ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σχεδιασμός και ανάπτυξη cyberphysical γαντιού με στόχο την ψηφιακή καταγραφή και απεικόνιση των κινήσεων του χεριού του χρήστη (δημιουργία ψηφιακού avatar).»

ΝΙΚΟΛΑΟΣ ΜΕΛΑΣ ΑΜ: 508

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

Χρήστος Αντωνόπουλος

Αντίρριο 2017

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

κ Χρήστος Αντωνόπουλος

ΤΡΙΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

κ .Χρήστος Αντωνόπουλος

κ. Νίκος Βώρος

κ. Μιχάλης Παρασκευάς

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα Καθηγητή μου κ. Χ. Αντωνόπουλο για την υπόδειξη του συγκεκριμένου θέματος, καθώς επίσης για την εμπιστοσύνη που μου έδειξε, για τη διαρκή βοήθειά του και για τα πράγματα που μου δίδαξε, τόσο σε προσωπικό όσο και σε επαγγελματικό επίπεδο.

Επίσης, θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Ν. Βώρο για την ευκαιρία που μου πρόσφερε να συμμετάσχω στο εργαστήριο Ενσωματωμένων Συστημάτων (ESDA Lab), για την φιλική υποδοχή μου στην ομάδα του και για την βοήθεια και τις συμβουλές του σε όλα τα επίπεδα.

Νοέμβριος 2017

Περιεχόμενα

Στόχος διπλωματικής εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι η δημιουργία ενός cyber physical γαντιού το οποίο θα μπορεί να παρακολουθεί και να καταγράφει τις κινήσεις του χεριού του χρήστη.

Για την υλοποίηση του cyberphysical γαντιού έχει χρησιμοποιηθεί η ενσωματωμένη πλατφόρμα frdm mbed kl25z η οποία περιέχει ενσωματωμένο τον αισθητήρα επιτάχυνσης mma8451q. Η πλατφόρμα έχει ενσωματωθεί σε ένα κοινό γάντι ώστε να του προσδώσει την απαιτούμενη «εξυπνάδα».

Πιο συγκεκριμένα ο αλγόριθμος που έχει αναπτυχθεί κινείτε σε δυο βασικούς άξονες. Ο πρώτος άξονας αφορά την απεικόνιση σε πραγματικό χρόνο της γωνιακή θέση του γαντιού, προσομοιώνοντας το με έναν τρισδιάστατο κύβο (ψηφιακό κλισίμετρο). Ο δεύτερος άξονας στοχεύει στην παρακολούθηση των αλλαγών της γωνιακή θέση του γαντιού με τελικό σκοπό την αναγνώριση δυο διαφορετικών σεναρίων περιστροφικής κίνησης (pattern recognition).

Αρχικά στην παρούσα διπλωματική εργασία παρουσιάζετε ένας απλός τρόπος για την συλλογή των μετρήσεων ενός αισθητήρα επιτάχυνσης από έναν υπολογιστή γενικού σκοπού μέσω της σειριακής θύρας.

Στην συνεχεία αναλύονται διεξοδικά κατάλληλα μαθηματικά μοντέλα καθώς και διαφορετικές τεχνικές σχετικά με την καταγραφή της γωνιακής περιστροφής (rotation) και τον προσδιορισμός της γωνιακής θέσης (orientaton) ενός άκαμπτου σώματος στον τρισδιάστατο χώρο.

Επιπλέον παρουσιάζονται εκτενώς οι ενσωματωμένες συναρτήσεις του αισθητήρα mma8451q. Μια εξ αυτών, και συγκεκριμένα η συνάρτηση tap ή pulse event έχει παραμετροποιηθεί μέσω κατάλληλου αλγορίθμου που έχει αναπτυχθεί στην επεξεργαστική μονάδα της ενσωματωμένης πλατφόρμας και έχει χρησιμοποιηθεί. Η λειτουργικότητα που της έχει δοθεί είναι να αναγνωρίζει την ξαφνική επαφή μεταξύ δυο πολύ συγκεκριμένων σημείων του γαντιού και ενός οποιουδήποτε εξωτερικού αντικείμενου με σκοπό να αποφασίζει κάθε φορά ποιο από τα δυο μονοπάτια αναγνώρισης θα θέσει προς αναγνώριση.

Επίσης στην παρούσα διπλωματική εργασία παρουσιάζετε η αξιολόγηση της απόδοση του γαντιού και στους δύο βασικούς άξονες που έχουν τεθεί ως στόχοι της διπλωματικής εργασίας. Στο πρώτο πείραμα η κλίση της πλατφόρμα ταυτίστηκε με διαφορετικές γωνίες ενός μοιρογνωμονίου, και έπειτα από επαναλαμβανόμενες μετρήσεις αξιολογήθηκε η ακρίβεια του υπολογισμού

των γωνιών από τον αλγόριθμο. Στο δεύτερο πείραμα ανατέθηκε στην πλατφόρμα να αναγνωρίσει δυο διαφορετικά σενάρια κίνησης.

Από πλευράς λογισμικού έχει αναπτυχθεί αλγόριθμος στην επεξεργαστική μονάδα της ενσωματωμένης πλατφόρμας με την χρησιμοποίηση των προγραμματιστικών γλωσσών c και c++.

Επίσης αλγόριθμος με την γλώσσα προγραμματισμού python και την βιβλιοθήκης pygame (που προσδίδει στην γλώσσα το γραφικό περιβάλλον) έχει αναπτυχθεί στην πλατφόρμα προγραμματισμού Atom η οποία και «τρέχει» σε έναν υπολογιστική γενικού σκοπού. Ο αλγόριθμος παρουσιάζετε αναλυτικά μέσω της χρήσης διαγραμμάτων ροής (flow chars).

1. ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

1.1 Ορισμός και εισαγωγικές έννοιες

Ένα ενσωματωμένο σύστημα ορίζετε ως ένα εξειδικευμένο υπολογιστικό σύστημα ώστε να διεκπεραιώνει μια ή δυο πολύ συγκεκριμένες λειτουργίες. Το σύστημα αυτό ενσωματώνετε ως ένα μέρος ενός μεγαλύτερου συστήματος, μιας συσκευής δηλαδή η οποία και περιέχει υλικό (hardware) όπως ηλεκτρικά και μηχανικά μέρη καθώς και λογισμικό (software).

Τα ενσωματωμένα συστήματα διαφέρουν από τους υπολογιστές γενικού σκοπού οι οποίοι σχεδιάζονται και κατασκευάζονται ώστε να διαχειρίζονται ένα μεγάλο εύρος λειτουργιών.

Επειδή λοιπόν τα ενσωματωμένα συστήματα σχεδιάζονται και κατασκευάζονται για να διεκπεραιώνουν πολύ συγκεκριμένες λειτουργίες, κατά την σχεδίαση και δημιουργία τους υπάρχει η δυνατότητα βελτιστοποίησης τους όσον αφορά την κατανάλωσης ισχύς, το μέγεθος, την απόδοση και την αξιοπιστία τους.

Τον έλεγχο των ενσωματωμένων συστημάτων αναλαμβάνουν ένας ή περισσότεροι πυρήνες επεξεργασίας (processing cores) με την μορφή είτε των μικροελεγχτών (microcontrollers), είτε επεξεργαστών ψηφιακού σήματος(Digital Signal Processor-DSP), είτε συστοιχίας επιτόπιας προγραμματιζόμενης πύλης (Field Programmable Gate Arrays - FPGA), είτε ολοκληρωμένων κυκλωμάτων ειδικών για εφαρμογές (Application Specific Integrated Circuits - ASIC). Τα παραπάνω επεξεργαστικά τμήματα (components) αλληλεπιδρούν με άλλα τμήματα που είναι ειδικά για την διαχείριση ηλεκτρικών ή μηχανικών διεπαφών (interface).

1.2 Συστατικά Μέρη

Το βασικό μέρος ενός ενσωματωμένου συστήματος αποτελείται από τον επεξεργαστή (processor), ο οποίος μπορεί να είναι ένας γενικός μικροεπεξεργαστής ή ένας μικροελεγχτής και προγραμματίζετε για να διεκπεραιώνει συγκεκριμένες λειτουργίες για τις οποίες και το συνολικό σύστημα έχει σχεδιαστεί.

Η ηλεκτρονική μνήμη αποτελεί ένα σημαντικό μέρος ενός ενσωματωμένου συστήματος ενώ οι βασικοί τύποι μνήμης είναι τρείς. Η RAM (Random Access Memory), η ROM (Read Only Memory) και η Cache. Η RAM αποτελεί ένα από τα βασικά μέρη υλικού (hardware component) όπου τα δεδομένα αποθηκεύονται προσωρινά κατά την διάρκειας της λειτουργίας του συστήματος.

Η ROM περιέχει ρουτίνες εισόδου – εξόδου οι οποίες είναι χρήσιμες για το σύστημα κατά τον χρόνο εκκίνησης του. Η Cache, αντίθετος χρησιμοποιείτε από τον επεξεργαστή για την προσωρινή αποθήκευση των δεδομένων κατά την διάρκεια της επεξεργασίας και της μεταφοράς τους.

Το ρολόι του συστήματος (system clock) χρησιμοποιείτε για οποιαδήποτε διεργασία «τρέχει» σε ένα ενσωματωμένο σύστημα και απαιτεί ακριβείς πληροφορίες χρονισμού. Το ρολόι αποτελείται γενικά από έναν ταλαντωτή καθώς και ορισμένα συνδεδεμένα ψηφιακά κυκλώματα.

Οι περιφερειακές συσκευές (peripherals) παρέχονται στις ενσωματωμένες πλατφόρμες για μια εύκολη ενσωμάτωση. Οι τυπικές συσκευές περιλαμβάνουν σειριακή θύρα, παράλληλη θύρα, θύρα δικτύου, πληκτρολόγιο και θύρες ποντικιού, θύρα μονάδας μνήμης και θύρα οθόνης. Ορισμένα ενσωματωμένα συστήματα έχουν επιπλέον άλλες θύρες όπως CAN-bus.

1.3 Χαρακτηριστικά

Η πλειοψηφία των ενσωματωμένων συστημάτων σχεδιάζονται ώστε να εκτελούν μια συγκεκριμένη λειτουργία σε χαμηλό κόστος. Τα περισσότερα από αυτά τα συστήματα επιπλέον έχουν περιορισμούς στην αποδοτικότητα τους σχετικά με το υλικό και το λογισμικό τους, όπως συστήματα που απαιτητέ να λειτουργούν σε πραγματικό χρόνο με υψηλές ταχύτητες κατά την εκτέλεση των λειτουργιών τους, αλλά επίσης μπορούν να είναι πιο ανεκτικά όταν εκτελούν διαφορετικές λειτουργίες.Τα χαρακτηριστικά ενός ενσωματωμένου συστήματος αναφέρονται σε ορισμένους μετρήσιμους παράγοντες κατά την διάρκεια της σχεδίασης του συστήματος.

Η κατανάλωση ενέργειας (power consumption) αποτελεί ένα πολύ σημαντικό παράγοντα για όλα τα ενσωματωμένα συστήματα που τροφοδοτούνται από μπαταρία. Έτσι, το ποσό της ενέργειας που καταναλώνετε από το σύστημα συνδέετε άρρηκτα με την χωρητικότητα ή την διάρκεια ζωής της μπαταρίας.

Η ευελιξία (Flexibility) ορίζετε ως η ικανότητα του συστήματος να προσαρμόζετε στις αλλαγές της λειτουργικότητα του. Οι αλλαγές προκύπτουν από την συντήρηση, τις βελτιώσεις και τις αναβαθμίσεις. Έτσι το λογισμικό θεωρείται πολύ ευέλικτο όταν μπορεί να ενημερωθεί ανά πάσα στιγμή από μια νέα έκδοση. Η ευελιξία γίνετε ακόμα πιο κρίσιμης σημασίας όταν το υπό ανάπτυξη ενσωματωμένο σύστημα έχει απαιτήσεις πραγματικού χρόνου. Ένας ευέλικτος σχεδιασμός θα επιτρέψει την προσθήκη επιπλέον καθηκόντων (tasks) χωρίς να επηρεάσει τα άλλα καθήκοντα προκαλώντας τους να χάσουν τις προθεσμίες.

Η επεξεργαστική ισχύς (processor power) ενός ενσωματωμένου συστήματος προσδίδει στον μικροελεγχτή που διαχειρίζεται το σύστημα, την δυνατότητα να χειριστεί αποτελεσματικά ένα ή πολλά καθήκοντα (tasks).

Το λειτουργικό σύστημα (operating system) ενός ενσωματωμένου συστήματος είναι συνήθως αρκετά περιορισμένο όσον αφορά την λειτουργικότητα του, και ανάλογα με την συσκευή μπορεί να εκτελεί ακόμα και μια μόνο εφαρμογή.Ωστόσο αυτή η μοναδική εφαρμογή είναι ζωτικής σημασίας για την σωστή λειτουργία του συστήματος. Οπότε ένα λειτουργικό σύστημα ενός ενσωματωμένου πρέπει να είναι αξιόπιστο καθώς και να έχει την δυνατότητα να εκτελείτε υπό αρκετούς περιορισμούς σε μνήμη, μέγεθος αλλά και ικανότητα επεξεργασίας.

Η Επίδοση (Performance) ενός ενσωματωμένου συστήματος φανερώνει το πόσο καλά εκτελεί το σύστημα την εργασία που έχει σχεδιαστεί να διεκπεραιώνει. Η επίδοση μπορεί να μετρηθεί από τον χρόνο εκτέλεσης ή από

το μέγιστο ρυθμός επεξεργασίας των απαιτούμενων δεδομένων (throughput) του συστήματος.

Η μνήμη (Memory) περιέχει στην ROM ή στην Flash memory το πρόγραμμα που έχει αναπτυχθεί για το ενσωματωμένο σύστημα και το αντιμετωπίζει ως υλικολογισμικό (firmware). Το firmware είναι ένα είδος λογισμικού το οποίο είναι γραμμένο σε γλώσσα μηχανής (ή σε συμβολική γλώσσα) και είναι φτιαγμένο αποκλειστικά και μόνο για κάθε ένα μοντέλο συσκευής.

Το μέγεθος (size) ενός ενσωματωμένου συστήματος θα πρέπει να είναι όσον το δυνατόν πιο μικρό. Το μέγεθος του λογισμικού μετράτε σε bytes και σε transistor ή IC ή πύλες.

Η αξιοπιστία (Reliability) ενός ενσωματωμένου συστήματος θα πρέπει να βρίσκετε σε όσο το δυνατόν σε υψηλοτέρα επίπεδα προκειμένου το σύστημα να επιτυγχάνετε όσο το δυνατόν καλύτερη απόδοση καθ’ όλη την διάρκεια της «ζωής» του.

Η Ασφάλεια (Safety) εξασφαλίζει ότι κατά την διάρκεια μιας αποτυχίας (failure) του συστήματος δεν θα προκληθεί ζημιά σε άλλους.

Η Ορθότητα (Correctness) αναφαίρετα στον έλεγχο της λειτουργικότητας του ενσωματωμένου συστήματος και υποδεικνύει αν είναι σωστή ή όχι.

Η Συντήρηση (Maintainability) είναι ο πιο σημαντικός παράγοντας στο οποίο το σύστημα μπορεί να επισκευαστεί ή να αντικατασταθεί μέσα σε ένα συγκεκριμένο χρονικό διάστημα.

1.4 Εφαρμογές

Αρχικά, τα ενσωματωμένα συστήματα χρησιμοποιήθηκαν για εφαρμογές κρίσιμης σημασίας όπως για την ασφάλεια αλλά και για τις επιχειρήσεις. Συγκεκριμένα χρησιμοποιήθηκαν για τον έλεγχο πυραύλων και δορυφόρων, έλεγχο παραγωγής ενέργειας, έλεγχο εναέριας κυκλοφορίας, έλεγχο τηλεφωνικών κέντρων.

Η έρευνα και η ανάπτυξη των ενσωματωμένων συστημάτων αφορά πλέον ένα μεγάλο ποσοστό των προηγμένων προϊόντων που σχεδιάζονται σε όλο το κόσμο. Κατά κάποιο τρόπο οι ενσωματωμένες τεχνολογίες διαχειρίζονται την παγκόσμια βιομηχανία μεταφορών που περιλαμβάνει αυτοκινητοβιομηχανία, τρένα και διαστημική βιομηχανία.

Επιπλέον τα ενσωματωμένα συστήματα χρησιμοποιούνται σε όλες τις ηλεκτρικές και ηλεκτρονικές συσκευές όπως κάμερες, παιχνίδια, τηλεοράσεις, οικιακές συσκευές, ηχητικά συστήματα και κινητά τηλέφωνα.

Συγκεκριμένα οι προηγμένες ενσωματωμένες τεχνολογίες αναπτύσσονται, αφενός στον έλεγχο διαδικασιών για την παραγωγή και διανομή ενέργειας, την αυτοματοποίηση και την βελτίωση των εργοστασίων καθώς και στις τηλεπικοινωνίες (δορυφόροι, κινητά τηλέφωνα, δίκτυα τηλεπικοινωνιών).

Αφετέρου, στην διαχείριση ενέργειας κατά την παραγωγή, την διανομή και την βελτιστοποιημένη χρήση της καθώς και στην ασφάλεια όπως το ηλεκτρονικό εμπόριο και οι έξυπνες κάρτες αλλά και στον τομέα της υγείας, δηλαδή σε νοσοκομειακός εξοπλισμός και στην παρακολούθηση ασθενών.

2. ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ FRDM KL25Z ΠΛΑΤΦΟΡΜΑΣ

2.1 Επεξεργαστικά χαρακτηριστικά

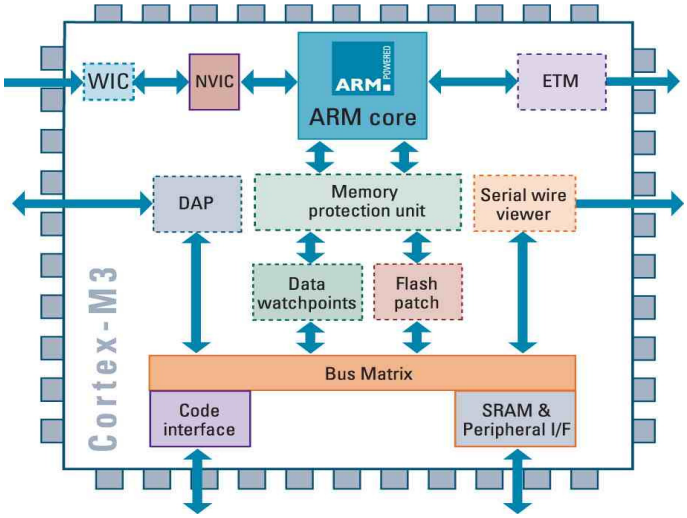
Ο επεξεργαστής ARM® Cortex®-M0+ είναι ο πλέον ενεργειακά αποδοτικός επεξεργαστής που είναι αυτή τη στιγμή διαθέσιμος. Έχει κατασκευαστεί με πρότυπο τον ήδη πολύ επιτυχημένο Cortex – M0 επεξεργαστή και διατηρεί την πλήρη συμβατότητα σε ότι αφορά τα εργαλεία και τις εντολές ενώ καταφέρνει να μειώσει ακόμη περισσότερο την κατανάλωση σε ενέργεια και αυξάνει την απόδοση. Όπως και στον Cortex – M0 η πολύ μικρή περιοχή πυριτίου, η χαμηλή κατανάλωση ισχύος και το ελάχιστο αποτύπωμα κώδικα αυτών των επεξεργαστών επιτρέπει στους προγραμματιστές να επιτύχουν μία απόδοση 32bit σε επεξεργαστές τιμών αναλόγων αυτών των 8bit (ARM). Ο επεξεργαστής Cortex M0+ περιλαμβάνει ένα πολύ μεγάλο αριθμό εφαρμογών ώστε να προσφέρει ιδιαίτερα ευέλικτη ανάπτυξη εφαρμογών. Η ομάδα των εντολών που χρησιμοποιείται είναι πλήρης και συμβατή με τις προηγούμενες εκδόσεις ώστε να χρησιμοποιούνται τα ίδια εργαλεία σε ότι αφορά το compile και το debug. Το pipeline του Cortex M0+ έχει μειωθεί από 3 σε 2 στάδια τα οποία μειώνουν και την κατανάλωση ισχύος (ARM). Στον Cortex M0+ έχουν επίσης προστεθεί τα χαρακτηριστικά των Cortex-M3 και Cortex-M4.

Τα βασικά χαρακτηριστικά του Cortex M0+ είναι (ARM):

- Αρχιτεκτονική ARMv6-M
- Pipeline 2 σταδίων
- Σετ εντολών :
- Thumb, χωρίς τις CBZ, CBNZ, IT.
- Thumb-2, μόνο οι BL, DMB, DSB, ISB, MRS, MSR.
- 32-bit hardware πολλαπλασιαζόμενο με 32-bit αποτέλεσμα
- 1 έως 32 διακοπές

2.2 Αρχιτεκτονικά χαρακτηριστικά

Η αρχιτεκτονική ARM εξελίχθηκε τα τελευταία χρόνια και έχει φθάσει σε σημείο τέτοιο ώστε να υποστηρίζει υλοποιήσεις ενός μεγάλου φάσματος σημείων απόδοσης. Η αρχιτεκτονική απλότητα των επεξεργαστών ARM οδήγησαν σε υλοποιήσεις πολύ μικρού μεγέθους, και υλοποιήσεις που επιτρέπουν τη χρήση συσκευών με πολύ χαμηλή κατανάλωση ενέργειας (ARM Architecture reference manual, 2005). Το μέγεθος, η απόδοση και η πολύ χαμηλή κατανάλωση ενέργειας αποτελούν τα βασικά χαρακτηριστικά εξέλιξης της αρχιτεκτονικής ARM.



Εικόνα α. Αρχιτεκτονική ARM

Η ARM ανήκει στην κατηγορία RISC καθώς περιλαμβάνει κάποια από τα βασικά χαρακτηριστικά των αρχιτεκτονικών αυτών όπως (ARM White Paper, 2014):

- Ένα μεγάλο ενιαίο αρχείο καταγραφής
- Μία αρχιτεκτονική load/store όπου οι λειτουργίες επεξεργασίας των δεδομένων λειτουργούν σε πλαίσια καταχώρησης και όχι άμεσα επάνω στα πλαίσια των μνημών
- Απλοί τρόποι διευθυνσιοδότησης, με όλες τις load/store διευθύνσεις να καθορίζονται από τους καταχωρητές και τα πεδία των εντολών

- Ενιαία και σταθερού μεγέθους πεδία εντολών ώστε να απλοποιείται η αποκωδικοποίση.

Επιπλέον η αρχιτεκτονική ARM παρέχει (ARM White Paper, 2014):

1. Έλεγχο στην ALU (Arithmetic Logic Unit) και shifter στις περισσότερες εντολές επεξεργασίας δεδομένων έτσι ώστε να μεγιστοποιηθεί η χρήση σε μία ALU και έναν shifter.
2. Τρόπους διευθυνσιοδότησης αυτό – αύξησης και αυτό – μείωσης έτσι ώστε να βελτιστοποιηθούν οι βρόγχοι των προγραμμάτων.
3. Πολλαπλές εντολές Load και Store έτσι ώστε να βελτιστοποιηθεί το throughput των δεδομένων .
4. Εκτέλεση κατά συνθήκη όλων των εντολών έτσι ώστε να μεγιστοποιηθεί το throughput της εκτέλεσης .

Οι βελτιώσεις αυτές σε μία βασική RISC επιτρέπουν στους επεξεργαστές ARM να επιτυγχάνουν ιδιαίτερα υψηλές αποδόσεις, μικρό μέγεθος κώδικα, χαμηλή κατανάλωση ενέργειας και μικρή περιοχή πυριτίου.

2.3 Μνήμη

Ο επεξεργαστής περιέχει έναν πίνακα διαύλου ο οποίους διαχειρίζεται τον πυρήνα του επεξεργαστή και προαιρετικά η θύρα πρόσβασης εντοπισμού σφαλμάτων (Debug Access Port-DAP) στην μνήμη έχει πρόσβαση τόσο στο εξωτερικό σύστημα μνήμης όσο και στα εσωτερικά NVIC και debug στοιχεία. Προτεραιότητα δίνετε πάντοτε στον επεξεργαστή ώστε να εξασφαλιστεί ότι οποιεσδήποτε προσβάσεις για debug θα είναι όσο το δυνατόν λιγότερο ενοχλητικές. Για ένα σύστημα μηδενικής αναμονής, όλες οι debug εντολές στην μνήμη του συστήματος, οι συσκευές NVIC και οι πόροι για το debug είναι εντολές μη-παρεμβατικές. Ο χάρτης μνήμης του συστήματος είναι ARMV6-M και είναι κοινός για την πρόσβαση τόσο στον debbuger όσο και στον πυρήνα.

