Středoškolská technika 2025

Setkání a prezentace prací středoškolských studentů na ČVUT

NAVÍJEČKA OBRAZŮ Z NITĚ

Jiří Frišman

Střední průmyslová škola elektrotechnická a Vyšší odborná škola Pardubice
Karla IV 13, 530 02 Pardubice

„Prohlašuji, že jsem maturitní práci vypracoval(a) samostatně a všechny použité literární prameny a zdroje informací cituji a uvádím v seznamu použité literatury a zdrojů informací.“

V Pardubicích dne ..... *.....*

podpis

Anotace

Tento projekt se zabývá návrhem a realizací automatizovaného zařízení pro řízené vinutí nití na základě dat uložených na SD kartu, které jsou později přečteny. Hlavním cílem je vytvořit systém, který umožní přesné vrtání otvorů pro hřebíky podle předem určeného tvaru obrazu, následné vložení hřebíků do otvorů a postupné vinutí nitě kolem hřebíků podle pořadí postupně čteného z SD karty. Zařízení bude řízeno mikrokontrolerem Arduino Mega 2560, který komunikuje s SD kartou přes SPI rozhraní. Parametry a informace o obrazu přečtené z SD karty poté systém automaticky přepočítá na instrukce a přecházení mezi jednotlivými stavy tvorby obrazu bude potvrzováno uživatelem. Všechen pohyb bude realizován pomocí krokových motorů a driverů pro krokové motory, které budou řízeny Arduinem.

Klíčová slova

Obraz, obraz z nitě, niť, hřebík, pin, mikroprocesor, Arduino, SD karta, krokový motor, driver pro krokové motory

Annotation

This project focuses on the design and implementation of an automated device for controlled thread winding based on data stored on an SD card, which is later read. The main goal is to create a system that enables precise drilling of holes for nails according to a predefined image shape, followed by the insertion of nails into the holes and the gradual winding of thread around the nails in the order read from the SD card. The device will be controlled by an Arduino Mega 2560 microcontroller, which communicates with the SD card via the SPI interface. The parameters and image data read from the SD card will be automatically converted by the system into instructions, and transitions between the individual states of the image creation process will be confirmed by the user. All motion will be carried out using stepper motors and stepper motor drivers, which will be controlled by the Arduino.

Key words

Piece of art, string art, string, nail, pin, microprocessor, Arduino, SD card, stepper motor, stepper motor driver

Obsah

1 Úvod

Výtvarné umění se neustále vyvíjí a s ním i způsoby jeho tvorby. V dnešní době se stále častěji uplatňují moderní technologie, které umožňují vznik originálních a precizně zpracovaných děl. Jednou z takových technik je tzv. *string art*, tedy umění vytváření obrazů pomocí nitě napnuté mezi hřebíky. Tento způsob tvorby kombinuje estetiku, matematiku a technologii, čímž představuje zajímavou oblast pro automatizaci.

Pro řízení stroje jsem zvolil Arduino Mega 2560, které poskytuje dostatek výpočetního výkonu i potřebné množství vstupů a výstupů. Celý systém je navržen tak, aby byl schopen autonomně provádět jednotlivé fáze procesu – od vrtání otvorů pro hřebíky až po samotné vinutí nitě. Člověk pouze musí potvrzovat přechody mezi jednotlivými stavy a lehce upravit geometrické rozložení konstrukce pouze pro dva stavy.

Tento projekt propojuje umění, techniku, matematiku a programování, což nejen rozšiřuje moje znalosti v oblasti automatizace a algoritmizace, ale zároveň umožňuje vytvořit esteticky hodnotná díla s praktickým využitím.

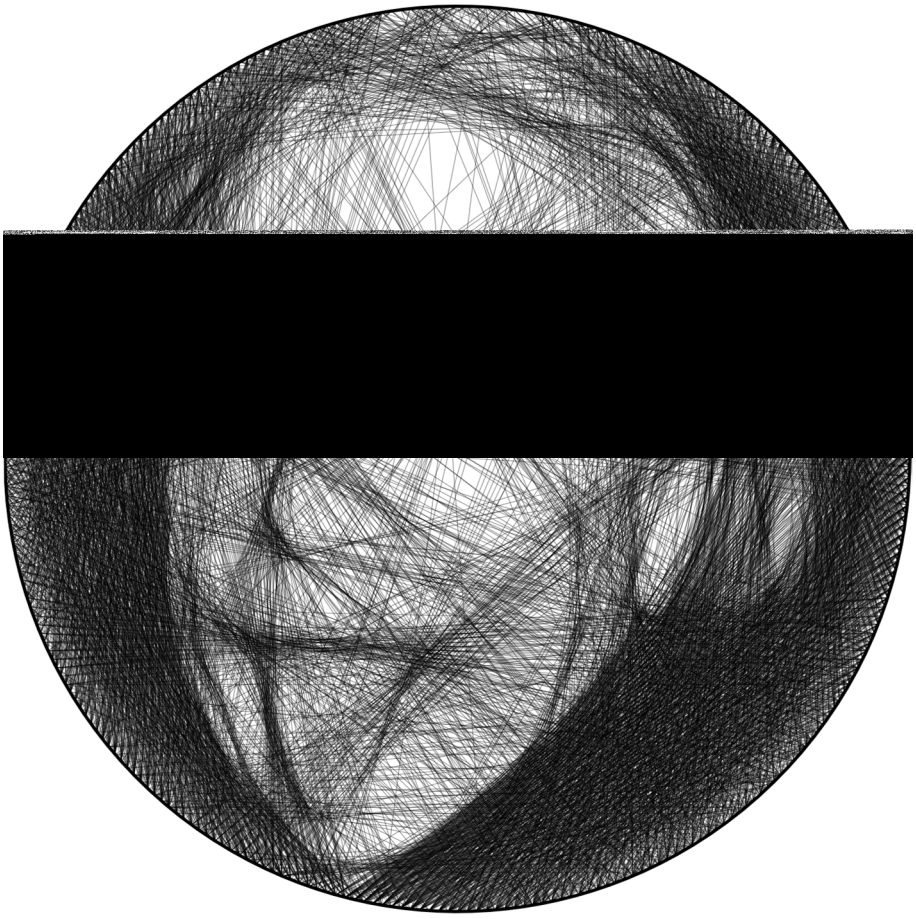
Cílem mého projektu je navrhnout a postavit zařízení, které dokáže vytvářet *string art* obrazy na základě předem připravených dat přečtených Arduinem z SD karty. Tento proces zahrnuje několik klíčových kroků – od návrhu samotného obrazu, přes jeho převedení na souřadnice hřebíků a trajektorii nitě, až po fyzickou realizaci na dřevěné matrici.

2 Teoretická část

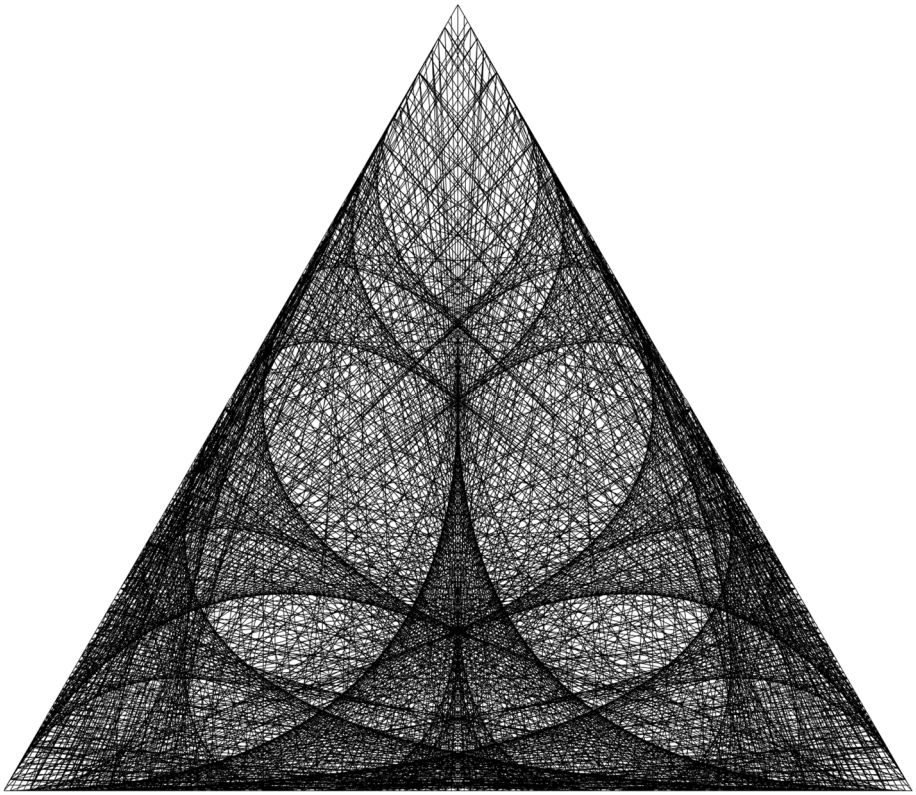
2.1 Rešerše

2.1.1 Historie string artu

Technika string artu se poprvé objevila na konci 19. století, kdy ji britská matematička Mary Everest Boole využila jako pomůcku pro výuku geometrie [1]. V průběhu 20. století se tento způsob tvorby rozšířil do uměleckého světa a získal popularitu jako dekorativní prvek. V dnešní době je string art často využíván nejen v ručním umění, ale také v kombinaci s moderními technologiemi, které umožňují automatizovanou tvorbu složitějších vzorů a motivů.



Obrázek 1: Příklad string artu: Albert Einstein



Obrázek 2: Příklad string artu: Mandala

2.1.2 Inspirace

Poprvé jsem narazil na string art ve formě obrazu v obchodě. Největší inspirací pro můj projekt je video na YouTube kanálu **Barton Dring** [2]. Barton Dring se rozhodl pro konstantní velikost a tvar obrazů s konstantním počtem pinů (hřebíků) na matrici. Jeho projekt je naprogramován v G-kódu. G-kód je velice dobrý způsob na ovládání krokových motorů, když se jedná o 3D tiskárny, CNC stroje a podobné stroje pro ovládání pohyblivých částí ve 3D prostoru, jelikož se G-kód dá programovat ve 3D souřadnicích.

Zdroj informací o ovládání krokových motorů pomocí Arduina pro mě bylo video od **How To Mechatronics** na YouTube [3], kde se rozebírá vše o konstrukci krokových motorů, typů driverů pro krokové motory a základní Arduino kód pro ovládání krokových motorů, který využívá *AccelStepper* knihovny [4], která je rozsáhlá a má velice dobrou dokumentaci.

2.2 Hardware

2.2.1 LyonZG S-200-24

Modulový napájecí zdroj LyonZG S-200-24 je síťově napájený spínaný zdroj s účinností 85 % a v projektu je užit jako zdroj energie driverů a jejich chladících ventilátorů. Jeho výstupní napětí je 24 V DC s možností jemného doladění (±10 % pomocí trimeru), maximálním výstupním proudem 8,3 A a výkonem 200 W. Jeho rozměry jsou přibližně 198 mm × 98 mm × 40 mm [5].

Obrázek 3: LYONZG S-200-24

Zdroj má šroubovací svorky pro připojení vstupu a výstupu:

- Vstupní svorky (230 V AC):

  L: Fázový vodič

  N: Nulový vodič

  PE: Zemnící vodič

- Výstupní svorky (24 V DC):

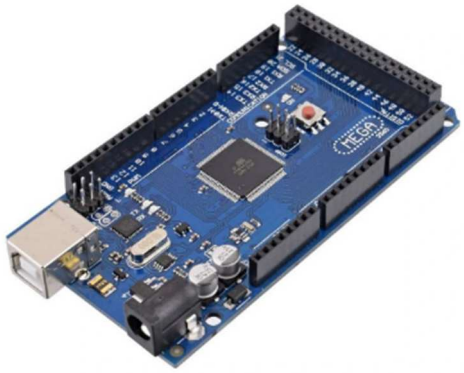
  2 × V+: 24 V

  2 × V–: GND

2.2.2 Arduino MEGA 2560

Arduino Mega 2560 je výkonná vývojová deska založená na mikrokontroleru ATmega2560. Oproti běžnějšímu modelu Arduino UNO nabízí výrazně více vstupů a výstupů, což z něj činí ideální volbu pro rozsáhlejší projekty s větším množstvím periferií. Má celkem 54 digitálních input/output pinů, z nichž 14 podporuje PWM. K dispozici je také 16 analogových vstupů a 4 hardwarová sériová rozhraní UART. Maximální pracovní frekvence procesoru je 16 MHz. Napájení je možné buď přes USB (5 V), nebo přes externí zdroj v rozsahu 6 V až 12 V. Deska obsahuje vestavěný stabilizátor napětí, který umožňuje stabilní provoz při různých napěťových vstupních

podmínkách. Pro komunikaci s dalšími zařízeními podporuje SPI, I2C a UART. Paměťová kapacita zahrnuje 256 kB Flash paměti (z toho 8 kB je vyhrazeno pro bootloader), 8 kB SRAM a 4 kB EEPROM [6].



Obrázek 4: LaskaKit Mega2560 rev3

2.2.3 MicroSD modul

MicroSD modul je zařízení umožňující ukládání a načítání dat na paměťové karty microSD pomocí komunikačního protokolu SPI. Používá se v různých embedded systémech, například pro ukládání měřených dat, logování událostí nebo čtení souborů s konfigurací. Pro práci se soubory se většinou používá knihovna SD.h, která umožňuje: otevřít soubor pro čtení/ukládání dat, číst soubor po řádcích nebo jednotlivých znacích, zpracovávat textová nebo číselná data.

Obrázek 5: microSD card modul SPI

2.2.4 Krokové motory

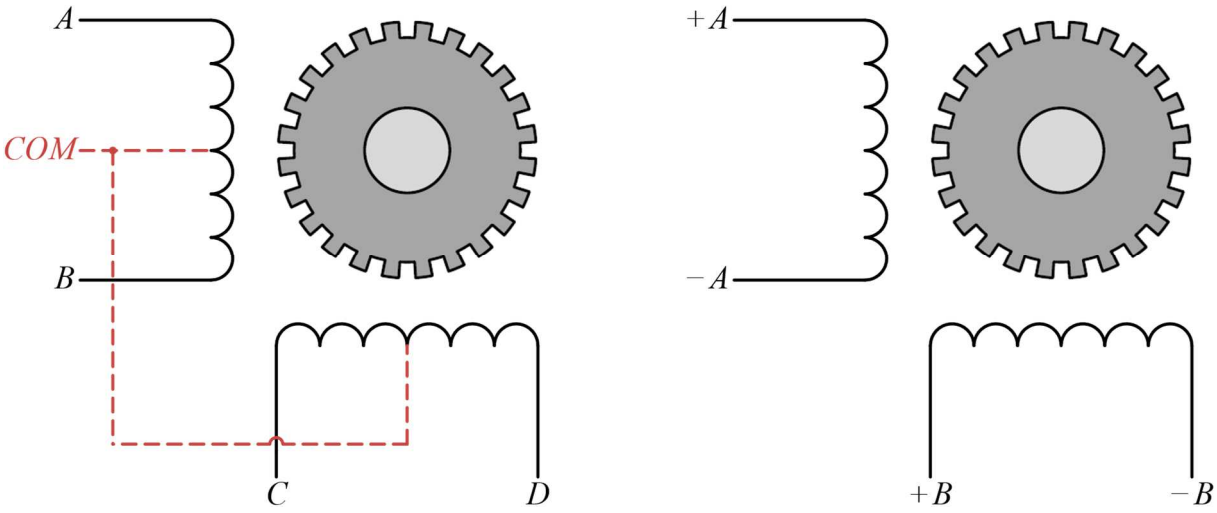
Krokové motory jsou speciální typy elektromotorů, které se pohybují po malých přesně definovaných krocích. Jejich diskrétnost je odlišuje od běžných motorů, které se otáčejí plynule. Díky své přesnosti se krokové motory často používají v automatizaci, CNC strojích, 3D tiskárnách a robotice.

Hlavním principem krokového motoru je rozdělení jedné otáčky na pevný počet kroků. Nejběžnější a nejpoužívanější krokové motory rozdělí jednu otáčku na 200

diskrétních kroků, a to znamená, že pootočení o jeden krok otočí hřídelí o 1,8° ($\frac{360°}{200}$). Počet kroků na otáčku je vypočten ze vzorce (1). Krokové motory se ovládají pulzními signály relativně velkého proudu (typicky 1 A až 3 A), přičemž každé přivedení signálu znamená jeden krok.

