SEMANA 3

📘 Lectura Comprende-Todo: Temas de Automatización y Actuadores

1. Introducción a la Automatización

La automatización es el uso de tecnología para controlar procesos industriales, mecánicos o informáticos con mínima intervención humana. Su objetivo principal es aumentar la eficiencia, precisión y seguridad en tareas repetitivas o complejas. En la actualidad, la automatización se encuentra presente en fábricas, hogares inteligentes, sistemas de riego, control de acceso, monitoreo ambiental, entre muchos otros campos.

Gracias a dispositivos como Arduino, ahora es posible desarrollar soluciones de automatización accesibles, económicas y personalizables. Esto ha permitido que tanto estudiantes como ingenieros puedan experimentar con sistemas reales de control automático.

---

2. Usos de la Automatización

La automatización tiene aplicaciones muy variadas:

- **En la industria:** Control de líneas de producción, manejo de maquinaria pesada.
- **En el hogar:** Sistemas de iluminación inteligente, control de temperatura, persianas automáticas.
- **Agricultura:** Sistemas de riego automático, medición de humedad del suelo, control de invernaderos.
- **Seguridad:** Sistemas de alarma, control de acceso biométrico, cámaras inteligentes.
- **Transporte:** Vehículos autónomos, drones, sistemas de navegación.
- **Energía:** Gestión de paneles solares, control de baterías, optimización del consumo eléctrico.

Arduino, por ejemplo, permite integrar sensores y actuadores para construir estos sistemas con bajo costo y gran flexibilidad.

---

3. Interrupciones en Arduino para Automatización

Las interrupciones son eventos que detienen temporalmente la ejecución del programa principal para atender una tarea urgente o crítica. En Arduino, esto es útil cuando necesitas responder a cambios externos sin tener que estar constantemente revisando el estado de un pin (lo cual sería ineficiente).

Por ejemplo, si deseas contar cuántas veces se pulsa un botón, puedes usar una interrupción para detectar ese evento exacto, sin necesidad de usar bucles continuos (`while` o `for`) que consumen recursos.

Arduino UNO dispone de dos pines dedicados a interrupciones externas: el **pin 2** e **interrupt 0**, y el **pin 3** e **interrupt 1**.

---

4. Tipos de Interrupciones en Arduino

En Arduino puedes configurar las interrupciones para que se activen ante diferentes tipos de señales:

- **RISING:** Se activa cuando el nivel del pin cambia de LOW a HIGH (flanco ascendente).
- **FALLING:** Se activa cuando el nivel del pin cambia de HIGH a LOW (flanco descendente).
- **CHANGE:** Se activa cuando hay cualquier cambio en el estado del pin (ya sea de HIGH a LOW o viceversa).
- **LOW:** Se activa mientras el pin esté en estado LOW.
- **HIGH:** Se activa mientras el pin esté en estado HIGH (solo disponible en algunas placas, no en Arduino UNO).

Cada tipo de interrupción tiene su uso específico dependiendo de lo que quieras lograr en tu proyecto.

---

5. ¿Reemplaza la Automatización al Factor Humano?

No, la automatización no elimina completamente la necesidad del factor humano, aunque sí reduce la intervención directa en tareas repetitivas o peligrosas. El rol del humano evoluciona hacia funciones más estratégicas, como supervisión, mantenimiento, programación y toma de decisiones críticas.

Por ejemplo, en una planta industrial automatizada, los operarios no están manipulando manualmente las máquinas, pero sí están monitoreando su funcionamiento, realizando ajustes, corrigiendo errores y asegurándose de que todo funcione correctamente.

También hay áreas donde la automatización no puede reemplazar completamente al ser humano, como en labores que requieren empatía, juicio ético o adaptabilidad en entornos impredecibles.

---

6. Comparación de Métodos: millis() vs. Interrupciones

En Arduino, existen varias formas de gestionar el tiempo y los eventos. Dos métodos comunes son el uso de `millis()` y las interrupciones.

Uso de `millis()`

La función `millis()` devuelve el número de milisegundos transcurridos desde que el Arduino comenzó a ejecutar el programa. Es útil para crear temporizadores no bloqueantes, es decir, que no detienen la ejecución del resto del código.

