Kalkulator z mehaničnimi stikali

Nik Bažato 63210019

Za projekt pri predmetu vhodno-izhodne naprave sem se odločil narediti kalkulator, ki ima mehanična stikala.

List uporabljenih komponent/orodji:

- Arduino Nano
- MX Switch (mehanično stikalo) ×16
- Pokrovček za tipko (Keycap) ×16
- Mnogo žic, in konektorjev
- 3D printer
- Onshape (za modeliranje)

Table of Contents

Software

Začnemo s programiranjem mikrokontrolerja; to lahko naredimo z Arduino IDE ali z razširitvijo za VS Code, imenovano »PlatformIO«. Ko imamo svoj IDE, lahko naložimo program na mikrokontroler. Moja koda je objavljena na GitHubu na tej povezavi: Povezava.

Spodaj bom podal razlago nekaterih delov kode ki niso intuitivni:

```
void setup()
{
  ld.setBright(1);
  ld.setDigitLimit(8);

  Serial.begin(9600);
  for (int i = button0; i <= buttonEnter; i++)
  {
    pinMode(i, INPUT_PULLUP);
  }
  pinMode(A6, INPUT);
  pinMode(A7, INPUT);
  pinMode(8, INPUT_PULLUP);
  pinMode(11, INPUT_PULLUP);

  pinMode(displayCLK, OUTPUT);
  pinMode(displayDIN, OUTPUT);
  pinMode(displayCS, OUTPUT);
}
```

Za vse digitalne vhode uporabljam »PULLUP« vpore, to pomeni, da je po defaultu digitalna vrednost HIGH, in ko pritisnemo stikalo se spremeni vrednost na LOW.

Branje vhodov

```
if (digitalRead(i) == LOW)
    {
      String out = pressKey(i - 5);
      if (out == "DBZ")
        DisplayError();
      else
        DisplayNum();
      if (out != "")
        Serial.println(out);
    }
    else
    {
      keyPressed[i - 5] = false;
    }
```

V `loop` funkciji gremo skozi vse digitalne vhode in pogledamo, če je digitalna vrednost LOW, če je pomeni, da je to stikalo bilo pritisnjeno.

```
if (analogRead(7) >= 512)
  {
    String out = pressKey(8);
    if (out == "DBZ")
      DisplayError();
    else
      DisplayNum();
    if (out != "")
      Serial.println(out);
  }
  else
```

```
  {
    keyPressed[8] = false;
  }
```

Za branje analognih vhodov uporabljam funkcijo `analogRead`, ki vrne 0, ko je 0v na vhodu in 1023 ko je 5v na vhodu. Seveda vrne tudi vmesne vrednosti npr 512 ko je 2.5v na vhodu.

Kot vhod v funkcijo pressKey podam index tega stikala. V primeru pritiska na številko se samo to število doda v string, ki se ga bo kasneje poračunalo.

```
 case 4 ... 6:
    calcString += String(key);
    numToDisplay += String(key);
    return "Pressed: " + String(key);
```

Če je bil pritisnjen operator, se zgodi nekaj zelo podobnega, edina razlika je, da se numtoDisplay pobriše.

```
  case 3:
    calcString += "+";
    numToDisplay = "";
    return "Pressed: +";
```

V primeru, da je uporabnik pritisnil na »=«, se sproži funkcija ProcessCalculation in rezultat se prikaže na ekran.

```
case 14:
    calc = ProcessCalculation(calcString);
    numToDisplay = calc.toFloat();
    calcString = calc;
    return (calc);
```

Procesiranje vhodov

Funkcija `ProcessCalculation` je najkompleksnejša funkcija, tako bom razložil samo najpomembnejše dele.

V funkciji je while zanka, ki teče dokler obstajajo operatorji v nizu.

```
while (CountOperators(string) > 0)
```

Funkcija `HighestOperatorPriority` vrne 2, če obstaja »*« ali »/« v nizu, če ne pa vrne 1.

