LUIS EDWIN GUTIERREZ ACEVEDO
SANTIAGO GARCÍA NARANJO

DESAFÍO 1

INTRODUCCIÓN:

Una onda es una perturbación que se propaga a través de un medio o del espacio, transportando energía de un lugar a otro sin un desplazamiento neto de materia.

Unas de las características principales de una onda son:

Amplitud: distancia máxima desde la posición de equilibrio hasta el punto más alto (cresta) o el punto más bajo (valle) de la onda. Existen muchas formas de calcular la amplitud, una forma sencilla de calcular la amplitud es:

$$A = |C - V| \quad (Ecuación\ 1)$$

donde C es la cresta, el punto más alto de la onda y V es el valle, el punto más bajo de la onda.

Frecuencia: número de ciclos completos que la onda realiza en un segundo. Se mide en hercios (Hz). La fórmula para calcular una onda es:

$$f = \frac{1}{T} \qquad (Ecuación\ 2)$$

Donde T Es el tiempo que tarda una onda en completar un ciclo.

Tipo de Onda: Los tipos de onda se refieren a la forma de la onda y cómo se propaga. Los principales tipos de onda son:

- Sinusoidal: Tiene una forma suave y repetitiva, como una sinusoide.
- Cuadrada: Alterna entre dos niveles de amplitud, creando una forma de onda cuadrada.
- Triangular: Tiene una forma de diente de sierra, subiendo y bajando linealmente.

En este desafío, se utilizará la herramienta Tinkercad para simular un arduino conectado a un generador de ondas, tal como se muestra en la siguiente imagen (**figura 1)** con el fin de crear un algoritmo capaz de medir la amplitud y la frecuencia de una onda e identificar el tipo de onda que se está obteniendo.

