Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη αλγόριθμου αναγνώρισης χειρονομιών σε χαμηλής κατανάλωσης Bluetooth τσιπ για εφαρμογές Internet-Of-Things** **Development of gesture recognition algorithm in low-power Bluetooth chip for Internet-Of-Things applications**
Ονοματεπώνυμο Φοιτητή	**Ιωάννα Πάνου**
Πατρώνυμο	**Κωνσταντίνος**
Αριθμός Μητρώου	**ΜΠΣΠ/ 12055**
Επιβλέπων	**Ψαράκης Μιχαήλ, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Ιούλιος 2015**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ψαράκης Μ. Επίκουρος καθηγητής	Πικράκης Α. Επίκουρος καθηγητής	Κοτζανικολάου Π. Επίκουρος καθηγητής

Ευχαριστίες

Η παρούσα μεταπτυχιακή διατριβή πραγματοποιήθηκε υπό την επίβλεψη του κ. Μιχάλη Ψαράκη, τον οποίο θα ήθελα να ευχαριστήσω ιδιαιτέρως για την πολύτιμη καθοδήγηση, τα εποικοδομητικά σχόλια καθώς και τις παρατηρήσεις του καθ' όλη την διάρκεια εκπόνησης της.

Ακόμα, θα ήθελα να ευχαριστήσω το προσωπικό της Dialog Semiconductor, που μας προμήθευσαν με τον αναπτυξιακό και μας έδωσαν την ευκαιρία να εκπονήσουμε μια τόσο ενδιαφέρουσα πτυχιακή διατριβή.

Επίσης, ένα θερμό ευχαριστώ στον συμφοιτητή μου Μάριο Τζανιδάκη, με τον οποίο συνεργάστηκα άψογα καθώς μοιράστηκε τον ίδιο ενθουσιασμό με εμένα καθ'όλη την διάρκεια εκπόνησης της εργασίας.

Επιπλέον, θα ήθελα να ευχαριστήσω τον καθηγητή κ. Πικράκη για τις πολύτιμες συμβουλές του σχετικά με την αναγνώριση προτύπων.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για την απεριόριστη υπομονή τους και συμπαράσταση.

Περίληψη

Η παρούσα μεταπτυχιακή διατριβή πραγματεύεται την επιλογή και τροποποίηση ενός αλγορίθμου αναγνώρισης προτύπων ο οποίος θα μπορεί να χρησιμοποιηθεί σε ενσωματωμένες συσκευές με επεξεργαστή χαμηλής απόδοσης και κατανάλωσης και περιορισμένη μνήμη. Ως ενσωματωμένη πλατφόρμα, επιλέξαμε το system-on-chip SmartBondDA14580 της Dialog Semiconductors που υποστηρίζει Bluetooth Low Energy και παρέχει την δυνατότητα ανάπτυξης εφαρμογών Bluetooth 4.1. Η χρησιμοποίηση του αυτούσιου αλγορίθμου ήταν αδύνατη στο SmartBondDA14580 τσιπ. Για τον λόγο αυτό παρουσιάζονται όλες οι ρυθμίσεις που επιλέχθηκαν και οι τεχνικές που ακολουθήθηκαν έτσι ώστε να μειωθεί το μέγεθος του κώδικα, η μνήμη που χρησιμοποιείται καθώς επίσης και να βελτιωθεί η ταχύτητα εκτέλεσης του αλγορίθμου. Παρόλο που υπάρχουν διάφοροι περιορισμοί στο σύστημα αναγνώρισης χειρονομιών όπως το ότι είναι εξαρτώμενο από τον χρήστη αποδείξαμε ότι υπάρχει τρόπος να υλοποιηθούν αλγόριθμοι αναγνώρισης προτύπων σε τέτοιες συσκευές, ανοίγοντας έτσι τον δρόμο για μια νέα γενιά συσκευών που θα παρέχουν περισσότερες δυνατότητες στους χρήστες.

Abstract

The current dissertation deals with the selection and modification of a pattern recognition algorithm that will be able to be used in devices with low power cpu and limited resources. As a demonstration vehicle, we choose the Dialog Semiconductors System-on-Chip SmartBond DA14580 that supports Bluetooth Low Energy and enables the development of Bluetooth 4.1 applications. The use of such an algorithm as it is, was impossible using the Smartbond DA14580 chip from Dialog Semiconductor. For this reason, we present in this paper all the modifications that were chosen and the techniques that were used in order to reduce the size of the code, the memory used and also to improve the execution time of the algorithm. Even though there are some restrictions in the recognition system, such as the fact that it is user-depended, we proved that there is a way to implement pattern recognition algorithms in such devices, leading the way for a new generation of devices that will provide the users with more capabilities.

Πίνακας περιεχομένων

Εισαγωγή

Διάφοροι τομείς έρευνας της πληροφορικής έχουν ως στόχο να φέρουν την αλληλεπίδραση του ανθρώπου με έναν υπολογιστή όσο πιο κοντά γίνεται με την αλληλεπίδραση του με έναν άλλο άνθρωπο. Οι χειρονομίες παίζουν ένα σημαντικό ρόλο στην καθημερινή μας ζωή, βοηθώντας μας να μεταδώσουμε πληροφορίες και να εκφράσουμε συναισθήματα. Από τα διάφορα ανθρώπινα μέλη, το χέρι είναι το πιο αποτελεσματικό, γενικού σκοπού εργαλείο αλληλεπίδρασης. Συνεπώς, η ανίχνευση και αναγνώριση προτύπων για τις χειρονομίες αυτές έχει γίνει ένας ενεργός τομέας έρευνας.

Η αναγνώριση προτύπων βρίσκει όλο και περισσότερες εφαρμογές από τις έξυπνες τηλεοράσεις μέχρι χειριστήρια παιχνιδομηχανών καθώς όλο και περισσότερες συσκευές ενσωματώνουν αισθητήρες εξαιτίας του χαμηλού κόστους και του μικρού μεγέθους τους. Όμως, επειδή χρησιμοποιεί αλγορίθμους που απαιτούν αρκετή υπολογιστική δύναμη και μεγάλο μέγεθος μνήμης θέτονται περιορισμοί στους τομείς που μπορούν να εφαρμοστούν.

Στην εποχή μας, κατακλυζόμαστε από διάφορες συσκευές οι οποίες μπορούν να αναγνωρίσουν η μία την άλλη καθώς και να αλληλεπιδράσουν μεταξύ τους. Ακόμα και σε ένα περιορισμένο χώρο όπως ένα διαμέρισμα, υπάρχουν συσκευές όπως υπολογιστές, κινητά τηλέφωνα, έξυπνες τηλεοράσεις οι οποίες αποτελούν το δικό μας προσωπικό Internet of Things. Σύμφωνα με την έρευνα ABI [16], περισσότερες από 40 δισεκατομμύρια συσκευές θα είναι συνδεδεμένες στο IoT μέχρι το 2020, και το bluetooth smart θα παίξει καταλυτικό ρόλο σε αυτό. Το γεγονός αυτό αναδεικνύει την ανάγκη για εύχρηστες και μικρές συσκευές εισόδου που θα μας επιτρέπουν την λειτουργία των συσκευών αυτών από απόσταση. Καθώς οι συσκευές αυτές συνήθως χρησιμοποιούν μικροεπεξεργαστές και είναι τόσο μικρές ώστε να μπορούν να τοποθετηθούν σε ένα δάχτυλο, η προσαρμογή των αλγορίθμων αναγνώρισης προτύπων στα νέα δεδομένα είναι το επόμενο βήμα.

Η αναγνώριση χειρονομιών αποτελεί έναν τομέα της επιστήμης των υπολογιστών που έχει ως στόχο την μετάφραση των ανθρώπινων χειρονομιών χρησιμοποιώντας μαθηματικούς αλγορίθμους. Οι χειρονομίες μπορούν να προέλθουν από οποιοδήποτε μέρος του σώματος, με το πρόσωπο και το χέρι να είναι τα πιο κοινά. Για τον σκοπό αυτό μπορούν να χρησιμοποιηθούν κάμερες οι οποίες καταγράφουν τις κινήσεις ή και αισθητήρες αδράνειας όπως επιταχυνσιόμετρα και γυροσκόπια.

Ένα επιταχυνσιόμετρο είναι μια συσκευή η οποία μετράει δυνάμεις επιτάχυνσης. Αυτές μπορεί να είναι είτε στατικές, όπως π.χ. το διάνυσμα της βαρύτητας ή δυναμικές όπως π.χ. η επιτάχυνση που αναπτύσσεται κουνώντας το επιταχυνσιόμετρο. Μελετώντας τα δεδομένα αυτά, δίδεται η δυνατότητα να αναλύσουμε τον τρόπο με τον οποίο κινείται η συσκευή. Συνεπώς, με την τοποθέτησή της σε ένα δάχτυλο του χεριού, μπορούμε να αναγνωρίσουμε τις κινήσεις που εκτελεί.

Η παρούσα μεταπτυχιακή διατριβή πραγματεύεται την επιλογή και στην συνέχεια προσαρμογή ενός αλγορίθμου αναγνώρισης προτύπων από δεδομένα που προέρχονται από έναν αισθητήρα αδράνειας ο οποίος είναι τοποθετημένος σε μορφή δαχτυλιδιού στον δείκτη του χεριού ενός χρήστη, έτσι ώστε να μπορεί να χρησιμοποιηθεί ακόμα και σε εφαρμογές στις οποίες χρησιμοποιούνται μικροεπεξεργαστές χαμηλού κόστους και απόδοσης.

Το δαχτυλίδι αυτό συνδέεται μέσω Bluetooth Low Energy ως συσκευή εισόδου σχεδόν σε όλα τα λειτουργικά συστήματα (windows, IOS, android 4.3 και άνω). Οι διάφορες χειρονομίες αναγνωρίζονται από τον αλγόριθμο ως πάτημα του δαχτύλου, σήκωμα του δαχτύλου ή πολύ γρήγορο συνδυασμό των δύο. Μετά την αναγνώριση, η κίνηση αποστέλλεται χρησιμοποιώντας το προφίλ HID του Bluetooth Low Energy στην συσκευή, η οποία και εκτελεί την κατάλληλη λειτουργία.

Αρχικά, θα γίνει μια πιο λεπτομερής παρουσίαση του προβλήματος. Ξεκινώντας από το κεφάλαιο 1, όπου γίνεται μια εισαγωγή στις έννοιες που θα χρησιμοποιηθούν σε όλη την μεταπτυχιακή διατριβή και συνεχίζοντας στο κεφάλαιο 2, όπου δίνεται μια πιο λεπτομερής περιγραφή τόσο της συσκευής και της λειτουργίας της, όσο και του υλικού που χρησιμοποιήθηκε για την κατασκευή της.

Στο κεφάλαιο 3, αναλύεται ο τρόπος με τον οποίο επιλέχθηκε ο αλγόριθμος για την αναγνώριση των χειρονομιών καθώς επίσης και μια λεπτομερής περιγραφή του αλγορίθμου αυτού, ενώ στο κεφάλαιο 4 περιγράφονται όλες οι αποφάσεις που ελήφθησαν για το σύστημα αναγνώρισης χειρονομιών όπως είναι η επιλογή της δειγματοληψίας, των χαρακτηριστικών κτλ.

Στην συνέχεια, στο κεφάλαιο 5 αναλύονται όλες οι αποφάσεις και οι τροποποιήσεις του αλγορίθμου που οδήγησαν σε μείωση των υπολογιστικών πράξεων και της μνήμης που χρησιμοποιεί το σύστημα αναγνώρισης χειρονομιών, ενώ το κεφάλαιο 6 αναφέρεται στην υλοποίηση του αλγορίθμου στο development kit.

Τέλος, το κεφάλαιο 7 αναφέρεται στην επιλογή των προτύπων αναφοράς και στο βοηθητικό πρόγραμμα που επιτρέπει στον προγραμματιστή να επιλέξει πρότυπα αναφοράς, να επιβεβαιώσει ότι επιλέχθηκαν σωστά πρότυπα και να μετρήσει τον χρόνο που περνάει η συσκευή στην συνάρτηση αναγνώρισης προτύπων κάθε φορά.

Κεφάλαιο 1: Εισαγωγικές έννοιες

1.1 Αλγόριθμοι αναγνώρισης προτύπων

Η αναγνώριση προτύπων είναι η επιστημονική αρχή της οποίας ο κύριος στόχος είναι η κατηγοριοποίηση αντικειμένων σε έναν αριθμό κατηγοριών ή κλάσεων. Ανάλογα με την εφαρμογή τα αντικείμενα αυτά μπορούν να είναι εικόνες, κυματομορφές σήματος ή γενικά μετρήσεις οι οποίες πρέπει να κατηγοριοποιηθούν.

Ενώ πριν το 1960 ήταν κυρίως το αποτέλεσμα θεωρητικής έρευνας στον τομέα της στατιστικής, η ραγδαία εξέλιξη των υπολογιστών αύξησε την ανάγκη για περισσότερες πρακτικές εφαρμογές της αναγνώρισης προτύπων, η οποία με την σειρά της έθεσε νέες απαιτήσεις για περεταίρω έρευνα.

Οι τομείς αναγνώριση προτύπων, μηχανική μάθηση (machine learning) και εξόρυξη γνώσεων στις βάσεις δεδομένων είναι δύσκολο να διαχωριστούν, καθώς συνήθως αλληλεπικαλύπτονται. Η μηχανική μάθηση είναι ο κοινός όρος με τον οποίο αναφερόμαστε σε μεθόδους εκμάθησης υπό επιτήρηση και προέρχεται από τον τομέα της τεχνητής νοημοσύνης, ενώ η εξόρυξη γνώσης είναι περισσότερο επικεντρωμένη σε μεθόδους μάθησης χωρίς επιτήρηση οι οποίες συνήθως βρίσκουν εφαρμογή στον επιχειρηματικό τομέα. Η αναγνώριση προτύπων έχει τις ρίζες της στην μηχανική και στο διάσημο τομέα της υπολογιστικής όρασης (computer vision). Όλοι αυτοί οι τομείς έχουν εξελιχθεί ουσιαστικά από τις ρίζες τους στην τεχνητή νοημοσύνη, μηχανική και στατιστική και έχουν γίνει παρόμοιοι ενσωματώνοντας ιδέες και τεχνικές ο ένας από τον άλλο.

Στην μηχανική μάθηση η αναγνώριση προτύπων είναι η ανάθεση μιας ετικέτας σε μια δοσμένη τιμή εισόδου. Ένα παράδειγμα αναγνώρισης προτύπων αποτελεί η κατηγοριοποίηση (classification), η οποία προσπαθεί να αναθέσει κάθε είσοδο σε μία από τις γνωστές κλάσεις (π.χ. να προσδιορίσει εάν ένα e-mail είναι spam ή όχι-spam). Ωστόσο, η αναγνώριση προτύπων είναι ένα πιο γενικό πρόβλημα το οποίο περιλαμβάνει και άλλους τύπους εξόδου. Γενικά, σκοπός της είναι να παρέχει μια λογική απάντηση σε όλα τα πιθανά δεδομένα εισόδου λαμβάνοντας υπόψη την στατιστική τους μεταβολή.

1.1.1 Κατηγορίες μάθησης

Η αναγνώριση προτύπων χωρίζεται σε τρεις κατηγορίες ανάλογα με τον τύπο της διαδικασίας μάθησης που χρησιμοποιείται για να προκύψει το αποτέλεσμα.

Η μάθηση υπό επιτήρηση (supervised learning) προϋποθέτει πως είναι διαθέσιμα ένα σύνολο από δεδομένα εκπαίδευσης τα οποία έχουν κατηγοριοποιηθεί με το χέρι. Στην συνέχεια, μια διαδικασία εκμάθησης παράγει ένα μοντέλο το οποίο προσπαθεί να έχει όσο το δυνατόν καλύτερο αποτέλεσμα στα δεδομένα εκπαίδευσης καθώς και σε άλλα άγνωστα δεδομένα.

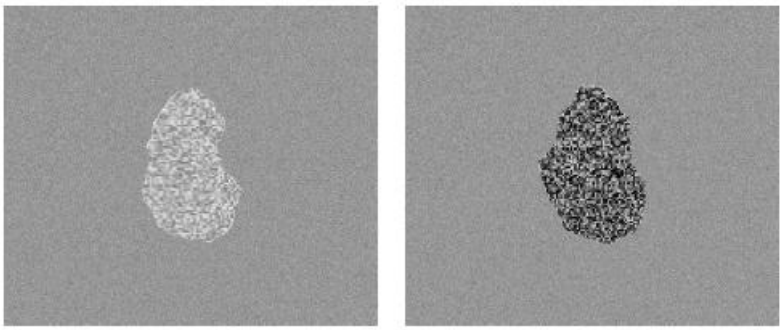
Στην μάθηση χωρίς επιτήρηση (unsupervised learning) τα δεδομένα εκπαίδευσης δεν έχουν κατηγοριοποιηθεί με το χέρι και επιχειρείται να βρεθούν διάφορα πρότυπα στα δεδομένα ώστε να μπορεί να προσδιοριστεί το αποτέλεσμα για νέα δεδομένα.

Ο συνδυασμός και των δύο, **μάθηση με ημί-επιτήρηση** (semi-supervised learning) χρησιμοποιεί έναν συνδυασμό λίγων δεδομένων εκπαίδευσης που έχουν κατηγοριοποιηθεί με ένα μεγαλύτερο αριθμό δεδομένων που δεν έχουν κατηγοριοποιηθεί. Ορισμένες φορές, διαφορετικές ονομασίες χρησιμοποιούνται για να περιγράψουν τις ανάλογες διαδικασίες εκμάθησης με επίβλεψη ή χωρίς οι οποίες όμως έχουν το ίδιο αποτέλεσμα. Παραδείγματος χάριν το χωρίς επίβλεψη αντίστοιχο της κατηγοριοποίησης (classification) είναι η συσταδοποίηση (clustering).

1.1.2 Δεδομένα εισόδου αλγορίθμων αναγνώρισης προτύπων

Τα δεδομένα εισόδου που θα χρησιμοποιήσει ο αλγόριθμος ώστε να επιστρέψει κάποιο αποτέλεσμα είναι ένα διάνυσμα χαρακτηριστικών. Η διαδικασία εξαγωγής χαρακτηριστικών (feature extraction) συντελεί στο να δημιουργηθούν νέα χαρακτηριστικά εφαρμόζοντας συναρτήσεις στα υπάρχοντα, ενώ η διαδικασία επιλογής χαρακτηριστικών περιλαμβάνει αλγορίθμους για την σωστή επιλογή των χαρακτηριστικών. Η επιλογή των χαρακτηριστικών αυτών είναι πολύ σημαντική καθώς μερικά από αυτά μπορούν να περιέχουν πληροφορία η οποία να είναι πλεονάζουσα ή μη χρήσιμη. Μερικά από τα πλεονεκτήματα της σωστής επιλογής χαρακτηριστικών είναι ο μικρότερος χρόνος εκπαίδευσης, η καλύτερη λειτουργία του μοντέλου καθώς αποφεύγεται η υπέρ-προσαρμογή (overfitting) δηλαδή η πολύ καλή προσαρμογή του αλγορίθμου στα δεδομένα εισόδου, χωρίς όμως αυτό να αντιδρά τόσο καλά στα υπόλοιπα δεδομένα.

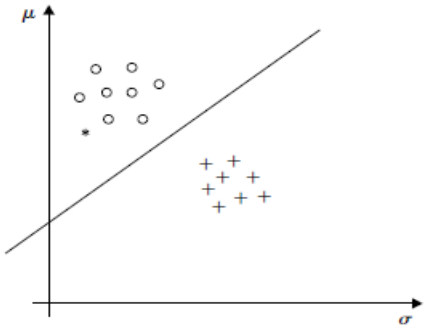
Στην Εικόνα 1, παρουσιάζονται οι εικόνες δύο κακώσεων που έχουν μια διακριτή περιοχή μέσα τους, η μία καλοήθης, κλάση-Α, και η δεύτερη κακοήθης, κλάση-Β (όγκος). Επίσης υποθέτουμε πως έχουμε στην διάθεσή μας περισσότερα πρότυπα (εικόνες) σε μια βάση δεδομένων με έναν αριθμό προτύπων, μερικά από τα οποία προέρχονται από την κλάση Α ενώ τα υπόλοιπα από την κλάση Β.



Εικόνα 1: Δύο κακώσεις, αριστερά παρουσιάζεται η εικόνα μιας καλοήθους κάκωσης, ενώ δεξιά παρουσιάζεται η εικόνα μιας κακοήθους κάκωσης (καρκίνου)

Το πρώτο βήμα είναι να εντοπίσουμε τις μετρήσιμες ποσότητες που κάνουν τις δύο περιοχές να διαφοροποιούνται η μία από την άλλη. Η Εικόνα 2 δείχνει ένα γράφημα της μέσης τιμής της πυκνότητας σε κάθε περιοχή ενδιαφέροντος έναντι της αντίστοιχης τυπικής απόκλισης γύρω από αυτή τη μέση τιμή. Κάθε σημείο αντιστοιχεί σε διαφορετική εικόνα από την βάση δεδομένων. Όπως φαίνεται, τα πρότυπα της κλάσης Α τείνουν να βρίσκονται σε διαφορετική περιοχή από αυτά της κλάσης Β. Η ευθεία γραμμή φαίνεται να μπορεί να διαχωρίσει τις δύο κλάσεις.

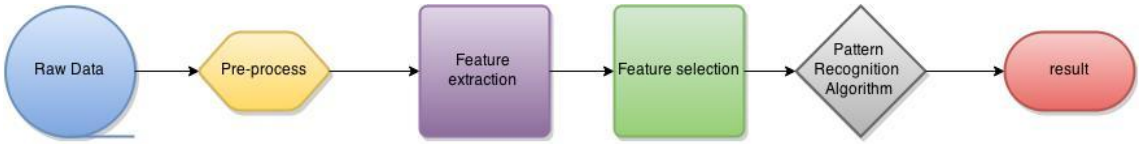
Αν υποθέσουμε πως μας δίδεται ένα νέο πρότυπο, το οποίο δεν γνωρίζουμε σε ποια κλάση ανήκει. Είναι λογικό να μετρήσουμε την μέση τιμή και την τυπική απόκλιση στην περιοχή ενδιαφέροντος και να σχεδιάσουμε το συγκεκριμένο σημείο στο γράφημα χρησιμοποιώντας τον αστερίσκο (*). Συνεπώς μπορούμε να υποθέσουμε πως είναι πιο πιθανό το άγνωστο πρότυπο να ανήκει στην κλάση Α από ότι στην κλάση Β.



Εικόνα 2: Διάγραμμα που παρουσιάζει την μέση τιμή των εικόνων των δύο διαφορετικών κλάσεων που διαθέτουμε ως προς την τυπική απόκλιση

Η προηγούμενη κατηγοριοποίηση σκιαγράφησε την λογική που έγκειται πίσω από ένα μεγάλο μέρος των προβλημάτων της αναγνώρισης προτύπων. Τα μέτρα που χρησιμοποιήθηκαν για την κατηγοριοποίηση, στην συγκεκριμένη περίπτωση η μέση τιμή και η τυπική απόκλιση αποτελούν τα χαρακτηριστικά. Σε μια πιο γενική περίπτωση l χαρακτηριστικά x_i χρησιμοποιούνται, i=1,2,.....,l και σχηματίζουν το διάνυσμα χαρακτηριστικών.

Συνεπώς, ολόκληρη η διαδικασία αναγνώρισης προτύπων, πραγματοποιείται όπως παρουσιάζεται στην Εικόνα 3.



Εικόνα 3: Η διαδικασία της αναγνώρισης προτύπων

1.1.3 Εφαρμογές της αναγνώρισης προτύπων

Η αναγνώριση προτύπων έχει βρει εφαρμογές σε πάρα πολλούς τομείς και όσο συνεχίζεται η ενσωμάτωση αισθητήρων σε συσκευές καθημερινής χρήσης θα χρησιμοποιείται ακόμα περισσότερο. Ένας τομέας στον οποίο έχει βρει εφαρμογή είναι αυτός της ιατρικής και της βιο-πληροφορικής με την υποβοηθούμενη από υπολογιστή διάγνωση ορισμένων παθήσεων όπως ο καρκίνος του μαστού. Μερικές ακόμα εφαρμογές αποτελούν η κατηγοριοποίηση κειμένου όπως π.χ. του ηλεκτρονικού ταχυδρομείου σε spam ή no spam, η αυτόματη αναγνώριση χειρόγραφων ταχυδρομικών κωδικών από τους ταχυδρομικούς φακέλους, η αναγνώριση προσώπων και πινακίδων κυκλοφορίας, η επεξεργασία δαχτυλικών αποτυπωμάτων καθώς και στον τομέα της άμυνας διάφορα συστήματα πλοήγησης και εντοπισμού στόχου. Τα παραπάνω αποτελούν ένα πολύ μικρό δείγμα των πολυάριθμων εφαρμογών της αναγνώρισης προτύπων, η οποία αναμένεται να ανθήσει στο προσεχές μέλλον.

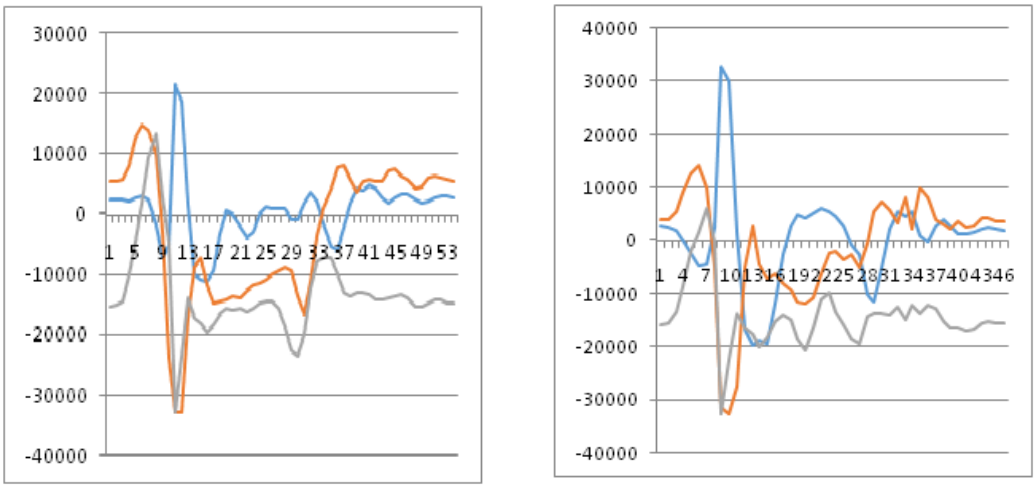
1.2 Αντιπαραβολή προτύπων

Σε ορισμένες περιπτώσεις διαθέτουμε ορισμένα πρότυπα, τα οποία ονομάζονται πρότυπα αναφοράς και πρέπει να αποφασίσουμε σε ποιο από αυτά τα πρότυπα ταιριάζει περισσότερο ένα άγνωστο (πρότυπο υπό εξέταση). Αυτά τα πρότυπα ποικίλουν και μπορεί να είναι αντικείμενα σε μία εικόνα, λέξεις ή φράσεις στον γραπτό ή προφορικό λόγο. Στην παρούσα εργασία, τα πρότυπα που θέλουμε να συγκρίνουμε είναι αλληλουχίες δεδομένων που προέρχονται από τα δεδομένα που μας παρέχονται από ένα επιταχυνσιόμετρο.

Ένα πρώτο βήμα για την επίλυση ενός τέτοιου προβλήματος αποτελεί η εύρεση ενός μέτρου ή ενός κόστους που μετράει την απόσταση ή την ομοιότητα ανάμεσα στα πρότυπα αναφοράς και στο πρότυπο εξέτασης έτσι ώστε στην συνέχεια να εκτελέσουμε την διαδικασία της αντιπαραβολής προτύπων. Κάθε πρότυπο αποτελείται από ένα διάνυσμα ή μήτρα με τα επιλεγμένα χαρακτηριστικά.

Ένα **πρότυπο αναφοράς** *r(i)*είναι ένα πρότυπο μιας γνωστής σε εμάς χειρονομίας. Αντίθετα, ένα **υπό εξέταση πρότυπο** *t(j)* είναι το άγνωστο πρότυπο το οποίο θέλουμε να αναγνωρίσουμε. Συγκρίνοντας το υπό εξέταση πρότυπο με ένα σύνολο από πρότυπα αναφοράς μπορούμε να αποφασίσουμε με ποιο από τα πρότυπα αναφοράς μοιάζει περισσότερο.

Μία από τις ιδιομορφίες της αντιπαραβολής προτύπων (από την αναγνώριση ομιλίας) έγκειται στο γεγονός ότι ακόμα και το ίδιο άτομο να προφέρει δύο φορές την ίδια λέξη οι κυμματομορφές θα διαφέρουν καθώς την μία φορά μπορεί να την προφέρει πιο γρήγορα και την επόμενη πιο αργά. Συνεπώς ο αλγόριθμος θα πρέπει να μπορεί να επεκτείνει ή να συρρικνώνει το υπό εξέταση πρότυπο στον χρόνο. Π.χ. στην Εικόνα 4 μπορούμε να δούμε δύο click, προερχόμενα από το ίδιο άτομο. Μπορούμε να παρατηρήσουμε πως την δεύτερη φορά, η ίδια κίνηση πραγματοποιήθηκε πιο γρήγορα (με μεγαλύτερη επιτάχυνση) και σε μικρότερο χρόνο καθώς η κίνηση είναι 54 δείγματα την πρώτη φορά και 46 δείγματα την δεύτερη.



Εικόνα 4: "Κλικ" προερχόμενα από το ίδιο άτομο με διαφορετική διάρκεια

Έχοντας ολοκληρώσει την προ-επεξεργασία των δεδομένων και την επιλογή των χαρακτηριστικών, ο στόχος είναι να ευθυγραμμίσουμε τα δύο πρότυπα μεταξύ τους ώστε να έχουμε το μικρότερο κόστος. Για να συμβεί αυτό, θα χρειαστεί να επεκτείνουμε (τεντώσουμε) ή να συρρικνώσουμε το υπό εξέταση πρότυπο στον χρόνο χρησιμοποιώντας διαδικασίες δυναμικού προγραμματισμού.

Η τεχνική της αντιπαραβολής προτύπων έχει διάφορες εφαρμογές μεταξύ των οποίων είναι η αναγνώριση ομιλίας καθώς επίσης και στον τομέα των αυτοματισμών όπου χρησιμοποιείται βιονική όραση.

1.3 Μονάδα μέτρησης αδράνειας (IMU)

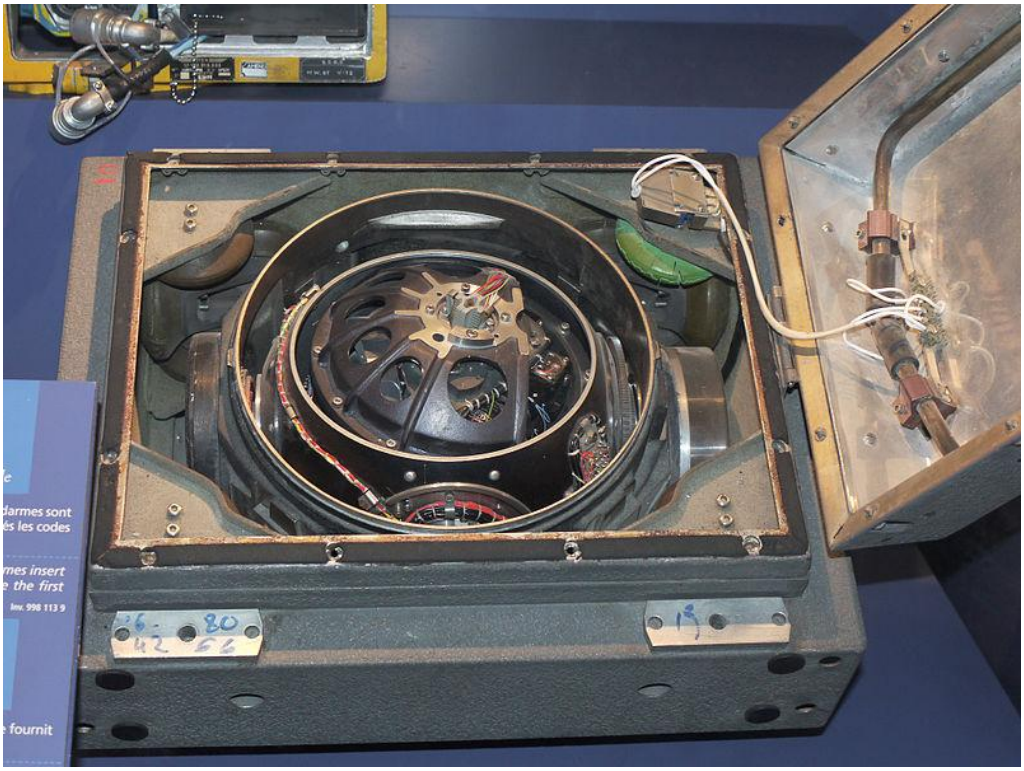
Μία μονάδα μέτρησης αδράνειας είναι μια ηλεκτρονική συσκευή η οποία μετράει την ταχύτητα, τον προσανατολισμό καθώς και τις δυνάμεις της βαρύτητας χρησιμοποιώντας έναν συνδυασμό αισθητήρων όπως είναι τα επιταχυνσιόμετρα, γυροσκόπια και ίσως μαγνητόμετρα. Συνήθως χρησιμοποιούνται για να τον έλεγχο των ελιγμών στα αεροσκάφη καθώς και στα διαστημόπλοια και τους δορυφόρους. Μια από τις πρόσφατες εφαρμογές της αποτελούντα GPS με αισθητήρες

αδράνειας, έτσι ώστε να λειτουργούν σωστά ακόμα και σε περιοχές που δεν υπάρχει GPS σήμα, όπως σε τούνελ ή κτίρια.