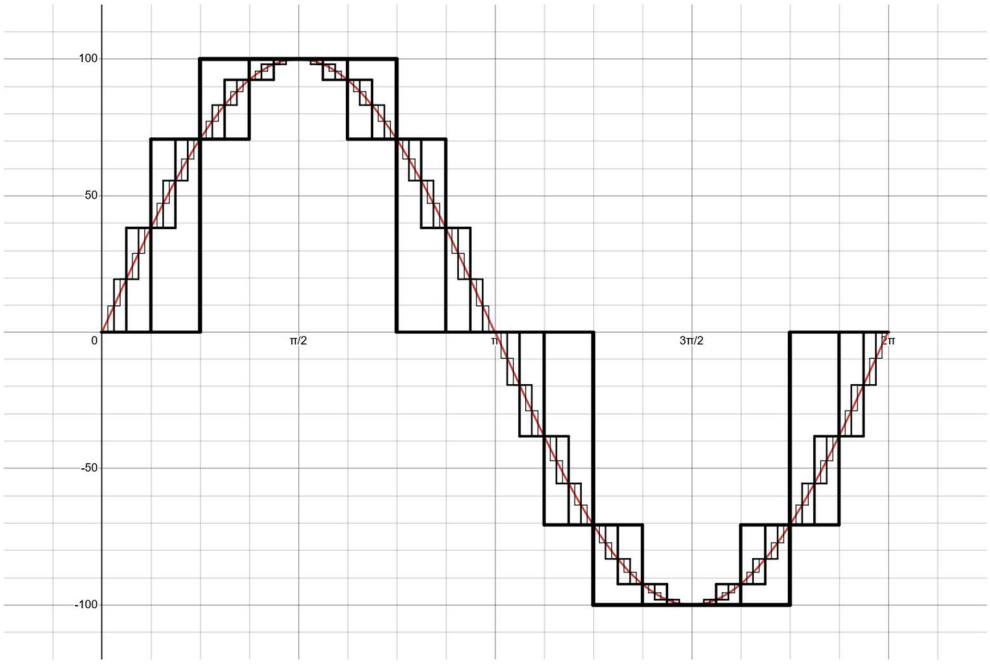
$$\frac{N}{m}, \text{kde } N \ldots \text{počet zubů v rotoru}, m \ldots \text{počet fází} \tag{1}$$

Existují dva hlavní typy krokových motorů: unipolární a bipolární. Unipolární motory mají jednodušší řízení, ale nižší výkon, jelikož se proudové signály posílají na vývody na koncích vynutí, která mají středový odboč. Jelikož proud v unipolárních vynutích teče pouze půl vynutím, tak mají cca poloviční točivý moment. Bipolární motory vyžadují složitější elektroniku, ale poskytují vyšší točivý moment, jelikož proudy tečnou celými rozsahy jednotlivých cívek.



Obrázek 6: Struktury krokových motorů – unipolární x bipolární

K řízení krokových motorů se často používají ovladače krokových motorů (tzv. *drivery*), které generují potřebné impulzy a umožňují tzv. *mikrokrokování*, tj. každý krok je rozdělen na několik menších kroků. Hlavní výhodou krokových motorů je vysoká přesnost polohování bez nutnosti zpětné vazby, ale jejich nevýhodou může být vyšší spotřeba energie a ztráta kroku při přetížení. Výhody krokových motorů jsou ale tak dobré, že se používají navzdory nevýhodám.



Obrázek 7: Mikrokrokování

Názvosloví krokových motorů je NEMA XX, kde XX je příruby pouzdra krokového motoru v palcích × 10 (např. NEMÁ 23 má rozměr 2,3 palce a tj. 56,4 mm). Obecně platí, že čím větší a těžší krokový motor je, tím větším proudem ho můžeme pohánět, tím větší je jeho točivý moment a taky jeho pouzdro lépe odvádí teplo od zahřátých vinutí.

Tabulka 1: Rozměry a jmenovité proudy krokových motorů

NEMA	Rozměr příruby [mm × mm]	Rozsah jmenovitého proudu [A]	
8	20,32 × 20,32	0,6	1,5
11	27,94 × 27,94	0,7	2,0
14	35,56 × 35,56	1,0	2,5
17	42,3 × 42,3	1,0	2,8
23	56,4 × 56,4	2,0	3,5
24	60,5 × 60,5	2,0	4,0
34	86 × 86	4,0	6,0
42	106,7 × 106,7	6,0	10,0

2.2.5 Drivery krokových motorů

Drivery krokových motorů jsou elektronické obvody, které prakticky slouží jako relé, které. Jejich hlavním úkolem je správné rozdělování elektrické energie do jednotlivých vinutí motoru tak, aby se otáčel požadovaným směrem a rychlostí. Zesilují vstupní ovládací signály a dle nich pouští nastavený proud do jednotlivých vynutí krokových motorů. Drivery regulují proud na uživatelem nastavenou hodnotu, čímž zajišťují stabilní chod motoru a zabraňují jeho přehřívání.

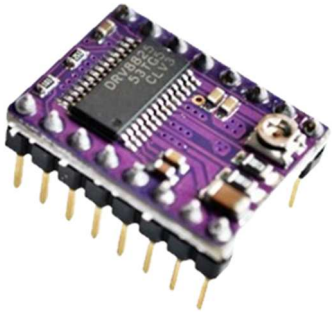
Jejich vnitřní logika umožňuje mikrokrokování. Díky mikrokrokování je pohyb krokového motoru plynulejší a snižují se tak vibrace a hluk. Mikrokrokování je obvykle nastavitelné a umožňuje zvolit různé režimy podle potřeb aplikace. Mikrokrokování je skoro vždy mocnina čísla 2 (např. 1/2 nebo 1/32).

Drivery se běžně připojují k řídicím systémům pomocí vstupních signálů, které určují počet kroků a směr otáčení. Napájení motoru je zajištěno samostatně, přičemž napětí a proud musí odpovídat specifikacím použitého motoru.

Využití driverů je široké, najdeme je v CNC strojích, 3D tiskárnách, průmyslové automatizaci a robotických systémech. Výběr správného driveru závisí na typu motoru, požadované přesnosti a výkonových požadavcích dané aplikace.

2.2.5.1 DRV8825

DRV8825 je driver pro krokové motory a je vylepšená verze klasických driverů typu A4988. Jeho maximální proud je 1,5 A s pasivním a 2,2 A s aktivním chlazením. Má zabudovanou tepelnou ochranu i proudovou ochranu. Jeho krokové režimy jsou: 1, 1/2, 1/4, 1/8, 1/16 a 1/32 kroku. Doporučené napětí zdroje výkonu pro točení motoru je 8,2 V až 36 V [7].



Obrázek 8: DRV8825

Tabulka 2: Mikrokrokování driveru DRV8825

MODE2	MODE1	MODE0	Krokový režim
0	0	0	1
0	0	1	1/2
0	1	0	1/4
0	1	1	1/8
1	0	0	1/16
1	0	1	1/32
1	1	0	1/32
1	1	1	1/32

2.2.5.2 Toshiba TB67S109

TB67S109 je driver pro krokové motory a je ještě lepší verzí driverů typu A4988. Jeho maximální proud je 2 A s pasivním a 4,5 A s aktivním chlazením. Má zabudovanou tepelnou ochranu i proudovou ochranu. Jeho krokové režimy jsou: 1, 1/2, 1/4, 1/8, 1/16 a 1/32 kroku. Doporučené napětí zdroje výkonu pro točení motoru je 10 V až 47 V [8].

Obrázek 9: TB67S109

Tabulka 3: Mikrokrokování driveru Toshiba TB67S109

MODE0	MODE1	MODE2	Krokový režim
0	0	0	OFF
0	0	1	1
0	1	0	1/2 (typ A)
0	1	1	1/4
1	0	0	1/2 (typ B)
1	0	1	1/8
1	1	0	1/16
1	1	1	1/32

2.2.5.3 TB6600

TB6600 je driver pro krokové motory a je celkově robustnější, méně náchylnější a obecně lepší verzí driverů Toshiba TB67S109, jelikož jeho vnitřní čip je TB67S109AFTG. Jeho maximální proud je 3 A s pasivním a 4,5 A s aktivním chlazením. Má zabudovanou tepelnou ochranu i proudovou ochranu. Jeho krokové režimy jsou: 1, 1/2, 1/4, 1/8, 1/16 a 1/32 kroku. Doporučené napětí zdroje výkonu pro točení motoru je 8 V až 47 V [8]. Jeho asi největší výhodou je mechanické nastavování mikrokrokování a maximálního výstupního proudu pomocí 6 integrovaných přepínačů, zatímco u menších driverů (A4988, DRV8825, TB67S109 a TMC2208) je

mikrokrokování nastavováno pomocí přivedení nastavovacích signálů (*viz* předešlé tabulky driverů – MODE X) a nastavený proud je nastaven pomocí malého šroubku na DPS driveru.

Obrázek 10: TB6600

Tabulka 4: Mikrokrokování driveru TB6600

SW1	SW2	SW3	Krokový režim
1	1	1	OFF
1	1	0	1
1	0	1	1/2 (typ A)
0	1	1	1/2 (typ B)
1	0	0	1/4
0	1	0	1/8
0	0	1	1/16
0	0	0	1/32

Tabulka 5: Proudové nastavení TB6600

SW4	SW5	SW6	Jmenovitý proud [A]	Špičkový proud [A]
1	1	1	0,5	0,7
1	0	1	1,0	1,2
1	1	0	1,5	1,7
1	0	0	2,0	2,2
0	1	1	2,5	2,7
0	0	1	2,8	2,9
0	1	0	3,0	3,2
0	0	0	3,5	4,0

2.3 Druhy souřadných systémů v 3D

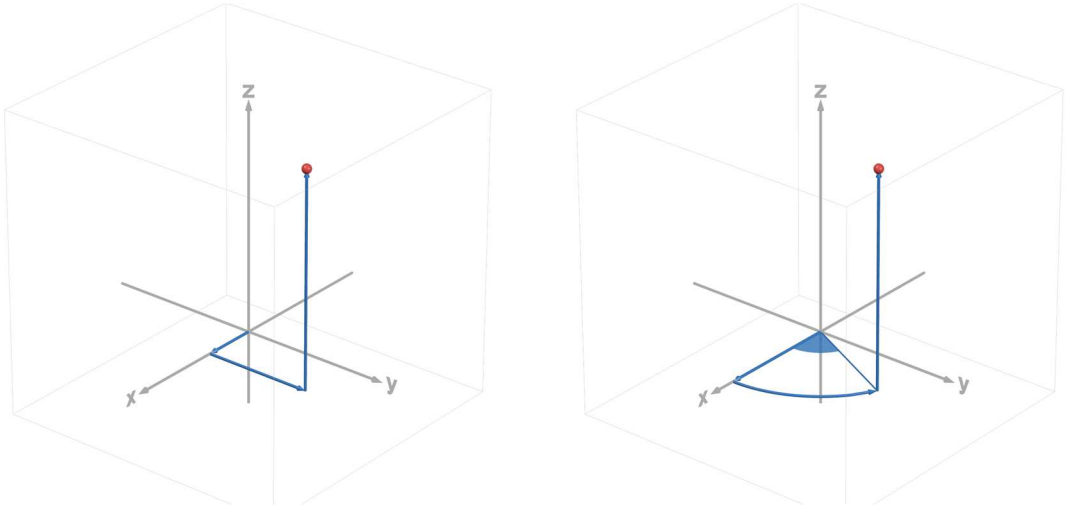
Můj projekt operuje v cylindrických souřadnicích, což ho odlišuje od většiny projektů podobných CNC strojům a 3D tiskárnám, které pracují v kartézském souřadném systému. Tento přístup přináší několik výhod i specifických výzev.

Ve standardním kartézském systému jsou souřadnice bodů definovány polohami na třech vzájemně perpendikulárních osách $[x,\ y,\ z]$, kde každá osa představuje pohyb v přímé linii. Tento souřadnicový systém je ideální pro aplikace, kde jsou lineární pohyby dominantní, jako jsou běžné CNC frézky nebo 3D tiskárny s posuvnými osami.

Na rozdíl od kartézských souřadnic souřadnice cylindrické popisují polohu pomocí radiální vzdálenosti r (vzdálenost od osy z), úhlu rotace θ (otočení kolem osy z) a výšky z. To znamená, že polohy bodů jsou popsány polárními souřadnicemi pro polohu kolmou k $[x,\ y]$ rovině a výška je určena samostatně souřadnicí z. Výhody cylindrických souřadnic jsou: efektivnější pohyb po kružnicích nebo křivkách podobných kružnici – plynulejší trajektorie; snížení lineárního posuvu – v mém případě je konstrukce značně jednodušší a robustnější; lepší využití prostoru – některé aplikace mohou efektivněji využívat dostupný pracovní prostor.

Možná snad jediná výzva je přepočet pohybových příkazů na cylindrické souřadnice *viz* (2), protože většina standardních knihoven pracuje v kartézských souřadnicích.

$$[x, y, z] \rightarrow [r, \theta, z], \text{kde } r = \sqrt{x^2 + y^2}, \theta = \arg([x, y]), z = z \tag{2}$$



Obrázek 11: Porovnání souřadnicových systémů – kartézské x cylindrické

2.4 Matematické vyjádření parametrických funkcí tvarů obrazů

Každý vinutý obraz samozřejmě musí mít nějaký tvar. Tvar obrazu do jisté míry ovlivňuje to, na co se každý jednotlivý obraz hodí. Parametrické funkce budou muset splňovat tyto podmínky:

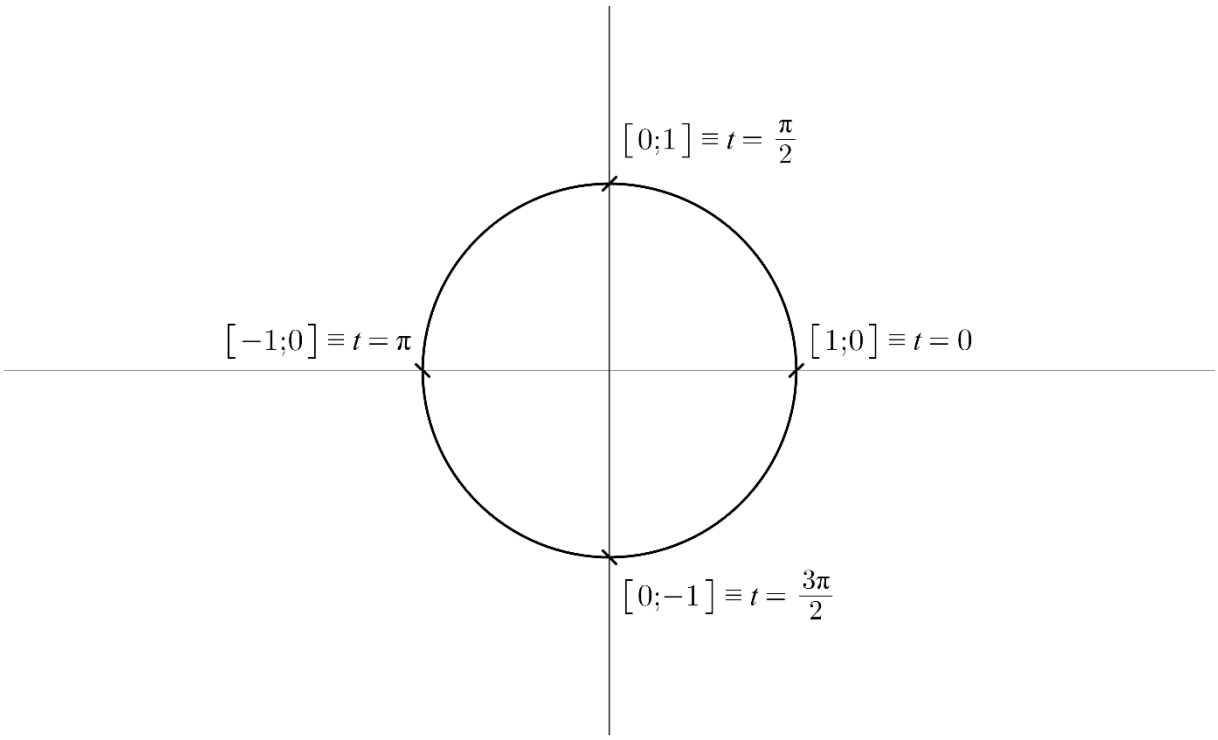
1) $f(t)$ vykreslí tvar v x, y souřadnicové soustavě pro $t \in \langle 0; 1\rangle$,
2) $f(t)$ bude periodická funkce s periodou $T = 1$,
3) $|f(t)| \leq$ poloměr matrice $- 1$

2.4.1 Kružnice

Kružnice je nejjednodušší tvar obrazu, který se hodí na portréty lidí a zvířat.

