DESAFIO 1

INFORME Y DOCUMENTACIÓN DEL PROYECTO

LUIS CARLOS ROMERO CARDENAS

IVAN ANDRES PERALTA CALLE

AUGUSTO ENRIQUE SALAZAR JIMENEZ

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERIA

2024

TABLA DE CONTENIDO

a. Análisis

La empresa informa2 solicita la implementación de un sistema que permita la identificación y clasificación de señales analógicas generadas a través de generador de función que permite generar tres tipos de señales principales, senoidal triangular y cuadrada. Además, debe incluir posibilidad de calcular la frecuencia en Hertz, la amplitud de la señal de entrada. Lo anterior mediante la herramienta Tinkercad en el cual se crea una simulación de un circuito electrónico el cual está compuesto por los siguientes elementos:

1. Arduino UNO: es una placa de prototipado de circuitos electrónicos, la cual hace el procesamiento de las entradas de lectura y salida de los componentes.

2. Generador de funciones: es el dispositivo que genera la entrada de la señal analógica hacia el Arduino que posteriormente será analizada de acuerdo con los requerimientos.

3. Pulsadores: componente electrónico que actúa como interruptor cuya función es permitir o interrumpir el flujo de corriente en un instante de tiempo. Se hace uso de dos pulsadores los cuales le van a permitir al usuario interactuar con el sistema de la siguiente forma:

el pulsador 1 tiene la función de indicar al sistema el inicio de la captura de la señal

el pulsador 2 tiene la función de indicar al sistema el fin de la captura de la señal

nota: los pulsadores incluyen una resistencia de 10 kΩ que tienen la función de controlar el flujo de la corriente.

4. Placa de prueba: elemento que permite y facilita la conexión entre los diferentes componentes del circuito.

5. Pantalla LCD 16x2: elemento a través del cual se mostrará al usuario los datos de salida requeridos.

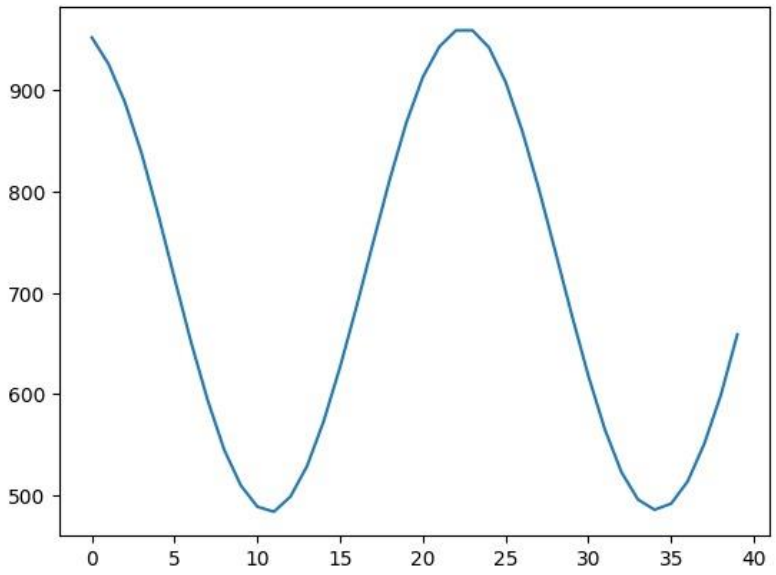
Por otro lado, en el componente del software se requiere el diseño e implementación de un algoritmo que permita identificar los tipos de señales, la frecuencia y la amplitud de señal y que permita la visualización de los resultados (tipo de señal, frecuencia en Hertz y amplitud en voltios).

Planteamiento de la solución:

