ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΤΗΛΕΠΟΙΚΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Υλοποίηση ασύρματης συσκευής εισόδου με βάση κλισιόμετρο 2 αξόνων πάνω από την πλατφόρμα Smart-its

Νικόλαος Φραγγογιάννης
Επιβλέπων: κ. Γεώργιος Ευθυβουλίδης
24/10/2007

Περιεχόμενα

Ευχαριστώ τον κ. Λάλη Σπυρίδωνα-Γεράσιμο για την ανάθεση της πτυχιακής μου εργασίας, την ενασχόληση και τη βοήθειά του στην εκπόνηση της, ο οποίος λόγω εκπαιδευτικής άδειας δεν κατέστη εφικτό να παραβρεθεί στην επιτροπή την περίοδο της παρουσίασής της.
Ευχαριστώ τους κ. Ευθυβουλίδη Γεώργιο, κα. Τσομπανοπούλου Παναγιώτα για την βοήθεια στην ολοκλήρωση την πτυχιακής μου εργασίας και την συμμετοχή τους στην εξεταστική επιτροπή.
Τέλος ευχαριστώ τον διδακτορικό υπότροφο Συρίβελη Δημήτριο για την εν γένη συμπαράστασή του και παραγωγικές συζητήσεις.

1 Εισαγωγή

Ο όρος «συσκευή εισόδου» (input device), χρησιμοποιείται για μηχανισμούς που παρέχουν τη δυνατότητα στο χρήστη να δώσει είσοδο-εντολή σε κάποιο σύστημα, επιτρέποντας έτσι την επικοινωνία ανθρώπου-μηχανής (Human - Computer Interaction). Συσκευή εισόδου, λοιπόν, αποτελεί το τηλεχειριστήριο του κλιματιστικού, τα κουμπιά μιας ηλεκτρονικής συσκευής, το ποντίκι ενός ηλεκτρονικού υπολογιστή κ.ο.κ. Μια συσκευή εισόδου χρησιμοποιεί ένα ή περισσότερους αισθητήρες ανάλογα με την λειτουργικότητα που εξυπηρετεί.

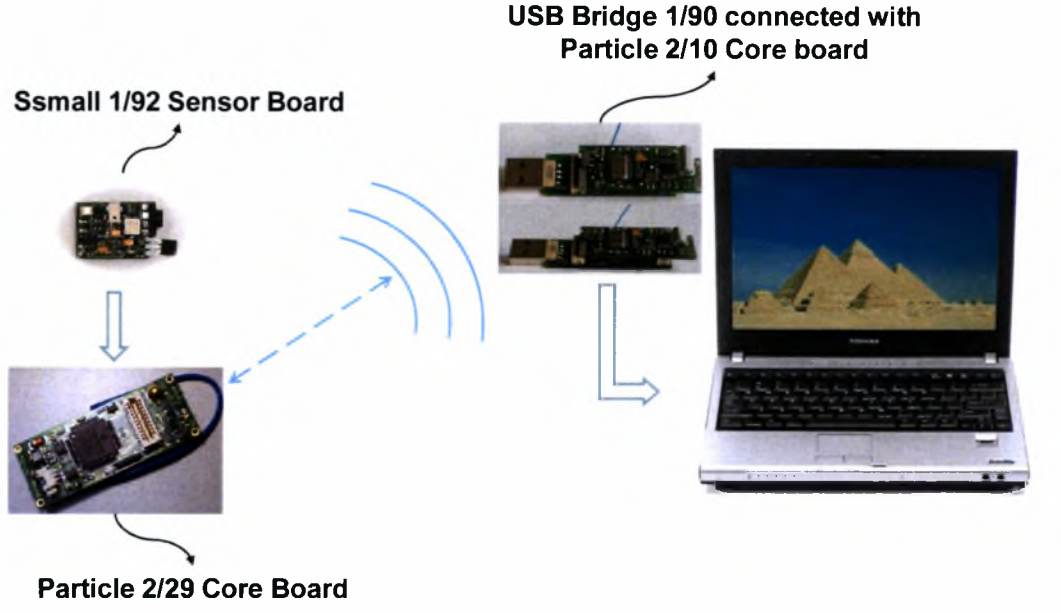
Σε αυτή τη διπλωματική εργασία θα κάνουμε χρήση της πλατφόρμας Smart-its[3] για την ανάπτυξη μιας ασύρματης συσκευής εισόδου, που θα διασφαλίζει παρόμοιες λειτουργικότητες με αυτές ενός συμβατικού mouse. Το wireless motion-sensing Particle airmouse είναι μια ασύρματη συσκευή εισόδου η οποία με την χρήση κλισιόμετρου[1] αντιλαμβάνεται το βαθμό κλίσης του χεριού στο οποίο έχει προσαρτηθεί, αναγνωρίζοντας την κίνησή του σε πραγματικό χρόνο. Η κατεύθυνση και η ταχύτητα του κέρσορα είναι παράμετροι οι οποίοι ορίζονται από το πρόγραμμα οδήγησης της συσκευής.

Το σύστημα παρέχει την δυνατότητα παράλληλης χρήσης της ασύρματης συσκευής εισόδου (wireless accelerometer-based airmouse) με το συμβατό mouse του ηλεκτρονικού υπολογιστή. Το wireless accelerometer-based airmouse είναι εύκολο στην χρήση. Στην περίπτωση που ο χρήστης επιθυμεί να μετακινήσει τον κέρσορα της οθόνης από μακριά, απλά τον ενεργοποιεί και στην συνέχεια γέρνει το χέρι του προς την αντίστοιχη κατεύθυνση.

Σε αυτή την διπλωματική θα περιγράψουμε πως αναπτύξαμε αυτό το πρωτότυπο Human-Computer Interaction (HCI) σύστημα, χρησιμοποιώντας Particle Computers[16].

2 Χαρακτηριστικά του συστήματος και συνδεσμολογία

Η ασύρματη συσκευή εισόδου αποτελείται από ένα Particle Core Board 2/29, συνδεδεμένο με ένα Ssmall Sensor Board 1/92 μέσω του FCI's Conan 21 pin connector. Η επικοινωνία μεταξύ ηλεκτρονικού υπολογιστή και συσκευής εισόδου πραγματοποιείται ασύρματα με την χρήση του Particle USB Bridge που είναι συνδεδεμένο στην USB θύρα του Ηλεκτρονικού Υπολογιστή (Εικόνα 1).



Εικόνα 1: Αρχιτεκτονική του συστήματος και συνδεσμολογία.

Αναλυτικότερα το σύστημα αποτελείται από:

1. μια πλατφόρμα **Core or Communication Boards Particle 2/29** (Εικόνα 1, κάτω αριστερά) με τα εξής χαρακτηριστικά:
   - Processor: PIC 18F6720 at 20 MHz, Internal Memory: 128kbyte program Flash, 4kbyte RAM, 1kbyte EEPROM
   - Additional Memory: 512 kbyte FLASH for data (particle file system)
   - RF Communication through RFM TR1001, 125kbit bandwidth, 868.35 ISM band Europe or 315 MHz ISM Band Japan
   - in circuit programming and debugging with off-the-shelf programmers, over-the-air programming via wireless radio from any Internet connected PC (XBridge or USB-Bridge needed)
   - Programmable in C, OS and Libraries
   - size 45x18 mm (45x27 mm with AAA battery)

2. ένα **Sensor Board ssmall 1/92** (Εικόνα 1, πάνω αριστερά), με τα εξής χαρακτηριστικά:
   - Interfaces to Smart-Its Particle
   - No own processor, controlled by Communication board processor
   - Power supply through main board (e.g. Smart-It) or separate, from 1-3.3V
   - Acceleration Sensor (2 axis, ADXL210 10g max, +/- 40 mg resolution), high responsive (<1ms)
   - Size: 17x22 mm
3. ένα **USB Bridge 1/90** (Εικόνα 1, πάνω κέντρο) συνδεδεμένο με ένα **particle 2/10**, που επιτρέπει ασύρματη επικοινωνία του particle 2/29(αποστολή/λήψη δεδομένων) με τον Ηλεκτρονικό Υπολογιστή (βλέπε Κεφάλαιο 4)
4. και ένα Ηλεκτρονικό Υπολογιστή με λειτουργικό σύστημα Linux gentoo 2.6.19-gentoo-r5.

3 Πρωτόκολλο επικοινωνίας Particle συσκευών - AwareCon

Το AwareCon Network Stack[12] είναι ένα βασικό στοιχείο για την δικτυακή επικοινωνία των Particle συσκευών. Το awareCon είναι ένα τροποποιημένο δικτυακό πρωτόκολλο (customized network protocol) για τις πλατφόρμες Particle που παρέχει ad hoc και synchronized μετάδοση δεδομένων. Παρακάτω φαίνονται τα επίπεδα του AwareCon Network Stack.

AwareCon Network Stack on a Particle Device	
ISO/OSI 7	**Application Layer**
3-4	**Application Convergence Layer (ACL)**
2	**Link Layer (LL)**
1	**RF Layer**

Εικόνα 2: AwareCon protocol layers.

Αναλυτικότερα το πρωτόκολλο αποτελείται από 3 επίπεδα όπως φαίνεται και στην Εικόνα 2:

1. Το ACL layer παρέχει ένα API για την αναπαράσταση των δεδομένων:
   - Η αναπαράσταση των δεδομένων γίνεται με την μορφή tuples.
   - Η πρόσβαση στα δεδομένα γίνεται με συναρτήσεις (αποστολή, λήψη, διάβασμα).
2. Το LL layer χρησιμοποιείται για έλεγχο πρόσβασης (access control), κωδικοποίηση δεδομένων (data encoding) και έλεγχο λαθών (CRC16).
3. Το RF layer χρησιμοποιείται για συγχρονισμό και κωδικοποίηση καναλιού (channel encoding).

3.1 Αναπαράσταση των δεδομένων: ConCom

Το ConCom[13] χρησιμοποιείται από το AwareCon protocol και ορίζει τον τρόπο αναπαράστασης των δεδομένων. Το Concom είναι μια γλώσσα που ορίζει τον τρόπο έκφρασης και ανταλλαγής πληροφορίας[21], όμοια με μια φυσική γλώσσα. Η οργάνωση της πληροφορίας βασίζεται στην χρήση προτάσεων. Μια πρόταση ξεκινάει με ένα *θέμα* (*subject*), που αποτελεί διακριτικό αυτού που την δημιουργεί, ακολουθούμενο από αυθαίρετο αριθμό *χαρακτηριστικών*(*attributes*), χωριζόμενων με κόμμα, που αναφέρονται στο θέμα. Τα δεδομένα του ACL layer του AwareCon οργανώνονται σε tuples με τρόπο που ορίζει η ConCom γλώσσα. Ένα tuple αποτελείται από 2 bytes που καθορίζουν το θέμα του, 1 byte που αφορά το μέγεθος των δεδομένων που περιέχει και n bytes δεδομένων, δηλαδή:

tuple=[TT][L][D]

όπου TT είναι ο 2 bytes τύπος του tuple (Tuple Type), L το μέγεθος των δεδομένων (Length) και D τα δεδομένα (Data).

Τα tuples συνδέονται αλυσιδωτά και τοποθετούνται στον payload buffer του ACL πακέτου. Τα πρώτα δύο bytes κάθε tuple δηλώνουν το θέμα του. Κάθε πακέτο πρέπει να μεταφέρει ένα τουλάχιστον tuple και κάθε tuple ένα θέμα. Ένα tuple για παράδειγμα μπορεί να έχει την εξής μορφή :

[234,128][1][0], η οποία αναλύεται σε:

- [234,128]: tuple με θέμα που αφορά τον αισθητήρα του ήχου (S AU, Sensor Audio),
- [1]: μέγεθος δεδομένων που περιέχει το tuple σε bytes,
- [0]: 0 η τιμή που μέτρησε ο αισθητήρας σε db.

3.2 Δομή AwareCon πακέτου

Η επικοινωνία μεταξύ των συσκευών particle γίνεται με το AwareCon πρωτόκολλο. Η δομή των πακέτων του AwareCon πρωτοκόλλου είναι η εξής: RF Header + LL Header + LL Payload + LL Tail. Αναλυτικότερα:
RF Header:

Synchronization	Reserved bits	Arbitration	PHY header

LL Header:

Version 5	Payload Length (1byte)	FieldStrength (1 byte)	ID (8 bytes)	Sequence number (1 byte)

LL Payload: περιέχει ACL tuples (μέγιστο μέγεθος ενός πακέτου είναι 64 bytes)

Type (high byte)	Type (low byte)	Data Length (1 byte)	Data (length bytes)	...

