****

**HOME AUTOMATION USING DTMF DECODER**

**Arduino-based Hardware Working model with Simulation through MATLAB**

**SIGNALS AND SYSTEMS**

**SUBJECT CODE: BECE202L**

**Slot: G1 + TG1**

**LOKESH R (23BEC1009)**

**GAUTHAM R (23BEC1069)**

**MATHESH (23BEC1152)**

**Submitted to**

**Prof Sivakumar S**

**CONTENTS**

|  |  |
| --- | --- |
| PART OF THE REPORT | PAGE NO |
| INTRODUCTION | **3** |
| COMPONENTS USED | **3** |
| THEORY | **3** |
| BLOCK DIAGRAM & CIRCUIT DIAGRAM | **5** |
| CODE USED FOR ARDUINO | **5** |
| MATLAB SIMULATIONS |  |
| 1. DTMF Signal Generation and Frequency Spectrum Plotting | **7** |
| 1. Enhanced DTMF Signal Generation, Decoding, and Frequency Detection | **11** |
| C) Interactive keyboard input and DTMF signal (sound) generation. | **18** |
| D) Interactive GUI-Based DTMF Keypad Code | **23** |
| WHAT WE LEARNED | **24** |
| FUTURE SCOPE | **25** |
| CONCLUSION | **25** |
| ATTACHMENTS | **25** |

**INTRODUCTION**

The objective of this project is to design and implement a home automation system utilizing a Dual-Tone Multi-Frequency (DTMF) decoder. This project enables users to remotely control home appliances using DTMF tones generated by a mobile phone. The system will be further analysed using MATLAB simulations at the end to validate the processing and highlight the functionality of the DTMF decoder.

**COMPONENTS USED**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **COMPONENT** | **SPECIFICATION** |
| 1 | Arduino UNO | - |
| 2 | DTMF Decoder Module | MT8870 |
| 3 | 2-Channel Relay Module | SPDT or DPDT |
| 4 | Fan | 12V DC / 220V AC |
| 5 | LED Strip | RGB, 12V |
| 6 | Mobile Phone | - |
| 7 | Connecting Wires | - |
| 8 | Power Supply | - |

**THEORY**

DTMF (Dual Tone Multi-Frequency) is the signal used by touch-tone telephones to represent numbers on the keypad. When a button is pressed, two simultaneous tones are generated, one from a high-frequency group and one from a low-frequency group. These tones uniquely identify the button pressed, and this project utilizes these signals to control appliances.

**DTMF A (Hardware)**

The project consists of several key hardware components that interact to decode DTMF tones and control appliances:

##### **1. DTMF Decoder (MT8870)**

##### The MT8870 DTMF decoder IC is at the core of the system. When a user presses a button on their mobile phone, it generates a DTMF tone that can be received by the phone's microphone, transmitted to the system, and decoded by the MT8870. The IC analyzes the incoming tone signal and outputs a binary representation of the pressed key through four output pins (Q1, Q2, Q3, Q4).

##### **2. Microcontroller (Arduino)**

The microcontroller (such as Arduino) reads the binary signals from the MT8870 and makes decisions based on the input. For example, pressing key ‘1’ may switch on a particular appliance, while pressing ‘2’ could switch it off.

##### **3. Relays**

Relays act as the interface between the microcontroller and the home appliances. A relay is an electromechanical switch that allows the microcontroller, operating at a low current, to control high-power devices like lights, fans, or other household appliances.

##### **4. Circuit Design Overview**

The system receives DTMF signals through a phone's audio port. These signals are fed into the DTMF decoder, which converts them into digital binary signals. The microcontroller processes the binary input and decides which relay to activate. Relays are connected to various home appliances, and based on the received DTMF tones, the corresponding appliances are switched on or off.

**DTMF B (Software)**

The software component of this project includes the microcontroller code that interfaces with the DTMF decoder, processes the binary input, and controls the relays. Additionally, MATLAB is used to simulate the DTMF signal generation and decoding to validate the system’s functionality.

##### **1. Microcontroller Code**

The microcontroller software is responsible for:

* Continuously reading inputs from the DTMF decoder.
* Decoding the binary signals from the MT8870 to determine which button was pressed.
* Controlling the relays based on the decoded input to operate appliances.

**2. MATLAB Simulation of DTMF**

In addition to the hardware implementation, the DTMF decoding process is simulated using MATLAB. The simulation involves generating DTMF tones for different keypad buttons and analysing the frequency content to ensure that the correct signal processing is taking place.

* **DTMF Tone Generation**: MATLAB generates the tones based on two sinusoidal signals, one from the low-frequency band and one from the high-frequency band.
* **Frequency Analysis**: The generated DTMF signals are analysed using the Fast Fourier Transform (FFT) to extract and verify the correct frequencies for each key press. MATLAB plots the frequency spectrum, highlighting the two key frequencies associated with each pressed key.

