Ατομική Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΤΗΛΕΧΕΙΡΙΣΤΗΡΙΟΥ ΣΕ ΣΥΣΚΕΥΗ IPAD Η ΟΠΟΙΑ ΕΛΕΓΧΕΙ ΕΛΙΚΟΠΤΕΡΟ ΜΕ ΜΙΚΡΟΕΛΕΓΧΤΗ ARDUINO

Μαργαρίτα Παπαευθυμίου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2013

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη τηλεχειριστηρίου σε συσκευή iPad
η οποία ελέγχει ελικόπτερο με μικροελεγχτή Arduino

Επιβλέπων καθηγητής

Βάσος Βασιλείου

Η ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου.

Μάιος 2013

Ευχαριστίες

Θα ήθελα να δώσω τις ευχαριστίες μου στον καθηγητή κύριο Βάσο Βασιλείου, για τις πολύτιμες συμβουλές και χρήσιμες πληροφορίες που μου παρείχε στα διάφορα στάδια της διπλωματικής μου εργασίας που είχαν σαν αποτέλεσμα την επιτυχή ολοκλήρωση της. Ακόμη, θα ήθελα να τον ευχαριστήσω για την προσφορά της κινητής συσκευής ούτως ώστε να γίνει κατορθωτός ο έλεγχος της σωστής λειτουργίας της εφαρμογής.

Επίσης, θα ήθελα να ευχαριστήσω τον καθηγητή κύριο Δημήτρη Ζεϊναλιπούρ για την βοήθεια που πρόσφερε για την τροφοδότηση της συσκευής με σκοπό τον έλεγχο της εφαρμογής στην συσκευή iPad.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου που στήριξε στις προσπάθειές μου, καθ' όλη τη διάρκεια των μαθητικών αλλά και ακαδημαϊκών μου σπουδών.

Περίληψη

Αντικείμενο της διπλωματικής μου εργασίας είναι η δημιουργία εφαρμογής τηλεχειριστηρίου παρόμοιας με αυτήν του RCTx στο Apple Store. Στόχος του τηλεχειριστηρίου αυτού είναι μέσω του πρωτοκόλλου RCOIP του πακέτου RCKit [12] να επικοινωνεί με συγκεκριμένες συχνότητες με το υλικό (hardware) του ελικοπτέρου. Το υλικό του ελικοπτέρου αποτελείται από την πλακέτα Arduino με ασπίδα Wi-fi (WifiShield) και θα επικοινωνεί με την εφαρμογή μέσω ασύρματου δικτύου. Με αυτή την επικοινωνία το τηλεχειριστήριο θα μπορεί να δώσει διάφορες παραμέτρους στο ελικόπτερο οι οποίες καθορίζουν την επιτάχυνση, την ταχύτητα, και τις συντεταγμένες τριών διαστάσεων (στον χώρο). Επιπρόσθετα η εφαρμογή θα διαθέτει δύο τηλεχειριστήρια και 1 διακόπτη για επικοινωνία με το υλικό του ελικοπτέρου. Το αριστερό τηλεχειριστήριο το οποίο θα ορίζει τιμές στα κανάλια 0 και 1 και το δεξιό τηλεχειριστήριο το οποίο θα ορίζει τιμές στα κανάλια 2 και 3. Ο διακόπτης θα ορίζει το κανάλι 4. Επίσης η εφαρμογή θα μπορεί να δώσει κίνηση στο ελικόπτερο μέσω της κίνησης του iPad. Ο συγχρονισμός των τιμών της εφαρμογής του τηλεχειριστηρίου θα γίνεται κάθε 1 χιλιοστό του δευτερολέπτου.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Εφαρμογή RCTx του AppleStore

Η διπλωματική μου εργασία έχει ως στόχο την δημιουργία εφαρμογής παρόμοιας με αυτή του Apple Store. Η εφαρμογή RCTx [19], είναι διαθέσιμη στο Apple Store στην τιμή των $9.99.

Η εφαρμογή αυτή είναι συμβατή με συσκευές iPhone, iPod touch και iPad. Απαιτεί λειτουργικό σύστημα iOS με έκδοση 4.3 ή νεότερη.

Χρησιμοποιεί ένα Wi-Fi Ad-Hoc δίκτυο και το πρωτόκολλο RCOIP για να επικοινωνήσει με το Arduino, το οποίο πρέπει να είναι εξοπλισμένο με WiShield.

Ο πομπός αυτός λειτουργεί με μικροελεγκτή Arduino, όπου ο κώδικας του πρέπει να είναι γραμμένος χρησιμοποιώντας το λογισμικό RCKit. Το λογισμικό αυτό είναι υπεύθυνο για την δημιουργία οχημάτων απομακρυσμένου ελέγχου, τα οποία διαθέτουν κινητήρες (motors), βηματικούς κινητήρες (steppers), σερβοκινητήρες (servos) , αναλογικές και ψηφιακές εισόδους / εξόδους.

Επίσης η εφαρμογή αυτή, δεν μπορεί να χρησιμοποιηθεί για έλεγχο ασύρματων συμβατικών οχημάτων (conventional radio control vehicles).

Προσεχείς εκδόσεις του RCTx μπορεί να περιέχουν ρυθμίσεις διασύνδεσης πομπού, κανάλια για το παγκόσμιο σύστημα θεσιθεσίας (GPS), απομακρυσμένη χαρτογράφηση της θέσης του οχήματος, απομακρυσμένες κάμερες πάνω στο όχημα, μακρό καταγραφή και άλλα.

Στην Εικόνα 1.1 βλέπουμε στιγμιότυπο από την κύρια οθόνη της εφαρμογής, και στην Εικόνα 1.2 και 1.3 βλέπουμε στιγμιότυπα από την εφαρμογή όπου γίνεται η διαμόρφωση του αποστολέα.

Εικόνα 1.1: Η εφαρμογή RCTx του Apple Store

Εικόνα 1.2: Στιγμιότυπο από την εφαρμογή RCTx για διαμόρφωση του αποστολέα

Εικόνα 1.3: Στιγμιότυπο από την εφαρμογή RCTx για διαμόρφωση του αποστολέα

Κεφάλαιο 2

Apple

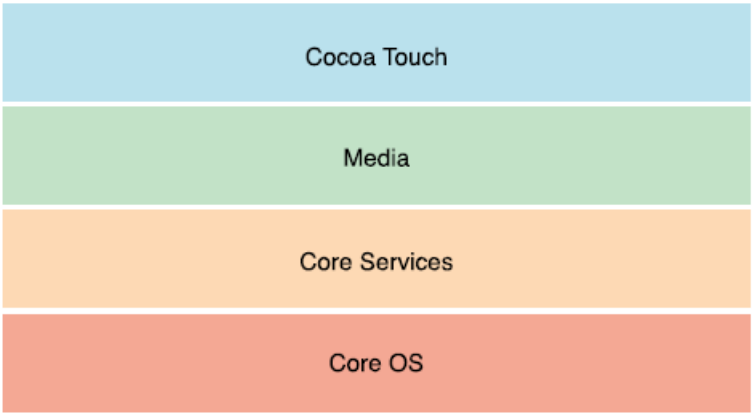
2.1 Λειτουργικό σύστημα iOS

Το iOS [3] είναι ένα λειτουργικό σύστημα για κινητές συσκευές που αναπτύχθηκε και διανέμεται από την εταιρεία Apple Inc. Αρχικά κυκλοφόρησε το 2007 για το iPhone και το iPodTouch και έχει επεκταθεί για να υποστηρίξει και άλλες Apple συσκευές όπως το iPad και το Apple TV.

Η Apple δεν δίνει την άδεια για την εγκατάσταση του iOS λειτουργικού συστήματος σε υλικό το οποίο δεν είναι της Apple.

Η υλοποίηση των iOS τεχνολογιών μπορεί να θεωρηθεί σαν ένα σύνολο στρωμάτων τα οποία απεικονίζονται στην Εικόνα 2.1. Στα κατώτερα στρώματα του συστήματος βρίσκονται οι θεμελιώδεις υπηρεσίες και τεχνολογίες στις οποίες βασίζονται όλες οι

εφαρμογές και τα υψηλότερου επιπέδου στρώματα περιέχουν περισσότερο εξελιγμένες υπηρεσίες και τεχνολογίες.



Εικόνα 2.1: Επίπεδα του λειτουργικού συστήματος iOS

Cocoa Touch Layer

Το στρώμα αυτό περιέχει όλα τα χαμηλού επιπέδου χαρακτηριστικά πάνω στα οποία χτίζονται οι περισσότερες άλλες τεχνολογίες. Αυτές οι τεχνολογίες μπορεί να μην χρησιμοποιούνται άμεσα από τις εφαρμογές μας, αλλά μπορεί να εμπεριέχονται στα πλαίσια εργασίας (frameworks) τα οποία χρησιμοποιούμε στις εφαρμογές μας. Σε περιπτώσεις που πρέπει να έχουμε ασφάλεια ή επικοινωνία με εξωτερικό υλικό πρέπει να χρησιμοποιηθούν άμεσα τα πλαίσια εργασίας αυτού του επιπέδου.

Κάποιες βασικές τεχνολογίες οι οποίες παρέχονται σε αυτό το επίπεδο είναι οι πιο κάτω.

Αυτόματη διάταξη (Auto Layout)

Εισάχθηκε για πρώτη φορά στο iOS 6. Με την αυτόματη διάταξη μπορεί να καθοριστούν κανόνες για το πως θα είναι η διάταξη. Για παράδειγμα, μπορεί να καθοριστεί ότι ένα κουμπί, θα πρέπει πάντα να είναι 20 πόντους από το αριστερό άκρο της οθόνης.

Υποστήριξη εγγράφων (Document Support)

Εισάχθηκε για πρώτη φορά στο iOS 5. Στο UIKit πλαίσιο εργασίας εισάχθηκε η κλάση UIDocument για την διαχείριση των δεδομένων που σχετίζονται με έγγραφα των χρηστών.

Multitasking

Όταν πιέζεται το κουμπί «Home» οι διεργασίες δεν τερματίζουν αλλά μεταφέρονται στο προσκήνιο.

Αναγνωριστές χειρονομιών (Gesture Recognizers)

Εισαχθήκανε στο iOS 3.2 στοιχεία αναγνώρισης χειρονομιών, όπως για παράδειγμα η περιστροφή (rotating) και η μεταφορά (dragging).

Media Layer

Το στρώμα αυτό περιέχει τις τεχνολογίες για γραφικά, ήχο και βίντεο με σκοπό τη δημιουργία των καλύτερων εμπειριών πολυμέσων, που μπορούν να διατίθενται σε φορητές συσκευές.

Core Services Layer

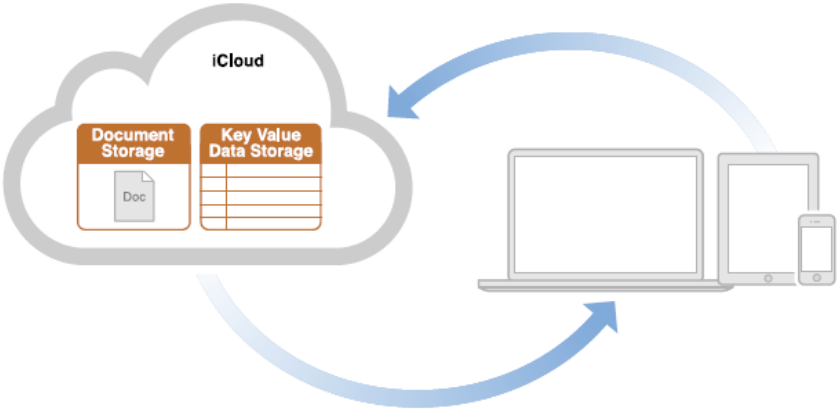
Το στρώμα αυτό περιέχει τις βασικές υπηρεσίες του συστήματος που χρησιμοποιούν όλες οι εφαρμογές. Ακόμη και αν δεν χρησιμοποιούνται άμεσα αυτές οι υπηρεσίες, πολλά μέρη του συστήματος χτίστηκαν πάνω τους.

Κάποιες βασικές τεχνολογίες οι οποίες παρέχονται σε αυτό το επίπεδο είναι οι πιο κάτω.

iCloud Storage

Εισάχθηκε στο iOS 5 και αυτό το σύστημα αποθήκευσης επιτρέπει στις εφαρμογές να γράφουν τα δεδομένα και τα έγγραφα του χρήστη σε μια κεντρική τοποθεσία (central location). Με αυτό τον τρόπο, οι χρήστες μπορούν να έχουν πρόσβαση σε αυτά τα

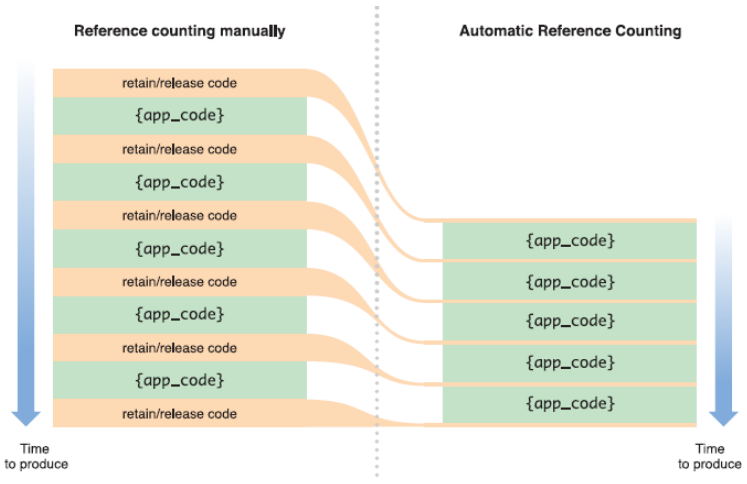
δεδομένα από οποιαδήποτε iOS συσκευή χωρίς να χρειάζεται να μεταφέρουν τα συγκεκριμένα αρχεία. Στην Εικόνα 2.2 φαίνεται παραστατικά η τεχνολογία αυτή.



Εικόνα 2.2: iCloud Storage που προσφέρει το στρώμα Core Services

Αυτόματη Αναφορά Μετρήσεων (Auto reference counting ,ARC))

Εισάχθηκε στο iOS 5. Είναι ένα χαρακτηριστικό στο επίπεδο του μεταγλωττιστή που απλοποιεί την διαχειριστή των αντικειμένων του κώδικα. Δεν χρειάζεται να θυμάσαι πότε πρέπει να διατηρήσεις ή να αποδεσμεύσεις ένα αντικείμενο. Η αυτόματη αναφορά μετρήσεων αξιολογεί τις απαιτήσεις στην ζωή των αντικειμένων και εισάγει αυτόματα στον κώδικα κατά την διάρκεια της μεταγλώττισης, τις κατάλληλες μεθόδους για την απελευθέρωση της μνήμης των αντικειμένων. Στην Εικόνα 2.3 φαίνεται παραστατικά η τεχνολογία αυτή.



Εικόνα 2.3: ARC που προσφέρει το στρώμα Core Services

Προστασία δεδομένων (Data Protection)

Εισάχθηκε στο iOS 4. Επιτρέπει στις εφαρμογές που λειτουργούν με ευαίσθητα δεδομένα του χρήστη, να εκμεταλλευτούν την ενσωματωμένη κρυπτογράφηση που είναι διαθέσιμη σε κάποιες συσκευές. Όταν μια εφαρμογή για παράδειγμα διαχειρίζεται ένα συγκεκριμένο αρχείο σαν προστατευόμενο, το σύστημα φυλάει το αρχείο αυτό σε κρυπτογραφημένη μορφή.

Core OS Layer

Το στρώμα αυτό περιέχει τα χαμηλού επιπέδου χαρακτηριστικά πάνω στα οποία βασίζονται πολλές άλλες τεχνολογίες. Όπως και στο προηγούμενο επίπεδο, οι υπηρεσίες που παρέχει το στρώμα αυτό μπορεί να μην χρησιμοποιούνται άμεσα, όμως τα περισσότερα μέρη του συστήματος χτίστηκαν πάνω τους. Χρησιμοποιώντας τα πλαίσια αυτού του στρώματος μπορούμε να δίνουμε την ασφάλεια ρητά, και ακόμη μπορεί να επιτυγχάνουμε επικοινωνία με αξεσουάρ του εξωτερικού υλικού (external hardware accessory).

2.2 iOS SKD

Το iOS SDK [11] είναι ένα πακέτο εφαρμογών ανάπτυξης λογισμικού που δημιουργήθηκε από την Apple Inc. και κυκλοφόρησε το Φεβρουάριο του 2008 για την ανάπτυξη εφαρμογών για το iOS.

Το εργαλείο για ανάπτυξη, Xcode [28] παρέχει όλα όσα χρειάζονται για την δημιουργία εκπληκτικών εφαρμογών για Mac, iPhone και iPad.

Το SDK περιέχει τα εργαλεία και τις διασυνδέσεις που απαιτούνται για την ανάπτυξη, εγκατάσταση, λειτουργία, και δοκιμή των εφαρμογών.

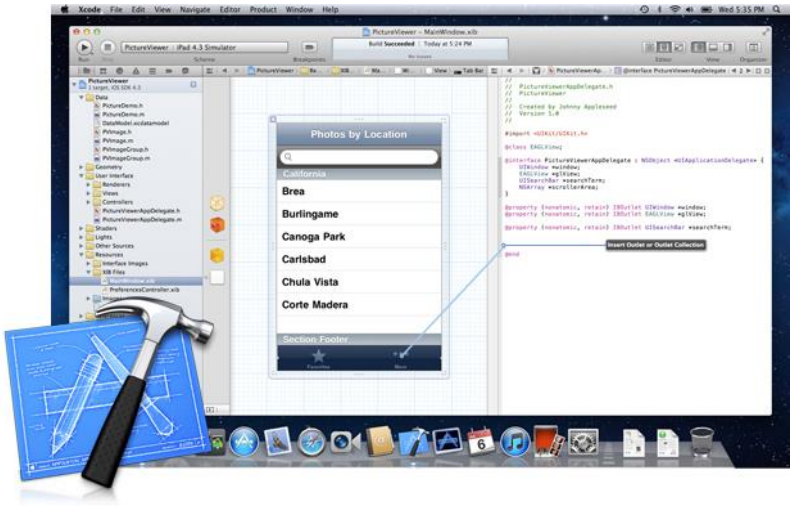
Το Xcode είναι στενά συνδεδεμένο με το Cocoa και το πλαίσιο εργασίας Cocoa Touch το οποίο δημιουργεί ένα παραγωγικό και εύκολο στη χρήση περιβάλλον ανάπτυξης εφαρμογών, το οποίο είναι αρκετά ισχυρό έτσι ώστε να είναι τα ίδια εργαλεία που χρησιμοποιούνται από την Apple για την παραγωγή του OS X και του iOS.

Τα εργαλεία τα οποία περιέχει το iOS SDK τα οποία βοηθούν στην ανάπτυξη των εφαρμογών είναι τα ακόλουθα:

- Xcode IDE
- Apple LLVM Compiler
- Όργανα για ανάλυση της επίδοσης και της συμπεριφοράς (Instruments for Performance and Behavior Analysis)
- iOS Simulator

2.2.1 Xcode IDE

Η κύρια εφαρμογή κατά την ανάπτυξη των iOS εφαρμογών. Χρησιμοποιείται για την επεξεργασία, την εκτέλεση και την αποσφαλμάτωση του κώδικα. Στην Εικόνα 2.4 βλέπουμε ένα στιγμιότυπο από την εφαρμογή αυτή.

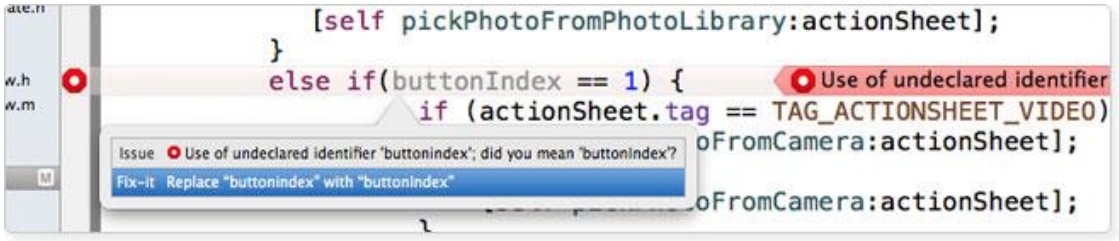


Εικόνα 2.4 Στιγμιότυπο από την εφαρμογή Xcode

2.2.2 Μεταγλωττιστής LLVM

Ο μεταγλωττιστής αυτός είναι της τελευταίας γενιάς, αφού δεν κάνει μόνο απλό κτίσιμο (build) της εφαρμογής. Η LLVM τεχνολογία έχει ενσωματωθεί σε ολόκληρη την εμπειρία της ανάπτυξης. Χρησιμοποιεί τον ίδιο συντακτικό αναλυτή (parser) που χρησιμοποιούν για κτίσιμο οι γλώσσες C / C++, ο οποίος προσφέρει απίστευτα ακριβή

ολοκληρωμένο κώδικα. Καθώς ο προγραμματιστής πληκτρολογεί, ο LLVM μελετά διαρκώς τον κώδικα, για τον εντοπισμό σφαλμάτων κωδικοποίησης, και προτείνει πιθανούς τρόπους για να διορθωθεί το σφάλμα. Στην Εικόνα 2.5 βλέπουμε ένα παράδειγμα διόρθωσης κάποιου σημείου στον κώδικα, από τον LLVM μεταγλωττιστή.



Εικόνα 2.5: Παράδειγμα διόρθωσης από τον LLVM μεταγλωττιστή

2.2.3 Όργανα για ανάλυση επίδοσης και συμπεριφοράς

Οι προγραμματιστές της Apple δίνουν αρκετή βαρύτητα στην εμφάνιση των εφαρμογών, χρησιμοποιώντας ωραίες εικόνες, κινούμενες εικόνες (animations) και έξυπνο σχεδιασμό, έτσι ώστε οι εφαρμογές να ευχάριστες και όσο το δυνατό περισσότερο φιλικές προς τον χρήστη. Σημαντικό, επίσης είναι οι εφαρμογές αυτές να είναι γρήγορες. Τα εργαλεία για προγραμματιστές Xcode περιλαμβάνουν όργανα (instruments), με σκοπό τον εντοπισμό προβλημάτων απόδοσης σε OS X και iOS εφαρμογές. Τα όργανα συλλέγουν δεδομένα, όπως το δίσκο, τη μνήμη, ή τη χρήση του επεξεργαστή (CPU) σε πραγματικό χρόνο. Τα συλλεγόμενα δεδομένα εμφανίζονται γραφικά και ως κομμάτια με την πάροδο του χρόνου, καθιστώντας το εύκολο προς τον προγραμματιστή να εντοπίσει τις προβληματικές περιοχές του κώδικα , έτσι ώστε να κάνει κάποιες διορθώσεις. Στην Εικόνα 2.6 βλέπουμε στιγμιότυπο από το όργανο αυτό.

Εικόνα 2.6: Εργαλεία Instruments

2.2.4 iOS Προσομοιωτής

Ο iOS προσομοιωτής (simulator) τρέχει την εφαρμογή ακριβώς με τον ίδιο τρόπο, που θα έτρεχε πάνω σε μια πραγματική iOS συσκευή. Μπορεί επίσης να προσομοιώσει την αφή χρησιμοποιώντας το ποντίκι. Επίσης ο χρήστης μπορεί να βεβαιωθεί ότι η εφαρμογή του λειτουργεί με το τρόπο που σκοπεύει, όταν περιστρέφεται η συσκευή.

Εικόνα 2.7: Προσομοιωτής iOS

2.3 Γλώσσα προγραμματισμού - Objective C

Η γλώσσα που χρησιμοποιήθηκε για τον προγραμματισμό στην συσκευή iPad είναι η Objective – C [17]. Είναι μια γενικής χρήσης, υψηλού επιπέδου αντικειμενοστρεφής γλώσσα. Χρησιμοποιεί το στυλ των μηνυμάτων της γλώσσας Smalltalk, τα οποία προσθέτει στην γλώσσα προγραμματισμού C. Είναι η κύρια γλώσσα προγραμματισμού που χρησιμοποιείται από την Apple για τα OS X και iOS λειτουργικά συστήματα και των αντίστοιχων API (Application Programming Interface) τους, το Cocoa και Cocoa Touch.

Η γλώσσα αυτή αρχικά αναπτύχθηκε στις αρχές του 1980, από τους Brad Cox και Tom Love και επιλέχθηκε ως η κύρια γλώσσα προγραμματισμού από την NeXT, για το λειτουργικό σύστημα NeXTSTEP, από το οποίο τα OS X και iOS προέρχονται.

Κεφάλαιο 3

Τεχνολογία λογισμικού

3.1 Μοντέλα ανάπτυξης λογισμικών

Στην τεχνολογία λογισμικού, υπάρχουν μια σειρά από μοντέλα ανάπτυξης που χρησιμοποιούνται για την ανάπτυξη λογισμικού. Όταν πρέπει να αναπτυχθεί ένα λογισμικό, αρχικά, υπάρχει μία συζήτηση για το ποιο είναι το πιο κατάλληλο μοντέλο που πρέπει να ακολουθηθεί για να αναπτυχθεί το συγκεκριμένο λογισμικό. Τα μοντέλα τα οποία μπορούν να εφαρμοστούν κατά τη διάρκεια ανάπτυξης των εφαρμογών είναι τα εξής [30]:

- Μοντέλο του καταρράκτη (Waterfall model)
- Σπειροειδής μοντέλο (Spiral model)
- Λειτουργικό μοντέλο (Operational model)
- Μοντέλο πρωτοτυποποίησης (Prototyping model)
- Μοντέλο αυτόματου προγραμματισμού (Automatic programming model)
- Μοντέλο σταδιακής βελτίωσης και επαναληπτικού εμπλουτισμού (Stepwise refinement and iterative enhancement model)
- Ευέλικτη ανάπτυξη (Agile development)
- Μοντέλο της επαναχρησιμοποίησης λογισμικού

Οι δραστηριότητες που απαιτούνται για την ανάπτυξη ενός συστήματος λογισμικού είναι:

- Καθορισμός προδιαγραφών λογισμικού
- Σχεδιασμός λογισμικού
- Επικύρωση – έλεγχος λογισμικού
- Συντήρηση και υποστήριξη λογισμικού

Τα μοντέλα διαδικασιών παραγωγής λογισμικού είναι αφηρημένες αναπαραστάσεις κάποιας διαδικασίας παραγωγής λογισμικού. Επομένως, κάθε μοντέλο παρέχει πληροφορίες για ορισμένες μόνο πλευρές της διαδικασίας.

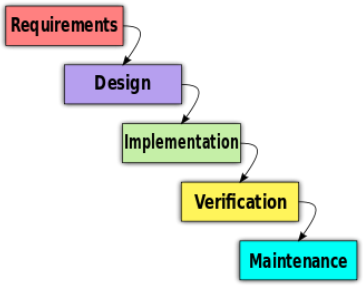
Το μοντέλο το οποίο εφαρμόστηκε στα πλαίσια της διπλωματικής μου εργασίας είναι το μοντέλο του καταρράκτη.

3.2 Μοντέλο που χρησιμοποιήθηκε

Το μοντέλο του καταρράκτη (Waterfall model) [24,25], είναι ένα μοντέλο που χρησιμοποιείται πολύ συχνά στην ανάπτυξη λογισμικού. Ονομάζεται έτσι επειδή χρησιμοποιεί μια προσέγγιση "από πάνω προς τα κάτω" ανεξάρτητα με το αν πρόκειται για ένα μοντέλο καταρράκτη στην δοκιμή, ή ένα μοντέλο καταρράκτη στον κύκλο ζωής ανάπτυξης του λογισμικού. Πρόκειται για ένα μοντέλο κλασσικό, το οποίο είναι από τα πρώτα μοντέλα που εισαχθήκανε στην τεχνολογία λογισμικού. Όπως κάθε άλλο μοντέλο ανάπτυξης, περιέχει μία σειρά από βήματα τα οποία πρέπει να ακολουθηθούν κατά την διάρκεια της ανάπτυξης του.

Τα διαδοχικά βήματα που αποτελούν το μοντέλο αυτό φαίνονται στην Εικόνα 3.1, τα οποία είναι:

1. Ανάλυση και καθορισμός απαιτήσεων
2. Σχεδιασμός συστήματος και λογισμικού
3. Υλοποίηση και δοκιμές υπομονάδων
4. Ενοποίηση και δοκιμές συστήματος
5. Λειτουργία και συντήρηση



Εικόνα 3.1: Βήματα του μοντέλου του καταρράκτη

1. **Ανάλυση και καθορισμός απαιτήσεων**

Αρχικά θα πρέπει να αναλυθεί πλήρως ο ορισμός του προβλήματος και όλες οι απαιτήσεις του έργου. Μόλις κατανοηθούν σε βάθος και εξοντωτικά όλες οι απαιτήσεις του συστήματος, πρέπει να καταγραφούν, έτσι ώστε να ακολουθήσει η επόμενη φάση.

2. **Σχεδιασμός συστήματος και λογισμικού**

Η φάση αυτή περιλαμβάνει τον καθορισμό και το σχεδιασμό των απαιτήσεων του υλικού και του λογισμικού και των σχέσεων μεταξύ τους. Ολόκληρη η πτυχή του λογισμικού του έργου, είναι κατανεμημένη σε διάφορες λογικές ενότητες ή τμήματα, τα οποία καθορίζονται και τεκμηριώνονται συστηματικά.

3. **Υλοποίηση και δοκιμές υπομονάδων**

Αυτή η φάση περιέχει την εγγραφή του κώδικα του λογισμικού, και συγκεκριμένα την εφαρμογή των προγραμματιστικών ιδεών και αλγορίθμων που έχουν σχεδιαστεί και αποφασιστεί στην διάρκεια της προηγούμενης φάσης.

4. **Ενοποίηση και δοκιμές συστήματος**

Μόλις ολοκληρωθεί η φάση της εγγραφής κώδικα και υλοποίησης, στην φάση αυτή δοκιμάζεται ο κώδικας, με σκοπό να καθοριστεί η ορθότητα του. Ο κώδικας που έχει γραφτεί υποβάλλεται σε μια σειρά εξετάσεων και δοκιμών, για να εντοπίσει και να καθορίσει αν υπάρχουν σφάλματα, λάθη ή αποτυχίες στο λογισμικό. Μόλις ολοκληρωθούν όλες οι διορθώσεις στα λάθη τα οποία προέκυψαν, δηλαδή διόρθωση και εκ νέου εγγραφή κάθε κομματιού του λανθασμένου κώδικα, ακολουθεί η επόμενη φάση.

5. Λειτουργία και συντήρηση

Η τελευταία φάση περιλαμβάνει την παράδοση του ολοκληρωμένου έργου στον πελάτη, και στη συνέχεια την εκτέλεση εργασιών συντήρησης σε τακτά χρονικά διαστήματα.

Το μοντέλο αυτό έχει κάποια μειονεκτήματα και πλεονεκτήματα τα οποία αναφέρονται πιο κάτω.

Πλεονεκτήματα

- Είναι το πιο απλό μοντέλο διαδικασίας λογισμικού όσον αφορά την πολυπλοκότητα και την ευκολία της εφαρμογής.
- Είναι εξαιρετικά εύκολο να κατανοηθεί.
- Χρησιμοποιεί μια συστηματική, ορθόδοξη μέθοδο της ανάπτυξης του έργου και την παράδοση.

Μειονεκτήματα

- Δυσκολία στην πραγματοποίηση τροποποιήσεων αφού η διαδικασία έχει ξεκινήσει.
- Πρέπει να ολοκληρωθεί μία φάση για να προχωρήσει η διαδικασία στην επόμενη φάση. Αφού είναι ένα αυστηρώς διαδοχικό μοντέλο, το άλμα εμπρός και πίσω μεταξύ δύο ή περισσοτέρων φάσεων δεν είναι δυνατό.
- Ο άκαμπτος διαμερισμός του έργου σε ξεχωριστά στάδια δυσχεραίνει την ανταπόκριση σε αλλαγές των απαιτήσεων του πελάτη.
- Τα σφάλματα και λάθη στον κώδικα δεν μπορεί να ανακαλυφθούν μέχρι και εάν το δοκιμαστικό στάδιο έχει επιτευχθεί. Αυτό μπορεί να οδηγήσει σε σπατάλη χρόνου και άλλων πολύτιμων πόρων (πχ. χρήμα).
- Το μοντέλο αυτό είναι ακατάλληλο για τα έργα όπου οι απαιτήσεις τους αλλάζουν συνεχώς.

Γιατί χρησιμοποιήθηκε το μοντέλο του καταρράκτη;

Για την ανάπτυξη του λογισμικού **PM JOYSTIC** έχει χρησιμοποιηθεί το μοντέλο του καταρράκτη, αφού στόχος ήταν η δημιουργία ενός λογισμικού που ήδη προυπήρχε.

Αυτό το γεγονός φανέρωσε όλες τις απαιτήσεις του πελάτη. Έτσι ήταν εφικτό να ολοκληρώνεται κάθε φάση με επιτυχία, χωρίς την επιστροφή σε προηγούμενες φάσεις.

Κεφάλαιο 4

Πρωτόκολλο επικοινωνίας

4.1 RCOIP

Το RCOIP [18] είναι ένα πρωτόκολλο που χρησιμοποιείται για να μεταφέρει απομακρυσμένες εντολές από ένα πομπό σε έναν δέκτη μέσω ενός IP μεταφοράς (IP transport), όπως Ethernet ή Wi-Fi. Στα πλαίσια της διπλωματικής μου εργασίας η επικοινωνία γίνεται μέσω Wi-Fi. Είναι ένα πρωτόκολλο 2 – πλευρών (2-way) που καθορίζει μηνύματα UDP (User Datagram Protocol) μεταξύ ενός πομπού και ενός δέκτη.

Αυτό το πρωτόκολλο έχει ως στόχο να μεταφέρει εντολές και μηνύματα κατάστασης ανάμεσα σε ένα ζευγάρι από συσκευές που λειτουργούν σαν συμβατικοί πομποί και δέκτες.

Το αποτέλεσμα είναι ότι οι εντολές που στέλλονται από τον πομπό προς τον δέκτη, προκαλούν στο δέκτη αλλαγή στη συμπεριφορά του (ίσως αλλάζοντας την τιμή μιας αναλογικής εξόδου σε ένα κινητήρα). Σε αντίθεση με τους συμβατικούς (conventional) RC δέκτες, ένας RCOIP δέκτης μπορεί να στείλει επίσης χρήσιμα και ενδιαφέροντα δεδομένα τηλεμετρίας πίσω στο πομπό (αν και με χαμηλότερη προτεραιότητα).

Έτσι, οι εντολές αποστέλλονται από τον πομπό στο δέκτη πάνω από το **UDP**, και περιοδικά σαν απάντηση, στέλνονται απαντήσεις πάνω από UDP από το δέκτη στον πομπό.

Αυτό το γεγονός καθιστά τον πομπό έναν UDP πελάτη και τον δέκτη έναν UDP εξυπηρετητή. Τα μηνύματα που αποστέλλονται από τον πομπό στον δέκτη περιέχουν εντολές οι οποίες ορίζουν κάποιες τιμές στις αναλογικές εξόδους του δέκτη, και τα μηνύματα που αποστέλλονται από τον δέκτη στον αποστολέα περιέχουν πληροφορίες της κατάστασης του δέκτη.

Ο δέκτης είναι συνήθως μία ελαφριά, χαμηλού κόστους συσκευή, με περιορισμένη δυνατότητα υπολογισμού. Δέκτης μπορεί να είναι ένας μικροελεγκτής Arduino με WiShield έτσι ώστε να υποστηρίζει Wi-Fi.

Ο πομπός μπορεί να είναι μια βαρύτερη συσκευή που να υποστηρίζει περισσότερες δυνατότητες, όπως διαδραστικές διεπαφές για τον χρήστη, κλπ. Ένας πομπός θα μπορούσε να είναι ένα iPhone, iPad ή ακόμα και ένα Arduino με WiShield συσκευή.

Τα μηνύματα που αποστέλλονται από τον πομπό στο δέκτη μπορεί να περιλαμβάνουν δεδομένα καναλιού. Κάθε κανάλι είναι μία αναλογική τιμή στο εύρος 0 έως 255. Κάθε κανάλι συνήθως ελέγχεται από κάποια φυσική είσοδο του πομπού. Κάθε κανάλι προκαλεί κάποια φυσική επίδραση στον δέκτη, και συνήθως η τιμή στο κανάλι αλλάζει με τη μετακίνηση κάποιας μονάδας ελέγχου στον πομπό.