Ένα επιταχυνσιόμετρο μετρά την επιτάχυνση την οποία αναπτύσσει ο αισθητήρας στους 3 άξονες (X,Y,Z). Μία από τις πιο γνωστές εφαρμογές τους αποτελεί ο εντοπισμός ελεύθερης πτώσης στους φορητούς υπολογιστές για προστασία του σκληρού δίσκου.

Ένα γυροσκόπιο μετράει την περιστροφή σε μοίρες ανά δευτερόλεπτο του αισθητήρα. Μία εφαρμογή τους αποτελεί η εναλλαγή του προσανατολισμού της οθόνης του κινητού ανάλογα με τον τρόπο που το κρατάει ο χρήστης.

Ένα μαγνητόμετρο χρησιμοποιείται για την μέτρηση του μαγνητικού πεδίου της γης και ενσωματώνονται στις συσκευές μέτρησης αδράνειας έτσι ώστε να μειωθούν τυχόν λάθη προσανατολισμού (orientation drift).



Εικόνα 5: Η μονάδα πλοήγησης ενός γαλλικού μεσαίου βελησνεκούς πυραύλου που χρησιμοποιεί IMU

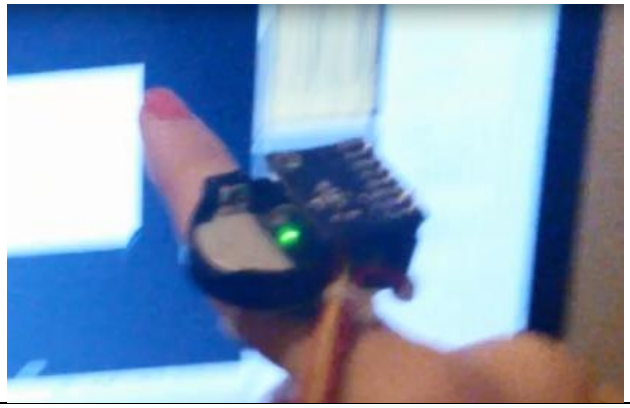
Κεφάλαιο 2: Περιγραφή της συσκευής εισόδου

2.1 Περίληψη της συσκευής και της λειτουργίας της

Ο αλγόριθμος αναγνώρισης προτύπων που προτείνεται στην παρούσα μεταπτυχιακή διατριβή χρησιμοποιήθηκε για την δημιουργία μιας φιλικής προς το χρήστη Bluetooth low energy συσκευής εισόδου (ποντίκι) [13] χρησιμοποιώντας το τσιπ της Dialog Semiconductor. Η συσκευή αυτή μπορεί να χρησιμοποιηθεί χωρίς να χρειάζεται για την λειτουργία της μια επίπεδη επιφάνεια, όπως συμβαίνει με την πλειοψηφία των συσκευών εισόδου που κυκλοφορούν στην αγορά.

Το ποντίκι που αναπτύχθηκε είναι μικρό σε μέγεθος, έχει το σχήμα δαχτυλιδιού που τοποθετείται στον δείκτη του χεριού και χρησιμοποιεί την μονάδα μέτρησης αδράνειας MPU6050 της Invensense για να αναφέρει την κίνηση του δαχτύλου καθώς και για να αναγνωρίσει τις διάφορες χειρονομίες που εκτελεί ο χρήστης . Πιο συγκεκριμένα, για να εντοπίσει την κίνηση του κέρσορα αναλύει τα δεδομένα που του παρέχονται από τον αισθητήρα του γυροσκοπίου, ενώ για την αναγνώριση των διάφορων χειρονομιών χρησιμοποιεί τα δεδομένα του επιταχυνσιόμετρου.

Υποστηρίζει το Human Interface Device (HID) προφίλ του Bluetooth, το οποίο περιγράφει το πως μπορεί να χρησιμοποιηθεί το USB HID πρωτόκολλο για να ανακαλυφθούν τα χαρακτηριστικά μιας hid συσκευής καθώς επίσης και τον τρόπο με τον οποίο μια συσκευή μπορεί να χρησιμοποιήσει τις HID υπηρεσίες. Αυτό, συνεπάγεται πως όλες οι συσκευές που υποστηρίζουν bluetooth low energy, ανεξαρτήτως λειτουργικού συστήματος (windows, Ios, android 4.3 και άνω), μπορούν να αναγνωρίσουν το ποντίκι ως συσκευή εισόδου και να συνδεθούν με αυτό.



Εικόνα 6: Η συσκευή εισόδου που αναπτύχθηκε και τοποθετείται στον δείκτη του χεριού

Κατά την διάρκεια της ανάπτυξης δόθηκε έμφαση στην κατανάλωση ενέργειας της συσκευής, για αυτό τον λόγο χρησιμοποιήθηκε η λειτουργία που παρέχεται από το τσιπ και επιτρέπει στην συσκευή να κλείνει τα διάφορα περιφερειακά της και να πέφτει σε εκτεταμένο ύπνο όταν δεν έχει να εκτελέσει κάποια εργασία[3]. Συνεπώς, μπορεί να λειτουργήσει σε τρείς διαφορετικές καταστάσεις:

- ενεργό κατάσταση (active state)
- κατάσταση εκτεταμένου ύπνου (extended sleep state)
- κατάσταση πλήρους ύπνου (permanent sleep state)

Για όσο χρονικό διάστημα ο χρήστης είτε κουνάει τον κέρσορα είτε εκτελεί κάποια χειρονομία η συσκευή παραμένει σε ενεργό κατάσταση. Αυτή είναι η κατάσταση με την μεγαλύτερη κατανάλωση ενέργειας η οποία αντιστοιχεί στα 59,31 μA. Στην ενεργό κατάσταση, οι διάφοροι τομείς του συστήματος (ο επεξεργαστής ARM, οι μνήμες SysRAM, ROM, κ.α.), η κεραία (συμπεριλαμβανομένου του Bluetooth Low Energy – BLEcore) καθώς και τα διάφορα περιφερειακά (UART1/2, I2C, SPI κ.α.) είναι ενεργά.

Στην περίπτωση που παρέλθει ένα δευτερόλεπτο από την τελευταία φορά που εντοπίστηκε κίνηση ή χειρονομία, η συσκευή θα απενεργοποιήσει τους διάφορους τομείς του συστήματος (εκτός από την SysRAM), την κεραία καθώς και τα διάφορα περιφερειακά και θα εισέλθει στην κατάσταση εκτεταμένου ύπνου. Στην κατάσταση αυτή η συσκευή εξακολουθεί να είναι συνδεδεμένη και μπορεί να ξυπνήσει είτε σύγχρονα με την χρήση του BLE χρονομέτρου, είτε ασύγχρονα με την διενέργεια ενός motion_interrupt (απότομη κίνηση του δαχτύλου). Σε κάθε μία από αυτές τις περιπτώσεις, η συσκευή θα ξυπνήσει και θα μεταβεί στην ενεργό κατάσταση. Η κατανάλωση ενέργειας στην κατάσταση εκτεταμένου ύπνου αντιστοιχεί σε 12μA.

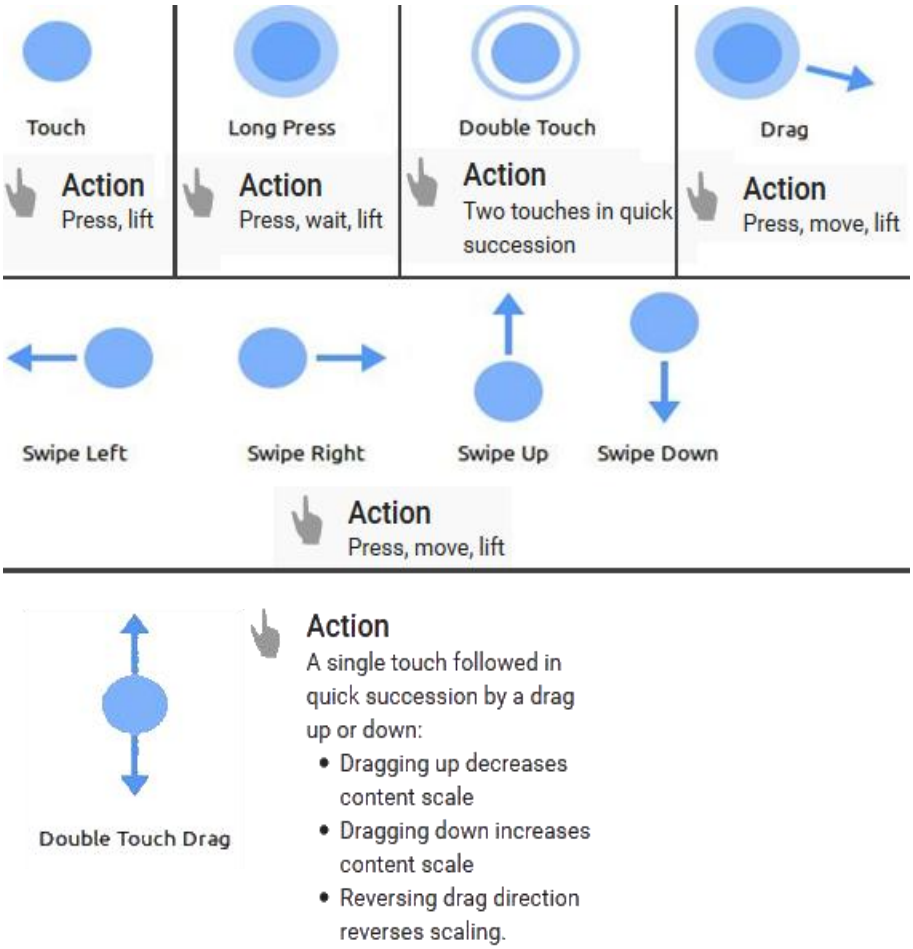
Για να μειωθεί περεταίρω η κατανάλωση ενέργειας, αφού περάσει ένα λεπτό από την στιγμή που η συσκευή εισέλθει στην κατάσταση εκτεταμένου ύπνου και δεν εντοπιστεί κίνηση του δαχτύλου για να ξυπνήσει η συσκευή, αυτή θα αποσυνδεθεί και θα κοιμάται μόνιμα. Θα ξυπνήσει μόνο σε περίπτωση απότομης κίνησης του δαχτύλου, θα διαφημιστεί και θα συνδεθεί ξανά με την προηγούμενη συσκευή.

Ο αλγόριθμος αναγνώρισης προτύπων τρέχει αποκλειστικά στο τσιπ της Dialog Semiconductor και ξεκινά την αναγνώριση από την στιγμή που αρχίζει μια χειρονομία και εξακολουθεί να τρέχει καθ' όλη τη διάρκεια αυτής. Μπορεί να αναγνωρίσει τρεις διαφορετικές χειρονομίες:

- press: πάτημα του δαχτύλου
- release: σήκωμα του δαχτύλου
- click: πολύ γρήγορος συνδυασμός των δύο ανωτέρω

Αυτές οι τρεις χειρονομίες μπορούν να συνδυαστούν για να εκτελέσουν πιο πολύπλοκες χειρονομίες. Παραδείγματος χάριν σε περιβάλλον android, όλες οι λειτουργίες πραγματοποιούνται με πάτημα/ σήκωμα του δαχτύλου στην οθόνη. Συνεπώς, με τις τρείς χειρονομίες που μπορεί να αναγνωρίσει το σύστημα μπορούν να εκτελεστούν όλες οι λειτουργίες που παρουσιάζονται στην Εικόνα 7 και είναι οι εξής:

- touch, το ισοδύναμο του click
- long press, το ισοδύναμο ενός press ακολουθούμενο από ένα release αφού πρώτα έχει περάσει λίγη ώρα
- double touch, το ισοδύναμο δύο πολύ γρήγορων click
- drag, το ισοδύναμο με press κίνηση του δαχτύλου σε μια ή περισσότερες κατευθύνσεις και στην συνέχεια release
- swipe,το ισοδύναμο με press κίνηση του δαχτύλου σε μια ή περισσότερες κατευθύνσεις και στην συνέχεια release
- double touch drag, click ακολουθούμενο από ένα πολύ γρήγορο press και κίνηση του δαχτύλου προς τα πάνω για zoom out ενώ κίνηση του δαχτύλου προς τα κάτω για zoom in.



Εικόνα 7: Οι διαφορετικές λειτουργίες που μπορεί να εκτελέσει η συσκευή εισόδου σε περιβάλλον android χρησιμοποιώντας τις τρεις βασικές χειρονομίες που μπορεί να αναγνωρίσει

Μόλις αναγνωριστεί μια χειρονομία, το σύστημα αποστέλλει την αντίστοιχη HID αναφορά. Από την στιγμή που θα ξεκινήσει η αναγνώριση μιας χειρονομίας και έως ότου το σύστημα αποφανθεί αν πρόκειται για έγκυρη χειρονομία ή όχι, δεν αποστέλλουμε αναφορές με την κίνηση του κέρσορα.

Διάφοροι τομείς όπου θα μπορούσε να βρει εφαρμογή μια τέτοια συσκευή εισόδου αποτελούν οι εξής:

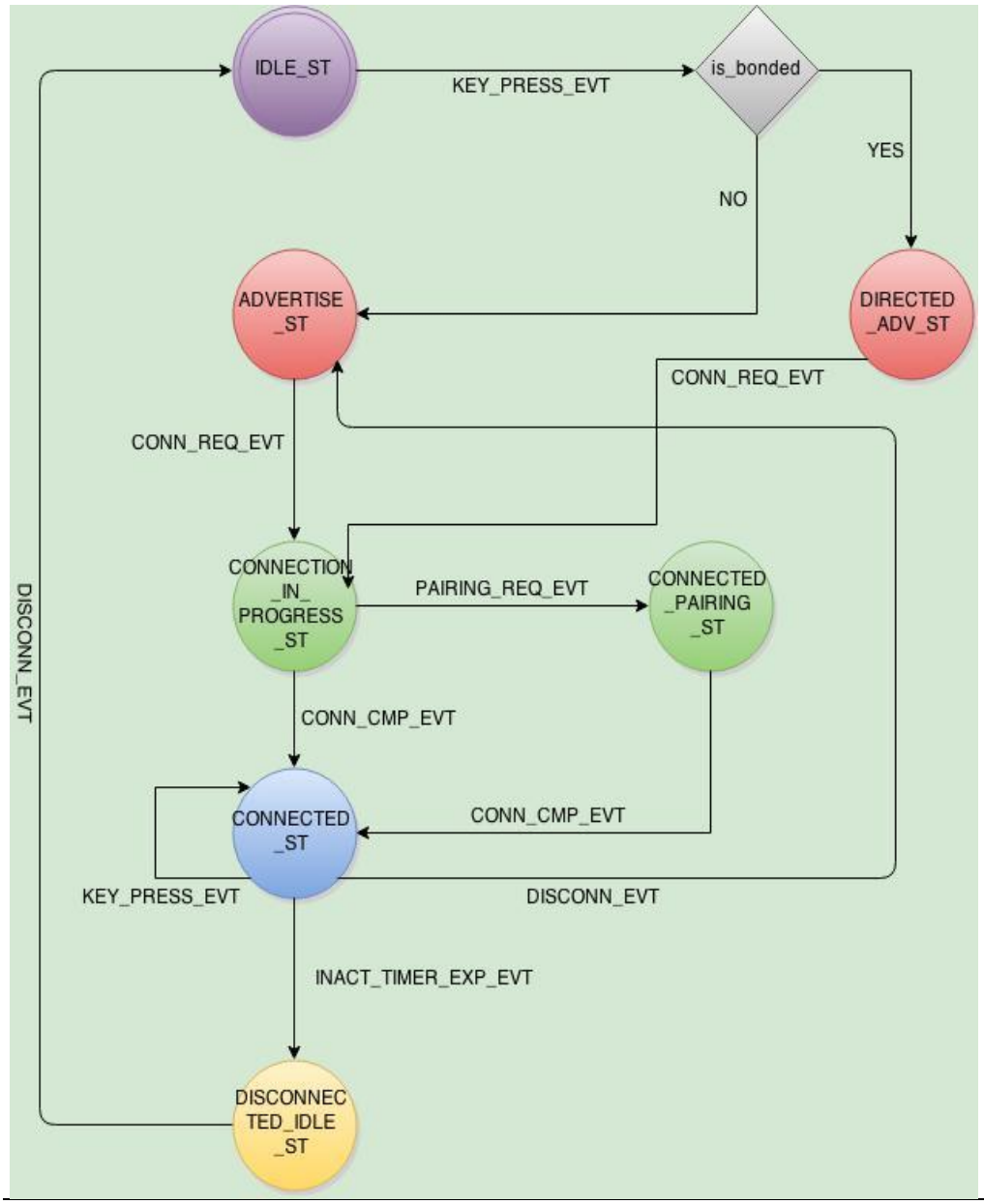
- ποντίκι παρουσίασης, όπου ο χρήστης δεν χρειάζεται να είναι δίπλα στον φορητό υπολογιστή για να αλλάξει σελίδες ή να δείξει κάπου
- λειτουργία οθόνης αφής φορώντας γάντια, ή όταν τα χέρια του χρήστη δεν είναι ελεύθερα
- ποντίκι όπου ο χρήστης μπορεί να λειτουργεί τον υπολογιστή από απόσταση όταν δεν υπάρχει διαθέσιμη μια λεία επιφάνεια.

Στην συνέχεια παρουσιάζεται ο τρόπος λειτουργίας της συσκευής. Αρχικά, η συσκευή είναι ανενεργή σε κατάσταση πλήρους ύπνου για εξοικονόμηση ενέργειας. Μόλις ο χρήστης κουνήσει απότομα το δάχτυλό του, η συσκευή θα ξυπνήσει και ανάλογα με το αν είναι ήδη συζευγμένη με μια συσκευή ή όχι, θα αρχίσει έναν από τους δύο υποστηριζόμενους τύπους διαφήμισης.

Αν η συσκευή δεν ήταν συζευγμένη, τότε θα διαφημίζεται προς όλες τις bluetooth low energy συμβατές συσκευές, ενώ αν ήταν θα διαφημιστεί μόνο προς τη συσκευή με την οποία

είναι ήδη συζευγμένη. Σε κάθε περίπτωση, μόλις έρθει ένα αίτημα για σύνδεση, θα συνδεθεί ως συσκευή εισόδου.

Η συσκευή θα παραμείνει συνδεδεμένη έως ότου ο χρήστης θελήσει να αποσυνδεθεί ή η συσκευή δεν εντοπίσει κίνηση για χρονικό διάστημα μεγαλύτερο του ενός λεπτού, όπου και θα αποσυνδεθεί. Η γενική λειτουργία της, παρουσιάζεται στην Εικόνα 8.



Εικόνα 8: FSM της συσκευής εισόδου

2.2 Υλικό που χρησιμοποιήθηκε

Στις ενότητες που ακολουθούν αναφέρεται το υλικό το οποίο χρησιμοποιήθηκε για την κατασκευή της συσκευής εισόδου, τα διάφορα χαρακτηριστικά τους καθώς επίσης και οι διάφοροι περιορισμοί τους οποίους διαθέτουν και πρέπει να διαχειριστεί η συσκευή.

Αρχικά, αναφερόμαστε στον επεξεργαστή και τις μνήμες του SmartBondDA14580 ο οποίος χρησιμοποιείται για την αναγνώριση των χειρονομιών και την αποστολή μέσω Bluetooth Low Energy των HID πακέτων στην συζευγμένη συσκευή, ενώ στην συνέχεια αναλύεται η μονάδα μέτρησης της αδράνειας GY-88 η οποία χρησιμοποιήθηκε για την ανίχνευση της κίνησης.

2.2.1 SmartBond DA14580 processor and memories

Το DA14580 είναι το μικρότερο, χαμηλότερης κατανάλωσης τσιπ Bluetooth Low Energy στον κόσμο το οποίο προσφέρει την δυνατότητα ανάπτυξης εφαρμογών Bluetooth 4.1. Χρησιμοποιώντας μόλις 4.9 mA κατά την αποστολή και λήψη πακέτων δίνει στις συσκευές πολύ μεγαλύτερη διάρκεια ζωής της μπαταρίας. Επιπροσθέτως, η δυνατότητά του να λειτουργεί με τάση τόσο μικρή όσο τα 0,9 V, το κάνει ιδανικό να λειτουργεί με μπαταρία coin-cell.

Το DA14580 υποστηρίζει μια ευέλικτη αρχιτεκτονική μνημών για να αποθηκεύει τόσο τα διάφορα προφίλ του Bluetooth, όσο και τον κώδικα της εφαρμογής ο οποίος μπορεί να ενημερωθεί over the air. Όλο το stack του bluetooth βρίσκεται σε μια αφιερωμένη ROM, ενώ όλο το λογισμικό τρέχει στον επεξεργαστή ARMCortexM0 χρησιμοποιώντας έναν μικρό δρομολογητή διεργασιών [2].

Στη συνέχεια, ακολουθούν ορισμένα χαρακτηριστικά που αφορούν τις μνήμες καθώς και τον επεξεργαστή του τσιπ:

- Επεξεργαστική δύναμη
  - 16MHz 32 bit Arm Cortex -M0
  - Dedicated Link Layer Processor
  - AES -128 bit encryption Processor
- Μνήμες
  - 32kbyte One Time Programmable memory
  - 42kbyte system SRAM
  - 84kbyte Rom
  - 8kbyte Retention SRAM

Εικόνα 9: DA14580 Development Kit

Οι μεγαλύτερες προκλήσεις της εφαρμογής είναι οι εξής: πρώτον να βεβαιωθούμε πως μπορεί να υπάρξει μια πιο "ελαφριά" έκδοση του αλγορίθμου αναγνώρισης προτύπων η οποία να είναι και μικρή σε μέγεθος ώστε να χωράει στην περιορισμένη μνήμη που διαθέτουμε, αλλά και να αναγνωρίζει σε ικανοποιητικό βαθμό τις κινήσεις του χρήστη. Ακόμα, θα πρέπει να απαιτεί μικρή υπολογιστική δύναμη έτσι ώστε ο επεξεργαστής να έχει τον απαραίτητο χρόνο για να στέλνει τα bluetooth πακέτα με την κίνηση, καθώς και να εκτελεί τον αλγόριθμο αναγνώρισης.

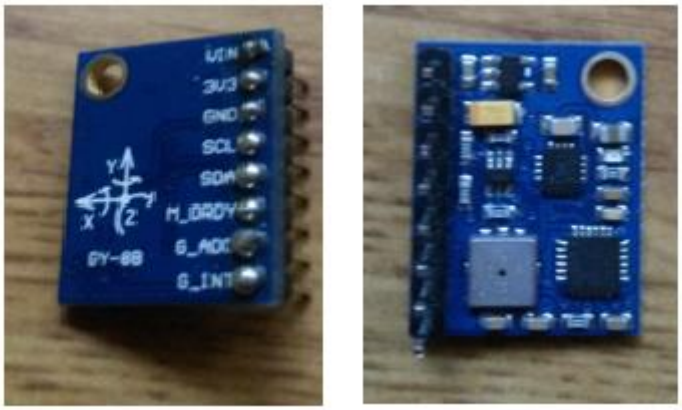
2.2.2IMU GY-88

Η μονάδα μέτρησης αδράνειας που επιλέχθηκε είναι η IMUGY-88 της Invensense [1]. Η συσκευή συνδυάζει γυροσκόπιο 3 αξόνων, επιταχυνσιόμετρο 3 αξόνων και έναν ψηφιακό επεξεργαστή κίνησης (Digital Motion Processor) σε ένα μικρό πακέτο διαστάσεων 4x4x0.9mm.

Διαθέτει τρείς μετατροπείς αναλογικού σε ψηφιακό σήμα των 16-bit για να ψηφιοποιεί την έξοδο του γυροσκοπίου και άλλους τρεις για το επιταχυνσιόμετρο. Για να μπορεί να παρακολουθήσει με ακρίβεια τόσο τις πιο γρήγορες κινήσεις όσο και τις πιο αργές, μπορεί να αλλάξει προγραμματιστικά το εύρος του γυροσκοπίου σε ένα από τα ακόλουθα: ±250, ±500, ±1000, ±2000°/sec. Αντίστοιχα, το εύρος του επιταχυνσιόμετρου μπορεί να διαμορφωθεί ως εξής: ±2g, ±4g, ±8g, ±16g.

Το τσιπ περιλαμβάνει μια ουρά FIFO των 1024 byte η οποία μειώνει την κατανάλωση ενέργειας του συστήματος αφού επιτρέπει στον επεξεργαστή να διαβάζει τα δεδομένα σε ομάδες και μετά να εισέρχεται σε λειτουργία χαμηλότερης κατανάλωσης ώσπου να μαζέψει περισσότερα δεδομένα.

Παρέχοντας την δυνατότητα να γίνει όλη η επεξεργασία της κίνησης μέσα στον Digital Motion Processor (DMP), ο επεξεργαστής του συστήματος είναι ελεύθερος να ασχοληθεί με άλλες εργασίες και όχι με τις πολύπλοκες αριθμητικές πράξεις που απαιτούνται για την αναγνώριση των κινήσεων.



Εικόνα 10: IMUGY-88

2.3 Οι χειρονομίες που πρέπει να αναγνωριστούν

Μία συσκευή εισόδου θα πρέπει να μας παρέχει την δυνατότητα να μετακινήσουμε έναν κέρσορα στην περιοχή που επιθυμούμε καθώς επίσης και την δυνατότητα να επιλέξουμε μια περιοχή της οθόνης έτσι ώστε να ανοίξουμε ή μετακινήσουμε φακέλους ή εφαρμογές. Για την διαδικασία της επιλογής στην οθόνη, η συσκευή θα πρέπει να μπορεί να αναγνωρίζει έναν αριθμό κινήσεων που εκτελεί ο χρήστης.

Όπως αναφέραμε και προηγουμένως, συνδυάζοντας έναν μικρό αριθμό βασικών χειρονομιών, μπορούμε να πραγματοποιήσουμε μια μεγαλύτερη συλλογή πολύπλοκων

κινήσεων. Για αυτόν τον λόγο, οι χειρονομίες που πρέπει να αναγνωριστούν από τον αλγόριθμο είναι οι εξής:

- press: πάτημα του δαχτύλου. Όπως παρουσιάζει η Εικόνα 11 το δάχτυλο από τη θέση ένα πηγαίνει στην θέση δύο όπου και παραμένει
- release: σήκωμα του δαχτύλου, το δάχτυλο από τη θέση δύο πηγαίνει στην θέση ένα όπου και παραμένει
- click: πολύ γρήγορος συνδυασμός των δύο ανωτέρω. Το δάχτυλο γρήγορα πηγαίνει από την θέση ένα στη θέση δύο και αμέσως πίσω στη θέση ένα όπου και παραμένει.

Εικόνα 11: Οι διαφορετικές θέσεις του δείκτη για την πραγματοποίηση των χειρονομιών.

Κεφάλαιο 3: Αλγόριθμος αναγνώρισης προτύπων

3.1 Επιλογή αλγορίθμου αναγνώρισης προτύπων- Σχετική έρευνα

Το πιο προκλητικό πρόβλημα της δυναμικής αναγνώρισης χειρονομιών αποτελεί η χωρική-χρονική μεταβλητότητα, όταν η ίδια χειρονομία μπορεί να διαφέρει στην ταχύτητα, στο σχήμα, στην διάρκεια και στην πληρότητα [10]. Ένας χρήστης, όσο και να προσπαθήσει, είναι σχεδόν αδύνατο να πραγματοποιήσει δύο ολόιδιες χειρονομίες. Για τον λόγο αυτό, ο αλγόριθμος που θα επιλεγεί θα πρέπει να μην ψάχνει για να βρει ακριβή αντιστοίχιση ανάμεσα στα δυο πρότυπα, αλλά αντίθετα να προσπαθεί να εντοπίσει τον βαθμό ομοιότητας ανάμεσά τους.

Για την επιλογή του κατάλληλου αλγορίθμου που θα μας βοηθήσει να ταιριάξουμε την υπό εξέταση χειρονομία με μία από τις γνωστές χειρονομίες που βρίσκονται στην βιβλιοθήκη χειρονομιών, θα πρέπει να εντοπίσουμε τις βασικές προϋποθέσεις που θα πρέπει να πληροί.

Αρχικά, καθώς πρόκειται για συσκευή εισόδου, θα πρέπει να βεβαιωθούμε πως έχει άμεση απόκριση, δηλαδή ο αλγόριθμος θα πρέπει να εκτελείται παράλληλα με την κίνηση και να αποφανθεί για αυτήν σε σύντομο χρονικό διάστημα από το πέρας αυτής. Ακόμα, λόγω των περιορισμών σε μνήμη, υπολογιστική δύναμη, και περιορισμένο χρόνο εκτέλεσης θα πρέπει να απαιτεί λίγη μνήμη, να έχει μικρή πολυπλοκότητα, να είναι γρήγορος και να απαιτεί λίγες υπολογιστικές πράξεις. Τέλος, όπως είναι αναμενόμενο, ο αλγόριθμος θα πρέπει να δίνει έγκυρο αποτέλεσμα.

Συνεπώς, θα πρέπει να ανατρέξουμε στην ήδη υπάρχουσα έρευνα στον τομέα της αναγνώρισης προτύπων με χρήση αισθητήρων αδράνειας προσπαθώντας να εντοπίσουμε μια λύση που να μπορεί να εφαρμοστεί στο σύστημά μας παρ' όλους τους περιορισμούς που θέτει καθώς επίσης και να πληροί όλες τις προϋποθέσεις που μόλις αναφέραμε.

Στο [4] το σύστημα αναγνώρισης προτύπων χρησιμοποιεί διάφορους αλγορίθμους για την αναγνώριση προτύπων, ανάμεσά τους ο Linear Time Warping, ο Dynamic Time Warping και τα Hidden Markov Models (τα οποία χρησιμοποιούνται και στο [9]). Η διαφορά όμως είναι πως στα δύο πρώτα η εκτέλεση του αλγορίθμου γίνεται offline ενώ ο εντοπισμός της αρχής και του τέλους μιας χειρονομίας γίνεται από άνθρωπο και όχι αυτόματα από το σύστημα. Η χρησιμοποίηση του αλγορίθμου Linear Time Warping είναι απαγορευτική για το περιορισμένης μνήμης σύστημά μας, αν αναλογιστούμε ότι απαιτεί περισσότερα από ένα πρότυπα αναφοράς για μία μόνο χειρονομία (η ίδια χειρονομία συρρικνωμένη ή τεντωμένη στον χρόνο). Ακόμα, υποθέτει πως όλη η χειρονομία θα εκτελεστεί πιο γρήγορα ή αργά ενώ υπάρχει η πιθανότητα να ισχύει αυτό για ένα μέρος της μόνο.

Η αναγνώριση μιας χειρονομίας στην έρευνα [6], πραγματοποιείται σε πραγματικό χρόνο με την χρήση μιας μηχανής πεπερασμένων καταστάσεων, όπου χρησιμοποιούνται ένα σύνολο από κατώφλια και κάθε φορά που η επιτάχυνση ξεπερνά ένα κατώφλι αλλάζει την κατάσταση στην οποία βρίσκεται. Όταν φτάνει στην τελική κατάσταση, εκτελείται η πράξη ανάλογα με την χειρονομία που εκτελέστηκε.

Στο [7], για την αναγνώριση μιας χειρονομίας αποθηκεύονται κάποια ειδικά μη αλληλεπικαλυπτόμενα χαρακτηριστικά για κάθε διαφορετική χειρονομία που μπορεί να αναγνωρίσει ο αλγόριθμος. Για να ανήκει η υπό εξέταση χειρονομία σε μία κατηγορία θα πρέπει να διαθέτει τα χαρακτηριστικά αυτά.

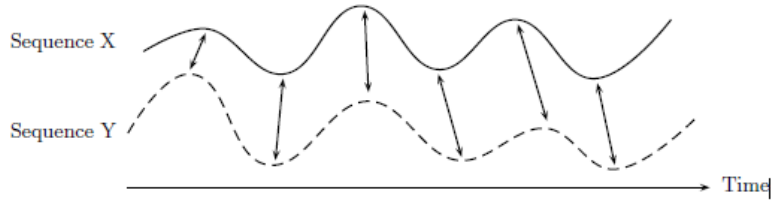
Ο αλγόριθμος DTW βρίσκει εφαρμογή στο [8], με την διαφορά ότι συμπεριλαμβάνει μια φάση προπαρασκευής όπου αρχικά βρίσκεται μια μέση τιμή για να μειωθεί ο όγκος των δεδομένων τα οποία θα πρέπει να επεξεργαστεί ο αλγόριθμος και στην συνέχεια ακολουθεί η διακριτοποίηση των δεδομένων της επιτάχυνσης. Ακόμα, το σύστημα αναγνώρισης περιλαμβάνει έναν μηχανισμό επιλογής προτύπων αναφοράς, δηλαδή μιας βιβλιοθήκης όπου βρίσκονται πολλά πρότυπα αναφοράς από διαφορετικούς χρήστες που έχουν συλλεχθεί σε διαφορετικές ημέρες και ένα σύστημα αξιολόγησης αυτών των προτύπων αναφοράς με στόχο να κρατηθούν αυτά με το μεγαλύτερο ποσοστό επιτυχίας αναγνώρισης. Στο δικό μας σύστημα όμως, ο περιορισμένος χρόνος εκτέλεσης δεν μας επιτρέπει να ελέγχουμε πολλές διαφορετικές εκδόσεις της ίδιας χειρονομίας. Το ίδιο ισχύει και με τον περιορισμό μνήμης που διαθέτουμε και

μας αναγκάζει να έχουμε ένα μόνο πρότυπο αναφοράς ανά χειρονομία. Ακόμα, στο [8] δίνεται έμφαση ότι όταν τα πρότυπα αναφοράς καθώς και το πρότυπο υπό εξέταση λαμβάνονται την ίδια ημέρα, τα αποτελέσματα αναγνώρισης του αλγορίθμου είναι υψηλότερα, καθώς με την πάροδο του χρόνου ένας χρήστης εκτελεί διαφορετικά την ίδια κίνηση.