Ο χάρτης μνήμης χωρίζετε σε περιοχές. Κάθε περιοχή έχει ένα καθορισμένο τύπο μνήμης και ορισμένες περιοχές έχουν πρόσθετα χαρακτηριστικά μνήμης. Ο τύπος και τα χαρακτηριστικά της μνήμης καθορίζουν την συμπεριφορά των προσπελάσεων στην περιοχή.

Οι τρεις διαφορετικοί τύποι μνήμης είναι:

- Normal: Ο επεξεργαστής μπορεί να αναδιατάξει την σειρά των ανταλλαγών δεδομένων ώστε να πετύχει αποδοτικότητα.
- Device: Ο επεξεργαστής καθορίζει την σειρά ανταλλαγών με βάση άλλες ανταλλαγές με την συσκευή ή με την Strongly-ordered μνήμη.
- Strongly-ordered: Ο επεξεργαστής διατηρεί τη σειρά ανταλλαγών σε σχέση με όλες τις άλλες ανταλλαγές.

  Οι διαφορετικές απαιτήσεις από την συσκευή ή από την strongly-ordered μνήμη σημαίνουν ότι το σύστημα μνήμης μπορεί να κάνει buffer μια εγγραφή στην μνήμη της συσκευής, αλλά όχι μια εγγραφή στην strongly-ordered μνήμη.

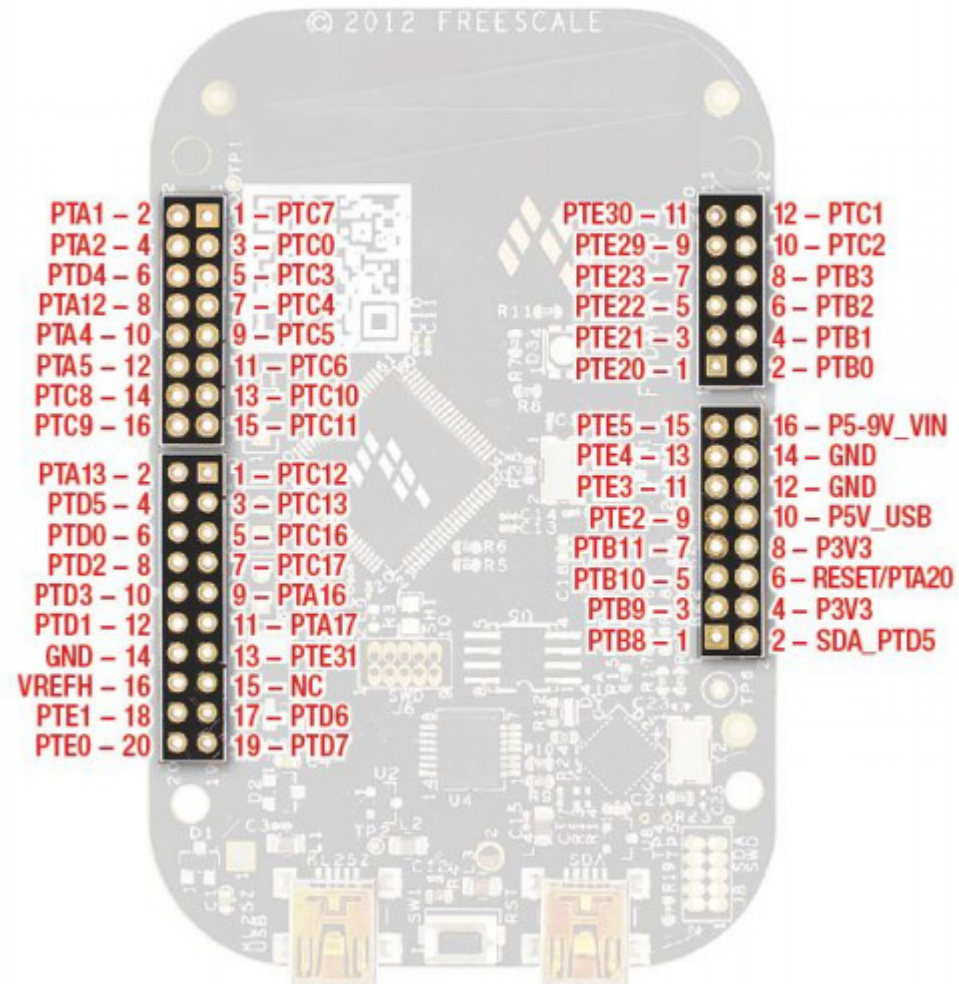
Αν η συσκευή σκοπεύετε να χρησιμοποιηθεί σε ένα σύστημα όπου η μνήμη διαμοιράζετε μεταξύ πολλών επεξεργαστών τότε το χαρακτηριστικό shearable (διαμερισμός) παρέχετε από την μνήμη. Συγκεκριμένα το σύστημα μνήμης παρέχει συγχρονισμό μεταξύ των masters του διαύλου, σε ένα σύστημα με πολλαπλούς bus masters, όπως για παράδειγμα μια συσκευή με έναν DMA ελεγκτή.

2.4 I/O interfaces

Ο επεξεργαστής CORTEX-MO+ υλοποιεί μια ειδική θύρα εισόδου/εξόδου μονού κύκλου (single-cycle I/O port) η οποία επιτυγχάνει υψηλής ταχύτητας πρόσβαση στα περιφερειακά σε ένα κύκλο. Η θύρα εισόδου/εξόδου είναι memory mapped και υποστηρίζει όλες τις εντολές load και store όπως ADR, LDM, LDR {type}, LDR, POP, PUSH, STM, STR {type}. Η θύρα εισόδου/εξόδου μονού κύκλου δεν υποστηρίζει την εκτέλεση κώδικα.

Ο μικροελεγκτής KL25Z128VLK4 είναι συσκευασμένος σε LQFP με 80 pins. Κάποια pins χρησιμοποιούνται στα on-board κυκλώματα και πολλά συνδέονται απευθείας με τις τέσσερις κεφαλίδες (headers) εισόδου/εξόδου.

Τα pins του KL25Z μικροελεγχτή έχουν πάρει το όνομα τους από την γενικού σκοπού λειτουργικότητα που παρέχουν. Για παράδειγμα το πρώτο pin της θύρας Α αναφέρετε ως PTA1.



Εικόνα β. KL25Z I/O pins

2.5 Sensors

Ο αισθητήρας επιτάχυνσης mma8451q βρίσκετε ενσωματωμένος στην πλατφόρμα KL25Z. Πρόκειται για ένα αισθητήρα με χαμηλή κατανάλωση (low - power) ο οποίος αναγνωρίζει επιταχύνσεις στους τρείς άξονες (3-axis) και επικοινωνεί με τον μικροελεγχτή μέσω ενός I2C πρωτόκολλου επικοινωνίας και μέσω δυο GPIO σημάτων (εικόνα γ).

MMA8451Q	KL25Z128
SCL	PTE24
SDA	PTE25
INT1	PTA14
INT2	PTA15

Εικόνα γ. Σήματα επικοινωνίας του αισθητήρα επιτάχυνσης

3. RPC ΠΡΩΤΟΚΟΛΛΟ ΕΠΙΚΟΙΝΩΝΙΑΣ

3.1 Ορισμός

Μια κλήση απομακρυσμένης διαδικασίας (Remote Procedure Call - RPC) είναι ένα πρωτόκολλο όπου ένα πρόγραμμα μπορεί να το χρησιμοποιήσει για να αιτηθεί μια υπηρεσία από ένα πρόγραμμα που βρίσκετε σε μια άλλη συσκευή ή σε ένα άλλο δίκτυο χωρίς να γνωρίζει τις λεπτομερείς του δικτύου.

Μια κλήση απομακρυσμένης διαδικασίας είναι επίσης γνωστή ως κλήση μιας συνάρτησης ή κλήση μιας υπορουτίνας.

Το RPC πρωτόκολλο χρησιμοποιεί το μοντέλο πελάτη – εξυπηρετητή (client – service model). Το πρόγραμμα που αιτείτε είναι ο πελάτης και το πρόγραμμα που εξυπηρετεί είναι ο εξυπηρετητής.

Όπως μια τοπική ή κανονική κλήση μιας διαδικασίας (procedure), ένα RPC είναι μια σύγχρονη διαδικασία (synchronous operation) που απαιτεί το αιτούμενο πρόγραμμα να ανασταλεί μέχρι να επιστραφούν τα αποτελέσματα της απομακρυσμένης διαδικασίας.

Με πιο απλά λόγια μια κλήση απομακρυσμένης διαδικασίας συμβαίνει όταν ένα κώδικας που εκτελείτε προκαλεί την εκτέλεση μιας υπορουτίνας (procedure) σε μια άλλη περιοχή μνήμης (address space) που βρίσκετε συνήθως σε ένα άλλο υπολογιστικό σύστημα, και ενώ έχει προγραμματιστεί σαν να ήταν μια τοπική κλίση διαδικασίας.

3.2 Υλοποίηση σειριακής RPC διασύνδεσης

Η δημιουργία μιας διεπαφής μεταξύ ενός λογισμικού υπολογιστή και μιας ενσωματωμένης πλατφόρμας κρύβει δυσκολίες γιατί απαιτείτε από εμάς να καθορίσουμε μια φόρμα επικοινωνίας και στην συνέχεια να αναπτύξουμε κώδικα για την χαμηλού επιπέδου επικοινωνία τόσο στο ενσωματωμένο όσο και στο υπολογιστικό σύστημα ώστε να πραγματοποιηθεί η διεπαφή[1].

Οι πλατφόρμες mbed όμως έχουν την δυνατότητα να λαμβάνουν και να ερμηνεύουν RPC εντολές και αυτή η ικανότητα μπορεί να μας φανεί ιδιαιτέρως χρήσιμη για να απλοποιήσει σημαντικά την δημιουργία της διεπαφής που καλούμαστε να πραγματοποιήσουμε κατά την παρούσα διπλωματική εργασία μεταξύ της ενσωματωμένης πλατφόρμας mbed frdm kl25z και του λογισμικού υπολογιστή.

Οι RPC εντολές έχουν προκαθορισμένη μορφή/φόρμα και μπορούν να αποσταλούν μέσω οποιουσδήποτε μηχανισμού μεταφοράς ο οποίος αρκεί να έχει την δυνατότητα αποστολής μιας ακολουθίας κειμένου (stream a text). Επιπλέον επιτρέπουν την άμεση αλληλεπίδραση με τα αντικείμενα της πλατφόρμας.

Για να υλοποιηθεί μια RPC διασύνδεση η οποία εκτός των άλλων να μπορεί να μεταφέρει τους βασικούς τύπους μεταβλητών πρέπει να εισάγουμε RPC λειτουργικότητα (RPC Functionality) τόσο στον κώδικα της πλατφόρμας όσο και στο λογισμικό του υπολογιστή γενικού σκοπού.

Υπάρχουν βιβλιοθήκες για αρκετές δημοφιλής γλώσσες προγραμματισμού όπως python, java, LabView, MATLAB, .NET, οι οποίες επιτρέπουν να χρησιμοποιηθεί το RPC πρωτόκολλο μέσω αρκετών μηχανισμών μεταφοράς.

Οι δυο βασικοί μηχανισμοί μεταφοράς που υποστηρίζει το RPC πρωτόκολλο είναι «πάνω» από σειριακή ή από HTTP.

Μια σειριακή σύνδεση μεταξύ της πλατφόρμας mbed frdm frdkl25 και του λογισμικού python έχει δημιουργηθεί για τις ανάγκες της παρούσας διπλωματικής εργασίας. Η βιβλιοθήκη που χρησιμοποιήθηκε κατά την ανάπτυξη κώδικα στην ενσωματωμένη πλατφόρμα ονομάζεται SerialRPCInterface και αποτελεί την τυπική βιβλιοθήκη ώστε να «στηθεί» μια σειριακή RPC σύνδεση. Βασική λειτουργία της είναι να ρυθμίζει όλες τις παραμέτρους της RPC διασύνδεση, καταγράφοντας (registering) όλες τις κλάσεις βάσης και τη σειριακή θύρα κάθε φορά που αποστέλλεται μια RPC εντολή.

Εκτός όμως από την υλοποίηση των κλάσεις «επισύναψης»(RPC wrapper classes) όπως DigitalIn, DigitalOut, PWMOut που μπορούν να δεχτούν εντολές (0 ή 1) από ένα τερματικό και για παράδειγμα να ανάψουν ή να σβήσουν ένα

LED στην πλατφόρμα, η βιβλιοθήκη μας παρέχει επιπλέον, έναν τρόπο για να προστεθεί στον κώδικα μας RPC λειτουργικότητα (RPC Functionality).

Τα RPCFUNCTION αντικείμενα κάνουν εφικτή την κλήση μιας συνάρτησης που έχει δημιουργηθεί από τον χρήστη «πάνω» από μια RPC διασύνδεση.

Επιπροσθέτως τα RPCVariable αντικείμενα επιτρέπουν σε μεταβλητές βασικού τύπου να είναι προσβάσημες μέσω μιας RPC διασύνδεσης.

Οι τρεις μεταβλητές που μεταφέρονται από το ενσωματωμένο στον υπολογιστή γενικού σκοπού μέσω τον RPCVariables αντικειμένων είναι τύπου float και περιέχουν τις μετρήσεις του αισθητήρα επιτάχυνσης MMA8451Q στους άξονες X, Y και Z . Η τέταρτη έχει ονομαστεί SR (source register) είναι τύπου int και χρησιμοποιείται για αναγνωρίζει σε ποιον ή ποιους άξονες αναγνωρίστηκε η πρόσκρουση της πλατφόρμας με το εξωτερικό αντικείμενο. [i]

4. ΤΕΧΝΙΚΕΣ ΓΙΑ ΤΗΝ ΑΝΑΓΝΩΡΙΣΗ ΘΕΣΗΣ (ΚΛΙΣΟΜΕΤΡΟ) ΑΛΛΑ ΚΑΙ ΣΗΜΕΙΟΥ ΣΤΙΓΜΙΑΙΑΣ ΕΠΑΦΗΣ

4.1 Βασικές μέθοδοι αναγνώρισης του προσανατολισμού ενός στερεού σώματος στον τρισδιάστατο χώρο

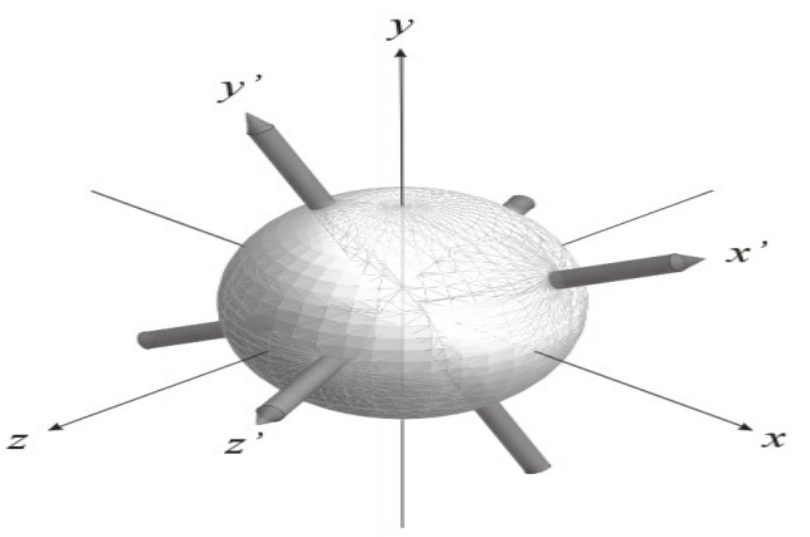
Ο γωνιακός προσανατολισμός (angular orientation) αναφέρετε στην θέση ενός στερεού σώματος σε σχέση με ένα σύστημα συντεταγμένων αναφοράς ή αλλιώς αρχικό σύστημα συντεταγμένων.

Καθορίζετε από μια περιστροφή (rotation) που χρειάζεται να συμβεί ώστε να κινήσει το στερεό σώμα από μια θέση αρχικά ευθυγραμμισμένη με ένα σύστημα συντεταγμένων αναφοράς σε μια καινούργια θέση.

Οι συντεταγμένες αναφοράς και οι συντεταγμένες του στερεού σώματος θεωρούνται στο καρτεσιανό επίπεδο με τους άξονες x, y, z και x',y',z, αντίστοιχα (εικόνα 1). Ο προσανατολισμός των αξόνων του συστήματος ακολουθεί τον κανόνα του δεξιού χεριού[2].

Ουσιαστικά για να καθορίσουμε τον γωνιακό προσανατολισμό ενός στερεού σώματος στον Ευκλείδειο χώρο απαιτούνται τρεις παράμετροι.

Έχουν αναπτυχτεί αρκετοί μέθοδοι παραμετροποίησης για την μαθηματική αναπαράσταση της μεταφοράς (transformation) ή της περιστροφής (rotation) ενός στερεού σώματος. Οι πιο κοινές μέθοδοι είναι οι πίνακες κατεύθυνσης μεταφοράς (Direction cosine metrices), οι γωνίες του Όηλερ (Euler Angles) και τα κουατερνια (Quaternions).



Εικόνα 1. Συστήματα συντεταγμένων αναφοράς και στερεού σώματος

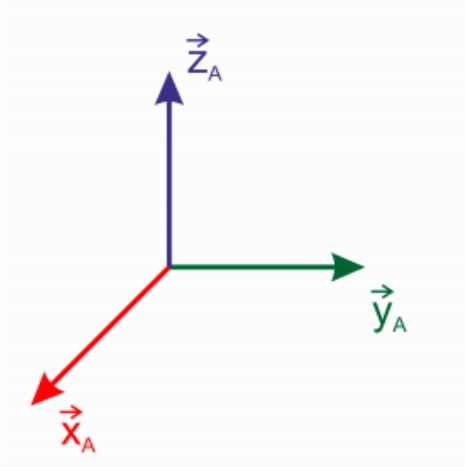
Σε αυτό το σημείο θεωρείτε σκόπιμο να διαλευκανθεί η διαφορά μεταξύ του μετασχηματισμού (transformation) και της περιστροφής (rotation). Στο πεδίο της αναπαράστασης στάσης λοιπόν, οι όροι μετασχηματισμός(transformation) και περιστροφή (rotation) συχνά δεν είναι σαφώς διαχωρισμένοι με αποτέλεσμα να προκύπτουν απρόσμενα υπολογιστικά αποτελέσματα ή λανθασμένες πληροφορίες.

Η περιστροφή(rotation) ενός διανύσματος u σε ένα σύστημα συντεταγμένων A είναι μια λειτουργία που τροποποιεί την αναπαράσταση του διανύσματος u στο A. Δοθείσας της αναπαράστασης του u και του A σε ένα σύστημα συντεταγμένων S μια περιστροφή αλλάζει τον προσανατολισμό (orientation) του u και στο σύστημα συντεταγμένων A και στο σύστημα συντεταγμένων S.

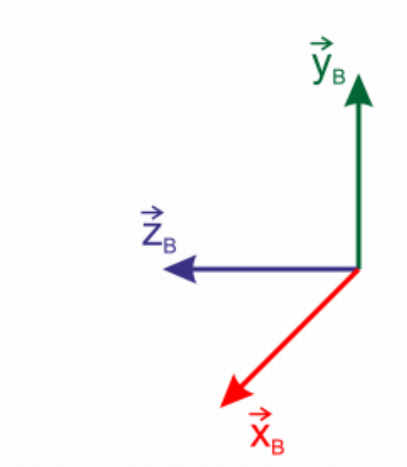
Ο μετασχηματισμός (transformation) μιας συντεταγμένης είναι μια λειτουργία η οποία περιγράφει την αναπαράσταση ενός διανύσματος u σε σχέση με ένα δεύτερο σύστημα συνταγμένων, B.

Δοθείσας της αναπαράστασης του A, του B και του u σε ένα σύστημα συντεταγμένων S, ο μετασχηματισμός δεν αλλάζει τον προσανατολισμό του u στο S.

Στις (εικόνες 2 και 3) παρουσιάζουν δυο συστήματα συντεταγμένων A και B. Για να υπολογίσουμε τα βασικά διανύσματα του B, θα πρέπει τα βασικά διανύσματα του A να περιστραφούν κατά 90^{o} γύρω από τον άξονα X_A .



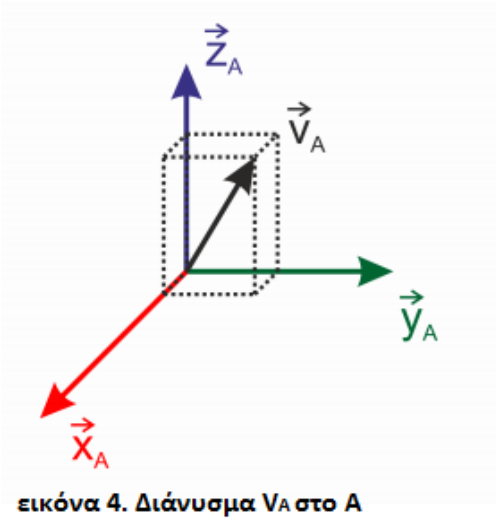
εικόνα 2. Σύστημα συντεταγμένων A



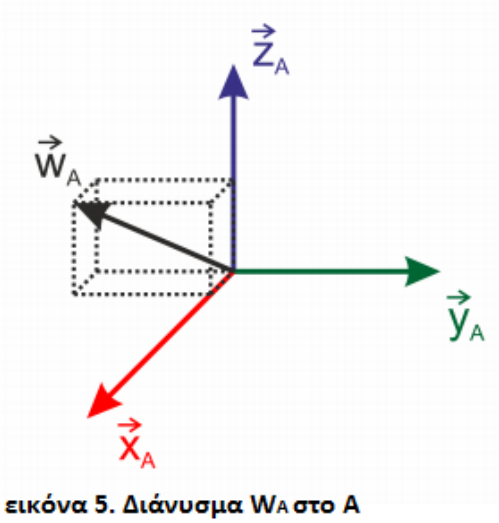
εικόνα 3. Σύστημα συντεταγμένων B

Αρχικά ορίζουμε ένα διάνυσμα στο A (εικόνα 4) και εφαρμόζουμε μια περιστροφή(rotation) διανύσματος 90^o γύρω από τον άξονα X_A . Το αποτέλεσμα αυτής της περιστροφής καλείτε W_A και εξακολουθεί να εκφράζετε στο A (εικόνα 5).

Αν το V_A = (1 2 3) στο A, τότε το περιστρεφόμενο διάνυσμα στο A είναι το W_A = (1 -3 2).



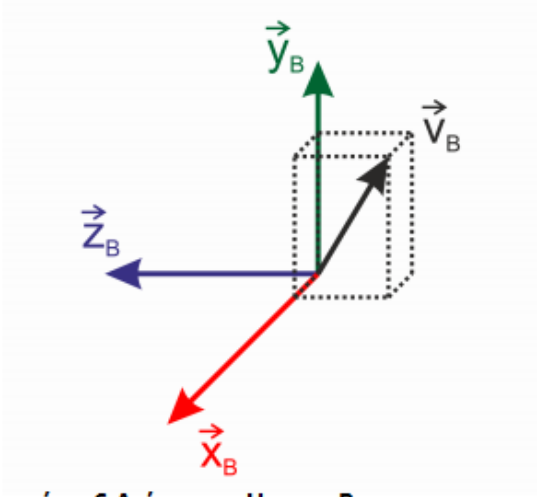
εικόνα 4. Διάνυσμα V_A στο A



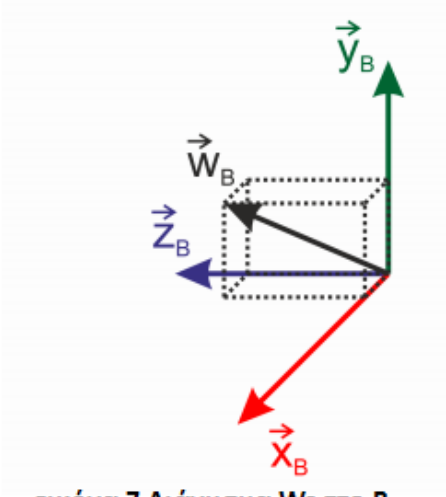
εικόνα 5. Διάνυσμα W_A στο A

Τώρα μετασχηματίζουμε (transform) τόσο το VA όσο και το W_A στο B. Ο προσανατολισμός των δυο διανυσμάτων στο χώρο παραμένει ο ίδιος παρόλο που το σύστημα συντεταγμένων τα περιγράφει αλλαγμένα. Τα μετασχηματισμένα διανύσματα καλούνται $\vec{V}_B$ και $\vec{W}_B$, αντίστοιχα.