Nato poiščemo prvi operator in naslednji operator za njim. S pomočjo indexov teh operatorjev lahko ugotovimo kje se nahajajo številke in jih z `substring` pridobimo in pretvorimo v float z `toFloat()`. Nato izračunamo obe številki in jih pretvorimo nazaj v `String`.To številko nato dodamo `String`-u, kateremu so bili ti dve izračunani števili odrezani.

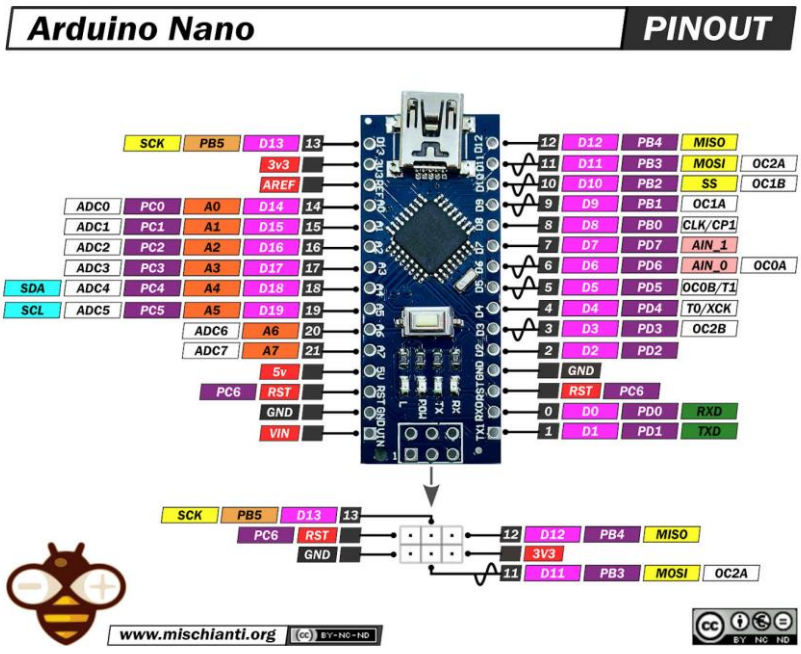
To se dogaja dokler ni število operatorjev enako 0.

```
if (HighestOperatorPriority(string) == 1)
    {
      int nextOp = FindNextOperator(string);
      String tempString = string.substring(nextOp
+ 1);
      float num1 = string.substring(0,
nextOp).toFloat();
      int nextNextOp =
FindNextOperator(tempString);
      float num2;
      if (nextNextOp == -1)
      {
        num2 = tempString.substring(0).toFloat();
      }
      else
      {
        num2 = tempString.substring(0,
nextNextOp).toFloat();
      };
      if (nextNextOp == -1)
```

```
      {
        string = (string[nextOp] == '+') ?
String(num1 + num2) : String(num1 - num2);
      }
      else
      {
        string = (string[nextOp] == '+') ?
String(num1 + num2) +
tempString.substring(nextNextOp) : String(num1 -
num2) + tempString.substring(nextNextOp);
      }
    }
```

