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 10:

Limpieza de Datos

Limpieza de Datos

1. Introducción

La limpieza de datos es un proceso esencial en el análisis de datos que consiste en identificar y corregir errores, inconsistencias y valores atípicos para mejorar la calidad de la información recopilada. En proyectos de Arduino, la limpieza de datos es especialmente importante cuando se trabajan con sensores, ya que estos pueden generar mediciones erróneas debido a interferencias, ruido eléctrico o fallas en el hardware.

Este documento explorará qué es la limpieza de datos, por qué es importante en proyectos de Arduino y qué técnicas se pueden emplear para mejorar la precisión de los datos recogidos por sensores.

2. ¿Qué es la Limpieza de Datos?

La limpieza de datos es el proceso de detectar y corregir problemas en los datos para que sean precisos, confiables y utilizables para su análisis o visualización. En un entorno como Arduino, donde los sensores y dispositivos de entrada pueden proporcionar lecturas inexactas, este proceso es crucial para evitar errores en el análisis o en la toma de decisiones basada en los datos recopilados.

Algunas de las anomalías más comunes en los datos incluyen:

- **Valores atípicos (outliers):** Mediciones que están fuera del rango esperado.
- **Datos faltantes:** Sensores que fallan en enviar datos en ciertos momentos.
- **Datos duplicados:** Repeticiones accidentales de valores.
- **Ruido:** Variaciones en los datos causadas por interferencias o fluctuaciones en el entorno.

3. Importancia de la Limpieza de Datos en Arduino

- **Mejora la precisión:** Los sensores pueden presentar errores debido a fluctuaciones de voltaje, ruido eléctrico o mal funcionamiento.
- **Facilita la toma de decisiones:** Datos limpios permiten realizar análisis más precisos y confiables.
- **Evita fallos en sistemas embebidos:** La presencia de datos incorrectos podría hacer que un sistema tome decisiones erróneas.
- **Optimiza el procesamiento de datos:** Un conjunto de datos limpio reduce la carga de procesamiento en microcontroladores con recursos limitados.

Existen varias estrategias para limpiar datos en proyectos de Arduino. A continuación, se presentan las más utilizadas:

4.1. Eliminación de Valores Atípicos

Los valores atípicos pueden ser detectados al comparar las lecturas actuales con rangos esperados.

Ejemplo en Arduino:

```
float temperatura = analogRead(A0) * (5.0 / 1023.0) * 100.0; // Conversión de lectura de sensor a °C

if (temperatura < -10 || temperatura > 50) {
    temperatura = 25; // Se reemplaza por un valor predeterminado si está fuera del rango aceptable
}
Serial.println(temperatura);
```

En este ejemplo, cualquier temperatura menor a -10°C o mayor a 50°C es considerada un error y reemplazada por un valor predeterminado de 25°C.

4.2. Filtrado de Ruido (Promediado de Datos)

Los sensores pueden producir valores fluctuantes debido a interferencias o cambios en el entorno. Para mitigar esto, se pueden aplicar técnicas de filtrado como el **promediado móvil**.

Ejemplo de filtrado de ruido con promedio móvil:

```
const int N = 10; // Número de lecturas a promediar
float suma = 0;
for (int i = 0; i < N; i++) {
    suma += analogRead(A0);
    delay(10);
}
float temperaturaFiltrada = (suma / N) * (5.0 / 1023.0) * 100.0;
Serial.println(temperaturaFiltrada);
```

Aquí se recopilan 10 lecturas del sensor y se promedian para reducir el ruido.

4.3. Manejo de Datos Faltantes

En ocasiones, un sensor puede fallar en proporcionar una lectura. Para manejar estos casos, se puede implementar una verificación de datos faltantes.

Ejemplo de detección de datos faltantes:

```
float temperatura = analogRead(A0);

if (isnan(temperatura)) {
    temperatura = 25.0; // Se asigna un valor predeterminado si el sensor no responde
}
Serial.println(temperatura);
```

La función `isnan()` verifica si la lectura del sensor es inválida y asigna un valor estándar en su lugar.

4.4. Eliminación de Datos Duplicados

En sistemas que recopilan datos en intervalos regulares, puede ocurrir que ciertos valores se repitan sin cambios. Esto puede ocurrir si el sensor sigue leyendo el mismo valor durante varios ciclos.

Ejemplo de eliminación de datos duplicados:

```
float temperaturaAnterior = 0;
float temperatura = analogRead(A0) * (5.0 / 1023.0) * 100.0;

if (temperatura != temperaturaAnterior) {
    Serial.println(temperatura);
    temperaturaAnterior = temperatura;
}
```

Ac
repetición innecesaria de datos.

4.5. Interpolación para Corregir Datos Perdidos

Si un sensor proporciona datos en intervalos irregulares, se puede usar interpolación lineal para estimar valores perdidos.

Ejemplo de interpolación lineal en Arduino:

```
float x1 = 1, y1 = 20; // Punto previo (Tiempo 1, Temperatura 20)
float x2 = 3, y2 = 25; // Punto siguiente (Tiempo 3, Temperatura 25)
float x = 2; // Momento en el que falta un dato

float yInterpolado = y1 + ((y2 - y1) / (x2 - x1)) * (x - x1);
Serial.println(yInterpolado);
```

Este método estima un valor faltante basándose en dos valores conocidos.

5. Aplicación de la Limpieza de Datos en un Proyecto Real

Supongamos que se desea monitorear la temperatura y humedad en un invernadero usando sensores DHT11 y enviar los datos a una plataforma en la nube. Un código en Arduino que implemente limpieza de datos podría verse así:

```
#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
    Serial.begin(9600);
    dht.begin();
}

void loop() {
    float temperatura = dht.readTemperature();
```

```
    // Verificar si los valores son válidos
    if (isnan(temperatura) || temperatura < 0 || temperatura > 50) {
        temperatura = 25; // Valor predeterminado
    }
    if (isnan(humedad) || humedad < 10 || humedad > 90) {
        humedad = 50; // Valor predeterminado
    }

    Serial.print("Temperatura: "); Serial.println(temperatura);
    Serial.print("Humedad: "); Serial.println(humedad);

    delay(2000);
}
```

6. Conclusión

La limpieza de datos es una tarea necesaria en el procesamiento de información en Arduino. Aplicando técnicas como la eliminación de valores atípicos, el filtrado de ruido y el manejo de valores faltantes, se pueden obtener datos más confiables y precisos para la toma de decisiones o el análisis posterior.

7. Bibliografía

1. Banzi, M., & Shiloh, M. (2014). *Getting Started with Arduino*. Maker Media, Inc.
2. Monk, S. (2015). *Programming Arduino: Getting Started with Sketches*. McGraw-Hill Education.
3. Arduino. (n.d.). *Arduino Reference*. Recuperado de https://www.arduino.cc/reference/en/
4. Adafruit. (n.d.). *Learning System*. Recuperado de https://learn.adafruit.com/
5. Simon Monk. (2014). *Arduino Cookbook*. O'Reilly Media.