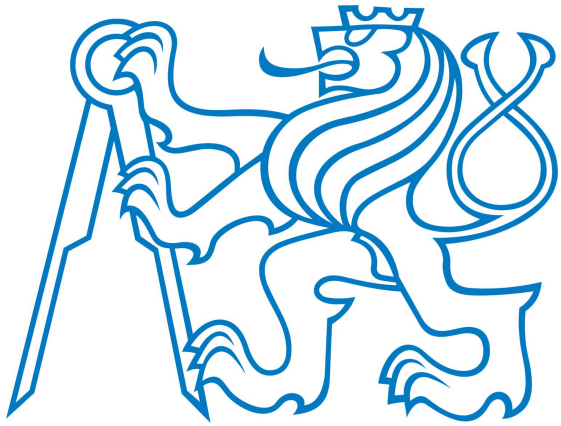
České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky



Bakalářská práce

Měřicí přístroj kruhovitosti součástek

Jiří Brázdil

Ing. Ondřej Tereň
vedoucí práce

Bakalářský program Kybernetika a robotika

Obor Robotika

Praha, Květen 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jiří B r á z d i l

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Měřicí přístroj kruhovitosti součástek

Pokyny pro vypracování:

Navrhněte a zrealizujte měřicí přístroj pro kontrolu kruhovitosti součástek. Měřicí přístroj bude využívat polohový senzor, jehož data budou zpracována pomocí řídicí desky s procesorem Kinetis MK20DN512, 4,5" displejem, dvěma drivery pro krokové motory a microSDHC kartou.

Cílem bakalářské práce je tvorba driveru pro polohový senzor, pokročilého driveru pro 4,5" displej, vytvoření rozhraní pro měření a zobrazení výsledků kruhovitosti součástek na displeji, propojení měřicího systému s PC pomocí sběrnice USB, realizace uživatelského rozhraní v PC, které bude poskytovat zobrazení naměřených dat.

Navržený měřicí systém rovněž experimentálně ověřte a zhodnoťte dosažené výsledky.

Seznam odborné literatury:

[1] K20 SubFamily Reference Manual, Doc. No.: K20P100M100SF2V2RM, Freescale, Jun 2012.
[2] Universal Serial Bus Specification, Revision 2.0, April 2000.

Vedoucí bakalářské práce: Ing. Ondřej Tereň

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 15. 2. 2016

Poděkování

Chtěl bych poděkovat panu doc. Ing. Jan Holubovi, Ph.D za to, že mi umožnil vyhotovit práci pod vedením katedry měření. Dále bych chtěl poděkovat svému vedoucímu, Ing. Ondřeji Tereňovi za příjemnou spolupráci a panu Františku Řezníčkovi za skvělé téma pro mou bakalářskou práci a za to, že jsem mohl tvořit skutečně praktický přístroj. Nakonec svému otci, Ing. Jaroslavu Brázdilovi za zázemí jak rodinné, tak technické, které mi umožnilo práci realizovat.

Prohlášení

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

V Praze dne 26.5.2016.

……...

Jiří Brázdil

Prázdná strana

Abstrakt

Tato práce dokumentuje tvorbu kontrolního zařízení pro přístroj sloužící k měření kruhovitosti součástek. U většiny komponentů použitých při výrobě zařízení se v rámci této práce jedná pouze o jejich úpravu. Práce pojednává o změně výchozí základní desky ze strany hardware i firmware, připojení periferií, tvorby rozhraní pro dotykový display a počítačového programu pro analýzu naměřených dat.

Klíčová slova: kruhovitost; měření; polohová; sonda; ARM; procesor; magnetický; senzor; dotykový; display; uživatelské; rozhraní

Abstract

This thesis documents the design process of a circularity measurement device. Subject of this work is mostly hardware or software adjustment of existing parts. In particular: hardware and firmware changes to an existing mainboard, connection of peripheral devices, user interface creation for a touch controlled display and a pc software for further data analysis.

Keywords: circularity; measurement; distance; ARM; processor; magnetic; sensor; touch; display; user; interface

Obsah

Kapitola 1
Úvod

V mnoha případech je třeba změřit zakřivení určitého objektu. Existuje větší množství způsobů, jak kruhovitost objektu měřit, včetně ultrazvuku, či optických senzorů. Tato práce se zaměřuje na použití mechanické metody měření tyčovým polohovým senzorem. Jelikož výsledné zařízení bude použito v praxi, byla zvolena technologie, která poskytuje dostatečné výsledky za co nejmenší cenu.

Úvodem práce je seznámení s metodami měření kruhovitosti a jejím výpočtem. Hlavním obsahem práce je tvorba přístroje pro měření kruhovitosti součástek, dokumentace úprav výchozí základní desky pro účely této práce a přizpůsobení jejího firmware. Práce také pokrývá rozbor komunikace polohového senzoru a jeho připojení k základní desce a tvorbu pc softwaru pro analýzu naměřených dat.

Měřené objekty jsou umístěny na ose připojené ke krokovému motoru. Senzor sám je na posuvné ose ovládán ručně. Během měření se senzor manuálně přisune k měřenému objektu, přiloží a vynuluje. Krokový motor provede po nájezdu do reference otáčku o 360° a senzor sejme v n bodech odchylku povrchu měřeného objektu od původní polohy. Tato data slouží k výpočtu kruhovitosti měřeného objektu. Výsledek je zobrazen na přiložené dotykové obrazovce a lze jej také analyzovat v libovolném počítači pomocí dalšího softwaru.

Cíle práce

1. Analýza výstupu polohového senzoru
2. Úprava základní desky
3. Úprava firmware procesoru
4. Připojení magnetického senzoru pro nájezd do reference
5. Tvorba firmware pro samotné měření
6. Zobrazení výsledků měření na dotykové obrazovce
7. Tvorba software pro podrobnější analýzu naměřených dat

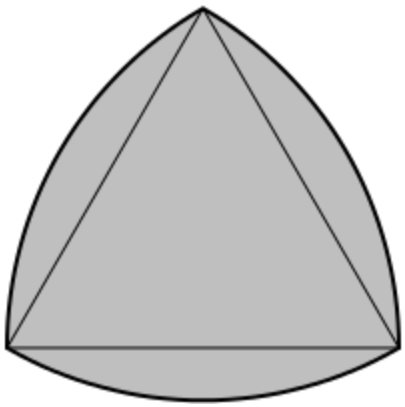
Fotka výsledného přístroje připojeného k šasi se nachází v příloze 12. Výsledný přístroj tedy dovede po přistavení sondy změřit objekt a zobrazit hodnotu průměrné odchylky od kružnice. Zároveň data vypíše do souboru, takže se dají načíst počítačovým programem po rozhraní USB, které je použito jako Mass Storage Device. V softwaru pro pc lze zobrazit konkrétní hodnoty všech měření a kromě obrysu objektu také graf jednotlivých odchylek pro všechna provedená měření, včetně hodnoty kruhovitosti a extrémů v naměřených datech.

Kapitola 2

Teorie

Kruhovitost určuje do jaké míry se tvar objektu blíží kruhu. Určujícím prvkem je celý obrys objektu, nikoliv jeho hranatost. Pravidelný vícehran tedy může mít vyšší kruhovitost než excentrická elipsa. Čím víc hran pravidelný vícehran má, respektive čím blíže leží ohniska elipsy k sobě, tím vyšší je kruhovitost měřeného objektu. Mezinárodní organizace pro normalizaci (ISO) definuje kruhovitost objektu jako rozdíl mezi poloměry kružnic vepsané a opsané pro tento tvar. Tedy mezi maximální velkostí kružnice, která se ještě vejde do obrysu tohoto objektu a minimální velikostí kružnice, do které se naopak vejde samotný objekt. [3]

Naskýtá se definovat kruhovitost jako rozdíl maxima a minima změřených průměrů objektu. Existují však mnohé objekty, které si drží konstantní šířku, ovšem kulaté zdaleka nejsou. Příkladem je Reuleaxův trojúhelník (obrázek 2.1). Tento postup tedy není vhodný. Dále je dobré si uvědomit, že kruhovitost nedefinuje odchýlení objektu od určité osy rotace, pouze jeho tvar. [3]



Obrázek 2.1 Reuleaxův trohúhelník

Měření kruhovitosti vyžaduje znalost tvaru obrysu měřeného objektu. Ten lze získat rotováním tohoto objektu a postupným měřením vzdálenosti povrchu od osy rotace.

V případě, že měření nevyžaduje vysokou přesnost, použije se ke zjištění kruhovitosti měřeného objektu pouze jeden obrys. Výsledky této metody jsou velmi náchylné k ovlivnění kvůli chybě zavedené výchylkou osy rotace od středu objektu. Vzhledem k tomu, že přístroj, který v rámci práce konstruuji má být použit za účelem měření válců, koulí a podobných objektů s obrysem blízkým kružnici, mohu použít toto jednodušší měření a chybu osy zanedbat. [3]

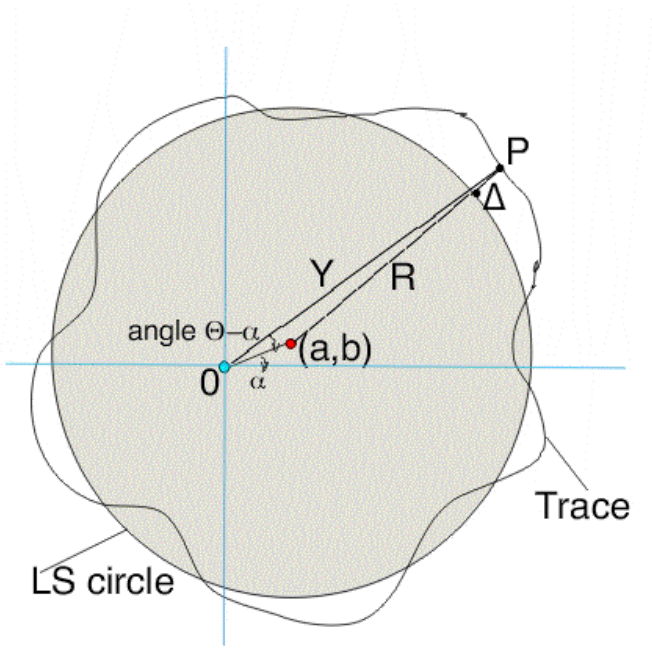
2.1 Metody měření

Metody měření jsou následující:

- Největší vepsanou kružnicí (*maximum inscribed circle)*
    - za odchylku od kruhovitosti je považována největší vzdálenost obrysu od největší vepsané kružnice.
- Nejmenší opsanou kružnicí (*minimum circumscribed circle)*
    - zde jde naopak o největší odchylku od nejmenší opsané kružnice
- Rozdílem kružnic (*minimum zone circle)*
    - za odchylku se považuje rozdíl poloměrů kružnic opsané a vepsané měřeného tvaru.
- Nejmenších čtverců (*least squares circle)*
    - za odchylkou od kruhovitosti je považována vzdálenost mezi dvěma nejvzdálenějšími body uvnitř a vně kružnice, která leží na objektu tak, že plocha vykrojená obrysem uvnitř kružnice se rovná ploše, kterou má obrys vně.

