Středoškolská technika 2022

Setkání a prezentace prací středoškolských studentů na ČVUT

Naváděcí a ovládací systém pro astronomické dalekohledy

Vojtěch Hora

Střední škola EDUCHEM, a. s.

Okružní 128, Meziboří

Poděkování

Chtěl bych poděkovat panu Ing. Burešovi za vedení mé závěrečné práce. Mgr. Martině Mrázové za korekci a kontrolu Českého jazyka. Firmě Inelsev s.r.o., oddělení MaR, dílně R121 pod vedením pana Petra Gažůra za umožnění tisku na 3D tiskárně v rámci mé odborné práce.

Prohlášení

Prohlašuji, že jsem maturitní práci zpracoval/a samostatně a že jsem uvedl/a všechny použité informační zdroje a literaturu v seznamu použitých zdrojů.

Anotace

V této práci je popsán návrh a následné zhotovení ovládacího a naváděcího systému pro astronomické dalekohledy přes mobilní telefon a Bluetooth technologii.

Obsah

1 Úvod

Cíl mé práce byl navrhnout a poté zhotovit mnou navržený řídící a automaticky navádějící systém pro astronomické dalekohledy. Jedním z hlavních důvodů, proč jsem se rozhodl vytvořit takový systém je příliš velká cena těchto systému dostupných na trhu. Hlavně proto bych chtěl dosáhnout co nejmenší ceny mého systému. Nižší ceny přesto nechci dosáhnout na úkor funkčnosti mého systému.

Systém budu navrhovat pro paralaktické astronomické montáže a měl by být schopen se pohybovat v obou osách, rektascenze i deklinace. Měl by redukovat otáčení Země, tím by teleskop uměl sledovat jeden objekt dlouhodobě. Vše by mělo být ovládáno přes mobilní telefon, včetně navolených objektů, na které by se měl teleskop sám nastavit a automaticky začít tyto objekty sledovat.

Následně popíšu jednotlivě každou část mého systému, proč jsem ji použil a k čemu v mém systému slouží. Na konci pak popíšu, jak tyto části fungují společně dohromady.

2 Arduino

Arduino je široce rozšířená a velice populární open-source vývojová deska založená na mikropočítači. Tyto vývojové desky se používají k výuce na školách, domácím projektům nebo odborným pracím. Arduino a jemu podobné vývojové desky jsou oblíbené hlavně kvůli jejich minimální pořizovací ceně a jednoduché manipulaci s nimi. Těmto deskám byl vytvořen programovací software nazývaný Arduino IDE a jednoduchý programovací jazyk Wiring, určený pro laiky, kteří nejsou programátoři.

Nejdůležitější část mého systému. Bude v něm uložený program, dle kterého se bude vše řídit a ovládat. Je to nezbytná součást systému, bez kterého by nic nefungovalo.

2.1 Historie Arduina

Jako první vznikl programovací jazyk a softwarové prostředí Wiring podobné dnešnímu Arduino IDE. Společně s tím vznikala i hardwarová vývojová deska Wiring, která se může považovat jako předchůdce prvního Arduina. Vše vzniklo jako diplomová práce Fernanda Barragána na Interaction Design Institute Ivrea (IDII) v Itálii. Fernand Barragán chtěl, aby Wiring usnadňoval umělcům a designerům práci s elektronikou a mohli se soustředit na své cíle. Touto prací byly položeny velké základy dnešnímu programovacímu jazyku Wiring pro Arduino. Například matematické, různé porovnávací, logické, ale i příkazy jako *digitalWrite(), digitalRead(), if(), while(), for(),* bez kterého se dnes programování Arduina v podstatě neobejde tam jsou použity, i identický syntax.

2.2 Vnitřnosti Arduina

Hlavní a nejdůležitější komponenta, srdce celého Arduina, je mikropočítač. Arduino už od svého počátku používá mikropočítače z řady ATmega od firmy Atmel. Mikropočítače ATmega328p, používané v Arduino Uno a Nano, jsou 8bitové mikropočítače. Každé Arduino obsahuje také krystal. V Arduinu běžně osciluje na 16 MHz. Využití takových krystalů má výhodu přesné a stabilní oscilace na chtěné frekvenci. Aby Arduino mohlo komunikovat s jinými Arduino deskami nebo s čidly a senzory, používá Arduino sériovou sběrnici I2C (IIC, Inter Integrated Circuit).

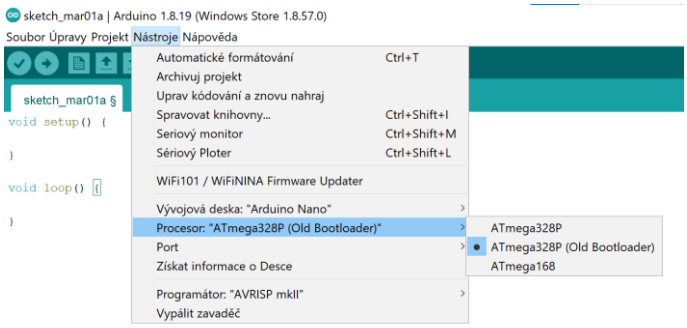
Arduino obsahuje paměť Flash, SRAM (Static Random Acess Memory) a EEPROM (Electrically Erasable Programmable Read Only Memory). Do paměti Flash se ukládá program námi naprogramovaný ve vývojovém prostředí. Obsah paměti je 32 kB. SRAM paměť se využívá pro ukládání proměnných, když se s nimi pracuje a vejde se do ní 2 kB dat. Ze SRAM se po ztrátě napájení vymaže veškerý obsah, proto je potřeba v programovém kódu nastavit výchozí data pro proměnné. Paměť EEPROM má obsah nejmenší, pouhých 1 kB, protože se používá na ukládání dlouhodobých dat.