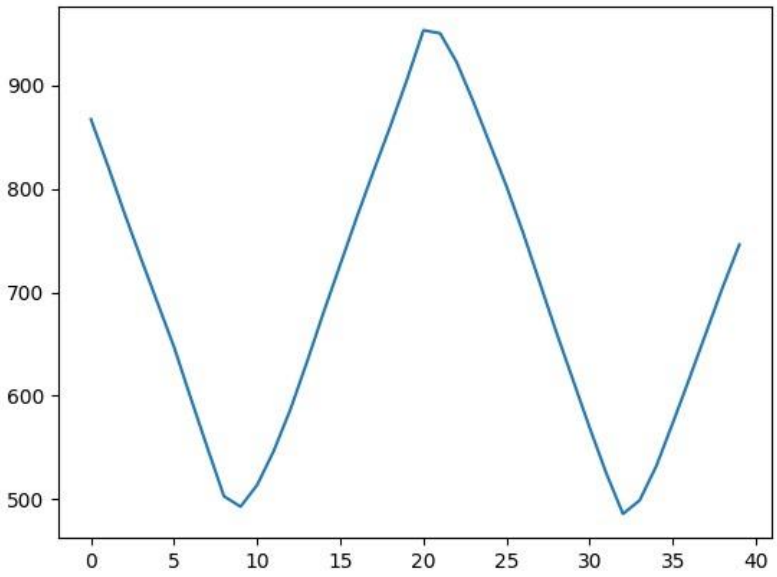
Para clasificar el tipo de señal analógica en los tres tipos mencionados en los requerimientos del cliente: senoidal, triangular, cuadrada y desconocida, se plantea la posible solución a partir de usar un enfoque basado en el análisis de los patrones de la señal.

A grandes rasgos, para distinguir estos tipos de señales:

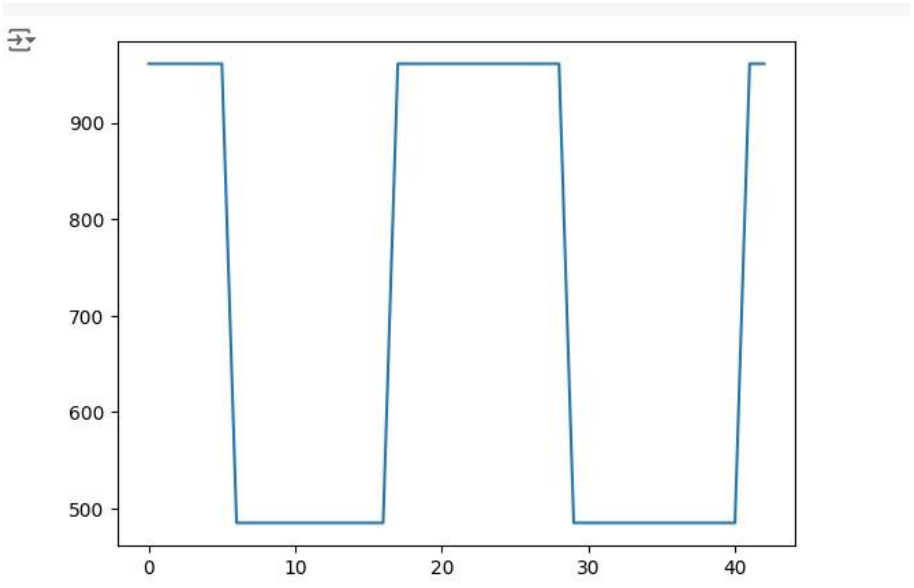
- **Señal senoidal**: En una señal senoidal, los ascensos y descensos no son perfectamente lineales, y las diferencias consecutivas de los valores de la señal (derivadas aproximadas) suelen ser menores y más suaves. Podemos calcular la variación media y usar un umbral bajo para identificar la senoidal.



- **Señal triangular**: En la señal triangular, los ascensos y descensos son lineales, por lo que las diferencias consecutivas entre puntos son casi constantes. Si detectamos cambios abruptos que no correspondan a un salto cuadrado, la señal será triangular.



- **Señal cuadrada**: Tiene transiciones abruptas entre valores altos y bajos (una forma de onda con saltos rectangulares).



Algoritmo en C++:

1. Representación de la señal: La señal analógica se almacenará en un arreglo dinámico para usar la memoria dinámica (requerimiento).

2. Clasificación de la señal: El algoritmo va a analizar la pendiente y la forma de la señal para clasificarla.

Circuito en Tinkercad:

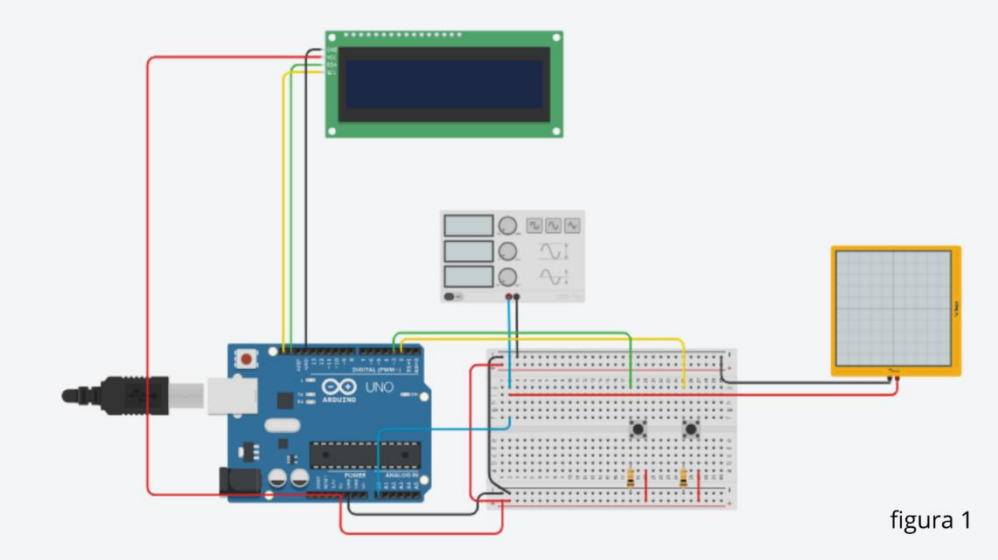
1. Componente hardware:

- **Arduino LCD de dos cables.**
- **Generador de función.**
- **Osciloscopio(opcional):** para visualizar el comportamiento grafico de las señales.
- **Pulsador.**
- **Resistencia.**

2. Componente software:

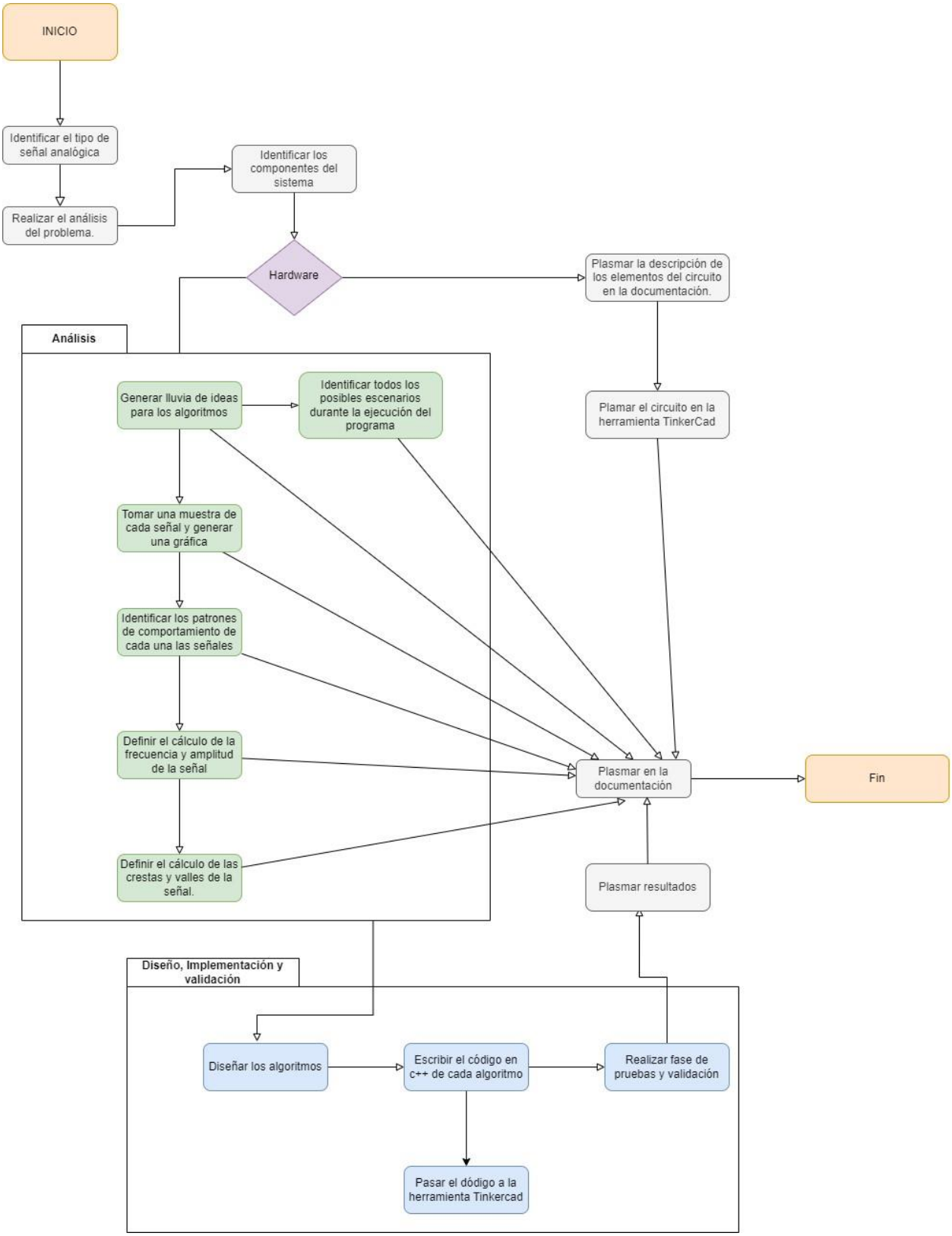
- **Lectura de datos:** se debe implementar un componente de software que permita la lectura del puerto análogo del Arduino dónde va a estar conectado el generador de funciones.
- **Indicador de inicio fin de captura de los datos de la señal:** es necesario implementar un componente de software que permita leer la señal digital de los pulsadores en los pines digitales 2 y 3, los cuales van a indicar el inicio y fin de la captura de los datos de la señal para su posterior análisis.

