26 de mayo de 2023.
Dictamen C.I. 07/2023

DICTAMEN
QUE PRESENTA LA COMISIÓN DE INVESTIGACIÓN DE LA DIVISIÓN DE CIENCIAS DE LA COMUNICACIÓN Y DISEÑO

ANTECEDENTES

I. El Consejo Divisional de Ciencias de la Comunicación y Diseño, en la sesión 08.23, celebrada el 2 de mayo de 2023, integró esta Comisión en los términos señalados en el artículo 55 de Reglamento Interno de los Órganos Colegiados Académicos.

II. El Consejo Divisional designó para esta Comisión a los siguientes integrantes:

a) Órganos personales:
- ✓ Dra. Margarita Espinosa Meneses, Jefa del Departamento de Ciencias de la Comunicación;
- ✓ Dra. Erika Cecilia Castañeda Arredondo, Jefa del Departamento de Teoría y Procesos del Diseño;
- ✓ Dr. Carlos Roberto Jaimez González, Jefe del Departamento de Tecnologías de la Información.

b) Representantes propietarios:
- Personal académico:
  - ✓ Dr. Diego Carlos Méndez Granados, Departamento de Ciencias de la Comunicación;
  - ✓ Dr. Manuel Rodríguez Viqueira, Departamento de Teoría y Procesos del Diseño;
  - ✓ Mtra. Betzabet García Mendoza, Departamento de Tecnologías de la Información.

CONSIDERACIONES

I. La Comisión recibió, para análisis y discusión, el segundo reporte parcial de resultados del proyecto de investigación denominado *"**Geometría en Movimiento 3**"* presentado por la Dra. Dina Rochman Beer, aprobado en la Sesión 13.20 celebrada el 24 de noviembre de 2020, mediante el Acuerdo DCCD.CD.04.13.20.

II. La Comisión de Investigación sesionó el 26 de mayo de 2023, fecha en la que concluyó su trabajo de análisis y evaluación del reporte parcial de resultados, con el presente Dictamen.

III. La Comisión tomó en consideración los siguientes elementos:

- *"Lineamientos para la creación de grupos de investigación y la presentación, seguimiento y evaluación de proyectos de investigación"* aprobados en la Sesión 06.16 del Consejo Divisional de Ciencias de la Comunicación y Diseño, celebrada el 6 de junio de 2016, mediante al acuerdo DCCD.CD.15.06.16.
- Protocolo de investigación.
- Relevancia para el Departamento.
- Objetivos planteados.
- Resultados obtenidos.

IV. **Objetivo general**:

Realizar un pantógrafo, al que se le llamará "Pantógrafo XYZ" con el cual se pueda reproducir las curvas de las partes del cuerpo de los animales de pequeña escala para realizar su impresión 3D a mayor escala.

V. **Actividades realizadas durante el segundo año:**

En el primer reporte del proyecto de investigación "Geometría en Movimiento 3", se entregó el primer boceto del Pantógrafo XYZ. En el video que se presentó se observó que el Pantógrafo XYZ funcionó manualmente correctamente, sin embargo, para poder desarrollar el pantógrafo final, se corrigió el boceto y se realizaron varias pruebas para analizar el mecanismo que tendrá el prototipo final y la profesora presenta el desarrollo de estas pruebas.

DICTAMEN

ÚNICO:
Tras evaluar el segundo reporte parcial de resultados del proyecto de investigación denominado **"Geometría en movimiento 3"** presentado por la Dra. Dina Rochman Beer, la Comisión de Investigación recomienda al Consejo Divisional de Ciencias de la Comunicación y Diseño aceptarlo.

VOTOS:

Integrantes	Sentido de los votos
Dra. Margarita Espinosa Meneses	A favor
Dra. Erika Cecilia Castañeda Arredondo	A favor
Dr. Carlos Roberto Jaimez González	A favor
Dr. Diego Carlos Méndez Granados	A favor
Dr. Manuel Rodríguez Viqueira	A favor
Mtra. Betzabet García Mendoza	A favor
Total de los votos	**6 votos a favor**

Coordinadora

Mtra. Silvia Gabriela García Martínez
Secretaria del Consejo Divisional de Ciencias de la Comunicación y Diseño

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

Ciudad de México 27 de febrero 2023

DTPD.042.23

Asunto:
Reporte anual de avances del proyecto:
"Geometría en Movimiento 3"

Dra. Gloria Angélica Martínez de la Peña
Presidenta del Consejo Divisional
División de Ciencias de la Comunicación y Diseño
Universidad Autónoma Metropolitana
Unidad Cuajimalpa
Presente

Por este medio hago de su conocimiento el segundo reporte anual de avances del proyecto de investigación "Geometría en Movimiento 3", cuya responsable es la Dra. Dina Rochman Beer, para su dictamen y aprobación.

El proyecto de investigación "Geometría en Movimiento 3" fue aprobado por el Consejo Divisional de la DCCD en la Sesión 13.20 del Consejo Divisional, mediante el acuerdo DCCD.CD.04.13.20 del 24 de noviembre de 2020, por un período de tres años, del 25-nov-20 al 24-nov-23.

Para su análisis y dictaminación, **se anexan los siguientes documentos**:

Reporte de avance anual de la Investigación.

Probatorios de los productos de investigación.

División de Ciencias de la Comunicación y Diseño
Unidad Cuajimalpa
DCCD | División de Ciencias de la Comunicación y Diseño
Jefatura del Departamento de Teoría y Procesos del Diseño
Torre III, 5to. piso. Av. Vasco de Quiroga 4871,
Colonia Santa Fe Cuajimalpa. Alcaldía Cuajimalpa de Morelos.
C.P. 05348, Ciudad de México.
Tel.: (+52) 55.5814.5348
http://dccd.cua.uam.mx

De igual forma, se anexan los siguientes documentos, con la intención de contextualizar el proyecto:

- Aprobación en el Consejo Divisional de CCD.

Sin más por el momento, quedo a sus órdenes para cualquier duda o aclaración y le envío un cordial saludo.

Atentamente

Casa abierta al tiempo

Dra. Erika Cecilia Castañeda Arredondo

Jefa del Departamento de Teoría y procesos del Diseño

*ccp. Archivo

28 de febrero del 2023.

Dra. Erika Cecilia Castañeda Arredondo

Jefa del Departamento de Teoría y Procesos del Diseño

Presente

Estimada Dra. Castañeda:

Por este medio le solicito de la manera más atenta se turne el segundo informe que le he enviado por WeTransfer del Proyecto de Investigación "Geometría en Movimiento 3" a la comisión de investigación y al consejo divisional para su análisis, discusión y aprobación.

Sin más por el momento, agradeciéndole su apoyo.

Atentamente

Dra. Dina Rochman Beer

PANTÓGRAFO XYZ

Dra. Dina Rochman Beer

2022

Agradecimientos

Quiero expresar mi agradecimiento a la Lic. Miriam Hernández Ramírez de MAC de la Fes Acatlán por su apoyo en el análisis de la movimiento de los engranajes y en la asesoría para la programación.

Quiero expresar mi agradecimiento al Lic. Alfredo Almaraz por la impresión 3D y funcionamiento del sensor pulsador.

Quiero expresar mi agradecimiento al Mtro. Jesús Hernández Cadena por su apoyo en las especificaciones técnicas para la construcción del prototipo e integración parcial de los componentes para el modelo funcional.

PANTÓGRAFO XYZ

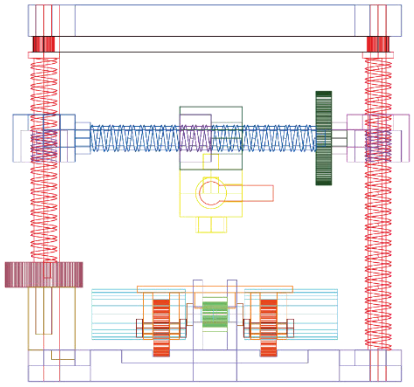
PROYECTO DE INVESTIGACIÓN SEGUNDO REPORTE

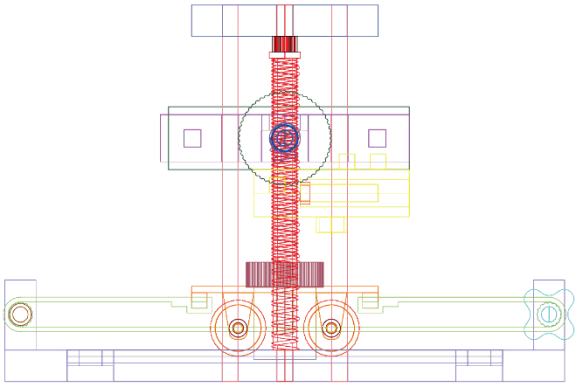
GEOMETRÍA EN MOVIMIENTO 3

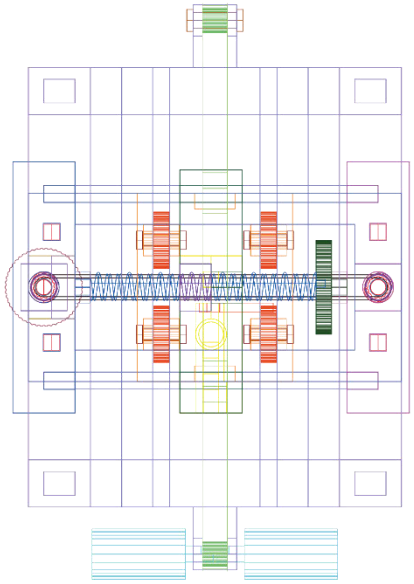
Dra. Dina Rochman Beer

2022

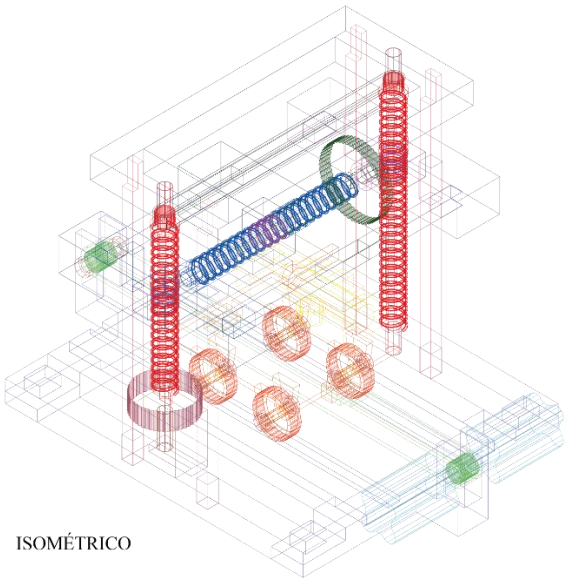
En el primer reporte del proyecto de investigación “Geometría en Movimiento 3”, se entregó el primer boceto del Pantógrafo XYZ.



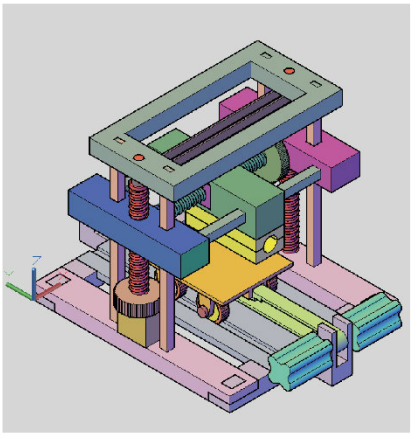




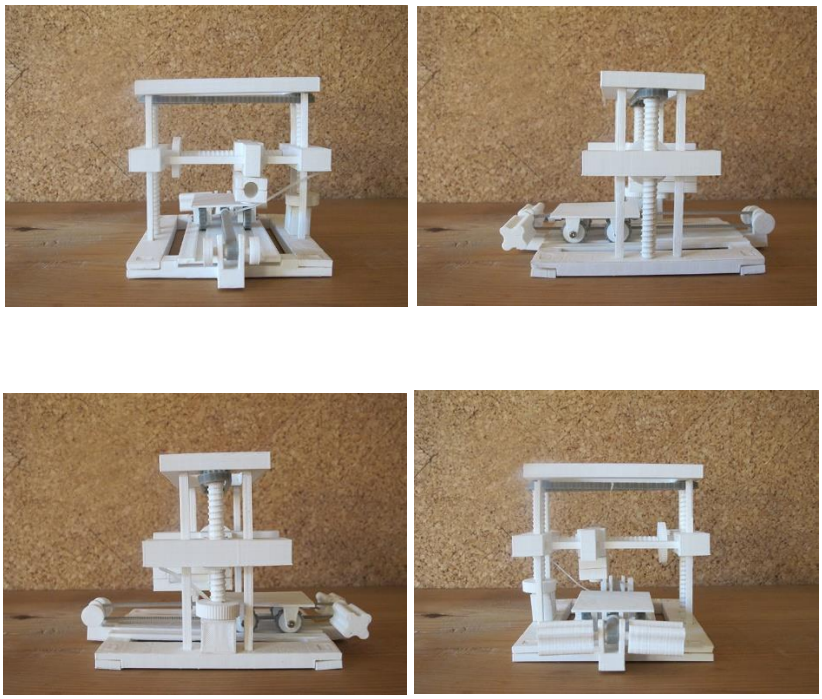
MONTEA



ISOMÉTRICO



ISOMÉTRICO

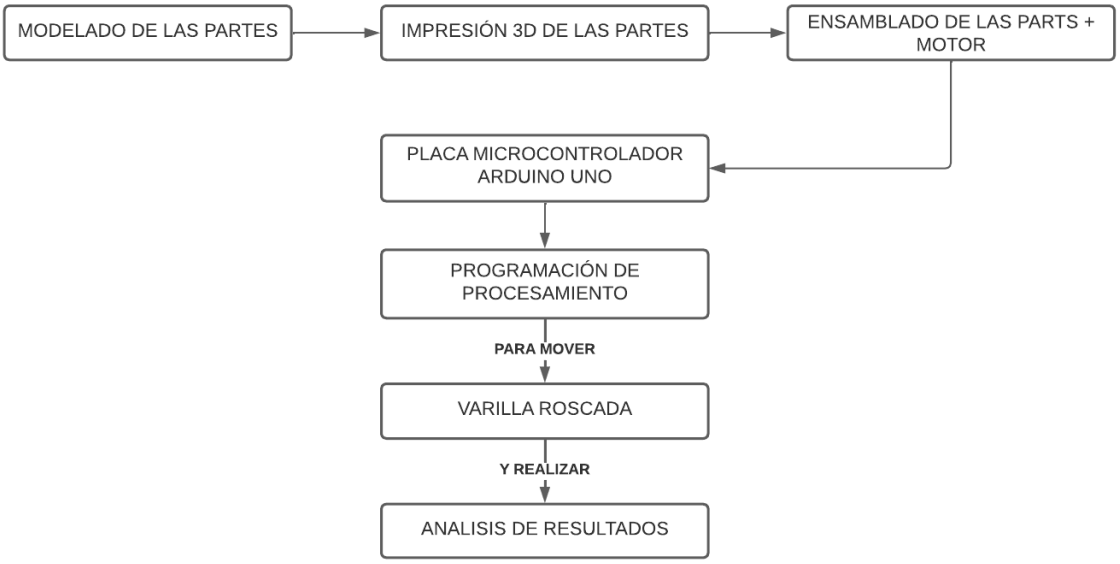


Fotografías

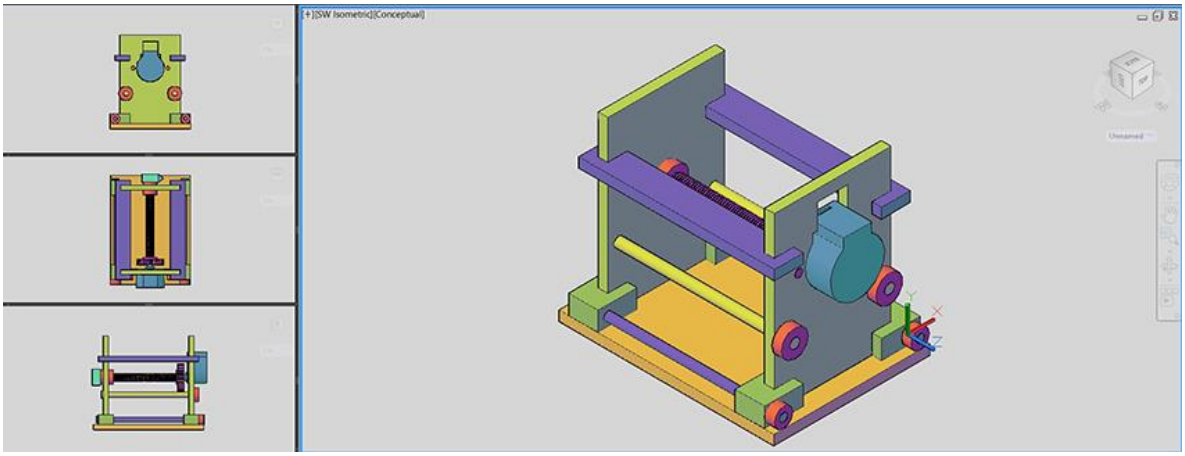
En el video que se presentó se observó que el Pantógrafo XYZ funcionó manualmente correctamente.

Pero para poder desarrollar el pantógrafo final, se corrigió el boceto y se realizaron varias pruebas para analizar el mecanismo que tendrá el prototipo final..

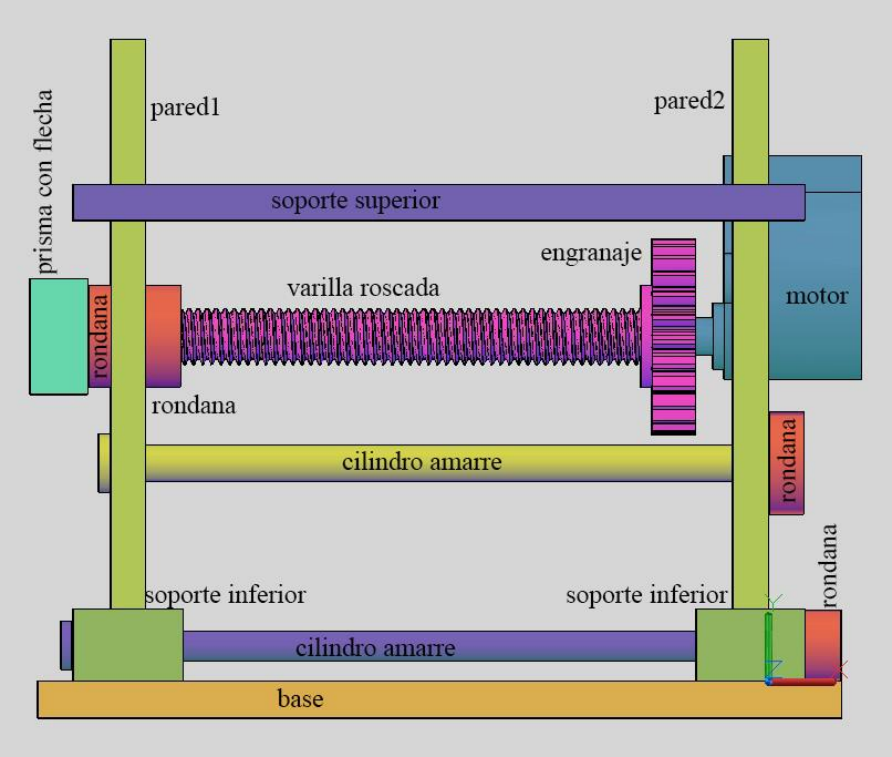
DESARROLLO DE LA PRIMERA PRUEBA



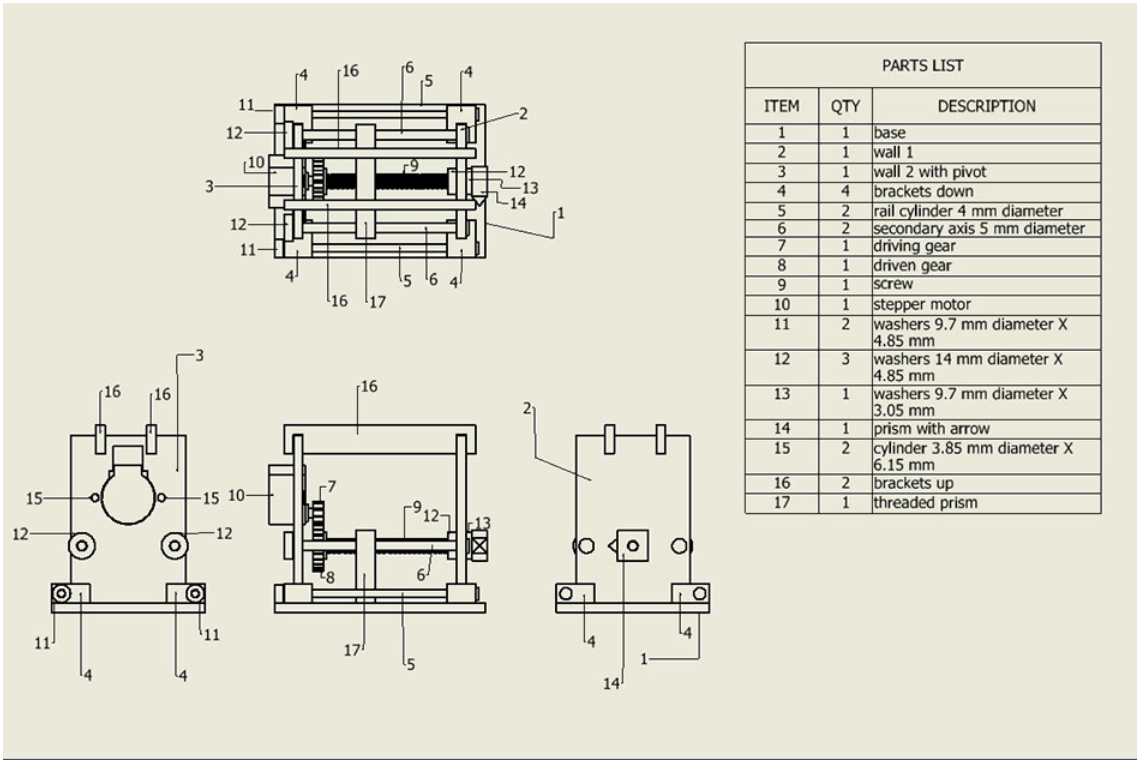
Metodología.



Modelado de las partes del primer modelo

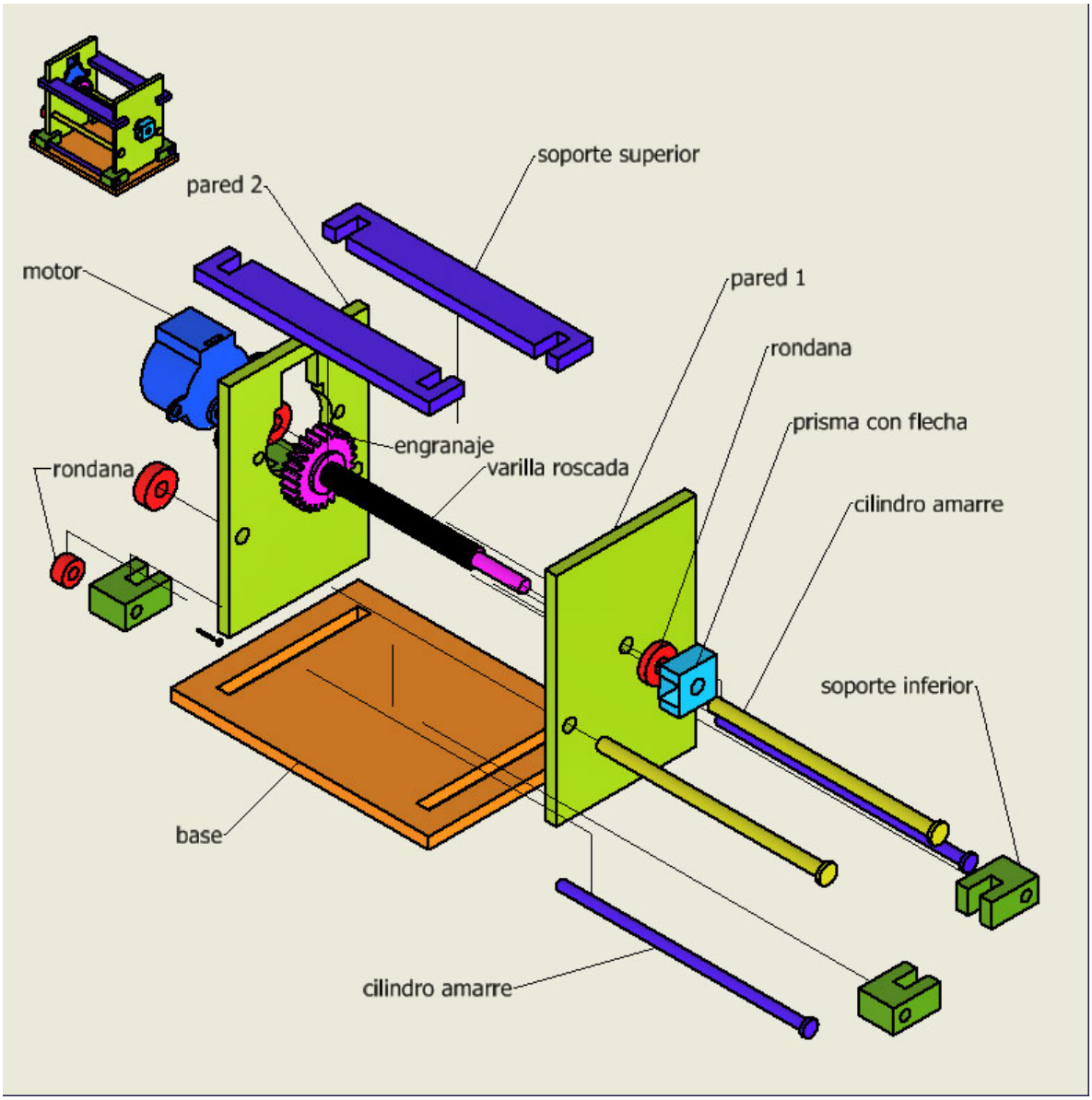


Vista lateral izquierda a mayor escala con los nombres de cada una de las partes del primer modelo.

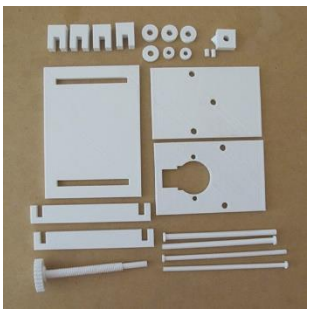


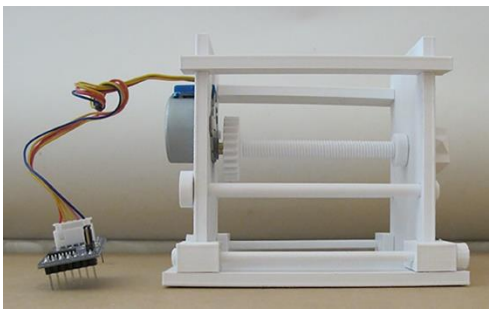
PARTS LIST		
ITEM	QTY	DESCRIPTION
1	1	base
2	1	wall 1
3	1	wall 2 with pivot
4	4	brackets down
5	2	rail cylinder 4 mm diameter
6	2	secondary axis 5 mm diameter
7	1	driving gear
8	1	driven gear
9	1	screw
10	1	stepper motor
11	2	washers 9.7 mm diameter X 4.85 mm
12	3	washers 14 mm diameter X 4.85 mm
13	1	washers 9.7 mm diameter X 3.05 mm
14	1	prism with arrow
15	2	cylinder 3.85 mm diameter X 6.15 mm
16	2	brackets up
17	1	threaded prism

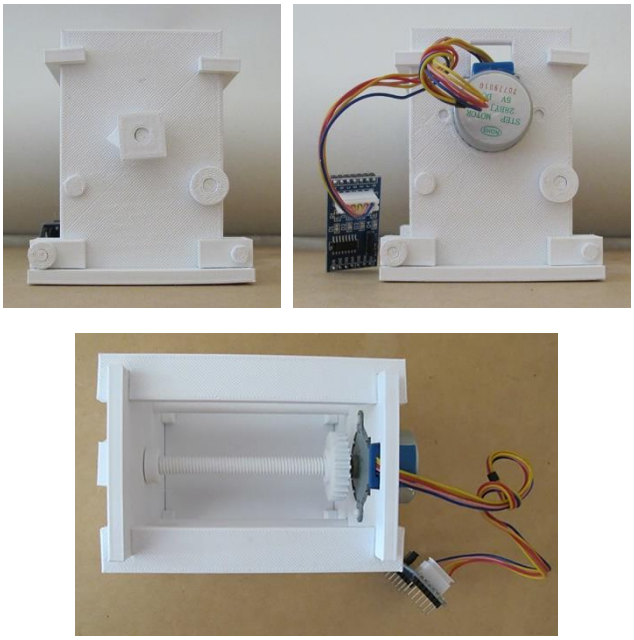
Partes del primer modelo.



Explosivo del primer modelo.







Impresión 3D del modelo.

El modelo consta de 23 partes: base, dos paredes, varilla roscada con engranaje, 6 rondanas, 4 cilindros de amarre, 4 soportes inferiores, dos soportes superiores, un prisma con flecha y dos cilindros para sujetar al motor.

Todo el modelo se imprimió en la impresora MakerBot Replicator 2 usando el material PLA (ácido poliláctico). El modelo fue modelado en el programa AutoCAD™ y en el programa Fusion 360™ (la varilla con rosca y el engranaje).

La varilla roscada que se modeló es la M8x1.25 (M = ISO métrica; 8mm = el principal diámetro del hilo; 1mm = es el tono y 0.25mm = es la duración del cambio).

Usamos un motor paso a paso 28BYJ-48 con un controlador ULN2003 y Arduino UNO.

Un motor paso a paso es un dispositivo electromecánico que convierte pulsos eléctricos en movimientos mecánicos discretos. El eje de un motor paso a paso gira en incrementos discretos cuando los impulsos de mando eléctrico se aplican a él en la secuencia correcta.

La secuencia de los pulsos aplicados se relaciona directamente con la dirección de rotación de los ejes del motor. La velocidad de rotación de los ejes del motor está directamente relacionada con la frecuencia de los pulsos de entrada y la duración de la rotación está directamente relacionada con el número de pulsos de entrada aplicada.

Una de las ventajas más importantes de un motor paso a paso es su capacidad para ser controlado con precisión un sistema de lazo abierto. Control de lazo abierto significa que ninguna información de retroalimentación de posición es necesario. Este tipo de control elimina la necesidad de costosos dispositivos de detección y regeneración como codificadores ópticos.

Los parámetros del motor paso a paso 28BYJ-48 son:

- Tensión nominal de entre 5V y 12V
- 4 fases
- Resistencia 50 Ω
- Par motor de 34 Newton/metro más o menos 0.34 kg por cm
- Consumo de unos 55 mA
- 64 pasos por vuelta (con medios pasos)
- Reducto de 1/64
- Ángulo de pasos: 5.625°
- Frecuencia: 100Hz
- Resistencia de la C.C: 50 Ω+7% (25°C)
- En tracción par > 34.3 mN.m (120Hz)
- Posicionamiento automático par > 34.3 mN.m
- Par de fricción: 600-1200 gf.cm
- Resistencia de aislamiento > 10MΩ (500V)
- Aislante de electricidad: 600VAC/1mA/1s
- Grado de aislamiento: A
- Subida de temperatura < 40K (120 Hz)
- Ruido < 35 db (120Hz, No carga, 10cm)

El motor 28BYJ-48 tiene un paso de 5.625 grados (64 pasos por vuelta usando half-step). El reductor interno tiene una relación de 1/64. Combinados, la precisión total de 4096 pasos por vuelta, equivalente a un paso de 0.088°, que es una precisión muy elevada.

