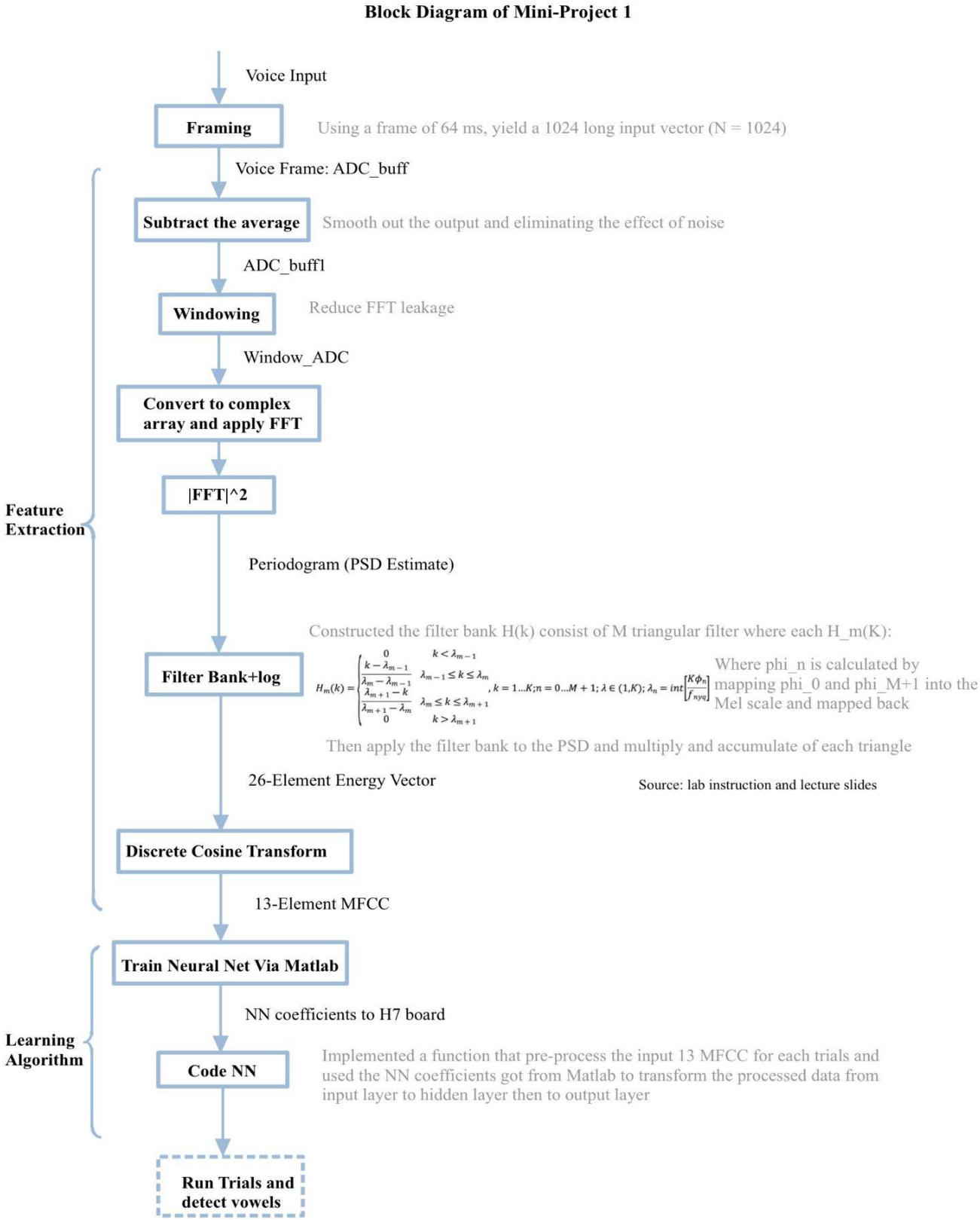
Ruixuan Shen
Hoang Mai Diem Pham
EC ENGR113DA

MINI-PROJECT 1: VOWEL RECOGNITION

1. Introduction

In this project, we built up a vowel recognition system by using a machine learning approach. We extracted the features of a vowel by computing its Mel Frequency Cepstral Coefficients (MFCCs) and built a database with 80 groups of 13-element MFCCs (20 for each vowel.) We then used Matlab to finish the NN training process and retrieve the NN coefficients. And then we implemented the NN function on the H7 board. In the end, the program could successfully recognize and distinguish the four vowels from new speakers whose voices are not part of the training set.

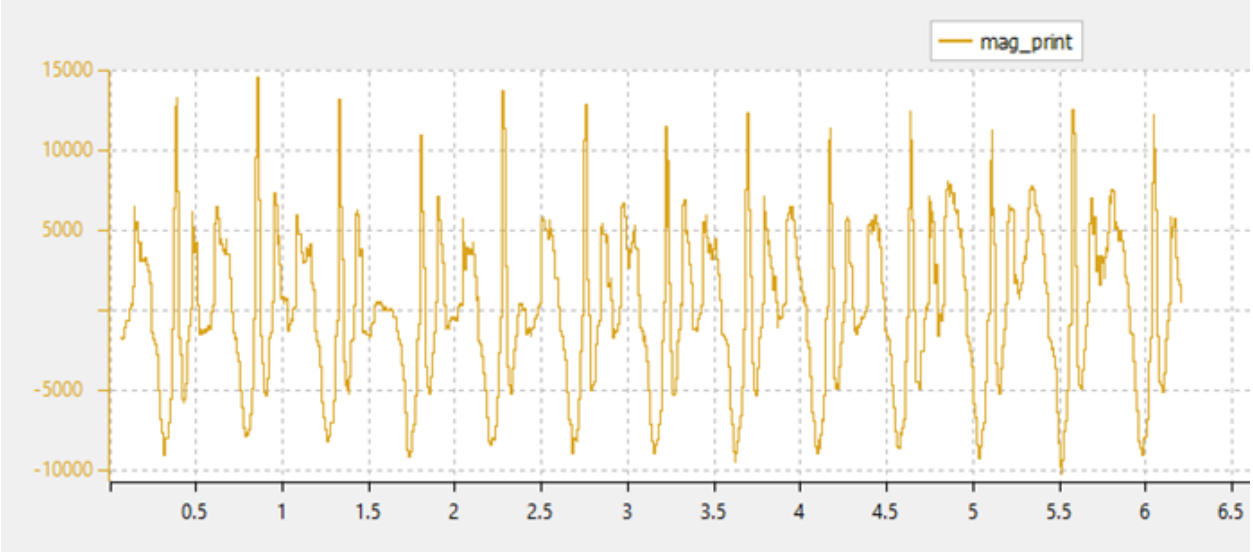
2. Result


Block Diagram of Mini-Project 1
Voice Input
Framing
Using a frame of 64 ms, yield a 1024 long input vector (N = 1024)
Voice Frame: ADC_buff
Subtract the average
Smooth out the output and eliminating the effect of noise
ADC_buff1
Windowing
Reduce FFT leakage
Window_ADC
Convert to complex array and apply FFT
|FFT|^2
Feature Extraction
Periodogram (PSD Estimate)
Constructed the filter bank H(k) consist of M triangular filter where each H_m(K):
Filter Bank+log
$H_m(k) = \begin{cases} 0 & k < \lambda_{m-1} \\ \frac{k-\lambda_{m-1}}{\lambda_m-\lambda_{m-1}} & \lambda_{m-1} \le k \le \lambda_m \\ \frac{\lambda_{m+1}-k}{\lambda_{m+1}-\lambda_m} & \lambda_m \le k \le \lambda_{m+1} \\ 0 & k > \lambda_{m+1} \end{cases}, k = 1 \ldots K; n = 0 \ldots M+1; \lambda \in (1,K); \lambda_n = int\left[\frac{K\phi_n}{f_{nyq}}\right]$
Where phi_n is calculated by mapping phi_0 and phi_M+1 into the Mel scale and mapped back
Then apply the filter bank to the PSD and multiply and accumulate of each triangle
26-Element Energy Vector
Source: lab instruction and lecture slides
Discrete Cosine Transform
13-Element MFCC
Train Neural Net Via Matlab
NN coefficients to H7 board
Learning Algorithm
Code NN
Implemented a function that pre-process the input 13 MFCC for each trials and used the NN coefficients got from Matlab to transform the processed data from input layer to hidden layer then to output layer
Run Trials and detect vowels

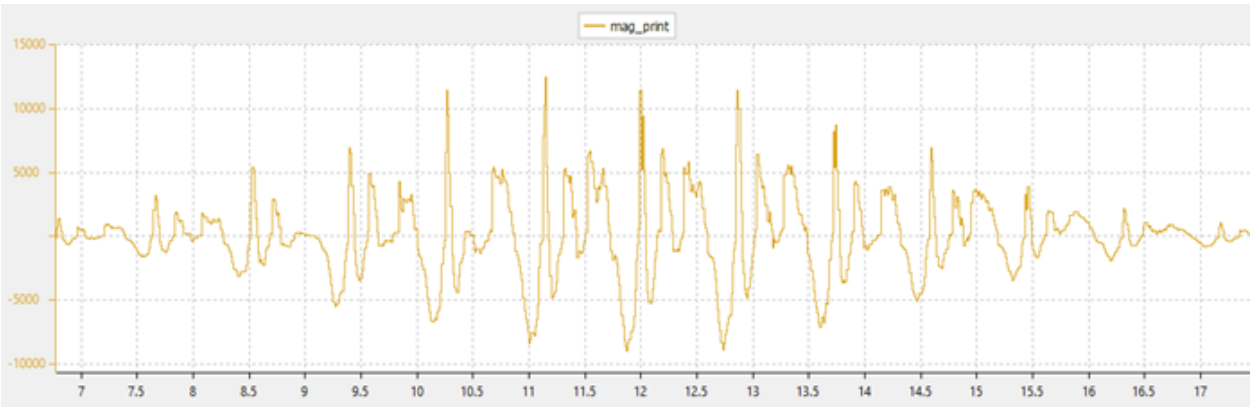
Plots and tables for each monophthong:

a) “ah”

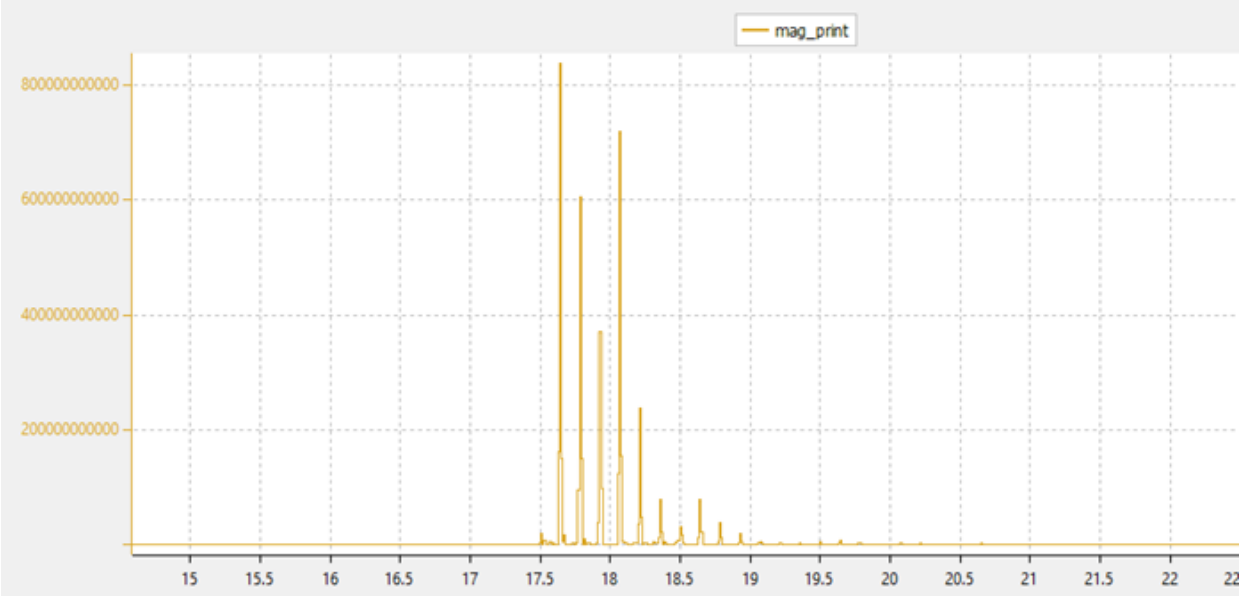
<u>After eliminating average:</u>



<u>After windowing:</u>



FFT magnitude 513 points:



26-long log-energy X vectors:

1st vector:

X	float [26]	0x24010a7c <X>
(x)= X[0]	float	11.1362762
(x)= X[1]	float	11.8662262
(x)= X[2]	float	10.802062
(x)= X[3]	float	11.6628113
(x)= X[4]	float	11.5513754
(x)= X[5]	float	11.8237152
(x)= X[6]	float	11.4395275
(x)= X[7]	float	11.1142168
(x)= X[8]	float	10.9604225
(x)= X[9]	float	10.8151703
(x)= X[10]	float	11.0386229
(x)= X[11]	float	10.7985258
(x)= X[12]	float	10.5175514
(x)= X[13]	float	10.1966906
(x)= X[14]	float	10.0961351
(x)= X[15]	float	10.3139753
(x)= X[16]	float	10.305912
(x)= X[17]	float	10.0344429
(x)= X[18]	float	10.0972853
(x)= X[19]	float	10.0979662
(x)= X[20]	float	10.1643877
(x)= X[21]	float	9.92140675
(x)= X[22]	float	9.93499565
(x)= X[23]	float	9.90165806
(x)= X[24]	float	10.0969448
(x)= X[25]	float	10.1554241

2nd vector:

X	float [26]	0x24010a7c <X>
(x)= X[0]	float	10.8683243
(x)= X[1]	float	11.8893518
(x)= X[2]	float	10.3435907
(x)= X[3]	float	10.8054876
(x)= X[4]	float	10.8130274
(x)= X[5]	float	11.3633289
(x)= X[6]	float	10.4682178
(x)= X[7]	float	11.2530098
(x)= X[8]	float	11.6203995
(x)= X[9]	float	10.4200668
(x)= X[10]	float	10.2649508
(x)= X[11]	float	10.0767031
(x)= X[12]	float	10.03654
(x)= X[13]	float	9.97338295
(x)= X[14]	float	10.1405764
(x)= X[15]	float	10.3748379
(x)= X[16]	float	10.1468506
(x)= X[17]	float	10.0475445
(x)= X[18]	float	10.2731228
(x)= X[19]	float	10.1121807
(x)= X[20]	float	10.0106859
(x)= X[21]	float	10.0761051
(x)= X[22]	float	9.9183712
(x)= X[23]	float	10.0513153
(x)= X[24]	float	10.126749
(x)= X[25]	float	10.0827303
Add new expression		

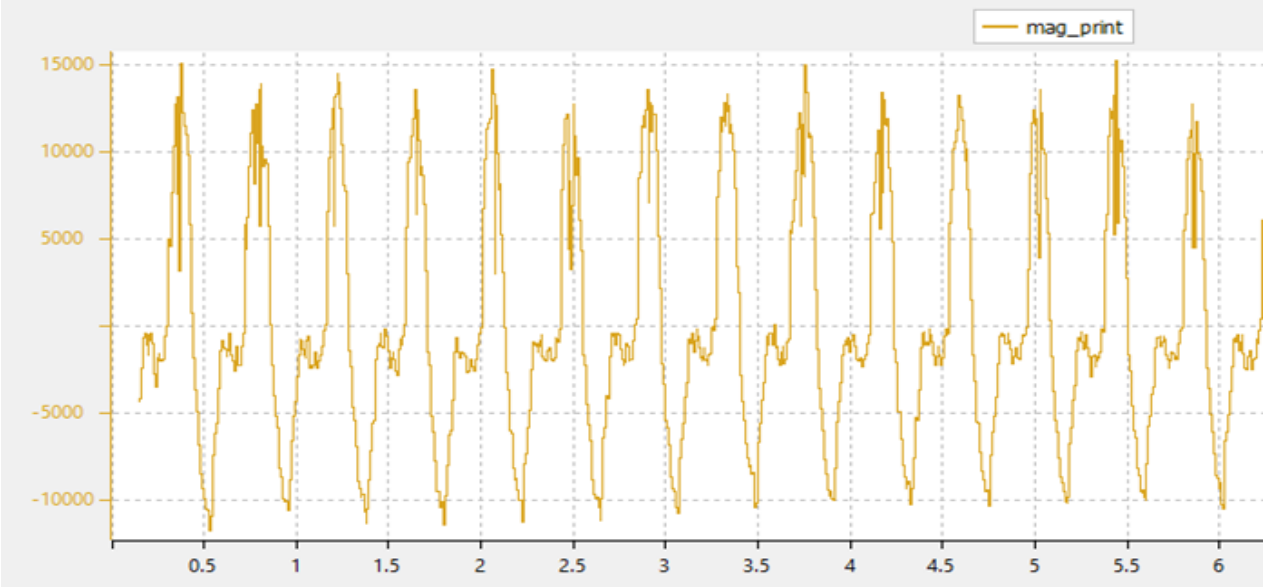
13-long Mel-frequency vectors:

12.084951 1.953838 0.205900 -0.090636 -1.857273 -2.771704 -1.091512 1.961280 0.348160 0.329331 -0.818015 -1.532363 -0.120790

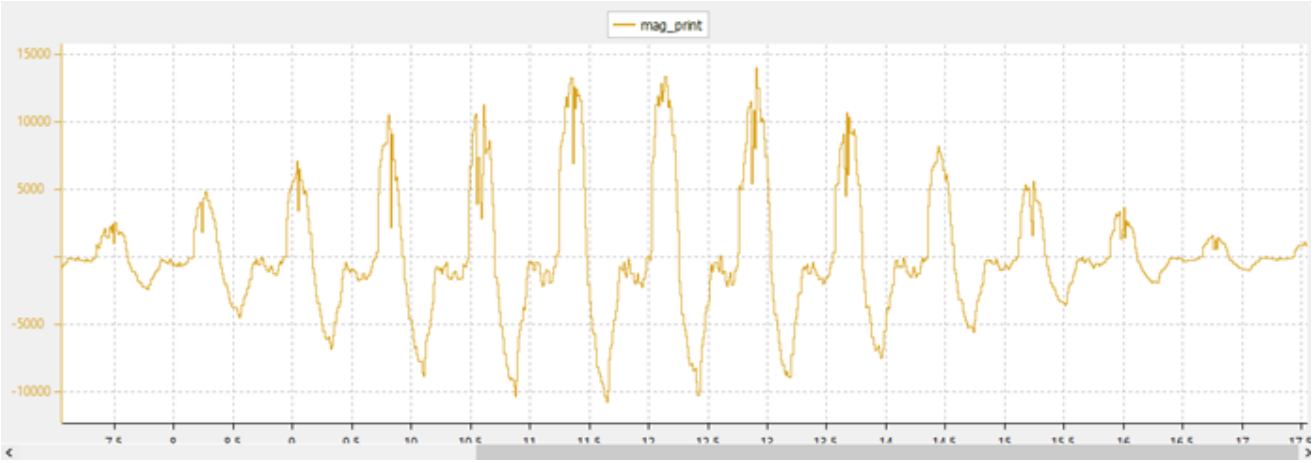
10.494513 2.405301 -0.268496 -0.617088 -2.258059 -2.568399 -1.223852 0.676256 2.165411 0.753365 -1.919819 -1.225553 -0.106328

b) “eh”

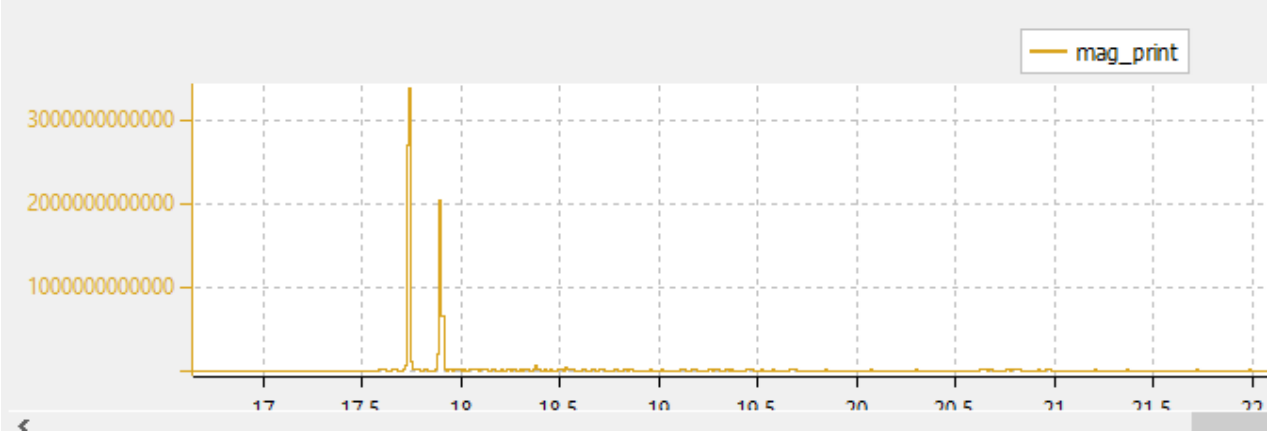
After eliminating average:



After windowing:



FFT magnitude 513 points:



26-long log-energy X vectors:

1st vector:

(x)= record_done	int	0
X	float [26]	0x24010a7c <X>
(x)= X[0]	float	10.0348606
(x)= X[1]	float	12.2603712
(x)= X[2]	float	12.0350828
(x)= X[3]	float	10.1021643
(x)= X[4]	float	10.2837801
(x)= X[5]	float	10.0384493
(x)= X[6]	float	10.108264
(x)= X[7]	float	10.9889526
(x)= X[8]	float	10.6184349
(x)= X[9]	float	10.475996
(x)= X[10]	float	10.1115942
(x)= X[11]	float	9.95864296
(x)= X[12]	float	9.99764347
(x)= X[13]	float	10.3877554
(x)= X[14]	float	10.1078558
(x)= X[15]	float	9.99976921
(x)= X[16]	float	9.96172047
(x)= X[17]	float	9.76507282
(x)= X[18]	float	9.7900486
(x)= X[19]	float	10.230628
(x)= X[20]	float	10.2665834
(x)= X[21]	float	10.0118322
(x)= X[22]	float	10.0487328
(x)= X[23]	float	10.1009817
(x)= X[24]	float	10.1285677
(x)= X[25]	float	10.1883898

2nd vector:

X	float [26]	0x24010a7c <X>
(x)= X[0]	float	9.71731949
(x)= X[1]	float	12.1357527
(x)= X[2]	float	12.0836983
(x)= X[3]	float	10.3918705
(x)= X[4]	float	10.6300745
(x)= X[5]	float	10.0751667
(x)= X[6]	float	10.0440779
(x)= X[7]	float	10.883316
(x)= X[8]	float	10.4062691
(x)= X[9]	float	10.1550455
(x)= X[10]	float	10.3041277
(x)= X[11]	float	9.94652271
(x)= X[12]	float	10.334693
(x)= X[13]	float	10.1176825
(x)= X[14]	float	10.0396643
(x)= X[15]	float	10.1777468
(x)= X[16]	float	10.2932768
(x)= X[17]	float	10.2217607
(x)= X[18]	float	10.0501747
(x)= X[19]	float	10.1945591
(x)= X[20]	float	10.1737337
(x)= X[21]	float	10.1468391
(x)= X[22]	float	10.2916622
(x)= X[23]	float	10.2954607
(x)= X[24]	float	10.2596292
(x)= X[25]	float	10.1953239