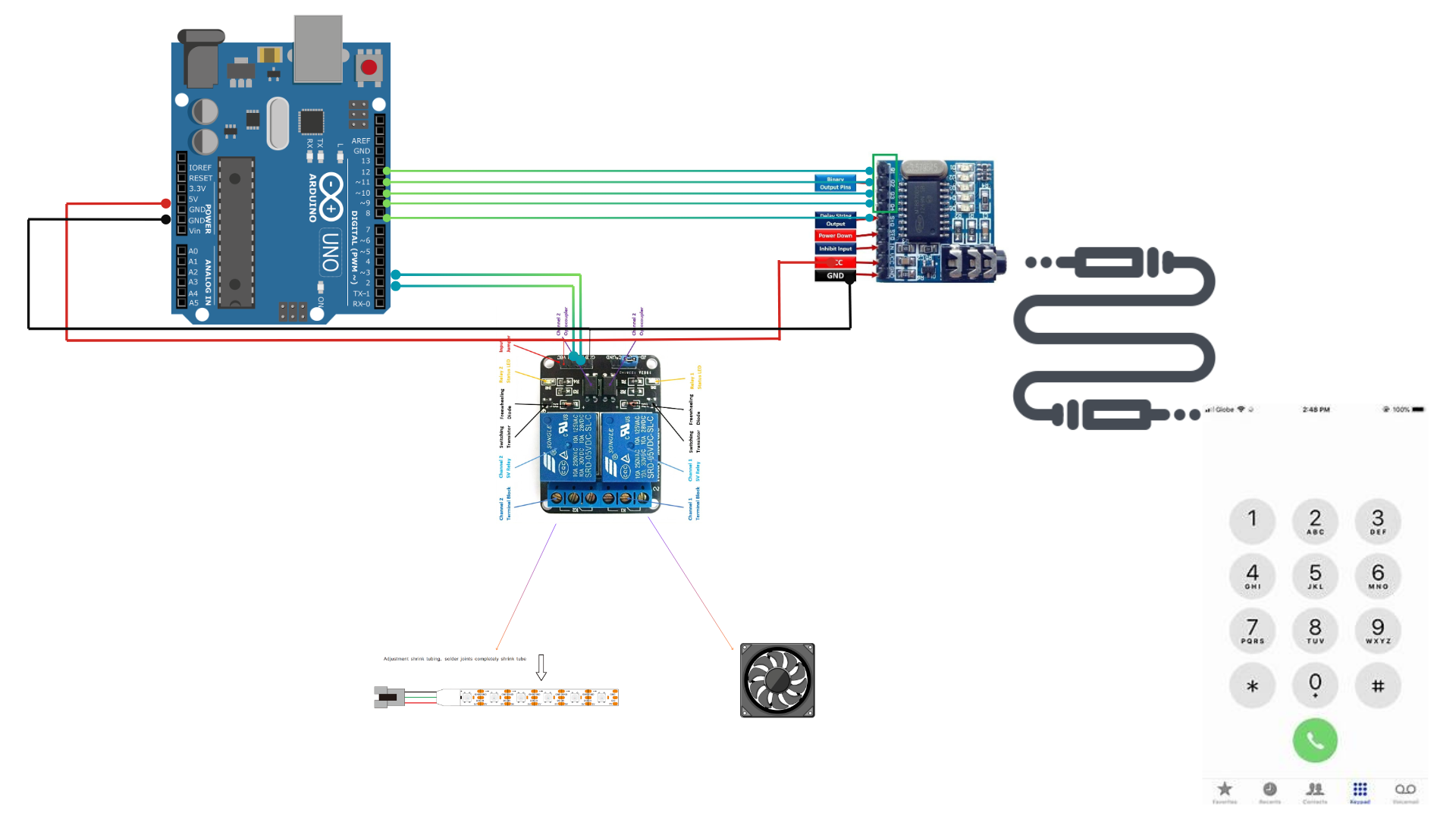
### **Basics of Fast Fourier Transform (FFT)**

The **Fast Fourier Transform (FFT)** is an algorithm used to efficiently compute the **Discrete Fourier Transform (DFT)** of a sequence, which converts a signal from its time domain to its frequency domain. This transformation helps in analysing the frequency components of signals, which is particularly useful in signal processing applications like DTMF decoding, audio analysis, and communications.

**BLOCK DIAGRAM**

Mobile Phone -> DTMF Decoder -> Arduino -> Relay Module -> Fan / LED Strip

**CIRCUIT DIAGRAM**



**CODE USED FOR ARDUINO**

***const int relay1 = 2;***

***const int relay2 = 3;***

***void setup() {***

***Serial.begin(9600);***

***pinMode(8, INPUT);***

***pinMode(9, INPUT);***

***pinMode(10, INPUT);***

***pinMode(11, INPUT);***

***pinMode(12, INPUT);***

***pinMode(relay1, OUTPUT);***

***pinMode(relay2, OUTPUT);***

***digitalWrite(relay1, HIGH);***

***digitalWrite(relay2, HIGH);***

***}***

***void loop() {***

***bool signal = digitalRead(8***

***if (signal == HIGH) {***

***delay(50);***

***Serial.print("Q1: ");***

***Serial.print(digitalRead(12));***

***Serial.print(", Q2: ");***

***Serial.print(digitalRead(11));***

***Serial.print(", Q3: ");***

***Serial.print(digitalRead(10));***

***Serial.print(", Q4: ");***

***Serial.println(digitalRead(9));***

***uint8\_t number\_pressed = (digitalRead(12) << 0) |***

***(digitalRead(11) << 1) |***

***(digitalRead(10) << 2) |***

***(digitalRead(9) << 3);***

***Serial.print("Detected Button (Binary): ");***

***Serial.println(number\_pressed, BIN);***

***switch (number\_pressed) {***

***case 0x01:***

***Serial.println("Button Pressed = 1, Appliance 1 ON");***

***digitalWrite(relay1, LOW); // Turn ON***

***break;***

***case 0x02:***

***Serial.println("Button Pressed = 2, Appliance 1 OFF");***

***digitalWrite(relay1, HIGH); // Turn OFF***

***break;***

***case 0x03:***

***Serial.println("Button Pressed = 3, Appliance 2 ON");***

***digitalWrite(relay2, LOW); // Turn ON***

***break;***

***case 0x04:***

***Serial.println("Button Pressed = 4, Appliance 2 OFF");***

***digitalWrite(relay2, HIGH); // Turn OFF***

***break;***

***default:***

***Serial.println("Unknown button or misread");***

***break;***

***}***

***delay(300);***

***}***

***delay(100);}***

**MATLAB SIMULATION**

1. **DTMF Signal Generation and Frequency Spectrum Plotting**

* This code generates DTMF signals for each key on the keypad (0-9, \*, #) and plots the frequency spectrum for each signal in a separate figure.
* Dual-Tone Multi-Frequency (DTMF) signalling involves generating two simultaneous sinusoidal tones for each key press—one from a low-frequency group (row frequency) and one from a high-frequency group (column frequency). The combination uniquely identifies each key. The Fast Fourier Transform (FFT) is used to compute the frequency spectrum to visualize these tones.
* For each key, the code creates a new figure showing the frequency spectrum and highlights the low and high frequencies used to generate the DTMF tone.

