ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

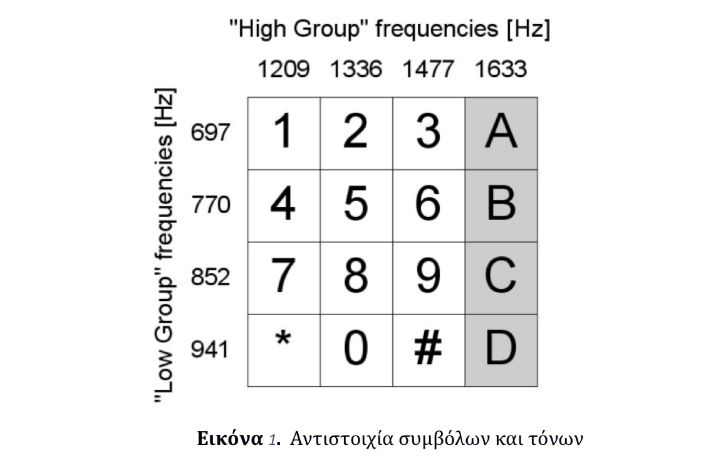
## Εργαστήριο 7

## Υλοποίηση DTMF αποκωδικοποιητή

|  |
| --- |
| Ομάδα 17 11-1-2019 |
| Ασημακόπουλος Κωνσταντίνος 1046966 |
| Λουκαρέας Παύλος 1046970 |

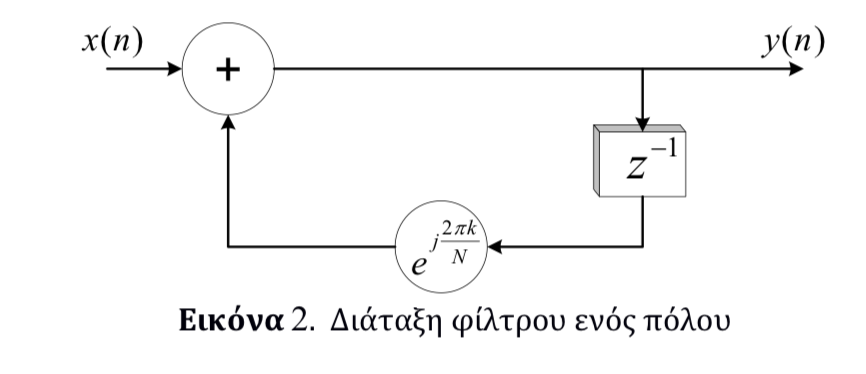
**Περίληψη**

Σε αυτή την εργαστηριακή άσκηση υλοποιήσαμε ένα Dual-Tone Multi-Frequency (DTMF) αποκωδικοποιητή χρησιμοποιώντας fixed point αριθμητική με τη χρήση της πλακέτας. Ο DTMF κωδικοποιητής μεταφράζει το πάτημα ενός πλήκτρου σε ένα σήμα διπλής συχνότητας. Αντίστοιχα, ο DTMF αποκωδικοποιητής μεταφράζει αυτό το σήμα σε ψηφιακή πληροφορία. Παρακάτω βλέπουμε την αντιστοιχία συμβόλων με τους τόνους.

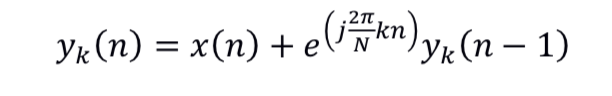


**Βασική θεωρία**

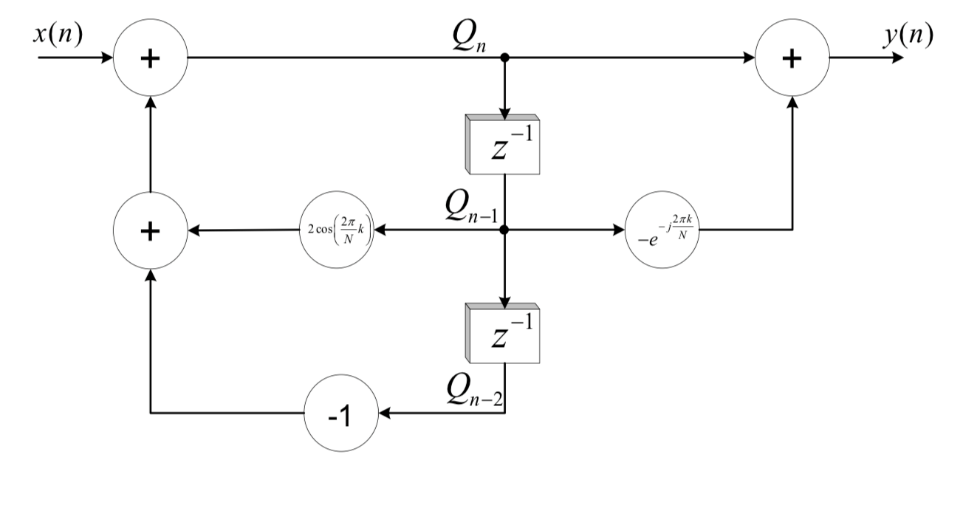
Ο αλγόριθμος Goertzel προκύπτει από τον DFT εκμεταλλευόμενος την περιοδικότητα του συντελεστή φάσης μειώνοντας έτσι την υπολογιστική πολυπλοκότητα. Με μαθηματική επεξεργασία προκύπτει ότι ο υπολογισμός των δειγμάτων του DFT είναι ισοδύναμος με την υλοποίηση του παρακάτω μονοπολικού φίλτρου.