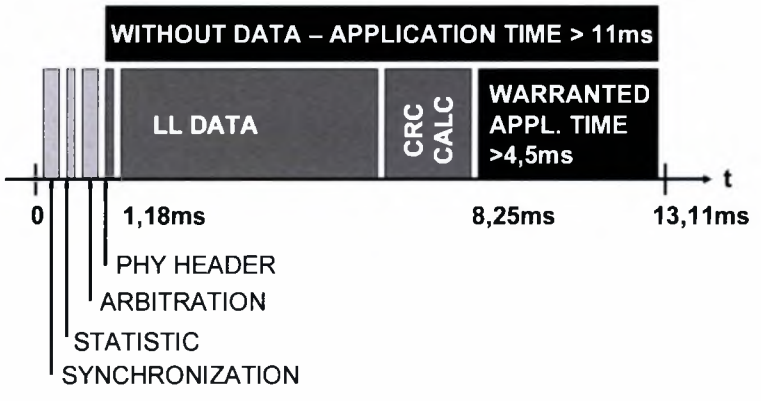
LL Tail:

CRC (high byte)	CRC (low byte)

Το LL Payload περιέχει ACL tuples αναπαριστώντας τα δεδομένα που πρέπει να σταλούν. Τα Header και tail κάθε πακέτου δημιουργούνται αυτόματα από το AwareCon communication stack.

3.3 Χρόνος σύστασης awarecon πακέτου

Η πλατφόρμα Particle χρησιμοποιεί το AwareCon protocol stack. Το AwareCon protocol stack είναι ένα synchronized πρωτόκολλο. Η εφαρμογή που τρέχει στον single task μικροεπεξεργαστή της πλατφόρμας, διακόπτεται περιοδικά κάθε 13ms για να εκτελεστούν εντολές που έχουν να κάνουν με το RF stack: synchronization, data traffic etc.

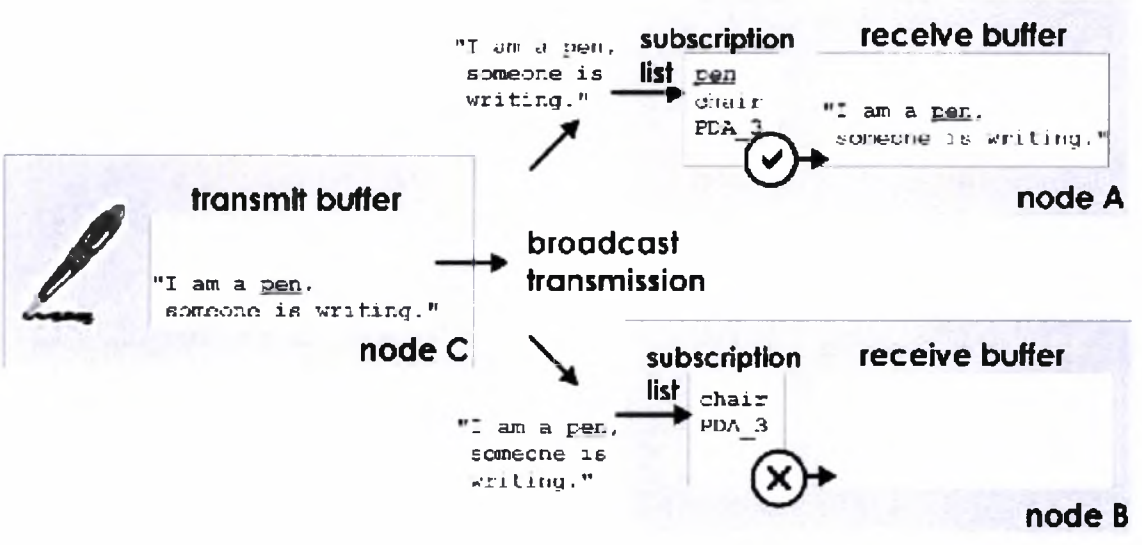


Εικόνα 3: Χρόνος σύσταση πακέτου.

Σε κάθε time slot διάρκειας 13ms στέλνεται μόνο ένα πακέτο. Στην Εικόνα 3 φαίνεται αναλυτικά ο χρόνος που απαιτείται για να εκτελεστούν εντολές που έχουν να κάνουν με το RF stack. Ο χρόνος που απομένει σε ένα slot για μια εφαρμογή που τρέχει στον μικροεπεξεργαστή της πλατφόρμας φαίνεται με το μαύρο χρώμα. Μπορούμε εύκολα να παρατηρήσουμε ότι ο χρόνος που χρειάζεται το σύστημα για την αποστολή ενός πακέτου είναι 8,25ms οπότε απομένουν περίπου 4,5ms για να ολοκληρωθεί η εφαρμογή που τρέχει στην πλατφόρμα. Στην περίπτωση που δεν απαιτείται αποστολή δεδομένων η εφαρμογή έχει στην διάθεσή της περίπου 11ms για να ολοκληρωθεί.

3.4 Broadcast/subscribe

Η επικοινωνία σε ένα ασύρματο δίκτυο από particle computers γίνεται με την χρήση του AwareCon protocol, βασίζεται στον μηχανισμό broadcast-and- subscribe (Εικόνα 4) και είναι connectionless. Κάθε μήνυμα εκπέμπεται με broadcast χωρίς διεύθυνση παραλήπτη. Κάθε ενεργός δέκτης ακούει όλα τα πακέτα, παραλαμβάνοντας μόνο αυτά τα οποία τον ενδιαφέρουν. Ουσιαστικά ο δέκτης φιλτράρει κάθε broadcast πακέτο που λαμβάνει με βάση μια τοπική λίστα (subscription list) που διατηρεί. Σε αυτό το φιλτράρισμα συντελεί και η δομή μιας ConCom πρότασης (Εικόνα 4). Το μέγεθος της λίστας αυτής περιορίζεται στα 7 θέματα (Particle AwareCon version 5). Τα πακέτα που περιέχουν θέματα (subjects) τα οποία υπάρχουν στην τοπική λίστα (subscription list) του δέκτη παραλαμβάνονται και επεξεργάζονται. Όλα τα υπόλοιπα απορρίπτονται και δεν παραδίδονται στην εφαρμογή του δέκτη.



Εικόνα 4: Μηχανισμός Broadcast/Subscribe.

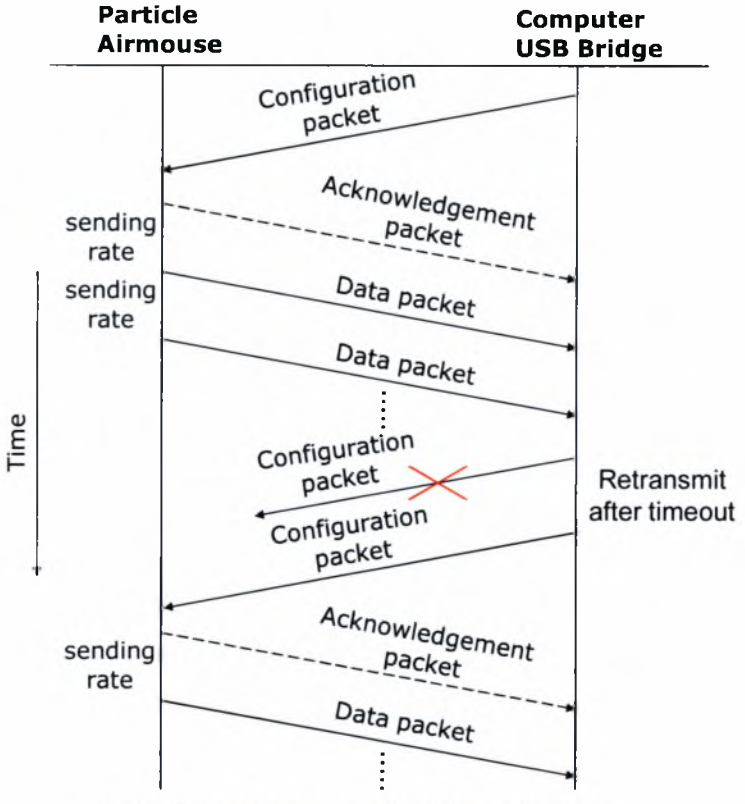
4 Επικοινωνία συσκευής εισόδου airmouse – Ηλεκτρονικού Υπολογιστή

Η επικοινωνία του airmouse αισθητήρα και του προγράμματος οδήγησης που τρέχει στον ηλεκτρονικό υπολογιστή είναι ασύρματη, γίνεται μέσω του USB Bridge και βασίζεται στο πρωτόκολλο AwareCon. Η διαδικασία είναι απλή και φαίνεται στην Εικόνα 5. Το πρόγραμμα οδήγησης αρχικά στέλνει ένα πακέτο configuration, ορίζοντας τον ρυθμό αποστολής των δεδομένων. Το airmouse απαντά με ένα πακέτο acknowledgement, πιστοποιώντας ότι αρχικοποιήθηκε με αυτή την νέα παράμετρο και ξεκινά να στέλνει τις μετρήσεις του αισθητήρα του περιοδικά. Στην περίπτωση που το πρόγραμμα οδήγησης δεν λάβει την απάντηση μέσα σε κάποιο χρονικό διάστημα τότε ξαναστέλνει το configuration πακέτο. Οποιαδήποτε χρονική στιγμή το πρόγραμμα οδήγησης μπορεί να αλλάξει τον ρυθμό αποστολής των δεδομένων του airmouse στέλνοντας πάλι ένα πακέτο reconfiguration, οπότε και η παραπάνω διαδικασία επαναλαμβάνεται από την αρχή. Το configuration πακέτο αποτελείται από δύο ACL tuples με τα ακόλουθα subject:

- CAC Address Control Field (102, 64): για το handshake μεταξύ airmouse – driver και την αποστολή πακέτου acknowledgement.
- CRS Control Remote Sensor (204, 232): για το configuration του airmouse, καθορίζοντας τον ρυθμό αποστολής των δεδομένων του airmouse.

Κάθε data πακέτο αποτελείται από δύο ACL tuples με τα ακόλουθα subjects:

- SGX Sensor GravityX (234,128): για την 16bit τιμή της επιτάχυνσης της βαρύτητας ως προς τον άξονα X.
- SGY Sensor GravityY (240,192): για την 16bit τιμή της επιτάχυνσης της βαρύτητας ως προς τον άξονα Y.



Εικόνα 5: Σενάριο επικοινωνίας driver – airmouse.

5 Επικοινωνία προγράμματος οδήγησης - Xserver

Η επικοινωνία του προγράμματος οδήγησης του airmouse με τον Xserver[19] βασίζεται στο IMPS/2 πρωτόκολλο και στην χρήση ενός named pipe(fifo)[20][9], ενός IPC μηχανισμού που επιτρέπει σε διαφορετικές διεργασίες να επικοινωνούν μεταξύ τους. Η διαδικασία ακολουθεί τα παρακάτω στάδια:

1. Το Λειτουργικό Σύστημα κατά την εκκίνηση του δημιουργεί μια named pipe (fifo) εκτελώντας ένα script που περιέχει την παρακάτω εντολή: mkfifo -m 666 /dev/mouse.
2. Το πρόγραμμα οδήγησης διαβάζει πακέτα από το USB Bridge και αφού τα επεξεργαστεί:

   2.1 τα μετατρέπει σε αντίστοιχα IntelliMouse-compatible πακέτα και

2.2 τα γράφει στο named pipe /dev/mouse.

3. Ο Xserver που έχει τροποποιηθεί[8] ώστε να χρησιμοποιεί το IMPS/2 πρωτόκολλο, διαβάζει κάθε φορά τα πακέτα από το named pipe και ζωγραφίζει την κίνηση του κέρσορα στην οθόνη.

5.1 Δομή IntelliMouse πακέτου

Η δομή ενός IntelliMouse[7] πακέτου φαίνεται στην Εικόνα 6. Τα LB, MB, RB είναι το αριστερό, δεξί και μεσαίο κουμπί του ποντικιού αντίστοιχα. Τιμή 1 υποδεικνύει κλικ του αντίστοιχου κουμπιού. Επειδή το πρόγραμμα οδήγησης υποστηρίζει μόνο δεξί κλικ οι τιμές των M, R bit είναι πάντα 0. Όσον αφορά το πρόσημο στις τιμές του X και Y, θετική τιμή του X (Xsign = 0) σημαίνει κίνηση προς το δεξί μέρος της οθόνης και θετική τιμή του Y (Ysign = 0) σημαίνει κίνηση προς το πάνω μέρος της οθόνης. Το 4ο byte κάθε πακέτου είναι 0 δεδομένου του ότι ο οδηγός δεν υποστηρίζει κάθετη περιήγηση (vertical scrolling). Το δεύτερο και τρίτο byte καθορίζονται από τον airmouse οδηγό, ύστερα από επεξεργασία των τιμών που έχουν ληφθεί από τον αισθητήρα, αναπαριστώντας την μετατόπιση του κέρσορα της οθόνης σε σχέση με προηγούμενη μετατόπιση.

