



Gymnázium, Praha 6, Arabská 14
Rubik's cube solver, vedoucí práce Mgr. J. Lána



Rubik's cube solver

Maturitní práce

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská¹⁴ oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 31.3 2021

Anotace

Úkolem mé ročníkové práce bylo vytvořit robota na skládání Rubikovi kostky. Součástí práce bylo vymyslet a navrhnout algoritmus, který by ovládal robota tak, aby byl robot schopen úspěšně vyřešit hlavolam. Práce se skládala ze 3 částí, ze samotného robota, z grafického uživatelského rozhraní, kde by uživatel zadal aktuální stav rozložené kostky, z něž by se následně vygenerovaly instrukce, podle kterých by byl robot následně schopen kostku složit a na konec ze samotného algoritmu, na jehož principu by robot prováděl sekvenci tahů, podle kterých by kostku složil.

Abstract

The main goal of this project was to make a robot for solving Rubik's cube. Part of my project was also to think up and design an algorithm, that would control the robot so that robot would be able to solve problem successfully. Project was compounded of 3 main parts, robot itself, graphic user interface, where user would fill in unsolved state of Rubik's cube, from which instructions for solving cube would generate and finally the project was compounded of the algorithm thanks to which would robot perform a sequence of moves, that would solve the cube.

Zadání

Úkolem mé ročníkové práce bude vytvořit robota na skládání Rubikovi kostky. Součástí práce bude vymyslet a navrhnout algoritmus, který bude ovládat robota tak, aby byl schopen jí složit. Uživatel by na začátku zadal do systému ručně aktuální rozložení kostky přes uživatelského prostředí, následně by se v počítači vypočítal algoritmus kroků, které by robot měl udělat pro složení kostky. Nakonec by se tyto kroky odeslali do robota, který by následně dokázali otáčet kostkou tak aby jí složili.

Obsah

1. Úvod	1
1.1. Cíl práce.....	1
2. Rubikova kostka	1
2.1. Historie	1
2.2. Známé algoritmy.....	3
2.2.1. <i>Začátečnický algoritmus</i>	4
2.2.2. <i>CFOP</i>	4
2.2.3. <i>Roux</i>	5
2.2.4. <i>ZZ</i>	6
3. Stavba robota	6
3.1. Použité technologie	6
1.3.1. <i>Arduino UNO R3</i>	6
3.2. Motory	7
3.3. Zapojení obvodu	8
3.4. Konstrukce robota	9
4.3.1. <i>Rameno</i>	9
4.3.2. <i>Základna</i>	10
4. GUI.....	11
4.1. Použité technologie	11
1.4.1. <i>Python</i>	11
1.4.2. <i>Pyserial</i>	11
1.4.3. <i>Tkinter</i>	11
1.4.4. <i>PyCharm ve verzi Preffesional</i>	11
4.2. Funkce a použití aplikace	12
5. Bibliografie.....	14
6. Seznam obrázků.....	15

1. Úvod

1.1. Cíl práce

Cílem této práce bylo napodobit již vymyšlené konstrukční řešení pro robota skládajícího Rubikovu kostku a následně se pokusit navrhnout vlastní algoritmus pro úspěšné složení kostky, které by bylo možné porovnat s již vytvořenými konkurenčními algoritmy. Součástí práce bylo vytvoření grafického uživatelského rozhraní, které by bylo uživatelsky přívětivé a minimalizovalo by nekorektní zadání kostky.

Jako základní impuls pro vznik této práce mi sloužil můj zápal pro problematiku skládání Rubikovy kostky.

Tato práce se bude zabývat konstrukční problematikou, problematikou výběru adekvátního algoritmu a neposlední řadě problematikou tvorby vlastní aplikace.

2. Rubikova kostka

2.1. Historie

Rubikova kostka byla poprvé zkonstruována v roce 1974 maďarským sochařem, architektem a vynálezcem Ernő Rubikem. Prvotní záměr Rubika bylo vytvořit učební pomůcku pro své studenty, díky které by byli schopni lépe porozumět 3D objektům. Následně si 30. ledna 1975 podal žádost o patent, a ještě v témže roce se obrátil s nabídkou spolupráce na společnost Politechnika Ipari Szövetkezet (později přejmenována na Politoys), která tehdy zásobovala svými plastovými šachovými soupravami a podobnými hrami společnost Trial, která tehdy zastávala funkci největšího distributora hraček v Maďarsku.

První testovní série 5000 kostek byla poprvé prodávána roku 1977 v Budapešti. Zprvu neměla od společnosti Trial moc velkou důvěru na úspěch, ale opak byl pravdou. Téměř okamžitě po uvedení do maďarských obchodů zaznamenala kostka enormní úspěch, takřka okamžitě byli všechny zásoby kostek vykoupeny.

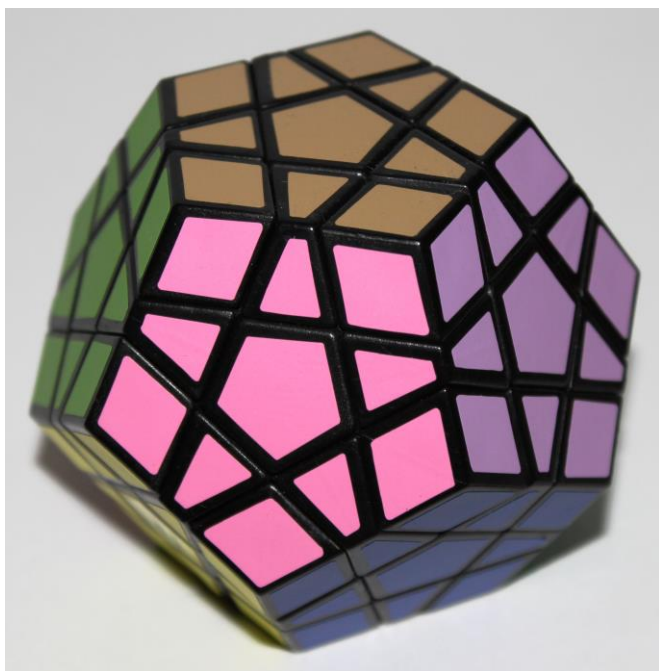


Obrázek 1 Rubikova kostka (1)

V roce 1979 došlo k dohodě mezi Rubikem a známou americkou společností Idea Toy, která přebrala celosvětovou distribuci Rubikovy kostky. Trend Magické kostky zaznamenal největší rozmach mezi roky 1980-1981, kdy americká společnost Idea Toy uvádí, že pouze na americkém trhu prodala v roce 1980 4,5 milionů kostek a o rok později neuvěřitelných 10 milionů kostek.