Η φυσική ερμηνεία των καναλιών και οι τιμές τους, εξαρτώνται από τη διαμόρφωση (configuration) του δέκτη, και ο πομπός και ο δέκτης αναμένεται να διαμορφωθούν με τέτοιο τρόπο έτσι ώστε να συμφωνούν μεταξύ τους. Για παράδειγμα, σε μια διαμόρφωση, το κανάλι 0 μπορεί να είναι το γκάζι, το κανάλι 1 μπορεί να είναι το τιμόνι και το κανάλι 2 μπορεί να είναι η προσγείωση. Σε μια άλλη διαμόρφωση, το κανάλι 0 μπορεί να είναι το τιμόνι, το κανάλι 1 μπορεί να είναι το γκάζι και το κανάλι 2 μπορεί να είναι το κλειστό / ανοιχτό.

4.2 Ανακάλυψη αποσταλμένων τιμών μέσω Wireshark

Για την ανακάλυψη των τιμών που στέλλονται από την εφαρμογή RCTx χρησιμοποιήθηκε το πρόγραμμα Wireshark [27], ένα ανοιχτού κώδικα λογισμικό ανάλυσης πρωτοκόλλων δικτύου υπολογιστών. Δίναμε μια οποιαδήποτε IP διεύθυνση που βρισκόταν στο δίκτυο μας στην εφαρμογή RCTx, στην οποία η εφαρμογή θα στέλλει τα δεδομένα, έτσι ώστε εμείς μέσω του Wireshark να δούμε τι δεδομένα περιέχουν τα απεσταλμένα πακέτα.

Στο μέρος της εφαρμογής RCTx, που διαμορφώνει τον αποστολέα καθορίσαμε ότι:

- Το κανάλι 0 αφορά τον άξονα x του αριστερού τηλεχειριστηρίου.
- Το κανάλι 1 αφορά τον άξονα y του αριστερού τηλεχειριστηρίου.
- Το κανάλι 2 αφορά τον άξονα x του δεξιού τηλεχειριστηρίου.
- Το κανάλι 3 αφορά τον άξονα y του δεξιού τηλεχειριστηρίου.
- Τα κανάλι 4 - 10 αφορούν τους διακόπτες (switches)

Επίσης γνωρίζουμε ότι το πρωτόκολλο δέχεται τιμές από 0 εώς 255 (καθορίζεται στο RCOIP).

Η πρώτη ψηφιολέξη (byte) που υπάρχει στο πακέτο αντιπροσωπεύει την έκδοση του RCOIP πρωτοκόλλου, όπου στην περίπτωση μας είναι η Έκδοση 1.

Τα δύο τηλεχειριστήρια λειτουργούν με την ίδια λογική, όσο αφορά τις τιμές που στέλλονται στα κανάλια (και στον άξονα x και στον άξονα y). Έτσι στην συνέχεια, θα παρουσιάσουμε μόνο τα πειράματα (κάποια πακέτα) που κάναμε για να βρούμε τις τιμές που στέλλονται μόνο από το αριστερό τηλεχειριστήριο και όχι από το δεξιό. Επίσης, θα δείξουμε τα πειράματα για την ανακάλυψη των τιμών που αφορούν τον τελευταίο διακόπτη, αφού και οι διακόπτες δουλεύουν ακριβώς με την ίδια λογική.

Στα πιο κάτω πακέτα δηλαδή μας ενδιαφέρουν μόνο τα κανάλια 0 και 1 τα οποία αφορούν το αριστερό τηλεχειριστήριο, και το κανάλι 9 το οποίο αφορά τον τελευταίο διακόπτη.

Τα πακέτα που βλέπουμε είναι στο δεκαεξαδικό σύστημα αρίθμησης, και θα γίνει μετατροπή τους στο δεκαδικό με σκοπό να βγάλουμε τα κατάλληλα συμπεράσματα (αφού οι τιμές μας στο δεκαδικό).

Τηλεχειριστήριο

Αρχική θέση του τηλεχειριστηρίου

Το δεδομένα ενός πακέτου που πήραμε όταν το αριστερό τηλεχειριστήριο βρίσκεται στην αρχική του θέση, βρίσκεται στην Εικόνα 4.1.

```
Data (11 bytes)
  Data: 017f7f6ea5000000000000
  [Length: 11]
```

Εικόνα 4.1: Δεδομένα του πακέτου όταν το αριστερό τηλεχειριστήριο βρίσκεται στην αρχική του θέση

Αποτελέσματα

Κανάλι 0 – Δεκαδικό σύστημα	Κανάλι 1 – Δεκαδικό σύστημα
127	127

Κίνηση του τηλεχειριστηρίου από το κέντρο προς τα πάνω

Στο πείραμα αυτό μας ενδιαφέρουν οι τιμές στο κανάλι 1 αφού με την κίνηση προς τα πάνω έχουμε αλλαγές στον άξονα y. Η σειρά κάποιων από τα απεσταλμένα πακέτα βρίσκεται στην Εικόνα 4.2.

```
Data (11 bytes)
  Data: 017d765b9d000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017f5f5fa8000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017f4f5eab000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017f4658ff000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017f3654ff000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017f2552ff000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017f184eff000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017f0f4fff000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017f084eff000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017f004eff000000000000
  [Length: 11]
```

Εικόνα 4.2: Τα δεδομένα 10 τυχαίων πακέτων με την κίνηση του τηλεχειριστηρίου από το κέντρο προς τα πάνω

Αποτελέσματα

Κανάλι 1 – Δεκαδικό σύστημα
118
95
79
70
54
37
24
15
8
0

Με τα πιο πάνω αποτελέσματα παρατηρούμε ότι με την κίνηση του τηλεχειριστηρίου προς τα πάνω μειώνεται η τιμή του καναλιού 1 από το 127 προς το 0.

Κίνηση του τηλεχειριστηρίου από το κέντρο προς τα κάτω

Στο πείραμα αυτό μας ενδιαφέρουν οι τιμές στο κανάλι 1 αφού με τη κίνηση προς τα κάτω έχουμε αλλαγές στον άξονα y. Η σειρά κάποιων από τα απεσταλμένα πακέτα βρίσκεται στην Εικόνα 4.3.

```
Data (11 bytes)
  Data: 017f8f818c000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017fa57f8a000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017fb3808b000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017fbe808c000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017fc8808b000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017fcd808a000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017fd7808b000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017fe5818c000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017ff5818b000000000000
  [Length: 11]
Data (11 bytes)
  Data: 017fff808c000000000000
  [Length: 11]
```

Εικόνα 4.3: Τα δεδομένα 10 τυχαίων πακέτων με την κίνηση του τηλεχειριστηρίου από το κέντρο προς τα κάτω

Αποτελέσματα

Κανάλι 1 – Δεκαδικό σύστημα
143
165
179
190
200
205
215
229
245
255

Με τα πιο πάνω αποτελέσματα παρατηρούμε ότι με την κίνηση προς τα κάτω αυξάνεται η τιμή του καναλιού 1 από το 127 (κέντρο) προς το 255.

Κίνηση του τηλεχειριστηρίου από το κέντρο προς τα δεξιά

Στο πείραμα αυτό μας ενδιαφέρουν οι τιμές στο κανάλι 0 αφού με της κίνηση προς τα δεξιά έχουμε αλλαγές στον άξονα x. Η σειρά κάποιων από τα απεσταλμένα πακέτα βρίσκεται στην Εικόνα 4.4.

```
Data (11 bytes)
  Data: 018b86869a000000000000
  [Length: 11]
Data (11 bytes)
  Data: 019a8179a1000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01a87f6fa1000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01ac7f71a0000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01ba8170a1000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01c18771a4000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01c77f70a1000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01d07f74a0000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01e27f749e000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01ff7f769f000000000000
  [Length: 11]
```

Εικόνα 4.4: Τα δεδομένα 10 τυχαίων πακέτων με την κίνηση του τηλεχειριστηρίου από το κέντρο προς τα δεξιά

Αποτελέσματα

Κανάλι 0 – Δεκαδικό σύστημα
139
145
168
172
186
193
199
208
226
255

Με τα πιο πάνω αποτελέσματα παρατηρούμε ότι με την κίνηση του αριστερού τηλεχειριστηρίου προς τα δεξιά αυξάνεται η τιμή του καναλιού 0 από το 127 προς το 255.

Κίνηση του τηλεχειριστηρίου από το κέντρο προς τα αριστερά

Στο πείραμα αυτό μας ενδιαφέρουν οι τιμές στο κανάλι 0 αφού με της κίνηση προς τα αριστερά έχουμε αλλαγές στον άξονα x. Η σειρά κάποιων από τα απεσταλμένα πακέτα φαίνονται στην Εικόνα 4.5.

```
Data (11 bytes)
  Data: 0173847cd9000000000000
  [Length: 11]
Data (11 bytes)
  Data: 016e80806f000000000000
  [Length: 11]
Data (11 bytes)
  Data: 015b7f8274000000000000
  [Length: 11]
Data (11 bytes)
  Data: 014f7f8476000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01437e857e000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01407f7f9f000000000000
  [Length: 11]
Data (11 bytes)
  Data: 013e8280a1000000000000
  [Length: 11]
Data (11 bytes)
  Data: 012c7e7fa2000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01107f7ca5000000000000
  [Length: 11]
Data (11 bytes)
  Data: 01007d7ea3000000000000
  [Length: 11]
```

Εικόνα 4.5: Τα δεδομένα 10 τυχαίων πακέτων με την κίνηση του τηλεχειριστηρίου από το κέντρο προς τα αριστερά

Αποτελέσματα

Κανάλι 0 – Δεκαδικό σύστημα
115
110
91
79
67
64
62
44
17
0

Με τα πιο πάνω αποτελέσματα παρατηρούμε ότι με την κίνηση προς τα αριστερά μειώνεται η τιμή του καναλιού 0 από το 127 προς το 0.

Διακόπτης

Στο πείραμα αυτό μας ενδιαφέρουν οι τιμές στο κανάλι 9 αφού χρησιμοποιούμε τον τελευταίο διακόπτη της εφαρμογής.

Ο διακόπτης βρίσκεται κάτω (απενεργοποιημένος)

Ένα πακέτο που παίρνουμε στην περίπτωση που ο διακόπτης είναι απενεργοποιημένος φαίνεται στην Εικόνα 4.6.

```
Data (11 bytes)
  Data: 017f7f7696000000000000
  [Length: 11]
```

Εικόνα 4.6: Τα δεδομένα ενός πακέτου όταν ο διακόπτης βρίσκεται κάτω (απενεργοποιημένος)

Ο διακόπτης βρίσκεται πάνω (ενεργοποιημένος)

Ένα πακέτο που παίρνουμε στην περίπτωση που ο διακόπτης είναι ενεργοποιημένος φαίνεται στην Εικόνα 4.7.

```
Data (11 bytes)
  Data: 017f7f6dff000000000001
  [Length: 11]
```

Εικόνα 4.7: Τα δεδομένα ενός πακέτου όταν ο διακόπτης βρίσκεται πάνω (ενεργοποιημένος)

Παρατηρούμε ότι όταν ο διακόπτης είναι προς τα κάτω (απενεργοποιημένος) στέλλεται η τιμή 0 και όταν είναι προς τα πάνω (ενεργοποιημένος) στέλλεται η τιμή 1.

Συμπεράσματα

Όταν το τηλεχειριστήριο κινείται από τα δεξιά προς τα αριστερά έχουμε τις τιμές 0 – 255, και από κάτω προς τα πάνω 255 – 0. Επίσης όταν ο διακόπτης είναι απενεργοποιημένος στέλλεται η τιμή 0 και όταν είναι ενεργοποιημένος στέλλεται η τιμή 1. Στην εφαρμογή PM JOYSTIC όσον αφορά τα τηλεχειριστήρια, καθορίστηκε ότι οι τιμές που θα παίρνουν τα κανάλια των τηλεχειριστηρίων θα είναι από 1 – 255.από τα αριστερά προς τα δεξιά, και 255 – 1 από κάτω προς τα πάνω.

Κεφάλαιο 5

Υλικό – Λογισμικό Ελικοπτέρου

5.1 Μικροελεγκτής ARDUINO

Το Arduino [5,6] είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή και αναλογικές εισόδους / εξόδους. Στην Εικόνα 5.1 μπορούμε να δούμε μία τέτοια πλακέτα.

Η πλακέτα αυτή μπορεί να αισθανθεί το περιβάλλον με την λήψη εισόδων από μια ποικιλία αισθητήρων, και μπορεί να επηρεάσει το περιβάλλον του με τον έλεγχο φώτων, κινητήρων και άλλων ενεργοποιητών.

Μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων (αυτόνομων) διαδραστικών αντικειμένων αλλά επίσης μπορεί να επικοινωνεί με ένα λογισμικό που τρέχει στον υπολογιστή (πχ. Processing, Max/MSP, Pure Data, Flash).

Μία πλακέτα Arduino αποτελείται από ένα μικροελεγκτή Atmel AVR (ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα

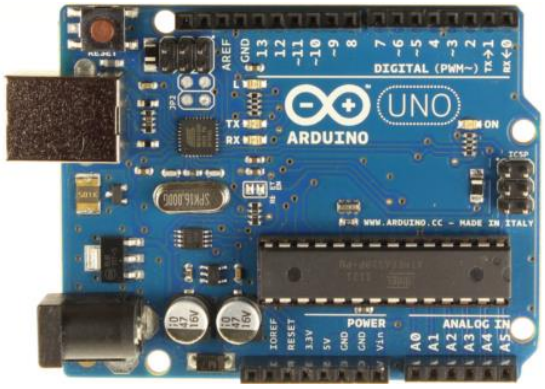
για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα.

Η πλακέτα Arduino προγραμματίζεται με την Arduino γλώσσα προγραμματισμού.

Το περιβάλλον ανάπτυξης του Arduino είναι γραμμένο σε γλώσσα Java και μπορεί να τρέξει σε πολλαπλές πλατφόρμες. Περιλαμβάνει επεξεργαστή κώδικα (επεξεργαστή κειμένου με διάφορα εύχρηστα εργαλεία) και μεταγλωττιστή, και έχει την ικανότητα να φορτώνει εύκολα το πρόγραμμα μέσω σειριακής θύρας από τον υπολογιστή στην πλακέτα. Στιγμιότυπο από το περιβάλλον ανάπτυξης του Arduino βλέπουμε στην Εικόνα 5.2.

Το περιβάλλον ανάπτυξης είναι βασισμένο στην Processing, ένα περιβάλλον ανάπτυξης σχεδιασμένο να εισαγάγει στον προγραμματισμό μη εξοικειωμένους χρήστες με την ανάπτυξη λογισμικού. Η συγκεκριμένη γλώσσα προγραμματισμού προέρχεται από την Wiring, μια γλώσσα που μοιάζει με την C η οποία παρέχει παρόμοια λειτουργικότητα για μια πιο περιορισμένης σχεδίασης πλακέτα, της οποίας το περιβάλλον ανάπτυξης βασίζεται επίσης στην Processing.

Ο κώδικας που γράφει ο προγραμματιστής φορτώνεται στην πλακέτα μέσω της σειριακής εξόδου του υπολογιστή (serial port).



Εικόνα 5.1: Πλακέτα Arduino

Εικόνα 5.2: Περιβάλλον ανάπτυξης του κώδικα για πλακέτες Arduino

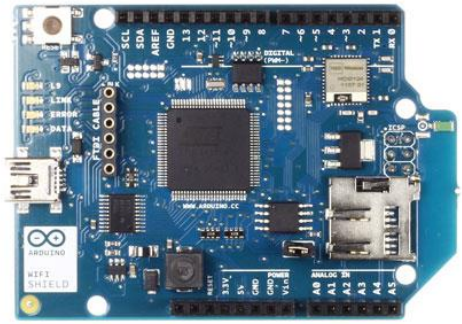
5.2 Ασπίδα Wi-fi

Η ασπίδα Wi-Fi (Wi-fi shield) [4,7] επιτρέπει στην πλακέτα Arduino να συνδεθεί στο διαδίκτυο, χρησιμοποιώντας την ασύρματη προδιαγραφή (wireless specification) 802.11 (Wi-Fi). Στην Εικόνα 5.3 βλέπουμε την πρόσοψη της ασπίδας Wi-Fi και στην Εικόνα 5.4 βλέπουμε την πίσω όψη της. Η Ασπίδα Wi-Fi μπορεί να συνδεθεί με ασύρματα δίκτυα που λειτουργούν σύμφωνα με τις 802.11b και 802.11g προδιαγραφές. Περιέχει τον μικροελεγκτή 32-bit ATMega 32UC3, ο οποίος υποστηρίζει UDP (User Datagram Protocol) και TCP (Transition Control Protocol).

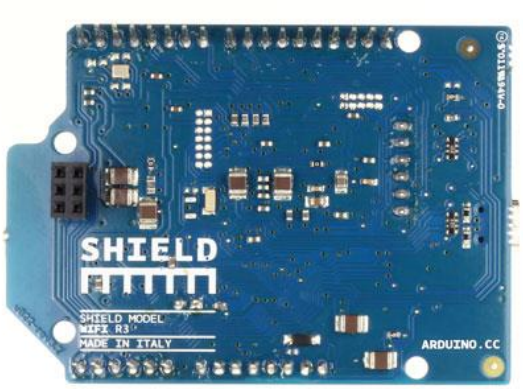
Για την εγγραφή του κώδικα είναι απαραίτητη η χρήση της βιβλιοθήκης Wi-Fi library , η οποία συνδέεται στο διαδίκτυο χρησιμοποιώντας την ασπίδα. Η ασπίδα Wi-Fi συνδέεται στην πλακέτα Arduino χρησιμοποιώντας wire-wrap headers, τα οποία εκτείνονται μέσα στο Wi-fi shield. Αυτό κρατά την διάταξη των pins άθικτη και έτσι επιτρέπεται και σε άλλες ασπίδες να στοιβάζονται στην κορυφή της ασπίδας. Επίσης στην Wi-Fi ασπίδα υπάρχει μια micro – SD υποδοχή κάρτας η οποία χρησιμοποιείται για την αποθήκευση αρχείων για σερβίρισμα μέσω διαδικτύου.

Για την πρόσβαση (διάβασμα) στα δεδομένα της micro SD κάρτας είναι απαραίτητη η χρήση της βιβλιοθήκης SD. Η ασπίδα Wi-Fi είναι συμβατή με πλακέτες Arduino Uno και Mega. Η πλακέτα Arduino επικοινωνεί με την ασπίδα και την κάρτα SD,

χρησιμοποιώντας τον δίαυλο SPI (Serial Peripheral Interface). Η ασπίδα μπορεί να συνδεθεί σε κρυπτογραφημένα δίκτυα (encrypted networks) που χρησιμοποιούν είτε WPA2 Personal ή WEP κρυπτογράφηση. Επίσης μπορεί να συνδεθεί σε ανοιχτά δίκτυα.



Εικόνα 5.3: Wi-Fi shield πρόσοψη



Εικόνα 5.4: Wi-Fi shield πίσω όψη

5.3 Βιβλιοθήκη RCKit

Η βιβλιοθήκη RCKit [18], παρέχει ένα σετ από αντικείμενα λογισμικού που καθιστούν εύκολη την δημιουργία ενός RCOIP (Remote Control Over IP) δέκτη Arduino. Λεπτομέρειες που αφορούν το πρωτόκολλο επικοινωνίας RCOIP έχουν αναφερθεί στο Κεφάλαιο 4.1. Η βιβλιοθήκη αυτή αποτελείται από διάφορες κλάσεις, όπου θα δούμε κάποιες από αυτές στην συνέχεια.

Η κλάση **RCRx** αντιπροσωπεύει ένα δέκτη RCOIP. Όταν δημιουργηθεί ένα αντικείμενο της κλάσης RCRx, πρέπει να δοθεί μια σειρά από αντικείμενα ορισμού (setters). Όταν ληφθεί ένα μήνυμα της δομής RCOIPv1CmdSetAnalogChannels από το αντικείμενο του τύπου RCRx, το αντικείμενο ορισμού που αντιστοιχεί σε κάθε τιμή καναλιού της δομής (struct) RCOIPv1CmdSetAnalogChannels, θα περάσει στο αντίστοιχο αντικείμενο ορισμού (setter) στον πίνακα των αναλογικών εξόδων. Το λογισμικό Arduino αναμένεται να διαμορφωθεί έτσι ώστε κάθε αναλογική έξοδος να συνδέεται με ένα αντικείμενο ορισμού που θα εφαρμόζει την εισαχθείσα τιμή σε μια αναλογική ή ψηφιακή έξοδο ενός pin του Arduino.

Πρέπει να αναφέρουμε ότι η **δομή (struct) RCOIPv1CmdSetAnalogChannels** χρησιμοποιήθηκε στον κώδικα της Objective – C. Η δομή αυτή καθορίζει την μορφή του μηνύματος που πρέπει να στείλει ο αποστολέας στο δέκτη – Arduino.

Τα παρακάτω αντικείμενα ορισμού (setter) παρέχονται στο RCKit. Μπορούν να χρησιμοποιηθούν για μετάφραση των τιμών του κάθε καναλιού σε φυσικές εξόδους:

- AnalogSetter
- DigitalSetter
- ServoSetter
- AccelStepperPositionSetter
- AccelStepperSpeedSetter
- HBridgeSetter
- DifferentialSetter

Κεφάλαιο 6

Ανάπτυξη της εφαρμογής

6.1 Εργαλεία ανάπτυξης (υλικό / λογισμικό)

Τα εργαλεία (υλικό / λογισμικό) που χρησιμοποιήθηκαν για δημιουργία και έλεγχο της εφαρμογής ήταν:

- Υπολογιστής MacBook Air
- Συσκευή iPad
- Xcode έκδοση 4.6.2
- iPad Simulator 6.1

6.2 Πλαίσια εργασίας και κινητήρες φυσικής

Για την ανάπτυξη της εφαρμογής, χρησιμοποιήθηκαν κάποια πλαίσια εργασίας (frameworks) και προσομοιωτές κινητήρων φυσικής (Physics engines).

Το πλαίσιο εργασίας (Framework) είναι ένα επαναχρησιμοποιήσιμο σύνολο από διεπαφές (interfaces) τα οποία χρησιμοποιούνται στην ανάπτυξη μιας εφαρμογής. Μπορεί να περιέχει αρχεία κεφαλίδας (header files), βιβλιοθήκες, εικόνες , ήχους και πολλά άλλα.

Οι κινητήρες φυσικής (Physics engine) είναι λογισμικά που παρέχουν μια κατά προσέγγιση προσομοίωση ορισμένων φυσικών συστημάτων, όπως η άκαμπτη (rigid) δυναμική των σωμάτων (συμπεριλαμβανομένου της ανίχνευσης σύγκρουσης), η μαλακή (soft) δυναμική των σωμάτων, και η ρευστότητα (fluid).

Τα πλαίσια και οι κινητήρες φυσικής που χρησιμοποιήθηκαν είναι τα πιο κάτω:

- Πλαίσιο εργασίας CoreGraphics.
- Πλαίσιο εργασίας OpenGLES.
- Πλαίσιο εργασίας QuartzCore.
- Πλαίσιο εργασίας UIKit.
- Πλαίσιο εργασίας AVFoundation.
- Πλαίσιο εργασίας Cocos2D.
- Κινητήρας φυσικής Box2D.

Στην συνέχεια θα δούμε κάποιες πληροφορίες για τα πιο πάνω, καθώς και τους λόγους για τους οποίους κρίθηκε απαραίτητο να χρησιμοποιηθούν.

6.2.1 Πλαίσιο εργασίας CoreGraphics

Το πλαίσιο εργασίας CoreGraphics [3] περιέχει τις διεπαφές για προγραμματισμό εφαρμογών Quartz 2D (API). Παρέχει υποστήριξη για γραμμικό (path-based) σχεδιασμό, απόδοση βελτίωσης παρουσίασης γραφικών (anti-aliased rendering), κλίσεις (gradients), εικόνες, χρώματα, μετασχηματισμούς συντεταγμένων στο χώρο (coordinate-space transformations).

6.2.2 Πλαίσιο εργασίας OpenGLES

Το πλαίσιο εργασίας OpenGLES [3] παρέχει μια κλάση βασισμένη στην C που χρησιμοποιείται για την δημιουργία δυσδιάστατων (2D) και τρισδιάστατων (3D) γραφικών (rendering graphics). Η συλλογή των αρχείων αυτών είναι γνωστή επίσης ως EAGL. Η EAGL παρέχει όλα τα γραφικά περιβάλλοντα που περιέχουν οι βιβλιοθήκες OpenGl ES για διαμόρφωση του επιπέδου Core Animation. Η EAGL επιτρέπει επίσης στα OpenGL ES αντικείμενα όπως για παράδειγμα σχηματομορφές (textures), πλαίσια απόδοσης (renderbuffers) και πλαίσια αποθήκευσης (framebuffers) να μοιράζονται σε 2 ή περισσότερα γραφικά περιβάλλοντα.

6.2.3 Πλαίσιο εργασίας QuartzCore

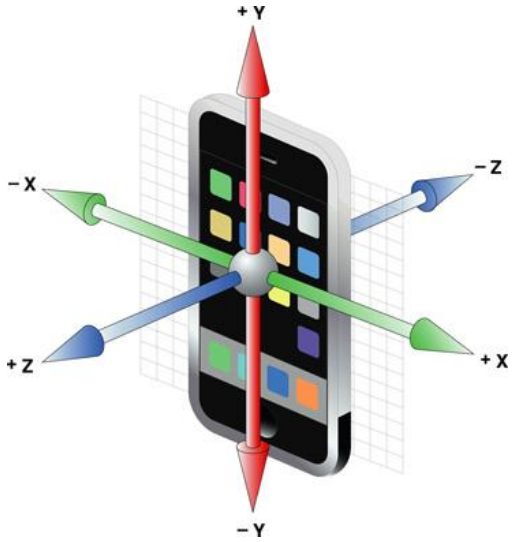
Το πλαίσιο εργασίας QuartzCore [3] περιέχει όλες τις κλάσεις του Core Animation. Το Core Animation είναι μια προηγμένη τεχνολογία κίνησης και σύνθεσης εικόνας (animation and compositioning) που χρησιμοποιεί μια βελτιστοποιημένη διαδρομή απόδοσης (rendering) για την σύνθεση σύνθετων κινήσεων εικόνας (animations) και οπτικών εφέ. Παρέχει υψηλού επιπέδου κλάσεις για την διαμόρφωση των κινήσεων εικόνας και των εφέ. Το Core Animation είναι ενσωματωμένο σε πολλά μέρη του συστήματος, όπως τις κλάσεις UIKit.

6.2.4 Πλαίσιο εργασίας UIKit

Το πλαίσιο εργασίας UIKit [3] παρέχει τις κλάσεις που απαιτούνται για την κατασκευή και την διαχείριση της διεπαφής του χρήστη. Παρέχει ένα αντικείμενο της εφαρμογής (application object), χειρισμό συμβάντων (event handling), σχεδιασμός του μοντέλου (drawing model), παράθυρα (windows), προβολές και ελέγχους που σχεδιάστηκαν ειδικά για την οθόνη αφής (touch screen).

Το πλαίσιο εργασίας αυτό είναι απαραίτητο για την δημιουργία και για την αλληλεπίδραση με την διεπαφή. Χρησιμοποιώντας αυτό το πλαίσιο εργασίας μπορούσαμε για παράδειγμα να βρούμε το σημείο στην οθόνη όπου αγγίζει ο χρήστης, χρησιμοποιώντας τις κλάσεις **UITouch** και **UIEvent**. Επίσης είναι σημαντικό να

αναφέρουμε ότι τα μηνύματα που εμφανίζονται στην οθόνη για καλύτερη ενημέρωση του χρήστη για την κατάσταση της εφαρμογής έγινε η χρήση της κλάσης **UIAlertView.** Επίσης μέσω της κλάσης **UIAccelerometer** μπορούμε να βρούμε την θέση της συσκευής στο χώρο των τριών διαστάσεων. Ανάλογα με τις τιμές αυτές μπορούμε να καθορίσουμε την κίνηση του ελικόπτερου. Οι τρεις άξονες όπως απεικονίζονται στην συσκευή φαίνονται στην Εικόνα 6.1.



Εικόνα 6.1: Διαστάσεις στον τρισδιάστατο χώρο πάνω σε συσκευή iPad

6.2.5 Πλαίσιο εργασίας AVFoundation

Το AVFoundation [3] είναι ένα πλαίσιο εργασίας που χρησιμοποιείται για την αναπαραγωγή και την εγγραφή ήχου και βίντεο.

Συγκεκριμένα στην εφαρμογή που αναπτύχθηκε κρίθηκε καλό να παράγεται κάποιος ήχος όταν ο χρήστης πατάει ένα κουμπί. Συγκεκριμένα έγινε χρήση της κλάσης **AVAudioPlayer**. Έτσι δίνοντας το αρχείο του ήχου που θέλαμε να παραχθεί, μπορούσαμε να καθορίσουμε ακόμη και την δύναμη του ήχου όπως και επίσης πόσες φορές θα παραχθεί.

6.2.6. Πλαίσιο εργασίας Cocos2

Το Cocos2d [9] είναι ένα πλαίσιο εργασίας, ανοιχτού κώδικα (open source) για την δημιουργία δυσδιάστατων (2D) παιχνιδιών και άλλων γραφικών και διαδραστικών παιχνιδιών.

Γενικές πληροφορίες

Το αρχικό πλαίσιο εργασίας Cocos2D είναι γραμμένο σε Python αλλά έκτοτε έχει μεταφερθεί και σε άλλες γλώσσες και πλατφόρμες (για παράδειγμα γράφτηκε στις γλώσσες C++, C#,Go, Java, JavaScript και Objective-C). Περισσότερα από 2500 παιχνίδια του AppStore χρησιμοποιούν ήδη αυτό το πλαίσιο εργασίας, συμπεριλαμβανομένων και πολλών παιχνιδιών με τις καλύτερες πωλήσεις (best - seller).

Το πλαίσιο εργασίας αυτό χρησιμοποιήθηκε στην εφαρμογή για την δημιουργία των διεπαφών και των κουμπιών. Για την δημιουργία της διεπαφής (μενού) χρησιμοποιήθηκαν οι κλάσεις **CCScene** και **CCLayer**. Για την δημιουργία των απλών κουμπιών (δηλαδή των κουμπιών που όταν τα πατήσουμε δεν αλλάζουν μορφή) χρησιμοποιήθηκε ένας συνδυασμός των κλάσεων **CCMenuItem**, **CCMenu** και για τα υπόλοιπα κουμπιά χρησιμοποιήθηκε μια επιπλέον κλάση η **CCMenuItemToggle**. Για τα αντικείμενα που δεν έχουν κάποια άμεση επαφή με τον χρήστη χρησιμοποιήθηκε η κλάση **CCSprite** (πχ για το φόντο). Επιπλέον, μέσω του πλαισίου εργασίας αυτού μπορέσαμε να ενεργοποιήσουμε το άγγιγμα (touch enable) της οθόνης.

Εικόνα 6.2: Λογότυπο του πλαισίου εργασίας Cocos2d

6.2.7 Κινητήρας φυσικής Box2D

Το Box2D [8] είναι ανοικτού κώδικα προσωμοιωτής κινητήρα φυσικής.

Γενικές πληροφορίες

Γράφτηκε σε Objective - C από τον Erin Catto και δημοσιεύτηκε κάτω από άδεια χρήσης zlib. Χρησιμοποιήθηκε σε πολλά γνωστά παιχνίδια όπως, Crayon Physics Deluxe,Limbo, Fantastic Contraption, Incredibots, Angry Birds, Tiny Wings, Happy Wheels, και σε πολλά άλλα διαδικτυακά Flash παιχνίδια, όπως και επίσης σε iPhone, iPad και παιχνίδια Android χρησιμοποιώντας τη μηχανή παιχνιδιών Cocos2D και το πλαίσιο εργασίας Corona.

Το πλαίσιο εργασίας αυτό χρησιμοποιήθηκε για τον χειρισμό των δύο τηλεχειριστηρίων. Με την δημιουργία **σωμάτων (bodies)** μπορούμε μέσω μίας συνάρτησης να βρούμε εύκολα αν ο χρήστης άγγιξε το τηλεχειριστήριο. Για να δούμε πιο λεπτομερώς και να κατανοήσουμε την χρήση του πλαισίου εργασίας αυτού θα παρουσιάσουμε και θα επεξηγήσουμε το συγκεκριμένο κομμάτι κώδικα της εφαρμογής στο Κεφάλαιο 8.3.

Εικόνα 6.3: Λογότυπο του Box2D framework

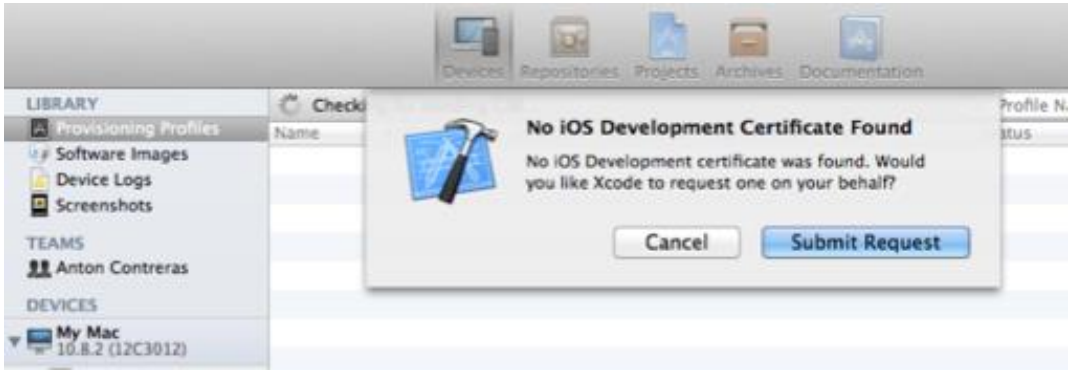
6.3 Τροφοδότηση συσκευής για ανάπτυξη

Αν και η εφαρμογή μπορεί να ελεγχθεί κατά την ανάπτυξη της μέσω του iOS προσομοιωτή, δεν μπορούν να ελεγχθούν αποτελεσματικά όλες οι πτυχές της εφαρμογής χωρίς δοκιμή σε μια συσκευή. Ένας κύριος λόγος που κρίθηκε απαραίτητο ο έλεγχος της εφαρμογής πάνω στην συσκευή, είναι ότι η εφαρμογή που αναπτύχθηκε κάνει χρήση του **Accelerometer** κάτι που ο προσομοιωτής iOS δεν μπορεί να προσομοιώσει, δηλαδή, δεν μπορούμε να δώσουμε κίνηση στον προσομοιωτή στο χώρο των τριών διαστάσεων. Επίσης, η εφαρμογή είναι **πολυαφής** (multitouch), όπου επίσης δεν μπορεί να ελεγχθεί στον προσομοιωτή.

Πιο κάτω θα δοθούν τα βήματα που ακολουθήθηκαν για το τρέξιμο της εφαρμογής πάνω στην συσκευή iPad.

Βήμα 1 – Αίτηση για πιστοποιητικό ανάπτυξης (Development Certificate)
Αρχικά θα πρέπει να κάνουμε αίτηση για πιστοποιητικό, όπως φαίνεται στην Εικόνα 6.4. Όταν ανανεώνουμε τα προφίλ εφοδιασμού (Provisioning Profiles), το Xcode δημιουργεί την υπογραφή του πιστοποιητικού. Το Xcode δημιουργεί τόσο το πιστοποιητικό για την ανάπτυξη όσο και το πιστοποιητικό για την διανομή για το λογαριασμό του προγραμματιστή, και το προσθέτει αυτόματα στην αλυσίδα κλειδιών (keychain). Τα βήματα είναι τα εξής:

1. Window → Organizer.
2. Επιλέγουμε Devices
3. Library section → Provisioning Profiles
4. Πατούμε το ανανέωση (refresh) που βρίσκεται κάτω δεξιά
5. Εισαγωγή του ονόματος χρήστη (user name) και του συνθηματικού (password), πατάμε σύνδεση (login)
6. Μετά την είσοδο εμφανίζεται μια προτροπή (prompt) και πατάμε το κουμπί Submit Request για να στείλουμε αίτηση πιστοποιητικού ανάπτυξης



Εικόνα 6.4: Αίτηση για πιστοποιητικό

Το πιστοποιητικό ανάπτυξης προστίθεται στην αλυσίδα κλειδιών και αργότερα προστίθεται στο προφίλ εφοδιασμού της ομάδας (iOS Provisioning Profile).

Βήμα 2 – Παροχή (provision) της συσκευής

Την πρώτη φορά που προσθέτεται μια συσκευή (Εικόνα 6.5), συνδέοντας την συσκευή στον υπολογιστή, το Xcode δημιουργεί το προφίλ εφοδιασμού ομάδας iOS (Team iOS Provisioning Profile) χρησιμοποιώντας το App ID, το πιστοποιητικό της συσκευής και το ID της συσκευής.