2.3 Programování Arduina

2.3.1 Arduino IDE

Arduino má svůj vlastní programovací software, vývojové prostředí Arduino IDE. Program je zdarma dostupný na oficiálních stránkách Arduina. Programovat se může ve webovém vývojovém prostředí nebo se může stáhnout. Vývojové prostředí obsahuje už nějaké jednoduché příkladné hotové programy, třeba pro displej z tekutých krystalů, servo motory nebo krokové motory. Příkladné programy jsou pod horní záložkou Soubor, následně v podsložce Příklady, kde už lze vybírat požadovaný program.

Pro úspěšné nahrání programu je potřeba v Arduino IDE nastavit, jaké Arduino se bude programovat a na jakém USB portu je Arduino připojeno k osobnímu počítači. Po rozšíření Arduina se také začali prodávat levnější kopie neboli klony. U těchto klonů, zejména u Arduino Nano se musí navíc nastavit procesor se starým bootloaderem. Veškeré tyto parametry se nastavují v horní záložce Nástroje.



Obr. 1

2.3.2 Programovací jazyky

Arduino má svůj vlastní programovací jazyk, který je navržený tak, aby se na Arduinu mohl učit a programovat naprostý laik. Vlastní programovací jazyk pro Arduino, Wiring, byl vytvořen z jiného složitějšího C++. Jazykem C++ lze Arduino také programovat. Každý programovací jazyk i Wiring má svůj vlastní syntax. Wiring lze rozdělit do tří hlavních sekcí podle jejich funkcí. Tyto sekce jsou funkce, hodnoty a struktura.

Funkce nastavují Arduino a určuje s čím bude pracovat a jak. Příkaz *pinMode()* je funkcí a používá se ve *void setup()* a nastavuje se s ním s jakým pinem budu pracovat a jestli bude pin výstupní nebo vstupní, za každém takovém zapsáním musíme napsat středník. *Void setup()* pracuje pouze jedno na začátku programu hned po spuštění. Výsledné zapsání pak vypadá takto: *pinMode (4, OUTPUT);* a znamená že digitální pin číslo čtyři bude pracovat jako výstupní. To je pouze počáteční nastavení pinu, se kterým chci pracovat. V samotném programu kdy už budu chtít aby pin pracoval, potřebuju příkaz *digitalWrite()* a zapisuji ho už do *void loop(),* který začne pracovat po *void setup()* a funguje do restování nebo odpojení desky od zdroje. Příkladné zapsání vypadá takto: *digitalWrite (4, HIGH);* Výsledné znění takového zapsání je digitální pin číslo čtyři se nahodí do logické jedničky. To znamená, že na takovém pinu bude 5 V, protože Arduino pracuje s pěti voltovou logikou. Funkce, ale nejsou jenom tyhle dva příkazy, stejně jako s digitálními piny mohu pracovat i s analogovými. Obsahují také matematické, časové nebo bitové a bajtové operace i nastavení sériové komunikace.

Hodnoty jsou příkazy, kterými přidávám něčemu, třeba pinům, určitou hodnotu. Pokud napíšu příkaz *pinMode (4, OUTPUT);* nebo *digitalWrite(4, HIGH);* tak celé tyto příkazy nejsou funkce. Funkcí je pouze *pinMode()* nebo *digitalWrite()*, to co napíšu do závorky jsou už hodnoty. Z toho vyplívá, že příkladným příkazem hodnoty je třeba *OUTPUT* nebo *HIGH*. Příkazem hodnoty může být i *int* neboli integer, který umožňuje pracovat s číselnou řadou nezávisle na sobě. Můžu nastavit i výchozí číslo v číselné řadě na začátek programu, od kterého bude integer pracovat.

Viz obr. 2 může vypadat jednoduchý program jenom za použití funkcí a hodnot. Program funguje jako jednoduchý blikač. Na začátku programu se nahodí digitální pin číslo dva do logické jedničky. Vzápětí musí mikropočítač počkat půl sekundy, než shodí stejný pin do logické nuly a musí znovu počkat půl sekundy. Takhle program funguje do nekonečna.

```
void setup() {

pinMode(2, OUTPUT);

}

void loop() {

digitalWrite(2, HIGH);
delay(500);
digitalWrite(2, LOW);
delay(500);

}
```

Obr. 2

Poslední struktura určuje syntax jazyku jako středník, závorky, jednořádkové komentování příkazem //, více blokové komentování /**/, jednoduché matematické operace jako násobení, dělení, sčítání odčítání, porovnávání, rovnání. Jsou to ale i takové příkazy, určující celou strukturu kódu, jako *loop()* a *setup()* nebo příkazové operace *If(), else(), while, for.*

Na obr. 3 je napsaný jednoduchý program za použití funkcí, hodnot i struktury. Program funguje podobně jako předchozí je pouze podmíněný. Je to tedy znovu blikač, aby blikač fungoval, musí být přivedena na digitální pin číslo tři logická jedna. Pin číslo tři musel být na začátku programu nadefinován jako vstupní a v programu se s ním musí pracovat jako s čtením hodnoty, ne jako zapisováním.

