběr je 3A. [16]

Pro napájení DC motoru, krokových motorů a například maticového LCD displeje s řadičem HD44780 je potřeba napájecí napětí 5V, proto byl do zapojení přidán druhý DC/DC měnič LM2576T-5, jehož maximální výstupní proud je 3A. [16]

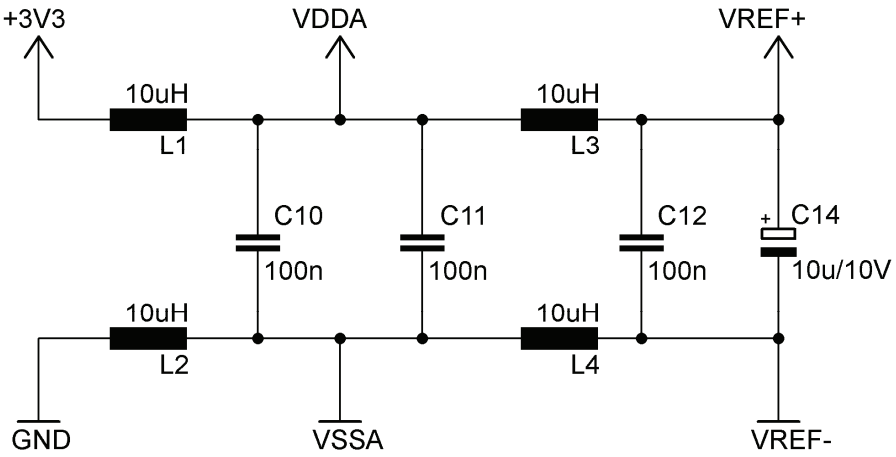
Měniče LM2576T dosahují vysoké účinnosti, pokud by odběry byly do 1A, tak je výhodnější použít lineární stabilizátor, protože účinnost u DC/DC měničů je závislá na odebíraném proudu. [16]



Obrázek 22. Napájení - Zdroj 3.3V a 5V [16]

3.3.3 Filtr pro mikrokontrolér STM32F107VCT6

Dle doporučení výrobce mikrokontroléru je potřeba mezi jednotlivé napájecí vstupy vložit filtry. Napájecí vstup VDDA slouží pro napájení AD převodníků a VREF+ je referenční hodnota napětí AD převodníků.



Obrázek 23. Napájení – Filtry [12]

3.3.4 Dimenzování DC/DC měničů

Pro použití vhodných DC/DC měničů potřebujeme znát maximální odběry jednotlivých obvodů připojených na daný DC/DC měnič. Uděláme si proto odběrovou bilanci celého zařízení.

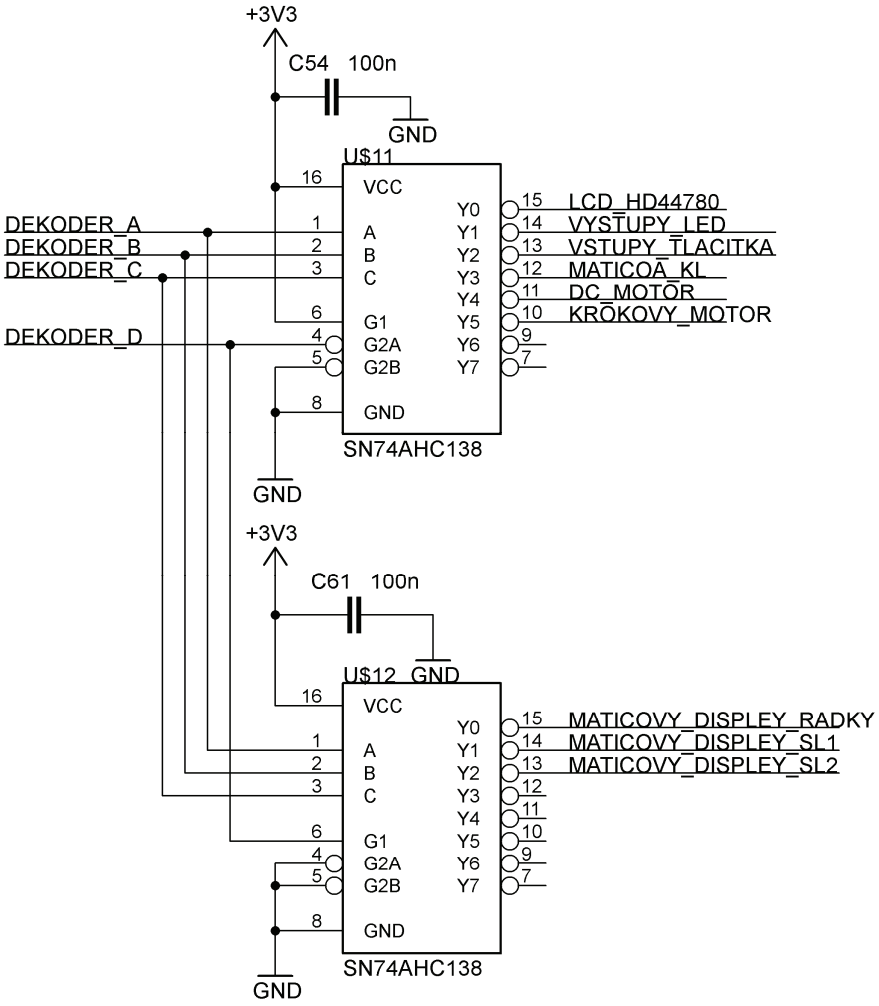
DC/DC měnič LM2576T-3,3 (3,3V/3A)		**DC/DC měnič LM2576T-5 (5V/3A)**	
STM32F107VCT6	350mA	L293D	2x1,2A
SN74AHC138	2x4mA	LCD HD44780	2mA
SN74LVC573	6x24mA	SN74LVC4245	3x24mA
SN74LVC4245	3x12mA		
TFT SSD1289	100mA		
PCA9554	2x85mA		
PCF8563	50mA		
AT24C128	5mA		
PCF8591	10mA		
MAX3232	1mA		
MAX3485	1mA		
DP83848	100mA		
AT450DB81D	15mA		
Celkem	990mA	Celkem	2450mA
Rezerva	2010mA	Rezerva	550mA

Tabulka 1. Maximální odběry z jednotlivých napájecích větví

Z tabulky je vidět, že vybrané DC/DC měniče plně dostačují. Do tabulky byly zaneseny kritické hodnoty spotřeby jednotlivých obvodů, udávané v katalogových listech. Ve skutečnosti bude spotřeba podstatně nižší, nikdy nenastane případ, že by běžely všechny periferie zároveň nebo na maximální výkon.

3.4 Periferie připojené k PD0-PD7 mikrokontroléru

Velikost pouzdra mikrokontroléru není neomezená, proto některé periferie musí sdílet vstupně/výstupní piny mikrokontroléru. Pro tento účel byly vybrány V/V piny PD0-PD7. Jelikož je potřeba nějakým způsobem řídit přístup jednotlivých periferií k PD0-PD7, tak pro tento účel byly do zapojení vloženy dva dekodéry 1 z N, které jsou ovládány piny PC0,PC6,PC7 a PC8 mikrokontroléru, pomocí dekodérů jsou jednotlivé periferie připojovány k V/V pinům PD0-PD7 mikrokontroléru.

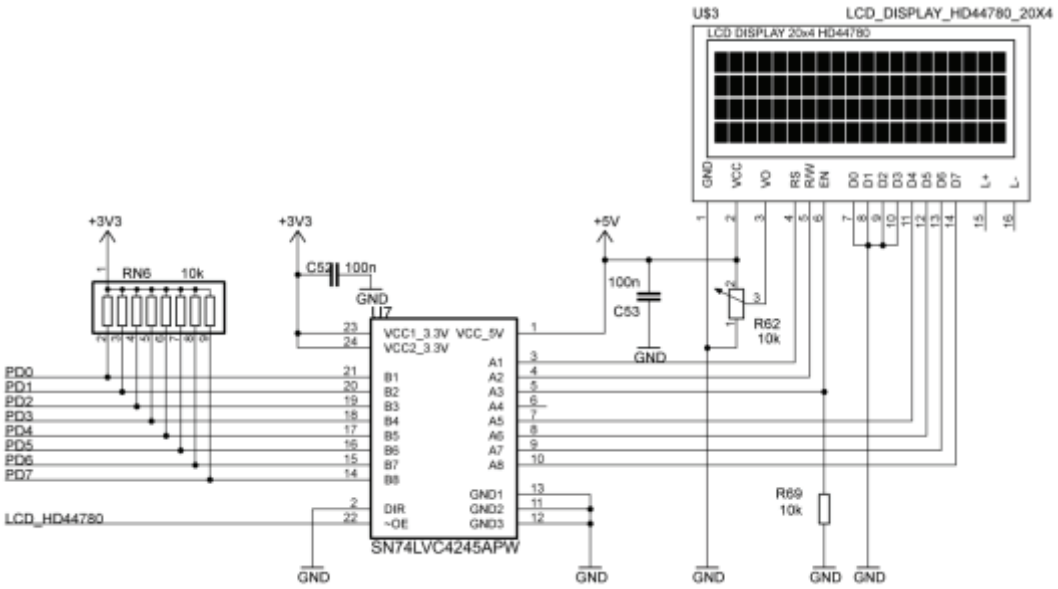


Obrázek 24. Dekodér 1 z N [17]

Pro adresaci periferií jsou k dispozici 4 adresové vodiče, proto je možno adresovat až 16 periferií. Byly použity dva adresové dekodéry SN74AHC138, které pracují se sníženým napětím 3,3V. U těchto dekodérů je vždy aktivní pouze jeden výstup (aktivní je v log 0). V další části budou popsány periferie, které jsou adresovány pomocí těchto dekodérů. [17]

3.4.1 Alfanumerický LCD displej s řadičem HD44780

Mezi standardně používané zobrazovací prvky patří alfanumerické LCD displeje s řadičem HD44780. Tyto displeje slouží pro zobrazování textu. Znaková sada je pevně definována a několik znaků lze dodefinovat (používá se pro zobrazení speciálních českých znaků). Komunikace s displejem je řešena pomocí 3 řídících vstupů a 8 datových vstupů. Prvním řídícím signálem je R/W, tímto vstupem se určuje, zda se bude z displeje číst nebo do něj zapisovat. Signálem RS se určuje, zda posílaná data představují řídící příkaz nebo data. Posledním řídícím signálem je EN, který slouží pro potvrzení dat na vstupech displeje. Jak již bylo řečeno, tak tento displej má 8 datových vstupů, které lze využít pro 8 bitovou nebo 4 bitovou komunikaci. [19]

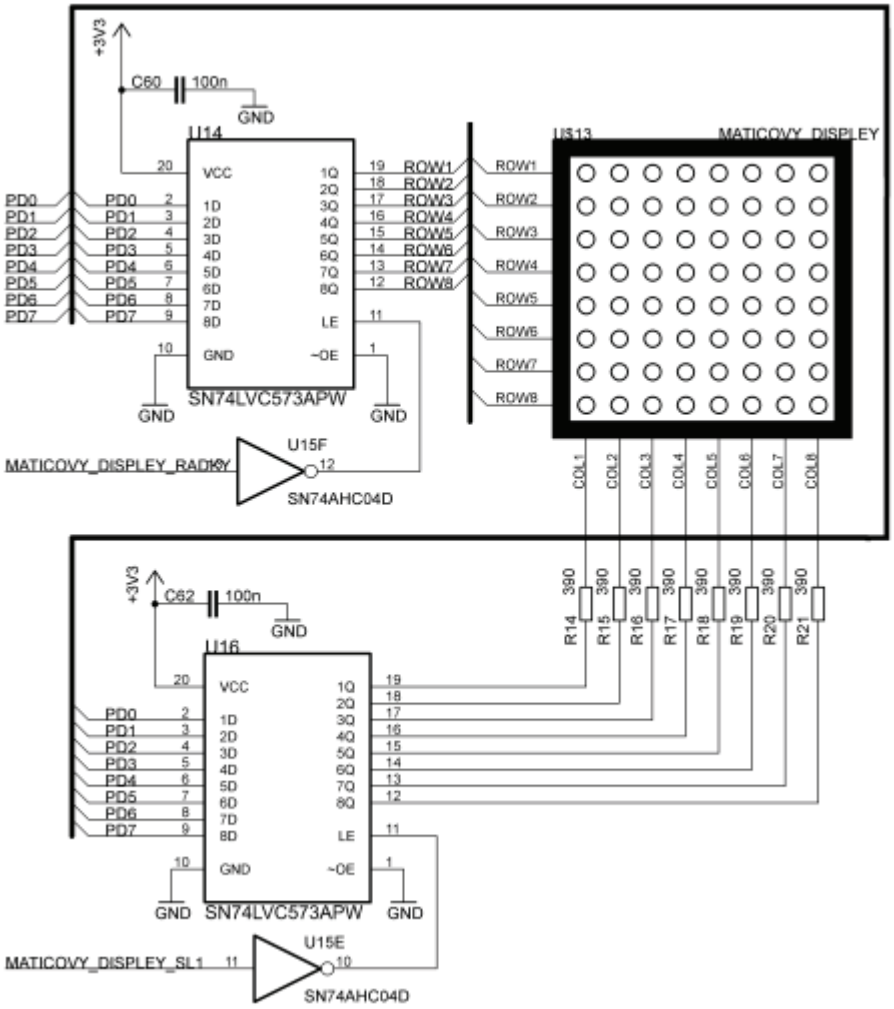


Obrázek 25. Alfanumerický LCD displej s řadičem HD44780 [18][19]

V tomto konkrétním zapojení bude displej používán ve 4 bitovém režimu z důvodu úspory vodičů. Jelikož displej podporuje komunikaci pouze v úrovních TTL (0-5V), tak musel být k displeji přidán měnič úrovní SN74LVC4245APW, protože mikrokontrolér pracuje s napětím maximálně 3,3V. [18]

3.4.2 Maticová zobrazovací jednotka

Mezi méně používané displeje patří maticové zobrazovací prvky. Jedná se o LED diody zapojené do matice. Mezi největší nevýhody těchto displejů patří velká spotřeba, ale nabízejí také výhody. Největší výhodou je vynikající čitelnost i za slunného počasí. Na následujícím obrázku je zapojení jedné matice:

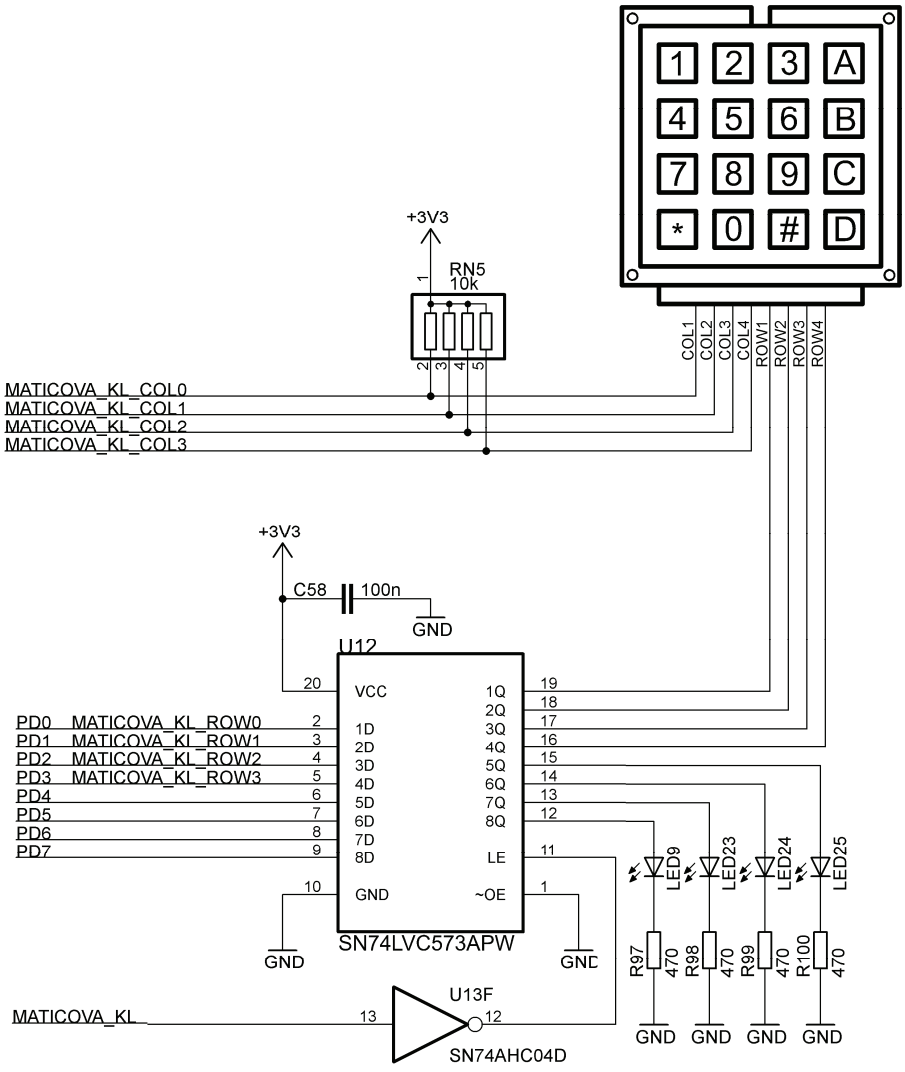


Obrázek 26. Maticová zobrazovací jednotka [20][21]

Obsluha tohoto displeje je poněkud složitější, protože data na displeji musí být obnovována. Pro obsluhu jedné matice jsou použity dva záchytné registry SN74LVC573APW. Jeden registr slouží pro obsluhu řádků a druhý pro obsluhu sloupců maticového displeje.

3.4.3 Maticová klávesnice

Jako vstupní jednotku lze použít například maticovou klávesnici, jejíž výhodou je, že pro čtení stavu 16 tlačítek potřebujeme pouze 8 vodičů. Princip čtení je velmi jednoduchý, postupně přizemňujeme jednotlivé řádky a čteme hodnoty sloupů. Jakmile v nějakém sloupci detekujeme nízkou úroveň napětí, tak bylo stisknuto tlačítko. Například tlačítko C je detekováno jako stisknuté, když na ROW1-ROW4 je přivedena kombinace 1110 a zároveň na COL1-COL4 je přečtena kombinace 1101.

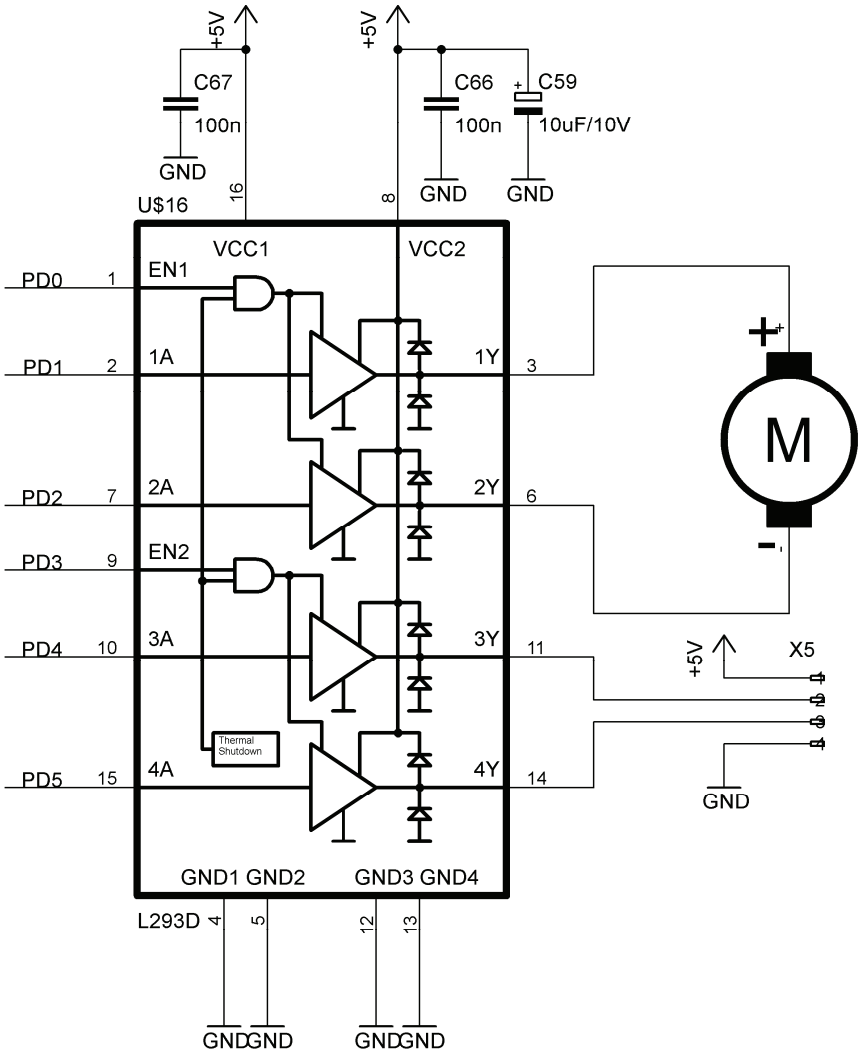


Obrázek 27. Maticová klávesnice [21][12]

Posílání dat na řádky maticové klávesnice je řešeno pomocí záchytného registru U12 a sloupce klávesnice jsou připojeny přímo k mikrokontroléru (PB0, PB1, PB14 a PB15). K záchytnému registru U12 jsou připojeny ještě 4 indikační LED diody, které lze využít pro jiné účely.

3.4.4 Stejnosměrný motor

Pro řízení stejnosměrných motorů se používají h-můstky. Jedná se o výkonové prvky, které se používají k řízení motorů. H-můstek můžeme složit z tranzistorů nebo využít některý z integrovaných obvodů, který v sobě většinou obsahuje i ochranné diody. Jedním z těchto obvodů je například L293D. Pomocí tohoto obvodu lze řídit směr otáčení motoru a jeho rychlost pomocí PWM. Pro roztočení motoru musíme PD0 nastavit do log 1, PD2 do log 0 a následně rychlost otáčení řídíme pomocí PWM na vstupu PD1. Zapojení h-můstku je na následujícím obrázku:

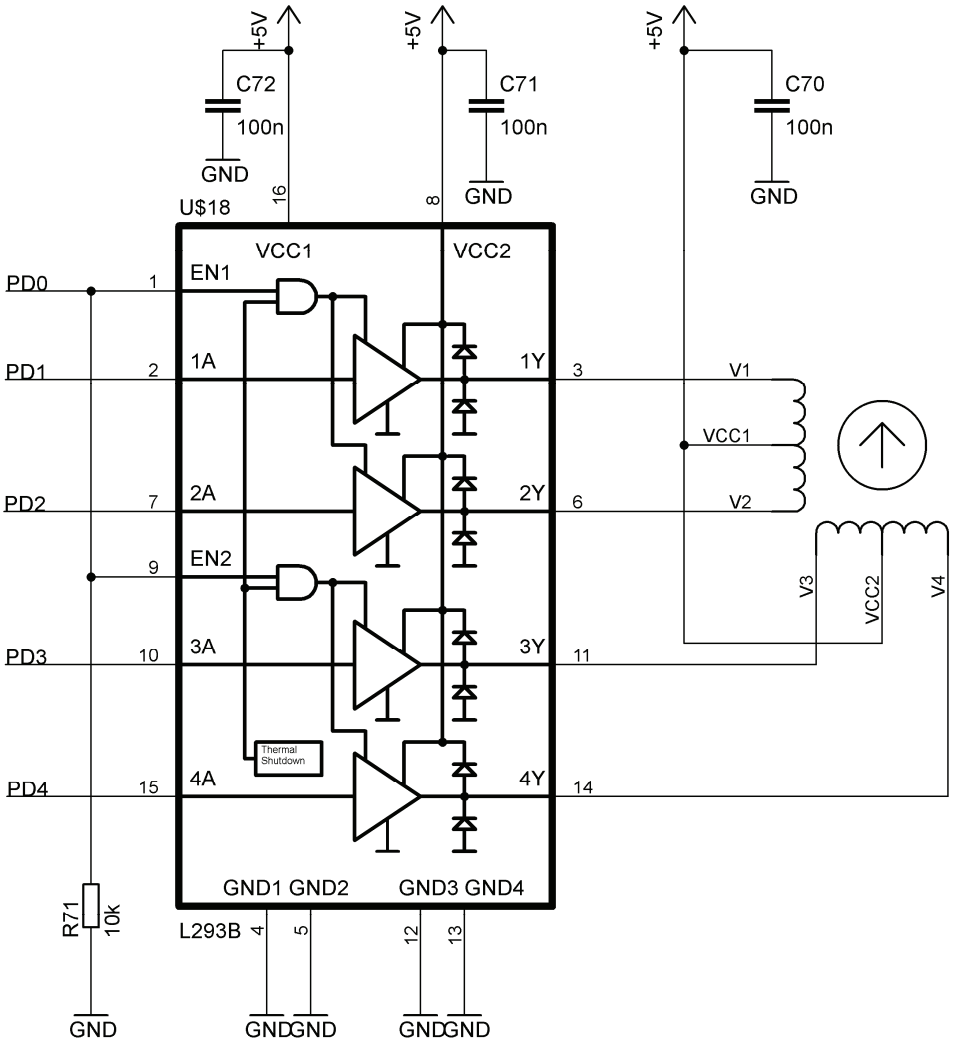


Obrázek 28. Stejnosměrný motor [22]

H-můstek pracuje s napětím logiky 5V, proto je k pinům PD0-PD7 připojen pomocí měniče napěťových úrovní SN74LVC4245APW obdobně jako alfanumerický LCD displej.

3.4.5 Krokový motor

Dalším typem motoru, který se používá především jako ekvivalent servopohonů, je krokový motor. Tyto motory lze použít pro přesné natáčení hřídele bez použití snímače polohy (poloha se dá dopočítat z počtu kroků). Krokový motor byl zapojen tak, aby bylo možné jej řídit unipolárně (proud v jeden okamžik teče pouze jednou větví). Při unipolárním řízení je sice moment motoru menší, ale zároveň je nižší spotřeba.



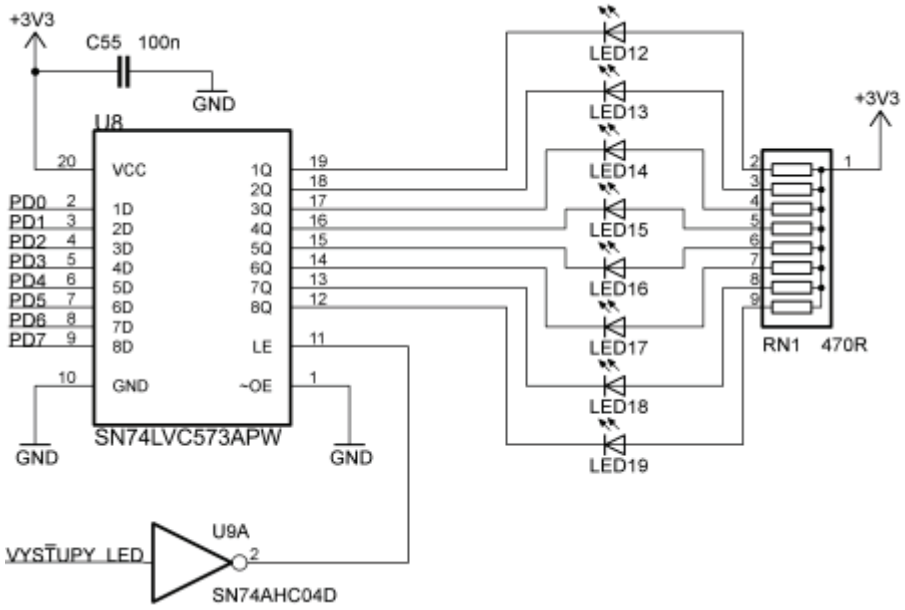
Obrázek 29. Krokový motor [23][24]

Krokový motor je k pinům PD0-PD4 připojen stejným způsobem jako stejnosměrný motor.

3.4.6 Tlačítka a LED diody

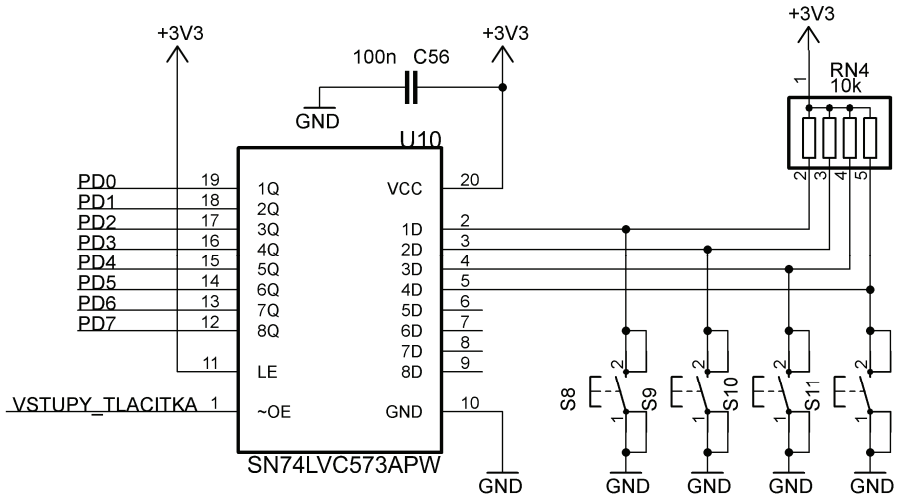
U výukového kitu samozřejmě nesmí chybět množství LED diod a tlačítek. LED diody a tlačítka jsou k pinům PD0 – PD7 připojeny pomocí záchytného registru SN74LVC573APW.

K rozsvícení diody dochází přivedením logické nuly na příslušný pin PD. Zapojení LED diod je na následujícím obrázku:



Obrázek 30. LED diody připojené k PD0-PD7 [20]

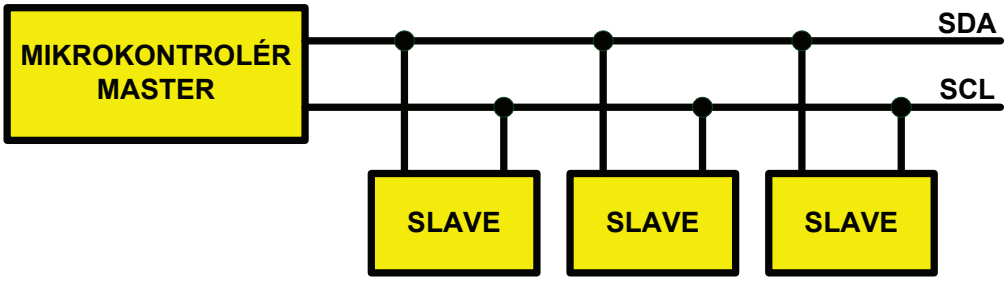
Příslušné tlačítko je stisknuté, pokud na daném pinu portu PD je logická 0. Zapojení tlačítek je na následujícím obrázku:



Obrázek 31. Tlačítka připojená k PD0-PD3 [20]

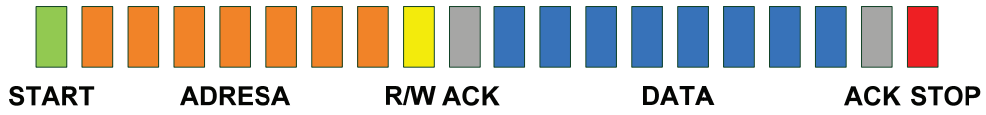
3.5 I2C sběrnice

Jedná se o dvouvodičovou sériovou datovou sběrnici, na kterou je možno připojit až 128 zařízení. Na sběrnici je možno připojit více mikrokontrolérů (masterů), ale ve většině případů si vystačíme s jedním masterem. Oba datové vodiče musí být přes pull-up rezistory připojeny na napájecí napětí. Vodič SCL slouží pro časování a SDA pro data. Jsou stanoveny i normy, které definují rychlosti komunikace na I2C. Nejběžnější minimální rychlosti komunikace I2C zařízení jsou 100kHz a 400kHz.



Obrázek 32. I2C sběrnice

Komunikace na sběrnici s jedním masterem je poměrně jednoduchá, v době kdy nejsou po datové sběrnici přenášena data, tak jsou SDA a SCL v logické jedničce. Komunikaci zahájí master přivedením logické nuly na SDA (jedná se o START bit), následně je vyslána sedmibitová adresa (adresa slave zařízení) a bit určující zda se bude do slave zařízení zapisovat nebo číst (R/W bit), následně jsou odeslána nebo přijata data (8 bitů) zařízením, po přijetí dat je vygenerováno potvrzení přijetí (ACK) a master vygeneruje STOP bit. Průběh komunikace je naznačen na následujícím obrázku:



Obrázek 33. I2C průběh komunikace na sběrnici

I2C sběrnice je vhodná pro komunikaci v rámci jednoho zařízení (komunikace na krátké vzdálenosti), proto i nabídka integrovaných obvodů komunikujících po I2C je velmi omezena. Mezi základní obvody komunikující po I2C patří například AD převodníky, DA převodníky, expandéry, RTC, snímače vzdálenosti, snímače teploty, SRAM a EEPROM.

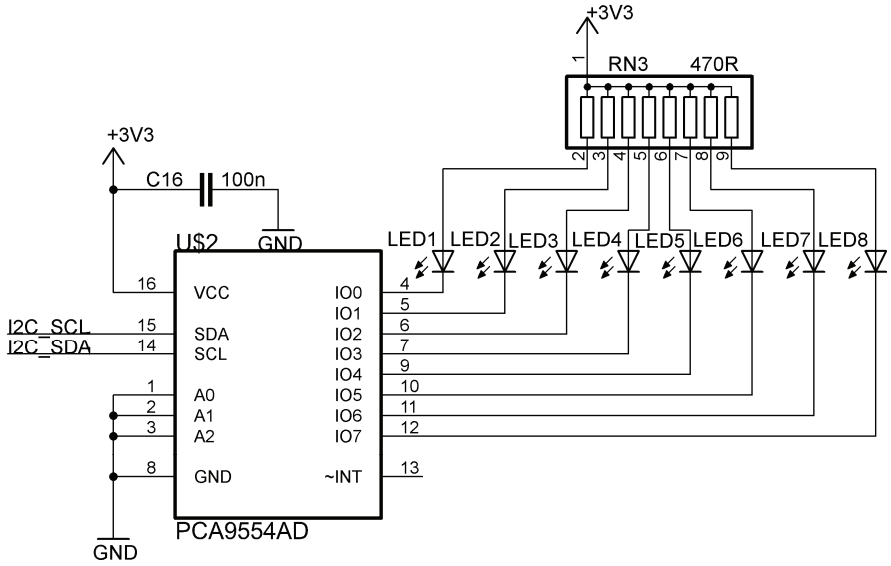
3.5.1 Tlačítka a LED diody připojené pomocí expandérů

Pro připojení tlačítek a led diod k I2C se používají expandéry, v tomto případě byl použit 8 bitový expandér PCA9554. Tento expandér je schopen komunikovat rychlostí 400kHz a z jeho výstupů lze číst i na ně zapisovat. Adresu expandéru lze měnit pomocí vstupů A2,A1 a A0, přičemž pevná část adresy je 0100. [25]

0	1	0	0	A2	A1	A0	RW

Obrázek 34. Adresa expandéru [25]

Pro LED diody je použita adresa 0100000 (vstupy A2, A1 a A0 jsou připojeny k logické nule). Anody LED diod jsou připojeny k napájecímu napětí přes 470Ω a katody jsou připojeny k expandéru PCA9554 (diody se rozsvítí přivedením logické nuly na příslušný výstup expandéru). [25]



Obrázek 35. I2C připojení LED k expandéru PCA9554 [25]

Tlačítka jsou připojena k expandéru s adresou 0100001 (vstupy A2 a A1 jsou připojeny k logické nule a A0 je připojen k logické jedničce). Zapojení je obdobné jako u LED diod. Pokud je na příslušném vstupu expandéru logická nula, tak je dané tlačítko stisknuto. [25]

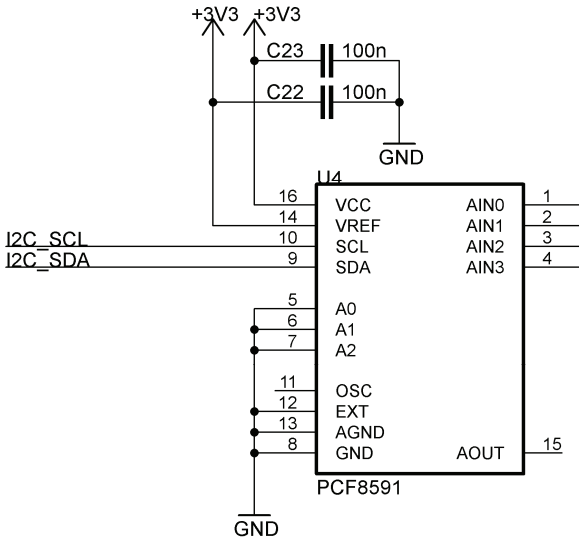
3.5.2 AD a DA převodníky

Pokud potřebujeme měřit nějaké analogové veličiny, tak můžeme použít AD převodník PCF8591, který má také jeden DA převodník. Tento AD DA převodník je schopen komunikovat po I2C rychlostí 400kHz. Adresu AD DA převodníku lze měnit pomocí vstupů A2,A1 a A0, přičemž pevná část adresy je 1001. [26]

1	0	0	1	A2	A1	A0	RW

Obrázek 36. Adresa AD a DA převodníku [26]

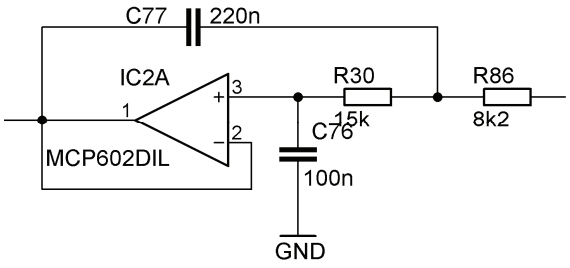
Pro AD a DA převodník je použita adresa 1001000 (vstupy A2, A1 a A0 jsou připojeny k logické nule). Referenční napětí AD převodníků je 3,3V. Celkem má tento převodník čtyři multiplexované vstupy AD převodníku. Na první vstup je připojen teplotní senzor, na druhý a třetí vstup je připojen potenciometr a na čtvrtý vstup je připojen výstup DA převodníku tohoto obvodu. [26]



Obrázek 37. I2C AD a DA převodník [26]

Teplotní senzor TMP36 je analogový teplotní senzor, který je v tomto zapojení napájen 3,3V a jeho výstupní hodnota napětí se v závislosti na teplotě pohybuje v rozsahu -18mV (pro -40°C) až 315mV(pro 125°C). Výstupní napětí teplotního snímače je následně zesíleno 10x pomocí operačního zesilovače a pomocí aktivního filtru je odfiltrován vysokofrekvenční šum. Takto upravený signál je přiveden na první vstup AD převodníku PCF8591. [26][27]

Mezi vstupy AD převodníku a měřeného zdroje napětí je přidána aktivní dolní propust typu Sallen Key, tato propust slouží k odfiltrování vysokofrekvenčního šumu.



