VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ELEKTRONICKÝ MINCOVNÍK PRO PROHERNÍ STROJ

COIN ACCEPTOR FOR LOOSE-IT-ALL MACHINE

DIPLOMOVÁ/BAKALÁŘSKÁ PRÁCE
BACHELOR'S/MASTER'S THESIS

AUTOR PRÁCE — Šimon Dolák
AUTHOR

VEDOUCÍ PRÁCE — Ing. Zdeněk Cejpek
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Šimon Dolák**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**Ing. Zdeněk Cejpek**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Elektronický mincovník pro proherní stroj

Stručná charakteristika problematiky úkolu:

Hazardní hry provází lidstvo již celá tisíciletí. Ačkoliv se může jednat o příjemnou kratochvíli či zajímavý námět k matematickým úvahám, mají hazardní hry na mnohé hráče negativní, až likvidační, dopady. Ke zmírnění těchto dopadů zajisté nejlépe poslouží prevence. Například prožití (pro)hry nanečisto.

Tato práce se zabývá návrhem a fyzickou realizací mincovníku jakožto součásti "výherního" automatu vypracovávaného v paralelně probíhající bakalářské práci. Automat bude sloužit pro prezentační a edukační účely.

Cíle bakalářské práce:

– krátká rešerše technologií rozpoznávání mincí,
– krátká rešerše komerčně dostupných elektronických mincovníků,
– návrh a realizace mechanického provedení hobby mincovníku (včetně žetonů, jež akceptuje),
– návrh a realizace rozhraní pro připojení k nadřazenému systému.

Seznam doporučené literatury:

VODA, Zbyšek a Martin STŘÍŽ. Průvodce světem Arduina. 2015. Bučovice. ISBN 978-80-87106-90-7.

ČERNOŠEK, M. Konstrukční návrh prodejního automatu na energetické tyčinky. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2011. 37 s. Vedoucí diplomové práce Ing. Michal Vaverka, Ph.D.

PRAŽÁK , J.: Návrh a konstrukce elektronického mincovníku. Absolventská práce, Vyšší odborná škola, Střední škola, Centrum odborné přípravy SEZIMOVO ÚST, Sezimovo Ústí, 2013.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Tato bakalářská práce se zabývá návrhem a konstrukcí elektronického mincovníku pro proherní stroj. Mincovník slouží pro verifikaci žetonů vlastního návrhu, jejich příjem a výdej. Je navržen pro nadřazený systém, se kterým komunikuje po sériové lince (USART). V práci je dále provedena krátká rešerše komerčně dostupných mincovníků, způsoby používané k rozpoznávání a verifikaci mincí.

ABSTRACT

This bachelor thesis deals with designing and construction of a coin acceptor for a loose-it-all machine. Coin acceptor is used for accepting, verifying and paying out proprietary tokens. It’s designed for a superior system, which communicates by a serial line (USART). Theoretical part of the thesis focuses on commercially avaiable coin acceptors and methods of verifying coins.

KLÍČOVÁ SLOVA

Mincovník, Arduino, Autodesk Inventor, 3D tisk, proherní stroj, verifikace mince

KEYWORDS

Coin acceptor, Arduino, Autodesk Inventor, 3D print, loose-it-all machine, coin verification

ÚSTAV AUTOMATIZACE A INFORMATIKY

2021

BIBLIOGRAFICKÁ CITACE

DOLÁK, Šimon. Elektronický mincovník pro proherní stroj. Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/131973. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Zdeněk Cejpek.

PODĚKOVÁNÍ

Tímto děkuji panu Ing. Zdeňkovi Cejpkovi za odborné vedení práce a cenné rady.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 22. 5. 2021 ……

Šimon Dolák

OBSAH

1 ÚVOD

Mince se používají jako platidlo již odnepaměti. Už jsme dlouho za dobou, kdy byl obchod založen na barteru. Dnešní svět je plný automatizovaných zařízení a transakce tak již často probíhají bez osobní přítomnosti obou stran. Ať už jde o parkovací automat, jídelní automat nebo automaty na jízdenky. Každý takový veřejný automat potřebuje mít zařízení, které dokáže platidlo přijmout, identifikovat a provést verifikaci. Právě takovému zařízení se bude práce věnovat.

V rešeršní části práce jsou popsány typy používaných mincovníků a metod identifikace a verifikace mincí, které využívají. Dále uvádí příklady mincovníků dostupných na českém trhu.

Cílem praktické části práce je navrhnout a vyrobit funkční mincovník pro nadřazený systém, který po vložení žetonu pomocí vybraných senzorů rozpozná jeho barvu, zkontroluje přítomnost magnetického jádra mince a následně minci zařadí do příslušného zásobníku. V případě falešné mince ji vrátí zpět uživateli. Pro navrhnutí konstrukce je použit program Autodesk Inventor. Veškeré nestandardní díly mincovníku byly vytištěny pomocí 3D tiskárny.

Ovládání mincovníku je uskutečněno programovatelnou deskou Arduino Mega 2560. Program byl napsán ve vývojovém prostředí Arduino IDE a je založen na stavovém automatu. V poslední části práce jsou popsány jednotlivé stavy mincovníku a komunikace pomocí příkazů po sériové lince s nadřazeným systémem.

2 MINCOVNÍKY

Mincovník se používá u automatických zařízení, která po vložení určité sumy provedou danou činnost. Po vložení mince musí mincovník verifikovat, zda je mince validní a zařadit ji do zásobníku. Pro některé aplikace je také třeba, aby bylo možné minci dostat ze zásobníku, tedy vyplatit určitou sumu. Verifikace mince spočívá v proměření vybraných parametrů mince a jejich porovnání s již známými charakteristikami mince, kterou má mincovník přijímat. Porovnávat můžeme nejrůznější vlastnosti mince, které dokážeme reprodukovatelně změřit. Mezi nejpoužívanější metody patří kontroly hmotnosti, průměru, tloušťky nebo magnetických vlastností mince.

2.1 Mechanické mincovníky

Hlavní výhodou mechanických mincovníků oproti elektronickým je, že k zachování běžného provozu není potřeba elektrická energie.

Obr. 1: Výdejník žvýkaček vybavený otočným mechanickým mincovníkem [2]

Mechanické mincovníky můžeme podle konstrukce rozdělit na dva typy, otočné a vysunovací. Otočné mincovníky můžeme najít například u starých automatů na žvýkačky (viz. obr. 1), kde jediný bezpečnostní prvek představuje kontrola velikosti mince. Vysunovací mincovník lze nalézt u stolního fotbálku. Správná mince propadne do zásobníku. Neplatnou minci lze uvolnit pomocí páčky nebo tlačítka a ta vypadne zpět ven.

Většina levných mechanických mincovníků jde ošálit mincí na provázku a pomocí drátku, který přidrží kolébku, aby mince po verifikaci nepropadla do zásobníku [1].

2.2 Mechanické komerční mincovníky dostupné na českém trhu

Při výběru mincovníku je třeba zohlednit několik faktorů. Nejpodstatnější je vědět, na co chceme mincovník využívat a také jak moc peněz do něj chceme investovat. Na českém trhu nenajdeme velké množství různých mincovníků a od toho se i odvíjí cena.

Lze pořídit levný otočný mincovník, ale u něj je velmi nízká bezpečnost vzhledem k omezené schopnosti verifikace mince. Mohli bychom ho použít u mechanických automatů, kde není hodnota mince příliš velká. Pořizovací cena mincovníku je 600 Kč.

Obr. 2: Komerční otočný mincovník [3]

Například mechanický mincovník MR 89, který je vhodný pro použití u stolního fotbálku nebo u kulečníkového stolu, lze pořídit za cenu kolem 2000 Kč. Tento mincovník slouží pouze pro přijímání jednoho typu mince [4].

Obr. 3: Komerční vysunovací mincovník [4]

Můžeme také naleznout kvalitnější mechanické mincovníky s cenou přibližně 9000 Kč, které dokážou rozeznávat 2 odlišné mince [5].

2.3 Elektronické mincovníky

Elektronické mincovníky využívají složitější principy pro verifikaci mince, takže je lze považovat za bezpečnější. Také díky mikroprocesoru umožňují rozeznávat více druhů mincí. Dokáží komunikovat s nadřazeným systémem o stavu zásobníků, tudíž jsou vhodné do automatických zařízení, kde je zapotřebí mince též vyplácet. Další výhoda je také mnohem menší šance na zaseknutí mince, což snižuje nároky na pravidelnou údržbu. Nevýhody těchto mincovníků představují vyšší pořizovací cena a závislost na napájení [1].

2.4 Elektronické komerční mincovníky dostupné na českém trhu

Obr. 4: Elektronický mincovník EU2 [6]

Mincovník EU2 dokáže rozeznat až 6 různých mincí najednou a má 2 banky, které se dají přepínat, takže je vhodný na provoz v příhraničních oblastech, kde by každá banka spravovala jiný typ měny. Dá se programovat pomocí počítače, ale také má tzv. Learning mode, který slouží ke stanovení parametrů vzorové mince bez potřeby použití PC či jiného programovacího zařízení. Prodejce nespecifikuje, jakým způsobem mincovník verifikuje mince. Tento typ se dá pořídit za cenu 2500 Kč [6].

Za cenu asi o 1000 Kč vyšší můžeme zakoupit elektrický mincovník Comestero RM5-G. Tento mincovník je dle prodejce velmi kvalitní a spolehlivý. Pro verifikaci používá 3 páry indukčních senzorů a tři páry optických senzorů, které se používají pro zpracování 6 různých parametrů týkajících se materiálu a hmotnosti mince. Jeho výhodou je automatický systém kompenzace optických senzorů pro změny teploty, napětí a osvětlení. Mincovník lze nastavovat z PC či specializovaného přenosného programátoru. [8].

Obr. 5: Elektronický mincovník Comestero RM5-G [9]

3 MOŽNOSTI VERIFIKACE MINCÍ

Hlavním účelem mincovníků je vydělávání peněz bez nutnosti přítomnosti lidského prodejce. Při běžném platebním styku můžeme měnu verifikovat sami pouhým pohledem, ale u mincovníku se musíme spoléhat na měření relativně snadno měřitelných vlastností mincí.

3.1 Měření rozměrů – mechanicky

U každého mincovníku najdeme nějaký způsob verifikace pomocí měření rozměrů. I velikost otvoru pro vložení mince se dá považovat za jeden ze způsobů ověření správného rozměru. Pokud je mince příliš velká, nemůže být do mincovníku ani vložena. Ve všech mincovnících je to první testování, kterým musí mince projít. Kontrola rozměru probíhá tak, že mince menšího rozměru propadne, popřípadě se zasekne a musí být uvolněna páčkou [11].

