ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Φασματικός Αναλυτής 8 Περιοχών με χρήση Arduino και NI USB-6001

ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΟΥ ΕΛΕΓΧΟΥ

Μακάριος Χρηστάκης | 4/7/2019

Περιεχόμενα

Εισαγωγή

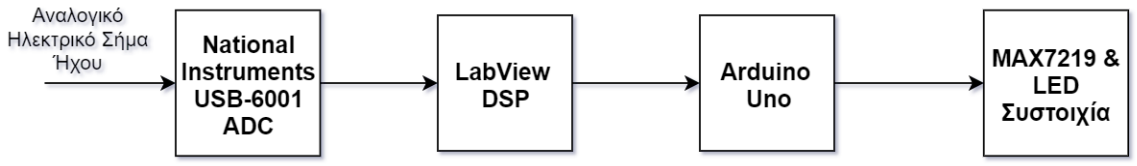
Σκοπός του project αυτού είναι η δημιουργία ενός φασματικού αναλυτή 8 συχνοτικών περιοχών (8-band) για ανάλυση και απεικόνιση του φάσματος ενός σήματος ήχου σε πραγματικό χρόνο. Για το σκοπό αυτό χρησιμοποιήθηκε το εξής hardware:

- National Instruments USB-6001 για A/D μετατροπή του αναλογικού σήματος
- 8x8 LED Matrix για απεικόνιση της κάθε φασματικής περιοχής
- MAX 7219 chip που περιλαμβάνει 2 ενσωματωμένους shift registers για έλεγχο του κάθε LED της παραπάνω συστοιχίας.
- Arduino Uno για έλεγχο του παραπάνω chip μέσω βιβλιοθήκης του.

Το παραπάνω hardware συνδυασμένο με τον κατάλληλο κώδικα LabView & Arduino που θα αναλυθεί παρακάτω είναι δυνατό να υλοποιήσει την επιθυμητή λειτουργία.

Γενική Επισκόπηση

Το βασικό block διάγραμμα του συστήματος φαίνεται παρακάτω:



Αρχικά το αναλογικό σήμα ήχου υφίσταται δειγματοληψία και κβάντιση στα 14 bit μέσω του Analog to Digital Converter του NI-USB6001. Κάθε φορά γίνεται δειγματοληψία 1024 δειγμάτων με $f_s = 20\ kHz$. Το πλαίσιο των 1024 δειγμάτων έπειτα μεταφέρεται στο LabView για επεξεργασία.

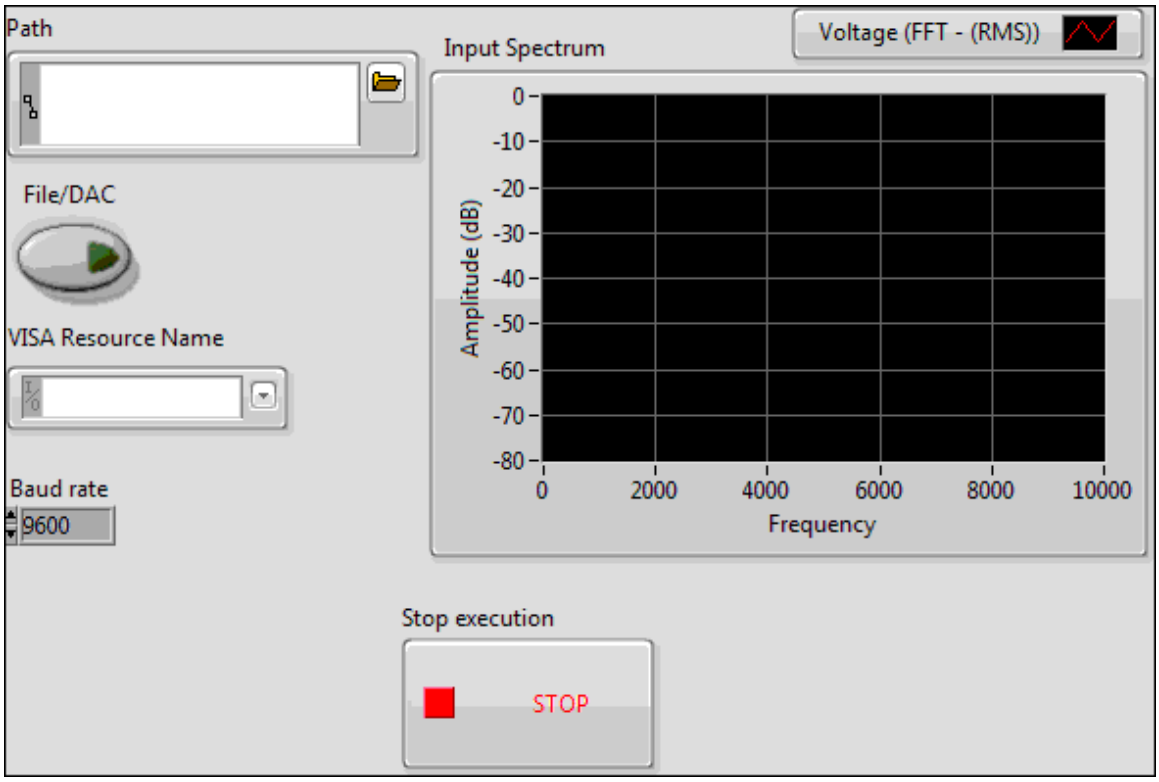
Στη συνέχεια το LabView παίρνει τα δείγματα αυτά, τα περνά από ένα φίλτρο για να αφαιρεθεί τυχόν DC συνιστώσα του αρχικού σήματος, κάνει FFT και συμπιέζει το φάσμα σε 8 αριθμούς, τους οποίους έπειτα στέλνει σαν χαρακτήρες στο Arduino μέσω του Serial Port.

