Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

ŘÍDICÍ SYSTÉM STANDU PRO TESTOVÁNÍ KVALITY ŠÍPŮ

CONTROL SYSTEM FOR ARROW QUALITY TESTING STAND

AUTOR VÍTĚZSLAV ŠIMÁK

VEDOUCÍ PRÁCE Ing. RADEK ŠKARDA

PLZEŇ, 2016

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Vítězslav ŠIMÁK**
Osobní číslo:	**A12B0637P**
Studijní program:	**B3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Název tématu:	**Řídicí systém standu pro testování kvality šípů**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování:

1. a) Seznamte se se základy práce s minipočítačem RaspberryPi a prototypovou deskou Arduino.
   b) Seznamte se se základy práce s řídicím systémem REX včetně tvorby vizualizací a základy programování v jazyce C.
   c) Seznamte se se základy teorie měření.

2. a) Vytvořte ovladač pro digitální úchylkoměr.
   b) Vytvořte řídicí systém standu pro testování kvality šípů.
   c) Vytvořte vizualizaci umožňující obsluhu standu.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **30-40 stránek A4**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Radek Škarda**
Nové technologie pro informační společnost

Datum zadání bakalářské práce: **1. listopadu 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

Doc. RNDr. Miroslav Lávička, Ph.D.
děkan

L.S.

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 1. listopadu 2014

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne _____

_____
podpis

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce, panu Ing. Radkovi Škardovi, za odborné vedení práce, trpělivost a ochotu při poskytování rad, které vedly k vypracování této práce. Dále bych chtěl poděkovat panu Ing. Ondřeji Severovi za mnohé rady při konzultacích a poděkování patří i mým rodičům za veškerou jejich podporu během studia.

Abstrakt

Tato bakalářská práce se zabývá návrhem řídicího systému pro stand pro testování kvality lukostřeleckých šípů. Do návrhu spadá nejprve výběr hardwarových komponent pro měřicí stand a realizace jejich zapojení. Následně se práce zabývá vytvořením komunikačních protokolů mezi jednotlivými komponentami, vyhodnocením získaných dat o šípu a vytvořením webové aplikace pro ovládání standu a grafické zobrazení výsledků. V poslední části jsou výstupy realizací experimentu.

Klíčová slova

lukostřelecký paradox, kvalita šípu, Raspberry Pi, Arduino, Mitutoyo, REX Controls

Abstract

This bachelors thesis deals with the design of control system for arrow quality testing stand. Within design falls choosing of hardware components and realisation of the wiring. Further it deals with creation of communication protocols between each of the components, evaluation of gathered data about the arrow and creation of web application used for control of the stand and graphical representation of results. In the last part are examples of outcomes of experiment realisations.

Keywords

archer's paradox, arrow quality, Raspberry Pi, Arduino, Mitutoyo, REX Controls

Obsah

Seznam obrázků

1 ÚVOD

Odjakživa bylo cílem usnadnit si práci vylepšováním používaných nástrojů a dnešní rozvoj počítačových technologií umožňuje usnadnění práce skoro v každém oboru. Ať se jedná o automobilový průmysl, automatizaci velkých továren nebo menší projekty specializující na požadavky jednotlivců. Jedním takovým požadavkem je i zadání od lukostřeleckého klubu na sestrojení stojanu, který by automatizoval a zjednodušil proces přeměřování rozložení tuhosti v závodních lukostřeleckých šípech, aby si klub mohl vyselektovat nejkvalitnější šípy vhodné na soutěže a sportovní klání. Tato práce se zabývá sestrojením prototypu takového přístroje a odhalením případných nedostatků v návrhu spolu s doporučením možných oprav a vylepšení.

Doposud se k měření tuhosti používá manuální tzv. spine tester. Je několik sériově vyráběných variant, ale většina lukostřelců, potažmo lukostřeleckých klubů, se uchyluje k domácké výrobě těchto testerů. V případě kutilských testerů se často používá na papíře rozkreslené měřítko, na kterém ručička propojená s prohnutím šípu pod závažím určuje tuhost daného šípu. Avšak tato metoda je poměrně nešťastná, už jen kvůli nepřesnosti měření způsobené oním rozkresleným měřítkem. Navíc nedokáže odhalit nepřesnosti, jejichž nalezení vyžaduje zadání problému, a to rozložení tuhosti v šípu a nikoliv jen samotnou tuhost, pro jejíž odhad může být tato metoda dostatečná. Sériově vyráběné testery mají úchylkometr, který měří prohnutí šípu. Použitím úchylkometru lze jednoznačně dosáhnout přesnějších hodnot a tedy alespoň rámcově odhadnout rozložení tuhosti v šípu. Nicméně rozdíl výchylky prohnutí šípu v jednotlivých směrech se pohybuje i v řádech setin milimetru a méně a to už jsou hodnoty, které nelze lidským okem ani odhadnout. Přichází proto na řadu použití digitálních úchylkometrů, ale takové testery se již v sériové výrobě nenacházejí. Digitální úchylkometry jsou jedním ze způsobů řešením tohoto problému a je to zároveň způsob, který je použit v této práci. Takový přístroj, který by splňoval požadavky této práce, není na dostupném trhu sériově vyráběn a pravděpodobně existuje jen vyráběn na zakázku pro potřeby profesionálních sportovců.

Cílem je vytvořit řídící systém, který bude získávat data z digitálního úchylkometru, ovládat krokový motor pro otáčení šípem a zpracovávat získaná data. Tyto informace dále přehledně graficky zobrazit v aplikaci dostupné z webového prohlížeče a rozlišit šípy s rovnoměrně rozloženou tuhostí od těch méně kvalitních. Tato webová aplikace bude zároveň sloužit k ovládání standu.

2 TEORETICKÁ ČÁST BAKALÁŘSKÉ PRÁCE

2.1 Teorie lukostřelby a popis problému

Popis základních termínů a problémů, kvůli kterým je tento přístroj realizován. Tato sekce tedy obsahuje určitou teorii důležitou pro pochopení motivace této práce.

2.1.1 Tuhost šípu

Tuhost šípu (používán i termín "spine")[1] je stěžejní vlastností šípu pro tuto bakalářskou práci. Ovlivňuje chování šípu při jeho vystřelení z luku (chování je popsáno v kapitole 2.1.2 - Lukostřelecký paradox). Tuhosti se rozdělují do dvou skupin, a to statická tuhost a tuhost dynamická. Dynamická tuhost popisuje jak šíp nakládá s uloženou energií z luku, kterou nabere po vystřelení, ale vzhledem k tomu, že pro její určení je až příliš mnoho faktorů a proměnných, tak se tuhost šípu primárně udává statickou variantou. Měření statické tuhosti se provádí následujícím způsobem.

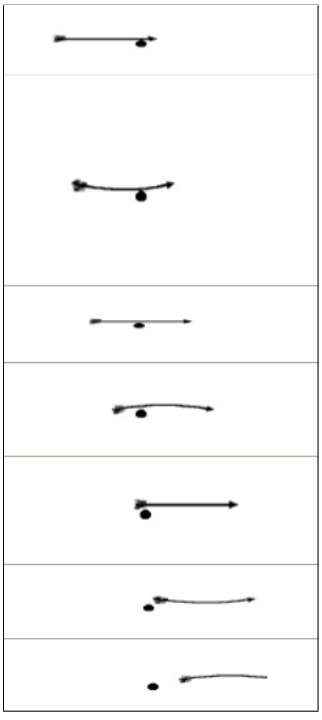
Šíp se položí na dva statické body vzdálené 28"od sebe a doprostřed se na něj zavěsí závaží o hmotnosti 1.96lb (890g). To způsobí prohnutí šípu a tuhost šípu se následně vypočte z výchylky tohoto prohnutí od klidové polohy. Tuhost je závislá na mnoha faktorech. Tím nejdůležitějším je jednoznačně materiál, ze kterého je šíp vyroben. Jsou zde brány na zřetel i pojící složky u kombinovaných materiálů (karbon/hliník apod.), geometrie šípu, vnitřní průměr i tvrdost materiálu. Neméně podstatným faktorem je i délka samotného šípu, protože delší šíp bude mít samozřejmě nižší tuhost než-li šíp kratší. Na spine má dokonce vliv i váha hrotu. Čím těžší hrot bude, tím menší bude celkový spine. Těžší hrot prakticky brání šípu v pohybu kupředu a způsobuje větší prohnutí při výstřelu.

Výběr šípu se správnou tuhostí je úzce spjatý s vlastnostmi luku a při změně luku nebo i jen jeho aspektů je třeba vždy znovu provést výběr šípů s jinými parametry. Změna šípů je důležitá zejména z hlediska přesnosti střelby. Tužší šíp letí více doleva a měkčí šíp se pak odklání na pravou stranu cíle.

2.1.2 Lukostřelecký paradox

Termín lukostřelecký paradox [2, 3] popisuje následující situaci. Při plném natažení tětivy osa šípu míří přímo na cíl. Tětiva, levý bok luku a cíl jsou v jedné rovině. V momentě, kdy je tětiva uvolněna, je na šíp aplikována tlaková síla přes končík skrz celou délku šípu. Navíc, v případě střelby třemi prsty, ještě působí síla na tětivu a končík tlačící je směrem od prstů doleva. Tětiva poté škubne zpět doprava stahujíc se zpět na střed luku. To způsobí, že hrot šípu se vychýlí doleva tím, jak je šíp tlačen proti luku. Jakmile zhruba třetina šípu projde okolo luku, šíp se prohne ještě více díky tlaku tětivy a pružnosti šípu, což způsobí, že dřík tlačí zakládku mírně doprava.

Následně se šíp odrazí od zakládky a neměl by se jí už nikdy dotknout. Setrvačnost hrotu šípu odolává pohybu doleva ale střed šípu ne. Hrot tedy pokračuje v pohybu kupředu kolem luku, což způsobuje, že šíp se ohýbá okolo zakládky. Když šíp opustí tětivu, která tlačila končík mírně doprava směrem k luku, končík zareaguje kmitnutím doleva. Na toto kmintnutí hrot zareaguje pohybem doleva spolu s končíkem a jak křidélka míjí luk, šíp je opět v rovině s osou cíle.



Obrázek 1: Průběh lukostřeleckého paradoxu

Když končík opouští luk, střed šípu se prohne doprava. Šíp se narovná a jeho střed se pak opět prohýbá doleva, ale již ne tolik jako předtím. Končík a hrot zůstávají víceméně v jedné ose, zatímco střed pokračuje v kmitání doleva a doprava, avšak s postupem času během letu se výchylka čím dál tím víc zmenšuje.

