



COSMO TOOLS

Guía de Inicio **NOVA-RP2350**



Bienvenido

Tal vez pienses en las computadoras como esos dispositivos con pantalla y teclado que tienes en tu escritorio, y es cierto, lo son. Pero no son el único tipo.

En esta guía exploraremos los microcontroladores: pequeñas unidades de procesamiento que seguramente ya tienes en casa sin saberlo. Es muy probable que tu lavadora, microondas o termostato estén controlados por uno. Sin embargo, esos dispositivos suelen ser cerrados: sus fabricantes dificultan o impiden que modifiques el software que ejecutan.

NOVA, por otro lado, es diferente. Se puede reprogramar fácilmente todas las veces que quieras a través de una simple conexión USB. En las siguientes páginas veremos cómo empezar a utilizar este hardware y cómo combinarlo con otros componentes electrónicos.

Lo que construyas con ellos depende totalmente de ti.

Esta es la guía de **NOVA**. Esta documentación está diseñada para hobbistas, estudiantes y entusiastas que quieren entender **cómo funcionan las cosas** sin perderse en tecnicismos matemáticos complejos.

Si tu objetivo es aprender electrónica programable, robótica básica y control de dispositivos, este es tu punto de partida.

Contenidos

Capítulo 1: Conoce tu NOVA

Familiarízate con tu nuevo y potente microcontrolador, aprende cómo usar sus pines e instala MicroPython para programarlo.

Capítulo 2: Programación con MicroPython

Conecta una computadora y comienza a escribir programas para tu NOVA usando el lenguaje MicroPython.

Capítulo 3: Computación Física

Aprende sobre los pines de tu NOVA y los componentes electrónicos que puedes conectar y controlar.

Capítulo 4: Computación Física con NOVA

Empieza a conectar componentes electrónicos básicos a tu NOVA y escribe programas para controlarlos y detectar sus señales.

Capítulo 5: Controlador de Semáforos

Crea tu propio mini sistema de cruce peatonal usando múltiples LEDs y un botón pulsador.

Capítulo 6: Juego de Reacción

Construye un juego simple de tiempos de reacción usando un LED y botones, para uno o dos jugadores.

Capítulo 7: Alarma Antirrobo

Usa un sensor de movimiento para detectar intrusos y activa la alarma con una luz intermitente y una sirena.

Capítulo 8: Medidor de Temperatura

Usa el ADC integrado de NOVA para convertir entradas analógicas y lee su sensor de temperatura interno.

Capítulo 9: Registrador de Datos (Data Logger)

Convierte tu NOVA en un dispositivo de registro de temperatura y desconéctalo de la computadora para hacerlo totalmente portátil.

Capítulo 10: Protocolos de Comunicación Digital: I2C y SPI

Explora estos dos protocolos de comunicación populares y úsalos para mostrar datos en una pantalla LCD.

1 Conoce tu NOVA

NOVA es una herramienta potente. En su corazón late el **Raspberry Pi RP2350A**, un microcontrolador moderno.

1.1 ¿Qué significa esto en la práctica?

- **Doble Núcleo (Dual Core):** Imagina que tienes dos cerebros. Mientras uno se encarga de mantener encendida una pantalla, el otro puede estar leyendo sensores o controlando un motor. Trabajan en paralelo (Multitasking real).
- **Velocidad (150 MHz):** Puede procesar millones de instrucciones por segundo. Para proyectos de luces, robots y sensores, es increíblemente rápido.
- **Pines de Propósito General (GPIO):** Son las "patitas" de la placa. A diferencia de un puerto USB normal de tu PC, estos pines se pueden programar para ser interruptores (encender/apagar cosas) o entradas (leer botones/sensores).

1.2 Componentes principales de tu NOVA

Para dominar NOVA, primero debemos llamar a cada cosa por su nombre. Aquí tienes el "diccionario" de tu hardware:

