Projekt K.O.R.N Rev 0.1 Beta

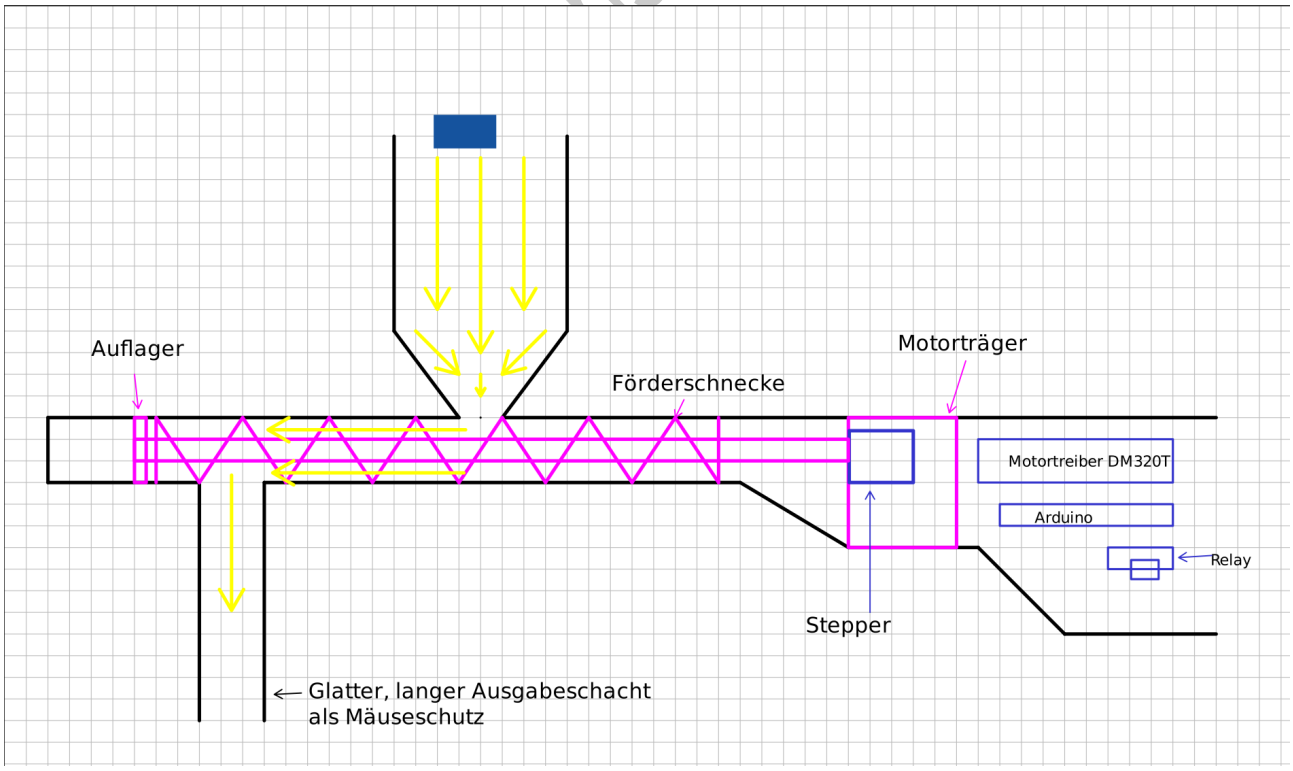
Katastrophal **o**rganisierter **r**under **N**ahrungsmittelspender

Beschreibung

K.O.R.N. wurde geplant als robuster, Open-Source/Hardware, wasserdichter, mit einfachen Mitteln konstruierter und Mäusesicherer Fütterungsautomat für Geflügel.

Aufgrund der enttäuschenden Erfahrung mit gekauften Fütterungsautomaten welche trotz der teils hohen Preise entweder nach drei Wochen defekt waren, oder ganze Mäusefamilien durchfütterten, musste eine Eigenkonstruktion her. Die Entscheidungsgrundlage für das gewählte System mit einer Förderschnecke in einem Rohr, basiert auf einer Recherche im Dubbel Ausgabe von 2001.

Es handelt es sich um einen Stetigförderer (Schnecke), der Schüttgut (Futter) aus einem Silo (KG-Rohr) in einen Auswurfschacht befördert. Die Abgabezeit und Dauer der Abgabe (Menge der Schritte des Steppers) ist einstellbar.



