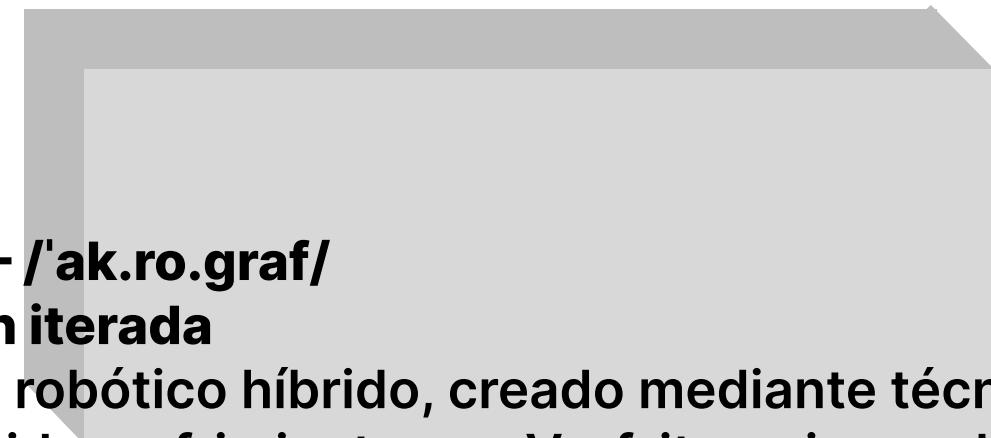


# AKROGRAFF



**Belinda Beethuizen Muga  
Prof: Joan Masdemont Fontás**



**AkroGraff v.1 — /'ak.ro.graf/**

**Una adaptación iterada**

**(n.)** Dispositivo robótico híbrido, creado mediante técnicas de prototipado rápido, sufrimiento con Vref, iteraciones de 3D y una cantidad estadísticamente preocupante de piezas 3D; diseñado para transformar coordenadas en trazos, vea :— frustración en aprendizaje.

## ÍNDICE

1. Motivación y objetivo del proyecto
2. Lista de componentes
3. Conexiones electrónicas y cableado del sistema
4. Software
5. Conclusiones

## Motivación y objetivo del proyecto

Desde el inicio se buscaba desarrollar un brazo robotizado capaz de realizar dibujos de manera autónoma. La elección de un brazo robótico articulado en lugar de un sistema cartesiano tradicional responde al interés por explorar una estructura con varios grados de libertad, más cercana a manipuladores reales, y también a la intención de mejorar la suavidad y limpieza del trazo respecto a otros proyectos amateurs existentes. Se quería estudiar cómo la estructura mecánica, la electrónica de control y la estabilidad del movimiento afectan al resultado final del dibujo.