- **Algoritmo para identificar señales:** a partir del análisis inicialmente se plantea una posible solución para crear un algoritmo desde los patrones de comportamiento de la señal, tomando como referencia características principales como las crestas, valles, diferencia entre un valor y el siguiente. Y de está forma clasificar correctamente la señal a la que pertenecen los datos que se han capturado.
- **Algoritmo para calcular la frecuencia y amplitud:** finalmente es requerido calcular la frecuencia y amplitud. Para la amplitud se debe tener en cuenta el máximo y el mínimo de la señal.



b. Diagrama de flujo de tareas

A partir de las diversas reuniones del grupo para analizar los requerimientos de la solución que la empresa necesita, se ha elaborado un diagrama de flujo que resume las tareas clave a nivel macro. Estas tareas representan los pasos fundamentales que se deben seguir para cumplir con las expectativas del cliente y servirán como base para desarrollar la solución en cada una de sus etapas.


INICIO
Identificar el tipo de señal analógica
Realizar el análisis del problema.
Identificar los componentes del sistema
Hardware
Plasmar la descripción de los elementos del circuito en la documentación.
Análisis
Generar lluvia de ideas para los algoritmos
Identificar todos los posibles escenarios durante la ejecución del programa
Plamar el circuito en la herramienta TinkerCad
Tomar una muestra de cada señal y generar una gráfica
Identificar los patrones de comportamiento de cada una las señales
Definir el cálculo de la frecuencia y amplitud de la señal
Definir el cálculo de las crestas y valles de la señal.
Plasmar en la documentación
Fin
Plasmar resultados
Diseño, Implementación y validación
Diseñar los algoritmos
Escribir el código en c++ de cada algoritmo
Realizar fase de pruebas y validación
Pasar el dódigo a la herramienta Tinkercad

c. Algoritmos implementados.

El desarrollo de la solución planteada se hace con la implementación de los siguientes algoritmos:

Identificación del tipo de señal que está siendo generada y se está capturando por el puerto analógico 0 del Arduino: el algoritmo se toma como base el cálculo de tres variables inicialmente, las cuales son: ascensos, descensos y saltos. Estás tres variables se comportan como contadores y se calculan a partir de la diferencia entre el dato de la señal actual (posición i) y restarle el dato anterior de la señal (posición i-1). El resultado de esta diferencia se clasifica en ascensos si es un valor positivo, descensos si es un valor negativo y saltos si el valor de la diferencia es superior al umbral de 1.0. Este último de saltos funciona para la identificación de la señal cuadrada.

