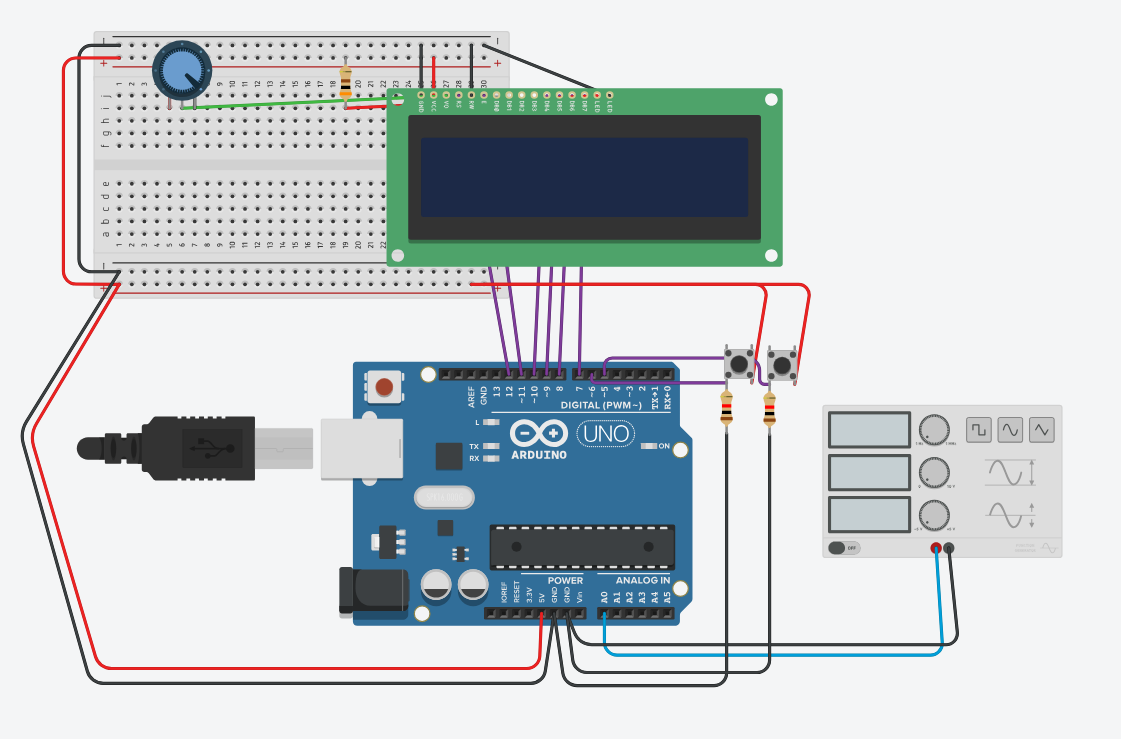
**DESAFIO 1.**

El generador de señales va a producir información constantemente, nuestro interés es almacenarla y con ella trabajar, la primera pregunta que surge es ¿cómo hacerlo? Como primera alternativa de solución y tras analizar el problema se plantea almacenar los valores que arroja el generador en una variable tipo entera, esta variable va a ser clave cuando el pulsador de la izquierda sea activado, pues esto va a indicar que se debe empezar a adquirir la información y guardarla, todo esto se va a realizar con un arreglo dinámico ya que la memoria destinada a este arreglo va a variar en base a la cantidad de datos enviados en el tiempo de pulsación del botón. El pulsador izquierdo va a determinar el momento en el que se van a adquirir los datos, el de la derecha va a ser el encargado de determinar el momento en el que se detiene la adquisición y se procesan los datos.

Para esta parte se planea usar un condicional que tenga como condición el estado de ambos pulsadores, es decir, si el de la izquierda se encuentra accionado y el de la derecha no, va a comenzar la adquisición de los datos. De lo contrario, con la información ya almacenada en el arreglo, si se activa el pulsador derecho se va a detener la adquisición de datos y se va a realizar el procesamiento de la información para mostrar los resultados en la pantalla LCD.

Para el desarrollo del desafío inicialmente se realiza un montaje prototipo del circuito en la plataforma tinkercad, tanto el código como el propio diseño del circuito pueden estar sujetos a cambios futuros, el montaje queda de la siguiente manera:



Los valores que arroja el generador de señales son la amplitud en instantes de tiempo, almacenando estos datos en un arreglo vamos a poder recorrerlo para calcular los demás parámetros. La idea de solución que se tiene para calcular la amplitud es un algoritmo que busque el valor máximo y mínimo que tome la amplitud en el arreglo, se restan el mayor por el menor y se divide por 2. Aprovechando que ya tenemos capturados los máximos y mínimos de las señales podemos usarlos como referencia para determinar los periodos, y en base a esta calcular la frecuencia, el problema aquí es que se desconocen los tiempos en los que el generador de funciones toma los valores, y no se encuentra información en la plataforma al respecto, cuando se tenga una estimación del tiempo entre valor y valor simplemente seria multiplicar este por la cantidad de datos que hay entre pico y pico, calculando así el periodo y por consecuente la frecuencia ya que simplemente seria dividir 1 entre el periodo encontrado.

