```matlab
%%%%% Main Function %%%%%

tic
clc; clear all;
Fs = 44100; nBits = 8; nChannels = 1;
Tw = 20;           % analysis frame duration (ms)
Ts = 10;           % analysis frame shift (ms)
alpha = 0.97;      % preemphasis coefficient
R = [ 300 3700 ];  % frequency range to consider
M = 20;            % number of filterbank channels
C = 13;            % number of cepstral coefficients
L = 22;            % cepstral sine lifter parameter
% hamming window (see Eq. (5.2) on p.73 of [1])
hamming = @(N)(0.54-0.46*cos(2*pi*[0:N-1].'/(N-1)));
%%
A = [];
for f = 1 : 4
    n = input( 'For how many sec you want to record?' );
    recObj = audiorecorder(Fs,nBits,nChannels);
    disp( 'Start speaking.' );
    recordblocking( recObj,n );
    disp( 'End of Recording.' );
    play(recObj);
    y = getaudiodata(recObj);
    [ b, FBEs, frames ] =  mfcc( y, Fs, Tw, Ts, alpha, hamming, R, M, C, L );
   A(:,:,f) = b;

end

%% Connection of Bluetooth module
a = Bluetooth('HC-05',1);
fopen(a);
'Bluetooth Successfully Connected'



%%
% load('C:\Users\User\Desktop\Audio.mat')
% while(1)

%   n = input( 'For how many sec you want to record?' );
    recObj = audiorecorder(Fs,nBits,nChannels);
    disp( 'Start speaking.' );
    recordblocking( recObj,1 );
    disp( 'End of Recording.' );
%     play(recObj);
    y = getaudiodata(recObj);
    [ b, FBEs, frames ] =  mfcc( y, Fs, Tw, Ts, alpha, hamming, R, M, C, L );


error = zeros(4,length(b));
for k = 1:4

    for j = 1:length(b)

        error(k,j) = dtw(A(:,:,k),circshift(b,j-1,2));
```

```
    end

end

err = min(error,[],2);

result = find(err == min(err))

fprintf(a,result);
% end
toc
```

```
% MFCC Mel frequency cepstral coefficient feature extraction.
%
%   MFCC(S,FS,TW,TS,ALPHA,WINDOW,R,M,N,L) returns mel frequency
%   cepstral coefficients (MFCCs) computed from speech signal given
%   in vector S and sampled at FS (Hz). The speech signal is first
%   preemphasised using a first order FIR filter with preemphasis
%   coefficient ALPHA. The preemphasised speech signal is subjected
%   to the short-time Fourier transform analysis with frame durations
%   of TW (ms), frame shifts of TS (ms) and analysis window function
%   given as a function handle in WINDOW. This is followed by magnitude
%   spectrum computation followed by filterbank design with M triangular
%   filters uniformly spaced on the mel scale between lower and upper
%   frequency limits given in R (Hz). The filterbank is applied to
%   the magnitude spectrum values to produce filterbank energies (FBEs)
%   (M per frame). Log-compressed FBEs are then decorrelated using the
%   discrete cosine transform to produce cepstral coefficients. Final
%   step applies sinusoidal lifter to produce liftered MFCCs that
%   closely match those produced by HTK [1].
%
%   [CC,FBE,FRAMES]=MFCC(...) also returns FBEs and windowed frames,
%   with feature vectors and frames as columns.
%
%   This framework is based on Dan Ellis' rastamat routines [2]. The
%   emphasis is placed on closely matching MFCCs produced by HTK [1]
%   (refer to p.337 of [1] for HTK's defaults) with simplicity and
%   compactness as main considerations, but at a cost of reduced
%   flexibility. This routine is meant to be easy to extend, and as
%   a starting point for work with cepstral coefficients in MATLAB.
%   The triangular filterbank equations are given in [3].
