UNIVERSIDAD DE ANTIOQUIA
FACULTA DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE TELECOMUNICACIONES
SEMESTRE II

Desafío 1
Señales Analógicas
Informe detallado del proyecto

Asignatura:
Informática II Teoría

Autor:
Oscar Miguel López Peñata

Tutores:
Anibal Guerra
Augusto Salazar

Medellín, Septiembre de 2024

INTRODUCCIÓN

El propósito de este informe es detallar el desarrollo inicial de un sistema de adquisición y procesamiento de señales implementado utilizando la plataforma Arduino UNO y simulado en Tinkercad. El objetivo de este proyecto es capturar y analizar señales analógicas generadas artificialmente para calcular parámetros básicos como frecuencia (hercios), amplitud (voltios) y forma de onda (seno, triángulo, cuadrado, etc.). El análisis se visualizará mediante un monitor en serie que replica el entorno de seguimiento y control.

Para realizar este proceso se diseñó un circuito en Tinkercad que incluía un Arduino UNO, un generador de señal que proporcionaba entrada analógica y dos pulsadores (botones) que permitían su uso. El sistema de adquisición también complementa la implementación del algoritmo C++, que es capaz de controlar el flujo de datos en tiempo real y calcular eficientemente los parámetros de la señal.

Los desafíos del desarrollo es la implementación de un posible sistema de rebote de software en los botones, con el objetivo de asegurar la estabilidad de las lecturas y evitar fluctuaciones innecesarias debido a las propiedades físicas de los botones o de la sincronización del tiempo. Además, se utiliza memoria dinámica para almacenar muestras de señales, optimizando así el uso de los recursos de Arduino y facilitando el procesamiento de datos.

Este informe provisional describe el estado actual del proyecto, incluida la creación de circuitos analógicos y el desarrollo de capacidades clave de adquisición, procesamiento y visualización. Aún se necesitan más pruebas para verificar el correcto funcionamiento del sistema en diferentes configuraciones de señal y optimizar la precisión de la identificación de la forma de onda. Estos pasos son esenciales para la finalización del proyecto y al finalizar producirán un informe final y una demostración del rendimiento del sistema.

DESCRIPCIÓN DEL PROBLEMA

El problema plantea el desarrollo de un sistema de adquisición y procesamiento de señales analógicas utilizando la plataforma Arduino UNO, simulada en el entorno virtual de Tinkercad. La señal analógica de entrada, generada artificialmente, puede variar en su forma de onda (senoidal, cuadrada, triangular, entre otras), frecuencia (en Hertz) y amplitud pico a pico (en Voltios). El desafío principal consiste en diseñar un sistema que sea capaz de capturar, almacenar y procesar estas señales con el propósito de calcular los siguientes parámetros clave:

- Frecuencia: Número de ciclos completos de la señal por segundo, que define la periodicidad de la onda.
- Amplitud: Diferencia de voltaje entre los picos máximos y mínimos de la señal, medida en voltios.
- Forma de onda: Identificación de la morfología de la señal, clasificándola como senoidal, triangular, cuadrada, u otras formas dependiendo de sus características.

La solución propuesta deberá implementarse en un entorno de simulación con Tinkercad, utilizando un generador de señales que envíe una señal analógica variable a la entrada del Arduino UNO (pin A0). Para cumplir con los objetivos del proyecto, se requiere desarrollar una solución algoritmica en C++ que capture las muestras de la señal, las almacene en un buffer dinámico en memoria y procese la información obtenida para calcular los parámetros de frecuencia y amplitud. Adicionalmente, el sistema deberá analizar los datos para identificar la forma de onda y finalmente presentar los resultados en el monitor serie del Arduino.

El diseño de este sistema representa un reto en la gestión de recursos del Arduino UNO, incluyendo la capacidad de almacenamiento y el manejo eficiente de la memoria dinámica, al mismo tiempo que asegura la precisión y fiabilidad en la medición de señales. Además, se busca garantizar un funcionamiento robusto y en tiempo real, replicando un entorno de monitoreo de señales en aplicaciones prácticas. Además representa un reto en cuando al diseño e implementación de soluciones creativas para un desarrollador de aplicaciones y/o Ingenerio de Telecomunicaciones.