2.2 Řídicí systémy a nástroje

2.2.1 Nástroj REX Control System

Řídicí systém REX je vyvíjen společností REX Controls, která se zabývá výzkumem a vývojem pokročilých řídicích systémů, podpůrných a diagnostických nástrojů pro řízení strojů a procesů operujících v reálném čase. Hlavním stavebním kamenem této společnosti je REX Control System, což je nástroj pro vývoj a implementaci řídicích systémů umožňující jednoduchou integraci nových algoritmů a komponent od různých výrobců. Nejvyššího poměru "cena/výkon"u tohoto nástroje se dosahuje v případě řízení a diagnostiky prototypů a výzkumných přístrojů.

Hlavními rysy systému REX jsou grafické programování bez nutnosti ručního kódování, uživatelské rozhraní pro stolní počítač, tablet i smartphone, široká škála podporovaných zařízení a vstupně výstupních jednotek. Dalšími rysy jsou i průmyslově ověřené řídící algoritmy a jednoduché začlenění do podnikových IT infrastruktur.

REX Control System nabízí grafické vývojové prostředí, ve kterém se algoritmy vytváří pomocí skládání takzvaných funkčních bloků. Knihovna funkčních bloků obsahuje nespočet položek, včetně časovačů, komparátorů, filtrů a mnoho dalších.

Komponentami tohoto nástroje jsou RexDraw, RexComp, RexCore a RexView. RexDraw je již zmiňované vývojové prostředí. Široká škála cílových platforem zahrnuje i Linux, IPC, WinPac, Raspberry Pi a další. RexComp je kompilátor, převádějící algoritmy složené z funkčních bloků na binární kód. Kompilace je volána z prostředí RexDraw. RexCore jádro beží, narozdíl od doposud zmíněných komponent, na cílovém zařízení a stará se o časování a exekuci algoritmu. Jednotlivé úlohy jsou spouštěny na základě preemptivního multitaskingu. RexView je diagnostický nástroj, přes který lze sledovat jádro provádějící algoritmus. Získají se jím detailní a hierarchické informace o běžícím řídicím algoritmu.

2.2.2 REXduino

Deska Raspberry Pi a deska Arduino jsou propojeny protokolem zvaným REXduino [4], který je vyvíjen na Západočeské Univerzitě v Plzni a je to open-source projekt. Je to ideální protokol pro spojení REX Control System a desky Arduino a celá komunikace je založena na master/slave hierarchii. Arduino v tomto případě slouží jako vstupně výstupní jednotka a je připojeno k řídicímu jádru systému REX přes USB. Veškeré programování se se provádí provádí pomocí funkčních bloků, takže je poměrně snadné vytvářet i složité řídící systémy.

2.3 Použité komponenty

2.3.1 Mikrometr Mitutoyo

V této práci je použit úchylkometr od společnosti Mitutoyo, která se zabývá výrobou měřicích přístrojů i vývojem systémů na zpracování obrazu. Konkrétně je zde k měření používán model 543-390B. Mikrometr je vybaven LCD displejem s rozmezím 12.7 cm a možností resetování nulové pozice a meřením do záporných hodnot. Je schopen vydržet na jednu baterii 7000 hodin nepřetržitého provozu. Samotné navržení protokolu pro přijímání dat je řešeno v praktické části.

Obrázek 2: Mikrometr Mitutoyo

2.3.2 Krokový motor 28BYJ-48

Krokový motor 28BYJ-48 je běžně používán s jednotkami Arduino už jen kvůli tomu, že je to malý převodovaný motůrek, kterému k napájení stačí 5V, kterými Aduino disponuje. Tento druh motoru je často využíván například u automatických clonítek a klimatizačních jednotek. Převodový poměr je specifikován na 64:1 a zvládá 15 otáček za minutu. Motor je dodáván s vlastní řídicí elektronikou. Hřídel se roztáčí pouštěním proudu ve správné sekvenci skrze 4 cívky uvnitř motoru. Hřídel má na sobě uvnitř sérii magnetů a právě pouštěním proudu do zmíněných elektromagnetů jsou tyto magnety přitahovány či odpuzovány, což má za následek roztáčení hřídele. 28BYJ-48 je unipolární, takže operuje jen v kladném rozmezí napětí 0-5V. Schéma sekvence a používání motoru je popsáno v praktické části.

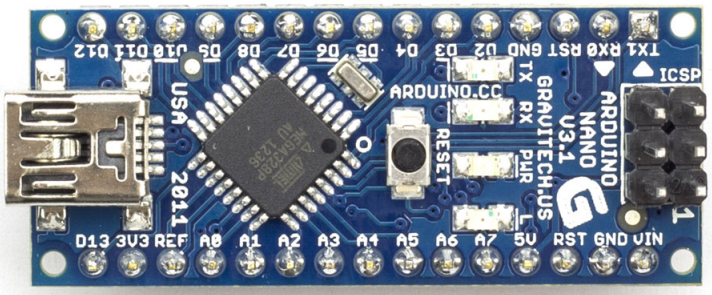
Obrázek 3: Krokový motor

2.3.3 Arduino

Arduino [6] je open-source platforma zakládající si na snadno používatelném hardwaru a softwaru. K programování Arduino mikrokontrolerů se používá jazyk specificky navržený pro tuto platformu, jehož syntaxe si bere inspiraci z jazyků C a Wiring. Samotné programování se pak provádí v prostředí Arduino IDE, které se podobá prostředí Processing.

Jednotka Arduino se skládá z 8-bitového, 16-bitového nebo 32-bitového AVR mikročipu společnosti ATMEL, krystalického oscilátoru a doplňkových komponent, které slouží k začlenění do obvodů. Od roku 2015 vyrábí Arduino i jiné společnosti, které využívají jiné mikročipy a místo krystalického oscilátoru někdy použijí keramický rezonátor, nicméně hlavní znaky těchto jednotek přetrvávají.

Takovým hlavním znakem desek Arduino je série I/O pinů pro připojení obvodů. Vlajkový produkt Arduino Uno má 14 I/O pinů. Z těchto pinů je 6 schopno produkovat



Obrázek 4: Arduino Nano

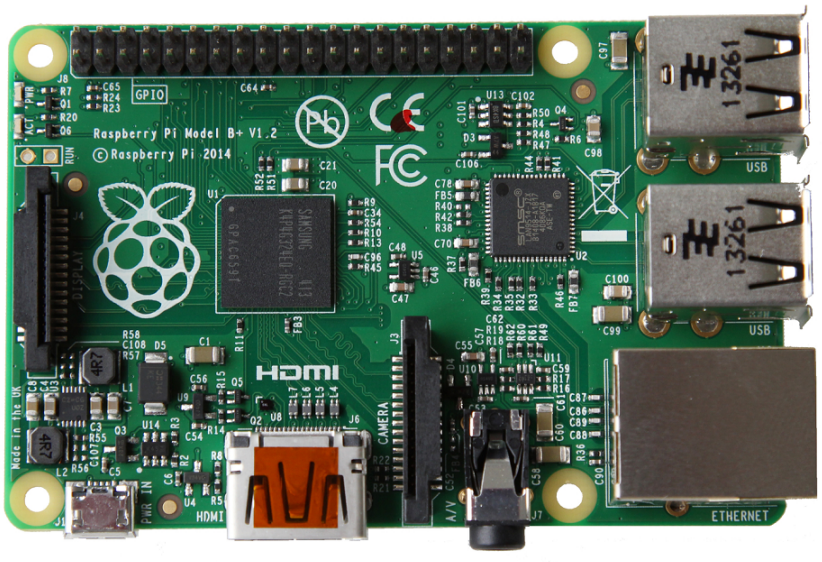
signál pulzně šířkové modulace. Dalších šest pinů je analogových. V případě Una jsou všechny tyto piny na vrchu desky a hojně se k této desce používají takzvané shieldy. Shieldy jsou desky se specifickým účelem, které jsou navrženy tak, že se snadno připojí na desku Arduina bez potřeby kabelů s headery či pájení. Patří mezi ně například Xbee shield, který zajišťuje bezdrátovou komunikace mezi Arduiny, nebo Motor Control shield, jenž se využívá pro ovládání motorů. Naopak deska Nano má piny na spodu desky a je proto snadno zapojitelná do nepájivých kontaktních polí.

První Arduino bylo uvedeno v roce 2005 na základě závěrečné práce jednoho italského studenta. Účelem bylo vytvořit levný a snadný způsob jak vytvořit zařízení, které by ovládalo aktuátory, sbíralo data ze senzorů apod. Běžně je Arduino využíváno nadšenci pro sestavení jednoduchých robotů, termostatů a dnes již i jako tzv. wearable technologie, nicméně se velmi uplatňuje i ve školách pro výukové účely.

2.3.4 Raspberry Pi

Raspberry Pi [7] je minipočítač tradičně velikosti kreditní karty vyvinutý ve Spojeném království společností Raspberry Pi Foundation. Primárně byl tento počítač vyvíjen za účelem výuky základních znalostí o počítačích ve školách a rozvojových zemích. První generace byla představena v únoru 2012 a od té doby se výčet navýšil již na osm modelů, z nichž posledním je Raspberry Pi 3 model B. Cena se většinou pohybuje mezi 20 až 35 americkými dolary, avšak v listopadu 2015 byl uveden model Pi Zero, který je doposud nejmenší variantou a vyjde na pouhých 5 amerických dolarů.

Primárně se na Raspberry Pi používají systémy založené na Linuxovém jádře. Tím hlavním, přímo pro tyto desky vyvíjeným, je Raspbian, který je nadstavbou systému Debian. Na modelu Raspberry Pi 2 může být instalována i aktuální verze Ubuntu, ale i speciální upravenou variantu systému Windows 10 označovanou IoT Core. Desky Raspberry Pi nemají žádnou interní paměť, takže data a operační systém jsou uloženy na pamětové kartě. V případě starších generací se používají SDHC karty a v případě novějších modelů karty MicroSDHC. Na kartu se nahraje obraz samotného systému nebo lze využít instalátor NOOBS, ve kterém je připraveno několik operačních systémů včetně zmiňovaného Raspbianu. Pro Raspberry Pi je dostupný i řídící systém REX od společnosti REX Controls, který nabízí možnost udělat z desky řídící jednotku například na regulaci vytápění či řízení solárních systémů.



Obrázek 5: Raspberry Pi B+

Deska Raspberry Pi B+ použitá v této bakalářské práci disponuje například ARM11 jednojádrovým CPU o frekvenci 700 MHz, 512MB SDRAM fyzickou pamětí, čtyřmi USB 2.0 porty, patnácti pinovým CSI konektorem pro Raspberry Pi kamery, HDMI konektorem, Broadcom VideoCore IV grafickým čipem, SDHC slotem na paměťovou kartu, 10/100 Mbit/s ethernetovým portem a v neposlední řadě čtyřiceti General Purpose Input/Output (GPIO) piny.