V roce 2009 byl celkový odhad prodaných kostek stanoven na enormních 350 milionů kusů. (2) V dnešní době je kostka považována za symbol osmdesátých let a zároveň se jedná o jeden z nejznámějších hlavolamů na světě. Své fanoušky si drží ve světě dodnes. Dokonce se i dnes pořádají mezinárodní soutěže v tzn. speedcubingu, ve kterém mají soutěžící za úkol složit kostku v co nejkratším čase. Rekord v této soutěži drží Yusheng Du z Číny, který dokázal kostku složit za 3,47 sekund. Prvenství v rychlosti z řad robotů si drží projekt amerického studenta Bena Katze, který dokáže problém složit za 0,38 sekundy (3).

Z původní $3 \times 3 \times 3$ kostky se k dnešnímu dni vyvinulo velké množství modifikací, jako například $2 \times 2 \times 2$, $4 \times 4 \times 4$, $5 \times 5 \times 5$, $6 \times 6 \times 6$ a $7 \times 7 \times 7$ kostky, Megaminx (dvanáctistěn), Pyraminx (čtyřstěn), Square-1 a v neposlední řadě i siamské kostky (2 kostky spojené dohromady). (4)



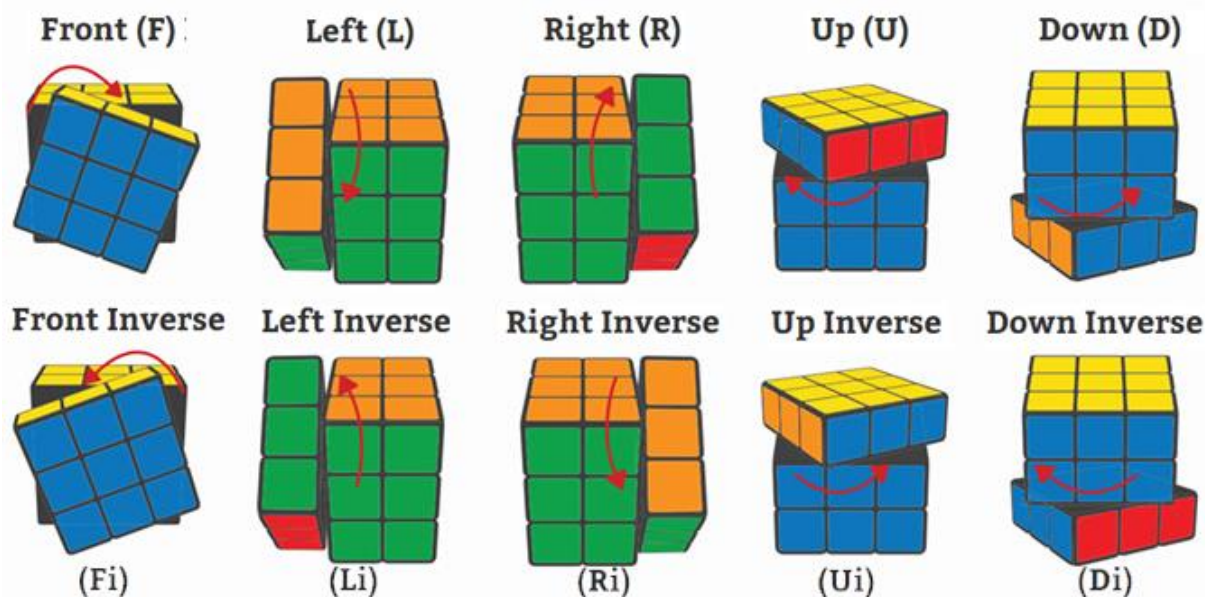
Obrázek 2 Megaminx (5)

2.2. Známé algoritmy

V této kapitole si blíže popíšeme nejznámější a nejpoužívanější algoritmy pro složení Rubikovy kostky.

Pro pochopení popisu algoritmů je nejprve nutné se seznámit se základní terminologií popisu algoritmů pro skládání Rubikovy kostky. Základní hlavolam 3x3x3 se skládá ze 6 stěn, z nichž každá má vlastní barvu, tradičně se jedná o bílou, modrou, červenou, zelenou, oranžovou a žlutou barvu. Každá stěna se skládá z 9 polí, konkrétně ze 4 rohů, 4 hran a jednoho středu. U hran i rohů lze libovolně měnit pozici na kteroukoliv jinou na kostce, středy však zůstávají vždy na stejné pozici. Proto lze strany kostky i při rozloženém stavu označit podle barvy středů.

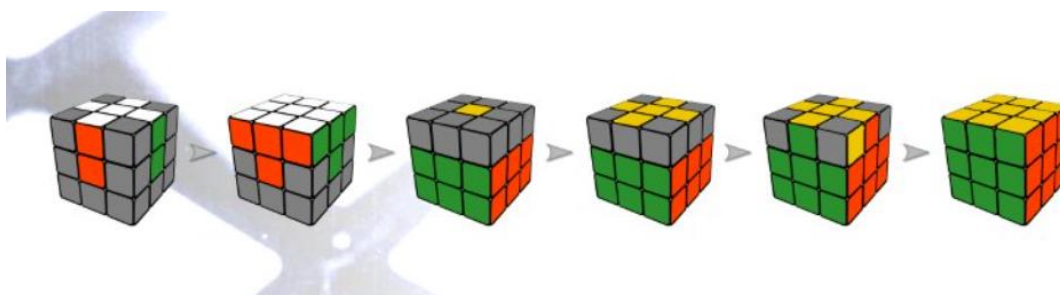
Před samotným skládáním si skládající osoba nejprve určí, která strana bude spodní a která strana bude přední. Jako spodní vrstva se tradičně volí buď bílá nebo žlutá strana, za přední se volí modrá strana. Skládající osoba pak následně po celou dobu skládání drží kostku tak, aby byla spodní strana vespod a přední strana vepředu. Díky tomu je možné následně jednoznačně určit, která strana je levá, pravá, zadní, dolní, přední, horní. To napomáhá k jednoduššímu popisu algoritmu. U zápisu jednotlivých kroků algoritmu se využívá první písmeno anglického překladu již zmíněných orientací stran. K písmenu se ještě přidává apostrof, který značí směr otáčení. Pro příklad zápis U' znamená, že otočíme horní stranou doprava.