První a druhá metoda mají výhodu ve své jednoduchosti a mohou být vhodné pro určité na přesnost méně náročné účely. Metoda rozdílem kružnic je robustnější a má výhodu, že její výsledky jsou uznávány standardem ISO. Já pro účel této práce zvolil metodu nejmenšími čtverci, protože se jedná o metodu nejrobustnější a protože není účelem u měřených objektů splňovat zmíněný standard. [4]

2.2 Metoda nejmenších čtverců



Obrázek 2.2 Diagram metody [3]

Po obvodu měřeného objektu je provedeno N měření Y_i odchylky povrchu od osy rotace v úhlech $\theta i \{ i = 1, \ldots, N \}$ rozmístěných rovnoměrně. Parametry kružnice, kterou budeme tak jako v obrázku 2.2 vepisovat do zkoumaného tvaru, odhadneme pomocí metody nejmenších čtverců:

$$\rho = \frac{1}{N} \sum_{i=1}^{N} Y_i$$

$$a = \frac{2}{N} \sum_{i=1}^{N} Y_i cos(\theta_i)$$

$$b = \frac{2}{N} \sum_{i=1}^{N} Y_i sin(\theta_i)$$

Odchylka v úhlu θ_i je potom

$$\Delta = Y_i - \rho - acos(\theta_i) - bsin(\theta_i)$$

Odchylku od kruhovitosti celého objektu potom definuji jako aritmetický průměr jednotlivých odchylek. Čím je odchylka menší, tím blíže má obrys objektu ke kružnici. [3]

Kapitola 3

Použitá zařízení

Během konstrukce bylo použito větší množství součástek a zařízení buď již zcela funkčních, nebo s nutnou modifikací. V této části práce podrobně představím jednotlivá zařízení a jejich parametry.

3.1 Základní deska

Hlavní součástí přístroje je základní deska s názvem K20-main, která je produkována firmou Brázdil s.r.o. (popis viz. příloha č. 1), původně vyvinutá pro řízení laboratorních přístrojů. Základní desku bylo nutné upravit, jelikož sama o sobě neměla rozhraní na příjem dat z polohového senzoru. Dále bylo nutné drobně upravit hardwarové zapojení procesoru. Tyto úpravy jsou podrobně rozebrány v části *Realizace*. Schéma základní desky i s popisem lze nalézt v příloze č. 2.

3.2 Procesor

Pro výpočty, řízení krokového motoru a zpracování dat z polohového senzoru je použit procesor typu ARM. Autorem této architektury je firma ARM Holdings. Tyto procesory se vyznačují nízkou spotřebou elektrické energie a nízkou pořizovací cenou. Jednotky architektury ARM jsou také vhodné pro řízení díky tomu, že jsou schopny velmi rychle vykonávat instrukce. Výhodou pro tento konkrétní projekt je vedle zmíněné nízké pořizovací ceny také to, že není nutné procesor díky jeho malé spotřebě chladit, což snižuje náročnost realizace.

Mezi typické vlastnosti těchto procesorů patří to, že do paměti lze přistupovat pouze instrukcemi Load nebo Store, což je důležité pro optimalizaci jednotky. Dále to, že se částečně překrývají registry a že existuje možnost podmíněného vykonávání instrukcí.

Konkrétně je použit 100 pinový, 32-bitový procesor Freescale K20. Výrobcem je firma Freescale a jednotka je založena na architektuře ARMv7. V dokumentu *ARM Architecture Reference Manual* firmy ARM Holdings lze nalézt informace o tom, co je každý procesor této architektury povinen splňovat. Použitý procesor spadá do třídy Cortex-M4, kde M značí Microcontroller.

Schéma připojení procesoru se nachází v příloze č. 2. Na schématu je vidět označení jednotlivých pinů a způsob připojení periferií včetně krokového motoru a polohového senzoru.

3.3 Polohový senzor

Jako senzor polohy je použit tyčový měřicí přístroj čínské provenience s posuvnou sondou. Tento senzor měří s přesností na jednu tisícinu milimetru s rozsahem 0-25,4 mm. Senzor má již implementovanou vnitřní elektroniku a její součástí je grafický display, který zobrazuje měřenou hodnotu v reálném čase. Ovládání přístroje spočívá v možnosti jeho zapnutí a vypnutí, vynulování a přepnutí mezi metrickým a imperiálním systémem jednotek.

Tento senzor ovšem nemá kompletní možnost komunikace. Původně uvažovaný senzor měl být schopen komunikovat po rozhraní USB. Dostupný byl nakonec senzor se sériovým synchronním výstupem, který se ukázal pro přímé připojení k CPU jako vhodnější. Bylo totiž možné použít SPI modul procesoru a poměrně jednoduchý ovladač. Součástí této práce je analýza výstupu senzoru a implementace komunikace po rozhraní SPI.

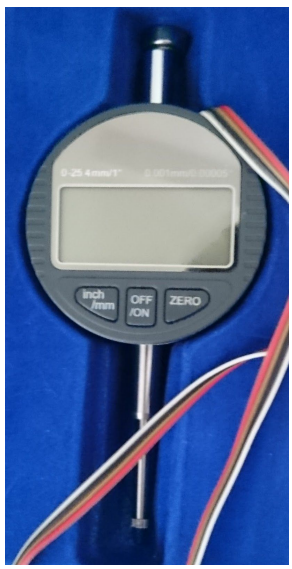
Fyzická podoba je zobrazena na obrázku 3.1. Lze vidět posuvnou sondu, pomocí jejíž výchylky měří přístroj polohu. Na obrázku je již připájen plochý kabel sloužící pro přenos dat na základní desku. Tento kabel byl během analýzy výstupu připojen k osciloskopu. Výsledek je shrnut v části práce 4.1 Analýza polohového senzoru.

3.4 Magnetický senzor

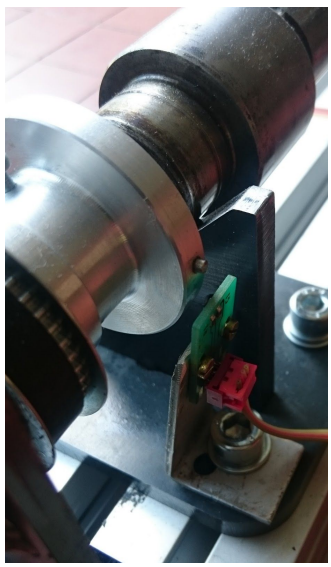
Senzor reference je připojen na binární vstup a je tvořen hallovou sondou zapojenou jako spínač s otevřeným kolektorem. Senzor je na obrázku 3.2.

Pokud se měří více obrysů na stejném objektu, je potřeba aby měření začínalo pod stejným úhlem. K tomu slouží referenční poloha označená na otočné ose magnetem. Po spuštění měření se osa otáčí tak dlouho, než magnetický senzor zaznamená tento magnet.

Přesnost tohoto řešení není ideální, ale pro mé účely dostačující. Pokud by řešení vyžadovalo přesnější nastavení reference, dalo by se použít inkrementální čidlo s přesně definovanou nulou.



Obrázek 3.1: Polohový senzor



Obrázek 3.2: Magnetický senzor

Kapitola 4

Realizace

Realizace se skládá ze tří částí. Jako první byl pomocí osciloskopu analyzován výstup polohového senzoru. Na základě informací z tohoto měření byl zjištěn obecný formát odesílaných dat. Následovalo upravení základní desky, aby byla schopna tato data předat procesoru, hardwarová úprava zapojení CPU a nakonec úprava firmware procesoru. Poslední částí realizace byla tvorba rozhraní pro dotykovou obrazovku.

4.1 Analýza polohového senzoru

Rozhraní SPI (serial peripheral interface) je typu Master/Slave. Řídicí zařízení (Master) vysílá data a zároveň patřičný hodinový takt. V této implementaci je řídicím prvkem senzor. Bude nás zajímat především datový kanál a způsob, jakým jsou data kódována. Abychom byli dále schopni data správně interpretovat, musíme zjistit několik důležitých informací. Jedná se o následující:

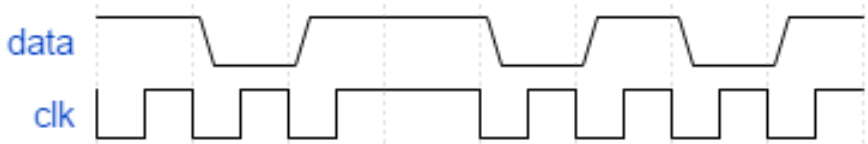
1. Zda načítat datové bity při sestupné, nebo při náběžné hraně hodin
2. Zda se přenáší binární hodnota celého čísla na obrazovce, nebo každá pozice zvlášť
3. Kolik bitů je použito k přenosu
4. Jakým způsobem je kódována záporná hodnota

Postup měření je takový, že vždy nastavíme manuálně polohu sondy senzoru na hodnoty které zkoumáme a poté porovnáme naměřené průběhy.

Poznámka: pro kreslení časovacích digitálních diagramů byl použit volně dostupný nástroj WaveDron [6].

4.1.1 Určení řídicí hrany hodin

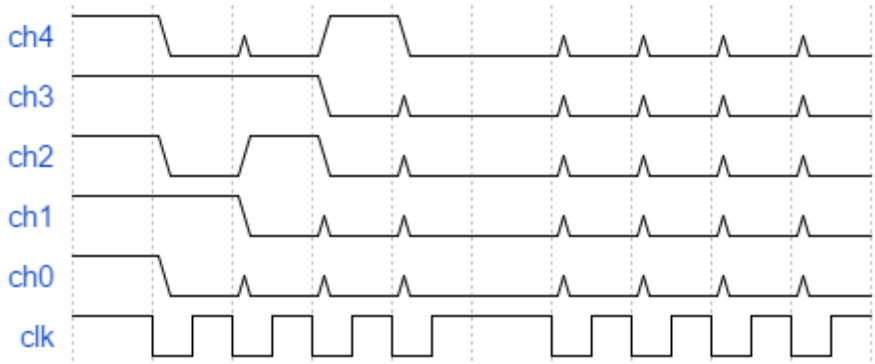
Pro zjištění reakce na hranu hodin stačí jediné měření. Měřením jsem zjistil, že datové bity jsou delší, než hodinové takty. Jeden takt hodin (okamžik, kdy jsou hodin v poloze 1) se vždy vejde do datového bitu. Takže v případě přenosu jednoho bitu informace zasahují do daného datového bitu obě hrany hodin - náběžná i sestupná. Je tedy jedno na které hraně budeme data číst. Tento fakt je vidět na obrázku 4.2. Upravená fotka obrazovky osciloskopu z tohoto měření se nachází v příloze 1.



Obrázek 4.1: Hrany hodin

4.1.2 Princip přenosu čísla a délka dat

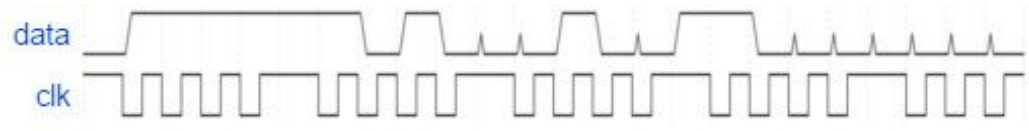
Pro určení způsobu, jakým se naměřené číslo přenáší byly na osciloskopu změřeny průběhy postupně pro hodnoty 0.000, 0.001, 0.002, 0.003 a 0.004. Tyto průběhy musí jasně stanovit, jakým způsobem se přenáší hodnota na poslední pozici obrazovky. Pro zjištění, zda se přenáší pozice zvlášť, nebo celé číslo najednou poslouží průběh pro hodnotu 25.695, který je zároveň použit pro další měření.



Obrázek 4.2: Formát dat

Na obrázku 4.2 jsou vidět průběhy pro jednotlivé hodnoty. Kanál ch0, kde byla přenášená hodnota 0.000, by měl na posledním místě mít binární hodnotu 0, tedy všechny příslušné bity v nule. Oproti němu kanál ch1 by měl mít LSB na hodnotě 1, jelikož přenáší číslo 0.001. Na obrázku jsou hodnoty jednotlivých bitů popsány. Vidíme, že LSB je bit na první náběžné hraně hodin a data postupují vpravo. Upravená fotka obrazovky osciloskopu z tohoto měření se nachází v příloze 2.