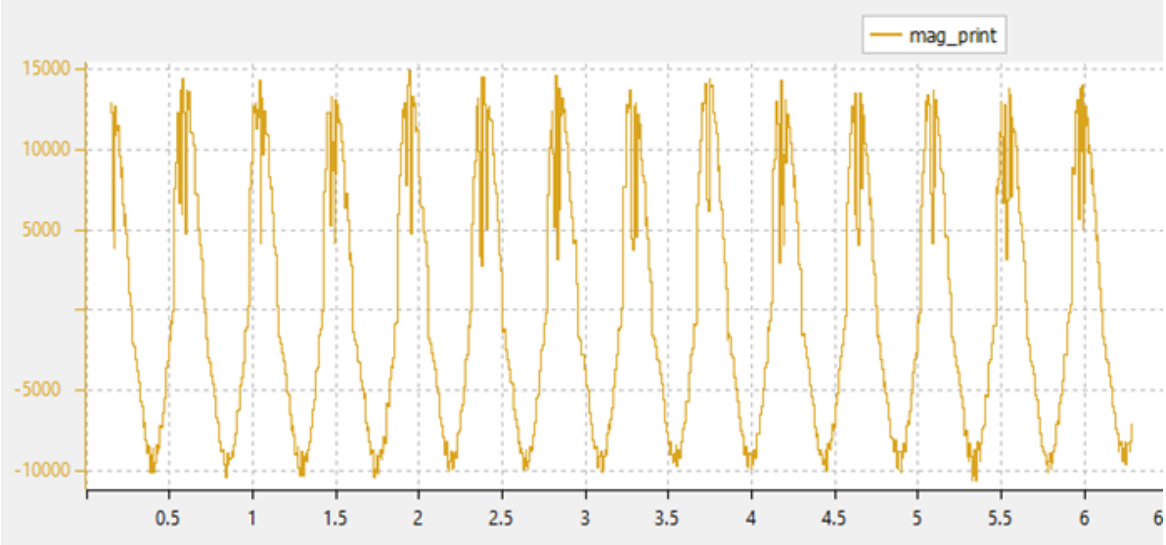
13-long Mel-frequency vectors:

7.653903 2.992535 4.579241 2.529202 -1.676808 -1.034887 2.399608 -0.851374 -4.819238 -3.081451 -2.955086 -1.204214 0.483910

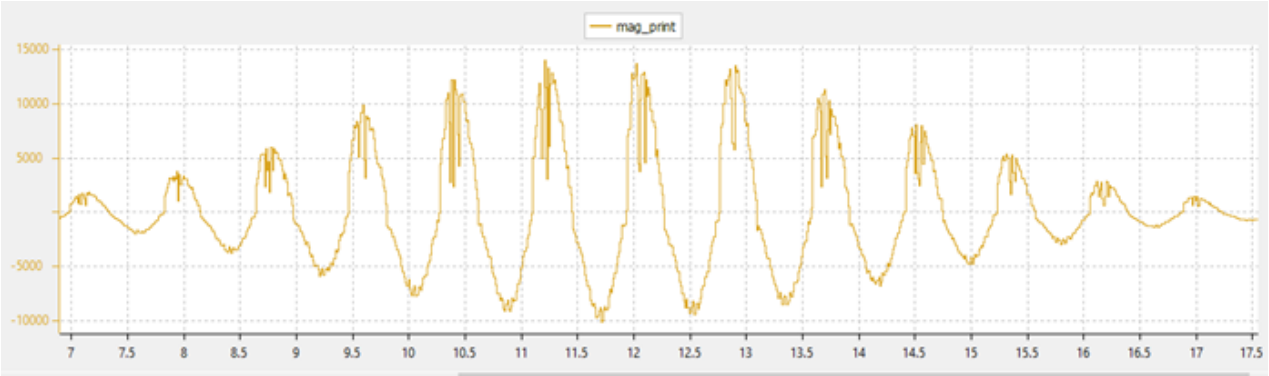
7.002313 3.344129 4.867488 1.606278 -1.769286 -0.844343 1.303077 -1.861995 -4.694458 -1.782274 -2.460690 -1.367033 1.305474

c) “ee”

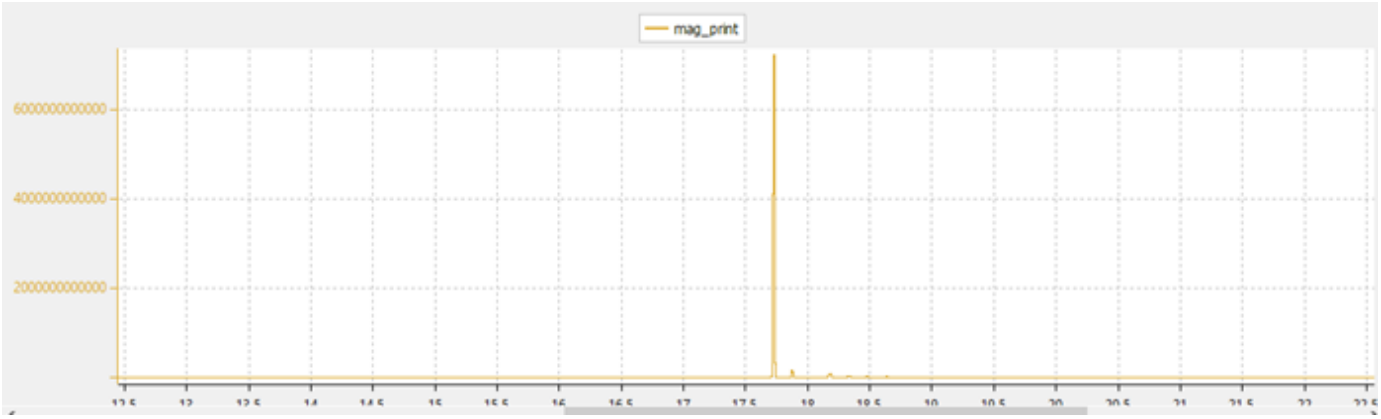
<u>After eliminating average:</u>



<u>After windowing:</u>



FFT magnitude 513 points:



26-long log-energy X vectors:

1st vector:

X	float [26]	0x24010a7c <X>
(x)= X[0]	float	9.8179493
(x)= X[1]	float	11.4106712
(x)= X[2]	float	10.4045162
(x)= X[3]	float	10.1061821
(x)= X[4]	float	10.0243807
(x)= X[5]	float	11.0823727
(x)= X[6]	float	10.5458517
(x)= X[7]	float	10.6198111
(x)= X[8]	float	10.7489004
(x)= X[9]	float	10.4645414
(x)= X[10]	float	10.1542788
(x)= X[11]	float	10.0138435
(x)= X[12]	float	10.050952
(x)= X[13]	float	10.2470989
(x)= X[14]	float	10.3602285
(x)= X[15]	float	10.4100513
(x)= X[16]	float	10.4290524
(x)= X[17]	float	10.4730253
(x)= X[18]	float	10.6248312
(x)= X[19]	float	10.6317787
(x)= X[20]	float	10.3229694
(x)= X[21]	float	10.2646656
(x)= X[22]	float	10.2692499
(x)= X[23]	float	10.3799934
(x)= X[24]	float	10.5402832
(x)= X[25]	float	10.5570831

2nd vector:

X	float [26]	0x24010a7c <X>
(x)= X[0]	float	10.1543264
(x)= X[1]	float	11.0115871
(x)= X[2]	float	9.74415398
(x)= X[3]	float	9.77688313
(x)= X[4]	float	9.84250355
(x)= X[5]	float	10.354599
(x)= X[6]	float	9.8753252
(x)= X[7]	float	9.64628029
(x)= X[8]	float	9.60533237
(x)= X[9]	float	9.80303192
(x)= X[10]	float	9.43195534
(x)= X[11]	float	9.45009804
(x)= X[12]	float	9.57917309
(x)= X[13]	float	9.47668648
(x)= X[14]	float	9.8150301
(x)= X[15]	float	9.60425758
(x)= X[16]	float	10.04181
(x)= X[17]	float	10.2536545
(x)= X[18]	float	10.1598272
(x)= X[19]	float	10.1048279
(x)= X[20]	float	9.7051878
(x)= X[21]	float	9.66534996
(x)= X[22]	float	9.51709843
(x)= X[23]	float	9.59130955
(x)= X[24]	float	9.46230221
(x)= X[25]	float	9.48024273

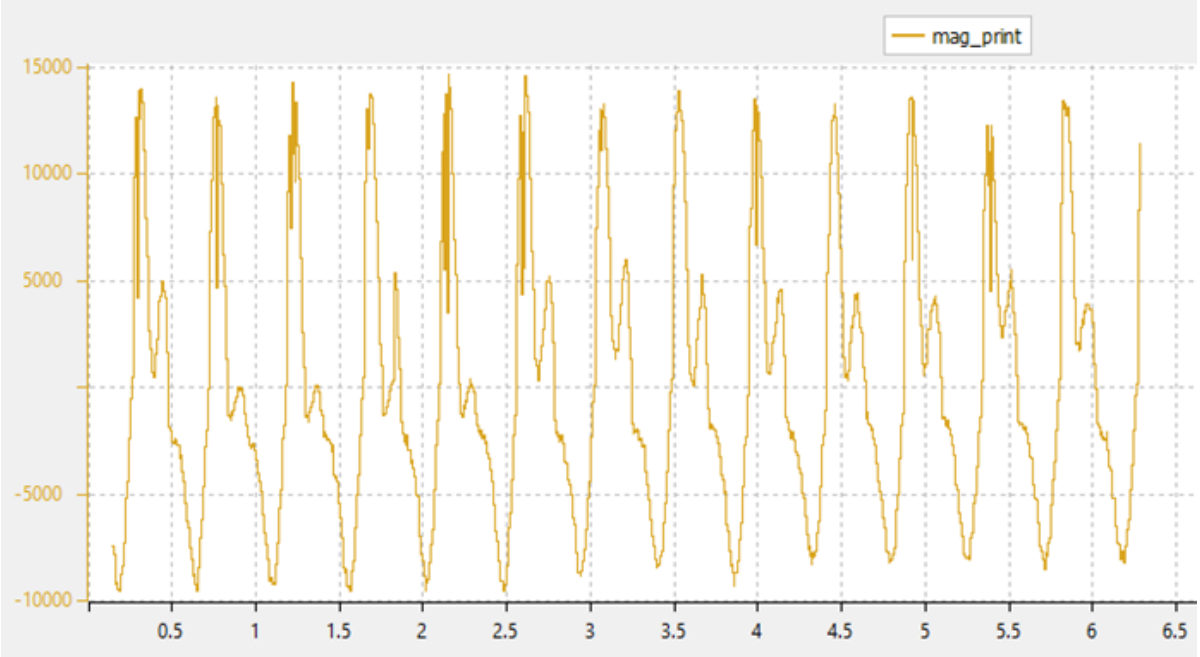
13-long Mel-frequency vectors:

2.776335 1.349880 5.513676 0.091480 -1.016242 1.153703 1.792039 2.416807 -0.072774 0.511730 1.999814 -1.843608 -0.285313

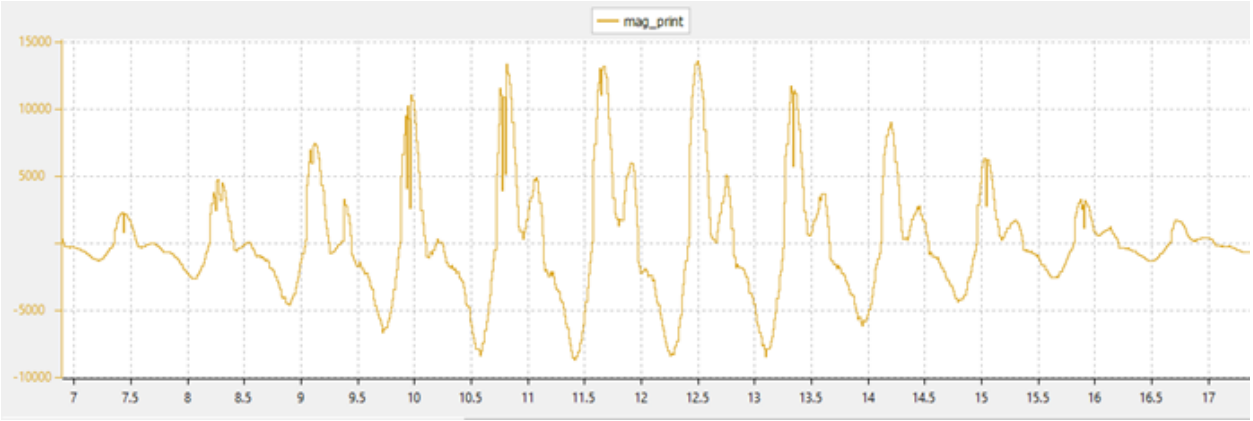
2.406469 1.774421 2.159808 0.372122 1.190490 1.393529 1.036618 2.218234 -0.469569 0.167986 0.044966 -2.123248 -1.836513

d) “oo”