Hardware

Pinout

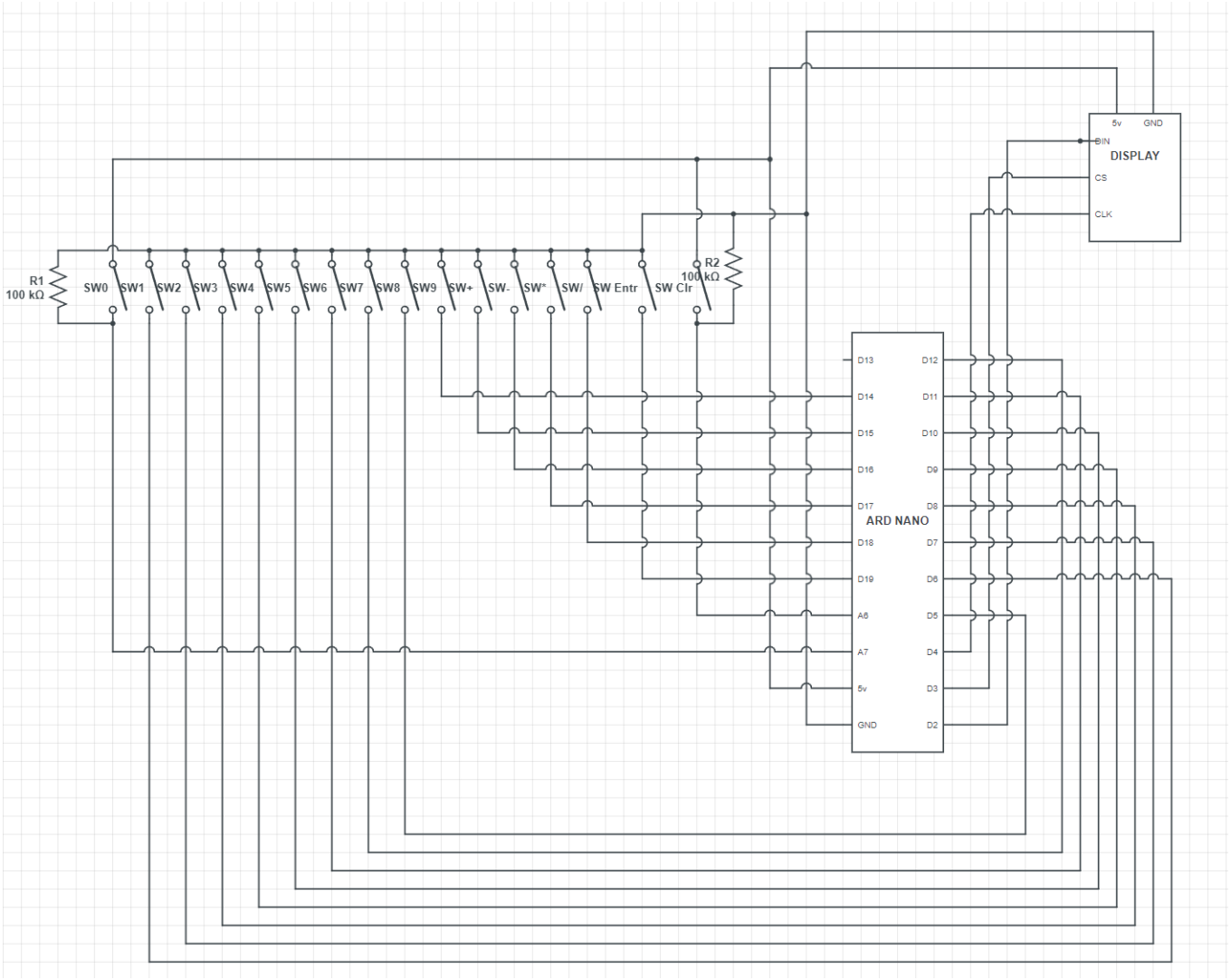


Nato lahko nadaljujemo na hardware. Vemo, da za povezavo vseh stikal z mikrokontrolerjem potrebujemo 16 priključkov (pinov), poleg tega potrebujemo še 3 za zaslon. To pomeni, da potrebujemo 19 digitalnih priključkov. Na prvi pogled se zdi, da s tem ni nobenega problema, saj na »Pinout« diagramu vidimo 20 digitalnih priključkov. Vendar sta D0 in D1 priklopljena na USB vodilo, zato ju ni pametno uporabljati za stikala. Prav tako je D13 povezan na integrirano LED diodo, kar pomeni, da bi morali za uporabo D13 odstraniti LED diodo ali njen upor.

