Informe del Desafio 1 Informatica 2

2024-2

Diego Alejandro Alegria Cano

Felipe Rodas Vasquez

Facultad de Ingeniería, Universidad de Antioquia

Informática II

Dr. Aníbal José Guerra Soler

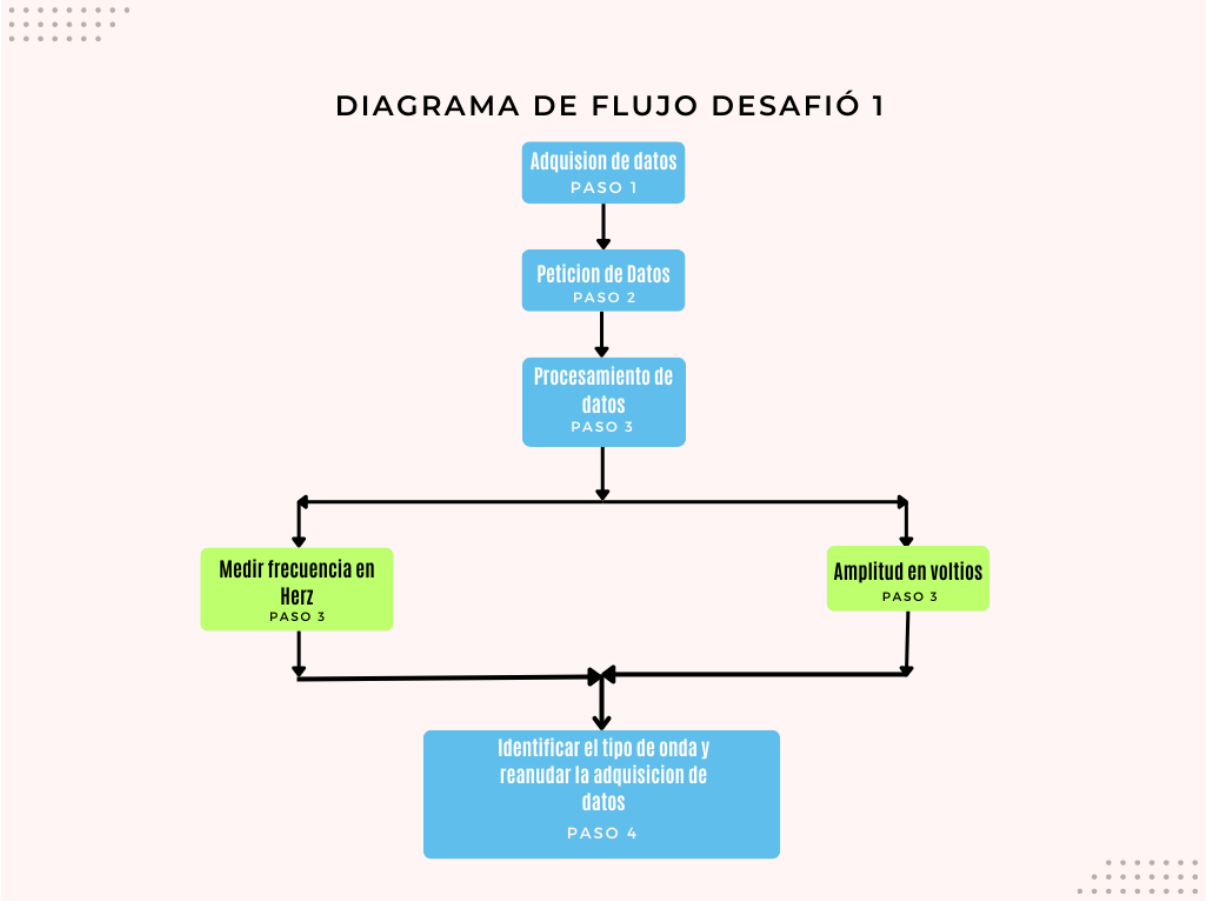
14 de Septiembre

Análisis del problema.