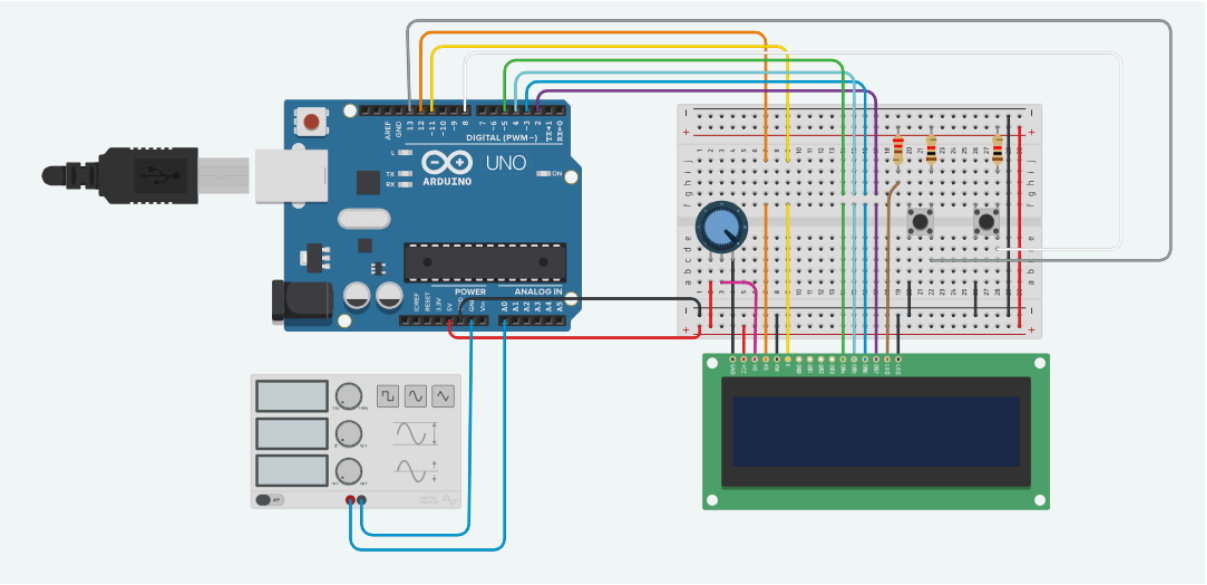
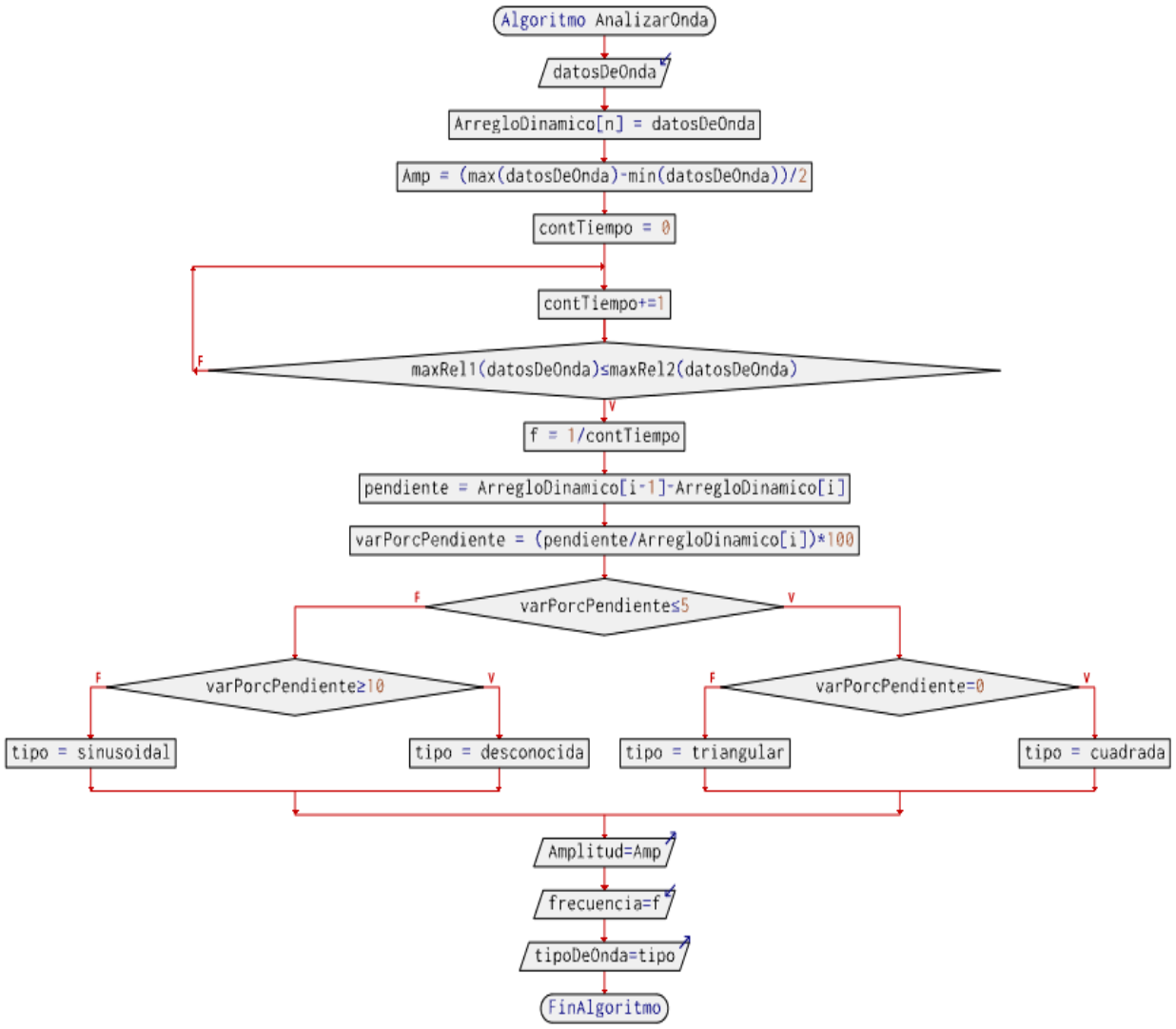


Figura 1: Generador de señales y pantalla LCD conectados a un arduino.

