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 4:

Instrucciones para trabajar con escritura/lectura de señales digitales en arduino

Instrucciones para Trabajar con Escritura/Lectura de Señales Digitales en Arduino

La capacidad de trabajar con señales digitales es una de las características fundamentales de la plataforma Arduino. Las señales digitales son aquellas que tienen solo dos estados posibles: alto (HIGH) y bajo (LOW). Este concepto es esencial para interactuar con una variedad de componentes electrónicos, como interruptores, sensores, y actuadores. En esta investigación, exploraremos cómo leer y escribir señales digitales en Arduino, proporcionando ejemplos prácticos y explicaciones detalladas.

1. Introducción a las Señales Digitales

Las señales digitales son representaciones de información en formato binario. En el contexto de Arduino, las señales digitales se utilizan para controlar y monitorear dispositivos. Un pin digital de Arduino puede estar en uno de dos estados:

- **HIGH (Alto)**: El pin está en un estado de voltaje alto, generalmente cerca de 5V en la mayoría de las placas Arduino.
- **LOW (Bajo)**: El pin está en un estado de voltaje bajo, aproximadamente 0V.

2. Configuración de Pines Digitales

Antes de trabajar con señales digitales, es necesario configurar el pin como entrada o salida. Esto se realiza en la función `setup()` del programa.

2.1. Configuración de un Pin como Salida

Para usar un pin digital como salida, se utiliza la función `pinMode(pin, mode)`, donde `pin` es el número del pin y `mode` es `OUTPUT`.

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

2.2. Configuración de un Pin como Entrada

Para usar un pin digital como entrada, se utiliza la misma función `pinMode(pin, mode)` con el modo `INPUT`.

```
void setup() {
  pinMode(2, INPUT); // Configura el pin 2 como entrada
}
```

3. Escritura de Señales Digitales

La escritura de señales digitales se realiza mediante la función `digitalWrite(pin, value)`, donde `value` puede ser `HIGH` o `LOW`.

3.1. Ejemplo de Control de un LED

Un ejemplo común es controlar un LED. Aquí se muestra cómo encender y apagar un LED conectado al pin 9.

```
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
}
```

4. Lectura de Señales Digitales

La lectura de señales digitales se realiza utilizando la función `digitalRead(pin)`, que devuelve el estado del pin configurado como entrada. Esta función puede devolver `HIGH` o `LOW`.

4.1. Ejemplo de Lectura de un Botón

Un ejemplo común es leer el estado de un botón. A continuación se muestra cómo leer un botón conectado al pin 2 y encender un LED en el pin 9 cuando el botón está presionado.

```
void setup() {
    pinMode(9, OUTPUT);    // Configura el pin 9 como salida
    pinMode(2, INPUT);     // Configura el pin 2 como entrada
}

void loop() {
    int estadoBoton = digitalRead(2); // Lee el estado del botón
    if (estadoBoton == HIGH) {        // Si el botón está presionado
        digitalWrite(9, HIGH);        // Enciende el LED
    } else {
        digitalWrite(9, LOW);         // Apaga el LED
    }
}
```

5. Consideraciones al Trabajar con Señales Digitales

5.1. Resistencia Pull-Down y Pull-Up

Al conectar un botón a un pin digital, es importante evitar lecturas erróneas. Esto se puede lograr utilizando resistencias pull-up o pull-down.

- **Resistencia Pull-Up**: Se conecta entre el pin y Vcc (5V). Cuando el botón no está presionado, el pin lee `HIGH`. Cuando el botón se presiona, el pin se conecta a GND y lee `LOW`.
- **Resistencia Pull-Down**: Se conecta entre el pin y GND. Cuando el botón no está presionado, el pin lee `LOW`. Cuando el botón se presiona, el pin se conecta a Vcc y lee `HIGH`.

Arduino también tiene resistencias pull-up internas que se pueden habilitar utilizando `pinMode(pin, INPUT_PULLUP)`.

5.2. Debouncing

Cuando se presiona un botón, puede haber un ruido eléctrico que causa múltiples lecturas rápidas (rebote). Para manejar esto, se puede implementar un retardo en el programa después de la lectura del botón o utilizar técnicas más avanzadas, como el uso de interruptores.

6. Conclusiones

La capacidad de leer y escribir señales digitales en Arduino es esencial para la creación de proyectos interactivos y la manipulación de dispositivos electrónicos. Con una comprensión clara de cómo configurar pines, escribir y leer señales, los usuarios pueden desarrollar una variedad de aplicaciones, desde simples circuitos con LEDs hasta sistemas más complejos que involucran múltiples sensores y actuadores. Las prácticas adecuadas, como el uso de resistencias pull-up y técnicas de debouncing, mejorarán la fiabilidad de las lecturas y el funcionamiento de los proyectos.

Bibliografía

1. Banzi, M., & Shiloh, M. (2014). *Getting Started with Arduino*. Maker Media, Inc.
    - Un libro introductorio sobre Arduino, que incluye ejemplos y prácticas sobre lectura y escritura de señales digitales.
2. Monk, S. (2015). *Programming Arduino: Getting Started with Sketches*. McGraw-Hill Education.
    - Proporciona una comprensión profunda de la programación de Arduino y ejemplos prácticos sobre señales digitales.
3. Arduino. (n.d.). *Arduino Reference*. Retrieved from https://www.arduino.cc/reference/en/
    - Documentación oficial de Arduino que detalla las funciones para trabajar con señales digitales.
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 de proyectos.
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.