Obrázek 3 Základní operace s Rubikovou kostkou (6)

2.2.1. Začátečnický algoritmus

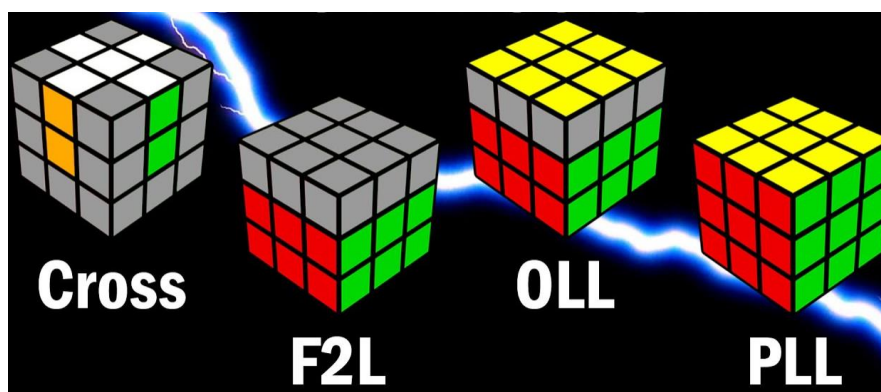
Základní myšlenka začátečnické metody je založena na metodě CFOP. Oproti metodě CFOP byl začátečnický algoritmus dosti zjednodušen, tak aby zvládl kostku složit naprosto každý. Algoritmus se skládá celkem ze 7 kroků, nejprve je složen tzn. „spodní kříž“, kdy se ve spodní vrstvě snaží řešitel hlavolamu dostat hrany kostky na svou pozici, tak aby všechny barvy hran souhlasily s barvou středů. Následně řešitel doplní všechny rohy spodní vrstvy, opět tak, aby všechny barevně seděli. Dalším krokem je složení střední vrstvy, kdy skládající doplní všechny hrany, které barevně pasují do středního pásu kostky. Následně je složen tzn. „horní kříž“, který ale na rozdíl od spodního kříže nemusí korespondovat s barvami středního pásu. Korekce barev u horního kříže se provede v následujícím kroku. V předposledním se opraví rozestavení horních rohů, které se dosadí do správné pozice podle barev. Nakonec se upraví orientace horních rohů, čímž je kostka složena.



Obrázek 4 Grafické znázornění začátečnické metody (7)

2.2.2. CFOP

CFOP neboli Fridrichova metoda je v dnešní době jedna z nejpoužívanějších metod rychlostního skládání Rubikovy kostky. Byla navržena Jessicou Fridrichovou (dříve Jiří Fridrich). Poprvé byla publikována její autorkou na jejím webu v roce 1995 pod názvem „Můj systém pro skládání Rubikovy kostky“. Metoda se skládá ze 78-119 algoritmů (záleží na variantě CFOP) a průměrný počet tahů na jedno složení se pohybuje v rozmezí od 55 do 60 tahů. Statisticky se momentálně jedná o nejrychlejší metodu skládání Rubikovy kostky. S touto metodou byly dokonce dosaženy všechny aktuální světové rekordy. (8)



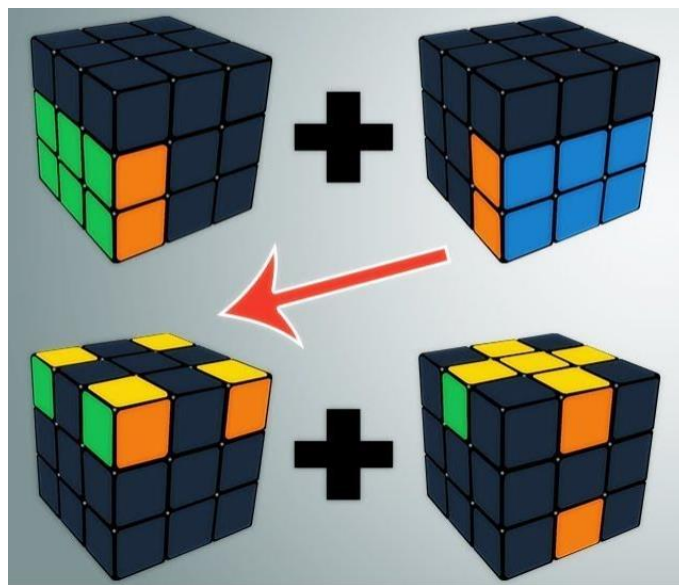
Obrázek 5 Grafické znázornění CFOP metody (9)

CFOP je zkratka pro jednotlivé kroky této metody, C pro cross (kříž), F pro first two layers (první dvě vrstvy), O pro Orientation of the Last Layer (orientace poslední vrstvy) a P pro Permutation of the Last Layer (permutace poslední vrstvy). V první fázi je podobně jako u začátečnické metody složen spodní kříž. Na rozdíl od začátečnické metody jsou však v druhé fázi skládány spodní rohy a střední hrany zároveň, což výrazně snižuje počet potřebných tahů a čas skládání. V předposledním kroku je za pomoci jednoho ze 57 algoritmů orientována horní vrstva tak, aby v horní vrstvě byly pouze barvy vrchní strany. Nakonec jsou všechny horní hrany a rohy za pomoci jednoho z 21 algoritmů přesunuty do správných pozic, čímž je skládání úspěšně dokončené. (8)

2.2.3. Roux

Roux je metoda vytvořena Gilles Roux v roce 2003. Oproti CFOP metodě disponuje hned dvěma výhodami, má v průměru méně tahů potřebných pro složení kostky a snižuje rotaci rukou, což z ní dělá ideální metodu pro jednoruční skládání. Metoda obsahuje 42 algoritmů a průměrný počet tahů je uváděn jako 48. (10)

Roux metoda se skládá ze 4 kroků. V prvním kroku je vytvořen kdekoli na kostce blok 3x2x1. Následně je vytvořen druhý blok 3x2x1, který barevně odpovídá bloku, který by měl ležet naproti již vytvořenému bloku. Ve 3 fázi jsou slouženy zbývající 4 rohy, které se nacházejí v horní vrstvě. Nakonec je opraveno i zbývajících 6 rohů, čímž je kostka složena.



