České vysoké učení technické v Praze
Fakulta elektrotechnická

Diplomová práce

Synchronizované obrazové snímače

Bc.Radek Řípa

Vedoucí práce: doc. Ing. Jan Fischer, CSc.

Studijní program: Kybernetika a robotika

Obor: Senzory a přístrojová technika

Leden 2012

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická
Katedra měření

Akademický rok **2011-2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	**Bc. Radek Řípa**
Program:	**Kybernetika a robotika**
Obor:	**Senzory a přístrojová technika**
Název tématu česky:	**Synchronizované obrazové snímače**
Název tématu anglicky:	**Synchronized Image Sensors**

Pokyny pro vypracování:

Navrhněte a realizujte modul obrazového snímače řízený procesorem řady ARM Cortex M3 STM32F2xx, který bude ovládán a synchronizován pomocí rozhraní Ethernet s využitím protokolu PTP IEEE1588. Vytvořte nejméně tři kusy snímačů, které budou podporovat činnost v síti, takže bude možno synchronně snímat obrazy jednoho objektu z více směrů pro účely určení jeho polohy a trajektorie ve 3D s možností verifikace. Navrhněte a realizujte příslušné obvody a vytvořte potřebné programové vybavení pro procesor STM32F207 i programy pro nadřazené PC.

Experimentálně ověřte správnost synchronizace všech obrazových snímačů při snímání velmi rychle proměnné scény.

Seznam odborné literatury:

[1] Yiu, J.: The definitive Guide to the ARM Cortex- M3. Elsevier, 2007

[2] YSTMicroelectronics: RM0033 Reference manual

[3] Gasvik, K. J.: Optical metrology, 3-th edition, Wiley, 2002

Vedoucí diplomové práce:	doc. Ing. Jan Fischer, CSc.
Datum zadání diplomové práce:	1. prosince 2011
Platnost zadání do[1]:	7. června 2013

Prof. Ing. Vladimír Haasz, CSc.
vedoucí katedry

Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 1. 12. 2011

[1] Platnost zadání je omezena na dobu tří následujících semestrů.

Poděkování

Na tomto místě bych rád poděkoval svým rodičům, přátelům kteří mě podporovali po celou dobu realizace této práce.

Dále bych rád poděkoval doc. Ing. Janu Fischerovi, CSc. za vedení diplomové práce, za jeho cenné rady, podklady a připomínky, a především za trpělivost v průběhu řešení.

Čestné prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 3.1.2013 .....

Abstract

This diploma thesis describes the complete design of synchronous image sensor with Ethernet interface. The system is based on 32-bit processor STM32F207. This work describes a hardware and software image sensors and the creation of software for superior PC.

Abstrakt

Tato diplomová práce popisuje kompletní návrh synchronních obrazových snímačů s rozhraním Ethernet,. Systém je založen na 32 bitovém procesoru STM32F207. Práce popisuje vytvoření hardware a software pro jednoty a vytvoření software pro nadřazený PC.

Obsah

1 Úvod **1**

2 Rozbor úlohy **2**

3 Micron/Aptina CMOS **5**

3.1 Parametry a vlastnosti 5

3.2 Přístup k senzoru a jeho ovládání 6

3.2.1 Formát výstupních dat 7

4 Popis periferií, využitých v mikrokontroleru STM32 **9**

4.1 Periferie 9

4.1.1 RCC - Periferie pro nastavení hodinových signálů mikrokontroleru 10

4.1.2 General-purpose I/Os - nastavení vstupně výstupních bran mikrokontroleru 10

4.1.3 DCMI - Digital camera interface 11

4.1.4 DMA - Periferie umožňující nastavit přímý přístup do paměti bez účasti procesoru 11

4.1.5 Ethernet - periferie pro komunikaci s řadičem fyzické vrstvy Ethernetu . . 12

4.1.6 Universal seria bus - Periferie obsluhující univerzální sériové rozhraní . . . 12

4.2 Periferie Inter-Integrated Circuit 12

4.3 Obvody čítačů a časovačů 12

4.4 Organizace paměti mikrokontroleru STM32 a její využití 12

4.4.1 Paměť STM32F207 13

4.5 Periferie 13

4.5.1 Paměť STM32F407 13

4.5.2 Flash Paměť 14

5 Použití lwIP stacku na mikrokontroleru STM32 **15**

5.1 Implementace lwIP na STM32 . . . 15

5.2 Organizace paměti stacku . . . 15

5.3 Nedostatky stacku . . . 16

6 Precision time protokol podle normy IEEE1588 **17**

6.1 Popis práce PTP . . . 17

6.2 Použití na STM32F . . . 18

6.3 Odesílání dat a optimalizace rychlosti pro PTP . . . 20

6.4 Využití PTP pro přesné měření . . . 20

7 Použití MAC na STM32 pro komunikaci se sítí Ethernet **23**

7.1 Descriptory . . . 24

7.1.1 Rozšíření deskriptorů pro PTP . . . 25

7.2 Optimalizace deskriptorů . . . 26

7.2.1 Zmenšení deskriptorů . . . 26

7.2.2 Přenastavování parametrů deskriptorů pro odesílání obrazových dat . . . 28

7.3 Odesílání dat pomoci deskriptoru bez lwIP stacku . . . 29

7.3.1 Návrh odesílacího algoritmu . . . 29

8 Komunikace s okolím **31**

8.1 Navrhy komunikace . . . 31

8.2 Komunikační protokol pro přenos obrazových dat . . . 32

9 Souběh jednotlivých částí programu **33**

9.1 Načítání dat z obrazového senzoru . . . 33

9.2 Odesílání bloků dat přes Ethernet . . . 34

9.3 Periodic handle . . . 34

9.4 Obsluhy přerušení a jejich priority . . . 34

10 Realizace **35**

10.1 Použití vývojové desky . . . 35

10.1.1 změna rozhraní MII-RMII . . . 35

10.2 Návrh procesorové desky plošného spoje pro kameru . . . 36

10.2.1 Rozbor obsahu desky plošného spoje . . . 36

10.2.2 Vytvoření desky plošného spoje . . . 42

10.2.3 Kompletní návrh desky 42

11 Vytvoření ovládacích prostředků pro PC **45**

11.1 Ovladač pro komunikaci s kamerovou jednotkou 45

11.2 Grafické uživatelské rozhraní 45

12 Testování synchronnosti snímačů a rychlosti vyčítání dat **48**

13 Závěr **50**

A Desky plošných spojů **54**

A.1 Deska plošného spoje V1/2 54

A.1.1 Schéma zapojení 54

A.1.2 Desky plošných spojů 59

A.1.3 Osazovací plán 62

A.1.4 Rozpiska součástek 65

A.2 Deska plošného spoje V3 66

A.2.1 Schéma zapojení 66

A.2.2 Desky plošných spojů 71

A.2.3 Osazovací plán 74

A.2.4 Rozpiska součástek 77

B Obsah CD **79**

Seznam obrázků

2.1 Principiální rozvržení synchronizované jednotky s obrazovým senzorem 2

2.2 Principiální rozvržení synchronizované jednotky s obrazovým senzorem 3

3.1 Signály, které vyuřívá CMOS obrazový senzor MICRON 6

3.2 Sekvence signálů pro čtení dat ze senzoru pomocí I2C rozhraní 7

3.3 Sekvence signálů pro zápis dat ze senzoru pomocí I2C rozhraní 7

3.4 Výstupní signály z CMOS obrazového senzoru 8

4.1 Periferie mikrokontroleru STM32F2/4 používané v projektu 9

4.2 Zapojení hodinového obvodu STM32F2/4 a jeho dopad na frekvenci jádra 10

4.3 Organizace paměti RAM mikrokontroleru STM32F2/4 13

6.1 Synchronizace času mezi master jednotkou a slave jednotkou 18

7.1 Naznačení organizace deskriptorů v řetězové struktuře a alokace paměti pro data deskriptoru 25

7.2 Porovnání rozšířeného deskriptoru na STM32F2/4 a klasického na STM32F107 . 26

7.3 Porovnání práce deskriptorů s daty menšími než 256B a s daty přesahujícími tuto velikost 27

7.4 Kopírování dat z pbuf struktur lwIP stacku do deskriptorů patřících rozhraní Ethernet 28

7.5 Principilní uspořádání lwIP stacku a využití pseudo deskriptorů pro posílání dat 29

7.6 Postup odesílání dat při použití pseudo deskriptorů 30

8.1 Typy připojen ke kameře 31

8.2 Typy zpráv podporované kamerou 32

8.3 Typy zpráv podporované kamerou 32

9.1 Znázornění principiálního řešení přenosů dat z DCMI do paměti RAM a dále do pseudo-deskriptorů které využije Ethernet pro odeslání dat 33

10.1 Fyzická úprava vývojové desky s rozhraním MII na rozhraní RMII 36
10.2 Zapojení vývojové desky s rozhraním RMII s obrazovým senzorem CMOS pro testování funkčnosti zapojení 37
10.3 Blokové schéma práce demonstračního programu pro ověření funkčnosti připojení senzoru CMOS 38
10.4 Obslužný program pro demonstraci funkčního připojení CMOS obrazového senzoru 39
10.5 Připojení konektoru pro CMOS obrazový snímač DCMI rozhraní mikrokontroleru STM32F2/4 40
10.6 Připojení konektoru pro SWD rozhraní k mikrokontroleru STM32F2/4 40
10.7 Připojení konektoru mini USB k mikrokontroleru STM32F2/4 41
10.8 Propojka pro aktivaci bootloader funkcionality mikrokontroleru 41
10.9 Připojení adapteru pro mikroSD kartu k STM32F2/4 42
10.10Konektor pro připojení rozhraní SPI a UART k STM32F2/4 43
10.11Připojení indikačních LED k mikrokontroleru STM32F2/4 43
10.12Obrys pro desku plošného spoje odpovídající vnitřním rozměrům krabičky KP59 44
10.13Kompletní osazená deska s mikrokontrolerem STM32F207 44

11.1 Jednoduchý program pro vyčítání obrazových dat z kamery 46
11.2 Grafické uživatelské rozhraní pro test kamer 46
11.3 Naznačení způsobu komunikace ovladače s jednotlivými kamerami 47

12.1 Uspořádání měření synchronnosti snímačů 48
12.2 Naznačení způsobu komunikace ovladače s jednotlivými kamerami 49
12.3 Naznačení způsobu komunikace ovladače s jednotlivými kamerami 49

Kapitola 1

Úvod

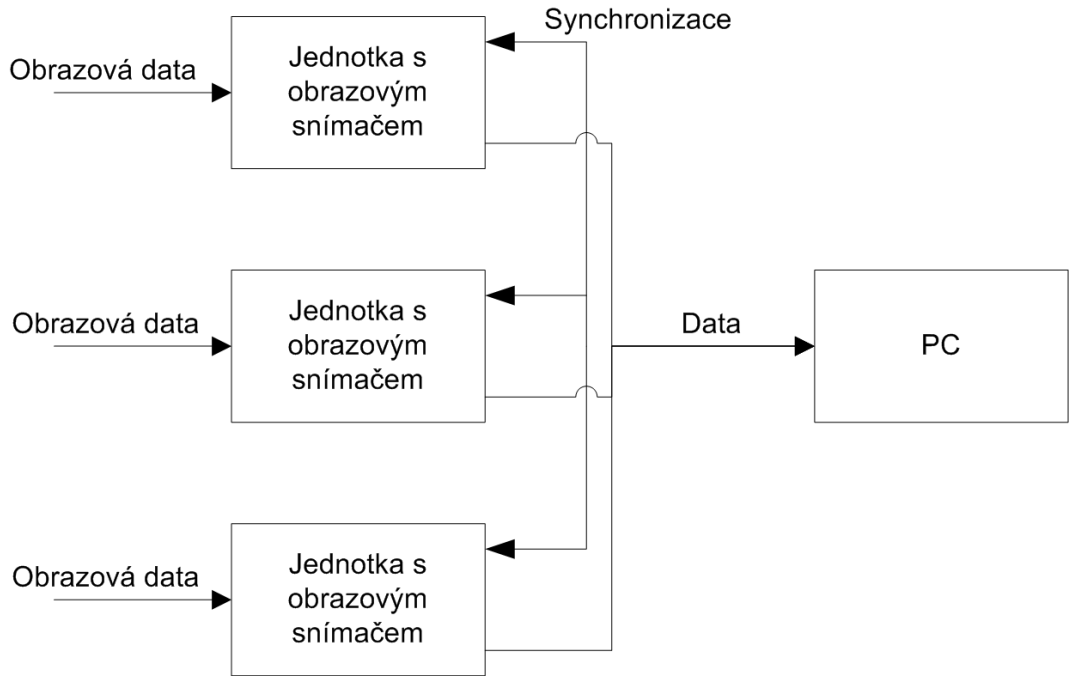
V moderní měřicí technice se klade stále větší požadavek na možnost optického měření veličin. Ať už jde o délku, plochu či rozpoznání scény a zároveň možnost měřit tyto veličiny na více nezávislých jednotkách ve stejný čas, nebo z nasnímaného 2D obrazu vytvořit 3D.

Cílem této práce je vytvoření takovéto senzorové sítě, tvořené jednotlivými jednotkami. Každá jednotka bude schopna snímat 2D obraz a zároveň bude mezi jednotkami provedena synchronizace zaručující, že jednotlivé snímky budou provedeny ve stejný okamžik. Nejprve bude zapotřebí vytvořit vlastní jednotky, a to jak z hardwarového tak i ze softwarového hlediska. A vytvořit obslužné programové vybavení pro získávání informací z jednotlivých jednotek.

Kapitola 2

Rozbor úlohy

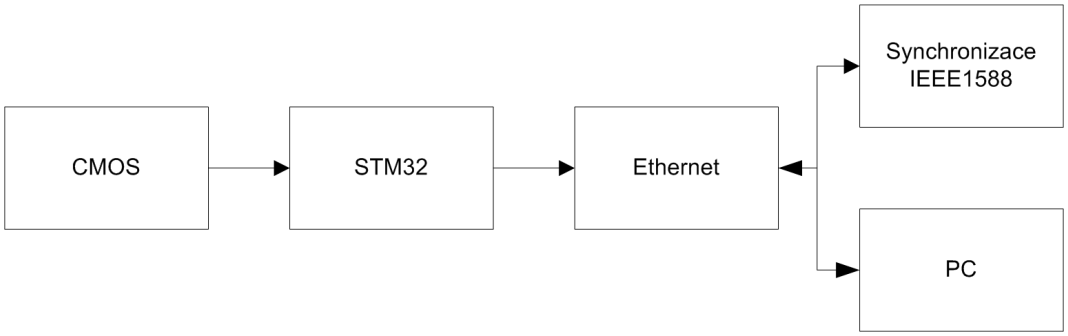
Cílem práce je vytvořit systém skládající se z několika jednotek schopných snímání obrazové informace v časově synchronních okamžicích, obr. 2.1. Každá jednotka tedy bude muset být



Obr. 2.1: Principiální rozvržení synchronizované jednotky s obrazovým senzorem

schopna zachytit obrazovou informaci, tuto informaci musí umět odeslat nadřazenému systému, popřípadě ji zpracovat a poslat informace o výsledku nadřazenému systému. Dále musí být jednotka schopna synchronizace mezi ostatními jednotkami.

Na obr. 2.2 je zobrazeno principiální rozvržení jednotky. Obrazová data budou načítána z obrazového snímače CMOS. Základním prvkem celého systému je mikrokontroler STM32, který bude řídit celou funkci jednotky. Pro komunikaci s okolím a synchronizaci zařízení bude použito



Obr. 2.2: Principiální rozvržení synchronizované jednotky s obrazovým senzorem

rozhraní Ethernet. Nakonec jako nadřazený systém je použito PC, které bude přijímat obrazová data a komunikovat s jednotkami. Diplomovou práci lze tedy rozdělit na jednotlivé části řešící výše popsanou problematiku.

Nejprve je třeba připojit CMOS obrazový senzor k mikrokontroleru STM32 a vytvořit jednotlivé ovládací rutiny pro jeho ovládání a načítání dat ze senzoru do mikrokontroleru. Diplomovou práci lze rozdělit na jednotlivé dílčí části. Na snímání obrazu provedené pomocí CMOS obrazového senzoru Micron. Dále na zpracování obrazu pomocí mikrokontroleru STM32F, komunikaci s okolím pomocí Ethernetu a IP stacku v STM32F. Další část se zabývá synchronizací času mezi jednotkami pomocí PTPd stacku pracujícím na STM32F a softwarovým vybavením pro komunikaci mezi jednotkami a PC. Dále se zabývá návrhem hardwarových komponentů pro senzory.

Pro snímání je použit snímač Micron MT9V032 nebo jeho novější verze od firmy Aptina MT9V034 (jedná se o podobný typ snímače, pouze byl změněn majitel společnosti). Výhodou použití CMOS snímače tohoto typu je možnost využití Global Shutter funkce, která zajistí uložení obrazové informace do snímače v jeden okamžik, na rozdíl od snímačů typu Rolling Shutter, kdy je snímek ukládán po částech. V případě rychle se měnící scény by tento způsob mohl působit viditelné problémy na měřených datech(např. kácení pohybujících se objektů, způsobené rozdílným časem expozice částí scény). Naměřená data jsou ze snímače přenesena po 8bitové paralelní sběrnici ze synchronizací řádků a celého obrazu. Zde je možné s výhodou použít periferii DCMI mikrokontroleru STM32F která je uzpůsobená ke zpracování dat v tomto formátu.

Další část se zabývá zpracováním přejímaných dat ze senzoru a jejich přenosem do ovládacího PC nebo jejich uložením v do paměti. Další částí je řešení komunikace pomocí lwIP stacku a jeho integrace do mikrokontroleru a hledání nejvhodnějšího způsobu jak přenášet velké množství

dat po Ethernetu s co nejmenším vytížením mikrokontroleru.

Poslední část zabývající se softwarem pro STM32F je využití PTPd pro synchronizaci snímání obrazových snímačů a optimalizace stacku pro periferie, jež umožní menší vytížení mikrokontroleru. Pro komunikaci s PC je navržen ovládací program v C, a jednání nejoptimálnější verze komunikace opět s ohledem na možnost rychlého přenosu velkého množství dat. Program je členěn do ovladače, který umožňuje ovládat kamerovou jednotku a dále zajistí jednoduchou integraci k ovládacímu grafickému programu. Druhou částí je vytvoření jednoduchého ovládacího programu, který bude demonstrovat funkci celého systému.

Kapitola 3

Micron/Aptina CMOS

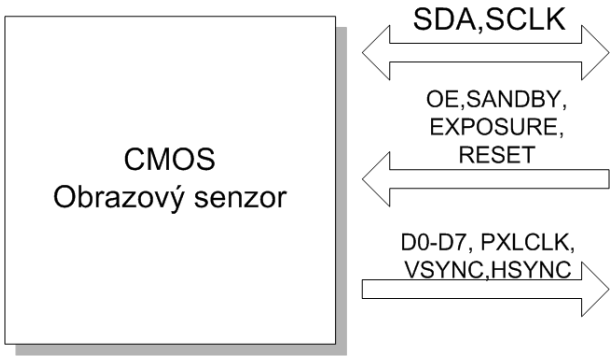
Pro snímání obrazu bude použit CMOS obrazový senzor od firmy Micron, dnes Aptina [14, 15].

3.1 Parametry a vlastnosti

Jedná se o senzor typu Global Shutter. Tento senzor tedy dokáže uložit obraz v jednu konkrétní chvíli. Tím je velice výhodné jeho použití pro synchronní snímání obrazu. Maximální rozlišení senzoru je až 752x480 bodů černobíle. Senzor umožňuje až 10bitovou hloubku obrazu, v našem případě bude použita pouze 8bitová hloubka obrazu, jelikož při použití 10bitové hloubky by se zdvojnásobila paměťová náročnost pro uložení obrazové informace do paměti mikrokontroleru.

Může nastat situace, že není zapotřebí využít plné rozlišení snímače. Senzor proto umožňuje využít dvě funkce pro snížení rozlišení. První funkcí je pixel binning, což je sčítání obrazové informace z vedlejších obrazových elementů senzoru. Je tedy možné sečíst informaci ze dvou či čtyř okolních pixelů a to jak horizontálně tak vertikálně. Výsledkem je tedy snížení rozlišení senzoru ve vertikálním či horizontálním směru 2x až 4x. Se snížením rozlišení zde dojde ovšem ke snížení rychlosti vysílání obrazových dat 2x až 4x.

Druhou možností jak snížit rozlišení snímače je použít funkci region of interest(zkráceně ROI). Tato funkce umožňuje vybrat si pixel v obrazovém poli snímače, který bude určovat levý horní roh nového obrazu a výšku a šířku, udávající nové rozlišení snímače. Na rozdíl od pixel binningu tato funkce tedy dovoluje nastavit rozlišení libovolně a ne pouze v násobcích původní hodnoty.



Obr. 3.1: Signály, které využívá CMOS obrazový senzor MICRON

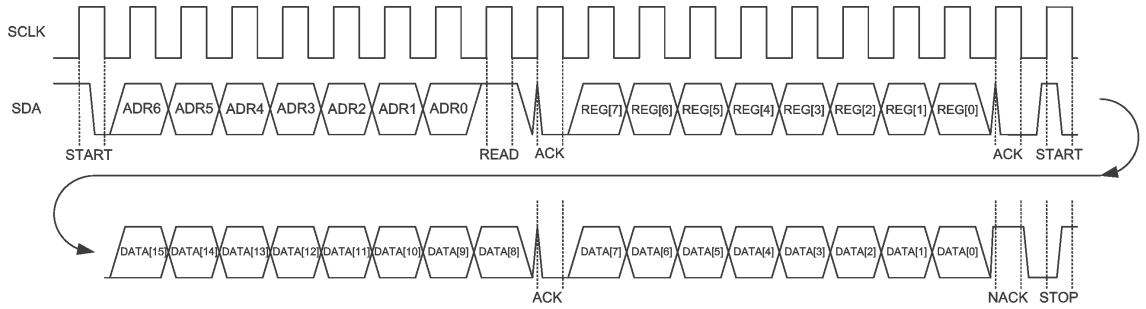
3.2 Přístup k senzoru a jeho ovládání

Pro ovládání senzoru je využito rozhraní I2C, pomocí kterého je možné konfigurovat nastavení senzoru. Jedná se o dvouvodičové rozhraní, první vodič slouží pro přenos dat druhý slouží pro přenosy hodinového signálu.Formát zpráv pro komunikaci se senzorem po I2C je následující. Nejprve je odesláno 8bitů, z nichž je prvních 7 adresních, určujících adresu zařízení(adresa zařízení je dána HW konfigurací).Poslední bit určuje, zda bude proveden zápis do registrů obrazového snímače, nebo čtení. Následuje 8bitů udávající číslo registru, který se bude číst nebo zapisovat a nakonec následuje 16bitů s přečtenými daty či s daty k zapsání.

