Proyecto Final: Caracterización de una taza de café a partir de su sonido

Juan David Ochoa, Federico Villadiego[1, *]

[1] *Departamento de Física, Universidad Nacional de Colombia, Bogotá, Colombia*

(Dated: 9 de febrero de 2022)

Resumen: A partir del conocimiento de la posición de golpes sobre la boca de una taza y la frecuencia que estos producen se busca determinar la dirección del asa. Para determinar la posición de cada golpe se utilizaron las salidas digitales de tres sensores de sonido KY-037 con los cuales se encuentra la diferencia de tiempos de llegada de un mismo frente de onda. Previo a esto se midió la temperatura con un sensor KY-001 para obtener un valor de la rapidez del sonido. Se utiliza la salida análoga de otro sensor KY-038 para obtener la amplitud de la señal en función del tiempo y se realiza la transformada de Fourier para asociar una frecuencia característica a cada golpe. Debido a la presición de los tiempos no fue posible reconstruir con exactitud la posición del centro de la taza y su radio, por lo que se decidió dar estos como parámetros en la solución del problema. Con esta información, se obtiene una gráfica con los puntos asociados a los golpes y sus frecuencias, con mediciones exitosas en el 64 % de los casos. Se observa la distribución de frecuencias que predice la teoría y finalmente se pinta una referencia que indica la posición del asa respecto a la taza con una incertidumbre de 20° medidos desde su centro, cumpliendo el objetivo principal del problema planteado dentro la presición lograda.

Palabras clave: Taza de café, sonido, multilateración, transformada de Fourier.

I. Introducción

A. Vibración de un anillo y efecto del asa en las frecuencias de una taza

Una taza de café típica tiene dos frecuencias generales asociadas a la posición del asa, diferente a un objeto con simetría cilíndrica donde no se observa este fenómeno. Con el propósito de describir este hecho, se empieza por modelar la boca de la taza como un anillo delgado [1]. Para ello se hacen las siguientes tres suposiciones:

- En un estado sin deformar, el anillo forma una circunferencia con radio R.
- La sección transversal del anillo con área A es siempre constante.
- La oscilación es libre, sin restricciones.

En base a esto, un análisis de las fuerzas y momentos flectores en el libro citado lleva a la deducción de la siguiente ecuación diferencial para la flexión radial w en función del ángulo θ medido desde el centro del anillo sin deformar:

$$\frac{\partial^6 w}{\partial \theta^6} + 2\frac{\partial^4 w}{\partial \theta^4} + \frac{\partial^2 w}{\partial \theta^2} + \frac{\rho A R^4}{E I_1}\frac{\partial^2}{\partial t^2}\left(\frac{\partial^2 w}{\partial \theta^2} - w\right) = 0 \tag{1}$$

Aquí E representa el módulo de Young, ρ la densidad e I_1 el momento de inercia de la sección transversal sobre un eje perpendicular al plano del anillo y que pasa por el centroide.

Se asume entonces una solución armónica:

$$w(\theta, t) = C_1 \sin(n\theta + \phi)e^{i\omega t} \tag{2}$$

Y se obtienen las frecuencias naturales:

* juochoaa@unal.edu.co Estudiante Dep. Física
fvilladiego@unal.edu.co Estudiante Dep. Física

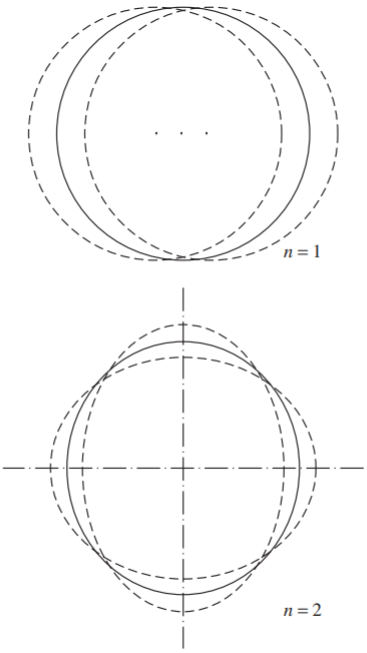


Figure 12.3 Mode shapes of a ring.

Figura 1: Modos de oscilación del anillo, el primero es equivalente a una traslación de un cuerpo rígido y por tanto se descarta. Tomado de [1].

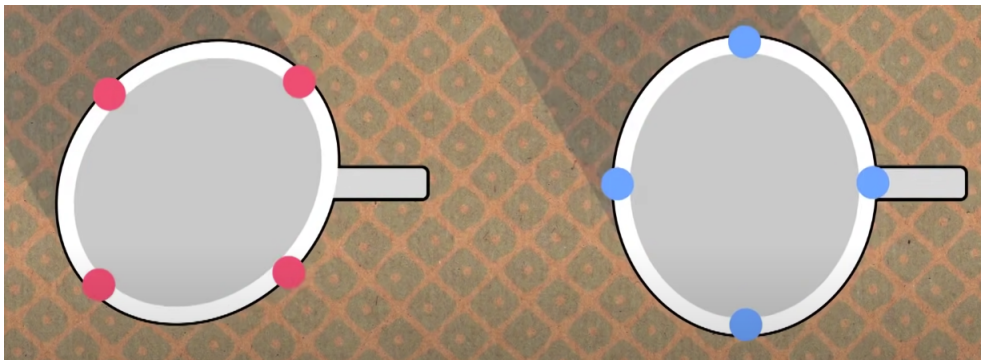


Figura 2: Oscilaciones radiales de una taza de café que es golpeada sobre puntos concretos. Tomado de [2].

$$\omega_n^2 = \frac{EI_1}{\rho A R^4} \frac{n^6 - 2n^4 + n^2}{n^2 + 1}, \quad n = 2, 3, 4... \tag{3}$$

La ecuación 2 representa la aproximación a la taza sin asa. La *Figura* 1 muestra los dos primeros modos. Las oscilaciones asociadas a una taza son descritas principalmente por el modo normal $n = 2$. Golpear un punto de la taza es equivalente a convertir ese punto en un antinodo, y siguiendo la ecuación 2, los puntos a un ángulo múltiplo de $\frac{\pi}{2}$ lo serán también.

La masa del asa incrementa la inercia y por ende, al golpear un punto sobre un ángulo medido desde del asa igual a $m\frac{\pi}{2}$, con $m \in \mathbf{Z}$ (puntos azules en la *Figura* 2), la frecuencia será menor a que si se golpea en un punto intermedio (puntos rojos). En este último caso, a $\theta = (2m+1)\frac{\pi}{4}$, el asa cae en un nodo lo que implica que no se mueve y se alcanza la mayor frecuencia. Por su parte, puntos intermedios a estos resultan en una combinación de los dos movimientos [3].