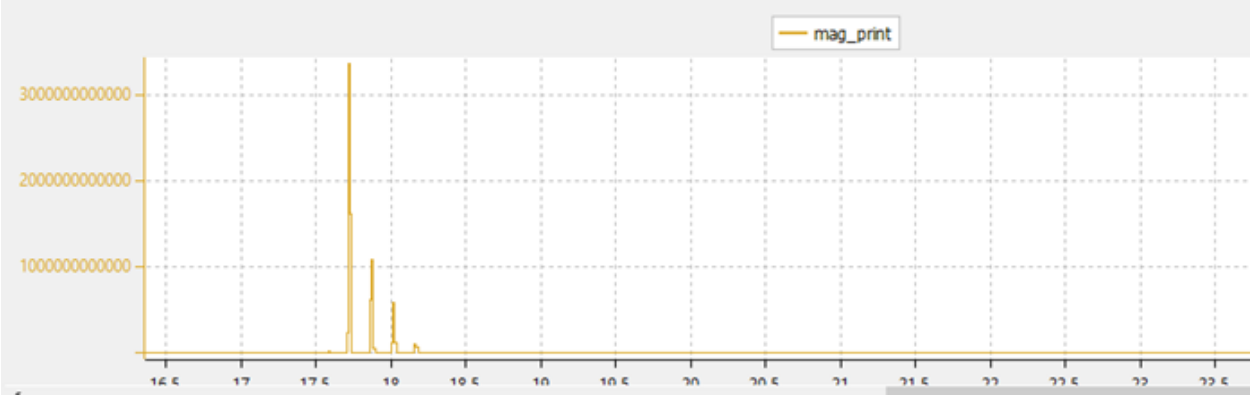
After eliminating average:



After windowing:



FFT magnitude 513 points:



26-long log-energy X vectors:

1st vector:

X	float [26]	0x24010a7c <X>
(x)= X[0]	float	11.0139408
(x)= X[1]	float	12.2121353
(x)= X[2]	float	10.7498407
(x)= X[3]	float	11.887145
(x)= X[4]	float	10.9248209
(x)= X[5]	float	11.1999435
(x)= X[6]	float	9.69194508
(x)= X[7]	float	9.94976044
(x)= X[8]	float	10.1462517
(x)= X[9]	float	10.3220768
(x)= X[10]	float	10.4107103
(x)= X[11]	float	10.1499062
(x)= X[12]	float	10.1980877
(x)= X[13]	float	9.96260929
(x)= X[14]	float	9.80613422
(x)= X[15]	float	9.60038662
(x)= X[16]	float	9.54794979
(x)= X[17]	float	9.85412025
(x)= X[18]	float	9.89289761
(x)= X[19]	float	9.84784508
(x)= X[20]	float	9.83824348
(x)= X[21]	float	9.93041039
(x)= X[22]	float	10.159852
(x)= X[23]	float	10.2017145
(x)= X[24]	float	10.0821295
(x)= X[25]	float	10.0767317

2nd vector:

X	float [26]	0x24010a7c <X>
(x)= X[0]	float	12.0345497
(x)= X[1]	float	12.1661825
(x)= X[2]	float	10.7926865
(x)= X[3]	float	10.9135675
(x)= X[4]	float	10.8382492
(x)= X[5]	float	10.5348949
(x)= X[6]	float	10.0822506
(x)= X[7]	float	10.092844
(x)= X[8]	float	10.3425179
(x)= X[9]	float	10.4585419
(x)= X[10]	float	10.415061
(x)= X[11]	float	10.4121943
(x)= X[12]	float	10.4704189
(x)= X[13]	float	10.325902
(x)= X[14]	float	10.2687168
(x)= X[15]	float	10.2832375
(x)= X[16]	float	10.1735744
(x)= X[17]	float	10.2463379
(x)= X[18]	float	10.0821514
(x)= X[19]	float	10.0745773
(x)= X[20]	float	10.1024027
(x)= X[21]	float	10.1415653
(x)= X[22]	float	10.1489534
(x)= X[23]	float	10.2030191
(x)= X[24]	float	10.3371286
(x)= X[25]	float	10.3401403
Add new expression		

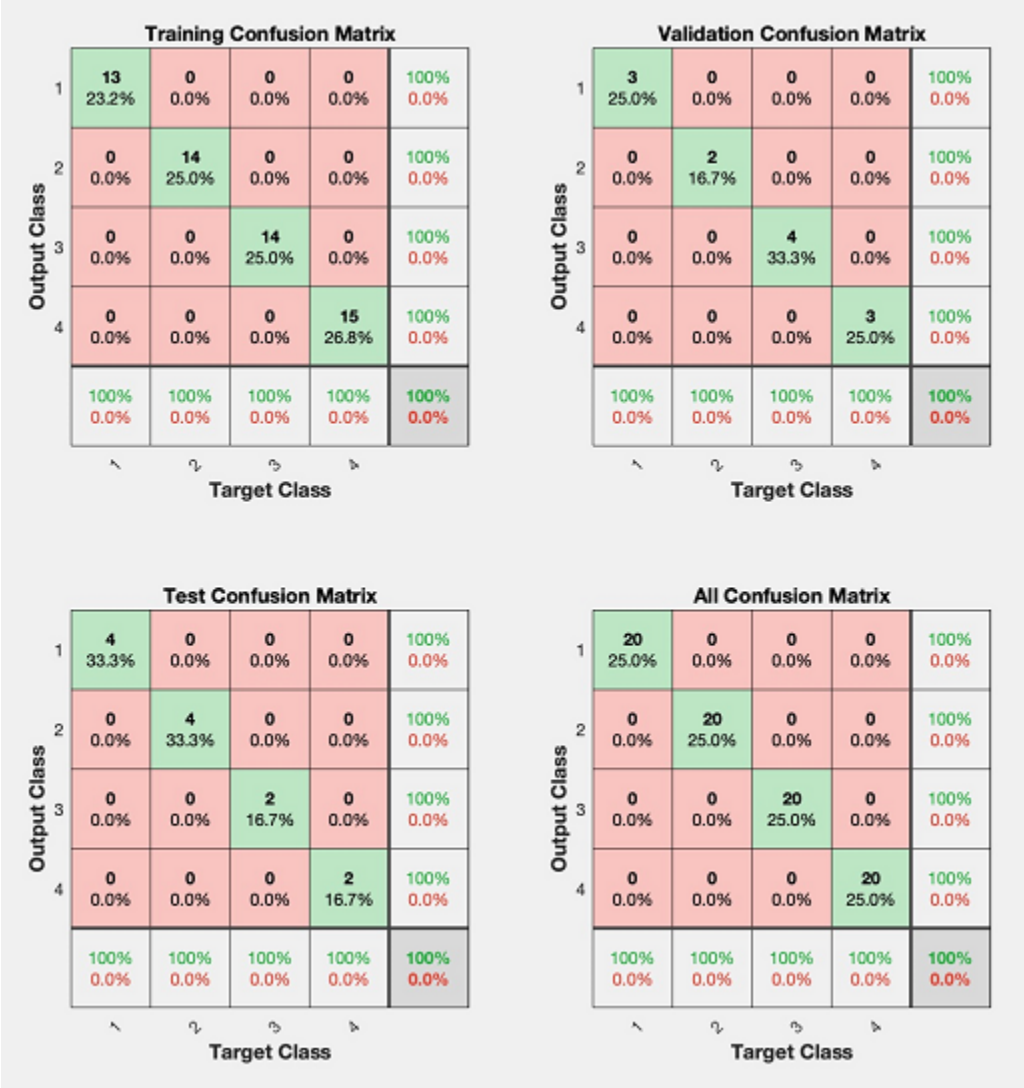
<u>13-long Mel-frequency vectors:</u>

6.134793 1.912525 1.019914 1.954047 2.509468 0.102138 -0.880429 0.232889 1.141834 0.258273 0.712141 0.146363 -0.394748

4.796525 2.012552 1.259746 2.523209 1.257365 0.856676 -1.066934 0.610541 0.820730 0.434272 0.990111 0.448948 -0.342150

3. Discussion

The performance of the NN:

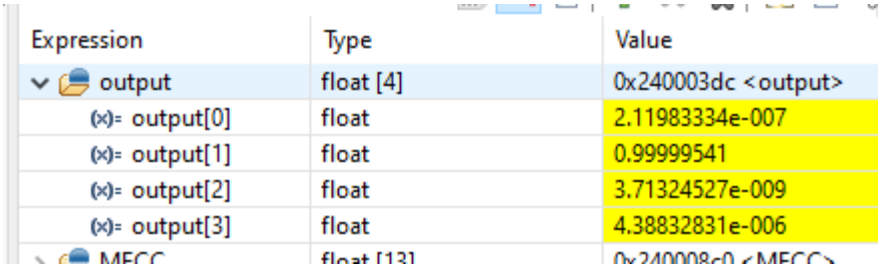


New voice accuracy, from a voice not used in the training set:

“ah”:

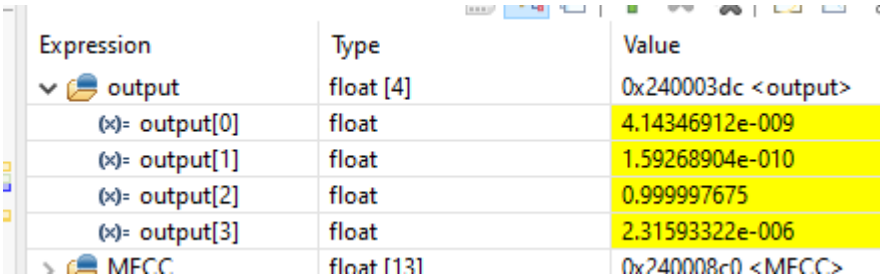
Expression	Type	Value
output	float [4]	0x240003dc <output>
(x)= output[0]	float	0.999959648
(x)= output[1]	float	1.11870882e-007
(x)= output[2]	float	3.95842294e-013
(x)= output[3]	float	4.01797151e-005

“eh”:



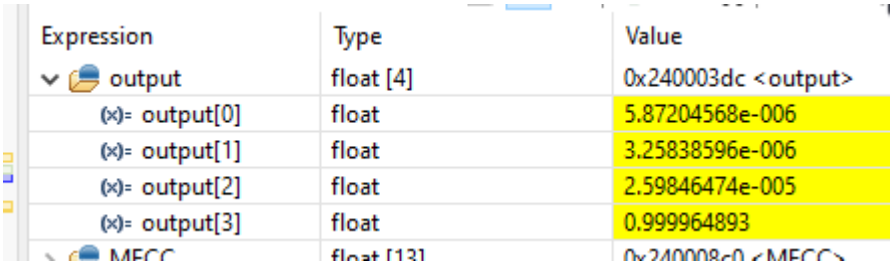
Expression	Type	Value
output	float [4]	0x240003dc <output>
output[0]	float	2.11983334e-007
output[1]	float	0.99999541
output[2]	float	3.71324527e-009
output[3]	float	4.38832831e-006
MFCC	float [13]	0x240008c0 <MFCC>

“ee”:



Expression	Type	Value
output	float [4]	0x240003dc <output>
output[0]	float	4.14346912e-009
output[1]	float	1.59268904e-010
output[2]	float	0.999997675
output[3]	float	2.31593322e-006
MFCC	float [13]	0x240008c0 <MFCC>

“oo”:



Expression	Type	Value
output	float [4]	0x240003dc <output>
output[0]	float	5.87204568e-006
output[1]	float	3.25838596e-006
output[2]	float	2.59846474e-005
output[3]	float	0.999964893
MFCC	float [13]	0x240008c0 <MFCC>

Challenges:

When we first collected the learning sets with more than 3 people's voice (both female and male voice), the MATLAB seemed to be confused and could not reach 100% for the confusion matrix even with multiple times of retrains. The implementations of NN on the H7 were all failed (cannot recognize different vowels). Later we found out that a better approach to solve this problem is to start with one person's voice, check if the NN implementation from a single person's voice work. Then build up more people's voice in the training set based on that.