***fs = 8000;***

***t = 0:1/fs:0.5;***

***key\_freqs = {***

***'1', 697, 1209;***

***'2', 697, 1336;***

***'3', 697, 1477;***

***'4', 770, 1209;***

***'5', 770, 1336;***

***'6', 770, 1477;***

***'7', 852, 1209;***

***'8', 852, 1336;***

***'9', 852, 1477;***

***'0', 941, 1336;***

***'\*', 941, 1209;***

***'#', 941, 1477;***

***};***

***for i = 1:size(key\_freqs, 1)***

***key = key\_freqs{i, 1};***

***f\_low = key\_freqs{i, 2};***

***f\_high = key\_freqs{i, 3};***

***dtmf\_signal = sin(2\*pi\*f\_low\*t) + sin(2\*pi\*f\_high\*t);***

***N = length(dtmf\_signal);***

***Y = fft(dtmf\_signal);***

***f = (0:N-1)\*(fs/N);***

***magnitude = abs(Y);***

***figure;***

***plot(f, magnitude);***

***title(['Frequency Spectrum of DTMF Signal for Key ' key]);***

***xlabel('Frequency (Hz)');***

***ylabel('Magnitude');***

***hold on;***

***xline(f\_low, '--r', 'Low Freq');***

***xline(f\_high, '--g', 'High Freq');***

***hold off;***

***end***

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white paper with black text

Description automatically generated

A white background with black text

Description automatically generated

1. **Enhanced DTMF Signal Generation, Decoding, and Frequency Detection**

* This code generates DTMF signals, plots their frequency spectrum in a multi-plot format, and includes logic to decode the generated DTMF tones back into the corresponding key.
* Like the previous code, DTMF signals are created for each key. A Hamming window is applied to the DTMF signal to reduce spectral leakage, followed by FFT with zero-padding for higher frequency resolution.
* The code extracts and identifies row and column frequencies using a threshold and tolerance for matching. It maps detected frequencies to the DTMF keypad layout and outputs the detected key.
* The code prints detected frequencies, matched row/column frequencies, and the corresponding detected key to the console. The frequency spectrum for each key is shown in a grid of subplots.

***fs = 8000;***

***t = 0:1/fs:0.5;***

***key\_freqs = {***

***'1', 697, 1209;***

***'2', 697, 1336;***

***'3', 697, 1477;***

***'4', 770, 1209;***

***'5', 770, 1336;***

***'6', 770, 1477;***

***'7', 852, 1209;***

***'8', 852, 1336;***

***'9', 852, 1477;***

***'0', 941, 1336;***

***'\*', 941, 1209;***

***'#', 941, 1477;***

***};***

***dtmf\_frequencies\_row = [697, 770, 852, 941];***

***dtmf\_frequencies\_col = [1209, 1336, 1477];***

***dtmf\_keypad = [***

***'1', '2', '3';***

***'4', '5', '6';***

***'7', '8', '9';***

***'\*', '0', '#'***

***];***

***figure;***

***for i = 1:size(key\_freqs, 1)***

***key = key\_freqs{i, 1};***

***f\_low = key\_freqs{i, 2};***

***f\_high = key\_freqs{i, 3};***

***dtmf\_signal = sin(2\*pi\*f\_low\*t) + sin(2\*pi\*f\_high\*t);***

***N = 2^nextpow2(length(dtmf\_signal));***

***windowed\_signal = dtmf\_signal .\* hamming(length(dtmf\_signal))';***

***Y = fft(windowed\_signal, N);***

***f = (0:N-1)\*(fs/N);***

***magnitude = abs(Y);***

***subplot(3, 4, i);***

***plot(f, magnitude);***

***title(['Key ' key]);***

***xlabel('Frequency (Hz)');***

***ylabel('Magnitude');***

***xlim([600 1600]);***

***xline(f\_low, '--r', 'Low Freq');***

***xline(f\_high, '--g', 'High Freq');***

***threshold = 5;***

***detected\_indices = find(magnitude > threshold);***

***detected\_freqs = f(detected\_indices);***

***fprintf('Key %s: Detected Frequencies: ', key);***

***disp(detected\_freqs);***

***tolerance = 10;***

***row\_freq = detected\_freqs(arrayfun(@(x) any(abs(x - dtmf\_frequencies\_row) < tolerance), detected\_freqs));***

***col\_freq = detected\_freqs(arrayfun(@(x) any(abs(x - dtmf\_frequencies\_col) < tolerance), detected\_freqs));***

***if ~isempty(row\_freq)***

***[~, closest\_row\_idx] = min(abs(dtmf\_frequencies\_row - row\_freq(1)));***

***row\_freq = dtmf\_frequencies\_row(closest\_row\_idx);***

***end***

***if ~isempty(col\_freq)***

***[~, closest\_col\_idx] = min(abs(dtmf\_frequencies\_col - col\_freq(1)));***

***col\_freq = dtmf\_frequencies\_col(closest\_col\_idx);***

***end***

***fprintf('Key %s: Matched Row Freq: %.2f Hz\n', key, row\_freq);***

***fprintf('Key %s: Matched Col Freq: %.2f Hz\n', key, col\_freq);***

***if ~isempty(row\_freq) && ~isempty(col\_freq)***

***row\_idx = find(dtmf\_frequencies\_row == row\_freq);***

***col\_idx = find(dtmf\_frequencies\_col == col\_freq);***

***detected\_key = dtmf\_keypad(row\_idx, col\_idx);***

***fprintf('Detected Key for input %s: %s\n', key, detected\_key);***

***else***

***fprintf('Failed to detect a valid key for input %s\n', key);***

***end***

***end***

***Key 1: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.6797 0.6816 0.6836 0.6855 0.6875 0.6895 0.6934 0.6953 0.6973 0.6992 1.2070 1.2090 1.2109 6.7891 6.7910 6.7930 7.3008 7.3027***

