"Introducción a la Robótica para la Enseñanza Técnico Profesional"

Módulo 3 – Programación aplicada a la robótica

Lección 8 – Variables y Constantes - Entender y usar datos en sistemas robóticos

📘 Introducción

En el desarrollo de sistemas robóticos, no basta con encender luces o activar motores: Necesitamos que el sistema reaccione según la información que percibe del entorno. Esa información puede variar constantemente: Un sensor que cambia su valor según la luz, un botón que se presiona o no, un tiempo que se cuenta para encender o apagar algo.
Para poder guardar, leer y modificar esos datos, usamos dos herramientas fundamentales de la programación: Las variables y las constantes.
Una variable nos permite almacenar un valor que puede cambiar durante la ejecución del programa: Por ejemplo, la lectura de un sensor de humedad, o un contador que incrementa en cada ciclo.
Una constante, en cambio, guarda un valor fijo que no se modifica, como el número del pin donde está conectado un LED.
Comprender estos conceptos no solo es importante para escribir código más organizado, sino también para modelar comportamientos más inteligentes, adaptativos y escalables en robótica educativa.

En esta lección, vas a aprender:

Qué son las variables y las constantes en un lenguaje de programación.
Cuándo conviene usar una u otra.
Cómo declararlas y nombrarlas correctamente en el entorno de Arduino.
Por qué su uso mejora la comprensión del sistema y previene errores.
Cómo integrarlas en ejemplos reales, con sensores, botones y actuadores.

Aprender a usar variables y constantes es dar el primer paso hacia el diseño de programas dinámicos y reutilizables, que responden a situaciones reales con flexibilidad y lógica técnica.

🔢 ¿Qué es una variable?

Una variable es un espacio reservado en la memoria del microcontrolador que permite guardar un dato para ser utilizado durante la ejecución del programa. Lo más importante: Ese dato puede cambiar a lo largo del tiempo, según lo que el sistema lea o necesite calcular.
En otras palabras, una variable es como una etiqueta que le ponemos a un cajón donde guardamos información temporal, que luego podemos consultar, modificar o comparar.

¿Para qué usamos variables en robótica?
En programación aplicada a sistemas físicos, como los robóticos, las variables son fundamentales para:

📥 Leer datos de sensores y almacenarlos para analizarlos o tomar decisiones:

```
int temperatura = analogRead(A1);  // Guarda la lectura del sensor de temperatura
```

🔄 Controlar ciclos, repeticiones o conteos:

```
int contador = 0;
contador = contador + 1;  // Incrementa el valor en cada vuelta del loop
```

📊 Comparar valores y definir acciones en estructuras condicionales:

```
if (temperatura > 30) {
  digitalWrite(led, HIGH);  // Enciende un LED si hay calor
}
```

📌 Modelar estados del sistema (por ejemplo, ON/OFF, abierto/cerrado, encendido/apagado):

```
boolean estadoMotor = false;
```

💻 **Ejemplo práctico:**

```
int luz = analogRead(A0);  // Guarda el valor del sensor de luz en la variable "luz"
```

int: indica que es una variable de tipo entero (sin decimales).

luz: es el nombre de la variable, elegido por el programador.

analogRead(A0): lee el valor del sensor conectado al pin A0.

El valor queda guardado en "luz" y puede usarse más adelante en el programa.

💬 **Buenas prácticas al usar variables**
Usar nombres claros y descriptivos: temperatura, estadoBoton, distanciaMedida.
Declararlas al comienzo del programa o en setup() si son de uso global.
Elegir el tipo de dato adecuado (entero, decimal, lógico).
Comentar el código para explicar qué guarda cada variable.

✅ Pensar en variables es pensar en sistemas dinámicos: Si todo el tiempo cambia el entorno, el programa también debe poder adaptarse a esos cambios, y eso solo es posible guardando y manejando datos en tiempo real.

🔒 **2. ¿Qué es una constante?**
Una constante es un espacio en la memoria, similar a una variable, pero con una diferencia fundamental: su valor no cambia durante toda la ejecución del programa. Desde que se declara, mantiene el mismo dato de forma fija.

En programación, las constantes se utilizan para representar datos que no deben modificarse, como por ejemplo:

El número de pin al que está conectado un LED o un botón.
Umbrales fijos de sensores (como un límite de humedad o luz).
Tiempos definidos que se repiten siempre igual (por ejemplo, 1000 ms de espera).

¿Por qué usar constantes en vez de escribir directamente los números?
Aunque podríamos escribir directamente los valores (como 13 o 1000), usar constantes nos permite:

📖 **Hacer el código más legible:** ledRojo es más fácil de entender que 13.
🛠️ **Facilitar la edición del programa:** Si cambiamos el pin, solo lo hacemos en un lugar.
🔐 **Evitar errores accidentales:** al ser inmodificable, una constante no puede sobre escribirse por accidente.