At the beginning when we tried to design our code so that we do not need to rebuild and redownload to the H7 after each test, we made a mistake. Since we forgot to reset a variable to calculate the sum, the program used the old values of the previous test and summed them up with new values of the new test. Then the program only recognized the monophthong correctly in the first turn. After we fix this mistake, the program can recognize all vowels successfully in all turns.

4. Code

4.1/ Global variables:

```
/* USER CODE BEGIN 0 */
#include <arm_math.h>
#include <stdio.h>
#include <stdlib.h>
#define ARM_MATH_CM7
#define __FPU_PRESENT 1

#define ADC_BUF_SIZE 1024

uint16_t ADC_buff[ADC_BUF_SIZE];
int record_done=0;
float mag_print;
float window[ADC_BUF_SIZE]; //Hamming window
float window_ADC[ADC_BUF_SIZE]; //after apply Hamming window to the input
float fft_square_mag[ADC_BUF_SIZE];
float phi_0 = 250.0; float phi_27 = 8000.0;
int M = 26;
int Z = 13;
float BETA = 0.46;
float ALPHA = 0.54;
float fmel[28];
float phi[28];
float lamda[28];
float H[28][513];
float energy_output[26];
float X[26]; // log output energy
float MFCC[13];
float ADC_buff1[ADC_BUF_SIZE];
float output[4];
```

4.2/ Function implement Neural Network

```
void NN_Test(){

    float x1_offset[13] = {0.168975,-0.483707,-4.723607,-7.080431,-5.415598,-5.550424,-4.647233,-3.029971,-6.161021,-4.6281,-4.196004,-3.845508,-
    float x1_gain[13] = {0.161912916918677,0.251867693396923,0.176197140637562,0.161440787998944,0.218752061054575,0.198516604524352,0.2838151165
    float x1_ymin = -1;

    float b1[5] = {0.050520948190351855356,0.70348200754524303768,-1.6159629461489435354,-0.268716337854188414,1.2246177545239944617};
    float IW1_1[5][13]= {{1.3845663377484549 45, -0.44985171672673329724, -0.27630685532942456106, 0.6222110370845526095, -0.59520115878139723264,
           {-0.22230587226674355938, -0.42836039552848043099, 1.40990844402417137, -0.23094381370271427345, -1.0758533635613003465, -1.276055770
           {2.2795905194465828636, -0.95058944914825882488, -2.2246429981402040532, -2.2015534912645584598, -1.068282131021232928, -1.4036945083
           {-2.5925877835402535432, 0.34517160861479179168, 1.1425911459977200479, -0.34783966107991698413, 1.4957963255827930737, 1.73852821261
           {1.9456528174172766921, -0.3596612709403338437, -3.53838922097893116, 4.9160359417693220152, -0.57201813140127588664, -5.151834975416
                  };

    float b2[4] = {1.224118198426011439,0.26262828789044301292,0.049748511555626914737,-1.2267498018903089108};
    float LW2_1[4][5] = {{0.96250487394378159145, -1.6572097177601095019, 5.3380275130970469277, -3.1656740604425834817, 1.1655241956577135909},
           {-0.37822411489095258963, 5.8786383096330387943, -2.0091087053128626749, -2.5244051761217538576, 2.5854408253040741528},
           {-1.4834105324697453021, -0.3091029000151264694, -3.0914900672369651 78, 2.6854292909987456106, -8.8054842784664177913},
           {-0.65771757901938732171, -3.5592461400504693536, -2.1867378906098764446, 0.94507864444246947322, 4.317920638129885802}
                  };

    // Data pre-processing
    float Y[13];
    for (int i=0; i<13; i++){
     Y[i] = MFCC[i] - x1_offset[i];
     Y[i] = Y[i] * x1_gain[i];
     Y[i] = Y[i] + x1_ymin;
    }

    // Input to hidden
    float h[5] = {0.0};
    for (int x=0; x<5; x++){
        float sum1 = 0.0;
        for (int k=0; k<13; k++){
            sum1 = sum1 + IW1_1[x][k]*Y[k];
        }
        h[x] = 2.0/(1.0+exp(-2*(sum1+b1[x]))) -1.0;
    }

    //hidden to output
    float o[4] = {0.0};
    for (int x=0; x<4; x++){
        float sum2 = 0.0;
        for (int k=0; k<5; k++){
            sum2 = sum2 + LW2_1[x][k]*h[k];
        }
        o[x] = sum2+b2[x];
    }
    float sum3 = 0.0;
    for (int x=0; x<4; x++){
        sum3 = sum3 + exp(o[x]);
    }
    for (int x=0; x<4; x++){
     output[x] = exp(o[x])/sum3;
    }
}
```

4.3/ Main()

```
222   /* USER CODE BEGIN 2 */
223   if(HAL_ADCEx_Calibration_Start(&hadc1, ADC_CALIB_OFFSET,ADC_SINGLE_ENDED) != HAL_OK){
224       Error_Handler();
225   }
226
227   // start timer TIM1
228   if (HAL_TIM_Base_Start(&htim1) != HAL_OK){
229       Error_Handler();
230   }
231
232   //FFT handler initialization
233   arm_cfft_instance_f32 fft_handler;
234   arm_cfft_init_f32(&fft_handler, ADC_BUF_SIZE);
235
236   /* USER CODE END 2 */
237
238   /* Infinite loop */
239   /* USER CODE BEGIN WHILE */
240   while (1)
241   {
242     /* USER CODE END WHILE */
243
244     /* USER CODE BEGIN 3 */
245
246     //HAL_Delay(1000);
247     // start sampling
248     if (HAL_ADC_Start_DMA(&hadc1, (uint32_t *)&ADC_buff, ADC_BUF_SIZE) != HAL_OK) {
249       Error_Handler();
250     }
251
```

```
251
252     while(1){
253         if(record_done == 1){break;}
254     }
255
256     float sum = 0.0;
257     for (int i=0; i < ADC_BUF_SIZE; i++) {
258         sum = sum + ADC_buff[i];
259     }
260     float avg = sum/ ADC_BUF_SIZE;
261
262     // eliminate average  and store in a float array
263     for (int i=0; i < ADC_BUF_SIZE; i++) {
264         ADC_buff1[i] = ADC_buff[i] - avg;
265     }
266
267     // Print out the signal after eliminate average
268     for(int i = 0; i < ADC_BUF_SIZE; i++){
269          mag_print = ADC_buff1[i];
270          HAL_Delay(5);
271     }
272
273     // window the signal
274     for (int i=0; i<ADC_BUF_SIZE; i++){
275         window[i] = ALPHA  - BETA*cos(2.0*PI*i/(ADC_BUF_SIZE-1));
276         window_ADC[i] = ADC_buff1[i] * window[i];
277     }
278
```

```
278
279     // Print out the signal after applying window
280    for (int i=0; i < ADC_BUF_SIZE; i++) {
281             mag_print = window_ADC[i];
282             HAL_Delay(10);
283     }
284
285     // Convert to complex array
286     float* complex_input = (float*)malloc(ADC_BUF_SIZE*2*sizeof(float));
287     for(int i = 0; i< ADC_BUF_SIZE; i++){
288      complex_input[2*i] = window_ADC[i];
289      complex_input[2*i + 1] = 0;
290     }
291
292     //Perform FFT
293     //Take K = N/2 +1 points = 513 points
294     arm_cfft_f32(&fft_handler, complex_input, 0, 1); //output is overwritten to complex_input
295     for (int i=0; i < 513; i++){
296         fft_square_mag[i] = (complex_input[2*i])*(complex_input[2*i]) + (complex_input[2*i+1])*(complex_input[2*i+1]);
297     }
298
299     // Print out the FFT square magnitude
300     for (int i=0; i < 513; i++) {
301         mag_print = fft_square_mag[i];
302         HAL_Delay(10);
303     }
304
305         //Calculate filter bank
306         //(1)
307         float fmel_0 = 2595.0 * log10 (1.0 + (phi_0 / 700.0));
308         float fmel_27 = 2595.0 * log10 (1.0 + (phi_27 / 700.0));
309         //(2)
310         fmel[0] = fmel_0;
311         for (int i = 1; i < 27; i++){
312             fmel[i] = fmel[i-1] + ((fmel_27 - fmel_0)/27.0);
313         }
314         fmel[27] = fmel_27;
315         //(3)
316         for (int i=0; i<28; i++){
317             phi[i] = (pow(10,(fmel[i]/2595.0))-1)*700.0;
318         }
319         // find lamda
320         for (int i=0; i<28; i++){
321             lamda[i] = (int) (513.0 * phi[i]/ 8000.0);
322         }
```

```
    //find H
    float num;
    float de;
    for (int m = 1; m < 27; m++){
        for (int k = 0; k < 513; k++){
            if (k <lamda[m-1]){
                H[m-1][k] = 0;
            }
            else if (lamda[m-1] <= k  && k <= lamda[m]){
                num = k - lamda[m-1];
                de = lamda[m] - lamda[m-1];
                H[m-1][k] = num / de;
            }
            else if (lamda[m] < k  && k <= lamda[m+1]){
                num = lamda[m+1] - k;
                de = lamda[m+1] - lamda[m];
                H[m-1][k] = num/de;
            }
            else if (k > lamda[m+1]) {
                H[m-1][k] = 0;
            }
        }
    }
```

```
    //Apply the filter bank
    for (int m = 0; m < 26; m++){
        energy_output[m] = 0.0;
    }
    for (int m = 0; m < 26; m++){
        for (int k = 0; k < 513; k++){
            energy_output[m] = energy_output[m] + fft_square_mag[k] * H[m][k];;
        }
    }

    //Take the logarithm of the output energy
    for (int m = 0; m < 26; m++){
        X[m] = log10(energy_output[m]);
    }

    //compute 13 MFCC values
    for (int i = 0; i < 13; i++){
        MFCC[i] = 0.0;
    }
    for (int i = 0; i < 13; i++){
        for (int m = 0; m < 26; m++){
            MFCC[i] = MFCC[i] + X[m] * cos((i+1)*((m+1) - 0.5)*PI/26.0);
        }
    }
```

```
372
373     // Print out 13 MFCC values for each monophthong
374     for (int i = 0; i < 13; i++){
375         printf("%.6f ", MFCC[i]);
376         HAL_Delay(5);
377     }
378     printf("\n");
379     HAL_Delay(5);
380
381     // Test the neural network
382     NN_Test();
383
384     record_done = 0;
385
386   }
387   /* USER CODE END 3 */
```