Nejjednodušší funkční vyjádření kružnice je vyjádření kružnice jednotkové:

$$k(t) = [\cos t\,; \sin t] \tag{3}$$



Obrázek 12: Jednotková kružnice

Abychom splnili podmínky 1) a 2), je nutné škálovat vstup do goniometrických funkcí v rovnici (3) tato:

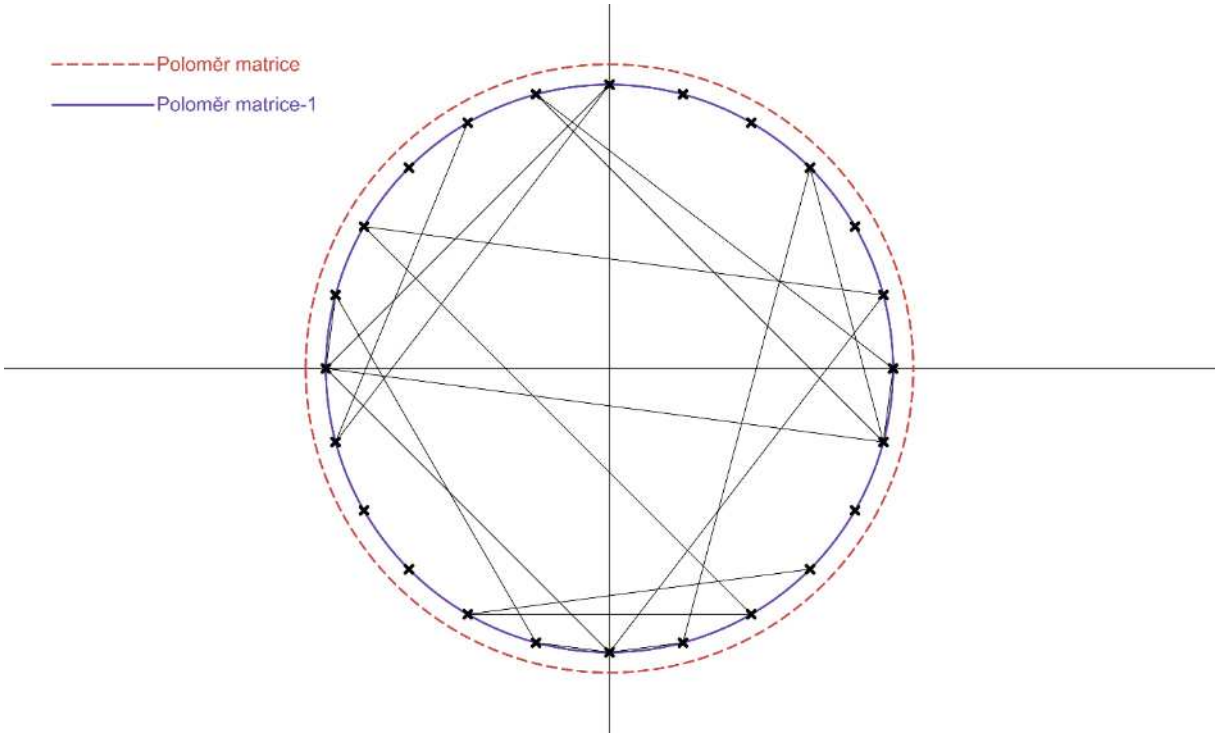
$$k(t) = [\cos(2\pi t)\,, \sin(2\pi t)] \tag{4}$$

Takže teď máme např.:

$k(1/2) = [\cos(2\pi \cdot 1/2)\,; \sin(2\pi \cdot 1/2)] = [-1; 0]$

Dále je třeba škálovat samotnou funkci z rovnice (4), jelikož nechceme, aby měli všechny obrazy poloměr 1 cm. Je nutné dodržet podmínku 3) a výsledný vztah pro kružnici je:

$$k(t, \text{poloměr}_{\text{matrice}}) = (\text{poloměr}_{\text{matrice}} - 1) \cdot [\cos(2\pi t)\,, \sin(2\pi t)] \tag{5}$$



Obrázek 13: Příklad tvaru obrazu: kružnice

2.4.2 Pravidelný n-úhelník

Pravidelný n-úhelník je po kružnici nejhezčí a nejužitečnější tvar obrazu. Hodí se spíše na mandaly a geometrické obrazce.

Základem bude jednotková kružnice, na které vybereme n rovnoměrně rozmístěných bodů, mezi kterými bude naše funkce lineárně interpolovat. Nultý bod bude mít souřadnici $[1; 0]$.

$$P_a = \left[\cos\left(a \cdot \frac{2\pi}{n}\right), \sin\left(a \cdot \frac{2\pi}{n}\right)\right] \text{ pro } a \in \{0; 1; 2; \ldots; n-1\} \tag{6}$$

Lineární interpolace dvou bodů je dána:

$\text{lerp}(A, B, t) = (1 - t) \cdot A + t \cdot B$ pro $t \in \langle 0; 1 \rangle$

Takže platí:

$$\begin{aligned}\operatorname{lerp}(A,B,0) &= 1\cdot A + 0\cdot B = A\\ \operatorname{lerp}\left(A,B,\frac{1}{2}\right) &= \frac{1}{2}\cdot A + \frac{1}{2}\cdot B = \frac{A+B}{2}\\ \operatorname{lerp}(A,B,1) &= 0\cdot A + 1\cdot B = B\end{aligned}$$

Interval $t \in \langle 0;1\rangle$ rozdělíme na n menších intervalů:

$$\left\{\left\langle\frac{0}{n};\frac{1}{n}\right\rangle;\left\langle\frac{1}{n};\frac{2}{n}\right\rangle;\left\langle\frac{2}{n};\frac{3}{n}\right\rangle;\ldots;\left\langle\frac{n-2}{n};\frac{n-1}{n}\right\rangle;\left\langle\frac{n-1}{n};\frac{n}{n}\right\rangle\right\} \tag{7}$$

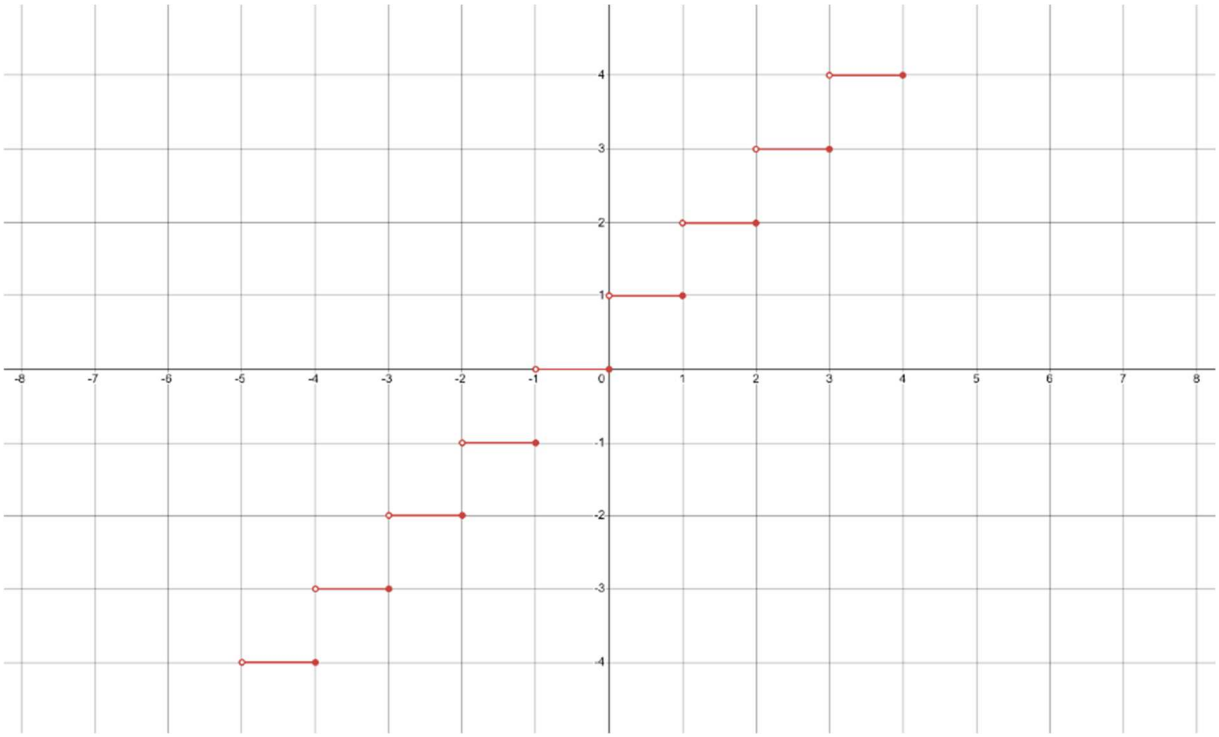
V každém intervalu *viz* (7) budeme interpolovat mezi body P_j a P_{j+1} takto:

$$\begin{aligned}\operatorname{lerp}(P_j,P_{j+1},t) &= a(t)\cdot P_j + b(t)\cdot P_{j+1}\\ \operatorname{lerp}\left(P_j,P_{j+1},\frac{j}{n}\right) &= P_j\\ \operatorname{lerp}\left(P_j,P_{j+1},\frac{j+1}{n}\right) &= P_{j+1}\end{aligned}$$

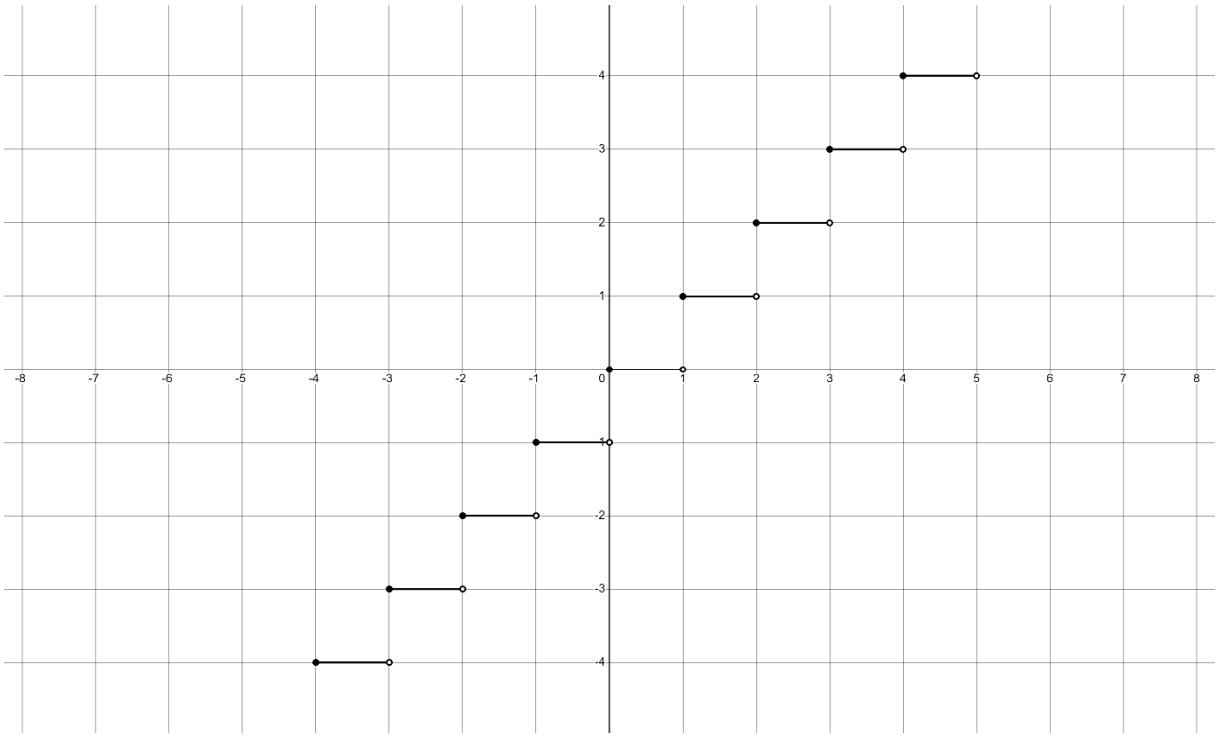
Od pohledu lze vidět, že $a(t)$ a $b(t)$ v $\operatorname{lerp}(P_j,P_{j+1},t)$ musí být tvaru:

$$\begin{aligned}\operatorname{lerp}(P_j,P_{j+1},t) &= n\cdot\left(\frac{(j+1)}{n}-t\right)\cdot P_j + n\cdot\left(t-\frac{j}{n}\right)\cdot P_{j+1}\\ &= \big((j+1)-nt\big)\cdot P_j + (nt-j)\cdot P_{j+1}\end{aligned} \tag{8}$$

Použitím zaokrouhlovacích funkcí $\operatorname{ceil}(x)$, $\operatorname{floor}(x)$ a dosazením do (8):



Obrázek 14: Funkce ceil(x)



Obrázek 15: Funkce floor(x)

Získáváme:

$$\begin{aligned}\operatorname{lerp}(P_j, P_{j+1}, t) &= (\operatorname{ceil}(nt) - nt) \cdot P_j + (nt - \operatorname{floor}(nt)) \cdot P_{j+1} \\ &= (\operatorname{ceil}(nt) - nt) \cdot P_{\operatorname{floor}(nt)} + (nt - \operatorname{floor}(nt)) \cdot P_{\operatorname{ceil}(nt)}\end{aligned}$$

$$\begin{aligned}\operatorname{polygon}_n(t) &= \operatorname{lerp}(P_j, P_{j+1}, t) = \operatorname{lerp}(P_{\operatorname{floor}(nt)}, P_{\operatorname{ceil}(nt)}, t) \\ &= (\operatorname{ceil}(nt) - nt) \cdot P_{\operatorname{floor}(nt)} + (nt - \operatorname{floor}(nt)) \cdot P_{\operatorname{ceil}(nt)}\end{aligned} \tag{9}$$

Dosazením (6) do (9) dostáváme:

$$\begin{aligned}\operatorname{polygon}_n(t) = (\operatorname{ceil}(nt) - nt) \cdot &\left[\cos\left(\operatorname{floor}(nt) \cdot \frac{2\pi}{n}\right), \sin\left(\operatorname{floor}(nt) \cdot \frac{2\pi}{n}\right)\right] \\ + (nt - \operatorname{floor}(nt)) \cdot &\left[\cos\left(\operatorname{ceil}(nt) \cdot \frac{2\pi}{n}\right), \sin\left(\operatorname{ceil}(nt) \cdot \frac{2\pi}{n}\right)\right]\end{aligned}$$

$$\operatorname{polygon}_n(t).x = (\operatorname{ceil}(nt) - nt) \cos\left(\frac{2\pi \operatorname{floor}(nt)}{n}\right) + (nt - \operatorname{floor}(nt)) \cos\left(\frac{2\pi \operatorname{ceil}(nt)}{n}\right)$$

$$\operatorname{polygon}_n(t).y = (\operatorname{ceil}(nt) - nt) \sin\left(\frac{2\pi \operatorname{floor}(nt)}{n}\right) + (nt - \operatorname{floor}(nt)) \sin\left(\frac{2\pi \operatorname{ceil}(nt)}{n}\right)$$

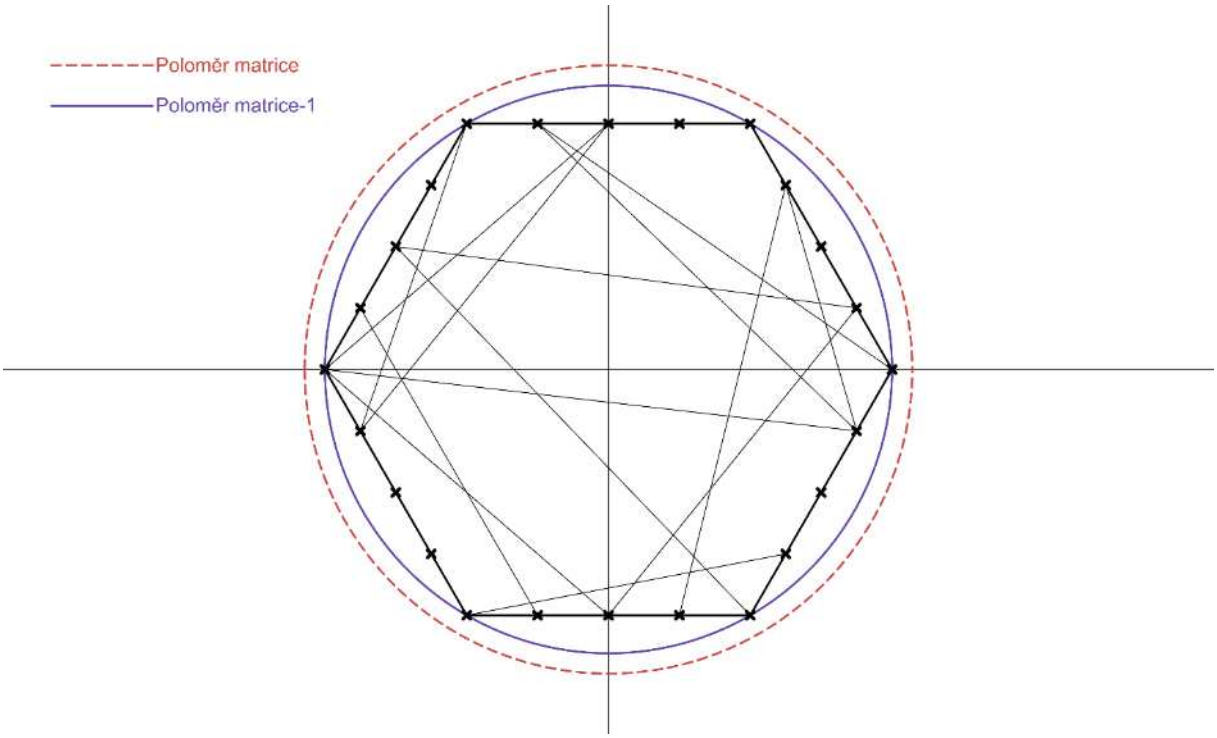
Použitím: $\operatorname{ceil}(x) - \operatorname{floor}(x) = 1 \ \forall x \in \mathbb{R}$, roznásobením a vytknutím dostáváme finální vztahy pro x a y souřadnice $\operatorname{polygon}_n(t)$:

$$\text{polygon}_n(t).x = $$
$$= \cos\left(\frac{2\pi\,\text{floor}(nt)}{n}\right) + (nt - \text{floor}(nt))\left(\cos\left(\frac{2\pi\,\text{ceil}(nt)}{n}\right) - \cos\left(\frac{2\pi\,\text{floor}(nt)}{n}\right)\right)$$

$$\text{polygon}_n(t).y = $$
$$= \sin\left(\frac{2\pi\,\text{floor}(nt)}{n}\right) + (nt - \text{floor}(nt))\left(\sin\left(\frac{2\pi\,\text{ceil}(nt)}{n}\right) - \sin\left(\frac{2\pi\,\text{floor}(nt)}{n}\right)\right)$$

