Załucki Roman

Rudny Maksymilian

Sonmez Isil

Mechatronic Design

Automatic litter box report

State of the art

Project assumptions

The project was developed with a thought of building an easy to use, cheap automatic litter box. We based our project on two assumptions. We wanted to design a product working on a competitive level that at the same time is more affordable then the rest.

Target audience

Our project was developed for cat owners that would like to get rid of some of the weight off their shoulders that comes from owning a pet. Automatic litter box is perfect for people that like or have to travel and also for everyone that want to have hygienic way of getting rid of pet waste.

Design assumptions

During development we had to decide methods for each part to met all of the requirements. Our product had to be automatic, with enough space for litter and waste, with an easy way to dispose of accumulated used sand.

Morphological analysis

The fist element of the design we had to decide on was the automatic cleaning system. First we thought about using rake that would be moving on rails through the whole part of the space designed for storing litter but we quickly abandoned that idea because of the problems that would come from the resistance of the sand, safety and a way to deliver used litter to the waste drawer. There was also an idea to use a second rail right outside the opening to the waste compartment but it also came with issues in designing a good sealed system of disposing said waste. Finally we decided on using rotation to move around the litter in a way that separates the waste from the rest of the sand with the waste drawer on the bottom.

Second design problem was whether we want to use an automatic litter refilling system or not. Automatic system although would be more effective it would come with more cost and space issues so we settled on manual way.

Third problem was the cat detection system that would start the self-cleaning process after enough time so the litter would lump. After quick research we decided on setting the timer for 20 minutes after detecting change in distance read by the ultrasonic sensor placed on top.

Morphological table:

Nr.	Aspect/Function	option 1	functionality (price) 1	option 2	functionality (price) 2	option 3	functionality (price) 3
1.	cleaning mechanism	rotation using net	9 (9)	translation using 1 rail	3 (7)	translation using 2 rails	5 (4)
2.	litter refilling	manual	5 (10)	automatical	7 (2)	X	
3.	waste disposal pocket placing	under the litter box	8 (7)	on the side of the litter box	5 (7)	at the back of the litter box	5 (7)
4.	average time between refills	1 week	4 (8)	2 weeks	7 (6)	3 weeks	9 (3)
5.	cleaning process duration	20 seconds	5 (5)	30 seconds	7 (5)	40 seconds	8 (5)
6.	design	minimalistic	9 (6)	futuristic	4 (6)	classic litterbox	1 (10)
7.	timer delay	15 minutes	5 (5)	25 minutes	7 (5)	20 minutes	8 (5)

Design:

Our design consists of eight models:

1) Rotating ball:

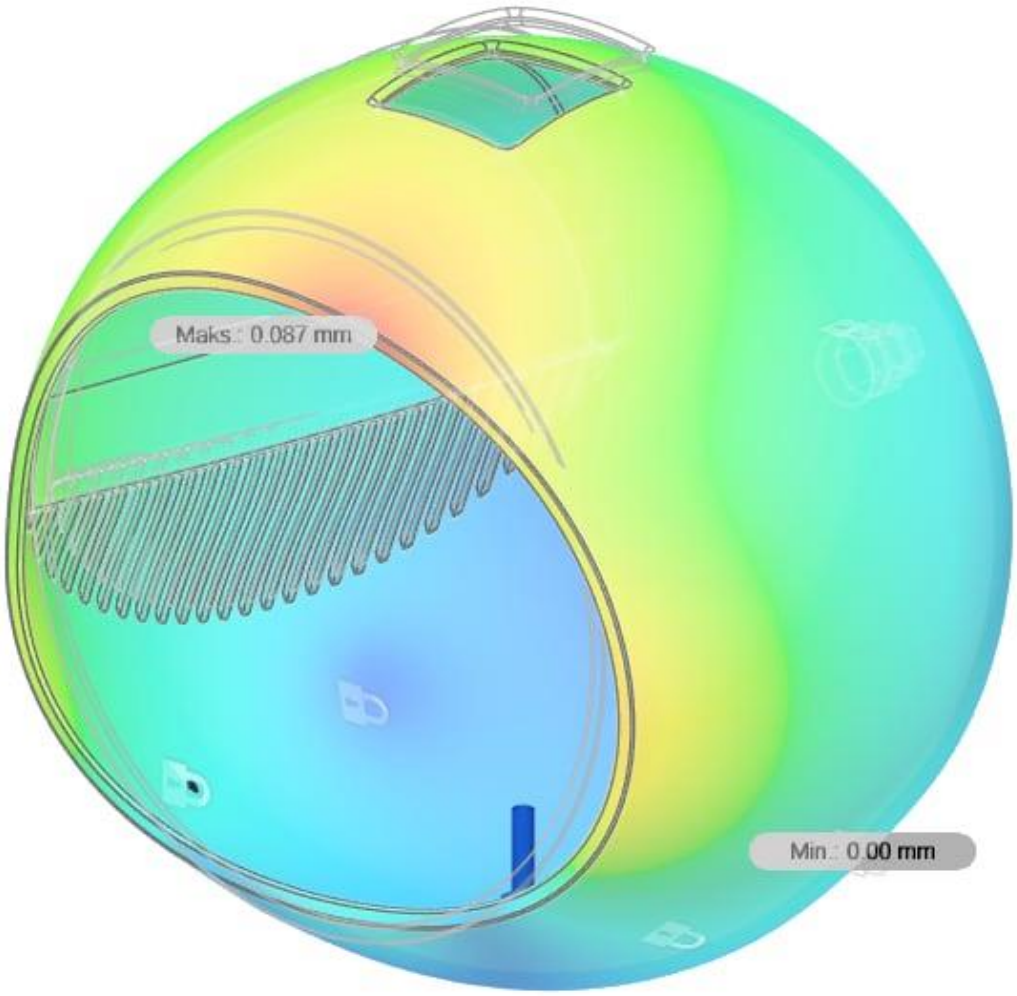
The design process of this element was fairly simple. At first we were thinking of making this part a cylinder, but we quickly abandoned that idea and opted for the ball shape for easier waste disposal. On the back of this element there is a crankshaft used to rotate the ball.