DESCRIPCIÓN DE TAREAS

Fecha	Nombre Tarea	Descripción
Septiembre 12 de 2024	Revisión de requerimientos.	Revisar los detalles específicos del desafío para asegurarme de comprender lo que se necesita, como la medición de frecuencia, amplitud y forma de onda con el Arduino UNO y Tinkercad. Verificar si hay algún ajuste adicional en los requerimientos o nuevas solicitudes de parte de los clientes.

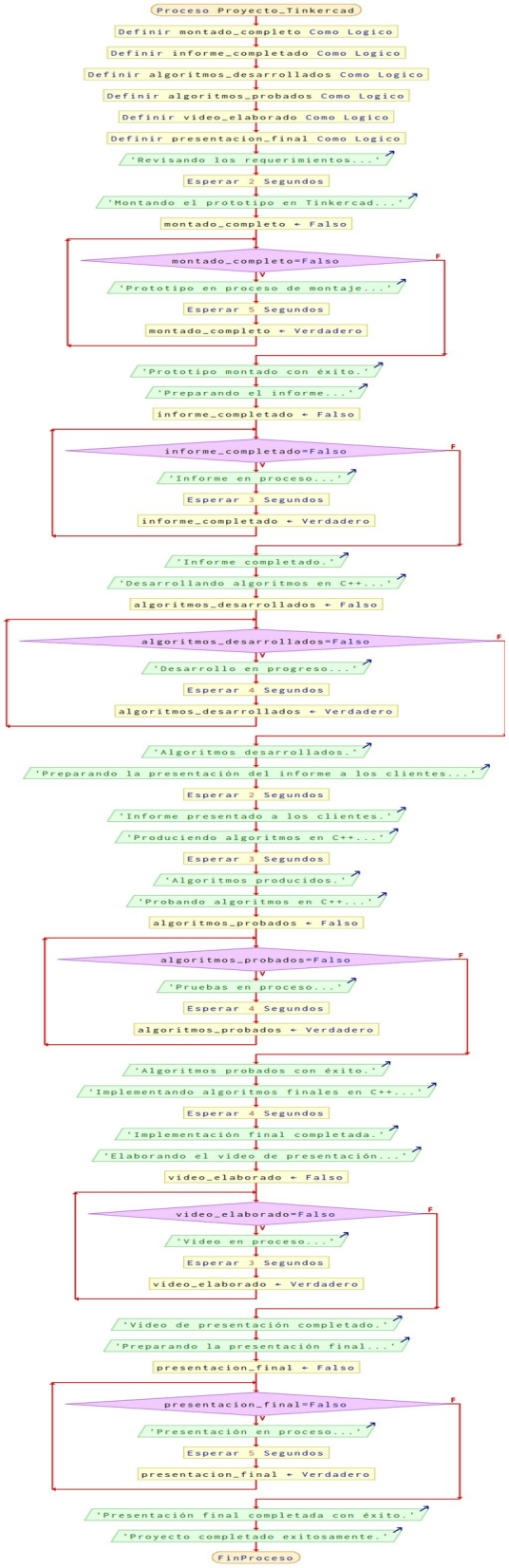
Fecha	Nombre Tarea	Descripción
Septiembre 13 de 2024	Montaje del prototipo en la plataforma Tinkercad (Tarea principal).	Crear el montaje en Tinkercad. Conectar el Arduino UNO. Añadir el generador de señales al pin A0 del Arduino. Conectar los dos pulsadores para iniciar la adquisición de las señales analogas. Asegurarme de que todo el hardware virtual esté correctamente conectado.

Fecha	Nombre Tarea	Descripción
Septiembre 13 de 2024	Preparación del informe.	Generar el documento y/o informe preliminar requerido por los clientes para verificar el direccionamiento de las soluciones algoritmicas propuestas para el diseño del prototipo en Tinkercad, así como observar el orden de las diversas soluciones propuestas para la implementación final.
Septiembre 13 de 2024	Elaboración de algoritmos de solución en C++ (Tarea alterna).	Recursos: QTCreator, VSCodium y Geany IDE. Usos: Creación, compilación y depuración de algoritmos. Lenguaje: C++
Septiembre 14 de 2024	Presentación del informe a los clientes.	Realizar carga del documento tipo informe del presente proyecto a respositorio principal del desarrollador y/o envío a sus respectivos correo para que los clientes tengan una idea principal de las diversas soluciones de la implementación final.

