FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ

ELEKTRONIKY A TELEKOMUNIKACÍ

DIPLOMOVÁ PRÁCE

Příklad aplikace založené na micro. NET frameworku

Autor práce: Ondřej Hála

Vedoucí práce: Ing. Petr Weissar, Ph.D.

Plzeň 2013

Anotace diplomové práce

Hála, O. - Příklad aplikace založené na micro .NET frameworku. Katedra aplikované elektroniky a telekomunikací, Západočeská univerzita v Plzni – Fakulta elektrotechnická, 2013, 58 s., vedoucí: Ing. Petr Weissar, Ph.D.

Diplomová práce obsahuje kompletní návrh embedded zařízení. Zařízení je vybaveno dotykovým LCD displejem, Wi-Fi a Bluetooth modulem, USB rozhraním a audio dekodérem. Návrh desky plošného spoje je uskutečněn v návrhovém systému Altium Designer. Softwarová část je napsaná v jazyce C# pro platformu .NET micro framework v nástroji Microsoft Visual Studio 2010.

Klíčová slova:

Micro .NET Framework, NETMF, software, hardware, audio, Bluetooth, Wi-Fi, LCD, embedded, C#, programování

Abstract

Hála, O. - Application based on Micro.NET Framework. Department of applied electronics and telecommunication, University of West Bohemia in Plzeň – Faculty of electrical engineering, 2013, 58 p., head: Ing. Petr Weissar, Ph.D.

Thesis contains complete design of an embedded system. The device is equipped with LCD with touchscreen, Wi-Fi and Bluetooth modules, USB interface and audio encoder. PCB is designed in Altium Designer. Software part is coded in C# language for .NET Micro Framework platform in Microsoft Visual Studio 2010.

Key words:

Micro .NET Framework, NETMF, software, hardware, audio, Bluetooth, Wi-Fi, LCD, embedded, C#, coding, programming

Prohlášení:

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

v Plzni, dne ……. ……

podpis diplomanta

Poděkování

Mé poděkování patří Ing. Petrovi Weissarovi, Ph.D., za hodnotné rady a odborné vedení mé diplomové práce. Dále pak Ing. Vladimíru Pavlíčkovi, Ph.D., za ochotu a pomoc s pájením integrovaných obvodů.

Poděkování též patří Katedře aplikované elektroniky a telekomunikací za finanční podporu.

Obsah

Seznam použitých zkratek

*API (**A**pplication **P**rogramming **I**nterface)*

*CAN (**C**ontroller **A**rea **N**etwork)*

*CLR (**C**ommon **L**anguage **R**untime)*

*GPIO (**G**eneral **P**urpose **I**nput/**O**utput)*

*NETMF (.**NET** **M**icro **F**ramework)*

*NTP (**N**etwork **T**ime **P**rotocol)*

*PCB (**P**rinted **C**ircuit **B**oard)*

*RLP (**R**untime **L**oadable **P**rocedures)*

*RTC (**R**eal **T**ime **C**lock)*

*SDK (**S**oftware **D**evelopment **K**it)*

*SoM (**S**ystem **o**n **M**odule)*

*SPP (**S**erial **P**ort **P**rofile)*

*SSP (**S**ynchronous **S**erial **P**ort)*

*UI (**U**ser **I**nterface)*

*URL (**U**niform **R**esource **L**ocator)*

*USB (**U**niversal **S**erial **B**us)*

*WPF (**W**indows **P**resentation **F**oundation)*

1. Úvod

Téma diplomové práce je příklad aplikace založený na micro .NET frameworku. Toto téma jsem si vybral, protože jsem se s NETMF setkal během studia na VŠ. Líbila se mi možnost programovat MCU pomocí vyššího programovacího jazyka. Základy v jazyce C# jsem získal během praxe ve firmě Zollner AG, ve které jsem programoval aplikaci pro vizualizaci procesu na zařízení pro lisování konektorů do PCB. Rozhodl jsem se zkombinovat vědomosti o návrhu HW získaných během studia na univerzitě a znalostí programovacího jazyka C# získaných během praxe.

Cílem diplomové práce je vytvořit zařízení, které bude komunikovat se sítí internet a bude ovladatelné LCD dotykovým displejem. Zařízení by mělo prezentovat schopnosti NETMF a to hlavně pokročilé funkce, jako je síťová komunikace, webový server, či připojení USB úložišť.

Stanovil jsem však ještě další požadavky na zařízení. Mezi ně patří audio výstup a to včetně dekódování běžných audio formátů, správa připojení do sítě Wi-Fi, načítání a zobrazování počasí pro vybranou lokalitu a možnost přehrávání webových rádií.

Práce je rozdělena do dvou hlavních kapitol. První kapitola je věnována popisu použitého hardwaru. Druhá kapitola je věnována softwaru, popisuje ovladače pro hardware a základní bloky UI.

2. Hardware

Kapitola obsahuje soupis dostupných modulů podporujících NETMF. U každého modulu jsou uvedeny základní informace. Podrobnější informace jsou k dispozici v on-line katalogu firmy Ghi Electronics[1]. Další část je pak věnována popisu základních vlastností součástek použitých v práci.

2.1 .NET Micro framework

.NET Micro framework, přináší vlastnosti a výhody managed kódu do oblasti programování embedded systémů. Microsoft .NET micro framework SDK umožňuje programování mikrokontrolérů ve vývojovém prostředí Microsoft Visual Studio včetně všech výhod, které tento nástroj nabízí pro vývoj aplikací určených pro PC.

Mezi výhody managed kódu patří:

- automatická správa paměti díky garbage collectoru, dále jen GC,
- vícevláknové aplikace (Multithreading) a jejich bezpečná synchronizace,
- správa výjimek,
- striktní bezpečné datové typy – type safety,
- bezpečný a robustní kód,
- propracovaný debugging.

CLR využívá GC, který automaticky uvolňuje nepoužívané objekty (objekty bez reference) z paměti. GC je spouštěn automaticky ve stanovených časech, ovšem pokud je zjištěn nedostatek volné paměti, je spuštěn okamžitě.

Díky tomu, že managed kód je vykonáván pod kontrolou CLR, které se stará o referencování objektů a přiřazování paměti, odpadá riziko vzniků problémů s pointery. V managed kódu nelze přímo přistupovat do paměti a využívat pointerovou aritmetiku.

[1] https://www.ghielectronics.com/catalog

Projekt .NET micro framework je veřejně přístupný na vývojářském portále CodePlex[2], na kterém lze nalézt SDK pro MS Visual Studio a porting kit, který umožňuje portaci NETMF na vlastní HW. Porting kit obsahuje RTIP TCP/IP stack od společnosti EBSnet Inc, open-source lwIP TCP/IP stack a distribuci OpenSSL.

NETMF knihovna obsahuje všechny hlavní namespace z „dospělého" frameworku známého z PC a rozšiřuje je o ovladače HW, vzdálenou aktualizaci firmware a kryptografické funkce psané na míru HW. Popis knihoven obsažených v NETMF SDK je k dispozici na serveru MSDN[3] včetně ukázkových příkladů.

2.2 MCU s portací .NETMF

Napsat portaci pro nové MCU je složité a vyžaduje kompletní znalosti portovaného MCU i samotného Porting Kitu a jeho závislostí. Díky tomu existuje jen několik firem, které se tvorbou těchto portů zabývají. Některé porty mají otevřený kód, ale z většiny jsou uzavřené. Příkladem open-source portu může být port pro MCU STM32F4, či Fez Hydra.

Ghi Electronics je společnost s nejrozsáhlejší nabídkou NETMF portovaných zařízení. Navíc ke každému zařízení nabízejí knihovny, které urychlují vývoj. K open-source zařízením nabízejí knihovny GHI.OSHW (Open Source Hardware). Knihovny obsahují třídy:

- *LCDController*
  - slouží k nastavení zabudovaného LCD řadiče,
- *RTC*
  - třída pro práci s obvodem reálného času,
- *Software I2C*
  - umožňuje vytvořit softwarové I^2C na libovolném páru GPIO,
- *StorageDev*
  - třída pro podporu SD karty,

[2] http://netmf.codeplex.com/

[3] http://msdn.microsoft.com/en-us/library/ee435793.aspx

- *LowLevel*
  - slouží k přímému přístupu k registrům MCU,
- *RLP*
  - umožňuje vyvolávat funkce v nativním kódu z managed kódu.

Dokumentace ke knihovně je k dispozici na webu Ghi Electronics[4] pod záložkou Support.

Mnohem rozsáhlejší knihovny *GHI.Premium*, které již nejsou určeny pro open-source, ale pouze pro moduly zakoupené od společnosti Ghi Electronics, obsahují třídy:

- *CAN*
  - třída slouží pro nastavení a ovládání řadiče CAN,
- *LCD*
  - slouží k nastavení parametrů zabudovaného LCD řadiče,
- *EMX / G120*
  - obsahuje výčet všech GPIO pro zvolený modul,
- *ParallelPort*
  - umožňuje využití 8 datových + 2 řídících GPIO jako paralelní port,
- *RTC*
  - třída pro práci s obvodem reálného času,
- *SignalCapture*, *SignalGenerator*
  - využívá čítačů na určených pinech k měření délky pulsů a generování průběhů,
- *SoftwareI2C*
  - umožňuje vytvořit softwarové I^2C na libovolném páru GPIO,
- *PowerClass*
  - třída umožňuje řízení přechodu do režimu s nízkou spotřebou, možnost probuzení z RTC,

[4] http://www.ghielectronics.com/

- *RegisterClass*
  - slouží k přímému přístupu k registrům MCU,
- *Watchdog,*
- *Persistent Storage*
  - třída pro podporu SD karty a USB paměťových médií,
- *Native – RLP*
  - umožňuje vyvolávat funkce v nativním kódu,
- *Ethernet* – built-in, ENC28J60, WiFiRS9110
  - umožňuje využití Wi-Fi modulu, dále pak periferie ethernet u EMX modulu a připojení externího MAC a PHY ENC28J60 u G120 modulu,
- *SQL Lite*
  - umožňuje vytvořit a spravovat databázi SQL Lite a to buď v paměti RAM, nebo na paměťovém médiu,
- *CRC, Math, XTEA*
  - rozšiřuje základní matematické funkce, přidává funkce CRC a šifru XTEA,
- *SystemUpdate*
  - třída slouží pro vzdálený update firmware v zařízení,
- *USB Host a USB Client*
  - umožňuje komunikovat s USB zařízeními.

2.2.1 System On Module (SoM)

Jedná se o produkty určené k návrhu zařízení s vlastním HW. Modulem se rozumí MCU na PCB doplněné o externí paměť a další periferie. Firma Ghi Electronics nabízí moduly popsané v této kapitole.

EMX Module

Obr. 2.1: Ghi Electronics EMX [6]

- LPC2478 ARM 72MHz
- 4,5MB paměť flash / 16MB RAM
- LCD řadič
- USB Host / USB Client, Ethernet PHY
- rozhraní pro SD kartu
- 3 x SPI, I2C, 2 x CAN
- GHI Premium knihovny

Cena při objednávce jednoho kusu k datu 6. 5. 2013 je $84,95.

G120 Module

Obr. 2.2: Ghi Electronics G120 [6]

- LPC1788 Cortex M3 120MHz
- 4,5MB paměť flash / 16MB RAM
- LCD řadič
- USB Host / USB Client
- rozhraní pro SD kartu
- 3 x SPI, I2C, 2 x CAN
- GHI Premium knihovny

Cena při objednávce jednoho kusu k datu 6. 5. 2013 je $42.49.

G400-D Module

Obr. 2.3: Ghi Electronics G400D [6]

- Atmel SAM9X35
- 4MB paměť flash / 128MB RAM
- LCD řadič
- USB Host / USB Client
- rozhraní pro SD kartu
- 2 x SPI, I2C, 2 x CAN, 4 x UART

Tento modul je nejvýkonnější z nabídky Ghi Electronics. Cena při objednávce jednoho kusu není stanovena, protože produkt je ve vývoji. Má být však levnější než modul G120.

2.2.2 .NET Gatgeteer

Gatgeteer je platforma pro vývoj vlastních elektronických zařízení, která jsou založena na MCU s NETMF. Cílem je co nejvíce usnadnit návrh HW i SW a tak urychlit realizaci svých myšlenek. Prodává se velké množství modulů kompatibilních s Gatgeteer. Jejich nabídka je velice pestrá. Součástí je „klikací“ software do Visual Studia, v kterém se moduly pospojují a vygeneruje se základní kód.

Obr. 2.4: Ukázka modulů standartu Gatgeteer [6]

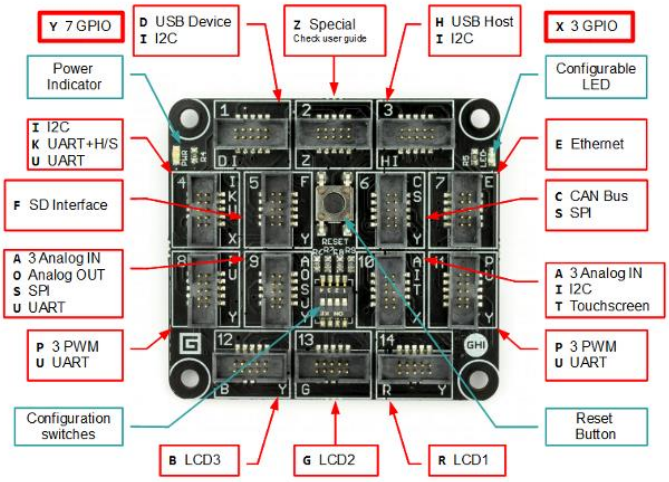
Konektory jsou označeny písmenem podle toho, jaký typ modulu podporují. Pin-out takto značených konektorů lze nalézt na oficiální stránce NETMF Gatgeteer[5].

Příkladem Gatgeteer modulů mohou být:

- ethernet, Wi-Fi,
- USB Host, kamera, SD karta,
- budiče RS-232, ODB-II, CAN,
- akcelerometr, barometr, kompas,
- GPS, RFID, XBee, GSM,
- budič motorů, H můstek.

Fez Spider Mainboard

PCB s osazeným EMX modulem a Gatgeteer konektory. EMX modul byl představen v kapitole 2.2.1 System On Module.

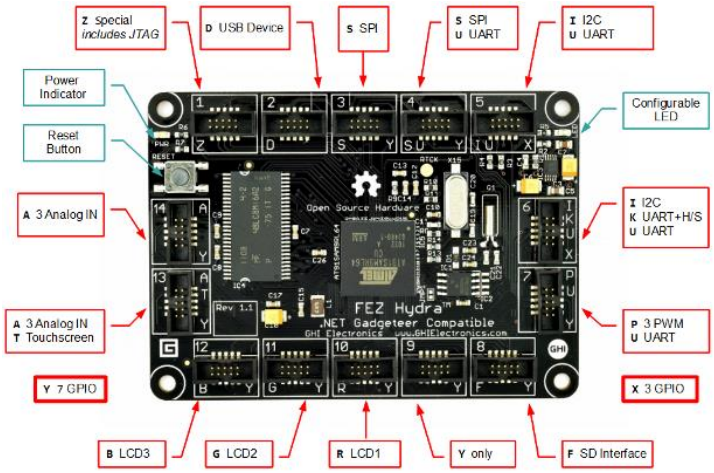


Obr. 2.5: Ghi Electronics Fez Spider [6]

[5] http://www.netmf.com/gadgeteer/

FEZ Hydra

Nejvýkonnější modul na PCB z nabídky Ghi Electronics. Pro použité MCU je k dispozici i port Linuxu. Hardware je open source.

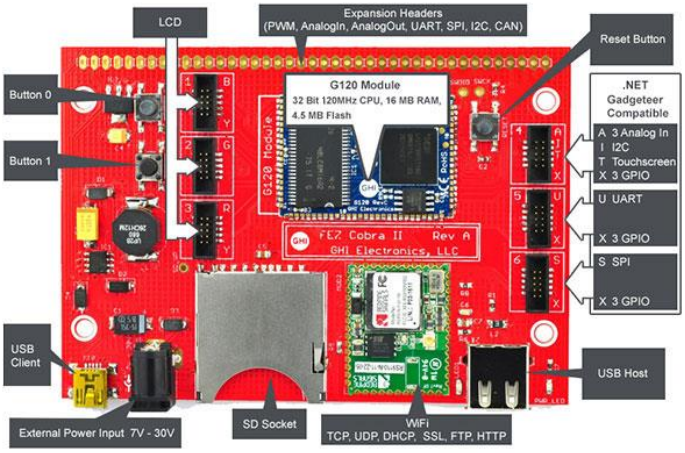


Obr. 2.6: Ghi Electronics Fez Hydra [6]

- 240Mhz ARM9
- 4MB paměť flash / 16MB RAM
- LCD řadič
- všechny další zařízení dostupné přes gatgeteer konektory

FEZ Cobra II

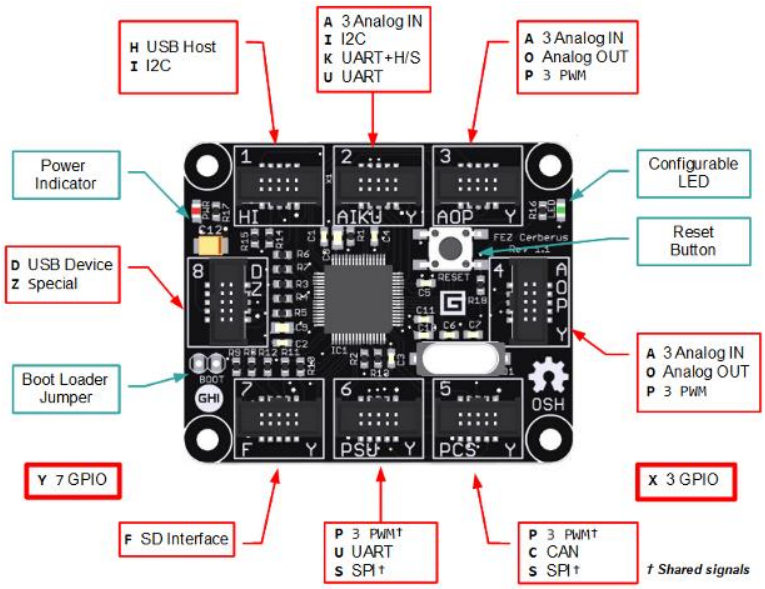
G120 modul umístěný na PCB. Tento produkt sloužil jako referenční design pro diplomovou práci.



Obr. 2.7: Ghi Electronics Fez Cobra II ve verzi s Wi-Fi [6]

Fez Cerberus

Nejlevnější Gatgeteer mainboard v nabídce. Port pro STM32F4 je open-source. K tomuto MCU lze navrhnout vlastní HW. MCU tedy není nutné zakupovat od společnosti Ghi Electronics, z čehož plyne jeho vhodnost pro sériovou výrobu.



Obr. 2.8: Ghi Electronics Fez Cerberus [6]

- 178MHz STM32F4 Cortex M4
- 1MB paměť flash / 192kB RAM
- RLPLite

2.3 Použitý hardware

V této kapitole jsou popsané HW moduly obsažené v práci. Z důvodu předepsaného rozsahu práce je tato kapitola spíše přehledová. Ke každému modulu je k dispozici datasheet od výrobce. Datasheety jsou k dispozici v příloze na CD ve složce */dokumentace/*.

2.3.1 MCU

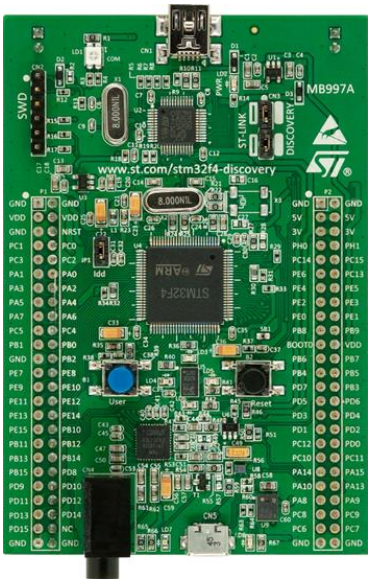
Mezi kandidáty jsem zařadil STM32F4 Disovery Kit a modul G120. V této kapitole popisuji jejich základní vlastnosti.

Jako MCU jsem nakonec zvolil modul G120. Splňoval všechny požadavky návrhu a jeho cena oproti konkurentům byla nejnižší. Proti však hovořil fakt, že se jedná o nový produkt a jeho firmware byl ještě ve fázi vývoje.

STM32F4 Discovery Kit

Vývojový kit od společnosti STMicroelectronics. Projekt portace NETMF na MCU STM32F4 je dostupný na stránkách CodePlex[6].

[6] http://netmf4stm32.codeplex.com/



Obr. 2.9: ST Microelectronics STM32F4 Discovery Kit [7]

- ST-LINK/V2 debugger na PCB
- 1MB Flash, 192kB RAM
- LS302DL 3osý akcelerometr
- MP45DT02 MEMS mikrofon
- CS43L22 Audio DAC se sluchátkovým zesilovačem

Kit jsem si zapůjčil na Katedře aplikované elektroniky a telekomunikací. Postup pro nahrání NETMF firmware je dostupný na stránkách projektu. Nahrání je možné například použitím aplikace STM32 ST-LINK Utility.

