Subject:	**Digital Signal Processing CEN-4021**
Department:	**Computer Engineering**
Semester Project:	**Dual Tone Frequency Multi Frequency (DTMF)**
Student:	**Muhammad Sami Uddin Rafay**
Batch:	**Fall – 2018**
Roll No.:	**F18604013**
Semester:	**6th**
Submitted to:	**Dr. Qasim Mehmood Chaudary**

History:

Before DTMF is created Telephone network used a system called DECATIC. This system was used extensively in telephone networks to dial numbers, it was a very useful system but limited to the local exchange connections requiring an operator to connect long distance calls. In the late years 1950, DTMF was been developed for the purpose of allowing tone signals to dial long distance numbers which could potentially be dialed not only via standard wired networks but also via radio links. The version of DTMF used for telephone tone dialing is known by the trademark term touchstone.

Introduction:

The DTMF is the popular signaling method between telephones and switching centers. DTMF signals are superposition of two sine waves with different frequencies.

DTMF Generate and Receive:

DTMF tone generation and detection can be with the help of PYTHON. Fast Fourier Transform package under SciPy and NumPy libraries used to detect DTMF Tones. DTMF has enabled the long distance signaling of dialed numbers in voice frequency range over telephone lines. IVR systems, Security and Call centers use it widely.

Algorithm:

1. Generate a DTMF signal with 8 different combinations of frequency.
2. DTMF signal is applied to the decoder.
3. FFT is applied to each signal.
4. Compare the FFT signal with lookup table.
5. Get the information of which button is pressed.
6. Connect the dialer to the receiver address obtained through decoder.

Working:

1. When the key is pressed, the tone of the column and the tone of the row are generated.
2. There are 16 different tones (but in my project I generated only 12 Tones).
3. Uses two tones to represent each key on Touch pad.
4. At the receiver, the tone frequencies are detected and the number is decoded.
5. The DFT algorithm can be used to detect the frequencies.

Use Cases:

- ✓ Home Automation System
- ✓ Cell Phone Controlled Vehicles
- ✓ Auto Emergency Call (Security)

Tools and Advantages:

- ➢ FFT (Fast Fourier Transform) is the mathematical tool that can be used to calculate the frequency component in given signal.

- DTMF Technology was first introduced in telephone system in 1963 but now a days it is used in different field of life.
- It reduces the waiting time, response time and increase the efficiency.
- The Dialer is now connected with receiver without involvement of third-party person (Telephone Exchange)
- Reducing the theft rate, burglary and Use DTMF in Security places (Military, Banks etc.).

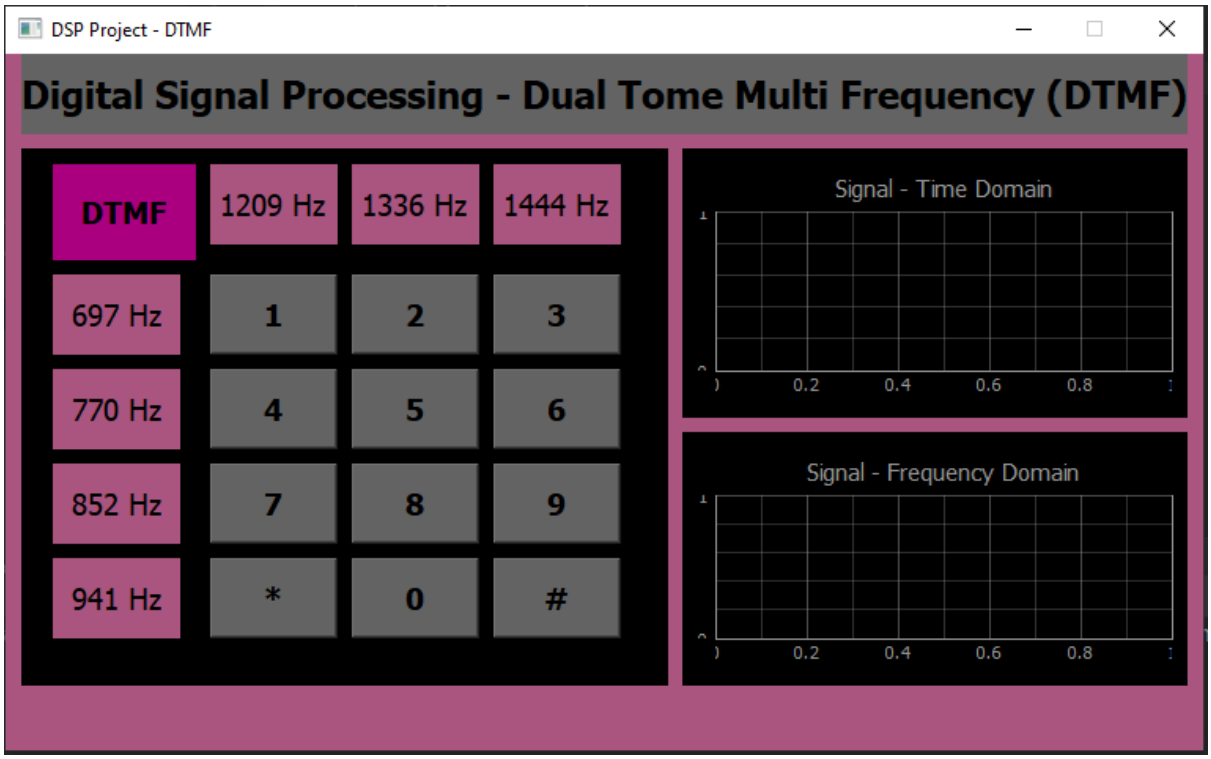
Improvement:

- ✓ Automatic calling to security institute by watching Burglary or guns or any inappropriate scene in camera using artificial intelligence (Computer Vision).
- ✓ Develop a smart phone application and interface the robot vehicles to control it with phone.
- ✓ Self-driving vehicles contain DTMF decoder to perform different tasks.

Software Tools and Libraries

- PyCharm
- Python
- NumPy
- Qt Designer
- Math
- Sound device
- Sys

Graphical User Interface



Implementation:

(Code)

```
# importing libraries
import sounddevice as sd
import pyqtgraph as pg
from math import pi
from PyQt5.uic import loadUi
import sys
import numpy as np
from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox

# Sampling Frequency
Fs = 2000;
# Sampling Time
Ts = 1/Fs
# Time axis
t = np.arange(-0.2, 0.2, Ts)
# Dual Tone Multi Frequencies
f1 = 1209
f2 = 1336
f3 = 1444
f4 = 697
f5 = 770
f6 = 852
f7 = 941

Time_plot = pg.GraphicsLayoutWidget()
Freq_plot = pg.GraphicsLayoutWidget()

class DTMF(QMainWindow):
    def __init__(self):
```

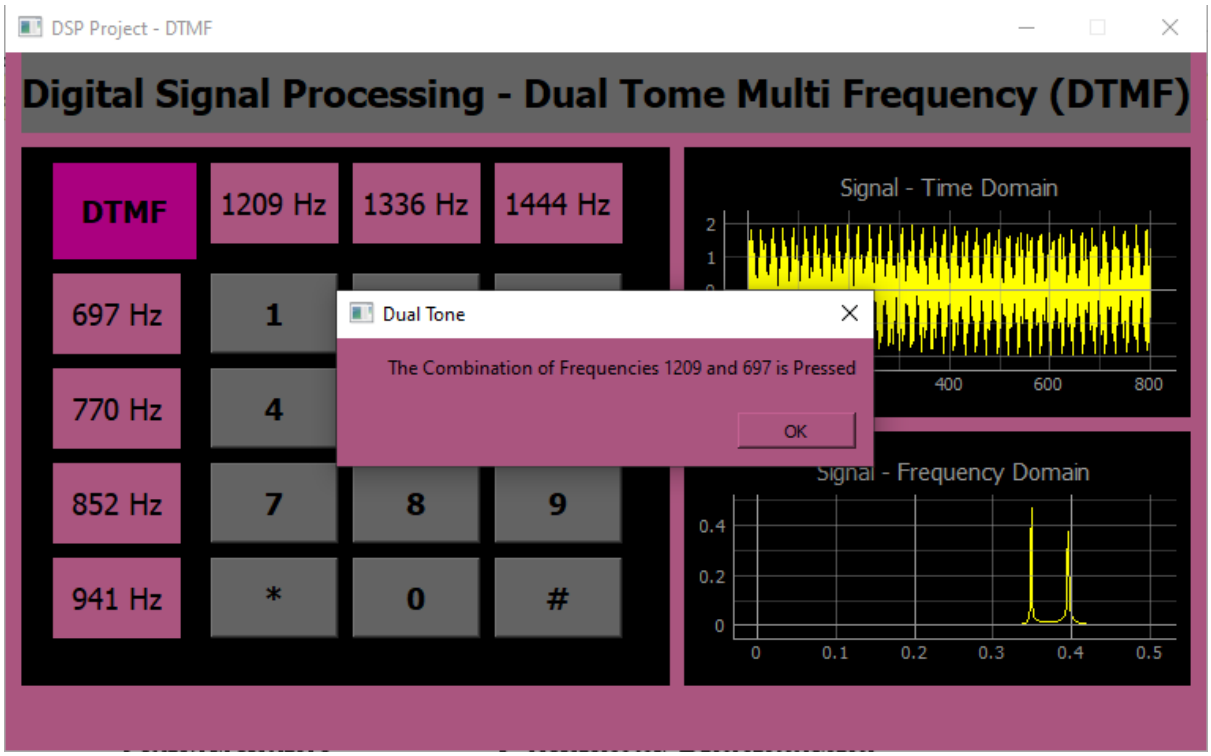
```
        super(DTMF, self).__init__()
        loadUi('DSP Project - DTMF.ui', self)
        self.setFixedSize(761, 442)
        self.gridLayout.addWidget(Time_plot)
        self.gridLayout_2.addWidget(Freq_plot)
        self.Time = Time_plot.addPlot(title="Signal - Time Domain")
        self.Time.showGrid(x=True, y=True, alpha=1)
        self.Frequency = Freq_plot.addPlot(title="Signal - Frequency
Domain")
        self.Frequency.showGrid(x=True, y=True, alpha=1)
        self.Time_Curve = self.Time.plot(pen='y')
        self.Freq_Curve = self.Frequency.plot(pen='y')
        self.Freq_Curve.setFftMode(True)

        self.pushButton.clicked.connect(self.Dial_1)
        self.pushButton_5.clicked.connect(self.Dial_2)
        self.pushButton_7.clicked.connect(self.Dial_3)
        self.pushButton_8.clicked.connect(self.Dial_4)
        self.pushButton_4.clicked.connect(self.Dial_5)
        self.pushButton_6.clicked.connect(self.Dial_6)
        self.pushButton_2.clicked.connect(self.Dial_7)