Το Arduino αφού λάβει τα δεδομένα για όλες τις συχνοτικές περιοχές, κάνοντας χρήση της βιβλιοθήκης LedControl στέλνει τα κατάλληλα σήματα στο MAX7219 το οποίο ελέγχει τη LED συστοιχία, ανάβοντας τα επιθυμητά LED.

LabView Κώδικες

1. MAIN VI

Το βασικό Front Panel του προγράμματος φαίνεται παρακάτω:



Με τον Boolean διακόπτη “File/DAC” μπορούμε να διαλέξουμε την πηγή του σήματος ήχου, από αρχείο .wav μέσα από τον υπολογιστή, ή από τον Digital to Analog Converter του NI USB6001.

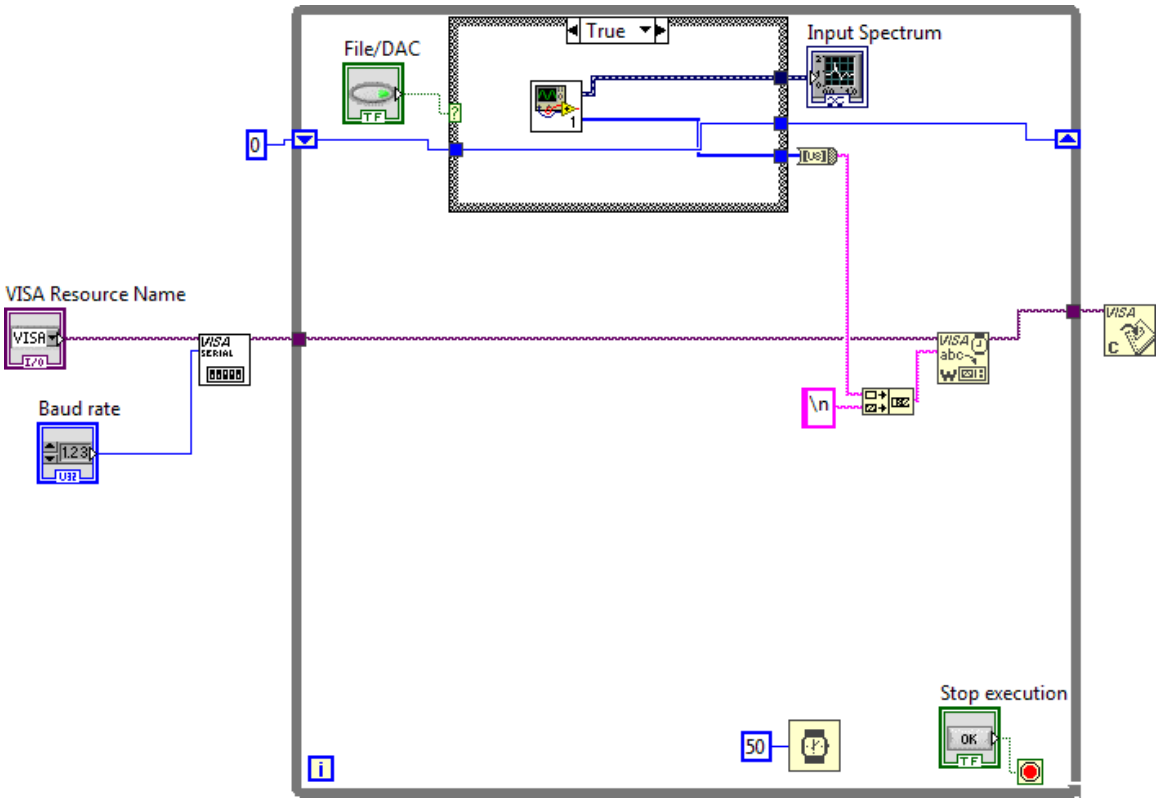
Στο πεδίο VISA Resource Name διαλέγουμε την σειριακή θύρα στο οποίο είναι συνδεδεμένο το Arduino Uno, συνήθως COM5 ή COM4.