***Columns 19 through 26***

***7.3047 7.3066 7.3105 7.3125 7.3145 7.3164 7.3184 7.3203***

***Key 1: Matched Row Freq: 697.00 Hz***

***Key 1: Matched Col Freq: 1209.00 Hz***

***Detected Key for input 1: 1***

***Key 2: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.6816 0.6836 0.6855 0.6875 0.6895 0.6934 0.6953 0.6973 0.6992 1.3340 1.3359 1.3379 1.3398 6.6602 6.6621 6.6641 6.6660 7.3008***

***Columns 19 through 26***

***7.3027 7.3047 7.3066 7.3105 7.3125 7.3145 7.3164 7.3184***

***Key 2: Matched Row Freq: 697.00 Hz***

***Key 2: Matched Col Freq: 1336.00 Hz***

***Detected Key for input 2: 2***

***Key 3: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.6816 0.6836 0.6855 0.6875 0.6895 0.6934 0.6953 0.6973 0.6992 1.4727 1.4746 1.4766 1.4785 1.4805 1.4844 1.4863 1.4883 1.4902***

***Columns 19 through 36***

***1.4922 1.4941 1.4961 6.5039 6.5059 6.5078 6.5098 6.5117 6.5137 6.5156 6.5195 6.5215 6.5234 6.5254 6.5273 7.3008 7.3027 7.3047***

***Columns 37 through 42***

***7.3066 7.3105 7.3125 7.3145 7.3164 7.3184***

***Key 3: Matched Row Freq: 697.00 Hz***

***Key 3: Matched Col Freq: 1477.00 Hz***

***Detected Key for input 3: 3***

***Key 4: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.7656 0.7676 0.7695 0.7715 0.7734 0.7773 0.7793 0.7812 0.7832 0.7852 0.7871 1.2070 1.2090 1.2109 6.7891 6.7910 6.7930 7.2129***

***Columns 19 through 28***

***7.2148 7.2168 7.2188 7.2207 7.2227 7.2266 7.2285 7.2305 7.2324 7.2344***

***Key 4: Matched Row Freq: 770.00 Hz***

***Key 4: Matched Col Freq: 1209.00 Hz***

***Detected Key for input 4: 4***

***Key 5: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.7656 0.7676 0.7695 0.7715 0.7734 0.7773 0.7793 0.7812 0.7832 0.7852 0.7871 1.3340 1.3359 1.3379 1.3398 6.6602 6.6621 6.6641***

***Columns 19 through 30***

***6.6660 7.2129 7.2148 7.2168 7.2188 7.2207 7.2227 7.2266 7.2285 7.2305 7.2324 7.2344***

***Key 5: Matched Row Freq: 770.00 Hz***

***Key 5: Matched Col Freq: 1336.00 Hz***

***Detected Key for input 5: 5***

***Key 6: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.7656 0.7676 0.7695 0.7715 0.7734 0.7773 0.7793 0.7812 0.7832 0.7852 0.7871 1.4727 1.4746 1.4766 1.4785 1.4805 1.4844 1.4863***

***Columns 19 through 36***

***1.4883 1.4902 1.4922 1.4941 6.5059 6.5078 6.5098 6.5117 6.5137 6.5156 6.5195 6.5215 6.5234 6.5254 6.5273 7.2129 7.2148 7.2168***

***Columns 37 through 44***

***7.2188 7.2207 7.2227 7.2266 7.2285 7.2305 7.2324 7.2344***

***Key 6: Matched Row Freq: 770.00 Hz***

***Key 6: Matched Col Freq: 1477.00 Hz***

***Detected Key for input 6: 6***

***Key 7: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.8477 0.8496 0.8516 0.8535 0.8555 0.8594 0.8613 0.8633 0.8652 0.8672 0.8691 1.2070 1.2090 1.2109 6.7891 6.7910 6.7930 7.1309***

***Columns 19 through 28***

***7.1328 7.1348 7.1367 7.1387 7.1406 7.1445 7.1465 7.1484 7.1504 7.1523***

***Key 7: Matched Row Freq: 852.00 Hz***

***Key 7: Matched Col Freq: 1209.00 Hz***

***Detected Key for input 7: 7***

***Key 8: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.8477 0.8496 0.8516 0.8535 0.8555 0.8594 0.8613 0.8633 0.8652 0.8672 0.8691 1.3340 1.3359 1.3379 1.3398 6.6602 6.6621 6.6641***

***Columns 19 through 30***

***6.6660 7.1309 7.1328 7.1348 7.1367 7.1387 7.1406 7.1445 7.1465 7.1484 7.1504 7.1523***

***Key 8: Matched Row Freq: 852.00 Hz***

***Key 8: Matched Col Freq: 1336.00 Hz***

***Detected Key for input 8: 8***

***Key 9: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.8477 0.8496 0.8516 0.8535 0.8555 0.8594 0.8613 0.8633 0.8652 0.8672 0.8691 1.4727 1.4746 1.4766 1.4785 1.4805 1.4844 1.4863***