Das Gehäuse besteht aus überall erhältlichen, robusten und günstigen HT-, bzw KG-Rohren. Diese sollen garantieren, dass das Futter vor Mäusen, Insekten und Feuchtigkeit geschützt ist.

Die elektronischen Bauteile bestehen aus Bauteilen, welche leicht verfügbar und standardisiert sind. Ausserdem sind alle Datenblätter und Wirings leicht mit einer Online Suche auffindbar. Alle Bauteile können in diversen Onlineshops leicht und relativ kostengünstig erworben werden.

Ein Relay wurde dazwischengeschaltet um den Stepper in der inaktiven Zeit stromlos zu schalten.

Die Bauteile der Förderschnecke wurden mittels Freecad (LGPL2+, CC-BY-3.0) entworfen.

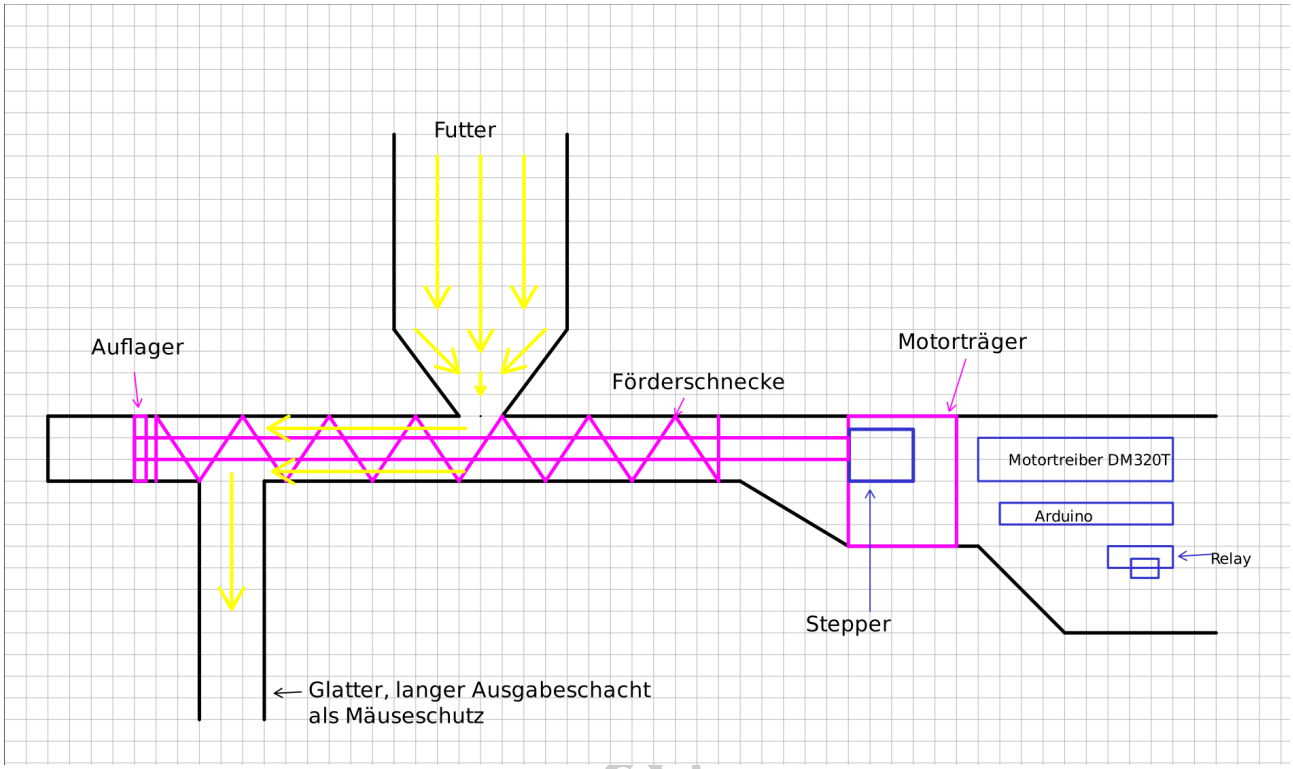
Die Förderschnecke wurde steckbar entworfen. Somit ist es möglich, die Schnecke auch auf kleineren 3D-Druckern zu drucken.

