VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# BAKALÁŘSKÁ PRÁCE

Brno, 2016 Juraj Giertl


# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS


# DEKODÉR SÉRIOVÝCH SBĚRNIC NA PLATFORMĚ ARDUINO
ARDUINO BASED SERIAL BUS DECODER

**BAKALÁŘSKÁ PRÁCE**
BACHELOR'S THESIS


**AUTOR PRÁCE** — **Juraj Giertl**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Ondřej Krajsa, Ph.D.**
SUPERVISOR



**BRNO 2016**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

***Student:*** Juraj Giertl ***ID:*** 164271
***Ročník:*** 3 ***Akademický rok:*** 2015/16

**NÁZEV TÉMATU:**

## Dekodér sériových sběrnic na platformě Arduino

**POKYNY PRO VYPRACOVÁNÍ:**

Na platformě Arduino realizujte automatizovaný dekodér sériových sběrnic. Analyzujte možnosti automatického rozpoznávání typu sběrnice, rychlosti přenosu a dalších parametrů. Dekodér bude data zobrazovat na displej a ukládat na paměťovou kartu.

**DOPORUČENÁ LITERATURA:**

[1] PEREA, Francis. Arduino Essentials. Olton Birmingham: Packt Publishing, 2015. ISBN 9781784398569.

[2] PURDUM, Jack. Beginning C for Arduino. Berkeley, CA: Apress, 2015. DOI: 10.1007/978-1-4842-0940-0. ISBN 1484209419.

***Termín zadání:*** 1.2.2016 ***Termín odevzdání:*** 1.6.2016

***Vedoucí práce:*** Ing. Ondřej Krajsa, Ph.D.
***Konzultant bakalářské práce:***

**doc. Ing. Jiří Mišurec, CSc.**, *předseda oborové rady*

## ABSTRAKT

Práca sa zaoberá možnosťami zachytávania a následného dekódovania dát z rôznych sériových zberníc, možnosti automatického rozpoznania zberníc. Uvažované sériové zbernice sú SPI, UART a I$^2$C, ktorých vlastnosti sú podrobne popísané. Práca taktiež podrobne popisuje návrh hardvéru a softvéru prenosného meracieho zariadenia, ktoré túto funkcionalitu umožňuje. Zariadenie je navrhnuté ako shield, pre podporu rôznych vývojových dosiek, ktoré sú kompatibilné s Arduino platformou. Shield obsahuje štyri izolované a päť neizolovaných vstupov pre zachytávanie dát, pamäťovú kartu, ethernetové rozhranie a dva tipy rozhraní pre rôzne displeje.



## KĽÚČOVÉ SLOVÁ

Arduino, dekodér, I$^2$C, logický analyzátor, sériové zbernice, SPI, UART



## ABSTRACT

This project deals with possibilities of capturing and decoding data from different serial buses. Then describe ways of automatic recognition of the buses. The project include hardware and software development of device used for capturing data from different serial buses such as SPI, UART and I$^2$C. Characteristics of these busses are described in detail. Device is design as shield to support Arduino boards. The Shield includes four fully isolated inputs and five non isolated imputs for capturing data, Ethernet interface, SD card and two display interfaces witch supports up to ten displays.



## KEYWORDS

Arduino, dekóder, I$^2$C, logic analyzer, serial busses, SPI, UART



GIERTL, Juraj *Dekodér sériových sběrnic na platformě Arduino*: bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 52 s. Vedúci práce bol Ing. Ondřej Krajsa, Ph.D.

# VYHLÁSENIE

Vyhlasujem, že som svoju bakalársku prácu na tému „Dekodér sériových sběrnic na platformě Arduino“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ..... .....

podpis autora

## POĎAKOVANIE

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Ondřeji Krajsovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.


Brno ..... .....

podpis autora


Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

## POĎAKOVANIE

Výzkum popsaný v tejto bakalárskej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora


EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

# OBSAH

# ZOZNAM OBRÁZKOV

# ZOZNAM TABULIEK

# ÚVOD

Dekodér sériových zberníc je zariadenie, ktoré umožní používateľovi zachytiť a zobraziť dáta, ktoré si medzi sebou posielajú pomocou sériovej zbernice napríklad procesor a analógovo-digitálny prevodník. Takýto dekodér má uplatnenie napríklad pri návrhu digitálnych systémov, kedy je možné komunikáciu odchytiť a následne verifikovať dáta.

Cieľom tejto práce je navrhnúť dekodér, ktorý by bol schopný zachytiť dáta z nastavenej zbernice a prípadne automaticky rozpoznať typ zbernice. Užívateľovi umožniť prehliadanie dát priamo na zaradení a niekoľko ďalších možností uloženia na mikro SD kartu a prenosu do PC cez počítačovú sieť alebo USB.

V nasledujúcich kapitolách je popísaný návrh celého zariadenia. V prvej časti sú podrobne popísane uvažované zbernice. V ďalšej časti je popísaný návrh hardvéru. Popis jednotlivých častí zariadenia je rozdelený do funkčných blokov. Posledná časť práce sa zaoberá návrhom softvéru, kde sú opäť v blokoch popísané algoritmy, ktoré obsluhujú jednotlivé hardvérové periférie a implementované princípy dekódovania zberníc.

# 1 LOGICKÝ ANALYZÁTOR

Dekodér sériových zberníc je jedna z funkcií logických analyzátorov. Logický analyzátor je meracie zariadenie, ktoré umožňuje zobrazovať a zachytávať viacero digitálnych signálov v čase. Na základe zachytených časových priebehov je logický analyzátor schopný dekódovať merané dáta a zobraziť ich v číselnej podobe.

V súčasnej dobe má logický analyzátor implementovaný skoro každý lepší osciloskop. Typicky bývajú označené skratkou MSO – mixed signal osciloscop. Digitálne vstupy týchto osciloskopov sú pripojené priamo k čipu, ktorý obsluhuje tieto funkcie. Príkladom je osciloskop MSO6014A [7] od firmy Keysight Technologies. Je to jeden zo základnejších osciloskopov s dobrým softvérovým vybavením. Osciloskop obsahuje 4 analógové kanály a 16digitálnych kanálov. Šírku pásma 100 Mhz, a hardvérové dekódovanie v reálnom čase. Cena takéhoto MSO sa pohybuje okolo 1000 €.

Existujú aj profesionálne logické analyzátory, príkladom je Tektronix Logic Analyzer TLA6400 [14]. Takéto typy zariadení sú postavené okolo priemyselného počítača, na ktorom zvyčajne beží operačný systém s pripravenou aplikáciou pre spracovanie dát. Základný model TLA6400 má 34 digitálnych vstupov, a na každý vstup má 64 Mb pamäte pre zachytené dáta. Dokáže zachytávať hodinové impulzy s frekvenciou do 667 MHz. Samozrejme s takýmto výkonom rastie aj cena. Popisovaný logický analyzátor začína na cene 10000 €.

Ďalšie z možností sú USB analyzátory. Túto skupinu analyzátorov tvorí veľa rôznych zariadení s rozličnými parametrami v rôznych cenových kategóriách. Príkladom lepšieho USB analyzátoru je Zeroplus LAP-C [21]. Umožňuje merať do 100 MHz, s dobrou hĺbkou pamäte až 32 Kb na kanál. Cena tohoto produktu sa pohybuje približne okolo 100 € kus. Z lacnejších zariadení možno uviesť USB analyzátor od firmy ANE. Tento analyzátor sa vojde do ceny 10 € za kus. Disponuje ôsmimi vstupmi, a dokáže merať signály maximálne do 10 MHz s presnosťou ±42 ns, čo je v dnešnej dobe málo pre solídny digitálny systém. Tento USB analyzátor nemá žiadnu pamäť pre zachytené dáta, pretože zariadenie slúži len ako rozhranie a všetky dáta sa posielajú rovno cez USB do počítača, kde ich spracuje softvér.

Aj keď sa zdá že USB analyzátory s kvalitnejším hardvérom sú dobrá voľba, stále majú jeden problém. Problém je, že nefungujú samostatne, vždy potrebujú počítač so softvérovým vybavením, čo je najväčší problém pri práci v teréne. Aj preto sa táto práca zaoberá návrhom zariadenia, ktoré by malo možnosť pracovať samostatne a ponúknuť zrozumiteľný výstup pre užívateľa.

# 2 POPIS ZBERNÍC

V tejto časti sú popísané najzákladnejšie sériové zbernice, ktoré môžme nájsť v každom digitálnom systéme. V modernej elektronike mikroprocesory pomocou sériových zberníc komunikujú s periférami.

## 2.1 SPI

Serial Peripheral Interface – sériové periférne rozhranie bolo vyvinuté firmou Motorola a táto zbernica sa stala de-facto štandard [19]. Toto viedlo k veľkému rozšíreniu v mikroprocesorových systémoch. Táto zbernica je určená pre komunikáciu na krátke vzdialenosti v rámci jedného zariadenia. Umožňuje plno-duplexný prenos a zavádza Master/Slave typ prenosu. Je špecifická iba jedným zariadením typu Master a môže komunikovať s viacerými zariadeniami typu Slave.
Rozhranie je definované štyrmi vodičmi:

1. MOSI – Master Output Slave Input
2. MISO – Master Input Slave Output
3. SCLK – Serial Clock. Taktovací signál zbernice.
4. CS – Chip Select. Pomocou tohoto signálu zariadenie typu master oznamuje, že chce komunikovať so Slave zariadením. Každé Slave zariadenie musí mať svoj vlastný CS, aby nedochádzalo ku kolíziám.

Komunikácia medzi zariadeniami pracuje tak, že zariadenie typu Master vyberie zariadenie s ktorým chce komunikovať tým, že pripne CS pin Slave zariadenia na logickú nulu. Ďalej na nábežnú, alebo zostupnú hranu taktovacieho signálu vysiela a zároveň prijíma dáta, na vodičoch MISO a MOSI. Po skončení prenosu sa pin CS pripne na logickú jednotku. Formát prenášaných dát je individuálny a záleží na samotnej implementácií tohoto rozhranie.

Rôzni výrobcovia implementujú toto rozhranie ako 3vodičové, kde pre prenos dát slúži len jeden vodič. Takéto rozhranie je často s 3stavovým dátovým pinom, čo umožňuje simplexný prenos dát v oboch smeroch. Prípadne je dátový vstup vyhradený len ako vstup a zariadeniu sa posielajú príkazy, príkladom môže byť digitálno-analógový prevodník ktorému sa posielajú dáta pre nastavenie výstupného napätia.

## 2.2 UART

Ďalšou známou zbernicou je UART – Universal Asynchronous Receiver Transmitter. Táto zbernica je taktiež neoddeliteľnou súčasťou skoro každého mikroprocesora.

Niektoré mikroprocesory disponujú aj viacerými rozhraniami tohoto typu. Rozhranie sa skladá s dvoch dátových vodičov Tx a Rx, ktoré sú vztažené k zemi. Vodič Tx – Transmit je určený na posielanie dát a vodič Rx – Receive je určený na prijímanie dát. Zbernica nepoužíva žiaden taktovací signál, vďaka ktorému by sa posielané dáta mohli synchronizovať. Zariadenia, ktoré medzi sebou komunikujú musia poznať na akej rýchlosti prijímajú dáta, aby ich správne dekódovali. Prenosové rýchlosti zbernice dosahujú 1,5 Mbps. Väčšinou komunikujú iba dve zariadenia.

Formát prenášaných dát je v rámcoch, ktoré začínajú štart bitom, podľa nastavenia nesleduje 5-8 dátových bitov, ďalej párny alebo nepárny paritný bit a 1-2 stop bity.

Tento spôsob prenosu dát používa aj rada ďalších podobných zberníc, priamo z UART vychádza zbernica USART, ktorá zavádza synchrónny prenos dát a dve signalizačné linky CTS – clear to send a RTS – request to send. Zbernica RS-232 rovnaký spôsob prenosu ako predošlé, len používa ±12 V pre prenos. Ďalšia podobná zbernica je RS-422 ktorá na vysielanie a príjem používa diferenčné páry, dosahuje rýchlosť do 10 Mbps pri najlepších podmienkach a maximálna vzdialenosť 1500 metrov [17].

## 2.3 I²C

Inter-integrated Circuit je príklad ďalšej známej sériovej zbernice. Táto zbernica bola navrhnutá firmou Philips[1] začiatkom osemdesiatych rokov minulého storočia [6]. Zbernica bola pôvodne navrhnutá pre rýchlosti 100 kbps a 400 kbps „fastmode“ prenos. Od roku 1998 táto zbernica umožňovala rýchlosť 3,4 Mbps „High-speed“.

Pre komunikáciu zbernica používa dve obojsmerné linky[2] SDA – Serial Data a SCL – Serial Clock. Vstupy/výstupy v zariadeniach sú typu open-drain, preto sa na zberniciach používajú pull-up rezistory. Hodnoty rezistorov závisia na požadovanej rýchlosti zbernice, napájacieho napätia, parazitných kapacít pripojených obvodov a počtu pripojených obvodov. Typická hodnota pull-up rezistorov je v rozmedzí 4–100 kΩ. Zbernica pracuje obvykle s 5 V, alebo 3,3 V logikou.
Na zbernici môže byť pripojených viacero zariadení typu master a viacero zariadení typu Slave. Počet pripojených zariadení môže byť maximálne 128, pretože sa používa 7bitová adresa.

Typická komunikácia na zbernici začína poslaním START bitu nasledovaním adresou zariadenia, s ktorým sa nadväzuje komunikácia, dátami a prenos je ukončený

[1] Dnes je firma známa ako NXP.
[2] preto sa niekedy označuje ako 2-wire interface

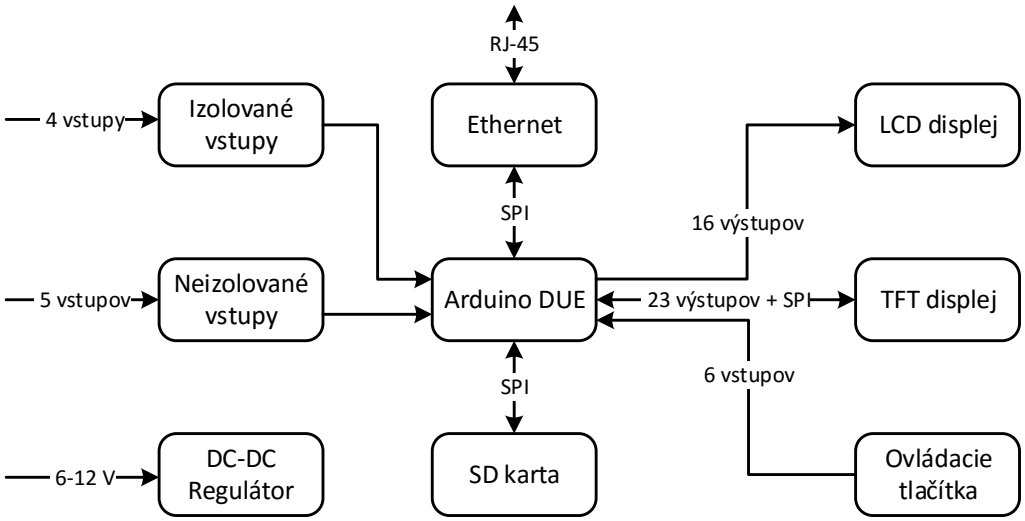
stop bitom. Priebeh ďalšej komunikácie záleží už na samotnej implementácii použitých zariadení, napríklad zariadenie môže odpovedať len potvrdením, alebo celou správou.

