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

Rubik's cube solver

Maturitní práce

Lumír Blažek 4.E

Březen 2021

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, Arabská 14 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 Rubikovy 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. První byl samotný robot, druhou částí grafické uživatelské rozhraní, kde by uživatel zadal aktuální stav rozložené kostky. Z něho by se následně vygenerovaly instrukce, podle kterých by byl robot schopen kostku složit. Třetí částí byl samotný algoritmus, 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 Rubikovy kostky. Součástí práce bude vymyslet a navrhnout algoritmus, který bude ovládat robota tak, aby byl schopen ji složit. Uživatel by na začátku zadal do systému ručně aktuální rozložení kostky přes uživatelské prostředí, následně by se v počítači vypočítal algoritmus kroků, které by robot měl udělat pro složení kostky. Nakonec by se tyto kroky odeslaly do robota, který by následně dokázal otáčet kostkou tak, aby ji složil.

Obsah

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 Ernö 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.

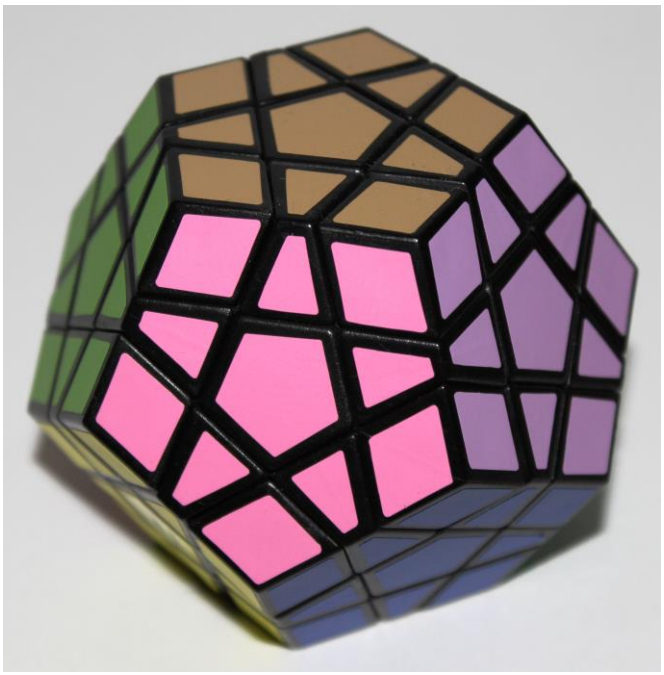
První testovní série 5000 kostek byla poprvé prodávána roku 1977 v Budapešti. Zprvu neměla od společnosti Trial moc velkou důvěru na úspěch, ale opak byl pravdou. Téměř okamžitě po uvedení do maďarských obchodů zaznamenala kostka enormní úspěch, takřka okamžitě byly všechny zásoby kostek vykoupeny.

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 3x3x3 kostky se k dnešnímu dni vyvinulo velké množství modifikací, jako například 2×2×2, 4×4×4, 5×5×5, 6×6×6 a 7×7×7 kostky, Megaminx (dvanáctistěn), Pyraminx (čtyřstěn), Square-1 a v neposlední řadě i siamské kostky (2 kostky spojené dohromady). (4)



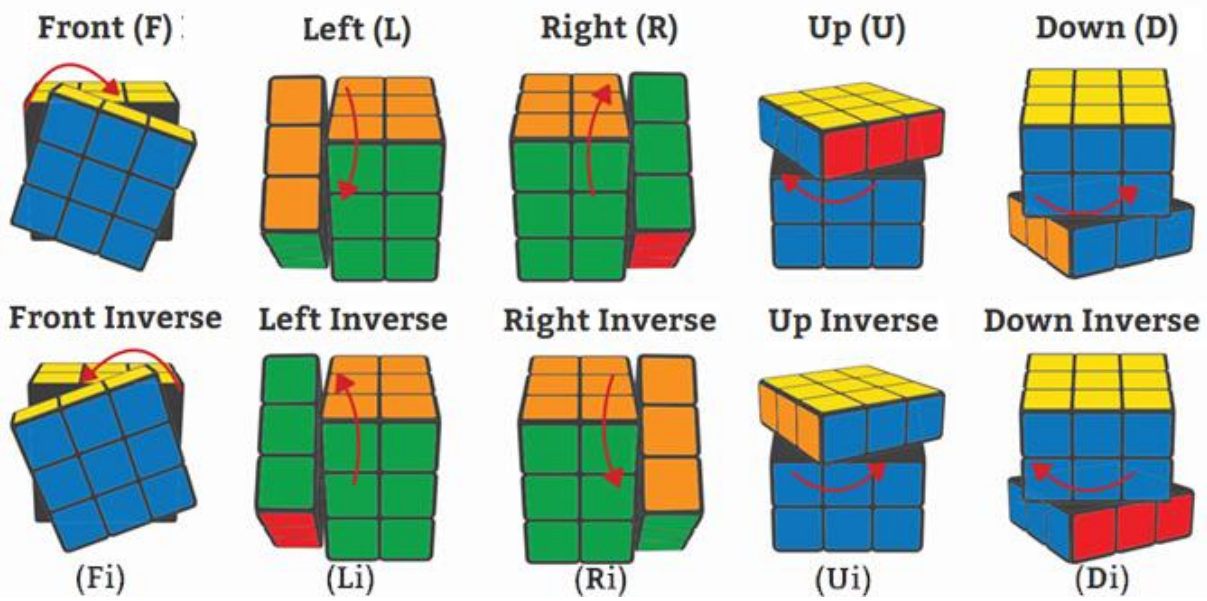
Obrázek 2 Megaminx (5)

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.