ANÁLISIS:

Luego de iniciar la simulación, se presiona el botón de la izquierda para recolectar los datos hasta que el usuario desee terminar el muestreo, para esto se presiona el botón de la derecha. Finalmente se debe mostrar en la pantalla la amplitud, frecuencia y forma de la onda registrada.

El siguiente diagrama de flujo muestra, a grandes rasgos, el análisis para lograr el objetivo del desafío.



El primer paso es recibir los datos del generador y luego almacenarlos en un arreglo dinámico, ya que no se conoce cuantos datos se podrán guardar durante la ejecución.

Para calcular la Amplitud se toma el dato más grande y se resta con el dato más pequeño del arreglo, el resultado se divide entre dos.

Para calcular la frecuencia se debe iniciar un contador para medir el tiempo que se demora la onda de llegar en este caso de una cresta (máximo relativo) a otra. La frecuencia es el inverso de este resultado.

Luego, para obtener la forma de la onda, se opta por tomar cada par de puntos de la onda y restarlos, esto corresponde a la pendiente, posteriormente se calcula la variación porcentual de estas pendientes y se consideran varios casos:

variación porcentual < 5 e igual a 0 o a la amplitud : las pendientes son 0 o la amplitud, la onda es cuadrada.

variación porcentual < 5 pero diferente de 0: las pendientes son casi constantes, la onda es triangular.
variación porcentual > 5 y mayor a 10: las pendientes varían mucho, la onda es desconocida.
variación porcentual < 5 y menor a 10: las pendientes varían ligeramente, la onda se parece mucho a una sinusoidal.

Por último, se imprimen los resultados en pantalla.

IMPLEMENTACIÓN DEL CÓDIGO:

1)Declaración de variables y funciones:

```
bool bottom = false;
float voltaje, voltajeMax, voltajeMin, frecuencia,periodo,puntoMedio;
String tipo;
short int capacidad = 11;
float* arr = new float[capacidad];
short int cantElementos = 0;
short int capMax = 317;

void voltajeMaximo(float voltaje,float& voltajeMax);
void voltajeMinimo(float voltaje,float& voltajeMin);
void guardarEnArreglo(float*& arr, short int& capacidad, float nuevoElemento,short int& cantElementos);
void redimArr(float*& arr, short int& capacidad);
void miCopy(float* inicio, float* fin, float* destino);
void mostrarResultados(float voltaje, float frecuencia, char tipo);
float calcularPeriodo(float* arr, short int cantElementos, float puntoMedio);
```

En este caso se decidió utilizar tipos de variables como float para los datos obtenidos por el generador de ondas, se utiliza un float ya que no se necesita tanta precisión, por tanto menos espacio,para estos números. Por otro lado, se utiliza short int ya que los números enteros que se manejan no son mayores a mil, por tanto este tipo de dato lo puede soportar perfectamente. Por último se utiliza un bool para los cambios de estado y un string para guardar la palabra que contiene el tipo de onda que se obtuvo.

El valor de capacidad es el tamaño inicial del arreglo, y capMax es el tamaño máximo que este arreglo tendrá, se define este límite ya que no se puede almacenar todos los datos de un tiempo indefinido, la cantidad de espacio en memoria dinámica que nos brinda este programa es limitado y si supera la capacidad máxima este mismo colapsa. La forma en que se halló capMax fue modificando el valor de capacidad y el índice de redimensionamiento (VEA PASO 6) luego se imprime en pantalla el número de elementos del arreglo (cantElementos) y redimensionando este mismo hasta que colapse la simulación.