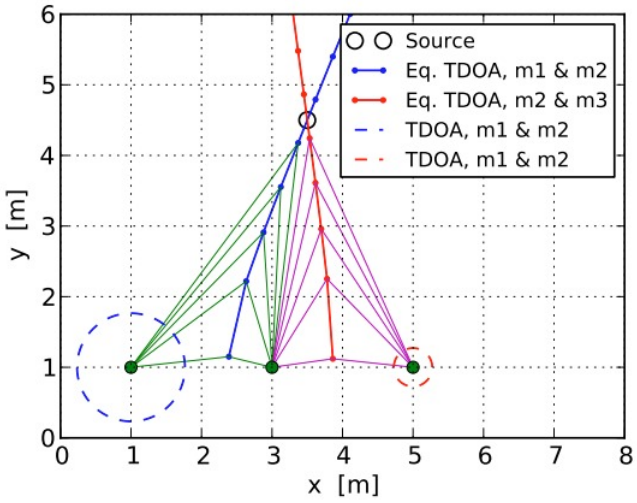


Figura 3: Ejemplo de multilateración. En la imagen, TDOA significa la diferencia en tiempos de llegada (time difference of arrival). Los círculos punteados corresponde al radio que recorre el sonido en ese tiempo. Tomado de [5].

B. Transformada de Fourier

La transformada de Fourier es un método ampliamente conocido para describir cierta función periodica en el tiempo en el dominio de las frecuencias [4]. Matemáticamente se define como:

$$X(\omega) = \int_{\infty}^{\infty} x(t) e^{-i\omega t} dt \tag{4}$$

Sin embargo para el propósito de nuestro proyecto resulta mas conveniente realizar una transformada rápida de fourier, FFT. Este es un algoritmo basado en una transformada discreta de Fourier pero mucho mas rápido computacionalmente. Este procedimiento está bien extendido en librerías de Python como numpy o scipy.

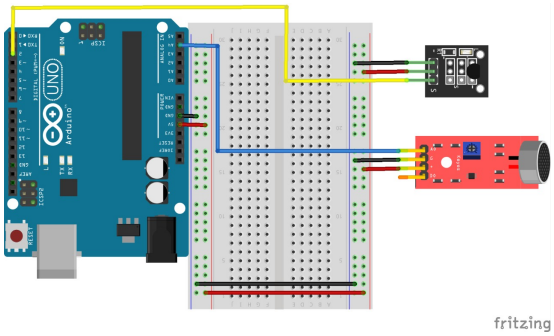
Es de aclarar que una transformada de Fourier devuelve una función $X(\omega)$ compleja, o en el caso de la FFT un conjunto de númeroos complejos ,por lo cual el procedimiento general es tomar el módulo de este y graficarlo respecto de las diferentes frecuencias que descomponen una señal.

C. Multilateración

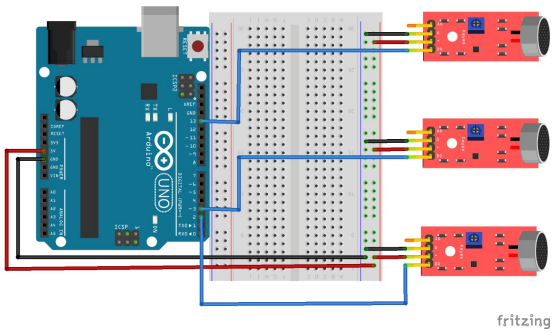
Es un método que usa la diferencia de tiempos de llegada a un par de sensores para localizar una señal [5]. Con la velocidad de la señal se encuentra la diferencia en distancias del origen a los sensores, y se considera que todos los puntos en el espacio en los que este valor es constante forman una hipérbola; si se toma ahora otro par de sensores el problema se reduce a encontrar la intercepción entre dos curvas y discernir entre las soluciones posibles la correcta. En un montaje como el de la *Figura* 3 se tienen tres pares de sensores cuyas hipérboles presentan intercepción triple única.

II. Montaje Experimental

Se utilizaron dos Arduino UNO en el montaje. Uno de ellos está conectado a un módulo KY-001 por medio de un pin digital y a un módulo KY-038 con función analógica (*Figura* 4a). Este parte del montaja se ubica justo al lado de la taza, a una distancia no mayor a 10,0 cm para garantizar una buena medición de la frecuencia. A 60,0 cm de la taza se encuentra la estructura ilustrada en la *Figura* 5. Los módulos KY-037 de la *Figura* 4b se ubican a la altura de la boca de la taza con una separación de 49,0 cm entre ellos borde a borde. Este valor se toma de esta forma debido a que el micrófono del módulo tiene aproximadamente 0,9 cm de diámetro, y si se considera el área de medición de la *Figura* 5, un frente de onda producido en la taza debe impactar primero en estos puntos.



(a) Medición de frecuencias (KY-038) y temperatura (KY-001).



(b) Medición de la posición (KY-037 × 3).

Figura 4: Esquema de conexión de los sensores.

En la práctica no se encontró diferencia entre los módulos KY-038 y KY-037 para los propósitos del proyecto, se utilizaron ambos por cuestión de disponibilidad. En la *Figura* 6 aparecen fotos del montaje, donde se uso cinta métrica y una escuadra para intentar medir con la mayor presición posible las coordenadas del centro de la taza. En la imagen también aparece la cuchara con la que se realizan los golpes; con el propósito de tener un camino despejado entre estos y los sensores, los golpes se limitan al hemisferio derecho de la taza vista como en la primera foto, más específicamente, a una apertura de alrededor 100° con una línea paralela a la horizontal como bisectriz. Está es también la razón por la que se requieren distancias mayores a 60,0 cm entre la taza y los tres sensores de sonido.

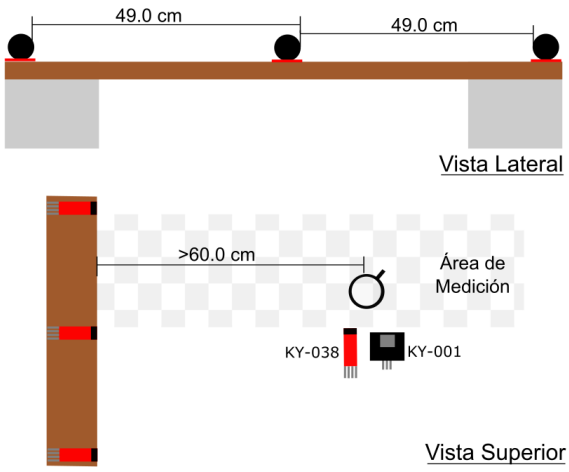


Figura 5: Ubicación de los sensores en el montaje.