3.2 Měření hmotnosti

Hmotnostní verifikace se často vyskytuje v mechanických mincovnících. Jak již bylo zmíněno, ve velkém počtu mincovníků je tento způsob verifikace spojený spolu se zařízením, které také pozná, že je mince menšího průměru, jakožto dodatečný způsob kontroly validity mince. Tato kontrola je realizovatelná snadno tzv. kolébkou, jež se vlivem dostatečné hmotnosti mince pootočí a propustí minci dostatečné hmotnosti na jinou trasu, než minci příliš lehkou [11].

3.3 Magnetické vlastnosti

Poslední typ verifikace u mechanických mincovníků využívá magnetických vlastností mincí. Například ve Spojených Státech, kde mince nejsou magnetické, tak magnet zachytí jakýkoliv jiný magnetický předmět, který by prošel předchozí testy, tedy shodoval by se rozměry i váhou [12].

Obr. 6: Mechanický mincovník [13]

3.4 Materiálové vlastnosti

Mechanické mincovníky časem ztratili na bezpečnosti kvůli stále inovativnějším cestám, jak tyto bezpečnostní kontroly obejít. Z tohoto důvodu se začaly používat preciznější metody testování. U těchto metod se snažíme proměřit hodnoty charakteristické pro materiál dané mince. Následně je pomocí mikroprocesoru porovnáme s již známými vlastnostmi mincí, které chceme přijímat. Nejpoužívanější způsob je kontrola materiálu mince pomocí indukčnosti. Spočívá na principu vložení jádra (tedy mince) do cívky. Po vložení se změní velikost permeability jádra cívky v závislosti na materiálu a velikosti mince, tedy i indukčnosti cívky, jež je elektronicky vyhodnocena [1].

3.5 Měření rozměrů – elektronicky

Další z možností je měření velikosti mince pomocí dvou laserových zábran. Mince, která se pohybuje po rampě přetne paprsek u prvního laseru, pokračuje dál a přetne i paprsek druhého. Z času mezi aktivací obou senzorů a známých poměrech dráhy mincovníku můžeme usuzovat o rozměrech mince, přesněji tedy za předpokladu, že minci lze považovat za homogenní válec. Podobným způsobem je možné také měřit čas pádu mince [14].

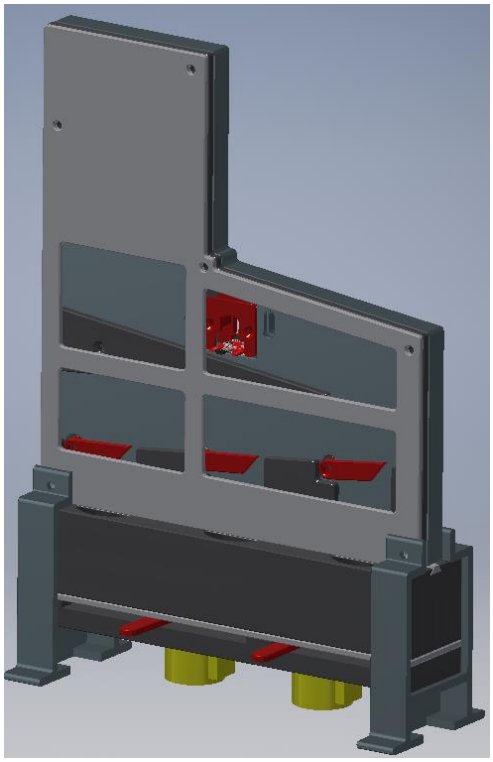
3.6 Verifikace bankovek

U verifikace bankovek můžeme použít optickou verifikaci, kdy kamera snímá malé pixely, které jsou různě rozmístěny na bankovce a mají různé rozměry. Senzor hledá tyto vzory, aby rozpoznal, co za bankovku bylo vloženo. Některé bankovky jsou fluorescenční, můžeme je osvítit ultrafialovým světlem a snímat takto ovlivněný obraz, abychom zjistili kompozici materiálu. Také lze u některých bankovek využít inkoustu, který má feromagnetické vlastnosti. Bankovka projde kolem permanentního magnetu, inkoust se zmagnetizuje a magnetický senzor pak měří zbytkový magnetismus v částicích inkoustu [15].

Obr. 7: Bankovka osvícená UV světlem [10]

4 SOUČÁSTI MINCOVNÍKU

Cílem práce bylo navrhnout konstrukci funkčního mincovníku, který bude sloužit pro nadřazený systém. Je potřeba, aby rozeznával různé druhy žetonů, zařadil je do příslušných zásobníků a vydával mince nazpět. Mincovník také musí s nadřazeným systémem komunikovat. Konstrukce, která byla vytisknuta pomocí 3D tiskárny, se skládá ze samotného těla mincovníku, kulis pro dráhu mince, krytu, zásobníku a příslušných držáků na senzory. Dále se zde nachází samotné senzory na rozpoznání mince, senzory přítomnosti a motory, které slouží pro operace s mincemi. Program, který má na starost chod mincovníku, běží na vývojové desce Arduino.



Obr. 8: Návrh mincovníku

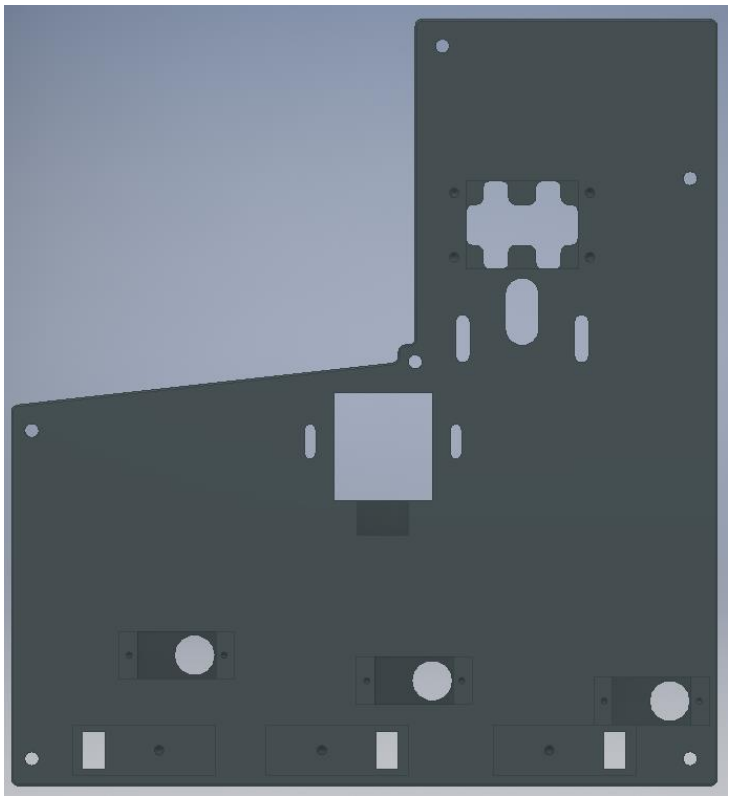
4.1 Mince

Pro mincovník je potřeba vytvořit mince, které se budou rozpoznávat. Na vyrobení těchto mincí byl použit 3D tisk. Mince se skládá ze 3 vrstev, a to dvou bočních dílů s vytištěným barevným vzorem a střední částí, jež obsahuje ocelovou podložku M10, tedy feromagnetické jádro se specifickou magnetickou signaturou – mince projíždějící okolo Hallova senzoru generuje dvojici pulzů.

Obr. 9: Mince

4.2 Šasi mincovníku

Základní konstrukční součástí mincovníku je jeho šasi, tedy deska, na které bude osazena většina součástí mincovníku, tedy motory, senzory a kulisy. Je důležité, aby otvory pro senzory a matice byly co nejmenší, protože v opačném případě by se mohla mince v těchto oblastech zaseknout. Otvory pro krokový motor s turniketem a pro držák Hall Effect senzoru jsou navrženy tak, aby se dala upravovat pozice senzoru a motoru kvůli snadnému jemnému nastavení.



Obr. 10: Šasi mincovníku

4.3 Kulisy

Kulisy vymezují dráhu mince, která musí projít kolem obou senzorů. Je potřeba zajistit, aby sklon dráhy mince byl dostatečný na to, aby se po nich mohla mince samovolně pohybovat. Kulisy jsou uzpůsobeny pro mince o průměru 30 mm a maximální tloušťce 3,5 mm.

4.4 3D tisk

Všechny konstrukční prvky jsou velmi specifické a jejich výroba konvenčními metodami by byla dosti náročná a pracná. Z toho důvodu byl pro výrobu zvolen 3D tisk. Konstrukční prvky byly navrženy tak, aby bylo možné je vyrobit pomocí této metody. Musí se počítat s mírnou nepřesností tisku, a proto bylo nutností navrhnout otvory pro šrouby a ostatní součásti větší zhruba o 0,5mm.

Princip 3D tiskárny spočívá v postupném přidáváním materiálu na rozdíl od výroby například pomocí CNC frézky, kdy materiál ubíráme. Tiskárny se dělí podle metody 3D tisku – FDM (Fused Deposition Modeling) a SLA(Stereolithography) [27].

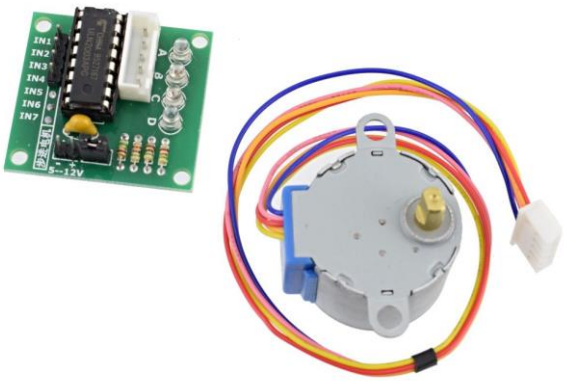
Obr. 11: 3D tiskárna Prusa i3 MK3 [28]

4.5 Krokové motory

Krokový motor se skládá z rotoru a statoru. Stator se skládá z cívek navinutých na magnetickém jádru v pevně definovaných pozicích kolem rotoru. Rotor je pak osazen buď dílem z permanentního magnetu (motor s aktivním rotorem), magneticky měkkým jádrem (VR – variable reluktance motor či pasivní krokový motor) či kombinaci obojího (hybridní krokový motor). Rotor s hřídelí se natáčí do pozice, při které jeho poloha odpovídá poloze s nejmenší možnou energií. Magnetické pole statoru může držet stanovenou pozici nebo rotovat. Rotace magnetického pole je dosaženo střídavou aktivací a deaktivací statorových cívek [16]. Lze si takto představit krokový motor jako několik elektromagnetů, které přitahují magnet rotoru.