Byte	D7	D6	D5	D4	D4	D2	D1	D0	Comment
1	0	0	Ysign	Xsign	1	MB	RB	LB	X/Y signs and R/L/M Buttons
2	X7	X6	X5	X4	X3	X2	X1	X0	X data byte
3	Y7	Y6	Y5	X4	Y3	Y2	Y1	Y0	Y data byte
4	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Z/wheel data byte

Εικόνα 6: Δομή IntelliMouse πακέτου.

Σημασιολογία συμβόλων:

LB: Κατάσταση του αριστερού κουμπιού. Pressed = 1.

RB: Κατάσταση του δεξιού κουμπιού. Pressed = 1.

MB: Κατάσταση του μεσαίου κουμπιού. Pressed = 1.

X7-X0: 8-bit προσημασμένος συμπληρώματος ως προς δυο ακέραιος(signed twos complement integer) με Xsing το bit που υποδηλώνει το πρόσημο, που αναπαριστά τη σχετική μετατόπιση του κέρσορα στον άξονα των X μετά την τελευταία μετάδοση δεδομένων. Θετική τιμή αποτελεί ένδειξη κίνησης προς τα δεξιά, αρνητική τιμή αποτελεί ένδειξή κίνησης προς τα αριστερά.

Y7-Y0: Ομοίως όπως για τα X7-X0. Θετική τιμή αποτελεί ένδειξη κίνησης προς τα επάνω, αρνητική τιμή αποτελεί ένδειξή κίνησης προς τα κάτω.

Z2-Z0: Z-wheel κίνηση. 3-bit προσημασμένος συμπληρώματος ως προς δυο ακέραιος(signed twos complement integer) με Z3 το bit που υποδηλώνει το πρόσημο. Χρησιμοποιείται για κάθετη περιήγηση των περιεχομένων της οθόνης(vertical scrolling).

Z7-Z4: Bits που παίρνουν ίδιες τιμές με το Z3. Χρησιμοποιούνται για θέματα συμβατότητας με παλαιότερους drivers.

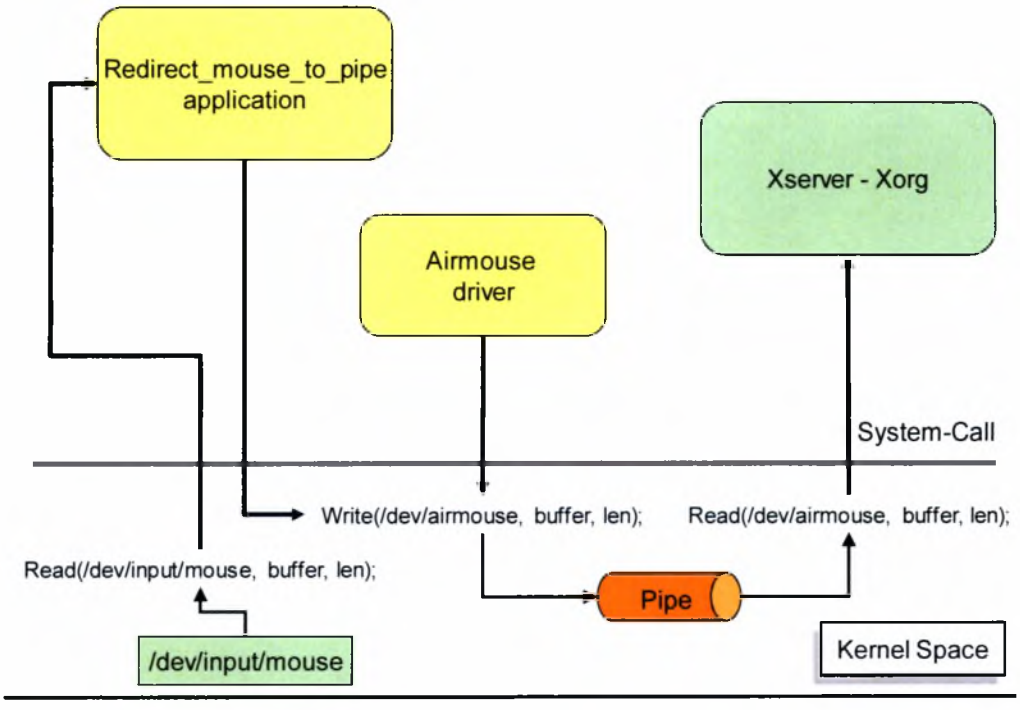
Το σύστημα υποστηρίζει την παράλληλη χρήση του airmouse με οποιοδήποτε IntelliMouse συσκευή εισόδου που χρησιμοποιεί το IMPS/2[5] πρωτόκολλο. Αυτό επιτυγχάνεται με την χρήση της εφαρμογής redirect_mouse_to_pipe, η οποία διαβάζει τα πακέτα του συμβατικού mouse driver από το special device /dev/input/mice και τα γράφει στο named pipe /dev/airmouse. Ο Xserver

διαβάζει τα δεδομένα από το named pipe fifo, τα οποία χρησιμοποιεί για να σχεδιάσει την κίνηση του κέρσορα στην οθόνη. Για να επιτευχθεί κάτι τέτοιο θα πρέπει να τροποποιηθεί το configuration file /etc/X11/xorg.conf του Xserver έτσι ώστε να υποστηρίζει το πρωτόκολλο IMPS/2 καθώς και να χρησιμοποιεί ως συσκευή εισόδου το named pipe /dev/airmouse. Έτσι το αρχείο /etc/X11/xorg.conf θα πρέπει να τροποποιηθεί έτσι ώστε να περιέχει τις παρακάτω γραμμές (Εικόνα 7).

```
Section "InputDevice"
Identifier "Mouse0"
Driver     "mouse"
Option     "Protocol" "IMPS/2"
Option     "Device" "/dev/airmouse"
EndSection
```

Εικόνα 7: configuration αρχείο του Xserver.

Στο Εικόνα 8 φαίνεται η πρόσβαση των τριών διεργασιών του συστήματος στο named pipe – fifo. Διευκρινίζουμε ότι δεν υπάρχουν προβλήματα συγχρονισμού στις πράξεις της ανάγνωσης και εγγραφής δεδομένων από και προς το named pipe, λόγω του ότι το named piped είναι ένας kernel space IPC μηχανισμός που παρέχει προστασία σε τέτοια θέματα.



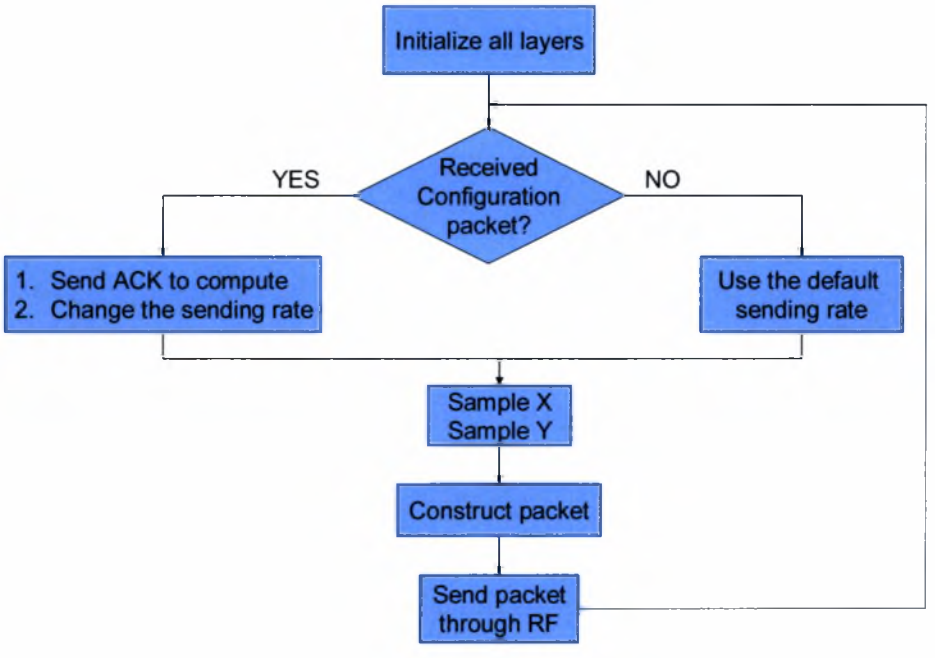
Εικόνα 8: Επικοινωνία διεργασιών του συστήματος.

6 Σχεδίαση του λογισμικού του airmouse

Υπάρχουν δυο υλοποιήσεις όσον αφορά το λογισμικό που τρέχει στο airmouse με μια και μόνο σημαντική διαφορά: τον τρόπο αποστολής των δεδομένων του αισθητήρα.

6.1 Πρώτη υλοποίηση airmouse λογισμικού

Κατά την εκκίνησή του, το airmouse περιμένει ένα configuration πακέτο από το πρόγραμμα οδήγησης που θα ορίζει τον ρυθμό αποστολής των μετρήσεων του αισθητήρα. Το σύστημα υποστηρίζει και on the fly reconfiguration, χωρίς να χρειάζεται να γίνει επανεκκίνηση. Όταν ληφθεί ένα reconfiguration πακέτο γίνεται αλλαγή του ρυθμού αποστολή των μετρήσεων. Σε αυτήν την υλοποίηση το airmouse δεν κάνει κανένα έλεγχο στις τιμές που παίρνει από τον αισθητήρα κατά την δειγματοληψία του και στέλνει τα πακέτα με το ρυθμό αποστολής που έχει αρχικοποιηθεί από τον χρήστη. Στο παρακάτω διάγραμμα ροής (Εικόνα 9) φαίνεται ο τρόπος με τον οποίο γίνεται η διαδικασία αποστολής των μετρήσεων του αισθητήρα του airmouse.



Εικόνα 9: Διάγραμμα ροής της πρώτης υλοποίησης του λογισμικού του airmouse.

6.1.1 On the fly reconfiguration

Ο όρος on the fly reconfiguration αναφέρεται στον μηχανισμό της αλλαγής του ρυθμού αποστολής των μετρήσεων του αισθητήρα του airmouse κατά την διάρκεια λειτουργίας του, χωρίς να είναι αναγκαία η επανεκκίνηση της λειτουργίας του. Η υλοποίηση του on the fly airmouse reconfiguration βασίζεται στην χρήση της συνάρτησης void SlotEndCallBack(), η οποία καλείται στο τέλος κάθε slot. Αξίζει να σημειωθεί ότι η παραπάνω συνάρτηση θα πρέπει να επιστρέφει πριν περάσουν τα 13ms! Με την βοήθεια αυτής της συνάρτησης στο τέλος κάθε slot γίνεται έλεγχος αν έχει σταλεί από το πρόγραμμα οδήγησης reconfiguration πακέτο, που ορίζει το νέο ρυθμό

αποστολής των μετρήσεων του αισθητήρα, οπότε και αρχικοποιείται το σύστημα με αυτή την νέα παράμετρο.