%
%   Inputs
%           S is the input speech signal (as vector)
%
%           FS is the sampling frequency (Hz)
%
%           TW is the analysis frame duration (ms)
%
%           TS is the analysis frame shift (ms)
%
%           ALPHA is the preemphasis coefficient
%
%           WINDOW is a analysis window function handle
%
%           R is the frequency range (Hz) for filterbank analysis
%
%           M is the number of filterbank channels
%
%           N is the number of cepstral coefficients
%             (including the 0th coefficient)
%
%           L is the liftering parameter
%
%   Outputs
%           CC is a matrix of mel frequency cepstral coefficients
%              (MFCCs) with feature vectors as columns
%
%           FBE is a matrix of filterbank energies
```

```
%               with feature vectors as columns
%
%           FRAMES is a matrix of windowed frames
%                  (one frame per column)
%
%   Example
%           Tw = 25;           % analysis frame duration (ms)
%           Ts = 10;           % analysis frame shift (ms)
%           alpha = 0.97;      % preemphasis coefficient
%           R = [ 300 3700 ];  % frequency range to consider
%           M = 20;            % number of filterbank channels
%           C = 13;            % number of cepstral coefficients
%           L = 22;            % cepstral sine lifter parameter
%
%           % hamming window (see Eq. (5.2) on p.73 of [1])
%           hamming = @(N)(0.54-0.46*cos(2*pi*[0:N-1].'/(N-1)));
%
%           % Read speech samples, sampling rate and precision from file
%           [ speech, fs, nbits ] = wavread( 'sp10.wav' );
%
%           % Feature extraction (feature vectors as columns)
%           [ MFCCs, FBEs, frames ] = ...
%                           mfcc( speech, fs, Tw, Ts, alpha, hamming, R, M, 
C, L );
%
%           % Plot cepstrum over time
%           figure('Position', [30 100 800 200], 'PaperPositionMode', 'auto', 
...
%                  'color', 'w', 'PaperOrientation', 'landscape', 'Visible', 
'on' );
%
%           imagesc( [1:size(MFCCs,2)], [0:C-1], MFCCs );
%           axis( 'xy' );
%           xlabel( 'Frame index' );
%           ylabel( 'Cepstrum index' );
%           title( 'Mel frequency cepstrum' );
%
%   References
%
%           [1] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D.,
%               Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D.,
%               Valtchev, V., Woodland, P., 2006. The HTK Book (for HTK
%               Version 3.4.1). Engineering Department, Cambridge University.
%               (see also: http://htk.eng.cam.ac.uk)
%
%           [2] Ellis, D., 2005. Reproducing the feature outputs of
%               common programs using Matlab and melfcc.m. url:
%               http://labrosa.ee.columbia.edu/matlab/rastamat/mfccs.html
%
%           [3] Huang, X., Acero, A., Hon, H., 2001. Spoken Language
%               Processing: A guide to theory, algorithm, and system
%               development. Prentice Hall, Upper Saddle River, NJ,
%               USA (pp. 314-315).
%
%   See also EXAMPLE, COMPARE, FRAMES2VEC, TRIFBANK.

%   Author: Kamil Wojcicki, September 2011
```

```
    %% PRELIMINARIES

    % Ensure correct number of inputs
    if( nargin~= 10 ), help mfcc; return; end;

    % Explode samples to the range of 16 bit shorts
    if( max(abs(speech))<=1 ), speech = speech * 2^15; end;

    Nw = round( 1E-3*Tw*fs );    % frame duration (samples)
    Ns = round( 1E-3*Ts*fs );    % frame shift (samples)

    nfft = 2^nextpow2( Nw );     % length of FFT analysis
    K = nfft/2+1;                % length of the unique part of the FFT


    %% HANDY INLINE FUNCTION HANDLES

    % Forward and backward mel frequency warping (see Eq. (5.13) on p.76 of
[1])
    % Note that base 10 is used in [1], while base e is used here and in HTK
code
    hz2mel = @( hz )( 1127*log(1+hz/700) );     % Hertz to mel warping
function
    mel2hz = @( mel )( 700*exp(mel/1127)-700 ); % mel to Hertz warping
function

    % Type III DCT matrix routine (see Eq. (5.14) on p.77 of [1])
    dctm = @( N, M )( sqrt(2.0/M) * cos( repmat([0:N-1].',1,M) ...