Στο πεδίο Baud rate μπορούμε να δηλώσουμε το bitrate που θα χρησιμοποιηθεί για τη σειριακή ζεύξη με το Arduino. Η default τιμή είναι 9600 bps.

Στα δεξιά στο Graph με όνομα Input Spectrum φαίνεται το πλήρες φάσμα του σήματος το οποίο αναλύουμε (με FFT 1024 σημείων) σε κλίμακα dB. Το φάσμα αυτό συμπιέζεται και στέλνεται στο Arduino για απεικόνιση.

Με το κουμπί Stop σταματάει το loop δειγματοληψίας και αποστολής δεδομένων στο Arduino, κλείνει ο δίαυλος επικοινωνίας (Serial) και τερματίζει το πρόγραμμα.

Το back panel του κύριου vi φαίνεται παρακάτω:



Στην αρχή ανοίγουμε την σειριακή θύρα με το επιθυμητό bitrate με το block “Visa Configure Serial Port”, έπειτα ανάλογα με την θέση του boolean επιλέγεται είτε το subVi για εισαγωγή δεδομένων από τον ADC είτε από αρχείο.

Για το subVi που διαβάζει από αρχείο .wav χρησιμοποιείται και μια μεταβλητή long integer (μπλε) που διατηρείται μέσω του shift register μέσα στο while loop. Αυτή η μεταβλητή υποδηλώνει το offset από την αρχή του αρχείου που θα διαβάσουμε τα επόμενα δείγματα στην επόμενη επανάληψη του loop. Για την λειτουργία με ADC προφανώς δεν χρησιμοποιείται, όπως φαίνεται και παραπάνω.

Και τα δυο subVi επιστρέφουν 2 τινά:

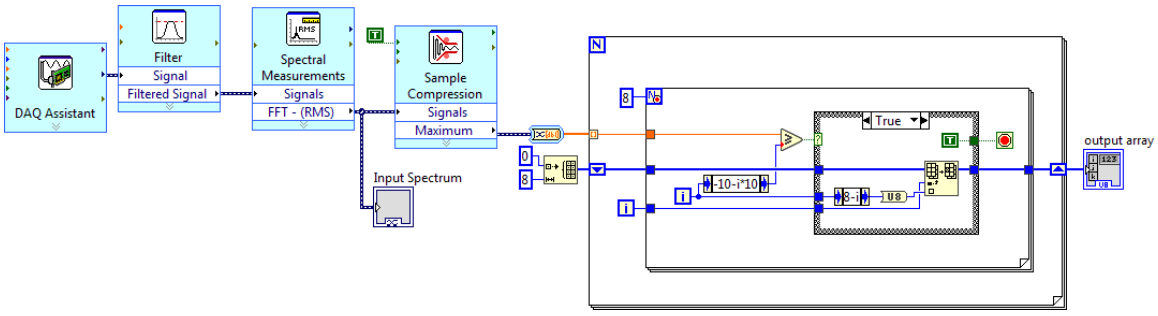
- Έναν array με short integer μήκους 8 στοιχείων, ο οποίος περιέχει τις τιμές των πλατών της κάθε φασματικής μπάντας, όπως υπολογίστηκαν από το subVi.
- Το πλήρες φάσμα του σήματος εισόδου, για απεικόνιση στο front panel.