## 2.4 Možnosti rozpoznania zberníc

Na základe konkrétnej aplikácie môže byť niekedy ťažké určiť o ktorú zbernicu sa jedná. Preto by musel byť rozhodovací algoritmus navrhnutý tak, aby sa v prvom rade dokázal prispôsobiť rýchlosti na zbernici. Podľa počtu pripojených signálov by bolo možné rozhodnúť o ktorú zbernicu by mohlo ísť. Pre zbernicu SPI sú to štyri, alebo tri signály, ktoré sú zvyčajne v stave logická 1 a pri komunikácií vždy prebieha nejaká zmena na všetkých vodičoch. Rozpoznanie jednotlivých pripojených vodičov zbernice možno rozhodnúť podľa počtu zmien na vodiči a periódou medzi zmenami. Taktovací vodič SCK je špecifický periodickou postupnosťou impulzov počas prenosu. Vodič CS je špecifický iba jednou zmenou na začiatku a na konci prenosu. Problém je s rozhodnutím medzi vodičmi pre dáta MOSI a MISO, keď že na týchto vodičoch sa prenášajú dáta, ktoré sa menia a nieje spôsob ako tieto dva vodiče rozlíšiť.

Zbernice UART a $I^2C$ komunikujú po dvoch signáloch, čiže nieje možné ich od seba odlíšiť na základe aktívnych vodičov. Preto je pre rozpoznanie nutné určiť periodicitu signálov. Zbernica $I^2C$ má na vodiči SCL taktovací signál, ktorý je periodicky opakujúci sa impulz rovnakej dĺžky. Na základe tohoto opakujúceho sa impulzu je možné rozlíšiť $I^2C$ od UART (USART), pretože je málo pravdepodobné, že zbernica UART bude pri komunikácií obsahovať všetky impulzy rovnakej dĺžky. Pri zbernici UART je opäť problém z pohľadu dekodéru určiť, ktorý je vysielací a ktorý je prijímací vodič zbernice, pretože podľa priebehu signálu na vodičoch nie je možné rozlíšiť, ktorý je určený pre vysielanie a ktorý pre príjem.

# 3 HARDVÉR A NÁVRH OBVODU

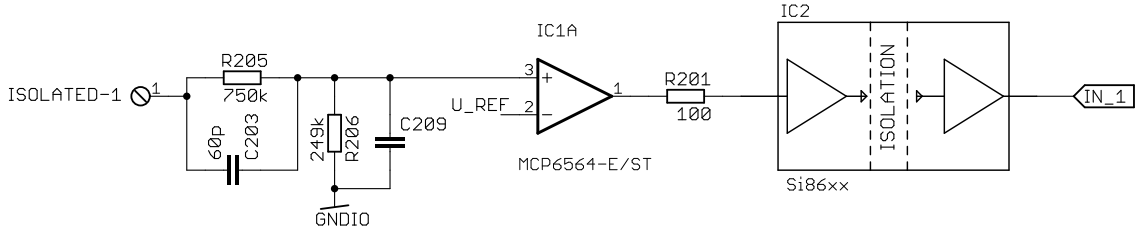



Obr. 3.1: Bloková schéma zariadenia

## 3.1 Návrh vstupného obvodu

Podľa zadania práce vstupy majú byť izolované, začal som hľadať spôsoby, akými by sa tento požiadavok dal realizovať. Ako prvé som uvažoval nad použitím klasických opto-členov, lenže po krátkom hľadaní som zistil, že sú pomalé a pre tento typ aplikácie nevhodné. Preto som vybral štvor-kanálový digitálny izolátor SI8645BA-B-IU [13] od firmy Silicon Labs, ktorý funguje na princípe rádio-frekvenčného vysielača a prijímača. Vďaka tejto aplikácií vybraná súčiastka dosahuje šírku pásma do 150 Mbps, čo je viac ako dosť pre tento typ aplikácie. Vybraté púzdro QSOP-16 zaisťuje izolačné vlastnosti do 2 kV. Ďalej som na vstup zaradil štvoricu rýchlych komparátorov MCP6564-E/ST [10] od firmy Microchip, ktoré obnovujú logické úrovne signálu zo vstupných deličov. Delič na vstupe s vysokou hodnotou odporu som použil preto, aby som čo najmenej zaťažoval meranú zbernicu. Pre komparátory je vytvorené komparačné napätie pomocou operačného zosilňovača, ktorý je zapojený ako buffer a na vstupe ma zapojený delič s polovičným pomerom, ako má vstupný delič. Ukážka navrhnutého vstupu je na obrázku 3.2.

Pre napájanie izolovanej vstupnej časti som použil riešenie od firmy Texas Instruments s obvodom SN6501 [16], ktoré nevyžaduje veľa externých prvkov, je kompletne




Obr. 3.2: Ukážka navrhnutého vstupu

izolované a nemá ani spätnú väzbu. Absencia spätnej väzby znamená, že obvod nedokáže prispôsobiť budenie transformátora na základe výstupného napätia, preto som na výstupe transformátora použil 5,6 V stabilizačnú diódu, ktorá chráni obvody pred prípadnými napäťovými špičkami. Taktiež musí byť pripojená minimálna záťaž, ktorú realizujú samotné obvody a LED dióda, tá tiež signalizuje napájanie obvodu. Z izolovanej časti je možné aj meraný obvod napájať, maximálny dodávaný prúd je limitovaný (150 mA) resetovateľnou poistkou typu PTC.

Kompletná schéma vstupnej časti je priložená v B.3

## 3.2 Implementácia Ethernetového rozhrania

Ethernetové rozhranie je postavené na známom riešení od firmy WIZnet a integrovaného obvodu W5100 [20]. Obvod W5100 má implementovanú TCP/IP sadu protokolov (TCP, UDP, ICMP, IPv4, ARP, IGMP, PPPoE), ochranu pred sieťovými útokmi typu flooding, spoofing, injection. Na fyzickej vrstve podporuje 10BaseT/100BaseTX Ethernet s funkciami Auto-Negotiation (rozpoznávanie rýchlosti) a Auto-MDIX (rozpoznávanie vysielacieho a prijímacieho páru) túto funkcionalitu je možné modifikovať pomocou pinov 63-65 obvodu W5100, ktoré sú pripojené na zem, čo zapína všetky spomínané funkcie. Pre pripojenie sieťového káblu slúži RJ-45 konektor, ktorý má vstavaný oddelovací transformátor. Tento konektor je potom pripojený k obvodu pomocou dvoch diferenčných párov s nominálnou impedanciou 100 Ω. Obvod komunikuje s Arduinom cez už spomínané SPI a CS pin je pripojený tak, aby bola dodržaná spätná kompatibilita s knižnicami, ktoré podporuje platforma Arduino. Obvod môže generovať prerušenie na základe rôznych udalostí, napríklad nadviazanie spojenia, pri odpojení, prijímaní dát a podobne. Prerušenie je generovanie na pine INT (56) a k Arduinu je pripojené cez prepojku na digitálnom pine 8.

Napájanie ethernetového rozhrania je realizované 3,3 V lineárnym stabilizátorom MCP1826T [9] od firmy Microchip. Tento stabilizátor som vybral preto, lebo umožňuje externému systému zapínať a vypínať výstup, teda ak sa ethernetové rozhranie

nepoužíva, je možné ho vypnúť. Taktiež má indikáciu dobrého výstupného napätia a je vyvedená na jeden pin, ktorý je typu otvorený kolektor, preto je použitý pull-up rezistor a je pripojený priamo na reset pin obvodu W5100. Táto funkcia zabezpečuje, že obvod W5100 sa vždy spustí až po doznení prechodových dejov pri zapnutí napájania.

Kompletná schéma je priložená v B.5

## 3.3 Výber a implementácia displejov

Analyzátor má byť schopný zobraziť dáta samostatne bez žiadnych ďalších zariadení. Preto je možnosť k analyzátoru pripojiť hneď niekoľko typov displejov. Implementoval som grafické displeje s radičom KS0108B, napríklad displej MG12864A-SBC/H [8] s rozlíšením 128x64 pixelov a TFT displeje s radičom SSD1289, napríklad modul TFT_320QVT.

Keď že grafický displej s radičom KS0108B pracuje výhradne s 5voltovou logikou, musel som použiť prekladač úrovní z 3,3 V na 5 V. Prekladač úrovní som vybral na základe ceny, počtu bitov, ktoré môže prekladať a rýchlosti. Ako najprijateľnejšia voľba bola použiť obvod 74LVCH16T245 [15] od firmy Texas Instruments. Má šírku 16bitov a tá úplne postačuje k naadresovaniu nielen spomínaného grafického displeja, ale aj mnoho ďalších displejov, z tejto kategórie, napríklad klasické znakové displeje s radičom HD44780. Napájanie displeja je realizované priamo z hlavného 5voltového napájania, jas je možné nastaviť pomocou malého trimra a pre pod-svietenie je použitý predradný odpor s malou hodnotou, približne 5–10 ohmov.

Už spomínané TFT displeje s radičom SSD1289, je možné kúpiť vo forme modulov s veľkosťami zobrazovacej plochy od 3” až 7”. Majú možnosť použiť dotykovú plochu, na zadnej strane displeja je osadený slot pre pamäťovú kartu, prípadne flash pamäť. Displej komunikuje cez 16bitové rozhranie a päť riadiacich signálov, SD karta, flash pamäť a ovládač pre dotykovú plochu sú vyvedené na konektore modulu samostatne a každé z nich používa SPI rozhranie pre komunikáciu. Pod-svietenie displeja je pre každý modul individuálne, napríklad pre 7” displej by bolo treba priviesť 5 V napájanie na pin, ktorý je pri ostatných verziách modulu nezapojený. Je to z toho dôvodu, že pod-svietenie displejov je realizované LED diódami v zapojených v sérií kde potrebujú napájanie do 30 V. Preto tieto moduly majú vlastný napájací zdroj pre pod-svietenie, ktorý je často realizovaný DC-DC meničom. Pod-svietenia je možné nechať trvalo zapnuté, alebo regulovať jas pomocou PWM signálu. Podľa vybraného modulu je možné displej napájať priamo z hlavného 5 V zdroja, alebo z 3,3 V lineárneho stabilizátora.

Schéma pripojenia displejov je priložená v B.2.

## 3.4 Hlavné napájanie

Napájanie celej dosky je realizované pomocou DC-DC znižujúceho spínaného zdroja, ktorý je realizovaný pomocou integrovaného regulátora ADP2303ARDZ-5.0 [1] od firmy Analog Devices. Tento obvod spína na frekvencií 700 kHz, maximálne vstupné napájanie je 20 V výstupné napätie je možne nastaviť deličom. Obvod dokáže dodávať výstupný prúd maximálne 3 A. Pri výstupnom napätí 5 V dosahuje účinnosť 90%. Firma Analog Devices má veľmi dobre spracované katalógové listy k svojim produktom a to zjednodušilo výber súčiastok. Vstupné napájanie som zvolil podľa odporúčaného rozsahu vstupného napätia pre Arduino DUE čo je 7-12 V. Na základe spomínaných požiadavkov som vybral cievku s nominálnou indukčnosťou 7 $\mu$H $\pm$ 30 % maximálnym prúdom 4,35 A, saturačným prúdom 3,62 A a s malou hodnotou DC odporu. Cievka je tienená a s malou výškou 3,8 mm čo zaručuje že cievka nebude zasahovať do častí obvodu Arduino dosky, ktorá bude pripojená presne nad napájacím zdrojom. Dióda pre spínaní zdroj je vybraná z ohľadom na strednú hodnotu prúdu, ktorý ňou môže pretekať pri maximálnej záťaži na výstupe spínaného zdroja. Vstupný a výstupný filtračný kondenzátor, bol vybraný na základe odporúčania výrobcu, sú to keramické kondenzátory s malou hodnotou ESR.

Kompletná schéma napájacieho zdroja je priložená v B.4 spolu s ostatnými napájacími obvodmi pre jednotlivé časti obvodu.

## 3.5 Arduino, SD karta a tlačítka

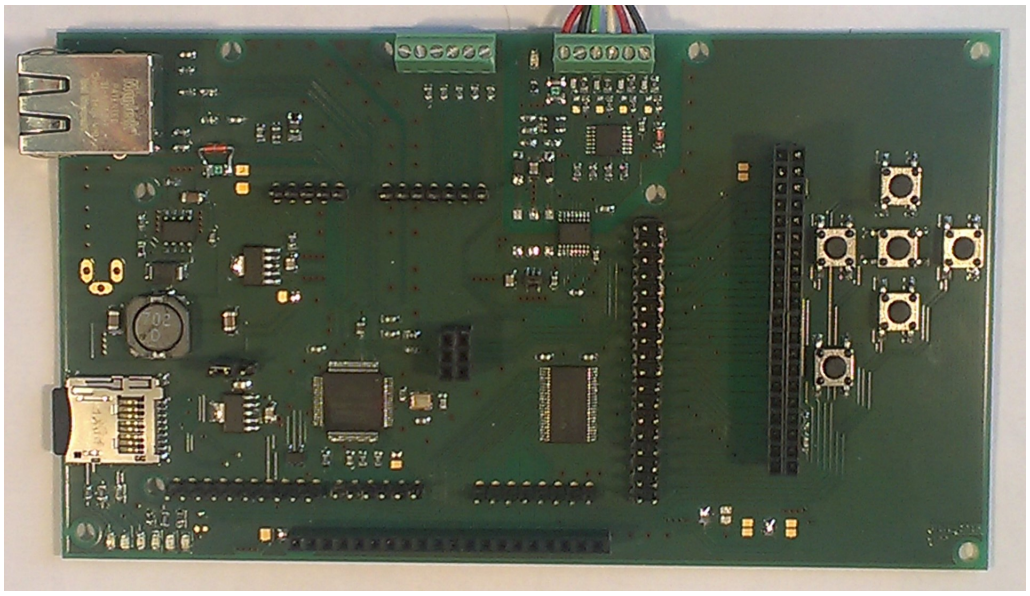
Z možností, ktoré platforma Arduino ponúka som vybral práve Arduino DUE, pretože je osadené mikroprocesorom ATSAM3X8E [5] od firmy Atmel. Mikroprocesor je postavený na 32bitovej architektúre ARM Cortex-M3. Jadro procesora je taktované pomocou PLL na 84 MHz, pamäť pre program je typu flash a je rozdelená do dvoch bánk po 256 KB, pamäť pre dáta je typu SRAM s veľkostnou 64+32 KB. Mikroprocesor disponuje so 103 plne konfigurovateľnými vstupno-výstupnými linkami. Mapovanie pinov, pre vývojovú dosku je priložené v prílohe A.