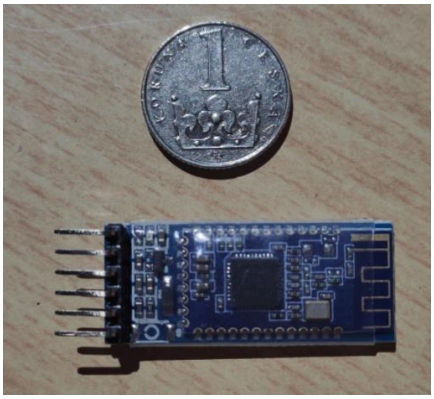
```
void setup() {

pinMode(2, OUTPUT);
pinMode(3, INPUT);

}

void loop() {

if (digitalRead(3) == HIGH) {

digitalWrite(2, HIGH);
delay(500);
digitalWrite(2, LOW);
delay(500);}


}
```

Obr. 3

3 Bluetooth a komunikace s mobilním telefonem

Tato technologie byla vyvinuta firmou Erisson a funguje jako sériová bezdrátová linka, navržena byla jako bezdrátová náhrada za sériovou linku RS-232. Umožňuje komunikaci bez potřeby mobilního signálu, internetu, Wi-Fi sítě nebo jiného prostředníka. Komunikace probíhá přímo mezi zařízením s Bluetooth a jiným zařízením s Bluetooth. Mohou tedy komunikovat i v odlehlejších částech Země, bez signálu nebo se nemusí zajišťovat jiný prostředník, přes kterou by komunikace probíhala. Takové technologie se třeba využívá u mobilních telefonů a bezdrátových sluchátkách, myších, klávesnicích, hodinkách a spousty další elektroniky. Může se využít i k přenosu dat mezi mobilními telefony či počítači. U nejnovější verze Bluetooth 5.0 se dají přenášet data rychlostí 2 Mbit/s. Bluetooth pracuje v pásmu 2,4 GHz a dosah je úměrně závislý na výkonu. Může to být od 1 m až do 100 m (1 mW až 100 mW).

Pro komunikaci mezi Arduinem a mobilním telefonem jsem použil bezdrátovou komunikační technologii Bluetooth, protože systém chci používat i mimo město. K Arduinu existuje celá řada různých přídavných modulů, čidel a senzorů kompatibilních s Arduinem. Jedním z nich jsou i Bluetooth moduly. Jsou dostupné různé typy Bluetooth modulů a každý je vhodný na něco jiného. Typ Bluetooth LE (Low Energy), který využívám je úspornější na spotřebovanou energii. Používá se proto v systémech, které používají baterii nebo nemají přístup k dlouhodobě stabilnímu zdroji elektřiny.



modul BLE

Obr. 4

Pro komunikaci mezi Arduinem a Bluetooth modulem je potřeba propojit sériové komunikační piny RX, TX a zároveň je nutné modul BLE napájet. Pozor se musí dát především na přepólování napájení, tyto moduly jsou na to hodně citlivé a mohou se zničit. Správné zapojení sériové linky je propojení pinu RX (received – přijato) na TX (transmited – odeslané) a opačně pokud se zapojí RX na RX a TX na TX komunikace nemůže proběhnout. Jednoduše data z pinů, které přijímají nebo jenom odesílají data nemůže fungovat komunikace.

```
int state = 0;
int flag = 0;

void setup() {

pinMode(2, OUTPUT);
Serial.begin(9600);

}

void loop() {

if(Serial.available()>0){
  state = Serial.read();
  flag=0;}

  if (flag == 0){
  if (state == '0'){
   digitalWrite(2, HIGH);
    flag = 1;
   }}}
```

Obr. 5

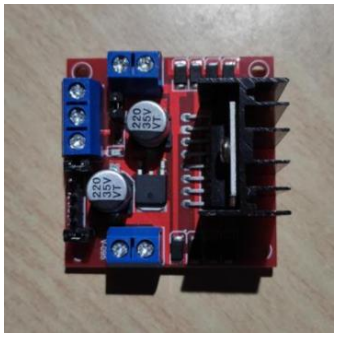
Na obr. 5 je napsaný jednoduchý program pro uvedení digitálního dva do logické jedničky přes mobilní telefon pomocí Bluetooth. V programu využívám dva integery, které jsou na začátku programu nastaveny na nulu. V módu *setup()* je nastaven digitální pin dva jako výstupní a sériovou komunikaci a její přenosovou rychlost. Přenosová rychlost se nastavuje v závorce a obě komunikující zařízení musejí být nastaveny na stejnou přenosovou rychlost, jinak nebude probíhat správně. Celý program pak funguje takhle: pokud je možná a je větší než 0, program se zeptá, jestli integer *state* se rovná nule. Zároveň integer *state* je roven *Serial.read*, pokud se integer *state* rovná nule, tak *Serial.read* přijal nulu a nahodí digitální pin dva do logické jedničky. Hned poté to byl digitální pin dva nahozen do jedničky, integer *flag* se rovná jedné.

4 Krokové motory a jejich řízení

Krokové motory umožňují přesné nastavení do požadované pozice. Takové motory se využívají třeba u robotů, kde je potřeba přesného natočení. Ze stejného důvodu jsem se rozhodl použití těchto motorů pro natáčení teleskopu.

Jsou to bezkartáčové motory s možností nastavení přesné polohy. To umožňují kroky, kterých může být v krokovém motoru až dvě stovky. Dělí se jako klasický motor na stator a rotor. Na rotoru jsou umístěny permanentní magnety, které představují jednotlivé kroky, u dvou set krokových motorů jich je tedy dvě stě. Každý krok má pak svoji polaritu a pokud má jeden krok severní polaritu. Vedlejší kroky musejí mít jižní. Na statoru pak jsou dimenzovány cívky. Podle zapojení cívek v motoru se dělí na unipolární a bipolární. Od toho se také odvíjí řízení těchto motorů.

