ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vitoušek** Jméno: **Martin** Osobní číslo: **466145**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Ovládací a měřící software pro testování roznětnic Armády ČR

Název bakalářské práce anglicky:

Control and measuring software for testing detonators for the Czech Army

Pokyny pro vypracování:

1. Proveďte rešerši na téma programovatelných desek Arduino
2. Analyzujte dodané zařízení a detailněji rozveďte možnosti dodané ovládací desky
3. Navrhněte a implementujte řídící a ovládací software
4. Proveďte testování funkčnosti finálního produktu
5. Kriticky zhodnoťte dosažené výsledky

Seznam doporučené literatury:

[1] B.W. Kernighan, D.M.Ritchie: The C programming language
[2] BLUM, Jeremy. Exploring Arduino: Tools and Techniques for Engineering Wizardry
[3] ATMEL CORPORATION. ATmega640/1280/1281/2560/2561 datasheet [online]

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Nečas, MSc., Ph.D., odbor mechaniky a mechatroniky FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **29.04.2019** Termín odevzdání bakalářské práce: **16.08.2019**

Platnost zadání bakalářské práce: _____

Ing. Martin Nečas, MSc., Ph.D. podpis vedoucí(ho) práce	prof. Ing. Milan Růžička, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

3.5.2019
Datum převzetí zadání

Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

FAKULTA
STROJNÍ
ČVUT V PRAZE

Bakalářská práce

Ovládací a měřicí software pro testování roznětnic Armády ČR

Martin Vitoušek

Ústav mechaniky, biomechaniky a mechatroniky

Vedoucí práce: Ing. Martin Nečas, MSc., Ph.D.

9. srpna 2019

Poděkování

Rád bych poděkoval Ing. Martinu Nečasovi, MSc., Ph.D., který vedl tuto bakalářskou práci, za jeho věcné připomínky a ochotu, se kterou mě vedl ke kýženému cíli. Dále bych chtěl poděkovat celé mé rodině a přátelům za podporu během studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 9. srpna 2019 ……

České vysoké učení technické v Praze

Fakulta strojní

© 2019 Martin Vitoušek. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě strojní. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Vitoušek, Martin. *Ovládací a měřicí software pro testování roznětnic Armády ČR*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta strojní, 2019.

Abstrakt

Cílem práce je návrh a implementace ovládacího a měřicího softwaru pro zařízení určené k testování roznětnic Armády České Republiky. Měřicí aparatura se skládá z běžného počítače, prototypu testovacího zařízení s deskou Arduino a multimetru Hewlett Packard 34401A. Systém je ovládán programem v počítači skrze grafické rozhraní, které bylo vytvořeno v programovacím jazyku C# pomocí programu Microsoft Visual Studio. Hlavní program pak řeší komunikaci se zbylými ovládacími jednotkami a celkové řízení procesu. Měřicí zařízení je řízeno vývojovou deskou Arduino Mega2560. Komunikace je zajištěna emulovanou sériovou linkou na USB kabelu. Multimetr 34401A je spojen pomocí General Purpose Interface Bus s počítačem. Komunikace multimetru a počítače je message-based.

Klíčová slova roznětnice, AČR, armáda, Arduino, C#, ovládání, měření, multimetr, GUI, sériová komunikace, GPIB

Abstract

The goal of this thesis is to design and implement control and measuring software for a system intended for the purpose of testing of blasting devices used by the Czech armed forces. The system consists of a common computer, a prototype testing device with an Arduino board and a Hewlett Packard 34401A multimeter. It is controled by the computer program through a graphical interface created in C# using Microsoft Visual Studio. The main program manages communication with remaining control units and overall process control. The measuring device is controlled by the Arduino Mega2560 development board. Communication is provided by the emulated serial line on a USB cable. Multimeter 34401A is connected to the computer via the General Purpose Interface Bus. Multimeter and computer communication is message-based.

Keywords blasting device, AČR, army, Arduino, C#, control, measurement, multimeter, GUI, serial communication, GPIB

Obsah

Seznam obrázků

Seznam zdrojových kódů

Úvod

Většina dnešních armád má ve své výzbroji zahrnuty přístroje určené k detonaci výbušných zařízení. Tyto přístroje lze obecně nazvat pojmem roznětnice. Ve výzbroji současné Armády České Republiky je mimo jiné množství roznětnic RKA a RK-1. Jedná se o kondenzátorové roznětnice staršího data výroby, které používala Československá armáda již před rokem 1989. Ačkoliv jsou dnes k dispozici modernější nástupci, velká část starých roznětnic je stále v plně funkčním stavu a je tedy možné je využít alespoň pro operace s menší prioritou nebo v méně náročném prostředí. Tímto krokem by se dalo docílit značné finanční úspory při nákupu nového vybavení. Důležitým faktorem však zůstává to, že starší roznětnice nemusí být všechny v dostatečně dobrém stavu a tudíž je potřeba jejich vlastnosti testovat. Stejně tak je potřeba průběžně kontrolovat stav již používaných roznětnic při jejich údržbě. Proto bylo vyvinuto jednoduché zařízení, které má sloužit právě k tomuto účelu a úkolem této práce bylo vytvoření software pro řízení celého zkušebního procesu.

Moje osobní motivace spočívala hlavně v tom, že jsem chtěl rozšířit své dovednosti v oblasti práce s Arduinem, programování a elektrotechniky. Během práce jsem byl nucen kombinovat znalosti získané při studiu ve škole, mimostudijní činnosti v laboratoři mechatroniky a informace které jsem získal až v průběhu. Zejména jsem prohloubil své znalosti programování vývojových desek Arduina a tvorby grafických rozhraní v Microsoft Visual Studiu. V neposlední řadě mě na této práci zaujala možnost spolupracovat na něčem, co bude mít reálné každodenní využití a případně i budoucí přínos pro ozbrojené síly České Republiky.

Řídicí software je navržen podle zadaného postupu. Hardware měřícího zařízení nebyl z mé strany nijak upravován a veškerá má práce se podřizovala limitům s tím spojeným.

V práci dále využívám část software, kterou navrhoval **Tomáš Kaňka** ve své

bakalářské práci **Řízení DC motoru na testovacím zařízení pro roznětnice**. Software, jenž využívám, slouží k samotnému řízení motoru v testovacím zařízení a jelikož jsme na zařízení pracovali současně, nebylo možné kódy od sebe plně oddělit.

Práce je rozdělena do sedmi kapitol včetně úvodu a závěru. V první kapitole jsou rozebrány cíle práce a rozvržení struktury. Ve druhé kapitole se čtenář může blíže seznámit s vývojovými deskami Arduino a některými z jejich možností. Třetí kapitola je zaměřena na analýzu dodaného systému a jsou v ní rozebrány jednotlivé součásti měřicího zařízení. V rámci čtvrté kapitoly je vysvětlena struktura navrhovaného softwaru s podrobnějším popisem některých částí kódu. V páté kapitole je věnován prostor testování navrženého systému a ukázce naměřených dat. Práci nakonec uzavírá kapitola „Závěr“.

Kapitola 1

Cíle práce

Práce se zabývá návrhem a implementací řídicího a ovládacího programu pro zařízení na testování roznětnic. Systém bude poskytovat následující funkce:

- Zajištění komunikace mezi jednotlivými částmi zařízení
  - Běžný počítač/notebook s řídicím programem
  - Multimetr HP 34401A
  - Arduino Mega2560 řídící otáčky motoru prostřednictvím zpětné vazby
- Grafické rozhraní umožňující
  - Ovládání procesu
  - Zobrazení naměřených hodnot
- Export naměřených dat v jednotném formátu

V teoretické části této práce je zařazena rešerše na téma Programovatelné desky Arduino (viz. kapitola 2). V rámci rešerše budou shrnuty základní informace o deskách Arduino, jejich historii, rozdělení, hardwaru a softwaru. Téma rešerše bylo vybráno v souvislosti s tím, že v rámci práce se využívá jedné z desek Arduino k řízení části zařízení.

Před samotným návrhem programu je potřeba analyzovat celý aparát a jeho jednotlivé části. V rámci kapitoly Analýza dodaného systému (viz. kapitola 3) budou shrnuty teoretické možnosti a omezení hardwaru spolu se stručným popisem jednotlivých částí. Trochu podrobnější pohled bude věnován možnostem dodané vývojové desky v rámci návaznosti na teoretickou část.

Na základě analytické části práce, bude v kapitole Návrh a implementace softwaru (viz. kapitola 4) navržen program splňující výše uvedené požadavky. Hotový řídicí program bude následně implementován na zařízení.

Nakonec bude program i s celým zařízením otestován.

Jelikož se pracuje na zapůjčeném zařízení, v žádném případě se nebude do hardwaru zařízení nijak zasahovat a žádné z jeho částí se nebudou nijak měnit.

Taktéž náplní práce není zabezpečení softwaru ani hardwaru proti nesprávné manipulaci nepoučenou osobou. Proto budou ošetřeny pouze běžné možnosti lidského pochybení, které do velké míry ovlivňují základní funkcionalitu. Avšak dle požadavků a vzhledem ke specifikacím měřicího procesu má být systém poloautomatický a tudíž bude vyžadovat občasný zásah poučeného operátora.

Kapitola 2

Programovatelné desky Arduino

Následující kapitola se věnuje bližšímu seznámení s programovatelnými deskami Arduino, historií jejich vzniku, základnímu rozdělení, hardwaru, softwaru a vysvětlení principu sériové komunikace.

Pro výrobu desek Arduino se využívají různé druhy mikroprocesorů. Obecně mají desky různé množství analogových a digitálních pinů pracujících v režimech input(vstup) nebo output(výstup). K základním deskám pak mohou být připojeny různá rozšíření (tzv. shieldy), která přidávají či rozšiřují funkce desky. Součástí komunikačního rozhraní základních desek dále bývá sériové komunikační rozhraní, včetně USB u některých modelů, které se mimo jiné využívají k nahrávání uživatelských programů z osobních počítačů do mikroprocesoru na desce. Mikrokontroléry na desce je možné programovat pomocí vyšších jazyků C nebo C++.

2.1 Historie Arduina

Projekt Arduino započal v roce 2005 na půdě italského Interaction Design Institute ve městě Ivrea[1][2]. V té době tamní studenti využívali desky BASIC Stamp, které nebyly levné a tudíž byly pro některé studenty hůře dostupné a Arduino tento problém vyřešilo. Po tomto úspěchu se tvůrci rozhodli nabídnout Arduino do celého světa[1].

Koncept „dnešního“ Arduina byl založen na diplomové práci Hernanda Barragána, který v roce 2003 vyvinul platformu Wiring. Platforma Wiring měla být nízkonákladový nástroj použitelný pro tvorbu jednoduchých projektů neodbornou veřejností. Platforma se skládala z desky plošných spojů(PCB) s mikroprocesorem ATmega168, integrovaného vývojového prostředí(IDE) založeného na prostředí Processing a knihoven obsahujících funkce pro snadnější programování desky[3]. Následně pak vedoucí Barragánovi práce Massimo Banzi spolu s dalším studentem Davidem Mellisem a Davidem Cuartiellesem rozšířili plat-

formu a nahradili drahý mikroprocesor ATmega168 levnějším ATmega8[2]. Avšak místo toho, aby pracovali na původním Wiring, oddělili projekt s touto změnou a pojmenovali ho Arduino. Barragán k novému projektu již přizván nebyl[3].

Odhaduje se, do roku 2011 bylo prodáno 300 000 oficiálních Arduino desek[4] a v roce 2013 bylo toto číslo již 700 000[5].

Jméno Arduino bylo vybráno podle jména baru, kde se část původního týmu zakladatelů pravidelně scházela. Zdejší bar nesl své jméno po italském králi, který se jmenoval Arduin z Ivrea, markrabě Ivrejský, a Itálii vládl v letech 1002 až 1014[6].

2.2 Rozdělení

Samotné jméno Arduino neoznačuje konkrétní desku nebo čip, je to název celé skupiny vývojových desek různých vlastností a funkcí. Jednotlivé desky pak mohou být využity pro rozdílné projekty, které mohou mít specifické požadavky na funkčnost, výkon nebo rozměry.

Na trhu se pohybuje nepřeberné množství desek Arduino a jejich klonů. Nemá smysl se zabývat výčtem jednotlivých variant, které se nemusí příliš lišit. V této práci je proto představeno pouze několik nejčastěji používaných desek s podrobnějším popisem jejich vlastností.

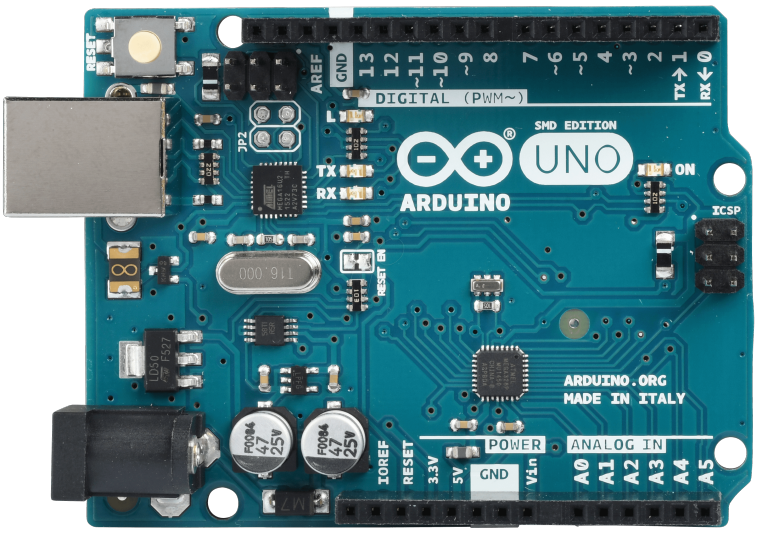
- Arduino UNO
- Arduino Nano
- Arduino Due
- Arduino Mega

2.2.1 Arduino UNO

Arduino UNO je jedna z nejrozšířenějších desek. Je vybavena procesorem ATmega328 pracujícím na frekvenci 16 MHz. Má 32 KB programové paměti, 1 KB EEPROM, 2 KB RAM. Na desku je dále vyvedeno 14 digitálních I/O pinů, 6 analogových vstupů a napěťové piny pro 5 V a 3,3 V. Uspořádání pinů na této desce umožňuje připojení většiny shieldů[1] na trhu[7][8].