💻 **Ejemplo práctico:**

```
const int ledRojo = 13;  // Declara una constante entera llamada "ledRojo" con valor 13
```

Luego podemos usarla en todo el programa:

```
pinMode(ledRojo, OUTPUT);          // Configura el pin como salida
digitalWrite(ledRojo, HIGH);       // Enciende el LED
```

Esto tiene las mismas funciones que escribir 13 directamente, pero es más claro y más seguro.

¿Qué significa const int?

const → indica que la variable es constante (su valor no cambia).
int → indica que es de tipo entero (número sin decimales).
ledRojo → es el nombre de la constante.
13 → es el valor que se guarda de forma fija.

Comparación

Característica	Variable	Constante
¿Cambia su valor?	Sí	No
¿Cuándo se usa?	Cuando necesito modificar o leer datos	Para definir referencias fijas
Ejemplo de uso	Valor de un sensor	Número de un pin
Declaración	int humedad = 0;	const int boton = 2;

🛠️ **Tipos comunes de datos en Arduino**

Tipo	Significado	Ejemplo de uso
int	Números enteros	int contador = 0;
float	Números con decimales	float temperatura = 23.5;
boolean	Solo dos valores: true o false	boolean estado = true;

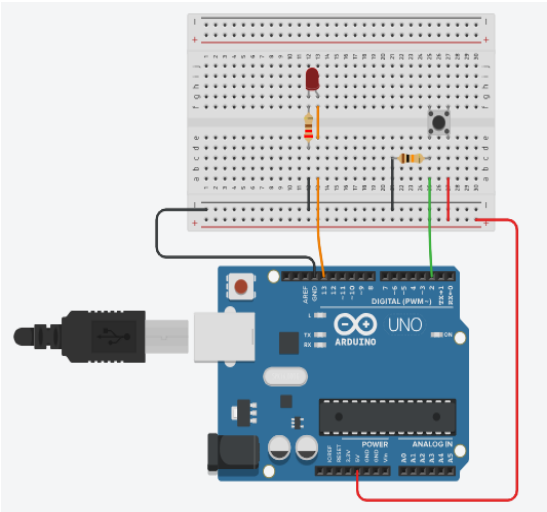
Veamos el ejemplo de la lección 7. Encender un LED al presionar un botón

🎯 **Objetivo:**
Cuando el usuario presiona un botón, se enciende un LED.
Cuando suelta el botón, el LED se apaga.

Componentes:
Arduino Uno
1 botón (pulsador)
1 resistencia de 10 kΩ (pull-down)
1 LED
1 resistencia de 220 Ω (protección para el LED)
Cables + protoboard

🔌 Conexiones físicas:

Elemento	Conexión
Botón (pata 1)	Pin digital **2** de Arduino
Botón (pata 2)	**+5 V**
Resistencia de 10 kΩ	Entre **pin 2 y GND** (pull-down)
LED (ánodo, pata larga)	Pin digital **13**
LED (cátodo, pata corta)	Resistencia de **220 Ω** → GND




```arduino
void setup() //Punto de partida del programa
{
  pinMode(2, INPUT);     // Configura el pin 2 como entrada digital (donde se conecta el botón)
  pinMode(13, OUTPUT);   // Configura el pin 13 como salida digital (donde se conecta el LED)
}

void loop() //Comportamiento dinámico del sistema
{
  // Lee el estado del botón (pin 2)
  // Si está presionado, el valor será HIGH (5 V)
  if (digitalRead(2) == HIGH) {
    digitalWrite(13, HIGH);  // Enciende el LED (pone el pin 13 en estado alto)
  } else {
    digitalWrite(13, LOW);   // Apaga el LED (pone el pin 13 en estado bajo)
  }
}
```