El controlador para el motor bipolar 28BYJ-48 tiene un tamaño de 42mm x 30 mm; un chip controlador de uso ULN2003, 500mA; Leds A, B, C, y D que indican las cuatro fases de las condiciones de trabajo del motor paso a paso; conector blanco estándar del motor paso a paso y pines de alimentación separados.

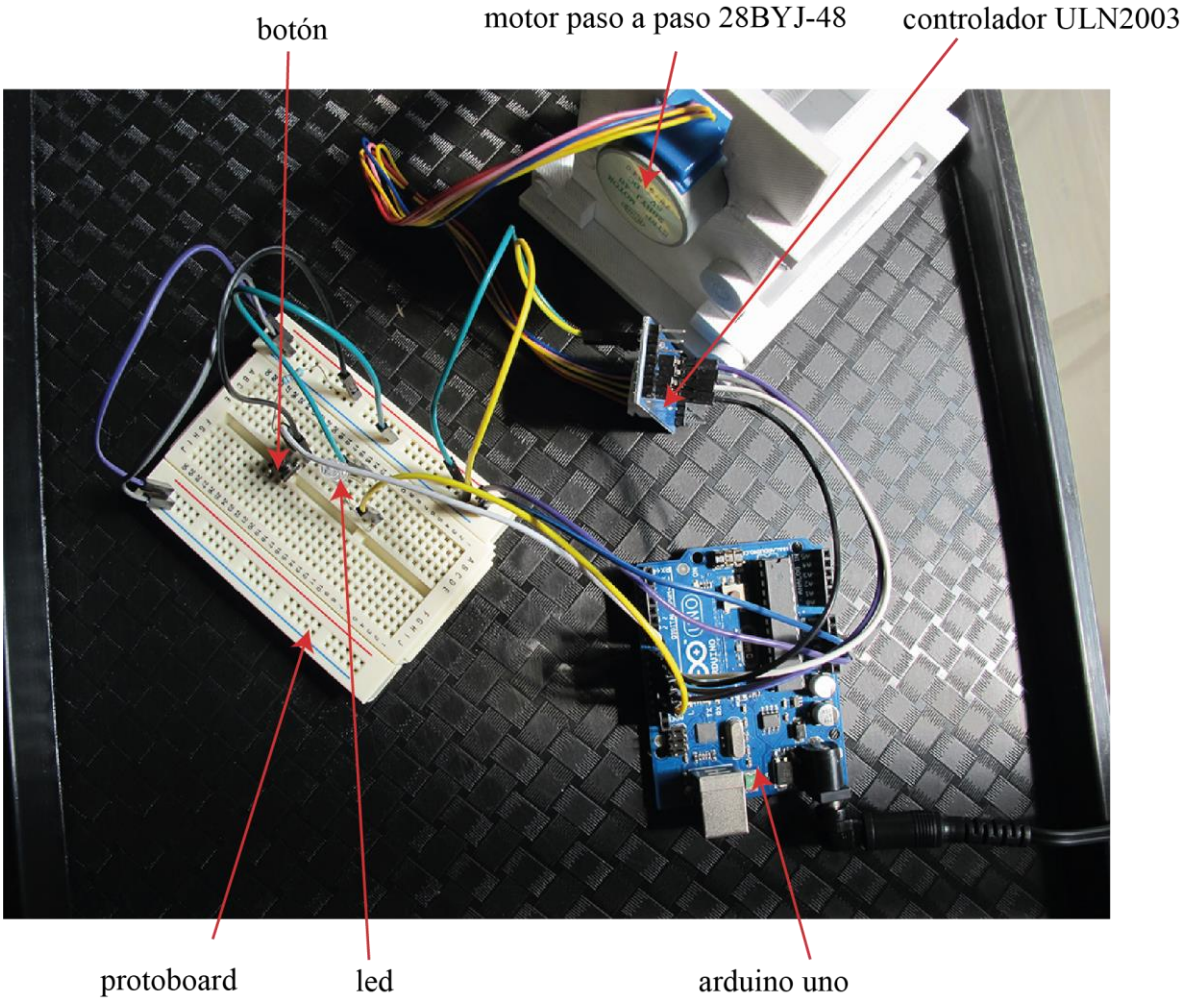
La secuencia de señal de control de 4 pasos es de 11.25 grados por paso, 32 pasos por una revolución de la parte interna del eje del motor. Su relación de transmisión es de 64. Dando un total de 64 x 32 = 2048 pasos.

Para programar el motor 28BYJ-48 utilizamos la librería Stepper que viene incluida con el entorno de desarrollo oficial Arduino. Esta librería nos facilitó el uso de este tipo de motores ya que viene adaptada para motores bipolares. En este caso la librería stepper usa el modo de paso completo en lugar del medio paso, el número de pasos por vuelta es 2048.

Paso	Bobina 1	Bobina 2	Bobina 3	Bobina 4
1	ON	OFF	OFF	OFF
2	OFF	ON	OFF	OFF
3	OFF	OFF	ON	OFF
4	OFF	OFF	OFF	ON

https://aprendiendoarduino.wordpress.com/tag/driver-stepper/

ESQUEMA DE CONEXIÓN EN ARDUINO UNO



El led se utiliza para comprobar que el motor enciende y, para apagar el motor y el led se pulsa el botón.

PROGRAMACIÓN DE PROCESAMIENTO EN ARDUINO.

```
#include <Stepper.h>
int pasosMotor1=1024;
int distancia=pasosMotor1; // una vuelta del engranaje conductor
int input;
int dato=0;
int boton1 = 7;
const int ledPin = 13;
Stepper motor1(pasosMotor1,8,10,9,11);

void setup() {
```

```
 pinMode(ledPin,OUTPUT);
  pinMode(boton1,OUTPUT);
  Serial.begin(9600);
}

//protocolo
 //[MotorEncendido]
 //[MotorApagado  ]
 //[LedEncendido  ]
 //[LedApagado    ]

void loop() {
   if(Serial.available()){
    dato=Serial.parseInt();
    int loop0();
    int loop1();
    int loop4();
    int loop5();
    if(input == 1){
      loop1();
    }else
     if(input == 0){
        loop0();
    }
    if(dato == 4){
      loop4();
    }
    if(dato == 5){
      loop5();
    }
   }
}
 void loop0(){
      digitalWrite(ledPin, LOW);
      delay(100);
  }
 void loop1(){
      digitalWrite(ledPin, HIGH);
      delay(100);
 }

 void loop4(){
   loop1();
   Serial.print("[MotorEncendido4]");
   Serial.print("[LedEncendido4  ]");
   (motor1, HIGH);
```

```
   for(int h = 0; h<=18; h++){
     motor1.setSpeed(30);
     motor1.step(-distancia*10);
     if(digitalRead(boton1) == true){
      Serial.print("[BotónPresionad]");
      (motor1, LOW);
      delay(100);
      break;
     }
   }
      loop0();
      (motor1, LOW);
      Serial.print("[MotorApagado4  ]");
      Serial.print("[LedApagado4    ]");
      delay(100);
}

 void loop5(){
    loop1();
    Serial.print("[MotorEncendido5]");
    Serial.print("[LedEncendido5  ]");
    (motor1, HIGH);
    for(int i = 0; i<=18; i ++){
     motor1.setSpeed(30);
     motor1.step(distancia*10);
     if(digitalRead(boton1) == true){
      Serial.print("[BotónPresionad]");
      (motor1, LOW);
      delay(100);
      break;
     }
    }
     loop0();
      (motor1, LOW);
      Serial.print("[MotorApagado4  ]");
      Serial.print("[LedApagado4    ]");
      delay(100);
}
```

RESULTADOS:

Una vez que se realizaron las conexiones entre el modelo y Arduino UNO, se escribió la programación. En ella se puede observar que se definieron: la librería de stepper; el paso del motor a 1024; las variables de distancia, de input y de datos. Y los pines en donde se conectaron el botón y el led.

En el void setup() se definieron que el led y el botón son salidas y la conexión entre el puerto serial para comunicar a Arduino con la computadora.

En el void loop() además de escribir que la serie está disponible y que la variable dato va a comprobar la cadena se tienen una serie de variables que se programaron independientemente para llamarlas en los loop 4 y loop 5. En estos loops se definió por medio del for el número de veces que se repetirá la distancia*10 con la velocidad del motor. La distancia es dada por el paso de la varilla roscada, que en este caso es de 0.25 mm. Al ser una distancia que no se puede medir fácilmente se multiplicó por 10 para que se pueda medir quedando de 2.5 mm.
La letra "h" es la variable que indica el número de pasos del motor paso a paso, en este caso serían 19 pasos.

```
for(int h = 0; h<=18; h++){
    motor1.setSpeed(30);
    motor1.step(-distancia*10);
```

En el input están el loop 0 y el loop1 que encienden y apagan el led.
Al realizar las pruebas se verificó que el sistema funcionó bien, pero va muy rápido por lo que nos dimos la tarea de desarrollar una segunda prueba para reducir la velocidad por medio de un sistema de engranajes.

SE ANEXA VIDEO "videoPrimeraPrueba"

DESARROLLO DE LA SEGUNDA PRUEBA

Los engranajes son componentes fundamentales en un amplio número de mecanismos de control del movimiento, así como en transmisiones mecánicas y electromecánicas.
Tipos de engranajes:

1. Aquellos que operan sobre ejes paralelos de rotación montados en un piñón: engranajes rectos, engranajes helicoidales, engranajes doble helicoidales, epicicloidales y otros.
2. Aquellos que trabajan sobre transmisiones entre ejes perpendiculares (engranajes helicoidales que se apoyan sobre un tornillo sin fin para la transmisión de par).
3. Engranajes cónicos cuya transmisión de movimiento es rotatorio sobre ejes concurrentes.

Relación de transmisión: básicamente se trata de la relación entre las velocidades de rotación de dos engranajes conectados entre sí, donde uno de ellos ejerce fuerza sobre el otro. Esta relación surge de la diferencia de diámetros de ambas ruedas. Denominándose piñón a aquella con un diámetro más reducido. Básicamente, este factor implica una diferencia entre velocidad de rotación de los dos ejes. De esta forma, y teniendo en cuenta el engrane del engranaje y piñón, la relación de transmisión se calcula a partir del número de dientes del engranaje dividido por el número de dientes de su piñón.

Diámetro de paso: determinado a partir del número de dientes y la distancia central a la que operan los engranajes.

Paso base: paso medido sobre la circunferencia base de generación de la envolvente.

Distancia al centro: equivalente la suma del diámetro de paso del piñón y el diámetro de paso del engranaje dividido por dos.

Paso primitivo: distancia circular desde un punto de un diente del engranaje a un punto del siguiente diente, tomado a lo largo del círculo primitivo. Dos engranajes deben tener el mismo circulo primitivo para engranar entre sí.

Paso diametral o módulo: una medida normativa del tamaño de los dientes. Se trata del número de dientes por pulgada del diámetro de paso. El incremento en el tamaño de los dientes reduce el paso diametral. Por lo general, los pasos diametrales fluctúan entre 25 y 1.

Distancia de montaje: es la distancia entre la intersección del eje del engranaje con la línea del ángulo primitivo y un punto de referencia del engranaje. Respetar esta distancia implica asegurar un correcto montaje y uso de los elementos dentados.

Ángulo de perfiles (ángulo de precisión): la pendiente del diente de engranaje en la posición del paso diametral. Si el ángulo de presión es cero, el diente es paralelo al eje de la rueda, lo que le convierte en un engranaje de dientes rectos.

Ángulo de la hélice: representa la inclinación del diente en una dirección longitudinal. Siempre que el ángulo de la hélice sea de cero grados, el diente es paralelo al eje de la rueda, por lo que hablaríamos también de un engranaje de dientes rectos.

Calcular reducción de relación: La relación o radio de reducción o transmisión consiste en la relación entre las velocidades de rotación de dos engranajes conectados entre sí, donde uno de ellos ejerce fuerza sobre el otro.
Esta relación se debe a la diferencia de diámetros de las dos ruedas, que implica una diferencia entre las velocidades de rotación de ambos ejes, lo que se puede verificar mediante el concepto de velocidad angular.
Asimismo, la relación de reducción de una caja de engranajes describe la relación entre las revoluciones por minuto (RPM) del eje de entrada y las RPM del eje de salida, ya sea para multiplicar el par, reducir la velocidad o ambas cosas.

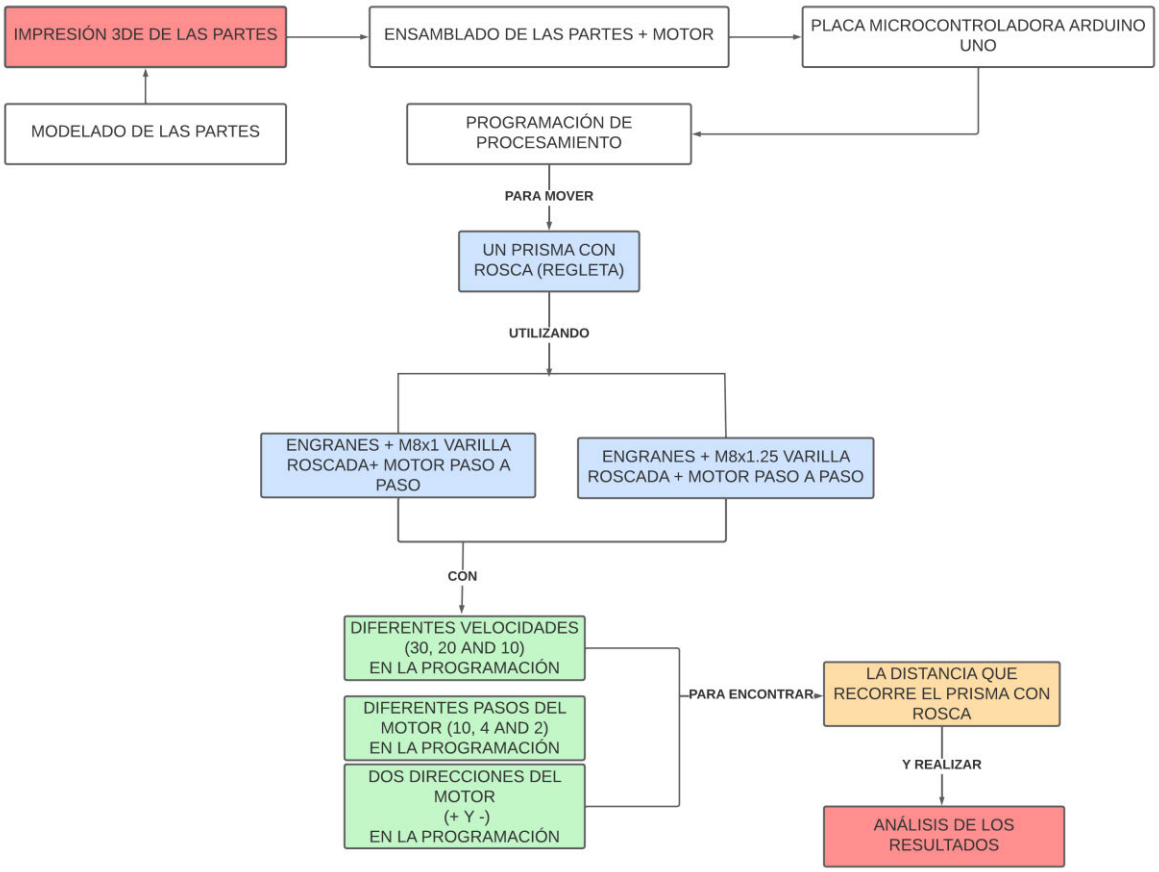
Antes de calcular la velocidad de reducción en el proyecto es fundamental tener claro la velocidad y par de salida que se quiere obtener en el reductor y la velocidad y el par de entrada del que se parte.
El diseño de un reductor será la consecuencia de estudiar todos los puntos anteriores y resolver cuestiones como ¿cuántos engranajes poner para poder conseguir dicha velocidad?, ¿de qué material?, ¿tamaño? O ¿qué motor utilizar para que con la cantidad mínima de engranajes consiga los valores requeridos?
Para encontrar la reducción se utiliza la siguiente fórmula:

$$i = \frac{Ws}{wE} = \frac{Ze}{Zs}$$

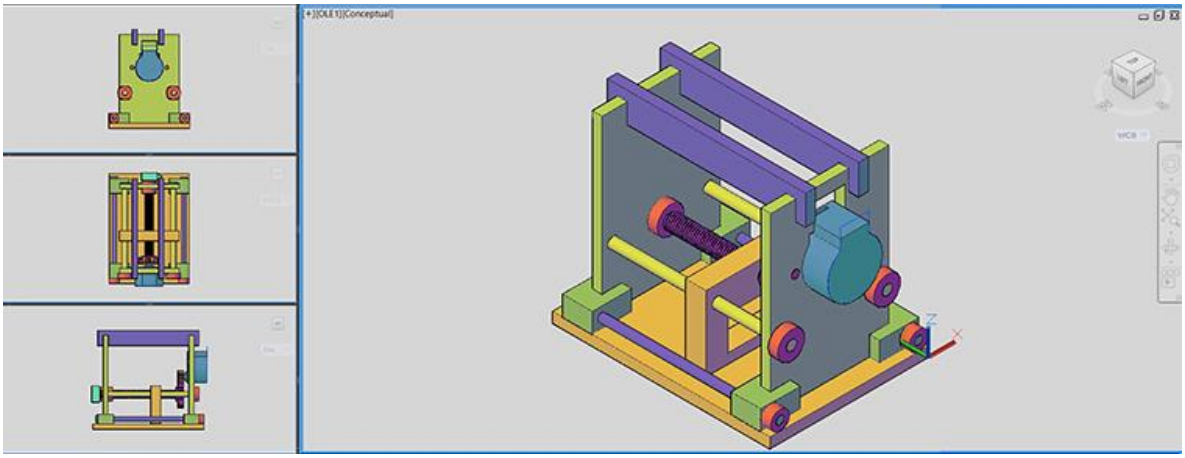
I = relación de transmisión
Ws = velocidad de salida
We = velocidad de entrada
Ze = número de dientes de los conductores
Zs = número de dientes de los conducidos

La reducción es la inversa de la relación de transmisión

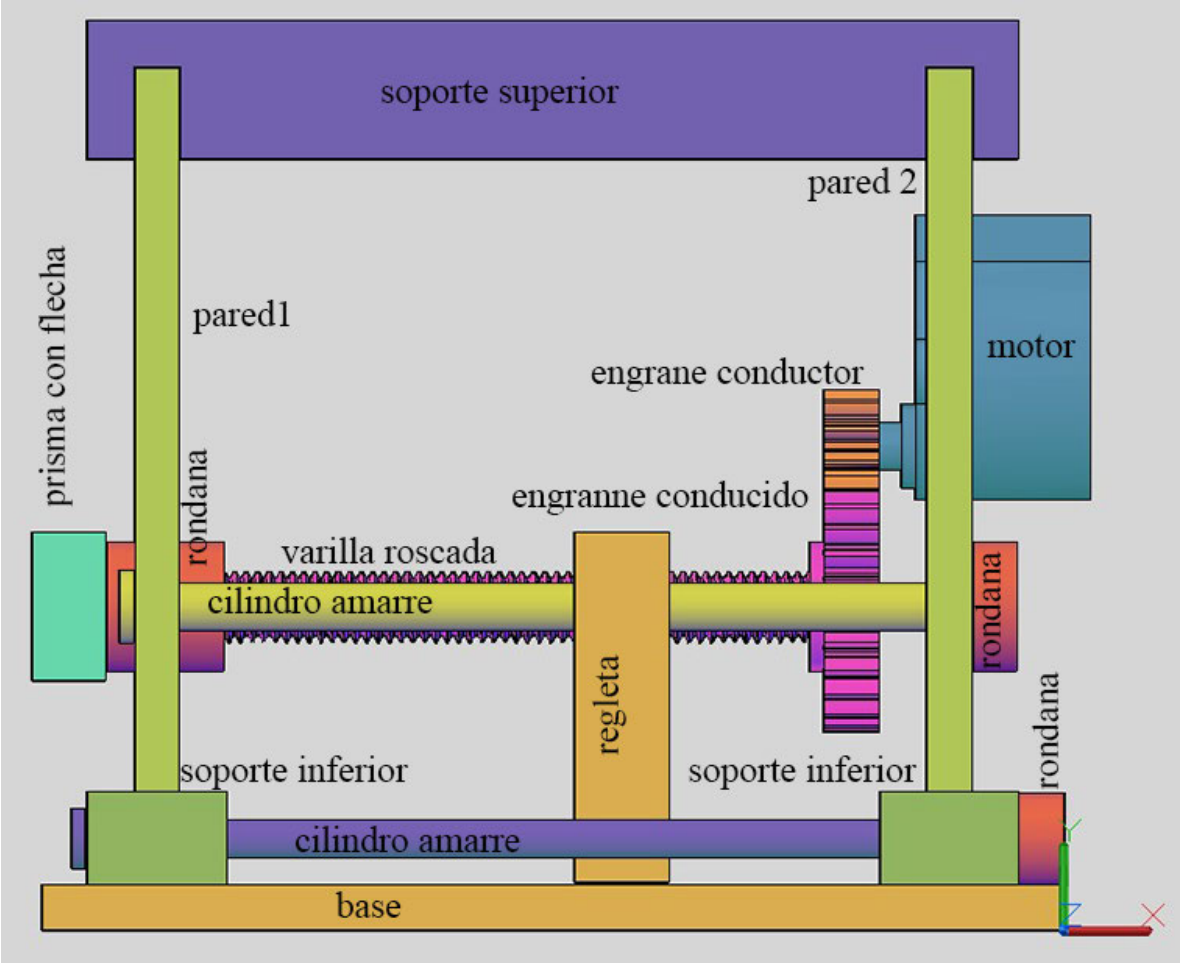
$$reducción = \frac{1}{i}$$



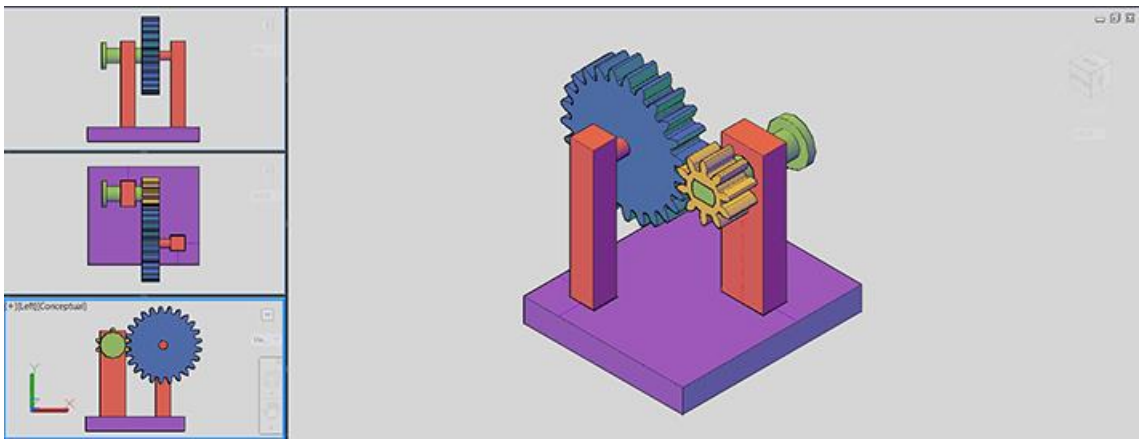
Metodología



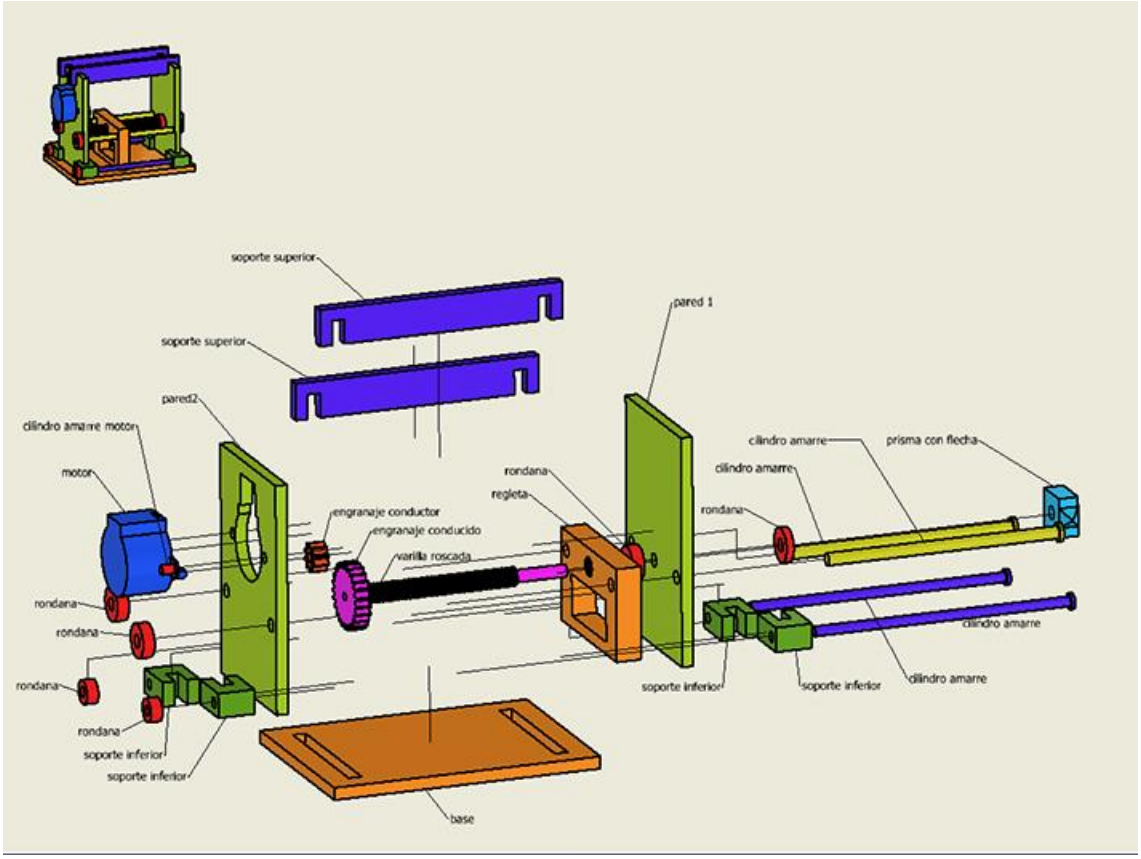
Modelado de las partes del segundo modelo



Vista izquierda a mayor escala con los nombres de cada una de las partes del segundo modelo.



Prueba manual de los engranajes.



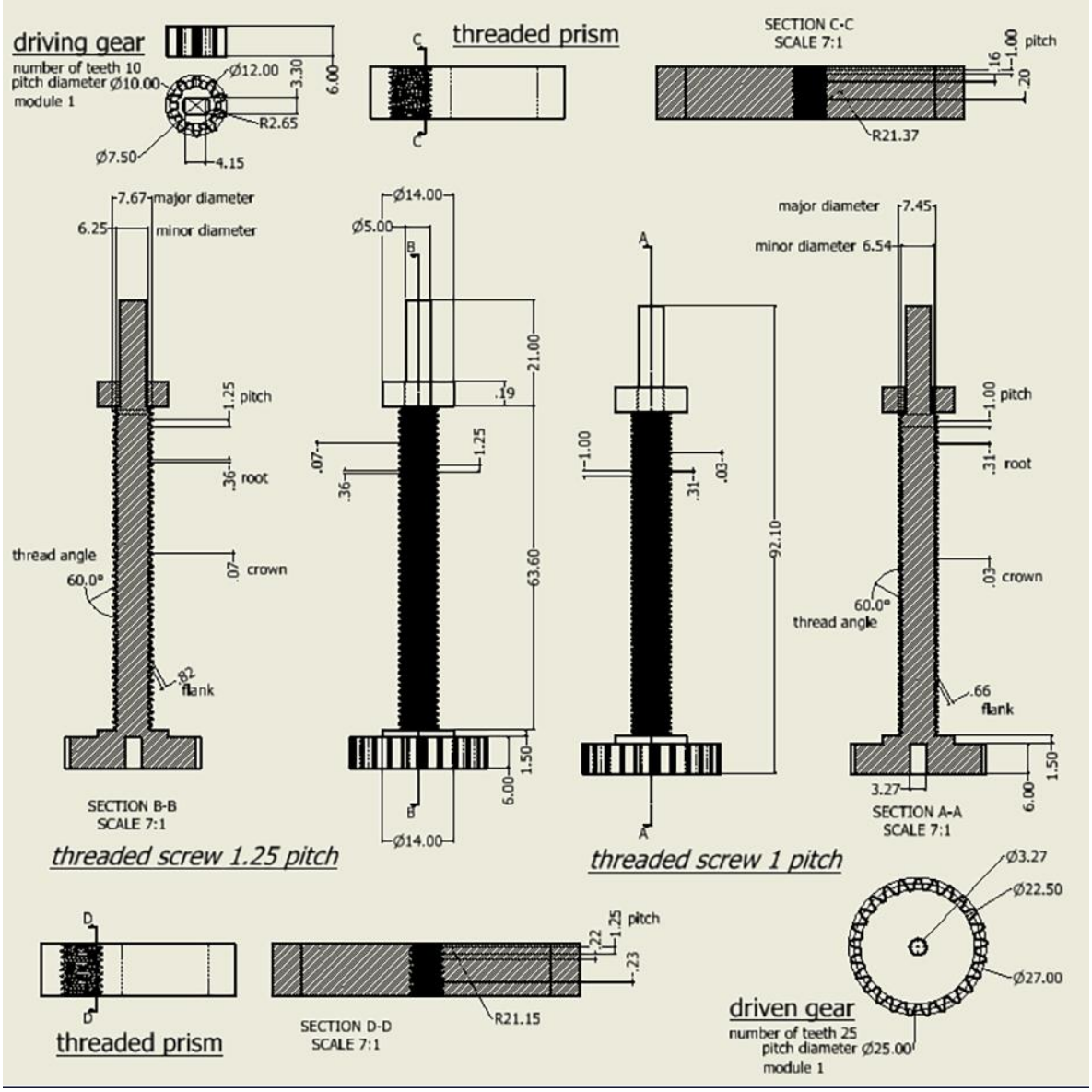
Explosivo segundo modelo.

El modelo consta de 26 partes: base, dos paredes, varilla roscada con el engranaje conducido, engranaje conductor, 6 rondanas, 4 cilindros de amarre, 4 soportes inferiores, dos soportes superiores, un prisma con flecha, una regleta y dos cilindros para sujetar al motor.
Todo el modelo se imprimió en la impresora MakerBot Replicator 2 usando el material PLA (ácido poliláctico). El modelo fue modelado en el programa AutoCAD™ y en el programa Fusion 360™ (la varilla con rosca y el engranaje).