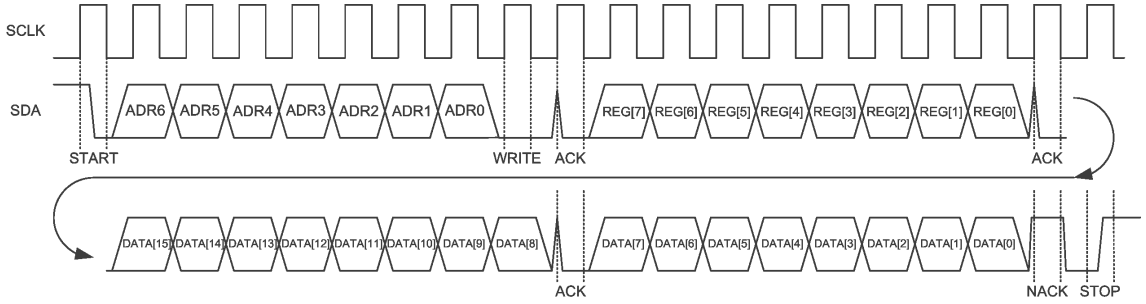
Průběh dat při čtení dat ze senzoru je na obr. 3.2 a zápis na obr. 3.3. Průběh je ilustrativní pro kontrolu průběhu v případě chyby zapojení. Při opomenutí připojení pull-up odporů nebude nezapojená sběrnice v logické jedničce. V případě špatného nastavení připojené jednotky, kdy je nastavena jednotka jako push-pull, se objeví na výstupu úroveň mezi logickou nulou a jedničkou.

Obrazový snímač po zapojení pracuje v základním nastavení, což znamená nastavení plného rozlišení a kontinuální vysíláni nasnímaného obsahu. Pro synchronní snímání obrazu však toto nastavení není vhodné, je tedy nutné přenastavit pomocí I2C senzor na snapshot mode. V tomto módu obrazový snímač nesnímá obraz kontinuálně, ale čeká na náběžnou hranu signálu EXPOSURE. Poté exponuje obraz a je odeslán pomocí paralelního rozhraní.

Dalším důležitým bodem nastavení obrazového senzoru je možnost ovlivnit rychlost odesílání dat. Senzor obsahuje dva registry, které umožňují zpomalit celý přenos exponovaného obrazu následujícím způsobem: Registr horizontal blanking umožňuje nastavit počet pixelů o kolik bude větší řádek snímače. Jedná se o virtuální zvětšení řádku snímače, kdy senzor po odeslaní skutečného obrazového řádku čeká na tento nastavený počet virtuálních pixelů. Tím se dosáhne prodloužení doby mezi vysílacími řádky.



Obr. 3.2: Sekvence signálů pro čtení dat ze senzoru pomocí I2C rozhraní



Obr. 3.3: Sekvence signálů pro zápis dat ze senzoru pomocí I2C rozhraní

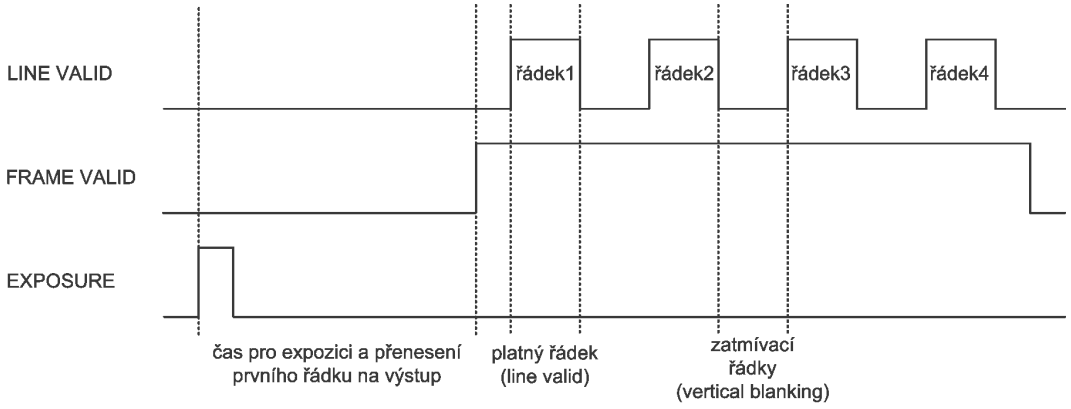
Druhým registrem je vertikal blanking, který po odeslání všech řádků snímače čeká na nastavený počet řádků, než označí přenos za kompletní. Tímto způsobem je možné získat čas pro zpracování obrazových dat ze senzoru. Nastavení tohoto registru přímo ovlivňuje přenosovou rychlost senzoru

3.2.1 Formát výstupních dat

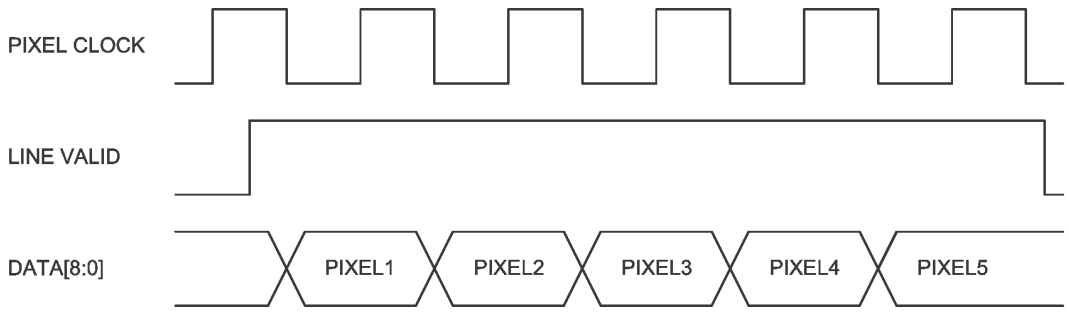
Výstupní data jsou ze senzoru vysílána paralelně po 8bitové sběrnici. Pro synchronizaci dat senzor používá signály Line Valid, který indikuje platný řádek a Frame Valid označuje, že je přenášen obrázek a signál Pixel Clock, který indikuje platná data odpovídající pixelům.

Na obr. 3.4a je znázorněn průběh obrázku o čtyřech řádcích. Po náběžné hraně signálu EXPOSURE, proběhne expozice snímku. Poté je snímek posílán po řádcích kde signál LINE VALID určuje platný řádek a signál FRAME VALID určuje platnost snímku.

Na obr. 3.4b je znázorněn průběh jednoho řádku obrazového snímače. Na každou náběžnou hranu hodinového signálu je odeslán na výstup jeden obrazový bod řádku.



(a)



(b)

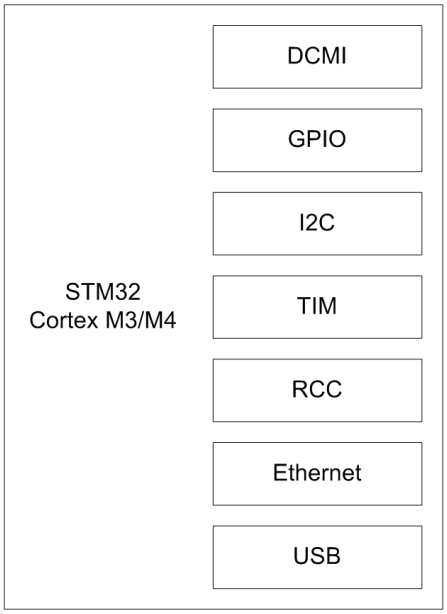
Obr. 3.4: Výstupní signály z CMOS obrazového senzoru

Kapitola 4

Popis periferií, využitých v mikrokontroleru STM32

Základním stavebním kamenem této práce je použití jednočipového mikrokontroleru STM32F207/407. Jedná se o 32bitový mikrokontroler s jádrem ARM cortex M3/M4. Jedná se o výrobek firmy STM [6, 7]. Mikrokontroler obsahuje velkou spoustu periferií, nyní si představme ty, které jsou v této úloze přímo využívány.

4.1 Periferie



Obr. 4.1: Periferie mikrokontroleru STM32F2/4 používané v projektu

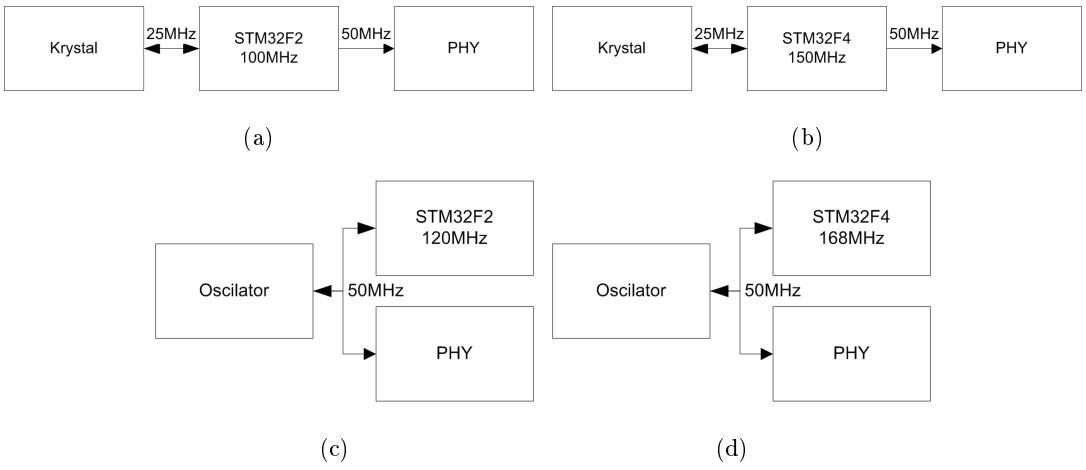
4.1.1 RCC - Periferie pro nastavení hodinových signálů mikrokontroleru

Jedná se o periferii pro nastavení hodinových signálů jednotlivým periferiím a nastavení hodinového signálu pro jádro mikrokontroleru. Zde je třeba dávat pozor při použití na jiném typu hardwaru, protože hodinový signál je zpravidla odvozován od externího oscilátoru, je třeba dávat pozor s jakou hodnotou program počítá.

Prvním zdrojem signálu je krystal, nejčastěji o frekvenci 25MHz(tato hodnota je použita na všech vývojových deskách) ale je možno použít i jiné hodnoty krystalů.

V tomto případě bylo třeba odlišit program pro mikrokontroler STM32F207 od STM32F407, jelikož oba mikrokontrolery mají různé maximální frekvence. Pro mikrokontroler STM32F207 je nastavena hodnota frekvence jádra na 100MHz obr. 4.2a , maximum mikrokontroleru je 120MHz. Této hodnoty není možné dosáhnout, jelikož je třeba, aby frekvence byla dělitelná na hodnotu 50MHz(signál nutný pro fyzickou vrstvu). Pro STM32F407 je nastavena frekvence jádra na 150MHz obr. 4.2b. maximum je opět 168MHz, je opět nutné, aby frekvence byla dělitelná na 50MHz.

Jiný případ nastane při použití 50MHz krystalového oscilátoru, který vytváří přímo hodinový signál o frekvenci 50MHz. Zde je možné, aby jádro procesoru běželo na maximální možnou frekvenci, to jest 168MHz pro STM32F4X7 obr. 4.2c a 120MHz pro STM32F2X7 obr. 4.2d.



Obr. 4.2: Zapojení hodinového obvodu STM32F2/4 a jeho dopad na frekvenci jádra

4.1.2 General-purpose I/Os - nastavení vstupně výstupních bran mikrokontroleru

General-purpose I/Os (dále jen jako GPIO) slouží pro nastavení jednotlivých bran mikrokontroleru a jejich pinů. GPIO umožní nastavit funkci pinu, zda bude vstupní čí výstupní nebo bude

podřízen některé jiné periferii(USART,SPI,Ethernet, ...). Bez tohoto nastavení není možné použít jakýkoliv pin ať už jako softwarově ovládaný nebo přiřazený jakékoliv periferii. Dalším důležitým nastavením je zde rychlost pinů. Pro snížení rušení generovaného mikrokontrolerem je vhodné nastavit rychlost pinů na nejbližší vyšší rychlost, která je požadována. Zde samozřejmě záleží na tom, jakou funkci bude daný pin vykonávat. V této úloze ale postačí mít všechny piny nastaveny na rychlost 50MHz(jedná se o vylepšení, které může pomoci pro snížení rušení od mikrokontroleru při EMC testech).

4.1.3 DCMI - Digital camera interface

Digital Camera Interface je periferie uzpůsobená pro příjem dat, typicky z CMOS obrazových senzorů. Umožňuje vyvolat přerušení pří čtení řádků a konci celého obrazového rámce. Zároveň umožňuje uložit pouze část přenesených dat. V tomto případě však tato funkce nebude použita. Periferie umí načítat obrazová data v rozsahu 8-14 bitů. Avšak je nutno si pamatovat, že pro překročení hranice 8bitů již jeden obrazový bod bude zabírat 16bitů. Proto je použita v úloze jen 8bitová hloubka obrazu, která ušetří dvojnásobné množství paměti oproti 10ti a vícebitové hloubce. Periferie bude využita pouze k načítání 8bitových dat ze senzoru a jejich předání DMA, které se postará o uložení.

4.1.4 DMA - Periferie umožňující nastavit přímý přístup do paměti bez účasti procesoru

DMA umožňuje přenos dat bez účasti jádra mikrokontroleru. V tomto zařízení je možné nastavit přenos slov o velikosti 8/16/32bitů a počet přenosů až 65536. Po přenesení všech dat přenos skončí přerušením nebo musí být konec přenosu hlídán softwarově. Pro přenos většího počtu dat nebo nekonečný přenos lze s výhodou použít circular mod, který plní vyhrazený adresový prostor stále dokola.

Přenos je prováděn jednak mezi pamětí mikrokontroleru, z FLASH do RAM nebo z RAM do RAM. V těch to případech je možné použít pouze normální přenos a není možné využít kurikulární mód, jelikož velikost přenosu musí být přesně dána.

Mnoho periferií STM32 ovšem umožňuje automaticky využívat periferii DMA a inicializovat přenos. Proto je možné využít přenosu Periferie do paměti nebo opačný směr z paměti do periferie. Zde je možné využívat cirkulární mód, kde není předem známa velikost dat. Využití tohoto modu bude probráno v následujících kapitolách.

4.1.5 Ethernet - periferie pro komunikaci s řadičem fyzické vrstvy Ethernetu

Ethernet periferie se skládá z media access control(dálej jen MAC) části, která zajišťuje příjem paketů na nejnižší vrstvě sítě Ethernet a jejich odesílání. Dále obsahuje periferie vestavěný DMA kontroler, který přenáší přijaté pakety z interní paměti MAC do paměti RAM mikrokontroleru, nebo pakety z RAM do MAC. MAC dále zajišťuje komunikační rozhraní pro komunikaci s řadičem fyzické vrstvy. Používáno je rozhraní MII respektive jeho redukovaná varianta RMII a rozhraní SMI pro komunikaci s řadičem fyzické vrstvy.

4.1.6 Universal seria bus - Periferie obsluhující univerzální sériové rozhraní

Universal serial bus(dále jen USB) umožňuje využívat toto rozhraní pro komunikaci s STM32. Pro univerzálnost je toto rozhraní přítomno na vývojové desce a je možno ho použít v jiném projektu.

Další možnost jak využít rozhraní USB je upgrade software, kdy interní bootloader umožňuje použít toto rozhraní pro nahrání nové aplikace, aniž by bylo nutno využít rozhraní pro debugování SWD/JTAG.

4.2 Periferie Inter-Integrated Circuit

Inter-Integrated Circuit neboli I2C dvouvodičová sběrnice typu slave-master. V této úloze je tato periferie použita pro nastavování parametrů obrazového senzoru.

4.3 Obvody čítačů a časovačů

V úloze budou použity pro generování hodinového signálu pro CMOS obrazový senzor. Druhý čítač poté bude generovat krátký pulz pro signál EXPOSURE obrazového senzoru. Tento čítač bude vnitřně spouštěn pomocí periferie Etherne, starající se o přesný čas.

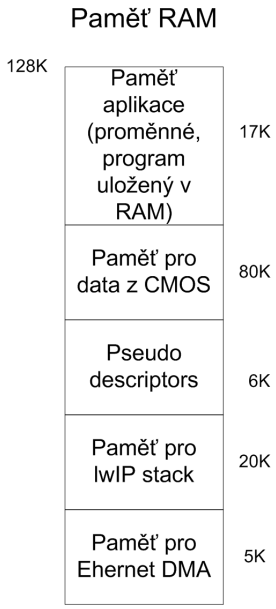
4.4 Organizace paměti mikrokontroleru STM32 a její využití

Důležitým prvkem při vytváření programového vybavení mikrokontroleru je rozdělení programové paměti. Jelikož není využito žádného operačního systému, je veškerá paměť přidělována pevně a během vykonávání programu již není možná jakákoliv změna jejího rozložení. Paměťový prostor pro oba typy mikrokontrolerů je poněkud odlišný. Novější typ mikrokontroleru obsahuje větší paměť RAM a tím poskytuje možnost uložení obrázku ve větším rozlišení do paměti.

4.4.1 Paměť STM32F207

Tento typ uP obsahuje 128KB paměti. Paměť je třeba rozdělit mezi následující periferie DCMI, EthernetMAC a lwIP stack. Jako nejvýhodnější rozložení bylo vybráno obr. 4.3.

4.5 Periferie



Obr. 4.3: Organizace paměti RAM mikrokontroleru STM32F2/4

EthernetMAC a lwIP stack mají přidělenou pevnou paměť. Ta může být využita jednak pro ukládání dat tak i pro případné zpracování uložených dat.

4.5.1 Paměť STM32F407

STM32F407 má mírně rozšířenou paměť oproti předchozímu typu. Z důvodu co největší kompatibility kódu je počítáno pouze se stejným místem jako má STM32F207. Rozšíření paměti se týká 64KB CCM (core coupled memory) RAM, které jsou však přístupné pouze z jádra mikrokontroleru a nikoli pomocí DMA. Není tedy možné využít tuto paměť k ukládání dat z DCMI. Jediné možné použití je přidělit tuto paměť lwIP stacku za podmínky, že data nebudou kopírována do jeho struktur pomocí DMA, ale pouze přesouvána pomocí vlastního jádra. Nebo je možné využít tuto paměť pro uložení programu.

4.5.2 Flash Paměť

FLASH paměť obou typů mikrokontrolerů je sama o sobě vcelku pomalá, pracuje pouze na 24MHz. Při sekvenčním kódu ovšem jádro přednačítá budoucí instrukce z paměti FLASH a zároveň je využíváno faktu, že instrukce netrvají pouze jeden hodinový cyklus ale déle. Problém se ovšem může vyskytnout v průběhu přerušení, kdy jádro nemá přednačtená žádná data a musí začít načítat znovu. Pro zrychlení tohoto procesu je možné převést kód do paměti RAM, čímž se zrychlí jeho načítání až 7x oproti paměti *FLASH*.

Přenesení vybraných funkcí do paměti RAM provedeme následovně. V kódu jsou označeny metody a funkce které chceme uložit do paměti RAM pomocí příkazu:

```
__attribute__((section ("RamCode")))
```

kde RamCode je označení bloku kódu pro linker. Nyní je již pouze třeba sdělit linkeru, že chceme blok označený RamCode uložit místo do paměti FLASH do paměti RAM. V programu Keil upravíme linker script následovně:

```
RW_IRAM1 0x20000000 0x00020000  {  ; RW data
  .ANY (+RW +ZI)
  *.o(RamCode)
 }
```

Kapitola 5

Použití lwIP stacku na mikrokontroleru STM32

lwIP stack pracuje jako mezičlánek mezi rozhraním MAC mikrokontroleru a mezi uživatelským programem. Stack je důležitý hlavně pro přenosy pomocí protokolu TCP, kde zajišťuje kontrolu došlých paketů a jejich potvrzování, případně se stará o jejich znovu odeslání.

Důležitou součástí stacku je jeho paměťový prostor který je možné definovat a tím ovlivňovat velikost stacku v paměti RAM. Stack bude použit pouze pro zprávy o délce do 256B. Není nutné mít obrovský paměťový prostor, který je mu v základním nastavení přiřazen tedy, přes 1500B na jednu zprávu.

5.1 Implementace lwIP na STM32

5.2 Organizace paměti stacku

Důležitou součástí stacku je jeho paměťový prostor, který je možné definovat a tím ovlivňovat velikost stacku v paměti RAM. Jelikož stack bude použit pouze pro zprávy o délce do 256B, není nutné mít obrovský paměťový prostor, který je mu v základním nastavení přiřazen. Soubor lwipopts.h umožňuje tento paměťový prostor přehledně upravovat. Nejdůležitější je paměť rezervovaná pro heap. Jedná se o paměťový prostor, který může být dynamicky alokovaný pro potřeby stacku či uživatele. Pro potřeby programu postačí 10KB. Další důležitou paměťovou strukturou je velikost PBUF struktur. Pro lepší kooperaci s deskriptory MAC byla zvolena stejná velikost, a to 256B. Jelikož PBUF strukturu používá stack primárně pro příjem a vysílání dat, je zvolen počet těchto struktur na 20(20x 256B), aby pokryl možnosti přijímaných a vysílaných dat. Dále je zde ještě možné ovlivnit počet souběžně hlídaných timeoutů, například

pro TCP zprávy či počet souběžně probíhajících připojení. Pro účel programu je vyžadováno pouze jedno TCP připojení a dvě UDP připojení, která využije PTPd.

5.3 Nedostatky stacku

Při rozhodování kterým způsobem budou odesílána obrazová data, bylo testováno odesílání dat přes TCP. Zde nastaly problémy se stabilitou stacku a špatnou kontrolou jeho paměti, kdy došlo k selhání stacku a jeho zaseknutí. Další nevýhodou byla nutnost uchovávat odesílaná data v paměti po dobu než budou pomocí TCP potvrzena, což vyžadovalo větší paměťovou náročnost než bylo možné stacku dát. Proto nakonec bylo zvoleno použití protokolu UDP k odesílání objemných dat. Další komplikací pro rychlý přenos jakýchkoli dat pomocí stacku je nutnost je kopírovat do deskriptorů MAC. Proto pro odesílání dat byl nakonec použit přímý způsob odesílání, který umožní využití DMA k odeslání dat. LwIP stack bude tedy použit pouze pro synchronizaci pomocí PTPd a pro řídicí zprávy mezi PC a mikrokontrolerem, prováděné přes TCP.

