DTMF Decoder

DSP Lab

Group - 6
Bhoomi Kanara
Kairav Pithadia
Maharsh Suryawala
Ronak Prajapati

Introduction

DTMF stands for Dual Tone Multiple Frequency, it's a telecommunication signaling system using voice-frequency band over telephone lines. It is also known as Touch-Tone. Every button in the keyboard, when pressed will generate a superimposed sine wave of multiple frequencies. Each number will possess different tone based on the frequency it generates.This Touch-Tone service for telephone keyboard has gradually replaced the use of Rotary dial and has become the industry standard for Landline and Mobile devices. Originally DTMF was decoded using filter banks but now Digital Signal Processing is the predominant technology.

Working

DTMF keypads are laid out on a 4x4 matrix, in which each row represents low frequency and each column represents high frequency. With DTMF, each key pressed on a phone generates two tones of specific frequencies. One tone is generated from a high-frequency group of tones, while the other is from a low-frequency group. DTMF systems use eight different frequency signals transmitted in pairs to represent 16 different numbers, letters and symbols.

Advantages

The advantages of DTMF include:

- By using this we can get a quick response
- It is not expensive to construct.
- High reliability and fast efficient
- By using a single key we control six devices.
- By using this one can control the home appliances wirelessly
- The power consumption will be reduced and power efficiency will be Increased.

Applications

The applications of DTMF are:

- The DTMF is used to identify the dialed numbers in the telephone switching centers
- These are used to operate the remote transmitters in the terrestrial stations
- DTMF is applicable in IVR systems,home automation, call centers, security systems as well as industrial application.

Implementation

We've implemented DTMF using software as well as hardware. In software, we've used 'The Goertzel Algorithm'. In the other hand, we've implemented basic filters to simulate and analyze logic of how DTMF works. Software part is implemented in MATLAB and hardware is on breadboard using resistors and capacitors.

Software

The Goertzel Algorithm

- Given a signal x[n] with corresponding 8 point DFT X[k], what is the most efficient way, in terms of total multiplications required, to compute X[k] from x[n]?

  Using a radix-2 FFT algorithm isn't the best choice as a judicious application of the canonical DFT equation is enough to beat it. Computing X[k] where k = 5 (Suppose), requires roughly 8log2(8)= 24 complex multiplications when employing a radix 2 FFT algorithm, since X[k] for k = 1,2,3,....,7 must be computed first.

  The Goertzel algorithm is a technique in digital signal processing (DSP) that provides a means for efficient evaluation of individual terms of the discrete Fourier transform (DFT), thus making it useful in certain practical applications, such as recognition of DTMF tones produced by the buttons pushed on a telephone keypad. In other words, the Goertzel algorithm is an efficient method in terms of multiplications for computing X[k] for a given k. When the required DFT coefficients are less than Nlog2(N), the Goertzel's algorithm beats Radix-2 FFT algorithm in terms of multiplications.

- ## First Stage
  - The first stage calculates an intermediate sequence, s[n]:
    $s[n] = x[n] + 2\cos(\omega_0)s[n-1] - s[n-2]$

The first filter stage can be observed to be a second-order IIR filter with a direct-form structure. This particular structure has the property that its internal state variables equal the past output values from that stage. Input values x [n] for $n < 0$ are presumed all equal to 0. To establish the initial filter state so that evaluation can begin at sample x [0], the filter states are assigned initial values, **$s[-2] = s[-1] = 0$**

- ## Second Stage
  - The second stage applies the following filter to s[n], producing output sequence y[n]:
    $Y[n] = s[n] - \exp(-j*\omega_0)s[n-1]$

The second-stage filter can be observed to be a FIR filter, since its calculations do not use any of its past outputs.