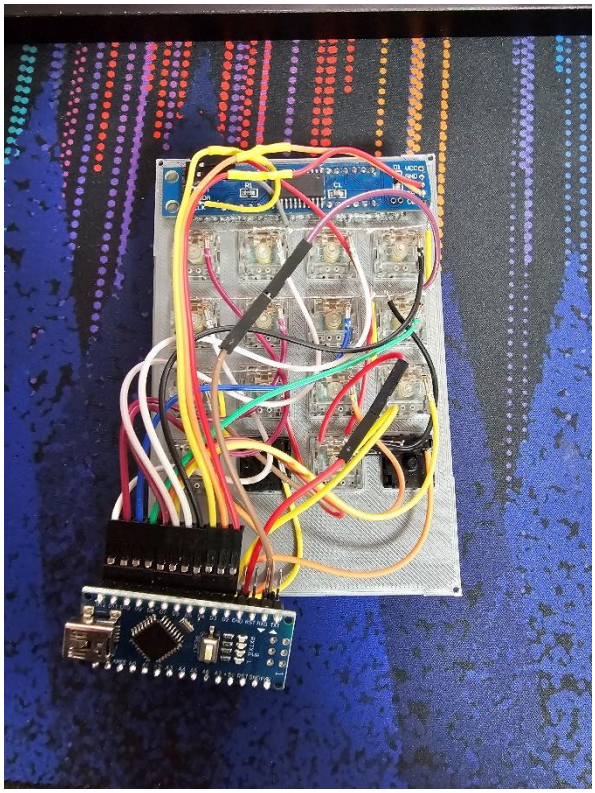
Vsi digitalni vhodi so sprogramirani na »INPUT_PULLUP«, to pomeni da je priključek po defaultu povezan preko upora na 5v.

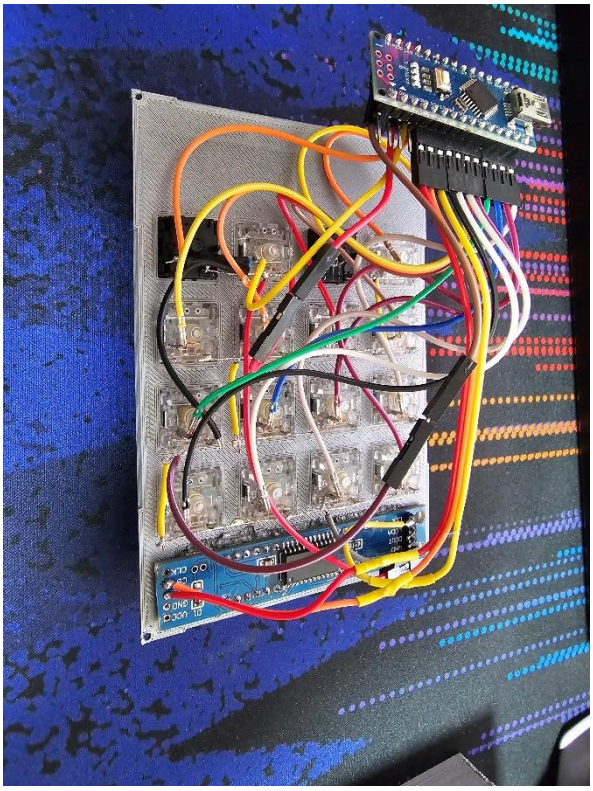
Diagram

Ker nisem hotel odstraniti D13 LED diode, sem se odločil uporabiti D2-D4 za zaslon, D5-D12 in D14-D19 za priklop stikal. To pa ni dovolj, saj imamo 16 stikal in samo 14 digitalnih priključkov. Ta problem sem rešil tako, da sem za 2 stikali uporabil analogne vhode. In sicer tako, da sem en priključek na stikalu priklopil na 5V, drugi pa skozi 100kΩ upor na GND in v A6 oz. A7 pri drugem stikalu. Za več podrobnosti in boljše razumevanje sem spodaj podal diagram.



Na diagramu so SW0 in SW Clr povezani na analogne priključje, ostala stikala pa so vsa povezana na digitalne priključje.