***Columns 19 through 36***

***1.4883 1.4902 1.4922 1.4941 6.5059 6.5078 6.5098 6.5117 6.5137 6.5156 6.5195 6.5215 6.5234 6.5254 6.5273 7.1309 7.1328 7.1348***

***Columns 37 through 44***

***7.1367 7.1387 7.1406 7.1445 7.1465 7.1484 7.1504 7.1523***

***Key 9: Matched Row Freq: 852.00 Hz***

***Key 9: Matched Col Freq: 1477.00 Hz***

***Detected Key for input 9: 9***

***Key 0: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.9238 0.9258 0.9277 0.9297 0.9316 0.9336 0.9375 0.9395 0.9414 0.9434 0.9453 1.3340 1.3359 1.3379 1.3398 6.6602 6.6621 6.6641***

***Columns 19 through 30***

***6.6660 7.0547 7.0566 7.0586 7.0605 7.0625 7.0664 7.0684 7.0703 7.0723 7.0742 7.0762***

***Key 0: Matched Row Freq: 941.00 Hz***

***Key 0: Matched Col Freq: 1336.00 Hz***

***Detected Key for input 0: 0***

***Key \*: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.9219 0.9238 0.9258 0.9277 0.9297 0.9316 0.9336 0.9375 0.9395 0.9414 0.9434 0.9453 1.2070 1.2090 1.2109 6.7891 6.7910 6.7930***

***Columns 19 through 30***

***7.0547 7.0566 7.0586 7.0605 7.0625 7.0664 7.0684 7.0703 7.0723 7.0742 7.0762 7.0781***

***Key \*: Matched Row Freq: 941.00 Hz***

***Key \*: Matched Col Freq: 1209.00 Hz***

***Detected Key for input \*: \****

***Key #: Detected Frequencies: 1.0e+03 \****

***Columns 1 through 18***

***0.9219 0.9238 0.9258 0.9277 0.9297 0.9316 0.9336 0.9375 0.9395 0.9414 0.9434 0.9453 1.4727 1.4746 1.4766 1.4785 1.4805 1.4844***

***Columns 19 through 36***

***1.4863 1.4883 1.4902 1.4922 1.4941 1.4961 6.5039 6.5059 6.5078 6.5098 6.5117 6.5137 6.5156 6.5195 6.5215 6.5234 6.5254 6.5273***

