1. Introduction

In this project, we have 16 DTMF symbols. Each of these symbols are represented by a superposition of two signals of different frequency. The objective is to detect and display these symbols in MATLAB environment without looking into their frequency spectrum.

Firstly, a DC bias circuit is utilized to shift the signal to positive y axis since Arduino doesn't accept negative signal values, and also the Arduino board samples the signal and sends it to the computer via USB port. Then, this signal is received in MATLAB environment.

The main.m have mainly 3 building blocks. Firstly, input_function() gets samples from the port and normalizes it. Secondly, processor_function() designs the FIR filter, filters the normalize data, and accumulates power of it in a power_vector. Finally, output_function() finds the first and second maximum power and corresponding frequency values inside the power_vector. Then, it searches for this pair of frequency in the dtmf map. Based on the decision, if it finds a symbol, it looks at some conditions before appending the symbol to the received sequence. In order to be able to append the symbol, the very maximum two power values have to be greater than 200. This means that we are currently receiving a DTMF signal. Secondly, a variable named memory has to be lower than 200. This variable stores the previous maximum power value. The reason is that this will prevent multiple displaying of the symbols when the button is being hold.

2. System Design

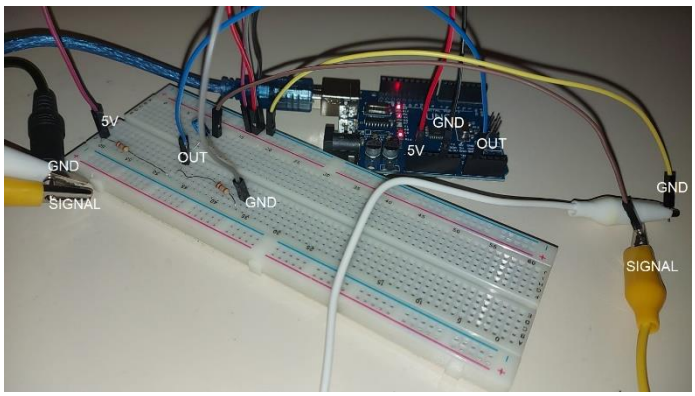

Fig. 1: Arduino setup

b) Detailed explanation of the code line by line

1. The open ports are found.
2. These open ports are closed.
3. The port to be used is selected.
4. empty
5. The size of the buffer block is set.
6. empty
7. Baudrate is set.
8. The selected port is opened.
9. A file named "dtmf_map.mat" is loaded. This variable includes the symbol names as strings, and corresponding two frequencies. This will be used later to find the selected signal from the catalogue.
10. The variable is converted to a simple cell
11. An array containing all unique frequency values is initialized.
12. A power vector is created. This will be used to store the power of each output of the FIR filter.
13. An empty string called received_seq is created. This will store the received symbols.

14. A variable named memory is created. This will store the previous maximum power value, and if it is more than a certain value, we will be notified that the button is being pressed continuously.
15. The start of the while loop which continuously reads data from the buffer.
16. Input function
    1. Start of the input function.
    2. Data is read from the port
    3. And data is normalized.
17. Processor function
    1. Start of the processor function
    2. Half_bw is set to 30 since if it is set to greater values, the resulting filters will be overlapping with each other and this can be result in wrong detection of the symbols. So our filters will have a bandwidth of 2*30 = 60 Hz.
    3. Sampling frequency of MATLAB is set to 8940.
    4. The order of the filter is set to 200. The higher we keep this value, the more accurate will be our filter and corresponding output.
    5. Beta is set to 0.5, which is the window parameter.
    6. Flag is set to “scale”
    7. Start of the for loop to filter different frequency values.
    8. First cutoff frequency is set.
    9. Second cutoff frequency is set.
    10. Empty
    11. For the filter design, the window vector is created.
    12. Empty
    13. The bandpass fir filter is created using the order N, window and cutoff frequencies.
    14. Using the filter coefficients, normalized data is filtered.
    15. The power of the signal is calculated and stored in a vector.
18. The output function which takes the received string sequence, power vector, the memory from the previous loop, the DTMF map, and the frequency of the symbols as input, and gives received sequence and the new generated memory as output.
    1. Start of the output function.
    2. empty
    3. The maximum power value and its index is calculated.
    4. The second maximum power and its index is calculated.
    5. The maximum powered frequency is found.
    6. The second maximum-powered frequency is found.
    7. Start of the for loop used for finding the detected symbol.
    8. We are checking if the detected frequency pair is equal to any pair in the map.
    9. We are also checking whether our maximum power values are over a threshold and the memory from the previous cycle is below a threshold to prevent overdetecting the same signal.
    16. The current maximum value is stored in memory.