- ## Properties Of First Filter
  - Applying z transform in order to study the properties of the filter cascade.
  - The z transform of the first filter stage equation,

$$\frac{S(z)}{X(z)} = \frac{1}{1 - 2\cos(\omega_0)z^{-1} + z^{-2}}$$
$$= \frac{1}{(1 - e^{+j\omega_0} z^{-1})(1 - e^{-j\omega_0} z^{-1})}.$$

- ## Properties Of Second Filter
  - The Z transform of the second filter stage is given by,

$$\frac{Y(z)}{S(z)} = 1 - e^{-j\omega_0} z^{-1}.$$

- ## Properties Of the cascade of two filters
  - The combined transfer function of the cascade of the two filter stages is then,

$$\frac{S(z)}{X(z)} \frac{Y(z)}{S(z)} = \frac{Y(z)}{X(z)} = \frac{(1 - e^{-j\omega_0} z^{-1})}{(1 - e^{+j\omega_0} z^{-1})(1 - e^{-j\omega_0} z^{-1})}$$
$$= \frac{1}{1 - e^{+j\omega_0} z^{-1}}.$$

- ## Equivalent Time Domain Sequence
  - This can be transformed back to an equivalent time-domain sequence, and the terms unrolled back to the first input term at index n = 0 ,

$$
\begin{aligned}
y[n] &= x[n] + e^{+j\omega_0} y[n-1] \\
&= \sum_{k=-\infty}^{n} x[k] e^{+j\omega_0 (n-k)} \\
&= e^{j\omega_0 n} \sum_{k=0}^{n} x[k] e^{-j\omega_0 k} \qquad \text{since } \forall k < 0, x[k] = 0.
\end{aligned}
$$

- ## DFT Computations
  - For the important case of computing a DFT term, the following special restrictions are applied. The filtering terminates at index n = N, where N is the number of terms in the input sequence of the DFT. The frequencies chosen for the Goertzel analysis are restricted to the special form → ω_0 = (2*pi*k)/N. where, k = {0,1,2,3,...N-1}

- ## Making the substitutions
  - The equation becomes,

$$
y[N] = \sum_{n=0}^{N} x[n] e^{-j2\pi \frac{nk}{N}}.
$$

Where, $e^{+j2\pi k} = 1$

- ## Observations

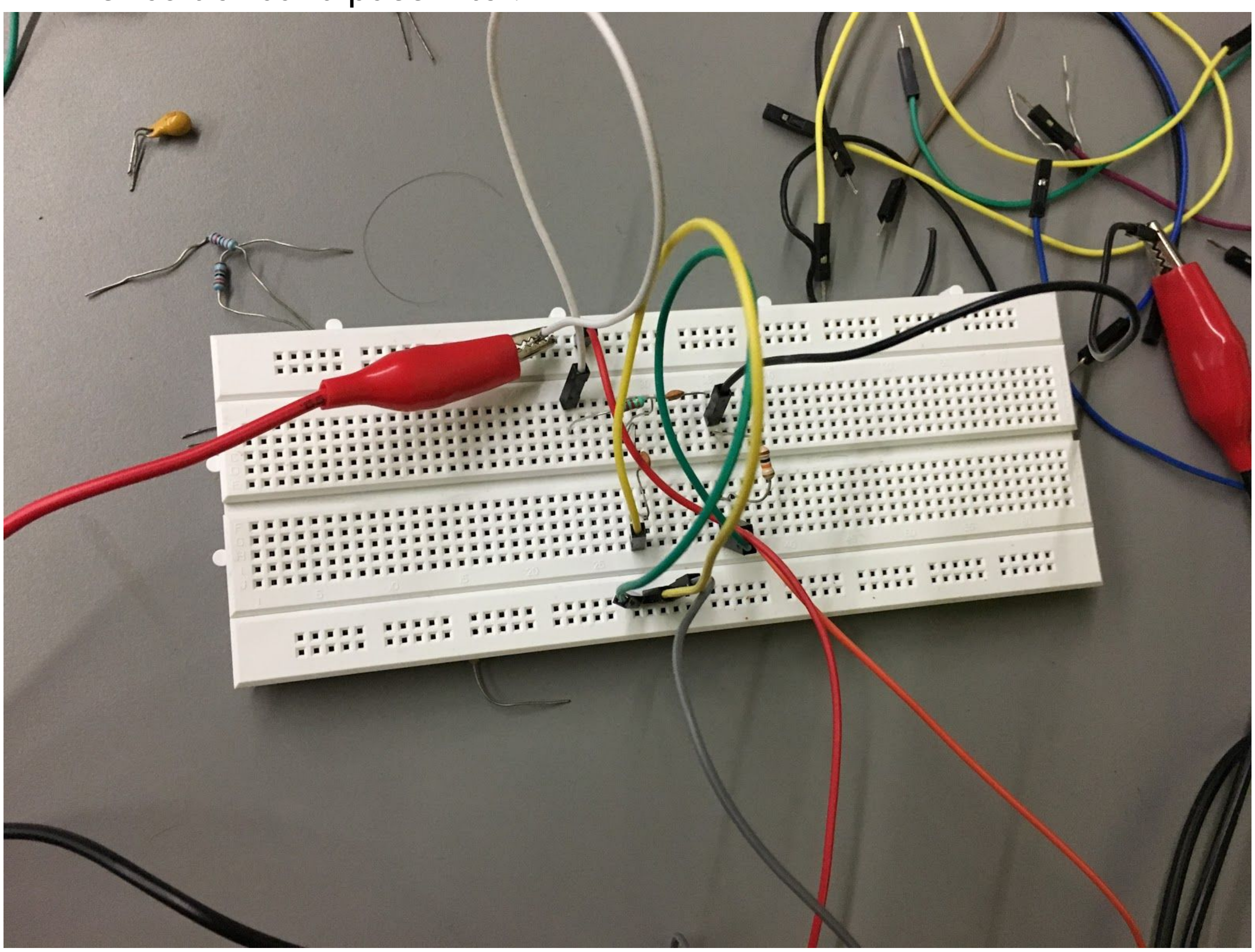
- Since we're only evaluating the output of this filter at y_k[N], the multiplier– W^k_N need only be used at time n = N.
- With this in mind, the algorithm requires N real multiplications and a single complex multiplication to compute X[k] for a given k.