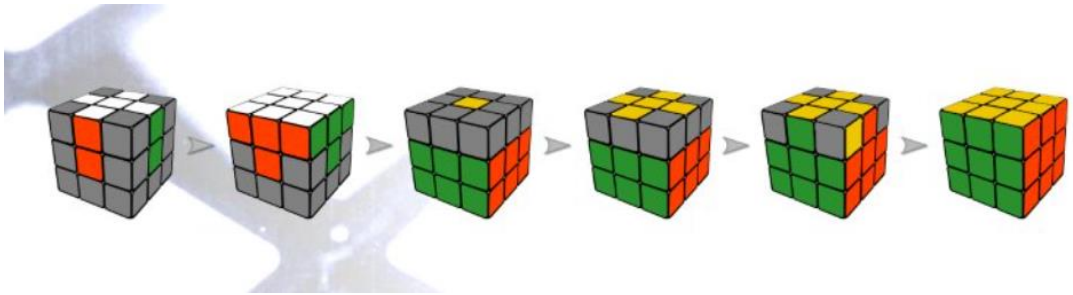
Před samotným skládáním si skládající osoba nejprve určí, která strana bude spodní a která strana bude přední. Jako spodní vrstva se tradičně volí buď bílá nebo žlutá strana, za přední se volí modrá strana. Skládající osoba pak následně po celou dobu skládání drží kostku tak, aby byla spodní strana vespod 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

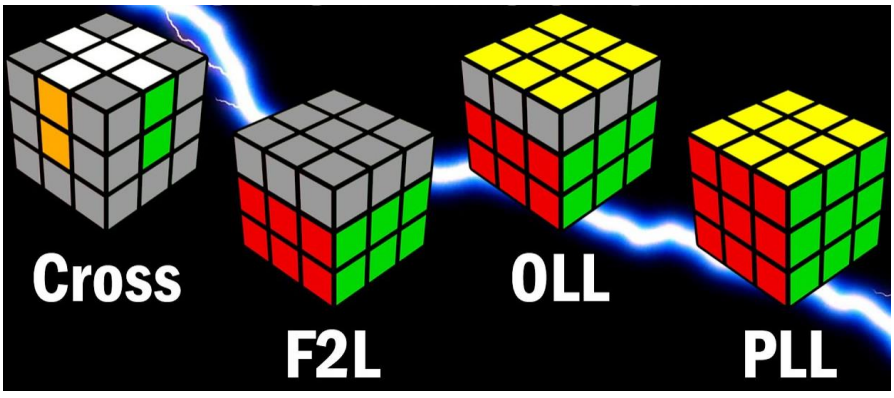
Základní myšlenka začátečnické metody je založena na metodě CFOP. Oproti metodě CFOP byl začátečnický algoritmus dosti zjednodušen, tak aby zvládl kostku složit naprosto každý. Algoritmus se skládá celkem ze 7 kroků, nejprve je složen tzn. „spodní kříž“, kdy se ve spodní vrstvě snaží řešitel hlavolamu 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 seděly. 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 rozestavění 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 navrhnuta 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)



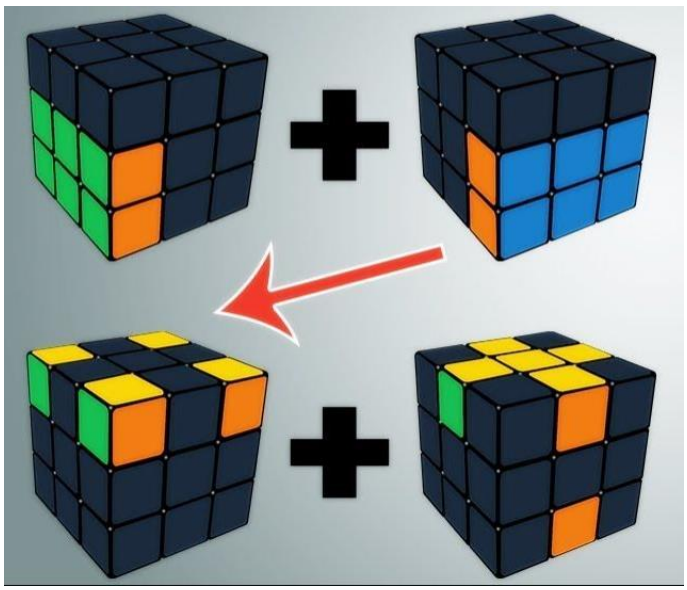
Obrázek 5 Grafické znázornění CFOP metody (9)

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 pomocí jednoho z 21 algoritmů přesunuty do správných pozic, čímž je skládání úspěšně dokončeno. (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 kdekoliv 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 složeny 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 nejkomplikovanější 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 je 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 postaveno 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 Shieldy. 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 vedle softwaru vytvořit i funkční verzi robota. První 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.

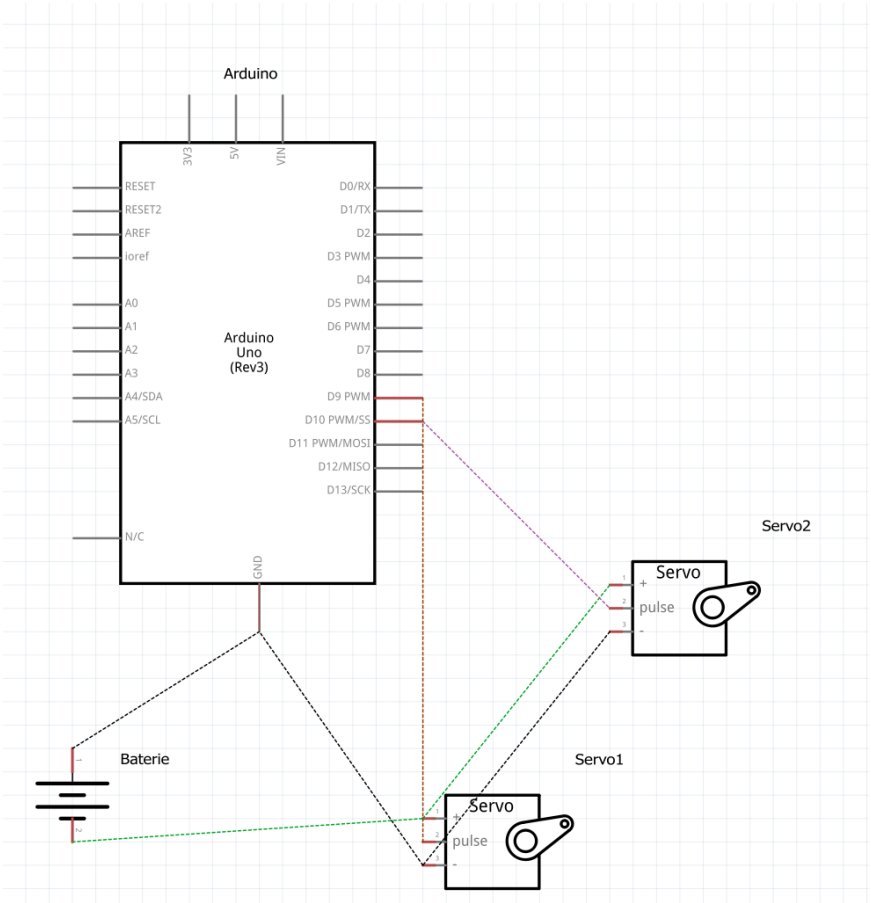
Pokud by se u projektu podařilo vyřešit problémy spojené s mechanickými nevýhodami servomotorů, jednalo by se o ideální volbu. Proto jsem u projektu rozhodl použít právě servomotory.