                                  .* repmat(pi*([1:M]-0.5)/M,N,1) ) );

    % Cepstral lifter routine (see Eq. (5.12) on p.75 of [1])
    ceplifter = @( N, L )( 1+0.5*L*sin(pi*[0:N-1]/L) );


    %% FEATURE EXTRACTION

    % Preemphasis filtering (see Eq. (5.1) on p.73 of [1])
    speech = filter( [1 -alpha], 1, speech ); % fvtool( [1 -alpha], 1 );

    % Framing and windowing (frames as columns)
    frames = vec2frames( speech, Nw, Ns, 'cols', window, false );

    % Magnitude spectrum computation (as column vectors)
    MAG = abs( fft(frames,nfft,1) );

    % Triangular filterbank with uniformly spaced filters on mel scale
    H = trifbank( M, K, R, fs, hz2mel, mel2hz ); % size of H is M x K

    % Filterbank application to unique part of the magnitude spectrum
    FBE = H * MAG(1:K,:); % FBE( FBE<1.0 ) = 1.0; % apply mel floor
```

```
    % DCT matrix computation
    DCT = dctm( N, M );

    % Conversion of logFBEs to cepstral coefficients through DCT
    CC =  DCT * log( FBE );

    % Cepstral lifter computation
    lifter = ceplifter( N, L );

    % Cepstral liftering gives liftered cepstral coefficients
    CC = diag( lifter ) * CC; % ~ HTK's MFCCs


% EOF
```

%%%% ARDUINO CODE %%%%

```
// bluetooth arduino matlab

const int p2 = 2;
const int p3 = 3;
const int p4 = 4;
const int p5 = 5;
int i = 0;
int val[2] = {0, 0};

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(p2, OUTPUT);
  pinMode(p3, OUTPUT);
  pinMode(p4, OUTPUT);
  pinMode(p5, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:

  if(Serial.available() > 0)
  {
    val[i] = Serial.read();
    i++;
```

```
if(i == 2)
{
  if(val[0] == 1)
  {
    while(Serial.available() == NULL)
    {
      digitalWrite(p2, LOW);
      digitalWrite(p3, HIGH);
      digitalWrite(p4, HIGH);
      digitalWrite(p5, LOW);
      delay(5);
      digitalWrite(p2, HIGH);
      digitalWrite(p3, HIGH);
      digitalWrite(p4, HIGH);
      digitalWrite(p5, HIGH);
      delay(10);
    }
  }
  else if(val[0] == 2)
  {
    while(Serial.available() == NULL)
    {
      digitalWrite(p2, LOW);
      digitalWrite(p3, HIGH);
      digitalWrite(p4, HIGH);
      digitalWrite(p5, HIGH);
      delay(5);
      digitalWrite(p2, HIGH);
      digitalWrite(p3, HIGH);
```

```
    digitalWrite(p4, HIGH);
    digitalWrite(p5, HIGH);
    delay(10);
  }
}
else if(val[0] == 3)
{
  while(Serial.available() == NULL)
  {
    digitalWrite(p2, HIGH);
    digitalWrite(p3, HIGH);
    digitalWrite(p4, HIGH);
    digitalWrite(p5, LOW);
    delay(5);
    digitalWrite(p2, HIGH);
    digitalWrite(p3, HIGH);
    digitalWrite(p4, HIGH);
    digitalWrite(p5, HIGH);
    delay(10);
  }
}
else if(val[0] == 4)
{
  while(Serial.available() == NULL)
  {
    digitalWrite(p2, HIGH);
    digitalWrite(p3, HIGH);
    digitalWrite(p4, HIGH);
    digitalWrite(p5, HIGH);
```

```
        delay(5);
        digitalWrite(p2, HIGH);
        digitalWrite(p3, HIGH);
        digitalWrite(p4, HIGH);
        digitalWrite(p5, HIGH);
        delay(10);
       }
      }

      i = 0;
    }
  }

}
```