El problema del Desafío 1 consiste en la implementación de un circuito de Arduino compuesto por un Arduino UNO, un generador de señales, una pantalla LCD y dos pulsadores. El funcionamiento del circuito se basa en la generación de señales mediante el generador de señales conectado al Arduino, los valores de la señal se guardaran siempre y cuando el primer pulsador sea presionado. Cuando se activa el segundo pulsador los valores de la señal que fueron almacenados serán procesados y convertidos en la información necesaria(frecuencia en Hertz, amplitud en voltios y el tipo de señal que sea).

Consideramos que una posible solución a este problema es la implementación de arreglos dinámicos unidimensionales, que se puedan redimensionar según la cantidad de valores guardados para ser procesados y posteriormente mostrados en la pantalla LCD.

Esquema.



Pasos del desarrollo	Información
Paso 1. Adquisición de datos	Se recibe la información del generador de señales y ésta es almacenada en un arreglo dinámico unidimensional capaz de redimensionarse cuando esté llegando al final de su capacidad actual.
Paso 2. Petición de datos	La petición de los datos se hará mediante un pulsador, en el momento que se oprima el pulsador se dejará de guardar datos en el arreglo y los datos que ya están en el arreglo serán procesados y utilizados en las funciones para medir la frecuencia y amplitud.
Paso 3. Procesamiento de datos	El procesamiento de datos se basa en calcular mediante teoría de ondas y fórmulas matemáticas la frecuencia de la onda y su amplitud en las medidas correspondientes para cada una Hertz y Voltios.
Paso 4. Identificar el tipo de onda y reanudar la adquisición de datos	Después del procesamiento de datos y de obtener la frecuencia y la amplitud, se identificara el tipo de onda utilizando funciones y arreglos para comparar la señal que tenemos con los tres tipos de señales establecidas o en caso tal que no ser ninguna de las tres decir que es una señal desconocida.

Algoritmos implementados.

```
// Función para redimensionar el arreglo
void redArr(int*& arr, int& capacidad) {
  int nuevaCap = capacidad * 2;
  int* nuevoArr = new int[nuevaCap];
  for (int i = 0; i < capacidad; i++) {
    nuevoArr[i] = arr[i];
  }
  delete[] arr;
  arr = nuevoArr;
  capacidad = nuevaCap;
}
```

- La función **redArr** está diseñada para redimensionar un arreglo dinámico de enteros, duplicando su capacidad. Los parámetros que recibe la función son: Un **puntero** por referencia a un arreglo dinámico de enteros. Se pasa por referencia para modificar el puntero original y hacer que apunte a un nuevo arreglo de mayor capacidad.

  Un **entero** por referencia que representa la capacidad actual del arreglo. Al pasarlo por referencia, se puede actualizar el valor de capacidad fuera de la función. Después de recibir los parámetros la función calcula una nueva capacidad, que es el doble de la actual, reserva memoria para un nuevo arreglo dinámico de enteros con la nueva capacidad, copia los elementos del arreglo original al nuevo arreglo en un bucle, libera la memoria del arreglo original para evitar fugas de memoria, asigna el puntero del nuevo arreglo al puntero **arr**, para que **arr** apunte al nuevo arreglo con mayor capacidad, ya por último la función actualiza el valor de la capacidad para reflejar el nuevo tamaño.

```
// Función para detectar si la señal es cuadrada
bool senalCuadrada(int* arr, int size) {
  int numerosUnicos = 0;
  int unicos[4];

  for (int i = 0; i < size; i++) {
    int j = 0;
    for (j = 0; j < numerosUnicos; j++) {
      if (arr[i] == unicos[j]) {
        j = numerosUnicos;
      }
    }
    if (j == numerosUnicos) {
      unicos[numerosUnicos++] = arr[i];
      if (numerosUnicos > 4) {
        return false;
      }
    }
  }
  return true;
}
```

- La función **senalCuadrada** está diseñada para detectar si una señal representada como un arreglo de enteros puede considerarse "cuadrada" basándose en la información del arreglo. Los parámetros que recibe la función son: Un **puntero** a entero que representa un arreglo de enteros, es decir, la secuencia de valores que forman la señal.