Deska UNO disponuje napájecím konektorem, který umožňuje napájet desku z externího zdroje, což je potřeba pro aplikace vyžadující větší elektrický výkon [1].

[1] shield – rozšiřující modul



Obrázek 2.1: Arduino UNO (převzato [9])

2.2.2 Arduino Nano

Arduino Nano je specifické tím, že je to jedna z nejmenších desek, které se běžně používají. Díky svým rozměrům je vhodné pro využití v těch nejmenších projektech, kde je potřeba šetřit jak místem tak hmotností. Arduino Nano je jedna z nejlevnějších Arduino desek, které jsou k dispozici[1].

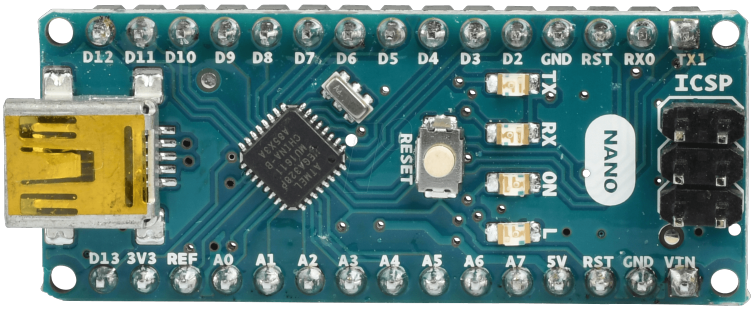
Stejně jako UNO obsahuje procesor ATmega328, s frekvencí 16 MHz, 32 KB programové paměti, 1 KB EEPROM, 2 KB RAM. Rozdíl je však u počtu pinů. Arduino Nano má 14 digitálních I/O pinů a 6 analogových vstupních pinů. Napěťové výstupy jsou také pro 5 V a 3,3 V[8].

Na rozdíl od Arduino UNO nemůže Nano kvůli rozložení pinů připojit shieldy, ale je možné ho připojit do nepájivého pole.

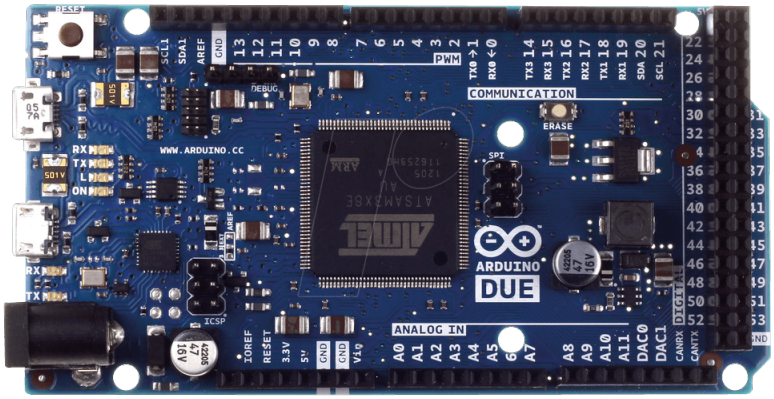
2.2.3 Arduino Due

Arduino Due je jednou z větších desek a také je první Arduino deskou s procesorem ARM. Je vhodná pro aplikace vyžadující větší výkon, či přesnější práci s analogovými signály[1].

Na rozdíl od ostatních není napájeno 5 V, ale vyžaduje pouze 3,3 V. Procesor Atmel SAM3X8E s taktovací frekvencí 84 MHz, má 512 KB ROM, 96 KB RAM, 54 digitálních I/O pinů (z toho 12 PWM kanálů), 12 analogových vstupů a 2 analogové výstupy. Due nemá žádnou paměť EEPROM a patří mezi dražší Arduino desky. Rozložení pinů je kompatibilní pro Arduino shieldy[7][8].



Obrázek 2.2: Arduino Nano (převzato [9])



Obrázek 2.3: Arduino Due (převzato [9])

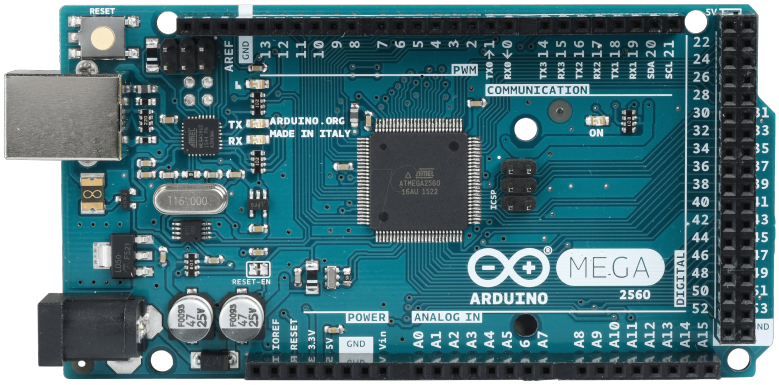
2.2.4 Arduino Mega 2560

Vzhled Arduino Mega vychází z prodlouženého designu Arduino UNO. Zvětšením desky vzniklo více prostoru pro větší a výkonnější čipy. Na desku byl tak umístěn mikročip ATmega1280 a nověji ATmega2560. Tato deska se hodí pro aplikace vyžadující větší výkon[1].

Procesor ATmega2560 pracuje při frekvenci 16 MHz, má 256 KB ROM, 8 KB RAM, 4 KB EEPROM a je napájen 5 V. Deska má 54 digitálních I/O pinů (z toho 15 PWM kanálů), 16 analogových vstupů a měla by být hardwarově kompatibilní s Arduino shieldy[7][8][10].

2.3 Hardware

Ačkoliv je hardwarový i softwarový design volně k dispozici, společnost si vyžádala exkluzivitu jména Arduino pouze pro oficiální produkty a nesmí být bez souhlasu použito pro jiné odvozené projekty. Nicméně existuje množství dal-



Obrázek 2.4: Arduino Mega (převzato [9])

ších klonových projektů, které často využívají ve svých jménech část „duino“, z čehož je patrná souvislost s původním návrhem.

Většina desek Arduino je opatřena 8-bitovým AVR mikrokontrolerem Atmel (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560)[1][10]. Jednotlivé mikrokontrolery se pak liší například množstvím paměti nebo počtem pinů[11]. Speciálním případem je pak Arduino Due uvedené na trh v roce 2012, které využívá 32-bitový Atmel SAM3X8E[1].

Na povrch desky jsou vyvedeny piny mikroprocesoru a bývají řazeny do jednoduchých nebo dvojitých řad. Piny mohou být pak realizovány buďto jako „zdířky“ (female pins) – vhodné pro připojení shieldů – a nebo jako „kolíky“ (male pins) – vhodné pro připojení například do nepájivého pole. V případě seskládání několika shieldů na sebe je možné, jednotlivé části ovládat pomocí sběrnice I^2C. Většina desek je pak vybavena 5 V lineárním stabilizátorem a 16 MHz piezokrystalovým oscilátorem[1][12].

Na mikrokontrolery Arduino bývají většinou již z výroby nainstalované bootloadery[2], které umožňují jednodušší nahrávání uživatelských programů do flash paměti čipu[13]. Bootloadery jednotlivých desek se samozřejmě liší, ovšem pokud programátor používá program Arduino IDE (vývojové prostředí Arduina viz. 2.4.1) nemusí tuto skutečnost nijak zvlášť řešit. Programy pro desku se běžně nahrávají ve formě neformátovaných souborů v hexadecimálním kódu pomocí sériové linky, kterou je deska připojena k počítači. V současné době je nejčastější připojení přes USB kabel a emulovanou sériovou linku, ale je samozřejmě možné využít i jiné metody, například Bluetooth.

[2]bootloader – zavaděč

2.3.1 Klony

Na trhu existuje velké množství desek, jejichž návrh je založen na Arduinu, ale o Arduino se nejedná. Tyto desky je možné označovat jako klony. Mnoho klonových desek je kompatibilních s Arduinem, a některé jsou dokonce funkcionálně plně zaměnitelné s originálními deskami. Jiné klony pak mohou funkcionalitu původních desek dokonce i vylepšovat. Mohou být vylepšeny jinými mikročipy, změněným zapojením nebo přidáním výstupních driverů[3], které například umožňují snazší řízení motoru. Často jsou pak využívány pro vzdělávací účely nebo v rámci široké komunity domácích „kutilů a bastlířů". Kompatibilita s Arduino shieldy je také závislá na vlastnostech jednotlivých desek[1].

2.3.2 Shieldy

Shieldy jsou desky plošných spojů, které je možné připojit k deskám Arduino. Tyto desky se využívají pro rozšíření funkcionality základních desek. Běžně používané shieldy poskytují například řízení motorů 3D tiskárny, rozhraní GPS, rozhraní pro připojení kabelu Ethernet, modul pro displej LCD[4] nebo přídavné nepájivé pole[1][11].

2.4 Software

Programování softwaru pro Arduino hardware je možné v jakémkoliv programovacím jazyce s kompilátorem, který je schopný vytvořit binární strojový kód pro konkrétní procesor. Firma Atmel nabízí ke svým AVR a ARM procesorům vývojová prostředí AVR Studio a Atmel Studio[1][14].

Nicméně existuje mnohem jednoduší a přívětivější prostředí pro nové uživatele. Je jím Arduino integrované vývojové prostředí neboli IDE. Arduino IDE je napsáno v jazyce Java a funguje na operačních systémech Windows, macOS a Linux[15]. Jeho struktura je založena na prostředí jazyka Processing a jazyka Wiring, na jejichž základu byl celý projekt Arduino vybudován (viz. 2.1)[1].

2.4.1 Arduino IDE

Arduino IDE obsahuje rozšířený textový editor přizpůsobený jazyku C. Například zvýrazňuje správně psanou syntaxi kódu, automaticky napovídá části kódu, páruje závorky, umožňuje vyhledávání v textu a nabízí možnost spouštění kompilace programu stisknutím jediného tlačítka. Další tlačítko pak obdobně slouží ke kompilaci a nahrání programu do čipu.

[3] driver – ovladač

[4] liquid crystal display

Dále umožňuje jednoduchý výběr čipu a portu ke kterému je čip připojen, na základě kterého již samo řeší veškeré náležitosti potřebné během kompilace a nahrávání programu. Součástí IDE je také vlastní „serial plotter a serial monitor“, tyto dvě funkce jsou velice užitečné a o jejich využití se zmíním dále v textu (podkapitoly 2.4.4 a 2.4.5).

Aplikace Arduino IDE podporuje psaní v jazyce C nebo C++ se základní knihovnou Wiring, která obsahuje mnoho základních funkcí používaných při tvorbě Arduino projektů[1]. Aplikace také umožňuje snadné stažení dalších knihoven, které může uživatel chtít začlenit do svého projektu. Díky open-source charakteru projektu Arduino je k dispozici značné množství softwarových knihoven, které ostatní vývojáři použili pro své projekty. Pro nové uživatele může pak být příjemné to, že vývojové prostředí Arduino IDE má v sobě již zahrnutou složku zdrojových kódů s příklady řešení různých problematik, která může posloužit jako nápověda nebo inspirace ve vlastních projektech.

2.4.2 Sketch

Program napsaný v Arduino IDE je „sketch“. Jednotlivé sketche se ukládají s koncovkou **.ino** v paměti počítače[16]. Nejmenší Arduino program obsahuje dvě funkce:

- setup() – tato funkce je v programu volaná pouze jednou a to hned na začátku programu. Využívá se pro inicializaci proměnných, I/O nastavení pinů a knihoven používaných v rámci programu. V jazyce C je analogická k funkci main().
- loop() – po ukončení funkce setup() je následně zahájena nekonečná smyčka volání funkce loop(). Tato funkce pak řídí desku až do jejího resetování nebo odpojení desky od napájení. V jazyce C je analogická k cyklu while(1).

2.4.3 Ukázka kódu pro rozblikání diody

Rozblikání diody patří k základním dovednostem programátora, který programuje desky Arduino a jim podobné. Na poli programovatelných desek se dá považovat za ekvivalent známého „Hello World!“[17], které je základem programování ve většině programovacích jazyků.

Desky Arduino mívají vestavěné diody LED připojené k pinu 13[18]. Této skutečnosti je možné využívat při různých aplikacích a také při testování.

