Gemelo digital 'offline' con Arduino UNO en Wokwi

Vas a construir un gemelo digital sencillo: un modelo virtual que observa el estado de un "dispositivo físico" (simulado) y lo controla.

El dispositivo tendrá un potenciómetro (sensor analógico), un botón (sensor digital) y un LED (actuador con brillo PWM).

El gemelo intercambia información por el Monitor Serie:

- envía telemetría JSON cada 1 segundo (lecturas y estado)
- recibe comandos de control (ON, OFF, SET <0..255>, STATUS, HELP) para actuar sobre el LED.

Explicación

Un gemelo digital es una representación virtual de un objeto físico con capacidad de observar su estado y controlarlo.

Un **gemelo digital** es una representación virtual de un objeto o sistema físico con capacidad de **observar** su estado y **controlarlo**. En este ejercicio:

- **Físico (simulado)**: potenciómetro (A0), botón (D2), LED (D9).
- **Gemelo (virtual)**: consola serie de Wokwi que:
    - Recibe **telemetría** (JSON cada 1 s).
    - Envía **mandos** de control por texto.

Bidireccionalidad:

- De físico a gemelo: lecturas (pot, button, temp) llegan al Serial.
- De gemelo a físico: comandos (ON, OFF, SET 127) cambian el LED.

Esto reproduce el patrón básico de los gemelos digitales industriales (sensado + control), pero **sin Internet ni librerías**, para que funcione en cualquier cuenta gratuita.

En este ejercicio el gemelo será la consola serie y el activo físico simulado será el Arduino con potenciómetro, botón y LED.

Resolución paso a paso

1. Crear proyecto Arduino UNO en Wokwi.

https://wokwi.com/projects/new/arduino-uno

Debes tener un archivo sketch.ino y un diagram.json

2. Copiar el archivo diagram.json con las conexiones especificadas.

```
{
  "version": 1,
  "author": "Gemelo Digital - Arduino UNO",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-uno", "id": "uno", "top": -8, "left": -8, "attrs": {} },
    { "type": "wokwi-potentiometer", "id": "pot1", "top": 40, "left": 220, "attrs": { "value": "50" } },
    { "type": "wokwi-led", "id": "led1", "top": -20, "left": 260, "attrs": { "color": "red" } },
```

```
    { "type": "wokwi-resistor", "id": "r1", "top": 12, "left": 300, "rotate": 90, "attrs": {
"value": "220" } },

    { "type": "wokwi-button", "id": "btn1", "top": -110, "left": 200, "attrs": { "color": "green"
} }

  ],

  "connections": [

    ["pot1:VCC", "uno:5V", "red"],

    ["pot1:GND", "uno:GND.1", "black"],

    ["pot1:SIG", "uno:A0", "blue"],


    ["led1:A", "r1:1", "orange"],

    ["r1:2", "uno:9", "orange"],       // LED por PWM en D9

    ["led1:C", "uno:GND.2", "black"],


    ["btn1:1.l", "uno:2", "green"],    // Botón en D2 (INPUT_PULLUP)

    ["btn1:2.r", "uno:GND.3", "black"]

  ]

}
```