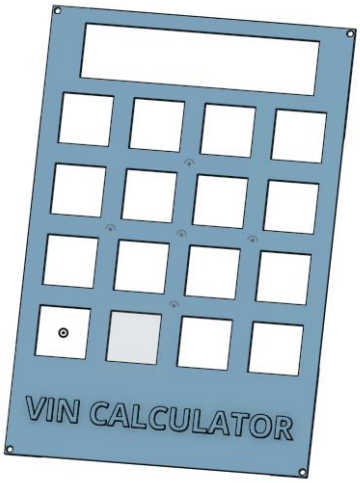


Tako zgleda diagram vresničen.

S tem je projekt funkcionalen, vendar ne še končan. Za končen produkt sem hotel še ohišje, za to sem s pomočjo spletne strani »onshape.com« naredil 2 modela, ki skupaj sestavljajo popolno ohišje. In nato to ohišje naredil z 3D printerjem.

Modeliranje

Pokrov

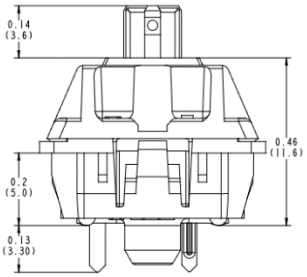


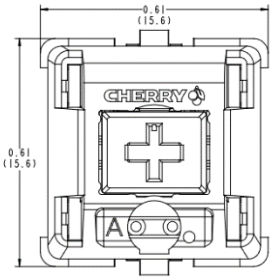
Izbral sem Onshape, ker je popolnoma brezplačno orodje za 3d modeliranje z vsemi funkcijami, ki sem jih potreboval in več.

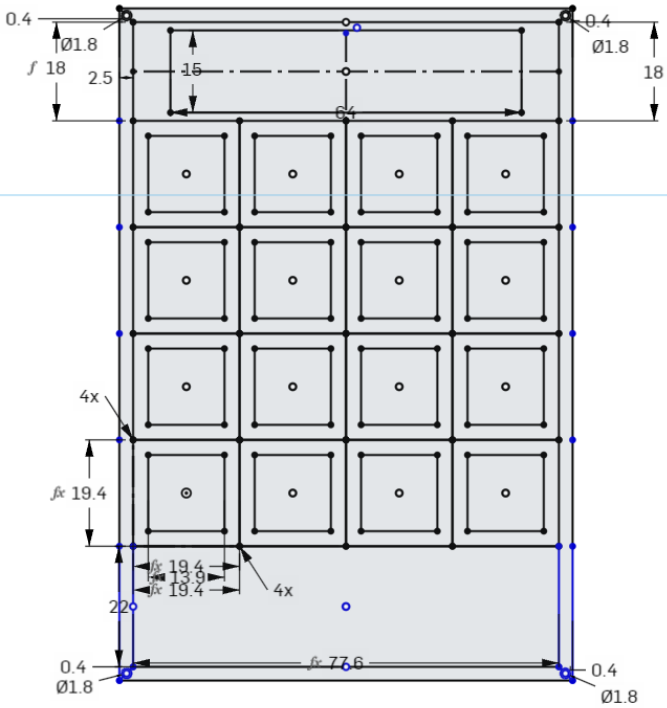
Začel sem z modeliranjem vrhnjega dela ohišja, ki ga bom od zdaj naprej imenoval »pokrov«.

Ta del je glavni del, ki bo držal vsa stikala in tudi zaslon, tako je moral biti čimbolj natančen saj so stikala pritrjena na pokrov in so brez druge podpore.

Ker je MX stikalo standard, sem lahko natančne velikosti enostavno pridobil iz Cherry spletne strani »TUKAJ«.





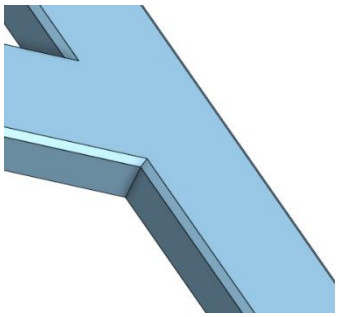


Tukaj so meritve mojega modela.