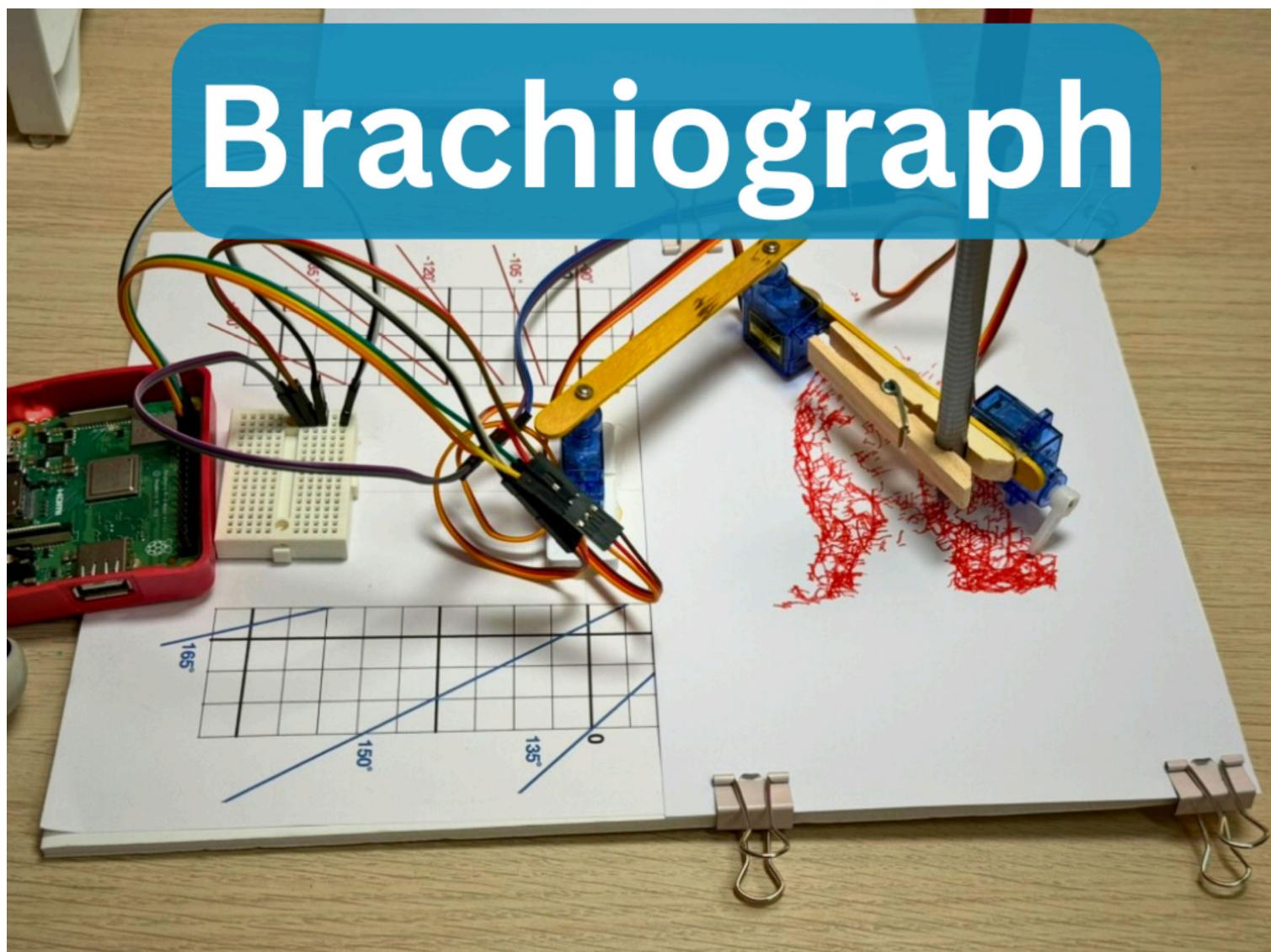


El objetivo principal consistió en construir un prototipo funcional que pudiera mover un portaplumas en el plano, levantándolo y bajándolo según fuera necesario, y generando trayectorias precisas a partir de instrucciones programadas. Esto implicó integrar tres tipos de movimiento: una base rotatoria actuando como eje principal, un primer brazo articulado encargado de levantar y bajar la estructura, y un segundo brazo que sostiene el portaplumas y define el movimiento hacia adelante-atrás y lateral. Durante el diseño se aplicó un criterio iterativo: se probaron múltiples formas de ensamblar los brazos, se reajustaron las medidas, se corrigieron holguras y se rediseñaron piezas hasta obtener un funcionamiento estable. Las longitudes finales de los brazos —231.93 mm para el brazo principal y 12.52 mm para el secundario (ambos con un grosor de 26 mm)— se decidieron tras varias pruebas, aunque se reconoce que estas dimensiones podrán modificarse en futuras iteraciones del proyecto.

En todo el diseño mecánico se ha comprobado que las piezas impresas en 3D requieren una holgura aproximada de 0.1 mm para que los ensamblajes encajen correctamente, lo cual obligó a reiterar muchas piezas, especialmente la base y los mecanismos de unión. También se ha empleado un reductor universal montado con tornillería M3 y M5, firmemente fijado para evitar vibraciones que afectarían negativamente a la calidad del trazado. La plataforma circular tiene un radio de 102 mm y una altura total de 80.35 mm.

Para definir el alcance del proyecto se revisaron propuestas existentes, en particular el BrachioGraph de Daniele Procida, un “pen-plotter” extremadamente sencillo basado en tres servomotores: uno para levantar el lápiz y los otros dos para moverlo en X e Y. Este proyecto destacó como referencia inicial debido a su simplicidad y facilidad de construcción, pero también mostró limitaciones significativas que motivaron una línea de diseño distinta.

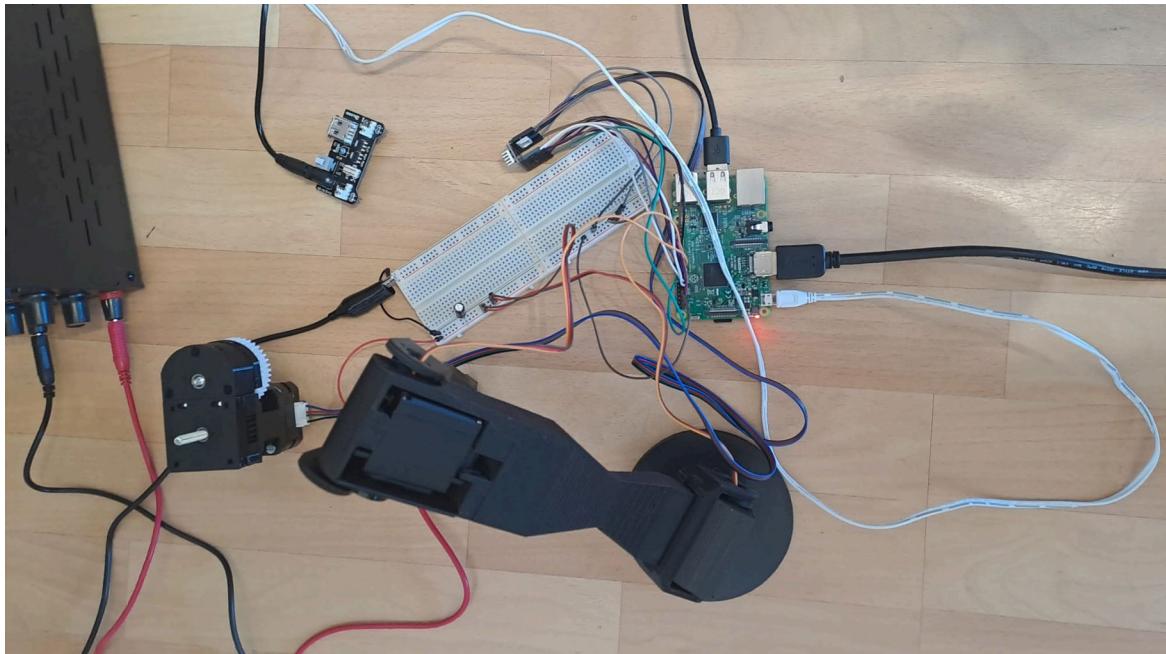
El BrachioGraph utiliza brazos muy ligeros y un mecanismo de sujeción de lápiz que genera interferencias, ruido y vibraciones. Esto, sumado a la naturaleza abrupta del movimiento de los servos, provoca que el dibujo final presente desviaciones apreciables respecto a la imagen de referencia. A partir de estas observaciones se decidió mejorar la estabilidad sustituyendo los dos servos principales por un motor paso a paso NEMA17 montado sobre una base giratoria con reducción mecánica, logrando un movimiento más controlado y suave. El uso de un stepper permitió además utilizar microstepping, evitando saltos bruscos y permitiendo que las trayectorias fueran más uniformes.