APPENDIX

1. Full data set of MFCC for 4 vowels used as input to MATLAB:

12.521294 0.174958 -1.344561 -0.076623 -1.724012 -2.587240 -1.369096 1.055934 1.235129 0.585653 -0.574375 -1.066263 0.358565
12.196064 2.858904 -1.362913 -0.172664 -1.732290 -2.749165 -1.135225 1.145651 2.722219 0.686349 -1.852131 -0.733001 0.984523
10.682019 0.088909 -0.470338 0.728294 -1.040092 -3.022297 -0.815288 1.412163 0.796246 1.829636 -0.157259 -1.279707 0.873794
12.084951 1.953838 0.205900 -0.090636 -1.857273 -2.771704 -1.091512 1.961280 0.348160 0.329331 -0.818015 -1.532363 -0.120790
10.494513 2.405301 -0.268496 -0.617088 -2.258059 -2.568399 -1.223852 0.676256 2.165411 0.753365 -1.919819 -1.225553 -0.106328
11.833959 3.197164 0.064380 -1.182013 -2.160507 -2.669070 -1.046731 2.345444 1.348341 0.072442 -1.746330 -1.111005 1.132734
9.502446 1.007894 -1.238622 -1.500237 -2.787935 -1.943257 -1.145675 0.166105 0.880239 1.595126 0.144771 -0.215908 0.551497
8.258411 -0.483707 -2.539245 -1.946263 -1.627562 -2.413054 -1.783580 0.648313 1.514612 1.321160 -0.572528 -0.155208 1.362937
12.147251 1.616168 -0.600164 0.458462 0.250166 -1.560410 -0.735556 1.263795 2.013748 2.227651 1.166829 0.076780 0.068194
11.245152 3.712317 -0.065163 -0.231182 -0.487215 -0.609304 -0.156422 0.714912 2.005636 1.060599 1.422316 0.980496 0.195385
11.701152 -0.348512 -3.830658 -4.090060 -1.018252 -1.200254 -0.868873 0.597185 -0.419109 1.058400 1.199742 0.428036 0.266682
9.098643 2.763535 -2.112976 -3.815985 -2.524427 -0.163539 -1.596501 -1.121448 -0.313826 1.436999 2.369367 0.964467 0.568347
8.132755 0.809295 -3.345814 -5.157874 -1.574333 0.189750 -0.927729 0.449796 2.019214 0.307116 -0.096433 0.348541 0.565460
12.059388 0.283882 -2.408491 -4.126391 -1.495228 -0.086376 -1.704588 -0.192652 0.642186 0.757482 2.330477 0.903519 -1.467948
9.298459 2.491360 -2.251161 -3.925737 -2.151158 -0.407013 -1.230878 -1.137123 1.233854 0.299741 0.587674 0.948178 -1.674482
5.012017 3.176749 -4.723607 -5.963263 -3.205498 0.153132 -1.200505 -0.223286 -1.233957 -0.373470 0.430601 -1.166369 -2.405948
7.391439 3.707999 -4.258865 -6.845537 -5.415598 -0.681430 -3.364177 -2.290572 -1.564105 0.131073 0.523813 -1.311441 -1.016457
6.821207 2.542604 -2.226503 -5.525793 -3.557868 -0.492445 -3.134067 -1.281811 -2.002284 -0.747043 1.053179 0.186445 -0.714736
6.426537 0.621784 -1.877508 -5.763745 -5.199039 -2.872720 -0.619914 -1.568397 -0.169961 0.258469 -0.211678 0.167719 0.526671
7.127992 0.117111 -3.043912 -7.080431 -3.077923 -2.354905 -3.314962 -0.529100 -0.775671 0.046130 -0.647705 -1.151770 -1.126190

8.000141 3.177424 6.627315 1.133401 -3.483103 0.940339 0.643302 -1.760913 -2.278487
0.379503 -0.889572 -0.468834 1.382815
8.223664 2.104690 4.345082 3.015715 -3.541087 0.415766 1.283904 -1.957562 -3.243599
-3.674354 -2.662391 -1.999819 1.141418
7.653903 2.992535 4.579241 2.529202 -1.676808 -1.034887 2.399608 -0.851374 -4.819238
-3.081451 -2.955086 -1.204214 0.483910
7.002313 3.344129 4.867488 1.606278 -1.769286 -0.844343 1.303077 -1.861995 -4.694458
-1.782274 -2.460690 -1.367033 1.305474
6.959472 3.540483 6.257175 0.631340 -1.587878 -0.068230 1.375002 -1.303783 -4.197305
-1.281420 -2.191026 -2.164634 0.723270
4.954088 5.010561 5.857918 2.258990 -1.453823 0.468379 2.349373 -2.643222 -5.301690
-2.160679 -1.868694 -1.266698 0.228342
7.635130 2.752355 5.498756 2.422173 -2.383076 -0.793640 1.478572 -0.647316 -3.603170
-1.984308 -2.417697 -1.361552 0.603778
8.156802 4.188470 4.222028 0.452344 -2.838296 -2.304440 0.376206 -0.107698 -4.784585
-2.548192 0.826109 0.271849 0.381884
4.448514 3.518846 2.521100 -2.835872 -2.549208 -4.404197 -3.122858 -3.029971 -6.161021
-1.127729 -0.576787 -1.875010 -1.653977
4.181564 1.528752 3.621434 -1.131467 -3.401796 -5.550424 -1.961695 -2.636583 -5.674644
-0.876143 -1.287924 -2.277220 -1.514829
7.158251 2.207701 1.277575 0.704031 -0.618194 -1.776287 -2.845358 -1.750890 -2.899096
-4.628100 -2.561395 -1.810125 -1.781341
9.982022 -0.090658 1.261495 -1.752615 -3.665403 -2.735932 -4.647233 -2.320327 -3.130254
-3.051744 -2.603276 -3.207509 -1.435448
4.232244 3.453812 1.480707 2.037630 0.744061 -1.321175 -0.886787 1.431330 -1.063781
-3.109822 -4.196004 -3.845508 -2.696941
4.833093 1.711769 2.152903 2.318296 0.695994 -0.964780 -1.466570 0.338956 -2.182795
-2.540259 -2.825215 -2.332813 -1.369955
6.730141 1.286935 1.245240 1.766413 1.177444 -1.962063 -1.671418 0.556445 -1.955640
-2.005966 -2.437952 -3.170290 -1.552523
7.562763 2.083011 3.233462 -0.192055 -3.279316 -2.605156 -1.115789 -0.264449 -2.147333
-1.417084 -1.945336 -1.559075 -1.189453
7.414279 0.641409 0.216928 2.407596 0.896903 -2.971477 -2.408737 0.702986 -1.901459
-1.628386 -2.881404 -2.981970 -2.220521
8.598079 2.224552 1.099279 -0.450157 2.216028 -0.148363 -1.528435 2.167074 -1.544742
-2.867620 -1.810260 -3.572266 -2.062668
8.330721 1.513442 1.390915 0.881303 -0.166891 -2.818058 -2.119398 -0.383246 -2.634270
-3.877439 -3.449375 -3.638525 -1.722170
5.946752 2.556117 1.337069 1.251523 0.202899 -2.856527 -1.733668 -1.702049 -3.520930
-3.648850 -3.198419 -3.495051 -3.194919

6.229072 4.618462 2.372729 1.467293 3.188258 4.524300 1.374426 1.275418 1.681749
1.080385 0.642026 -0.601284 0.035683

6.068496 5.345365 0.307241 2.372569 3.137924 2.770425 2.046208 1.449355 1.666713
0.469813 0.140802 0.669243 0.252079
7.452881 4.864214 2.113792 2.458277 2.425083 2.765859 1.404480 1.686773 1.287930
1.096357 0.478068 0.716119 0.294501
7.651528 5.469891 3.127917 2.806591 1.421231 2.218976 0.294183 2.296579 0.645353
1.928151 0.974419 1.160884 0.850402
8.270948 5.809175 1.605910 2.124715 2.812132 2.217942 0.862173 0.901173 0.630780
0.597355 1.949896 1.067222 1.026824
8.108044 5.541804 2.504148 3.209169 2.258501 1.798066 -0.040370 0.760140 1.507458
0.303354 1.504317 1.583834 0.771768
9.906258 4.953206 4.273740 2.477361 1.631714 1.822890 0.435606 0.044568 1.559748
1.215108 1.275270 1.672515 -0.039217
7.395000 4.017940 2.403948 1.263791 1.608301 2.934785 1.748818 1.180648 1.500205
1.348687 2.102161 1.207973 1.326200
7.367524 4.443199 2.846906 1.531516 2.903266 2.465232 1.981242 1.230872 1.400427
0.979831 0.311593 0.885644 0.084154
6.708427 4.008759 3.259073 1.884767 3.048493 3.372756 1.514836 1.175615 0.921994
1.506515 -0.170393 -0.330580 0.427292
4.145224 2.936863 3.511090 -0.814751 1.108271 0.983005 0.760190 2.171040 -1.779770
-1.436730 -0.240132 -1.782560 0.076449
2.776335 1.349880 5.513676 0.091480 -1.016242 1.153703 1.792039 2.416807 -0.072774
0.511730 1.999814 -1.843608 -0.285313
2.406469 1.774421 2.159808 0.372122 1.190490 1.393529 1.036618 2.218234 -0.469569
0.167986 0.044966 -2.123248 -1.836513
2.373346 0.898815 5.417112 -2.694492 -0.415008 0.957840 -1.953483 0.887665 -1.200251
-1.183738 0.000994 -3.519074 -1.233740
2.852253 1.020054 5.295193 -0.156795 -0.383401 0.931203 -0.765930 1.546051 -0.438797
0.536148 0.653183 -3.432788 -3.021470
2.131775 3.590356 2.158103 -0.714321 -1.582152 0.556171 0.534479 -1.647709 -2.156955
-2.236937 -2.574389 -1.786257 -1.464951
0.168975 1.020846 0.589154 -2.250949 -2.433496 0.140742 -1.773854 -1.009575 1.802903
-0.063991 0.531857 0.325546 -0.003714
1.599208 1.780453 1.544209 -1.897564 -3.247868 0.984631 -1.144011 -0.461860 0.137287
0.250188 1.339396 0.653599 -0.033818
1.070043 3.811641 3.166573 -1.003281 -1.370715 1.944491 0.834362 1.261846 1.602796
2.025660 0.787548 -0.681016 -0.935514
1.771560 1.044914 4.411692 -2.243901 -0.040038 2.292631 -0.912739 2.434160 -0.812431
-0.692388 -0.952471 -3.236523 -3.517020

9.162055 6.926170 2.938713 2.917564 1.968343 1.805116 0.138012 0.680611 1.306108
0.889103 -0.563999 0.796851 0.209229
10.412630 6.268187 3.456868 3.925261 2.936525 -0.751046 -0.154545 0.602680 0.527274
1.771343 1.645077 0.379385 -0.538107

9.785851 6.524476 2.546709 3.547203 2.338303 0.336596 -0.467217 1.750331 0.167272
1.370137 1.014204 -0.004614 0.024033
8.768291 7.456970 3.155121 5.308012 1.120787 -0.780947 -0.146255 0.457127 1.239515
1.596618 2.096518 0.196924 -0.189902
10.227699 5.302967 2.139040 3.805108 2.397553 1.557785 0.550783 1.034283 1.639994
1.353690 1.358233 1.283847 0.528893
7.877414 4.665617 2.428472 3.190732 2.320799 1.554312 1.170868 0.953304 2.687410
2.172795 2.189018 1.606814 0.103733
9.411840 5.302052 2.889334 3.060170 1.798026 0.594309 0.580756 1.175735 1.713119
1.726400 2.387502 1.259259 0.512056
8.930830 4.453854 2.777008 3.561657 2.119388 0.826657 0.073774 0.940436 0.451609
2.767158 1.850958 1.772820 0.566036
7.088650 6.252358 1.796905 4.490477 3.727173 0.162039 0.462850 0.483821 -0.171765
0.641807 1.351474 0.957944 0.290240
9.314441 5.103889 3.523875 3.398692 3.181037 0.322349 0.217161 -0.246986 1.148404
0.201489 0.793789 -0.155379 -0.323016
6.134793 1.912525 1.019914 1.954047 2.509468 0.102138 -0.880429 0.232889 1.141834
0.258273 0.712141 0.146363 -0.394748
6.474701 3.320892 0.353951 1.754865 -1.789180 -1.883777 0.722706 1.585988 1.985508
2.028599 1.154304 0.260774 -0.277602
6.868758 2.291903 0.239902 1.156809 -0.932858 -1.112580 -0.516657 2.129200 1.836162
2.040912 2.102670 0.696643 0.312477
6.428533 3.066974 0.852586 1.661873 0.987962 -0.715916 -0.620886 0.719238 0.932605
1.293938 1.048014 0.762227 0.170938
4.062714 3.250232 0.226133 3.239033 0.971932 -0.368868 -0.940291 0.970618 0.804764
1.191258 1.398092 0.915787 -0.076676
6.286328 2.593553 0.173979 2.964668 2.847868 0.133524 -0.910833 0.082153 0.258801
-0.547986 0.525333 0.180324 0.364945
4.796525 2.012552 1.259746 2.523209 1.257365 0.856676 -1.066934 0.610541 0.820730
0.434272 0.990111 0.448948 -0.342150
3.791531 3.340684 0.268206 2.646048 2.078447 0.011420 -0.595823 0.757658 0.945085
1.109105 1.106709 0.768914 -0.246119
4.016283 3.047761 0.754758 4.447306 2.430424 1.001392 0.298307 0.035592 -0.276841
0.619042 -0.386815 -0.337445 -0.931199
3.822670 2.189681 0.097101 2.777539 2.106683 3.417777 0.568612 0.258137 0.161841
0.813384 -0.504197 -0.754163 -0.198605