        self.pushButton_3.clicked.connect(self.Dial_8)
        self.pushButton_10.clicked.connect(self.Dial_9)
        self.pushButton_9.clicked.connect(self.Star)
        self.pushButton_11.clicked.connect(self.Zero)
        self.pushButton_12.clicked.connect(self.Hash)

    def Dial_1(self):
        signal = np.sin(2 * pi * f1 * t)
        signal1 = np.sin(2 * pi * f4 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Freq_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f1) + " and " + str(f4)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Dial_2(self):
        signal = np.sin(2 * pi * f2 * t)
        signal1 = np.sin(2 * pi * f4 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f2) + " and " + str(f4)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Dial_3(self):
        signal = np.sin(2 * pi * f3 * t)
        signal1 = np.sin(2 * pi * f4 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f3) + " and " + str(f4)
```

```
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Dial_4(self):
        signal = np.sin(2 * pi * f1 * t)
        signal1 = np.sin(2 * pi * f5 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f1) + " and " + str(f5)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Dial_5(self):
        signal = np.sin(2 * pi * f2 * t)
        signal1 = np.sin(2 * pi * f5 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f2) + " and " + str(f5)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Dial_6(self):
        signal = np.sin(2 * pi * f3 * t)
        signal1 = np.sin(2 * pi * f5 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f3) + " and " + str(f5)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Dial_7(self):
        signal = np.sin(2 * pi * f1 * t)
        signal1 = np.sin(2 * pi * f6 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f1) + " and " + str(f6)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Dial_8(self):
        signal = np.sin(2 * pi * f2 * t)
        signal1 = np.sin(2 * pi * f6 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f2) + " and " + str(f6)
+ " is Pressed"
```

