**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 9:**

Preprocesamiento de datos

# **Preprocesamiento de Datos en Proyectos de Arduino**

El preprocesamiento de datos es una etapa fundamental en el análisis de datos, que implica la preparación de los datos para su posterior análisis o modelado. En proyectos de Arduino, el preprocesamiento se convierte en una necesidad, especialmente cuando se trabaja con datos recogidos de sensores o dispositivos. Esta investigación abordará el concepto de preprocesamiento de datos, sus etapas, técnicas comunes y su aplicación en proyectos de Arduino.

## **1. Concepto de Preprocesamiento de Datos**

El preprocesamiento de datos se refiere a una serie de pasos realizados para limpiar, transformar y preparar los datos antes de que sean analizados. Estos pasos son cruciales para garantizar que los datos sean precisos, consistentes y adecuados para el análisis posterior. En el contexto de Arduino, el preprocesamiento puede incluir la recolección de datos de sensores, la limpieza de esos datos y su transformación para facilitar el análisis.

## **2. Importancia del Preprocesamiento de Datos**

* **Mejora de la Calidad de los Datos**: El preprocesamiento ayuda a identificar y corregir errores en los datos, lo que mejora la calidad de la información utilizada para el análisis.
* **Eliminación de Ruido**: Los datos recopilados a menudo contienen ruido, es decir, valores no representativos. El preprocesamiento ayuda a filtrar este ruido.
* **Facilitación del Análisis**: Preparar los datos en un formato adecuado facilita el análisis y la modelización, lo que permite obtener mejores resultados.

## **3. Etapas del Preprocesamiento de Datos**

El preprocesamiento de datos generalmente incluye varias etapas, que pueden variar según el contexto y los requisitos del proyecto:

### **3.1. Recolección de Datos**

La primera etapa del preprocesamiento es la recolección de datos, que en el caso de Arduino implica obtener datos de sensores y otros dispositivos. Esta etapa puede involucrar la configuración del hardware y la programación para leer datos.

**Ejemplo**:

| float temperatura = analogRead(pinSensor); // Lectura de datos de un sensor de temperatura |
| --- |

### **3.2. Limpieza de Datos**

La limpieza de datos implica identificar y corregir errores o inconsistencias en los datos. Esto puede incluir la eliminación de valores atípicos, la corrección de errores de lectura y el manejo de valores faltantes.

**Ejemplo**:

| if (temperatura < 0 || temperatura > 100) {  temperatura = 0; // Reemplaza valores atípicos con un valor predeterminado  } |
| --- |

### **3.3. Transformación de Datos**

La transformación de datos implica cambiar el formato o la escala de los datos para que sean adecuados para el análisis. Esto puede incluir la normalización, la discretización y la conversión de unidades.

**Ejemplo**:

| float temperaturaCelsius = (temperatura \* 5.0 / 1024.0) \* 100.0; // Conversión a grados Celsius |
| --- |

### **3.4. Integración de Datos**

Si se están utilizando múltiples fuentes de datos, la integración de datos implica combinar estos datos en un conjunto cohesivo. Esto es especialmente relevante si se están utilizando varios sensores.

**Ejemplo**:

| float humedad = analogRead(pinSensorHumedad); // Lectura de datos de un sensor de humedad  float temperaturaHumedad = (temperaturaCelsius + humedad) / 2; // Integración simple de datos |
| --- |

## **4. Técnicas Comunes de Preprocesamiento**

### **4.1. Filtrado de Ruido**

El filtrado de ruido es una técnica utilizada para eliminar variaciones no deseadas en los datos. En Arduino, se pueden aplicar técnicas de promediado para suavizar las lecturas de los sensores.

**Ejemplo**:

| float sumaTemperaturas = 0;  for (int i = 0; i < 10; i++) {  sumaTemperaturas += analogRead(pinSensor);  }  float temperaturaFiltrada = sumaTemperaturas / 10; // Promedio de 10 lecturas |
| --- |

### **4.2. Normalización**

La normalización se utiliza para escalar los datos a un rango específico. Esto es útil cuando se combinan diferentes tipos de datos que pueden tener diferentes escalas.

**Ejemplo**:

| float temperaturaNormalizada = (temperaturaCelsius - 0) / (100 - 0); // Normaliza entre 0 y 1 |
| --- |

### **4.3. Manejo de Valores Faltantes**

En ocasiones, los datos pueden estar incompletos. El manejo de valores faltantes implica decidir cómo tratar estos casos, ya sea eliminando la observación o imputando un valor.

**Ejemplo**:

| if (temperatura == -1) {  temperatura = 25; // Imputación de un valor predeterminado  } |
| --- |

## **5. Aplicación en Proyectos de Arduino**

El preprocesamiento de datos es esencial en proyectos de Arduino que involucran la recopilación de datos de sensores, como estaciones meteorológicas, sistemas de monitoreo de salud y proyectos de automatización del hogar. Un ejemplo práctico podría ser un sistema de monitoreo ambiental que recoge datos de temperatura y humedad, los limpia y los transforma antes de enviarlos a una base de datos o visualizarlos en una interfaz de usuario.

**Ejemplo de Proyecto**:

| void setup() {  Serial.begin(9600);  }  void loop() {  float temperatura = analogRead(pinSensor); // Lectura de temperatura  if (temperatura < 0 || temperatura > 100) {  temperatura = 25; // Manejo de valores atípicos  }  temperatura = (temperatura \* 5.0 / 1024.0) \* 100.0; // Conversión a Celsius  Serial.println(temperatura); // Imprime la temperatura procesada  delay(1000);  } |
| --- |

## **6. Conclusiones**

El preprocesamiento de datos es una etapa crucial en el análisis de datos que asegura la calidad y la precisión de la información utilizada. En proyectos de Arduino, este proceso permite preparar los datos de manera adecuada para su análisis y visualización. Comprender las técnicas de preprocesamiento y aplicarlas correctamente contribuye al éxito de los proyectos de sistemas embebidos.

### **Bibliografía**

1. Banzi, M., & Shiloh, M. (2014). *Getting Started with Arduino*. Maker Media, Inc.  
   * Este libro proporciona una introducción a la plataforma Arduino y su aplicación en la recolección de datos.
2. Monk, S. (2015). *Programming Arduino: Getting Started with Sketches*. McGraw-Hill Education.  
   * Un recurso que presenta ejemplos prácticos sobre la programación en Arduino y el manejo de datos.
3. Arduino. (n.d.). *Arduino Reference*. Retrieved from https://www.arduino.cc/reference/en/  
   * Documentación oficial de Arduino que detalla funciones y métodos para la manipulación de datos.
4. Adafruit. (n.d.). *Learning System*. Retrieved from https://learn.adafruit.com/  
   * Un recurso en línea que ofrece tutoriales sobre programación de Arduino y el manejo de datos.
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, incluyendo el manejo de datos.