Informe Desafio 1 – Juan Angel Omaña

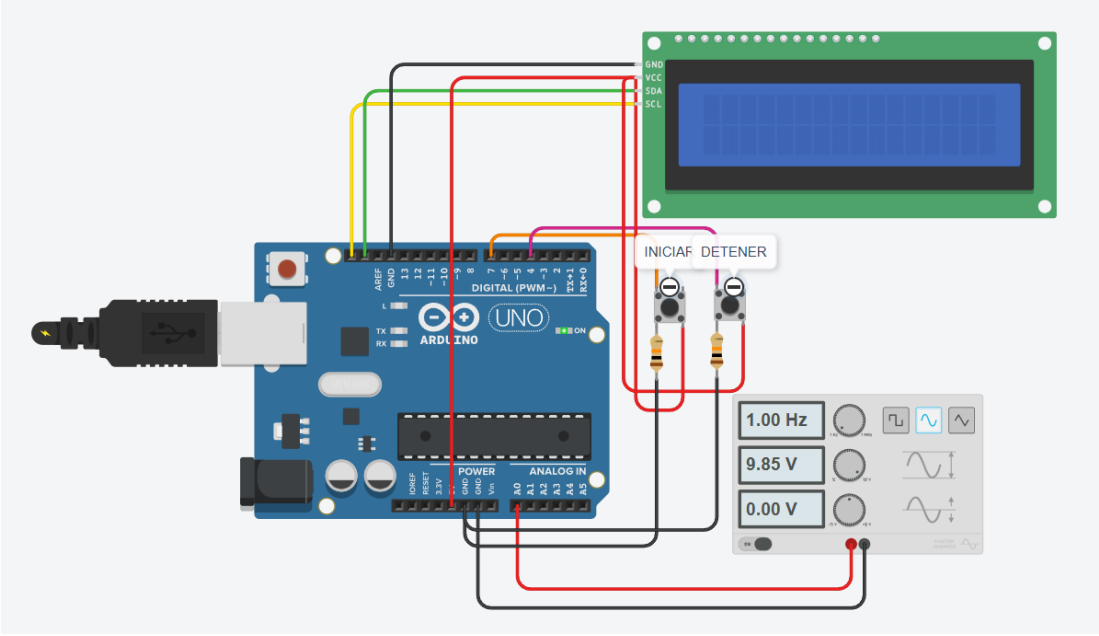
a. Análisis del Problema y Alternativa de Solución Propuesta

Problema: El objetivo del proyecto es diseñar un sistema de adquisición y visualización de señales utilizando Arduino, con capacidad para medir frecuencia en Hertz, amplitud en Voltios, y detectar la forma de onda de la señal capturada.

Solución Propuesta: Se ha diseñado un sistema utilizando un Arduino, un generador de señales y un LCD para la visualización. El sistema incluye un algoritmo para medir la frecuencia y la amplitud de la señal, y otro para identificar la forma de onda.

b. Esquema de Tareas

1. **Configuración del Hardware:**
    - Conectar el Arduino al generador de señales y al LCD.
    - Configurar los pines para los botones de inicio y detención.



2. **Implementación del Algoritmo de Adquisición de Datos:**
    - Leer los valores de la señal utilizando el ADC del Arduino.
    - Almacenar los valores en un arreglo dinámico.
    - Implementar la redimensión del arreglo cuando sea necesario.
3. **Implementación del Algoritmo de Medición:**
    - Calcular la frecuencia de la señal medida.

- Determinar la amplitud de la señal.

4. **Implementación del Algoritmo de Identificación de la Forma de Onda:**
    - Analizar los datos para identificar la forma de onda (senoidal, cuadrada, triangular, desconocida).
5. **Visualización y Salida de Datos:**
    - Mostrar los datos de la señal y los resultados de la medición en el LCD.
    - Enviar los datos y resultados al monitor serial.
6. **Pruebas y Ajustes:**
    - Realizar pruebas con diferentes formas de onda y ajustar el algoritmo según sea necesario.

c. Algoritmos Implementados

7. **Algoritmo de Adquisición de Datos:**

```
void loop() {
  if (digitalRead(botonInicio) == HIGH) {
    acquiringData = true;
    lcd.clear();
    lcd.print("Adq. datos");
  }

  if (digitalRead(botonDetener) == HIGH) {
    acquiringData = false;
    lcd.clear();
    lcd.print("Adq. detenida");
    printArray();
    calcularAmplitudYFrecuencia();
  }

  if (acquiringData) {
    if (cantElementos == capacidad) {
      redArr(arr, capacidad);
    }
    arr[cantElementos] = analogRead(analogPin);
    Serial.println(arr[cantElementos]);
    cantElementos++;
    delay(50);
  }
}
```

8. **Algoritmo de Redimensión de Arreglo:**

```
void redArr(int16_t*& arr, int& capacidad) {
  int nuevaCap = capacidad * 2;
  int16_t* nuevoArr = new int16_t[nuevaCap];
  for (int i = 0; i < capacidad; i++) {
    nuevoArr[i] = arr[i];
  }
  delete[] arr;
  arr = nuevoArr;
  capacidad = nuevaCap;
}
```

9. **Algoritmo de Cálculo de Amplitud y Frecuencia:**

```
void calcularAmplitudYFrecuencia() {
  amplitud = (maxValue - minValue) * (5.0 / 1023.0);
  Serial.print("Amplitud: ");
  Serial.print(amplitud);
  Serial.println(" V");

  if (frecuencia > 0) {
    Serial.print("Frecuencia: ");
    Serial.print(frecuencia);
    Serial.println(" Hz");
  } else {
    Serial.println("No se detectó una frecuencia válida");
  }

  maxValue = 0;
  minValue = 1023;
  peakTime1 = 0;
  peakTime2 = 0;
  frecuencia = 0;
}
```

10. **Algoritmo de Identificación de Forma de Onda:** (Este algoritmo se desarrollará una vez que el análisis de datos esté completo).

d. Problemas de Desarrollo

- **Limitaciones de Memoria:** Inicialmente, el uso de arreglos de tamaño fijo y el tipo de dato `int` provocaban problemas debido a la limitada memoria disponible en el Arduino. Se resolvió cambiando el tipo de dato a `int16_t` y utilizando memoria dinámica para ajustar el tamaño del arreglo según sea necesario.
- **Precisión en la Medición:** La precisión en la medición de frecuencia y amplitud fue un desafío, especialmente con señales de alta frecuencia. Se mejoró ajustando los intervalos de muestreo y realizando cálculos más precisos.

e. Evolución de la Solución y Consideraciones

- **Evolución:** La solución pasó de usar arreglos de tamaño fijo a arreglos dinámicos con el tipo de dato `int16_t`, lo que permitió manejar grandes cantidades de datos sin exceder la memoria disponible. Se implementaron mejoras en el cálculo de la frecuencia y la amplitud para obtener resultados más precisos.
- **Consideraciones:** Se debe considerar el límite de memoria en el entorno de Arduino al diseñar sistemas de adquisición de datos. La optimización del código y el uso eficiente de los recursos son cruciales para garantizar un rendimiento adecuado.