Ďalšia zvažovaná možnosť bola použiť Arduino Mega ktoré je osadené mikroprocesorom ATmega2560 [4] od firmy Atmel. Je to 8bitový procesor postavený na architektúre AVR. Jadro procesora je taktované maximálne na 16 MHz, pamäť programu je typu flash s veľkostnou 256 KB a pamäť pre dáta typu SRAM s veľkostnou iba 8 KB. Práve malá taktovacia frekvencia mikroprocesora by neumožňovala spracovávať rýchlejšie zbernice a pamäť SRAM by tiež nemusela stačiť pre uloženie potrebných dát.

Ovládanie zariadenia som zvolil pomocou tlačidiel, ktoré sú rozmiestnené do kríža, ktoré budú slúžiť na posuv v menu a podobne. V strede tohoto kríža je tla-

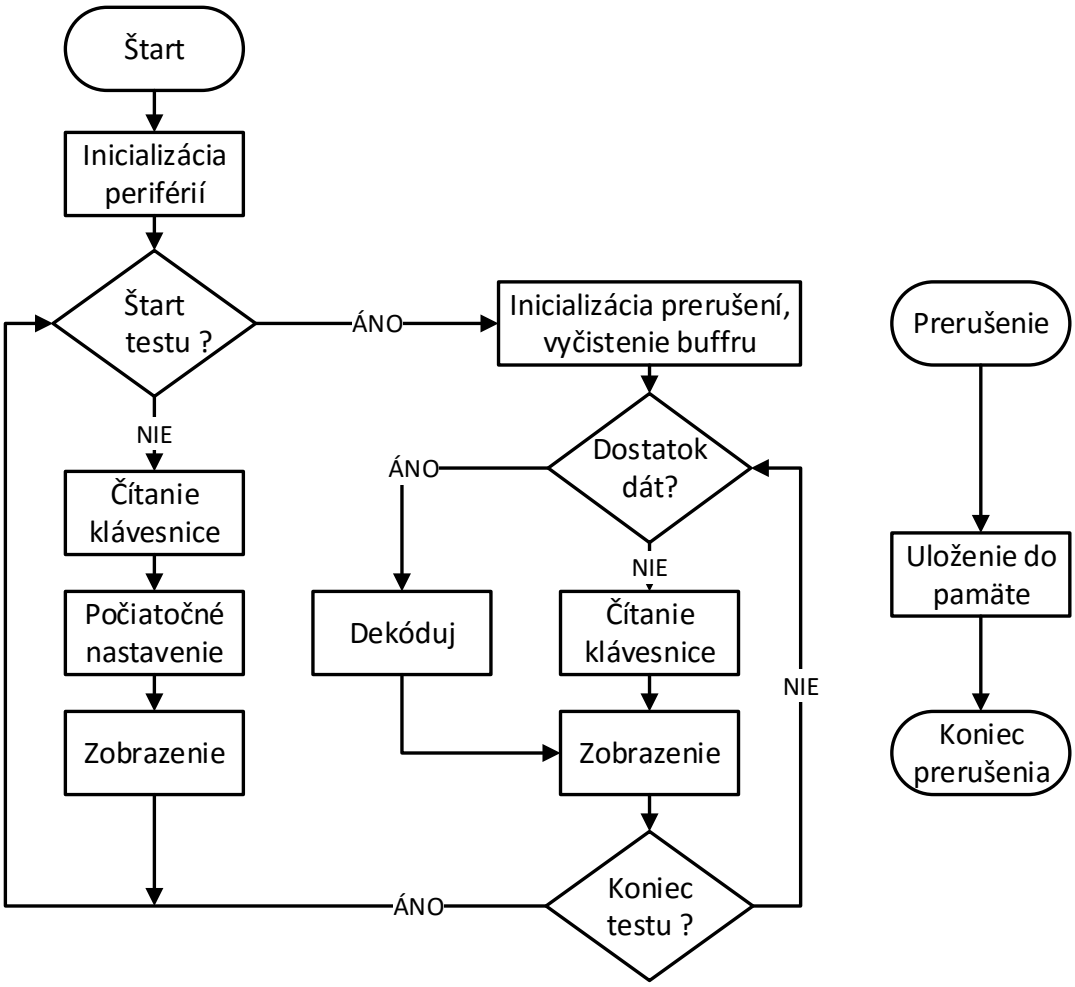
čidlo, ktoré bude slúžiť na potvrdzovanie a posledné tlačidlo bude slúžiť pre návrat z menu. V prípade použitia displeja s dotykovou plochou nebude užívateľovi umožnené používať tlačítka, iba dotykovú plochu pre ovládanie dekodéru. Dekodér je vybavený aj vlastnou SD pamäťovou kartou, ktorá je taktiež pripojená k zbernici SPI. Pamäťová karta primárne slúži pre uloženie dekódovaných dát v textovom formáte a konfiguračných súborov pre celé zariadenie.

Schéma pripojenia vývojovej dosky Arduino DUE, SD karty a tlačidiel je priložená v B.1.



Obr. 3.3: Osadená DPS dekodéru

# 4 SOFTVÉR




Obr. 4.1: Vývojový diagram softvéru pre Arduino

## 4.1 Architektúra programu

Zdrojový kód programu je rozdelený po blokoch do niekoľko súborov. Podľa názvu jednotlivých súborov možno určiť, akú sadu funkcií obsahuje. Napríklad súbor *uartHandler.ino* obsahuje funkcie pre dekódovanie a spracovanie dát zo zbernice UART. Podrobnejší popis je uvedený v podkapitole 4.2.2.

Program je navrhnutý ako stavový automat. Z uvedeného vývojového diagramu možno vidieť, že program sa štandardne môže nachádzať v dvoch základných stavoch, stav dekódovanie a stav pripravený. V stave pripravený sa užívateľovi zobrazujú

základne nastavenia pre konkrétne zvolenú zbernicu a to hlavne pripojenie vstupov zariadenia k zbernici. V tomto stave má užívateľ možnosť vstúpiť do nastavovacieho menu, alebo spustiť dekódovanie.

Pri vstupe do nastavovacieho menu je na prvej stránke užívateľovi umožnené vybrať protokol a základné nastavenia pre konkrétny protokol. Na ďalšej karte menu je možnosť nastaviť ukladanie a výpis dekódovaných dát. Tretia a posledná karta menu je určená pre ostatné nastavenia a to zapínanie Ethernetu, DHCP klienta a nastavenie intervalu obnovovania zobrazovaných dát na displeji.

Pri spustení dekódovania sa zariadenie chová podľa predošle nastavených parametrov. Pokiaľ nie je zapnuté zobrazovanie dekódovaných dát na displej, zariadenie užívateľa na displeji informuje, kam sú informácie zasielané. Ak je zapnuté zachytávanie na mikro SD kartu, zobrazuje sa aj názov súboru, do ktorého sa zachytené dáta ukladajú. Užívateľ je taktiež o spustenom zachytávaní dát informovaný pomocou blikajúcej zelenej LED diódy v dolnej ľavej časti zariadenia. Na ľavo je ďalej červená dióda, ktorá blikne pri vyčítaní zachytených dát z vyrovnávacej pamäte a následnom uložení na mikro SD kartu, vypísaní na terminál. Ďalšia štvorica diód informuje užívateľa o stave ethernetového rozhrania.

Premenné tohoto stavového automatu sú združené v globálnej štruktúre s názvom *DeviceSetup*, ktorej definíciu možno nájsť na začiatku hlavného súboru *DecoderBasic.ino*. Okrem premenných pre obsluhu stavov zariadenia, táto štruktúra obsahuje aj premenné pre obsluhu vykresľovania dát na displej, premenné pre obsluhu ethernetového rozhrania a premenné pre obsluhu mikro SD karty. Význam a účel jednotlivých premenných je popísaný v nasledujúcich blokoch, ktoré sa podrobne venujú softvérovej obsluhe jednotlivých hardvérových častí.

Celý zdrojový kód programu možno nájsť na priloženom CD v archíve *DecoderBasic.zip*.

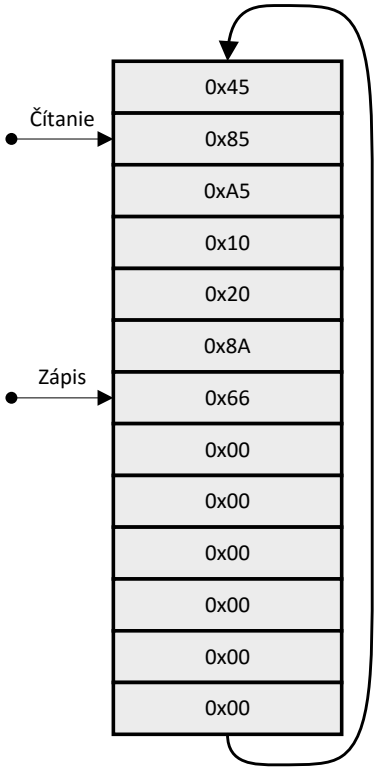
## 4.2 Dekódovanie sériových zberníc

Pre každú zbernicu som vytvoril samostatný súbor napríklad *spiHandler.ino*, v ktorom sú základné funkcie pre zachytávanie a dekódovanie dát. Každý súbor obsahuje trojicu hlavných funkcií, ktoré obsluhujú štart dekódovania, proces dekódovania a koniec dekódovania. Funkcia pre štart a koniec dekódovania sa zavolajú len jeden krát a funkcia pre proces dekódovania je pri spustení dekódovania volaná nepretržite. V hlavnom súbore *DecoderBasic.ino* možno nájsť funkcie *prepareDecode()*, *decoding()* a *endDecode()*, ktoré podľa globálnej premennej *DeviceSetup.protokol* obsluhujú konkrétnu akciu pre aktuálne nastavenú dekódovanú zbernicu. Tieto funkcie sa volajú pomocou troch stavových premenných *startDecode*, *setupDecode* a *endDe-*

*code*, ktoré sú opäť súčasťou štruktúry *DeviceSetup*. Premenné sú typu bool, čiže môžu obsahovať len dve hodnoty a to TRUE a FALSE. Pri štarte dekódovania sa premenné *startDecode* a *setupDecode* jednorázovo nastavia na TRUE, pričom premenná *setupDecode* sa po správnom vykonaní funkcie *prepareDecode()* nastaví na FALSE. Pomocou premennej *startDecode* je možne dekódovanie pozastaviť a opätovne spustiť. Nakoniec premenná *endDecode* pomocou ktorej za jednorázovo zavolá funkcia *endDecode()*.

### 4.2.1 Kruhová vyrovnávacia pamäť

Zariadenie pri načítaní dát používa kruhovú vyrovnávaciu pamäť. Veľkosť vyrovnávacej pamäte je umožnené zmeniť pred štartom samotného dekódovania. Pamäť je dynamicky alokovaná pomocou funkcie *calloc()*, túto funkciu som použil práve preto, lebo zaistí inicializáciu pamäťového bloku. Pre zápis a čítanie do bloku pamäte sa používanú dva pointre, ktoré ukazujú na miesto zápisu novej hodnoty a miesto čítania poslednej hodnoty. Pointer pre zápis je ošetrený tak, aby nikdy neprepisoval hodnoty, ktoré ešte nie sú vyčítané z pamäte. Podobne je ošetrený pointer pre čítanie, ten v prípade vyčítania všetkých dát z pamäte nepreskočí pointer pre zápis.




Obr. 4.2: Ukážka alokovanej pamäte

Pre obsluhu vyrovnávacej pamäte sú vytvorené dve funkcie *storeInRingBuff()* a *readFromRingBuff()*, ktoré možno nájsť v súbore *helpFunctions.ino*. Tieto funkcie slúžia pre prácu s 8bitovými dátami, v súbore možno nájsť aj modifikované funkcie , ktoré pracujú zo 16bitovými dátami. Vstupné parametre funkcií sú pointer na predošle alokovanú pamäť a jej veľkosť, pointre pre čítanie a zápis. Pri funkcií pre zápis sa ešte predávajú samotné dáta a ďalší pointer, ktorý slúži ako počítadlo celkovo uložených dát.

### 4.2.2 Dekódovanie zbernice UART

Obsluha zbernice UART je implementovaná pomocou hardvérových periférií mikroprocesora. Po prejdení už implementovaných funkcií Arduino IDE som sa rozhodol použiť len funkciu pre inicializáciu periférií. Keďže táto funkcia nastavuje aj obsluhu prerušení, ktoré sa vyvolávajú pri prijatí dát na zbernici, musel som pôvodné obslužné funkcie nahradiť svojimi vlastnými, ktoré som už pripravil pre svoju aplikáciu. Pôvodné obslužné funkcie možno nájsť v zložkách implementácie Arduino DUE v súbore *variant.cpp*. Nové obslužné funkcie pre prerušenie možno nájsť na začiatku súboru *uartHandler.ino*. Podľa nastavenia vstupov je potom už len zachytená hodnota zapísaná do konkrétnej vyrovnávacej pamäte.

Pre integritu dát a lepšiu čitateľnosť kódu som pre zbernicu UART vytvoril štruktúru, ktorá obsahuje premenné pre obsluhu pamätí a pomocné premenné pre nastavenie rýchlosti prenosu zbernice, počet prenášaných bitov, paritné a stop bity, číslo súboru do ktorého sa dáta ukladajú, počítadlo zachytených dát a dvoj-stavovú premennú *rxCapture* pre spustenie zachytávania na oboch linkách (RX a TX). Túto štruktúru možno nájsť v *definitions.h* je zadefinovaná ako prototyp a v súbore *uartHandler.ino* je podľa tohoto prototypu vytvorená štruktúra *uartCapture*. Tento spôsob implementácie som zvolil preto, lebo z touto štruktúrou sa pracuje naprieč celým programom a to pri zobrazovaní informácií o stave zachytávania, ukladaní a čítaní nastavených dát z mikro SD karty a nastavovaní konkrétnych parametrov pre zachytávanie. Rovnaká myšlienka je použitá aj pri ostatných zberniciach, ale s menšími modifikáciami, ktoré sú popísané v podkapitolách nižšie.

Zbernica UART obsahuje dva dátové vodiče RX pre príjem a TX pre posielanie dát primárne medzi dvoma komunikačnými prvkami. Z pohľadu dekodéra sú oba dátové vodiče pripojené na RX piny mikroprocesoru, teda používam dve hardvérové rozhrania. Stále je ale možnosť, že zariadenia budú medzi sebou komunikovať len jednosmerne, alebo stačí zachytávať len výstup TX, v tom prípade stačí mať zapnuté len jedno hardvérové rozhranie. Pri spracovávaní dát len z jednej dátovej linky sa dáta spracúvajú kontinuálne pomocou funkcie *captureUart()*, nečaká sa na naplnenie vyrovnávacej pamäte do určitej hodnoty zaplnenia pamäte[1]. Keď je vyrovnávacia pamäť prázdna funkcia pre čítanie z pamäte *uartTxMemoryReadHandler()* vráti hodnotu $-1$. Na ukážke 4.1 možno vidieť konkrétnu funkciu pre priame vyčítanie zachytených dát s vyrovnávacej pamäte a následné spracovanie dát, podľa nastavenia zariadenia.