Τα μοντέλα HMM απαιτούν εκτενή δεδομένα εκπαίδευσης για να είναι αποτελεσματικά. Ακόμα, συχνά απαιτούν εκ των προτέρων γνώση του λεξιλογίου, ώστε να μπορεί να επιτευχθεί ρύθμιση του μοντέλου π.χ. του αριθμού των καταστάσεων σε ένα μοντέλο. Συνεπώς, τα μοντέλα που χρησιμοποιούν HMM μειονεκτούν όταν οι χρήστες μπορούν να επιλέξουν τις χειρονομίες μόνοι τους ή όταν πρόκειται για αναγνώριση εξατομικευμένων χειρονομιών. Αντίθετα ο DTW είναι πολύ αποτελεσματικός όταν έχουμε λιγοστά δεδομένα εκπαίδευσης και μικρό λεξιλόγιο, τα οποία συνήθως ταιριάζουν με την αλληλεπίδραση χρησιμοποιώντας εξατομικευμένες χειρονομίες που παρέχουν τα περισσότερα καταναλωτικά προϊόντα. Λαμβάνοντας υπόψη τα παραπάνω καθώς επίσης και την μικρή πολυπλοκότητα του αλγορίθμου DTW επιλέξαμε αυτόν για την αναγνώριση των χειρονομιών που εκτελεί ο χρήστης.

3.2 Αρχική έκδοση του αλγορίθμου DTW

Ο Dynamic Time Warping (DTW) είναι ένας γνωστός αλγόριθμος που προσπαθεί να βρει μια βέλτιστη ευθυγράμμιση ανάμεσα σε δύο εξαρτώμενες από τον χρόνο ακολουθίες χαρακτηριστικών κάτω από ορισμένους περιορισμούς (Εικόνα 12).



Εικόνα 12: Η ευθυγράμμιση των δύο προτύπων που επιτυγχάνει ο αλγόριθμος DTW στον χρόνο

Στόχος του είναι να εντοπίσει το βέλτιστο μονοπάτι το οποίο αντιστοιχεί στο μικρότερο κόστος ανάμεσα στις δύο ακολουθίες επεκτείνοντας ή συμπιέζοντας το υπό εξέταση πρότυπο έτσι ώστε να ταιριάζει περισσότερο με το πρότυπο αναφοράς χρησιμοποιώντας την αρχή του Bellman και δυναμικό προγραμματισμό.

3.2.1 Μέτρα τα οποία βασίζονται σε τεχνικές βασισμένες στο βέλτιστο μονοπάτι

Έστω r(i), i=1,2,3,..,I, t(j), j=1,2,3,.....,J είναι η ακολουθία των διανυσμάτων χαρακτηριστικών του πρότυπου αναφοράς και του υπό εξέταση προτύπου αντίστοιχα, όπου γενικά I≠J. Ο σκοπός είναι να αναπτυχθεί ένα κατάλληλο μέτρο για την μέτρηση της απόστασης μεταξύ των δύο ακολουθιών.

Συνεπώς, σχηματίζουμε ένα δισδιάστατο πλέγμα με τα στοιχεία των δύο ακολουθιών, με το πρότυπο αναφοράς στον οριζόντιο άξονα i και το υπό εξέταση πρότυπο στον κάθετο άξονα j.

Η Εικόνα 13 αποτελεί ένα παράδειγμα για I=6, J=5. Κάθε σημείο του πλέγματος (κόμβος) σηματοδοτεί μια αντιστοιχία ανάμεσα στα σχετικά στοιχεία των δύο ακολουθιών. Για παράδειγμα, ο κόμβος (3,2) αντιστοιχεί το στοιχείο r(3) με το t(2).

Κάθε κόμβος (i,j) του πλέγματος συσχετίζεται με ένα κόστος ,το οποίο είναι μια κατάλληλα καθορισμένη συνάρτηση d(i,j), η οποία μετράει την "απόσταση" ανάμεσα στα στοιχεία των ακολουθιών t(j) και r(i).

Ένα μονοπάτι από έναν αρχικό κόμβο (i_0, j_0) σε ένα τελικό κόμβο (i_f , j_f) είναι ένα διατεταγμένο σύνολο κόμβων της μορφής

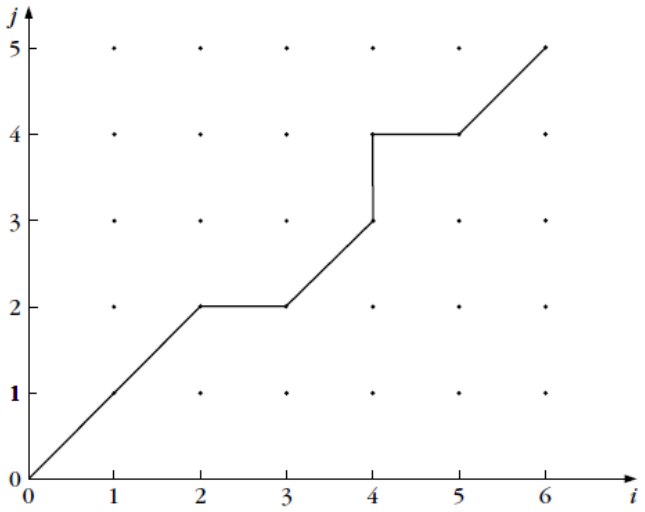
(i_0, j_0), (i_1, j_1), (i_2, j_2), . . . , (i_f , j_f)

Κάθε μονοπάτι συσχετίζεται με ένα συνολικό κόστος D, το οποίο ορίζεται ως εξής:

$$D = \sum_{k=0}^{K-1} d(i_k, j_k)$$

όπου το K είναι ο αριθμός των κόμβων που απαρτίζουν το μονοπάτι.

Π.χ. στην Εικόνα 13, ισχύει ότι K=8.Το συνολικό κόστος ώσπου να φτάσουμε στον κόμβο (i_k,j_k) δηλώνεται με D(i_k,j_k) ενώ θέτουμε ότι το D(0,0)=0 καθώς επίσης και d(0,0)=0. Το μονοπάτι είναι ολοκληρωμένο στην περίπτωση που ισχύει ότι (i_0, j_0)=(0,0) και (i_f , j_f)=(I,J).



Εικόνα 13: Βέλτιστο μονοπάτι ανάμεσα σε δύο ακολουθίες μήκους i και j

Η απόσταση ανάμεσα σε δύο ακολουθίες ορίζεται ως το ελάχιστο D ανάμεσα σε όλα τα πιθανά μονοπάτια. Την ίδια στιγμή, το μονοπάτι με το μικρότερο κόστος αποκαλύπτει την βέλτιστη αντιστοιχία ζευγών ανάμεσα στις δύο ακολουθίες, το οποίο είναι ένα κρίσιμο σημείο, καθώς οι δύο ακολουθίες έχουν διαφορετικό μήκος. Με άλλα λόγια, η διαδικασία του βέλτιστου μονοπατιού επιτυγχάνει την ευθυγράμμιση ή διαστρέβλωση των στοιχείων του υπό εξέταση προτύπου με αυτά του προτύπου αναφοράς, προσπαθώντας να πετύχει μεγαλύτερο βαθμό ομοιότητας. Υπάρχουν διάφορες παραλλαγές σε αυτό το σχήμα π.χ. δεν είναι απαραίτητο να έχουμε τον περιορισμό του να ψάχνουμε ένα ολοκληρωμένο μονοπάτι αλλά να υιοθετήσουμε ένα πιο "χαλαρό" περιορισμό που ονομάζεται περιορισμός τελικών σημείων.

Ακόμα, θα μπορούσαμε να ορίσουμε ένα κόστος, όχι μόνο σε κάθε κόμβο αλλά και στην μετάβαση ανάμεσα σε κόμβους, κάνοντας μερικές μεταβάσεις να έχουν μεγαλύτερο κόστος από άλλες. Σε τέτοιες περιπτώσεις, το κόστος σε έναν κόμβο (i_k,j_k) εξαρτάται επίσης και από την συγκεκριμένη μετάβαση, δηλαδή από ποιον κόμβο (i_{k-1},j_{k-1}) φτάσαμε στον κόμβο αυτό. Συνεπώς, το κόστος d είναι τώρα της μορφήςd(i_k,j_k |i_{k-1},j_{k-1}) και το συνολικό κόστος του μονοπατιού είναι

$$D = \sum_{k} d(i_k, j_k | i_{k-1}, j_{k-1})$$

Σε ορισμένες περιπτώσεις το συνολικό κόστος ορίζεται ως το γινόμενο

$$D = \prod_k d(i_k, j_k | i_{k-1}, j_{k-1})$$

Υπάρχουν περιπτώσεις όπου το d επιλέγεται έτσι ώστε να επιτυγχάνεται μεγιστοποίηση αντί για ελαχιστοποίηση του κόστους. Προφανώς, σε όλες αυτές τις παραλλαγές πρέπει να υιοθετηθούν οι κατάλληλες αρχικές συνθήκες. Για να εντοπίσουμε το βέλτιστο μονοπάτι πρέπει να ερευνήσουμε πρώτα όλα τα δυνατά μονοπάτια που υπάρχουν. Όμως αυτό αποτελεί μια διαδικασία με υψηλό υπολογιστικό κόστος. Για να μειωθεί η υπολογιστική περιπλοκότητα θα υιοθετηθούν αλγόριθμοι δυναμικού προγραμματισμού βασισμένοι στην αρχή του Bellman.

3.2.2 Αρχή βελτιστοποίησης του Bellman και δυναμικός προγραμματισμός

Έστω ότι το βέλτιστο μονοπάτι ανάμεσα στον αρχικό κόμβο (i_0,j_0) και τον τελικό κόμβο(i_f,j_f) συμβολίζεται ως εξής:

(i_0,j_0)$\overset{opt}{\longrightarrow}$ (i_f,j_f)

Αν (i,j) είναι ένας ενδιάμεσος κόμβος ανάμεσα στους (i_0,j_0) και (i_f,j_f), τότε θα δηλώσουμε ότι το βέλτιστο μονοπάτι θα είναι αναγκασμένο να περάσει από τον κόμβο (i,j)

(i_0,j_0)$\overset{opt}{\overset{(i,j)}{\longrightarrow}}$ (i_f,j_f)

Η αρχή του Bellman δηλώνει ότι

(i_0,j_0)$\overset{opt}{\overset{(i,j)}{\longrightarrow}}$ (i_f,j_f)= (i_0,j_0)$\overset{opt}{\longrightarrow}$ (i,j) +(i,j)$\overset{opt}{\longrightarrow}$($i_f$,$j_f$)

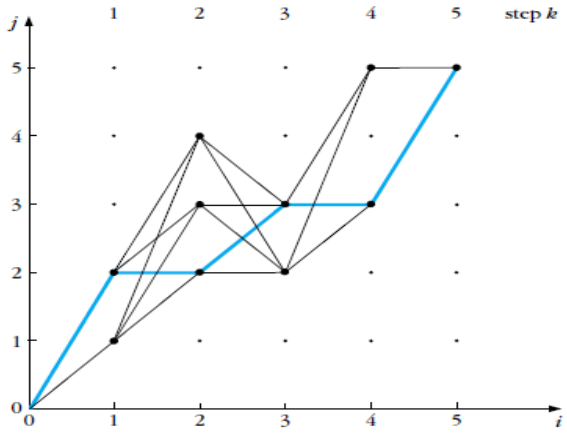
όπου το + υποδηλώνει την συνένωση των μονοπατιών. Με άλλα λόγια, η αρχή του Bellman δηλώνει ότι το βέλτιστο μονοπάτι από τον κόμβο (i_0,j_0) στον κόμβο (i_f,j_f) δια μέσω του (i,j) είναι η συνένωση των βέλτιστων μονοπατιών από τον (i_0,j_0) στον (i,j) και του βέλτιστου μονοπατιού από τον (i,j) στον (i_f,j_f). Ως αποτέλεσμα αυτής της αρχής είναι ότι μόλις βρισκόμαστε στον κόμβο (i,j) έχοντας ακολουθήσει το βέλτιστο μονοπάτι, τότε για να φτάσουμε στον κόμβο (i_f,j_f) πρέπει να ψάξουμε μόνο για το βέλτιστο μονοπάτι από τον (i,j) στον (i_f,j_f).

Έστω πως ξεκινήσαμε από τον κόμβο (i_0,j_0) και ότι ο $\kappa_{ος}$ κόμβος του μονοπατιού είναι ο (i_k,j_k). Ο στόχος είναι να υπολογίσουμε το ελάχιστο κόστος που απαιτείται για να φτάσουμε στον τελευταίο κόμβο. Η μετάβαση στον (i_k,j_k) μπορεί να λάβει μέρος από έναν από τους πιθανούς κόμβους που μπορούν να είναι στην (κ-1) θέση του μονοπατιού (αυτός είναι ο (i_{k-1},j_{k-1}) κόμβος). Για κάθε κόμβο στο πλέγμα, υποθέτουμε πως υπάρχει ένα σύνολο από επιτρεπτούς προγόνους, οι οποίοι καθορίζονται από τους τοπικούς περιορισμούς. Συνεπώς, σύμφωνα με την αρχή του Bellman

$$D_{min}(i_k, j_k) = \min_{i_{k-1}, j_{k-1}} [D_{min}(i_{k-1}, j_{k-1}) + d(i_k, j_k | i_{k-1}, j_{k-1})] \quad [1]$$

το οποίο σημαίνει ότι το συνολικό κόστος για να φτάσουμε στον κόμβο (i_k,j_k) είναι το ελάχιστο κόστος μέχρι τον κόμβο (i_{k-1},j_{k-1}) συν το επιπλέον κόστος της μετάβασης από τον (i_{k-1},j_{k-1}) στον (i_k,j_k). Αυτή η διαδικασία επαναλαμβάνεται για κάθε κόμβο στο πλέγμα. Ωστόσο, σε πολλές περιπτώσεις δεν συμμετέχουν όλοι οι κόμβοι του πλέγματος και η αναζήτηση της εύρεσης του βέλτιστου μονοπατιού λαμβάνει μέρος σε ένα υποσύνολο των κόμβων , το οποίο καθορίζεται από τους καθολικούς περιορισμούς. Ο προκύπτων αλγόριθμος είναι γνωστός ως δυναμικός προγραμματισμός. Η παραπάνω εξίσωση μπορεί να τροποποιηθεί για την περίπτωση που επιθυμούμε μεγιστοποίηση του κόστους.

Η Εικόνα 14 δείχνει αυτή τη διαδικασία. Το σύνολο των κόμβων που συμμετέχουν στην βελτιστοποίηση (καθολικοί περιορισμοί) φαίνονται ως μαύρες κουκίδες, ενώ οι τοπικοί περιορισμοί που καθορίζουν τις επιτρεπόμενες μεταβάσεις από κόμβο σε κόμβο παρουσιάζονται με μαύρες γραμμές.



Εικόνα 14: Βέλτιστο μονοπάτι ανάμεσα σε δύο ακολουθίες. Οι μαύρες γραμμές υποδηλώνουν τις επιτρεπόμενες μεταβάσεις ανάμεσα στους κόμβους

Έχοντας αποφασίσει να ελέγξουμε ολόκληρο το μονοπάτι και υποθέτοντας ότι το D(0,0)=0, χρησιμοποιώντας την εξίσωση [1] υπολογίζονται όλα τα πιθανά κόστη $D(i_1,j_1)$ για κ=1 για όλες τις πιθανές μεταβάσεις (σε αυτή την περίπτωση υπάρχουν δύο μόνο πιθανές μεταβάσεις (1,1) και (1,2)). Στην συνέχεια, υπολογίζονται τα κόστη των τριών κόμβων για κ=2 και η διαδικασία επαναλαμβάνεται ώσπου να φτάσουμε στον τελικό κόμβο (I,J). Η ακολουθία των μεταβάσεων που οδηγούν στο ελάχιστο D(I,J) του τελικού κόμβου καθορίζει το μονοπάτι με το ελάχιστο κόστος, το οποίο παρουσιάζεται με μπλε γραμμή.

Συνοψίζοντας, αφού έχουμε ολοκληρώσει την φάση της προπαρασκευής και της επιλογής χαρακτηριστικών, το πρότυπο αναφοράς και το υπό εξέταση πρότυπο παρουσιάζονται ως ακολουθίες διανυσμάτων χαρακτηριστικών r(i) και t(j). Ο σκοπός μας είναι να υπολογίσουμε την καλύτερη αντιστοιχία ανάμεσα στα δύο πρότυπα. Για να το πετύχουμε αυτό, το πρότυπο υπό εξέταση θα πρέπει να τεντωθεί στον χρόνο (ένα δείγμα του προτύπου υπό εξέταση αντιστοιχεί σε περισσότερα από ένα δείγματα του προτύπου αναφοράς) ή να συμπιεστεί στον χρόνο (περισσότερα από ένα δείγματα του προτύπου υπό εξέταση αντιστοιχούν στο σε ένα δείγμα του προτύπου αναφοράς).

Αυτή η ευθυγράμμιση των διανυσμάτων των δύο προτύπων επιτυγχάνεται με την διαδικασία του δυναμικού προγραμματισμού. Αρχικά, τοποθετείται το πρότυπο υπό εξέταση στον κάθετο άξονα ενώ το πρότυπο αναφοράς στον οριζόντιο. Στην συνέχεια πρέπει να καθοριστούν τα εξής:

- Καθολικοί περιορισμοί
- Τοπικοί περιορισμοί
- Περιορισμοί τελικών σημείων
- Το κόστος d των μεταβάσεων

i. Περιορισμοί τελικών σημείων

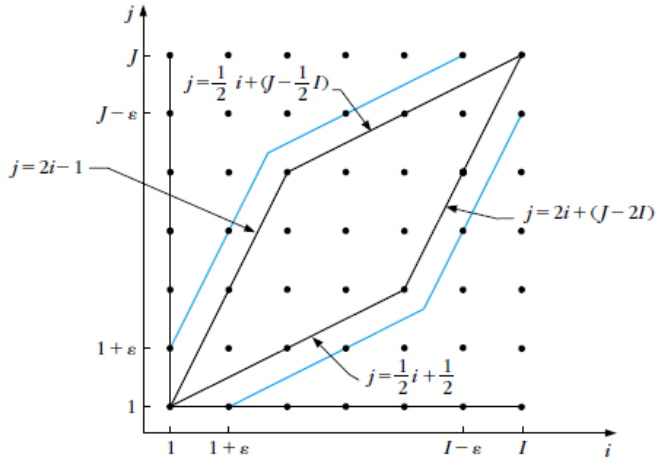
Μέχρι στιγμής, αναζητούμε το βέλτιστο μονοπάτι το οποίο πάντα ξεκινά στον κόμβο (0,0) και τελειώνει στον κόμβο (I,J) και του οποίου η πρώτη μετάβαση είναι στον κόμβο (1,1). Μία απλή παραλλαγή του ολόκληρου μονοπατιού συμβαίνει αν δεν ξέρουμε εκ των προτέρων τα άκρα των ακολουθιών (τον αρχικό και τελικό κόμβο) αλλά γνωρίζουμε πως βρίσκονται σε απόσταση ε από τα σημεία (1,1) και (I,J), ενώ η διαδικασία εντοπισμού τους αφήνεται στον αλγόριθμο.

ii. Καθολικοί περιορισμοί

Οι καθολικοί περιορισμοί καθορίζουν την περιοχή των κόμβων που πρέπει να ελεγχθούν για το βέλτιστο μονοπάτι. Οι κόμβοι εκτός αυτής της περιοχής δεν ελέγχονται. Βασικά, οι καθολικοί περιορισμοί καθορίζουν την συνολική συμπίεση ή τέντωμα που επιτρέπεται από την διαδικασία.

Ένα τέτοιο παράδειγμα παρουσιάζεται στην Εικόνα 15. Οι περιορισμοί αυτοί είναι γνωστοί ως περιορισμοί Itakura, και επιβάλλουν έναν μέγιστο παράγοντα που ισούται με 2 για οποιαδήποτε επέκταση ή συμπίεση του υπό εξέταση προτύπου ώστε να μοιάζει περισσότερο με το πρότυπο αναφοράς. Οι επιτρεπόμενοι κόμβοι βρίσκονται μέσα στο παραλληλόγραμμο που καθορίζεται από τις μαύρες γραμμές. Οι γαλάζιες γραμμές ανταποκρίνονται στις ίδιες καθολικές μεταβλητές αλλά όταν υιοθετούνται οι περιορισμοί τελικών σημείων που αναφέρθηκαν προηγουμένως.

Μπορούμε να παρατηρήσουμε πως τα μονοπάτια στις απέναντι πλευρές του παραλληλόγραμμου μπορούν να μεγαλώσουν ή να μικρύνουν κατά έναν συντελεστή που ισούται με 2, ο οποίος είναι ο μέγιστος συντελεστής που μπορεί να επιτευχθεί. Αυτός ο συντελεστής αυξάνει ή ελαττώνει τον αριθμό των κόμβων που πρέπει να ελεγχθούν για να βρεθεί το βέλτιστο μονοπάτι.



Εικόνα 15: Καθολικοί περιορισμοί

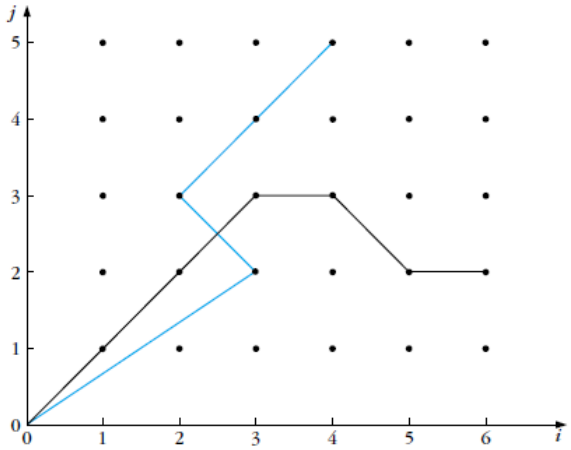
iii. Τοπικοί περιορισμοί

Οι τοπικοί περιορισμοί καθορίζουν το σύνολο των προγόνων και των επιτρεπόμενων μεταβάσεων σε έναν κόμβο του πλέγματος. Συγκεκριμένα, υπαγορεύουν τα όρια για τον μέγιστο παράγοντα συμπίεσης / επέκτασης. Μία ιδιότητα που πρέπει να διαθέτουν όλοι οι τοπικοί περιορισμοί είναι η μονοτονικότητα. Δηλαδή

$i_{k-1} \leq ik$ και $j_{k-1} \leq jk$

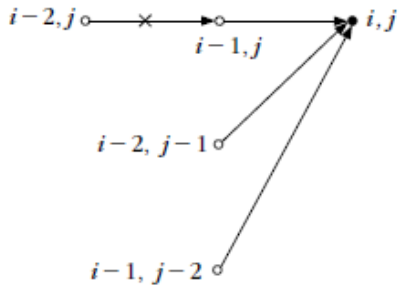
το οποίο υποδηλώνει πως όλοι οι πρόγονοι ενός κόμβου βρίσκονται στα αριστερά και νότια από αυτόν. Η ιδιότητα αυτή εγγυάται ότι η διαδικασία ακολουθεί την φυσική εξέλιξη του χρόνου και δεν μπερδεύεται π.χ. στην αναγνώριση λέξεων να μπερδέψει την λέξη αργία, με την λέξη άγρια.

Δύο παραδείγματα μη μονοτονικών μονοπατιών φαίνονται στην Εικόνα 16.



Εικόνα 16: Παραδείγματα μη μονοτονικών μονοπατιών

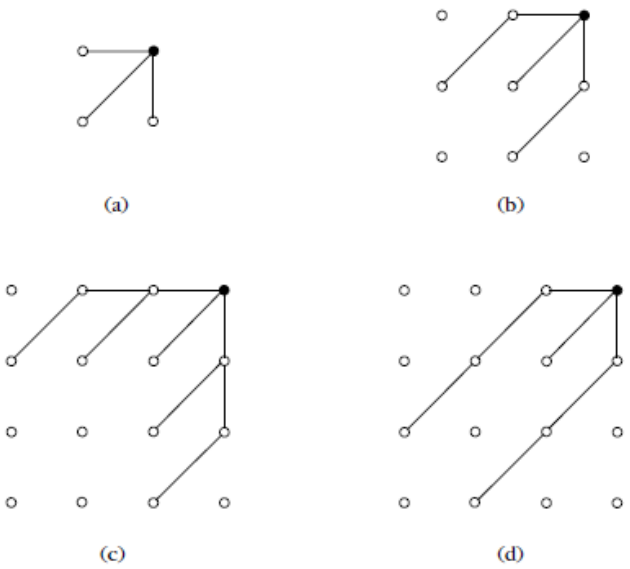
Ένα δημοφιλές σύνολο τοπικών περιορισμών το οποίο είναι γνωστό ως περιορισμοί Itakura, παρουσιάζεται στην Εικόνα 17. Η μέγιστη δυνατή συμπίεση (επέκταση) η οποία μπορεί να επιτευχθεί είναι 2 και είναι αποτέλεσμα επαναλαμβανόμενων μεταβάσεων του τύπου (i-1,j-2) στον κόμβο (i,j). Ένα ακόμη αξιοσημείωτο χαρακτηριστικό των περιορισμών Itakura είναι το ότι ενώ επιτρέπονται οι οριζόντιες μεταβάσεις, δεν μπορούν να είναι συνεχόμενες, γεγονός το οποίο υποδηλώνει το χ πάνω από το βέλος.



Εικόνα 17: Περιορισμοί Itakura, όπου δύο συνεχόμενες οριζόντιες μεταβάσεις δεν επιτρέπονται

Επιπλέον, οι περιορισμοί αυτοί επιτρέπουν στο μονοπάτι να παραλείψει τουλάχιστον ένα δείγμα στο υπό εξέταση πρότυπο (την θέση j-1 στον κατακόρυφο άξονα) και το μονοπάτι πηγαίνει από το (i-1, j-2) στον κόμβο (i,j). Αντιθέτως, τα δείγματα στο πρότυπο αναφοράς δεν μπορούν να παραλειφθούν και όλα λαμβάνουν μέρος στο βέλτιστο μονοπάτι. Τέτοιοι περιορισμοί είναι γνωστοί ως ασύμμετροι περιορισμοί.

Ένας αριθμός εναλλακτικών τοπικών περιορισμών έχουν προταθεί και χρησιμοποιηθεί στην πράξη. Η Εικόνα 18 παρουσιάζει τέσσερεις διαφορετικούς τύπους περιορισμών που προτάθηκαν από τους Sakoe και Chiba. Για το 18.α δεν υπάρχει όριο στον συντελεστή επέκτασης/συμπίεσης , εφόσον συνεχόμενες οριζόντιες ή κάθετες μεταβάσεις μπορούν να συμβούν, έως ότου βγουν έξω από την περιοχή που ορίζεται από τους καθολικούς περιορισμούς. Αντίθετα, στο 18.β οι οριζόντιες και κάθετες μεταβάσεις επιτρέπονται έπειτα από μία διαγώνια μετάβαση και στο 18.δ έπειτα από δύο διαδοχικές διαγώνιες μεταβάσεις. Τέλος, στο 18.γ το πολύ δύο διαδοχικές κάθετες ή οριζόντιες μεταβάσεις επιτρέπονται μόνο μετά από μία διαγώνια μετάβαση. Οι παράγοντες συμπίεσης/ επέκτασης που καθορίζονται για κάθε έναν από τους περιορισμούς α, β, γ και δ είναι ∞, 2, 3 και 3/2 αντίστοιχα.



Εικόνα 18: Τοπικοί περιορισμοί Shakoe Shiba. Στο β επιτρέπεται οριζόντια ή κάθετη μετάβαση μόνο μετά από μια διαγώνια, στο γ επιτρέπονται δύο συνεχόμενες οριζόντιες ή κάθετες μεταβάσεις μόνο μετά από μια διαγώνια ενώ στο δ επιτρέπεται μια οριζόντια μετάβαση έπειτα από δύο συνεχόμενες διαγώνιες μεταβάσεις

iv. Το κόστος d των μεταβάσεων

Ένα κόστος που χρησιμοποιείται συχνά, είναι η Ευκλείδεια απόσταση ανάμεσα στα $r(j_k)$ και $t(j_k)$, τα οποία αντιστοιχούν στον κόμβο (i_k,j_k) και ισούται με

$$d(i_k, j_k | i_{k-1}, j_{k-1})] = ||r(i_k) - t(j_k)|| \equiv d(i_k j_k)$$

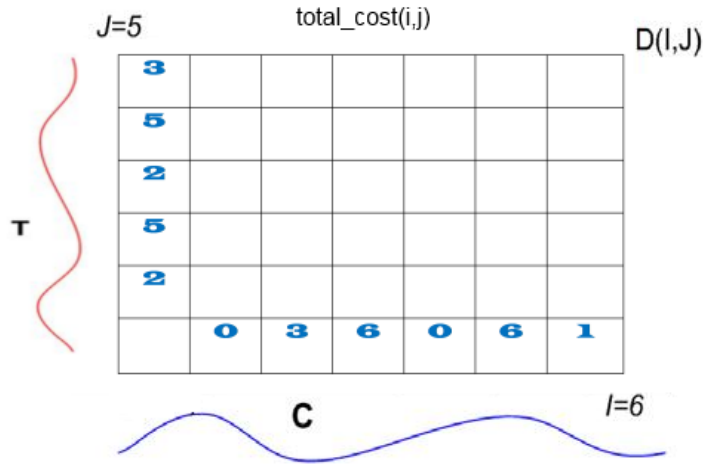
με αυτόν τον τρόπο υποθέτουμε ότι κανένα κόστος δεν συσχετίζεται με την μετάβαση σε έναν συγκεκριμένο κόμβο , ενώ το κόστος εξαρτάται ολοκληρωτικά από τα διανύσματα χαρακτηριστικών που αντιστοιχούν στον συγκεκριμένο κόμβο.

Ακόμα, πρέπει να αναφερθεί ότι πρέπει να πραγματοποιηθεί μια κανονικοποίηση του συνολικού κόστους D, για να αντισταθμίσουμε την διαφορά στα μήκη των μονοπατιών προσφέροντας ίσες ευκαιρίες σε όλα τους. Μία λογική επιλογή αποτελεί το να διαιρεθεί το συνολικό κόστος D με το μήκος κάθε μονοπατιού.

3.2.3 Επεξήγηση του αλγορίθμου DTW

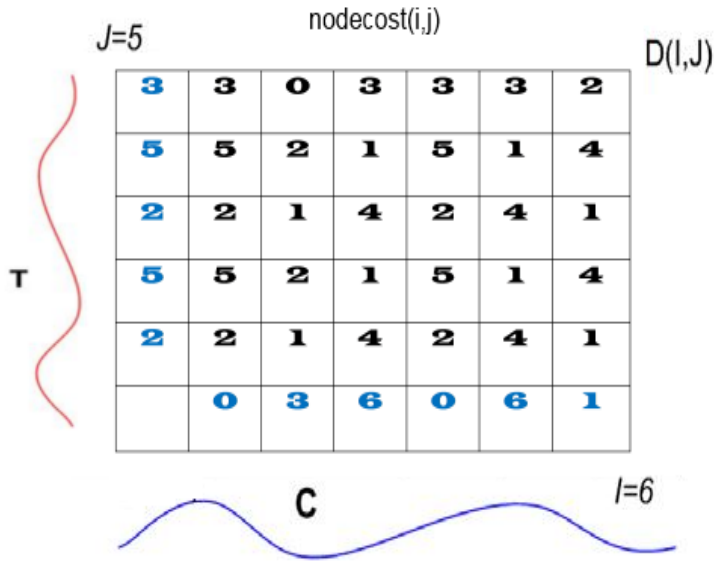
Στην ενότητα αυτή θα επεξηγήσουμε την λειτουργία του αλγορίθμου DTW χωρίς περιορισμούς τελικών σημείων, ο οποίος δέχεται ως είσοδο δύο ακολουθίες χαρακτηριστικών r(i) μήκους I και t(i) μήκους J, όπου συνήθως I≠J, οι οποίες αντιστοιχούν στο πρότυπο αναφοράς και στο πρότυπο υπό εξέταση αντίστοιχα. Τοποθετώντας το πρότυπο υπό εξέταση στον κάθετο άξονα και το πρότυπο αναφοράς στον οριζόντιο, ο στόχος του αλγορίθμου είναι να υπολογίσει και να συμπληρώσει τον πίνακα ολικού κόστους, ο οποίος παρουσιάζεται στην Εικόνα 19.

Για λόγους απλούστευσης, η κάθε ακολουθία χαρακτηριστικών έχει ένα μόνο χαρακτηριστικό, και συνεπώς ο υπολογισμός του κόστους γίνεται με μια απλή αφαίρεση (δες την ενότητα 4.3).



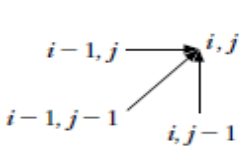
Εικόνα 19: Ο πίνακας ολικού κόστους τον οποίο πρέπει να συμπληρώσει ο αλγόριθμος DTW

Αρχικά, ο αλγόριθμος υπολογίζει το κόστος κάθε κόμβου, δηλαδή το πόσο διαφέρει το κάθε στοιχείο του ενός προτύπου από το άλλο. Π.χ. στην Εικόνα 20, για να υπολογίσουμε το κόστος του στοιχείου κάτω αριστερά (κόμβος (1,1)), αφαιρούμε την τιμή του προτύπου η οποία βρίσκεται στην ίδια γραμμή με αυτό από την τιμή του προτύπου η οποία βρίσκεται στην ίδια στήλη με αυτό και στην συνέχεια παίρνουμε την απόλυτη τιμή του αποτελέσματος, δηλαδή |2-0|=2. Η διαδικασία αυτή συνεχίζεται για κάθε κόμβο έως ότου συμπληρωθούν όλες οι τιμές του πίνακα.