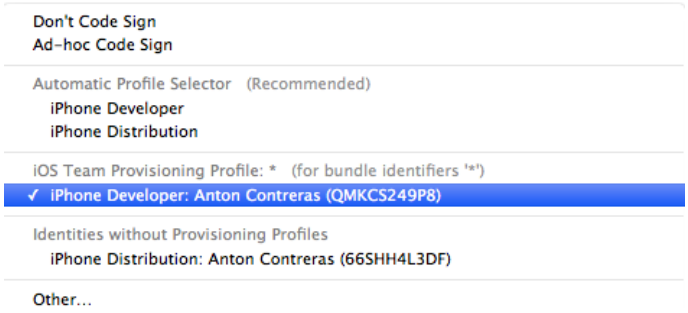
Στην συνέχεια εγκαθιστά αυτόματα στη συσκευή στο προφίλ εφοδιασμού (Provisioning Profile.)

1. Σύνδεση της συσκευής με τον υπολογιστή
2. Window → Organizer → Devices
3. Στην ενότητα Device επιλέγουμε την συσκευή iOS
4. Πατούμε το κουμπί "Use for Development", όπως φαίνεται στην Εικόνα 6.5

Εικόνα 6.5: Χρήση για ανάπτυξη της συγκεκριμένης συσκευής

Βήμα 3 – Εισαγωγή κωδικού στην εφαρμογή (code sign)

1. Επιλέγουμε το Project
2. Επιλέγουμε Build Settings
3. Επιλέγουμε All
4. Πληκτρολογούμε Code Signing στο πεδίο αναζήτησης
5. Από το αναδυόμενο μενού Code Signing Identity, στο μέρος iOS Team Provisioning Profile, επιλέγουμε το πιστοποιητικό που ξεκινά από iPhone Developer, όπως φαίνεται στην Εικόνα 6.6.



Εικόνα 6.6: Επιλογή του προφίλ του προγραμματιστή

Βήμα 4 – Εκκίνηση της εφαρμογής στην συσκευή

Η εκκίνηση της εφαρμογής στην συσκευή μπορεί να γίνει μέσω του XCode.

1. Επιλέγουμε Product > Scheme > Edit Scheme
2. Επιλέγουμε την συσκευή από το αναδυόμενο μενού, όπως φαίνεται και στην Εικόνα 6.7.
3. Πατούμε OK και στην συνέχεια Run, και η εφαρμογή τρέχει στην συσκευή που επιλέξαμε.


Εικόνα 6.6: Επιλογή της συσκευής πάνω στην οποία θα τρέξει η εφαρμογή

Κεφάλαιο 7

Εφαρμογή

7.1 Παρουσίαση εφαρμογής

Η αρχική διεπαφή που εμφανίζεται στον χρήστη όταν ξεκινήσει την εφαρμογή φαίνεται στην Εικόνα 7.1.

Εικόνα 7.1: Αρχική οθόνη της εφαρμογής PM JOYSTIC

Με το πάτημα του δεύτερου κουμπιού στη μέση της οθόνης της διεπαφής, ενεργοποιείται η κίνηση του ελικοπτέρου με την κίνηση του iPad, και εμφανίζεται ένα ενημερωτικό μήνυμα στην οθόνη, όπως φαίνεται στην Εικόνα 7.2. Με το πάτημα ξανά του κουμπιού αυτού απενεργοποιείται η δυνατότητα αυτή, εμφανίζοντας και πάλι το ανάλογο μήνυμα, όπως φαίνεται στην Εικόνα 7.3.

Εικόνα 7.2: Ενεργοποίηση της κίνησης του ελικοπτέρου και εμφάνιση του ανάλογου μηνύματος στον χρήστη

Εικόνα 7.3: Απενεργοποίηση της κίνησης του ελικοπτέρου και εμφάνιση του ανάλογου μηνύματος στον χρήστη

Με το πάτημα του μεσαίου κουμπιού στο πάνω μέρος της διεπαφής απενεργοποιείται ο ήχος που παράγει η εφαρμογή, εμφανίζοντας το αντίστοιχο μήνυμα στην οθόνη, όπως φαίνεται στην Εικόνα 7.4. Με το πάτημα ξανά του κουμπιού αυτού ενεργοποιείται ο ήχος, δίνοντας το αντίστοιχο μήνυμα όπως φαίνεται στην Εικόνα 7.5.

Εικόνα 7.4: Απενεργοποίηση του ήχου με το ανάλογο ενημερωτικό μήνυμα

Εικόνα 7.5: Ενεργοποίηση του ήχου με το ανάλογο ενημερωτικό μήνυμα

Με το πάτημα του δεξιού κουμπιού στο πάνω μέρος της οθόνης δίνεται η δυνατότητα στο χρήστη να τερματίσει την εφαρμογή. Εμφανίζεται το μήνυμα “Are you sure you want to exit?” στην οθόνη όπως φαίνεται στην Εικόνα 7.6, και ο χρήστης επιλέγει “Yes” αν επιθυμεί να τερματίσει την εφαρμογή ή “No” αν θέλει να ακυρώσει την ενέργεια αυτή.

Εικόνα 7.6: Μήνυμα που αναμένει βεβαίωση από τον χρήστη, για έξοδο από την εφαρμογή

Με το πάτημα του πρώτου κουμπιού, που βρίσκεται στη μέσης της οθόνης, κάθε φορά, αλλάζει το χρώμα γαλάζιο σε κόκκινο και το αντίθετο. Στην περίπτωση που το κουμπί έχει χρώμα γαλάζιο (Εικόνα 7.7) είναι ενεργοποιημένο και στην περίπτωση που έχει χρώμα κόκκινο (Εικόνα 7.8) είναι απενεργοποιημένο.

Εικόνα 7.7: Ενεργοποιημένο κουμπί χρώματος γαλάζιο

Εικόνα 7.8: Απενεργοποιημένο κουμπί χρώματος κόκκινο

Με το πάτημα του πρώτου κουμπιού στο πάνω μέρος της αρχικής οθόνης που βλέπουμε στην Εικόνα 7.8, ο χρήστης μπορεί να δώσει την διεύθυνση IP του Arduino και τον αριθμό του port όπου θα γίνει η επικοινωνία (Εικόνα 7.9).

Εικόνα 7.9: Οθόνη για ρυθμίσεις της επικοινωνίας

7.2 Ευχρηστία εφαρμογών

7.2.1 Τι είναι HCI;

Η αλληλεπίδραση ανθρώπου-υπολογιστή (HCI) [1,13] είναι το επιστημονικό πεδίο που μελετά την αλληλεπίδραση μεταξύ των χρηστών και των υπολογιστών (εφαρμογών). Η αλληλεπίδραση μεταξύ ανθρώπου (χρήστη) και υπολογιστή γίνεται στο επίπεδο της διεπαφής χρήστη (user interface), μέσω του κατάλληλου λογισμικού και υλικού.

Στόχος της αλληλεπίδρασης ανθρώπου-υπολογιστή είναι να βελτιώσει την επικοινωνία μεταξύ χρηστών και υπολογιστών (εφαρμογών), μέσω της ορθής σχεδίασης εύχρηστων και εργονομικών υπολογιστών, προσανατολισμένων στην ανθρωποκεντρική σχεδίαση. Παρέχει κατανόηση των αναγκών των χρηστών, οδηγίες ορθού σχεδιασμού διεπαφών, μεθόδους αξιολόγησης χρηστικότητας υπολογιστικών συστημάτων και μία βάση γνώσεων για τις διαθέσιμες τεχνολογίες αλληλεπίδρασης. Τα πεδία έρευνας της αλληλεπίδρασης ανθρώπου υπολογιστή είναι τα εξής:

- μεθοδολογίες σχεδίασης διεπαφών (π.χ. για δεδομένη εργασία και ομάδα χρηστών, τη σχεδίαση της βέλτιστης διασύνδεσης χρήστη)
- μεθόδους υλοποίησης διεπαφών
- τεχνικές σύγκρισης και αξιολόγησης διεπαφών
- ανάπτυξη νέων διεπαφών και τεχνικών αλληλεπίδρασης
- ανάπτυξη περιγραφικών και προβλεπτικών μοντέλων και θεωριών αλληλεπίδρασης

Με την αλληλεπίδραση ανθρώπου-υπολογιστή ασχολούνται και σχεδιαστές, οι οποίοι μελετούν την πρακτική εφαρμογή μεθοδολογιών σχεδίασης σε προβλήματα του πραγματικού κόσμου. Η εργασία τους αφορά τον σχεδιασμό γραφικών διεπαφών χρήστη και δικτυακών εφαρμογών.

7.2.2 Τι είναι Ευχρηστία;

Σύμφωνα με το πρότυπο ISO 9241-11, ευχρηστία [29] είναι η δυνατότητα ενός προϊόντος συστήματος ή υπηρεσίας που χρησιμοποιείται από καθορισμένους χρήστες με καθορισμένους στόχους, υπό καθορισμένες συνθήκες χρήσης, να παρέχει αποτελεσματικότητα, αποδοτικότητα και υποκειμενική ικανοποίηση στους χρήστες του.

Η κυριότερη μέθοδος αξιολόγησης είναι η ευρετική αξιολόγηση, η οποία αναπτύχθηκε από τον Jacob Nielsen στις αρχές της δεκαετίας του 90.

7.2.3 Ευρετικά του Nielsen

Οι δέκα συνηθέστεροι κανόνες στην ευρετική αξιολόγησης δημιουργήθηκαν από τον Nielsen [26] και είναι οι πιο κάτω:

1. **Χρήση κατανοητής προς τους χρήστες γλώσσα**

Η γλώσσα που χρησιμοποιείται στον υπολογιστή καθώς και οι εκφράσεις, οι έννοιες και οι όροι, θα πρέπει να συμβαδίζει με τη γλώσσα του χρήστη και να είναι κατανοητή από αυτόν.

2. **Παροχή ανάδρασης (feedback)**

Το σύστημα πρέπει να παρέχει σε εύλογο χρόνο ενημέρωση στους χρήστες για την εξέλιξη των εργασιών που δέχεται ο υπολογιστής.

3. **Έλεγχος και ελευθερία χρήστη**

Είναι απαραίτητη η ύπαρξη σαφών και εύκολων διεξόδων (π.χ. κουμπιά) στην περίπτωση που ο χρήστης προβεί σε αθέμιτες ενέργειες.

4. **Αποφυγή περιττών στοιχείων (μινιμαλισμός)**

Δεν πρέπει να υπάρχουν πληροφορίες και λεπτομέρειες οι οποίες δεν χρειάζονται και δεν βοηθούν πουθενά. Είναι χρήσιμο να παραλείπονται για να μην προκαλούν σύγχυση και να βαραίνουν τον χρήστη.

5. **Αναγνώριση και όχι ανάκληση**

Ελαχιστοποίηση του μνημονικού φόρτου του χρήστη. Οι ενέργειες θα πρέπει να είναι φανερές χωρίς να απαιτούν από τον χρήστη να θυμάται εντολές και λεπτομέρειες.

6. **Συνοχή και συνέπεια**

Δεν πρέπει να αναρωτιέται ο χρήστης αν οι φράσεις που χρησιμοποιούνται πραγματοποιούν την ίδια ενέργεια, με τον ίδιο τρόπο πάντα σε κάθε διεπιφάνεια.

7. **Παροχή συντομεύσεων (shortcuts)**

Πρέπει να υπάρχει δυνατότητα πραγματοποίησης ενεργειών που ίσως χρησιμοποιούνται συχνά από τους έμπειρους γρηγορότερα, οι οποίες είναι αόρατες στους "αρχάριους" χρήστες.

8. **Πρόβλεψη σφαλμάτων**

Εμφάνιση μηνυμάτων λάθους σε κατανοητή προς το χρήστη γλώσσα, ύπαρξη διεξόδων και λύσης τους δεδομένου προβλήματος.

9. **Αποφυγή λαθών**

Έλεγχος των συνδέσμων, έλεγχος εγκυρότητας και αποφυγή συνδέσμων που δεν οδηγούν σε κανένα αποτέλεσμα.

10. **Βοήθεια και τεκμηρίωση**

Ύπαρξη βοηθητικού υλικού για την ευκολότερη πλοήγηση του χρήστη καθώς και επεξεργασία ή βοήθεια για την επίτευξη των επιθυμητών ενεργειών.

7.3 Ανάλυση ευχρηστίας εφαρμογής PM JOYSTIC

1. Κατά την έξοδο του χρήστη από την εφαρμογή εμφανίζεται ένα μήνυμα στην οθόνη έτσι ώστε να επιβεβαιώσει ότι θέλει να κλείσει την εφαρμογή (Εικόνα 7.10). Αν ο χρήστης πατούσε το κουμπί εξόδου κατα λάθος θα αποφευγόταν να κλείσει η εφαρμογή κατευθείαν. Έτσι με τον τρόπο αυτό **αποφεύγονται τα λάθη**.

Εικόνα 7.10: Μήνυμα επιβεβαίωσης εξόδου από την εφαρμογή

2. Όταν πατήσει ο χρήστης ένα κουμπί εμφανίζονται **μηνύματα που ενημερώνουν** τον χρήστη για το τι συμβαίνει στο σύστημα. Μηνύματα ενημέρωσης εμφανίζονται όταν:

    - Ενεργοποιήσουμε την κίνηση με το iPad (Εικόνα 7.11)
    - Απενεργοποιήσουμε την κίνηση με το iPad
    - Ενεργοποιήσουμε τον ήχο
    - Απενεργοποιήσουμε τον ήχο
    - Ενεργοποιήσουμε το πάνω κουμπί στην μέση της οθόνης
    - Απενεργοποιήσουμε το πάνω κουμπί στην μέση της οθόνης
    - Όταν εισάγουμε λάθος διεύθυνση IP ή λάθος port

Εικόνα 7.11: Μήνυμα ενημέρωσης ότι η κίνηση με το iPad είναι ενεργοποιημένη

3. Όταν εισάγουμε λάθος διεύθυνση IP ή λάθος port εμφανίζεται ένα μήνυμα (Εικόνα 7.12) έτσι ώστε ο χρήστης να γνωρίζει ότι έχει κάνει **λάθος εισαγωγή**. Επίσης δίνει στον χρήστη προτεινόμενες τιμές που μπορεί να εισάγει έτσι ώστε να γνωρίζει τον λόγο που δεν έγινε αποδεκτή η εισαγωγή του.

Εικόνα 7.12: Μήνυμα ενημέρωσης σε περίπτωση λάθος εισόδου

4. Η εφαρμογή είναι σχετικά απλή και μινιμαλιστική. Όλες οι διαθέσιμες ενέργειες βρίσκονται στην αρχική διεπαφή έτσί ώστε η εφαρμογή να είναι φιλική προς τον χρήστη. Επίσης η εφαρμογή αποτελείται μόνο από 2 διεπαφές έτσι ώστε ο χρήστης να ενεργεί πιο **εύκολα και γρήγορα.** Επίσης η κίνηση με το iPad και η κίνηση με την χρήση του τηλεχειριστηρίου βρίσκονται στην ίδια διεπαφή.

5. Σε περίπτωση που ο χρήστης πατήσει το κουμπί των ρυθμίσεων κατά λάθος τότε υπάρχει **εύκολη διαφυγή** μέσω του κουμπιού «πίσω».

6. Στα κουμπιά χρησιμοποιούνται εικόνες που δίνουν εύκολα την δυνατότητα στον χρήστη να καταλάβει τον σκοπό τους. Τα **χρώματα** που χρησιμοποιούνται δίνουν συγκεκριμένα μηνύματα, αφού με τέτοιο τρόπο είναι χτισμένο το γνωστικό μοντέλο των περισσότερων χρηστών. Τα χρώματα που χρησιμοποιούνται έχουν τις εξής έννοιες. Η χρήση του κόκκινου δίνει το νόημα του απενεργοποιημένου, το γαλάζιο και το πράσινο του ενεργοποιημένου. Για παράδειγμα η πράσινη εικόνα όταν ο ήχος είναι ενεργοποιημένος (Εικόνα 7.13) και η κόκκινη εικόνα όταν η κίνηση με το iPad είναι απενεργοποιημένη.

Εικόνα 7.13: Κουμπί χρώματος πρασίνου όταν ο ήχος είναι ενεργοποιημένος

7. Όταν εμφανίζονται τα μηνύματα στην οθόνη, φεύγουν από την οθόνη μετά από μικρό χρονικό διάστημα (2 δευτερόλεπτα), και έτσι ο χρήστης **δεν** αναγκάζεται να πατήσει οποιοδήποτε κουμπί ή να **περιμένει αρκετό χρόνο**.

Βιβλιογραφία

[1] . 2013. [ONLINE] Available at:http://athanasis.karoulis.gr/Data/Lessons%20HCI/HE.pdf. [Accessed 17 May 2013].

[2] Apple (2000). Inside Cocoa. Object Oriented Programming and the Objective-C Language. USA – CA : Apple Computer, Inc.

[3] Apple Developer 2013. [ONLINE] Available at: http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf [Accessed 17 May 2013]

[4] Arduino - ArduinoWiFiShield . 2013. Arduino - ArduinoWiFiShield. [ONLINE] Available at: http://arduino.cc/en/Main/ArduinoWiFiShield. [Accessed 17 May 2013]

[5] Arduino - HomePage. 2013.Arduino - HomePage. [ONLINE] Available at: http://www.arduino.cc/. [Accessed 17 May 2013]

[6] Arduino - Βικιπαίδεια. 2013. Arduino - Βικιπαίδεια. [ONLINE] Available at:http://el.wikipedia.org/wiki/Arduino. [Accessed 17 May 2013]

[7] Arduino Wi-Fi shield available, costs $85 USD. 2013.Arduino WiFi shield available, costs $85 USD. [ONLINE] Available at: http://hackaday.com/2012/08/21/arduino-wifi-shield-available-costs-85-usd/. [Accessed 17 May 2013]

[8] Box2D - Wikipedia, the free encyclopedia. 2013. Box2D - Wikipedia, the free encyclopedia. [ONLINE] Available at:http://en.wikipedia.org/wiki/Box2D. [Accessed 17 May 2013]

[9] Cocos2d - Wikipedia, the free encyclopedia. 2013. Cocos2d - Wikipedia, the free encyclopedia. [ONLINE] Available at:http://en.wikipedia.org/wiki/Cocos2d. [Accessed 17 May 2013]

[10] Cocos2d for iPhone. 2013. Cocos2d for iPhone. [ONLINE] Available at: http://www.cocos2d-iphone.org/forum/. [Accessed 17 May 2013]

[11] Developer Tools Overview - Apple Developer. 2013. Developer Tools Overview - Apple Developer. [ONLINE] Available at:https://developer.apple.com/technologies/tools. [Accessed 17 May 2013]

[12] Grady Booch, James Rumbaugh & Ivar Jacobson (2005) The Unified Modeling Language User Guide SECOND EDITION. New Jersey: Addison Wesley Professional

[13] HCI 2013. [ONLINE] Available at: http://www.syros.aegean.gr/users/kgp/BookSlides/1-2.HCI_Foundations_Koutsabasis.pdf. [Accessed 17 May 2013]

[14] James Bucanek (2006) Beginning Xcode. Indianapolis: Wiley Publishing, Inc.

[15] Jeff Hawkins (2011) Cocoa and Objective-C Cookbook. Birmingham Packt Publishing Ltd.

[16] MacRumors Forums - Mac News and Rumor Discussion. 2013. MacRumors Forums - Mac News and Rumor Discussion. [ONLINE] Available at:http://forums.macrumors.com/. [Accessed 17 May 2013]

[17] Objective-C - Wikipedia, the free encyclopedia. 2013.Objective-C - Wikipedia, the free encyclopedia. [ONLINE] Available at:https://en.wikipedia.org/wiki/Objective-C.[Accessed 17 May 2013]

[18] RCKit: RCKit library for Arduino. 2013. RCKit: RCKit library for Arduino. [ONLINE] Available at: http://www.airspayce.com/mikem/arduino/RCKit. [Accessed 17 May 2013]

[19] RCTx for iPhone, iPod touch, and iPad on the iTunes App Store. 2013. RCTx for iPhone, iPod touch, and iPad on the iTunes App Store. [ONLINE] Available at: https://itunes.apple.com/us/app/rctx/id567423127?mt=8. [Accessed 17 May 2013]

[20] Rod Strougo & Ray Wenderlich (2012) A Hands-On guide to building iOS Games with Cocos2D, Box2D, and Chipmunk. Boston: Pearson Education, Inc.

[21] Scott Stevenson (2010) Cocoa and Objective – C: Up and Running. United States of America: O' Reilly Media Inc.

[22] Stack Overflow. 2013. Stack Overflow. [ONLINE] Available at: http://stackoverflow.com/. [Accessed 17 May 2013]

[23] Stephen G. Kochan (2011) Programming in Objective C Third Edition. USA: Pearson Education, Inc.

[24] Waterfall model - Wikipedia, the free encyclopedia. 2013.Waterfall model - Wikipedia, the free encyclopedia. [ONLINE] Available at: http://en.wikipedia.org/wiki/Waterfall_model. [Accessed 17 May 2013]

[25] Waterfall Model in Software Engineering. 2013.Waterfall Model in Software Engineering. [ONLINE] Available at: http://www.buzzle.com/articles/waterfall-model-in-software-engineering.html. [Accessed 17 May 2013]

[26] Web Usability - Ευρετική Αξιολόγηση. 2013. Web Usability - Ευρετική Αξιολόγηση. [ONLINE] Available at: http://web-omada10.wikispaces.com/%CE%95%CF%85%CF%81%CE%B5%CF%84%CE%B9%CE%BA%CE%AE+%CE%91%CE%BE%CE%B9%CE%BF%CE%BB%CF%8C%CE%B3%CE%B7%CF%83%CE%B7. [Accessed 17 May 2013]

[27] Wireshark · Go deep 2013. Wireshark · Go deep. [ONLINE] Available at:http://www.wireshark.org/. [Accessed 17 May 2013]

[28] Xcode 4 Downloads and Resources - Apple Developer. 2013. Xcode 4 Downloads and Resources - Apple Developer. [ONLINE] Available at: https://developer.apple.com/xcode/. [Accessed 17 May 2013]

[29] Ευχρηστία - Βικιπαίδεια. 2013.Ευχρηστία - Βικιπαίδεια. [ONLINE] Available at:http://el.wikipedia.org/wiki/%CE%95%CF%85%CF%87%CF%81%CE%B7%CF%83%CF%84%CE%AF%CE%B1. [Accessed 17 May 2013]

[30] Κεφ.1 Μοντέλα ανάπτυξης λογισμικού - Sya. 2013. Κεφ.1 Μοντέλα ανάπτυξης λογισμικού - Sya. [ONLINE] Available at:http://sya.a.wiki-site.com/index.php?title=%CE%9A%CE%B5%CF%86.1_%CE%9C%CE%BF%CE%BD%CF%84%CE%AD%CE%BB%CE%B1_%CE%B1%CE%BD%CE%AC%CF%80%CF%84%CF%85%CE%BE%CE%B7%CF%82_%CE%BB%CE%B

F%CE%B3%CE%B9%CF%83%CE%BC%CE%B9%CE%BA%CE%BF%CF%8D&printable=yes. [Accessed 17 May 2013]

Παράρτημα Α

Στο παρόν παράρτημα θα παρουσιάσουμε τα έγγραφα που δημιουργήθηκαν κατά την ανάπτυξη της εφαρμογής PM JOYSTIC στα πλαίσια της Τεχνολογίας λογισμικού. Στο πρώτο μέρος θα παρουσιάσουμε το έγγραφο **"Απαιτήσεις συστήματος"**, στο δεύτερο μέρος το έγγραφο **"Προδιαγραφές σχεδιασμού"** και στο τρίτο μέρος το έγγραφο **"Πλάνο ελέγχου"**.

Α1 Απαιτήσεις συστήματος (System Requirements)

PM JOYSTIC

IEEE Standard 830 Software Requirements

1. Εισαγωγή

Αυτή η ενότητα περιλαμβάνει την σύνοψη ολόκληρου του εγγράφου.

1.1. Σκοπός: Σκοπός αυτού του εγγράφου είναι να καθορίσει τις απαιτήσεις της εφαρμογής PM JOYSTIC. Ο κ. Βάσος Βασιλείου θα πρέπει να εξετάσει και να παρουσιάσει τις απαιτήσεις και τις νέες απαιτήσεις (αν υπάρξουν καινούργιες) κατά την διάρκεια ανάπτυξης της εφαρμογής. Αν δοθούν νέες απαιτήσεις η εφαρμογή θα πρέπει να αναπτυχθεί συμπεριλαμβανομένου και των νέων απαιτήσεων.

1.2. Στόχος συστήματος: Στόχος, είναι η δημιουργία εφαρμογής τηλεχειριστηρίου παρόμοιας με αυτήν του RCTx του Apple Store. Στόχος του τηλεχειριστηρίου αυτού είναι μέσω του πρωτοκόλλου RCOIP να επικοινωνεί με συγκεκριμένες συχνότητες με το υλικό (hardware) του ελικοπτέρου. Το υλικό του ελικοπτέρου αποτελείται από πλακέτα Arduino με ασπίδα Wi-fi (Wi-Fi - shield) και θα επικοινωνεί με την εφαρμογή που θα δημιουργήσουμε μέσω ασύρματου δικτύου (WI - FI). Με αυτή την επικοινωνία το τηλεχειριστήριο θα μπορεί να δώσει διάφορες παραμέτρους οι οποίες θα καθορίζουν την επιτάχυνση, την ταχύτητα, και την θέση του ελικοπτέρου στον χώρο τριών διαστάσεων. Επιπρόσθετα η εφαρμογή θα διαθέτει δύο τηλεχειριστήρια και 1 διακόπτη για

επικοινωνία με το υλικό του ελικοπτέρου. Το αριστερό τηλεχειριστήριο το οποίο θα ορίζει τιμές στα κανάλια 0 και 1 και το δεξιό τηλεχειριστήριο το οποίο θα ορίζει τιμές στα κανάλια 2 και 3. Ο διακόπτης θα ορίζει το κανάλι από το 4. Επίσης η εφαρμογή θα μπορεί να δώσει κίνηση στο ελικόπτερο μέσω της κίνησης του iPad. Ο συγχρονισμός των τιμών της εφαρμογής θα γίνεται κάθε 1 χιλιοστό του δευτερολέπτου.

1.3. Συντομογραφίες και ακρώνυμα

RCKIT πλαίσιο: σετ από αντικείμενα λογισμικού για την δημιουργία RCOIP δέκτη σε μικροελεγκτή Arduino.

RCTx: Εφαρμογή που βρίσκεται στο Apple Store για απομακρυσμένο έλεγχο συσκευής (μέσω Wi-fi) μέσω του πρωτοκόλλου RCOIP. Η συσκευή αυτή είναι απαραίτητο να χρησιμοποιεί το RCKit.

RCOIP (Remote Control over IP): πρωτόκολλο που χρησιμοποιείται για την μεταφορά απομακρυσμένων εντολών ελέγχου από ένα πομπό σε ένα δέκτη μέσω του IP μεταφοράς, όπως για παράδειγμα Ethernet ή Wi-fi.

ARDUINO: μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή και εισόδους / εξόδους.

WIFI SHIELD: επιτρέπει σε μια πλακέτα Arduino να συνδέεται στο διαδίκτυο, χρησιμοποιώντας την βιβλιοθήκη Wi-fi Library.

GPS (Παγκόσμιο Σύστημα Θεσιθεσίας): είναι ένα παγκόσμιο σύστημα εντοπισμού θέσης, το οποίο βασίζεται σε ένα "πλέγμα" εικοσιτεσσάρων δορυφόρων της Γης.

COCOA: Διεπαφή προγραμματισμού εφαρμογών (Application programming interface- API) για Mac OS X λειτουργικό σύστημα.

IOS: λειτουργικό σύστημα που δημιουργήθηκε από την Apple, και τρέχει πάνω σε Apple συσκευές.

1.4. Γενική επισκόπηση

Το υπόλοιπο του εγγράφου έχει την ακόλουθη διάταξη:

Ενότητα 2: Προσφέρει την γενική περιγραφή της εφαρμογής και τις απαιτήσεις της.

Ενότητα 3: Προσφέρει τις συγκεκριμένες απαιτήσεις της εφαρμογής, συμπεριλαμβανομένου των εσωτερικών και των εξωτερικών διεπαφών του χρήστη, λειτουργικές απαιτήσεις, απαιτήσεις υλικού, λογισμικού και τα χαρακτηριστικά συστήματος.

2. Γενική περιγραφή

Αυτή η ενότητα περιλαμβάνει την περιγραφή των απαιτήσεων της εφαρμογής PM JOYSTIC. Παρέχει την προοπτική για κατανόηση των συγκεκριμένων απαιτήσεων της Ενότητας 3.

a. **Προοπτική προϊόντος:** Το λογισμικό αποτελεί ανεξάρτητο προϊόν / εφαρμογή η οποία θα ελέγχει ένα ελικόπτερο με υλικό (hardware) Arduino.

b. **Λειτουργίες προϊόντος:** Το λογισμικό PM JOYSTIC θα μπορεί να δώσει στο ελικόπτερο επιτάχυνση περιστροφή και συντεταγμένες σε κίνηση στο χώρο σε τρεις διαστάσεις.

c. **Χαρακτηριστικά χρήστη:** Ο οποιοσδήποτε μπορεί να χρησιμοποιήσει την εφαρμογή.

d. **Περιορισμοί:** Θα πρέπει ο χρήστης να διαθέτει iPad. Οι χρήστες μπορεί να είναι ο οποιοσδήποτε ακόμη και μικρά παιδιά. Η εφαρμογή μαζί με το ελικόπτερο θα πρέπει να χρησιμοποιούνται σε ανοικτό χώρο χωρίς εμπόδια. Περισσότερη ανάλυση ακολουθεί στην Ενότητα 3.

e. **Εικασίες και άλλες εξαρτήσεις:** Η εφαρμογή τρέχει μόνο σε συσκευές iPad, σε υλικό ARDUINO το οποίο διαθέτει Wi-Fi shield και το πλαίσιο εφαρμογών RCKIT.

f. **Απαιτήσεις υποσυνόλων:** Νέες κυκλοφορίες του RCTx θα περιλαμβάνουν διεπαφή ρυθμίσεων του ασύρματου πομπού, κανάλια για καθορισμό θέσης με GPS, απομακρυσμένη χαρτογράφηση οχήματος, απομακρυσμένες κάμερες οχήματος, μακρό-βιντεογράφηση και άλλα.

3. Συγκεκριμένες απαιτήσεις

Πιο κάτω περιγράφονται οι συγκεκριμένες απαιτήσεις της εφαρμογής PM JOYSTIC.

a. Απαιτήσεις εξωτερικών διεπαφών

i. **Διεπαφές χρήστη:** Η εφαρμογή θα χρησιμοποιήσει τα πακέτα εφαρμογών COCOA και θα αναπτυχθεί σε λειτουργικό σύστημα MACOS X. Η εφαρμογή θα τρέχει μόνο σε συσκευές iPad.

ii. **Διεπαφές υλικού :** Μόνο συσκευές iPad μπορούν να τρέξουν την εφαρμογή. Επίσης το ελικόπτερο θα διαθέτει υλικό (hardware) ARDUINO. Μελλοντικά η εφαρμογή με τις κατάλληλες τροποποιήσεις θα μπορεί να τρέξει και σε λειτουργικά MACOS X.

iii. **Απαιτήσεις σε λογισμικό:** Λογισμικό για ανάπτυξη της εφαρμογής: MACBOOK PRO με ελάχιστη ταχύτητα δύο επεξεργαστών στα 2,4GHz με ελάχιστη μνήμη 2GB , με σκληρό δίσκο χωρητικότητας 150 GB, και κάρτα γραφικών 256M. Λογισμικό για έλεγχο της εφαρμογής, IOS έκδοση 6.1.

iv. **Διεπαφές επικοινωνίας:** Τουλάχιστο 10Mb/s διαθέσιμης κάρτας δικτύου.

b. Λειτουργικές απαιτήσεις

Τα ακόλουθα περιγράφουν τις λειτουργικές απαιτήσεις οι οποίες αναγνωρίστηκαν για την ανάπτυξη του λογισμικού.

i. Λειτουργική απαίτηση 1

1. **Εισαγωγή:** Η εφαρμογή θα μπορεί να προσδιορίζει που θα κινηθεί το ελικόπτερο, που αποτελεί το υλικό μέσω κίνησης του iPad από τον χρήστη.
2. **Μονάδες εισόδου:** iPad
3. **Επεξεργασία:** Με την κίνηση του iPad θα στέλνονται τιμές για τα κανάλια 2 και 3 που αφορούν το δεξιό τηλεχειριστήριο
4. **Αποτέλεσμα:** Η σωστή τοποθέτηση του ελικοπτέρου στο χώρο

ii. Λειτουργική απαίτηση 2

1. **Εισαγωγή:** Η εφαρμογή θα μπορεί να δώσει επιτάχυνση και περιστροφή στο ελικόπτερο μέσω του αριστερού τηλεχειριστηρίου από τον χρήστη.
2. **Μονάδες εισόδου:** Αριστερό τηλεχειριστήριο.
3. **Επεξεργασία:** Με την κίνηση του αριστερού τηλεχειριστηρίου θα στέλνονται τιμές στα κανάλια 0 και 1 .
4. Αποτέλεσμα: Η επιτάχυνση και η περιστροφή του ελικοπτέρου.

iii. Λειτουργική απαίτηση 3

1. **Εισαγωγή:** Η εφαρμογή θα μπορεί να προσδιορίζει που θα κινηθεί το ελικόπτερο, που αποτελεί το υλικό, μέσω του δεξιού τηλεχειριστηρίου από τον χρήστη.
2. **Μονάδες εισόδου:** Δεξιό τηλεχειριστήριο.
3. **Επεξεργασία:** Με την κίνηση του δεξιού τηλεχειριστηρίου θα στέλνονται τιμές στα κανάλια 2 και 3.
4. Αποτέλεσμα**:** Η σωστή τοποθέτηση του ελικοπτέρου στο χώρο.

iv. Λειτουργική απαίτηση 4

1. **Εισαγωγή:** Η εφαρμογή θα μπορεί να επικοινωνεί με τον διακόπτη (Δεν δόθηκαν λεπτομέρειες)
2. **Μονάδες εισόδου:** Ένα κουμπί στη μέση της οθόνης.
3. **Επεξεργασία:** Με το πάτημα του κουμπιού θα στέλνονται τιμές στο κανάλι 4.
4. **Αποτέλεσμα:** Δεν δόθηκαν λεπτομέρειες

c. Απαιτήσεις απόδοσης: Υπάρχει ανάγκη για χρήση του λογισμικού από ένα χρήστη με χρόνο ανταπόκρισης 1 χιλιοστό του δευτερολέπτου.

d. Περιορισμοί σχεδίασης της εφαρμογής Δεν υπάρχει καμιά ειδική ανάγκη σε θέματα ασφάλειας προς το παρόν για την σχεδίαση της εφαρμογής.

i. Διαδικασία έγκρισης: Δεν υπάρχει.

ii. Περιορισμοί στο υλικό: Πληκτρολόγιο , Ποντίκι, , MACBOOK RPO με δύο επεξεργαστές , 2 GB RAM , 150 GB δίσκο και κάρτα γραφικών 256MB.

e. Χαρακτηριστικά λογισμικού: Σε αυτή την ενότητα δίνεται ιδιαίτερη σημασία για την ποιότητα του λογισμικού.

i. **Παράγοντες διαθεσιμότητας:** Η εφαρμογή θα είναι διαθέσιμη αφού εγκατασταθεί στο iPad που θα χρησιμοποιηθεί ως τηλεχειριστήριο.

ii. **Απαιτήσεις ασφάλειας:** Το λογισμικό δεν έχει ιδιαίτερες απαιτήσεις ασφάλειας προς το παρόν. Όταν ζητηθεί μελλοντικά θα μπορούσε η εφαρμογή να χρειάζεται είσοδο (password protected area) για λειτουργία και χρησιμοποίηση του τηλεχειριστηρίου.

iii. **Απαιτήσεις συντήρησης:** Η εφαρμογή θα αναπτυχθεί σε γλώσσα προγραμματισμού Objective C. Θα αναφέρονται σχόλια κατά την ανάπτυξη του κώδικα για εύκολη κατανόηση και επαναχρησιμοποίηση του κώδικα. Επιπρόσθετα θα δοθούν και σχεδιαγράμματα ροής δεδομένων για εύκολη κατανόηση του κώδικά καθώς και εγχειρίδιο εκτέλεσης της εφαρμογής (αρχείο READ ME – Παράρτημα Γ).

f. **Άλλες απαιτήσεις:** Δεν υπάρχουν άλλες απαιτήσεις.

A2. Προδιαγραφές σχεδιασμού (Design Specifications)

PM JOYSTIC

IEEE 1016 Design Specifications

1. Εισαγωγή

Αυτή η ενότητα περιλαμβάνει την σύνοψη ολόκληρου του εγγράφου.