Dalším úkolem je zjistit, zda se přenáší čísla na obrazovce po jednom, nebo vcelku a kolik bitů je potřeba k přenosu. Pro toto měření je potřeba průběh dat pro větší číslo. Byla zvolena hodnota 25.695, která je zároveň maximální výchylkou polohové sondy.

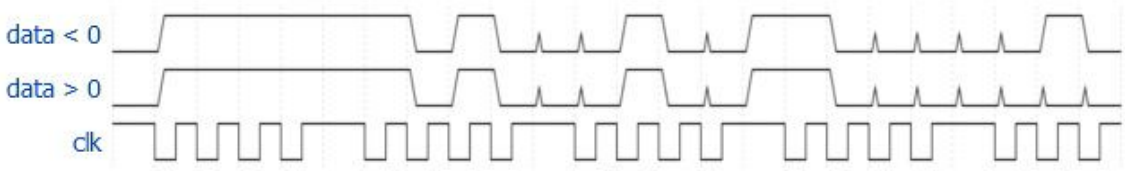


Obrázek 4.3: Délka datového řetězce

Na obrázku 4.3 je toto měření znázorněno. V popsaném prvním kanálu jsou data pro hodnotu 25.695, jejíž binární hodnota je 0110010001011111. Z tohoto měření tedy víme, že data jsou posílána ve formátu jednoho 16-bitového čísla. Další 4 bity nejsou využity, jelikož hodnota 25.695 je hraniční. Existuje možnost, že vnitřní elektronika senzoru by pro delší sondu využila i další 4 bity, nemáme však důvod toto ověřovat. Upravená fotka obrazovky osciloskopu z tohoto měření se nachází v příloze 3.

4.1.3 Přenos záporné hodnoty

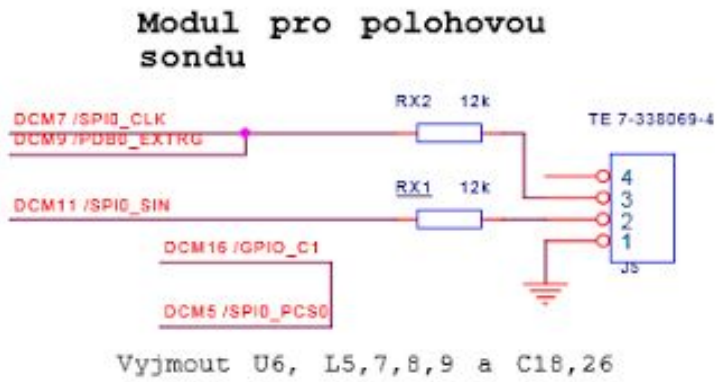
Pro zjištění způsobu kódování záporné hodnoty bylo provedeno měření hodnoty -0.001 a výsledek je znázorněn na obrázku 4.4 společně s průběhem z předchozího měření.



Obrázek 4.4: Signum

Předpokladem bylo, že informace o zápornosti hodnoty bude kódována někde za daty pro přenášené číslo. Obrázek 4.4 tento předpoklad potvrzuje na pozici 21. bitu. Tento bit je pro hodnotu -0.001 v poloze 1 a pro hodnotu 25.695 v poloze 0. Bit 21. v pořadí tedy kóduje signum, pokud je v poloze 1, přenáší senzor záporné číslo. Dalším experimentálním měření byla tato informace ověřena, k doložení stačí toto měření.

4.2 Úprava základní desky



Obrázek 4.5: Schéma připojení polohového senzoru

Aby bylo možné připojit senzor polohy rozhraním SPI, bylo potřeba na základní desce udělat drobné úpravy. Pro připojení byly použity vstupy procesoru původně určené k řízení stejnosměrných motorů. Byl odstraněn budič pro příslušný motor a jeho konektor byl použit pro připojení dat a hodin rozhraní SPI podle schématu na obrázku 4.5. Další úprava byla nutná po zjištění, že procesor nepřijímá příchozí data. Komplikace nastala z důvodu nízkého napětí na výstupu sondy. Procesor požaduje totiž na vstupu úroveň logické jedničky 2,5 V, senzor ovšem na výstupu dává pouze 1,48 V. Ideálním řešením by bylo použít převodník logické úrovně. To by však znamenalo razantní zásah do existující desky. Proto bylo zvoleno řešení pomocí pull-up odporů v procesoru společně s vnějšími sériovými odpory. Úroveň logického signálu ze senzoru na vstupu procesoru byla tímto zvýšena na 2,12 V. Tato hodnota umožňuje v tomto případě bezchybnou funkci procesoru, přestože nedosahuje požadovaného standartu.

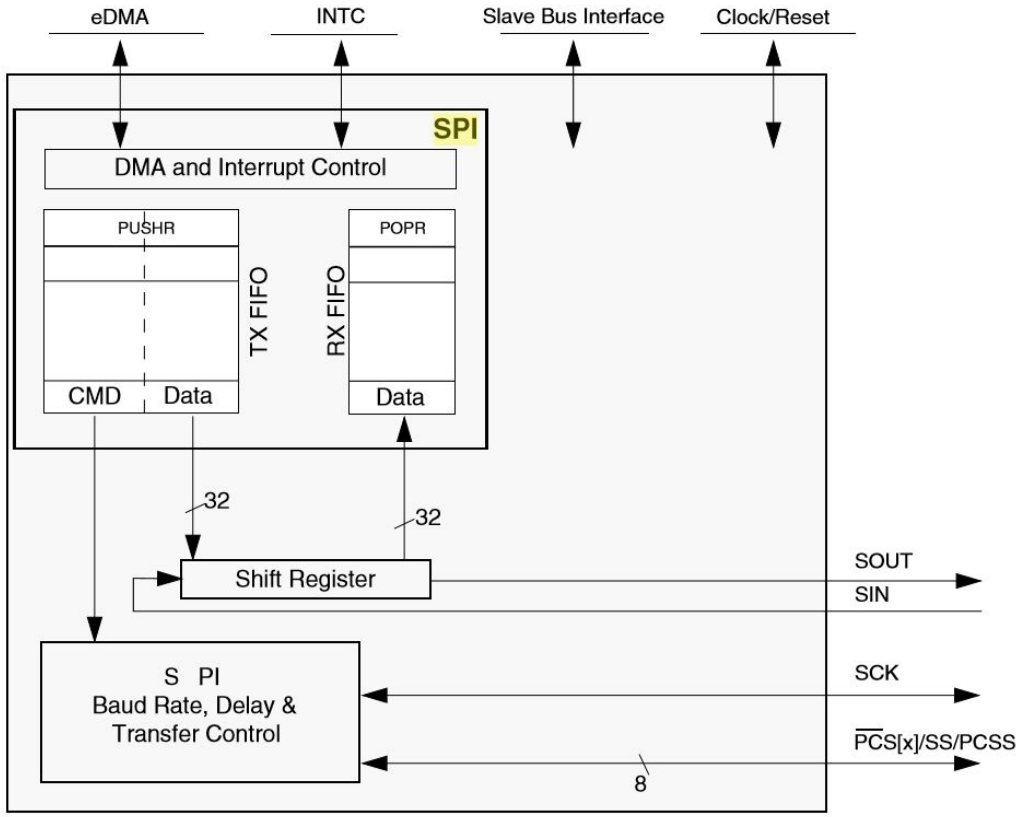


Figure 46-1. DSPI Block Diagram

Obrázek 4.6: Schéma rozhraní SPI v CPU [1]

Modul SPI v procesoru (viz. obrázek 4.6) přijímá data portem SIN do posuvného registru, odkud jsou dále zpracována. Zápis dat do registru probíhá na hranu hodin SCK, synchronizaci zajišťuje signál chipselectu PCS. Použitá sonda chip select neposkytuje, proto je v tomto projektu pro synchronizaci použit softwarově generovaný Chip-Select. Hodinové pulsy ze senzoru jsou kromě SPI rozhraní přivedeny ještě do PDB modulu procesoru. Tento modul se chová jako monostabilní klopný obvod, který generuje přerušení. V praxi to znamená, že na první detekovanou hranu překlopí PDB do jedničky a pokud se delší dobu hrana neobjeví, překlápí zpět do nuly. Tím je vygenerováno přerušení a obslužný program vyšle synchronizační puls na výstupu DCM16/SPI0_C1 (viz obrázek 4.5), který je dále přiveden na synchronizační vstup PCS/SS/PCSS modulu SPI, tedy na špičku DCM5/SPI0_PCSO procesoru.

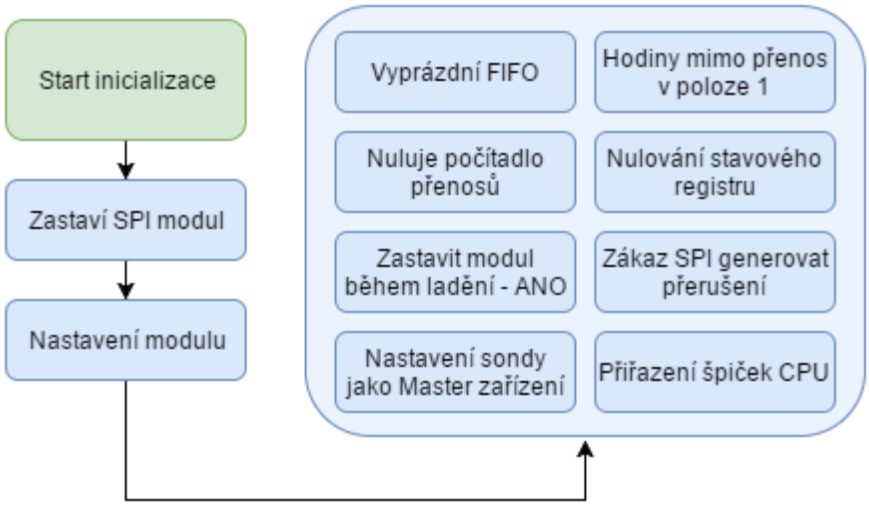
4.3 Firmware

Základní deska byla navržena pro operační systém MQX. Obecné čtení vstupů a ovládání krokových motorů bylo již naimplementováno. Driver pro dotykový display již uměl následující tři funkce: vykreslit obdélník dané barvy, vykreslit obrázek ve formátu bitmapy a vytisknout text na dané pozici. Základní deska také uměla přečíst místo dotyku uživatele. Cílem mé práce bylo přečíst data z polohového senzoru, ta pak zobrazit na display a zprostředkovat pc aplikaci. *Poznámka: pro kreslení vývojových diagramů byl použit volně dostupný nástroj Draw.io [7].*