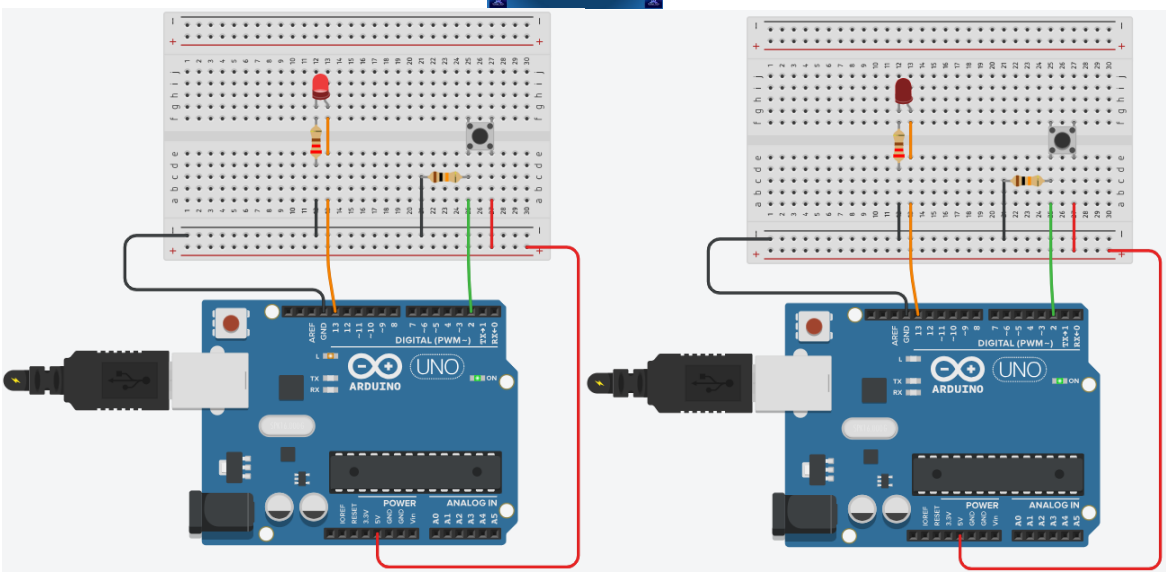
Ejemplo práctico – LED con botón usando constante y variable

```arduino
const int boton = 2;      // Declara una constante entera: el botón está conectado al pin digital 2
const int led = 13;       // Declara una constante entera: el LED está conectado al pin digital 13
int estadoBoton = 0;      // Declara una variable entera que guardará el estado del botón (0 o 1)

void setup() {
  pinMode(boton, INPUT);  // Configura el pin 2 (botón) como entrada digital
  pinMode(led, OUTPUT);   // Configura el pin 13 (LED) como salida digital
}

void loop() {
  estadoBoton = digitalRead(boton);  // Lee el estado del botón (HIGH o LOW) y lo guarda en la variable

  if (estadoBoton == HIGH) {         // Si el botón está presionado (5 V en el pin)
    digitalWrite(led, HIGH);         // Enciende el LED (pone 5 V en el pin 13)
  } else {                           // Si el botón NO está presionado
    digitalWrite(led, LOW);          // Apaga el LED (pone 0 V en el pin 13)
  }
}
```



https://www.tinkercad.com/things/hMhxMTZEqNe-variables-y-constantes

¿Qué enseña este código?
Cómo declarar y usar constantes para entradas y salidas.
Cómo guardar el estado de un botón en una variable.
Cómo usar una estructura condicional **if / else** para tomar decisiones. (Veremos más adelante)
Cómo se relacionan los conceptos de entrada, procesamiento y salida en un programa robótico.

🎓 **Aplicación en robótica educativa**
Ayuda a que los estudiantes entiendan qué componente está conectado a qué pin.
Fomenta la escritura ordenada y profesional del código.
Prepara para proyectos más grandes, donde mantener la claridad del código es fundamental.

💡 ***Enseñar a usar constantes desde el inicio es enseñar buenas prácticas técnicas, clave para la comprensión y el trabajo en equipo.***

📖 **Lectura reflexiva: Nombrar para pensar, guardar para decidir**
En el mundo físico, reconocemos elementos por su función y su ubicación: una llave, un interruptor, un sensor de movimiento. En la programación, hacemos lo mismo, pero con nombres escritos en un código: Llamamos boton al pin que recibe una señal de entrada, **ledRojo** al que emite luz, o **estadoBoton** a la información que queremos evaluar. Esa forma de nombrar y almacenar datos es lo que permite a un programa "entender" y actuar con lógica.
Enseñar el uso de variables y constantes no es solo una cuestión técnica. Es enseñar a pensar cómo se construye una decisión, cómo se organiza un sistema para responder a su entorno, y cómo evitar errores a través del orden y la claridad.
Una constante como **const int led = 13**; puede parecer algo simple, pero representa una idea potente: ese componente no cambiará, es fijo, es parte de la arquitectura del sistema. En cambio, una variable como **int sensor = analogRead(A0);** refleja lo contrario: Un dato que se mueve, que se adapta, que necesita ser interpretado.
Comprender esta diferencia no es un detalle menor. Es la base para pensar técnicamente un sistema automatizado, donde la información que entra (de sensores o del entorno) debe ser guardada, procesada y usada con precisión.
Como docentes, tenemos la oportunidad de mostrar que la programación no se trata solo de escribir comandos, sino de estructurar el pensamiento técnico. Y eso empieza, simplemente, por darle nombre a lo que queremos controlar, y guardar lo que queremos usar para decidir.

🛠️ Pensar con variables es pensar en cambio. Pensar con constantes es pensar en estructura. Programar es saber cuándo usar cada una.