[1]Hodnota zaplnenia pamäte je inak pri UART nastavená na polovicu veľkosti vyrovnávacej pamäte

Výpis 4.1: Vyčítanie vyrovnávacej pamäte

```
1  bool captureUart(void)
2  {
3    int inByte;
4    inByte = uartTxMemoryReadHandler();
5    if (inByte != -1)
6    {
7      if (uartCapture.txSdStoring)
8        txData.write(inByte);
9      if (DeviceSetup.TerminalCapture)
10       Serial.write(inByte);
11 #ifdef ETH_ON
12     if (DeviceSetup.TelnetCapture)
13       if (DeviceSetup.client)
14         server.write(inByte);
15 #endif
16   }
17   return true;
18 }
```

V prípade ukladania dát na mikro SD kartu sa pri tomto type zachytávania dát pomocou funkcie *prepareUart()* otvorí aj súbor s príslušným názvom, do ktorého sa zapisujú jednotlivé znaky z vyrovnávacej pamäte. Aby súbor neostal prázdny, treba ho zavrieť, to je ošetrené vo funkcií *stopUart()*. Pred samotným zavretím súboru sa ešte zapíše zvyšok vyrovnávacej pamäte, ktorá sa nestihla spracovať pomocou funkcie *captureUart()* a to pomocou modifikovanej funkcie *captureUartBeforeEnd()*, ktorá vyčíta všetky dáta z vyrovnávacej pamäte až do konca. Vyčítané dáta pomocou tejto funkcie sú opäť zobrazené aj na ostatné periférie, podľa nastavenia zariadenia.

Pri zvolení možnosti zachytávania na oboch linkách (RX a TX), som zvolil sekvenčný prístup k spracovaniu dát. Viedla k tomu hlavne nemožnosť mať dva súčasne otvorené súbory na mikro SD karte, v prípade rozdelenia dát z každej linky zvlášť. Ďalej možnosti rozlíšenia dát v textových výstupoch na termináli a cez Telnet. Preto sa dáta vypisujú po blokoch, ktoré sú vždy označené, z ktorého dátového vodiča boli zachytené. Veľkosť vypisovaného bloku je nastavená ako polovičná veľkosť vyrovnávacej pamäte, ktorú nastaví užívateľ. Alokovanie vyrovnávacej pamäte sa opäť uskutoční vo funkcií *prepareUart()*, tento krát sa alokujú tri premenné, dve sú určené pre zachytené dáta a tretia ako pomocná. Pomocná premenná neskôr vo funkcií *captureUartRxTx()* slúži pre vyčítanie bloku dát jednej z vyrovnávacích pamätí akonáhle je konkrétna vyrovnávacia pamäť zaplnená na nutné minimum (jej polovička). Ďalej sa dáta z pomocnej premennej zapíšu na terminál, Telnet, alebo

do súboru. Pri výpise dát na terminál a Telnet sa bloky dát označia a oddelia jednoduchým textom tvoreným pomlčkami a názvom bloku dát. V prípade zápisu do súboru sa konkrétny súbor vždy otvorí, jednorázovo za zapíše blok dát a súbor sa ukončí. Keďže proces ukladania dát na mikro SD kartu je časovo náročný[2], je pri ukladaní dát lepšie zvoliť väčšiu vyrovnávaciu pamäť, aby sa predišlo zahadzovaniu dát. Pri ukončení zachytávania dát sa prevedie vyčítanie vyrovnávacích pamätí pomocou modifikovanej funkcie *captureUartRxTxBeforeEnd()*.

### 4.2.3 Dekódovanie zbernice SPI

Dekódovanie zbernice SPI je realizované pomocou prerušenia, ktoré je nastavené na taktovací pin zbernice. Prerušenie je možno vyvolať na nábežnej, alebo zostupnej hrane. V prerušení sa otestuje CS pin a ak je na logickej hodnote 0, tak sa prečítajú stavy dátových pinov a zapíšu sa do lokálnych statických premenných, ktoré sa pri ôsmom správnom prerušení zapíšu do kruhovej vyrovnávacej pamäte. Veľkosť dátového rámca je teda implicitne nastavená na osem bitov, veľkosť možno ešte podľa požiadavkov upraviť.

Konkrétnu implementáciu možno nájsť v súbore *spiHandler.ino*, kde možno nájsť opäť funkcie pre prípravu, proces a ukončenie dekódovania. Funkcia pre prípravu vynuluje stavové premenné v štruktúre, ktorá bola vytvorená pre dekódovanie zbernice SPI. Ďalej sa preložia vstupy nastavené užívateľom na konkrétne hardvérové vstupy pomocou funkcie *translateImput()*, alokuje sa vyrovnávacia pamäť pre zachytávané dáta a nakoniec sa priradí prerušenie na taktovací pin zbernice.

Proces dekódovania je veľmi podobný ako sekvenčný prístup pri zbernici UART s tým rozdielom, že dáta sa formátujú a užívateľ môže zvoliť medzi desiatkovou a šestnástkovou číselnou sústavou. Pre zobrazenie dát na terminál, mikro SD kartu a telnet je nastavené zobraziť iba šestnásť dátových buniek na riadok pre lepšiu čitateľnosť. pri ukončení dekódovania sa vypne prerušenie vyčítajú sa zvyšné dáta z vyrovnávacích pamätí a keď sa dáta ukladali na mikro SD kartu, tak sa aktualizuje číslo pre označenie súboru.

### 4.2.4 Dekódovanie zbernice I²C

Dekódovanie zbernice I²C sa vykonáva pomocou troch prerušení na troch rôznych vstupoch dekodéra, pričom dva vstupy sú spoločne pripojené na dátovú linku zbernice. Je to z toho dôvodu, aby dekodér vedel spoľahlivo rozpoznať START a STOP bit na zbernici. START bit je definovaný ako zmena dátovej linky z log. 1 na log.

[2]Len otvorenie a uzavrenie súboru trvá približne 30 ms.

0, pričom taktovacia linka je na úrovni log 1, teda sa detekuje zostupná hrana dátového signálu. STOP bit je definovaný ako nábežná hrana, keď je taktovacia linka zbernice na log 1. Pri prenose dát sa teda zmena dátového signálu musí vykonať iba v prípade, keď je taktovacia linka na hodnote log 0. Takto je definovaný základný prenos po sériovej zbernici $I^2C$. Podľa tohoto popisu sú implementované aj obsluhy prerušení vyvolané týmito stavmi na zbernici. Dáta sú ukladané v podobe 8bitových rámcov do vyrovnávacej pamäte.

Rovnako ako predošlé sériové zbernice, pre zbernicu $I^2C$ som vytvoril samostatný súbor *i2cHandler.ino*, kde sa nachádzajú funkcie pre prípravu, proces a ukončenie dekódovania. Proces dekódovania podobný ako kontinuálny proces spracovávania dát pri zbernici UART, rozdiel je len v interpretácií a formátovaní dát. Algoritmus pre dekódovanie dát dokáže rozpoznať START a STOP bit na zbernici. Na základe toho sa vo výstupných dátach tieto stavy značkujú pomocou písmen „S“ pre START bit a „E“ pre STOP bit. Pri rozpoznaní STOP bitu sa automaticky prejde na nový riadok pri zobrazovaní dát na Telnet, terminál a aj pri ukladaní do súboru.

Funkcie pre prípravu a stop dekódovania dodržujú rovnakú myšlienku ako pri zbernici SPI, s tým rozdielom, že pre zbernicu $I^2C$ je opäť vytvorená samostatná štruktúra *i2cStruct*. Táto štruktúra obsahuje špecifické premenné pre dekódovanie $I^2C$.

## 4.3 Displej a zobrazenie dát

Z podporovaných displejov je dekodér osadený 3,2” TFT farebným grafickým displejom s rozlíšením 320x240 pixelov s dotykovou plochou. Pre programovú obsluhu displeja som použil už vytvorenú knižnicu UTFT (verzia 2.75), ktorá je dostupná na [18]. Keďže pri návrhu DPS som pre rozhranie displeja použil zapojenie totožné ako pre CTE TFT LCD/SD Shield pre Arduino Due, je nutné zadefinovať použitie tohoto shieldu v súbore *HW_ARM_defines.h*, ktorý sa nachádza v pod-adresároch knižnice UTFT. Funkcie pre obsluhu displeja sú znovu rozdelené do niekoľkých súborov, pre lepší prehľad. V súbore *displayControl.ino* možno nájsť funkciu pre inicializáciu displeja a dotykovej plochy, ktorá sa volá len raz pri spustení dekodéra. Druhá funkcia v tomto súbore *displayLoop()* v prvej časti zabezpečuje vykresľovanie dát na displeji podľa hodnôt premenných v stavovom automate. V druhej časti sa kontroluje, či bol detekovaný dotyk na dotykovej ploche, pokiaľ áno, vykoná sa obslužná funkcia podľa premenných stavového automatu. Z dôvodu toho, že ku každej vykreslenej obrazovke na displeji treba ekvivalentnú funkciu, ktorá bude obsluhovať ovládacie prvky vykreslené na displeji, som vždy vytvoril dvojicu funkcií. Názov jednej z funkcií vždy začína *draw* pre vykreslenie a názov druhej funkcie začína

*check* pre kontrolu a obsluhu ovládacích prvkov[3]. Funkcie s názvom *check* taktiež nastavujú premenné stavového automatu.

### 4.3.1 Optimalizácia vykresľovania

Po oživení displeja sa ukázalo, že vykresľovanie textu na displeji je veľmi pomalé a teda aj vykresľovanie celej stránky. Preto som v štruktúre *DeviceSetup* vytvoril ďalšie dve premenné *change* a *reDraw*, ktoré ošetrujú vykresľovanie a to tak, že premenná *reDraw* slúži len pre vykreslenie statického textu a obrysov na displeji. Premenná *change* ošetruje len meniaci sa text na stránke. Aby som určil, kde sa na stránke meniaci text nachádza, rozdelil som stránku na maticu o dvoch stĺpcoch a päť riadkov[4]. Každá bunka matice má v premennej *change* vyhradený svoj vlastný bit, to je výhodné pri menení viacero prvkov naraz. Tento prístup síce zaručil svižnosť zobrazovania v rámci jednej stránky, ale pri prepínaní medzi stránkami sa ďalšia stránka prekresľovala dlho. Preto som začal hľadať prečo je vykresľovanie textu také pomalé. Problém bol v tom, že text na displeji otáčam o 180 ° a toto otáčanie v použitej knižnici obsluhované funkciou, ktorá premiesťovala každý pixel vykresľovaného znaku, podľa zadaného uhlu. Preto som sa pokúsil upraviť použitú knižnicu.

Úprava spočívala v nájdení „pomalej“ funkcie a dôvodu, prečo je pomalá. V knižnici som našiel funkciu *UTFT::rotateChar()*, ktorá vykresľovanie zo zadaným uhlom vykonáva. Táto funkcia je napísaná univerzálne, takže je možné vykresľovať pod akýmkoľvek uhlom. Funkcia pomocou goniometrických funkcií pre každý pixel znaku zo znakovej sady vypočítala nové súradnice pre tento pixel a následne ho zobrazila na displeji. V prvom kroku som upravil vzorce pre výpočet nových súradníc, tak že som minimalizoval počet matematických operácií. Vo vzorcoch sa zbytočne vykonávali goniometrické funkcie pre každý pixel, tieto operácie som vypočítal ako konštantu pred samotným cyklom pre výpočet nových polôh. Táto úprava zrýchlila vykresľovanie textu len mierne. Časovo najnáročnejší príkaz v tejto funkcií je nastavovanie súradníc pre nový pixel. Táto funkcia zapisuje hodnoty priamo do registrov modulu displeja a pri použití znakovej sady o veľkosti $16^2$ pixelov sa táto funkcia volá 256-krát a to len pre jeden znak. Keď že pre moju aplikáciu požadujem svižné vykreslenie textu len pod uhlom 180 ° vo funkcií som vytvoril podmienku, ktorá práve pri uhle 180 ° vykreslí text priamo zo znakovej sady, pri ostatných uhloch použije pôvodnú funkciu. Vykreslenie priamo zo znakovej sady je riešené podobne ako vo funkcií *UTFT::printChar()*, len som upravil indexovanie polôh jednotlivých častí

[3] Napríklad funkcia *drawSetup1()* vykreslí druhú stránku v menu a funkcia *checkSetup1()* reaguje na vstup užívateľa.

[4] Toto rozdelenie je jasne vidno v nastavovacom menu.

znaku, tak aby bola zachovaná spätná kompatibilita s pôvodnou funkciou a správne vykreslenie znaku. Výhoda tohoto prístupu je, že súradnice sa nastavia len raz a od týchto súradníc sa potom vykreslí celý znak zo znakovej sady. Teda namiesto nastavovania súradníc pre každý pixel znaku sa súradnice nastavia len raz pre znak, čo vo výsledku ušetrí 255 prístupov do registru modulu displeja. Táto úprava viditeľne zrýchlila vykreslenie textu a tým sa zlepšila aj odozva zariadenia. Upravenú knižnicu možno nájsť na priloženom CD.

### 4.3.2 Zobrazenie dát

Pre zobrazenie dekódovaných dát som vytvoril funkcie, ktoré vyčítajú z kruhovej vyrovnávacej pamäte vždy najaktuálnejšie dáta nezávisle od spracovávania dát pre ostatné výstupy. Formát, akým sa dáta zobrazia záleží na vybranej zbernici. Pri zberniciach SPI a $I^2C$ môže užívateľ zvoliť medzi hexadecimálnym a dekadickým zobrazením. Veľkosť zobrazovacej plochy je desať riadkov a v každom riadku je možno zobraziť devätnásť znakov. Zobrazené čísla sú rozdelené medzerou a na displeji sú usporiadané tak, aby sa zmestilo čo najviac dát. Pri zobrazovaní hexadecimálnych čísiel sa zobrazí maximálne šesťdesiat čísiel a pri dekadickom päťdesiat 8bitových čísiel. Obrazovka sa automaticky aktualizuje ak sa vo vyrovnávacej pamäti nachádza dostatočný počet nových dát.

Zobrazenie dát dekódovaných zo zbernice UART som realizoval v podobe ASCII znakov. Opäť sa zobrazia vždy najaktuálnejšie dáta z vyrovnávacej pamäte. Tieto dáta sú ešte pred samotným zobrazením formátované s dôvodu ušetrenia miesta na displeji. Formátovanie spočíva v nahradení postupnosti bielych znakov jedným. Toto formátovanie je výhradne len pre displej, ostatné výstupy zostanú nezmenené. Aktualizácia sa v tomto prípade vykonáva podľa intervalu obnovenia displeja, ktorý sa dá nastaviť na poslednej stránke menu.