Kapitola 6

Precision time protokol podle normy IEEE1588

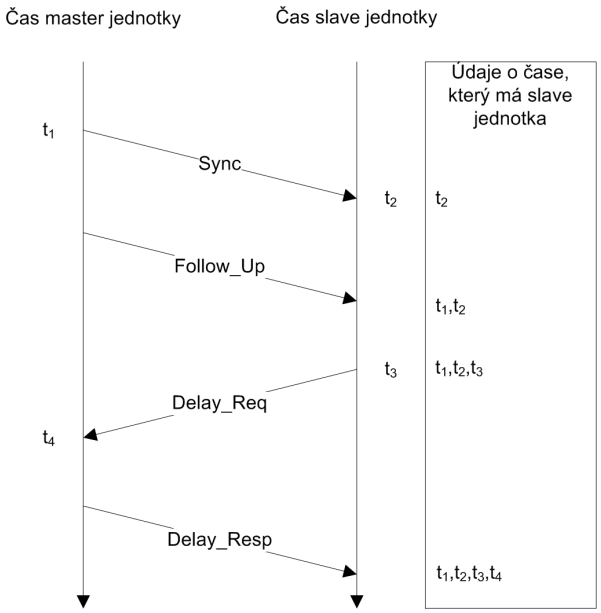
Protokol PTP umožňuje synchronizaci času v sítích Ethernet. Využívá přitom jednoduchého způsobu odesílání zpráv a měření času jejich odchodu/příchodu mezi dvěma body sítě.

6.1 Popis práce PTP

PTP protokol pracuje v uzavřené síti, kdy existuje vždy jeden PTP master, jehož čas se snaží podřízené jednotky dosáhnout. Existují dvě možná uspořádání sítě buď end to end(zkráceně E2E), kdy se předpokládá viditelnost mezi masterem a více jednotkami, nebo peer to peer kdy master vidí vždy pouze na jednu podřízenou jednotku. V této úloze je využito zapojení end to end jelikož jednotky jsou propojeny přes switch, který nepodporuje protokol PTP. Omezíme se tedy na popsání pouze případu E2E.

Jednotka s nejlepšími parametry, které jsou pevně nastaveny, získá možnost být masterem. V případě jejího odpojení od sítě se stane masterem jednotka s druhými nejlepšími parametry. Jednotky si při synchronizaci času mění pouze poslední z parametrů, který je počítán v zavilosti na zpožděních v komunikaci. Může se tedy stát že jednotka s dobrými předchozími parametry se stane podřízenou díky vysokým zpožděním na síti. Jelikož jednotky mohou mít měnící se adresy, je využito vysílání Broadcastových zpráv(zpráva je poslána všem účastníkům sítě) pomocí protokolu UDP. Komunikace probíhá pomocí pěti zpráv následovně, obr. 6.1.

Master vyslílá periodicky *Annonce_Message*, ve které jsou uloženy parametry vedoucí jednotky Pokud některá z jednotek zjistí, že její parametry jsou lepší, vyšle *Annonce_Message* ona a tím přebere vedení sítě. Pokud se *Annonce_Message* od vedoucí jednotky odmlčí na



Obr. 6.1: Synchronizace času mezi master jednotkou a slave jednotkou

dobu po kterou vyžadují jednotky opětovné potvrzení *Annonce_ Message*, vyšle jednotka *Annonce_ Message* zprávu, aby byla zajištěna přítomnost jednotky typu master v síti.

Master má dále za úkol periodicky vysílat *Sync* zprávu a uloží si čas t_1 kdy jí odeslal a následně pošle *Follow_ Up* zprávu s časem t_1 pro podřízené jednotky. Podřízená jednotka (slave jednotka) přijme zprávu *Sync* a uloží si čas t_2, kdy zpráva došla. Po příchodu zprávy *Follow_ Up* od jednotky master vypočítá zpoždění linky od mastera k podřízené jednotce. Jednotka poté může vyslat zprávu *Delay_ Req* a uloží si čas vyslání t_3. Jednotka master přijme zprávu *Delay_ Req* a uloží si čas jejího příchodu t_4. Ten odešle zpět ve zprávě *Delay_ Resp*. Podřízená jednotka má tedy možnost z časů t_3 a t_4 spočítat zpoždění sítě na cestě k masterovi. Z obou zpoždění je poté vypočítán aritmetický průměr, který udává zpoždění mezi masterem a podřazenou jednotkou. Nakonec podřízená jednotka z obdržených časů upraví svůj vlastní čas, aby odpovídal co nejvíce jednotce master.

6.2 Použití na STM32F

STM32F podporuje možnost pracovat s protokolem PTP na úrovni fyzické vrstvy. Každá zpráva která přijde obsahuje čas, kdy byla přijata. V přerušení je obsah PTP zprávy předáván lwIp stacku, překopírován do struktury pbuf (paměťová struktura lwIP stacku), která je rozšířena oproti klasické lwIp struktuře o místo pro uložení příchozího času.

Při odesílaní je deskriptor nastaven tak, aby po odeslání zprávy uložil čas odeslání. Program

poté čeká až bude čas uložen a tento pak uloží do struktury pbuf.

Toto uspořádání odesílání bylo však nepraktické, jelikož program by musel vždy čekat na odeslání deskriptoru. Jelikož program nerozlišuje zda jde o PTP data nebo normální data kde není třeba čekat na PTP značku. Muselo dojít k dalšímu rozšíření struktury pbuf o identifikátor, zda bude vyžadováno uložení přesného času. Pokud ano, program vyčkává na uložení dat z deskriptoru. V opačném případě pokračuje dále.

Vlastní PTP je převzato z projektu PTPd který se stará o správné fungování PTP a úpravu času v mikrokontroleru. PTPd portace a STM32 vznikla v rámci projektu TA01010988. Portace je kompatibilní s mikrokontrolerem STM32F207/407. Nebrání tedy nic v jejím použití.

Důležitou částí je pro nás pouze část PTPd a to net.c, která oproti verzi STM32F107 pracuje s rozšířením pro nastavení čekání na čas v struktuře pbuf a po jejím odeslání na vynulování tohoto nastavení.

```
    /* send the buffer. */
#if LWIP_PTP
    if(time!=NULL){
        p->wait_for_ptp=0x1;//wait only for event.
    }
#endif
    result = udp_sendto(pcb, p, (void *)addr, pcb->local_port);

    if (ERR_OK != result)
    {
        ERROR("netSend: Failed to send data (%d)\n", result);
        goto fail02;
    }

    if (NULL != time)
    {
#if LWIP_PTP
        time->seconds = p->time_sec;
        time->nanoseconds = p->time_nsec;
        p->wait_for_ptp=0x0;//now not wait
#else
```

Tímto způsobem je optimalizováno odesílání normálních zpráv, aniž by musely čekat na data, která program nepoužívá. PTPd je implementováno následujícím způsobem. Po inicializaci Ethernetového rozhraní je inicializován lwIP stack a poté může být inicializován PTPd stack.

```
  PTPd_Init();
```

Poté je již periodicky v hlavní programové smyčce, volána metoda

```
ptpd_Periodic_Handle(LocalTime);
```

která obsluhuje vlastní PTPd stack.

6.3 Odesílání dat a optimalizace rychlosti pro PTP

Standardní způsob, který používá ST pro odesílání PTP zpráv pomocí lwIP je funkční. Bohužel zvolené řešení může limitovat výkon Ethernetových přenosů pro odesílání dat u kterých není potřeba ukládat data o časové značce. Z tohoto důvodu byla provedena optimalizace, pomocí které byla rozšířena struktura lwIP stacku tak, aby uchovávala informaci o tom, zda je odesílaná zpráva PTP či nikoliv. V případě, že zpráva neobsahuje požadavek na PTP čas, je odeslána a již program nečeká na odesílaný čas. V případě, že je třeba čas uchovat, program vyčká ve smyčce na uložení času, který je pak dále zpracováván.

```
if(p->wait_for_ptp!=0){//je odesilana zprava PTP?
      /*metoda ceka na cas z PTP a ulozi ho do pbuf struktory*/
    if( ETH_SUCCESS == ETHPTP_Prepare_Transmit_Descriptors(framelength,&timestamp)) ↩
        {
      p->time_sec = timestamp.tv_sec;
      p->time_nsec = timestamp.tv_nsec;
      p->wait_for_ptp=0x0;
    } else {
          ETH_LowLevelInUse=0;
      return ERR_IF;
    }

}else{
          //neni treba cekat na PTP cas, zprada je odeslana standartim zpusobem
    ETH_Prepare_Transmit_Descriptors(framelength);
}
```

6.4 Využití PTP pro přesné měření

Pro účely měření pomocí kamer je třeba aby měl mikrokontroler možnost zahájit expozici obrázku v definovaném čase. Ideálním řešením, které zajistí nejpřesnější možné spuštění, je využít možnosti nastavit periferii Ethernetu tzv target time, kdy je pomocí metody

```
void ETH_SetPTPTargetTime(uint32_t HighValue, uint32_t LowValue);
```

nastaven čas ve kterém se provede spuštění EXPOZICE. Může se jednat buď o přerušení, které se v tento čas provede a ve kterém se může provést spuštění měření. Nebo výhodněji periferie Ethernetu umožňuje pro tento případ využít vnitřní spojení s triggerem periferie TIMER2.

```
    /*
        TIM2 Exposure init
        setup TIM2 to One Pulse Mode
    */
    TIM_TimeBaseStructure.TIM_Period = 0x10000;
    TIM_TimeBaseStructure.TIM_Prescaler = 1;
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

    TIM_OC1FastConfig(TIM2,TIM_OCFast_Enable);

    /* PWM1 Mode configuration: Channel1 */
    TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM2;
    TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStruct.TIM_Pulse = 1;                    //CCR1
    TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;

    TIM_OC1Init(TIM2, &TIM_OCInitStruct);

    TIM_ARRPreloadConfig(TIM2, ENABLE);

    /* One Pulse Mode selection */
    TIM_SelectOnePulseMode(TIM2, TIM_OPMode_Single);

    /* Input Trigger selection */
    TIM_RemapConfig(TIM2,TIM2_ETH_PTP);
    TIM_SelectInputTrigger(TIM2,TIM_TS_ITR1);

    /* Slave Mode selection: Trigger Mode */
    TIM_SelectSlaveMode(TIM2,TIM_SlaveMode_Trigger);
```

Toto nastavení vygeneruje krátký pulz, který v definovaný čas spustí uložení obrázku senzorem CMOS. Jelikož je vše řízeno pouze hardwarem, nevzniká při tomto řešení žádné zpoždění, které by bylo způsobeno například nutností obsluhovat přerušení. A dále toto řešení nezbrzdí program v hlavní programové smyčce. Nakonec po uložení a zpracování obrázku je nutné opět nastavit metodu ETH_SetPTPTargetTime po opětovné spuštění.

```
void ETH_SetPTPTargetTime(uint32_t HighValue, uint32_t LowValue);
```

Kapitola 7

Použití MAC na STM32 pro komunikaci se sítí Ethernet

Mikroprocesor STM32F207(407) obsahuje periferii, která po přípojení řadiče fyzické vrstvy (dále jen PHY) dokáže komunikovat se sítí typu Ethernet. Vlastní periferie se dá rozdělit na dvě části: na část starající se o přístup k síti MAC a část, která se stará o předávání dat do paměti pomocí DMA.

Příjem dat je realizován pomocí PHY která převede data do paralelní formy a pomocí 4/2 vodičového paralelního rozhraní je předá do MAC. Pro výstup dat opět obsahuje paralelní 4/2 vodičovou sběrnici, pomocí které je možné data vysílat.

Ke konfiguraci PHY slouží sériové rozhraní I2C, pomocí kterého lze nastavovat parametry PHY nebo je z ní vyčítat. Tohoto rozhraní se využívá při inicializaci MAC, kdy jsou pomocí auto-negotiation zjištěny parametry sítě(rychlost 10/100Mb, Half/Full duplex režim) a ty jsou poté nastaveny do MAC. Dále je možné povolit funkci MIDI-MDIX, pomocí které dokáže PHY zjistit typ kabelu kterým je připojena a v případě potřeby změní vysílací piny na přijímací a přijímací na vysílací. Uživatel se tak nemusí obávat použití nesprávného typu kabelu.

Auto-negotiation probíhá pouze na straně PHY. Při inicializaci Ethernetu v mikrokontroleru musí nejprve program vyčkat na inicializaci auto-negotiation v PHY. Proběhne-li auto-negotiation v pořádku, indikuje to příslušný bit v PHY a parametry Ethernetu jsou nastaveny podle vyčtených parametrů. Není-li možné auto-negotiation provést, nastaví se základní parametry pro mikrokontroler i pro PHY.

Zde nastává menší problém. Jelikož není možné jednoduše zjistit stav PHY, je jen těžko kontrolovatelné nastavení sítě. Při změně sítě, což může být rychlost nebo zapojení do jiné sítě za běhu mikrokontroleru, musí dojít k restartu programu a nové inicializaci Ethernetu, jinak

nebudou nové parametry sítě nastaveny a komunikace nebude pracovat.

Při přizpůsobování stacku pro mikrokontroler a fyzickou vrstvu je nutné nastavit správné adresy registrů v PHY, do kterých bude procesor nastavovat parametry sítě. Základní program počítá s použitím fyzické vrstvy od National Semiconductors DP83848. Při nepřizpůsobení této části je možná nefunkčnost PHY. V souboru stm32f2x7_eth_conf.h byla povedena změna, která bude podporovat obě fyzické vrstvy, a to ST802RT i DP83848.

```
#ifdef ST802RT
    #define PHY_SR     ((uint16_t)17) /* Value for DP83848 PHY */
#endif
#ifdef DP83848
    #define PHY_SR     ((uint16_t)16) /* Value for ST802RT PHY */
#endif

/* The Speed and Duplex mask values change from a PHY to another, so the user
   have to update this value depending on the used external PHY */
#ifdef ST802RT
    #define PHY_SPEED_STATUS            ((uint16_t)0x0200) /* Value for ST802RT PHY */↩
            //0x0002
    #define PHY_DUPLEX_STATUS           ((uint16_t)0x0100) /* Value for ST802RT PHY */↩
            //0x0004
#endif
#ifdef DP83848
    #define PHY_SPEED_STATUS            ((uint16_t)0x0002) /* Value for DP83848 PHY */↩
            //0x0002
    #define PHY_DUPLEX_STATUS           ((uint16_t)0x0004) /* Value for DP83848 PHY */↩
            //0x0004
#endif
```

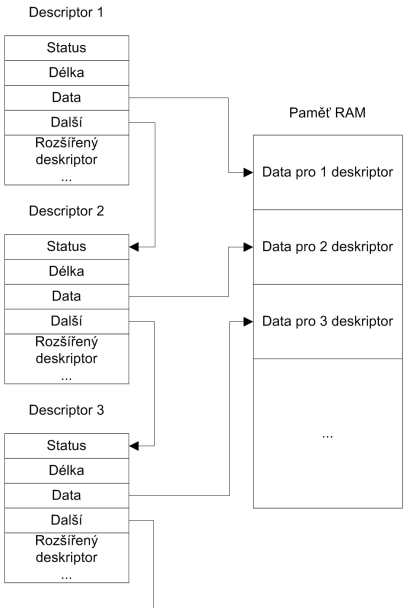
7.1 Descriptory

Před spuštěním Ethernetového rozhraní je třeba nastavit tzv. deskriptory. Jedná se o struktury v paměti RAM, které jsou využívány pro příjem a vysílání dat pomocí Ethernetu.

U STM32F107 mají deskriptory velikost 4slov tedy 4x 32bitů. První slovo uchovává informace o nastavení deskriptoru. Druhé slovo nese informaci o velikosti paměti, kterou příslušný deskriptor obhospodařuje. Třetí slovo ukazuje na adresu v paměti, kde jsou data náležící deskriptoru a čtvrté ukazuje na následující deskriptor.

Popsaná struktura deskriptorů umožňuje, aby byly zapojeny do řetězu, kdy každý deskriptor ukazuje na následující člen. Mikrokontroler umožňuje i jiné uspořádání do kruhu, kdy je znám

počet deskriptorů, avšak v této aplikaci je využito pouze uspořádaní do řetězu, obr. 7.1.



Obr. 7.1: Naznačení organizace deskriptorů v řetězové struktuře a alokace paměti pro data deskriptoru

7.1.1 Rozšíření deskriptorů pro PTP

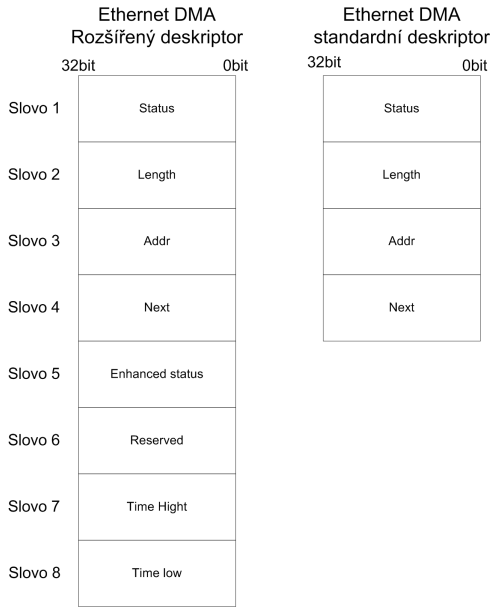
Komplikace s tímto uspořádáním nastává v použití protokolu pro přesný čas PTP, kdy je třeba použít deskriptor k uchování času příchozí/odchozí zprávy. V tuto chvíli je využito třetí a čtvrté slovo k uchování času. Je proto nutné mít po dobu kdy deskriptor nepoužívá program ale mikrokontroler, aby tato data byla uchována v nějaké nezávislé struktuře a po přečtení časových dat z deskriptoru opět obnovena.

Tento problém řeší mikrokontroler STM32F207/407 možností využít rozšířený typ deskriptoru na 8 slov, z nichž dvě poslední jsou již pevně určena pro časovou PTP značku, obr. 7.2.

Ovladač vrstvy Ethernetu však nebyl pro tuto možnost vybaven, proto bylo třeba upravit program tak, aby využíval možnosti rozšířených deskriptorů, místo použití klasického přeukládání jejich hodnot.

Nejprve je nutné povolit rozhraní Ethernet, aby využilo možnosti použít rozšířený deskriptor:

```
#ifdef USE_ENHANCED_DMA_DESCRIPTORS
  /* Enable the Enhanced DMA descriptors */
  ETH->DMABMR |= ETH_DMABMR_EDE;
```



Obr. 7.2: Porovnání rozšířeného deskriptoru na STM32F2/4 a klasického na STM32F107

```
#endif /* USE_ENHANCED_DMA_DESCRIPTORS */
```

Tím zajistíme že časová data budou ukládána do sedmého a osmého slova deskriptoru.

7.2 Optimalizace deskriptorů

7.2.1 Zmenšení deskriptorů

Základní nastavení deskriptorů počítá s velikostí rezervovaných dat na pevnou velikost odpovídající maximální možné velikosti Ethernetového rámce pro sítě do 100Mb. Každý deskriptor tedy rezervuje paměť 1514B. Toto provedení sebou nese jisté komplikace v případě, kdy je třeba využít paměť RAM i pro jiné účely. V tomto případě očekáváme periodicky přicházející PTP zprávy a zároveň i možné řídicí zprávy, jejichž velikost nebude zdaleka nikdy dosahovat 1514B. Největší objem dat tady bude spočívat na obrazových datech.

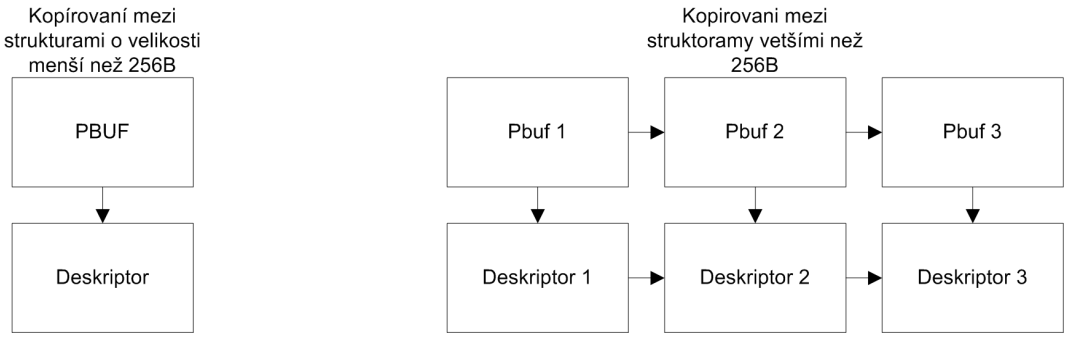
Jako nejjednodušší řešení tedy vychází možnost použít vlastností deskriptorů, kdy je možné použít menší deskriptor než je délka maximálního Ethernetového rámce a v případě nutnosti uložit tuto zprávu do více deskriptorů. Ethernetový ovladač byl tedy upraven tak, aby používal pouze deskriptory o velikosti 256B. V případě, že přijde větší zpráva, nebo bude-li třeba odeslat delší zprávu než 256B bude tato zpráva rozdělena do více deskriptorů, obr. 7.3.

Dalším problémem, který byl řešen, je předávání dat mezi Ethernetovým stackem a deskriptory, kdy stack počítá pouze s možností kdy je deskriptor větší než je paměťová struktura stacku.