Obrázek 6 Grafické znázornění Roux metody (11)

2.2.4. ZZ

ZZ metoda byla vytvořena v roce 2006 polským profesionálním kostkařem Zbigniew Zborowským, po němž byla i jeho iniciály tato metoda pojmenována. Ze všech zmíněných se jedná o nejkomplicovanější a nejhůře naučitelnou metodu, dohromady se totiž skládá z až 1074 algoritmů. Metoda si získala oblibu pro svou rychlost provedení všech tahů. Průměrný počet tahů je 45.

První krok se označuje jako EOLine. V této fázi se pohledově nic neskládá, nýbrž se provádí pouze úprava všech hran kostky, tak aby se kostka ve finálním kroku nacházela, v co možno nejoptimálnější pozici. Na začátku jsou všechny hrany kostky podle své pozice označeny buď jako dobré, nebo jako špatné. Následně se pomocí jednoho z 537 sekvencí tahů uvedou všechny hrany do „dobré“ pozice.

Po první fázi se postupuje obdobně jako u CFOP metody, tudíž se provede F2L sekvence tahů, čímž se docílí složení prvních 2 vrstev. Následně jsou již všechny zbývající hrany na svém místě. Toho bylo docíleno, díky EOLine přípravě. Nakonec ne provedena sekvence tahů, které horní vrstvu dokončí, čímž složí celou kostku. (12)

3. Stavba robota

3.1. Použité technologie

1.3.1. Arduino UNO R3

Arduino je open-source platforma s vlastním IDE. Arduino vzniklo v roce 2005 v Itálii, jako učební pomůcka pro studenty. Hlavním cílem vývoje bylo vytvořit prototypovací platformu pro rychlý, snadný a pohodlný vývoj jednoduchých projektů. Po uvedení zaznamenalo Arduino poměrnou oblibu, společnost uvádí že do roku 2010 prodala kolem 120 000 těchto desek. Arduino se vyrábí celkem v 11 modelových řadách.

Mnou použité Arduino UNO R3 je postavené na 8bitovém čipu ATmega328P, který je vyvíjen společností Atmel. Pro svou kompaktnost se jedná o nejpoužívanější Arduino desku. Arduino UNO R3 je vyráběno ve 3 variantách, buď s USB portem, ethernet portem nebo Bluetooth modulem. UNO R3 disponuje 32 KB flash pamětí, 14 digitálními I/O piny (z toho 6 jich je PWM) a 6 analogovými piny. Pro jednoduchost zapojení bez nutnosti pájení se zde využívá standardizované patice, které se přezdívá Shields. Přestože je pro propojení Arduina použit USB konektor, umí Arduino softwarově simulovat sériovou komunikaci přes linku RS-232. (13)

3.2. Motory

Vzhledem k tomu, že cílem práce bylo i vedle softwaru vytvořit i funkční verzi robota. Prvním krokem byl výběr vhodných motorů. Mezi motory vhodnými ke stavbě robota jsem vybíral ze 3 základních typů: DC motory, krokové motory a servomotory. Vzhledem k charakteru projektu jsou pro výběr motorů zásadní dva parametry, tj. přesnost a rychlost.

DC motory jsou konstrukčně nejjednodušší ze všech zmíněných motorů. Pracují na jednoduchém principu, pokud jsou připojeny ke zdroji, tak se točí, pokud ne, tak se netočí. Rychlost otáčení lze redukovat pomocí snížení napětí. DC motory jsou velice rychlé, ale vzhledem k tomu, že zde není zaveden žádný mechanismus, který by kontroloval korektnost otáčení o požadovaný úsek, jsou DC motory poměrně nepřesné.

Krokové motory jsou složeny z cívek a rotoru. Cívky jsou rozestavěny naproti ve dvojicích. Rotace rotoru je následně dosaženo postupným zapínáním napětí jednotlivých dvojic cívek, které tak následně vytvoří magnetické pole. Podle počtu cívek lze následně v motoru definovat tzn. kroky, o které lze rotor otočit. Krokové motory jsou poměrně přesné, ale zároveň s tím i poměrně pomalé. Také je u nich poměrně těžké určit přesnou potřebnou polohu.

Servomotory se skládají ze 3 částí: z DC motoru, ovládací desky a potenciometru. Ovládací deska otáčí motorem a zároveň získává údaje od potenciometru, který měří o kolik se rotor otočil. Díky tomu je možné přesně ovládat, o kolik stupňů se motor otáčí. Servomotor poskytuje velkou přesnost (přesnější než u krokových motorů), větší rychlost a stabilitu provedených úkonů i při změně zátěže. Jediné nevýhody jsou jejich omezenost pohybu, jelikož jim jejich konstrukce neumožňuje větší rotaci než o určitý počet stupňů (u drtivé většiny servomotorů je maximální stupeň rotace 180 stupňů) a zároveň o něco dražší cena, která se pohybuje u základních modelů kolem 350 korun za kus.