Se integraron dos servos MG996R únicamente en los ejes articulados, donde su rapidez y fuerza eran más importantes que la precisión absoluta. Esta configuración híbrida entre stepper y servos permite equilibrar precisión, velocidad y simplicidad mecánica. Además, el proyecto se alimentó de diversos recursos procedentes de bibliotecas abiertas como Thingiverse, desde soportes para motores hasta cajas y bases que pudieron modificarse. Aunque disponer de modelos previos aceleró el proceso, gran parte del trabajo consistió en adaptar, medir y rediseñar estos modelos, ya que muchos no encajaban exactamente en el brazo o requerían tolerancias específicas. Esto implicó un trabajo extenso con mallas 3D, conversión a formatos utilizables y reestructuración de geometrías en programas como FreeCAD, lo cual requirió un aprendizaje adicional y prolongó significativamente la fase de diseño.

En conjunto, el proyecto parte de un análisis crítico de diseños existentes, toma elementos de ellos y los reinterpreta mediante un enfoque propio donde la prioridad es mejorar la calidad del trazado y construir una plataforma mecánicamente más estable y reutilizable para experimentos futuros en cinemática, planificación de trayectoria y control.

## Lista de componentes



Para la construcción del brazo robotizado se han seleccionado y adaptado diferentes elementos mecánicos, electrónicos y estructurales, cada uno cumpliendo una función específica dentro del conjunto. La elección de estos componentes no fue arbitraria; se basó en la necesidad de garantizar precisión, estabilidad, facilidad de control y capacidad

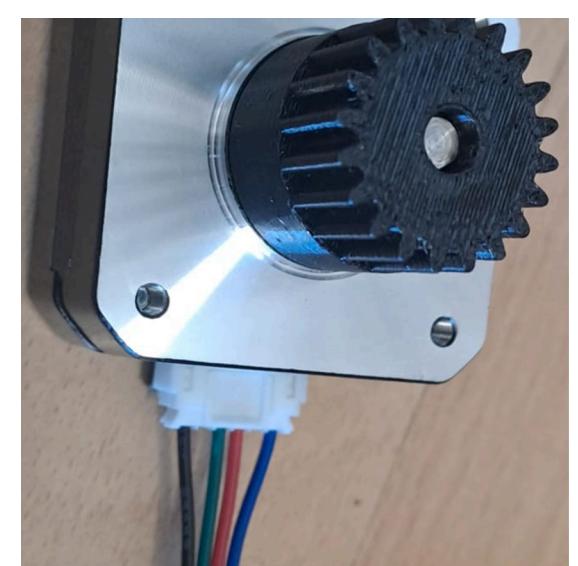
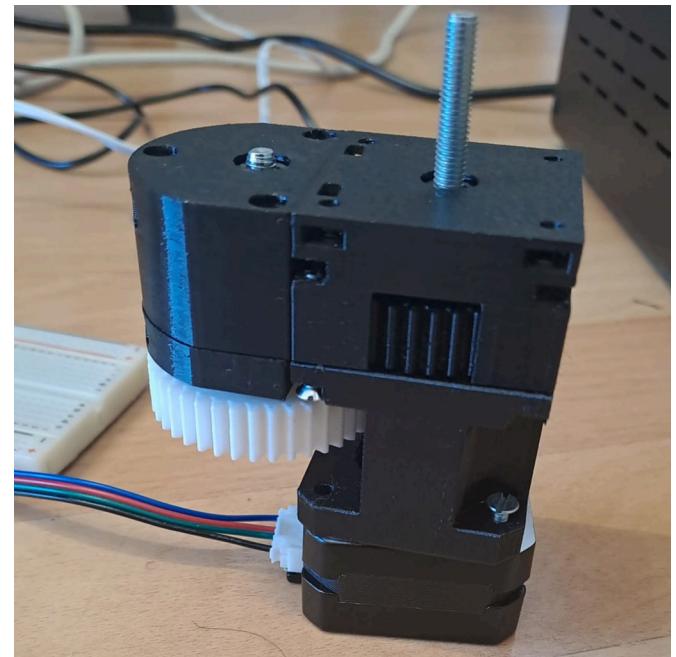
de soportar las cargas mecánicas generadas por el movimiento. Durante el desarrollo se han enfrentado varias dificultades derivadas tanto de la compatibilidad entre piezas como de la propia naturaleza de los motores utilizados.