Obr. 7.3: Porovnání práce deskriptorů s daty menšími než 256B a s daty přesahujícími tuto velikost

Proto muselo být upraveno ukládání tak, aby byly zohledněny nastavené velikosti jak stacku tak paměťové struktury stacku, obr. 7.4.

```
    while(write_len!=q->len){
        /* Check if the descriptor is owned by the ETHERNET DMA (when set) or CPU (↩
            when reset) */
        if((DMATxDescActual->Status & ETH_DMATxDesc_OWN) != (u32)RESET)
        {
        /* Return ERROR: OWN bit set */
            ETH_LowLevelInUse=0;
            return ERR_MEM;
        }
        DMATxDescActual->Status=DMATxDescActual->Status& (~(ETH_DMATxDesc_FS|↩
            ETH_DMATxDesc_LS));
        /*kontrola zda je kopirovany retezec vetsi nez zbyvajici velikost descriptoru↩
            */
        if((q->len-write_len+buffer_len)>=ETH_TX_BUF_SIZE){
            buffer =  (u8 *)(DMATxDescActual->Buffer1Addr);
        memcpy((u8_t*)&buffer[buffer_len],(u8_t*)(((u32)q->payload)+write_len), ↩
            ETH_TX_BUF_SIZE-buffer_len);
            write_len+= ETH_TX_BUF_SIZE-buffer_len;
            DMATxDescActual=(ETH_DMADESCTypeDef *)(DMATxDescActual->↩
                Buffer2NextDescAddr);
            buffer_len=0;
        }else{
             buffer =  (u8 *)(DMATxDescActual->Buffer1Addr);
         memcpy((u8_t*)&buffer[buffer_len],(u8_t*)(((u32)q->payload)+write_len), q->↩
             len-write_len-buffer_len);
             write_len+= q->len-write_len-buffer_len;
             buffer_len+=q->len-write_len;
        }
    }
```



Obr. 7.4: Kopírování dat z pbuf struktur lwIP stacku do deskriptorů patřících rozhraní Ethernet

7.2.2 Přenastavování parametrů deskriptorů pro odesílání obrazových dat

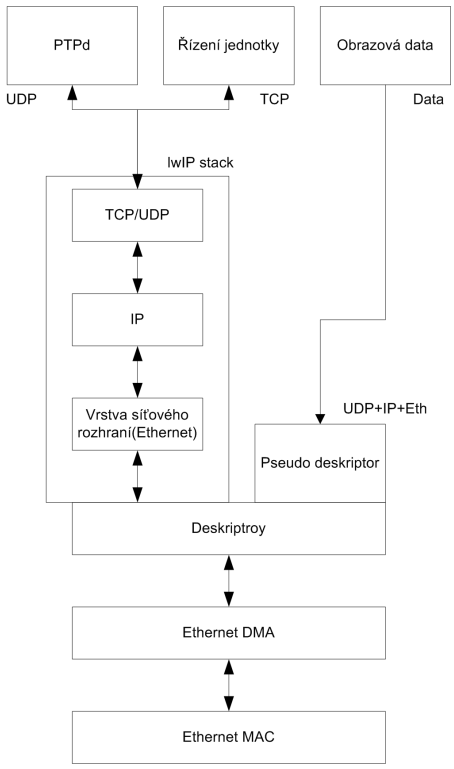
Představené uspořádání stacku sebou nese jednu významnou nevýhodu a to tu, že program obsahuje jednak deskriptory a jednak vlastní stack, který pří odesílání/přijímání dat musí překopírovávat data mezi pamětí deskriptoru a pamětí stacku, což zpomaluje běh programu a zatěžuje jádro mikrokontroleru.

V případě normálního použití pro PTPd či obsluhu programu toto zpomalení je poměrně zanedbatelné, avšak při větším množství dat například obrazových, kdy je preferována co nejvyšší přenosová rychlost, není toto uspořádání efektivní. Proto byla vytvořena možnost odesílat data co nejrychleji bez nutnosti kopírování a tedy i bez nutnosti použití stacku, obr. 7.5.

Pro tuto možnost bylo v paměti vytvořeno místo pro odesílanou zprávu a pro hlavičku dané zprávy(Ethernet+IP+UDP). Zpráva je typu UDP, kde není nutné potvrzovat došlá data ani hlídat jejich pořadí. Každý z těchto pseudo-deskriptorů je nadefinován při inicializaci Ethernetové periferie. Je mu přiřazena pevná zdrojová MAC adresa, stejná jakou používá mikrokontroler a IP adresa jakou má mikrokontroler. Toto uspořádání nám tedy dovolí použít deskriptor o maximální možné velikosti, kterou je možné odeslat po Ethernetu. Po překopírování dat do tohoto pseudo-deskriptoru může být inicializováno jeho odeslání, obr. 7.6a.

Program zkontroluje který deskriptor je volný pro odeslání dat přes Ethernet. Jelikož deskriptory mají velikost 256B, program upraví velikost na velikost odesílaných dat a přepíše ukazatel na příslušná data pseudo-deskriptoru, čekající na odeslání, obr. 7.6b. Zároveň zálohuje původní data patřící k deskriptoru, jako velikost a umístění jeho původní paměti. Aby bylo možné tento deskriptor po odeslání dat vrátit do původního stavu, nastaví program v prvním slově možnost vyvolat přerušení po odeslání tohoto deskriptoru. Nakonec je deskriptor předán rozhraní MAC, která se postará o jeho odeslání, obr. 7.6c.

Po odeslání je provedeno přerušení. Jelikož mikrokontroler má pouze jedno přerušení pro



Obr. 7.5: Principilní uspořádání lwIP stacku a využití pseudo deskriptorů pro posílání dat

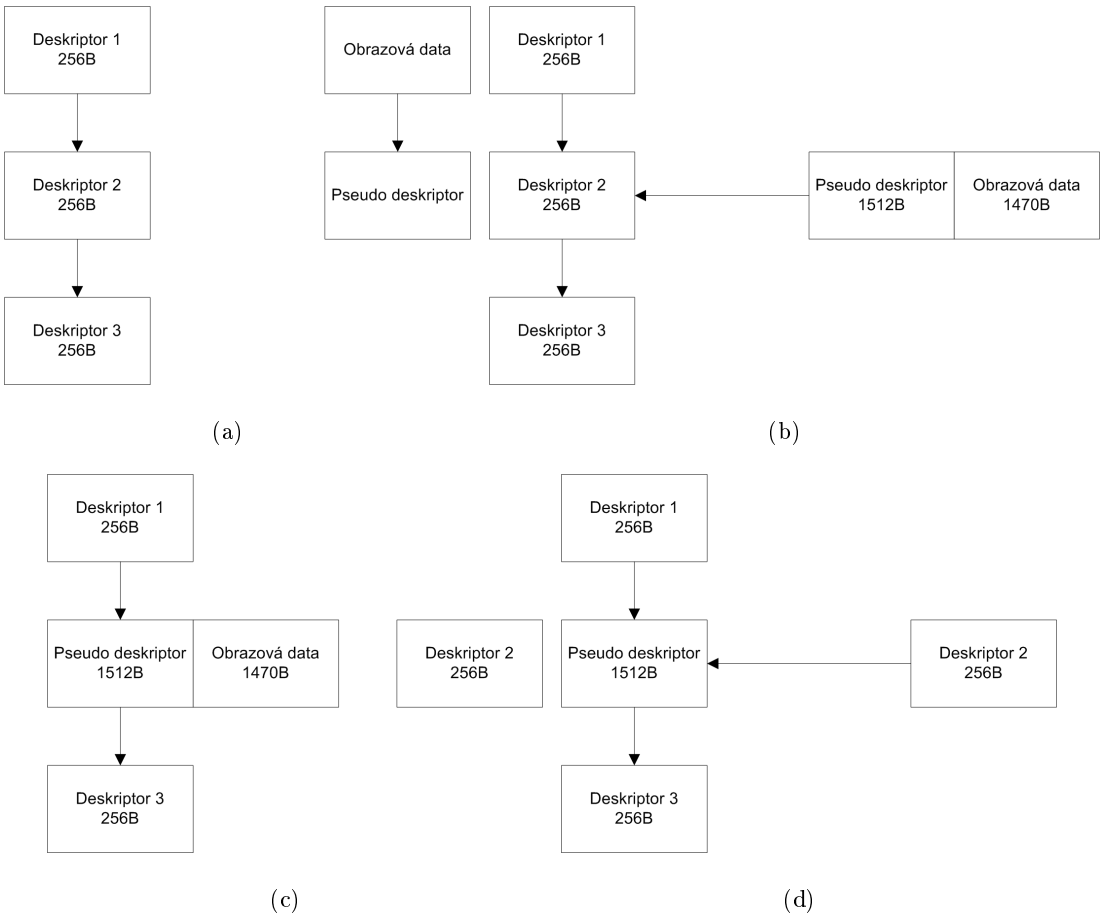
periferii Ethernet, musí být jako první v přerušení zkontrolováno zda přišlo od odcházejícího deskriptoru, aby mohl být tento deskriptor opět obnoven, obr. 7.6d. V případě že by jako první byla provedena kontrola příchozích zpráv, mohlo by se stát, že v tomto průběhu by se stack pokusil použít deskriptor, který používáme pro odeslání jiných dat, čímž by došlo ke komplikacím při vracení paměti a zároveň k možnosti ztráty dat.

7.3 Odesílání dat pomoci deskriptoru bez lwIP stacku

Pro odesílání dat je třeba využít co největší rychlosti kterou nabízí Ethernet a zároveň je zapotřebí co nejméně vytížit jádro mikrokontroleru, aby bylo možno zpracovávat i jiná data. Z tohoto důvodu byla navržena metoda, která využívá hardwarových periferií mikrokontroleru pro odesílání dat bez nutnosti využití softwaru.

7.3.1 Návrh odesílacího algoritmu

Odesílání dat je navrženo tak, aby využilo možností DMA periferií a krátké obsluhy přerušení pro co nejmenší možné zatížení jádra mikrokontroleru.



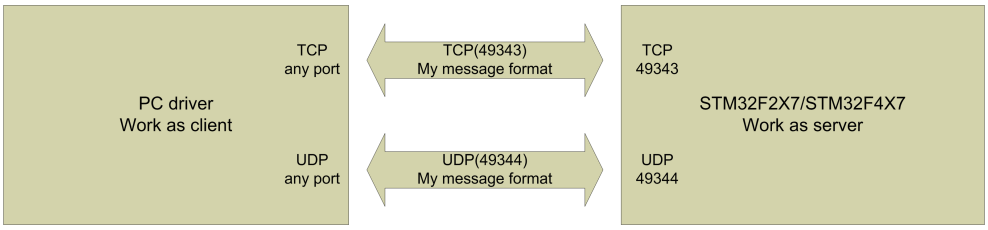
Obr. 7.6: Postup odesílání dat při použití pseudo deskriptorů

Kapitola 8

Komunikace s okolím

8.1 Navrhy komunikace

Rozvržení komunikace s okolím je na obr. 8.1. Ke komunikaci je primárně používán protokol TCP, který zaručuje potvrzení přijetí zprávy popřípadě její znovu odeslání. Vzhledem k těmto vlastnostem bude využíván protokol TCP pro řídicí zprávy pro kameru. Druhým typem připojení je UDP, které bude používáno pro rychlý přenos obrazových dat. Pro komunikaci pomocí



Obr. 8.1: Typy připojen ke kameře

TCP zpráv je pro kameru vytvořen jednoduchý komunikační protokol. První dva bajty zprávy obsahují hlavičku která je pro kameru specifická(0xFAF9).

Následující dva bajty již udávají typ zprávy, která bude provedena. Dostupné TCP zprávy reprezentuje obr. 8.2. Zprávu "Init UDP"vysílá klient spolu s číslem portu UDP, na kterém bude očekávat příchozí data od kamery. Po přijetí této zprávy inicializuje program deskriptory pro odesílání dat přes UDP a přiřadí jim port který je obsažen ve zprávě "Init UDP"a IP adresu, kterou zjistí z došlé TCP zprávy.

Po nastavení odešle zpět klientovi zprávu "UDP open", která informuje o připravenosti kamery vysílat data. Klient poté inicializuje odesílání dat příkazem "Posílej UDP data"následně kamera odešle požadovaná obrazová data pomocí UDP protokolu.

Po skončení přenosu odešle kamera zprávu "UDP data poslana", která obsahuje počet odeslaných paketů a zároveň čas ve kterém byla pořízena obrazová data. Poté klient zkontroluje počet došlých dat. V případě, že se počet příchozích dat liší od počtu v kontrolní zprávě, odešle klient zprávu "Potvrzení chybného příjmu UDP dat"a přenos se opakuje.

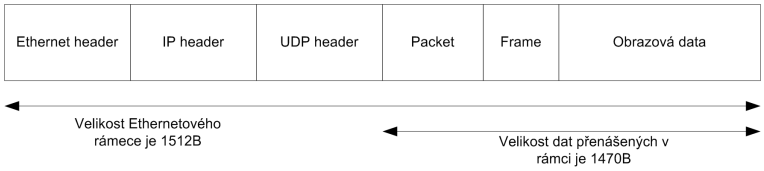
V opačném případě vyšle klient zprávu "Posilej UDP data"a server opět vyšle požadovaná data. V případě kdy je ukončena komunikace, klient odešle zprávu "Close UDP"a server již nereaguje na předchozí zprávy kromě zprávy "Init UDP"

Komunikace Aplikace ->kamera

Obr. 8.2: Typy zpráv podporované kamerou

8.2 Komunikační protokol pro přenos obrazových dat

Pro přenos obrazových dat byl nakonec zvolen protokol UDP, který nejvíce vyhovuje požadavkům na rychlý přenos dat s co nejmenším zpožděním a náročností na paměť a na procesorový čas pro lwIP stack. Na obr. 8.3 je znázorněn formát UDP zprávy, které obsahuje obrazová data, a každý paket obsahuje pořadové číslo z aktuálního snímka(Packet) a identifikační číslo snímku(Frame).



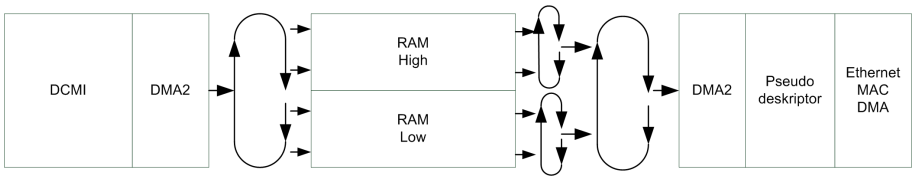
Obr. 8.3: Typy zpráv podporované kamerou

Kapitola 9

Souběh jednotlivých částí programu

9.1 Načítání dat z obrazového senzoru

Data ze senzoru jsou načítána automaticky pomocí rozhraní DCMI. DCMI je předá DMA, které je přesouvá do paměti RAM, která je určena pro ukládání obrazových dat. Jelikož senzor má větší rozlišení než je paměť dostupná v mikrokontroleru a jelikož DMA není výhodné zastavovat, je DMA nastaveno na cirkular mód. Po dosažení konce přidělené paměti RAM začne DMA opět od začátku. Tím se ovšem stane, že začne přepisovat data, která byla uložena na začátku. Aby data nebyla smazána musí být přesunuta, v tomto případě budou odeslána pomocí rozhraní Ethernet.



Obr. 9.1: Znázornění principiálního řešení přenosů dat z DCMI do paměti RAM a dále do pseudo-deskriptorů které využije Ethernet pro odeslání dat

K touto účelu využijeme přerušení, které obsahuje DMA a to HalfTransferComplete(dále jen HTC) a TransferComplete(dále jen HTC). V každé obsluze tohoto přerušení inicializujeme přenos pomocí rozhraní Ethernet.

9.2 Odesílání bloků dat přes Ethernet

Data jsou odesílána automaticky dokud nedojde k odeslání všech dat uložených přes rozhraní DCMI.

9.3 Periodic handle

Program se snaží pracovat co nejvíce asynchronně využívaje v maximálním množství přerušení od jednotlivých periferií. V hlavní programové smyčce je však potřeba obsluhovat části, které nemají asynchronní část. PeriodicHandle pro lwIp stack se stará o kontrolu čítačů hlídajících TCP připojení. Umožňuje periodickou kontrolu IP adres a je využit pro odeslání zpráv v případě že je program nebyl schopen odeslat přímo.

Druhou částí která se periodicky zpracovává je stack PTPd, který se stará o nastavování přesného času mikrokontroleru. Zároveň zpracovává přijaté zprávy PTP a vysílá vlastní PTP zprávy.

Poslední periodickou částí která zde musí být ošetřena je kontrola vlastních přenosů mezi pamětí RAM a Ethernetovými deskriptory, jelikož se občas může stát, že přerušení od jedné této periferie nemůže spustit další následný přenos. Tato kontrola tedy řeší stav, kdy by program musel čekat na asynchronní událost a tím by zbytečné plýtval časem který by zde mohl být použít pro kontrolu těchto periodických služeb.

9.4 Obsluhy přerušení a jejich priority

Jelikož úloha je postavena především za pomocí asynchronních událostí, které jsou obsluhovány pomocí přerušení, je třeba aby byly jasně stanoveny jednotlivé priority přerušení. Ty budou definovat jaké rozhraní bude obsluhováno přednostně tak, aby nedošlo k zhroucení programu nebo k jeho zpomalení.

Největší priorita tedy byla přisouzena rozhraní DMA, které se stará o přenos dat z DCMI. Tento přenos nelze nijak ovlivnit, proto je třeba, aby data byla odebrána vždy když, o to periferie DCMI požádá. Proto také má DMA pro DCMI maximální možnou prioritu přenosu.

Nižší prioritu má přerušení od DMA, které se stará o přenos z paměti RAM, kde jsou uložena obrazová data do pseudo-deskriptorů pro Ethernet.

Nejnižší prioritu tedy má přerušení obsluhující Ethernet. Jelikož všechny přenosy které byly zmíněny výše, jsou závislé na souběhu, není možné aby rozhraní Ethernet přerušovalo ostatní periferie.

Kapitola 10

Realizace

Kapitola rozbírá postup práce při vytváření hardwarového vybavení k řešení diplomové práce

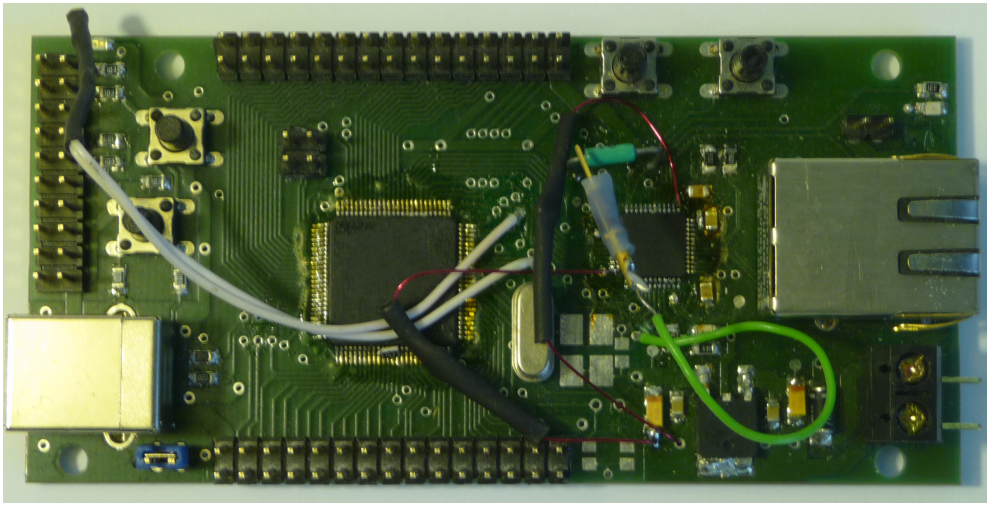
10.1 Použití vývojové desky

Z počátku byla použita vývojová deska z [12] kterou vytvořil Adam Bařtipán jako součást bakalářské práce. Při použití této vývojové desky bylo odhaleno několik vad, které způsobovaly jisté nestandardní chování při použití řadiče fyzické vrstvy.

10.1.1 změna rozhraní MII-RMII

Vývojová deska měla však jednu podstatnou nevýhodu, která jí neumožňovala použití spolu s obrazovým senzorem. Deska má spojení mezi řadičem fyzické vrstvy a mikrokontrolerem vytvořeno pomocí MII. Toto rozhraní však využívá pro komunikaci 17vodičů, zde ovšem dochází ke kolizi s rozhraními,která jsou nutná pro komunikaci s obrazovým senzorem(I2C). Proto je potřeba provést hardwarovou úpravu zapojení, která umožní změnu na RMII, které využívá ke komunikaci pouze 9vodičů. Zároveň také uvolní rozhraní nutná ke komunikaci s obrazovým senzorem. Fyzická úprava desky je na obr.10.1.

Na obr. 10.2 je výsledné zapojení vývojové desky a obrazového senzoru CMOS. Pro ověření funkčnosti daného zapojení byla vytvořena jednoduchá aplikace, která umožní odesílat obrázek v polovičním rozlišení CMOS senzoru a to 376x240 bodů. Princip práce programu pro mikrokontroler je na obr. 10.3. Program nejprve spustí odběr dat ze senzoru do paměti RAM. Po uložení redukovaného obrázku do paměti jsou data odeslána pomocí rozhraní Ethernet do obslužné aplikace v PC. Na obr. 10.4 je ukázka grafického uživatelského prostředí pro PC. Program se pouze připojí na danou IP adresu a pak pouze čeká na příchozí data. Po příjmu celého obrázku



Obr. 10.1: Fyzická úprava vývojové desky s rozhraním MII na rozhraní RMII

data zobrazí. Program pro mikrokontroler a obslužný program na PC jsou přiloženy na CD.

10.2 Návrh procesorové desky plošného spoje pro kameru

Po otestovaní zapojení mikrokontroleru a řadiče fyzické vrstvy Ethernetu pomocí RMII a vyzkoušení funkčnosti SDIO karty bylo možno začít s návrhem desky plošného spoje pro kameru.