Εικόνα 20: Ο πίνακας τοπικού κόστους

Για την συμπλήρωση του ολικού κόστους, αρχικά πρέπει να δούμε τις επιτρεπόμενες μεταβάσεις από κόμβο σε κόμβο. Στο παράδειγμα αυτό χρησιμοποιούμε τους περιορισμούς Sakoe Shiba, οι οποίοι παρουσιάζονται στην Εικόνα 21, σύμφωνα με τους οποίους δεν υπάρχει όριο στις οριζόντιες ή κάθετες μεταβάσεις, δηλαδή το υπό εξέταση πρότυπο μπορεί να επεκταθεί ή να συμπιεστεί όσο θέλει, αρκεί βέβαια να μη βγαίνει έξω από τα όρια που ορίζουν οι καθολικοί περιορισμοί.



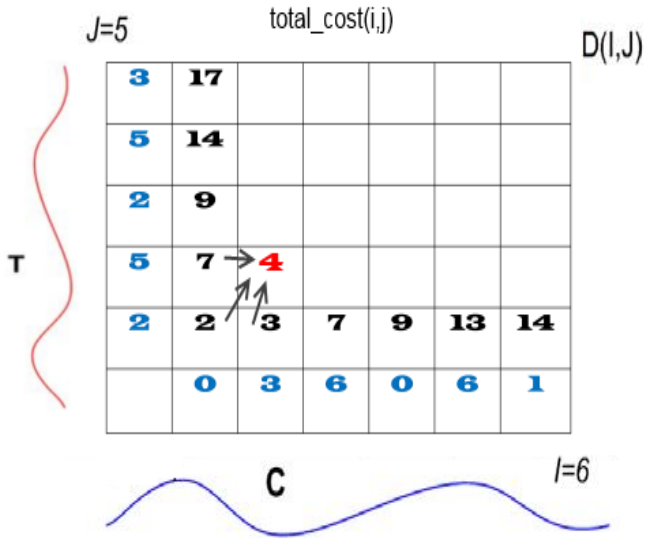
Εικόνα 21: Περιορισμοί SakoeShiba

Υποθέτουμε ότι το D(0,0)=0. Καθώς μπορούμε να φτάσουμε σε κάθε κόμβο n(i,j) από τρεις διαφορετικούς προγόνους n(i-1,j),(i-1,j-1),(i,j-1) αρχικά υπολογίζουμε το ολικό κόστος για την πρώτη γραμμή και την πρώτη στήλη. Αυτό συμβαίνει γιατί στα στοιχεία της πρώτης στήλης (εκτός από τον κόμβο (0,0) ο οποίος δεν διαθέτει κανένα πρόγονο) μπορούμε να πάμε από ένα μόνο πρόγονο, ο οποίος είναι ο (i,j-1). Το ίδιο συμβαίνει με τα στοιχεία της πρώτης γραμμής στα οποία μπορούμε να πάμε μόνο από τον πρόγονο (i-1,j).

Για τον υπολογισμό του ολικού κόστους, όπως παρουσιάζει η Εικόνα 22 χρησιμοποιείται ο παρακάτω τύπος

$$D_{min}(i_k, j_k) = \min_{i_{k-1}, j_{k-1}} [D_{min}(i_{k-1}, j_{k-1}) + d(i_k, j_k | i_{k-1}, j_{k-1})]$$

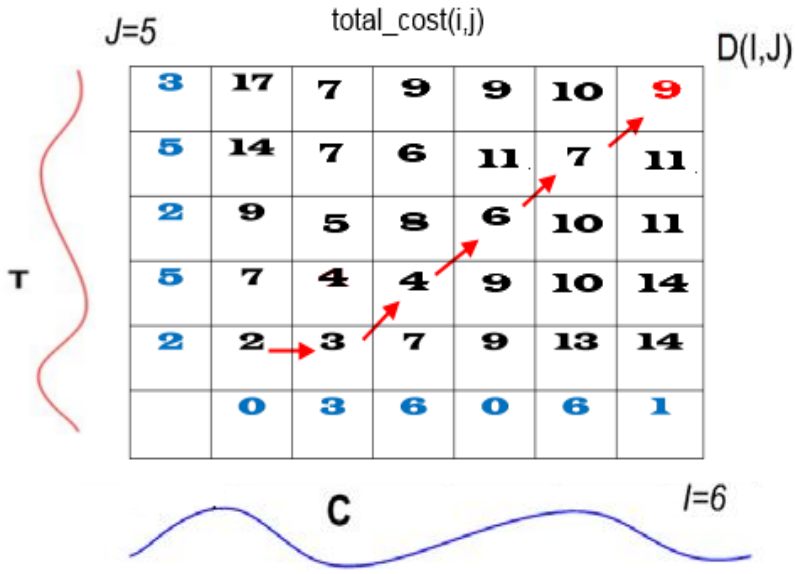
- Για τον κόμβο (1,1) ο οποίος δεν διαθέτει προγόνους, το συνολικό κόστος είναι ίσο με το τοπικό κόστος για αυτόν τον κόμβο, δηλαδή ίσο με 2.
- Για τον κόμβο (1,2) ο οποίος διαθέτει μόνο τον πρόγονο (1,1), το συνολικό κόστος είναι ίσο με το τοπικό κόστος για αυτόν τον κόμβο(δηλαδή ίσο με 5) συν το ολικό κόστος του προγόνου (2). Επομένως το ολικό κόστος για τον κόμβο αυτό ισούται με 7.
- Για τον κόμβο (2,2) ο οποίος διαθέτει τρεις προγόνους (1,1),(1,2) και (2,1), το συνολικό κόστος είναι ίσο με το τοπικό κόστος για αυτόν τον κόμβο (δηλαδή ίσο με 2) συν το ελάχιστο ολικό κόστος από τους τρεις προγόνους (δηλαδή ίσο με 2). Επομένως το ολικό κόστος για τον κόμβο αυτό ισούται με 4.



Εικόνα 22: Συμπλήρωση του ολικού κόστους από τον αλγόριθμο χρησιμοποιώντας το μικρότερο ολικό κόστος από τους τρεις επιτρεπόμενους προγόνους

Η διαδικασία αυτή συνεχίζεται ώσπου να συμπληρωθεί όλος ο πίνακας. Στον κόμβο (I,J), όπως φαίνεται και στην Εικόνα 23, βρίσκεται το ελάχιστο κόστος από όλα τα δυνατά μονοπάτια που κατάφερε να εντοπίσει ο αλγόριθμος. Στην περίπτωση που αποθηκεύαμε και το ποιος

πρόγονος χρησιμοποιήθηκε για να πραγματοποιηθεί η κάθε μετάβαση, θα μπορούσαμε με μια τεχνική που λέγεται backtracking να έχουμε ολόκληρο το βέλτιστο μονοπάτι.



Εικόνα 23: Το βέλτιστο μονοπάτι που μας οδηγεί στον κόμβο (I,J) με ολικό κόστος 9

3.2.4 Κώδικας Matlab για τον αλγόριθμο DTW με περιορισμούς Sakoe Shiba και χρήση Ευκλείδειας απόστασης ως μέτρο κόστους

Στην ενότητα αυτή παρουσιάζεται ο κώδικας Matlab που παρέχεται στο [14] και ο οποίος επιλέχτηκε και τροποποιήθηκε στην παρούσα μεταπτυχιακή διατριβή

```
%Initialization
I=size(ref,2);
J=size(test,2);
NodeCost=zeros(J,I);
D=zeros(J,I);

for j=1:J
fori=1:I
%Euclidean distance
NodeCost(j,i)=sqrt(sum((ref(:,i)-test(:,j)).^2));
end
end

%Initialization
D(1,1)=NodeCost(1,1);
for j=2:J
D(j,1)=D(j-1,1)+NodeCost(j,1);
end
fori=2:I
D(1,i)=D(1,i-1)+NodeCost(1,i);
end

%Main Loop
for j=2:J
```

```
for i=2:I
%Sakoe-Chiba local path constraints
[D(j,i),ind]=min([D(j-1,i-1) D(j-1,i) D(j,i-1)]+NodeCost(j,i));
end%for i=2:I
end%for j=2:J
%End of Main Loop

MatchingCost=D(J,I);
```

Κεφάλαιο 4: Επιλογή ρυθμίσεων του συστήματος αναγνώρισης χειρονομιών

4.1 Επιλογή δειγματοληψίας

Η επιλογή του ρυθμού με τον οποίο συλλέγουμε δεδομένα από το επιταχυνσιόμετρογια την αναγνώριση των χειρονομιών είναι υψηλής σημασίας. Η μονάδα μέτρησης αδράνειας που χρησιμοποιούμε μας δίνει την δυνατότητα να ρυθμίσουμε τον ρυθμό δειγματοληψίας σύμφωνα με τις απαιτήσεις της εφαρμογής. Για την ρύθμιση του ρυθμού δειγματοληψίας προγραμματίζουμε κατάλληλα το πεδίο SMPRT_DIV του καταχωρητή Sample Rate Divider.

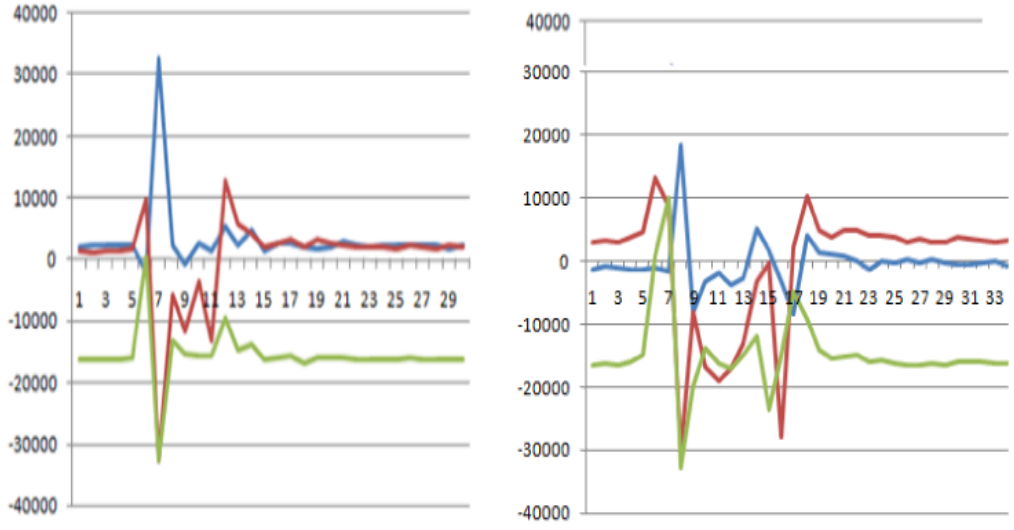
Η τιμή της δειγματοληψίας υπολογίζεται διαιρώντας τον ρυθμό εξόδου του επιταχυνσιόμετρου ο οποίος είναι 1kHz με την τιμή SMPRT_DIV.

Sample Rate = Accelerometer Output Rate / (1 + SMPRT_DIV)

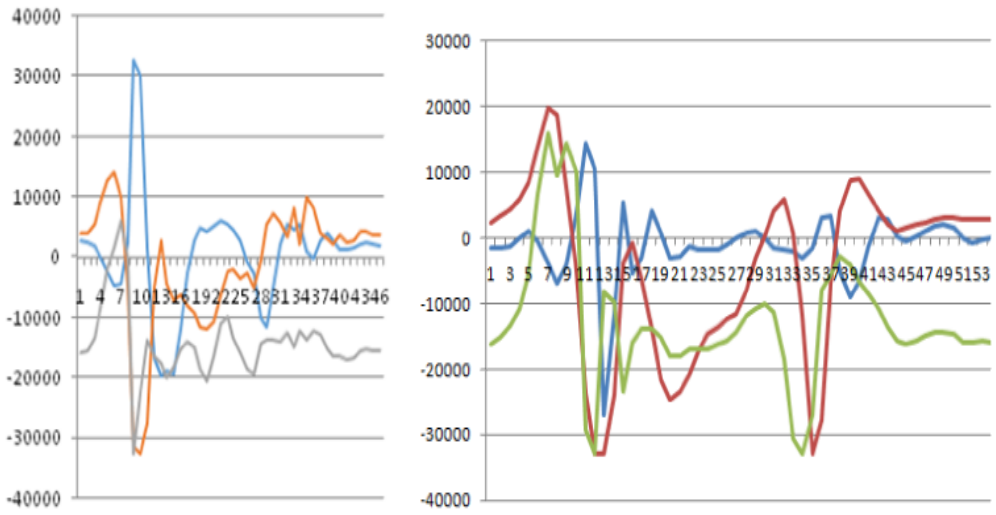
Καθώς το σύστημά μας οδηγείται από interrupts και ο Dynamic Time Warping εκτελείται μέσα στο interrupt service routine, πρέπει να βεβαιωθούμε πως ο χρόνος που μεσολαβεί ανάμεσα σε δύο interrupt είναι αρκετός για να τρέξει το κομμάτι εκείνο του αλγορίθμου που απαιτεί τον περισσότερο χρόνο εκτέλεσης. Το κομμάτι αυτό του αλγορίθμου είναι η επεξεργασία των έξι πρώτων δειγμάτων της χειρονομίας στην κατάσταση START_GST[όπως εξηγείται στην ενότητα6.5].

Ακόμα, ο ρυθμός δειγματοληψίας πρέπει να είναι αρκετά μεγάλος ώστε να συλλέγουμε αρκετή πληροφορία σχετικά με την κίνηση του δαχτύλου, αλλά όχι τόσο μεγάλος ώστε να προλαβαίνει να τρέξει ο αλγόριθμος.

Λαμβάνοντας υπόψη τα παραπάνω, ξεκινώντας από μια δειγματοληψία των 25Hz, παρουσιάζουμε τα δεδομένα που συλλέγουμε για μια κίνηση (κλικ) με τους διάφορους ρυθμούς δειγματοληψίας.



Εικόνα 24: Κλικ με δειγματοληψία 25Hz αριστερά και 40Hz δεξιά



Εικόνα 25: Κλικ με δειγματοληψία 100Hz αριστερά και 125Hz δεξιά, παρατηρούμε πως όσο μεγαλώνει η δειγματοληψία μεγαλώνει και το μέγεθος της χειρονομίας

Θεωρούμε πως μια δειγματοληψία των 100Hz, όπου δεχόμαστε ένα δείγμα ανά 10 msec είναι ικανοποιητική έτσι ώστε να διαθέτουμε και αρκετή πληροφορία για την κίνηση αλλά και αρκετό χρόνο για την επεξεργασία της. Με την δειγματοληψία που επιλέχθηκε, ο μέγιστος χρόνος που παραμένουμε στο interrupt service routine είναι 5,799msec από τα δέκα msec που έχουμε στην διάθεσή μας, το οποίο σημαίνει ότι η μέγιστη χρησιμοποίηση του επεξεργαστή κατά την διάρκεια της αναγνώρισης των χειρονομιών είναι μικρότερη του 58%.

4.2 Διαδικασία εξαγωγής χαρακτηριστικών

Τα δεδομένα εισόδου που χρησιμοποιεί ο αλγόριθμος ώστε να επιστρέψει το κόστος ανάμεσα στα δύο πρότυπα εισόδου είναι δύο ακολουθίες διανυσμάτων χαρακτηριστικών. Η διαδικασία εξαγωγής χαρακτηριστικών (feature extraction) συντελεί στο να δημιουργηθούν νέα χαρακτηριστικά εφαρμόζοντας συναρτήσεις στα υπάρχοντα. Τα δεδομένα τα οποία έχουμε στην διάθεσή μας είναι οι τιμές που μας παρέχει το επιταχυνσιόμετρο στους τρεις άξονες x,y και z.

Στην φάση αυτή έχοντας τις τιμές του επιταχυνσιόμετρου, δοκιμάστηκαν τέσσερις τεχνικές:

- μετατροπή επιτάχυνσης σε g
- διακριτοποίηση της επιτάχυνσης
- χρησιμοποίηση τεχνικής Sliding Window για απαλοιφή θορύβου
- μεταβολή επιτάχυνσης

Για την μετατροπή της επιτάχυνσης σε g, πήραμε την τιμή όπως μας την δίνει ο αισθητήρας και την διαιρέσαμε διά 16384. Στην συνέχεια, πραγματοποιήσαμε διακριτοποίηση της επιτάχυνσης σύμφωνα με το παρακάτω

Accel_g in what range	Accel_ discrete	Accel_g in what range	Accel_ discrete
-0,002...0,002	0		
-0,1...0,002	-1	0,002...0,1	1

-0,2...-0,1	-2	0,1...0,2	2
-0,3...-0,2	-3	0,2...0,3	3
-0,4...-0,3	-4	0,3...0,4	4
-0,5...-0,4	-5	0,4...0,5	5
-0,6...-0,5	-6	0,5...0,6	6
-0,7...-0,6	-7	0,6...0,7	7
-0,8...-0,7	-8	0,7...0,8	8
-0,9...-0,8	-9	0,8...0,9	9
-1...-0,9	-10	0,9...1	10
-1,2...-1	-11	1...1,2	11
-1,4...-1,2	-12	1,2...1,4	12
-1,6...-1,4	-13	1,4...1,6	13
-1,8...-1,6	-14	1,6...1,8	14
-2...-1,8	-15	1,8...2	15

Στην επόμενη τεχνική παίρναμε διαδοχικά δείγματα επιτάχυνσης και τα αφαιρούσαμε έτσι ώστε να μας μένει μόνο η μεταβολή της επιτάχυνσης σε κάθε άξονα, ενώ στην τελευταία τεχνική παίρναμε έξι διαδοχικά δεδομένα κάθε φορά (ανά τρία επικαλυπτόμενα) και βρίσκαμε την μέση τιμή τους.

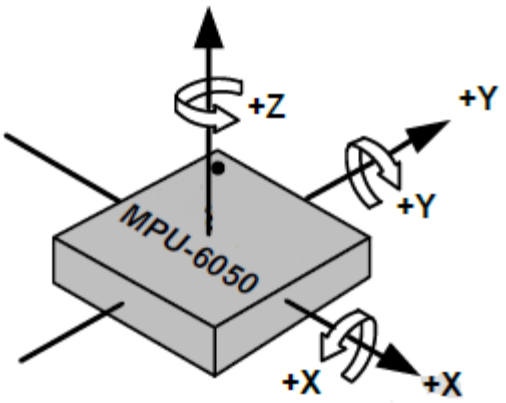
Σε επόμενη φάση, ένα σύνολο χειρονομιών που είχαν περάσει από τις τεχνικές αυτές συγκρίθηκαν με τον αλγόριθμο. Καθώς δεν υπήρξε αξιοσημείωτη βελτίωση στα αποτελέσματα του αλγορίθμου και η χρήση τους θα απαιτούσε παραπάνω μνήμη και πολύτιμο υπολογιστικό χρόνο απορρίφθηκε η χρησιμοποίησή τους.

4.3 Διαδικασία επιλογής χαρακτηριστικών

Η διαδικασία επιλογής χαρακτηριστικών περιλαμβάνει αλγορίθμους για την σωστή επιλογή των χαρακτηριστικών που παρήγαγε η εξαγωγή χαρακτηριστικών. Η επιλογή των χαρακτηριστικών αυτών είναι πολύ σημαντική καθώς μερικά από αυτά μπορούν να περιέχουν πληροφορία η οποία να είναι πλεονάζουσα ή μη χρήσιμη, ενώ η σωστή επιλογή χαρακτηριστικών επιφέρει μικρότερο χρόνο εκπαίδευσης και καλύτερη λειτουργία του μοντέλου καθώς αποφεύγεται η υπέρ-προσαρμογή (overfitting).

Τα χαρακτηριστικά τα οποία διαθέτουμε μετά το πέρας της φάσης εξαγωγής χαρακτηριστικών είναι r(i)=$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$, i=1,2,3,....k, όπου κ το μέγεθος του εκάστοτε προτύπου.

Αν υποθέσουμε ότι ο αισθητήρας βρίσκεται παράλληλα με το έδαφος, ο άξονας x θα μας εμφάνιζε την επιτάχυνση αν ο αισθητήρας πραγματοποιούσε μια κίνηση προς τα δεξιά ή αριστερά, ο άξονας y θα μας εμφάνιζε την επιτάχυνση αν ο αισθητήρας κινούνταν μπροστά ή πίσω και τέλος ο άξονας z θα μας εμφάνιζε την επιτάχυνση αν ο αισθητήρας κινείται πάνω-κάτω, όπως παρουσιάζεται και στην Εικόνα 26.



Εικόνα 26: Οι τρεις άξονες στους οποίους μετράει επιτάχυνση ο αισθητήρας

Στην περίπτωση που αποφασίσουμε να χρησιμοποιήσουμε παραπάνω από έναν άξονα για την αναγνώριση, το κόστος θα υπολογίζεται με την χρήση του τύπου που ακολουθεί

$$d = \sqrt{(r(i)_x - t(i)_x)^2 + (r(i)_y - t(i)_y)^2 + (r(i)_z - t(i)_z)^2}$$

ενώ, αν επιλεγεί ένας μόνο άξονας το κόστος μπορεί να υπολογιστεί με την χρήση μιας μόνο αφαίρεσης

$d = |r(i)_k - t(i)_k|$, όπου κ=x ή y ή z

Και οι τρεις χειρονομίες που θέλουμε να αναγνωρίσουμε δεν απαιτούν κίνηση του δαχτύλου (και συνεπώς του αισθητήρα) δεξιά-αριστερά. Επομένως μπορούμε εύκολα να αποκλείσουμε τον άξονα χ ως χαρακτηριστικό.

Στην περίπτωση που το αισθητήριο είναι τοποθετημένο στο τραπέζι και δεν ασκείται καμία δύναμη πάνω του (εκτός από αυτή της βαρύτητας), οι τιμές που παίρνουμε στους τρείς άξονες είναι

- άξονας x≈0
- άξονας y≈0
- άξονας z≈1g =16384

Όπως βλέπουμε, ο άξονας Z είναι ένας άξονας ο οποίος επηρεάζεται σημαντικά από μια εξωτερική δύναμη, την βαρύτητα. Συνεπώς, θεωρούμε πως ο άξονας y αποτελεί την καλύτερη επιλογή, καθώς επηρεάζεται λιγότερο από την επιτάχυνση της βαρύτητας και μπορεί να περιγράψει την κίνηση του δαχτύλου.

Στην συνέχεια εκτελέσαμε ένα σύνολο από τις τρεις χειρονομίες (20 για κάθε χειρονομία). Χρησιμοποιώντας Matlab, εκτελέσαμε τον αλγόριθμο DTW ανάμεσα τους.

Μέση τιμή κόστους που παράγεται από τον DTW χρησιμοποιώντας τον άξονα y ως χαρακτηριστικό			
	click	press	release
click	104632	269583	275797
press	269583	68280	403941
release	275797	403941	63565

Μέση τιμή κόστους που παράγεται από τον DTW χρησιμοποιώντας τον άξονα z ως χαρακτηριστικό			
	click	press	release
click	54907	127154	113670
press	127154	59762	141527
release	113670	141527	44831

Όπως δείχνουν οι παραπάνω πίνακες, όταν συγκρίνουμε κάθε κλικ που κάναμε με όλα τα υπόλοιπα κλικ, το μέσο κόστος ισούται με 104.632. Συγκρίνοντας όλα τα κλικ με όλα τα press, το μέσο κόστος ισούται με 269.583 κοκ. Παρατηρούμε πως όταν συγκρίνουμε μεταξύ τους διαφορετικές κινήσεις, ο άξονας y παρουσιάζει μεγαλύτερη διαφορά στο μέσο κόστος από ότι ο άξονας z (ένα κλικ συγκρινόμενο με τα κλικ έχει μέσο κόστος 104632 ενώ συγκρινόμενο με τα presses έχει 269583, το οποίο σημαίνει ότι συγκρινόμενο με άλλη κίνηση το μέσο παραγόμενο κόστος είναι 61% πιο μεγάλο από ότι όταν μια κίνηση συγκρίνεται με κίνηση ίδιου τύπου (αντίστοιχα είναι 56% όταν χρησιμοποιείται ο z άξονας)).

Τελικά, για επιβεβαίωση των όσων αναφέραμε παραπάνω, όσον αφορά τον άξονα y, επιλέχθηκε για κάθε χειρονομία εκείνη η ακολουθία διανυσμάτων χαρακτηριστικών η οποία όταν συγκρίθηκε με ίδιου τύπου χειρονομίες έβγαλε το μικρότερο μέσο κόστος (η χειρονομία αυτή που είναι περισσότερο όμοια με τις υπόλοιπες χειρονομίες ίδιου τύπου). Στην συνέχεια, συγκρίναμε όλες τις χειρονομίες με τις τρεις με το χαμηλότερο κόστος όπου και επιβεβαιώσαμε το ότι οι χειρονομίες αναγνωρίζονται σωστά.

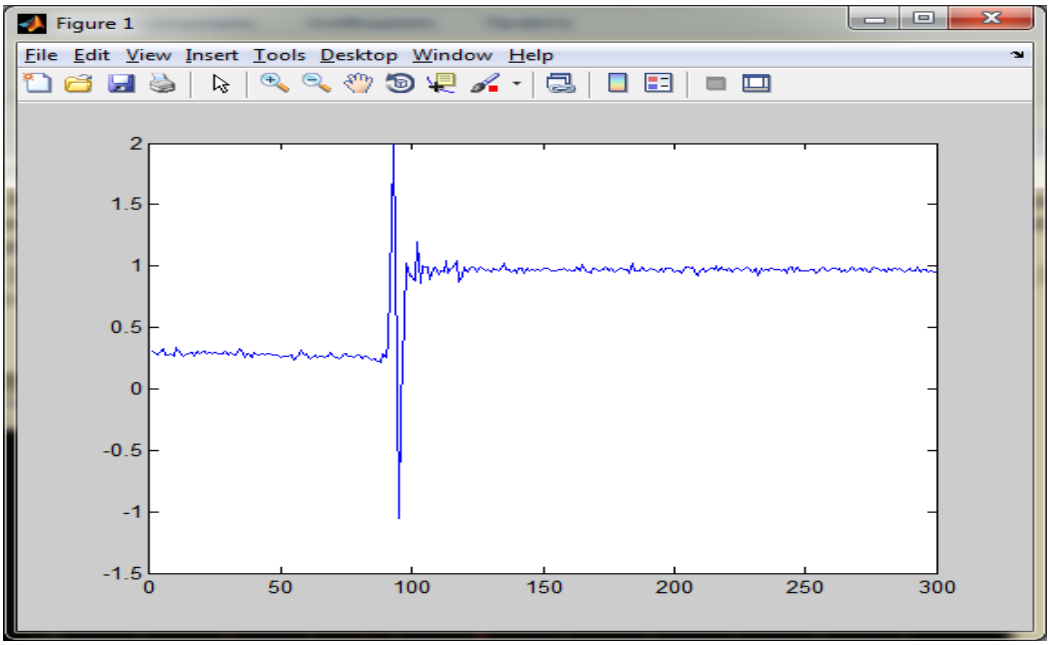
4.4 ΕΝΤΟΠΙΣΜΟΣ ΑΡΧΗΣ ΚΑΙ ΤΕΛΟΥΣ ΚΙΝΗΣΗΣ

Ο αισθητήρας παράγει συνέχεια μετρήσεις επιτάχυνσης, τις οποίες αποθηκεύει στην ουρά που διαθέτει η μονάδα μέτρησης αδράνειας. Για να ξεκινήσουμε την αναγνώριση μιας χειρονομίας, θα πρέπει να γνωρίζουμε πότε αυτή ξεκινάει και πότε τελειώνει. Στην πλειοψηφία της βιβλιογραφίας, η χειρονομία είτε απομονώνεται με το χέρι, είτε το πάτημα/σήκωμα ενός κουμπιού σηματοδοτεί την αρχή και το τέλος της. Ενώ το πάτημα ενός κουμπιού αποτελεί την πιο εύκολη λύση, δεν θεωρούμε πως από θέμα ευχρηστίας είναι η καλύτερη επιλογή. Για αυτό τον λόγο αναζητήσαμε στην βιβλιογραφία λύσεις οι οποίες δέχονται τις συνεχόμενες μετρήσεις ως είσοδο και αποφασίζουν προγραμματιστικά πότε είναι η αρχή ή το τέλος μιας χειρονομίας.

Στο σύστημα αναγνώρισης που προτείνεται στο [5], ο εντοπισμός της δραστηριότητας που σηματοδοτεί την αρχή και το τέλος μιας χειρονομίας γίνεται με την σύγκριση της διακύμανσης ενός συνόλου τιμών με ένα προκαθορισμένο κατώφλι, ενώ στο [7] χρησιμοποιήθηκε η τυπική απόκλιση για τον εντοπισμό του τέλους μιας χειρονομίας.

Καθώς προσπαθούμε να κρατήσουμε στο ελάχιστο τις υπολογιστικές πράξεις, και η τετραγωνική ρίζα της διακύμανσης ισούται με την τυπική απόκλιση, επιλέχθηκε η διακύμανση για την απομόνωση των χειρονομιών. Ακόμη, καθώς ο άξονας z, είναι αυτός που δέχεται τις μεγαλύτερες μεταβολές κατά την διάρκεια μιας χειρονομίας είναι ο άξονας του οποίου η υπολογιζόμενη διακύμανση θα μας βοηθήσει να εντοπίσουμε την αρχή και το τέλος της χειρονομίας.

Στην φάση αυτή αποθηκεύθηκαν διάφορες ακολουθίες χαρακτηριστικών μήκους 300, κάθε μία από τις οποίες περιείχε μια χειρονομία, όπως φαίνεται στην Εικόνα 27.



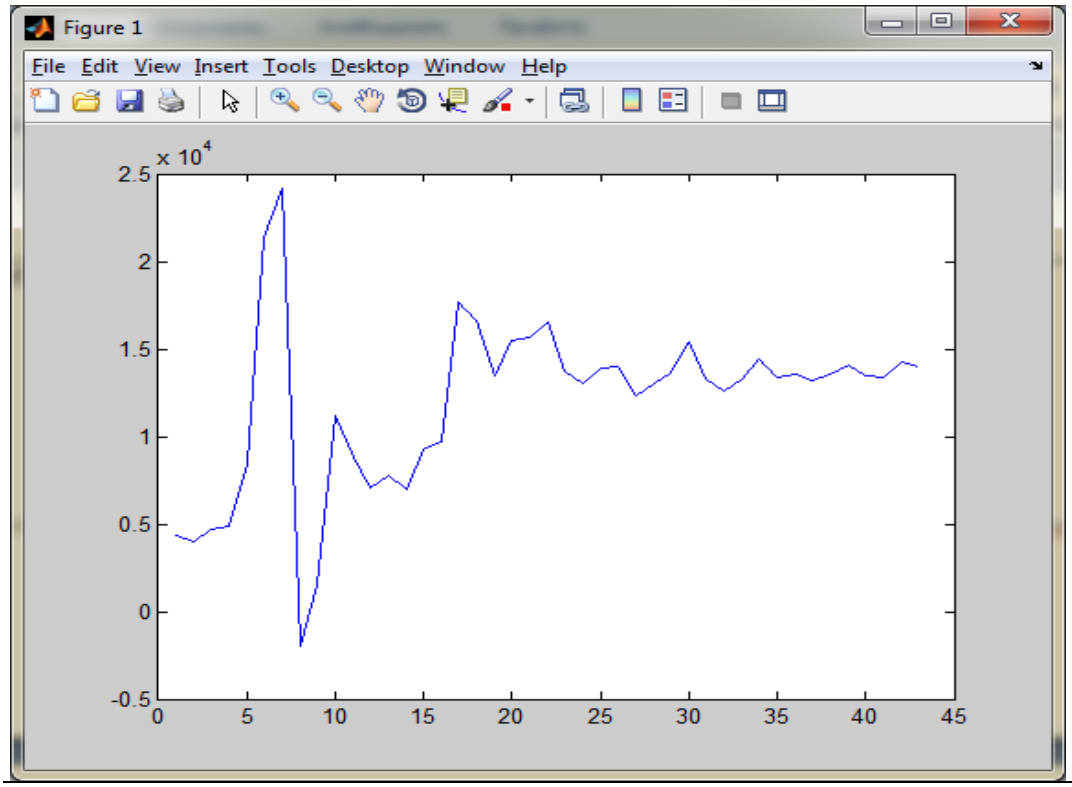
Εικόνα 27: Μια ακολουθία 300 δειγμάτων του άξονα z, η οποία περιλαμβάνει ένα "release"

Στη συνέχεια, χρησιμοποιώντας matlab, φτιάξαμε μια συνάρτηση η οποία βάζει στην ακολουθία σημαίες "έναρξης" και "λήξης" μιας χειρονομίας. Τα ορίσματα εισόδου της είναι η ακολουθία χαρακτηριστικών, το μέγεθος του παραθύρου m στο οποίο θέλουμε να εντοπίζουμε την διακύμανση (δηλαδή ανά πόσες τιμές), το κατώφλι και τέλος η επιθυμητή επικάλυψη n ανάμεσα στα παράθυρα.

Η συνάρτηση αυτή παίρνει με την σειρά m στοιχεία από την ακολουθία (ανά n επικαλυπτόμενα με το προηγούμενο παράθυρο) και υπολογίζει την διακύμανση των στοιχείων αυτών. Αν η διακύμανση είναι μεγαλύτερη από το κατώφλι που έχουμε δώσει ως είσοδο και δεν έχει εισαχθεί πουθενά "σημαία έναρξης", τότε εισάγεται μία καθώς θεωρούμε πως η αυξημένη διακύμανση είναι αποτέλεσμα της έναρξης μιας χειρονομίας.

Στην περίπτωση που έχει εισαχθεί μια σημαία έναρξης, η συνάρτηση εξακολουθεί να υπολογίζει την διακύμανση των επόμενων παραθύρων και βάζει μια "σημαία λήξης" όταν η διακύμανση τεσσάρων συνεχόμενων παραθύρων είναι μικρότερη από το κατώφλι.