Το διάνυσμα $\vec{V}_B$ στο B = (1 3 -2) όπως φαίνετε στην εικόνα 6 ενώ το διάνυσμα $\vec{W}_B$ = (1 2 3) όπως φαίνετε στην εικόνα 7.



εικόνα 6 Διάνυσμα U_B στο B



εικόνα 7 Διάνυσμα W_B στο B

Αποδείχτηκε ότι η περιστροφή ενός διανύσματος (vector rotation) που φαίνετε στις (εικόνες 4 και 5) είναι διαφορετική από τον μετασχηματισμός ενός διανύσματος που φαίνετε στις (εικόνες 4 5 6). Επιπλέον μπορούμε να παρατηρήσου ότι η περιστροφή και ο μετασχηματισμός είναι αντίθετες λειτουργίες[3].

Πίνακες Κατεύθυνσης Μεταφοράς (Direction Cosines Matrices): Ένας πίνακας κατεύθυνσης μεταφοράς είναι ένας πίνακας μεταφοράς ο οποίος αποτελείται από τις τιμές κατεύθυνσης συνημίτονου μεταξύ του αρχικού συστήματος συντεταγμένων και του συστήματος συντεταγμένων «στόχου».

Έστω A το αρχικό σύστημα συντεταγμένων και B το σύστημα συντεταγμένων του «στόχου» στο όποιο θα πραγματοποιηθεί η μεταφορά.

Τα διανύσματα βάσης του A δίνονται ως x_A, y_A, z_A ενώ όπου x_B, y_B, Z_B είναι τα διανύσματα βάσης του συστήματος B. Ο πίνακας κατεύθυνσης μεταφοράς ο οποίος μεταφέρει ένα διάνυσμα από το σύστημα A στο σύστημα B καλείται $T_{B<-A}$ και ορίζετε ως εξής:

$$T_{B\leftarrow A} = \begin{pmatrix} cos\sphericalangle(\vec{x}_A,\vec{x}_B) & cos\sphericalangle(\vec{y}_A,\vec{x}_B) & cos\sphericalangle(\vec{z}_A,\vec{x}_B) \\ cos\sphericalangle(\vec{x}_A,\vec{y}_B) & cos\sphericalangle(\vec{y}_A,\vec{y}_B) & cos\sphericalangle(\vec{z}_A,\vec{y}_B) \\ cos\sphericalangle(\vec{x}_A,\vec{z}_B) & cos\sphericalangle(\vec{y}_A,\vec{z}_B) & cos\sphericalangle(\vec{z}_A,\vec{z}_B) \end{pmatrix} = \begin{pmatrix} \vec{x}_A\cdot\vec{x}_B & \vec{y}_A\cdot\vec{x}_B & \vec{z}_A\cdot\vec{x}_B \\ \vec{x}_A\cdot\vec{y}_B & \vec{y}_A\cdot\vec{y}_B & \vec{z}_A\cdot\vec{y}_B \\ \vec{x}_A\cdot\vec{z}_B & \vec{y}_A\cdot\vec{z}_B & \vec{z}_A\cdot\vec{z}_B \end{pmatrix}$$

Ο πίνακας κατεύθυνσης μεταφοράς $T_{B<-A}$ είναι ένας ορθογώνιος πίνακας καθώς τα διανύσματα βάσης του A και B είναι ορθογώνια μοναδιαία διανύσματα. Ως εκ τούτου, η μεταφορά ενός πίνακα κατεύθυνσης μεταφοράς είναι το ίδιο με ένα

πίνακα κατεύθυνσης μεταφοράς που αναπαριστά την αντίστροφη μεταφορά. Συνεπώς ισχύει:

$$T^{T}_{B\leftarrow A} = T^{-1}_{B\leftarrow A} = T_{A\leftarrow B} = 1$$

και

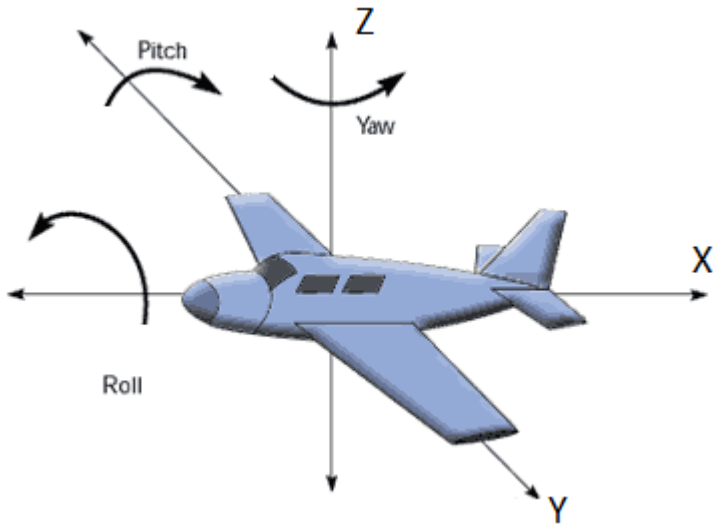
$$det(T_{A\leftarrow B}) = det(T_{B\leftarrow A})$$

Η μεταφορά ενός συστήματος συντεταγμένων γύρω από καθέναν από τα βασικά του διανύσματα για μια περιστροφή (και όχι μεταφορά!) κατά μια γωνία θ μπορεί να περιγραφεί από τον στοιχειώδη πίνακα μετασχηματισμού:

$$R^{x}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & sin\,\theta \\ 0 & -sin\,\theta & cos\,\theta \end{bmatrix} \quad R^{y}(\theta) = \begin{bmatrix} cos\,\theta & 0 & -sin\,\theta \\ 0 & 1 & 0 \\ sin\,\theta & 0 & cos\,\theta \end{bmatrix} \quad R^{z}(\theta) = \begin{bmatrix} cos\,\theta & sin\,\theta & 0 \\ -sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Γωνίες Όηλερ (Oiler Angles): Ο προσανατολισμός ενός στερεού σώματος σε σχέση με ένα σύστημα συντεταγμένων αναφοράς μπορεί να περιγράφει από τρεις πετυχημένες μεταφορές γύρω από τους σταθερούς του άξονες.

Οι τρείς γωνίες που χρησιμοποιούνται για τις πετυχημένες μεταφορές είναι η γωνίες Όηλερ. Συνήθως χρησιμοποιούνται για γραφικές απεικονίσεις του προσανατολισμού αεροσκαφών.



Εικόνα 3. Οι γωνίες του Όηλερ

Η γωνία pitch ή αλλιώς elevation καθορίζετε από την περιστροφή του αεροπλάνου γύρω από τον άξονα των Y (εικόνα 3). Θα μπορούσαμε να φανταστούμε την γωνία pitch να αλλάζει κατά την απογείωση αλλά και κατά την προσγείωση του αεροπλάνου[ii].

Η γωνία roll καθορίζετε από την περιστροφή του αεροπλάνου γύρω από τον άξονα των X. Θα μπορούσαμε αυτή τη φορά να φανταστούμε την γωνία roll να αλλάζει καθώς ο πιλότος πραγματοποιεί ελιγμούς κατά την διάρκεια της πτήσης.

Τέλος η γωνία yaw καθορίζετε από την οριζόντια αλλαγή της κατεύθυνση του αεροπλάνου όπως για παράδειγμα η αλλαγή κατεύθυνσης όσο αυτό βρίσκετε στο έδαφος.

Οποιοσδήποτε σταθερός άξονας του στερεού σώματος (για παράδειγμα του αεροπλάνου) μπορεί να χρησιμοποιηθεί για την αρχική περιστροφή.

Η δεύτερη περιστροφή πρέπει να πραγματοποιηθεί γύρω από οποιονδήποτε από τους άξονες που δεν χρησιμοποιήθηκε κατά την πρώτη περιστροφή.

Η Τρίτη περιστροφή μπορεί να πραγματοποιηθεί σε οποιονδήποτε από τους δύο άξονες που δεν χρησιμοποιήθηκε κατά την δεύτερη περιστροφή.

Έτσι προκύπτουν δώδεκα διαφορετικές διαδοχικές μεταφορές (transformation sequences) για να περιγράψουν την στάση του στερεού σώματος.

Οι πίνακες μεταφοράς (transformation matrices) για τις σειρές μεταφοράς προκύπτουν από τον πολλαπλασιασμό των τριών στοιχειωδών πινάκων μετασχηματισμού.

Για παράδειγμα έστω ότι θέλουμε να περιγράψουμε την ακολουθία μεταφοράς (Z -> X -> Y) ώστε να αναπαραστήσει τον προσανατολισμό ενός συστήματος συντεταγμένων B (με διανύσματα βάσης [XB, YB, ZB]) σε σχέση με το σύστημα συντεταγμένων A (με διανύσματα βάσης A [XA,YA,ZA]).

- Ο Πρώτος μετασχηματισμός γύρω από τον βασικό άξονα Z κατά γωνία Θz ($R^Z\theta(Z)$) θα μετασχηματίσει το A στο A’.
- Ο δεύτερος μετασχηματισμός γύρω από τον βασικό άξονα X κατά γωνία Θx ($R^X\theta(X)$) θα μετασχηματίσει το A’ σε A’’.
- Ο τρίτος μετασχηματισμός γύρω από τον βασικό άξονα Y κατά γωνία Θy κατά γωνία Θy ($R^Y\theta(Y)$) θα μετασχηματίσει το A’’ σε B.

Οι τρείς γωνίες Θx Θy και Θz ονομάζονται γωνίες του Οηλερ. Τα A’ και A’’ είναι τα ενδιάμεσα συστήματα συντεταγμένων με διανύσματα βάσης [$X_{A'}$, $Y_{A'}$,$Z_{A'}$] και [$X_{A''}$, $Y_{A''}$,$Z_{A''}$], αντίστοιχα. Η σειρά μεταφορών μπορεί να πολλαπλασιαστεί ως εξής:

$$B = R^y(\theta_y) \cdot A'' = R^y(\theta_y) \cdot R^x(\theta_x) \cdot A' = R^y(\theta_y) \cdot R^x(\theta_x) \cdot R^z(\theta_z) \cdot A \quad (2)$$

Σημειώστε ότι οι πίνακες έχουν πολλαπλασιαστεί διαδοχικά ξεκινώντας από αριστερά, και έτσι ο πρώτος πίνακας μεταφοράς $R^Z\theta(Z)$ έρχεται τελευταίος.

Αντικαθιστώντας την σχέση (1) στην σχέση (2) προκύπτει:

$$R_{B\leftarrow A}^{zxy} = \begin{bmatrix} -sin\theta_x sin\theta_y sin\theta_z + cos\theta_x cos\theta_z & sin\theta_x sin\theta_y cos\theta_z + cos\theta_x sin\theta_z & -sin\theta_x cos\theta_y \\ -cos\theta_y sin\theta_z & cos\theta_y cos\theta_z & sin\theta_y \\ cos\theta_x sin\theta_y sin\theta_z + sin\theta_x cos\theta_z & -cos\theta_x sin\theta_y cos\theta_z + sin\theta_x sin\theta_z & cos\theta_x cos\theta_y \end{bmatrix}$$

Άξονες και γωνία (Axis and angle): Το θεώρημα περιστροφής του Όηλερ δηλώνει επίσης ότι οποιαδήποτε περιστροφή γύρω από τους τρεις άξονες του συστήματος συντεταγμένων μπορεί να εκφραστεί ως μια μόνο περιστροφή γύρω από έναν καινούργιο άξονα[4] .

Η αναπαράσταση μιας περιστροφής με την χρήση της μεθόδου «Άξονες και περιστροφή» βασίζετε στην παραμετροποίηση ενός ζεύγους τιμών. Η πρώτη παράμετρος είναι ένα μοναδιαίο διάνυσμα που αναπαριστά τον άξονα περιστροφής ενώ η δεύτερη παράμετρος είναι η γωνία περιστροφής γύρω από αυτόν τον άξονα.

Κουατέρνια (quaternions) : Η αναπαράσταση του σχετικού προσανατολισμού με την χρήση των γωνιών του Όηλερ είναι εύκολο να αναπτυχτεί και να οπτικοποιηθεί όμως απαιτεί αρκετή υπολογιστική ισχύς.

Η ευρέως χρησιμοποιουμένη αναπαράσταση με την χρήση των κουατερνιονς βασίζετε στο θεώρημα περιστροφής του Όηλερ το οποίο δηλώνει ότι ο σχετικός προσανατολισμός δυο συστημάτων συντεταγμένων μπορεί να περιγραφεί από μια μόνο περιστροφή γύρω από έναν σταθερό άξονα.

Ένα κουατερνιον είναι ένας πίνακας με διαστάσεις 4x1, του οποίου τα στοιχεία αποτελούνται από δύο μέρη. Το ένα μέρος ονομάζετε μέρος πραγματικών αριθμών (scalar part) και το συμβολίζουμε με το s, ενώ το άλλο μέρος ονομάζετε διανυσματικό μέρος (vector part) ή φανταστικό μέρος (Imaginary part) και το συμβολίζουμε με το $\vec{u}$. Σημειώστε ότι το scalar part είναι το πρώτο στοιχείο του πίνακα.

$$q = \begin{bmatrix} s \\ \vec{v} \end{bmatrix} = \begin{bmatrix} s \\ v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix}$$

Όπως αναφέρθηκε παραπάνω, σύμφωνα με το θεώρημα περιστροφής του Όηλερ ένα κουατερνιο καθορίζετε από ένα άξονα περιστροφής (rotation axis)

και μια γωνία περιστροφής (rotation angle). Ένα κουατερνιο που αναπαριστά μια μεταφορά συντεταγμένης από ένα σύστημα A σε ένα σύστημα B, $q_{B<-A}$ ορίζετε στην παρακάτω εξίσωση

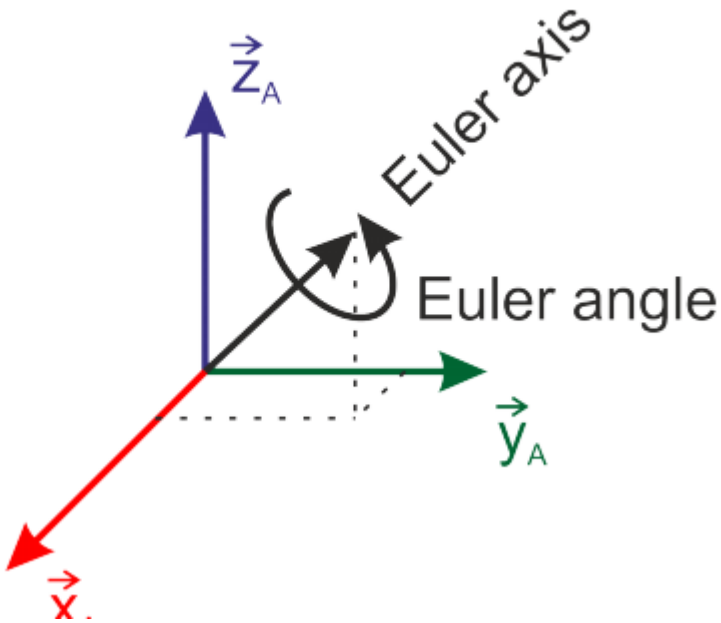
$$q = \begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \|\vec{e}\| \cdot \sin\frac{\theta}{2} \end{bmatrix}$$

Όπου ||e|| είναι ο ομαλοποιημένος άξονας περιστροφής και θ η γωνία μεταφοράς και όχι η γωνία περιστροφής.

Κάθε περιστροφή ή μεταφορά μπορεί να εκφραστεί από δύο κουατέρνια q και $\bar{q}$ όπου:

$$\bar{q} = \begin{bmatrix} -q_s \\ -q_x \\ -q_y \\ -q_z \end{bmatrix} = \begin{bmatrix} \cos\frac{2\pi-\theta}{2} \\ \|-\vec{e}\| \cdot \sin\frac{2\pi-\theta}{2} \end{bmatrix} = \begin{bmatrix} \cos(\pi - \frac{\theta}{2}) \\ \|-\vec{e}\| \cdot \sin(\pi - \frac{\theta}{2}) \end{bmatrix}$$

Αυτό μπορεί να γίνει εμφανές, αν φανταστούμε το $\bar{q}$ ως μια περιστροφή (ή μεταφορά) κατά γωνία 2π – θ και αντίθετο άξονα $-\vec{e}$, ως προς το q. Έτσι προκύπτει μια χαρτογράφηση 1:2 για ένα σχετικό προσανατολισμό και την αναπαράσταση του ως ένα κουατερνιο.

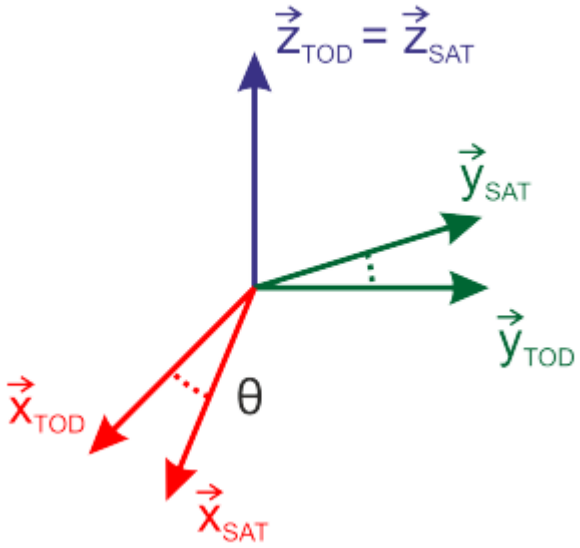


Εικόνα 8. Η ιδέα του θεωρήματος περιστροφής του Όηλερ σε ένα κουατερνιο

Για παράδειγμα ας υποθέσουμε ότι το σύστημα συντεταγμένων TOD (εικόνα 9) είναι το σταθερό πλαίσιο αναφοράς και το σύστημα συντεταγμένων SAT το σταθερό πλαίσιο του σώματος. Το SAT περιστρέφετε κατά μια γωνία θ γύρων από τον άξονα $\vec{Z}_{TOD}$ = [0 0 1] κατά γωνία θ = 30°. Έτσι η γωνία μεταφοράς είναι -30° αντιστοίχως. Η στάση του SAT ως προς το TOD δίνετε από το παρακάτω κουατερνιο:

$$q_{SAT \leftarrow TOD} = \begin{bmatrix} \cos\frac{-\theta}{2} \\ \|\vec{z}_A\| \cdot \sin\frac{-\theta}{2} \end{bmatrix} = \begin{bmatrix} 0.9659 \\ 0 \\ 0 \\ -0.2588 \end{bmatrix}$$

Σχετικά με το αλγεβρικό πρόσημο του θ, αν μια θετική τιμή περιγράφει των αξόνων του SAT γύρω από το TOD, η μεταφορά ενός διανύσματος από το TOD στο SAT υπολογίζετε μέσω μιας αρνητικής τιμής. Όπως αποδείχτηκε παραπάνω η περιστροφή και η μεταφορά είναι αντίθετες μεταξύ τους.



Εικόνα 9. Κατασκευή ενός μετασχηματισμού κουατερνιου

Η σημειογραφία των κουρτενιονς μπορεί να διαφέρει μεταξύ αναφορών και γλωσσών προγραμματισμού όσον αφορά την ακολουθία στοιχείων των διανυσμάτων και του πρόσημου των γωνιών. Για παράδειγμα η αεροδιαστημική εργαλειοθήκη στην γλωσσά προγραμματισμού MATLAB:

$$\mathrm{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T = \begin{bmatrix} \cos\frac{\theta}{2} \\ \|\vec{e}\| \cdot \sin\frac{\theta}{2} \end{bmatrix}$$

Όπου θ δεν είναι η γωνία μεταφοράς αλλά η γωνία περιστροφής.

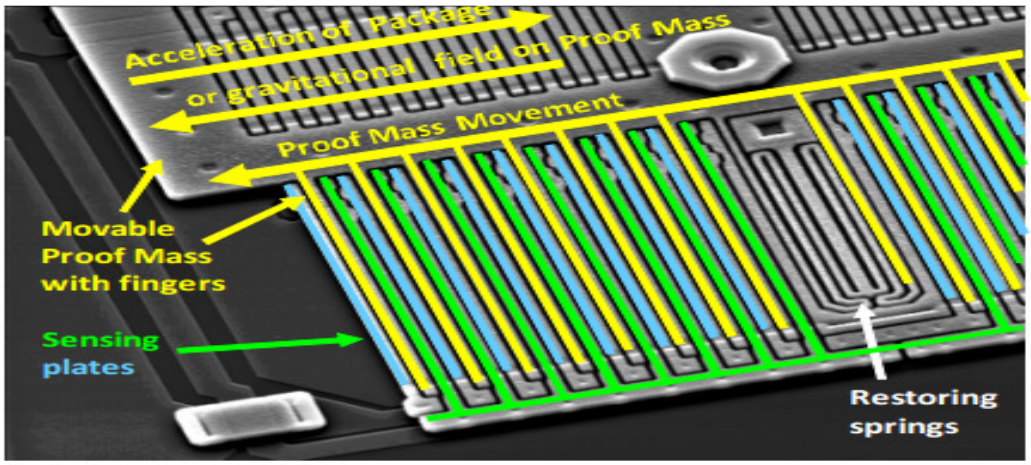
4.2 Οι έξοδοι των MEMS αισθητήρων επιτάχυνσης υπό την βαρύτητα και την επιτάχυνση

Οι MEMS αισθητήρες επιταχύνσεις (όπως ο MMA8451Q που έχει χρησιμοποιηθεί στην παρούσα διπλωματική εργασία) μπορούν να ανιχνεύουν την διάφορα μεταξύ της γραμμικής επιτάχυνσης του αισθητήρα καθώς επίσης και την επιτάχυνση της βαρύτητας της γης.

Τα data sheet του καθενός αισθητήρα επιτάχυνσης δηλώνουν ποιοι είναι οι θετικοί x, y και z άξονες επιτάχυνσης στο πακέτο του αισθητήρα και , από σύμβασης, αυτοί έχουν προκαθοριστεί έτσι ώστε μια επιτάχυνση που ασκείτε κατά μήκος και με κατεύθυνση προς αυτούς τους άξονες να δίνει ως έξοδο μια θετική τιμή.

Μια συνιστώσα του βαρυτικού πεδίου ευθυγραμμισμένη κατά μήκος του ίδιου άξονα θα δώσει ,παρόλα αυτά, ένα αρνητικό «διάβασμα» ή αλλιώς μια αρνητική τιμή στον αισθητήρα επιτάχυνσης.

Τόσο στην περίπτωση όπου στον αισθητήρα ασκείτε ένα βαρύτιμο πεδίο προς τα αριστερά όσο και στην περίπτωση που του ασκηθεί μια γραμμική επιτάχυνση προς τα δεξιά, η δοκιμαστική κινούμενη μάζα (movable roof mass) θα παρεκτραπεί προς την αριστερή κατεύθυνση (εικόνα 3).



Εικόνα 3. MEMS αισθητήρας επιτάχυνσης

Η εκτροπή της δοκιμαστικής μάζας μετριέται από την αλλαγή στη χωρητικότητα μεταξύ των δακτύλων της δοκιμαστικής μάζας (proof mass)και των δακτύλων των πιάτων ευαισθησίας (sensing plates).

Ένα κύκλωμα εσωτερικά του αισθητήρα μετατρέπει την μικροσκοπική χωρητικότητα σε ένα σήμα τάσης, το οποίο στους ψηφιακούς αισθητήρες επιτάχυνσης, ψηφιοποιείτε και μεταφέρετε ως ψηφιακή λέξη μέσω ενός σειριακού διαύλου (serial bus).