4.3.1 Připojení polohové sondy

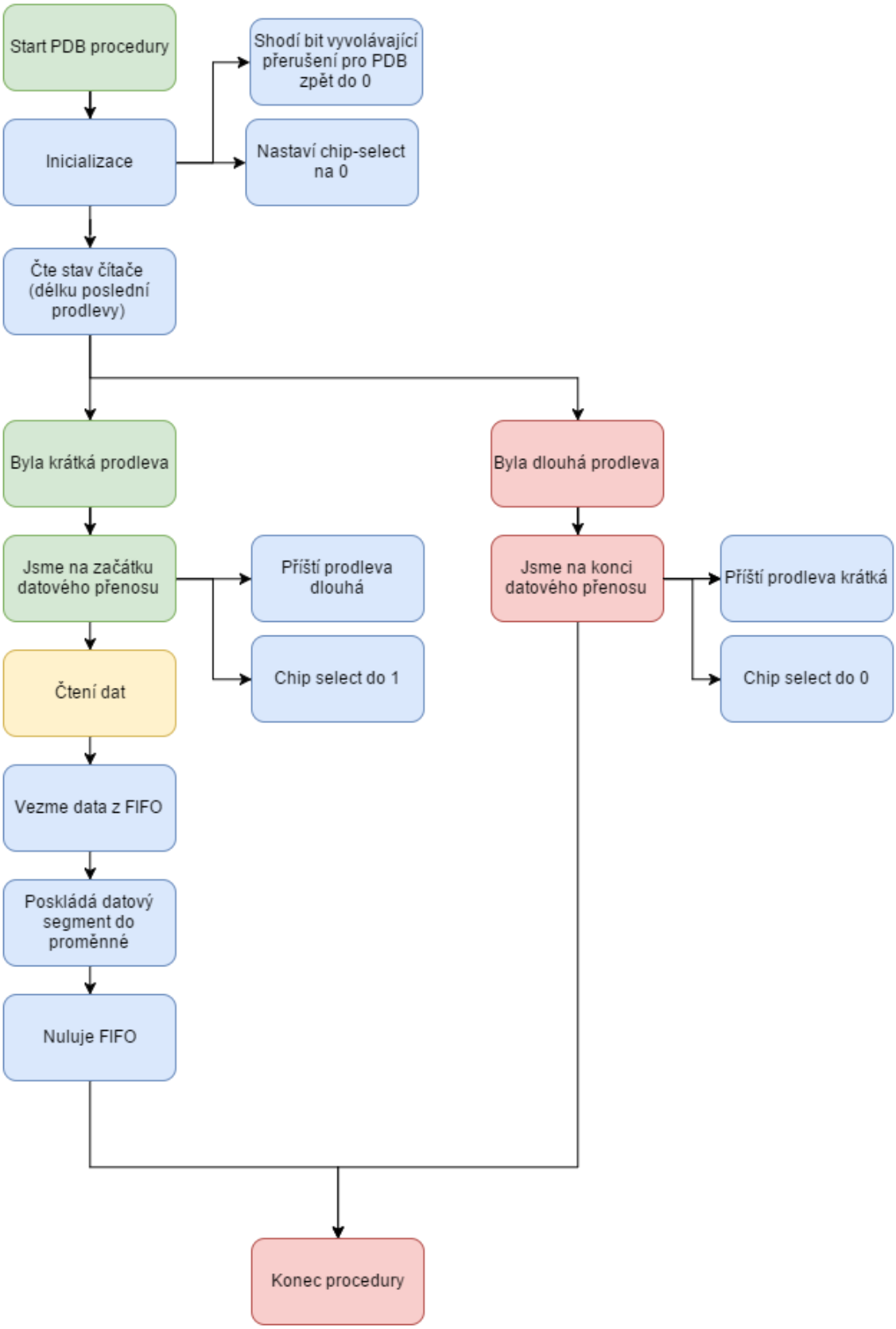
K přípojení sondy byly využity moduly SPI a PDB procesoru. Princip funkce modulu SPI je popsán v části 4.2 Úprava základní desky, spolu s vysvětlením použití modulu PDB pro generování chip-select signálu pro SPI modul.

Oba moduly se musí inicializovat. Obrázek 4.7 ukazuje inicializaci modulu SPI. Kód, který tento obrázek reprezentuje se nachází v příloze 5. Modul se nejprve zastaví. Po zastavení se mu vynulují fronty, stavové registry a počítadla. Dále se musí modulu nastavit nezbytné informace o plánovaném typu přenosu: které zařízení bude master a které slave, v jakém stavu mají být hodiny mimo přenos, zda se během ladění má přenos pozastavit, zda má modul generovat systémová přerušení. Nakonec se nastaví vstupy a výstupy modulu na příslušné špičky procesoru.



Obrázek 4.7: Inicializace SPI modulu

Procedura, kterou vykonává modul PDB je vidět na obrázku 4.8. Příslušný kód se nachází v příloze 6. Tato procedura je prováděna na začátku a na konci datového přenosu. Kromě inicializace PDB nastavuje prodlevu klopného obvodu, který generuje přerušení pro SPI modul na různou hodnotu. Díky tomu je poznat, zda se nacházíme na začátku, nebo na konci přenosu. Pokud jde o přerušení na začátku, dojde k načtení dat z FIFO fronty modulu SPI do meziproměnných. Z těchto proměnných se poskládá výsledná hodnota, která je uložena a přístupná pro použití v ostatních částech systému.



Obrázek 4.8: PDB procedura

4.3.2 Uživatelské rozhraní

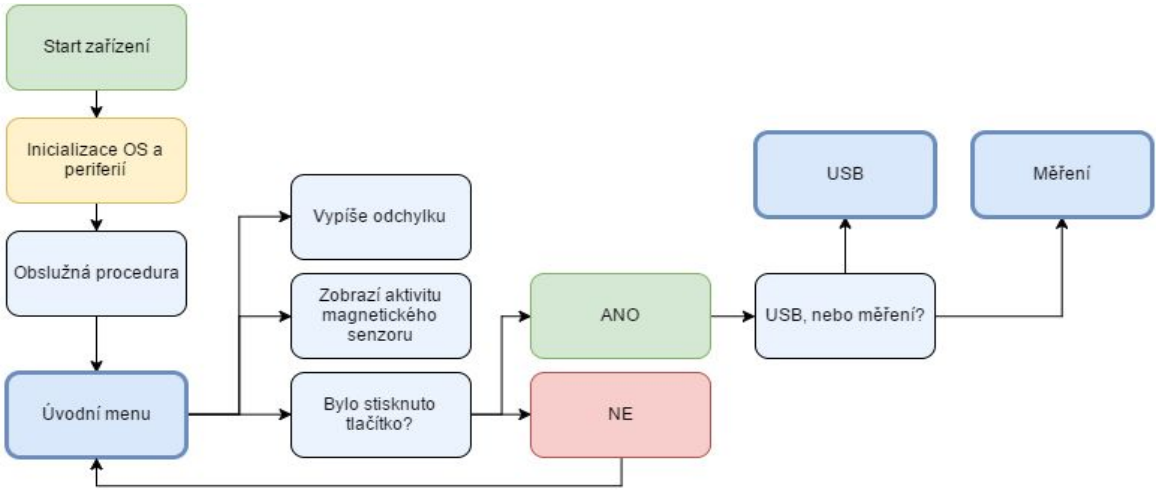
Uživatelské rozhraní, které jsem navrhl přímo pro měřicí přístroj je velmi jednoduché. Uživatel má mít možnost bez dalších komplikací a zbytečných dat provést samotné měření. V případě, že měření proběhlo v pořádku, uživatel může potvrdit uložení naměřených dat. K jakékoliv další práci s daty a jejich zobrazení slouží pc aplikace.

Zdrojový kód rozhraní je součástí periodicky volané obslužné procedury, kterou operační systém volá po své inicializaci. Během této obslužné procedury provádím samotné měření, i obsluhu obrazovky. Pro obsluhu obrazovky je vždy použita podprocedura, kterou operační systém během každého průběhu volá. Tato podprocedura je unikátní pro každou obrazovku rozhraní. Podoba úvodního menu je vidět na obrázku 4.10 a jeho obslužná procedura na obrázku 4.9. Příslušný kód se nalézá v příloze 7.

Vždy, když uživatel stiskne tlačítko, které odkazuje na jiné menu, procedura překreslí obrazovku a do proměnné Mloop_Proc uloží odkaz na funkci tohoto menu. V další iteraci obslužné smyčky se tak bude volat tato procedura, místo současné.

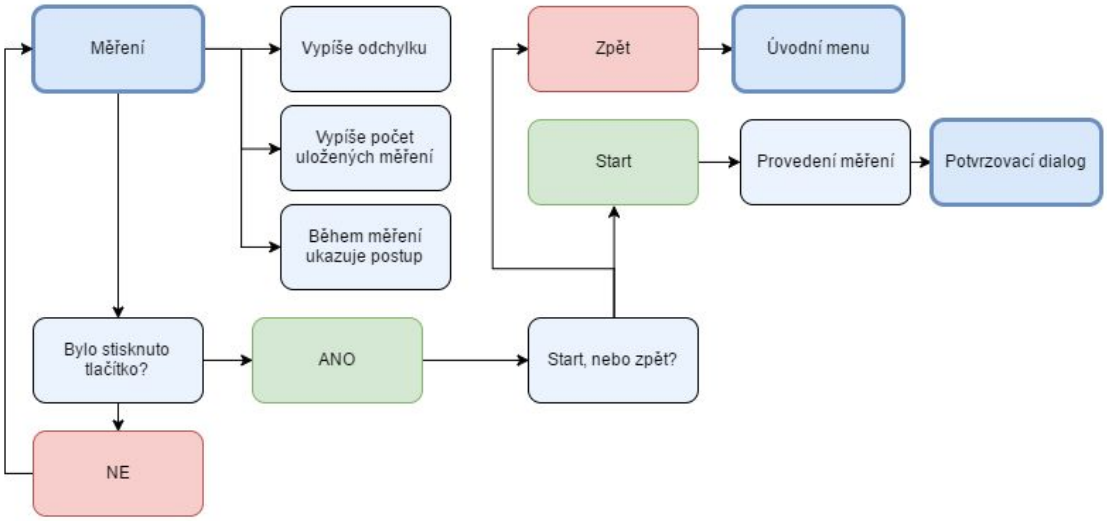
Uživatelské rozhraní se skládá z celkem čtyřech obrazovek. Následuje jejich výpis a stručný popis jejich funkce.

- Úvodní menu (obrázek 4.9 a 4.10)
  - přepíná do USB menu a do menu měření
  - vypisuje aktuální odchylku polohového senzoru
  - ikonou ukazuje, zda je zrovna aktivní magnetický senzor reference
- Menu měření (obrázek 4.11 a 4.12)
  - startuje měření
  - vypisuje aktuální odchylku polohového senzoru a počet uložených měření
  - během měření ukazuje v zápatí čárový graf signalizující postup měření
- USB menu (obrázek 4.13 a 4.14)
  - dovoluje připojení k pc tak, aby nedošlo ke konfliktu se zařízením
- Potvrzovací dialog (obrázek 4.15 a 4.16)
  - uživatel zde zvolí, zda data z provedeného měření uložit, či ne
  - zobrazuje číslo měření a jeho průměrnou odchylku