Krokové motory není možné používat plnohodnotně bez driverů. Jediným způsobem, jak řídit krokový motor bez driverů je spínáním jednotlivě cívky v motoru pomocí několika tlačítek zapojených k motoru do H obvodu. Tímto způsobem však nejde docílit plné rychlosti, který je motor schopný a funguje spíš jako demonstrace krokování motoru. H obvod existuje i v elektronické podobě, kde místo tlačítek jsou zapojené MOSFET tranzistory a další podpůrná a ochranná elektronika. Jedním z těchto ochranných prvků jsou diody zapojené na výstup tranzistorů. Takto zapojené diody chrání tranzistory před zpětném napětím vytvořené otáčením motoru, když motor není ovládán elektronicky. Může to způsobit neznalý člověk, který zahýbe s motor a může tak vygenerovat až několik desítek mV. Takové napětí může tranzistor poškodit. Takové obvody se dají použít jako drivery. Drivery fungují na stejném způsobu, akorát celá sekvence sepnutí cívek pracuje samozřejmě ve vyšších rychlostech než jednotlivé spínání tlačítek ručně.



H-můstek L298N

Obr. 6

Obvod L298N obsahuje dva H můstky, dokáže tak řídit jeden krokový motor, nebo dva klasické 12V motory. Pro řízení krokového motoru za použití L298N do něj potřebuji pouštět jednotlivé impulzy, podle kterých se bude motor řídit, spínat cívky. Pokud budu řídit bipolární krokový motor, který má dvě cívky, motor budu ovládat přes čtyři piny, vývody z cívek. Ze dvou piny musím vždy udělat vstup a ze zbylých dvou zase výstup. Abych toho docílil budu přes Arduino otevírat a zavírat tranzistory v H můstku.

Step	C1	C2	C3	C4
1	1	0	1	0
2	0	1	1	0
3	0	1	0	1
4	1	0	0	1

řídící sekvence

Obr. 6

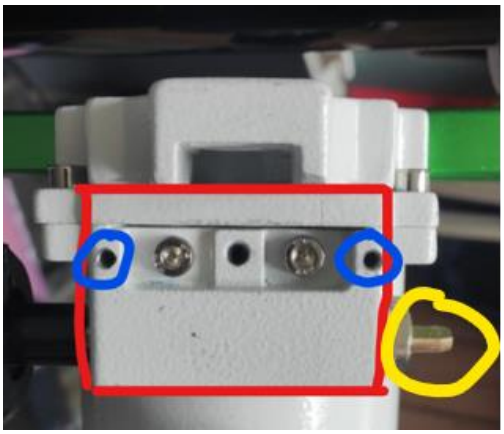
```
void call_Step_DEC(int d){
  switch (d) {
    case 0:  // 1010
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
    break;
    case 1:  // 0110
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
    break;
    case 2:  //0101
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, HIGH);
    break;
    case 3:  //1001
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, HIGH);
    break;
  }}
```

výsledný řídící program

Obr. 7

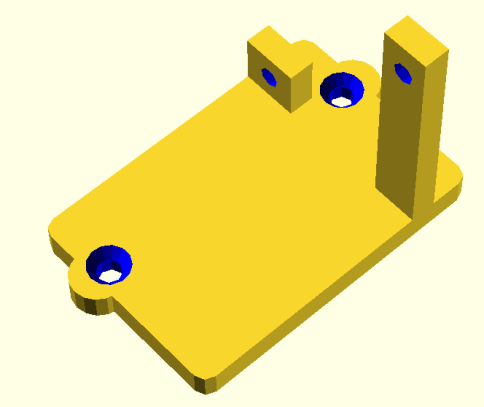
5 Mechanické části

Samostatné motory na teleskop umístit nejdou, proto je potřeba vymyslet přídavný systém, který bude schopen přišroubovat na teleskop a zároveň bude pevně držet motor. Každá osa není stejně uzpůsobena pro uchycení motorů, proto musel být přídavný uchycovací systém navrhnut jinak, pro každou osu zvlášť. Zároveň u každého místa na, osách zamýšlené pro umístění motoru, je z boku vytažena hřídel typu D pro náhon otáčecí hlavy osy, na obr. 8 žlutě znázorněna na deklinační ose. Na obr. 8 je červeně vyznačena plocha pro umístění motoru a modře znázorněny závitové otvory pro přišroubování. Plocha pro motor na deklinační ose není nijak složitá, takže upevňovací systém byl od začátku jasný. Vytvořit destičku přišroubovatelnou k montáži, zároveň na destičce musím upevnit i motor viz obr. 9.



prostor pro motorizaci deklinační osy

Obr. 8



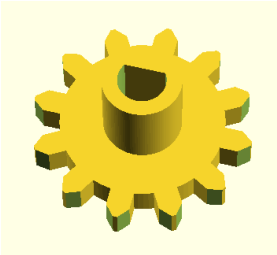
návrh upevňovací desky pro motor DEC osy

Obr. 9

výsledný produkt z 3D tiskárny

Obr. 10

Když je motor přidělaný k montáži, je potřeba ještě zařídit přenášení otáčivé síly motoru na hřídel deklinační osy, aby se teleskop mohl otáčet. To jde zajistit pomocí ozubených kol. Ozubená kola jsem musel vytvořit jiná pro motory a jiná pro poháněné osy, protože mají rozdílný průměr. Ozubené kola jsem se nejdřív snažil vytvořit. Když jsem mnou navržená ozubená kola v programu OpenScad vyzkoušel, ve skutečnosti jsem zjistil spousty chyb. Největší chyby byly malý průměr a moc velké zuby. Návrh těchto špatných kol byl časově náročný. Rozhodl jsem se proto, použít dokonalejší a dle potřeby nastavitelná ozubená kola namodelovaná v programu OpenScad pro 3D tisk.