## 4.4 Užívateľské rozhranie

Užívateľské rozhranie je navrhnuté tak, aby bolo možné zariadenie čo najrýchlejšie nastaviť a spustiť dekódovanie.

Po spustení zariadenia je užívateľ informovaný o správnosti inicializácie mikro SD karty. Pokiaľ inicializácia zlyhala, sú možnosti práce s kartou obmedzené a to tak, že nie je možné spustiť zachytávanie dát na kartu a taktiež zmenené nastavenia zariadenia sa neuložia. V nastaveniach sa to prejaví tak, že text nastavovacieho tlačítka má červenú farbu. Následne je užívateľ informovaný o stave ethernetového rozhrania a po uplynutí dvoj-sekundového oneskorenia program skočí do hlavnej slučky programu. Na úvodnej obrazovke, ktorú je možno vidieť na obrázku 4.3a sú

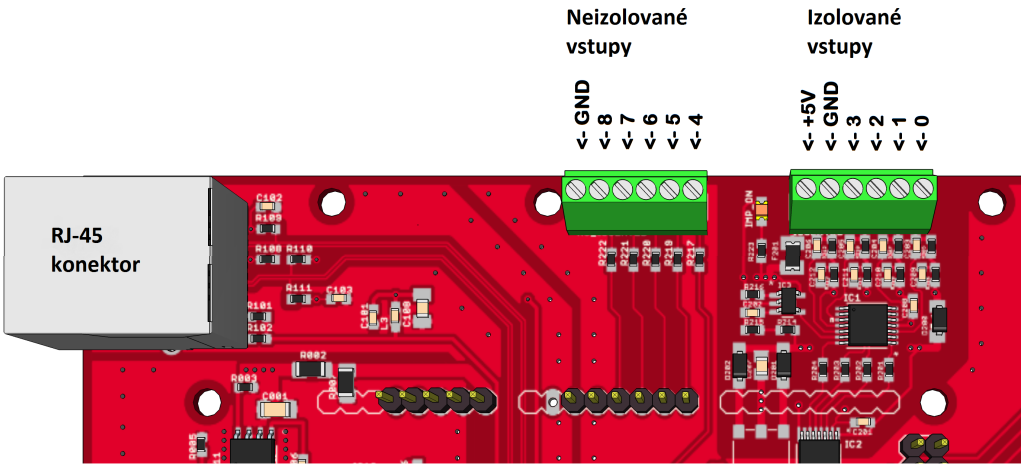

Obr. 4.3: Ukážka užívateľského rozhrania

užívateľovi podľa vybraného protokolu zobrazené základné nastavenia a pripojenie vstupov dekodéra k meranej zbernici. Vstupy sú číslované od 0 po 8, pričom prvé štyri vstupy sú izolované, viz. obrázok 4.4. Z úvodnej obrazovky má užívateľ na výber vstúpiť do nastavovacieho menu, alebo spustiť dekódovanie.

Ak užívateľ spustí dekódovanie pomocou tlačítka štart na hlavnej stránke, začne sa dekódovanie podľa nastavených parametrov. V prípade že v menu je vypnuté zobrazovanie zachytených dát na displej, užívateľovi sa zobrazia informácie o počte zachytených dát a kde možno zachytávané dáta prezerať, viz obrázok 4.3b.

Pri vstupe do menu sa užívateľovi na prvej stránke zobrazí výber protokolu a nastavenie jeho parametrov. Každý protokol má vytvorené ďalšie nastavovacie podstránky, na ktoré sa prepína pomocou tlačítka v ľavom dolnom rohu, viz. obrázok 4.3c. Pre prejdenie na ďalšiu stránku menu slúži tlačítko v pravom dolnom rohu. Na druhej stránke menu je možnosť nastavovať kde sa budú dekódované dáta ďalej posielať a zobrazovať, viz. obrázok 4.3d. Posledná stránka v menu slúži už len na

zapínanie ethernetového rozhrania, protokolu DHCP a nastavenia obnovovacieho intervalu displeja.




Obr. 4.4: Označenie vstupov pre pripojenie zbernici

## 4.5 Mikro SD karta

Implementácia mikro SD karty je postavená na knižnici „SD“, ktorá je obsiahnutá priamo s Arduino softvérom. Táto knižnica podporuje FAT16 a FAT32 súborový systém na štandardných SD a SDHC pamäťových kartách. Názvy súborov sú knižnicou obmedzené na formát 8.3, napríklad *dir/file.txt*. Nevýhodou je, že do názvu súboru sa počíta aj cesta k súboru, preto som názvy priečinkov a súborov volil čo najmenšie.

Funkcie pre obsluhu mikro SD karty sú opäť združené v jednom súbore *microSdCard.ino*. Vo funkcií *microSdInit()* sa okrem inicializácie sa v prípade novo sformátovanej karty vytvorí systém priečinkov a súborov, ktorých je obsiahnutá konfigurácia jednotlivých prvkov zariadenia. Štruktúru systému súborov s popisom položiek možno vidieť v sekcií 4.5.1. Ak mikro SD karta už obsahuje súborový systém z uložených konfiguračných súborov sa načítajú nastavené dáta. Po správnom prevedení inicializácie sa do hlavnej štruktúry programu *DeviceSetup* jednorázovo zapíše, že je karta pripojená a pripravená na použitie.

Ďalšie funkcie obsiahnuté v súbore *microSdCard.ino* s názvami začínajúcimi *read* a *update* slúžia na prečítanie a aktualizáciu konfiguračných súborov. Napríklad funkcia *readSpiConfFromSd()* prečíta konfiguráciu pre dekódovanie zbernice SPI a prečítané dáta zapíše do štruktúry ktorá je určené pre zbernicu SPI a naopak funkcia *updateSpiConfToSd()* prepíše konfiguračný súbor podľa dát v štruktúre. Funkcie

pre aktualizáciu dát sa volajú vždy po prevedení nastavenia zariadenia a správnosť aktualizácie sa potvrdí výpisom do terminálového okna.

Na konci súboru možno nájsť ešte dve funkcie, ktoré slúžia na zápis dekódovaných dát na mikro SD kartu. Jedna z funkcií zapisuje dáta v podobe ASCII znakov, tato funkcia sa používa pri sekvenčnom prístupe so zbernicou UART. Druhá funkcia podporuje formátovanie dát ako číslo v hexadecimálnom, alebo dekadickom vyjadrení, pričom maximálny počet dekódovaných dát je 16 na riadok. Táto funkcia sa používa so zbernicou SPI.

### 4.5.1 Systém súborov

```
/ ..... Koreňový adresár mikro SD karty
├── CAP: ..... Adresár pre zachytené súbory
│   ├── I2C: .....
│   ├── SPI: .....
│   ├── UART: .....
│   ├── CONF.TXT ..... Uložené číslovanie zachytávaných súborov
│   └── README.TXT ..... Info pre užívateľa
└── CONF: ..... Adresár pre konfiguračné súbory.
    ├── DEVICE.TXT ..... Konfigurácia zariadenia
    ├── ETH0.TXT ..... Konfigurácia ethernetového rozhrania
    ├── I2C.TXT ..... Nastavenie pre zbernicu I²C
    ├── SPI.TXT ..... Nastavenie pre zbernicu SPI
    ├── UART.TXT ..... Nastavenie pre zbernicu UART
    └── README.TXT ..... Info pre užívateľa
```

## 4.6 Ethernetové rozhranie

Ethernetové rozhranie je implementované pomocou integrovaného obvodu W5100 a Arduino softvér obsahuje knižnicu priamo pre obsluhu tohoto obvodu. Funkcie pre obsluhu ethernetového rozhrania sú v súbore *ethernetHandler.ino*. Podobne ako pre displej, tak pre ethernet som vytvoril dve hlavné funkcie, jedna pre inicializáciu *ethernetInit()* a druhá *ethernetLoop()* pre kontrolovanie nových spojení zostavených s dekodérom. Keď že zariadenie má možnosť ethernetové rozhranie úplne vypnúť a zapnúť kedykoľvek v priebehu používania, musel som zaistiť jednorázovú inicializáciu hardvérových periférií ethernetu priamo v hlavnej slučke programu. Táto jednorázová inicializácia je ošetrená pomocou premennej *ethStarted*, ktorá je súčasťou štruktúry *DeviceSetup* spolu s premennými *ethOn*, *useDhcp* a *client*. Premenná *client* slúži pre indikáciu pripojenia klienta k dekodéru.

Pri zapnutí dekodéru, alebo pri spustení rozhrania je užívateľ automaticky informovaný o stave konfigurácie a keď je zapnutá automatická konfigurácia pomocou

protokolu DHCP čaká sa na pridelenie konfigurácie, ktorá sa následne zobrazí na displeji a aj v terminálovom okne. Ak je konfigurácia pomocou protokolu DHCP neúspešná, alebo je vypnutá, potom sa rozhraniu pridelí konfigurácia uložená na pamäťovej karte a ak pamäťová karta nieje pripojená tak sa nastaví defaultná konfigurácia, ktorá je súčasťou zdrojového kódu programu.

Dekodér je nakonfigurovaný ako Telnet server na štandardnom porte 23. Po pripojení k Telnet serveru je užívateľ informovaný o aktuálnej konfigurácií a stavu zariadenia. Užívateľovi je umožnené na diaľku spúšťať, pozastavovať a skončiť dekódovanie trojicou príkazov, ktoré sa píšu priamo do okna terminálu. Príkazy sú START, STOP a END, nezáleží na veľkosti písma. Zachytené dáta sa do terminálového okna vypisujú iba ak je táto možnosť nastavená v menu.

Na ukážke 4.1 si je možno všimnúť podmienku pre pre-procesor kompilátora *#ifdef ETH_ON*. Pomocou tejto podmienky je možné úplne vyradiť všetky časti kódu, ktoré sa uplatňujú pre ethernetové rozhranie. Pri písaní ostatných častí kódu som túto možnosť využil a vďaka tomu boli jednotlivé iterácie skompilované a nahrané do mikroprocesora o pár sekúnd rýchlejšie.

## 4.7 Programovanie Arduino DUE

Pre nahratie programu do mikroprocesora Arduino používa ICSP In-Circuit Serial Programer. Tato funkcionalita je umožnená pomocou ďalšieho mikroprocesora, konkrétne ATmega16, ktorý emuluje prevodník USB na UART, obsluhuje napájanie procesora a spúšťa aj bootloader, ktorý je uložený v ATSAM3X8E mikroprocesory. Bootloader sa spustí tak, že mikroprocesor ATSAM3X8E sa vymaže. Po resetnutí, keď mikroprocesor neobsahuje žiaden spustiteľný kód, sa automaticky spustí bootloader. Vymazanie je možno vykonať pomocou tlačítka ERASE, ktoré je osadené na vývojovej doske, alebo otvorenie sériovej linky na programovacom porte s rýchlostnou 1200 Baud/s a emulovaný prevodník zaistí vymazanie pomocou tranzistora, ktorý vykonáva funkciu tlačítka ERASE. Podrobnejší popis implementácie bootloadra možno nájsť v článku [2]

Keď že som upravoval aj zdrojové knižnice priamo od Arduina, nemusí sa vždy podariť tieto zmeny replikovať[5], vždy je však možné úpravy vykonať na základe výpisu pri preklade zdrojového kódu. Preto som vytvoril jednoduchý Batch skript *upload.bat*, ktorý po spustení vyzve užívateľa k pripojeniu vývojovej dosky na programovací port a zadania názvu tohoto portu. Následne sa vývojová doska vymaže a spustí sa nahrávanie binárneho súboru pomocou bootloadru. Program pre dekodér je celkom rozsiahly, bootloader komunikuje len s rýchlostnou 9600 Baud/s a preto

[5]Užívateľ nemá administrátorské práva na používanom PC

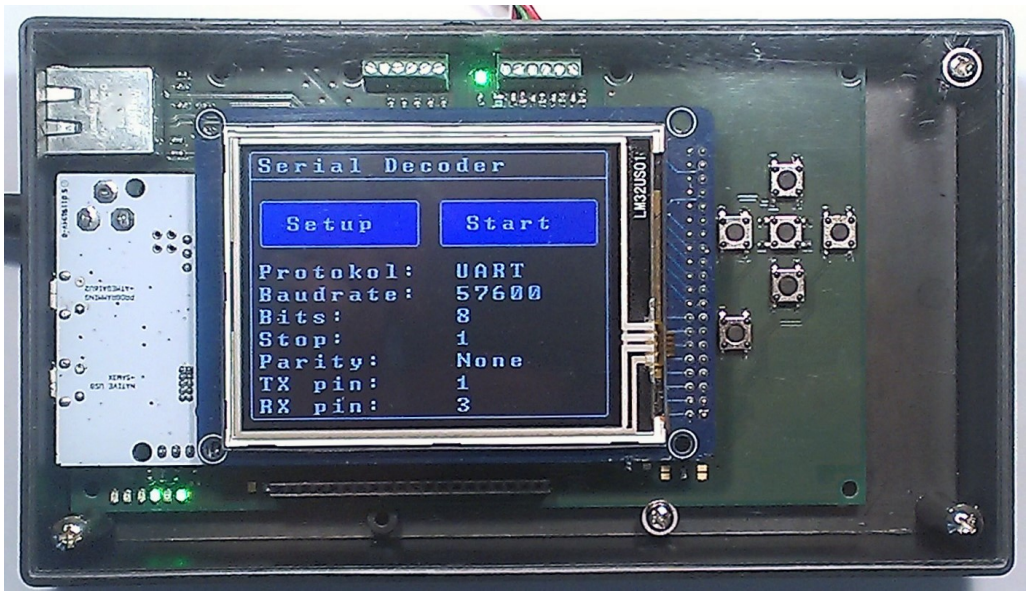
nahranie trvá približne jednu minútu.

Preložený binárny program spolu zo skriptom pre nahranie programu možno nájsť na priloženom CD v archíve *DecoderBasicBin.zip*.

Ďalšie informácie a možnosti ako implementovať ICSP sú popísané v aplikačnej poznámke [3] priamo od firmy Atmel.

# 5 TECHNICKÉ PARAMETRE DEKODÉRA

Dekodér používa 32bitový ARM Cortex-M3 mikroprocesor taktovaný na 104 MHz. Pre načítanie dát sa používa dynamicky alokovaný kruhový buffer do veľkosti 64 kB. Na neizolovaných vstupoch dekodér pracuje s 3,3 V logikou, izolované vstupy pracujú s 3,3 V a 5 V logikou. Pre zbernicu SPI je rýchlosť spracovania dát do 7 Mb/s, pre $I^2C$ do 1 Mb/s a UART do 195 kb/s. Maximálna podporovaná veľkosť Micro SD karty s FAT32 je 32 GB. Ethernetové rozhranie na fyzickej vrstve podporuje 10BaseT/100BaseTX. Rozhranie podporuje len IPv4 protokol. K dekodéru je možné sa pripojiť pomocou sériovej linky emulovanej na programovacom porte vývojovej dosky Arduino s rýchlostnou 115200Baud/s, 8 bitov dáta, 1 stop bit a žiadna parita. Pre zobrazenie je použitý 3,2" farebný displej s dotykovou plochou. Dekodér je napájaný 12 V/1 A adaptérom.