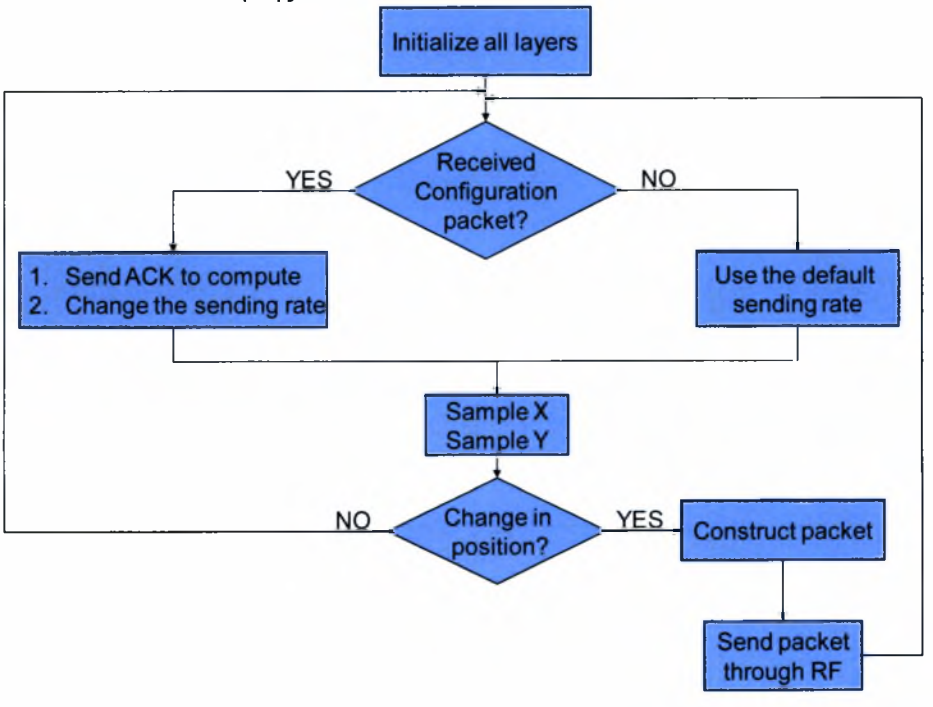
6.2 Δεύτερη υλοποίηση airmouse λογισμικού – state_machine based υλοποίηση

Η διαφορά της state_machine-based υλοποίησης από την πρώτη έγκειται στον έλεγχο των τιμών του αισθητήρα (accelerometer sensor) προτού αποσταλούν. Το λογισμικό υποστηρίζει διακριτές καταστάσεις με αποτέλεσμα το airmouse να στέλνει το πακέτο με τις μετρήσεις μόνο όταν διαπιστώσει αλλαγή στην θέση του. Έτσι κάθε φορά πριν γίνει αποστολή και μάλιστα πριν καν αρχίσει να γίνεται η διαδικασία τοποθέτησης των τιμών του αισθητήρα σε πακέτο, ξεκινά μια διαδικασία σύγκρισης της τρέχουσας μέτρησης, τόσο του άξονα x όσο και του άξονα y, με την προηγούμενη έτσι ώστε μόνο όταν διαπιστωθεί διαφορά τότε αρχίζει η αποστολή. Σε αυτή την υλοποίηση το airmouse δεν στέλνει συνεχόμενα πακέτα με μετρήσεις, αλλά κάθε πακέτο που στέλνει υποδηλώνει έμμεσα την αλλαγή της θέσης του. Συνεπώς στην δεύτερη υλοποίηση το airmouse προσαρτίζεται με την «ικανότητα»:

- να θυμάται προηγούμενες καταστάσεις,
- να μπορεί να συγκρίνει την τρέχουσα κατάστασή του με την προηγούμενη
- και να στέλνει αντίστοιχο πακέτο, μόνο όταν διαπιστώσει αλλαγή στην θέση του,

εξοικονομώντας μικρότερη χρήση του RF.

Με αυτόν τον τρόπο γίνεται σαφώς καλύτερη διαχείριση της ενέργειας που καταναλώνει η συσκευή κατά την διάρκεια λειτουργίας της, έχοντας ως άμεσο αποτέλεσμα την αύξηση της διάρκειας ζωής της μπαταρίας. Στο παρακάτω διάγραμμα ροής (Εικόνα 10) φαίνεται η εκτέλεση της state_machine based υλοποίησης στο airmouse.



Εικόνα 10: Διάγραμμα ροής state_machine_based υλοποίησης.

6.2.1 Αντιστοίχηση των τιμών του αισθητήρα σε γωνίες κλίσης της θέσης του

Πρακτικά η state_machine-based υλοποίηση κάνει χρήση 14 διακριτών καταστάσεων: 7 για τον άξονα X (Εικόνα 12) και 7 για τον άξονα Y. Από τις 7 καταστάσεις που αντιστοιχούν στον X οι 4 χρησιμοποιούνται για την αριστερή κίνηση και αναφέρονται στην κλίση του airmouse στο επίπεδο (0^0, 30^0, 45^0, 60^0 , αντίστοιχα) ενώ οι άλλες 3 χρησιμοποιούνται για την δεξιά κίνηση και αναφέρονται στην κλίση των -30^0, -45^0, -60^0 αντίστοιχα. Ομοίως από τις 7 καταστάσεις του Y οι 4 χρησιμοποιούνται για την κίνηση προς τα πάνω και αναφέρονται στην κλίση του airmouse στο επίπεδο (0^0, 30^0, 45^0, 60^0 , αντίστοιχα) ενώ οι υπόλοιπες 4 χρησιμοποιούνται για την κίνηση προς τα κάτω και αναφέρονται στην κλίση των -30^0, -45^0, -60^0 αντίστοιχα. Σε κάθε κατάσταση αντιστοιχεί κάποιο πλήθος τιμών του αισθητήρα. Κάθε κατάσταση έχει ένα μέγιστο και ένα ελάχιστο κατώφλι περιέχοντας όλες τις ενδιάμεσες τιμές. Η εύρεση των κατωφλίων κάθε κατάστασης βασίζεται σε δυο παραμέτρους:

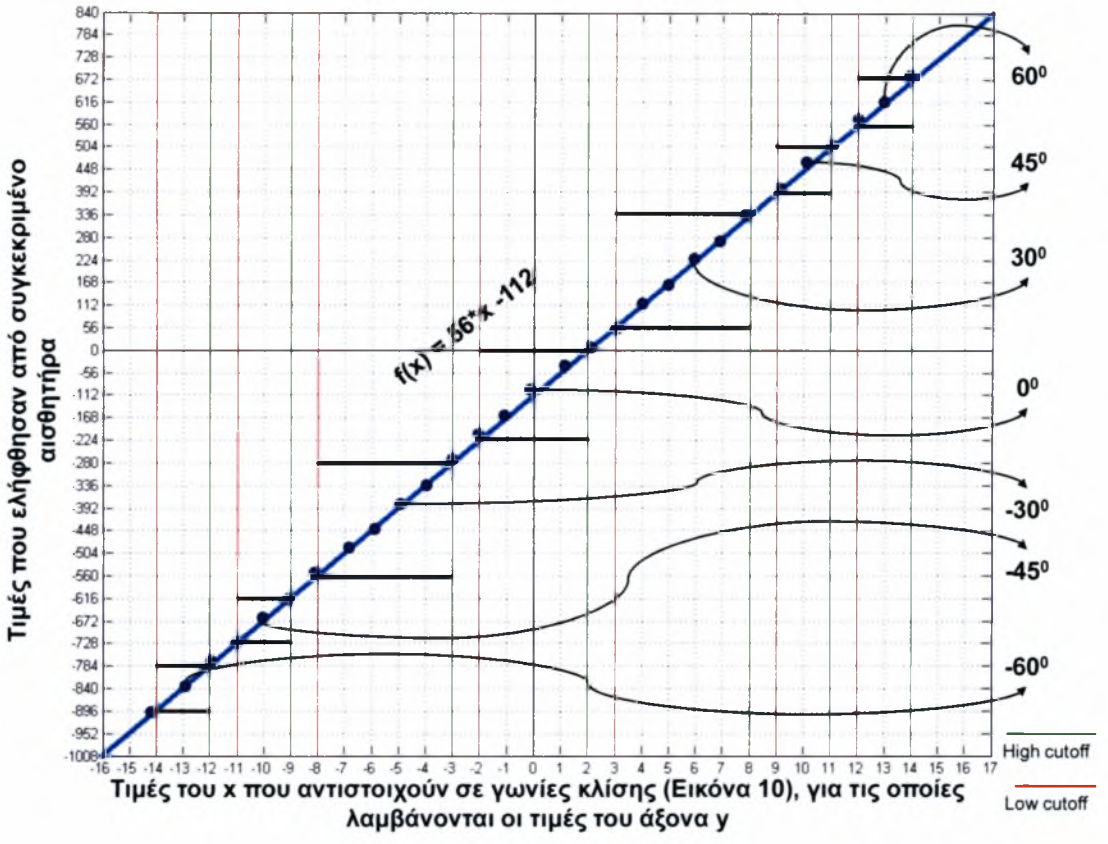
- στην ευαισθησία του αισθητήρα (sensor sensitivity)
- στην τιμή a , b για τον άξονα x, y που επιστρέφει ο αισθητήρας όταν βρεθεί σε οριζόντια θέση, κατά την οποία δεν υπάρχει κίνηση του κέρσορα στην οθόνη. Η τιμή αυτή συμβάλει στον υπολογισμό του μέγιστου και ελάχιστου κατωφλίου της κατάστασης που αναφέρεται σε γωνιακή κλίση 0^0 μοιρών.

Η γραμμική συνάρτηση για την αντιστοίχηση των τιμών του αισθητήρα σε γωνιακή κλίση της θέσης του, για τον άξονα x είναι η $f_x(k) = k*s+a$ και για τον άξονα y είναι η $f_y(k) = k*s+b$. Όπου s η ευαισθησία του αισθητήρα, $k \in Z \cap [-14, 14]$, a η τιμή του αισθητήρα για τον x – άξονα σε οριζόντια θέση και b η τιμή του αισθητήρα για τον y – άξονα σε οριζόντια θέση. Το μέγιστο και το ελάχιστο κατώφλι κάθε κατάστασης προσδιορίζονται από τις τιμές που επιστρέφουν οι συναρτήσεις $f_x(s)$ και $f_y(s)$ για τους x, y άξονες, για συγκεκριμένες τιμές του k. Στον παρακάτω πίνακα (Εικόνα 11) φαίνεται η αντιστοίχηση των κατωφλίων κάθε κατάστασης για συγκεκριμένες τιμές του k.

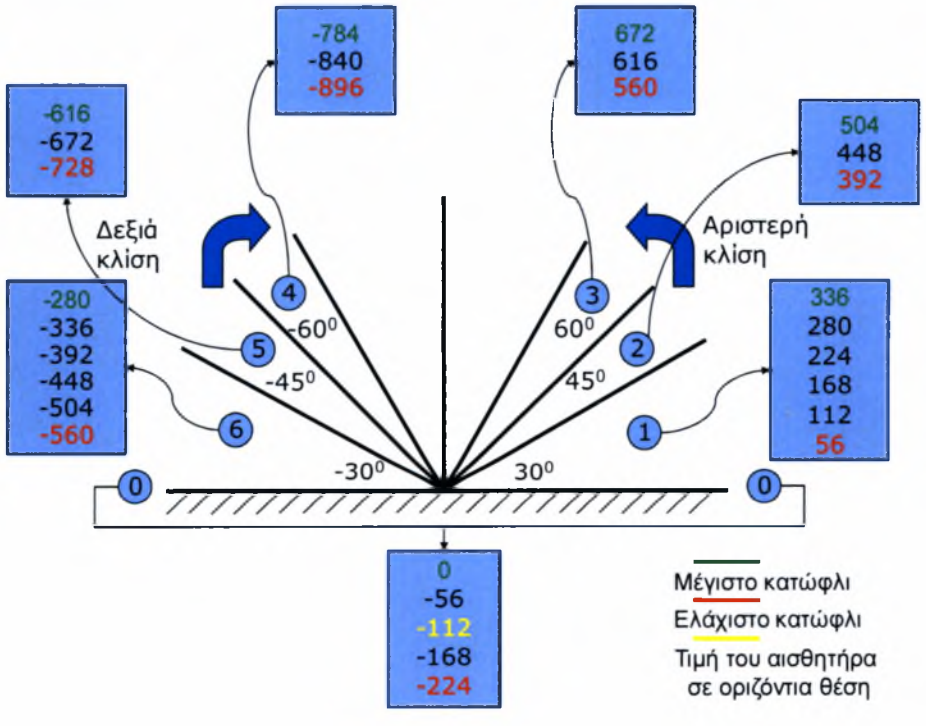
Κατάσταση	x-axis
0^0	High_cutoff_0^0 → 2
	Low_cutoff_0^0 → -2
30^0	High_cutoff_30^0 → 8
	Low_cutoff_30^0 → 3
-30^0	High_cutoff_-30^0 → -3
	Low_cutoff_-30^0 → -8
45^0	High_cutoff_45^0 → 11
	Low_cutoff_45^0 → 9
-45^0	High_cutoff_-45^0 → -9
	Low_cutoff_-45^0 → -11
60^0	High_cutoff_60^0 → 14
	Low_cutoff_45^0 → 12
-60^0	High_cutoff_-60^0 → -12
	Low_cutoff_-45^0 → -14

Εικόνα 11: Αντιστοίχηση καταστάσεων και cutoffs.