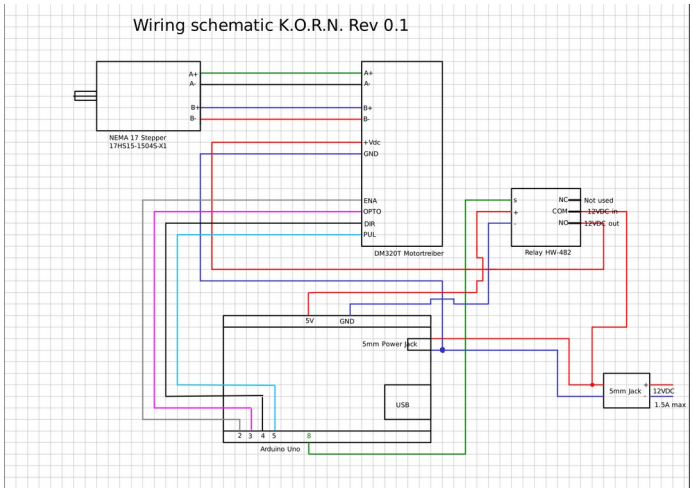
Der Code wurde mithilfe von codeium.com entworfen und manuell angepasst. Aufgrund der fehlenden Schnittstelle zu einem Zeitgeber (RTC oder Wlan) wurde der Code entworfen, dass das Arduino 10sec nach dem Power up das Relay anzieht. Dabei wird der Motortreiber mit Spannung versorgt. Kurz danach startet das Arduino Programm den Stepper mit folgenden Werten und anschließend alle 24h:

```
// Define variables for stepper parameters

int acceleration = 100; // Beschleunigung des Steppers in Steps pro Sekunde

int targetRPM = 100;    // Zielgeschwindigkeit in RPM

int stepsToRun = 6000;  // Menge der Steps, die der Stepper laufen soll. Eine komplette Runde sind 200 Steps.

const long powerOffDelay = 5000; // Verzögerung ab wann der Stepper nach jedem Lauf stromlos gemacht wird.

const int relayDelay = 2000; // Verzögerung in Milisekunden zwischen der Aktivierung des Relay und dem Start des Steppers
```

Zeichnungen

Grobe Funktion:





Material-Liste

Gekauft:

- 1x NEMA-17 Stepper Motor 17HS15-1504S-X1
- 1x Relay HW-482
- 1x Arduino Uno
- 1x DM320T Motortreiber
- Diverse KG/HT Rohre
- Diverse Kabel
- 5mm Stecker für Arduino
- 12V-1,5A Netzteil + evtl. 5mm Stecker
- Kleber für PLA (z.B. Pattex Ultra Gel Sekundenkleber)
- 4x Schrauben M3 für Stepper

Selbst gedruckt:

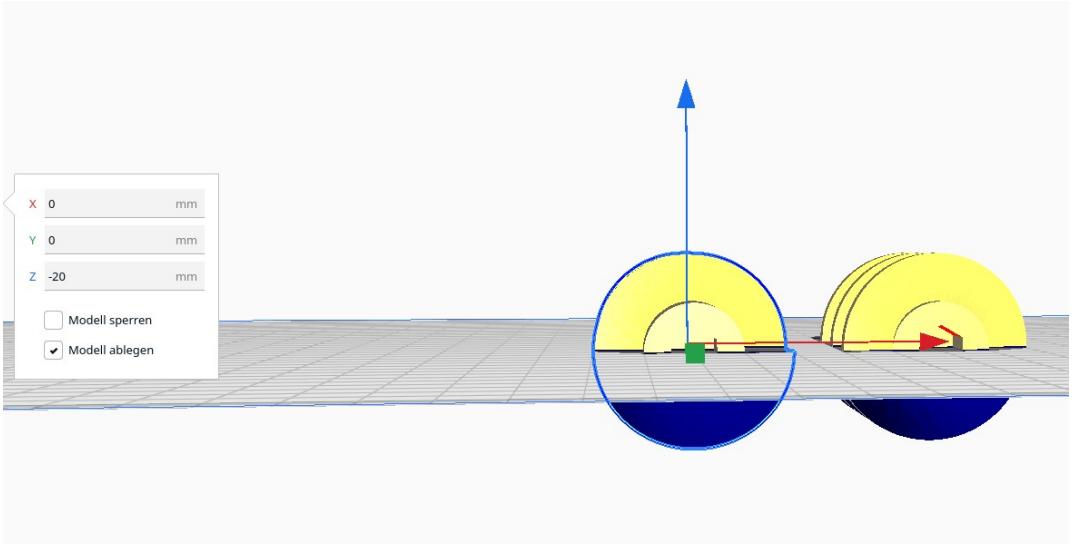
- 3x **3D-Druck** Förderschnecke/Wendel
- 1x **3D-Druck** Achsträger
- 1x **3D-Druck** Motorträger
- 1x **3D-Druck** Zahnrad-Achse
- 1x **3D-Druck** Achse-Achsträger