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.



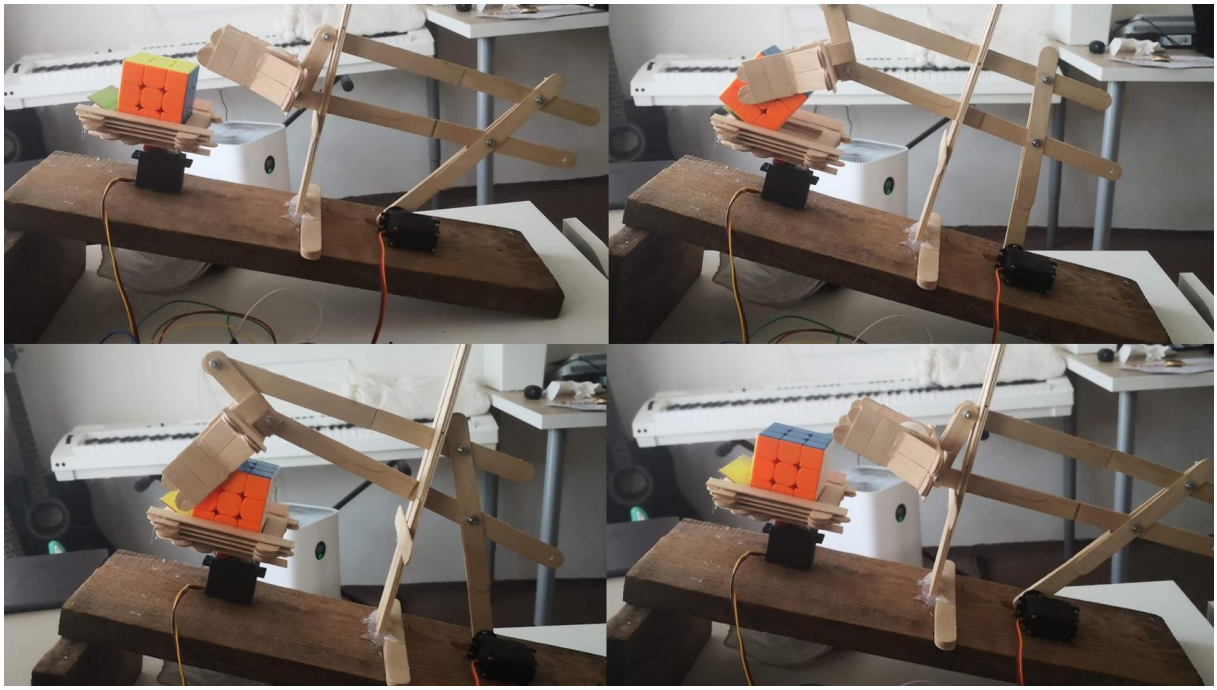
Obrázek 7 Schéma propojení součástek

3.4. Konstrukce robota

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

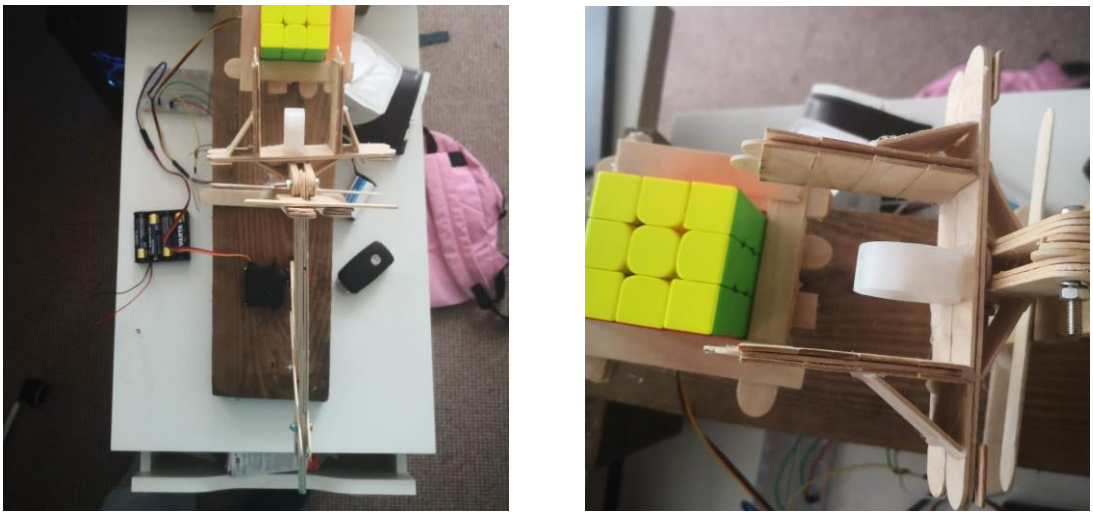
4.3.1. Rameno

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 převracení kostky

Rameno je podpořeno dvěma vzpěrami, které minimalizují vychýlení ramena z osy při otáčení stran kostky. Při převracení kostky může dojít hned k několika chybám, např. se rameno otočí moc rychle, takže je následně kostka vyvrhnuta ze základny, nebo se kostka zaklíní ve vidlici, takže se kostka nepřevrátí, nebo vypadne ze základny. Celý mechanismus je proto navrhnut tak, aby minimalizoval chybovost provedených operací. Při převracení 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.