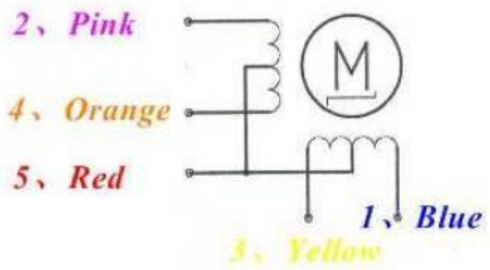
Je vyráběna řada příslušenství k těmto deskám. Mezi ty nejpoužívanější patří Raspberry Pi Camera, NoIR Camera a UniPi. Obě kamery jsou schopny snímat FullHD video (tedy 1920x1080p) při rychlosti 30 snímků za sekundu s tím že rozdílem, že NoIR kamera nemá infračervený filtr. Rozšíření UniPi disponuje vstupy s LED signalizací, 0-10V analogové vstupy i výstupy, přepínací relátka a real-time časový modul.

3 PRAKTICKÁ ČÁST BAKALÁŘSKÉ PRÁCE

3.1 Algoritmická část - komunikační protokoly a ovládání motoru

3.1.1 Práce s krokovým motorem

3.1.1.1 Fungování krokového motoru lze popsat na následujícím obrázku. Písmenem "M"je označen rotor, který je uváděn do pohybu dvěma cívkami na obrázku, které jsou rozdělené a fungují jako 4 cívky. Elektromagnety i rotor jsou ozubené a sekvenčním napájením cívek je rotor otáčen tím, že je magnetickou silou přitahován k napájené cívce tak, aby se zuby elektromagnetu a rotoru srovnali. Cívky jsou napájeny v rychlém sledu po sobě hned několika možnými způsoby, z nichž každý poskytuje jiné vlastnosti. Rozdíl je také v typu použitého motoru, jestli je použit unipolární nebo bipolární krokový motor. Pro potřeby této práce je použit unipolární krokový motor, který se od motoru bipolárního liší primárně dvěma faktory. Hlavním faktorem je používané napětí, tedy fakt, že oba druhy používají stejně rozpětí napětí, ale unipolární motor používá škálu od 0V do 5V (závislé samozřejmě na potřebném napětí) a bipolární od -2,5V do 2,5V. Druhým důležitým rozdílem je, že unipolární motor má o drátek víc, který dělí cívky napůl a umožňuje tok proudu jedním nebo druhým směrem, což má za následek rozdílné polarity. Tyto dva směry, tedy dvě polarity magnetického pole, simulují možnosti bipolárního krokového motoru.

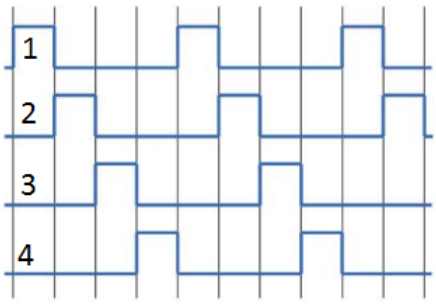


Obrázek 6: Schéma vinutí motoru

Způsoby jakými lze krokový motor ovládat jsou 4. Ve své podstatě jde vždy jen o pořadí pouštění proudu skrze jednotlivé cívky, nicméně u každé metody jsou odlišnosti v počtu kroků na otáčku a v točivém momentu.

3.1.1.2 Metody ovládání krokového motoru

- Wave drive

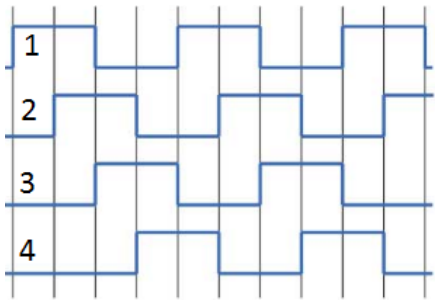


Obrázek 7: Schéma metody Wave drive

V tomto způsobu je v danou chvíli aktivována vždy jen jediná fáze. Má totožný počet kroků jako nadcházející metoda, ale nedosahuje takového točivého momentu, protože rotor je v jeden moment přitahován k jedinému elektromagnetu. Tato metoda je zřídkakdy používána.

Metoda wave drive je z pohledu této práce velmi nepraktická vzhledem k tomu, že pro otáčení zatíženým šípem je třeba velkého točivého momentu, a proto ani nebylo testována její funkčnost na standu.

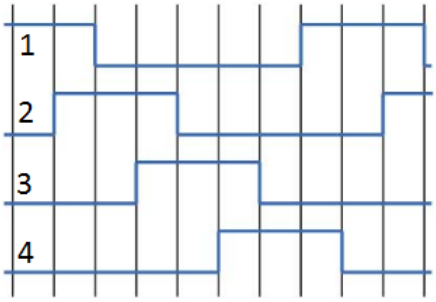
- Full step drive



Obrázek 8: Schéma metody Full step drive

Naopak metoda Full step drive je používána naprosto nejčastěji pro ovládání krokového motoru, protože poskytuje nejvyšší točivý moment. Tento způsob spočívá v tom, že v jednu chvíli jsou vždy napájeny dva elektromagnety. Takže v ten samý moment se jednou cívkou přestane pouštět proud a další cívkou v pořadí se proud pustí.

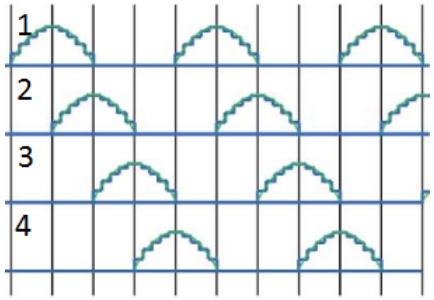
- Half step drive



Obrázek 9: Schéma metody Half step drive

Half step drive dá se říct kombinuje předchozí dvě metody. Využívá, jak sepnutí dvou cívek najednou, tak sepnutí cívky jen jedné. Má nižší točivý moment než full step, ale díky faktu, že na jedno otočení využije 8 kroků, tedy 8 pozic, a ne jen 4, tak je možné touto metodou dosáhnout dvakrát větší přesnosti.

- Microstepping



Obrázek 10: Schéma metody Microstepping

Používá se zde i termín "sine cosine microstepping"a to proto, že průběh proudu aproximativně připomíná sinovou či cosinovou funkci. Čím jsou kroky menší, tím je běh krokového motoru hladší a je eliminována výraznější rezonance v součástkách motoru.

Při návrhu řídicího systému standu byly vyzkoušeny na ovládání motoru dvě metody. Metoda "half step drive"a "full step drive". První zmiňovaná metoda sice zajišťovala více než vyhovující přenost otáčení i po zpřevodování, ale točivý moment zde byl nedostatečný. V případě zatíženého šípu se občas hnací kolo protáčelo i se silným přítlakem kola k šípu.

Proto je ve výsledném řídicím systému pro ovládání krokového motoru použita metoda "full step drive". I s větším hnacím kolem lze dosáhnout dostatečné přesnosti otáčení, ale hlavně s dostatečným přítlakem hnacího kola k šípu pro vyšší trakci dosahuje vysoké spolehlivosti.

3.1.1.3 Převodový poměr a přesnost otáčení jsou důležitými aspekty řešenými v mikrokontroléru Arduino. Vzhledem k tomu, že šípy, které se budou měřit, nemají konstantní průměr, je třeba toto kritérium zohlednit. Proto je průměr šípu jedním z parametrů zadávaných při spouštění měření. Pro tento prototyp bylo navrženo i několik variant hnacích kol. Bylo třeba najít variantu, která bude nejvýhodnějším kompromisem mezi kritériem přesnosti a kritériem spolehlivosti.

Problém je totiž ten, že větší hnací kolo zajišťuje lepší vlastnosti z pohledu spolehlivostního kritéria, ale na úkor toho, že se sníží dosažitelná přesnost otočení šípem. Veličina, pomocí které lze spočítat počet kroků nutný k otočení šípem o 360° při různých průměrech šípu a hnacího kola, se nazývá *převodový poměr*, jejíž vzorec [8] vypadá následovně.

$$i = \frac{n_1}{n_2} = \frac{d_1}{d_2} = \frac{z_1}{z_2} \tag{3.1}$$

- n_1...frekvence otáčení hnacího kola
- n_2...frekvence otáčení šípu
- d_1...průměr hnacího kola
- d_2...průměr šípu
- z_1...počet kroků motoru potřebný pro jednu otáčku hnacího kola
- z_2...počet kroků motoru potřebný pro jednu otáčku šípu

V klasicky definovaném vztahu pro převodový poměr jsou veličiny z_1 a z_2 asociovány s počtem zubů na převodových kolech. Avšak pro naše potřeby lze počet zubů nahradit počtem kroků krokového motoru pro jednu otáčku daným kolem (případně šípem), na platnosti vztahu to nic nemění.

Například pokud bychom měli hnací kolo o průměru d_1 *= 10 cm*, šíp o průměru d_1 *= 0,5 cm* a pro jednu otáčku hnacím kolem jsme potřebovali z_1 *= 1024* kroků, pak dosazením do vztahu (3.1) získáme počet kroků nutný pro jednu otáčku šípem.

$$\begin{aligned} \frac{10}{0,5} &= \frac{1024}{z_2} \\ z_2 &= \frac{1024}{20} \\ z_2 &= 51,2 \end{aligned} \tag{3.2}$$

Zjistíme, že počet kroků pro otočení šípem o 360° je $z_2 = 51{,}2$. Z této informace jsme schopni ze vztahu zjistit přesnost otočení šípem při jednom kroku krokového motoru.

$$p_d = \frac{360°}{z_2} \tag{3.3}$$

- p_d...počet stupňů o kolik se otočí šíp při jednom kroku motoru

Dosazením tedy počtu kroků na jednu otáčku šípu do vztahu (3.3) získáme p_d

$$\begin{aligned} p_d &= \frac{360°}{51,2} \\ p_d &\doteq 7° \end{aligned} \tag{3.4}$$

V tomto případě bychom docílili toho, že jedním krokem motoru otočíme šípem o 7° a to není přesnost, která by předpokladům měření vyhovovala. Jsou zvoleny 4 varianty počtů měření na jeden šíp a to 4, 8, 12 a 16 měření na jeden šíp. Tudíž potřebujeme co nejpřesněji dosáhnout hodnot 90°, 45°, 30° a 22,5°.

V případě hnacího kola o průměru 10 cm je již z předchozího úkázkového výpočtu jasné, že s přesností na 7° u šípu s průměrem 0,5 cm nelze dosáhnout ideálního otočení šípem.

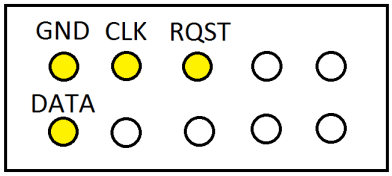
Následující tabulka obsahuje dosažitelnou přesnost otočení šípem pro jednotlivé úhly a pro rozdílné průměry šípů. Vše je počítáno s použitím hnacího kolečka o průměru 4.45 cm, které je použito ve finálním návrhu prototypu standu.