Montageanleitung/Fotos

Schritt 1:

Download der 3D-Druck Dateien und slicen mit einem Programm der Wahl z.B. Cura. Es wird empfohlen die Wendel horizontal zu trennen und doppelt auszudrucken, um Stützstruktur zu vermeiden. Danach können die beiden Hälften zusammengeklebt werden.

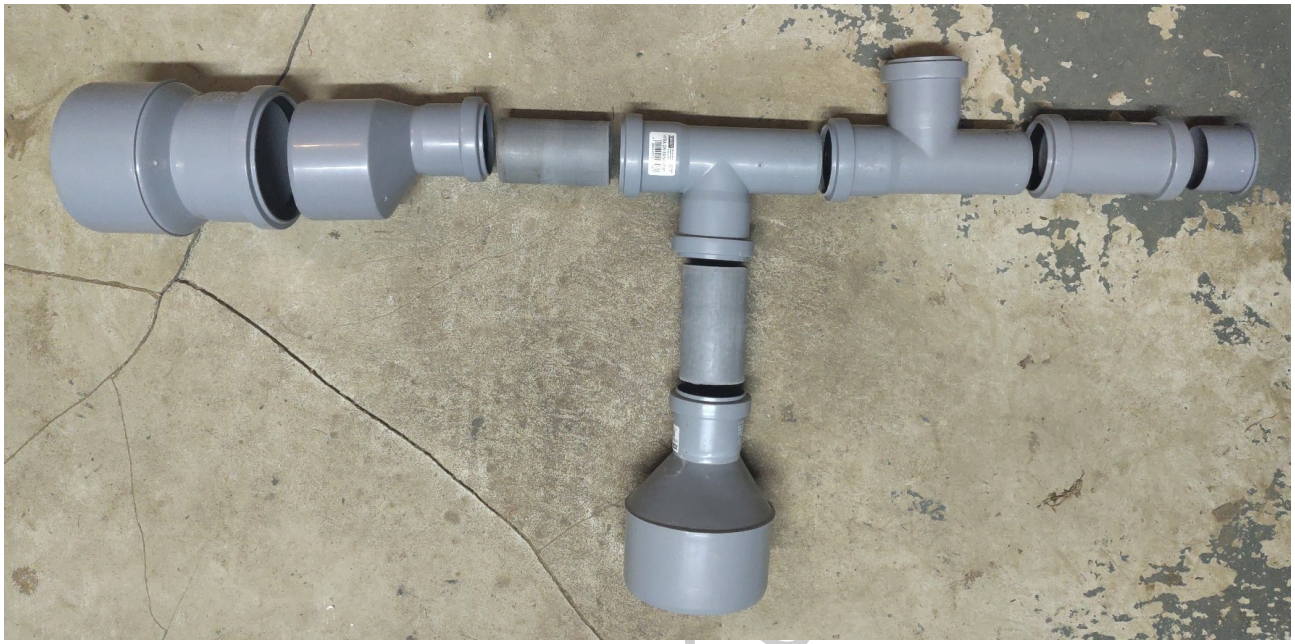
Achtung: Eine Hälfte muss um 180 Grad gedreht werden!



Die weiteren Bauteile können ohne weitere Komplikationen gedruckt werden.