Na kitu jsem vyzkoušel komunikaci s akcelerometrem sběrnicí I^2C. Ukázkové kódy jsou přiloženy na CD ve složce */SW/STM32F4/*. Mezi požadavky na HW, které jsem stanovil, bylo dekódování MP3 souborů. Na kitu je však k dispozici pouze DA převodník. Dekódování souborů MP3 by značně vytížilo MCU a tím pádem by i znatelně limitovalo možnosti NETMF. Rozhodl jsem se tedy vydat cestou připojení DSP audio dekodéru.

Gatgeteer modul společnosti Ghi Electronics s označením Music Module byl nejjednodušší cestou. Obslužný kód pro dekodér jsem upravil k použití na kitu STM32F4 Discovery. Ukázkový kód je opět k nalezení v příloze na CD ve složce */SW/STM32F4/*.

Obr. 2.10: Ghi Electronics Music Module [6]

Už při programování tohoto driveru bylo nutností zvažovat připojení externí paměti RAM. Paměť o velikosti 192 kB byla pro NETMF značně nedostačující. Připojení externí paměti by znamenalo velkou úpravu portace. To nebylo cílem diplomové práce, a proto jsem se rozhodl od této varianty ustoupit. Avšak pro méně náročné aplikace je tato varianta naprosto dostačující.

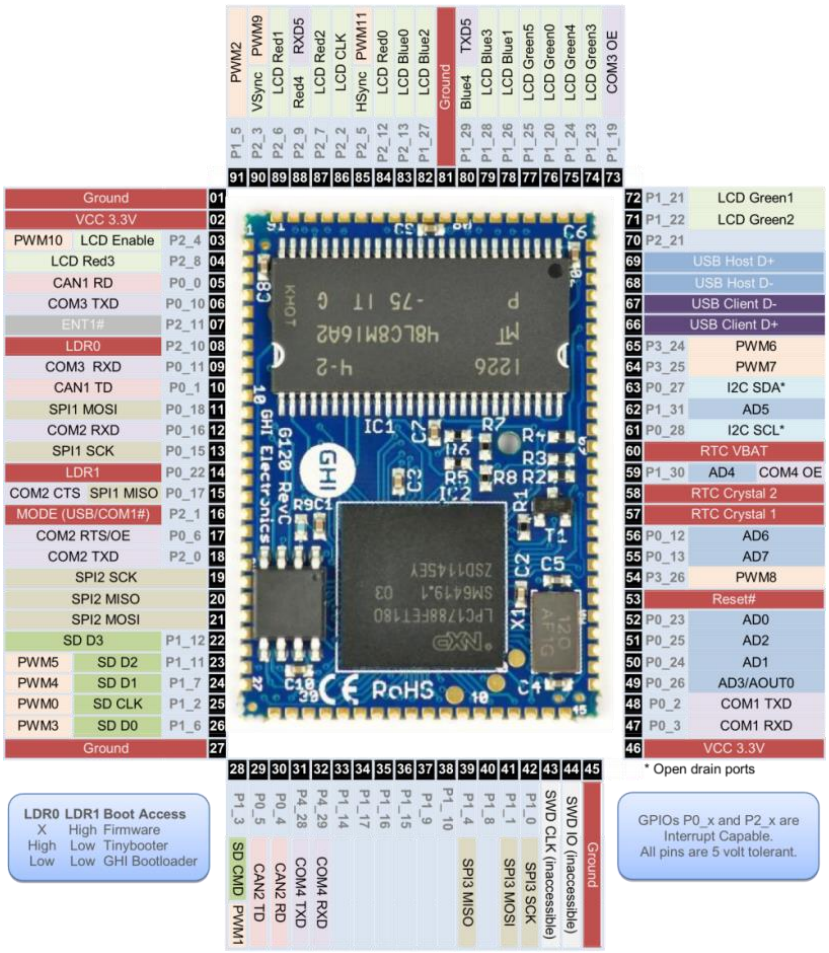
G120

G120 je SoM od společnosti Ghi Electronics určený pro povrchovou montáž. Modul je umístněný na vícevrstvém plošném spoji, což snižuje požadavky na počet vrstev výsledného plošného spoje.

Výhodou G120 modulu nejsou jen jeho HW parametry, jako jsou: 32-bit ARM Cortex-M3 (NXP LPC1788 – 120 MHz) procesor, rozšiřující paměť RAM a periferie, ale také jeho snadná aplikovatelnost a vyměnitelnost ve větších systémech jak z pohledu HW, tak SW.

G120 podporuje všechny funkce knihovny *GHI.Premium*, jako například: práci se souborovým systémem FAT, TCP/IP stack, grafiku, Wi-Fi, USB Host, SQL Lite a další. V neposlední řadě zahrnuje i možnost vzdálené aktualizace. Všechny tyto funkce a mnohé další jsou již zahrnuty v ceně produktu.

Místo dalšího výčtu funkcí raději přikládám pin-out G120 modulu viz Obr. 2.11, z kterého je jasně vidět jakými periferiemi je G120 modul vybaven.



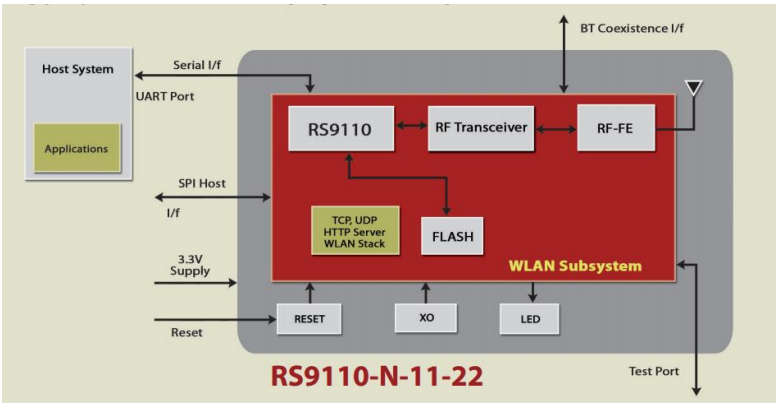
Obr. 2.11: Ghi Electronics G120 modul – pin-out [6]

2.3.2 Wi-Fi modul

Při volbě Wi-Fi modulu jsem se rozhodoval hlavně podle dostupnosti ovladačů. Ghi Electronics nabízí ke svým produktům ovladač pro modul RS9110 od společnosti Redpine Signals. Programování je poté mnohem jednodušší a lze využívat přímo sockety v NEMTF. Díky tomu se programování síťové komunikace v NETMF neliší od „dospělého“ frameworku.

RS9110-N-11-22-04

- podpora síťových standardů IEEE 802.11b/g/n
- přenosová rychlost až 65 Mbit/s
- podpora zabezpečovacích algoritmů: 802.11i: TKIP, WEP, WPA, WPA2-PSK
- sběrnice UART či SPI
- protokoly: TCP, ARP, UDP, IPv4, DHCP klient
- napájecí napětí 3.1 - 3.6 V
- anténa integrována na modulu



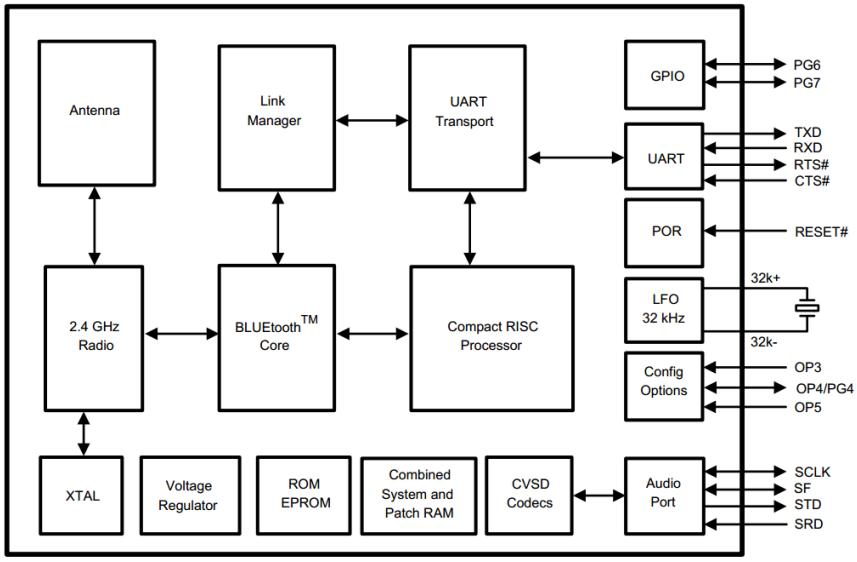
Obr. 2.12: Blokové schéma modulu Redpine Signals RS9110-N-11-22 [8]

2.3.3 Bluetooth

Bluetooth modul je plánováno využít pouze v režimu sériového portu – SPP. Externí krystal je nutné osazovat pouze v případě požadavků na velice nízkou spotřebu energie.

LMX9838

- kompletní Buetooth 2.0 stack
  - Baseband a Link Manager
  - protokoly: L2CAP, RFCOMM, SDP
  - profily: GAP, SDAP, SSP
- integrovaný krystal, anténa, EEPROM a LDO
- UART rozhraní – až 912,6 kbits/s
- pokročilé audio rozhraní pro externí PCM kodek
- napájecí napětí 3,0 – 3,6 V

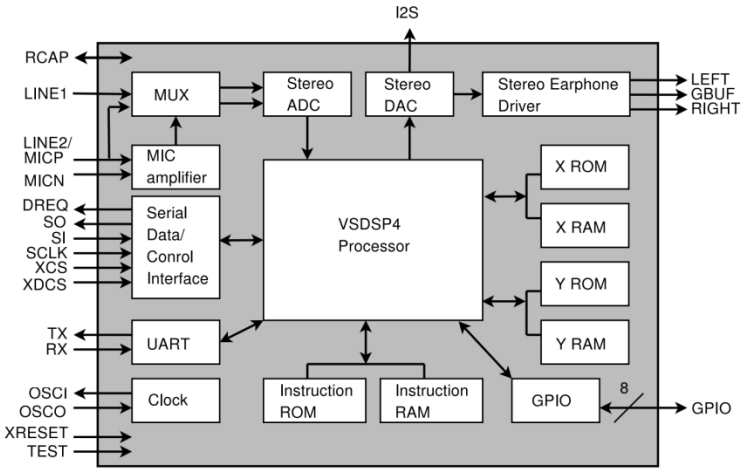


Obr. 2.13: Blokové schéma modulu Texas Instruments LMX9838 [9]

2.3.4 Audio dekodér

VS1053b

- kompletní řešení pro přehrávání a nahrávání zvuku
- integrovaný DAC s kompenzovaným fázovým zpožděním mezi kanály
- 18-bit oversampling sigma-delta DAC
- budič pro sluchátka, zesilovač pro mikrofon
- podpora pro stream MP3 a WAV – např. webové rádio
- SPI rozhraní, UART pro debug, I^2C pro externí DAC
- RAM pro uživatelský software a rozšíření



Obr. 2.14: Blokové schéma VLSI Solution - VS1053 [10]

Podporované audio formáty:

- MP3 = MPEG 1 & 2 audio layer III (CBR+VBR +ABR),
- MP1/MP2 = layers I & II optional,
- MPEG4 / 2 AAC-LC(+PNS),
- HE-AAC v2 (Level 3) (SBR + PS),
- WMA 4.0/4.1/7/8/9 all profiles (5-384kbps),
- general MIDI 1 / SP-MIDI format 0 files,
- FLAC se SW pluginem,
- WAV (PCM + IMA ADPCM),
- Ogg Vorbis se SW pluginem,
- stereo IMA ADPCM / PCM.

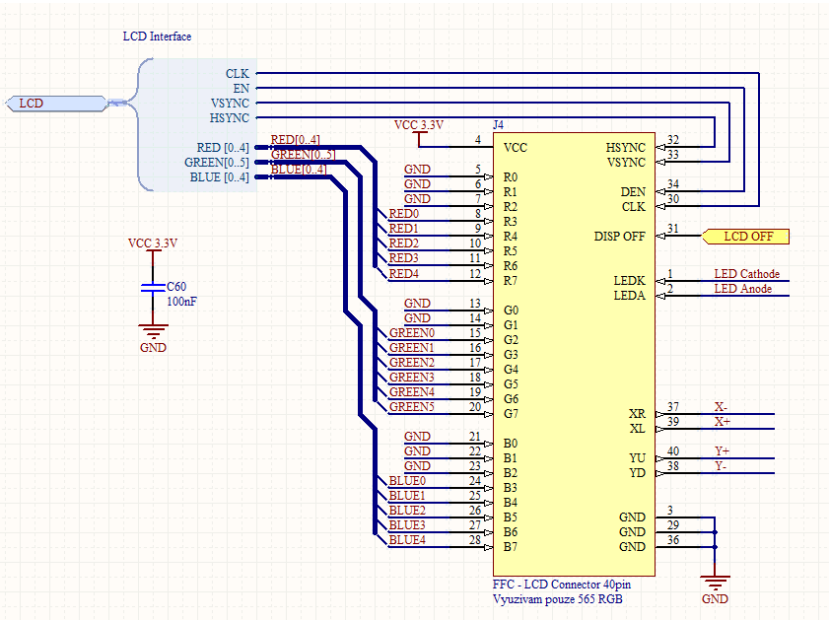
2.3.5 LCD

Výběr LCD byl limitován maximálním rozlišením 800 x 480 bodů. Toto rozlišení je maximální podporované rozlišení pro modul G120.

Rozměry	120,7(V) x 75,80(Š) x 3,40(H) mm
Rozměry zobrazovače	108,00 (V) x 64,80(Š)
Rozlišení	800 x 480 - RGB
Podsvícení	bílé
Dotykový display	rezistivní

Tabulka 2.1: Konfigurace LCD řadiče

LCD obsahuje na ohebném plošném spoji měnič pro napájení budičů a budiče displeje. Jako napájecí napětí postačuje 3,3 V. Plošný spoj lze zapojit do standardizovaného 40pinového FPC konektoru s rozestupem kontaktů 0,5mm. Zvolil jsem konektor se zámkem.



Obr. 2.15: Schéma připojení FFC LCD konektoru

RGB rozhraní

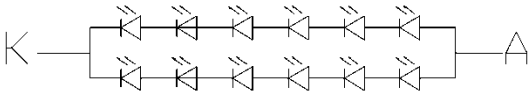
Pro každou složku barvy je vyvedeno 8bitové rozhraní. Protože řadič displeje na MCU pracuje v režimu 5-6-5, což znamená 5 bitů červené, 6 bitů zelené a 5 bitů modré, je nepoužitým LSB bitům pevně přiřazena úroveň logické 0.

Adresace

Pro adresaci bodů jsou vyvedeny signály HSYNC, VSYNC, CLK a DEN. Generování signálů je záležitostí LCD řadiče. V MCU stačí pouze nastavit příslušné periferii správné parametry časování displeje.

Podsvícení

Podsvícení displeje je uskutečněno bílými LED. Zapojení diod je patrné z Obr. 2.16. Na buzení těchto diod je zapotřebí napětí o velikosti alespoň 15 V pro vytvoření napětí alespoň 2,5 V na každém z přechodů.

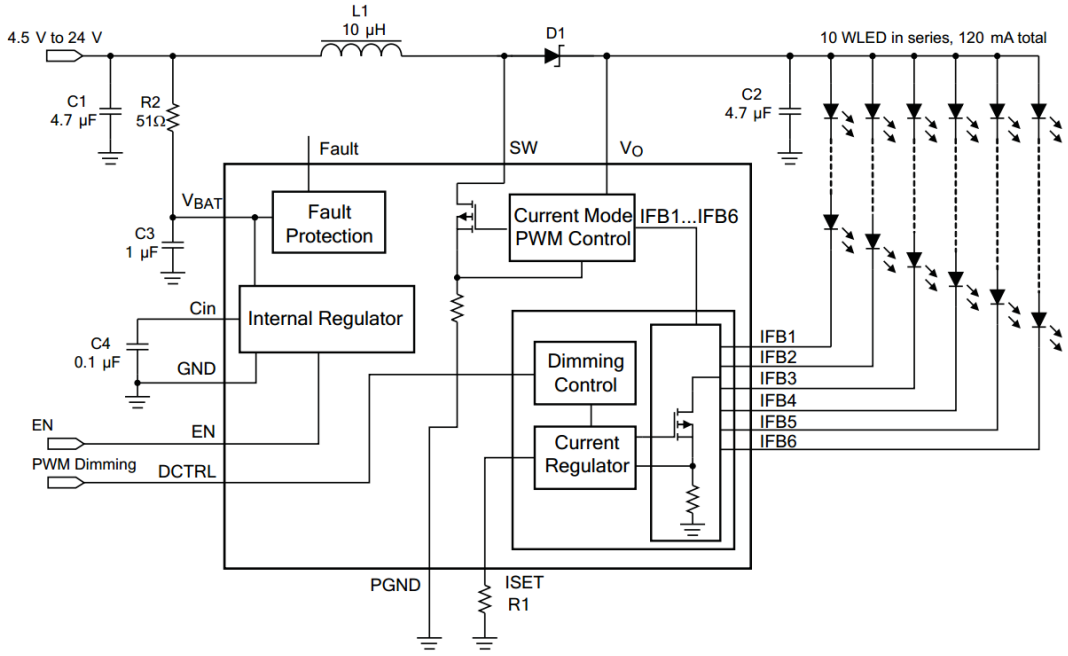


Obr. 2.16: Zapojení LED použité v podsvícení displeje

Vhodné je použití proudového buzení. Zvolil jsem obvod TPS61181A, který je napájen ze zdroje 5 V.

Obvod TPS61181A

- napájecí napětí 4,5 až 24 V
- maximální výstupní napětí 38 V
- integrovaný 1,5 A / 40 V MOSFET tranzistor
- adaptabilní step-up měnič
- ovládání PWM
- kompenzace nelinearit LED



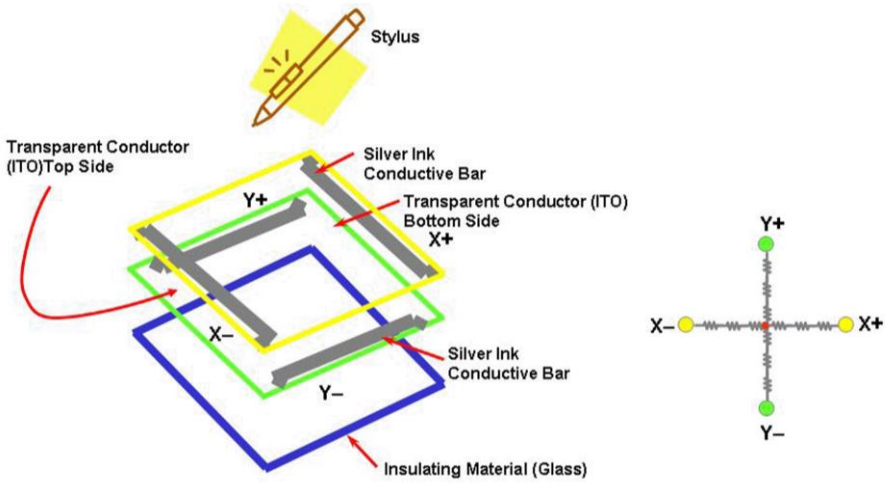
Obr. 2.17: Blokové schéma obvodu Texas Instruments TPS61181A [11]

Na další vývody ohebného plošného spoje jsou připojeny 4 signály pro dotykový panel, popsané v kapitole 2.3.6 TouchScreen, a signál DISP OFF, který umožňuje deaktivaci displeje a přechod do režimu s nízkou spotřebou.

2.3.6 TouchScreen

LCD display popsaný v předchozí kapitole je vybaven dotykovým rozhraním. Toto rozhraní je založeno na principu změny rezistivity mezi deskami.

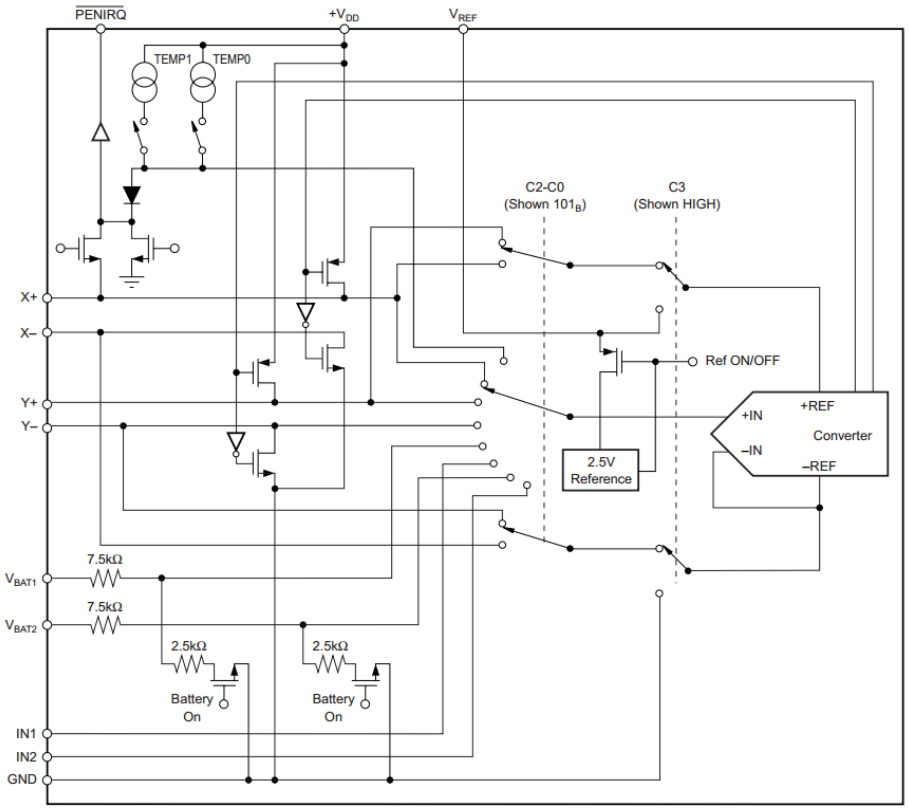
Z tohoto celku jsou vyvedeny signály X+, X-, Y+ a Y-.