Tabulka 1: Příklad srovnání požadovaného úhlu a dosažitelného úhlu

úhel \průměr šípu [cm]	0,5	0,6	0,7
90°	90,74°	89,96°	90,52°
45°	45,37°	45,63°	44,7°
30°	29,72°	29,99°	30,17°
22,5°	21,9	22,16°	22,35°

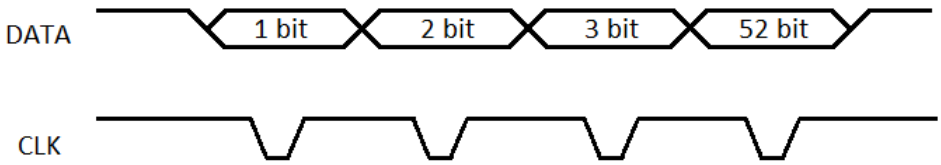
3.1.2 Zpracování dat z mikrometru

Velikost měření lze z mikrometru získat čtením jednotlivých pinů na konektoru. Na konektoru jsou celkem 4 piny, které mají nějakou funkci z nichž dva jsou výstupní, jeden je vstupní a čtvrtý slouží k uzemnění.



Obrázek 11: Schéma konektoru z mikrometru

Tím vstupním pinem je pin RQST. Když je na RQST přivedena sestupná hrana, tak se konektorem začnou odesílat data měření. Spolu s GND jsou oba piny připojeny na *zem* na desce Arduino, data měření se tedy odesílají neustále. Z pinů DATA a CLK se pak dají získávat data podle následujícího schématu [5].

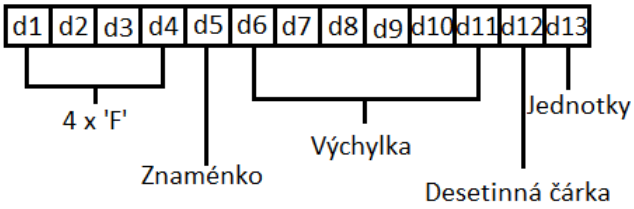


Obrázek 12: Schéma toku dat z mikrometru

Jedno měření se skládá ze třinácti bajtů z toho každý bajt je složen ze čtyř bitů. Informace je tedy dlouhá 52 bitů. Ze schématu je vidět, že CLK přepne na sestupnou a následně na vzestupnou hranu po tom, co se na DATA objeví nový bit. Jsou tedy dvě možnosti, kdy je nejvhodnější zjišťovat hodnotu, a to hned po tom co se na CLK objeví sestupná hrana a nebo ve chvíli, kdy se po sestupné objeví vzestupná. V algoritmu použitém v této práci se čeká na hranu vzestupnou, přečte se hodnota na pinu DATA a zapíše.

Podle dokumentace přiložené k mikrometru by se v případě uzemněného pinu RQST, tedy neustálého toku dat, mělo objevit měření každých 10ms - 150ms. Z heuristického hlediska se však nejvíce vyplatilo počkat se čtením nových dat 170ms.

Tímto způsobem je tedy možné přečíst a uložit 13 bajtů každých 170 ms. Těchto 13 bajtů je posláno za sebou ve formě graficky znázorněné na následujícím obrázku [5].



Obrázek 13: Tvar zprávy z mikrometru

První 4 bajty jsou dle dokumentace složené ze samých jedniček, tedy bajty s nulovou informační hodnotou. V pátém bajtu je informace o znaménku naměřené hodnoty, nula značí kladnou hodnotu a osmička značí hodnotu zápornou. V dalších 6 bajtech, tedy bajtech d6 až d11, je obsažena hodnoty výchylky jehly mikrometru od nulové pozice. V předposledním bajtu je informace o pozici desetinné čárky. V tomto bajtu se může vyskytnout buď 2, 3, 4 nebo 5 a toto číslo udává počet desetinných míst. Poslední bajt, bajt d13, v sobě již nese informaci o jednotkách, ve kterých je honota výchylky měřena. Pokud jsou to milimetry, tak obsahuje 0, a pokud jsou to palce, tak obsahuje 1.

3.1.3 Algoritmus probíhající na desce Arduino

V kódech pro desky Arduino jsou důležité dvě části. První je funkce *setup*(), která je vykonávána jen jednou, a to po nahrání programu, zapnutí desky nebo po jejím restartu. Tou druhou částí je funkce *loop*(), jejíž kód se po proběhnutí funkce *setup*() vykonává pořád dokola.

Nejdříve se do kódu přidruží potřebné knihovny, zadefinují se proměnné a konstanty. Ve funkci *setup*() se otevírá komunikace po sériové lince, naplňuje se buffer pro příchozí data samými nulami a digitální piny na nichž jsou připojeny CLK a DATA se nastavují jako vstupní piny. Po té se ještě nastaví na hodnotu HIGH, protože při nově příchozích datech je od CLK nejdříve očekávána sestupná hrana, pak náběžná a teprve po tom se čte hodnota z pinu DATA.

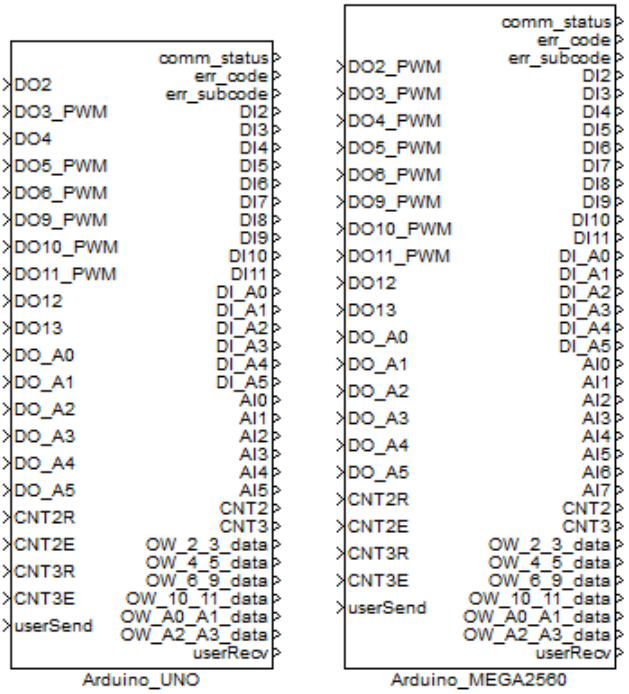
Funkce *loop*() je již část, ve které se odehrává veškeré zpracování signálu. Nejnadřazenější je v této funkci cyklus *while*, který probíhá neustále dokud není přerušen nesplněním jedné z podmínek. V tomto cyklu se čtou znaky přenesené po sériové lince a ukládají se

do bufferu. Pokud je posledním znakem znak pro další řádek, tak se přejde k vyhodnocení přijaté zprávy. Ze zprávy se dekóduje požadovaný počet měření a průměr šípu. Na základě vztahů z kapitoly 3.1.1.3 se vypočte celkový počet kroků potřebný pro otočení šípem o 360°, tato hodnota se uloží a vždy při parciálním otočením šípem mezi měřeními se počet provedených kroků odečte od celkového počtu kroků k provední. Pokud je při posledním otočení šípem požadován počet kroků větší, než počet kroků, které zbývá udělat, tak se namísto požadového počtu kroků provedou kroky zbývající. Před čtením hodnoty z mikrometru je nastavená pauza v kódu na 500 ms, aby měření nebylo ovlivněno případným zvětšením výchylky kvůli rozpohybovanému závaží. Poté se dle popsaného schématu v kapitole 3.1.2 zpracuje zpráva přicházející z mikrometru.

Je to velmi výjimečný případ, ale může se stát, že v době kdy se začnou číst jednotlivé piny, není odchycen první bit ale například již šestý a údaje získané z mikrometru pak nedávají smysl. Je zde proto podmínka, že hodnota výchylky musí být platná v ohledu rozpětí měření mikrometru, jinak se po 170 ms provádí měření znovu, dokud není získáno platné měření. Jestliže získané měření odpovídá podmínce, pak se již jen poskládá odchozí zpráva pro řídicí systém REX a odešle po sériové lince.

3.1.4 Komunikační protokol mezi Arduinem a řídícím systémem REX

Jako první verze komunikace mezi řídícím systémem REX a deskou Arduino byl zvolen projekt REXduino, který má pod záštitou společnost vyvíjející řídící systém REX. Slave program na desce Arduino rozšířený o ovládání krokového motoru a zpracování dat z mikrometru pak sbíral data z jednotlivých pinů na základě příkazů, které mu byly zaslány po sériové lince ze systému REX, a zpracovaná data zasílal zpět ve specifické formě odpovědi. V balíku projektu REXduino jsou pak bloky ArduinoUNO a ArduinoMEGA2560, které přímo ovládají piny na Arduinu.

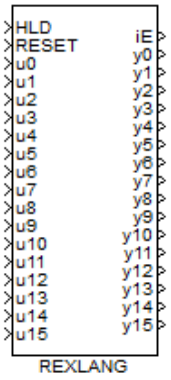


Obrázek 14: REXduino masky

Pro personalizované zpracování dat v Arduinu bylo třeba upravit Slave kód a příkaz se blokově poskládaný posílal přes přípojný bod userSend. Arduino pak příkaz vyhodnotilo a odeslalo odpověď, kterou bylo třeba opět z userRecv rozložit. Zbylé přípojné body ale zůstaly nevyužité, protože bylo domluveno, že lepší varianta bude, když data bude zpracovávat arduino a jen je připravené předávat systému REX, který data zpracuje a dá jim potřebnou formu pro webovou aplikaci, která je na REX přímo napojená. Proto se dospělo k rozhodnutí, že takováto varianta je zbytečně robustní a vzhledem k nutnosti blokového skládání příkazu i zbytečně složitá, a z REXduina se využije blok REXLANG [9], který je ukrytý uvnitř pod zmíněnými maskami.

Bloky v systému REX pokrývají většinu základních i pokročilých operací, které se nad různými druhy dat dají dělat. Najdou se však případy, kdy je vhodnější nějaká data přepracovat kódově a pak je předat do dalších bloků. Právě pro tyto případy slouží blok REXLANG. Jeho schéma je vidět na následujícím obrázku.

Blok disponuje šestnácti vstupy u0 až u15 a také šestnácti výstupy y0 až y15. Dále má bod HLD, který vypíná zpracování kódu, když je na něj přivedena náběžná hrana. V případě přivedení náběžné hrany na RESET se provede celý kód v REXLANGu od začátku i s inicializací. Na výstupním bodě iE je pak zobrazena chybová hláška. V samotném kódu jsou pak důležité tři části - sekci init, main a exit. Init se provádí jen při nahrání exekutivy nebo po přivedení náběžné hrany na bod RESET. Část main se provede při

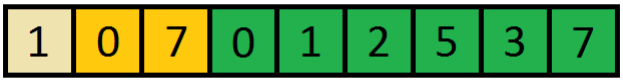


Obrázek 15: Blok REXLANG

každém plánovaném "ticku"celé exekutivy, zde je tedy hlavní část kódu, která zpracovává přijmutá data, a exit se provede při ukončení systému nebo nahrávání nového algoritmu.