Hardware

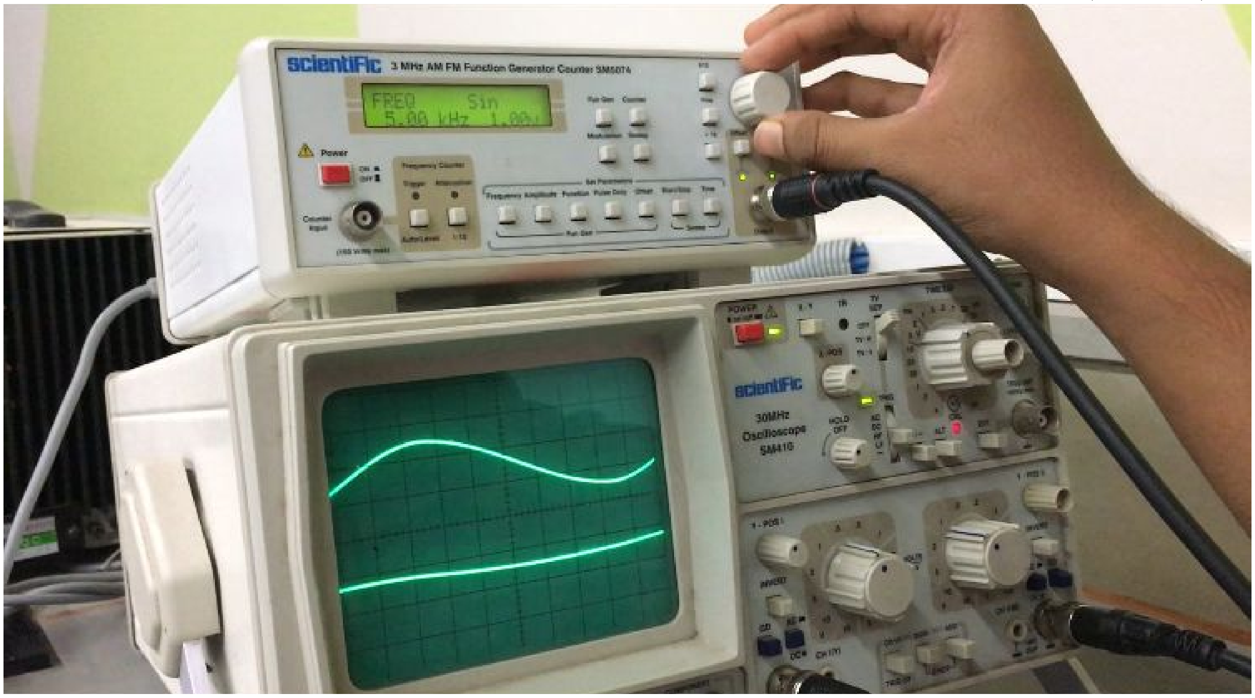
We've used resistors and capacitors to create filters accordingly. Lowpass, Band pass and high pass filters can be made to implement logic of how DTMF works.