Program pro rozblikání vestavěné diody se jmenuje „blink“. Po spuštění se rozbliká dioda na desce. Interval blikání je pak udržován v nekonečné smyčce. Ve zdrojovém kódu je využito třech funkcí ze základních knihoven z IDE. Jsou to funkce:

Obrázek 2.5: Prázdný sketch v Arduino IDE

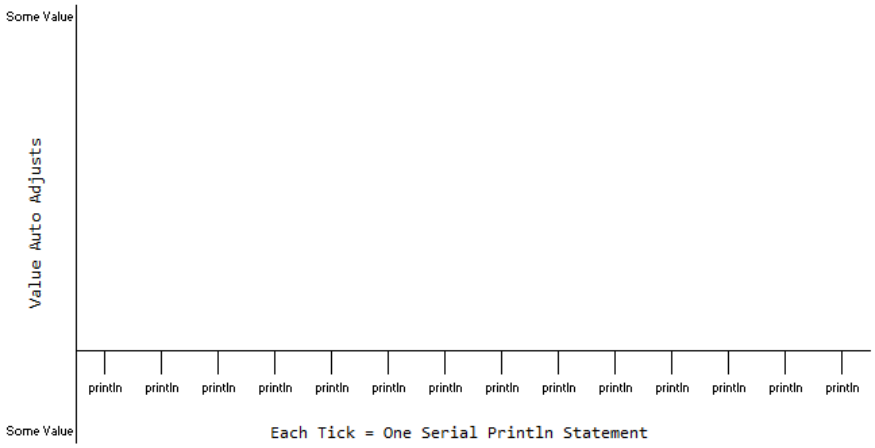
- pinMode() – nastavuje I/O charakter pinu
- digitalWrite() – zápis hodnoty HIGH/LOW na digitální pin
- delay() – čekání programu

```
#define LED_PIN 13

void setup() {
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    digitalWrite(LED_PIN, HIGH);
    delay(1000);
    digitalWrite(LED_PIN, LOW);
    delay(1000);
}
```

Kód 2.1: Blink – blikání diodou

2.4.4 Serial plotter

Serial plotter je jeden z velmi užitečných nástrojů vývojového prostředí Arduino IDE. Umožňuje zobrazovat naměřená data z Arduina do grafu přímo v pracovním prostředí. Data přijímá z desky po sériové lince a ihned je vykresluje do jednoduchého grafu. Pro správnou funkci plotteru je nutné nastavit stejný baud rate[5] pro sériovou komunikaci jak v IDE tak i na desce[19].



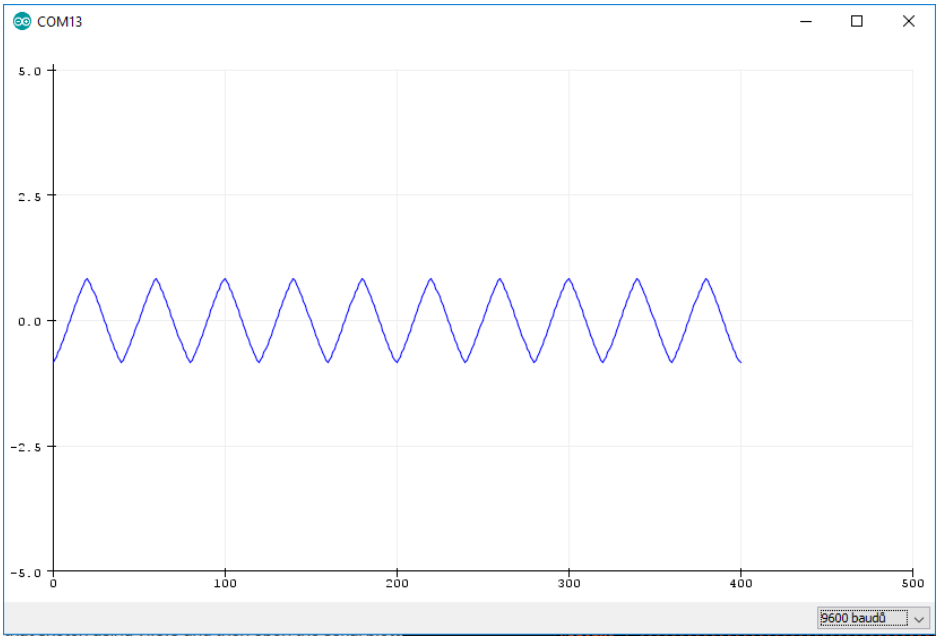
Obrázek 2.6: Serial plotter – princip vykreslování (převzato [19])

Graf plotteru se vykresluje do osy X a Y. Vertikální osa Y se sama přizpůsobuje funkční hodnotě a úměrně výstupu se pak snižuje nebo zvyšuje. Na horizontální ose X se vykreslí nová hodnota při každém vykonání příkazu println(). Okno plotteru se pak aktualizuje v momentě kdy je přidána nová hodnota. Šířka osy X je pevně vázaná na 500 bodů (hodnot, dat) a je nezávislá na šíři okna. V případě že je přidán bod 501, je současně zahozen bod 1 [19].

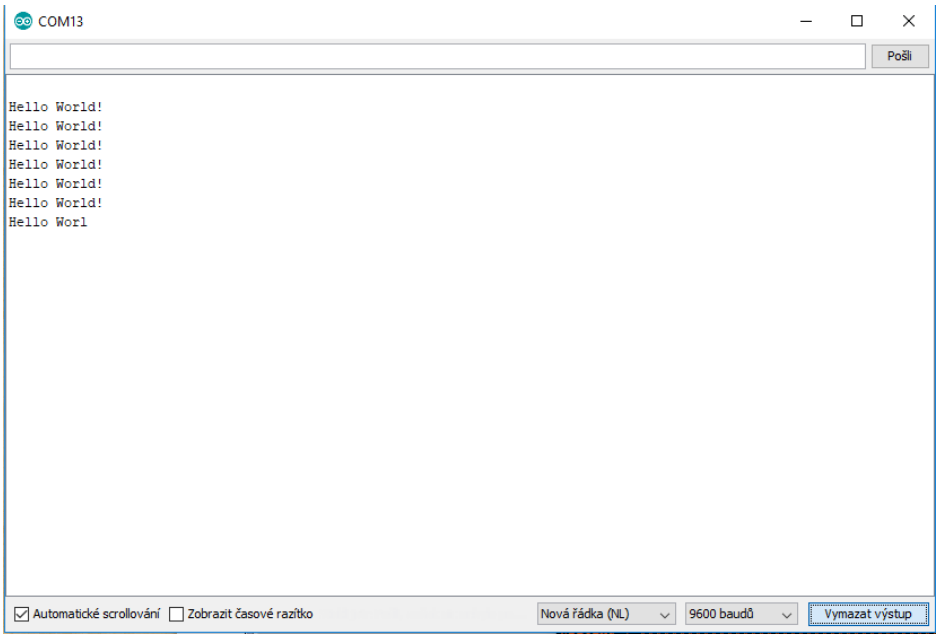
2.4.5 Serial monitor

Serial monitor je nástroj, který uživateli zobrazuje výstup na sériovou linku z desky Arduino. Okno nástroje je možné vyvolat jedním kliknutím a při nastavení stejné baud rate jako je v programu desky monitor vypisuje po řádcích informace vysílané čipem. Této skutečnosti se dá využít například pro vytvoření „externího monitoru“ desky pro sledování například změřených dat nebo zpětné vazby na ovládací veličiny.

[5] baud rate – rychlost komunikace



Obrázek 2.7: Serial plotter – ukázka vykreslení periodické funkce



Obrázek 2.8: Serial monitor

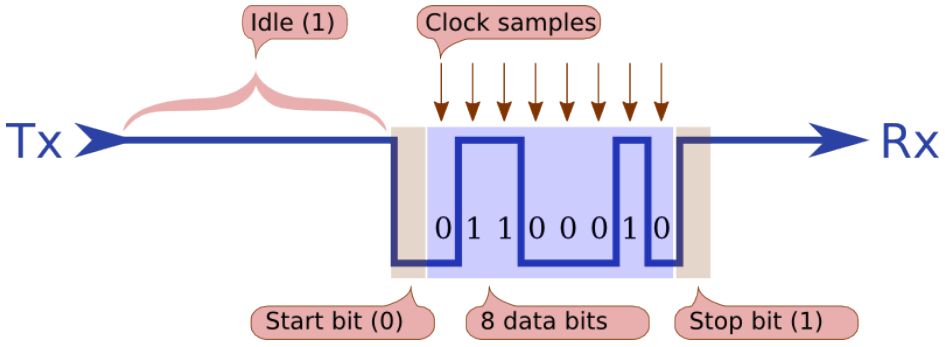
2.5 Sériová komunikace

Sériová komunikace je jednou z jednodušších metod přenosu dat. Touto metodou jsou data přenášena komunikačním kanálem postupně bit po bitu na rozdíl od paralelní komunikace, kde je vždy několik bitů přeneseno najednou v několika kanálech. Je využívána především při přenosu dat na dlouhé vzdálenosti, kvůli nižší ceně přenosového kabelu – pouze jeden datový kabel v jednom směru. V případě asynchroní sériové komunikace je další výhodou fakt, že není potřeba řešit synchronizaci přijímače s vysílačem[20][21].

2.5.1 Asynchroní sériová komunikace

Na rozdíl od SPI[6], USB[7] nebo I^{2}C[8] se u asynchroní sériové komunikace nevyužívá clock[9]. Řídící clock se nazývá baud rate a určuje kolik bitů se přenese za jednu sekundu. V případě asynchroní sériové komunikace musí mít jak vysílač tak přijímač stejnou hodnotu baud rate. V opačném případě by přijímač četl nesmyslná data. Při čtení je potřeba číst odeslané bity se stejnou frekvencí s jakou byly odeslány[21].

Asynchronost komunikace znamená, že jednotlivé bajty dat mohou být odeslány v jakémkoliv čase s libovolnými časovými rozestupy. Začátek přenášeného bytu je označen takzvaným „start bitem“, následuje 6 až 9 bitů (nejčastěji 8 bitů) nesoucích informaci bajtu a poté „stop bit“ který označuje konec bajtu[21].



Obrázek 2.9: Sériová komunikace – příklad odeslání písmene 'F' (převzato z [21])

Na obrázku 2.10 je vidět způsob jakým je přenášena informace v případě odeslání písmene 'F'. V hexadecimálním zápisu podle ASCII tabulky je písmeno F psáno jako 0x46 a jako 0b1000110 v zápisu binárním. Po sériové lince

[6] Serial Peripheral Interface
[7] Universal Serial Bus
[8] Inter-Integrated Circuit
[9] takt – signál umožňující synchronizaci jednotlivých částí digitálního obvodu

je pak z vysílače odesláno binární číslo v opačném pořadí, tedy od nejnižšího bitu (podle LSB[10]). Převrácený tvar bude tedy vypadat jako 0110001. Hardware přijímače se pak stará o to, aby bylo číslo načteno ve správném pořadí[21].

Výstupu na obrázku 2.10 na většině desek Arduino dosáhneme použitím kódu 2.2 a nastavením správných hodnot baud rate.

```
void setup()
{
    Serial.begin(9600);
    Serial.print("F");
}

void loop ()
{

}
```

Kód 2.2: Sériová linka – F

V případě, že na sériové lince není posílána žádná informace a je v nečinném stavu, je nastavená trvalá hodnota „1“ připojením „PULL UP rezistoru“. PULL UP udržuje stálou hodnotu HIGH a využívá se proto, aby byla zajištěna požadovaná „1“.

Pokud však vysílač chce odeslat bajt s informací, umístí na jeho začátek start bit, který indikuje začátek bajtu. Start bit má podobu „0“ nastavené na 1,5 násobek času přenosu jednoho bitu. Během tohoto času je přeskočen start bit a půlka prvního bitu nesoucího informaci.

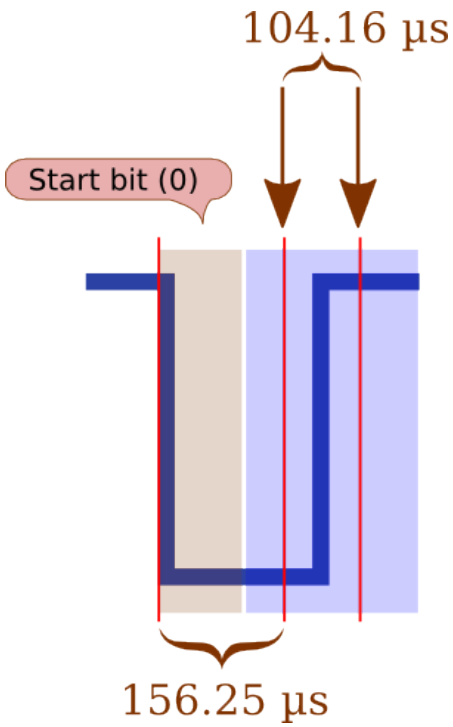
Čas přenosu jednoho bitu se snadno určí z baud rate. Jelikož je baud rate počet bitů poslaných za jednu sekundu, čas jednoho bitu je pak převrácená hodnota baud rate. Tedy pro příklad baud rate 9600, je čas potřebný pro jeden bit[11] 104,16 μs, neboť platí výpočet:

$$Sample\ rate = \frac{1}{baud\ rate} = \frac{1}{9600} = 0{,}00010416\,s = 104{,}16\,\mu s \qquad (2.1)$$

V momentě kdy přijímač načte start bit, počká 1,5 násobek vzorkovací frekvence, to je 156,25 μs pro baud rate 9600.

$$Start\ bit = 1{,}5 \times sample\ rate = 1{,}5 \times 104{,}16 = 156{,}25\,\mu s \qquad (2.2)$$

[10]LSB – Least Significant Bit neboli Nejméně Významný Bit
[11]sample rate neboli vzorkovací frekvence



Obrázek 2.10: Sériová komunikace – start bit (převzato z [21])

Po uplynutí času určeného pro start bit začne přijímač číst jednotlivé bity se vzorkovací frekvencí odpovídající nastavené baud rate. Bit je díky fázovému posunu vzniklému od start bitu načten vždy ve svém středu, tudíž je zamezeno chybám v přenosu informace, které by mohly vzniknout při přechodových jevech na okraji bitu.