4.5.1 Motor použitý v mincovníku

V práci byli využity 3 krokové motory 28BYJ-48 s řadičem. Jedná se o unipolární motor, který přeměňuje elektrické impulzy na mechanický pohyb. Výhodou motoru je snadná a levná realizace otočení o libovolný úhel. Motory jsou napájeny z externího zdroje 5V a ovládání jednotlivých kroků provádíme pomocí 4 pinů, kterými jsou přímo ovládány spínací tranzistory čtyř statorových cívek motoru.



Obr. 12: Krokový motor 28BYJ-48 [17]

Jak již bylo zmíněno, krokový motor ovládáme pomocí impulzů na řídících pinech. V programu jednotlivé kroky vyvoláváme následující funkcí, jedná se o jeden krok z 8. Celá osmice těchto funkcí realizuje posunutí motoru o cca. 6 stupňů (přesněji vykoná 1 plnou otáčku motoru, jež je na výstupní hřídel přenesena přes převodovku s převodovým poměrem 1:64).

```
void krok1(int a, int b, int c, int d){
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  delay(velocity);
}
```

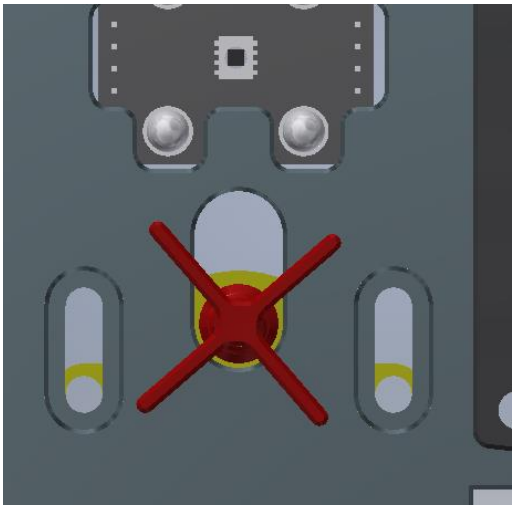
4.6 Modelářské servomotorky

Servomotory nám umožňují nastavit přesnou polohu ovládaného mechanismu a následně držet tuto polohu. Výhodou servomotorů je jejich malý rozměr a velký točivý moment, vyvozený díky převodovce s vysokým převodovým číslem. Tyto servomotory jsou obvykle konstruovány tak, aby se dokázaly natočit v rozmezí 90° či 180°, výjimečně i více. Existují však i kontinuální verze sloužící k udržování zadané rychlosti [7]. Velikost úhlu natočení určíme délkou zaslaného impulsu [18]. U mincovníku jsme použili servo SG90 9g.

Obr. 13: Servo SG90 9g [21]

4.7 Turniket

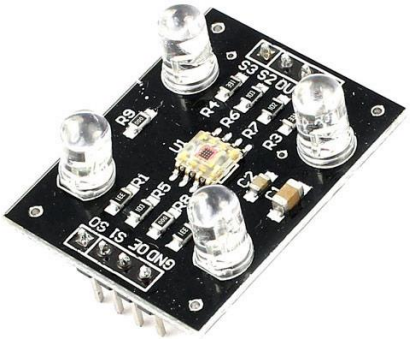
Abychom mohli změřit barvu mince, potřebujeme minci na okamžik zastavit v jedné pozici. Provedeme změření a jakmile víme, co za minci se v mincovníku nachází, pomocí krokového motoru turniketem otočíme o 90°. Druhou funkcí u turniketu je zachytit další vhozené mince a pozdržet je, dokud nebude dokončeno zpracování předchozí mince.



Obr. 14: Turniket pod detektorem barvy

4.8 Detektor barvy

V projektu byly využity 2 způsoby verifikace rozeznávání mincí. Jedním z nich je stanovení barvy okraje mince. Ke změření barvy okraje mince byl použit modul se senzorem TCS230. Senzor má pole s 64 fotodiodami, z nichž 16 je s červeným filtrem, 16 s modrým filtrem, 16 se zeleným filtrem a 16 s žádným. Také se zde nachází 4 bílé LED diody, které slouží pro zajištění definovaného osvětlení testovaného předmětu. Princip spočívá v nasvícení objektu, světlo se odrazí zpět do senzoru a ten následně změří intenzitu jednotlivých barevných složek R, G a B. Naměřenou informaci senzor poskytuje nadřazenému systému po čtyřech samostatných vodičích. Na každém vodiči generuje napěťové pulzy, jejichž délka je úměrná naměřené intenzitě dané složky dopadajícího světla [19]. V práci tento senzor též plní funkci prostého senzoru přítomnosti mince na turniketu, tedy mince čekající na zpracování.



Obr. 15: Detektor barvy TCS230 [20]

Pro určení barvy mince potřebujeme jednotlivě načíst frekvence pro každou ze složek barevného spektra. V programu na to budeme používat funkce loadRed, loadGreen a loadBlue.

```
int loadRed(){
  digitalWrite(pinColorMeasure_S2, LOW);
  digitalWrite(pinColorMeasure_S3, LOW);
  delay(50);
  return pulseIn(pinColorMeasure_Out, LOW);
}

int loadGreen(){
  digitalWrite(pinColorMeasure_S2, HIGH);
  digitalWrite(pinColorMeasure_S3, HIGH);
  delay(50);
  return pulseIn(pinColorMeasure_Out, LOW);
}

int loadBlue(){
  digitalWrite(pinColorMeasure_S2, LOW);
  digitalWrite(pinColorMeasure_S3, HIGH);
  delay(50);
  return pulseIn(pinColorMeasure_Out, LOW);
}
```

Pomocí změřených hodnot je pak možné rozeznat, jaká mince se nachází v mincovníku. Jak již bylo zmíněno, detektor barvy slouží i jako optický senzor, proto je nutné rozpoznat, jestli se před detektorem nachází mince nebo ne. Pomocí krytu mincovníku se snažíme zamezit, aby okolní podmínky ovlivňovaly přesnost měření. Tímto způsobem zjistíme hodnoty RGB pro jednotlivé mince a hodnoty, když se v mincovníku žádná mince nenachází. Funkcí getCoinColor určíme typ mince.

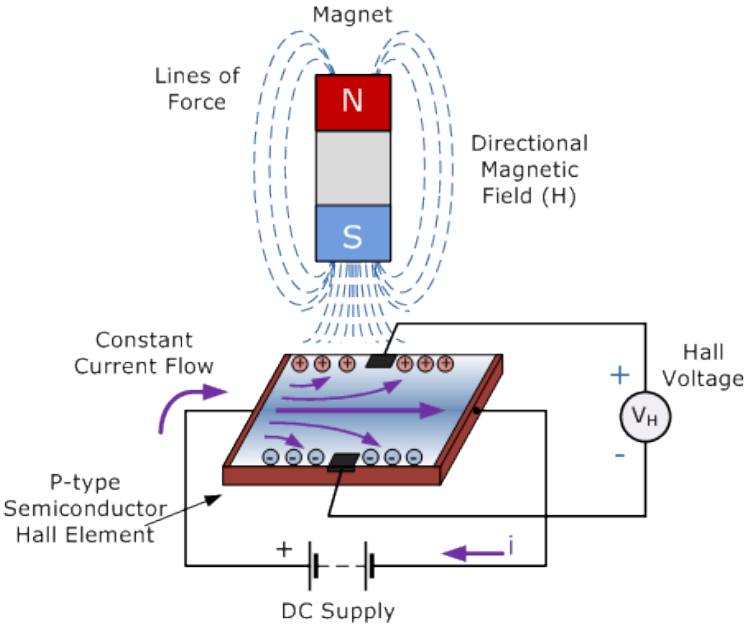
```
Color getCoinColor()
{
    int redMeasured = loadRed();
    int greenMeasured = loadGreen();
    int blueMeasured = loadBlue();
    Color col;
    if (redMeasured > 100 | greenMeasured > 100 | blueMeasured > 100){
    Serial.print(" Žádná mince");
    Serial.println();
    col = NO_COL;
    }
    else{
       if (redMeasured < 50) {
        Serial.print(" | Detekce cervene. ");
        Serial.println();
        col = RED;
        }
       else if (greenMeasured < 60) {
        Serial.print(" | Detekce zelene. ");
        Serial.println();
        col = GREEN;
        }
       else if (blueMeasured < 60) {
        Serial.print(" | Detekce modre. ");
        Serial.println();
        col = BLUE;
        }
       else {
        Serial.print(" | Detekce mince. ");
        Serial.println();
        col = UNKNOWN_COIN_COL;
        }
     }

    return col;
}
```

4.9 Hall Effect sensor

Hallův jev je nejpoužívanější metoda pro měření magnetického pole a Hall Effect senzory jsou velmi populární a mají spoustu aplikací. Mohou být použity v automobilech například pro měření rychlosti otáčení kola nebo jako senzor pro zjištění pozice klikového hřídele. Dále se dají použít jako spínače nebo senzory přítomnosti.

4.9.1 Hallův jev

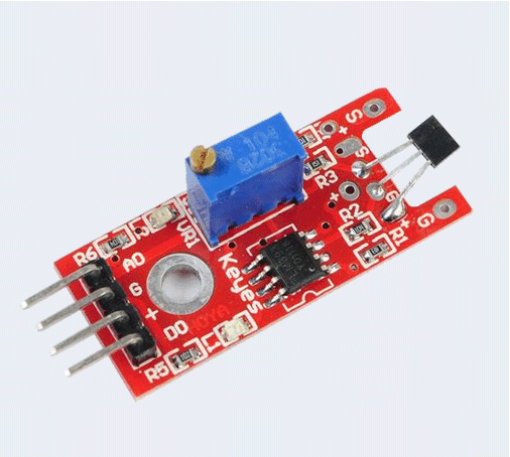
Pokud elektrický proud teče deskou, elektrony přirozeně proudí z makroskopického hlediska přímou čarou z jedné strany na druhou. Po přiložení magnetu kolmo k desce začnou elektrony interagovat s magnetickým polem a odchýlí se od původní přímé trajektorie k jednomu okraji desky, elektronové díry pak na druhý. Mezi kraji desky můžeme tedy naměřit napětí, které se nazývá Hallovo, a tento napěťový signál dále zpracovávat [22].



Obr. 16: Hallův jev [29]

4.9.2 Hall Effect senzor použitý v mincovníku

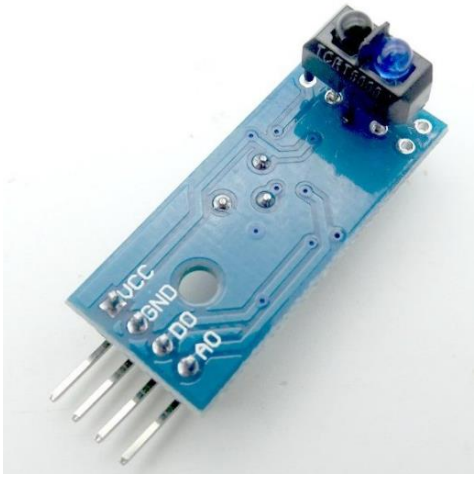
Existují Hall Effect senzory analogové a digitální. Analogové senzory v závislosti na intenzitě magnetického pole dávají různé hodnoty napětí, kdežto digitální nám pouze dávají informaci 0 nebo 1, tedy zda magnetické pole dosáhlo požadované úrovně či nikoliv [22]. V projektu použijeme právě digitální verzi senzoru. Kdybychom chtěli mít přesnější verifikaci, bylo by možné použít analogový, který by mohl poskytnout naměření dat vhodných ke zpracování technikami z kategorie machine learning. Pro účely této bakalářské práce však postačí digitální, který bude detekovat přítomnost feromagnetického materiálu, tedy ocelové podložky vlepené v používaných žetonech.