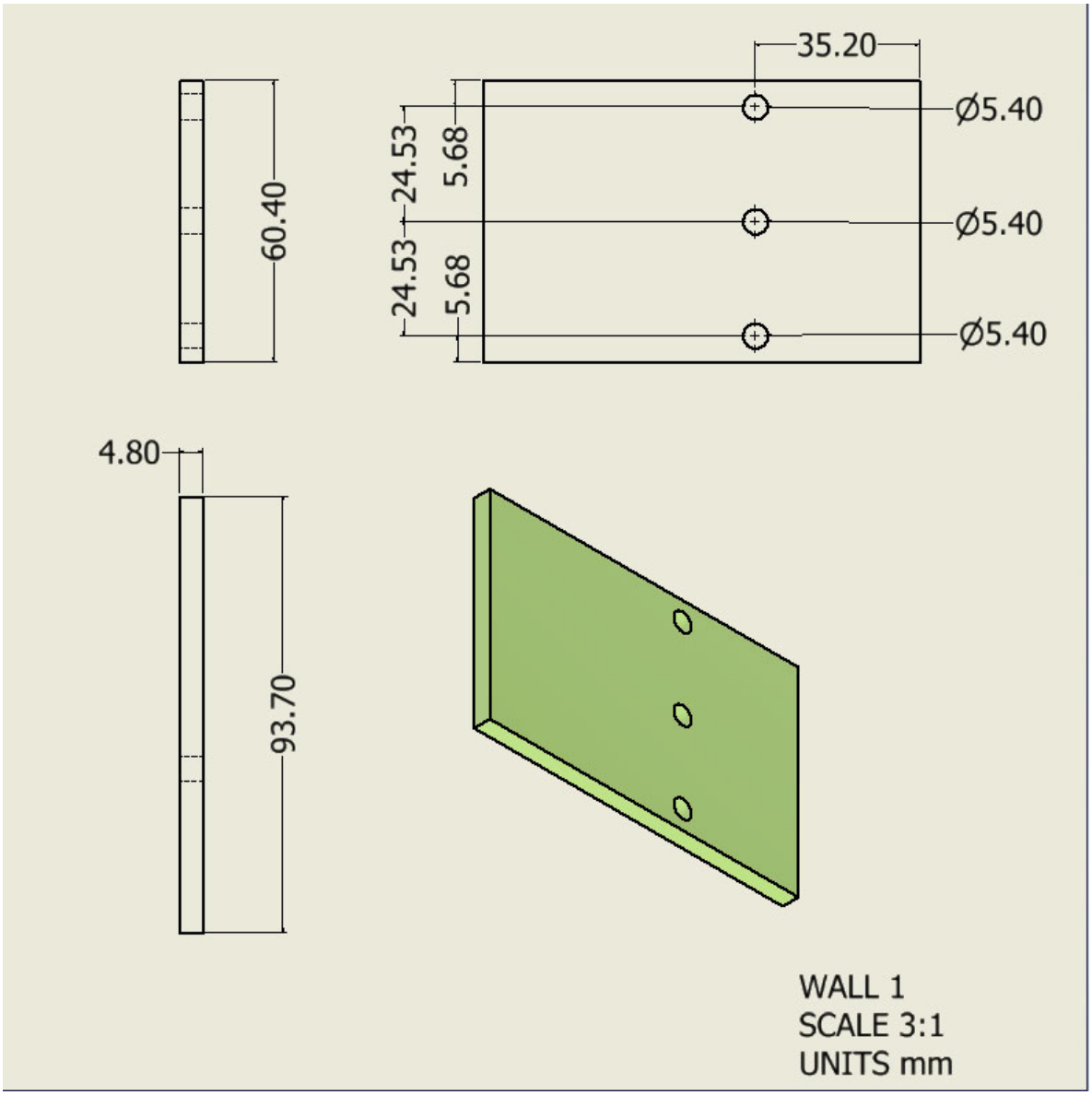
Para realizar el análisis se modelaron dos varillas roscadas. La primera varilla roscada es la M8x1.25 (M = ISO métrica; 8mm = el principal diámetro del hilo; 1mm = es el tono y 0.25mm = es la duración del cambio). La segunda varilla roscada es la M8x1 (M = ISO métrica; 8mm = el principal diámetro del hilo; 1mm = es el tono).

Usamos un motor paso a paso 28BYJ-48 con un controlador ULN2003 y Arduino UNO.

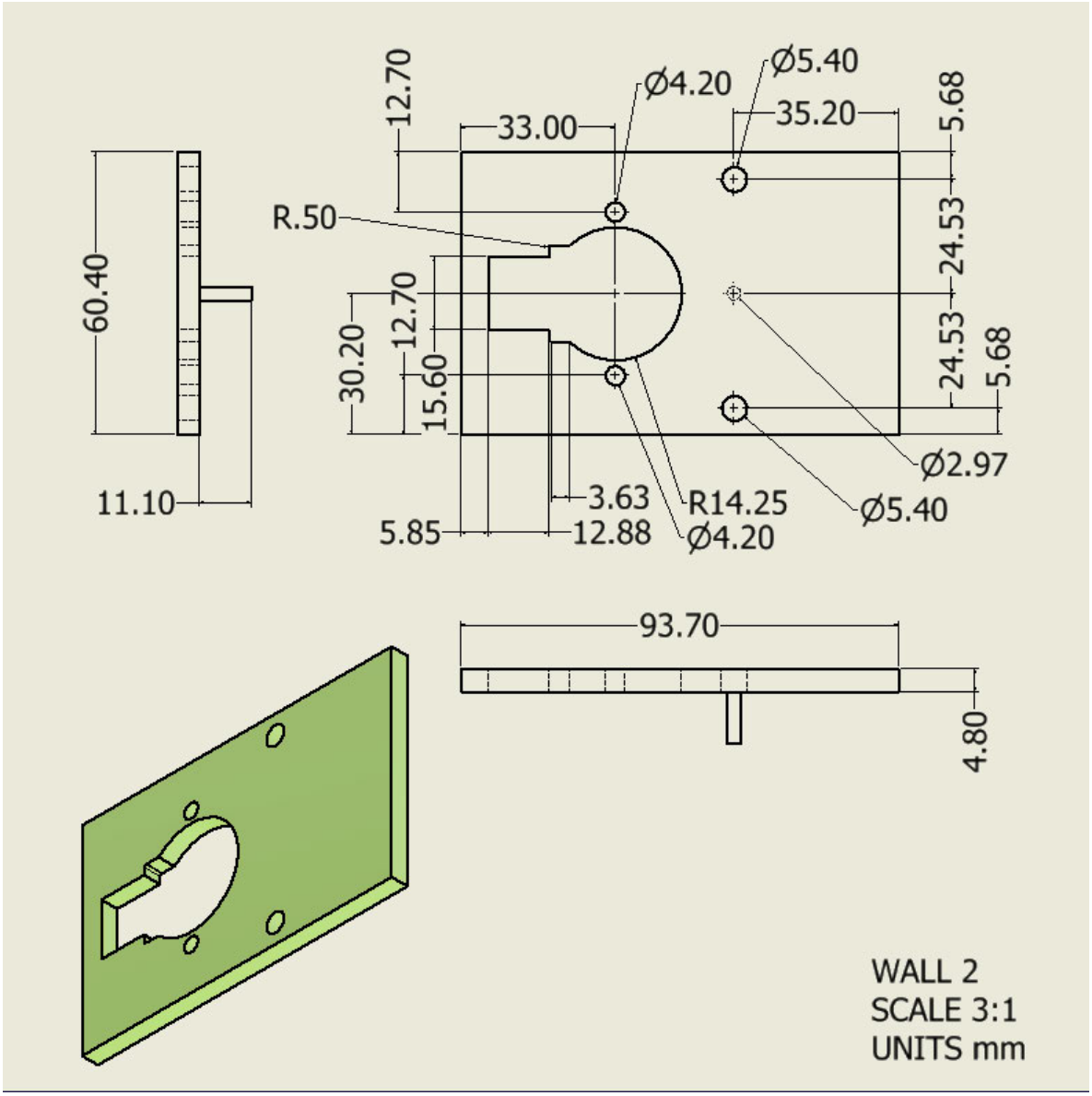
PLANOS CONSTRUCTIVOS DE LAS PIEZAS DEL PRIMER Y SEGUNDO MODELO



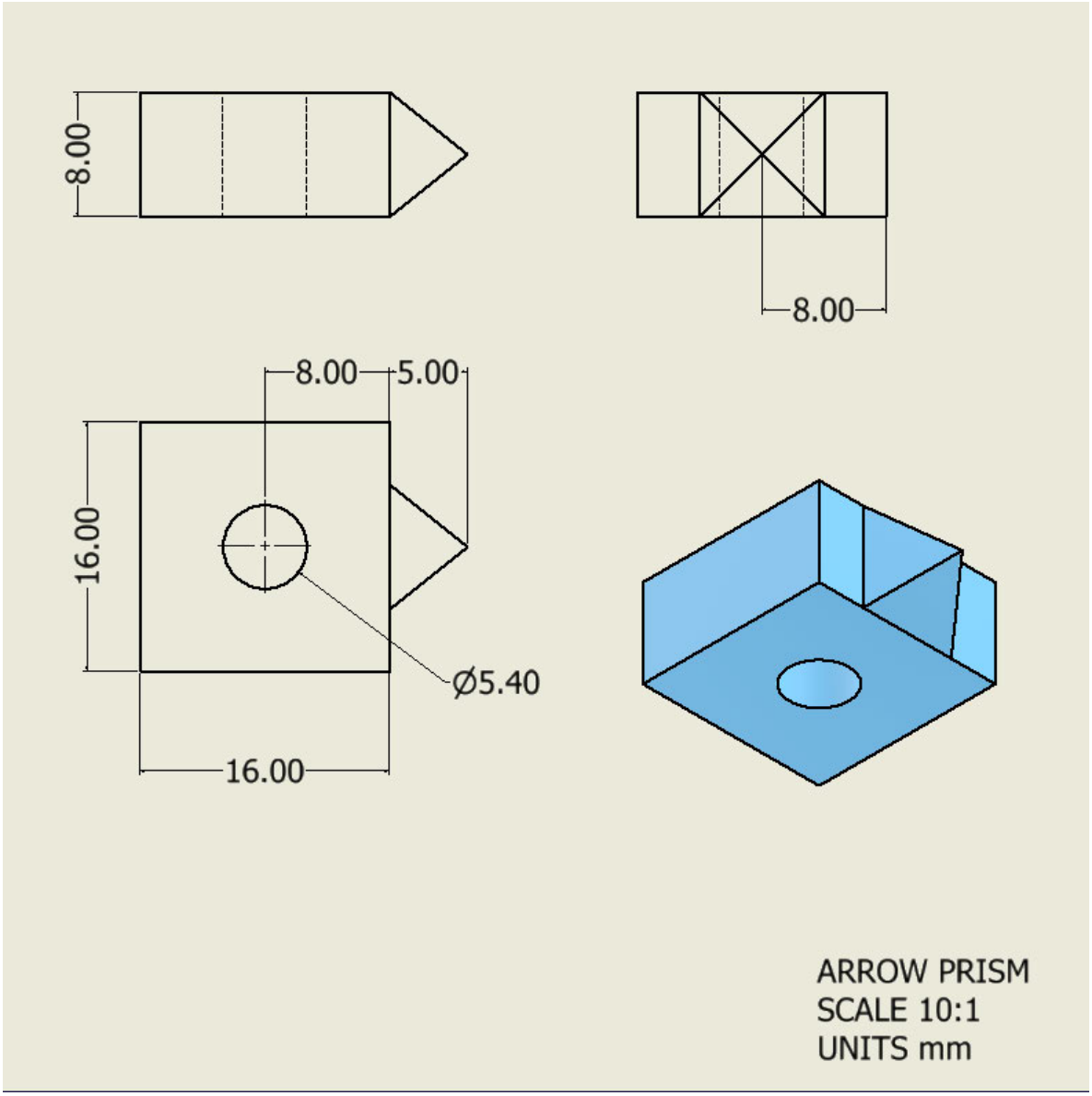
Varillas roscadas, engrane conductor, engrane conducido y regleta.


35.20
Ø5.40
Ø5.40
Ø5.40
24.53
5.68
24.53
5.68
60.40
4.80
93.70
WALL 1
SCALE 3:1
UNITS mm

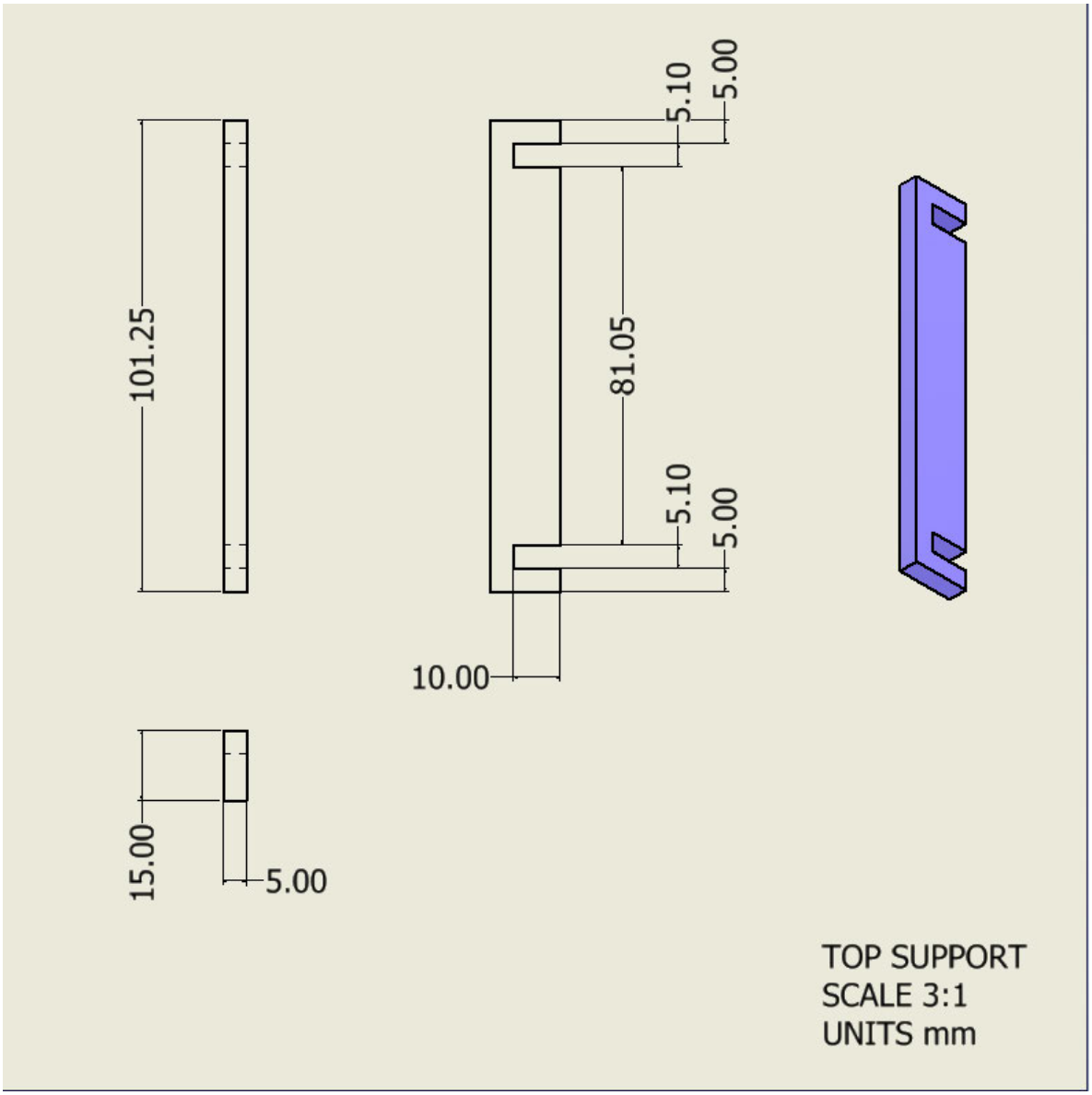
Pared1.


12.70
33.00
Ø4.20
Ø5.40
35.20
5.68
R.50
24.53
60.40
12.70
30.20
15.60
24.53
5.68
Ø2.97
11.10
3.63
R14.25
Ø5.40
5.85
12.88
Ø4.20
93.70
4.80
WALL 2
SCALE 3:1
UNITS mm

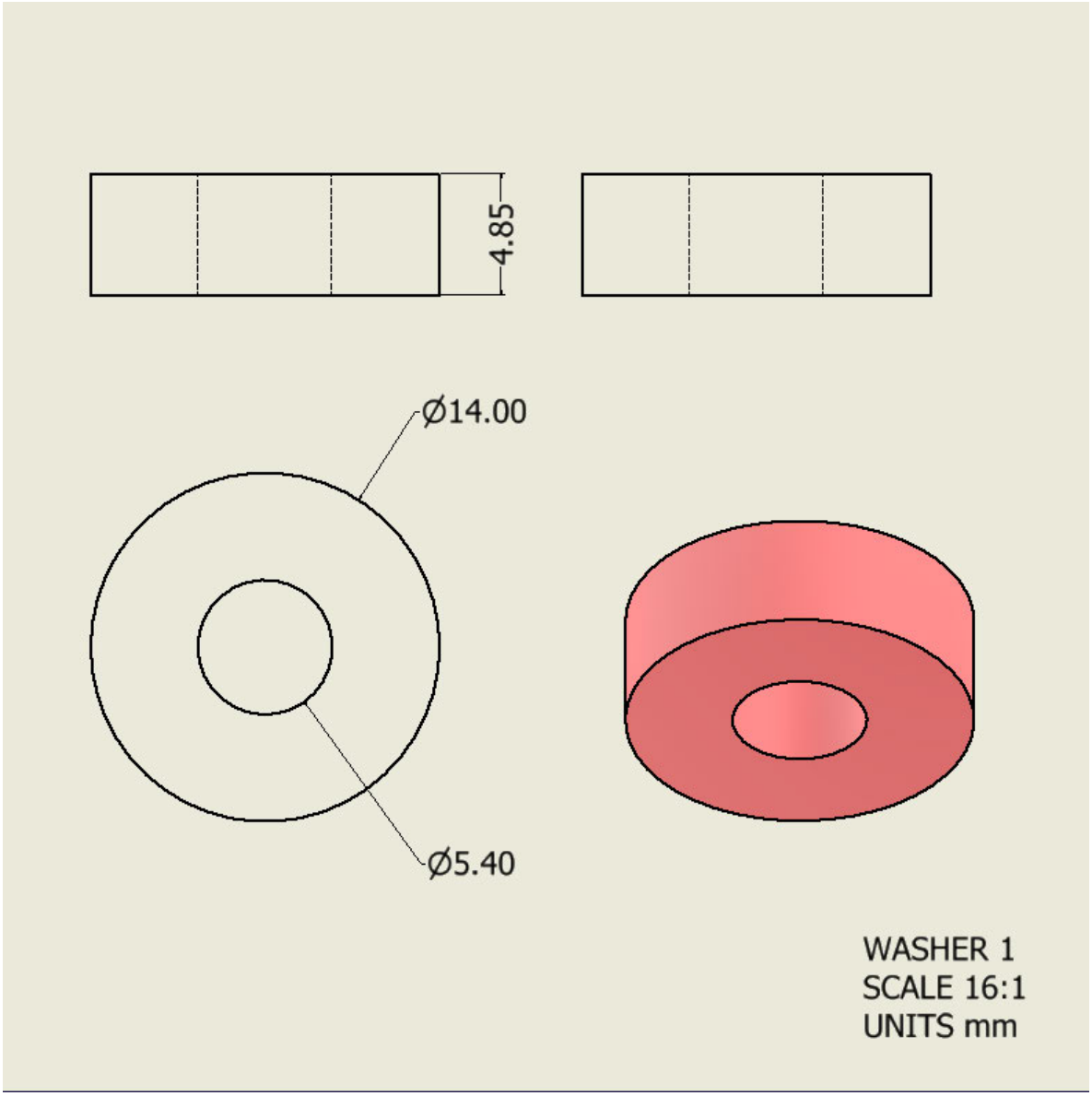
Pared2.



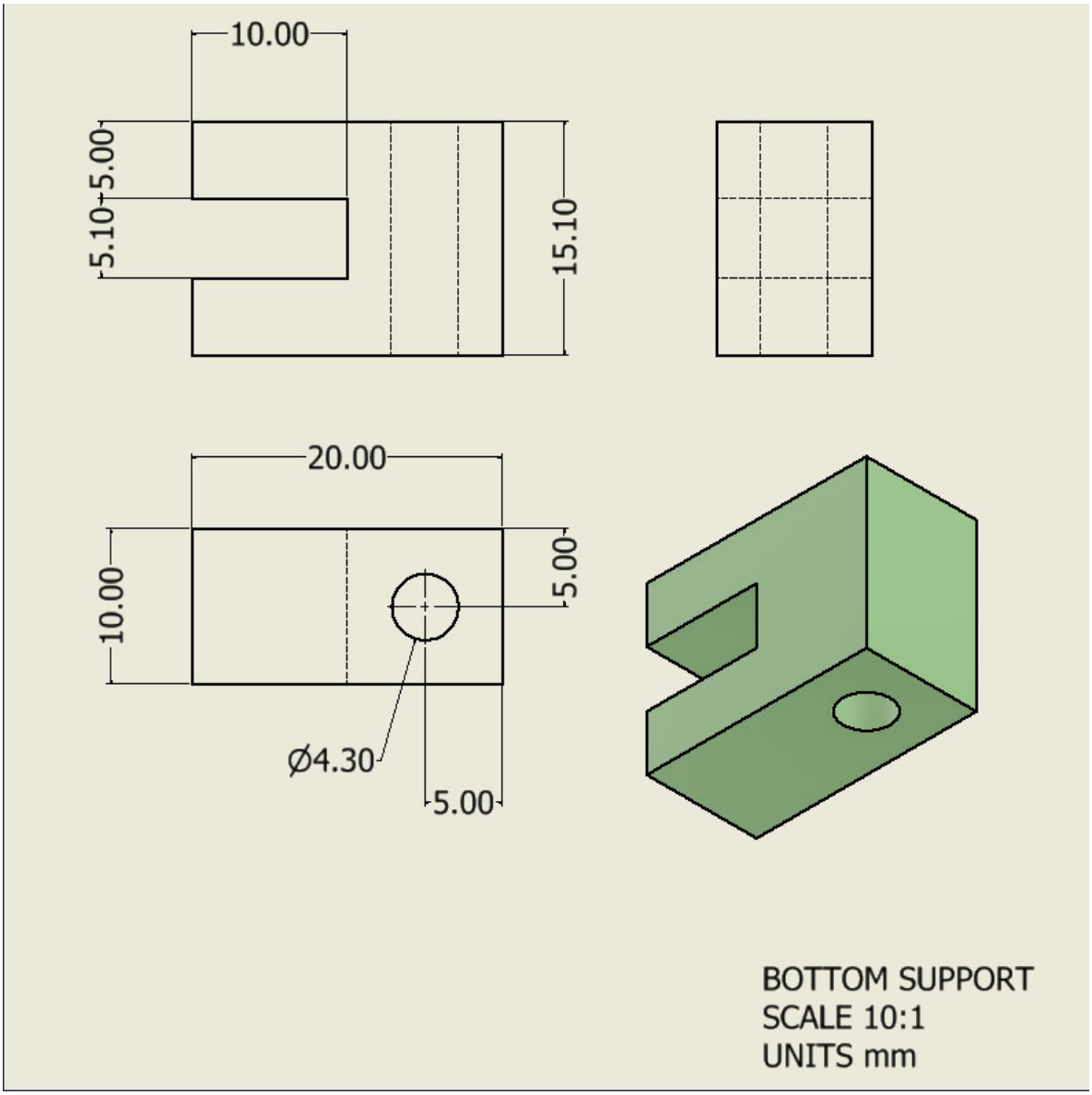
Prisma con flecha.



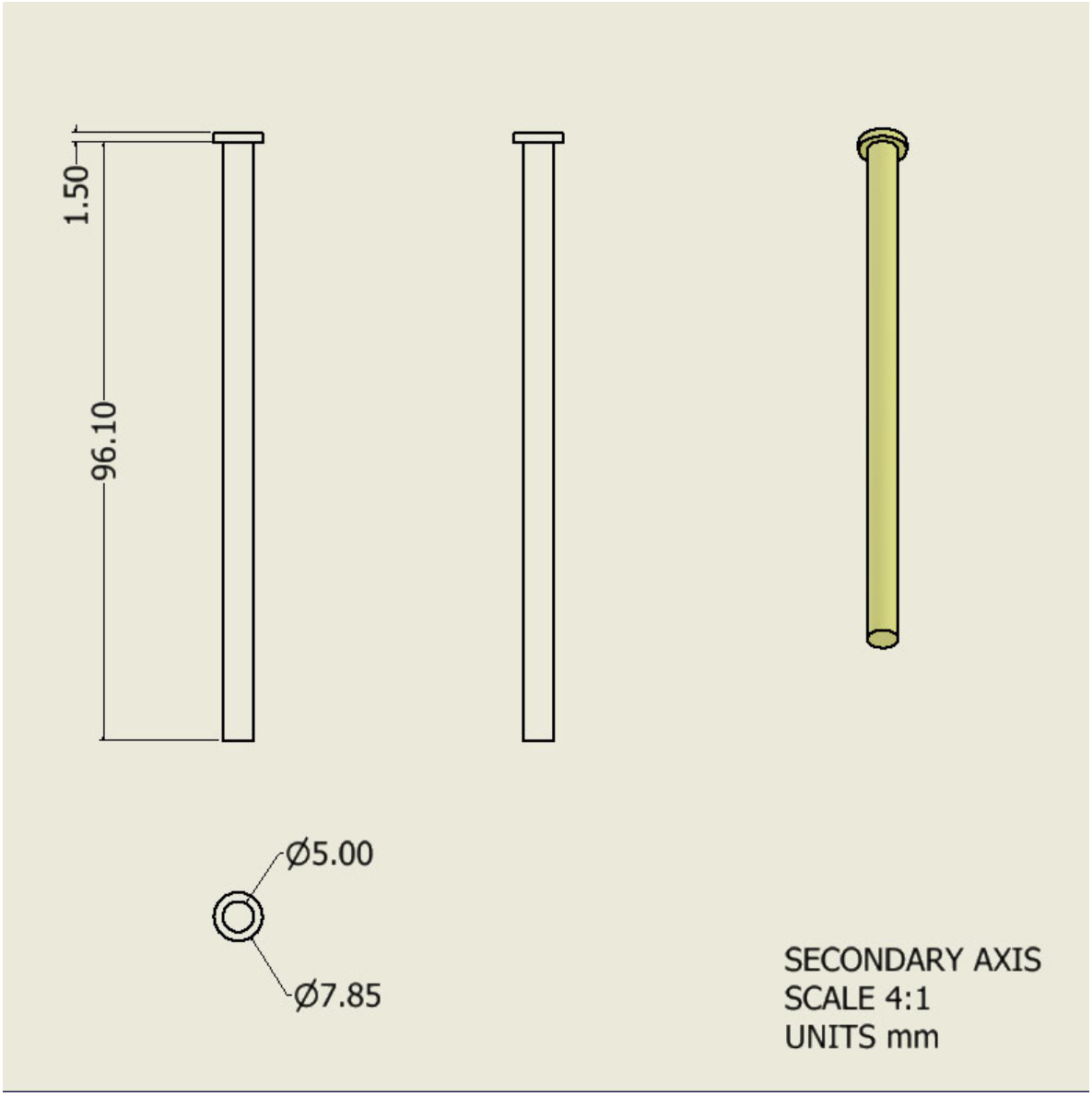
Soporte superior.



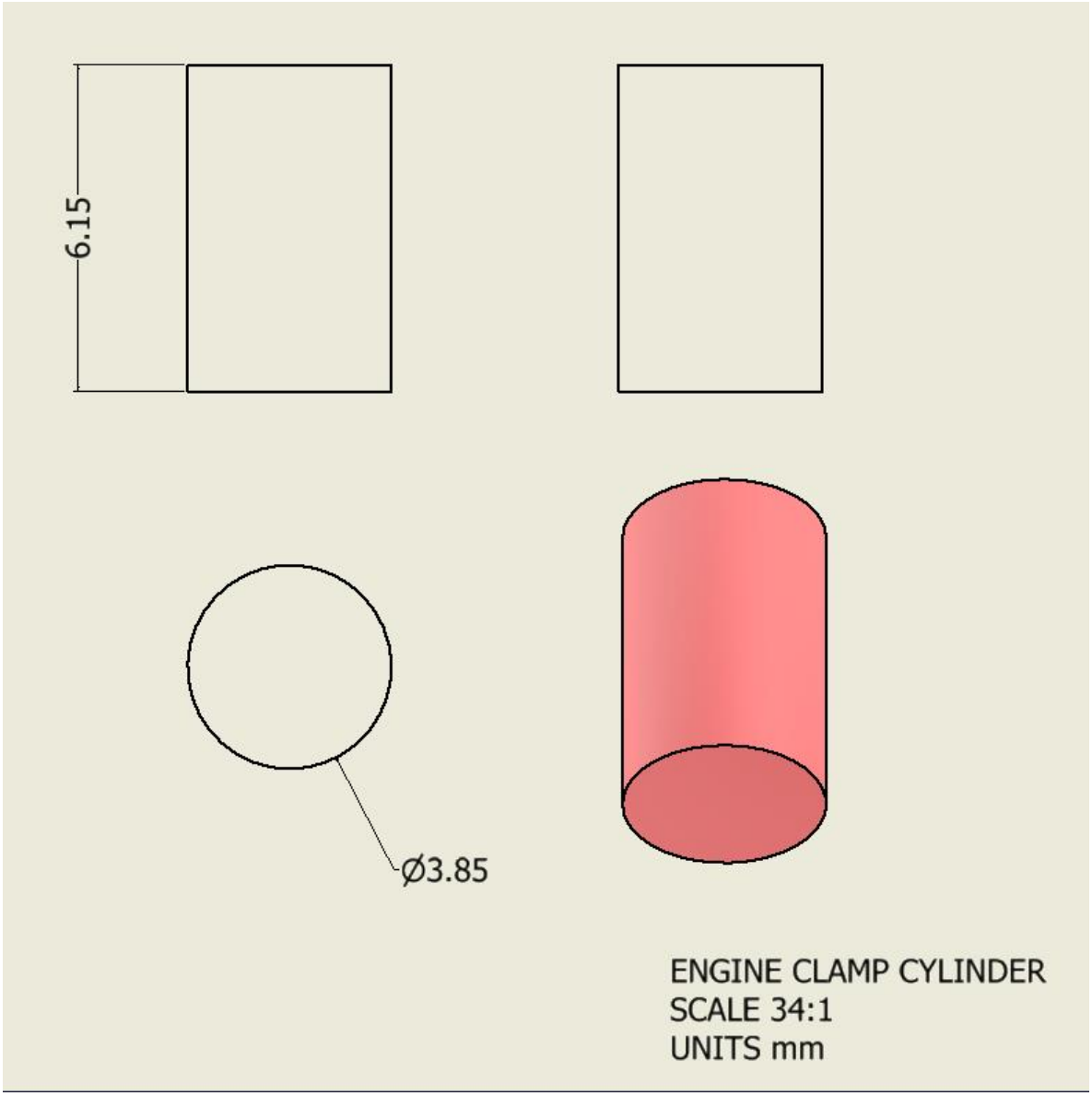
Rondana 1.