### Base rotatoria y motor paso a paso NEMA17

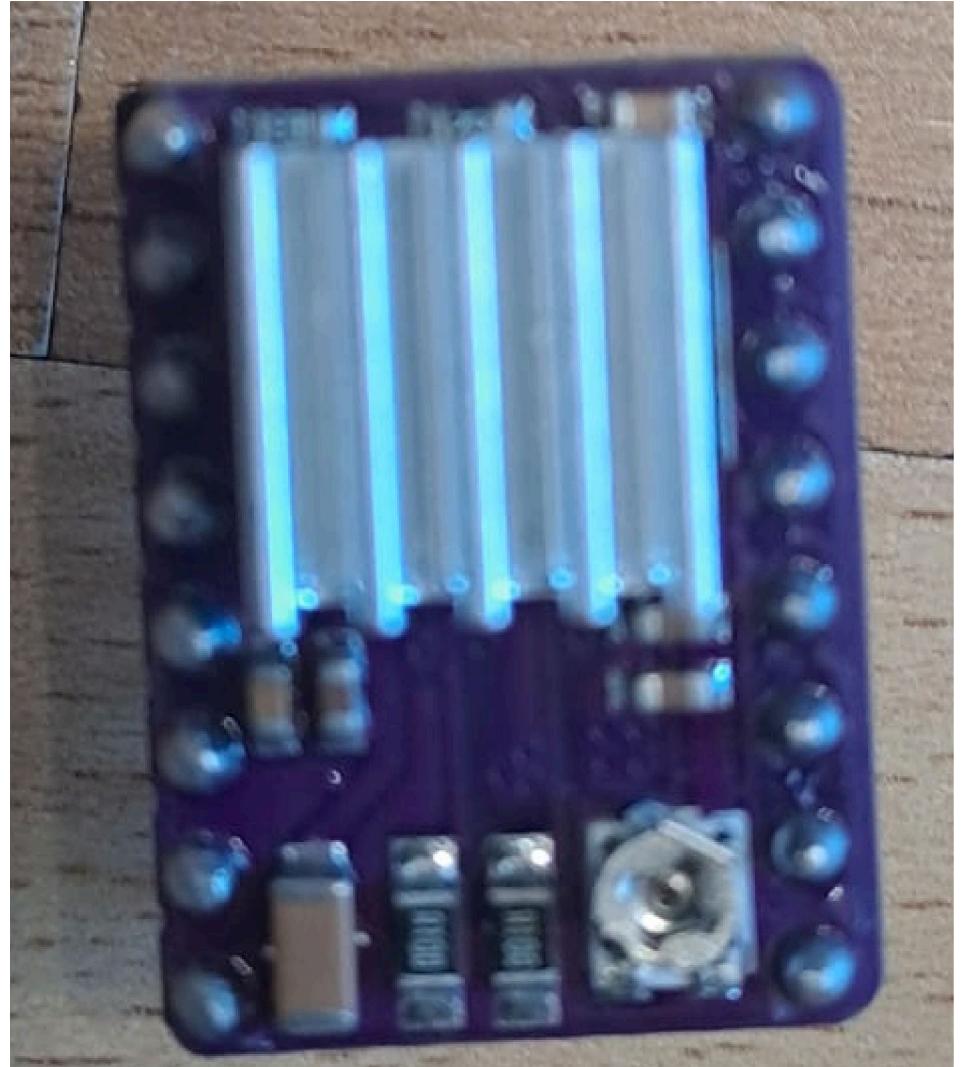
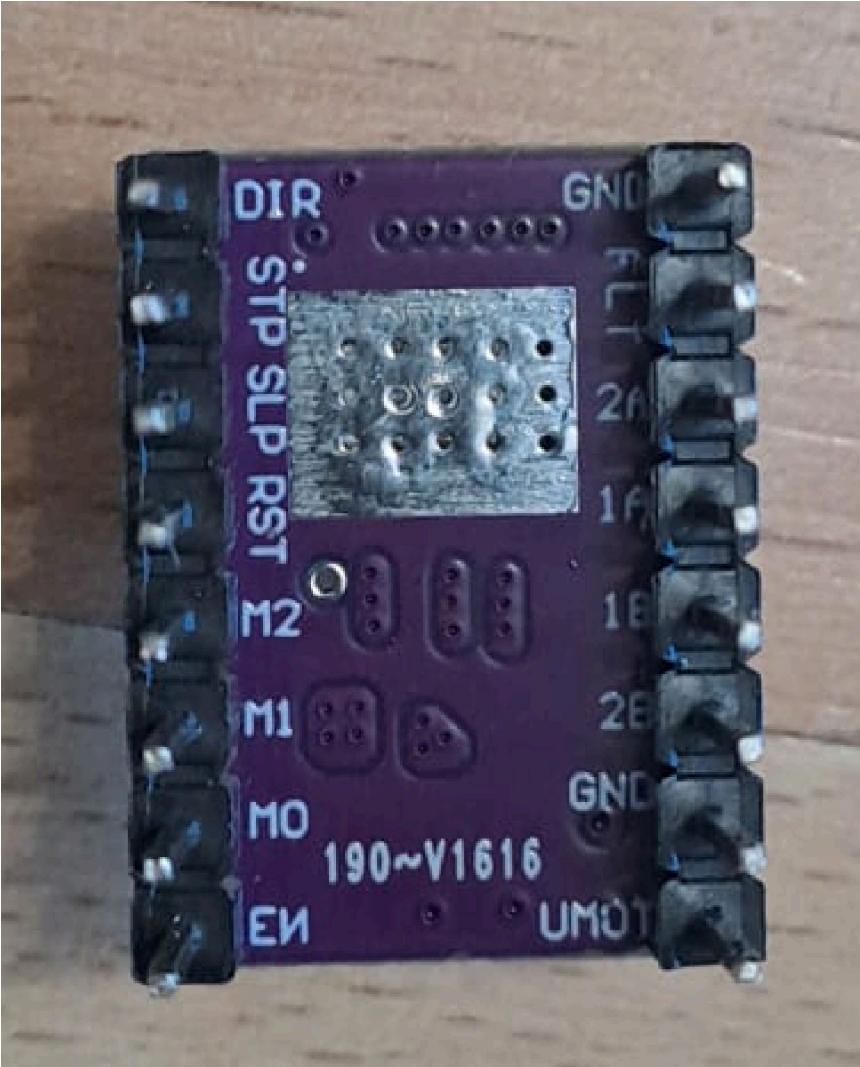
La base del brazo constituye el elemento estructural principal y ha sido diseñada para garantizar la máxima estabilidad. Sobre esta base se ha instalado un motor NEMA17 de tipo bipolar, que dispone de cuatro pines, correspondientes a las dos bobinas internas del motor (A1-A2 y B1-B2). Estos pines no deben interpretarse como conectores ya ordenados; se ha comprobado mediante mediciones de continuidad que el orden del cableado no siempre coincide con el que requiere el driver. Para evitar vibraciones, par intermitente y pérdida de pasos, se han identificado los pares correctos y posteriormente se han reubicado los pines en el conector, garantizando la correspondencia exacta con las salidas del driver DRV8825: A1 → 1A, A2 → 2A, B1 → 1B, B2 → 2B.

