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 12:

Graficación de Datos por BoxPlot

Graficación de Datos por BoxPlot

1. Introducción

La graficación de datos es una herramienta fundamental en el análisis estadístico, permitiendo visualizar tendencias, distribuciones y anomalías en conjuntos de datos. Uno de los métodos más utilizados para representar la distribución de datos es el **BoxPlot**, también conocido como diagrama de caja y bigotes. Esta técnica proporciona un resumen visual de la distribución estadística de un conjunto de datos, mostrando la mediana, los cuartiles y los valores atípicos (outliers).

En proyectos con Arduino y otros sistemas embebidos, los **BoxPlots** pueden ayudar a analizar datos de sensores, detectar valores atípicos y mejorar la interpretación de la información recolectada. En este documento, exploraremos los fundamentos de los BoxPlots, su aplicación en la visualización de datos y ejemplos prácticos de implementación.

2. Concepto de BoxPlot

El **BoxPlot** es una representación gráfica que permite visualizar de manera resumida la distribución de un conjunto de datos, proporcionando información sobre la dispersión, la simetría y la presencia de valores atípicos. Para construir un BoxPlot, se consideran cinco valores estadísticos clave que permiten segmentar los datos en diferentes rangos.

El valor mínimo corresponde al dato más bajo dentro del rango permitido sin considerar valores atípicos. El primer cuartil (Q1) representa el percentil 25, lo que significa que el 25% de los datos están por debajo de este valor. La mediana, también conocida como Q2, es el percentil 50 y marca el valor central de los datos cuando están ordenados de menor a mayor. El tercer cuartil (Q3) representa el percentil 75, es decir, el 75% de los datos están por debajo de este valor. Finalmente, el valor máximo es el dato más alto dentro del rango permitido sin considerar outliers.

Los valores atípicos o **outliers** son datos que se encuentran significativamente alejados del resto de los valores y se identifican utilizando el **rango intercuartil (IQR)**, que se calcula como la diferencia entre Q3 y Q1. Para determinar si un valor es un outlier, se emplean los siguientes límites:

$$\text{Límite inferior} = Q1 - 1.5 \times IQR \quad \text{Límite superior} = Q3 + 1.5 \times IQR$$

Los valores que caen fuera de este rango se consideran valores atípicos y se representan como puntos individuales en el BoxPlot, lo que permite identificar posibles anomalías o errores en los datos.

3. Aplicación de BoxPlot en Arduino

En sistemas embebidos como **Arduino**, los BoxPlots pueden ser útiles para:

- Analizar datos de sensores (temperatura, humedad, voltaje, etc.).
- Detectar valores atípicos o fallas en mediciones.
- Comparar distribuciones de datos en diferentes experimentos.
- Identificar variabilidad en lecturas de sensores.

Como Arduino no puede generar gráficos directamente, se pueden enviar datos a Python o MATLAB para su visualización.

Ejemplo: Recolección de datos con Arduino

```
const int sensorPin = A0;
int valores[50];

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

void loop() {
    for (int i = 0; i < 50; i++) {
        valores[i] = analogRead(sensorPin);
        delay(100);
    }

    for (int i = 0; i < 50; i++) {
        Serial.println(valores[i]);
    }
    delay(5000);
}
```

Este código recopila 50 mediciones de un sensor y las envía por el puerto serie para su análisis en Python.

4. Generación de BoxPlot en Python

Una vez que los datos han sido recolectados, podemos utilizar Python para graficar el **BoxPlot**.

```
import serial
import numpy as np
import matplotlib.pyplot as plt

# Leer datos desde el puerto serie
ser = serial.Serial('COM3', 9600)  # Cambia 'COM3' según tu puerto
valores = []

while len(valores) < 50:
    try:
        line = ser.readline().decode('utf-8').strip()
        valores.append(int(line))
    except:
        continue

ser.close()

# Crear BoxPlot
plt.boxplot(valores, vert=True, patch_artist=True, labels=["Sensor A0"])
plt.title("Distribución de datos del sensor")
plt.ylabel("Valor de lectura")
plt.grid(True)
plt.show()
```

En este código, primero se establece la conexión con el puerto serie en el que está conectado Arduino. Luego, se recibe la información enviada por el microcontrolador y se almacena en una lista. Una vez completada la recolección de datos, se cierra la conexión con el puerto serie y se genera un **BoxPlot** utilizando Matplotlib, donde la caja representa la distribución de las lecturas del sensor.

La personalización del gráfico, como el color de la caja y la adición de una cuadrícula, ayuda a mejorar su claridad y facilita la interpretación de los datos.

5. Interpretación del BoxPlot

Al analizar un **BoxPlot**, se pueden extraer varias conclusiones. Si la caja es muy estrecha, indica que los datos tienen poca variabilidad y están concentrados en un rango reducido. Si la caja es ancha, implica que hay una mayor dispersión en los

valores. Cuando aparecen muchos outliers, esto puede sugerir la presencia de ruido en las mediciones o errores en los datos. Si la mediana no está centrada dentro de la caja, es indicativo de una distribución asimétrica en los datos.

Un análisis adecuado de un **BoxPlot** ayuda a tomar decisiones sobre la calidad de los datos, su fiabilidad y la presencia de valores extremos que puedan afectar la interpretación de los resultados.

6. Conclusión

La graficación mediante **BoxPlot** es una herramienta que sirve para poder analizar distribuciones de datos y detectar valores atípicos en mediciones de sensores en proyectos con **Arduino**. Al combinar Arduino con Python, es posible recopilar, procesar y visualizar datos de manera efectiva, mejorando la interpretación de la información obtenida. Comprender la distribución de los datos y sus posibles anomalías es crucial en proyectos de análisis de señales en sistemas embebidos, ya que permite tomar decisiones más informadas y mejorar la precisión de los dispositivos de medición.

7. Bibliografía

1. McKinney, W. (2017). *Python for Data Analysis*. O'Reilly Media.
2. Arduino. (n.d.). *Arduino Reference*. Recuperado de https://www.arduino.cc/reference/en/
3. Matplotlib. (n.d.). *Box plot documentation*. Recuperado de https://matplotlib.org/stable/gallery/statistics/boxplot.html