Strength analysis:

Maximum force on 1 support wheel:

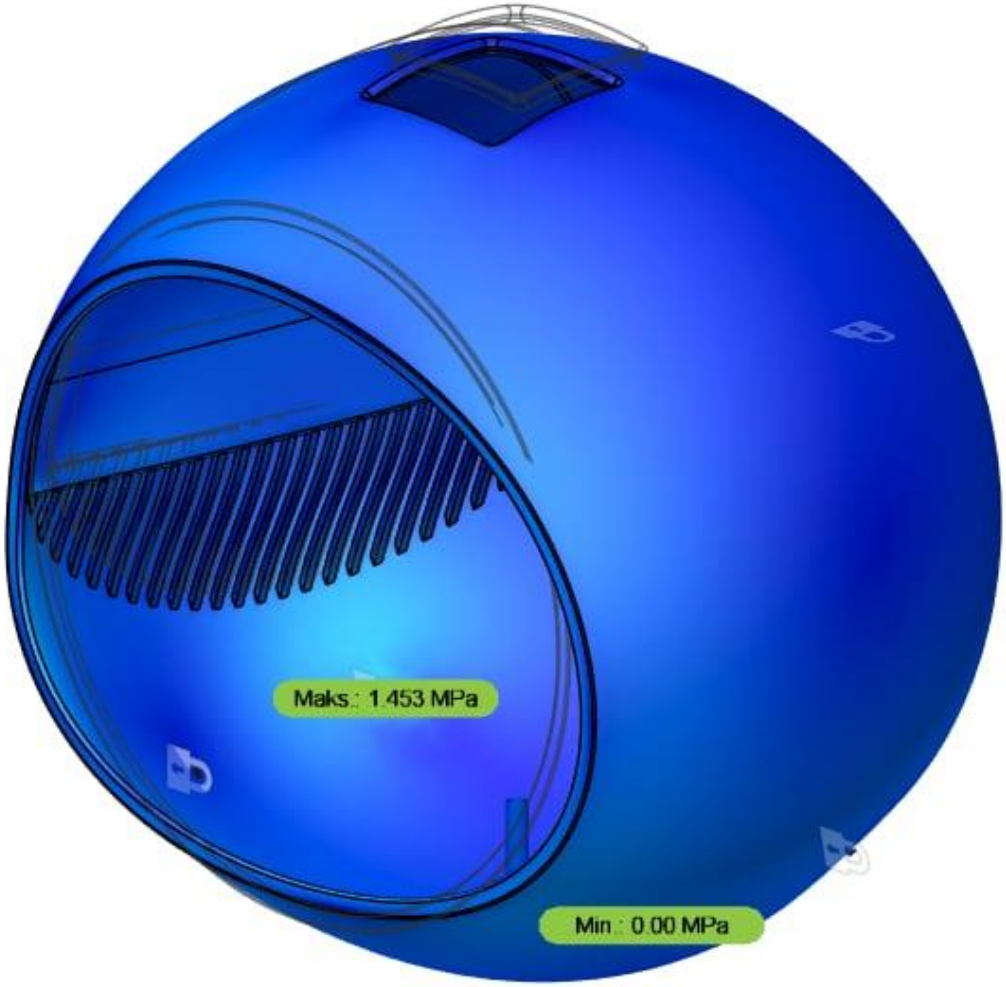
Force exerted inside the ball was 200 N, and maximum force on one wheel is less than 50 N, so our choice of PLA is valid.

Maximum deformation:



The element deformed only 0.087mm. Taking into consideration the size of this ball this deformation does not matter.

Maximum tension:

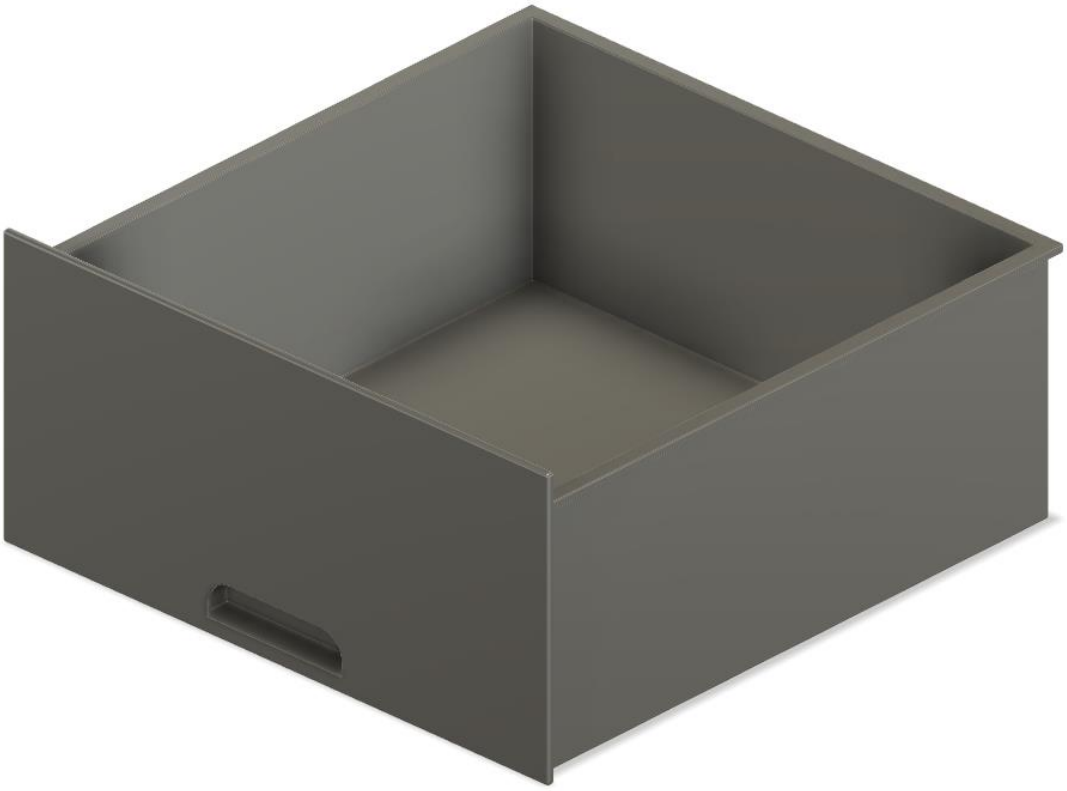


PLA is safe to use up to 10 MPA so this result is satisfying.

2) Base of the litter box:

In the design of this part there was some concern about mounting the ultrasonic sensor, but it was fitted by slimming the casing walls, so this part works as rails for the lid. On the bottom there is space for the waste drawer and all electronic elements. The used litter compartment is placed in front, so its easily removable. There is space for fitting a seal around the drawer.

3) Waste drawer:

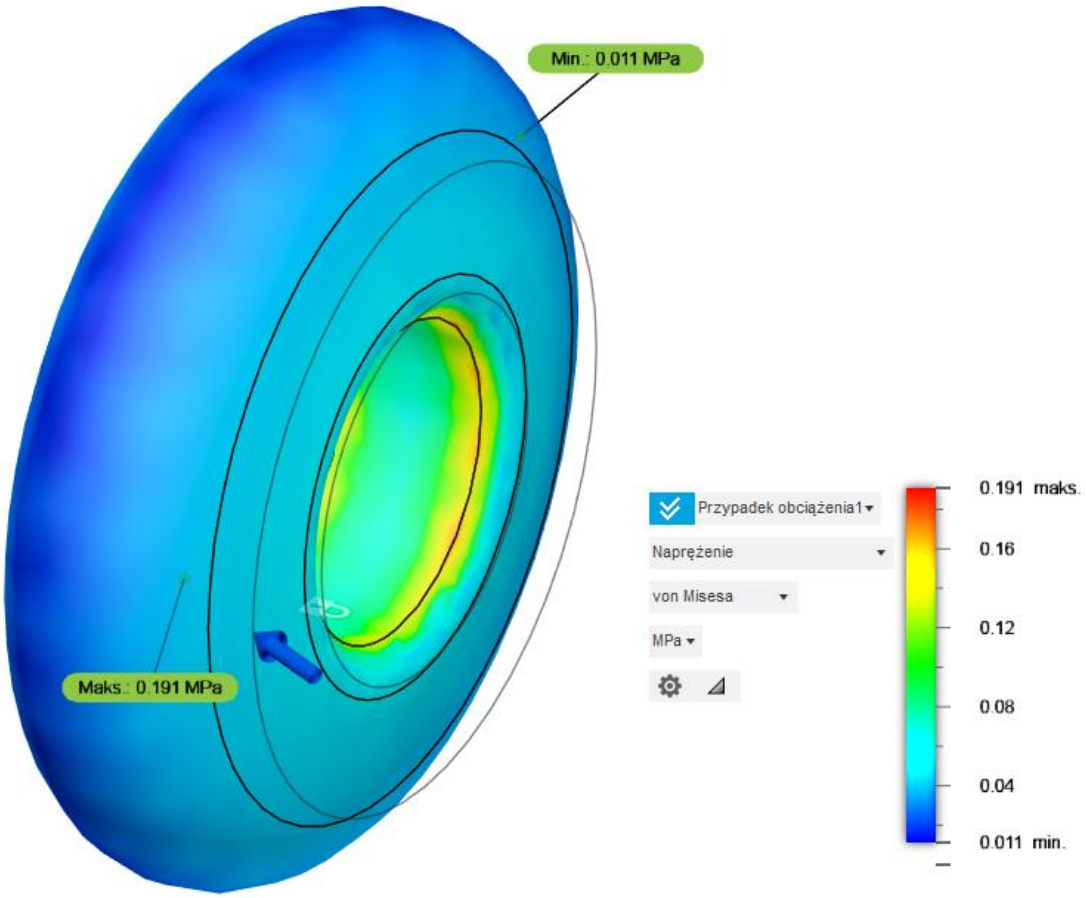


This element is part is visible with the casing on so the front of this drawer had to look minimalistic in front.