3. Copiar el código sketch.ino con el programa completo.

```
/*
```

```
 GEMELO DIGITAL "OFFLINE" - ARDUINO UNO (Wokwi)
 ----------------------------------------------
 Sin librerías externas. Emula un activo y su gemelo:
 Sensores:
  - Potenciómetro en A0 -> 0..1023 (normalizado 0..1)
  - Botón en D2 (INPUT_PULLUP): suelto=1, pulsado=0
 Actuador:
  - LED en D9 (PWM 0..255)
 Interfaz (el gemelo):
  - Telemetría JSON cada 1000 ms por Serial (115200)
  - Comandos por Serial:
     HELP        -> ayuda
     STATUS      -> estado actual
     ON          -> LED = 255
     OFF         -> LED = 0
     SET <0..255> -> LED = valor
*/

#include <Arduino.h>

// Pines
const uint8_t PIN_POT = A0;
```

```
const uint8_t PIN_BTN = 2;   // INPUT_PULLUP
const uint8_t PIN_LED = 9;   // PWM

// Estado del "activo"
struct Estado {
  float  potNorm;   // 0..1
  bool   btn;       // true=suelto, false=pulsado
  uint8_t led;      // 0..255
  unsigned long ms; // millis
  float  tempSim;   // temperatura simulada
} st;

// --- Utilidades de hardware ---
// Ajusta el brillo del LED (0..255) por PWM en D9
void setLed(uint8_t v) { st.led = v; analogWrite(PIN_LED, v); }
// Lee el potenciómetro y normaliza a 0..1
float leerPotNorm()    { return constrain(analogRead(PIN_POT) / 1023.0f, 0.0f,
1.0f); }
// Lee el botón con pull-up: HIGH=suelto, LOW=pulsado
bool  leerBtnSuelto()  { return digitalRead(PIN_BTN) == HIGH; }
// Genera una temperatura simulada con variación suave + ruido
float simularTemp(unsigned long ms) {
  float base = 22.0f + 5.0f * sin(ms / 30000.0f);
```

```
  float ruido = (random(-30, 30) / 100.0f);
  return base + ruido;
}

// --- Telemetría ---
// Imprime una línea JSON con el estado actual (fácil de parsear)
void imprimirTelemetria() {
  Serial.print(F("{\"ms\":"));      Serial.print(st.ms);
  Serial.print(F(",\"temp\":"));    Serial.print(st.tempSim, 2);
  Serial.print(F(",\"pot\":"));     Serial.print(st.potNorm, 3);
  Serial.print(F(",\"button\":"));  Serial.print(st.btn ? 1 : 0);
  Serial.print(F(",\"led\":"));     Serial.print((int)st.led);
  Serial.println(F("}"));
}
// Mensaje de ayuda
void imprimirAyuda() {
  Serial.println(F("---- GEMELO DIGITAL (Arduino UNO, Wokwi) ----"));
  Serial.println(F("Comandos: HELP | STATUS | ON | OFF | SET <0..255>"));
  Serial.println(F("Telemetria JSON cada 1000 ms"));
}

// --- Parser de comandos (desde el Monitor Serie) ---
```

```
void procesarComandosSerial() {
  static String buf;
  while (Serial.available()) {
    char c = (char)Serial.read();
    if (c == '\n' || c == '\r') {     // fin de línea
      buf.trim();
      if (buf.length() > 0) {
        String line = buf;  buf = "";
        String up = line; up.trim(); up.toUpperCase();

        if (up == "HELP")   { imprimirAyuda(); return; }
        if (up == "STATUS") { imprimirTelemetria(); return; }
        if (up == "ON")     { setLed(255);
Serial.println(F("{\"event\":\"cmd\",\"cmd\":\"ON\",\"led\":255}")); return; }
        if (up == "OFF")    { setLed(0);
Serial.println(F("{\"event\":\"cmd\",\"cmd\":\"OFF\",\"led\":0}"));  return; }

        // Comando con argumento: "SET 123"
        int sp = line.indexOf(' ');
        if (sp > 0) {
          String head = line.substring(0, sp);
          String arg  = line.substring(sp + 1);
          head.trim(); arg.trim(); head.toUpperCase();
```