Obr. 2.18: Řešení rezistivní dotykové vrstvy [4]

Obvod TSC2003

- napájecí napětí 2,5 V až 5,25 V
- interní 2,5 V reference
- přímé měření napětí na baterii (0,5 až 6 V)
- měření teploty na chipu
- měření tlaku dotyku – souřadnice Z
- I^2C komunikace
- automatické přechody do režimu s nízkou spotřebou

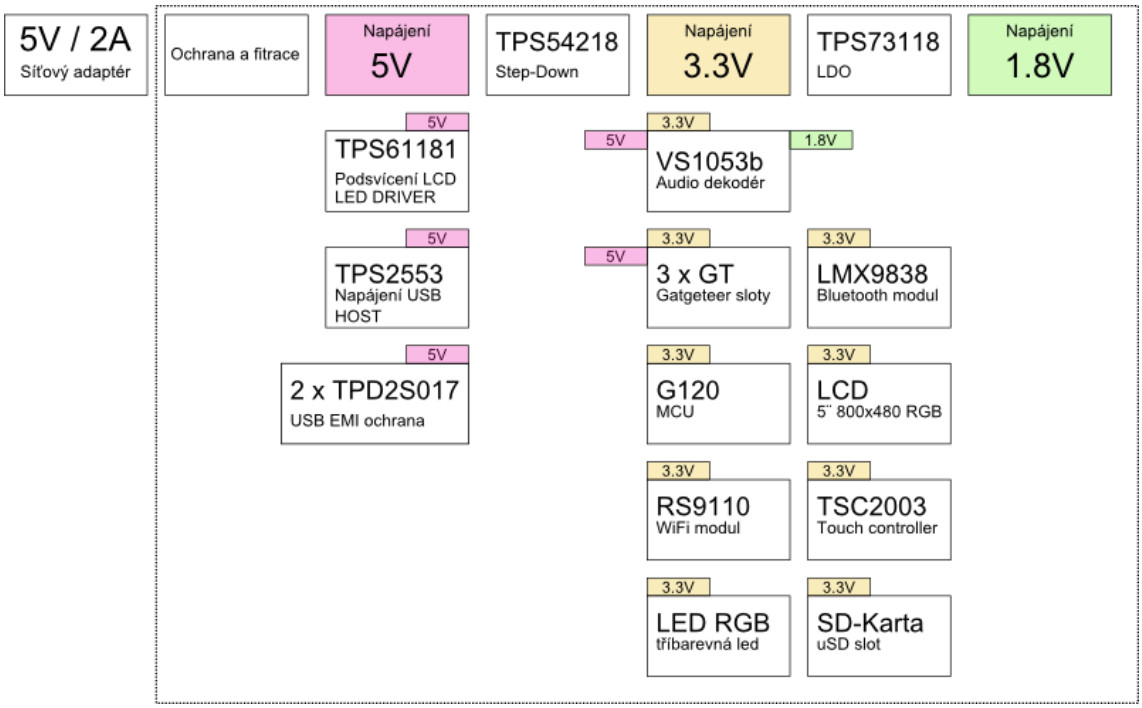


Obr. 2.19: Analogová část obvodu Texas Instruments TSC2003 [12]

Obvod obsahuje budič, DA převodník a řadič pro zmíněnou komunikační sběrnici I^2C. Při detekci dotyku je aktivován výstup PENIRQ, který informuje MCU. MCU poté vyšle povel ke změření příslušné souřadnice a následně vyčte změřenou hodnotu.

Surová napětí čtená z řadiče je však nutno kalibrovat na použitý display, protože není známa hodnota odporu jednotlivých cest a mechanických nepřesností při výrobě.

2.3.7 Napájení



Obr. 2.20: Blokové schéma napájení

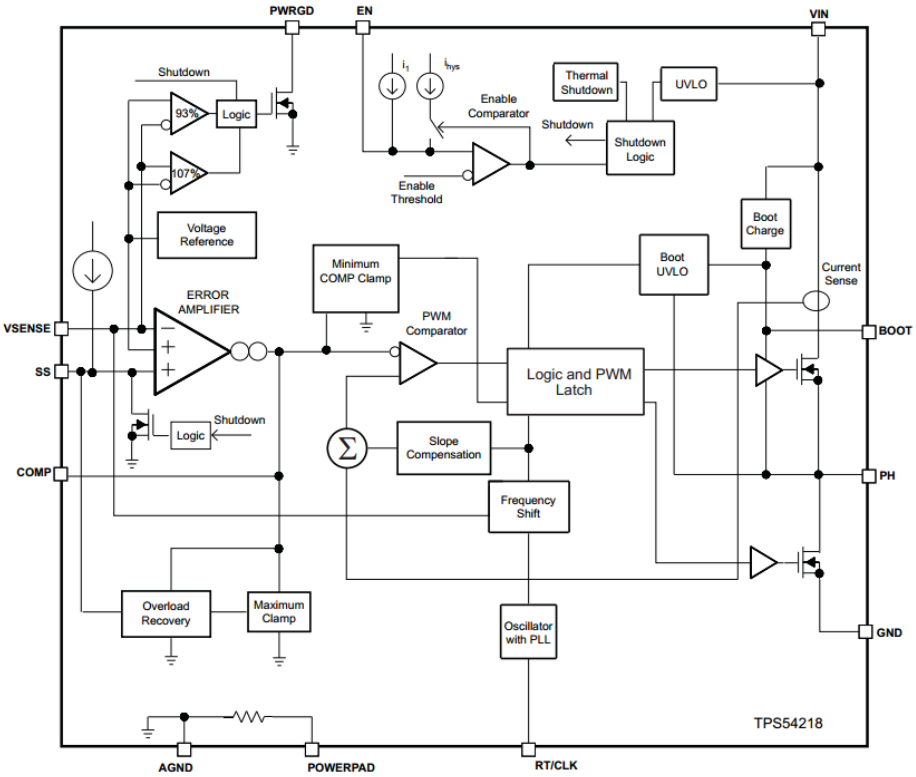
Jako hlavní zdroj napájení jsem zvolil spínaný síťový adaptér s výstupním napětím 5 V. V dnešní době je tento adaptér velice častý. Je lehce nahraditelný a jeho cena je velice příznivá. Použité součástky vyžadují tří napájecí napětí: 5 V, 3,3 V a 1,8 V. Součástky a jejich požadavky na napájení jsou zakresleny v napájecím stromě, Obr. 2.20.

Veškeré zdroje jsem volil výhradně od společnosti Texas Instrument. Důvodem je jejich vzorkový program a webová aplikace pro návrh a simulaci obvodů. Díky této návrhové a simulační aplikaci jsem dobu potřebnou pro výpočet hodnot externích součástek zkrátil na minimum.

Na volbu součástek k impulsním zdrojům jsou u každého obvodu stanoveny určité požadavky. Hlavní důraz je kladen na dielektrikum u vícevrstvých keramických kondenzátorů (třída X5R nebo X7R). U cívek pak na jejich vnitřní odpor a rezonanční frekvenci. Na tyto požadavky jsem hleděl a vybral jsem součástky s nejpodobnějšími vlastnostmi.

TPS54218

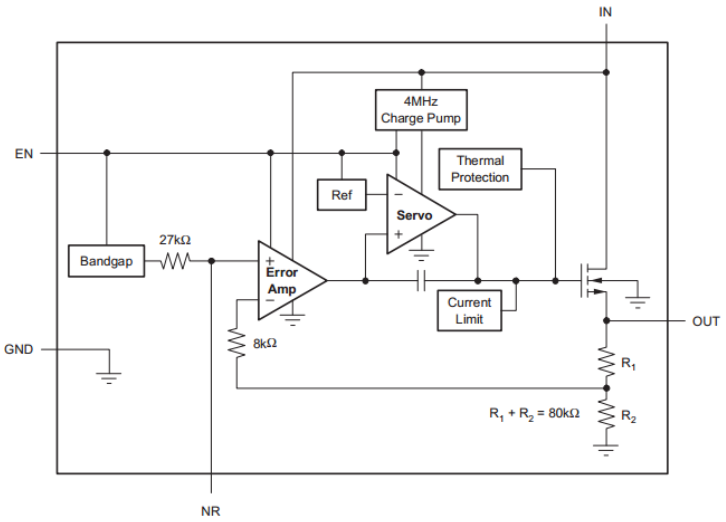
- synchronní step-down měnič, spínací frekvence 200 kHz – 2 MHz
- napájecí napětí 2,95 V až 6 V
- nastavitelné výstupní napětí, výstupní proud až 2 A
- vnitřní reference 0,8 V s přesností ± 1%
- 2 integrované výkonové MOSFET tranzistory
- integrované ochrany



Obr. 2.21: Blokové schéma obvodu Texas Instruments TPS54218 [13]

TPS73118

- LDO stabilizátor
- napájecí napětí 1,7 V - 5,5 V
- drop-out voltage: 30 mV
- bez nutnosti externích kondenzátorů
- výstupní napětí: 1,8 V



Obr. 2.22: Blokové schéma obvodu Texas Instruments TPS73118 [14]

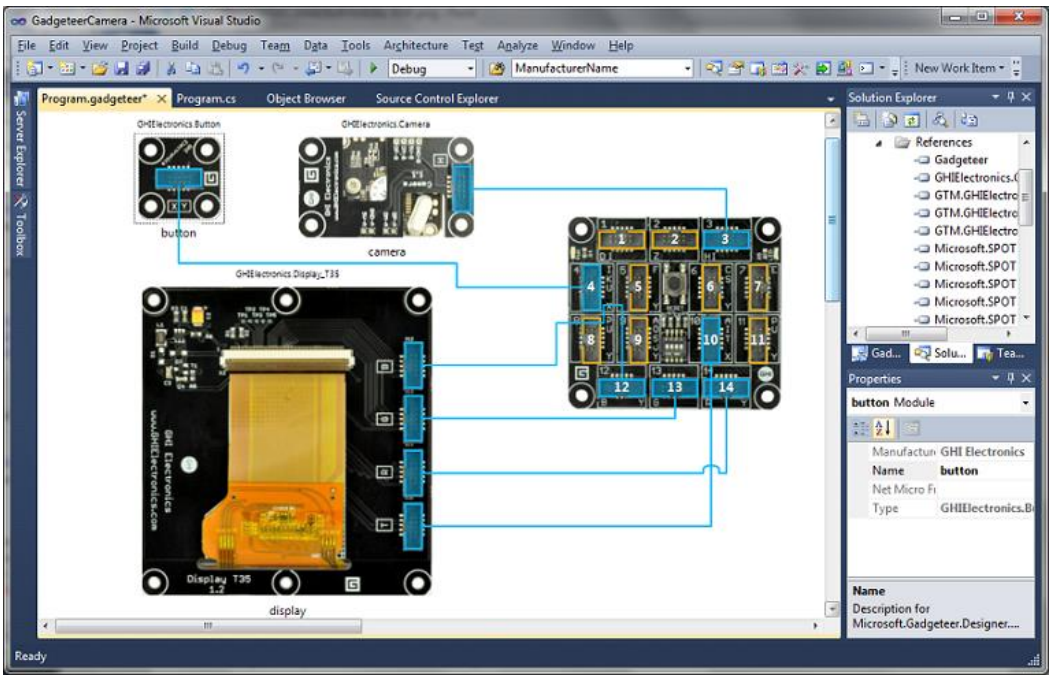
2.3.8 Gadgeteer

Použitý modul G120 nabízí plno GPIO. Všechny jsem ale nevyužil. Rozhodl jsem vyvést nepoužité GPIO na konektor pro případné rozšíření desky. Jako standart konektorů jsem zvolil Gatgeteer. Gatgeteer je popsaný v kapitole 2.2.2 .NET Gatgeteer.

Na PCB jsem umístil konektory s následujícím označením:

- S X I
  - S = SPI
  - X = 3 x GPIO
  - I = I2C
- U
  - U = UART
- I C
  - I = I2C
  - C = CAN

Díky přítomnosti těchto konektorů lze zařízení rozšířit o další komponenty velice jednoduše.



Obr. 2.23: Snímek obrazovky - projekt typu Gatgeteer – MS Visual Studio 2010

2.4 Návrh PCB

Pro návrh a vygenerování podkladů pro výrobu PCB jsem zvolil návrhové prostředí Altium Designer. S tímto prostředím jsem pracoval prvně. Jedná se o komplikovaný SW, a tudíž jsem značnou část vývoje strávil učením se v něm.

Knihovny součástek jsem z většiny kreslil a jsou přiloženy u projektu na CD ve složce */PCB/*. U některých byl k dispozici i 3D model od výrobce.

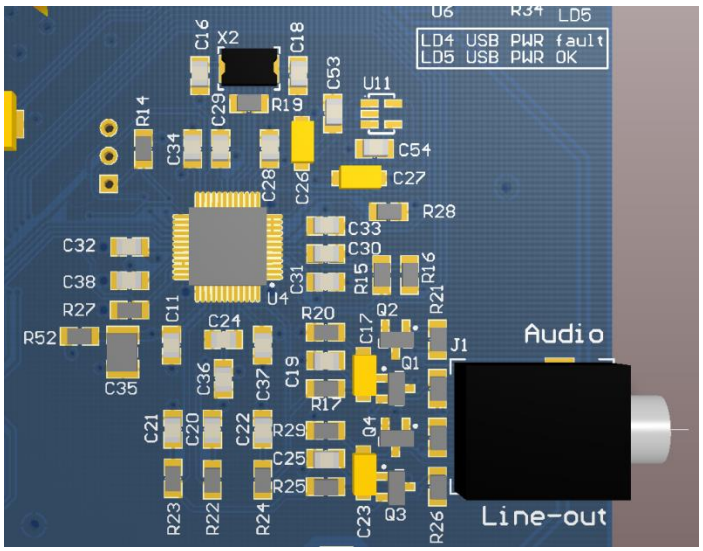
V příloze jsou k nalezení vygenerované 3D pohledy na PCB před výrobou. Možnost prohlížení a editace v 3D značně urychlila rozmístění objektů na PCB.

Layout

Při tvorbě layoutu jsem postupoval dle zásad návrhu daných v datasheetech. Impulsní zdroje použité v práci pracují na vysokých spínacích frekvencích. Je u nich proto kladen důraz na správné rozmístění součástek.

Další velice důležitou částí z hlediska rozmístění součástek a layoutu je okolí audio dekodéru VS1053. Zmíněný dekodér vyžaduje přesné umístnění filtračních kondenzátorů. Jelikož dekodér obsahuje i DA převodník je zde kladen i veliký důraz na kvalitu napájení.

Kompletní vygenerované CAM soubory včetně projektu v Altium Designeru jsou v příloze na CD ve složce */PCB/*.



Obr. 2.24: Detail rozmístění součástek v okolí dekodéru VS1053

3. Software

V této kapitole je popsaný princip zacházení s naprogramovanými třídami. Třídy jsou psané tak, aby bylo možno lehce pochopit jejich funkci. Není proto nutné uvádět celý popis funkčnosti. Zaměřuji se pouze na použití třídy v aplikaci.

Kompletní Solution v MS Visual Studio 2010 je na přiloženém CD ve složce: */SW/Multimedia Board/*.

3.1 Popis driverů

Driverem se rozumí ovladač pro jednotlivé HW prvky na PCB. Tato kapitola popisuje veřejnou část těchto tříd ovladačů.

3.1.1 LedRGB

LedRGB je statická třída, která slouží pro ovládání tříbarevné LED. Inicializace je provedena metodou *Init*, která požaduje tři PWM kanály. PWM výstupy jsou vyvedeny na jednotlivé katody tříbarevné LED.

Třída umožňuje nastavení svítivosti každé z LED v procentech. Hodnota je poté přepočítána na střídu a nastavena příslušnému PWM kanálu. Pro zobrazení barvy v RGB kódu je připravena metoda *ShowColor*. Další možností je volání metody Blink, která umožňuje nastavit libovolný počet bliknutí s parametrizovatelnou dobou sepnutí a vypnutí.

Ukázky kódu: metoda pro nastavení intenzity svitu LED – kód 3.1 a metoda pro zobrazení barvy z RGB kódu – kód 3.2.

```
public static void SetRed(int percentage)
{
     //invert percentage
     if (!_initialized) throw new NullReferenceException("Class not initialized.
     Call Init first");


     if (percentage == 0) RedLed.DutyCycle = 1;
     else  RedLed.DutyCycle = 1 - ((double)percentage / 10000 * _LedGain);
}
```

Kód 3.1: Metoda *SetRed* z třídy *Multimedia_Board.Drivers.LedRGB*

```
public static void ShowColor (byte red, byte green, byte blue, bool
WaitForBlinkingEnd = true)
{
    if (!_initialized) throw new NullReferenceException("Class not initialized.
    Call Init first");
    if (WaitForBlinkingEnd) LedOff(); //turn the led off

    _ActualColor = ColorUtility.ColorFromRGB(red, green, blue);

    SetRed((int)System.Math.Floor((red * 0.39216)));
    SetGreen((int)System.Math.Floor((green * 0.39216)));
    SetBlue((int)System.Math.Floor((blue * 0.39216)));
}
```

Kód 3.2: Metoda *ShowColor* z třídy *Multimedia_Board.Drivers.LedRGB*

3.1.2 LCD

Statická třída sloužící pro nastavení LCD řadiče a ovládání podsvícení. Inicializace je provedena metodou *Init*, která požaduje 3 GPIO a jeden PWM kanál pro nastavení intenzity podsvícení. GPIO: LCD Off signál, EN pro budič podsvícení, signál FAULT budiče podsvícení.

Nastavení vestavěného LCD řadiče je možno provést přímo zápisem hodnot do určených registrů nebo skrze inicializační třídy z *GHI.Premium* knihoven. Zvolil jsem metodu inicializace třídou. V metodě *SetUpLCDController* lze nalézt konfiguraci, která je funkční se zvoleným displejem. Všechny vlastnosti nebyly v datasheetu k LCD uvedeny, proto jsem musel s hodnotami experimentovat. Funkční parametry zobrazuje Tabulka 3.1.

Název parametru	**Hodnota**	**Název parametru**	**Hodnota**
Width	800	Height	480
OutputEnableIsFixed	false	VerticalSyncPulseWidth	2
OutputEnablePolarity	true	VerticalSyncPolarity	false
HorizontalBackPorch	0	VerticalBackPorch	0
HorizontalFrontPorch	0	VerticalFrontPorch	0
PixelPolarity	true	PixelClockRateKHz	8300

Tabulka 3.1: Nastavení LCD řadiče

Intenzita podsvícení je řízena střídou PWM. Frekvence PWM je pevně nastavena na 2,5 kHz. Třída umožňuje nastavení podsvícení v jednotkách procent. Při nastavení 0 % se budič deaktivuje a přejde do stavu s nízkou spotřebou.

Ukázka kódu: nastavení intenzity podsvícení - kód 3.3.

```
public static void SetBacklight(int percentage)
{
     if (!_initialized) throw new NullReferenceException("Class not initialized.
     Call Init first");

     if (percentage == 0)
     {
          //disable the backlight controller
          DisableBacklight();
          return;
     }
     else if (!BacklightEN.Read()) EnableBacklight();

     BacklightPWM.DutyCycle = (double)percentage / 100;
}
```

Kód 3.3: Metoda *SetBacklight* z třídy *Multimedia_Board.Drivers.LCD*

3.1.3 LMX9838

Třída pro ovládání a komunikaci s Bluetooth modulem, dále jen BT modulem. Konstruktor vyžaduje označení sériového portu, ke kterému je BT modul připojen a GPIO RST. V .NET je sériový port identifikován řetězcem „COMx“, kde x je číslo portu. Metoda *SendPacket* slouží k odeslání packetu. Packety mohou být různého typu (dáno výčtem *PacketType)* a mít různý *OpCode* (opět dáno výčtem). Dále metoda vyžaduje pole typu byte, které obsahuje data určená k odeslání v packetu.

Přenosová rychlost je dána zapojením pinů OP3, OP4 a OP5. Při návrhu jsem zvolil nejvyšší rychlost, tj. 921 600 baudů. Rychlost je nastavena ve třídě konstantou *_baudrate*.