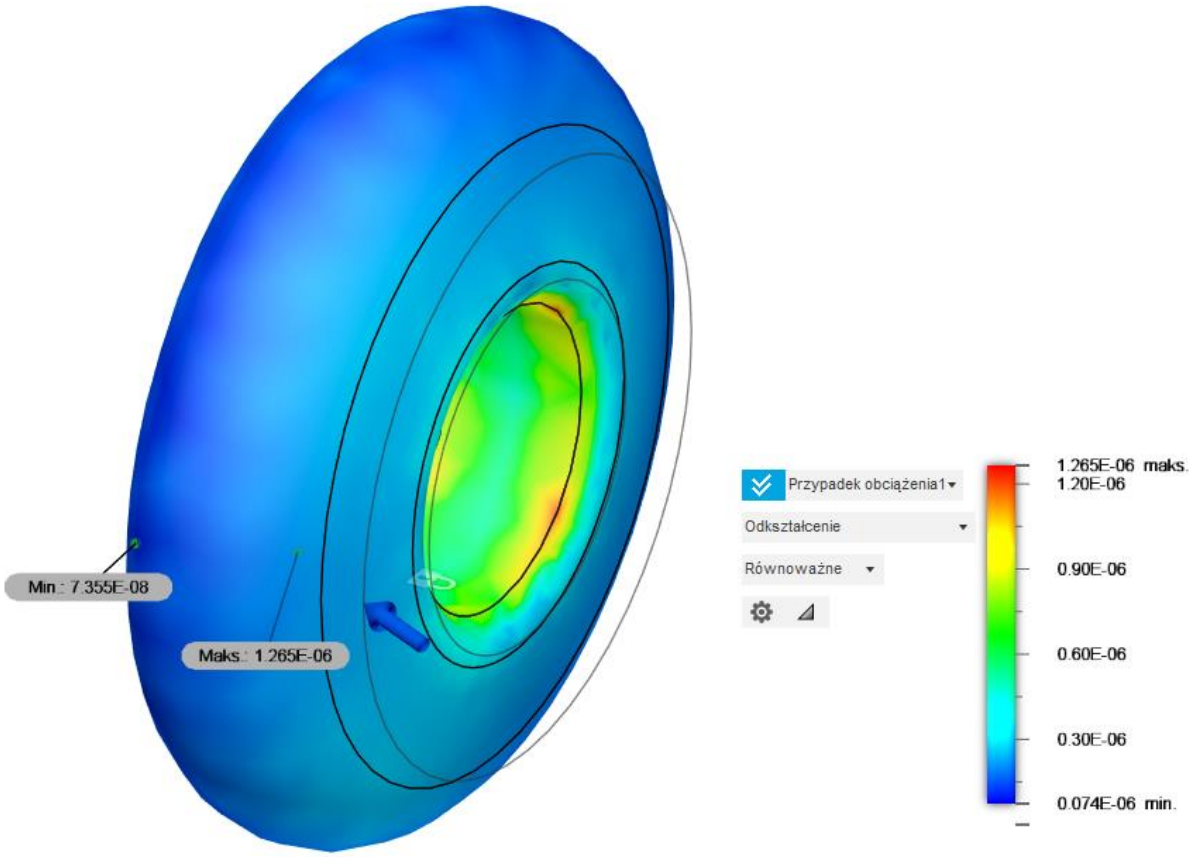
4) Support wheel:

This part was designed to support the rotating ball.

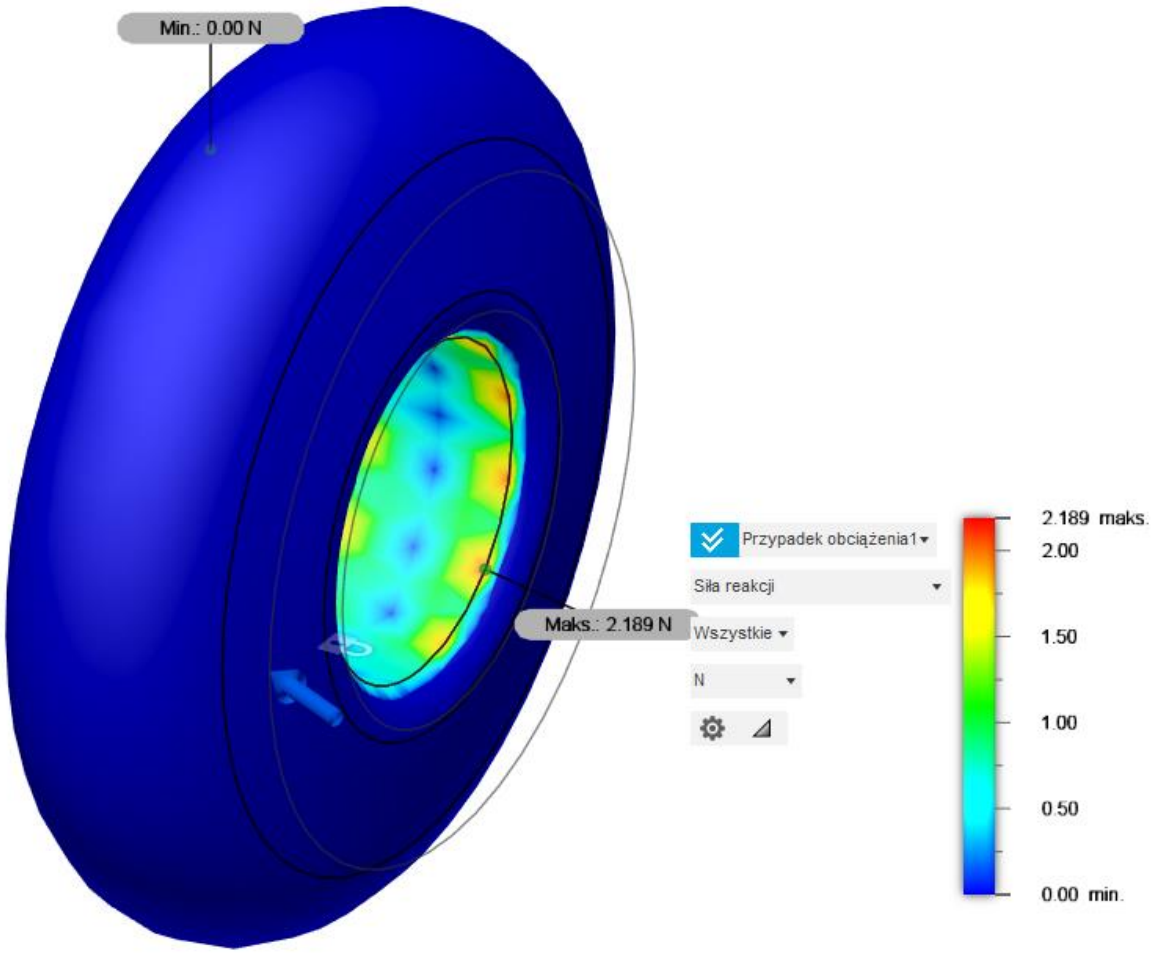
Strength analysis:



PLA is safe to use up to 10 MPA so this result is satisfying.