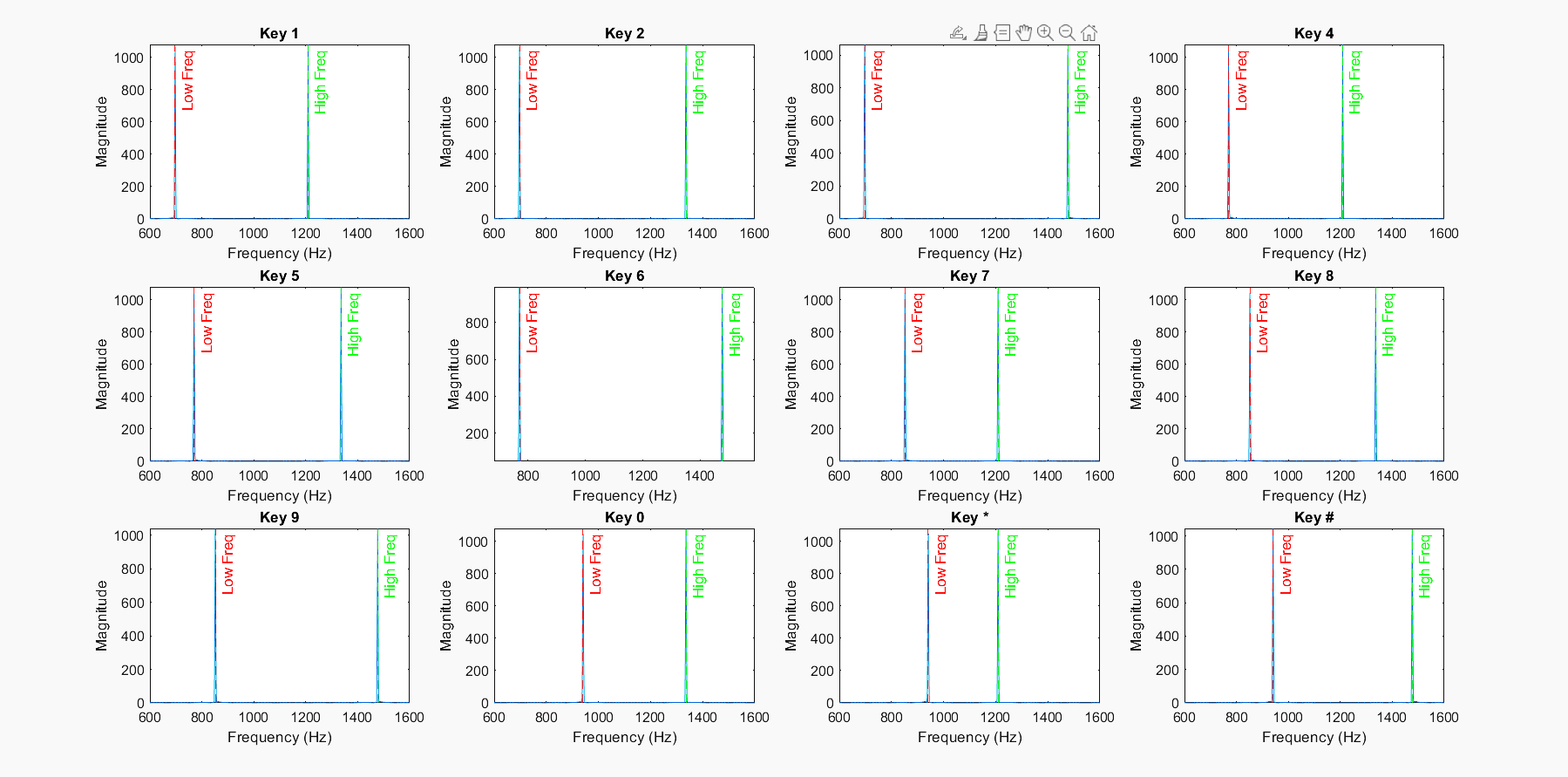
***Columns 37 through 48***

***7.0547 7.0566 7.0586 7.0605 7.0625 7.0664 7.0684 7.0703 7.0723 7.0742 7.0762 7.0781***

***Key #: Matched Row Freq: 941.00 Hz***

***Key #: Matched Col Freq: 1477.00 Hz***

***Detected Key for input #: #***

******

1. **Interactive keyboard input and DTMF signal (sound) generation.**

This code allows the user to input a DTMF key (0-9, \*, #) from the keyboard. The corresponding DTMF signal is generated and played, and the frequency spectrum is displayed. The code is executed to play the corresponding sound, and the graphs are attached here.