- **Puerto USB-C (Energía y Datos):** Es el cordón umbilical de la placa. Por aquí recibe electricidad (5V) para funcionar y datos desde tu computadora cuando la programas.
- **Chip RP2350 (El Microcontrolador):** Es el "cerebro" real. Es el componente cuadrado y negro en el centro. Aquí es donde "vive" tu código y se toman las decisiones a 150 Mhz.
- **Memoria Flash (Almacenamiento):** Es como el disco duro de tu computadora, pero en miniatura. Aquí se guarda tu programa para que no se borre cuando desconectas la placa.
- **GPIOs (Pines de Entrada/Salida):** Esas "patitas" o agujeros dorados a los lados. Son las manos y ojos de NOVA. Puedes configurarlos para **leer** el mundo (sensores) o **actuar** sobre él (encender luces, mover motores).
- **Botón BOOT (Modo de carga):** Si lo mantienes presionado al conectar el USB, le dices a NOVA: "¡Alto! No ejecutes nada, voy a instalarte un nuevo sistema".
- **Botón RESET (Reinicio):** Es como desenchufar y volver a enchufar la placa, pero más rápido. Útil si tu programa se queda atascado.
- **LED de Usuario (Test):** Una pequeña luz integrada en la placa que tú puedes controlar. Es perfecta para realizar tus primeras pruebas sin conectar nada extra.

