Desafio I

Informática II

Eder Luis Polo Humanez

Isabella Carvajal Cuartas

Universidad de Antioquía

Semestre 2024-2

Según nuestro análisis descubrimos la necesidad de seccionar nuestro proyecto a partir de momentos los cuales nos permitirán facilitar el entendimiento y la buena distribución de las tareas. Los momentos fueron creados en una situación de jerarquía en los cuales cada momento depende del momento inmediatamente anterior. Definidos de la siguiente manera:

1. Configuración del Arduino: En esta sección establecemos qué componentes necesitamos y su respectiva conexión (para luego elaborar el código a partir de tales conexiones).

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Cantidad** | **Componente** |
| Microcontrolador | 1 | Arduino Uno R3 |
| pulsador I, pulsador F | 2 | Pushbutton |
| FUNC1 | 1 | 2 Hz, 1 V, 0 V, Square Function Generator |
| RR1, RR2 | 2 | 10 kΩ Resistor |
| U3 | 1 | LCD 16 x 2 |
| Rpot1 | 1 | 10 kΩ Potentiometer |
| RR-LCD | 1 | 220 Ω Resistor |

Componentes:

La vista esquemática es: **(anexo a.).**

1. Acondicionamiento del código: en esta sección evaluamos las salidas y entradas imprescindibles para el código que se desarrollará en el momento posterior.

* Ya que estaremos haciendo uso de una pantalla LCD de 2x16, debemos incluir la librería *LiquidCrystal.h,* definiendo sus entradas y salidas de la siguiente manera LiquidCrystal “nombre de la función” (RC, E, DB4, DB5, DB6, DB7).
* Como tenemos un generador de funciones y dos pulsadores, sus entradas serían (Pin análogo 0, pin digital 3 y 2).
* Así mismo, la declaración de respectivas entradas en el código.

1. Desarrollo del código: en este apartado desarrollaremos toda la estructura de funcionamiento del programa y para un mejor entendimiento de este, tendrá 4 sub momentos que contienen los aspectos determinantes para la elaboración del código y la división de la carga de trabajo.
   1. Creación arreglo: en un arreglo bidimensional dinámico con filas definidas por el tamaño disponible en la memoria y dos filas basadas en el tiempo y el valor de la función en aquel instante de tiempo.
   2. Configuración inicio/final de captura: este apartado estará compuesto por dos condicionales los cuales indican el inicio y final de la captura de datos dependiendo de los pulsadores conectados a las entradas digitales 3 y 2 respectivamente. Acompañado de la definición de 3 funciones que serán las determinantes del proceso actual, ya sea de inicio, captura o final recolección de datos. Estas funciones se encargarán también de la impresión.
   3. Captura de datos: en este momento realizaremos la captura de los datos del generador de funciones los cuales serán registrados en arreglo previamente creado.
   4. Análisis de datos: a partir de los datos capturados en el momento anterior, tendremos:
      1. Definir una función que determine el tipo de señal a partir del comportamiento de tal señal y sus características, como la pendiente, los picos y valles y las transiciones abruptas. El porcentaje para decretar qué función es del 70%.
      2. Definir una función la cual reciba el tipo de señal de la función anterior y dependiendo de esta calcule la amplitud y la frecuencia y fase.
   5. Liberación: Se definirá una función para liberar memoria del arreglo dinámico (matriz) a partir de la invocación de esta en las funciones mencionadas en el apartado 3.2.

**Análisis de los momentos:**

Para la elaboración de los momentos fue necesario hacer uso de ciertos conceptos teóricos tales como:

Momento 1

* Documentación de Arduino UNO r3.
* Funcionamiento de los pulsadores.
* Funcionamiento de potenciómetro.
* Valor de una resistencia.
* Documentación de pantalla LCD 16x2
  + Análisis de sus pines.
  + Uso de pines en código.
  + Uso de la librería de *LiquidCrystal.h.*

Momento 2

* Implementación de la documentación anterior:
  + Conexión de LCD en Arduino (pines digitales del 4-9).
  + Conexión de pulsores (Pin digital 2 y 3) con configuración PULL\_DOWN.
    - *const int startPin = 3;*
    - *const int stopPin = 2.*
  + Conexión de generador de funciones
    - *const int signalPin = A0*.
  + Uso de la librería *LiquidCrystal.h.*
    - *LiquidCrystal lcd(4, 5, 6, 7, 8, 9).*

Momento 3

3.1

-Para indicar el inicio del programa se imprimirá en la pantalla LCD de esta manera:

*lcd.setCursor(0, 0);*

*lcd.print(“Inicio programa”);*

- Para hacer la creación del arreglo dinámico bidimensional lo hacemos definiendo un puntero de punteros llamado matriz[filas][columnas]. Para crear las filas y columnas de aquel arreglo es necesario primero establecer la cantidad de columnas, sabemos que serán dos ya que se almacenarán dos datos definidos (tiempo y voltaje de la señal con respecto al tiempo), el numero de filas fue definido a partir de la experimentación, la consideración de la cantidad de datos mínima para extraer las características de la señal y la memoria necesaria para los otros elementos del código.

3.2.

-En este apartado serán necesarias tres funciones que están íntimamente relacionadas con los pines y el inicio y final de la captura de datos. Así mismo informan al usuario el estado actual del programa.