Na konec posílaného bajtu zařazuje vysílač takzvaný „stop bit“, který je nastaven na hodnotu „1“ a zajišťuje aby mezi každými dvěma bajty bylo patrné oddělení. Pokud by nebyl stop bit zařazen na konec bajtu, tak v případě že by bajt končil nulovým bitem, přijímač by nebyl schopen rozeznat, zda se jedná o konec bajtu, nebo start bit bajtu následujícího. Tedy pomocí stop bitu je zajištěno, že před dalším start bitem bude vždy hodnota „1“ a přijímač tak bude schopen zaznamenat pokles vyvolaný start bitem[20][21].

Kapitola 3

Analýza dodaného systému

Tato kapitola se zabývá analýzou dodaného testovacího zařízení, jednotlivých jeho částí a periferií, a dalších použitých přístrojů, popisu testovaných roznětnic RK-1 a RKA. Také zde bude věnován prostor vysvětlení testovacího postupu, pro který je celý systém navrhován.

3.1 Kondenzátorové roznětnice

Roznětnice je přenosný zdroj elektrického proudu, sloužícího k odpalu rozbušky iniciující výbuch hlavní nálože. Obecně se využívají především pro těžbu a demoliční práce. Princip jejich funkce je poměrně jednoduchý. Hlavní součástí zařízení jsou kondenzátory. Ty mohou být nabity buď z baterií nebo například roztáčením induktoru pomocí kličky. K zahájení roznětu pak může sloužit typická páka ve tvaru písmene 'T'[22] nebo například tlačítko. Po sepnutí je pak uvolněna elektrická energie uložená v kondenzátorech v podobě elektrického proudu, která způsobí detonaci elektrické rozbušky.

Pro účely této práce byly zapůjčeny dvě kondenzátorové roznětnice RK-1 a RKA. Obě roznětnice jsou určeny pro odpalování roznětových okruhů s můstkovými detonátory. Roznětnice RKA pak navíc může měřit i odpor elektrického odpalovacího okruhu. Rozněcovadla připojované k odpalovacím okruhům mohou být zapojeny sériově nebo serioparalelně[23][24].

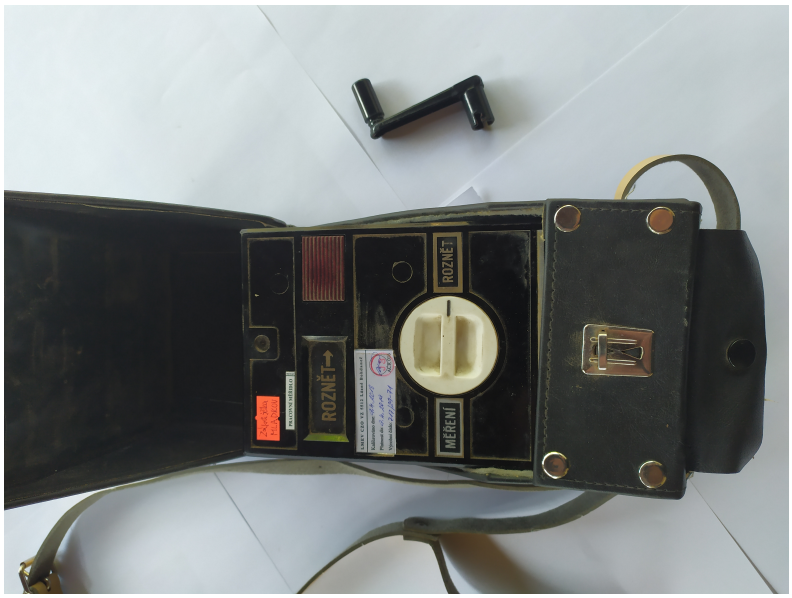
U obou zapůjčených roznětnic je pro nabití kondenzátoru používán induktor, který se roztáčí kličkou. V případě roznětnice RKA jsou k induktoru připojeny dva okruhy, první slouží k napájení kondenzátoru a druhý k měření odporu roznětové smyčky pomocí vestavěného ohmmetru[23]. Během měření odporu je současně mechanicky blokováno tlačítko roznětu a zároveň odpojeny diody od induktoru, tudíž je zamezeno nabíjení kondenzátoru. Mezi jednotlivými režimy je možné libovolně přecházet pomocí kruhového přepínače.

Obrázek 3.1: Roznětnice (převzato z [22])

3.2 Postup testovacího měření

Během testování roznětnic je potřeba dodržet kroky dodaného testovacího postupu. Jednotlivé kroky jsou pro obě roznětnice téměř stejné a liší se většinou pouze v požadovaných hodnotách testovaných veličin. Během testování se provádí následující kontroly:

Obrázek 3.2: Roznětnice RK-1



Obrázek 3.3: Roznětnice RKA

1. Napětí při 160 ot./min
2. Napětí po stisku vybíjecího tlačítka
3. Napětí za 20 s po nabití roznětnice
4. Napětí zážehu doutnavky
5. Energie roznětnice
6. Měření proudu a odporu – pouze pro roznětnici RKA
7. Závěr a exportování výsledků

V následujících podkapitolách bude podrobnější vysvětlení jednotlivých testovacích kroků. Pro zjednodušení zápisu budou postupy pro obě roznětnice sloučeny. V místech kde se požadované hodnoty liší využiji sdruženého zápisu ve tvaru „hodnota RK-1“[„hodnota RKA“].

3.2.1 Napětí při 160 ot./min

- zapnout motor
- nastavit otáčky na 160 ot/min
- změřit napětí – minimálně 1000 V – zobrazit, uložit a vyhodnotit
- vypnout motor

3.2.2 Napětí po stisku vybíjecího tlačítka

- stisknutí vybíjecího tlačítka na 5 s[2 s] – provede obsluha
- změřit napětí – maximálně 6 V[15 V] – zobrazit, uložit a vyhodnotit

3.2.3 Napětí za 20 s po nabití roznětnice

- zapnout motor
- nastavit otáčky odpovídající 1400 V
- vypnout motor a sledovat pokles napětí
- při poklesu napětí na 1260 V odpojit voltmetr a spustit časovač 20 s
- po uplynutí 20 s připojit voltmetr a ihned změřit napětí – minimálně 1100 V – zobrazit, uložit a vyhodnotit
- vybít roznětnici – provede obsluha

3.2.4 Napětí zážehu doutnavky

- zapnout motor a pomalu navyšovat otáčky – maximálně 200 ot/min
- hlídat doutnavku
- změřit napětí v momentě zážehu doutnavky – zážeh musí být nejdříve při 1000 V – zobrazit, uložit a vyhodnotit
- vybít roznětnici – provede obsluha

3.2.5 Energie roznětnice

- zapnout motor
- nastavit otáčky odpovídající 1260 V[1060 V]
- vypnout motor a sledovat pokles napětí
- při poklesu napětí na 1060 V[990 V] paralelně připojit kondenzátor
- změřit napětí a provést výpočet energie kondenzátoru podle určeného postupu – minimální energie 2 J[4 J] – zobrazit, uložit a vyhodnotit
- vybít roznětnici – provede obsluha

3.2.6 Měření proudu a odporu – pouze pro roznětnici RKA

- přepínač na roznětnici nastavit do polohy „MĚŘENÍ“ – provede obsluha
- přenastavit zkušební zařízení pro měření proudu (změna svorek na multimetru HP 34401A) – provede obsluha
- zapnout motor
- nastavit otáčky motoru na 120 ot/min
- změřit proud – maximální proud 10 mA – zobrazit, uložit a vyhodnotit
- vypnout motor
- přenastavit zkušební zařízení pro měření odporu (odpojit multimetr a připojit dekádu) – provede obsluha
- postupně nastavovat odpory dekády a zapisovat hodnoty na ohmetru roznětnice – provede obsluha
- vypnout motor

3.2.7 Závěr a exportování výsledků

Vyhodnocení naměřených hodnot, rozhodnutí o stavu roznětnice a export naměřených dat do souboru.

3.3 Multimetr Hewlett-Packard 34401A

Multimetr Hewlett-Packard 34401A je digitální multimetr, umožňující měření stejnosměrného a střídavého proudu a napětí až do 1000 V, frekvenci a odpory do 100 MΩ. Je schopný měřit rychlostí 1000 měření/s s přesností až 15 miliontin měřeného rozsahu pro měření stejnosměrného proudu a až 600 miliontin pro měření hodnot střídavého proudu. Taktéž umožňuje použití některých matematických funkcí pro zpracování naměřených dat, která jsou zároveň zobrazována na předním 6 1/2 displeji a ukládána v interní paměti. Interní paměť multimetru dokáže pojmout až 512 naměřených hodnot. Je vybaven rozhraním GPIB (více viz. 3.3.1) a RS-232 pro komunikaci s dalšími zařízeními nebo s počítačem. Multimetr je napájen ze sítě[25].

Obrázek 3.4: Multimetr Hewlett Packard 34401A – přední pohled

3.3.1 National Instruments GPIB-USB-HS

National Instruments GPIB-USB-HS je ovládací karta umožňující připojení USB portu počítače přes sběrnici IEEE 488 s měřicím multimetrem. GPIB[12]

[12]GPIB – General Purpose Interface Bus

Obrázek 3.5: Multimetr Hewlett Packard 34401A – zadní pohled

původně označovaná jako HP-IB[13] byla vyvinuta v 60 letech minulého století společností Hewlett-Packard a sloužila k propojení různých automatických měřicích přístrojů jako digitálních multimetrů a logických analyzátorů. Komunikační standard sběrnice HP-IB byl pojmenován IEEE 488[26][27].

IEEE 488 je 8-bitový standard paralelní komunikace pro krátké vzdálenosti. Ačkoliv je dnes již většinou nahrazena novými standardy v měřicí technice stále zabírá široké pole působnosti. Používá 24 pinové spojení, 16 signálových linek a 8 zemnících vratných linek. Z 16 signálových linek je pak 8 určeno pro přenos dat, 3 pro handshake[14] a 5 pro udržování komunikace na sběrnici[27].

Součástí zařízení GPIB-USB-HS jsou NI-488.2 softwarové ovladače zajišťující snadné a spolehlivé připojení k jiným přístrojům, například k osobním počítačům. Této skutečnosti bude využito v návrhové části práce[26].

3.4 Testovací zařízení s DC motorem

Testovací zařízení s DC motorem je hlavní část testovacího systému. Zkoumané roznětnice jsou připevněny k pojezdové desce a jejich induktor je propojen s hřídelí motoru. Motor zajišťuje příslušnou rychlost otáček na základě řízení z

[13]HP-IB – Hewlett-Packard Interface Bus

[14]handshake – pojem používaný v telekomunikaci pro navázání kontaktu mezi dvěma zařízeními

Obrázek 3.6: National Instruments GPIB-USB-HS

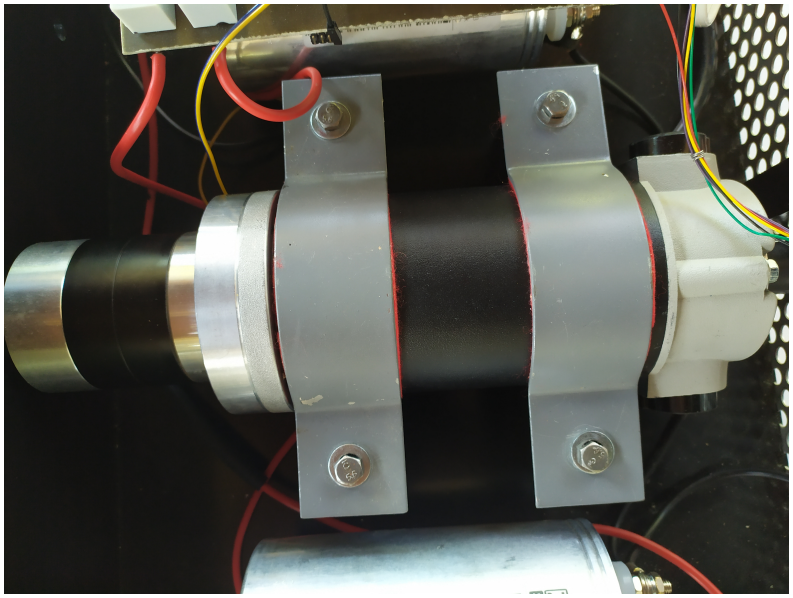
programovatelné desky Arduino Mega2560. Deska Arduino kromě řízení motoru zajišťuje i zpětnou vazbu z rotačního enkodéru motoru a fotorezistoru kontrolujícího stav doutnavky na roznětnici a také komunikaci s osobním počítačem. Celé zařízení je pak právě přes Arduino řízeno ovládacím softwarem z počítače.

3.4.1 DC motor EC180.24E

DC motor EC180.24E je jednou z nejdůležitějších částí testovacího zařízení, neboť díky němu je možné roztáčet induktor roznětnic na požadované otáčky. Při testování je hřídel induktoru spojena s hřídelí motoru podobnou mechanickou vazbou, která je použita u kličky roznětnice.

Jedná se o stejnosměrný kartáčový elektromotor s permanentními magnety. Smysl otáčení je možné řídit změnou polarity. Motor je napájen stejnosměrným napětím 24 V a proudem 10,8 A ze spínaného elektrického zdroje vestavěného do zařízení. Výkon motoru je 180 W a kroutící moment je 0,57 Nm[28][29].

Motor je schopen pracovat ve dvou režimech S1 a S2. Režim S1 je běžný pracovní režim pro dlouhodobou zátěž v trvalém provozu a jehož parametry jsou popsány v předcházejícím odstavci. Režim S2 je určen pouze pro krátkodobou zátěž, která by neměla přesáhnout dobu 30 minut, neboť poté je potřeba motor odstavit z důvodu chlazení, tak aby nebyla překročena předepsaná tepelná hranice. Mezi režimy S1 a S2 je automaticky přepnuto v momentě, kdy je překročena hodnota přiváděného proudu pro režim S1, tedy 10,8 A[28][29].