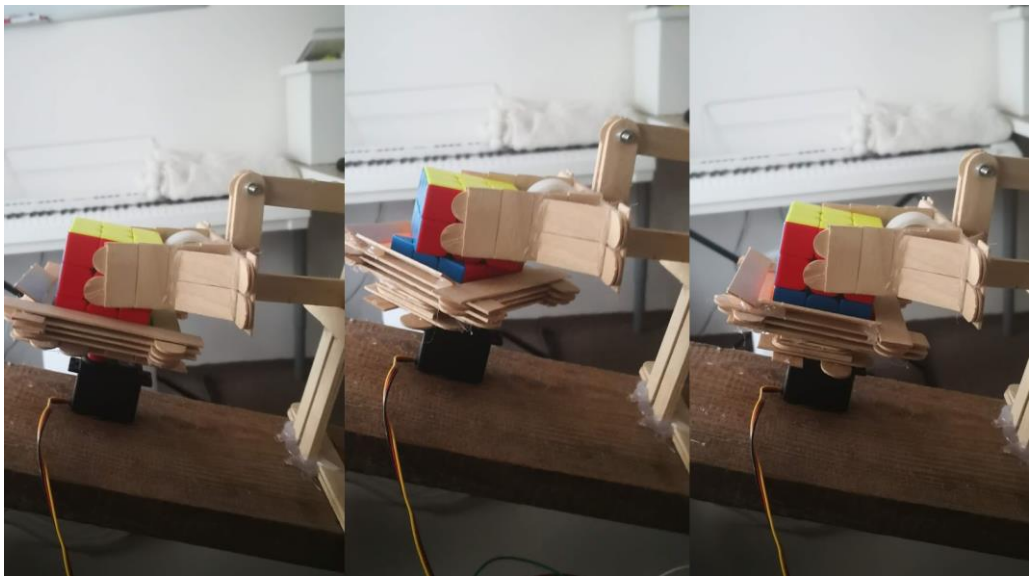
Obrázek 9 Pohled na rameno a vidlici seshora

4.3.2. Základna

Funkce základny jsou otáčení kostkou, držení kostky na místě, společně s ramenem převracení kostky a otáčení požadovanou stranou. Základna je vytvořena z několik lékařských dřívek a jednoho servomotoru.

Vzhledem k použití servomotorů je rotace základny omezena pouze na rozmezí od 0 do 180 stupňů. To nám vytváří problém. Pokud bychom otočili jednou stranou a následně bychom jí chtěli otočit znovu, tak by to čistě z mechanického hlediska nebylo možné. Základnou by totiž nešlo dál otočit ve stejném směru. Proto bude v našem programu nutné ošetřit to, aby se základna i s kostkou uvedly do výchozí pozice vždy po otočení stranou.

Základna disponuje několika nakloněnými rovinami, které napomáhají k minimalizování chybovosti převracení kostky. Celá základna i s ramenem jsou připevněny na nakloněné podložce, což vytváří ideální sklon pro převracení 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 zvládání 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 nemohly existovat, čímž se právě pro začátečníky stává čitelnější a srozumitelnější. Velkou výhodou Pythonu je také velké množství uživateli vytvořených knihoven, které z něho dělají velice silný nástroj pro vývojáře. Python byl zvolen do mého projektu, jelikož je pro něj 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.

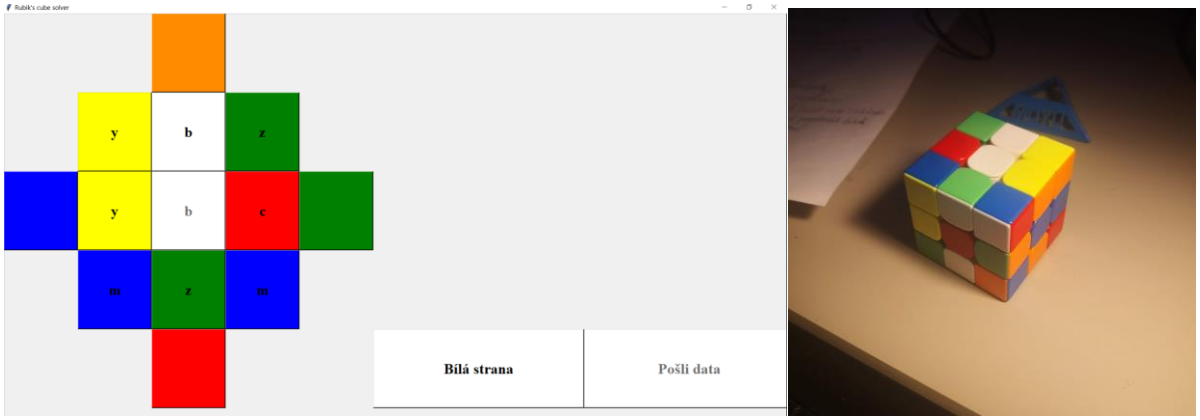
1.4.4. PyCharm ve verzi Proffesional

PyCharm je IDE vyvíjené českou společností JetBrain. 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

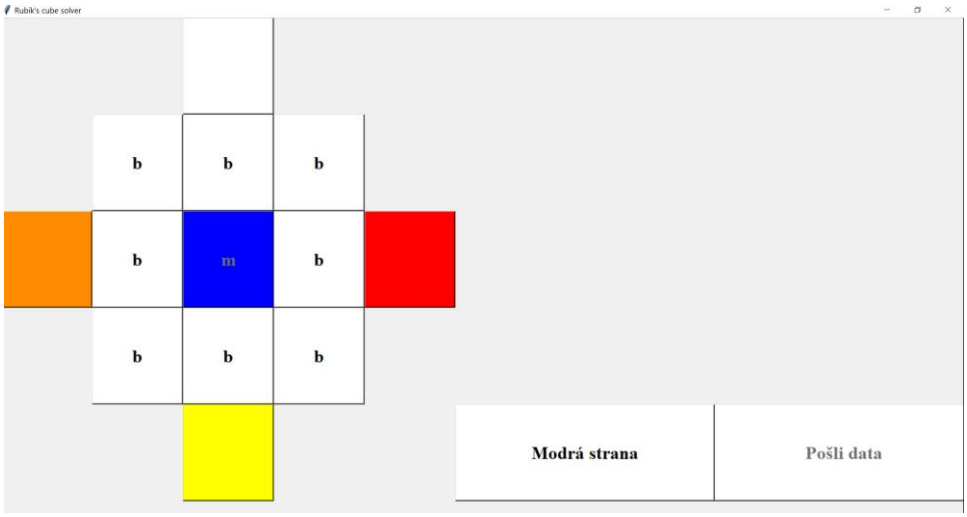
Samotná aplikace je strukturována do dvou částí, v první části jsou funkce spojené se zadáním informací od uživatele, ve druhé části jsou tyto informace odeslány do Arduina.