La base tiene una altura total de 80.35 mm, sobre la cual descansa una plataforma circular de 102 mm de radio. Estas dimensiones surgieron de un proceso iterativo de ajustes, ya que cada cambio en la estructura superior exigía modificar la base. A ello se suma la necesidad de dejar holguras adecuadas: se ha comprobado que las piezas impresas en 3D requieren una tolerancia aproximada de 0.1 mm para que los ensamblajes encajen y no generen fricción excesiva o deformación.

Para suavizar el movimiento del motor y evitar saltos bruscos, se ha instalado un reductor universal, firmemente fijado mediante tornillería M3 y M5. La rigidez de esta unión es fundamental para evitar que pequeñas vibraciones se traduzcan en movimientos no deseados en el lápiz, lo que afectaría directamente la calidad del dibujo.



## Driver DRV8825 y alimentación del motor paso a paso



El motor NEMA17 no puede alimentarse directamente desde la fuente debido a las variaciones de corriente que requiere. Por ello se ha empleado un driver DRV8825, encargado de controlar el flujo de corriente hacia cada bobina mediante microstepping. Este driver incluye los pines STEP, DIR, ENABLE, SLP, RST, FLT y los pines de microstepping M0, M1 y M2.

Para evitar estados flotantes, estos últimos se han fijado a nivel bajo mediante pines GPIO configurados en LOW, ya que cuando permanecían sin referencia el motor presentaba comportamientos erráticos, cambios inesperados de microstepping o vibraciones.

Se ha incorporado un condensador electrolítico ( $100 \mu\text{F}$ , 35–50 V) conectado entre VMOT y GND, situado lo más cerca posible del driver. Este componente actúa como amortiguador frente a los picos de tensión que se generan al conmutar las bobinas del motor; sin él, el sistema presentaba reinicios y colapsos momentáneos del voltaje.

El ajuste del Vref, necesario para regular la corriente máxima del motor, presentó múltiples dificultades. El potenciómetro del driver mostraba un comportamiento anómalo: durante casi todo el recorrido entregaba 0 mV y en un punto muy específico saltaba abruptamente a valores superiores a 2.5–3 V. Este fenómeno impedía realizar el ajuste estándar. Para resolverlo, se ha utilizado un método alternativo, monitorizando el consumo de corriente directamente en la fuente (fuente de alimentación / fuente de poder / fuente de 12 V). Se ha ajustado el potenciómetro lentamente hasta obtener un consumo y un comportamiento adecuados en el motor: par suficiente sin sobrecalentamiento.

Este proceso requirió numerosas pruebas y ha sido una de las partes más complicadas del proyecto.

## Servomotores MG996R para el movimiento de los brazos

Los brazos articulados emplean dos servomotores MG996R, uno como articulación principal (elevación del primer brazo) y otro como articulación final que sostiene el portaplumas. Estos servos disponen de tres conexiones: señal PWM, VCC y GND. Se alimentan a 5–6 V y se ha optado por usar una alimentación independiente derivada de la misma fuente de 12 V, pero bajada mediante un regulador. No se ha unificado esta alimentación con el motor paso a paso porque:

los servos requieren 5–6 V; el NEMA usa 12 V; cada tipo de motor tiene necesidades eléctricas distintas;

el driver del stepper y los servos necesitan filtrado independiente;

solo se dispone de un condensador para el driver del NEMA, lo que imposibilita añadir la protección necesaria a múltiples drivers.



El uso de los servos se ha limitado a movimientos donde su rapidez y fuerza compensan sus limitaciones de precisión. En este sentido, combinarlos con un motor paso a paso permite aprovechar lo mejor de ambos: precisión en la base, fuerza y velocidad en las articulaciones.

## Brazos impresos en 3D y portaplumas

El brazo principal tiene unas dimensiones finales de  $64.61 \times 231.93 \times 26.44$  mm, mientras que el segundo brazo mide  $64.61 \times 12.52 \times 26$  mm. Estos valores son el resultado de múltiples iteraciones, ya que cada cambio en la articulación o en la forma de acoplar los motores obligaba a modificar las piezas anteriores.

Durante el diseño se ha trabajado mayoritariamente con geometrías tipo mesh, lo que ha dificultado su manipulación en programas como FreeCAD, especialmente a la hora de generar animaciones o modelos paramétricos. Por esta razón, gran parte de las piezas tuvieron que reajustarse manualmente.

El portaplumas, aunque funcional, presenta limitaciones: el orificio hexagonal previsto para alojar distintos tipos de bolígrafos no sujetá correctamente el instrumento porque la holgura necesaria para introducirlo reduce su firmeza. El tornillo lateral destinado a fijarlo tampoco logra ejercer presión suficiente sobre el cuerpo del lápiz. Esta pieza se considera experimental y deberá rediseñarse en iteraciones futuras.





## Conexiones electrónicas y cableado del sistema

El sistema combina un motor paso a paso NEMA17, un driver DRV8825, dos servomotores MG996R y una Raspberry Pi como unidad de control. Dado que todos los elementos tienen requisitos eléctricos distintos, el cableado se ha realizado siguiendo un enfoque modular y con especial atención a la protección contra fluctuaciones de tensión.