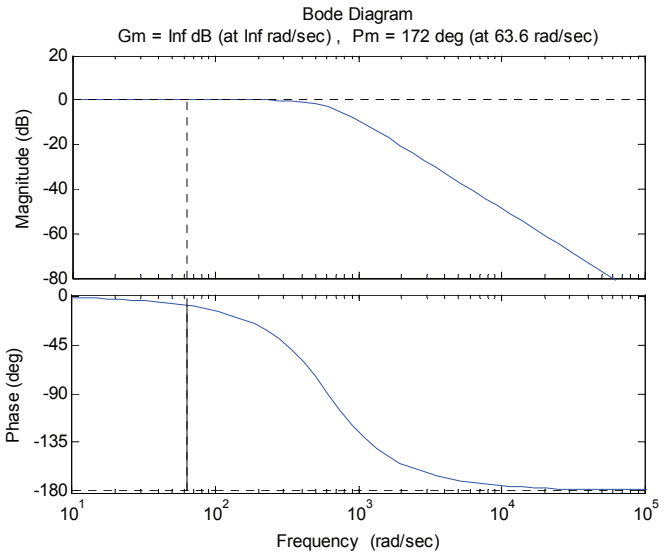
Obrázek 38. Dolní propust Sallen Key [31]

Dolní propust slouží k odfiltrování signálů o frekvenci větší než 100kHz. Přenos navrženého filtru Sallen Key je:

$$F(p) = \frac{1}{C_{14}C_{15}R_3R_4p^2 + (R_3 + R_4)C_{15}p + 1} \qquad (1)[31]$$

$$F(p) = \frac{1}{2.706 \cdot 10^{-6}p^2 + 0.00232p + 1} \qquad (2)[31]$$

Navržený filter byl ještě simulován v matlabu. Z výsledků je vidět, že filter tlumí signály o frekvenci větší než 100kHz. V případě, že potřebujeme strmější charakteristiku, tak zvýšíme řád filtru.



Obrázek 39. Frekvenční charakteristika dolní propusti typu Sallen Key

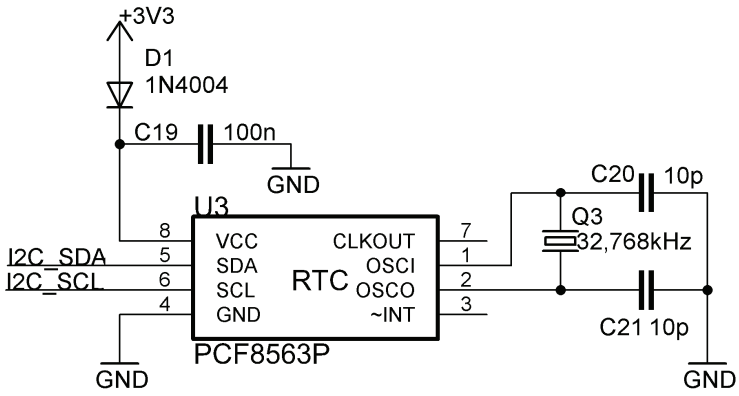
3.5.3 Hodiny reálného času (RTC)

U většiny dnešních zařízení je potřeba znát aktuální čas, z tohoto důvodu se používají hodiny reálného času (RTC), které v sobě uchovávají aktuální čas (roky, měsíce, dny, hodiny, minuty a sekundy). Pro tento účel byl vybrán RTC obvod PCF8563, který je schopen komunikovat rychlostí 400kHz. Adresu RTC nelze měnit, je dána napevno (1010001). [29]

1	0	1	0	0	0	1	RW

Obrázek 40. Adresa RTC obvodu [29]

Pro taktování hodin reálného času se používá krystal 32,768kHz, který se přes kondenzátory C20 a C21 připojuje k zemi.



Obrázek 41. I2C Hodiny reálného času (RTC) [29]

3.5.4 EEPROM

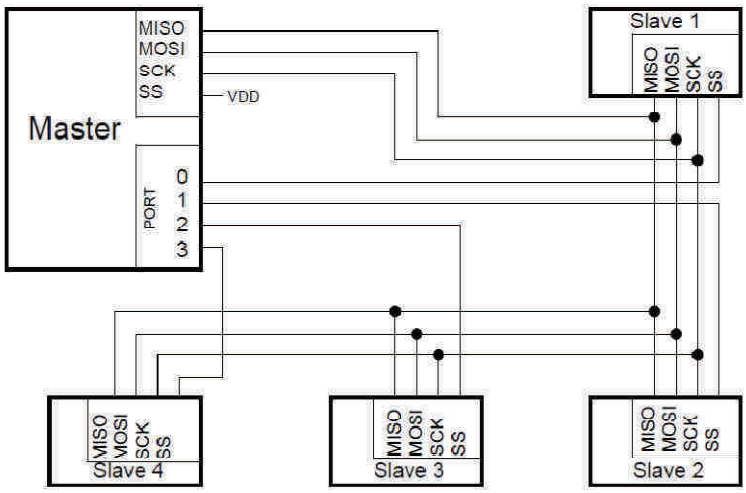
Posledním obvodem připojeným k I2C sběrnici je paměť EEPROM je AT24C128 o velikosti 128kB, která je schopná komunikovat rychlostí 400kHz. Adresu EEPROM lze měnit pomocí vstupů A1 a A0, přičemž pevná část adresy je 10100. V tomto zapojení je adresa 1010000. [30]

1	0	1	0	0	A1	A0	RW

Obrázek 42. Adresa EEPROM paměti AT24C128 [30]

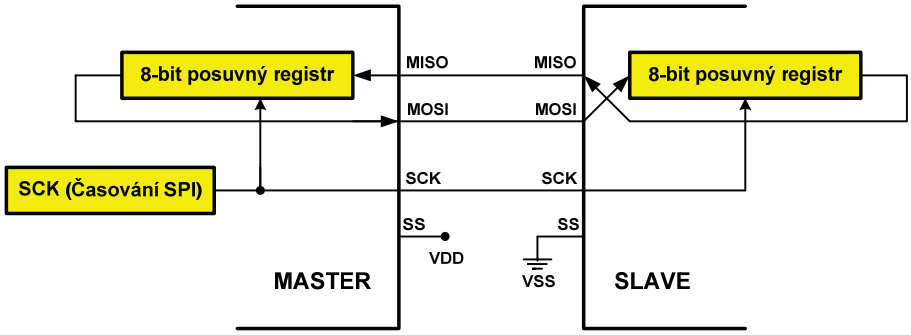
3.6 SPI FLASH paměť

SPI je sériové rozhraní pro vysokorychlostní synchronní přenos dat mezi mikrokontrolérem a externím zařízením. Propojení master - slave se dá realizovat podle níže uvedeného obrázku. Pro komunikaci s určitým zařízením nastaví master chip select (SS nebo CS) žádaného slave zařízení na logickou 0, ostatní CS slave zařízení jsou dále v logické úrovni 1. [31]



Obrázek 43. Připojení více SPI zařízení k jednomu master zařízení [31]

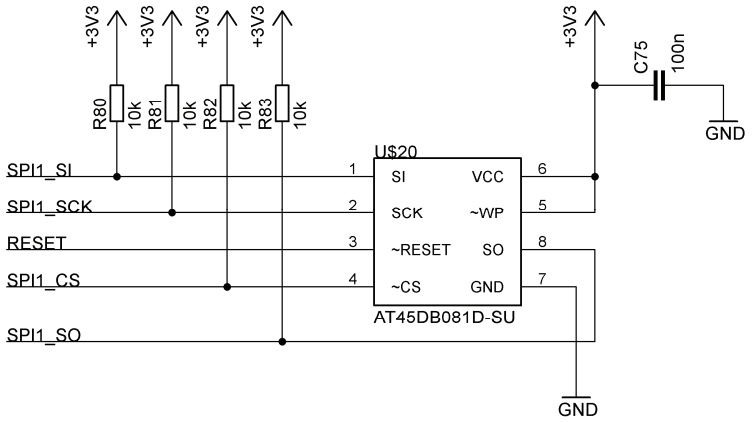
SPI komunikace je založena na posuvných registrech, při správném nastavení stačí zapsat data do příslušného datového registru a odeslání se provede automaticky hardwarově.



Obrázek 44. Průběh SPI komunikace [31]

Po zapsání dat do datového registru se aktivuje časování SCK a data se přes výstup MOSI začínají odesílat, během odesílání je přes vstup MISO přijímán obsah datového registru slave zařízení. Přenos končí, jakmile jsou odeslána všechna data. [31]

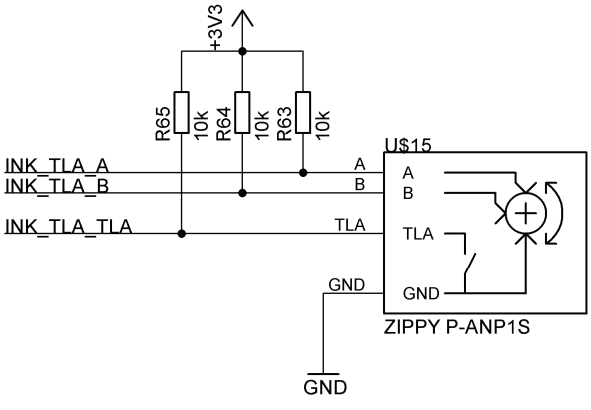
Pro vývojový kit byla vybrána 8MB SPI FLASH paměť AT45DB081D, která je schopna pracovat na frekvenci až 66MHz. Paměť je připojena na SPI1 mikrokontroléru STM32F107VCT6.



Obrázek 45. SPI FLASH paměť AT45DB081D [33]

3.7 Inkrementální snímač

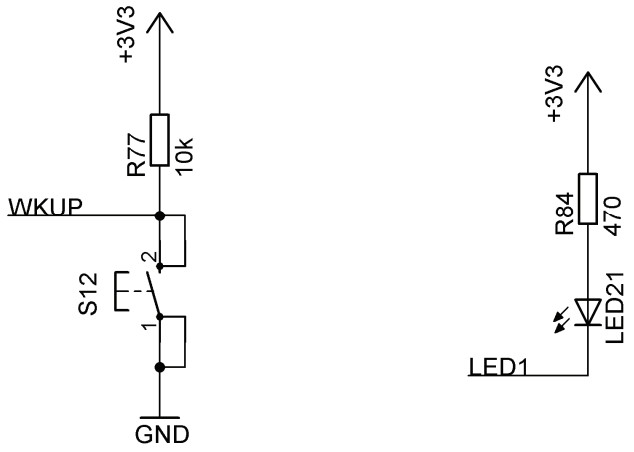
Mezi základní ovládací prvky každého moderního zařízení patří tlačítka popřípadě obousměrné inkrementální snímače otáček. V tomto zapojení byl použit obousměrný inkrementální snímač, jehož výstupy byly přivedeny na vstupy časovačů mikrokontroléru (v tomto zapojení bude možné měřit přesně směr a rychlost otáčení inkrementálního snímače). Součástí inkrementálního snímače je i jedno tlačítko.



Obrázek 46. Inkrementální snímač s tlačítkem [34]

3.8 Tlačítka a LED diody

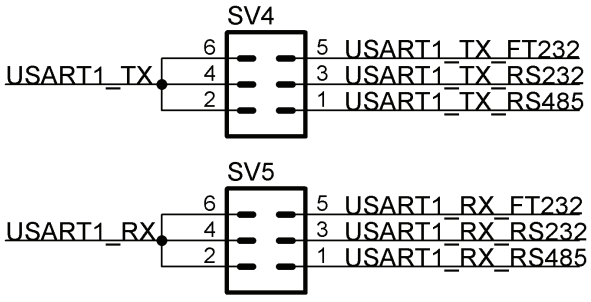
K mikrokontroléru byla připojena tři tlačítka, dvě jsou připojena k pinům PA11 a PA12 a třetí je připojeno k pinu PA0, který zároveň slouží k probouzení mikrokontroléru (WKUP). Dále byly přidány i dvě LED diody připojené k pinům PB6 a PB7. Zapojení tlačítek a LED diod je na následujícím obrázku:



Obrázek 47. Tlačítka a LED připojené přímo k mikrokontroléru

3.9 Synchronní/asynchronní linka USART1

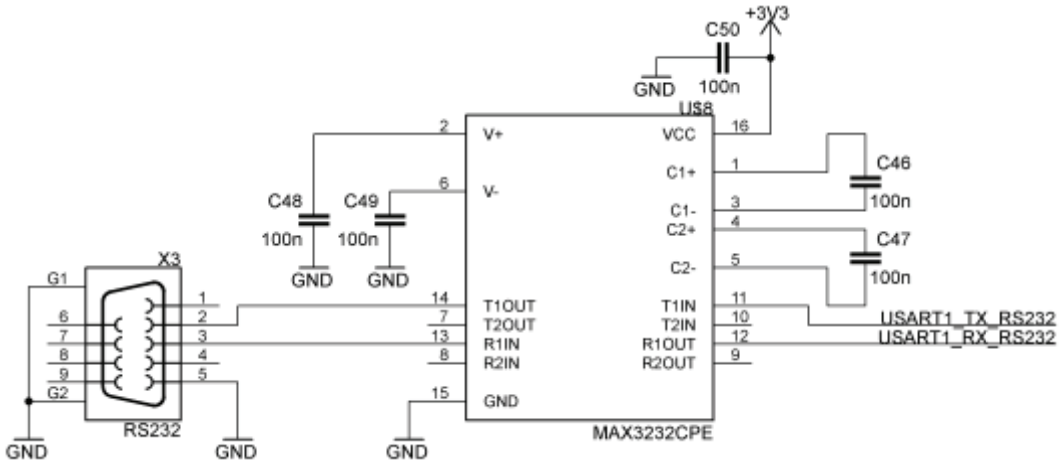
Jedná se o sériovou synchronní/asynchronní linku, pomocí které lze s mikrokontrolérem komunikovat nebo do něj za pomoci integrovaného bootloaderu nahrát program. K USART1 je možno se připojit přes USB, RS232 nebo RS485 (přepínání je řešeno pomocí jumperů).



Obrázek 48. Přepínání vstupů USARTu1

3.9.1 Převodník RS232 na UART

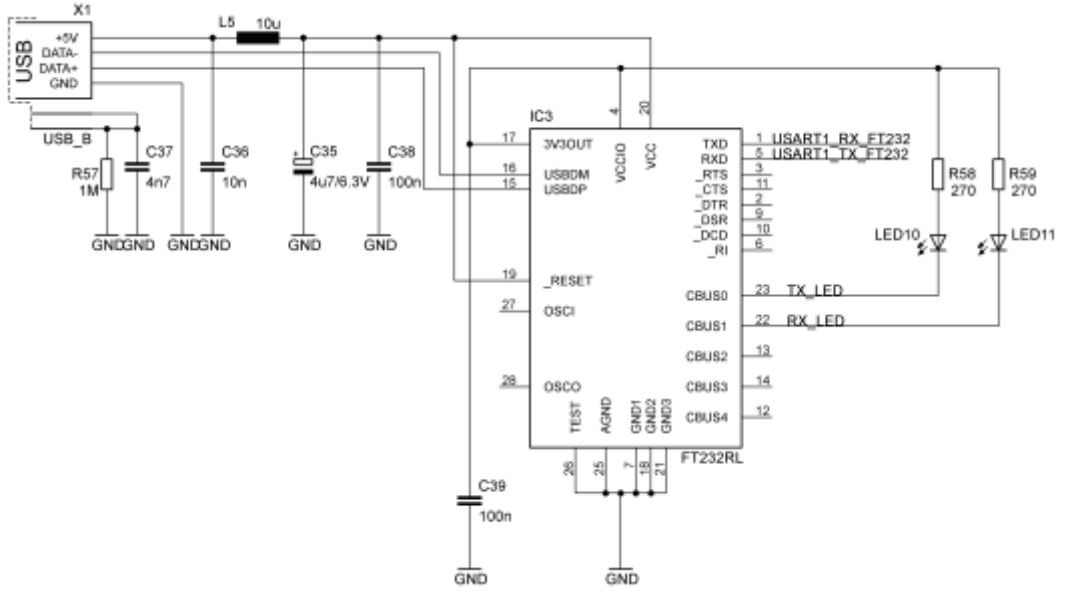
Jelikož standart RS232 komunikuje v jiných napěťových úrovních než UART, tak je potřeba provést konverzi napětí. Pro převod je použit oblíbený integrovaný obvod firmy MAXIM MAX3232CPE.



Obrázek 49. Převodník RS232 na UART [35]

3.9.2 Převodník USB na UART

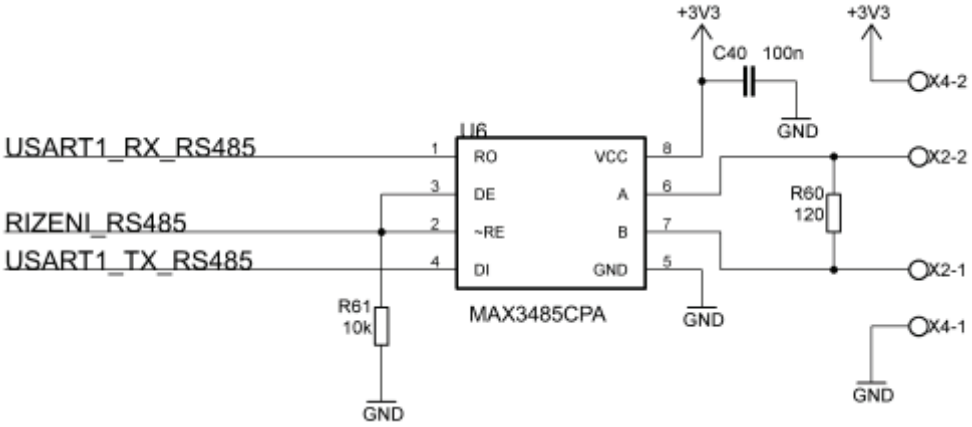
Komunikace po USB je velmi komplikovaná a proto firma FTDI vyvinula převodník FT232RL, který USB komunikaci převádí na UART. S tímto převodníkem se dá pracovat jako s virtuálním COM portem nebo pomocí driverů napsaných s využitím knihovny D2XX se chová jako USB zařízení.



Obrázek 50. Převodník USB na UART [36]

3.9.3 Převodník RS485 na UART

Pro komunikaci na velké vzdálenosti se používá RS485 (kroucený pár). Délka vedení je až 1200m. Tento standard je oblíbený především v průmyslové automatizaci, protože je odolný proti rušení, jedinou nevýhodou tohoto standardu je, že se nedosahuje příliš velkých rychlostí.

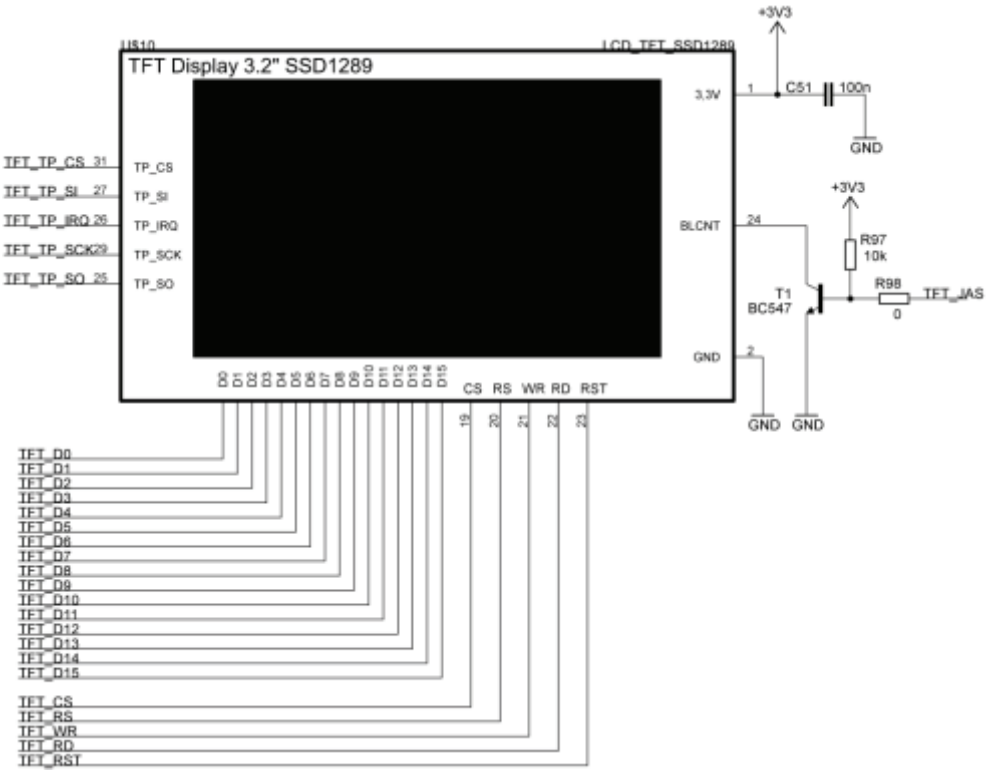


Obrázek 51. Převodník RS485 na UART [37]

V zapojení byl použit převodník firmy MAXIM MAX3485CPA, který je k mikrokontroléru připojen pomocí standardních signálů UARTU (RX,TX) a navíc obsahuje signál pro řízení směru komunikace.

3.10 Dotykový TFT displej 3,2“ s řadičem SSD1289

V dnešní moderní době si uživatelé zvykli, že každé modernější zařízení se dá ovládat pomocí dotykových obrazovek, proto i tento vývojový kit bude vybaven dotykovým displejem s rozlišením 320x240 pixelů.

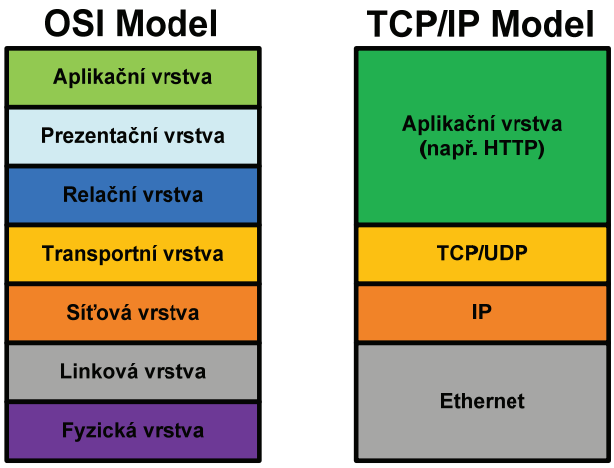


Obrázek 52. TFT dotykový displej 3,2" s řadičem SSD1289 [38][39]

Tento displej má šestnáctibitovou datovou sběrnici (připojeno na port PE mikrokontroléru) a pět řídících signálů (připojeno na piny portu PD). Ovládání jasu podsvícení TFT displeje je možno realizovat pomocí PWM (pin TFT_JAS). Komunikace s dotykovou folií je řešena pomocí SPI (při doteku je vyvoláno přerušení – aktivuje se výstup TFT_TP_IRQ). [38][39]

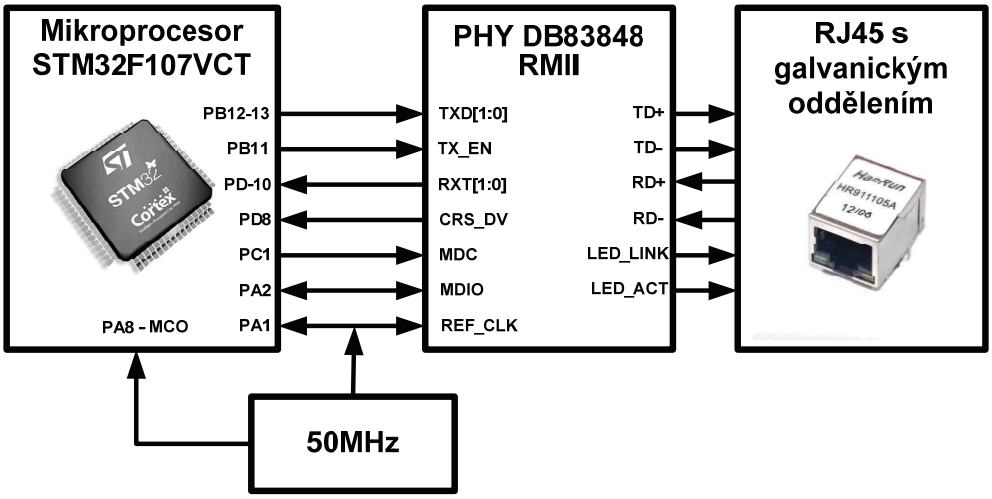
3.11 Ethernet

Poslední částí vývojového kitu je ethernet, který se dostává do všech oblastí techniky a je nedílnou součástí všech komunikačních zařízení distribuujících informace. Komunikaci v síti popisuje TCP/IP model, vycházející z ISO/OSI modelu.



Obrázek 53. TCP/IP model vs. ISO/OSI model

V první části bude popsána hardwarová část, která je tvořena fyzickou a linkovou vrstvou (ethernet). Principielně lze HW část vyjádřit následujícím způsobem:



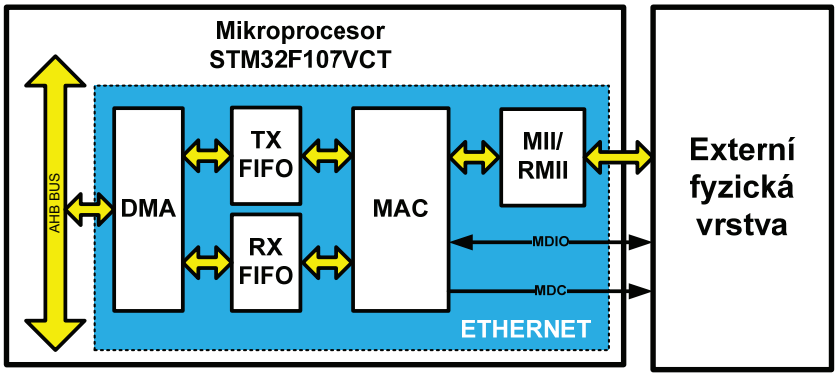
Obrázek 54. Principielní zapojení ethernetu u navrženého vývojového kitu

Jelikož v mikrokontroléru je integrována pouze MAC vrstva, tak se musí Phy (fyzická vrstva) přidat externě. Co se týká komunikace s fyzickou vrstvou, máme dvě možnosti. První variantou je MII komunikace, tento typ komunikace zabírá příliš mnoho pinů mikrokontroléru, výhodou je, že fyzickou vrstvu pak stačí časovat krystalem 25MHz. Jelikož je snaha ušetřit co nejvíce pinů mikrokontroléru, tak byla

zvolena komunikace pomocí RMII(data jsou odesílána nadvakrát po polovičním počtu vodičů, fyzická vrstva je taktována krystalem 50MHz).

Na základě výše uvedených vlastností byla vybrána fyzická vrstva DB83848, která podporuje RMII komunikaci. Tato fyzická vrstva byla ještě doplněna konektorem RJ45 s integrovanými oddělovacími trafičky.

Vlastní ethernetové rozhraní je z velké části součástí mikrokontroléru a obsahuje: MAC obvod, přijímací a vysílací buffer (velikost bufferů je 2kB) a DMA.



Obrázek 55. Zapojení linkové a fyzické vrstvy

Pro přístup k registrům fyzické vrstvy používá mikrokontrolér sériové rozhraní SMI (serial management interface), MDC je hodinový signál pro MDIO (datový vodič). SMI umožňuje přístup k 32 registrům fyzické vrstvy. Komunikační rámec je následující:

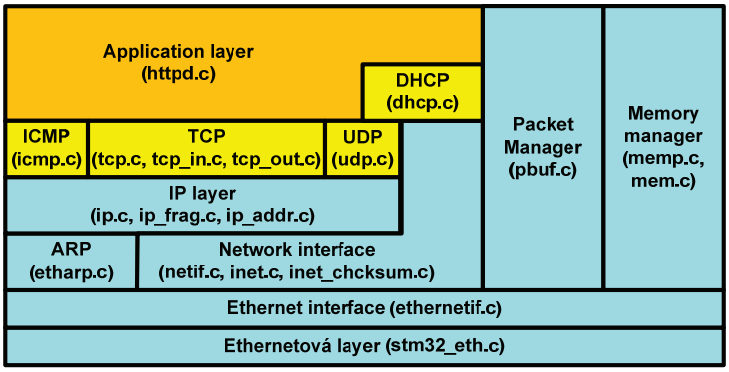
	Preambule	Start	Operace	PADDR	RADDR	TA	Data	Klidový stav
Čtení	1..1	01	10	PPPPP	RRRRR	Z0	D0-D15	Z
Zápis	1..1	01	01	PPPPP	RRRRR	10	D0-D15	Z

Tabulka 2. Komunikační rámec SMI

Nejprve je odesílána preambule, která slouží k synchronizaci (32bitů), následně je odeslán start (01), typ operace (10 – čtení, 01 - zápis), adresa fyzické vrstvy (5bitů), adresa registru (5bitů), TA – oddělovač mezi daty a adresační sekvencí, příjem nebo odeslání požadovaných dat a nakonec se datová sběrnice uvede do klidového stavu.

Pro komunikaci po ethernetu byla výrobcem mikrokontroléru vytvořena knihovna stm32_eth.h (komunikace s MAC vrstvou), knihovnu používá knihovna stm32f107.h, ve které je provedena inicializace ethernetové komunikace. Tímto je nastavena fyzická a linková vrstva (ethernet).

Nad ethernetovou vrstvou je postaven TCP/IP stack, který provádí zpracování přijatých dat po ethernetu. Rozhraní zajišťující transfer dat mezi ethernetem a stackem je v knihovně ethernetif.h. Strukturu celého TCP/IP stacku LwIP je pak možno znázornit následujícím způsobem:



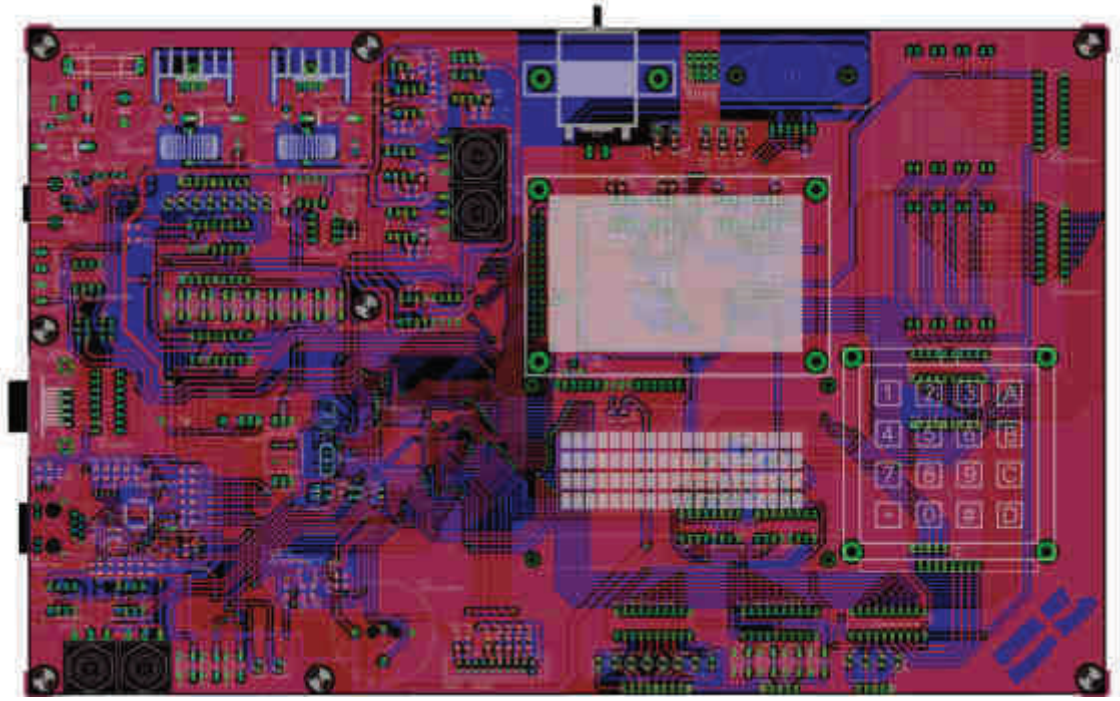
Obrázek 56. Struktura TCP/IP stacku LwIP

Povinou součástí TCP/IP stacku LWIP jsou soubory podbarvené na obrázku 56 modře, tyto soubory zajišťují základní funkčnost. Nad touto částí stacku je vytvořen například TCP protokol, UDP protokol atd… Nad těmito protokoly je pak postavena vlastní aplikace (např. webové stránky).

Pomocí ukázkové aplikace od STM bylo otestováno správné zapojení fyzické vrstvy ethernetu.

3.12 Výsledný vzhled navržené DPS

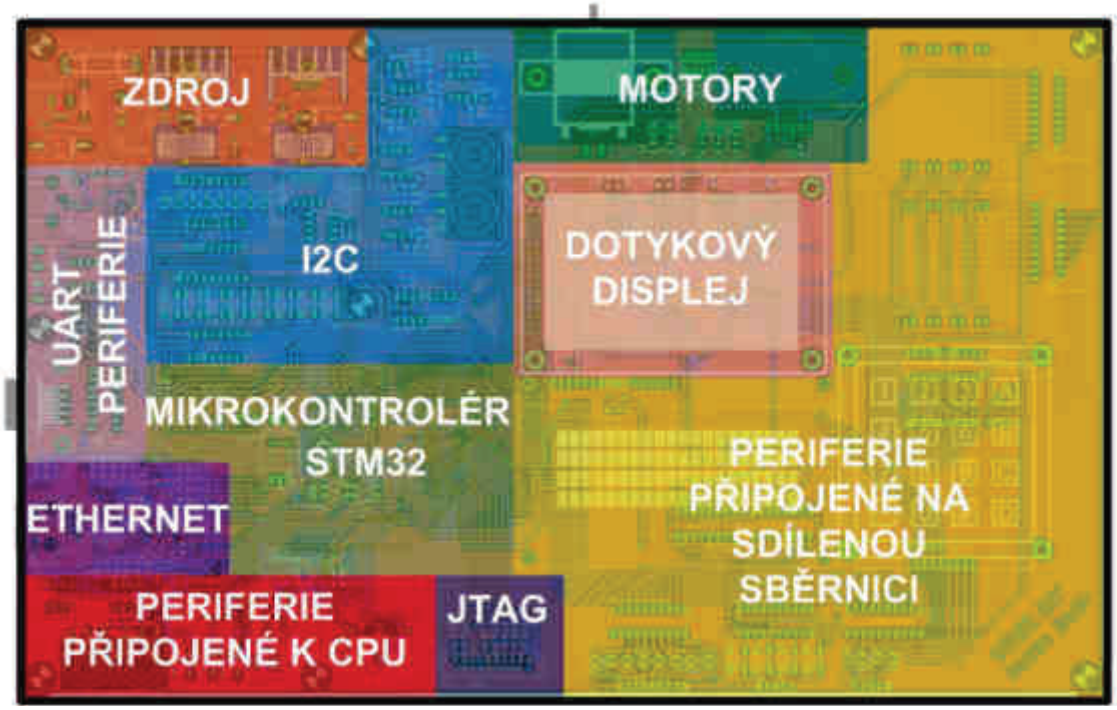
Na základě výše uvedených schémat zapojení byla navržena deska plošného spoje. Jelikož se jedná již o poměrně rozsáhlé zapojení, tak návrh na jednovrstvou DPS byl zamítnut. Výsledná DPS byla navržena na dvoustrannou DPS. Rozměr navržené DPS je 21cm x 35cm.



Obrázek 57. Navržená dvoustranná DPS

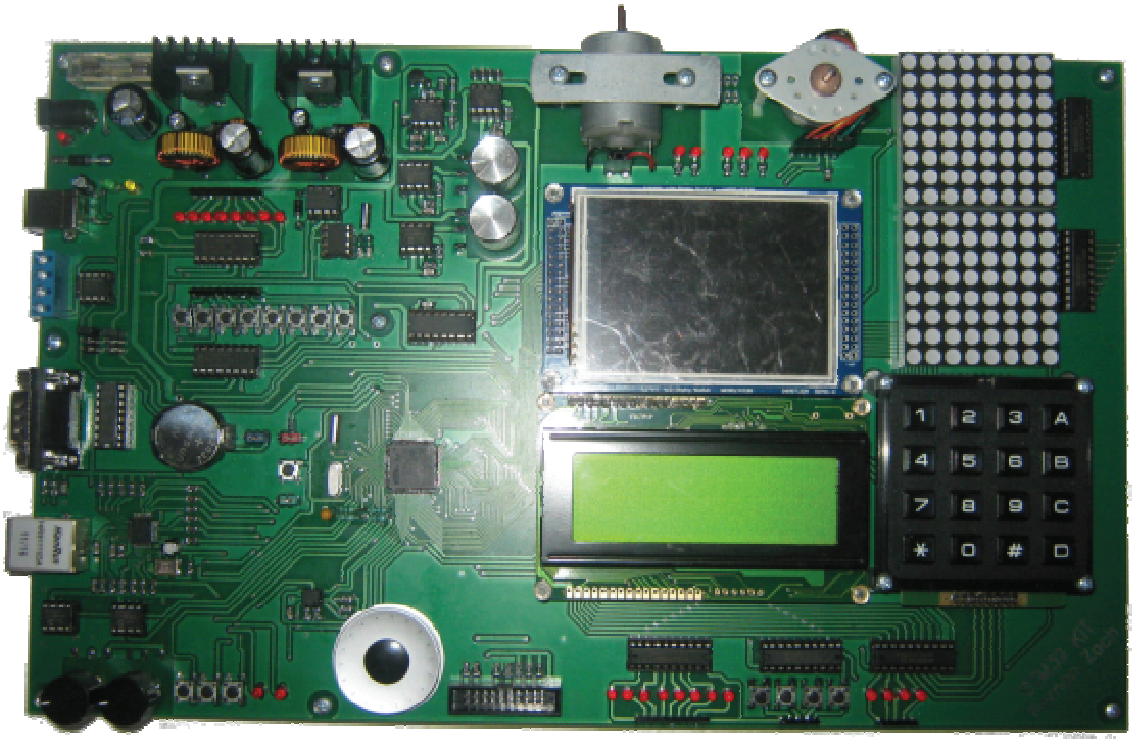
Celá DPS je rozčleněna do několika oblastí. Ve zdrojové oblasti jsou dva DC/DC měniče, které ze vstupního napětí 12V vytváří 3.3V a 5V. V bloku UART periferie jsou převodníky USB na UART, RS232 na UART a RS485 na UART. V oblasti I2C jsou zařízení připojena na I2C sběrnici, jedná se o AD převodníky, DA převodníky, expandéry, hodiny reálného času atd.. V oblasti motory je jeden DC motor a jeden krokový motor. Největší oblast tvoří periferie připojené na sdílenou sběrnici (piny PD0-PD7 mikroprocesoru). V této oblasti jsou displeje, maticová klávesnice, led diody, tlačítka a je k ní připojena i oblast nazvaná motory. Dále samozřejmě nesmí chybět ethernet, vyvedení JTAG konektoru a periferie připojené přímo k mikrokontroléru (2x LED, 3x tlačítko, 2x AD převodník a obousměrný inkrementální snímač).