2. <u>Coefficients from myNeuralNetworkFunction:</u>

float x1_offset[13] = {0.168975, -0.483707, -4.723607, -7.080431, -5.415598, -5.550424,
-4.647233, -3.029971, -6.161021, -4.6281, -4.196004, -3.845508, -3.51702};
float x1_gain[13] = {0.161912916918677,0.251867693396923, 0.176197140637562,
0.161440787998944, 0.218752061054575, 0.198516604524352, 0.283815116589121,
0.366023435382497, 0.22514307842634, 0.270443573435842, 0.303789500609554,
0.355977792681381, 0.408177010042175};

```
float x1_ymin = -1;

float b1[5] = {0.050520948190351855356, 0.70348200754524303768,
-1.6159629461489435354, -0.268716337854188414, 1.2246177545239944617};
float IW1_1[5][13]= {{1.384566337748454945, -0.44985171672673329724,
-0.27630685532942456106, 0.6222110370845526095, -0.59520115878139723264,
-0.06723842420797142283, 0.29963809246286626786, 0.035482640888365064857,
0.51736728592787062375, 1.4510342830099445255, 1.5340735805871064112,
0.65936759642707187812, 0.52629307724332796692},
                        {-0.22230587226674355938, -0.42836039552848043099,
1.40990844402417137, -0.23094381370271427345, -1.0758533635613003465,
-1.2760557705613160273, -0.32909556027408526369, -0.93183560125679554265,
-1.6283550662262369357, -1.5193640972555380042, -0.86176504131490117011,
-1.077773107409351061, 0.44727747563771630412},
                        {2.2795905194465828636, -0.95058944914825882488,
-2.2246429981402040532, -2.2015534912645584598, -1.068282131021232928,
-1.4036945083419574143, -0.89684628495974993978, -0.53359046932510700856,
0.92876624617887260094, 0.10958639741177725324, -0.32012727178488270541,
-0.28180968078716783776, 0.66026281142599219098},
                        {-2.5925877835402535432, 0.34517160861479179168,
1.1425911459977200479, -0.34783966107991698413, 1.4957963255827930737,
1.7385282126140975123, 0.49405964976698280022, 0.23646794566132098292,
0.6129699312548611001, 0.53657503155923347293, 0.62322521536110464524,
0.67131513559555033854, -0.28125726707517445524},
                        {1.9456528174172766921, -0.3596612709403338437,
-3.53838922097893116, 4.9160359417693220152, -0.57201813140127588664,
-5.1518349754164916021, 0.27455485743265967136, -1.58092019957196194482,
-0.10440626539176907361, 3.5850125877196803437, -1.2675029247541076405,
-2.2516787176543027194, 0.4786657865822083191}
                                        };

float b2[4] =
{1.224118198426011439,0.26262828789044301292,0.049748511555626914737,-1.226749801
8903089108};
float LW2_1[4][5] = {{0.96250487394378159145, -1.6572097177601095019,
5.3380275130970469277, -3.1656740604425834817, 1.1655241956577135909},
                        {-0.37822411489095258963, 5.8786383096330387943,
-2.0091087053128626749, -2.5244051761217538576, 2.5854408253040741528},
                        {-1.4834105324697453021, -0.3091029000151264694,
-3.091490067236965178, 2.6854292909987456106, -8.8054842784664177913},
                        {-0.65771757901938732171, -3.5592461400504693536,
-2.1867378906098764446, 0.94507864444246947322, 4.317920638129885802}
                                        };
```

3. Full main.c file:
https://drive.google.com/drive/folders/1hyxMnJWpbiXtb1JQDanS3eZ4BXrlbOQv?usp=sharing

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "string.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
```

```
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
#if defined ( __ICCARM__ ) /*!< IAR Compiler */

#pragma location=0x30040000
ETH_DMADescTypeDef  DMARxDscrTab[ETH_RX_DESC_CNT]; /* Ethernet Rx DMA
Descriptors */
#pragma location=0x30040060
ETH_DMADescTypeDef  DMATxDscrTab[ETH_TX_DESC_CNT]; /* Ethernet Tx DMA
Descriptors */
#pragma location=0x30040200
uint8_t Rx_Buff[ETH_RX_DESC_CNT][ETH_MAX_PACKET_SIZE]; /* Ethernet Receive Buffers
*/

#elif defined ( __CC_ARM )  /* MDK ARM Compiler */

__attribute__((at(0x30040000))) ETH_DMADescTypeDef
DMARxDscrTab[ETH_RX_DESC_CNT]; /* Ethernet Rx DMA Descriptors */
__attribute__((at(0x30040060))) ETH_DMADescTypeDef
DMATxDscrTab[ETH_TX_DESC_CNT]; /* Ethernet Tx DMA Descriptors */
__attribute__((at(0x30040200))) uint8_t
Rx_Buff[ETH_RX_DESC_CNT][ETH_MAX_PACKET_SIZE]; /* Ethernet Receive Buffer */

#elif defined ( __GNUC__ ) /* GNU Compiler */

ETH_DMADescTypeDef DMARxDscrTab[ETH_RX_DESC_CNT]
__attribute__((section(".RxDecripSection"))); /* Ethernet Rx DMA Descriptors */
ETH_DMADescTypeDef DMATxDscrTab[ETH_TX_DESC_CNT]
__attribute__((section(".TxDecripSection")));   /* Ethernet Tx DMA Descriptors */
uint8_t Rx_Buff[ETH_RX_DESC_CNT][ETH_MAX_PACKET_SIZE]
__attribute__((section(".RxArraySection"))); /* Ethernet Receive Buffers */

#endif

ETH_TxPacketConfig TxConfig;
ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

ETH_HandleTypeDef heth;

TIM_HandleTypeDef htim1;

UART_HandleTypeDef huart3;
```

```
PCD_HandleTypeDef hpcd_USB_OTG_FS;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ETH_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_DMA_Init(void);
static void MX_ADC1_Init(void);
static void MX_TIM1_Init(void);
static void MX_USB_OTG_FS_PCD_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
#include <arm_math.h>
#include <stdio.h>
#include <stdlib.h>
#define ARM_MATH_CM7
#define __FPU_PRESENT 1

#define ADC_BUF_SIZE 1024

uint16_t ADC_buff[ADC_BUF_SIZE];
int record_done=0;
float mag_print;
float window[ADC_BUF_SIZE]; //Hamming window
float window_ADC[ADC_BUF_SIZE]; //after apply Hamming window to the input
float fft_square_mag[ADC_BUF_SIZE];
float phi_0 = 250.0; float phi_27 = 8000.0;
int M = 26;
int Z = 13;
float BETA = 0.46;
float ALPHA = 0.54;
float fmel[28];
float phi[28];
float lamda[28];
```

```
float H[28][513];
float energy_output[26];

float X[26]; // log output energy
float MFCC[13];

float ADC_buff1[ADC_BUF_SIZE];

float output[4];

void NN_Test(){

	float x1_offset[13] =
{0.168975,-0.483707,-4.723607,-7.080431,-5.415598,-5.550424,-4.647233,-3.029971,-6.16102
1,-4.6281,-4.196004,-3.845508,-3.51702};
	float x1_gain[13] =
{0.161912916918677,0.251867693396923,0.176197140637562,0.161440787998944,0.218752
061054575,0.198516604524352,0.283815116589121,0.366023435382497,0.22514307842634,
0.270443573435842,0.303789500609554,0.355977792681381,0.408177010042175};
	float x1_ymin = -1;

	float b1[5] =
{0.050520948190351855356,0.70348200754524303768,-1.6159629461489435354,-0.2687163
37854188414,1.2246177545239944617};
	float IW1_1[5][13]= {{1.384566337748454945, -0.44985171672673329724,
-0.27630685532942456106, 0.6222110370845526095, -0.59520115878139723264,
-0.06723842420797142283, 0.29963809246286626786, 0.035482640888365064857,
0.51736728592787062375, 1.4510342830099445255, 1.5340735805871064112,
0.65936759642707187812, 0.52629307724332796692},
			{-0.22230587226674355938, -0.42836039552848043099,
1.40990844402417137, -0.23094381370271427345, -1.0758533635613003465,
-1.2760557705613160273, -0.32909556027408526369, -0.93183560125679554265,
-1.6283550662262369357, -1.5193640972555380042, -0.86176504131490117011,
-1.077773107409351061, 0.44727747563771630412},
			{2.2795905194465828636, -0.95058944914825882488,
-2.2246429981402040532, -2.2015534912645584598, -1.068282131021232928,
-1.4036945083419574143, -0.89684628495974993978, -0.53359046932510700856,
0.92876624617887260094, 0.10958639741177725324, -0.32012727178488270541,
-0.28180968078716783776, 0.66026281142599219098},
			{-2.5925877835402535432, 0.34517160861479179168,
1.1425911459977200479, -0.34783966107991698413, 1.4957963255827930737,
1.7385282126140975123, 0.49405964976698280022, 0.23646794566132098292,
0.6129699312548611001, 0.53657503155923347293, 0.62322521536110464524,
0.67131513559555033854, -0.28125726707517445524},
```

```
                        {1.9456528174172766921, -0.3596612709403338437,
-3.53838922097893116, 4.9160359417693220152, -0.57201813140127588664,
-5.1518349754164916021, 0.27455485743265967136, -1.5809201995719619482,
-0.10440626539176907361, 3.5850125877196803437, -1.2675029247541076405,
-2.2516787176543027194, 0.4786657865822083191}
                                        };

        float b2[4] =
{1.2241118198426011439,0.26262828789044301292,0.049748511555626914737,-1.226749801
8903089108};
        float LW2_1[4][5] = {{0.96250487394378159145, -1.6572097177601095019,
5.3380275130970469277, -3.1656740604425834817, 1.1655241956577135909},
                        {-0.37822411489095258963, 5.8786383096330387943,
-2.0091087053128626749, -2.5244051761217538576, 2.5854408253040741528},
                        {-1.4834105324697453021, -0.3091029000151264694,
-3.091490067236965178, 2.6854292909987456106, -8.8054842784664177913},
                        {-0.65771757901938732171, -3.5592461400504693536,
-2.1867378906098764446, 0.94507864444246947322, 4.317920638129885802}
                                        };

        // Data pre-processing
        float Y[13];
        for (int i=0; i<13; i++){
         Y[i] = MFCC[i] - x1_offset[i];
         Y[i] = Y[i] * x1_gain[i];
         Y[i] = Y[i] + x1_ymin;
        }

        // Input to hidden
        float h[5] = {0.0};
        for (int x=0; x<5; x++){
                float sum1 = 0.0;
                for (int k=0; k<13; k++){
                        sum1 = sum1 + IW1_1[x][k]*Y[k];
                }
                h[x] = 2.0/(1.0+exp(-2*(sum1+b1[x]))) -1.0;
        }

        //hidden to output
        float o[4] = {0.0};
        for (int x=0; x<4; x++){
                float sum2 = 0.0;
                for (int k=0; k<5; k++){
                        sum2 = sum2 + LW2_1[x][k]*h[k];
```

```
                }
                o[x] = sum2+b2[x];
        }
        float sum3 = 0.0;
        for (int x=0; x<4; x++){
                sum3 = sum3 + exp(o[x]);
        }
        for (int x=0; x<4; x++){
         output[x] = exp(o[x])/sum3;
        }
}



/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
```

```
  MX_ETH_Init();
  MX_USART3_UART_Init();
  MX_DMA_Init();
  MX_ADC1_Init();
  MX_TIM1_Init();
  MX_USB_OTG_FS_PCD_Init();
  /* USER CODE BEGIN 2 */
  if(HAL_ADCEx_Calibration_Start(&hadc1, ADC_CALIB_OFFSET,ADC_SINGLE_ENDED) !=
HAL_OK){
          Error_Handler();
  }

  // start timer TIM1
  if (HAL_TIM_Base_Start(&htim1) != HAL_OK){
          Error_Handler();
  }

  //FFT handler initialization
  arm_cfft_instance_f32 fft_handler;
  arm_cfft_init_f32(&fft_handler, ADC_BUF_SIZE);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