Por ejemplo, puedes hacer parpadear un LED cada cierto tiempo sin usar `delay()`, lo que permite ejecutar otras acciones simultáneamente.

```
unsigned long previousMillis = 0;
const long interval = 1000;

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // Código a ejecutar cada segundo
    previousMillis = currentMillis;
  }
}
```

Sin embargo, `millis()` no responde a eventos externos; solo funciona basado en el tiempo transcurrido.

Uso de Interrupciones

Las interrupciones, por otro lado, responden a eventos externos (como pulsar un botón o recibir una señal). Son ideales para tareas que deben ejecutarse de forma inmediata y precisa.

Por ejemplo, puedes usar una interrupción para registrar el momento exacto en que un sensor detecta movimiento, sin necesidad de estar preguntando constantemente por su estado.

```
volatile int state = LOW;

void setup() {
  pinMode(13, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(2), blink, RISING);
}

void blink() {
  state = !state;
}

void loop() {
  digitalWrite(13, state);
}
```

Conclusión:

- Usa `millis()` cuando necesites acciones basadas en el tiempo.
- Usa interrupciones cuando debas responder a eventos externos con precisión.

---

7. ¿Qué es un Actuador?

Un actuador es un dispositivo que convierte energía (eléctrica, hidráulica, neumática) en movimiento físico u otra acción tangible. En sistemas automatizados, los actuadores son responsables de interactuar con el mundo real: encender luces, mover brazos robóticos, abrir puertas, girar motores, etc.

Los actuadores pueden ser lineales (movimiento rectilíneo) o rotativos (movimiento circular), y pueden ser controlados mediante microcontroladores como Arduino.

---

8. Actuadores Eléctricos

Los actuadores eléctricos utilizan corriente eléctrica como fuente de energía. Algunos ejemplos comunes incluyen:

- **Relés:** Interruptores controlados eléctricamente que permiten manejar cargas de alta potencia.
- **Servomotores:** Motores que permiten un control preciso del ángulo de giro.
- **Motores DC:** Motores simples que giran continuamente al aplicar voltaje.
- **Electroválvulas:** Dispositivos que controlan el flujo de líquidos o gases.
- **Actuadores lineales eléctricos:** Generan movimiento lineal mediante un motor y un sistema de tornillo.

Estos actuadores son ampliamente usados en proyectos de automatización porque son fáciles de controlar con Arduino y ofrecen buen rendimiento.

---

9. Tipos de Motores

Los motores son uno de los actuadores más utilizados en automatización. Existen varios tipos, cada uno con características distintas:

Motor DC (Corriente Directa)

Es el tipo más básico. Gira continuamente cuando se le aplica voltaje. La velocidad puede controlarse con PWM (Pulse Width Modulation). Ideal para aplicaciones donde se necesita rotación continua, como carros robot.

Motor Paso a Paso (Stepper)

Este motor avanza en pasos discretos, lo que permite un control muy preciso de la posición. Se utiliza comúnmente en impresoras 3D, CNC y robots. Requiere un controlador especializado.

Servomotor

Un servomotor contiene dentro un motor DC, un circuito de control y un potenciómetro. Permite controlar el ángulo de rotación (por ejemplo, de 0° a 180°) con alta precisión. Muy usado en brazos robóticos, direcciones de vehículos y mecanismos de control angular.

---

10. Aplicaciones de los Motores en Automatización

Cada tipo de motor tiene su lugar en la automatización:

- **Motor DC:** Ideal para movimientos continuos como cintas transportadoras, ventiladores, robots móviles.
- **Motor Stepper:** Perfecto para posicionamiento preciso como en impresoras 3D, cortadoras láser y fresadoras CNC.
- **Servomotor:** Excelente para movimientos angulares controlados, como en brazos robóticos, dirección de autos RC, control de válvulas y antenas.

Con Arduino, puedes controlar estos motores usando componentes como drivers de motor (L298N, L293D, A4988) o shields especializados.

🔗 Referencias

- Arduino Official Reference - Interrupciones
- Actuadores Eléctricos – Conceptos Básicos
- Diferencia entre millis() e interrupciones
- Tipos de Motores Eléctricos