### Alimentación general del sistema

Se ha empleado una única fuente de alimentación de 12 V, desde la cual se distribuye energía a los distintos componentes:

12 V directos → Driver DRV8825 → Motor NEMA17

12 V → Regulador DC-DC → 5–6 V → Servos MG996R

Debido a que solo se dispone de un condensador electrolítico, ha sido imposible unificar los tres servos y el stepper sobre una misma placa o alimentación común debidamente filtrada. Por ello, cada sistema se ha cableado por separado para evitar picos de tensión o caídas de corriente que afectaran al resto de motores.

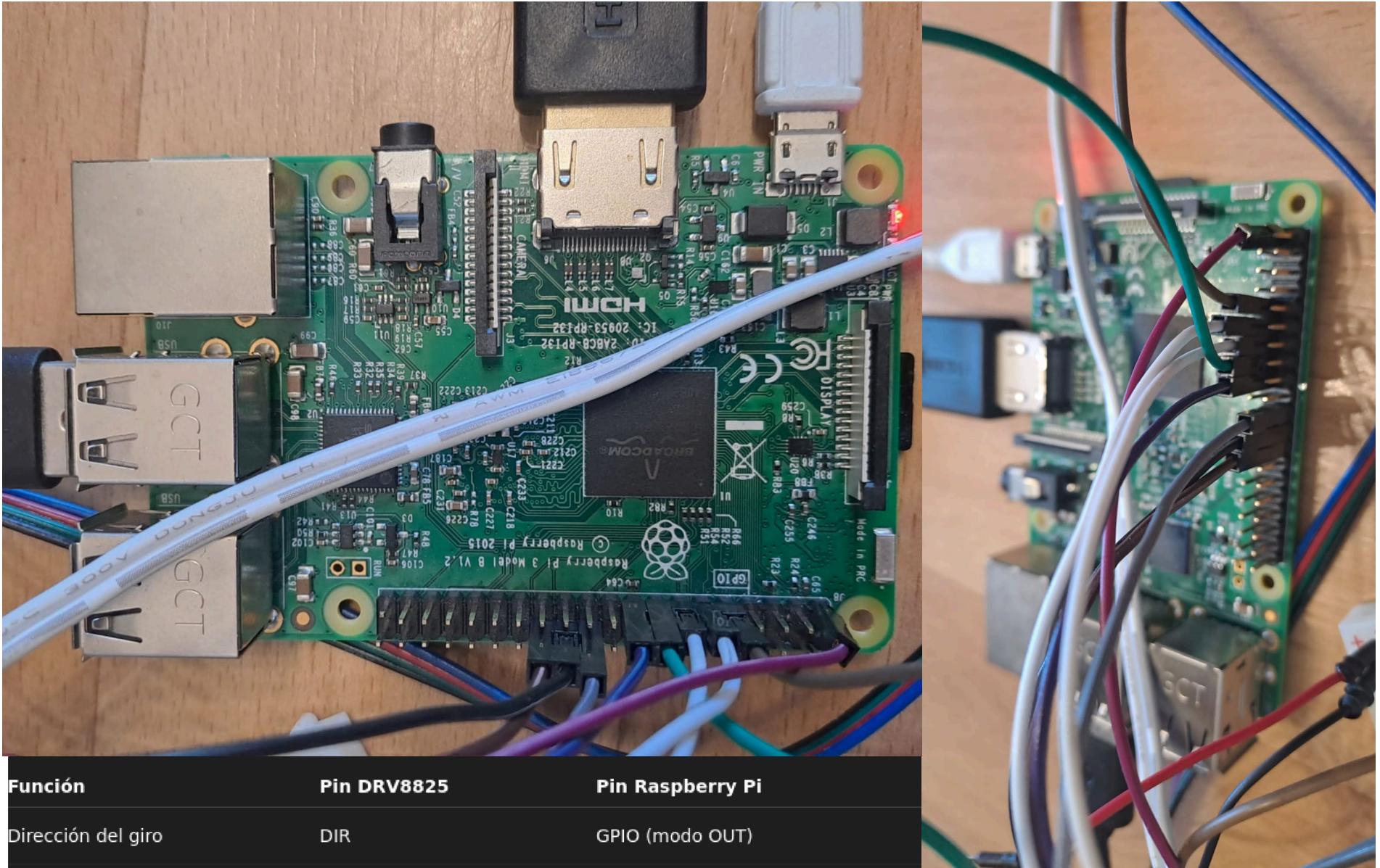
### Driver DRV8825 → Motor NEMA17

El NEMA17 empleado es bipolar, con dos bobinas independientes.

Tras identificar los pares correctos con el multímetro, las conexiones finales han quedado así:

A1 → 1A  
A2 → 2A  
B1 → 1B  
B2 → 2B

Esto garantiza que las bobinas están energizadas en el orden adecuado para lograr un movimiento estable y evitar vibraciones o inversión involuntaria del giro.



Función	Pin DRV8825	Pin Raspberry Pi
Dirección del giro	DIR	GPIO (modo OUT)
Avance de pasos	STEP	GPIO (modo OUT)
Activación	EN	GPIO (forzado a LOW)
Reset	RST	GPIO (forzado a HIGH o puenteado)
Sleep	SLP	GPIO (forzado a HIGH o puenteado)
Microstepping	M0, M1, M2	GPIO en LOW para modo completo

No se han alimentado desde los 5 V de la Raspberry porque:

- consumirían demasiado amperaje,
- generarían caídas de tensión,
- podrían resetear la Raspberry o causar daños.