1.1. Σκοπός κειμένου: Το κείμενο αυτό παρουσιάζει τις προδιαγραφές του σχεδιασμού της εφαρμογής PM JOYSTIC.

1.2. Στόχος του συστήματος: Στόχος της εφαρμογής, είναι η δημιουργία εφαρμογής τηλεχειριστηρίου παρόμοιας με αυτήν του RCTx του Apple Store. Στόχος του τηλεχειριστηρίου αυτού είναι μέσω του πρωτοκόλλου RCOIP να επικοινωνεί με συγκεκριμένες συχνότητες με το υλικό (hardware) του ελικοπτέρου. Το υλικό του ελικοπτέρου αποτελείται από πλακέτα Arduino με ασπίδα Wi-fi (Wi-fi shield). Το υλικό θα επικοινωνεί με την εφαρμογή που θα δημιουργήσουμε μέσω ασύρματου δικτύου (Wi - fi). Με αυτή την επικοινωνία το τηλεχειριστήριο θα μπορεί να δώσει διάφορές παραμέτρους οι οποίες θα καθορίζουν την επιτάχυνση, την ταχύτητα, και την θέση του ελικοπτέρου στον χώρο τριών διαστάσεων. Επιπρόσθετα η εφαρμογή θα διαθέτει δύο τηλεχειριστήρια και 1 διακόπτη για επικοινωνία με το υλικό του ελικοπτέρου. Το αριστερό τηλεχειριστήριο το οποίο θα ορίζει τιμές στα κανάλια 0 και 1 και το δεξιό τηλεχειριστήριο το οποίο θα ορίζει τιμές στα κανάλια 2 και 3. Ο διακόπτης θα ορίζει το κανάλι από το 4. Επίσης η εφαρμογή θα μπορεί να δώσει κίνηση στο ελικόπτερο μέσω της κίνησης του iPad. Ο συγχρονισμός των τιμών θα γίνεται κάθε 1 χιλιοστό του δευτερολέπτου.

2. Γενική επισκόπηση

Το υπόλοιπο του κειμένου αυτού έχει την ακόλουθη διάταξη:

Ενότητα 3: Παρέχει την αρχιτεκτονική του λογισμικού, και συμπεριλαμβάνει τη γενική επισκόπηση των αυτοτελών λογισμικών μονάδων (modules) και των ρουτινών, που συνδυαζόμενων με άλλα στοιχεία, συνθέτουν το συνολικό πρόγραμμα (components). Επιπρόσθετα παρουσιάζει την δομή και τις σχέσεις των αυτοτελών λογισμικών μονάδων μεταξύ τους καθώς και θέματα που σχετίζονται με το σχεδιασμό της διεπαφής για το χρήστη.

Ενότητα 4: Παρέχει τη λεπτομερή περιγραφή των ρουτινών που, συνδυαζόμενων με άλλα στοιχεία, συνθέτουν το συνολικό πρόγραμμα (components). Επιπρόσθετα παρουσιάζει τη λεπτομερή ανάλυση των ρουτινών για, πρώτο εντολές επιτάχυνσης στο ελικόπτερο χρησιμοποιώντας την κίνηση του iPad και την χρήση του αριστερού τηλεχειριστηρίου, δεύτερο τον ορισμό συντεταγμένων στο ελικόπτερο μέσω του δεξιού τηλεχειριστηρίου και τέλος την επικοινωνία της εφαρμογής με το ελικόπτερο μέσω του διακόπτη.

Ενότητα 5: Περιγράφει την επαναχρησιμοποίηση κώδικα και την σχέση τους με άλλα προϊόντα.

Ενότητα 6: Παρέχει τις αποφάσεις για την σχεδίαση της εφαρμογής καθώς και άλλες εμπορικές συναλλαγές.

Παραρτήματα: Παρέχει λεπτομέρειες για τα συστήματα και υποσυστήματα της εφαρμογής (Systems - Subsystems), τη δομή της εφαρμογής (File structure), διαγράμματα ακολουθίας (Sequence Diagrams), διαγράμματα καταστάσεων,(State Diagrams) και τέλος τα διαγράμματα των κλάσεων. (Class Diagrams).

3. Περιγραφή αρχιτεκτονικής εφαρμογής

Γενική επισκόπηση λογισμικών μονάδων και ρουτινών λογισμικού (modules / components)

Αυτή η υποενότητα θα παρουσιάσει τις διαφορετικές ρουτίνες λογισμικών και αυτοτελής μονάδες λογισμικών

Η εφαρμογή θα αποτελείται από τα ακόλουθες αυτοτελείς λογισμικές μονάδες και ρουτίνες λογισμικών.

Περιγραφή ρουτινών λογισμικού (Components)	Υποστηριζόμενες μονάδες λογισμικού (Modules)
Κίνηση ελικόπτερου χρησιμοποιώντας την κίνηση του iPad.	Κίνηση Κατεύθυνση
Επιτάχυνση και περιστροφή ελικόπτερου χρησιμοποιώντας το αριστερό τηλεχειριστήριο	Επιτάχυνση Περιστροφή
Κίνηση ελικόπτερου χρησιμοποιώντας το δεξιό τηλεχειριστήριο	Κίνηση Κατεύθυνση
Επικοινωνία της εφαρμογής με το ελικόπτερο μέσω του διακόπτη	----------

Δομή και σχέση ρουτινών λογισμικού εφαρμογής

Η εφαρμογή PM JOYSTIC θα περιλαμβάνει της ρουτίνες λογισμικού κίνησης με το iPad και κίνησης με το δεξιό τηλεχειριστήριο που αφορούν τα κανάλια 2 και 3. Κατά την εκκίνηση της εφαρμογής θα γίνεται χρήση του δεξιού τηλεχειριστηρίου. Αν ο χρήστης της θέλει να πετύχει κίνηση χρησιμοποιώντας την κίνηση της συσκευής θα υπάρχει το κατάλληλο μέσο που θα ενεργοποιεί την επιλογή αυτή (κουμπί) ή θα την απενεργοποιεί. Δηλαδή δεν θα μπορεί να γίνεται χρήση και των δύο υπορουτίνων ταυτόχρονα. Η ρουτίνα λογισμικού κίνησης με το αριστερό τηλεχειριστήριο θα είναι

αυτόνομη από της προηγούμενες ρουτίνες που αναφέραμε και θα λειτουργεί ταυτόχρονα μαζί τους.

Θέματα της διεπαφής χρήστη

Η διεπαφή του λογισμικού PM JOYSTIC με το χρήστη θα γίνει όσο πιο εύχρηστη και φιλική προς το χρήστη. Για αυτό το λόγο δεν θα υλοποιηθούν μηχανισμοί ασφάλειας και οι χρήστες θα έχουν την δυνατότητα να χρησιμοποιήσουν όλες τις διαθέσιμες επιλογές στην πρώτη διεπαφή (Interface) που θα εμφανίζεται.

4. Αναλυτική περιγραφή ρουτινών (components)

Περιγραφή πρότυπου λογισμικού ρουτινών

Παράδειγμα	
Αναγνώριση	Όνομα που περιγράφει την ρουτίνα στο σύστημα
Τύπος	Ενότητα , υποπρόγραμμα, αρχείο δεδομένων, διαδικασία ελέγχου
Σκοπός	Με το σχεδιασμό της ρουτίνας γίνεται υλοποίηση της λειτουργίας και των απαιτήσεων απόδοσης
Λειτουργίες	Τι κάνει κάθε ρουτίνα, τι χρειάζεται ως δεδομένα εισόδου, πώς τα χρησιμοποιεί, που αποθηκεύονται τα δεδομένα και τι αλλαγές εμφανίζονται.
Δευτερεύον ζητήματα	
Εξαρτήσεις (Dependencies)	Ποια είναι η σχέση μεταξύ των λειτουργιών της υπορουτίνας και των υπόλοιπων υπορουτινών. Πως οι άλλες υπορουτίνες χρησιμοποιούν αυτή την ρουτίνα.
Διεπαφές	Οι διεπαφές θα δώσουν λεπτομερή περιγραφή των κινήσεων ενός χρήστη . Θα υπάρχουν ερωτήσεις επιβεβαίωσης; ή μηνύματα επιτυχών ή ανεπιτυχών ενεργειών για λειτουργία του συστήματος.
Πόροι	Μια ολοκληρωμένη περιγραφή όλων των πόρων που θα χρησιμοποιηθούν (υλικό / λογισμικό)
Επεξεργασία	Ο κώδικας της λειτουργίας
Δεδομένα	Η ρουτίνα, περιγράφει την μέθοδο , αρχικές τιμές , διαφορές και μορφή των δεδομένων.

Επιτάχυνση και περιστροφή του ελικόπτερου χρησιμοποιώντας το αριστερό τηλεχειριστήριο

Επιτάχυνση και περιστροφή του ελικόπτερου χρησιμοποιώντας το αριστερό τηλεχειριστήριο	
Αναγνώριση	Η εφαρμογή θα αναγνωρίζει το άγγιγμα από τον χρήστη στο αριστερό τηλεχειριστήριο.
Τύπος	Ενότητα υποσυστήματος τηλεχειριστηρίου. Αρχείο τύπου Objective C με δυο κλάσεις και δεκαοκτώ μεθόδους.
Σκοπός	Η εφαρμογή θα αναγνωρίζει το άγγιγμα από τον χρήστη στο αριστερό τηλεχειριστήριο και αναλόγως θα δίνει επιτάχυνση και περιστροφή στο ελικόπτερο.
Λειτουργίες	Επιτάχυνση Περιστροφή
Δευτερεύον ζητήματα	
Εξαρτήσεις (Dependencies)	Η ρουτίνα αυτή είναι ανεξάρτητη από τις υπόλοιπες υπορουτίνες και θα μπορεί να χρησιμοποιηθεί ταυτόχρονα μαζί τους. Η ρουτίνα εξαρτάται από την ρουτίνα που επεξεργάζεται την επικοινωνία.
Διεπαφές	Διεπαφή κύριας οθόνης με το αριστερό τηλεχειριστήριο.
Πόροι	XCode Mac iPad Arduino

Επιτάχυνση και περιστροφή του ελικόπτερου χρησιμοποιώντας το αριστερό τηλεχειριστήριο	
Επεξεργασία	Ενεργοποίηση του τηλεχειριστηρίου με άγγιγμα. Αναγνώριση αγγίγματος τηλεχειριστηρίου μέσα στα πλαίσια του κύκλου και αν είναι έξω από τα όρια υπολογίζει το σημείο πάνω στο κύκλο. Ανάλογα με τη θέση που άγγιξε ο χρήστης υπολογίζει τις τιμές που πρέπει να στείλει στα κανάλια 0 και 1. Μετά η ρουτίνα της κύριας οθόνης ενημερώνει τις τιμές στην ρουτίνα επικοινωνίας.
Δεδομένα	Η ρουτίνα ορίζει σε μεταβλητές τα κανάλια επικοινωνίας και τις τιμές που στέλνονται.

Κίνηση του ελικόπτερου χρησιμοποιώντας το δεξιό τηλεχειριστήριο

Κίνηση του ελικόπτερου χρησιμοποιώντας το δεξιό τηλεχειριστήριο	
Αναγνώριση	Η εφαρμογή θα αναγνωρίζει το άγγιγμα από τον χρήστη στο δεξιό τηλεχειριστήριο.
Τύπος	Ενότητα υποσυστήματος τηλεχειριστηρίου. Αρχείο τύπου Objective C με δυο κλάσεις και δεκαοκτώ μεθόδους.
Σκοπός	Η εφαρμογή θα αναγνωρίζει το άγγιγμα από τον χρήστη στο αριστερό τηλεχειριστήριο και αναλόγως θα δίνει επιτάχυνση και περιστροφή στο ελικόπτερο.
Λειτουργίες	Κίνηση Κατεύθυνση
Δευτερεύον ζητήματα	
Εξαρτήσεις (Dependencies)	Η ρουτίνα αυτή χρησιμοποιείται όταν δεν γίνεται χρήση της ρουτίνας του iPad. Η ρουτίνα εξαρτάται από την ρουτίνα που επεξεργάζεται την επικοινωνία.
Διεπαφές	Διεπαφή κύριας οθόνης με το δεξιό τηλεχειριστήριο.
Πόροι	XCode Mac iPad Arduino

Κίνηση του ελικόπτερου χρησιμοποιώντας το δεξιό τηλεχειριστήριο	
Επεξεργασία	Ενεργοποίηση του τηλεχειριστηρίου με άγγιγμα. Αναγνώριση αγγίγματος τηλεχειριστηρίου μέσα στα πλαίσια του κύκλου και αν είναι έξω από τα όρια υπολογίζει το σημείο πάνω στο κύκλο. Ανάλογα με τη θέση που άγγιξε ο χρήστης υπολογίζει τις τιμές που πρέπει να στείλει στα κανάλια 2 και 3. Μετά η ρουτίνα της κύριας οθόνης ενημερώνει τις τιμές στην ρουτίνα επικοινωνίας.
Δεδομένα	Η ρουτίνα ορίζει σε μεταβλητές τα κανάλια επικοινωνίας και τις τιμές που στέλνονται.

Κίνηση ελικόπτερου χρησιμοποιώντας την κίνηση του iPad

Κίνηση του ελικόπτερου χρησιμοποιώντας την κίνηση του iPad.	
Αναγνώριση	Η εφαρμογή θα αναγνωρίζει την κίνηση του iPad.
Τύπος	Ενότητα υποσυστήματος iPad. Αρχείο τύπου Objective C με μια κλάση και έξι μεθόδους.
Σκοπός	Η εφαρμογή θα αναγνωρίζει την κίνηση του iPad και αναλόγως θα δίνει κίνηση στο ελικόπτερο και κατεύθυνση.
Λειτουργίες	Κίνηση Κατεύθυνση
Δευτερεύον ζητήματα	Η μεταφορά των παραμέτρων στις κλάσεις για κίνηση του δεξιού τηλεχειριστηρίου
Εξαρτήσεις (Dependencies)	Για να ενεργοποιηθεί η χρήση της ρουτίνας απαραίτητα πρέπει να ενεργοποιηθεί η ρουτίνα για ενεργοποίηση του iPad Επίσης η ρουτίνα αυτή εξαρτάται από τη ρουτίνα κίνησης του ελικοπτέρου με το δεξιό τηλεχειριστήριο.
Διεπαφές	Διεπαφή κύριας οθόνης με το κουμπί για το iPad. Μήνυμα ενεργοποίησης / απενεργοποίησης του iPad
Πόροι	XCode Mac iPad Arduino

Κίνηση του ελικόπτερου χρησιμοποιώντας την κίνηση του iPad.	
Επεξεργασία	Ενεργοποίηση της συσκευής και κίνηση του Arduino Με το iPad. Η ρουτίνα παίρνει την τοποθεσία του iPad των αξόνων x, και y . Μετά κάνει υπολογισμό της θέσης που πρέπει να είναι το δεξιό τηλεχειριστήριο και κινεί αναλόγως του υλικό του Arduino.
Δεδομένα	Η ρουτίνα αυτή πέρνει την θέση της συσκευής στον χώρο (άξονας x και y). Οι μεταβλητές αυτές χρησιμοποιούνται για υπολογισμό τη θέσης του δεξιού τηλεχειριστηρίου. Η ρουτίνα ορίζει σε μεταβλητές τα κανάλια επικοινωνίας και τις τιμές που στέλνονται.

Επικοινωνία της εφαρμογής με το ελικόπτερο μέσω του διακόπτη

Κίνηση του ελικόπτερου χρησιμοποιώντας το δεξιό τηλεχειριστήριο	
Αναγνώριση	Η εφαρμογή θα αλλάζει την συχνότητα επικοινωνίας της εφαρμογής με το ελικόπτερο.
Τύπος	Ενότητα υποσυστήματος διακόπτη. Αρχείο τύπου Objective C με μια κλάση και τρεις μεθόδους.
Σκοπός	Η εφαρμογή θα αναγνωρίζει την ενεργοποίηση του διακόπτη και αναλόγως θα αλλάζει τη συχνότητα.
Λειτουργίες	Αλλαγή συχνότητας
Δευτερεύον ζητήματα	----------
Εξαρτήσεις (Dependencies)	----------
Διεπαφές	Διεπαφή κύριας οθόνης με το κουμπί για το διακόπτη Μήνυμα ενεργοποίησης / απενεργοποίησης του διακόπτη
Πόροι	XCode Mac iPad Arduino
Επεξεργασία	Αν ενεργοποιηθεί ο διακόπτης στέλνει την τιμή 1 ενώ αν είναι απενεργοποιημένος στέλνει την τιμή 0.
Δεδομένα	Η τιμή στο κανάλι και η τιμή που θα σταλεί.

5. Επαναχρησιμοποίηση κώδικα και σχέση με άλλα προϊόντα

Για την δημιουργία της εφαρμογής θα χρησιμοποιηθεί κώδικας από τις ακόλουθες βιβλιοθήκες:

- RCKit
- AsyncUdpSocket
- Cocos2d
- Box2d
- Πλαίσια εργασίας (frameworks) της Apple

Επιπρόσθετα έχουν χρησιμοποιηθεί διάφορες ομάδες συζητήσεων και σχετικά παραδείγματα από βιβλία.

6. Αποφάσεις που διέπουν την σχεδίαση και εμπορικές συναλλαγές.

Τα ακόλουθα σημεία περιγράφουν τις αποφάσεις μας για την φάση της σχεδίασης καθώς και άλλες εμπορικές συναλλαγές.

- Αποφασίστηκε να σχεδιαστεί η εφαρμογή εξολοκλήρου από εμάς χωρίς την αγορά έτοιμης της διεπαφής χρήστη ή έτοιμης εφαρμογής.
- Θα υπάρχει μόνο μια διεπαφή χρήστη στην οποία θα παρέχονται όλες οι διαθέσιμες λειτουργίες για κίνηση του ελικοπτέρου από τον χρήστη.

Παραρτήματα

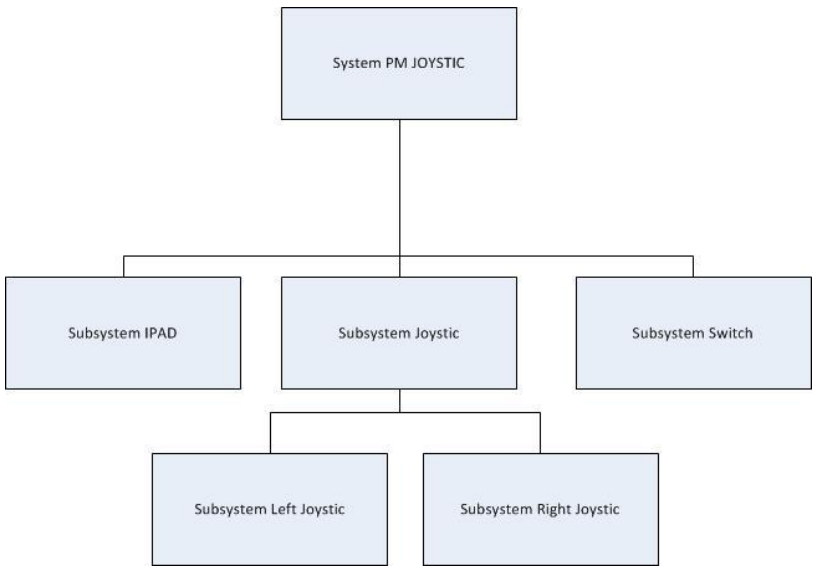
Παράρτημα Α - Συστήματα – Υποσυστήματα

Με την ανάλυση και την μελέτη των απαιτήσεων της εφαρμογής το σύστημα μας, Εικόνα Α.2.1 μπορεί να σπάσει σε 4 διαφορετικά υποσυστήματα τα οποία είναι

- Υποσύστημα IPAD (Εικόνα Α.2.2) : ανάλογα με την κίνηση του iPad δίνεται κίνηση στο ελικόπτερο
- Υποσύστημα τηλεχειριστήριο (Joystic) (Εικόνα Α.2.3): αποτελείται από το υποσύστημα αριστερό τηλεχειριστήριο (leftJoystic) και δεξιό τηλεχειριστήριο (right Joystick)
    - Υποσύστημα αριστερό τηλεχειριστήριο (leftJoystic): με την κίνηση του αριστερού τηλεχειριστηρίου καθορίζεται η επιτάχυνση και οι περιστροφή του ελικοπτέρου
    - Υποσύστημα δεξιό τηλεχειριστήριο (rigthJoystic) : με την κίνηση του δεξιού τηλεχειριστηρίου καθορίζεται η θέση στο χώρο, του ελικοπτέρου
- Υποσύστημα διακόπτη (Switch) (Εικόνα Α.2.4): διαχειρίζεται το διακόπτη (switch) που υπάρχει στην εφαρμογή

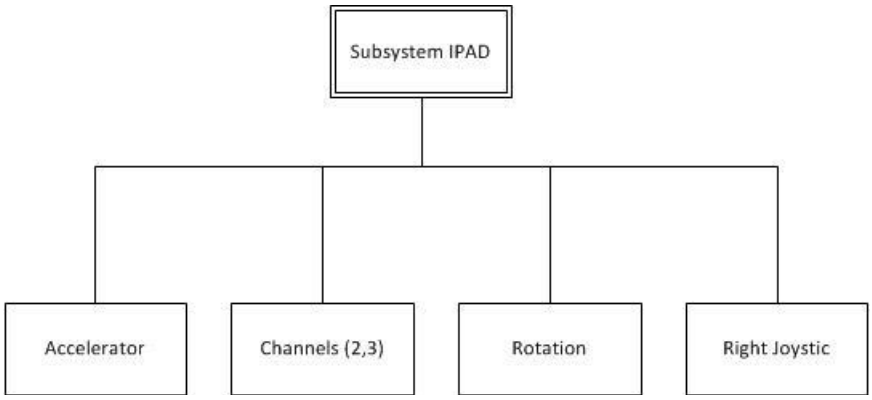
Οι κοινές κλάσεις όλων των υποσυστημάτων παρουσιάζονται στην Εικόνα Α.2.5, όπου η κλάση Communication αφορά την επικοινωνία της εφαρμογής με το ελικόπτερο (υπεύθυνη για την αποστολή των πακέτων).

Η εφαρμογή PM JOYSTIC και τα υποσυστήματα της



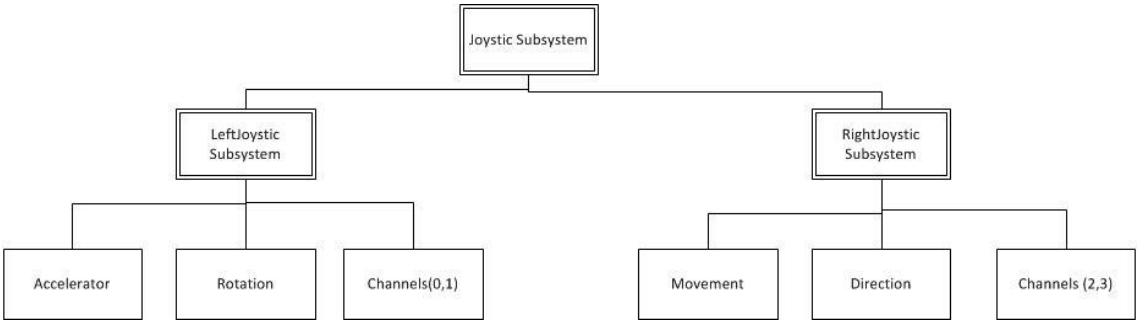
Εικόνα Α.2.1: Σύστημα - Υποσυστήματα

Υποσύστημα iPad



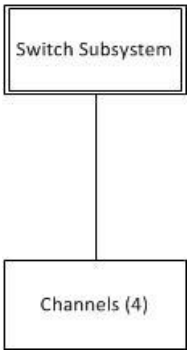
Εικόνα Α.2.2: Υποσύστημα iPad

Υποσύστημα τηλεχειριστήριο (Joystic)



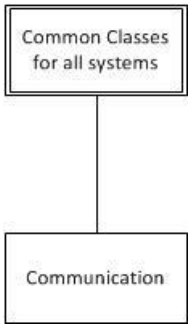
Εικόνα Α.2.3: Υποσύστημα τηλεχειριστήριο (Joystick)

Υποσύστημα Διακόπτης (Switch)



Εικόνα Α.2.4: Υποσύστημα διακόπτης

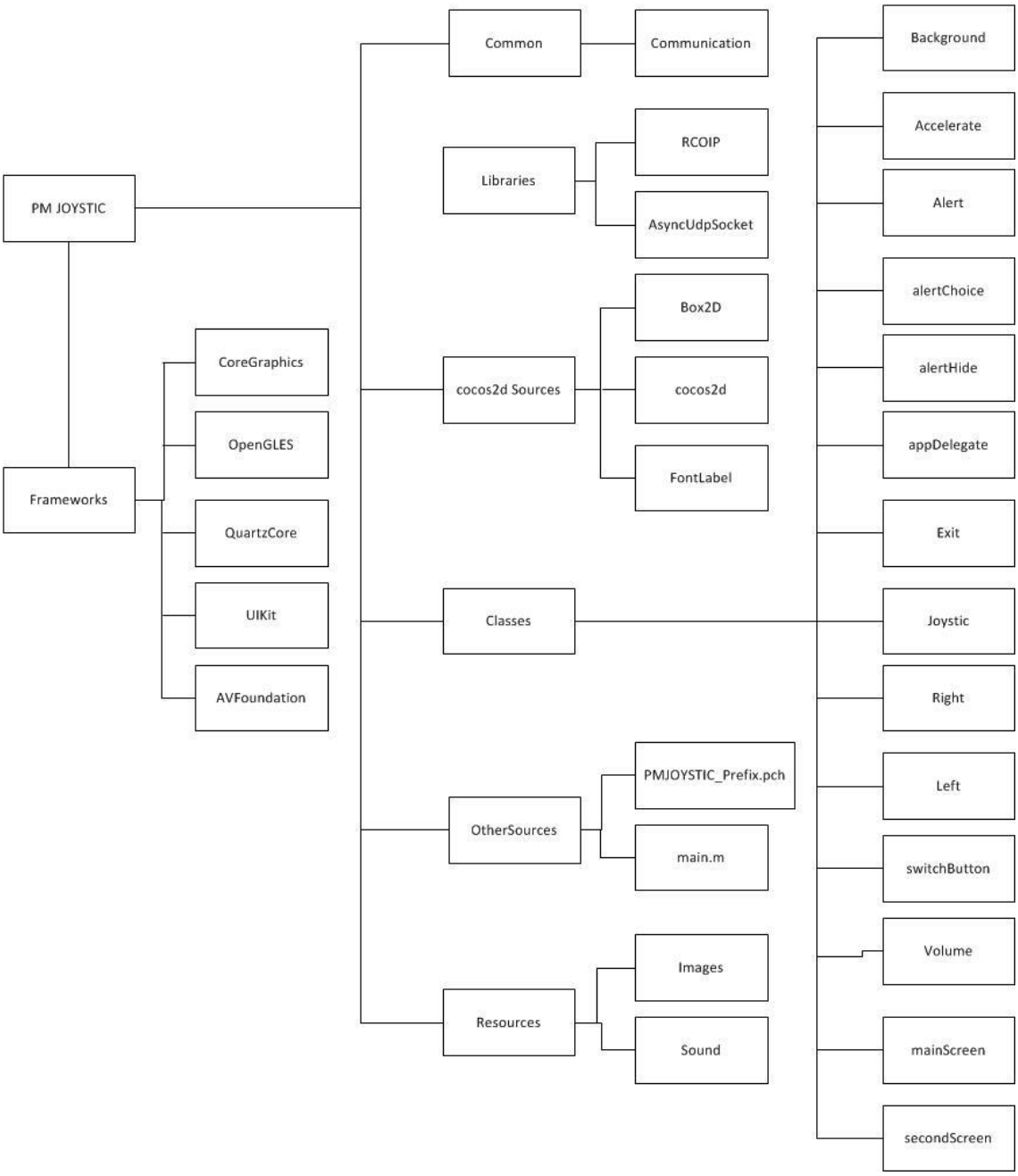
Κοινές κλάσεις για όλα τα υποσυστήματα



Εικόνα Α.2.5: Κοινές κλάσεις για όλα τα υποσυστήματα

Δομή της εφαρμογής (File Structure)

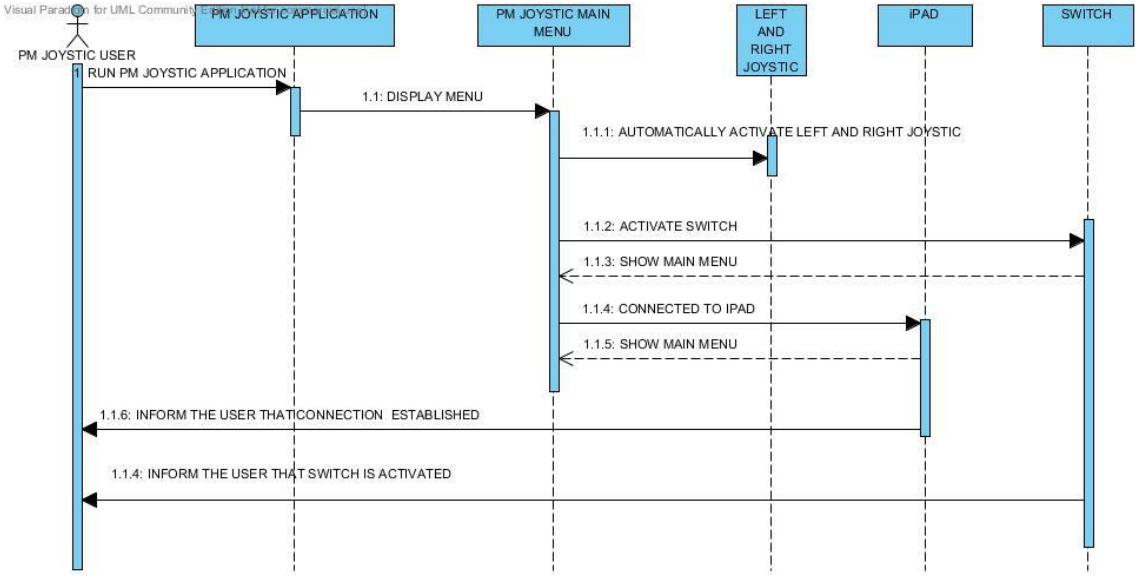
Στην εικόνα Α.2.6, βλέπουμε την δομή της εφαρμογής PM JOYSTIC.



Εικόνα Α.2.6: Δομή της εφαρμογής PM JOYSTIC

Διάγραμμα ακολουθίας (Sequence diagram)

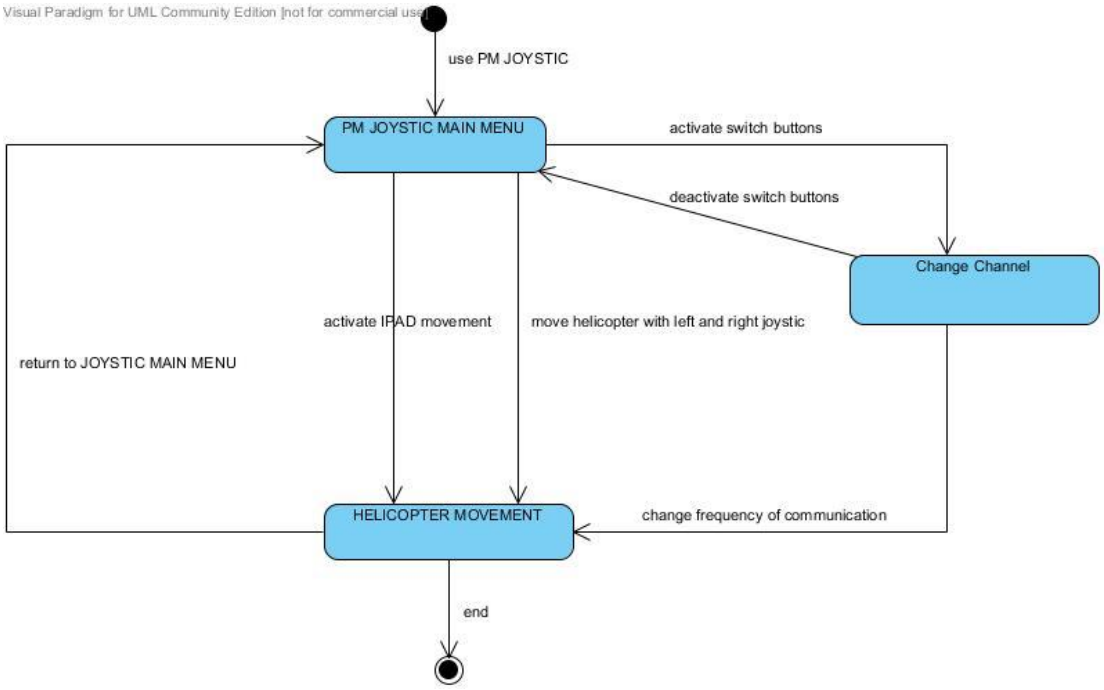
Στην εικόνα Α.2.7, βλέπουμε το διάγραμμα ακολουθίας της εφαρμογής PM JOYSTIC.



Εικόνα Α.2.7: Διάγραμμα ακολουθίας της εφαρμογής PM JOYSTIC

Διάγραμμα καταστάσεων (State diagram)

Στην εικόνα Α.2.8, βλέπουμε το διάγραμμα κατστάσεων της εφαρμογής PM JOYSTIC.



Εικόνα Α.2.8: Διάγραμμα καταστάσεων της εφαρμογής PM JOYSTIC

Διάγραμμα κλάσεων (Class diagram)

Στην εικόνα Α.2.9, βλέπουμε το διάγραμμα των κλάσεων της εφαρμογής. Όλοι οι μεθόδοι των κλάσεων είναι δημόσιοι, και όλες οι μεταβλητές των κλάσεων είναι ιδιωτικές (private).

Οι κλάσεις που κληρονομούν από την κλάση **CCLayer** είναι οι εξής:

- Background
- Accelerate
- Exit
- Joystic
- Volume
- viewLayer
- ipLayer
- switchButton

Οι κλάσεις που κληρονομούν από την κλάση **CCScene** είναι οι εξής:

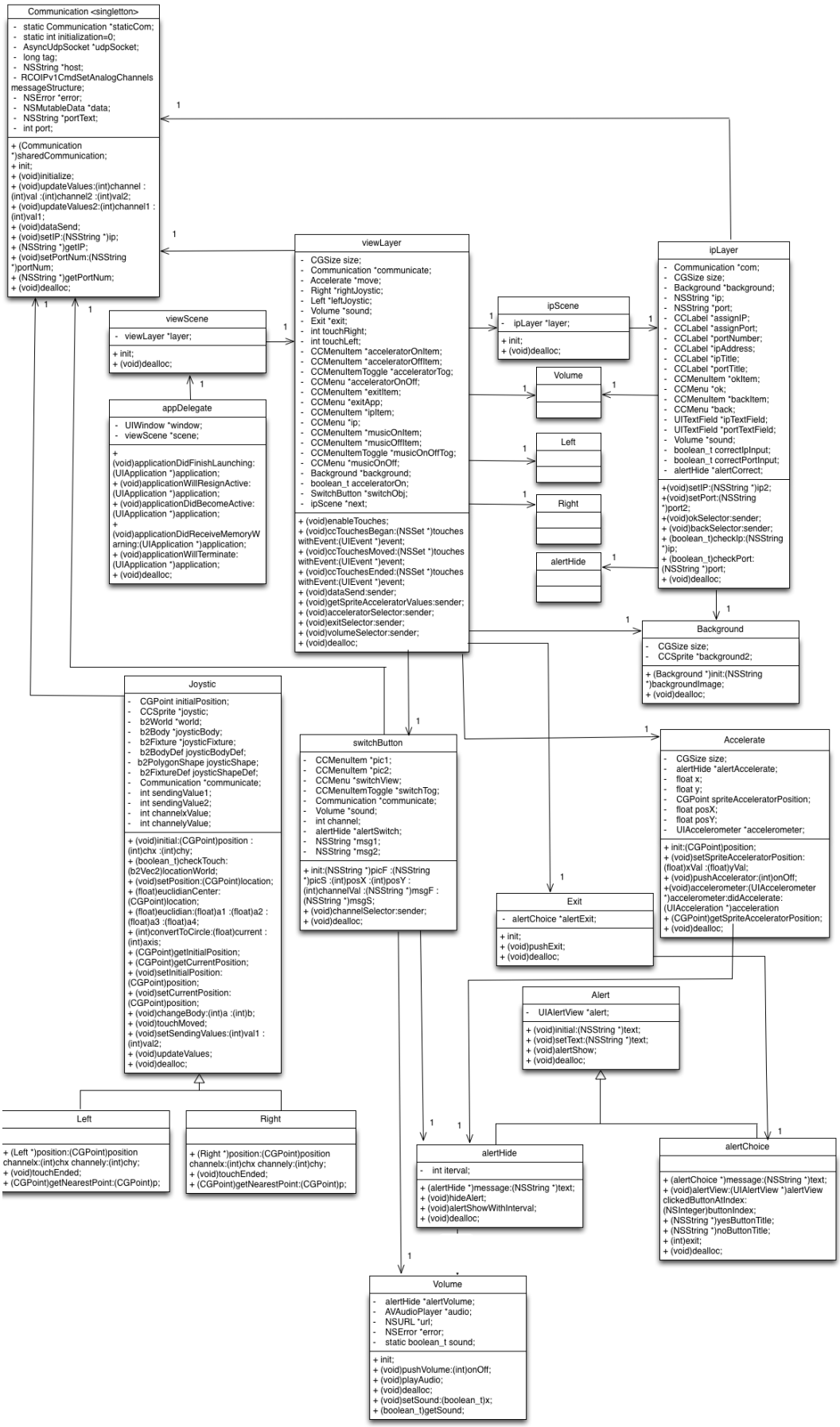
- viewScene
- ipScene

Οι κλάσεις που κληρονομούν από την κλάση **UIViewController** είναι οι εξής:

- Alert

Οι κλάσεις που κληρονομούν από την κλάση **NSObject** είναι οι εξής:

- appDelegate
- Communication



Εικόνα Α.2.9: Διάγραμμα κλάσεων της εφαρμογής PM JOYSTICK

Α3. Πλάνο ελέγχου συστήματος (Testing)

PM JOYSTIC

Software Quality Assurance Plan

1. Εισαγωγή

Σκοπός του εγγράφου αυτού είναι να παρουσιάσει το σχέδιο και τα αποτελέσματα των δοκιμαστικών ελέγχων που έχουν γίνει για το λογισμικό **PM JOYSTIC**.