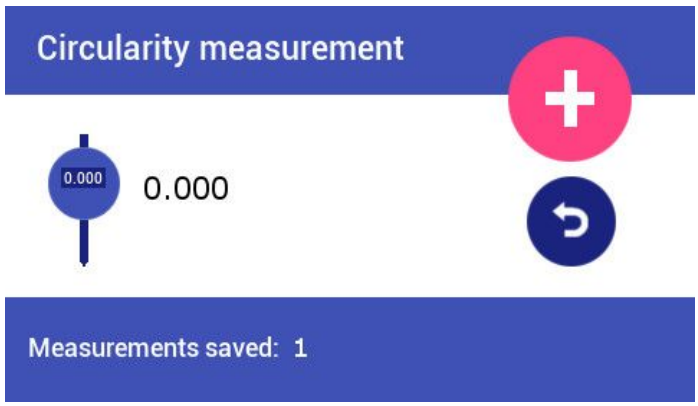


Obrázek 4.9: Diagram funkce úvodního menu

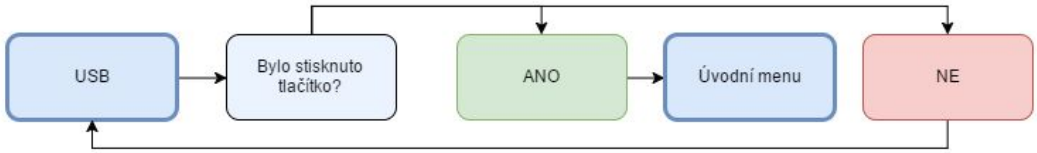
Obrázek 4.10: Úvodní menu



Obrázek 4.11: Diagram funkce menu měření

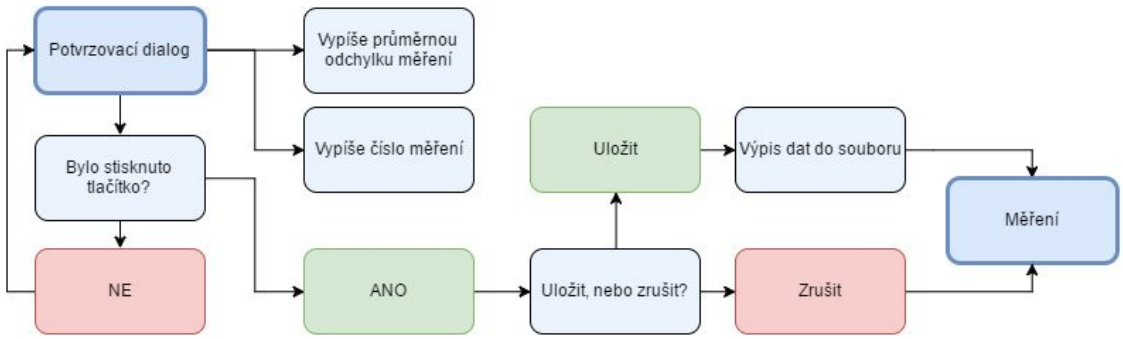


Obrázek 4.12: Menu měření

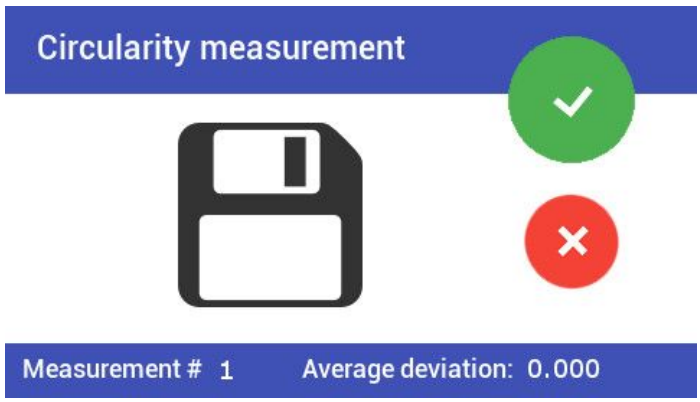


Obrázek 4.13 Diagram funkce USB menu

Obrázek 4.14: USB menu



Obrázek 4.15: Diagram funkce potvrzovacího dialogu



Obrázek 4.16: Potvrzovací dialog

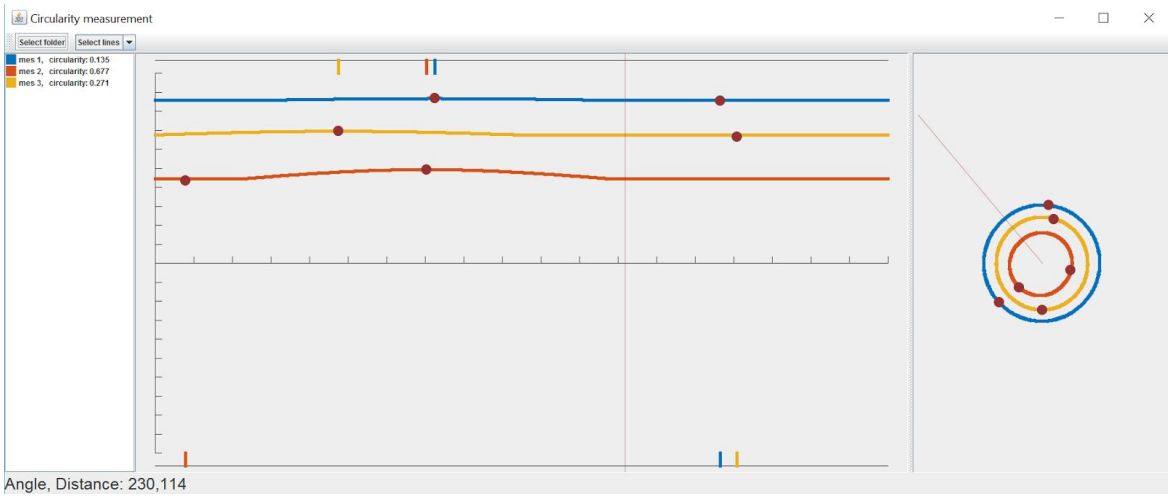
4.3.3 Připojení k PC

Připojení k pc probíhá pomocí USB sběrnice. Měřicí zařízení simuluje Mass Storage Device, chová se tedy jako standardní přenosná paměť. Uložená měření se nacházejí v textových souborech označených čísly. V každém souboru je 360 jednotlivých odchylek vypsaných do řádků. Po připojení k lze tato data načíst obslužnou aplikací.

Zařízení používá uSDHC kartu, na které má operační systém MQX aplikován vlastní File Systém. Mezi soubory v tomto prostoru patří bitmapy pro vykreslování uživatelského rozhraní a data z měření. Operační systému Windows k uSDHC kartě přistupuje přímo na sektory karty pomocí vlastního File Systému. V případě, že je zařízení připojeno přes USB k pc, může dojít ke konfliktu mezi oběma File systémy. Takový konflikt končí zamrznutím systému. Nastává v případě, že například zařízení chce vykreslit na display obrázek, který je uložen na uSDHC kartě a Windows pc na tuto kartu zároveň přistupuje. Z důvodu těchto konfliktů je v systému USB menu, které zastaví použití uSDHC karty měřicím zařízením a dovolí uživateli data načíst.

4.4 Software - rozhraní

Součástí řešení měřícího zařízení je pc software psaný v jazyce Java. Tento software obstarává samotný výpočet kruhovitosti a zároveň graficky zobrazuje naměřená data. Pro každé měření také zobrazí extrémy. Celé rozhraní je vidět na obrázku 4.17. Dále následují detaily rozhraní a poté analýza zdrojového kódu.

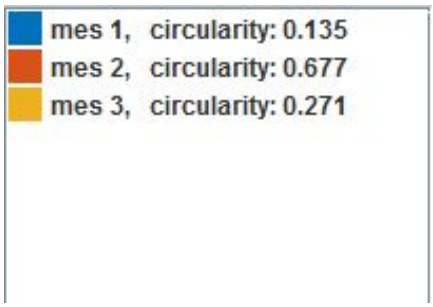


Obrázek 4.17: Uživatelské rozhraní PC softwaru

Rozhraní se skládá zleva doprava ze tří částí: listu měření a kruhovitostí, grafu naměřených odchylek a grafu obrysů měřených objektů. V záhlaví se nachází panel nástrojů, kde lze zobrazit a skrýt jednotlivá měření a změnit složku, ze které se měření načítají. V levém dolním rohu aplikace ukazuje hodnotu měření v místě, kam ukazuje kurzor uživatele a příslušný úhel, ve kterém bylo měření provedeno. Horizontální pohyb kurzoru kopíruje vertikální přímka v levém grafu a úsečka vedoucí ze středu kruhu v pravém grafu. Uživatel tak vidí jaký úhel na objektu koresponduje s naměřenou hodnotou a naopak. Aplikace také plně funguje i na dotykové obrazovce. Uživateli stačí dotykem označit místo, které ho zajímá, grafické kurzory se přesunou na dané místo a aplikace zobrazí příslušné hodnoty.

4.4.1 List měření a kruhovitostí

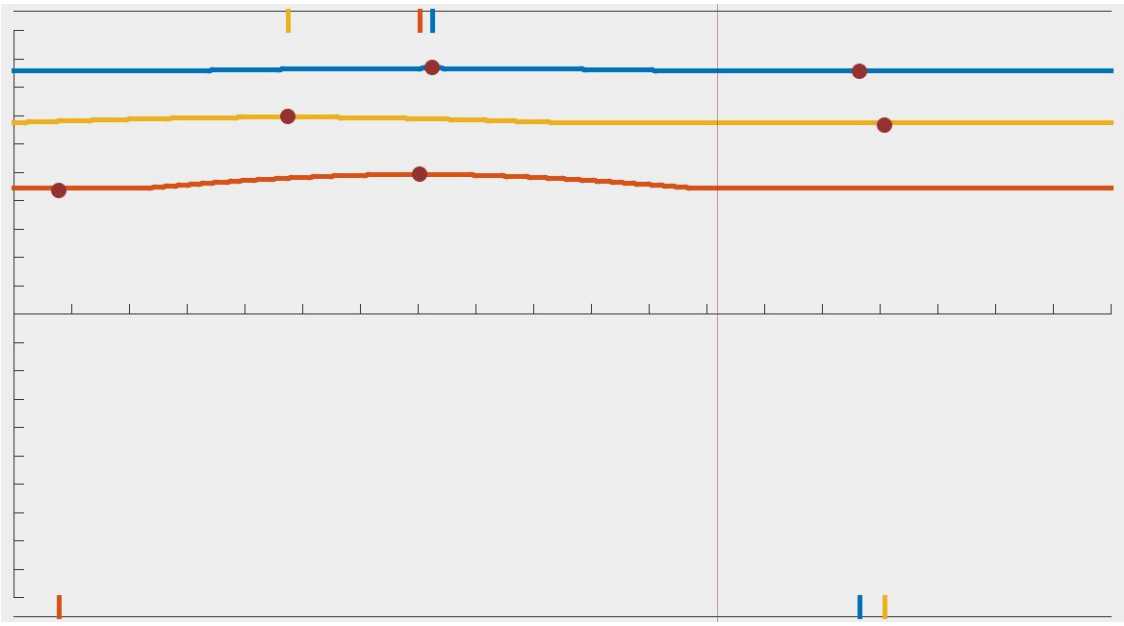
Tento list (obrázek 4.18) zobrazuje přehled měření s jejich číslem, barvou v grafech a spočtenou kruhovitostí.



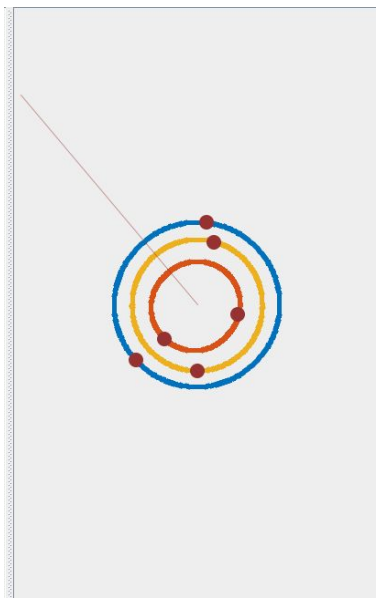
Obrázek 4.18: List měření a kruhovitostí

4.4.2 Grafy odchylek a obrysů

Graf odchylek (obrázek 4.19) zobrazuje naměřené odchylky v závislosti na úhlu objektu. Zároveň označuje extrémy tmavou tečkou a čárkou příslušné barvy. Nad grafem jsou vyznačena maxima, pod grafem jsou vyznačena minima. V obou grafech je vidět přímka, kopírující pohyb kurzoru uživatele. Graf obrysů je vidět na obrázku 4.20.