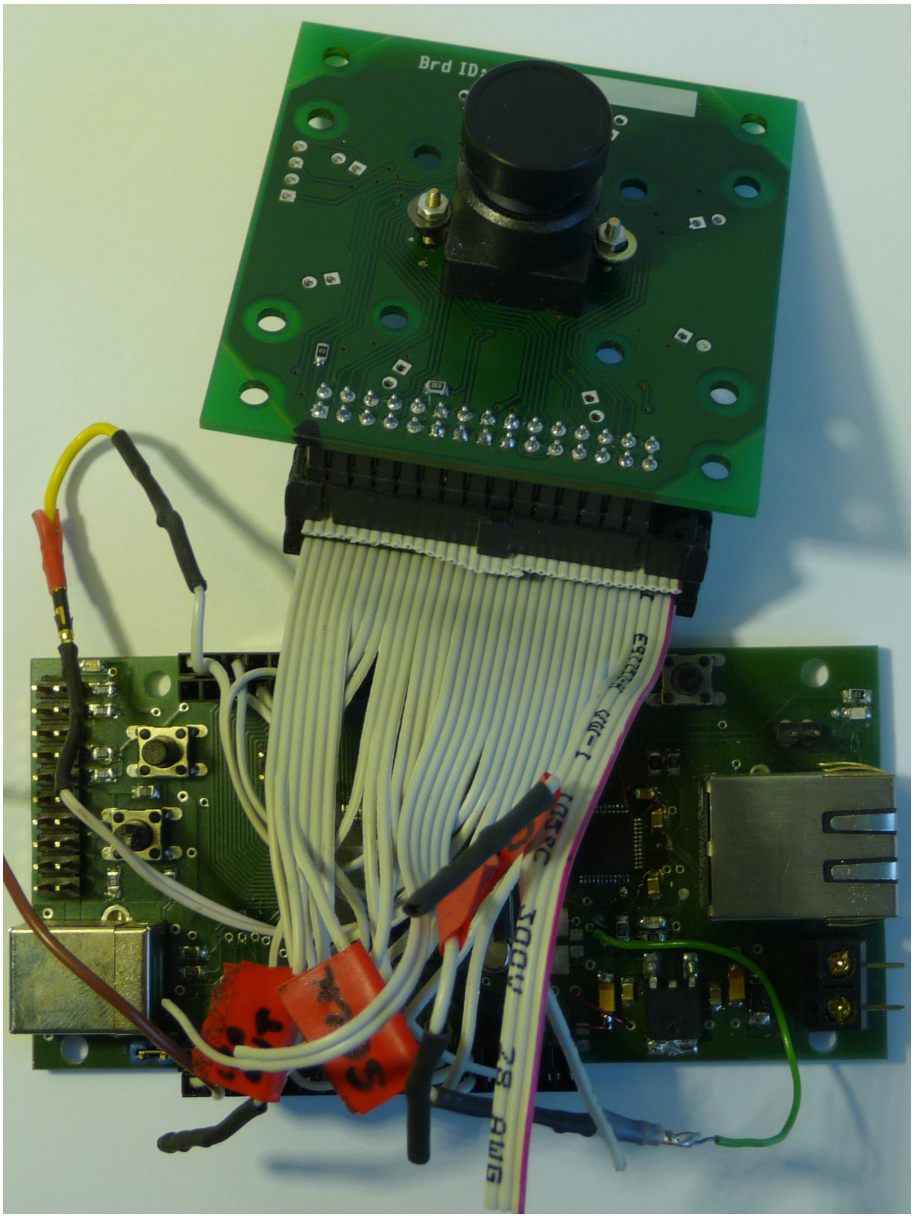
10.2.1 Rozbor obsahu desky plošného spoje

Z hlavních požadavků na desku plošného spoje je přítomnost 30pinového konektoru pro připojení obrazových senzorů, stejných v rámci laboratoře videometrie obr. 10.5.

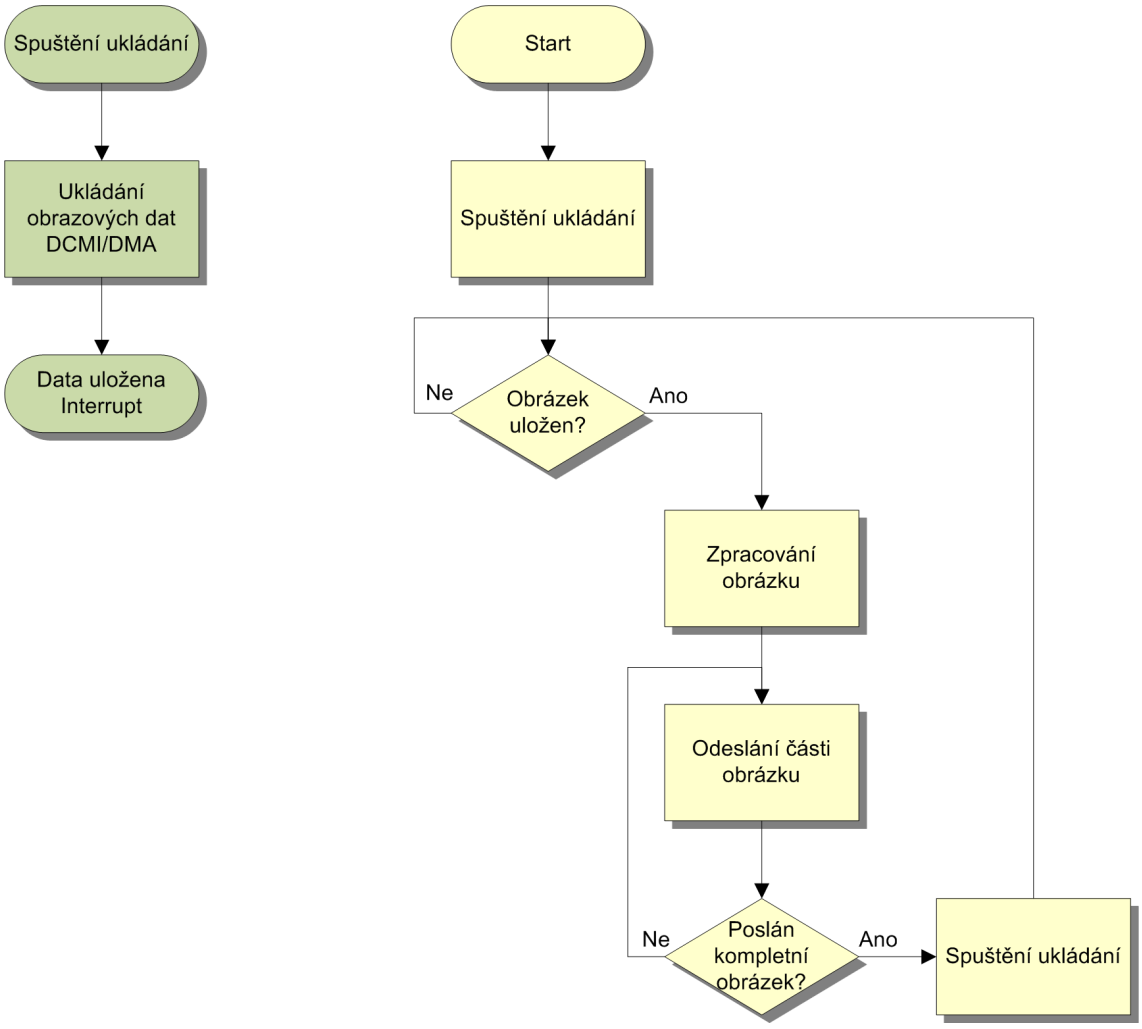
Dalším požadavkem je přítomnost řadiče fyzické vrstvy Ethernetu. V první verzi desky plošného spoje byl použit řadič fyzické vrstvy ST802RT [8]. Ve třetí verzi došlo z důvodů ukončení výroby předešlého řadiče fyzické vrstvy ke změně na vrstvu DP83848 [9].

Pro programování mikrokontroleru je využito rozhraní SWD, jež obsahuje kromě signálu pro data a hodinový signál i vodič pro společnou zem a vodič pro RESET. Toto uspořádání nebylo zvoleno náhodně, ale využívá podobného rozložení pinů jako má SWD konektor u kitu STM32F4discovery. Díky této podobnosti je možné pomocí tohoto kitu jednoduše programovat mikrokontroler, obr. 10.6.

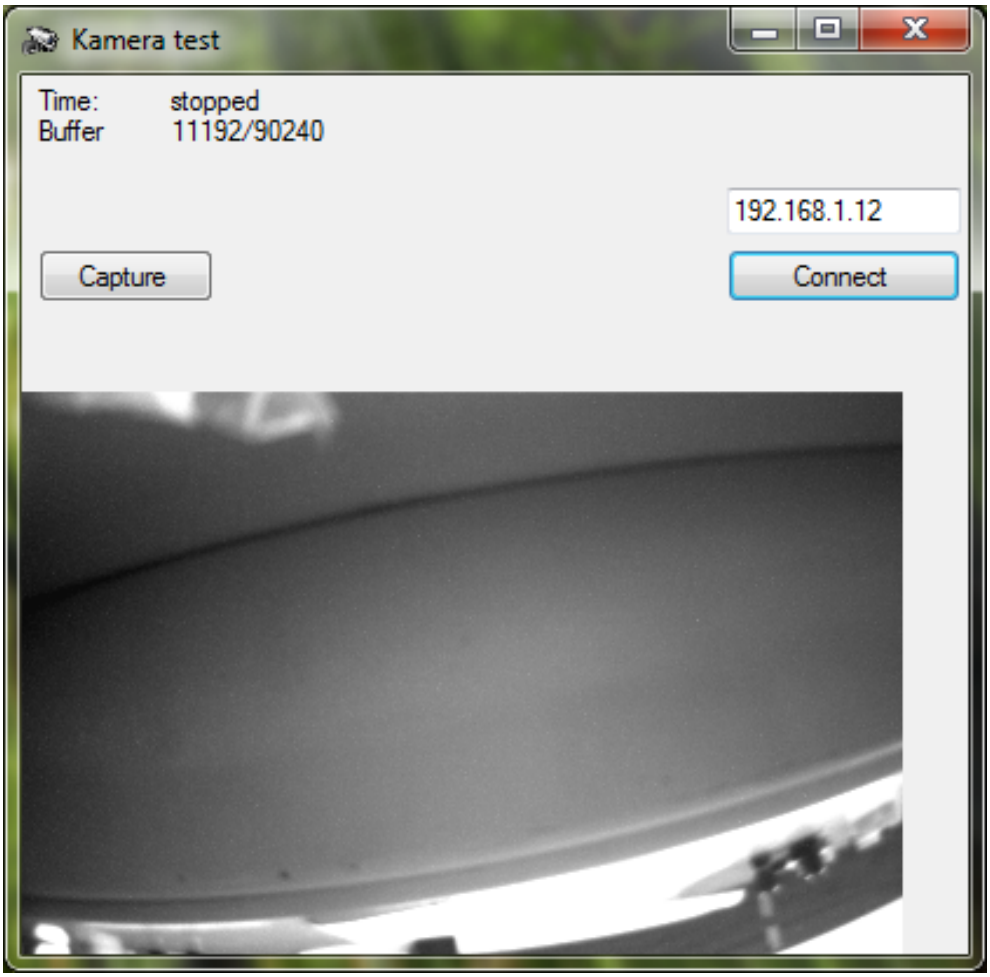
Pro větší univerzálnost desky plošného spoje bude také vybavena rozhraním USB pro možnost programování přes bootmode procesoru, popřípadě pro možnost využít USB ke komunikaci a vytvořit tak USB kameru obr. 10.7. V případě využití výše zmíněného bootloaderu, musí být propojena následující propojka na obr. 10.8.



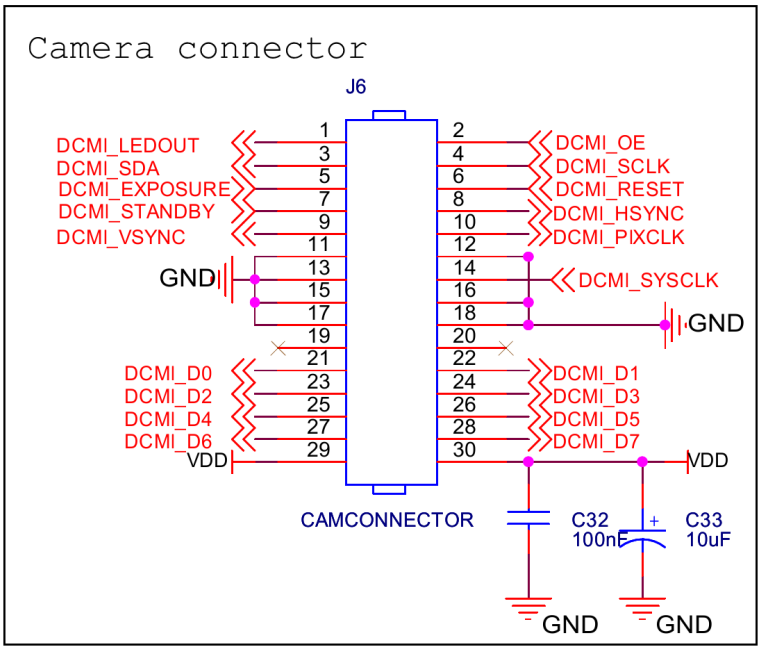
Obr. 10.2: Zapojení vývojové desky s rozhraním RMII s obrazovým senzorem CMOS pro testování funkčnosti zapojení



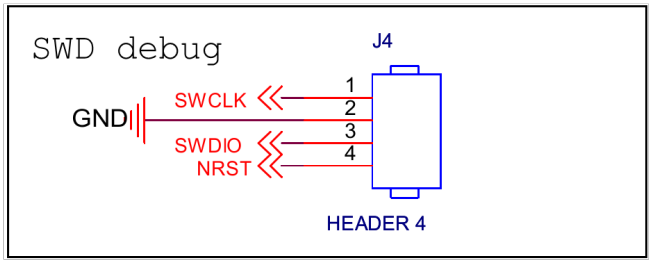
Obr. 10.3: Blokové schéma práce demonstračního programu pro ověření funkčnosti připojení senzoru CMOS



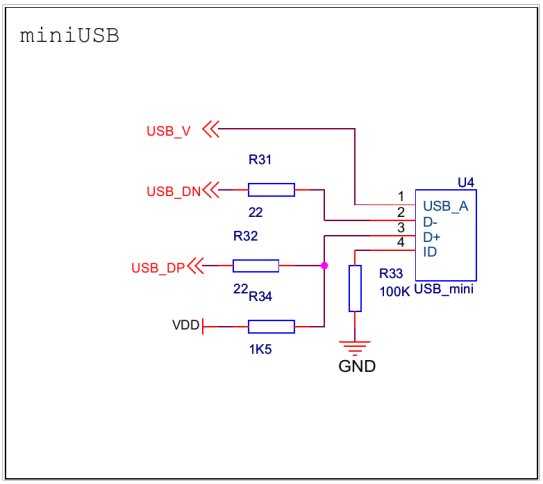
Obr. 10.4: Obslužný program pro demonstraci funkčního připojení CMOS obrazového senzoru



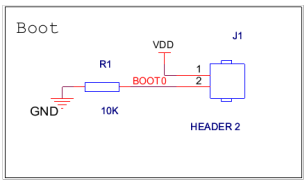
Obr. 10.5: Připojení konektoru pro CMOS obrazový snímač DCMI rozhraní mikrokontroleru STM32F2/4



Obr. 10.6: Připojení konektoru pro SWD rozhraní k mikrokontroleru STM32F2/4

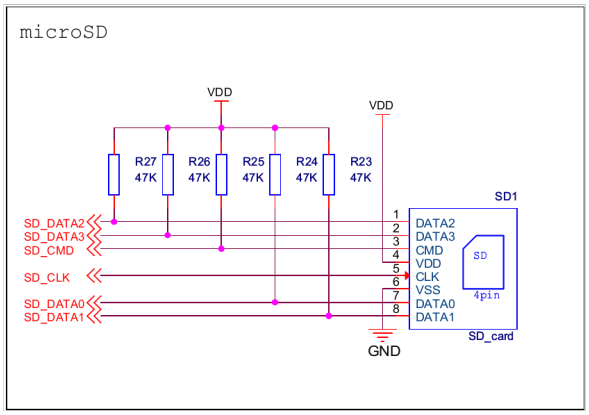


Obr. 10.7: Připojení konektoru mini USB k mikrokontroleru STM32F2/4



Obr. 10.8: Propojka pro aktivaci bootloader funkcionality mikrokontroleru

Pro možnost využití kamery jako záznamového zařízení, bude přítomen adaptér pro mikroSD kartu, která bude sloužit jako velkokapacitní paměť obr. 10.9.



Obr. 10.9: Připojení adapteru pro mikroSD kartu k STM32F2/4

Posledními rozhraními, která budou přítomna, bude možnost využít USART a SPI pro možnost sériové komunikace, obr. 10.10. Nakonec bude deska plošného spoje obsahovat několik indikačních LED, obr. 10.11 a jedno uživatelské tlačítko a tlačítko pro reset mikrokontroleru.

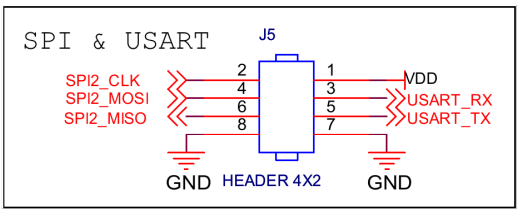
10.2.2 Vytvoření desky plošného spoje

Před vytvořením obrysu desky bylo nutné rozhodnout, zda bude počítáno s možností zapouzdření do krabičky. Nakonec byla vybrána krabička typu KP59(www.krabicky.cz). Pro tuto krabičku byl po jejím přeměření vytvořen obrys pro desku plošného spoje na obr. 10.12.

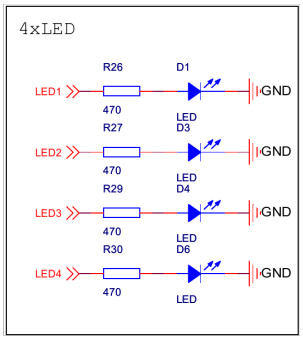
10.2.3 Kompletní návrh desky

Kompletní navržená deska obsahuje všechny zmiňované komponenty. Navíc je možné ji použít samostatně jako vývojovou desku pro jiné typy aplikací. Při návrhu se podařilo umístit většinu součástek pouze na jednu stranu desky plošného spoje, kromě adaptéru pro mikroSD kartu. Toto řešení velice usnadňuje osazování desky plošného spoje, obr. 10.13.

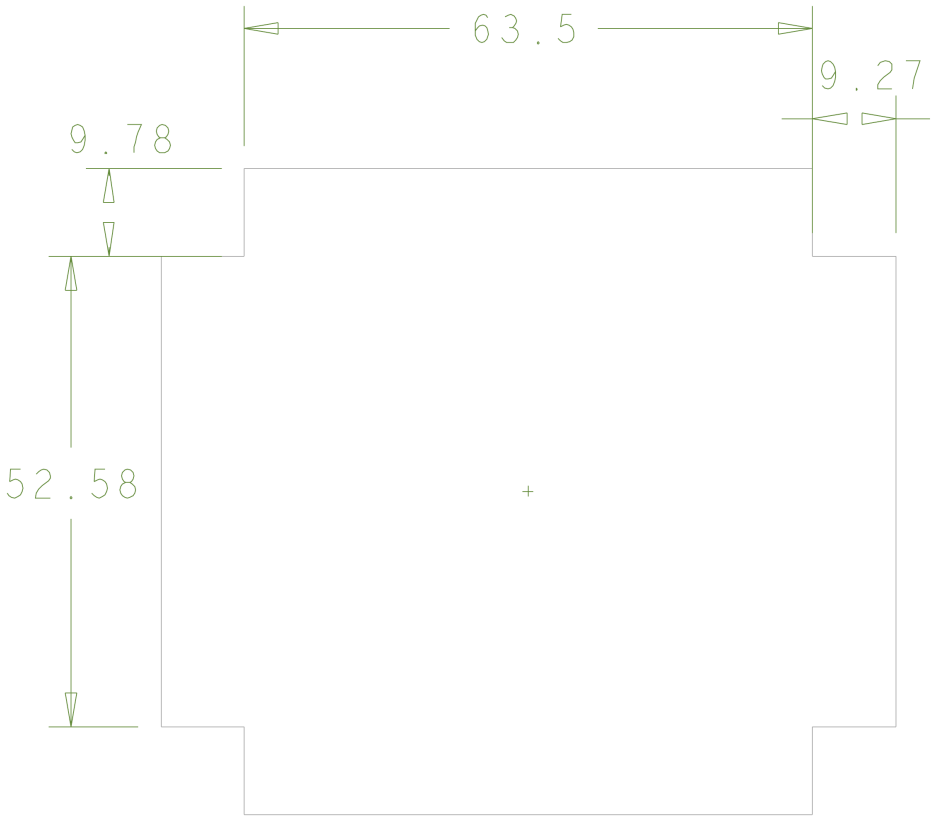
Osazovací plán, kompletní schéma zapojení a rozpisku součástek lze najít dále v dodatcích.



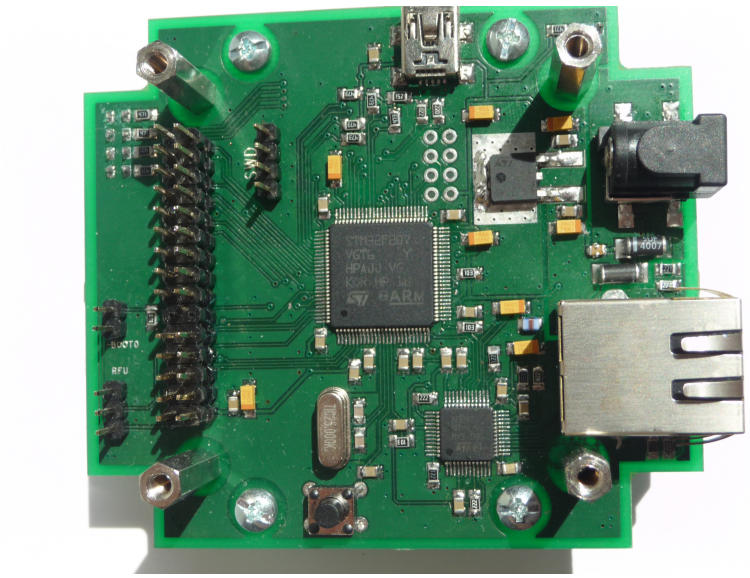
Obr. 10.10: Konektor pro připojení rozhraní SPI a UART k STM32F2/4



Obr. 10.11: Připojení indikačních LED k mikrokontroleru STM32F2/4



Obr. 10.12: Obrys pro desku plošného spoje odpovídající vnitřním rozměrům krabičky KP59



Obr. 10.13: Kompletní osazená deska s mikrokontrolerem STM32F207

Kapitola 11

Vytvoření ovládacích prostředků pro PC

11.1 Ovladač pro komunikaci s kamerovou jednotkou

Pro ovládání jednotek z PC byl vytvořen ovladač v programovacím jazyce C#. Ovladač je vytvořen jako objekt a umožňuje připojení k jedné jednotce.