III. Código fuente

Se utilizaron 3 códigos de arduino y 3 códigos de python. Un código de arduino es usado para obtener la temperatura ambiente previo al muestreo de los golpes. Luego, los otros dos códigos de arduino se cargan en placas diferentes y los códigos de python que reciben información del monitor serial se ejecutan en paralelo desde consolas distintas. Finalmente, el tercer código de python lee archivos generados por estos dos últimos y entrega la gráfica final. En el diagrama de flujo de la última hoja se resume el proceso completo, donde los recuadros azules representan código de arduino y los verdes de python. A continuación se adjuntan los códigos usados, las funciones aparecen detalladamente descritas en comentarios dentro del código.

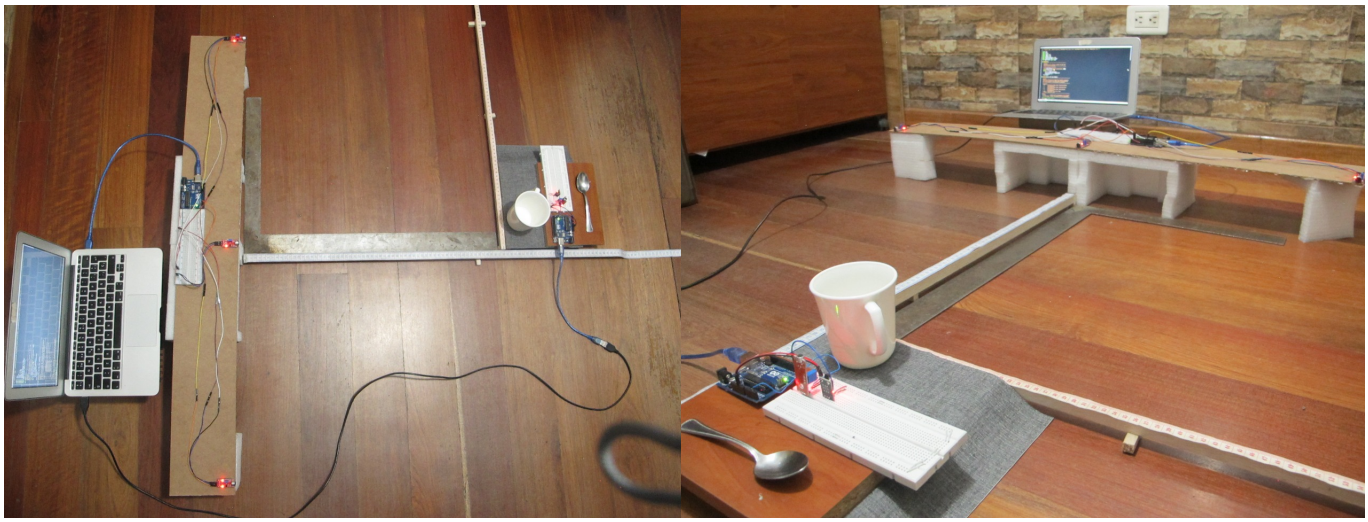


Figura 6: Fotos del montaje realizado.

A. Arduino

Para la medición de la temperatura se utilizan las librerías OneWire y DallasTemperature para recibir e interpretar datos del sensor KY-001:

```
#include <OneWire.h>
#include <DallasTemperature.h>
//Sensor conectado al pin digital 2
#define KY001 2
//Configuraci n de librer as
OneWire oneWire(KY001);
DallasTemperature sensors(&oneWire);
void setup(void)
{
  //Inicializaci n puerto serial
  Serial.begin(9600);
  //Inicializaci n sensores
  sensors.begin();
}
void loop(void)
{
  //Recibe datos de sensores
  sensors.requestTemperatures();
  //Y los imprime en el puerto seria
  Serial.println(sensors.getTempCByIndex(0));
}
```

Listing 1: Código de arduino para la medición de la temperatura

Otro código de arduino utiliza las entradas digitales de los KY-037 para encontar la diferencia de tiempos entre las detecciones de los sensores. Por las limitaciones de la placa de Arduino UNO, solo se utilizan interrupciones para dos de los sensores, la otra medición se hace desde el loop.

```
//Pines digitales conectados a los sensores Ky-037
int micro0_Pin = 2;
int micro1_Pin = 3;
int micro2_Pin = 13;

//Seguro para evitar detecciones m ltiples
int timerL;
int timerC;
int timerR;

//Diferencias de tiempo en microsegundos
int tLC;
```

```
int tCR;
int tLR;

//Tiempos en microsegundos
unsigned long TimeL;
unsigned long TimeC;
unsigned long TimeR;

void setup() {
  Serial.begin(9600);
  pinMode(micro0_Pin, INPUT);
  attachInterrupt(digitalPinToInterrupt(micro0_Pin), Left, RISING);
  pinMode(micro1_Pin, INPUT);
  attachInterrupt(digitalPinToInterrupt(micro1_Pin), Center,RISING);
  pinMode(micro2_Pin, INPUT);
}
void Left() {
  // Detecci n del sensor izquierdo
  if (timerL == 0){
    TimeL = micros();
    timerL = 1;
  }
}
void Center() {
  // Detecci n del sensor central
  if (timerC == 0){
    TimeC = micros();
    timerC = 1;
  }
}
void loop() {
  //Detecci n del sensor derecho
  if (timerR == 0 && digitalRead(micro2_Pin) == HIGH){
    TimeR = micros();
    timerR = 1;
  }
  if (timerL*timerC*timerR==1){
    //Si se realizaron tres detecciones, imprime las diferencias de tiempos
    tLR=TimeL-TimeR;
    tLC=TimeL-TimeC;
    tCR=TimeC-TimeR;
    Serial.println(tLR);
    Serial.println(tLC);
    Serial.println(tCR);
    delay(1000);
    timerL=0;
    timerC=0;
    timerR=0;
  }
}
```

Listing 2: Código de arduino con las entradas digitáles de los sensores Ky-037

El otro código simplemente recibe la información de la salida análoga del sensor KY-038 y la imprime en el monitor serial:

```
//Pin conectado a la salida an loga del sensor KY-038
const float OUT_PIN = A4;

void setup() {
    Serial.begin(9600);
}
void loop() {
  //Impresi n de los valores en el monitor serial
    Serial.println(analogRead(OUT_PIN));
}
```