Images below indicates our experiment of implementing band pass filter using two capacitors of value 1nF each and two resistors of 10k ohm and 2.2k ohm respectively. It makes our range of frequencies to be passed by band pass from 15.9 kHz to 72.4 kHz.

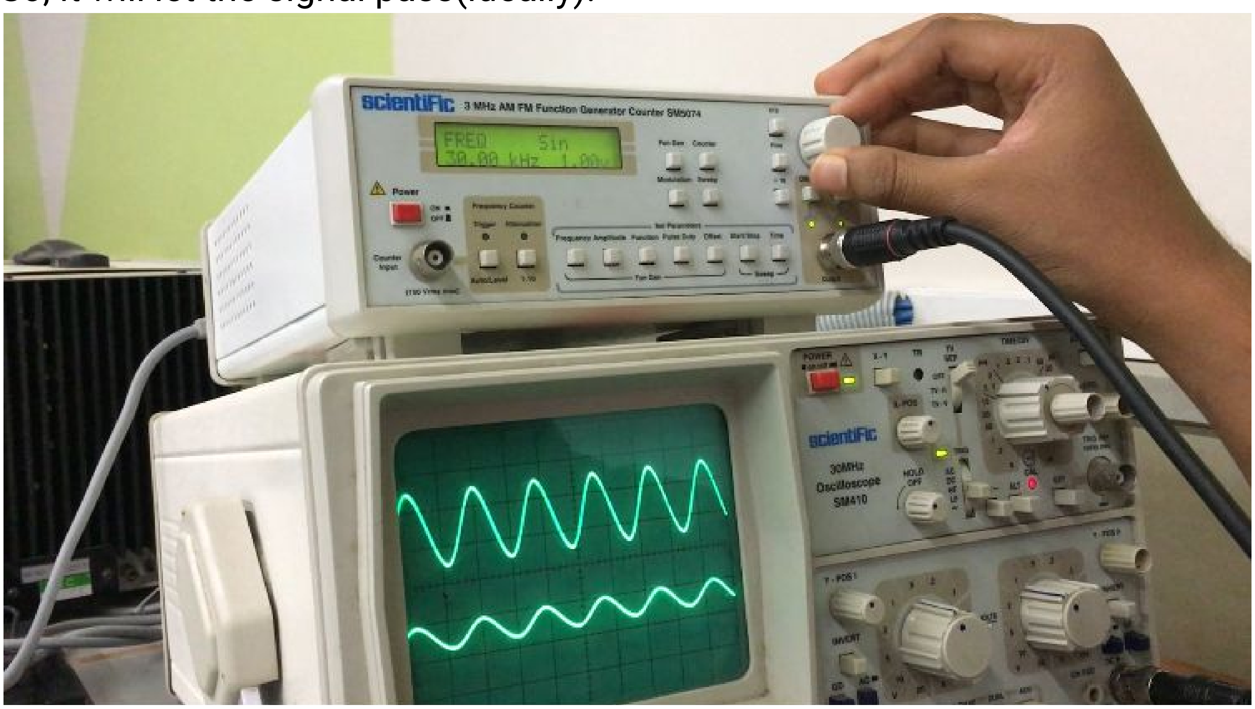
Circuit of band pass filter:



When passing 5 kHz input frequency, it is out of the range of band pass filter. So, it won’t let the signal pass with that much low frequency(ideally).



When passing 30 kHz input frequency, it is in the range of band pass filter. So, it will let the signal pass(ideally).



When passing 300 kHz input frequency, it is out of the range of band pass filter. So, it won’t let the signal pass(ideally).