Poslední nejdůležitější částí je mikrokontrolérová část, ve které je mikrokontrolér STM32F107VCT6 se součástkami potřebnými pro chod mikrokontroléru (resetovací obvod, krystaly, externí flash, zálohovací baterie interních hodin reálného času).



Obrázek 58. Rozvržení periferií na navržené DPS

Celá tato DPS byla vyrobena, následně osazena a oživena. Výsledný vzhled osazené DPS je následující:



Obrázek 59. Výsledný vzhled osazené DPS

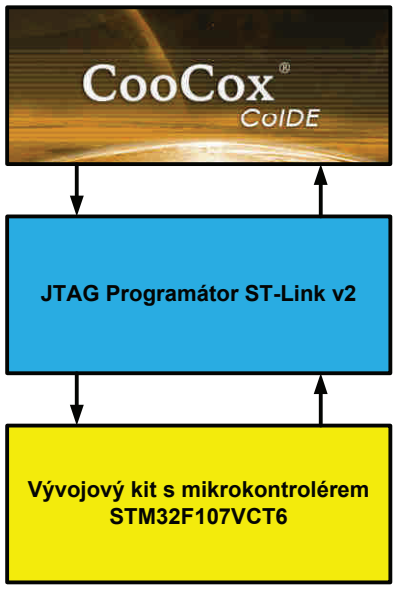
4 SOFTWARE

V této části práce bude ukázáno zprovoznění komunikace mezi vývojovým prostředím a mikrokontrolérem. V další části pak budou uvedeny vzorové úlohy se stručným komentářem. Podrobnější informace k jednotlivým úlohám budou dostupné na přiloženém DVD.

4.1 Propojení vývojového prostředí s mikrokontrolérem

Jako vývojové prostředí bude použito CoIDE projektu CooCox. Toto vývojové prostředí je pod licencí GNU. Vybral jsem jej, protože obsahuje již ovladače pro většinu komerčních i nekomerčních programátorů a jeho zprovoznění je velmi snadné.

Struktura komunikace mezi jednotlivými částmi je na následujícím diagramu. Vývojové prostředí komunikuje s vývojovým kitem přes ST-Link v2 (je možno použít i jiný, poslední verze CoIDE podporuje asi 20 programátorů).

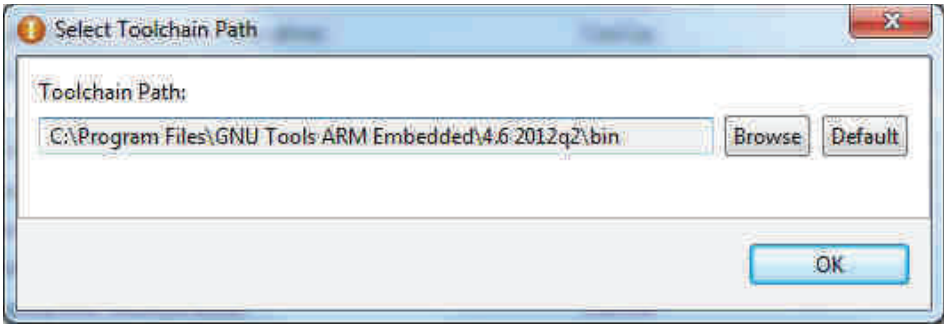


Obrázek 60. Propojení vývojového prostředí a mikrokontroléru

V první fázi je nejprve potřeba nainstalovat překladač GCC ARM, jelikož není součástí vývojového prostředí. Následně se nainstalují ovladače ST-Linku v2 a nakonec nainstalujeme vývojové prostředí CoIDE. Pro správnou funkci je pak potřeba provést propojení jednotlivých částí s vývojovým prostředím dle níže uvedeného postupu.

4.1.1 Propojení překladače a vývojového prostředí

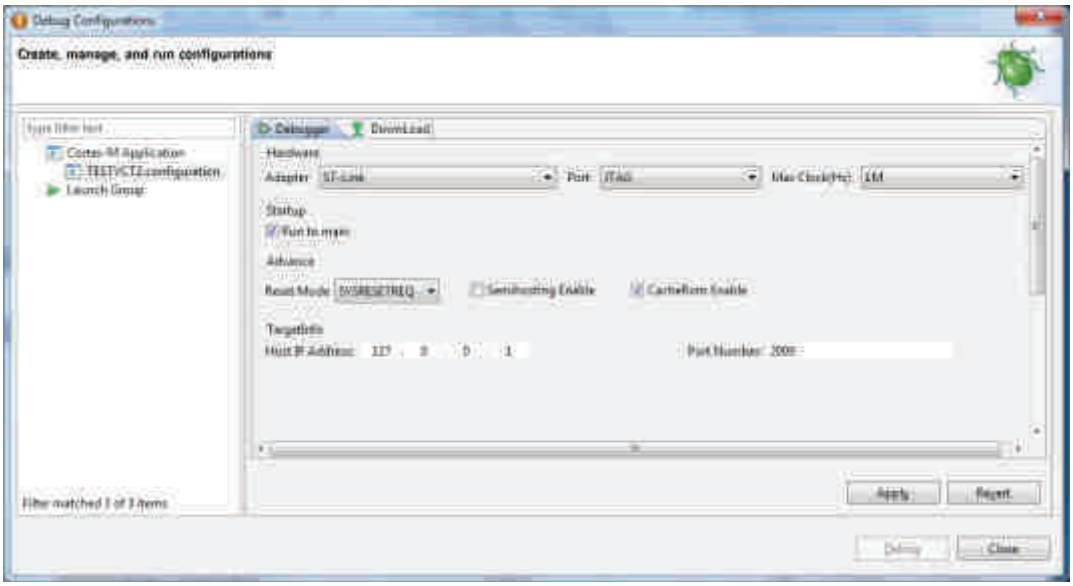
Po instalaci překladače je potřeba provést jeho propojení s vývojovým prostředím, to se provede následujícím způsobem. Po spuštění vývojového prostředí klikneme na Projekt => Select Toolchain Path a nastavíme cestu ke GCC překladači například následovně:



Obrázek 61. Propojení GCC překladače a CoIDE

4.1.2 Propojení ST-Linku v2 a vývojového prostředí CoIDE

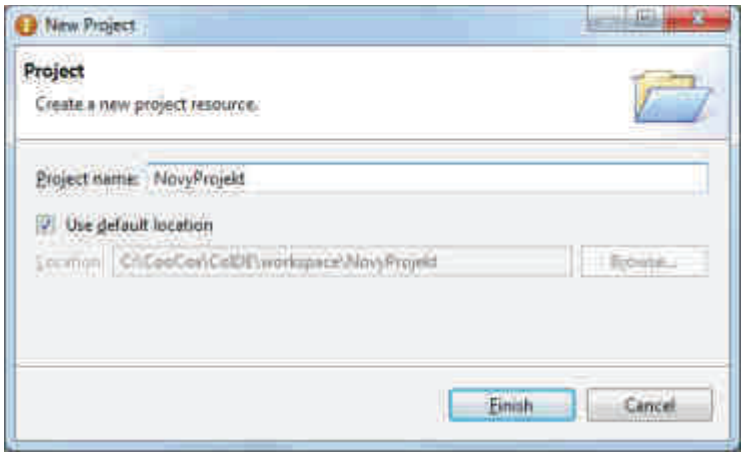
Před propojením je potřeba nainstalovat ovladače ST-Linku. Pokud je již máme nainstalovány, tak stačí přes Debug => Debug configuration ve vývojovém prostředí CoIDE provést následující nastavení:



Obrázek 62. Propojení ST-Linku a CoIDE

4.2 Založení projektu

Po spuštění vývojového prostředí se nový projekt založí stiskem záložky Projekt => New projekt. Zde zadáme jméno projektu a adresář kam se mají ukládat zdrojové soubory (v tomto případě je ponecháno výchozí nastavení).



Obrázek 63. Založení nového projektu v CoIDE

Nyní se nám vytvořil nový projekt a otevřela se nám záložka „Repository“, ve které vybereme výrobce „ST“, mikrokontrolér „STM32F107VC“ a nakonec se nám objeví tabulka „Select Components“, ve které si vybereme periferie, které budeme používat, knihovny k těmto periferiím se nám automaticky přidají k námi vytvořenému projektu. Pokud bychom měli nějaké vlastní knihovny, tak je do projektu můžeme přiřadit pomocí „Project=>Configuration“ a v záložce „Include paths“ pomocí add přidáme cestu ke složce s knihovnami (bez tohoto přiřazení nelze s knihovnami pracovat).

COMMON	
C Library	Implement the minimal functionality required to allow newlib to link
Retarget printf	Implementation of printf(), sprintf() to reduce memory footprint
Semihosting	Implementation of Semihosting GetChar/SendChar
CMSIS core	CMSIS core for Cortex M3 V1.30
BOOT	
SH 70xx	READ-OUT
CMSIS Boot	STM32F10x CMSIS Boot Driver
PERIPHERAL ST	
RCC	STM32F10x Reset and clock control driver
CRC	STM32F10x Cyclic redundancy check driver
PWR	STM32F10x Power control driver
BKP	STM32F10x Backup registers driver
GPIO	STM32F10x General-purpose I/O ports driver

Obrázek 64. Výběr knihoven

Pro vyzkoušení práce s vývojovým prostředím si vyzkoušíme vytvořit program, který bude blikat diodami. K projektu je nutno přidat následující knihovny: CMSIS core, CMSIS Boot (tyto dvě knihovny obsahují funkce pro práci s jádrem ARM), RCC (knihovna pro povolování jednotlivých periferií) a GPIO(knihovna pro práci s porty). Výsledný program může vypadat například následovně:

```
#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"

int main(void)
{
    int i;
    //Vytvoření inicializační struktury
        GPIO_InitTypeDef  GPIO_InitStructure;
    //Povolení hodin portu B
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    //Nastavení parametrů pinů PB6 a PB7, na kterých jsou připojeny LED diody
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOB, &GPIO_InitStructure);

    while(1)
    {
        //Negování stavu PB6
            GPIOB->ODR ^= GPIO_Pin_6;
        //Zpoždění
            for(i=0;i<0x100000;i++);
        //Negování stavu PB7
            GPIOB->ODR ^= GPIO_Pin_7;
        //Zpoždění
            for(i=0;i<0x100000;i++);
    }
}
```

Obrázek 65. Program pro vyzkoušení práce s vývojovým prostředím

Nyní, když máme napsaný vlastní program, tak jej musíme přeložit Projekt => Rebuild. Následně se nám do console napíše průběh překladu, pokud vše proběhlo v pořádku, tak můžeme program nahrát do mikrokontroléru Flash => Program download, po nahrání se nám na kitu rozblikají dvě led diody. Jelikož máme možnost ladění programu přímo na čipu, tak se můžeme přepnout do debug modu Debug => Debug a krokovat vlastní program jako v jiných vývojových prostředích.

Nyní jsme si vytvořili první aplikaci, která ověřila funkčnost vývojového prostředí a správného nastavení komunikace s mikrokontrolérem.

4.3 Úloha 1: Práce s porty a časovači

Cílem úlohy je vyzkoušet si práci s I/O porty a časovači mikrokontroléru STM32F107VCT6. V programu budou požívány různé režimy GPIO, časovače budou použity pro vytvoření jednoduchého čítače s využitím přerušení.

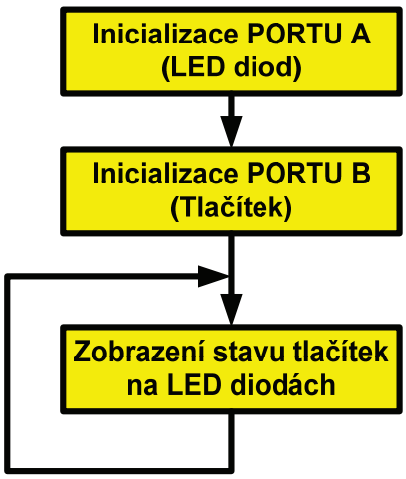
4.3.1 Zadání

1) Vytvořte program, který bude pomocí tlačítek ovládat LED diody na vývojovém kitu. V prvním kroku zkuste ovládat LED diody připojené přímo k portu mikrokontroléru. V druhém kroku proveďte to samé s LED diodami a tlačítky připojenými k mikrokontroléru přes záchytné registry (jedná se o periferie, které sdílí piny mikrokontroléru s jinými periferiemi).

2) Nyní, když umíme ovládat základní periferie, tak by bylo dobré LED diody rozblikat. Pro přesné měření času se používají časovače. Proto si vytvořte jednoduchý čítač, který bude čítat počty přerušení od časovače. Doporučuji nastavit časovač, aby u časovače nastalo přerušení každou 1 sekundu. Počet přerušení následně zobrazujte na LED diodách

4.3.2 Vypracování:

V prvním úkolu máme ovládat led diody (LED21-PB6, LED22-PB7) pomocí tlačítek (S13-PA11 ,S14-PA12). První program bude zobrazovat stav tlačítek na led diodách. Vývojový diagram programu může být například následující:

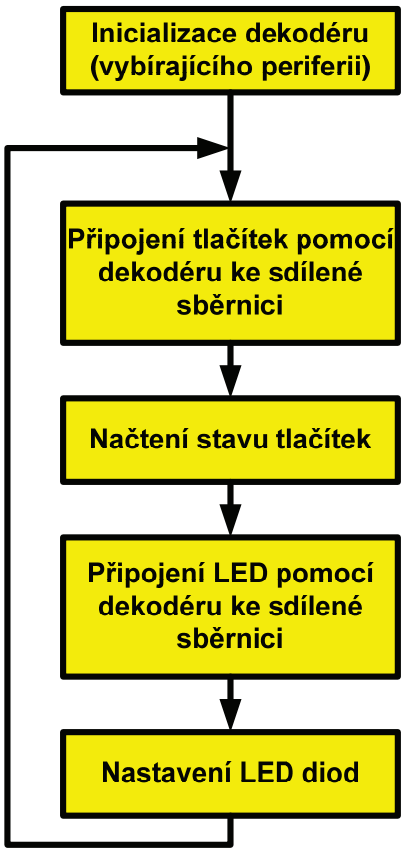


Obrázek 66. Vývojový diagram programu, který zobrazuje stav tlačítek na LED diodách

V programu je nutné pomocí inicializační struktury GPIO_InitTypeDef provést inicializaci LED diod a tlačítek. LED diody nastavíme do výstupního režimu

s otevřeným kolektorem (OUT_OD) a tlačítka jako plovoucí vstup (IN_FLOATING). Pro čtení stavu tlačítek můžeme využít funkci GPIO_ReadInputDataBit a pro zápis na LED diody funkci GPIO_WriteBit.

Když jsme si vyzkoušeli ovládání I/O portů mikrokontroléru, tak nyní vytvoříme program, který bude provádět to stejné s tím rozdílem, že budeme zobrazovat stav tlačítek na led diodách připojených ke sdílené sběrnici D0-D7. Výběr jednotlivých periferii je prováděn pomocí dekodéru 1 z N (viz. schéma zapojení). Tlačítka i LED diody jsou připojeny ke sdílené sběrnici pomocí záchytných registrů. Výsledný program může vypadat například následujícím způsobem:



Obrázek 67. Vývojový diagram programu, který zobrazuje stav tlačítek na LED diodách, připojených ke sdílené sběrnici PD0-PD7

V programu jsou využity stejné postupy jako v předchozím případě, proto zde zmíním jen funkce, které byly vytvořeny pro výběr jednotlivých periferií připojených ke společné sběrnici.

První funkce slouží k inicializaci dekodéru, který vybírá periferie připojené ke sdílené sběrnici PD0 – PD7.

```
void init_dekoder(void);
```

Jedná se o dekodér 1 z 16. Proto jsou k jeho ovládání potřeba 4 vstupy (PC0,PC6,PC7 a PC8), které jsou v této funkci inicializovány.

Druhá funkce slouží k výběru požadované periferie a nastavení režimu této periferie. K nastavení režimu tato funkce používá funkci init_bus, která sdílenou sběrnici nastavuje do vstupního nebo výstupního režimu (k tomu slouží parametr GPIO_Mode).

```
void set_dekoder(char periferie, GPIOMode_TypeDef GPIO_Mode);
```

Jednotlivé periferie jsou dekodérem vybírány podle následující tabulky, ve které číslo periferie udává hodnotu na vstupu dekodéru.

Periferie	Číslo periferie
LCD_HD44780	0
LED	1
TLAČITKA	2
MATICOVA_KL	3
DC_MOTOR	4
KROKOVY_MOTOR	5
MATICOVY_DISP_1	8
MATICOVY_DISP_2	9
MATICOVY_DISP_3	10

Tabulka 3. Dekódovací tabulka pro výběr periferií připojených ke sběrnici D0-D7

Celý program je pak tvořen pouze cyklickým voláním výše uvedených funkcí podle diagramu uvedeného na obrázku 67.

Posledním úkolem je zprovoznit časovač a pomocí něho vytvořit jednoduchý čítač, jehož stav bude zobrazován na LED diodách. Program je znázorněn na obrázku 68.

Nejprve si vybereme LED diody připojené ke sdílené sběrnici pomocí dekodéru, obdobně jako v předchozí úloze. Stav led diod bude ukládán do pomocné proměnné, která reprezentuje hodnotu čítače. Následně zbývá nastavit časovač. V tomto případě je vhodné povolit u časovače přerušení, nežli kontrolovat stav flagu udávajícího přetečení časovače. Inicializace přerušení od časovače TIM2 je provedena ve funkci init_irq_tim2.

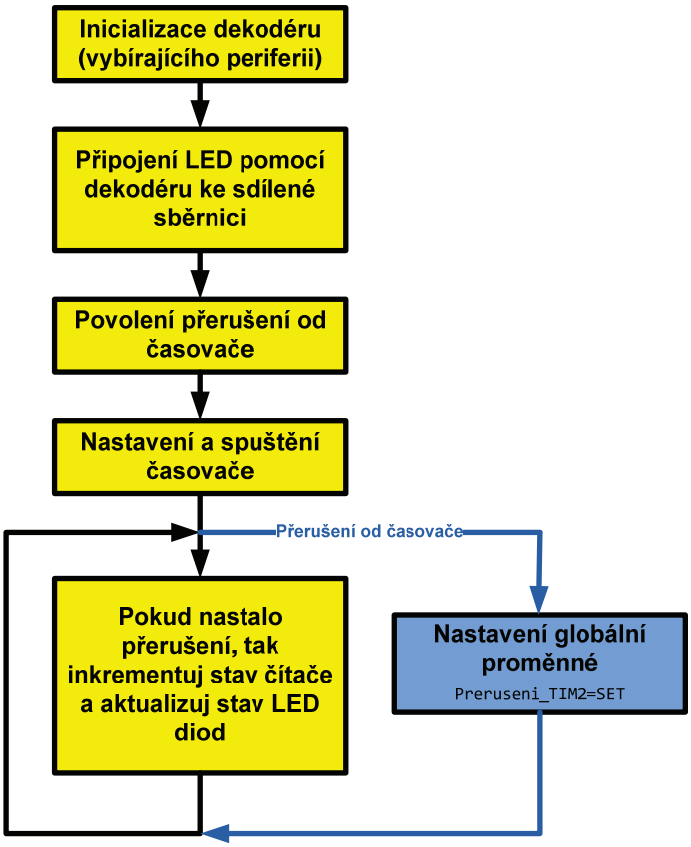
```
void init_irq_tim2(void);
```

K inicializaci přerušení se používá inicializační struktura NVIC_InitTypeDef, ve které se nastaví kanál přerušení, priorita a povolení přerušení. Inicializace přerušení od časovače TIM2 se pak provede zavoláním funkce NVIC_Init.

Pro přerušení od časovače TIM2 je potřeba následně vytvořit přerušovací rutinu, která provede požadovanou operaci.

```
void TIM2_IRQHandler(void);
```

V tomto případě pouze nastaví globální proměnnou Preruseni_TIM2 a vymaže příznak indikující přerušení od TIM2.



Obrázek 68. Vývojový diagram programu čítače

Když máme povoleno přerušení a vytvořenu přerušovací rutinu, tak zbývá nastavit vlastní časovač a spustit jej. Nastavení časovače provádí funkce init_tim2.

```
void init_tim2(int Cas_ms);
```

Ve funkci se nastaví čítání směrem nahoru a čas jak často má dojít k přerušení. Nastavení času přerušení je dáno následujícím vztahem:

$$T = \frac{TIM\_CLK}{(PSC + 1)(ARR + 1)(RCR + 1)} \qquad (3)[12]$$

Tímto máme provedena veškerá nastavení, zbývá vytvořit hlavní smyčku, která bude aktualizovat stav led diod udávajících hodnotu čítače (počtu přerušení od časovače). Hlavní smyčka je velmi jednoduchá, pokud se nastaví proměnná Preruseni_TIM2, tak se aktualizuje stav proměnné udávající hodnotu čítače a zobrazí se její obsah na LED diodách.

Všechny programy k jednotlivým bodům zadání jsou přiloženy na doprovodném DVD včetně podrobnějšího návodu.

4.4 Úloha 2: UART

Pro komunikaci na větší vzdálenosti se u mikrokontrolérů používá UART (universal asynchronous receiver/transmitter) doplněný o převodník na USB, RS232 nebo RS485.

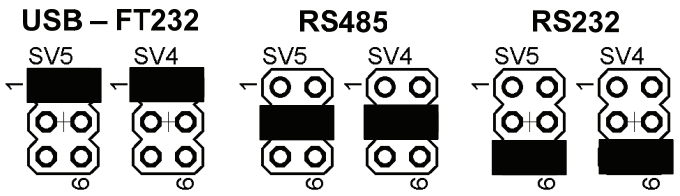
4.4.1 Zadání

1) Vytvořte program, který bude přes UART komunikovat s počítačem. V prvním kroku naprogramujte jednoduchý LOOPBACK (data co přijdou, od PC, posílejte nezměněná zpět)

2) Rozšiřte program o zpracování přijatých dat. Například: při přijetí řetězce <1> se rozsvítí LED1 a LED2 (PB6,PB7) a při přijetí řetězce <2> zhasnou.

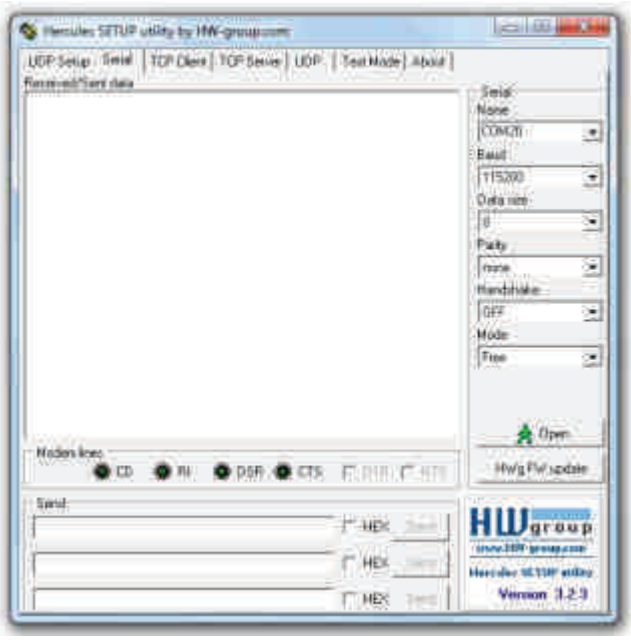
4.4.2 Vypracování

Pro komunikaci po UARTu s mikrokontrolérem můžeme použít tři rozhraní (USB, RS232 nebo RS485). Tato rozhraní jsou připojena pomocí převodníků k USART1 mikrokontroléru. USART lze nastavit pro synchronní nebo asynchronní komunikaci (v tomto případě bude použit asynchronní režim). V prvním kroku je potřeba na vývojovém kitu vybrat rozhraní, po kterém budeme s mikrokontrolérem komunikovat. Výběr rozhraní se provádí pomocí propojek. RX a TX signály od převodníků jsou vyvedeny na piny SV5 a SV6. Pro výběr jednotlivých komunikačních rozhraní musíme propojky nastavit následujícím způsobem:



Obrázek 69. Volba komunikačního rozhraní

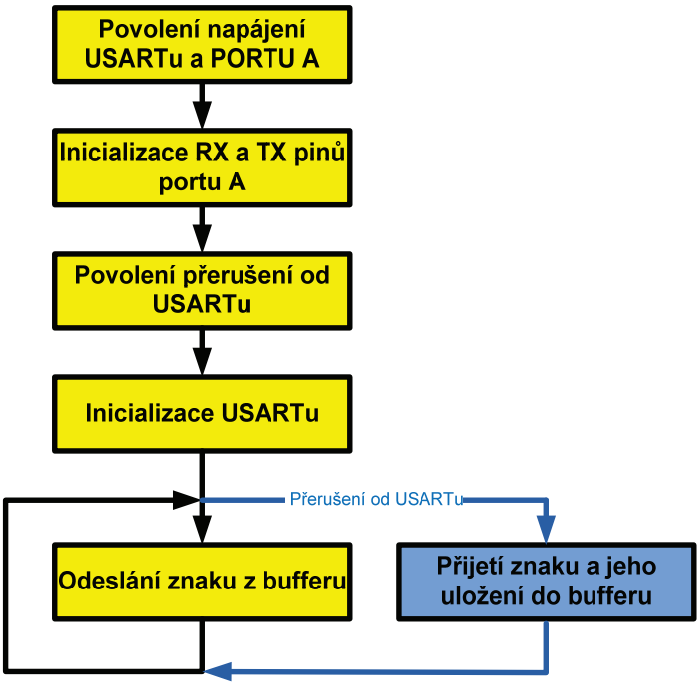
Nyní, když máme vybrané komunikační rozhraní, tak zvolíme vhodný terminál pro ladění komunikace po sériové lince. Doporučuji použít terminál Hercules, je celkem jednoduchý a obsahuje vše potřebné pro ladění komunikace.



Obrázek 70. Terminál Hercules

Pro navázání komunikace musíme nejprve nastavit parametry komunikace (záložka Seriál - Nastavení, které zde zvolíme, musíme dodržet i v mikrokontroléru). Jakmile máme nastavení dokončené, tak otevřeme komunikační kanál pomocí tlačítka Open. Nyní můžeme začít odesílat znaky mikrokontroléru (záložka Send). Pro odeslání ascii znaku stačí napsat znak do řádku a potvrdit tlačítkem Send. Pokud potřebuje odeslat hexadecimální číslo, tak stačí napsat $ a příslušné hexadecimální číslo (např. $FF), pro odeslání dekadického čísla odešleme znak # (např #255).

Když máme vše připravené, tak můžeme začít psát program pro mikrokontrolér. Vývojový diagram vypadá následovně:



Obrázek 71. Vývojový diagram programu, který komunikuje po UARTu (LOOPBACK)

V prvním kroku povolíme hodiny a napájení USARTU1 a PORTU A, na kterém je USART1 vyveden.

V dalším kroku musíme nastavit vstupní a výstupní piny PORTU A, na které jsou přivedeny signály RX (PA10) a TX (PA9). Pin PA10 nastavíme jako vstupní (GPIO_Mode_IN_FLOATING) a pin PA9 jako výstupní (GPIO_Mode_AF_PP).

Dále je potřeba povolit přerušení od USARTU1. To se provádí pomocí funkce NVIC_Init, které předáme strukturu s povolením přerušení příslušného kanálu a nastavením priority přerušení.

Nakonec provedeme nastavení vlastního USARTU1. Rychlost komunikace je nastavena na 115200 b/s, délka slova je 8bit, jeden STOP bit, bez paritního bitu, HW řízení komunikace je vypnuto (na tomto kitu není možné jej používat!). Tyto nastavené hodnoty je potřeba dodržet i při nastavování terminálu, pomocí kterého s mikrokontrolérem budeme komunikovat.

Tímto krokem máme hotovu nejtěžší část programu (inicializaci), můžeme zvolit i jiné nastavení, možností jsou spousty. Nyní již stačí napsat rutinu pro obsluhu přerušení od USARTU1 a následně zpracovat přijatá data.

Vzhledem k tomu, že příjem dat musí být co nejrychlejší, tak není možné data zpracovávat v rutině obsluhující přerušení, proto je potřeba vytvořit dostatečně velký kruhový buffer, do kterého se budou ukládat přijatá data. Tyto data pak budou zpracována mimo přerušovací rutinu.

Přerušovací rutina slouží k přijímání dat, která jsou odeslána z počítače po zvoleném komunikačním rozhraní. V přerušovací rutině hlídáme stavový registr USARTU, ve kterém se bit FLAG_RXNE nastaví v případě, že byl přijat jeden znak po USARTU. Tento znak vyzvedneme pomocí funkce ReceiveData a uložíme jej do bufferu. Za poslední přijatý znak ukládáme symbol \0, na základě kterého následně můžeme detekovat konec přijatého řetězce.

```
void USART1_IRQHandler(void);
```

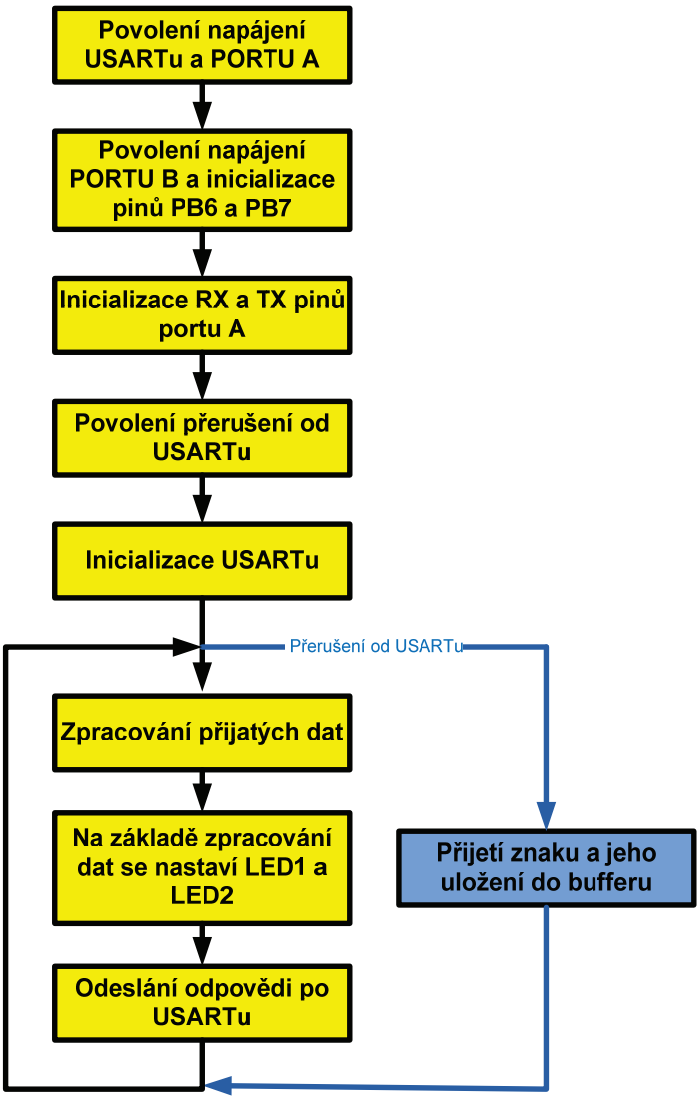
Nyní, když máme hotov příjem dat, tak vytvoříme program, který bude odesílat přijatá data zpět do počítače po daném komunikačním rozhraní (LOOPBACK). V bufferu si budeme hlídat, zda je na aktuální pozici znak \0, jakmile je tento znak přepsán, tak znak odešleme pomocí funkce send_uart a inkrementujeme pozici v bufferu (toto provádíme cyklicky).

Funkce send_uart, která odesílá řetězec je vystavěna na funkci dodávané v knihovně STM (USART_SendData). Funkce send_uart odesílá předávaný řetězec znak po znaku a kontroluje nastavení příznaku odeslání znaku.

```
void send_uart(unsigned char *buffer, unsigned long count);
```

Nyní už stačí komunikaci vyzkoušet pomocí terminálu. Výsledný kód je přiložen na doprovodném DVD včetně podrobnějšího návodu.

Pokud nám komunikace funguje, tak můžeme výsledný kód upravit podle druhého bodu zadání. Nebude to nic složitého. Stačí doplnit inicializaci PORTU B, na který jsou připojeny LED1 a LED2 a provést zpracování přijatých dat. Vývojový diagram programu se upraví následujícím způsobem:



Obrázek 72. Vývojový diagram programu ovládajícího LED diody pomocí USARTu

U této úlohy stačí jenom provést inicializaci PORTU B (provede se stejně jako u PORTU A), upravit zpracování přijatých dat a na základě přijatých dat provést nastavení LED diod. Pro setování a resetování stavu výstupu mikrokontroléru doporučuji použít funkce GPIO_SetBits a GPIO_ResetBits (LED svítí, když je hodnota výstupu rovna 0).

4.5 Úloha 3: Maticová klávesnice

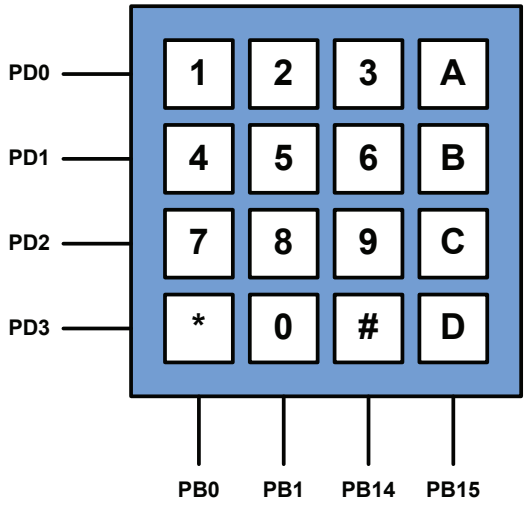
Pokud máme nějaký embedded systém, tak u něj ve většině případů nesmí chybět nějaký vstup. Mezi nejjednodušší vstup patří maticová klávesnice, která je poměrně levná a jednoduchá na obsluhu.

4.5.1 Zadání

1) Vytvořte funkci, která bude číst stav maticové klávesnice. Funkci odlaďte pomocí JTAG rozhraní.

4.5.2 Vypracování

Maticová klávesnice má připojené sloupce přímo na vstupy mikrokontroléru (PB0,PB1,PB14 a PB15), řádky jsou k mikrokontroléru připojeny pomocí sdílené sběrnice na pinech PD0 - PD3. Principielní schéma zapojení maticové klávesnice je na následujícím obrázku:



Obrázek 73. Principielní schéma připojení maticové klávesnice

Pro čtení stavu maticové klávesnice se používá algoritmus s postupující nulou, jehož princip je naznačen v následujících tabulkách:

Krok	PD0	PD1	PD2	PD3	PB0	PB1	PB14	PB15
1.	0	1	1	1	1	1	1	1
2.	1	0	1	1	1	1	1	1
3.	1	1	0	1	1	1	1	1
4.	1	1	1	0	1	1	1	1

Tabulka 4. Algoritmus pro čtení maticové klávesnice - není stisknuta žádná klávesa

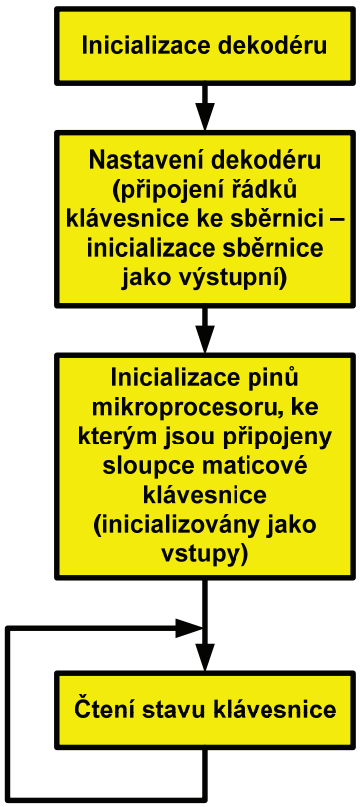
Krok	PD0	PD1	PD2	PD3	PB0	PB1	PB14	PB15
1.	0	1	1	1	1	1	1	0
2.	1	0	1	1	1	1	1	1
3.	1	1	0	1	1	1	1	1
4.	1	1	1	0	1	1	1	1

Tabulka 5. Algoritmus pro čtení maticové klávesnice – stisknuta klávesa A

Krok	PD0	PD1	PD2	PD3	PB0	PB1	PB14	PB15
1.	0	1	1	1	1	1	1	1
2.	1	0	1	1	1	0	1	1
3.	1	1	0	1	1	1	1	1
4.	1	1	1	0	1	1	1	1

Tabulka 6. Algoritmus pro čtení maticové klávesnice – stisknuta klávesa 5

Červeně jsou podbarveny hodnoty přiváděné na řádky maticové klávesnice, modře jsou podbarveny hodnoty čtené na sloupcích maticové klávesnice, když není stisknuta žádná klávesa, a zelenou barvu mají hodnoty čtené na sloupcích maticové klávesnice, když je stisknuta daná klávesa. Celý program je tvořen inicializační sekvencí a voláním funkce, která zajišťuje čtení stavu maticové klávesnice. Program může vypadat například:



Obrázek 74. Vývojový diagram programu, který čte stav maticové klávesnice.

Pro čtení stavu klávesnice byla napsána funkce klávesnice, která vrací hodnotu znaku maticové klávesnice.

```
char klavesnice(void);
```

Správnost čtení lze odladit například pomocí JTAG rozhraní, UARTU, popřípadě zobrazením stisknutého znaku na alfanumerickém displeji atd..

4.6 Úloha 4: Maticový displej

Jedním z používaných typů zobrazovacích zařízení jsou maticové displeje, které využívají pro zobrazování klasické LED diody. Jejich hlavní výhodou je dobrá čitelnost i za špatných světelných podmínek.

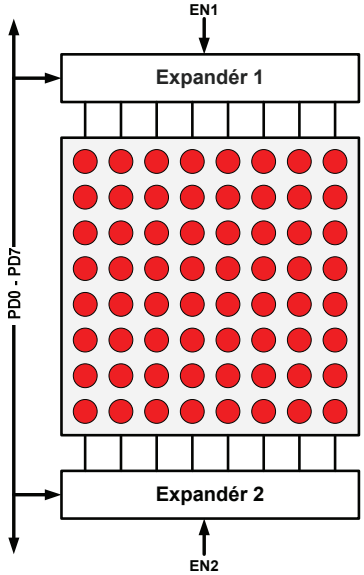
4.6.1 Zadání

1) Napište program, který bude postupně rozsvěcovat jednotlivé led diody na maticovém displeji

2) Napište funkci, která na maticovém displeji zobrazí libovolný obrázek.