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 mu 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, potvrdí ho stiskem tlačítka nesoucího název ve formátu *„barva strany“*. 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, jež jsou zapsána na barevných políčkách. Tento postup provede uživatel celkem 6x. Po zadání žluté strany se všechna 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 při zapnutí aplikace za pomoci příkazu *serial.Serial(port, baudrate)*, kdy je jako první parametr uveden komunikační port, na kterém se nachází Arduino. Jako druhý parametr se udává baudrate, neboli počet bitů za sekundu, který se definuje při otevírání sériového portu na Arduinu. 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 Arduina, čímž je celé odeslání dat úspěšně dokončeno.

```
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. Arduino

5.1. Použité technologie

1.5.1. Arduino IDE

Arduino IDE je software určený pro vývoj aplikací pro programovatelné mikrokontrolery Arduino. IDE je napsané v Javě a svou strukturu zakládá na platformě Wiring. Podobně jako Wiring i Arduino využívá pro zápis kódu jazyky C a C++, u Arduina se ovšem jedná o zjednodušené varianty. Samotný Wiring, na kterém je Arduino založené, má svou specifickou strukturu kódu a velkou škálu vlastních knihoven, proto je někdy Wiring označován jako samostatný programovací jazyk. (16)

5.2. Struktura programu

V této kapitole si blíže představíme strukturu programu Arduina, který se stará o samotné skládání kostky. Program by se dal rozčlenit na tři části, z nichž má každá svůj specifický úkol.

2.5.1. Reprezentace kostky

S kostkou budeme muset v programu provádět stejné operace jako v reálném světě, proto je nutné zvolit adekvátní reprezentaci kostky. V mém programu jsem zvolil strukturu, kdy je nadefinováno pro každou jednotlivou stranu jedno pole. Každé pole obsahuje 9 prvků, každý prvek symbolizuje jedno políčko na dané straně. Tato struktura pohodlně umožňuje jak zápis příchozích dat z uživatelského rozhraní, tak i pohodný zápis základní operací, které jsou robotem prováděny.

2.5.2. Komunikace s GUI

Jak už bylo nastíněno v kapitole o GUI, tak celá komunikace mezi GUI a Arduinem probíhá za pomoci sériového portu. O celou komunikaci se stará funkce *prijemdatgui()*. V první operaci je zavolána funkce *prijmustr(),* která čte postupně písmeno po písmeni odeslaná data, ze kterých následně utvoří původní string. Následně je zavolána metoda *prepis_pole(),* která má za úkol převést přijmutý string do jednotlivých polí, čímž je komunikace mezi Arduinem a GUI úspěšně dokončena.

