***CODE FOR FFT:***

clc;

clear device;

close all;

% Specify the Excel file name and sheet name

filename = 'ML\_input.xlsx';

sheet = 'Sheet1';

% Create headers for the table

headers = {'Material', 'Frequency', 'Amplitude'};

% Open the Excel file for writing (create it if it doesn't exist)

%xlswrite(filename, headers, sheet, 'A1:C1');

% Initialize a row counter for adding data to the table

row = 2;

device = serialport("com3", 9600);

H = zeros(1000);

t = zeros(1000);

for c=1:1000

data = readline(device); %if your device uses serial print

H(c)=data;

t(c)=c;

end

fr=abs(str2num(readline(device)))

%s=spectrogram(H(1));

fre=ceil(fr);

subplot(2,1,1)

plot(t,H);

y=fft(H);

m = abs(y);

f = (0:length(y)-1)\*100/length(y);

%s=spectrogram(f)

subplot(2,1,2)

plot(f,m);

amp=m(fre)

% Simulate real-time data processing (replace this with your real data source) %while true

% Simulate data acquisition (replace this with your data acquisition code)

material = 'Aluminium'; % Replace with your actual material data

frequency = fr; % Replace with your actual frequency data

amplitude = amp; % Replace with your actual amplitude data

% Create a row of data

rowData = {material, frequency, amplitude};

% Read the data from the Excel sheet as a table

data = readtable(filename, 'Sheet', sheet);

% Specify the column name for which you want to find the last row number

%column\_name = 'Material'; % Replace with the actual column name

% Find the last row number with data in the specified column

%last\_row = find(~isnan(data.(column\_name)), 1, 'last')

%last\_row()

last\_row = size(data, 1)

row=last\_row+2

pos=strcat('A',num2str(row))

% Append the new data to the Excel file

if(~(isnan(amp)))

xlswrite(filename, rowData, sheet, [pos]);

end

% Increment the row counter

%row = row + 1;

% You can introduce a pause to control the data acquisition rate

pause(1); % Adjust the time interval as needed

%end

clear device1;

device1 = serialport("com5", 9600);

pause(30);

write(device1,3,"int8");

data = readline(device1)

pause(30);

write(device1,1,"int8");

data = readline(device1)

pause(20);

***CODE FOR ARDUINO NANO:***

#include "FreqPeriodCounter.h"

#include <SoftwareSerial.h>

SoftwareSerial XBee(2,3);

const byte counterPin = A0;

const byte counterInterrupt = A1; // = d3

FreqPeriodCounter counter(counterPin, micros);

void setup(void)

{ attachInterrupt(counterInterrupt, counterISR, CHANGE);

Serial.begin(9600);

pinMode(A0, INPUT);

XBee.begin(9600);

}

void loop(void)

{

int c=analogRead(A0);

//Serial.println(c);

if(c==0)

{

for(int i=0;i<1000;i++)

{

c=analogRead(A0);

Serial.println(c);

XBee.write(c);

}

digitalWrite(A1,LOW);

int period;

if(counter.ready()) period = counter.period;

Serial.println((float)1/period);

c=abs((float)1/period);

XBee.write(c);

digitalWrite(A1,HIGH);

//delay(1000);

}

}

void counterISR()

{ counter.poll();

}

***CODE FOR NODE MCU:***

#include <SoftwareSerial.h>

//Use this file to store all of the private credentials

//and connection details

int led=BUILTIN\_LED;

#define SECRET\_SSID "ASHU 9158" // replace MySSID with your WiFi network name

#define SECRET\_PASS "747J0(q8" // replace MyPassword with your WiFi password

#define SECRET\_CH\_ID 2275453 // replace 0000000 with your channel number

#define SECRET\_WRITE\_APIKEY "CEJ00W8KMQJ3SARE" // replace XYZ with your channel write API Key

//https://www.electroniclinic.com/

// Download Libraries: https://www.electroniclinic.com/arduino-libraries-download-and-projects-they-are-used-in-project-codes/

#include "ThingSpeak.h"

#include <ESP8266WiFi.h>

char ssid[] = SECRET\_SSID; // your network SSID (name)

char pass[] = SECRET\_PASS; // your network password

int keyIndex = 0; // your network key Index number (needed only for WEP)

WiFiClient client;

unsigned long myChannelNumber = SECRET\_CH\_ID;

const char \* myWriteAPIKey = SECRET\_WRITE\_APIKEY;

// Initialize our values

//int number1 = 0;

// int number1 = random(0,100);

// String myStatus = "";

SoftwareSerial zigBee(12,13);

void setup()

{

Serial.begin(9600);

zigBee.begin(9600);

WiFi.mode(WIFI\_STA);

ThingSpeak.begin(client); // Initialize ThingSpeak

pinMode(led,OUTPUT);

}

double p=0,y=0,t=0;

void loop()

{

// if(zigBee.available()>0){

// for(int i=0;i<1000;i++){

// if(zigBee.available()>0)

// {

// //if(i>3)

// {

// p=zigBee.read();

// Serial.println(p);

// }

// }

// //ThingSpeak.setField(1, p);

// }

digitalWrite(led,HIGH);

int incomingByte=0;

if (Serial.available() > 0) {

// read the incoming byte:

incomingByte = Serial.read();

// ThingSpeak.setField(1, incomingByte);

delay(1000);

digitalWrite(led,LOW);

}

// // int incomingByte=0;

// // if (Serial.available() > 0) {

// // // read the incoming byte:

// // incomingByte = Serial.read();

// // // ThingSpeak.setField(1, incomingByte);

// // // int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

// // // if(x == 200){

// // // Serial.println("Channel update successful.");

// // // }

// // // else{

// // // Serial.println("Problem updating channel. HTTP error code " + String(x));

// // // }

// // digitalWrite(led,LOW);

// // }

// Connect or reconnect to WiFi

if(WiFi.status() != WL\_CONNECTED){

Serial.print("Attempting to connect to SSID: ");

Serial.println(SECRET\_SSID);

while(WiFi.status() != WL\_CONNECTED){

WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP network

Serial.print(".");

delay(5000);

}

Serial.println("\nConnected.");

}

// digitalWrite(led,HIGH);

if(incomingByte!=0)

{

ThingSpeak.setField(1, incomingByte);

int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

if(x == 200){

Serial.println("Channel update successful.");

}

else{

Serial.println("Problem updating channel. HTTP error code " + String(x));

}

}

// set the fields with the values

// send data only when you receive data:

// if (Serial.available() > 0) {

// // read the incoming byte:

// int incomingByte = Serial.read();

// //ThingSpeak.setField(1, incomingByte);

// }

// ThingSpeak.setField(1, incomingByte);

// figure out the status message

// if((number1 > number2) && (number1 > number3)){

// myStatus = String("field1 is set as status");

// }

// else if((number2 > number1) && (number2 > number3)){

// myStatus = String("field2 is set as status");

// }

// else{

// myStatus = String("field3 is set as status");

// }

// // set the status

// ThingSpeak.setStatus(myStatus);

// write to the ThingSpeak channel

// int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

// if(x == 200){

// Serial.println("Channel update successful.");

// }

// else{

// Serial.println("Problem updating channel. HTTP error code " + String(x));

// }

// change the values

// number1++;

// if(number1 > 99){

// number1 = 0;

// }

// number2 = random(0,100);

// number3 = random(0,100);

// number4 = random(0,100);

//delay(20000); // Wait 20 seconds to update the channel again

digitalWrite(led,HIGH);

}

//}