4.6.2 Vypracování

Maticový displej je ve svém principu velmi jednoduchý na obsluhu, stačí zapsat na příslušný vstup log 1 a příslušné výstupy přizemnit pro zobrazení jednoho bodu.



Obrázek 75. Principielní zapojení maticového displeje

Bohužel dodavatel mi dodal maticový displej s jiným zapojením, než uváděl v katalogovém listu. Původně se měly jedním expandérem ovládat řádky a druhým sloupce. Nynější zapojení je trochu komplikovanější na obsluhu, jelikož na každém expandéru se ovládají řádky i sloupce.

Pro rozsvícení jednoho bodu na maticovém displeji je potřeba nastavit správně hodnotu expandéru 1 a expandéru 2. Pro zapsání dat ze sběrnice PD0-PD7 na výstup expandéru je potřeba aktivovat signál EN1 nebo EN2, to se provede pomocí dekodéru, kterým je řízeno připojování periferií ke sběrnici. Pro aktivování signálu EN1 je potřeba na vstup dekodéru zapsat 0x8, pro aktivaci EN2 0xA.

Pro zjednodušení obsluhy maticového displeje byly vytvořeny dvě tabulky, ve kterých jsou uloženy hodnoty, které mají být zapsány na výstupy expandérů pro rozsvícení příslušného bodu na maticovém displeji.

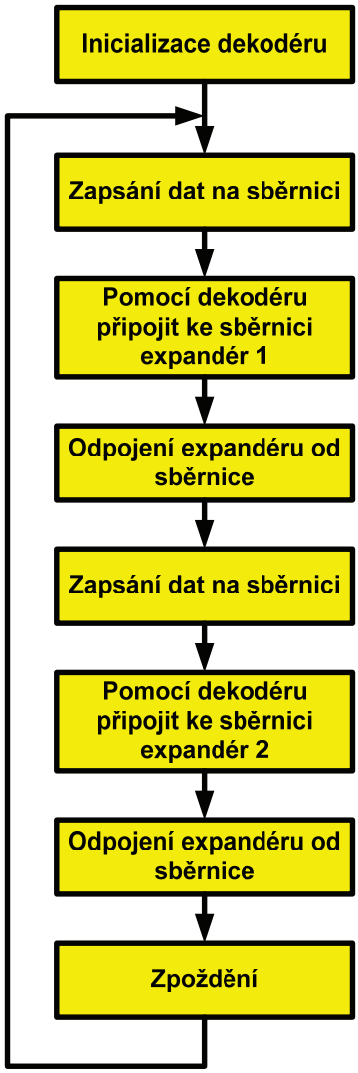
0xAA	0xA8	0xAA	0xA2	0xAA	0x8A	0xAA	0x2A
0xAB	0xA9	0xAB	0xA3	0xAB	0x8B	0xAB	0x2B
0xAA	0xA8	0xAA	0xA2	0xAA	0x8A	0xAA	0x2A
0xAE	0xAC	0xAE	0xA6	0xAE	0x8E	0xAE	0x2E
0xAA	0xA8	0xAA	0xA2	0xAA	0x8A	0xAA	0x2A
0xBA	0xB8	0xBA	0xB2	0xBA	0x9A	0xBA	0x3A
0xAA	0xA8	0xAA	0xA2	0xAA	0x8A	0xAA	0x2A
0xEA	0xE8	0xEA	0xE2	0xEA	0xCA	0xEA	0x6A

Tabulka 7. Nastavení hodnoty expandéru 1 pro rozsvícení určitého bodu maticového displeje

0x56	0x57	0x53	0x57	0x47	0x57	0x17	0x57
0x54	0x55	0x51	0x55	0x45	0x55	0x15	0x55
0x5C	0x5D	0x59	0x5D	0x4D	0x5D	0x1D	0x5D
0x54	0x55	0x51	0x55	0x45	0x55	0x15	0x55
0x74	0x75	0x71	0x75	0x65	0x75	0x35	0x75
0x54	0x55	0x51	0x55	0x45	0x55	0x15	0x55
0xD4	0xD5	0xD1	0xD5	0xC5	0xD5	0x95	0xD5
0x54	0x55	0x51	0x55	0x45	0x55	0x15	0x55

Tabulka 8. Nastavení hodnoty expandéru 2 pro rozsvícení určitého bodu maticového displeje

Pro splnění prvního úkolu stačí vzít zmíněné dvě tabulky a s časovými prodlevami zapisovat hodnoty na výstupy expandérů. Výsledný program může vypadat následovně:



Obrázek 76. Vývojový diagram programu, který ovládá maticový displej

Pro první program byla vytvořena funkce, matic_disp_postupne_rozsvecovani, která postupně rozsvěcuje jednotlivé diody na maticovém displeji.

```
void matic_disp_postupne_rozsvecovani(void);
```

Když jsme si vyzkoušeli práci s maticovým displejem, tak nám nezbývá nic jiného, než napsat program, který bude zobrazovat libovolné obrázky na displeji. Jedná se pouze o úpravu předchozí funkce.

```
void matic_disp_obraz(char* obraz, int velikost);
```

Funkci předáváme ukazatel na vektor, který obsahuje data, které pixely matice se mají rozsvítit a které ne. Druhý předávaný parametr je počet pixelů.

4.7 Úloha 5: Displej s řadičem HD44780

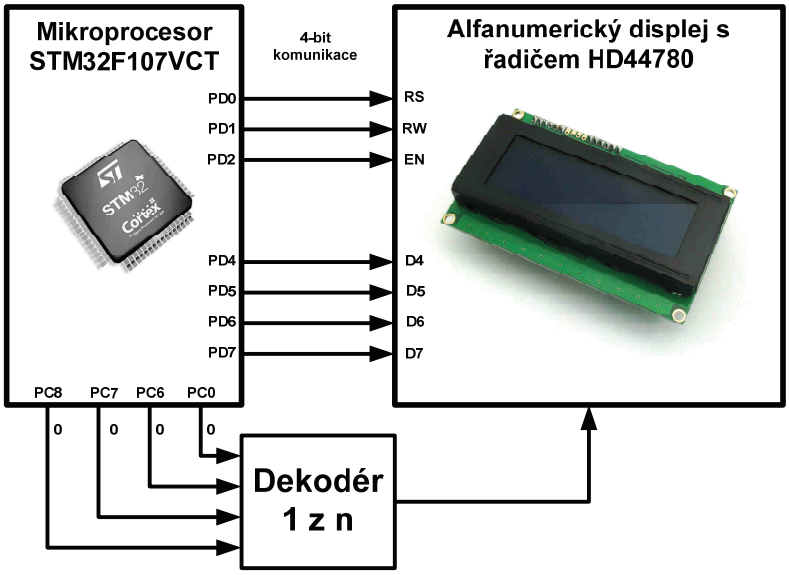
Mezi nejpoužívanější alfanumerické displeje patří displeje obsahující řadič HD44780, na trhu se téměř nesetkáme s alfanumerickými displeji, které obsahují jiný řadič. Tyto displeje se používají především díky jednoduché obsluze a minimálním požadavkům na výkon mikrokontroléru.

4.7.1 Zadání

1) Napište knihovnu pro ovládání alfanumerického displeje s řadičem HD44780. Následně přesměrujte funkci printf na tento displej.

4.7.2 Vypracování

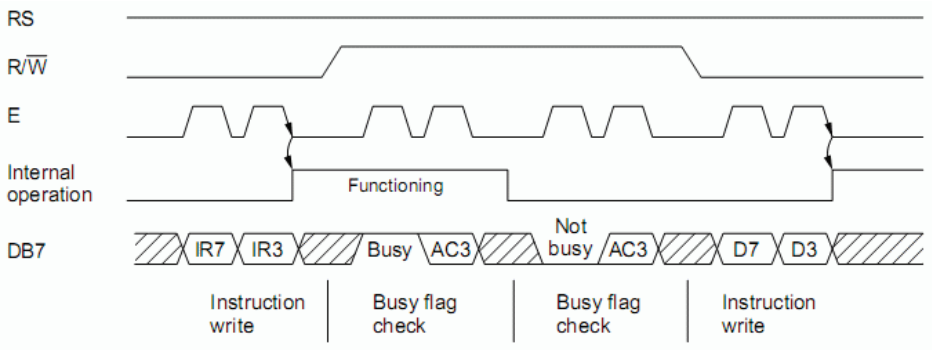
Na tomto kitu je alfanumerický displej připojen ke sdílené sběrnici PD0-PD7, proto je potřeba displej ke sběrnici připojit správným nastavením dekodéru. Komunikace s displejem bude 4-bitová z důvodu úspory komunikačních vodičů. Principielní schéma zapojení je uvedeno níže i se správným nastavením dekodéru.



Obrázek 77. Principielní schéma připojení alfanumerického displeje

S displejem se dá komunikovat 4 bitově nebo 8 bitově (v našem případě je použita 4 bitová komunikace). Řadič displeje používá pro komunikaci tři řídicí vodiče. Řídicím vodičem RS určujeme, zda displeji posíláme řídící příkaz nebo data (0-zápis instrukce, 1-zápis dat). RW slouží k určení, zda budeme z řadiče číst nebo do něj zapisovat (0 – zápis, 1 - čtení). EN slouží pro zahájení zápisu/čtení (1 – povolení zápisu/čtení). [19]

Princip čtyřbitové komunikace je na níže uvedeném obrázku. Po zapsání dvou niblů dat, je potřeba čekat určitou dobu, než je instrukce zpracována řadičem. Lze to řešit čekáním nebo čtením busy flagu. [19]



Obrázek 78. Čtyřbitová komunikace s řadičem HD44780 [19]

Pro řízení alfanumerického displeje s řadičem HD44780 se používají řídicí příkazy. Jejich seznam je v následující tabulce:

Instrukce	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Mazání displeje a nastavení kurzoru na první pozici	0	0	0	0	0	0	0	0	0	1
Nastavení kurzoru na první pozici	0	0	0	0	0	0	0	0	1	*
Směr posunu kurzoru a textu	0	0	0	0	0	0	0	1	I/D	S
Zapnutí displeje, kurzoru a blikání kurzoru	0	0	0	0	0	0	1	D	C	B
Směr posunu kurzoru	0	0	0	0	0	1	S/C	R/L	*	*
Výběr komunikace	0	0	0	0	1	DL	N	F	*	*
Přepnutí zápisu do CGRAM	0	0	0	1	CGRAM adresa					
Přepnutí zápisu do DDRAM	0	0	1	DDRAM adresa						
Čtení busy flagu	0	1	BF	CGRAM/ DDRAM adresa						
Zápis dat	1	0	zápis dat							
Čtení dat	1	1	čtení dat							

Tabulka 9. Řídicí příkazy řadiče HD44780 [19]

Význam jednotlivých bitů v řídících příkazech alfanumerického displeje s řadičem HD44780 je následující:

Bit	Funkce
I/D	Posun kurzoru (0 - vlevo, 1 - vpravo)
S	Posouvání textu (0 – neposouvat, 1 - posouvat)
D	Vypnutí/zapnutí displeje (0 – vypnutí, 1 - zapnutí)
C	Vypnutí/zapnutí kurzoru (0 – vypnutí, 1 - zapnutí)
B	Blikání kurzoru (0 – vypnutí, 1 - zapnutí)
S/C	Posun kurzoru nebo displeje (0 – kurzoru, 1 - displeje)
R/L	Směr posunu (0 – vlevo, 1 - vpravo)
DL	Komunikace (0 – 4-bit, 1 – 8-bit)
N	Počet řádků displeje (0 – jednořádkový, 1 - dvouřádkový)
F	Font (0 – 5x7, 1 – 5x10)
BF	Busy flag (1 - zaneprázdněn)

Tabulka 10. Funkce jednotlivých bitů řídicích příkazů řadiče HD44780 [19]

Poslední důležitou věcí k alfanumerickému displeji s řadičem HD44780 je jeho znaková sada, která je následující:

Obrázek 79. Znaková sada alfanumerického displeje

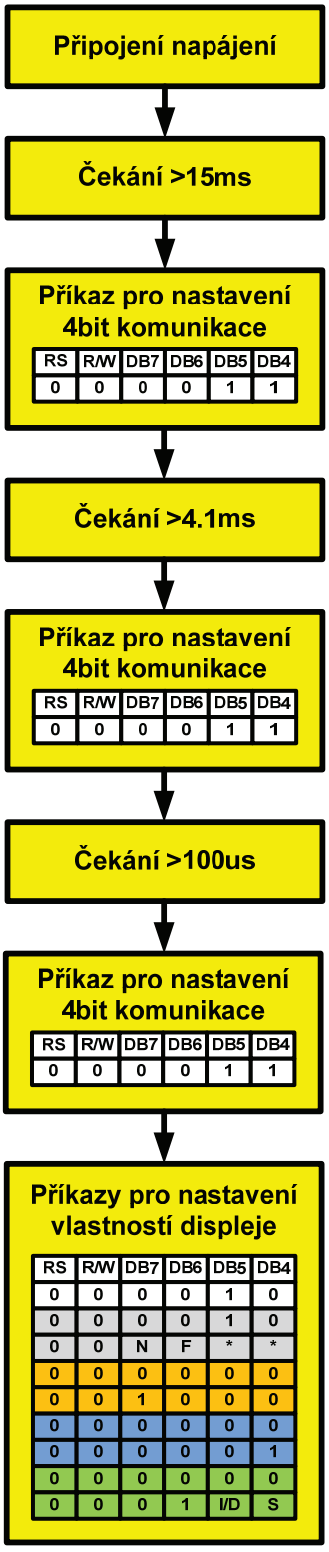
Nyní když jsme se seznámili s řadičem našeho displeje, tak můžeme začít psát knihovnu pro komunikaci s displejem. Pro komunikaci s displejem jsou potřeba dvě základní funkce, jedna pro zápis řídících příkazů a druhá pro zápis dat. Já jsem si funkce napsal tak, že každá odešle vždy jen jeden nibl, proto je potřeba danou funkci volat vždy dvakrát pro odeslání celého příkazu nebo dat. Hlavičky mých funkcí jsou následující:

```
void zapis_data_dsp(char DB7, char DB6,char DB5, char DB4);
void zapis_prikaz_dsp(char DB7, char DB6,char DB5, char DB4);
```

Dále si můžeme napsat jednoduchou funkci pro mazání displeje, která odesílá řadiči příkaz 0x01. Hlavička funkce je následující:

```
void clrscr_dsp(void);
```

Když máme napsány základní příkazy pro odesílání řídících příkazů a dat, tak můžeme napsat inicializační rutinu, která se dle katalogových údajů musí chovat následovně:

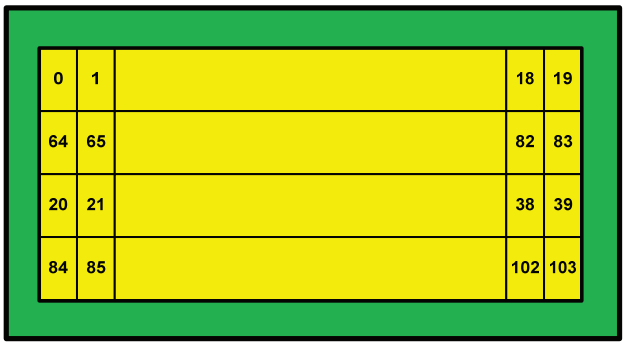


Obrázek 80. Inicializace alfanumerického displeje s řadičem HD44780 (4-bit komunikace)

Výsledkem je napsaná inicializační funkce, která nastaví čtyřbitovou komunikaci, zapne displej, nastaví parametry kurzoru a způsoby rotace displeje.

```
void init_dsp(void);
```

Po inicializaci displeje ještě potřebujeme vytvořit funkci, která nám bude nastavovat kurzor na námi žádanou pozici. Jelikož máme čtyřřádkový displej, se kterým řadič komunikuje jako s dvouřádkovým, proto si vytvoříme funkci, která přepočítá námi zadané souřadnice na souřadnice řadiče. Adresace pozice kurzoru displeje je následující:



Obrázek 81. Adresace pozice kurzoru na displeji [19]

Pro nastavování kurzoru byla vytvořena funkce pozice_dsp, která nastaví kurzor na požadovanou pozici. Pozice se nastaví pomocí x a y souřadnice. Hodnota x souřadnice nabývá hodnot 0-19 a hodnota y souřadnice 0-3.

```
void pozice_dsp(unsigned char x, unsigned char y);
```

Když jsme si vytvořili funkci pro nastavení pozice kurzoru, tak nyní vytvoříme funkci pro výpis znaků na displej, k tomu využijeme funkci printf, jejíž výstup přesměrujeme na displej. K tomuto účelu je CoIDE vytvořena funkce PrintChar v souboru printf.c. Do této funkce napíšeme příkazy pro výpis dat na alfanumerický displej (voláme funkci zapis_data_dsp).

```
void PrintChar(char znak);
```

Tímto jsme přesměrovali výstup funkce printf na alfanumerický displej s řadičem HD44780. Všechny funkce pro práci s displejem jsou v nově vytvořené knihovně lcd_hd44780.h.

4.8 Úloha 6: AD převodníky

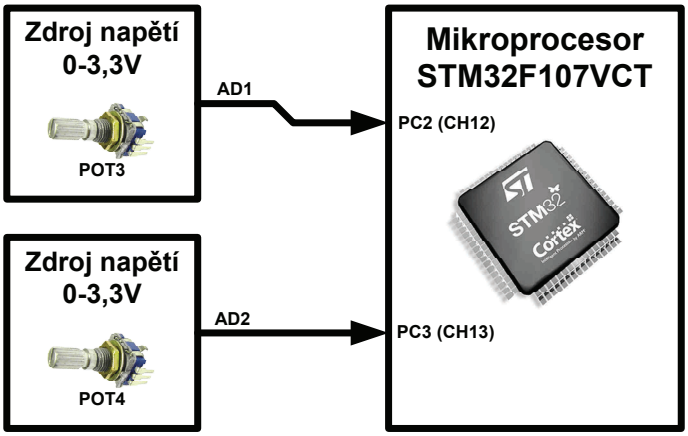
AD převodníky jsou v dnešní době již v podstatě nutností v aplikacích zabývajících se řízením procesů, proto je většina výrobců integruje do většiny nových mikrokontrolérů určených pro embedded systémy.

4.8.1 Zadání

1) Napište program, který bude číst hodnotu na vstupu interních AD převodníků mikrokontroléru STM32F107VCT6 a zobrazovat jejich hodnotu na alfanumerickém displeji.

4.8.2 Vypracování

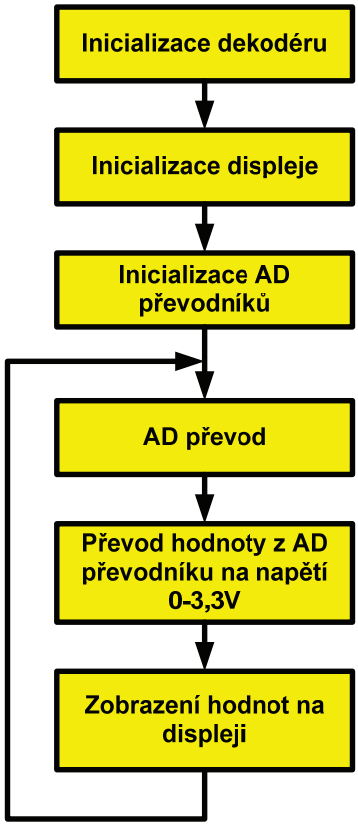
Mikrokontrolér STM32F107VCT6 obsahuje dva 12-bitové AD převodníky s postupnou aproximací. Na vývojové desce jsou připojeny výstupy potenciometrů na kanály 12 a 13 těchto AD převodníků. Principielní schéma zapojení potenciometrů je na následujícím obrázku:



Obrázek 82. Připojení potenciometrů na vstupy interních AD převodníků mikrokontroléru

V programu musí být provedena inicializace dekodéru (připojení alfanumerického displeje ke sdílené sběrnici), inicializace alfanumerického displeje a AD převodníků. V hlavní smyčce se provede obsluha AD převodníků, převod 12-bit hodnoty na napětí 0-3,3V a zobrazení těchto hodnot na displeji.

Vývojový diagram programu pro čtení hodnoty na vstupu AD převodníků je následující:



Obrázek 83. Vývojový diagram programu, který čte a zobrazuje hodnoty napětí na vstupu interních AD převodníků mikrokontroléru STM32F107VCT6

Inicializace displeje a dekodéru byla popsána v předchozích úlohách, proto se zaměřím pouze na inicializaci AD převodníků, k tomuto účelu byla napsána funkce init_ADC. V této funkci byla provedena inicializace GPIO vstupů AD převodníku (nastaven režim AIN pro piny PC2 a PC3), nastavení hodin pro AD převodník a nastavení režimu AD převodu.

Jelikož maximální povolená frekvence pro AD převod je 14MHz, tak je potřeba frekvenci mikroprocesoru (72MHz) snížit pomocí děličky frekvence. Abychom splnili tento požadavek, tak musíme taktovací frekvenci jádra vydělit šesti. AD převodníky pak budou taktovány frekvencí 12MHz. [12]

Pomocí struktury ADC_InitStructure provedeme nastavení obou AD převodníků. Struktura nám nabízí spousty nastavení. Doporučuji zapnout nezávislý kontinuální režim převodu, vypnout externí trigrování a hlavně nezapomenout nastavit správný vstupní kanál (pro toto konkrétní zapojení CH12 nebo CH13).

Při inicializaci je ještě dobré provést automatickou kalibraci ADC převodníků. Nakonec provedeme spuštění AD převodu. Inicializace se provede pomocí funkce init_ADC.

```
void init_ADC(void);
```

V hlavní smyčce programu je pak čtena hodnota z AD převodníku pomocí funkce ADC_GetConversionValue(ADCx), následně je tato hodnota převedena na napětí pomocí funkce prevod_na_napeti a zobrazena na displeji.

```
float prevod_na_napeti(unsigned int ADC_hodnota);
```

Výsledný program včetně podrobnějšího návodu k nastavení AD převodníků je na doprovodném DVD.

4.9 Úloha 7: SPI FLASH

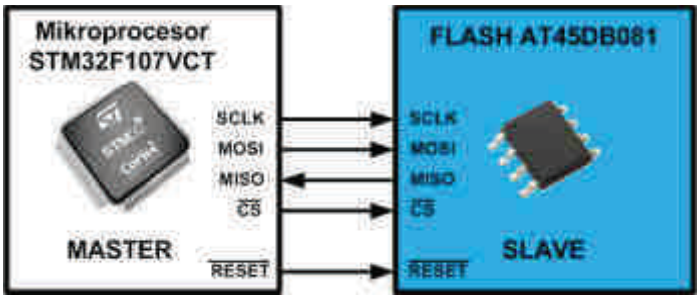
SPI se používá nejčastěji ke komunikaci mezi jedním masterem a více slave zařízeními. Jedná se o sériové synchronní rozhraní. V této úloze je k masteru připojeno pouze jedno slave zařízení (FLASH paměť).

4.9.1 Zadání

1) Napište program, který bude komunikovat pomocí SPI s FLASH pamětí AT45DB081D. Ke komunikaci s pamětí použijte hardwarové SPI rozhraní mikrokontroléru STM32F107VCT6. V prvním kroku nejprve přečtěte z flash paměti ID výrobku a výrobce. Jakmile získáte správné údaje, tak napište funkce pro zápis a čtení a proveďte jednoduchý test části paměti.

4.9.2 Vypracování

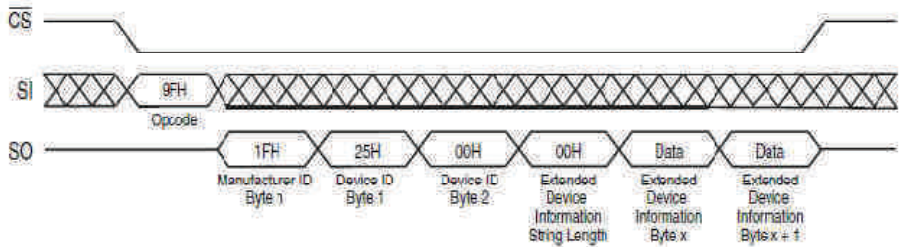
Principielní připojení FLASH paměti k mikrokontroléru je uvedeno na následujícím obrázku:



Obrázek 84. Principielní schéma připojení FLASH paměti k mikrokontroléru

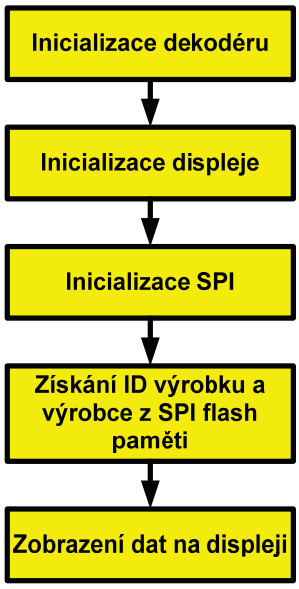
Použitá flash paměť používá pro zápis a čtení dva zásobníky o velikosti 256 bytů. Čtení lze provádět i přímým čtením z paměti na rozdíl od zápisu, který lze provádět pouze pomocí zásobníků (není možný zápis pouze jednoho bytu, jako u EEPROM pamětí). Paměť je organizována do patnácti sektorů, sektory se dělí na bloky a ty se dále dělí na stránky. Paměť obsahuje spoustu příkazů, například i příkaz pro čtení ID výrobku a výrobce (tento příkaz doporučuji využít pro ověření správného nastavení komunikace).

První program bude sloužit pro nastavení SPI rozhranní a čtení ID výrobku a výrobce, k tomu slouží příkaz 0x9F. Po odeslání řídícího příkazu pomocí SPI přečteme dva byty. První byte udává ID výrobce a druhý byte ID výrobku. Komunikace je zachycena na níže uvedeném obrázku.



Obrázek 85. Komunikace po SPI - čtení ID výrobce a ID zařízení [33]

V prvním kroku nastavíme SPI rozhraní mikrokontroléru STM32F107VCT6 a následně odešle příkaz pro čtení dat z SPI flash paměti. Výsledný diagram může vypadat například následovně:



Obrázek 86. Vývojový diagram, který čte ID a výrobce FLASH paměti

Pro inicializaci SPI rozhranní byla napsána funkce SPI_init a pro získání ID výrobce a výrobku funkce SPI_zjisteni_vyrobce_a_ID. Zbylé funkce byly převzaty z minulých úloh.

```
void SPI_init(void);
```

Inicializace se skládá z několika kroků, v prvním kroku je potřeba povolit hodiny SPI1 rozhraní a GPIOA, na němž je SPI1 vyvedeno. Následně se provede inicializace portů. Nakonec je provedena konfigurace SPI rozhraní. U rozhraní se nastaví plný duplex, master mód, délka slova 8 bitů, klidová úroveň hodinového signálu (CPOL=low), čtení dat při první hraně přechodu z klidové úrovně hodin do aktivní (CPHA=vzestupná hrana), nastavení děličky a způsob odesílání dat (MSB/LSB). Tímto je inicializace kompletní. Pro odesílání dat po SPI byla vytvořena funkce SPI_odesly_data a pro příjem dat funkce SPI_prijem_dat. Tyto funkce hlídají, zda již bylo dokončeno vysílání předchozích dat (kontrolují flagy), jakmile je příjem nebo vysílání dokončeno, tak provedou příslušnou operaci.

Na základě obrázku 85 byla napsána funkce pro čtení ID výrobce a výrobku flash paměti. Výsledná funkce vypadá následovně:

```
Vyrobce_a_ID SPI_zjisteni_vyrobce_a_ID(void);
```

Pokud přečteme ID výrobce 0x1F a ID výrobku 0x25, tak nám komunikace po SPI funguje a může se vytvořit funkce pro čtení a zápis dat do flash paměti.

Pro čtení dat z paměti byla vytvořena funkce SPI_cteni_flash, která využívá přímý přístup do flash paměti (není použito čtení přes zásobník):

```
void SPI_cteni_flash(u16 adrr_stranky,u16 adrr_byte_stranky,
                     u16 pocet_byte, u8* buffer);
```

Funkci předáváme adresu stránky, počet bytů, které máme přečíst, a ukazatel na zásobník, kam se přečtená data mají uložit.

Pro zápis dat do flash paměti byla vytvořena funkce SPI_zapis_flash, která zapisuje do paměti pomocí zásobníku (ten se do paměti zapíše po ukončení komunikace s flash pamětí nebo po zapsání 256bytů).

```
void SPI_zapis_flash(u16 adrr_stranky,u16 adrr_byte_stranky,
                     u16pocet_byte, u8* buffer);
```

Funkci předáváme adresu stránky, počet bytů, které máme zapsat, a ukazatel na zásobník, který obsahuje data k zápisu.

Pomocí těchto funkcí napište jednoduchý test paměti. Tento test náhodně vygenerujte data a zapíše je do paměti, zapsaná data následně přečte a pomocí funkce memcmp porovná zapsaná a přečtená data.

4.10 Úloha 8: Externí přerušení a inkrementální snímač

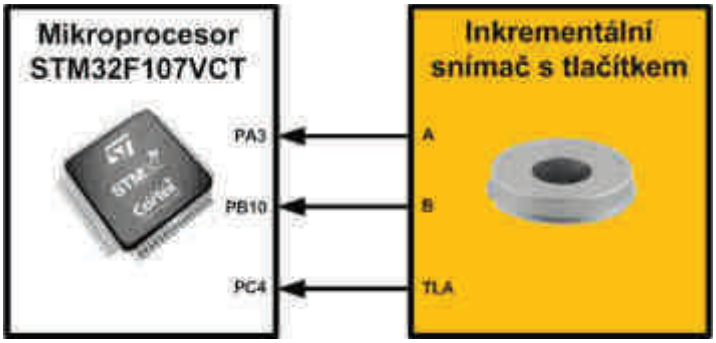
Inkrementální snímače se používají především pro snímání otáček, aby bylo docíleno vysoké přesnosti, tak k vyhodnocení stavu inkrementálního snímače bývá použito externí přerušení.

4.10.1 Zadání

1) Napište program, který pomocí externího přerušení bude zjišťovat směr otáčení inkrementálního snímače a počítat počet impulsů.

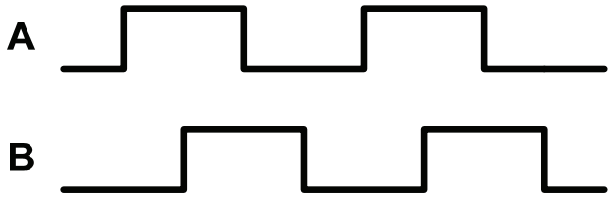
4.10.2 Vypracování

Mikrokontrolér STM32F107VCT6 disponuje externím přerušením, které lze nastavit na téměř všechny jeho piny. Externí přerušení lze využít k vyhodnocování směru otáčení inkrementálního snímače. Principielní schéma zapojení je následující:

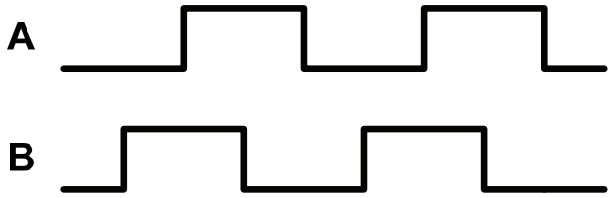


Obrázek 87. Principielní schéma připojení inkrementálního snímače

Inkrementální snímač otáček má dva výstupy navzájem posunuté, pokud jeden výstup přivedeme na vstup externího přerušení mikrokontroléru, tak při každém přerušení nám stačí kontrolovat, zda je na druhém výstupu opačná logická úroveň nebo stejná. Výstup inkrementálního snímače při otáčení jedním směrem je následující:

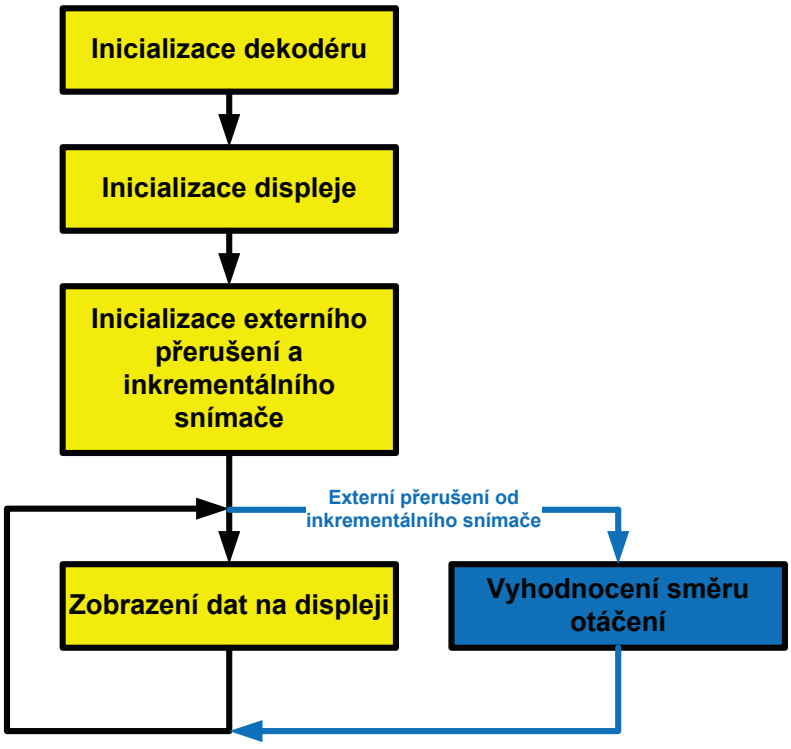


Obrázek 88. Výstup inkrementálního snímače při otáčení doprava



Obrázek 89. Výstup inkrementálního snímače při otáčení doleva

Výsledný program pro obsluhu inkrementálního snímače může vypadat následovně (pro zobrazování je použit alfanumerický displej):



Obrázek 90. Vývojový diagram programu, který obsluhuje inkrementální snímač

Pro inicializaci externího přerušení a inkrementálního snímače byla napsána funkce INK_init. V této funkci jsou piny PA3,PB10 a PC4 nastaveny jako vstupní, je povoleno externí přerušení na pinu PA3 a nastaveny vlastnosti externího přerušení (přerušení je vyvoláno při vzestupné i sestupné hraně).

```
void INK_init(void);
```

Po přerušení skáče program do přerušovací rutiny, která vyhodnotí výstupy inkrementálního snímače. Pokud se otáčí inkrementální snímač doprava, tak je globální proměnná inkrementována, v opačném případě je dekrementována.

```
void EXTI3_IRQHandler(void);
```

4.11 Úloha 9: I2C sběrnice

I2C je sériová sběrnice sloužící ke komunikaci mezi integrovanými obvody v rámci jednoho zařízení. Komunikace probíhá pouze po dvou vodičích SDA (data) a SCL (hodiny). I2C sběrnice může být adresována 7-bitově nebo 10-bitově. Rychlost komunikace je 100kHz nebo v případě rychlé I2C sběrnice až 400kHz. Sběrnice se dá použít i v multi-master režimu, ale ve většině případů je na sběrnici pouze jeden master, který komunikuje s větším množstvím slave zařízení.

4.11.1 Zadání

1) Napište program, který bude číst stav tlačítek připojených ke sběrnici I2C pomocí expandéru PCA9554A a následně zobrazí jejich stav na LED diodách připojených ke sběrnici pomocí stejného expandéru.

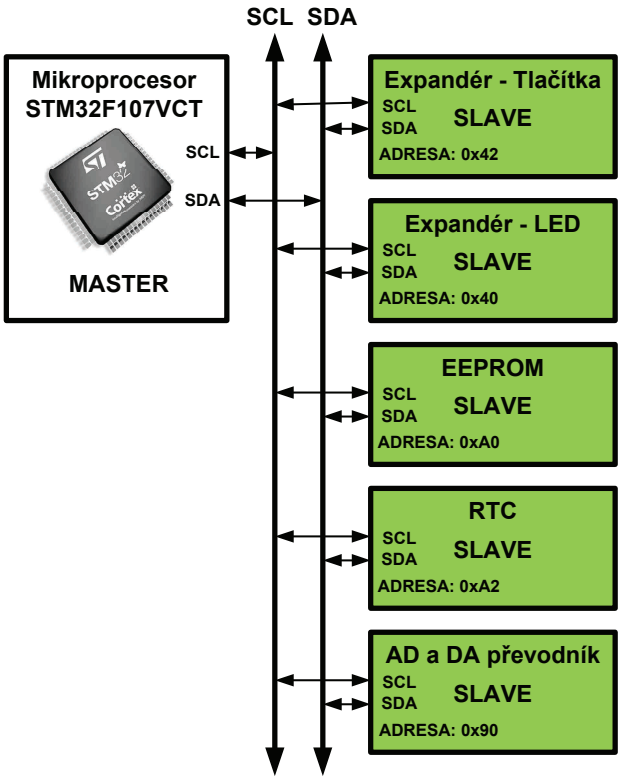
2) Napište program, který bude obsluhovat AD a DA převodník PCF8591.

3) Napište program, který zapíše do EEPROM paměti připojené k I2C sběrnici náhodně vygenerovaná data, následně tato data přečteme a porovnáme je se zapisovanými daty (doporučuji použít funkci memcmp)

4) Poslední program bude sloužit ke čtení přesného času z RTC obvodu. Napište i funkce pro nastavení času.

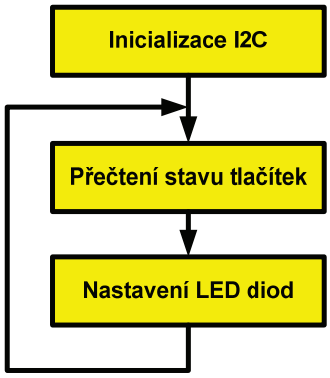
4.11.2 Vypracování

V tomto případě je použita sedmibitová adresace a všechna zařízení na sběrnici jsou schopna komunikovat rychlostí až 400kHz. Principielní schéma zapojení je následující:



Obrázek 91. Principielní zapojení I2C sběrnice na vývojovém kitu

V prvním programu je potřeba zprovoznit I2C komunikaci a naučit se číst data na vstupu expandéru s tlačítky a následně jejich stav zapsat na výstup expandéru s LED diodami. Program je znázorněn na následujícím blokovém diagramu:



Obrázek 92. Vývojový diagram popisující komunikaci s expandéry připojenými k I2C sběrnici

Pro inicializaci I2C sběrnice byla vytvořena funkce I2C_init, která provádí přemapování I2C (na piny PB8 a PB9) a vlastní inicializaci. Během inicializace jsou povoleny hodiny I2C1,GPIOB a AFIO(slouží pro přemapování). Následně jsou inicializovány piny PB8 a PB9 a nakonec inicializace vlastního I2C.

```
void I2C_init(void);
```

Během inicializace I2C byl nastaven režim (I2C), povoleno ACK potvrzování (v případě, že master potřebuje přijatá data potvrdit NACK, tak je potřeba ACK potvrzování vypnout), nastavena 7-bitová adresace a rychlost komunikace.