Při přijetí packetu je packet rozkódován a zkontrolován jeho kontrolní součet. Pokud je v pořádku je vyvolána událost *PacketReceived*, která předá rozkódovaný packet, pokud dojde k chybě, je vyvolána událost *PacketReceiveError*, která předá chybný packet v surovém tvaru.

Ukázky kódů: sestavení a odeslání packetu - kód 3.4 a generování kontrolního součtu – kód 3.5.

```
public void SendPacket(PacketType packType, OpCode opCode, byte[] data)
      {
      ushort dataLenght;

      if (data != null)
      {
            dataLenght = (ushort)data.Length;
      }
      else dataLenght = 0;

      byte[] packet = new byte[7 + dataLenght];

      packet[0] = _packetStartCh;

      packet[1] = (byte)packType;
      packet[2] = (byte)opCode;

      packet[3] = (byte)(dataLenght & 0x00FF);   //second byte
      packet[4] = (byte)(dataLenght >> 8);     //first byte

      packet[5] = GenerateCheckSum(packet, 1, 3); //calculate Checksum for fields
      1 to 3

      if (data != null) data.CopyTo(packet, 6);

      packet[4 + dataLenght + 2] = _packetEndCh;

      COM.Write(packet, 0, packet.Length);
}
```

Kód 3.4: Metoda *SendPacket* z třídy *Multimedia_Board.Drivers.LMX9838*

```
private byte GenerateCheckSum(byte[] mes, int start, int end)
{
      ushort sum = 0;

      for (int i = start; i <= end; i++)
      {
       sum += mes[i];
      }

      return (byte)(sum & 0x00FF);
}
```

Kód 3.5: Metoda *GenerateCheckSum* z třídy *Multimedia_Board.Drivers.LMX9838*

3.1.4 RemMedia

RemMedia je statická třída, která zajišťuje automatickou inicializaci přenosných médií se souborovým systémem FAT. Tím se rozumí USB MassStorage a SD karta. Inicializace je provedena metodou *Init*, která vyžaduje GPIO DetectPin od slotu pro SD kartu, který předává informaci o tom, zda je karta vložena.

Třída generuje tři události:

- *RemMediaDeviceMounted*, je vyvolána při úspěšném připojení média,
- *RemMediaDeviceUnMounted*, která je vyvolána při odpojení média,
- *RemMediaErrorMounting*, která je vyvolána při neúspěšném připojení média, např. po vložení nenaformátovaného média.

Informace o zařízení, ke kterému se událost vztahuje, je předávána formou výčtu.

Ukázka kódu: vlákno zajištující připojení SD karty – Kód 3.6.

```
private static void SDMountThread()
{
      const int POLL_TIME = 1000; // check every second

      while (true)
      {
            try // If SD card was removed while mounting, it may throw exceptions
            {
               _sdExists = sdDetect.Read() == true;

               // make sure it is fully inserted and stable
              if (_sdExists)
              {
               Thread.Sleep(50);
               _sdExists = sdDetect.Read() == true;
               }

               if (_sdExists && _psSD == null)
               {
               _psSD = new PersistentStorage("SD");
               _psSD.MountFileSystem();
               }
               else if (!_sdExists && _psSD != null)
               {
               _psSD.UnmountFileSystem();
               _psSD.Dispose();
               psSD = null;
               }
            }
            catch
            {
```

```
                if (_psSD != null)
                {
                        _psSD.Dispose();
                    _psSD = null;
                }
            }
        Thread.Sleep(POLL_TIME);
        }
}
```

Kód 3.6: Metoda *SDMountThread* z třídy *Multimedia_Board.Drivers.RemMedia*

3.1.5 RS9110

RS9110 je statická třída, která je wrapperem pro třídu *WiFiRS9110* z namespace *GHI.Premium.Hardware*. Slouží pro ovládání Wi-Fi modulu. Přidává metody, které dodržují konvenci volání metod z třídy *GHI.Premium.Hardware.WiFiRS9110* a provede navázání Wi-Fi modulu do NETMF TCP/IP stacku.

Dále přidává metodu pro aktualizaci času z internetu protokolem NTP a implementuje protokol NetBIOS. Protokol NetBIOS umožňuje přiřazení názvu koncovým zařízením v síti. Je využit pro snadné adresování zařízení na síti LAN bez nutnosti znalosti IP adresy zařízení, která bývá často přiřazená dynamicky.

Třída obsahuje událost *StatusChanged*, která je volána při změně stavu. Stav je předáván formou výčtu. Pokud je modul připojen k síti, je předávána informace o síti, ke které je zařízení připojeno a pokud má platnou IP adresu, tak i TCP/IP konfigurace.

Ukázka kódu: připojení k Wi-Fi síti – kód 3.7.

```
public static bool TryConnect(string SSID, string pass)
{
      if (_wifiConnected && _wifi.IsLinkConnected) Disconnect();
      else _wifiConnected = false;

      _connecting = true;
      OnRS9110StatusChanged(Status, null, null);

      try
      {
            GHINET.WiFiNetworkInfo[] scanResult = _wifi.Scan();

            if (scanResult == null) return false; // no wlan found

            for (int i = 0; i < scanResult.Length; i++)
            {
                  if (scanResult[i].SSID == SSID)
                  {
                        _wifi.Join(scanResult[i], pass);
```

```
                    uint Time = 0;

                    while (!_wifiConnected) // give it time to connect
                    {
                        if (Time >= 2000) //try it for two second
                        {
                        _wifi.Disconnect();
                        return false;
                        }

                        Thread.Sleep(50);
                        Time += 50;
                    }
                    return true;
                }
            }
        }
        catch
        {
        _connecting = false;
        OnRS9110StatusChanged(Status, null, null);
        return false;
        }

        _connecting = false;
        OnRS9110StatusChanged(Status, null, null);
        return false;
```

Kód 3.7: Metoda TryConnect z třídy *Multimedia_Board.Drivers.RS9110*

```
}
```

3.1.6 TSC2003

TSC2003 je statická třída, která komunikuje s řadičem rezistivního dotykového panelu. Dále zajišťuje reakci na stisknutí HW tlačítek pro návrat na hlavní obrazovku nebo spuštění kalibrace.

Inicializace je provedena metodou *Init*, která vyžaduje GPIO piny, na kterých jsou připojena tlačítka, dále pak Touch Interrupt pin a rychlost I^2C komunikace. Pro měření napětí na libovolném vstupu jsem vytvořil metodu *MeasureVoltage*, která vrací hodnotu přepočítanou na napětí v jednotkách Volt. Metoda *BeginTouchCollecting* spustí vyčítání dotyků z řadiče a jejich předávání k dalšímu zpracování. Zavolání metody *CalibrateScreen* spustí kalibrační okno a provede kalibraci dotykového panelu.

Kalibrace dotykového displeje je integrována v NETMF knihovnách. Tyto knihovny využívají AD převodníky na MCU. Informaci o tom, na kterých vývodech MCU se měření provádí, jsem v době návrhu HW neměl k dispozici, protože tato část

portace byla stále ještě ve vývoji. Z tohoto důvodu nebylo možné použít „vnitřní“ detekci a kalibraci dotyků.

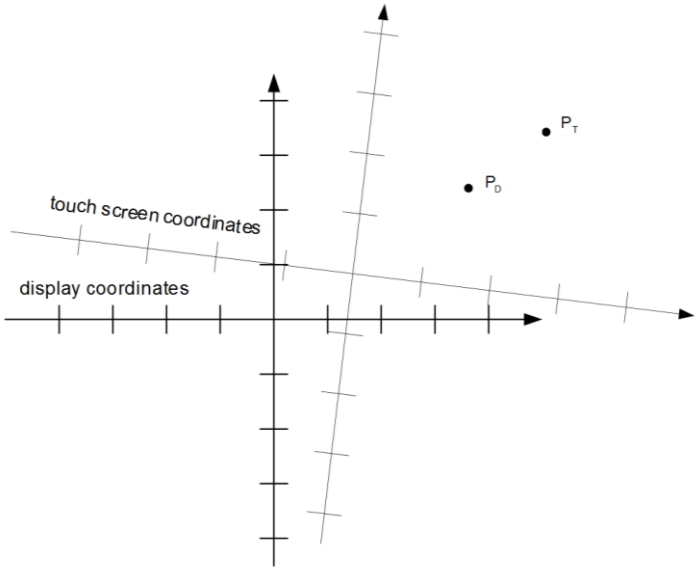
Sběr dotyků do UI je prováděn přes volání metod správce UI oken Glide. Glide bude vysvětlen v kapitole 3.2. Uživatelské rozhraní – UI.

Kalibrací se rozumí transformování naměřených hodnot do reálných souřadnic na LCD zobrazovači. Obr. 3.1 ukazuje bod P_D, který je bodem na LCD displeji. Bod P_T má souřadnice získané z řadiče při dotyku na bod P_D.

$$P_T = [X_T, Y_T],$$

$$P_D = [X_D, Y_D]$$

Rovnice 3.1



Obr. 3.1: Bod na displeji v porovnání s bodem získaným měřením dotyku [4]

Dochází ke třem chybám: chyba natočením, offset souřadnic a chyba měřítka. Chyby jsou znázorněny na obr. 3.1. Soustava rovnic 3.2 zohledňuje všechny zmíněné chyby. X_D a Y_D jsou souřadnice bodu na displeji. X_T a Y_T jsou souřadnice bodů získané z řadiče.

$$X_D = A(X_T) + B(Y_T) + C,$$

$$Y_D = D(X_T) + E(Y_T) + F$$

Rovnice 3.2

$$X_{D0} = A(X_{T0}) + B(Y_{T0}) + C,$$
$$X_{D1} = A(X_{T1}) + B(Y_{T1}) + C,$$
$$X_{D2} = A(X_{T2}) + B(Y_{T2}) + C,$$
$$Y_{D0} = D(X_{T0}) + E(Y_{T1}) + F,$$
$$Y_{D1} = D(X_{T1}) + E(T_{T2}) + F,$$
$$Y_{D2} = D(X_{T2}) + E(T_{T2}) + F$$

Rovnice 3.3

Koeficienty A, B, C, D, E a F je nutné vypočítat během kalibrace. Pro jejich výpočet je zapotřebí hodnot ze tří bodů. Koeficienty jsou uloženy ve třídě typu *CalibrationMatrix* v 64bitovém integeru. Celý tento objekt je poté serializován a uložen pomocí NETMF třídy *ExtendetWeakReference* do paměti FLASH. Obsah třídy se vyčítá při startu programu. Pokud koeficienty nejsou k dispozici, spustí se kalibrace displeje.

```
cal.Divider =  (XT0 - XT2) * (YT1 - YT2) - (XT1 - XT2) * (YT0 - YT2);

     cal.An =  (XD0 - XD2) * (YT1 - YT2) - (XD1 - XD2) * (YT0 - YT2);

     cal.Bn =  (XT0 - XT2) * (XD1 - XD2) - (XD0 - XD2) * (XT1 - XT2);

     cal.Cn =  YT0 * ((XT2) * (XD1) - (XT1) * (XD2)) +
               YT1 * ((XT0) * (XD2) - (XT2) * (XD0)) +
               YT2 * ((XT1) * (XD0) - (XT0) * (XD1));

     cal.Dn =  (YD0 - YD2) * (YT1 - YT2) - (YD1 - YD2) * (YT0 - YT2);

     cal.En =  (XT0 - XT2) * (YD1 - YD2) - (YD0 - YD2) * (XT1 - XT2);

     cal.Fn =  YT0 * ((XT2) * (YD1) - (XT1) * (YD2)) +
               YT1 * ((XT0) * (YD2) - (XT2) * (YD0)) +
               YT2 * ((XT1) * (YD0) - (XT0) * (YD1));
```

Kód 3.8: Transformace bodů získaných z řadiče do prostoru LCD, metoda *RecalculateTouch* z třídy *Multimedia_Board.Drivers.TSC2003*

```
dispX = ((cal.An * Xtouch) + (cal.Bn * Ytouch) + cal.Cn) / cal.Divider;
dispY = ((cal.Dn * Xtouch) + (cal.En * Ytouch) + cal.Fn) / cal.Divider;
```

Kód 3.9: Výpočet kalibračních koeficientů, metoda *SetCalibration* z třídy *Multimedia_Board.Drivers.TSC2003*

3.1.7 VS1053

Je statická třída, která slouží pro obsluhu audio dekodéru. Inicializuje se metodou *Init*. Metoda má nepovinný parametr typu bytové pole, ve kterém může být přiložen plugin pro rozšíření podporovaných typů audio souborů. Třída volá funkce z nativního kódu (v jazyce C++) skrze RLP od Ghi Electronics. Díky RLP je možno z managed kódu vyvolávat nativní funkce a předávat jim parametry. Stejně tak i vyčítat vrácené hodnoty.

Ghi Electornics nabízí RLP.h soubor, který definuje rozšíření. Obsahuje základní struktury pro práci s GPIO a správu Tasks. Výsledný kód se zkompiluje například volným ARM kompilátorem Yagarto a výsledný ELF soubor se přiloží jako Resource k projektu.

Nativní funkce vycházejí z volně dostupného projektu na portále Google Code[7]. Autor projekt již odstranil a nelze se na něj odkazovat. Funkce byly psané pro MCU použité v modulu EMX - LPC2478. Na modulu G120 je MCU LPC1788. Obě MCU jsou architektury ARM a jsou od výrobce NXP. Díky tomu mají i stejné zacházení s periferiemi. S pomocí datasheetu jsem změnil adresy periferií a potřebných registrů. Dále jsem nastavil děličky hodin pro periferie a upravil použité GPIO.

Z managed kódu jsou vyvolávány následující nativní funkce:

- *VsInit*
  - Provede inicializaci GPIO a nastavení SSP do režimu SPI.
- *VsStreamData*
  - Slouží k předání audio dat do vysílacího zásobníku a povoluje jejich odesílání.
- *VsNoMoreData*
  - Zavolání této funkce uvědomí vysílací funkce o ukončení streamu.
- *VsLoadPluginFromArray*
  - Nahraje plug-in do dekodéru.
- *VsSetVolume*
  - Nastaví požadovanou hlasitost v dekodéru.

[7] http://code.google.com/

Metoda *PlayFile* která má jako parametr cestu k souboru, slouží k přehrání souboru. Pokud má soubor příponu MP3 je načten také IDv2 či IDv3 tag. Ten je poté dostupný z položky *MP3info*. Metoda *PauseFile* slouží k pozastavení právě přehrávaného souboru. Metoda *ResumeFile* slouží pro navrácení do režimu přehrávání. Metoda *StopFile* slouží pro ukončení přehrávání ještě před koncem souboru.

Aktuální stav zařízení je dán výčtem v položce *Status*. Lze z něj zjistit, jakou operaci právě ovladač vykonává.

Metody pro přehrávání internetového rádia z MP3 streamu jsou *ConnectToStation* a *StopRadio*. Metoda *ConnectToStation* vyžaduje název serveru, port a cestu nebo kompletní URL.

Třídu jsem doplnil o 4 události. *VsExceptionRaised*, která je vyvolána při chybě během přehrávání a nese s sebou chybovou hlášku. Událost *VsStatusChanged*, která je vyvolána při změně stavu. Aktuální stav je přenášen formou již dříve zmíněného výčtu.

Při změně hlasitosti je vyvolána událost *VsVolumeChanged*. Při úspěšném získání informací o skladbě je vyvolána událost *VsRadioMetaDataReceived*.

Informace o skladbě se dá získat ze serveru, po přidání parametru do hlavičky HTTP dotazu: „Icy-MetaData:1“. Server pak bude do toku MP3 přidávat tzv. metadata.

Rámec obsahující metadata začíná jeho délkou. Získaná data z rámce jsou ve formě znaků. Po převedení na řetězec získáme informace o aktuální skladbě.

Obr. 3.2: Datový rámec přijímaný ze serveru [5]

Více informací o showcast streamu lze najít na stránkách SmackFu[8].

[8] http://www.smackfu.com/stuff/programming/shoutcast.html

Ukázky kódů: vlákno zajišťující vyčítání dat ze souboru – přehrávání, kód 3.10 a funkce pro čtení registru z dekodéru VS1053 v jazyce C++, kód 3.11.

```
_playThread = new Thread(() =>
{
      bool noMoreDataInvoked = false;

            byte inUse = 0;
            int length = (int)file.Length;
            int read = file.Read(_buffer[inUse], 0, BUFFER_SIZE);

            if (read == 0) return;

            while (_playThreadFlag)
            {
             _vsStreamData.InvokeEx(_buffer[inUse], read);

             inUse ^= 1;
             read = file.Read(_buffer[inUse], 0, BUFFER_SIZE);

             if (read == 0)
             {
                 _vsNoMoreData.Invoke();
                 noMoreDataInvoked = true;
                 break;
             }

             _wait.WaitOne();
```

Kód 3.10: Výňatek z metody *playFile* ze třídy *Multimedia_Board.Drivers.VS1053*

```
            }
}

static unsigned short VsReadRegister(unsigned char reg)
{
      while (LPC_SSP2->SR & SSPSR_BSY);

      SetClock7Mhz2(); //clock prescaler 10 -> 7.2MHz clock

      VsClearRxFifo();

      RLPext->GPIO.WritePin(XDCSPin, 1);  //set XDCS high
      RLPext->GPIO.WritePin(XCSPin, 0); //set XCS low

      while (!DREQ);

      VsReadWriteByte(0x03);
      VsReadWriteByte(reg);

      unsigned short result = VsReadWriteByte(0x00) << 8;
      result |= VsReadWriteByte(0x00);


      RLPext->GPIO.WritePin(XDCSPin, 0); //set XDCS low
      RLPext->GPIO.WritePin(XCSPin, 1); //set XCS high

      while (LPC_SSP2->SR & SSPSR_BSY);
```

```
    SetClock12Mhz(); //clock prescaler 6 -> 12MHz clock

    return result;
}
```

Kód 3.11: Funkce *VsReadRegister*, nativní kód – soubor *VS1053.c*

3.2 Uživatelské rozhraní - UI

Uživatel má možnost ovládání zařízení dvěma způsoby. Primárním rozhraním je LCD display s dotykovou vrstvou, v případě selhání kalibrace pak HW tlačítka. Pro vývoj tohoto UI jsem vytvořil emulátor.

Díky webovému serveru běžícímu v zařízení je zařízení možné ovládat i vzdáleně. Pro tuto variantu jsem vytvořil webovou aplikaci běžící v prohlížeči. Ta přistupuje k zařízení přes vlastní API.

3.2.1 Dotykový display

NETMF má zabudovanou podporu pro LCD display. Programátor vytvoří objekt třídy Bitmap o rozměrech displeje. Po zavolání jeho metody Flush se obsah bitmapy vykreslí na display.

Třída Bitmap umožňuje nastavovat jednotlivé pixely a kreslit grafické elementy, tj. úsečky, obdélníky a elipsy. Pomocí ní lze také vykreslovat text v libovolném fontu převedeném do formátu TinyFont, či vykreslovat další Bitmapy. Zajímavou vlastností je možnost nastavení průhlednosti jednotlivých objektů.

Barvy v NETMF jsou kódovány jako výčet *Color*. Jeho hodnota odpovídá RGB kódu barvy. Některé standartní barvy výčet obsahuje a ty, které neobsahuje lze lehce dodefinovat přetypováním RGB hodnoty barvy na výčet *Color*.

GLIDE

Glide je grafická knihovna pro NETMF zařízení, která umožňuje rychlý návrh rozložení displeje. Přidává propracovanější prvky a zjednodušuje zacházení s objekty než je tomu u WPF pro NETMF. Glide může být provozován v emulátoru

nebo na libovolném NETMF zařízením s podporou grafických funkcí, tj. s podporou třídy Bitmap.

Projekt NETMF Glide je umístěn na serveru CodePlex[9], odkud je možné stáhnout zdrojový kód a zkompilovat knihovnu. Díky jinému způsobu sběru dotyků, než je použití vnitřních NETMF funkcí, jsem musel knihovnu upravit.

K úpravám došlo v kalibračním okně, Slideru a MessageBox manažeru. Úprava spočívala ve vytvoření cesty do knihovny umožňující předávání dotyků a zrušení vyčítání dotyků z vnitřních NETMF tříd.