1.1. Αντικειμενικοί στόχοι

Αντικειμενικοί στόχοι του δοκιμαστικού ελέγχου είναι να γίνει δοκιμαστικός έλεγχος ξεχωριστά για κάθε ένα από τα 3 υποσυστήματα τα οποία αναπτύχθηκαν για το λογισμικό PM JOYSTIC. Ο έλεγχοι αυτοί θα πρέπει να γίνουν μέχρι το Σάββατο **11/05/2013**.

1.2. Στρατηγική δοκιμών

Ο δοκιμαστικός έλεγχος θα γίνει πρωταρχικά σε κάθε ενότητα κάθε υποσυστήματος και μετά σε κάθε υποσύστημα ξεχωριστά. Αφού οι δοκιμαστικοί έλεγχοι είναι επιτυχημένοι για τα τρία υποσυστήματα τότε ο έλεγχος θα γίνει για όλο το σύστημα.

Τα υποσυστήματα στα οποία θα γίνει ο δοκιμαστικός έλεγχος είναι:

- Το υποσύστημα iPad
- Το υποσύστημα του αριστερού και δεξιού τηλεχειριστηρίου
- Το υποσύστημα με το διακόπτη

1.3. Αντικειμενικός σκοπός

Ο δοκιμαστικός έλεγχος θα γίνεται καθόλη τη διάρκεια ανάπτυξης του λογισμικού. Ο δοκιμαστικός έλεγχος περιλαμβάνει τον έλεγχο των τριών υποσυστημάτων ξεχωριστά με την αναβάθμιση του λογισμικού ανάπτυξης του προϊόντος. Ο δοκιμαστικός έλεγχος γινόταν καθόλη τη διάρκεια ανάπτυξης του λογισμικού με προ-προγραμματισμένους δοκιμαστικούς ελέγχους του λογισμικού και προκαθορισμένο χρονοδιάγραμμα δοκιμαστικού ελέγχου με το iPad και Arduino.

1.4. Υλικό αναφορών

Πιο κάτω φαίνεται ο κατάλογος με όλες τις αναφορές σε άλλα έγγραφα

- Απαιτήσεις Συστήματος (PM JOYSTIC-SR)
- Προδιαγραφές Σχεδιασμού (PM JOYSTIC-SD)

2. Αντικείμενα Ελέγχου

Τα έγγραφα PM JOYSTIC-SR και PM JOYSTIC-SD θα είναι σημείο αναφοράς για την ανάπτυξη του συστήματος. Θα ετοιμαστεί επίσης οδηγός χρήσης για να μπορέσει κάποιος να τρέξει την εφαρμογή. Όσον αφορά την επαλήθευση και έγκριση του λογισμικού αυτή θα γίνεται ανά τακτά χρονικά διαστήματα και κατά μεγάλες αλλαγές ή εντοπισμό σφαλμάτων στον κώδικα.

3. Ενότητες προγράμματος

Οι έλεγχοι που πρέπει να γίνουν είναι πρώτο ο έλεγχος στο υποσύστημα iPad. Ότι δηλαδή το λογισμικό μέσω του iPad δίδει εντολές στο υλικό για κίνηση. Δεύτερο είναι ο έλεγχος στο υποσύστημα JOYSTIC , δηλαδή στην κίνηση του υλικού Arduino μέσω του αριστερού και δεξιού τηλεχειριστηρίου του λογισμικού. Και τέλος είναι ο έλεγχος του υποσυστήματος του διακόπτη.

3.1. Διαδικασίες ελέγχου εργασίας

Οι έλεγχοι θα γίνουν με την πιο κάτω σειρά:

- Έλεγχος στο υποσύστημα iPad
- Έλεγχος στο υποσύστημα JOYSTIC
- Έλεγχος στο υποσύστημα διακόπτης

Τέλος αφού γίνουν όλοι οι πιο πάνω έλεγχοι θα γίνει και συνολικός έλεγχος του συστήματος με τη συσκευή Arduino.

3.2. Διαδικασία χρηστών

Στο τέλος κάθε δοκιμαστικού ελέγχου θα κρατούνται σημειώσεις για τη δημιουργία του τελικού εγχειριδίου για χρήση του συστήματος. Θα ετοιμαστεί αρχείο Read Me για την διαδικασία εκκίνησης του λογισμικού PM JOYSTIC το οποίο θα

ενσωματωθεί ως παράρτημα στην διπλωματική εργασία (ΠΑΡΑΡΤΗΜΑ Γ).

4. Λειτουργίες που θα ελεγχτούν

Οι λειτουργίες που θα ελεχθούν για το σύστημα είναι οι ακόλουθες:

- Όλες οι λειτουργίες του υποσυστήματος JOYSTIC
  - ✓ Όλες οι λειτουργίες του αριστερού τηλεχειριστηρίου
  - ✓ Όλες οι λειτουργίες του δεξιού τηλεχειριστηρίου
- Όλες οι λειτουργίες του υποσυστήματος iPad
- Η λειτουργία ενεργοποίησης του διακόπτη.

5. Λειτουργίες που ΔΕΝ θα ελεγχτούν

Οι έλεγχοι κίνησης της συσκευής Arduino με το υποσύστημα JOYSTIC και iPad λόγω μη διαθεσιμότητας του υλικού Arduino.

6. Μεθοδολογία

Η γενική μεθοδολογία που ακολουθήθηκε είναι πρώτα ο έλεγχος των υπομονάδων και ρουτινών για σωστή λειτουργία και εν συνεχεία η ενσωμάτωση τους σε ενότητες υποσυστήματα. Μετά από την υλοποίηση των υποσυστημάτων η ενοποίηση και η ενσωμάτωση τους στο τελικό σύστημα.

6.1. Έλεγχος υποσυστημάτων (Συστήματος)

Έλεγχος των μονάδων κάθε ενότητας για διασφάλιση της ορθότητας της λογικής του προγράμματος και ότι η μονάδα είναι ολοκληρωμένη και ότι έχει το αποτέλεσμα που είχε κατά το σχεδιασμό.

6.2. Έλεγχος ενσωματώσεων

Έλεγχος του υποσυστήματος του iPad, με το υλικό του iPad.

6.3. Έλεγχος μετατροπών

Έχει γίνει ο κατάλληλος δοκιμαστικός έλεγχος του συστήματος και οι κατάλληλες τροποποιήσεις έτσι ώστε το σύστημα να τα τρέχει με τη τελευταία έκδοση ανάπτυξης του λογισμικού Xcode 4.6.2.

6.4. Έλεγχος στο περιβάλλον παραγωγής

Ο προγραμματισμένος έλεγχος με την συσκευή Arduino δεν πραγματοποιήθηκε λόγω μη διαθεσιμότητας της συσκευής. Οι υπόλοιποι έλεγχοι έχουν πραγματοποιηθεί.

6.5. Έλεγχος αποδοχής

Έχει γίνει έλεγχος και είναι ο χρόνος ανταπόκρισης και αποδοχής της εφαρμογής (ανταπόκριση κάθε 1 χιλιοστό του δευτερολέπτου).

7. ΚΡΙΤΗΡΙΟ ΑΠΟΤΥΧΙΑΣ / ΕΠΙΤΥΧΙΑΣ

Το κριτήριο επιτυχίας / αποτυχίας ενός δοκιμαστικού ελέγχου είναι το αποτέλεσμα του δοκιμαστικού ελέγχου. Αν το αποτέλεσμα του δοκιμαστικού έλεγχου είναι η λειτουργία του υποσυστήματος / συστήματος κατά 90% και άνω τότε ο δοκιμαστικός έλεγχος θεωρείται επιτυχημένος και σε άλλη περίπτωση αποτυχημένος.

7.1. Κριτήριο αναβολής

Μη διαθεσιμότητα των απαραίτητων υλικών (Arduino)

7.2. Κριτήριο επανάληψης

Στην περίπτωση που το επιθυμητό αποτέλεσμα του δοκιμαστικού ελέγχου ήταν αποτυχημένο θα γίνουν βελτιωτικές προσθήκες στην σχεδίαση και παραμετροποίηση του κώδικα έτσι ώστε το επιθυμητό αποτέλεσμα να γίνει επιτυχημένο.

7.3. Κριτήριο έγκρισης

Το επιθυμητό αποτέλεσμα του δοκιμαστικού ελέγχου ήταν 90% - 100% επιτυχημένο.

8. ΔΙΑΔΙΚΑΣΙΑ ΕΛΕΓΧΟΥ

Τα παρακάτω σημεία συνοψίζουν την διαδικασία ελέγχου του λογισμικού.

8.1. Παραδοτέα ελέγχου

Κατά τον έλεγχο των υποσυστημάτων έχει γίνει καταγραφή των τιμών που στέλνει η εφαρμογή σε ένα συγκεκριμένο IP (που ανήκει στο ίδιο δίκτυο με την συσκευή) μέσω του λογισμικού wireshark.

8.2. Εργασίες ελέγχου

Κατά τις εργασίες ελέγχου έχουν ελεγχθεί τα κανάλια 0,1 ,2 , 3 και 4 ότι παίρνουν τις ορθές τιμές.

8.3. Χρονοδιάγραμμα

Α/Α	**Περιγραφή**	**Ημερομηνία**
1.	Έλεγχος αριστερού τηλεχειριστηρίου	5.5.2013
2.	Έλεγχος δεξιού τηλεχειριστηρίου	6.5.2013
3	Έλεγχος υποσυστήματος τηλεχειριστηρίου	7.5.2013
4.	Έλεγχος υποσυστήματος iPad	8.5.2013
5.	Έλεγχος αριστερού τηλεχειριστηρίου, με το υποσύστημα iPad	9.5.2013
6.	Έλεγχος διακόπτη	10.5.2013
7.	Έλεγχος ολόκληρου του συστήματος (PM JOYSTIC) – Με τον μικροελεγκτή Arduino	11.5.2013

9. Έλεγχοι

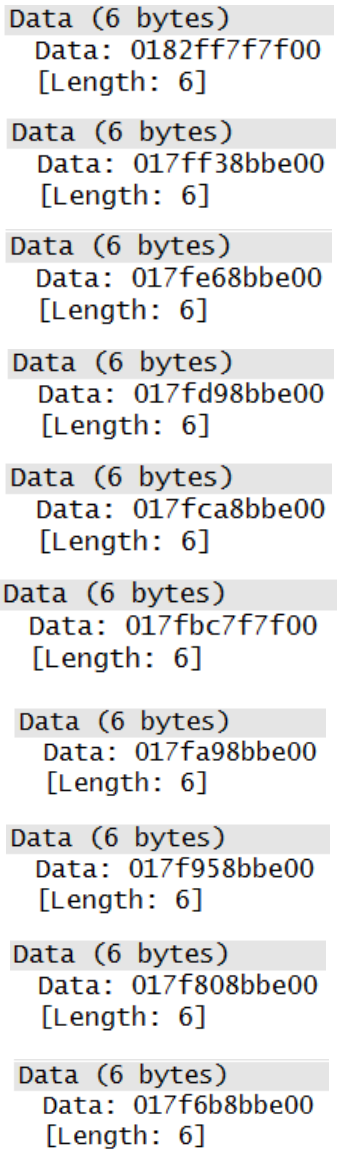
Για να ελέγξουμε ότι τα κανάλια παίρνουν τις τιμές σωστές θα δείξουμε κάποια από τα πακέτα που παίρνουμε από το λογισμικό wireshark.

9.1. Έλεγχος αριστερού τηλεχειριστηρίου

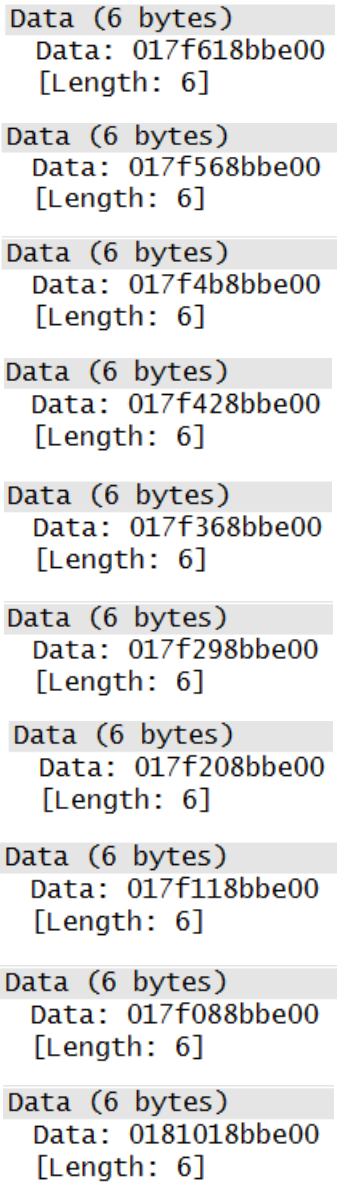
Πιο κάτω θα δούμε τα περιεχόμενα 40 πακέτων. Στην πρώτη περίπτωση (Εικόνα Α.3.1 τα πρώτα 10 πακέτα και Εικόνα Α.3.2 τα υπόλοιπα 10 πακέτα) θα γίνει δοκιμή της κίνησης του τηλεχειριστηρίου από κάτω προς τα πάνω (20 πακέτα), και στην δεύτερη περίπτωση (Εικόνα Α.3.3 τα πρώτα 10 πακέτα και Εικόνα Α.3.3 τα υπόλοιπα 10 πακέτα) θα γίνει δοκιμή της κίνησης του τηλεχειριστηρίου από τα αριστερά στα δεξιά (20 πακέτα).

<u>Κίνηση τηλεχειριστηρίου από κάτω προς τα πάνω</u>

Στην περίπτωση αυτή μας ενδιαφέρουν οι τιμές που παίρνουμε στο κανάλι 1 (αλλαγή τιμής στον άξονα y)



Εικόνα Α.3.1: Δεδομένα πακέτων 1 - 10 στην περίπτωση κίνησης του αριστερού τηλεχειριστηριου από κάτω προς τα πάνω



Εικόνα Α.3.2: Δεδομένα πακέτων 11 - 20 στην περίπτωση κίνησης του αριστερού του τηλεχειριστηρίου από κάτω προς τα πάνω

Αποτελέσματα

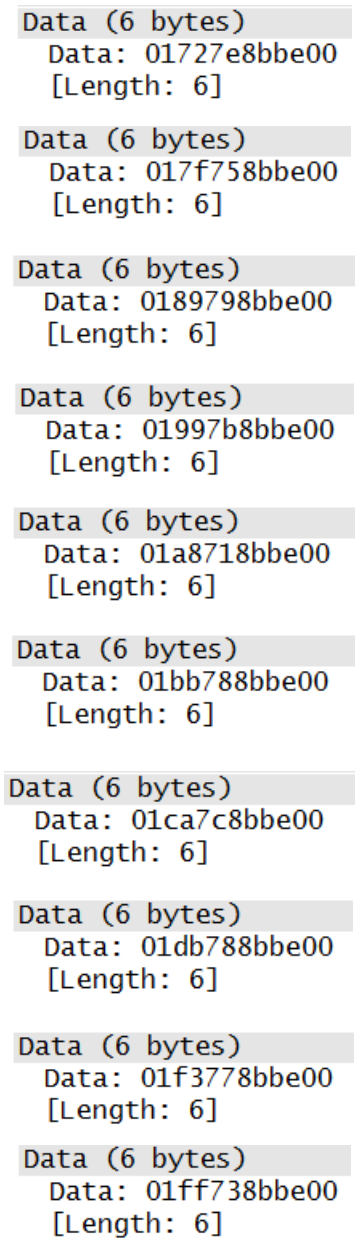
Κανάλι 1 – Δεκαδικό σύστημα
255
243
230
217
202
188
169
149
128
107
97
86
75
66
54
41
32
17
8
1

Κίνηση τηλεχειριστηρίου από τα αριστερά προς τα δεξιά

Στην περίπτωση αυτή μας ενδιαφέρουν οι τιμές που παίρνουμε στο κανάλι 0 (αλλαγή τιμής στον άξονα x)

```
Data (6 bytes)
  Data: 0101718bbe00
  [Length: 6]

Data (6 bytes)
  Data: 010c6e8bbe00
  [Length: 6]

Data (6 bytes)
  Data: 01196c8bbe00
  [Length: 6]

Data (6 bytes)
  Data: 01236f8bbe00
  [Length: 6]

Data (6 bytes)
  Data: 012d6d8bbe00
  [Length: 6]

Data (6 bytes)
  Data: 01387f8bbe00
  [Length: 6]

Data (6 bytes)
  Data: 01426f8bbe00
  [Length: 6]

Data (6 bytes)
  Data: 01517c8bbe00
  [Length: 6]

Data (6 bytes)
  Data: 015b808bbe00
  [Length: 6]

Data (6 bytes)
  Data: 01647d8bbe00
  [Length: 6]
```

Εικόνα Α.3.3: Δεδομένα πακέτων 1 - 10 στην περίπτωση κίνησης του αριστερού τηλεχειριστηριου από αριστερά προς τα δεξιά



Εικόνα Α.3.4: Δεδομένα πακέτων 11 - 20 στην περίπτωση κίνησης του αριστερού τηλεχειριστηρίου από αριστερά προς τα δεξιά

Αποτελέσματα

Κανάλι 0 – Δεκαδικό σύστημα
1
16
25
35
45
56
66
81
91
100
114
127
137
153
168
187
202
219
243
255

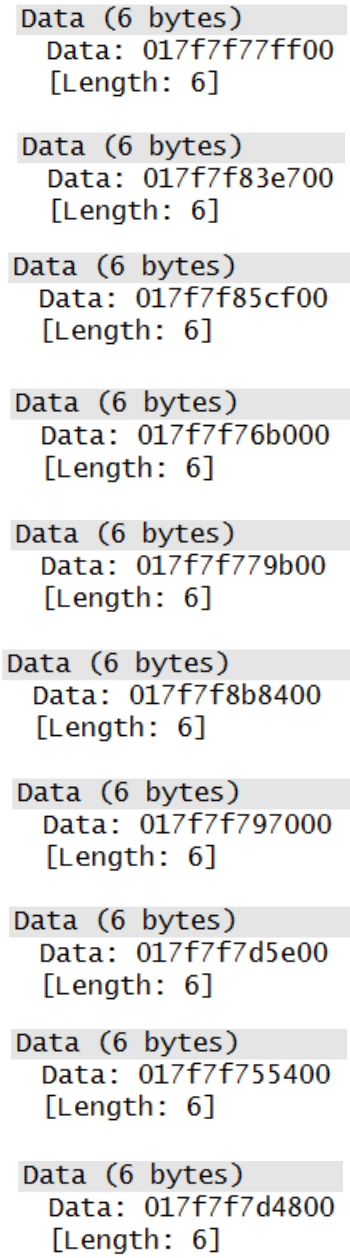
Με την κίνηση του αριστερού τηλεχειριστηρίου από κάτω προς τα πάνω βλέπουμε ότι οι τιμές μειώνονται από το 255 στο 1. Με την κίνηση από αριστερά στα δεξιά βλέπουμε ότι οι τιμές αυξάνονται από το 1 στο 255. Έτσι συμπεραίνουμε ότι τα αποτελέσματα μας είναι ορθά.

9.2. Έλεγχος δεξιού τηλεχειριστηρίου

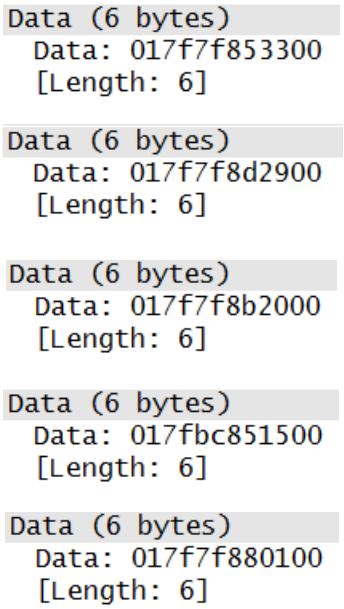
Πιο κάτω θα δούμε τα περιεχόμενα 30 πακέτων. Στην πρώτη περίπτωση (Εικόνα Α.3.5 τα πρώτα 10 πακέτα και Εικόνα Α.3.6 τα υπόλοιπα 5 πακέτα) θα γίνει δοκιμή της κίνησης του τηλεχειριστηρίου από κάτω προς τα πάνω (15 πακέτα), και στην δεύτερη περίπτωση (Εικόνα Α.3.7 τα πρώτα 10 πακέτα και Εικόνα Α.3.8 τα υπόλοιπα 5 πακέτα) θα γίνει δοκιμή της κίνησης του τηλεχειριστηρίου από τα αριστερά στα δεξιά (15 πακέτα).

Κίνηση τηλεχειριστηρίου από κάτω προς τα πάνω

Στην περίπτωση αυτή μας ενδιαφέρουν οι τιμές που παίρνουμε στο κανάλι 3 (αλλαγή τιμής στον άξονα y)



Εικόνα Α.3.5: Δεδομένα πακέτων 1 - 10 στην περίπτωση κίνησης του δεξιού τηλεχειριστηρίου από κάτω προς τα πάνω



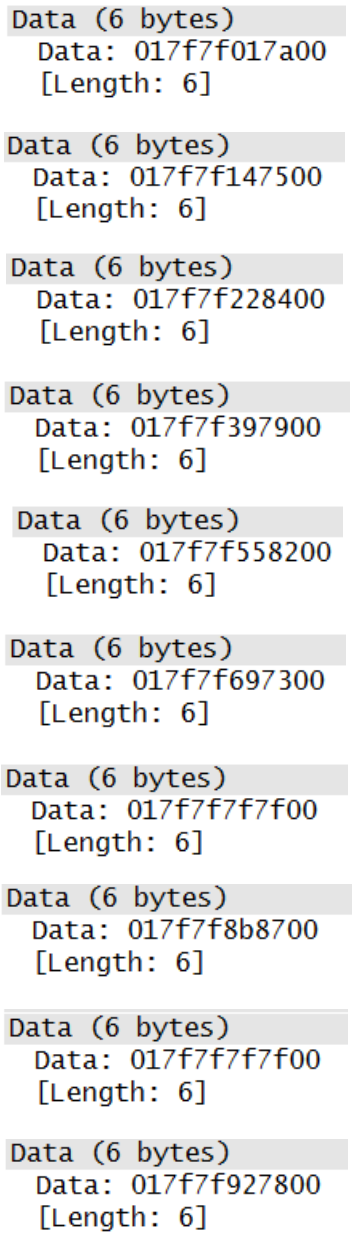
Εικόνα Α.3.6: Δεδομένα πακέτων 11 - 15 στην περίπτωση κίνησης του τηλεχειριστηρίου από κάτω προς τα πάνω

Αποτελέσματα

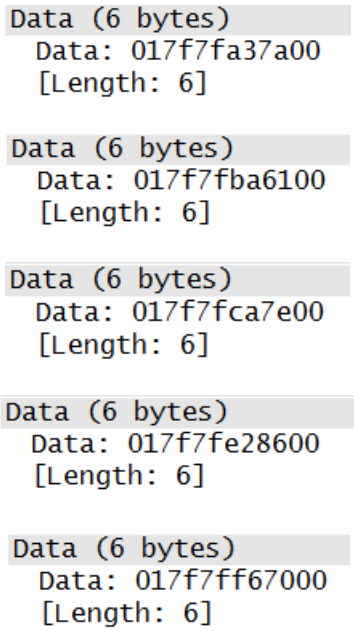
Κανάλι 3 – Δεκαδικό σύστημα
255
231
207
176
155
132
112
94
84
72
51
41
32
20
1

Κίνηση τηλεχειριστηρίου από τα αριστερά προς τα δεξιά

Στην περίπτωση αυτή μας ενδιαφέρουν οι τιμές που παίρνουμε στο κανάλι 2 (αλλαγή τιμής στον άξονα x)



Εικόνα Α.3.7: Δεδομένα πακέτων 1 - 10 στην περίπτωση κίνησης του δεξιού τηλεχειριστηρίου από αριστερά προς τα δεξιά



Εικόνα Α.3.8: Δεδομένα πακέτων 11 - 15 στην περίπτωση κίνησης του δεξιού τηλεχειριστηρίου από αριστερά προς τα δεξιά

Αποτελέσματα

Κανάλι 2 – Δεκαδικό σύστημα
1
20
40
57
85
105
127
139
146
163
186
202
226
246
255

Με την κίνηση του τηλεχειριστηρίου από κάτω προς τα πάνω βλέπουμε ότι οι τιμές μειώνονται από το 255 στο 1. Με την κίνηση από αριστερά στα δεξιά βλέπουμε ότι οι τιμές αυξάνονται από το 1 στο 255. Έτσι συμπεραίνουμε ότι τα αποτελέσματα μας είναι ορθά.

9.3. Έλεγχος υποσυστήματος τηλεχειριστηρίου

Στην περίπτωση αυτή θα δούμε τα περιεχόμενα 5 πακέτων, τα οποία φαίνονται στην Εικόνα Α.3.9. Η κίνηση των δύο τηλεχειριστηρίων γίνεται ταυτόχρονα και προς τα πάνω. Άρα, μας ενδιαφέρουν οι τιμές στα κανάλια 1 και 3.

```
Data (6 bytes)
  Data: 017e5a757200
  [Length: 6]
Data (6 bytes)
  Data: 018b497c6300
  [Length: 6]
Data (6 bytes)
  Data: 01812e784500
  [Length: 6]
Data (6 bytes)
  Data: 0181127e2600
  [Length: 6]
Data (6 bytes)
  Data: 017a01830100
  [Length: 6]
```

Εικόνα Α.3.9: Δεδομένα 5 πακέτων στην περίπτωση κίνησης των δύο τηλεχειριστηρίων προς τα πάνω

Αποτελέσματα

Κανάλι 1 – Δεκαδικό σύστημα	Κανάλι 3 – Δεκαδικό σύστημα
90	114
73	99
46	69
18	38
1	1

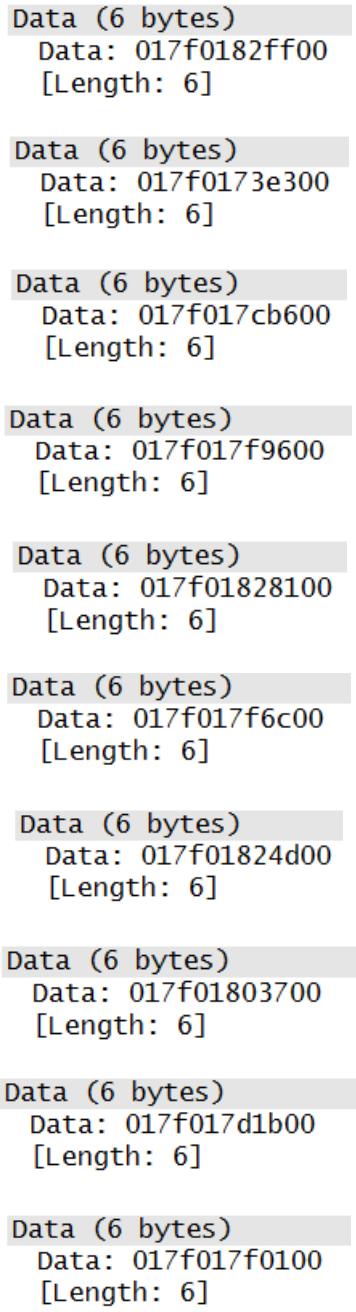
Με την κίνηση και των δύο τηλεχειριστηρίων ταυτόχρονα και προς τα πάνω, βλέπουμε ότι οι τιμές των καναλιών 1 και 3 μειώνονται προς το 1, άρα τα αποτελέσματα μας είναι ορθά.

9.4. Έλεγχος υποσυστήματος iPad

Πιο κάτω θα δούμε τα περιεχόμενα 20 πακέτων. Στην πρώτη περίπτωση (Εικόνα Α.3.10)θα γίνει δοκιμή της κίνησης του τηλεχειριστηρίου (μέσω του iPad) από κάτω προς τα πάνω (10 πακέτα), και στην δεύτερη περίπτωση (Εικόνα Α.3.11) θα γίνει δοκιμή της κίνησης του τηλεχειριστηρίου (μέσω του iPad) από τα αριστερά στα δεξιά (10 πακέτα).

Κίνηση τηλεχειριστηρίου από κάτω προς τα πάνω

Στην περίπτωση αυτή μας ενδιαφέρουν οι τιμές που παίρνουμε στο κανάλι 3 (αλλαγή τιμής στον άξονα y). Τα πακέτα που πήραμε στην περίπτωση αυτή απεικονίζονται στην Εικόνα Α.3.10.



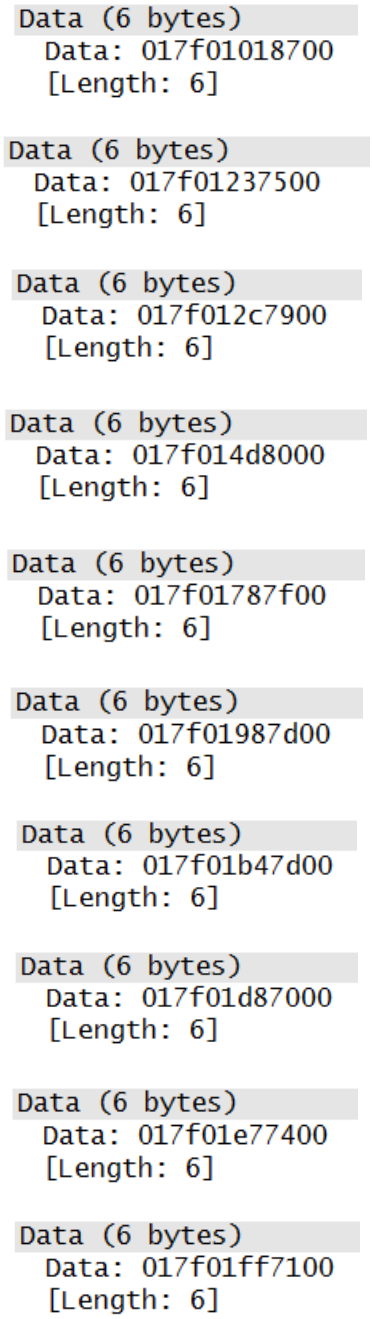
Εικόνα Α.3.10: Δεδομένα 10 πακέτων στην περίπτωση κίνησης του ελικοπτέρου με την κίνηση της συσκευής (κίνηση του δεξιού τηλεχειριστηρίου από κάτω προς τα πάνω)

Αποτελέσματα

Κανάλι 3 – Δεκαδικό σύστημα
255
227
182
150
129
108
77
55
27
1

Κίνηση τηλεχειριστηρίου από αριστερά προς τα δεξιά

Στην περίπτωση αυτή μας ενδιαφέρουν οι τιμές που παίρνουμε στο κανάλι 2 (αλλαγή τιμής στον άξονα x)



Εικόνα Α.3.11: Δεδομένα 10 πακέτων στην περίπτωση κίνησης του ελικοπτέρου με την κίνηση της συσκευής (κίνηση του δεξιού τηλεχειριστηρίου από αριστερά προς τα δεξιά

Αποτελέσματα

Κανάλι 2 – Δεκαδικό σύστημα
1
35
44
77
120
152
180
216
231
255

Με την κίνηση του τηλεχειριστηρίου από κάτω προς τα πάνω βλέπουμε ότι οι τιμές μειώνονται από το 255 στο 1. Με την κίνηση από αριστερά στα δεξιά βλέπουμε ότι οι τιμές αυξάνονται από το 1 στο 255. Έτσι συμπεραίνουμε ότι τα αποτελέσματα μας είναι ορθά.

9.5. Έλεγχος αριστερού τηλεχειριστηρίου, με το υποσύστημα iPad

Στην Εικόνα Α.3.12 βλέπουμε τα περιεχόμενα 5 πακέτων που παίρνουμε, με την κίνηση του αριστερού τηλεχειριστηρίου προς τα πάνω και με τη κίνηση του iPad (που κατευθύνει το δεξιό τηλεχειριστήριο προς τα πάνω). Άρα, μας ενδιαφέρουν οι τιμές στα κανάλια 1 και 3.

```
Data (6 bytes)
  Data: 017f64805d00
  [Length: 6]
Data (6 bytes)
  Data: 017f55845000
  [Length: 6]
Data (6 bytes)
  Data: 017f34822c00
  [Length: 6]
Data (6 bytes)
  Data: 017f20801d00
  [Length: 6]
Data (6 bytes)
  Data: 017f018f0100
  [Length: 6]
```

Εικόνα Α.3.12: Δεδομένα 5 πακέτων με την κίνηση του αριστερού τηλεχειριστηρίου προς τα πάνω και με τη κίνηση του iPad (που κατευθύνει το δεξιό τηλεχειριστήριο προς τα πάνω)

Αποτελέσματα

Κανάλι 1 – Δεκαδικό σύστημα	Κανάλι 3 – Δεκαδικό σύστημα
100	93
85	80
52	44
32	29
1	1

Με την κίνηση του αριστερού τηλεχειριστηρίου προς τα πάνω και τη κίνηση του iPad, βλέπουμε ότι οι τιμές των καναλιών 1 και 3 μειώνονται προς το 1, άρα τα αποτελέσματα μας είναι ορθά.

9.6. Έλεγχος διακόπτη

Πιο κάτω θα δούμε τα περιεχόμενα του πακέτου στην πρώτη περίπτωση (Εικόνα Α.3.13) όπου ο διακόπτης είναι απενεργοποιημένος και στην δεύτερη περίπτωση όπου είναι ενεργοποιημένος (Εικόνα Α.3.14). Ο διακόπτης αφορά το κανάλι 4.

Απενεργοποιημένος διακόπτης

```
Data (6 bytes)
  Data: 017f7f7f7f00
  [Length: 6]
```

Εικόνα Α.3.13: Δεδομένα πακέτου σε περίπτωση που ο διακόπτης είναι απενεργοποιημένος

Ενεργοποιημένος διακόπτης

```
Data (6 bytes)
  Data: 017f7f7f7f01
  [Length: 6]
```

Εικόνα Α.3.14: Δεδομένα πακέτου σε περίπτωση που ο διακόπτης είναι ενεργοποιημένος

Από τα πακέτα βλέπουμε ότι τα αποτελέσματα μας είναι ορθά αφού όταν ο διακόπτης είναι απενεργοποιημένος παίρνουμε την τιμή 0 στο κανάλι 4, και στην δεύτερη περίπτωση παίρνουμε την τιμή 1. Άρα τα αποτελέσματα μας είναι ορθά.

10. Απαιτήσεις περιβάλλοντος

Το περιβάλλον δοκιμαστικού ελέγχου πρέπει να είναι ανοικτός χώρος για δοκιμή της εφαρμογής με το Arduino. Επίσης το περιβάλλον πρέπει να είναι εφοδιασμένο με συσκευή δικτύου ασύρματης πρόσβασης.