  Un **entero** que indica el tamaño del arreglo, es decir, la cantidad de elementos en el arreglo. Una vez recibidos los parámetros la función Inicializa una variable llamada **numerosUnicos** a 0, que llevará la cuenta de la cantidad de números únicos encontrados en la señal, declara un arreglo **unicos[4]** que podrá almacenar hasta 4 números únicos, luego, la función recorre cada valor en el arreglo **arr** usando un bucle **for**. Si después de recorrer todo el arreglo **arr[]** se han encontrado 4 o menos números únicos, la función retorna **true**, lo que indica que la señal es **cuadrada**.

```
// Función para medir la amplitud
float medirAmplitud(int* arr, int size) {
  int valorMaximo = 0;
  int valorMinimo = 1023;

  for (int i = 0; i < size; i++) {
    if (arr[i] > valorMaximo) {
      valorMaximo = arr[i];
    }
    if (arr[i] < valorMinimo) {
      valorMinimo = arr[i];
    }
  }

  float amplitud = (valorMaximo - valorMinimo) * (5.0 / 1023.0); // Convertir a voltios
  return amplitud;
}
```

- La función **medirAmplitud** está diseñada para medir la amplitud de una señal representada por un arreglo de enteros. La amplitud es la diferencia entre el valor máximo y el valor mínimo de la señal, convertida a voltios. Los parámetros que recibe

la función son: Un **puntero** a entero que representa un arreglo de enteros. Este arreglo contiene los valores de la señal que se analizará.

Un **entero** que indica el tamaño del arreglo, es decir, la cantidad de elementos en el arreglo, Una vez recibidos los parámetros la función se inicia la variable **valorMaximo** en 0, y se usará para almacenar el valor más alto encontrado en el arreglo, también se inicia la variable **valorMinimo** en 1023, el valor máximo posible de la señal (asumiendo que se trata de una señal con 10 bits de resolución, que varía entre 0 y 1023), luego, la función recorre el arreglo **arr** utilizando un bucle **for** que itera desde 0 hasta el tamaño del arreglo. Después de recorrer el arreglo, calcula la **amplitud** de la señal como la diferencia entre **valorMaximo** y **valorMinimo**. Esta diferencia se multiplica por el factor de conversión **(5.0 / 1023.0)** para convertir la amplitud a **voltios**, dado que el rango de entrada es de 0 a 1023 y corresponde a un rango de 0 a 5 voltios. Finalmente, devuelve el valor de la amplitud convertida a voltios.

NOTA: Dado que todavía estamos en el proceso de desarrollo estas funciones pueden estar sujetas a cambios, así como también se pueden implementar funciones nuevas.

Problemas en el desarrollo del Desafío.

Durante el desarrollo del desafío surgieron inconvenientes a la hora de calcular la frecuencia y la amplitud de la señal, ya que no teníamos el conocimiento necesario sobre señales y cómo poder calcular la información necesaria, sin embargo después de varias investigaciones logramos plantear una solución adecuada utilizando la fórmula de la frecuencia que es 1/periodo de la señal. Para lograr esta solución tuvimos que buscar la forma de cómo medir el tiempo desde el momento en que se oprime el pulsador para calcular los datos hasta el momento en que se deben mostrar los resultados en la pantalla.

Otro problema al que nos enfrentamos al momento del desarrollo es la dificultad de diferenciar entre una señal triangular y una señal senoidal, dado que las características de las dos señales son muy similares. Se ha intentado abarcar este problema desde el análisis de la pendiente entre puntos de las señales y patrones repetitivos en los valores máximos de la señal pero no se ha llegado a un resultado satisfactorio.