původní mnou navržené ozubené kolo

Obr. 11

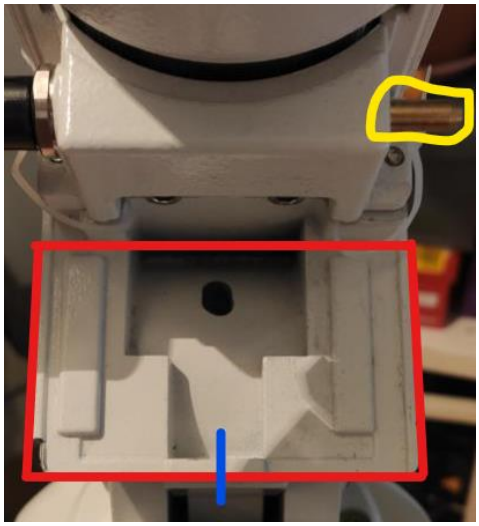
výsledné ozubené kolo

Obr. 12

vytisknutá ozubená kola

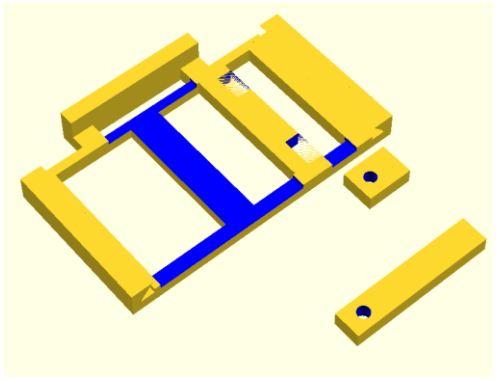
Obr. 13

Nejsložitější bylo vymyslet uchycení motoru k montáži pro rektascenžní osu. V části montáže, kde má být umístěn motor. Není žádná možnost přišroubovat uchycovací desku pro motor. Jediná možnost je využít závitového otvoru pro šroubek, který držel krytku přes tuto část pro motor. Problém byl především nerovná plocha, na které má být deska s motorem umístěna. Plochá má navíc uprostřed prohlubeň. Nejprve jsem navrhl desku, která bude připevněna zmíněným jedním šroubkem. Nedošlo mi, že motor na tolik těžký, že se deska bude třepat a nebude s motorem na místě pevně držet. Desku jsem obohatil o drážky, přes které se bude nasouvat na místo a pojistí se šroubkem. Na obr. 14 je červeně znázorněna plocha pro umístění desky pro motor, modře pojišťovací šroubek a žlutě hřídel pro pohon osy. Na obr. 15 je namodelovaná plocha k upevnění motoru k montáži. Model je rozložen kvůli složitým tvarům modelu, který by 3D tiskárna nedokázala v pořádku vytisknout.



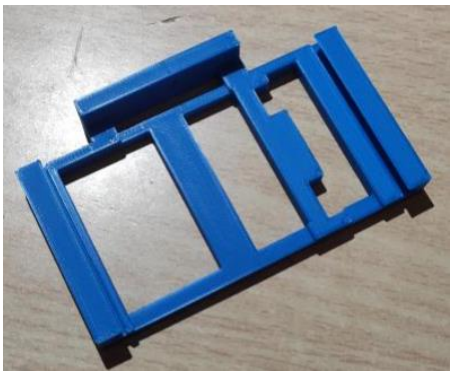
prostor pro motorizaci rektascenžní osy

Obr. 14



návrh uchycovací plochy pro DEC osu

Obr. 15

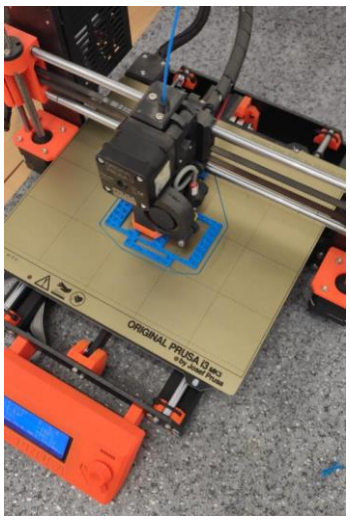


výsledný produkt 3D tiskárny

Obr. 16

5.1 3D tisk

3D tisk je technologický postup výroby námi vybraného produktu. První pokusy s 3D tiskem probíhal už v 2. pol. 20. století, ale největší rozmach zažívá v posledním desetiletí. Je to díky tomu, že se začaly vyvíjet 3D tiskárny s nízkou pořizovací cenou. Metody 3D tisku je spousty, ale nejrozšířenější je typ FDM (Fused Depostion Modeling). Tisknout se pak dá z různých materiálů, především z plastových filamentů. Nejpoužívanější typy filamentu jsou PLA (polylaktid), PET (akrylonitrilbutadienstyren) a ABS (polyethylentereftalát). Velkou revoluci v rozšíření 3D tisku zasluhuje firma Prusa Research, která začala tisknout díly pro svoje 3D tiskárny. Tímto nápadem Česká firma Prusa Research snížila cenu za svoje tiskárny a dnes je to nejrozšířenější osobní 3D tiskárna na světě.