Obr. 17: Hall Effect senzor [23]

4.10 Senzor sledování čáry

Pro potvrzení, že mince propadla do zásobníku, byl využit infračervený senzor sledování čáry s LM393. Senzory jsou umístěny pod padacími dveřmi u zásobníků. Senzor se skládá z fotodiody a infračervené diody. Infračervená dioda nasvětluje snímaný prostor infračerveným zářením, které se odráží s intenzitou závislou na přítomnosti/vzdálenosti povrchu a jeho odrazivosti. V-A charakteristika fotodiody se mění v závislosti na intenzitě odraženého světla. Úbytek napětí na diodě s rostoucí intenzitou odraženého světla klesá a pomocí komparátoru LM393 je napětí prahované na dvě logické úrovně.

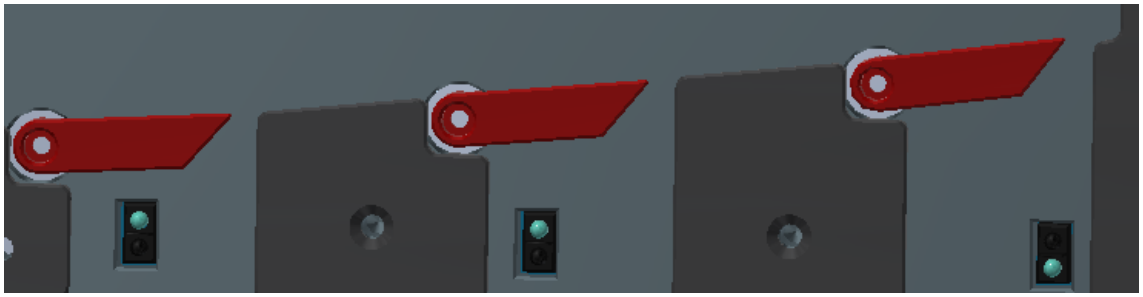


Obr. 18: Senzor sledování čáry [24]

4.11 Padací dveře

K propadení mince do zásobníku potřebujeme součást, která zablokuje cestu ven z mincovníku a zároveň umožní volný průchod dále do příslušného zásobníku. Slouží nám k tomu právě tyto padací dveře ovládané modelářským servomotorkem. Vzhledem

k tomu, že máme 3 zásobníky, budeme také používat 3 servomotory se 3 padacími dveřmi. Padací dveře se otevřou, když senzor barvy a hallův senzor identifikují platnou minci (typu příslušného daným padacím dveřím). Pro zavření dveří je využit signál ze senzorů sledování čáry detekujících přítomnost mince, tedy její propadnutí do zásobníku.



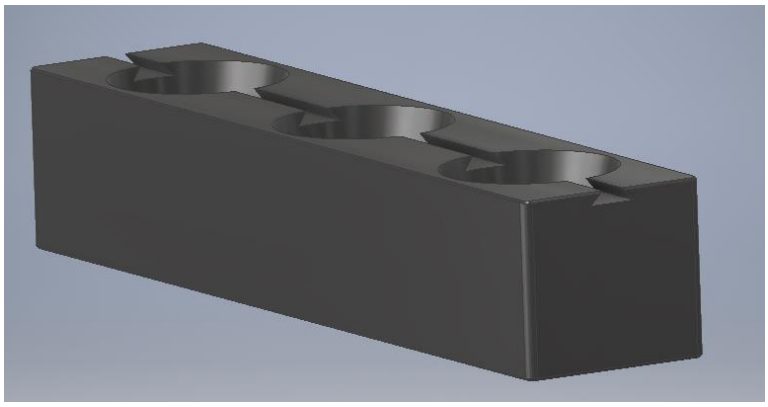
Obr. 19: Padací dveře se senzory

Funkce ovládající otevírání padacích dveří pro každou z barev pomocí servomotorů pak v programu vypadá následovně:

```
void openRelevantTrapDoor(Color col)
{
  switch (col)
        {
          case RED:
            openRedTrapDoor();
            break;

          case GREEN:
            openGreenTrapDoor();
            break;

          case BLUE:
            openBlueTrapDoor();
            break;
        }
}
```

4.12 Zásobník

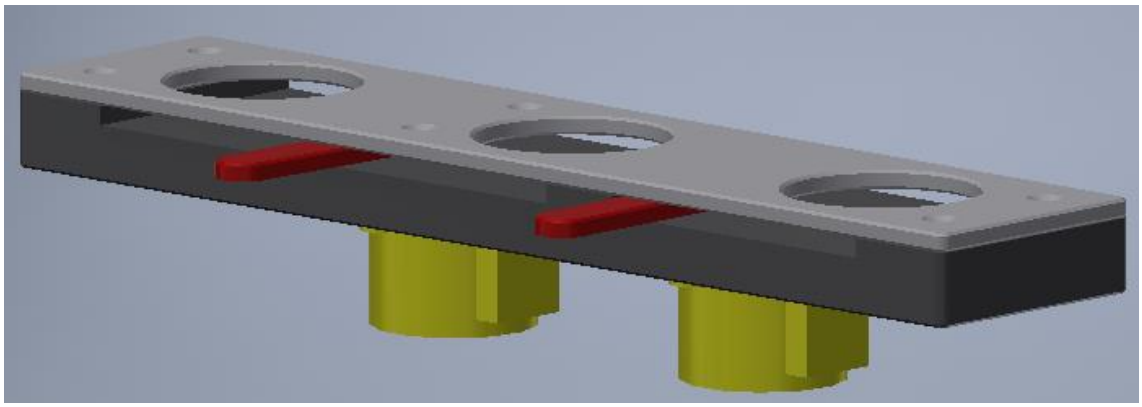
Pro skladování mincí je použit jednoduchý zásobník. Zásobník má otvory na 3 různé typy mincí. Lze ho jednoduše vysunout, odebrat či vložit mince do zásobníku a zase zasunout na původní pozici. Zásobník též slouží jako zdroj k vyplácení mincí.



Obr. 20: Zásobník na mince

4.13 Vyhazovací zařízení

Jelikož se jedná o mincovník pro proherní stroj, je zapotřebí, aby bylo možné také mince ze zásobníku vydávat uživateli. Vydávání bylo vyřešeno pomocí 2 krokových motorů s pákami, které vytlačí minci ven. Mechanismus byl navržen tak, že jeden krokový motor umí vyhazovat až 2 různé typy mincí. V programu je to vyřešeno tak, že při vydání příslušné mince si mincovník pamatuje polohu, aby při vydávání další mince otočil motor v opačném směru než při předchozím vyplácení. Oba motory se otáčejí pouze po krocích o velikosti 180°.

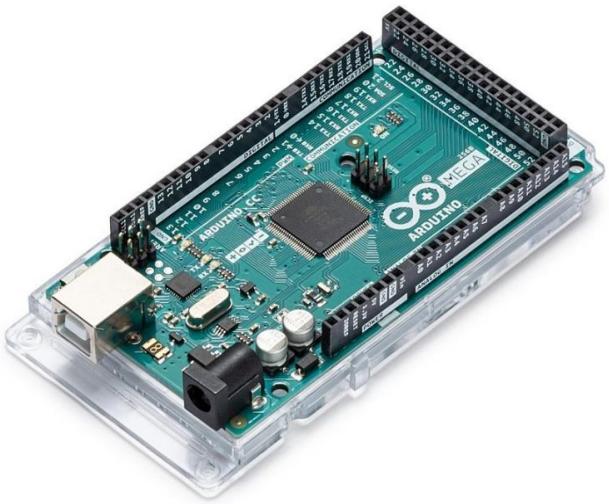


Obr. 21: Vyhazovací zařízení

5 OVLÁDÁNÍ MINCOVNÍKU

5.1 Arduino

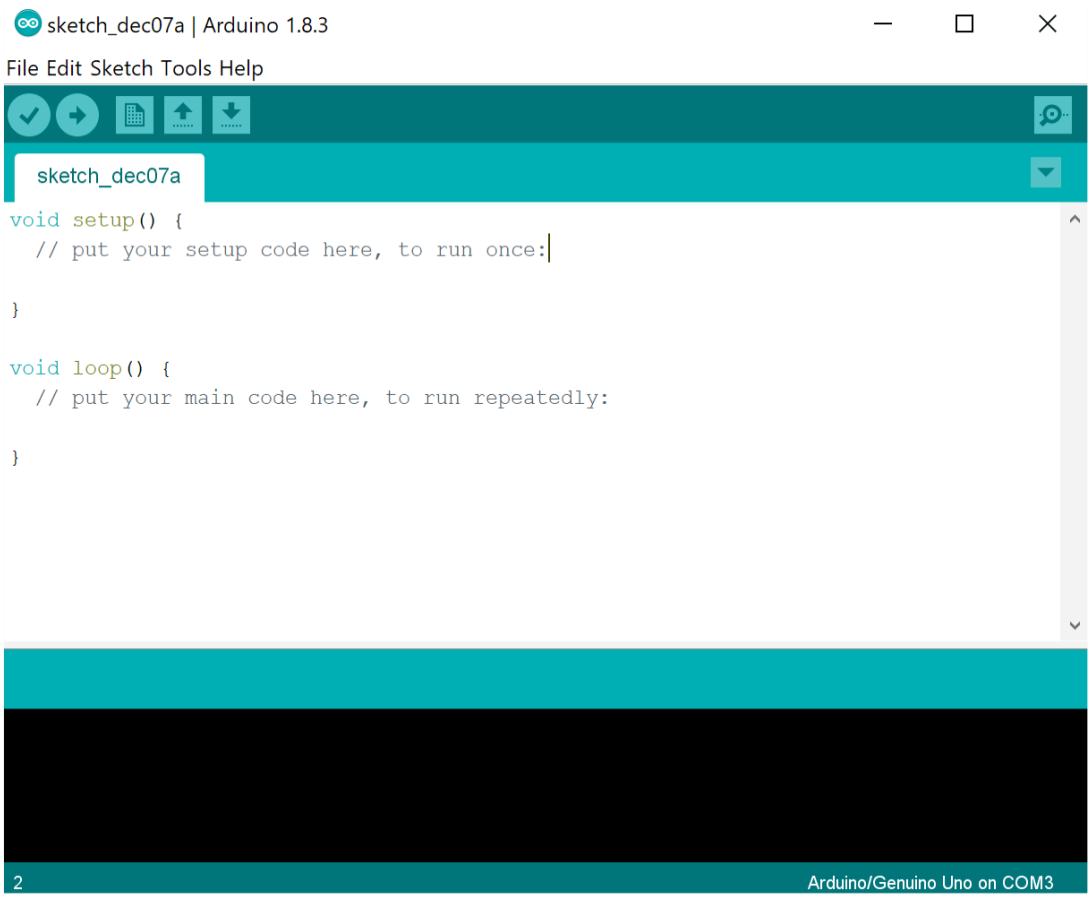
Pro ovládání mincovníku bylo využito vývojové desky Arduino. Arduino je open-source elektronická platforma založená na hardwaru a softwaru, který se jednoduše používá. Deska je založena na mikrokontrolerech ATmega od firmy Atmel. V práci budeme používat konkrétně desku Arduino Mega2560. Tato vývojová deska je vhodná pro projekty, které vyžadují větší velikost paměti a větší množství I/O pinů. Právě kvůli počtu pinů budeme využívat právě tuto desku. Deska nabízí 54 digitálních a 16 analogových pinů, což je pro tento projekt dostatek [25]. Využijeme celkem 24 digitálních pinů – 12 na ovládání krokových motorů, 3 na ovládání modelářských servomotůrků, 3 signály ze senzorů sledování čáry, 5 na ovládání detektoru barvy a 1 pin na signál z Hall Effect senzoru.