Listing 3: Código de arduino con las entradas análogas del sensor Ky-038

B. Python

El primer código de python recibe desde consola la información de la posición de la taza y la temperatura que usa para estimar la velocidad del sonido[1], además se comunica con el arduino que tiene las entradas digitales para encontrar las hipérbolas donde la diferencia en las distancias entre el origen del sonido y cada par de sensores es constante; con esto es posible marcar un punto aproximado sobre la circunferencia de la taza por medio de la intercepción de curvas.

```
1
2  """ Ejecutar como:  python3 Origins.py <Coordenada Centro Taza X (cm)> <Coordenada Centro Taza Y (
       cm)> <Temperatura ( C )> """
3
4  import sys
5  import math
6  import serial
7  import numpy as np
8  import sympy as sym
9  import pandas as pd
10 import matplotlib.pyplot as plt
11 import matplotlib.ticker as ticker
12 from sympy.solvers import solve
13
14 def ReadSave():
15     """-----------------------------------------------
16     ReadSave:
17     Se comunica con el arduino y guarda los datos del
18     monitor serial en Data.txt.
19     -----------------------------------------------"""
20     File = open("Data.txt",'w')
21     ser = serial.Serial('/dev/cu.usbmodem14101', 9600)
22     print("\nReady to go\n")
23     for ii in range(3):
24         data = str(ser.readline())
25         File.write(data[2:(len(data)-5)])
26         File.write('\n')
27     File.close()
28
29 def Secants(LR, LC, CR, cup, radius):
30     """-------------------------------------------------------------
31     Secants:
32     Dadas las diferencias de tiempo entre sensores, encuentra
33     los posibles origenes del sonido sobre la circunferencia
34     de la taza mediante la intercepci n de hip rbolas.
35     -------------------------------------------------------------
36     Argumentos:
37      * LR     : diferencia tiempo sensor izquierdo-derecho
38                 (microsegundos)
39      * LC     : diferencia tiempo sensor izquierdo-centro
40                 (microsegundos)
41      * CR     : diferencia tiempo sensor centro-derecha
42                 (microsegundos)
43      * cup    : coordenadas centro de la taza
44      * radius : radio de la taza
45     -------------------------------------------------------------
46     Return:
47      * Ix,Iy         : Arrays con las coordenadas de los origenes
48      * a, b2, Center : Arrays con par metros de las hip rboelas
49     -------------------------------------------------------------"""
50     #Convierte unidades a metros
51     Cup = [0,0]
52     for ii in range(2):
53         Cup[ii] = cup[ii] / 100
54     Radius = radius/100
55     #Arreglos necesarios para guardar informaci n
56     Center = np.zeros(3)
57     a = np.zeros(3)
```

[1] Se usa la fórmula $c[ms^{-1}] = \sqrt{T[K]/273{,}15}$ recuperada de Wikipedia

```
    b2 = np.zeros(3)
    c = np.zeros(3)
    Ix = []
    Iy = []
    #Llena los arreglos con los p rametros de la taza
    a[0] = LR
    a[1] = LC
    a[2] = CR
    a[:] *= v/(2e6) #Convierte tiempos a distancias en metros
    Center[0] = 0.45
    Center[1] = -49/2+0.45
    Center[2] = 49/2+0.45
    Center[:] /= 100
    c[0] = 49
    c[1] = 49/2
    c[2] = 49/2
    c[:] /= 100
    #Encuentra la intercepci n de cada hip rbole con la taza usando sympy
    P = sym.Symbol('P', real = "True")
    for jj in range(3):
        b2[jj] = c[jj]**2 - a[jj]**2
        Solution = solve( (P-Cup[0])**2 + (-sym.sqrt( b2[jj]*( (P-Center[jj])**2 / a[jj]**2 - 1))-
    Cup[1])**2 - Radius**2, P)
        for kk in Solution:
            #Simplifica la soluci n y la guarda en un arreglo
            xs = sym.N(kk)
            Ix.append(xs*100) #Convierte a cent metros
            ys = -math.sqrt(b2[jj]*( (xs-Center[jj])**2 / a[jj]**2 -1))
            #Descarte puntos en el hemisferio sur de la taza
            if ys < Cup[1]:
                Ix.pop(-1)
            else:
                Iy.append(ys*100) #Convierte a cent metros
    return Ix, Iy, a, b2, Center

def Plot(X,Y,a,b2,Center,cup,radius):
    """-------------------------------------------------------------
    Plot:
    Grafica de la secci n transversal de la taza, las hip rbolas
    que la atraviesan y los puntos de origen.
    -------------------------------------------------------------
    Argumentos:
     * X,Y           : Coordenadas de los puntos de origen (cm)
     * a, b2, Center : Arrays con par metros de las hip rbolas
     * cup           : coordenadas centro de la taza
     * radius        : radio de la taza
    -------------------------------------------------------------"""
    #Convierte las listas a arrays de numpy
    X = np.array(X)
    Y = np.array(Y)
    plt.style.use('dark_background') #Fondo negro
    fig, ax = plt.subplots()
    #Origenes
    ax.scatter(X,Y, color='m')
    #Sensores (normalmente fuera de plano)
    ax.scatter([-48.55,0,48.55],[0,0,0], color='r')
    #Taza
    circle = plt.Circle((cup[0], cup[1]), radius, color='b', fill=False)
    ax.add_patch(circle)
    #Hiperbolas
    xr = np.linspace(-100, 100, 400)
    yr = np.linspace(0, -100, 400)
    xr, yr = np.meshgrid(xr, yr)
    Center[:] *= 100
    a *= 100
    b2 *= 10000
    ax.contour(xr, yr,((xr-Center[0])**2/a[0]**2 - (yr)**2/b2[0]), [1], colors='w', linestyles='
    dashed', linewidths=0.5)
    ax.contour(xr, yr,((xr-Center[1])**2/a[1]**2 - (yr)**2/b2[1]), [1], colors='w', linestyles='
    dashed', linewidths=0.5)
```

```
    ax.contour(xr, yr,((xr-Center[2])**2/a[2]**2 - (yr)**2/b2[2]), [1], colors='w', linestyles='
    dashed', linewidths=0.5)
    Center[:] /= 100
    a /= 100
    b2 /= 10000
    #Par metros de la gr fica
    xlim_d = cup[0] - 3*radius
    xlim_u = cup[0] + 3*radius
    ylim_d = cup[1] - 3*radius
    ylim_u = cup[1] + 3*radius
    ax.set_xlim(xlim_d,xlim_u)
    ax.set_ylim(ylim_d,ylim_u)
    ax.set_aspect('equal')
    ax.xaxis.set_major_locator(ticker.MultipleLocator(5))
    ax.yaxis.set_major_locator(ticker.MultipleLocator(5))
    ax.grid(linestyle='-', linewidth=0.3)
    plt.show()

def mid_point(X,Y,cup,radius):
    """-------------------------------------------------------------
    mid_point:
    Encuentra el punto medio sobre la taza entre los puntos
    suministrados
    -------------------------------------------------------------
    Argumentos:
     * X,Y           : Coordenadas de los puntos de origen (cm)
     * cup           : coordenadas centro de la taza
     * radius        : radio de la taza
    -------------------------------------------------------------
    Return:
     * Point         : Coordenadas del punto medio (cm)
    -------------------------------------------------------------"""
    mean_theta = 0
    for ii in range(len(X)):
        mean_theta += math.atan( (X[ii] - cup[0]) / (Y[ii] - cup[1]) )
    mean_theta /= len(X)
    Point = [ radius*math.sin(mean_theta) + cup[0], radius*math.cos(mean_theta) + cup[1]]
    return  Point

def print_array(A):
    """----------------
    Imprime el array A
    -----------------"""
    for ii in A:
        print("{:.2f}".format(ii),"\t",end="")
    print()

"""---------------------------
             MAIN
---------------------------"""
v = 331.3 * math.sqrt( 1 - (float(sys.argv[3]) / 273) )  #Rapidez del sonido
cup = [float(sys.argv[1]), float(sys.argv[2])] #Posici n del centro de la taza
radius = 4.1 #radio de la taza
#Se guarda la informaci n de la taza
Sx = [cup[0]]
Sy = [cup[1]]
new = "y" #string para continuar mediciones
while("y" == new):
        #Comunicaci n con el arduino
        ReadSave()
        #Lee informaci n de Data.txt
        Data = np.loadtxt("Data.txt")
        #Estima el punto de origen un golpe a la taza
        print("Finding point. This could take a while...")
        X,Y,a,b2,Center = Secants(Data[0],Data[1],Data[2],cup,radius)
        if not len(X) == 0:
            #Encuentra el punto medio
            Mean = mid_point(X,Y,cup,radius)
            Plot([Mean[0]],[Mean[1]],a,b2,Center,cup,radius)
            accept = input("Do you accept this point? [y/n] ")
```

```
            if ("y" == accept):
                #Guarda el punto
                Sx.append(Mean[0])
                Sy.append(Mean[1])
            else:
                #Retorna todas las intersecciones detectadas para escoger manualmente
                print_array(X)
                print_array(Y)
                Plot(X,Y,a,b2,Center,cup,radius)
                select = input("Select a point (n for none): ")
                if  select  in  {"0","1","2","3","4","5"}:
                    #Guarda el punto
                    Sx.append(X[int(select)])
                    Sy.append(Y[int(select)])
        #Pregunta para hacer m s mediciones
        new = input("Continue? [y/n] ")
#Guarda en  position.csv
print("Saving in position.csv...")
Table = np.zeros((len(Sx),3))
Table[:,0] = Sx
Table[:,1] = Sy
Table[0,2] = radius
pd.DataFrame(Table).to_csv("position.csv")
```