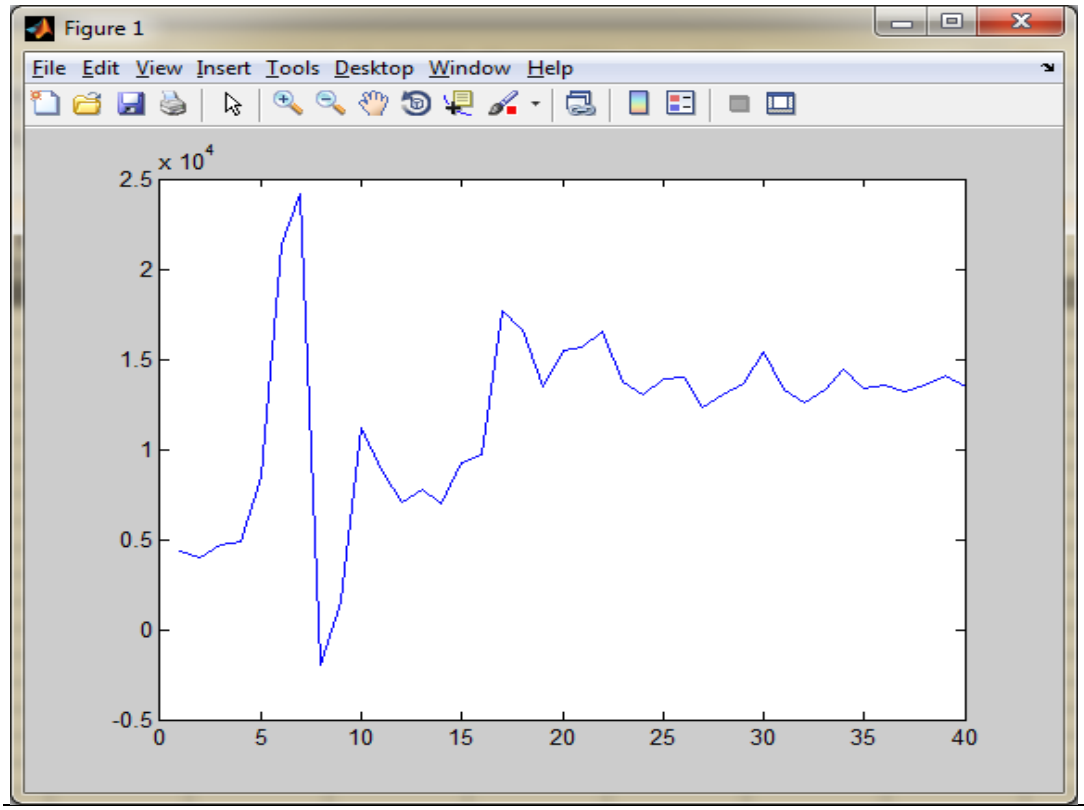
Στις Εικόνες 28 και 29, παρουσιάζεται η ίδια κίνηση, η οποία έχει απομονωθεί χρησιμοποιώντας δύο διαφορετικά κατώφλια ίσα με 300000 και 600000 αντίστοιχα.



Εικόνα 28: Το αποτέλεσμα της συνάρτησης που εντοπίζει μια χειρονομία με κατώφλι 300000

Το επόμενο βήμα, είναι η επιλογή του κατάλληλου κατωφλιού και μεγέθους παραθύρου, η οποία πραγματοποιήθηκε με πολλές δοκιμές. Το μέγεθος του παραθύρου πρέπει να είναι σχετικά μικρό έτσι ώστε να μην παίρνει πάρα πολλές τιμές πριν και μετά την χειρονομία, αλλά να υπάρχουν δύο-τρεις τιμές που να δείχνουν σε τι επίπεδα κυμαινόταν η επιτάχυνση πριν την εκτέλεση της χειρονομίας καθώς βοηθούν την σωστή αναγνώριση της κίνησης. Ακόμα παρατηρούμε πως όσο μεγαλώνει το κατώφλι, τόσο μικραίνει το μέγεθος της κίνησης, καθώς κόβονται τιμές από την αρχή και το τέλος της χειρονομίας. π.χ. στις εικόνες 28 και 29 το μέγεθος της ίδιας χειρονομίας η οποία έχει απομονωθεί χρησιμοποιώντας δυο διαφορετικά κατώφλια είναι 44 και 40 αντίστοιχα.

Έπειτα από πολλές δοκιμές, καταλήξαμε ότι ένα παράθυρο μεγέθους 6 και κατώφλι διακύμανσης =300000 μας δίνει ένα πολύ ικανοποιητικό αποτέλεσμα.



Εικόνα 29: Το αποτέλεσμα της συνάρτησης που εντοπίζει μια χειρονομία με κατώφλι 600000. Παρατηρούμε ότι όσο πιο μεγάλο είναι το κατώφλι, τόσο πιο μικρή η χειρονομία

4.4.1 Χρήση motion interrupt για να σηματοδοτήσει την αρχή της χειρονομίας

Όπως είδαμε στην προηγούμενη ενότητα, ο εντοπισμός της αρχής και του τέλους της κίνησης μπορεί να πραγματοποιηθεί με την χρήση της διακύμανσης. Πιο συγκεκριμένα, ο επεξεργαστής πρέπει να υπολογίζει την διακύμανση έξι τιμών κάθε φορά και αν αυτή ξεπερνάει ένα κατώφλι να σηματοδοτεί την αρχή μιας χειρονομίας. Αυτό όμως σημαίνει πως ο επεξεργαστής θα πρέπει συνέχεια να διαβάζει τιμές από την ουρά και να εκτελεί πράξεις, γεγονός που του δίνει επιπλέον φόρτο εργασίας και ανεβάζει την κατανάλωσή ενέργειας. Για τον λόγο αυτό, χρησιμοποιήσαμε το motion_interrupt [1][11] το οποίο δημιουργεί ένα σήμα interrupt στο πινάκι INT και μεταφέρει φόρτο εργασίας από τον επεξεργαστή του τσιπ, στον επεξεργαστή του αισθητηρίου. Το motioninterrupt είναι ένα προγραμματιζόμενο από τον χρήστη interrupt, το οποίο μπορούμε να ενεργοποιήσουμε γράφοντας "1" στο 7ο bit του καταχωρητή INT_ENABLE, όπως φαίνεται στην Εικόνα30.

4.16 Register 56 – Interrupt Enable INT_ENABLE

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
38	56		MOT_EN		FIFO _OFLOW _EN	I2C_MST _INT_EN	-	-	DATA _RDY_EN

Εικόνα 30: Καταχωρητής 56 (INT_ENABLE): Ο καταχωρητής 56 ρυθμίζει ποια interrupt είναι ενεργοποιημένα κάθε φορά

Στην συνέχεια, κάθε φορά που μπαίνουμε στο interrupt service routine, διαβάζοντας τον καταχωρητή INT_STATUS όπως δείχνει η Εικόνα 31, μπορούμε να δούμε ποιο interrupt κάλεσε την ρουτίνα. Στην περίπτωση που το 7ο bit του καταχωρητή είναι "1", τότε έχουμε ένα motion_interrupt.

4.17 Register 58 – Interrupt Status INT_STATUS

Type: Read Only

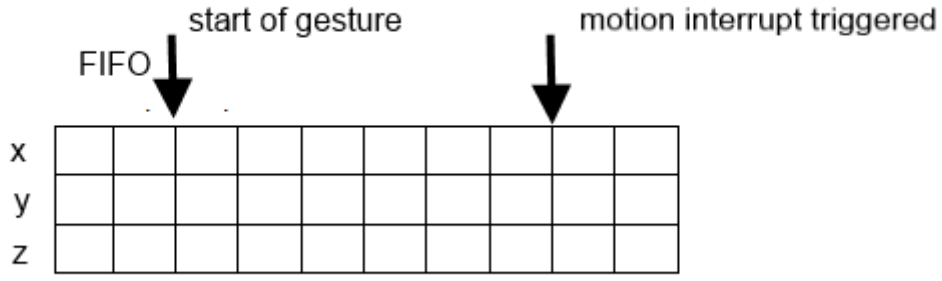
Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3A	58	-	MOT_INT	-	FIFO _OFLOW _INT	I2C_MST _INT	-	-	DATA _RDY_INT

Εικόνα 31:Καταχωρητής 58 (INT_STATUS): Διαβάζοντας τον καταχωρητή 58 μπορούμε να εντοπίσουμε ποιο interrupt έχει καλέσει το ISR

Για να χρησιμοποιήσουμε το motion_interrupt, πρέπει να δώσουμε τιμές στις παρακάτω μεταβλητές:

- MOT_THR_SET, το κατώφλι πάνω από το οποίο πυροδοτείται το motion_interrupt
- MOT_DUR_SET, ο αριθμός των τιμών οι οποίες πρέπει να ξεπερνούν το κατώφλι έτσι ώστε να πυροδοτηθεί το motion_interrupt
- MOT_DECR_RATE, ο ρυθμός με τον οποίο μειώνεται ο μετρητής που μετράει τις τιμές που ξεπερνούν το κατώφλι

Για να βρούμε τις σωστές τιμές στις παραπάνω μεταβλητές, πραγματοποιήθηκαν δοκιμές. Η τιμή του κατωφλιού πρέπει να είναι τέτοια ώστε να πυροδοτείται από την κίνηση του δαχτύλου αλλά επίσης να μην πυροδοτείται πολύ συχνά (ακόμα και όταν ο χρήστης κουνάει το χέρι έτσι ώστε να κουνήσει τον κέρσορα). Ακόμα, οι μεταβλητές MOT_DUR_SET και MOT_DECR_RATE επηρεάζουν το πόσες τιμές που ανήκουν στην χειρονομία θα έχουν ήδη συμβεί προτού πυροδοτηθεί το motion_interrupt, όπως παρουσιάζεται στην Εικόνα 32. Για τον λόγο αυτό χρησιμοποιήθηκε ένα βοηθητικό πρόγραμμα, το οποίο εκτυπώνει τις τιμές που μας δίνει το επιταχυνσιόμετρο από την στιγμή που πυροδοτείται το motion_interrupt έως την λήξη της χειρονομίας χρησιμοποιώντας την διακύμανση. Με διάφορες δοκιμές, αποφασίσαμε πως λαμβάνοντας τις τελευταίες έξι τιμές που υπάρχουν στην ουρά πριν πυροδοτηθεί το motion_interrupt και βάζοντάς τες στην αρχή της χειρονομίας, έχουμε μια ολοκληρωμένη χειρονομία. Για τον λόγο αυτό, κάθε φορά που αδειάζουμε την ουρά για να μην έχουμε υπερχείλιση, αφήνουμε τα τελευταία έξι στοιχεία ώστε αν πυροδοτηθεί motion_interrupt να μην υπάρχει περίπτωση να χαθεί η αρχή της χειρονομίας.



Εικόνα 32: τμήμα της ουράς, όπου φαίνονται η αρχή της χειρονομίας καθώς επίσης και η ενεργοποίηση του motion_interrupt

4.5 Επιλογή περιορισμών τελικών σημείων

Οι περιορισμοί τελικών σημείων είναι αυτοί που μας δίνουν την δυνατότητα να παραλείψουμε, αν το επιθυμούμε, n στοιχεία από την αρχή ή και m στοιχεία από το τέλος του προτύπου με σκοπό να επιτύχουμε μικρότερο κόστος. Η εισαγωγή τους συνεπάγεται περισσότερο κώδικα στην συνάρτηση αναγνώρισης, γεγονός το οποίο καθιστά αποτρεπτικό η περιορισμένη One Time Programmable μνήμη που διαθέτουμε.

Όμως, όπως παρατηρήσαμε με πολλές δοκιμές, στην δική μας περίπτωση η εισαγωγή των περιορισμών αυτών μείωνε την διαφορά κόστους που υπήρχε συγκρίνοντας το άγνωστο πρότυπο με τα πρότυπα αναφοράς, κάνοντας πιο δύσκολη την σωστή αναγνώριση των χειρονομιών. Αυτό συμβαίνει επειδή η περίοδος ηρεμίας πριν και μετά την χειρονομία διαφέρει για τις τρεις κινήσεις, οπότε η σύγκρισή τους σε διαφορετικού τύπου χειρονομίες έχει ως αποτέλεσμα μεγαλύτερο κόστος. Εφόσον η εισαγωγή των τελικών περιορισμών δεν προσφέρει κανένα όφελος στο σύστημα ενώ αντίθετα θα αύξανε το μέγεθος του κώδικα, δεν υιοθετήθηκε η χρησιμοποίησή της.

4.6 Επιλογή τοπικών περιορισμών

Οι τοπικοί περιορισμοί είναι αυτοί που υπαγορεύουν ποιες μεταβάσεις από κόμβο σε κόμβο είναι αποδεκτές ή όχι. Όπως αναφέραμε στην ενότητα iii του κεφαλαίου 3.2.2 υπάρχουν διάφορα είδη τοπικών περιορισμών, οι πιο γνωστοί εκ των οποίων είναι οι περιορισμοί Itakura και Sakoe Shiba.

Οι περιορισμοί Itakura προσφέρουν μέγιστο παράγοντα επέκτασης / συμπίεσης ίσο με δύο, ενώ οι περιορισμοί Sakoe Shiba επιτρέπουν απεριόριστη επέκταση / συμπίεση του υπό εξέταση προτύπου. Καθώς το ελάχιστο μέγεθος μιας χειρονομίας είναι 26 δείγματα και το μέγιστο μέγεθος είναι 60 δείγματα, συμπεραίνουμε ότι δεν μας επηρεάζει το γεγονός ότι οι δεύτεροι περιορισμοί έχουν ∞ παράγοντα συμπίεσης/ επέκτασης. Συνεπώς η επιλογή έγινε με διαφορετικό κριτήριο.

Πραγματοποιώντας διάφορα πειράματα, παρατηρήσαμε ότι και οι δύο περιορισμοί έχουν σχεδόν την ίδια απόδοση, ενώ οι περιορισμοί Sakoe Shiba απαιτούν λιγότερο κώδικα για την υλοποίησή τους καθώς επίσης έχουν πολύ λιγότερες συνθήκες διακλαδώσεων. Λαμβάνοντας υπόψη τα παραπάνω, επιλέχθηκαν οι περιορισμοί Sakoe Shiba καθώς η χρησιμοποίησή τους αντί των περιορισμών Itakura έκανε τον αλγόριθμο πιο γρήγορο και μικρότερο σε μέγεθος.

Κεφάλαιο 5: Τροποποιήσεις για την μείωση της χρησιμοποιούμενης μνήμης και των υπολογιστικών πράξεων

5.1 Περιορισμοί που υπήρχαν

Η υλοποίηση μιας συσκευής εισόδου σε έναν μικροεπεξεργαστή με περιορισμένα resources η οποία θα μπορεί να αναγνωρίσει χειρονομίες που εκτελεί ο χρήστης είναι γεμάτη προκλήσεις εξαιτίας των διάφορων περιορισμών που έχει το σύστημα, οι οποίοι είναι οι εξής:

- Μέγιστο μέγεθος προγράμματος 32Kbyte.
- Περιορισμένη μνήμη SRAM
- Μέγιστο χρόνο εκτέλεσης του υπολογισμού κόστους του αλγορίθμου για έξι δείγματα της χειρονομίας μικρότερο από 10 msec.
- Ταχύτητα απόκρισης του συστήματος

Πιο συγκεκριμένα, ο κώδικας ο οποίος περιλαμβάνει την κίνηση του ποντικιού, την αναγνώριση των κινήσεων, την δημιουργία, διαχείριση και αποστολή των αναφορών με χρήση του πρωτοκόλλου HID του Bluetooth Low Energy , τον scheduler του τσιπ όπως επίσης και την διαχείριση του Bluetooth stack, πρέπει να χωράει στην 32k One Time Programmable (OTP) μνήμη του τσιπ. Αυτό συνεπάγεται πως ο αλγόριθμος θα πρέπει να είναι όσο το δυνατό μικρός σε μέγεθος.

Ακόμα, η περιορισμένη μνήμη SRAM, περιορίζει το μέγεθος και το πλήθος των χρησιμοποιούμενων μεταβλητών, το οποίο πρέπει να κρατηθεί στο ελάχιστο. Η έλλειψη FPU από το τσιπ επιβάλλει την αποφυγή χρησιμοποίησης μεταβλητών τύπου float ή double, και συνεπώς πράξεων που έχουν ως αποτέλεσμα την δημιουργία μεταβλητών τέτοιου είδους.

Επιπροσθέτως, όπως είναι αναμενόμενο, μία συσκευή εισόδου θα πρέπει να έχει άμεση απόκριση το οποίο σημαίνει πως ο ενδιάμεσος χρόνος μεταξύ του τέλους μιας χειρονομίας και της επιτυχούς αναγνώρισής της πρέπει να διατηρηθεί όσο το δυνατόν πιο μικρός.

Τέλος, το γεγονός ότι ο αλγόριθμος αναγνώρισης προτύπων τρέχει εξολοκλήρου μέσα στο interrupt service routine συνεπάγεται ότι θα πρέπει να είναι αρκετά γρήγορος και ότι δεν θα περιλαμβάνει περίπλοκες υπολογιστικές πράξεις. Πιο συγκεκριμένα, το πιο αργό μέρος του αλγορίθμου, αυτό το οποίο υπολογίζει το κόστος για τα έξι αρχικά δείγματα της χειρονομίας, θα πρέπει να μπορεί να τρέξει σε λιγότερο από 10 msec, που είναι η δειγματοληψία μας.

5.2 Τεχνικές που μείωσαν την χρήση μνήμης και το μέγεθος του κώδικα

Στην ενότητα αυτή θα αναφερθούμε στις διάφορες τεχνικές που χρησιμοποιήθηκαν με σκοπό να μειωθεί η μνήμη η οποία χρησιμοποιείται από την συσκευή εισόδου καθώς επίσης και το μέγεθος του κώδικα της εφαρμογής. Οι τεχνικές αυτές είναι οι εξής:

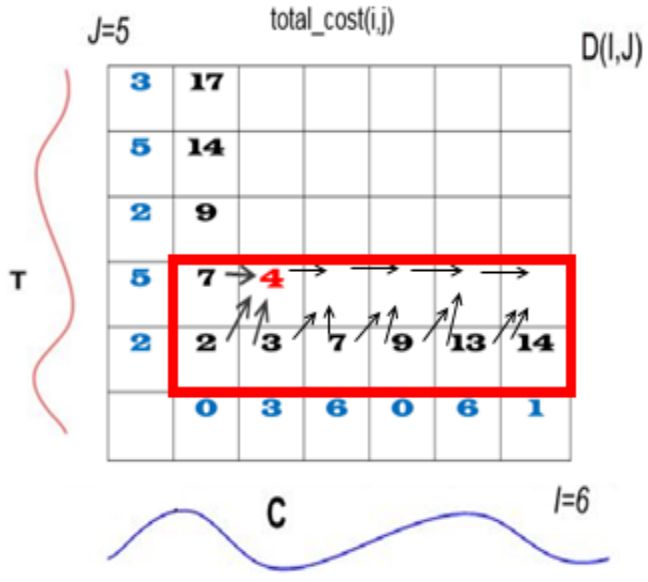
- μη χρήση περιορισμών τελικών σημείων
- επιλογή τοπικών περιορισμών Sakoe Shiba
- χρήση motion interrupt
- επιλογή ενός μόνο άξονα (άξονα y) ως χαρακτηριστικού
- χρήση ενός άξονα για υπολογισμό της διακύμανσης
- διατήρηση ενός μόνο μέρους του πίνακα ολικού κόστους των κόμβων

Όπως αναφέραμε και προηγουμένως, ο κώδικας πρέπει να χωράει στα 32k της one time programmable μνήμης του SmartBond DA14580 τσιπ. Πολλές αποφάσεις που πάρθηκαν όπως η επιλογή να μη χρησιμοποιηθούν περιορισμοί τελικών σημείων [βλέπε ενότητα 4.5] καθώς επίσης και η επιλογή των τοπικών περιορισμών Sakoe Shiba [βλέπε ενότητα 4.6] ελήφθησαν έτσι ώστε να διατηρήσουν το μέγεθος του κώδικα όσο πιο μικρό γίνεται. Ακόμα, η χρήση motion_interrupt (βλέπε ενότητα 4.4.1) μειώνει το μέγεθος του κώδικα εφόσον δεν απαιτείται

έξτρα κώδικας ο οποίος θα εκτελείται συνέχεια προσπαθώντας να εντοπίσει την αρχή μιας κίνησης.

Στην ενότητα 4.3 όπου πραγματοποιήθηκε η επιλογή των χαρακτηριστικών που αποτελούν το διάνυσμα εισόδου του αλγορίθμου, επιλέχθηκε ένας μόνο άξονας από τους τρεις. Αυτό συνεπάγεται πως για την αποθήκευση των προτύπων αναφοράς απαιτείται το ένα τρίτο της μνήμης από ότι αν χρησιμοποιούσαμε και τους τρείς άξονες. Ακόμα, το γεγονός ότι χρησιμοποιείται ένας άξονας, αφαιρεί την χρησιμοποίηση πολύπλοκων υπολογιστικών πράξεων όπως τετραγωνική ρίζα και δυνάμεις του δύο [βλέπε τον υπολογισμό του κόστους ενότητα 4.3], οι οποίες θα πραγματοποιούνταν με την χρήση λογισμικού εφόσον το τσιπ δεν διαθέτει Floating Point Unit. Επίσης, η χρήση ενός άξονα για να σηματοδοτήσει την αρχή και το τέλος μιας χειρονομίας στον υπολογισμό της διακύμανσης [βλέπε ενότητα 4.4] απαιτεί την χρήση λιγότερων μεταβλητών.

Ακόμα ο αλγόριθμος DTW χρησιμοποιεί δύο πίνακες για να αποθηκεύει τα τοπικά καθώς επίσης και τα τελικά κόστη. Εφόσον το ελάχιστο μήκος μιας χειρονομίας είναι 26 και το μέγιστο επιτρεπόμενο μήκος είναι 60, τότε ο κάθε πίνακας τοπικών κοστών των κόμβων έχει μέγιστο μέγεθος 60x60x4 bytes, και επειδή έχουμε έναν πίνακα για κάθε είδος χειρονομίας που θέλουμε να αναγνωρίσουμε έχουμε43.200 bytes. Αντίστοιχα το ελάχιστο μέγεθος του πίνακα τοπικών κοστών(αν υποθέσουμε πως τα πρότυπα αναφοράς και υπό εξέταση έχουν μήκος 26) είναι 8112bytes. Το ίδιο ισχύει και για τους πίνακες ολικών κοστών. Αυτό σημαίνει πως μόνο αυτές οι δύο μεταβλητές θα απασχολούσαν ολόκληρη την sysRam που διαθέτουμε (42kB) χωρίς να αφήνουν χώρο για τις υπόλοιπες μεταβλητές. Για τον λόγο αυτό τα τοπικά κόστη των κόμβων υπολογίζονται την ώρα που πρέπει να χρησιμοποιηθούν, δηλαδή δεν υπάρχει μεταβλητή η οποία τα να κρατάει. Ακόμα, όσον αφορά τα ολικά κόστη των κόμβων, αντί να διατηρείται ολόκληρος ο πίνακας με τα κόστη, διατηρούμε κάθε φορά τα κόστη του προηγούμενου δείγματος μόνο έτσι ώστε να υπολογίσουμε τα νέα κόστη όπως δείχνει η Εικόνα 33. Αυτό συνεπάγεται ότι χρησιμοποιούμε το ελάχιστο 2x26x4= 208bytes και μέγιστο 2x60x4=480 bytes. Δηλαδή και για τις τρεις χειρονομίες χρησιμοποιούμε 624 με 1140 bytes. Δηλαδή μόνο όσον αφορά το μέγιστο κόστος των πινάκων ολικού κόστους επιτύχαμε **μείωση της χρησιμοποιούμενης μνήμης κατά 97%.**



Εικόνα 33: Κάθε φορά, για τον υπολογισμό του ολικού κόστους ενός δείγματος του προτύπου υπό εξέταση, χρειαζόμαστε μόνο τα ολικά κόστη που είχαμε υπολογίσει για το προηγούμενο δείγμα

5.3 Τεχνικές που μείωσαν τις υπολογιστικές πράξεις

Στο κεφάλαιο 4 όπου πάρθηκαν πολλές αποφάσεις για το σύστημα αναγνώρισης χειρονομιών, προσπαθήσαμε να κρατήσουμε τις υπολογιστικές πράξεις στο ελάχιστο, έτσι ώστε ο αλγόριθμος να τρέχει όσο πιο γρήγορα γίνεται. Αυτό πραγματοποιήθηκε ως εξής:

- Χρήση motion interrupt
- Υπολογισμός κόστους με λιγότερες υπολογιστικές πράξεις
- Επιλογή διακύμανσης αντί για τυπικής απόκλισης

Αρχικά, όπως αναφέρθηκε στην ενότητα 4.4, ο εντοπισμός της αρχής και του τέλους της κίνησης γίνεται με την χρήση της διακύμανσης. Πιο συγκεκριμένα, ο επεξεργαστής πρέπει να υπολογίζει την διακύμανση έξι τιμών κάθε φορά και αν αυτή ξεπερνάει ένα κατώφλι να σηματοδοτεί την αρχή μιας χειρονομίας. Αυτό όμως σημαίνει πως ο επεξεργαστής θα πρέπει συνέχεια να διαβάζει τιμές από την ουρά και να εκτελεί πράξεις, γεγονός που του δίνει επιπλέον φόρτο εργασίας και ανεβάζει την κατανάλωσή ενέργειας. Για τον λόγο αυτό, χρησιμοποιήσαμε το motion_interrupt.

Ακόμα, στην ενότητα επιλογής χαρακτηριστικών (ενότητα 4.3), η επιλογή ενός μόνο άξονα μετέτρεψε τον υπολογισμό του κόστους από το α) που θα είχαμε αν χρησιμοποιούσαμε και τους τρεις άξονες στο β), όπου γλιτώνουμε πράξεις όπως τετραγωνική ρίζα και δυνάμεις του δύο.

$$d = \sqrt{(r(i)_x - t(i)_x)^2 + (r(i)_y - t(i)_y)^2 + (r(i)_z - t(i)_z)^2} \ (\alpha)$$

$$d = |r(i)_y - t(i)_y| \ (\beta)$$

Τέλος, με την επιλογή της διακύμανσης αντί για την τυπική απόκλιση στην ενότητα 4.4 για τον εντοπισμό του τέλους μιας χειρονομίας αποφύγαμε την χρήση μιας τετραγωνικής ρίζας, ενώ και πάλι η επιλογή ενός μόνο άξονα για τον υπολογισμό της διακύμανσης μείωσε τις υπολογιστικές πράξεις.

5.4 Τεχνικές για βελτίωση του χρόνου απόκρισης του αλγορίθμου

Η απόκριση του συστήματος είναι πολύ σημαντική, καθώς μια συσκευή εισόδου πρέπει να ανταποκρίνεται γρήγορα στις επιθυμίες του χρήστη για να θεωρηθεί εύχρηστη. Με σκοπό να τρέχει γρηγορότερα ο αλγόριθμος χρησιμοποιήσαμε τις ακόλουθες τεχνικές:

- εκτέλεση του αλγορίθμου αναγνώρισης παράλληλα με την εκτέλεση της χειρονομίας από τον χρήστη
- Αποφυγή πράξης διαίρεσης αλλάζοντας το μέγεθος που παραθύρου από έξι σε τέσσερα ώστε να χρησιμοποιηθεί ολίσθηση δεξιά για δύο θέσεις.
- επιλογή περιορισμών τοπικών περιορισμών Sakoe Shiba αντί για Itakura
- Χρήση λιγότερων εντολών διακλάδωσης

Αρχικά, για να βελτιώσουμε την απόκριση του συστήματος, ο αλγόριθμος τρέχει παράλληλα με την εκτέλεση μιας χειρονομίας και όχι μετά το πέρας αυτής. Αυτό σημαίνει πως όσο χρόνο ο χρήστης εκτελεί την κίνηση ο αλγόριθμος υπολογίζει τα ενδιάμεσα ολικά κόστη για κάθε ένα από τα πρότυπα αναφοράς.

Ακόμα, το παράθυρο σύμφωνα με το οποίο ελέγχουμε την διακύμανση για να αποφασίσουμε για την λήξη μιας χειρονομίας έγινε τέσσερα, από έξι που ήταν προηγουμένως. Αυτό συνέβη για να γλιτώσουμε χρόνο (καθώς δεν έχουμε FPU) και να μετατραπεί η διαίρεση διά έξι σε μια μετατόπιση των bit προς τα αριστερά κατά δύο θέσεις.

Επίσης, έγινε επιλογή των τοπικών περιορισμών Sakoe Shiba έναντι των περιορισμών Itakura, καθώς οι δεύτεροι έχουν πολλές εντολές διακλάδωσης, οι οποίες θα εισήγαγαν επιπλέον καθυστέρηση στους υπολογισμούς του αλγορίθμου.

Τέλος, ξαναγράφτηκε ο κώδικας του αλγορίθμου έτσι ώστε να χρησιμοποιηθούν λιγότερες εντολές διακλάδωσης υπό συνθήκη, το οποίο μείωσε τον χρόνο εκτέλεσης κατά 1 msec. Οι λιγότερες εντολές διακλάδωσης οδήγησαν σε λίγο μεγαλύτερο μέγεθος κώδικα, το οποίο όμως

είναι αποδεκτό καθώς έχουμε 10 msec για την εκτέλεση του αλγορίθμου και πετύχαμε 10% μείωση του χρόνου εκτέλεσης.

Κεφάλαιο 6: Υλοποίηση του αλγορίθμου στην αναπτυξιακή πλατφόρμα της Dialog Semiconductors

Στις ενότητες που ακολουθούν παρουσιάζεται αναλυτικά ολόκληρη η υλοποίηση του αλγορίθμου στο DA14580 development kit της Dialog Semiconductors. Η συσκευή εισόδου, με το που συνδεθεί με μια άλλη και δεν βρίσκεται σε κατάσταση εκτεταμένου ύπνου για εξοικονόμηση ενέργειας, μπορεί να βρίσκεται σε δύο καταστάσεις:

- σε κατάσταση όπου δεν έχει ξεκινήσει η αναγνώριση κάποιας χειρονομίας και η συσκευή απλά στέλνει αναφορές με την κίνηση του κέρσορα
- σε κατάσταση όπου έχει ξεκινήσει η αναγνώριση μιας χειρονομίας. Σε αυτή την κατάσταση, δεν αποστέλλονται αναφορές με κίνηση του κέρσορα έως ότου να τελειώσει η διαδικασία αναγνώρισης.

Καθώς οι δύο αυτές καταστάσεις έχουν διαφορετικές ανάγκες από την μονάδα μέτρησης αδράνειας έχουν ρυθμίσει διαφορετικά interrupt, τα οποία θα αναλυθούν στην ενότητα που ακολουθεί.

6.1 Ρύθμιση των interrupt στις διαφορετικές καταστάσεις

Όπως αναφέραμε προηγουμένως, η μονάδα μέτρησης αδράνειας παρέχει στον προγραμματιστή ένα πλήθος από προγραμματιζόμενα interrupt, καθένα από τα οποία εξυπηρετεί κάποιον σκοπό. Τα interrupt αυτά ενεργοποιούνται γράφοντας "1" σε συγκεκριμένα bit του καταχωρητή INT_ENABLE, ενώ όταν έχει πυροδοτηθεί, διαβάζοντας τον καταχωρητή INT_STATUS μπορούμε να δούμε ποιο από τα interrupt πυροδοτήθηκε. Τα τρία διαφορετικά interrupt που χρησιμοποιούμε είναι τα εξής:

- motion_interrupt: το interrupt αυτό πυροδοτείται κάθε φορά που έχουμε απότομη κίνηση η οποία συνεπάγεται έναν αριθμό τιμών οι οποίες ξεπερνούν ένα κατώφλι.
- data_ready_interrupt: το interrupt αυτό πυροδοτείται κάθε φορά που γράφονται και οι τρεις τιμές για x,y και z στους καταχωρητές του αισθητηρίου, δηλαδή κάθε φορά που έχουμε νέα δεδομένα
- fifo_overflow: το interrupt αυτό πυροδοτείται κάθε φορά που γεμίζουν τα 1024 bytes της ουράς. Το interrupt αυτό είναι πολύ σημαντικό γιατί στην περίπτωση που τα δεδομένα που αποθηκεύουμε στην ουρά δεν διαιρούνται ακριβώς με το 1024, τότε θα χαθεί η σειρά με την οποία αποθηκεύονται τα νέα δεδομένα (π.χ. θα μπερδευτούν οι άξονες κ.α.) πράγμα το οποίο επιλύεται με την επανεκκίνηση (reset) της ουράς.

Στην περίπτωση που βρισκόμαστε στην κατάσταση όπου το σύστημα αναφέρει την κίνηση του κέρσορα, ο αισθητήρας βάζει τις τιμές που διαβάζει από το επιταχυνσιόμετρο στην ουρά που διαθέτει, και είναι η δουλειά του προγραμματιστή να αδειάζει τακτικά την ουρά ώστε να μη παθαίνει υπερχείλιση. Ακόμα, πρέπει να είναι έτοιμο να ξεκινήσει την αναγνώριση μιας χειρονομίας, στην περίπτωση που πραγματοποιηθεί κάποια από τον χρήστη. Για τον λόγο αυτό, στην κατάσταση αυτή είναι ενεργοποιημένο το motion_interrupt ώστε να ξεκινήσει η αναγνώριση μιας χειρονομίας όταν χρειαστεί. Επίσης, είναι ενεργοποιημένο το fifo_overflow_interrupt, καθώς πρέπει να γνωρίζουμε πότε συμβαίνει υπερχείλιση της ουράς, καθώς αν δεν πραγματοποιηθεί επανεκκίνησή της, τα δεδομένα που θα λάβουμε θα είναι λάθος.