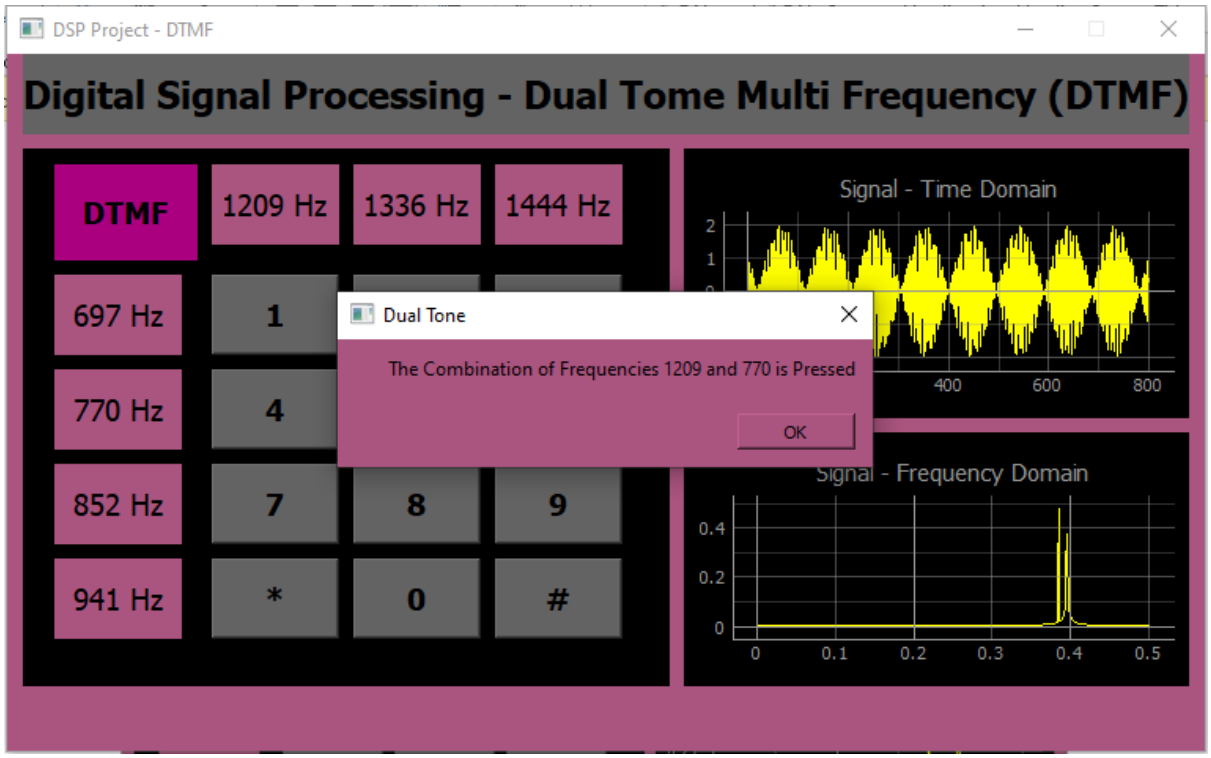
```
        QMessageBox.about(self, "Dual Tone", S)

    def Dial_9(self):
        signal = np.sin(2 * pi * f3 * t)
        signal1 = np.sin(2 * pi * f6 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f3) + " and " + str(f6)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Star(self):
        signal = np.sin(2 * pi * f1 * t)
        signal1 = np.sin(2 * pi * f7 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f1) + " and " + str(f7)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Zero(self):
        signal = np.sin(2 * pi * f2 * t)
        signal1 = np.sin(2 * pi * f7 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f2) + " and " + str(f7)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

    def Hash(self):
        signal = np.sin(2 * pi * f3 * t)
        signal1 = np.sin(2 * pi * f7 * t)
        signalx = signal + signal1
        self.Time_Curve.clear()
        self.Time_Curve.setData(signalx)
        self.Freq_Curve.setData(signalx)
        sd.play(signalx, Fs)
        S = "The Combination of Frequencies " + str(f3) + " and " + str(f7)
+ " is Pressed"
        QMessageBox.about(self, "Dual Tone", S)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = DTMF()
    window.setWindowTitle('DSP Project - DTMF')
    window.show()
    sys.exit(app.exec_())
```