Jelikož dosavadní interpolační funkce byly brány, že jejich vstupní parametrická proměnná t je v rozsahu $\langle 0; 1\rangle$ a že interpolace probíhá mezi body na jednotkové kružnici, stačí pouze škálovat body na $\text{poloměr}_{\text{matrice}} - 1$ a dostáváme:

$$\text{polygon}_n(t, \text{poloměr}_{\text{matrice}}) = (\text{poloměr}_{\text{matrice}} - 1) \cdot [\text{polygon}_n(t).x;\ \text{polygon}_n(t).y]$$



Obrázek 16: Příklad tvaru obrazu: n-úhelník

2.4.2.1 Minimální poloměr pravidelného n-úhelníku

Aby nenarážela jehla na hřebíky, musí vjet dostatečně ke středu a od středu a to znamená, že je nutno spočítat $\min(|\text{polygon}_n(t, \text{poloměr}_{\text{matrice}})|)$ a to tím, že vezmeme parciální derivaci (10) podle t a položíme rovno 0, tj.:

$$|\text{polygon}_n(t, \text{poloměr}_{\text{matrice}})| \tag{10}$$

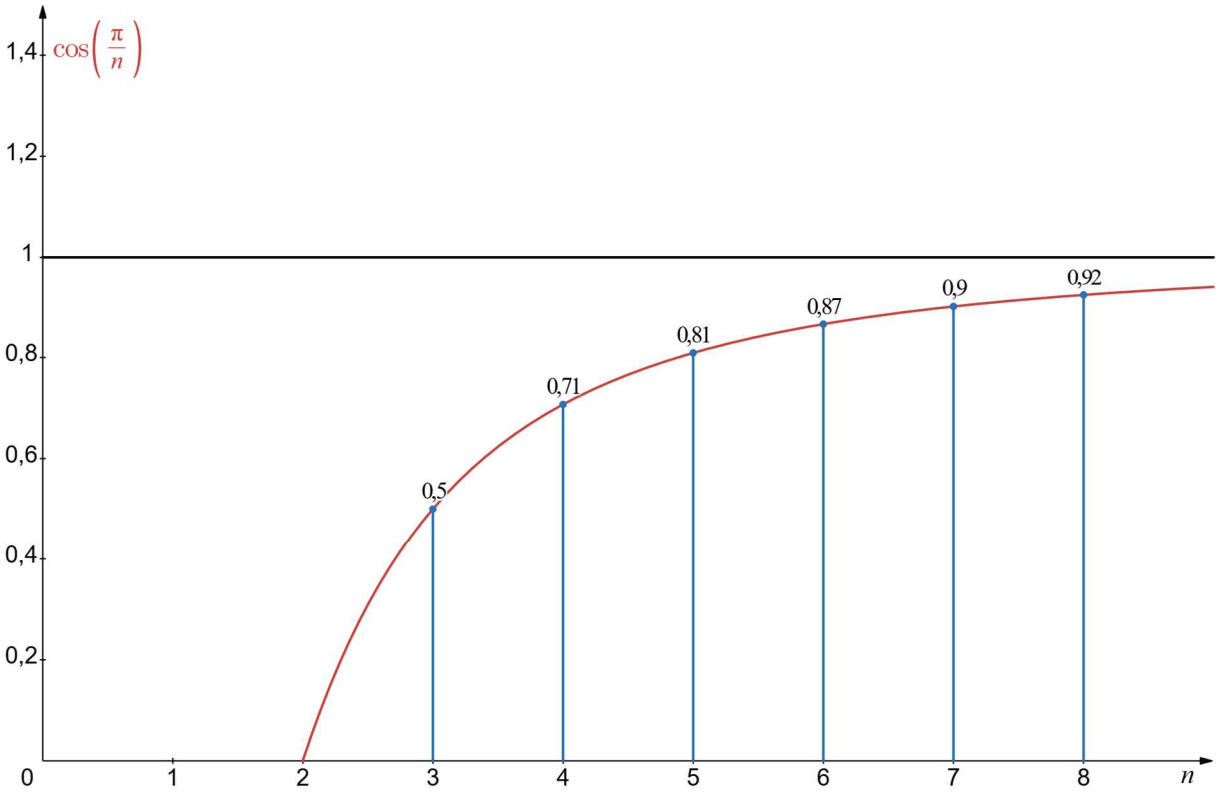
$$\frac{\partial}{\partial t}(|\text{polygon}_n(t_{\min}, \text{poloměr}_{\text{matrice}})|) = 0 \tag{11}$$

Výpočtem rovnice (11) a úpravou dostáváme $t_{\min}$ *viz* (12):

$$t_{\min} = \frac{1}{2n} + \frac{k}{n} \text{ pro } k \in \{0; 1; \ldots; n-1\} \tag{12}$$

$t_{\min}$ *viz* (12) dosadíme do (10) a dostáváme vztah (13):

$$\begin{aligned} |\text{polygon}_n(t_{\min}, \text{poloměr}_{\text{matrice}})| &= \min(|\text{polygon}_n(t, \text{poloměr}_{\text{matrice}})|) \\ &= (\text{poloměr}_{\text{matrice}} - 1) \cdot \cos\left(\frac{\pi}{n}\right) \end{aligned} \tag{13}$$



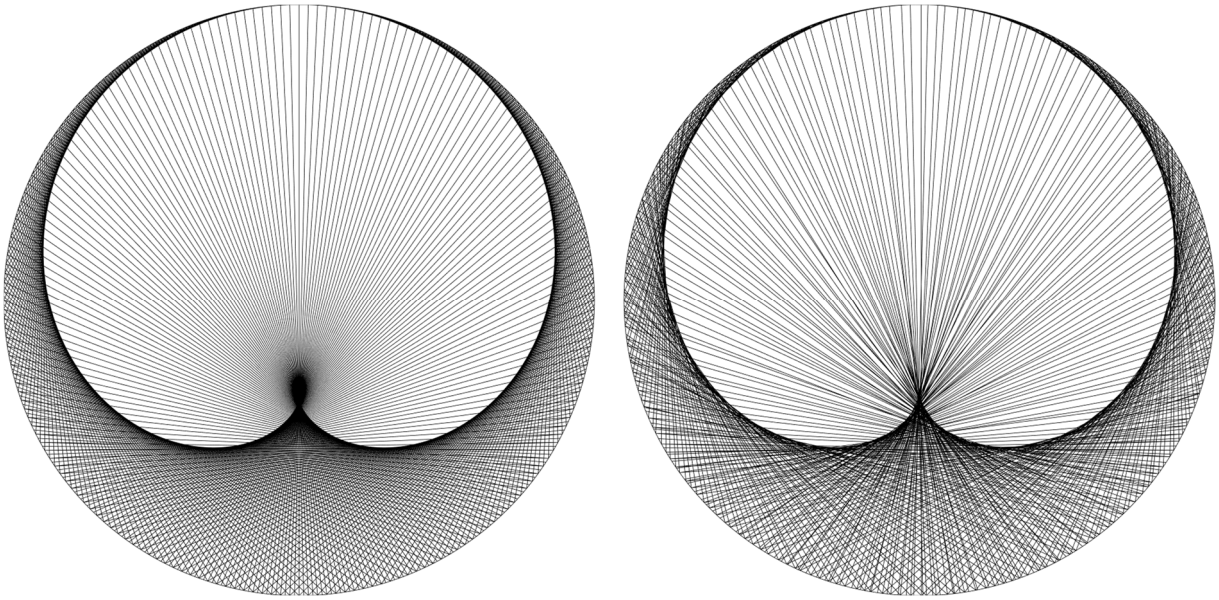
Obrázek 17: Graf minimálního poloměru hřebíků pravidelného n-úhelníku

Z grafu na *Obrázek 17* lze pozorovat to, že čím více stran n-úhelník má, tím více se jeho minimální poloměr blíží k $\text{poloměr}_{\text{matrice}} - 1$ (tj. ke kružnici).

2.5 Nepřesnosti v obrazech

Jako u všech větších projektů, je nutné předpokládat s odchylkami mezi teoretickým a praktickým řešením dané problematiky. String art, jako technika vytváření obrazců může být vizuálně působivá, ale zároveň náchylná k různým nepřesnostem. Tyto nepřesnosti mohou ovlivnit celkový vzhled výsledného obrazu a jeho estetickou kvalitu. Drobné odchylky mohou vést k deformaci vzoru a tím i celého obrazu. Přestože jsou některé nepřesnosti téměř neviditelné, v detailním pohledu mohou ovlivnit symetrii a celkový dojem z hotového obrazu. Porozumění nepřesnostem je důležité jak pro jejich

minimalizaci a dosažení co nejpřesnějšího výsledku, tak i k realističtějšímu očekávání ze strany uživatele.



Obrázek 18: Ukázka nepřesnosti – teorie / „skutečnost“

Pro můj projekt existuje několik faktorů, které budou ovlivňovat přesnost vinutí obrazů a to jsou:

1) nepřesnost navrtání otvorů pro hřebíky a to: a) úhlové odchylky mezi jednotlivými hřebíky, b) jejich vzdálenost od středu,
2) tloušťka hřebíků (jelikož hřebíky nejsou infinitesimální – nekonečně malé – body),
3) tloušťka nitě.

Nepřesnost typu 1) a) vzniká kvůli nerealizovatelnosti kompletně přesného natočení matrice na požadovaný úhel, jelikož je matrice točena krokovým motorem s konečným počtem kroků. Pro minimalizaci této nepřesnosti můžeme zvětšit poloměr matrice, jelikož poté každé posunutí otvoru pro hřebík od ideální pozice bude relativně menší oproti větší velikosti matrice, bohužel na úkor namáhaní krokového motoru podstavy větším proudem a tím pádem vyšším výkonem, jelikož je hmotnost matrice přímo úměrná druhé mocnině jejího poloměru *viz* rovnice (14).

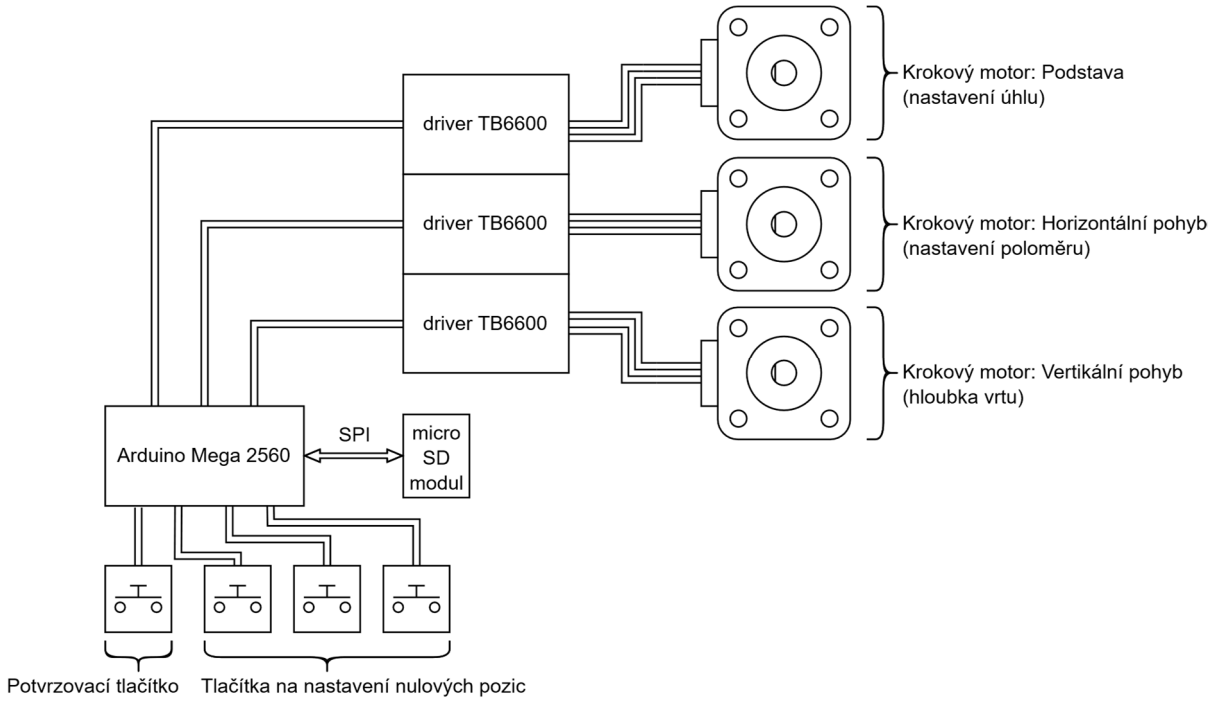
$$\begin{aligned} m_{\text{matrice}} &= V_{\text{matrice}} \cdot \rho = \pi \cdot r^2 \cdot v \cdot \rho \\ m_{\text{matrice}} &= K \cdot r^2 \end{aligned} \tag{14}$$

Dalším potlačením nepřesnosti 1) a) může být zvolení jemnějšího mikrokrokování krokového motoru v podstavě, nebo zvolení lepšího poměru převodu řemenic. Obě tato řešení vedou k vyššímu rozlišení kroků mezi jednotlivými hřebíky a to např.:

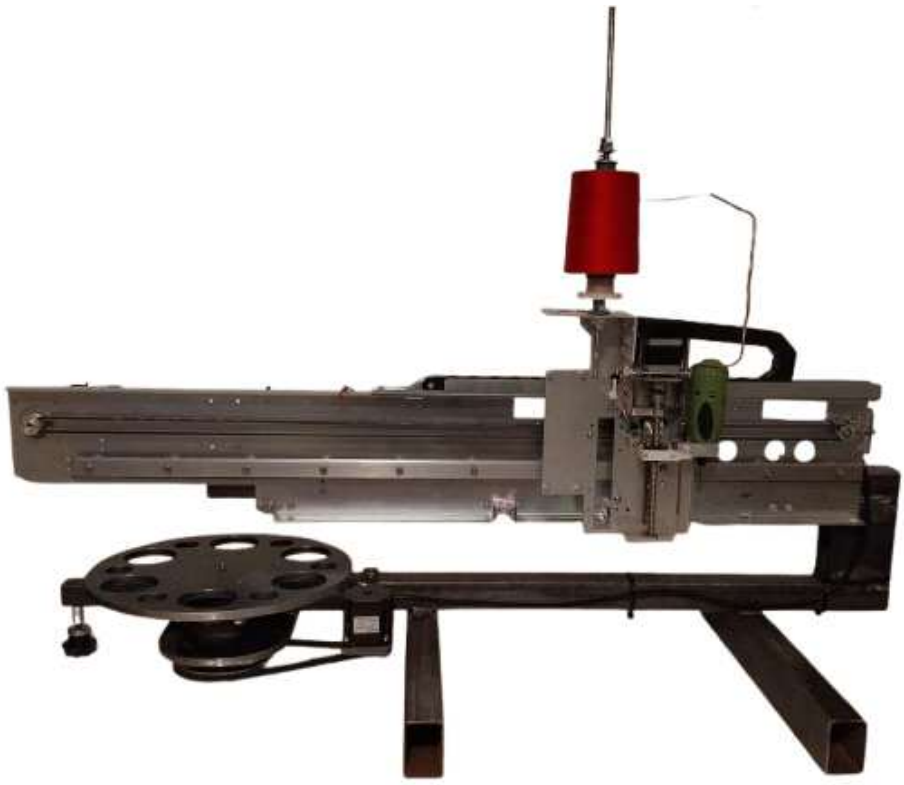
$$\text{tvar} = \text{kružnice}$$

$$\text{základní počet kroků na otáčku} = 200$$
$$\text{převod řemenic} = \frac{N_2}{N_1} = 8 \ldots \text{kde } N_{1,2} \ldots \text{počty zubů}$$
$$\text{mikrokrokování} = 16$$
$$\text{počet hřebíků} = 250$$
$$\text{úhel hřebíků} = \frac{360^\circ}{250} = 1{,}44^\circ$$
$$\text{celkový počet kroků na otáčku} = 200 \cdot 8 \cdot 16 = 25\ 600$$
$$\text{úhel kroku} = \frac{360^\circ}{25\ 600} = 0{,}0140625^\circ = 0^\circ 0{,}84375'$$
$$\frac{\text{úhel hřebíků}}{\text{úhel kroku}} = \frac{1{,}44^\circ}{0^\circ 0{,}84375'} = 102{,}8$$

3 Praktická část



Obrázek 19: Blokový diagram projektu



Obrázek 20: Výsledný projekt

3.1 Konstrukce

3.1.1 Lineární pojezd

Nejvíce integrální část mého projektu je lineární 2D pojezd údajně japonské výroby, ale nelze jakkoli dohledat jeho původ (nejspíše je ale nemocničního původu nebo CNC původu). Lineární pojezd zajišťuje jak horizontální posuv vrtačky/jehly na poloměr určený z dat přečtených z SD karty, tak vertikální pohyb vrtačky při vrtání. Jelikož byl vertikální pohyb originálně kolmý k horizontálnímu pohybu, ale otvory pro hřebíky bylo nutné vrtat pod úhlem odlišným od 90° (zvolil jsem 75°), tak bylo nutné naklonit plech s krokovým motorem. Toho jsem docílil převrtáním jednoho připevňovacího bodu v plechu a výsledný úhel odchýlení od 90° byl přibližně 15°40′. Při procesu vinutí je však vertikální pohyb nastaven kolmě. K lineárnímu pojezdu je přichycen držák nitě a držák vrtačky a jehly.