Fecha	Nombre Tarea	Descripción
Septiembre 14 de 2024	Producción de algoritmos de solución en C++ (Tarea alterna).	Recursos: QTCreator, VSCodium y Geany IDE. Usos: Creación, compilación y depuración de algoritmos. Lenguaje: C++ Actividad secuencial: Pruebas y ejecución de diversas soluciones en C++ para el prototitpo en Tinkercad.
Septiembre 15 de 2024	Prueba de algoritmos de solución en C++ (Tarea alterna).	Recursos: QTCreator, VSCodium y Geany IDE. Usos: Creación, compilación y depuración de algoritmos. Lenguaje: C++ Actividad secuencial: Prueba de las posibles soluciones presentadas en el diseño de Tinkercad.
Septiembre 16 de 2024	Implementación final de algoritmos en C++	Desarrollo de funciones que permiten capturar, procesar y analizar señales analógicas. Generar los cambios en el repositorio principal en Github acerca del proyecto.
Septiembre 16 de 2024	Elaboración video de presentación.	Realizar video de la presentación del prototipo elaborado en Tinkercad. Generar un link para compartir el video.

Fecha	Nombre Tarea	Descripción
Septiembre 17 de 2024	Presentación final de la implementación.	Compartir el link de de Github sobre el proyecto elaborado en lenguaje C++. Compartir el link del video de la presentación del prototipo en Tinkercad. Recursos: Github, OBS Studio, Ude@, YouTube, Zoom.

ESQUEMA GRÁFICO DESCRIPCIÓN DE TAREAS



Nota: *Elaboración propia*

ALGORITMOS IMPLEMENTADOS

```
#include <Arduino.h>
#include <Adafruit_LiquidCrystal.h>

// Declaración de constantes
const int pinPulsadorAdquisicion = 2;
const int pinPulsadorProcesar = 3;
const int pinSignalInput = A0;

// Declaración de variables globales
bool adquisicionActiva = false;
bool procesarSignal = false;
int frecuencia = 0;
float amplitud = 0.0;
char formaOnda[20];

// Declaración de funciones
void setup() {
  // Inicializar pines de entrada y salida
  pinMode(pinPulsadorAdquisicion, INPUT);
  pinMode(pinPulsadorProcesar, INPUT);
  pinMode(pinSignalInput, INPUT);

  // Inicializar pantalla LCD
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("Informa2 Desafio 1");
}

void loop() {
  // Leer pulsador de adquisición
```

```
  if (digitalRead(pinPulsadorAdquisicion) == HIGH) {
    adquisicionActiva = true;
  }

  // Leer pulsador de procesar
  if (digitalRead(pinPulsadorProcesar) == HIGH) {
    procesarSignal = true;
  }

  // Adquisición de datos
  if (adquisicionActiva) {
     int* datos = new int[100]; // Reserva de memoria dinámica para almacenar los datos de la señal
    for (int i = 0; i < 100; i++) {
      datos[i] = analogRead(pinSignalInput);
    }

    // Procesar datos
    if (procesarSignal) {
      frecuencia = calcularFrecuencia(datos, 100);
      amplitud = calcularAmplitud(datos, 100);
      strcpy(formaOnda, identificarFormaOnda(datos, 100));

      // Mostrar resultados en pantalla LCD
      lcd.setCursor(0, 1);
      lcd.print("Frecuencia: ");
      lcd.print(frecuencia);
      lcd.print(" Hz");
      lcd.setCursor(0, 2);
      lcd.print("Amplitud: ");
      lcd.print(amplitud);
      lcd.print(" V");
```

```
      lcd.setCursor(0, 3);
      lcd.print("Forma de onda: ");
      lcd.print(formaOnda);

      // Liberar memoria dinámica
      delete[] datos;
    }
  }
}

// Funciones para calcular frecuencia, amplitud y forma de onda
int calcularFrecuencia(int* datos, int tamano) {
  // Implementar algoritmo para calcular frecuencia
  // ...
  return frecuencia;
}

float calcularAmplitud(int* datos, int tamano) {
  // Implementar algoritmo para calcular amplitud
  // ...
  return amplitud;
}

char* identificarFormaOnda(int* datos, int tamano) {
  // Implementar algoritmo para identificar forma de onda
  // ...
  return formaOnda;
}
```

Nota: El desarrollo de la implementación inicialmente llevaba este enfoque pero con el pasar del análisis la tendencia de mejoramiento debe quedar fundamentada en cada proceso del diseño.