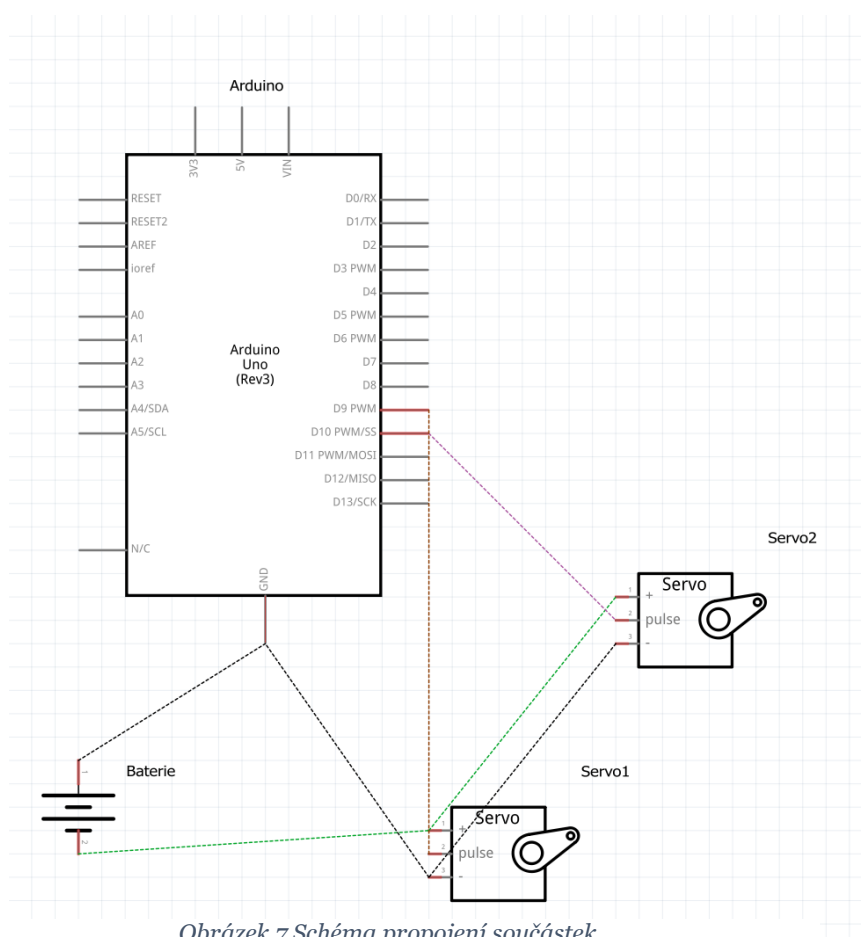
Pokud by se u projektu podařilo vyřešit problémy spojené s mechanickými nevýhodami servomotorů, jednalo by se o ideální volbu. Proto jsem u projektu rozhodl použít právě servomotory.

3.3. Zapojení obvodu

Samotné Arduino lze napájet přes USB konektor, který do Arduina přivádí napětí 5V, které je výrobcem doporučené. Alternativně lze Arduino napájet samostatným adaptérem přímo ze sítě. Pro můj projekt je však esenciální komunikace za pomoci USB se stolním počítačem, proto by samostatné napájení adaptérem nedávalo smysl.

Proto aby bylo sníženo riziko poškození destičky Arduino, je doporučováno pro zapojení servomotorů využít samostatného externího zdroje. Toho bylo dosaženo čtyřmi tužkovými bateriemi, které dohromady tvoří napětí 6V.

Servomotor disponuje 3 konektory, jeden pro přívod napětí, jeden pro uzemnění a poslední pro propojení ovládací destičky s Arduinem. Propojení mezi Arduinem a servomotorem je nutné přes PWM port, který je schopen redukce napětí přiváděného do servomotoru. Uzemnění celého obvodu je provedeno skrz GND port na desce Arduino.



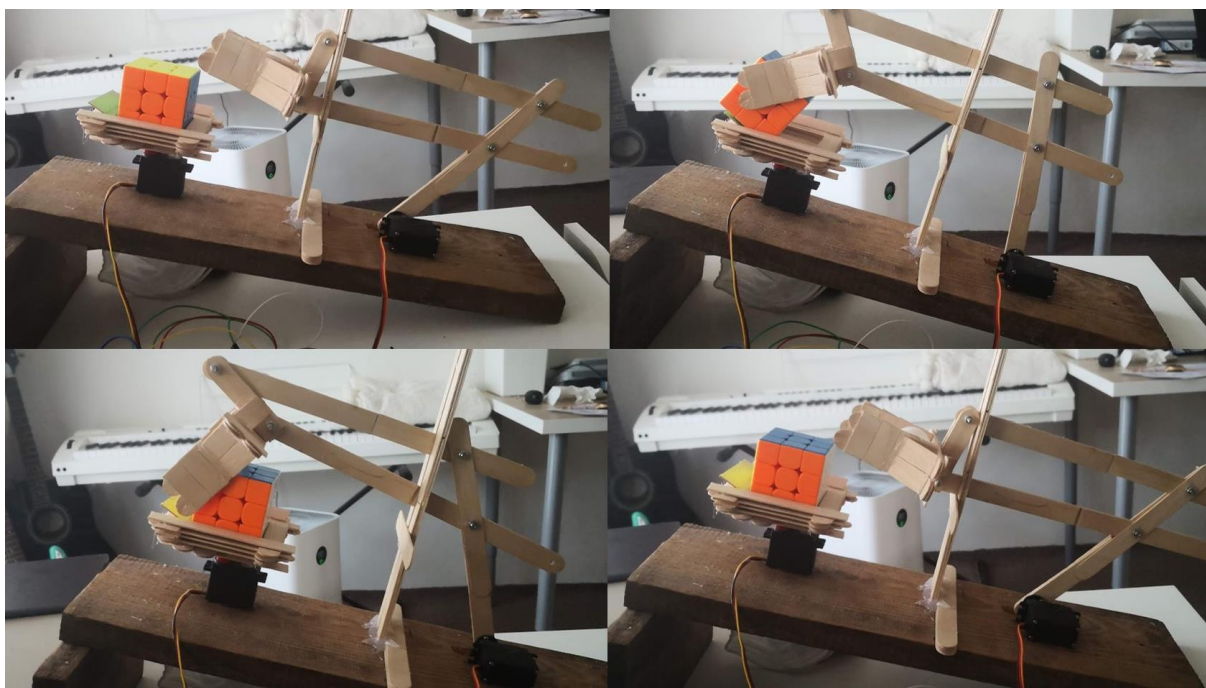
Obrázek 7 Schéma propojení součástek

3.4. Konstrukce robota

V této kapitole se blíže seznámíme s jednotlivými částmi robota a jejich funkcemi.