Εικόνα 4. Απλοποιημένο μοντέλο μετατροπέα

4.3 Αναγνώριση του προσανατολίσου μιας συσκευής με ενσωματωμένο έναν αισθητήρα επιτάχυνσης

Προκειμένου να υπολογιστεί ο προσανατολισμός (orientation) μιας συσκευής με βάση το βαρυτικό πεδίο της γης, η σύμβαση που θα υιοθετηθεί είναι ότι ο αισθητήρας επιτάχυνσης που έχει ενσωματωθεί στην συσκευή θα δίνει ως έξοδο 1g σε οποιονδήποτε άξονα είναι ευθυγραμμισμένος με το βαρυτικό πεδίο της γης και με κατεύθυνση προς τα κάτω.

Με βάση την παραπάνω σύμβαση , ένας τριών αξόνων αισθητήρας επιτάχυνσης, τοποθετημένος σε μια συσκευή που έχει προσανατολισμό προς το βαρυτικό πεδίο της γης και στην οποία ασκείτε μια γραμμική επιτάχυνση ar θα παράγει μια έξοδο:

$$\boldsymbol{G_p} = \begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix} = \boldsymbol{R}(\boldsymbol{g} - \boldsymbol{a_r})$$

Εικόνα 5. Μαθηματική αναπαράσταση της εξόδου του αισθητήρα επιτάχυνσης

Όπου R συμβολίζετε ο πίνακας περιστροφής ο οποίος περιγράφει τον προσανατολισμό της συσκευής σε σχέση πάντα με το πλαίσιο συντεταγμένων της γης, ενώ τα Gpx, Gpy και Gpz συμβολίζουν τις μετρήσεις του αισθητήρα στους X, Y και Z άξονες, αντίστοιχα.

Προκειμένου να είναι εφικτός ο προσδιορισμός του προσανατολισμού της συσκευής θα πρέπει να γίνουν δυο παραδοχές.

Η Πρώτη παραδοχή είναι να μην ασκείτε γραμμική επιτάχυνση στον αισθητήρα επιτάχυνσης ar ≈ 0. Με τον όρο γραμμική επιτάχυνση εννοείτε η κίνηση (και όχι η περιστροφή) της συσκευής σε ένα ή περισσότερους άξονες.

Αυτή η προϋπόθεση χρησιμεύει για την επίλυση της παραπάνω σχέσης (εικόνα 5) ως προς τον πίνακα περιστροφής R , και ως συνέπεια οποιαδήποτε γραμμική επιτάχυνση προέρχεται είτε από handshake είτε λόγω θορύβου είτε από άλλες αιτίες θα προσθέσει λάθη στον υπολογισμό του προσανατολισμού.

Η δεύτερη παραδοχή είναι ότι ο αρχικός προσανατολισμός της συσκευής είναι επίπεδος με το βαρυτικό πεδίο της γης (flat στο τραπέζι), ευθυγραμμισμένος με τον άξονα Z.

Με αυτές τις επιπλέον υποθέσεις, οι έξοδοι από το τον αισθητήρα επιτάχυνσης της συσκευής (μετρημένες στην φυσική μονάδα της επιτάχυνσης g) θα προκύπτουν από την παρακάτω μαθηματική σχέση:

$$\boldsymbol{G_p} = \begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix} = \boldsymbol{Rg} = \boldsymbol{R}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Ο προσανατολισμός μια συσκευής μπορεί να προσδιοριστεί από τις roll, pitch και yaw γωνίες περιστροφής από μια αρχική θέση. Ακολουθούν οι πίνακες περιστροφής οι οποίοι μετατρέπουν ένα διάνυσμα (όπως αυτό του βαρυτικού πεδίου της γης) γύρω από ένα σύστημα συντεταγμένων κατά γωνίες: φ για την περιστροφή roll, θ για την περιστροφή pitch και ψ για την περιστροφή yaw γύρο από τους άξονες x, y, z, αντίστοιχα.

$$\boldsymbol{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}$$

$$\boldsymbol{R}_y(\theta) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$\boldsymbol{R}_z(\psi) = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Υπάρχου έξι πιθανές διατάξεις αυτών των τριών πινάκων περιστροφής και, κατ αρχήν, τις θεωρούμε όλες εξίσου έγκυρες. Ο σύνθετος πίνακας περιστροφής R(ο οποίος αποτελείται από τους Rx, Ry και Rz) εξαρτάται από την σειρά με την οποία εφαρμόζονται οι περιστροφές roll, pitch και yaw.

Είναι διδακτικό να υπολογίσουμε τους έξι πιθανούς πίνακες περιστροφής R και να καθορίσουμε την επίδραση που θα έχουν από το βαρυτικό πεδίο της γης (του 1g) το οποίο είναι ευθυγραμμισμένο προς τα κάτω με τον άξονα Z.

$$\boldsymbol{R}_{xyz}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \text{(A)}$$

$$= \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi & \cos\theta\sin\phi \\ \cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \text{(B)}$$

$$= \begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \quad \text{(Γ)}$$

εικόνα 6. Πίνακας περιστροφής για την ακολουθίας (Yaw -> Roll -> Pitch)

$$R_{yxz}\begin{pmatrix}0\\0\\1\end{pmatrix} = R_y(\theta)R_x(\phi)R_z(\psi)\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (\Delta)$$

$$= \begin{pmatrix} \cos\psi\cos\theta - \sin\theta\sin\phi\sin\psi & \sin\psi\cos\theta + \sin\theta\sin\phi\cos\psi & -\sin\theta\cos\phi \\ -\cos\phi\sin\psi & \cos\phi\cos\psi & \sin\phi \\ \cos\theta\sin\phi\sin\psi + \sin\theta\cos\psi & -\cos\psi\cos\theta\sin\phi + \sin\psi\sin\theta & \cos\theta\cos\phi \end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (E)$$

$$= \begin{pmatrix} -\sin\theta\cos\phi \\ \sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \quad (\Sigma T)$$

Εικόνα 7. πινάκας περιστροφής της ακολουθίας (Yaw -> Pitch -> Roll)

$$R_{xzy}\begin{pmatrix}0\\0\\1\end{pmatrix} = R_x(\phi)R_z(\psi)R_y(\theta)\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (Z)$$

$$= \begin{pmatrix} \cos\theta\cos\psi & \sin\psi & -\cos\psi\sin\theta \\ -\cos\phi\cos\theta\sin\psi + \sin\phi\sin\theta & \cos\phi\cos\psi & \cos\theta\sin\phi + \cos\phi\sin\theta\sin\psi \\ \cos\theta\sin\psi\sin\phi + \cos\phi\sin\theta & -\cos\psi\sin\phi & \cos\theta\cos\phi - \sin\theta\sin\phi\sin\psi \end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (H)$$

$$= \begin{pmatrix} -\cos\psi\sin\theta \\ \cos\theta\sin\phi + \cos\phi\sin\psi\sin\theta \\ \cos\phi\cos\theta - \sin\theta\sin\phi\sin\psi \end{pmatrix} \quad (I)$$

$$R_{yzx}\begin{pmatrix}0\\0\\1\end{pmatrix} = R_y(\theta)R_z(\psi)R_x(\phi)\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (IA)$$

$$= \begin{pmatrix} \cos\psi\cos\theta & \cos\phi\cos\theta\sin\psi + \sin\theta\sin\phi & \cos\theta\sin\phi\sin\psi - \sin\theta\cos\phi \\ -\sin\psi & \cos\phi\cos\psi & \cos\psi\sin\phi \\ \cos\psi\sin\theta & -\cos\theta\sin\phi + \cos\phi\sin\psi\sin\theta & \cos\theta\cos\phi + \sin\theta\sin\phi\sin\psi \end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (IB)$$

$$= \begin{pmatrix} \cos\theta\sin\phi\sin\psi - \cos\phi\sin\theta \\ \cos\psi\sin\phi \\ \cos\theta\cos\phi + \sin\theta\sin\phi\sin\psi \end{pmatrix} \quad (I\Gamma)$$

$$\boldsymbol{R}_{zxy}\begin{pmatrix}0\\0\\1\end{pmatrix} = \boldsymbol{R}_z(\psi)\boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (\text{IΔ})$$

$$= \begin{pmatrix} \cos\psi\cos\theta + \sin\theta\sin\phi\sin\psi & \cos\phi\sin\psi & \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi \\ -\cos\theta\sin\psi + \cos\psi\sin\phi\sin\theta & \cos\phi\cos\psi & \cos\psi\cos\theta\sin\phi + \sin\theta\sin\psi \\ \cos\phi\sin\theta & -\sin\phi & \cos\theta\cos\phi \end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (\text{IE})$$

$$= \begin{pmatrix} \cos\theta\sin\phi\sin\psi - \cos\psi\sin\theta \\ \cos\psi\cos\theta\sin\phi + \sin\theta\sin\psi \\ \cos\theta\cos\phi \end{pmatrix} \quad (\text{IΣT})$$

$$\boldsymbol{R}_{zyx}\begin{pmatrix}0\\0\\1\end{pmatrix} = \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi)\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (\text{IZ})$$

$$= \begin{pmatrix} \cos\psi\cos\theta & \cos\phi\sin\psi + \cos\psi\sin\phi\sin\theta & \sin\phi\sin\psi - \cos\phi\cos\psi\sin\theta \\ -\cos\theta\sin\psi & \cos\psi\cos\phi - \sin\theta\sin\phi\sin\psi & \cos\psi\sin\phi + \cos\phi\sin\psi\sin\theta \\ \sin\theta & -\cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (\text{IH})$$

$$= \begin{pmatrix} \sin\phi\sin\psi - \cos\phi\cos\psi\sin\theta \\ \cos\psi\sin\phi + \cos\phi\sin\psi\sin\theta \\ \cos\theta\cos\phi \end{pmatrix} \quad (\text{IΘ})$$

Μπορεί εύκολα να γίνει αντιληπτό από τις παραπάνω εξισώσεις ότι οι έξι σύνθετοι πίνακες περιστροφής και οι έξι τιμές από το μετρημένο διάνυσμα της βαρύτητας είναι μεταξύ τους διαφορετικές.

Ως συνέπεια διαπιστώνετε ότι οι roll, pitch και yaw γωνίες περιστροφής δεν έχουν νόημα εάν πρωτίστως δεν έχει διευκρινιστεί η σειρά με την οποία αυτές οι περιστροφές έχουν εφαρμοστεί.

Οι τέσσερις αλληλουχίες από την εξίσωση (Z) έως και την (ΙΘ) μπορούν απ ευθείας να απορριφτούν ως ακατάλληλες για να καθορίσουν τον προσανατολισμό της συσκευής.

Οι έξοδοι του αισθητήρα επιτάχυνσης αποτελούνται από τρείς συνιστώσες όμως, από την στιγμή κατά την οποία το μέτρο του διανύσματος πρέπει πάντα να ισούται με 1g απόντος της γραμμικής επιτάχυνσης, έχει μόλις δυο βαθμούς ελευθερίας (degrees of freedom).

Μπορούμε να φανταστούμε το διάνυσμα του αισθητήρα επιτάχυνσης να βρίσκετε στην επιφάνεια μιας σφαίρας με ακτίνα 1g.

Επομένως δεν είναι δυνατόν να λυθεί για τρείς μοναδικές τιμές των roll, pitch και yaw γωνιών.

Οι τέσσερις τελευταίες αλληλουχίες στις παραπάνω εξισώσεις έχουν ως αποτέλεσμα οι έξοδοι από τον αισθητήρα επιτάχυνσης να αποτελούν συνάρτηση και των τριών γωνιών περιστροφής και ως εκ τούτου να μην είναι δυνατόν να επιλυθούν.

Αντίθετος, οι δυο πρώτες ακολουθίες περιστροφής από την εξίσωση (Α) έως την (ΣΤ), εξαρτώνται μόνο από τις roll και pitch γωνίες και επομένως μπορούν και να επιλυθούν.

Η έλλειψη οποιασδήποτε εξάρτησης από την yaw γωνία περιστροφής γίνετε εύκολα αντιληπτή καθώς η πρώτη περιστροφή γίνετε γύρο από τον άξονα Z ο οποίος είναι αρχικά ευθυγραμμισμένος με το βαρυτικό πεδίο και δείχνει προς τα κάτω.

Όλοι οι αισθητήρες επιτάχυνσης είναι εντελώς «αναίσθητοι» στο να υπολογίζουν περιστροφές γύρο από το διάνυσμα του βαρυτικού πεδίου, δηλαδή την γωνία περιστροφής yaw ή αλλιώς azimuth, και δεν μπορούν να χρησιμοποιηθούν για να υπολογίσουν μια τέτοια περιστροφή.

Είναι συμβατικό επομένως να επιλέξουμε είτε την ακολουθία περιστροφής Rxyz είτε την ακολουθία Ryxz ώστε να εξαλείψουμε την yaw περιστροφής γύρω από την ψ γωνία, και να δώσουμε λύσεις στις roll (Χ) και pitch (Υ) γωνίες.

Η άγνωστη γωνία yaw αντιπροσωπεύει την περιστροφή της συσκευής γύρο από τον βορρά και για τον προσδιορισμό της απαιτείτε η ενσωμάτωση ενός μαγνητόμετρου ώστε να δημιουργηθεί μια ψηφιακή πυξίδα (eCompass).

Για να επιλύσουμε την εξίσωσης (Γ) που προκύπτει από την ακολουθία Rxyz, αρχικά μπορούμε ξαναγράψουμε την εξίσωση με την μορφή της εξίσωσης (ΚΑ), ώστε να δημιουργήσουμε ένα συσχετισμό μεταξύ των roll φ και pitch θ γωνιών και των ομαλοποιημένων μετρήσεων του αισθητήρα επιτάχυνσης Gp.

$$\frac{\boldsymbol{G}_p}{\|\boldsymbol{G}_p\|} = \begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \Rightarrow \frac{1}{\sqrt{G_{px}^{\ 2} + G_{py}^{\ 2} + G_{pz}^{\ 2}}} \begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix} = \begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \quad (\text{ΚΑ})$$

Λύνοντας την εξίσωση (ΚΑ) ως προς τις roll και pitch γωνίες και χρησιμοποιώντας τον xyz δείκτη για να υποδηλώσουμε ότι οι roll και pitch γωνίες έχουν υπολογιστεί σύμφωνα με την ακολουθία περιστροφής Rxyz (δηλαδή η πρώτη περιστροφή είναι η roll και μετέπειτα η pitch) καταλήγουμε στις τελικές εξισώσεις (ΚΒ) και (ΚΓ) υπολογισμού των δυο γωνιών.

$$\tan\phi_{xyz} = \left(\frac{G_{py}}{G_{pz}}\right) \qquad \text{(ΚΒ)}$$

$$\tan\theta_{xyz} = \left(\frac{-G_{px}}{G_{py}\sin\phi + G_{pz}\cos\phi}\right) = \frac{-G_{px}}{\sqrt{G_{py}^{\,2} + G_{pz}^{\,2}}} \qquad \text{(ΚΓ)}$$

Η ακολουθία περιστροφής Rxyz χρησιμοποιείτε ευρέως στην αεροδιαστημική βιομηχανία και ονομάζετε αεροδιαστημική ακολουθία περιστροφής (aerospace rotation sequence).

Ομοίως η εξίσωση (ΣΤ) μπορεί να επιλυθεί για οποιεσδήποτε μετρήσεις του αισθητήρα επιτάχυνσης Gp.

$$\frac{\boldsymbol{G}_p}{\|\boldsymbol{G}_p\|} = \begin{pmatrix} -\sin\theta\cos\phi \\ \sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \Rightarrow \frac{1}{\sqrt{G_{px}^{\,2} + G_{py}^{\,2} + G_{pz}^{\,2}}} \begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix} = \begin{pmatrix} -\sin\theta\cos\phi \\ \sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \qquad \text{(ΚΔ)}$$

Λύνοντας την εξίσωση (ΚΔ) ως προς τις roll και pitch γωνίες και χρησιμοποιώντας τον yxz δείκτη για να υποδηλώσουμε ότι οι γωνίες έχουν υπολογιστεί σύμφωνα με την ακολουθία περιστροφής Ryxz (πρώτα pitch και μετά roll) προκύπτουν οι εξισώσεις (ΚΕ) και (ΚΣΤ) για τον υπολογισμό των δυο γωνιών.

$$\tan\phi_{yxz} = \frac{G_{py}}{\sqrt{G_{px}^{\,2} + G_{pz}^{\,2}}} \qquad \text{(ΚΕ)}$$

$$\tan\theta_{yxz} = \left(\frac{-G_{px}}{G_{pz}}\right) \qquad \text{(ΚΣΤ)}$$

Οι τέσσερις εξισώσεις (ΚΒ), (ΚΓ), (ΚΕ) και (ΚΣΤ) παρόλα αυτά δίνουν διαφορετικό αποτέλεσμα των roll και pitch γωνιών για τα ίδια «διαβάσματα» του αισθητήρα επιτάχυνσης.

Όπως αναφέρθηκε και νωρίτερα αυτό είναι απλός μια επίπτωση του γεγονότος ότι οι πίνακες περιστροφής δεν μετατρέπονται. Η σειρά των περιστροφών είναι σημαντικός παράγοντας και πρέπει πάντα να διευκρινίζετε όταν γίνετε αναφορά σε συγκεκριμένες γωνίες προσανατολισμού.

Η επόμενη δυσκολία που πρέπει να ξεπεραστεί είναι ότι οι γωνίες roll φ και pitch θ στις εξισώσεις (ΚΒ),(ΚΓ) και (ΚΕ),(ΚΣΤ) έχουν άπειρο αριθμό από λύσεις στα πολλαπλάσια των 360^{o}.

Ο περιορισμός των γωνιών roll και pitch μεταξύ των -180^{o} και 180^{o} βοηθάει κάπως όμως η επόμενη παράγραφος φανερώνει ότι αυτός ο περιορισμός, ακόμη οδηγεί σε δυο μοναδικές λύσεις για τις roll και pitch γωνίες.

Αξιολογώντας την εξίσωση (Γ) για pitch γωνίες π-θ και roll γωνίες φ+π και εφαρμόζοντας τυπικές ιδιότητες τριγωνομετρίας παρατηρούμε ότι οι μετρήσεις του αισθητήρα επιτάχυνσης είναι ίδιες με εκείνες που προκύπτουν από τις περιστροφές θ και φ.

$$\begin{pmatrix} -sin(\pi-\theta) \\ cos(\pi-\theta)sin(\phi+\pi) \\ cos(\pi-\theta)cos(\phi+\pi) \end{pmatrix} = \begin{pmatrix} -sin\,\theta \\ cos\,\theta sin\,\phi \\ cos\,\theta cos\,\phi \end{pmatrix}$$

Ομοίως αξιολογώντας την εξίσωση (ΣΤ) για pitch γωνίες θ+π και roll γωνίες π-φ επίσης παρατηρούμε ότι οι μετρήσεις του αισθητήρα επιτάχυνσης είναι ίδιες με αυτές που θα εμφανίζονταν για περιστροφές κατά θ και φ.

$$\begin{pmatrix} -sin(\theta+\pi)cos(\pi-\phi) \\ sin(\pi-\phi) \\ cos(\theta+\pi)cos(\pi-\phi) \end{pmatrix} = \begin{pmatrix} -sin\,\theta cos\,\phi \\ sin\,\phi \\ cos\,\theta cos\,\phi \end{pmatrix}$$

Η τελική λύση είναι να περιορίσουμε είτε την roll είτε την pitch γωνία (όχι και τις δυο) ώστε να βρίσκετε μεταξύ -90^o και 90^o.

Η σύμβαση που ακλουθείτε στην αεροδιαστημική ακολουθία περιστροφής αναφέρει ότι η pitch γωνία περιορίζετε μεταξύ -90^o και 90^o με αποτέλεσμα να εξαλειφθεί η μια από τις δυο λύσεις της, ενώ η roll γωνία έχει εύρος μεταξύ -180^o και 180^o.

Στην πράξη αυτό δεν αποτελεί πρόβλημα για μια εφαρμογή σε λογισμικό καθώς χρησιμοποιείτε η μαθηματική συνάρτηση ATAN2 η οποία επιστρέφει την ζητούμενη γωνία στο σωστό τεταρτημόριο βασιζόμενη στο πρόσημο των δυο ορισμάτων που της διοχετεύονται [iii].

4.4 Βασική τεχνική για την αναγνώριση του σημείου κρούσης μεταξύ ενός αισθητήρα επιτάχυνσης και ενός στερεού σώματος

Μέσω της επεξεργασίας των τιμών εξόδου ενός αισθητήρα επιτάχυνσης είναι δυνατή η αναγνώριση της πρόσκρουσης μεταξύ της πλατφόρμας που περιέχει τον αισθητήρα, και ενός οποιουδήποτε άκαμπτου σώματος. Επιπλέον υπάρχει η δυνατότητα να προσδιοριστεί η περιοχή της πλατφόρμας η όποιο και ήρθε σε επαφή με το άκαμπτο σώμα.

Έτσι στο τελικό χρήστη θα παρέχετε πληροφορία τόσο για το αν συνέβη μια πρόσκρουση καθώς και σε ποιο σημείο (μέσω από ένα εύρος πιθανών περιοχών) της πλατφόρμας αυτή συνέβη.

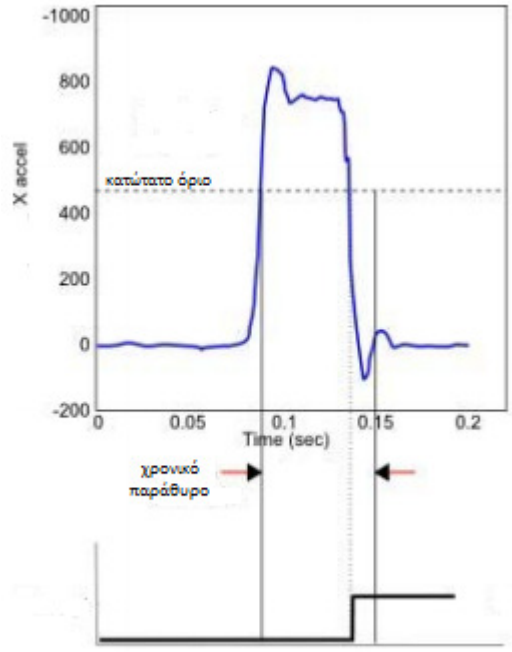
Μια απεικόνιση των τιμών εξόδου του αισθητήρα με την χρησιμοποίηση ενός ψηφιακού παλμογράφου θα μας βοηθήσει να κατανοήσουμε καλύτερα την ιδέα πίσω από την αναγνώριση ενός συμβάντος πρόσκρουσης.

Για να δημιουργηθεί η υποψία ύπαρξης μιας πρόσκρουσης θα πρέπει η μέτρηση της επιτάχυνσης του αισθητήρα σε έναν τουλάχιστον από τους τρείς βασικούς του άξονες να ξεπεράσει ένα κατώτατο όριο που έχει οριστεί από εμάς και είναι ουσιαστικά μια τιμή επιτάχυνσης αρκετά μεγάλη.

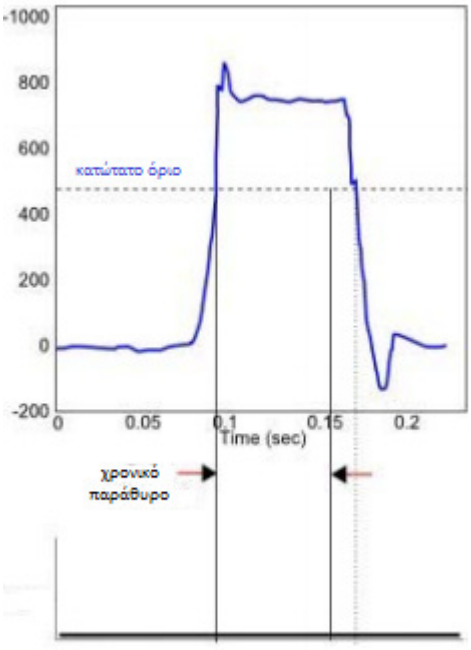
Την χρονική στιγμή κατά την οποία ο παλμός του ενός ή των δυο ή και των τριών αξόνων ξεπεράσουν αυτό το όριο (ή αλλιώς η τιμή της εξόδου του αισθητήρα επιτάχυνσης να ισούται με την τιμή που έχει τεθεί στο ανώτατο όριο) στιγμιαία πυροδοτείτε ένα ρολόι αντίστροφης μέτρησης.