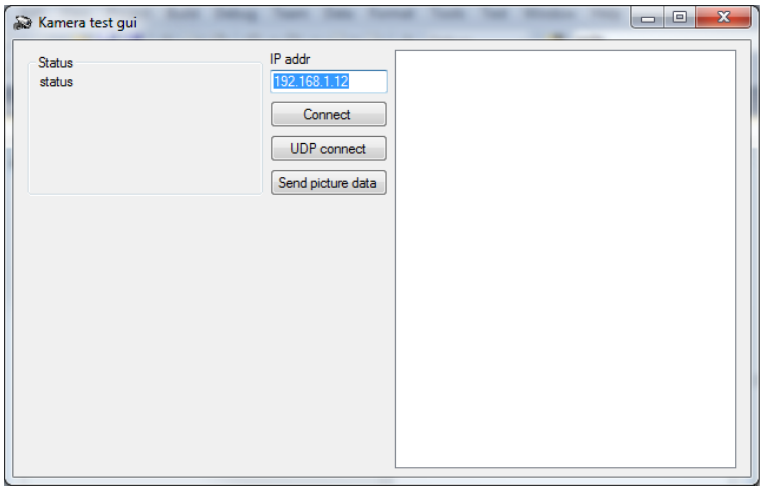
Ovladač se připojí pomocí TCP definovaným protokolem k jednotce. Obsahuje pak metody podporující jednotlivé příkazy definované v kapitole Komunikace s okolím.

Ovladač rovněž podporuje připojení pomocí UDP kde hlídá příchozí data, ty ukládá do paměti a po načtení kompletního obrázku data uloží do bitmapy a umožní předání nadřazenému programu.

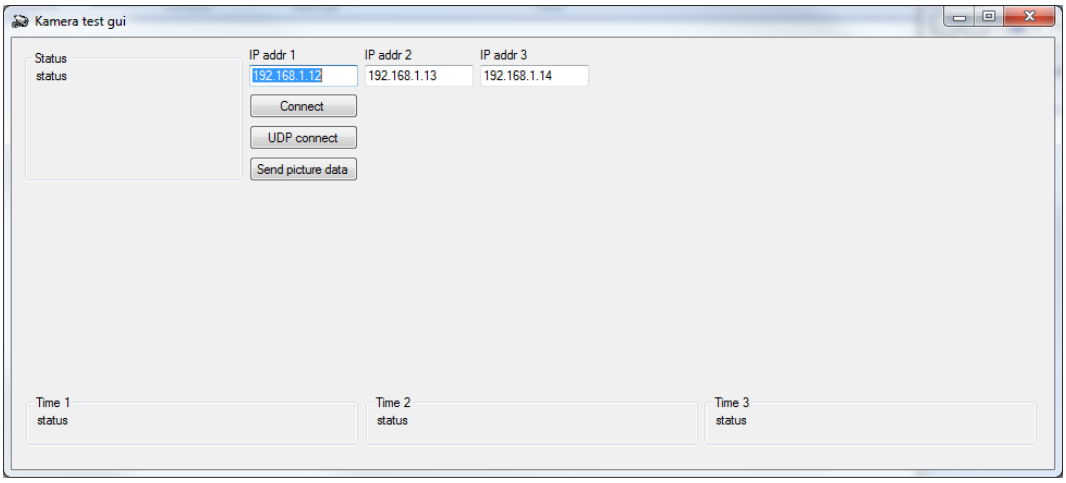
11.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní bylo vytvořeno pro ověření funkčnosti ovladače. Na obr. 11.1 je jednoduché rozhraní pro komunikaci s jednou kamerovou jednotkou pro test rychlosti odesílání dat.

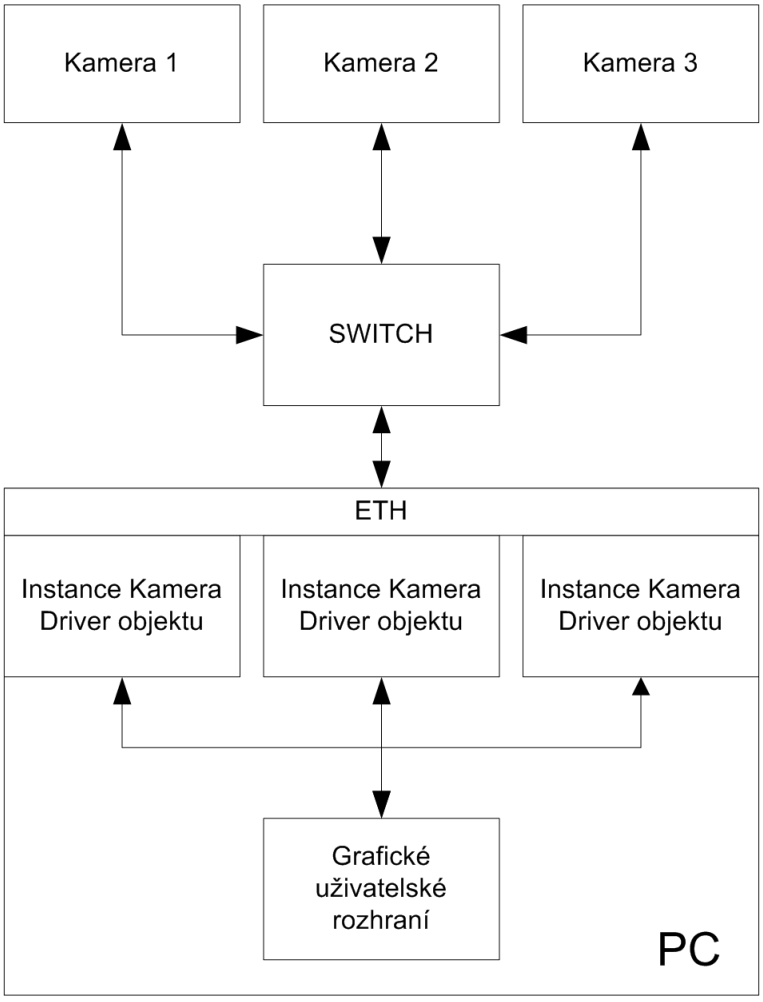
Grafické rozhraní pro více kamer je na obr. 11.2. Na obr. 11.3 je naznačena principiální práce ovladače při komunikaci s kamerami.



Obr. 11.1: Jednoduchý program pro vyčítání obrazových dat z kamery



Obr. 11.2: Grafické uživatelské rozhraní pro test kamer

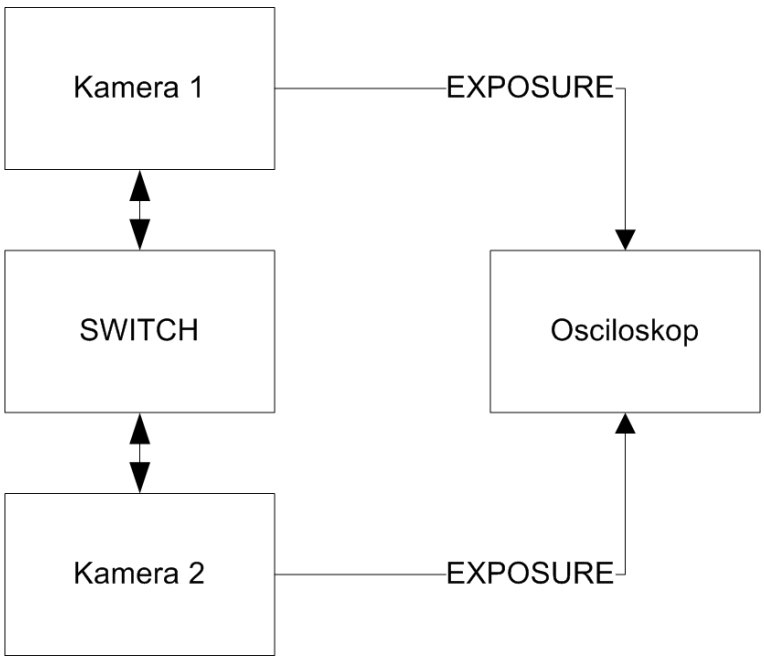


Obr. 11.3: Naznačení způsobu komunikace ovladače s jednotlivými kamerami

Kapitola 12

Testování synchronnosti snímačů a rychlosti vyčítání dat

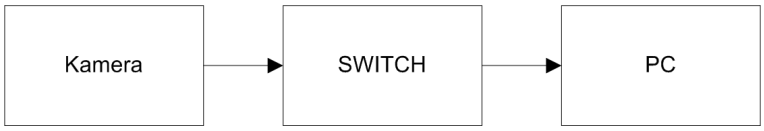
Synchronnost snímačů byla měřena pomocí osciloskopu, kdy oba snímače obdélníkový signál ve stejný nastavený čas. Uspořádání měření je na obr. 12.1, kde byly kamery nastaveny pro generování obdélníkového signálu na pinu EXPOSURE s periodou 2s a střídou 50%. Z vypozorovaných dat nelze odvodit jednoznačnou hodnotu na kterou se jednotky synchronizovaly, jelikož pro každou jednotky byly hodnoty jiné. Jde však určit nejhorší čas na který jsou jednotky schopny se synchronizovat. Rozdíl v časech jednotek nikdy nepřekročil hodnotu jedné mikrosekundy.



Obr. 12.1: Uspořádání měření synchronnosti snímačů

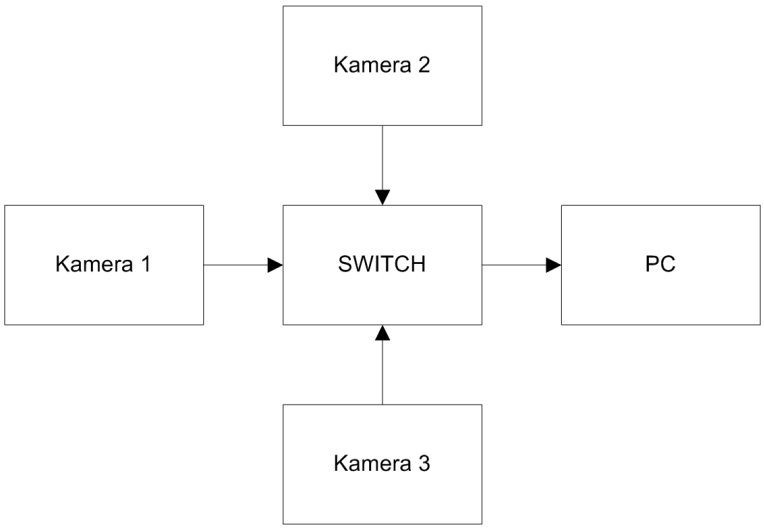
Dalším důležitým krokem byl test rychlosti přenosu dat, obr. 12.2. Zde byly provedeny dva testy, nejprve test pro přenosy dat pouze z paměti RAM mikrokontroleru. Maximální rychlost přenosu byla změřena pomocí programu Wireshark. Zde program naměřil rychlost vysílání dat na 11MB/s, což je očekávaná hodnota vzhledem k nevytíženosti procesoru ostatními periferii bez nutnosti transportu obrázku z obrazového senzoru.

Při testu rychlosti přenosu obrázku bylo dosaženo hodnoty 2MB/s, kdy je možné přenést až 5 obrázků za 1s. Při pokusech o zrychlování docházelo ke ztrátám dat při přenosu. Nakonec



Obr. 12.2: Naznačení způsobu komunikace ovladače s jednotlivými kamerami

byl vytvořen jednoduchý demonstrační program pro demonstraci práce synchronních kamer, obr. 12.3.



Obr. 12.3: Naznačení způsobu komunikace ovladače s jednotlivými kamerami

Kapitola 13

Závěr

Cílem práce bylo vytvořit několik synchronních obrazových snímačů s rozhraním Ethernet. Nejprve bylo třeba navrhnout a vyrobit desky plošných spojů. Byly tedy navrženy univerzální desky plošných spojů pro připojení obrazového senzoru, možnosti připojení k Ethernetu a dalším periferiím. Jako procesor je možné na tyto desky osadit jak mikrokontroler STM32F207 tak i novější mikrokontroler s podporou DSP instrukcí STM32F407.

Vzhledem k nedostatečnému množství řadičů fyzické vrstvy byla navržena a vyrobena další verze desek pro řadiče DP83848.

Pro mikrokontrolery byl vytvořen ovládací program umožňující komunikaci pomocí Ethernetu a umožňující rychlé odeslání naměřených dat.

Pro příjem dat na PC byl vytvořen univerzální ovladač, umožňující navázat komunikaci s kamerou pomocí sítě Ethernet.

Nakonec byl vytvořen jednoduchý demonstrační program demonstrující synchronní snímání kamer v rámci rychlosti, kterou dokáže kamera data přenášet do PC.

Literatura

[1] Fischer, Jan.*Optoelektronické senzory a videometrie.*
Praha: ČVUT, 2002, 143s. ISBN 80-01-02525-X.

[2] YIU, Joseph.*Definitive guide to the ARM Cortex-M3.*
Boston: Newnes, c2007, 359 s. ISBN 978-075-0685-344.

[3] ZÁHLAVA, Vít.*Návrh a konstrukce desek plošných spojů: principy a pravidla praktického návrhu.*
1. vyd. Praha: BEN - technická literatura, 2010, 123 s. ISBN 978-80-7300-266-4.

[4] STMICROELECTRONICS. *RM0033 Reference manual:*
STM32F205xx, STM32F207xx, STM32F215xx and STM32F217xx advanced ARM-based 32-bit MCUs [online]. Rev 3. 1317 s. [cit. 2012-02-20]. Dostupné z: `http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/REFERENCE_MANUAL/CD00225773.pdf`

[5] STMICROELECTRONICS. *RM0090 Reference manual:*
STM32F405xx, STM32F407xx, STM32F415xx and STM32F417xx advanced ARM-based 32-bit MCUs [online]. Rev. 1. 2012, 1316 s. [cit. 2012-05-09]. Dostupné z: `http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/REFERENCE_MANUAL/DM00031020.pdf`

[6] STMICROELECTRONICS. *Datasheet STM32F205xx, STM32F207xx:*
[online]. Rev. 7. 2011 [cit. 2012-05-09]. Dostupné z: `http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00237391.pdf`

[7] STMICROELECTRONICS. *Datasheet STM32F415xx, STM32F417xx:* [online]. Rev. 2. 2012
[cit. 2012-05-09]. Dostupné z: `http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATUR/EDATASHEET/DM00035129.pdf`

[8] STMICROELECTRONICS. *Datasheet ST802RT1A, ST802RT1B:* [online]. Rev. 1. 2010
[cit. 2012-05-09]. Dostupné z: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00263864.pdf

[9] National Semiconductor. *Datasheet DP83848C:* [online]. Rev. 1. 2008
[cit. 2012-05-09]. Dostupné z: http://www.ti.com/lit/ds/symlink/dp83848c.pdf

[10] STMICROELECTRONICS. *AN3320 Application note:* Getting started with STM32F20xxx/21xxx MCU hardware development [online]. Rev. 2. 2011
[cit. 2012-05-09]. Dostupné z: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/APPLICATION_NOTE/CD00292095.pdf

[11] STMICROELECTRONICS. *Errata sheet STM32F20x and STM32F21x:* [online]. Rev. 2. 2011
[cit. 2012-05-09]. Dostupné z: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/ERRATA_SHEET/DM00027213.pdf

[12] Bařtipán, Adam. *Moduly s rozhraním ethernet pro systém sběru dat:* [online]. 2011
[cit. 2012-05-09]. Dostupné z: http://measure.feld.cvut.cz/cs/system/files/files/cs/vyuka/zaverecne_prace/BP_2011_Bartipan_locked.pdf

[13] Adam Dunkels. *Design and Implementation of the lwIP TCP/IP Stack:* [online]. Rev. 2. 2010
[cit. 2012-05-09]. Dostupné z: http://www.es.sdu.edu.cn/project/doc/Design%20and%20Implementation%20of%20the%20lwIP%20tcpIP%20stack.pdf

[14] MICRON. *Datasheet MT9V032:* [online]. Rev. B. 2006
[cit. 2012-05-09]. Dostupné z: http://www.aptina.com/assets/downloadDocument.do?id=78

[15] APTINA. *Datasheet MT9V034:* [online]. Rev. C. 2008
[cit. 2012-05-09]. Dostupné z: http://www.aptina.com/assets/downloadDocument.do?id=741

[16] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems :*
IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), vol., no., pp.c1-269, July 24 2008

[17] Postel, J . *RFC 768:* User Datagram Protocol [online]. 1980
[cit. 2012-05-09]. Dostupné z: http://tools.ietf.org/html/rfc768

[18] Information Sciences Institute University of Southern California. *RFC 791:* INTERNET PROTOCOL [online]. 1981
[cit. 2012-05-09]. Dostupné z: http://tools.ietf.org/html/rfc791

[19] IEEE Computer Society. *IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements :*
[online]. 2008. [cit. 2012-05-09]. Dostupné z: http://standards.ieee.org/getieee802/download/802.3-2008_section1.pdf

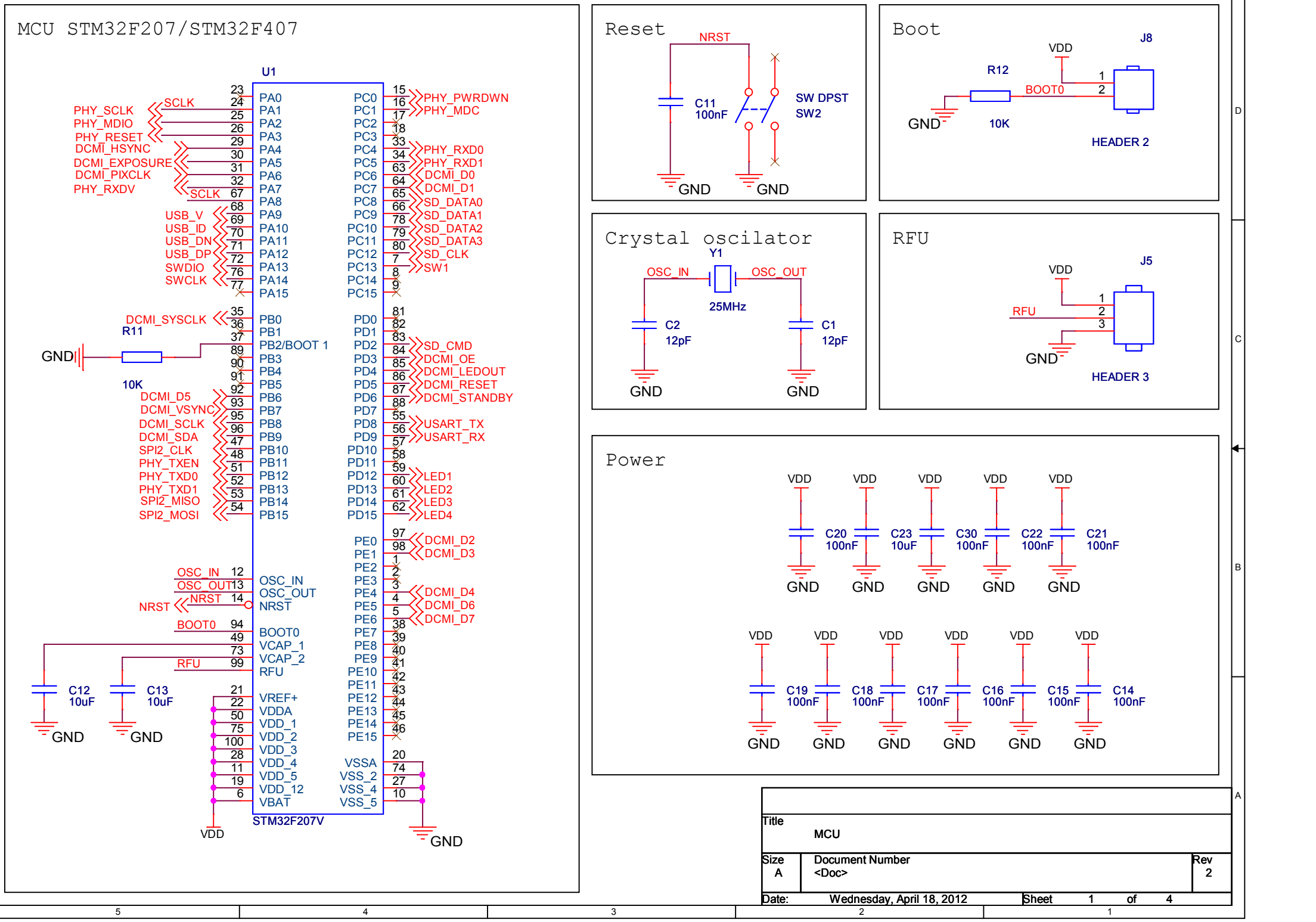
[20] SD Group. *SD Specifications:*
Part 1 Physical Layer Simplified Specification : [online]. 2010. [cit. 2012-05-09]. Dostupné z: https://www.sdcard.org/downloads/pls/simplified_specs/Part_1_Physical_Layer_Simplified_Specification_Ver_3.01_Final_100518.pdf