Výčet komponentů v GLIDE:

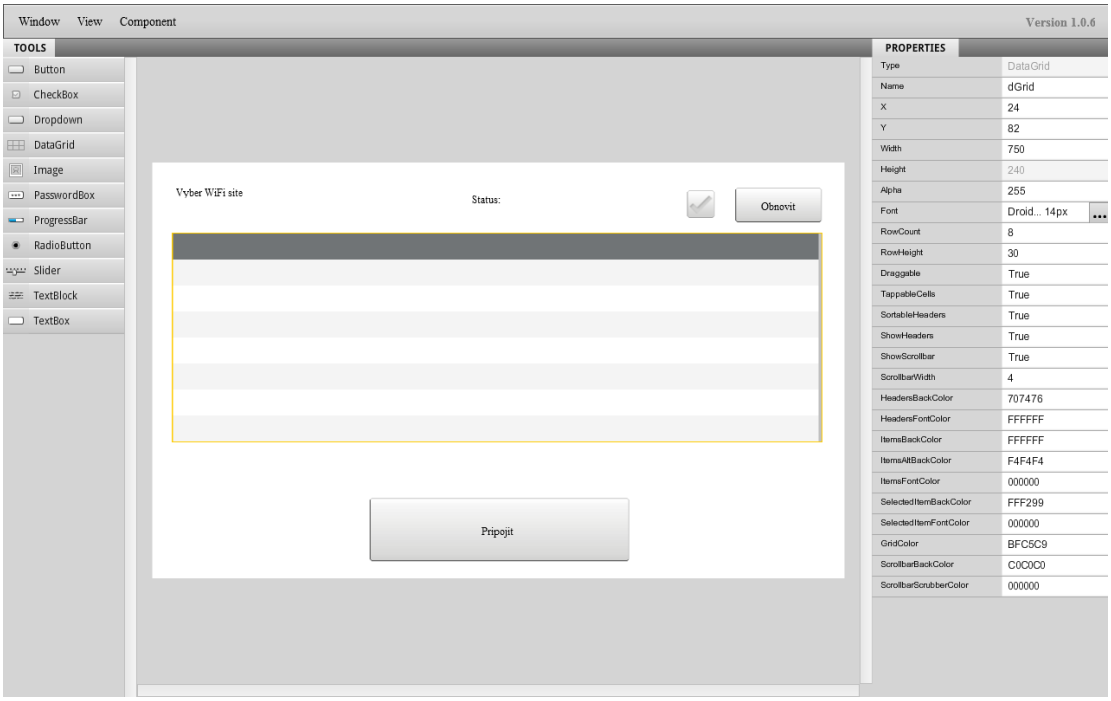
Button, Canvas, CheckBox, Dropdown, DataGrid, Image, List, MessageBox, PasswordBox, ProgressBar, RadioButton, Slider, TextBlock, TextBox.

Ke Glide je navíc dostupná webová aplikace Glide designer, která umožňuje rozmístit grafické prvky. Aplikace generuje XML kód s pozicemi a nastavením prvků.

Glide dále integruje:

- animace při překreslování oken,
- okno pro kalibraci dotykového displeje,
- možnost navázání na vlastní zdroj dotyků,
- plnohodnotnou klávesnici spolu s její obsluhou,
- správu MessageBoxů.

[9] http://netmfglide.codeplex.com/



Obr. 3.3: Snímek obrazovky - prostředí Glide Designer

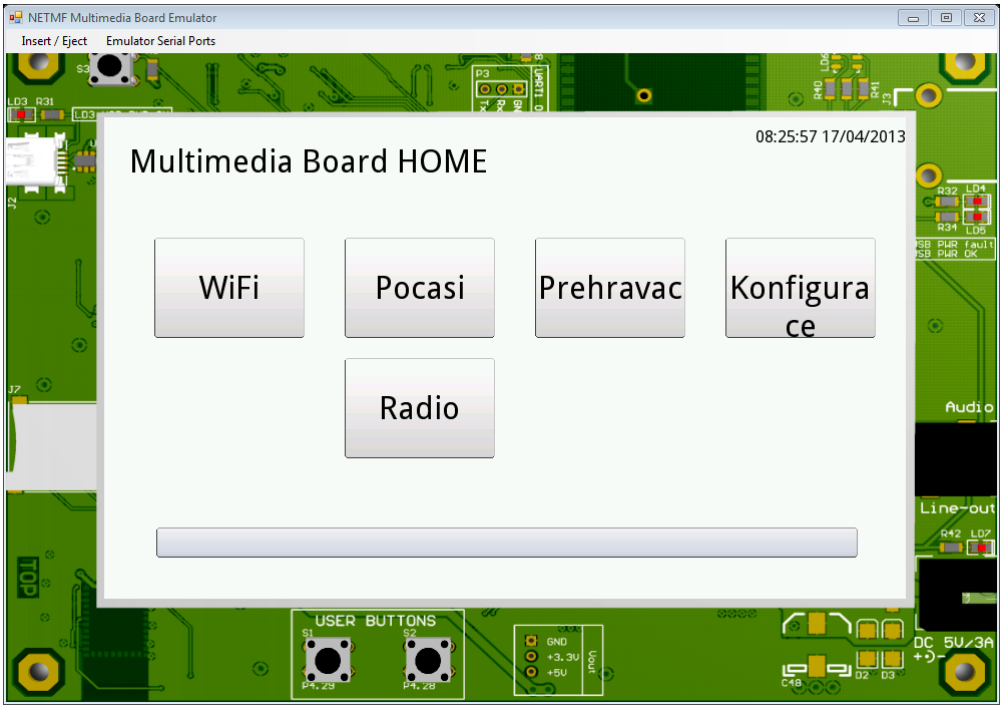
3.2.2 Simulátor

NETMF API nabízí simulátor zařízení. V simulátoru je možné simulovat GPIO, SPI zařízení, I^2C zařízení, sériové porty a síťovou komunikaci. Dále pak LCD včetně snímání dotyků z kliknutí myši.

Umožňuje nastavení velikosti paměti RAM a FLASH, rychlosti procesoru a dalších parametrů simulátoru. Za zmínku stojí i simulace baterie, která je vhodná pro návrh přenosných zařízení. Zařízení pro simulátor se programují v jazyce XML. Ukázkové kódy jsou k dispozici na přiloženém CD ve složce */SW/Emulator/*.

Během čekání na dokončení výroby plošného spoje, jsem vytvořil jednoduchý simulátor pro testování UI. Z možností jsem využil pouze simulaci LCD displeje se snímání dotyků a dvou vstupních GPIO portů pro uživatelská tlačítka.

V simulátoru jsem UI částečně nasimuloval. Tvorba kompletního simulátoru, včetně připojených zařízení, by pro tuto jednoduchou aplikaci byla zbytečná.



Obr. 3.4: Snímek obrazovky - Emulátor

3.2.3 Webová aplikace – tenký klient

Pro komunikaci s webovou aplikací jsem vytvořil jednoduché API. Tenký klient v prohlížeči tak může na pozadí komunikovat se zařízením, vyčítat jeho stav a dávat mu povely.

Na webovém serveru jsou vytvořeny virtuální cesty. Cesty začínající slovem *get* vracejí hodnoty. Při zavolání cesty se volá konkrétní obsluha, která provede požadovanou akci a odpoví na požadavek. Data v odpovědi jsou serializována dle standartu JSON. Důvodem volby tohoto typu serializace je jeho implementovanost do JavaScriptu, který jsem použil pro programování tenkého klienta s komunikací na pozadí (AJAX).

```
/func/getTime
```

Vrací čas v milisekundách v UNIX formátu. UNIX formát je používán v JavaScriptu. Ve formátu UNIX je čas uložen v počtu milisekund od půlnoci 1.1.1970. Na rozdíl od .NET, kde je čas nesen v počtu 100ns intervalů od půlnoci 1.1.0001.

`/func/getMemoryUsage`

Vrací velikost celkové a využité paměti RAM.

`/func/getWiFiStatus`

Vrací stav připojení k Wi-Fi síti, informace o síti a nastavení adres TCP/IP.

`/func/getAudioStatus`

Slouží k získání aktuálního stavu třídy VS1053 – audio dekodéru.

`/func/getAudioMP3Tag`

Vrací informaci o právě přehrávané skladbě a hodnoty z MP3 ID tagů.

`/func/audioCommand`

Slouží k ovládání přehrávání, povel je odeslán serveru metodou GET.

Ukázka kódu v JQuery pro vyčtení informací o využití paměti z API – kód 3.12. Kompletní kód webové stránky je na přiloženém CD ve složce */SW/Web/*.

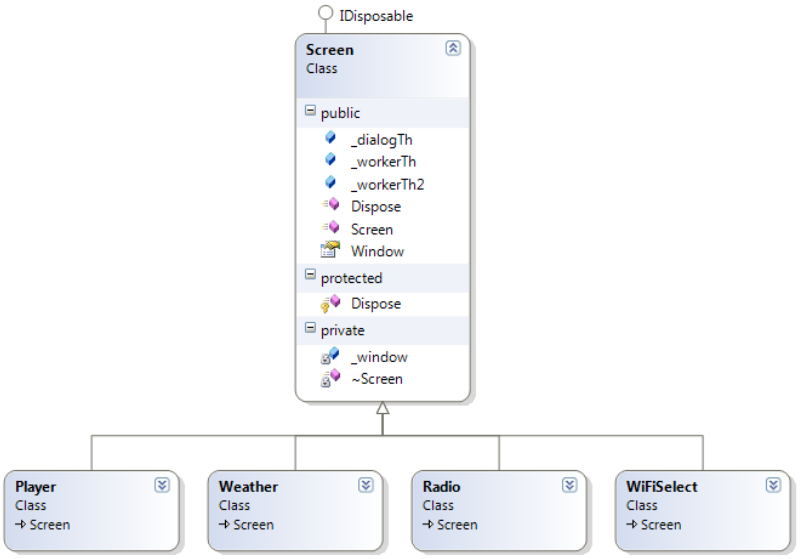
```
function GetMemory(){

    $.getJSON("http://mmboard/func/getMemoryUsage", { ForceGC: "false" })
        .done(function (data)
        {

         var total = parseInt(data.Total);
         var free = parseInt(data.Free);

         var totalMB =  total / 1048576;
         var freeMB = free / 1048576;

         var used = (total - free);
         var usedMB = used / 1048576;

         //console.log('used:' + usedMB);
         $("#MemBar").progressbar("option", "max", total);
         $("#MemBar").progressbar("option", "value", used);

         $('#MemText').empty();
         $("#MemText").append('Total: ' + totalMB.toFixed(2) + 'MB; Used: ' +
          usedMB.toFixed(2) + 'MB; Free: ' + freeMB.toFixed(2) + 'MB');

         console.log("Memory updated");
        }
);
```

Kód 3.12: Funkce *GetMemory* z webové aplikace

3.2.4 Třída Screen

Aplikace obsahuje více obrazovek, mezi kterými se přepíná. Vytvořil jsem tedy bázovou třídu *Screen*. Od této třídy poté dědí následovníci. Následovníci pak dodávají vlastní funkčnost obrazovky, ovládací prvky a metody. Konstruktor třídy *Screen* i jeho následovníků požaduje XML soubor vygenerovaný z aplikace Glide Designer. XML soubory lze umístit na SD kartu, USB zařízení nebo je přiložit k projektu jako Resource. Zvolil jsem způsob Resource. Visual Studio přikládá tyto soubory ke kódu do paměti FLASH. Konstruktor bázové třídy je doplněn inicializací grafických prvků z XML.

Po přepnutí na mezi obrazovkami je nutné uvolnit nepoužitou obrazovku z paměti, aby zbytečně nezabírala paměť RAM. Paměť RAM využitelná aplikací má ve výsledku velikost asi 7 MB. Jedna obrazovka má velikost asi 2 MB. Z tohoto důvodu implementuje každá poděděná třída metodu *Dispose*, po její zavolání se z paměti uvolní veškeré využívané prostředky. Po vykonání *Dispose* následovníka se volá ještě *Dispose* bázové třídy. V bázové třídě jsem deklaroval ještě 3 vlákna, která lze využít v poděděných třídách. Tyto vlákna jsou ukončena po zavolání metody *Dispose*. To znamená, že poděděné třídy nemusí jejich ukončení řešit.



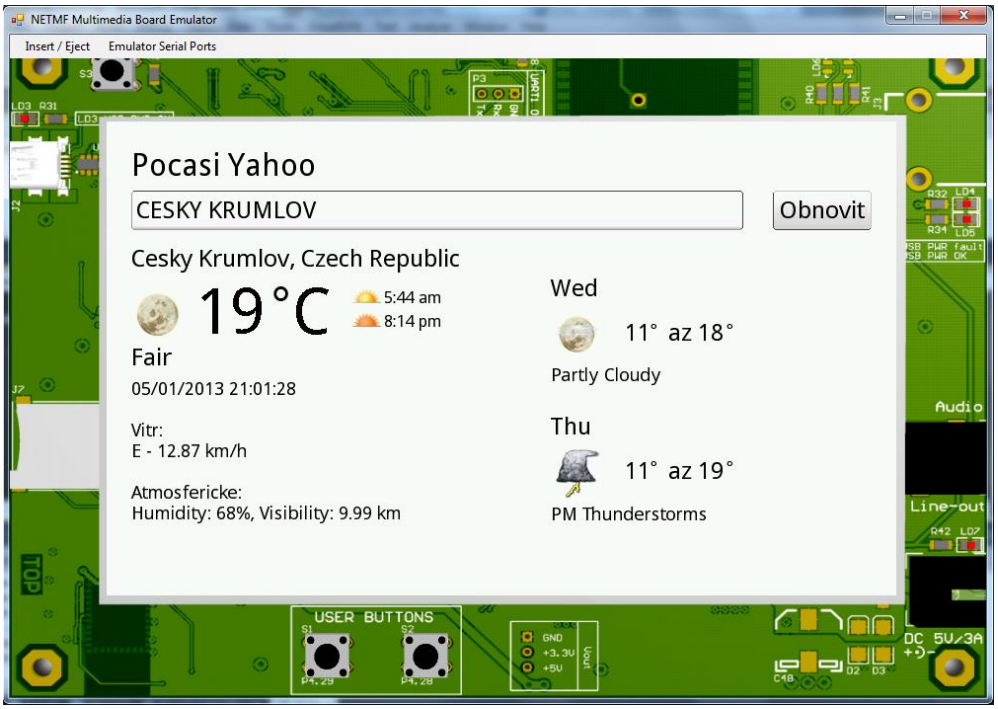
Obr. 3.5: ClassDiagram - třída Screen a její následovníci

3.2.5 Třída YahooWeather

Třída YahooWeather zajišťuje komunikaci s API webového serveru Yahoo![10]. Server vrací formou XML informace o počasí v požadované lokalitě. Oblasti jsou klíčovány podle WOEID = Where On Earth Identfier. WOEID lze získat dle názvu oblasti z API. Třída obsahuje dvě *HashTable*. Obě jsou použity jako cache, jedna pro WOEID a druhá pro obrázky znázorňující počasí.

Konstruktor nepřijímá žádné parametry, pouze inicializuje třídu. Metoda *SetWeatherPlace* využívá API pro získání WOEID, které je poté uchováno. Metoda vrací buď true a nebo false, podle toho, jestli bylo pro požadovaný název nalezeno příslušné WOEID.

Metoda *UpdateWeather* provede dotaz na API pod zvoleným WOEID. Odpověď je poté XML parserem rozkódována a předána do položky *LastWeatherCondition*. Ta obsahuje aktuální předpověď počasí.



Obr. 3.6: Snímek obrazovky – Počasí Yahoo

[10] http://www.yahoo.com/

Ukázka kódu: parsování XML souboru s počasím – kód 3.13.

```
HttpWebRequest webRequest = null;
HttpWebResponse webResponse = null;
XmlReader xmlReader = null;

YahooWeatherCondition condition = new YahooWeatherCondition();

webRequest = (HttpWebRequest)HttpWebRequest.Create(getWeatherUrl);
webResponse = (HttpWebResponse)webRequest.GetResponse();
xmlReader = XmlReader.Create(webResponse.GetResponseStream());

if (xmlReader.ReadToFollowing("yweather:location"))
{
      YahooLocation fLocation = new YahooLocation();
      fLocation.City = xmlReader.GetAttribute("city");
      fLocation.Region = xmlReader.GetAttribute("region");
      fLocation.Country = xmlReader.GetAttribute("country");
      condition.Location = fLocation;
}
else return false;

string unitTemp, unitDistance, unitPressure, unitSpeed;

if (xmlReader.ReadToFollowing("yweather:units"))
{
   unitTemp = xmlReader.GetAttribute("temperature");
   unitDistance = xmlReader.GetAttribute("distance");
   unitPressure = xmlReader.GetAttribute("pressure");
   unitSpeed = xmlReader.GetAttribute("speed");
}
else return false;

if (xmlReader.ReadToFollowing("yweather:wind"))
{
   YahooWindConditions fWind = new YahooWindConditions();
   fWind.Chill = xmlReader.GetAttribute("chill");
   fWind.Direction = double.Parse(xmlReader.GetAttribute("direction"));
   fWind.Speed = xmlReader.GetAttribute("speed") + " " + unitSpeed;
   condition.WindConditions = fWind;
}
else return false;

if (xmlReader.ReadToFollowing("yweather:atmosphere"))
{
   YahooAtmosphereConditions fAtmoshpere = new YahooAtmosphereConditions();
   fAtmoshpere.Humidity = xmlReader.GetAttribute("humidity");

   string pressureString = xmlReader.GetAttribute("pressure");
   double pressureValue = double.Parse(pressureString);
   double deltaPressure = pressureValue - 1013.25;
   fAtmoshpere.Pressure = pressureString + " " + unitPressure + " (" +
   (deltaPressure >= 0 ? "+" : string.Empty) + deltaPressure.ToString("f2") +
   ")";
   fAtmoshpere.Visibility = xmlReader.GetAttribute("visibility") + " " +
   unitDistance;
   condition.AtmosphereConditions = fAtmoshpere;
}
```

```
else return false;

if (xmlReader.ReadToFollowing("yweather:astronomy"))
{
   YahooAstronomy fAstronomy = new YahooAstronomy();
   fAstronomy.SunRise = xmlReader.GetAttribute("sunrise");
   fAstronomy.SunSet = xmlReader.GetAttribute("sunset");
   condition.Astronomy = fAstronomy;
}
else return false;

if (xmlReader.ReadToFollowing("yweather:condition"))
{
   condition.State = xmlReader.GetAttribute("text");
   condition.StateImage =
   GetImageForState(xmlReader.GetAttribute("code").Trim());
   condition.Temperature = xmlReader.GetAttribute("temp") + "°" + unitTemp;
}
else return false;
```

Kód 3.13: Výňatek z metody *UpdateWeather* ze třídy *Multimedia_Board.Apps.Weather.YahooWeather*

4. Závěr

První revize PCB obsahovala chybu v prohození vývodů footprintů, u napájecího konektoru a LED modulu. Obě tyto chyby se daly opravit přerušením cesty a následným přemostěním vodiči. V příloze je PCB projekt v revizi 1.1, který má tyto chyby opravené. Revize 1.1 neobsahuje žádné chyby.

Výsledkem práce je funkční prototyp s následujícími funkcemi:

- správa připojení k sítím Wi-Fi s možností zadání síťového klíče,
- aplikace pro získání počasí pro libovolné místo z webového portálu Yahoo!,
- přehrávač audio souborů z µSD karty nebo USB zařízení s možností dálkového ovládání přes webový prohlížeč ze zařízení v síti,
- přehrávač internetových rádií, který vyčítá seznam rádií z API webového portálu Play.cz[11].

Do SW ještě plánuji implementovat třídu pro ukládání nastavení zařízení do souboru XML, který bude uložený na µSD kartě. K úpravě této konfigurace je plánován přístup z LCD UI a webového rozhraní. Dále ještě plánuji přidat možnost vytváření a přehrávání playlistů.

K zařízení bude v budoucnu připojeno měřící PCB, které bylo výstupem mé semestrální práce. Je založené na 8bitovém MCU a slouží k měření průtoku výčepního zařízení a k regulaci jeho chlazení. Tento modul bude komunikovat s multimediální deskou skrze Bluetooth v profilu SSP.

[11] http://www.play.cz/

Seznam použitých obrázků

Použitá literatura

[1] KÜHNER, Jens. *Expert .NET Micro Framework*. 2. ed. Berkeley, CA: Apress, 2009, 481 s. ISBN 978-143-0223-870.

[2] TROELSEN, Andrew. *C# and the .NET Platform*. 2. ed. Berkeley, Calif: Apress, 2003, 1158 s. ISBN 15-905-9055-4.