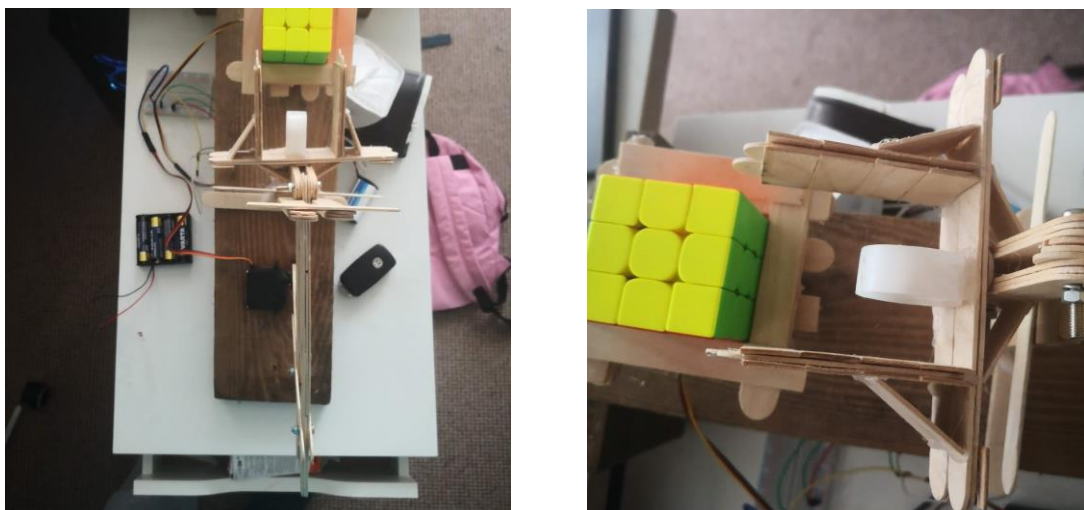
4.3.1. Rameno

Tato část robota provádí dvě hlavní operace, převrací a drží kostku. Rameno je poháněno jedním servomotorem, který díky důmyslné konstrukci zvládá provádět všechny požadované operace. Rameno je sestaveno z několika dřevěných doktorských lopatek, 4 šroubů a plastového kolečka.



Obrázek 8 Průběh převrácení kostky

Rameno je podpořeno dvěma vzpěrami, které minimalizují vychýlení ramena z osy při otáčení stran kostky. Při převrácení kostky může dojít hned k několika chybám, např. se rameno otočí moc rychle, takže je následně kostka vyvrhnutá ze základny, nebo se kostka zaklíní ve vidlici, takže se kostka nepřevrátí, nebo vypadne ze základny. Celý mechanismus je proto navrhnout tak, aby minimalizoval chybovost provedených operací. Při převrácení kostky tak dojde k naklonění vidlice, díky čemuž se minimalizuje riziko zaklínění kostky v meziprostoru vidlice. Uprostřed vidlice je také přiděláno plastové kolečko, které díky své povrchu snižuje tření, a tak nedochází ke strhnutí kostky zpět do původní pozice bez převrácení. Kolečko také napomáhá směřovat tlak na pouze na střední vrstvu kostky, proto nedochází k nechtěnému otočení stran.



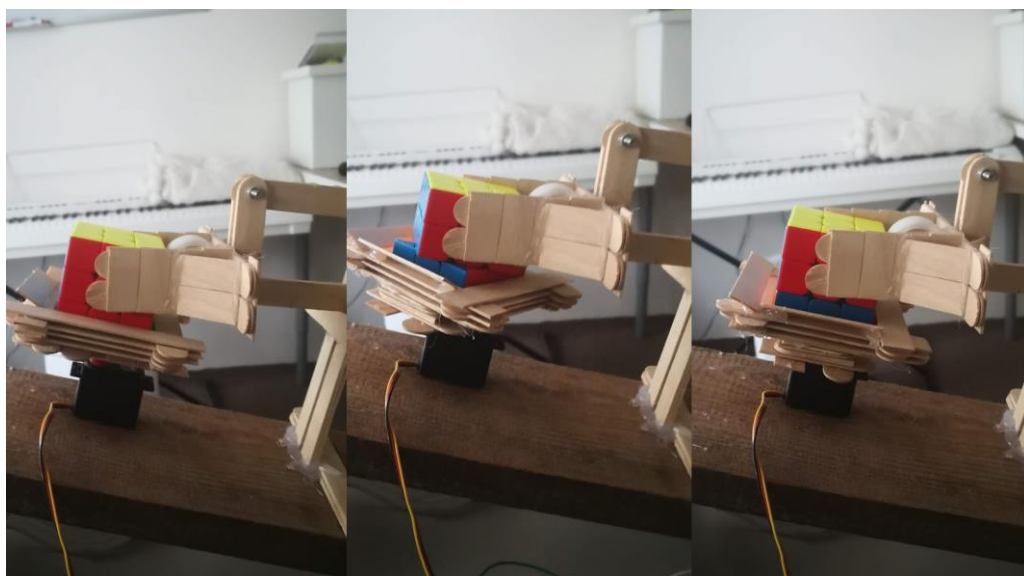
Obrázek 9 Pohled na rameno a vidlici seshora

4.3.2. Základna

Funkce základny jsou otáčení kostky, držení kostky na místě, společně s ramenem převrací kostku a otáčí požadovanou stranou. Základna je vytvořena z několik lékařských dřivek a jednoho servomotoru.

Vzhledem k použití servomotorů je rotace základny omezena pouze na rozmezí od 0 do 180 stupňů. To nám vytváří problém, jelikož pokud bychom otočili stranou a následně bychom tou samou chtěli otočit znovu, tak by to čistě z mechanického hlediska nešlo, nebylo by kam otáčet. Proto bude v našem programu nutné ošetřit to, aby se základna i s kostkou uvedli do výchozí pozice vždy po otočení stranou.

Základna disponuje několika nakloněnými rovinami, které napomáhají k minimalizování chybovosti převrácení kostky. Celá základna i s ramenem jsou připevněná na nakloněné podložce, což vytváří ideální sklon pro převrácení kostky.