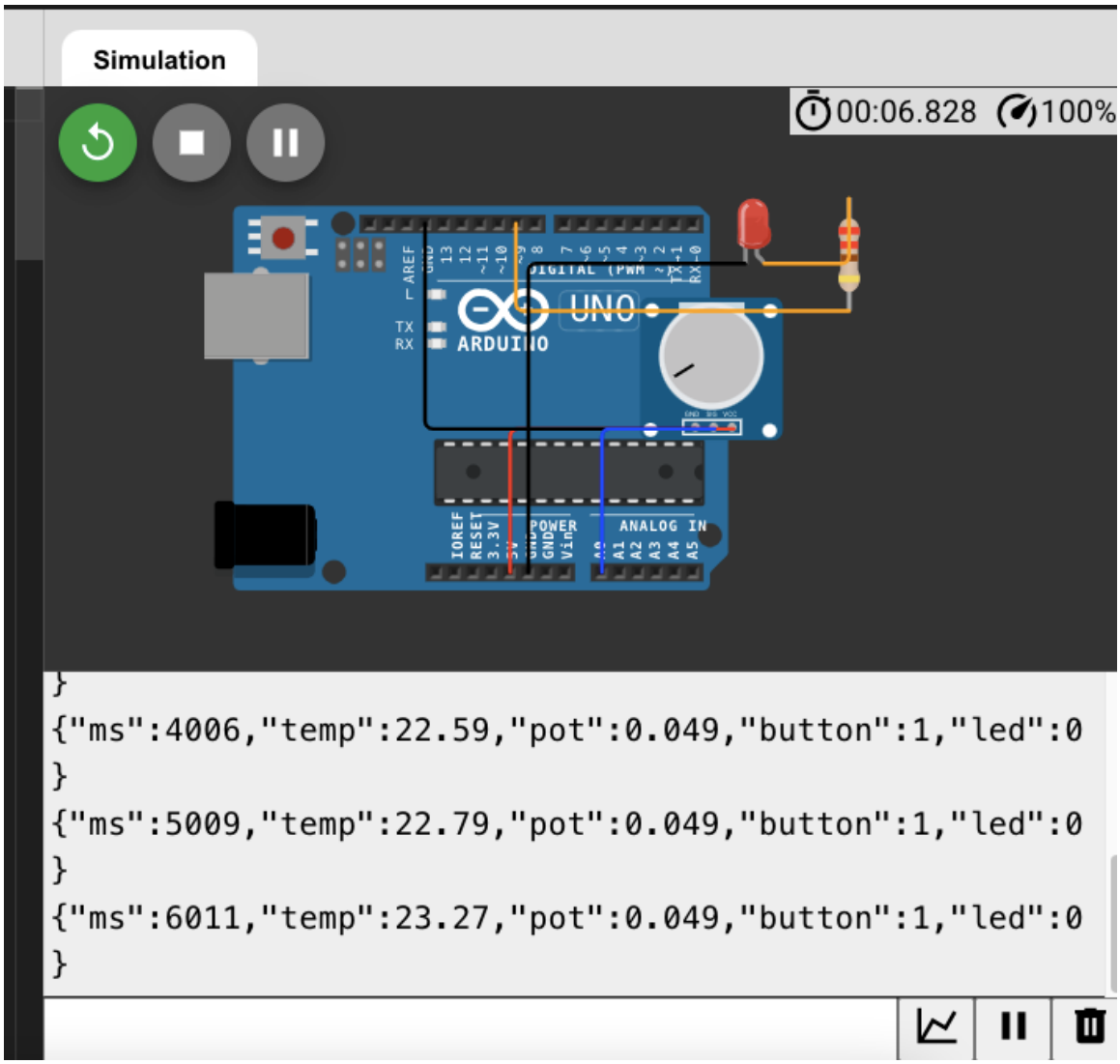
```
        if (head == "SET") {
          int v = constrain(arg.toInt(), 0, 255);
          setLed((uint8_t)v);
          Serial.print(F("{\"event\":\"cmd\",\"cmd\":\"SET\",\"led\":"));
          Serial.print(v);
          Serial.println(F("}"));
          return;
        }
      }

      // Desconocido
      Serial.print(F("{\"event\":\"error\",\"msg\":\"Comando desconocido: "));
      Serial.print(line);
      Serial.println(F("\"}"));
      Serial.println(F("Tip: HELP"));
    }
  } else {
    buf += c; // acumula caracteres hasta fin de línea
  }
 }
}
```

```
// --- Ciclo Arduino ---
unsigned long tUlt = 0;

void setup() {
  Serial.begin(115200);
  delay(50);
  pinMode(PIN_BTN, INPUT_PULLUP);
  pinMode(PIN_LED, OUTPUT);
  setLed(0);
  randomSeed(analogRead(PIN_POT));
  imprimirAyuda();
}

void loop() {
  unsigned long now = millis();
  st.ms      = now;
  st.potNorm = leerPotNorm();
  st.btn     = leerBtnSuelto();
  st.tempSim = simularTemp(now);

  // Telemetría periódica (cada 1000 ms)
  if (now - tUlt >= 1000) { tUlt = now; imprimirTelemetria(); }
```