Στην παρακάτω γραφική παράσταση (Εικόνα 12) τα μπλε σημεία αναπαριστούν την τιμή που δίνει ο αισθητήρας (άξονας y) όταν βρεθεί σε συγκεκριμένη θέση για διακριτές τιμές του x. Στην Εικόνα 12 και Εικόνα 11 φαίνεται η αντιστοίχηση των τιμών του αισθητήρα σε γωνιακή κλίση της θέσης του. Ο αισθητήρας έχει ευαισθησία s = 56 mg και σε οριζόντια θέση δίνει a = -112 mg.



Εικόνα 12: Γραφική παράσταση συνάρτησης αντιστοίχησης των τιμών του αισθητήρα σε γωνιακή κλίση της θέσης του.



Εικόνα 13: Αντιστοίχηση τιμών αισθητήρα σε καταστάσεις γωνιακής κλίσης της θέσης του.

Αυτή η προσέγγιση χρήσης των συγκεκριμένων καταστάσεων εξυπηρετεί τρεις σκοπούς:

- Ο πρώτος σκοπός βασίζεται στην μεγάλη ευαισθησία του αισθητήρα που με την παραμικρή κίνησή του προκαλείται ανεπιθύμητη κίνηση στον κέρσορα της οθόνης. Με την χρήση

αυτών των καταστάσεων το σύστημα γίνεται πλέον ανεκτικό στο τρεμούλιασμα του χεριού του χρήστη με αποτέλεσμα η χρήση του airmouse γίνεται πολύ πιο εύκολη.

- Ο δεύτερος σκοπός έχει να κάνει με το πλήθος των δεδομένων που στέλνει ο αισθητήρας και συνεπώς με τη χρήση του RF: όσο λιγότερες οι καταστάσεις, όσο δηλαδή μικρότερο το εύρος κινήσεων του αισθητήρα, τόσο μικρότερο το πλήθος των αποσταλμένων πακέτων και συνεπώς τόσο μικρότερη η χρήση του RF και αντίστροφα όσο περισσότερες οι καταστάσεις τόσο μεγαλύτερο το πλήθος των αποσταλμένων πακέτων και συνεπώς τόσο μεγαλύτερη η χρήση του RF. Στην χειρότερη περίπτωση που κάθε κατάσταση περιλαμβάνει μια τιμή του αισθητήρα συνεπάγεται ότι βρισκόμαστε στην πρώτη υλοποίηση.
- Ο τρίτος σκοπός έχει να κάνει με τον ίδιο το χρήστη και την ικανότητά του λόγω συνήθειας να αντιλαμβάνεται ευκολότερα τις 30^0, 45^0 και 60^0 μοίρες από άλλες γωνίες κλίσης.

6.2.2 Μειονεκτήματα και επέκταση δεύτερης υλοποίησης

Η δεύτερη υλοποίηση υστερεί της πρώτης υλοποίησης σε ένα βασικό σημείο: στη διαχείριση απώλειας πακέτων. Στην πρώτη υλοποίηση το ενδεχόμενο απώλειας πακέτου δεν επηρεάζει άμεσα την κίνηση του κέρσορα στην οθόνη, λόγω του ότι το airmouse τροφοδοτεί συνεχώς με πακέτα πληροφορίας της θέσης το πρόγραμμα οδήγησης. Η απώλεια πακέτου κρίνεται αμελητέα μπροστά στην συνεχόμενη και με μεγάλο ρυθμό αποστολή δεδομένων.

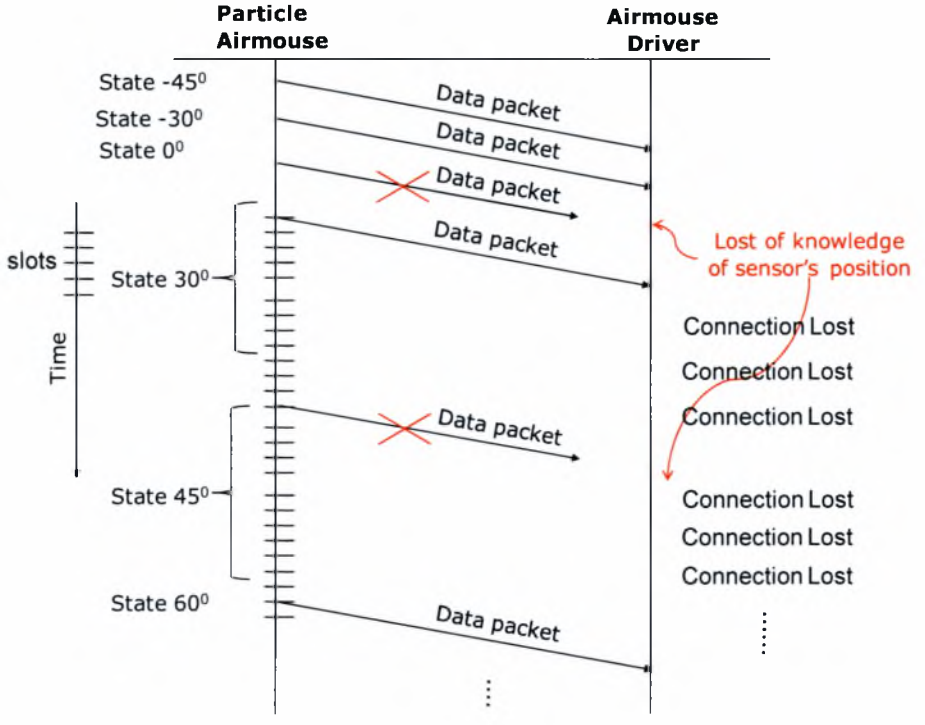
Τι συμβαίνει στην δεύτερη υλοποίηση σε περίπτωση απώλειας πακέτου; Το μειονέκτημα της δεύτερης υλοποίησης είναι ότι η αποστολή πακέτου γίνεται μόνο μια φορά και μόνο όταν παρατηρηθεί αλλαγή της θέσης του αισθητήρα. Αυτή η υλοποίηση καταδικάζει το πρόγραμμα οδήγησης σε απώλεια γνώσης της θέσης του αισθητήρα όταν το αντίστοιχο πακέτο χαθεί. Σε ένα τέτοιο σενάριο και για όσο διάστημα ο αισθητήρας βρίσκεται στην ίδια θέση, κάτι που συνεπάγεται ότι δεν στέλνει πλέον άλλα πακέτα, το πρόγραμμα οδήγησης θα έχει την λανθασμένη εντύπωση για την θέση του airsmouse ότι ο αισθητήρας βρίσκεται σε προηγούμενη κατάσταση κατά την οποία και είχε λάβει αντίστοιχο πακέτο. Αυτό το μειονέκτημα οφείλεται στο γεγονός ότι κάθε πακέτο στέλνεται αφενός μια φορά και μόνο όταν αυτό κριθεί απαραίτητο, αφετέρου στο ότι η αποστολή και λήψη του αποκτά μεγάλη σημασία για το πρόγραμμα οδήγησης.

Ένα άλλο ζήτημα που προκύπτει με την δεύτερη υλοποίηση έχει να κάνει με την αναγνώριση απώλειας σύνδεσης μεταξύ οδηγού και airmouse. Στην πρώτη υλοποίηση ενδεχόμενη απώλεια σύνδεσης γίνεται άμεσα αντιληπτή από το λογισμικό του προγράμματος οδήγησης μόλις διαπιστωθεί μεγάλη καθυστέρηση στην λήψη πακέτων. Ένα τέτοιο ενδεχόμενο δεν είναι προφανές στην δεύτερη υλοποίηση και αυτό γιατί η καθυστέρηση στην λήψη πακέτου μπορεί να οφείλεται σε δύο αιτίες:

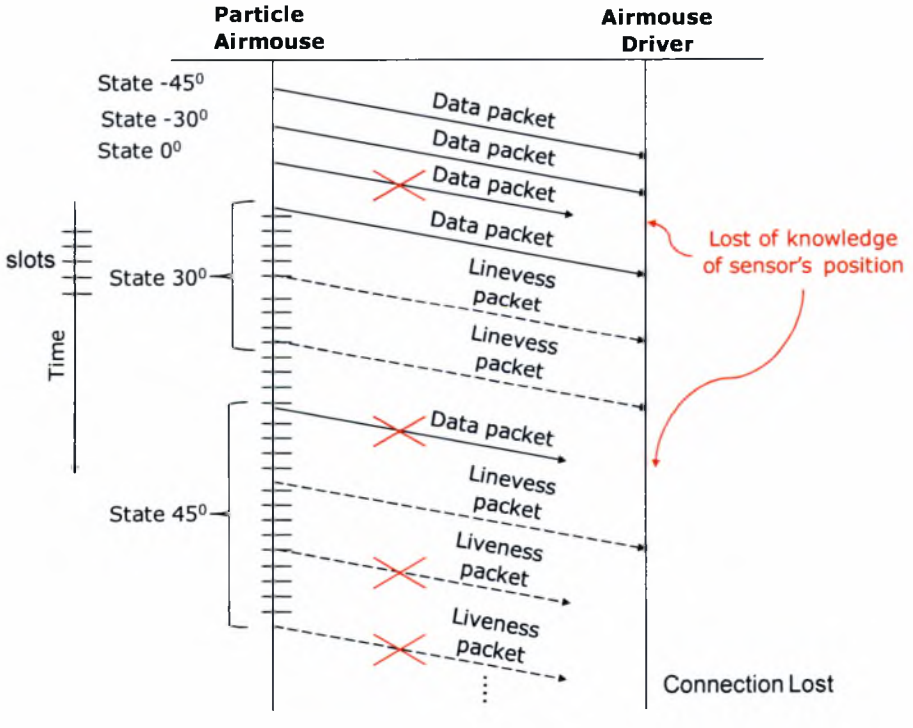
- ο αισθητήρας βρίσκεται στην ίδια θέση για χρονικό διάστημα ίσο με το χρόνο καθυστέρησης λήψης πακέτου από το πρόγραμμα οδήγησης
- απώλεια ασύρματης σύνδεσης (connection lost) μεταξύ οδηγού – airmouse λόγω τεχνικού προβλήματος (βλάβη του airmouse, συνεχόμενη απώλεια πακέτων κ.α.).

Η λύση που προτείνεται στο πρόβλημα της αναγνώρισης απώλειας σύνδεσης μεταξύ οδηγού και airmouse έχει να κάνει με την χρήση liveness πακέτων. Τα liveness πακέτα είναι data πακέτα, όπως έχουν περιγράφει και παραπάνω, με την διαφορά ότι στέλνονται περιοδικά και για όσο χρονικό διάστημα ο αισθητήρας βρίσκεται στην ίδια κατάσταση. Η περίοδος αποστολής των liveness πακέτων είναι συνάρτηση του ρυθμού αποστολής των data πακέτων. Έτσι αν το airmouse στέλνει τις μετρήσεις του αισθητήρα κάθε x ms τότε τα liveness πακέτα αποστέλλονται κάθε α*x ms, όπου α=5, 6, 7, ... Με αυτόν τον τρόπο το airmouse επιβεβαιώνει την ύπαρξη σύνδεσης ακόμα και αν

βρεθεί ακίνητο. Παρακάτω (Εικόνα 14, Εικόνα 15) φαίνεται ένα σενάριο επικοινωνίας μεταξύ προγράμματος οδήγησης – airmouse (state_machine_based υλοποίηση). Παρατηρείστε (Εικόνα 14) ότι με την χρήση των liveness πακέτων ο οδηγός γνωρίζει την κατάσταση του airmouse. Σε ενδεχόμενη απώλεια liveness πακέτων ο οδηγός αναγνωρίζει απώλεια σύνδεσης (connection lost).



Εικόνα 14: Σενάριο επικοινωνίας driver - airmouse χωρίς την χρήση liveness πακέτων.



Εικόνα 15: Σενάριο επικοινωνίας driver - airmouse με χρήση liveness πακέτων.

6.3 Μετρήσεις κατανάλωσης ενέργειας

Πραγματοποιήσαμε ένα αριθμό πειραμάτων για να υπολογίσουμε την κατανάλωση ενέργειας του airmouse για τις διαφορετικές υλοποιήσεις του λογισμικού του, μετρώντας την μέση ισχύ που απαιτεί το σύστημα για διαφορετικά σενάρια χρήσης. Για τις μετρήσεις χρησιμοποιήσαμε τροφοδοτικό σταθερής τάσης στα 1.2V και ένα αμπερόμετρο. Στις μετρήσεις μας περιελάβαμε 3 διαφορετικά σενάρια:

1. Πρώτη υλοποίηση του airmouse με συνεχόμενα περιοδική αποστολή πακέτων.
2. Δεύτερη state_machine_based υλοποίηση με συνεχόμενη αλλαγή θέσης.
3. Δεύτερη state_machine_based υλοποίηση με ακινητοποιημένη θέση στο σημείο αναφοράς (οριζόντια θέση) .