21. The serial port is closed.

c) To decide which threshold to use for the power, I experimented many times every symbol, and found out that, when I fulled the sound, the detected power values are almost always greater than 400, and the other filters give the outputs lower than 5. Considering this, I set this threshold to be a middle value which is 200, and it worked.

d)

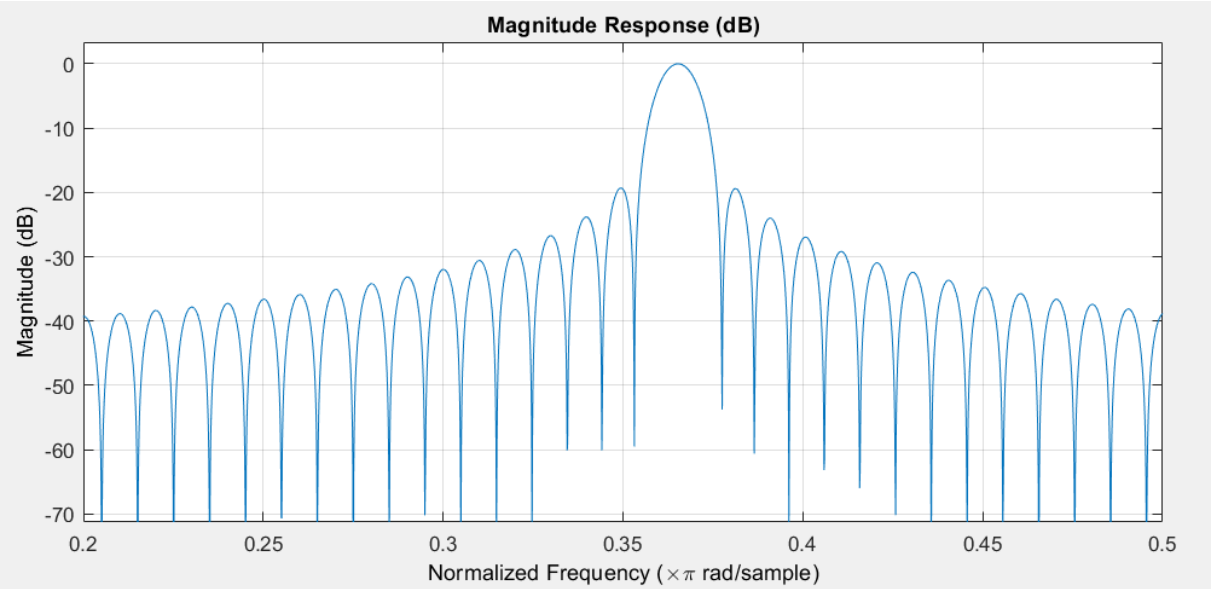

Fig. 2: Magnitude response of one of the filters.

3. Results and Conclusion

The system is working perfectly considering the design criteria. No wrong detections is observed. The system detects the symbols in the blink of eye. When the button is pressed, it doesn't prints the value again and again.

The value of the symbol is printed when I pressed the button. In other words, it doesn't wait for me to release the button. One improvement can be done here. I am not very sure but, if it is possible to decrease the buffer size without affecting the accuracy, this will improve the speed. When I navigate on the buttons fast, it cannot detect it sometimes. This reduction in the buffer size may be improve the detection speed.

I encountered difficulty while deciding on the threshold value. Instead of calculating it by hand, I have done many trials, collected the data, and based on that I did the decision. The second difficulty was the case of pressing the button continuously. I solved this problem by storing the maximum power value from the previous cycle, and if this value is higher than a certain threshold (I assumed that system will detect the not pressed intervals even if they are really small) this means that I am holding the button. So print the symbol just for once.