Obrázek 10 Průběh otočení strany

4. GUI

4.1. Použité technologie

1.4.1. Python

Python je open-source, objektivně orientovaný a dynamicky překládaný programovací jazyk. Jeho jádro a základní funkce jsou vytvořeny pomocí jazyka C, díky čemuž se jedná o velice rychlý jazyk, vhodný pro rychlé zvládnání náročných výpočtů. Je velice populární pro svou jednoduchost a přehlednost. Mnohými je označován jako ideální programovací jazyk pro začátečníky, jelikož má velice dobře pochopitelný zápis. Díky několika pravidlům formátování se obejde bez jakýkoliv závorek čárek a středníků, bez kterých by jiné jazyky nemohli existovat, čímž se právě pro začátečníky stává čitelnější a srozumitelnější. Velkou výhodou pythonu je také velké množství uživatelů vytvořených knihoven, které z pythonu dělají velice silný nástroj pro vývojáře. Python byl zvolen do mého projektu, jelikož je pro Python vytvořena knihovna Pyserial, která perfektně splňovala mé požadavky pro tvorbu komunikace mezi Arduinem a GUI. Zároveň byla její implementace do projektu velice snadná (14)

1.4.2. Pyserial

Pyserial je knihovna pro Python, která poskytuje sériovou komunikaci přes RS-232 port. Například tato knihovna umožňuje komunikaci se starými sériovými porty, Bluetooth zařízeními, infračervenými porty nebo i pro nás potřebným Arduinem.

1.4.3. Tkinter

Tkinter je knihovna, která umožňuje vytváření grafického uživatelského rozhraní v Pythonu. Už v základu je součástí instalace Pythonu.

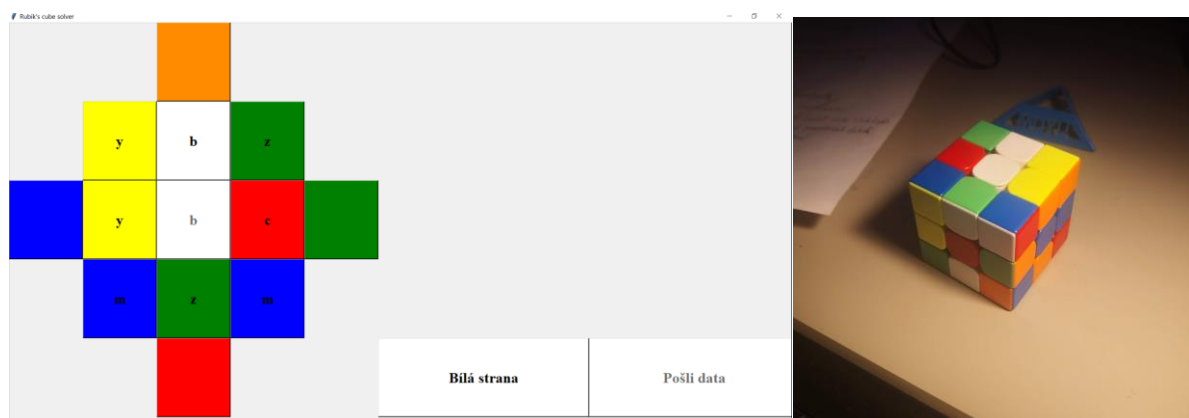
1.4.4. PyCharm ve verzi Professional

PyCharm je IDE vyvíjené českou společností JetBrains. První beta verze byla zveřejněna v roce 2010. Poskytuje analýzu kódu, grafický debugger, integrovaný tester jednotek, integraci s systémy pro správu verzí (VCSes) a podporuje vývoj webových aplikací pomocí Django a mnoho dalších pomocných nástrojů pro vývoj aplikací v Pythonu. PyCharm je dostupný na všech nejpoužívanějších operačních systémech, tedy na Windows, MacOS a Linuxu. (15)

4.2. Funkce a použití aplikace

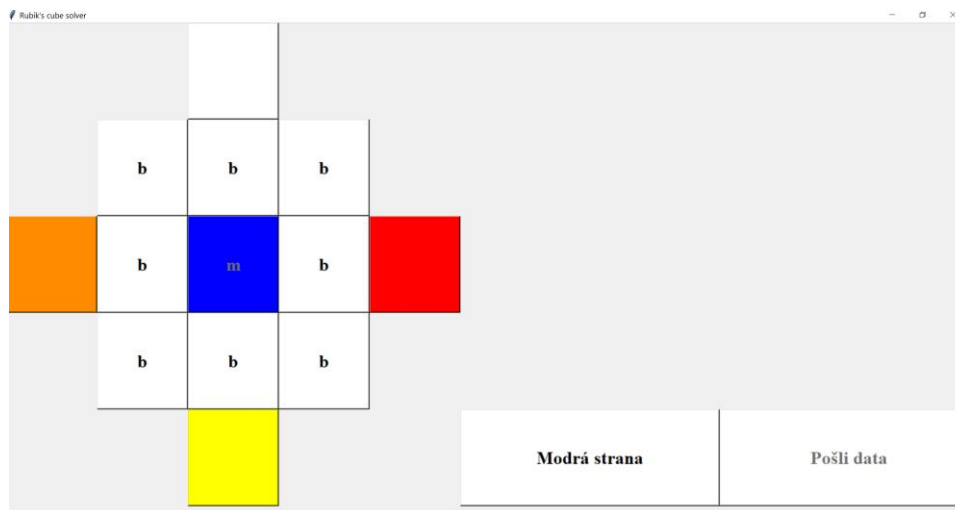
Samotná aplikace je strukturována do dvou částí, v první části jsou funkce spojené se zadáním informací od uživatele, ve druhé části jsou tyto informace odeslány do Arduina.

Po spuštění aplikace se otevře základní okno a zároveň se naváže sériová komunikace mezi GUI a Arduinem (viz obrázky 11 a 13). Uživatel zde zadá aktuální rozložení kostky. Pro správné zadání stavu kostky slouží uživateli postranní barvy, které uživateli napovídají, v jakém pořadí má daná políčka do aplikace zadat. Pro omezení rizika zadání nekorektního stavu kostky, je uživateli odepřen přístup ke změně prostřední barvy, která na kostce zůstává konstantní a zároveň udává pořadí, ve které má postupně uživatel zadávat jednotlivé strany. Po stisku jednotlivých tlačítek pro zadání kostky se podle aktuálního písmene změní barva na další v pořadí. Pokud se uživatel dostane až na konečnou žlutou barvu, je po dalším stisknutí barva změněna na výchozí bílou barvu.