Αν ο παλμός ενός τουλάχιστον άξονα έχει εξέρθει αυτού του κατώτατου ορίου προς την αντίθετη φορά, πριν από την ολοκλήρωση της αντίστροφης μέτρησης τότε αναγνωρίζετε επιτυχώς μια πρόσκρουση της πλατφόρμας του αισθητήρα με ένα άκαμπτο σώμα (εικόνα 8).

Στην περίπτωση κατά την οποία ο παλμός εξέρθει του κατώτατου ορίου μετά το πέρας του χρονικού παράθυρου τότε συμπεραίνουμε ότι η πλατφόρμα δεν έχει έρθει σε επαφή με κάποιο άκαμπτο σώμα και έτσι το συμβάν της πρόσκρουσης δεν αναγνωρίζετε (εικόνα 9).



Εικόνα 8. Προϋποθέσεις επιτυχημένης αναγνώρισης συμβάντος πρόσκρουσης



Εικόνα 9. Αποτυχημένη αναγνώριση συμβάντος πρόσκρουσης

Η πληροφορία για την περιοχή της πλατφόρμας που έχει συμμετάσχει στην πρόσκρουσης εξάγετε με βάση ποιος ή ποιοι από τους βασικούς άξονες, κατά την χρονική περίοδο της αντιστροφής μέτρησης ξεπέρασαν το ανώτατο όριο, αρχικά

προς μια κατεύθυνση (θετική ή αρνητική) και μετέπειτα εξήλθαν προς την αντίθετη κατεύθυνση.

Επιπλέον για τον ακριβέστερο προσδιορισμό της περιοχής πρόσκρουσης λαμβάνονται υπόψη και τα πρόσημα του ή των αξόνων που συγκρούστηκαν με το άκαμπτο σώμα.

Έτσι λοιπόν, διαχωρίζοντας την πλατφόρμα σε περιοχές με βάση τους άξονες και τα πρόσημα τους μπορούμε να αναγνωρίζουμε την περιοχή της πλατφόρμας που ήρθε σε επαφή με το άκαμπτο σώμα με βάση τα πρόσημα του ή των αξόνων που συμμετείχαν σε αυτήν.

5. ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΟΥ ΓΙΑ ΤΗΝ ΨΗΦΙΑΚΗ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΓΩΝΙΑΚΗΣ ΘΕΣΗΣ ΕΝΟΣ ΨΗΦΙΑΚΟΥ ΤΡΙΣΔΙΑΣΤΑΤΟΥ ΚΥΒΟΥ ΚΑΙ ΤΗΝ ΑΝΑΓΝΩΡΗΣΗ ΜΟΝΟΠΑΤΙΩΝ ΚΙΝΗΣΗΣ

5.1 Ανάπτυξη αλγορίθμου για την δημιουργία και την απεικόνιση ενός ψηφιακού κύβου ως σύνολο ψηφιακών σημείων στον τρισδιάστατο χώρο

Στην παρούσα διπλωματική εργασία έχει δημιουργηθεί ένας τρισδιάστατος κύβος ο οποίος προσομοιώσει τον γωνιακό προσανατολισμό της πλατφόρμας σε πραγματικό χρόνο. Ο κώδικας που έχει αναπτυχθεί στην ενσωματωμένη πλατφόρμα frdm KL25Z αποστέλλει τις τιμές των μετρήσεων του αισθητήρα επιτάχυνσης MMA8451Q (ο οποίος και βρίσκετε ενσωματωμένος στην πλατφόρμα) στον υπολογιστή γενικού σκοπού για περεταίρω επεξεργασία.

Το λογισμικό της python συλλέγει τις μετρήσεις του αισθητήρα επιτάχυνσης και στην συνέχεια με την χρήση της βιβλιοθήκης pygame έχει αναπτυχθεί το γραφικό περιβάλλον που χρησιμοποιείτε για την ψηφιακή απεικόνιση της κίνησης της πλατφόρμας.

Ο τρισδιάστατος κύβος προσομοιώνει την κίνηση της πλατφόρμας στις τρεις διαστάσεις ακλουθώντας την κίνηση της και καταγράφοντας με εικονικές ψηφιακές οθόνες (virtual lcd’s) τις γωνίες προσανατολισμού (pitch και roll).

Προκειμένου να προβάλουμε αλλά και να χειριστούμε ένα αντικείμενο στην οθόνη ενός γραφικού περιβάλλοντος, θα πρέπει πρωταρχικά να το περιγράψουμε με μαθηματικούς όρους.

Να περιγράψουμε δηλαδή μαθηματικά την ιδέα της δημιουργίας του τρισδιάστατου κύβου και μετέπειτα να τον εμφανίσουμε στο γραφικό περιβάλλον ενώ στο τελευταίο στάδιο θα πρέπει να του προσθέσουμε την λειτουργικότητα που θέλουμε.

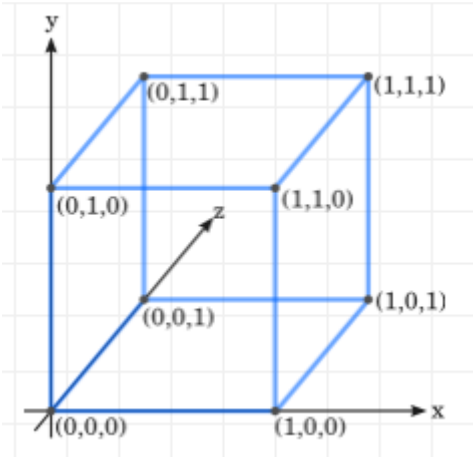
Ένας απλός τρόπος για να περιγράψουμε μαθηματικά έναν 3D κύβο, είναι με την χρήση μιας πλειάδας από κόμβους(nodes), από κορυφές (edges) καθώς και από προσόψεις(faces).

Οι κόμβοι (nodes) ορίζονται ως σημεία αποτελούμενα από τις τρείς συντεταγμένες (x, y, z) του χώρου. Οι κορυφές (edges) είναι ουσιαστικά οι γραμμές οι οποίες ενώνουν δυο κόμβους (nodes) μεταξύ τους, ενώ οι προσόψεις (faces) αποτελούν τις επιφάνειες οι οποίες περικλείονται από τις κορυφές. Ένας

τριών διαστάσεων κύβος αποτελείται από οκτώ κόμβους, από δώδεκα κορυφές και από έξι προσόψεις.

Για τον ορισμό των οκτώ κόμβων έχει αναπτυχθεί και χρησιμοποιηθεί μια κλάση Nodes η οποία και δημιουργεί αντικείμενα κόμβους. Το κάθε αντικείμενο κόμβος της κλάσης έχει τρία χαρακτηριστικά (x,y,z) τα οποία και αναφέρονται στις συντεταγμένες του κάθε κόμβου εντός του γραφικού περιβάλλοντος.

Έτσι έχουν δημιουργηθεί συνολικά οκτώ αντικείμενα της κλάσης nodes καθένα από τα οποία θα χρησιμοποιηθεί για να αναπαραστήσει έναν κόμβο του κύβου (εικόνα Α).



Εικόνα Α. Απεικόνιση των οκτώ κόμβων ενός τρισδιάστατου κύβου.

Καθένας από τους οκτώ κόμβους δέχεται ως είσοδο μια τριάδα τιμών με 0 ή 1 αναλόγως με την θέση του σε σχέση με τους βασικούς άξονες ενός συστήματος συντεταγμένων αναφοράς.

Ορισμένες φορές ο κύβος τοποθετείται στο μέσο των συντεταγμένων αναφοράς με αποτέλεσμα να προσδιορίζετε η θέση των κόμβων του με βάση μια τριάδα συντεταγμένων αποτελούμενη από -1 ή 1.

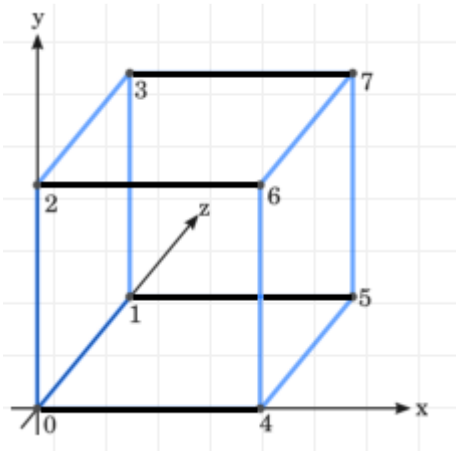
Με βάση λοιπόν τις συντεταγμένες x,y,z όταν κάποιος κόμβος βρίσκετε πάνω σε κάποιον ή σε κάποιους βασικούς άξονες παίρνει την τιμή 0 στον αντίστοιχο ή στους αντίστοιχους άξονες. Ενώ αν βρίσκετε σε διαφορετική θέση παίρνει την τιμή 1.

Για παράδειγμα ο κόμβος που βρίσκετε στην αρχή τον αξόνων έχει πάρει τις τιμές (0,0,0) για τους x,y,z άξονες, αντίστοιχα ενώ αυτός που έχει πάρει τις τιμές

(1,1,1) δεν βρίσκετε πάνω σε κανέναν από τους βασικούς άξονες αλλά στην «απέναντι θέση» από αυτούς.

Η περιγραφή της ιδέας καθώς και ο ορισμός των κορυφών (edges) είναι λίγο πιο απαιτητική διαδικασία. Ως μια πιο απλή προσέγγιση σε προγραμματιστικό επίπεδο θα μπορούσαμε να ομαδοποιήσουμε τις κορυφές ανάλογα με τον βασικό άξονα του κύβου που αυτές είναι παράλληλες.

Έτσι, για παράδειγμα, όπως φαίνετε από την (εικόνα Β) οι κορυφές που είναι παράλληλες στον άξονα των x ενώνουν μεταξύ τους τα ζεύγη των κόμβων (0,4), (1,5), (2,6), (3,7).



Εικόνα Β. οι τέσσερεις παράλληλες με τον άξονα x κορυφές

Στην κλάση που έχει δημιουργηθεί για της προσομοίωσης του κύβου έχει οριστεί μια λίστα η οποία περιέχει τις 12 κορυφές του κύβου που ενώνουν τον καθένα κόμβο με τους γειτονικούς του κόμβους.

Ουσιαστικά κάθε στοιχείο της λίστας των κορυφών δημιουργεί ένα αντικείμενο της κλάσης σημείων (Nodes) που αναφέρθηκε παραπάνω.

Για να ορίσουμε τις έξι προσόψεις του κύβου έχει οριστεί μια λίστα με το όνομα faces. Το κάθε στοιχείο της λίστας έχει δημιουργηθεί ώστε να περιέχει ουσιαστικά μια ακολουθία από τέσσερα γειτονικά μεταξύ τους στοιχεία της λίστα των κορυφών που αναφέρθηκε παραπάνω.

Πλέον έχουμε δημιουργήσει την ιδέα ενός κύβου. Το επόμενο βήμα που πρέπει να ακολουθηθεί είναι να τον απεικονίσουμε στην οθόνη γραφικών της

βιβλιοθήκης pygame της python. Για να συμβεί αυτό θα πρέπει να μετατρέψουμε τις τρισδιάστατες συντεταγμένων (x,y,z) σε δισδιάστατες συντεταγμένες (screen_x, screen_y). Η τεχνική της διασύνδεσης ενός τριών διαστάσεων συστήματος συντεταγμένων με ένα δυο διαστάσεων σύστημα συντεταγμένων ονομάζετε προβολή (projection). Γι αυτόν ακριβώς το σκοπό στο πρόγραμμα μας έχει αναπτυχθεί μια μέθοδος που ονομάζετε projection και ανήκει στην κλάσης nodes.

Για να κατανοήσουμε την προβολή του κύβου στο γραφικό περιβάλλον θα μπορούσαμε να φανταστούμε ότι φέγγουμε ένα φως στο πίσω μέρος του κύβου και εμείς κοιτάζουμε την σκιά που δημιουργείτε.

Στην πραγματικότητα, επειδή οι αμφιβληστροειδείς μας είναι ουσιαστικά 2D, το μόνο που βλέπουμε είναι οι προβολές αντικειμένων (αν και στερεοσκοπικές προβολές). Έτσι, για να ξεγελάσουμε το μυαλό μας ώστε να σκεφτεί ότι το σχήμα 2D στην οθόνη είναι στην πραγματικότητα 3D, πρέπει να δούμε τι θα σχηματίσουν τα 2D σχήματα στον αμφιβληστροειδή όταν προβάλλεται το 3D αντικείμενο.

Υπάρχουν πολλοί διαφορετικοί τρόποι προβολής ενός 3D αντικειμένου σε μια οθόνη που αντιστοιχούν στην προβολή του αντικειμένου από διαφορετικές γωνίες και προοπτικές. Η απλούστερη προβολή είναι να φανταστούμε ότι η οπτική επαφή μας με τον κύβο είναι παράλληλη ή αλλιώς κατά μήκος, με του άξονα z. Σε αυτή την περίπτωση, ο άξονας z δεν συνεισφέρει καμία πληροφορία σε αυτό που βλέπουμε και έτσι μπορούμε απλά να τον αγνοήσουμε.

5.2 Ανάπτυξη αλγορίθμου για την απεικόνιση της γωνιακής θέσης της πλατφόρμας (προσομοιωμένη ως ένας ψηφιακός κύβος) με την χρήση της αεροδιαστημικής ακολουθίας

Στην προηγούμενη ενότητα δημιουργήσαμε την μαθηματική περιγραφή του κύβου και στην συνεχεία δώσαμε υπόσταση στην ιδέα της περιγραφής του κύβου, εμφανίζοντας τον κύβο στο γραφικό μας περιβάλλον. Το τελευταίο στάδιο για ολοκληρωθεί η προσομοίωση της πλατφόρμας μέσω του κύβου είναι να του προσδώσουμε λειτουργικότητα με την χρήση φυσικά των πραγματικών μετρήσεων του αισθητήρα επιτάχυνσης.

Επιπλέον θα πρέπει με κάποια τεχνική να επαναπροσδιορίζουμε την θέση των κόμβων σε κάθε αλλαγή του προσανατολισμού και μάλιστα με τρόπο ώστε να μην χαλάει η κατανομή του σχήματος (οι μπροστινές προσόψεις θα πρέπει να καλύπτουν τις πίσω). Η κάθε περιστροφή (rotation) αλλάζει τις τιμές δυο συντεταγμένων (για παράδειγμα του x και του y) κατά μια συνάρτηση και των δυο αυτών συντεταγμένων. Αυτό σημαίνει ότι και οι τρείς συντεταγμένες αλληλεπιδρούν και έτσι η συνταγμένη z πλέον θα έχει σημαντικό ρόλο και θα εμφανίζετε στο γραφικό περιβάλλον καθώς θα επηρεάζει τις τιμές των συντεταγμένων x και y. Για την περιστροφή ενός κόμβου ή αλλιώς σημείου με συντεταγμένες (x,y,z) σε μια νέα θέση έχει αναπτυχθεί μια κλάση (rotator) η οποία κάνει χρήση των δυο μεθόδων RotateX, RotateY της κλάσης Nodes.

Η μέθοδος RotateX για παράδειγμα, δέχεται ως παράμετρο μια γωνία (angle) και με βάση το πυθαγόρειο θεώρημα δημιουργεί τον πίνακα περιστροφής (rotation matrix) υπό τον άξονα x. Ουσιαστικά υπολογίζει τις νέες συντεταγμένες (δηλαδή το νεο y και το νέο z του κόμβου) που θα βρεθεί το αντικείμενο κόμβος αν περιστραφεί ως προς τον άξονα X κατά μια γωνία angle.

Η περιστροφή του σημείου (x,y,z) ως προς τον άξονα τον X θα αφήσει ανεπηρέαστη την συντεταγμένη x καθώς η περιστροφή πραγματοποιείτε γύρω από τον άξονα X. Έτσι η μέθοδος RotateX για είσοδο (x,y,z) θα επιστρέψει ως έξοδο μια νέα θέση του κόμβου με συντεταγμένες (x,y’,z’). Την ίδια ακριβώς λειτουργικότητα έχει και η μέθοδος RotateY για την περιστροφή ενός κόμβου ως προς τον άξονα Y , υπολογίζονται φυσικά τα νέα x και z.

Όπως αναφέρθηκε παραπάνω μια πλήρη περιστροφή (rotation) περιλαμβάνει την περιστροφή όλων των σημείων αρχικά κατά μια γωνία ψ με σταθερό τον άξονα των z (την οποία την ονομάσαμε yaw), κατά μια γωνία φ (την οποία την ονομάσαμε roll) και κατά μια γωνία θ (την οποία την ονομάσαμε pitch).

Συνεπώς για να μπορέσει ο αλγόριθμος να υπολογίσει την νέα θέση του κύβου θα πρέπει αρχικά να προσδιορίσουμε την γωνία περιστροφής (angle) και

αφετέρου να πολλαπλασιάσουμε της μεθόδους RotateX, RotateY (δηλαδή τους πίνακες περιστροφής ως προς x και ως προς y) μεταξύ τους με την σειρά που θα πραγματοποιηθούν οι περιστροφές της πλατφόρμας, για καθέναν από τους οκτώ κόμβους.

Για κάθε διαφορετική ακολουθία που ακλουθείτε, ή αλλιώς σειρά που θα εφαρμοστούν οι περιστροφές όπως για παράδειγμα πρώτα ως προς x μετά ως προς y, προκύπτουν από τον πολλαπλασιασμό των πινάκων περιστροφής και διαφορετικές εξισώσεις υπολογισμού των γωνιών.

Η ακολουθία που ακλουθεί ο κώδικας που έχει αναπτυχθεί στην παρούσα διπλωματική εργασία ονομάζετε αεροδιαστημική ακολουθία και βασίζετε στην σειρά περιστροφής των σημείων: πρώτα γύρω από τον άξονα των z η οποία και δεν επηρεάζει τον προσδιορισμό του προσανατολισμού καθώς αρχικά η πλατφόρμα βρίσκετε παράλληλα ή επίπεδα πάνω στο τραπέζι, μετά γύρω από τον άξονα των x και τέλος γύρο από τον άξονα των y (yaw -> roll -> pitch).

Με βάση αυτήν την ακολουθία και τους πολλαπλασιασμούς των αντίστοιχων πινάκων περιστροφής προκύπτουν οι εξισώσεις υπολογισμού των γωνιών. Το πρόγραμμα υπολογίζει την νέα θέση του κάθε κόμβου με την παρακάτω εντολή:

Point_Orientation = RotateY(roll_angle)*RotateX(pitch_angle)

Οι δυο γωνίες περιστροφής που διοχετεύονται ως ορίσματα στις δυο μεθόδους περιστροφής υπολογίζονται βάση της μαθηματικής ανάπτυξης των πινάκων περιστροφή. Η roll γωνιά υπολογίζετε μέσω της συνάρτηση (math.atan2(-yacc,zacc)) ενώ για τον υπολογισμό της γωνία pitch χρησιμοποιείτε η εξίσωση (math.atan2(xacc, math.sqrt(yacc*yacc + zacc*zacc))). Οι μεταβλητές xacc, yacc και zacc αναπαριστούν τις μετρήσεις επιτάχυνσης του αισθητήρα στους άξονες x, y, z αντίστοιχα.

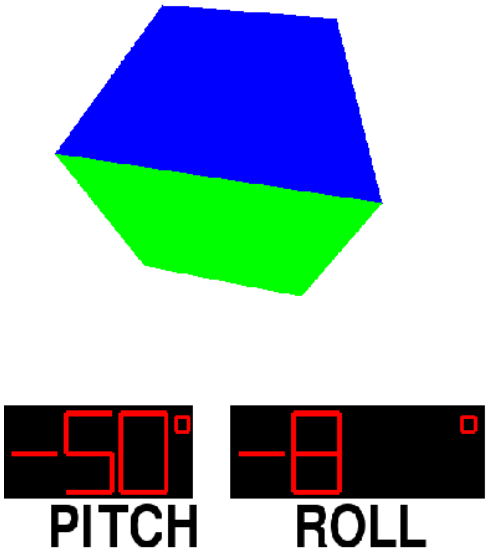
Στην συνέχεια καλείτε η μέθοδος project της κλάσης Nodes προκειμένου να μεταφέρει του κόμβους από τις τρείς στις δυο διαστάσεις και συνεπώς να γίνει δυνατή η «ορθολογική» αναπαράσταση του νέου προσανατολισμού του κύβου στην οθόνη των γραφικών. Η μέθοδος αυτή επιστρέφει αφενός τις συντεταγμένες των pixel εντός του γραφικού περιβάλλοντος όπου θα πρέπει να σχεδιαστεί ο κάθε κόμβος ως προς τους άξονες x και y καθώς και πληροφορία για την θέση του σημείου σε σχέση με τον άξονα z.

Φυσικά η μέθοδος project δέχεται ως όρισμα, την λίστα με τις νέες συντεταγμένες των κόμβων στις τρείς διαστάσεις που υπολογίστηκαν από την παραπάνω ακολουθία (αεροδιαστημική ακολουθία). Θα μπορούσαμε με βάση την λίστα με τις νέες συντεταγμένες να σχεδιάσουμε τον κύβο όμως το

πρόβλημα που θα δημιουργείτο είναι ότι οι πίσω προσόψεις θα κάλυπταν τις μπροστινές. Έτσι το τελευταίο ζήτημα που θα πρέπει να διευθετηθεί πριν την επιτυχή απεικόνιση της γωνιακής θέσης του κύβου είναι να επαναπροσδιοριστούν οι συντεταγμένες του κάθε κόμβου για κάθε πρόσοψη.

Στην συνέχεια του προγράμματος οι νέες συντεταγμένες των κόμβων στις δυο διαστάσεις 2D καταχωρούνται σε μια λίστα. Μετέπειτα χρησιμοποιώντας μια επανάληψη η οποία υπολογίζετε τη μέση τιμή της κάθε πρόσοψης ως προς τις τιμές του άξονα των z. Αύτη η διαδικασία μας παρέχει ουσιαστικά πληροφορία για την θέση της κάθε πρόσοψης του κύβου στο γραφικό περιβάλλον σε σχέση με τον άξονα των z.

Παρακάτω ταξινομούμε την λίστα με τις μέσες τιμές του άξονα z για κάθε πρόσοψη από την μικρότερη προς την μεγαλύτερη και υπολογίζουμε τις νέες συντεταγμένες των σημείων ως προς x και y για κάθε μια πρόσοψη από αυτήν με την μικρότερη μέση τιμή ως προς z έως αυτή με την μεγαλύτερη μέση τιμή ως προς z. Η λογική αυτή είναι η λογική που περιγράφετε στον αλγόριθμο του ζωγράφου (painter's algorithm)[iv] και μας εξασφαλίζει ότι οι μπροστινές προσόψεις, αυτές δηλαδή με την μεγαλύτερη μέση τιμή των τεσσάρων κορυφών τους, δεν θα υπερκαλύπτονται από τις πιο πίσω οι οποίες είναι και αυτές με την μικρότερη μέση τιμή πάντα ως προς τον άξονα z. Έτσι για την τελική απεικονίσει του προσανατολισμένου κύβου χρησιμοποιούμε τις τελικές συντεταγμένες x και y τις κάθε πρόσοψης.



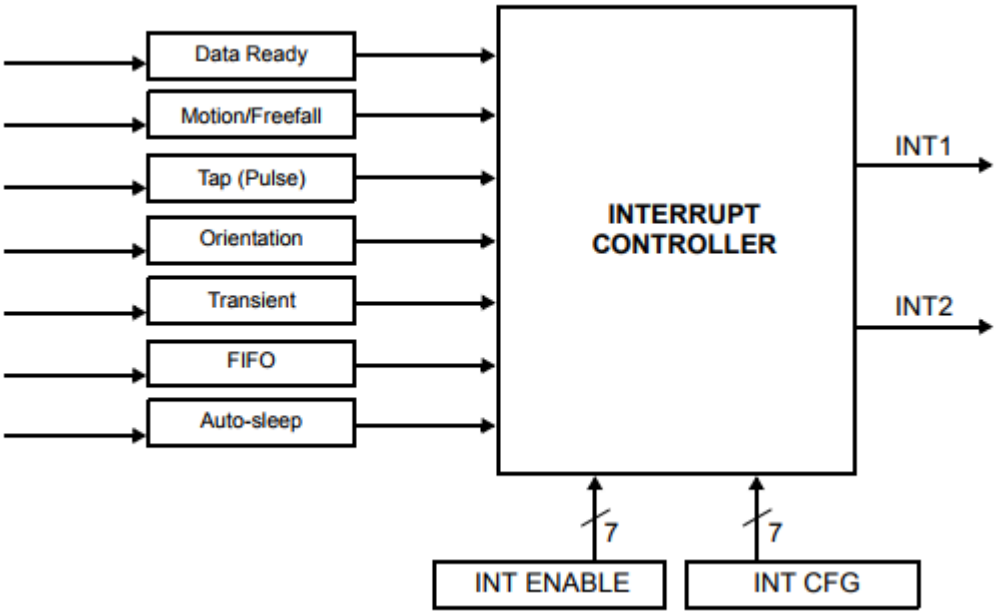
Εικόνα Γ. Στιγμιότυπο από το γραφικό περιβάλλον του ψηφιακό κύβο κατά την στιγμή όπου η γωνιακή του θέση είναι στις -50 κατά pitch και -8 κατά roll μοίρες.