[3] KOMUNITA *Ghi Electronics* [online]. [2013-04-25]. Dostupné z: http://www.ghielectronics.com/community

[4] *AN2173: Touch Screen Control and Calibration -- Four-Wire, Resistive* [online]. [2013-05-02]. Dostupné z: http://www.psocdeveloper.com/docs/appnotes/an-mode/detail/an-pointer/an2173.html

[5] SmackFu: *Shoutcast Metadata Protocol*. [online]. [2013-05-02]. Dostupné z: http://www.smackfu.com/stuff/programming/shoutcast.html

[6] Ghi Electronics: *Catalog*. [online]. [2013-05-02]. Dostupné z: http://www.ghielectronics.com/catalog/

[7] STMicroelectronics: *Embedded Processing Catalog*. [online]. [2013-05-02]. Dostupné z: http://www.st.com/web/en/catalog/mmc

[8] Redpine Signals, Inc.*: RS9110-N-11-22 – 802.11bgn Self-contained WLAN Module with Networking Stack Datasheet*. Dostupné z: http://www.redpinesignals.us/

[9] Texas Instruments*: LMX9838 Datasheet*. Dostupné z: http://www.ti.com/

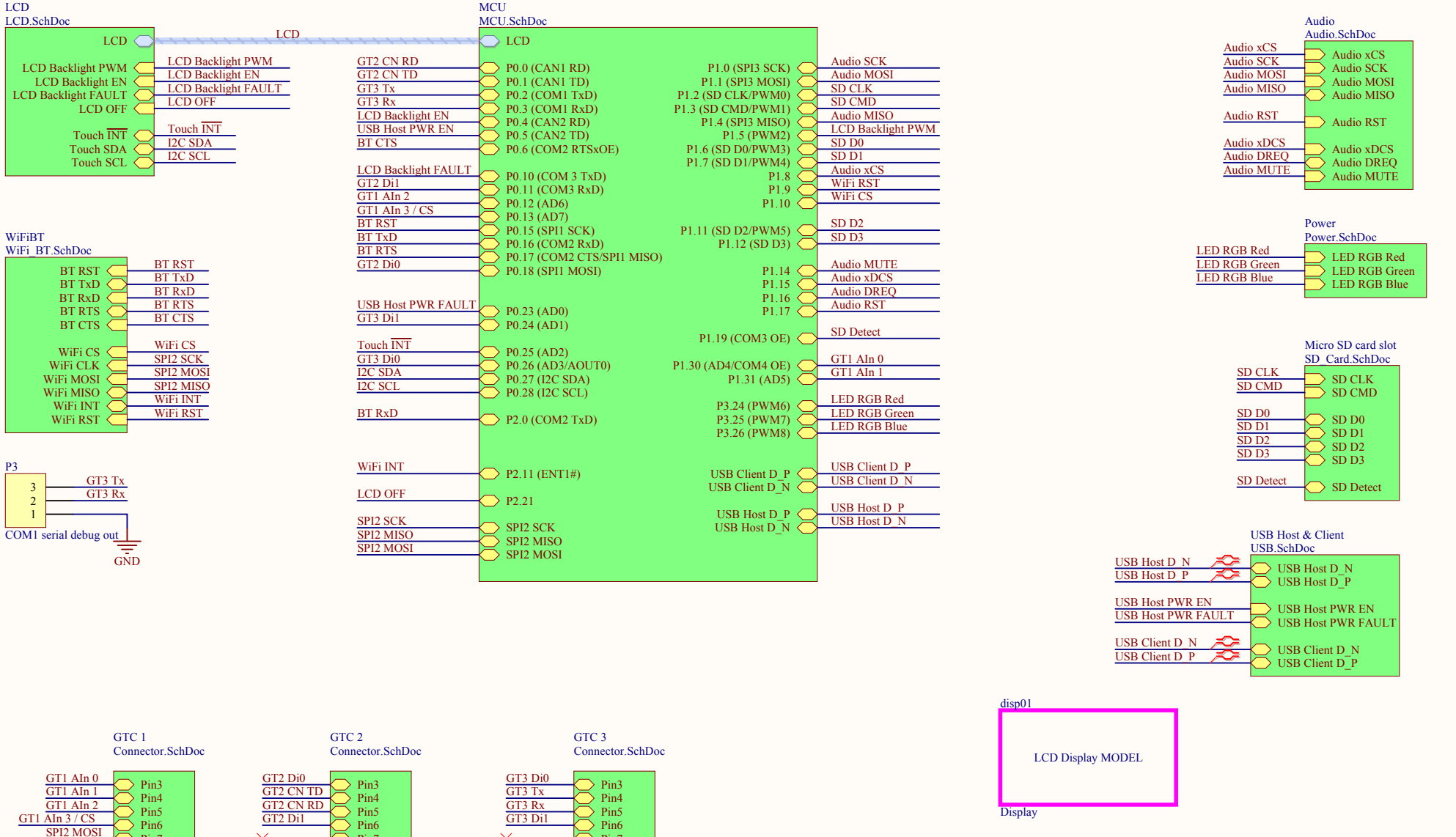
[10] VLSI Solution: *VS1053b Datasheet*. Dostupné z: http://www.vlsi.fi/

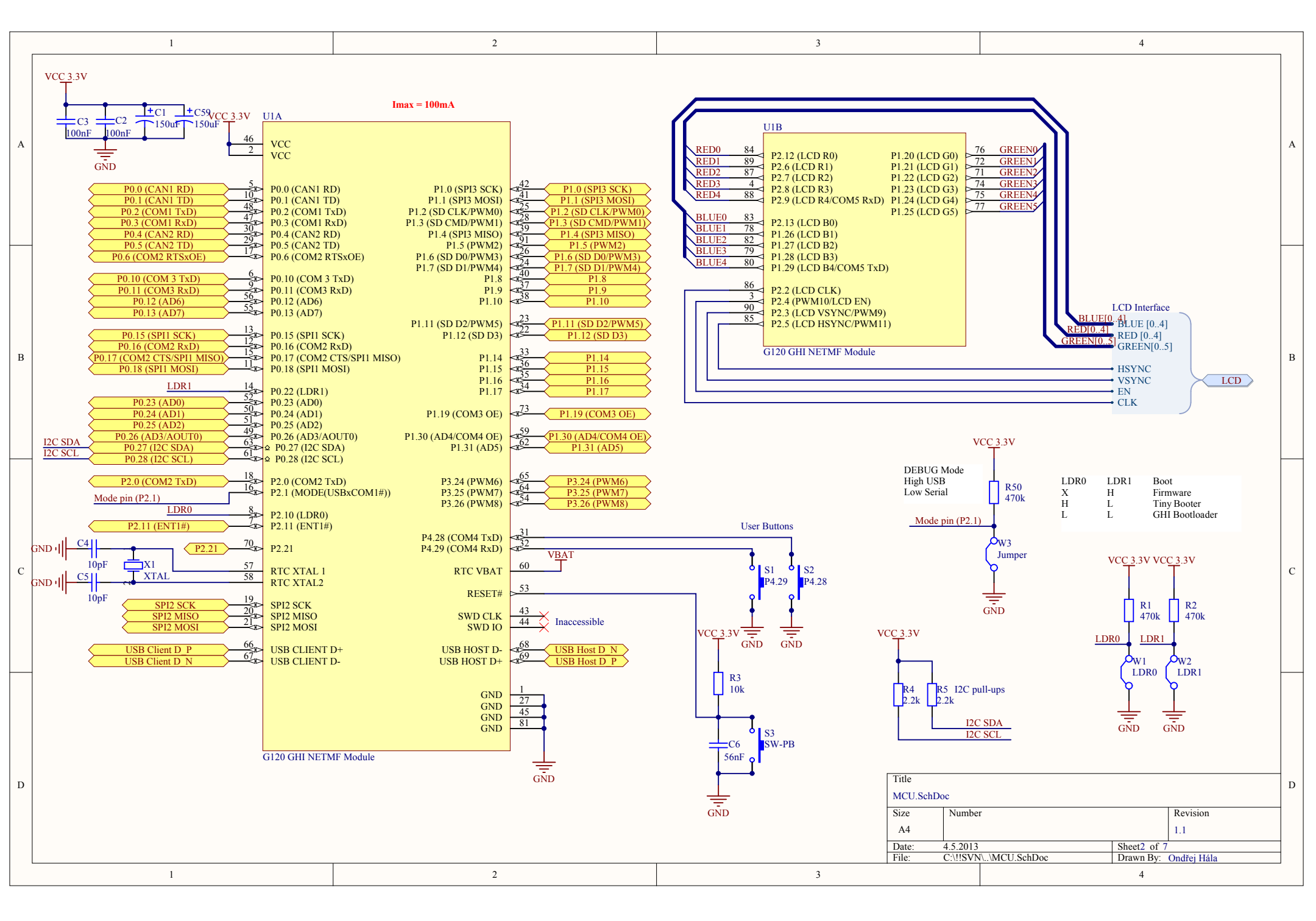
[11] Texas Instruments*: TPS61181A Datasheet*. Dostupné z: http://www.ti.com/

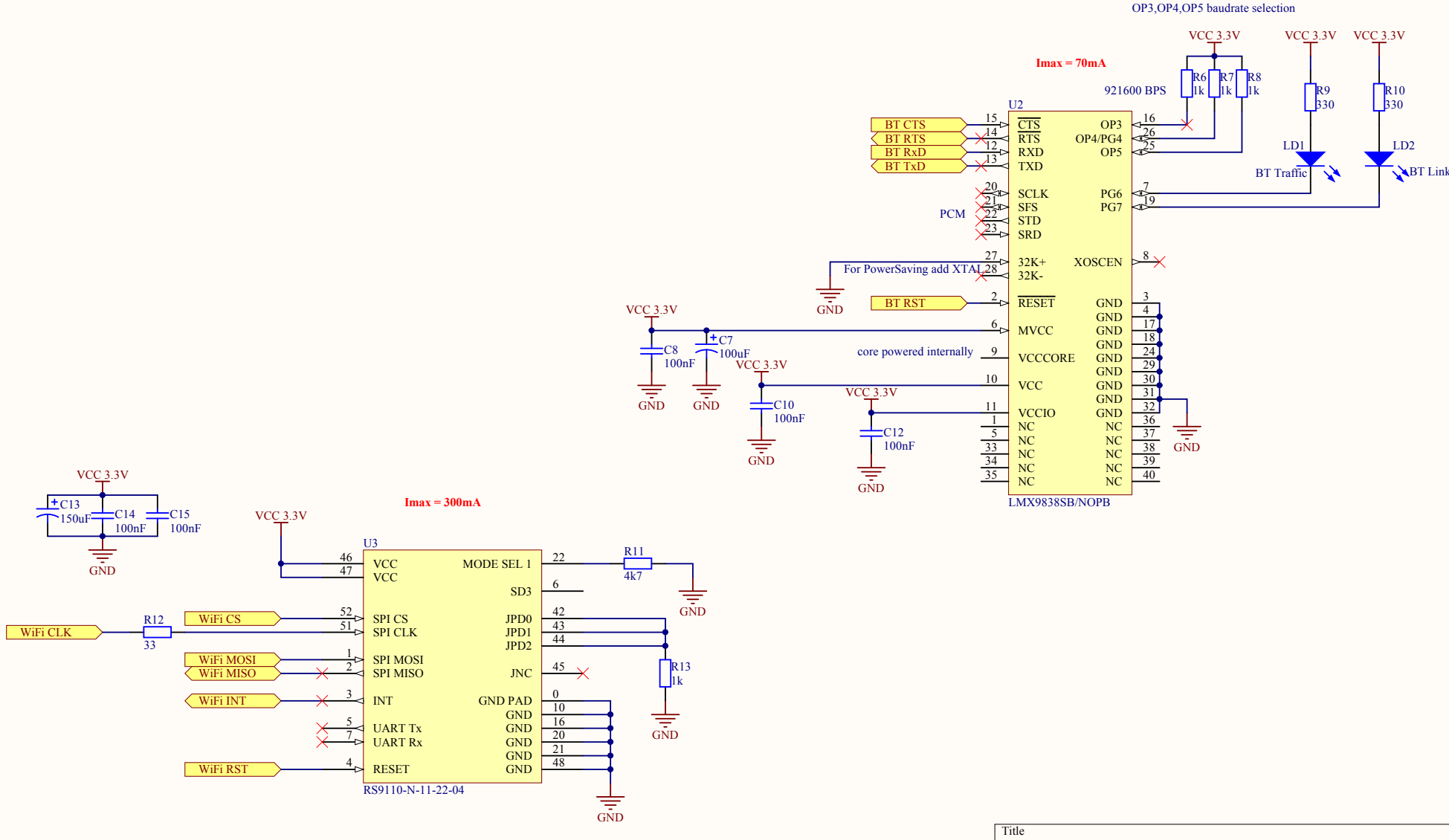
[12] Texas Instruments*: TSC2003 Datasheet*. Dostupné z: http://www.ti.com/

[13] Texas Instruments*: TPS54218 Datasheet*. Dostupné z: http://www.ti.com/

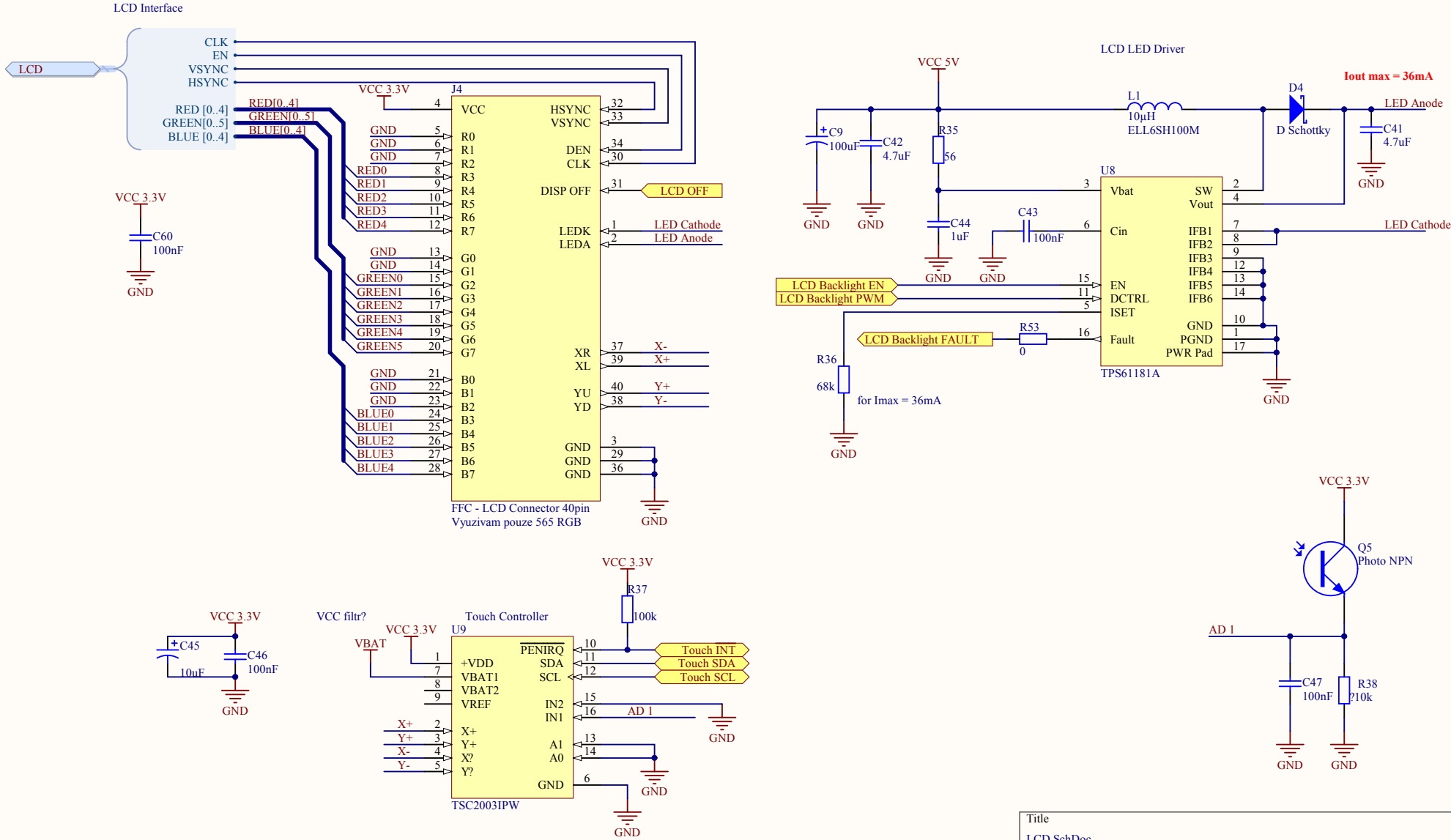
[14] Texas Instruments*: TPS73118 Datasheet*. Dostupné z: http://www.ti.com/

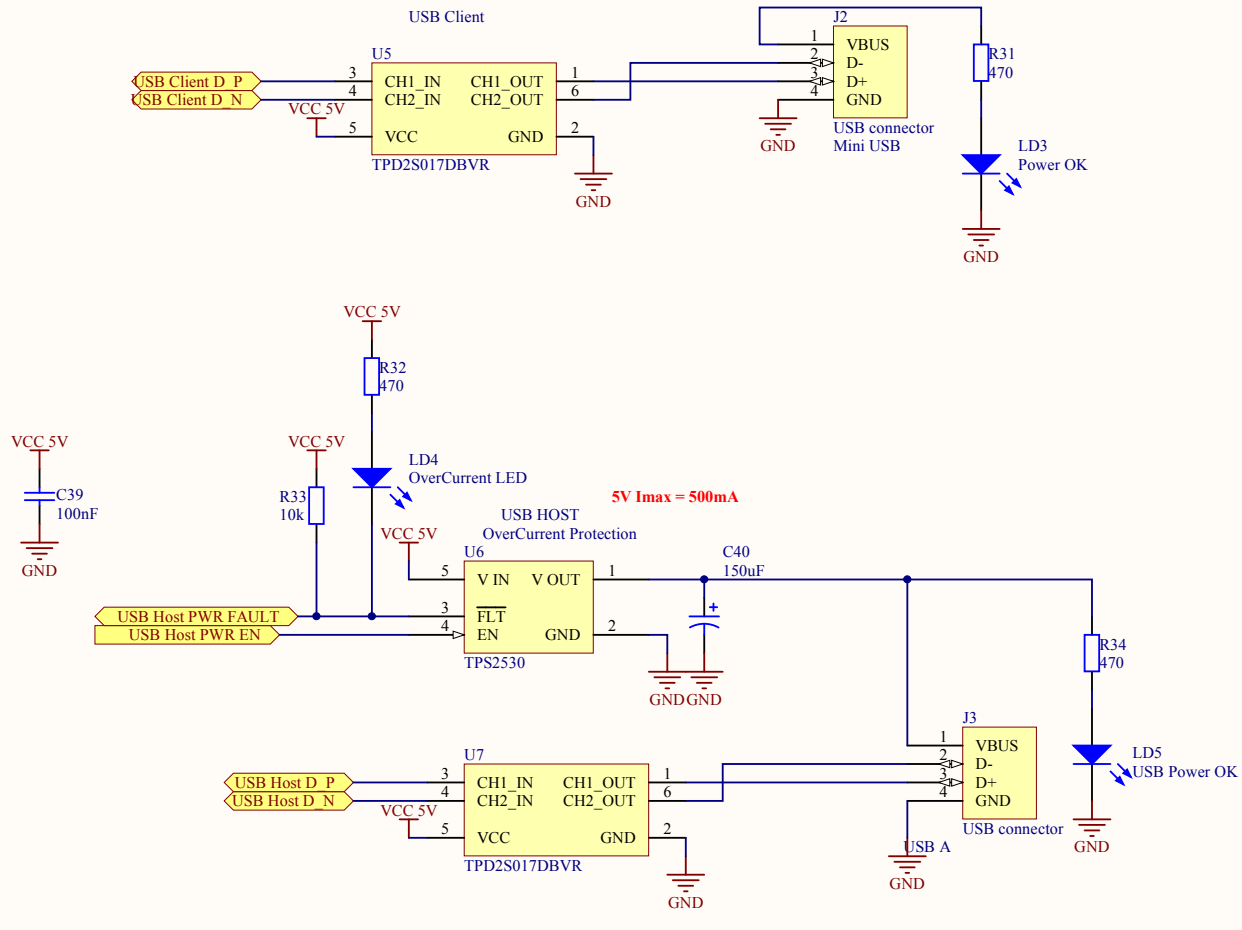

LCD
LCD.SchDoc
LCD
LCD Backlight PWM
LCD Backlight EN
LCD Backlight FAULT
LCD OFF
Touch INT
Touch SDA
Touch SCL
LCD Backlight PWM
LCD Backlight EN
LCD Backlight FAULT
LCD OFF
Touch INT
I2C SDA
I2C SCL
MCU
MCU.SchDoc
LCD
GT2 CN RD
GT2 CN TD
GT3 Tx
GT3 Rx
LCD Backlight EN
USB Host PWR EN
BT CTS
LCD Backlight FAULT
GT2 Di1
GT1 AIn 2
GT1 AIn 3 / CS
BT RST
BT TxD
BT RTS
GT2 Di0
USB Host PWR FAULT
GT3 Di1
Touch INT
GT3 Di0
I2C SDA
I2C SCL
BT RxD
WiFi INT
LCD OFF
SPI2 SCK
SPI2 MISO
SPI2 MOSI
P0.0 (CAN1 RD)
P0.1 (CAN1 TD)
P0.2 (COM1 TxD)
P0.3 (COM1 RxD)
P0.4 (CAN2 RD)
P0.5 (CAN2 TD)
P0.6 (COM2 RTSxOE)
P0.10 (COM 3 TxD)
P0.11 (COM3 RxD)
P0.12 (AD6)
P0.13 (AD7)
P0.15 (SPI1 SCK)
P0.16 (COM2 RxD)
P0.17 (COM2 CTS/SPI1 MISO)
P0.18 (SPI1 MOSI)
P0.23 (AD0)
P0.24 (AD1)
P0.25 (AD2)
P0.26 (AD3/AOUT0)
P0.27 (I2C SDA)
P0.28 (I2C SCL)
P2.0 (COM2 TxD)
P2.11 (ENT1#)
P2.21
SPI2 SCK
SPI2 MISO
SPI2 MOSI
P1.0 (SPI3 SCK)
P1.1 (SPI3 MOSI)
P1.2 (SD CLK/PWM0)
P1.3 (SD CMD/PWM1)
P1.4 (SPI3 MISO)
P1.5 (PWM2)
P1.6 (SD D0/PWM3)
P1.7 (SD D1/PWM4)
P1.8
P1.9
P1.10
P1.11 (SD D2/PWM5)
P1.12 (SD D3)
P1.14
P1.15
P1.16
P1.17
P1.19 (COM3 OE)
P1.30 (AD4/COM4 OE)
P1.31 (AD5)
P3.24 (PWM6)
P3.25 (PWM7)
P3.26 (PWM8)
USB Client D_P
USB Client D_N
USB Host D_P
USB Host D_N
Audio SCK
Audio MOSI
SD CLK
SD CMD
Audio MISO
LCD Backlight PWM
SD D0
SD D1
Audio xCS
WiFi RST
WiFi CS
SD D2
SD D3
Audio MUTE
Audio xDCS
Audio DREQ
Audio RST
SD Detect
GT1 AIn 0
GT1 AIn 1
LED RGB Red
LED RGB Green
LED RGB Blue
USB Client D_P
USB Client D_N
USB Host D_P
USB Host D_N
Audio
Audio.SchDoc
Audio xCS
Audio SCK
Audio MOSI
Audio MISO
Audio RST
Audio xDCS
Audio DREQ
Audio MUTE
Power
Power.SchDoc
LED RGB Red
LED RGB Green
LED RGB Blue
Micro SD card slot
SD_Card.SchDoc
SD CLK
SD CMD
SD D0
SD D1
SD D2
SD D3
SD Detect
USB Host & Client
USB.SchDoc
USB Host D_N
USB Host D_P
USB Host PWR EN
USB Host PWR FAULT
USB Client D_N
USB Client D_P
WiFiBT
WiFi_BT.SchDoc
BT RST
BT TxD
BT RxD
BT RTS
BT CTS
WiFi CS
WiFi CLK
WiFi MOSI
WiFi MISO
WiFi INT
WiFi RST
SPI2 SCK
SPI2 MOSI
SPI2 MISO
P3
3
2
1
GT3 Tx
GT3 Rx
COM1 serial debug out
GND
GTC 1
Connector.SchDoc
GT1 AIn 0
GT1 AIn 1
GT1 AIn 2
GT1 AIn 3 / CS
SPI2 MOSI
SPI2 MISO
SPI2 SCK
Pin3
Pin4
Pin5
Pin6
Pin7
Pin8
Pin9
S X I
GTC 2
Connector.SchDoc
GT2 Di0
GT2 CN TD
GT2 CN RD
GT2 Di1
I2C SDA
I2C SCL
IC
GTC 3
Connector.SchDoc
GT3 Di0
GT3 Tx
GT3 Rx
GT3 Di1
U
disp01
LCD Display MODEL
Display
Title
Top.SchDoc
Size
A4
Number
Revision
1.1
Date:
4.5.2013
Sheet1 of 7
File:
C:\!!SVN\..\Top.SchDoc
Drawn By:
Ondřej Hála