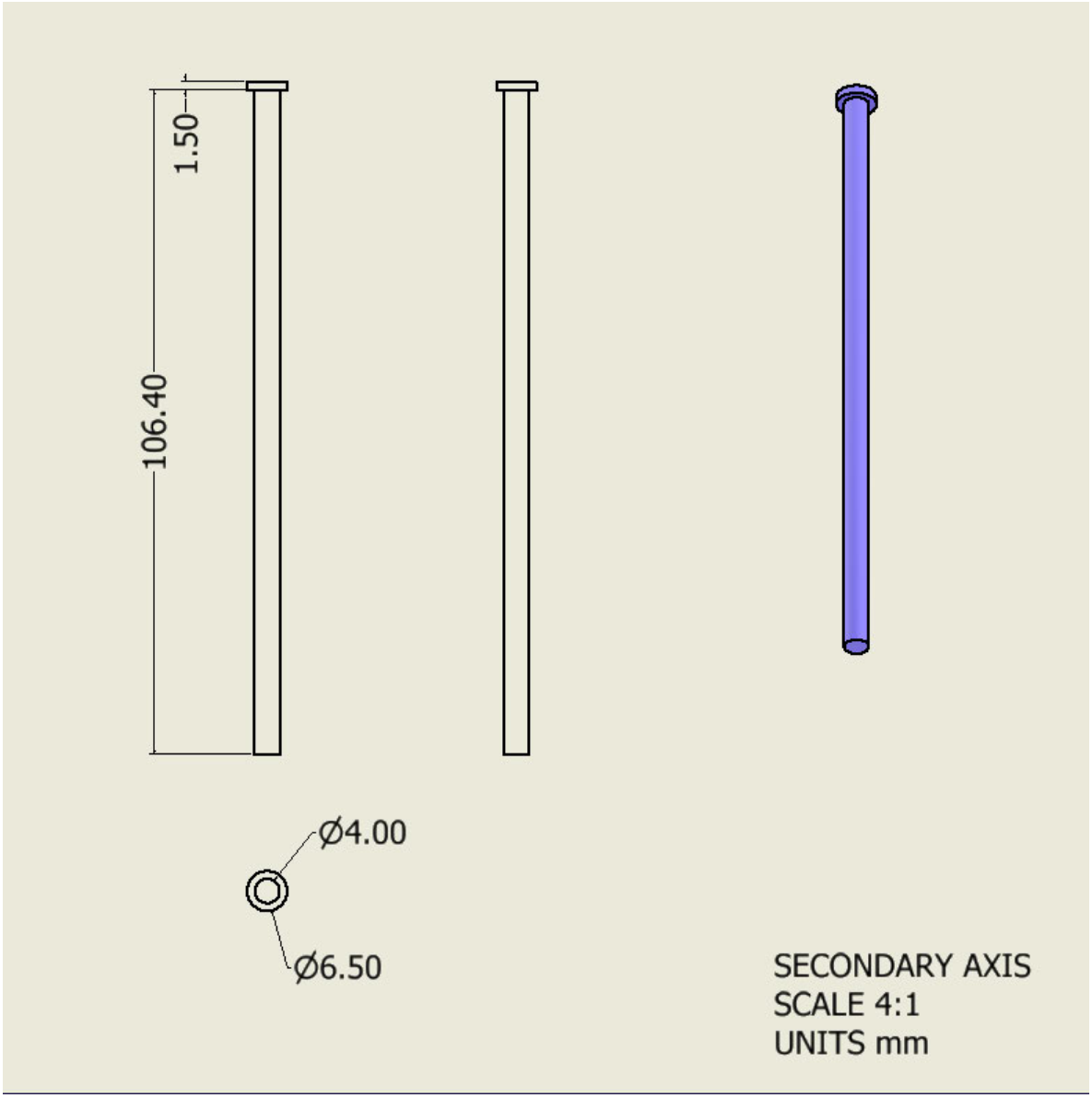
Soporte inferior.



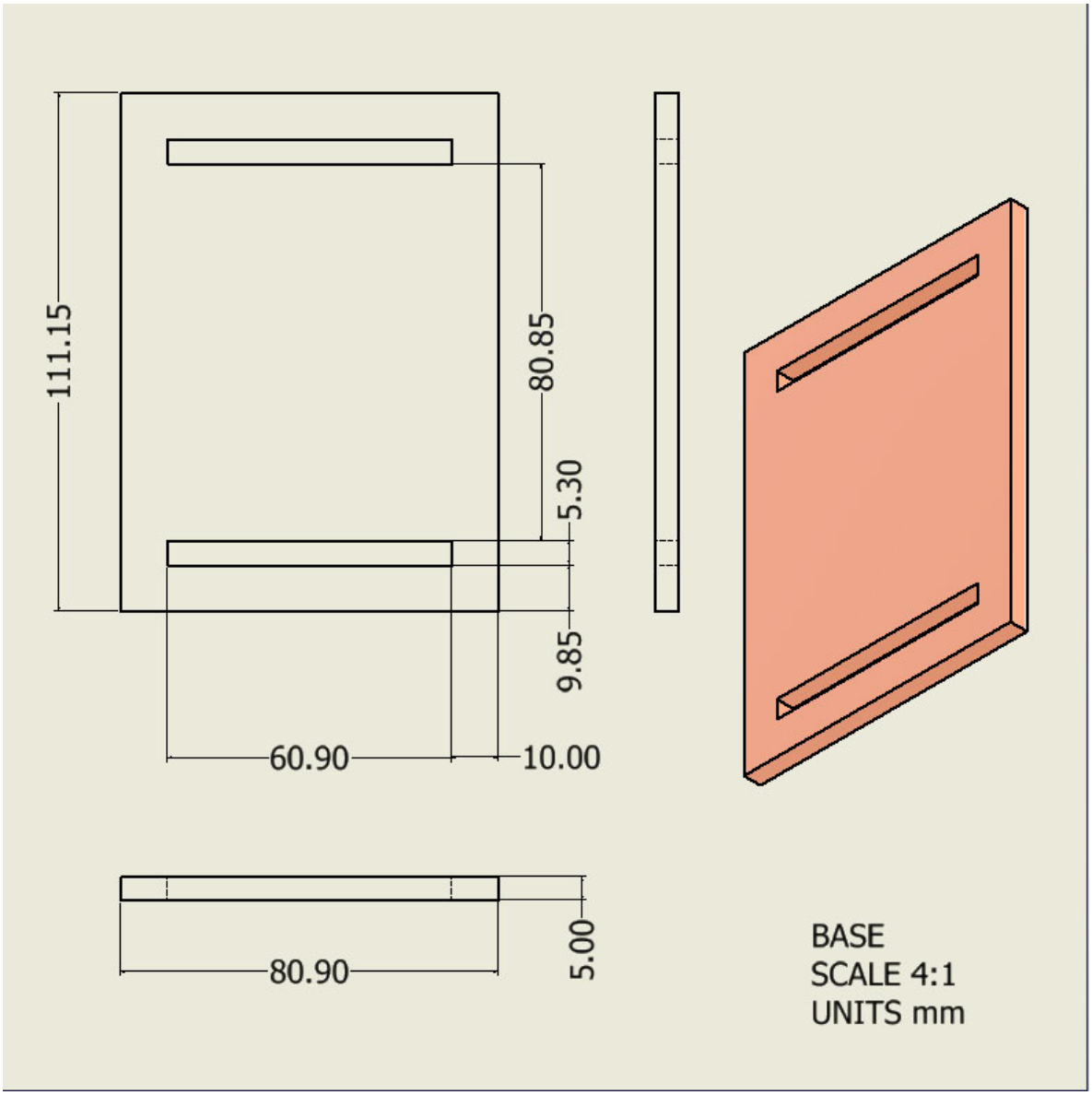
Cilindro amarre regleta, eje secundario.



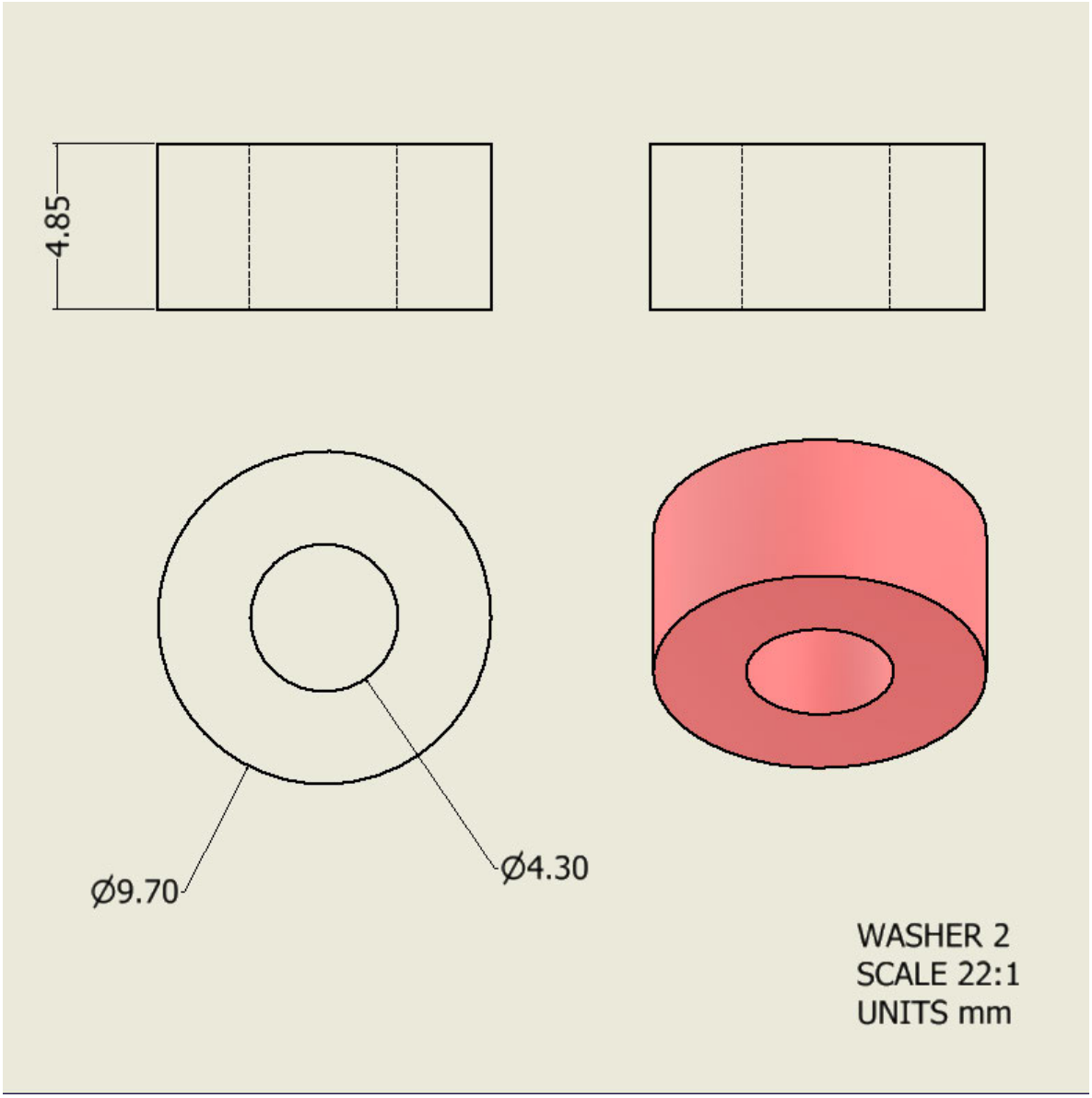
Cilindro de sujeción del motor.



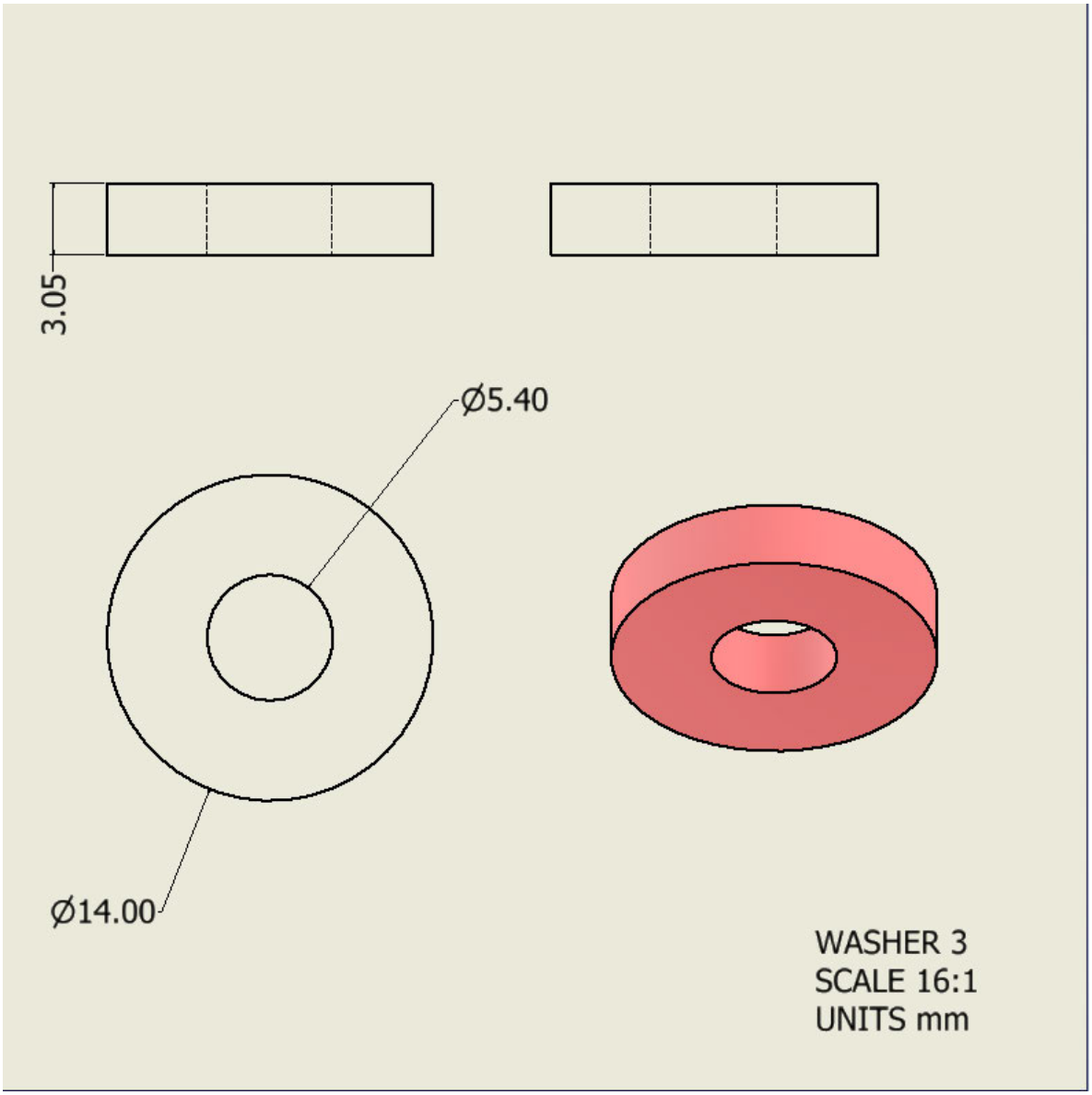
Cilindro amarre soportes inferiores, eje secundario.



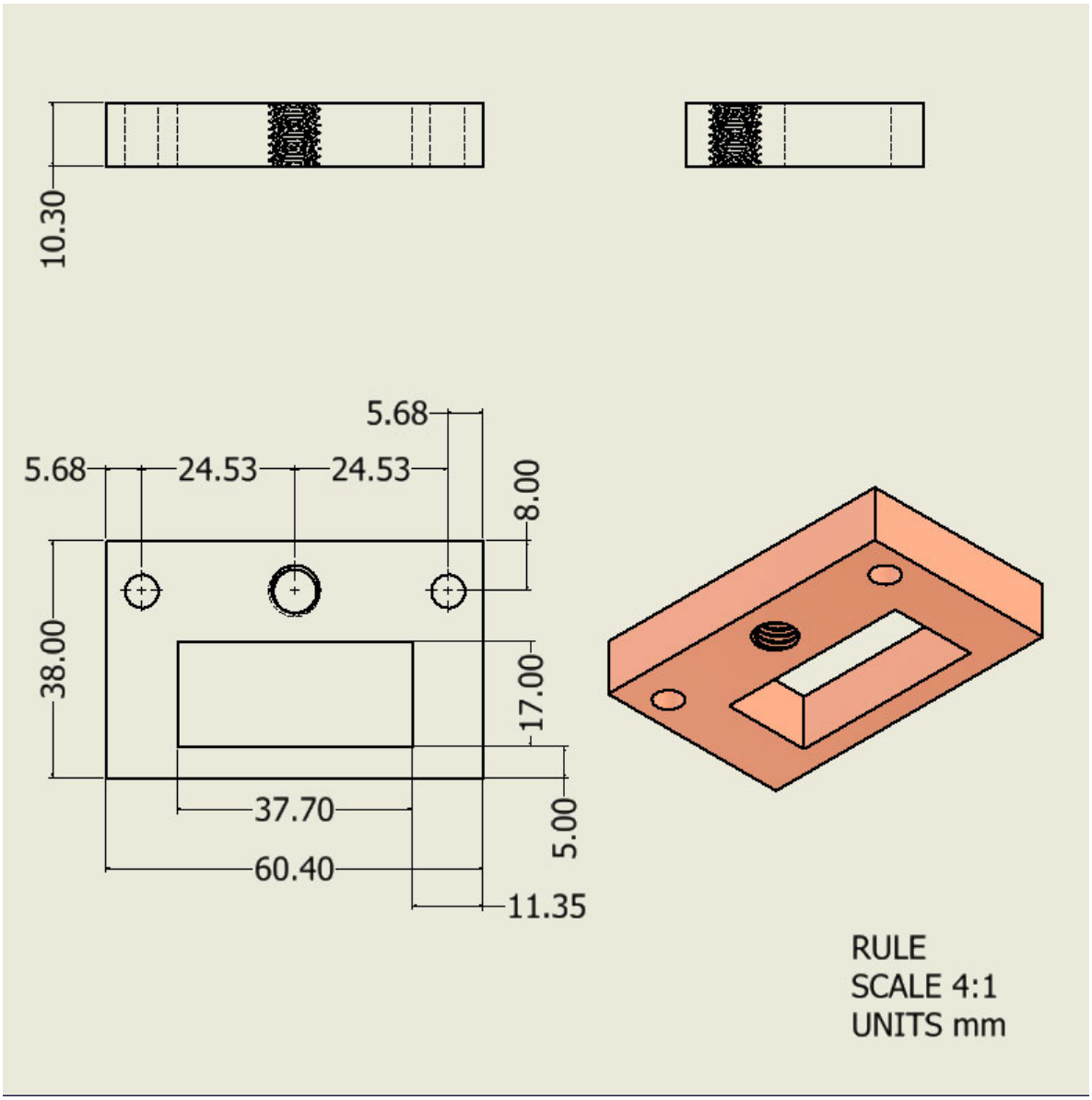
Base.



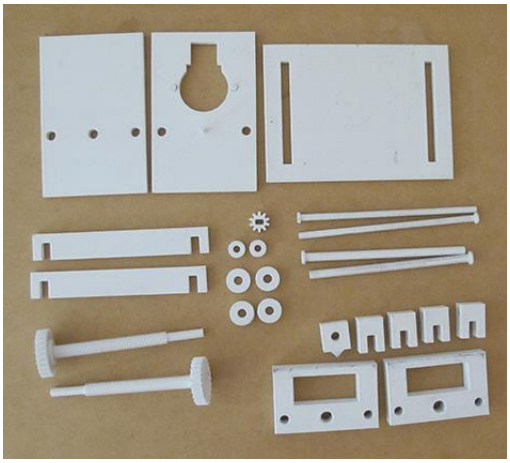
Rondana 2.

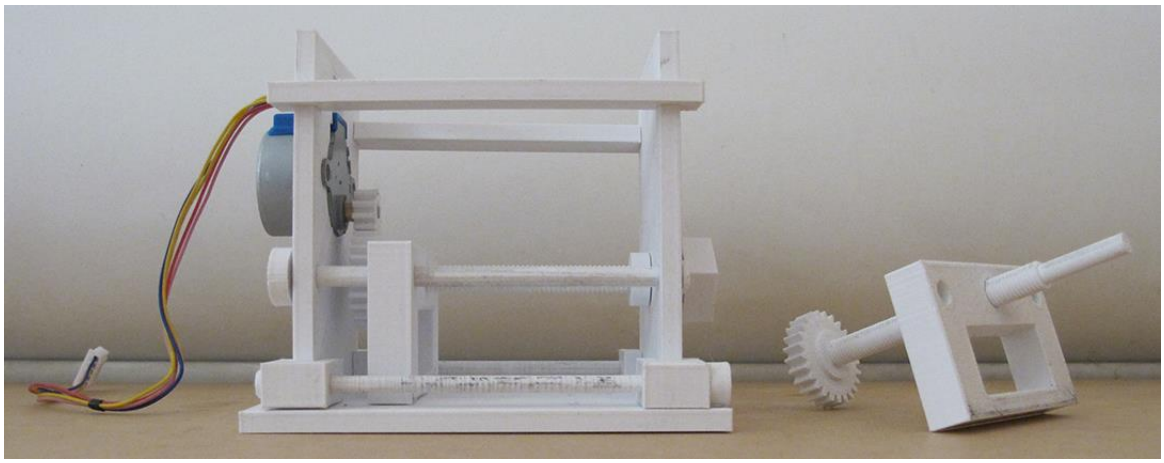


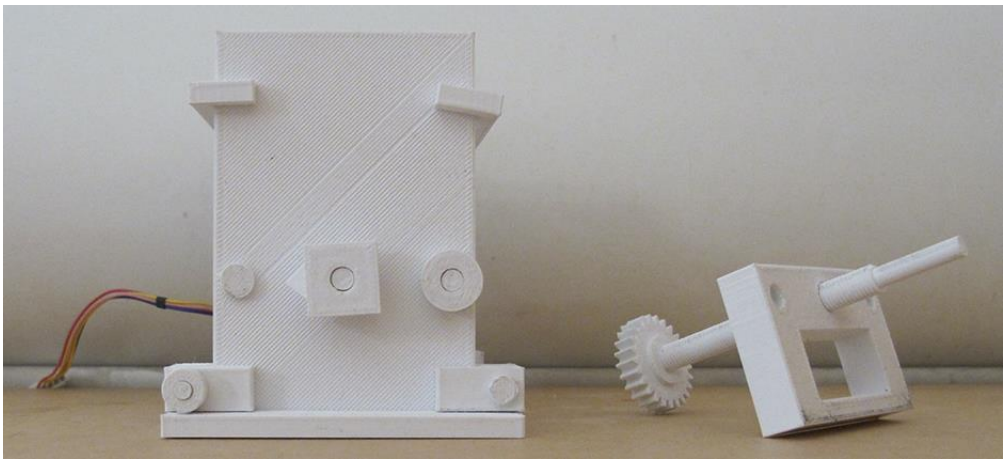
Rondana 3.

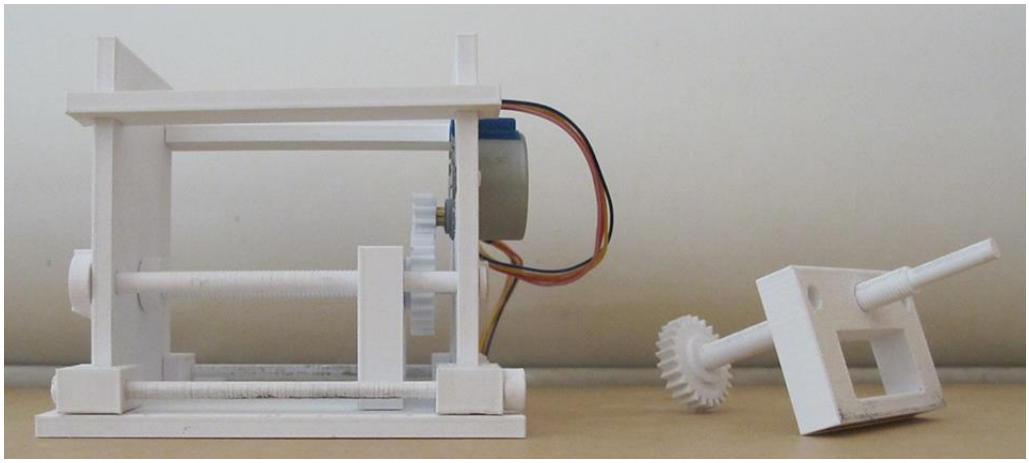

10.30
5.68
5.68
24.53
24.53
8.00
38.00
17.00
37.70
5.00
60.40
11.35
RULE
SCALE 4:1
UNITS mm

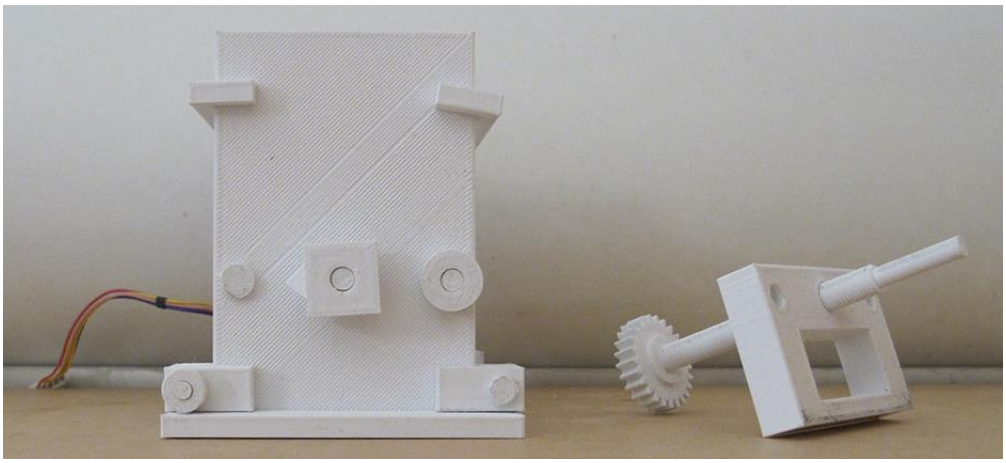
Regleta.

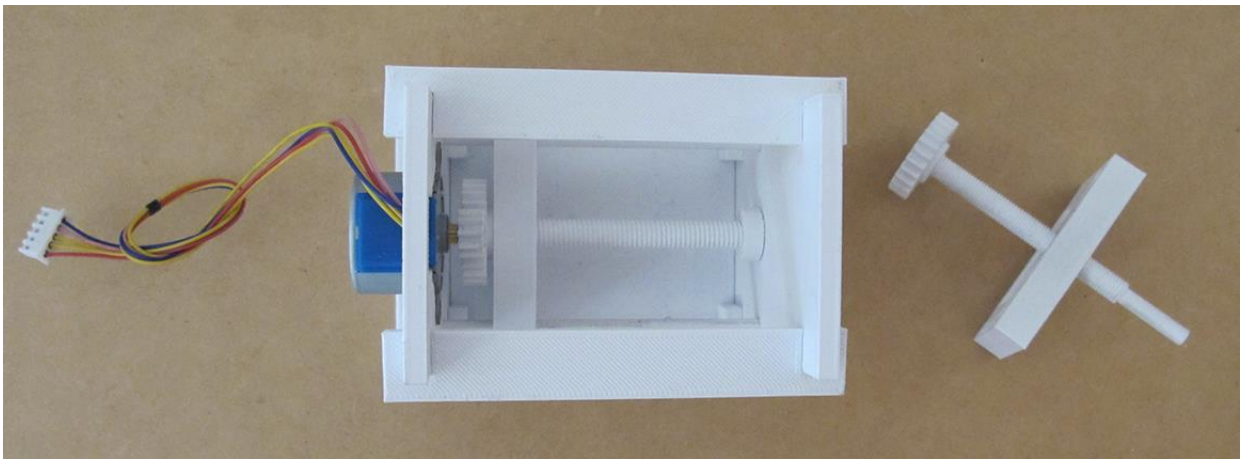








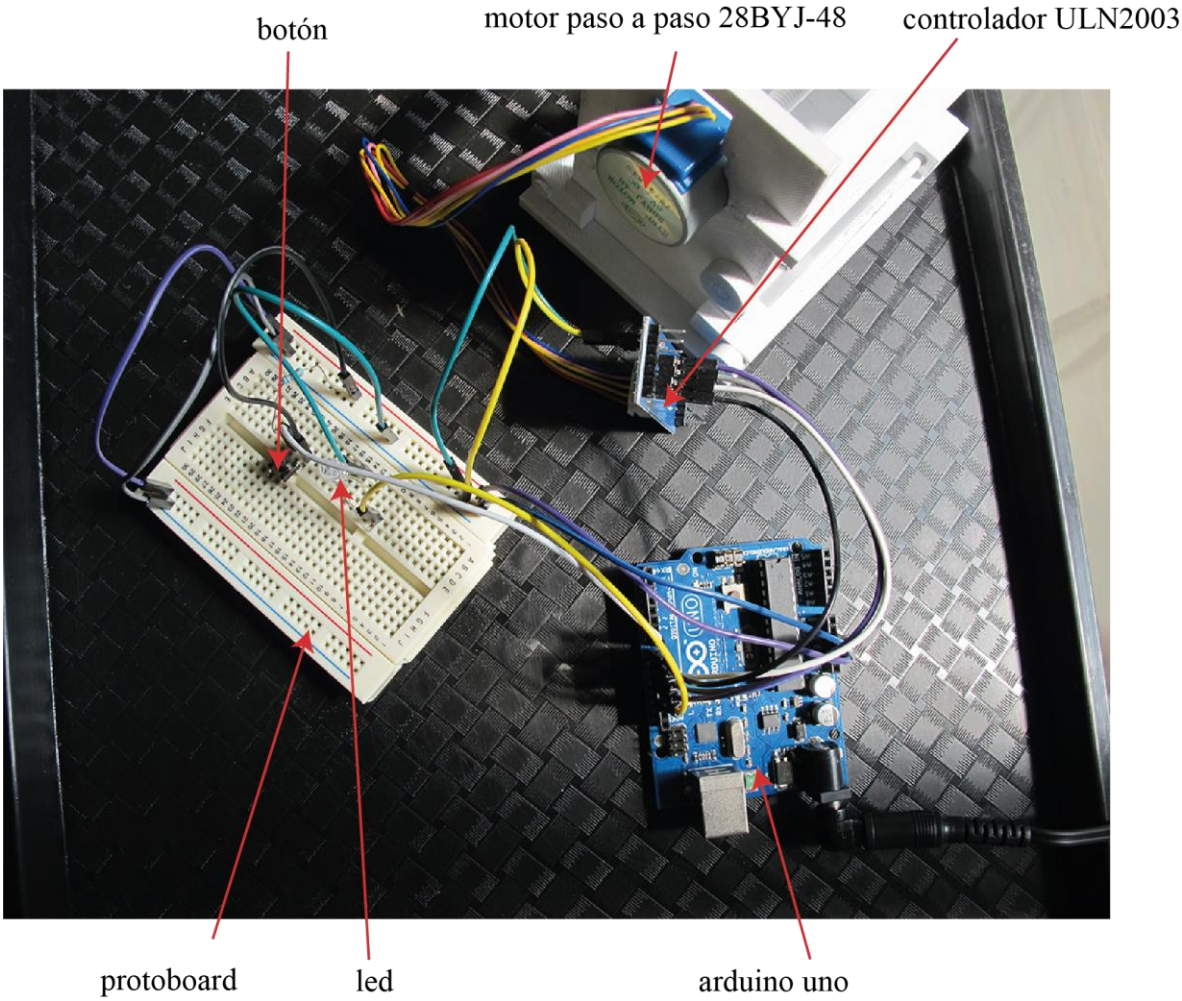




Impresión 3D del modelo.