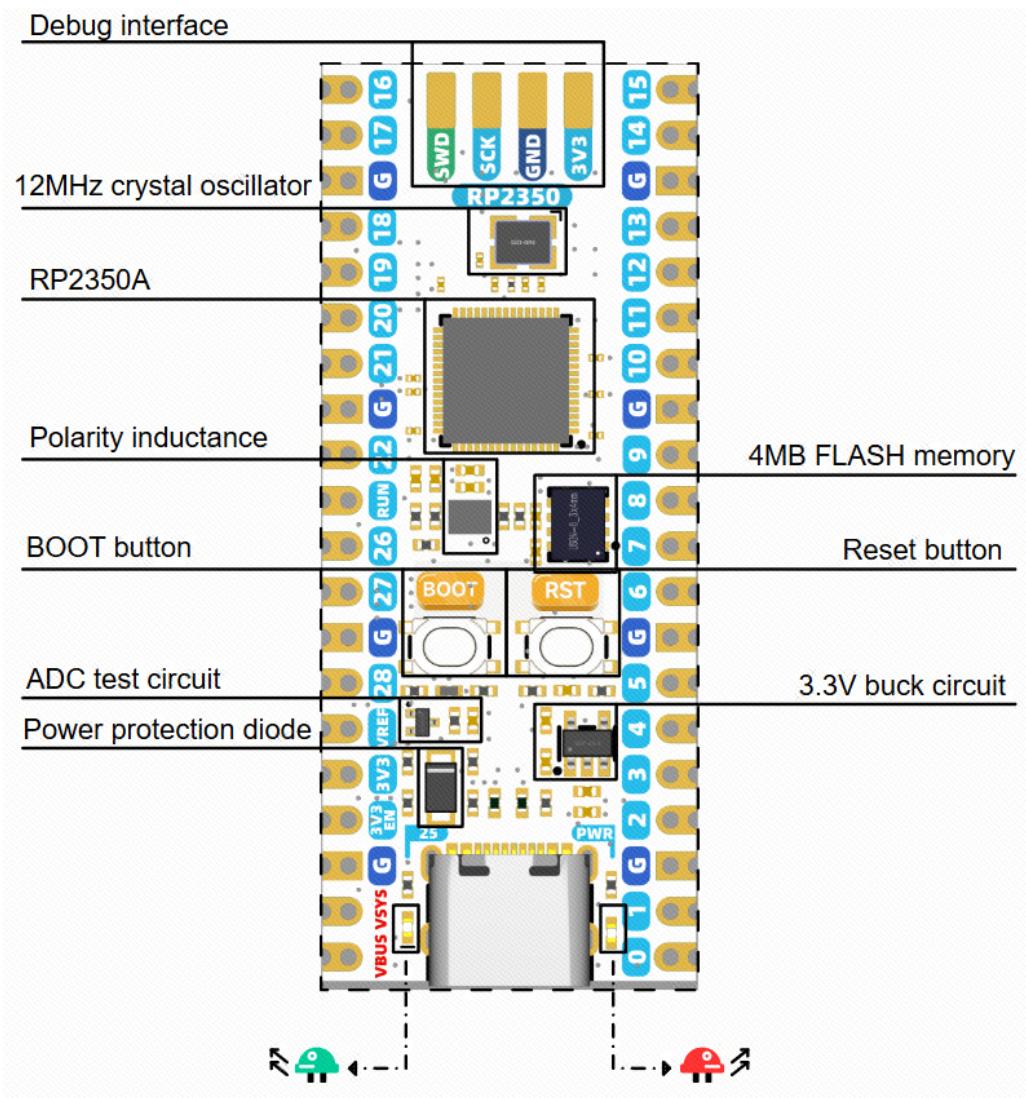


Figure 1: Mapa de componentes de NOVA

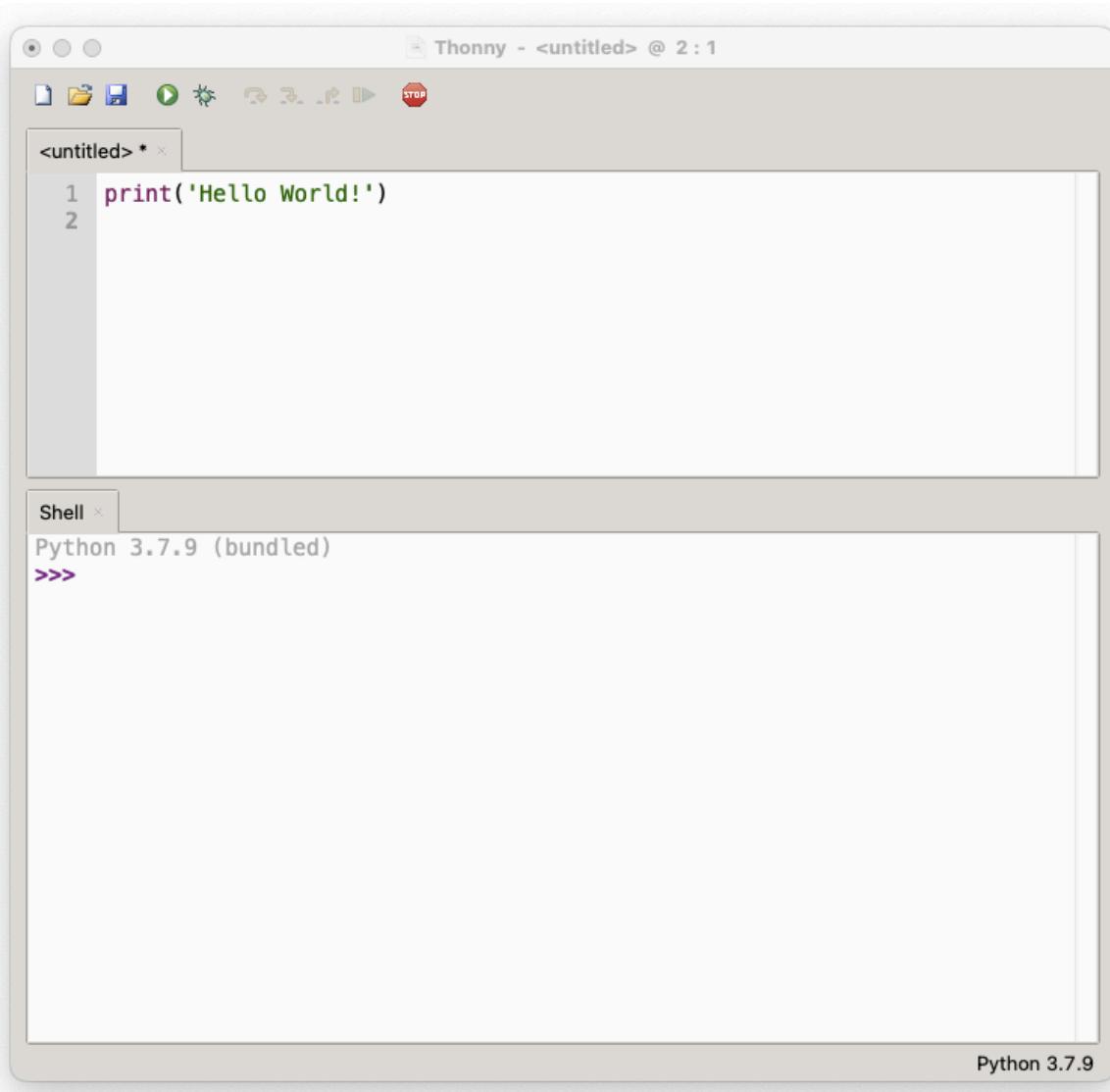
2 Programación con MicroPython

Para comunicarnos con NOVA, usaremos **MicroPython**.

- **¿Qué es?** Es una versión de Python 3 optimizada para funcionar en microcontroladores. Es el lenguaje ideal para empezar porque es legible y potente.
- **La Herramienta:** Usaremos **Thonny IDE**. Es un editor de código simple que ya trae todo lo necesario para hablar con tu placa.