Obrázek 4.19: Graf odchylek

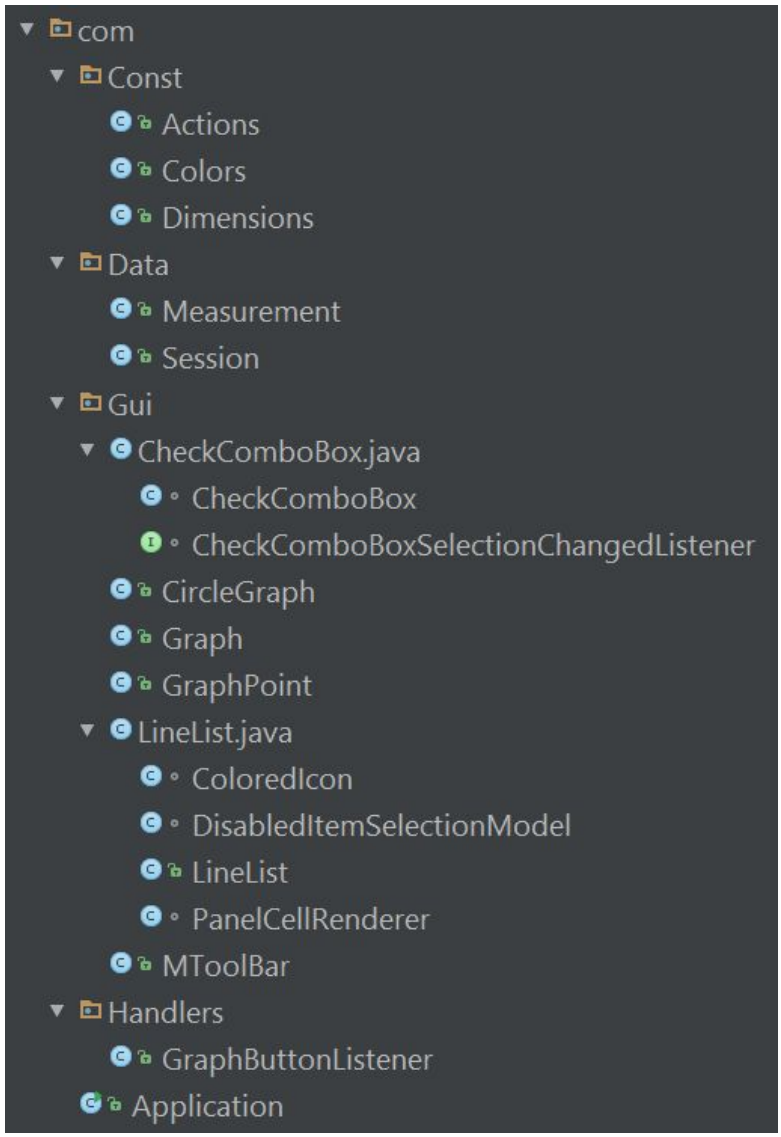


Obrázek 4.20: Graf obrysů

4.5 Software - zdrojový kód

Na obrázku 4.21 je vidět seznam tříd použitých v celé aplikaci. Stěžejní jsou třídy v balíčku Data, tedy třídy Measurement a Session. Třída Measurement reprezentuje konkrétní měření jednoho objektu. Třída Session je souhrn měření načtených ve zvoleném adresáři. Tři základní grafické komponenty popsané v části 4.4 jsou vytvářeny třídami Graph, Circle Graph a LineList.

Poznámka: v panelu nástrojů byla použita komponenta CheckComboBox z knihovny ControlsFX, která není standardní knihovnou jazyka java. [5]



Obrázek 4.21: Seznam tříd

4.5.1 Inicializace programu

Po spuštění program zobrazí dialog, kde uživatel vybere složku s daty. V případě právě provedeného měření je to příslušný disk měřicího přístroje. Program projde složku a postupně z každého souboru, jehož jméno začíná řetězcem data_, načte naměřené hodnoty (viz příloha 8). Funkce addMeasurement() třídy Session vytvoří instanci třídy Measurement, která vezme jako argument konstruktoru řetězec s cestou k souboru a během své inicializace načte ze souboru data.

Při běhu má tedy aplikace jednu instanci třídy Session, ve které je uložen list instancí třídy Measurement. V každé z nich je list naměřených hodnot. Po načtení dat ze souborů aplikace inicializuje grafické komponenty.

4.5.2 Grafické komponenty

Vykreslování grafů probíhá velmi podobně, takže pro ilustraci popíšu tento proces pouze pro graf odchylek. Nejprve se ručně vykreslí osy grafu se značkami, potom následuje zobrazení samotných dat. Provedená měření se zobrazují jedno po druhém. Pro reprezentaci dat používám třídu GraphPoint, která rozšiřuje standardní třídu Point z knihoven jazyka Java. Tato třídá má navíc informace o tom, zda je daný bod extrémem a jakou barvou bude vykreslen. Proces vytvoření množiny těchto bodů je vidět v příloze 9. Po vytvoření této množiny program vykreslí tyto body a pospojuje je čarou barvy příslušného měření. Pokud je bod extrémem, postará se program o jeho zvýraznění a zakreslení čar do záhlaví a zápatí grafu.

Kurzor reprezentovaný vertikální přímkou je vykreslován společně s celým grafem. Třída Graph tedy implementuje třídu MouseMotionListener standardních knihoven jazyka Java, konkrétně využívá metodu mouseMoved(). Při každém pohybu kurzoru se tedy překreslí celý graf a zároveň třída Graph pošle informaci o změně celé aplikaci, aby došlo k přepsání zobrazované hodnoty na poloze kurzoru a k překreslení kurzoru v grafu obrysů. Graf se překresluje velmi často a třída GraphPoint tento proces velmi usnadňuje a dělá ho přehledným.

Kapitola 5

Závěr

Cíle tohoto projektu se podařilo splnit. Jako nejnáročnější se nakonec ukázala analýza sondy a tvorba rozhraní pro display a pro PC. Samotné získání dat ze vstupu sondy a základní ovládání obrazovky se ukázalo nebýt až tak velkými problémy, jak jsem byl z počátku přesvědčen. Moji práci velmi usnadnil operační systém MQX tím, že v základním stavu již implementuje obecné čtení vstupů a ovládání výstupů. Povedlo se mi také připojit i magnetický senzor pro nájezd do reference. Tato funkce nebyla součástí zadání, ale ukázala se jako klíčová pro korektní funkčnost zařízení. Absence USB rozhraní pro polohovou sondu se nakonec ukázala jako výhoda, jelikož mohl být použit přímo SPI modul procesoru. Fotka zařízení zapojeného k měřicímu šasi se nachází v příloze 12.

Tento prototyp má stále řadu drobných nedostatků, které by měly být pro případnou produkční verzi doladěny. Měl by jít v pc aplikaci nastavit offset, aby mohl uživatel po drobném nastavení koukat na reálné průměry a nikoliv jen na odchylky naměřené sondou. Také by mělo být lépe vyřešeno šasi a umístění hardware. Uchycení sondy také není ideální. V produkční verzi by měla být sonda umístěna na posuvném vozíku, aby mohlo být bez problému měřeno více objektů na jedné hřídeli se stejným offsetem, nebo by mělo být použito více sond.

Offset pro určení kruhovitosti problém nepředstavuje, jelikož nezáleží na průměru měřeného objektu. Uchycení sondy by však určitou nepřesnost představovat mohlo, jelikož není zaručen kolmý styk tyče sondy s měřeným objektem. Tento probém by se měl před použitím zařízení vyřešit úpravou šasi sondy.

Summary

The goals of this thesis were met. The most complicated parts of the project turned out to be the analysis of the distance sensor and coding the user interface. Getting data from inputs were not as big issues as I suspected in the beginning. My work has been made easier by the MQX operating system and its default components for reading inputs and controlling outputs. Even though it was not an original goal of my work, I managed to connect a magnetic sensor for reference finding. It turned out to be a crutial feature for the functionality of the device. The absence of USB interface for the distance sensor made its connection even easier, because I was able to utilize the SPI interface of the central processing unit.

There are several small issues with the prototype, that should be resolved before actual production of the device. Offset setting should be made possible in the software. The chasis also needs to be redone. The distance sensor should be fixed to prevent it to be set in a non-perpendicular position to a measured object, or more sensors should be used.

However, offset is no issue for the actual measurement of circularity, because circularity is not affected by the objects diameter. The sensor’s position could affect the measurement and should be resolved in the next version of the chasis.

Obrázky

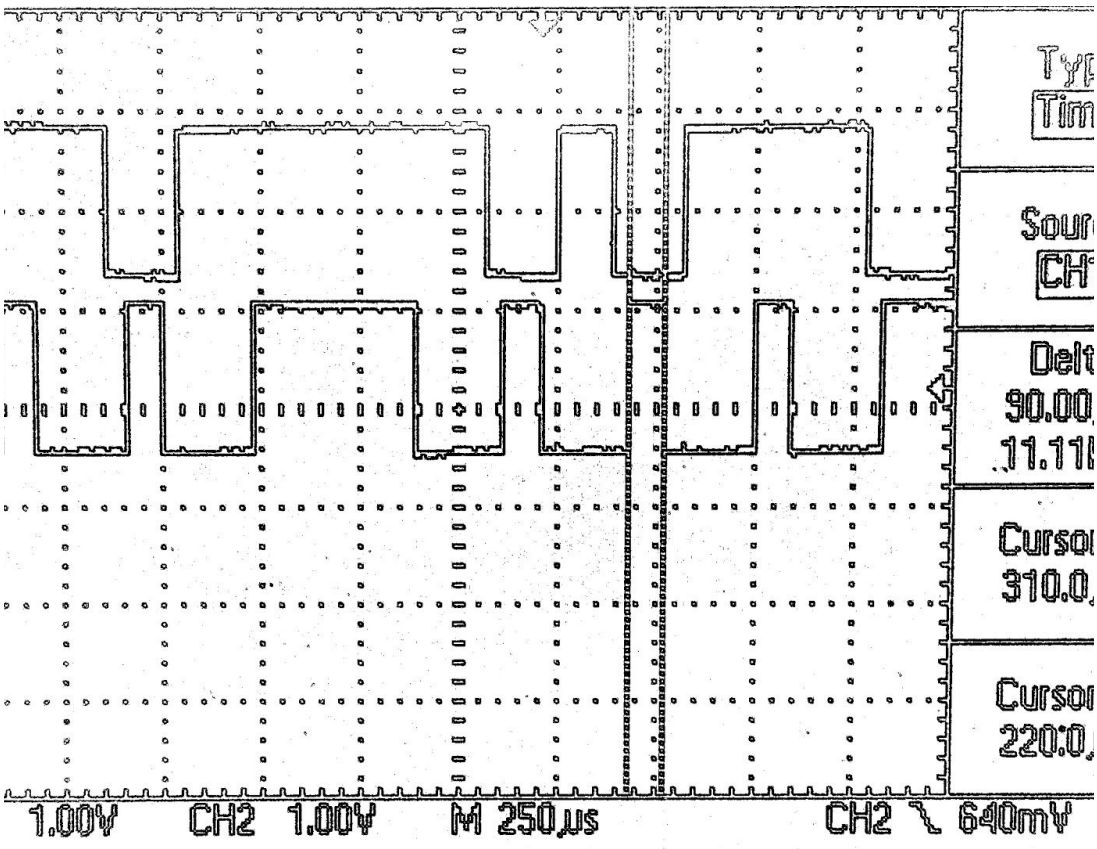
Reference

[1] K20 Sub-Family Reference Manual, Doc. No.: K20P100M100SF2V2RM, Freescale, Jun 2012.
[2] Universal Seriál Bus Specification, Revision 2.0, April 2000.
[3] NIST/SEMATECH e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/, aktualizace 30.10.2013
[4] TAYLOR HOBSON PRECISION, A guide to the Measurement of Roundness, http://www.tarkkuustuonti.fi/Kampanjat/Brochure_Roundness_Booklet.pdf, verze 26.5.2016

