**Desafío I**

Presentado a: Aníbal Guerra

Augusto Salazar

Presentado por: Mariana Marín

Adriana Erazo

Universidad de Antioquia

Facultad de Ingeniería

Informática II

2024 – II

**Desafío I**

**Contenido:**

- Análisis del proyecto:

¿Cuál es el propósito del código?

¿Como funciona el código?

¿Cuál es la salida del código?

- Esquema del proyecto.

- Análisis del problema y consideraciones para la alternativa de solución propuesta.

- Problemas de desarrollo que afrontamos.

- Evolución de la solución y consideraciones para tener en cuenta en la implementación.

- Link Tinkercad: https://www.tinkercad.com/things/5aq4jRujp7x-smashing-kup-hango/editel?sharecode=RXEG0xnkFQwNJPq1dEVGDZRn7zWQXDrjhCM3JhOCINY

**Desafío I**

**Análisis**

* ¿Cuál es el propósito del proyecto?

El propósito del proyecto es desarrollar un código capaz de analizar una onda que se transmite por medio de un generador de señales para determinar si corresponde a una señal válida y, en caso afirmativo, identificar el tipo de señal específica. Esto implica procesar los datos de la onda, compararlos con los patrones característicos de señales conocidas, como ondas senoidales, cuadradas o de picos.

* ¿Como funciona el código del proyecto?

El código del proyecto funciona por medio de dos pulsadores, un pulsador de inicio y un pulsador de final, dos resistencias, una placa Arduino, un generador de señales y una pantalla LCD. Los pulsadores van a definir el momento de inicio a tomar datos y el momento en que finaliza la toma de datos. Los datos de la señal que transmite el generador se irán guardando en un arreglo de memoria dinámica, los cuales usaremos para analizar y validar la señal ingresada, tendremos un puntero que recorra el arreglo para tener claro las características específicas que tendrá la señal.

* ¿Cuál es la salida del código del proyecto?

El objetivo final es ofrecer una clasificación precisa de la señal que se detecta, proporcionando el tipo de señal y su información sobre frecuencia, periodo y amplitud, obteniendo así a un análisis más detallado de la misma.

**Desafío I**

**Esquema**

El esquema del codigo del proyecto tiene 3 componentes fundamentales:

1. Variables globales
2. Función void setup.
3. Función void loop.

Dentro de cada uno están:

1. **Variables globales:**

En este espacio vamos a definir algunas variables que usaremos para todo el código. Entre ellas serán:

* Inserción de librería para la pantalla LCD.
* Asignación de pines

En este espacio el código tendrá:

const int pinBotonIniciar = 2; //Se usa const para mantener la constante del pin del pulsador

const int pinBotonDetener = 4;

bool tomandoDatos = false;

unsigned long tiempoInicio = 0; //Se usa unsigned para tomar solamente valores positivos o mayores que cero

unsigned long tiempoFin = 0;

Adafruit\_LiquidCrystal lcd(0); //Librería para inicializar la pantalla LCD

1. **Función void setup:**

En esta función en Arduino se define:

* La inicialización de pines se configura si los pines se usaran como entradas o salidas.

En este espacio el código tendrá:

pinMode(pinBotonIniciar, INPUT); //Se usa INPUT para leer el tipo de entrada al arduino

pinMode(pinBotonDetener, INPUT);

Serial.begin(9600); //Es un comando que inicia la velocidad de comunicación en el Arduino a 9600 baudios por segundo

lcd.begin(16, 2); // Inicializa e indica que es una pantalla con 2 filas y 16 columnas

lcd.setBacklight(1); // Enciende la iluminación "de fondo"

lcd.clear(); // Limpia la pantalla

lcd.setCursor(0, 0); // Indica que la impresión inicia en la fila 0 y columna 0

1. **Función void loop:**

En esta función en Arduino se define:

* El código que vamos a ejecutar repetidamente, usando lo que ya se definió en la void setup, permitiendo que el programa responda a entradas, controle las salidas y ejecute.

En este espacio el código tendrá:

* Inicio de toma de datos
* Fin de la toma de datos
* Duración de la toma de datos
* Validación de la señal tomada
* Características de la señal tomada
* Muestreo en LCD de resultados

**Desafío I**

**Análisis del problema y consideraciones para la alternativa de solución propuesta.**

El desafío I de informática II el objetivo del código era tomar valores que provenían de un generador de señales conectado a una placa Arduino uno, en la plataforma de Tinkercad realizamos el montaje usando dos pulsadores, la placa Arduino, el generador de señales, las resistencias y una pantalla LCD para la salida del código.