Para encontrar el valor de los instantes de tiempo aproximados en los que el generador toma los datos se plantea realizar algunos experimentos con datos conocidos y fáciles de apreciar donde se pueda evidenciar la cantidad de datos tomados, es decir, se pone como configuración del generador una frecuencia de 1 Hz (por facilidad) y se cuentan cuantos valores hay entre un pico máximo y otro pico máximo, con esto podemos medir aproximadamente la diferencia de tiempo entre los datos, dividiendo 1 (un ciclo) por la cantidad de datos.

Una consideración importante es que, tras analizar las ideas, se intuye que la búsqueda de máximos y mínimos de la función se puede implementar directamente en el algoritmo de la creación del arreglo, logrando así mejorar la eficiencia ya que evita hacer un algoritmo aparte que recorra el arreglo buscando estos valores.

Surge otra alternativa de solución para el cálculo de la frecuencia a través de la función propia de Arduino “millis”, la cual mide el tiempo en milisegundos desde que comienza la simulación, con esto vamos a poder temporizar cuanto transcurre entre ciclo y ciclo para obtener el periodo. Aparece una dificultad y es que todos los parámetros del generador de señales van a poder cambiar y por ello si tomamos como punto de referencia los puntos máximos de la función para calcular la frecuencia va a presentar problemas cuando cambie la amplitud durante la recolección de datos, lo mismo ocurriría si se toma como punto de referencia el valor de la amplitud donde la función corta y se cambia el parámetro del desfase de CC. Si ambos parámetros varían durante la recolección de datos lo más probable es que se produjera una señal desconocida, donde va a ser complejo encontrar la frecuencia dado a su falta de periodicidad y como para calcular la frecuencia se necesita determinar cuánto tiempo transcurre en un periodo provisionalmente se va a determinar que el desfase de CC se va a mantener constante durante la adquisición de datos, este se va a poder cambiar antes de activar el pulsador, pero durante se va a mantener constante. Con este cambio también va a ser necesario calcular esto para saber en base a que valor de la amplitud se va a calcular el periodo.

Se descubre que la función serial.begin() dependiendo del número que tenga en su argumento va a aumentar o disminuir la cantidad de datos que van a ser transmitidos, por tanto nos interesa tener un numero alto que nos brinde siempre la mayor información posible para calcular la amplitud y la frecuencia con mayor exactitud, además que también va a ser útil para determinar el tipo de señal.

Para determinar el tipo de señal se plantea como solución comparar las diferencias entre las amplitudes del arreglo, si los cambios son muy bruscos va a ser una señal cuadrada, esto debido a que este tipo de señal solamente tiene 2 valores; si los cambios son constantes entre cada valor va a ser una señal triangular y si los cambios se determinan como “suaves” va a ser una señal senoidal, si la diferencia entre los valores de amplitud no entra en ninguna de las opciones anteriores va a ser una señal desconocida.

Identificación del tipo de señal a través del análisis de las diferencias de amplitud

1. Señal cuadrada:

- Se caracteriza por alternar abruptamente entre dos niveles de amplitud.

- Se detecta midiendo las grandes y constantes divergencias entre magnitudes consecutivas, pues solo presenta dos valores de amplitud claramente distintos.

2. Señal triangular:

- Adopta la forma de un triángulo al variar su magnitud de modo lineal.

- Se determina observando que las discrepancias entre amplitudes adyacentes permanecen iguales y siguen un patrón regular.

3. Señal senoidal:

- Modifica suave y continuamente su tamaño, emulando la forma de onda de un seno.

- Se distingue por las diferencias de amplitud que fluctúan de manera fluida y homogénea.

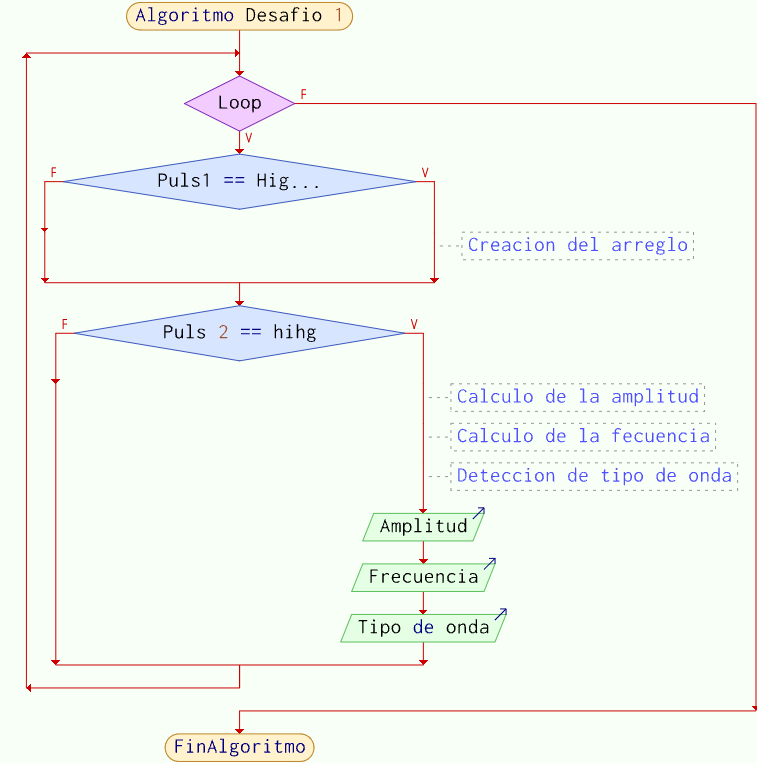
4. Señal desconocida:

- No se ajusta a los esquemas cuadrático, triangular o senoidal.