Příloha A

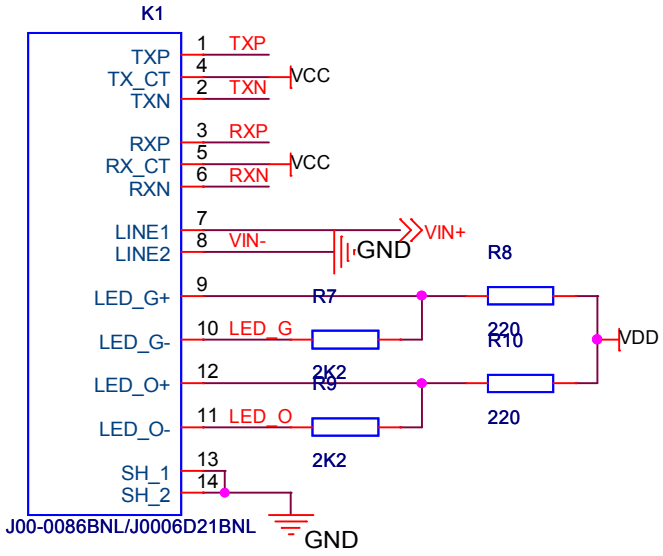
Desky plošných spojů

A.1 Deska plošného spoje V1/2

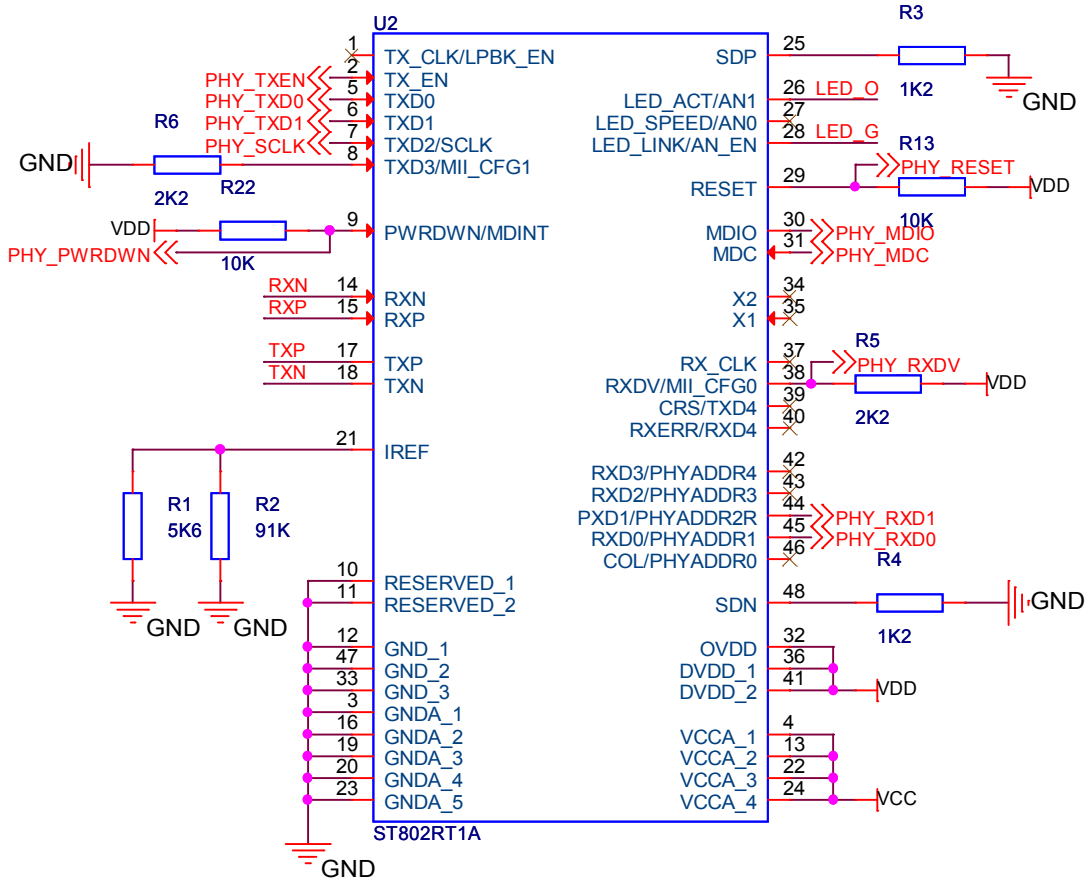
A.1.1 Schéma zapojení


MCU STM32F207/STM32F407
U1
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB2/BOOT 1
PB3
PB4
PB5
PB6
PB7
PB8
PB9
PB10
PB11
PB12
PB13
PB14
PB15
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9
PC10
PC11
PC12
PC13
PC14
PC15
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
PD8
PD9
PD10
PD11
PD12
PD13
PD14
PD15
PE0
PE1
PE2
PE3
PE4
PE5
PE6
PE7
PE8
PE9
PE10
PE11
PE12
PE13
PE14
PE15
OSC_IN
OSC_OUT
NRST
BOOT0
VCAP_1
VCAP_2
RFU
VREF+
VDDA
VDD_1
VDD_2
VDD_3
VDD_4
VDD_5
VDD_12
VBAT
VSSA
VSS_2
VSS_4
VSS_5
STM32F207V
SCLK
PHY_SCLK
PHY_MDIO
PHY_RESET
DCMI_HSYNC
DCMI_EXPOSURE
DCMI_PIXCLK
PHY_RXDV
USB_V
USB_ID
USB_DN
USB_DP
SWDIO
SWCLK
DCMI_SYSCLK
R11
10K
DCMI_D5
DCMI_VSYNC
DCMI_SCLK
DCMI_SDA
SPI2_CLK
PHY_TXEN
PHY_TXD0
PHY_TXD1
SPI2_MISO
SPI2_MOSI
PHY_PWRDWN
PHY_MDC
PHY_RXD0
PHY_RXD1
DCMI_D0
DCMI_D1
SD_DATA0
SD_DATA1
SD_DATA2
SD_DATA3
SD_CLK
SW1
SD_CMD
DCMI_OE
DCMI_LEDOUT
DCMI_RESET
DCMI_STANDBY
USART_TX
USART_RX
LED1
LED2
LED3
LED4
DCMI_D2
DCMI_D3
DCMI_D4
DCMI_D6
DCMI_D7
C12
10uF
C13
10uF
GND
VDD
Reset
NRST
C11
100nF
SW DPST
SW2
Boot
J8
VDD
R12
BOOT0
10K
HEADER 2
Crystal oscilator
Y1
OSC_IN
OSC_OUT
25MHz
C2
12pF
C1
12pF
RFU
J5
HEADER 3
Power
C20
100nF
C23
10uF
C30
100nF
C22
100nF
C21
100nF
C19
100nF
C18
100nF
C17
100nF
C16
100nF
C15
100nF
C14
100nF
Title
MCU
Size
A
Document Number
<Doc>
Rev
2
Date:
Wednesday, April 18, 2012
Sheet
1
of
4

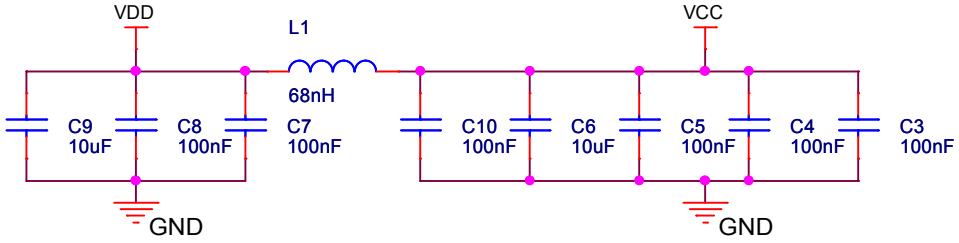
RJ-45 Connector



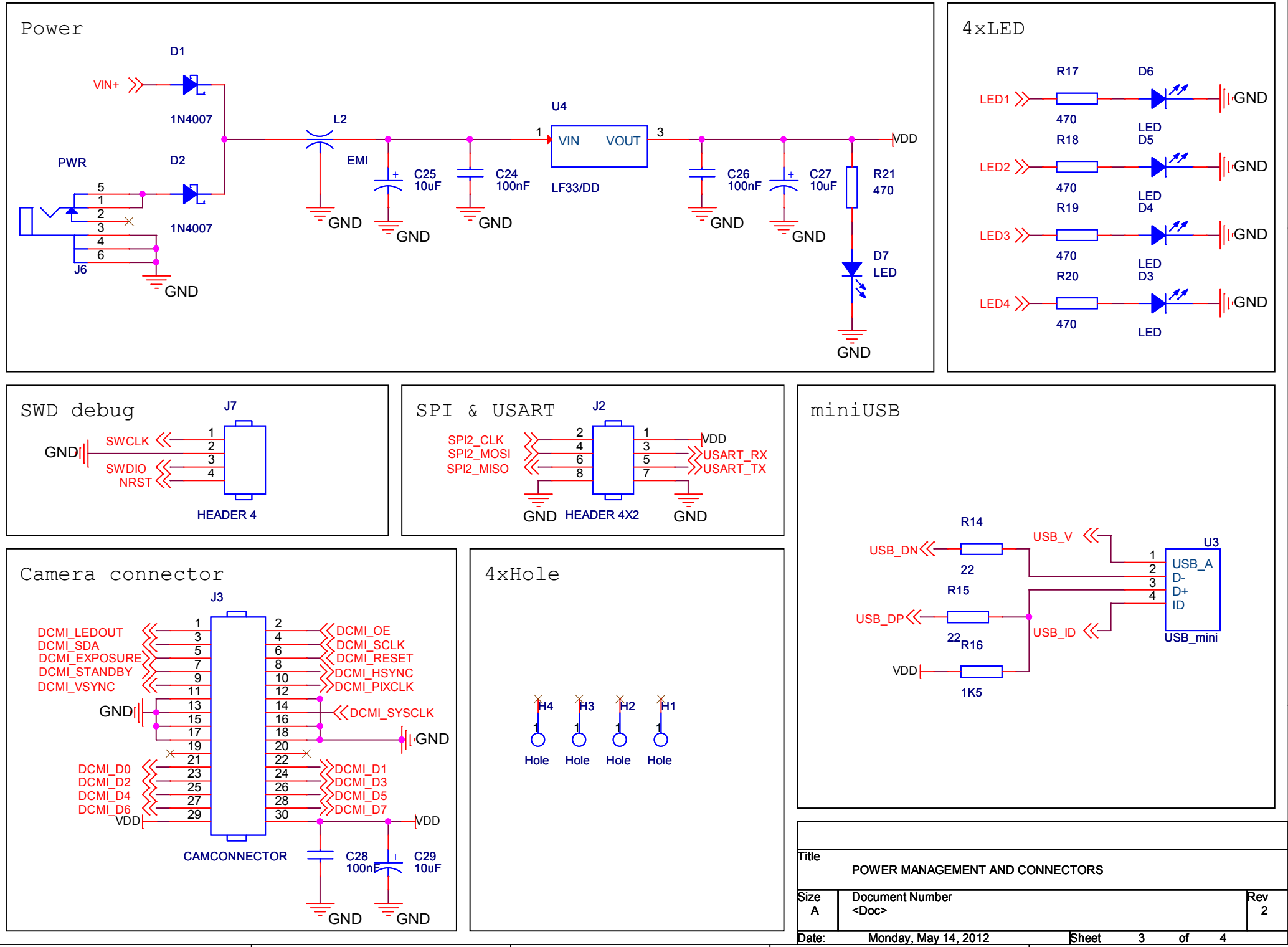
PHY ST802RT1A



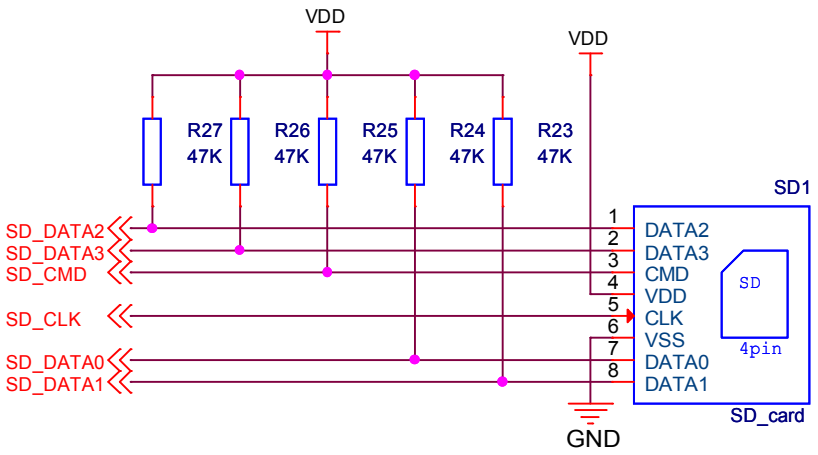
Power



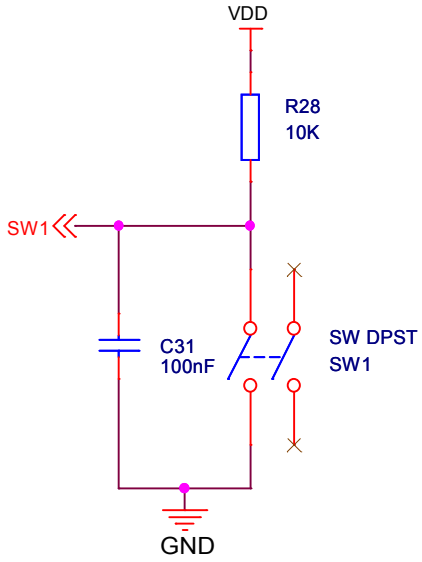
Title	PHY	
Size A	Document Number <Doc>	Rev 2
Date: Wednesday, April 18, 2012	Sheet 2 of 4	


Power
D1
VIN+
1N4007
PWR
D2
1N4007
J6
GND
L2
EMI
GND
C25
10uF
GND
C24
100nF
GND
U4
VIN
VOUT
LF33/DD
C26
100nF
GND
C27
10uF
GND
R21
470
D7
LED
GND
VDD
4xLED
R17
D6
LED1
470
LED
GND
R18
D5
LED2
470
LED
GND
R19
D4
LED3
470
LED
GND
R20
D3
LED4
470
LED
GND
SWD debug
J7
GND
SWCLK
SWDIO
NRST
HEADER 4
SPI & USART
J2
SPI2_CLK
SPI2_MOSI
SPI2_MISO
VDD
USART_RX
USART_TX
GND
HEADER 4X2
GND
miniUSB
R14
USB_DN
22
USB_V
R15
USB_DP
22
R16
VDD
1K5
USB_ID
U3
USB_A
D-
D+
ID
USB_mini
Camera connector
J3
DCMI_LEDOUT
DCMI_SDA
DCMI_EXPOSURE
DCMI_STANDBY
DCMI_VSYNC
DCMI_OE
DCMI_SCLK
DCMI_RESET
DCMI_HSYNC
DCMI_PIXCLK
GND
DCMI_SYSCLK
GND
DCMI_D0
DCMI_D2
DCMI_D4
DCMI_D6
VDD
DCMI_D1
DCMI_D3
DCMI_D5
DCMI_D7
VDD
CAMCONNECTOR
C28
100nF
C29
10uF
GND
GND
4xHole
H4
H3
H2
H1
Hole
Hole
Hole
Hole
Title
POWER MANAGEMENT AND CONNECTORS
Size
A
Document Number
<Doc>
Rev
2
Date:
Monday, May 14, 2012
Sheet
3
of
4

microSD

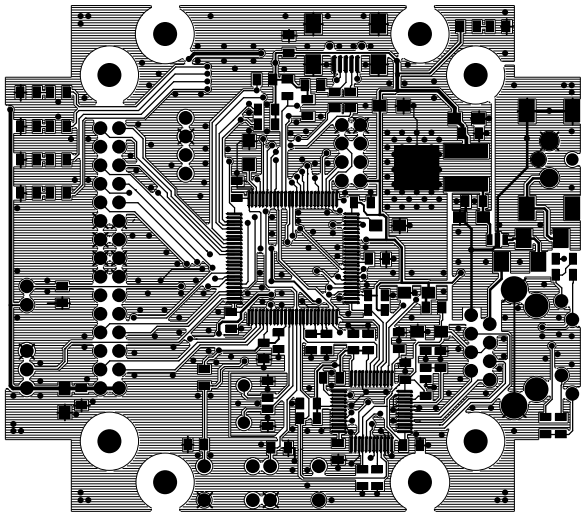


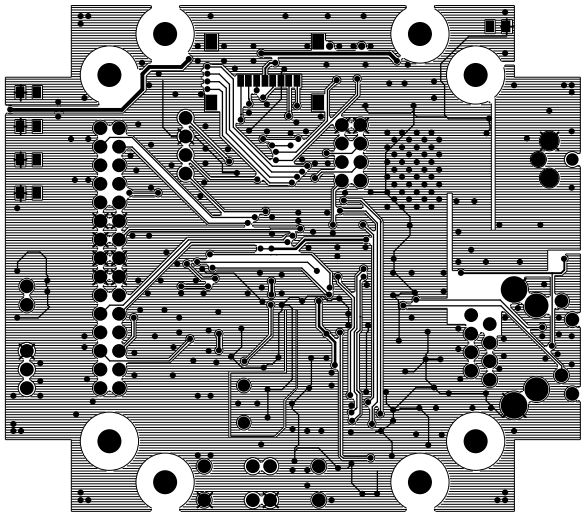
SW1



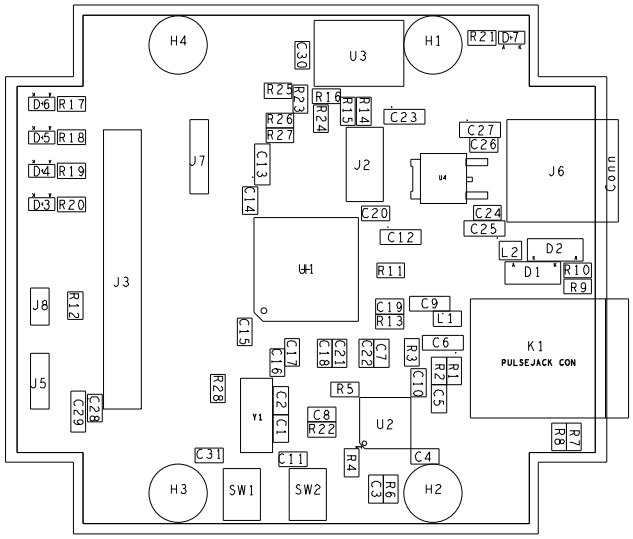
Title	microSD card adapter	
Size A	Document Number <Doc>	Rev v2
Date: Wednesday, April 18, 2012	Sheet 3 of 4	

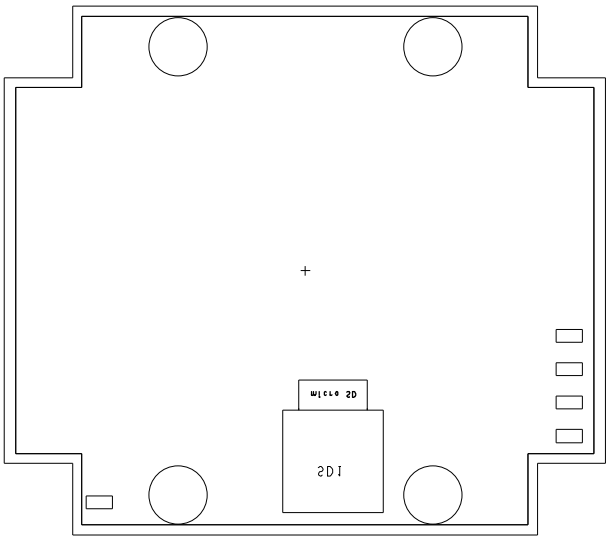
A.1.2 Desky plošných spojů





A.1.3 Osazovací plán


H4
U3
H1
R21
D7
C30
R25
R23
R16
R14
R15
R24
C23
C27
C26
D6
R17
D5
R18
D4
R19
D3
R20
J7
R26
R27
C13
C14
J2
U4
J6
Conn
C20
C24
C25
C12
L2
D2
D1
R10
R9
J3
U1
R11
J8
R12
C19
R13
C9
L1
C15
C17
C16
C21
C18
C7
C22
R3
C6
R1
R2
C5
C10
K1
PULSEJACK CON
J5
R28
C29
C28
Y1
C2
C1
R5
C8
R22
U2
R7
R8
C31
C11
R4
C4
H3
SW1
SW2
R6
C3
H2


SD1

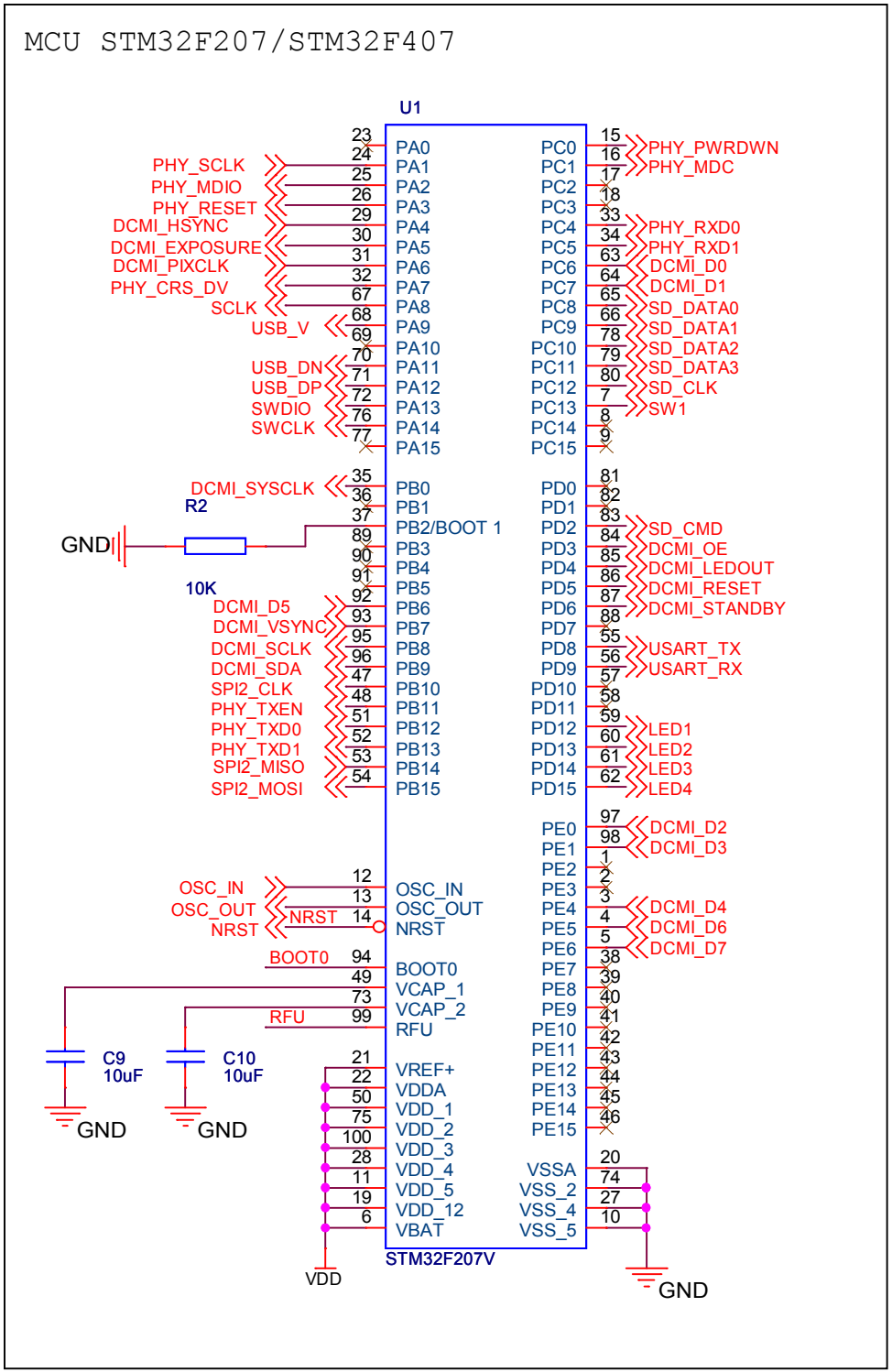
A.1.4 Rozpiska součástek

Položka	Množství	Reference	Hodnota	Pouzdro	Typ
1	2	C1, C2	12pF	805	
2	20	C3, C4, C5, C7, C8, C10, C11, C15, C16, C17, C18, C19, C20, C21, C22, C24, C26, C28, C30, C31	100nF	805	
3	8	C6, C9, C12, C13, C23, C25, C27, C29	10uF	A	
4	2	D1, D2	1N4007		
5	5	D3, D4, D5, D6, D7	LED	805	
6	1	J2	HEADER 4X2	header 4x2	
7	1	J3	CAMCONNECTOR	header 15x2	
8	1	J5	HEADER 3	header 3x1	
9	1	J6	PWR		Farnell: 1243245
10	1	J7	HEADER 4	header 4x1	
11	1	J8	HEADER 2	header 2x1	