PROBLEMAS DE DESARROLLO

Gestión de memoria dinámica: Existe el riesgo de fugas de memoria o errores de segmentación si no se gestionan adecuadamente las asignaciones y liberaciones de memoria. Esto require un monitoreo cuidadoso del uso de memoria para evitar posibles fallos durante la ejecución.

Precisión en el cálculo de parámetros: El cálculo de parámetros como la frecuencia y la amplitud enfrenta dificultades debido a la naturaleza limitada del ADC (Convertidor Analógico-Digital) de Arduino. Es necesario realizar ajustes en los algoritmos para mejorar la precisión dentro de las limitaciones del hardware.

Rebotes en los pulsadores: El posible comportamiento errático de los pulsadores debido a los rebotes mecánicos presenta un desafío en la estabilidad del sistema. Sin una gestión adecuada, los pulsadores pueden múltiples activaciones no deseadas, lo que compromete la funcionalidad del sistema.

Simulación en Tinkercad: Si bien Tinkercad es una herramienta poderosa para simular circuitos, presenta limitaciones en términos de velocidad y precisión cuando se simulan señales en tiempo real. La simulación de señales con diferentes formas de onda, frecuencias y amplitudes algunas veces no refleja con precisión el comportamiento real del sistema. Esto obliga a realizar ajustes en los algoritmos para adaptarse mejor a las condiciones de simulación.

Sincronización de la adquisición y procesamiento: Uno de los principales desafíos es lograr una sincronización eficiente entre la adquisición de señales y su procesamiento en tiempo real. La adquisición continua de muestras puede sobrecargar el sistema si no se implementan mecanismos que regulen el flujo de datos y el tiempo dedicado a cada tarea (adquisición vs procesamiento).

EVOLUCIÓN DE LA SOLUCIÓN Y CONSIDERACIONES PARA TENER EN CUENTA EN LA IMPLEMENTACIÓN

Evolución de la solución

En la etapa inicial, la implementación se centró en la adquisición de la señal analógica, donde se utilizó un buffer dinámico para almacenar las muestras de la señal capturada a través del pin A0. Sin embargo, se identificaron problemas de sincronización entre la adquisición y el procesamiento de los datos, lo que llevó a ajustar los algoritmos para mejorar la eficiencia en tiempo real.

Para resolver estos inconvenientes, se optará por una estructura más simple que divide el proceso en fases: primero se realiza la captura de las muestras, y luego, una vez lleno el buffer, se procede al cálculo de los parámetros críticos como la frecuencia, amplitud y forma de onda. Esto permitirá (en lo posible) reducir la sobrecarga del sistema y mejorar la precisión en los cálculos.

En una etapa posterior, se añadirá un sistema de antirebote por software para gestionar correctamente los pulsadores que controlan la adquisición y visualización de los resultados. Esto puede ser esencial para asegurar que las entradas de usuario no se vean comprometidas por rebotes mecánicos en los pulsadores.

Finalmente, se podrán mejorar los algoritmos de procesamiento de señales, haciendo que el cálculo de frecuencia y amplitud fuera más robusto, incluso dentro de las limitaciones del Arduino UNO. Estas mejoras permetirán según análisis que el sistema pueda identificar la forma de onda con mayor precisión, adaptándose mejor a las señales generadas.

Consideraciones para la implementación

Antirebote por software: Dado que los pulsadores tienden a generar múltiples señales involuntarias cuando se presionan, la implementación de un filtro de antirebote puede ser fundamental. Este filtro podría asegurar que las lecturas sean precisas, evitando activaciones erróneas del sistema.

Sincronización de adquisición y procesamiento: Otro factor importante es lograr una sincronización efectiva entre la adquisición de datos y su procesamiento. Al no poder realizar ambas tareas simultáneamente debido a las limitaciones del procesador, es recomendable adoptar un enfoque secuencial, en el cual se adquieran todas las muestras primero, y una vez completado

el buffer, se realicen los cálculos pertinentes. Esto garantiza que los datos procesados sean consistentes y se eviten errores durante el cálculo de los parámetros.

Simulación en Tinkercad: Aunque Tinkercad es una herramienta valiosa para simular sistemas con Arduino, presenta algunas limitaciones en la representación exacta de señales en tiempo real, especialmente en términos de rendimiento. Por ello, es importante validar los resultados obtenidos en la simulación y, si es posible, realizar pruebas adicionales en un entorno físico con el hardware real para garantizar que los algoritmos funcionen correctamente en condiciones reales (consideración a futuro).