Obrázek 3.7: DC motor

Jelikož jsou otáčky motoru 3000 ot/min a požadované otáčky na induktoru se pohybují do 200 ot/min, je k motoru připojena planetová převodovka P62 s převodovým poměrem 1:14. Vstupní hřídel je uložena ve dvou řadách kuličkových ložisek, aby bylo dosaženo vyšší únosnosti v radiálním a axiálním směru[30][31].

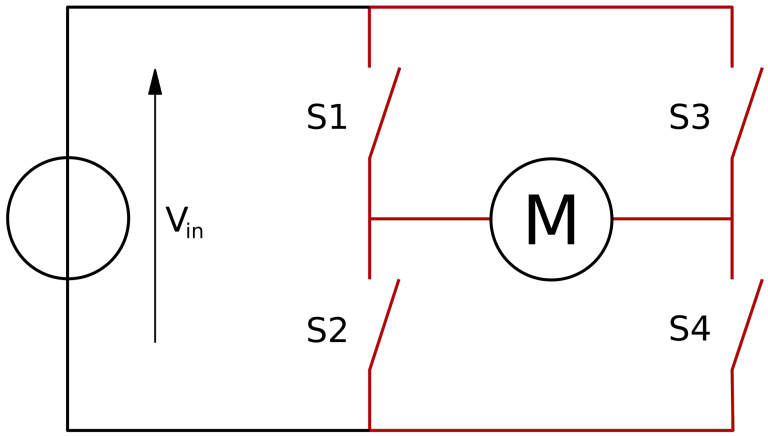
3.4.2 H-bridge

H-bridge nebo česky H-můstek je druh elektronické součástky používané pro změnu polarity napětí na stejnosměrných motorech. Změnou polarity je možné u DC motorů řídit směr jejich otáčení. Název H-můstek je odvozen ze schématu zapojení můstku, které svým tvarem připomíná velké tiskací písmeno 'H'.

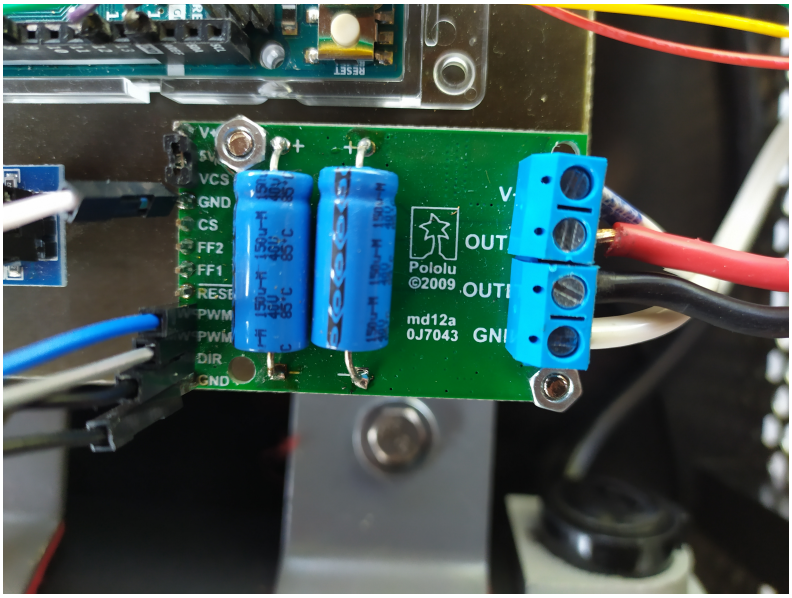
Princip jeho funkce je poměrně jednoduchý. Pomocí 4 spínačů (podle obrázku 3.8) je řízen směr průtoku proudu skrze motor. Spínače jsou sepnuty vždy po dvojicích. Pro první smysl otáčení jsou sepnuty spínače S1 a S4, zatímco S2 a S3 jsou rozpojeny. Pro druhý smysl otáčení jsou spínače nastaveny naopak, tedy S2 a S3 sepnuty a S1 s S4 rozepnuty. Zkratový stav, který je nežádoucí, nastane v případě že by byly zároveň sepnuty S1 a S2 nebo S3 a S4.

3.4.3 Elektrický spínaný LED zdroj TLPZ-24-480W

Elektrický spínaný LED zdroj TLPZ-24-480W je zdrojem stejnosměrného napětí pro vnitřní použití. Hodnota výstupního napětí je 24 V a výstupní proud



Obrázek 3.8: Schéma H-můstku (převzato z [32])



Obrázek 3.9: H-můstek

může dosahovat až 20 A. Výkon zdroje je 480 W. Zdroj je napájen střídavým napětím v rozsahu 180 V až 264 V. Systém je vybaven celou řadou bezpečnostních pojistek. Primárně je chráněn proti přehřátí, zkratu a přetížení. Součástí zdroje je vlastní systém aktivního chlazení s vestavěným ventilátorem[33].

Elektrický zdroj v zařízení slouží pouze k napájení DC motoru určeného pro roztáčení induktoru roznětnic při testovacím procesu.

3.4.4 Kondenzátory

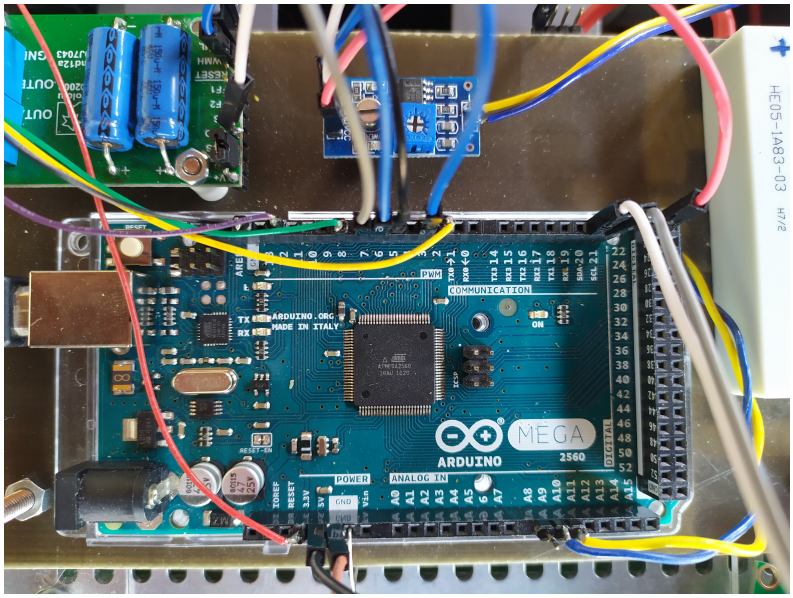
Kondenzátory mají svou úlohu v pátém kroku kontrolního postupu – výpočet energie roznětnice. Příslušný kondenzátor má být pomocí relé paralelně připojen k měřenému obvodu a podle poklesu napětí změřeném pomocí multimetru je pak vyhodnocena energie kondenzátoru v roznětnici. V zařízení bylo po-

Obrázek 3.10: Testovací kondenzátor 8 μF – PVAJP 12-3,2/8

užito kondenzátorů PVAJP 12-3,2/8 pro roznětnici RKA a PVAJP 11-3,2/4 pro roznětnici RK-1.

3.4.5 Vývojová deska Arduino Mega2560

Deska Arduino Mega2560 umožňuje řídit jednotlivé části testovacího zařízení podle instrukcí z počítače. Její primární funkcí je řízení otáček motoru, na základě zpětné vazby z enkodéru připojeného k motoru a zároveň podle instrukcí, které deska dostane z programu v počítači. Komunikaci Arduina s počítačem zajišťuje USB kabel, přes který jsou přenášena data z počítače na sériovou linku napojeni na čip Arduina. Další úkol Arduina je spínání příslušných relé pro pátý testovací krok ke zjištění energie roznětnic. V rámci čtvrtého kroku ještě načítá výstup A/D převodníku ke kterému je připojen fotorezistor kontrolující stav doutnavek. Jako poslední úkol Arduina je pak zajištění obrazového výstupu na vestavěný LCD displej, kde je potřeba zobrazit základní informace o průběhu kontrolního procesu a stavu zařízení.



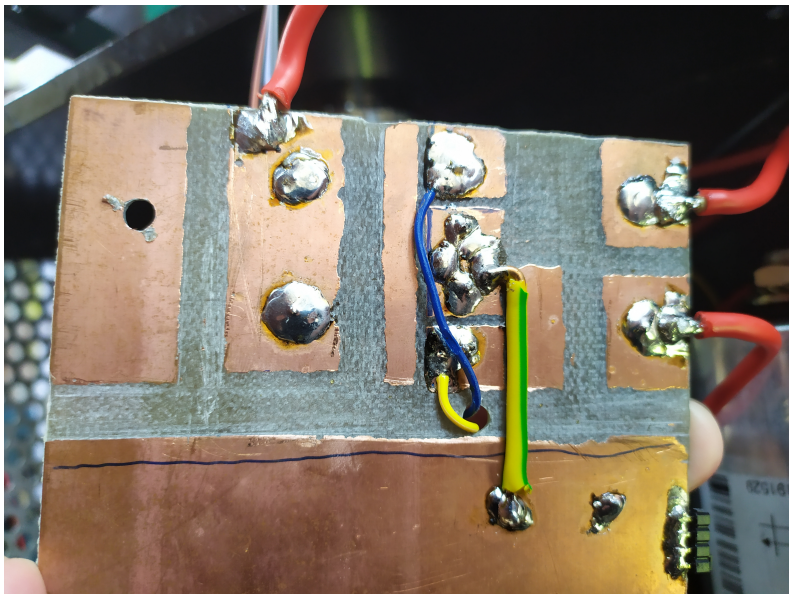
Obrázek 3.11: Vývojová deska Arduino Mega2560

3.4.6 Relé Meder HE05-1A83-03

V rámci zařízení jsou umístěny dvě relé Meder HE05-1A83-03, která slouží k paralelnímu připojení kondenzátorů k měřenému obvodu při pátém kroku. Připojením příslušného kondenzátoru se „přelije“ část energie z roznětnic a na základě změřeného poklesu napětí je možné vyhodnotit energii roznětnice. Řízení relé je zajištěno Arduinem. Relé je připájeno k plošnému spoji, ke kterému je taktéž připojen kondenzátor a ovládací výstup Arduina.

Sepnuté relé je schopno přenést proud až 5 A a napětí do 7,5 kV AC/DC, dohromady maximálně 50 W. Spínací cívka má odpor 48 Ω a je určena pro napětí 5 V DC. Výkon potřebný k sepnutí je 0,521 W. Čas pro sepnutí kontaktu je 3,2 ms[34].

Obrázek 3.12: Relé



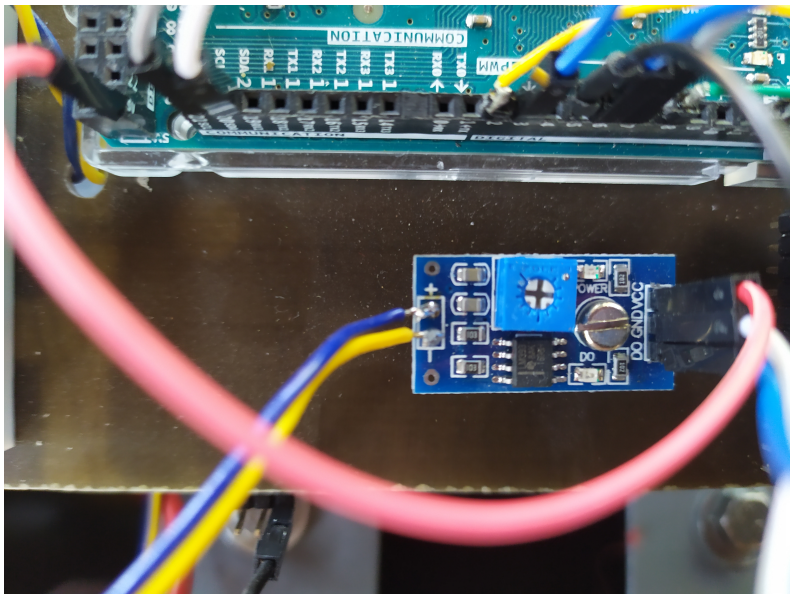
Obrázek 3.13: Plošný spoj k ovládání relé

3.4.7 LCD displej

LCD displej na zařízení měl původně zajišťovat předávání informací o zařízení uživateli. Vzhledem k tomu, že tuto funkci je možné mnohem lépe vyřešit pomocí grafického rozhraní programu, bylo by jeho začlenění nadbytečné a neefektivní. Z tohoto důvodu nebude LCD displej v rámci této práce nijak využit.

3.4.8 Fotorezistor

Fotorezistor má kontrolovat stav doutnavky, jejíž rozsvícení indikuje nabití roznětnice. Ačkoliv by bylo možné načítat analogový signál přímo z fotorezistoru, zařízení bylo opatřeno A/D převodníkem, jehož digitální výstup je připojen k Arduinu. To pak bude vyhodnocovat již převedenou hodnotu. Nastavení citlivosti A/D převodníku je řešeno manuálně obsluhou přístroje.



Obrázek 3.14: AD převodník k fotorezistoru

3.5 Problémy dodaného zařízení

V rámci analýzy byly na dodaném zařízení zjištěny některé zásadní nedostatky. V první řadě jde o zapojení obou relé, která v aktuálním zapojení nemohou být ovládána deskou Arduino. Dalším problémem je proměnlivost světelných podmínek, která může výrazně ovlivnit kontrolu doutnavky fotorezistorem.