Στη συνέχεια ο πίνακας αυτός μετατρέπεται σε πίνακα χαρακτήρων (string) και προστίθεται στο τέλος του ο τερματικός χαρακτήρας \n, τον οποίο ανιχνεύει το Arduino για να τερματίσει την ανάγνωση data από το serial port σε κάθε loop.

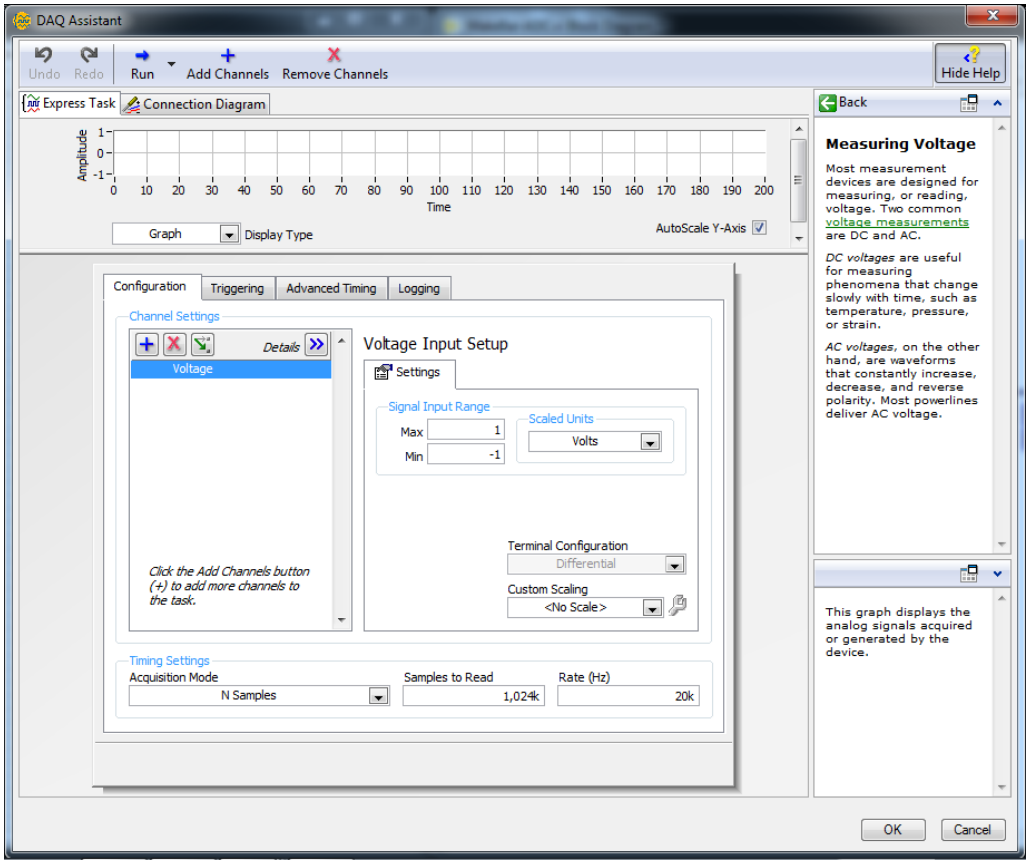
Τα δεδομένα αυτά αποστέλλονται στη serial θύρα με τη συνάρτηση “VISA Write” και η loop συνεχίζει με καθυστέρηση 50 msec, έτσι ώστε να έχει χρόνο το Arduino να λάβει και να επεξεργαστεί τα δεδομένα. Αυτό συμβαίνει διότι η CPU του υπολογιστή είναι πολύ πιο γρήγορη από αυτή του Arduino και συνεπώς πρέπει να «περιμένει» πριν στείλει τα επόμενα δεδομένα.