Listing 4: Código de Python que detecta la posición del golpe sobre la taza.

El código que se ejecuta en paralelo es el que realiza la transformada rápida de Fourier de los datos de la entrada análoga. Devuelve una gráfica de intensidad contra tiempo e intensidad contra frecuencia y guarda el valor de la frecuencia asociada a cada golpe. Previo a la ejecución, se observa en el serial plotter que el sensor este ajustado a una sensibilidad tal que en auscencia de golpe devuelva un valor de 660. El umbral para el golpe se encuentra en 700, tal que apenas se supere este dato, el código comience a guardar los siguientes valores de amplitud, correspondientes a los frentes de onda del golpe de la taza. Se guardan 50 datos, lo cual es suficiente para obtener las oscilaciones totales de la taza antes de que la amplitud disminuya a valores de ruido. Así mismo se activa una instrucción para obtener el tiempo respectivo de las 50 mediciones. Finalmente, estos datos se guardan en un archivo csv.

```
import time
import serial
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy import genfromtxt
from scipy import fftpack

def Golpe():
    """-----------------------------------------------
    Golpe:
    Se comunica con el arduino y guarda los datos de
    amplitud en funci n del tiempo para realizar la
    transformada r pida de Fourier (FFT) de donde se
    obtiene la frecuencia que corresponde a un m ximo
    en el espacio de frecuencias.
    Se asume que el sensor esta ajustado con una
    sensibilidad  ~660.
    -----------------------------------------------
    Return:
     * Frecuencia de la se al
    -----------------------------------------------"""
    #Comunicaci n con el arduino
    ser = serial.Serial('/dev/cu.usbmodem14201', 9600, timeout=1)
    time.sleep(2)
    #N mero de datos a tomar
    M=50
    #Umbral en la amplitud para el inicio de la toma de datos
    U = 700
    #Espera a que se detecte un golpe
    B = False
    print('Listo para iniciar lectura de datos')
    time.sleep(1)
    for k in range(2000):
        line = ser.readline()
```

```
        line = str(line)
        try:
            #Juzga si los datos superan el umbral
            if int(str(ser.readline())[2:5]) > U:
                B = True
                break
        except:
            pass
    if B == False:
        #Si no se detecta golpe en el tiempo dado se cierra el programa
        print('No se ley  ning n dato')
        exit()
    else:
        print('Inicio')
    #En caso afirmativo, se inicia la lectura datos
    data = []
    start = time.time()
    for i in range(M):
        line = ser.readline()
        line = str(line)
        if line:
            try:
                dat4 = float(line[2:5])
                data.append(dat4)
            except:
                print("ERROR: replace method error")
    ser.close()
    #Tiempo de Lectura
    T  = float(time.time() - start)
    tm = np.linspace(0,T,num=np.shape(data)[0])
    #Tabla de datos
    Table = np.zeros((np.shape(data)[0],2))
    #Guarda tiempos y amplitudes por columnas
    Table[:,0] = tm
    Table[:,1] = data
    #Guardar datos en one.csv
    pd.DataFrame(Table).to_csv("one.csv")
    time.sleep(2)
    one = genfromtxt('one.csv', delimiter=',')
    #Lee los datos guardados
    t1 = one[1:,1]
    y1 = one[1:,2]
    #Realiza la transformada r pida de Fourier para encontrar frecuencia de la se al
    N1 = np.shape(t1)[0]
    T1 = t1[-1]
    Ts1 = T1/N1
    Fs1 = (1)/(Ts1)
    fstep = Fs1 / N1
    fstep2 = 9600 / N1 #La frecuencia de sampleo es 9600 como se inicializa en el Arduino
    f1 = np.linspace(0, (N1-1)*fstep2, N1)
    Y1 = np.abs(np.fft.fft(y1))/N1
    M = int(np.size(np.abs(Y1))/2)
    Y1max = np.argmax(np.abs(Y1)[1:M])
    print('Max-freq:', f1[Y1max+1], 'Hz')
    #Gr ficas en el espacio de tiempo
    plt.subplot(2,1,1)
    plt.plot(t1,y1)
    plt.title('Amplitud vs. Tiempo')
    plt.xlabel('Tiempo [s]')
    plt.ylabel('Amplitud')
    plt.grid()
    #Gr fica en el espacio de frecuencias
    plt.subplot(2,1,2)
    plt.plot(f1,Y1)
    plt.title('Abs(Y) vs. Frecuencia')
    plt.xlabel('Frecuencia [1/s]')
    plt.ylabel('Abs(Y)')
    plt.grid()
    plt.axis([1500,2500,0,50])
    plt.text( 2000, 30 , 'Frecuencia  M xima: {} Hz'.format(np.round(f1[Y1max+1],2)))
```

```
    plt.subplots_adjust(hspace=1)
    plt.show()

    return(f1[Y1max+1])

def Accept():
     """-------------------------------------------------
     Accept:
     Permite al usuario repetir mediciones las veces que
     desee, guardar o no un dato y finalmente guardar en
     un formato csv las frecuencias obtenidas
    -------------------------------------------------"""
    #Frecuencia de la se al
    F = Golpe()
    a = input('Acepta este dato? [y/n] ')
    if a == 'y':
        #Guarda la frecuencia en un array
       f.append(F)
       b = input('Desea tomar mas datos? [y/n] ')
       if b == 'y':
           #Se llama a si misma parqa continuar la toma de datos
           Accept()
       if b == 'n':
           #Guarda en freqs.csv
           pd.DataFrame(f).to_csv("freqs.csv")
           print(f)
           exit()
       else:
        b = input('Entrada no v lida. Desea tomar mas datos? [y/n] ')
    if a == 'n':
      b = input('Desea tomar mas datos? [y/n] ')
      if b == 'y':
          #Se llama a si misma para continuar la toma de datos
           Accept()
      if b == 'n':
          #Guarda en freqs.csv
           pd.DataFrame(f).to_csv("freqs.csv")
           print(f)
           exit()

"""---------------------------
             MAIN
---------------------------"""
#Array donde se guardar n las frecuencias
f = []
#Toma de datos
Accept()
```