```
void prijemdatgui() {

  prijmustr();
  delay(1000);
  prepis_pole();
```

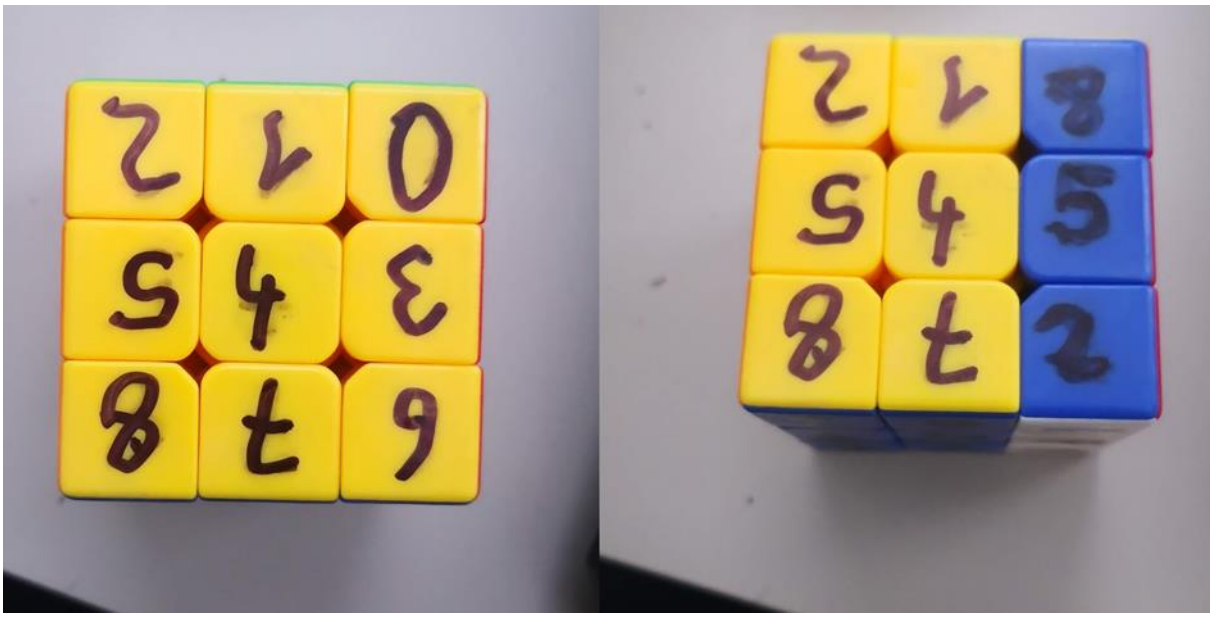
Obrázek 14 Ukázka funkce prijemdatgui()

2.5.3. Pohybové funkce

Pohybové funkce jsou takové, které se starají o pohyb servomotorů a simulaci stavu reálné kostky po otočení jakoukoliv stranou. Jsou strukturovány do tří vrstev. První vrstva je zastoupena pouze jednou funkcí, která se stará o kultivovaný pohyb servomotorů o určitý počet stupňů. Tuto vrstvu by bylo možné naprosto vynechat, jelikož jde při otáčení servomotorů zadat pouze finální potřebnou polohu. U tohoto řešení by ovšem nebylo možné ovládat rychlost otáčení, tudíž by docházelo k velké chybovosti při průběhu skládání kostky (např. vymrštění kostky ze základny). Funkce která ovládá servomotory se nazývá *pohni_servo(int start, int konec, int pin)* a obsahuje tři parametry. První parametr je výchozí pozice, ve které se aktuálně servomotor nachází. Druhý parametr je pozice, na kterou chceme dané servo otočit. Jako poslední parametr je pin, na který je daný servomotor připojen k Arduinu, díky čemuž je možné rozlišit servomotor pro otáčení základny a servomotor pro otáčení ramena. Díky těmto parametrům je možné lehce určit, jakým motorem budeme hýbat a o jaký počet stupňů se servomotor otočí. Ve funkci se následně provede otáčení motorem z pozice start do pozice konec přesně po jednom stupni, tak, že je mezi jednotlivými stupni je provedena přestávka, čímž je docíleno snížení rychlosti.

Druhá vrstva je složena z několika funkcí, které definují určitou pozici ramene nebo základny. U ramene se jedná o pozice, kdy rameno kostku překlopí, podrží, nebo ji pustí. Když se blíže podíváme na úkony, které jsou potřebné s kostkou dělat, jedná se o pohyby, které vždy vyžadují, aby byla kostka otočena o 90°. Proto má základna nadefinované pouze tři pozice, tj. 0°, 90° a 180°.

Ve třetí vrstvě jsou následně za pomoci dvou předchozích vrstev prováděny softwarově i hardwarově jednotlivé otáčení danou stranou. V první řadě je provedeno otáčení hardwarové. Vzhledem k tomu, že program počítá s tím, že bude kostka vždy postavena stejně jako při výchozí pozici a zároveň zde zůstává omezení 180°, je při každém otočení potřeba kostku do dané výchozí pozice dostat. Výchozí pozice je definována tak, že pokud je základna otočena do pozice 2, tj otočena o 90°, musí být dolní strana kostky bílá a strana směřující k ramenu modrá. Celé otáčení stranou je následně zapsáno jako posloupnost tahů, kdy je daná strana otočena a následně je kostka uvedena do výchozí pozice. Po ukončení těchto tahů je provedena simulace otáčení softwarově. To probíhá tak, že jsou jednotlivé pole stran přepsány na hodnoty, které byly původně zapsány v jiném poli. Pro příklad, pokud je otočeno pravou stranou nahoru, je žluté pole na pozicích 0, 3 a 9 přepsáno na hodnoty modrého pole na pozicích 8, 5 a 2. (viz obrázek 15)



Obrázek 15 Ukázka přepisu žlutého pole při otočení pravou stranou nahoru

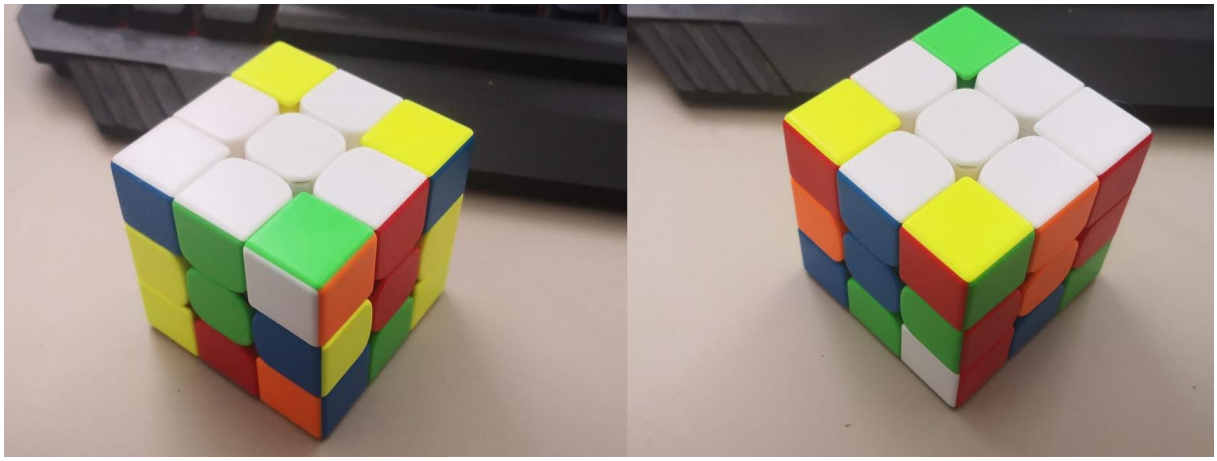
2.5.4. Algoritmus

Algoritmus pro skládání kostky tvořil nejdůležitější a největší část mé práce. Jako inspirace pro návrh algoritmu mi sloužil již popsaný začátečnický algoritmus ve druhé kapitole, který jsem lehce modifikoval proto, aby lépe korespondoval s mojí prací. Celý algoritmus probíhá v 9 krocích.