Schritt 2:

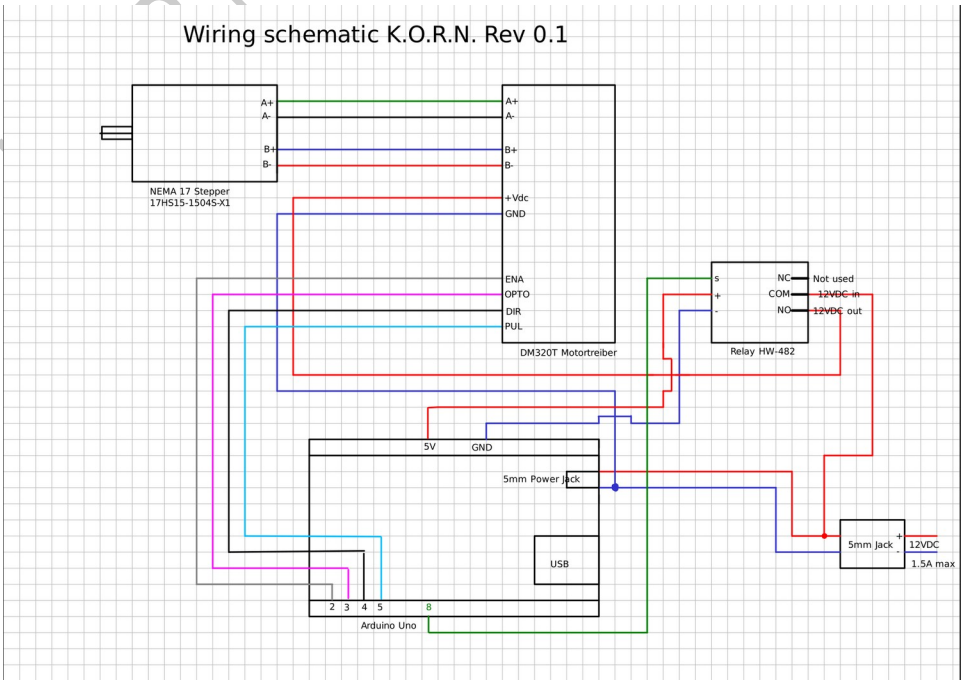
Zusammenfügen der KG-, HT-Rohre gemäß Foto:

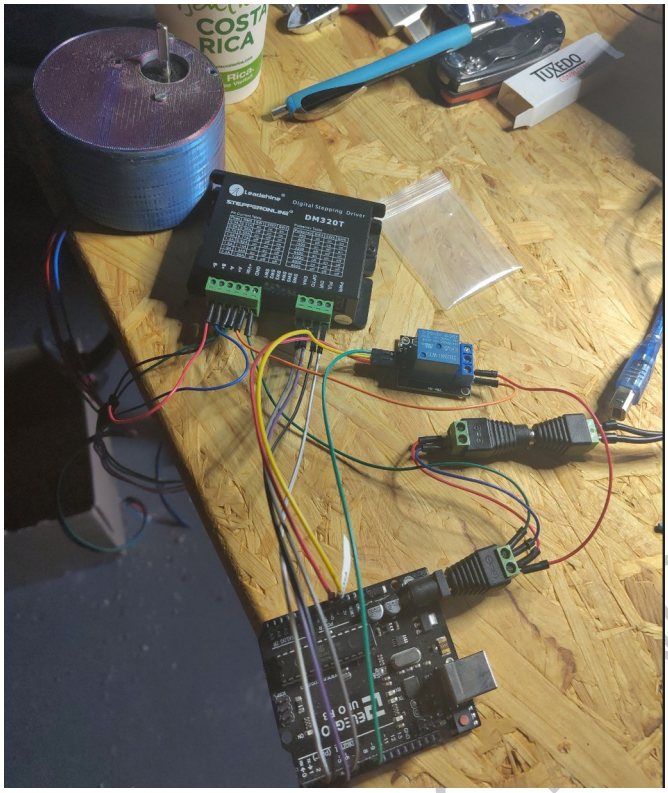


Schritt 3:

Einbau des Steppers in den Motorträger. Verkleben aller 3D-Druck Bauteile.