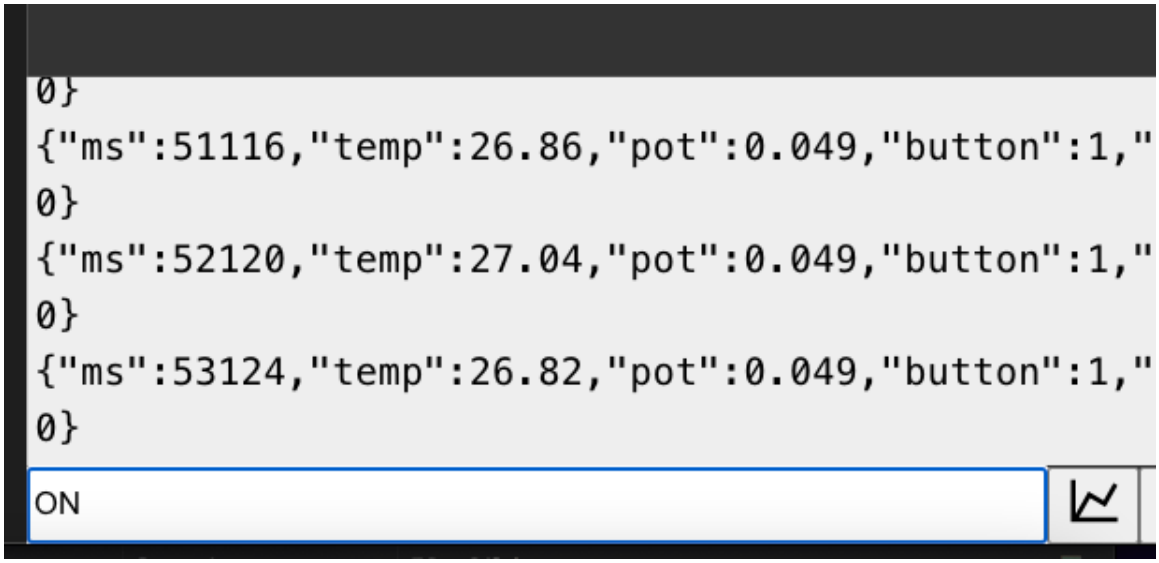
```
  // Procesa comandos del gemelo
  procesarComandosSerial();

  delay(4);
}
```

4. Ejecutar la simulación y abrir el Monitor Serie.

    1. Pulsa **Start Simulation**.
    2. Abre el **Serial Monitor**
    3. Deberías ver:



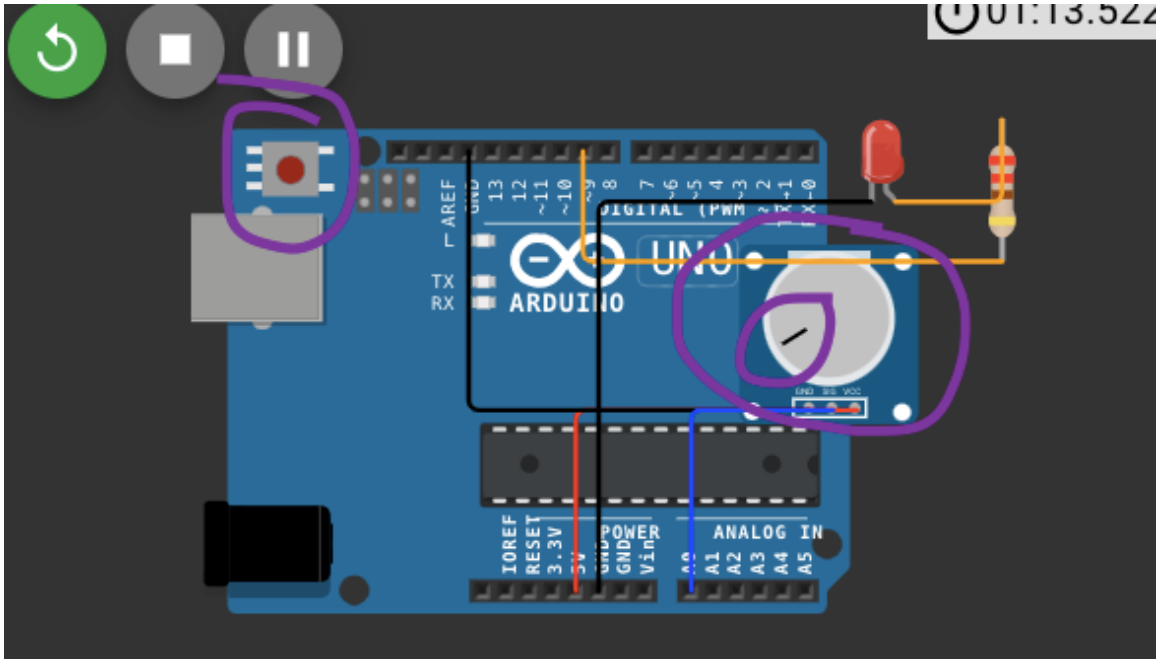
5. Probar comandos (ON, OFF, SET, STATUS, HELP). En la línea de la consola, como el ON de la foto de abajo.