3.5.1 Relé – přetížení obvodu

Bylo zjištěno, že relé, která měli sloužit k připojení kondenzátorů, nemohou být řízena Arduinem. V rámci analýzy této součástky byly vyhledány informace o spínací cívce relé. Cívka má odpor 48 Ω a měla být připojena k napětí 5 V (viz. podkapitola 3.5.1). Podle Ohmova zákona platí výpočet:

$$I = \frac{U}{R} = \frac{5}{48} = 0{,}10416\,A = 104{,}16\,mA \tag{3.1}$$

Tudíž proud procházející cívkou bude 104,16 mA. Výpočet byl i experimentálně ověřen pomocí vlastního multimetru UNI-T UT71B, kterým byla naměřena hodnota 98,9 mA. Bohužel výstupní piny procesoru ATmega2560 mají podle katalogového listu maximální limit 40 mA[10]. A jelikož při současném zapojení jsou spínací cívky relé připojeny přímo k těmto pinům, není možné s nimi pracovat. V případě jejich sepnutí by došlo k více než 2,5 násobnému přetížení, které by poškodilo čip.

Obrázek 3.15: Multimetru UNI-T UT71B použitý pro ověření vypočtené hodnoty

V závislosti na tomto problému, nebude možné v rámci programu realizovat pátý krok – měření energie – u obou roznětnic, dokud nebude současné zapojení opraveno. Současně s tím nemá smysl okno kalibrace kondenzátorů, které by bylo zařazeno pouze kvůli bodu 5. Nicméně v programu budou ponechány některé části kódu, které mohou umožnit snazší dodělání vynechaného pátého kroku. Případné doplnění programu už ovšem nebude náplní této práce.

3.5.2 Fotorezistor – nestálé světelné podmínky a nevhodné řešení

Ve čtvrtém kroku, kde má být kontrolována doutnavka roznětnice, je využíváno fotorezistoru, který je přes A/D převodník připojen k Arduinu. Během jednotlivých měření však můžou být různé světelné podmínky, tudíž může být potřeba přizpůsobit nastavenou hranici A/D převodníku. Vzhledem k tomu, že toho lze dosáhnout pouze mechanickým zásahem na součástce, která se nachází uvnitř zařízení, není toto řešení úplně vhodné.

Výhodnější a jednoduší by bylo, načítat přímo hodnotu analogového výstupu fotorezistoru pomocí jednoho z analogových pinů desky Arduino. Díky tomu by bylo možné kalibrovat hranici pro posouzení intenzity světla přímo v počítači skrze grafické rozhraní. To však nebude náplní této práce.

Jako částečné řešení problému bude v programu implementována možnost manuálního potvrzení rozsvícení doutnavky. V momentě kdy se rozsvítí doutnavka, obsluha stisknutím příslušného tlačítka předá programu instrukci, že doutnavka je rozsvícena.

Kapitola 4

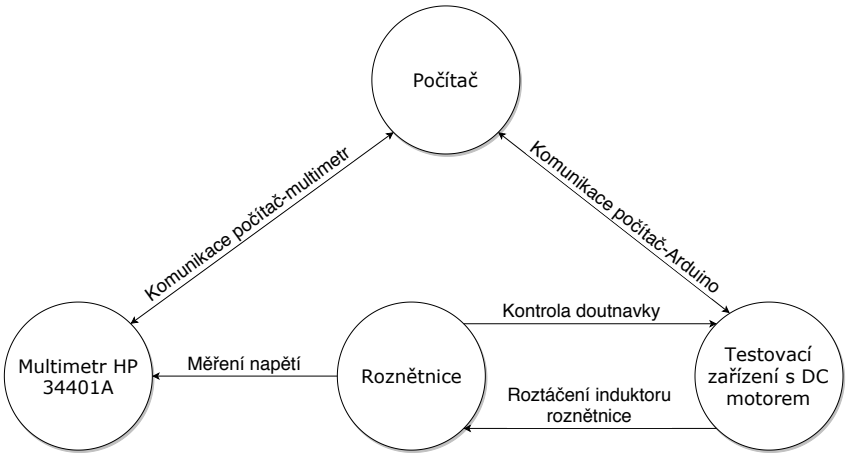
Návrh a implementace softwaru

V následující kapitole je vysvětlen návrh a implementace softwaru pro celý systém. Při psaní kódu bylo použito Microsoft Visual Studio pro hlavní program, který byl vytvořen v jazyce C# jako Windows Forms Application. Kód pro desku Arduino Mega2560 byl napsán ve vývojovém prostředí Arduino IDE.

V této chvíli je nutné znovu připomenout, že na testovacím zařízení současně pracoval i kolega Tomáš Kaňka, který se zabýval řízením DC motoru na v rámci své bakalářské práce „Řízení DC motoru na testovacím zařízení pro roznětnice“. Součástí jeho práce bylo vytvoření kódu pro Arduino k řízení otáček samotného DC motoru pomocí PID regulace. V rámci své práce proto využívám jeho kód pro řízení otáček na základě instrukcí z mého hlavního programu. Zároveň jsem však přejatý kód doplnil o vlastní komunikační protokol, tak aby bylo možné propojit počítač s Arduinem.

4.1 Hlavní struktura

Celý software se skládá z řídícího programu, který bude spuštěn na běžném počítači a programu pro desku Arduino Mega2560, který bude řídit jednotlivé úkony vlastního měřicího zařízení. Dále bylo potřeba zajistit komunikaci mezi počítačem a deskou, neboť počítačový program bude řídit celý proces a proto je nutné, aby skrze něj bylo možné Arduino ovládat, a také komunikaci mezi počítačem a multimetrem tak, aby bylo možné načítat naměřené hodnoty a dále s nimi pracovat. Hlavní program bude vybaven grafickým uživatelským rozhraním, přes které bude moci obsluha řídit jednotlivé kroky měření.



Obrázek 4.1: Návrhové schéma systému

4.2 Kód pro Arduino

Ze strany Arduina je potřeba se především postarat o respektování komunikačního protokolu používaného hlavním programem. Arduino přijímá instrukce ze sériové linky a naopak odesílá zpětnou vazbu.

Kód je sestaven tak, aby bylo možné snadno změnit počet předávaných instrukcí. Toho jsem využil především při sestavování celého systému při odstraňování chyb a ladění detailů komunikace.

4.2.1 Přijímání dat

Příchozí data jsou nejprve načtena do pole proměnných buf_receive[] (kód 4.2, řádek 11). Velikost pole je definovaná jako BUF_SIZE (kód 4.1, řádek 1) v hlavičce kódu. Načtení dat je provedeno funkcí Serial.readBytesUntil(), která načítá bajty ze sériové linky dokud nenarazí na stanovený znak, v tomto případě je to '\n'[15] a nebo načte stanovené množství znaků, které je určeno jako BUF_SIZE-1. Jakmile je splněno jedno z těchto dvou kriterií je čtení ukončeno.

[15] Znak '\n' značí „nový řádek“

```
#define BUF_SIZE 20

int input[5];
char buf_send[BUF_SIZE];
char buf_receive[BUF_SIZE];
int state = 0;
int dir = 1;

int T_RPM=100; // [ms]vzorkovani pro vypocet RPM
```

Kód 4.1: Definice základních proměnných pro sériovou komunikaci

```
void loop()
{
    if (Serial.available() > 0)
    {
        for(int i = 0; i < BUF_SIZE; i++)
        {
            buf_receive[i] = 0;
            buf_send[i] = 0;
        }

        Serial.readBytesUntil('\n', buf_receive, BUF_SIZE - 1);
        if(sscanf(buf_receive, "w %d %d %d %d", &input[0],
        ↪ &input[1], &input[2], &input[3]) == 4)
        {
            RPM_target = input[0];
            //input[1]....pouze send
            dir = input[2];
            state = input[3];
        }
    }
    doutnavka = !doutnavka_check();
    sprintf(buf_send, "%d-%d-%d-%d", (int)RPM_target, (int)RPM,
    ↪ dir,doutnavka);
}
```

Kód 4.2: Syntaktická analýza informací ze sériové linky

Následně je nutné provést na příchozích datech takzvaný parsing[16]. V praxi to znamená, že je potřeba vyfiltrovat z příchozího řetězce jednotlivé hodnoty, které Arduinu předává řídící program. Přijímané hodnoty jsou vždy celá čísla

[16]parsing – syntaktická analýza

a jsou odděleny mezerou. Díky tomu je možné snadno použít funkci sscanf() (kód 4.2, řádek 12), která z příchozího řetězce znaků vybere jednotlivá čísla a zapíše je do příslušných pozic v poli proměnných input[], které pro tento účel využívám.

Aby nedošlo k přijmutí nesmyslných nebo jinak poškozených instrukcí, funkce sscanf() kontroluje, zda-li byly úspěšně načteny 4 vstupní hodnoty, které předchází písmeno 'w'. Pokud by data přišla v jiném formátu nebo byla neúplná, nebude splněna podmínka a program nepřepíše původní hodnoty proměnných. Tímto je zajištěno, že v případě kdy nebudou dostupné nové instrukce, nebo budou špatné, program bude udržovat své poslední nastavení a zamezí tak nebezpečným poruchovým stavům.

4.2.2 Odesílání dat

Odesílání dat bylo řešeno podobně jako jejich přijímání. Nejprve je nutné jednotlivé hodnoty, jenž mají být odeslány uložit do pole buf_send[]. Zápis do pole provádí funkce sprintf() (kód 4.2, řádek 21). Následně jsou data obsažená v poli buf_send[] odeslána na sériovou linku bajt po bajtu. O odeslání se stará funkce Serial.println() (kód 4.3, řádek 13), která se nachází v bloku podmínky zajišťující časování vzorkovací frekvence.

```
1  Time1 = millis();
2
3      Time3 = Time1 -= Time2;
4
5      if (Time3 >= T_RPM)
6      {
7          Time2 = millis ();
8          int PulsSpeed = Encoder_Count;
9          Encoder_Count = 0;
10         RPM=(int)((PulsSpeed/(float)T_RPM)*600/14);
11         rpmPID.Compute();
12
13         Serial.println(buf_send);
14     }
```

Kód 4.3: Odesílání zpětné vazby ve stanovené vzorkovací frekvenci

Podmínka pro vzorkování z kódu 4.3 byla napsána Tomášem Kaňkou v rámci jeho práce. Avšak zařazení funkce Serial.println() do jeho části kódu bylo logické a jednoduché řešení, tudíž jsem nevytvářel vlastní redundantní blok programu.

4.3 Hlavní program

Jak již bylo řečeno hlavní program je vytvořen v prostředí Microsoft Visual Studia. Kód je psán v jazyce C#. Program je opatřen grafickým rozhraním a pomocí něj může uživatel ovládat celý měřicí proces. Aplikace je realizována jako několik dialogových oken, mezi kterými je možné se přesouvat pomocí příslušných tlačítek. Každé okno je určeno pro jednotlivé kroky měření, nastavení měřicího procesu nebo komunikace. Tato struktura byla zvolena na základě požadavku ze strany AČR. V průběhu měření jsou naměřené hodnoty ukládány v rámci třídy určené pro výměnu a uchovávání dat.

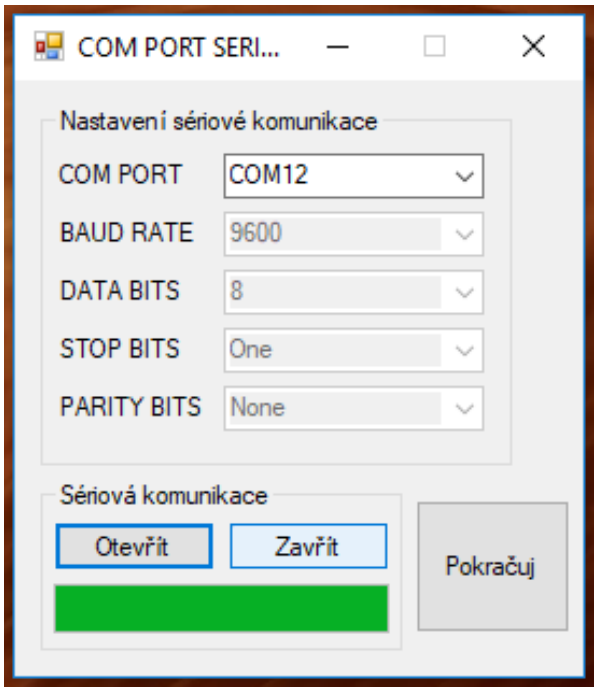
```
1  private void btnOpen_Click(object sender, EventArgs e)
2  {
3      tbox = tBoxDataIN;
4      try
5      {
6          serialPort1.PortName = cBoxCOMPORT.Text;
7          serialPort1.BaudRate = 9600;
8          serialPort1.DataBits = 8;
9          serialPort1.StopBits =
           ↪  (StopBits)Enum.Parse(typeof(StopBits), "One");
10         serialPort1.Parity = (Parity)Enum.Parse(typeof(Parity),
           ↪  "None");
11
12         serialPort1.Open();
13         progressBar1.Value = 100;
14
15         data.port = serialPort1;
16     }
17     ...
18 }
```

Kód 4.4: Nastavení parametrů sériové komunikace a volba portu

4.3.1 Nastavení sériové komunikace

Pro správné fungování sériové komunikace je nutné správně nastavit její parametry na obou stranách, tedy jak v Arduinu tak i v počítači. Komunikační rychlost baud rate je nastavena na 9600, dále bude používáno 8 datových bitů, 1 stop bit a žádný parity bit (kód 4.4). Toto nastavení je pevně dáno a nebude ho možné uživatelsky měnit. Naopak po obsluze bude vyžadováno, aby nastavila příslušný komunikační port. Toto je nutné, protože testovací zařízení může být pokaždé připojeno k různým portům. Výběr je možné provést skrze

textové okno s rolovacím seznamem dostupných portů (obrázek 4.2), které jsou zjištěny při načtení okna programu (kód 4.5).