Στην περίπτωση που το σύστημα βρίσκεται στην κατάσταση αναγνώρισης μιας χειρονομίας (το οποίο συνεπάγεται ότι έχει πυροδοτηθεί το motion_interrupt), το motion_interrupt απενεργοποιείται καθώς χρησιμοποιείται μόνο για να σηματοδοτεί την αρχή μιας χειρονομίας. Επίσης, είναι ενεργοποιημένο το fifo_overflow_interrupt, καθώς θέλουμε και πάλι να ελέγχουμε για τυχόν υπερχείλιση της ουράς. Τέλος, καθώς χρησιμοποιούμε ένα παράθυρο για να ελέγξουμε για το τέλος της χειρονομίας, πρέπει να γνωρίζουμε κάθε φορά που

έχουμε καινούργια δεδομένα από το επιταχυνσιόμετρο. Για τον λόγο αυτό είναι ενεργοποιημένο το data_ready_interrupt.

6.2 Προτεραιότητα των διαφορετικών interrupt

Όπως αναφέραμε προηγουμένως, στην περίπτωση που πυροδοτηθεί ένα ή περισσότερα από τα interrupt, είναι δουλειά του προγραμματιστή να εντοπίσει ποιο ή ποια από αυτά πυροδοτήθηκαν στο interrupt service routine διαβάζοντας την τιμή του καταχωρητή INT_STATUS.

Καθώς δεν είναι απαγορευτικό να πυροδοτηθούν δύο ή και περισσότερα interrupt την ίδια στιγμή (π.χ. data_ready_interrupt μαζί με fifo_overflow_interrupt), πρέπει να δοθεί κάποια προτεραιότητα ανάμεσα τους. Εφόσον οι δύο καταστάσεις της συσκευής υπαγορεύουν πως δεν μπορεί να συνυπάρξει motion_interrupt με data_ready_interrupt, πρέπει να εξετάσουμε την προτεραιότητα τους με το fifo_overflow_interrupt.

Στην περίπτωση που συμβεί motion_interrupt μαζί με motion_overflow (γεγονός το οποίο δεν πρέπει να συμβεί καθώς η ουρά διαβάζεται τακτικά), η ISR θα λάβει μέτρα για την αντιμετώπιση της υπερχείλισης, καθώς τα δεδομένα της ουράς δεν θα είναι σε σωστή σειρά έως ότου γίνει επανεκκίνηση της ουράς.

Στην περίπτωση που συμβεί data_ready_interrupt μαζί με motion_overflow, η ISR θα προβεί σε επανεκκίνηση της ουράς και θα βγει από την κατάσταση αναγνώρισης χειρονομιών καθώς τα δεδομένα της ουράς δεν θα είναι σωστά.

6.3 Επιλογή ευαισθησίας επιταχυνσιόμετρου

Το επιταχυνσιόμετρο που συμπεριλαμβάνεται στην μονάδα μέτρησης της αδράνειας που χρησιμοποιούμε, μας δίνει την δυνατότητα να επιλέξουμε το εύρος των τιμών της επιτάχυνσης που μπορεί να μετρήσει [11], γράφοντας την αντίστοιχη τιμή AFS_SEL στον καταχωρητή ACCEL_CONFIG, όπως δείχνει και Εικόνα 34.

4.5 Register 28 – Accelerometer Configuration ACCEL_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

Εικόνα 34: Ο καταχωρητής ACCEL_CONFIG

Καθώς το σύστημα θέλει να παρακολουθεί την κίνηση του δαχτύλου, θέσαμε το εύρος των τιμών που μπορεί να μετρήσει το επιταχυνσιόμετρο σε ±2g, δηλαδή AFS_SEL=0.

AFS_SEL	Full scale range
0	±2 g
1	±4 g
2	±8 g
3	±16 g

6.4 Αποθήκευση δεδομένων στην ουρά

Η ουρά που διαθέτει ο αισθητήρας, μπορεί να αποθηκεύσει τις τιμές από το επιταχυνσιόμετρο, τον άξονα x/y/z του γυροσκοπίου ή/και τιμές θερμοκρασίας, όπως παρουσιάζεται στην Εικόνα 35.

Καθώς δεν χρειάζεται να κρατάμε όλες τις τιμές από το γυροσκόπιο, και κάθε φορά που χρειαζόμαστε τις τιμές του (κάθε 10msec) απλά διαβάζουμε τους αντίστοιχους καταχωρητές, στην ουρά αποθηκεύονται μόνο οι τιμές από το επιταχυνσιόμετρο. Αυτό πραγματοποιείται γράφοντας ένα "1" στο τέταρτο bit του καταχωρητή FIFO_EN.

4.7 Register 35 – FIFO Enable
FIFO_EN

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
23	35	TEMP_ FIFO_EN	XG_ FIFO_EN	YG_ FIFO_EN	ZG_ FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN

Εικόνα 35: Ο καταχωρητής FIFO_EN

Ο καταχωρητής αυτός καθορίζει ποιες μετρήσεις του αισθητήρα θα αποθηκευτούν στην ουρά. Τα δεδομένα τα οποία είναι αποθηκευμένα στους καταχωρητές δεδομένων του αισθητήρα, θα φορτωθούν στην ουρά αν το αντίστοιχο bit στον καταχωρητή FIFO_EN είναι άσσος. Η δειγματοληψία του αισθητήρα γίνεται σύμφωνα με την τιμή SMPRT_DIV(Εικόνα 36) στον καταχωρητή SampleRateDivider. Η τιμή της δειγματοληψίας υπολογίζεται διαιρώντας τον ρυθμό εξόδου του επιταχυνσιόμετρου ο οποίος είναι 1kHz με την τιμή SMPRT_DIV

Sample Rate = Accelerometer Output Rate / (1 + SMPRT_DIV)

(βλέπε ενότητα 4.1).

4.2 Register 25 – Sample Rate Divider
SMPRT_DIV

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
19	25	SMPLRT_DIV[7:0]							

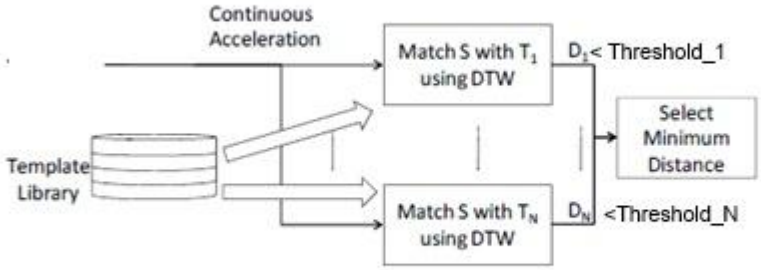
Εικόνα 36: Ο καταχωρητής SMPRT_DIV

Για να επιτύχουμε την δειγματοληψία των 100Hz του επιταχυνσιόμετρου, η τιμή που πρέπει να δώσουμε στο SMPRT_DIV είναι ίση με 9.

6.5 Επεξήγηση του FSM της αναγνώρισης χειρονομιών

Ο μηχανισμός αναγνώρισης χειρονομιών που χρησιμοποιείται είναι πολύ απλός. Κάθε φορά που έχουμε την έναρξη μιας χειρονομίας, οι συνεχείς τιμές της επιτάχυνσης που ανήκουν στην χειρονομία δίνονται ως είσοδος στον αλγόριθμο DTW. Αυτός, συγκρίνει τις τιμές αυτές με όλα τα πρότυπα αναφοράς T_i που έχει στην βιβλιοθήκη του και υπολογίζει ένα κόστος D_i το οποίο υποδεικνύει το πόσο ταιριάζει η χειρονομία με το κάθε πρότυπο αναφοράς.

Στην συνέχεια, αν το κόστος αυτό είναι μικρότερο από ένα ανώτερο κατώφλι κόστους Th_i το οποίο επιτρέπεται να έχει το κάθε πρότυπο αναφοράς, βρίσκουμε το πρότυπο εκείνο που έχει το μικρότερο κόστος και αναγνωρίζουμε την κίνηση ως μια έγκυρη κίνηση που ανήκει στην ίδια κατηγορία με αυτό. Στην περίπτωση που τα κόστη αυτά είναι μεγαλύτερα από τα αντίστοιχα Th_i θεωρούμε πως η χειρονομία αυτή δεν ανήκει σε καμία από τις γνωστές μας χειρονομίες και συνεπώς απορρίπτεται, όπως δείχνει η Εικόνα 37.



Εικόνα 37: Ο αλγόριθμος δέχεται ως είσοδο συνεχείς τιμές επιτάχυνσης και τις συγκρίνει με τα διάφορα πρότυπα αναφοράς

Ολόκληρη η διαδικασία αναγνώρισης χειρονομιών συντονίζεται από ένα finite state machine. Η εναλλαγή των διαφόρων καταστάσεων πραγματοποιείται λαμβάνοντας υπόψη τα διάφορα interrupt που πυροδοτούνται.

Το σύστημά μας μπορεί να βρίσκεται σε μία από τις ακόλουθες καταστάσεις:

- **IDLE_GST**, η μοναδική κατάσταση του συστήματος στην οποία δεν πραγματοποιείται αναγνώριση κάποιας χειρονομίας. Στην κατάσταση αυτή το σύστημα αναφέρει την κίνηση του κέρσορα και αναμένει κάποιο motion_interrupt
- **START_GST**, η κατάσταση στην οποία έχει μόλις ξεκινήσει η αναγνώριση της χειρονομίας. Αποτελεί την πρώτη από τις δύο καταστάσεις στις οποίες εκτελείται ο αλγόριθμος DTW. Στην κατάσταση αυτή, υπολογίζεται το κόστος για τα δέκα πρώτα δείγματα της χειρονομίας που θέλουμε να αναγνωρίσουμε.
- **LOG_GST**, η δεύτερη από τις δύο καταστάσεις στις οποίες εκτελείται ο αλγόριθμος DTW. Το σύστημα παραμένει σε αυτήν την κατάσταση έως ότου υπάρξει υπερχείλιση της ουράς, έρθει το τέλος της χειρονομίας ή το μέγεθος της ξεπεράσει τα 60 στοιχεία.
- **END_GST**, η κατάσταση στην οποία συγκρίνονται τα κόστη που έχουν υπολογισθεί από τον αλγόριθμο και αν πρόκειται για έγκυρη κίνηση μπαίνει στον gesturebuffer έτσι ώστε να αποσταλεί αναφορά με την συγκεκριμένη χειρονομία.
- **PREPARE_NEXT_GST**, η κατάσταση στην οποία έχει τελειώσει η διαδικασία αναγνώρισης και το σύστημα αρχικοποιεί τις μεταβλητές, απενεργοποιεί το data_ready_interrupt ενώ ενεργοποιεί ξανά το motion_interrupt.

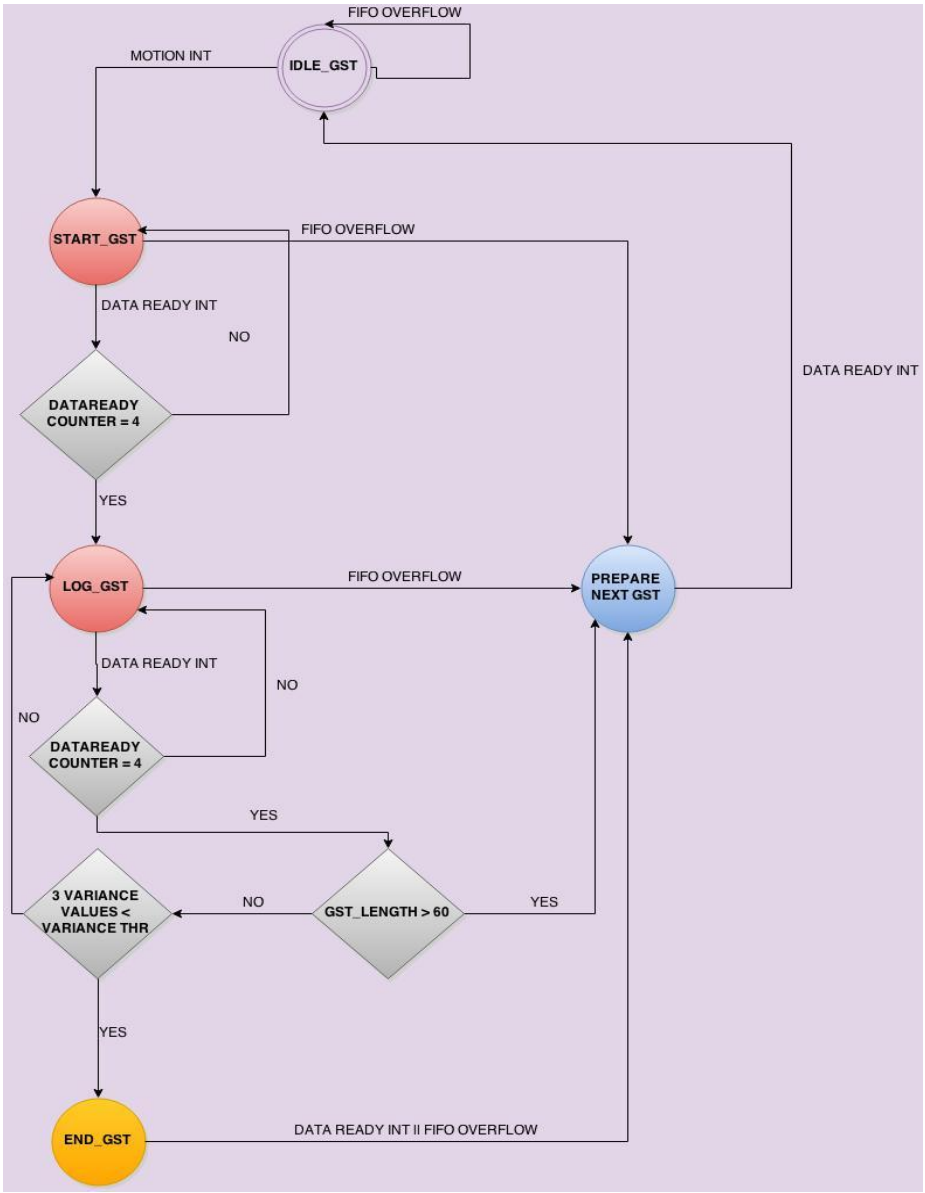
Στην συνέχεια αναλύονται οι διάφορες καταστάσεις της αναγνώρισης χειρονομιών καθώς επίσης και οι μεταβάσεις από κατάσταση σε κατάσταση ανάλογα με το γεγονός που έχει συμβεί.

Τρέχουσα κατάσταση	**Γεγονός**	**Περιγραφή**
IDLE_GST	fifo overflow	στην περίπτωση που πυροδοτηθεί fifo_overflow_interrupt, τότε θα γίνει επανεκκίνηση της ουράς χωρίς να υπάρξει αλλαγή κατάστασης.
	motion interrupt	στην περίπτωση που πυροδοτηθεί motion_interrupt, τότε έχουμε την έναρξη μιας χειρονομίας και το

		σύστημα μεταβαίνει στην κατάσταση START_GST.
START_GST	fifo overflow	στην περίπτωση που πυροδοτηθεί fifo_overflow_interrupt, τότε θα γίνει επανεκκίνηση της ουράς, το σύστημα θα βγει απο την φάση της αναγνώρισης και θα μεταβεί στην κατάσταση PREPARE_NEXT_GST, έτσι ώστε να αρχικοποιηθεί ξανά (να ενεργοποιηθούν τα κατάλληλα interrupt, να αρχικοποιηθούν μεταβλητές κοκ).
	data ready interrupt	μετά την μετάβαση στην τρέχουσα κατάσταση, στο πρώτο data_ready_interrupt που θα πραγματοποιηθεί, παίρνουμε τα έξι τελευταία στοιχεία της ουράς (τα οποία αποτελούν την αρχή της χειρονομίας μας) και τα συγκρίνουμε με τον DTW με κάθε ένα από τα πρότυπα αναφοράς. Στα υπόλοιπα data_ready_interrupt, περιμένουμε πότε θα έχουμε το τέταρτο έτσι ώστε να εκτελέσουμε πάλι τον αλγόριθμο για τις τέσσερις αυτές τιμές του υπό εξέταση προτύπου και να μεταβούμε στην κατάσταση LOG_GST.
LOG_GST	fifo overflow	στην περίπτωση που πυροδοτηθεί fifo_overflow_interrupt, τότε θα γίνει επανεκκίνηση της ουράς, το σύστημα θα βγει απο την φάση της αναγνώρισης και θα μεταβεί στην κατάσταση PREPARE_NEXT_GST, έτσι ώστε να αρχικοποιηθεί ξανά (να ενεργοποιηθούν τα κατάλληλα interrupt, να αρχικοποιηθούν μεταβλητές κοκ).
	data ready interrupt	μετά την μετάβαση στην τρέχουσα κατάσταση, στο τέταρτο data_ready_interrupt που θα πραγματοποιηθεί, παίρνουμε τα τέσσερα στοιχεία της ουράς, τα συγκρίνουμε με τον DTW με κάθε ένα από τα πρότυπα αναφοράς και βρίσκουμε την διακύμανσή τους στον άξονα z. Αν η διακύμανση είναι μικρότερη από το κατώφλι, τότε αυξάνεται ένας μετρητής που χρησιμοποιείται για να σηματοδοτήσει την λήξη της χειρονομίας. Αν η διακύμανση είναι μεγαλύτερη τότε μηδενίζεται ο μετρητής. Στην περίπτωση που το μήκος της χειρονομίας είναι μεγαλύτερο από 60, τότε η χειρονομία απορρίπτεται και το σύστημα μεταβαίνει στην κατάσταση PREPARE_NEXT_GST, έτσι ώστε να αρχικοποιηθεί ξανά. Στην περίπτωση που ο μετρητής είναι μικρότερος από τέσσερα, τότε το σύστημα παραμένει στην ίδια κατάσταση, ενώ αν είναι ίσος με τέσσερα, τότε η χειρονομία έχει φτάσει στο τέλος της, οπότε μεταβαίνουμε στην κατάσταση END_GST.

END_GST	fifo overflow	στην περίπτωση που πυροδοτηθεί fifo_overflow_interrupt, τότε θα γίνει επανεκκίνηση της ουράς, το σύστημα θα μεταβεί στην κατάσταση PREPARE_NEXT_GST, έτσι ώστε να αρχικοποιηθεί ξανά (να ενεργοποιηθούν τα κατάλληλα interrupt, να αρχικοποιηθούν μεταβλητές κοκ).
	data ready interrupt	μετά την μετάβαση στην τρέχουσα κατάσταση, στο πρώτο data_ready_interrupt που θα πραγματοποιηθεί, θα συγκριθούν όλα τα κόστη D_i που έχουν προκύψει συγκρίνοντας με τον DTW το υπό εξέταση πρότυπο με κάθε ένα από τα πρότυπα αναφοράς T_i . Στην συνέχεια, αν το κόστος αυτό είναι μικρότερο από ένα ανώτερο κατώφλι κόστους Th_i το οποίο επιτρέπεται να έχει το κάθε πρότυπο αναφοράς, βρίσκουμε το πρότυπο εκείνο που έχει το μικρότερο κόστος και αναγνωρίζουμε την κίνηση ως μια έγκυρη κίνηση που ανήκει στην ίδια κατηγορία με αυτό. Στην περίπτωση που τα κόστη αυτά είναι μεγαλύτερα από τα αντίστοιχα Th_i θεωρούμε πως η χειρονομία αυτή δεν ανήκει σε καμία από τις γνωστές μας χειρονομίες και συνεπώς απορρίπτεται. Το σύστημα μεταβαίνει στην κατάσταση PREPARE_NEXT_GST.
PREPARE_NEXT_GST	fifo overflow	η κατάσταση στην οποία έχει τελειώσει η διαδικασία αναγνώρισης και το σύστημα αρχικοποιεί τις μεταβλητές, απενεργοποιεί το data_ready_interrupt ενώ ενεργοποιεί ξανά το motion_interrupt.
	data ready interrupt	μετά την μετάβαση στην τρέχουσα κατάσταση, στο πρώτο data_read_yinterrupt που θα πραγματοποιηθεί, το σύστημα θα ρυθμίσει κατάλληλα τα interrupt, θα αρχικοποιήσει τις μεταβλητές του και θα μεταβεί στην κατάσταση IDLE_GST.

Στην Εικόνα 38, παρουσιάζεται αναλυτικά το Finite State Machine που διαχειρίζεται την διαδικασία της αναγνώρισης.



Εικόνα 38: FSM της αναγνώρισης χειρονομιών. Ο αλγόριθμος DTW εκτελείται στις καταστάσεις START_GST και LOG_GST

Ακόμα, ο τρόπος με τον οποίο έχει δημιουργηθεί το finite state machine της αναγνώρισης, καθορίζει το μέγεθος της ελάχιστης και μέγιστης χειρονομίας η οποία μπορεί να αναγνωριστεί. Όπως είδαμε προηγουμένως, το μέγιστο επιτρεπόμενο μήκος μιας χειρονομίας είναι 60, καθώς κάθε χειρονομία που ξεπερνάει αυτό το μέγεθος απορρίπτεται.

Εφόσον στην κατάσταση START_GST, πραγματοποιείται ο υπολογισμός του κόστους των δέκα πρώτων τιμών του υπό εξέταση προτύπου και για να τερματίσει μια χειρονομία πρέπει η διακύμανση τεσσάρων δειγμάτων ίσων με το παράθυρό μας (τέσσερα) να είναι μικρότερη από ένα κατώφλι, βλέπουμε πως το ελάχιστο μήκος που μπορεί να έχει μια χειρονομία είναι 26.

6.6 Περιορισμοί της συσκευής

Η συσκευή εισόδου που αναπτύχθηκε διαθέτει διάφορους περιορισμούς στην λειτουργία της. Αρχικά, η αναγνώριση των χειρονομιών είναι εξατομικευμένη, καθώς τα πρότυπα αναφοράς έχουν επιλεχθεί από τις κινήσεις ενός μόνο χρήστη. Αυτό συνεπάγεται πως η αναγνώριση των χειρονομιών δεν θα λειτουργεί τόσο καλά στην περίπτωση που το χειριστεί κάποιος άλλος χρήστης χωρίς πρώτα να προηγηθεί η διαδικασία της επιλογής προτύπων αναφοράς η οποία αναλύεται στο κεφάλαιο επτά.

Ακόμα, το γεγονός ότι δεν έχει αφαιρεθεί το διάνυσμα της βαρύτητας από τις τιμές της επιτάχυνσης επιβάλλει στον χρήστη να έχει το χέρι του παράλληλα με το έδαφος όταν πραγματοποιεί τις διάφορες χειρονομίες καθώς σε οποιαδήποτε άλλη περίπτωση δεν θα αναγνωριστούν σωστά. Αυτό συμβαίνει επειδή όταν πραγματοποιήθηκε η επιλογή των προτύπων αναφοράς, ο χρήστης εκτελούσε τις διάφορες κινήσεις έχοντας το χέρι παράλληλα με το έδαφος.

Τέλος, ο αριθμός των προτύπων αναφοράς και συνεπώς ο αριθμός των χειρονομιών τις οποίες μπορεί να αναγνωρίσει η συσκευή εξαρτάται από την μνήμη που έχουμε στην διάθεσή μας και το κατά πόσο ο αλγόριθμος προλαβαίνει να υπολογίσει το κόστος για κάθε πρότυπο ανάλογα με την δειγματοληψία που έχουμε επιλέξει.

Κεφάλαιο 7: Βοηθητικό πρόγραμμα για λήψη προτύπων αναφοράς, μέτρηση χρόνων εκτέλεσης αλγορίθμου και επαλήθευσης των προτύπων αναφοράς

Όπως αναλύσαμε στα προηγούμενα κεφάλαια, για να επιλέξουμε την δειγματοληψία, τις διάφορες μεταβλητές που καθορίζουν το πότε πυροδοτείται το motion_interrupt κ.α. πραγματοποιήσαμε διάφορες χειρονομίες και παρατηρήσαμε τις γραφικές παραστάσεις της επιτάχυνσης. Για να συμβεί αυτό, χρειαζόμαστε ένα πρόγραμμα το οποίο επιτρέπει την αλλαγή των διάφορων μεταβλητών και εκτυπώνει τις ακόλουθες χειρονομίες με την χρήση σειριακής .

Ακόμα, η επιλογή της δειγματοληψίας και του χρόνου εκτέλεσης του αλγορίθμου αποτελούν ζητήματα υψίστης σημασίας, καθώς οποιαδήποτε καθυστέρηση (αν η εκτέλεση του αλγορίθμου διαρκούσε πάνω από 10 msec) θα είχε ως αποτέλεσμα να χαθεί κάποιο data_ready_interrupt. Για τον λόγο αυτό, το βοηθητικό αυτό πρόγραμμα περιλαμβάνει μια επιλογή με την οποία μπορούμε να μετρήσουμε τον χρόνο εκτέλεσης του interrupt service routine (και συνεπώς του αλγορίθμου αναγνώρισης χειρονομιών) έτσι ώστε να βεβαιωθούμε ότι δεν ξεπερνάει τον χρόνο εκείνο που έχουμε στη διάθεσή μας.

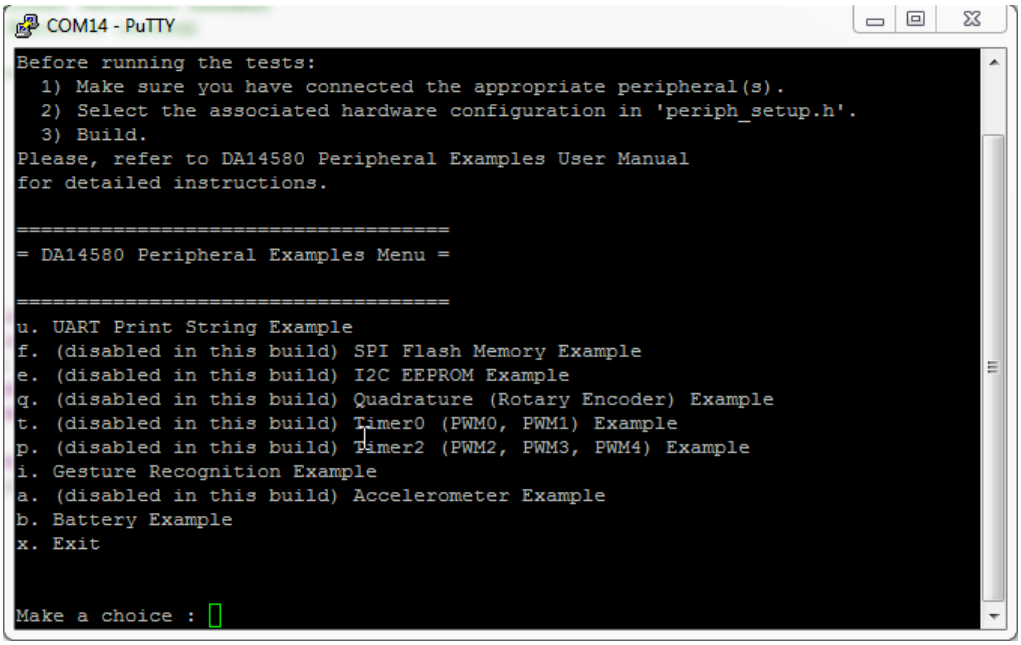
Επίσης, το βοηθητικό αυτό πρόγραμμα παρέχει την επιλογή της εκτύπωσης πολλών συνεχόμενων χειρονομιών στην σειριακή, με σκοπό να χρησιμοποιηθούν στην διαδικασία της επιλογής των προτύπων αναφοράς που θα αναλυθεί στην συνέχεια.

Τέλος, το βοηθητικό αυτό πρόγραμμα δίνει την δυνατότητα στο χρήστη να εκτελεί διάφορες χειρονομίες με τα πρότυπα αναφοράς που επέλεξε και να βλέπει στην οθόνη μέσω εκτύπωσης στην σειριακή σαν τι χειρονομία αναγνωρίστηκε η κίνησή του. Με τον τρόπο αυτό επιβεβαιώνεται η σωστή επιλογή των συγκεκριμένων προτύπων αναφοράς.

Το πρόγραμμα είναι βασισμένο στον κώδικα peripheral_examples της Dialog semiconductors. Το βασικό μέρος του κώδικα βρίσκεται στο αρχείο DA14580_examples.c, όπου γίνεται η αρχικοποίηση του αισθητήρα και του motioninterrupt. Στην συνέχεια εισέρχεται σε ένα ατέρμονο βρόχο, όπου ελέγχει μια μεταβλητή η οποία εκφράζει την ύπαρξη ενός interrupt και στην συνέχεια αδειάζει την ουρά έτσι ώστε να αποφύγει την υπερχείλιση. Στην περίπτωση που πυροδοτηθεί ένα motion_interrupt, τότε αυτό απενεργοποιείται και το σύστημα εισέρχεται σε κατάσταση αναγνώρισης χειρονομιών. Μόλις φθάσει στο τέρμα η διαδικασία αναγνώρισης, ενεργοποιείται ξανά το motion_interrupt και περιμένει να συμβεί η επόμενη κίνηση.

Εκτελώντας το πρόγραμμα, και ανοίγοντας ένα πρόγραμμα για την εκτύπωση των δεδομένων της σειριακής, εμφανίζεται το μενού που παρουσιάζεται στην Εικόνα 39. Ο χρήστης, πρέπει να πληκτρολογήσει "i" έτσι ώστε να μπορεί να χρησιμοποιήσει το παράδειγμα του gesturerecognition. Στο παράδειγμα αυτό ανάλογα με την τιμή της μεταβλητής APP_MODE (την οποία την αλλάζουμε προγραμματιστικά) έχουμε τις ακόλουθες λειτουργίες:

- λειτουργία επιλογής προτύπων αναφοράς (TEMPLATE_SELECTION), στην λειτουργία αυτή ο χρήστης εκτελεί διάφορες χειρονομίες οι συνεχείς τιμές επιτάχυνσης των οποίων εκτυπώνονται στην σειριακή
- λειτουργία επιβεβαίωσης προτύπων αναφοράς (TEMPLATE_TESTING), στην λειτουργία αυτή ο χρήστης εκτελεί μία χειρονομία και ο αλγόριθμος εκτυπώνει στην σειριακή σαν τι χειρονομία την αναγνώρισε
- λειτουργία μέτρησης χρόνων εκτέλεσης του αλγορίθμου (TIME_MEASURE), στην λειτουργία αυτή χρησιμοποιείται ένας χρονομετρητής για να μετρήσουμε πόσο χρόνο διαρκεί η εκτέλεση κάθε τμήματος του αλγορίθμου αναγνώρισης χειρονομιών.



Εικόνα 39: Το μενού επιλογών του βοηθητικού προγράμματος

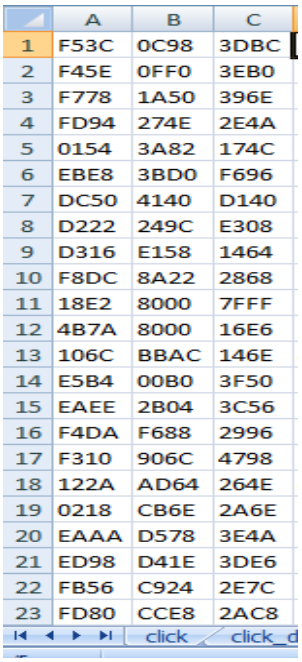
Στις ενότητες που ακολουθούν αναλύονται οι διάφορες επιλογές που μας δίνει το βοηθητικό πρόγραμμα καθώς επίσης και ολόκληρη η διαδικασία της επιλογής των προτύπων αναφοράς.

7.1 Επιλογή προτύπων αναφοράς

Στην παρούσα ενότητα αναλύεται η διαδικασία που ακολουθείται για την επιλογή των προτύπων αναφοράς. Για την διαδικασία αυτή απαιτείται η χρήση του MatLab, του Microsoft Excel και του Putty.

Βάζοντας την επιλογή **TEMPLATE_SELECTION** στην μεταβλητή APP_MODE του προγράμματος, μας δίνεται η δυνατότητα να εκτυπώσουμε στην σειριακή μια σειρά από συνεχόμενες χειρονομίες, με σκοπό να χρησιμοποιηθούν για την επιλογή των προτύπων αναφοράς.

Ο χρήστης, φοράει το δαχτυλίδι στον δείκτη, και πραγματοποιεί μια σειρά από συνεχόμενες ίδιες χειρονομίες π.χ. 30 click, οι οποίες εκτυπώνονται στην σειριακή με χρήση μιας κονσόλας (PUTTY). Στην συνέχεια, οι 30 αυτές χειρονομίες αποθηκεύονται σε ένα .txt από όπου φορτώνονται στο excel, χρησιμοποιώντας το κόμμα ως οριοθέτη. Προσοχή, στην επιλογή της μορφοποίησης στήλης δεδομένων με πατάμε την σειρά πάνω σε κάθε μία από τις τρεις στήλες και επιλέγουμε μορφοποίηση στήλης δεδομένων: κείμενο. Τα δεδομένα μας θα πρέπει να έχουν την ίδια μορφή όπως φαίνεται στην Εικόνα 40.