2. MAKEBAR-ADC

Στόχος του subVi αυτού είναι να πάρει τα δείγματα από τον ADC και να επεξεργαστεί έτσι ώστε να βγάλει τον πίνακα 8 ακεραίων με τα πλάτη των φασματικών περιοχών. Το Vi φαίνεται παρακάτω:

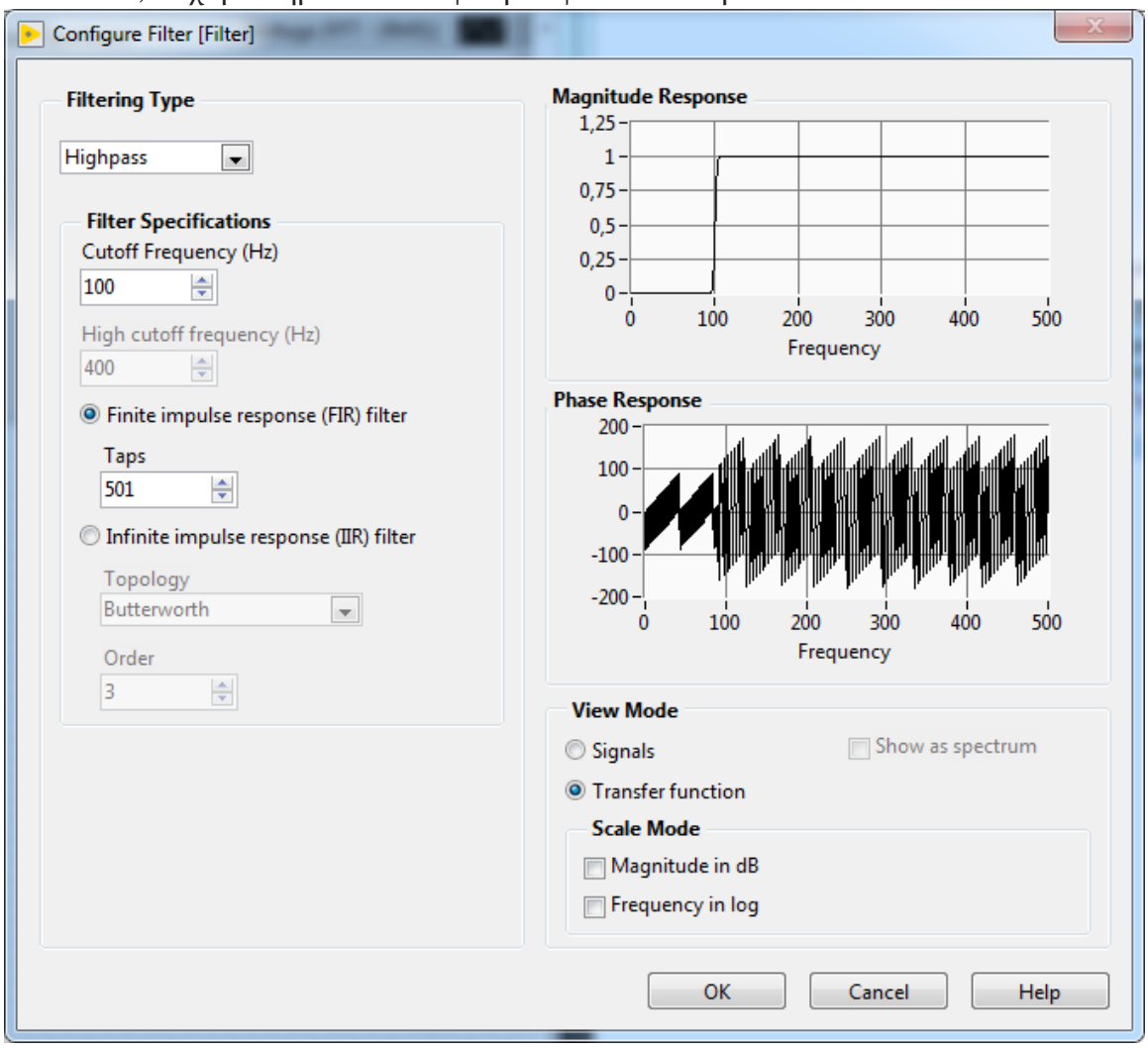


Αρχικά οι ρυθμίσεις του DAQ Assistant είναι οι εξής:



Δειγματοληπτούμε με την μέγιστη δυνατή συχνότητα δειγματοληψίας $f_s = 20kHz$ σε πλαίσια 1024 δειγμάτων.