Následující obrázek je již grafické schéma informace, která se odesílá ze systému REX do Arduina. První bajt (podbarven béžově) obsahuje informaci o typu příkazu. Tato vlastnost je převzata z projektu REXduino, avšak zde prozatím nemá funkci, je to jen bajt připravený pro pozdější vylepšování prototypu. Další dva bajty (podbarveny žlutě) obsahují informaci o počtu měření. Poslední dva bajty (podbarveny zeleně) nesou informaci o průměru šípu. Průměr šípu je zde jako celé číslo, tedy jsou to desetiny milimetru.

Obrázek 16: Schéma zprávy pro Arduino

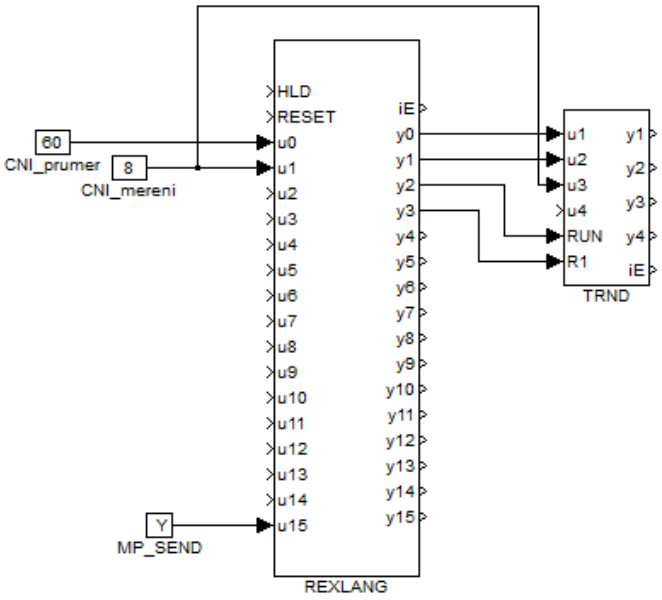


Obrázek 17: Schéma zprávy pro REX

Dále je tu schéma zprávy jdoucí opačným směrem, tedy z Arduina do REXu. Zpráva se skládá z 9 bajtů a všechno jsou to jen číselné informace. První bajt (béžově podbarvený) může nabývat jen hodnot 1 nebo 2 a obsahuje informaci o znaménku měření. Mikrometr totiž povoluje i měření v záporných hodnotách. Další dva bajty (žlutě podbarvené) udávají pořadí daného měření. Původně na tomto místě byl úhel, který se pojil ke konkrétnímu

měření, ale na to byly potřeba další dva bajty, jelikož úhel mohl být až trojmístný a ještě byla potřeba informace o desetinné čárce. Proto se přešlo k formě pořadí měření a úhel je následně dopočten v kódu REXLANGu. Poslední sekcí přenesené informace je samotná naměřená hodnota a ta je obsažena v posledních šesti bajtech (zeleně podbarvené). Opět je informace celočíselná a v bloku REXLANG se převádí na reálnou hodnotu, protože u informace o měření se ví, že desetinná čárka je před řádem tisíců celého čísla, má tedy tři desetinná místa.

3.1.5 Popis modelu exekutivy a algoritmu v bloku REXLANG



Obrázek 18: Model exekutivy pro REX

Kromě bloku REXLANG [9] jsou v modelu ještě další 3 typy bloků.

- CNI

  Blok CNI je blok, jehož výstupem je konstanta typu Integer. V modelu jsou tyto bloky dva, a to *CNIprumer* a *CNImereni*. Jejich názvy značí jakou informaci obsahují. V jednom bloku je průměr šípu jako celé číslo, takže je v desetinách milimetru, a v druhém bloku je obsažen požadovaný počet měření. Oba bloky jsou napojeny na webovou aplikaci a čerpají údaje z ní. Mají v sobě přednastavené hodnoty na šest milimetrů průměr a osm měření. Obě konstanty jsou pak přivedeny do REXLANGu přes vstupy u0 a u1.

- MP

  Dalším využitým blokem je blok Manual Pulse. Tento blok pa aktivaci vytvoří signál s náběžnou hranou po zvolenou dobu trvání a je napojen do webové aplikace na tlačítko start a kliknutím na toto tlačítko se odešle příkaz do Arduina. V tomto modelu má Manual Pulse periodu 200ms. Se stejnou periodou je spouštěna i celá exekutiva. Je důležité aby Manual Pulse i celá exekutiva měli stejnou periodu, jelikož chceme, aby se příkaz do arduina odeslal právě jednou. Náběžná hrana je přiváděna na vstup u15 a v kódu následně zpracovávána.

- TRND

  Tento blok je schopen v reálném čase ukládat až 4 signály a vykreslovat jednotlivé hodnoty signálů do grafu s časovými značkami. Data jsou ukládána do bufferu a celý blok je napojen na objekt ve webové aplikaci, v němž jsou výsledky zobrazovány. Pro potřeby vykreslování výsledků této práce jsou do bloku TRND přivedeny tři signály. Na u1 přichází hodnota měření výchylky šípu, na u2 jde příslušný úhel otočení šípu a do u3 pak jde signál obsahující počet měření. Na bloku jsou pak ještě využity přípojné body RUN a R1. TRND ukládá data do bufferu jen v případě, že na má na vstupu bodu RUN náběžnou hranu, takže je na přípojný bod RUN posílána náběžná hrana, ale jen ve chvíli, kdy byla přijata data zt Arduina a REXLANG je má zpracovaná. Zamezí se tím ukládání nepotřebných hodnot. R1 pak slouží k vymazání obsahu bufferů v bloku TRND a buffery jsou mazány vždy, když se odešle zpráva pro Arduino.

3.1.5.1 Kód bloku REXLANG

je psán jazyce podobném jazyku C. Samotný kód je obsažen v přílohách.

Jako první v kódu probíhá inicializace veškerých proměnných od nastavení vstupů a výstupů až po velikosti bufferů. Po vytvoření proměnných a definování konstant proběhne metoda *init*, ve které proběhne část kódu, která se provádí jen při nahrání exekutivy nebo při resetu bloku REXLANG. V této metodě se nastaví proměnná vyjadřující stav komunikace po sériové lince na -1, tedy že sériová komunikace není otevřena. Dále ještě výstupy do bodů R1 a RESET v bloku TRND se nastaví na $false$.

Pokračuje se pak v kódu metodou *main*, což je část kódu, která probíhá s periodou exekutivy. Probíhá zde nejdůležitejší část algoritmu. Na začátku metody se ještě vytvoří nějaké proměnné a jiné zas vynulují. První důležitá část je otevření komunikace s deskou

Arduino. V případě, že se nepodařilo navázat komunikaci, zbytek kódu se neprovádí a čeká se na otevření komunikace. Když je komunikace otevřena, tak se první zjišťuje jestli není na vstupu, kde je přiveden Manual Pulse, náběžná hrana. Pokud ano, tak se poskládá příkaz pro desku arduino z dostupných hodnot, vyresetují se buffery bloku TRND a nahrají se do nich nulové hodnoty. Je třeba do TRND odeslat prvně nulové hodnoty, protože dostupná funkce na sběr dat z TRND v rámci webové aplikace má někdy tendence vynechávat data z první pozice bufferů. V případě, že z Manual Pulse náběžná hrana nejde, čtou se případná příchozí data ze sériové linky. Pokud byla zaznamenána zpráva ve správné formě, tak se zpráva dekóduje a jednotlivé hodnoty se odešlou na výstupy z bloku REXLANG a sepne pravdivostní hodnota pro ukládání dat do TRND. Takto se do bloku TRND uloží postupně počet trojic hodnot roven počtu požadovaných měření.

Poslední metodou je metoda *exit*. Tato metoda se spouští jen při ukončení exekutivy nebo například při nahravání exektivy nové. Obsahuje jen uzavření sériové linky.

3.2 Vývoj a vzhled webové aplikace

3.2.1 Návrh webové aplikace

Návrh webové aplikace je vytvořen použitím RexHMI rozšíření pro open-source vektorový grafický editor Inkscape. Nativním formátem editoru Inkscape je .SVG a je hojně využíván právě pro návrhy webových aplikací všeho druhu. V rozšíření je knihovna několika předem vytvořených objektů jako jsou tlačítka, posuvníky, led diody a například i real-time graph z bloku TRND. Lze vyvářet i objekty vlastní a přidat je do knihoven.