3.1.2 Upevňovací kostra

Celý projekt do sebe musel zapadat s co možná nejpřesnějšími tolerancemi a musí být dostatečně robustní. Upevňovací kostra je 30–40kilogramový rám ve tvaru podlouhlého „C“ svařeného ze 40 mm × 40 mm jäklového profilu s podstavnými podporami ze 40 mm × 60 mm jäklového profilu. K horní části „C“ je přivrtán lineární pojezd a připevněno Arduino, napájecí zdroj, microSD card modul, destička s tlačítky, drivery pro krokové motory a ventilátory pro chlazení driverů. Ke spodní části „C“ je připevněn držák krokového motoru otáčející podstavou a svařené podstavné podpory a ložisko. Ze spodní části „C“ vede vysouvací balanční noha, která brání překlopení celého stroje.

3.1.3 Ložisko v podstavě

Jelikož je dnes moderní ekologie a recyklace, tak je ložisko v podstavě sestrojeno z demontovaného náboje předního kola BMW 525 TDS. Náboj má z předešlého užití najeto 440 tisíc kilometrů a v nové roli funguje bez nejmenších problémů. Ložisko je k upevňovací kostře svařeno pomocí ocelové destičky. Ke spodu ložiska je uchycena hliníková řemenice se 72 zuby a na hřídeli krokového motoru je řemenice s 12 zuby.

Obrázek 21: Rotační část

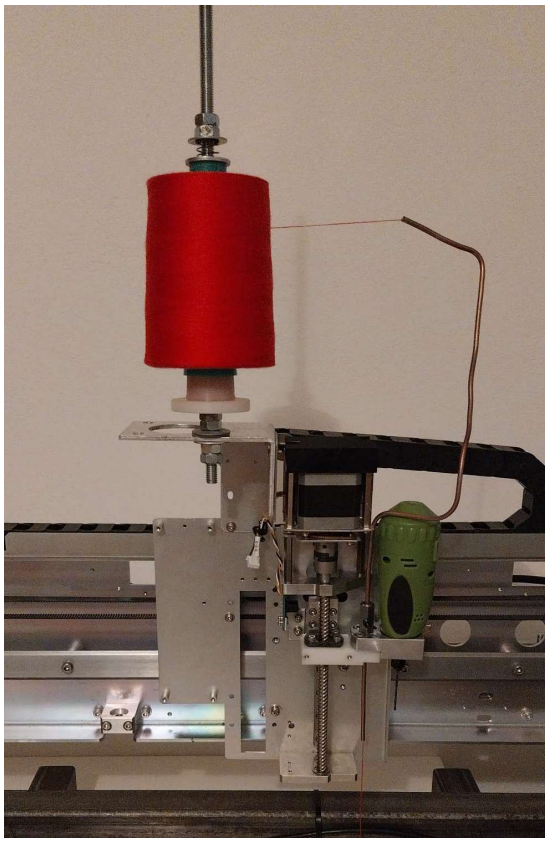
Nad ložiskem je uchycena otočná podstava s průměrem přibližně 295 mm obrobena z duralu. Skrz střed otočné podstavy je proveden zbroušený šroub M8, který slouží k vycentrování matrice – uprostřed spodní strany matrice je vždy předvrtán otvor s průměrem 8 mm a matrice tak jednoduše zapadne na centrovací kolík. Poté je ze spodní strany matrice přišroubována k otočné podstavě.

3.1.4 Držák vrtačky a jehly

Držák vrtačky a jehly je kus obrobeného hliníku, který je upevněn ke kusu plastu, který byl originálně součástí lineárního pojezdu. Uprostřed plastové části držáku je vnořena matice, skrze kterou prochází trapézový šroub přidělaný ke hřídeli krokového motoru. Trapézový šroub má průměr 8 mm a stoupání 12 mm (jedna otočka trapézového šroubu posune matici o 12 mm). Hliníková část držáku má dva otvory – jeden otvor slouží k přichycení vrtačky a druhým otvorem vede šestihranná ocel s vyvrtaným středem, obrobeným závitem a postranním otvorem pro imbusový šroub. Vyvrtaným středem šestihranné oceli vede měděná trubička z ledničky, která je k šestihranné oceli připevněna imbusovým šroubem. Měděná trubička slouží jako jehla – vede skrz ni niť.

3.1.5 Držák nitě

Držák nitě je M8 šroub s plastovým válcem. Ve válci je ložisko, aby se niť vyvíjela hladce. Nad špulí nitě je malá pružinka, která působí proti nadbytečnému vyvíjení nitě. Čím více je pružinka stlačená, tím více tře a brzdí špuli, aby niť byla vždy napnutá.



Obrázek 22: Držák vrtačky a jehly, držák nitě

3.2 Návod k použití/princip funkce

1) Nastavení/kontrola parametrů: poloměr matrice, tloušťka matrice, proudová nastavení driverů, režimy krokování driverů, …

2) Zapsání parametrů do „param.txt“ na microSD kartě ve formátu:
   tvar = $\begin{cases} 0 & \text{kružnice} \\ \geq 3 & \text{n} - \text{úhelník} \end{cases}$ (celé číslo)
   pocet_hrebiku = 250 (celé číslo)
   polomer_matrice = 40.2 (desetinné číslo – pokud celé číslo, tak napsat např. 55.0)

3) Zapsání pořadí propojení jednotlivých hřebíků do „poradi.txt“ na microSD kartě ve formátu:
   3000,0,159,251,56,248,55,..... ,176

   délka pořadí je **3000**

   délka celé posloupnosti čísel je **3001**

   První číslo např. **3000** udává počet propojení jednotlivých hřebíků (délka cyklu vnitřního stavu „vinutí“)

4) Zapojení celého zařízení do sítě 240 V, 50 Hz

5) Vsunutí microSD karty – z karty se automaticky přečtou parametry

6) Po přečtení microSD karty se vnitřní stav přepne do „příprava před vrtáním“
7) Příprava před vrtáním:
   A) Vložení matrice na kolík, přivrtání matrice k podstavnému disku
   B) Kontrola vrtáku
   C) Nulování pozic bílými tlačítky:
      a) Podstava – otočení na libovolnou pozici
      b) Lineární pohyb – hrot vrtáku na poloměr matrice
      c) Vrtání – hrot vrtáku skoro na povrch matrice
   D) Zapnutí vrtačky
   E) Zmáčknutí zeleného tlačítka potvrdí přechod na stavu „vrtání“

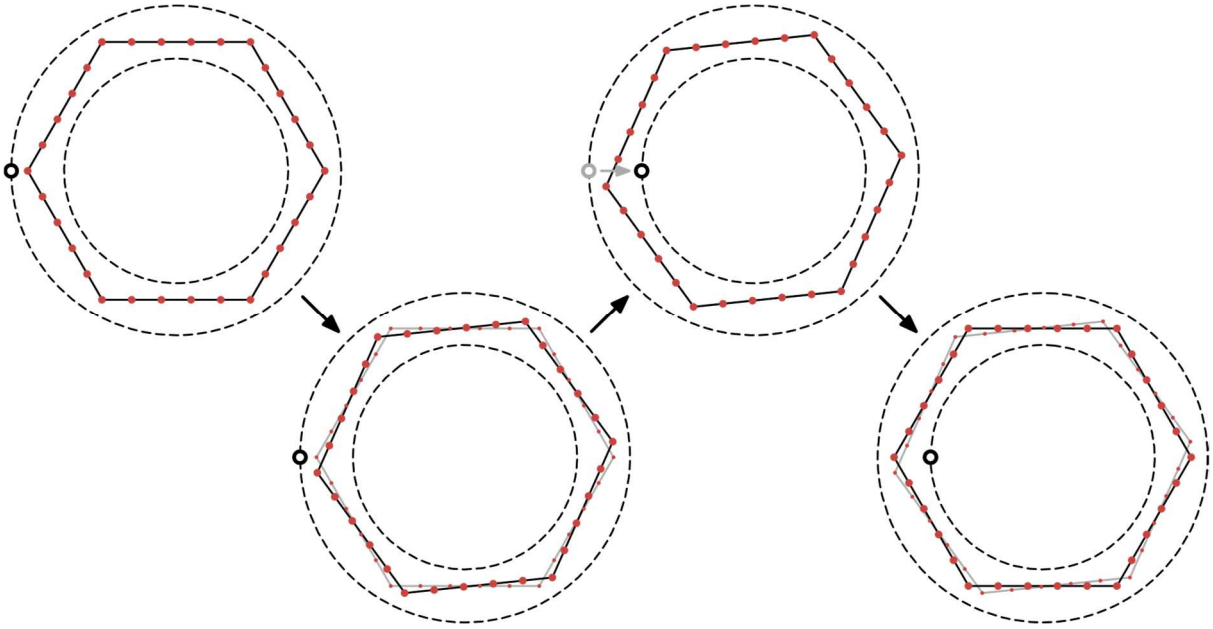
   poznámka: Když jsou v daném okamžiku krokové motory v klidu a enable piny driverů jsou na úrovni HIGH, drivery do krokových motorů pouští udržovací proud, a proto musíme před ručním hýbáním mechanismů enable piny nastavit na LOW. To je zajištěno tím, že první stisknutí tlačítka příslušného motoru vypne udržovací

proud (což dovolí ruční posunutí mechanismu na nulovou pozici) a druhé stisknutí tlačítka nastaví současnou pozici jako nulovou a opět zapne udržovací proud.

8) Vrtání = cyklus: Vrták se zvedne na výšku 1 cm nad matricí; matrice se otočí o 1 hřebík; vrtačka se posune dolu a vyvrtá otvor. Cyklus se opakuje až do dovrtání posledního otvoru pro hřebík a poté přejde do stavu „čekání na hřebíky“

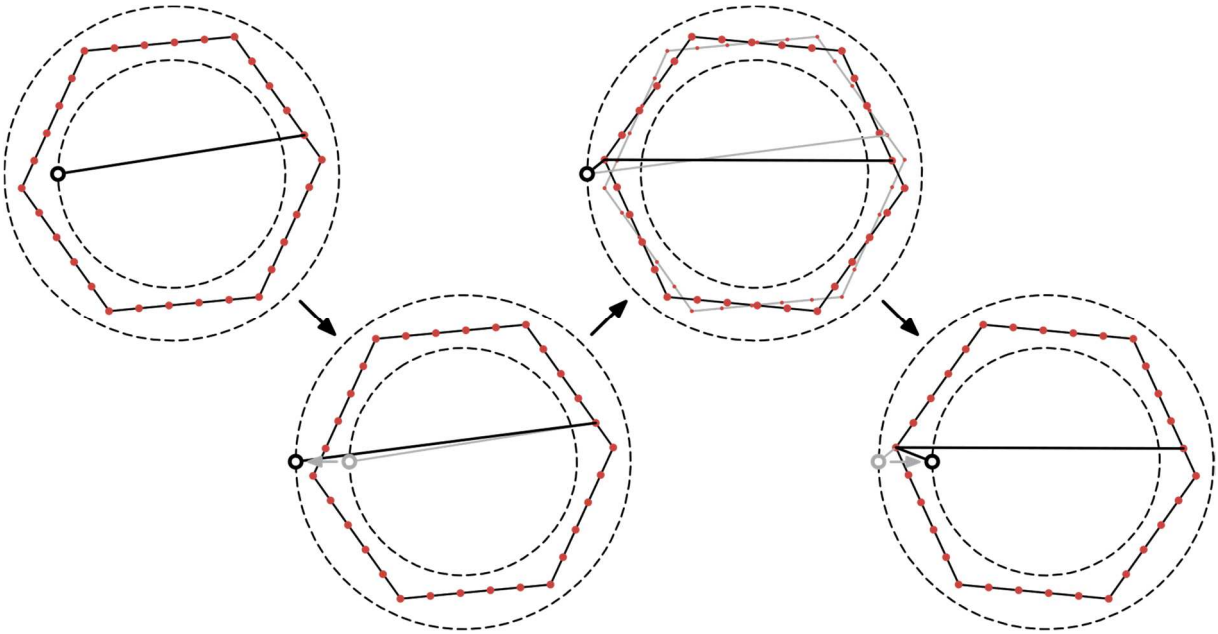
9) Čekání na hřebíky (příprava před vinutím):
   A) Vypneme vrtačku
   B) Zasuneme hřebíky do vyvrtaných otvorů
   C) Připevníme niť na nultý hřebík
   D) Nulování pozic bílými tlačítky v pořadí:
      a) Podstava – otočení matrice tak, že nultý hřebík bude přesně pod jehlou
      b) Lineární pohyb – hrot jehly na poloměr matrice
      c) Vrtání – hrot jehly přibližně do půlky výšky hřebíku nad matricí
   F) Zmáčknutí zeleného tlačítka potvrdí přechod na stavu „vinutí“

10) Vinutí:
   A) Jelikož po předchozím kroku jehla leží přímo za 0. hřebíkem na vnější straně obrazu a chceme, aby proces vinutí začínal s jehlou na vnitřní straně obrazu, tak se jednorázově provede následující: matrice se pootočí o půl hřebíku, jehla vjede mezi dvěma hřebíky ke středu a matrice se otočí zpět na nulový úhel.



Obrázek 23: Jednorázový postup před vinutím

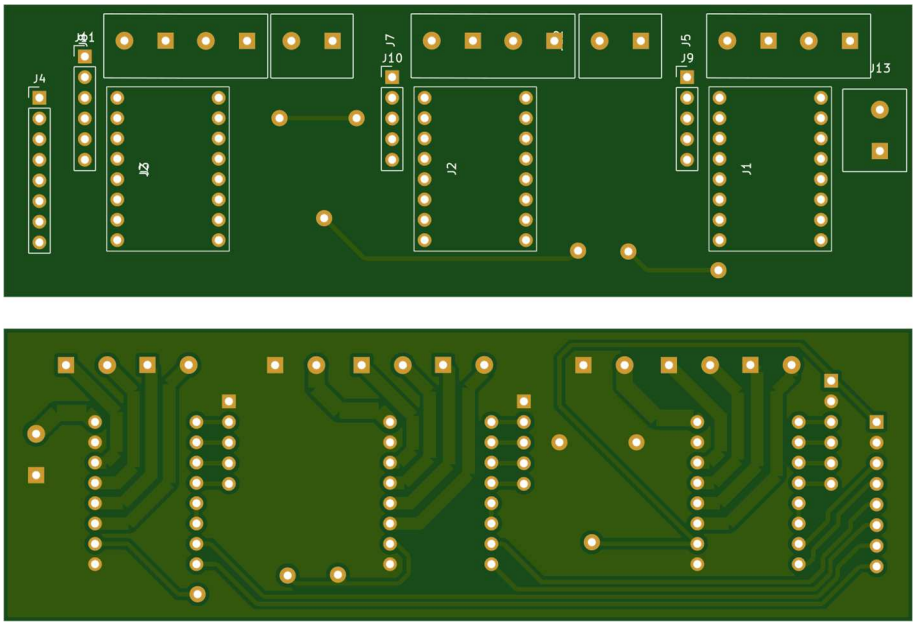
B) Zbytek vinutí = cyklus: Matrice je otočena mezi dva hřebíky a to hřebík, který je další v pořadí (hřebík $\boldsymbol{j}$) a hřebík, který je hned vedle něj (hřebík $\boldsymbol{j-1}$) tím nejefektivnějším způsobem – pootočení k dalšímu hřebíku v pořadí je vždy méně než 180° (např. Pokud je celkový počet hřebíků 200, současný hřebík je hřebík 180 a máme se otočit ke hřebíku 20, je efektivnější otočit se na hřebík „220“, což je posun o 40 hřebíků (místo 160 hřebíků), jelikož 220 je kongruentní s 20 modulo 200: $220 \equiv 20 \pmod{200}$); jehla vyjede ven od středu mezi dvěma hřebíky ($\boldsymbol{j}$ a $\boldsymbol{j-1}$); matrice se otočí mezi hřebík $\boldsymbol{j}$ a hřebík $\boldsymbol{j+1}$; jehla vjede mezi hřebíky ($\boldsymbol{j}$ a $\boldsymbol{j+1}$) zpět ke středu a tím je omotán hřebík $\boldsymbol{j}$.