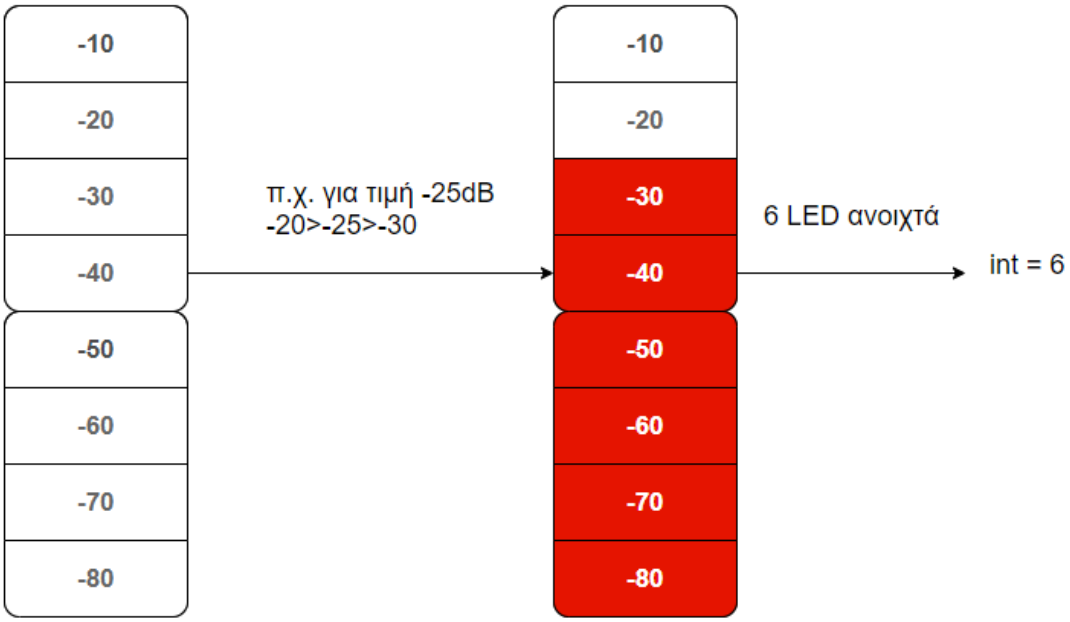
Έπειτα τα δείγματα αυτά περνάνε από ένα HighPass φίλτρο για να αφαιρέσουμε τυχόν DC συνιστώσα, τα χαρακτηριστικά του φίλτρου φαίνονται παρακάτω:



Στη συνέχεια υπολογίζεται το φάσμα σε dB του σήματος μέσω FFT 1024 σημείων και έπειτα με το block “Sample Compression” οι 512 συνιστώσες συχνοτήτων του FFT που προκύπτουν χωρίζονται σε 8 διαστήματα (πχ δείγματα 0-63, 64-127 κλπ.) και για κάθε διάστημα κρατείται μόνο η μέγιστη τιμή σε αυτό το διάστημα. Συνεπώς πετυχαίνουμε μια συμπίεση 1:64 για τις 512 τιμές που προκύπτουν από τον FFT.

Τέλος οι 8 προκύπτουσες στάθμες χωρίζονται μετατρέπονται μέσω της διπλής for loop σε ακέραιες τιμές των 16bit (short integer) με διαδοχικές συγκρίσεις. Η λογική μετατροπής είναι απλή:

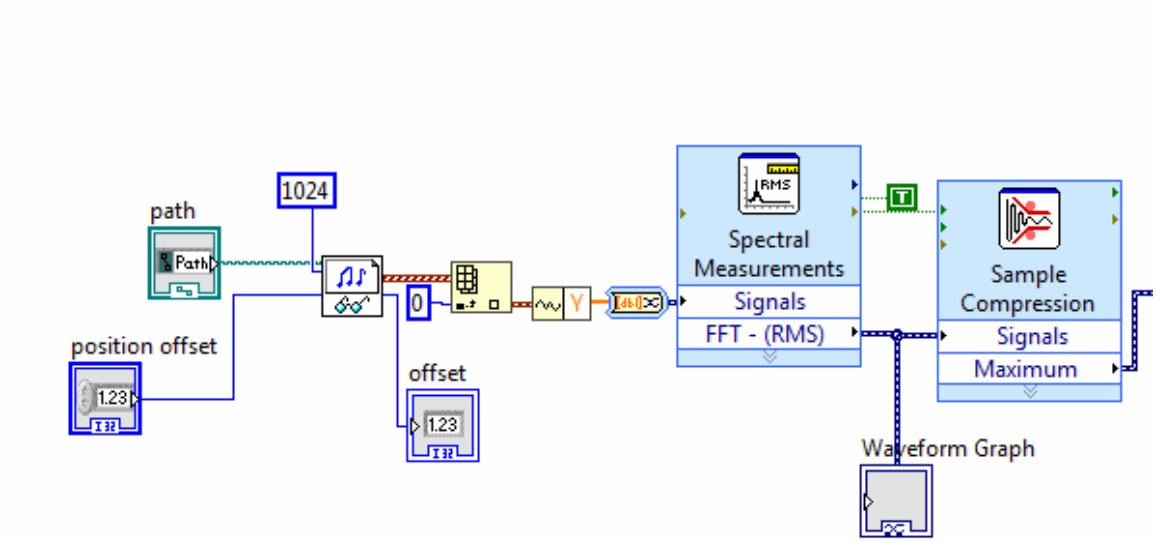
Έχοντας στη διάθεση μας 8 LED για κάθε φασματική μπάντα χωρίζουμε το πλάτος σε db σε 8 ίσες στάθμες από -10 μέχρι -80 dB και τσεκάρουμε αν είναι πάνω ή κάτω από το όριο αυτό και ανάβουμε τόσα LED όσα αντιστοιχούν σε στάθμες μικρότερες ή ίσες από αυτή του ύψους της φασματικής συνιστώσας που θέλουμε να απεικονίσουμε.



Στο παραπάνω παράδειγμα για να ανάψουμε 6 LED (κόκκινα στο σχήμα) στον πίνακα ακεραίων θα πρέπει να γράψουμε τον αριθμό 6.

3. MAKEBAR-FILE

Για το subVi αυτό το μόνο διαφορετικό που συμβαίνει είναι πως στην αρχή χρησιμοποιούμε τη συνάρτηση Sound file read simple.vi της βιβλιοθήκης του LabView για να διαβάσουμε δεδομένα από το αρχείο .wav, όπως φαίνεται παρακάτω:



Όλη η υπόλοιπη διαδικασία είναι η ίδια.

Κώδικας Arduino

Ο κώδικας σε γλώσσα Arduino φαίνεται παρακάτω:

```
#include "LedControl.h"
void bargraph(short int *bars);
/*
   Χρησιμοποιούμε τα pin 12,11 και 10 για την διεπαφή SPI με το MAX7219
   Pin 12 => DATA IN-pin of the MAX7219
   Pin 11 => CLK-pin of the MAX7219
   Pin 10 => LOAD(/CS)-pin of the MAX7219
*/

LedControl mat = LedControl(12, 11, 10, 1);
short int bars[8] = {0};

void setup() {
  mat.shutdown(0, false);
  mat.clearDisplay(0);
  mat.setIntensity(0, 8);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available()) { //See if data is there
    String in = Serial.readStringUntil('\n');
    for (int i = 0; i < 8; i++) {
      bars[i] = in.charAt(i);
    }
  }
  bargraph(bars);
}

void bargraph(short int *bars) {
  int i, j;
  for (i = 0; i < 8; i++) { //iterate over rows
    byte row_states = 0;
    for (j = 0; j < 8; j++) { //iterate over all columns
      if (bars[j] >= i + 1) {
        row_states |= (0x80 >> j);
      }
    }
    mat.setRow(0, 7 - i, row_states);
  }
}
```

Στην συνάρτηση setup() αρχικοποιείται η διεπαφή του MAX7219 με όλα τα LED κλειστά και επιπλέον αρχικοποιείται το Serial port με bitrate 9600bps.