En este modelo se tienen dos varillas roscadas (pag 16), la primera es M8x1 y M8x1.25 que nos definirán la distancia que se mueve la regleta acorde a los pasos y dos regletas con rosca para que se mueva. (pag. 29),

ESQUEMA DE CONEXIÓN EN ARDUINO UNO



El esquema de conexión del Arduino es la misma que se utilizó anteriormente, la diferencia está que la regleta se va a mover en ambos sentidos, con diferentes velocidades y pasos.

PROGRAMACIÓN DE PROCESAMIENTO EN ARDUINO.

En este modelo se realizaron varias pruebas con las dos varillas roscadas y con su respectiva regleta por lo que explicaré la programación que se fue realizando.

Programación 1: en este caso se utilizaron dos velocidades diferentes de 10 y 30 con signos + y – para que la regleta se mueva de ida y de regreso.

```
#include <Stepper.h>
int pasosMotor1=1024;
int pararMotor1=LOW;
```

```
Stepper motor1(pasosMotor1,8,10,9,11);
void setup() {

}

void loop() {
  if(pararMotor1 == LOW){
    pararMotor1 = HIGH;
  }
  for(int i = 0; i<=5; i++){
    motor1.setSpeed(30);
   motor1.step(pasosMotor1*10);
  }
         delay(3000);
         for(int h = 0; h <=3; h++){
            motor1.setSpeed(10);
        motor1.step(-pasosMotor1*10);
         }

        delay(3000);
        for(int j = 0; j<= 10; j++){
           motor1.setSpeed(30);
        motor1.step(-pasosMotor1*10);
        }
        delay(3000);
        pararMotor1 = LOW;
        delay(3000);
        pararMotor1 = HIGH;
}
```

Programación 2: en este caso se agregó un contador y el led en la programación. La velocidad es la misma, pero se cambiaron los pasos.

```
#include <Stepper.h>
int pasosMotor1=1024;
int pararMotor1=LOW;
const int ledPIN = 13;
int contador = 0;
Stepper motor1(pasosMotor1,8,10,9,11);

void setup() {
  pinMode(ledPIN , OUTPUT);
  //pinMode(boton, INPUT);
  Serial.begin(9600);
}
```

```
void loop() {
  if(pararMotor1 == LOW){
   delay(200);
  }
    contador++;
    Serial.println(contador);
    digitalWrite(ledPIN , HIGH);
    motor1.setSpeed(10);
    motor1.step(pasosMotor1*10);
    digitalWrite(ledPIN , LOW);
    delay(20000);
    contador++;
    Serial.println(contador);
    digitalWrite(ledPIN , HIGH);
    motor1.setSpeed(10);
    motor1.step(pasosMotor1*4);
    digitalWrite(ledPIN , LOW);
    delay(20000);
    contador++;
    Serial.println(contador);
    digitalWrite(ledPIN , HIGH);
    motor1.setSpeed(10);
    motor1.step(pasosMotor1*4);
    digitalWrite(ledPIN , LOW);
    delay(20000);
    contador++;
    Serial.println(contador);
    digitalWrite(ledPIN , HIGH);
    motor1.setSpeed(10);
    motor1.step( pasosMotor1*2);
   digitalWrite(ledPIN , LOW);
   delay(18000);
   digitalWrite(ledPIN , HIGH);
}
```

Programación 3: esta programación es la misma que se utilizó en el primer modelo, pero se agregó el loop10 para que el movimiento de ida y de regreso sea constante.

```
#include <Stepper.h>
int pasosMotor1=1024;
int distancia=pasosMotor1; // una vuelta del engranaje conductor
int input;
int dato=0;
int boton1 = 7;
const int ledPin = 13;
Stepper motor1(pasosMotor1,8,10,9,11);
```

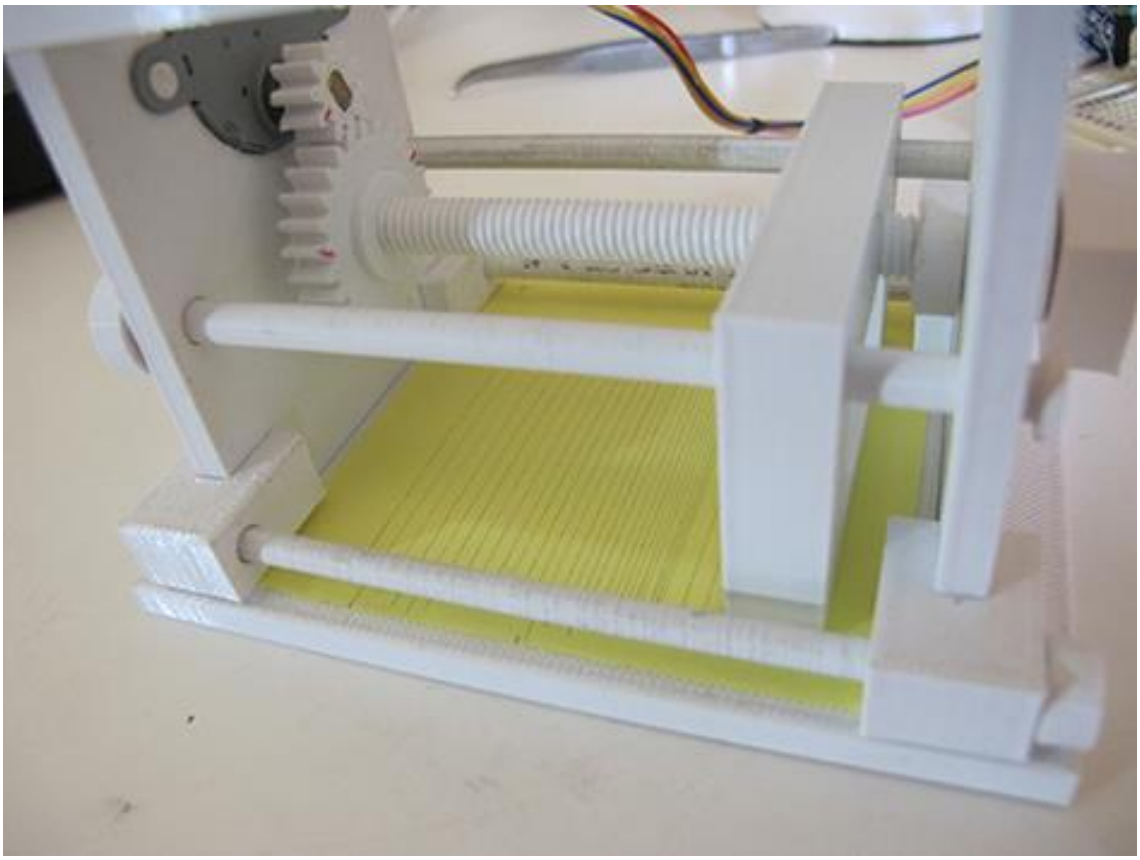
```
 void setup(){
  pinMode(ledPin,OUTPUT);
  pinMode(boton1,OUTPUT);
  Serial.begin(9600);
 }

 //protocolo
 //[MotorEncendido]
 //[MotorApagado  ]
 //[LedEncendido  ]
 //[LedApagado    ]

void loop(){
 if(Serial.available()){
    dato=Serial.parseInt();
    int loop0();
    int loop1();
    int loop4();
    int loop5();
    int loop10();
    if(input == 1){
      loop1();
    }else
     if(input == 0){
        loop0();
    }
    if(dato == 4){
      loop4();
    }
    if(dato == 5){
      loop5();
    }
    if(dato == 10){
      loop10();
    }
  }
}

 void loop0(){
     digitalWrite(ledPin, LOW);
     delay(100);
  }
 void loop1(){
     digitalWrite(ledPin, HIGH);
     delay(100);
```

```
}

 void loop4(){
   loop1();
   Serial.print("[MotorEncendido4]");
   Serial.print("[LedEncendido4  ]");
   (motor1, HIGH);
    for(int h = 0; h<=18; h++){
      motor1.setSpeed(30);
      motor1.step(-distancia*10);
      if(digitalRead(boton1) == true){
       Serial.print("[BotónPresionad]");
       (motor1, LOW);
      delay(100);
      break;
      }
   }
      loop0();
      (motor1, LOW);
      Serial.print("[MotorApagado4  ]");
      Serial.print("[LedApagado4    ]");
      delay(100);
}

void loop5(){
   loop1();
   Serial.print("[MotorEncendido5]");
   Serial.print("[LedEncendido5  ]");
   (motor1, HIGH);
   for(int i = 0; i<=18; i ++){
     motor1.setSpeed(30);
     motor1.step(distancia*10);
      if(digitalRead(boton1) == true){
       Serial.print("[BotónPresionad]");
       (motor1, LOW);
       delay(100);
       break;
     }
   }
    loop0();
      (motor1, LOW);
      Serial.print("[MotorApagado4  ]");
      Serial.print("[LedApagado4    ]");
      delay(100);
}
```

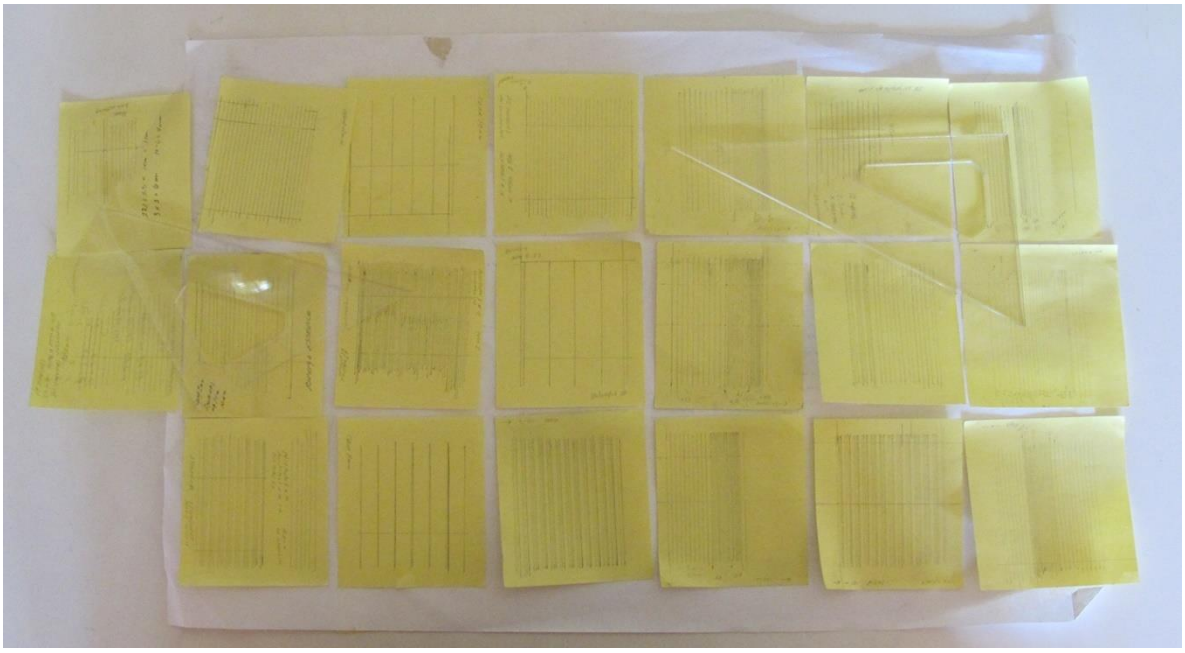
```
void loop10(){
 loop4();
 delay(100);
 loop5();
 delay(100);
}
```

DISTANCIA QUE RECORRE LA REGLETA (prisma con rosca)

El objetivo de esta segunda prueba es el de verificar y analizar que la impresión 3D que se realizó del modelo si funcionará en el prototipo final del pantógrafo XYZ.
Por lo que nos dimos a la tarea de ir marcando sobre una hoja de papel las distancias que recorre la regleta acorde a la programación que anteriormente se presentó.



Líneas trazadas sobre la hoja de papel.



Hojas con los trazos de las líneas.

ANÁLISIS DE LOS RESULTADO

Se realizaron varios análisis para comprobar que la impresión 3D del sistema formado por la varilla roscada, la regleta o prisma roscado y los engranajes rectos funcionan correctamente y que los incrementos sean menores a 3 mm a través de la programación del Arduino UNO.
Para realizar el análisis, en la programación del Arduino UNO, la velocidad del motor, los pasos del motor, las direcciones motrices y el retardo se consideran como se indica en la Tabla 1:

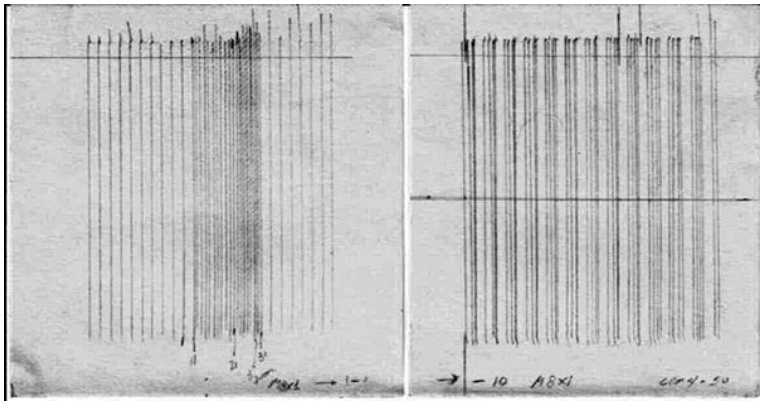
	Primer caso: engranaje + varilla roscada M8x1	Segundo caso: engranaje + varilla roscada M8x1.25
Velocidad del motor	30, 20, 10	30, 20, 10
Pasos del motor	10, 4 2	10, 4 2
Dirección del motor	+ y -	+ y -
Demora	2000	2000

Tabla 1.

En ambos casos, las pruebas se realizaron tanto de ida y vuelta, es decir, la dirección del motor en el Arduino UNO sería positiva o negativa (+ o -).
Comprobamos con el prisma con flecha las vueltas que da el engranaje conducido cuando el motor se mueve. En 2048 pasos, el engranaje impulsor da cinco vueltas y el engranaje impulsado da dos vueltas.
Con el contador comprobamos que el cambio de pasos es correcto ya que, se programó para que el motor tenga diferentes pasos 10, 4 y 2 en una sola ejecución.
En la programación utilizamos un retardo de 2000 microsegundos (dos segundos) para que cuando el procesador espera, podíamos marcar la línea en el papel donde se detiene el prisma roscado.

Resultados del primer caso:

Debido a los residuos que dejó la impresión 3D, la regleta o el prisma roscado no se acopló inicialmente a la varilla roscada. Después de varias pasadas, los residuos de impresión 3D se eliminaron debido a la fricción.
Cuando escribimos en la programación de Arduino UNO que la velocidad del motor es 30 o 20, el hilo del prisma saltó el paso de rosca de la varilla roscada. Encontrando que la distancia cuando la regleta se movía era variable y la línea que estaba marcada en el papel no era paralela a la regleta.
Continuamos con la velocidad de 10 del motor en la programación de Arduino UNO, con los diferentes pasos del motor, y en los dos sentidos (+ y -). En este caso, el hilo del prisma no saltó el paso de la varilla roscada.
La distancia que comenzamos a analizar fue de diez pasos en donde el engranaje impulsor da cinco vueltas y el engranaje conducido da dos vueltas. Se comprobó que la distancia que recorre la regleta es constante a 2 mm.
Continuamos el análisis marcando las líneas de los pasos en el papel, primero de 10, 4 y 2 pasos en cada diez intervalos, y segundo, alternamos los pasos 10, 4 y 2 en un solo intervalo. Las líneas marcadas en el papel están paralelas al prisma roscado.



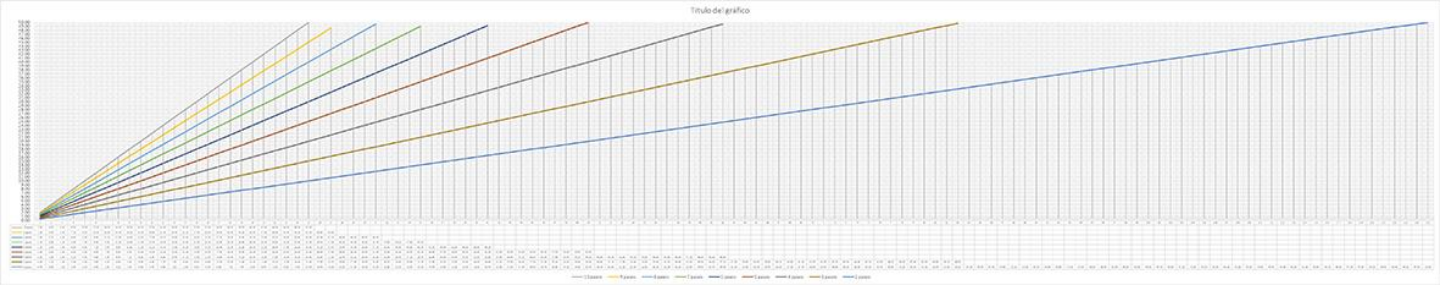
Líneas marcadas en el papel.

La distancia que recorre el prisma roscado según el número de pasos sería la siguiente (Tabla. 2):

Pasos	10	9	8	7	6	5	4	3	2
Distancia(mm)	2	1.8	1.6	1.4	1.2	1	0.8	0.6	0.4

Tabla 2.

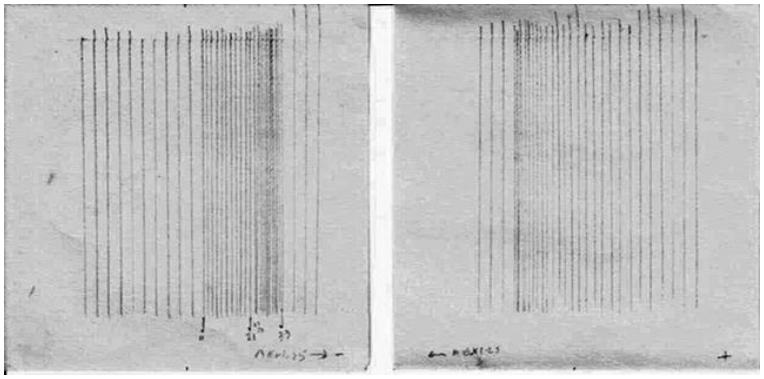
En la programación que se presentó anteriormente se utilizó el “for” en donde se especificó el número de pasos que se quiere que realice el motor paso a paso indicado con la letra “h” que es la variable. En la figura que a continuación se presenta se puede observar el análisis que se realizó para determinar el número de pasos acorde a la distancia total que se quiere que se mueva la regleta o prisma con rosca.



Análisis pasos/distancias.

Resultados del Segundo caso:

Al igual que en el caso 1, inicialmente la regleta o prima con rosca no se acoplaba a la varilla roscada debido a los residuos que dejaban la impresión 3D. Después de varias pasadas, los residuos de impresión 3D se eliminaron debido a la fricción.
Al programar el Arduino UNO con una velocidad de motor de 30, 20 y 10, la regleta o prisma roscado no saltó el paso de rosca de la varilla roscada.
La distancia que empezamos a analizar fue de diez pasos, el engranaje impulsor da cinco vueltas, y el engranaje conducido da dos vueltas para comprobar que es la misma distancia que el paso de la varilla roscada, 2,5 mm.
Continuamos el análisis marcando las líneas de los pasos en el papel, primero de 10, 4 y 2 pasos en intervalos de diez, y segundo, alternamos pasos de 10 a 2 en intervalos de tres.
Las líneas marcadas en el papel quedan paralelas al prisma roscado.



Líneas marcadas en el papel.

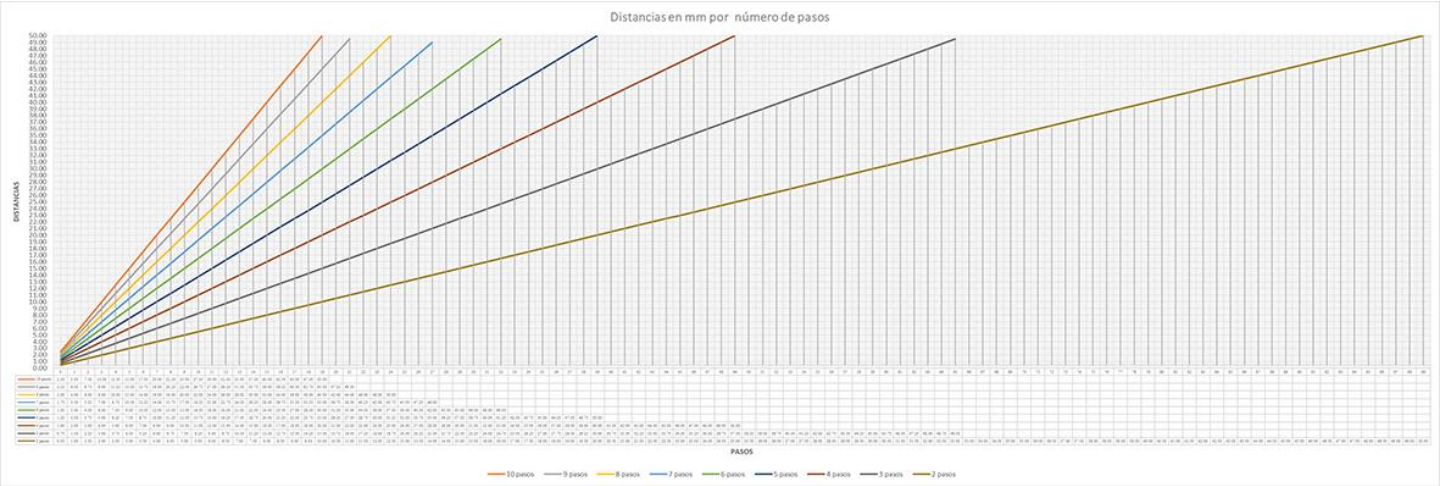
La distancia que recorre el prisma roscado según el número de pasos sería la siguiente (Tabla. 3):

Pasos	10	9	8	7	6	5	4	3	2
Distancia(mm)	2.5	2.25	2	1.75	1.5	1.25	1	0.75	0.5

Tabla 3.

En la programación que se presentó anteriormente se utilizó el “for” en donde se especificó el número de pasos que se quiere que realice el motor paso a paso indicado con la letra “h” que es la variable.

En la figura que a continuación se presenta se puede observar el análisis que se realizó para determinar el número de pasos acorde a la distancia total que se quiere que se mueva la regleta o prisma con rosca.



Análisis pasos/distancias.

Es importante mencionar que los resultados que se tiene en los dos gráficos serán la base para determinar cuántos pasos se quiere dar al motor paso a paso para que se divida la distancia total del largo del élitro de los escarabajos entre la distancia que recorre la regleta acorde al número de pasos. A partir de los datos que se van a obtener de las coordenadas X, Y, la estructura geométrica de los élitros de los escarabajos se realizará con mayor precisión en el programa de AutoCAD™.

Todo este análisis que realizamos del sistema de varillas roscadas, el prisma roscado y los engranajes rectos, nos llevó a concluir que la impresión 3D es eficaz para mover cualquier robot o máquina CNC a distancias inferiores a 3 mm.

SE ANEXA VIDEO “videoSegundaPrueba”

DESARROLLO DE INTERFAZ GRÁFICA Y PROGRAMACIÓN EN NETBEANS 8.2

Al terminar todo el análisis que realizamos del segundo modelo, nos preguntamos ¿cómo podemos realizar la interacción humano-ordenador, HCI (Human-Computer Interaction) para facilitar el uso del Arduino UNO? Diseñamos para este caso una interfaz gráfica en NetBeans 8.2 en donde el usuario solamente tiene que poner los datos que quiera para que el motor paso a paso y el led se prendan.

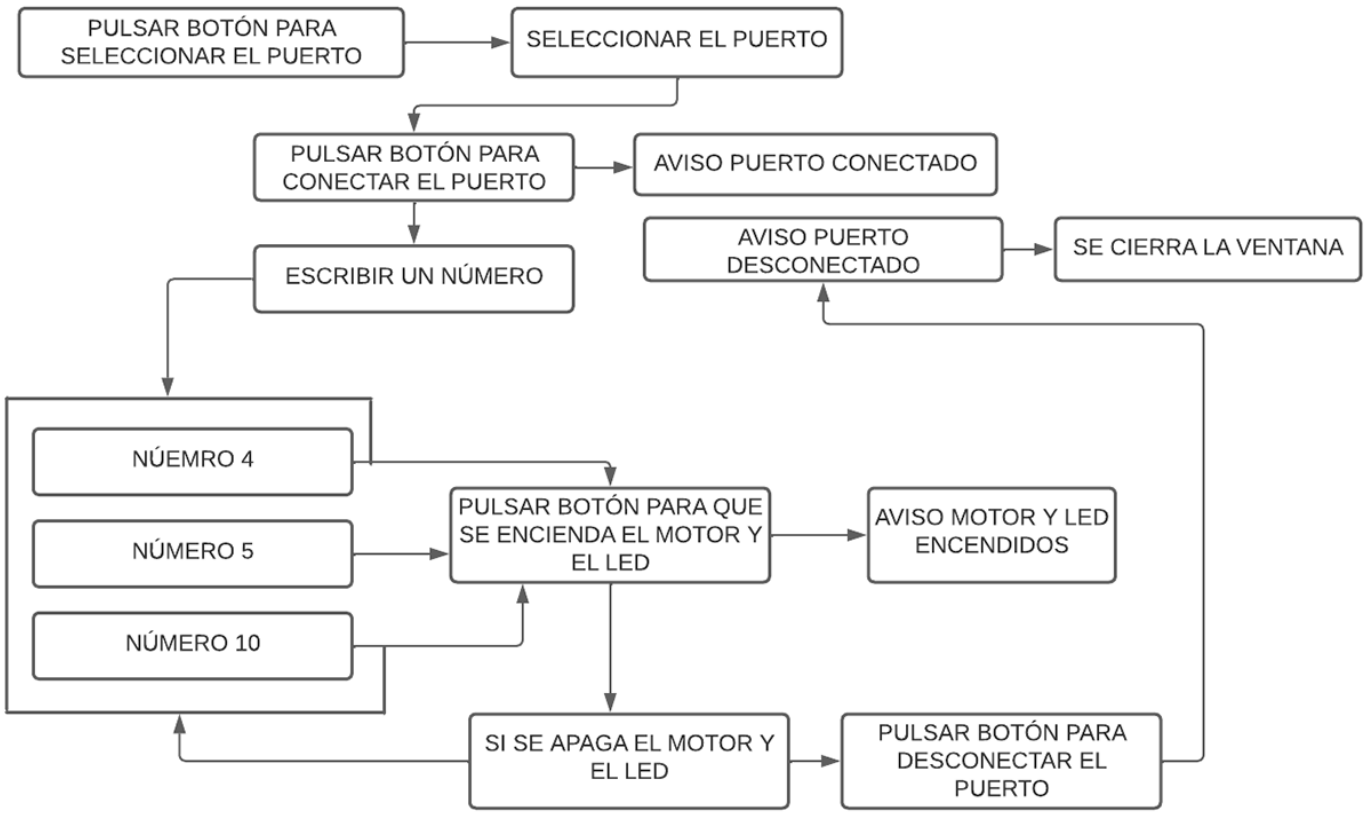
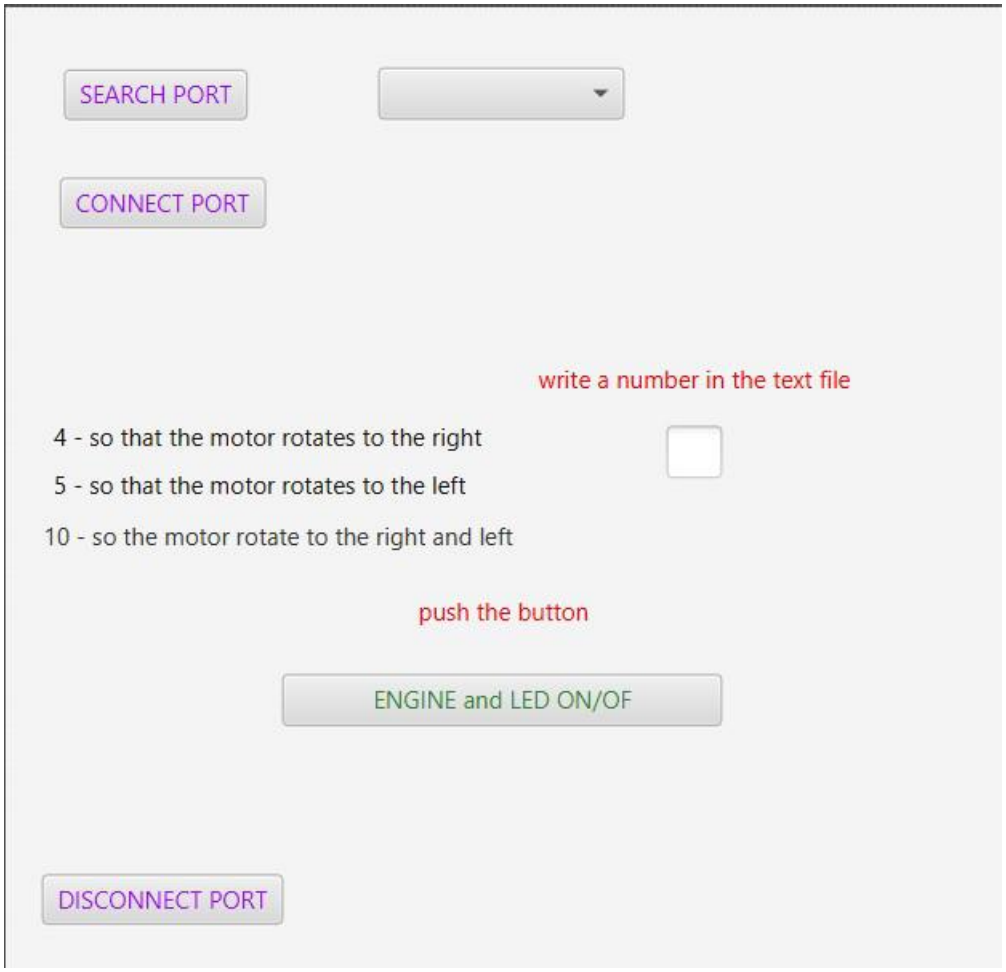


Diagrama de la interfaz gráfica.