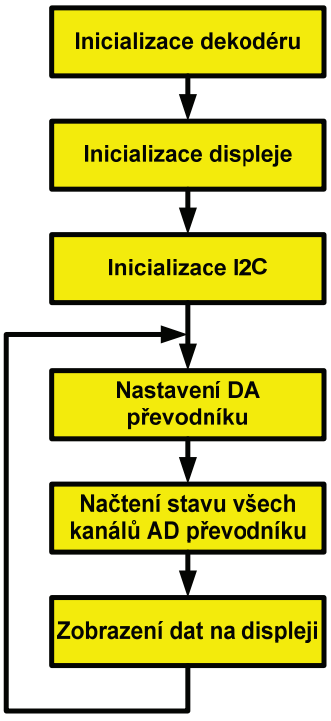
Pro čtení stavu tlačítek byla vytvořena funkce I2C_cti_Tlacitka, která nastaví příslušný expandér jako vstupní a přečte hodnoty na vstupu expandéru.

```
u8 I2C_cti_Tlacitka(void);
```

Pro zápis na LED diody byla vytvořena funkce I2C_odesly_na_LED, která nastaví příslušný expandér jako výstupní a odešle na LED diody požadovaný nový stav.

```
void I2C_odesly_na_LED(u8 data);
```

Druhý program má obsluhovat AD a DA převodník PCF8591. Výsledný program může vypadat následovně:



Obrázek 93. Vývojový diagram popisující komunikaci s PCF8591 připojeného k I2C sběrnici

Obvod PCF8591 obsahuje čtyř-kanálový AD převodník a jeden DA převodník. Na první kanál AD převodníku je připojen snímač teploty, na druhý a třetí kanál jsou připojeny potenciometry a na poslední kanál je připojen výstup DA převodníku. Pro zobrazení hodnot AD převodníku je použit alfanumerický displej.

Pro nastavení hodnoty DA převodníku byla vytvořena funkce I2C_nastav_DA, která obvodu PCF8591 posílá hodnotu 0-255 (DA převodník jí převede na 0-3.3V).

```
void I2C_nastav_DA(u8 hodnota);
```

Dále byla vytvořena funkce pro čtení stavu určitého kanálu AD převodníku. Tato funkce vrací hodnotu 0-255, která reprezentuje napětí 0-3.3V na vstupu AD převodníku.

```
u8 I2C_cti_AD(u8 kanal);
```

Výše uvedenou funkci následně využívá funkce I2C_AD_Teplotni_Cidlo, která vrací teplotu v místnosti ve stupních celsia.

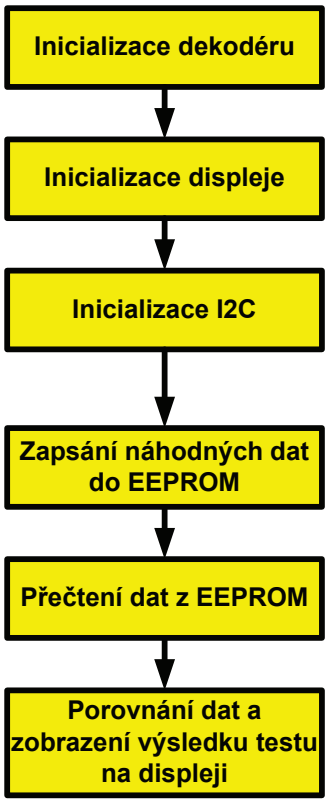
```
float I2C_AD_Teplotni_Cidlo(void);
```

Teplota v místnosti se určí dosazením do následujícího vztahu, ve kterém U reprezentuje napětí na vstupu AD převodníku:

$$Teplota = 100\left(\frac{U}{1{,}65} - 0{,}5\right)°C \qquad (4)[27]$$

Třetí program má provést jednoduchý test EEPROM paměti připojené k I2C sběrnici. Program bude obsahovat dvě funkce, jedna bude sloužit pro zápis a druhá pro čtení. V prvním kroku zapíšeme náhodná data do EEPROM paměti, následně je přečteme a data pomocí funkce memcmp porovnáme. Výsledek testu pak vypíšeme na alfanumerický displej.

Výsledný program může vypadat například takto:



Obrázek 94. Vývojový diagram popisující komunikaci s EEPROM připojené k I2C sběrnici

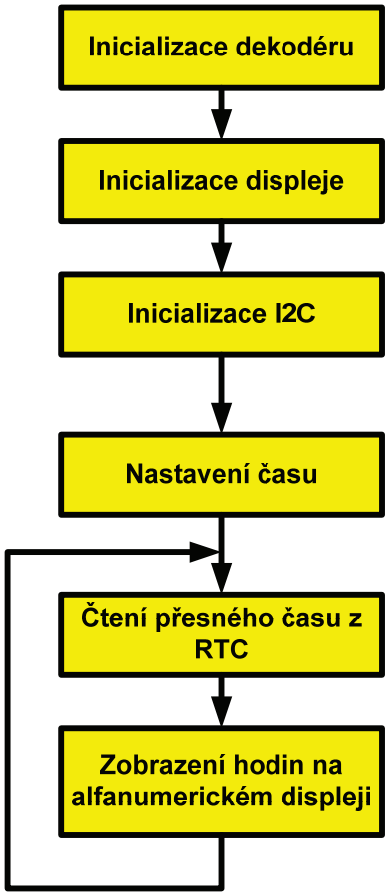
Pro zápis do EEPROM byla vytvořena funkce I2C_Zapis_Buffer_EEPROM, která zapíše obsah předaného zásobníku do EEPROM paměti na požadovanou pozici.

```
void I2C_Zapis_Bufferu_EEPROM(u8* Buffer, u16 AdresaZapisu,
                              u16 PocetBytuKZapisu);
```

Pro čtení dat z EEPROM byla vytvořena funkce I2C_Cteni_EEPROM, které předáváme ukazatel na zásobník, do kterého se uloží přečtená data.

```
void I2C_Cteni_EEPROM(u8* Buffer, uint16_t ReadAddr,
                      uint16_t precist_bytu);
```

Poslední program má číst přesný čas z RTC obvodu připojeného k I2C sběrnici. Součástí programu musí být i funkce pro nastavení času. Vlastní program může vypadat například takto:



Obrázek 95. Vývojový diagram popisující komunikaci s RTC připojeného k I2C sběrnici

Pro nastavení RTC obvodu byla vytvořena funkce I2C_Zapis_RTC, pomocí které můžeme nastavit aktuální čas zápisem hodnoty na příslušnou adresu:

```
void I2C_Zapis_RTC(u8 Prikaz, u8 Data);
```

Touto funkcí se nastaví výchozí čas a nezbývá nic jiného než číst aktuální čas z RTC a vypisovat ho na displej. Pro čtení byla vytvořena funkce I2C_Cti_RTC_Cas, která vrací strukturu, ve které jsou uloženy hodiny, minuty a sekundy v BCD kódu.

```
void I2C_Cti_RTC_Cas(STR_CAS* Cas);
```

4.12 Úloha 10: Grafický displej s dotykovou folií

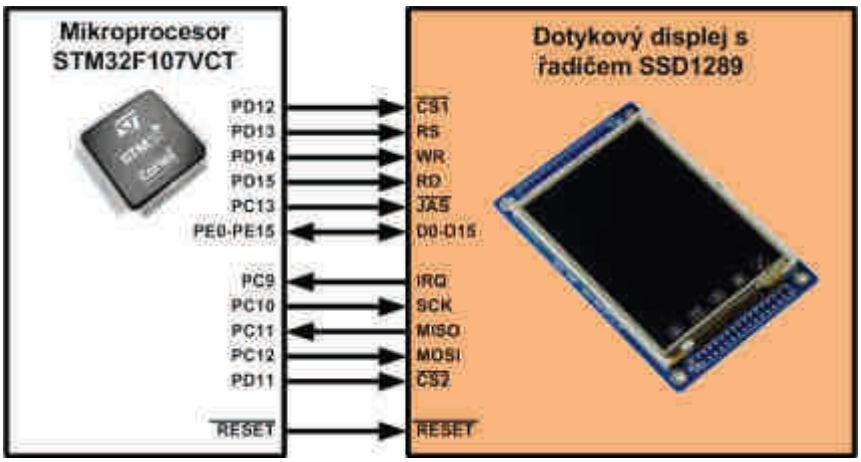
Poslední dobou je zákazníky vyžadováno implementovat do embedded systémů grafické dotykové displeje, proto jsem se rozhodl jej použít i ve svém kitu a vytvořit pro něj jednoduchou ukázkovou úlohu.

4.12.1 Zadání

1) Napište program, který pomocí dotykové plochy grafického displeje bude ovládat LED diody. Naimplementujte základní funkce pro zobrazování dat na displeji.

4.12.2 Vypracování

Součástí vývojového kitu je grafický displej s řadičem SSD1289 a dotykovou folii, která je obsluhována pomocí řadiče ADS7843. S řadičem SSD1289 se komunikuje pomocí 16-bitové paralelní sběrnice. Řadič ADS7843 je připojen pomocí SPI rozhraní.



Obrázek 96. Principielní schéma zapojení grafického displeje

V prvním kroku bude rozchozen řadič SSD1289, který slouží k zobrazování dat na displeji. Funkce pro práci s tímto řadičem jsou obsaženy v knihovně SSD1289.h.

Pro komunikaci s displejem byly vytvořeny dvě základní funkce, první slouží k odesílání řídících příkazů displeji a druhá odesílá data.

```
void SSD1289_Odesli_CMD(u16 Prikaz);
void SSD1289_Odesli_DATA(u16 Data);
```

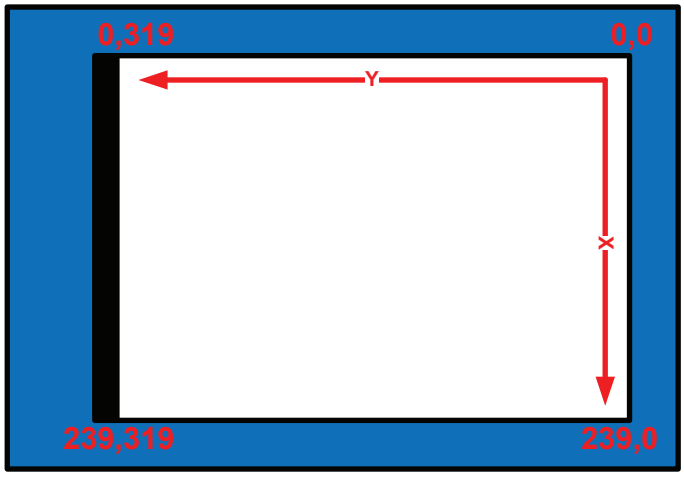
Spojením těchto funkcí vznikla funkce pro odesílání řídícího příkazu a dat, které požaduje řídicí příkaz.

```
void SSD1289_Odesli_CMD_a_DATA(u16 Prikaz,u16 Data);
```

Tato funkce je pak používána pro inicializaci displeje, během inicializace je provedeno nastavení vlastností displeje. Například nastavení souřadného systému, barevné hloubky (nastaveno 65kbitů), zapnutí displeje, nastavení inverze barev, nastavení oscilátoru, napájení a spousta dalšího viz datasheet [39]. Inicializace je provedena funkcí SSD1289_init.

```
void SSD1289_init(void);
```

Souřadný systém displeje byl nastaven následujícím způsobem (počátek souřadného systému je v pravém horním rohu):

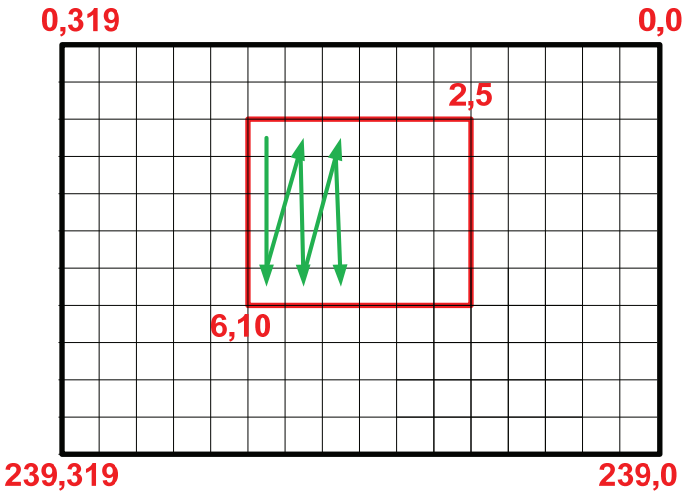


Obrázek 97. Nastavený souřadný systém grafického displeje

Abychom na displej mohli vypisovat data, tak nejprve musíme vybrat oblast, do které budeme zapisovat. Pro výběr oblasti a nastavení kurzoru byla napsána funkce SSD1289_Nastav_Adresu, které předáváme výchozí a koncové souřadnice x a y.

```
void SSD1289_Nastav_Adresu(u16 x1,u16 y1,u16 x2,u16 y2);
```

Funkci v první řadě pošleme příkaz 0x0044, který nastavuje výchozí a koncovou souřadnici x (pro níže uvedený obrázek odešleme data 0x0602), potom nastavíme výchozí y souřadnici příkazem 0x0045 (pro níže uvedený obrázek odešleme data 0x0005) a koncovou souřadnici y příkazem 0x0046 (pro níže uvedený obrázek odešleme data 0x000A). Tímto jsme vybrali plochu, do které budeme zapisovat. Nakonec nastavíme kurzor, to provedeme příkazem 0x004e a 0x004f.



Obrázek 98. Způsob vykreslování na displej s řadičem SSD1289

Jakmile máme vybranou oblast, do které chceme zapisovat, tak stačí začít posílat barvy jednotlivých pixelů v následujícím formátu (je používána 16-bitová hloubka barev):

RRRRRGGGGGGBBBBB

Jestliže potřebujeme změnit barvu pouze jednoho pixelu, tak použijeme funkci SSD1289_Nastav_Barvu_Pixelu, které předáme souřadnice pixelu a barvu.

```
void SSD1289_Nastav_Barvu_Pixelu(u16 x, u16 y, u16 Barva);
```

Dále byly vytvořeny funkce pro překreslování barvy celého displeje a obdélníkové plochy.

```
void SSD1289_Nastav_Barvu_Plochy(u16 Barva);
void SSD1289_Obdelnik(u16 x1, u16 y1, u16 x2, u16 y2, u16 Barva);
```

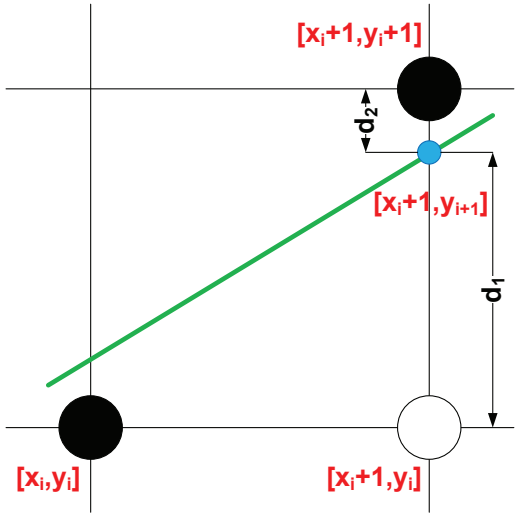
Pokud potřebujeme vykreslit pouze obdélníkový rámec jedné barvy, tak k tomuto účelu byla vytvořena funkce SSD1289_Obdelnik_Ramecek, která vykreslí obdélníkový rámeček požadované šířky a barvy.

```
void SSD1289_Obdelnik_Ramecek(u16 x1, u16 y1, u16 x2, u16 y2,
          u16 BarvaRamecku,u16 SirkaOramovaniPx);
```

Pro vykreslování přímek byla vytvořena funkce využívající pro vykreslování Bresenhamův algoritmus (rasterizace přímky využívající pouze celočíselné operace).

```
void SSD1289_Primka_Bresenham(u16 x1, u16 y1, u16 x2, u16 y2, u16 Barva);
```

Princip Bresenhamova algoritmu je velmi jednoduchý, v principu určuje pouze vzdálenost d1 a d2 od dvou pixelů, mezi kterými se rozhodujeme, který z těchto dvou pixelů náleží naší přímce. Princip je zachycen na následujícím obrázku:



Obrázek 99. Princip rasterizace přímky [31]

Pokud se nacházíme v bodě [x_i, y_i], tak pro výpočet nového bodu přímky inkrementujeme x_i souřadnici a následně dopočítáme novou y_{i+1} souřadnici. Vše vychází z rovnice přímky:

$$y_{i+1} = k(x_i + 1) + q \qquad (5)[43]$$

Pomocí výše uvedeného vztahu můžeme vyjádřit vzdálenosti pixelů [x_i+1, y_i] a [x_i+1, y_i+1] od skutečného bodu přímky [x_i+1,y_{i+1}] následovně:

$$d_1 = k(x_i + 1) + q - y_i \qquad (6)[43]$$

$$d_2 = y_i + 1 - k(x_i + 1) - q \qquad (7)[43]$$

Jelikož požadujeme, aby rasterizace probíhala celočíselně, tak rozhodování nebudeme provádět na základě přesných vzdáleností d_1 a d_2, ale na základě znaménka prediktoru. Prediktor je definován jako rozdíl vzdáleností d_1 a d_2 vynásobený Δx, abychom eliminovali vliv směrnice $k = \Delta y/\Delta x$.Na základě výše uvedených výpočtů vzdáleností upravíme prediktor následovně:

$$Prediktor = (d_1 - d_2) \cdot \Delta x \qquad (8)[43]$$

$$Prediktor = (2k(x_i + 1) - 2y_i + 2q - 1) \cdot \Delta x \qquad (9)[43]$$

$$Prediktor = \left(2\frac{\Delta y}{\Delta x}(x_i + 1) - 2y_i + 2q - 1\right) \cdot \Delta x \qquad (10)[43]$$

$$Prediktor = 2\Delta y x_i - 2\Delta x y_i + 2\Delta y + \Delta x(2q - 1) \qquad (11)[43]$$

Výsledkem těchto úprav je vztah, který počítá hodnotu prediktoru, aniž by použil operaci dělení (z toho vyplývá, že prediktor je celočíselný). U hodnoty prediktoru nás zajímá pouze znaménko, které určuje pixel náležící přímce. Pokud je prediktor záporný, tak se vykreslí pixel [x_i+1, y_i], v opačném případě je vykreslen pixel [x_i+1, y_i+1]. [31]

Funkce pro vykreslení přímky Bresenhamův algoritmus rozšiřuje, aby bylo možné vykreslovat přímky ve všech čtyřech kvadrantech 2D prostoru.

K vykreslení obrázku na celý displej byla vytvořena funkce SSD1289_ZobrazObr, které předáváme ukazatel na pole charů.

```
void SSD1289_Zobraz_Obr(const u8* Image);
```

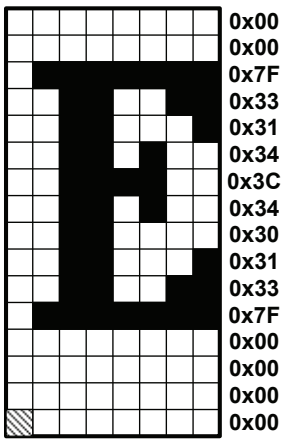
V tomto poli jsou uloženy barvy jednotlivých pixelů. Dva byty určují barvu jednoho pixelu v RGB. Vykreslený obrázek může vypadat například takto:

Obrázek 100. Vykreslení obrázku na grafický displej

Poslední důležitou funkcí je výpis znaku na displej, k tomu slouží funkce SSD1289_Vypis_Znak_8x16, která využívá znakovou sadu 8x16 pixelů. Funkci předáváme pozici, kam má být znak vypsán, barvu znaku, barvu pozadí a ascii hodnotu znaku (char).

```
void SSD1289_Vypis_Znak_8x16(u16 x, u16 y,char Znak,u16 BarvaTextu,
                             u16 BarvaPozadi);
```

Znaková sada je definována pomocí dvourozměrného pole, ve kterém je každý znak definován pomocí 16 bytů. Každý řádek znaku je definován pomocí jednoho bytu (každý bit určuje, zda je daný pixel vybarven nebo ne).



Obrázek 101. ASCII znak

Pro výpis řetězce znaků byla vytvořena funkce SSD1289_Vypis_Retezec_8x16, která automaticky inkrementuje pozici dalšího znaku a opakovaně volá výše uvedenou funkci pro výpis znaku.

```
void SSD1289_Vypis_Retezec_8x16(u16 x, u16 y, u16 BarvaTextu,
     u16 BarvaPozadi,char *RetezecZnaku,u16 PocetZnaku);
```

Tímto byly vytvořeny základní funkce pro zobrazování dat na dotykovém displeji, tyto funkce jsou obsaženy v knihovně SSD1289.h. V dalším kroku byla vytvořena knihovna pro ovládání dotykové plochy (ASD7843.h). Inicializace je prováděna pomocí funkce:

```
void ADS7843_init(void);
```

Komunikace s řadičem ADS7843 je řešena pomocí softwarového SPI, jelikož řídící příkazy mají délku 8bitů a vyčítaná data 12bitů. Pro zápis a čtení byly vytvořeny následující funkce:

```
void SSD1289_TFT_TP_ADS7843_Zapis(u8 data);
  u16 SSD1289_TFT_TP_ADS7843_Cteni(void);
```

Při dotyku je od řadiče ADS7843 generováno přerušení na IRQ výstupu (nastavení do logické jedničky), jakmile tento stav nastane, tak musíme vyčíst místo dotyku, k tomu slouží následující funkce:

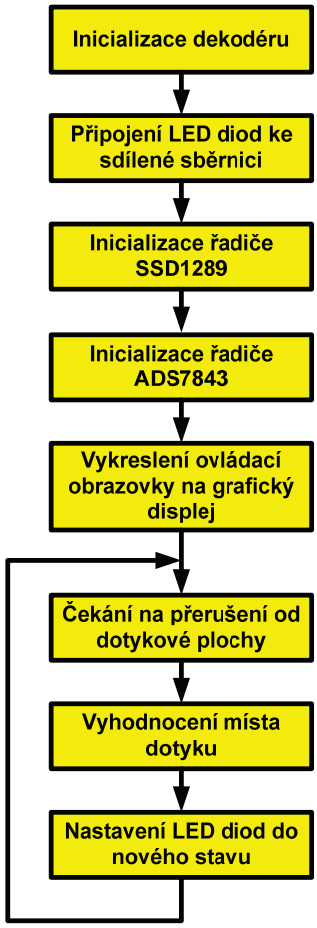
```
void SSD1289_TFT_TP_ADS7843_xy(StrPozice* Pozice);
```

Jelikož se jedná o odporovou dotykovou folii, tak místo dotyku je vyhodnocováno pomocí dvou 12-bitových AD převodníků (ze změny odporu dotykové plochy). Proto vyčtením hodnoty x a y získáme hodnoty 0-4095, které reprezentují místo dotyku.

Protože dotyková plocha má jiný souřadnicový systém (počátek je v levém horním rohu), tak pro lepší přehlednost přepočítáme místo dotyku na souřadnicový systém displeje. Zároveň provedeme přepočet hodnot AD převodníků doškové plochy na rozlišení displeje, abychom získali pixel dotyku. Pro tento účel byla vytvořena funkce:

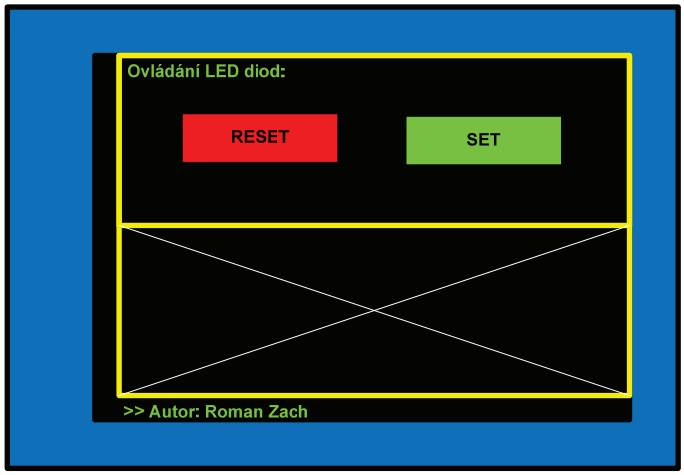
```
void SSD1289_TFT_TP_ADS7843_xy_prepoctena(StrPozice* Pozice);
```

Tímto máme knihovny vytvořené a můžeme psát program pro obsluhu led diod, pro jednoduchost budeme pouze nastavovat nebo resetovat diody na sdílené sběrnici PD0-PD7.



Obrázek 102. Program pro ovládání LED diod pomocí grafického displeje s touch folií

Pomocí výše zmíněných funkcí byla provedena inicializace a nakreslena ovládací plocha, která vypadá následovně (výsledný program je na doprovodném DVD):



Obrázek 103. Ukázková aplikace pro ovládání LED diod

4.13 Úloha 11: Stejnosměrný a krokový motor

V praxi se poměrně často používají krokové motory popřípadě stejnosměrné motory, především kvůli jednoduchosti jejich řízení, proto jsem si nemohl odpustit, aby takovéto prvky chyběly na mém vývojovém kitu.

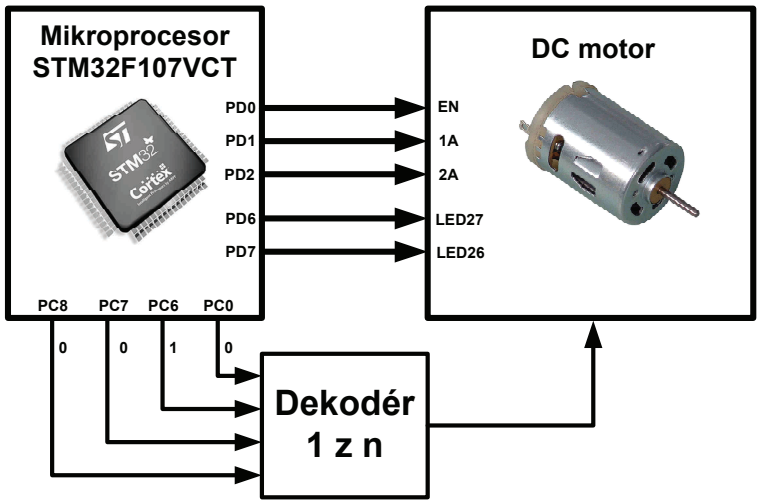
4.13.1 Zadání

1) Napište program, který roztočí DC motor při stisku tlačítka S13 jedním směrem a při stisku tlačítka S14 opačným směrem. Směr otáčení indikujte na diodách LED26 a LED27

1) Napište program, který bude řídit krokový motor pomocí unipolárního řízení. Rychlost otáčení krokového motoru nastavujte pomocí inkrementálního snímače a směr ovládejte pomocí tlačítek S13 a S14.

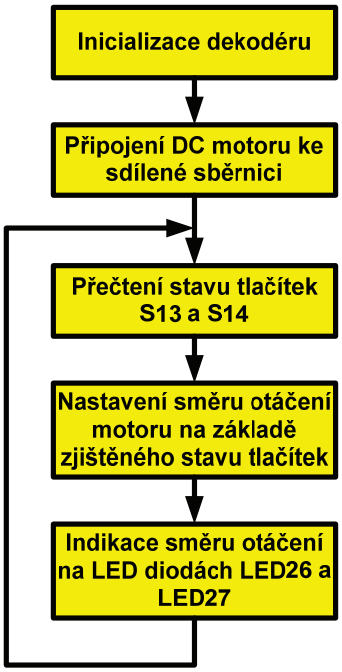
4.13.2 Vypracování

DC motor je připojen pomocí můstkového budiče ke sdílené sběrnici PD0-PD7. Principielní schéma zapojení je následující:



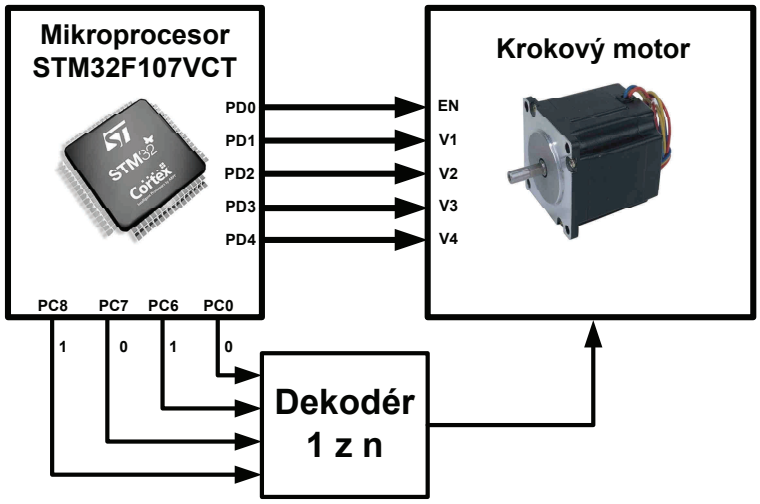
Obrázek 104. Principielní schéma zapojení stejnosměrného motoru

Signálem EN se výstupy budičů obvodu L293, ke kterému je připojen DC motor, přepnou do stavu, který je přiveden na vstupy 1A a 2A. Motor roztočíme tak, že na vstup 1A přivedeme logickou jedničku a na vstup 2A přivedeme logickou nulu, pokud chceme, aby se motor otáčel opačným směrem, tak logické úrovně prohodíme. Výsledný program může vypadat například takto:



Obrázek 105. Vývojový diagram programu ovládajícího stejnosměrný motor

Druhým úkolem je napsat program pro ovládání krokového motoru. Principielní schéma zapojení krokového motoru je následující:



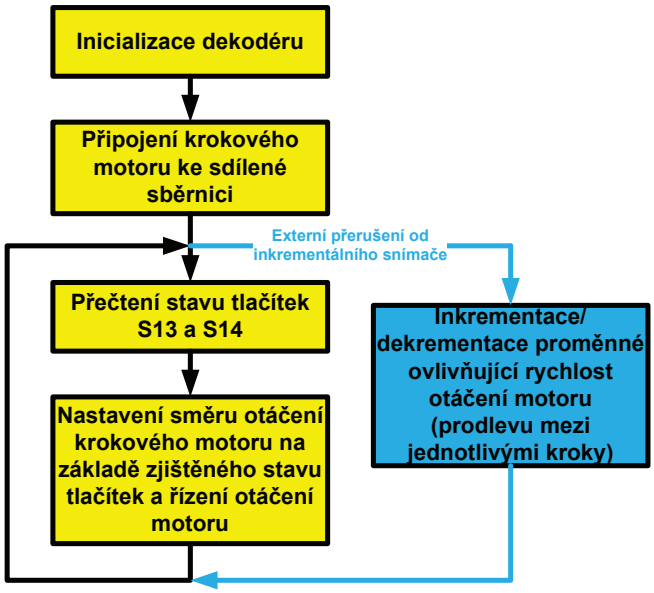
Obrázek 106. Principielní schéma zapojení krokového motoru

U vytvořeného zapojení se nám nabízí využít unipolární dvoufázové řízení. Napětí na jednotlivé cívky krokového motoru budeme přivádět na základě následující tabulky (pokud chceme, aby se krokový motor otáčel opačným směrem, tak sekvence půjde pozpátku):

Krok	V1	V2	V3	V4
1.	0	1	0	1
2.	0	1	1	0
3.	1	0	1	0
4.	1	0	0	1

Tabulka 11. Dvoufázové unipolární řízení krokového motoru

Rychlost otáčení pak nastavíme pomocí zpoždění mezi jednotlivými kroky. Pomocí inkrementálního snímače můžeme inkrementovat/dekrementovat proměnnou, kterou budeme ovlivňovat velikost prodlevy mezi jednotlivými kroky. Výsledný program může vypadat tímto způsobem:



Obrázek 107. Vývojový diagram programu, který řídí krokový motor

Tato úloha je celkem jednoduchá a využívá knihovny vytvořené v předchozích úlohách. Vzorové programy jsou přiloženy na doprovodném DVD stejně jako u všech předchozích úloh.

5 ZÁVĚR

V první části práce byla provedena rešerše vývojových kitů. Pro rešerši byly vybrány vývojové kity tří největších výrobců mikrokontrolérů (STM,NXP a ATMEL). Zaměřil jsem se především na mikrokontroléry s jádrem ARM.

Následně byly hledány způsoby jak dané mikrokontroléry programovat, bylo zvoleno rozhraní JTAG, pomocí kterého je možné program krokovat přímo v mikrokontroléru. V rešerši byly vybrány tři programátory (J-Link, ST-Link a programátory založené na obvodu FT2232). J-Link je komerční programátor, který podporuje programování/debugování mikrokontrolérů všech výrobců, bohužel jeho cena je poměrně vysoká. Programátory založené na obvodu FT2232 patří mezi nekomerční programátory, které pro svůj chod používají OpenOCD ovladače, velkou výhodou je velmi nízká cena a podpora většiny mikrokontrolérů na trhu. Posledním programátorem je ST-LINK, který je omezen na mikrokontroléry STM, jeho výhodou je cena na úrovni programátorů založených na obvodu FT2232.

Pro vývojový kit byl vybrán mikrokontrolér STM32F107VCT6 výrobce STM, jelikož STM patří mezi největší výrobce mikrokontrolérů s jádrem ARM a nabízí programátor/debuger jejich mikrokontrolérů za velmi nízkou cenu.

V druhé části práce byl proveden návrh hardwaru vývojového kitu. Nejprve byla vytvořena bloková struktura celého kitu a následně bylo vytvořeno schéma zapojení a vlastní návrh DPS. Na vývojovém kitu jsou tři displeje (maticový, alfanumerický a grafický), maticová klávesnice, tlačítka, LED diody, stejnosměrný motor, krokový motor, I2C periferie (AD a DA převodník, hodiny reálného času, EEPROM a expandéry), SPI flash paměť, inkrementální snímač otáček, rozhraní připojená pomocí UARTu (RS232, RS485 a USB) a ethernet. Na základě všech komponent byla stanovena celková spotřeba a navržen napájecí zdroj. Pro napájení je použit adaptér 12V, toto napětí je následně pomocí DC/DC měničů sníženo na napětí potřebné pro napájení jednotlivých periferií (3,3V a 5V). Na základě návrhu byla DPS vyrobena, osazena a oživena.

V poslední části práce bylo vybráno nekomerční vývojové prostředí (CoIDE) a překladač (GCC pro ARM). Následně bylo vymyšleno zadání pro jednotlivé periferie a naprogramováno vzorové řešení ve zmíněném vývojovém prostředí.

Literatura

[1] **ARM.** *CORTEX-M3 Technical reference manual* [online]. [cit. 2012-4-8]. URL:<http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337i/DDI0337I_cortexm3_r2p1_trm.pdf>.

[2] **STM.** *Datasheet STM32VLDiscovery* [online]. [cit. 2012-4-8]. URL:<http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATA_BRIEF/CD00277245.pdf>.

[3] **STM.** *Datasheet STM32F100RBT6B* [online]. [cit. 2012-4-8]. URL:< http://pdf1.alldatasheet.com/datasheet-pdf/view/332587/STMICROELECTRONICS/STM32F100RBT6B.html>.

[4] **STM.** *Datasheet STM32VLDiscovery User manual* [online]. [cit. 2012-4-8]. URL:<http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/USER_MANUAL/CD00267113.pdf>.

[5] **ATMEL.** *Datasheet SAM3N Evaluation kit*[online]. [cit. 2012-4-8]. URL:< http://www.atmel.com/Images/doc11080.pdf>.

[6] **ATMEL.** *Datasheet SAM3N* [online]. [cit. 2012-4-8]. URL:< http://www.atmel.com/Images/doc11011.pdf>.

[7] **NXP.** *Datasheet LPCXpresso* [online]. [cit. 2012-4-8]. URL:<http://ics.nxp.com/support/documents/microcontrollers/pdf/lpcxpresso.getting.started.pdf>.

[8] **NXP.** *NXP 32-bit Microcontrollers* [online]. [cit. 2012-4-8]. URL:< http://www.nxp.com/documents/brochure/75017243.pdf>.

[9] **SEGGER.** *Datasheet J-LINK* [online]. [cit. 2012-4-8]. URL:<http://www.segger.com/cms/admin/uploads/productDocs/UM08001_JLinkARM.pdf>.

[10] **FTDI.** *Datasheet FTDI2232D* [online]. [cit. 2012-4-17]. URL:<http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT2232D.pdf>.

[11] **EGNITE GMBH.** *Turtelizer 2* [online]. [cit. 2012-4-17]. URL:< http://www.ethernut.de/en/hardware/turtelizer/index.html>.

[12] **STM.** *Datasheet STM32F107VCT6* [online]. [cit. 2012-4-8]. URL:<http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00220364.pdf>.

[13] **STM.** *Datasheet AN2606 Application note* [online]. [cit. 2013-4-25]. URL:<http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1031/LN1564/PF221020?s_searchtype=partnumber#>.

[14] **MICROCHIP.** *Datasheet MCP120/130* [online]. [cit. 2013-4-25]. URL:<http://ww1.microchip.com/downloads/en/DeviceDoc/11184d.pdf>.

[15] **STM.** *Datasheet UM1075 User manual* [online]. [cit. 2013-4-25]. URL:<http://www.st.com/stwebui/static/active/en/resource/technical/document/user_manual/DM00026748.pdf>.

[16] **TEXAS INSTRUMENTS.** *Datasheet LM2576* [online]. [cit. 2013-4-25]. URL:<http://www.ti.com/lit/ds/symlink/lm2576.pdf>.

[17] **TEXAS INSTRUMENTS.** *Datasheet SN54AHC138,SN74AHC138 3-line to 8-line decoders/demultiplexers* [online]. [cit. 2013-4-25]. URL:<http://www.ti.com/lit/ds/symlink/sn74ahc138.pdf>.

[18] **TEXAS INSTRUMENTS.** *Datasheet SN74LVC4245A* [online]. [cit. 2013-4-25]. URL:<http://www.ti.com/lit/ds/scas375h/scas375h.pdf>.

[19] **HITACHI.** *Datasheet HD44780U (LCD-II)* [online]. [cit. 2013-4-25]. URL:<http://pdf1.alldatasheet.com/datasheet_pdf/view/63673/HITACHI/HD44780.html>.

[20] **TME.** *Datasheet 8*8 Single Color Dot Matrix Displays* [online]. [cit. 2013-4-25]. URL:<http://www.tme.eu/dok/L/lm-88xx19.pdfl>.