Obr. 5.1: Hotový výrobok

# 6 ZÁVER

Cieľom bolo vytvoriť univerzálne zariadenie, ktoré bude schopné spracovávať zachytené dáta zo sériových liniek. Práca sa venuje návrhu hardvéru a softvérového riešenia pre dekodér sériových zberníc. Hardvér je navrhnutý modulárne, príkladom je možnosť použitia viacerých typov displejov čo bolo časovo náročné, keď že som musel pre každý displej overiť pripojenie, napäťové úrovne a maximálne prúdy, s ktorými pracujú. Väčšie displeje už vyžadujú napájanie v veľkým prúdom, preto bol použitý kvalitnejší DC-DC menič. Výhoda navrhnutého zariadenia vďaka implementovaným modulom (Ethernet, SD karta, TFT a LCD displejom) je, že zariadenie možno použiť aj na iné účely, napríklad jednoduchý analyzátor sieťovej komunikácie.

Výhodou dekodéra je, že má kompletne izolované vstupy z vlastným napájaním. V pôvodnom návrhu sa prehrieva budiaci obvod pre transformátor, pretože vybraný transformátor má malú indukčnosť. Izolované vstupy by mali byť dostatočne rýchle pretože použitý digitálny izolátor má šírku pásma 150 MHz. Reálna rýchlosť čítania dát je však limitovaná rýchlosťou vyčítania logických úrovní zo vstupných portov mikroprocesora.

Pri návrhu layout-u DPS som musel urobiť niekoľko kompromisov, aby výsledné zariadenie bolo čo najnižšie. Pre DC-DC menič som musel vybrať cievku s nízkym profilom a na DPS umiestniť tak, aby nezavadzala vyčnievajúcim komponentom dosky Arduina, ktorá sa pripája priamo na dosku. Rovnako tiež transformátor použitý pre napájanie výstupnej časti som musel umiestniť tak, aby nezavadzal DC-DC meniču na Arduino vývojovej doske. Odhad ceny zariadenia vychádza približne na 60 €, pričom je do tejto sumy zahrnutá navrhnutá doska zo súčiastkami, vývojová doska Arduino DUE a použitý displej.

Softvér pre dekodér som opäť navrhol modulárne čo zaisťuje jednoduchšiu rozšíriteľnosť zariadenia o nové funkcie. Viac ako polovička zdrojového kódu ošetruje užívateľské rozhranie, ktoré som navrhol tak, aby bolo pre užívateľa intuitívne a ľahko použiteľné. Pri vývoji užívateľského rozhrania som narazil na problém pomalého vykresľovania textu na displeji. Tento problém som sa snažil vyriešiť niekoľkými mechanizmami, ktoré som v práci popísal. Nakoniec som ale upravil použitú knižnicu a tým sa problém pomalého vykresľovania textu vyriešil. Ďalšou zaujímavou časťou softvéru je implementácia dynamicky alokovanej kruhovej vyrovnávacej pamäte pri spracovávaní dát, ktorej veľkosť môže zvoliť užívateľ. Vďaka tejto implementácií je spracovávanie dát spoľahlivejšie. Podobný mechanizmus ako pri ošetrení vyrovnávacej pamäte som použil aj pri vykresľovaní formátovaných dát na displej pri zbernici UART, kde sa výstup na displej opäť indexuje do kruhu. Dekodér je pripravený aj pre prácu po sieti, pre konfiguráciu IP adresy som použil protokol DHCP, užívateľ môže však IP adresu nakonfigurovať ručne. Dekodér sa v sieti správa ako Telnet

server a ponúka textový výstup, ktorý je možné ďalej spracovávať. Taktiež je možnosť dekodér na diaľku ovládať jednoduchými príkazmi, ktorými je možné spúšťať, pozastavovať a ukončovať dekódovanie. Príkazy by bolo možné v budúcnosti rozšíriť aby bolo možné zariadenie na diaľku aj nastaviť.

Hotový výrobok má uplatnenie pri práci so zbernicami a verifikovaní posielania správnych dát na dekódovaných zberniciach. Vďaka modulárnemu návrhu je dekodér možno rozšíriť o ďalšie sériové zbernice, napríklad $I^2S$, 1-Wire, MIDI, atd.

# LITERATÚRA


[1] Analog Devices, Inc. *Katalógový list.* [online]. [cit. 29. 11. 2015]. Dostupné z URL: <http://bit.ly/27Oqk36>.

[2] Arduino Due Bootloader Explained. *Arduino* [online]. [cit. 01. 05. 2016]. Dostupné z URL: <http://bit.ly/1TEVTWR>.

[3] Atmel AT02333: Safe and Secure Bootloader Implementation for SAM3/4 *Aplikačná poznámka.* [online]. [cit. 7. 12. 2015]. Dostupné z URL: <http://goo.gl/oK1ii0>.

[4] Atmel ATmega2560. *Katalógový list.* [online]. [cit. 7. 12. 2015]. Dostupné z URL: <http://bit.ly/1U6Jjvq>.

[5] Atmel. ATSAM3X8E. *Katalogový list.* [online]. [cit. 6. 12. 2015]. Dostupné z URL: <http://bit.ly/1U6Jjvq>.

[6] I2C Bus: *I2C – What's That?* [online]. [cit. 30. 11. 2015]. Dostupné z URL: <http://www.i2c-bus.org/>.

[7] Keysight Technologies: *MSO6014A Mixed Signal Oscilloscope.* [online]. [cit. 5. 12. 2015]. Dostupné z URL: <http://bit.ly/1TCHP0a>.

[8] Liquid Crystal Display Module, ATM12864D. *Katalógový list.* [online]. [cit. 29. 11. 2015]. Dostupné z URL: <http://www.gme.cz/img/cache/doc/513/119/mg12864a-sbc-h-datasheet-1.pdf>.

[9] Microchip. *Katalógový list.* [online]. [cit. 29. 11. 2015]. Dostupné z URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/22057B.pdf>.

[10] Microchip. *Katalógový list.* [online]. [cit. 29. 11. 2015]. Dostupné z URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/22139C.pdf>.

[11] PEREA, Francis. *Arduino Essentials. Olton Birmingham: Packt Publishing, 2015. ISBN 9781784398569.*

[12] PURDUM, Jack. *Beginning C for Arduino. Berkeley, CA: Apress, 2015. DOI: 10.1007/978-1-4842-0940-0. ISBN 1484209419.*

[13] Silicon Labs. *Katalógový list.* [online]. [cit. 29. 11. 2015]. Dostupné z URL: <http://bit.ly/25gAVoA>.

[14] Tektronix Logic Analyzers. *TLA6400* [online]. [cit. 5. 12. 2015]. Dostupné z URL: <http://bit.ly/1Vf4ZL2>.

[15] Texas Instruments. *Katalógový list.* [online]. [cit. 29. 11. 2015]. Dostupné z URL: <http://www.ti.com/lit/ds/symlink/sn74lvch16t245.pdf>.

[16] Texas Instruments. *Katalógový list.* [online]. [cit. 29. 11. 2015]. Dostupné z URL: <http://www.ti.com/lit/ds/sllsea0g/sllsea0g.pdf>.

[17] Texas Instruments. *RS-422 and RS-485 Standards Overview and System Configurations* [online]. [cit. 4. 12. 2015]. Dostupné z URL: <http://www.ti.com/lit/an/slla070d/slla070d.pdf>.

[18] Rinky-Dink Electronics. *UTFT library for Arduino* [online]. [cit. 28. 4. 2016]. Dostupné z URL: <http://www.rinkydinkelectronics.com/library.php?id=51>.

[19] Wikipedia. *Serial Peripheral Interface Bus.* [online]. [cit. 4. 12. 2015]. Dostupné z URL: <https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus>.

[20] WIZnet. *Katalógový list.* [online]. [cit. 29. 11. 2015]. Dostupné z URL: <http://bit.ly/1NFzurX>.

[21] Zeroplus LAP-C. *Špecifikácie produktu* [online]. [cit. 29. 11. 2015]. Dostupné z URL: <http://bit.ly/1U6Ltv0>.

# ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

| | |
|---|---|
| DPS | Doska plošných spojov |
| ESR | Equivalent series resistance |
| FAT | File Allocation Table |
| $I^2C$ | Inter-Integrated Circuit |
| $I^2S$ | Integrated Interchip Sound |
| MSO | Mixed signal osciloscop |
| PLL | Phase-locked loop – slučka fázového závesu |
| SD | Secure Digital – pamäťová karta |
| SDHC | Secure Digital High Capacity |
| SPI | Serial Peripheral Interface – sériové periférne rozhranie |
| UART | Universal Asynchronous Receiver Transmitter |

# ZOZNAM PRÍLOH

# A PRIPOJENIE VÝVOJOVEJ DOSKY

Tab. A.1: TFT modul – displej

| Arduino DUE číslo pinu | TFT Displej názov pinu |
|---|---|
| 25 | LCD_RS |
| 26 | LCD_WR |
| 27 | LCD_CD |
| 28 | LCD_RST |
| 33–39 | DB_0–7 |
| 44–51 | DB_15–8 |
| 3 | LED_PWM |

Tab. A.2: TFT modul – dotyková plocha

| Arduino DUE číslo pinu | Dotiková plocha názov pinu |
|---|---|
| 29 | TP_BUSY |
| 31 | TP_IRQ |
| 32 | TP_DIN |
| 41 | TP_CLK |
| 42 | TP_DOUT |
| 43 | TP_CS |

Tab. A.3: TFT modul – SD karta, FLASH pamäť

| Arduino DUE číslo pinu | Názov pinu |
|---|---|
| 74 (MISO) | SD_DO |
| 75 (MOSI) | SD_DIN |
| 76 (SCLK) | SD_CLK |
| 52 | F_CS |
| 53 | SD_CS |

Tab. A.4: Ethernetové rozhranie

| Arduino DUE číslo pinu | Názov pinu |
|---|---|
| 74 (MISO) | MISO |
| 75 (MOSI) | MOSI |
| 76 (SCLK) | SCLK |
| 10 | ETH_CS |
| 2 | ETH_INT |
| 11 | ETH_SHDN |

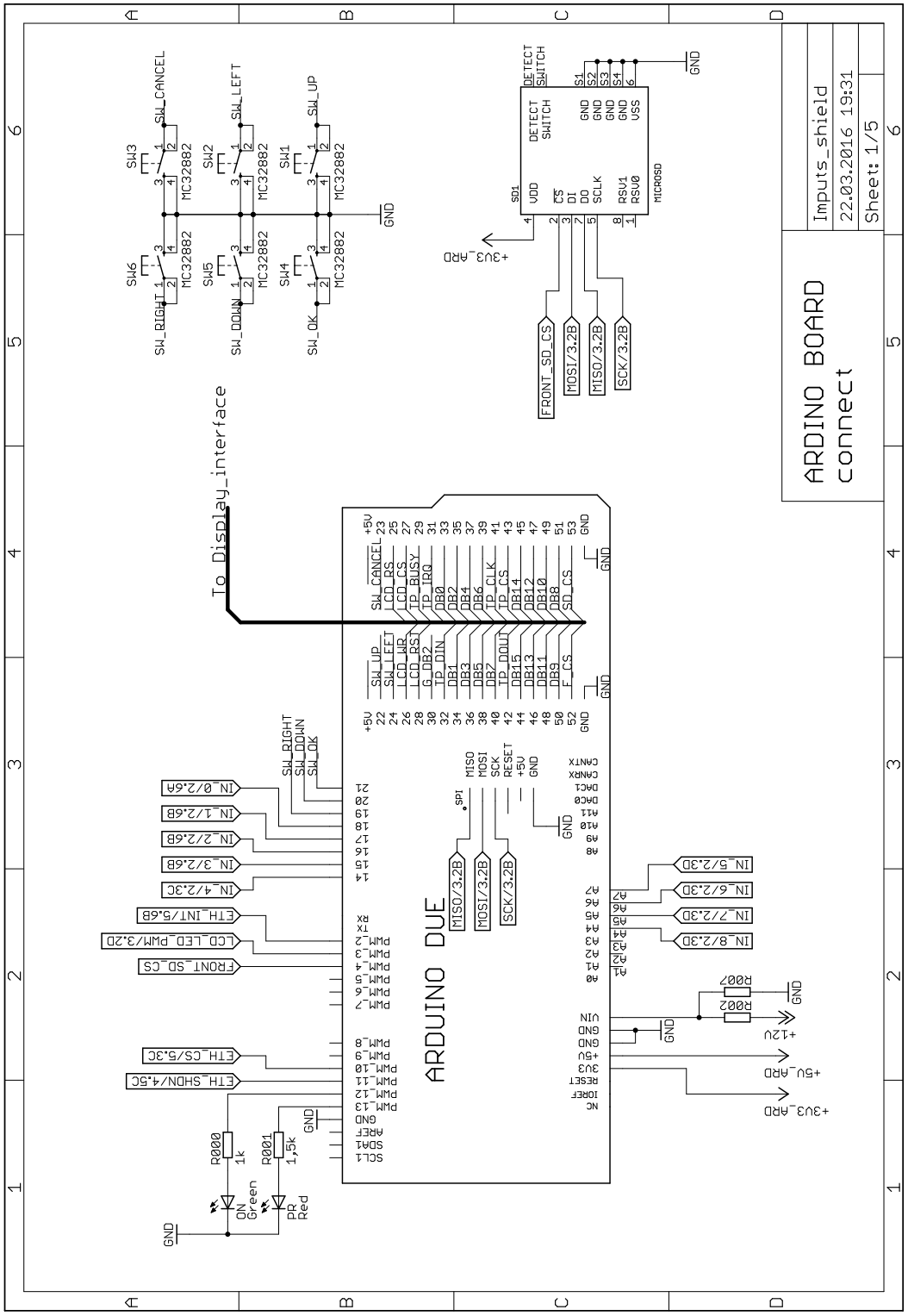
Tab. A.5: Tlačítka

| Arduino DUE číslo pinu | Názov pinu |
|---|---|
| 19 | SW_RIGHT |
| 20 | SW_DOWN |
| 21 | SW_OK |
| 22 | SW_UP |
| 23 | SW_CANCEL |
| 24 | SW_LEFT |