Interfaz gráfica.

PROGRAMACIÓN

FXM DocumentArduino7.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ComboBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.shape.Circle?>
```

```
<AnchorPane id="AnchorPane" fx:id="prenderMotor" maxHeight="1.7976931348623157E308"
maxWidth="1.7976931348623157E308" prefHeight="481.0" prefWidth="499.0"
xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="javafxarduino7.FXMLDocumentArduino7Controller">
   <children>


      <Button layoutX="29.0" layoutY="31.0" mnemonicParsing="false" onAction="#botonBuscar"
text="SEARCH PORT" textFill="#9f07f7" />
      <ComboBox fx:id="combo" layoutX="186.0" layoutY="30.0" prefHeight="26.0" prefWidth="123.0" />
      <Button layoutX="27.0" layoutY="85.0" mnemonicParsing="false" onAction="#botonConectar"
text="CONNECT PORT" textFill="#9f07f7" />
      <Button layoutX="18.0" layoutY="433.0" mnemonicParsing="false" onAction="#botonDesconectar"
text="DISCONNECT PORT" textFill="#9f07f7" />
      <Label fx:id="label2" layoutX="27.0" layoutY="126.0" prefHeight="18.0" prefWidth="350.0" />
      <Label fx:id="label3" layoutX="32.0" layoutY="437.0" prefHeight="18.0" prefWidth="339.0" />
      <Button fx:id="encenderMotor1" alignment="CENTER" contentDisplay="CENTER"
layoutX="138.0" layoutY="333.0" mnemonicParsing="false" onAction="#botonEncenderMotor1"
prefHeight="26.0" prefWidth="220.0" text="ENGINE and LED ON/OF" textFill="#317e2e" />
      <TextField fx:id="textoMotor1" alignment="CENTER" layoutX="330.0" layoutY="209.0"
prefHeight="26.0" prefWidth="28.0" />
      <Label alignment="TOP  LEFT" layoutX="24.0" layoutY="206.0" prefHeight="18.0"
prefWidth="290.0" text="4 - so that the motor rotates to the right" textFill="#131212" />
      <Label alignment="TOP  LEFT" contentDisplay="CENTER" layoutX="24.0" layoutY="230.0"
prefHeight="18.0" prefWidth="290.0" text="5 - so that the motor rotates to the left"
textAlignment="CENTER" textFill="#131212" />
      <Label alignment="CENTER" contentDisplay="CENTER" layoutX="227.0" layoutY="177.0"
prefHeight="18.0" prefWidth="233.0" text="write a number in the text file" textFill="#f70505" />
      <Label alignment="CENTER" layoutX="139.0" layoutY="293.0" prefHeight="18.0"
prefWidth="220.0" text="push the button" textFill="#f70505" />
      <Label fx:id="label4" alignment="CENTER" contentDisplay="CENTER" layoutX="138.0"
layoutY="375.0" prefHeight="18.0" prefWidth="220.0" textFill="#0a72f2" />
      <Circle fx:id="c1" fill="#2197ff00" layoutX="419.0" layoutY="312.0" radius="10.0"
stroke="TRANSPARENT" strokeType="INSIDE" />
      <Label alignment="TOP  LEFT" contentDisplay="CENTER" layoutX="19.0" layoutY="256.0"
prefHeight="18.0" prefWidth="290.0" text="10 - so the motor rotate to the right and left" />
   </children>
</AnchorPane>
```

JavaFxArduino7.java

```
package javafxarduino7;
import java.io.InputStream;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.transform.Translate;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import jssc.SerialPort;
import jssc.SerialPortEvent;
import jssc.SerialPortEventListener;
import jssc.SerialPortException;
import jssc.SerialPortList;
/**
 *
 * @author usuario Dina Rochman
 */
public class JavaFXArduino7 extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("FXMLDocumentArduino7.fxml"));
        Scene scene = new Scene(root, 500, 500);
        stage.setTitle("PANTÓGRAFO XYZ");
        stage.setResizable(false);
        stage.setScene(scene);
        stage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws SerialPortException {
        launch(args);
        String puertos[]= SerialPortList.getPortNames();
        for(String n : puertos){
            System.out.println(n);
        }
```

```
        SerialPort sp = new SerialPort("COM4");
        try{
            sp.openPort();
            sp.setParams(SerialPort.BAUDRATE_9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);
            sp.addEventListener(new LecturaSerial(sp), SerialPort.MASK_RXCHAR);
            Thread.sleep(1500);
            while(true){
              Scanner sc = new Scanner(System.in);
              System.out.println("");
              String csd = sc.next();
               System.out.println("Enviando");
               sp.writeString(csd);
               Thread.sleep(5000);
            }
        } catch (InterruptedException ex) {
            Logger.getLogger(JavaFXArduino7.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

class LecturaSerial implements SerialPortEventListener{

    private SerialPort sp;

    public LecturaSerial(SerialPort sp, Circle c1, Circle c2){
        this.sp=sp;
    }

    LecturaSerial(SerialPort sp) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated
methods, choose Tools | Templates.
    }
    @Override
    public void serialEvent(SerialPortEvent serialPortEvent) {
        try{
            String msg="";
            msg=sp.readString(17);
            System.out.println(msg);
        } catch (SerialPortException ex) {
            Logger.getLogger(LecturaSerial.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```

FXMLDocumentArduino7Controller.java

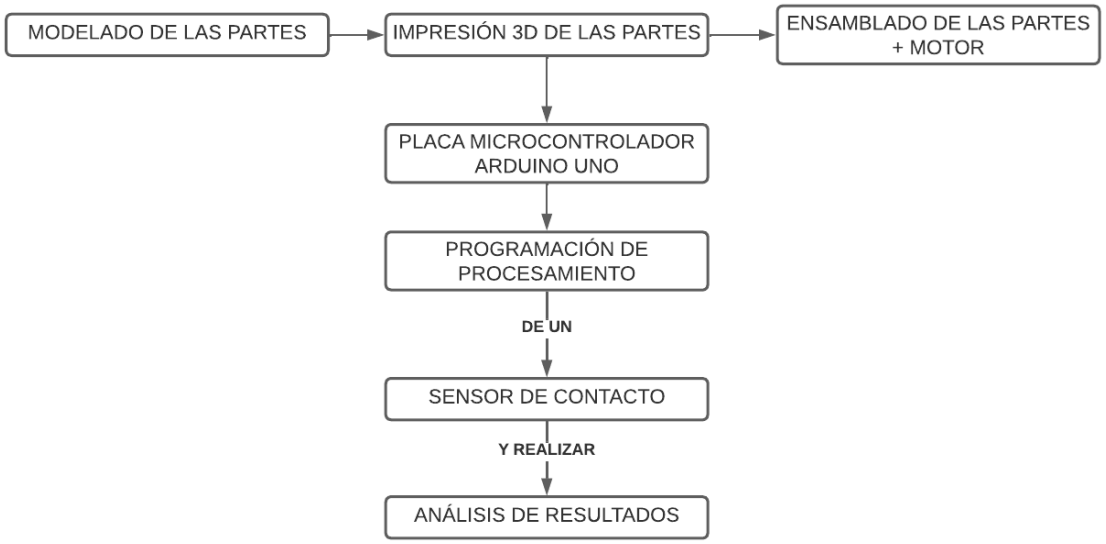
```
package javafxarduino7;

import java.net.URL;
import java.util.ResourceBundle;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import static javafx.scene.transform.Transform.translate;
import javafx.scene.transform.Translate;
import jssc.SerialPort;
import jssc.SerialPortEvent;
import jssc.SerialPortException;
import jssc.SerialPortList;

/**
 *
 * @author usuario Dina Rochman
 */
public class FXMLDocumentArduino7Controller implements Initializable {

    @FXML
    private Label label2;

    @FXML
    private Label label3;

    @FXML
    private Label label4;

    @FXML
```

```
    private AnchorPane prenderMotor;

    @FXML
    private ComboBox combo;

    @FXML
    private TextField texto;
    private SerialPort sp;

    @FXML
    private TextField textoMotor1;

    @FXML
     private Button encenderMotor1;

     private Object stage;


    @FXML
    private void botonBuscar(){
        ObservableList puertosCombo=FXCollections.observableArrayList();
        String puertos[]= SerialPortList.getPortNames();
        for(String n : puertos){
            puertosCombo.add(n);
        }
        combo.setItems(puertosCombo);
        combo.getSelectionModel().selectFirst();
    }

    @FXML
    private void botonConectar() throws InterruptedException{
        System.out.println("puerto conectado");
        label2.setText("puerto conectado");
         sp = new SerialPort(combo.getSelectionModel().getSelectedItem().toString());
        try{
            sp.openPort();
            sp.setParams(SerialPort.BAUDRATE_9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);
            sp.addEventListener(new LecturaSerial(sp), SerialPort.MASK_RXCHAR);
        } catch (SerialPortException ex) {
            Logger.getLogger(FXMLDocumentArduino7Controller.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    @FXML
    private void botonDesconectar(){
```

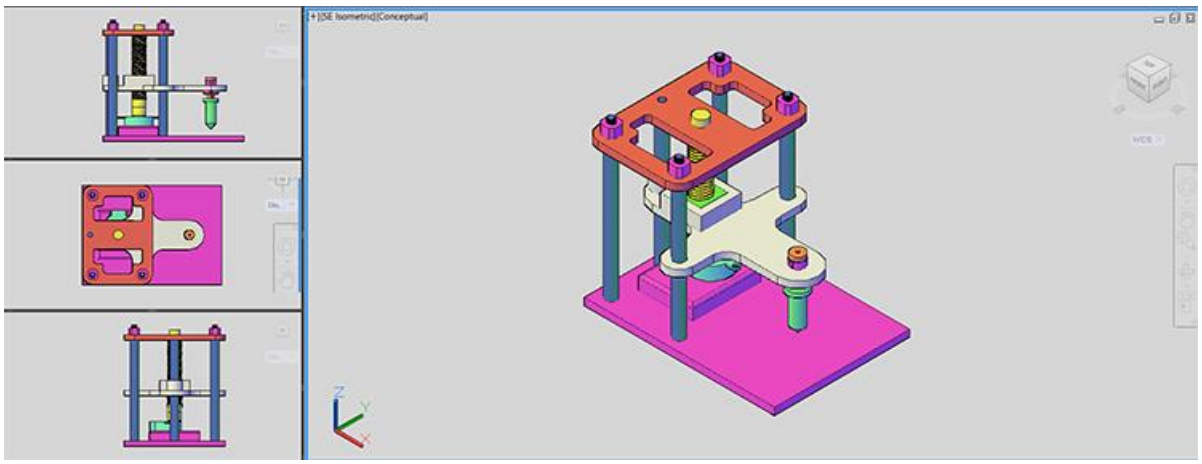
```
        System.out.println("puerto desconectado");
        label3.setText("puerto desconectado");
        try{
            sp.closePort();
            System.exit(0);
        } catch (SerialPortException ex) {
            Logger.getLogger(FXMLDocumentArduino7Controller.class.getName()).log(Level.SEVERE, null, ex);
        }
    }


    @FXML
    private void botonEncenderMotor1(){ //prende y apaga motor y leed
        label4.setText("ENGINE and LED ON");
        try{
            sp.writeString(textoMotor1.getText());
            textoMotor1.setText("");
        } catch (SerialPortException ex) {
            Logger.getLogger(FXMLDocumentArduino7Controller.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }
}
```

MODELO SENSOR DE CONTACTO

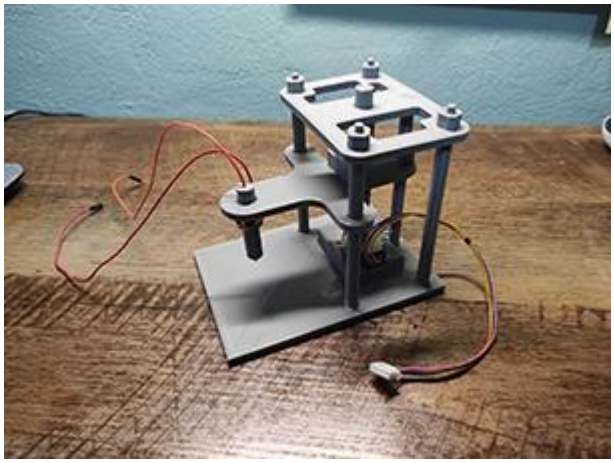
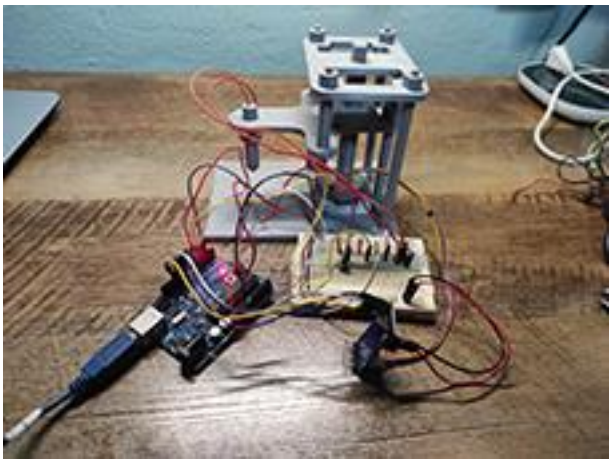
El modelo que a continuación se presenta es el de un sensor de contacto que sustituirá al botón pulsador que se tiene en el protoboard.



Metodología



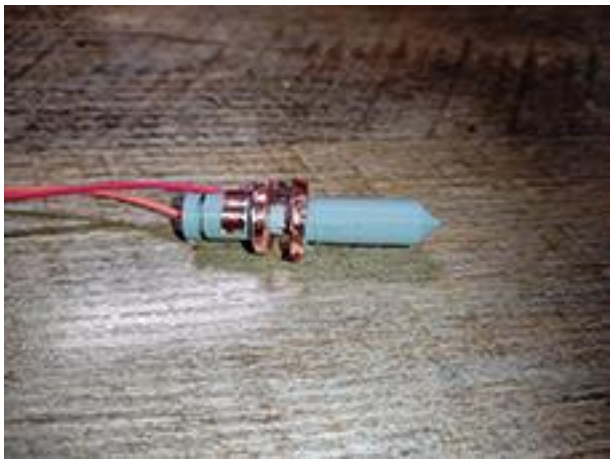
Modelado de las partes del sensor de contacto.

Impresión 3D, ensamblado + Arduino UNO.

El sensor de contacto consta de tres piezas. Para que funcione como botón se le adhirió lámina de cobre y se soldaron en cada una de las partes los cables (+ y -) para conectarlos al Arduino UNO.





Fotografías sensor de contacto con lámina de cobre y cables (+ y -).

PROGRAMACIÓN DE PROCESAMIENTO EN ARDUINO.

En este caso en la programación no se utilizó la librería stepper.

```
// codigo para el movimiento del giro del motor a pasos y sensor de movimiento o boton



byte PHorario = 2;     // giro en sentido horario
byte PAntiHorario = 3; // giro en sentido anti horario
byte PPasos = 4;       // numero de pasos
byte PVel = 5;         // velocidad del motor



byte IN1=8;  // 28BYJ48 In1
byte IN2=9;  // 28BYJ48 In2
byte IN3=10; // 28BYJ48 In3
byte IN4=11; // 28BYJ48 In4

int horario=1;


int paso=4; // numero de pasos
int Cpaso=0; //contador de pasos

int vel[5]={5,10,30,1000}; //velocidad
int Cvel=0; //contador de Velocidad

int conf=1; // secuencia de pasos

//secuencia 1 paso
const int UnPaso[4] = {      B1000,
                        B0100,
                        B0010,
                        B0001 };

//secuencia 2 pasos
const int DosPasos[4] = {    B1100,
                        B0110,
                        B0011,
                        B1001 };

// Secuencia a medio paso
byte const  MedioPaso[8] = { B1000,
                        B1100,
                        B0100,
```

```
                B0110,
                B0010,
                B0011,
                B0001,
                B1001 };


void puerto(int bits,int ini,int fin){
  for(int i=ini;i<=fin;i++)
  {
    digitalWrite(i,bitRead(bits,i-ini));
  }
}

void setup() {

  //pines entrada
  for(int i=2;i<=5;i++){
    pinMode(i,INPUT);
  }
  //pines salidas
  for(int i=IN1;i<=IN4;i++){
    pinMode(i,OUTPUT);
  }

}

void loop() {



  // Giro en Sentido Horario
  if(digitalRead(PHorario))  // Pregunta si pulsador horario fue presionado
  {

    horario=1;
    Cpaso=-1;
     delay(200);

  }

  // Giro en Sentido Anti-Horario
  if(digitalRead(PAntiHorario))  // Pregunta si pulsador horario fue presionado
  {
     horario=0;
    Cpaso=1;
```

```
  delay(0);


 }

 // Cambio de la secuencia de pasos
 if(digitalRead(PPasos))  // Pregunta si pulsador horario fue presionado
 {
  delay(100); //Anti-Rebote
  while(digitalRead(PPasos)); //Espera hasta soltar el boton
  delay(100); //Anti-Rebote
  conf++;
  //Si ya paso por las 3 configuraciones reinicie
  if(conf>3)
    conf=1;
  if(horario==1)
    Cpaso=-1;
  else
    Cpaso=paso;

  puerto(B0000,IN1,IN4);
 }

 // Velocidad del Motor
 if(digitalRead(PVel))  // Pregunta si pulsador horario fue presionado
 {
  delay(100); //Anti-Rebote
  while(digitalRead(PVel)); //Espera hasta soltar el boton
  delay(100); //Anti-Rebote
  Cvel++;
  //Si ya paso por las 5 velocidades reinicie
  if(Cvel>4)
    Cvel=0;
 }

 //contadores
 if(horario==1)
 {
  Cpaso++;                    //Incremente la variable cont
  if(Cpaso>=paso)
     Cpaso=0;                      //Se pone Contador de pasos en cero
 }
 else{
  Cpaso--;                   //Decremente la variable cont
  if(Cpaso<0)
     Cpaso=paso-1;               //Se pone Contador igual al paso
```

```
 }


 //movimiento del motor
    switch(conf){
    case 1:
          puerto(UnPaso[Cpaso],IN1,IN4); //Envíe al puerto la información de la tabla
          paso=4;
          break;
    case 2:
          puerto(DosPasos[Cpaso],IN1,IN4); //Envíe al puerto la información de la tabla
          paso=4;
          break;
    case 3:
          puerto(MedioPaso[Cpaso],IN1,IN4); //Envíe al puerto la información de la tabla
          paso=8;
          break;
    }
     delay(vel[Cvel]);                  //Retardo de 100 milisegundos

}
```

RESULTADOS

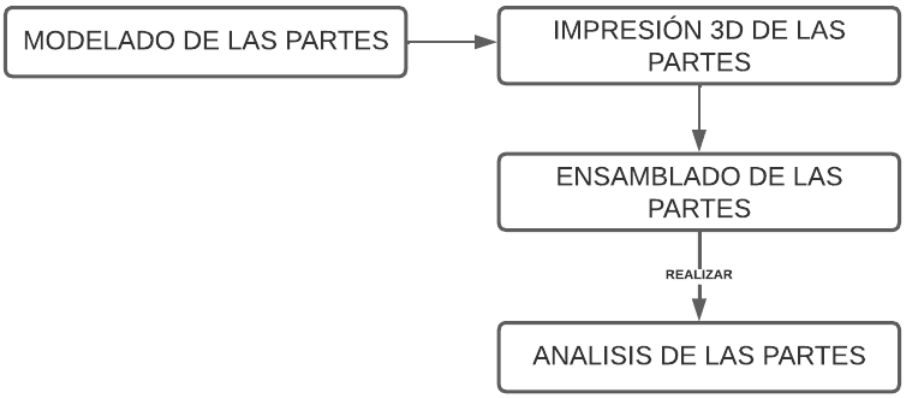
El sensor de contacto funciona correctamente ya que al momento que las dos láminas de cobre hacen el contacto el led se prende. Si la programación se cambia a que apague o prenda el motor funcionaría también correctamente.

SE ANEXAN DOS VIDEOS “videoSensor1” y “videoSensor2”

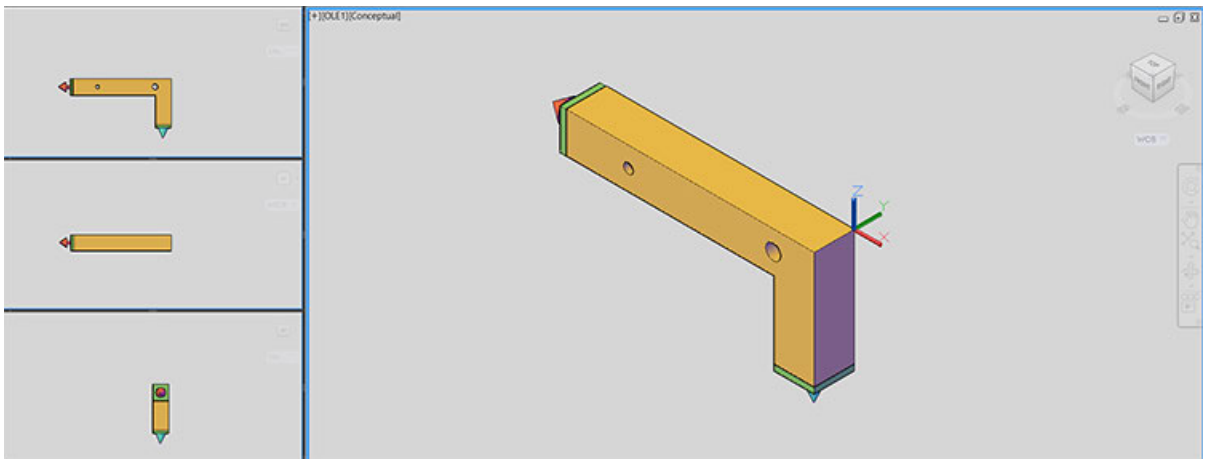
Este modelo de sensor de contacto nos llevó a desarrollar varios botones pulsadores los cuales serán parte del prototipo final.

BOTÓN PULSADOR 1

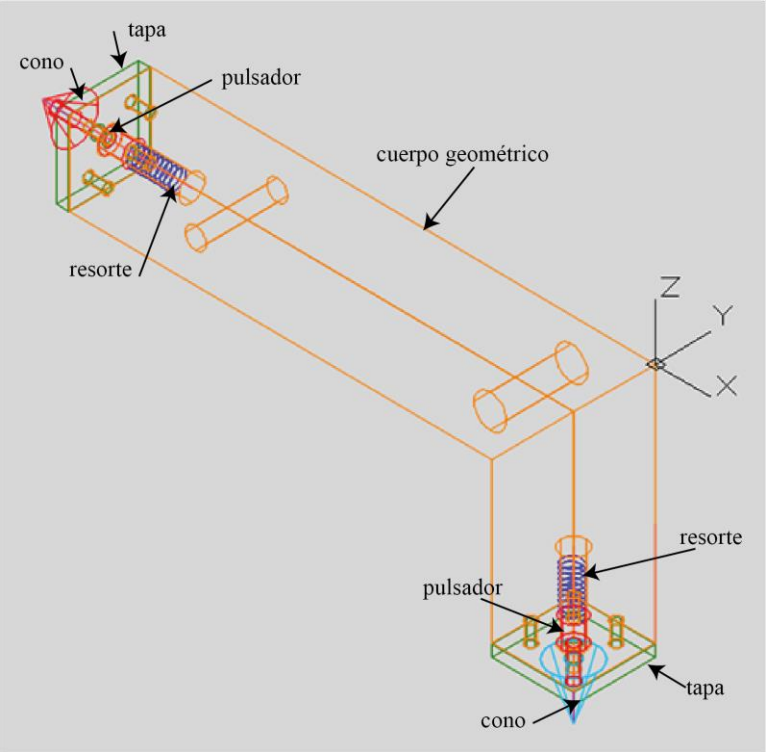
El objetivo principal de realizar la prueba del botón pulsador es el de verificar que la fuerza que ejerce el motor paso a paso es suficiente para que cuando la punta del botón pulsador hace contacto con una pared el resorte se contrae y cuando la punta del botón pulsador no hace contacto con la pared el resorte llegue a su estado original.



Metodología.

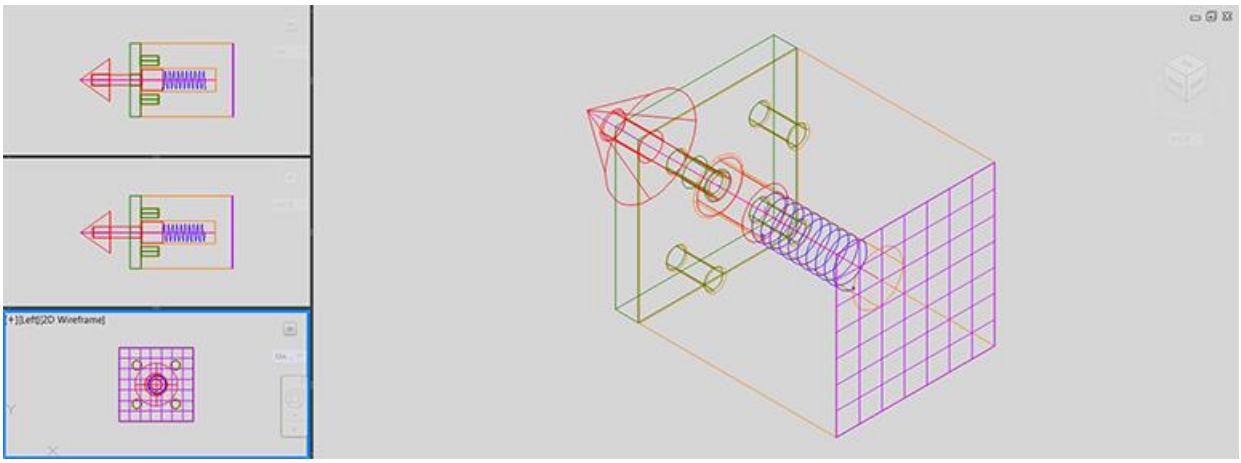


Modelo botón pulsador.

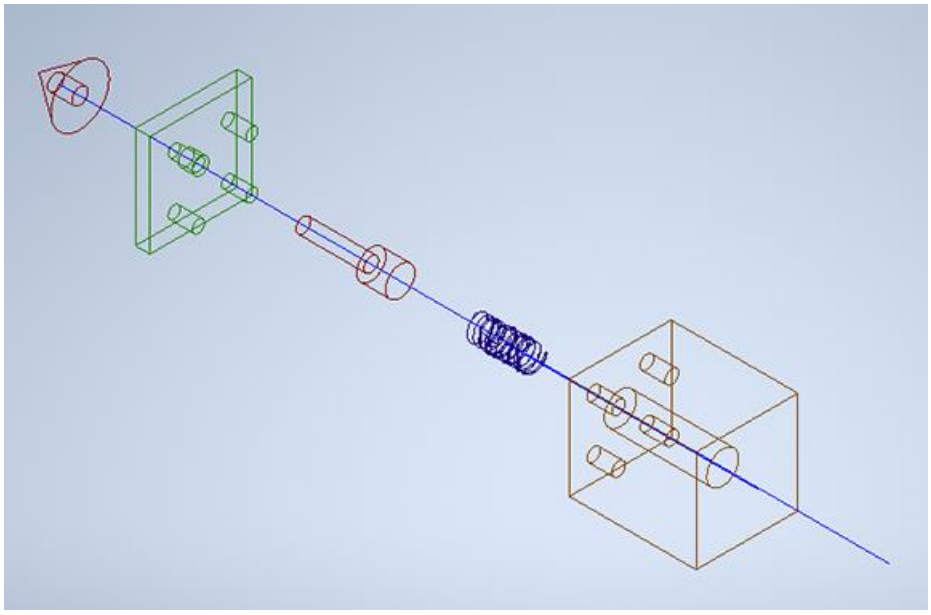


Partes del botón pulsador.

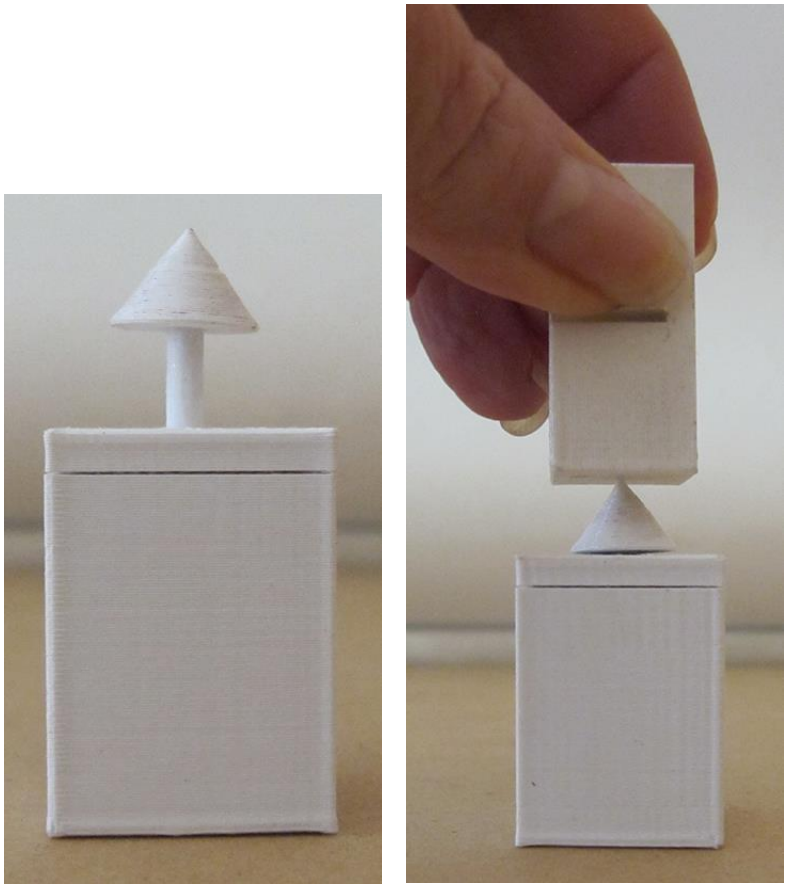
Para comprobar el objetivo que mencionamos anteriormente, cortamos la pieza para realizar la impresión 3D.



Detalle botón pulsador1.