[21] **TEXAS INSTRUMENTS.** *Datasheet SN54LVC573A, SN74LVC573A Octal transparent d-type latches with 3-state outputs* [online]. [cit. 2013-4-25]. URL:<http://www.ti.com/lit/ds/symlink/sn74lvc573a.pdf>.

[22] **TEXAS INSTRUMENTS.** *Datasheet L293, L293D Quadruple half-h drivers* [online]. [cit. 2013-4-25]. URL:< http://www.datasheetcatalog.org/datasheet/texasinstruments/l293d.pdf>.

[23] **SGS-THOMSON MICROELECTRONICS.** *Datasheet L293B Push-pull four channel drivers* [online]. [cit. 2013-4-25]. URL:< http://www.datasheetcatalog.org/datasheet/stmicroelectronics/1328.pdf>.

[24] **PORTESCAP.** *Datasheet 26M048b* [online]. [cit. 2013-4-25]. URL:<http://media.digikey.com/pdf/Data%20Sheets/Portescap%20Danaher%20PDFs/26M048B.pdf>.

[25] **NXP.** *Datasheet PCA9554,PCA9554A 8-bit I2C-bus and SMBus I/O port with interrupt* [online]. [cit. 2013-4-25]. URL:<http://www.mouser.com/ds/2/302/PCA9554_9554A-186182.pdf>.

[26] **PHILIPS.** *Datasheet PCF8591 8-bit A/D and D/A converter* [online]. [cit. 2013-4-25]. URL:< http://www.nxp.com/documents/data_sheet/PCF8591.pdf>.

[27] **ANALOG DEVICES.** *Datasheet Low Voltage Tempeature Sensors TMP35/TMP36/TMP37* [online]. [cit. 2013-4-25]. URL:<http://www.analog.com/static/imported_files/data_sheets/TMP35_36_37.pdf>.

[28] **MICROCHIP.** *Datasheet MCP601/1R/2/3/4* [online]. [cit. 2013-4-25]. URL:<http://ww1.microchip.com/downloads/en/DeviceDoc/21314g.pdf>.

[29] **NXP.** *Datasheet PCF8563 Real-time clock/calendar* [online]. [cit. 2013-4-25]. URL:< http://www.nxp.com/documents/data_sheet/PCF8563.pdf>.

[30] **ATMEL.** *Datasheet Two wire seriál EEPROMs AT24C128* [online]. [cit. 2013-4-25]. URL:< http://www.nxp.com/documents/data_sheet/PCF8563.pdf>.

[31] **ZACH, R.** *Analogový indikátor pro průmyslovou automatizaci*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 60s. Vedoucí bakalářské práce doc. Ing. Zdeněk Bradáč, Ph.D.

[32] **DOSTÁL, T.** *Teorie elektronických obvodů*. Skripta FEKT VUT, Brno, 2006.

[33] **ATMEL.** *Datasheet 8-Megabit 2.7-volt Only Seriál DataFlash AT45DB081* [online]. [cit. 2013-4-25]. URL:<http://pdf1.alldatasheet.com/datasheet_pdf/view/56155/ATMEL/AT45DB081.html>.

[34] **ZIPPY TECHNOLOGY CORP.** *KC Series Rotary Switches.* Zippy.com.tw [online].©2011 [cit. 2013-4-25]. URL:<http://www.zippy.com.tw/con_product_detail.asp?ps_rfnbr=337&pcs_rfnbr=27&ps_code=KC%20Series&pcs_type=1&lv_rfnbr=2#Ordering%20Informationl>.

[35] **MAXIM.** *Datasheet MAX3232* [online]. [cit. 2013-4-25]. URL:< http://www.datasheetcatalog.org/datasheets/270/497537_DS.pdf>.

[36] **FTDI.** *Datasheet FT232R USB UART IC Version 2.10* [online]. [cit. 2013-4-25]. URL:<http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>.

[37] **MAXIM.** *Datasheet MAX3485* [online]. [cit. 2013-4-25]. URL:< http://datasheets.maximintegrated.com/en/ds/MAX3483-MAX3491.pdf>.

[38] **TEXAS INSTRUMENTS.** *Datasheet ADS7843* [online]. [cit. 2013-4-25]. URL:< http://www.ti.com/lit/ds/sbas090b/sbas090b.pdf>.

[39] **SOLOMON SYSTECH.** *Datasheet SSD1289* [online]. [cit. 2013-4-25]. URL:< http://www.kosmodrom.com.ua/el/STM32-TFT/SSD1289.pdf>.

[40] **TEXAS INSTRUMENTS.** *AN-1405 DP83848Single 10/100 Mb/s Ethernet Transceiver Reduced Media Independent Interface (RMII) Mode* [online]. [cit. 2013-4-25]. URL:< http://www.ti.com/lit/an/snla076/snla076.pdf>.

[41] **HANRUN.** *RJ45 Connector with integrated magnetics* [online]. [cit. 2013-4-25]. URL:< http://www.hqew.net/datasheet/HR911105A-datasheet-pdf-download-273480.html>.

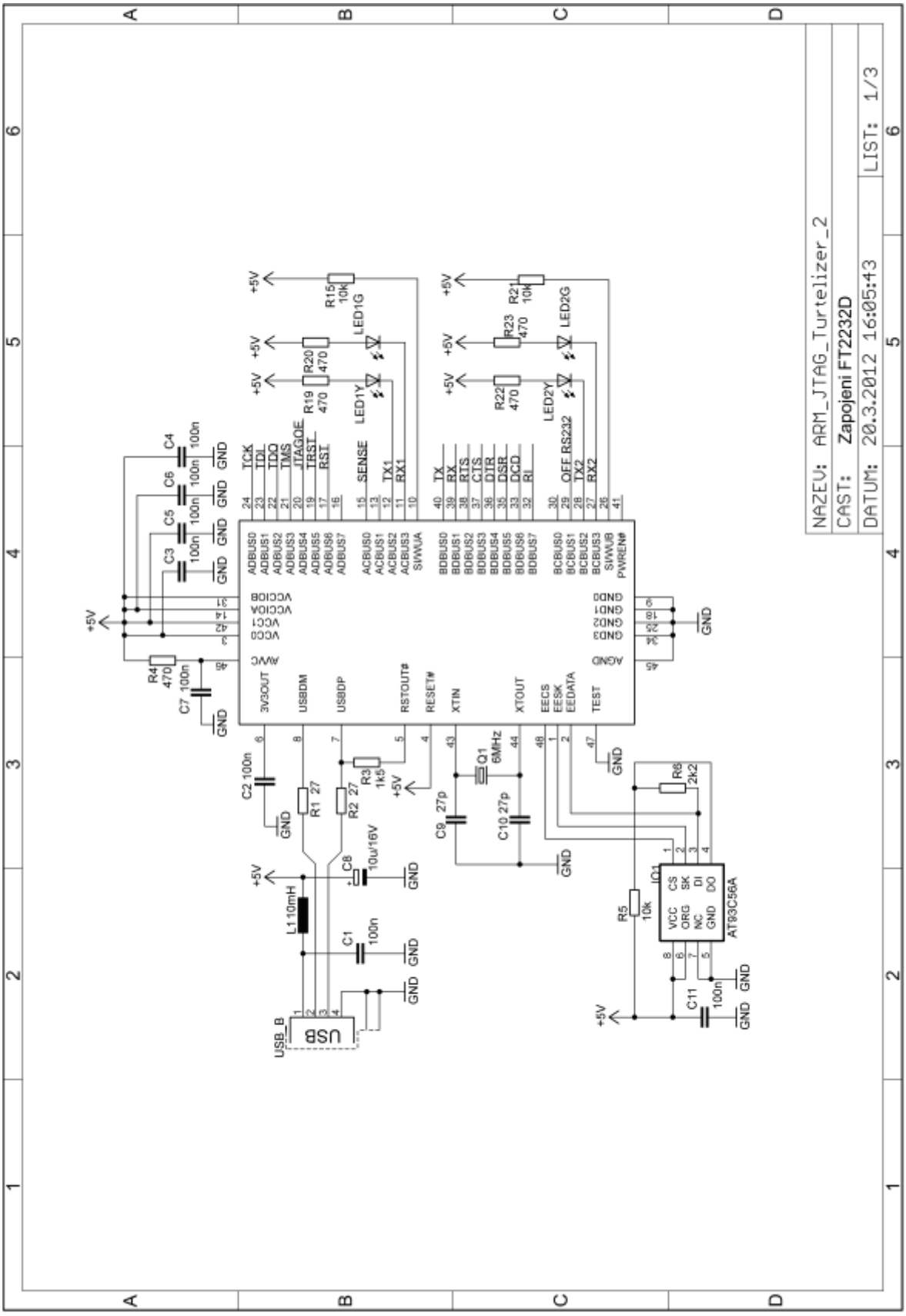
[42] **SEMTECH.** *CFPS-72,-73 Semtech Reference – Stratum 4* [online]. [cit. 2013-4-25]. URL:< http://pdf1.alldatasheet.com/datasheet-pdf/view/105306/SEMTECH/CFPS-73.html>.

[43] **KRŠEK, P.** *Základy počítačové grafiky*. Studijní opora FIT VUT, Brno, 2003.

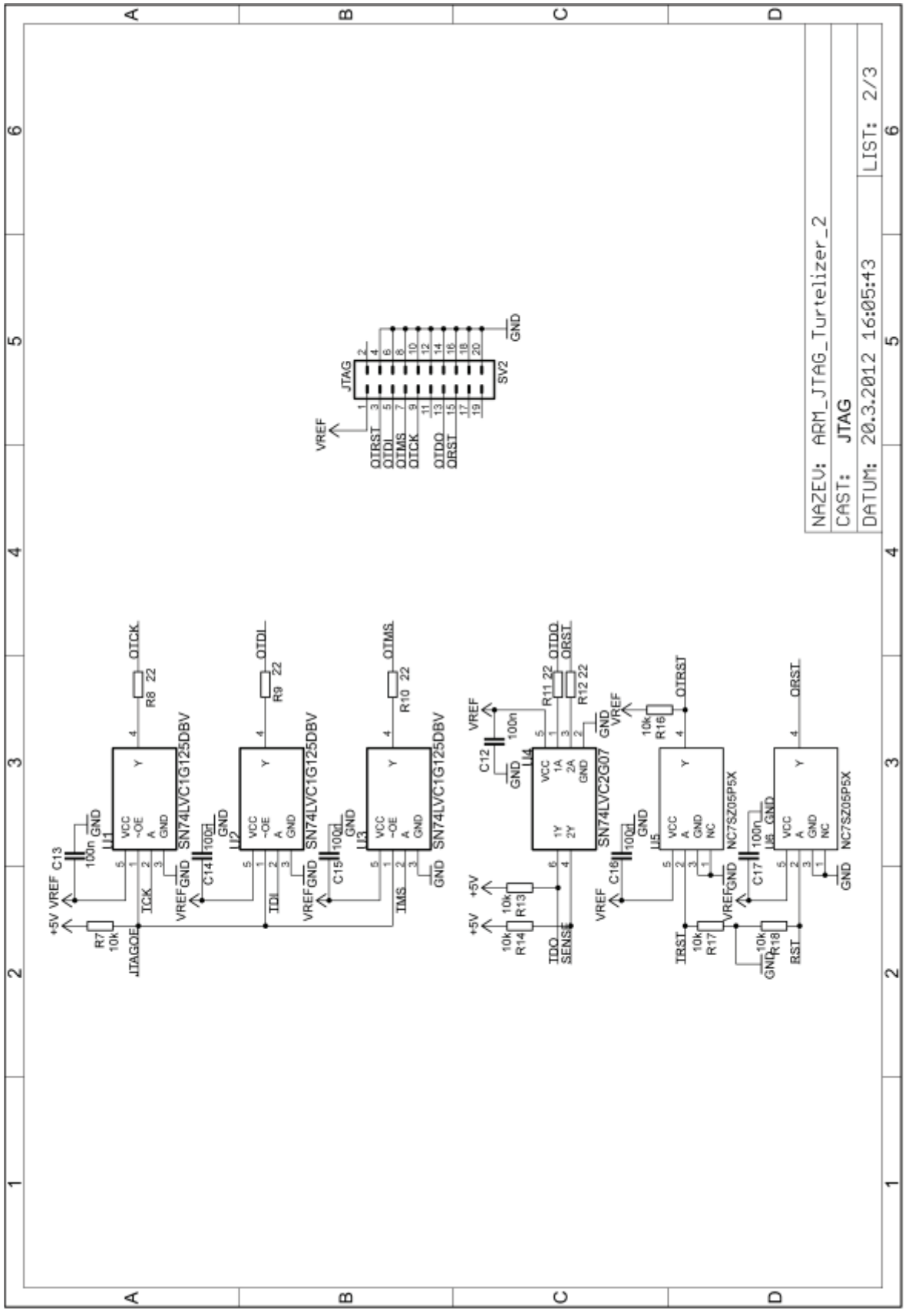
Přílohy:

Přílohy na DVD:

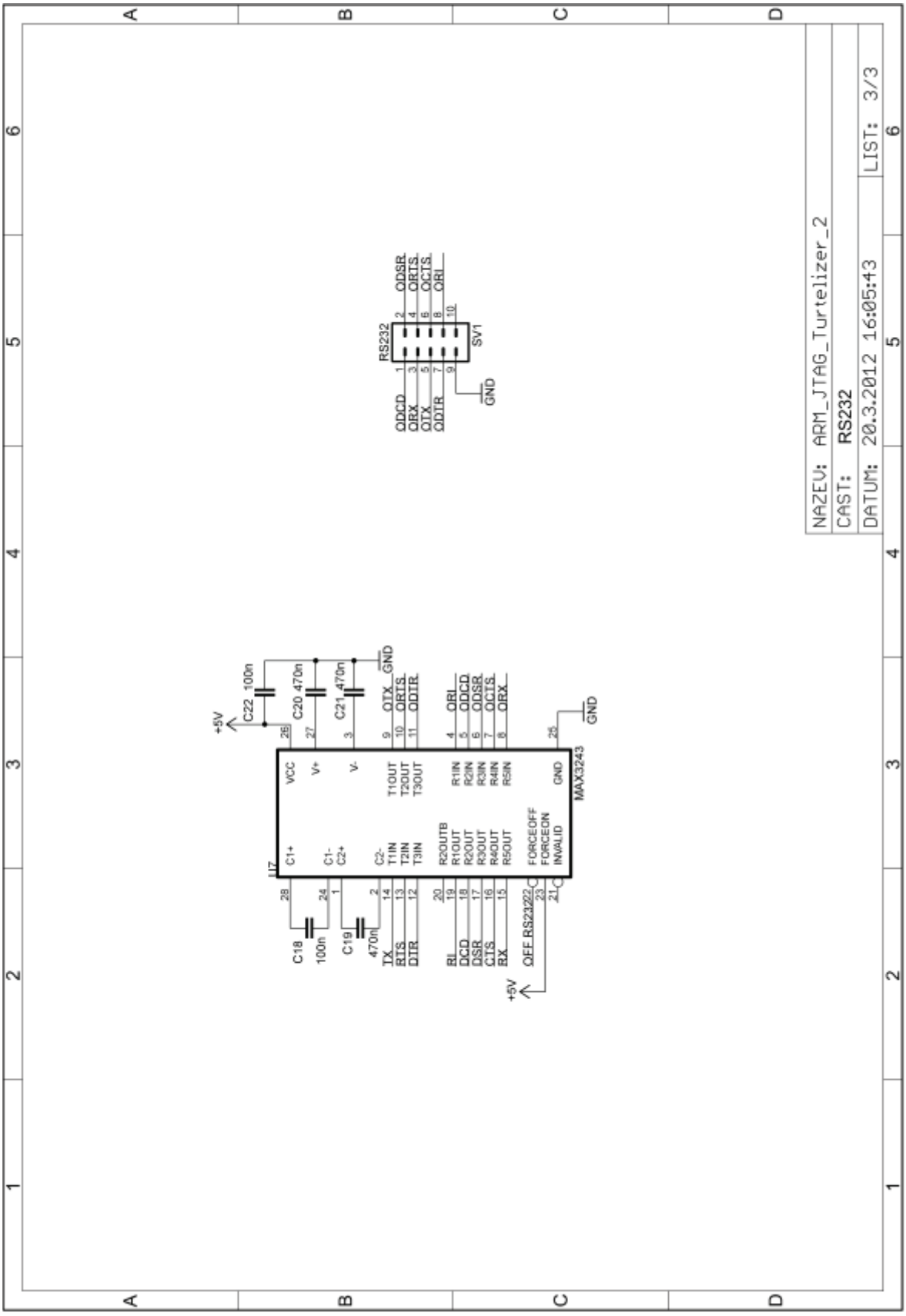
Datasheety použitých obvodů
Dokumentace k diplomové práci
Návrh konstrukce v programu EAGLE 5.7
Zdrojové soubory pro mikrokontrolér STM32F107VCT6
Návody k jednotlivým úlohám



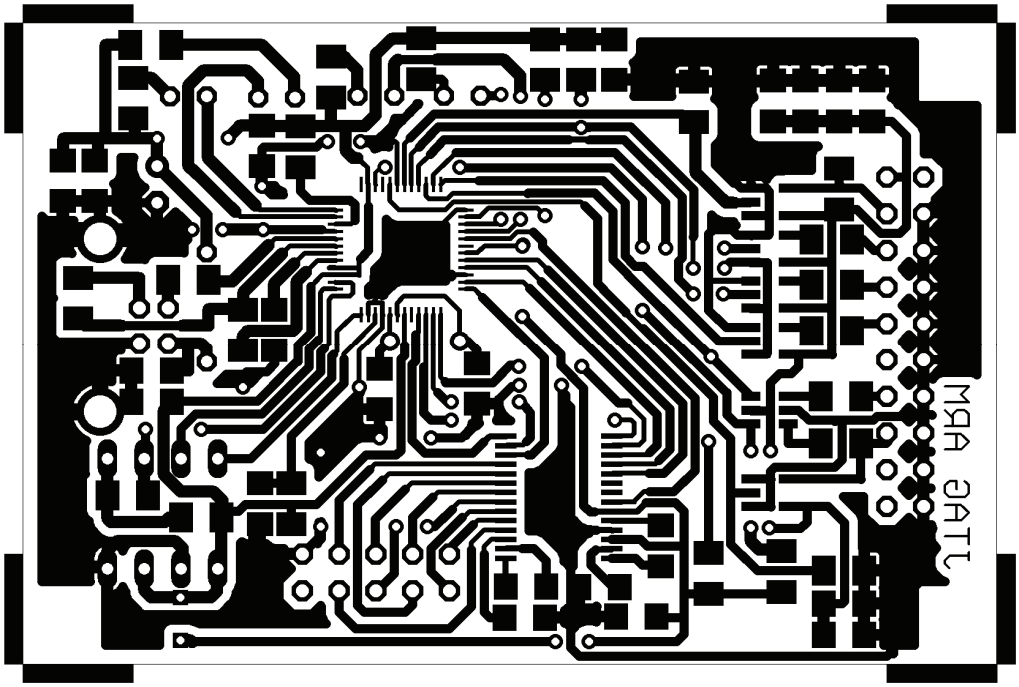
Příloha 1. Programátor Turtelizer 2 - FT2232D



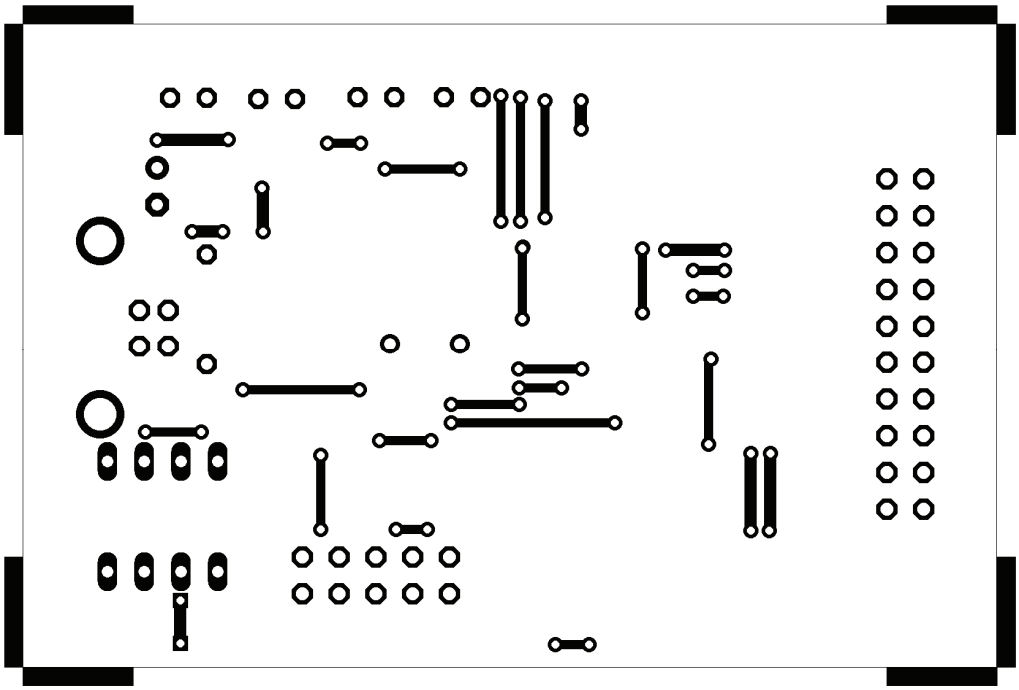
Příloha 2. Programátor Turtelizer 2 - JTAG V/V obvody



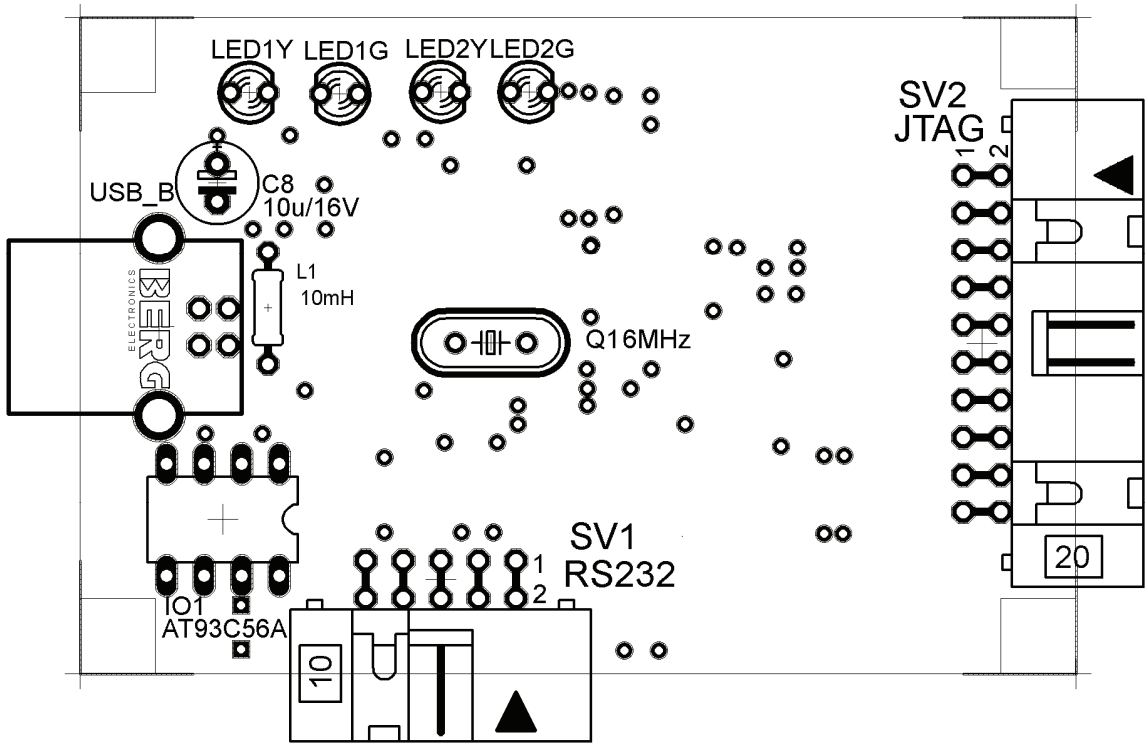
Příloha 3. Programátor Turtelizer 2 - RS232 převodník



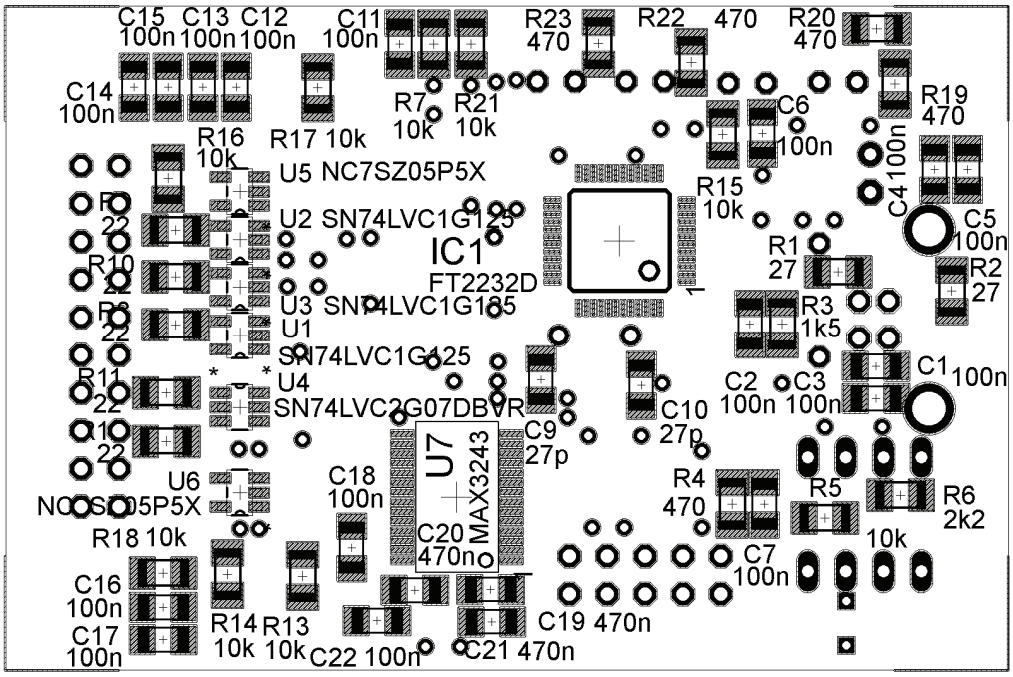
Příloha 4. Programátor Turtelizer 2 - DPS (2:1) – bottom



Příloha 5. Programátor Turtelizer 2 - DPS (2:1) – top



Příloha 6. Programátor Turtelizer 2 - Osazovací plán – top



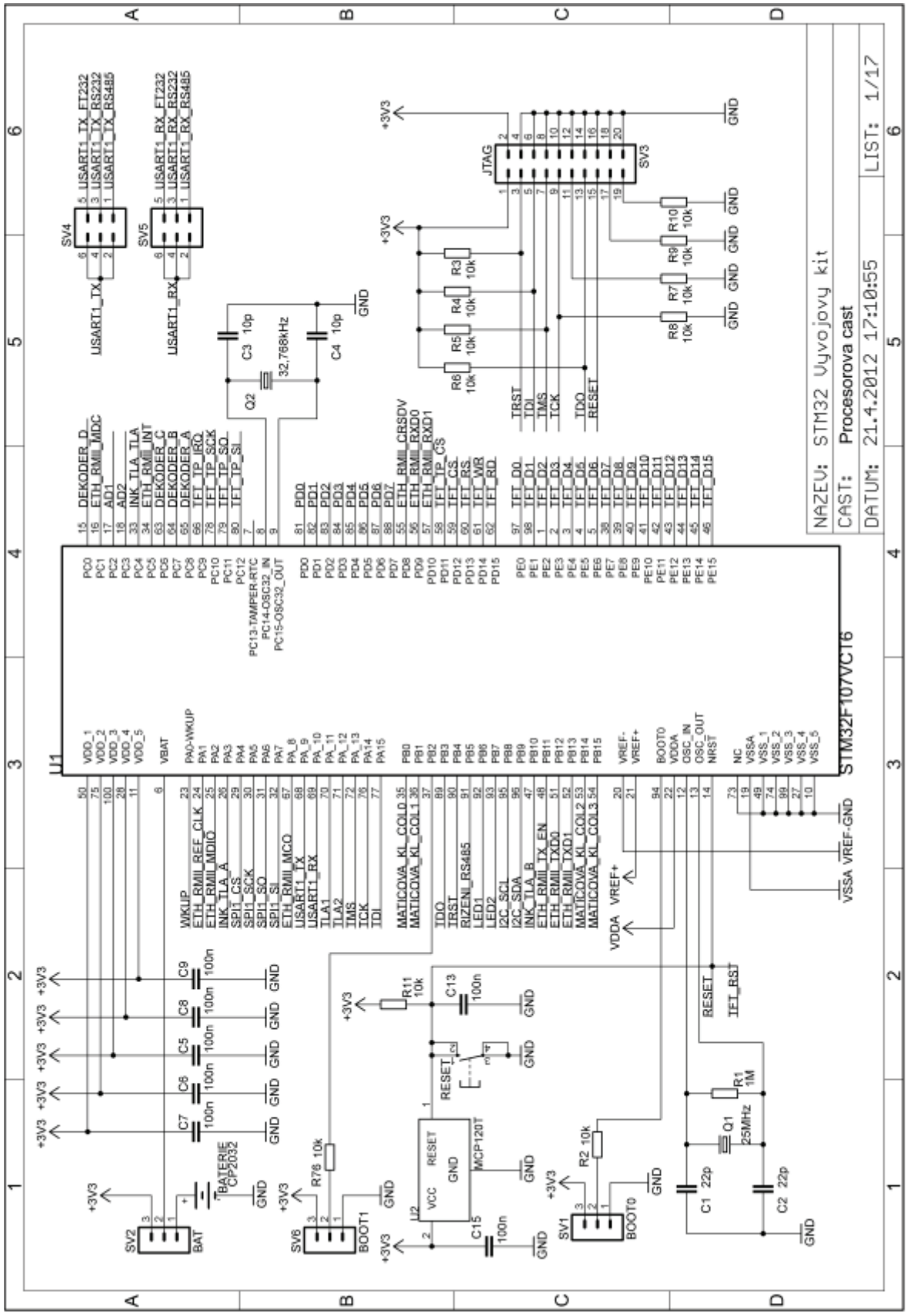
Příloha 7. Programátor Turtelizer 2 - Osazovací plán – bottom

Součástka	Hodnota	Pouzdro
C1	100n	C1206
C2	100n	C1206
C3	100n	C1206
C4	100n	C1206
C5	100n	C1206
C6	100n	C1206
C7	100n	C1206
C8	10u/16V	E2,5-6E
C9	27p	C1206
C10	27p	C1206
C11	100n	C1206
C12	100n	C1206
C13	100n	C1206
C14	100n	C1206
C15	100n	C1206
C16	100n	C1206
C17	100n	C1206
C18	100n	C1206
C19	470n	C1206
C20	470n	C1206
C21	470n	C1206
C22	100n	C1206
IC1	FT2232D	LQFP48
IO1	AT93C56A	DIL08
LED1G	LED3mm	LED3mm
LED1Y	LED3mm	LED3mm
LED2G	LED3mm	LED3mm
LED2Y	LED3mm	LED3mm
Q1	6MHz	HC49/S
R1	27Ω	M1206
R2	27Ω	M1206
R3	1,5kΩ	M1206
R4	470Ω	M1206
R5	10kΩ	M1206
R6	2,2Ω	M1206
R7	10kΩ	M1206
R8	22Ω	M1206
R9	22Ω	M1206
R10	22Ω	M1206
R11	22Ω	M1206
R12	22Ω	M1206
R13	10kΩ	M1206
R14	10kΩ	M1206
R15	10kΩ	M1206

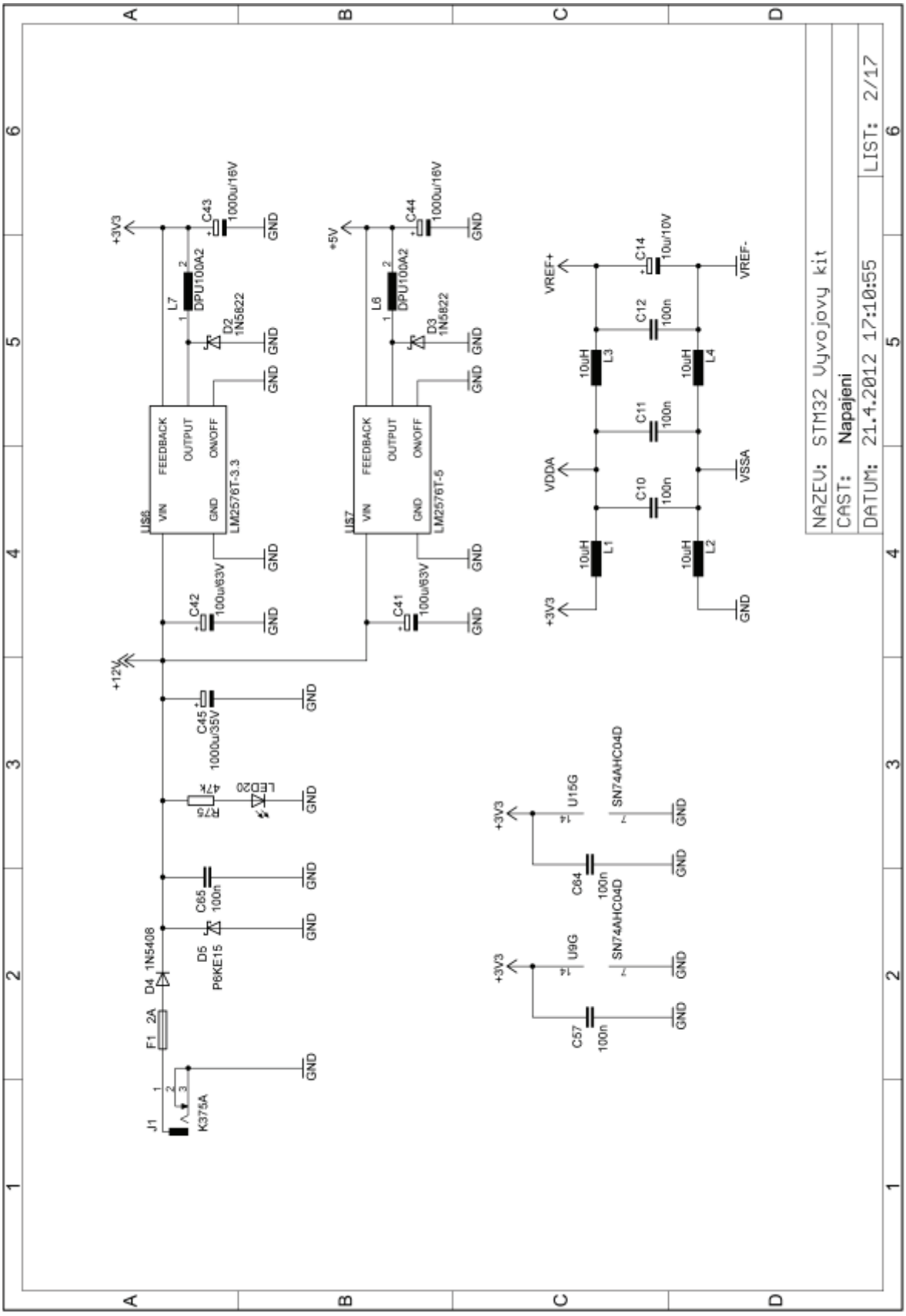
Příloha 8. Programátor Turtelizer 2 - Seznam součástek 1

Součástka	Hodnota	Pouzdro
R16	10kΩ	M1206
R17	10kΩ	M1206
R18	10kΩ	M1206
R19	470Ω	M1206
R20	470Ω	M1206
R21	10kΩ	M1206
R22	470Ω	M1206
R23	470Ω	M1206
SV1		ML10L
SV2		ML10L
U1	SN74LVC1G125	SOT23
U2	SN74LVC1G125	SOT23
U3	SN74LVC1G125	SOT23
U4	SN74LVC2G07DBVR	SOT23
U5	NCSZ05P5X	SOT23
U6	NCSZ05P5X	SOT23
U7	MAX3243	SSOP28
USB_B		USB_B

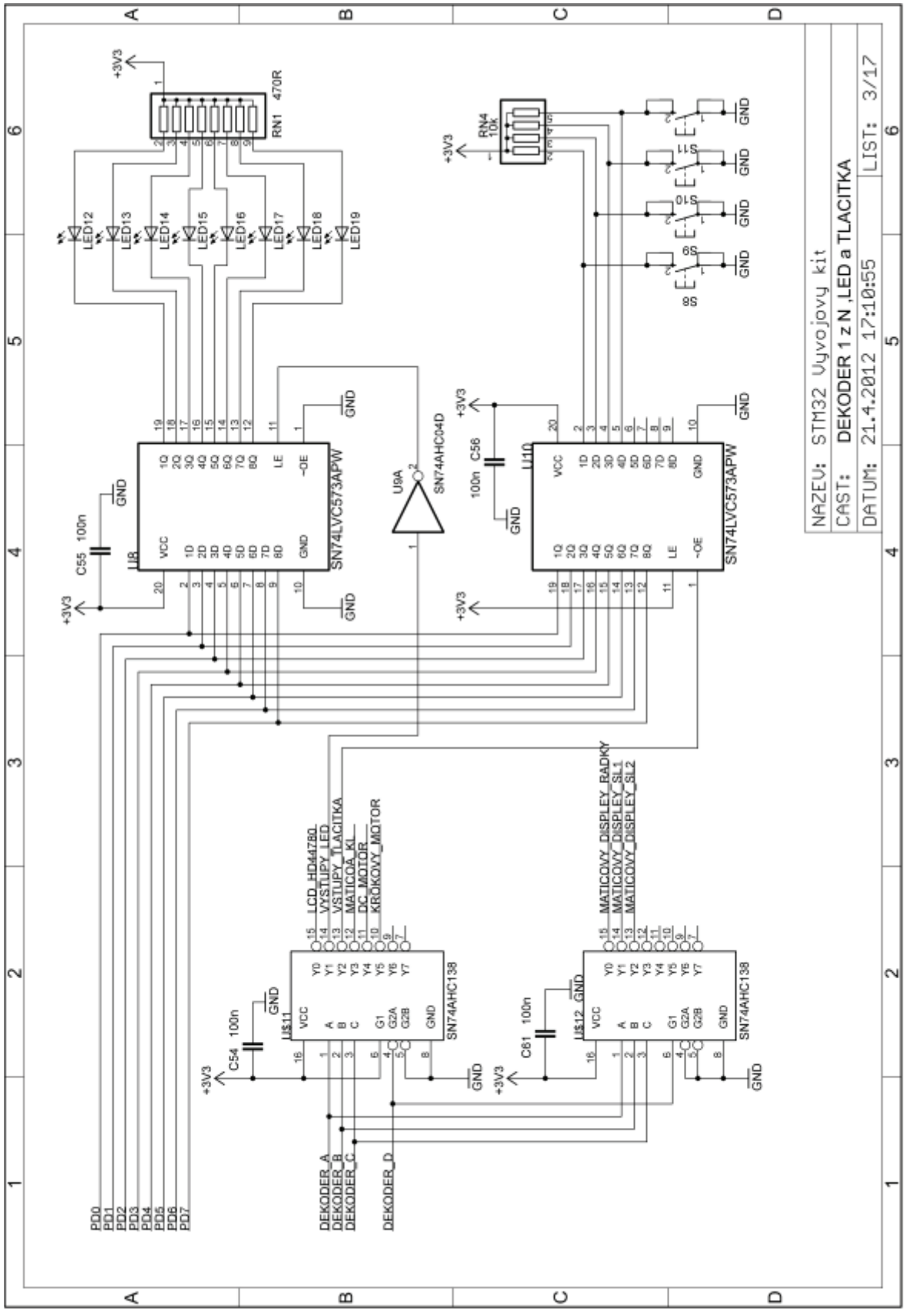
Příloha 9. Programátor Turtelizer 2 - Seznam součástek 2