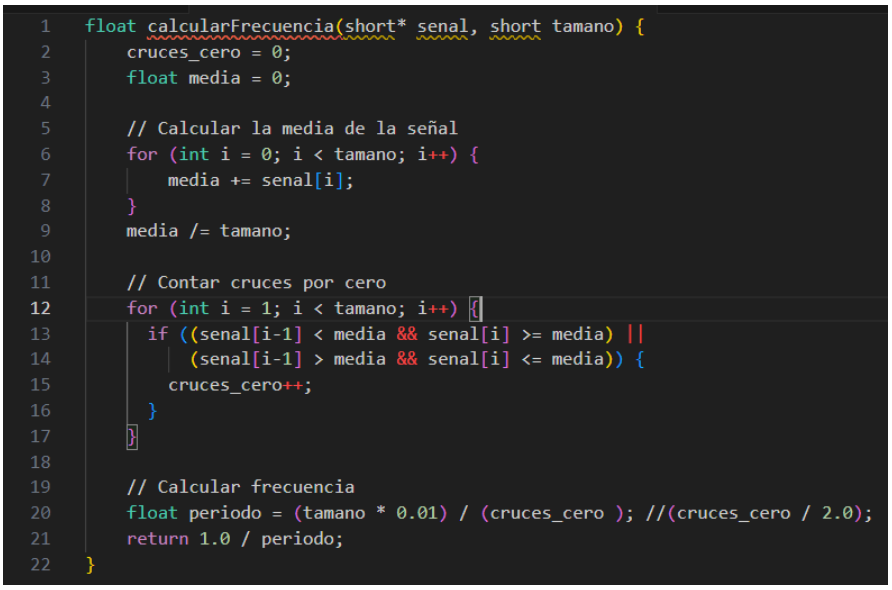
```
void clasificarSenal(short * datos, short tamano) {
    int ascensos = 0, descensos = 0, saltos = 0
    float sumaDiferencias = 0.0

    for (int i=1;i < tamano;++i) {
        float dif = datos[i] - datos[i - 1]
        sumaDiferencias += abs(dif)
        if (dif > 0) {
            ascensos++
        } else if (dif < 0) {
            descensos++
        }

        if (abs(dif) > 1.0) {
            saltos++
        }
    }


    Serial.println("ASCENSOS:")
    Serial.println(ascensos)
    Serial.println("DESCENSOS: ")
    Serial.println(descensos)
    Serial.println("SALTOS: ")
    Serial.println(saltos)
    Serial.println("TAMANO: ")
    Serial.println(tamano)
    Serial.println("PROMEDIO: ")
    Serial.println(sumaDiferencias/(tamano-1))
    Serial.println("MAX: ")
    Serial.println(max_val)
    Serial.println("MIN: ")
    Serial.println(min_val)
    Serial.println("CRUCES 0: ")
    Serial.println(cruces_cero)
```

```
36
37      if (ascensos == descensos | |abs(ascensos-descensos) == 1) {
38          lcd_1.setCursor(0, 0)
39          lcd_1.print("Senal cuadrada")
40      } else if (ascensos == descensos) {
41          lcd_1.setCursor(0, 0)
42          lcd_1.print("Senal senoidal")
43      } else if (ascensos != descensos) {
44          lcd_1.setCursor(0, 0)
45          lcd_1.print("Senal triangular")
46      }else {
47          lcd_1.setCursor(0, 0)
48          lcd_1.print("Senal no identificada")
49      }
50  }
51
```

Cálculo de la amplitud en voltios: el algoritmo toma el valor máximo y mínimo de la señal que posteriormente se divide entre 1023 para normalizar la lectura del voltaje del ADC y se multiplica por el valor del voltaje de referencia del Arduino (5V). Finalmente se aplica la fórmula de cálculo de la amplitud en voltios que es una resta entre el máximo y el mínimo valor de voltaje de la señal.

```
1
2   float medirAmplitud(short* datos, short tamano) {
3
4       max_val = datos[0];
5       min_val = datos[0];
6
7       for (int i = 1; i<tamano ; i++) {
8
9           if (datos[i] > max_val) {
10              max_val = datos[i];
11          }
12          if (datos[i] < min_val) {
13              min_val = datos[i];
14
15          }
16      }
17
18    float max_val_ = (max_val / 1023.0) * 5;
19    float min_val_ = (min_val / 1023.0) * 5;
20
21      return (float)(max_val_ - min_val_);// / 2; // Amplitud en Voltios
22  }
```