10.1. Υλικό

Το υλικό που είναι απαραίτητο για την διεξαγωγή των δοκιμαστικών ελέγχων είναι:

- Υπολογιστής MAC
- Συσκευή iPad
- Συσκευή Arduino
- Δρομολογητής (Router)

10.2. Λογισμικό

Το λογισμικό που είναι απαραίτητο για την διεξαγωγή των δοκιμαστικών ελέγχων είναι:

- Xcode IDE:
  - ✓ Apple LLVM Compiler
  - ✓ IOS Simulator
- Wireshark

10.3. Ασφάλεια

Στο περιβάλλον το οποίο θα γίνει ο δοκιμαστικός έλεγχος δεν πρέπει να έχει μικρά παιδιά και να είναι ελεύθερο από κάθε είδους εμπόδια.

10.4. Ρίσκα και υποθέσεις

Το μόνο ρίσκο και περιορισμός που υπήρξε κατά την διάρκεια του ελέγχου ήταν η

διαθεσιμότητα του υλικού Arduino. Το υλικό Arduino ήταν απαραίτητο για όλους τους ελέγχους αλλά λόγω προβλημάτων μη λειτουργίας της συσκευής δεν ήταν κατορθωτό να γίνει ο προγραμματισμένος δοκιμαστικός έλεγχος.

11. Διαδικασία διαχείρισης μεταβολών

Κατά την διάρκεια του δοκιμαστικού ελέγχου θα καταγράφονται οι παρατηρήσεις και τα αποτελέσματα του ελέγχου. Αν υπάρχει ανάγκη αλλαγής θα καταγράφεται και θα γίνεται καθορισμός των αλλαγών που πρέπει να γίνουν.

12. Εγκρίσεις πλάνου ελέγχου

Ονοματεπώνυμο	**Ημερομηνία**	**Υπογραφή**
Μαργαρίτα Παπαευθυμίου	……	……
Βάσος Βασιλείου	……	……

Παράρτημα Β

Στο παρόν παράρτημα θα παρουσιαστούν βασικά κομμάτια από τον κώδικα της εφαρμογής PM JOYSTIC και θα επεξηγηθούν. Τα βασικά κομμάτια που θα παρουσιαστούν είναι τα πιο κάτω:

1. Δημιουργία ενός Sprite.

2. Βασικά κομμάτια κώδικα για την κίνηση του ελικοπτέρου μέσω της κίνηση της συσκευής.

3. Βασικά κομμάτια κώδικα για την δημιουργία και την διαχείριση των ενημερωτικών και επιβεβαιωτικών μηνυμάτων (UIAlert)

4. Δημιουργία κόσμου Box2D (box2D world) για τα 2 τηλεχειριστήρια.

5. Εύρεση της τιμής που πρέπει να σταλεί στο ελικόπτερο ανάλογα με την θέση του δεξιού τηλεχειριστηρίου.

6. Εύρεση του σημείου που πρέπει να ορισθούν τα τηλεχειριστήρια σε περίπτωση που το άγγιγμα είναι εκτός των ορίων του κύκλου.

7. Βασικά κομμάτια κώδικα διαχείρισης του ήχου.

8. Κυριότερο κομμάτι του κώδικα – η **επικοινωνία** συσκευής – ελικοπτέρου

9. Δημιουργία πεδίου κειμένου (text field).

10. Κομμάτια συναρτήσεων που διαχειρίζονται τα αγγίγματα.

1. Δημιουργία ενός Sprite.

```
//constructor - returns a background object
- (Background *) init: (NSString *)backgroundImage{
    self = [super init];
    if (self != nil) {

        //initialize size of the screen
        size = [[CCDirector sharedDirector] winSize];

        //set the image black.jpg to the sprite
        background = [CCSprite spriteWithFile:backgroundImage];

        //define background (sprite) position
        background.position = ccp(size.width/2, size.height/2);

        //add the background to the layer
        [self addChild: background];
    }
    return self;
}
```

Το “background” είναι ένα αντικείμενο τύπου Sprite. Στον πιο πάνω κομμάτι κώδικα αρχικοποιούμε το Sprite αυτό με την εικόνα “backgroundImage”, στην συνέχεια θέτουμε σαν θέση του το κέντρο της οθόνης, δίνοντας του το σημείο (size.width/2, size.height/2). Στην συνέχεια προσθέτουμε το Sprite αυτό στο επίπεδο (Layer) το οποίο βρισκόμαστε έτσι ώστε να είναι ορατό στην οθόνη.

2. Βασικά κομμάτια κώδικα για την κίνηση του ελικοπτέρου μέσω της κίνηση της συσκευής.

```
accelerometer = [UIAccelerometer sharedAccelerometer];
//accelerometer update interval is 0.001
accelerometer.updateInterval = .001;

accelerometer.delegate = self;
```

Αρχικά δημιουργούμε το αντικείμενο "accelerometer" του τύπου UIAccelerometer και θέτουμε το χρόνο που θα ανανεώνονται οι τιμές του κάθε 1 χιλιοστό του δευτερολέπτου, αφού και οι τιμές στο ελικόπτερο θα στέλλονται στο ελικόπτερο κάθε 1 χιλιοστό του δευτερόλεπτο.

```
//updates the accelerometer values
- (void)accelerometer:(UIAccelerometer *)accelerometer
didAccelerate:(UIAcceleration *)acceleration {

    [self setSpriteAcceleratorPosition:acceleration.x :acceleration.y];
}
```

Η συνάρτηση "accelerometer" καλείται κάθε 1 χιλιοστό του δευτερολέπτου και διαβάζουμε μόνο τις τιμές στο άξονα x και στον άξονα y. Η τιμή στον άξονα z δεν μας ενδιαφέρει αφού με την περιστροφή της συσκευής δεν έχουμε καμιά αλλαγή στις τιμές που στέλλονται στο ελικόπτερο.

```
//this function updates the right joystic sprite position
-(void)setSpriteAcceleratorPosition:(float)xVal :(float)yVal
{
    //define the position of the sprite
    spriteAcceleratorPosition.x=posX-yVal*SPRITE_POSITION;
    spriteAcceleratorPosition.y=posY+xVal*SPRITE_POSITION;
}
```

Στην συνάρτηση setSpriteAcceleratorPosition γίνεται ο υπολογισμός και ανάθεση της θέσης του δεξιού τηλεχειριστηρίου (Sprite) με τον πολλαπλασιασμό της σταθεράς SPRITE_POSITION.

3. Βασικά κομμάτια κώδικα για την δημιουργία και την διαχείριση των ενημερωτικών και επιβεβαιωτικών μηνυμάτων. (UIAlert).

```
[alert setMessage:text];
```

Για την εισαγωγή μηνύματος στο αντικείμενο του τύπου UIAlert

```
[alert show];
```

Για την εμφάνιση του μηνύματος (alert) στη οθόνη

```
[alert addButtonWithTitle:@"Yes"];
[alert addButtonWithTitle:@"No"];
```

Για να εισάγουμε 2 κουμπιά στο μήνυμα (yes, no)

```
//called when we push an alert button
- (void)alertView:(UIAlertView *)alertView
clickedButtonAtIndex:(NSInteger)buttonIndex{

    //returns the name of the button the we pushed of the specific index
    NSString * buttonPressedName = [alertView buttonTitleAtIndex:buttonIndex];

    // if we push yes button the application is terminated
    if([buttonPressedName isEqualToString: [self yesButtonTitle]])
    {
        NSLog(@"Application exit");
        [self exit];
    }
    // else do nothing
    else if ([buttonPressedName isEqualToString:[self noButtonTitle]])
        NSLog(@"Application not exit");}
```

Η συνάρτηση alertView καλείται όταν πατήσουμε ένα κουμπί από το μήνυμα (alert) και αναλόγως από το κουμπί που θα πατήσουμε εκτελείται ο αντίστοιχος κώδικας. Δηλαδή, αν πατηθεί το κουμπί Yes εκτελείται το πρώτο if και τερματίζεται η εφαρμογή, αλλιώς αν πατηθεί το No εκτελείται το else και φεύγει το μήνυμα από την οθόνη.

```
//hides the alert from the screen
- (void) hideAlert {
        [alert dismissWithClickedButtonIndex:0 animated:YES];}
```

Για να εξαφανιστεί το μήνυμα από την οθόνη χρησιμοποιούμε την πιο πάνω

συνάρτηση.

```
//after "interval" seconds the alert is hide
 [self performSelector:@selector(hideAlert) withObject:nil afterDelay:interval];
```

Μετά από χρόνο “interval” καλείται η συνάρτηση hideAlert με αποτέλεσμα η ειδοποίηση να εξαφανιστεί από την οθόνη.

4. Δημιουργία κόσμου Box2D (box2D world) για τα 2 τηλεχειριστήρια.

```
//create world
b2Vec2 gravity = b2Vec2(0.0f, -10.0f);
bool Sleep = true;
world = new b2World(gravity, Sleep);

//First cirlce sprite
CCSprite *joy=[CCSprite spriteWithFile:@"joystick2.png"];
joy.position = ccp(position.x,position.y);
[self addChild:joy];

//create circle and add it to the layer
joystic = [CCSprite spriteWithFile:@"but3.png"];
joystic.position = ccp(position.x,position.y);
 [self addChild:joystic];

//create circle body
joysticBodyDef.type = b2_dynamicBody;
joysticBodyDef.position.Set((position.x)/PTM_RATIO, (position.y)/PTM_RATIO);
joysticBodyDef.userData = joystic;
joysticBody = world->CreateBody(&joysticBodyDef);

//create circle shape
joysticShape.SetAsBox(joystic.contentSize.width/PTM_RATIO/2,
                      joystic.contentSize.height/PTM_RATIO/2);

//create shape definition and add to body
joysticShapeDef.shape = &joysticShape;
joysticShapeDef.density = 7.0f;
joysticShapeDef.friction = 2.0f;
joysticShapeDef.restitution = 0.1f;
joysticFixture = joysticBody->CreateFixture(&joysticShapeDef);
```

Στον πιο κάτω κώδικα βλέπουμε τη χρήση του Box2d πλαίσιου εργασίας(framework). Κάθε πρόγραμμα Box2D ξεκινά με τη δημιουργία ενός αντικειμένου του τύπου b2World. Το b2World είναι το κομβικό σημείο (physics hub) της φυσικής που διαχειρίζεται τη μνήμη, τα αντικείμενα και την προσομοίωση. Για την δημιουργία του Box2D κόσμου αρχικά πρέπει να ορίσουμε το διάνυσμα της βαρύτητας όπου αυτό

γίνεται με την εντολή b2Vec2 gravity = b2Vec2(0.0f,-1 0.0f). Αφού δημιουργήσαμε τον κόσμο της φυσικής μπορούμε να προσθέσουμε τα αντικείμενα μας. Αρχικά δημιουργούμε ένα αντικείμενο του τύπου CCSprite, όπου είναι ο εσωτερικός κύκλος του τηλεχειριστηρίου. Αφού δημιουργήσαμε το Sprite, για πρώτο βήμα, δημιουργούμε τον ορισμό του σώματος (body definition). Αρχικά ορίζουμε τον τύπο του σώματος, όπου θα είναι του τύπου b2_dynamicBody (joystickBodyDef.type = b2_dynamicBody) έτσι ώστε το σώμα να μπορεί να κινηθεί χειροκίνητα από τον χρήστη. Στην συνέχεια ορίζουμε την θέση του σώματος μέσω της εντολής joystickBodyDef.position.Set((positionX)/PTM_RATIO, (positionY)/PTM_RATIO) όπου η πρώτη της παράμετρος είναι η θέση στον άξονα x και δεύτερη παράμετρος η θέση στον άξονα y. Οι θέσεις που ορίζουμε στον κόσμο b2World,είναι οι θέσεις που θέλουμε να βρίσκεται το αντικείμενο διαιρώντας την πάντα με τον αριθμό 32 (σταθερά PTM_RATIO), αφού έτσι καθορίζεται από το Box2D . Στην συνέχεια συνδέουμε το σώμα με τα δεδομένα του χρήστη μέσω της εντολής joystickBodyDef.userData = joystick;. Ακολούθως δημιουργούμε ένα σώμα στον κόσμο (world) με το συγκεκριμένο ορισμό σώματος (body definition) μέσω της συνάρτησης CreateBody, όπου το σώμα αυτό έχει το όνομα “joysticbody”. Στην συνέχεια πρέπει να ορίσουμε το fixture του σώματος. Αρχικά πρέπει να ορίσουμε το σχήμα του σώματος μέσω της εντολής joystickShape.SetAsBox(joystick.contentSize.width/PTM_RATIO/2, joystick. contentSize . height/PTM_RATIO/2);. Βρίσκουμε το μήκος και το πλάτος του Sprite του τηλεχειριστηρίου, και το κουτί του πολύγονου σχήματος (shape) που δημιουργούμε πρέπει να έχει κέντρο, το κέντρο του σώματος. Στην συνέχεια συνδέουμε το σχήμα που δημιουργήσαμε με τον ορισμό του σχήματος (shape definition) μέσω της εντολής joystickShapeDef.shape = &joystickShape; Στην συνέχεια επίσης ορίζουμε την πυκνότητα (density), την τριβή (friction) και την περιστροφή (restitution) του ορισμού του σχήματος (shape definition). Τέλος, δημιουργούμε το fixture του σώματος μέσω της συνάρτησης CreateFixture δίνοντας παράμετρο τον ορισμό του σχήματος(“joysticShapeDef”).

```
//called to learn if we touch the joystic
-(boolean_t)checkTouch:(b2Vec2)locationWorld {
    if (joysticFixture->TestPoint(locationWorld))
        return true;
    else
        return false;
}
```

Με την δημιουργία των fixtures μπορούμε να δούμε εύκολα αν ο χρήστης άγγιξε το συγκεκριμένο σώμα. Αυτό γίνεται μέσω της συνάρτησης checkTouch.

Η συνάρτηση αυτή δέχεται ως είσοδο το σημείο όπου άγγιξε ο χρήστης όπου πρέπει να είναι του τύπου b2Vec2 και όχι του τύπου CGPoint,αφού πρόκειται για αντικείμενα που βρίσκονται στον κόσμο του Box2D. Γίνεται έλεγχος αν ο χρήστης άγγιξε ένα fixture και συγκεκριμένα το fixture του τηλεχειριστηρίου μέσω της συνάρτησης joystickFixture->TestPoint(locationWorld), όπου locationWorld είναι το σημείο που άγγιξε ο χρήστης. Η συνάρτηση αυτή επιστρέφει true όταν το σημείο αυτό βρίσκεται στο συγκεκριμένο fixture, αλλιώς επιστρέφει false.

```
-(void)changeBody: (int)a :(int)b
{
    joystickBody->SetTransform(b2Vec2(a,b), 0);
}
```

Στο πρώτο κομμάτι κώδικα στο Παράρτημα Β σημείο 4, ορίζαμε την θέση που θα βρίσκεται το σώμα. Και στα δύο τηλεχειριστήρια αρχικά βρίσκεται στο κέντρο του κύκλου. Στο δεξιό τηλεχειριστήριο όμως όταν το κινήσουμε και το αφήσουμε πάντα η τελική το θέση θα είναι το κέντρο. Έτσι δεν είναι απαραίτητο να αλλάξουμε την θέση του σώματος. Στο αριστερό τηλεχειριστήριο όμως όταν το αφήσουμε μπορεί να καταλήξει σε κάποια άλλη θέση εκτός του κέντρου. Έτσι για να μπορούμε να το αγγίξουμε την επόμενη φορά και να κινηθεί θα πρέπει να αλλάξουμε την θέση του σώματος του. Αν για παράδειγμα, όταν το αφήσουμε και παραμείνει τοποθετημένο στην θέση((size.width/2 -282)/PTM_RATIO, (size.height/2 -128)/PTM_RATIO)) τότε και η θέση του σώματος του θα πρέπει να ορισθεί με την τιμή αυτή.

5. Εύρεση της τιμής που πρέπει να σταλεί ανάλογα με την θέση του δεξιού τηλεχειριστηρίου.

```
//find the sending values depending on the position of the sprite
-(int) convertToCircle: (float)current :(int)axis
{
   int data=0;
   float pointx;
   //axis y
   if (axis==AXIS_Y){
      pointx=[self getInitialPosition].y-current;
      data=128+(int)pointx;}
   else if (axis==AXIS_X)
   {
      //axis x
      pointx=[self getInitialPosition].x-current;
      data=128-(int)pointx;
   }
   return data;
}
```

Με την συνάρτηση convertToCircle καθορίζουμε τις τιμές που πρέπει να σταλούν στο συγκεκριμένο κανάλι ανάλογα με την θέση που βρίσκεται το τηλεχειριστήριο. Στον άξονα x πρέπει να στέλλονται οι τιμές 0 – 255, αυξάνοντας την τιμή κατά 1 κάθε φορά, από αριστερά προς τα δεξιά. Στον άξονα y πρέπει να στέλλονται επίσης οι τιμές από 0 – 255 αυξάνοντας και πάλι την τιμή κατά 1 κάθε φορά από πάνω προς τα κάτω. Η συνάρτηση αυτή δέχεται ως είσοδο την θέση στον ένα άξονα που βρίσκεται το τηλεχειριστήριο, και σε ποιον άξονα θέλουμε να κάνουμε τον υπολογισμό της απεσταλμένης τιμής. Για να γίνει εύκολα αντιληπτός ο πιο πάνω κώδικας δίνουμε το πιο κάτω παράδειγμα:

Υποθέτουμε ότι η αρχική θέση του τηλεχειριστηρίου είναι η (250,250)

Υποθέτουμε ότι έγινε κίνηση 5 θέσεις προς τα πάνω και 5 θέσεις προς τα αριστερά. Δηλαδή το τηλεχειριστήριο βρίσκεται στην θέση (245,255). Η συνάρτηση αυτή θα καλεστεί δυο φορές για τον υπολογισμό των τιμών των καναλιών και στους δύο άξονες.

Στην πρώτη περίπτωση (για τον άξονα x) έχουμε:

pointx==[self getInitialPosition].x-current = 250-245=5

data=128-(int)pointx = 128 -5= 123

Στην δεύτερη περίπτωση (για τον άξονα y) έχουμε:

pointx=[self getInitialPosition].y-current =250-255=-5

data=128+(int)pointx = 128 -5= 123

```
-(void)updateValues
{
   [communicate
updateValues:sendingValue1 :channelxValue:sendingValue2 :channelyValue];
}
```

Εδώ γίνεται η ανανέωση των αποσταλμένων τιμών στα συγκεκριμένα κανάλια.

6. Εύρεση του σημείου που πρέπει να ορισθούν τα δύο τηλεχειριστηρίων σε περίπτωση που το άγγιγμα είναι εκτός των ορίων του κύκλου.

```
//return the nearest position on the circle
-(CGPoint) getNearestPoint:(CGPoint) p
{
   //initialize nearest point random
   float minx=  [self getInitialPosition].x + DIAMETER* cos(0);
   float  miny=  [self getInitialPosition].y + DIAMETER * sin(0);

   float  min=[self euclidian:minx:miny:p.x:p.y];
   float euclidean;
   float x;
   float y;

   //search for the nearest point at the diameter of the circle
   for (int i=0;i<=DEGREES;i++)
   {
      x=  [self getInitialPosition].x + DIAMETER * cos(i);
      y = [self getInitialPosition].y + DIAMETER * sin(i);
      euclidean=[self euclidian:x:y:p.x:p.y];
      if (min>euclidean)
      {
         min=euclidean;
         minx=x;
         miny=y;
      }

   }
   //return the nearest point
   return CGPointMake(minx, miny);
}
```

Σε περίπτωση που ο χρήστης χρησιμοποιεί κάποιο τηλεχειριστήριο τότε υπάρχει πιθανότητα να συνεχίσει την κίνηση αυτή εκτός των ορίων του κύκλου. Στην περίπτωση αυτή, θα καλεστεί η συνάρτηση getNearestPoint, με στόχο να τοποθετήσει το τηλεχειριστήριο (εσωτερικό Sprite) στο κοντινότερο σημείο που βρίσκεται στην περιφέρεια του κύκλου, από το σημείο το οποίο άγγιξε ο χρήστης.

Η συνάρτηση αυτή παίρνει σαν είσοδο ένα σημείο (τύπου CGPoint). Τα σημεία που βρίσκονται πάνω στον κύκλο μπορούμε να τα βρούμε μέσω των εξισώσεων:

$x = a + r\ \cos t$ και

$y = b + r\ \sin t$

όπου:

(x,y): συντεταγμένη που βρίσκεται στην περιφέρεια του κύκλου

(a,b): κέντρο του κύκλου

r: ακτίνα του κύκλου

cost: συνημίτονο της γωνιάς t

Αρχικά ορίζουμε το κοντινότερο σημείο με τυχαίες συντεταγμένες, όπου το σημείο αυτό βρίσκεται στην περιφέρεια του κύκλου, με γωνία 0. Η ακτίνα του κύκλου έχει τιμή 128. Με την επανάληψη for (i=0;i<= DEGREES; i++) όπου η σταθερά DEGREES είναι 360, περνούμε όλα τα υποψήφια κοντινότερα σημεία που βρίσκονται στην περιφέρεια του κύκλου. Έχοντας το σημείο όπου άγγιξε ο χρήστης κάθε φορά βρίσκουμε την ευκλείδεια απόσταση μεταξύ του σημείου που άγγιξε ο χρήστης και του υποψήφιου κοντινότερου σημείου που βρίσκεται πάνω στον κύκλο. Η ευκλείδεια απόσταση βρίσκεται από την εξίσωση:

$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}.$$

όπου:

qi: τιμή στην διάσταση i του σημείου q

pi: τιμή στην διάσταση I του σημείου p

n: αριθμός διαστάσεων

Στο τέλος, επιστρέφουμε το σημείο που βρίσκεται πιο κοντά από το σημείο που άγγιξε ο χρήστης.

7. Βασικά κομμάτια κώδικα διαχείρισης του ήχου

```
- (id) init{
    self = [super init];
    if (self != nil) {

        //allocate memory for the alert
        alertVolume=[[alertHide alloc]message:@"Volume\n\n"];

        //file that plays is beep.mp3
        url = [NSURL fileURLWithPath:[NSString
stringWithFormat:@"%@/beep.mp3", [[NSBundle mainBundle] resourcePath]]];

        //allocate memory for the AVAudioPlayer
        audio = [[AVAudioPlayer alloc] initWithContentsOfURL:url error:&error];

        //play the sound once
        [audio setNumberOfLoops:0];

        //full volume
        [audio setVolume:1];
    }
    return self;
}
```

Πιο πάνω βλέπουμε το κομμάτι του κώδικα για τον ήχο που παράγεται στην εφαρμογή. Γίνεται χρήση της κλάσης NSURL έτσι ώστε να αναφερθούμε στο αρχείο του ήχου που θέλουμε να χρησιμοποιήσουμε. Δίνουμε το όνομα του αρχείου του ήχου, όπως και επίσης με την [[NSBundle mainBundle] resourcePath]] παίρνουμε την τοποθεσία του αρχείου, αφού η εντολή αυτή επιστρέφει την πλήρη διαδρομή του υποκαταλόγου που περιέχει τους πόρους της εφαρμογής μας. Στην συνέχεια δημιουργούμε ένα αντικείμενο της κλάσης AVAudioPlayer δίνοντας τον ήχο που θέλουμε να παράξει. Στην συνέχεια με την εντολή audio.numberOfLoops = -1, ορίζουμε ότι ο ήχος θέλουμε να παράγεται μόνο μια φορά και στην συνέχεια με την εντολή audio.volume=1 ορίζουμε ότι ο ήχος θα παράγεται στο μέγιστο.

Επίσης για την παραγωγή του ήχου χρησιμοποιούμε την πιο κάτω εντολή

```
[audio play];
```

8. Κυριότερο κομμάτι κώδικα – η **επικοινωνία** συσκευής - ελικοπτέρου

```
//STATIC VARIABLES
static Communication *staticCommunication = nil;
static int onceInitialization=0;

// calling [Communication sharedCommuniation] will return the singleton object
+(Communication *)sharedCommunication
{
   onceInitialization++;
   // ensure that singleton is thread safe
   @synchronized(self) {
      if(!staticCommunication)
      {
         staticCommunication = [[Communication alloc] init];

         // initialize the sending values just one time
         if (onceInitialization==1)
            [staticCommunication initialize];
      }
      return staticCommunication;
   }}
```

Η συνάρτηση sharedCommunication αυτή επιστρέφει ένα στατικό αντικείμενο τύπου Communication.

Καλώντας [Communication sharedCommunication] παίρνουμε αυτό το στατικό αντικείμενο. Η κλάση αυτή περιέχει ένα αντικείμενο της δομής (struct) RCOIPv1CmdSetAnalogChannels. Το αντικείμενο μας πρέπει να είναι στατικό, επειδή χρησιμοποιείται από διαφορετικές κλάσεις όπου η κάθε μια ανανεώνει και διαφορετικό κομμάτι του πίνακα (κανάλι) που πρέπει να αποστείλουμε. Επίσης χρειαζόμαστε την μεταβλητή static int onceInitialization έτσι ώστε οι αρχικοποιήσεις του πίνακα με τα κανάλια να αρχικοποιούνται μόνο μια φορά.

```
- (id) init{
    if (self != nil) {
        //create the udp socket
        udpSocket=[[AsyncUdpSocket alloc] initWithDelegate:self];
        //check for errors while binding
        error=nil;
        if (![udpSocket bindToPort:0 error:&error])
            NSLog(@"Error while binding");
        //host
        host = @"192.168.10.4";
        if ([host length] == 0)
            NSLog(@"error");
        //port - default RCOIP receiver UDP port number is 9048
        portText = @"8888";
        port = [portText intValue];
        if (port <= 0 || port > 65535)
            NSLog(@"Wrong port");
    }
    return self;
}
```

Στην συνάρτηση init βλέπουμε την δημιουργία του udp socket του τύπου AsynUdpSocket. Επίσης ορίζουμε ποιος θα είναι ο παραλήπτης του πακέτου “host” , και πιο θα είναι το port επικοινωνίας, “port”.

```
//update the 2 channels (2 table positions) for the joystic
-(void)updateValues: (int)channel1 :(int)val1 :(int)channel2 :(int)val2
{
    NSLog(@"update channel %d with value %d",channel1,val1);
    NSLog(@"update channel %d with value %d",channel2,val2);
    messageStructure.channels[channel1]=val1;
    messageStructure.channels[channel2]=val2;
}
```

Εδώ γίνεται η αλλαγή των τιμών της δομής του τύπου RCOIPv1SetAnalogChannels.

```
- (void)dataSend{
    //allocate memory for the send data for udpSocket
    data = [NSMutableData data];

    unsigned char byte;

    byte = (NSInteger)messageStructure.version;
    //append version at the first position of the mutable data as the message structure
requires
    [data appendBytes:&byte length:1];

    //append channels
    for (int i=0;i<CHANNELS; i++)
    {
        byte=(NSInteger)messageStructure.channels[i];
        [data appendBytes:&byte length:1];
    }
    //send data to the arduino
    [udpSocket sendData: data toHost:host port:port withTimeout:-1 tag:tag];
    //increase by one the tag every time you send the packet
    tag++;
}
```

Η συνάρτηση αυτή είναι υπεύθυνη για την δημιουργία του πίνακα του τύπου NSMutableData με τις τιμές των καναλιών. Αρχικά κάνουμε επισύναψη (append) την έκδοση του πρωτοκόλλου και στην συνέχεια τα κανάλια για τα 2 τηλεχειριστήρια και για το κουμπί. Η αποστολή γίνεται μέσω της εντολής [udpSocket sendData: data toHost:host port:port withTimeout:-1 tag:tag]; η οποία καθορίζει ότι θα αποσταλεί ο πίνακας “data”, σε ένα συγκεκριμένο “host”, σε ένα συγκεκριμένο “port” και το πακέτο θα έχει την ετικέτα “tag”.

9. Δημιουργία πεδίου εισόδου (text field).

```
//ip field
ipTextField = [[UITextField alloc] initWithFrame:CGRectMake(260, 500, 280.0f,
40.0f)];
ipTextField.placeholder = @"IP address";
ipTextField.backgroundColor = [UIColor whiteColor];
ipTextField.textColor = [UIColor blackColor];
ipTextField.font = [UIFont systemFontOfSize:25.0f];
ipTextField.borderStyle = UITextBorderStyleRoundedRect;
ipTextField.textAlignment = NSTextAlignmentLeft;
ipTextField.transform=CGAffineTransformMakeRotation(M_PI/2);
```

Στο πιο πάνω κώδικα βλέπουμε την δημιουργία του πεδίου εισόδου του τύπου UITextField. Η θέση του είναι η (260, 500), και το μέγεθος του έχει μήκος 280, και πλάτος 40 . Το κείμενο που έχει γραμμένο αρχικά είναι "IP address", το χρώμα του είναι άσπρο και το πλαίσιο του μαύρο. Επίσης το μέγεθος των γραμμάτων είναι 25, η εισαγωγή γίνεται από τα αριστερά προς τα δεξιά και μέσω της εντολής CGAffineTransformMakeRotation(M_PI/2) καθορίζουμε ότι θα είναι στοιχισμένο οριζόντια.

```
//check ip address
-(boolean_t)checkIp:(NSString *)ipString
{
   NSArray *field = [ipString componentsSeparatedByString:@"."];
   if ([field count]!=4)
      return false;

   int num1 = [field[0] intValue];
   int num2 = [field[1] intValue];
   int num3 = [field[2] intValue];
   int num4 = [field[3] intValue];

   if (num1>=0 && num1<=255 && num2>=0 && num2<=255 && num3>=0 &&
num3<=255 && num4>=0 && num4<=255)
      return true;
   return false;}
```

Η συνάρτηση checkIp ελέγχει ότι δώσαμε ορθή IP διεύθυνση (κάθε κομμάτι της από 0 - 255)

```
//check port number
-(boolean_t)checkPort:(NSString *)portString
{
   int portNum = [portString intValue];
   if (portNum <= 0 || portNum > 65535)
      return false;
   return true;
}
```

Η πιο πάνω συνάρτηση ελέγχει ότι δώσαμε σωστό port (από 0 μέχρι 65535)

10. Κομμάτια συναρτήσεων που διαχειρίζονται τα αγγίγματα

Όταν ο χρήσης αγγίζει την οθόνη καλούνται μια σειρά από συναρτήσεις.

Μόλις αγγίξει την οθόνη καλείται η ccTouchesBegan, όταν δώσει κίνηση στο άγγιγμα του καλείται η ccTouchesMoved, και όταν απομακρύνει το χέρι του από την οθόνη καλείται η ccTouchesEnded. Απαραίτητη προϋπόθεση για την λειτουργία των συναρτήσεων αυτών είναι να ορίσουμε self.isTouchEnabled=YES; στο συγκεκριμένο delegate, και στην συνάρτηση που είναι υπεύθυνη για την εκκίνηση της εφαρμογής [window setMultipleTouchEnabled:Yes]; όπου “window” είναι αντικείμενο του τύπου UIWindow.

```
//called when touch began
- (void)ccTouchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    NSArray *touchTable = [touches allObjects];

    UITouch *firstTouch;
    UITouch *secondTouch = nil;

    CGPoint firstPoint;
    CGPoint secondPoint;

    if ([touchTable count] ==1)
    {
        //UITouch object for a touch
        firstTouch = [touchTable objectAtIndex:0];


        firstPoint = [firstTouch locationInView:[firstTouch view]];
        firstPoint = [[CCDirector sharedDirector] convertToGL:firstPoint];
    }

    if ([touchTable count]==2)
    {
        //UITouch object for first touch
        firstTouch= [touchTable objectAtIndex:0];

        //UITouch object for second touch
        secondTouch = [touchTable objectAtIndex:1];

        // Convert each UITouch object to a CGPoint, which has x/y coordinates we
can actually use

        firstPoint = [firstTouch locationInView:[firstTouch view]];
        firstPoint = [[CCDirector sharedDirector] convertToGL:firstPoint];

        secondPoint = [secondTouch locationInView:[secondTouch view]];
        secondPoint = [[CCDirector sharedDirector] convertToGL:secondPoint];

    }
```

```
//if we touch left joystic set flag touchRight true
    if (([leftJoystic checkTouch:b2Vec2(firstPoint.x/PTM_RATIO,
firstPoint.y/PTM_RATIO)] || [leftJoystic
checkTouch:b2Vec2(secondPoint.x/PTM_RATIO, secondPoint.y/PTM_RATIO)])
&&touchLeft==false)
        touchLeft=true;

    //if we touch right joystic set the flag touchLeft true
    if (([rightJoystic checkTouch:b2Vec2(firstPoint.x/PTM_RATIO,
firstPoint.y/PTM_RATIO)] && acceleratorOn==false && touchRight==false)
||([rightJoystic checkTouch:b2Vec2(secondPoint.x/PTM_RATIO,
secondPoint.y/PTM_RATIO)] && acceleratorOn==false &&touchRight==false))
        touchRight=true;
}
```

Η συνάρτηση ccTouchesBegan καλείται μόλις ο χρήστης αγγίξει την οθόνη χωρίς να κάνει καμιά κίνηση στην οθόνη.

Μέσω του πιο κάτω κώδικα

```
firstTouch = [touchTable objectAtIndex:0];
firstPoint = [firstTouch locationInView:[firstTouch view]];
firstPoint = [[CCDirector sharedDirector] convertToGL:firstPoint];
```

μπορούμε να βρούμε την θέση που άγγιξε ο χρήστης. Το objectAtIndex:0 καθορίζει το πρώτο άγγιγμα και το objectAtIndex:1 καθορίζει το δεύτερο άγγιγμα. Αναλόγως μπαίνουμε σε ένα από τα δύο if, στο πρώτο αν έχει ένα άγγιγμα και στο δεύτερο αν είχε 2 αγγίγματα Στην συνέχεια καλώντας την συνάρτηση checkTouch μπορούμε να βρούμε αν άγγιξε ο χρήστης κάποιο από τα 2 τηλεχειριστήρια. Αν άγγιξε το αριστερό τηλεχειριστήριο θέτουμε, την σημαία (boolen flag) touchleft σε true. Αν άγγιξε το αριστερό τηλεχειριστήριο, θέτουμε την σημαία (boolen flag) touchRight σε true.

Ποιο κάτω βλέπουμε το κυριότερο κομμάτι κώδικα της συνάρτησης CCTouchMoved.

```
firstTouch = [touchTable objectAtIndex:0];
firstPoint = [firstTouch locationInView:[firstTouch view]];
firstPoint = [[CCDirector sharedDirector] convertToGL:firstPoint];


float dist=[leftJoystic euclidian:firstPoint.x :firstPoint.y:[leftJoystic
getCurrentPosition].x:[leftJoystic getCurrentPosition].y];
float dist2=[rightJoystic euclidian:firstPoint.x :firstPoint.y:[rightJoystic
getCurrentPosition].x:[rightJoystic getCurrentPosition].y];
```

```
if (dist<dist2){
        if (touchLeft==true)
          {
           float distance=[leftJoystic euclidianCenter:firstPoint];

           //touch in bounds else out of bounds
           if (distance<=DIAMETER)
              //set the joystic position at the position that user touched
              [leftJoystic setPosition:firstPoint];
           else
              //set the position of joystic on the nearestn point on the perimeter of the
circle
              [leftJoystic setPosition:[leftJoystic getNearestPoint:firstPoint]];

           //call the touchMoved for left joystic
           [leftJoystic touchMoved];
}
}
```

Για παράδειγμα αν άγγιξε το αριστερό τηλεχειριστήριο, αρχικά βρίσκουμε την ευκλείδεια απόσταση από το σημείο που άγγιξε ο χρήστης από τον κέντρο του κύκλου. Αν η απόσταση αυτή είναι μικρότερη από 128 (ακτίνα του κύκλου) τότε ο χρήστης άγγιξε μέσα στον κύκλο αλλιώς άγγιξε έξω από τον κύκλο. Στην πρώτη περίπτωση θα καλέσουμε την συνάρτηση [leftJoystic setPosition:location] η οποία θα ορίσει την θέση του τηλεχειριστηρίου ως την θέση που άγγιξε ο χρήστης. Στην δεύτερη περίπτωση θα βρούμε την θέση στην περιφέρεια του κύκλου που είναι πιο κοντά στο σημείο που άγγιξε ο χρήσης μέσω της συνάρτησης getNearestPoint (επεξήγηση λεπτομερώς στο Παράρτημα Β σημείο 6) και στην συνέχεια θα θέσουμε αυτό το σημείο ως την θέση του τηλεχειριστηρίου. Τέλος καλείται η συνάρτηση touchMoved, η οποία θέτει την τελική θέση του τηλεχειριστηρίου και ανανεώνει τις καινούργιες τιμές που πρέπει να σταλούν στο ελικόπτερο.

.

Παράρτημα Γ

Στο παράρτημα αυτό θα δώσουμε τα βήματα που χρειάζονται για να τρέξει η εφαρμογή στον προσομοιωτή iOS, πάνω σε υπολογιστή της Apple.

Σε περίπτωση που θέλουμε να τρέξουμε την εφαρμογή σε υπολογιστή με μη λειτουργικό iOS απαιτείται η χρήση εικονικής μηχανής.