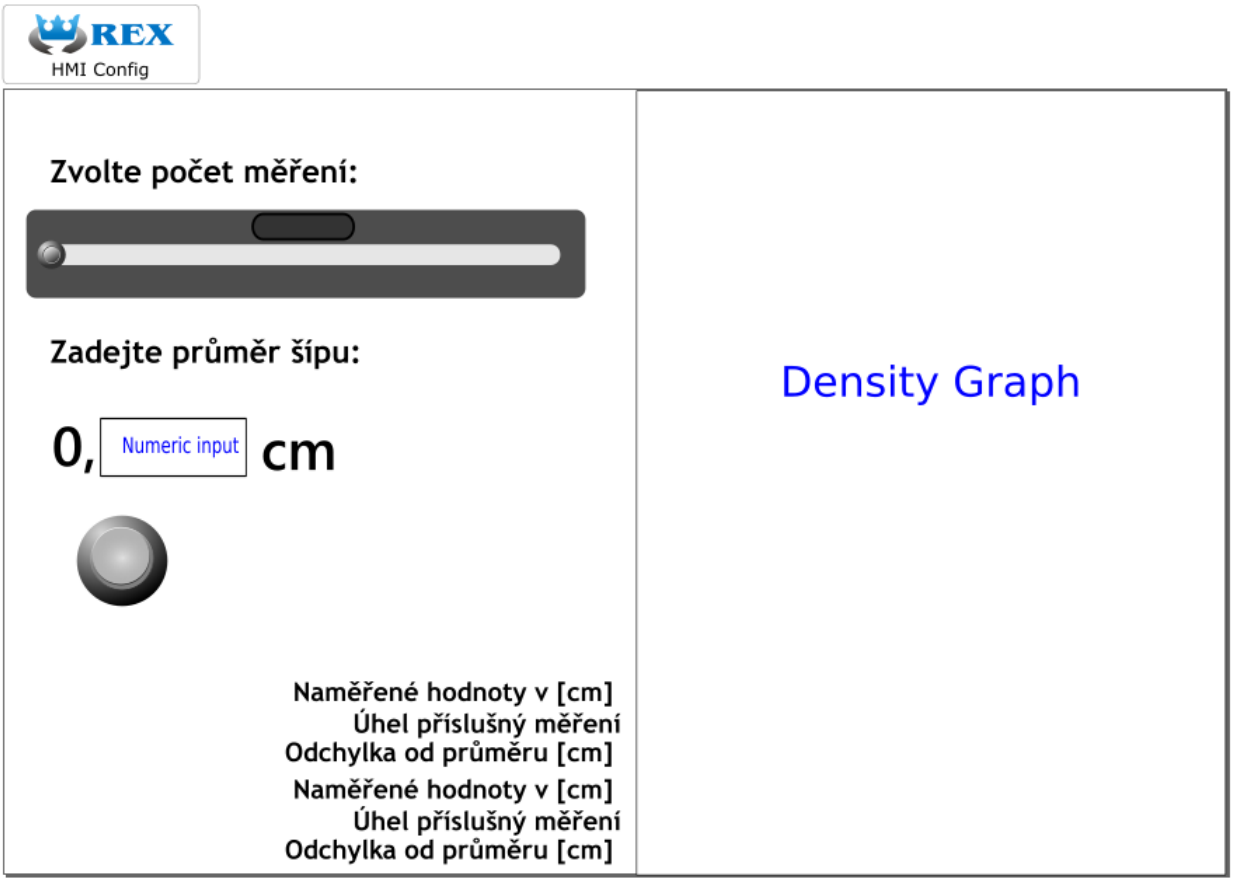
Původní návrh webové aplikace byl vytvářen v samotném programu Inkscape a sloužil jako náhled, jak by mohla aplikace vypadat a fungovat, a také jako odrazový můstek pro vykreslování získaných dat. K rozšíření RexHMI se přešlo hlavně kvůli jednoduchosti následného napojení aplikace na systém REX. V této aplikaci jsou využity tři objekty z připravené nabídky a jeden objekt vytvořený pro potřeby práce.

Jedním objektem je posuvník, pomocí kterého se zadává požadovaný počet meření. Posuvník je omezen pouze na čtyři stavy, odpovídající přednastavenému počtu měření, tedy 4, 8, 12 a 16. Posuvník je připojen na blok CNIprumer v REXu a s každým během exekutivy si konstanta aktualizuje svou hodnotu z posuvníku.

Pod posuvníkem je políčko nadepsané "Numeric input". Do tohoto pole se zadává celé číslo udávající průměr šípu. Blok je omezen na hodnoty 40 - 99 a pokud je vložena jiná hodnota, tak se do REXu nic neodešle, zůstává nastavená předchozí hodnota a pozadí

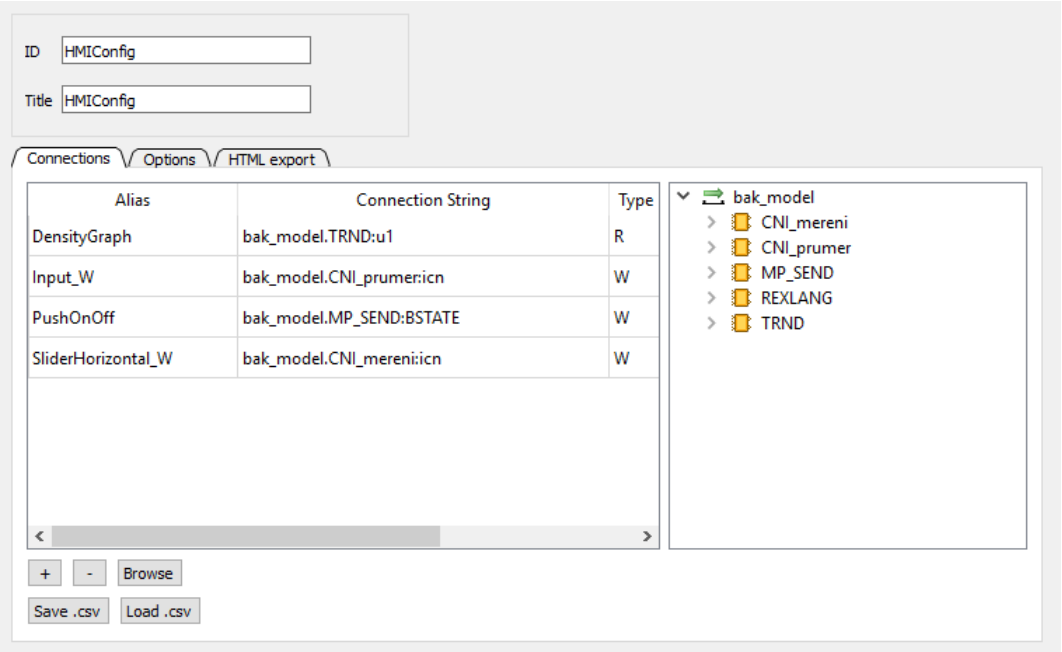
bloku se začervená, že nebyla zadána platná hodnota. Objekt je pak připojen na konstantu typu Integer CNIprumer v blokovém schématu.

Posledním prvkem z dostupné knihovny, který je využit, je tlačítko. Při stisku tlačítka se vyšle náběžná hrana z "Manual Pulse"bloku do REXLANGu.



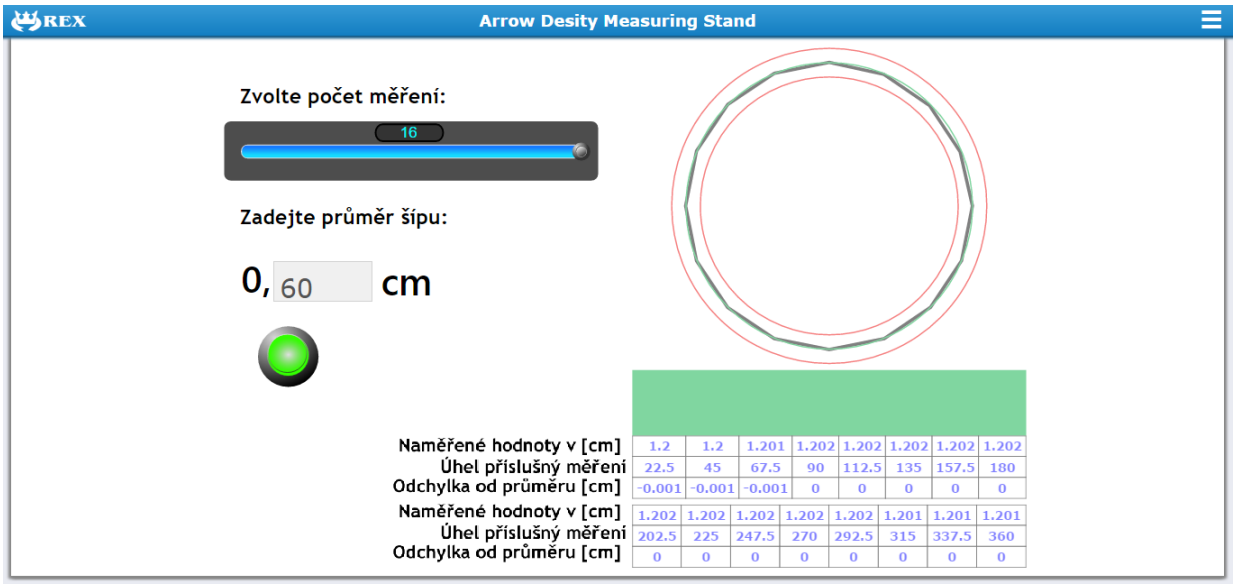
Obrázek 19: Návrh webové aplikace v programu Inkscape

Nejdůležitějším prvkem je vytvořený objekt "Density Graph", který má na starosti veškeré vykreslování dat tak, aby byla uživatelsky čitelná. Zde se vykreslí charakteristika rozložení tuhosti šípu v několika jeho směrech. Zelená kružnice udává průměrnou naměřenou hodnotu tuhosti šípu a červené kružnice znamenají tolerovatelné meze v závislosti na průměrné tuhosti. Pod charakteristikou se po průběhu měření zobrazuje zelený nebo červený obdélník, který signalizuje, zda-li šíp vyhovuje kvalitativnímu kritériu nebo ne. V poslední řadě se pak ve spodní části objektu vykreslí tabulka naměřených hodnot, spolu s příslušnými úhly měření a odchylkou naměřené hodnoty od hodnoty průměrné.



Obrázek 20: Propojení aplikace a REXu

Propojení webové aplikace a REXu je díky rozšíření RexHMI o to jednodušší. V programu Inkscape stačí otevřít dialogové okno "Edit Element"a zobrazí se seznam objektů, kde každý má svoji položku "Connection String". V případě, že je spuštěná požadovaná exekutiva, stačí kliknout na tlačítko "Browse"a zobrazí se seznam přípojných bodů. Jednotlivým objektům pak stačí do atributu připojení přiřadit požadovaný přípojný bod.



Obrázek 21: Finální GUI

Zde je již finální vzhled webové aplikace pro průběh s šestnácti měřeními.

3.2.2 Density Graph

Objekt Density Graph má svůj vlastní *.js* soubor, který se stará o vykreslování dat. V první řadě čeká na nově příchozí data do bloku TRND a načítá je do bufferu. Z bloku TRND se mu dostává tří informací, hodnoty měření, úhlu otočení pojící se k danému měření a uživatelem zadaný požadovaný počet měření. Počítá si kolik měření dostal a ve chvíli, kdy se množství přijmutých dat rovná požadovanému počtu měření, přejde ke zpracování a vykreslení dat.