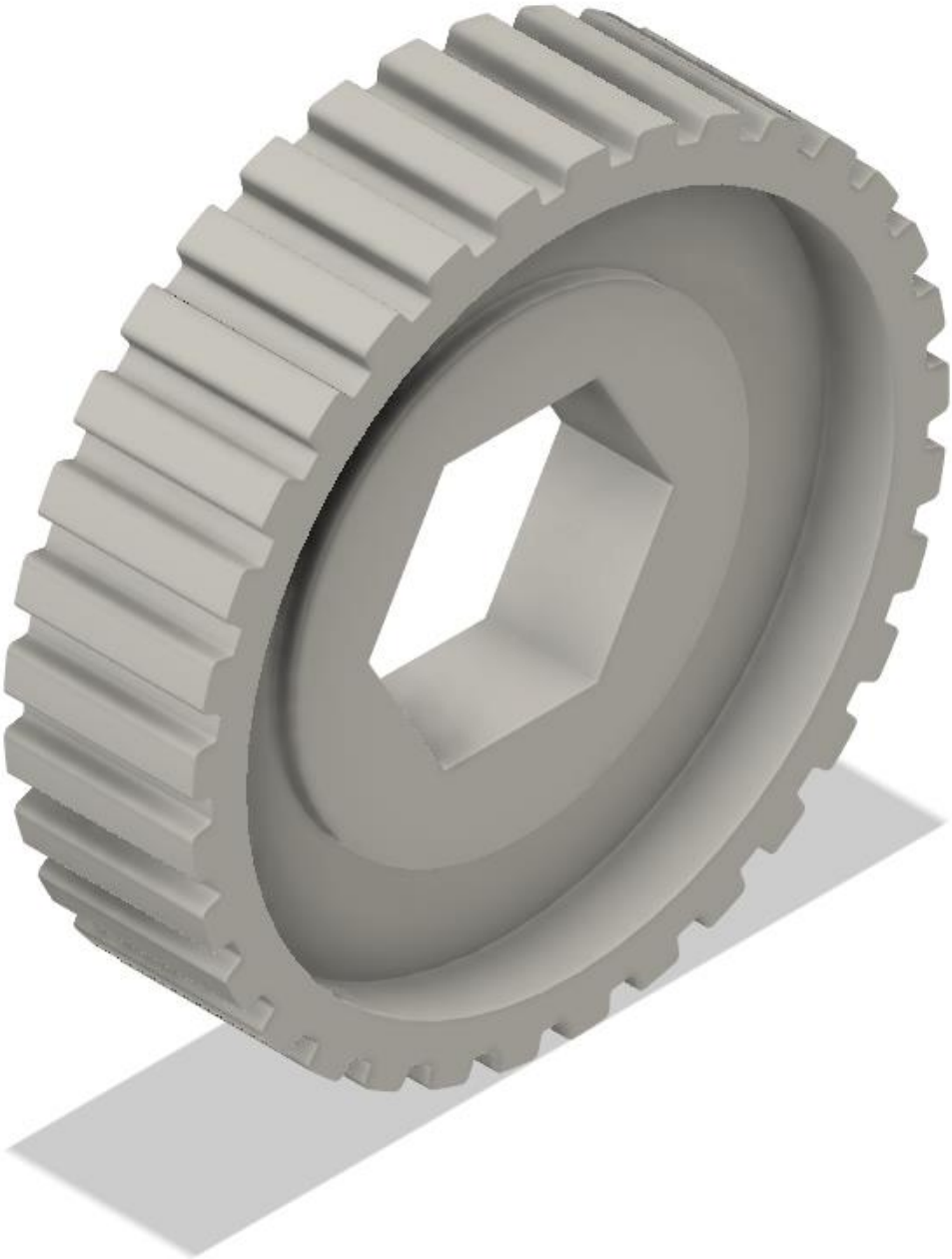


The element deformed by marginal values



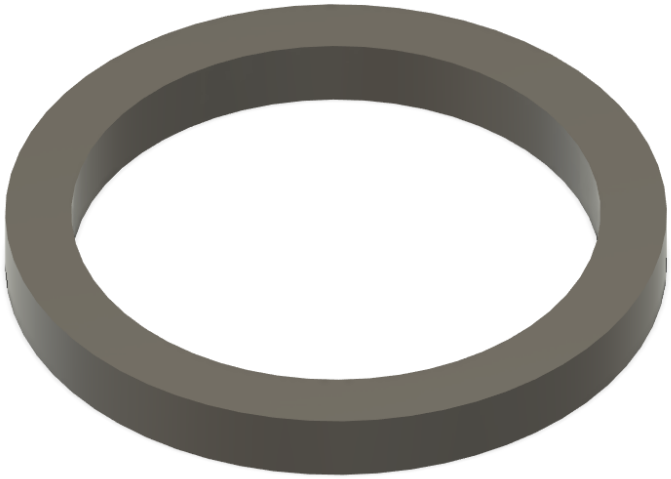
Maximum reaction force on this element was around 2 N, the force exerted on the part was diagonal and placed near od suspected point of contact with the rotating element and its value was 50 N, this value was taken from strength test od the ball element.