5.3 Ανάπτυξη αλγορίθμου για την αναγνώριση της στιγμιαίας κρούσης μεταξύ πλατφόρμας και ενός άκαμπτου εξωτερικού σώματος

Ο αισθητήρας επιτάχυνσης MMA8451Q ο οποίος όπως έχει αναφερθεί παραπάνω βρίσκετε ενσωματωμένος στην πλατφόρμα frdm KL25Z διαθέτει επτά ενσωματωμένες συνάρτησης, τις οποίες μπορεί ο χρήστης να ενεργοποιήσει και να χρησιμοποιήσει για την ανάπτυξη εφαρμογών.

Συγκεκριμένα οι συναρτήσεις αυτές ή αλλιώς τα events που διατίθενται προς χρήση είναι τα εξής: Data Ready, Motion/Freefall, Tap (Pulse), Orientation, Transient, FIFO και Auto-SLEEP.

Ο αισθητήρας επιτάχυνσης MMA8451Q διαθέτει επτά παραμετροποιήσημα (confiqurable) interrupts. Αυτές οι επτά πηγές interrupts μπορούν να δρομολογηθούν σε δύο interrupt pins (εικόνα Γ1). Οι πηγές για το καθένα interrupt ξεχωριστά πρέπει να παραμετροποιηθούν καθώς και να ενεργοποιηθούν. Εάν η σημαία του interrupt υψωθεί λόγο της ανίχνευσης κάποιου event το αντίστοιχο interrupt pin, INT1 είτε INT2 θα “χτυπήσει” και συνήθως θα κληθεί μια συνάρτηση που θα εκτελέσει ένα μπλοκ κώδικα.



Εικόνα Γ1. Σχηματικό διάγραμμα του συστήματος γεννήτριας interrupts

Αρχικά πρέπει να γίνει γνωστό ότι ο κάθε register έχει χωρητικότητα μνήμης της τάξεως του ενός byte, αποτελείτε δηλαδή από 8 bits. Το πρώτο μας βήμα λοιπόν για να ενεργοποιηθεί το μονό tap ή αλλιώς pulse event είναι να ενεργοποιηθεί η ενσωματωμένη στον αισθητήρα συνάρτηση αναγνώρισης του μονού tap σε καθένα άξονα. Υπεύθυνος register για την λειτουργία της ενεργοποίησης του tap event σε κάθε άξονα είναι ο PULSE_CFG Register 0X21 (εικόνα Δ).

Table 1. Register 0x21 PULSE_CFG Register (Read/Write) and Description

Tap Enable	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	DPA	ELE	ZDPEFE	ZSPEFE	YDPEFE	YSPEFE	XDPEFE	XSPEFE
Single Tap	0	1	0	1	0	1	0	1
Double Tap	0	1	1	0	1	0	1	0
Both S & D	0	1	1	1	1	1	1	1

Xaxis Single Pulse Enable

ELE BIT: Latch Enable

Zaxis Single Pulse Enable

Yaxis Single Pulse Enable

εικόνα Δ. Ο register ώστε ενεργοποιηθούν οι άξονες για να αναγνωρίζετε το event.

Θέτοντας τον XSPEFE (bit 0) , τον YSPEFE (bit 2) και τον ZSPEFE (bit 3)σε λογικό ‘1’ ενεργοποιούμε την αναγνώριση μονού tap στους άξονες X, Y και Z, αντίστοιχα. Τα υπόλοιπα bits του register θα τεθούν σε λογικό ‘0’ και έτσι λοιπόν η τιμή του PULSE_CFG Register θα είναι 0b00010101 σε δυαδική μορφή ή 0x15 σε δεκαεξαδική μορφή. Αφού λοιπόν ενεργοποιήσαμε τους τρεις άξονες για να αναγνωρίζει το tap event το επόμενο βήμα είναι να αξιοποιηθούν οι μετρήσεις του αισθητήρα.

Αρχικά θα οριστεί ένα κατώτατο όριο στον άξονα X για να ανιχνεύει το event. Υπεύθυνος register για τον ορισμό του ορίου αναγνώρισης του tap event στον άξονα X είναι ο PULSE_THSX Register (0x23).

Table 2. Register 0x23 PULSE_THSX Register (Read/Write) Pulse Threshold Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	THSX6	THSX5	THSX4	THSX3	THSX2	THSX1	THSX0

1 - 127 Απόλυτες τιμές, όπου 1 = 0b00000001 και 127 = 0b01111111 = 0x7F

63 = 0b00111111 = 0x3F

Εικόνα Ε. Ορισμός κατώτατου ορίου για την αναγνώριση του event στον άξονα X.

Οι τιμές του PULSE_THSX Register κυμαίνονται μεταξύ του 1 – 127 (θετικοί ακέραιοι) για σταθερό εύρος επιτάχυνσης +- 8g (ανεξάρτητα αν εμείς έχουμε ορίσει δυναμικό εύρος (dynamic range)τα 2 ή τα 4 g ο PULSE_THSX Register λειτουργεί με βάση τα 8g) .

Κάθε step είναι 0.063g/LSB. Έτσι αν πολλαπλασιαστεί η τιμή που έχει τεθεί στον PULSE_THSX Register με την τιμή του κάθε step προκύπτει ως αποτέλεσμα το ανώτατο όριο σε μονάδες G (επιτάχυνσης της βαρύτητας). Για παράδειγμα αν ο PULSE_CFG Register έχει τεθεί σε 0x3F (0b00111111) τότε 63 * 0,063g = 4g.

Θέτοντας την ίδια τιμή ορίου και στους Register PULSE_THSY (0X24) και PULSE_THSZ (0X25) έχουν οριστεί τα κατώτατα όρια για τους άξονες Y και Z, αντίστοιχα.

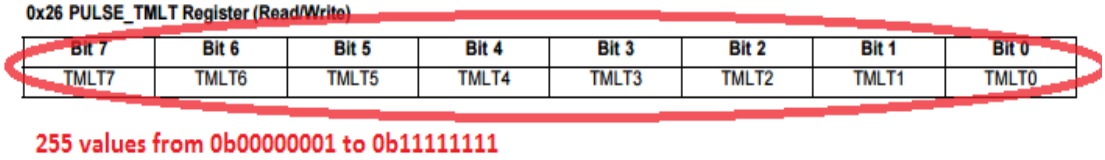
Επόμενο βήμα είναι ο ορισμός του χρονικού διαστήματος κατά το οποίο η τιμή της μέτρησης της επιτάχυνσης σε κάθε άξονα πρέπει να παραμείνει πάνω από το όριο των 4g που τέθηκε με την παραπάνω εντολή.

Επίσης εντός του ίδιου χρονικού διαστήματος η τιμή της μέτρησης θα πρέπει να πέσει κάτω από την τιμή του ορίου ώστε η αναγνώριση του tap event να θεωρηθεί έγκυρη.

Για να γίνει ο υπολογισμός του χρονικού παραθύρου πρέπει να ληφθεί υπόψη η χρονική διάρκεια κάθε βήματος (time step) η οποία είναι άρρηκτα συνδεδεμένη με τον ρυθμό εκροής δεδομένων (ODR) καθώς και με το Oversampling mode(στην περίπτωση μας είναι normal).

Η μέγιστη χρονική διάρκεια που μπορεί να έχει το παράθυρο υπολογίζετε από τον χρόνο βήματος παλμού (time step pulse) επί το 255.

Υπεύθυνος register για να τεθεί το χρονικό διάστημα ή αλλιώς το χρονικό παράθυρο είναι ο PULSE_TMLT (0x26).



0x26 PULSE_TMLT Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TMLT7	TMLT6	TMLT5	TMLT4	TMLT3	TMLT2	TMLT1	TMLT0

255 values from 0b00000001 to 0b11111111

Εικόνα ΣΤ. Ο Register που είναι υπεύθυνος για τον ορισμό του χρονικού παραθύρου

Τα βήματα(time steps) που είναι διαθέσιμα εξαρτώνται ή αλλιώς συνδέονται άμεσα με το oversampling mode καθώς και με το αν το χαμηλοπερατό φίλτρο (Low pass filter) είναι ενεργό ή όχι. Η χρονική διάρκεια του κάθε time step φαίνετε στον παρακάτω πίνακα.

Table 50. Time Step for PULSE Time Limit (Reg 0x0F) Pulse_LPF_EN = 0

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.159	0.159	0.159	0.159	0.625	0.625	0.625	0.625
400	0.159	0.159	0.159	0.319	0.625	0.625	0.625	1.25
200	0.319	0.319	0.159	0.638	1.25	1.25	0.625	2.5
100	0.638	0.638	0.159	1.28	2.5	2.5	0.625	5
50	1.28	1.28	0.159	2.55	5	5	0.625	10
12.5	1.28	5.1	0.159	10.2	5	20	0.625	40
6.25	1.28	5.1	0.159	10.2	5	20	0.625	40
1.56	1.28	5.1	0.159	10.2	5	20	0.625	40

Για ODR = 800Hz και Oversampling mode = Normal και LowPassFilter = 0 : time step = 0.625

εικόνα Ζ. Πίνακας με την χρονική διάρκεια κάθε βήματος.

Η τελευταία παραμετροποίηση που πρέπει να φροντιστεί είναι η χρονική καθυστέρηση η οποία ξεκινάει αμέσως μετά την αναγνώριση του event , και μέχρι το πέρας αυτής της χρονικής καθυστέρησης καμιά αναγνώριση tap event δηλαδή χτυπήματος δεν μπορεί να γίνει.

Υπεύθυνος register για να τεθεί το χρονικό διάστημα της καθυστέρησης είναι ο PULSE_LTCY(0x27).

0x27: PULSE_LTCY Pulse Latency Timer Register

0x27 PULSE_LTCY Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LTCY7	LTCY6	LTCY5	LTCY4	LTCY3	LTCY2	LTCY1	LTCY0

255 values from 1 = 0b00000001 to 255 = 0b11111111

Εικόνα Η. Register υπεύθυνος για την ελάχιστη χρονική διάρκεια μεταξύ δυο επιτυχημένων tap events.

Όπως και στην παραμετροποίηση του PULSE_TMLT Register έτσι και στον PULSE_LTCY Register, το βήμα χρόνου (time step) εξαρτάται από την τιμή του ODR, την κατάσταση του Oversampling Mode και αν είναι ενεργοποιημένο ή όχι το LPF.

Table 53. Time Step for PULSE Latency @ ODR and Power Mode (Reg 0x0F) Pulse_LPF_EN = 0

ODR (Hz)	Max Time Range (s)				Time Step (ms)			
	Normal	LPLN	HighRes	LP	Normal	LPLN	HighRes	LP
800	0.318	0.318	0.318	0.318	1.25	1.25	1.25	1.25
400	0.318	0.318	0.318	0.638	1.25	1.25	1.25	2.5
200	0.638	0.638	0.318	1.276	2.5	2.5	1.25	5
100	1.276	1.276	0.318	2.56	5	5	1.25	10
50	2.56	2.56	0.318	5.1	10	10	1.25	20
12.5	2.56	10.2	0.318	20.4	10	40	1.25	80
6.25	2.56	10.2	0.318	20.4	10	40	1.25	80
1.56	2.56	10.2	0.318	20.4	10	40	1.25	80

Με ODR = 800HZ, Oversampling Mode = Normal και LPF = 0 : Time Step = 1.25 ms

Εικόνα Θ. Πίνακας με την χρονική διάρκεια της καθυστέρηση (Latency)

Ο μέγιστος χρόνος που μπορεί να διαρκέσει η καθυστέρηση (Latency) είναι το ο χρόνος βήματος (time step) * 255. Για να τεθεί ένα παράθυρο καθυστέρησης λοιπόν της τάξεως των 300 ms θα διαιρεθεί με το time step, 200ms / 1.25ms = 160 counts ή 0XA0 σε δεκαεξαδική μορφή.

Αφού λοιπόν θέσαμε τις παραμέτρους για την επιτυχείς αναγνώριση του tap event, στο βασικό μας πρόγραμμα (δηλαδή στην main()) καλούμε μια ρουτίνα εξυπηρέτησης διακοπής (interrupt service routine). Η «δουλειά» αυτής της ρουτίνας είναι κάθε φορά που θα αναγνωρίζετε ένα επιτυχημένο tap event , να «κτυπάει» ένα interrupt στο αντίστοιχο interrupt pin (INT 1)που έχει οριστεί για την αναγνώριση του συγκεκριμένου event .

Τότε καλείτε μια συνάρτηση που έχει ονομαστεί SingleTapTrue η οποία και αλλάζει την τιμή μιας λογικής (boolean) RPC μεταβλητής με το όνομα TAP.

Ο κώδικας που μπορεί να γραφεί σε έναν οποιονδήποτε υπολογιστή γενικού σκοπού μπορεί να διαβάζει μέσω της RPC σειριακής σύνδεσης σε πραγματικό χρόνο

την τιμή της μεταβλητής TAP (η οποία αλλάζει λογική τιμή κάθε φορά που «χτυπάει» ένα interrupt)και να αναγνωρίζει την αλλαγή της τιμής από true σε false ή από false σε true και έτσι θα αναγνωρίζει την πρόσκρουση της πλατφόρμας με κάποιο άκαμπτο εξωτερικό σώμα.

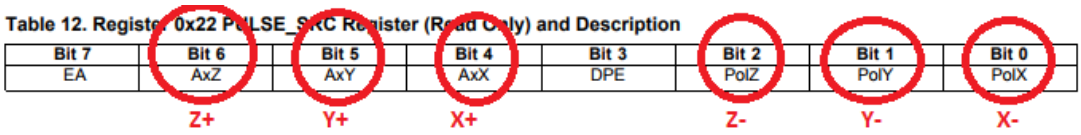
5.4 Ανάπτυξη αλγορίθμου για τον προσδιορισμό του σημείου κρούσης της πλατφόρμας με ένα άκαμπτο εξωτερικό σώμα

Προχωρώντας ένα βήμα παρακάτω από απλά την αναγνώριση μιας πρόσκρουσης της πλατφόρμας με ένα άκαμπτο εσωτερικό σώμα θα μπορούσαμε να προσδιορίσουμε και τους βασικούς άξονες του αισθητήρα επιτάχυνσης που ευθύνονται ή αλλιώς που εμπλέκονται στην πρόσκρουση.

Προεκτείνοντας λίγο ακόμα αυτή τη λογική είναι δυνατόν να προσδιοριστεί η περιοχή της πλατφόρμας (μέσα από ένα εύρος πιθανών περιοχών) που ήρθε σε επαφή με το άκαμπτο σώμα και συνεπώς προκάλεσε την πρόσκρουση.

Ο Pulse Source Register (0X22) όταν αναγνωριστεί ένα tap event ανυψώνει μια σημαία και μας παρέχει πληροφορία σχετικά με το ποιοι άξονες συμμετείχαν στην πρόκληση του tap event.

Ο PSR όπως και οι υπόλοιποι register έχει χωρητικότητα μνήμης της τάξεως των 8 bits εκ των οποίων τα 6 bit (εικόνα Ι) αναφέρονται στους έξι βασικούς άξονες επιτάχυνσης του αισθητήρα (εικόνα Κ). Όταν στον αντίστοιχο άξονα τον οποίο αναπαριστούν, αναγνωριστεί με επιτυχία ένα tap event τότε τίθενται σε λογικό ‘1’.



Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EA	AxZ	AxY	AxX	DPE	PolZ	PolY	PolX
	Z+	Y+	X+		Z-	Y-	X-

Εικόνα Ι. Register υπεύθυνος για τον καθορισμό των αξόνων στους ο οποίους προκλήθηκε η πρόσκρουση.

Το bit3 τίθεται πάντα σε λογικό ‘0’ που σημαίνει ότι η αναγνώριση του event της διπλής πρόσκρουση (double tap detection) είναι απενεργοποιημένη ενώ το bit7 έχει πάντα την λογική τιμή ‘1’.

Στην παρούσα διπλωματική έχει γραφεί μια επιπλέον συνάρτηση στην βασική βιβλιοθήκη του αισθητήρα mma8451q με όνομα get_source_reg_value, η οποία καλείτε από το βασικό πρόγραμμα και σκοπός της είναι να αποστέλλει το περιεχόμενο του PSR το οποίο και καταχωρείτε σε μια RPC μεταβλητή.

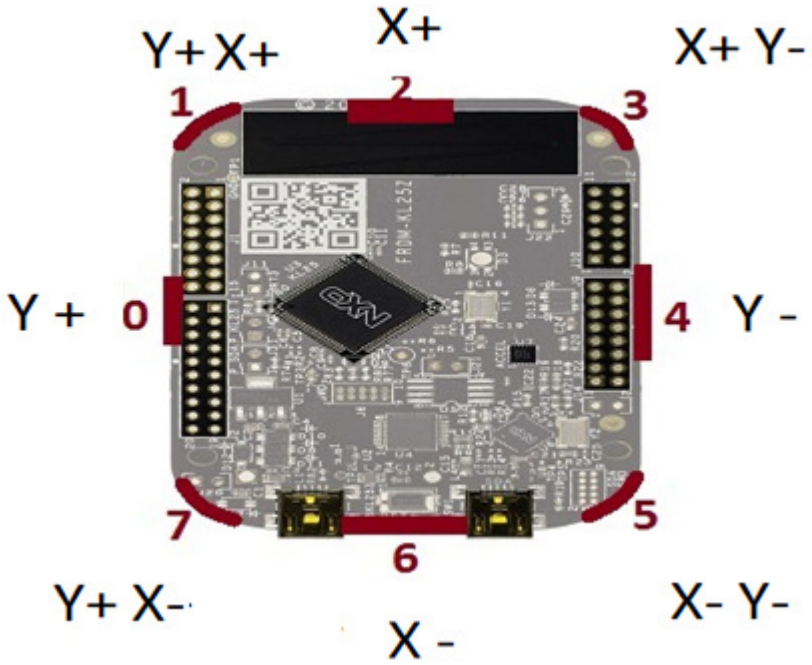
Η τιμή της RPC μεταβλητής μεταφέρεται μέσω της σειριακής θύρας στο προγραμματιστικό περιβάλλον που βρίσκετε στον υπολογιστή γενικού σκοπού,

όπως ακριβώς συνέβη και με τις προηγούμενες RPC μεταβλητές και ο κώδικας της python καταχωρεί τις τιμές των μετρήσεων σε μια μεταβλητή.

Αν η τιμή της μεταβλητής (σε δυαδική μορφή) φανερώνει την πρόσκρουση της πλατφόρμας στον y θετικό άξονα δηλαδή ισούται με ‘0b10100000’ τότε ο αλγόριθμος θέτει προς αναγνώριση το πρώτο μονοπάτι.

Αντίστοιχα εξετάζετε αν η τιμή του source register ισούται με την δυαδική τιμή ‘0b10100010’που φανερώνει ότι η σημαία υψώθηκε λόγο της πρόσκρουσης της πλατφόρμας στον αρνητικό άξονα y. Τότε ενεργοποιείτε το δεύτερο μονοπάτι προς αναγνώριση.

Επιλογή αναγνώρισης δεύτερου μονοπατιού με ένα απαλό χτύπημα στην αριστερή πλευρά του γαντιού.



Εικόνα Κ. Πιθανά σημεία πρόσκρουσης της πλατφόρμας με ένα άκαμπτο σώμα

5.5 Ανάπτυξη αλγορίθμου για την αναγνώριση συγκεκριμένων μονοπατιών(pattern recognition) με βάση την γωνιακή κίνηση της πλατφόρμας

Ο αλγόριθμος για την αναγνώριση δυο συγκεκριμένων μονοπατιών με βάση την γωνιακή κίνηση της πλατφόρμας βασίζετε στις τιμές των μετρήσεων του αισθητήρα επιτάχυνσης καθώς και στην πληροφορία που μεταφέρει ο source register σχετικά με την αναγνώριση του σημείου μιας πρόσκρουσης της πλατφόρμας.

Το πρώτο μονοπάτι έχει οριστεί ως μια ακολουθία περιστροφών με πολύ συγκεκριμένη σειρά κατά μια ορισμένη γωνία που την αποκαλούμε γωνία αναγνώρισης. Στον αλγόριθμο μπορούν να οριστούν ταυτόχρονα παραπάνω από μια γωνίες αναγνώρισης.

Για την επιτυχημένη ολοκλήρωση του πρώτου μονοπατιού το γάντι θα πρέπει αρχικά να ξεπεράσει την ορισμένη γωνία ως προς τον αρνητικό άξονα της pitch. Με άλλα λόγια να καταγραφεί από τον αλγόριθμο μια γωνία pitch με τιμή μεγαλύτερη από αυτή που ορίζει η γωνία αναγνώρισης. Στην συνέχεια θα πρέπει η τιμή της γωνία pitch να είναι μεγαλύτερη από την γωνία αναγνώρισης ως προς τον θετικό άξονα των x, τον άξονα δηλαδή που μετράει τις θετικές τιμές της γωνίας pitch. Μετέπειτα και εφαρμόζοντας την ίδια λογική, η τιμή της roll γωνίας θα πρέπει να ξεπεράσει την γωνία αναγνώρισης ως προς τον θετικό άξονα και τέλος ως προς τον αρνητικό άξονα.

Η ακολουθία για να θεωρηθεί ως έγκυρη θα πρέπει αφενός η γωνιακή κίνηση του γαντιού να πραγματοποιηθεί αυστηρά με την συγκεκριμένη σειρά που περιγράφει η ακολουθία και αφετέρου να εκτελεστεί εντός ενός συγκεκριμένου χρονικού παραθύρου (time window) το οποίο και έχει οριστεί στα 7 δευτερόλεπτα.

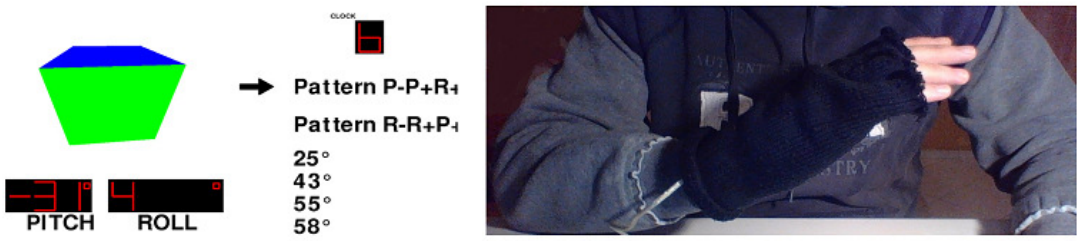
Για την δεύτερη ακολουθία έχει εφαρμοστεί η ίδια ακριβώς λογική με την διαφορά ότι έχει αλλαχθεί η σειρά με την οποία θα πρέπει το γάντι να περιστραφεί κατά τις τέσσερεις γωνίες περιστροφής.

Σε προγραμματιστικό επίπεδο, σε κάθε frame οι τιμές των μετρήσεων του αισθητήρα επιτάχυνσης μεταφράζονται στις αντίστοιχες γωνίες pitch και roll. Πρωταρχική συνθήκη για να ξεκινήσει η διαδικασία αναγνώρισης ή αλλιώς να ξεκινήσει το ρολόι την αντίστροφη μέτρηση των 7 δευτερόλεπτων είναι η γωνία pitch να γίνει μεγαλύτερη από την αρνητική τιμή των δεκαπέντε μοιρών. Τότε καλείται μια συνάρτηση η οποία και πυροδοτεί το χρονόμετρο. Ο αλγόριθμος στην συνέχεια με βάση το μονοπάτι που έχει επιλεχθεί (βάση της μέτρησης του source register) καλεί την αντίστοιχη συνάρτησης αναγνώρισης. Το κάθε μονοπάτι μπορεί να επιλεχτεί διαδραστικά με ένα απαλό χτύπημα του γαντιού στο αριστερό μέρος για την ενεργοποίηση του πρώτου μονοπατιού και στο δεξιό μέρος για την ενεργοποίηση του δεύτερου μονοπατιού.