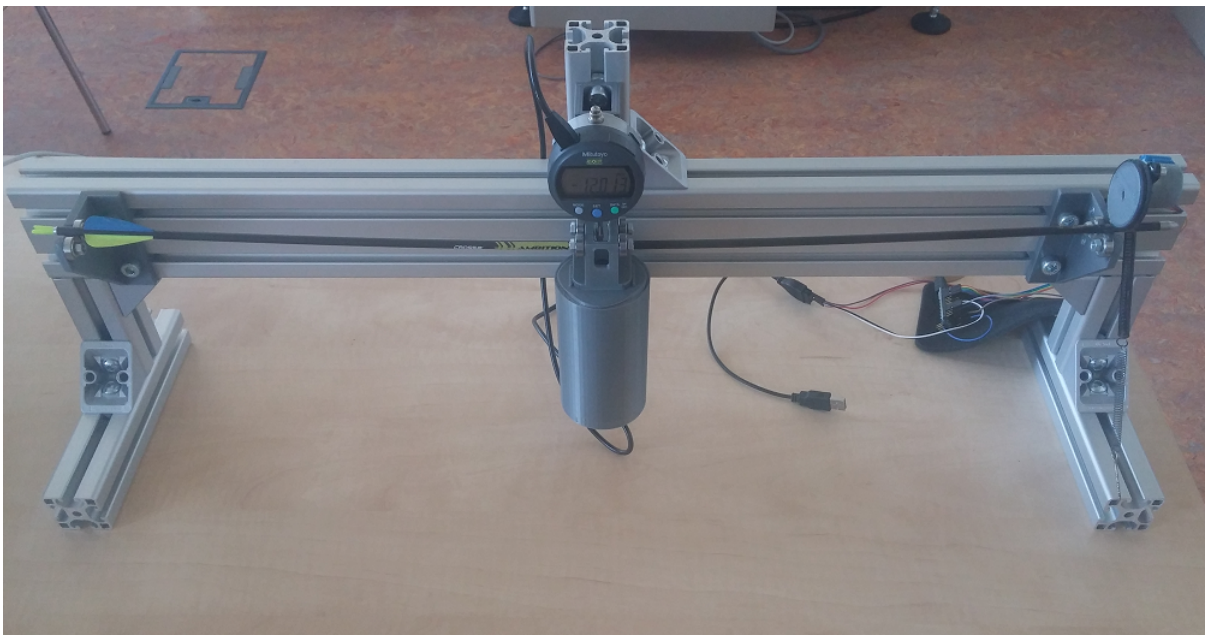
Ve zpracování se nejprve spočte průměrná hodnota ze všech měření. Tabulka je zpracovávána jiným skriptem a jako parametr jí dává HTML příkaz. Po spočtení průměrné hodnoty je tedy poskládán příkaz vykreslení tabulky. Všechny tři kružnice jsou fixně usazeny v objektu a charakteristika je vykreslována poměrně k nim. Souřadnice jednotlivých vrcholů charakteristiky jsou na základě poměrů příšlušného měření a průměrné hodnoty spolu s přepočtem na vzdálenost od středu vykreslování. Místo vzdálenosti od středu je ale potřeba získat kartézské souřadnice vrcholu a k tomu je v algoritmu využita sinová věta, kterou lze využít díky znalosti úhlů pro daná měření. Cyklicky se tedy projdou všechna získaná data a poskládá se cesta vykreslování pro objekt charakteristiky, který je tvořen použitím elementu *path*. Následně dojde již jen k vykreslení všech připravených objektů, které se při spuštění nového měření smažou a vykreslí opět po dokončení měření.

3.3 Popis standu, návod k jeho ovládání a blokové schéma

3.3.1 Provedení standu

bylo navrženo vedoucím práce a stand je vyhotoven z hliníkových profilů. Části, které podpírají šíp, rameno na kterém je připevněn krokový motor a hnací kolečko byly vytištěny na 3-D tiskárně. Při vývoji standu bylo použito několik variant hnacích koleček a zákonitě k nim i několik variant ramen.

Stand i závaží jsou v kontaktu se šípem přes ložiska, tak aby se co nejvíce usnadnilo otáčení šípu. Jehla mikrometru je umístěna uprostřed mezi podpěrnými body a dotýká se šípu v jeho středu. Hnací kolečko krokového motoru je pak opatřeno na obvodu gumovými těsnicími kroužky pro lepší trakci a rameno, na němž je krokový motor upevněn, je taženo k šípu pružinou uchycenou ke spodní části konstrukce. Mikrometr je usazen v další vytištěné části a výška usazení mikrometru, stejně jako vzdálenost mezi opěrnými body, se dá upravit. Úprava vzdálenosti opěrných bodů je na snadě v případě, že se měří dětské šípy, které jsou o poznání kratší.



Obrázek 22: Vyhotovení standu

3.3.2 Návod k ovládání

1. Před měřením šípu je třeba nejprve zkalibrovat mikrometr. Vložením šípu bez závaží do stojanu, spuštěním jehly na šíp a dvojitým stiskem prostředního tlačítka po displejem se vyresetuje měření mikrometru na nulovou hodnotu v klidové poloze šípu.

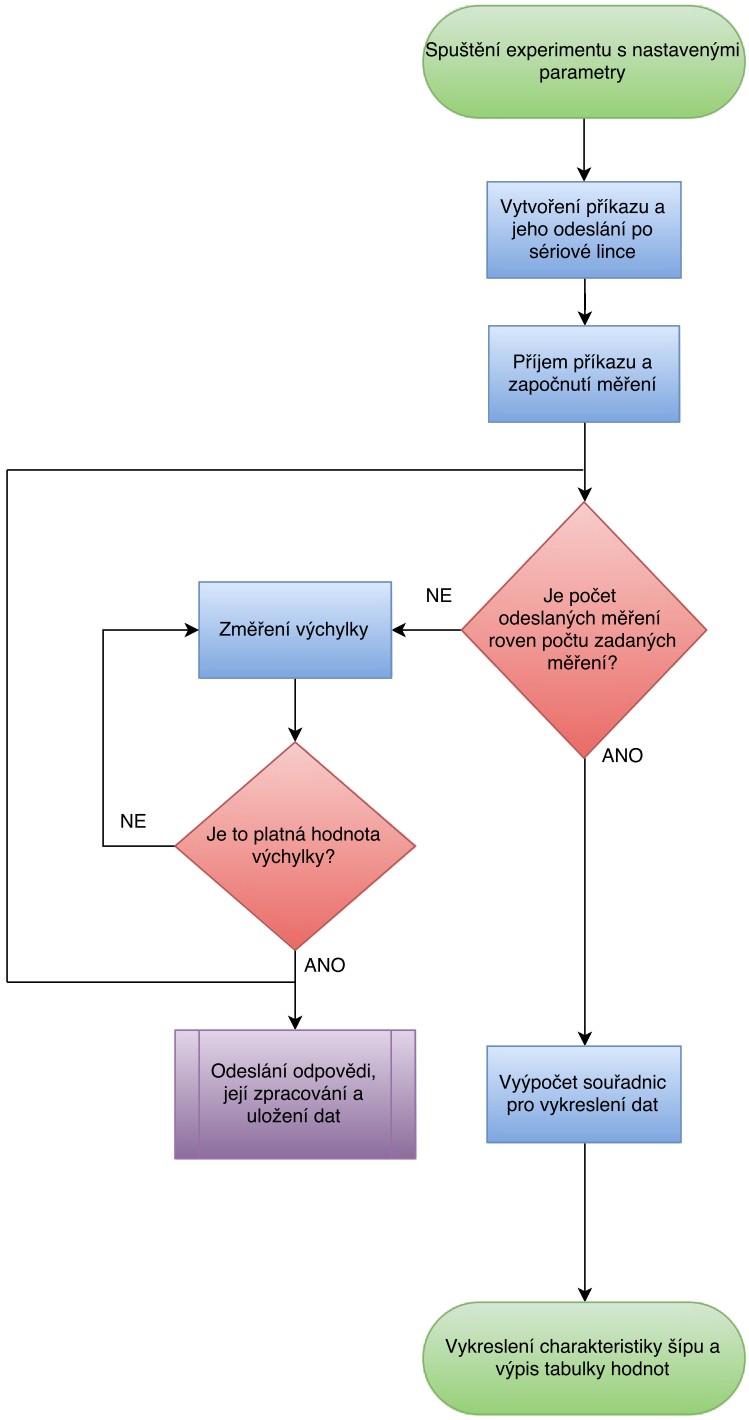
2. Dalším krokem je výběr požadovaného počtu měření. Posuvník nabízí čtyři možnosti a záleží na uživateli jak důkladné měření požaduje. S počtem měření se zvyšuje i doba trvání experimentu.

3. Následně je třeba aplikaci zadat průměr šípu. Je dobré průměr šípu přeměřit posuvným měřítkem s přesností na dvě desetinná místa centimetru. Toto dvouciferné číslo za desetinnou čárkou se zadává do daného pole a musí být v rozmezí 40 - 99.

4. Pak již stačí jen stisknout tlačítko a vyčkat na skončení experimentu, vykreslení charakteristiky šípu a tabulky hodnot.

3.3.3 Blokové Schéma

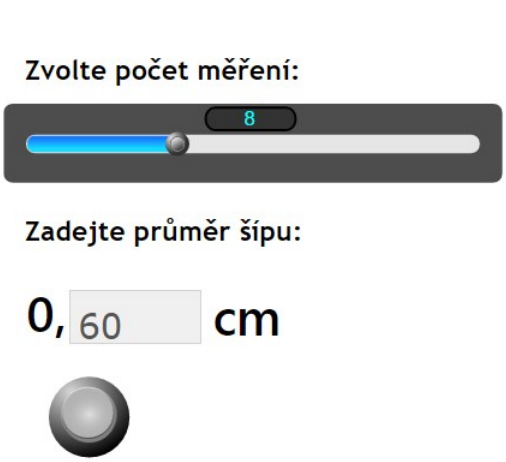
Zde je blokové schéma řídicího systému pro zpřehledění jeho průběhu.

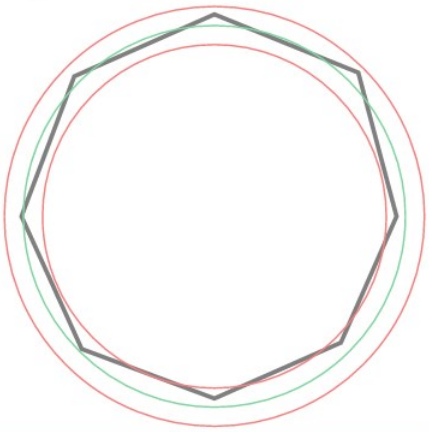


Obrázek 23: Blokové schéma řídicího systému

3.4 Příklady měření

V této sekci jsou dva příklady měření. První obrázek (Obrázek 24) je případ, kdy změřený šíp byl vyhodnocen jako kvalitní. V charakteristice je vidět, že žádný její vrchol nepřekračuje stanovené kvalitativní meze, ačkoliv se jim na mnoha místech blíží. Tabulka pak obsahuje přesné hodnoty odchylek. které se pohybují v tisícínách centimetru.