Obrázek 4.2: Okno nastavení portu sériové komunikace

```
1  private void Form1_Load(object sender, EventArgs e)
2  {
3      string[] ports = SerialPort.GetPortNames();
4      cBoxCOMPORT.Items.AddRange(ports);
5  }
```

Kód 4.5: Zjištění dostupných portů

4.3.2 Syntaktická analýza přijatých dat

Po otevření sériové linky je opět potřeba parsovat příchozí data. V rámci hlavního programu je to řešeno pomocí metody split() (kód 4.6, řádek 9), která ukládá jednotlivé hodnoty načtené ze sériové linky do pole bufferString[]. Příchozí řetězec je nejprve kontrolován jestli splňuje požadovaný tvar a poté je teprve dělen podle znaku '-', jenž odděluje jednotlivé hodnoty.

4.3.3 Komunikace s multimetrem HP 34401A

Komunikace s multimetrem je realizována jako message-based. V programu je využíváno knihovny National Instruments VisaNS, která umožňuje snadnou

```
private void ShowData(object sender, EventArgs e)
{
    string temp = "";
    Trace.WriteLine($"{dataIN}");
    if (!dataIN.Contains('-'))
    {
        return;
    }
    var bufferString = dataIN.Split(new[] { '-' });
    if (bufferString.Length < 4)
    {
        return;
    }
    ...
}
```

Kód 4.6: Syntaktická analýza zpětné vazby z Arduina

komunikaci s multimetrem pomocí sběrnice GPIB.

V každém dialogovém okně, kde bude pro měření využit multimetr HP 34401A je potřeba inicializovat message-based komunikaci a definovat jméno proměnné symbolizující multimetr (kód 4.7).

```
private MessageBasedSession _session;

public Form_03_mereni_1()
{
    InitializeComponent();

    string resourceName = "GPIB0::22::INSTR";
    _session = new NationalInstruments.
    ↪  VisaNS.MessageBasedSession(resourceName);

    _spawnMeasure();
}
```

Kód 4.7: Inicializace message based komunikace s multimetrem

Načtení naměřené hodnoty do počítače je pak již velice snadné. Ukázka požadavku na změření a následné uložení hodnoty je vidět v kódu 4.8. První řádek odešle do multimetru správu ve specifikovaném formátu. V tomto případě se dotazuje na změření stejnosměrného napětí. V momentě, kdy multimetr obdrží požadavek, změří požadovanou hodnotu a odešle jí zpět. Druhý řádek pak

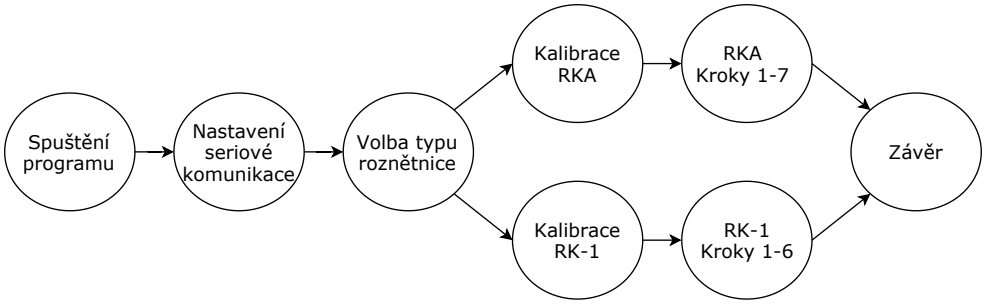
pouze načítá příchozí řetězec se změřenou hodnotou a ukládá ho do proměnné „res“. Následně je potřeba obsah proměnné podrobit syntaktické analýze, neboť příchozí data jsou ve formě textového řetězce a je potřeba je převést na číslo, které je dále využíváno.

```
1 _session.Write("MEAS:VOLT:DC?");
2 var res = _session.ReadString();
```

Kód 4.8: Načtení změřené hodnoty do proměnné v počítači

4.3.4 Princip fungování programu

Struktura hlavního programu a grafické uživatelské rozhraní je navrženo a přizpůsobeno požadavkům ze strany AČR. Celá struktura je formována tak, aby program fungoval na způsob stavového automatu. Jednotlivé stavy programu jsou reprezentovány příslušnými dialogovými okny, mezi kterými je přepínáno na základě uživatelského vstupu skrze GUI. Každý stav je programován pro specifický úkon. Jmenovitě se jedná o nastavení sériové komunikace, nastavení parametrů roznětnic a jednotlivé měřicí kroky. Grafické schéma programu je vidět na obrázku 4.3.

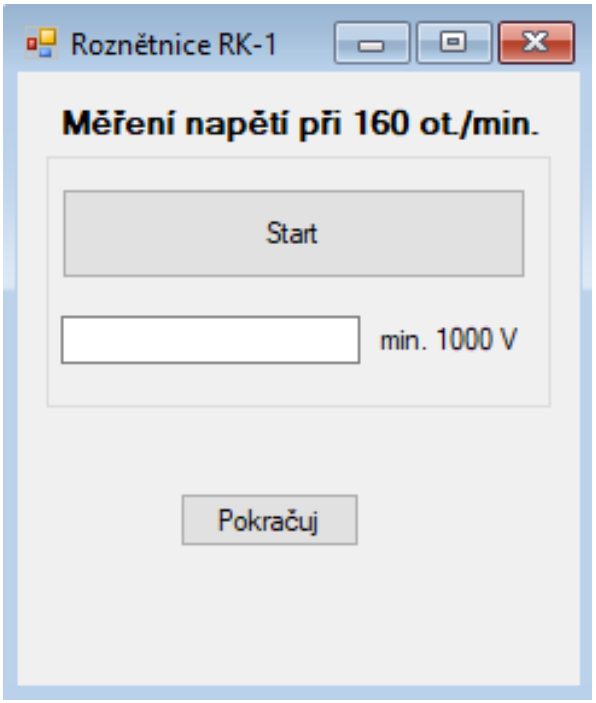


Obrázek 4.3: Pracovní schéma hlavního programu

4.3.5 Ukázka jednoho měřicího kroku

Na obrázku 4.4 je ukázka jednoho z dialogových oken. Konkrétně jde o první měřicí krok roznětnice RK-1 kde je měřeno napětí při 160 otáčkách za minutu. Po zahájení měření stisknutím tlačítka „Start“ je předána instrukce Arduinu (kód 4.9, řádek 5), které nastaví požadované otáčky. Jak je vidět na řádku 3, data jsou formátována podle předpisu sériové komunikace popsaného výše (viz. podkapitola 4.2.1).

Po odeslání požadavku je spuštěn cyklus while, jehož podmínka kontroluje zda bylo dosaženo nastavené hodnoty otáček. Pokud se tak dosud nestalo,



Obrázek 4.4: Ukázka dialogového okna pro měření prvního kroku

program počká 100 ms, a poté vyhodnocuje podmínku znovu. V případě splnění podmínky je cyklus ukončen a je nastaveno čekání dalších 2000 ms. Tím je dán prostor pro ustálení měřeného napětí.

Následně je napětí na roznětnici změřeno a uloženo do proměnné. Jelikož je příchozí údaj o napětí ve formátu řetězce a ne čísla, je provedena syntaktická analýza (kód 4.9, řádek 17) a načtené číslo je vypsáno v textovém poli v grafickém rozhraní a zároveň je uloženo do proměnné ve třídě určené pro přechování naměřených dat.

Nakonec je automaticky nastavena hodnota otáček na nulu a tato instrukce je opět předána do Arduina. Vynulováním otáček je ukončeno měření tohoto kroku. V případě potřeby je možné opětovným stisknutím tlačítka „Start“ znovu provést měřicí sekvenci.

Pokud měření proběhlo správně a uživatel potřebuje postoupit k dalšímu, stiskne tlačítko „Pokračuj“, čímž se program přesune do dalšího kroku.

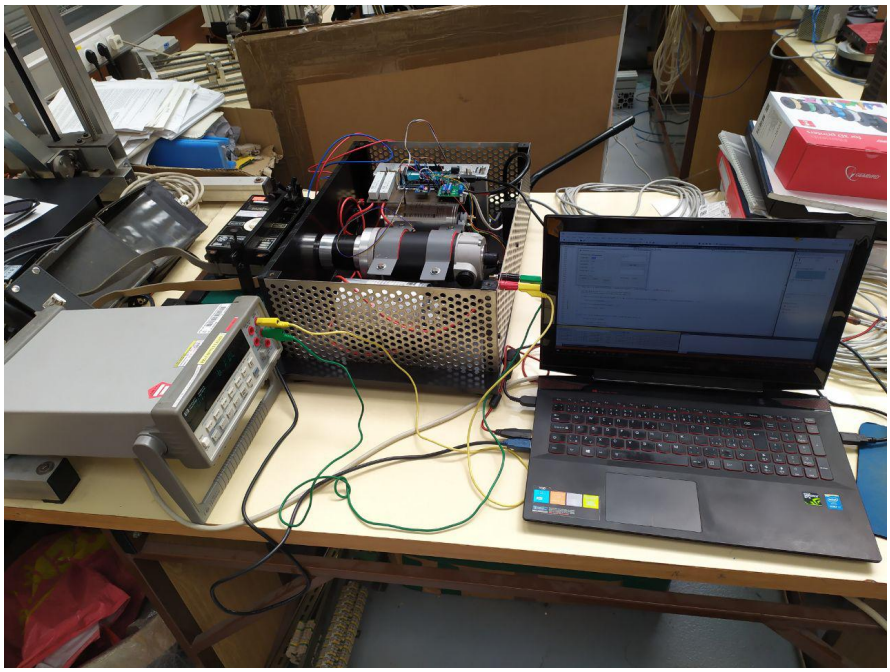
```
1  private void button_start_Click(object sender, EventArgs e)
2  {
3      string dataOUT = "w 160 0 1 0";
4      data.stav = 1;
5      data.port.Write(dataOUT);
6
7      while(!data.dostatecneOtacky)
8      {
9          Wait(100);
10     }
11
12     Wait(2000);
13
14     _session.Write("MEAS:VOLT:DC?");
15     var res = _session.ReadString();
16
17     voltage_temp = Math.Abs(double.Parse(res, new
       ↪  System.Globalization.CultureInfo("en-US")));
18     textBox_napeti_1.Text = $"{voltage_temp}";
19     data.state_1_voltage = voltage_temp;
20
21     dataOUT = "w 0 0 1 0";
22     data.port.Write(dataOUT);
23 }
```

Kód 4.9: Měření prvního kroku

Kapitola 5

Testování funkčnosti systému

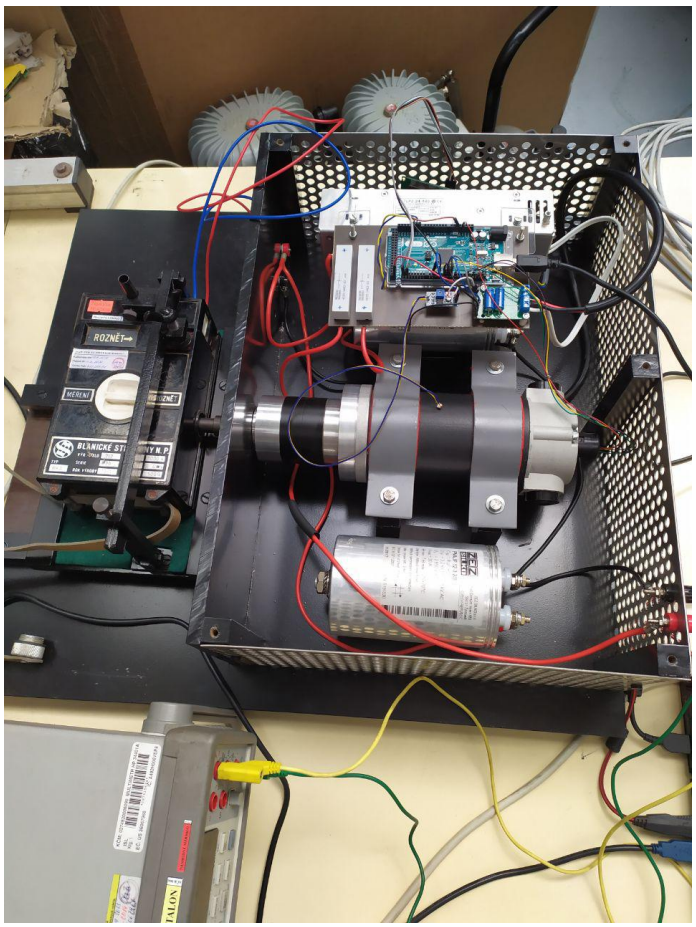
V rámci této kapitoly bude věnován prostor testování celého systému a kontrole funkčnosti implementovaného softwaru. Součástí kapitoly bude i ukázka vyhodnocení naměřených dat a náhled do souboru s exportovanými výsledky.



Obrázek 5.1: Testování funkčnosti systému

Funkčnost systému byla ověřena při zkušebním měření s roznětnicí typu RKA. Na základě zjištění problémů v analytické části práce (3.5.1), bylo z testovacího procesu vyřazeno měření energie a s tím spojená kalibrace kondenzátorů (kondenzátory mají význam pouze pro měření energie). Taktéž nebylo začle-

něno měření odporu v 6. měřicím kroku, protože to se podřizuje specifickému postupu, který je prováděn manuálně mimo automatizovaný proces.