1. Ανοίγουμε τον MAC υπολογιστή μας και εισάγουμε το CD.
2. Ανοίγουμε τον φάκελο "Ατομική διπλωματική Εργασία"→ Code → PMJOYSTIC και στην συνέχεια ανοίγουμε το αρχείο PMJOYSTIC.xcodeproj μέσω του προγράμματος XCode έκδοση 4.6.2, και έτσι ξεκινά η εφαρμογή XCode.
3. Για επιβεβαίωση ότι διαθέτετε έκδοση που μπορεί να τρέξει την εφαρμογή πηγαίνετε στο Xcode → About Xcode και στο παράθυρο αυτό μπορείτε να διακρίνετε την έκδοση.
4. Στο πάνω μέρος του προγράμματος στο Scheme → iPad 6.1 Simulator
5. Στο Project Navigator → PMJOYSTIC, Targets → PMJOYSTIC
   - **5.1.** BuildSettings → Architectures →BaseSDK → Latest iOS (iOS 6.1)
   - **5.2.** BuildSettings → BuildOptions → Compiler for C/C++/Objective C → Default compiler (Apple compiler LLVM 4.2)
6. Για κτίσιμο και τρέξιμο της εφαρμογής στον iPad simulator πατάμε το κουμπί 

7. Για τερματισμό της εφαρμογής πατάμε το κουμπί 

Παράρτημα Δ

Στο παρόν παράρτημα δίνεται ο κώδικας της εφαρμογής.

Background.h

```
/*
 Background.h
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "cocos2d.h"

//INTERFACE Background INHERITS FROM CCLayer
@interface Background : CCLayer

//INTERFACE VARIABLES
{
    //size of the screen
    CGSize size;
    //sprite for the background
    CCSprite *background2;
}

//PROPERTIES

@property (nonatomic,assign) CGSize size;
@property (nonatomic,retain) CCSprite *background;

//FUNCTIONS

//constructor - returns a background object
- (Background *) init: (NSString *)backgroundImage;

//memory deallocation
-(void) dealloc;

@end
```

Background.m

```
/*
 Background.m
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Background.h"
```

```
//IMPLEMENTATION
@implementation Background

//SYNTHESIZE VARIABLES
@synthesize size;
@synthesize background;

//constructor - returns a background object
- (Background *) init: (NSString *)backgroundImage{
    self = [super init];
    if (self != nil) {

        //initialize size of the screen
        size = [[CCDirector sharedDirector] winSize];

        //set the image black.jpg to the sprite
        background = [CCSprite spriteWithFile:backgroundImage];

        //define background (sprite) position
        background.position = ccp(size.width/2, size.height/2);

        //add the background to the layer
        [self addChild: background];
    }
    return self;
}

//memory deallocation
-(void) dealloc
{
    [background release];
    [super dealloc];
}

@end
```

Accelerate.h

```
/*
 Accelerate.h
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "cocos2d.h"
#import "alertHide.h"
#import <AVFoundation/AVFoundation.h>
#import "Joystic.h"

//CONSTANT VARIABLES
//diameter of the joystic
```

```
#define DIAMETER 128
//helps to find the sprite position
#define SPRITE_POSITION 256
//ipad movement enable
#define IPAD_ON 1

//INTERFACE Accelerate INHERITS FROM CCLayer
@interface Accelerate : CCLayer

//INTERFACE VARIABLES
{
   //size for the screen
   CGSize size;
   //alert of the accelerator
   alertHide *alertAccelerate;
   //acceleration for axis x
   float x;
   //acceleration for axis y
   float y;
   //position of the sprite created from the movement
   CGPoint spriteAcceleratorPosition;
   //initial position of joystic - axis X
   float posX;
   //initial position of joystic - axis Y
   float posY;
   //accelerometer object
   UIAccelerometer *accelerometer;
}

//PROPERTIES

@property (nonatomic,assign) CGSize size;
@property (nonatomic, retain) alertHide *alertAccelerate;
@property (nonatomic, assign) float userAccelerationX;
@property (nonatomic, assign) float userAccelerationY;
@property (nonatomic, assign) CGPoint spriteAcceleratorPosition;
@property (nonatomic, assign)float posX;
@property (nonatomic, assign) float posY;
@property (nonatomic,retain) UIAccelerometer *accelerometer;

//FUNCTIONS

//constructor - returns an object of Accelerate
-(id)init:(CGPoint)position;
//called when move iPad - setter for the position of the sprite
-(void)setSpriteAcceleratorPosition:(float)xVal :(float)yVal;
//called from the accelerator selector
-(void)pushAccelerator:(int)onOff;
//updates the accelerometer values
-(void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration
*)acceleration;
//returns the new position of the sprite that created from the movement - getter for the position
of the sprite
-(CGPoint)getSpriteAcceleratorPosition;
```

```
//memory deallocation
-(void)dealloc;
```

Accelerate.mm

```
/*
 Accelerate.mm
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Accelerate.h"

//IMPLEMENTATION
@implementation Accelerate

//SYNTHESIZE VARIABLES
@synthesize alertAccelerate;
@synthesize userAccelerationX;
@synthesize userAccelerationY;
@synthesize spriteAcceleratorPosition;
@synthesize size;
@synthesize posX;
@synthesize posY;
@synthesize accelerometer;

//constructor - returns an Accelerate object
- (id) init:(CGPoint)position{
    self = [super init];
    if (self != nil) {

        //get the dimensions of the screen
        size=[CCDirector sharedDirector].winSize;

        //alert - appears when we push the accelerator button
        alertAccelerate=[[alertHide alloc]message:@"IPad movement\n\n"];

        posX=position.x;
        posY=position.y;

        accelerometer = [UIAccelerometer sharedAccelerometer];

        //accelerometer update interval is 0.001
        accelerometer.updateInterval = .001;

        accelerometer.delegate = self;

    }
    return self;

}
```

```
//called from the accelerator selector (when we push the accelerator button)
-(void)pushAccelerator:(int)onOff{

    //show the alert that accelerator is on
    if (onOff==IPAD_ON)
        [alertAccelerate setText:@"IPad movement is on!"];
    else
        //else show the alert the accelerator is off
        [alertAccelerate setText:@"IPad movement is off"];

    //show the alert
    NSLog(@"Show alert for the accelerator");
    [alertAccelerate alertShow];

    //update the interval from the begining
    [alertAccelerate alertShowWithInterval];
}

//updates the accelerometer values
- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration
*)acceleration
{
    [self setSpriteAcceleratorPosition:acceleration.x :acceleration.y];
}

//this function updates the right joystic sprite position
-(void)setSpriteAcceleratorPosition:(float)xVal :(float)yVal
{
    //define the position of the sprite
    spriteAcceleratorPosition.x=posX-yVal*SPRITE_POSITION;
    spriteAcceleratorPosition.y=posY+xVal*SPRITE_POSITION;
}

//get the new position of the sprite by the ipad movement
-(CGPoint)getSpriteAcceleratorPosition
{
    return spriteAcceleratorPosition;
}

//memory deallocation
-(void) dealloc
{
    [alertAccelerate release];
    [accelerometer release];
    [super dealloc];
}

@end
```

Alert.h

```
/*
 Alert.h
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "cocos2d.h"

//INTERFACE Alert INHERITS FROM UIViewController
@interface Alert:UIViewController <UIAlertViewDelegate>

//INTERFACE VARIABLES
{
    //alert to show messages to the screen for the user
    UIAlertView *alert;
}

//PROPERTIES

@property (readwrite, retain) UIAlertView *alert;
@property(readwrite, retain) UILabel *textLabel;

//FUNCTIONS

//constructor of the alert
-(void) initial:(NSString*)text;
//set the message of the alert
-(void)setText:(NSString *)text;
//called to show the alert to the screen
-(void)alertShow;

//memory deallocation
-(void)dealloc;

@end
```

Alert.m

```
/*
 Alert.m
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Alert.h"

//IMPLEMENTATION
```

```
@implementation Alert

//SYNTHESIZE VARIABLES
@synthesize alert;

//constructor of alert
- (void) initial:(NSString*)text{

    //allocate memory for UIAlertView
    alert = [[UIAlertView alloc] init];

    //set the title of the alert
    [alert setTitle:text];
}

//set message
-(void)setText:(NSString *)text
{
    //set the string to the message of the alert
    [alert setMessage:text];
}

//called to show the alert to the screen
-(void)alertShow
{
    //show the alert
    [alert show];
}

//memory deallocation
-(void)dealloc
{
    [alert release];
    [super dealloc];
}

@end
```

alertChoice.h

```
/*
 alertChoice.h
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Alert.h"

//INTERFACE alertChoice INHERITS FROM Alert
@interface alertChoice : Alert

//FUNCTIONS
```

```
//Constructor - return an alertChoice object
-(alertChoice*) message:(NSString*)text;
//called when we push a button of the alert
-(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex;
//return yes string
- (NSString *) yesButtonTitle;
//returns no string
- (NSString *) noButtonTitle;
// exit the application
-(int)exit;

//memory deallocation
-(void)dealloc;

@end
```

alertChoice.m

```
/*
 alertChoice.m
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "alertChoice.h"
#import "Exit.h"

//IMPLEMENTATION
@implementation alertChoice

//constructor of the alertChoice - return an object of the alertChoice
- (alertChoice*) message:(NSString*)text{
    self = [super init];
    if (self != nil)
    {
        //initialize the title of the alert
        [super initial:text];

        // Add Buttons yes and no
        [alert setDelegate:self];
        [alert addButtonWithTitle:@"Yes"];
        [alert addButtonWithTitle:@"No"];
    }
    return self;
}

//called when we push an alert button
- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex{

    //returns the name of the button the we pushed of the specific index
    NSString * buttonPressed = [alertView buttonTitleAtIndex:buttonIndex];
```

```
    // if we push yes button the application is terminated
    if([buttonPressed isEqualToString: [self yesButtonTitle]])
    {
        NSLog(@"Application exit");
        [self exit];
    }
    // else do nothing
    else if ([buttonPressed isEqualToString:[self noButtonTitle]])
        NSLog(@"Application not exit");
}

// exit the application
-(int)exit
{
    [[CCDirector sharedDirector]end];
    exit(0);
}

//return yes string
- (NSString *) yesButtonTitle{
    return @"Yes";
}

//return no string
- (NSString *) noButtonTitle{
    return @"No";
}

//memory deallocation
-(void)dealloc
{
    [super dealloc];
}

@end
```

alertHide.h

```
/*
 alertHide.h
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Alert.h"

//INTERFACE alertHide INHERITS FROM Alert
@interface alertHide : Alert

//INTERFACE VARIABLES
{
```

```
    //the time that the alert will be on the screen
    int iterval;
}

//PROPERTIES

@property (nonatomic, assign)int interval;

//FUNCTIONS

//constructor - return an alertHide object
- (alertHide*) message:(NSString*)text;
//hides the alert from the screen
- (void) hideAlert;
//updates the interval of the alert
-(void)alertShowWithInterval;

//memory deallocation
-(void) dealloc;

@end
```

alertHide.m

```
/*
 alertHide.m
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
 */

//IMPORT LIBRARIES
#import "alertHide.h"

//IMPLEMENTATION
@implementation alertHide

//SYNTHESIZE VARIABLES
@synthesize interval;

//constructor - return an alertHide object
- (alertHide*) message:(NSString*)text{
    self = [super init];
    if (self != nil)
    {
        [super initial:text];

        //set the interval for 2 seconds
        interval=2;
    }
    return self;
}

//hides the alert from the screen
```

```
- (void) hideAlert {
        [alert dismissWithClickedButtonIndex:0 animated:YES];
}

//updates the interval of the alert
-(void)alertShowWithInterval
{
    //after "interval" seconds the alert is hide
    [self performSelector:@selector(hideAlert) withObject:nil afterDelay:interval];
}

//memory deallocation
-(void)dealloc
{
    [super dealloc];
}
@end
```

appDelegate.h

```
/*
 appDelegate.h
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "appDelegate.h"
#import "cocos2d.h"
#import "mainScreen.h"

//INTERFACE appDelegate INHERITS FROM CCLayerUIApplicationDelegate
@interface appDelegate : NSObject <UIApplicationDelegate>

//INTERFACE VARIABLES
{
    //window of the application
        UIWindow *window;
    //initial scene
    viewScene *scene;
}

//PROPERTIES

@property (nonatomic, retain) UIWindow *window;
@property (nonatomic, retain) viewScene *scene;

//FUNCTIONS

//called when applications launches
- (void) applicationDidFinishLaunching:(UIApplication*)application;
//called when the application is about to move from active to inactive state
- (void)applicationWillResignActive:(UIApplication *)application;
```

```
//restarts any tasks that were paused (or not yet started) while the application was inactive
- (void)applicationDidBecomeActive:(UIApplication *)application;
//tells the delegate when the application receives a memory warning from the system
- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application;
//called when the application is about to terminate
- (void)applicationWillTerminate:(UIApplication *)application;

//memory deallocation
- (void)dealloc;

@end
```

appDelegate.mm

```
/*
 appDelegate.mm
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "appDelegate.h"

//IMPLEMENTATION
@implementation appDelegate

//SYNTHESIZE VARIABLES
@synthesize window;
@synthesize scene;

- (void) applicationDidFinishLaunching:(UIApplication*)application
{
        window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];

        [window setUserInteractionEnabled:YES];

    //set multitouch enable
        [window setMultipleTouchEnabled:YES];

        if( ! [CCDirector setDirectorType:CCDirectorTypeDisplayLink] )
          [CCDirector setDirectorType:CCDirectorTypeDefault];

    //pixel format
        [[CCDirector sharedDirector] setPixelFormat:kPixelFormatRGBA8888];

        [CCTexture2D setDefaultAlphaPixelFormat:kTexture2DPixelFormat_RGBA8888];

    //orientation: landscape to left
        [[CCDirector sharedDirector] setDeviceOrientation:CCDeviceOrientationLandscapeLeft];

        [[CCDirector sharedDirector] setAnimationInterval:1.0/60];

        [[CCDirector sharedDirector] setDisplayFPS:NO];
```

```
        [[CCDirector sharedDirector] attachInView:window];

        [window makeKeyAndVisible];

    //application will launch with viewScene object

    scene = [[viewScene alloc]init];

        [[CCDirector sharedDirector] runWithScene: scene];
}

//called when the application is about to move from active to inactive state
- (void)applicationWillResignActive:(UIApplication *)application {
        [[CCDirector sharedDirector] pause];
}

//restarts any tasks that were paused (or not yet started) while the application was inactive
- (void)applicationDidBecomeActive:(UIApplication *)application {
        [[CCDirector sharedDirector] resume];
}

//tells the delegate when the application receives a memory warning from the system
- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application {
        [[CCTextureCache sharedTextureCache] removeUnusedTextures];
}

//called when the application is about to terminate
- (void)applicationWillTerminate:(UIApplication *)application {
        [[CCDirector sharedDirector] end];
}

//memory deallocation
- (void)dealloc {
        [[CCDirector sharedDirector] release];
        [window release];
        [super dealloc];
}

@end
```

Exit.h

```
/*
 Exit.h
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "cocos2d.h"
#import "alertChoice.h"
```

```
//INTERFACE Exit INHERITS FROM CCLayer
@interface Exit : CCLayer

//INTERFACE VARIABLES
{
   //alert that asks to exit the programm
   alertChoice *alertExit;
}

//PROPERTIES

@property (nonatomic,retain) alertChoice *alertExit;

//FUNCTIONS

//constructor - returns an Exit object
-(id)init;
//called when we push exit button (from the viewLayer)
-(void)pushExit;

//memory deallocation
-(void) dealloc;

@end
```

Exit.mm

```
/*
 Exit.mm
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Exit.h"

//IMPLEMENTATION
@implementation Exit

//SYNTHESIZE VARIABLES
@synthesize alertExit;

//constructor - returns an Exit object
- (id) init{
   self = [super init];
   if (self != nil) {

      // allocate an alertHide object
      alertExit =[[alertChoice alloc]message:@"\n"];
   }
   return self;
}
```

```
//called when we push exit button (from the viewLayer)
-(void)pushExit
{
    [alertExit setText:@"Are you sure you want to exit?\n"];
    NSLog(@"Show alert for exit");
    //show the alert of exit
    [alertExit alertShow];
}

//memory deallocation
-(void) dealloc
{
    [alertExit release];
    [super dealloc];
}

@end
```

Joystic.h

```
/*
 Joystic.h
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
 */

//IMPORT LIBRARIES
#import "cocos2d.h"
#import "Box2D.h"
#import "Communication.h"

//DEFINE CONSTANT VARIABLES

//axis y 1
#define AXIS_Y 1
//axis x 0
#define AXIS_X 0
//diameter of the joystic
#define DIAMETER 128
//left joystic
#define LEFT_JOYSTIC 0
//right joystic
#define RIGHT_JOYSTIC 1
//cirle degrees
#define DEGREES 360

//INTERFACE Joystic INHERITS FROM CCLayer
@interface Joystic : CCLayer

//INTERFACE VARIABLES
{
    //the positiion of the sprite of the joystic
```

```
    CGPoint initialPosition;
    //the inner sprite for the joystic
    CCSprite *joystic;
    //world for box2d
    b2World *world;
    //joystic body
    b2Body *joysticBody;
    //joystic fixture
    b2Fixture *joysticFixture;
    //joystic body definition
    b2BodyDef joysticBodyDef;
    //joystic shape
    b2PolygonShape joysticShape;
    //joystic shape definition
    b2FixtureDef joysticShapeDef;
    //communication shared object
    Communication *communicate;
    //first channel number
    int sendingValue1;
    //second channel number
    int sendingValue2;
    //first channel value
    int channelxValue;
    //second channel value
    int channelyValue;
}

//PROPERTIES

@property (nonatomic,assign) CGPoint initialPosition;
@property (nonatomic,retain) CCSprite *joystic;
@property (nonatomic,assign) b2World *world;
@property (nonatomic,assign) b2Body *joysticBody;
@property (nonatomic,assign) b2Fixture *joysticFixture;
@property (nonatomic,assign) b2BodyDef joysticBodyDef;
@property (nonatomic,assign) b2PolygonShape joysticShape;
@property (nonatomic,assign) b2FixtureDef joysticShapeDef;
@property (nonatomic,retain) Communication *communicate;
@property (nonatomic,assign) int sendingValue1;
@property (nonatomic,assign) int sendingValue2;
@property (nonatomic,assign) int channelxValue;
@property (nonatomic,assign) int channelyValue;

//FUNCTIONS

//constructor takes the position of the joystic, the channels of the current joystic
-(void) initial: (CGPoint)position :(int)chx :(int)chy;
//called to learn if we touch the joystic
-(boolean_t)checkTouch:(b2Vec2)locationWorld;
//set the position of the joystic
-(void)setPosition:(CGPoint)location;
//find the euclidean distance from the center of the joystic
-(float) euclidianCenter:(CGPoint)location;
//find the euclidean distance of two points
```

```
-(float) euclidian: (float)a1 :(float)a2 :(float)a3 :(float)a4;
//convert the current position of the joystic to the value that we must send
-(int) convertToCircle: (float)current :(int)axis;
//returns the initial position of the joystic (center)
-(CGPoint) getInitialPosition;
//return the current position of the joystic
-(CGPoint) getCurrentPosition;
//set initial position of the joystic
-(void) setInitialPosition: (CGPoint)position;
//set the current position of the joystic
-(void) setCurrentPosition: (CGPoint) position;
//changes the body of the sprite of the joystic
-(void)changeBody: (int)a :(int)b;
//called when the touch moved - finds the value that we will must send
-(void)touchMoved;
//Set the sending values
-(void)setSendingValues: (int)val1 :(int)val2;
//Set the sending values to the communication class
-(void)updateValues;

//memory deallocation
-(void)dealloc;

@end
```

Joystic.mm

```
/*
 Joystic.mm
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
 */

//IMPORT LIBRARIES
#import "Joystic.h"

//CONSTANT VARIABLES
#define PTM_RATIO 32

//IMPLEMENTATION
@implementation Joystic

//SYNTHESIZE VARIABLES
@synthesize initialPosition;
@synthesize joystic;
@synthesize world;
@synthesize joysticBody;
@synthesize joysticFixture;
@synthesize joysticBodyDef;
@synthesize joysticShape;
@synthesize joysticShapeDef;
@synthesize communicate;
@synthesize sendingValue1;
```

```
@synthesize sendingValue2;
@synthesize channelxValue;
@synthesize channelyValue;

- (void) initial: (CGPoint)position :(int)chx :(int)chy{

    //communication tou update the values of channels of the joystics
    communicate=[Communication sharedCommunication];

    //set the initial position of the joystic
    [self setInitialPosition:position];

    //set the channel number of the joystic
    sendingValue1=chx;
    sendingValue2=chy;

    //create world
    b2Vec2 gravity = b2Vec2(0.0f, -10.0f);
    bool Sleep = true;
    world = new b2World(gravity, Sleep);

    //First cirlce sprite
    CCSprite *joy=[CCSprite spriteWithFile:@"joystick2.png"];
    joy.position = ccp(position.x,position.y);
    [self addChild:joy];

    //create circle and add it to the layer
    joystic = [CCSprite spriteWithFile:@"but3.png"];
    joystic.position = ccp(position.x,position.y);
    [self addChild:joystic];

    //create circle body
    joysticBodyDef.type = b2_dynamicBody;
    joysticBodyDef.position.Set((position.x)/PTM_RATIO, (position.y)/PTM_RATIO);
    joysticBodyDef.userData = joystic;
    joysticBody = world->CreateBody(&joysticBodyDef);

    //create circle shape
    joysticShape.SetAsBox(joystic.contentSize.width/PTM_RATIO/2,
                          joystic.contentSize.height/PTM_RATIO/2);

    //create shape definition and add to body
    joysticShapeDef.shape = &joysticShape;
    joysticShapeDef.density = 7.0f;
    joysticShapeDef.friction = 2.0f;
    joysticShapeDef.restitution = 0.1f;
    joysticFixture = joysticBody->CreateFixture(&joysticShapeDef);
}

//called to learn if we touch the joystic
-(boolean_t)checkTouch:(b2Vec2)locationWorld {
    if (joysticFixture->TestPoint(locationWorld))
        return true;
    else
```

```
        return false;
}

//set the position of the joystic
-(void)setPosition:(CGPoint)location
{
    joystic.position = ccp(location.x, location.y);
}

//find the euclidian distance from the center
-(float) euclidianCenter:(CGPoint)location{
    float d=sqrt((location.x-initialPosition.x)*(location.x-initialPosition.x)+(location.y-
initialPosition.y)*(location.y-initialPosition.y));
    return d;
}

//find the euclidean distance between two points
-(float) euclidian: (float)a1 :(float)a2 :(float)a3 :(float)a4{
    float d=sqrt((a1-a3)*(a1-a3)+(a2-a4)*(a2-a4));
    return d;
}

//find the sending values depending on the position of the sprite
-(int) convertToCircle: (float)current :(int)axis
{
    int data=0;
    float pointx;
    //axis y
    if (axis==AXIS_Y){
        pointx=[self getInitialPosition].y-current;
        data=128+(int)pointx;}
    else if (axis==AXIS_X)
    {
        //axis x
        pointx=[self getInitialPosition].x-current;
        data=128-(int)pointx;
    }
    return data;
}

// get initial position of the inner sprite
-(CGPoint) getInitialPosition
{
    return initialPosition;
}

//get current position of the inner sprite
-(CGPoint) getCurrentPosition
{
    return joystic.position;
}

// set initial position of the inner sprite
-(void) setInitialPosition :(CGPoint)position
```

```
{
    initialPosition=position;
}

// set current position of the inner sprite
-(void) setCurrentPosition :(CGPoint)position
{
    joystic.position=ccp(position.x,position.y);
}

//change the position of the joystic body
-(void)changeBody: (int)a :(int)b
{
    joysticBody->SetTransform(b2Vec2(a,b), 0);
}

-(void)touchMoved
{
    //get sending values
    int x=[self convertToCircle: [self getCurrentPosition].x :0];
    int y=[self convertToCircle: [self getCurrentPosition].y :1];
    [self setSendingValues:x :y];

}

-(void)setSendingValues: (int)val1 :(int)val2
{
    channelxValue=val1;
    channelyValue=val2;
    //where value are updated change the value to the communication
    [self updateValues];
}

//update the sending values to the communication
-(void)updateValues
{
    [communicate updateValues:sendingValue1 :channelxValue:sendingValue2 :channelyValue];
}

//memory deallocation
-(void)dealloc
{
    [joystic release];
    [super dealloc];
}

@end
```

Right.h

```
/*
 Right.h
 PMJOYSTIC
```

```
 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Joystic.h"

//INTERFACE Right INHERITS FROM Joystic
@interface Right : Joystic

//FUNCTIONS

//constructor of the right joystic
-(Right*) position:(CGPoint)position channelx:(int)chx channely:(int)chy;
//called when touch ended
-(void)touchEnded;
//return the nearest position on the circle
-(CGPoint) getNearestPoint:(CGPoint) p;

@end
```

Right.mm

```
/*
 Right.mm
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
 */

//IMPORT LIBRARIES
#import "Right.h"

//IMPLEMENTATION
@implementation Right

//constructor of the right joystic
-(Right*) position:(CGPoint)position  channelx:(int)chx channely:(int)chy{
    self = [super init];
    if (self != nil)
    {
        [super initial:position :chx :chy];

    }return self;
}

-(void)touchEnded
{
    //set the position when touch ended as the initial position
    [super setPosition:[super getInitialPosition]];

    //update sending values
    [communicate updateValues:sendingValue1:127:sendingValue2:127];
}

//return the nearest position on the circle
-(CGPoint) getNearestPoint:(CGPoint) p
{
    //initialize nearest point random
    float minx=  [self getInitialPosition].x + DIAMETER* cos(0);
    float  miny=  [self getInitialPosition].y + DIAMETER * sin(0);

    float  min=[self euclidian:minx:miny:p.x:p.y];
    float euclidean;
    float x;
    float y;

    //search for the nearest point at the diameter of the circle
    for (int i=0;i<=DEGREES;i++)
    {
        x=  [self getInitialPosition].x + DIAMETER * cos(i);
        y = [self getInitialPosition].y + DIAMETER * sin(i);
        euclidean=[self euclidian:x:y:p.x:p.y];
        if (min>euclidean)
        {
```

```
            min=euclidean;
            minx=x;
            miny=y;
        }

    }
    //return the nearest point
    return CGPointMake(minx, miny);
}

@end
```

Left.h

```
/*
 Left.h
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Joystic.h"

//define constant variables
//for box2d positions
#define PTM_RATIO 32

//INTERFACE Left INHERITS FROM Joystic
@interface Left : Joystic

//FUNCTIONS

//constructor of the right joystic
-(Left*) position:(CGPoint)position channelx:(int)chx channely:(int)chy;
//called when touch ended
-(void)touchEnded;
//return the nearest position on the circle
-(CGPoint) getNearestPoint:(CGPoint) p;

@end
```

Left.mm

```
/*
 Left.mm
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
 */

//IMPORT LIBRARIES
#import "Left.h"
```

```
//IMPLEMENTATION
@implementation Left

//constructor of the right joystic
- (Left*) position:(CGPoint)position  channelx:(int)chx channely:(int)chy{
    self = [super init];
    if (self != nil)
    {
        [super initial:position: chx:chy];
    }return self;
}

//called when touch ended
-(void)touchEnded
{
    //set the new position
    [self setPosition:CGPointMake([super getInitialPosition].x, [super getCurrentPosition].y)];

    //change body position because the new position may not be the initial pposition
    [super changeBody:[super getInitialPosition].x/PTM_RATIO:[super
getCurrentPosition].y/PTM_RATIO];

    //update the sending values
    [communicate updateValues:sendingValue1 :127:sendingValue2:[super convertToCircle:[self
getCurrentPosition].y :1]];
}

//return the nearest position on the circle
-(CGPoint) getNearestPoint:(CGPoint) p
{
    //initialize nearest point random
    float minx=  [self getInitialPosition].x+ DIAMETER* cos(0);
    float  miny= [self getInitialPosition].y + DIAMETER * sin(0);

    float  min=[self euclidian:minx:miny:p.x:p.y];
    float euclidean;
    float x;
    float y;

    //search for the nearest point at the diameter of the circle
    for (int i=0;i<=DEGREES;i++)
    {
        x=  [self getInitialPosition].x + DIAMETER * cos(i);
        y = [self getInitialPosition].y + DIAMETER * sin(i);
        euclidean=[self euclidian:x:y:p.x:p.y];
        if (min>euclidean)
        {
            min=euclidean;
            minx=x;
            miny=y;
        }

    }
```

```
    //return the nearest point
    return CGPointMake(minx, miny);
}

@end
```

Volume.h

```
/*
 Volume.h
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import <Foundation/Foundation.h>
#import "cocos2d.h"
#import "alertHide.h"
#import "alertHide.h"
#import <AVFoundation/AVAudioPlayer.h>

//INTERFACE Volume INHERITS FROM CCLayer
@interface Volume : CCLayer

//INTERFACE VARIABLES
{
    //alert for volume
    alertHide *alertVolume;
    //class to play the sound
    AVAudioPlayer *audio;
    //url for the file to play
    NSURL *url;
    //to see the error if sound not exist
    NSError *error;
}

//PROPERTIES

@property (nonatomic,retain) alertHide *alertVolume;
@property (nonatomic,retain) AVAudioPlayer *audio;
@property (nonatomic,retain) NSURL *url;
@property (nonatomic,retain) NSError *error;

//FUNCTIONS

//returns an object of Volume
-(id)init;
//called from the selectos or the accelerator
-(void)pushVolume:(int)onOff;
//play the file (sound)
-(void)playAudio;
//set sound
```

```
+(void)setSound:(boolean_t)x;
//get sound
+(boolean_t) getSound;

//memory deallocation
-(void)dealloc;

@end
```

Volume.m

```
/*
 Volume.m
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Volume.h"

//IMPLEMENTATION
@implementation Volume

//STATIC VARIABLES
static boolean_t sound;

//SYNTHESIZE VARIABLES
@synthesize alertVolume;
@synthesize audio;
@synthesize url;
@synthesize error;

- (id) init{
    self = [super init];
    if (self != nil) {

        //allocate memory for the alert
        alertVolume=[[alertHide alloc]message:@"Volume\n\n"];

        //file that plays is beep.mp3
        url = [NSURL fileURLWithPath:[NSString stringWithFormat:@"%@/beep.mp3",
[[NSBundle mainBundle] resourcePath]]];

        //allocate memory for the AVAudioPlayer
        audio = [[AVAudioPlayer alloc] initWithContentsOfURL:url error:&error];

        //play the sound once
        [audio setNumberOfLoops:0];

        //full volume
        [audio setVolume:1];
    }
    return self;
```

```
}

-(void)pushVolume:(int)onOff
{
    if (onOff==0)
        //change the alert message
        [alertVolume setText:@"Volume is off"];
    else
        //change the alert message
        [alertVolume setText:@"Volume is on!"];

    //show the alert
    [alertVolume alertShow];

    //update the interval from the begining
    [alertVolume alertShowWithInterval];
}

-(void)playAudio
{
    NSLog(@"Play sound");
    //if audio is nil the print the error description
    if (audio == nil)
        NSLog(@"%@",[error description]);
    //else play the sound
    else
        [audio play];
}

//set sound
+(void)setSound:(boolean_t)x
{
    sound=x;
}

//get sound
+(boolean_t) getSound
{
    return sound;
}

//memory deallocation
-(void) dealloc
{
    [alertVolume release];
    [audio release];
    [url release];
    [error release];
    [super dealloc];
}

@end
```

mainScreen.h

```
/*
 mainScreen.h
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
 */

//IMPORT LIBRARIES
#import "cocos2d.h"
#import "Box2D.h"
#import "AsyncUdpSocket.h"
#import "RCOIP.h"
#import "Communication.h"
#import "Exit.h"
#import "Right.h"
#import "Left.h"
#import "Accelerate.h"
#import "Volume.h"
#import "Background.h"
#import "secondScreen.h"
#import "switchButton.h"

//DEFINE CONSTANT VARIABLES

//diameter of the joystic
#define DIAMETER 128
//sending interval
#define INTERVAL 0.001
//accelerator on
#define ACCELERATOR_ON 1
//accelerator off
#define ACCELERATOR_OFF 0

//INTERFACE viewLayer INHERITS FROM CCLayer
@interface viewLayer : CCLayer

//INTERFACE VARIABLES
{
    //size of the screen
    CGSize size;
    //communication shared object
    Communication *communicate;
    //accelerator object
    Accelerate *move;
    //right joystic
    Right *rightJoystic;
    //left joystic
    Left *leftJoystic;
    //volume object for noise
    Volume *sound;
    //exit object to exit the application
    Exit *exit;
    //boolean flag - true if we touched right joystic else false
```

```
    int touchRight;
    //boolean flag - true if we touched left joystic else false
    int touchLeft;
    //first item of the accelerator button
    CCMenuItem *acceleratorOnItem;
    //second item of the accelerator button
    CCMenuItem *acceleratorOffItem;
    //toggle button for the accelerator
    CCMenuItemToggle *acceleratorTog;
    //menu for the toggle button to view to the screen
    CCMenu *acceleratorOnOff;
    //item for the exit button
    CCMenuItem *exitItem;
    //button to the screen
    CCMenu *exitApp;
    //item for the exit button
    CCMenuItem *ipItem;
    //button to the screen
    CCMenu *ip;
    //first item of the music toggle
    CCMenuItem *musicOnItem;
    //second item of the music toggle
    CCMenuItem *musicOffItem;
    //toggle button for the music
    CCMenuItemToggle *musicOnOffTog;
    //menu to view the toggle button to the screen
    CCMenu *musicOnOff;
    //backgroud for the app
    Background *background;
    //boolean flag - true if the ipad movement is on else falses
    boolean_t acceleratorOn;
    //next view
    ipScene *next;
}

//PROPERTIES

@property (nonatomic,assign) CGSize size;
@property (nonatomic,retain) Communication *communicate;
@property (nonatomic,retain) Right *rightJoystic;
@property (nonatomic,retain) Left *leftJoystic;
@property (nonatomic,retain) Accelerate *move;
@property (nonatomic,retain) Volume *sound;
@property (nonatomic,assign) int touchRight;
@property (nonatomic,assign) int touchLeft;
@property (nonatomic,retain) CCMenuItem *acceleratorOnItem;
@property (nonatomic,retain) CCMenuItem *acceleratorOffItem;
@property (nonatomic,retain) CCMenu *acceleratorOnOff;
@property (nonatomic,retain) CCMenuItemToggle *acceleratorTog;
@property (nonatomic,retain) CCMenu *exitApp;
@property (nonatomic,retain) CCMenuItem *exitItem;
@property (nonatomic,retain) CCMenu *ip;
@property (nonatomic,retain) CCMenuItem *ipItem;
@property (nonatomic,retain) CCMenu *musicOnOff;
```

```
@property (nonatomic,retain) CCMenuItem *musicOnItem;
@property (nonatomic,retain) CCMenuItem *musicOffItem;
@property (nonatomic,retain) CCMenuItemToggle *musicOnOffTog;
@property (nonatomic,retain) Background *background;
@property (nonatomic,assign) boolean_t volumeOn;
@property (nonatomic,assign) boolean_t acceleratorOn;
@property (nonatomic,retain) ipScene *next;

//FUNCTIONS

//enable touch
-(void)enableTouches;
//called when touch began
-(void)ccTouchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
//called when we have movement on the screen
-(void)ccTouchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
//called when touch ended
-(void)ccTouchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
//called every 1msec to send the updated data to the arduino
-(void)dataSend:(id)sender;
//function to get the sprite position from the accelerator
-(void)getSpriteAcceleratorValues:(id)sender;

//Selectors

//called when we push the the Accelerator button
-(void)acceleratorSelector:(id)sender;
//called when we push Exit button
-(void)exitSelector:(id)sender;
//called when we push volume button
-(void)volumeSelector:(id)sender;

//memory deallocation
- (void)dealloc;

@end

//INTERFACE viewScene INHERITS FROM CCScene
@interface viewScene : CCScene

//INTERFACE VARIABLES
{
   viewLayer *layer;
}

//PROPERTIES
@property (nonatomic, retain) viewLayer *layer;

//FUNCTIONS

//return scene object
- (id) init;

//memory deallocation
```

```
-(void)dealloc;

@end
```