***fs = 8000;***

***t = 0:1/fs:0.5;***

***key = input('Enter a DTMF key (0-9, \*, #): ', 's');***

***key\_freqs = {***

***'1', 697, 1209;***

***'2', 697, 1336;***

***'3', 697, 1477;***

***'4', 770, 1209;***

***'5', 770, 1336;***

***'6', 770, 1477;***

***'7', 852, 1209;***

***'8', 852, 1336;***

***'9', 852, 1477;***

***'0', 941, 1336;***

***'\*', 941, 1209;***

***'#', 941, 1477;***

***};***

***freq\_pair = key\_freqs(strcmp(key\_freqs(:, 1), key), 2:3);***

***if isempty(freq\_pair)***

***error('Invalid key entered.');***

***end***

***f\_low = freq\_pair{1};***

***f\_high = freq\_pair{2};***

***dtmf\_signal = sin(2\*pi\*f\_low\*t) + sin(2\*pi\*f\_high\*t);***

***sound(dtmf\_signal, fs); % Play the generated DTMF tone***

***figure;***

***subplot(2, 1, 1);***

***plot(t, dtmf\_signal);***

***title(['DTMF Signal for Key ' key]);***

***xlabel('Time (s)');***

***ylabel('Amplitude');***

***N = 2^nextpow2(length(dtmf\_signal));***

***Y = fft(dtmf\_signal .\* hamming(length(dtmf\_signal))', N);***

***f = (0:N-1)\*(fs/N);***

***magnitude = abs(Y);***

***subplot(2, 1, 2);***

***plot(f, magnitude);***

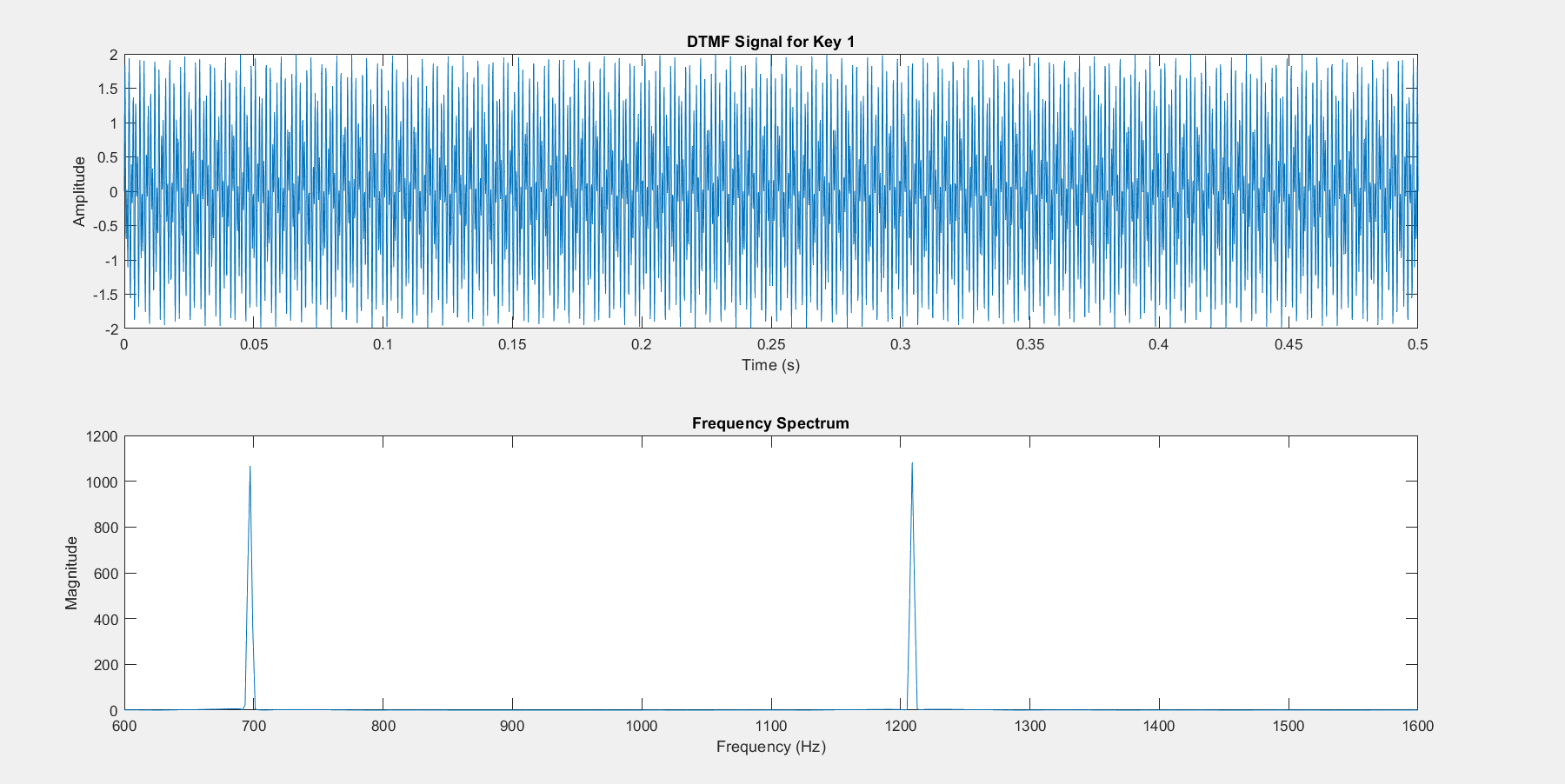
***title('Frequency Spectrum');***

***xlabel('Frequency (Hz)');***

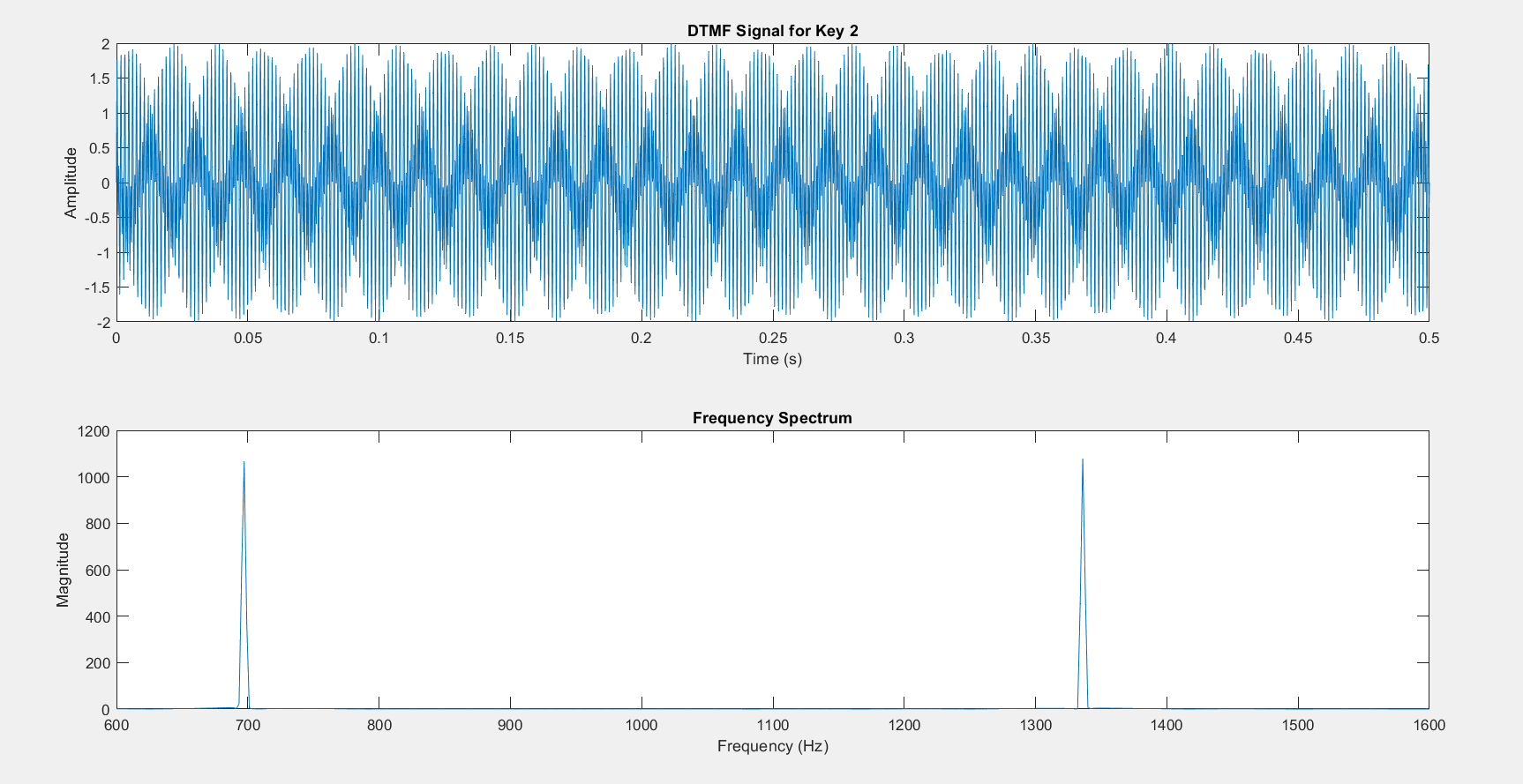
***ylabel('Magnitude');***

***xlim([600 1600]);***

***Enter a DTMF key (0-9, \*, #): 1***



***Enter a DTMF key (0-9, \*, #): 2***



***Enter a DTMF key (0-9, \*, #): 3***

A screenshot of a computer screen

Description automatically generated

***Enter a DTMF key (0-9, \*, #): 4***

A screen shot of a graph

Description automatically generated

***Enter a DTMF key (0-9, \*, #): 5***

A screenshot of a graph

Description automatically generated

***Enter a DTMF key (0-9, \*, #): 6***

A screen shot of a graph

Description automatically generated

***Enter a DTMF key (0-9, \*, #): 7***

A screenshot of a computer screen

Description automatically generated

***Enter a DTMF key (0-9, \*, #): 8***

A screen shot of a graph

Description automatically generated

***Enter a DTMF key (0-9, \*, #): 9***

A screenshot of a computer screen

Description automatically generated

***Enter a DTMF key (0-9, \*, #): 0***

A screen shot of a graph

Description automatically generated

***Enter a DTMF key (0-9, \*, #): \****

A close-up of a graph

Description automatically generated

***Enter a DTMF key (0-9, \*, #): #***

***A screen shot of a graph

Description automatically generated***

1. **Interactive GUI-Based DTMF Keypad Code**

This code creates a simple GUI resembling a telephone keypad. The user can click on any button to generate and play the corresponding DTMF signal. This is useful for quick testing and validation of DTMF generation without additional interfaces. The GUI provides an interactive way to simulate keypad behaviour, ideal for demonstrations and educational purposes.