Obrázek 24: Cyklus postupu pří vynutí

3.3 Prototypová deska plošného spoje

Abych mohl celou dobu výroby zkoušet, zda projekt funguje, ale zároveň se neomezit z konstrukčních hledisek, musel jsem si vyrobit prototypové zapojení celého projektu, které mělo za úkol pouze propojit všechny členy projektu – Arduino, drivery a krokové motory. Mohl jsem zvolit breadboard (nepájivé pole) nebo prototypovou DPS. Ze začátku jsem předpokládal, že drivery, které budu používat, budou „destičkového“ (pro nedostatek lepšího slova) typu. Pro čistě testovací účely ale DPS stačila a posvítila tak na nutné změny. Z testování vyplynulo, že budou potřeba drivery robustnější a méně náchylné poruchovým veličinám – drivery TB6600.



Obrázek 25: Prototypová DPS – horní / spodní strana

3.4 Popis Arduino kódu

3.4.1 Knihovny

```
#include <AccelStepper.h>
#include <SPI.h>
#include <SdFat.h>
#include <math.h>
```

Knihovna *AccelStepper* je pro ovládání krokových motorů pomocí Arduina

Knihovna *SPI* je pro SPI komunikaci s SD kartou přes SD card modul

Knihovna *SdFat* překládá uživatelem psaný kód na SPI příkazy a je pro práci s SD kartou jako čtení a ukládání dat

Knihovna *math* je rozšířením základních matematických dovedností Arduina

3.4.2 Konstantní parametry

```
#define uhel_vrtani_stupne 74.333
#define buff_hrebiku 1.5
#define linPohyb_krok_cm 200/5
#define vrtani_krok_cm 200/1.2
#define mikrokrokovani_podstava 8
#define mikrokrokovani_linPohyb 8
#define mikrokrokovani_vrtani 8

long celk_pocet_kroku_podstava = 200*6*mikrokrokovani_podstava;
```

uhel_vrtani_stupne je úhel naklonění vrtáku oproti povrchu matrice

buff_hrebiku je vzdálenost hřebíků od poloměru matrice

linPohyb_krok_cm je počet kroků, který musí krokový motor lineárního pohybu provést, aby se vrták (nebo jehla) posunul o 1 cm

vrtani_krok_cm je počet kroků, který musí krokový motor vrtání provést, aby se vrták posunul o 1 cm

mikrokrokovani_podstava nastavení krokovacího režimu driveru, který ovládá motor v podstavě

mikrokrokovani_linPohyb nastavení krokovacího režimu driveru, který ovládá motor lineárního pohybu

mikrokrokovani_vrtani nastavení krokovacího režimu driveru, který ovládá motor vrtání
celk_pocet_kroku_podstava je spočten $200 \cdot 6 \cdot$ mikrokrokovani_podstava, kde:

200 .....základní počet kroků na jednu otáčku krokového motoru

6.....převod ozubených kol

3.4.3 Pomocné proměnné pro proces vinutí

`BOD f;` Proměnná, která udržuje x,y souřadnice hřebíku, který má být omotán.

`BOD f_minus_1;` Proměnná, která udržuje x,y souřadnice hřebíku, který je hned vedle hřebíku, který má být omotán.

`BOD f_plus_1;` Proměnná, která udržuje x,y souřadnice hřebíku, který je hned vedle hřebíku, který má být omotán.

`float min_polomer;` Proměnná, která určuje, jak moc ke středu má zajet lineární pojezd.

`float max_polomer;` Proměnná, která určuje, jak moc od středu má zajet lineární pojezd.

`float soucasny_uhel=0;` Úhel, na který je v daný okamžik natočena matrice.
`float chteny_uhel_minus_05;` Úhel daný průměrem argumentů "f"
a "f_minus_1"
`float chteny_uhel_plus_05;` Úhel daný průměrem argumentů "f" a "f_plus_1"

`float optimalni_pootoceni_minus_05;` Proměnná, která určuje, o kolik radiánů se má matrice otočit, aby byla natočena na úhel "chteny_uhel_minus_05".

`float optimalni_pootoceni_plus_05;` Proměnná, která určuje, o kolik radiánů se má matrice otočit, aby byla natočena na úhel "chteny_uhel_plus_05".

`float optimalni_pootoceni;` Proměnná, která se nastaví na "optimalni_pootoceni_minus_05" nebo "optimalni_pootoceni_plus_05" tak, aby byl pohyb matrice co nejkratší.

3.4.4 Definice krokových motorů pomocí accelstepper knihovny

```
AccelStepper motor_podstava(AccelStepper::DRIVER, stepPinPodstava,
dirPinPodstava);
AccelStepper motor_linpohyb(AccelStepper::DRIVER, stepPinLinPohyb,
dirPinLinPohyb);
AccelStepper motor_vrtani(AccelStepper::DRIVER, stepPinVrtani,
dirPinVrtani);
```

Definice motoru je pomocí *AccelStepper* ve tvaru:

AccelStepper motor(Typ driveru, STEP pin, DIR pin);

3.4.5 Matematické funkce

3.4.5.1 Deklarace struktury „bod“ s x,y souřadnicemi

```
  struct BOD{
    float x;
    float y;
  };
```

3.4.5.2 Funkce, která přemění x,y souřadnice na bod

```
BOD xy_na_bod(float x, float y){
  BOD bod;
  bod.x = x;
  bod.y = y;

  return bod;
}
```

3.4.5.3 Funkce pro čtení absolutního argumentu bodu

```
float argument_rad_nula_2pi(BOD bod){
  float pomocna;
  if (atan2(bod.y,bod.x)<0.0){pomocna=atan2(bod.y,bod.x)+2.0*PI;}
  else{pomocna=atan2(bod.y,bod.x);}
  return pomocna;
}
```

Funkce *atan2(y ,x)* je z knihovny *math* a vrací argument bodu $[x; y]$ v rozsahu $(-\pi; +\pi\rangle$. Jelikož ale potřebuji úhly v rozsahu $\langle 0; 2\pi)$, tak pro záporný obor hodnot funkce *atan2* přičítám 2π, což úhel z intervalu $(-\pi; 0)$ „otočí“ do intervalu $(\pi; 2\pi)$.

3.4.5.4 Funkce pro čtení modulu bodu

```
float modul(BOD bod){
  return sqrt(bod.x*bod.x+bod.y*bod.y);
}
```

Výpočet modulu je Pythagorovou větou: $|[x; y]| = \sqrt{x^2 + y^2}$

3.4.5.5 Funkce pro optimální pootočení matrice

```
float optimalniPootoceni(float soucasny_uhel, float chteny_uhel){

  float rozdil_uhlu = chteny_uhel - soucasny_uhel;

  if (rozdil_uhlu > PI){
    rozdil_uhlu -= 2.0*PI;
  }else if (rozdil_uhlu < -PI){
    rozdil_uhlu += 2*PI;
  }

  return rozdil_uhlu;
}
```

Nejdříve je ve funkci *optimalniPootoceni* vypočten rozdíl úhlu současného a úhlu, na který se má matrice otočit. Poté funkce určí, zda je absolutní hodnota úhlového rozdílu větší než π – pokud ano, tak od úhlového rozdílu přičte nebo odečte 2π a to docílí toho, že se matrice nebude zbytečně otáčet o více než 180°

3.4.6 Parametrické funkce

Parametrické funkce v této části kódu popisují x,y souřadnice hřebíků na matrici pro $t \in \langle 0; 1)$ a s maximálním poloměrem 1. Tyto parametrické funkce byly odvozeny v kapitole 2.4.

3.4.6.1 Kružnice

```
BOD kruznice(float t){

  BOD bod;
  bod.x = cos(2.0*PI*t);
  bod.y = sin(2.0*PI*t);

  return bod;
}
```

3.4.6.2 Pravidelný n-úhelník

```
BOD polygon(float t, int n){

  BOD bod;
  bod.x = cos((2.0*PI*floor(n*t))/n)+((n*t-
floor(n*t)))*(cos((2.0*PI*ceil(n*t))/n)-cos((2.0*PI*floor(n*t))/n));

  bod.y = sin((2.0*PI*floor(n*t))/n)+((n*t-
floor(n*t)))*(sin((2.0*PI*ceil(n*t))/n)-sin((2.0*PI*floor(n*t))/n));

  return bod;
}
```

3.4.6.3 Funkce, která rozhodne, zda se jedná o kružnici, či n-úhelník

```
BOD funkce_vybrana(float t, int n, float polomer_matrice_cm){

  BOD bod;
  if (n == 0){
        bod.x = (polomer_matrice_cm-buff_hrebiku)*(kruznice(t).x);
        bod.y = (polomer_matrice_cm-buff_hrebiku)*(kruznice(t).y);
  }else{
        bod.x = (polomer_matrice_cm-buff_hrebiku)*(polygon(t, n).x);
        bod.y = (polomer_matrice_cm-buff_hrebiku)*(polygon(t, n).y);
    }

  return bod;
}
```

Tato funkce posoudí, zda je $n = 0$ – pokud ano, tak vrátí body na kružnici a pokud ne, tak vrátí body na pravidelném n-úhelníku. Funkce vrací body, které už jsou škálovány na maximální vzdálenost poloměr matrice − 1

3.4.6.4 Funkce, která vrátí minimum všech vzdáleností bodů od středu

```
float min_polomer_funkce(int n, float polomer_matrice_cm){

  if (n == 0){return (polomer_matrice_cm-buff_hrebiku);}
  else       {return (polomer_matrice_cm-buff_hrebiku)*cos(PI/n);}
}
```

Tato funkce opět posoudí, zda je $n = 0$ a podle toho vrátí příslušnou minimální vzdálenost od středu.

3.4.7 Funkce pro ovládání krokových motorů

3.4.7.1 Funkce pro nulování pozic krokových motorů

```
void nulovani_pozic(){
  if (digitalRead(nulovaciPinPodstava) == LOW){
Detekce stisknutí nulovacího tlačítka motoru, který otáčí podstavou
while(digitalRead(nulovaciPinPodstava) == LOW){delay(50);}
Čekání na puštění nulovacího tlačítka podstavy (ošetření zákmitů tlačítka)
      digitalWrite(enablePinPodstava, LOW);
Vypnutí driveru motoru, který otáčí podstavou = vypnutí udržovacího proudu
      delay(50);
Ošetření zákmitů tlačítka
while(digitalRead(nulovaciPinPodstava) == HIGH){delay(50);}
Čekání na další zmáčknutí tlačítka (ošetření zákmitů tlačítka)
      digitalWrite(enablePinPodstava, HIGH);
Zapnutí driveru motoru, který otáčí podstavou = zapnutí udržovacího proudu
      delay(50);
Ošetření zákmitů tlačítka
      motor_podstava.setCurrentPosition(0);
Nastavení nulové pozice
    }
while(digitalRead(nulovaciPinPodstava) == LOW){delay(50);}
Čekání na puštění nulovacího tlačítka podstavy (ošetření zákmitů tlačítka)
.....
}
```

poznámka: Stejný postup je i pro ostatní krokové motory až na „nulovou“ pozici lineárního pojezdu, která je nastavena na polomer_matrice.

3.4.7.2 Funkce pro posunutí lineárního pojezdu na zadaný poloměr

```
void posunuti_linpohybu_na_polomer_cm(float polomer_cm){

  motor_linpohyb.moveTo(round((float)polomer_cm*linPohyb_krok_cm*
*mikrokrokovani_linPohyb));   Nastavení cílového počtu kroků

  motor_linpohyb.runToPosition();   Provedení kroků nastavených v moveTo()
}
```

3.4.7.3 Funkce pro otočení podstavy/matrice/obrazu na absolutní úhel

```
void otoceni_podstavy_na_rad(float na_uhel_rad){

  motor_podstava.moveTo(round((float)na_uhel_rad/(2.0*PI)*celk_
pocet_kroku_podstava));   Nastavení cílového počtu kroků

  motor_podstava.runToPosition();   Provedení kroků nastavených v moveTo()
}
```

3.4.7.4 Funkce pro otočení podstavy/matrice/obrazu o relativní úhel

```
void otoceni_podstavy_o_rad(float o_uhel_rad){

    motor_podstava.move(round((float)o_uhel_rad/(2.0*PI)*celk_
pocet_kroku_podstava));

    while (motor_podstava.isRunning()){
      motor_podstava.run();
      }
}
```

Tato funkce nemůže použít *runToPosition*, protože nefunguje pro *move*, proto funkce využívá *isRunning*, což je binární proměnná říkající, zda ještě motoru zbývají kroky, které má provést.

3.4.7.5 Funkce pro vrtání

```
void vrt_cm(float hloubka_cm){

  motor_vrtani.moveTo(round((float)hloubka_cm/sin((float)uhel_vrtani
_stupne*PI/180)*vrtani_krok_cm*mikrokrokovani_vrtani));

  motor_vrtani.runToPosition();
  delay(250);
```

```
  motor_vrtani.moveTo(-
round((float)hloubka_cm/sin((float)uhel_vrtani_stupne*PI/180)*vrtani
_krok_cm*mikrokrokovani_vrtani));

  motor_vrtani.runToPosition();
}
```

Tato funkce nejdříve posune vrták pod povrch matrice, počká 250 milisekund a poté vyjede s vrtákem zpět nad povrch matrice.

3.4.7.6 Funkce pro omotání hřebíku

```
void omotej_hrebik(int jaky_hrebik, int pocet_hrebiku, int tvar,
float polomer_matrice_cm){

  f = funkce_vybrana((float)(jaky_hrebik)/pocet_hrebiku,tvar,
(float)polomer_matrice_cm);

  f_minus_1 = funkce_vybrana((float)(jaky_hrebik-1)/pocet_hrebiku,
tvar,(float)polomer_matrice_cm);

  f_plus_1 = funkce_vybrana((float)(jaky_hrebik+1)/pocet_hrebiku,
tvar,(float)polomer_matrice_cm);

  max_polomer = modul(f) + 2.0;

  chteny_uhel_minus_05 = argument_rad_nula_2pi(xy_na_bod(
(float)(f.x+f_minus_1.x)/2.0,(float)(f.y+f_minus_1.y)/2.0));


  chteny_uhel_plus_05 = argument_rad_nula_2pi(xy_na_bod(
(float)(f.x+f_plus_1.x)/2.0,(float)(f.y+f_plus_1.y)/2.0));

  optimalni_pootoceni_minus_05 =
optimalniPootoceni((float)soucasny_uhel,
(float)chteny_uhel_minus_05);

  optimalni_pootoceni_plus_05 =
optimalniPootoceni((float)soucasny_uhel,
(float)chteny_uhel_plus_05);

  if(abs(optimalni_pootoceni_minus_05)<abs(optimalni_pootoceni_
plus_05)){

optimalni_pootoceni=optimalni_pootoceni_minus_05;
```

```
otoceni_podstavy_o_rad((float)optimalni_pootoceni);
posunuti_linpohybu_na_polomer_cm((float)max_polomer);

optimalni_pootoceni=optimalniPootoceni((float)chteny_uhel_minus_05,
(float)chteny_uhel_plus_05);

otoceni_podstavy_o_rad((float)optimalni_pootoceni);
posunuti_linpohybu_na_polomer_cm((float)min_polomer);

soucasny_uhel=chteny_uhel_plus_05;
}else{

optimalni_pootoceni=optimalni_pootoceni_plus_05;

otoceni_podstavy_o_rad((float)optimalni_pootoceni);
posunuti_linpohybu_na_polomer_cm((float)max_polomer);

optimalni_pootoceni=optimalniPootoceni((float)chteny_uhel_plus_05,
(float)chteny_uhel_minus_05);

otoceni_podstavy_o_rad((float)optimalni_pootoceni);
posunuti_linpohybu_na_polomer_cm((float)min_polomer);

soucasny_uhel=chteny_uhel_minus_05;
  }
}
```

Tato funkce nejdříve aktualizuje proměnné, které uchovávají informace o současném natočení, cílových úhlech a optimálních pootočeních. Poté určí, v jakém směru se bude matrice otáčet tak, aby dorazila ke chtěnému hřebíku co nejrychleji. Poté otočí matricí tak, že jehla bude mezi chtěným hřebíkem a hřebíkem hned vedle něj. Jehlou posune mezi dvěma hřebíky ven (na *max_polomer*), matrice se pootočí mezi další dva hřebíky a jehlou posune zpět ke středu (na *min_polomer*).