- Se identifica al no concordar las variaciones de tamaño con los modelos previos, exhibiendo alteraciones irregulares.

Este método permite identificar el tipo de señal según las amplitudes entre puntos del arreglo.

Ejemplo del algoritmo a realizar:



IMPLEMENTACION.

Se implementa algoritmo de recopilación de información, mientras el pulsador que da la señal se encuentre activo el sistema va a entrar en el primer condicional donde se forma el arreglo a través del uso de la memoria dinámica, con una variable contador se van a contabilizar el número de datos recibidos y con esta misma variable se van a introducir en el arreglo, la variable tamaño se va a actualizar según la cantidad de datos entrantes, esta se va a actualizar de 1 en 1 para evitar que en el arreglo haya basura; la actualización del tamaño del arreglo se realiza con un arreglo auxiliar al cual se le asigna la memoria deseada y se le copian todos los datos del arreglo anterior, luego se usa delete en el apuntador al primer arreglo y finalmente a este mismo se le asigna el contenido del arreglo auxiliar.

Problema: Arduino posee una memoria dinámica limitada, por tanto, si el arreglo llega a cierta cantidad de elementos se agota, pausando por completo el funcionamiento del código.

Solución: Con varias pruebas se determina que el tamaño máximo del arreglo de flotantes es aproximadamente 135, se asigna una constante entera con un valor cercano a este límite, cuando el tamaño del arreglo alcance este numero se reinicia por completo su contenido y memoria asignada, logrando así dar continuidad al programa el tiempo que sea necesario.

Cuando se decida activar el otro pulsador se entrará en el condicional donde se procesará e imprimirá la información, dentro hay tres funciones que realizan lo siguiente:

* La función calcularAmplitud es un algoritmo que recorre todo el arreglo hecho previamente y busca el valor máximo y mínimo dentro de este, cuando estos valores se han encontrado se resta el máximo por el mínimo y se divide entre dos, encontrando así la amplitud en voltios.
* La frecuencia fue muy problemática y en ciertos valores no brinda el valor correctamente, el funcionamiento del algoritmo es localizar picos máximos dentro del arreglo, comparando el valor inmediato con el valor anterior y posterior, si el valor en cuestión es mayor que los otros dos se determina que es un pico máximo y se suma al contador de picos, cuando se encuentra el primero se empieza a aumentar de uno en uno una variable que contabiliza la cantidad de datos que hay entre pico y pico.

Durante la creación del arreglo a través de la función de Arduino millis se cronometra cada cuanto tiempo se ingresan los datos al arreglo, este va a ser nuestro tiempo de muestreo, con experimentos y logrando reducir el tiempo lo máximo posible se determina un tiempo de entre 25 y 28 ms entre dato y dato. Para calcular el periodo se multiplica la cantidad de datos que hay entre picos por el tiempo cronometrado (aproximadamente 26.5 ms), luego se divide 1000 ms (1s) entre el periodo, calculando así aproximadamente la frecuencia en Hertz.

Problema: No siempre en el arreglo hay la misma cantidad de datos entre pico y pico, dando mucha imprecisión al calculo de la frecuencia. Además, hay frecuencias que no van a poder ser calculadas, el tiempo de muestreo no va a almacenar la suficiente cantidad de datos entre pico y pico para calcular la frecuencia correctamente.

* El algoritmo para determinar el tipo de señal es una función que invoca a otras funciones para comprobar a cuál categoría pertenece, estas funciones retornan valores booleanos, true si es de la categoría, false de lo contrario. La función cuadrada es reconocida si en el arreglo los valores varían entre un numero y su negativo únicamente; su correcto funcionamiento esta limitado por la constancia del parámetro desfase de CC.

La función triangular se caracteriza por aumentar o disminuir a valores constantes, el algoritmo funciona restando los valores del arreglo y sacando el promedio aritmético, si este valor es muy cercano a cero se caracteriza como triangular ya que la resta de valores cercanos va a tender a cero.

El reconocimiento de la función senoidal no se logra implementar, estas funciones aumentan o disminuyen describiendo cambios suaves, con el numero de datos que se almacenan en el arreglo y el tiempo de muestreo no se logra detectar estos cambios, es decir, entre dato y dato hay mucha diferencia, tomando el comportamiento de una onda triangular.

Si no se cumple ninguno de los 2 criterios que se implementaron se define que la onda es desconocida.

Si el pulsador de la derecha deja de ser accionado se empieza a recopilar información nuevamente. Con esto concluye el código, tiene bastantes deficiencias pero es lo que se logró en el tiempo propuesto.