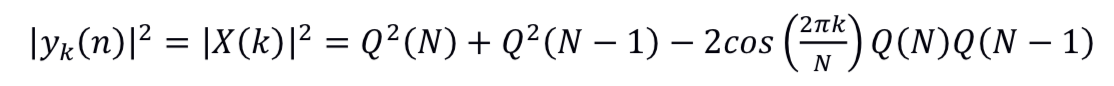
Η εξίσωση διαφορών του φίλτρου είναι:



Όπως βλέπουμε η παραπάνω εξίσωση περιέχει μιγαδικούς πολλαπλασιασμούς που ο καθένας απαιτεί 4 πραγματικούς πολλαπλασιασμούς και 4 πραγματικές προσθέσεις. Για να αντιμετωπίσουμε αυτό το πρόβλημα επεξεργαζόμαστε περαιτέρω την παραπάνω εξίσωση και καταλήγουμε στο παρακάτω φίλτρο.



Με αυτή την τροποποιημένη διάταξη χρειάζεται ένας μιγαδικός πολλαπλασιασμός για τον υπολογισμό της εξόδου. Για την υλοποίηση του αλγόριθμου, το feedback κομμάτι επαναλαμβάνεται Ν-φορές και το feed-forward μόνο μία. Επιπλέον, επειδή στη DTMF αναγνώριση χρειαζόμαστε μόνο το μέτρο της εξόδου ο αλγόριθμος μπορεί να απλοποιηθεί ακόμη περισσότερο. Το τετράγωνο του μέτρου προκύπτει από την εξής σχέση:



Παρατηρούμε ότι για τον υπολογισμό του τετραγώνου του μέτρου απαιτούνται προσθέσεις και πολλαπλασιασμοί πραγματικών αριθμών μόνο.

**Άσκηση 7.1**

Σε αυτό το ερώτημα θέλουμε να υλοποιήσουμε τον αλγόριθμο Goertzel σε MATLAB. Για να ελέγξουμε τη σωστή λειτουργία του παράγουμε όλους τους DTMF τόνους χρησιμοποιώντας τις συναρτήσεις ημιτόνου της MATLAB. Αρχικά, παράγουμε όλους τους DTMF τόνους. Ύστερα για κάθε μία από τις 8 διαφορετικές συχνότητες του πίνακα εκτελούμε τον αλγόριθμο Goertzel. Αφού πάρουμε το φάσμα εξόδου βρίσκουμε το μέγιστο και τη θέση του μεγίστου του φάσματος και κάνουμε αντιστοίχιση με τα πλήκτρα. Ο κώδικας που εκτελεί τα παραπάνω είναι:

fr\_LOW = [697 770 852 941];

fr\_HIGH = [1209 1336 1477 1633];

cf\_LOW = [1.703275 1.635585 1.562297 1.482867];

cf\_HIGH = [1.163138 1.008835 0.790074 0.559454];

tone = ['1','2','3','A';

        '4','5','6','B';

        '7','8','9','C';

        '\*','0','#','D'];

spect\_LOW = [0 0 0 0];

spect\_HIGH = [0 0 0 0];

for i = 1:4

    for j = 1:4

        % composing every freq combination to produce 16 signals, 205 samples

        % 8kHz sampling freq

        X = sin(2 \* pi \* (1:205) \* fr\_LOW(i) / 8000)

        + sin(2 \* pi \* (1:205) \* fr\_HIGH(j) / 8000);

        for k = 1:4

            % compute spectrum

            spect\_LOW(k) = Goertzel(X, cf\_LOW(k));

            spect\_HIGH(k) = Goertzel(X, cf\_HIGH(k));

            % tone position based on the max amplitude

            [frH, posH] = max(spect\_HIGH);