Listing 5: Código de Python que calcula la frecuencia más notable del golpe.

El último código lee los datos de los dos archivos csv generados anteriormente, realiza gráfica de la taza y encuentra las posibles posiciones del asa al ubicar el punto con menor frecuencia asociada.

```
import numpy as np
import matplotlib.pyplot as plt

#Importar datos de frequencias
datf = np.genfromtxt('freqs.csv',delimiter = ',')
f= datf[1:,1]
#Importar datos de posici n
data = np.genfromtxt('position.csv',delimiter =',')
#Informaci n de la taza (Coordenadas de centro y radio)
xT = data[1,1]
yT = data[1,2]
rT = data[1,3]
#Golpes
x = data[2:,1]
y = data [2:,2]
#Transformaci n al sistema de la taza
x = x - xT
y = y - yT
```

```
#Clasificaci n de las frecuencias en altas (azules) y bajas (rojas)
r_index = []
b_index = []
for i in range(len(f)):
    if f[i] < 1900:
        r_index.append(i)
    else:
        b_index.append(i)
print(r_index)
print(b_index)
xR = []
yR = []
xB = []
yB = []
for i in r_index:
    xR.append(x[i])
    yR.append(y[i])
for i in b_index:
    xB.append(x[i])
    yB.append(y[i])
"""Dibujo de taza con diferentes frecuencias"""
#Borde de la taza
theta = np.linspace(0,2*np.pi,1000)
xC = rT*np.cos(theta)
yC = rT*np.sin(theta)
plt.plot(xC,yC,'k',lw=0.8)
#Puntos
plt.plot(xR,yR,'r.', markersize = 6)
plt.plot(xB,yB,'b.', markersize = 6)
#Ejes en d nde se puede encontrar el asa
mR = np.array(yR) / np.array(xR)
t = np.linspace(-2-rT,2+rT,1000)

for i in range(len(mR)):
    plt.plot(t,t*mR[i],'k--',lw='0.8')

    if len(xR) == 1:
        plt.plot(t,-t/(mR[i]),'k--',lw='0.8')

plt.axis([-2-rT,2+rT,-2-rT,2+rT])
plt.show()
```

Listing 6: Código de Python que grafica los resultados obtenidos para la ubicación de los golpes y la frecuencia asociada.

IV. Resultados

En la implementación original del problema se utilizo el método de multilaración para intentar obtener el origen de un golpe. Sin embargo un análisis de las incertidumbres y varios muestreos revelaron las limitaciones del método. Se considera una incertidumbre en la posición de los sensores de 0,9 cm (aproximadamente el diámetro del sensor), la incertidumbre en la temperatura es, de acuerdo a [6], 0,5° C y la incertidumbre en las diferencias de tiempo de los sensores es de 16μs; con todo esto en consideración se obtuvo la *Figura* 7. Es de resaltar que se lográ una mayor precisión en la coordenada x de la posición ($\approx$ 3 cm, en contraste a los $\approx$ 10 cm sobre el eje y), razón por la cual se planteó que era mejor incluir la restricción de la boca de la taza y encontrar las intersecciones que realizan las hipérbolas con ella. Se usa por defecto el promedio del ángulo medido desde el centro de la taza, aunque se mantiene la opción de elegir una única de estas intersecciones como origen del golpe haciendo la consideración que alguna de las hipérbolas presenta menos error que las otras.

En la *Figura* 8 aparece el resultado de dos mediciones diferentes de la taza. Bajo la nueva aproximación, ambas mediciones se consideran como válidas y coinciden con la posición del golpe determinada por simple inspección. En la *Figura* 8a, la intersección de hipérbolas coincide con la boca de la taza, sin embargo es un hecho que no es común, siendo más representativa la *Figura* 8b. La taza de mediciones exitosas (se encuentra intercepción y esta coincide con lo observado) es de 64 % sobre una muestra de 50 pruebas.

La obtención de las frecuencias probó ser más consistente. Sobre los puntos azules de la *Figura* 2 se encontró la

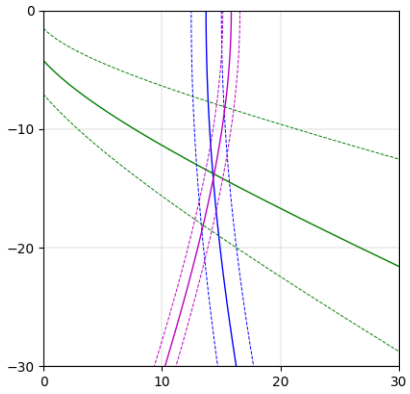
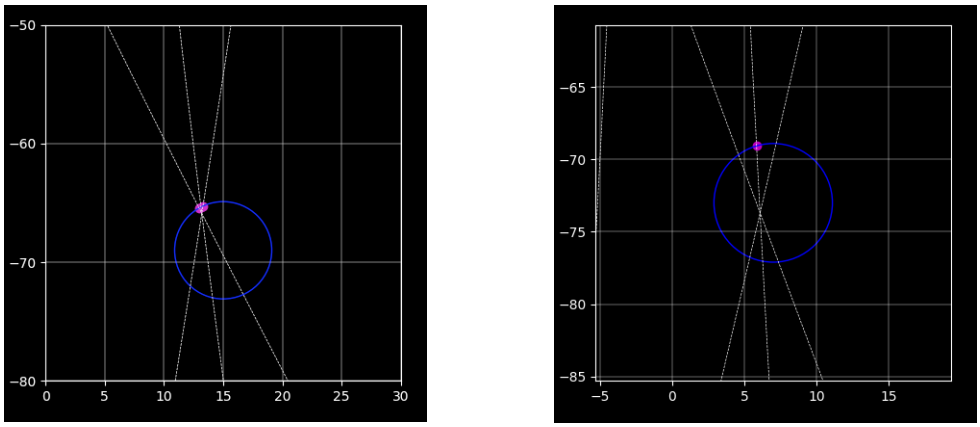