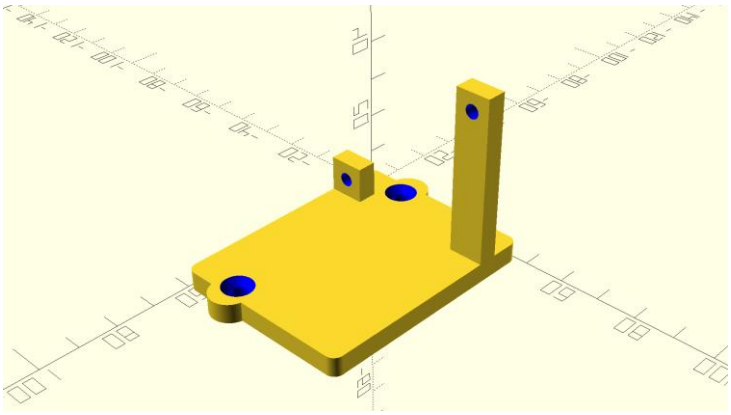


průběh tisku na 3D tiskárně PRUSA

Obr. 17

5.1.1 Modelování

Pro modelování objektu pro 3D tisk jsem použil program OpenScad. OpenScad je volně dostupný program typu CAD. Program je schopný animace, ale primárně se zaměřuje na vytváření 3D modelů technických součástek. Modelování v tomto programuje je jednoduché. Je dobré vždy před začátkem napsat příkaz *$fn=100.* Tento příkaz určuje počet fragmentů ve výchozím nastavení je nastavena na nulu. Nedoporučuje se zadávat hodnotu pro počet fragmentů větší než 100. Vysoká čísla mohou zabírat příliš paměti a používat značnou část CPU (central processing unit). Základní příkazy pro útvary jsou *cube([ ]);* pro kostku, lze udělat i kvádr. *Sphere()* pro vytvoření koule, v závorce se uvádím poloměr, pokud je potřeba uvést průměr jde použít příkaz *d* =. Pokud chci kouli s odlišným počtem fragmentů, než jsem si nastavil na začátku modelování, jde samostatně nastavit v závorce stejným příkazem. Příkazem *cylinder([ ])* namodeluju válec, kužel nebo komolý kužel. Pomocí *translate([ ])* a *rotate([ ])* dokážu s modelem hýbat a otáčet. Další užitečné příkazy jsou *union() { }* nebo *difference() { }.* Příkazem *union() { }* se všechny předměty v hranatých závorkách spojí do jednoho modelu. Příkaz *difference() { }* utvoří díru z jednoho předmětu do druhého. Díra se utvoří předmětem zapsaným jako druhý v závorkách do předmětu zapsaným jako první.



příkladný namodelovaný předmět

Obr. 18

```
$fn = 100;

  difference(){
 difference(){
 union(){
minkowski(){

 cube([55, 42, 4]);
 translate([2.5, 2.5, 0])
   cylinder(1, d=5);
}

translate([59, 13.4, 0])
 cylinder(5, d=10);


translate([1, 13.4, 0])
 cylinder(5, d=10);
}


translate([3.5, 13.3, -4])
 color("blue")
union(){

  cylinder(8, d=4.5);

  translate([0, 0, 7])
 cylinder(2.1, 2.25, 4);}}



 translate([56.5, 13.3, -4])
 color("blue")
union(){

  cylinder(8, d=4.5);

  translate([0, 0, 7])
 cylinder(2.1, 2.25, 4);}}



 difference(){
translate([13.5-5, 0, 0])
 cube([5, 9, 9+5]);

  color("blue")
  translate([7.5, 4.5, 9.5])
 rotate([0, 90, 0])
cylinder(7, d=3.5);}


 difference(){
translate([13.5-5, 38, 0])
 cube([5, 9, 42+5]);

  color("blue")
  translate([7.5, 38+4.5, 47-4.5])
 rotate([0, 90, 0])
cylinder(7, d=3.5);}
```

příkladné namodelování v příkazech (viz obr. 18)

Obr. 19

6 Výsledný systém

Celý systém řídí Arduino, ve kterém je naprogramovaný program. Na začátku programu ve *void setup() { }* jsou nadefinovány digitální piny 3-6 jako výstupní pro jeden krokový motor a piny 8-11 jako výstupní na druhý krokový motor. Sériová komunikace se nastaví na přenosovou rychlost 9600 baudů. Pojmenují se všechny integery a nastaví se na jejich počáteční nastavení číslo, z kterého budou vycházet. Vše je nastaveno pouze jednou po zapojení Arduina ke zdroji.

```
void setup() {

 pinMode(in1, OUTPUT);
 pinMode(in2, OUTPUT);
 pinMode(in3, OUTPUT);
 pinMode(in4, OUTPUT);

 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);

 Serial.begin(9600);


  #define in1 11 // driver RA input
  #define in2 10
  #define in3 9
  #define in4 8

  #define IN1 6 // driver DEC input
  #define IN2 5
  #define IN3 4
  #define IN4 3

  int osaDEC = 0;
  int osaRA = 0;
  int StepDEC;
  int StepRA;
  int ModeRA = 0;
  int ModeDEC = 0;
  int state = 0;
  int flag = 0;
  int Speed = 0;

}
```

program ve void setup() { }

Obr. 21

```
void loop() {

   if(Serial.available()>0){
    state = Serial.read();
    flag=0;
  }

  if (Speed == 100){
  if (state == '6'){
    if (flag ==0){
     Speed = 0;
      Serial.println("fast");
      flag = 1;
    }
  }}

  if (Speed == 0){
  if (state == '6'){
    if (flag ==0){
     Speed = 100;
      Serial.println("slow");
      flag = 1;
    }
  }}
```

program pro přepínání rychlosti motoru

Obr. 22

Na začátku *void loop() { }* nastavuji sériovou komunikaci a ptám se jestli je možná, pokud ano program dá integer *flag* do jedničky. Nezávisle na tom pracují příkazy pod tím, které umožní přepínat rychlosti motoru pro motorizaci rektascenze. Na přepínání stačí jedno tlačítko, protože program dokáže poznat v jaké je poloze a díky ní následující zmáčknutí tlačítka vyvolá druhou možnost. Pokud vše proběhne správně, tak se nastaví rychlost na požadovanou a do telefonu odešle program zprávu, v jakém je stavu rychlosti. Viz obr. 22.