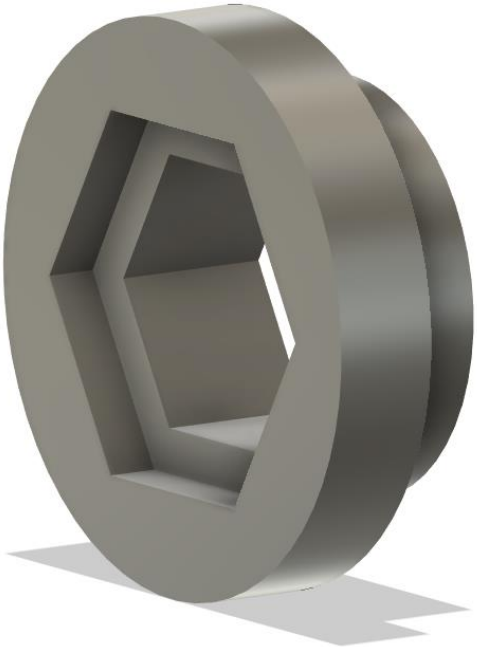
5) Crankshaft pulley:



This crankshaft pulley was designed to work with AT10 toothed belt and it is responsible for rotating the ball.

6) Sleeves:



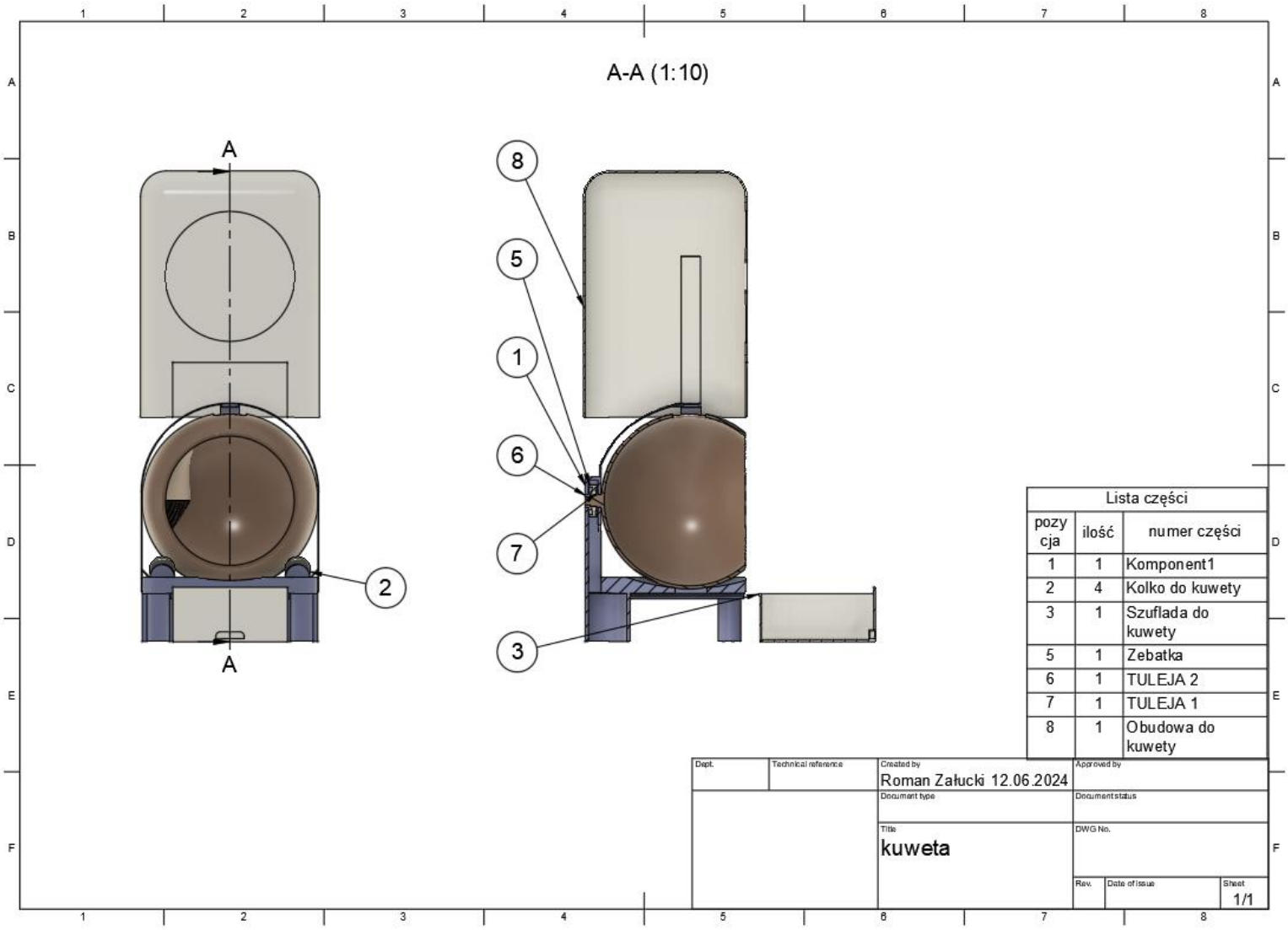


Those sleeves were designed to keep the crankshaft pulley in place and allow the elements to rotate as one rigid part.

7) Casing:

During planning this part we were thinking of minimalistic look. Designing this part was dictated by previous parts since it is supposed to cover them up neatly.

Assembly drawing:



Lista części		
pozycja	ilość	numer części
1	1	Komponent1
2	4	Kolko do kuwety
3	1	Szuflada do kuwety
5	1	Zebatka
6	1	TULEJA 2
7	1	TULEJA 1
8	1	Obudowa do kuwety

Dept.	Technical reference	Created by: Roman Załucki 12.06.2024	Approved by
		Document type	Document status
		Title: kuweta	DWG No.
			Rev. / Date of issue / Sheet: 1/1

Electronic design:

When it comes to the electronical design we use 4 elements:

1) Arduino UNO

We opted for the microprocessor that we are well accustomed to. Arduino is easy to set up and easy to work with. The other upside is that we did not have to buy a new board because we had one.