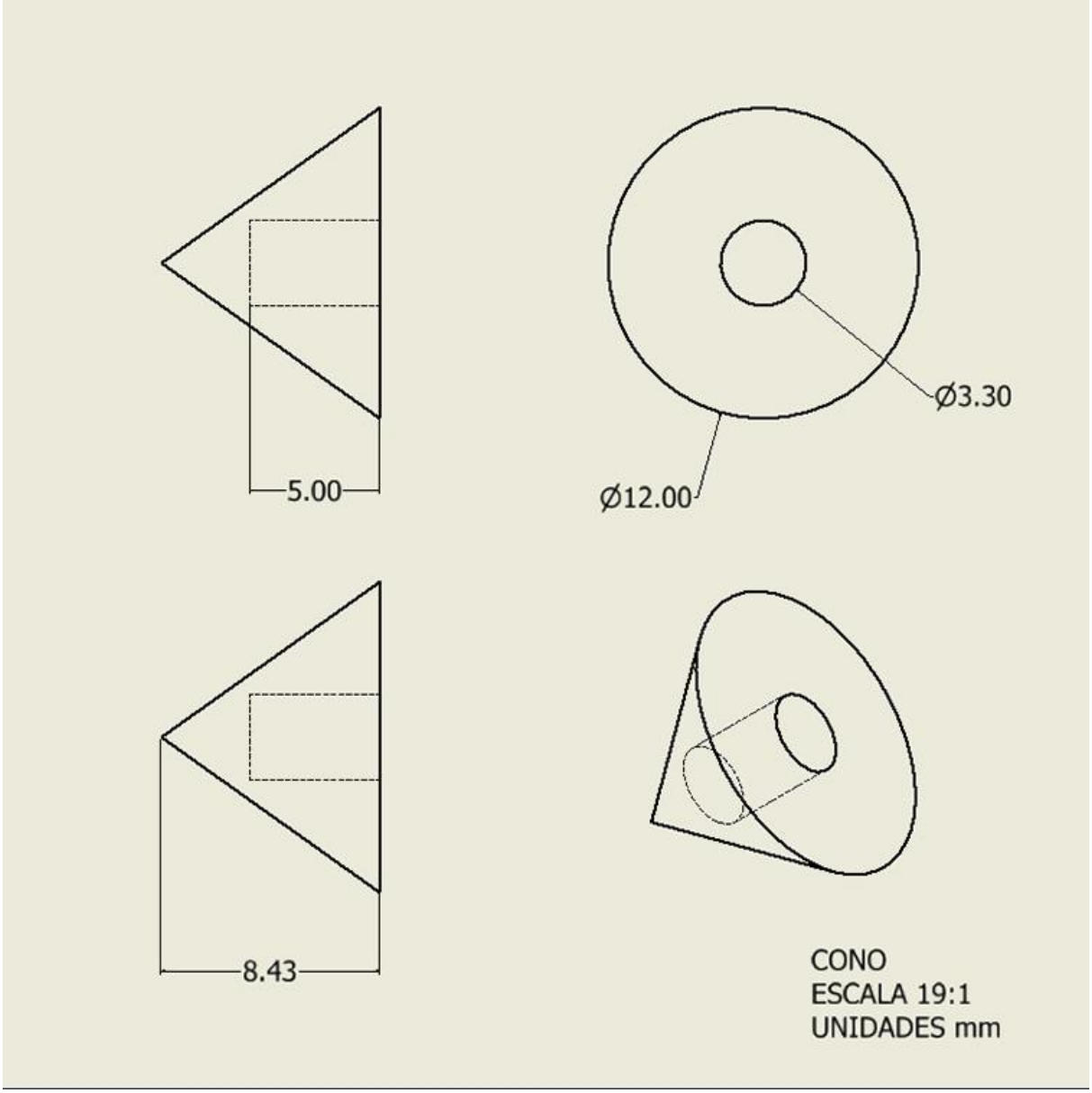


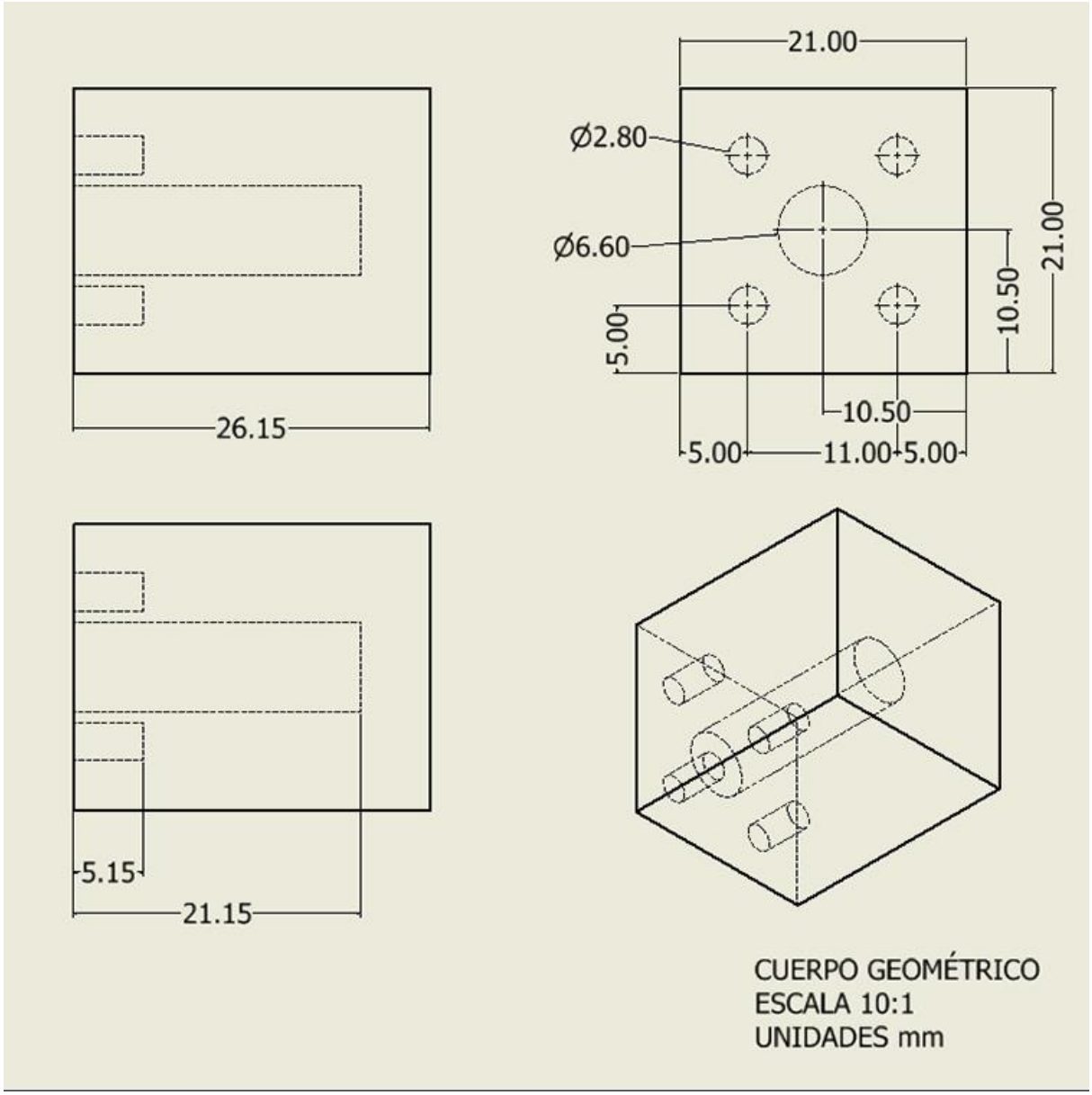
Explosivo detalle botón pulsador1.

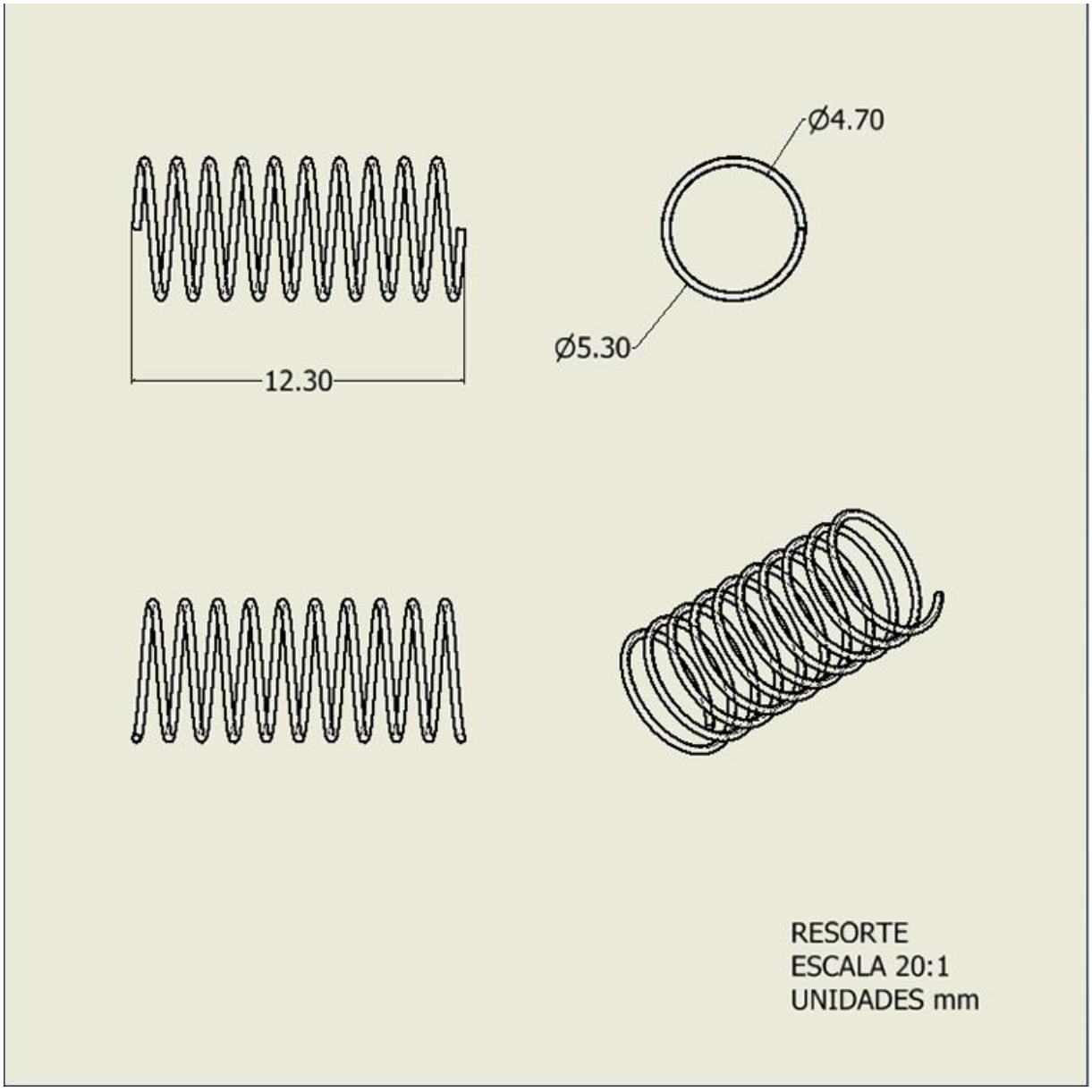


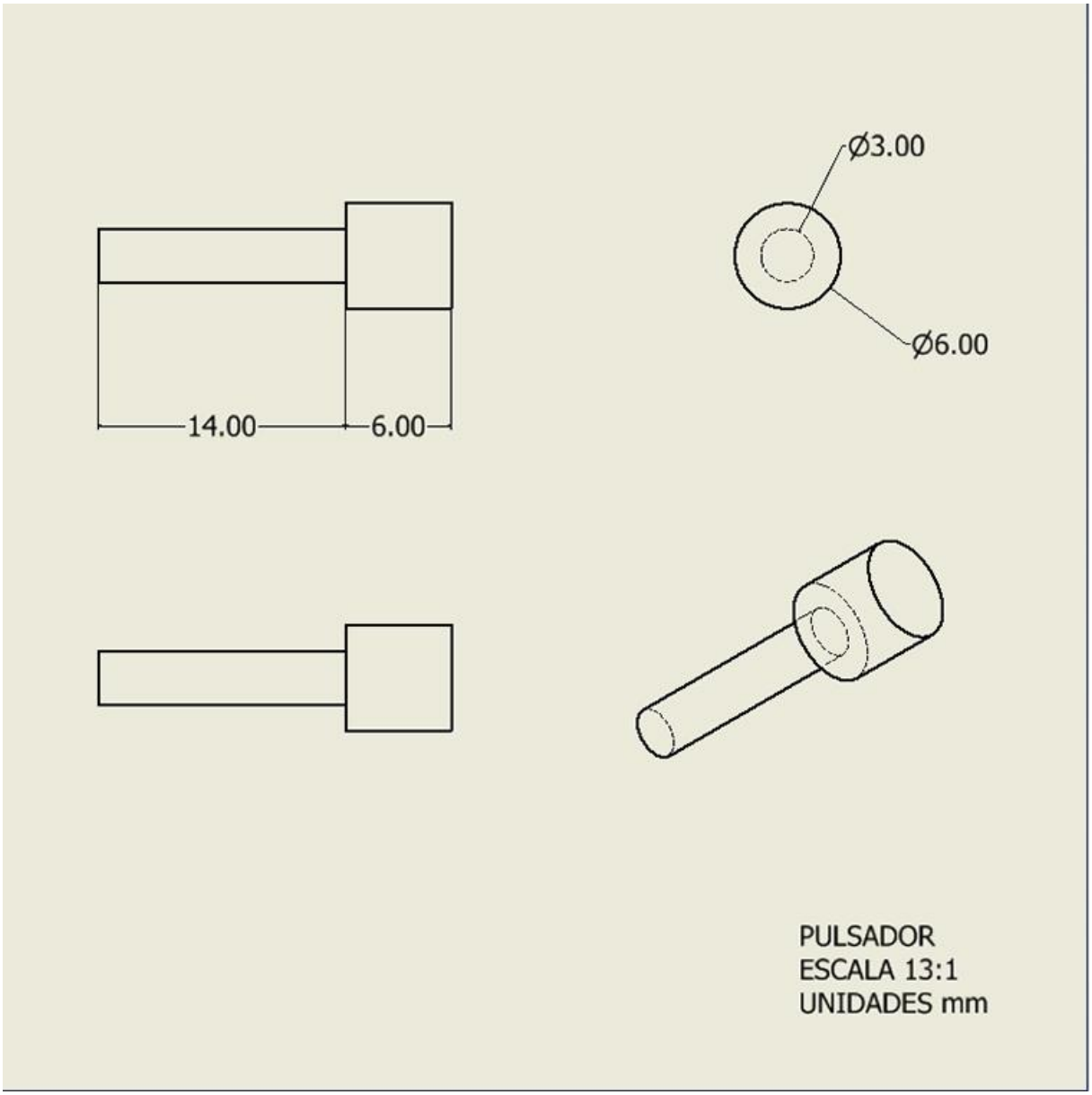
Impresión 3D botón pulsador.

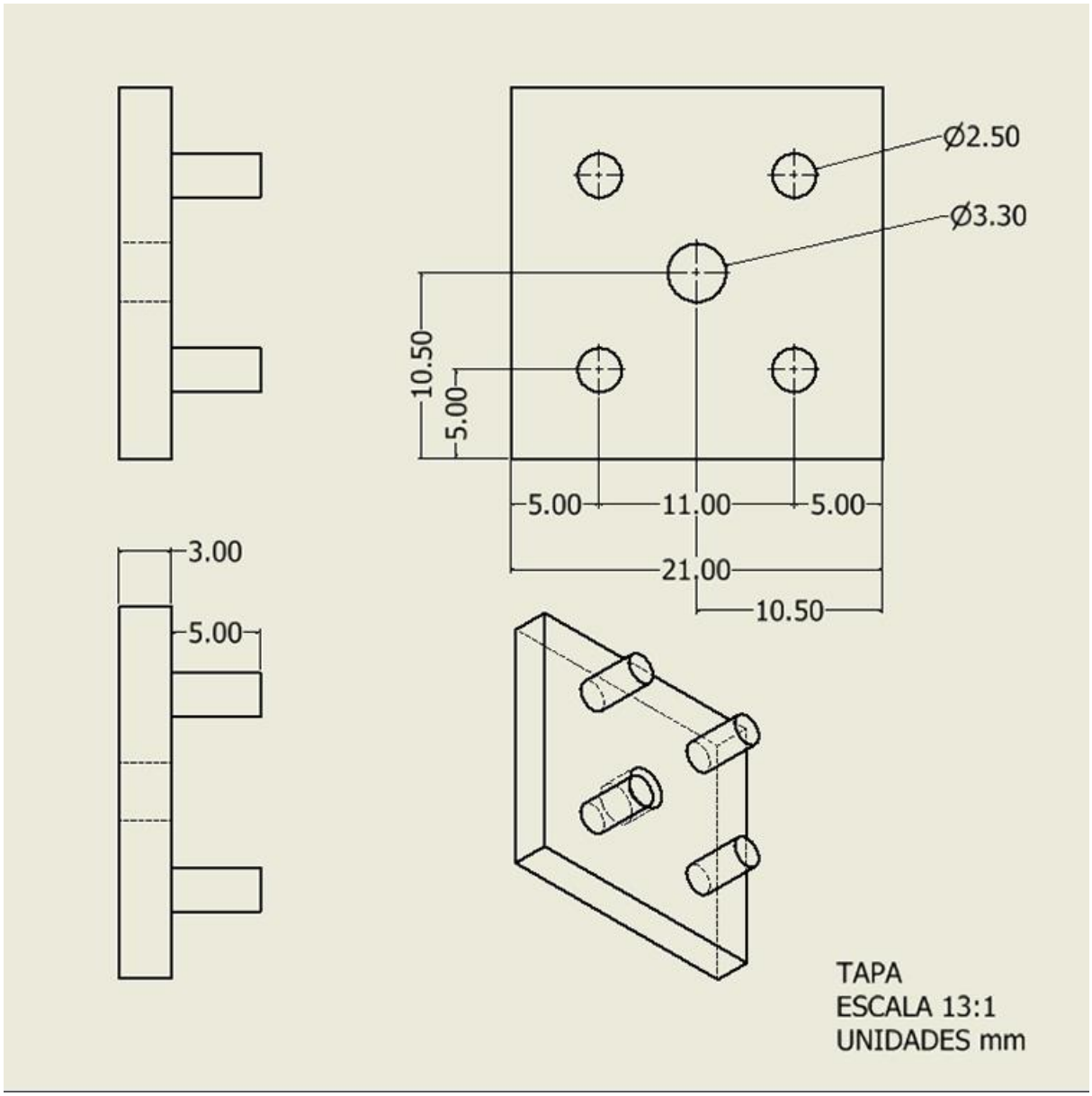
PLANOS CONSTRUCTIVOS DE LAS PIEZAS DEL BOTÓN PULSADOR




21.00
Ø2.80
Ø6.60
21.00
10.50
5.00
10.50
26.15
5.00
11.00
5.00
5.15
21.15
CUERPO GEOMÉTRICO
ESCALA 10:1
UNIDADES mm


Ø4.70
Ø5.30
12.30
RESORTE
ESCALA 20:1
UNIDADES mm


Ø3.00
Ø6.00
14.00
6.00
PULSADOR
ESCALA 13:1
UNIDADES mm


Ø2.50
Ø3.30
10.50
5.00
5.00
11.00
5.00
21.00
10.50
3.00
5.00
TAPA
ESCALA 13:1
UNIDADES mm

PROGRAMACIÓN DE PROCESAMIENTO EN ARDUINO.

```
#include <Stepper.h>
int pasosMotor1 = 1024;
int distancia = pasosMotor1;
int input;
int dato = 0;
int contador = 0;
int boton1 = 7;
const int ledPin = 13;
Stepper motor1(pasosMotor1, 8, 10, 9, 11);

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(boton1, OUTPUT);
  Serial.begin(9600);
}

//protocolo
//[MotorEncendido]
//[MotorApagado  ]
//[LedEncendido  ]
//[LedApagado    ]
//[contador      ]

void loop() {
  if (Serial.available()) {
    dato = Serial.parseInt();
    int loop0();
    int loop1();
    int loop4();
    int loop20();
    if (input == 1) {
      loop1();
    } else if (input == 0) {
      loop0();
    }
    if (dato == 4) {
      loop4();
    }
    if (dato == 5) {
      loop5();
    }
    if (dato == 10) {
      loop10();
    }
    if (dato == 20) {
```

```
      loop20();
    }
  }
}

void loop0() {
  digitalWrite(ledPin, LOW);
  delay(100);
}
void loop1() {
  digitalWrite(ledPin, HIGH);
  delay(100);
}

void loop4() {
  loop1();
  Serial.print("[MotorEncendido4]");
  Serial.print("[LedEncendido4  ]");
  (motor1, HIGH);
  for (int h = 0; h <= 18; h++) {
    motor1.setSpeed(30);
    motor1.step(distancia * 10);
    if (h == 15) {
      loop0();
      (motor1, LOW);
      Serial.print("[MotorApagado4  ]");
      Serial.print("[LedApagado4    ]");
      //Serial.print("[BotónPresionad]");
      delay(1000);
      break;
    }
  }
}

void loop5() {
  loop1();
  Serial.print("[MotorEncendido4]");
  Serial.print("[LedEncendido4  ]");
  (motor1, HIGH);
  for (int i = 0; i <= 18; i++) {
    motor1.setSpeed(30);
    motor1.step(-distancia * 10);
    if (i == 15) {
      loop0();
      (motor1, LOW);
      Serial.print("[MotorApagado4  ]");
```

```
      Serial.print("[LedApagado4    ]");
      //Serial.print("[BotónPresionad]");
      delay(1000);
      break;
    }
  }
}


void loop10() {
  loop4();
  delay(1000);
  loop5();
  delay(1000);
}

void loop20() {
  contador++;
  Serial.print(contador);
  loop1();
  Serial.print("[MotorEncendido4]");
  Serial.print("[LedEncendido4  ]");
  (motor1, HIGH);
  for (int i = 0; i <= 44; i++) {
    motor1.setSpeed(30);
    motor1.step(-distancia * 4);
    loop0();
    (motor1, LOW);
    Serial.print("[MotorApagado4  ]");
    Serial.print("[LedApagado4    ]");
    delay(1000);
    loop20();
    break;
  }
}
```

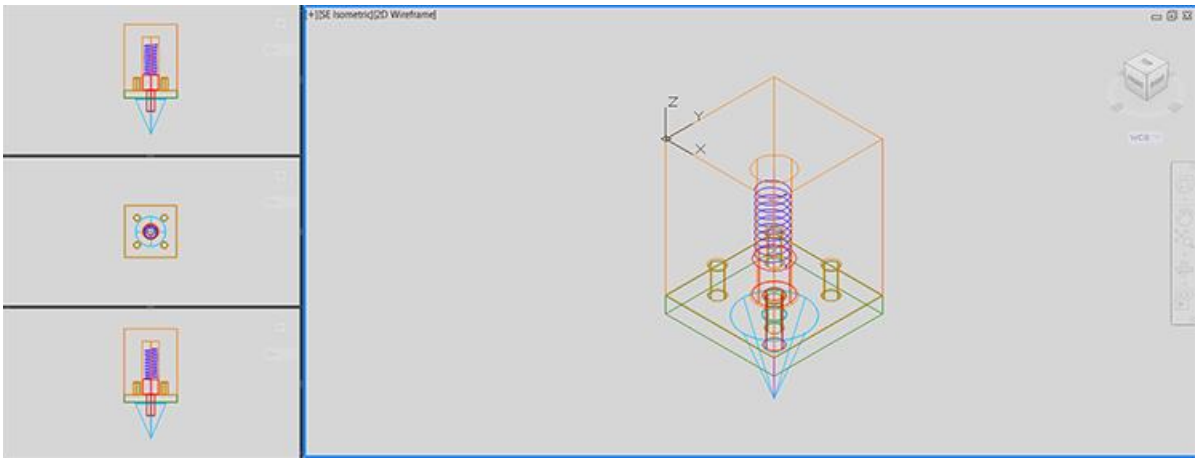
RESULTADOS DEL FUNCIONAMIENTO DEL BOTÓN PULSADOR1.

En el video que se anexa, “videoBotonPulsador” se puede observar que cuando la punta del cono del botón pulsador choca con la pared la regleta (prisma con rosca) sigue moviéndose hasta que el cono choca con la tapa del botón. Por lo que concluimos que la fuerza del motor sí es suficiente para que cuando el botón pulsador hace contacto con una pared el resorte se contrae y cuando la punta del botón pulsador no hace contacto con la pared el resorte llegue a su estado original.

Ya que en el prototipo final se van a utilizar varios botones pulsadores diseñamos otro dos botones pulsadores. El segundo botón pulsador que es el que queda en la parte de abajo del cuerpo geométrico, es el que va a chocar

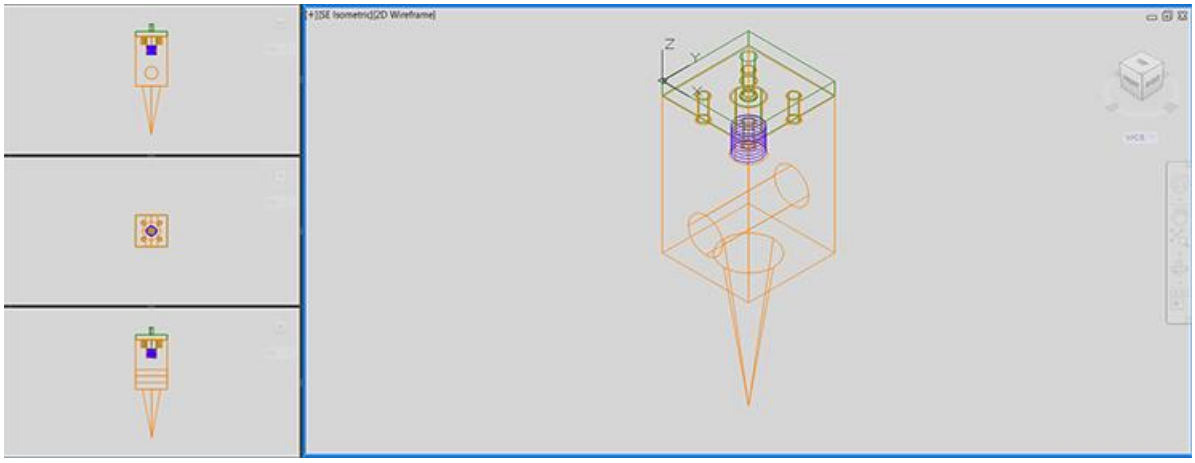
con el élitro del escarabajo. Por lo que el pulsador sobresale de la base solamente 0.3 mm para que sea muy rápida la marcación de las coordenadas X, Y. En el tercer botón pulsador, el pulsador queda en la parte de arriba para que se pueda pulsar manualmente hasta que la tapa choque con el cuerpo. En este caso también la separación entre el cuerpo y la tapa es de 0.3 mm.

BOTÓN PULSADOR 2



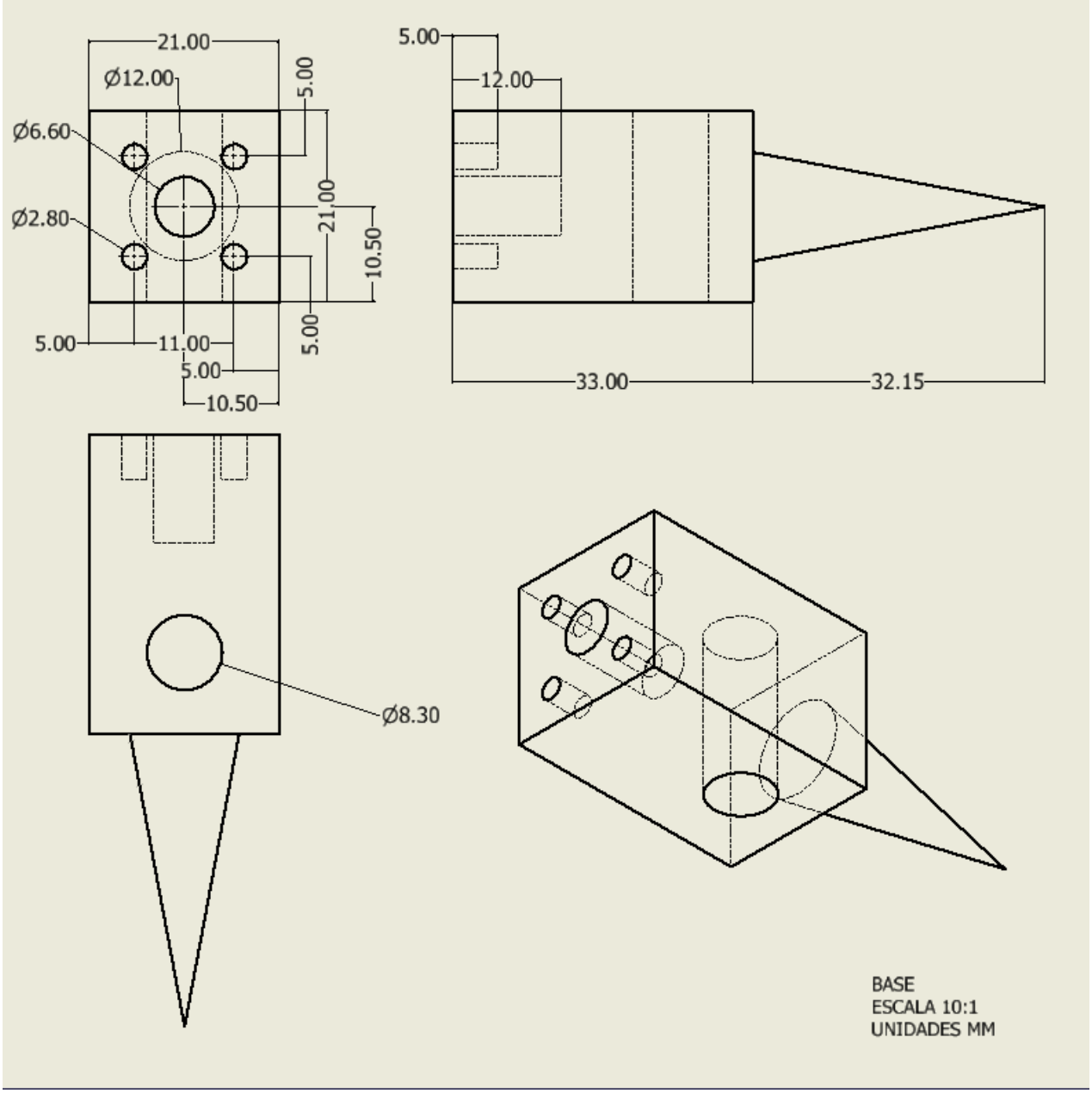
Detalle botón pulsador2.