En primer momento los integrantes del grupo pensamos en memoria dinámica para almacenar los datos y creamos el código **void loop()** porque se repite en esta función ya que con cada nueva iteración la señal iba a tomar valores distintos y la memoria dinámica me permite hacer eso, tuvimos la intención de poner el limite de en 255 como cantidad de valores que almacenaría Arduino, porque no sabíamos a ciencia cierta cuantos valores necesitaríamos para llegar a tener la frecuencia, periodo y amplitud que nos pedían. Después nos pusimos a pensar como sería para que independientemente de cuanto tiempo este tomando valores, no se vaya a detener el código, pero esto era imposible por temas de memoria ya que, en un principio solo teníamos considerado nada mas 255, pero ese almacenamiento se llenaría en cuestión de segundos, entonces, lo que pensamos para resolver el tema de la toma de datos por un largo rango de tiempo es un arreglo en memoria dinámica que se va redimensionando hasta una capacidad máxima de 100 valores, después de eso lo que le implementamos fue que una vez que se llenara el arreglo, con los 100 valores, el valor 100+1, iba a ocupar el primer lugar del arreglo, así sucesivamente con todos los valores que se fueran ingresando, entonces creamos un puntero que vaya realizando esta operación cíclica, entonces al final, lo que íbamos a terminar analizando para sacar los datos solicitados, era como el ultimo segmento de la onda, los últimos 100 valores que tomara, esos serán los valores, que irán en el arreglo y sobre ellos se hará el análisis. Con un poco de tiempo más, analizando bien y mejorando la lógica de la toma de datos de manera cíclica, pudimos aumenta el arreglo a una capacidad máxima de 400 datos en entero lo que equivale a 800 bytes del total del espacio de memoria del Arduino, ya que el Arduino cuenta con 2048 bytes, entonces básicamente estoy dejando 1248 bytes para la demás codificación.

Después de haber llegado a la conclusión de usar memoria dinámica, pensamos en como íbamos a liberar la memoria para que nuestro código funcione correctamente, debido a eso creamos la función **liberarMemoria()** que se encarga como ya anteriormente mencionado, de liberar la memoria que ya no necesitamos en el programa. Primero, la función revisa si hay algún espacio de memoria ocupado, y si lo hay, lo libera para que no se desperdicie por eso, ella se actualiza cada que ejecutamos el código. Después, se asegura de "borrar", esto se hace asignando **nullptr** a la variable valores. Esto evita que el programa intente usar ese espacio de memoria que ya se liberó, lo que podría causar errores, dejando el espacio completamente libre y evitando problemas si intentamos usarla más tarde. Esto es importante porque, en el Arduino, la memoria es muy limitada, y si no la liberamos, podríamos quedarnos sin espacio para seguir trabajando.

Después pensamos en cómo sacar los datos de la señal que nos piden y creamos esta función: **void calcularParametros(int \*valores, int tamano, float &frecuencia, float &periodo, float &amplitud)** que básicamente lo que hace es recorrer el arreglo dinámico que tenemos donde almacenamos los valores de la señal capturada para encontrar el valor más alto y el más bajo. Estos valores son importantes porque nos dicen qué tan grande es la variación en la señal, lo que llamamos **amplitud**. Además, la función está lista para detectar ciertos puntos importantes donde la señal cambia de dirección (como si estuviera subiendo y de repente comienza a bajar), para calcular cuánto tiempo pasa entre esos cambios, lo que nos permite medir el **periodo** y la **frecuencia** de la señal, que nos dice qué tan rápido está ocurriendo ese ciclo de subidas y bajadas.

Explicación de los parámetros de la función **void calcularParametros(int \*valores, int tamano, float &frecuencia, float &periodo, float &amplitud)**:

**int \*valores**: Es un puntero a un arreglo de enteros que contiene los valores de la señal capturada. Este puntero permite acceder a los datos almacenados en el arreglo para realizar cálculos sobre ellos. Tomamos el arreglo en enteros ya que los valores que genera la señal es en enteros y además un entero pesa menos que un float en la memoria.