	A	B	C
1	F53C	0C98	3DBC
2	F45E	0FF0	3EB0
3	F778	1A50	396E
4	FD94	274E	2E4A
5	0154	3A82	174C
6	EBE8	3BD0	F696
7	DC50	4140	D140
8	D222	249C	E308
9	D316	E158	1464
10	F8DC	8A22	2868
11	18E2	8000	7FFF
12	4B7A	8000	16E6
13	106C	BBAC	146E
14	E5B4	00B0	3F50
15	EAEE	2B04	3C56
16	F4DA	F688	2996
17	F310	906C	4798
18	122A	AD64	264E
19	0218	CB6E	2A6E
20	EAAA	D578	3E4A
21	ED98	D41E	3DE6
22	FB56	C924	2E7C
23	FD80	CCE8	2AC8

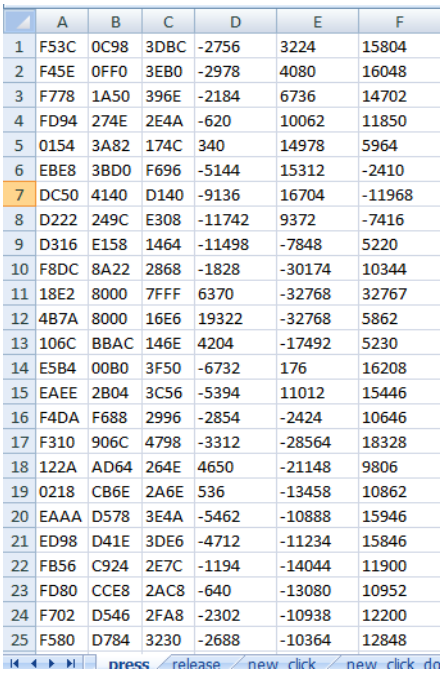
click click_d

Εικόνα 40: Εισαγωγή των χειρονομιών σε Excel

Στην συνέχεια, βάζουμε τον εξής τύπο στο πρώτο κελί της στήλης D

=MOD(HEX2DEC(A1)+POWER(2;15);POWER(2;16))-POWER(2;15),

και επιλέγοντας το κάτω δεξιά μέρος του κελιού D1, σέρνουμε προς τα δεξιά έτσι ώστε να εφαρμοστεί ο τύπος στην πρώτη γραμμή των δύο επόμενων στηλών. Στην συνέχεια, επιλέγοντας το κάτω δεξιά μέρος του κουτιού F1, και σέρνοντας προς τα κάτω για όσο έχουμε δεδομένα στις τρεις πρώτες στήλες, καταφέρνουμε να εφαρμοστεί ο τύπος σε όλα τα δεδομένα. Τα δεδομένα μας θα πρέπει να είναι τώρα όπως παρουσιάζει η Εικόνα 41.

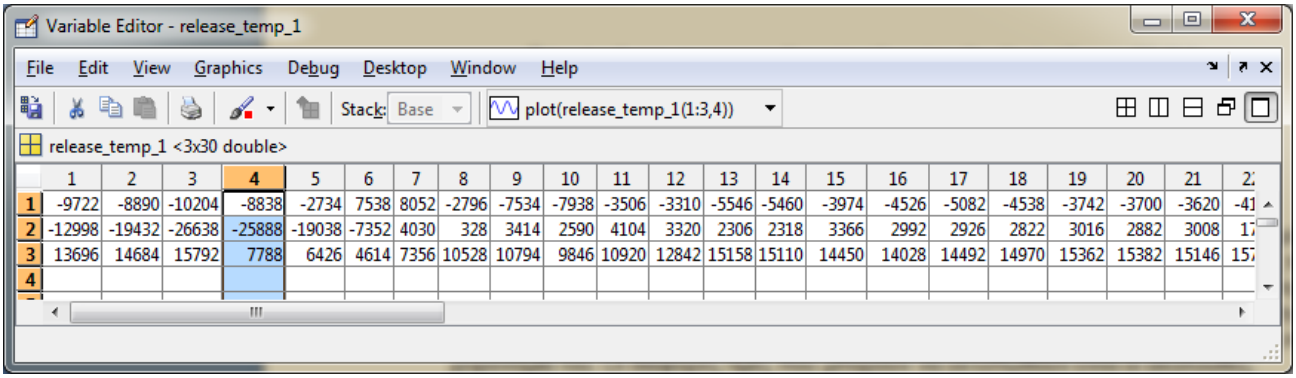


	A	B	C	D	E	F
1	F53C	0C98	3DBC	-2756	3224	15804
2	F45E	0FF0	3EB0	-2978	4080	16048
3	F778	1A50	396E	-2184	6736	14702
4	FD94	274E	2E4A	-620	10062	11850
5	0154	3A82	174C	340	14978	5964
6	EBE8	3BD0	F696	-5144	15312	-2410
7	DC50	4140	D140	-9136	16704	-11968
8	D222	249C	E308	-11742	9372	-7416
9	D316	E158	1464	-11498	-7848	5220
10	F8DC	8A22	2868	-1828	-30174	10344
11	18E2	8000	7FFF	6370	-32768	32767
12	4B7A	8000	16E6	19322	-32768	5862
13	106C	BBAC	146E	4204	-17492	5230
14	E5B4	00B0	3F50	-6732	176	16208
15	EAEE	2B04	3C56	-5394	11012	15446
16	F4DA	F688	2996	-2854	-2424	10646
17	F310	906C	4798	-3312	-28564	18328
18	122A	AD64	264E	4650	-21148	9806
19	0218	CB6E	2A6E	536	-13458	10862
20	EAAA	D578	3E4A	-5462	-10888	15946
21	ED98	D41E	3DE6	-4712	-11234	15846
22	FB56	C924	2E7C	-1194	-14044	11900
23	FD80	CCE8	2AC8	-640	-13080	10952
24	F702	D546	2FA8	-2302	-10938	12200
25	F580	D784	3230	-2688	-10364	12848

press release new_click new_click_do

Εικόνα 41: Μετατροπή της επιτάχυνσης σε 16δικό

Έπειτα, οι χειρονομίες μεταφέρονται μία-μία (αφού έχει γίνει πρώτα αντιμετάθεση και τα δεδομένα μας από τρεις στήλες είναι πλέον 3 γραμμές) σε πίνακες Matlab,όπου και αποθηκεύονται με όνομα : gesture_temp_i, όπου gesture= click ή press ή release και i=1,2,...,30, όπως φαίνεται στην Εικόνα 42.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	2
1	-9722	-8890	-10204	-8838	-2734	7538	8052	-2796	-7534	-7938	-3506	-3310	-5546	-5460	-3974	-4526	-5082	-4538	-3742	-3700	-3620	-41
2	-12998	-19432	-26638	-25888	-19038	-7352	4030	328	3414	2590	4104	3320	2306	2318	3366	2992	2926	2822	3016	2882	3008	17
3	13696	14684	15792	7788	6426	4614	7356	10528	10794	9846	10920	12842	15158	15110	14450	14028	14492	14970	15362	15382	15146	157
4																						

Εικόνα 42: Εισαγωγή των χειρονομιών σε πίνακες στο Matlab

Η ίδια τεχνική ακολουθείται και για τις υπόλοιπες δύο χειρονομίες, έως ότου έχουμε στο MatLab 30 πίνακες για κάθε χειρονομία.

Στην συνέχεια, ακολουθεί η σύγκριση κάθε τύπου χειρονομίας με όλες τις υπόλοιπες χειρονομίες ίδιου τύπου στο Matlab χρησιμοποιώντας τον εξής κώδικα (ο κώδικας αυτός αναφέρεται στην εύρεση του κόστους συγκρίνοντας τα click μεταξύ τους χρησιμοποιώντας τον αλγόριθμο DTW, παρόμοια φτιάχνουμε τον κώδικα για τα press και τα release):

```
loop=30;
cost_click=zeros(loop,loop);
fori=1:loop
for j=1:loop
    temp_1=load(strcat('click_temp_',num2str(i)));
    temp11=temp_1.(strcat('click_temp_',num2str(i)));
    temp_2=load(strcat('click_temp_',num2str(j)));
    temp22=temp_2.(strcat('click_temp_',num2str(j)));
cost_click(i,j)=DTWSakoe(temp11(2,:),temp22(2,:));
end;
end;
```

Ο πίνακας cost_click, περιέχει το κόστος που επέστρεψε ο αλγόριθμος DTW συγκρίνοντας κάθε click με όλα τα υπόλοιπα click. Κάνουμε αντιγραφή τα δεδομένα του πίνακα και τα επικολλούμε σε ένα έγγραφο excel. Εκεί, βρίσκουμε τον μέσο όρο των τιμών κάθε γραμμής (ή στήλης). Στην περίπτωση που ορισμένες γραμμές (ή στήλες) έχουν πολύ μεγαλύτερο μέσο όρο κόστους από τις υπόλοιπες τις απορρίπτουμε και υπολογίζουμε ξανά τον μέσο όρο χωρίς να χρησιμοποιούμε τις γραμμές και στήλες που έχουμε απορρίψει. Αυτό συμβαίνει γιατί κάποιες φορές μπορεί να μην έχουμε εκτελέσει σωστά την χειρονομία και δεν θέλουμε να επηρεαστεί το αποτέλεσμα. Π.χ. αν εκτελέσουμε αργά την χειρονομία click μπορεί ο αλγόριθμος να κόψει την χειρονομία πριν το σήκωμα του δαχτύλου και να έχουμε ένα πρότυπο press μέσα στα πρότυπα click μας.

Η γραμμή με το μικρότερο μέσο κόστος αποτελεί το πρότυπο αναφοράς για την συγκεκριμένη χειρονομία.

Αφού επαναλάβουμε την διαδικασία και για τις υπόλοιπες χειρονομίες, και έχουμε επιλέξει τα τρία πρότυπα αναφοράς, ακολουθεί μια μικρή επιβεβαίωση των τριών προτύπων αναφοράς. Για την επιβεβαίωση αυτή, συγκρίνουμε όλα τα πρότυπα που έχουμε στην διάθεση μας για μια συγκεκριμένη χειρονομία με τα τρία πρότυπα αναφοράς που επιλέξαμε. Επιθυμούμε

το πρότυπο της ίδιας χειρονομίας να βγάζει μικρότερο κόστος από ότι τα πρότυπα διαφορετικών χειρονομιών. όπως παρουσιάζει η Εικόνα 43, όταν η άγνωστη χειρονομία μας είναι τύπου click, όταν συγκρίνεται με τα τρία διαφορετικά πρότυπα αναφοράς, το πρότυπο αναφοράς που ανήκει στην χειρονομία click πρέπει να δίνει μικρότερο κόστος από ότι τα πρότυπα αναφοράς για press και release.

Unknown Gesture	Click Template	Press Template	Release Template
click	90000	120000	134000

Εικόνα 43: Όταν η χειρονομία υπό εξέταση είναι τύπου click, όταν συγκρίνεται χρησιμοποιώντας τον αλγόριθμο DTW με το click-πρότυπο αναφοράς, πρέπει να έχει μικρότερο κόστος από ότι όταν συγκρίνεται με τα πρότυπα αναφοράς για press- release

Στην περίπτωση που δεν αναγνωρίζονται σωστά οι κινήσεις π.χ. ο αλγόριθμος βγάζει λανθασμένα αποτελέσματα ανάμεσα σε click και press, μπορούμε να επιλέξουμε το πρότυπο αυτό που έχει σχετικά μικρό κόστος όταν συγκρίνεται με κινήσεις ίδιου τύπου, αλλά όταν συγκρίνεται με διαφορετικού τύπου κινήσεις παράγει μεγαλύτερο κόστος.

7.2 Επιβεβαίωση προτύπων αναφοράς

Βάζοντας την επιλογή **TEMPLATE_TESTING** στην μεταβλητή APP_MODE του προγράμματος, μας δίνεται η δυνατότητα να επιβεβαιώσουμε την ορθότητα των προτύπων αναφοράς που επιλέξαμε χρησιμοποιώντας την διαδικασία που αναλύθηκε στην ενότητα 7.1. Ο χρήστης, φορώντας την συσκευή εισόδου στον δείκτη, πραγματοποιεί την μία χειρονομία μετά την άλλη και βλέπει σε πραγματικό χρόνο στην σειριακή να εκτυπώνεται ως τι αναγνωρίστηκε η κάθε χειρονομία του. Οι διάφορες τιμές που μπορούν να εκτυπωθούν είναι οι ακόλουθες:

- click, η χειρονομία αναγνωρίστηκε ως click
- press, η χειρονομία αναγνωρίστηκε ως press
- release , η χειρονομία αναγνωρίστηκε ως release
- toolarge, η χειρονομία απορρίφθηκε καθώς είχε μήκος μεγαλύτερο από 60 στοιχεία
- no, το κόστος της χειρονομίας ξεπερνούσε το UPPER_GST_COST που έχει δηλωθεί

Μετά το πέρας της διαδικασίας της επιλογής των προτύπων, τα επιλεγμένα πρότυπα πρέπει να αντικαταστήσουν τα υπάρχοντα στο αρχείο gst_template.h. Στο ίδιο αρχείο, πρέπει να ενημερωθούν και τα διάφορα μήκη των προτύπων, έτσι ώστε να αντιστοιχούν στα νέα πρότυπα.

Ακόμα, στην περίπτωση που τα πρότυπα που έχουμε επιλέξει έχουν διαφορετικό μέσο κόστος όταν συγκρίνονται με χειρονομίες του ίδιου είδους, θα ήταν καλό να πραγματοποιήσουμε δοκιμές με τα διάφορα UPPER_GST_COST, όπου GST=CLICK ή PRESS ή RELEASE, τα οποία βρίσκονται στο αρχείο gst_recon.h. Στην δική μας περίπτωση, το μέσο κόστος των click ήταν 100000 και το UPPER_CLICK_COST είχε οριστεί σε 130000. Σε κάθε περίπτωση, στην διαδικασία επιβεβαίωσης των προτύπων αναφοράς, πειραματιζόμενος με τα UPPER_GST_COST, μπορούμε να δούμε πότε ελαττώνονται σημαντικά τα αποτελέσματα NO στην αναγνώριση και αντικαθιστούνται με την αναγνώριση μιας χειρονομίας.

Στην Εικόνα 44 φαίνονται τα αποτελέσματα της διαδικασίας επιβεβαίωσης των προτύπων αναφοράς.

Εικόνα 44: Εκτύπωση στην σειριακή των χειρονομιών που εκτελεί ο χρήστης

7.3 Μέτρηση χρόνου εκτέλεσης του αλγορίθμου

Βάζοντας την επιλογή **TIME_MEASURE** στην μεταβλητή APP_MODE του προγράμματος, μας δίνεται η δυνατότητα να μετρήσουμε τον χρόνο εκτέλεσης για κάθε διαφορετικό κομμάτι του αλγορίθμου DTW. Η μέτρηση του χρόνου αυτού είναι πολύ σημαντική, καθώς πρέπει να βεβαιωθούμε πως δεν χάνουμε κάποιο data_ready_interrupt όσο χρόνο μένουμε στο interrupt_service_routine.

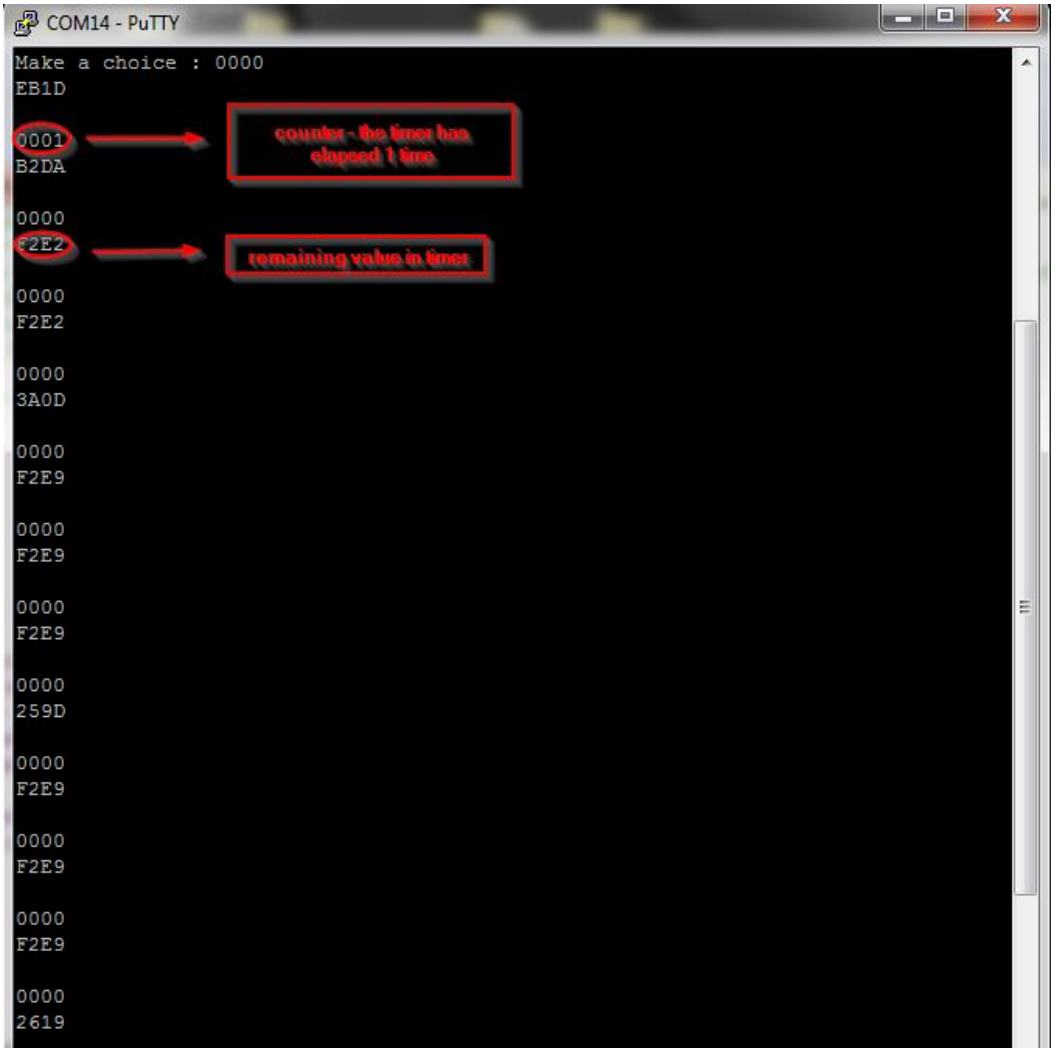
Για να μετρήσουμε τον χρόνο, χρησιμοποιούμε τον TIMER0 [12], τον οποίο ενεργοποιούμε χρησιμοποιώντας την συνάρτηση void enableTimer (void). Ο συντελεστής διαίρεσης του 16MHz ρολογιού ορίζεται να είναι ίσος με ένα, και η τιμή που φορτώνεται στον χρονομετρητή είναι ίση με 65000. Κάθε φορά που λήγει ο χρονομετρητής (time_counter), πυροδοτείται ένα interrupt το οποίο αυξάνει την τιμή ενός μετρητή (counter). Η τιμή του μετρητή αυτού όπως επίσης και η εναπομένουσα τιμή του χρονομετρητή εκτυπώνονται στην σειριακή κάθε φορά που βγαίνουμε από το interrupt service routine, όπως παρουσιάζεται και στην Εικόνα 45.

Καθώς αρχικά είναι ενεργοποιημένα μόνο τα fifo_overflow και motion_interrupt, δεν εκτυπώνεται τίποτα στην σειριακή. Μόλις ο χρήστης πυροδοτήσει motion_interrupt, τότε θα ενεργοποιηθούν τα data_ready_interrupt και θα μπορούμε να μετράμε τον χρόνο που παίρνει στο finite state machine της αναγνώρισης χειρονομιών να εκτελέσει κάθε φορά τον κώδικά του.

Όπως αναφέραμε στην ενότητα 6.5, η πρώτη δυάδα μετρήσεων που εμφανίζεται στην Εικόνα 39 είναι ο χρόνος που περνάει από την στιγμή που πυροδοτείται το motion_interrupt (το σύστημά μας βρίσκεται στην κατάσταση IDLE_GST, και μεταβαίνει στην κατάσταση START_GST αφού πρώτα απενεργοποιήσει τα motion_interrupt και ενεργοποιήσει το data_ready_interrupt) έως ότου βγει από το interrupt service routine. Ο χρόνος που περνάει για να τα κάνει όλα αυτά υπολογίζεται ως εξής:
$\frac{(counter\ +1)\times 65000 - time\_counter}{16000000}$, δηλαδή

$$\frac{(0+1)\times 65000 - 60189}{16000000} = 0{,}0003sec$$

Η δεύτερη δυάδα αριθμών, μετράει τον χρόνο της πιο χρονοβόρας κατάστασης του finite state machine, αυτής όπου παίρνουμε τα έξι τελευταία στοιχεία από την ουρά (τα έξι πρώτα στοιχεία της χειρονομίας) και εκτελούμε τον αλγόριθμο DTW για αυτά με κάθε ένα από τα πρότυπα αναφοράς.



Εικόνα 45: Εκτύπωση στην σειριακή των χρόνων που περνάει ο αλγόριθμος για την διαδικασία αναγνώρισης χειρονομιών κάθε φορά που έχουμε ένα interrupt για μία χειρονομία

Ομοίως, όλοι οι υπόλοιποι αριθμοί αντιστοιχούν στον χρόνο που διαρκεί η εκτέλεση του ISR για κάθε data_ready_interrupt,έως ότου τελειώσει η διαδικασία αναγνώρισης.

7.3.1 Ενδιάμεσος χρόνος από το τέλος μιας χειρονομίας μέχρι την αρχή της επόμενης

Η αρχή της διαδικασίας αναγνώρισης πραγματοποιείται μόνο στην κατάσταση IDLE_GST με την πυροδότηση ενός motion_interrupt.

Όπως αναλύσαμε στο κεφάλαιο 6.5, το τέλος μιας χειρονομίας εντοπίζεται στην κατάσταση LOG_GST, κάθε φορά που η υπολογιζόμενη διακύμανση για τέσσερα συνεχόμενα παράθυρα της χειρονομίας είναι μικρότερη από ένα κατώφλι. Συνεπώς, για να βρούμε τον ενδιάμεσο χρόνο ανάμεσα σε δύο χειρονομίες πρέπει να υπολογίσουμε το χρονικό διάστημα

που μεσολαβεί από την στιγμή που είμαστε στην κατάσταση LOG_GST έως ότου μεταβούμε στην κατάσταση IDLE_GST.

Ενώ βρισκόμαστε στην κατάσταση LOG_GST, μετά το τέλος μιας χειρονομίας, η κατάσταση αλλάζει σε END_GST. Αυτό, συμβαίνει σε λιγότερο από 10msec που μεσολαβούν από ένα data_ready_interrupt σε ένα άλλο. Μόλις χτυπήσει το πρώτο data_ready_interrupt (σε λιγότερο από 10 msec), το σύστημα θα αναγνωρίσει την κίνηση και θα εκτελέσει τον κώδικα που υπάρχει στην κατάσταση END_GST. Ο κώδικας αυτός θα αλλάξει την κατάσταση σε PREPARE_NEXT_GST, ο κώδικας της οποίας θα εκτελεστεί στο επόμενο data_ready_interrupt και το σύστημα μας θα είναι έτοιμο να δεχθεί νέες χειρονομίες για αναγνώριση. Συνεπώς, όπως βλέπουμε, ο ενδιάμεσος χρόνος ανάμεσα σε δύο χειρονομίες είναι λιγότερος από 20msec.

Συμπεράσματα

Η εργασία αυτή είχε ως στόχο της να αποδείξει πως είναι εφικτή η χρησιμοποίηση αλγορίθμων αναγνώρισης προτύπων σε μικροεπεξεργαστές με περιορισμένο μέγεθος κώδικα, υπολογιστική ισχύ και μνήμη. Αποδείξαμε πως υπάρχουν τεχνικές που μπορούν να χρησιμοποιηθούν όπως και διάφορες αποφάσεις που μπορεί να ληφθούν έτσι ώστε να εφαρμοστεί μια απλοποιημένη έκδοση του επιλεγμένου αλγορίθμου η οποία θα ανταποκρίνεται στις ανάγκες της κάθε συσκευής.

Τέτοιες τεχνικές ήταν η αποφυγή χρήσης μεταβλητών τύπου float και double όπως επίσης και πράξεων που έχουν τέτοιες μεταβλητές ως αποτέλεσμα καθώς δεν διαθέτουμε Floating Point Unit. Για τον λόγο αυτό πρέπει από την αρχή να εντοπίσουμε τους περιορισμούς του συστήματός μας και να ξεκινήσουμε να προσαρμόζουμε τον αλγόριθμο σε κάθε απόφαση που θα κληθούμε να λάβουμε με βάση τους περιορισμούς αυτούς.

Παρόλο που το σύστημα διαθέτει ορισμένους περιορισμούς, όπως το ότι είναι εξατομικευμένο και ότι το χέρι στην αρχή κάθε χειρονομίας πρέπει να είναι παράλληλα με το έδαφος, θεωρούμε πως υπάρχουν τρόποι να ξεπεραστούν ακολουθώντας τις τεχνικές που αναλύονται στην ενότητα που ακολουθεί.

Εν κατακλείδι, αποδείξαμε ότι υπάρχει τρόπος να υλοποιηθούν αλγόριθμοι αναγνώρισης προτύπων σε τέτοιες συσκευές, ανοίγοντας έτσι τον δρόμο για μια νέα γενιά συσκευών που θα παρέχουν περισσότερες δυνατότητες στους χρήστες και σκοπεύουμε στο μέλλον να μελετήσουμε και άλλες τεχνικές που θα έχουν παρόμοια αποτελέσματα και δεν εφαρμόστηκαν στην παρούσα εργασία.

Μελλοντική έρευνα- πιθανές βελτιώσεις

Το σύστημα αναγνώρισης προτύπων που προτείνεται στην παρούσα μεταπτυχιακή διατριβή είναι ικανοποιητικό για την χρήση για την οποία προορίζεται, χωρίς όμως αυτό να σημαίνει πως δεν μπορεί να βελτιωθεί. Ενώ μπορεί να αναγνωρίζει γρήγορα τις διάφορες χειρονομίες του χρήστη, διαθέτει ορισμένους περιορισμούς οι οποίοι αναφέρθηκαν στην ενότητα 6.6.

Αρχικά, επιλέχθηκε για να χρησιμοποιηθεί ως είσοδος στον αλγόριθμο αναγνώρισης προτύπων ένας μόνο από τους τρεις διατιθέμενους άξονες, γεγονός που μείωσε τις υπολογιστικές πράξεις, τον χρόνο εκτέλεσης του αλγορίθμου όπως επίσης και την χρησιμοποιούμενη μνήμη. Το γεγονός αυτό ήταν αποδεκτό εξαιτίας της φύσης των χειρονομιών που θέλαμε να αναγνωρίσουμε (κίνηση σε μία κατεύθυνση). Στην περίπτωση όμως που θέλουμε να αναγνωρίσουμε περισσότερες κινήσεις σε περισσότερες κατευθύνσεις, θα πρέπει να χρησιμοποιήσουμε περισσότερους άξονες ως διάνυσμα εισόδου στον αλγόριθμο και να εντοπίσουμε άλλα μέτρα μέτρησης του κόστους d που θα απαιτούν λιγότερες υπολογιστικές πράξεις και λιγότερη μνήμη.

Επίσης, θα μπορούσαμε να χρησιμοποιήσουμε την DigitalMotionProcessing μονάδα της μονάδας μέτρησης αδράνειας, η οποία θα παρείχε την δυνατότητα να γίνει όλη η επεξεργασία της κίνησης μέσα στον DigitalMotionProcessor (DMP), αφήνοντας τον επεξεργαστή του συστήματος ελεύθερος να ασχοληθεί με άλλες εργασίες και όχι με τις πολύπλοκες αριθμητικές πράξεις που απαιτούνται για την αναγνώριση των κινήσεων. Επιπροσθέτως, η μονάδα αυτή μπορεί να αφαιρέσει το διάνυσμα της βαρύτητας από την επιτάχυνση, και συνεπώς δεν θα είμαστε πια εξαρτημένοι από τον προσανατολισμό της συσκευής καθώς θα επιτυγχάνουμε καλύτερα αποτελέσματα στην αναγνώριση των χειρονομιών.

Ακόμα, θα μπορούσαμε να πειραματιστούμε με την δειγματοληψία, καθώς θεωρούμε πως ένας μικρότερος ρυθμός δειγματοληψίας από αυτόν που έχουμε (ένα δείγμα ανά 10msec), θα ήταν ικανοποιητικός για την αναγνώριση των κινήσεων. Με την μείωση του ρυθμού δειγματοληψίας, θα είχαμε λιγότερα δείγματα δεδομένα να επεξεργαστούμε (μικρότερα πρότυπα αναφοράς καθώς επίσης και υπό εξέταση πρότυπα). Ακόμα, θα είχαμε περισσότερο χρόνο για την εκτέλεση του πιο χρονοβόρου τμήματος του αλγορίθμου DTW, το οποίο συνεπάγεται ότι θα είχαμε χρόνο να συγκρίνουμε την υπό εξέταση χειρονομία με περισσότερα πρότυπα αναφοράς, αρκεί να μας το επέτρεπε η μνήμη.

Τέλος, το σύστημά αναγνώρισης χειρονομιών είναι εξατομικευμένο, καθώς τα πρότυπα αναφοράς έχουν ληφθεί από έναν μόνο χρήστη και αντικατοπτρίζουν τις κινήσεις του. Για την λήψη και την επιλογή εξατομικευμένων προτύπων για κάθε χρήστη θα μπορούσε να αναπτυχθεί μια android εφαρμογή η οποία θα μπορεί να επιλέγει το πρότυπο αναφοράς για κάθε μία από τις χειρονομίες και χρησιμοποιώντας το SoftwarePatchOverTheAir (SPOTAR) module του αναπτυξιακού θα μπορεί να αποθηκεύει τα νέα πρότυπα στο τσιπ.

Βιβλιογραφία

[1] *http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf*

[2] *http://www.dialog-semiconductor.com/docs/site-pdf/da14580_ds_v3-1.pdf?sfvrsn=2*

[3] *UM-B-006_DA14580_Sleep_mode_configuration.pdf*

[4] *Programming-By-Example Gesture Recognition Kevin Gabayan, Steven Lansel ,2006*

[5] *An Inertial Measurement Framework for Gesture Recognition and Applications Ari Y. Benbasat and Joseph A. Paradiso*

[6] *MobiRing: A Finger-Worn Wireless Motion Tracker,Yi-Lin Chen_, Yi-Lung Tsai_, Kailing Huang, Pai H. Chou*

[7] *Intuitive Interface device for Wearable Computers Jong-WoonYoo, Woo-Min Hwang, Sung-HoonBaek, and Kyu-Ho Park*

[8] *uWave: Accelerometer-based Personalized GestureRecognition and Its ApplicationsJiayang Liu, Zhen Wang, and Lin Zhong*

[9] *An Accelerometer-Based Gesture Recognition Algorithm and its Application for 3D InteractionJianfeng Liu, Zhigeng Pan, and Xiangcheng Li*

[10] *Muller, M., Information Retrieval for Music and Motion*

[11]*https://www.olimex.com/Products/Modules/Sensors/MOD-MPU6050/resources/RM-MPU-60xxA_rev_4.pdf*

[12]*UM-B-004_DA14580_Peripheral_Drivers.pdf*

[13] *Ανάπτυξη συσκευής εισόδου με χρήση αισθητηρίων αδράνειας, Τζανιδάκης Μάριος*

[14] *An introduction to Pattern Recognition: A Matlab approach,Sergios Theodoridis, Konstantinos Koutroubas*

[15] *Pattern Recognition 4th Edition, Sergios Theodoridis, Konstantinos Koutroubas*

[16] *https://www.abiresearch.com/press/the-internet-of-things-will-drive-wireless-connect/*

Παράρτημα Α' : User Manual

1. Introduction

This document describes the Gesture Ring application based on the DA14580. The Gesture Ring application runs on a ring shaped hardware which is equipped with a GY-88 IMU sensor. The schematic and the connections to the Dialogs Expert Development Kit are also provided in chapter 2.

The User Manual is provided with two .rar files gesture_ring.rar and SDK_v_3.0.2.0_Gesture_recon.rar. The first .rar file contains the source code for the Gesture Ring mouse and the latter contains an auxiliary code for template isolation and testing.

The auxiliary source code is used for isolating and testing the gestures that will be used as templates, and provides the amount of time that the program spends executing the gesture recognition algorithm. The functions and explanation of the Gesture Ring mouse source code is given in chapter 3 and the examples and settings for the auxiliary code are provided in chapter 4.

The source code for the application, explained in chapter 3, is provided in the gesture_ring.rar file. Is based on the DA1458x_SDK_3.0.6 Software Development Kit. The developer should extract the .rar file and place the extracted contents (three folders) in the directories explained below:

- The gesture_ring folder is the application source code. This folder should be placed in the "\DA1458x_SDK_3.0.6\DA1458x_SDK_3.0.6\dk_apps\keil_projects\" directory.
- The i2c folder is a modified version i2c_eeprom driver. This folder should be placed in the "\DA1458x_SDK_3.0.6\DA1458x_SDK_3.0.6\dk_apps\src\plf\refip\src\driver" directory.
- The mpu_6050 folder includes the driver for the sensor and also should be placed in the "DA1458x_SDK_3.0.6\DA1458x_SDK_3.0.6\dk_apps\src\plf\refip\src\driver" directory.

The user, after placing the files in previously mentioned directories, can open the \DA1458x_SDK_3.0.6\DA1458x_SDK_3.0.6\dk_apps\keil_projects\ gesture_ring\gesture_ring\ gesture_ring.uvproj file using keil compiler. Note: The **program runs in O2 optimization**.

The SDK_v_3.0.2.0_Gesture_recon.rar file, explained in chapter 4, includes the auxiliary code for gesture isolation and testing. Is based on the SDK_v_3.0.2.0 software development kit. The user should extract the .rar file and open the SDK_v_3.0.2.0_Gesture_recon\ plain_gesture_recon_alg \peripheral_examples\DA14580_peripheral_setup.uvproj file using keil compiler.

2. Hardware Overview

The application for the acquisition of inertia data uses the GY-88 chip which consists from three different sensors:

- HMC5883L – Digital pressure sensor
- BMP085 – 3Axis Digital Compass
- MPU6050 – Accelerometer and Gyroscope

The application only uses the MPU6050 to acquire acceleration and angular velocity readings. The sensor is attached on a copper board where the I2C circuit and the battery are located. Figure 1 shows the schematic and the connections with the Dialogs Expert Development Kit. The J2 header shows where the GY-88 is placed.