Verdrahten der Bauteile nach Zeichnung:





Schritt 4:

Flashen des Arduino mit dem Code:

```
#include <AccelStepper.h>


// Define the pins connected to the motor driver and relay
#define PUL_PIN 2
#define DIR_PIN 3
#define OPTO_PIN 4
#define ENA_PIN 5
#define RELAY_PIN 8


// Create an instance of the AccelStepper class
AccelStepper stepper(AccelStepper::DRIVER, PUL_PIN, DIR_PIN);


// Define variables for stepper parameters
int acceleration = 100; // Acceleration in steps per second per second
int targetRPM = 100;    // Target speed in RPM
int stepsToRun = 6000;  // Number of steps the stepper should run
const long powerOffDelay = 5000; // Delay in milliseconds to power off the stepper after each run (5 seconds)
const int relayDelay = 2000; // Delay in milliseconds between activating relay and starting stepper motor
```

```
// Define variables to track time
unsigned long previousMillis = 0;
const long initialDelay = 10 * 1000;  // 10 seconds in milliseconds
const long repeatInterval = 24 * 60 * 60 * 1000; // 24 hours in milliseconds

bool initialMovementDone = false;
bool relayActivated = false;

void setup() {
  // Set the maximum speed based on target RPM
  stepper.setMaxSpeed(targetRPM * 200); // 200 steps per revolution

  // Set the acceleration
  stepper.setAcceleration(acceleration);

  // Set the enable pin as an output and disable the stepper initially
  pinMode(ENA_PIN, OUTPUT);
  digitalWrite(ENA_PIN, LOW);

  // Set the relay pin as an output
  pinMode(RELAY_PIN, OUTPUT);
}

void loop() {
  unsigned long currentMillis = millis();

  // Check if the initial movement delay has passed and initial movement is not done
  if (!initialMovementDone && currentMillis - previousMillis >= initialDelay) {
    // Activate the relay if it's not already activated
    if (!relayActivated) {
      digitalWrite(RELAY_PIN, HIGH);
      relayActivated = true;
      delay(relayDelay); // Delay before starting the stepper motor
    }

    // Enable the stepper
    digitalWrite(ENA_PIN, HIGH);

    // Move the stepper motor the specified number of steps counterclockwise
    stepper.moveTo(-stepsToRun);
    stepper.runToPosition();

    // Disable the stepper
    digitalWrite(ENA_PIN, LOW);
```

```
    // Deactivate the relay
    digitalWrite(RELAY_PIN, LOW);

    // Set initial movement flag
    initialMovementDone = true;

    // Update previousMillis for the repeat interval
    previousMillis = currentMillis;
  }

  // Check if 24 hours have passed after the initial movement
  if (initialMovementDone && currentMillis - previousMillis >= repeatInterval) {
    // Activate the relay if it's not already activated
    if (!relayActivated) {
      digitalWrite(RELAY_PIN, HIGH);
      relayActivated = true;
      delay(relayDelay); // Delay before starting the stepper motor
    }

    // Enable the stepper
    digitalWrite(ENA_PIN, HIGH);

    // Move the stepper motor the specified number of steps counterclockwise
    stepper.moveTo(-stepsToRun);
    stepper.runToPosition();

    // Disable the stepper
    digitalWrite(ENA_PIN, LOW);

    // Delay to keep the stepper powered off for a specified duration
    delay(powerOffDelay);

    // Deactivate the relay
    digitalWrite(RELAY_PIN, LOW);

    // Update previousMillis for the next repeat interval
    previousMillis = currentMillis;
  }
}
```

Schritt 5:

Zusammenbau von Motorträger, Wendel und Achsträger.

Finaler test, der Motor dreht wenige Sekunden, nachdem das Arduino ein Power-up macht. Danach alle 24h. Evtl anpassen der fett gedruckten Werte an die eigenen Bedürfnisse. Sie auch Beschreibung.

Lizenz

Ausblick

Geplant ist vorerst das Projekt zu veröffentlichen und mit anderen Geflügelhaltern ins Gespräch zu kommen.

Auf diesem Wege soll konstruktive Kritik, Ideen und Anregungen gesammelt werden.

Gleichzeitig läuft der Prototyp täglich um Schwachstellen, hauptsächlich in den Wintermonaten, festzustellen.

Ein fertiges Muster soll gegen Ende des Frühlings 2025 fertig sein. Gleichzeitig wird an weiteren Erweiterungen gearbeitet:

- Austausch des alten Arduino UNO gegen ein WLAN fähiges Gerät
- Anbindung in einen Telegram-Bot oder ähnliches um auch im Urlaub eine tägliche Statusmeldung über durchgeführte Fütterungen zu erhalten
- Zeiterfassung via WLAN oder RTC
- Manuelle Bedienelemente mit Display um am Gerät die Fütterungszeit, die Menge und eine manuelle Abgabe einstellen zu können.
- Förderschnecke in stabilerem Material drucken als PLA