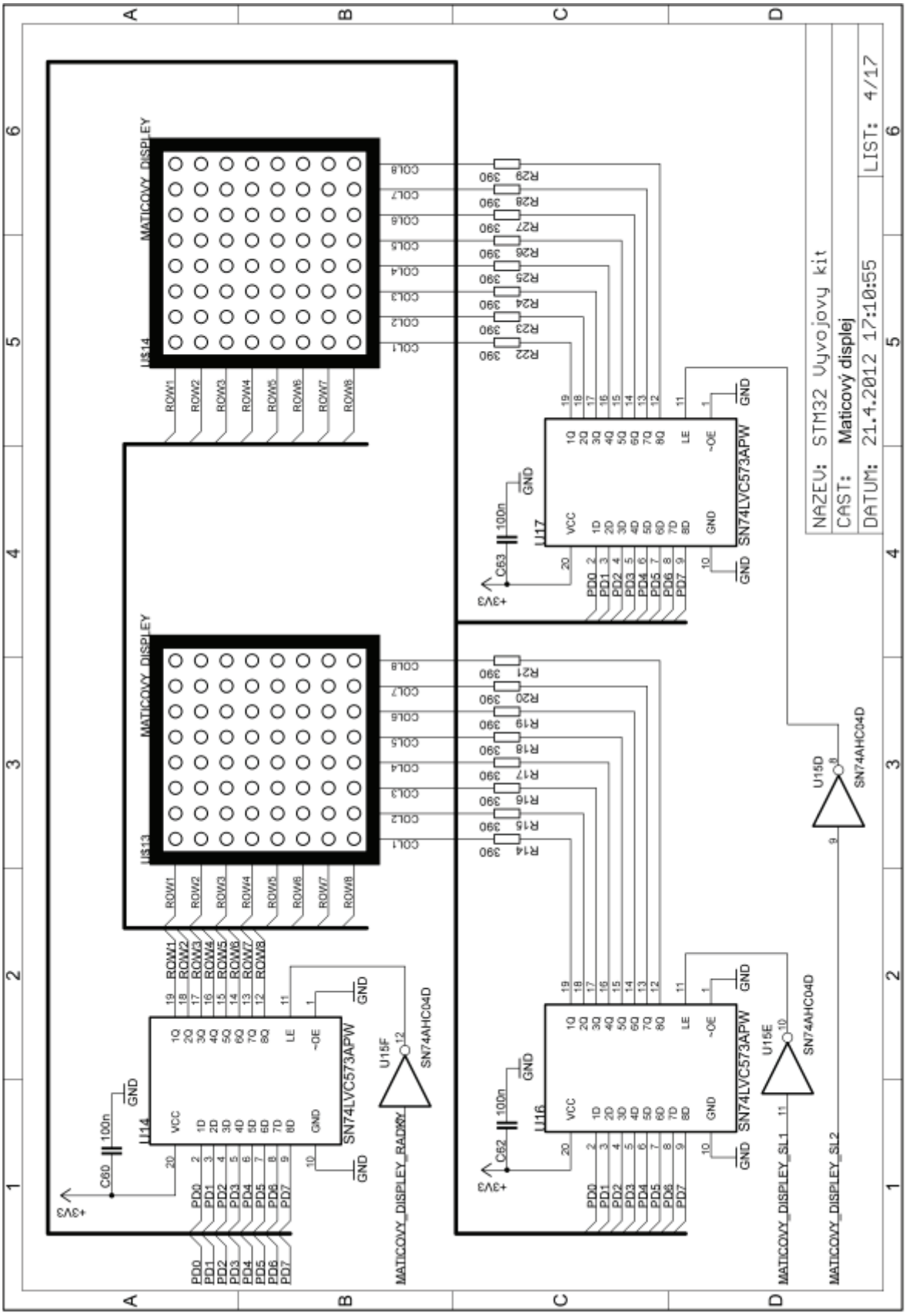
Příloha 10. Vývojový kit - Mikrokontrolér STM32F107VCT6



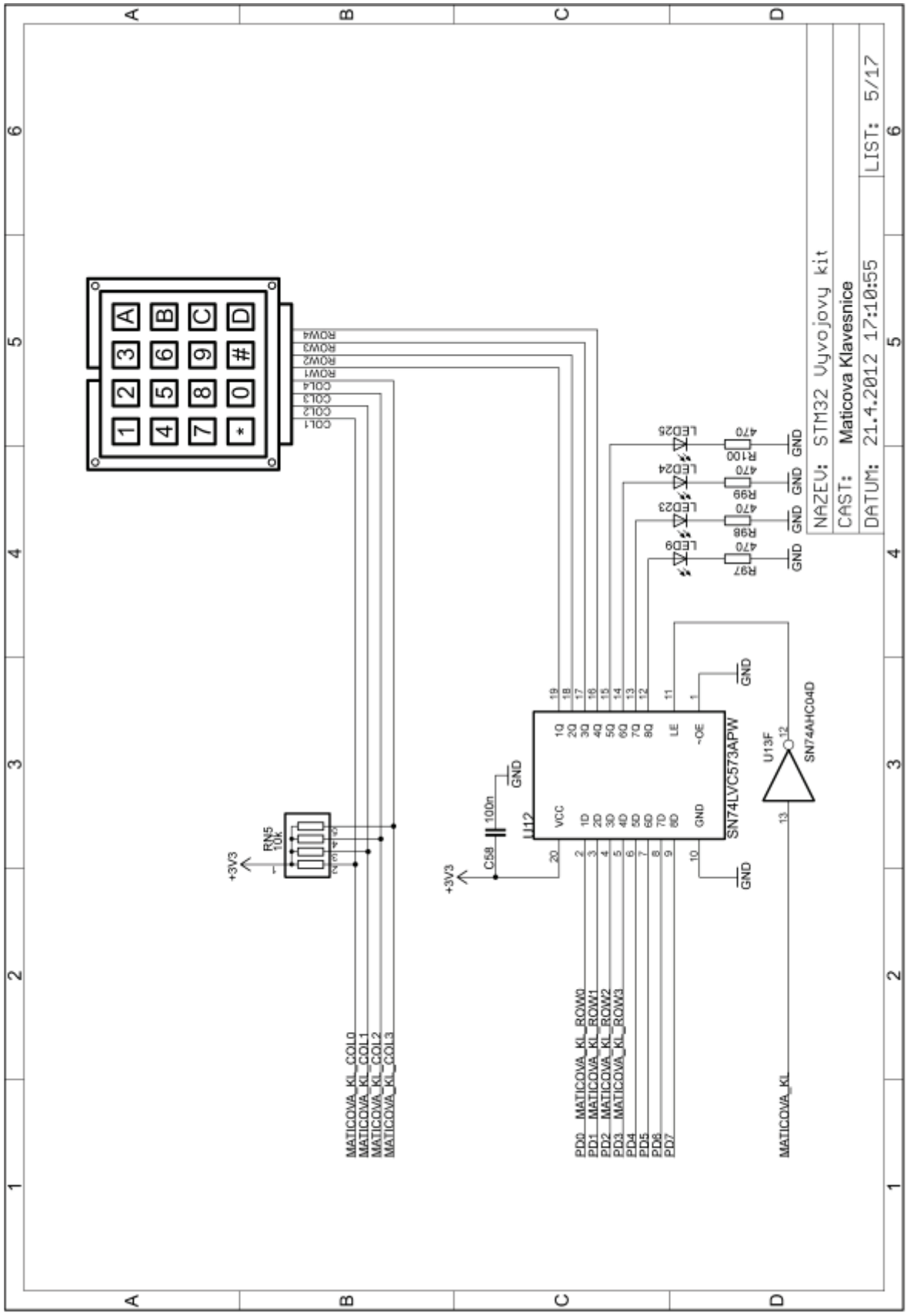
Příloha 11. Vývojový kit – Napájení



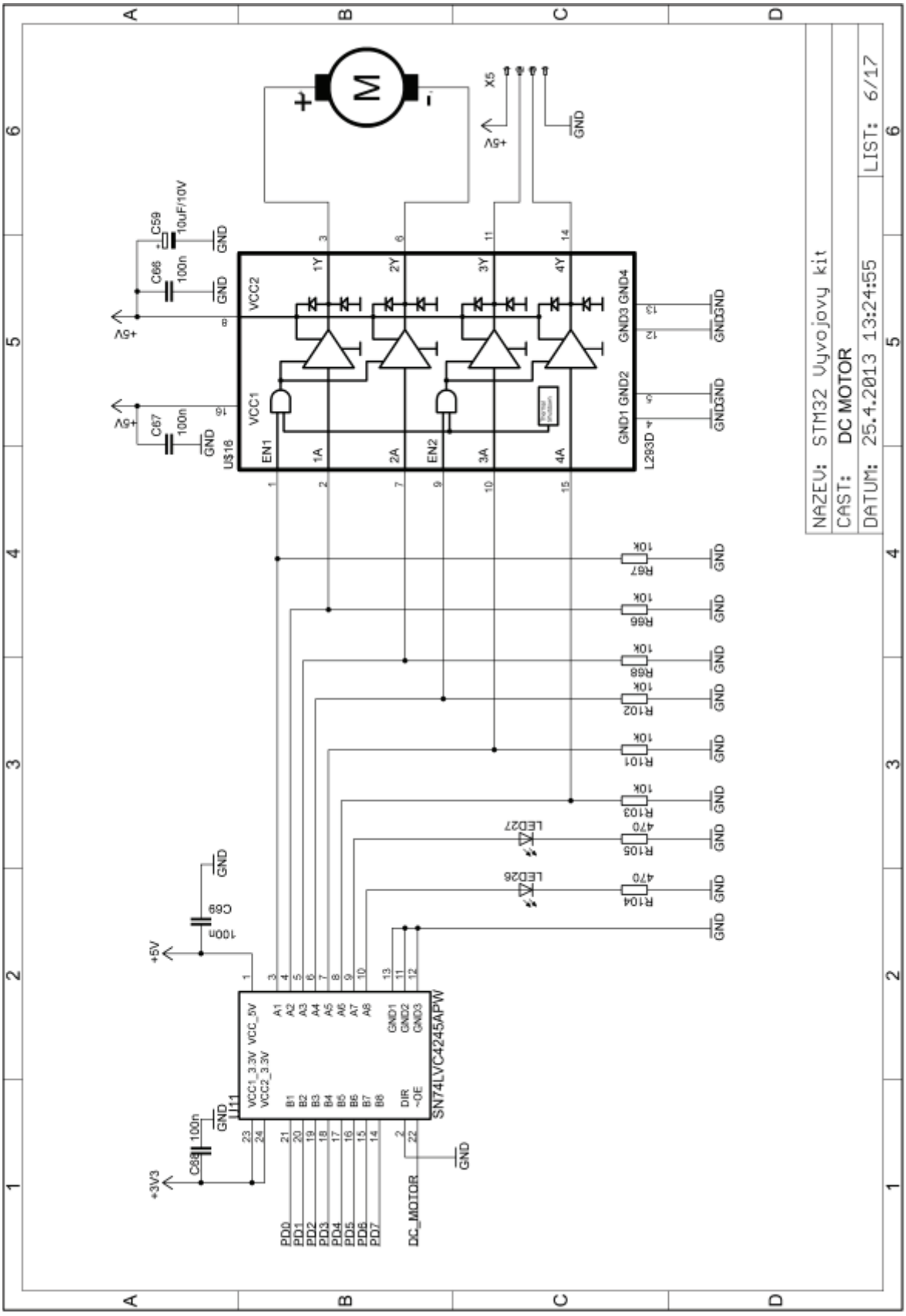
Příloha 12. Vývojový kit – Dekodér 1 z N, Tlačítla a LED



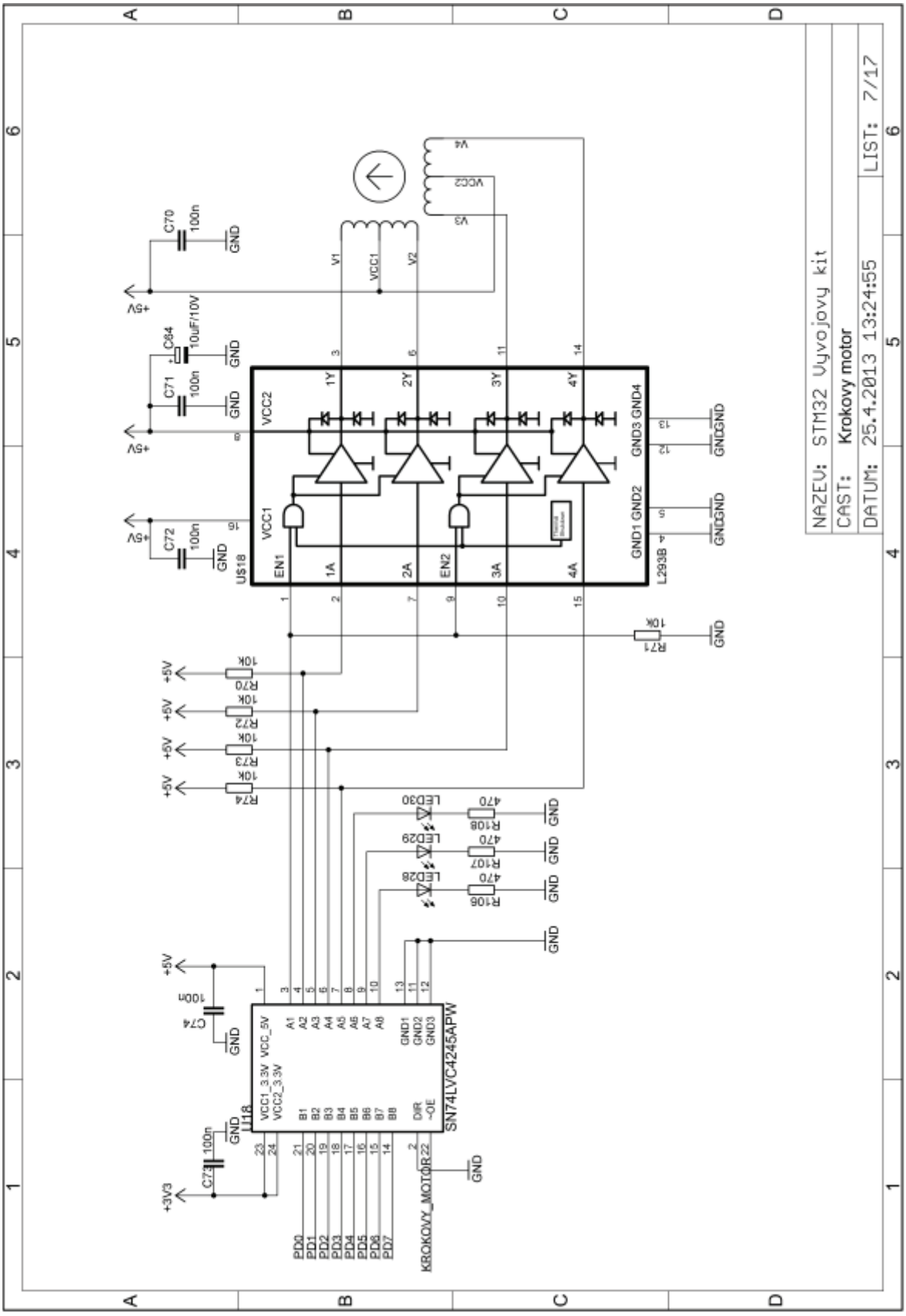
Příloha 13. Vývojový kit – Maticový displej



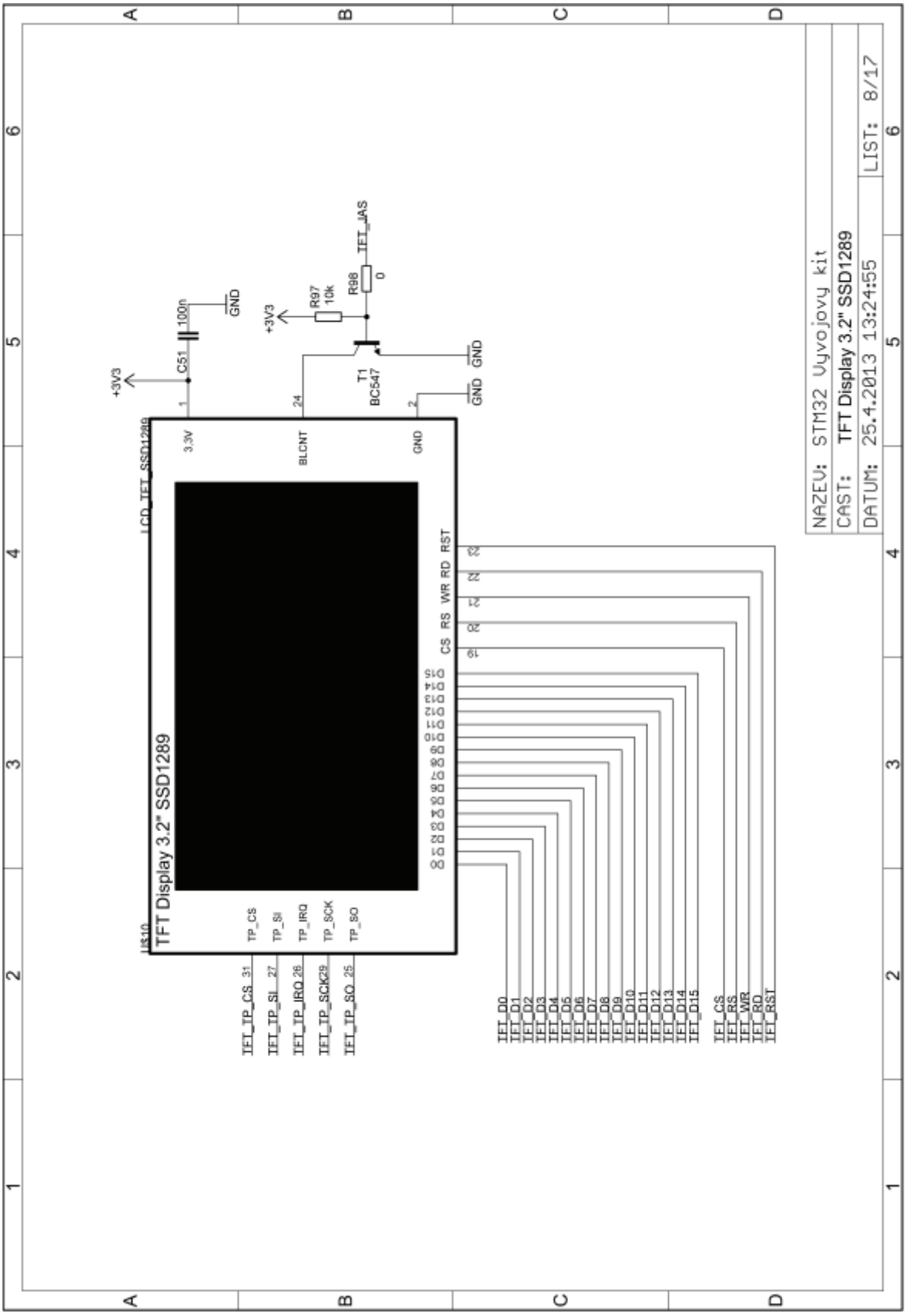
Příloha 14. Vývojový kit – Maticová klávesnice



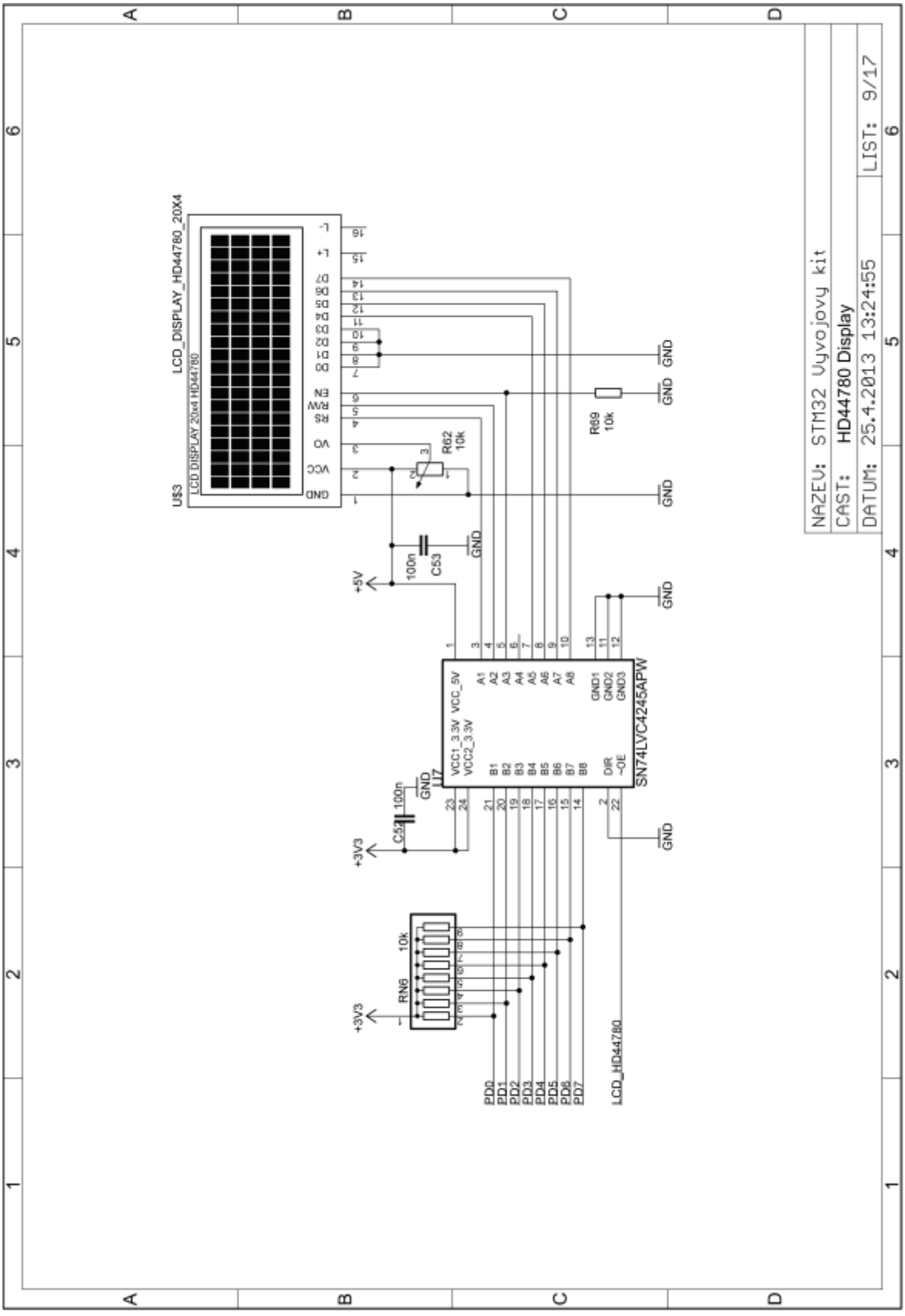
Příloha 15. Vývojový kit – Stejnosměrný motor



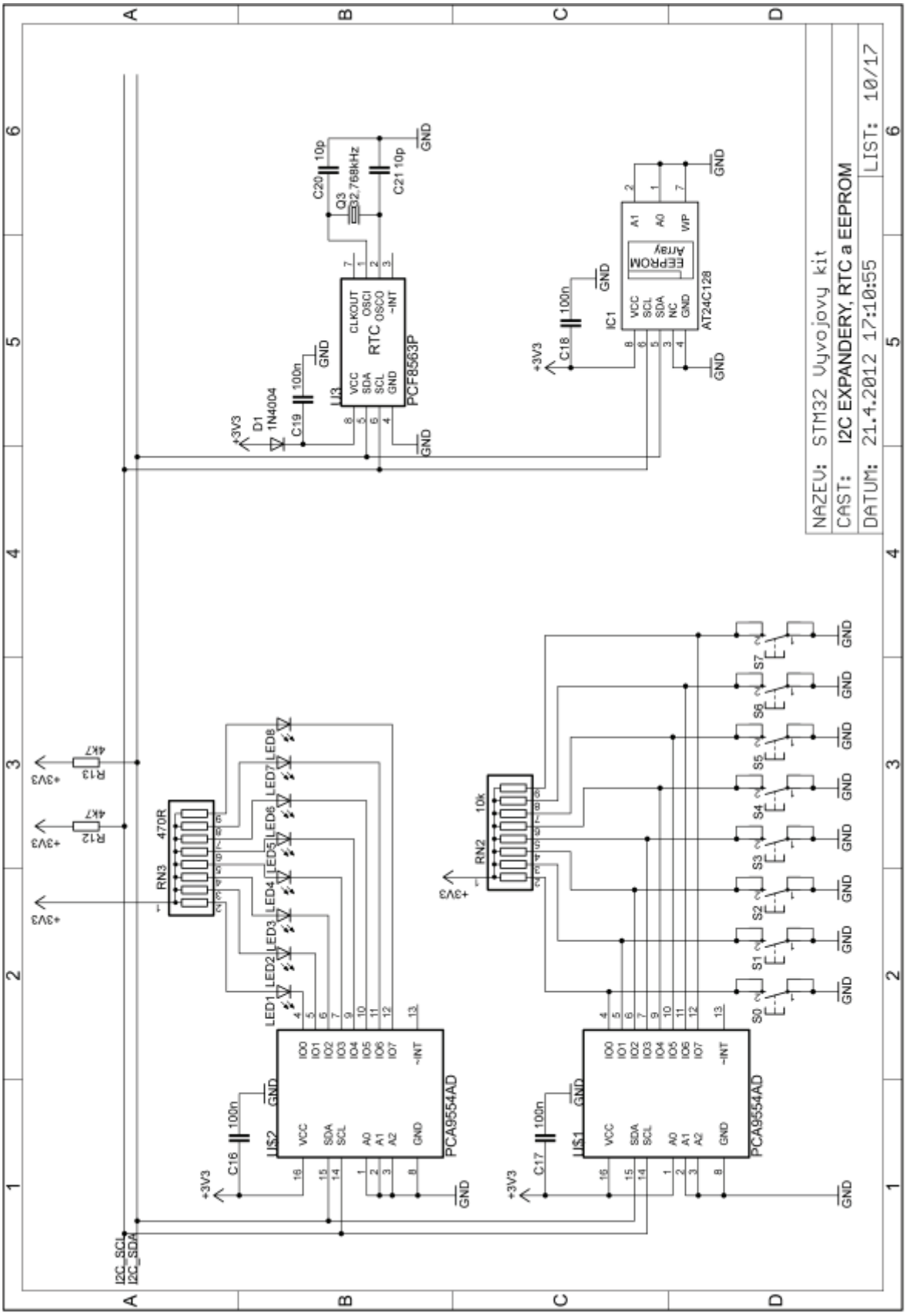
Příloha 16. Vývojový kit – Krokový motor



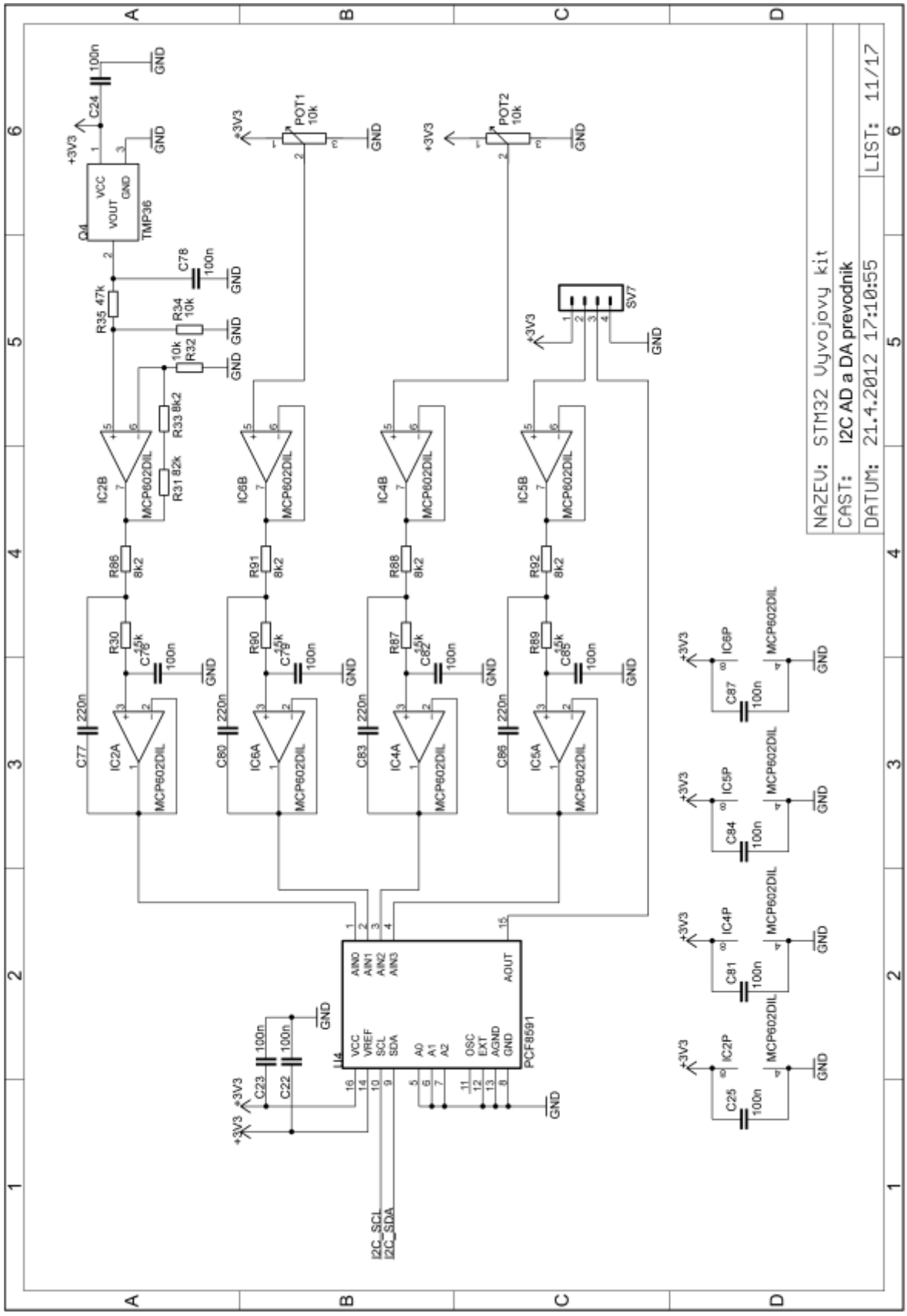
Příloha 17. Vývojový kit – TFT dotykový displej 3,2" s řadičem SSD1289



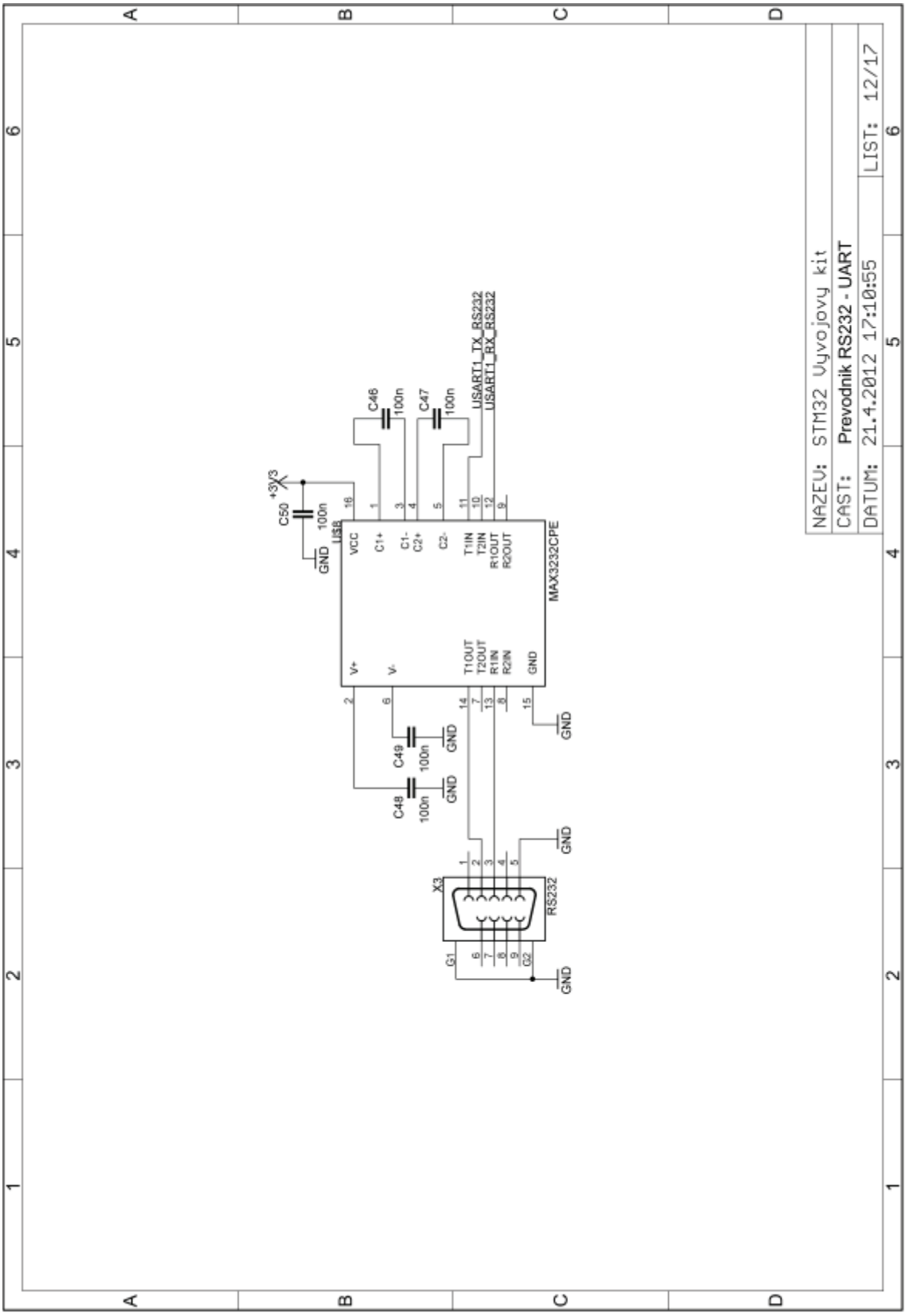
Příloha 18. Vývojový kit – Maticový displej s řadičem HD44780



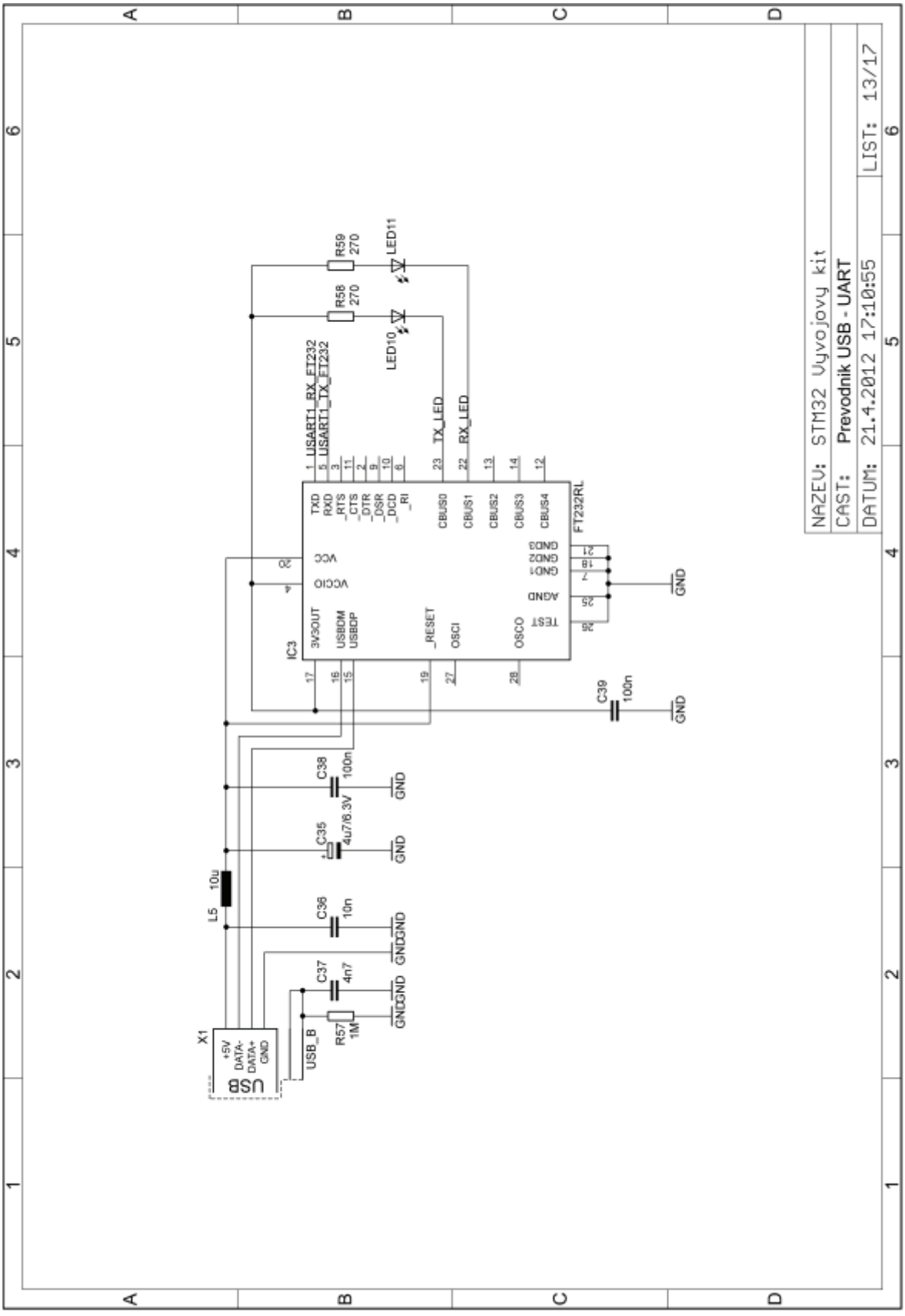
Příloha 19. Vývojový kit – I2C sběrnice (Expandéry, RTC, EEPROM)