Para evitar ruido eléctrico y garantizar que el driver no fluctuara entre configuraciones de microstepping, los pines M0, M1 y M2 se han fijado manualmente a nivel LOW mediante GPIO configurados como salidas. Mientras estuvieron flotando, el motor vibraba, producía pasos irregulares o cambiaba repentinamente de resolución.

#### 4.4. Condensador de protección

Se ha instalado un condensador electrolítico de 100  $\mu\text{F}$  entre:

- VMOT (positivo del driver)
- GND (tierra del driver)

Este condensador se ha colocado físicamente pegado al módulo, lo más cerca posible de los pines VMOT–GND, con el fin de:

- amortiguar picos de tensión generados por el motor,
- proteger el DRV8825 de caídas bruscas de voltaje,
- evitar reinicios momentáneos del sistema.

Sin esta protección, el motor provocaba inestabilidad eléctrica en todo el circuito.

#### 4.5. Servo MG996R → Raspberry Pi y alimentación

Cada servo tiene tres cables:

- Rojo → 6 V
- Marrón/Negro → GND
- Amarillo/Naranja → Señal PWM (GPIO Raspberry Pi)

Todos los grounds (GND) se han unificado entre sí para establecer una referencia común, imprescindible para que las señales PWM sean estables.

## Software

Para el control de los servomotores se empleó la librería pigpio, ya que permite generar señales PWM por hardware con gran estabilidad, algo esencial para el funcionamiento preciso de los MG996R. El script desarrollado establece una función que convierte un ángulo entre 0° y 180° en un pulso comprendido entre 500 y 2500 µs, rango estándar aceptado por este tipo de servos. A partir de esa función se programó un movimiento repetitivo en el que ambos servos realizan barridos completos en sentidos opuestos durante ocho ciclos, lo que permitió comprobar su comportamiento, la respuesta mecánica y la ausencia de vibraciones o bloqueos.

```
try:
    # Repite el movimiento completo 8 veces.
    for _ in range(8):

        # Barrido desde 0° hasta 180°.
        # Avanza en incrementos de 5 grados.
        for angle in range(0, 181, 5):
            set_angle(serv01, angle)      # primer servo sube
            set_angle(serv02, 180 - angle) # segundo servo baja (movimiento opuesto)
            time.sleep(0.02)             # suaviza el movimiento

        # Barrido desde 180° hasta 0°.
        for angle in range(180, -1, -5):
            set_angle(serv01, angle)
            set_angle(serv02, 180 - angle)
            time.sleep(0.02)

    finally:
        # Se detienen los pulsos para evitar vibraciones.
        pi.set_servo_pulsewidth(serv01, 0)
        pi.set_servo_pulsewidth(serv02, 0)

        # Se desconecta pigpio correctamente.
        pi.stop()
```

```
import RPi.GPIO as GPIO
import time

# -----
# GPIO PIN DEFINITIONS (BCM)
# -----
DIR = 27      # Pin que determina el sentido de giro del motor.
STEP = 17     # Pin que envía pulsos de avance.
ENABLE = 22    # Pin de habilitación del driver (LOW = habilitado).

M0 = 25       # Pines que configuran el microstepping (M0-M2).
M1 = 8
M2 = 7

# -----
# MOTOR SETTINGS
# -----
SPR = 200      # Número de pasos por revolución en full-step (1.8°).
REVS = 3        # Número de revoluciones que hará en cada dirección.
MIN_DELAY = 0.0004 # Velocidad máxima (poco delay = más rápido).
MAX_DELAY = 0.003   # Velocidad mínima (inicio/parada).
ACCEL_STEPS = 40   # Número de pasos dedicados a acelerar y desacelerar.
```

```
def accel_step(delay_start, delay_end, steps, direction):
    """
    Función que genera una aceleración o deceleración progresiva.
    delay_start = delay inicial (lento)
    delay_end   = delay final (rápido)
    steps       = cuántos pasos se usan para acelerar
    """
    GPIO.output(DIR, direction)

    # Cambio gradual de velocidad por paso
    delta = (delay_end - delay_start) / steps

    for i in range(steps):
        delay = delay_start + delta * i
        GPIO.output(STEP, GPIO.HIGH)
        time.sleep(delay)
        GPIO.output(STEP, GPIO.LOW)
        time.sleep(delay)
```

```
import pigpio
import time

# Connect to pigpiod
# Se conecta al daemon pigpio, necesario para generar PWM por hardware.
pi = pigpio.pi()
if not pi.connected:
    print("Failed to connect to pigpiod")
    exit()

# Define servo pins
# Pines GPIO donde están conectados los servos.
serv01 = 10 # GPIO pin for first servo
serv02 = 9  # GPIO pin for second servo

# Function to move servo to a specific angle (0-180)
def set_angle(pin, angle):
    # Convierte un ángulo (0-180°) en un pulso entre 500 y 2500 µs.
    # Esto corresponde aproximadamente al rango mecánico del MG996R.
    pulse = 500 + (angle / 180.0) * 2000
    pi.set_servo_pulsewidth(pin, pulse)
```

```
-----  
-----  
to CW, luego CCW  
ection, name in [(GPIO.HIGH, "CW"), (GPIO.LOW, "CCW")]:  
  
    print(f"\nMoving {name} for {REVS} revolutions...")  
  
    in range(REVS):  
  
        # Fase de aceleración
        accel_step(MAX_DELAY, MIN_DELAY, ACCEL_STEPS, direction)  
  
        # Velocidad constante
        run_constant_speed(  
            MIN_DELAY,  
            SPR - (2 * ACCEL_STEPS), # parte central sin aceleración  
            direction  
        )  
  
        # Fase de desaceleración
        accel_step(MIN_DELAY, MAX_DELAY, ACCEL_STEPS, direction)
```

```
time.sleep(0.4) # pausa entre giros  
  
print("Test complete.")  
  
GPIO.output(ENABLE, GPIO.HIGH) # Apaga el driver  
cleanup() # Limpia los pines GPIO  
GPIO cleaned up. Driver disable (↓)
```

```

# -----
# SETUP
# -----
GPIO.setmode(GPIO.BCM)
GPIO.setup([DIR, STEP, ENABLE, M0, M1, M2], GPIO.OUT)

# Set full-step mode
# Los pines M0, M1 y M2 en LOW activan modo full-step.
GPIO.output(M0, GPIO.LOW)
GPIO.output(M1, GPIO.LOW)
GPIO.output(M2, GPIO.LOW)

# Disable driver initially
GPIO.output(ENABLE, GPIO.HIGH)
print("Driver disabled. Waiting 2s...")
time.sleep(2)

# Enable driver
GPIO.output(ENABLE, GPIO.LOW)
print("Driver enabled.")

def run_constant_speed(delay, steps, direction):
    """
    Movimiento a velocidad constante (sin aceleración).
    """
    GPIO.output(DIR, direction)

    for _ in range(steps):
        GPIO.output(STEP, GPIO.HIGH)
        time.sleep(delay)
        GPIO.output(STEP, GPIO.LOW)
        time.sleep(delay)

#