Cálculo de la frecuencia: la frecuencia se calcula en primera instancia calculando el promedio de la señal, el cual va a ser insumo para determinar si un punto de la señal es un corte con el eje x en 0. Una vez se han calculado todos los cruces con cero se calcula se determina el periodo de la función. Finalmente, se aplica la fórmula de la frecuencia que es 1/T.



```
float calcularFrecuencia(short* senal, short tamano) {
    cruces_cero = 0;
    float media = 0;

    // Calcular la media de la señal
    for (int i = 0; i < tamano; i++) {
        media += senal[i];
    }
    media /= tamano;

    // Contar cruces por cero
    for (int i = 1; i < tamano; i++) {
      if ((senal[i-1] < media && senal[i] >= media) ||
          (senal[i-1] > media && senal[i] <= media)) {
        cruces_cero++;
      }
    }

    // Calcular frecuencia
    float periodo = (tamano * 0.01) / (cruces_cero ); //(cruces_cero / 2.0);
    return 1.0 / periodo;
}
```

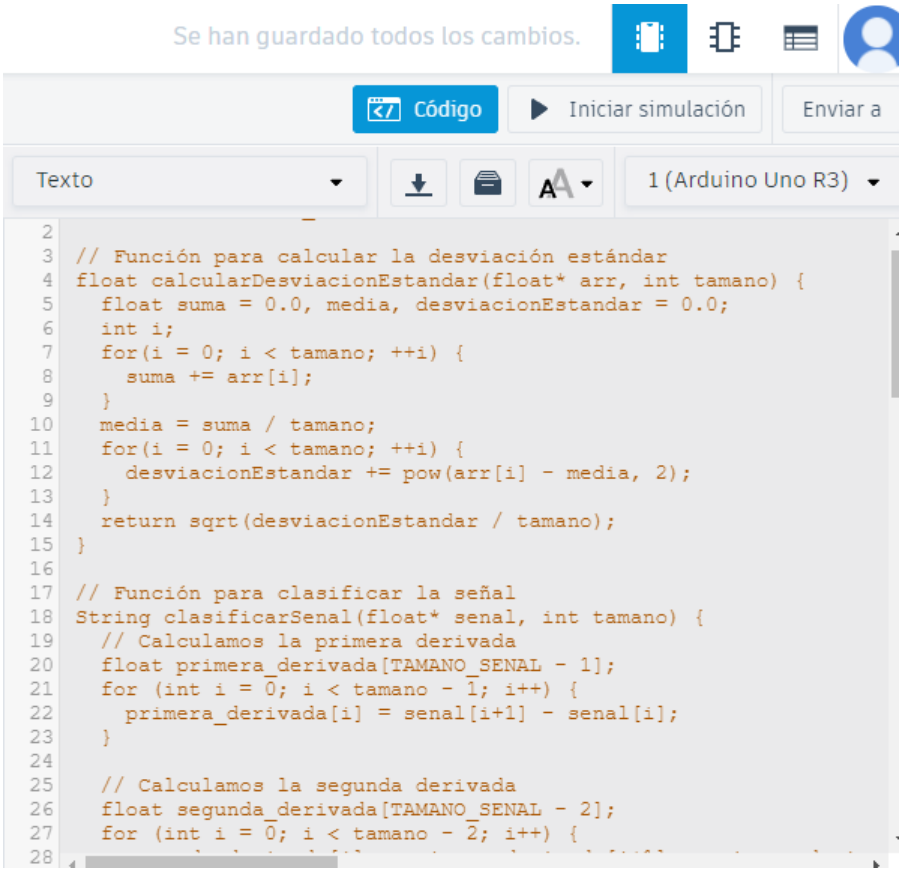
d. Problemas de desarrollo que afrontamos

En el trascurrir del desarrollo de la solución se presentaron los siguientes desafíos:

Configuración de los pulsadores: con el valor de la resistencia del circuito anti-rebote, el cual se solucionó después de investigar llegando a la conclusión que el valor de la resistencia es de 10 kΩ.

Limitación de memoria del Arduino: inicialmente se usó variables de tipo int, lo cual ocupaba 4 bytes por posición del arreglo de memoria dinámica, lo cual limitaba la cantidad de valores en el arreglo a medida que se avanzaba en la implementación del código, puesto que las demás variables requerían memoria algunas de tipo float, necesarias por los decimales. Se decide guardar los datos en un array de memoria dinámica de tipo short el cual ocupa 2 bytes por posición. Lo cual permitió aumentar a 480 datos máximo-posibles para el análisis de la señal. Usar más datos causó lentitud en el procesamiento e incluso lectura errónea de los datos al guardarlos con caracteres especiales.