Figura 7: Incertidumbre en el método de multilareación. Las líneas sólidas representan el dato central, las líneas punteadas del mismo color son el límite de incertidumbre para la respectiva hipérbola. El origen se encuentra en la intercepción de las áreas que encierran las líneas punteadas. Las posiciones están expresadas en centímetros y los sensores se encuentran distribuidos sobre el eje x. Se considera que la boca de la taza esta a la misma altura que los sensores.



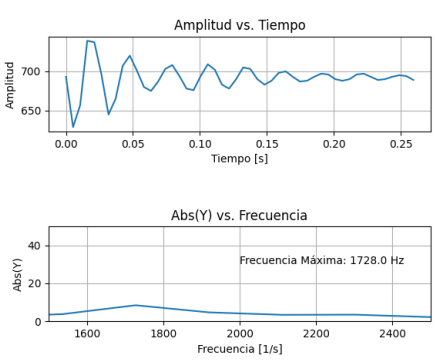
(a) El punto encontrado coincide con el método de multilateración.

(b) La intercepción de hipérbolas ocurre en un punto cercano a la circunferencia de la taza.

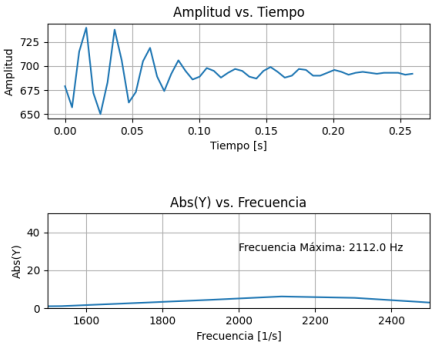
Figura 8: Determinación de la posición a partir del conocimiento de la ubicación de la taza (círculo azul). El punto rosado corresponde a la intercepción promedio. Las línas punteadas representan las hipérbolas. Las distancias están en centímetros.

menor frecuencia propia al rededor de 1700 ± 100 Hz. En los puntos rojos se midió una frecuencia de aproximadamente 2100 ± 100 Hz. La incertidumbre corresponde a la mitad de la distancias entre puntos sucesivos sobre el eje x en el espacio de frecuencias asociada a la frecuencia de muestreo [7] para los 50 datos tomados. En otros puntos se encontraron picos en estas dos frecuencias o curvas más planas en las que encontrar el máximo no lleva a un resultado concluyente. Se dice entonces que estos puntos son en efecto combinación de los otros dos modos. En algunas ocasiones se midieron frecuencias por encima de los 2400 Hz hasta los 4800 Hz, sin embargo estos resultados no fueron reproducibles y solo se hace el comentario. Con vista en estos resultados se considera que la frecuencia de muestreo ($\sim$ 9600 Hz) es apropiada, según el Teorema de Nyquist. La duración de la oscilación fue de alrededor de $0,25$ s, lo cual limitó la toma de datos a solo 50 como ya se menciono.

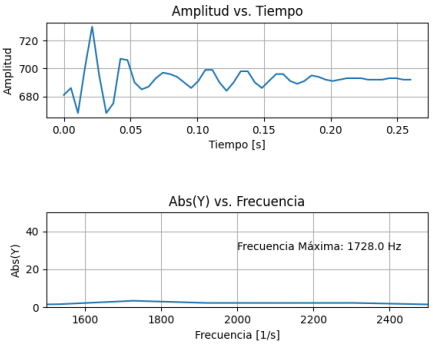
Finalmente, de entre las medidas se selecciona la posición de la mayor frecuencia y se marca entorno a que ejes es posible que se encuentre el asa, con una incertidumbre de 20° (*Figura* 10).



(a) Frecuencia de un golpe a 90° del asa (grave)



(b) Frecuencia de un golpe a 45° del asa (agudo)



(c) Frecuencia del golpe en puntos intermedios. El sonido resultado es una combinación de ambos modos

Figura 9: Transformada de Fourier para golpes producidos en zonas concretas de la taza

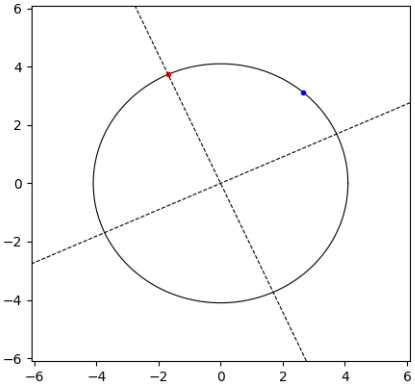


Figura 10: Imagen final de la taza con los posibles ejes sobre los que se puede encontrar el asa. El punto rojo corresponde a una frecuencia aguda y el azul a una grave. Las distancias están en centímetros y el centro de coordenadas coincide con el centro de la taza. El resultado entra de lo esperado con una presición de 20°

A. Dificultades encontradas y sugerencias

Al principio la separación entre sensores fue de 22,5 cm, por prueba y error se encontro que eran necesarios valores mayores, sin embargo al aumentar mucho la distancia es preciso alejar más la taza para seguir garantizando un camino libre entre el punto del golpe y cada sensor; sin embargo, llega un punto en el que la apertura de las curvas de la *Figura* 7 hace que la precisión disminuya, por lo que se debe encontrar un lugar optimo empíricamente antes de comenzar la toma de datos. Si se quiere evitar pasar información de la taza de antemano, sería posible usar otro conjunto de sensores ubicados a 90° de los primeros y que solo se dediquen a encontrar la posición sobre su eje, de este modo también se podría medir toda la circunferencia de la taza, aunque debido a la simetría del problema, esto tampoco es indispensable.