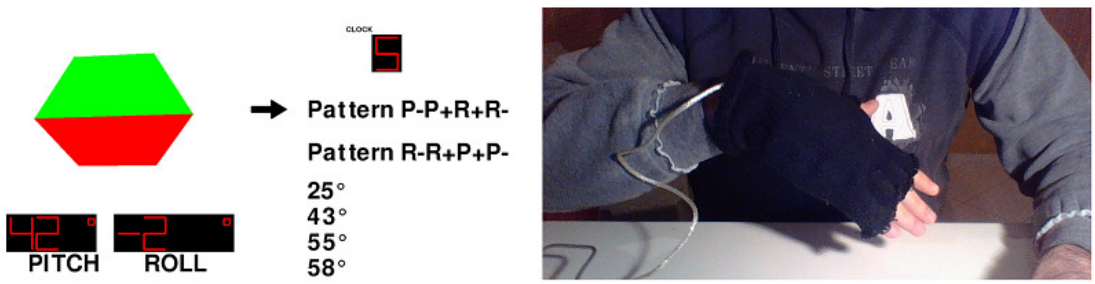
Η συνάρτηση αναγνώρισης εξετάζει τις τιμές των γωνιών pitch και roll. Αν η πρώτη περιστροφή επιτυγχάνετε για παράδειγμα με μια γωνιά μεγαλύτερη από 25 τότε θα εξετάσει αν η γωνία pitch είναι μεγαλύτερη από -25. Αν δεν είναι θα επιστρέψει την λειτουργικότητα του αλγόριθμου στο κυρίως πρόγραμμα για να ληφθεί μια νέα μέτρηση. Αν πάλι η μέτρηση ξεπερνάει την γωνία των -25 μοιρών τότε καταχωρεί σε μια λίστα τεσσάρων στοιχείων το αριθμό '1' στο τέταρτο στοιχείο της. Έτσι στο επόμενο frame γνωρίζει ότι το πρώτο από τα τέσσερα βήματα του μονοπατιού έχει πραγματοποιηθεί και πλέον εξετάζει την επόμενη συνθήκη.

Αν αναγνωρίσει με επιτυχία και τις τέσσερις συνθήκες εντός του χρονικού παραθύρου των 7 δευτερολέπτων τότε η συγκεκριμένη ακολουθία του γαντιού έχει αναγνωριστεί επιτυχώς και στο γραφικό περιβάλλον που έχει αναπτυχθεί εμφανίζετε μια ετικέτα που ενημερώνει σχετικά τον χρήστη.

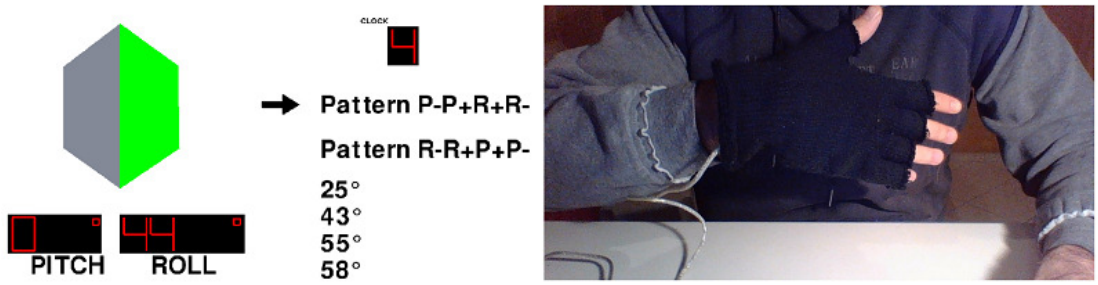
Στα παρακάτω τέσσερα καρέ απεικονίζονται τα τέσσερα στάδια που πρέπει να εκπληρωθούν ώστε να αναγνωριστεί επιτυχώς από τον αλγόριθμο το πρώτο μονοπάτι.



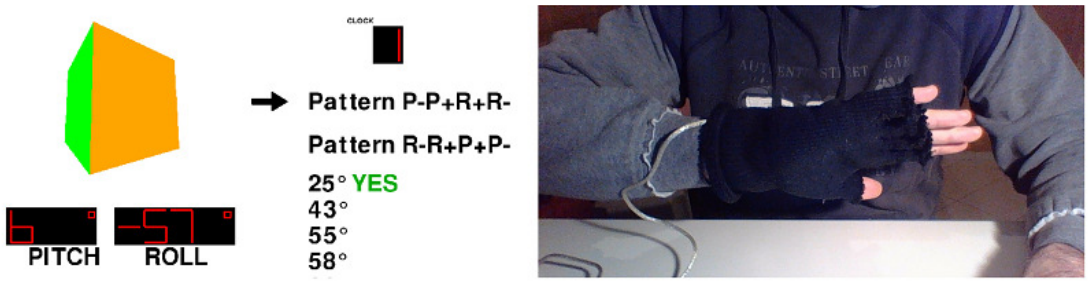
Εικόνα Α.Στιγμιότυπο από το ψηφιακό περιβάλλον την στιγμή οπού το ψηφιακό γάντι έχει ενεργοποίηση την αντίστροφη μέτρηση καθώς και έχει ολοκληρώσει την αναγνώριση της πρώτης συνθήκης του πρώτου pattern.



Εικόνα Β.Στιγμιότυπο από το ψηφιακό περιβάλλον την στιγμή οπού το ψηφιακό γάντι έχει ολοκληρώσει την αναγνώριση της δεύτερης συνθήκης του πρώτου pattern.

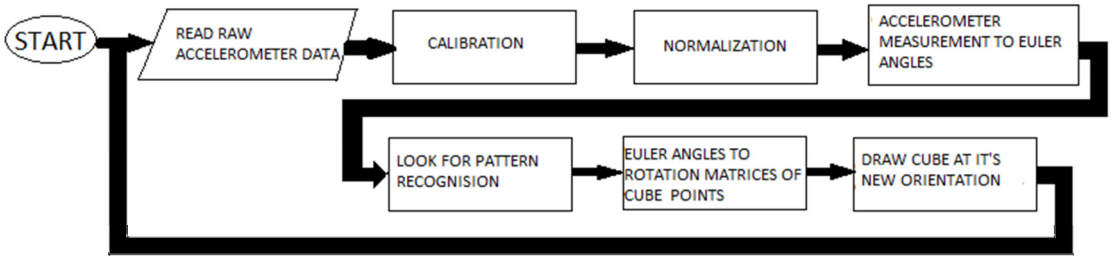


Εικόνα Γ. Στιγμιότυπο από το ψηφιακό περιβάλλον την στιγμή οπού το ψηφιακό γάντι έχει ολοκληρώσει την αναγνώριση της τρίτης συνθήκης του πρώτου pattern.

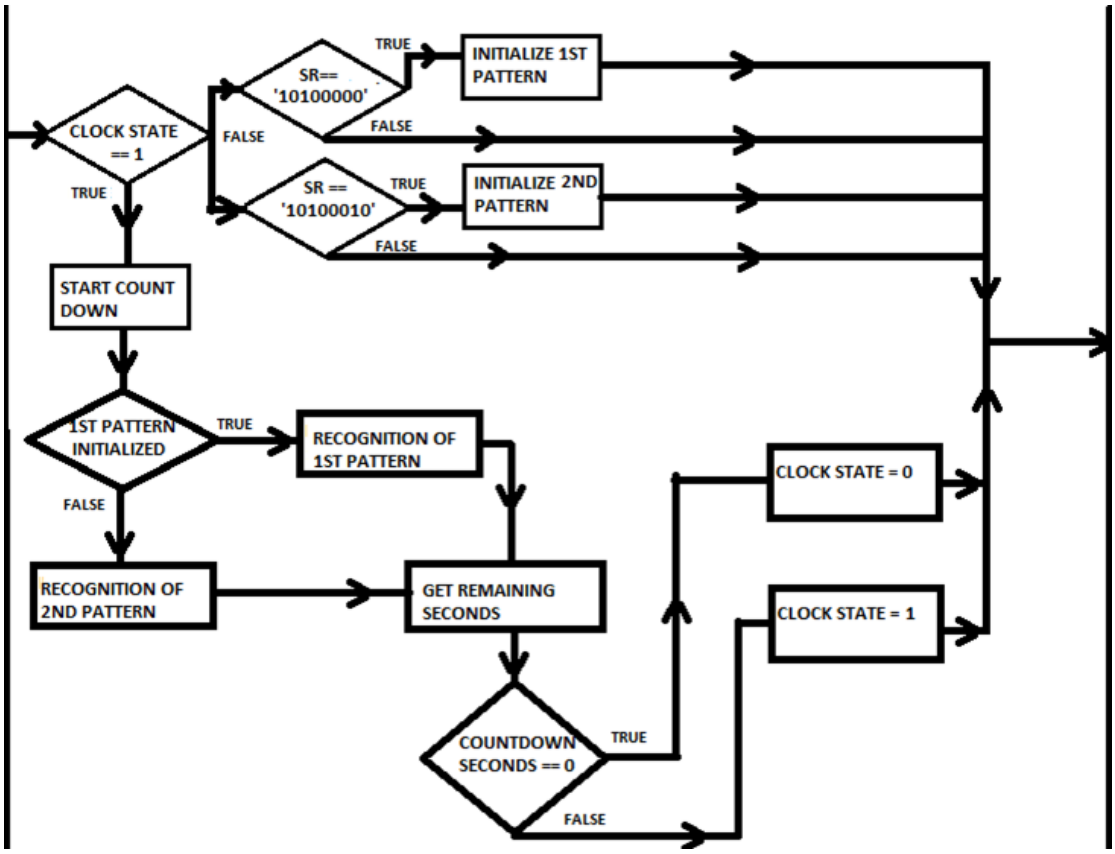


Εικόνα Δ. Στιγμιότυπο από το ψηφιακό περιβάλλον κατά την αναγνώριση του πρώτου pattern ως προς την γωνία 25 μοιρών και με εναπομείναντα χρόνο ενός δευτερολέπτου.

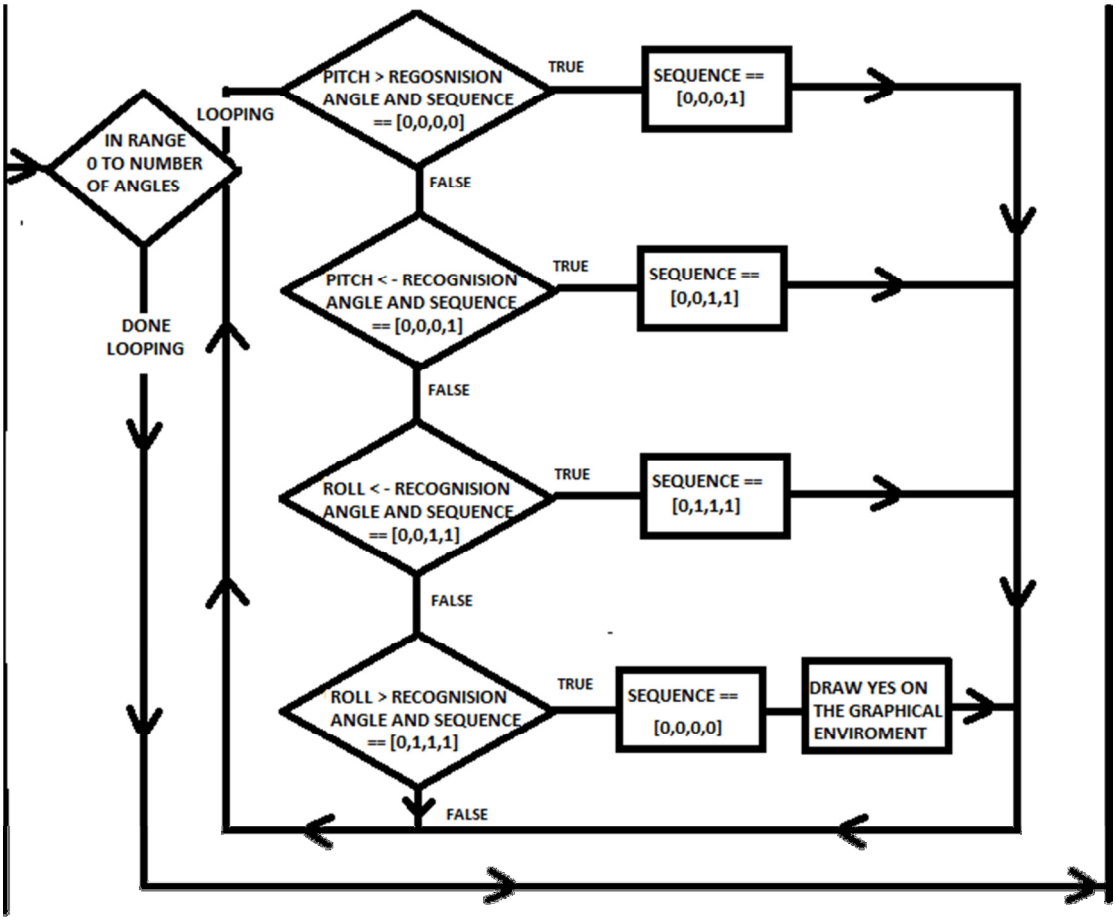
5.6 Διαγράμματα Ροής



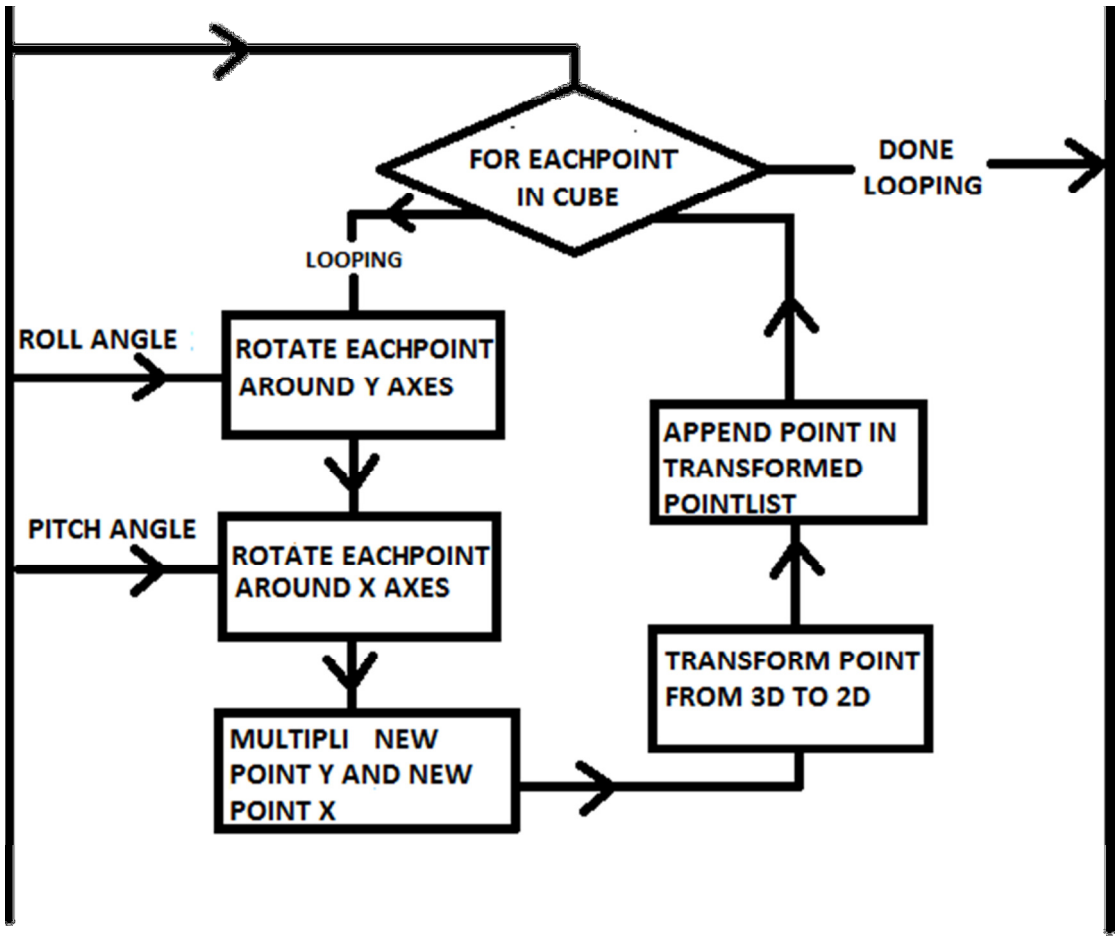
Εικόνα Ε. Διάγραμμα ροής της main μεθόδου της κλάσης cyberphysical glove.



Εικόνα Στ. Διάγραμμα ροής της συνάρτησης look for pattern recognition



Εικόνα Z. Διάγραμμα ροής της συνάρτησης Recognition of 1st pattern



ΕΙΚΟΝΑ Η. Διάγραμμα ροής της συνάρτησης Euler angles to rotation matrix of cube

6. ΑΞΙΟΛΟΓΗΣΗ ΤΗΣ ΑΠΟΔΟΣΗΣ ΤΗΣ ΠΛΑΤΦΟΡΜΑΣ ΩΣ ΨΗΦΙΑΚΟ ΚΛΙΣΙΜΕΤΡΟ

6.1 Αξιολόγηση απόδοσης γωνιών pitch

Με σκοπό την αξιολόγηση της απόδοσης του ψηφιακού κλισίμετρου που έχει δημιουργηθεί στην παρούσα διπλωματική εργασία έχουν πραγματοποιηθεί δοκιμές σε πραγματικές συνθήκες.

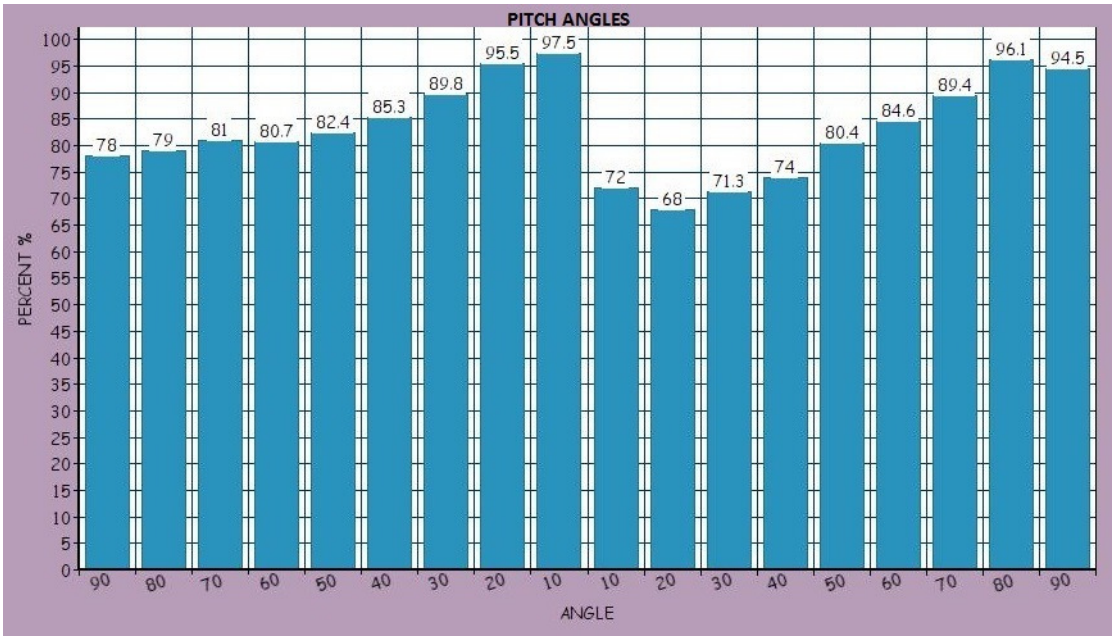
Ο πρώτος στόχος ήταν να αξιολογηθεί το κατά ποσό οι μετρήσεις (σε μοίρες) της γωνίας pitch που εξάγονται από τον αλγόριθμο που έχει αναπτυχθεί ανταποκρίνονται στις πραγματικές γωνίες κλίσης της πλατφόρμας.

Για τον σκοπό αυτό η πλατφόρμα τοποθετήθηκε πλάι σε ένα μοιρογνωμόνιο. Για κάθε μια γωνία μέσα στο ευρέος από -90^o έως 90^o στο οποίο και κυμαίνετε εξ ορισμού η γωνία pitch, πάρθηκαν οι μετρήσεις για κάθε μια γωνία ανά διάστημα 10 μοιρών. Σε κάθε μέτρηση ο αισθητήρας επιτάχυνσης παρέμενε ταυτισμένος με το μοιρογνωμόνιο υπό την μετρήσιμη κλίση του δείγματος για ένα λεπτό. Με το πέρας του ενός λεπτού το άθροισμα όλων των μετρήσεων της συγκεκριμένης γωνίας διαιρέθηκε με τον αριθμό των δειγμάτων. Το πείραμα επαναλήφθηκε για είκοσι φορές σε κάθε μια από τις γωνίες του δείγματος και τα αποτελέσματα που εξήχθηκαν παρουσιάζονται στον παρακάτω πίνακα (εικόνα Λ).

Γωνίες	Μετρήσεις
-10^o	-7.8^o
-20^o	-15.8 ^{o}
-30^o	-20.8 ^{o}
-40^o	-29.1 ^{o}
-50^o	-39.7 ^{o}
-60^o	-50.1 ^{o}
-70^o	-62.7 ^{o}
-80^o	-76.4 ^{o}
-90^o	-87.8 ^{o}
10^o	7.2 ^{o}
20^o	13.6 ^{o}
30^o	21.4 ^{o}
40^o	29.6 ^{o}
50^o	40.2 ^{o}
60^o	50.8 ^{o}
70^o	62.6 ^{o}
80^o	76.9 ^{o}
90^o	85.1 ^{o}

Εικόνα Λ. Πίνακας μετρήσεων πραγματικών κλίσεων pitch γωνιών του ψηφιακού κλισίμετρου

Για να έχουμε μια πιο σαφή εικόνα για την απόδοση του ψηφιακού κλισίμετρου η ακρίβεια των μετρήσεων παρουσιάζετε σε ποσοστά επί τις εκατό (εικόνα Μ). Οι αποκλίσεις που προέκυψαν μπορεί να οφείλονται είτε στο handshake, είτε στο θόρυβο του αισθητήρα επιτάχυνσης που προκαλείτε λόγο της γραμμικής επιτάχυνσης που προσδίδετε στον αισθητήρα κατά την διαδικασία λήψης των τιμών.



Εικόνα Μ. Ακρίβεια μετρήσεων γωνιών pitch από το ψηφιακό κλισίμετρο σε ποσοστό επί τις 100.