Obr. 22: Deska Arduino Mega2560 [26]

5.2 Arduino IDE

Pro psaní programu a komunikaci s deskou Arduino Mega2560 byl využita aplikace Arduino IDE. Arduino IDE podporuje psaní kódu v jazyku C a C++ a obsahuje spoustu knihoven [31]. V práci byly použity knihovny Servo.h pro ovládání servomotorů a EEPROM.h pro ukládání informací do paměti použité desky.



Obr. 23: Prostředí Arduino IDE [30]

5.3 Program

Ke zprovoznění funkce mincovníku bylo potřeba vytvořit program, který zvládne plnit vše, co po něm nadřazený systém požaduje – přijmout minci, ověřit její validitu, zařadit ji do příslušného zásobníku a případně vyplatit výhru.

V první části programu jsou nadefinovány všechny proměnné a pro čitelnost programu pojmenovány všechny piny shodně se jmény zařízení (příp. konkrétními signály daných zařízení), jež jsou na tyto piny připojena.

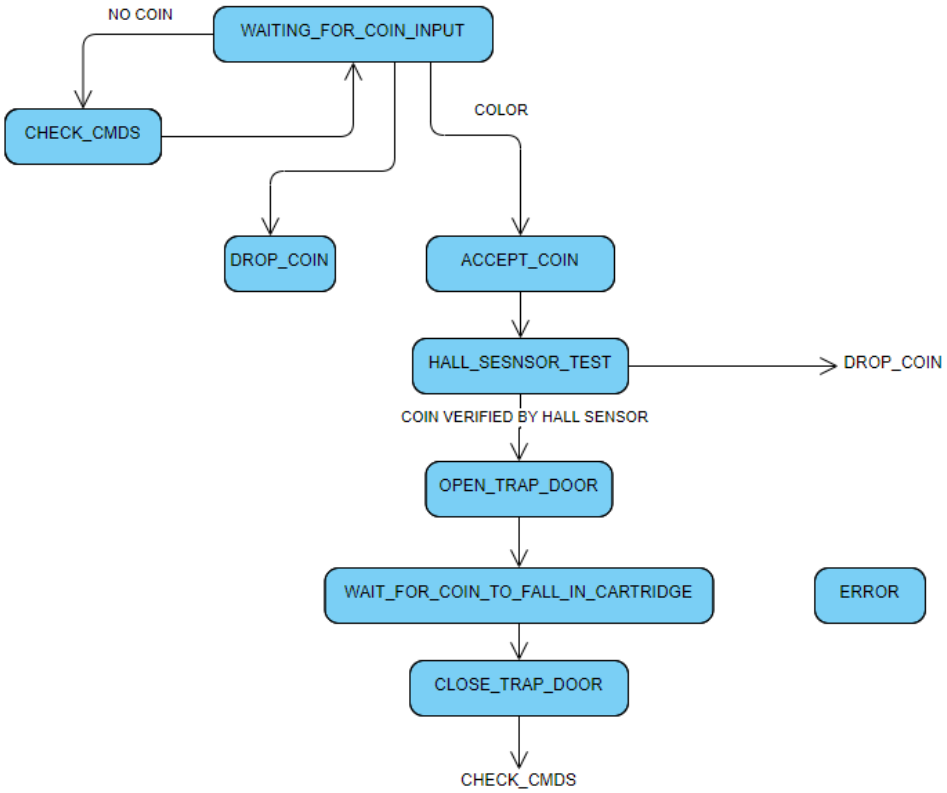
Dále následuje definice funkce setup(), jež je provedena pouze jednou po startu (či brown-outu). Slouží k inicializaci, tedy načtení remanentních hodnot z paměti a konfiguraci pinů na OUTPUT a INPUT. Také zde dochází k zahájení komunikace po sériové lince.

```
positionOfDropperRedGreen = EEPROM.read(saveLastRedGreenDropperPosition);
positionOfDropperBlue = EEPROM.read(saveLastBlueDropperPosition);
redCoinsInCartridge = EEPROM.read(saveredCoinsInCartridgeCount);
greenCoinsInCartridge = EEPROM.read(savegreenCoinsInCartridgeCount);
blueCoinsInCartridge = EEPROM.read(saveblueCoinsInCartridgeCount);
```

Obr. 24: Kód zajišťující načtení hodnot remanentních proměnných

Hlavní část programu se nachází ve funkci loop. Tato část běží nepřetržitě ve smyčce. Základ této funkce představuje stavový automat. Stavový automat vykonává pro každý stav relativně jednoduchý kód, zajišťující vždy pouze velmi omezenou funkcionalitu. Komplexnější funkcionality je dosaženo přepínáním mezi těmito stavy.

Konstrukce založená na stavovém automatu umožňuje vytváření velmi komplexních funkcionalit při lineárně rostoucí složitosti kódu. Ta odpovídá přibývající funkcionalitě programu, vazby na dřívější kód jsou omezeny na přechody mezi stavy (je snížené riziko interference nesouvisejících součástí kódu).



Obr. 25: Stavový automat mincovníku

5.3.1 CHECK_CMDS

Počátečním stavem automatu je CHECK_CMDS. Tento stav slouží ke kontrole, jestli po sériové lince nepřišel nějaký příkaz, který by měl mincovník provést. Pokud byl příkaz přijat, je přednostně zpracován. Po tuto dobu je pozastaveno přijímaní mincí. Následně

se stavový automat přepne do dalšího stavu, což bude vždy WAITING_FOR_COIN_INPUT.

```
case CHECK_CMDS:
      delay(500); // have enough time to receive whole message
      if(Serial.available()) {
        orderCmd = Serial.readStringUntil('\n');
        executeCommand();
      }
      state = WAITING_FOR_COIN_INPUT;
      break;
```

5.3.2 WAITING_FOR_COIN_INPUT

Stav WAITING_FOR_COIN_NIPUT slouží k rozpoznání barvy mince a rozhodnutí, jaká mince se v mincovníku nachází. K určení barvy slouží uživatelsky definovaná funkce getCoinColor. Proměnná col může nabývat 5 hodnot – v mincovníku se nenachází žádná mince (NO_COL), je přítomna mince neznámé barvy (UNKNOWN_COIN_COL) a barvy RED, GREEN a BLUE. Právě pomocí této proměnné určíme, jaký stav bude následovat.

```
case WAITING_FOR_COIN_INPUT: // Rozpoznání mince
      col = getCoinColor();
       switch(col)
      {
        case UNKNOWN_COIN_COL:
          state = DROP_UKNOWN_COIN;
          break;

        case NO_COL:
          state = CHECK_CMDS;
          break;

        case GREEN:
          case RED:
          case BLUE:
            state = ACCEPT_COIN;
            break;
      }
      break;
```

5.3.3 DROP_UNKNOWN_COIN

Pokud byla v předchozím stavu vyhodnocena mince neznámé barvy, přepne se automat do stavu DROP_UNKNOWN_COIN a otočením turniketu propustí minci dále. Po uplynutí 5 vteřin se vrátí do počátečního stavu CHECK_CMDS.

```
case DROP_UKNOWN_COIN:
        passCoinThroughGate();
        delay(5000);
        state =CHECK_CMDS;
      break;
```

5.3.4 ACCEPT_COIN

Do stavu ACCEPT_COIN se automat přepne v případě, že při kontrole barvy měla mince jednu ze tří akceptovaných hodnot. Otočením turniketu dojde k propuštění mince k dalšímu zpracování – stav bude přepnut na HALL_SENSOR_TEST.

```
case ACCEPT_COIN:
      passCoinThroughGate();
      state =HALL_SENSOR_TEST;
      break;
```

5.3.5 HALL_SENSOR_TEST

Druhá kontrola validity mince proběhne ve stavu HALL_SENSOR_TEST. Nejprve se do proměnné zapíše čas od spuštění programu, aby se systém neuvedl do deadlocku. V případě, že mince nevykazuje požadovanou magnetickou signaturu či dojde k chybě měření. Pokud je tedy rozdíl mezi časem spuštění programu a proměnnou startTime větší než 5 vteřin, mincovník se přepne do původního stavu CHECK_CMDS. V takovém případě lze předpokládat, že mince propadla – byla vrácena zákazníkovi. V případě, že Hall Effect senzor zaznamená přítomnost feromagnetického jádra mince, stav se přepne na OPEN_TRAP_DOOR.