          //HAL_Delay(1000);
          // start sampling
          if (HAL_ADC_Start_DMA(&hadc1, (uint32_t *)&ADC_buff, ADC_BUF_SIZE) !=
HAL_OK) {
            Error_Handler();
          }

          while(1){
                  if(record_done == 1){break;}
          }

          float sum = 0.0;
          for (int i=0; i < ADC_BUF_SIZE; i++) {
```

```
                sum = sum + ADC_buff[i];
        }
        float avg = sum/ ADC_BUF_SIZE;

        // eliminate average  and store in a float array
        for (int i=0; i < ADC_BUF_SIZE; i++) {
                ADC_buff1[i] = ADC_buff[i] - avg;
        }

        // Print out the signal after eliminate average
        /*for(int i = 0; i < ADC_BUF_SIZE; i++){
                 mag_print = ADC_buff1[i];
                 HAL_Delay(5);
        }*/

        // window the signal
        for (int i=0; i<ADC_BUF_SIZE; i++){
                window[i] = ALPHA  - BETA*cos(2.0*PI*i/(ADC_BUF_SIZE-1));
                window_ADC[i] = ADC_buff1[i] * window[i];
        }

        // Print out the signal after applying window
  /* for (int i=0; i < ADC_BUF_SIZE; i++) {
                          mag_print = window_ADC[i];
                          HAL_Delay(10);
        }*/

        // Convert to complex array
        float* complex_input = (float*)malloc(ADC_BUF_SIZE*2*sizeof(float));
        for(int i = 0; i< ADC_BUF_SIZE; i++){
         complex_input[2*i] = window_ADC[i];
         complex_input[2*i + 1] = 0;
        }

        //Perform FFT
        //Take K = N/2 +1 points = 513 points
        arm_cfft_f32(&fft_handler, complex_input, 0, 1); //output is overwritten to complex_input
        for (int i=0; i < 513; i++){
                fft_square_mag[i] = (complex_input[2*i])*(complex_input[2*i]) +
(complex_input[2*i+1])*(complex_input[2*i+1]); //square magnitude
        }

        // Print out the FFT square magnitude
        /*for (int i=0; i < 513; i++) {
```

```
        mag_print = fft_square_mag[i];
        HAL_Delay(10);
}*/

//Calculate filter bank
//(1)
float fmel_0 = 2595.0 * log10 (1.0 + (phi_0 / 700.0));
float fmel_27 = 2595.0 * log10 (1.0 + (phi_27 / 700.0));
//(2)
fmel[0] = fmel_0;
for (int i = 1; i < 27; i++){
        fmel[i] = fmel[i-1] + ((fmel_27 - fmel_0)/27.0);
}
fmel[27] = fmel_27;
//(3)
for (int i=0; i<28; i++){
        phi[i] = (pow(10,(fmel[i]/2595.0))-1)*700.0;
}
// find lamda
for (int i=0; i<28; i++){
        lamda[i] = (int) (513.0 * phi[i]/ 8000.0);
}
//find H
float num;
float de;
for (int m = 1; m < 27; m++){
        for (int k = 0; k < 513; k++){
                if (k <lamda[m-1]){
                        H[m-1][k] = 0;
                }
                else if (lamda[m-1] <= k  && k <= lamda[m]){
                        num = k - lamda[m-1];
                        de = lamda[m] - lamda[m-1];
                        H[m-1][k] = num / de;
                }
                else if (lamda[m] < k  && k <= lamda[m+1]){
                        num = lamda[m+1] - k;
                        de = lamda[m+1] - lamda[m];
                        H[m-1][k] = num/de;
                }
                else if (k > lamda[m+1]) {
                        H[m-1][k] = 0;
                }
        }
```

```
        }


        //Apply the filter bank
        for (int m = 0; m < 26; m++){
                energy_output[m] = 0.0;
        }
        for (int m = 0; m < 26; m++){
                for (int k = 0; k < 513; k++){
                        energy_output[m] = energy_output[m] + fft_square_mag[k] * H[m][k];;
                }
        }

        //Take the logarithm of the output energy
        for (int m = 0; m < 26; m++){
                X[m] = log10(energy_output[m]);
        }

        //compute 13 MFCC values
        for (int i = 0; i < 13; i++){
                MFCC[i] = 0.0;
        }
        for (int i = 0; i < 13; i++){
                for (int m = 0; m < 26; m++){
                        MFCC[i] = MFCC[i] + X[m] * cos((i+1)*((m+1) - 0.5)*PI/26.0);
                }
        }

        // Print out 13 MFCC values for each monophthong
        for (int i = 0; i < 13; i++){
                printf("%.6f ", MFCC[i]);
                HAL_Delay(5);
        }
        printf("\n");
        HAL_Delay(5);

        // Test the neural network
        NN_Test();

        record_done = 0;

  }
  /* USER CODE END 3 */
}
```

```
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Supply configuration update enable
  */
  HAL_PWREx_ConfigSupply(PWR_LDO_SUPPLY);
  /** Configure the main internal regulator output voltage
  */
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);

  while(!__HAL_PWR_GET_FLAG(PWR_FLAG_VOSRDY)) {}
  /** Macro to configure the PLL clock source
  */
  __HAL_RCC_PLL_PLLSOURCE_CONFIG(RCC_PLLSOURCE_HSE);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
  RCC_OscInitStruct.HSIState = RCC_HSI_DIV1;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 18;
  RCC_OscInitStruct.PLL.PLLP = 2;
  RCC_OscInitStruct.PLL.PLLQ = 3;
  RCC_OscInitStruct.PLL.PLLR = 2;
  RCC_OscInitStruct.PLL.PLLRGE = RCC_PLL1VCIRANGE_3;
  RCC_OscInitStruct.PLL.PLLVCOSEL = RCC_PLL1VCOMEDIUM;
  RCC_OscInitStruct.PLL.PLLFRACN = 6144;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
```

```
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2
                              |RCC_CLOCKTYPE_D3PCLK1|RCC_CLOCKTYPE_D1PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.SYSCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB3CLKDivider = RCC_APB3_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_APB1_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_APB2_DIV1;
  RCC_ClkInitStruct.APB4CLKDivider = RCC_APB4_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_MultiModeTypeDef multimode = {0};
  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */
  /** Common config
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV2;
  hadc1.Init.Resolution = ADC_RESOLUTION_16B;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc1.Init.LowPowerAutoWait = DISABLE;
```

```
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIG_T1_TRGO;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
  hadc1.Init.ConversionDataManagement = ADC_CONVERSIONDATA_DMA_ONESHOT;
  hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
  hadc1.Init.LeftBitShift = ADC_LEFTBITSHIFT_NONE;
  hadc1.Init.OversamplingMode = DISABLE;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure the ADC multi-mode
  */
  multimode.Mode = ADC_MODE_INDEPENDENT;
  if (HAL_ADCEx_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_15;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
  sConfig.OffsetSignedSaturation = DISABLE;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief ETH Initialization Function
  * @param None
  * @retval None
  */
```

```
static void MX_ETH_Init(void)
{

  /* USER CODE BEGIN ETH_Init 0 */

  /* USER CODE END ETH_Init 0 */

   static uint8_t MACAddr[6];

  /* USER CODE BEGIN ETH_Init 1 */

  /* USER CODE END ETH_Init 1 */
  heth.Instance = ETH;
  MACAddr[0] = 0x00;
  MACAddr[1] = 0x80;
  MACAddr[2] = 0xE1;
  MACAddr[3] = 0x00;
  MACAddr[4] = 0x00;
  MACAddr[5] = 0x00;
  heth.Init.MACAddr = &MACAddr[0];
  heth.Init.MediaInterface = HAL_ETH_RMII_MODE;
  heth.Init.TxDesc = DMATxDscrTab;
  heth.Init.RxDesc = DMARxDscrTab;
  heth.Init.RxBuffLen = 1524;

  /* USER CODE BEGIN MACADDRESS */

  /* USER CODE END MACADDRESS */

  if (HAL_ETH_Init(&heth) != HAL_OK)
  {
    Error_Handler();
  }

  memset(&TxConfig, 0 , sizeof(ETH_TxPacketConfig));
  TxConfig.Attributes = ETH_TX_PACKETS_FEATURES_CSUM |
ETH_TX_PACKETS_FEATURES_CRCPAD;
  TxConfig.ChecksumCtrl = ETH_CHECKSUM_IPHDR_PAYLOAD_INSERT_PHDR_CALC;
  TxConfig.CRCPadCtrl = ETH_CRC_PAD_INSERT;
  /* USER CODE BEGIN ETH_Init 2 */

  /* USER CODE END ETH_Init 2 */

}
```

```
/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 0;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 4687;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterOutputTrigger2 = TIM_TRGO2_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */
```

```
  /* USER CODE END TIM1_Init 2 */

}

/**
  * @brief USART3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART3_UART_Init(void)
{

  /* USER CODE BEGIN USART3_Init 0 */

  /* USER CODE END USART3_Init 0 */

  /* USER CODE BEGIN USART3_Init 1 */

  /* USER CODE END USART3_Init 1 */
  huart3.Instance = USART3;
  huart3.Init.BaudRate = 115200;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart3.Init.ClockPrescaler = UART_PRESCALER_DIV1;
  huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart3) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_UARTEx_SetTxFifoThreshold(&huart3, UART_TXFIFO_THRESHOLD_1_8) !=
HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_UARTEx_SetRxFifoThreshold(&huart3, UART_RXFIFO_THRESHOLD_1_8) !=
HAL_OK)
  {
    Error_Handler();
  }
```

```
  if (HAL_UARTEx_DisableFifoMode(&huart3) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART3_Init 2 */

  /* USER CODE END USART3_Init 2 */

}

/**
  * @brief USB_OTG_FS Initialization Function
  * @param None
  * @retval None
  */
static void MX_USB_OTG_FS_PCD_Init(void)
{

  /* USER CODE BEGIN USB_OTG_FS_Init 0 */

  /* USER CODE END USB_OTG_FS_Init 0 */

  /* USER CODE BEGIN USB_OTG_FS_Init 1 */

  /* USER CODE END USB_OTG_FS_Init 1 */
  hpcd_USB_OTG_FS.Instance = USB_OTG_FS;
  hpcd_USB_OTG_FS.Init.dev_endpoints = 9;
  hpcd_USB_OTG_FS.Init.speed = PCD_SPEED_FULL;
  hpcd_USB_OTG_FS.Init.dma_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.phy_itface = PCD_PHY_EMBEDDED;
  hpcd_USB_OTG_FS.Init.Sof_enable = ENABLE;
  hpcd_USB_OTG_FS.Init.low_power_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.lpm_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.battery_charging_enable = ENABLE;
  hpcd_USB_OTG_FS.Init.vbus_sensing_enable = ENABLE;
  hpcd_USB_OTG_FS.Init.use_dedicated_ep1 = DISABLE;
  if (HAL_PCD_Init(&hpcd_USB_OTG_FS) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USB_OTG_FS_Init 2 */

  /* USER CODE END USB_OTG_FS_Init 2 */
```

```
}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Stream0_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream0_IRQn);

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();
  __HAL_RCC_GPIOE_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LD1_Pin|LD3_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(USB_OTG_FS_PWR_EN_GPIO_Port, USB_OTG_FS_PWR_EN_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
```

```
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : LD1_Pin LD3_Pin */
  GPIO_InitStruct.Pin = LD1_Pin|LD3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : USB_OTG_FS_PWR_EN_Pin */
  GPIO_InitStruct.Pin = USB_OTG_FS_PWR_EN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(USB_OTG_FS_PWR_EN_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : USB_OTG_FS_OVCR_Pin */
  GPIO_InitStruct.Pin = USB_OTG_FS_OVCR_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(USB_OTG_FS_OVCR_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc){
        if(HAL_ADC_Stop_DMA(&hadc1) != HAL_OK){
                Error_Handler();
        }
        record_done = 1;
}
```

```
int _write(int file, char *ptr, int len)
{
  /* Implement your write code here, this is used by puts and printf for example */
  int i=0;
  for(i=0 ; i<len ; i++)
    ITM_SendChar((*ptr++));
  return len;
}
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
          HAL_GPIO_WritePin(GPIOB, LD3_Pin, GPIO_PIN_SET);
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

/********** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/