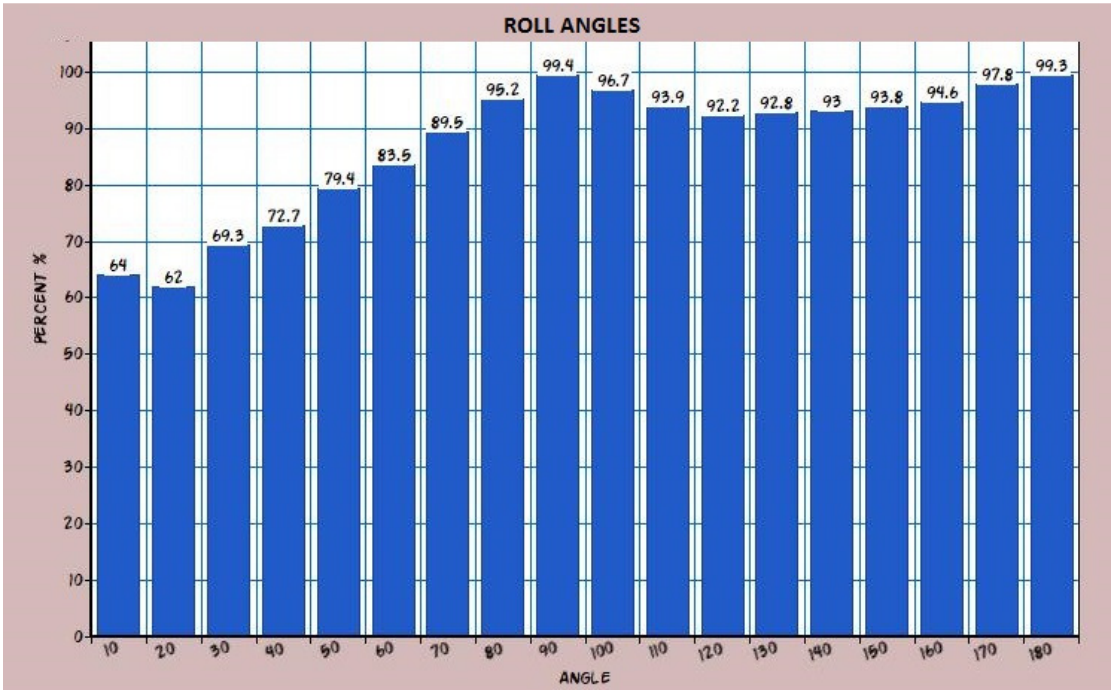
6.2 Αξιολόγηση απόδοσης γωνιών roll

Η ίδια ακριβώς διαδικασία που εφαρμόστηκε για τις γωνίες περιστροφής pitch ακολουθήθηκε και για τις γωνίες roll. Το εύρος των γωνιών roll για το οποίο πραγματοποιήθηκε το πείραμα κυμαίνετε από τις 10 έως τις 180 μοίρες. Ο πίνακας αποτελεσμάτων παρουσιάζετε παρακάτω (εικόνα Ν).

10^o	6.4^o
20^o	12.4^o
30^o	20.8^o
40^o	29.1^o
50^o	39.7^o
60^o	50.1^o
70^o	62.7^o
80^o	76.2^o
90^o	89.5^o
100^o	104.3^o
110^o	117.7^o
120^o	129.3^o
130^o	139.3^o
140^o	149.7^o
150^o	159.2^o
160^o	168.5^o
170^o	173.7^o
180^o	178.9^o

Εικόνα Ν. Πίνακας αποτελεσμάτων των μετρήσεων πραγματικών γωνιών από το ψηφιακό κλισίμετρο.

Τα αποτελέσματα της δειγματοληψίας παρουσιάζονται επίσης στο παρακάτω διάγραμμα με βάση την εκατοστιαία ποσοστιαία μονάδα.



Εικόνα Ξ. Ακρίβεια μετρήσεων γωνιών roll από το ψηφιακό κλισίμετρο σε ποσοστό επί τις 100.

7. Βιβλιογραφία

1. Douglas C. Schmidt (xx)" *Overview of Remote Procedure Calls (RPC*)" , Washington University, st. Louis διαθέσιμο στο διαδίκτυο στην ιστοσελίδα http://www.cs.wustl.edu/~schmidt/PDF/rpc4.pdf , τελευταία προσπέλαση 05/10/2017.

2. Sašo Tomažič Sara Stančin (2011) "Simultaneous Orthogonal Rotations Angle" , Faculty of Electrical Engineering, University of Ljubljana διαθέσιμο στο διαδίκτυο στην ιστοσελίδα http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.7591&rep=rep1&type=pdf τελευταία προσπέλαση 11/10/2017.

3. Karsten Groÿekatthöfer, Dr. -Ing. Zizung Yoon (2012), "Introduction into quaternions for spacecraft attitude representation" διαθέσιμο στο διαδίκτυο στην ιστοσελίδα http://www.tu-berlin.de/fileadmin/fg169/miscellaneous/Quaternions.pdf τελευταία προσπέλαση 13/10/2017.

4. Absolute Astronomy, "Axis and Angles" διαθέσιμο στο διαδίκτυο στην ιστοσελίδα http://www.absoluteastronomy.com/topics/Axis_angle τελευταία προσπέλαση 14/11/2017

8. Ιστογραφία

i. Janssen C. & D. (2017) , "*Embedded System*" Techopedia Inc Διαθέσιμο στο διαδίκτυο στην ιστοσελίδα https://www.techopedia.com/about τελευταία προσπέλαση 10/10/2017.

ii. Makeagift.com (2015) Διαθέσιμο στο διαδίκτυο στην ιστοσελίδα https://makeagif.com/gif/airplane-control-roll-pitch-yaw-JK39i8 τελευταία προσπέλαση 11/10/2017

iii. Mark Pedley (2013), "Titl sensing using a Three-Axis accelerometer" Διαθέσιμο στο διαδίκτυο στην ιστοσελίδα https://www.nxp.com/docs/en/application-note/AN3461.pdf τελευταία προσπέλαση 14/10/2017.

9. Παράρτημα

Βασικό πρόγραμμα cyberphysical_glove.py

```
import  time, pygame, ctypes
from RPCmbed import *
from Libraries_cyber_glove import *
from operator import itemgetter


black = (0,0,0)
red = (255,0,0)
green = (0,170,0)
class cyber_glove:
   def __init__(self):
      self.mbed = SerialRPC('COM6',9600)
      '''raw accelerometer data'''
      self.accy = RPCVariable(self.mbed,"y-axis")
      self.accx = RPCVariable(self.mbed,"x-axis")
      self.accz = RPCVariable(self.mbed,"z-axis")
      '''source register variable'''
      self.source_register = RPCVariable(self.mbed ,"sr")
      '''patern recognition variables'''
      self.time_start = 0
      #initialize the clock to off
      self.clock_state = 0
      #set the countdown in seconds
      self.time_window = 7
      self.first_sequence = []
```

```
        self.second_sequence = []
        #set the angles which partisepate in the patern recognision
        self.sequence_angle = [25,43,55,58,60]
        #seconds left to stop the countdown
        self.seconds = 7
        '''tap recognition variable'''
        self.axes = 'y+'
        '''3d cube variables'''
        self.vertices = [
                Nodes(-1,1,-1),
                Nodes(1,1,-1),
                Nodes(1,-1,-1),
                Nodes(-1,-1,-1),
                Nodes(-1,1,1),
                Nodes(1,1,1),
                Nodes(1,-1,1),
                Nodes(-1,-1,1) ]
        self.faces = [(0,1,2,3),(1,5,6,2),(5,4,7,6),(4,0,3,7),(0,4,5,1),(3,2,6,7)]
        self.colors                                                    =
[(255,69,0),(255,165,0),(0,255,0),(131,137,150),(0,0,255),(255,0,0)]
        '''graphics variables'''
        pygame.init()
        self.screen = pygame.display.set_mode((1366, 768))
        self.clock = pygame.time.Clock()
        self.FPS = 50
        self.smallfont = pygame.font.SysFont(None,80)
        self.tinyfont = pygame.font.SysFont(None,25)
```

```
    pygame.display.set_caption ("Esda Lab")
    self.arrow = pygame.image.load('arrow.png')
    #list containes the 'YES' for the recognized paterrn
    self.succed_patern = []

def monitor(self,x,y,digits, value,newx1 = 0, start = (0,0), stop = (0,0)):
    if digits == 3:
        self.screen_text = self.smallfont.render("PITCH", True, black)
        self.screen.blit (self.screen_text,[100,590])
        pygame.draw.rect (self.screen,black,(x,y,digits*70,88))
        #display the symbol o
        pygame.draw.rect (self.screen,red, (x+190,y+10,15,15),4)
    elif digits == 4:
        self.screen_text = self.smallfont.render("ROLL", True, black)
        self.screen.blit (self.screen_text,[370,590])
        pygame.draw.rect (self.screen,black,(x,y,digits*70,88))
        pygame.draw.rect ( self.screen,red, (x+255,y+10,15,15),4)
    elif digits == 1:
        self.screen_text = self.tinyfont.render("CLOCK", True, black)
        self.screen.blit (self.screen_text,[840,60])
        pygame.draw.rect (self.screen ,black ,(x,y,digits*70,88))

    value = str(value)
    for i in range(0,digits):
        #six nodes tha compose a digit
        pointslist = [(x + 10 + newx1, y+ 7.5),
```

```
        (x + 60 + newx1, y + 7.5),
        (x + 10 + newx1, y + 45),
        (x + 60 + newx1, y + 45),
        (x + 10 + newx1, y + 82.5),
        (x + 60 + newx1, y + 82.5)]
#six vertices tha connect the nodes
vertices = [(pointslist[0],pointslist[1]), # 0
      (pointslist [2],pointslist [3]), # 1
      (pointslist [4],pointslist [5]), # 2
      (pointslist [0],pointslist [2]),# 3
      (pointslist [1],pointslist [3]), # 4
      (pointslist [2],pointslist [4]), # 5
      (pointslist [3],pointslist [5])] # 6

number = value[i:i+1]
linescounter = -1
if number == '0':
   for n in vertices:
      linescounter += 1
      if linescounter == 1:
         continue
      start = n[0]
      stop = n[1]
      pygame.draw.line(self.screen,red,start,stop,5)
elif number == '1':
   for n in vertices:
```

```
            linescounter += 1
            if linescounter == 4 or linescounter == 6:
                start = n[0]
                stop = n[1]
                pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '2':
            for n in vertices:
                linescounter += 1
                if linescounter == 3 or linescounter == 6:
                    continue
                start = n[0]
                stop = n[1]
                pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '3':
            for n in vertices:
                linescounter += 1
                if linescounter == 3 or linescounter == 5:
                    continue
                start = n[0]
                stop = n[1]
                pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '4':
            for n in vertices:
                linescounter += 1
                if linescounter == 0 or linescounter == 5 or linescounter == 2:
                    continue
```

```
            start = n[0]
            stop = n[1]
            pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '5':
            for n in vertices:
                linescounter += 1
                if linescounter == 4 or linescounter == 5:
                    continue
                start = n[0]
                stop = n[1]
                pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '6':
            for n in vertices:
                linescounter += 1
                if linescounter == 4 or linescounter == 0:
                    continue
                start = n[0]
                stop = n[1]
                pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '7':
            for n in vertices:
                linescounter += 1
                if linescounter == 3 or linescounter == 5 or linescounter == 1 or
linescounter == 2:
                    continue
                start = n[0]
                stop = n[1]
```

```
            pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '8':
            for n in vertices:
                linescounter += 1
                start = n[0]
                stop = n[1]
                pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '9':
            for n in vertices:
                linescounter += 1
                if linescounter == 5:
                    continue
                start = n[0]
                stop = n[1]
                pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '-' :
            for n in vertices:
                linescounter += 1
                if linescounter == 1:
                    start = n[0]
                    stop = n[1]
                pygame.draw.line(self.screen,red,start,stop,5)
        elif number == '.' :
            break
            #pygame.draw.line (self.screen,red,(x + 10 + newx1, y + 110),(x + 15 +
newx1, y + 110),5)
```

```
            newx1 += 60



    def rotatetor(self):
        '''SET THE NEW ANGLE POSITION OF THE CUBE - (FIND THE NEW PLACE OF THE ROTATED POINTS)'''
        # It will hold transformed vertices.
        t = []
        for v in self.vertices:
            # Rotate the point around X axis, then around Y axis, and finally around Z axis.
            r = v.rotateY (self.roll ).rotateX (self.pitch)
            # Transform the point from 3D to 2D
            p = r.project(1366, 768, 256, 2.8)
            # Put the point in the list of transformed vertices
            t.append(p)
        # Calculate the average Z values of each face.
        avg_z = []
        i = 0
        for f in self.faces:
            z = (t[f[0]].z + t[f[1]].z + t[f[2]].z + t[f[3]].z) / 4.0
            avg_z.append ([i,z])
            i = i + 1
        # Draw the faces using the Painter's algorithm:
        # Distant faces are drawn before the closer ones.
        for tmp in sorted(avg_z, key=itemgetter(1),reverse=True):
            face_index = tmp[0]
```

```
        f = self.faces[face_index]
        pointlist = [(t[f[0]].x, t[f[0]].y), (t[f[1]].x, t[f[1]].y),
                (t[f[1]].x, t[f[1]].y), (t[f[2]].x, t[f[2]].y),
                (t[f[2]].x, t[f[2]].y), (t[f[3]].x, t[f[3]].y),
                (t[f[3]].x, t[f[3]].y), (t[f[0]].x, t[f[0]].y)]
        #draw the new angular position of the cube
        pygame.draw.polygon(self.screen,self.colors[face_index],pointlist)

    def pattern_graphics(self):
        #draw the patterns label
        self.pattern1_text = self.smallfont.render("Pattern P-P+R+R-", True, black)
        self.screen.blit(self.pattern1_text,[750,230])
        self.pattern2_text = self.smallfont.render("Pattern R-R+P+P-", True, black)
        self.screen.blit(self.pattern2_text,[750,330])
        #draw the angles of pattern
        s = 0
        for each_angle in self.sequence_angle:
            each_angle = str (each_angle)
            self.patternangle1 = self.smallfont.render (each_angle + u'\N{DEGREE
SIGN}', True, black)
            self.screen.blit(self.patternangle1,[750,420+s])
            s += 60
        #draw the succesive symbol next to each succeded angle
        if len (self.succed_patern) != 0:
            k = 0
            for i in range (0,len(self.succed_patern)):
                #get a random string
```

```
            newstring = id_generator()
            newstring = self.smallfont.render("YES", True, green)
            self.screen.blit (newstring,[850,420+k])
            k += 60

        #draw an arrow to display the users selection of the pattern
        if self.axes == 'y+':
            self.screen.blit (self.arrow,(630,215))
            #make the arrow alive :)
            for i in range(0,20):
                self.screen.blit (self.arrow,(630 - i, 215))
                pygame.display.flip()
        else:
            self.screen.blit (self.arrow,(630,315))
            #make the arrow alive :)
            for i in range(0,20):
                self.screen.blit (self.arrow,(630 - i, 315))
                pygame.display.flip()

    def execution(self):
        while True:
            '''GET PITCH AND ROLL ANGLES'''
            self.pitch, self.roll = get_pitch_and_roll (self.accx, self.accy, self.accz)

            if self.clock_state == 0:
                '''GET AXE OF TAP'''
```

```
            #variable to help the programm rember the content of the axes variable, when no tap event has occur

            last_axes = self.axes

            #tap axes has point (exei nohma) only when clock is off

            self.axes = get_axes_tap (self.source_register,last_axes)

            '''GET PATERN TO RECOGNISED'''

            if self.axes == 'y+':

              #search for sequence { pitch+, pitch- , roll+, roll- }. returns a list reference to sequence = [0,0,0,0]

              self.first_sequence = initialize_1st_sequence(self.sequence_angle, self.first_sequence)

            elif self.axes == 'y-':

              #search for sequence { roll- , roll+, pitch-, pitch+ }. returns a list reference to sequence = [0,0,0,0]

              self.second_sequence = initialize_2nd_sequence(self.sequence_angle,self.second_sequence)

            '''SET A COUNTDOWN'''

            #start the countdown if has to

            self.clock_state= fire_the_countdown (self.pitch)

            if self.clock_state == 1:

              #get current time, this function must run only one time every time period

              self.time_start = get_current_time()

            self.succed_patern = []

          '''GET THE PROGRESS OF THE PATERN'''

          if self.clock_state == 1 and self.axes =='y+':

            self.first_sequence,self.succed_patern = patern_recognisionPprR (self.pitch,self.roll,self.sequence_angle,

            self.first_sequence,self.succed_patern)
```

```
            print self.first_sequence
            #check if time is over
            self.clock_state, self.seconds = get_clock (self.time_start, self.time_window)
        elif self.clock_state == 1 and self.axes == 'y-':
            self.second_sequence,succed_patern = patern_recognisionRrpP (self.pitch,self.roll,self.sequence_angle,
            self.second_sequence,self.succed_patern)
            #check if time is over
            self.clock_state,self.seconds=get_clock(self.time_start, self.time_window)

        '''DRAW THE GRAPHICS'''
        #draw the graphical enviroment white
        pygame.draw.rect(self.screen,(255,255,255),(0,0,1366,768))
        #draw the angular position of the cube
        self.rotatetor()
        #draw the values of pitch, roll and countdown clock on a digital lcd
        self.monitor(50,500,3,self.pitch)
        self.monitor(300,500, 4,self.roll)
        self.monitor(900,80,1, self.seconds)
        self.pattern_graphics()
        pygame.display.flip()
        self.clock.tick (self.FPS)
        # shut down the graphical enviroment if the quit window button pressed
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
```

```
            pygame.quit()


if __name__ == "__main__":
    cyber_glove().execution()
```

<u>Βιβλιοθήκη Libraries cyber_glove.py του βασικού προγράμματος cyberphysical_glove.py</u>

```
import math, time, pygame, random, string
from operator import itemgetter
#1
def get_pitch_and_roll (accx, accy, accz):
    y =[]*6
    y = accy.read() [:6]
    if 'r' in y:
        i = 0
        while y[i] != 'r':
            i = i + 1
        y = y [:i]
    x =[]*6
    x = accx.read() [:6]
    if 'r' in x:
        j = 0
        while x[j] != 'r':
            j = j + 1
        x = x [:j]
    z =[]*6
```

```
    z = accz.read() [:6]
    if 'r' in z:
        n = 0
        while z[n] != 'r':
            n = n + 1
        z = z [:n]
    fx = float(x)
    fy = float(y)
    fz = float(z)
    fxfilt = 0
    fyfilt = 0
    fzfilt = 0
    alpha = 0.7
    fxfilt = fx * alpha + (fxfilt * (1.0 - alpha))
    fyfilt = fy * alpha + (fyfilt * (1.0 - alpha))
    fzfilt = fz * alpha + (fzfilt * (1.0 - alpha))
    pitch_angle = (math.atan2(fxfilt, math.sqrt (fyfilt *fyfilt + fzfilt *fzfilt)) *
180.0) / math.pi
    roll_angle = (math.atan2(-fyfilt,fzfilt) * 180.0) / math.pi
    return pitch_angle , roll_angle


#2
def get_axes_tap (source_register,last_axes):
    sourcereg = source_register.read()
    #eliminate the 'r' from the serial port
    if sourcereg[:3] == '0 r':
        return last_axes
```

```
    if sourcereg[:3] != '0 r':
        sourcereg = int (sourcereg[:3])
        sourcereg = bin(sourcereg)
        if sourcereg == '0b10100000':
            return 'y+'
        elif sourcereg == '0b10100010':
            return 'y-'



#3
#initialize the list contaenes the order which the angles have to occurr in order to recognize the patern (up, down, right,left)
def initialize_1st_sequence(sequence_angle,first_sequence):
    for x in range (0, len (sequence_angle)):
            first_sequence.insert(0,[0,0,0,0])
    #also make sure to clear the sequence list after every try
    return first_sequence [0:len(sequence_angle)]
#initialize the list contaenes the order which the angles have to occurr in order to recognize the patern (left,right,up,down)
def initialize_2nd_sequence(sequence_angle,second_sequence):
    for x in range (0, len (sequence_angle)):
        second_sequence.insert(0,[0,0,0,0])
    #also make sure to clear the sequence list after every try
    return second_sequence [0:len(sequence_angle)]


'''MECHANISM FOR RECOGNIZE PATERN'''
#function for recognize sequence: pitch+, pitch-, roll+, roll-
```

```
def
patern_recognisionPprR(pitch,roll,sequence_angle,first_sequence,succed_pate
rn):

  #until the 1st angle recognised in first_sequence

  for this_angle in range (0, len (sequence_angle)):

    if pitch < -sequence_angle [this_angle] and first_sequence [this_angle] ==
[0,0,0,0]:

      first_sequence [this_angle].append(1)

      del first_sequence [this_angle][0]


    elif pitch > sequence_angle [this_angle] and first_sequence[this_angle] ==
[0,0,0,1]:

      first_sequence [this_angle].append(1)

      del first_sequence [this_angle][0]


    elif roll > sequence_angle[this_angle] and first_sequence [this_angle] ==
[0,0,1,1]:

      first_sequence[this_angle].append(1)

      del first_sequence[this_angle][0]


    elif roll < -sequence_angle [this_angle] and first_sequence [this_angle] ==
[0,1,1,1]:

      succed_patern.append (this_angle)

      first_sequence[this_angle] = [0,0,0,0]


  return first_sequence, succed_patern

def
patern_recognisionRrpP(pitch,roll,sequence_angle,second_sequence,succed_p
atern):
```

```
    for this_angle in range (0, len (sequence_angle)):
        if roll > sequence_angle[this_angle] and second_sequence[this_angle] == [0,0,0,0]:
            second_sequence [this_angle].append(1)
            del second_sequence[this_angle][0]
        elif roll < -sequence_angle[this_angle] and second_sequence[this_angle] == [0,0,0,1]:
            second_sequence [this_angle].append(1)
            del second_sequence[this_angle][0]
        elif pitch > sequence_angle[this_angle] and second_sequence[this_angle] == [0,0,1,1]:
            second_sequence[this_angle].append(1)
            del second_sequence[this_angle][0]
        elif pitch < -sequence_angle[this_angle] and second_sequence[this_angle] == [0,1,1,1]:
            succed_patern.append(this_angle)
            second_sequence[this_angle] = [0,0,0,0]

    return second_sequence, succed_patern


def fire_the_countdown(pitch):
    if pitch < -15:
        #start countdown
        clock_state = 1
        return clock_state
    else:
        return 0
def get_current_time():
```

```
    time_start = time.time()
    return time_start
def get_clock (time_start, time_window):
        #calculate the time passed until now in seconds
        seconds = int (time.time() - time_start)
        #check if the countdown is over
        if seconds == time_window:
            #reset the countdown
            return 0, 7
        else:
            return 1, 7-seconds


#4
class Nodes:
    def __init__ (self, x= 0 , y = 0, z = 0):
        #Nodes of 3dcube
        self.x, self.y, self.z = float(x), float(y), float(z)


    def rotateX(self, angle):
        #rotation matrix of x node
        rad = angle * math.pi / 180
        cosa = math.cos(rad)
        sina = math.sin(rad)
        y = self.y * cosa - self.z * sina
        z = self.y * sina + self.z * cosa
        return Nodes(self.x, y, z)
```

```
    def rotateY(self, angle):
        #Rotation matrix of y node
        rad = angle * math.pi / 180
        cosa = math.cos(rad)
        sina = math.sin(rad)
        z = - self.z * cosa + self.x * sina
        x = self.z * sina + self.x * cosa
        return Nodes(x, self.y, z)


    #method to represent the projection of the node on the GUI screen (perspective projection)
    def project(self, win_width, win_height, fov, viewer_distance):
        transformation = [550,220]
        factor = fov / (viewer_distance + self.z)
        x = self.x * factor + win_width  / 1.6
        y = -self.y * factor + win_height / 1.6
        return Nodes(x - transformation[0], y - transformation[1], self.z)


#random string generator
def id_generator(size=6, chars=string.ascii_uppercase + string.digits):
    return ''.join(random.choice(chars) for _ in range(size))
```

Αλγόριθμος στο ενσωματωμένο σύστημα

```
#include "mbed.h"
#include "MMA8451Q.h"
#include "SerialRPCInterface.h"
#define MMA8451_I2C_ADDRESS (0x1d<<1)
PinName const SDA = PTE25;
PinName const SCL = PTE24;
MMA8451Q acc(SDA, SCL, MMA8451_I2C_ADDRESS); //orizw ta SDA kai SCL pins kathws kai tin I2C dieuthinsi

SerialRPCInterface RPC(USBTX, USBRX);
//Setup the interrupts for the MMA8451Q
InterruptIn accInt1(PTA14);
float X, Y, Z;
int SR;
RPCVariable<float> RPC_X(&X, "x-axis");
RPCVariable<float> RPC_Y(&Y, "y-axis");
RPCVariable<float> RPC_Z(&Z, "z-axis");
RPCVariable<int> RPC_SR(&SR, "sr");
//interuppt tap event service routine
void SingletapTrue(){
   int static flag = 0;
    if (flag == 0){
        flag = 1;
        TAP = 1;
```

```
                }
    else{
       flag = 0;
        TAP = 0 ;

         }
       }

int main(void) {
   acc.lv1setSingleTap();
    accInt1.rise(&SingletapTrue); // call interuppt service rutine
    acc.disable_lpfilter();

   while(1) {
X = acc.getAccX();
Y = acc.getAccY();
Z = acc.getAccZ();
SR = acc.get_source_reg_value();
    }
```