V první fázi jsou dvě východiska pro její úspěšné vyřešení. V prvním případě je už kostka ve stavu, kdy je složen korektně spodní bílý kříž. V tom případě je první i druhá fáze přeskočena rovnou na třetí fázi. V opačném případě složí robot horní bílý kříž. Program postupně zkontroluje, zdali se na jedné z 24 hran nenachází pole bílé. Následně se podle pozice nalezeného bílého pole provede sekvence tahů, tak, aby se toto pole nacházelo na jedné ze 4 hranových pozic žluté strany, při čemž je kontrolováno to, aby při této sekvenci tahů nedocházelo k rozbití již vytvořeného horního kříže. Po tom, co jsou všechna horní hranová pole obsazena bílými poli, je první fáze dokončena.

Obrázek 16 Příklad horního bílého kříže

Ve druhé fázi jsou horní bílé hrany zavedeny do správné pozice v dolní vrstvě. Začíná se od první strany, kdy je podle barvy nacházející se na horním hranovém políčku přední vrstvy otáčeno vrchní vrstvou tak dlouho, dokud tato barva nesouhlasí s barvou prostředního pole jedné ze stran prostřední vrstvy. Tento proces je prováděn postupně pro každou horní hranu, dokud není kostka uvedena do stavu, kdy je složen dolní bílý kříž, čímž je druhá fáze dokončena.

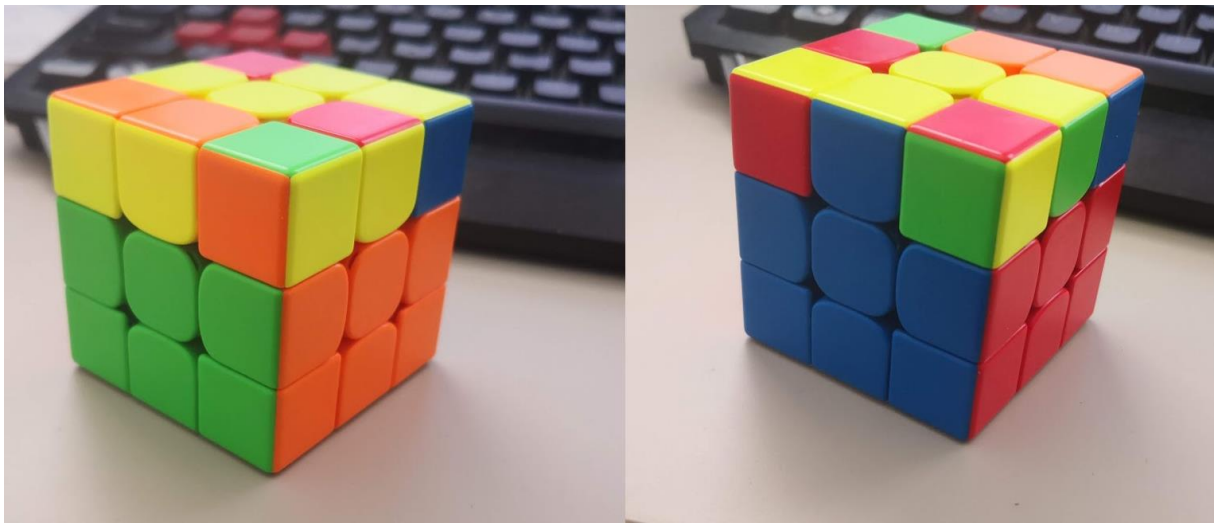


Obrázek 17 Příklad korektního bílého kříže

Ve třetí fázi je doplněna celá spodní vrstva o své rohy. Celkem se v této fázi nachází 24 polí, ve kterých se může vyskytovat bílá barva. Podle výskytu bílé barvy můžeme rozdělit situace na tři druhy. Bílé pole se nachází na jednom z osmi polí horní střední rohové vrstvy, bílé pole se nachází kdekoliv na spodním rohu, bílé pole se nachází v jednom ze čtyř horních rohových polí. Pokud se bílé pole nachází v jednom z osmi polí prvního případu, je velice lehké daný roh vložit do správné pozice ve spodní vrstvě. Proto se budeme snažit pole druhého případu dostat pomocí sekvence tahů do pozice prvního případu. Pokud se bílé pole nachází na pozici třetího případu, je toto pole převedeno do pozice druhého případu a při dalším průběhu kontroly kostky je převedeno do pozice prvního případu. Odtud je následně zavedeno do korektní pozice. Třetí fáze končí, je-li celá spodní vrstva hotová.

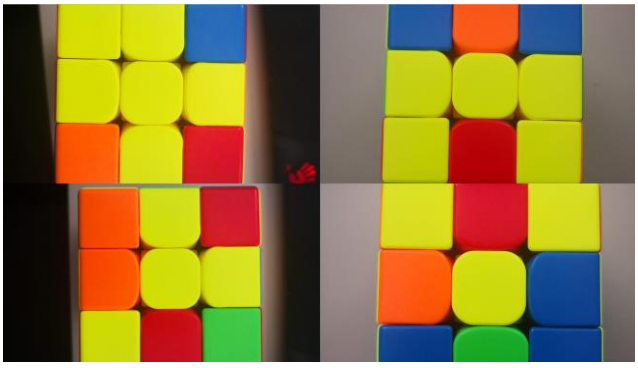
Obrázek 18 Složená spodní vrstva

Ve čtvrté fázi jsou doplněny hrany střední vrstvy. V této fázi nás zajímají pouze hrany horní a střední vrstvy, jelikož ty ve spodní vrstvě jsou již složené. Tyto hrany budeme dělit na dva základní typy, na ty, které mají žluté pole a na ty, které žluté pole nemají. Pozice žlutých hran nás v této fázi zajímat nebude, protože potřebujeme pouze doplnit nežluté hrany do správné pozice. Tyto hrany mohou mít dvě pozice, buď ve střední vrstvě, nebo v horní vrstvě. Pokud je tato hrana v horní vrstvě, existuje sekvence tahů, díky nimž je možné hranu horní vrstvy vyměnit za hranu střední vrstvy bez toho, aby byl předchozí progres jakkoliv narušen. Díky znalosti této sekvence tahů, je následně možné hrany střední a horní vrstvy prohazovat tak, aby byly hrany té střední dosazeny tam, kam patří. Poté, co jsou všechny hrany střední vrstvy na svém místě začíná pátá fáze.



Obrázek 19 Složená střední vrstva

V páté fázi je složen žlutý kříž. V tento okamžik jsou čtyři možnosti, jak může vypadat vrchní vrstva. Buď jsou dvě žlutá pole vedle sebe, nebo jsou dvě žlutá pole naproti sobě, nebo nejsou ve žluté vrstvě žádné žlutá pole. Nakonec mohou být v horní vrstvě i všechna žlutá pole, čímž by mohla být pátá fáze přeskočena. Proto, aby byl žlutý kříž složen, existuje postup tahů, po jejichž opakování se na vrchní vrstvě utvoří žlutý kříž. Bez toho, aby bylo cokoliv ve spodních dvou vrstvách poničeno. Pokud je žlutý kříž složen, tak je započata šestá fáze.