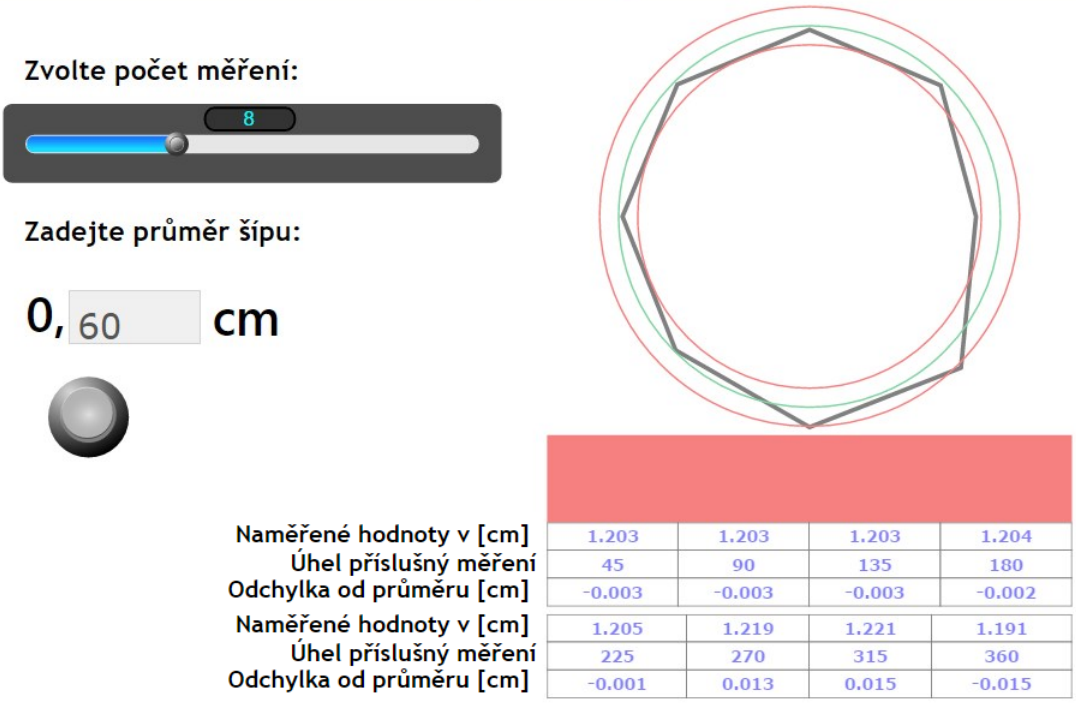


Naměřené hodnoty v [cm]	1.208	1.207	1.204	1.201
Úhel příslušný měření	45	90	135	180
Odchylka od průměru [cm]	0.008	0.007	0.005	0.001
Naměřené hodnoty v [cm]	1.198	1.194	1.192	1.194
Úhel příslušný měření	225	270	315	360
Odchylka od průměru [cm]	-0.002	-0.005	-0.008	-0.006

Obrázek 24: Vizualizace výsledků experimentu č.1

Druhý obrázek (Obrázek 25) je výstupem měření šípu, který nesplňuje požadované kritérium kvality. Z vykreslené charakteristiky je poznat, že dokonce ve dvou jejích vrcholech překračuje jednoprocentní kvalitativní kritérium. V těchto bodech je tedy hodnota měření větší než 101% průměrné hodnoty a v jednom bodě není kritérium splněno opačným směrem a hodnota měření je menší než 99% průměrné hodnoty.

Červený obdélník tedy signalizuje, že šíp je nevyhovující a přesné hodnoty odchylek jsou opět k vidění v tabulce.



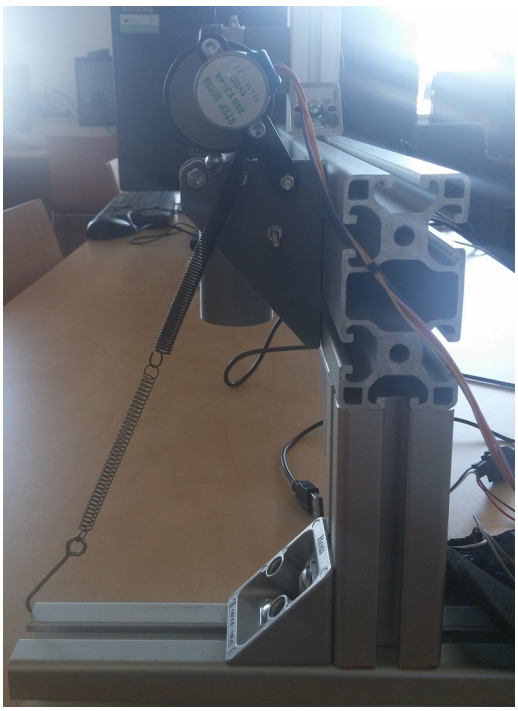
Obrázek 25: Vizualizace výsledků experimentu č.2

3.5 Zjištěné nedostatky a prostor pro vylepšení

Pravděpodobně nejproblémovější částí tohoto prototypu je krokový motor. Úkolem bylo navrhnout stand za co nejnižší možnou cenu, proto byl zvolen krokový motor, kterému stačí napájení dostupné z desky Arduino a nepotřebuje externí zdroj. Nejprve se používalo malé hnací kolečko a metoda ovládání krokového motoru, která zajišťovala velký počet kroků, ale takto se dalo otáčet jen šípem bez zatížení. Při zatížení šípu buď prokluzovalo kolečko, nebo se motor neotáčel vůbec. Muselo se tedy přejít ke kolečku většímu a použití metody, která poskytuje větší točivý moment za cenu nižšího počtu kroků.

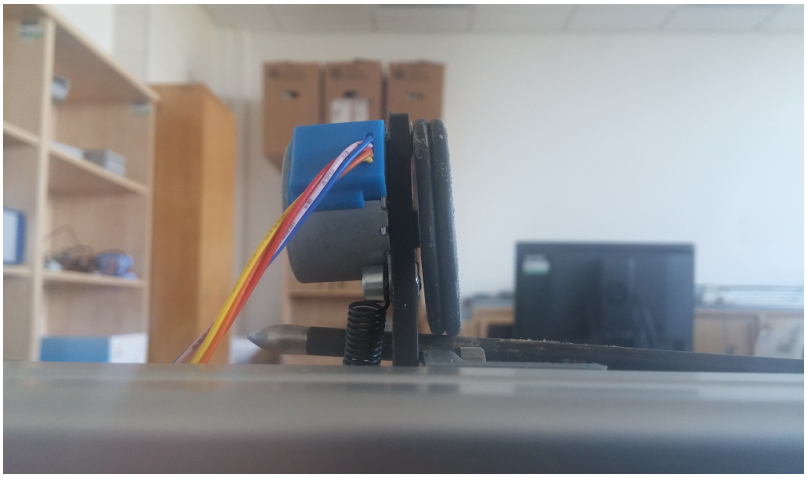
Takto se dá již poměrně spolehlivě otáčet, nicméně čím více se motor používá, tím rychleji se opotřebovává, ztrácí točivý moment a případné prokluzy jsou častější. V konečném návrhu standu, který bude lukostřelecký klub používat, by určitě měl být výkonnější krokový motor i za cenu externího napájení a vyšší pořizovací ceny.

Motor je při měření tažen k šípu pružinou, která je v jednom místě připojena k rameni, na němž je krokový motor umístěn, a druhý konec se po vložení šípu háčkem připevní ke konstrukci. Způsob úchytu spodní části pružiny je třeba změnit hlavně proto, že rameno a přípojný bod na hliníkové kontrukci nejsou v jedné ose a rameno to táhne nejen dolů, ale částečně i do strany. Tento problém činí složitější i změnu vzdálenosti mezi podpěrnými konstrukcemi při měření kratších šípů.



Obrázek 26: Boční pohled na stand

Tato výchylka ramene do strany je vidět na Obrázku 27. Není však způsobena primárně silou pružiny, ale je způsobena tím, že rameno se otáčí kolem osy, která má vůli ve svém upevnění. Tento problém by mělo vyřešit lepší upevnění stávající osy nebo přepracování podpěrné konstrukce.



Obrázek 27: Pohled na vychýlení ramene

Po vyřešení zmiňovaných problémů by se měla dát použít slabší pružina, tudíž by se snížilo namáhání krokového motoru a docházelo by k menšímu opotřebení.

Prostor pro vylepšení není ale jen na straně provedení standu. Do výsledného řidicího systému bude po konzultaci s lukostřeleckým klubem dobré zakomponovat volbu kvalitativního kritéria. Momentálně je toto kritérium nastavené tak, že pro vyhodnocení šípu jako kvalitní musí splňovat podmínku, že odchylka naměřené hodnoty od průměrné hodnoty nesmí být větší než jedno procento průměrné hodnoty.

Důležitým bodem zlepšení funkce standu je úprava ovladače mikrometru. V aktuální verzi ovladače může nastat situace, že hodnoty přicházející z mikrometru nejsou čteny od správného okamžiku a získaná informace je chybná. Ačkoliv je určitým způsobem tento problém řešen, tak nepokryje každou variantu z možných chyb a to způsobuje, že měření není stoprocentně spolehlivé a je lepší provést na jednom šípu vícero experimentů.

4 Závěr

Cílem této práce bylo vytvoření řídicího systému standu pro testování kvality šípů. V teoretické části došlo k seznámení s problémem a motivací pro vytvoření zmiňovaného standu. Dále byly představeny nástroje a komponenty využité pro tuto práci.

Prvním krokem praktické části bakalářské práce bylo vyřešení ovládání krokového motoru sloužícího k otáčení šípem a vytvoření ovladače pro mikrometr, který zajišťuje měření výchylky. Na tuto sekci navázalo zpracování získaných dat a vytvoření komunikačního protokolu mezi deskou Arduino a systémem REX.

Dalším blokem této práce pak byl i návrh webové aplikace tak, aby se z ní dal celý experiment ovládat a zároveň se v ní zobrazovali získané výsledky. Pro potřeby vykreslování výsledků byl v této části vytvořen nový objekt, který zobrazuje charakteristiku šípu a tabulku měření.

Dále byl součástí práce popis řešení samotného standu a návod na obsluhu a nastavení parametrů. V této části je i blokové schéma průběhu experimentu.

Závěr práce pak byl věnován ukázkovým výsledkům experimentu a zjištěným nedostatkům takto navrhnutého standu a jeho řídicího systému spolu s navržením možných řešení.

Použitá Literatura

[1] Hunter's Friend LLC, http://www.huntersfriend.com/archery-help/hunting-target-arrows-selection-guide-chapter-3.html

[2] Eccles Archery Club, "Archery and Arrow Flight: Recurve Bow Tuning" http://www.meta-synthesis.com/archery/archery.html

[3] Easton Archery, "Arrow Tunning and Maintenance Guide"

[4] Jaroslav Sobota, Dokumentace projektu REXDuino https://www.github.com/jaroslavs/REXduino/blob/master/REXduinoUserGuide.pdf

[5] Digimatic Indicator ID-C112X/1012X, User's Manual

[6] Webové stránky projektu Arduino,https://www.arduino.cc/

[7] Webové stránky projektu Raspberry PI, https://www.raspberrypi.org/

[8] Ing. Hynek Palát, "Rozdělení převodů, třecí převody" http://www.strojka.opava.cz/UserFiles/File/sablony/SPSIII/VY32INOVACEC-08-01.pdf

[9] REX Controls s.r.o., Dokumentace bloku REXLANG, https://www.rexcontrols.com/media/HTML/DOC/ENGLISH/REXLANG.html