- ***<u>Ανάλυση κώδικα Arduino και Προτεινόμενης Μεθόδου Επεξεργασίας Σήματος – Blinking Classification/Recognition</u>***

Η κύρια μέθοδος επεξεργασίας του σήματος, με χρήση του Arduino, που θα ακολουθήσουμε για να ανιχνεύσουμε το πλήθος των blinkings θα είναι ο FFT (Fast Fourier Transform) και συγκεκριμένα μια παραλλαγή του, ο FHT (Fast Hartley Transform). Ο FHT είναι μια ειδική περίπτωση FFT, απλά αφορά καθαρά συστήματα με πραγματική είσοδο και πραγματική έξοδο, δηλαδή είναι απαλλαγμένη από μιγαδικές οντότητες. Στην περίπτωση μας έχουμε πραγματικά σήματα, όμως ο κύριος λόγος που επιλέγουμε τον FHT αντί του FFT είναι επειδή ο FHT είναι αρκετά πιο γρήγορος και απαιτεί λιγότερη μνήμη, χαρακτηριστικά πάρα πολύ σημαντικά για τις συνθήκες που θέλουμε, αφού επιθυμούμε real time ανάλυση και έχουμε περιορισμένη μνήμη λόγω του μικροεπεξεργαστή του Arduino Uno.

Συνεπώς, για να έχουμε καλή διακριτική ικανότητα και πολλά δείγματα, δειγματοληπτούμε με το clock του εσωτερικού ADC του Arduino με 256 δείγματα ανά περίοδο δειγματοληψίας. Επειδή εμπειρικά έχουμε βρει χονδρικά κάποιες τιμές της επικρατούσας συχνότητας όπου αντιστοιχεί το blinking για διάφορες περιπτώσεις, με την μέθοδο αυτή μπορούμε να απομονώσουμε τις χαρακτηριστικές συχνότητες των σωστών κυματομορφών (εδώ η περίπτωση του blinking), αποκόπτοντας στο φάσμα τις ανεπιθύμητες συχνότητες που με τεχνικές trial and error βρήκαμε πως δεν αποτελούν περίπτωση blinking. Έτσι, εισάγουμε κάποιους αρκετά αυστηρούς περιορισμούς με χρήση κάποιων κατωφλίων συχνότητας, που όταν ικανοποιηθούν θα έχουμε ισχυρή ένδειξη για την ύπαρξη blinking.

Ο FHT αποτελεί το πρωταρχικό/θεμελιώδες επίπεδο περιορισμών και επεξεργασίας. Αν, σπανίως βέβαια, για οποιοδήποτε λόγο δεν καταφέρει να ανιχνεύσει blinking, τότε σειρά έχει το βοηθητικό/δευτερεύον επίπεδο ανίχνευσης blinking, που εισάγει κάποιους λιγότερο αυστηρούς περιορισμούς/κατώφλια. Εδώ υπάγονται οι περιπτώσεις όπου έχουμε ισχυρό blinking (δυνατό ανοιγοκλείσιμο ματιών). Αν συμβεί αυτό, με τη χρήση των μεταβλητών up, down, elapsed ανιχνεύουμε και αυτού του είδους blinking. Οι δύο πρώτες μεταβλητές ενεργοποιούνται όταν ικανοποιηθεί κατά την άνοδο της ενεργής κυματομορφής το κατώφλι για την up και αντίστοιχα για την down κατά την κάθοδο. Έτσι εξασφαλίζεται μια πλήρη περίοδος ενεργής κυματομορφής, η οποία υφίσταται όταν η τάση του αισθητήρα ξεπεράσει πχ τα 4.3 V στην άνοδο (ενεργοποίηση) και κατά την κάθοδο (απενεργοποίηση-λήξη ενεργής περιόδου). Αυτό θα σημάνει την αναγνώριση του (ισχυρού) blinking.

Τέλος, για επιπλέον προστασία έναντι false positives, μετράμε αυτόν το χρόνο ανόδου-καθόδου της ενεργής περιοχής με την elapsed. Εμπειρικά, για τον εν λόγω κώδικα, μετρήθηκε ότι θα πρέπει να είναι στο διάστημα τιμών 52000-54000 και συνεπώς εισάγεται έτσι και ο τρίτος περιορισμός.

Ο FHT δεν θα μπορούσε να αναγνωρίσει εύκολα τέτοια ισχυρά blinking διότι σε αυτήν την περίπτωση, το πλάτος της επικρατούσας συχνότητας υφίσταται κορεσμό στις 676 μονάδες περίπου και θεωρήσαμε συνετό να μην υπάγεται στην FHT επεξεργασία.