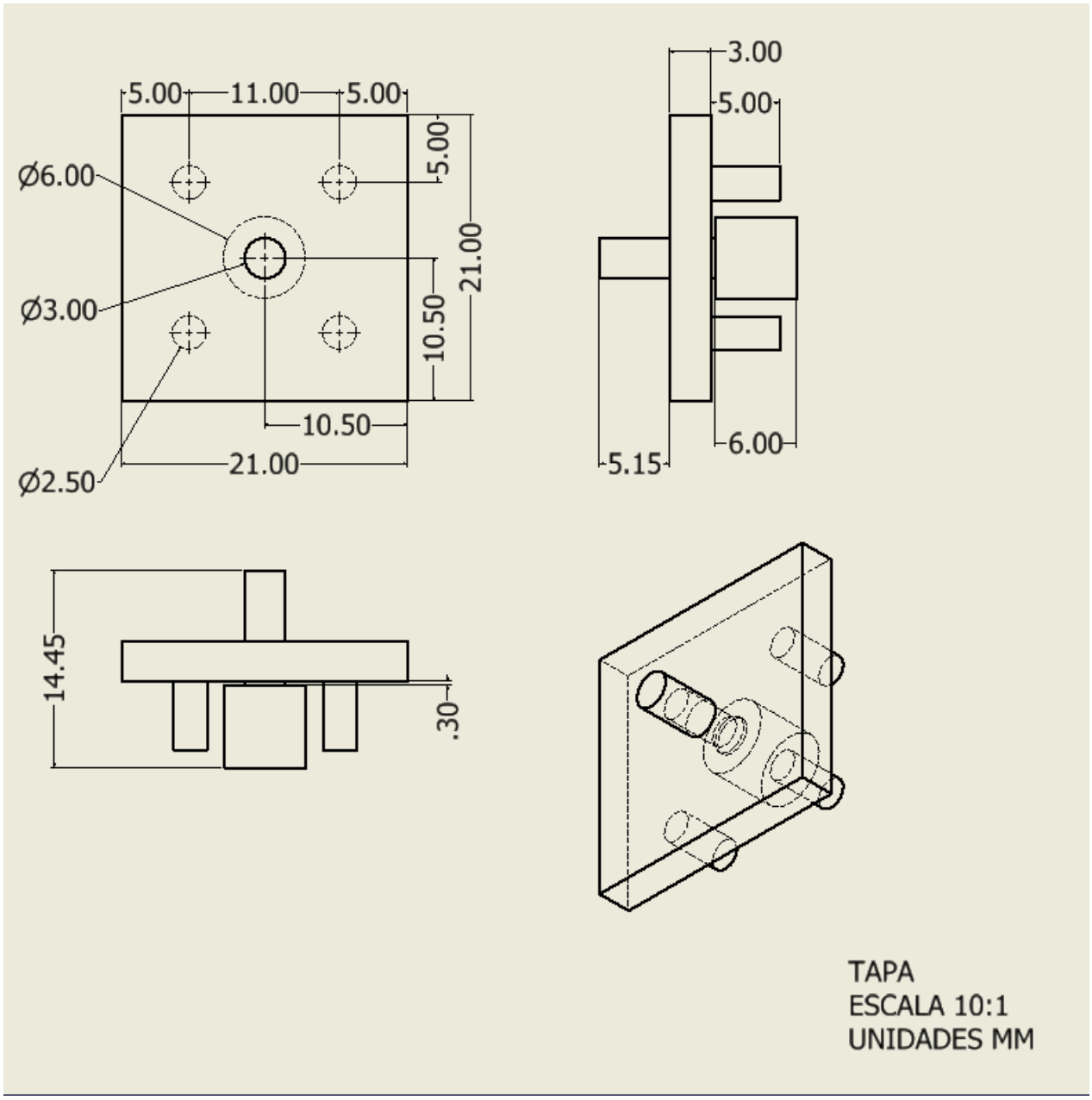
BOTÓN PULSADOR 3



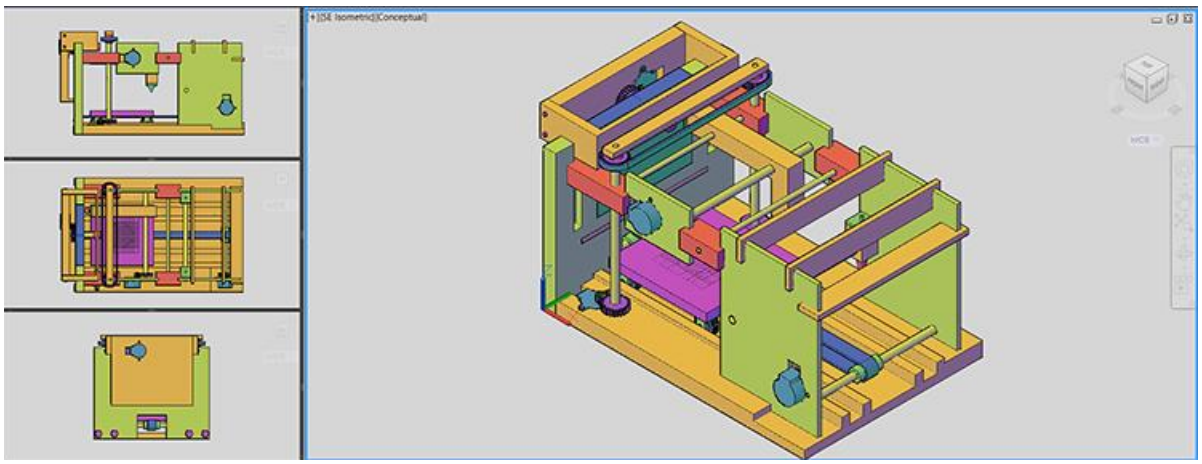
Detalle botón pulsador3.

PLANOS CONSTRUCTIVOS DE LAS PIEZAS DEL BOTÓN PULSADOR3

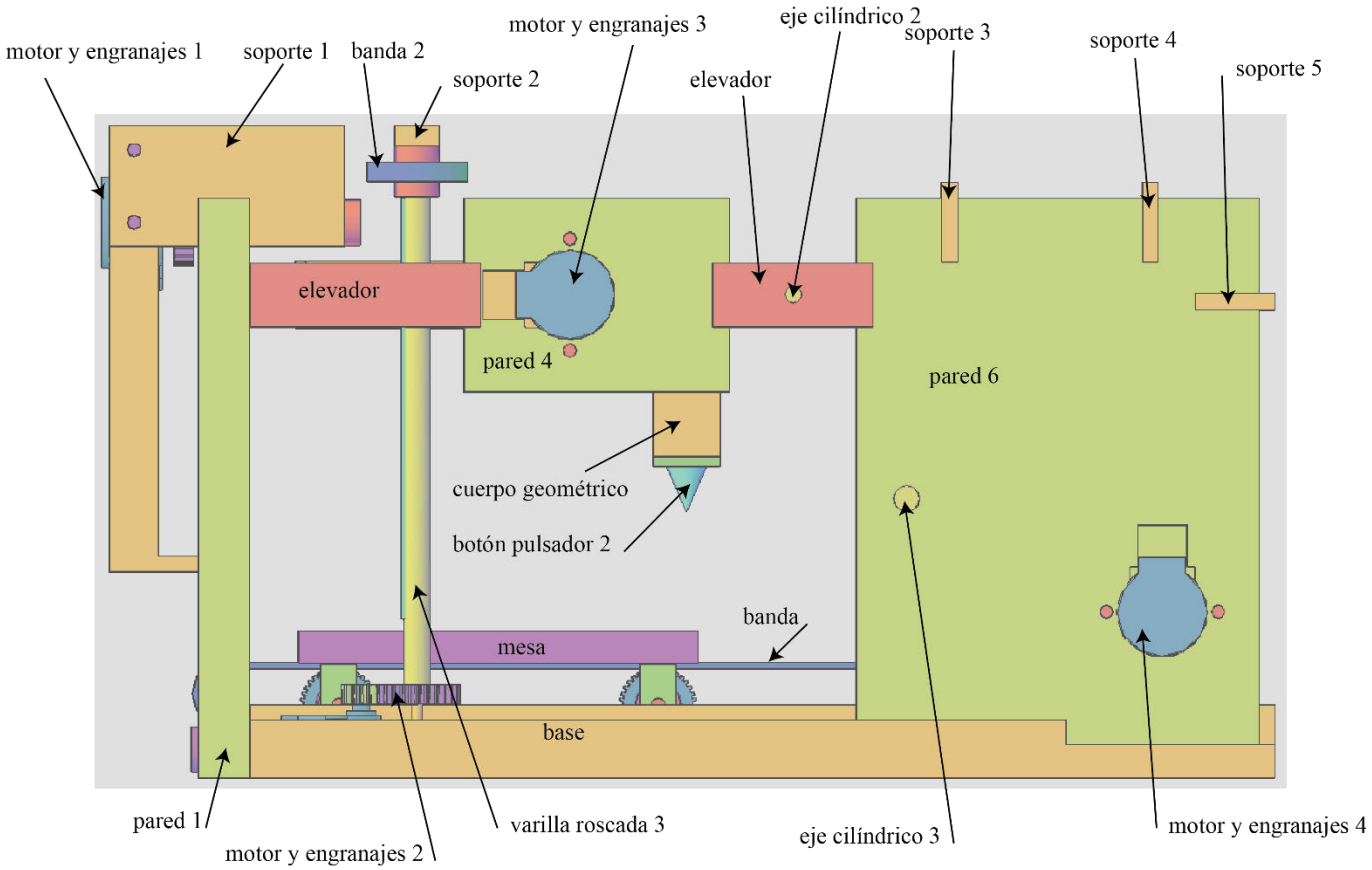



5.00
11.00
5.00
5.00
21.00
10.50
10.50
21.00
Ø6.00
Ø3.00
Ø2.50
3.00
5.00
5.15
6.00
14.45
.30
TAPA
ESCALA 10:1
UNIDADES MM

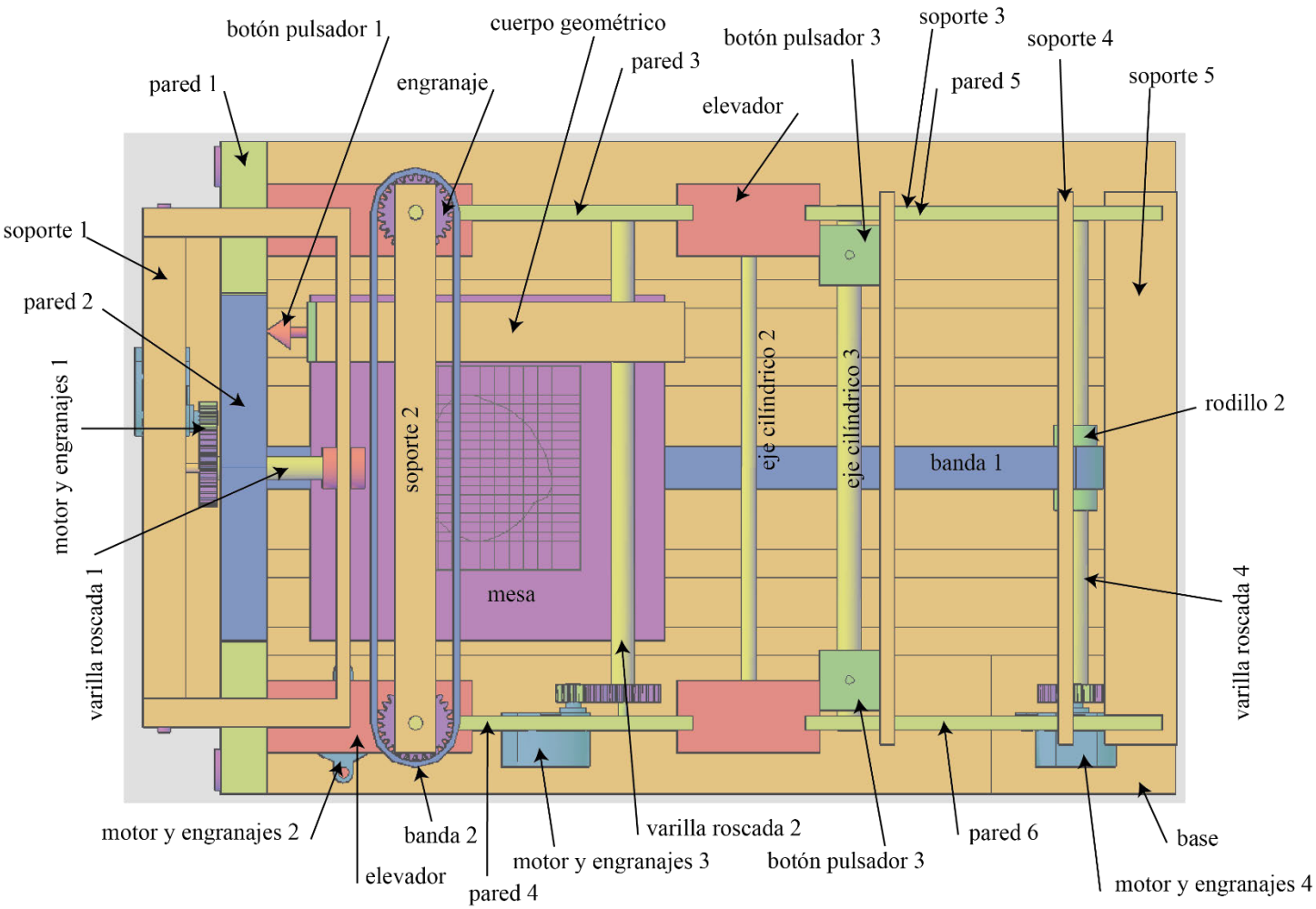
MODELADO DEL PROTOTIPO FINAL



Prototipo final.



Vista lateral del prototipo final.



Vista superior del prototipo final.

FUNCIONAMIENTO DEL PROTOTIPO FINAL

El cuerpo geométrico con los dos botones pulsadores estará arriba hasta su máxima altura y al centro de la mesa. La posición inicial de la mesa es la que se ve en la vista superior del prototipo final.
Los puntos X=0 y Y= 0 de la mesa están en el centro del lado izquierdo en la vista superior.

Paso 1. Se prender el motor 4, girará en sentido positivo para que la banda desplace a la mesa hacia la derecha. Se apaga el motor 4 cuando la mesa haya recorrido 125 mm que es el largo de la mesa.

Sobre la mesa se coloca el élitro del escarabajo a una distancia de 21 mm de la coordenada X de la mesa conservando la coordenada Y = 0.

Paso 2. Se prende el motor 4, girará en sentido negativo para que la banda desplace a la mesa cada 2.5 mm (medida que se trabajó en los modelos expuestos anteriormente) y se detenga para poder mover los botones pulsadores 3

alrededor del élitro del escarabajo y pulsarlos manualmente. En este caso se encontrarán las coordenadas X, Y del contorno de élitro.

Paso 3. Se detendrá el motor 4 cuando la mesa llegue a su posición original.

Paso 4. Se prende el motor 4, girará en sentido positivo hasta que llegue al último punto central del largo del élitro coincidiendo con el botón pulsador 2.

Paso 5. Se prende el motor 2, girará en sentido negativo, los elevadores bajarán hasta que el botón pulsador 2 apague el motor.

Paso 6. Se prende el motor 1, girará en sentido positivo para mover la pared 2 hacia la derecha hasta que el botón pulsador 1 apague el motor. En este caso se encontrará la coordenada Z.

Paso 7. Se prenderá el motor 1, girará en sentido negativo y se apagará cuando la pared 2 llegue a su posición original.

Paso 8. Se prende el motor 2, girará en sentido positivo hasta que el cuerpo geométrico llegue a su posición inicial, es decir hasta arriba.

Paso 9. Se prenderá el motor 2, girará en sentido negativo y se apagará cuando la banda desplace a la mesa 2.5 mm. Esta distancia es la misma que la del Paso 2.

Como el botón pulsador 2 está en el centro de la mesa, que es el punto de referencia se encontrarán las coordenadas Z de cada uno de los puntos primero hacia la derecha y después hacia la izquierda.
En este caso la distancia que se va a mover el motor 3 se indicará acorde a las distancias de las coordenadas X que se encontraron en el paso 2.
Suponiendo que la distancia que conviene es de 1.5 mm (véase Tabla 3, pág. 40).

Paso 10. Se prende el motor 3, girará en sentido positivo (hacia la derecha) y se apagará el motor cuando haya recorrido 1.5mm.

Paso 11. Se prenderá el motor 2, girará en sentido negativo los elevadores bajarán hasta que el botón pulsador 2 apague el motor.

Paso 12. Se prende el motor 1, girará en sentido positivo para mover la pared 2 hacia la derecha hasta que el botón pulsador 1 apague el motor. En este caso se encontrará la coordenada Z.

Paso 13. Se prenderá el motor 1, girará en sentido negativo y se apagará cuando la pared 2 llegue a su posición original.

Paso 14. Se prende el motor 2, girará en sentido positivo hasta que el cuerpo geométrico llegue a su posición inicial, es decir hasta arriba.

Paso 15. Se prenderá el motor 2, girará en sentido negativo y se apagará cuando la banda desplace a la mesa 2.5 mm. Esta distancia es la misma que la del Paso 2.

Una vez que se termina de encontrar las coordenadas Z del lado derecho, el cuerpo geométrico se regresará al centro para continuar hacia el lado izquierdo.
Se repite del Paso 10 al Paso 15 hasta que se tengan todas las coordenadas X, Y, Z de cada uno de los punto en una base de datos.

UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo
Unidad Cuajimalpa

Dina Rochman Beer <drochman@cua.uam.mx>

[CAD23] Submission Update ID 26

CAD23 <mail-sender@openconf.org> 9 de febrero de 2023, 17:39
Responder a: lap@cadxx.net
Para: drochman@cua.uam.mx, lap@cadxx.net

Submission ID: 26

Title: Development and 3D printing of a system with a displacement of increments of less than 3mm using Arduino UNO programming

Submission Type: Extended Abstract - consider for Journal Invitation

Author 1:
First Name: Dina
Last Name: Rochman
Organization: Autonomous Metropolitan University-Cuajimalpa
Country: Mexico
Email: drochman@cua.uam.mx

Contact Author: Author 1

Alternate Contact: dinarochman@gmail.com

Topic(s):
- Assembly design
- Rapid prototyping

Keywords: Robot, Spur Gear, Threaded Rod, Threaded Prism, Intervals, Arduino UNO.

Abstract: In this paper, we present the development of the prototype in 3D printing of one of the parts of the robot that is formed by a structure that supports a threaded rod, a system of spur gears, a threaded prism, and a prism with an arrow. We use a 28BYJ-48 Stepper Motor with a driver and Arduino UNO.
To verify that the 3D printing of the system formed by the threaded rod, the threaded prism, and the spur gears works correctly with the Arduino UNO programming, we carried out several tests to verify that interval increments of less than 3mm are possible.

Comments:

SE ANEXA EL ESCRITO DEL ABSTRACT QUE SE ENVIÓ PARA EL CONGRESO CAD´23.

Title:
Development and 3D printing of a system with a displacement of increments of less than 3mm using Arduino UNO programming

Authors:
Dina Rochman, drochman@cua.uam mx, Metropolitan Autonomous University - Cuajimalpa

Keywords:
Robot, Spur Gear, Threaded Rod, Threaded Prism, Intervals, Arduino UNO.

DOI: 10.14733/cadconfP.2023.xxx-yyy

Introduction:
The field of robotics is booming thanks to additive manufacturing. Among the 3D printed robots, we find humanoid robots, Kengoro, InMoov, Reachy, Aspir V2, Atlas, and others. Four-legged robots, Flexoskeletons, Open Cat, Standbeest, and others. Small-scale robots, Smart, Little Beat Ima Juno, Buddy, and others. The robotic arms, Little Arm V3, Niryo: Niryo One, Little Arm Big, and others. And the automata robots [3].

Each robot can be customized according to the needs. Hence, we begin this writing with the primary goal of the project we are carrying out at the Metropolitan Autonomous University - Cuajimalpa "Development of a 3D printing robot to move a threaded prism in increments of less than 3mm".

In this paper, we present the development of the prototype in 3D printing of one of the parts of the robot that is formed by a structure that supports a threaded rod, a system of spur gears, a threaded prism, and a prism with an arrow. We use a 28BYJ-48 Stepper Motor with a driver and Arduino UNO (Fig. 1).

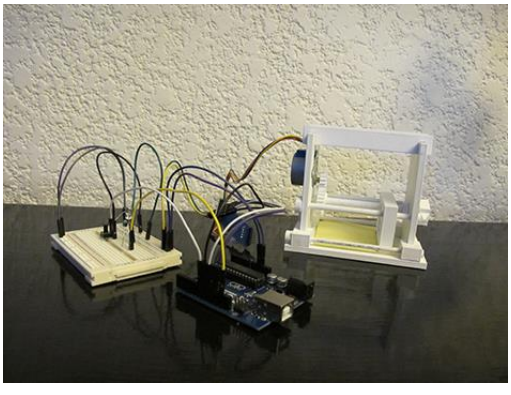


Fig. 1: Prototype in 3d printing.

To verify that the 3D printing of the system formed by the threaded rod, the threaded prism, and the spur gears works correctly with the Arduino UNO programming, we carried out several tests to verify that interval increments of less than 3mm are possible.

It is worth mentioning that stores sold only threaded rods made of steel, stainless steel, aluminum, brass, copper, titanium, plastic, nylon, PVC, or rubber of different diameters and lengths. Stores do not sell threaded rods printed on the 3D printer.

Main Sections:

It is a fact that robots are machines. A machine and a robot need to be programmed to perform tasks programmed for them. However, a robot can perform its tasks much more efficiently than a machine since it requires little or no human intervention [4].

The prototype presented in this article is classified as a robot since it will perform various tasks, move to the right/left with different increments of less than 3mm, and at different speeds that would be impossible for a human operator to perform [2].

Therefore, the main objective of the prototype is to analyze if the system of 3D printing of the threaded rod, the threaded prism, and the spur gears works correctly in the displacement of increments of less than 3mm through the Arduino UNO programming.

All the prototype was printed on the MakerBot Replicator 2 printer using PLA (polylactic acid) material and was modeled in the Fusion 360™ program.

Two threaded rods were modeled. The first is M8x1 (M = ISO metric; 8 = the basic major diameter of the thread, and 1= is the pitch), and the second is M8x1.25 (M = ISO metric; 8 = the basic major diameter of the thread; 1 = is the pitch, and 0.25 = is the length of the shift).

The spur gear system consists of a 10-tooth drive gear with a 10 mm pitch diameter attached to the stepper motor. And a 25-tooth driven gear with a 25 mm pitch diameter to reduce speed.

The transmission ratio of the gears and the speed reduction of the driven gear were found with the following formulas [1]:

$$i = \frac{Z_e}{Z_s} \tag{2.1}$$

$$W_s = \frac{Z_e * W_e}{Z_s} \tag{2.2}$$

The transmission ratio (i) of the gear is equal to 0.4. Ze = 10 and Zs = 25. (2.1). The driven gear (Ws) speed is 400 rpm. The velocity of the driver gear (We) is 1000 rpm (2.2).

Each threaded rod was printed together with a driven gear.

The threaded prism will move according to the threaded rod's thread pitch at the motor's speed.

The prism with an arrow is used to observe in which direction the threaded rod rotates and to count the number of turns the driven gear gives.

The Arduino Uno was used and programmed with different speeds (30, 20, and 10), with three motor steps (10, 4, and 2), and the motor rotation in both directions (- and +). Added a counter to check the steps of the motor. And a led that turns on and off to check when the motor turns on and off.

In the NetBeans IDE 8.2 program, we made the graphical interface. We put four buttons: Search port, Connect port, Engine and led ON/OF, and Disconnect port. A Combo Box to find the Port, which in this case is COM4. And a Text Field to write the number 0 or number 1. 0 is to turn on and 1 is to turn off the motor and the led (Fig. 2).

Fig. 2: Graphical interface.

Methodology:
The methodology used to reach the results of the proposed objective is the following (Fig 3):

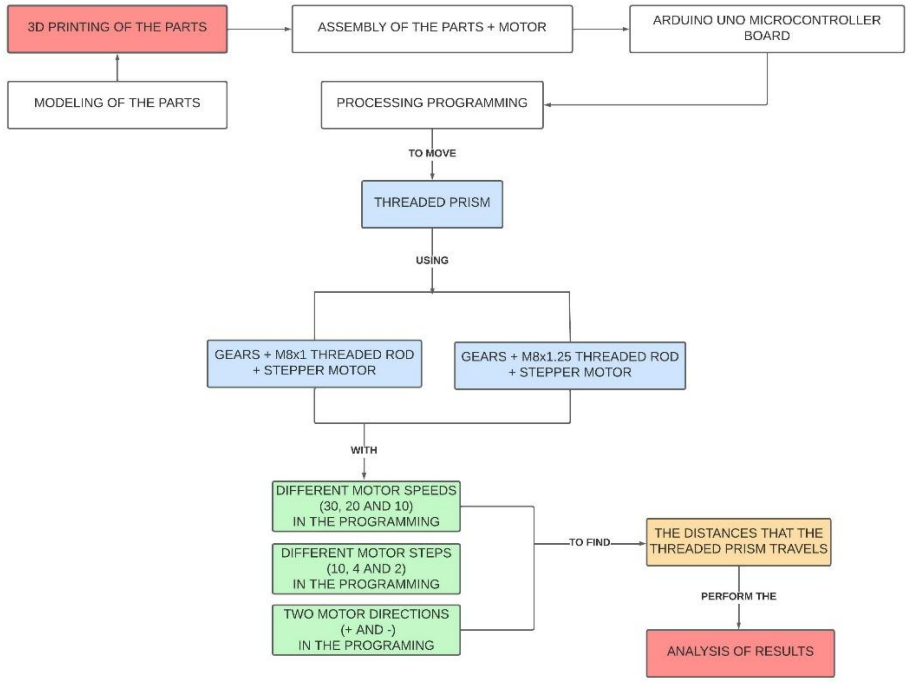


Fig. 3: Methodology.

Results:
Several analyzes were carried out to verify that the 3D printing of the system formed by the threaded rod, the threaded prism, and the spur gears works correctly and that the increments are less than 3mm through the programming of the Arduino UNO.

To carry out the analysis, in the programming of the Arduino UNO, the motor speed, the motor steps, the motor directions, and the delay are considered as indicated in Table 1:

	Case 1 Gear + M8x1 road	Case 2 Gear + M8x1.25 road
motor speed	30, 20 and 10	30, 20 and 10
motor steps	10, 4 and 2	10, 4 and 2
motor direction	+ and -	+ and -
delay	2000	2000

Tab. 1: Cases 1 and 2.

In both cases, the tests were carried out both roundtrip, that is, the direction of the motor in the Arduino UNO programming would be positive or negative (+ or -).

We check with the prism with an arrow the turns that the driven gear makes when the motor moves. In 2048 steps, the driving gear makes five turns, and the driven gear makes two turns.

With the counter, we corroborate that the change of steps is correct since, in one test, we programmed to have different motor steps 10, 4, and 2 in a single run.

In the programming, we use a delay of 2000 microseconds (two seconds) so that when the processor waits, we mark the line on the paper where the threaded prism stops (Fig. 4).

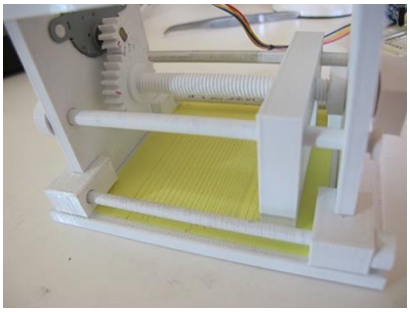


Fig. 4: Lines marked on the paper on the prototype.

Case 1

Due to residues left by 3D printing, the threaded prism was not initially coupled to the threaded rod. After several passes, the 3D printing residues were removed due to friction.

When we wrote in the Arduino UNO programming that the speed of the motor is 30 or 20, the thread of the prism skipped the thread pitch of the threaded rod. Finding that the distance that the threaded prism moved was variable and the line that was marked on the paper was not parallel to the prism.

We then continue with the motor speed of 10 in the Arduino UNO programming, with the different motor steps, and in the two directions (+ and -). In this case, the thread of the prism did not skip the pitch of the threaded rod.

The distance that we began to analyze was ten steps, the driving gear makes five turns, and the driven gear makes two turns to verify that it is the same distance as the threaded rod pitch, 2 mm.

We continue the analysis by marking the lines of the steps on the paper, first of 10, 4, and 2 steps at every ten intervals, and second, we alternate steps 10, 4, 4, and 2 at a single interval.

The lines marked on the paper were parallel to the threaded prism (Fig. 5).

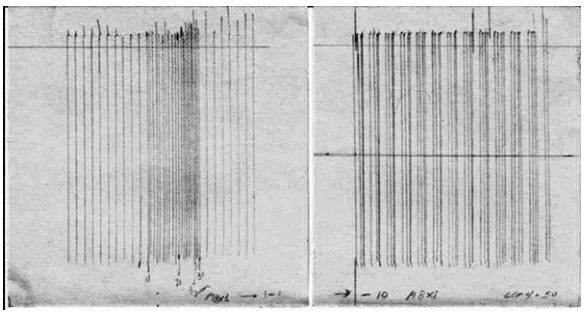


Fig. 5: Lines marked on the paper, case 1.

The distance that the threaded prism moves according to the number of steps would be as follows (Tab. 2):

Steps	10	9	8	7	6	5	4	3	2
Distance (mm)	2	1.8	1.6	1.4	1.2	1	0.8	0.6	0.4

Tab. 2: Steps and distances case 1.

Case 2

As in case 1, initially, the threaded strip did not couple to the threaded rod due to the residues left by the 3D printing. After several passes, the 3D printing residues were removed due to friction.

When programming the Arduino UNO with a motor speed of 30, 20, and 10, the threaded prism did not skip the thread pitch of the threaded rod.

The distance that we began to analyze was ten steps, the driving gear makes five turns, and the driven gear makes two turns to verify that it is the same distance as the threaded rod pitch, 2.5 mm.

We continue the analysis by marking the lines of the steps on the paper, first of 10, 4, and 2 steps in intervals of ten, and second, we alternate steps from 10 to 2 in intervals of three.

The lines marked on the paper were parallel to the threaded prism (Fig. 6).

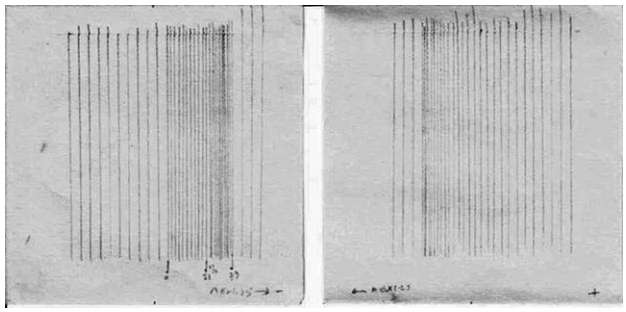


Fig. 6: Lines marked on the paper, case 2.

The distance that the threaded prism moves according to the number of steps would be as follows (Tab. 3):

Steps	10	9	8	7	6	5	4	3	2
Distance (mm)	2.5	2.25	2	1.75	1.5	1.25	1	0.75	0.5

Tab. 3: Steps and distances case 2.

Conclusions:

With the analysis that we carried out of the threaded rod system, the threaded prism, and the spur gears, we concluded that 3D printing is effective for moving any robot at distances of less than 3mm.

Acknowledgments*:*

I want to express my gratitude to Miriam Hernández Ramírez for her support in the analysis of the movement of the gears.

References:

[1] Bailey, 0.; Pickup, R.; Lewis, R.; Patient, P.: Modular course in technology, School Council, Oliver & Boyd, Great Britain, Edinburgh, Internet archive, ISBN 0050033867, 1988 Ninth Impression.
[2] Robots Science, https://www robotsscience.com, Difference between a robot and a machine.
[3] All3dp, https://all3dp.com, 3D printed robots.
[4] Tecno-Simple, https://tecno-simple.com, Difference between a robot and a machine.

Dina Rochman, http://orcid.org/0000-0001-8902-3513