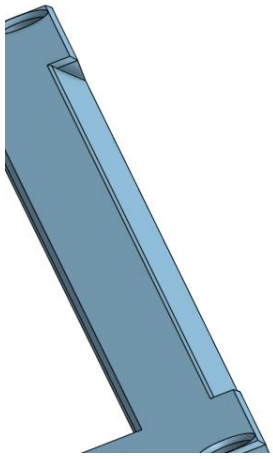
Iz tega diagrama se lahko razbere vse meritve, premere in zamike.

Model ima tudi druge značilnosti, ki niso vidne v diagramu z meritvami, kot so: Posneti robovi, 4 vdolbine za lažjo pritrditev na spodnji del ohišja.

Posneti rob je rešitev za »slonovo nogo«, to je problem ki nastane med 3D printanjem. Prvi sloj printa je širši kot ostali, tako z posnetim robom kompenziramo.

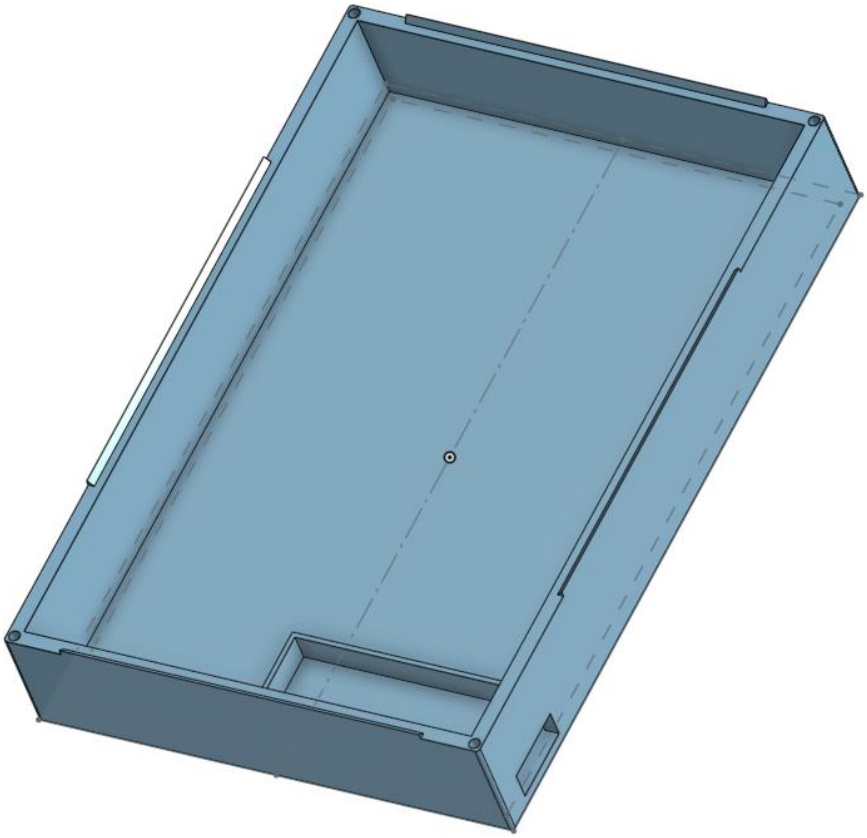


Posneti rob



Vdolbina

Spodnje ohišje

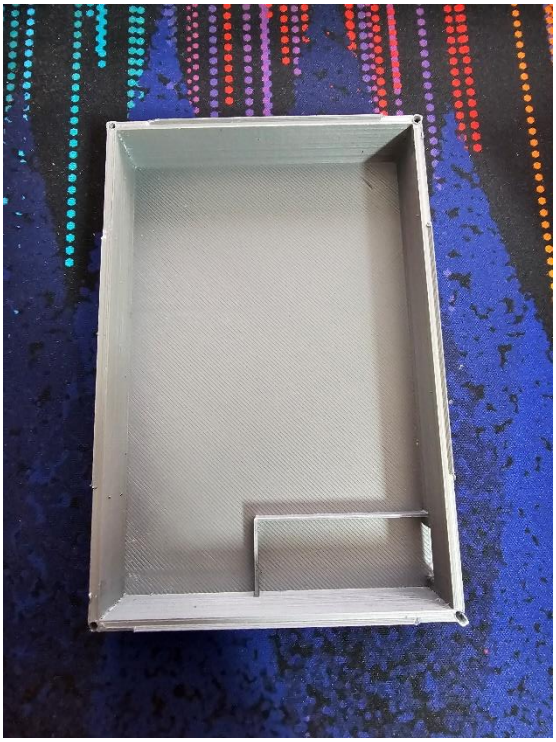


Drugi del ohišja je bil lažje modeliran, saj je samo kvader iste višine in dolžine kot pokrov.