Οι μετρήσεις κάθε σεναρίου έγιναν για τους διαφορετικούς ρυθμούς αποστολής των 13, 26 και 52ms (Πίνακας 1). Παρουσιάζουμε 5 διαφορετικές μετρήσεις (διάρκειας 1' για κάθε σενάριο).

Sending_period (ms)	First_implementation Full speed (mW)	State_machine_based Full move (mW)	State_machine_based Idle (mW)
13	98.28	92.52	81.84
26	91.32	88.08	81.12
52	85.32	82.80	77.40

Πίνακας 1: Μέσος όρος απαιτούμενης ισχύος για διαφορετικά σενάρια χρήσης.

Υπολογίσαμε το κέρδος απαιτούμενης ισχύς για την state_machine_based υλοποίηση στην περίπτωση συνεχόμενης κίνησης του airmouse (Πίνακας 2), σε σχέση με την πρώτη υλοποίηση του airmouse με συνεχόμενα περιοδική αποστολή πακέτων . Όμοια υπολογίσαμε το κέρδος ισχύς της state_machine_based υλοποίησης στην περίπτωση ακινητοποίησης του airmouse στην θέση αναφοράς (Πίνακας 3), σε σχέση με την πρώτη υλοποίηση. Από τα αποτελέσματα παρατηρούμε ότι στην δεύτερη state_machine_based υλοποίηση γίνεται σαφώς μεγαλύτερη εξοικονόμηση ενέργειας.

Ρυθμός αποστολής	Κέρδος
13ms	5,8%
26ms	3,54%
52ms	2,95%

Πίνακας 2: Κέρδος σε απαιτούμενη ισχύς της state_machine_based υλοποίησης με συνεχόμενη αλλαγή της θέσης του airmouse σε σχέση με την πρώτη υλοποίηση(συνεχόμενα περιοδική αποστολή πακέτων) του λογισμικού του airmouse.

Ρυθμός αποστολής	Κέρδος
13ms	16,72%
26ms	11,16%
52ms	9,28%

Πίνακας 3: Κέρδος σε απαιτούμενη ισχύς της state_machine_based υλοποίησης με ακινητοποίηση της θέσης του airmouse στην θέση αναφοράς – οριζόντια θέση σε σχέση με την πρώτη υλοποίηση(συνεχόμενα περιοδική αποστολή πακέτων) του λογισμικού του airmouse.

7 Σχεδίαση και υλοποίηση airmouse προγράμματος οδήγησης

Υπάρχουν δυο προσεγγίσεις στον σχεδιασμό και στην υλοποίηση του προγράμματος οδήγησης, η διαφορά των οποίων και σε αυτό το επίπεδο έγκειται στον τρόπο διαχείρισης της ενέργειας που καταναλώνει το airmouse κατά την λειτουργία του.

7.1 Πρώτη υλοποίηση προγράμματος οδήγησης airmouse

Στην πρώτη υλοποίηση η κίνηση του κέρσορα εξαρτάται αποκλειστικά από την κίνηση του airmouse. Μια τέτοια προσέγγιση έχει σαν αποτέλεσμα την ανάγκη για συνεχόμενη αποστολή δεδομένων από το airmouse έτσι ώστε να επιτευχθεί μια ομαλή κίνηση του κέρσορα στην οθόνη. Κάτι τέτοιο οδηγεί στην αδυναμία του συστήματος να χρησιμοποιηθεί μαζί με την δεύτερη υλοποίηση του λογισμικού του airmouse, το οποίο αντί να στέλνει συνέχεια πακέτα με τιμές του X και Y άξονα, στέλνει πακέτα, μόνο όταν διαπιστωθεί αλλαγή στην θέση του. Συνεπώς κρίνεται αναγκαίος ο συνδυασμός της πρώτης υλοποίησης του airmouse προγράμματος οδήγησης που τρέχει στον ηλεκτρονικό υπολογιστή μας με την πρώτη υλοποίηση του λογισμικού που τρέχει στην πλατφόρμα του airmouse. Αυτός ο συνδυασμός επιβαρύνει σε μεγάλο βαθμό την διαχείριση της ενέργειας της συσκευής, βασιζόμενοι στο γεγονός ότι για να επιτευχθεί ομαλότερη κίνηση του κέρσορα στη οθόνη απαιτείται ο μεγαλύτερος δυνατός ρυθμός αποστολής δεδομένων. Έτσι οδηγούμαστε στην δημιουργία ενός αντιοικονομικού ενεργειακά συστήματος, το οποίο είναι αναγκασμένο να κάνει όσον τον δυνατόν μεγαλύτερη χρήση του RF, κάτι που συνεπάγεται και μεγαλύτερη χρήση των ενεργειακών πόρων (μπαταρία) του, έτσι ώστε να πραγματοποιήσει τον σκοπό για τον οποίο σχεδιάστηκε. Επιπρόσθετα η απώλεια των μεταδιδόμενων από το airmouse πακέτων παίζει μεγάλο ρόλο στην κίνηση του κέρσορα, αφού κάθε χαμένο πακέτο πρακτικά συνεπάγεται την διακοπή της ομαλής κίνησης του κέρσορα στην οθόνη του υπολογιστή. Συμπερασματικά, η πρώτη υλοποίηση του προγράμματος οδήγησης του airmouse υστερεί:

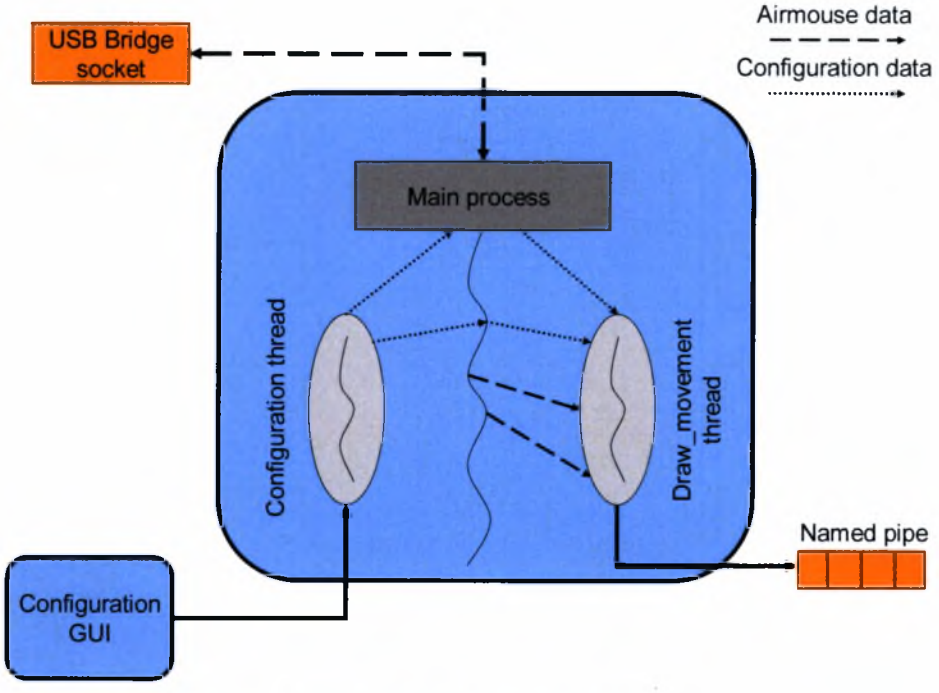
- τόσο στην διαχείριση κατανάλωσης ισχύος,
- όσο στην ευέλικτη διαχείριση απώλειας πακέτων.

7.2 Δεύτερη υλοποίηση προγράμματος οδήγησης airmouse

Η δεύτερη υλοποίηση προσπαθεί να λύσει ακριβώς αυτό το πρόβλημα της κατανάλωσης ενέργειας και να το διαχειριστεί όσο πιο έξυπνα μπορεί, με βάση τα υπάρχοντα προγραμματιστικά εργαλεία. Σκοπός αυτής της υλοποίησης είναι η μείωση της χρήσης του RF, της ασύρματης επικοινωνίας μεταξύ της airmouse συσκευής και του αντίστοιχου προγράμματος οδήγησής της στην μεριά του υπολογιστή.

Για να επιτευχθεί μια τέτοια προσέγγιση απαιτείται η σχεδίαση του προγράμματος οδήγησης να βασίζεται στην εξής διαπίστωση: Η κίνηση του κέρσορα στην οθόνη θα πρέπει να εξαρτάται το λιγότερο δυνατό από τον ρυθμό με τον οποίο το airmouse στέλνει τα δεδομένα – μετρήσεις του αισθητήρα του. Κάτι τέτοιο μπορεί εύκολα να υλοποιηθεί με την χρήση μιας διεργασίες – νήματος η οποία επιφορτίζεται με την αποστολή πακέτων στον Xserver που καθορίζουν αυτή την κίνηση. Το νήμα draw_movement, τμήμα του προγράμματος οδήγησης, στέλνει δεδομένα στον Xserver με ένα παραμετροποιήσιμο από τον χρήστη ρυθμό αποστολής, ανεξάρτητο από αυτόν με τον οποίο λαμβάνει δεδομένα από το airmouse. Ουσιαστικά το draw_movement νήμα «ζωγραφίζει» την κίνηση του κέρσορα πάνω στην οθόνη μέχρι το main

process να παραλάβει κάποιο νέο πακέτο από το airmouse με την νέα κατεύθυνση οπότε και αρχίζει να «ζωγραφίζει» προς τη νέα κατεύθυνση. Στο Εικόνα 16 φαίνεται η κύρια δομή του airmouse προγράμματος οδήγησης.



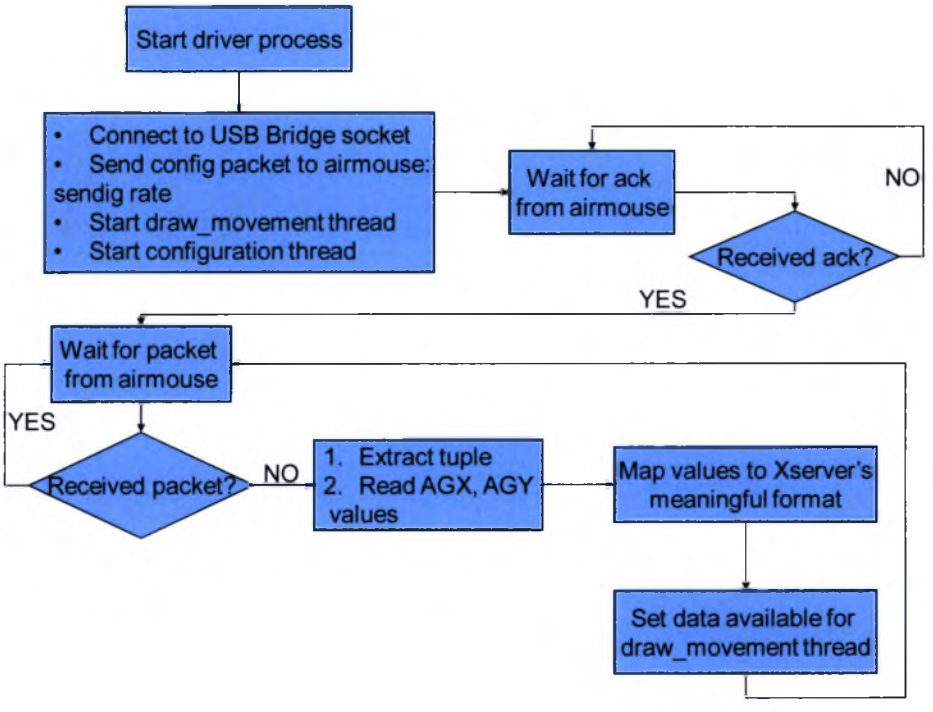
Εικόνα 16: Δομή airmouse driver.