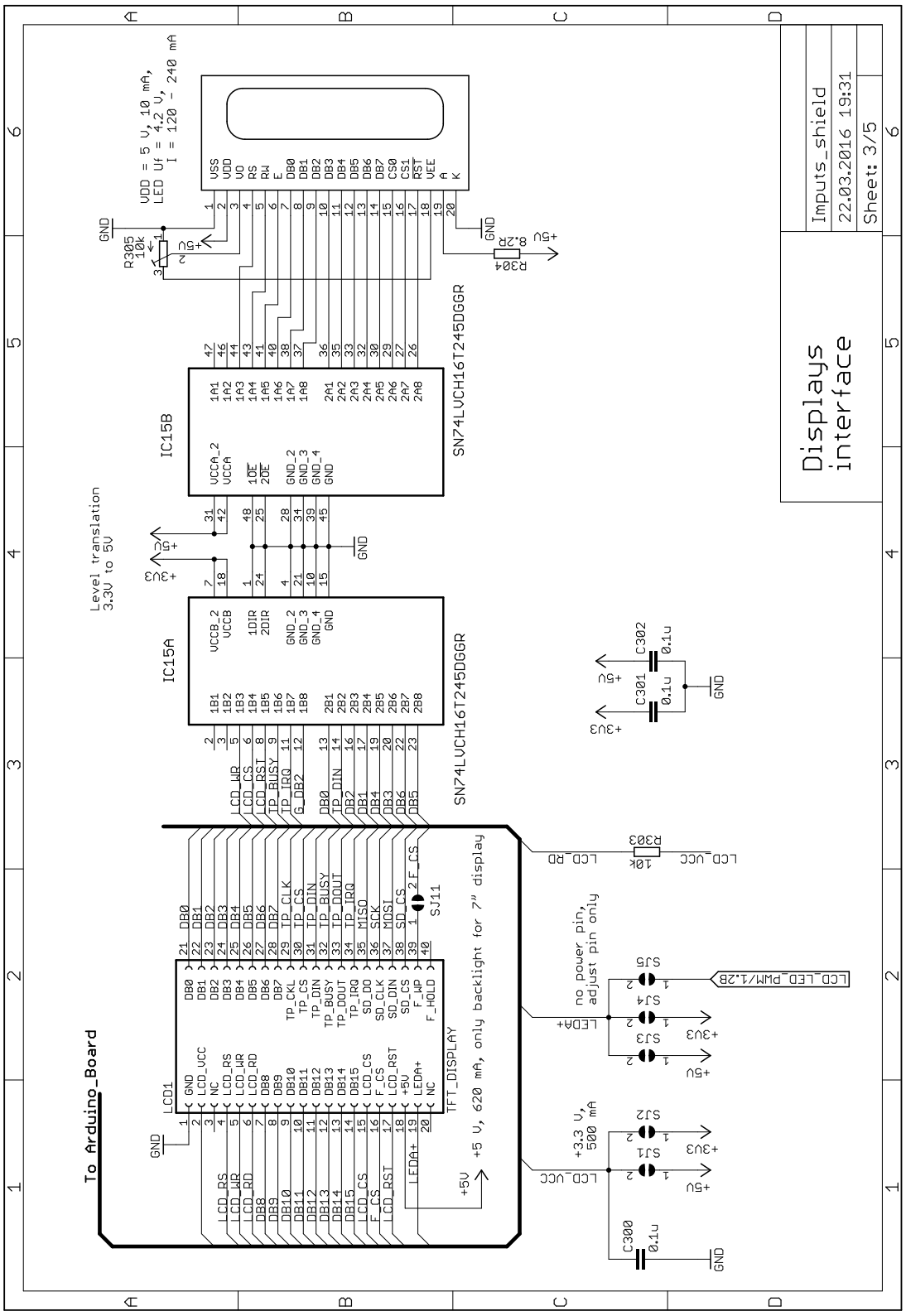
Tab. A.6: Mikro SD karta

| Arduino DUE číslo pinu | SD karta |
|---|---|
| 74 (MISO) | SD_DO |
| 75 (MOSI) | SD_DIN |
| 76 (SCLK) | SD_CLK |
| 4 | SD_CS |

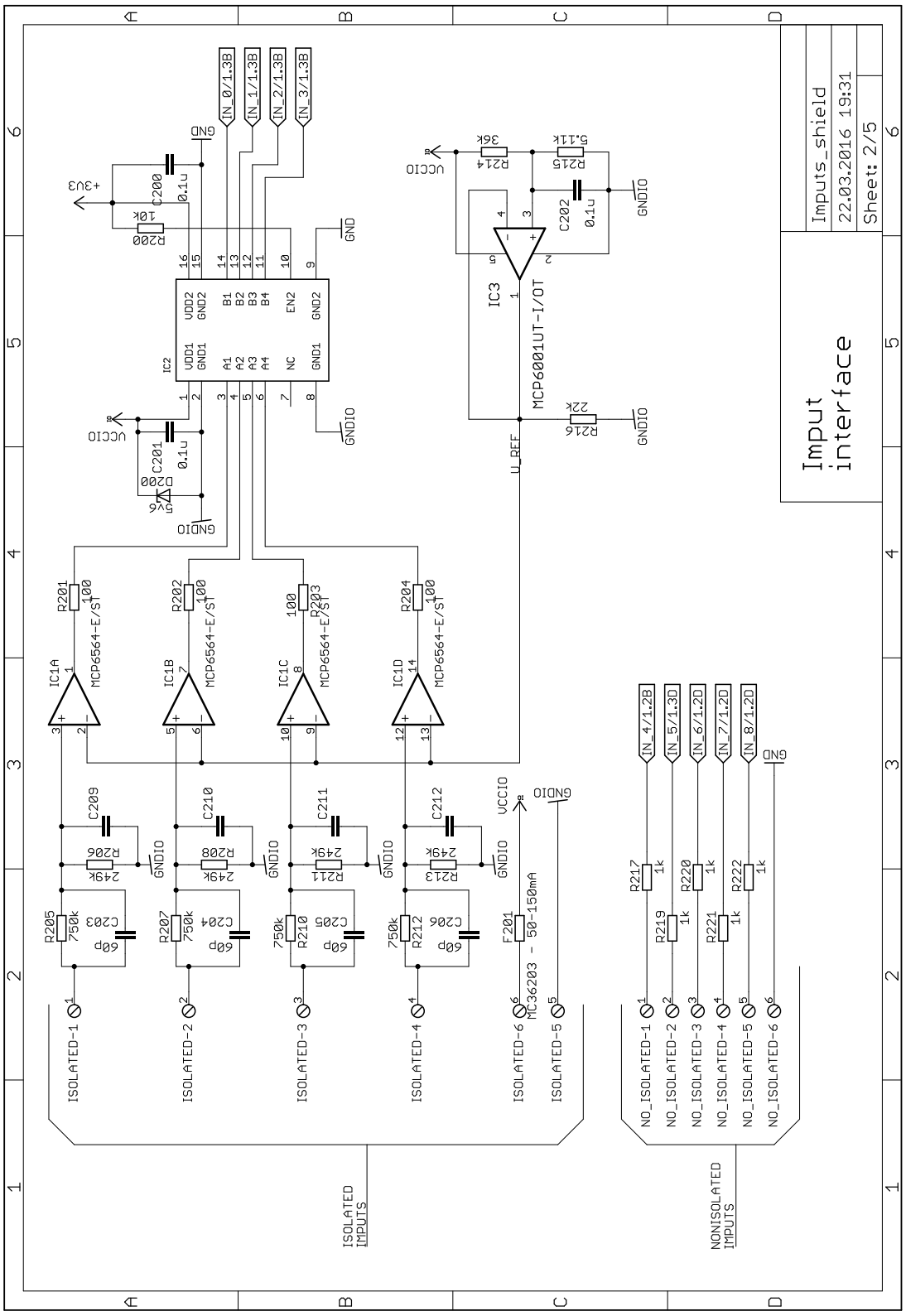
# B SCHÉMY ZAPOJENIA



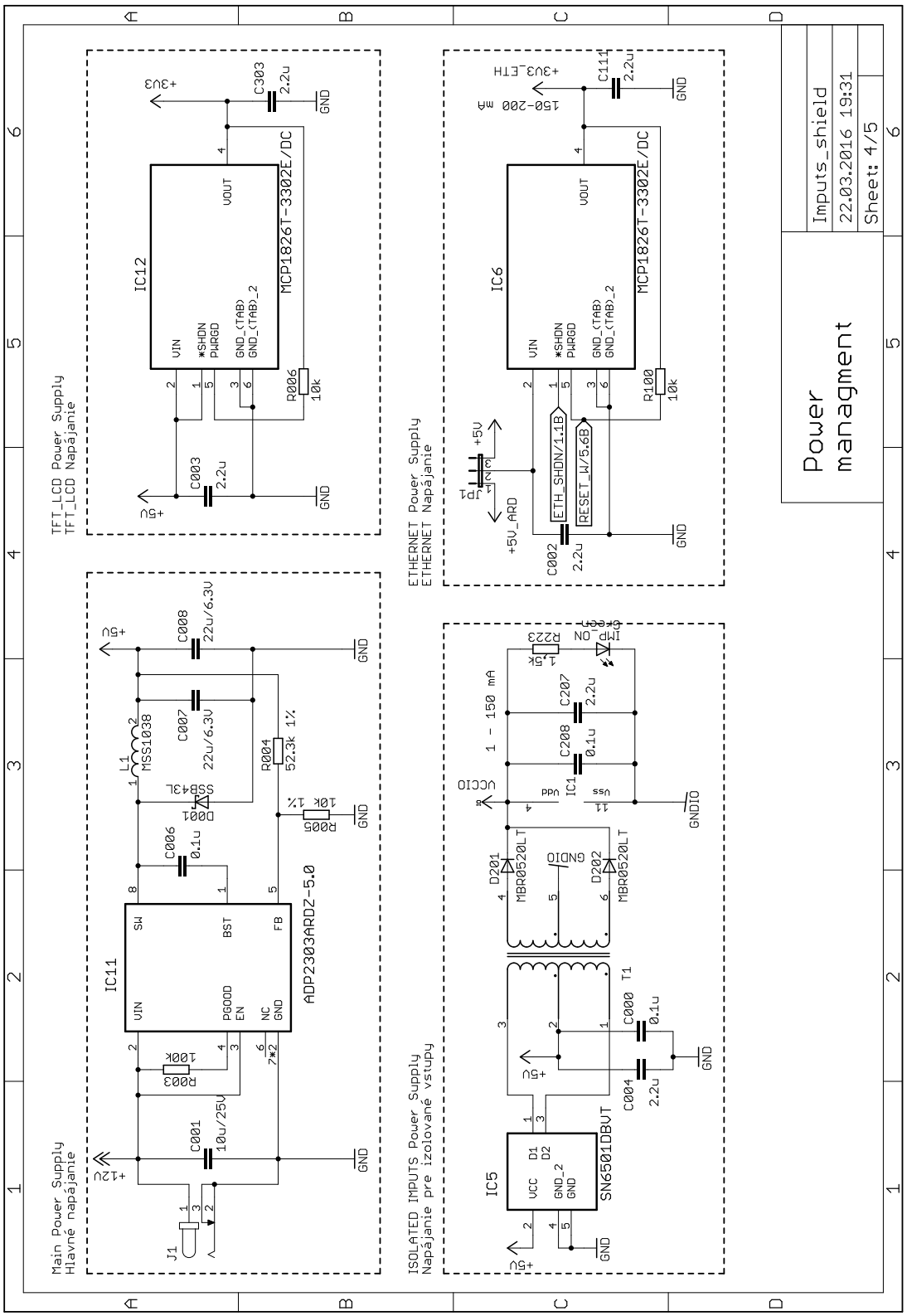

Obr. B.1: Schéma zapojenia Arduino DUE, tlačidiel a slotu pre Mikro SD karty.



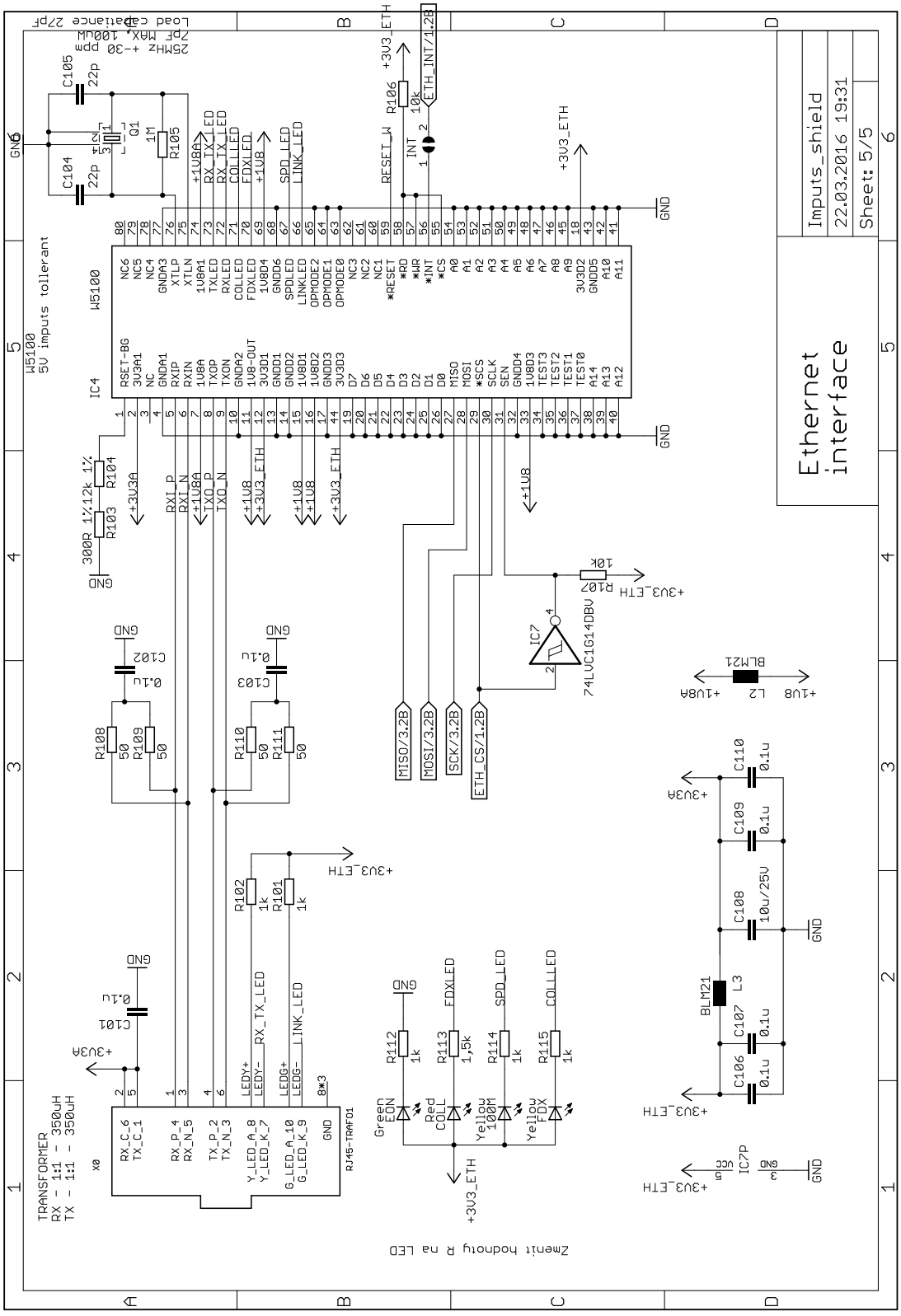

Obr. B.2: Schéma zapojenia displejov.