Ima 4 zobe, ki se sovpadajo z vdolbinami na pokrovu. S pomočjo tega, se celotno ohišje lažje sestavi skupaj, saj se pokrov avtomatsko usmeri v pravilen položaj.

Na spodnji desni strani je predel za Arduino nano. S pomočjo tesnega prileganja (friction fit) ostane mikrokontroler na pravem mestu. Model je tako narejen, da je dostop do USB konektorja na mikrokontrolerju neoviran.

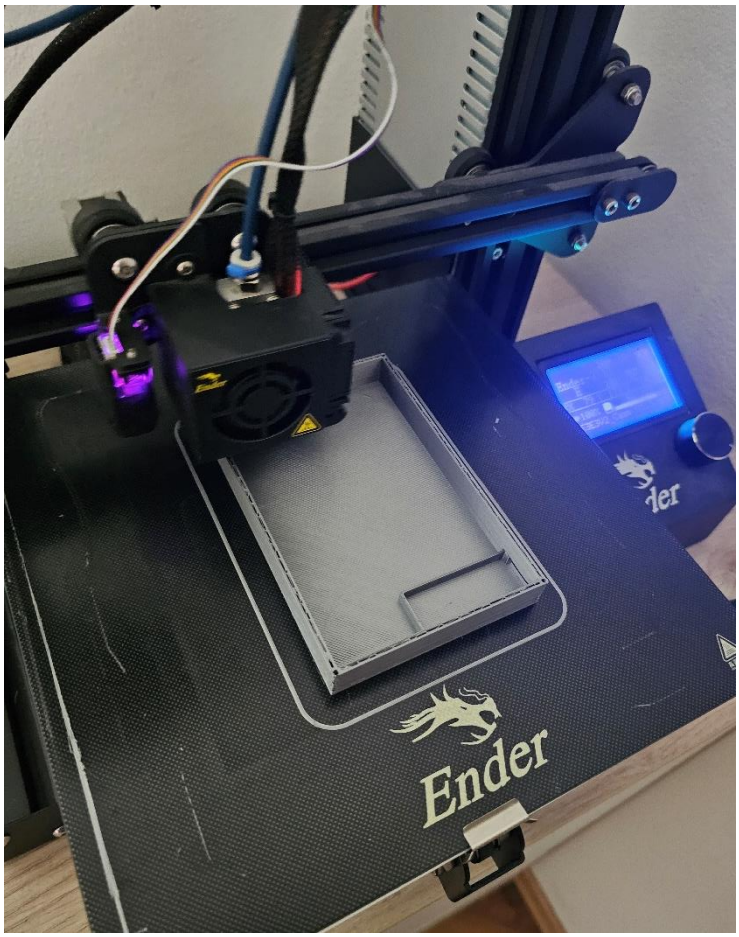
Na 3D printerju imam 0.2mm šobo (nozzle), tako sem naredil debelino zidov 2.4mm in ne 2.5mm saj bi bil končen sprintan objekt manj točen.