Σε γενικές γραμμές μια περιγραφή της κύριας διεργασίας και των δύο νημάτων είναι η εξής:

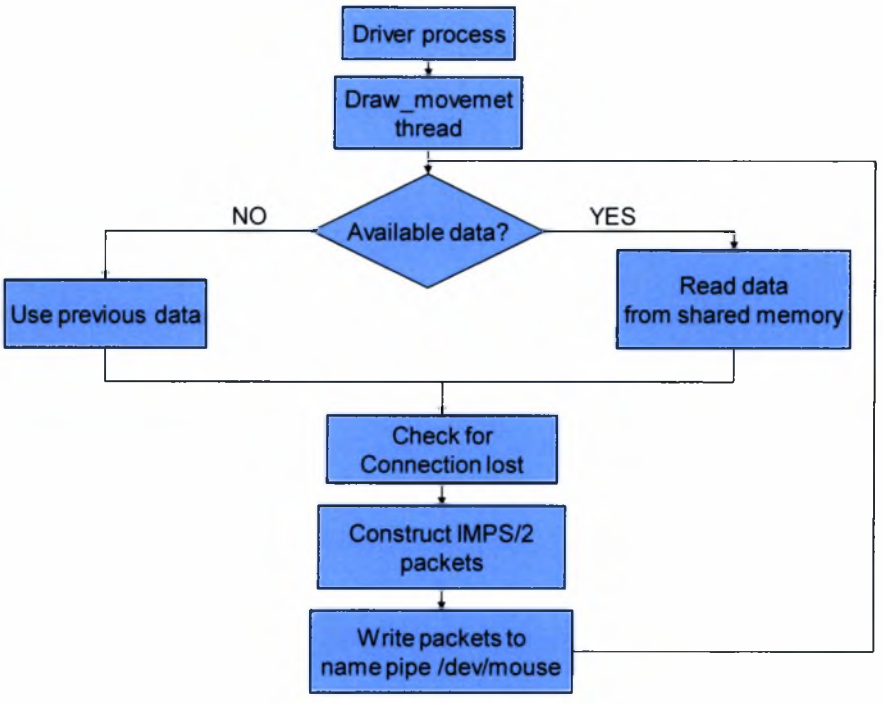
- Το Main process:
  1. Ανοίγει το socket για την επικοινωνία με το airmouse.
  2. Στέλνει ένα configuration πακέτο στο airmouse για να καθορίσει τον επιθυμητό ρυθμό με τον οποίο περιμένει να του αποστείλει τα δεδομένα – μετρήσεις.
  3. Αρχικοποιεί και ξεκινάει τα δύο νήματα: α)configuration thread και β)Draw_movement thread.
  4. Περιμένει να λάβει τις μετρήσεις από το airmouse. Για κάθε καινούργιο πακέτο που λαμβάνει :
     - διαβάζει τα tuples του πακέτου και αφού τα επεξεργαστεί,
     - ενημερώνει το draw_movement νήμα για αλλαγή στις συντεταγμένες, βάση των οποίων γίνεται η σχεδίαση της κίνησης του κέρσορα στην οθόνη του υπολογιστή.
- Το configuration thread:
  1. Περιμένει από το χρήστη μέσω του γραφικού περιβάλλοντος, που παρέχεται σαν επιπλέον εφαρμογή να λάβει τις καινούργιες παραμέτρους.
  2. Ενημερώνει κατάλληλα, αφού λάβει τις καινούργιες παραμέτρους, τόσο το main process όσο και το draw_movement thread.
- Το draw_movement thread:

1. Στέλνει τις συντεταγμένες του x, y άξονα στον Xserver μέσω του named pipe – fifo /dev/mouse, για την διαδικασία κίνησης του κέρσορα στην οθόνη. Η αποστολή των δεδομένων γίνεται περιοδικά, με ρυθμό ο οποίος εξαρτάται από την επιλογή του χρήστη.
2. Ενημερώνεται για τις συντεταγμένες του x, y άξονα καθώς και το ρυθμό με τον οποίο γράφει τα δεδομένα στο named pipe στην περίπτωση που κάτι τέτοιο απαιτηθεί τόσο από το main process όσο από το configuration thread.
3. Ελέγχει περιοδικά αν υπάρχει απώλεια σύνδεσης (connection lost) μεταξύ airmouse και προγράμματος οδήγησης.

Παρακάτω φαίνονται τα διαγράμματα ροής του main driver process (Εικόνα 17) και του draw_movement thread (Εικόνα 18) αντίστοιχα.



Εικόνα 17: Διάγραμμα ροής main driver process.



Εικόνα 18: Διάγραμμα ροής draw_movement_thread

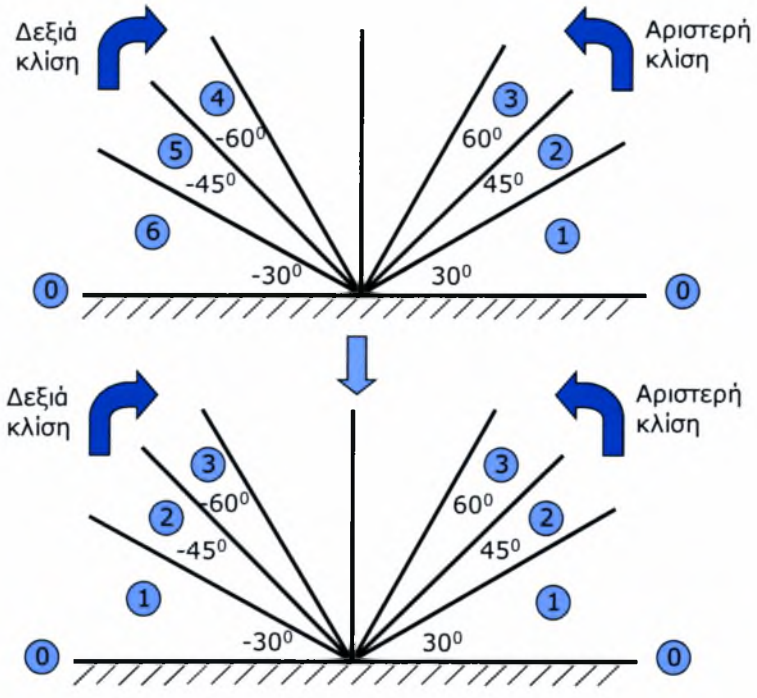
7.2.1 Αντιστοίχηση των τιμών του αισθητήρα σε ταχύτητα κίνησης του κέρσορα της οθόνης

Η αντιστοίχηση των τιμών του αισθητήρα σε ταχύτητα κίνησης του κέρσορα της οθόνης βασίζεται σε τρεις παράγοντες:

1. Στην αντιστοίχηση των τιμών του αισθητήρα σε καταστάσεις γωνιακής κλίσης.
2. Στον ρυθμό με τον οποίο γράφει τα δεδομένα το draw_movement thread στο named pipe, απ' όπου τα διαβάζει ο Xserver και σχεδιάζει την αντίστοιχη κίνηση του κέρσορα στην οθόνη.
3. Στην τιμή των δεδομένων.

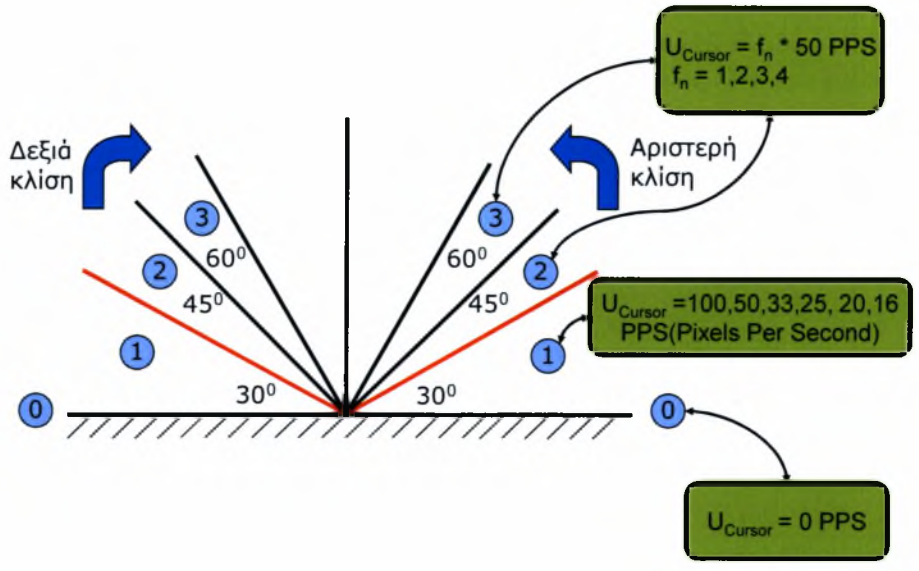
Η αντιστοίχηση των τιμών του αισθητήρα στις καταστάσεις γωνιακής κλίσης των 0^0, 30^0, 45^0, 60^0, -30^0, -45^0, -60^0 γίνεται όπως ακριβώς έχει περιγραφεί στο Κεφάλαιο 6.2.1. Η μόνη διαφορά οφείλεται στο πλήθος των καταστάσεων που χρησιμοποιούμε. Οι 7 καταστάσεις κάθε άξονα μπορούν να συμπτυχθούν σε 4 αν τις κάνουμε ζευγάρια: η κατάσταση των 30^0 δεξιά επί του άξονα X με την κατάσταση των -30^0 αριστερά επί του άξονα X γίνονται μια, η κατάσταση των 45^0 δεξιά επί του άξονα X με την κατάσταση των -45^0 αριστερά επί του άξονα X γίνονται μια, η κατάσταση των 60^0 δεξιά επί του άξονα X με την κατάσταση των 60^0 αριστερά επί του άξονα X γίνονται μια. Ομοίως και με τις καταστάσεις επί του άξονα Y. Αυτή η σύμπτυξη των καταστάσεων (Εικόνα 19) βασίζεται στις εξής δύο παραδοχές:

- η ταχύτητα του κέρσορα στις καταστάσεις κάθε ζευγαριού είναι ίδια,
- κάθε ζευγάρι καταστάσεων αντιστοιχίζεται σε διαφορετική ταχύτητα του κέρσορα.



Εικόνα 19

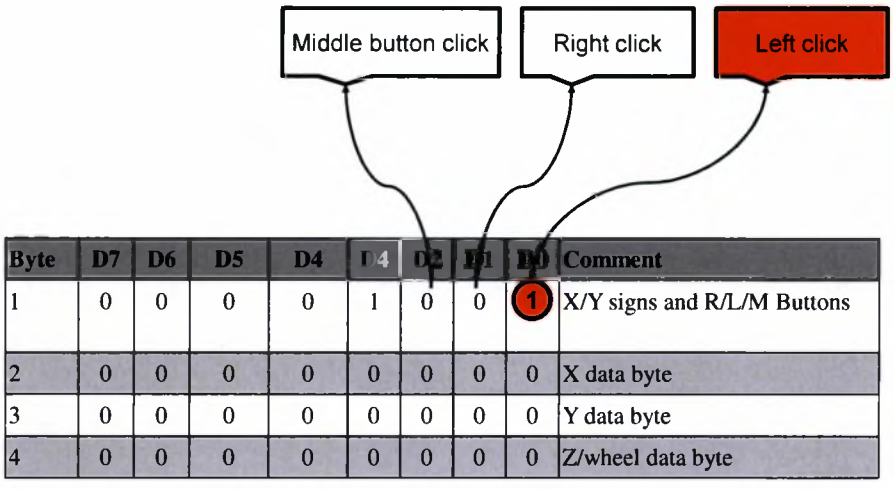
Ο ρυθμός με τον οποίο στέλνει το draw_movement thread τα δεδομένα στον Xserver καθορίζει την ταχύτητα του κέρσορα. Το πρόγραμμα οδήγησης υποστηρίζει τρεις διαφορετικές ταχύτητες: 100 πιξελς το δευτερόλεπτο (Pixels Per Second), 50 PPS, 33 PPS, 25 PPS, 20 PPS, 16 PPS και Ο PPS. Τα ζευγάρια τον καταστάσεων χρησιμοποιούνται για την υλοποίηση της επιπλέον επιτάχυνσης (extra acceleration) του κέρσορα. Ο χρήστης μέσω του προγράμματος οδήγησης μπορεί να καθορίσει ένα ζευγάρι καταστάσεων ως κατώφλι. Στην πραγματικότητα κάθε ζευγάρι καταστάσεων αποτελεί ενδεχόμενο κατώφλι. Μόλις η θέση του αισθητήρα ξεπεράσει το συγκεκριμένο κατώφλι η ταχύτητα του κέρσορα πολλαπλασιάζεται με ένα παράγοντα f_n = 1, 2, 3, 4. Στο Εικόνα 20 φαίνεται η αντιστοίχηση μεταξύ των καταστάσεων που αναγνωρίζει το πρόγραμμα οδήγησης με βάση την θέση του airmouse και των υποστηριζόμενων ταχυτήτων του κέρσορα στην περίπτωση που χρησιμοποιηθεί σαν κατώφλι το ζευγάρι καταστάσεων των 30^0 .



Εικόνα 20: Αντιστοίχιση τιμών του αισθητήρα σε ταχύτητα του κέρσορα.