Následující příkazy jsou pro komunikaci mezi Arduinem a mobilním telefonem. Jsou rozděleny pro DEC a RA osu zvlášť a jsou přímo propojeny s programem pro řízení motorů. Příkazy nastavují chod motorů podle hodinových ručiček, proti hodinových ručiček a zastavují motory. K příkazu zastavení motoru jsou všechny vstupy na řízení motoru nastaveny do logické nuly. Pokud by se tohle nenastavilo výstupy na motory by byly sepnuty v poslední sekvenci, ve které došlo k zastavení. Motory by tak držely pevně pozici, a to není dobré. Pokud bych chtěl vypnout motorizaci pouze elektronicky, ne fyzicky, a chtěl bych manipulovat s teleskopem, nešlo by to. Při práci krokových motorů je typický jejich zvuk. Bzučení znamená spínání cívek, takže při zastavení by vydávaly zvuk a to je zbytečné. Také je to šetrnější ke spotřebě elektřiny. Viz obr. 25. Na obr. 23 a 24 je zapsaná řídící sekvence pro spínání cívek motorů.

```
void call_Step_RA(int r){      // RA
 switch (r) {
  case 0: // 1010
   digitalWrite(in1, HIGH);
   digitalWrite(in2, LOW);
   digitalWrite(in3, HIGH);
   digitalWrite(in4, LOW);
  break;
  case 1: // 0110
   digitalWrite(in1, LOW);
   digitalWrite(in2, HIGH);
   digitalWrite(in3, HIGH);
   digitalWrite(in4, LOW);
  break;
  case 2: //0101
   digitalWrite(in1, LOW);
   digitalWrite(in2, HIGH);
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
  break;
  case 3: //1001
   digitalWrite(in1, HIGH);
   digitalWrite(in2, LOW);
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
  break;
 }
}
```

řídící program pro motor rektascenze

Obr. 23

```
void call_Step_DEC(int d){      // DEC
 switch (d) {
  case 0: // 1010
   digitalWrite(IN1, HIGH);
   digitalWrite(IN2, LOW);
   digitalWrite(IN3, HIGH);
   digitalWrite(IN4, LOW);
  break;
  case 1: // 0110
   digitalWrite(IN1, LOW);
   digitalWrite(IN2, HIGH);
   digitalWrite(IN3, HIGH);
   digitalWrite(IN4, LOW);
  break;
  case 2: //0101
   digitalWrite(IN1, LOW);
   digitalWrite(IN2, HIGH);
   digitalWrite(IN3, LOW);
   digitalWrite(IN4, HIGH);
  break;
  case 3: //1001
   digitalWrite(IN1, HIGH);
   digitalWrite(IN2, LOW);
   digitalWrite(IN3, LOW);
   digitalWrite(IN4, HIGH);
  break;
 }
}
```

řídící program pro motor deklinace

Obr. 24

```
if (state == '1'){
 //ModeRA = 1;
 if(flag ==0){
  ModeRA = 1;
  flag = 1;
 }
}

if (state == '0'){
 if(flag ==0){
  ModeRA = 0;
  Serial.println("stop_RA");
   digitalWrite(in1, LOW);
   digitalWrite(in2, LOW);
   digitalWrite(in3, LOW);
   digitalWrite(in4, LOW);
  flag = 1;
 }
}


if (state == '2'){
 //ModeRA = 2;
 if(flag ==0){
  ModeRA = 2;
  flag = 1;
 }
}


if (state == '4'){
 //ModeDEC = 1;
 if(flag ==0){
  ModeDEC = 1;
  flag = 1;
 }
}

if (state == '3'){
 if(flag ==0){
  ModeDEC = 0;
  Serial.println("stop_DEC");
   digitalWrite(IN1, LOW);
   digitalWrite(IN2, LOW);
   digitalWrite(IN3, LOW);
   digitalWrite(IN4, LOW);
  flag = 1;
 }
}

if (state == '5'){
 //ModeDEC = 2;
 if(flag ==0){
  ModeDEC = 2;
  flag = 1;
 }
}


if(ModeRA==1){
 Serial.println("CW_RA");
StepRA = StepRA+1;
if(StepRA>3){StepRA=0;}
call_Step_RA(StepRA);  //CW_RA
 delay(Speed);
 }

if(ModeDEC==1){
 Serial.println("CW_DEC");
StepDEC = StepDEC+1;
if(StepDEC>3){StepDEC=0;}
call_Step_DEC(StepDEC);  //CW_DEC
 delay(20);
 }



if(ModeRA==2){
 Serial.println("CCW_RA");
StepRA = StepRA-1;
if(StepRA<0){StepRA=3;}
call_Step_RA(StepRA);  //CCW_RA
 delay(Speed);
 }

if(ModeDEC==2){
 Serial.println("CCW_DEC");
StepDEC = StepDEC-1;
if(StepDEC<0){StepDEC=3;}
call_Step_DEC(StepDEC);  //CCW_DEC
 delay(20);
 }
```