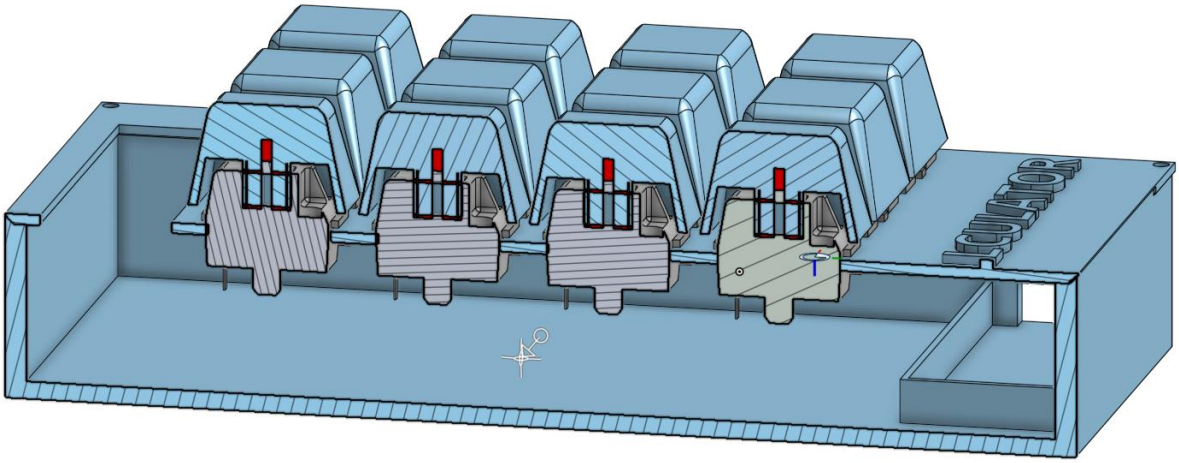
Tako zgleda končen objekt.

Ponekod je potreboval malo obdelave ampak nič velikega. Glavno je da se vrhni zobje skladajo z udolbinami na pokrovu; ker 3D printanje ni perfektno, je potrebno očistiti robove in ostale ostre elemente.

Končen izdelek sem tudi v Onshape sestavil, tako sem lahko videl približno kako bo zgledalo pred printanjem, saj printanje je vzelo več ur. Tukaj je slika spodnjega dela v procesu printanja.

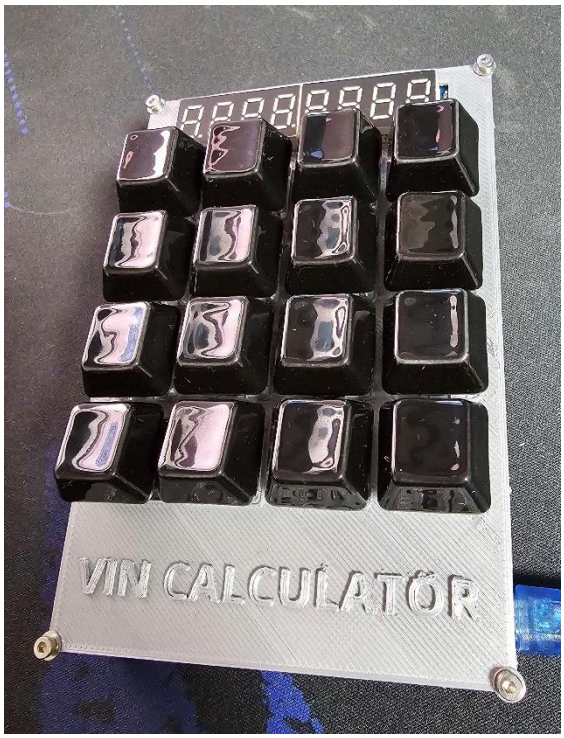


V orodju Onshape je obstaja zelo uporabna funkcija »Section View«. S pomočjo tega sem si vizualiziral koliko prostora bom imel v ohišju, za vse povezave.



Končen izdelek

Tukaj je primerjava med simuliranim izdelkom in realnim.



Pri končnem izdelku je videti tipične značaje 3D printanja kot so diagonalne črte.

Ker je končen produkt bil prelahek, saj je samo plastika, sem uporabil keramične tipke(keycaps). To doda kalkulatorju dovolj mase, da je občutek teže očiten.

In kot je opaziti tipke nimajo legende. To je zato ker mi je izgled lepši brez legende.