            [frL, posL] = max(spect\_LOW);

            disp(sprintf(('Tone is %c\n'), tone(posL, posH)));

        end

    end

end

Η συνάρτηση που υλοποιεί τον αλγόριθμο Goertzel υπολογίζει αναδρομικά (Ν = 205) τα Qn και υλοποιεί την εξίσωση για τον υπολογισμό του τετραγώνου του μέτρου χρησιμοποιώντας τα Qn. Ο κώδικας είναι ο παρακάτω:

function [spectrum] = Goertzel(input, coeff)

    Qn\_1 = [0];

    Qn\_2 = [0];

    for i = 1:205

        Qn = input(i) + coeff \* Qn\_1 - Qn\_2;

        Qn\_2 = Qn\_1;

        Qn\_1 = Qn;

    end

    Q1 = Qn.^2;

    Q2 = Qn\_1.^2;

    spectrum = Q1 + Q2 - coeff \* Qn \* Qn\_1;

end

**Άσκηση 7.2**

Σε αυτό το ερώτημα υλοποιήσαμε τον αλγόριθμο του Goertzel στην πλακέτα σε γλώσσα C. Η υλοποίηση έγινε με τη χρήση διακοπών και τη διαχείριση τους. Η διαδικασία που ακολουθήσαμε ήταν να υλοποιήσουμε τον αλγόριθμο του Goertzel με μία συνάρτηση κατά αντιστοιχία με το προηγούμενο ερώτημα. Η συνάρτηση είναι η παρακάτω:

short Goertzel(short coef, short\* input, int flag) {

    short spectrum, t;

    int i;

    short Qn;

    short Qn\_1 = 0, Qn\_2 = 0;

    for(i = 0; i < 205; i++) {

        Qn = input[i] + Q15\_Mult(coef, Qn\_1, flag) - Qn\_2;

        Qn\_2 = Qn\_1;

        Qn\_1 = Qn;

    }

    t = Q15\_Mult(coef, Qn, flag);

    spectrum = Q15\_Mult(Qn, Qn, 0) + Q15\_Mult(Qn\_1, Qn\_1, 0) - Q15\_Mult(Qn\_1, t, 0);

    return spectrum;

}

Να σημειωθεί ότι η τρίτη παράμετρος της συνάρτησης είναι ο δείκτης που μας δείχνει πότε οι συντελεστές πρέπει να πολλαπλασιαστούν επί δύο κατά τη διάρκεια των πολλαπλασιασμών.

Για να υλοποιηθούν οι εξισώσεις του αλγορίθμου χρειάζεται να εκτελέσουμε fixed point πολλαπλασιασμούς. Για το σκοπό αυτό ορίσαμε τη συνάρτηση Q15\_Mult().

short Q15\_Mult(short coeff, short data, int flag) {

    int temp;

    short Q;

    if (flag) (Q = (coeff \* (int)data) >> 14);

    else (Q = (coeff \* (int)data) >> 15);

    return Q;

}

Για την εξυπηρέτηση των διακοπών τροποποιήσαμε την ρουτίνα εξυπηρέτησης διακοπών. Η ρουτίνα αυτή διαβάζει δείγματα και τα αποθηκεύει σε έναν buffer μεγέθους 205. Μόλις ο buffer γεμίσει εκτελούμε τον αλγόριθμο του Goertzel για τα δείγματα αυτά και αποθηκεύουμε τις τιμές των φασμάτων για κάθε συχνότητα.

interrupt void serial\_port\_rcv\_isr()

{

    int i;

    int flag = 1;

    int data;

    data = input\_leftright\_sample();

    int posLow, posHigh;

    n++;

    if (n < 205) {

        BufferData[n-1] = (short)data;

    } else {

        n = 0;

        for (i = 0; i < 4; i++) {

            SpecLow[i] = Goertzel(CoeffLow[i], BufferData, 1);

            if (i == 2 || i == 3) flag = 0;

            SpecHigh[i] = Goertzel(CoeffHigh[i], BufferData, flag);

        }

        posLow = max\_pos(SpecLow);

        posHigh = max\_pos(SpecHigh);

        if (previous != Symbols[posLow][posHigh]) {

            printf("The character is: %c\n", Symbols[posLow][posHigh]);

            previous = Symbols[posLow][posHigh];

        }

    }

    output\_leftright\_sample(data);

}

Για να βρούμε το ζευγάρι των σωστών συχνοτήτων πρέπει να βρούμε το μέγιστο φάσμα για τις υψηλές και για τις χαμηλές συχνότητες. Γι’ αυτό δημιουργήσαμε τη συνάρτηση max\_pos() η οποία με δεδομένο έναν πίνακα επιστρέφει τη θέση του μεγίστου.

int max\_pos(short\* buff) {

    int Pos\_max = 0;

    int i;

    short int m = -1;

    for(i = 0; i < 4; i++) {

        if(buff[i] > m) {

            m = buff[i];

            Pos\_max = i;

        }

    }

    return Pos\_max;

}