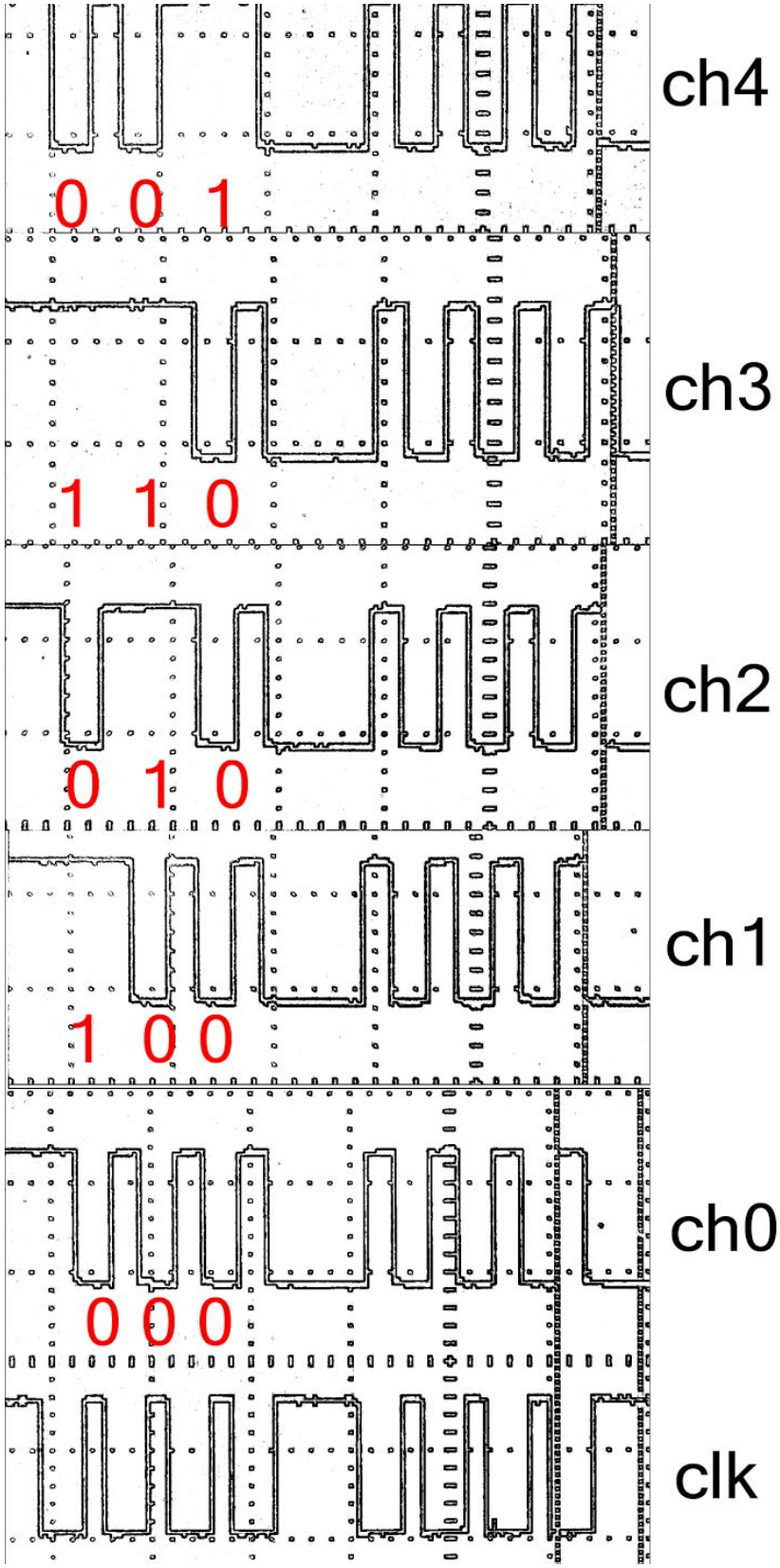
[5] **(kód)** ControlsFX 8.4.10, http://controlsfx.bitbucket.org/org/controlsfx/control/CheckComboBox.html
[6] **(nástroj)** WaveDrom http://wavedrom.com/editor.html, verze 26.5.2016
[7] **(nástroj)** JGraph Ltd. Draw, http://draw.io, verze 26.52016

Přílohy

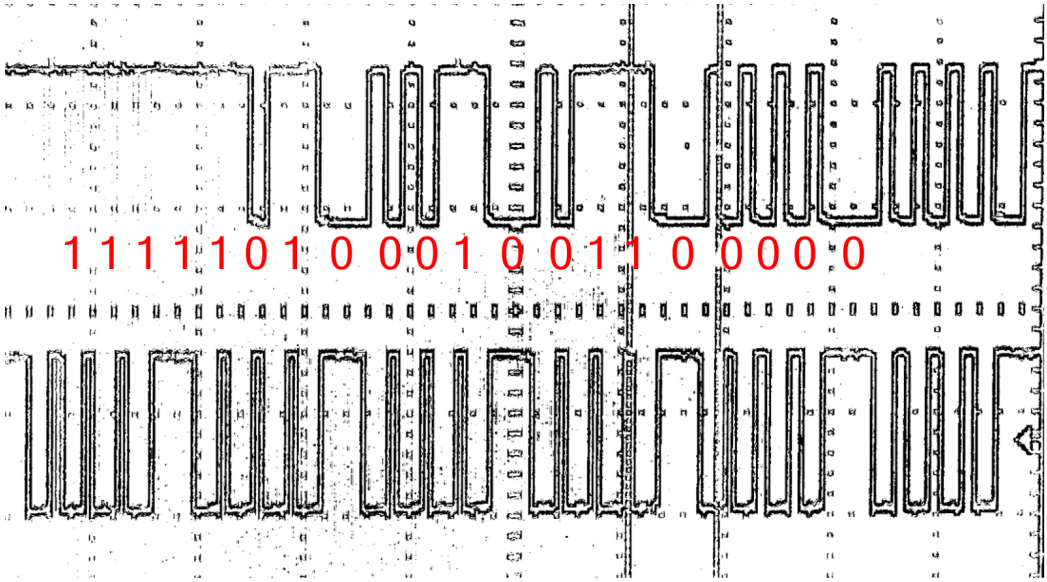
Příloha 1 - Osciloskop - hrana hodin



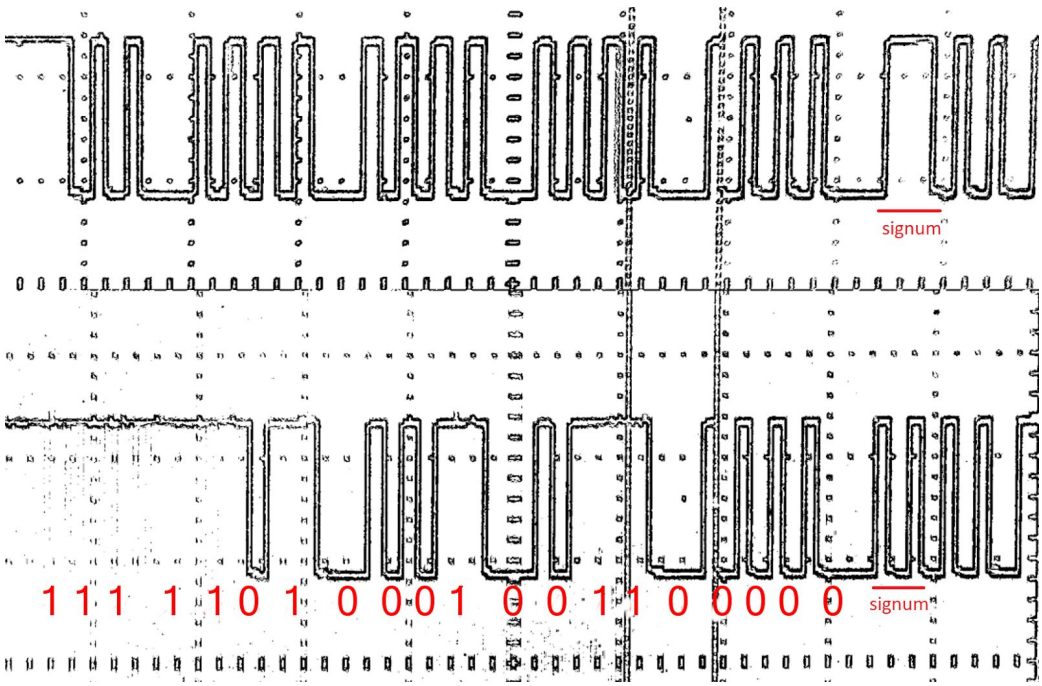
Příloha 2 - Osciloskop - formát dat



Příloha 3 - Osciloskop - délka datového řetězce



Příloha 4 - Osciloskop - signum



Příloha 5 - Inicializace SPI modulu

```
1.  // Zastaví modul SPI
2.  SPI0_MCR = SPI_MCR_HALT_MASK;
3.
4.  // Vyprázdní FIFO
5.  SPI0_MCR    =    (SPI_MCR_CLR_TXF_MASK    |    SPI_MCR_CLR_RXF_MASK    |
    SPI_MCR_HALT_MASK );
6.
7.  // Zastaví modul během ladění |  Nastavení hodin
8.  SPI0_MCR = ( SPI_MCR_FRZ_MASK | SPI_MCR_CONT_SCKE_MASK);
9.
10. // Nuluje počítadlo provedených přenosů
11. SPI0_TCR = 0x00U;
12.
13. // Nastaví sondu jako master zařízení
14. SPI0_CTAR0 = SPI_CTAR_SLAVE_FMSZ(21);
15.
16. // Hodiny jsou mimo přenos v 1
17. SPI0_CTAR0 |= SPI_CTAR_SLAVE_CPOL_MASK;
18.
19. // Vynulování stavového registru
20. SPI0_SR = SPI_SR_TCF_MASK |
21.          SPI_SR_EOQF_MASK |
22.          SPI_SR_TFUF_MASK |
23.          SPI_SR_TFFF_MASK |
24.          SPI_SR_RFOF_MASK |
25.          SPI_SR_RFDF_MASK;
26.
27. // Zakáže SPI modulu vyvolat systémová přerušení
28. SPI0_RSER = 0;
29.
30. // Nastaveni pinů
31. PORTC_PCR4 = (uint32_t)(0x00000204UL);    // CS
32. PORTC_PCR5 = (uint32_t)(0x00000204UL);    // CLK
33. PORTC_PCR7 = (uint32_t)(0x00000204UL);    // IN
```