2) Stepper motor JK57HS76-2804

The only requirement for the stepper motor is that it is strong enough to rotate the ball, so we went with JK57HS76-2804 with a torque equal to 1,89Nm which is more than enough.

Bipolar four-wire stepper motor with 200 steps per revolution (1.8 degrees). It is powered by 3 V, draws 2800 mA per coil. Torque is 19.0 kg*cm (1.89 Nm).

3) Ultrasonic distance sensor HC-SR04

When it comes to the sensor which will detect the cat we will use the classical HC-SR04, which we have a lot of experience with.

Range: from 2 cm to 200 cm.
Powered with voltage of 5 V.

4) Power source

The only requirement that the power source had to met is enough voltage for both Arduino and the stepper motor.

Voltage: from 110 V to 220 V.
Output voltage: 12 VDC.
Output current: 5 A.

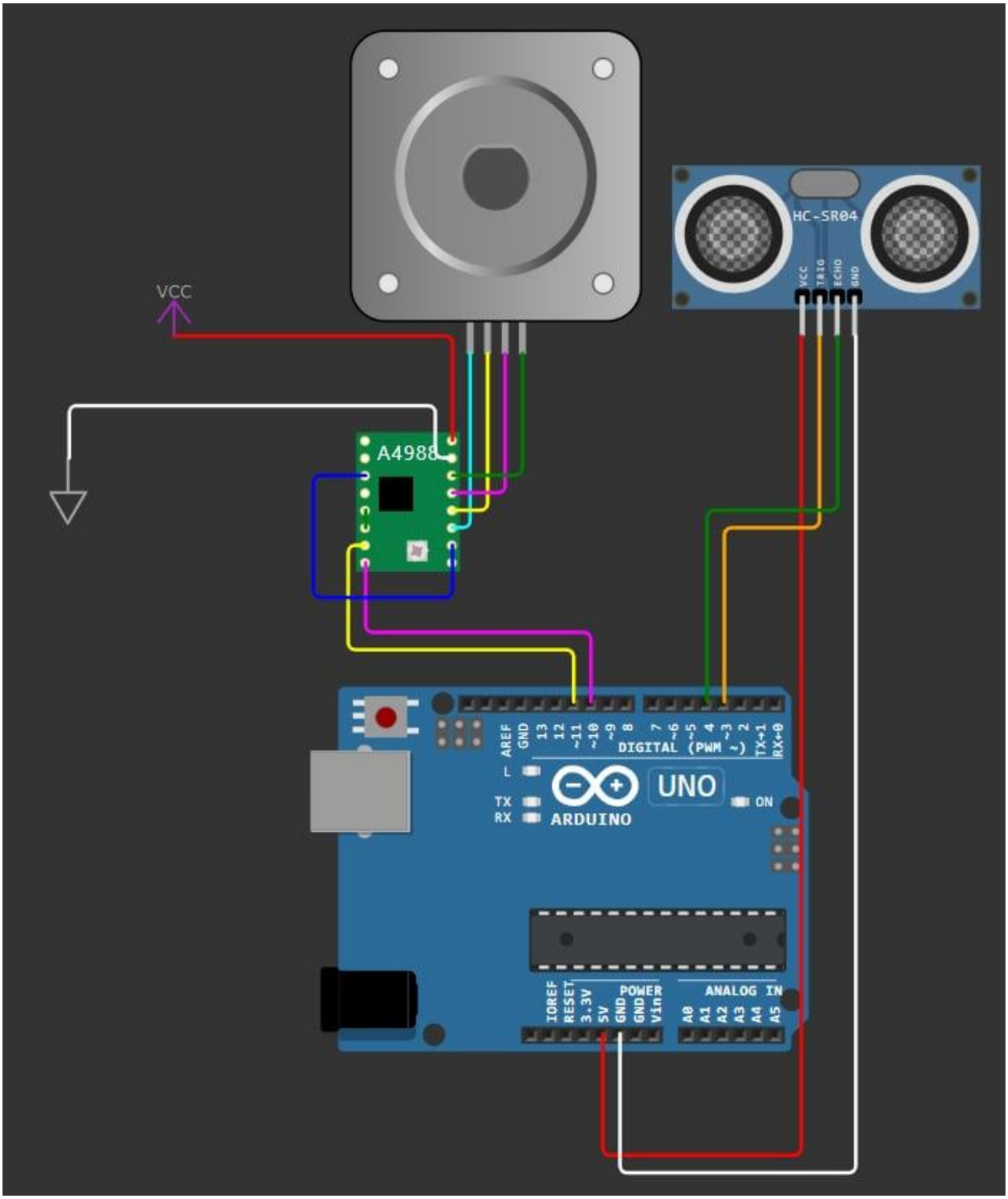
5) Stepper motor driver Bigtreetech TMC2226

Again only requirement that the driver had to met was enough power to be used with our stepper motor and that would allow us to precisely control the motion of the motor.