Zde je prostor pro případné budoucí zlepšení. Používané žetony totiž obsahují jádro ve tvaru mezikruží, lze tedy očekávat sekvenci dvou po sobě jdoucích pulzů. Toto by bylo možno naprogramovat přidáním dalšího stavu HALL_SENSOR_TEST_2, jež by obdobně očekávala druhý pulz. Tato varianta nebyla realizována z časových důvodů, především kvůli potížím se správným nastavením citlivosti Hallova sensoru.

```
case HALL_SENSOR_TEST:
        if (startTime ==0){
          startTime = millis();
        }
          if( (millis() - startTime ) > 5000)
          {
            state = CHECK_CMDS;
            startTime = 0;
          }

          if( coinVerifiedByHallSensor())
          {
            state = OPEN_TRAP_DOOR;
            startTime = 0;
          }
      break;
```

5.3.6 OPEN_TRAP_DOOR

Pokud se ověří pravost mince, následuje zařazení mince do příslušného zásobníku. Provede se to pomocí funkce openRelevantTrapDoor, kde vstupem je proměnná col. Podle barvy mince tedy zareaguje příslušné servo, otevřou se padací dveře a mince propadne do zásobníku.

```
case OPEN_TRAP_DOOR:
        openRelevantTrapDoor(col);
        state = WAIT_FOR_COIN_TO_FALL_IN_CARTRIDGE;
        break;
```

5.3.7 WAIT_FOR_COIN_TO_FALL_IN_CARTRIDGE

Po otevření padacích dveří je třeba počkat, než mince propadne do zásobníku. Abychom zjistili, jestli můžeme padací dveře znovu zavřít, využijeme senzorů sledování čáry, využitých ke snímání přítomnosti mince, které jsou umístěné u vstupů do zásobníku. Ve stavu čekáme na signál z příslušného senzoru, který získáme z funkce waitForCoinToFall.

```
case WAIT_FOR_COIN_TO_FALL_IN_CARTRIDGE:
        waitForCoinToFall(col);
        state = CLOSE_TRAP_DOOR;
        break;
```

5.3.8 CLOSE_TRAP_DOOR

V posledním stavu proběhne zavření padacích dveří. Zavření se provede podobně jako u otevírání pomocí funkce closeRelevantTrapDoor. Vstupem je opět proměnná col. Po zavření padacích dveří se mincovník přepne zpět do stavu CHECK_CMDS a další mince může být vložena.

```
case CLOSE_TRAP_DOOR:
        closeRelevantTrapDoor(col);
        state =CHECK_CMDS;
        break;
```

5.4 Komunikace s nadřazeným systémem

Ke komunikaci s nadřazeným systémem byla zvolena komunikace pomocí sériové linky. Při navrhování se muselo zohlednit několik faktorů a rozhodnout, co za informace se bude posílat. Bylo nutné zvolit formát přijaté informace a navrhnout, jak bude mincovník informaci zpracovávat a následně podle typu informace provést danou operaci.

Vzhledem k tomu, že mincovník bude primárně sloužit pro proherní stroj, je důležité, aby uměl mincovník komunikovat o stavu zásobníků, jelikož by při naplnění kapacity nemohl přijímat žádné další mince. Další nutností je přijímání informace o vyplacení určitého počtu mincí, v případě jackpotu příkaz o vyplacení všech mincí, které se nachází v zásobníku. Pokud se vyjme zásobník, je potřeba vynulovat počet mincí v zásobníku, což je realizováno dalším typem příkazu.

5.4.1 Formát přijatého příkazu

Abychom mohli komunikovat s nadřazeným systémem, je potřeba vytvořit jasný formát, ze kterého půjde snadno zjistit informace, co po nás systém žádá. První znak přijaté informace bude sloužit k rozpoznání typu žádané operace. U operací jako vypsání stavu zásobníků nebo vyplacení jackpotu stačí tato část zprávy, ovšem u vyplacení konkrétního počtu mincí a připsání mincí do zásobníků je nutné určit, kolik mincí příslušných barev je žádáno. To je řešeno šestimístným číslem, kdy vždy 2 místa určují jednu barvu v pořadí červená, zelená, modrá. Konec zprávy je určen tak, že se za ni připíše výraz \n. Celkový formát je tedy názevOperaceRRGGBB\n. Různé operace v programu vyvoláváme pomocí funkce switch, která kontroluje první znak zprávy.

```
void getCommandOrder(){
if (Serial.available()) {
    orderCmd = Serial.readStringUntil('\n');
    orderCmd.trim();
    switch(orderCmd[0]){
```

5.4.2 Vyplácení mincí

První typ příkazu je vyplacení určité částky – payout. Příkaz tedy začíná znakem 'p'. Následují informace o počtu mincí, které mají být vyplaceny. Ze zprávy to pro jednotlivé barvy mincovník získá pomocí funkcí getRedCoinsFromCmd, getGreenCoinsFromCmd a getBlueCoinsFromCmd. Aby mohlo dojít k vyplacení výhry, je potřeba mít dostatek mincí v zásobnících. Pokud je tato podmínka splněna, provede se postupně vyplacení všech druhů mincí. V opačném případě se vypíše zpráva o nedostatku mincí. Ukázkový příkaz, který vyplatí 2 červené a 1 modrou minci, je p020001.

```
case 'p':
        redToPayout = getRedCoinsFromCmd(orderCmd);
        greenToPayout = getGreenCoinsFromCmd(orderCmd);
        blueToPayout = getBlueCoinsFromCmd(orderCmd);

        if (enoughCoinsToPayout(redToPayout, greenToPayout,
            blueToPayout))
        {
          payoutRed(positionOfDropperRedGreen, redToPayout);
          payoutGreen(positionOfDropperRedGreen, greenToPayout);
          payoutBlue(positionOfDropperBlue, blueToPayout);
        }
        else {
          Serial.println("Nedostatek mincí");
        }
        break;
```

5.4.3 Informace o stavu zásobníků

Aby se nadřazený systém mohl dotazovat na stav zásobníku, byl vytvořen příkaz, který se vyvolává znakem 'n'. Po odeslání informace se vypíše počet zbývajících mincí.

```
case 'n':
        serialPrintAmountOfCoins(redCoinsInCartridge,
        greenCoinsInCartridge, blueCoinsInCartridge);
        break;
```

5.4.4 Nastavení počtu mincí

Pokud dojde ke změně počtu mincí v zásobníku vlivem údržby, je třeba tuto informaci nastavit i v programu mincovníku. Bylo proto nutné, aby jedním z příkazů bylo vynulování hodnot u proměnných, které nám určují počet zbývajících mincí v zásobnících.

```
case 'z':
        redCoinsInCartridge = 0;
        greenCoinsInCartridge = 0;
        blueCoinsInCartridge = 0;
        break;
```

5.4.5 Manuální přidání počtu mincí

Vzhledem ke konstrukci vyhazovacího zařízení zůstane ve vyhazovacím mechanismu vždy jedna mince (pokud tedy nebyl zásobník příslušné barvy zcela vyprázdněn). Aby se po vyjmutí nemuselo pomocí funkce vyplácení zbavit zbylých mincí, je možné po vynulování hodnot přidat mince jednotlivě. Příkaz vypadá stejně jako u vyplácení mincí, ovšem je nutné začít znakem 'a'. Program pak načte hodnoty z přijaté zprávy a připočítá je k aktuálnímu počtu mincí.

```
case 'a':
        redToPayout = getRedCoinsFromCmd(orderCmd);
        greenToPayout = getGreenCoinsFromCmd(orderCmd);
        blueToPayout = getBlueCoinsFromCmd(orderCmd);
        redCoinsInCartridge += redToPayout;
        greenCoinsInCartridge += greenToPayout;
        blueCoinsInCartridge += blueToPayout;
        break;
```

5.4.6 Vyplacení jackpotu

Mincovník byl navržen pro proherní stroj, proto je nutné, aby uměl vyplatit jackpot, neboli vyplatit všechny mince, které se nachází v zásobníku. Příkaz se vyvolává pomocí znaku 'j'.

```
case 'j':
        Serial.println("JACKPOT");
        payoutRed(positionOfDropperRedGreen, redCoinsInCartridge);
        payoutGreen(positionOfDropperRedGreen, greenCoinsInCartridge);
        payoutBlue(positionOfDropperBlue, blueCoinsInCartridge);
        break;
```

6 ZÁVĚR

V teoretické části byly shrnuty informace o typech mincovníků a nejpoužívanějších metodách pro rozeznávání validitu mince. Dále zde byly zmíněny některé komerčně dostupné mincovníky na českém trhu a jejich využití.

V praktické části byl vytvořen mincovník, který pomocí detektoru barvy a Hall Effect senzoru rozlišuje 3 typy proprietárních žetonů. Mechanický návrh mincovníku byl realizován s využitím programu Autodesk Inventor. Nestandardní součásti byly následně vytištěny pomocí 3D tiskárny. Mincovník je ovládán pomocí desky Arduino Mega 2560, program je napsaný ve vývojovém prostředí Arduino IDE. Dále byly popsány všechny použité součásti a funkce, díky kterým mincovník plní svoji funkci.

Sestrojený mincovník byl navržen především pro nasazení v "proherním stroji", pro který musí vykonávat jak příjem mincí, tak vyplácení výhry. Mincovník i nadřazený systém slouží k edukačním účelům.

Pro autora byla práce přínosem, jelikož mu dala cenné zkušenosti jak v oblasti navrhování modelů, tak v programování.

7 SEZNAM POUŽITÉ LITERATURY

[1] Underthepier: Coin Mechanisms [online]. [cit. 2021-5-21]. Dostupné z: https://www.underthepier.com/01_howtocoinmech.htm

[2] Gumball Machine Warehouse: Original Bubble Gum Machine [online]. [cit. 2021-5-21]. Dostupné z: https://www.gumball-machine.com/products/original-bubble-gum-candy-machine

[3] Velenet.cz: Mincovník do mechanického automatu [online]. [cit. 2021-5-21]. Dostupné z: https://www.velenet.cz/mincovnik-do-mechanickeho-automatu-p505

[4] Kulečníky - šipky: Mincovník MR 89 [online]. [cit. 2021-5-21]. Dostupné z: https://www.kulecniky-sipky.cz/mincovniky/mincovnik-mr-89-na-10-kc/?gclid=CjwKCAjwtJ2FBhAuEiwAIKu19sxPVZYwUu7AoljoQcFDTKWlL1m4MxiqpYMXFBM_VpkhEMvIQTvK-BoCKp4QAvD_BwE

[5] Kränzle ČR: Mincovník MPR 108 [online]. [cit. 2021-5-21]. Dostupné z: https://www.kranzle.cz/mincovnik-mpr-108-e1476.htm#ac-description

[6] Zábavka: Elektronický mincovník EU2 [online]. [cit. 2021-5-21]. Dostupné z: https://zabavka.cz/eshop/k1679-mincovniky/100-elektronicky-mincovnik-eu2.html

[7] HWKITCHEN: EF90D micro:servo 360° [online]. [cit. 2021-5-21]. Dostupné z: https://www.hwkitchen.cz/ef90d-micro-servo-360-kontinualni-pro-microbit-s-kolem/

[8] NERO AMUSEMENT: Elektronický mincovník Comestero RM5-G [online]. [cit. 2021-5-21]. Dostupné z: https://www.neroamusement.cz/mincovniky/elektronicky-mincovnik-comestero-rm5-g/

[9] https://www.neroamusement.cz/mincovniky/elektronicky-mincovnik-comestero-rm5-g/

[10] Quicktest.co.uk: U. V. torches [online]. [cit. 2021-5-21]. Dostupné z: https://www.quicktest.co.uk/uv.htm

[11] ČERNOŠEK, M. Konstrukční návrh prodejního automatu na energetické tyčinky. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2011. 37 s. Vedoucí diplomové práce Ing. Michal Vaverka, Ph.D.