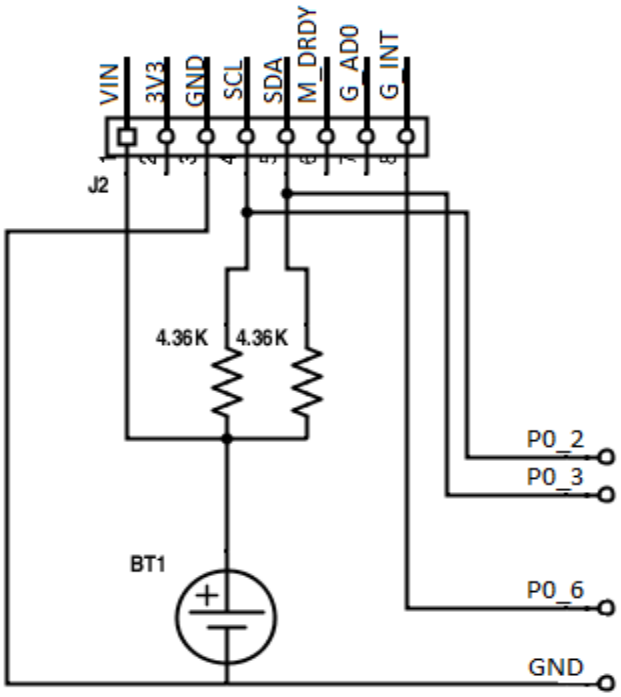


Figure 1

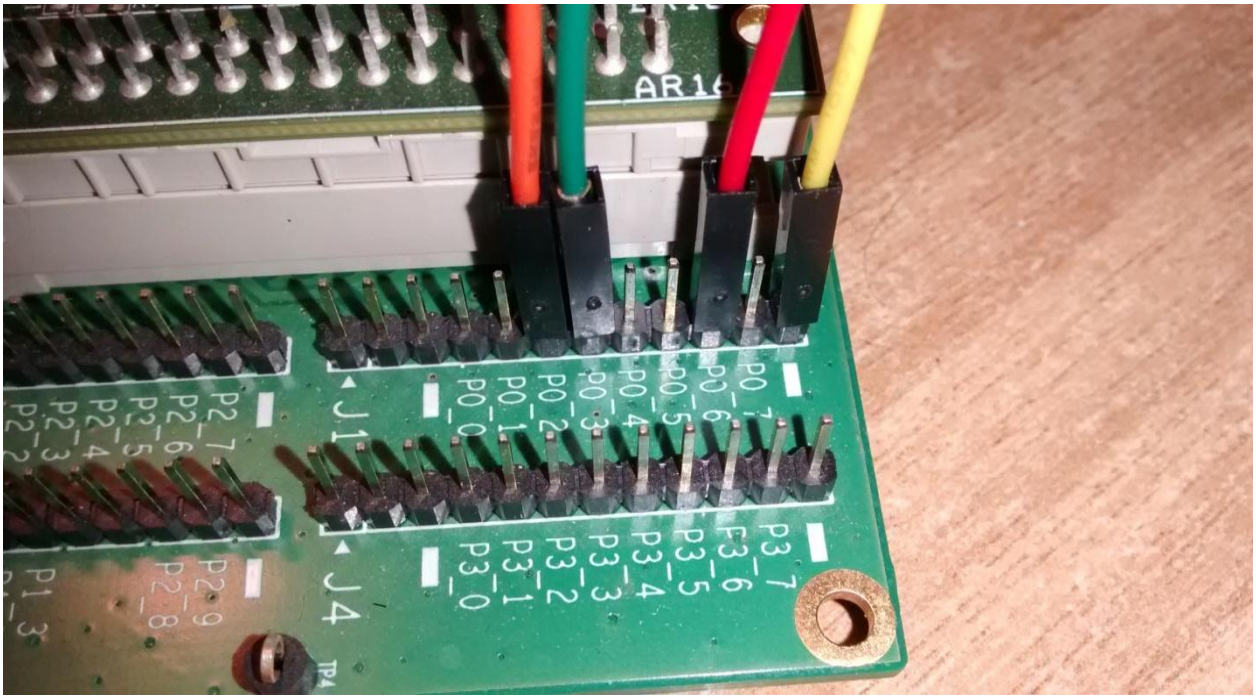


Figure 2

3. Application Source code structure

In this chapter the Gesture Ring mouse source code is explained. All the functions below and the .uvproj file can be found in \DA1458x_SDK_3.0.6\DA1458x_SDK_3.0.6\dk_apps\keil_projects\gesture_ring\gesture_ring directory.

1. gst_ring_config.h

This file contains all the possible configurations that are available to the programmer.

Sensor configurations

<table>
<tr><td>ACCEL_FS_SEL</td><td>selects the full scale range of the gyroscope outputs<table><tr><td>AFS_SEL</td><td>Full scale range</td></tr><tr><td>0</td><td>±2 g</td></tr><tr><td>1</td><td>±4 g</td></tr><tr><td>2</td><td>±8 g</td></tr><tr><td>3</td><td>±16 g</td></tr></table></td></tr>
<tr><td>GYRO_FS_SEL</td><td>selects the full scale range of the gyroscope outputs<table><tr><td>FS_SEL</td><td>Full scale range</td></tr><tr><td>0</td><td>±250 °/sec</td></tr><tr><td>1</td><td>±500 °/sec</td></tr><tr><td>2</td><td>±1000 °/sec</td></tr><tr><td>3</td><td>±2000 °/sec</td></tr></table></td></tr>
<tr><td>SENSOR_DLPF</td><td>configures the digital low pass filter for accelerometer and gyroscope</td></tr>
<tr><td>ACCEL_DHPF</td><td>configures the accelerometer digital high pass filter</td></tr>
<tr><td>SMRT_DIV</td><td>configures the sampling rate
Sample Rate = Accelerometer Output Rate / (1 + SMPRT_DIV),
Accelerometer Output Rate=1kHz</td></tr>
<tr><td>MOT_THR_SET</td><td>motion interrupt threshold</td></tr>
<tr><td>MOT_DUR_SET</td><td>Number of values that must exceed the motion threshold in order to trigger the interrupt</td></tr>
<tr><td>MOT_DECR_RATE</td><td>Motion detection counter decrement rate set in 1</td></tr>
</table>

Timers configurations

INACTIVITY_ TIMER_ QUOTA	Inactivity time (in multiples of 10msec) before the device goes into permanent sleep
INACTIVITY_ TIMER_COUNTER_ MAX	Time in multiples of INACTIVITY_TIMER_QUOTA
INACTIVITY_ COUNTER_MAX	The time that we will stay in active mode after the last movement in multiples of 10msec
MOVE_INTERVAL	Sampling time for gyroscope

Mouse movement configurations

MOVE_OFFSET	The difference a current gyroscope reading must have from the previous reading in order to

	consider that the mouse has moved
GYRO_WINDOW	The length of the buffer that keeps successive values from the gyroscope

FIFO and Gesture Recognition configurations

DUMMY_READS_MAX	The number of data that has to be in fifo in order to empty it
FIFO_GST_BEF	Datasets in fifo before the read burst
DATA_LEFT_IN_FIFO	Datasets that are left in fifo after every read burst *The values that enter the fifo after the motion interrupt has triggered are part of the gesture, but usually the gesture begins 4-6 datasets before the motion interrupt is triggered. That's why we always keep the last 6 elements of the fifo when we perform burst reads in order to empty it(the fifo)
MEAN_WINDOW	Mean window size in datasets *In order to find the end of the gesture we need to check the variance of MEAN_WINDOW_SIZE samples
MEAN_WIN_THR	Threshold used for isolating the gesture
MAX_GST_LEN_INIT_VAL	Initial value of measurements when starting the gesture recognition algorithm
MAX_GST_SIZE	The max length a valid gesture can have
VARIANCE_COUNT_UNDER_THR	Count the variance values under the threshold that indicate the end of a gesture

2. app_rng_fsm.c

Function name	**start_adv_directed**
inputs	void
Return value	void
description	Starts the process of directed advertising

Function name	**app_state_update**
inputs	Evt: an event that will cause a state change
Return value	void
description	Function that handles the main finite state machine of Gesture Ring

Function name	**app_rng_init_vars**
inputs	void
Return value	void
description	Reinitializes the variables used in the FSM

3. gesture_recon.c

Function name	**interrupt_events**
inputs	void
Return value	gst_recon_events: the type of interrupt that was triggered
description	This function executes every time an interrupt occurs and returns to us the interrupt that was triggered

Function name	**gestureRecon**
inputs	Evt:the interrupt that was triggered
Return value	void
description	This function is the gesture recognition finite state machine

Function name	**variance**
inputs	Buffer: array containing the 4 elements whose variance we want to compute
Return value	The computed variance
description	Function that computes the variance for the z axis of 4 values

Function name	**DTWSakoeImp**
inputs	input: array containing the samples that we want to compare to the templates using DTW input_size: the number of samples we want to compare calc_first_row: boolean expressing whether it's the first row that we give as input or not
Return value	void
description	Function that computes the dynamic time warping cost of the provided test template samples when compared to the 3 templates

Function name	**min**
inputs	a: the cost generated from DTW while comparing the unknown gesture to click template b: the cost generated from DTW while comparing the unknown gesture to press template c: the cost generated from DTW while comparing the unknown gesture to release template
Return value	void
description	This function decides what template is more similar to our unknown gesture and if it is considered valid gesture (the cost is below a threshold) adds this gesture into the gesture buffer

4. gst_templates.h

This file contains the templates that we use in order to identify the unknown gesture. In case that we change one of these templates, we have to change the size of it so that it corresponds to the new template size(e.g. RELEASE_SIZE_TEMPL).

5. app_reports.c

Function name	**send_mouse_report**
inputs	void
Return value	1 if everything went alright, 0 otherwise
description	Function that will update the attribute db with the new values that we want to send

Function name	**ring_process_gesture**
inputs	Gst: the gesture that was performed
Return value	1 if everything went alright, 0 otherwise
description	Function that will take the gesture that was performed, create a report and push it to kbd_trm_list

Function name	**ring_proccess_motion_8**
inputs	move: the movement of the cursor
Return value	1 if everything went alright, 0 otherwise
description	Function that will take the movement of the cursor, create a report and push it to kbd_trm_list

Function name	**app_gst_prepare_reports**
inputs	void
Return value	1 if everything went alright, 0 otherwise
description	Function that will take each item in the gst_buffer and create an appropriate report for each one of the gestures in order to be sent

6. mpu6050_func.c

Function name	**app_initialize_sensor**
inputs	void
Return value	void
description	Function that will initialize the sensor when the app will start

Function name	**app_disable_sensor**
inputs	void
Return value	void
description	Function that will disable the fifo and set

	only the motion interrupt active

Function name	**app_ reinit_sensor**
inputs	void
Return value	void
description	Function that will re-initialize the sensor after we wake up and re-connect to the host

Function name	**severFIFOData**
inputs	Buffer_dst: the destination buffer the data will be put in Buffer_src: the source buffer that contains the data Size: the size of the buffer in bytes Data_sel: select whether accelerometer or gyroscope data
Return value	void
description	Function that will take the FIFO data and isolate only accelerometer or only gyroscope data according to data_sel value

Function name	**mean_value**
inputs	Buffer: the buffer that contains the values we want to find the mean value of
Return value	The mean value
description	Function that will find the mean value of the elements of the given buffer

Function name	**getGyroData**
inputs	void
Return value	The gyroscope values
description	Function that will read the gyroscope values from the register and return the higher 8-bits of the gyroscope value *since the gyroscope values are 16-bit and we need to map them to an 8-bit number, we perform this mapping by using only the 8-high bits of the value

6. mpu6050.c

Function name	**setClockSource**
inputs	Source: the clock source of the sensor
Return value	void
description	Function that sets the clock source of the sensor

Function name	**sensorReset**

inputs	void
Return value	void
description	Function that resets the sensor

Function name	**setSensorSleepMode**
inputs	void
Return value	void
description	Function that configures the power mode

Function name	**setDLPFMode**
inputs	Mode: the digital low pass filter mode
Return value	void
description	Function that configures the digital low pass filter

Function name	**setRate**
inputs	Rate: the desired sampling rate
Return value	void
description	Function that configures the rate used to put the data into fifo

Function name	**setGyroFullRange**
inputs	Mode: selects the full scale range of the Gyroscope
Return value	void
description	Function that configures the full scale range of the gyroscope sensor

Function name	**setAccelFullRange**
inputs	Mode: selects the full scale range of the Accelerometer
Return value	void
description	Function that configures the full scale range of the accelerometer sensor

Function name	**setMotionDetectionThreshold**
inputs	Threshold:the desired threshold used to trigger motion interrupt
Return value	void
description	Function that configures the threshold used in order to trigger the motion interrupt

Function name	**setMotionDetectionDuration**
inputs	duration: the desired duration used to trigger motion interrupt
Return value	void

description	Function that configures the number of values that must exceed the motion threshold in order to trigger the interrupt

Function name	**setMotionDetectionCountDecrement**
inputs	decrement: the desired motion detection counter decrement rate
Return value	void
description	Function that configures the Motion detection counter decrement rate

Function name	**setAccelDHPFMode**
inputs	mode: the desired mode for digital high pass filter
Return value	void
description	Function that configures the digital high pass filter of the accelerometer

Function name	**setInterruptConfiguration**
inputs	Int_cfg: the enabled interrupt configuration
Return value	void
description	Function that configures interrupts that are enabled

Function name	**setEnabledInterrupts**
inputs	Int_en: the interrupts we want to enable
Return value	void
description	Function that enables the desired interrupts

Function name	**setFIFOData**
inputs	Fifo_data: the data that we want stored in the fifo
Return value	void
description	Function that configures what data (accel or gyro) will be stored in the fifo

Function name	**enableFIFO**
inputs	void
Return value	void
description	Function that enables the fifo in order to store data

Function name	**disableFIFO**
inputs	void
Return value	void
description	Function that disables the fifo

Function name	**countFIFO**
inputs	void
Return value	The total bytes of the datasets stored in fifo
description	Function that returns the total bytes of the datasets stored in fifo

Function name	**resetFIFO**
inputs	void
Return value	void
description	Function that resets the fifo

Function name	**readFIFO**
inputs	Buffer: the buffer that will hold the datasets we read from FIFO Size: the number of bytes we want to read from fifo
Return value	void
description	Function that reads as many bytes we want from the fifo and stores them in the provided buffer

7. app_sleep.h

Function name	**app_async_trm**
inputs	void
Return value	true, to force calling of scedule false, else
description	In app_async_trm function, we prepare the reports that we want to send. Also, we set APP_RING_REPORT_SYNCHRONIZATION_TIMER, a timer that is used to synchronize the cursor reports and also to keep the device awake.

Function name	**app_async_proc**
inputs	void
Return value	true, to force calling of scedule false, else
description	In app_async_proc function, every time we have a mouse_report_synchronization_event_triggered event, we poll the gyroscope and put the values in a buffer that can contain 8 elements. We do this because as we stated earlier, by the moment the motion interrupt triggers, 4-6 samples (the start of the gesture) have already occured. If these values are polled just before the motion interrupt triggers, then there is a high chance that a mouse cursor move report will be sent moving the cursor away from the actual point where the user wants to perform the gesture. So, every time we poll the

	gyroscope, the values read will be written in the next empty space of the buffer. At the moment all 8 spots are full, we will take the average of the 4 older values and compare the produced average with the previous cursor movement we sent earlier. If there is a difference between them, we consider that the cursor has moved and we store the cursor movement in trm_list in order to be sent. As it is expected, if a motion interrupt occurs, the contents of this buffer are cleared.

Function name	**app_async_sleep_proc**
inputs	void
Return value	void
description	In app_async_sleep_proc function, we check the pin to make sure we didn't miss any interrupt. Also, in case we are not in gesture recognition mode, we empty all the data in the FIFO except the last 6, so that it doesn't get overflow.

Function name	**app_sleep_prepare_proc**
inputs	sleep_mode: the sleep mode rwip_sleep returned
Return value	void
description	In app_sleep_prepare_proc function, we check the pin to make sure we didn't miss any interrupt. Also, if the device is in active mode, we prevent it from going to sleep. Finally, if going to sleep is inevitable, we set the wakeup interrupt service routine.

Function name	**app_sleep_entry_proc**
inputs	sleep_mode: the sleep mode rwip_sleep returned
Return value	void
description	in app_sleep_entry_proc function, we check one last time the pin to make sure we don't miss any interrupt, and use lower clocks.

4. Auxiliary Source code Structure

In this chapter the auxiliary source code is explained the .uvproj file should be located in SDK_v_3.0.2.0_Gesture_recon\plain_gesture_recon_alg\peripheral_examples\DA14580_peripheral_setup.uvproj. The specific program is used in order to acquire and test the templates that are used as reference to recognize the gestures that the user performs. The program is based on dialog's peripheral_examples source code. The main section of the program is located in the DA14580_examples.c. The code initializes the IMU sensor, sets and enables the motion interrupt and enters a while loop where it keeps polling an interrupt variable and then empties

the FIFO of the sensor in order to avoid overflow. If a motion interrupt occurs then the motion interrupt is deactivated and the program enters the gesture recognition mode. As soon as the gesture recognition finishes, the program reactivates the motion interrupt and waits for the next gesture to occur.

4.1 Program Modes

An additional choice is added in the HARDWARE_CONFIGURATION_INDEX definition in periph_setup.h file. By setting the HARDWARE_CONFIGURATION_INDEX to 4 the TRYOUT_MPU6050_EXAMPLE is defined in order to make the appropriate settings for the MPU6050 sensor. The user then can set the APP_MODE definition to one of the three different modes:

- TIME_MEASURE: prints the execution time of the gesture recognition algorithm.
- TEMPLATE_SELECTION: prints accelerometer values after a completed gesture.
- TEMPLATE_TESTING: prints recognized gestures to test the accuracy of the templates.

An extra option is added “Gesture Recognition Example” which is triggered by pressing “i” in the example menu on the console (Figure 3). Depending on the value of APP_MODE definition the program responds with the corresponding outputs.

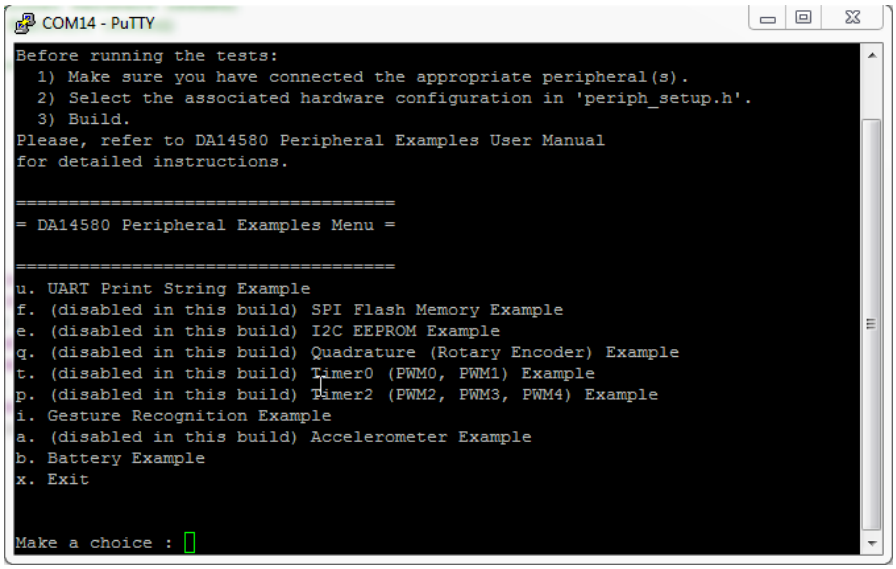


Figure 3

1. In the **TIME_MEASURE** option the program uses the timer in order to count how long the program executes the gesture recognition algorithm every time an interrupt occurs (dataready or motion interrupt).

The program (void enableTimer(void)) enables TIMER0, sets the clock of the timer with division factor 1 (16MHz) and loads the value 65000. When the timer elapses the interrupt is triggered and a counter increases (void user_callback_function_test(void)). The value of the counter and the remaining value in the timer are printed every time we exit the ISR in order to calculate the time period the program spent in the gesture recognition algorithm. The output of the program is shown in Figure 4.

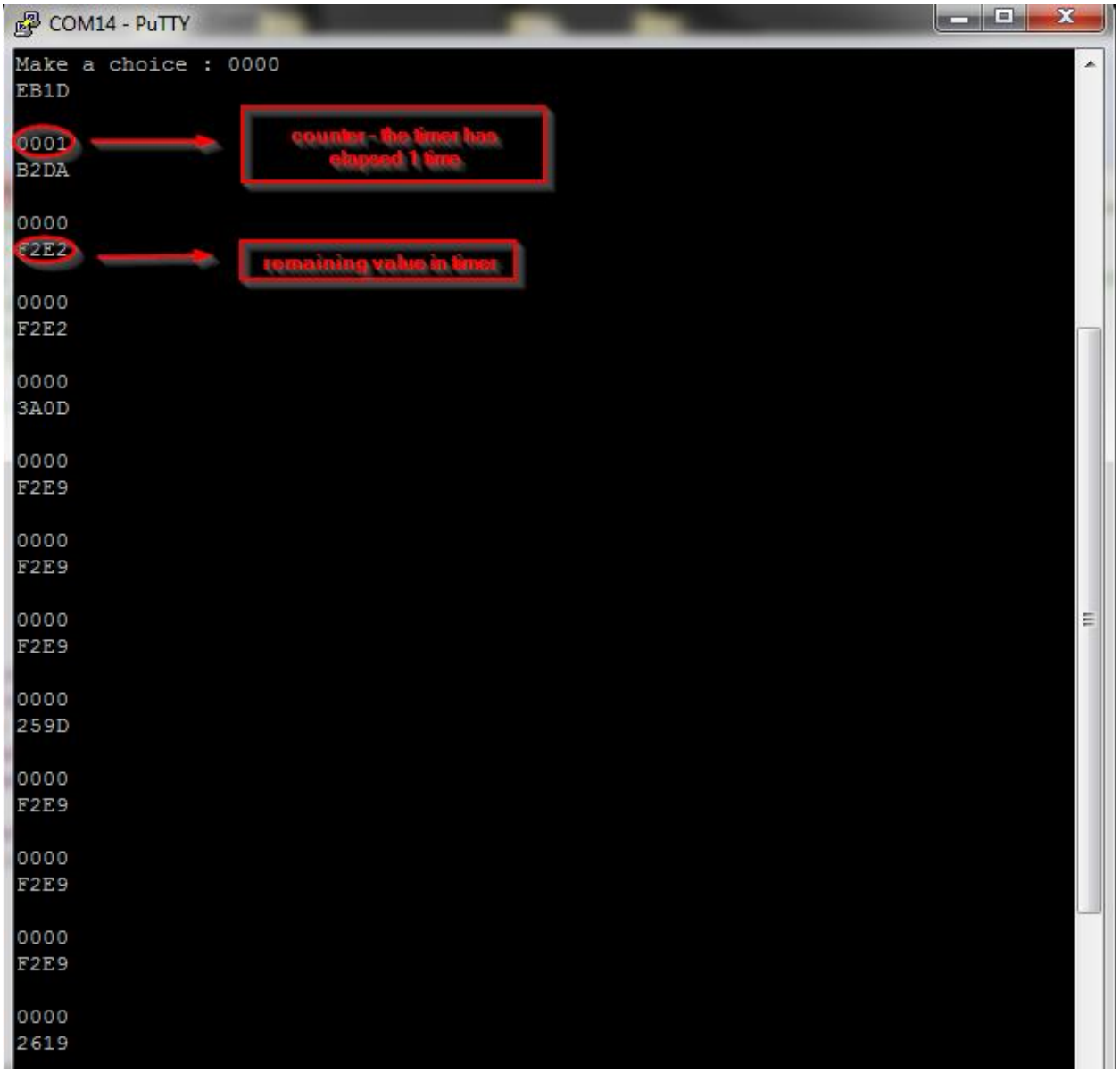


Figure 4

2. In the **TEMPLATE_SELECTION** option an additional buffer (print_buffer[60]) is added to the program. All the accelerometer values that the program reads from the sensor (while a gesture is detected) are copied in this buffer. As soon as the program detects the end of a gesture it prints the buffer (Figure 5). The values from the sensor appear in columns in x – y – z order.

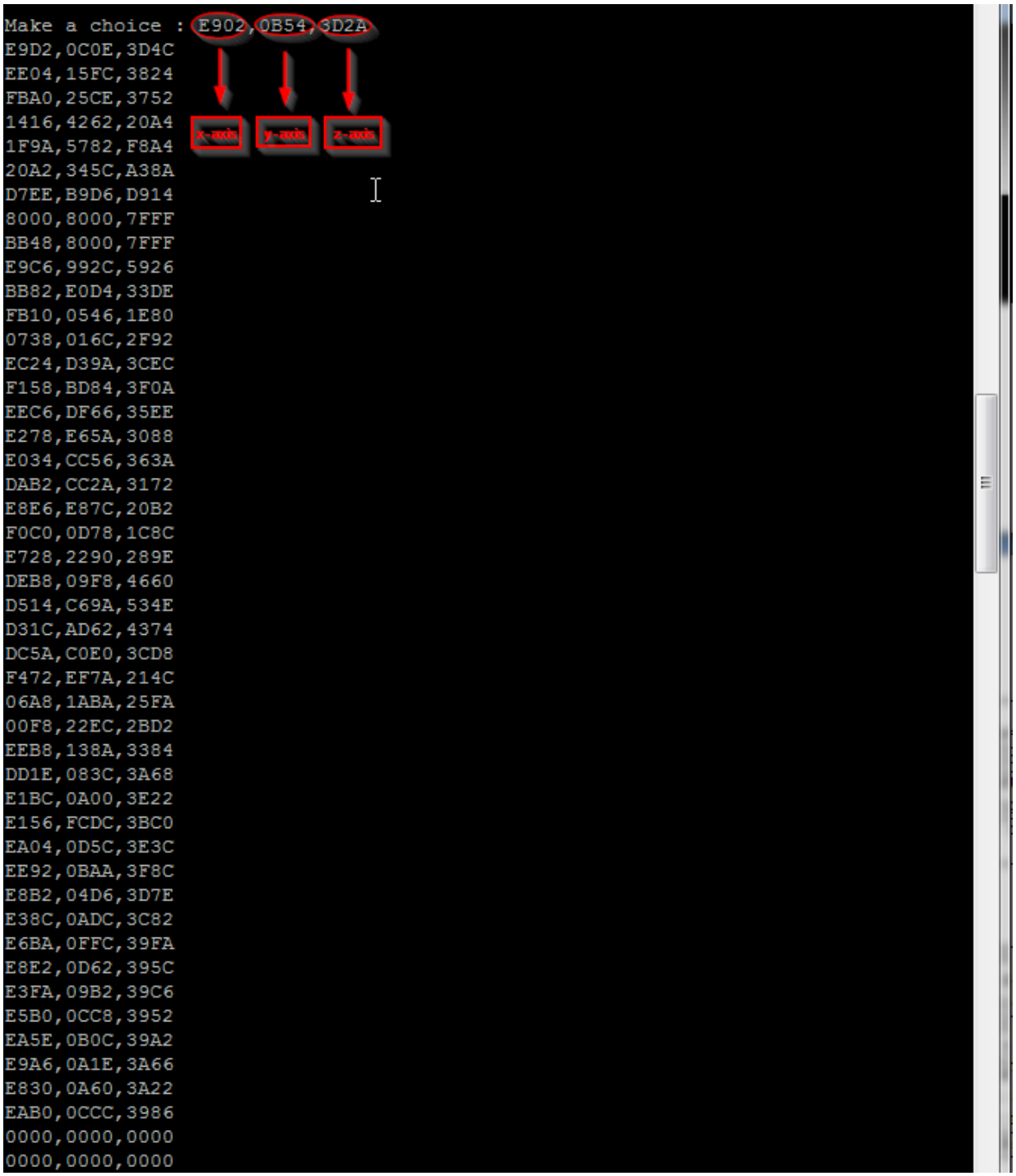
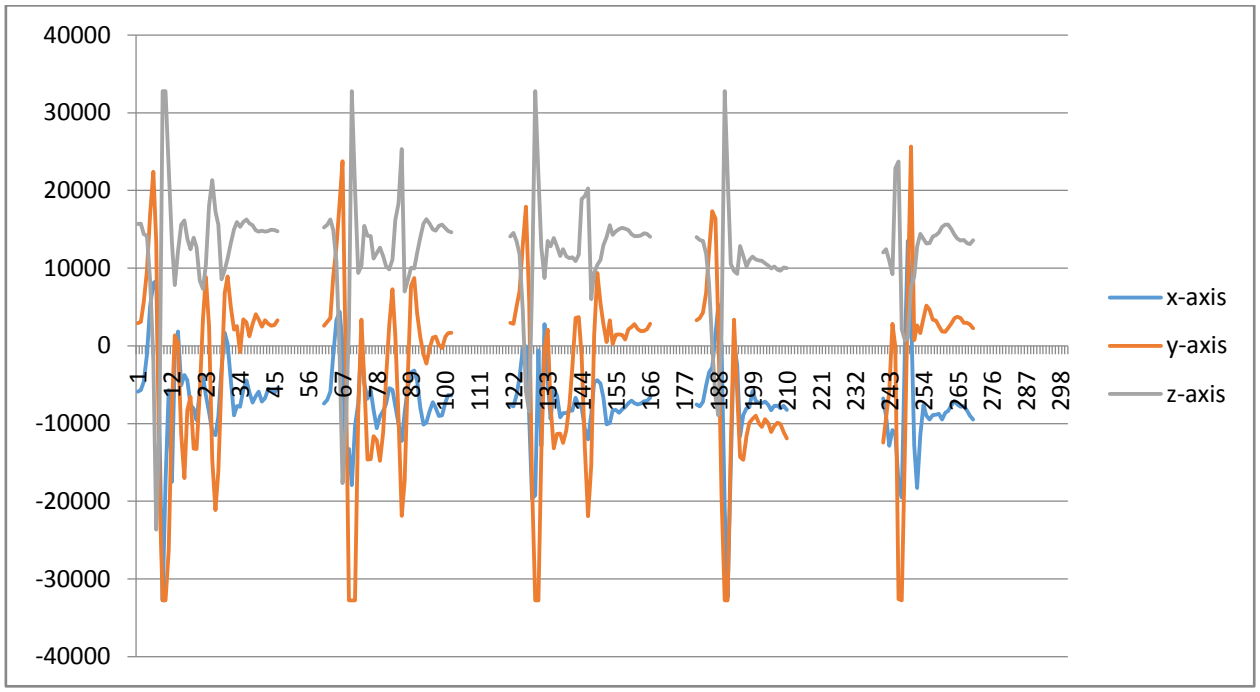


Figure 5

The user can visualize the data using an excel program. The image below shows 3 clicks, 1 press and 1 release performed.



3. In the **TEMPLATE_TESTING** option the user can perform gestures and test if the gesture templates are sufficiently recognized by the program. The user templates are located in the gst_templates.h file along with the corresponding array sizes.

Figure 6

Also the user can configure an upper cost sensitivity in the gesture_recon.h file. With the upper cost sensitivity the user can define the maximum cost value the program should accept as a valid gesture. In Figure 6, gestures are tested against the current templates, defined in gst_templates.h file. The “no” tag indicates that the gesture has greater cost value than any of upper cost values so the algorithm doesn’t recognize it as a valid gesture. The “to large” tag indicates that the gesture is larger than 60 samples.

4.2 Additional Configurations

In periph_setup.h file the user can also set:

- The motion interrupt condition definitions:
  - MOT_THR_SET: acceleration threshold.
  - MOT_DUR_SET: number of values over the MOT_THR_SET threshold.
  - MOT_DECR_RATE: decrement rate of non qualified threshold sample.
- The FIFO’s storing data rate:
  - SMRT_DIV
- The accelerometer and gyroscope range settings:
  - ACCEL_FS_SEL: accelerometer range.
  - GYRO_FS_SEL: gyroscope range.
- The Digital Low Pass Filter for gyroscope and accelerometer:
  - SENSOR_DLPF

In the gesture_recon.c file the user can set:

- The conditions met in order to properly isolate a gesture:
  - MEAN_WIN_THR: the variance value under which we consider that a gesture has ended.
  - VARIANCE_COUNT_UNDER_THR: every time the value of variance is under the MEAN_WIN_THR a counter increases. If the counter reaches VARIANCE_COUNT_UNDER_THR (a number of successive variance values under MEAN_WIN_THR) we consider that the gesture has ended.

4.3 Program Files

In the current paragraph functions used in the source code are being described.

1. MPU_6050.c

The file contains functions related to the MPU sensor.

Function Name	**MPU_Initialize**
Inputs	Void
Return Value	Void
Description	The function Initializes the mpu sensor, sets the range of the gyroscope and accelerometer, sets the fifo update rate and the digital low pass filter.

Function Name	**enable_MPU_Interrupts**

Inputs	Void
Return Value	Void
Description	The function sets and enables the interrupts on the sensor. It configures the motion interrupt, sets the accelerometer high pass filter and enables the motion and the FIFO overflow interrupt.

Function Name	**ring_enable_kbd_irq**
Inputs	Void
Return Value	Void
Description	Enables the interrupts on the processor and registers the callback function.

Function Name	**MPU_FIFO_Initialize**
Inputs	Void
Return Value	Void
Description	Configures the FIFO to store accelerometer data and enables the FIFO.

Function Name	**MPU_FIFOReset**
Inputs	Void
Return Value	Void
Description	Resets the FIFO

Function Name	**enableTimer**
Inputs	Void
Return Value	Void
Description	Sets, enables and registers the callback function for the timer when the software is in TIME_MEASUREMENT mode.

Function Name	**FIFOCount**
Inputs	Void
Return Value	Uint16_t

Description	Returns the number of valid data in the FIFO.

The majority of all functions related to the gesture recognition algorithm are identical to the functions used in the application software that were explained in chapter 3.