2.1 Pasos rápidos:

1. Conecta tu NOVA al ordenador manteniendo pulsado el botón **BOOT**.
2. Tu PC lo verá como una memoria USB.
3. Descarga la **última versión** del firmware .uf2 para RP2350 desde <https://micropython.org/download/rp2-pico2/>. Copia o arrastra el archivo descargado en esa memoria. (La placa se reiniciará automáticamente, eso es normal).
4. Abre Thonny y configura el intérprete.
 - **Ubicación:** Haz clic en el texto que aparece en la **esquina inferior derecha** de la ventana principal (o ve al menú *Ejecutar → Configurar intérprete*).
 - **Selección:** Elige la opción **"MicroPython (Raspberry Pi Pico)"**.
 - **¿Por qué este?** Aunque tu placa se llama NOVA, su cerebro (RP2350) pertenece a la familia RP2. Thonny necesita este controlador específico para poder comunicarse por USB, enviar tu código y mostrar los mensajes de la placa en la consola.



Interfaz de Thonny con el primer código

2.2 Tu primer programa: Hola Mundo (Blink)

En el mundo de la programación de software, lo primero que aprendemos es a imprimir un texto en pantalla. En el mundo de los microcontroladores, nuestro "Hola Mundo" es hacer parpadear un LED.

Copia el siguiente código en el editor principal de Thonny:

```
1 from machine import Pin
2 import time
3
4 # Configuramos el LED integrado (Pin 25)
5 led = Pin(25, Pin.OUT)
6
7 while True:
8     led.toggle()      # Cambia estado (ON/OFF)
9     print("Blink!")   # Mensaje en consola
10    time.sleep(1)    # Espera 1 segundo
```

2.2.1 Explicación del código:

1. **Importar librerías:** `machine` nos da control sobre el hardware y `time` maneja el tiempo.
2. **Configuración:** Creamos un objeto `led` en el pin 25 y le indicamos que será de **SALIDA** (OUT), es decir, enviará voltaje.
3. **Bucle Infinito:** `while True`: crea un ciclo sin fin para que el programa nunca se detenga.
4. **Acción:** `.toggle()` invierte el estado actual del LED.

Pulsa el botón **Ejecutar** (ícono de "Play" verde) en Thonny. ¡Verás el LED de tu NOVA parpadear!

¡Felicidades!

Has logrado programar tu primer código y encender el LED correctamente. ¡Este es tu primer gran paso en el control de hardware!

Por cierto, ¿sabes lo que es hardware, firmware y software? ¿No? Bueno, te explico rápido:

Imagina que tu NOVA es un músico.

- El **Hardware** es el instrumento (la guitarra, el piano). Es la parte física, lo que puedes tocar y, si se cae, se rompe.
- El **Firmware** es el **traductor experto**. Tu placa solo entiende de electricidad (voltajes), y tú escribes palabras en inglés (Python). El firmware es el programa que vive en el chip y se encarga de traducir tus órdenes humanas a acciones eléctricas reales.
- El **Software** es el **director de orquesta** (¡Ese eres tú!). Es el código lógico que escribes en Thonny para decidir cuándo y cómo debe sonar el instrumento. Sin tu software, el hardware está mudo y el firmware aburrido.

¡Tú acabas de componer tu primera obra!

3 Hardware y Conexiones

Aquí entramos en el mundo de la **Electrónica**: conectar el software con el mundo real. Hasta ahora, tu código vivía encerrado en el chip; ahora vamos a darle sentidos (sensores) y músculos (motores/luces).

3.1 El Mapa de tu NOVA (Pinout)

Para conectar componentes sin "quemar" nada, necesitas un mapa. A los pines metálicos los llamamos **GPIO** (*General Purpose Input/Output*). NO todos son iguales, así que presta atención a este diagrama:

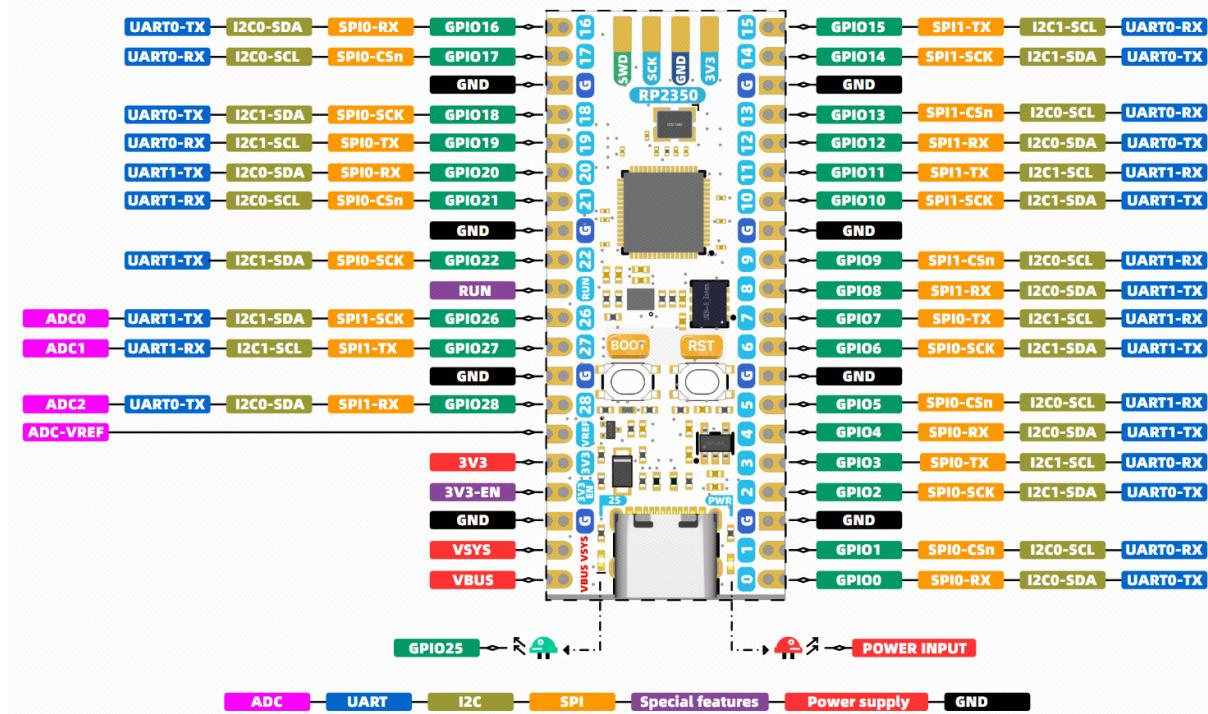


Figure 2: Diagrama de pines (Pinout) de NOVA RP2350

3.2 Tipos de Pines y su función

- **GP0 - GP29 (Digitales):** Son los pines "normales". Solo entienden dos estados: **3.3V** (Encendido/1) y **0V** (Apagado/0).
- **GND (Tierra/Masa):** ¡El más importante! Es el polo negativo. Todos los circuitos electrónicos deben conectarse a GND para cerrarse y funcionar. En NOVA, los verás marcados en negro o con formas cuadradas en el diagrama.
- **3V3 (Energía):** Estos pines siempre entregan 3.3 voltios. Úsalos para "alimentar" sensores o componentes que necesiten energía constante.
- **ADC (Analógicos):** Pines especiales (GP26, GP27, GP28) que pueden medir voltajes variables (como el volumen de una perilla o la intensidad de luz), no solo "todo o nada".

3.3 Protocolos de Comunicación (Para el futuro)

Además de encender luces, algunos pines saben "hablar" idiomas complejos para conectarse con sensores avanzados o pantallas:

- **I2C:** Usa 2 cables (SDA, SCL). Ideal para pantallas OLED y sensores de movimiento.
- **UART:** Comunicación serie clásica. Útil para módulos GPS o Bluetooth.
- **SPI:** Comunicación ultra-rápida. Usada en lectores de tarjetas SD o pantallas a color.

4 Control Digital: Luces y Botones

En el capítulo anterior vimos el mapa. Ahora vamos a manejar el coche. Los pines GPIO digitales son la forma más básica de comunicación: solo entienden dos palabras: **ENCENDIDO (1/High)** y **APAGADO (0/Low)**.

4.1 Salidas Digitales (Escribir)

Cuando configuras un pin como **SALIDA (OUT)**, el microcontrolador "impone" un voltaje. Es como abrir o cerrar un grifo de agua.

Vamos a controlar un LED externo. A nivel eléctrico, los LEDs son delicados; siempre necesitan una **resistencia de protección** (220Ω o 330Ω) para no quemarse por exceso de corriente.

Circuito:

- **Pin 15** → Pata Larga del LED (+).
- **GND** → Resistencia → Pata Corta del LED (-).