Στη main συνάρτηση loop() το Arduino περιμένει να πάρει δεδομένα από το serial port και διαβάζει μέχρι να ανιχνεύσει τον τερματικό χαρακτήρα \n. Έπειτα τα αποθηκευμένα δεδομένα μετατρέπονται από char σε short integer και τοποθετούνται στον πίνακα bars, ο οποίος χρησιμοποιείται έπειτα από τη συνάρτηση bargraph που κάνει χρήση των

συναρτήσεων της βιβλιοθήκης LedControl.h για να ανάψει τα αντίστοιχα LED κάθε συχνοτικής μπάντας.

Επίλογος

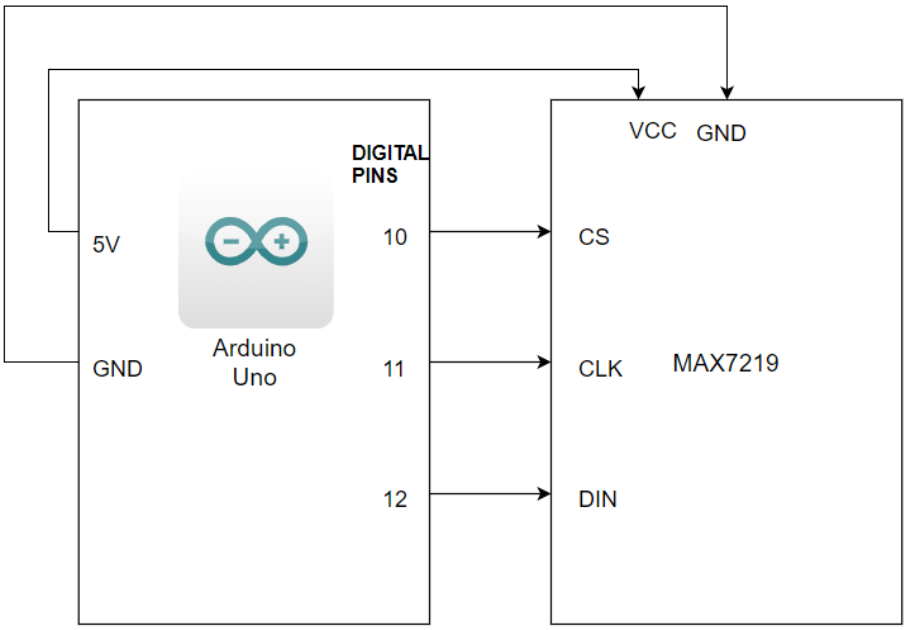
Με το project αυτό καταφέρνουμε να έχουμε ευκρίνεια FFT 1024 σημείων στο φάσμα του σήματος εισόδου και να το συμπιέζουμε σε 8 συνιστώσες που αντιστοιχούν στις περιοχές συχνοτήτων 0-1250Hz, 1251 – 2500Hz κλπ. Δηλαδή έχουμε βήμα συχνότητας ίσο με 1250Hz στο τελικό φάσμα που αποτυπώνεται στα LED.

Αυτό είναι καλύτερο από το να κάναμε FFT 8 σημείων, αφού έτσι θα χάναμε την πληροφορία για τις ενδιάμεσες συχνότητες και το φάσμα που θα παίρναμε δεν θα ήταν τόσο ακριβές.

Τέλος θα ήθελα να προσθέσω πως για ορθότερη λειτουργία του συστήματος αυτού θα πρέπει να είμαστε σίγουροι ότι το αναλογικό σήμα εισόδου έχει φάσμα που δεν ξεπερνά τα 10kHz, δηλαδή το μισό της συχνότητας δειγματοληψίας, αλλιώς θα έχουμε φαινόμενο aliasing στο απεικονιζόμενο φάσμα. Αυτό μπορεί να γίνει με χρήση ενός LowPass φίλτρου με συχνότητα αποκοπής λίγο πιο πριν τα 10KHz, πράγμα που δεν μπόρεσα να φτιάξω διότι απαιτεί μικρές τιμές πυκνωτή (για ένα απλό RC Lowpass filter) που δεν είχα στη διάθεση μου.

Παράρτημα: Συνδεσμολογίες

- Σύνδεση Arduino με MAX7219



- Σύνδεση αναλογικού σήματος ήχου (jack 3.5mm) σε ADC