mainScreen.mm

```
/*
 mainScreen.mm
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "mainScreen.h"

//IMPLEMENTATION
@implementation viewLayer

//SYNTHESIZE VARIABLES
@synthesize size;
@synthesize communicate;
@synthesize leftJoystic;
@synthesize rightJoystic;
@synthesize move;
@synthesize sound;
@synthesize touchRight;
@synthesize touchLeft;
@synthesize acceleratorOnItem;
@synthesize acceleratorOffItem;
@synthesize acceleratorOnOff;
@synthesize acceleratorTog;
@synthesize exitApp;
@synthesize exitItem;
@synthesize musicOnItem;
@synthesize musicOffItem;
@synthesize musicOnOff;
@synthesize musicOnOffTog;
@synthesize background;
@synthesize acceleratorOn;
@synthesize ipItem;
@synthesize ip;
@synthesize next;

- (id) init
{
    self = [super init];
    if (self != nil) {

        //size of the screen
        size = [CCDirector sharedDirector].winSize;

        //communication to send to the arduino the channel values every 1msec
        communicate=[Communication sharedCommunication];

        //call method to enable touch
        [self enableTouches];
```

```
        //send the data to the arduino with interval 1msec
        [self schedule:@selector(dataSend:)interval:INTERVAL];

        //at the benning the volume is on
        [Volume setSound:true];

        //at the beginig the accelerator is off
        acceleratorOn=false;

        //initialize that the 2 joystics are not touched
        touchRight=false;
        touchLeft=false;

        /********************************** BACKGROUND ************************/
        background=[[Background alloc]init:@"black.jpg"];
        [self addChild:background];

/*****************************************************************************/

        /*********************************  ACCELERATOR TOGGLE  ************/
        //pictures items for the accelerator toggle
        acceleratorOnItem =[CCMenuItemImage itemFromNormalImage:@"on.gif"
selectedImage:@"on.gif"];
        acceleratorOffItem =[CCMenuItemImage itemFromNormalImage:@"off.gif"
selectedImage:@"off.gif"];
        //add the items to the accelerator toggle
        acceleratorTog =[CCMenuItemToggle itemWithTarget:self
selector:@selector(acceleratorSelector:) items:acceleratorOffItem,acceleratorOnItem, nil];
        //add to the menu the accelerator toggle
        acceleratorOnOff = [CCMenu menuWithItems:acceleratorTog, nil];
        //define the position of the accelerator button
        acceleratorOnOff.position =ccp(size.width/2,size.height/2-120);
        //add the accelerometer button to the layer
        [self addChild:acceleratorOnOff];

/*****************************************************************************/

        /*************************************  EXIT BUTTON  *****************/
        // button for exit the application
        exitItem=[CCMenuItemImage itemFromNormalImage:@"exit.png"
selectedImage:@"exit.png" target:self selector:@selector(exitSelector:)];
        exitApp = [CCMenu menuWithItems:exitItem,nil];
        //set the position of the exit button
        exitApp.position =ccp(size.width-80,size.height-70);
        //add the button to the layer
        [self addChild:exitApp];

/*****************************************************************************/

        /*************************************  VOLUME BUTTON  ***************/
        musicOnItem =[CCMenuItemImage itemFromNormalImage:@"music_on.gif"
selectedImage:@"music_on.gif"];
        musicOffItem =[CCMenuItemImage itemFromNormalImage:@"music_off.gif"
selectedImage:@"music_off.gif"];
```

```
        musicOnOffTog = [CCMenuItemToggle itemWithTarget:self
selector:@selector(volumeSelector:) items:musicOnItem ,musicOffItem, nil];
        musicOnOff = [CCMenu menuWithItems:musicOnOffTog, nil];
        musicOnOff.position =ccp(size.width-220,size.height-70);
        [self addChild:musicOnOff];

/*****************************************************************************/

        /********************************** SOUND *********************************/
        //allocate memory for volume
        sound=[[Volume alloc]init];
        //add the volume to the layer
        [self addChild:sound];

/*****************************************************************************/

        /*************************************LEFT JOYSTIC **********************/
        leftJoystic = [[Left alloc] position:CGPointMake(size.width/2 - 282, size.height/2-60)
channelx:0 channely:1];
        [self addChild:leftJoystic];

/*****************************************************************************/

        /*************************************RIGHT JOYSTIC **********************/
        rightJoystic = [[Right alloc] position:CGPointMake(size.width/2 + 282,size.height/2-60)
channelx:2 channely:3];
        [self addChild:rightJoystic];

/*****************************************************************************/

        /************************************ SWITCH BUTTON *****************/

        switchButton *switchObj=[[switchButton
alloc]init:@"but1.png":@"but2.png":size.width/2:size.height/2:4:@"Disable":@"Enable"];
        [self addChild:switchObj];


/*****************************************************************************/

        /************************************ SETTINGS BUTTON *****************/
        ipItem=[CCMenuItemImage itemFromNormalImage:@"Settings.png"
selectedImage:@"Settings.png" target:self selector:@selector(ipSelector:)];
        ip = [CCMenu menuWithItems:ipItem,nil];
        //set the position of the settings button
        ip.position =ccp(size.width-320,size.height-70);
        //add the button to the layer
        [self addChild:ip];

/*****************************************************************************/
    }
    return self;
}

-(void)enableTouches
```

```
{
    //touch enable
    self.isTouchEnabled=YES;
}

-(void)dataSend:(id)sender
{
    //send the data
    [communicate dataSend];
}

//called when touch began
- (void)ccTouchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    NSArray *touchTable = [touches allObjects];

    UITouch *firstTouch;
    UITouch *secondTouch = nil;

    CGPoint firstPoint;
    CGPoint secondPoint;

    //one touch
    if ([touchTable count] ==1)
    {
        //UITouch object for one touch
        firstTouch = [touchTable objectAtIndex:0];


        firstPoint = [firstTouch locationInView:[firstTouch view]];
        firstPoint = [[CCDirector sharedDirector] convertToGL:firstPoint];
    }

    //two touches
    if ([touchTable count]==2)
    {
        //UITouch object for first touch
        firstTouch= [touchTable objectAtIndex:0];

        //UITouch object for second touch
        secondTouch = [touchTable objectAtIndex:1];

        //convert UITouch object to CGPoint, to get the touching position

        firstPoint = [firstTouch locationInView:[firstTouch view]];
        firstPoint = [[CCDirector sharedDirector] convertToGL:firstPoint];

        secondPoint = [secondTouch locationInView:[secondTouch view]];
        secondPoint = [[CCDirector sharedDirector] convertToGL:secondPoint];

    }

    //if we touch left joystic set flag touchRight true
    if (([leftJoystic checkTouch:b2Vec2(firstPoint.x/PTM_RATIO, firstPoint.y/PTM_RATIO)] ||
```

```
[leftJoystic checkTouch:b2Vec2(secondPoint.x/PTM_RATIO, secondPoint.y/PTM_RATIO)])
&&touchLeft==false)
        touchLeft=true;

    //if we touch right joystic set the flag touchLeft true
    if (([rightJoystic checkTouch:b2Vec2(firstPoint.x/PTM_RATIO, firstPoint.y/PTM_RATIO)]
&& acceleratorOn==false && touchRight==false) ||([rightJoystic
checkTouch:b2Vec2(secondPoint.x/PTM_RATIO, secondPoint.y/PTM_RATIO)] &&
acceleratorOn==false &&touchRight==false))
        touchRight=true;
}

//called when we have movement on the screen
-(void)ccTouchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
{

    NSArray *touchTable = [touches allObjects];

    UITouch *firstTouch;
    UITouch *secondTouch;
    CGPoint firstPoint;
    CGPoint secondPoint;

    //one touch
    if ([touchTable count] ==1)
    {

        firstTouch = [touchTable objectAtIndex:0];


        firstPoint = [firstTouch locationInView:[firstTouch view]];
        firstPoint = [[CCDirector sharedDirector] convertToGL:firstPoint];

        float dist=[leftJoystic euclidian:firstPoint.x :firstPoint.y:[leftJoystic
getCurrentPosition].x:[leftJoystic getCurrentPosition].y];
        float dist2=[rightJoystic euclidian:firstPoint.x :firstPoint.y:[rightJoystic
getCurrentPosition].x:[rightJoystic getCurrentPosition].y];

        if (dist<dist2){
            //if we touch left
            if (touchLeft==true)
            {
                float distance=[leftJoystic euclidianCenter:firstPoint];

                //touch in bounds else out of bounds
                if (distance<=DIAMETER)
                    //set the joystic position at the position that user touched
                    [leftJoystic setPosition:firstPoint];
                else
                    //set the position of joystic on the nearestn point on the perimeter of the circle
                    [leftJoystic setPosition:[leftJoystic getNearestPoint:firstPoint]];

                //call the touchMoved for left joystic
                [leftJoystic touchMoved];
```

```
        }}
    else
      //if touch right
      if (touchRight==true)
    {
      float distance=[rightJoystic euclidianCenter:firstPoint];

      //touch in bounds else out of bounds
      if (distance<=DIAMETER)
        //set the joystic position at the position that user touched
        [rightJoystic setPosition:firstPoint];
      else
        //set the position of joystic on the nearestn point on the perimeter of the circle
        [rightJoystic setPosition:[rightJoystic getNearestPoint:firstPoint]];

      //call the touchMoved for right joystic
      [rightJoystic touchMoved];
    }

  }

  //two touches
  if ([touchTable count]==2)
  {
    firstTouch = [touchTable objectAtIndex:0];

    secondTouch = [touchTable objectAtIndex:1];

    //convert UITouch to point

    firstPoint = [firstTouch locationInView:[firstTouch view]];
    firstPoint = [[CCDirector sharedDirector] convertToGL:firstPoint];

    secondPoint = [secondTouch locationInView:[secondTouch view]];
    secondPoint = [[CCDirector sharedDirector] convertToGL:secondPoint];


    float dist=[leftJoystic euclidian:firstPoint.x :firstPoint.y:[leftJoystic
getCurrentPosition].x:[leftJoystic getCurrentPosition].y];
    float dist2=[leftJoystic euclidian:secondPoint.x :secondPoint.y:[leftJoystic
getCurrentPosition].x:[leftJoystic getCurrentPosition].y];

    CGPoint first=secondPoint;
    CGPoint second=firstPoint;


    if (dist<dist2)
    {
      first=firstPoint;
      second=secondPoint;
    }

    //if user touched the right joystic
    if (touchLeft==true)
```

```
        {
            float distance=[leftJoystic euclidianCenter:first];

            //touch in bounds else out of bounds
            if (distance<=DIAMETER)
                //set the joystic position at the position that user touched
                [leftJoystic setPosition:first];
            else
                //set the position of joystic on the nearestn point on the perimeter of the circle
                [leftJoystic setPosition:[leftJoystic getNearestPoint:first]];

            //call the touchMoved for left joystic
            [leftJoystic touchMoved];
        }

        //if user touched the left joystic
        if (touchRight==true)
        {
            float distance=[rightJoystic euclidianCenter:second];

            //touch in bounds else out of bounds
            if (distance<=DIAMETER)
                //set the joystic position at the position that user touched
                [rightJoystic setPosition:second];
            else
                //set the position of joystic on the nearestn point on the perimeter of the circle
                [rightJoystic setPosition:[rightJoystic getNearestPoint:second]];

        //call the touchMoved for right joystic
        [rightJoystic touchMoved];
        }
    }
}

//called when touch ended
- (void)ccTouchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
    //if the touch was on the right joystic
    if (touchLeft==true)
        [leftJoystic touchEnded];

    //if the touch was on the left joystic
    if (touchRight==true)
        [rightJoystic touchEnded];

    //when touch ended then define again that 2 joysticks are untouch
    touchRight=false;
    touchLeft=false;
}

//this function is called when we push the accelerator button
-(void)acceleratorSelector:(id)sender
{
```

```
    if ([Volume getSound]==true)
        [sound playAudio];

    //enable accelerometer else disable
    if (acceleratorTog.selectedIndex==1)
    {
        NSLog(@"Accelerator is on.");
        acceleratorOn=true;

        [self schedule:@selector(getSpriteAcceleratorValues:) interval:INTERVAL];

        //accelerator is on - memory allocation for accelerator
        move=[[Accelerate alloc]init:[rightJoystic getInitialPosition]];
        [move pushAccelerator:ACCELERATOR_ON];

    }
    else if (acceleratorTog.selectedIndex==0)
    {
        NSLog(@"Accelerator is off.");

        acceleratorOn=false;

        [rightJoystic setPosition:[rightJoystic getInitialPosition]];
        [self unschedule:@selector(getSpriteAcceleratorValues:)];

        //accelerator is off - show the alert that is off and deallocate the memory
        [move pushAccelerator:ACCELERATOR_OFF];
        //[move release];
    }
}

//function to get the sprite position from the accelerator
-(void)getSpriteAcceleratorValues:(id)sender
{
    CGPoint foundPosition;

    foundPosition=[move getSpriteAcceleratorPosition];

    float distance=[rightJoystic euclidianCenter:foundPosition];

    //define position of the sprite from iPad movement
    if (distance<=DIAMETER)
        [rightJoystic setPosition:foundPosition];
    else
        [rightJoystic setPosition:[rightJoystic getNearestPoint:foundPosition]];

    //find the sending values
    [rightJoystic setSendingValues:[rightJoystic convertToCircle:[rightJoystic
getCurrentPosition].x  :0]:[rightJoystic convertToCircle:[rightJoystic getCurrentPosition].y :1]];
}

//called when we oush exit button
-(void)exitSelector:(id)sender
{
```

```
    if ([Volume getSound]==true)
        [sound playAudio];

    exit=[[Exit alloc]init];

    NSLog(@"Asks for exit");

    //show the alert for exit
    [exit pushExit];
}

-(void)ipSelector:(id)sender
{
    if ([Volume getSound]==true)
        [sound playAudio];

    next = [[ipScene alloc]init];
         [[CCDirector sharedDirector] pushScene:[CCFadeTransition transitionWithDuration:1.0
scene:next withColor:ccBLACK]];

}

//called when we push volume button
-(void)volumeSelector:(id)sender
{
    if (musicOnOffTog.selectedIndex==1)
    {
        NSLog(@"Volume is off.");

        //sound is off - show the alert that is off and deallocate the memory
        [sound pushVolume:0];
        [sound release];

        //set the sound false
        [Volume setSound:false];

    }
    else if (musicOnOffTog.selectedIndex==0)
    {
        NSLog(@"Volume is on.");

        sound=[[Volume alloc]init];
        [self addChild:sound];

        [sound pushVolume:1];

        //play the sound
        [sound playAudio];

        //set the sound true
        [Volume setSound:true];
    }

}
```

```
//memory deallocation
-(void)dealloc
{
    [communicate release];

    if (acceleratorOn==true)
      [move release];

     if ([Volume getSound]==true)
       [sound release];

    [leftJoystic release];

    [rightJoystic release];

    [acceleratorOnItem release];
    [acceleratorOffItem release];
    [acceleratorOnOff release];
    [acceleratorTog release];

    [exitApp release];
    [exitItem release];

    [musicOnItem release];
    [musicOffItem release];
    [musicOnOff release];
    [musicOnOffTog release];

    [background release];

    [super dealloc];
}

@end

//IMPLEMENTATION
@implementation viewScene

//SYNTHESIZE VARIABLES
@synthesize layer;

//return scene object
- (id) init
{
    self = [super init];
    if (self != nil) {

        layer = [[viewLayer alloc]init];
        [self addChild: layer];
    }

         return self;
}
```

```
//memory deallocation
-(void) dealloc
{
    [layer release];
    [super dealloc];
}

@end
```

secondScreen.h

```
/*
 secondScreen.h
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import <Foundation/Foundation.h>
#import "Alert.h"
#import "cocos2d.h"
#import "Background.h"
#import "Communication.h"
#import "Volume.h"

//INTERFACE ipLayer INHERITS FROM CCLayer
@interface ipLayer:CCLayer

//INTERFACE VARIABLES
{
    //communication shared object
    Communication *com;
    //size of the screen
    CGSize size;
    //background of the screen
    Background *background;
    //ip number
    NSString *ip;
    //port number
    NSString *port;
    //label for ip
    CCLabel *assignIP;
    //label for port
    CCLabel *assignPort;
    //port label
    CCLabel *portNumber;
    //ip label
    CCLabel *ipAddress;
    //ip title for text field
    CCLabel *ipTitle;
    //port title for text field
    CCLabel *portTitle;
```

```
    //item for ok button
    CCMenuItem *okItem;
    //ok button
    CCMenu *ok;
    //item for back button
    CCMenuItem *backItem;
    //back button
    CCMenu *back;
    //text field for ip
    UITextField *ipTextField;
    //text field for port
    UITextField *portTextField;
    //sound object
    Volume *sound;
    //correct ip text
    boolean_t correctIpInput;
    //correct port text
    boolean_t correctPortInput;
    //alert for the correctness of the input
    alertHide * alertCorrect;
}

//PROPERTIES

@property (nonatomic,retain) Communication *com;
@property (nonatomic,assign) CGSize size;
@property (nonatomic,retain) Background *background;
@property (nonatomic,retain) NSString *ip;
@property (nonatomic,retain) NSString *port;
@property (nonatomic,retain) CCLabel *assignIP;
@property (nonatomic,retain) CCLabel *assignPort;
@property (nonatomic,retain) CCLabel *portNumber;
@property (nonatomic,retain) CCLabel *ipAddress;
@property (nonatomic,retain) CCLabel *ipTitle;
@property (nonatomic,retain) CCLabel *portTitle;
@property (nonatomic,retain) CCMenuItem *okItem;
@property (nonatomic,retain) CCMenu *ok;
@property (nonatomic,retain) CCMenuItem *backItem;
@property (nonatomic,retain) CCMenu *back;
@property (nonatomic,retain) UITextField *ipTextField;
@property (nonatomic,retain) UITextField *portTextField;
@property (nonatomic,retain) Volume *sound;
@property (nonatomic,assign) boolean_t correctIpInput;
@property (nonatomic,assign) boolean_t correctPortInput;
@property (nonatomic,retain) alertHide *alertCorrect;

//FUNCTIONS

//set ip address
-(void)setIP: (NSString *)ip2;
// set port number
-(void)setPort: (NSString *)port2;
// selector for ok button
-(void)okSelector:(id)sender;
```

```
//selector for back
-(void)backSelector:(id)sender;
//check correctness of ip
-(boolean_t)checkIp:(NSString *)ip;
//check correctness of port
-(boolean_t)checkPort:(NSString *)port;

//memory deallocation
-(void) dealloc;

@end

//INTERFACE ipScene INHERITS FROM CCSene
@interface ipScene: CCScene

//INTERFACE VARIABLES
{
   ipLayer *layer;
}

//PROPERTIES

@property (nonatomic,assign) ipLayer *layer;

//FUNCTIONS

-(id) init;
-(void) dealloc;

@end
```

secondScreen.mm

```
/*
 secondScreen.mm
 PMJOYSTIC

 Copyright (c) 2013 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "secondScreen.h"
#import "Background.h"
#import "Volume.h"
#import "mainScreen.h"
#import "Volume.h"

//IMPLEMENTATION
@implementation ipLayer

//SYNTHESIZE VARIABLES
@synthesize com;
@synthesize size;
@synthesize background;
```

```
@synthesize ip;
@synthesize port;
@synthesize assignIP;
@synthesize assignPort;
@synthesize portNumber;
@synthesize ipAddress;
@synthesize ipTitle;
@synthesize portTitle;
@synthesize okItem;
@synthesize ok;
@synthesize backItem;
@synthesize back;
@synthesize ipTextField;
@synthesize portTextField;
@synthesize sound;
@synthesize correctIpInput;
@synthesize correctPortInput;
@synthesize alertCorrect;

- (id) init{
    self = [super init];
    if (self != nil) {

        com=[Communication sharedCommunication];

        size = [CCDirector sharedDirector].winSize;

        /********************************** ADD BACKGROUND ********************/
        background=[[Background alloc]init:@"black.jpg"];
        [self addChild:background];

/*****************************************************************************/

        /********************************** LABELS FOR IP *********************/
        assignIP = [CCLabel labelWithString:@"Assign IP address is:" fontName:@"Marker Felt"
fontSize:30];
        assignIP.position = ccp(size.width/2-130, size.height/2+250);
        assignIP.color=ccc3(255,255,255);
        [self addChild:assignIP];

        ipAddress = [CCLabel labelWithString:[com getIP] fontName:@"Marker Felt"
fontSize:30];
        ipAddress.position = ccp(size.width/2+130, size.height/2+250);
        ipAddress.color=ccc3(255,255,255);
        [self addChild:ipAddress];

        ipTitle = [CCLabel labelWithString:@"Give IP address" fontName:@"Marker Felt"
fontSize:30];
        ipTitle.position = ccp(size.width/2, size.height/2+60);
        ipTitle.color=ccc3(255,255,255);
        [self addChild:ipTitle];

/*****************************************************************************/
```

```
	/********************************** LABELS FOR PORT ********************/
	assignPort = [CCLabel labelWithString:@"Assign port number is:" fontName:@"Marker
Felt" fontSize:30];
	assignPort.position = ccp(size.width/2-112, size.height/2+200);
	assignPort.color=ccc3(255,255,255);
	[self addChild:assignPort];

	portNumber = [CCLabel labelWithString:[com getPortNum] fontName:@"Marker Felt"
fontSize:30];
	portNumber.position = ccp(size.width/2+110, size.height/2+200);
	portNumber.color=ccc3(255,255,255);
	[self addChild:portNumber];

	portTitle = [CCLabel labelWithString:@"Give port number" fontName:@"Marker Felt"
fontSize:30];
	portTitle.position = ccp(size.width/2, size.height/2-50);
	portTitle.color=ccc3(255,255,255);
	[self addChild:portTitle];

/*****************************************************************************/

	/********************************** OK BUTTON ***************************/
	// button for ok
	okItem=[CCMenuItemImage itemFromNormalImage:@"ok-1.png" selectedImage:@"ok-
1.png" target:self selector:@selector(okSelector:)];
	ok = [CCMenu menuWithItems:okItem,nil];
	//set the position of the ok button
	ok.position =ccp(size.width/2,size.height/2 - 180);
	//add the button to the layer
	[self addChild:ok];

/*****************************************************************************/

	/********************************** BACK BUTTON *************************/
	// button for back
	backItem=[CCMenuItemImage itemFromNormalImage:@"back.gif"
selectedImage:@"back.gif" target:self selector:@selector(backSelector:)];
	back = [CCMenu menuWithItems:backItem,nil];
	//set the position of the back button
	back.position =ccp(100,size.height - 60);
	//add the button to the layer
	[self addChild:back];

/*****************************************************************************/

	/********************************** FIELDS FOR IP & PORT ******************/
	//ip field
	ipTextField = [[UITextField alloc] initWithFrame:CGRectMake(260, 500, 280.0f, 40.0f)];
	ipTextField.placeholder = @"IP address";
	ipTextField.backgroundColor = [UIColor whiteColor];
	ipTextField.textColor = [UIColor blackColor];
	ipTextField.font = [UIFont systemFontOfSize:25.0f];
	ipTextField.borderStyle = UITextBorderStyleRoundedRect;
	ipTextField.textAlignment = NSTextAlignmentLeft;
```

```
        ipTextField.transform=CGAffineTransformMakeRotation(M_PI/2);
        [[[CCDirector sharedDirector] openGLView] addSubview:ipTextField];

        //port field
        portTextField = [[UITextField alloc] initWithFrame:CGRectMake(150, 500, 280.0f,
40.0f)];
        portTextField.placeholder = @"Port number";
        portTextField.backgroundColor = [UIColor whiteColor];
        portTextField.textColor = [UIColor blackColor];
        portTextField.font = [UIFont systemFontOfSize:25.0f];
        portTextField.borderStyle = UITextBorderStyleRoundedRect;
        portTextField.textAlignment = NSTextAlignmentLeft;
        portTextField.transform=CGAffineTransformMakeRotation(M_PI/2);
        [[[CCDirector sharedDirector] openGLView] addSubview:portTextField];

/*****************************************************************************/
    }
    return self;
}

-(void)okSelector:(id)sender
{
    //play sound if is true
    if ([Volume getSound]==true)
    {
        sound=[[Volume alloc]init];
        [sound playAudio];
    }

    // check the correctness of the fiels - then go to main screen
    if ([self checkIp:[ipTextField text]] && [self checkPort:[portTextField text]])
    {
        for (UIView *view in [[[CCDirector sharedDirector]
openGLView].viewForBaselineLayout subviews])
        {
            [view removeFromSuperview];
        }
        [com setIP:[ipTextField text]];
        [com setPortNum:[portTextField text]];
        [[CCDirector sharedDirector] popScene];
    }
    //if we have at least one wrong input
    else
    {
        NSString *alertMsg = [[NSString alloc]init];

        if (![self checkIp:[ipTextField text]])
            {
                //attach to the alert msg
                alertMsg = [alertMsg stringByAppendingString:@"Wrong IP address. Must be from
0.0.0.0.to 255.255.255.255\n\n"];
            }

        if (![self checkPort:[portTextField text]])
```

```
            {
                //attach to the alert msg
                alertMsg = [alertMsg stringByAppendingString:@"Wrong port number. Must be from
0 to  65535"];
            }

        alertCorrect=[[alertHide alloc]message:@"Wrong Input\n\n"];
        [alertCorrect setText:alertMsg];

        //show the alert
        [alertCorrect alertShow];
        [alertCorrect alertShowWithInterval];
    }

}

//back selector
-(void)backSelector:(id)sender
{
    //play sound if is true
    if ([Volume getSound]==true)
    {
        sound=[[Volume alloc]init];
        [sound playAudio];
    }

    //go to the main screen - remove all subviews (text fields)
    for (UIView *view in [[[CCDirector sharedDirector] openGLView].viewForBaselineLayout
subviews])
    {
        [view removeFromSuperview];
    }
    [[CCDirector sharedDirector] popScene];
}

//check ip address
-(boolean_t)checkIp:(NSString *)ipString
{
    NSArray *field = [ipString componentsSeparatedByString:@"."];
    if ([field count]!=4)
        return false;

    int num1 = [field[0] intValue];
    int num2 = [field[1] intValue];
    int num3 = [field[2] intValue];
    int num4 = [field[3] intValue];

    if (num1>=0 && num1<=255 && num2>=0 && num2<=255 && num3>=0 &&
num3<=255 && num4>=0 && num4<=255)
        return true;

        return false;
}
```

```
//check port number
-(boolean_t)checkPort:(NSString *)portString
{
    int portNum = [portString intValue];
    if (portNum <= 0 || portNum > 65535)
        return false;
    return true;
}

//set ip
-(void)setIP: (NSString *)ip2
{
    ip=ip2;
}

//set port
-(void)setPort: (NSString *)port2
{
    port=port2;
}

//memory deallocation
-(void) dealloc
{
    [background release];
    [ip release];
    [port release];
    [assignIP release];
    [assignPort release];
    [portNumber release];
    [ipAddress release];
    [ipTitle release];
    [portTitle release];
    [okItem release];
    [ok release];
    [ipTextField release];
    [portTextField release];
    [alertCorrect release];
    [super dealloc];
}

@end

//IMPLEMENTATION
@implementation ipScene

//SYNTHESIZE VARIABLES
@synthesize layer;

//constructor
- (id) init {
    self = [super init];
    if (self != nil) {
```

```
        layer = [[ipLayer alloc]init];
        [self addChild: layer];
    }

         return self;
}

//memory deallocation
-(void) dealloc
{
    [layer release];
    [super dealloc];
}
@end
```

switchButton.h

```
/*
 switchButton.h
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
 */

//IMPORT LIBRARIES
#import "cocos2d.h"
#import "communication.h"
#import "Volume.h"
#import "mainScreen.h"

//INTERFACE Switch INHERITS FROM CCLayer
@interface switchButton : CCLayer

//INTERFACE VARIABLES
{
    //item for the first picture of the toggle
    CCMenuItem *pic1;
    //item for the second picture of the toggle
    CCMenuItem *pic2;
    //menu to view the toggle button to the screen
    CCMenu *switchView;
    //toggle button the the switch
    CCMenuItemToggle *switchTog;
    //object for the communication
    Communication *communicate;
    //sound when push switch
    Volume *sound;
    //channel of the switch
    int channel;
    //alert of the switch that hides with interval
    alertHide *alertSwitch;
    //fisrt msg of the alert
    NSString *msg1;
    //second message of the alert
    NSString *msg2;
}

//PROPERTIES

@property (nonatomic,retain) CCMenuItemToggle *switchTog;
@property (nonatomic,retain) CCMenuItem *pic1;
@property (nonatomic,retain) CCMenuItem *pic2;
@property (nonatomic,retain) CCMenu *switchView;
@property (nonatomic,retain) Communication *communicate;
@property (nonatomic,retain) Volume *sound;
@property (nonatomic,retain) alertHide *alertSwitch;
@property (nonatomic,retain) NSString *msg1;
@property (nonatomic,retain) NSString *msg2;
```

```
//FUNCTIONS

//constructor that takes the images, the position, the channel, the messages for the alert of the
switch
- (id) init:
(NSString*)picF :(NSString*)picS :(int)posX :(int)posY :(int)channelVal :(NSString*)msgF :(N
SString*)msgS;

//Selector
//called when we push the switch
-(void)channelSelector: (id)sender;

//memory deallocation
-(void)dealloc;

@end
```

switchButton.mm

```
/*
 switchButton.mm
 PMJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
 */

//IMPORT LIBRARIES
#import "switchButton.h"

//IMPLEMENTATION
@implementation switchButton

//SYNTHESIZE VARIABLES
@synthesize switchTog;
@synthesize pic1;
@synthesize pic2;
@synthesize switchView;
@synthesize communicate;
@synthesize sound;
@synthesize alertSwitch;
@synthesize msg1;
@synthesize msg2;

//constructor that takes the images, the position, the channel, the messages for the alert of the
switch
- (id) init:
(NSString*)picF :(NSString*)picS :(int)posX :(int)posY :(int)channelVal :(NSString*)msgF :(N
SString*)msgS{
    self = [super init];
    if (self != nil) {

        //communication to update the sending values for each switch
        communicate=[Communication sharedCommunication];
```

```
        //sound for the switches
        sound=[[Volume alloc]init];

        msg1=msgF;
        msg2=msgS;

        alertSwitch=[[alertHide alloc]message:@"Button alert\n\n"];

        // switches items
        pic1=[CCMenuItemImage itemFromNormalImage:picF selectedImage:picF];
        pic2=[CCMenuItemImage itemFromNormalImage:picS selectedImage:picS];

        //toggle button
        switchTog = [CCMenuItemToggle itemWithTarget:self
selector:@selector(channelSelector:) items:pic1,pic2,nil];


        switchView = [CCMenu menuWithItems:switchTog, nil];

        //position of the switch
        switchView.position =ccp(posX,posY);

        channel=channelVal;

        [self addChild:switchView];
    }
    return self;

}

//called when we push the switch
-(void)channelSelector:(id)sender{
     if (switchTog.selectedIndex==0)
    {
        //change the message of the alert
        [alertSwitch setText:msg1];
        //update the sending values
        [communicate updateValues2:channel:0];
    }
    else
    {
        //change the message of the alert
        [alertSwitch setText:msg2];
        //update the sending values
        [communicate updateValues2:channel:1];
    }

    if ([Volume getSound]==true)
    {
        //play sound
        [sound playAudio];
    }

    //show the alert
```

```
    [alertSwitch alertShow];

    //update the interval of the alert from the begining
    [alertSwitch alertShowWithInterval];
}

//memory deallocation
-(void) dealloc
{
    [msg1 release];
    [msg2 release];
    [switchTog release];
    [pic1 release];
    [pic2 release];
    [switchView release];
    [super dealloc];
}

@end
```

Communication.h

```
/*
 Communication.h
 MPJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "cocos2d.h"
#import "AsyncUdpSocket.h"
#import "RCOIP.h"

//INTERFACE Communication INHERITS FROM NSObject
@interface Communication:NSObject

//INTERFACE VARIABLES
{
    //object for the communication to the arduino over udp
    AsyncUdpSocket *udpSocket;
    //the tag of the sending packet
    long tag;
    //arduino ip
    NSString *host;
    //struct from RCKIT library that define the sending message structure
    RCOIPv1CmdSetAnalogChannels messageStructure;
    //error while binding
    NSError *error;
    //the sending data for the udp Socket
    NSMutableData *data;
    // port string
    NSString *portText;
    // integer port
```

```
    int port;
}

//PROPERTIES

@property (nonatomic,retain) AsyncUdpSocket *udpSocket;
@property (nonatomic, assign) long tag;
@property (nonatomic,retain) NSString *host;
@property (nonatomic,assign) RCOIPv1CmdSetAnalogChannels messageStructure;
@property (nonatomic,retain) NSError *error;
@property (nonatomic,retain) NSMutableData *data;
@property (nonatomic,retain) NSString *portText;
@property (nonatomic,assign) int port;

//FUNCTIONS

//create the singletton
+(Communication *)sharedCommunication;
//returns a Communication object
- (id) init;
//initialize the initial sending values
-(void)initialize;
//update the 2 channels (2 table positions) for the joystic
-(void)updateValues: (int)channel :(int)val :(int)channel2 :(int)val2;
///update the channel(table position) for the switch
-(void)updateValues2: (int)channel1 :(int)val1;
//send the data to the specify host (arduino)
-(void)dataSend;
// set ip address
-(void)setIP: (NSString *)ip;
// get ip address
-(NSString *)getIP;
//set port number
-(void)setPortNum:(NSString *)portNum;
//get port number
-(NSString *)getPortNum;

// memory deallocation
-(void)dealloc;

@end
```

Communication.mm

```
/*
 Communication.mm
 MPJOYSTIC

 Copyright (c) 2012 Margarita. All rights reserved.
*/

//IMPORT LIBRARIES
#import "Communication.h"
```

```
//DEFINE CONSTANT VARIABLES

//diameter of the joystic
#define CHANNELS 5

//IMPLEMENTATION
@implementation Communication:NSObject

//SYNTHESIZE VARIABLES
@synthesize udpSocket;
@synthesize tag;
@synthesize host;
@synthesize messageStructure;
@synthesize error;
@synthesize data;
@synthesize portText;
@synthesize port;

//STATIC VARIABLES
static Communication *staticCommunication = nil;
static int onceInitialization=0;

// calling [Communication sharedCommuniation] will return the singleton object
+(Communication *)sharedCommunication
{
    onceInitialization++;
    //the singleton is thread safe
    @synchronized(self) {
        if(!staticCommunication)
        {
            staticCommunication = [[Communication alloc] init];

            // initialize the sending values just one time
            if (onceInitialization==1)
                [staticCommunication initialize];
        }
        return staticCommunication;
    }

}

//returns a Communication object
- (id) init{
    if (self != nil) {

        //create the udp socket
        udpSocket=[[AsyncUdpSocket alloc] initWithDelegate:self];

        //check for errors while binding
        error=nil;
        if (![udpSocket bindToPort:0 error:&error])
            NSLog(@"Error while binding");

        //host
```

```
        host = @"192.168.10.6";
        if ([host length] == 0)
            NSLog(@"error");

        //port - default RCOIP receiver UDP port number is 9048
        portText = @"8888";
        port = [portText intValue];
        if (port <= 0 || port > 65535)
            NSLog(@"Wrong port");
    }
    return self;
}

//send the data to the specify host (arduino)
- (void)dataSend{
    //allocate memory for the send data for udpSocket
    data = [NSMutableData data];

    unsigned char byte;

    byte = (NSInteger)messageStructure.version;
    //append version at the first position of the mutable data as the message structure requires
    [data appendBytes:&byte length:1];

    //append channels
    for (int i=0;i<CHANNELS;i++)
    {
        byte=(NSInteger)messageStructure.channels[i];
        [data appendBytes:&byte length:1];
    }

    //send data to the arduino
    [udpSocket sendData: data toHost:host port:port withTimeout:-1 tag:tag];

    //increase by one the tag every time you send the packet
    tag++;
}

//initialize the initial sending values
-(void) initialize
{
    //version
    messageStructure.version=RC_VERSION;

    //left joystic
    messageStructure.channels[0]=127;
    messageStructure.channels[1]=127;

    //right joystic
    messageStructure.channels[2]=127;
    messageStructure.channels[3]=127;

    //button
    messageStructure.channels[4]=0;
```

```
}

//update the 2 channels (2 table positions) for the joystic
-(void)updateValues: (int)channel1 :(int)val1 :(int)channel2 :(int)val2
{
    NSLog(@"update channel %d with value %d",channel1,val1);
    NSLog(@"update channel %d with value %d",channel2,val2);
    messageStructure.channels[channel1]=val1;
    messageStructure.channels[channel2]=val2;
}

//update the channel(table position) for the switch
-(void)updateValues2 :(int)channel1 :(int)val1
{
    NSLog(@"update channel %d with value %d",channel1,val1);
    messageStructure.channels[channel1]=val1;
}

//set ip address
-(void)setIP: (NSString *)ip
{
    host=ip;
}

//get ip addess
-(NSString *)getIP
{
    return host;
}

//set port number
-(void)setPortNum:(NSString *)portNum
{
    portText = portNum;
    port = [portText intValue];
}

//get port number
-(NSString *)getPortNum
{
    return portText;
}

//memory deallocation
-(void) dealloc
{
    [udpSocket release];
    [host release];
    [error release];
    [data release];
    [super dealloc];
}

@end
```