Η σχέση που δίνει την πραγματική (επικρατούσα) συχνότητα με βάση τον κώδικα, για i το εκάστοτε bin/δείγμα (0-255), πλήθος δειγμάτων 256 και μέγιστο πλάτος max_freq (το οποίο ΔΕΝ είναι συχνότητα, αλλά το μέγιστο πλάτος που αντιστοιχεί στην επικρατούσα συχνότητα, η οποία δίνεται από τη σχέση που ακολουθεί) είναι: i*9615/256, όπου 9615 Hz είναι η συχνότητα μετατροπής κατά τη δειγματοληψία του εσωτερικού ADC του Arduino Uno.

Έτσι πχ αν η max_freq προκύψει να είναι για το i=20, τότε αυτό θα σημαίνει πως τότε, για εκείνη τη μέτρηση/δείγμα, η επικρατούσα συχνότητα μέσω FHT ανάλυσης του σήματος, θα είναι στα 751,17 Hz περίπου, με (σχετικό) πλάτος/«πιθανότητα» επικράτησης ίσο με max_freq. Άρα, το εύρος συχνοτήτων που εξετάζεται είναι 0-9577,44 Hz (για i = 255).

Για την υλοποίηση του FHT, χρησιμοποιήθηκε στο Arduino η βιβλιοθήκη ArduinoFHT4 του «OpenMusicLabs». Οι βασικές εντολές/συναρτήσεις που καλούνται είναι οι:

- ✓ *fht_window();* // window the data for better frequency response
- ✓ *fht_reorder();* // reorder the data before doing the fht
- ✓ *fht_run();* // process the data in the fht
- ✓ *fht_mag_lin();* // take the output of the fht

ενώ απαραίτητα είναι τα define:

- *#define LIN_OUT 1* // use the log output function
- *#define FHT_N 256* // set to 256 point fht
- *#define SCALE 256*

Επιλέγουμε μεταξύ των διαθέσιμων μεθόδων απεικόνισης της εξόδου, την γραμμική (LIN), διότι εμπειρικά βρήκαμε να είναι η πιο αποτελεσματική και εύχρηστη.

Τέλος, με το *« #include "FHT.h"* » εισάγουμε την βιβλιοθήκη στο .ino πηγαίο αρχείο μας.

Όλα τα κατώφλια στον κώδικα προέκυψαν εμπειρικά με χρονοβόρες και επίπονες τεχνικές trial and error. Ενδέχεται από άτομο σε άτομο, αναλόγως τύπου επιδερμίδας, αν είναι λεία, λιπαρή, καθαρισμένη, αν έχει σπυράκια, ιδρώτα, τρίχες κλπ, να διαφοροποιούνται, εξαρτώμενα φυσικά και από τη σωστή τοποθέτηση των ηλεκτροδίων (περαιτέρω διαφοροποίηση ίσως για άλλης μάρκας/ποιότητας ηλεκτρόδια) και του gel. Γι'αυτό, ανά περίπτωση και δοκιμή, συνιστούμε να ελέγχονται και να προσαρμόζονται (tweaking) στα δεδομένα του χρήστη και τις συνθήκες της μέτρησης. Μια πρόταση λύσης απέναντι σε αυτό το ζήτημα παρατίθεται στο Παράρτημα, όπου παρουσιάζονται συνοπτικά μελλοντικοί τομείς έρευνας (Future Research) και βελτιώσεις (Improvements/Optimization) αυτής της εργασίας. Ο αναγνώστης, για περαιτέρω διευκρινήσεις στον κώδικα μπορεί να ανατρέξει στα σχόλια εντός του κώδικα.

Τέλος, αξίζει να περιγράψουμε σύντομα κάποιες συνθήκες που έχουμε εισάγει στον FHT. Γενικά, ηρεμία έχουμε όταν το max_freq είναι κάτω από 110. Όταν το max_freq πέσει κάτω από 130 και αληθεύουν οι 3 counter (δείτε κώδικα), τότε έχουμε αναγνώριση blinking και οι αντίστοιχες τιμές παίρνουν τις default τιμές τους για εκ νέου αναγνώριση blinking. Για να αληθεύουν όμως οι υπο-συνθήκες των counter αυτών, θα πρέπει πρώτα να εγκριθούν από τις πρότερες συνθήκες των 4 πιο πάνω if δομών. Οι μεταβλητές eye_up_counter και eye_down_counter προσπαθούν να ελέγξουν αν ο χρήστης απλώς κοιτάει έντονα πάνω ή κάτω. Αυτή η περίπτωση από μόνη της, λόγω κατωφλίων για την max_freq, θα οδηγούσε σε λανθασμένη εκτίμηση blinking (false positive). Όμως με τη χρήση αυτών των μεταβλητών, απορρίπτουμε όσο το δυνατόν γίνεται τέτοιες περιπτώσεις, ως εξής: Εμπειρικά έχουμε βρει ότι όταν κοιτάμε έντονα πάνω, ο FHT δίνει αρκετές τιμές για την max_freq στο διάστημα 140-250 περίπου, ενώ για έντονα κάτω ακόμα περισσότερες τιμές στο ίδιο διάστημα, χονδρικά. Αν διαπιστωθεί ότι έχουμε τέτοιο μεγάλο πλήθος τιμών μέσα σε τόσο λίγο εύρος, τότε δεν το δεχόμαστε σαν blinking. Βεβαίως θα μπορούσε κάποια φορά να είναι όντως blinking, αλλά