3.4.8 Práce s SD kartou

3.4.8.1 Deklarace SD karty a složek

```
SdFat SD;
File myFile1;
File myFile2;
```

3.4.8.2 Funkce pro načtení parametrů ze souboru

```
void precteniParametru(const char *filename){
    if(!SD.begin(SD_CS)){
        return;
    }

    myFile2 = SD.open(filename, FILE_READ);
    if(!myFile2){
        return;
    }

    char buffer[50];
    int index = 0;

    while(myFile2.available()){
        char c = myFile2.read();
        if(c == '\n' || index >= 49){

            buffer[index] = '\0';
            parseLine(String(buffer));
            index = 0;
        }else{
            buffer[index++] = c;
        }
    }
    myFile2.close();
}
```

Funkce nejdříve zkontroluje, zda je chip select pin aktivní a zda je složka dosažitelná – pokud ne, funkce okamžitě skončí, ale pokud ano, tak přečte celý obsah složky a uloží ho do proměnné *c*. Poté čte buffer dokud nenarazí na další řádek (\n) nebo dokud není buffer plný a ukončí string (\0). Poté předá načtený řádek do funkce *parseLine* a vynuluje index, aby mohl být načten další řádek. Po zpracování celého obsahu složky složku zavře.

3.4.8.3 Funkce pro parsování jednoho řádku

```
void parseLine(String line) {
    int equalIndex = line.indexOf('=');
    if (equalIndex == -1) return;

    String key = line.substring(0, equalIndex);
    String value = line.substring(equalIndex + 1);

    key.trim();
    value.trim();
```

```
    if(key == "tvar"){
        tvar = value.toInt();
    }else if(key == "pocet_hrebiku"){
        pocet_hrebiku = value.toInt();
    }else if(key == "polomer_matrice"){
        polomer_matrice_cm = value.toFloat();
    }
}
```

Funkce nejdříve hledá znak „=“ a pokud ho nenajde, tak celý řádek ignoruje a ukončí svoji funkci. Poté rozdělí řádek na klíč a hodnotu podle toho, na jaké straně „=“ se nacházejí, odřízne od nich mezery a poté podle přečteného klíče nastaví příslušnou proměnnou na přečtenou hodnotu.

3.4.8.4 Funkce pro čtení pořadí ze souboru

```
int precteniPoradi(const char *filename, int i) {

    File myFile1 = SD.open(filename, FILE_READ);
    if(!myFile1){
        return;
    }

    int index = 0;
    String number = "";

    while(myFile1.available()){
        char c = myFile1.read();
          if(c == ',' || c == '\n'){
             if (index == i) {
                myFile1.close();
                return number.toInt();
              }
            number = "";
            index++;
        }else{
            number += c;
          }
      }
      myFile1.close();
}
```

Funkce zkontroluje, zda se dá soubor otevřít – pokud ne, tak se ukončí. Poté hledá znaky a to buď „,“ nebo nový řádek (\n) a postupně zvyšuje hodnotu *index* dokud není na zadaném pořadí – pokud ano, tak funkce vrátí číslo v zadaném pořadí a zavře soubor.

3.4.9 Enumerace stavů

```
int SOUCASNY_STAV = CEKANI_SD;
enum STAV{
  CEKANI_SD,    Čekání na vsunutí SD karty a zahájení komunikace
  CTENI_SD,  Čtení parametrů obrazu z SD karty
  PRIPRAVA_PRED_VRTANIM,
  VRTANI,    Vrtání otvorů pro hřebíky
  CEKANI_NA_HREBIKY,  Čekání na vsunutí hřebíků do vyvrtaných otvorů
  VINUTI,    Vynutí obrazu
  KONEC,    Konec
};
```

Zde je pomocí funkce *enum* každému stavu přiřazeno celé číslo od 0 (0, 1, 2, …).

```
void setup() {
  Serial.begin(9600);
```

Nastavení režimů pinů

```
  pinMode(pokracovaciPin, INPUT_PULLUP);
  pinMode(nulovaciPinPodstava, INPUT_PULLUP);
  pinMode(nulovaciPinLinPohyb, INPUT_PULLUP);
  pinMode(nulovaciPinVrtani, INPUT_PULLUP);
  pinMode(enablePinLinPohyb, OUTPUT);
  pinMode(enablePinPodstava, OUTPUT);
  pinMode(enablePinVrtani, OUTPUT);
```

Definice parametrů krokových motorů

```
  motor_podstava.setMaxSpeed(2000);    Maximální rychlost – 2000 kroků/s
  motor_podstava.setAcceleration(500);    Zrychlení – 500 kroků/s^2
  motor_podstava.setMinPulseWidth(50);    Minimální délka impulzů – 50 µs

  motor_linpohyb.setMaxSpeed(2000);
  motor_linpohyb.setAcceleration(500);
  motor_linpohyb.setMinPulseWidth(50);

  motor_vrtani.setMaxSpeed(2000);
  motor_vrtani.setAcceleration(500);
  motor_vrtani.setMinPulseWidth(50);

  digitalWrite(enablePinLinPohyb, LOW);
  digitalWrite(enablePinPodstava, LOW);
  digitalWrite(enablePinVrtani, LOW);
}
```

```
void loop(){
switch (SOUCASNY_STAV){

    case PRIPRAVA_PRED_VRTANIM: {

            while(digitalRead(pokracovaciPin) ==
HIGH){nulovani_pozic();}

            SOUCASNY_STAV=VRTANI;
            break;
            }

    case CEKANI_NA_HREBIKY: {

            digitalWrite(enablePinLinPohyb, LOW); Vypnutí driverů
            digitalWrite(enablePinPodstava, LOW);
            digitalWrite(enablePinVrtani, LOW);

            while(digitalRead(pokracovaciPin) ==
HIGH){nulovani_pozic();}

            SOUCASNY_STAV=VINUTI;
            break;
            }

    case KONEC: {

            digitalWrite(enablePinLinPohyb, LOW); Vypnutí driverů
            digitalWrite(enablePinPodstava, LOW);
            digitalWrite(enablePinVrtani, LOW);
            while(digitalRead(pokracovaciPin) == HIGH){}
            SOUCASNY_STAV=CEKANI_SD;
            break;
            }
  }

  switch(SOUCASNY_STAV)
  {
    case CEKANI_SD: {

            if(!SD.begin(SD_CS)){
              SOUCASNY_STAV=CTENI_SD;
            }else{
              delay(1000);
              SOUCASNY_STAV=CTENI_SD;
            }
            break;
            }
```

```
    case CTENI_SD: {
            if (!SD.begin(SD_CS)){
              SOUCASNY_STAV = CEKANI_SD;
            }else{
              precteniParametru("param.txt");
              SOUCASNY_STAV = PRIPRAVA_PRED_VRTANIM;
            }
            break;
            }

    case VRTANI: {
            motor_vrtani.moveTo(-
round((float)1.0/sin((float)uhel_vrtani_stupne*PI/180)*vrtani_krok_
cm*mikrokrokovani_vrtani));

            motor_vrtani.runToPosition();
```

Cyklus projde všemi hodnotami od 0 do pocet_hrebiku − 1

```
            for(int i=0;i<pocet_hrebiku;i++){
```

Podstava se otočí na úhel i-tého hřebíku

```
        otoceni_podstavy_na_rad((float)argument_rad_nula_2pi(funkce_
vybrana((float)i/pocet_hrebiku, tvar, (float)polomer_matrice_cm)));
```

Lineární pojezd posune vrtákem na modul i-tého hřebíku

```
        posunuti_linpohybu_na_polomer_cm((float)modul(funkce_vybrana
((float)i/pocet_hrebiku, tvar, (float)polomer_matrice_cm)));
```

Vrt do hloubky jednoho centimetru

```
        vrt_cm(1.0);
            }
```

Otočení a podstavy a posunutí vrtáku na 0. hřebík

```
      otoceni_podstavy_na_rad((float)2.0*PI);
      posunuti_linpohybu_na_polomer_cm((float)modul(funkce_vybrana(
(float)0.0, tvar, (float)polomer_matrice_cm)));

      motor_podstava.setCurrentPosition(0);

      digitalWrite(enablePinLinPohyb, LOW); Vypnutí driverů
      digitalWrite(enablePinPodstava, LOW);
      digitalWrite(enablePinVrtani, LOW);

      while(digitalRead(pokracovaciPin) == HIGH)
{nulovani_pozic();}
```

```
            SOUCASNY_STAV=VINUTI;
            break;
            }


    case VINUTI: {
```

Nastavení minimální vzdálenosti od středu obrazu jehly

```
  min_polomer = min_polomer_funkce(tvar, (float)polomer_matrice_cm)
- 2.0;
```

```
  digitalWrite(enablePinVrtani, LOW);
```

Vypnutí driveru vrtání, aby nedocházelo k přehřátí motoru udržovacím proudem

Vysunují jehly od hřebíků

```
  posunuti_linpohybu_na_polomer_cm((float)polomer_matrice_cm + 2.0);
```

Otočení podstavy mezi 0. a 1. hřebík

```
  otoceni_podstavy_na_rad(argument_rad_nula_2pi(xy_na_bod(
(float)((funkce_vybrana((float)1.0/pocet_hrebiku, tvar,
(float)polomer_matrice_cm)).x+(funkce_vybrana((float)0.0/pocet_
hrebiku, tvar, (float)polomer_matrice_cm)).x)/2.0,
(float)((funkce_vybrana((float)1.0/pocet_hrebiku, tvar,
(float)polomer_matrice_cm)).y+(funkce_vybrana((float)0.0/pocet_
hrebiku, tvar, (float)polomer_matrice_cm)).y)/2.0)));
```

Posunutí jehly ke středu

```
  posunuti_linpohybu_na_polomer_cm((float)min_polomer_funkce(tvar,
(float)polomer_matrice_cm) - 2.0);
```

Otočení podstavy zpět na úhel 0. hřebíku

```
  otoceni_podstavy_na_rad((float)0.0);

  for(int i=0;i<precteniPoradi("poradi.txt",0)-1;i++){

    omotej_hrebik(precteniPoradi("poradi.txt", i+1),
(float)pocet_hrebiku, tvar, (float)polomer_matrice_cm);
            }

            SOUCASNY_STAV=KONEC;
            break;
            }
  }
}
```

3.5 Soupisky materiálů

3.5.1 Elektrická soupiska

Položka	Název	Typ	Počet	Poznámka
1	napájecí zdroj	LyonZG S-200-24	1	24 V
2	mikroprocesor	Arduino Mega 2560	1	
3	microSD modul		1	SPI rozhraní
4	konektor M	39-01-2040 MOLEX	3	4pin (2 × 2)
5	konektor F	39-01-2041 MOLEX	3	4pin (2 × 2)
6	driver	TB6600	3	
7	ventilátor		3	24 V, 50 mA
8	krokový motor	NEMA 23 NEMA 17	2 1	podstava, lineární pojezd vrtání
9	vrtačka	EXTOL CRAFT 404121	1	s kleštinou 1,7 mm
10	tlačítko	PBS-18B-G PBS-18B-W	1 3	zelená bílá
11	svorkovnice	KF128-2.54	4	2pin, šroubovací

3.5.2 Mechanická soupiska

Položka	Název	Typ	Počet	Poznámka
1	lineární pojezd		1	885 mm × 122 mm
2	upevňovací kostra		1	
3	držák krokového motoru		1	
4	držák na vrtačku a jehlu		4	
5	držák špule nitě		1	

4 Závěr

V celku se projekt povedl dle očekávání a splnil požadavky zadání. Jelikož byl projekt relativně složitý jak v teoretických, tak i v praktických ohledech, naučil jsem se několik nových dovedností a to: čtení dat z SD karty, ovládání krokových motorů a driverů pomocí Arduina, lepší přemýšlení o praktických a technických problémech, kreativnější myšlení o řešení problémů.

4.1 Komplikace při realizaci projektu

Jednou z komplikací, a to nejdéle trvající komplikací, bylo čtení dat z SD karty. Prohledával jsem všechny dostupné zdroje, které se zabývaly tím, jak čtení dat z SD karty provést a jak oživit SD kartové moduly. Zkusil jsem hned několik iterací řešení pro čtení SD karty a to: Arduino UNO + SD card modul; Arduino Mega 2560 + SD card modul; Arduino UNO + microSD modul; Arduino Mega 2560 + microSD modul. Pro Arduino Mega 2560 jsem zkoušel i rozdíly mezi piny 10 až 13 a 50 až 53. Jediné řešení, které fungovalo, bylo použít Arduino Mega 2560 (piny 50 až 53) + microSD modul + SDfat.h knihovnu, která nešla najít na internetu jako řešení mého problému, jelikož všechny zdroje uvádějí skoro výhradně SD.h knihovnu.

Další komplikací/výzvou bylo umělé přidání tření, aby se nevyvíjela niť kvůli získané setrvačnosti kónusu nitě. Při testování vinutí obrazů se niť vyvíjela více než bylo třeba a snižovalo se tak její napnutí, což vedlo k táhnutí nitě po povrchu obrazu. Ačkoli tato komplikace neovlivňovala kvalitu vinutých obrazů, představovala potenciální problém, jelikož se niť mohla zadrhnout v nežádoucím místě a následovně se přetrhnout. Problém byl poté vyřešen jednoduchou pružinkou, která třela o špuli.

4.2 Plány do budoucna

Přestože můj projekt operuje v cylindrických souřadnicích $[r, \theta, z]$ (na rozdíl od kartézských souřadnic $[x, y, z]$), může pracovat prakticky úplně identicky k CNC strojům nebo 3D tiskárnám, jelikož je přepočet z kartézských souřadnic na cylindrické souřadnice velice jednoduchý, a proto mám mnoho možností na možná budoucí rozšíření. V budoucnosti mám v plánu rozšířit tento projekt tak, aby zvládal gravírování laserem a pletení obrazů ve tvarech odlišných od kružnic a pravidelných n-úhelníků, jelikož obrazy ve tvarech obdélníků jsou mnohem méně vídané než kružnice a mandaly. Obdélníky jsou všeobecně mnohem častější tvary obrazů, které se připevňují na zeď.

Seznam použité literatury a zdrojů informací

[1] *String art*. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001. Dostupné z: https://en.wikipedia.org/wiki/String_art. [cit. 2025-03-18].

[2] DRING, Barton. *A New Spin on String Art Machines* [@Barton Dring]. Online. 2019. Dostupné z: YouTube, https://youtu.be/M1gXuKFspgY. [cit. 2025-03-24].

[3] *Stepper Motors and Arduino - The Ultimate Guide* [@How To Mechatronics]. Online. 2022. Dostupné z: YouTube, https://youtu.be/7spK_BkMJys. [cit. 2025-03-24].

[4] MCCAULEY, Mike. *AccelStepper library*. Online. 2010. Dostupné z: https://www.airspayce.com/mikem/arduino/AccelStepper/. [cit. 2025-03-24].

[5] LÁSKAKIT. *LYONZG S-200-24 modulový napájecí 230V AC-DC zdroj 24V/8,3A 200W*. Online. Láska kit – by makers for makers. 21. století. Dostupné z: https://www.laskakit.cz/lyonzg-s-200-24-modulovy-napajeci-230v-ac-dc-zdroj-24v-8-3a-200w/. [cit. 2025-03-24].

[6] ATMEL. *Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V Datasheet*. Online. Microchip. 21. století. Dostupné z: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf. [cit. 2025-03-24].

[7] TEXAS INSTRUMENTS. *DRV8825 Datasheet*. Online. Texas Instruments. 21. století. Dostupné z: https://www.ti.com/lit/ds/symlink/drv8825.pdf?ts=1742732489057. [cit. 2025-03-24].

[8] TOSHIBA. *TB67109AFTG, TB67109AFNG Datasheet*. Online. Toshiba. 21. století. Dostupné z: https://toshiba.semicon-storage.com/info/TB67S109AFTG_datasheet_en_20241210.pdf?did=14642&prodName=TB67S109AFTG. [cit. 2025-03-24].

Seznam zkratek a termínů

2D ..... dvoudimenzionální

3D ..... trojdimenzionální

AC ..... Alternating Current (střídavý)

arg(B) ..... argument bodu B (úhel, který bod B svírá s kladnou částí x-osy

CNC ..... Computer Numerical Control

DC ..... Direct Current (stejnosměrný)

DPS ..... Deska Plošného Spoje

dural ..... slitina hliníku, mědi a dalších přísad s poměrem Al:Cu přibližně 19:1

I2C ..... Inter-Integrated Circuit

kongruence ..... vztah mezi dvěma čísly, který říká, že jejich rozdíl je celočíselně dělitelný číslem m

kónus ..... cívka/špule, na které je navinuta niť

matrice ..... spojitá složka kompozitního matriálu, zastávající funkci pojiva celku nebo jeho částí

příruba ..... stěna, ze které vychází hřídel motoru

PWM ..... Pulse Width Modulation (pulzně šířková modulace)

UART ..... Universal Asynchronous Receiver-Transmitter (univerzální asynchronní přijímač-vylílač)

udržovací proud ..... proud, který v krokovém motoru vytváří konstantní magnetické pole, které udržuje natočení motoru a brání mu v pohybu

Seznam obrázků a jejich zdrojů

Obrázek 1: Zdroj: Jiří Frišman.