Algoritmo de identificación de la señal: a partir de los patrones de la señal se intenta implementar un algoritmo para clasificar cada tipo de señal. La cuadrada tiene un patrón más fácil de identificar debido a sus variaciones abruptas al pasar de la cresta al valle de la señal. Sin embargo, las señales senoidales y triangulares tienen un patrón de comportamiento similar las variaciones en este caso no son tan abruptas. Para darle solución se aborda desde perspectivas como el cálculo de la varianza, la desviación estándar, la cantidad de máximos, mínimos, cruces con cero, promedio general, estandarización de la señal con el método de máximo y mínimo. Pero el algoritmo sigue fallando en la clasificación de senoidal y triangular.



Cálculo de la frecuencia: para este cálculo se requiere el tiempo de muestreo, el cual después de algunas pruebas usando la función micros() de Arduino la cual determina el valor en 24 que al convertirlo a milisegundos da alrededor de 0.02. Pero este valor en cálculo se aleja del valor real, al reducirlo a un aproximado de 0.01 milisegundos la precisión del cálculo mejora.


```
  Serial.println("La señal triangular se clasifica como: " + resu
  Serial.println("La señal senoidal se clasifica como: " + result

  delay(5000); // Espera 5 segundos antes de la siguiente clasifi
}*/
// C++ code
//
const int pinDigSalidaLed = 3;
const int pinAnalogEntradaPotenc = A0;
unsigned long tiempoInicio;
unsigned long tiempoFinal;
unsigned long intervalo;

void setup()
{
  pinMode(pinDigSalidaLed, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
    tiempoInicio = micros();


  int potenciValor = analogRead(pinAnalogEntradaPotenc);

    tiempoFinal = micros();

  // Calcula el intervalo de tiempo
  intervalo = tiempoFinal - tiempoInicio;
  Serial.println(intervalo);


  //int valorSalidaLed = map(potenciValor, 0, 1023, 0, 255);

  //analogWrite(pinDigSalidaLed, valorSalidaLed);
  //delay(100); // Wait for 1000 millisecond(s)

}
```

$$\frac{24\,\mu s}{1000} = 0.024\,ms$$

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

Consideramos hacer la declaración de variables tipo short en la mayor medida posible esto para evitar conflictos en el funcionamiento de la memoria de Arduino la cual es muy limitada.

Consideramos tener un mayor rango de captura de datos, pero al simular el sistema la memoria colapsaba y dejaba de leer datos continuamente por lo cual establecimos un límite de captura de 470 datos en la memoria dinámica.

Calcular frecuencia: para este cálculo inicialmente se planteó una solución para el periodo con el cual se podía obtener luego el valor de la frecuencia este consistía en contar el número de veces que un valor pasaba de negativo a positivo que es lo mismo que decir que se cumplió un ciclo de la señal, esto tenia fallas cuando la señal continua esta por encima de cero en solución a lo anterior decidimos plantear una solución que consiste en tomar la media del total de valores capturados y hacer una comparación entre los valores consecutivos y la media si el valor anterior al actual comparado es menor a la media y el valor actual es igual o mayor a la media significa que abría un cruce por cero esto como si fuera ascendente aplicamos la misma solución para contar los cruces por cero cuando es un valor descendente. También para el tiempo de muestreo de la señal que requeríamos para hallar el periodo inicialmente nos planteamos usar la función millis() de Arduino que hace conteo de los milisegundos pero al darnos cuenta de que es muy imprecisa descubrimos que para mayor precisión existe la función micros().

Clasificación de señales: inicialmente no teníamos claro como plantear una diferencia de valores para diferenciar entre la señal triangular y senoidal por lo cual nuestra solución inicial no sabía distinguir entre estos dos casos de señal y siempre arrojaba que la señal era triangular, aunque la señal generada fuera senoidal por lo cual se usó el método de estandarización de datos por mínimo y máximo que para el caso se refiere a cresta y valles de la señal y que permite mantener los valores entre 0 y 1 de esta manera calcular el valor de la diferencia entre la cresta de la señal y el punto anterior.

f. Anexos

Video: https://youtu.be/kE08MeOqPkI

Tinkercad: https://www.tinkercad.com/things/eCic5PBUzgN-desafioi/editel?returnTo=%2Fdashboard%2Fdesigns%2Fcircuits&sharecode=zlr9sa2OqVm29qj677da-itjvy8JpVt-Ow3plD6obnY