En la **línea blanca inferior** del Serial Monitor (caja de entrada):

- Escribe ON, pulsa Enter y verás que el LED se enciende (255).
- Escribe SET 120, pulsa Enter y verás que el LED pasa a brillo medio.
- Escribe OFF, pulsa Enter y verás el LED apagado.
- STATUS: imprime el estado actual.
- HELP : muestra la ayuda.

6. Interactuar con los sensores: girar el potenciómetro y pulsar el botón.



El mío, por si queréis copiar el código.

https://wokwi.com/projects/446416895445471233

Qué hace este Arduino

Imagina que el **Arduino es una máquina pequeña** —por ejemplo, un ventilador inteligente o una lámpara que se puede controlar desde un ordenador.
En esta práctica, esa máquina tiene tres partes:

- **Un potenciómetro**: simula un **sensor** que mide algo (como la temperatura o la velocidad).
- **Un botón**: simula otro **sensor digital**, que detecta si algo está encendido o apagado.
- **Un LED**: representa una **salida o actuador**, algo que el sistema puede controlar (por ejemplo, encender una luz o un motor).

Y el ordenador (el **gemelo digital**) recibe los datos de esa máquina y le puede mandar órdenes.

Qué hace el programa

1. **Lee los sensores:**
    - Mide el valor del potenciómetro (entre 0 y 1023), eso simula una medida real.
    - Mira si el botón está pulsado o no.
2. **Simula una temperatura:**
    - Genera un número que sube y baja poco a poco (como si midiera la temperatura de verdad).
3. **Envía datos al ordenador:**

- Cada segundo, el Arduino manda por el **Monitor Serie** una línea con los valores de sus sensores.
  Por ejemplo:
- {"ms":1000,"temp":22.4,"pot":0.52,"button":1,"led":0}

Eso significa:

- Han pasado 1000 milisegundos.
- Temperatura 22.4 ºC.
- El potenciómetro está a la mitad.
- El botón no está pulsado.
- El LED está apagado.

4. **Recibe comandos del ordenador:**
   - Si escribes ON en el Monitor Serie, el LED se enciende.
   - Si escribes OFF, el LED se apaga.
   - Si escribes SET 100, el LED se enciende con brillo medio.
   - Si escribes STATUS, te devuelve los valores actuales.

Qué significan los valores que ves

- temp: es una **temperatura simulada** que cambia un poco cada segundo.
- pot: es el valor del **potenciómetro**, va de 0 (giro a la izquierda) a 1 (giro a la derecha).
- button: vale **1** cuando el botón está suelto, **0** cuando lo pulsas.
- led: indica el brillo actual del LED (0 apagado, 255 brillo máximo).

Para qué sirve esto en la vida real

Este proyecto es una **versión mini** de los sistemas que se usan en la industria y el Internet de las Cosas (IoT).

Un **gemelo digital** permite tener una copia virtual de una máquina real para saber cómo está y controlarla sin tocarla físicamente.

Ejemplos:

- **En una fábrica:** los sensores de una máquina envían datos a un panel digital para detectar si hay fallos.
- **En un coche:** se envía la temperatura del motor y otros datos al ordenador del taller.
- **En una casa inteligente:** los sensores de luz o temperatura avisan al sistema, y desde el móvil puedes encender las luces o ajustar la calefacción.