NOVA RP2350 — Guía de Inicio Práctico

Introducción al Mundo Maker y Electrónica

Bienvenido a 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.

---

1. Conociendo tu Herramienta: El Chip RP2350

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

¿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).

---

2. Preparando el Entorno

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.

Pasos rápidos:

1. Conecta tu NOVA al ordenador manteniendo pulsado el botón **BOOT**.
2. Tu PC lo verá como una memoria USB.
3. Copia el archivo de firmware `.uf2` de MicroPython en esa memoria. (La placa se reiniciará, eso es normal).
4. Abre Thonny y selecciona el intérprete "MicroPython (Raspberry Pi Pico)".

---

3. Práctica 1: Control Digital (Salidas)

Lo primero en electrónica es aprender a controlar el estado de un pin: **Alto (3.3V)** o **Bajo (0V)**.

En este ejemplo, controlaremos un LED externo. A nivel eléctrico, el LED necesita una resistencia (aprox 220Ω) para limitar la corriente y no quemarse.

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

El Código: Observa cómo importamos la librería `machine`. Esta librería es el puente entre el código software y el hardware físico.

```
from machine import Pin
import time

# Configuramos el Pin 15 como SALIDA (OUT)
# Esto permite que el pin entregue voltaje (3.3V)
led = Pin(15, Pin.OUT)

print("Iniciando secuencia de control...")

while True:
   led.on()            # Envía 3.3V al pin (Encendido)
   time.sleep(1)       # Pausa el procesador 1 segundo
   led.off()           # Corta el voltaje (Apagado)
   time.sleep(1)
```

---

4. Práctica 2: Lógica y Aleatoriedad (Simulación de Comportamiento)

Los robots parecen "vivos" no porque tengan conciencia, sino porque usan algoritmos para tomar decisiones. Vamos a usar el módulo `random` para simular un comportamiento impredecible usando dos LEDs (como si fueran ojos).

Concepto Clave: Aquí veremos cómo crear **funciones**. Las funciones nos permiten encapsular comportamientos (como "parpadear" o "mirar a un lado") y reutilizarlos.

Código:

```
from machine import Pin
import time
import random

# Definimos los pines de los LEDs
ojo_izq = Pin(15, Pin.OUT)
ojo_der = Pin(14, Pin.OUT)

# --- Definición de Comportamientos ---

def parpadeo_normal():
   """Simula un parpadeo humano rápido"""
   ojo_izq.off()
   ojo_der.off()
   time.sleep(0.15)
   ojo_izq.on()
   ojo_der.on()

def escaneo():
   """Simula un robot analizando el entorno"""
   print("Escaneando...")
   for i in range(4):
       ojo_izq.toggle() # toggle() invierte el estado actual
       time.sleep(0.05)
       ojo_der.toggle()
       time.sleep(0.05)
   # Aseguramos que queden encendidos al final
   ojo_izq.on()
   ojo_der.on()

# --- Bucle Principal ---
print("Sistema Iniciado. Ejecutando IA básica.")
ojo_izq.on()
ojo_der.on()

while True:
   # El sistema espera un tiempo aleatorio entre acciones
   wait_time = random.uniform(2.0, 5.0)
   time.sleep(wait_time)

   # Decisión aleatoria
   decision = random.randint(1, 10)

   if decision <= 7:
       # 70% de probabilidad de parpadeo normal
       parpadeo_normal()
   else:
       # 30% de probabilidad de escaneo
       escaneo()
```

---

5. Práctica 3: Movimiento y PWM (Servomotores)

Un servomotor no funciona simplemente "encendiendo y apagando" la corriente. Necesita una **señal de control precisa**.

¿Qué es PWM (Pulse Width Modulation)?

Es una técnica para simular señales analógicas o enviar datos codificados pulsando una señal digital muy rápido. * Para un LED, el PWM controla el brillo. * Para un Servo, el PWM le indica **en qué ángulo posicionarse** (0° a 180°).

Advertencia de Energía ■■

Un servomotor es un dispositivo mecánico. Consume mucha más energía de la que el puerto USB de un ordenador o el regulador de la placa pueden dar cómodamente si hay carga. * **Correcto:** Alimentar el servo con una fuente externa (4 pilas AA o batería 5V) y unir las tierras (GND). * **Señal:** Solo el cable de

señal (Naranja/Amarillo) va al pin de NOVA.

Código de Control de Servo:

```
from machine import Pin, PWM
import time

# Configuración del PWM en el Pin 16
# Los servos estándar funcionan a una frecuencia de 50Hz
servo = PWM(Pin(16))
servo.freq(50)

def mover_servo(angulo):
   """
   Convierte un ángulo (0-180) al ciclo de trabajo (duty) necesario.
   Cálculo aproximado para servos SG90:
   - 0 grados   ~ 1000 us pulso
   - 180 grados ~ 9000 us pulso
   Nota: Estos valores pueden variar según el fabricante del servo.
   """
   # Mapeo simple: 0-180 -> duty_u16 (aprox 1638 a 8192)
   duty = int(1638 + (angulo / 180) * (8192 - 1638))
   servo.duty_u16(duty)

while True:
   print("Moviendo a 0 grados")
   mover_servo(0)
   time.sleep(1)

   print("Moviendo a 90 grados")
   mover_servo(90)
   time.sleep(1)

   print("Moviendo a 180 grados")
   mover_servo(180)
   time.sleep(1)
```

---

6. Siguientes Pasos Intermedios

Una vez domines las salidas digitales (LEDs) y el PWM (Servos), el siguiente nivel es la **lectura de datos** y la **comunicación**.

Sensores Analógicos (ADC)

El mundo real no es "1 o 0". La temperatura, la luz o el sonido varían suavemente. * NOVA tiene conversores **Analogico-Digitales (ADC)** de 12-bits. * Permiten leer voltajes variables (0V a 3.3V) y convertirlos en un número (0 a 65535). Esto es fundamental para leer potenciómetros, sensores de temperatura LM35, etc.

Protocolos de Comunicación

En la industria y proyectos avanzados, los componentes "hablan" entre sí. * **I2C:** Para conectar sensores modernos (acelerómetros, pantallas OLED) usando solo 2 cables. * **UART/Serial:** Para comunicar tu NOVA con otros módulos como GPS, Bluetooth o WiFi (ESP-01).

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.