Results

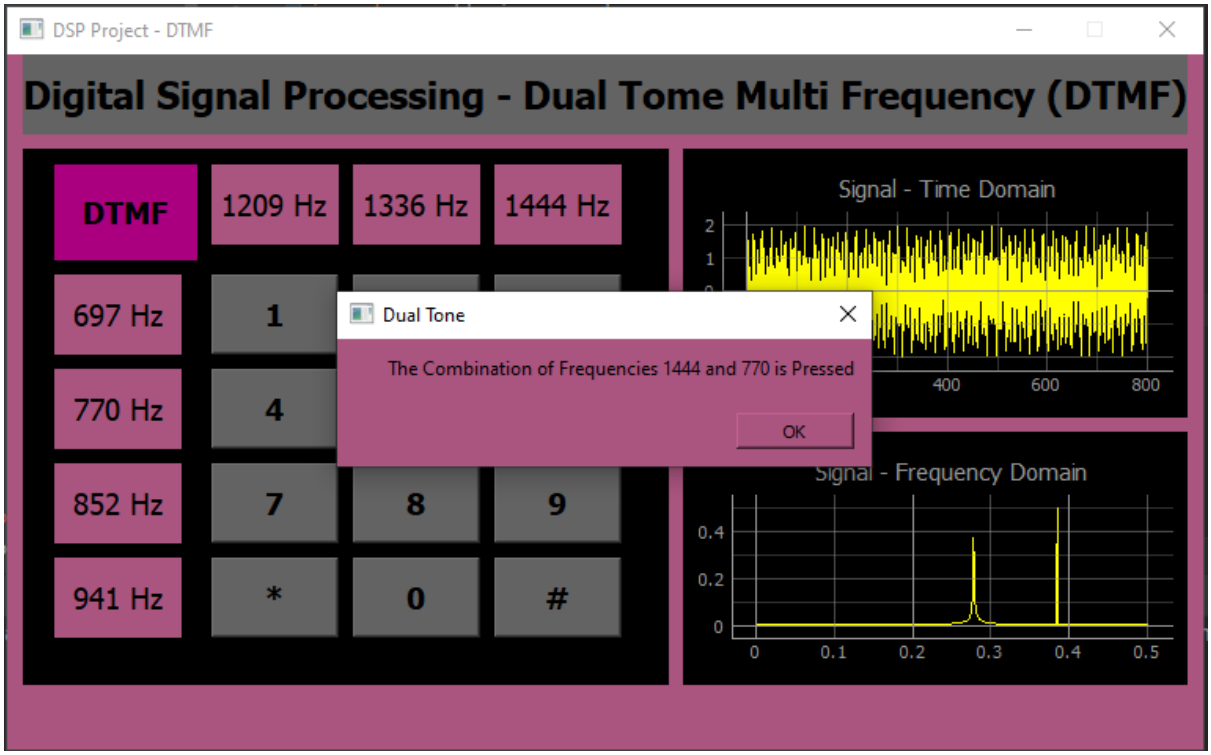
Pressed 1:



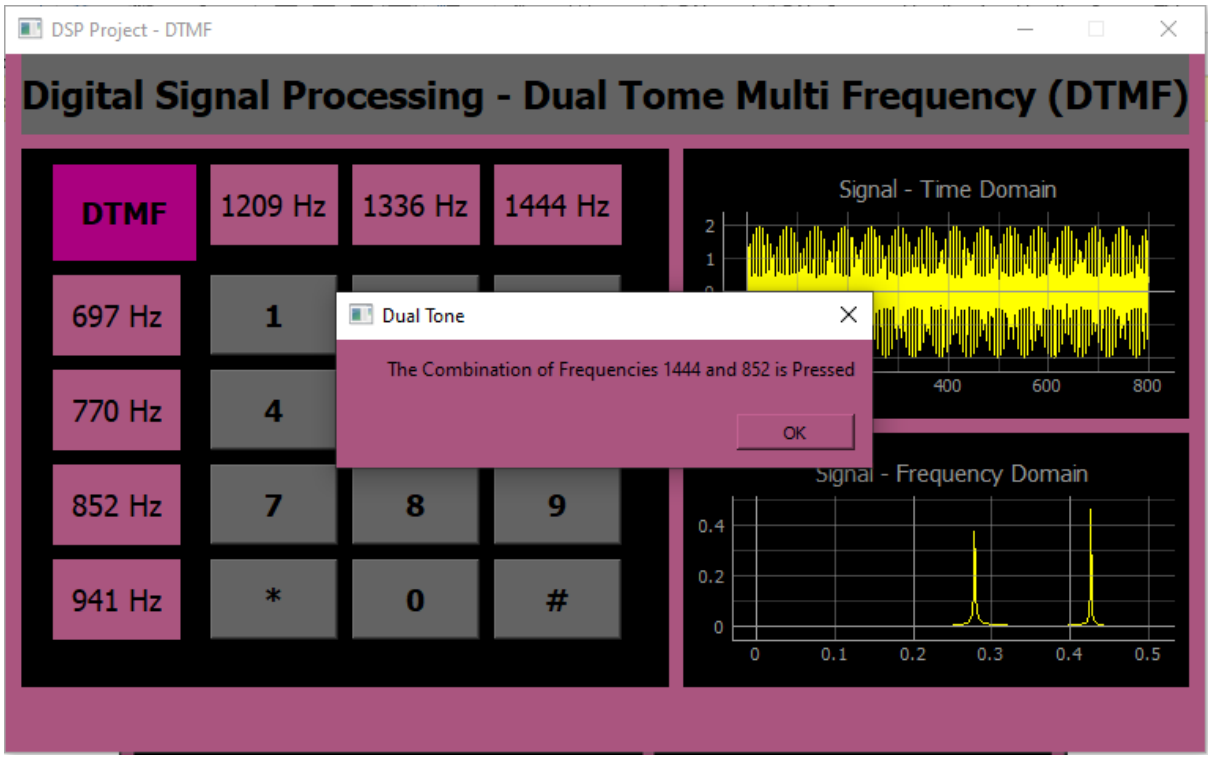
Pressed 4:



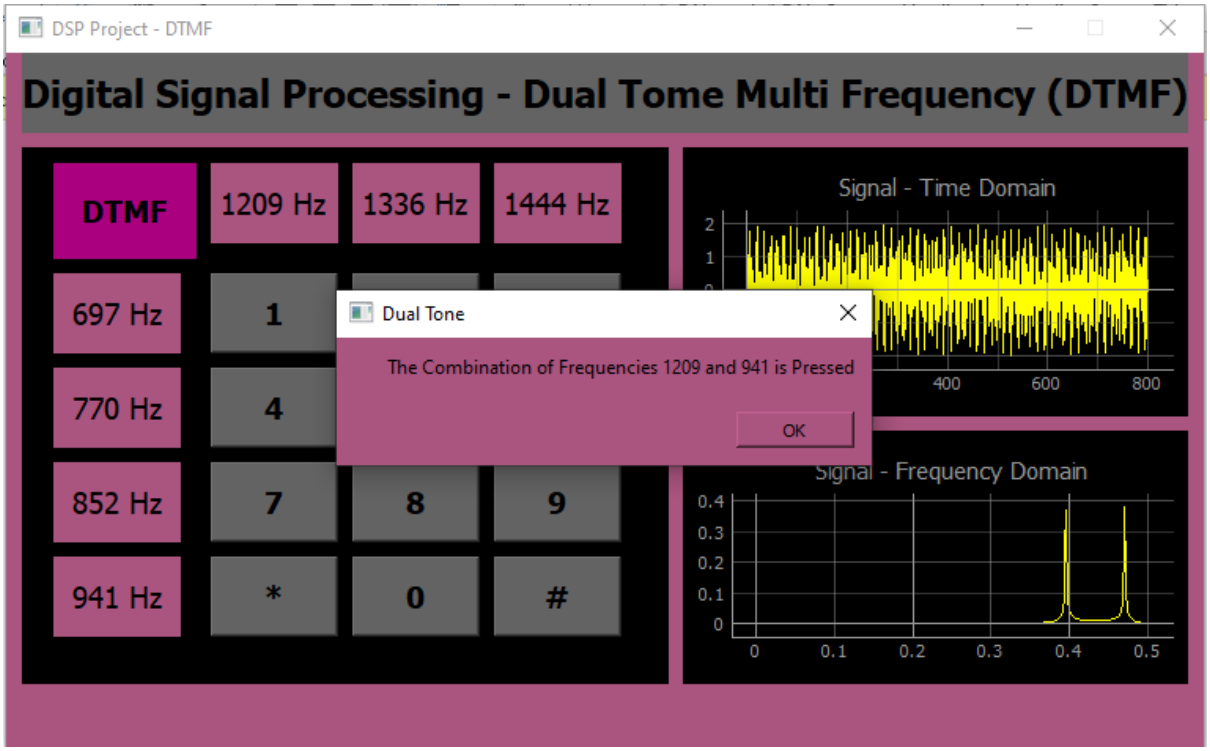
Pressed 6:



Pressed 9:



Press Star:



Demo Video:

Please watch the video on following link given below: -

(YouTube Link)

https://youtu.be/gJo3JTAQnyU

Observations

If we implement the DTMF with development boards like Arduino, MSP430, PIC Microcontrollers or STM32 so that will be the plus point. The system can be more perfect with the approach of Embedded Machine Learning and Edge Computing.

The End!