2) Función loop()

```
void loop() {
    if (digitalRead(13) == LOW) // si detecta botón inicio (izquierdo)
    {
        bottom = true;
        voltajeMax = -10.0;
        voltajeMin = 10.0;
    }
    while (bottom == true)
    {
        voltaje = analogRead(A0)* (5.0 / 1023.0);
        guardarEnArreglo(arr, capacidad, voltaje, cantElementos);
        //Serial.println(millis()); //Al hacer esto se observa que la tasa de refresco es de 6 milisegundos
        voltajeMaximo(voltaje,voltajeMax);
        voltajeMinimo(voltaje,voltajeMin);
        Serial.println(voltaje);
        if (digitalRead(8) == LOW)
        {
            bottom = false;
            puntoMedio = (voltajeMax + voltajeMin) / 2.0;
            periodo = calcularPeriodo(arr, cantElementos, puntoMedio);
            frecuencia = 1.0 / periodo;
            tipo=tipoOnda(arr,cantElementos,voltajeMax,puntoMedio,frecuencia);
            mostrarResultados((voltajeMax - voltajeMin) / 2.0, frecuencia, tipo); // imprimir en pantalla
            delete[] arr;
            cantElementos = 0;
            capacidad = 11;
            arr = nullptr;
            arr = new float[capacidad];

        }
    }
}
```

Al iniciar la simulación no pasará nada hasta que se presione el botón izquierdo (13), cuando esto suceda se cambiará los valores de bottom (para entrar al ciclo while) y asignará los valores de los voltajes mínimos y máximos a los valores observados en la imagen ya que estos si o si cambiarán debido al rango del voltaje.

Se entra al ciclo while y se hace la conversión de bits a voltaje, debido a que el arduino utiliza un convertidor analógico-digital (ADC) de 10 bits, estos datos se guardan en un arreglo dinámico.

Al presionar el boton derecho (8) se procesan los datos obtenidos y se guardan los valores más relevantes, para mostrarlos en pantalla y luego liberar espacio en memoria borrando el arreglo dinámico y reiniciando otros valores como la capacidad y la cantidad de elementos.

3)Función voltajeMax y voltajeMin:

```
void voltajeMaximo(float voltaje,float& voltajeMax)
{
    if (voltaje>=voltajeMax)
    {
        voltajeMax = voltaje;
    }

}

void voltajeMinimo(float voltaje,float& voltajeMin)
{
    if (voltaje<=voltajeMin)
    {
        voltajeMin = voltaje;
    }

}
```

compara los valores de los voltajes obtenidos y si hay algún cambio modifica el valor anterior de la variable de ámbito global correspondiente.

4)Función tipoOnda:

```
String tipoOnda(float* arr, int cantElementos,float voltajeMax,float puntoMedio,float frecuencia){
    float pendienteCresta1,diffPendientes,pendientePM1;
    float comparacion=frecuencia*(voltajeMax-voltajeMin)/100;
    bool repe=true;
    String tipo;

    for (int i=3;i<cantElementos;i++)
    {
        if ((voltajeMax - arr[i]<=0.1)&& repe ) //cerca de una cresta
        {
            pendienteCresta1=arr[i]-arr[i-1];
            repe=false;
        }
        if ((arr[i-1] < puntoMedio && puntoMedio <= arr[i])&& !repe) //cerca del punto medio
        {
            pendientePM1=arr[i]-arr[i-1];
            //}
            break;
        }
    }
    diffPendientes=pendientePM1-pendienteCresta1;
    if (diffPendientes==0||diffPendientes==(voltajeMax-voltajeMin)/2)
    {
        tipo="cuadrada";
    }
    else if (diffPendientes>0 && diffPendientes<=frecuencia*(voltajeMax-voltajeMin))
    {
        if(diffPendientes<comparacion)
        {
            tipo="triangular";
        }
        else
        {
            tipo="sinusoidal";
        }
    }
    else
    {
        tipo="no identificada";
    }

    return tipo;
}
```

Recibe el arreglo con los datos más recientes del arreglo dinámico para luego recorrerlo, luego, cuando esté cerca de una cresta, en esos puntos calcula la diferencia entre dos puntos consecutivos, su pendiente. Lo mismo hace para cuando está cerca del punto medio.