Resultaría apropiado estudiar la sensibilidad del área del micrófono para disminuir la incertidumbre de los focos; aunque más importante, la mayor fuente de error se encuentra en las diferencias de tiempos. Una placa Arduino con más pines que permiten interrupciones talvez hubiera resultado en mejores mediciones. Se reconoce que la implementación por la cual se usan sentencias *if* puede que no sea la más apropiada debido a que existe un retraso mientras se ejecuta esta instrucción pero se desconoce el valor exacto. En un punto, se propuso usar un condensador para convertir la señal oscilante en solo un pico que disminuyera progresivamente pero no pudo ser implementado por cuestiones de tiempo.

En el aspecto de la transformada de Fourier, una taza con una vibración que se prolongue más en el tiempo podría

llevar a resultados más finos donde se distingan más claramente las frecuencias permitiendo analizar la distribución de los modos.

V. Conclusiones

Se obtuvieron dos frecuencias características al golpear la taza asociadas a los picos 1700 ± 100 Hz y 2100 ± 100 Hz para golpes asociados al asa en un antinodo y un nodo respectivamente. La duración de la vibración limitó el análisis de frecuencia solo a los picos. Con una vibración mas prolongada, por ende mas datos, se podría analizar mejor el espectro de frecuencias asociado.

Se mejoraron los resultados para la multilateración con un montaje que permitiera un camino libre entre la fuente y el sensor receptor. Esto limitó el análisis de la taza a solo el hemisferio que daba contra los sensores. La mayor fuente de incertidumbre se obtuvo en las diferencias de tiempo, la cual se vió sobretodo reflejada en la coordenada perpendicular a la línea de los sensores.

[1] Vibration of Circular Rings and Curved Beams. (2019). Vibration of Continuous Systems, 399–425. https://doi.org/10.1002/9781119424284.ch12

[2] Tokieda, T. (2016, 5 septiembre). Coffee Cup Vibrations - Numberphile. YouTube. Recuperado 6 de febrero de 2022, de https://www.youtube.com/watch?v=MfzNJE4CK_s

[3] Gravitational Waves Ringing Teacups. (2018, 22 noviembre). ThatsMaths. Recuperado 6 de febrero de 2022, de https://thatsmaths.com/2018/11/22/ringing-teacups-and-gravitational-waves/

[4] Gan, W. S. (2020). 3 Fourier Transform - Fast Fourier Transform. In Signal Processing and image processing for acoustical imaging. essay, Springer Singapore.

[5] Aalborg University, Dalskov, D. (2014, junio). Locating Acoustic Sources with Multilateration. https://projekter.aau.dk/projekter/files/198526294/main.pdf

[6] A. (2021, 2 diciembre). KY-001 Temperature Sensor Module. ArduinoModulesInfo. Recuperado 5 de febrero de 2022, de https://arduinomodules.info/ky-001-temperature-sensor-module/

[7] Sampling Rate of Arduino Uno Card. (2015, 24 agosto). Arduino Forum. Recuperado 8 de febrero de 2022, de https://forum.arduino.cc/t/sampling-rate-of-arduino-uno-card/331268/4

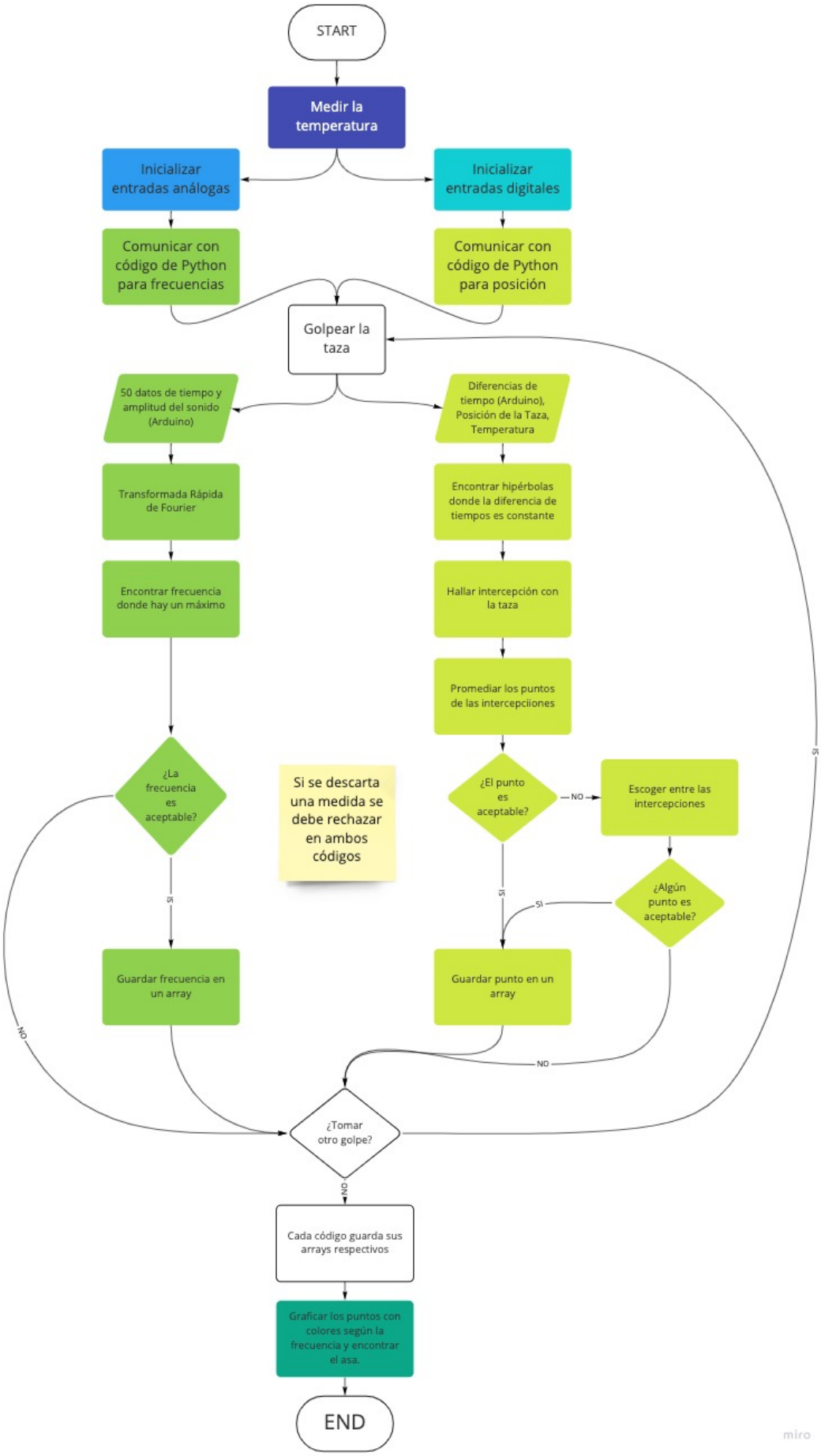


Figura 11: Diagrama de flujo de la toma de datos.