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 5 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.
   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 e impresión: 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 de impresión la cual recibe el tipo de señal, si es desconocida imprimirá “Señal desconocida” y si es cualquier otro tipo de señal invoque la siguiente función (mencionada en el apartado 3.3.3.). La impresión se realizará en una pantalla LCD de 2x16. También retornará 1.
      3. 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 del retorno 1 de la función de impresión (3.3.2).

**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, 1);*

*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 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.

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

*LCD.setCursor(0, 1);*

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

Stop pin.será tomado en cuenta si no se han pasado los 10 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 10 segundos anteriormente mencionados, así entonces, el programa se ejecutará automáticamente.

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

*LCD.setCursor(0, 1);*

*LCD.scrollDisplayLeft(“Captura datos finalizada”);*

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().* Esta recolección de datos la haremos con un tiempo de muestreo cada 100 milisegundos.

3.4.

3.4.1.

-Definiremos la función llamada string tiposenal() la cual no recibe ningún parámetro, sin embargo retornará el tipo de señal de las cuales hay 4 posibles, basado en un 70% de afinidad con las características principales de la señal.

- 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.

3.4.2.

-Esta función se llamará int mostrarmensaje ( string tiposenal) esta función recibirá el string de la función anterior. Si la señal es desconocida imprimirá que es desconocida. En cambio, si la señal es conocida invocará la función siguiente(analisis\_senal(tiposenal(),amplitud,frecuencia,fase)), imprimirá los valores de las variables cuyo contenido fue cambiado en la función.

3.4.3.

-Esta función recibirá el nombre void analisis\_senal(string 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.

LCD.setCursor(0, 1);

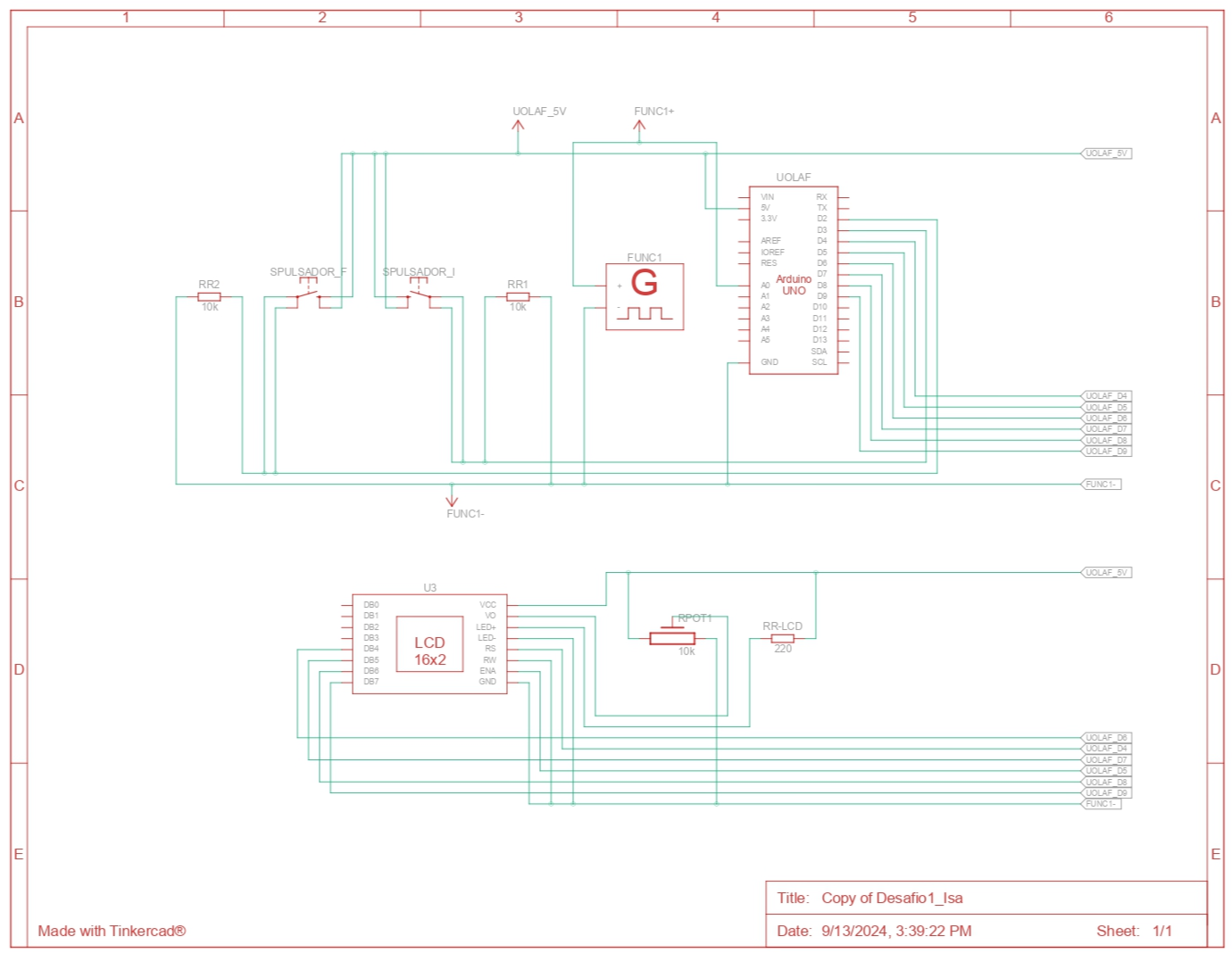
LCD.scrollDisplayLeft(“La señal es: ” + tiposenal);

LCD.setCursor(1, 1);

LCD.scrollDisplayLeft(“Fr: ” + frecuencia+”am: “+ amplitud+”fa: “+fase );

3.5

-Esta función llamada liberacion() se ejecutará siempre y cuando se haya hecho la ejecución de tipo de señal, y liberará la memoria del arreglo dinámico fila por fila.



**Anexos:**

a.