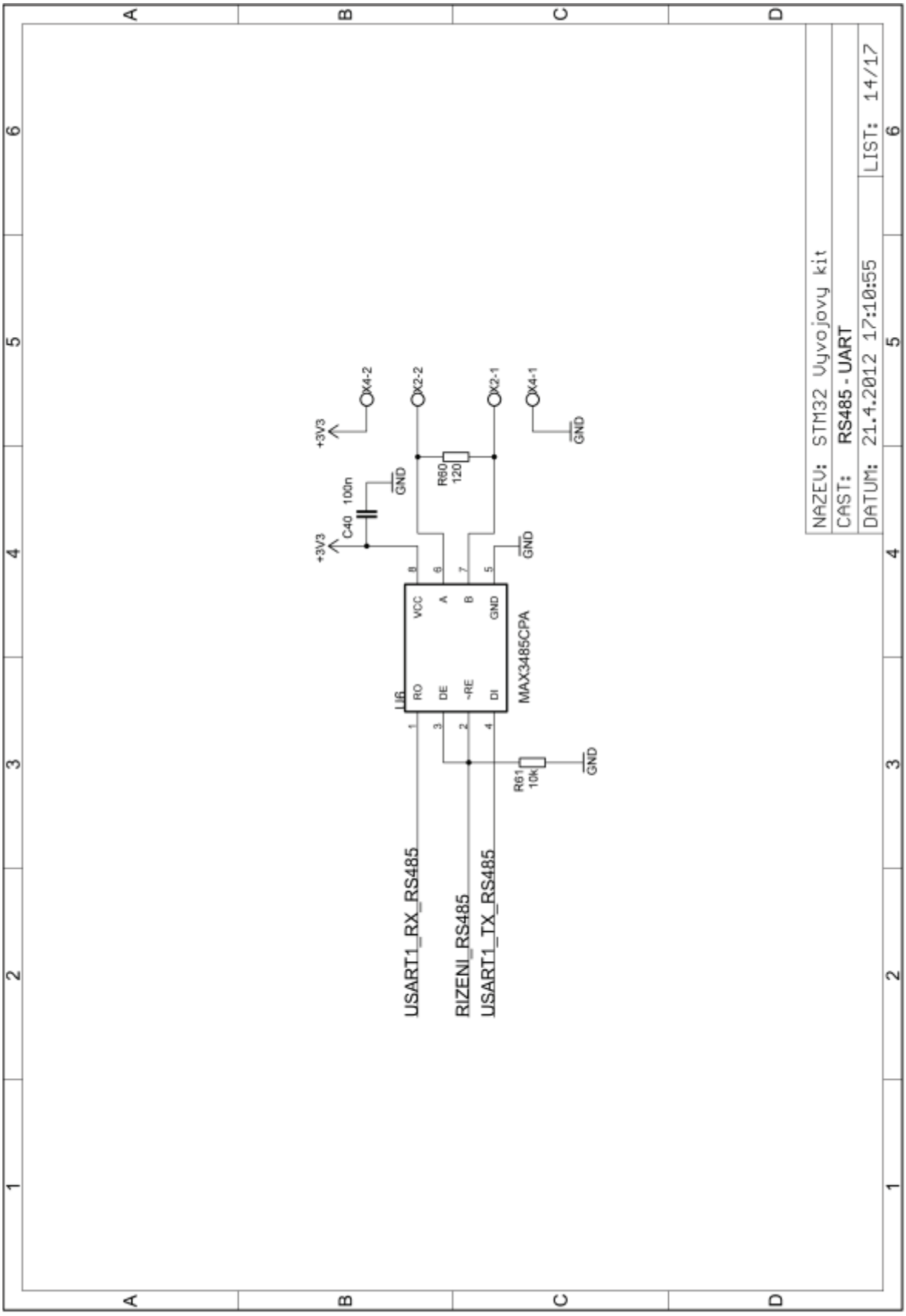
Příloha 20. Vývojový kit – I2C sběrnice (AD a DA převodník)



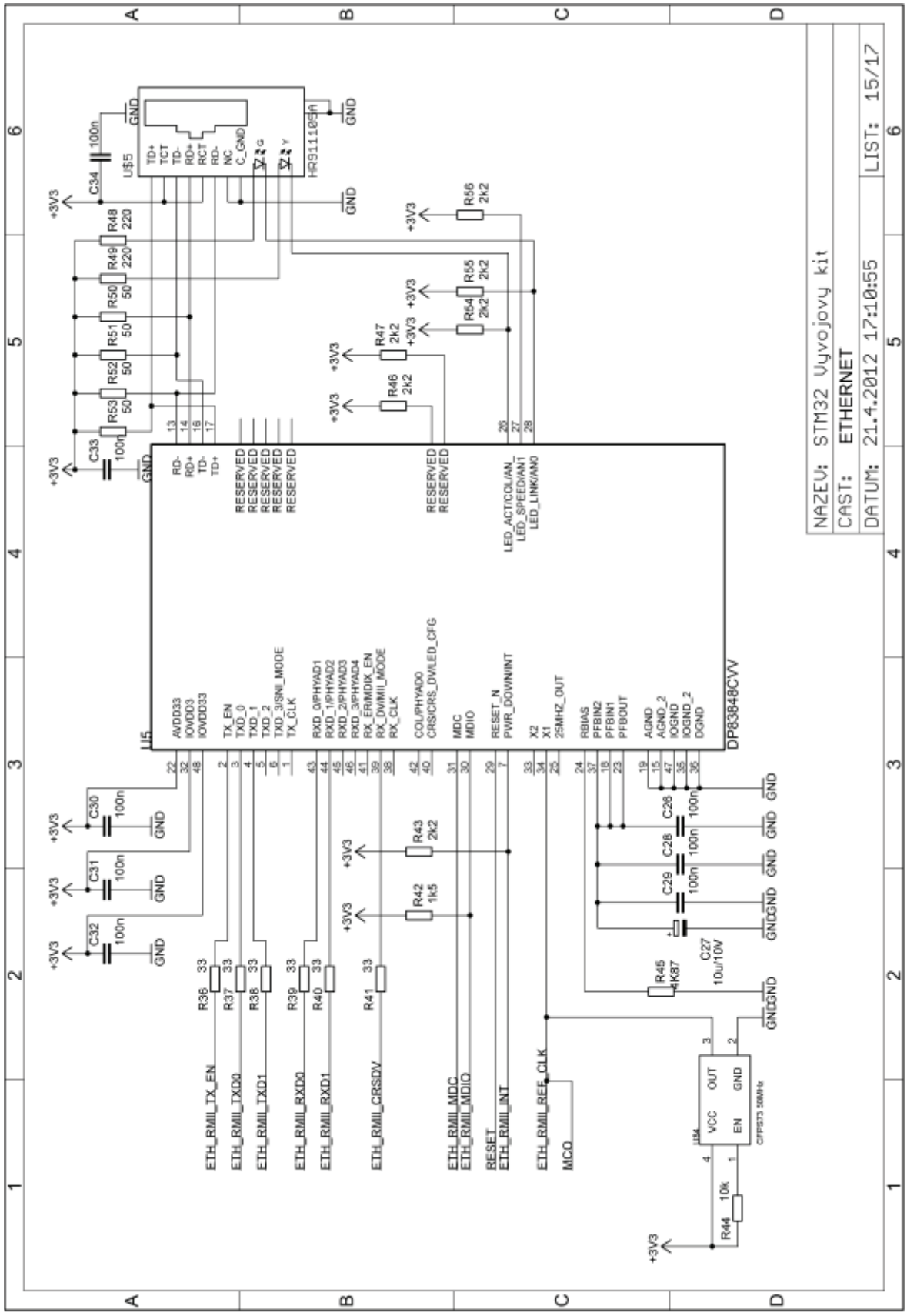
Příloha 21. Vývojový kit – RS232



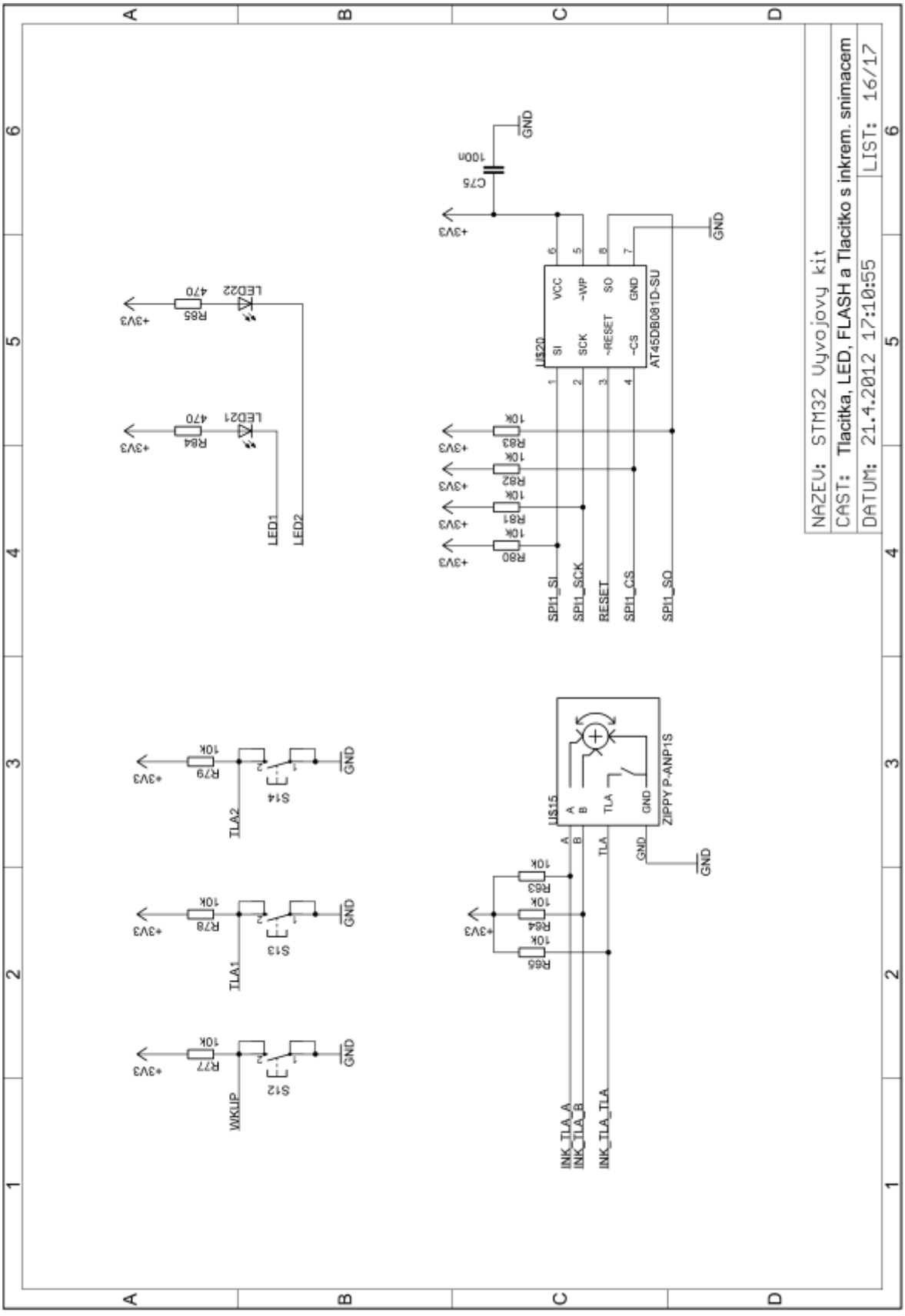
Příloha 22. Vývojový kit – USB s FT232RL převodníkem



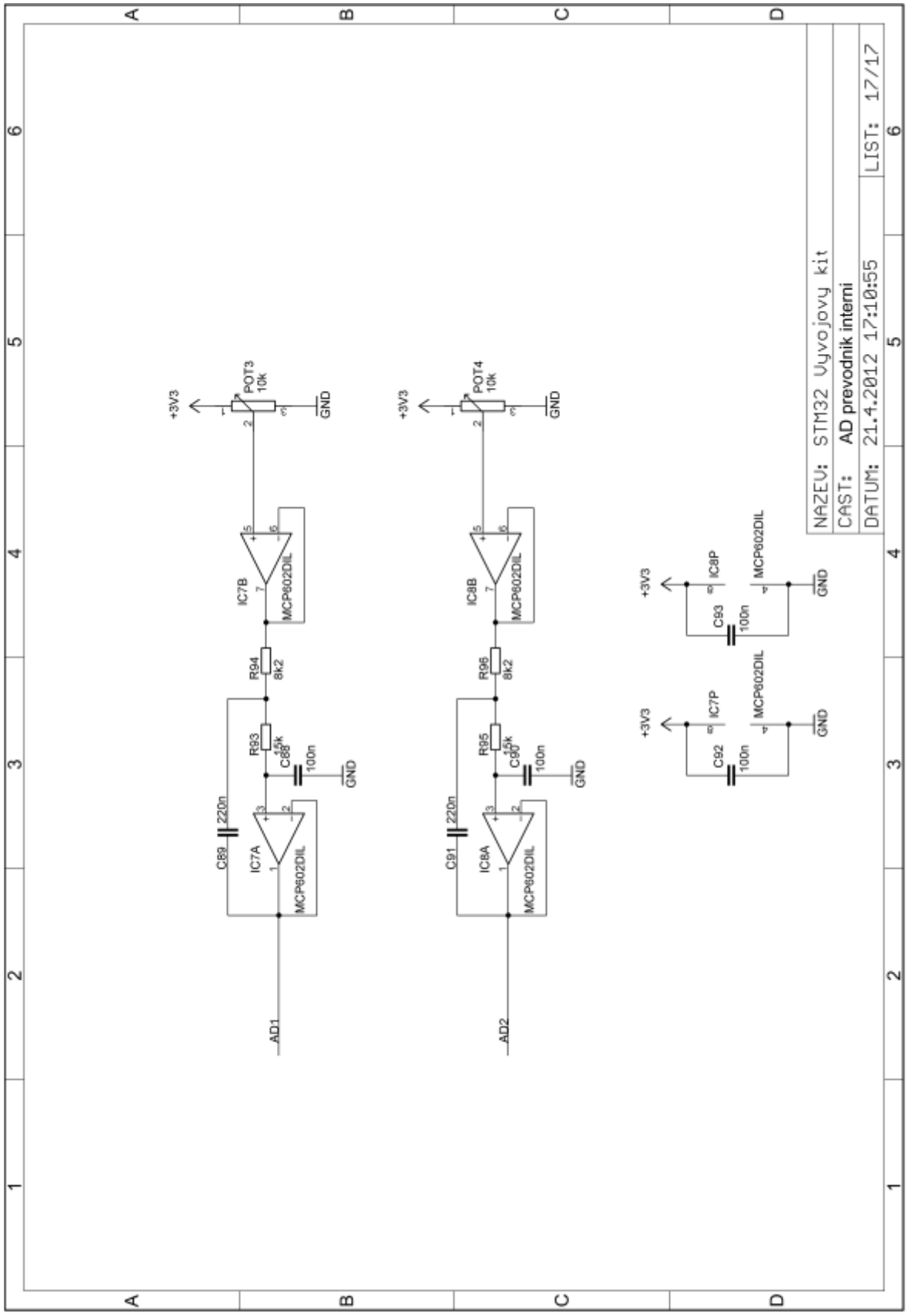
Příloha 23. Vývojový kit – RS485



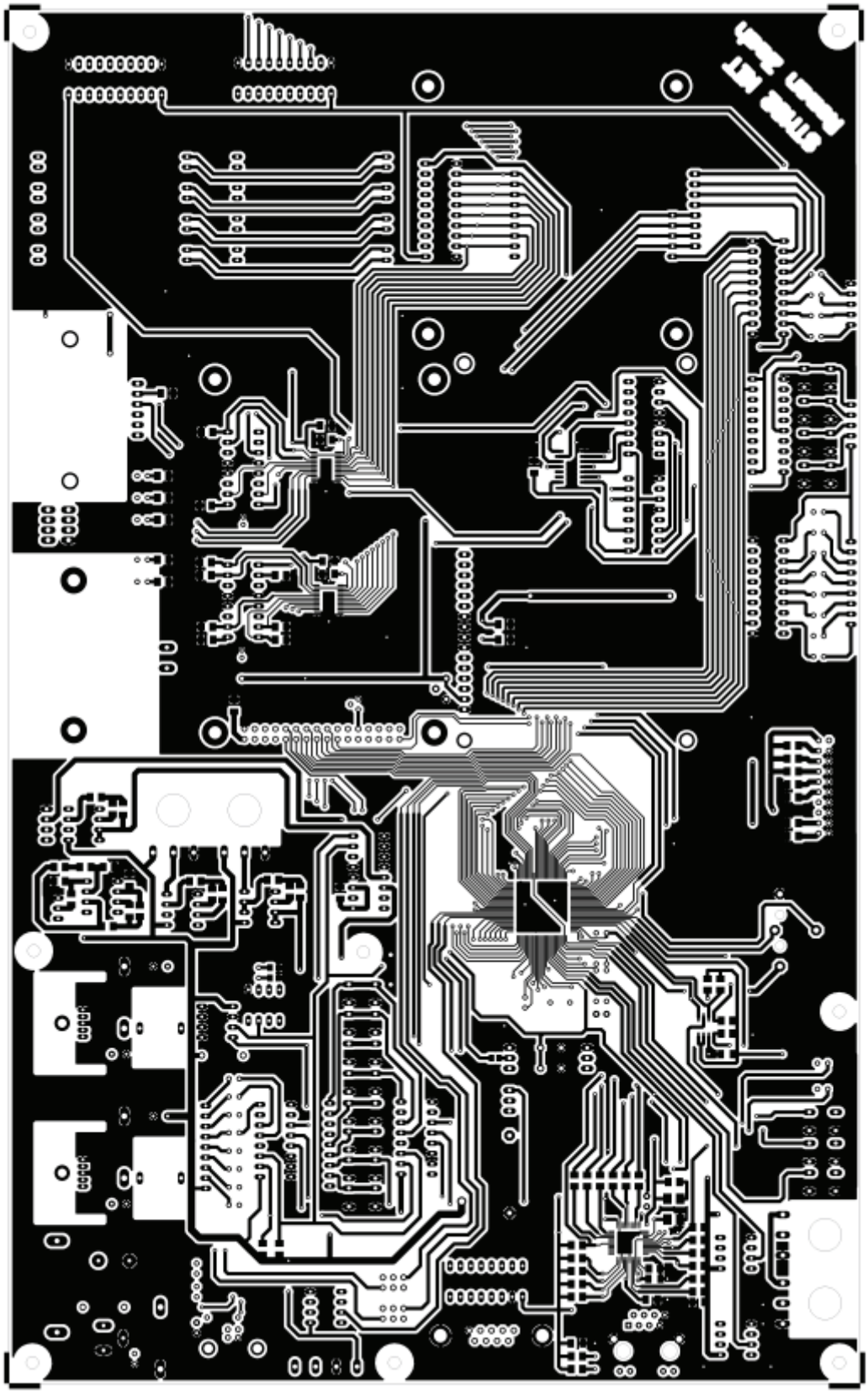
Příloha 24. Vývojový kit – Fyzická vrstva ethernetu



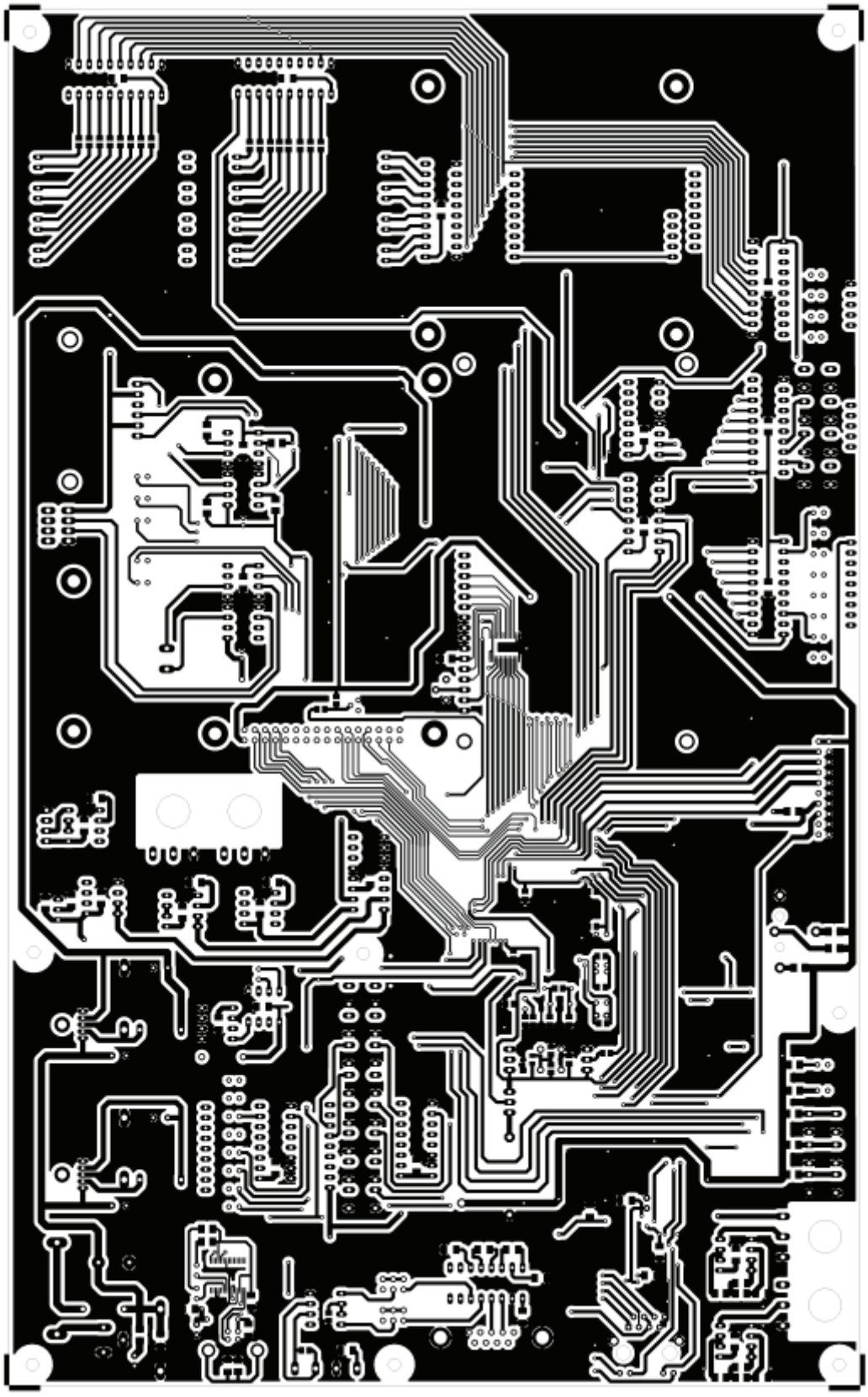
Příloha 25. Vývojový kit – Inkrementální snímač, SPI FLASH, Tlačítka a LED



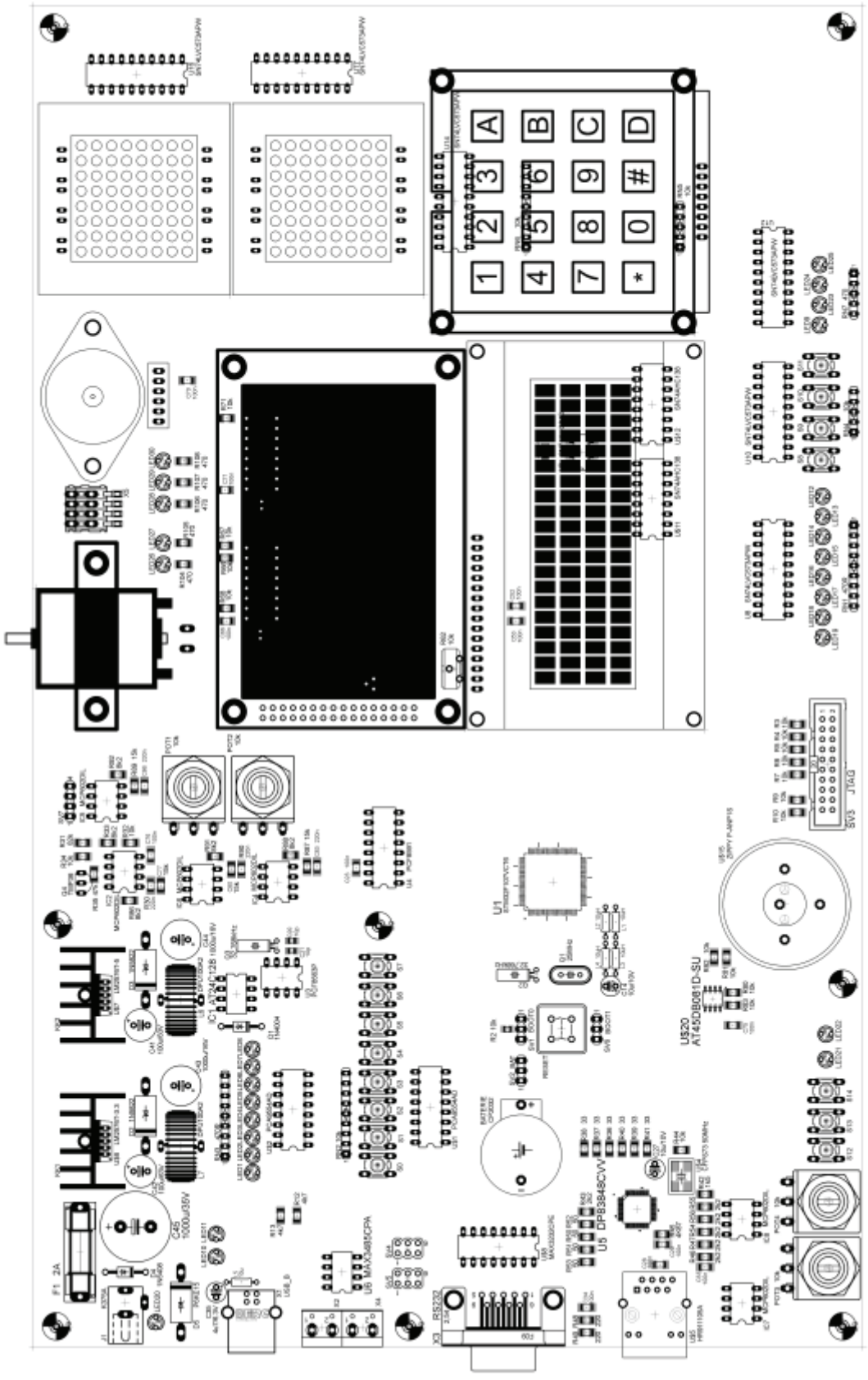
Příloha 26. Vývojový kit – Vstupní obvody interních AD převodníků mikrokontroléru STM32F107VCT6



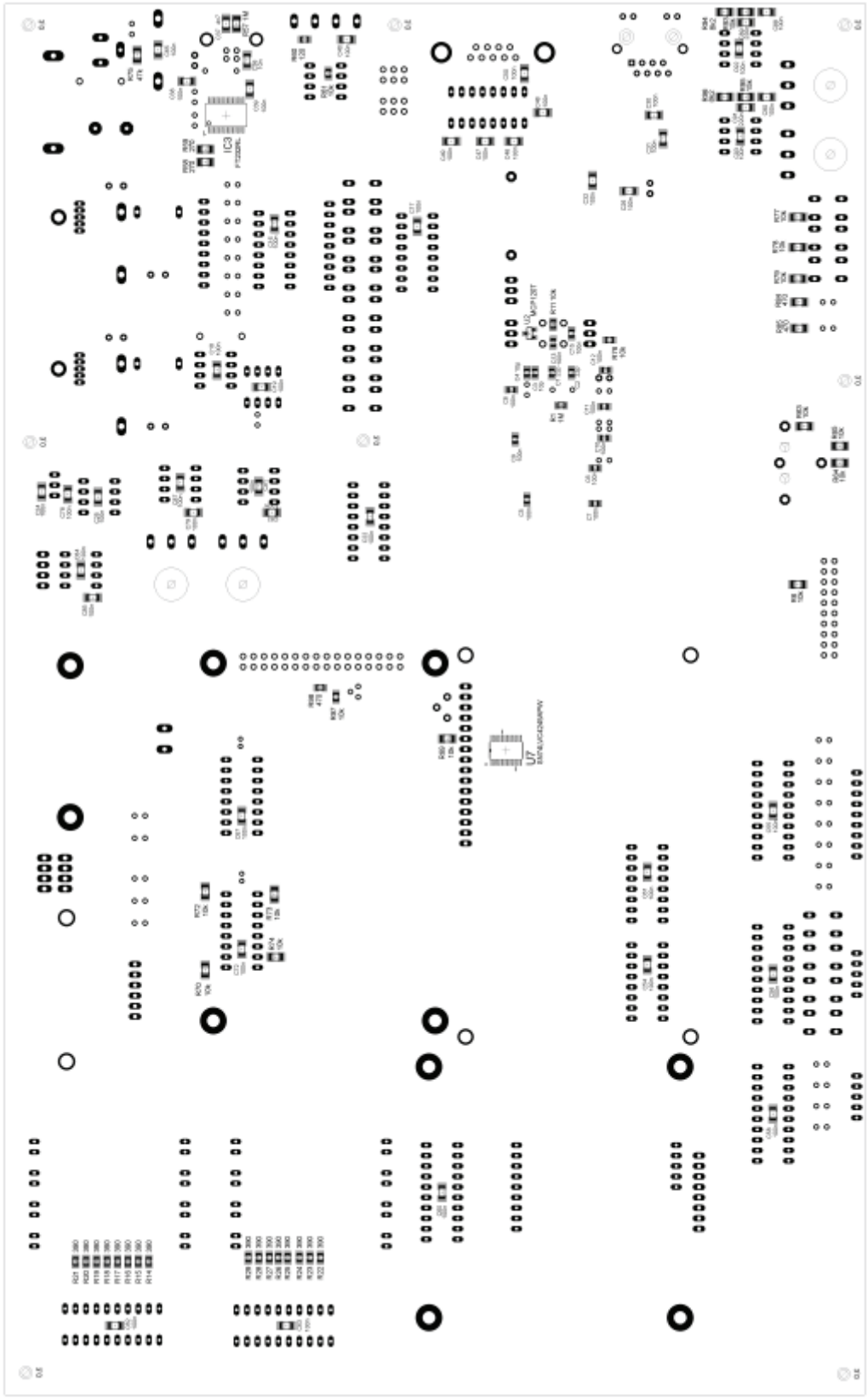
Příloha 27. Vývojový kit – DPS (1:1,5) – top



Příloha 28. Vývojový kit – DPS (1:1,5) – bottom



Příloha 29. Vývojový kit – Osazovací plán – top



Příloha 30. Vývojový kit – Osazovací plán – bottom

Součástka	Hodnota	Pouzdro
C1	22p	C0805
C2	22p	C0805
C3	10p	C0805
C4	10p	C0805
C5	100n	C0805
C6	100n	C0805
C7	100n	C0805
C8	100n	C0805
C9	100n	C0805
C10	100n	C0805
C11	100n	C0805
C12	100n	C0805
C13	100n	C0805
C14	10u/10V	E2,5-5
C15	100n	C1206
C16	100n	C1206
C17	100n	C1206
C18	100n	C1206
C19	100n	C1206
C20	10p	C0805
C21	10p	C0805
C22	100n	C1206
C23	100n	C1206
C24	100n	C1206
C25	100n	C1206
C26	100n	C1206
C27	10u/10V	E2,5-5
C28	100n	C1206
C29	100n	C1206
C30	100n	C1206
C31	100n	C1206
C32	100n	C1206
C33	100n	C1206
C34	100n	C1206
C35	4u7/6.3V	E2,5-5
C36	10n	C1206
C37	4n7	C1206
C38	100n	C1206
C39	100n	C1206
C40	100n	C1206
C41	100u/63V	E3,5-8
C42	100u/63V	E3,5-8
C43	1000u/16V	E3,5-10
C44	1000u/16V	E3,5-10
C45	1000u/35V	E7,5-18

Příloha 31. Vývojový kit - Seznam součástek 1

Součástka	Hodnota	Pouzdro
C46	100n	C1206
C47	100n	C1206
C48	100n	C1206
C49	100n	C1206
C50	100n	C1206
C51	100n	C1206
C52	100n	C1206
C53	100n	C1206
C54	100n	C1206
C55	100n	C1206
C56	100n	C1206
C57	100n	C1206
C58	100n	C1206
C59	10u/10V	E1,8-4
C60	100n	C1206
C61	100n	C1206
C62	100n	C1206
C63	100n	C1206
C64	10u/10V	E1,8-4
C65	100n	C1206
C66	100n	C1206
C67	100n	C1206
C68	100n	C1206
C69	100n	C1206
C70	100n	C1206
C71	100n	C1206
C72	100n	C1206
C73	100n	C1206
C74	100n	C1206
C75	100n	C1206
C76	100n	C1206
C77	220n	C1206
C78	100n	C1206
C79	100n	C1206
C80	220n	C1206
C81	100n	C1206
C82	100n	C1206
C83	220n	C1206
C84	100n	C1206
C85	100n	C1206
C86	220n	C1206
C87	100n	C1206
C88	100n	C1206
C89	220n	C1206
C90	100n	C1206

Příloha 32. Vývojový kit - Seznam součástek 2

Součástka	Hodnota	Pouzdro
C91	220n	C1206
C92	100n	C1206
C93	100n	C1206
D1	1N4004	0204/7
D2	1N5822	0207/10
D3	1N5822	0207/10
D4	1N5408	0204/7
D5	P6KE15	0204/7
F1	2A	SH22,5
IC1	AT24C128	DIL8
IC2	MCP602	DIL8
IC3	FT232FL	SSOP28
IC4	MCP602	DIL8
IC5	MCP602	DIL8
IC6	MCP602	DIL8
IC7	MCP602	DIL8
IC8	MCP602	DIL8
J1	K375A	DC21/55
L1	10uH	0204/7
L2	10uH	0204/7
L3	10uH	0204/7
L4	10uH	0204/7
L5	10uH	0204/7
L6	100uH	SFT830D
L7	100uH	SFT830D
LED 1-30	LED 2mA	LED 3mm
POT 1-4	10k	Potenciometr
Q1	25MHz	HC49/S
Q2-3	32,768kHz	TC38H
Q4	TMP36	TMP36
R1	1M	M0805
R2	10k	M0805
R3-R10	10k	M1206
R11	10k	M0805
R12-R13	4k7	M1206
R14-R29	390	M0805
R30	15k	M1206
R31	82k	M1206
R32	10k	M1206
R33	8k2	M1206
R34	10k	M1206
R35	47k	M1206
R36-R41	33	M1206
R42	1k5	M1206
R43	2k2	M1206

Příloha 33. Vývojový kit - Seznam součástek 3

Součástka	Hodnota	Pouzdro
R44	10k	M1206
R45	4k7	M1206
R46-R47	2k2	M1206
R48-R49	220	M1206
R50-R53	50	M1206
R54-R56	2k2	M1206
R57	1M	M1206
R58-R59	220	M1206
R60	120	M0805
R61-R62	10k	M0805
R63-R74	10k	M1206
R75	47k	M1206
R76	10k	M0805
R77-R83	10k	M1206
R84-R85	470	M1206
R86	8k2	M1206
R87	15k	M1206
R88	8k2	M1206
R89	15k	M1206
R90	15k	M1206
R91	8k2	M1206
R92	8k2	M1206
R93	15k	M1206
R94	8k2	M1206
R95	15k	M1206
R96	8k2	M1206
R97	10k	M0805
R98	470	M0805
R101-R103	10k	M1206
R104-R108	470	M1206
RESET	TLAČÍTKO	5x6mm
RN1	470	SIL9
RN2	10k	SIL9
RN3	470	SIL9
RN4	10k	SIL5
RN5	10k	SIL5
RN6	10k	SIL9
RN7	470	SIL5
S0-S14	TLAČÍTKO	5x6mm
SV1	PIN	3/1
SV2	PIN	3/1
SV3	PIN	10/2
SV4	PIN	3/2
SV5	PIN	3/2
SV6	PIN	3/1

Příloha 34. Vývojový kit - Seznam součástek 4

Součástka	Hodnota	Pouzdro
SV7	PIN	4/1
T1	BC547	TO92
U$1	PCA9554AD	DIL16
U$2	PCA9554AD	DIL16
U$3	LCD HD44780	LCD HD44780
U$4	50MHz	CFPS73
U$5	HR911105A	ETHERNET Konektor
U$6	LM2576T-3.3	TO90
U$7	LM2576T-5	TO90
U$8	MAX3232CPE	DIL16
U$9	MATICOVÁ KLÁVESNICE	MATICOVÁ KLÁVESNICE
U$10	LCD SSD1289	LCD SSD1289
U$11	SN74AHC138	DIL16
U$12	SN74AHC138	DIL16
U$13	MATICOVÝ DISPLEJ	8X8
U$14	MATICOVÝ DISPLEJ	8X8
U$15	INKREMENTÁLNÍ SNIMAC	ZIPPY
U$16	L293D	DIL16
U$17	DC MOTOR	DC MOTOR
U$18	L293B	DIL16
U$19	KROKOVÝ MOTOR	KROKOVÝ MOTOR
U$20	AT45DB081D-SU	SOIC127
U1	STM32F107VCT6	TQFP144
U2	MCP120T	SOT23
U3	PCF8563P	DIL8
U4	PCF8591	DIL16
U5	DP83848CVV	TQFP48
U6	MAX3485CPA	DIL8
U7	SN74LVC4245APW	DIL24
U8	SN74LVC573APW	DIL20
U9	SN74AHC04D	SO14
U10	SN74LVC573APW	DIL20
U11	SN74LVC4245APW	DIL24
U12	SN74LVC573APW	DIL20
U14	SN74LVC573APW	DIL20
U16	SN74LVC573APW	DIL20
U17	SN74LVC573APW	DIL20
U18	SN74LVC4245APW	DIL24
X1	USB B	USB B
X2	W237-132	W237-132
X3	RS232	RS232
X4	W237-132	W237-132
X5	W234-204	W234-204

Příloha 35. Vývojový kit - Seznam součástek 5