```

Para el motor paso a paso NEMA17 se utilizó la librería RPi.GPIO, configurando los pines DIR, STEP y ENABLE del driver DRV8825 junto con los pines M0–M2 en modo full-step. Se implementaron dos rutinas principales: una de aceleración y otra de velocidad constante. De este modo se simuló un perfil trapezoidal básico, en el que el motor acelera progresivamente, mantiene una velocidad estable y finalmente desacelera antes de detenerse. El programa realiza tres revoluciones en ambos sentidos (horario y antihorario), lo que permitió verificar la correcta conexión del motor, el ajuste del Vref y el comportamiento general del conjunto.

Tanto el código de servos como el del stepper se desarrollaron como pruebas funcionales independientes con el objetivo de validar cada subsistema antes de integrarlo en el brazo robótico. Estos ensayos resultaron fundamentales para confirmar la estabilidad de la señal PWM, el funcionamiento del DRV8825 con la corriente adecuada y la respuesta mecánica de las articulaciones. A partir de estos resultados se ajustó el diseño mecánico y se verificó que la electrónica era capaz de ejecutar los movimientos necesarios para el proyecto.

## Conclusiones

A lo largo del proyecto se presentaron múltiples dificultades que afectaron tanto al diseño mecánico como al desarrollo electrónico y al proceso de integración general. El trabajo con modelos 3D supuso uno de los mayores retos iniciales, debido a la necesidad de manipular archivos en formato mesh y a la incompatibilidad de estos con programas paramétricos como FreeCAD. Esto obligó a rehacer piezas completas, convertir formatos repetidamente y realizar numerosas iteraciones hasta conseguir que cada componente encajara con las tolerancias adecuadas. La base giratoria, los brazos y las carcasa de los motores tuvieron que ser rediseñados varias veces, lo que consumió gran parte del tiempo total del proyecto.

En la parte electrónica, el motor NEMA17 y su driver DRV8825 representaron el obstáculo más crítico. El ajuste del Vref no funcionaba como se esperaba: el potenciómetro no ofrecía valores intermedios y saltaba de 0 mV a tensiones elevadas, indicando un comportamiento anómalo del driver o posibles errores de medición. Para superar este problema se recurrió a un método alternativo, regulando el potenciómetro observando directamente el consumo de corriente en la fuente de alimentación y ajustando paulatinamente hasta obtener un punto de funcionamiento estable. También fue necesario identificar manualmente los pares de bobinas del motor, reorganizar los cables y fijar los pines M0–M2 a un estado lógico bajo desde la Raspberry Pi para evitar fluctuaciones indeseadas.

Los servomotores MG996R presentaron menos problemas, pero requirieron una alimentación separada y una correcta unificación de tierras para asegurar que las señales PWM fueran interpretadas de forma precisa. El uso de la librería pigpio permitió generar un PWM estable y realizar pruebas de movimiento que confirmaron la respuesta adecuada del sistema. Del mismo modo, los ensayos con el motor paso a paso mediante RPi.GPIO fueron esenciales para validar el comportamiento del driver, comprobar la suavidad del movimiento tras la incorporación del reductor universal y verificar que el conjunto mecánico soportaba las cargas esperadas sin vibraciones excesivas.

La integración completa del sistema demostró la importancia de trabajar de manera modular: cada componente fue probado individualmente antes de ensamblarse, lo que permitió localizar fallos de forma más eficiente. Las dificultades encontradas se resolvieron combinando iteración, análisis, pruebas controladas y modificaciones tanto en hardware como en software. Este proceso, aunque largo, permitió obtener un brazo funcional y estable, capaz de realizar los movimientos previstos y de responder correctamente a las señales generadas desde la Raspberry Pi.

Como consideración futura, sería recomendable rediseñar el portaplumas para asegurar una sujeción más firme, mejorar la estructura de los brazos utilizando tolerancias más precisas o materiales más robustos, y sustituir las conexiones de protoboard por soldaduras definitivas que eviten falsos contactos. También se plantea la posibilidad de emplear drivers más avanzados, añadir retroalimentación mediante encoders o integrar una fuente de alimentación con salidas múltiples para simplificar el cableado. Asimismo, sería útil unificar la electrónica en una PCB propia y refinar el software para incluir perfiles de movimiento más suaves y controlados. Estas mejoras contribuirían a aumentar la precisión, reducir vibraciones y facilitar el mantenimiento futuro del sistema.