Obrázek 2: Zdroj: Jiří Frišman.

Obrázek 3: Zdroj: *LyonZG S-200-24*. Online. In: LÁSKA KIT. Láska kit – by makers for makers. [21. století]. Dostupné z: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/5775_5775-lyonzg-s-200-24-modulovy-napajeci-230v-ac-dc-zdroj-24v-8-3a-200w.jpg?6137b46c. [cit. 2025-03-06].

Obrázek 4: Zdroj: *LaskaKit Mega2560 rev3*. Online. In: LÁSKA KIT. Láska kit – by makers for makers. [21. století]. Dostupné z: https://www.laskakit.cz/arduino-mega2560/. [cit. 2025-03-06].

Obrázek 5: Zdroj: *microSD card modul*. Online. In: LÁSKA KIT. Láska kit – by makers for makers. [21. století]. Dostupné z: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/422-2_422-2-microsd-card-modul-spi.jpg?6137b46c. [cit. 2025-03-06].

Obrázek 6: Zdroj: Jiří Frišman.

Obrázek 7: Zdroj: Jiří Frišman.

Obrázek 8: Zdroj: *DRV8825: Driver pro krokové motory*. Online. In: LÁSKA KIT. Láska kit – by makers for makers. [21. století]. Dostupné z: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/1307-1_drv8825-driver-pro-krokove-motory.jpg?61d95cbb. [cit. 2025-03-06].

Obrázek 9: *Zdroj: Toshiba TB67S109: Driver pro krokové motory. Online. In: LÁSKA KIT. Láska kit – by makers for makers. [21. století]. Dostupné z: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/2147_toshiba-tb67s109-driver-pro-krokove-motory.jpg?61d95d3a. [cit. 2025-03-06].*

Obrázek 10: Zdroj: *TB6600: Driver pro krokové motory*. Online. In: SHARPLAYERS. Sharplayers. [21. století]. Dostupné z: https://sharplayers.s18.cdn-upgates.com/_cache/5/1/512b5178d145b41f306b23155f1148cd-660e46b68e6bc4.jpg. [cit. 2025-03-19].

Obrázek 11: Zdroj: Jiří Frišman.

Obrázek 12: Zdroj: Jiří Frišman.

Obrázek 13: Zdroj: Jiří Frišman.

Obrázek 14: Zdroj: Jiří Frišman.

Obrázek 15: Zdroj: Jiří Frišman.

Obrázek 16: Zdroj: Jiří Frišman.

Obrázek 17: Zdroj: Jiří Frišman.

Obrázek 18: Zdroj: Jiří Frišman.

Obrázek 19: Zdroj: Jiří Frišman.
Obrázek 20: Zdroj: Jiří Frišman.
Obrázek 21: Zdroj: Jiří Frišman.
Obrázek 22: Zdroj: Jiří Frišman.
Obrázek 23: Zdroj: Jiří Frišman.
Obrázek 24: Zdroj: Jiří Frišman.
Obrázek 25: Zdroj: Jiří Frišman.

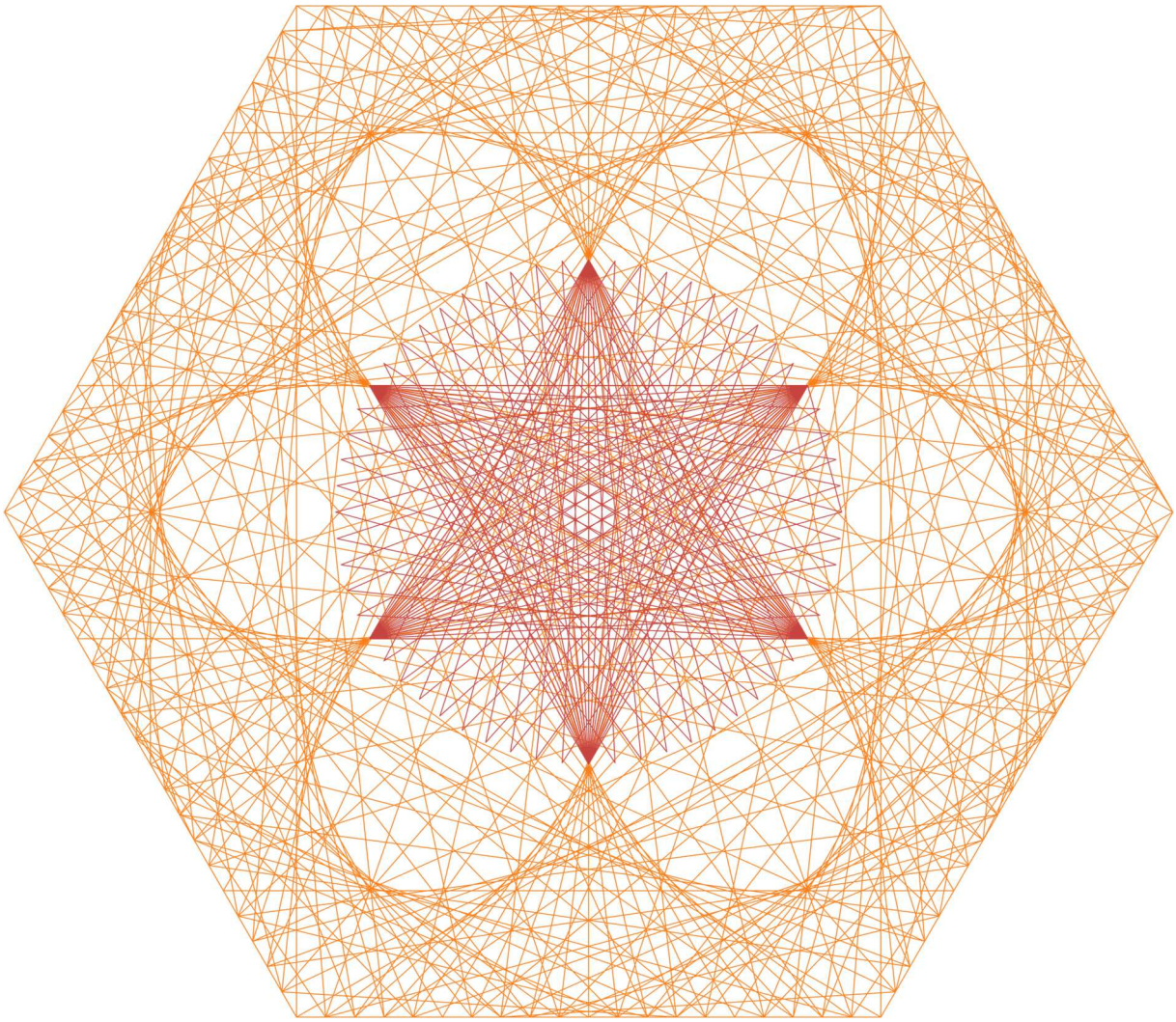
Seznam tabulek

Přílohy

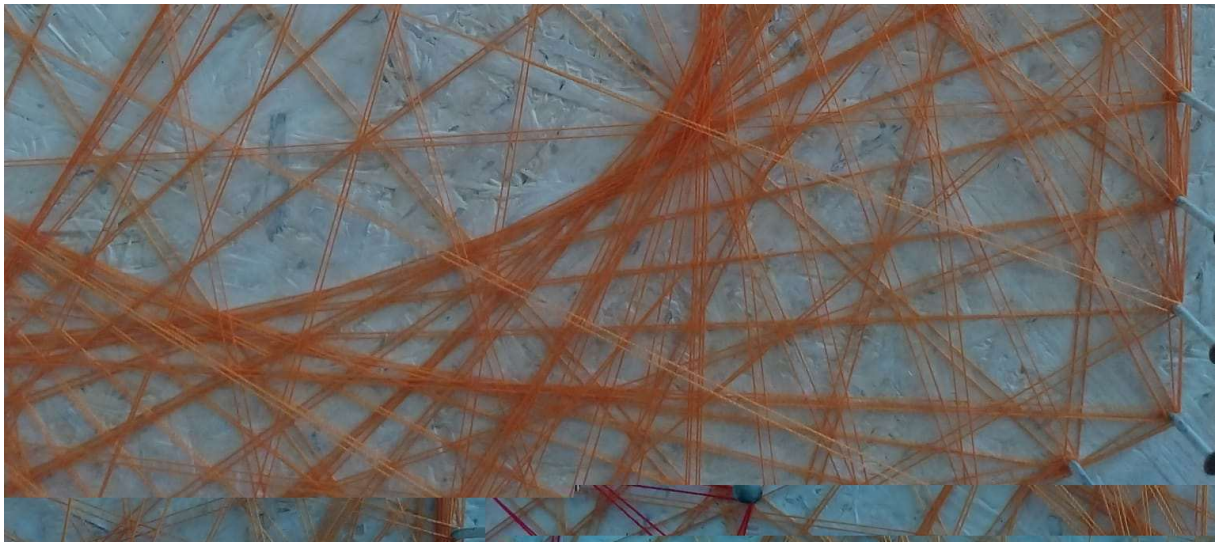
Příloha 1: První obraz – celý
Příloha 2: První obraz – teoretický výsledek
Příloha 3: První obraz – zblízka
Příloha 4: Finální projekt
Příloha 5: Držák vrtačky a jehly
Příloha 6: Rotační část

Příloha 1: První obraz – celý

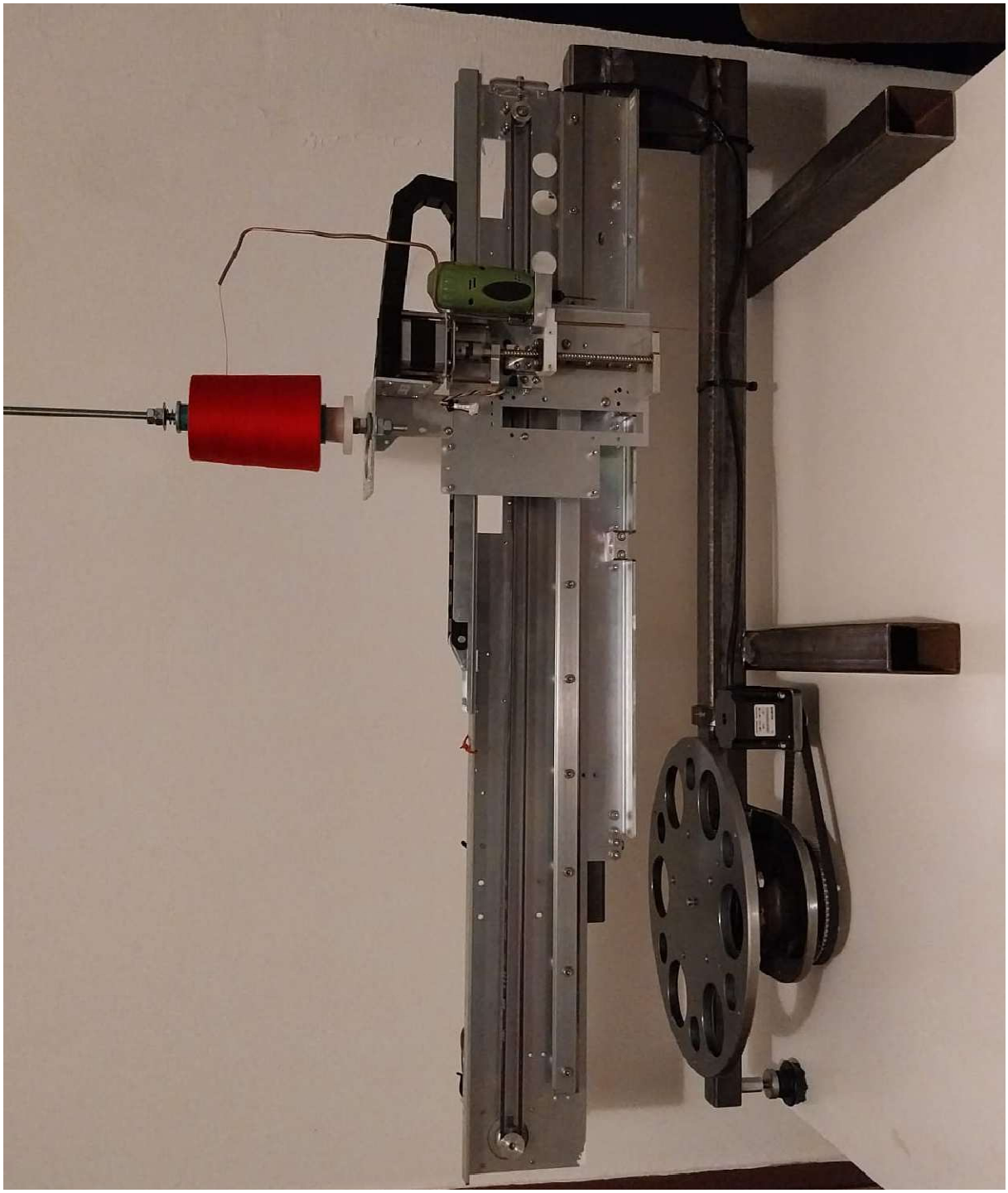
Příloha 2: První obraz – teoretický výsledek



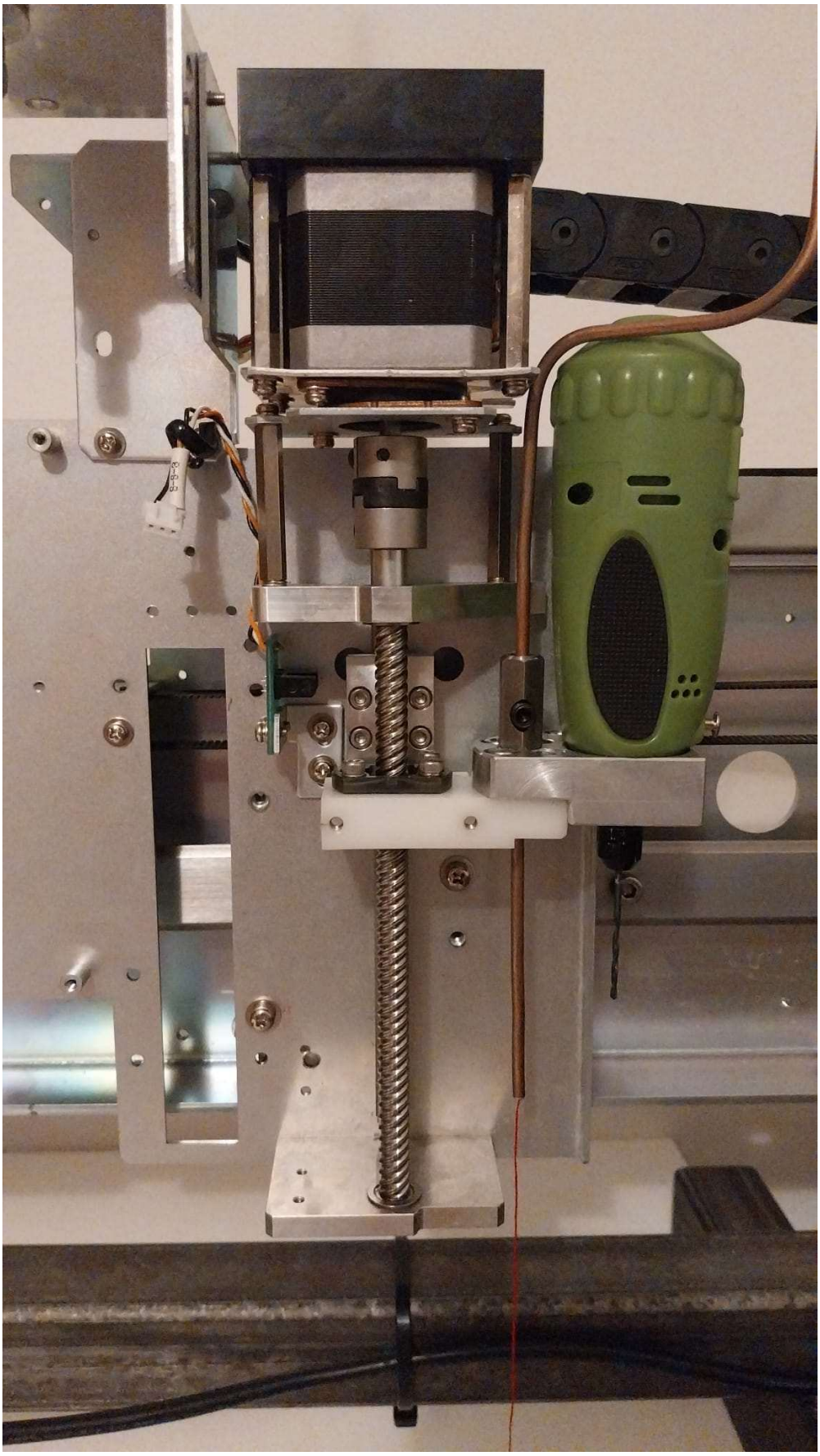
Příloha 3: První obraz – zblízka



Príloha 4: Výsledný projekt



Příloha 5: Držák vrtačky a jehly



Příloha 6: Rotační část