[12] Flippers: How coin validators work on pinball machines [online]. [cit. 2021-5-21]. Dostupné z: https://www.flippers.be/basics/101_coinmechs.html

[13] https://www.coinmech.com/product_profile.cfm?id=524

[14] Coin World: How do vending machines know if a coin is real? [online]. [cit. 2021-5-21]. Dostupné z: https://www.coinworld.com/news/precious-metals/how-vending-machines-know-coins-real-or-fake-coin-world-buzz.html

[15] Kiosk Marketplace: Frequently asked questions about bill acceptors [online]. [cit. 2021-5-21]. Dostupné z: https://www.kioskmarketplace.com/news/mei-3-frequently-asked-questions-about-bill-acceptors/

[16] Electronic Components: Krokové motory [online]. [cit. 2021-5-21]. Dostupné z: https://www.tme.eu/cz/news/library-articles/page/41861/krokovy-motor-druhy-a-priklady-aplikaci-krokovych-motoru/

[17] https://dratek.cz/arduino/832-eses-krokovy-motor-driver-pro-jednodeskove-pocitace.html

[18] Návody Drátek: Servo motor [online]. [cit. 2021-5-21]. Dostupné z: https://navody.dratek.cz/arduino-projekty/servo-motor.html

[19] Návody Drátek: Arduino detektor barvy TCS230 [online]. [cit. 2021-5-21]. Dostupné z: https://navody.dratek.cz/navody-k-produktum/arduino-detektor-barvy.html

[20] https://dratek.cz/arduino/890-detektor-barvy.html

[21] https://dratek.cz/arduino/897-eses-servo-motor-9g.html

[22] HowToMechatronics: What is Hall Effect and How Hall Effect Sensors Work [online]. [cit. 2021-5-21]. Dostupné z: https://howtomechatronics.com/how-it-works/electrical-engineering/hall-effect-hall-effect-sensors-work/

[23] https://etqan.sa/product/linear-magnetic-hall-effect-sensor-module-for-arduino/?lang=en

[24] https://www.laskarduino.cz/arduino-infracerveny-senzor-sledovani-cary-s-lm393/

[25] Laskarduino.cz: Arduino Mega2560 rev3, original [online]. [cit. 2021-5-21]. Dostupné z: https://www.laskarduino.cz/arduino-mega2560-rev3--original/

[26] https://www.laskarduino.cz/arduino-mega2560-rev3--original/

[27] Alza.cz: 3D tisk: jak funguje, kde stáhnout předlohy a jak začít? [online]. [cit. 2021-5-21]. Dostupné z: https://www.alza.cz/3d-tisk

[28] https://www.alza.cz/prusa-i3-mk3-d5271426.htm

[29] https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html

[30] https://support.microsoft.com/cs-cz/topic/uploading-board-code-and-arduino-ide-a9723765-1314-49e0-a69b-bb5c3e1f628d

[31] Arduino: Software [online]. [cit. 2021-5-21]. Dostupné z: https://www.arduino.cc/en/software

[32] HWKITCHEN: EF90D micro:servo 360° [online]. [cit. 2021-5-21]. Dostupné z: https://www.hwkitchen.cz/ef90d-micro-servo-360-kontinualni-pro-microbit-s-kolem/

8 SEZNAM PŘÍLOH

Příloha 1: Zdrojový kód programu
Příloha 2: Sestava modelu (pouze elektronicky)