Luego, calcula las diferencias entre estas pendientes y en base a esto se puede dar un veredicto sobre la forma de la onda dependiendo de su amplitud y su frecuencia.

5) Función calcular periodo:

```
float calcularPeriodo(float* arr, short int cantElementos, float puntoMedio) {
    bool cruzando = false;
    float tiempoCruzadoInicio = 0;
    float tiempoCruzadoFinal = 0;
    short int cruces = 0;

    for (int i = 1; i < cantElementos; i++)
    {
        if ((arr[i-1] < puntoMedio && puntoMedio <= arr[i]) || (arr[i-1] > puntoMedio && puntoMedio >= arr[i]))
        {
            if (!cruzando)
            {
                tiempoCruzadoInicio = millis();
                cruzando = true;
            }
            else {
                cruces++;
                if (cruces == 2)
                {
                    tiempoCruzadoFinal = millis();
                    break;
                }
            }
        }
        if(cruzando)
        {
            delay(6);
        }
    }

    tiempoCruzadoFinal=millis()-tiempoCruzadoInicio;

    if (cruces < 2)
    {
        return 0; // No se detectaron suficientes cruces para calcular el periodo
    }

    return (tiempoCruzadoFinal)/1000.0; // Convertir a segundos
}

void mostrarResultados(float voltaje, float frecuencia, String tipo) {
```

Recibe el arreglo dinámico y lo recorre, cuando pasa por el punto medio registra el tiempo de inicio, se repite este proceso hasta que pasa dos veces por el punto medio y ahí registra el tiempo final. Cabe resaltar que Tinkercad registra los datos cada milisegundos, por lo tanto se debe tomar en cuenta para el recorrido del arreglo.

Al final, se calcula el intervalo del tiempo final- inicial y se convierte en segundos para obtener el periodo de la onda registrada.

6) Función myCopy, redimArr y guardar en arreglo:

ALTERNATIVAS:

Para encontrar la forma de la onda con más seguridad se puede calcular las pendientes punto a punto y luego compararlas con las pendientes esperadas para asegurarse de que sean lo más similares posibles.

Esta solución aunque muy precisa es poco eficiente ya que se necesitaría mucho más espacio en memoria del usado en la propuesta original.

```
void miCopy(float* inicio, float* fin, float* destino) {
    while (inicio != fin) {
        *destino = *inicio;
        ++inicio;
        ++destino;
    }
}

void redimArr(float*& arr, short int& capacidad) {
    short int nuevaCap = capacidad * 30;
    float* nuevoArr = new float[nuevaCap];
    miCopy(arr, arr + capacidad, nuevoArr);
    delete[] arr;
    arr=nullptr;
    arr = nuevoArr;
    capacidad = nuevaCap;
}

void guardarEnArreglo(float*& arr, short int& capacidad, float nuevoElemento, short int& cantElementos) {
    if (cantElementos == capacidad) {
        redimArr(arr, capacidad);
    }
    arr[cantElementos] = nuevoElemento;
    if (cantElementos < capMax) {
        cantElementos++;
    } else {
        for (int j = 0; j < capacidad; j++) {
            arr[j] = arr[j + 1];
        }
    }
}
```

La función myCopy toma dos arreglos y copia sus datos.

La función redimArr redimensiona un arreglo basado en un índice el cual en este caso, luego de varios intentos y colapsos de la simulación, se decidió que será 30, esto significa que redimensiona 30 veces su capacidad original.

La función guardarEnArreglo primero revisa si la capacidad original ha sido excedida, si lo hizo entonces redimensiona el arreglo, luego almacena el nuevo elemento y por último revisa si se supera la capacidad máxima, si lo hace, borra el dato más antiguo.

ANEXOS: Circuit design Fabulous Amur-Snaget - Tinkercad - Link de la simulación en Tinkercad.