Código Básico (Parpadeo):

```
1 from machine import Pin
2 import time
3
4 led = Pin(15, Pin.OUT) # Configuramos Pin 15 como SALIDA
5
6 while True:
7     led.toggle()      # Cambia el estado (si estaba on, se apaga)
8     time.sleep(0.5)
```

4.2 Proyecto: Ojos de Robot (Lógica)

Vamos a subir el nivel. Usaremos el módulo `random` y funciones de Python para dar "personalidad" a dos LEDs, simulando ojos que parpadean o escanean.

```
1 from machine import Pin
2 import time
3 import random
4
5 # Definimos los pines de los LEDs
6 ojo_izq = Pin(15, Pin.OUT)
7 ojo_der = Pin(14, Pin.OUT)
8
9 def parpadeo_normal():
10     """Simula un parpadeo rápido simultáneo"""
11     ojo_izq.off()
12     ojo_der.off()
13     time.sleep(0.15)
14     ojo_izq.on()
15     ojo_der.on()
16
17 def escaneo():
18     """Altera luces como buscando algo"""
19     print("Escaneando entorno...")
20     for i in range(4):
```

```

21     ojo_izq.toggle()
22     time.sleep(0.08)
23     ojo_der.toggle()
24     time.sleep(0.08)
25 # Al finalizar, encendemos ambos
26 ojo_izq.on()
27 ojo_der.on()
28
29 # --- Bucle Principal ---
30 print("Sistema Iniciado. Ejecutando IA basica.")
31 ojo_izq.on()
32 ojo_der.on()
33
34 while True:
35     # Espera aleatoria entre acciones (2 a 5 segundos)
36     time.sleep(random.uniform(2.0, 5.0))
37
38     # 70% Probabilidad de parpadeo, 30% escaneo
39     if random.randint(1, 100) <= 70:
40         parpadeo_normal()
41     else:
42         escaneo()

```

4.3 Entradas Digitales (Leer Botones)

Hasta ahora NOVA solo "hablaba". Ahora va a "escuchar". Para leer un botón, configuramos el pin como **ENTRADA (IN)**.

Pero hay un problema: ¿Qué voltaje hay en el cable cuando el botón NO está presionado? ¿0V? ¿3.3V? La respuesta es radiación electromagnética aleatoria (ruido). Para evitar falsas lecturas, activamos una resistencia interna llamada **PULL_DOWN**, que conecta el pin a tierra (0V) suavemente cuando nadie lo toca.

Reto: Crea un interruptor de luz.

```

1 from machine import Pin
2 import time
3
4 led = Pin(15, Pin.OUT)
5 boton = Pin(16, Pin.IN, Pin.PULL_DOWN) # PULL_DOWN asegura 0V si no pulsas
6
7 print("Listo para recibir ordenes...")
8
9 while True:
10     estado = boton.value() # Leemos el estado (0 o 1)
11
12     if estado == 1:
13         led.on()
14     else:
15         led.off()
16
17     time.sleep(0.05) # Pequeña pausa para estabilidad

```

4.4 Movimiento y PWM (Servomotores)

Un servomotor no funciona simplemente "encendiendo y apagando" la corriente. Necesita una **señal de control precisa** usando PWM (Pulse Width Modulation).

Código de Control de Servo:

```
1 from machine import Pin, PWM
2 import time
3
4 # Configuracion del PWM en el Pin 16
5 # Los servos estandar funcionan a una frecuencia de 50Hz
6 servo = PWM(Pin(16))
7 servo.freq(50)
8
9 def mover_servo(angulo):
10     """
11     Convierte un angulo (0-180) al ciclo de trabajo (duty) necesario.
12     """
13     # Mapeo simple: 0-180 -> duty_u16
14     duty = int(1638 + (angulo / 180) * (8192 - 1638))
15     servo.duty_u16(duty)
16
17 while True:
18     print("Moviendo a 0 grados")
19     mover_servo(0)
20     time.sleep(1)
21
22     print("Moviendo a 90 grados")
23     mover_servo(90)
24     time.sleep(1)
25
26     print("Moviendo a 180 grados")
27     mover_servo(180)
28     time.sleep(1)
```

Nota: Los capítulos siguientes (5 al 10) están en desarrollo para futuras actualizaciones.

Recursos Adicionales

- **Documentación Oficial MicroPython:** docs.micropython.org
- **Datasheet RP2350:** Para cuando necesites los detalles técnicos profundos de los registros y memoria.
- **Repositorio del Proyecto:** Ejemplos de código y librerías específicas para NOVA.

¡Bienvenido al nivel intermedio! Aquí es donde la creatividad se encuentra con la ingeniería.