Obrázek 11 Ukázka aplikace a korektního zadání rozložené kostky

Pokud je uživatel spokojen se svým zadáním, stiskne tlačítko nesoucí název ve formátu „barva strana“ strana. Stisknutím tohoto tlačítka se provede hned několik operací. Za prvé se všechna tlačítka pro zadávání stavu kostky uvedou do výchozího stavu. Za druhé se uživatelské prostředí přizpůsobí na zadávání dané strany, takže se po zadání bílé strany změní text tlačítka „bílá barva“ na „modrá barva“, prostřední políčko strany se změní na modrou barvu a jako poslední se změní i barva okolních políček sloužících jako nápověda pro uživatele na barvy napovídající pro danou stranu (viz Obrázek 12). Nejdůležitější operace z hlediska funkčnosti aplikace se odehrává mimo vidění uživatele, jelikož program převede všechny zadané hodnoty do předdefinovaného pole. Každá strana kostky má nadefinované své vlastní pole, do kterého se zapisují v určitém pořadí písmena zapsána na barevných políčkách. Tento postup provede uživatel celkem 6x. Po zadání žluté strany se všechny tlačítka pro uživatele uzamknou, odemkne se pouze tlačítko pošli data.



Obrázek 12 Stav GUI po zadání první strany

Po stisku tlačítka „pošli data“ je zavolána stejnojmenná funkce *posliData()* (viz obrázek 13). Sériová komunikace je započata za pomoci příkazu *serial.Serial(port, baudrate)*, kdy je jako první parametr uveden komunikační port, na kterém se nachází Arduino a jako druhý parametr se udává baudrate, neboli počet bitů za sekundu, který se definuje při otvírání sériového portu na Arduino. Funkce *posliData()* nejprve za pomoci funkce *resetujArduino()* uvede Arduino do počátečního stavu, následně zavolá funkci *zapisPoleDoString()*, která má za úkol vytvořit ze všech hodnot polí, v kterých jsou uložena data o aktuálním stavu kostky, jeden globální string *data*. String *data* je nakonec odeslán jako posloupnost bytů za pomoci příkazu *ser.write()* na sériový port Arduino, čímž je celé odeslání dat úspěšně dokončené.

```

50
51 ser = serial.Serial('COM3', 9600)
52
53 def posliData():
54
55     global data
56     resetujArduino()
57     zapisPoleDoString()
58
59     time.sleep(50)
60     ser.write(data.encode("utf-8"))
61     print("Data odeslána")
62     ser.close()

```

Obrázek 13 Funkce *posliData()*

5. Bibliografie

1. **Piqsels.** 3, krychle, selektivní, zaostřovací fotografie, abstraktní, rozmazat, detail, barvy. [Online] 2019. [Citace: 28. 3 2021.] <https://www.piqsels.com/cs/public-domain-photo-zkqfh>.
2. **Cruise, Rodney.** The history of Rubik's Cube. [Online] 2015. [Citace: 28. 3 2021.] <https://www.lexology.com/library/detail.aspx?g=dbcf1ba-19co-4c70-942b-38e95a4f5ba7>.
3. **Rubik's.** World's Fastest. [Online] 2019. [Citace: 28. 3 2021.] <https://www.rubiks.com/en-us/speed-cubing>.
4. **volume, Chinese Journal of Mechanical Engineering.** Overview of Rubik's Cube and Reflections on Its Application in Mechanism. [Online] 2018. [Citace: 28. 3 2021.] <https://cjme.springeropen.com/articles/10.1186/s10033-018-0269-7>.
5. **cubemeister.com.** Megaminx solved cubemeister. [Online] 2012. [Citace: 28. 3 2021.] https://commons.wikimedia.org/wiki/File:Megaminx_solved_cubemeister_com.jpg.
6. **How, Show Me.** Practical tutorials. [Online] 2011. [Citace: 29. 3 2021.] <https://cz.pinterest.com/pin/189995678008478420/>.
7. **CUBE3X3.** How to solve a Rubik's cube. [Online] 2018. [Citace: 29. 3 2021.] <https://cube3x3.com/>.
8. **Speedsolving.com.** CFOP method. [Online] 2021. [Citace: 29. 3 2021.] https://www.speedsolving.com/wiki/index.php/CFOP_method.
9. **Perm, J.** Rubik's Cube: How to Learn the CFOP Speedcubing Method. [Online] 2020. [Citace: 29. 3 2021.] https://www.youtube.com/watch?v=MS5jByTX_pk.
10. **speedsolving.com.** Roux method. [Online] 2019. [Citace: 29. 3 2021.] https://www.speedsolving.com/wiki/index.php/Roux_method.
11. **CriticalCubing.** How to Solve the 3x3 Rubik's Cube: Beginner's Roux Method (Easy Tutorial). [Online] 2017. [Citace: 29. 3 2021.] https://www.youtube.com/watch?v=ZXWbV8_CDyM&t=7s.
12. **speedsolving.com.** ZZ method. [Online] 2019. [Citace: 29. 3 2021.] https://www.speedsolving.com/wiki/index.php/ZZ_method.
13. **Hughes, J. M.** Arduino: A Technical Reference by J. M. Hughes. [Online] 2019. [Citace: 30. 3 2021.] <https://www.oreilly.com/library/view/arduino-a-technical/9781491934319/ch01.html>.
14. **Foundation, Python Software.** What is Python? Executive Summary. [Online] 2016. [Citace: 31. 3 2021.] <https://www.python.org/doc/essays/blurb/>.
15. **jetbrains.** Get started. [Online] 2021. [Citace: 31. 3 2021.] <https://www.jetbrains.com/help/pycharm/quick-start-guide.html>.
16. **ShadowClaw20017.** Tkinter. [Online] 2021. [Citace: 31. 3 2021.] <https://wiki.python.org/moin/TkInter>.

6. Seznam obrázků

Obrázek 1 Rubikova kostka (1)	1
Obrázek 2 Megaminx (5)	2
Obrázek 3 Základní operace s Rubikovou kostkou (6)	3
Obrázek 4 Grafické znázornění začátečnické metody (7)	4
Obrázek 5 Grafické znázornění CFOP metody (9)	4
Obrázek 6 Grafické znázornění Roux metody (11)	5
Obrázek 7 Schéma propojení součástek	8
Obrázek 8 Průběh převrácení kostky	9
Obrázek 9 Pohled na rameno a vidlici seshora	10
Obrázek 10 Průběh otočení strany	10
Obrázek 11 Ukázka aplikace a korektního zadání rozložené kostky	12
Obrázek 12 Stav GUI po zadání první strany	13
Obrázek 13 Funkce posliData()	13