**int tamano**: Es el número de elementos en el arreglo valores. Indica cuántos valores de la señal están disponibles para análisis. Este parámetro es importante para saber hasta dónde recorrer el arreglo y para evitar acceder a posiciones fuera del rango.

**float &frecuencia**: Es una referencia a una variable de tipo float donde se almacenará el resultado calculado de la frecuencia de la señal. Usa una referencia que permite modificar la variable original que fue definida en la función, lo que significa que el valor calculado se reflejará fuera de la función.

**float &periodo**: Similar a frecuencia, es una referencia a una variable de tipo float donde se almacenará el resultado calculado del período de la señal. De nuevo, al ser una referencia, el valor calculado se actualizará directamente en la variable original.

**float &amplitud**: También una referencia a una variable de tipo float, en la que se almacenará el resultado calculado de la amplitud de la señal. Esta referencia permite que el valor calculado se actualice en la variable original.

¿Qué Hace Cada Parámetro en la Función?

**valores** se usa para acceder a los datos de la señal y calcular la amplitud, frecuencia y período.

**tamano** ayuda a determinar cuántos datos del arreglo se deben considerar en los cálculos.

**frecuencia** y **periodo** se calculan usando la información de los cambios en la señal. La frecuencia se calcula a partir del período, que es el tiempo que tarda la señal en repetirse.

**amplitud** se calcula como la diferencia entre el valor máximo y el valor mínimo en los datos de la señal.

Usamos la función **String detectarTipoDeOnda(int \*valores, int tamano)** para identificar la onda, usamos esta lógica para cada una:

**Onda indefinida**: Si el número de picos es más del doble de los valles o viceversa, o si hay una diferencia entre el número de picos y valles, se considera **una onda** **indefinida**.

**Onda cuadrada**: Si el puntero que recorre el arreglo encuentra dos valores contiguos o más, de manera positiva, será el pico y de manera negativa, será el valle, si hay valores consecutivos y en el recorrido del arreglo cambia de un numero a otro y mantiene la contigüidad, entonces será una **onda cuadrada.**

**Desafío I**

**Problemas de desarrollo que afrontamos:**

El tema de la memoria en Arduino nos generó muchos problemas, no sabíamos bien como capturar los datos con la idea de que pueda estar corriendo por mucho tiempo sin que la memoria se llene, logramos solucionarlo con el redimensionamiento de datos cíclico, logramos que llegara hasta un máximo de 400 valores, para a partir de ahí tomar los valores.

Para periodo y frecuencia, la lógica que usamos fue el cambio de un valor positivo a negativo y viceversa, entonces en esos dos instantes se va a tomar el tiempo en milisegundos para calcular el periodo, una vez teniendo el periodo, queríamos sacar la frecuencia con la lógica de que la frecuencia =1/periodo, pero no tuvimos en cuenta algunas variaciones de la frecuencia, a baja frecuencia los cálculos si se procesan, pero a muy alta frecuencia el código no esta diseñado para dar una respuesta correcta.

Para la amplitud sentimos que era mas sencillo, porque era encontrar el valor mas alto y el valor mas bajo, restarlos, ya que como uno es positivo y el otro negativo, en realidad estaríamos sumando y después se divide en 2, así obtenemos el valor de la amplitud.

**Desafío I**

**Evolución de la solución y consideraciones para tener en cuenta en la implementación:**

Para los botones, debemos presionar el botón de inicio por unos segundos para que el botón dé la señal de la toma de datos y pasa lo mismo para detener la toma de datos, hay que sostener presionado el botón por unos segundos para que este se detenga.

El programa esta diseñado para que salgan datos antes, durante y después de la toma de datos, pero en muy pocas ocasiones se puede tomar por segunda vez los datos, no encontramos la razón de este suceso. Por eso es recomendable detener en el tinkercad directamente el código y volver a ejecutarlo sin usar los pulsadores.

El desface en el generador de señales debe estar siempre sobre cero, porque de lo contrario, el código no tomará el tiempo para el valor de periodo, ya que no habrá un numero base para saber el cambio de signo.

El voltaje es de 5 voltios, si este se le aumenta, entonces es probable que el código falle. Además, la frecuencia debe ser de máximo 10hz para que el código pueda detectar el periodo y con ello calcular la frecuencia.