-Es necesario resaltar que estas funciones imprimirán tanto en el serial como en el LCD porque hemos detectado ciertos problemas con la pantalla LCD, que en algunas ocasiones sin razón específica no imprime. Creemos que son limitaciones aleatorias que tiene el simulador.

*void iniciarCaptura(): adecua las variables, así mismo la bandera para la recolección de datos e imprime el inicio de la captura.*

*void capturarDatos(): almacena en el arreglo dinámico los valores del tiempo y los valores de la señal para cada tiempo.Así mismo imprime la matriz para visualizar los valores que toman las columnas.*

*void finalizarCaptura(): Cambia la bandera llamada capturaActiva para que no ingrese en el condicional anterior. Así mismo, invoca las funciones para analizar la señal y por último libera la memoria. Dependiendo el tipo de señal imprimirá el tipo en el serial y la pantalla LCD. Para imprimir los valores del análisis en la pantalla LCD utilizamos un ciclo que esta respaldado por la librería LiquidCrystal.h*

- En este apartado es necesario la creación de dos condicionales que tendrán sus respectivas banderas “pull-down” las cuales servirán:

Start pin para indicar el inicio y la respectiva captura de datos.

Cuando Start pin se encuentre en HIGH y una variable tipo bool llamada capturaActiva sea diferente de verdadera entonces se invocará la función iniciarcaptura();

+ Para indicar el inicio de la recolección de datos se imprimirá en la pantalla LCD y en el serial de esta manera:

*lcd.setCursor(0, 0);*

*lcd.scrollDisplayLeft(“Inicio captura datos”);*

*Serial.println("Inicio captura");*

Cuando capturaActiva sea verdadera tendrá indentada dos condicionales:

1. Si el tiempo actual menos el tiempo de inicio es menor que el tiempo máximo hará la captura de datos (es decir invoca la función capturarDatos()), de lo contrario, finalizará la captura de datos (Invocando la función finalizarCaptura()).
2. Si el stop Pin es presionado, es decir igual a HIGH antes de los 22 segundos, invocara la función finalizarCaptura(), para finalizar la captura de datos.

Estos condicionales fueron realizados porque:

Stop pin.será tomado en cuenta si no se han pasado los 13 segundos máximos de recolección de datos, y al ser presionado se definirá el tiempo de captura de datos.

Stop pin no será tomado en cuenta, al momento en el que el tiempo supere los 13 segundos anteriormente mencionados, así entonces, el programa se ejecutará automáticamente.

+ Para indicar la finalización de la recolección de datos se imprimirá en la pantalla LCD de esta manera:

*lcd.setCursor(0, 0);*

*lcd.print("Captura finalizada");*

*Serial.println("Finalizando captura de datos...");*

3.3.

-En esta sección teniendo en cuenta que ya se indico el inicio en los pulsadores, los datos suministrados por el generador de señales serán ingresados en el arreglo con su tiempo respectivo, obtenido mediante la función *millis().*El tiempo máximo de recolección será de 13 segundos*.* Esta recolección de datos la haremos con un tiempo de muestreo cada 100 milisegundos.

3.4.

3.4.1.

- Esta función será invocada por finalizarCaptura().

-Definiremos la función llamada void tiposenal() la cual recibe como parámetro un entero y haremos uso del paso de parámetro por referencia para que la función cambie directamente el contenido de aquella variable (tipoSenal1).

- Definiremos ciertas variables las cuales son las características que destacan en nuestras señales para así clasificarlas. (picos, valles, transiciones bruscas, pendiente anterior, y umbral pendiente) La variable de umbral pendiente define un rango de error para depurar los cambios en pequeñas diferencias entre pendientes.

- Las variables anteriores nos permitirán identificar el tipo de señal. Para definir el tipo será dándole un valor al entero que dirá; 1: tipo senoidal; 2: tipo triangular; 3: tipo cuadrada; 4: tipo desconocida.

3.4.2.

-Esta función será invocada por la función finalizarCaptura().

-Esta función recibirá el nombre void analisis\_senal(int tipoSenal, float &amplitud, float &frecuencia, float &fase) hará uso del paso de parámetro por referencia para cambiar directamente el contenido de la variable en su dirección de memoria.

-Según los patrones del tipo de señal que nos indicara el primer parámetro calcularemos la amplitud, la frecuencia y la fase (la cual aún no es seguro si esta será calculada, sin embargo la tomamos en cuenta).

3.5

-Esta función llamada void liberacion() se ejecutará cuando la función finalizarCaptura() la invoque justo antes del final de la ejecución.

**Esquema**

El esquema seleccionado fue una tabla la cual se encuentra en el repositorio bajo el nombre de esquema y es un archivo de una tabla de Excel en la que se podrá apreciar la distribución de tareas, funciones, etc. **(anexo b)**

Para entender la tabla se define:

Resp.: Responsable; c: conjunto; Isa: Isabella Carvajal; Eder: Eder Polo.

Funciones: Funciones asignadas a cada responsable.

Variables: Variables asociadas a tales funciones.

Variables conjunto: variables que necesita la función, pero son definidas por fuera de la función, ya sea que se definan en el programa principal o en otra función.

Inv.: Indica si aquella función o momento invoca otra función.

**Anexos: se encuentran en el repositorio.**

1. Archivo Circuito.pdf.
2. Archivo Esquema.xlsx