Imax = 100mA
U1A
G120 GHI NETMF Module
U1B
G120 GHI NETMF Module
LCD Interface
LCD
User Buttons
Inaccessible
DEBUG Mode
High USB
Low Serial
LDR0 LDR1 Boot
X H Firmware
H L Tiny Booter
L L GHI Bootloader
Title
MCU.SchDoc
Size A4
Number
Revision 1.1
Date: 4.5.2013
Sheet2 of 7
File: C:\!!SVN\..\MCU.SchDoc
Drawn By: Ondřej Hála

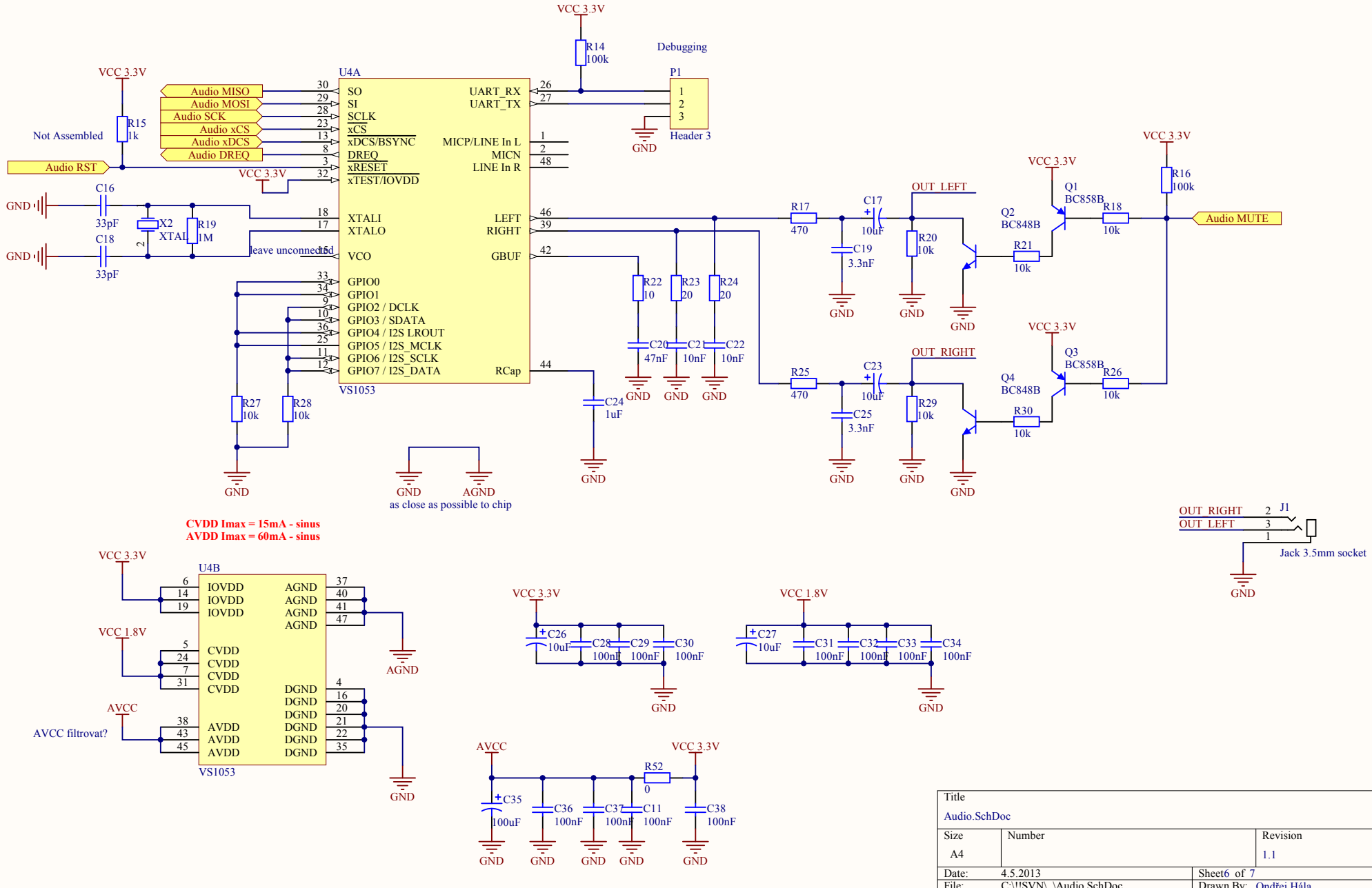


Title	WiFi_BT.SchDoc	
Size: A4	Number	Revision: 1.1
Date: 4.5.2013		Sheet 3 of 7
File: C:\!!SVN\..\WiFi_BT.SchDoc		Drawn By: Ondřej Hála

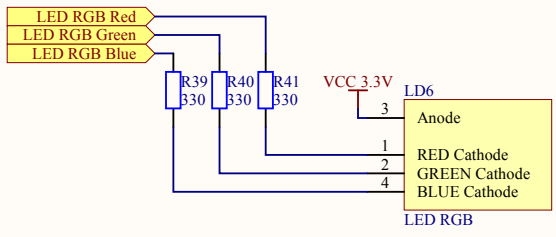

LCD Interface
LCD
CLK
EN
VSYNC
HSYNC
RED [0..4]
GREEN[0..5]
BLUE [0..4]
RED[0..4]
GREEN[0..5]
BLUE[0..4]
VCC 3.3V
C60
100nF
GND
J4
VCC
HSYNC
VSYNC
DEN
CLK
DISP OFF
LCD OFF
LEDK
LEDA
LED Cathode
LED Anode
R0
R1
R2
R3
R4
R5
R6
R7
G0
G1
G2
G3
G4
G5
G6
G7
B0
B1
B2
B3
B4
B5
B6
B7
XR
XL
YU
YD
X-
X+
Y+
Y-
GND
FFC - LCD Connector 40pin
Vyuzivam pouze 565 RGB
LCD LED Driver
VCC 5V
Iout max = 36mA
L1
10µH
ELL6SH100M
D4
D Schottky
LED Anode
C9
100uF
C42
4.7uF
R35
56
C41
4.7uF
U8
Vbat
SW
Vout
Cin
IFB1
IFB2
IFB3
IFB4
IFB5
IFB6
EN
DCTRL
ISET
GND
PGND
PWR Pad
Fault
C44
1uF
C43
100nF
LED Cathode
LCD Backlight EN
LCD Backlight PWM
LCD Backlight FAULT
R53
0
R36
68k
for Imax = 36mA
TPS61181A
VCC filtr?
Touch Controller
U9
VBAT
+VDD
VBAT1
VBAT2
VREF
X+
Y+
X?
Y?
PENIRQ
SDA
SCL
IN2
IN1
A1
A0
R37
100k
Touch INT
Touch SDA
Touch SCL
AD 1
TSC2003IPW
C45
10uF
C46
100nF
Q5
Photo NPN
C47
100nF
R38
?10k
Title
LCD.SchDoc
Size
A4
Number
Revision
1.1
Date:
4.5.2013
Sheet4 of 7
File:
C:\!!SVN\..\LCD.SchDoc
Drawn By:
Ondřej Hála

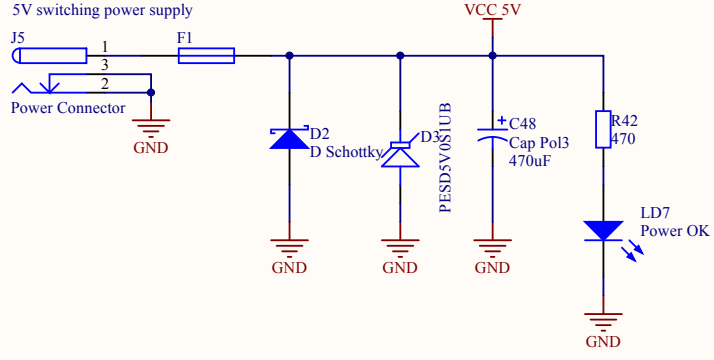


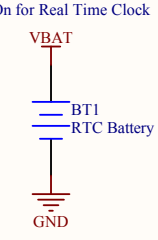
Title: USB.SchDoc		
Size: A4	Number:	Revision: 1.1
Date: 4.5.2013		Sheet5 of 7
File: C:\!!SVN\..\USB.SchDoc		Drawn By: Ondřej Hála

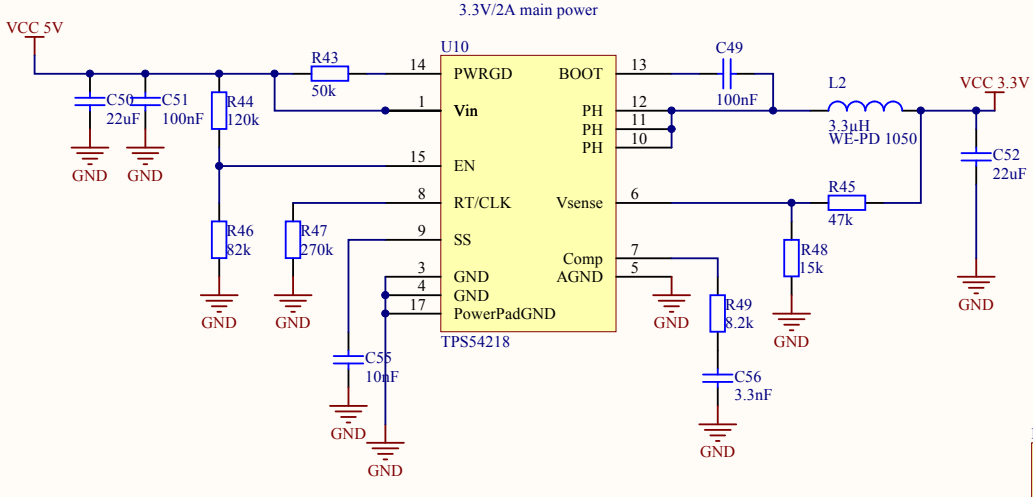


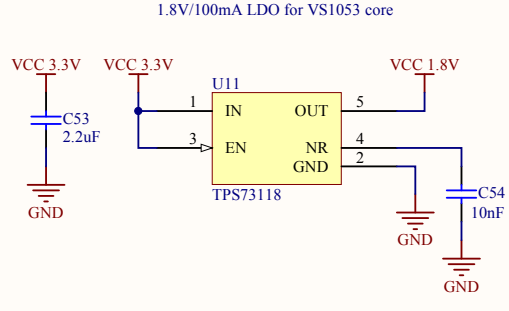
Title	Audio.SchDoc	
Size: A4	Number:	Revision: 1.1
Date: 4.5.2013	Sheet6 of 7	
File: C:\!!SVN\..\Audio.SchDoc	Drawn By: Ondřej Hála	

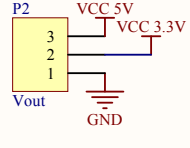




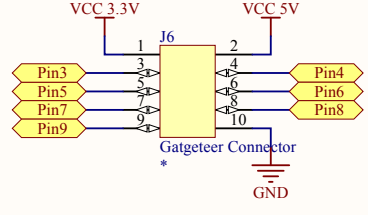






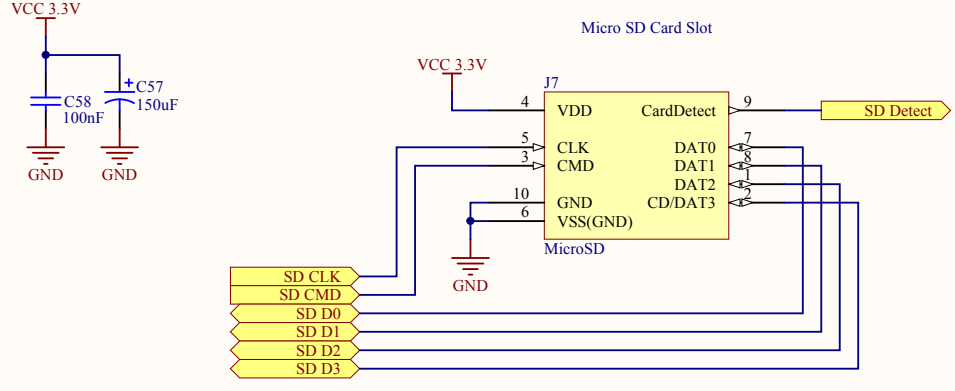


Title: Power.SchDoc		
Size: A4	Number:	Revision: 1.1
Date: 4.5.2013		Sheet7 of 7
File: C:\!!SVN\..\Power.SchDoc		Drawn By: Ondřej Hála

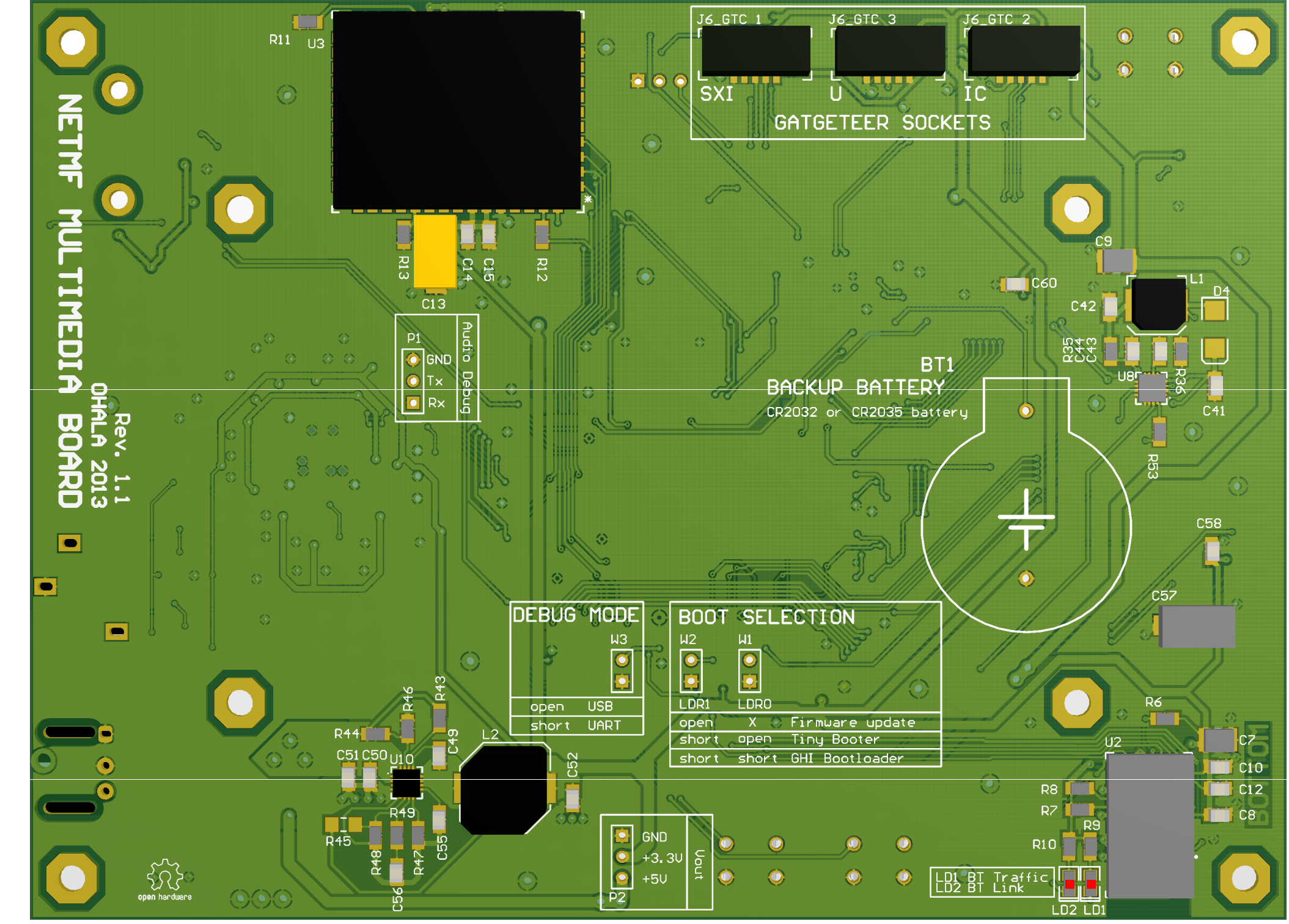


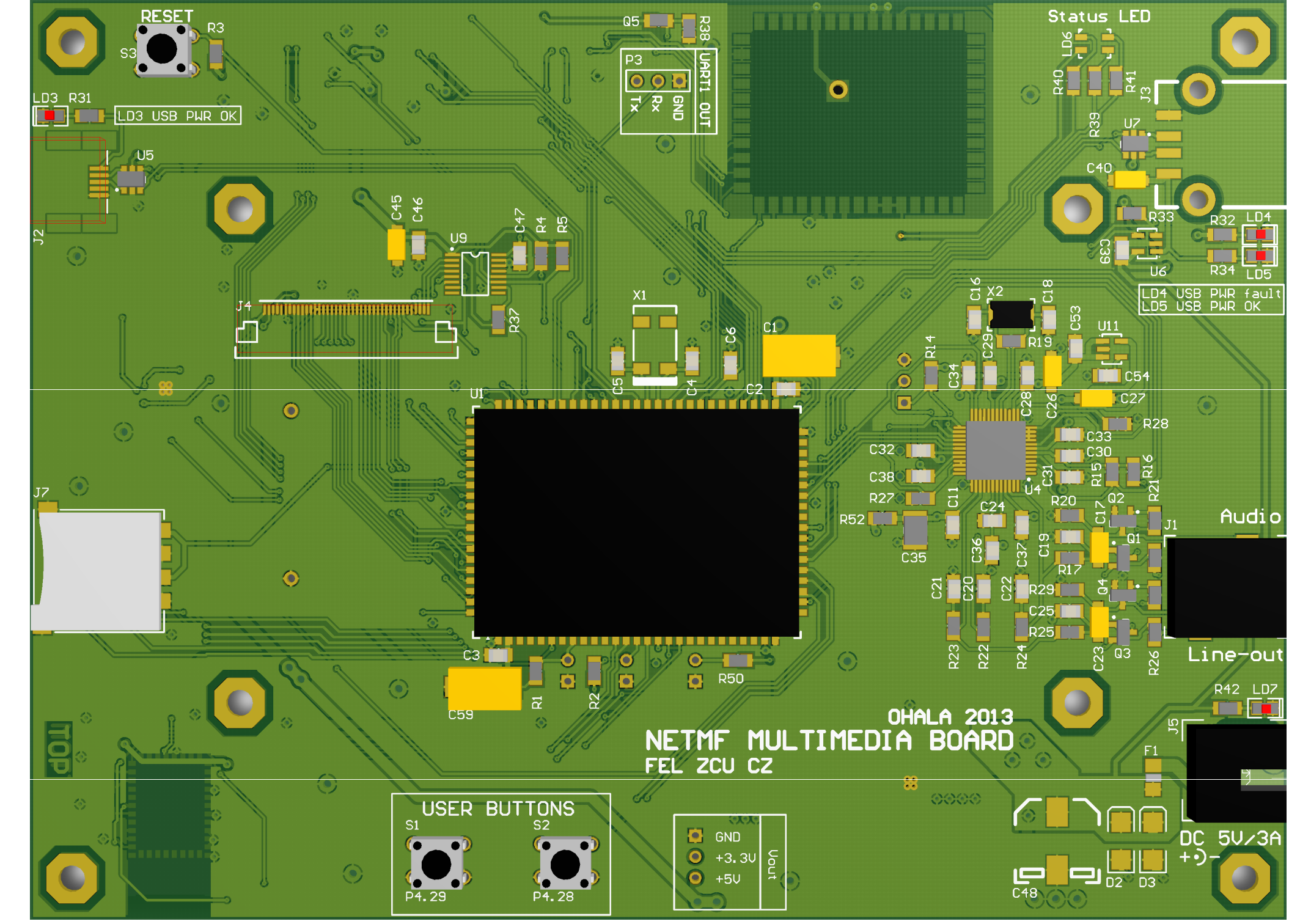
Gadgeteer:
http://gadgeteer.codeplex.com/wikipage?title=.NET%20Gadgeteer%20Socket%20Types&referringTitle=Documentation