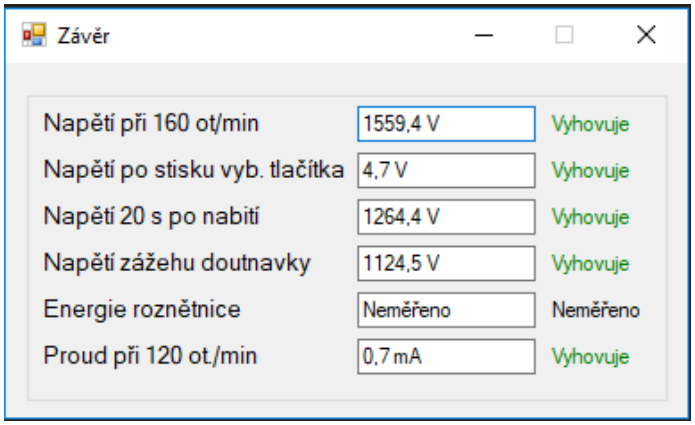
Obrázek 5.2: Testovaná roznětnice RKA připojená k testovacímu zařízení

Zapojení celého systému je zobrazeno na obrázku 5.1. Detail připojení roznětnice RKA k motoru je pak vidět na obrázku 5.2. Roznětnice je upnuta ve speciálním přípravku, který ji udržuje na místě a zároveň po celou dobu měření drží tlačítko „Roznět“ v sepnuté poloze.

Během testování nebyly zaznamenány žádné nedostatky implementovaného softwaru a celý měřicí proces proběhl v pořádku.

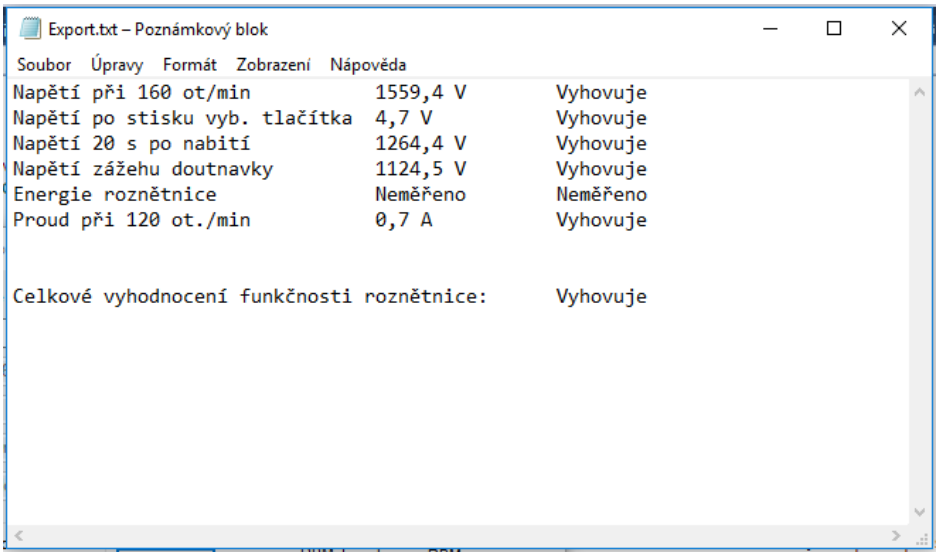
Kontrolovaná roznětnice RKA byla na základě naměřených dat vyhodnocena jako „vyhovující“, neboť všechna kritéria testovacího postupu byla splněna. Vyhodnocení výsledků měření spolu se získanými hodnotami je zobrazeno na obrázku 5.3.

Exportování naměřených a vyhodnocených dat proběhlo taktéž v pořádku.



Obrázek 5.3: Vyhodnocení naměřených dat

Data byla uložena do textového souboru „Export.txt“. Na posledním řádku souboru se ještě nachází posouzení celkového stavu roznětnice. V tomto případě byla testovaná roznětnice shledána jako vyhovující. Ukázka exportovaných výsledků je na obrázku 5.4



Obrázek 5.4: Ukázka exportu výsledků do souboru

Závěr

Cílem práce bylo implementovat software pro testovací zařízení roznětnic dodané ze strany AČR. Před provedením samotné implementace bylo nejprve nutné provést analýzu zařízení a poté navrhnout strukturu programu. Samotný návrh byl v rámci možností podřízen specifickým požadavkům zadavatele. Současně byla provedena také rešerše na programovatelné desky Arduino, neboť náplň části práce byla tomuto tématu velmi blízká.

Výstupem práce je funkční program s grafickým rozhraním, kterým je možné řídit kontrolní proces při testování roznětnic. Program je v současném stavu plně funkční, což bylo ověřeno v rámci testování funkčnosti navrženého systému. Nicméně nebylo možné vytvořit testovací systém v plném rozsahu, z důvodu objevených nedostatků na dodaném zařízení.

V rámci práce bylo nutné pracovat současně ve vývojovém prostředí Arduino IDE a Microsoft Visual Studiu. V obou případech jsem se setkal s novou pro mě dosud neznámou problematikou a bylo pro mě výzvou ji nastudovat a vyřešit. Získal jsem tak mnoho nových poznatků, které doufám využiji pro další práci a studium. Zajímavou součástí mojí práce bylo také to, že jsem musel navázat na bakalářskou práci Tomáše Kaňky, který vytvořil software pro řízení otáček motoru, jenž já ve své práci využívám.

Práce se bude určitě dále rozšiřovat. Po napravení nedostatků v hardwaru bude možné dokončit vynechané části programu, které v současném stavu nebylo možné implementovat. Jmenovitě jde především o opravu zapojení relé a fotorezistoru. Dále bych se rád zaměřil na ošetření možných chyb, které by mohly vzniknout neodbornou manipulací s programem. Také bych se chtěl věnovat zdokonalení výstupního formátu exportovaných dat, avšak nejprve bude nutné projednat bližší podrobnosti se zadavatelem.

Věřím, že po doladění nedostatků v hardwaru a doplnění s tím spojených částí softwaru, bude nakonec finální produkt zaveden do praxe.

Literatura

1. VODA, Zbyšek. *Průvodce světem Arduina*. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

2. KUSHNER, David. *The Making of Arduino* [online]. 2011 [cit. 2019-07-15]. Dostupné z: https://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino.

3. BARRAGÁN, Hernando. *The Untold History of Arduino* [online]. 2016 [cit. 2019-07-15]. Dostupné z: https://arduinohistory.github.io.

4. *How many Arduinos are "in the wild?" About 300,000* [online]. Adafruit Industries, 2011 [cit. 2019-07-15]. Dostupné z: https://blog.adafruit.com/2011/05/15/how-many-arduinos-are-in-the-wild-about-300000/.

5. *Arduino FAQ – With David Cuartielles* [online]. Malmö University, 2013 [cit. 2019-07-15]. Dostupné z: http://medea.mah.se/2013/04/arduino-faq/.

6. LAHART, Justin. *Taking an Open-Source Approach to Hardware* [online]. 2009 [cit. 2019-07-15]. Dostupné z: https://www.wsj.com/articles/SB10001424052748703499404574559960271468066.

7. MITCHELL, Robin. *A Comparison of Popular Arduino Boards* [online]. 2018 [cit. 2019-07-15]. Dostupné z: https://maker.pro/arduino/tutorial/a-comparison-of-popular-arduino-boards.

8. *Compare board specs* [online]. Arduino [cit. 2019-08-06]. Dostupné z: https://www.arduino.cc/en/products.compare.

9. *Reichelt elektronik* [online]. Reichelt elektronik GmbH & Co. KG [cit. 2019-07-16]. Dostupné z: https://m.reichelt.de.

10. *ATmega640/1280/1281/2560/2561 datasheet* [online]. 2014 [cit. 2019-08-06]. Dostupné z: https://ww1.microchip.com/downloads/en/devicedoc/

atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf.

11. *Arduino Products* [online]. Arduino [cit. 2019-08-06]. Dostupné z: https://www.arduino.cc/en/Main/Products.

12. BLUM, Jeremy. *Exploring Arduino: Tools and Techniques for Engineering Wizardry*. 1st ed. Wiley, 2013. ISBN 978-1118549360.

13. WESTFW. *Optiboot* [online] [cit. 2019-08-06]. Dostupné z: https://github.com/Optiboot/optiboot.

14. *Building an Arduino project with MegunoLink and Atmel Studio* [online] [cit. 2019-08-06]. Dostupné z: https://archive.is/20130128115914/http://www.megunolink.com/Building_an_Arduino_project_with_MegunoLink_and_Atmel_Studio_(Blink_Tutorial).

15. *Software* [online]. Arduino [cit. 2019-08-06]. Dostupné z: https://www.arduino.cc/en/main/software.

16. MONK, Simon. *Programming Arduino: Getting Started With Sketches*. 1st ed. McGraw-Hill Education TAB, 2011. ISBN 978-0071784221.

17. KERNIGHAN, Brian W.; M.RITCHIE, Dennis. *The C programming language*. 2nd ed. Prentice Hall, 1988. ISBN 978-0131103627.

18. *Blink* [online]. Arduino [cit. 2019-08-06]. Dostupné z: https://www.arduino.cc/en/Tutorial/Blink.

19. DAN. *New Arduino Serial Plotter* [online]. Rheingold Heavy, 2015 [cit. 2019-07-17]. Dostupné z: https://rheingoldheavy.com/new-arduino-serial-plotter/.

20. SWAROOP. *What is Serial Communication and How it works?* [online]. 2018 [cit. 2019-08-06]. Dostupné z: https://www.codrey.com/embedded-systems/serial-communication-basics/.

21. GAMMON, Nick. *How does serial communications work on the Arduino?* [online]. Stack Exchange, 2016 [cit. 2019-07-18]. Dostupné z: https://arduino.stackexchange.com/questions/19756/how-does-serial-communications-work-on-the-arduino.

22. SAUBER, Wolfgang. *Museum of Mining and Gothic art in Leogang (Salzburg state): Blasting machine* [online]. 2010 [cit. 2019-07-21]. Dostupné z: https://commons.wikimedia.org/wiki/File:BGML_-_Z%C3%BCndmaschinen.jpg.

23. DOUBRAVA, Lukáš. *Kondenzátorová roznětnice RKA* [online] [cit. 2019-07-21]. Dostupné z: http://www.csla.cz/vyzbroj/zenijniprostredky/rka.htm.

24. *Trhací práce* [online] [cit. 2019-07-21]. Dostupné z: http://podzemi.solvayo%20vylomy.cz/prirucka/technika/odstrely.htm.

25. *HP 34401A Multimeter User´s Guide* [online]. 1996 [cit. 2019-07-21]. Dostupné z: http://ecee.colorado.edu/~mathys/ecen1400/pdf/references/HP34401A_BenchtopMultimeter.pdf.

26. *GPIB-USB-HS* [online]. National Instruments [cit. 2019-07-21]. Dostupné z: https://www.ni.com/en-us/support/model.gpib-usb-hs.html.

27. *GPIB Hardware Installation Guide and Specifications* [online]. 2015 [cit. 2019-07-21]. Dostupné z: http://www.ni.com/pdf/manuals/370426r.pdf.

28. *DC motor - řada EC* [online]. Raveo [cit. 2019-07-25]. Dostupné z: https://www.raveo.cz/stejnosmerny-motor-EC.

29. *DC Electric motors* [online] [cit. 2019-07-21]. Dostupné z: https://www.raveo.cz/sites/default/files/download/2019/05/ec.pdf.

30. *Planetová převodovka Transtecno - řada P* [online]. Raveo [cit. 2019-07-25]. Dostupné z: https://www.raveo.cz/planetova-prevodovka-p.

31. *Planetary gear units - type P* [online] [cit. 2019-07-21]. Dostupné z: https://www.raveo.cz/sites/default/files/download/2019/02/transtecno_p_planetary_gear_units_2019_en_v0119a.pdf.

32. BUTTAY, Cyril. *Structure of an H bridge* [online]. 2006 [cit. 2019-07-25]. Dostupné z: https://en.wikipedia.org/wiki/H_bridge%5C#/media/File:H_bridge.svg.

33. *Katalogovy list LED zdroj TLPZ-24-480* [online] [cit. 2019-07-21]. Dostupné z: https://tled-web.mywac.cz/download.asp?appid=1%5C&oid=A20000000EFC.

34. *MEDER HE05-1A83-03 Relay* [online] [cit. 2019-07-21]. Dostupné z: https://www.tme.eu/%20Document/8353fdf9f0fe6c98706b5039314cff04/HE05-1A83-03.pdf.

Dodatek A

Seznam použitých zkratek

AC Alternating Current

AČR Armáda České republiky

DC Direct Current

EEPROM Electrically Erasable Programmable Read-Only Memory

GPIB General Purpose Interface Bus

GUI Graphical User Interface

HP Hewlett-Packard

HP-IB Hewlett-Packard Interface Bus

IDE Integrated Development Environment

I^2C Inter-Integrated Circuit

LCD Liquid-Crystal dDsplay

LED Light-Emitting Diode

LSB Least Significant Bit

NI National Instruments

PCB Printed Circuit Board

PWM Pulse Width Modulation

RAM Random Access Memory

SW Software

SPI Serial Peripheral Interface

USB Universal Serial Bus

Dodatek B

Obsah přiložené paměťové karty

- `ArduinoComm.rar` ..... projekt z MS Visual Studia se zdrojovými kódy
- `blink`
  - `blink.ino` ..... zdrojový kód pro rozblikání diody
- `DC_roznetnice`
  - `DC_roznetnice.ino` ..... zdrojový kód Arduino Mega2560
- `program_testovani_roznetnic.rar` ..... archiv s instalačním balíčkem programu
- `serial_monitor`
  - `serial_monitor.ino` ..... zdrojový kód pro výpis do serial monitoru
- `serial_plotter`
  - `serial_plotter.ino` ..... zdrojový kód pro vykreslení sinusoidy