program pro komunikaci telefon – Arduino

Obr. 25

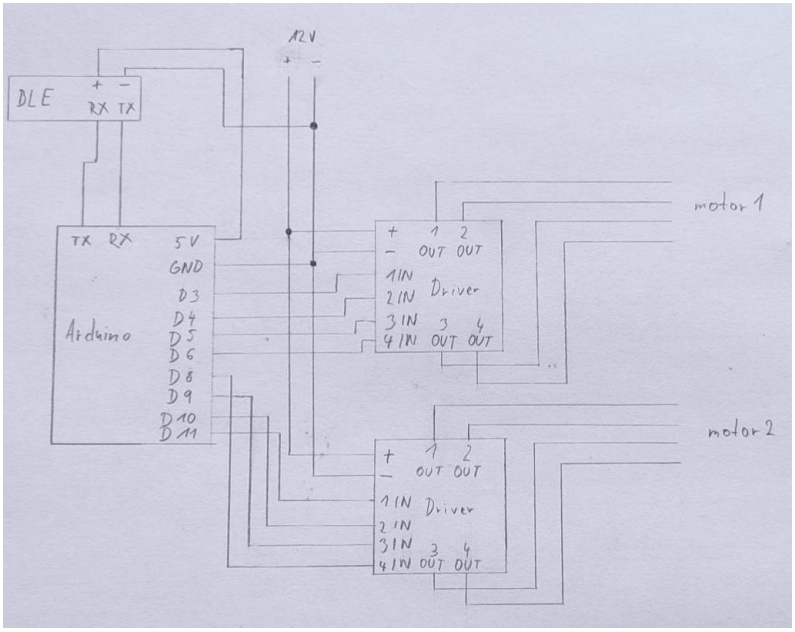
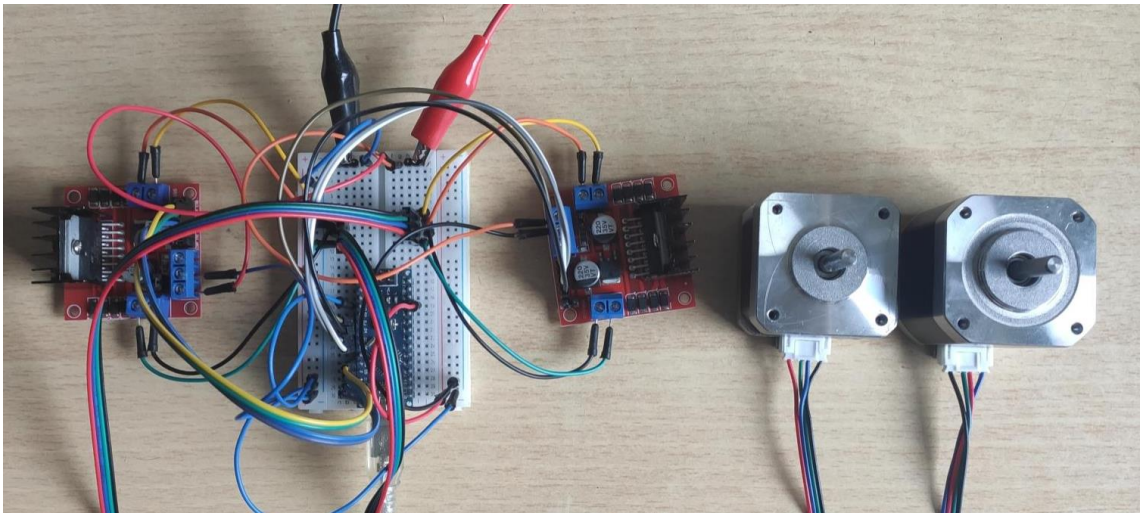


schéma celkového zapojení

Obr. 26



celý zapojený systém

Obr. 27

7 Závěr

Vývoj tohoto systému byl časově i konstrukčně náročný, i přesto že systém není dokonalý a má spoustu chyb. Myslím, že má smysl a potenciál. Vývoj mě bavil a chtěl bych tento systém v budoucnu vylepšovat a zdokonalovat. Chtěl bych systém dovést do plně funkčního a plnohodnotně ho používat. Funkční systém by měl být opensource, aby mohl na moji práci někdo navázat, vylepšit ji nebo o něco obohatit. Upevňovací součástky jsou z 3D tiskárny, lze je tedy upravit na jinou montáž. Zároveň si systém může sestavit podle plánu a programů kdokoliv a kdekoliv.

Cenově mě tento systém zatím stál něco málo přes 1400 Kč. Nejlevnější systémy stojí od 2500 Kč, pro tento typ montáže se pohybuje až od 14000 Kč a výš, kdy je systém zabudovaný v samotné montáži teleskopu.

8 Zdroje

BARRAGÁN, Hernando. Historie Arduina. *GitHub* [online]. 2016 [cit. 2022-03-29]. Dostupné z: https://arduinohistory.github.io/

BARRAGÁN, Hernando. Historie Arduina. *People.interactionivrea.org* [online]. 2004 [cit. 2022-03-29]. Dostupné z: http://people.interactionivrea.org/h.barragan/thesis/thesis_low_res.pdf

Co je Arduino a historie. *Arduino.cc* [online]. 2021 [cit. 2022-03-29]. Dostupné z: https://www.arduino.cc/en/about

Vnitřnosti Arduina. *Arduino.cc* [online]. 2018 [cit. 2022-03-29]. Dostupné z: https://www.arduino.cc/en/Tutorial/Foundations/Memory

Krkové motory. *Tme.eu* [online]. 2022 [cit. 2022-03-29]. Dostupné z: https://www.tme.eu/cz/news/library-articles/page/41861/krokovy-motor-druhy-a-priklady-aplikaci-krokovych-motoru/

Program OpenScad. *Openscad.org* [online]. 2022 [cit. 2022-03-29]. Dostupné z: https://openscad.org/about.html