PŘÍLOHY

Zdrojový kód programu:

```
#include <Servo.h>
#include <EEPROM.h>
Servo redTrapDoorServo;
Servo greenTrapDoorServo;
Servo blueTrapDoorServo;
String orderCmd;

#define pinColorMeasure_S0 4
#define pinColorMeasure_S1 5
#define pinColorMeasure_S2 6
#define pinColorMeasure_S3 7
#define pinColorMeasure_Out 8
#define pinRedCoinEnteringCartridge 34
#define pinGreenCoinEnteringCartridge 35
#define pinBlueCoinEnteringCartridge 36
#define pinHallSenzor 3
#define saveLastRedGreenDropperPosition 0
#define saveLastBlueDropperPosition 1
#define saveredCoinsInCartridgeCount 2
#define savegreenCoinsInCartridgeCount 3
#define saveblueCoinsInCartridgeCount 4

enum State
{
  CHECK_CMDS,
  WAITING_FOR_COIN_INPUT,
  ACCEPT_COIN,
  HALL_SENSOR_TEST,
  OPEN_TRAP_DOOR,
  WAIT_FOR_COIN_TO_FALL_IN_CARTRIDGE,
  CLOSE_TRAP_DOOR,
  DROP_UKNOWN_COIN,
  ERROR_COIN_STUCK
};

//int redCnt, greenCnt, blueCnt;
bool positionOfDropperRedGreen, positionOfDropperBlue, hall;

const int pinGateStepperCoil_1 = 22;
const int pinGateStepperCoil_2 = 23;
const int pinGateStepperCoil_3 = 24;
const int pinGateStepperCoil_4 = 25;
const int pinDropper1StepperCoil_1 = 26;
const int pinDropper1StepperCoil_2 = 27;
const int pinDropper1StepperCoil_3 = 28;
const int pinDropper1StepperCoil_4 = 29;
const int pinDropper2StepperCoil_1 = 30;
const int pinDropper2StepperCoil_2 = 31;
const int pinDropper2StepperCoil_3 = 32;
const int pinDropper2StepperCoil_4 = 33;

const int velocity = 1;

unsigned long startTime = 0;
```

```
int redCoinEnteringCartridge, greenCoinEnteringCartridge,
    blueCoinEnteringCartridge, hallSenzor;

State state = CHECK_CMDS;

int redCoinsInCartridge = 0;
int greenCoinsInCartridge = 0;
int blueCoinsInCartridge = 0;

enum Color
{
  RED, GREEN, BLUE, NO_COL, UNKNOWN_COIN_COL
};

Color col;

void setup() {
  pinMode(pinColorMeasure_S0, OUTPUT);
  pinMode(pinColorMeasure_S1, OUTPUT);
  pinMode(pinColorMeasure_S2, OUTPUT);
  pinMode(pinColorMeasure_S3, OUTPUT);
  pinMode(pinColorMeasure_Out, INPUT);

  pinMode(pinDropper1StepperCoil_1, OUTPUT);
  pinMode(pinDropper1StepperCoil_2, OUTPUT);
  pinMode(pinDropper1StepperCoil_3, OUTPUT);
  pinMode(pinDropper1StepperCoil_4, OUTPUT);

  pinMode(pinDropper2StepperCoil_1, OUTPUT);
  pinMode(pinDropper2StepperCoil_2, OUTPUT);
  pinMode(pinDropper2StepperCoil_3, OUTPUT);
  pinMode(pinDropper2StepperCoil_4, OUTPUT);

  digitalWrite(pinColorMeasure_S0, HIGH);
  digitalWrite(pinColorMeasure_S1, LOW);



  Serial.begin(9600);

  redTrapDoorServo.attach(9);
  greenTrapDoorServo.attach(10);
  blueTrapDoorServo.attach(11);
  closeRedTrapDoor();
  closeGreenTrapDoor();
  closeBlueTrapDoor();

  positionOfDropperRedGreen =
        EEPROM.read(saveLastRedGreenDropperPosition);
  positionOfDropperBlue = EEPROM.read(saveLastBlueDropperPosition);
  redCoinsInCartridge = EEPROM.read(saveredCoinsInCartridgeCount);
  greenCoinsInCartridge = EEPROM.read(savegreenCoinsInCartridgeCount);
  blueCoinsInCartridge = EEPROM.read(saveblueCoinsInCartridgeCount);

  pinMode(pinRedCoinEnteringCartridge, INPUT);
  pinMode(pinGreenCoinEnteringCartridge, INPUT);
  pinMode(pinBlueCoinEnteringCartridge, INPUT);
  pinMode(hall, INPUT);
```

```
  pinMode(pinGateStepperCoil_1, OUTPUT);
  pinMode(pinGateStepperCoil_2, OUTPUT);
  pinMode(pinGateStepperCoil_3, OUTPUT);
  pinMode(pinGateStepperCoil_4, OUTPUT);

  Serial.setTimeout(10);

}
void loop() {
  redCoinEnteringCartridge = digitalRead
(pinRedCoinEnteringCartridge);
  greenCoinEnteringCartridge = digitalRead
(pinGreenCoinEnteringCartridge);
  blueCoinEnteringCartridge = digitalRead
(pinBlueCoinEnteringCartridge);

  switch(state){

    case CHECK_CMDS:
      delay(500); // have enough time to receive whole message + to
                  restrain the count of messages on Serial line
      if(Serial.available()) {
        orderCmd = Serial.readStringUntil('\n');
        executeCommand();
      }
      state = WAITING_FOR_COIN_INPUT;
      break;


    case WAITING_FOR_COIN_INPUT: // Rozpoznání mince
      col = getCoinColor();
       switch(col)
      {
        case UNKNOWN_COIN_COL:
          state = DROP_UKNOWN_COIN;
          break;

        case NO_COL:
          state = CHECK_CMDS;
          break;

        case GREEN:
          case RED:
          case BLUE:
            state = ACCEPT_COIN;
            break;
      }
      break;
    case ACCEPT_COIN:
      passCoinThroughGate();
      state =HALL_SENSOR_TEST;
      break;
```

```
    case HALL_SENSOR_TEST:
        if (startTime ==0){
          startTime = millis();
        }
          if( (millis() - startTime ) > 5000)
          {
            state = WAITING_FOR_COIN_INPUT;
            startTime = 0;
          }

          if( coinVerifiedByHallSensor())
          {
            state = OPEN_TRAP_DOOR;
            startTime = 0;
          }
      break;

      case OPEN_TRAP_DOOR:
        openRelevantTrapDoor(col);
        state = WAIT_FOR_COIN_TO_FALL_IN_CARTRIDGE;
        break;

      case WAIT_FOR_COIN_TO_FALL_IN_CARTRIDGE:
        waitForCoinToFall(col);
        state = CLOSE_TRAP_DOOR;
        break;

      case CLOSE_TRAP_DOOR:
        closeRelevantTrapDoor(col);
        state =CHECK_CMDS;
        break;
      case DROP_UKNOWN_COIN:
        passCoinThroughGate();
        delay(5000);
        state =CHECK_CMDS;
        break;

      case ERROR_COIN_STUCK:
        Serial.println("ERROR");
        break;
  }
  EEPROM.write(saveLastRedGreenDropperPosition,
               positionOfDropperRedGreen);
  EEPROM.write(saveLastBlueDropperPosition, positionOfDropperBlue);
  EEPROM.write(saveredCoinsInCartridgeCount, redCoinsInCartridge);
  EEPROM.write(savegreenCoinsInCartridgeCount, greenCoinsInCartridge);
  EEPROM.write(saveblueCoinsInCartridgeCount, blueCoinsInCartridge);
}
void waitForCoinToFall(Color col)
{
  switch (col)
        {
           case RED:
             while (redCoinEnteringCartridge){
               redCoinEnteringCartridge = digitalRead
               (pinRedCoinEnteringCartridge);
             }
             redCoinsInCartridge++;
             break;
```

```
            case GREEN:
              while (greenCoinEnteringCartridge){
                greenCoinEnteringCartridge = digitalRead
(pinGreenCoinEnteringCartridge);
              }
              break;
              greenCoinsInCartridge++;
            case BLUE:
              while (blueCoinEnteringCartridge){
                blueCoinEnteringCartridge = digitalRead
(pinBlueCoinEnteringCartridge);
              }
              blueCoinsInCartridge++;
              break;
         }
}

void closeRelevantTrapDoor(Color col)
{
  switch (col)
  {
            case RED:
              closeRedTrapDoor();
              break;
            case GREEN:
              closeGreenTrapDoor();
              break;
            case BLUE:
              closeBlueTrapDoor();
              break;
  }
}

void openRelevantTrapDoor(Color col)
{
  switch (col)
        {
          case RED:
            openRedTrapDoor();
            break;

          case GREEN:
            openGreenTrapDoor();
            break;

          case BLUE:
            openBlueTrapDoor();
            break;
        }
}

inline void openGreenTrapDoor()
{
  greenTrapDoorServo.writeMicroseconds(1400);
}

inline void openBlueTrapDoor()
{
 blueTrapDoorServo.writeMicroseconds(1400);
}
```

```
inline void openRedTrapDoor()
{
  redTrapDoorServo.writeMicroseconds(1400);
}

inline void closeGreenTrapDoor()
{
  greenTrapDoorServo.writeMicroseconds(500);
}


inline void closeBlueTrapDoor()

{
  blueTrapDoorServo.writeMicroseconds(500);
}

inline void closeRedTrapDoor()
{
  redTrapDoorServo.writeMicroseconds(500);
}


int coinVerifiedByHallSensor()
{
  return digitalRead(pinHallSenzor);
}
```

```
Color getCoinColor()
{
    int redMeasured = loadRed();
    int greenMeasured = loadGreen();
    int blueMeasured = loadBlue();
    Color col;
    if (redMeasured > 100 | greenMeasured > 100 | blueMeasured > 100){
    Serial.print(" Žádná mince");
    Serial.println();
    col = NO_COL;
    }
    else{
       if (redMeasured < 50) {
        Serial.print(" | Detekce cervene. ");
        Serial.println();
        col = RED;
        }
       else if (greenMeasured < 60) {
        Serial.print(" | Detekce zelene. ");
        Serial.println();
        col = GREEN;
        }
       else if (blueMeasured < 60) {
        Serial.print(" | Detekce modre. ");
        Serial.println();
        col = BLUE;
        }
       else {
        Serial.print(" | Detekce mince. ");
        Serial.println();
        col = UNKNOWN_COIN_COL;
        }
     }

    return col;
}

void rotacePoSmeru(int a, int b, int c, int d) {
  krok1(a,b,c,d);
  krok2(a,b,c,d);
  krok3(a,b,c,d);
  krok4(a,b,c,d);
  krok5(a,b,c,d);
  krok6(a,b,c,d);
  krok7(a,b,c,d);
  krok8(a,b,c,d);
}

void rotaceProtiSmeru(int a, int b, int c, int d) {
  krok8(a,b,c,d);
  krok7(a,b,c,d);
  krok6(a,b,c,d);
  krok5(a,b,c,d);
  krok4(a,b,c,d);
  krok3(a,b,c,d);
  krok2(a,b,c,d);
  krok1(a,b,c,d);
}
```

```
void krok1(int a, int b, int c, int d){
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  delay(velocity);
}

void krok2(int a, int b, int c, int d){
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  delay(velocity);
}

void krok3(int a, int b, int c, int d){
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  delay(velocity);
}

void krok4(int a, int b, int c, int d){
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  delay(velocity);
}

void krok5(int a, int b, int c, int d){
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  delay(velocity);
}

void krok6(int a, int b, int c, int d){
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  delay(velocity);
}

void krok7(int a, int b, int c, int d){
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  delay(velocity);
}
```

```
void krok8(int a, int b, int c, int d){
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  delay(velocity);
}

int loadRed(){
  digitalWrite(pinColorMeasure_S2, LOW);
  digitalWrite(pinColorMeasure_S3, LOW);
  delay(50);
  return pulseIn(pinColorMeasure_Out, LOW);
}

int loadGreen(){
  digitalWrite(pinColorMeasure_S2, HIGH);
  digitalWrite(pinColorMeasure_S3, HIGH);
  delay(50);
  return pulseIn(pinColorMeasure_Out, LOW);
}

int loadBlue(){
  digitalWrite(pinColorMeasure_S2, LOW);
  digitalWrite(pinColorMeasure_S3, HIGH);
  delay(50);
  return pulseIn(pinColorMeasure_Out, LOW);
}

void passCoinThroughGate()
{
  for(int i=0;i<(128);i++)
  {
    rotacePoSmeru(pinGateStepperCoil_1, pinGateStepperCoil_2,
                  pinGateStepperCoil_3, pinGateStepperCoil_4);
  }
}
```

```
void payoutGreen(bool pos, int cnt){
   while(cnt > 0){
      if(pos){
        for(int i=0;i<(256);i++){
        rotaceProtiSmeru(pinDropper1StepperCoil_1,
                         pinDropper1StepperCoil_2,
                         pinDropper1StepperCoil_3,
                         pinDropper1StepperCoil_4);
      }
        positionOfDropperRedGreen = false;
        cnt--;
        greenCoinsInCartridge--;
      }
      else if(pos==false){
        for(int i=0;i<(256);i++){
        rotacePoSmeru(pinDropper1StepperCoil_1,
                      pinDropper1StepperCoil_2,
                      pinDropper1StepperCoil_3,
                      pinDropper1StepperCoil_4);
        }
        positionOfDropperRedGreen = true;
        cnt--;
        greenCoinsInCartridge--;
      }
      pos = positionOfDropperRedGreen;
   }
}

void payoutRed(bool pos, int cnt){
  while(cnt > 0){
      if(pos){
        for(int i=0;i<(256);i++){
        rotacePoSmeru(pinDropper1StepperCoil_1,
                      pinDropper1StepperCoil_2,
                      pinDropper1StepperCoil_3,
                      pinDropper1StepperCoil_4);
        }
        positionOfDropperRedGreen = false;
        cnt--;
        redCoinsInCartridge--;
      }
      else if(pos==false){
        for(int i=0;i<(256);i++){
        rotaceProtiSmeru(pinDropper1StepperCoil_1,
                         pinDropper1StepperCoil_2,
                         pinDropper1StepperCoil_3,
                         pinDropper1StepperCoil_4);
        }
        positionOfDropperRedGreen = true;
        cnt--;
        redCoinsInCartridge--;
      }
      pos = positionOfDropperRedGreen;
  }
}
```

```
void payoutBlue(bool pos, int cnt){
  while(cnt>0){
      if(pos){
        for(int i=0;i<(256);i++){
        rotacePoSmeru(pinDropper2StepperCoil_1,
                      pinDropper2StepperCoil_2,
                      pinDropper2StepperCoil_3,
                      pinDropper2StepperCoil_4);
      }
      positionOfDropperBlue = false;
      cnt=cnt-1;
      blueCoinsInCartridge--;
      }
      else if(pos == false){
        for(int i=0;i<(256);i++){
        rotaceProtiSmeru(pinDropper2StepperCoil_1,
                         pinDropper2StepperCoil_2,
                         pinDropper2StepperCoil_3,
                         pinDropper2StepperCoil_4);
        }
      positionOfDropperBlue = true;
      cnt=cnt-1;
      blueCoinsInCartridge--;
      }
      pos = positionOfDropperBlue;
  }
}

void executeCommand(){
        int redToPayout;
        int greenToPayout;
        int blueToPayout;
    switch(orderCmd[0]){
      case 'p':
        redToPayout = getRedCoinsFromCmd(orderCmd);
        greenToPayout = getGreenCoinsFromCmd(orderCmd);
        blueToPayout = getBlueCoinsFromCmd(orderCmd);

        if (enoughCoinsToPayout(redToPayout, greenToPayout,
            blueToPayout))
        {
          payoutRed(positionOfDropperRedGreen, redToPayout);
          payoutGreen(positionOfDropperRedGreen, greenToPayout);
          payoutBlue(positionOfDropperBlue, blueToPayout);
        }
        else {
          Serial.println("Nedostatek mincí");
        }
        break;
       case 'n':
        serialPrintAmountOfCoins(redCoinsInCartridge,
                                 greenCoinsInCartridge,
                                 blueCoinsInCartridge);
        break;
       case 'z':
        redCoinsInCartridge = 0;
        greenCoinsInCartridge = 0;
        blueCoinsInCartridge = 0;
        break;
```

```
case 'a':
        redToPayout = getRedCoinsFromCmd(orderCmd);
        greenToPayout = getGreenCoinsFromCmd(orderCmd);
        blueToPayout = getBlueCoinsFromCmd(orderCmd);
        redCoinsInCartridge += redToPayout;
        greenCoinsInCartridge += greenToPayout;
        blueCoinsInCartridge += blueToPayout;
        break;
       case 'j':
        Serial.println("JACKPOT");
        payoutRed(positionOfDropperRedGreen, redCoinsInCartridge);
        payoutGreen(positionOfDropperRedGreen, greenCoinsInCartridge);
        payoutBlue(positionOfDropperBlue, blueCoinsInCartridge);
        break;
    }
}

bool enoughCoinsToPayout(int redToPayout, int greenToPayout,
int blueToPayout)
{
  return (redCoinsInCartridge >= redToPayout &
          blueCoinsInCartridge >= greenToPayout &
          greenCoinsInCartridge >= blueToPayout);
}

int getRedCoinsFromCmd(String command){
        return 10*( command[1] - '0' ) + ( command[2] - '0');
}
int getGreenCoinsFromCmd(String command){
        return 10*( command[3] - '0' ) + ( command[4] - '0');

}
int getBlueCoinsFromCmd(String command){
        return 10*( command[5] - '0' ) + ( command[6] - '0');
}

void serialPrintAmountOfCoins(int r, int g, int b){
 Serial.println();
 Serial.print("Počet červených mincí: ");
 Serial.print(r);
 Serial.print(" | Počet zelených mincí: ");
 Serial.print(g);
 Serial.print(" | Počet modrých mincí: ");
 Serial.print(b);
 Serial.println();
}
```