αυτό είναι κάπως απίθανο. Όπως καθίσταται φανερό, το σύστημα αυτό δεν μπορεί να είναι 100% επιτυχές. Απλά η προσπάθεια κατευθύνεται στο να γίνει όσο το δυνατόν πιο εύρωστο και αποτελεσματικό στα πλαίσια της εργασίας, με αυτόν τον αισθητήρα, αυτά τα ηλεκτρόδια και κυρίως, με τις επεξεργαστικές δυνατότητες ενός Arduino.

Οι ονομαστικές τιμές περιορισμών για blinking είναι στα 140-460 και με μάλιστα λιγότερες των 9 φορών ικανοποίησης αυτών από την max_freq, αλλιώς θεωρούμε πως εμπίπτουμε πάλι στις παραπάνω λανθασμένες περιπτώσεις. Τελικά, όλες αυτές οι if δομές κλειδώνουν σε περίπτωση που όντως δεν είναι blinking και ενεργοποιούνται πάλι όταν τείνει να επέλθει η ηρεμία (max_freq <= 135).

- *Αποτελέσματα – Ποσοστά επιτυχίας:*

  Εφαρμόσαμε το σύστημα μας και σε περιπτώσεις κανονικού blinking, αλλά και extreme, δηλαδή να κοιτάμε έντονα πάνω-κάτω χωρίς blinking, έντονο ή και γρήγορο blinking, κούνημα κεφαλιού, ταυτόχρονα ομιλία, κλπ. Σε κανονικές συνθήκες καταφέραμε να πετύχουμε 90-100 / 101 blinkings -> ποσοστό επιτυχίας 89-99 %, ενώ σε δύσκολες περιπτώσεις όπως περιγράψαμε, το ποσοστό αυτό έπεσε στο ικανοποιητικό φυσικά 71-85 / 101 blinkings -> ποσοστό επιτυχίας 70-84 %. Σαφώς όλα αυτά ενδέχεται να διαφοροποιούνται αναλόγως το gel, τα ηλεκτρόδια (φθορά, βρωμιές, κλπ) και την θέση τους, την επιδερμίδα του χρήστη, κλπ κλπ…

✓ Πιστεύουμε δηλαδή πως σε γενικές γραμμές, αποτελεί έναν αρκετά αποτελεσματικό ταξινομητή για blinkings σε Arduino.

• ΠΑΡΑΡΤΗΜΑ

❖ *Suggestions for Future Research – Improvements:*

- Καλύτερα ηλεκτρόδια και pads επαφής αυτών.
- Καλύτερη στερέωση και σταθερότητα ηλεκτροδίων.
- Κατασκευή δική μας πλακέτα PCB με αισθητήρα και ενσωματωμένο Bluetooth.
- Καλύτερος μικροεπεξεργαστής, ενδεχομένως ARM, με περισσότερη μνήμη.
- Αποστολή δεδομένων για περαιτέρω επεξεργασία και visualization σε υπολογιστή (Matlab, RStudio).
- Επιπρόσθετα φίλτρα και ηλεκτρονικές διατάξεις απομάκρυνσης παρεμβολών και θορύβου – εξομάλυνση σήματος.
- Εκπαίδευση νευρωνικού δικτύου για την αυτόματη ανίχνευση ιδανικών τιμών κατωφλίων (thresholds), προσαρμοσμένα στον εκάστοτε χρήστη για τις διάφορες συνθήκες μέτρησης.
- Εκπαίδευση classifier με training και testing data, όπου θα παρέχουμε ανάδραση στο νευρωνικό δίκτυο του classifier, για να μελετά με τεχνικές machine learning (μηχανικής μάθησης) σε περίπτωση blinking τι είδους κυματομορφή παράγεται με λεπτομέρεια (training data) και έπειτα αναγνώριση διαφόρων περιπτώσεων blinking και εφαρμογή σε unclassified data (testing data).
- Συνδυασμένη αξιοποίηση για αναγνώριση blinking του OpenCV για machine learning και εκπαίδευση του νευρωνικού.