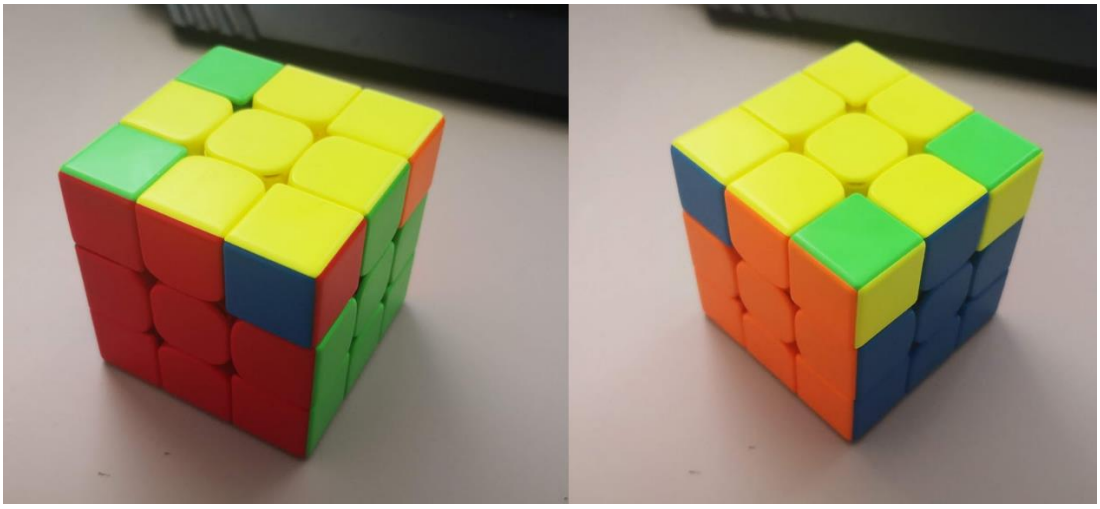


Obrázek 20 Čtyři možnosti, jak může v 5 fázi vypadat vrchní vrstva

Šestá fáze je pouze příprava pro sedmou fázi, protože se zde pouze posune horní vrstvou tak, aby horní modrá vrstva navazovala na střední modrou vrstvu.

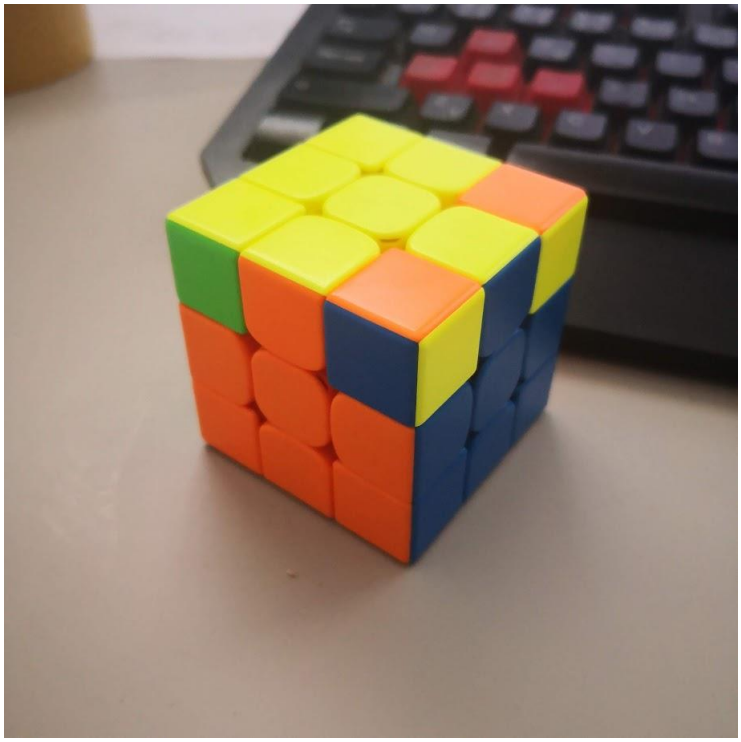
Obrázek 21 Správná návaznost horní modré hrany

V sedmé fázi je provedena korekce barev horních hran. Po dosazení modré hrany v šesté fázi došlo k zjednodušení analýzy pořadí barev. Z původních 4! pozic horních 4 barevných polí, nám k analýze zbývá po dosazení modré hrany pouze 3! pozic zbývajících tří hran horní vrstvy. Z toho nás jeden případ nebude zajímat, jelikož jsou v tomto rozložení všechny barvy v korektním pořadí. Proto bude stačit zkontrolovat pouze 5 případů, v jakých se může kostka nalézat, a následně prohazovat jednotlivé strany mezi sebou, tak, aby byl horní žlutý kříž opraven. Po opravě žlutého kříže se přechází na fázi osmou.



Obrázek 22 Příklad opraveného žlutého kříže

V předposlední fázi jsou rohy horní vrstvy přeskupeny tak, aby byl každý ve své korektní pozici. Korektní pozice je definována tak, že všechny barvy tohoto rohu odpovídají v libovolném pořadí jednotlivým barvám středů okolních třech stran, které na daný roh navazují (Viz obrázek 23).



Obrázek 23 Příklad korektní pozice horního rohu

V horní vrstvě se vždy nachází buď jeden roh v korektní pozici, nebo jsou v korektní pozici všechny rohy. Pokud se nachází na kostce pouze jeden korektní roh, program ho najde a následně provede posloupnost tahů, která postupně zbývající tři rohy uvede do korektní pozice také. Pokud jsou všechny rohy ve správné pozici, přejde program do poslední fáze.

V poslední fázi jsou všechny rohy orientovány tak, aby bylo jejich horní pole žluté. Poté, co jsou všechny rohy správně orientovány, je už jen horní vrstva za pomoci funkce ze šesté fáze otočena tak, aby horní modrá hrana barevně navazovala na modrý střed. Tím je kostka úspěšně složena.

Obrázek 24 Složená kostka

6. Závěr

6.1. Zhodnocení

Přesto, že jsem si byl vědom obtížného zadání, byla pro mě samotná práce náročná nad moje očekávání. V průběhu práce mě postihovala řada problémů, které mě uváděly do hluboké frustrace. Každopádně díky brzkému započetí práce s dostatečným časovým předstihem, byly její výsledky uvedeny do plnohodnotné a funkční verze. Po hardwarové stránce by bylo možné do budoucna k robotovi také dodělat senzor pro automatické snímání barev kostky. Dále by bylo možné předělat externí zdroj motorků z baterií na powerbanku, či napájení ze sítě, čímž by se docílilo snížení spotřeby baterií. Po softwarové stránce by bylo možné poupravit určité části algoritmu, čímž by se mohl snížit počet zbytečných kroků, a tím by se snížil i čas skládání kostky. Každopádně jsem s celkovým výsledkem práce nadmíru spokojen.

7. 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=dbcfd1ba-19c0-4c70-942b-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. **Wikimedia Foundation, Inc.** Arduino IDE. [Online] 2021. [Citace: 31. 3 2021.] https://en.wikipedia.org/wiki/Arduino_IDE.

17. **ShadowClaw20017.** Tkinter. [Online] 2021. [Citace: 31. 3 2021.] https://wiki.python.org/moin/TkInter.

8. Seznam obrázků