Položka	Množství	Reference	Hodnota	Pouzdro	Typ
12	1	K1	J00-0086BNL J0006D21BNL		
13	1	L1	68nH	805	
14	1	L2	EMI		Farnell: 1686518
15	1	R1	5K6	805	
16	1	R2	91K	805	
17	2	R3, R4	1K2	805	
18	4	R5, R6, R7, R9	2K2	805	
19	2	R8, R10	220R	805	
20	5	R11, R12, R13, R22, R28	10K	805	
21	2	R14, R15	22R	805	
22	1	R16	1K5	805	
23	5	R17, R18, R19, R20, R21	470R	805	
24	5	R23, R24, R25, R26, R27	47K	805	
25	1	SD1	SDB_card		Farnell: 1686452
26	2	SW1, SW2	SW DPST		
27	1	U1	STM32F207V	LQFP100	
28	1	U2	ST802RT1A	LQFP48	
29	1	U3	USB_mini		
30	1	U4	LF33/DD	DPAC	
31	1	Y1	25MHz		

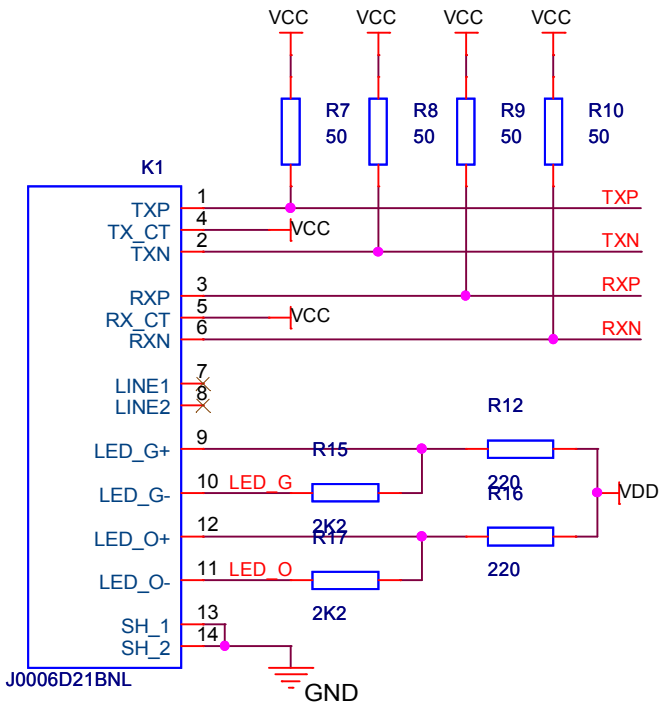
A.2 Deska plošného spoje V3

A.2.1 Schéma zapojení

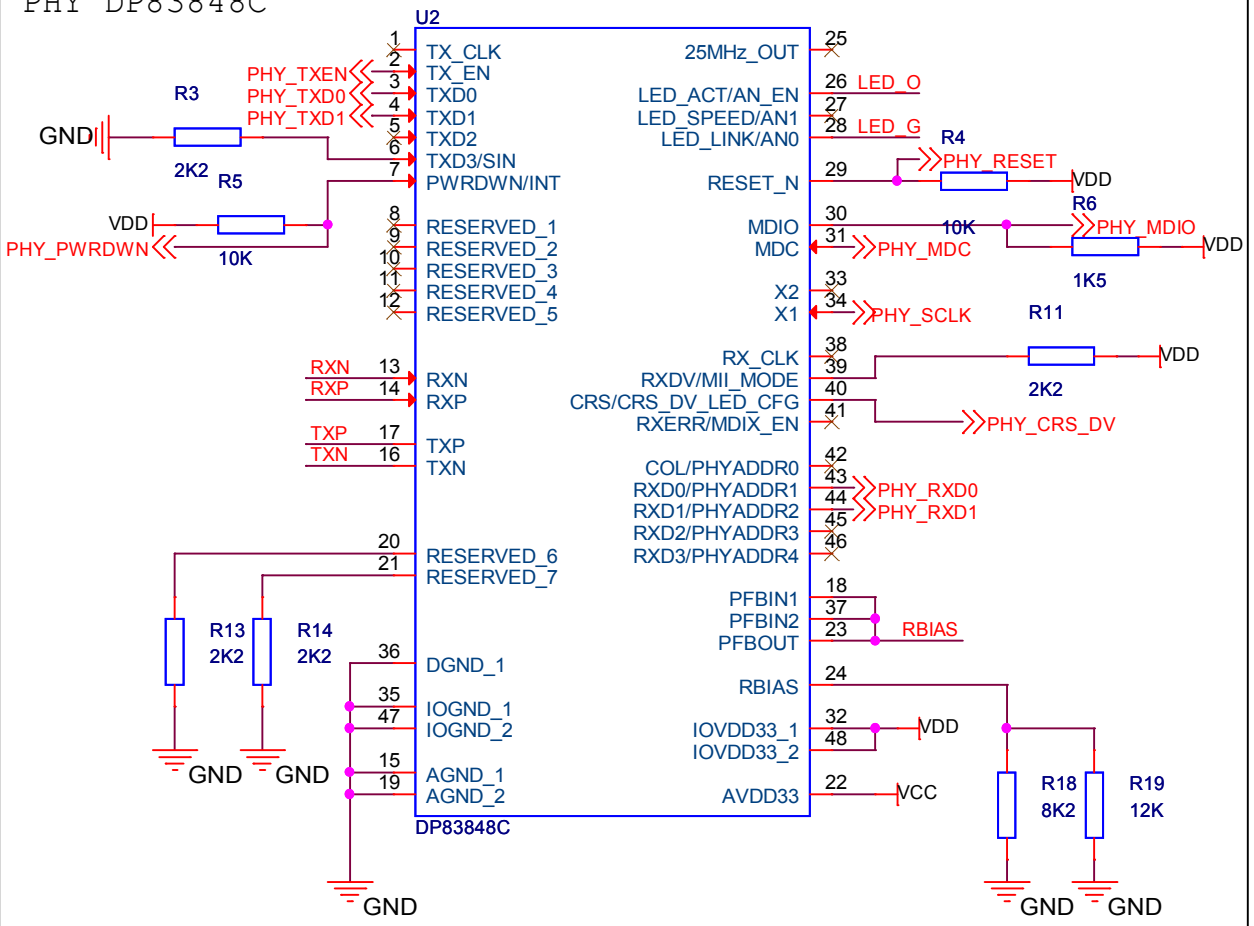


Title	MCU	
Size A	Document Number <Doc>	Rev 2
Date: Sunday, November 11, 2012	Sheet 1 of 4	

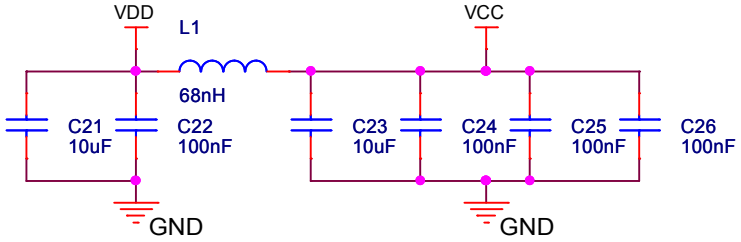
RJ-45 Connector

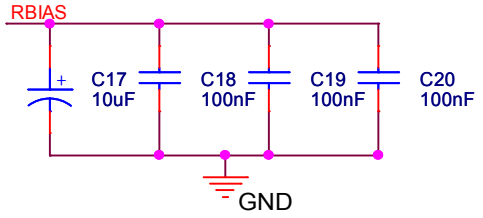


PHY DP83848C



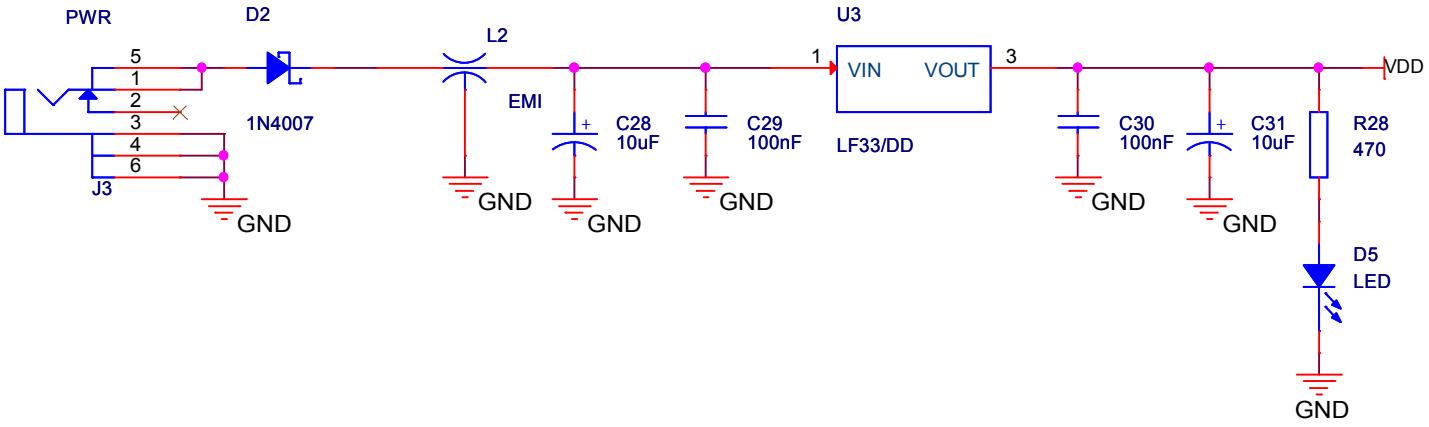
Power



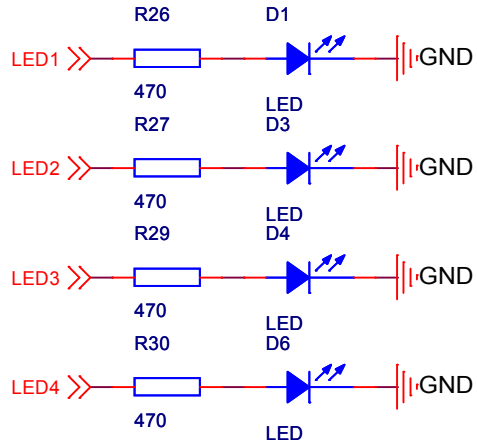


Title	PHY	
Size A	Document Number <Doc>	Rev 2
Date: Tuesday, November 06, 2012	Sheet 2 of 4	

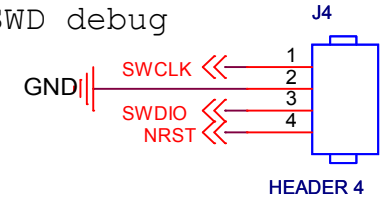
Power



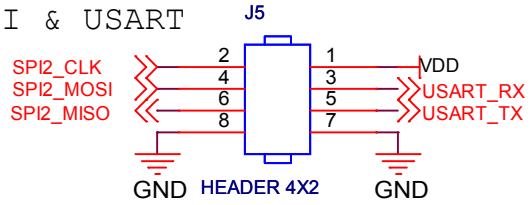
4xLED



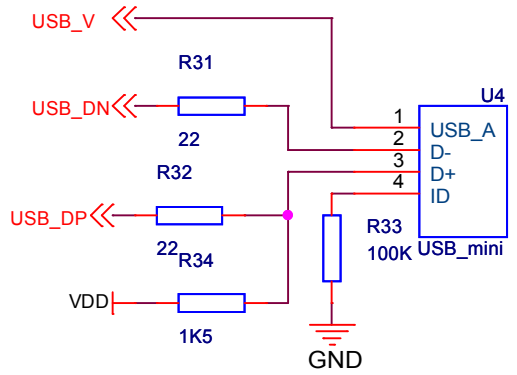
SWD debug



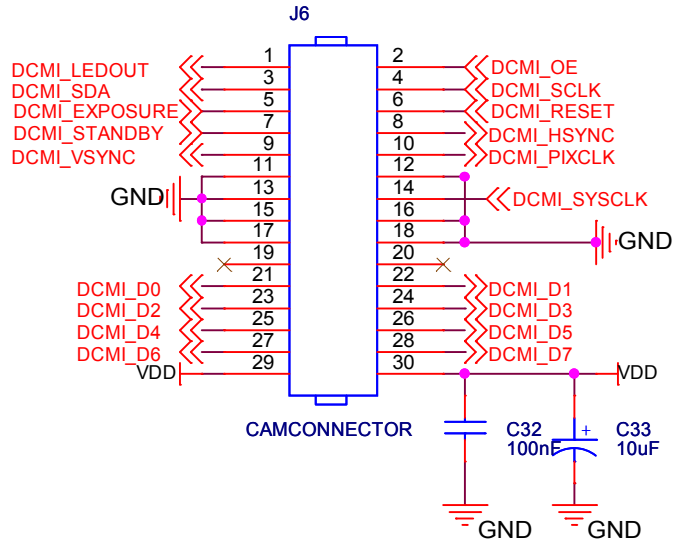
SPI & USART



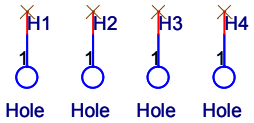
miniUSB



Camera connector

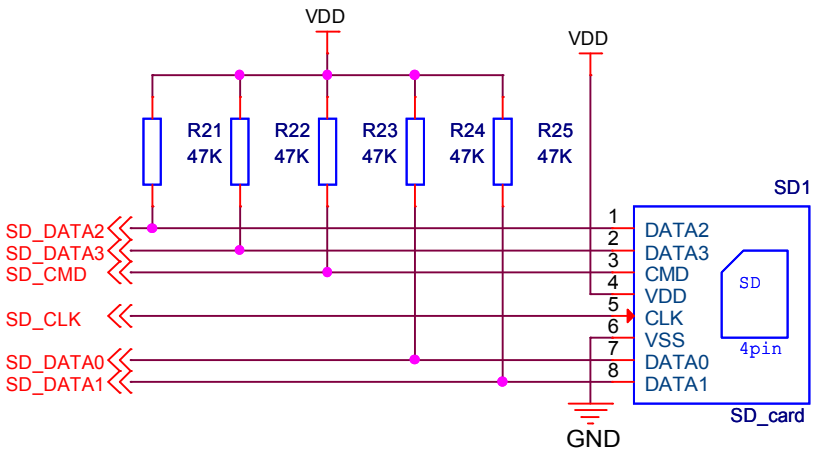


4xHole

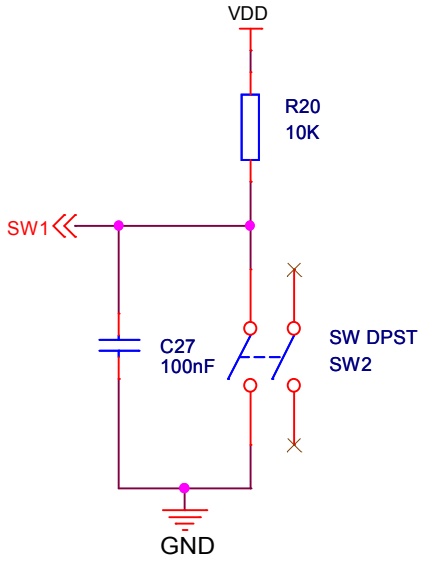


Title	POWER MANAGEMENT AND CONNECTORS	
Size A	Document Number <Doc>	Rev 2
Date: Tuesday, November 06, 2012	Sheet 3 of 4	

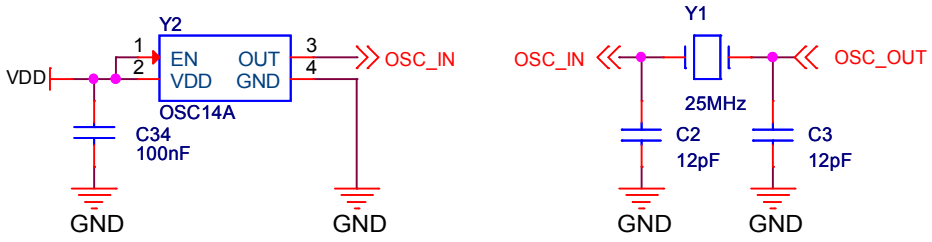
microSD



SW1

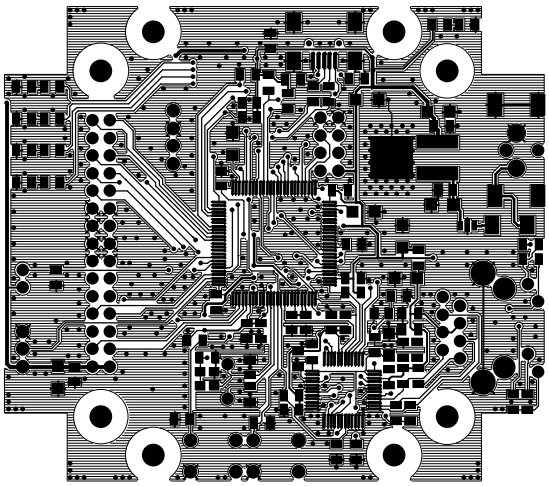


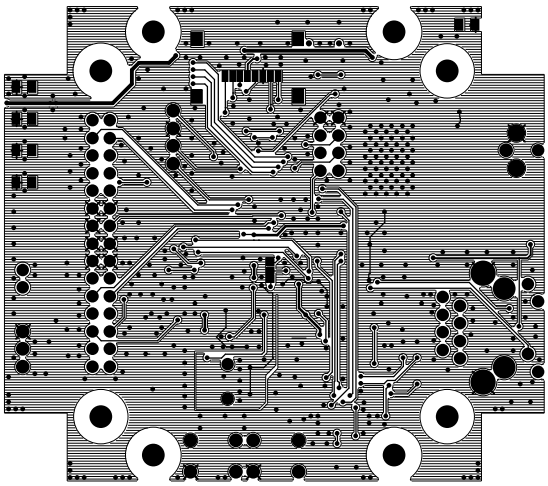
Oscilator/Crystal oscilator



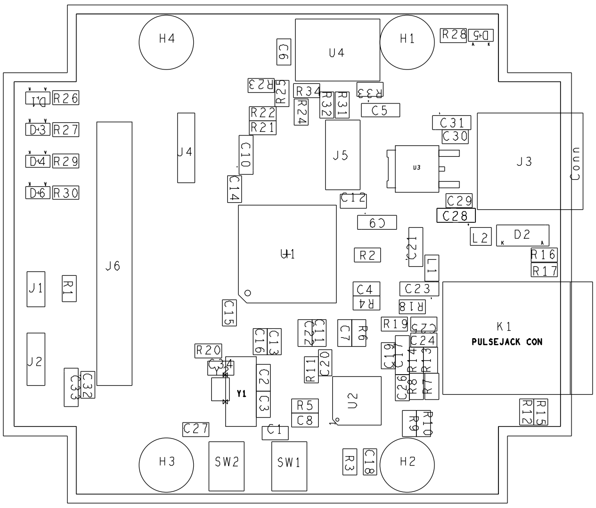
Title: microSD card adapter		
Size: A	Document Number: <Doc>	Rev: v2
Date: Sunday, November 11, 2012	Sheet 3 of 4	

A.2.2 Desky plošných spojů





A.2.3 Osazovací plán


H4
U4
H1
R28
D5
R26
R27
R29
R30
J6
J4
J5
J3
Conn
U1
K1
PULSEJACK CON
J1
R1
J2
Y1
U2
H3
SW2
SW1
H2
D2
L2
L1
R16
R17

SD1
micro SD
BT1

A.2.4 Rozpiska součástek

Položka	Množství	Reference	Hodnota	Pouzdro	Typ
1	23	C1, C4, C6, C7, C8, C11, C12, C13, C14, C15, C16, C18, C19, C20, C22, C24, C25, C26, C27, C29, C30, C32, C34	100nF	805	
2	2	C2, C3	12pF	805	
3	9	C5, C9, C10, C17, C21, C23, C28, C31, C33	10uF	1206	
4	5	D1, D3, D4, D5, D6	LED	805	
5	1	D2	1N4007		
6	4	H1, H2, H3, H4	Hole		
7	1	J1	HEADER 2	Header 2x1	
8	1	J2	HEADER 3	Header 3x1	
9	1	J3	PWR		Farnell: 1243245
10	1	J4	HEADER 4	header 4x1	
11	1	J5	HEADER 4X2		

Položka	Množství	Reference	Hodnota	Pouzdro	Typ
12	1	J6	CAMCONNECTOR	header 15x2	
13	1	K1	J0006D21BNL		
14	1	L1	68nH	805	
15	1	L2	EMI		Farnell: 1686518
16	5	R1, R2, R4, R5, R20	10K	805	
17	6	R3, R11, R13, R14, R15, R17	2K2	805	
18	2	R6, R34	1K5	805	
19	4	R7, R8, R9, R10	50	805	
20	2	R12, R16	220	805	
21	1	R18	8K2	805	
22	1	R19	12K	805	
23	5	R21, R22, R23, R24, R25	47K	805	
24	5	R26, R27, R28, R29, R30	470	805	
25	2	R31, R32	22	805	
26	1	R33	100K	805	
27	1	SB1	SB		
28	1	SD1	SD_card		Farnell: 1686452
29	2	SW1, SW2	SW DPST		
30	1	U1	STM32F207V	LQFP100	
31	1	U2	DP83848C	LQFP48	
32	1	U3	LF33/DD	DPAC	
33	1	U4	USB_mini		
34	1	Y1	25MHz		
35	1	Y2	50MHz smd		Farnell: 2101315

Příloha B

Obsah CD

- Programy pro PC
- Program pro mikrokontroler
- Text diplomové práce
- Dokumenty
- Foto