Obr. B.3: Schéma zapojenia izolovaných a neizolovaných vstupov.

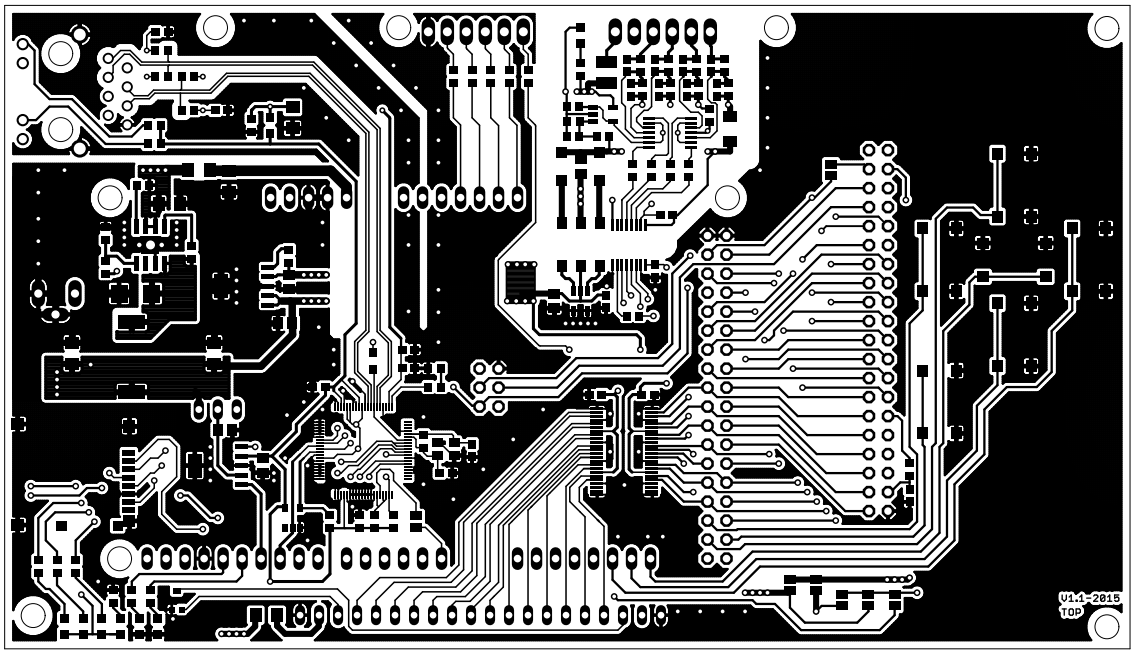



Obr. B.4: Schéma zapojenia napájacích častí obodu.

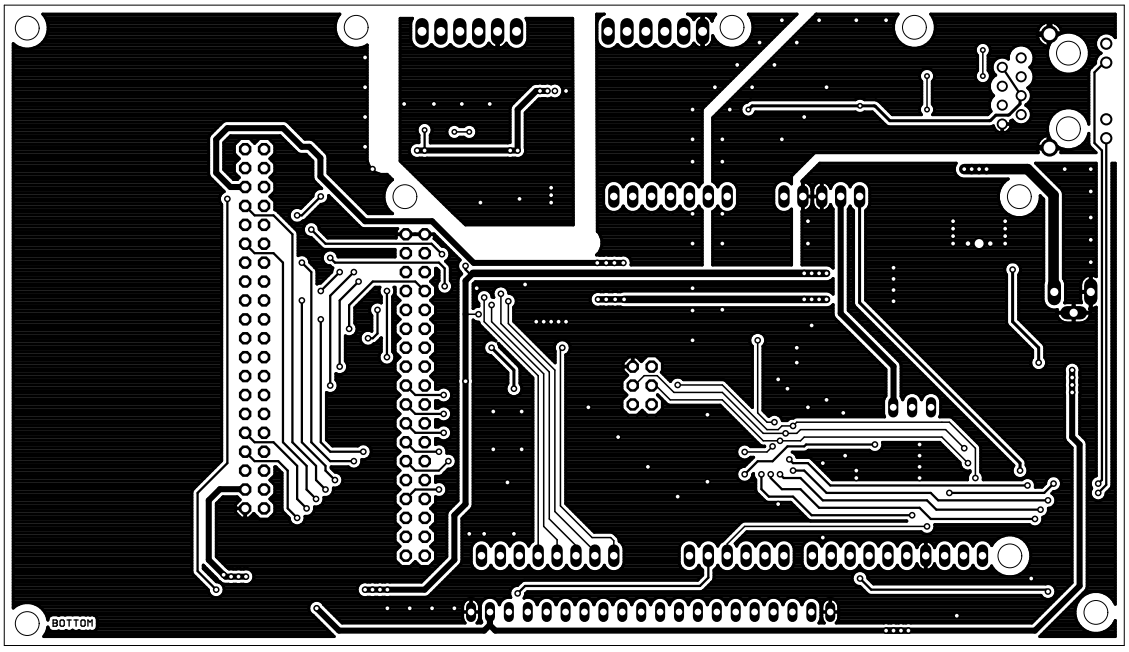



Obr. B.5: Schéma zapojenia Ethernetu s integrovaním obvodom W5100.

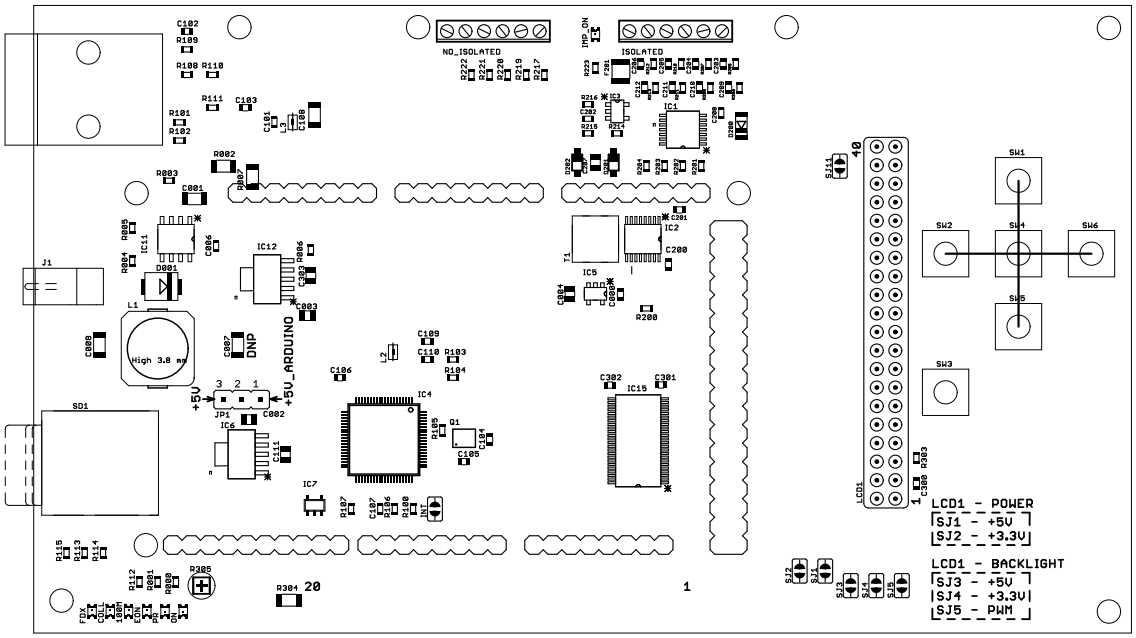
# C NÁVRH PLOŠNÉHO SPOJA




Obr. C.1: Doska plošného spoju, vrstva top.




Obr. C.2: Doska plošného spoju, vrstva bottom.




Obr. C.3: Doska plošného spoju, osadzovací výkres

# D ZOZNAM SÚČIASTOK

Tab. D.1: Zoznam pasívnych súčiastok

| Počet | Hodnota | Súčiastka | Púzdro | Označenie |
|---|---|---|---|---|
| 1 | 8,2R | R-EU_R1206 | R1206 | R304 |
| 4 | 50R | R-EU_R0603 | R0603 | R108, R109, R110, R111 |
| 4 | 100R | R-EU_R0603 | R0603 | R201, R202, R203, R204 |
| 1 | 300R 1% | R-EU_R0603 | R0603 | R103 |
| 11 | 1k | R-EU_R0603 | R0603 | R000, R101, R102, R112, R114, R115, R217, R219, R220, R221, R222 |
| 3 | 1,5k | R-EU_R0603 | R0603 | R001, R113, R223 |
| 1 | 5,11k | R-EU_R0603 | R0603 | R215 |
| 7 | 10k | R-EU_R0603 | R0603 | R005, R006, R100, R106, R107, R200, R303 |
| 1 | 12k 1% | R-EU_R0603 | R0603 | R104 |
| 1 | 22k | R-EU_R0603 | R0603 | R216 |
| 1 | 36k | R-EU_R0603 | R0603 | R214 |
| 1 | 52,3k 1% | R-EU_R0603 | R0603 | R004 |
| 1 | 100k | R-EU_R0603 | R0603 | R003 |
| 4 | 249k | R-EU_R0603 | R0603 | R206, R208, R211, R213 |
| 4 | 750k | R-EU_R0603 | R0603 | R205, R207, R210, R212 |
| 1 | 1M | R-EU_R0603 | R0603 | R105 |
| 2 |  | R-EU_R1206 | R1206 | R002, R007 |
| 1 | 10k | R-TRIMMPVZ2A | PVZ2A | R305 |
| 2 | 22p | C-EUC0603K | C0603K | C104, C105 |
| 4 | 60p | C-EUC0603K | C0603K | C203, C204, C205, C206 |
| 16 | 0,1u | C-EUC0603K | C0603K | C000, C006, C101, C102, C103, C106, C107, C109, C110, C200, C201 C202, C208, C300, C301, C302 |
| 6 | 2,2u | C-EUC0805K | C0805K | C002, C003, C004, C111, C207, C303 |
| 2 | 10u / 25V | C-EUC1206K | C1206K | C001, C108 |
| 2 | 22u / 6,3 V | C-EUC1206K | C1206K | C007, C008 |

Tab. D.2: Zoznam diód

| Počet | Hodnota | Súčiastka | Púzdro | Označenie |
|---|---|---|---|---|
| 2 | Yellow | LEDCHIPLED_0805 | CHIPLED_0805 | 100M, FDX |
| 2 | Red | LEDCHIPLED_0805 | CHIPLED_0805 | COLL, PR |
| 3 | Green | LEDCHIPLED_0805 | CHIPLED_0805 | EON, IMP_ON, ON |
| 1 | 5v6 | ZENER-DIODESOD80C | SOD80C | D200 |
| 1 | SSB43L | SSB43L | SMB | D001 |
| 2 | MBR0520LT | MBR0520LT | SOD123 | D201, D202 |

Tab. D.3: Zoznam integrovaných obvodov

| Počet | Hodnota | Súčiastka | Púzdro | Označenie |
|---|---|---|---|---|
| 1 | 74LVC1G14DBV | 74LVC1G14DBV | SOT-23 | IC7 |
| 1 | ADP2303ARDZ-5.0 | ADP2303ARDZ-5.0 | SOIC-8 | IC11 |
| 1 | SI8645BA-B-IU | SI8645BA-B-IU | QSOP | IC2 |
| 1 | SN6501DBVT | SN6501DBVT | SOT23-5 | IC5 |
| 1 | SN74LVCH16T245 | SN74LVCH16T245 | TSSOP-48 | IC15 |
| 2 | MCP1826T-3302E/DC | MCP1826T-3302E/DC | SOT223-5 | IC6, IC12 |
| 1 | MCP6001UT-I/OT | MCP6001UT-I/OT | SOT23-5 | IC3 |
| 1 | MCP6564-E/ST | MCP6564-E/ST | TSSOP | IC1 |
| 1 | W5100 | W5100 | SQFP-80 | IC4 |

Tab. D.4: Zoznam ostatných súčiastok

| Počet | Hodnota | Súčiastka | Púzdro | Označenie |
|---|---|---|---|---|
| 1 | TXC7M | TXC7M | MICROSD | Q1 |
| 1 | MSS1038 | MSS1038 | MSS1038 | L1 |
| 2 | BLM21 | WE-CBF_0805 | 805 | L2, L3 |
| 1 | TRPWB-1.5-CL | TRPWB-1.5-CL | COILCRAFT | T1 |
| 1 | MC36203 | R-EU_R1210 | R1210 | F201 |
| 6 | MC32882 | TACTILE-SWITCH | MC32882 | SW1–6 |
| 1 | MICROSD | MICROSD | MICROSD | SD1 |
| 1 | RJ45-TRAFO1 | RJ45-TRAFO1 | RJ45-TRAFO | X0 |
| 2 |  | MPT6 | 6POL254 | ISOLATED, NO_ISOLATED |

# E OBSAH PRILOŽENÉHO CD

/ ..... koreňový adresár priloženého CD
- Hardvér:
  - BOM.xlsx
  - Imputs_shield.brd ..... Eagle 7.4.0
  - Imputs_shield.sch ..... Eagle 7.4.0
- Katalogove listy:
  - Displays: ..... dokumentácie k TFT displejom
  - ADP2303 - step down regulator.pdf
  - Atmel-SAM3X-SAM3A_Datasheet.pdf
  - MCP1826 - LDO.pdf
  - MCP6001 - OPAMP.pdf
  - MCP656x - comparator.pdf
  - MSS1038 - inductor.pdf
  - power plug.pdf
  - PTC.pdf
  - RJ-45 connector.pdf
  - SD mmc Connector.pdf
  - Si86xx - digital_Isolator.pdf
  - sn6501 - Transformer_Driver.pdf
  - SN74LVCH16T245 - 16-BIT BUS TRANSCEIVER.pdf
  - W5100_Datasheet_v1.2.6.pdf
- Knižnice pre Arduino: ..... Použité knižnice pre Arduino
  - UTFT.zip
  - UTFTmod.zip ..... Modifikovaná knižnica pre TFT displej
  - UTouch.zip
- Obrázky: ..... Obrázky zhotoveného výrobku
- Softvér:
  - DecoderBasic.zip ..... Zdrojový kód programu (Arduino 1.6.9)
  - DecoderBasicBin.zip ... Preložený program a skript pre rýchle nahranie
- Dekoder seriovych zbernic.pdf ..... Táto práca v elektronickej podobe