***f = figure('Name', 'DTMF Keypad', 'NumberTitle', 'off');***

***keys = {'1', '2', '3', '4', '5', '6', '7', '8', '9', '\*', '0', '#'};***

***for i = 1:length(keys)***

***uicontrol('Style', 'pushbutton', 'String', keys{i}, ...***

***'Position', [50 + (mod(i-1, 3) \* 50), 200 - (floor((i-1) / 3) \* 50), 40, 40], ...***

***'Callback', @(src, event) generateDTMF(keys{i}, fs, t));***

***end***

***function generateDTMF(key, fs, t)***

***key\_freqs = {***

***'1', 697, 1209;***

***'2', 697, 1336;***

***'3', 697, 1477;***

***'4', 770, 1209;***

***'5', 770, 1336;***

***'6', 770, 1477;***

***'7', 852, 1209;***

***'8', 852, 1336;***

***'9', 852, 1477;***

***'0', 941, 1336;***

***'\*', 941, 1209;***

***'#', 941, 1477;***

***};***

***freq\_pair = key\_freqs(strcmp(key\_freqs(:, 1), key), 2:3);***

***f\_low = freq\_pair{1};***

***f\_high = freq\_pair{2};***

***dtmf\_signal = sin(2\*pi\*f\_low\*t) + sin(2\*pi\*f\_high\*t);***

***sound(dtmf\_signal, fs);***

***disp(['Key Pressed: ' key]);***

***end***

***Key Pressed: 1***

***Key Pressed: 4***

***Key Pressed: 3***

***Key Pressed: 2***

***Key Pressed: 5***

***Key Pressed: 8***

***Key Pressed: 7***

***Key Pressed: \****

***Key Pressed: 0***

***Key Pressed: 9***

***A screenshot of a computer

Description automatically generated***

### **WHAT WE LEARNED**

### By completing this project, we learned several key concepts and practical skills:

1. **Understanding DTMF Signals:** We learned how DTMF signals are generated and decoded using a DTMF decoder and how these signals can be interpreted to control appliances.
2. **Relay Control:** We gained hands-on experience with controlling relays using microcontrollers, specifically Arduino, to switch appliances on and off.
3. **Arduino Programming:** We furthesr improved our skills in writing and debugging Arduino code, especially working with digital inputs and outputs.
4. **Circuit Design:** We learned how to design and assemble a functional circuit using an Arduino, relay modules, and a DTMF decoder module.
5. **Telecommunication Basics:** We explored the principles behind DTMF technology used in telecommunication systems and applied them to a practical use case.
6. **Problem-Solving:** This project helped us understand how to troubleshoot issues in hardware connections and code logic, enhancing our analytical and debugging skills.
7. How to generate and manipulate signals programmatically using **MATLAB.**
8. The working principles behind **DTMF signalling.**
9. How to use **FFT** for frequency analysis of signals.
10. How to design a simple GUI to simulate DTMF key inputs

**FUTURE SCOPE**

The DTMF-based home automation project offers significant potential for future expansion and integration with advanced technologies:

**Integration with IoT:** By connecting the system to the Internet of Things (IoT), users can control appliances over the internet from anywhere, allowing for better monitoring and automation.

**Voice-Controlled Automation:** Voice recognition technology can be added, enabling users to control appliances using voice commands through virtual assistants like Alexa or Google Assistant.

**Multiple Device Control:** The current system can be scaled to control more appliances by adding additional relays and enhancing the logic for more complex home automation setups.

**Security Features:** Adding a security layer, such as password protection or authentication, can make the system more robust and prevent unauthorized access.

**Energy Monitoring:** Energy-efficient solutions can be integrated by adding real-time monitoring and control to reduce energy consumption of household appliances.

**Smart Feedback Mechanism:** The system could provide feedback on appliance status, letting users know whether a device is currently on or off, with notifications sent directly to a smartphone.

**Real-time DTMF Signal Processing**: Implementing a real-time DTMF decoder using a microphone input, which can analyze signals from an external DTMF source (e.g., phone lines).

**Noise Robustness**: Improving the frequency detection algorithm to work under noisy conditions.

**Software-Hardware Integration**: Integrating the software-based decoder with hardware systems for applications like remote control systems.

**Multitone Signal Handling**: Expanding the program to handle multitone signaling systems beyond DTMF, for more advanced telecommunication applications.

**CONCLUSION**

The DTMF-based home automation system is an efficient, low-cost solution for controlling household appliances using mobile phones. By utilizing DTMF signals and relays, the project demonstrates how simple telecommunication technology can be applied to real-world automation scenarios. The system provides basic home automation capabilities, which can be further extended in the future, making it a stepping stone for more sophisticated smart home applications. The project successfully shows how existing technologies can be leveraged for modern-day convenience and control.

### **ATTACHMENTS**

**GitHub Link: (Contains all documents, presentation slides and codes)**

[**CLICK HERE**](https://github.com/gautham-here/DTMFwithArduino)