Příloha 6 - Procedura modulu PDB

```
GPOUT_OFF_SPICS;  // Nastaví chip-select na 0

// Bit vyvolávající přerušení pro tuto proceduru
// shazuji zpět do 0
PDB0_SC &= ~PDB_SC_PDBIF_MASK;

// Čtu stav čítače (délku poslední prodlevy)
// Pokud je menší než nula, jedná se o začátek datového přenosu
if (PDB0_IDLY < 0x150) {
    PDB0_IDLY =  0xc000;   //  nastavuji délku příští prodlevy

    // Čtení dat z FIFO
    SPI_KM_d1  =  SPI0_POPR & 0x3F;
    SPI_KM_d2  =  SPI0_POPR & 0x3F;
    SPI_KM_d4  =  SPI0_POPR & 0x3F;

    // Poskládání celého datového segmentu
    SPI_KM_d1  = SPI_KM_d1>>26;
    SPI_KM_d1 |= SPI_KM_d2>>20;
    SPI_KM_d1 |= SPI_KM_d3>>14;
    SPI_KM_d1 |= SPI_KM_d4>> 8;

    //Data uložím do proměnné
    Deviation = SPI_KM_d1;
    // Zastavení modulu SPI
    SPI0_MCR = SPI_MCR_HALT_MASK;
    // Nulování FIFO modulu SPI
    SPI0_MCR = (SPI_MCR_CLR_TXF_MASK | SPI_MCR_CLR_RXF_MASK |
                                                SPI_MCR_HALT_MASK);
    SPI0_MCR = ( SPI_MCR_FRZ_MASK);
}
// Konec datového přenosu
else {
    GPOUT_ON_SPICS;     // Chip-select do 1
    PDB0_IDLY = 0x100;  // Nastaví krátkou příští prodlevu
}
// Zápis parametrů PDB modulu
PDB0_SC |=  PDB_SC_LDOK_MASK;
```

Příloha 7 - Procedura úvodního menu

```
/**
 * Procedura obsluhující úvodní menu
 */
void startMenu(void) {
    // Pokud došlo ke stisku, uloží do key id tlačítka
    key = Kom_Zona_Uk->KZ_TSI_TestKey(TSI_FIRE_ON);

    printDeviation();  // Vytiskne na display současnou odchylku polohového senzoru

    if (!(GPIOE_PDIR & 0x2000000))  {      // Pokud je magnetický senzor aktivní
       Kom_Zona_Uk->KZ_AjsPan_draw(255,112,"a:\\BMP\\magnet_on.bmp");
    }
    else{                                  // Pokud aktivní není
       Kom_Zona_Uk->KZ_AjsPan_draw(255,112,"a:\\BMP\\magnet_off.bmp");
    }

    switch (key) {
       case 0x01:  // Tlačítko start
          Kom_Zona_Uk->KZ_AjsPan_draw(0,0,"a:\\BMP\\measuring.bmp");
          Mloop_Proc = sessionMenu;
          break;
       case 0x02:  // Tlačítko USB
          Kom_Zona_Uk->KZ_AjsPan_draw(0,0,"a:\\BMP\\usb.bmp");
          Mloop_Proc = usbMenu;
          break;
       }
}
```

Příloha 8 - Vytvoření datové struktury

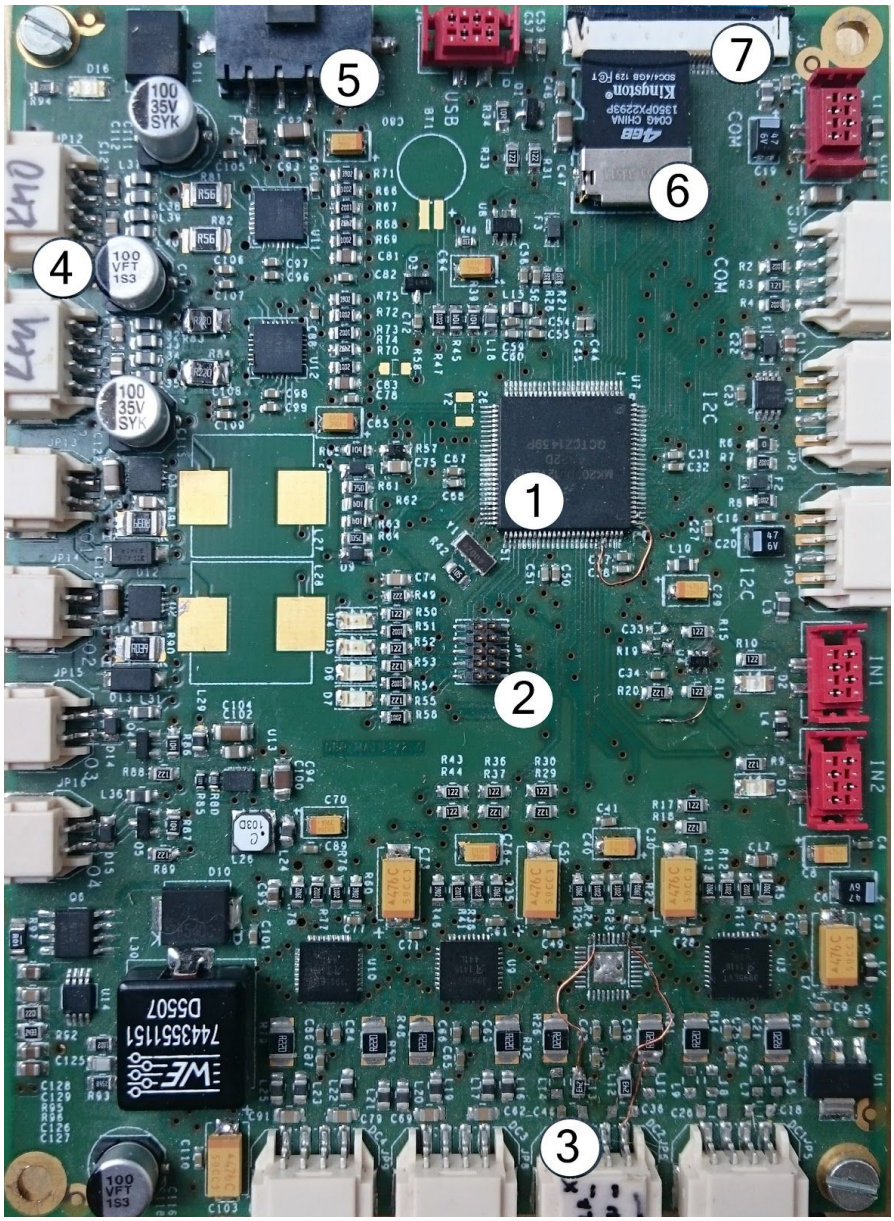
```
File[] listOfFiles = folder.listFiles();

for(int i=0; i<listOfFiles.length; i++){
    if(    listOfFiles[i].isFile()
           && listOfFiles[i].getName().toLowerCase().startsWith("data")){
        if(session == null){
            session = new Session();
        }
        session.addMeasurement(listOfFiles[i].getAbsolutePath());
    }
}
```

Příloha 9 - Vytvoření list GraphPoint objektů

```
1.  // Create graph points
2.      if (graphPoints.size() == id) {
3.          graphPoints.add(new ArrayList<GraphPoint>());
4.          for (int i = 0; i < measurement.LENGTH; i++) {
5.              int x = (int) (i * xScale + Dimensions.BORDER_GAP);
6.              int y = (int) ((session.maxScore - measurement.data.get(i)) *
7.                                          yScale / 2 + Dimensions.BORDER_GAP);
8.
9.              GraphPoint gp;
10.             if (Double.compare(measurement.max(), measurement.data.get(i)) == 0
11.              || Double.compare(measurement.min(), measurement.data.get(i)) == 0)
12.             {
13.                 gp = new GraphPoint(x, y, false, true, measurement.data.get(i));
14.                 extremes++;
15.             } else {
16.                 gp = new GraphPoint(x, y, false, false, measurement.data.get(i));
17.             }
18.
19.             graphPoints.get(id).add(gp);
20.         }
21.         linePoints = graphPoints.get(id);
22.     } else {
23.         linePoints = graphPoints.get(id);
    }
```

Příloha 10 - Schéma základní desky

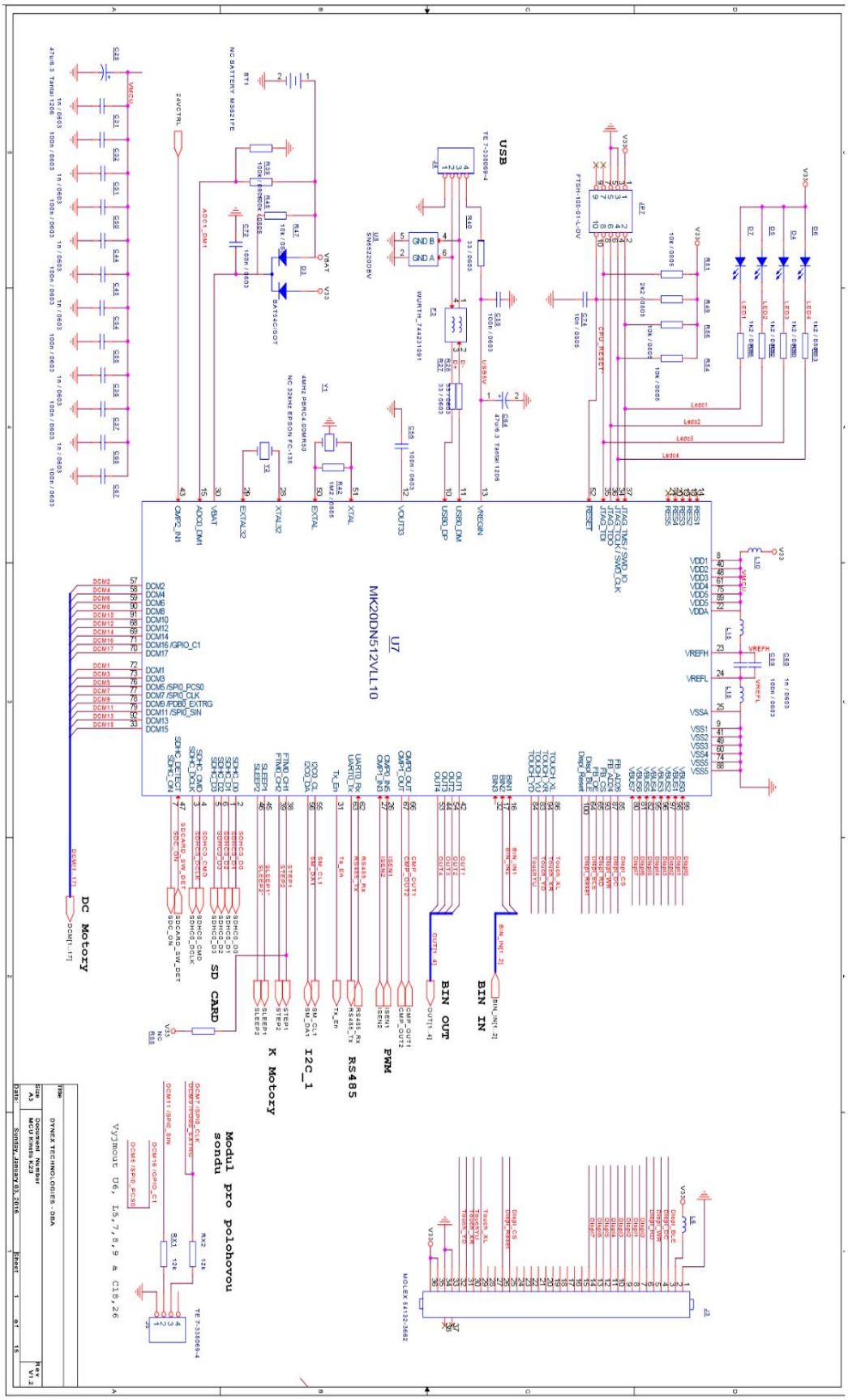


Obrázek p.1 Popis základní desky

Popis pro tento projekt důležitých součástí základní desky:

1. ARM procesor
2. Konektor pogramovacího rozhraní JTAG
3. Upravený vstup pro polohový senzor
4. Výstupy pro krokové motory
5. Napájecí konektor
6. Čtečka microSD karet
7. Konektor pro připojení dotykové obrazovky

Příloha 11 - Schéma zapojení procesoru



Příloha 12 - Fotka výsledného zařízení