Voltage: from 4,75V to 29V

Max current: 2,8A

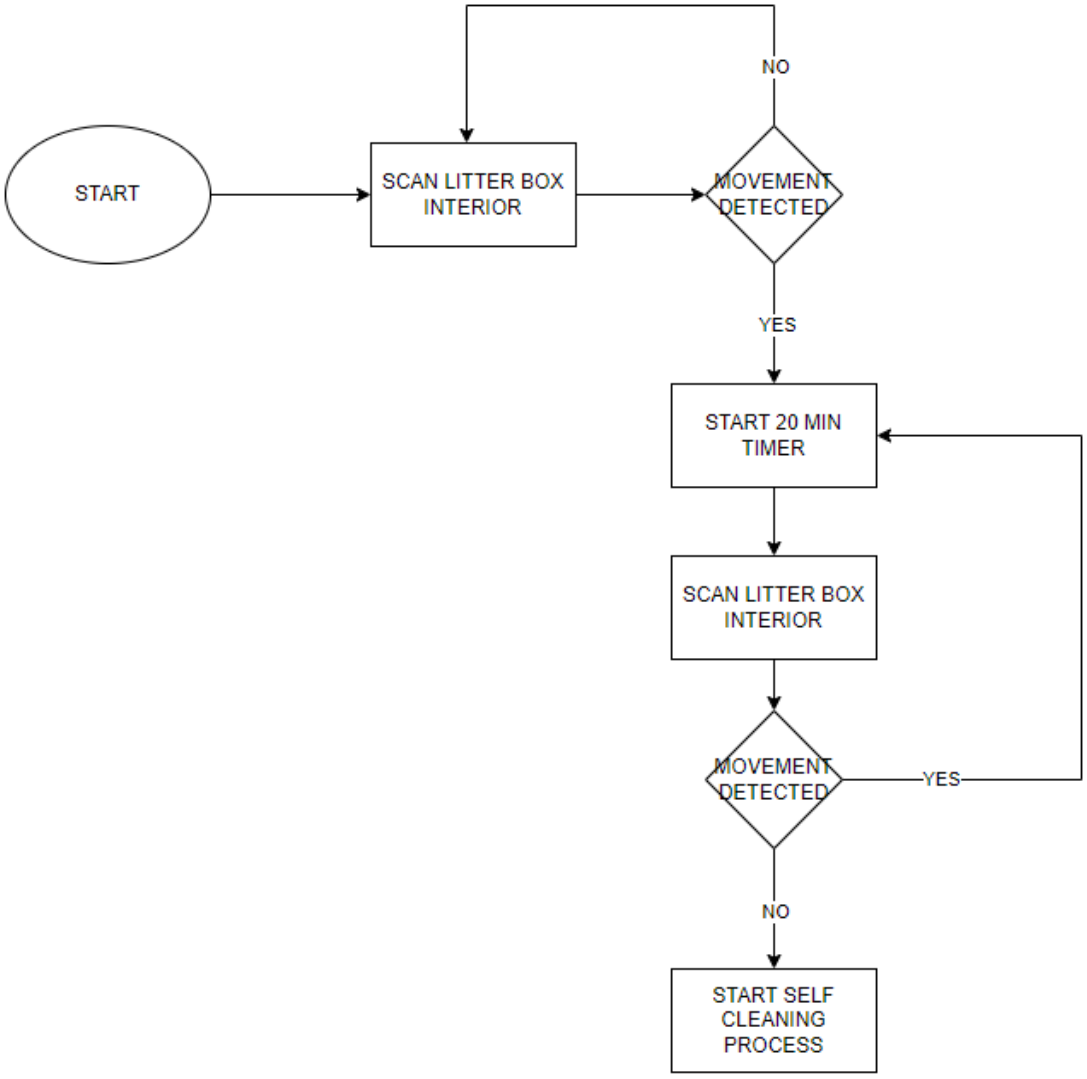
Electronical assembly



Here we have a visual demonstration od the electronical assembly. The only thing that differs from the real project is the stepper motor driver because we could not find an Arduino assembly maker with such model. Assembly is not very complicated as there are few elements.

Software design

Operation block diagram



As previously stated we worked on Arduino UNO

Arduino prototype code:

```
#include <Stepper.h>

const int stepsPerRevolution = 2048;

const int trigPin = 6;
const int echoPin = 7;

const int stepPin1 = 8;
const int stepPin2 = 9;
const int stepPin3 = 10;
const int stepPin4 = 11;



Stepper myStepper(stepsPerRevolution, stepPin1, stepPin2, stepPin3, stepPin4);

long duration;
int distance;
int previousDistance = 0;
unsigned long timerStart = 0;
bool timerRunning = false;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  myStepper.setSpeed(3000);
}

void loop() {
```

```
// Measure distance
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);
distance = duration * 0.034 / 2;



if (abs(distance - previousDistance) > 20) {
  resetTimer();
}



if (timerRunning) {
  unsigned long elapsed = millis() - timerStart;
  unsigned long remaining = 1200000 - elapsed;

  if (remaining > 1000) {
    Serial.print("Countdown: ");
    Serial.print(remaining / 1000);
    Serial.println(" seconds remaining");
    delay(1000);
  } else {
    Serial.println("Countdown finished! Starting stepper movements.");



    moveStepper(stepsPerRevolution * 0.5, 30000);
```

```
      delay(500);

      moveStepper(-stepsPerRevolution * 0.625, 40000);
      delay(500);

      moveStepper(stepsPerRevolution * 0.125, 15000);
      delay(500);
      timerRunning = false;
      Serial.println("Stepper movements completed. Timer reset.");
    }
  }

  previousDistance = distance;
  delay(50);
}

void moveStepper(int steps, int duration) {
  int stepDelay = duration / abs(steps);

  if (steps > 0) {
    for (int i = 0; i < steps; i++) {
      myStepper.step(1);
      delay(stepDelay);
    }
  } else {
    for (int i = 0; i < abs(steps); i++) {
      myStepper.step(-1);
      delay(stepDelay);
    }
  }
```

```
}

void resetTimer() {
  timerStart = millis();
  timerRunning = true;
  Serial.println("Timer reset due to change in distance.");
}
```