Materia:

DISEÑO ELECTRÓNICO BASADO EN SISTEMAS EMBEBIDOS

Alumno:

Posadas Pérez Isaac Sayeg

Paniagua Rico Juan Julian

García Azzúa Jorge Roberto

Grado y grupo:

8°G

Profesor:

Garcia Ruiz Alejandro Humberto

Tarea 6:

Inst: Delay propósito, sintaxis, ejemplos, etc.

Instrucciones para el uso de `delay` en Arduino: Propósito, Sintaxis y Ejemplos

La función `delay` es una de las funciones más básicas y utilizadas en Arduino. Permite pausar la ejecución del programa durante un período específico de tiempo, lo que es útil en una variedad de situaciones, como controlar la temporización de eventos, la interacción con el usuario o la sincronización de dispositivos. A continuación, exploraremos el propósito de `delay`, su sintaxis, y proporcionaremos ejemplos prácticos de su uso.

1. Propósito de `delay`

El propósito principal de la función `delay` es introducir una pausa en el flujo de ejecución de un programa. Esto se utiliza para:

- **Controlar la temporización**: Permite que ciertas acciones se realicen en intervalos específicos. Por ejemplo, encender y apagar un LED cada segundo.
- **Sincronización de eventos**: Ayuda a coordinar la interacción entre diferentes componentes del circuito, como sensores y actuadores.
- **Interacciones con el usuario**: Puede usarse para proporcionar tiempo suficiente para que los usuarios lean mensajes en el monitor serie o respondan a entradas.

2. Sintaxis de `delay`

La sintaxis de la función `delay` es bastante sencilla:

delay(ms);

- `ms`: Especifica el tiempo de espera en milisegundos (ms). Un segundo equivale a 1000 milisegundos. Por lo tanto, si deseas que el programa se detenga durante 2 segundos, deberías usar `delay(2000);`.

La función `delay` detiene la ejecución de todo el programa durante el tiempo especificado. Esto significa que no se pueden realizar otras tareas o interacciones durante este período.

3. Ejemplos de Uso de `delay`

3.1. Ejemplo Básico: Encender y Apagar un LED

A continuación se muestra un ejemplo simple que utiliza `delay` para encender y apagar un LED conectado al pin 9, alternando cada segundo.

```
void setup() {
  pinMode(9, OUTPUT); // Configura el pin 9 como salida
}

void loop() {
  digitalWrite(9, HIGH); // Enciende el LED
  delay(1000);           // Espera 1 segundo
  digitalWrite(9, LOW);  // Apaga el LED
  delay(1000);           // Espera 1 segundo
}
```

3.2. Ejemplo: Monitoreo de un Sensor

En este ejemplo, se utiliza `delay` para leer un sensor de temperatura y mostrar el valor en el monitor serie cada 2 segundos.

```
void setup() {
  Serial.begin(9600); // Inicia la comunicación serie a 9600 bps
}

void loop() {
  int temperatura = analogRead(A0); // Lee el valor del sensor de temperatura
  Serial.println(temperatura);      // Imprime el valor en el monitor serie
  delay(2000);                      // Espera 2 segundos antes de la siguiente lectura
}
```

3.3. Ejemplo: Interacción con el Usuario

Este ejemplo muestra cómo utilizar `delay` para proporcionar tiempo a un usuario para leer un mensaje en el monitor serie antes de continuar.

```
void setup() {
```

```
    Serial.begin(9600); // Inicia la comunicación serie a 9600 bps
    Serial.println("Iniciando el sistema...");
    delay(3000); // Espera 3 segundos para que el usuario lea el mensaje
}

void loop() {
    Serial.println("El sistema está en funcionamiento.");
    delay(5000); // Espera 5 segundos antes de imprimir el siguiente mensaje
}
```

4. Consideraciones sobre el Uso de `delay`

4.1. Limitaciones de `delay`

- **No es Multitarea**: Dado que `delay` detiene la ejecución de todo el programa, no es ideal para aplicaciones que requieren una respuesta rápida o procesamiento continuo de datos, como el control de motores o la lectura de sensores en tiempo real.
- **Alternativas**: Para aplicaciones que requieren temporización más compleja o no bloqueante, se recomienda utilizar la función `millis()`, que permite medir el tiempo transcurrido sin detener el programa. Esto se puede hacer implementando un sistema de temporización basado en intervalos.

4.2. Uso de `millis()`

La función `millis()` devuelve el número de milisegundos desde que la placa Arduino comenzó a ejecutar el programa. Se puede utilizar para crear temporizadores que no bloqueen la ejecución del programa.

```
unsigned long tiempoAnterior = 0; // Variable para almacenar el tiempo anterior

void loop() {
    unsigned long tiempoActual = millis(); // Captura el tiempo actual

    if (tiempoActual - tiempoAnterior >= 1000) { // Si han pasado 1 segundo
        digitalWrite(9, !digitalRead(9)); // Cambia el estado del LED
        tiempoAnterior = tiempoActual; // Actualiza el tiempo anterior
    }
}
```

5. Conclusiones

La función `delay` es una herramienta en la programación de Arduino para introducir pausas en la ejecución de un programa. Su uso es común en ejemplos simples y aplicaciones de temporización. Sin embargo, para aplicaciones más complejas que requieren multitarea, es recomendable considerar el uso de `millis()` para manejar temporizaciones sin bloquear el flujo del programa.

Bibliografía

1. Banzi, M., & Shiloh, M. (2014). *Getting Started with Arduino*. Maker Media, Inc.
    - Este libro proporciona una introducción a la plataforma Arduino, incluyendo el uso de funciones como `delay`.
2. Monk, S. (2015). *Programming Arduino: Getting Started with Sketches*. McGraw-Hill Education.
    - Un recurso que presenta ejemplos prácticos sobre la programación en Arduino, incluyendo temporización con `delay`.
3. Arduino. (n.d.). *Arduino Reference*. Retrieved from https://www.arduino.cc/reference/en/
    - Documentación oficial de Arduino que detalla la función `delay` y su uso.
4. Adafruit. (n.d.). *Learning System*. Retrieved from https://learn.adafruit.com/
    - Un recurso en línea que ofrece tutoriales sobre programación de Arduino y ejemplos prácticos.
5. Simon Monk. (2014). *Arduino Cookbook*. O'Reilly Media.
    - Un libro que ofrece una colección de recetas y ejemplos prácticos para trabajar con Arduino en diferentes proyectos.