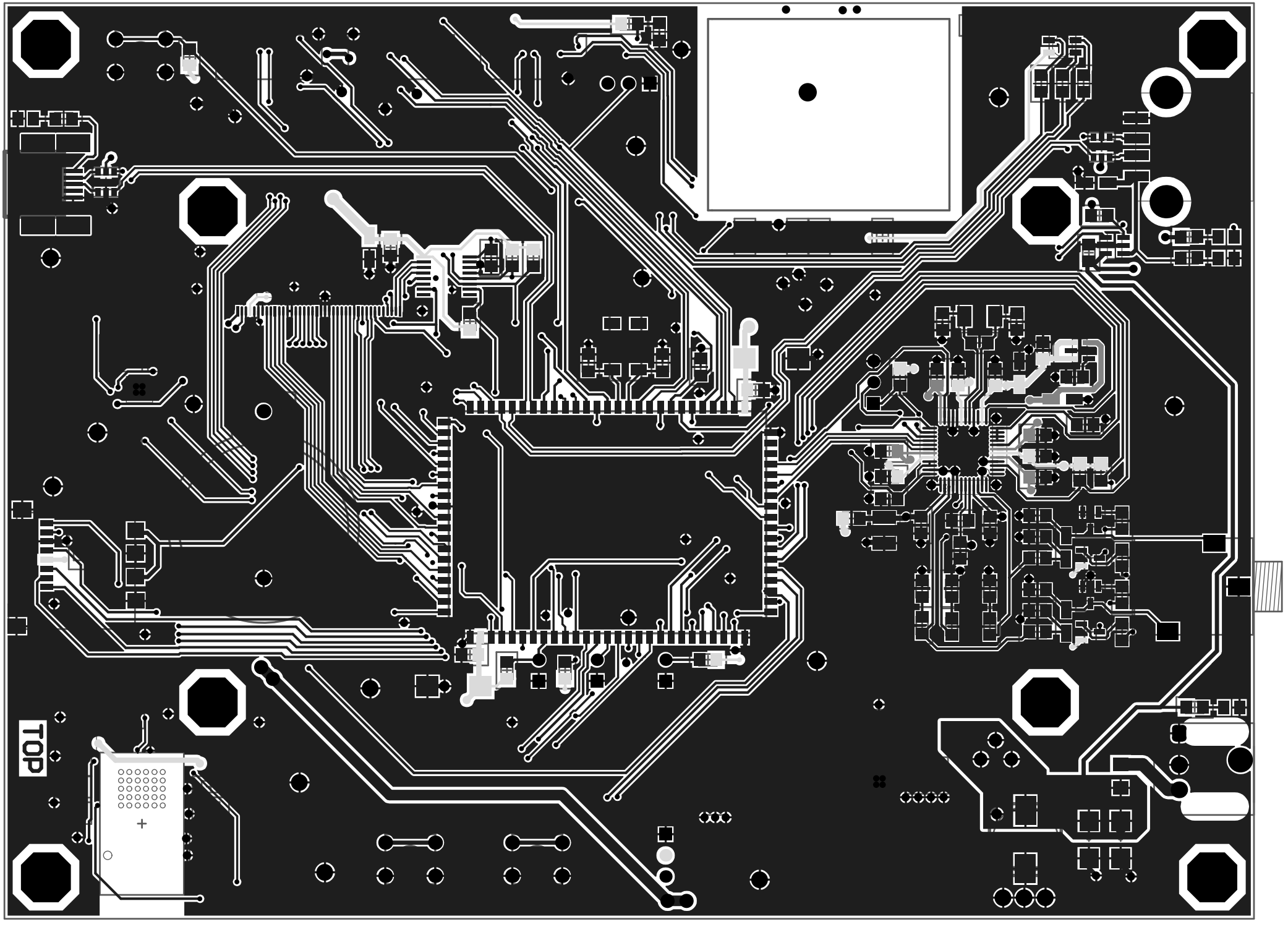
Title	Connector.SchDoc		
Size	Number		Revision
A4			1.1
Date:	4.5.2013	Sheet * of *	
File:	C:\!!SVN\..\Connector.SchDoc	Drawn By:	Ondřej Hála

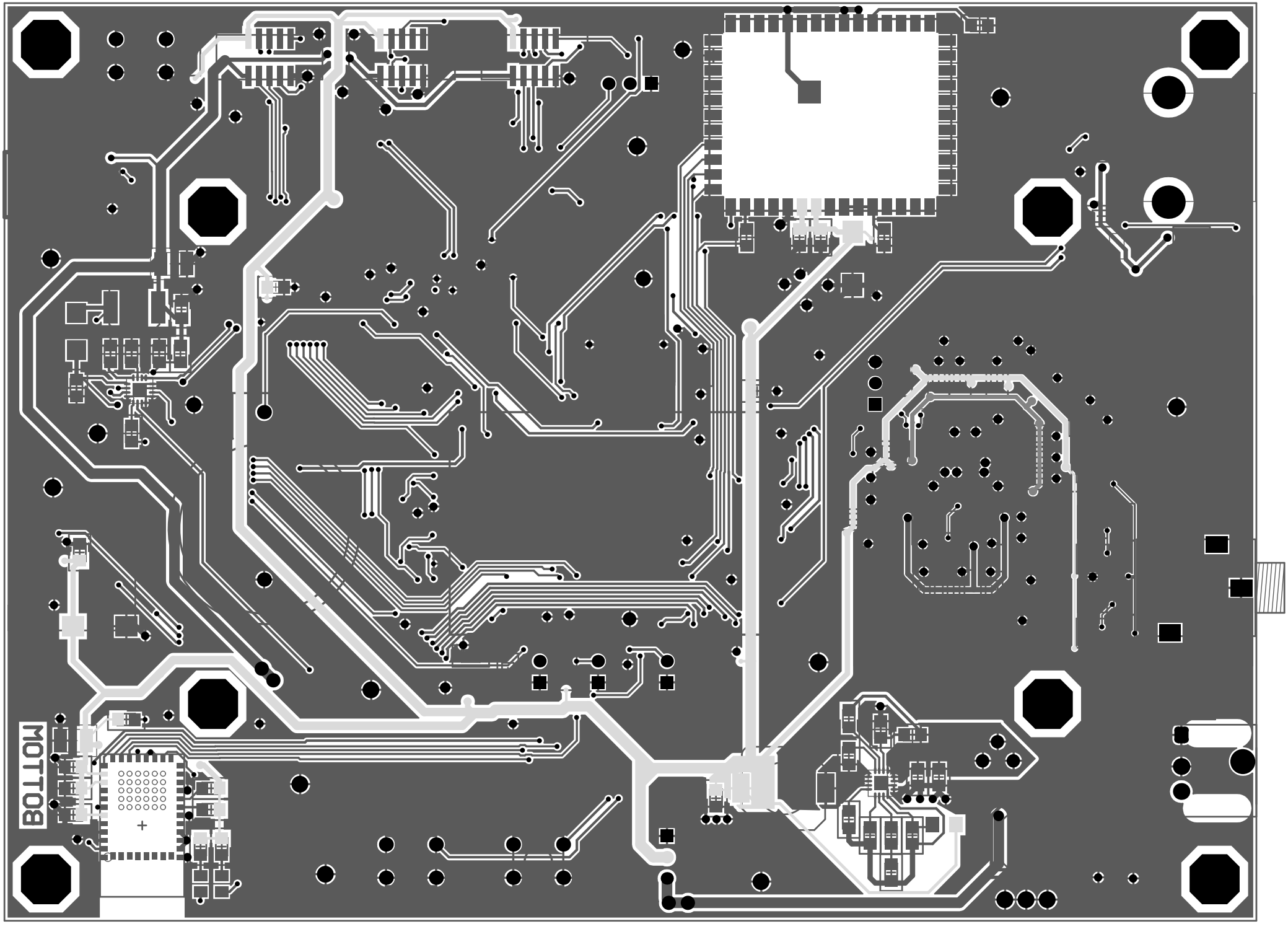


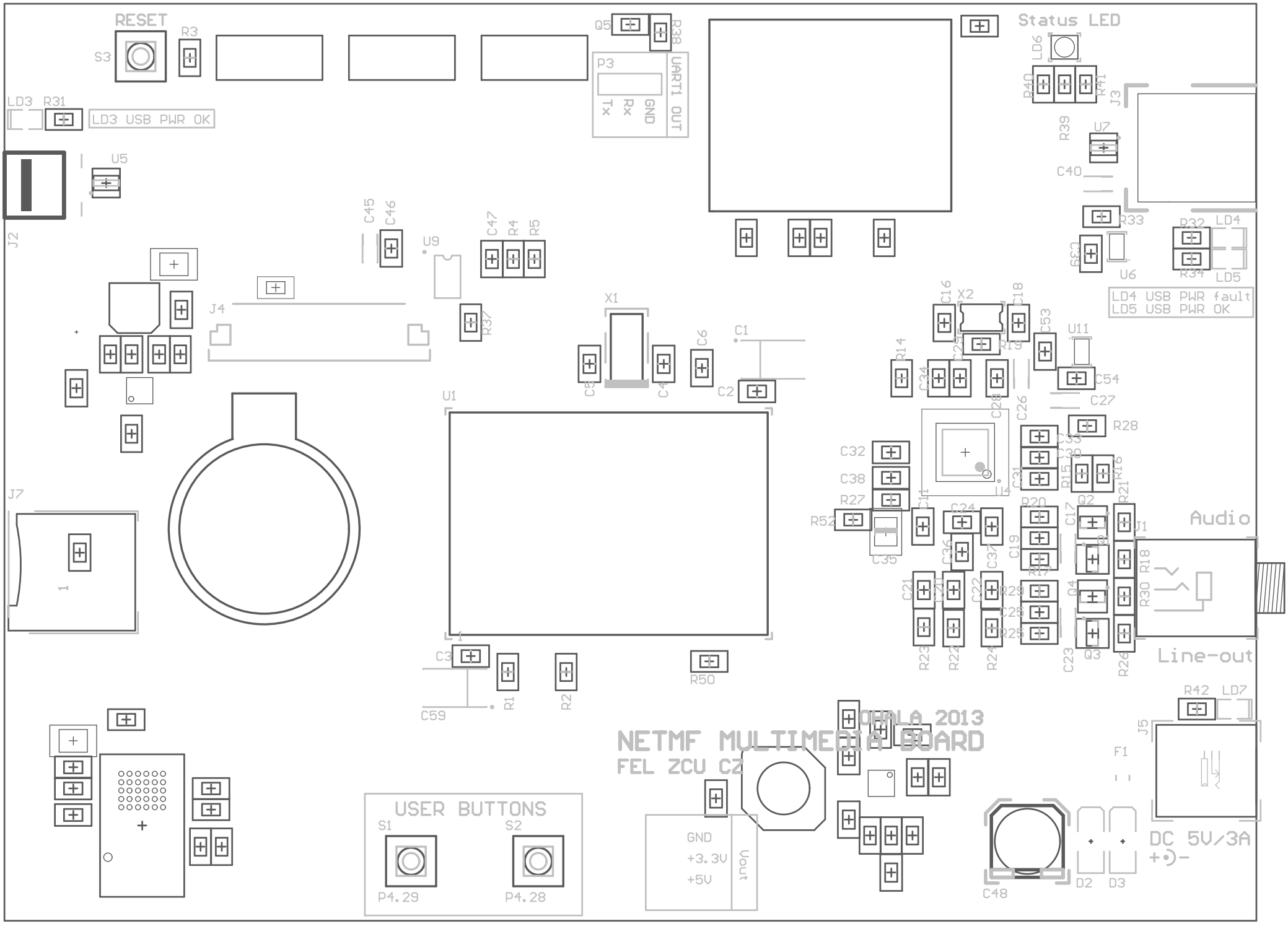
Title: SD_Card.SchDoc		
Size: A4	Number:	Revision: 1.1
Date: 4.5.2013		Sheet* of *
File: C:\!!SVN\..\SD_Card.SchDoc		Drawn By: Ondřej Hála

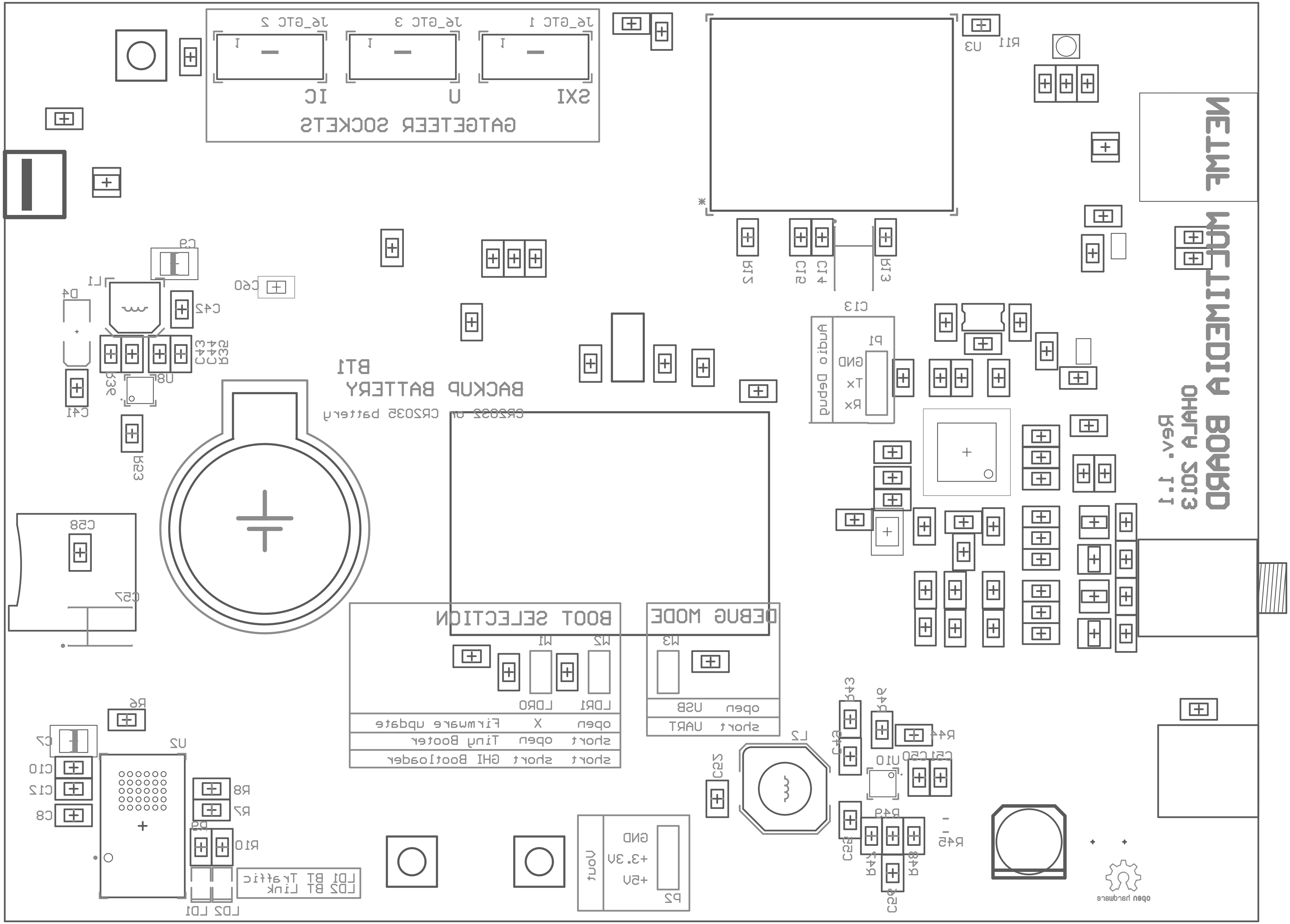

NETMF MULTIMEDIA BOARD
Rev. 1.1
OHALA 2013
J6_GTC 1
J6_GTC 3
J6_GTC 2
SXI
U
IC
GATGETEER SOCKETS
R11
U3
R13
C14
C15
R12
C13
P1
GND
Tx
Rx
Audio Debug
BT1
BACKUP BATTERY
CR2032 or CR2035 battery
C9
C60
L1
D4
C42
R35
C44
C43
U8
R36
C41
R53
C58
C57
DEBUG MODE
W3
open USB
short UART
BOOT SELECTION
W2
W1
LDR1
LDR0
open X Firmware update
short open Tiny Booter
short short GHI Bootloader
R46
R43
R44
C51
C50
U10
C49
L2
C52
R49
R45
R48
R47
C55
C56
GND
+3.3V
+5V
Vout
P2
R6
U2
C7
C10
C12
C8
R8
R7
R9
R10
LD1 BT Traffic
LD2 BT Link
LD2
LD1
open hardware


RESET
S3
R3
LD3 R31
LD3 USB PWR OK
U5
J2
Q5
R38
P3
UART1 OUT
Tx
Rx
GND
Status LED
LD6
R40
R41
J3
R39
U7
C40
R33
C39
U6
R32
LD4
R34
LD5
LD4 USB PWR fault
LD5 USB PWR OK
C45
C46
U9
C47
R4
R5
J4
R37
X1
C5
C4
C6
C1
C2
U1
C16
X2
C18
C29
R19
C53
U11
C54
C27
R14
C34
C28
C26
R28
C33
C30
R15
R16
C32
C38
R27
R52
C35
C11
C24
C31
U4
R20
C17
Q2
Q1
R21
J1
Audio
C36
C37
C19
R17
C21
C20
C22
R29
Q4
C25
R25
R23
R22
R24
C23
Q3
R26
Line-out
J7
C3
C59
R1
R2
R50
R42
LD7
J5
F1
TOP
OHALA 2013
NETMF MULTIMEDIA BOARD
FEL ZCU CZ
USER BUTTONS
S1
S2
P4.29
P4.28
GND
+3.3V
+5V
Uout
C48
D2
D3
DC 5V/3A
+
-


TOP


BOTTOM


RESET
S3
R3
Q5
R38
P3
UART1 OUT
Tx
Rx
GND
Status LED
LD6
R40
R41
J3
LD3
R31
LD3 USB PWR OK
U5
J2
U7
R39
C40
R33
R32
LD4
R34
LD5
U6
LD4 USB PWR fault
LD5 USB PWR OK
C45
C46
U9
C47
R4
R5
J4
R37
X1
C1
C2
C6
C16
X2
C18
C53
U11
C54
C27
R14
C34
C25
C28
C26
R28
C32
C38
R27
R52
C35
C11
C24
C36
C37
C31
R15
R16
R21
Q2
C17
C19
R20
J1
Audio
R18
R30
Q4
Q3
C23
R26
Line-out
J7
U1
C3
C59
R1
R2
R50
R42
LD7
J5
F1
NETMF MULTIMEDIA BOARD
FEL ZCU CZ
OHLA 2013
USER BUTTONS
S1
S2
P4.29
P4.28
GND
+3.3V
+5V
Vout
C48
D2
D3
DC 5V/3A


GADGETEER SOCKETS
J6_GTC 1
J6_GTC 2
J6_GTC 3
IC
U
SXI
NETMF MULTIMEDIA BOARD
OHALA 2013
Rev. 1.1
BT1
BACKUP BATTERY
CR2032 or CR2035 battery
Audio Debug
P1
GND
Tx
Rx
BOOT SELECTION
W1
W2
LDR0
LDR1
open X Firmware update
short open Tiny Booter
short short GHI Bootloader
DEBUG MODE
W3
open USB
short UART
GND
+3.3V
+5V
Vout
P2
LD1 BT Traffic
LD2 BT Link
open hardware