7.2.2 Υλοποίηση κλικ

Το πρόγραμμα οδήγησης μπορεί να ρυθμιστεί από τον χρήστη ώστε να υποστηρίζει την λειτουργικότητα του αριστερού κλικ. Η ιδέα που κρύβεται πίσω από την υλοποίηση είναι απλή: *το πρόγραμμα οδήγησης κάθε φορά που διαπιστώνει ότι το airmouse βρίσκεται στην θέση ισορροπίας για χρονικό διάστημα ίσο του click_delay στέλνει ένα "κλικ event" στον Xserver.* Η καθυστέρηση του κλικ από την στιγμή που διαπιστωθεί ακινητοποίηση του αισθητήρα στην θέση αναφοράς, θέση κατά την οποία δεν υπάρχει κίνηση του κέρσορα στην οθόνη, είναι παράμετρος που καθορίζεται από τον χρήστη και μπορεί να λάβει τις εξής τιμές: click_delay = 0, 1, 2, 3, 4, 5 sec. Ο οδηγός αφού διαπιστώσει ότι το airmouse βρίσκεται στην θέση ισορροπίας του για χρονικό διάστημα ίσο με το click_delay που έχει ορίσει ο χρήστης, δημιουργεί ένα νέο IntelliMouse data πακέτο με την μορφή που φαίνεται στην Εικόνα 21 και το γράφει στο named pipe, απ' όπου το διαβάζει ο Xserver, ο οποίος και υλοποιεί την διαδικασία του κλικ.



Byte	D7	D6	D5	D4	D3	D2	D1	D0	Comment
1	0	0	0	0	1	0	0	1	X/Y signs and R/L/M Buttons
2	0	0	0	0	0	0	0	0	X data byte
3	0	0	0	0	0	0	0	0	Y data byte
4	0	0	0	0	0	0	0	0	Z/wheel data byte

Εικόνα 21: Δομή IMPS/2 πακέτου για υλοποίηση αριστερού κλικ

Στην ειδική περίπτωση όπου click_delay = 0, το πρόγραμμα οδήγησης δεν υλοποιεί την λειτουργικότητα του αριστερού κλικ.

7.2.3 Αξιοσημείωτα μέρη κώδικα

Παραθέτουμε τμήματα του προγράμματος που αφορούν:

- στη μετατροπή των τιμών του αισθητήρα σε αριθμούς συμπληρώματος ως προς δυο, τιμές οι οποίες μπορούν να χρησιμοποιηθούν από τον airmouse οδηγό,

```
signed int acl_parser(short high, short low){
        unsigned short return_value ;
        unsigned short msb;
                /*1. combine hi and low byte: x = hi*256 + low
            the resulting number x is in 2-complement, x is 16bit wide */
        return_value =  (high<<8) + low ;
        /*2. Compute the decimal value from 2-complement:
            • get the most significant bit
            • if 1 then sign is "-" otherwise "+" */
        msb = return_value >> 15;
        if ( msb == 0 ) {
                return (signed int)return_value;
        }
        else {
        /*Compute the decimal value from the negative 2-complement:
            1. the bits are inverted, or "flipped", by using the bitwise NOT operation
            2. the value of 1 is then added to the resulting value.*/
                return_value = ~return_value;
                return_value = return_value + 1;
                return (signed int) -1 * return_value;
        }
}
```

- στον επιπρόσθετο χειρισμό κάποιον βασικών signals: SIGINT, SIGTERM για καλύτερη διαχείριση των πόρων που καταλαμβάνει το πρόγραμμα οδήγησης,

```
void handler()
{
 fprintf(stderr, "\n%s: Mouse killed!\n", "Airmouse Driver");
 close(FIFO);                /*close named pipe fifo */
 p_pkt_free(r_packet);
 p_pkt_free(packet);

 p_socket_close(receive_socket);
 p_socket_close(send_socket);
 exit (0);
}
```

- στην δημιουργία IMPS/2 πακέτων για αποστολή των δεδομένων στον Xserver.

```
void track_mouse(unsigned char x, unsigned char y,unsigned char click,char sign_x, char sign_y)
{
        unsigned char out[4],outbyte;
        int ret;
        /*create first byte*/
        outbyte = 0x8;
        if(sign_y == 1)
                outbyte |= 0x20;
        if (sign_x == 1)
                        outbyte |= 0x10;
        if( click == 0)
                out[0]=outbyte;
        else
                out[0] = outbyte | 0x1;

        out[1] = x;             /*create second byte*/
        out[2] = y;             /*create third byte*/
        out[3] = 0x0;
        ret = write(FIFO, out, 4);
}
```

8 Driver configuration GUI

Θα πρέπει να αναφέρουμε ότι το γραφικό περιβάλλον δεν είναι ενσωματωμένο στην υλοποίηση του προγράμματος οδήγησης. Ο οδηγός και το configuration GUI δεν αποτελούν ένα μονολιθικό σύστημα, αντίθετα υλοποιούνται από δύο διαφορετικές διεργασίες. Αυτό δίνει την δυνατότητα στον χρήστη να μπορεί εύκολα να αλλάζει τις ρυθμίσεις και παραμέτρους του προγράμματος οδήγησης:

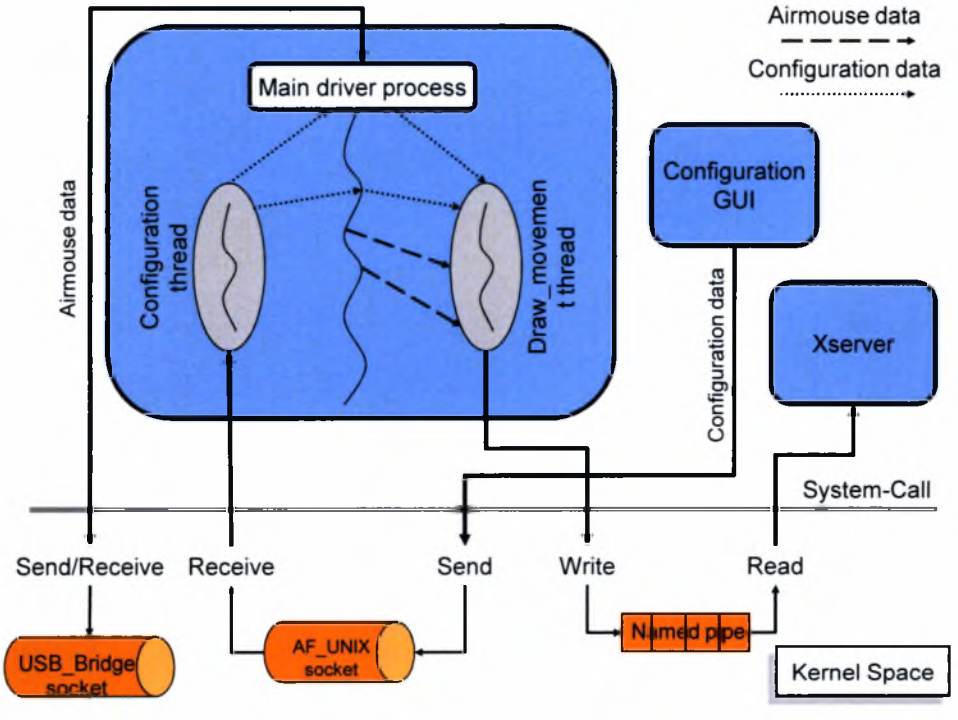
1. απομακρυσμένα, π.χ. πάνω από Ethernet, χρησιμοποιώντας το GUI σε κάποιον άλλο υπολογιστή,
2. με την χρήση οποιασδήποτε άλλης εφαρμογής η οποία κάνει χρήση του πρωτοκόλλου που έχουμε ορίσει για αυτή την επικοινωνία.

8.1 Πρωτόκολλο επικοινωνίας προγράμματος οδήγησης – GUI

Το πρωτόκολλο βασίζεται στην χρήση UNIX address family(AF_UNIX) socket και η επικοινωνία είναι connection-oriented (τύπος socket: SOCK_STREAM). Κάθε πακέτο που αποστέλλεται από το configuration GUI περιέχει πληροφορία για:

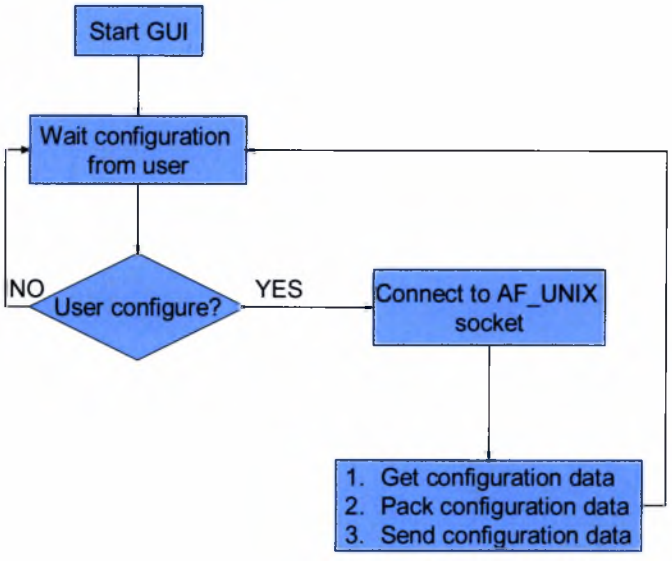
- τον ρυθμό με τον οποίο "ζωγραφίζει" το draw_movement thread την κίνηση του κέρσορα, draw_movement_sending_rate (ταχύτητα σε PPS)
- επιπλέον επιτάχυνση (παράγοντας f_n)
- την κλίση του αισθητήρα, πάνω από την οποία θα υποστηρίζεται extra acceleration, tilt_threshold
- την καθυστέρηση του κλικ, click_delay
- τον ρυθμό αποστολής των μετρήσεων του αισθητήρα του airmouse, sensors_sending_rate

Η επικοινωνία γίνεται μεταξύ του GUI και του configuration thread του προγράμματος οδήγησης όπως φαίνεται στην Εικόνα 22.

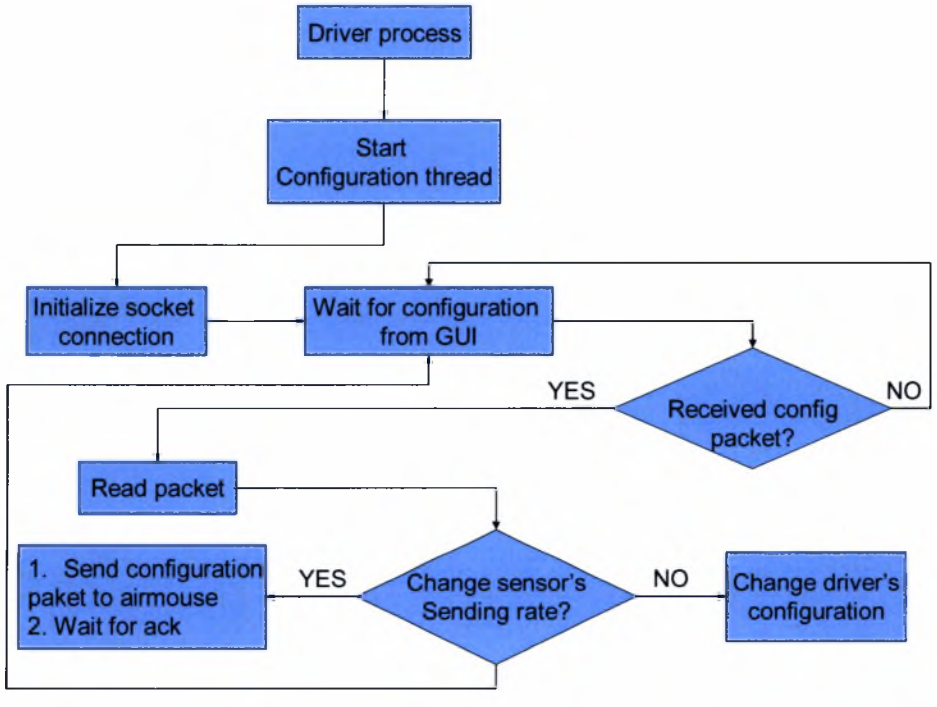


Εικόνα 22: Επικοινωνία των διεργασιών του συστήματος.

Παρακάτω φαίνονται τα διαγράμματα ροής του configuration GUI (Εικόνα 23) και του Configuration thread του airmouse οδηγού (Εικόνα 24).



Εικόνα 23: Διάγραμμα ροής διεργασίας GUI.



Εικόνα 24: Διάγραμμα ροής Configuration thread.

9 Calibration tool

Το calibration tool υλοποιείται από μια καινούργια διεργασία ανεξάρτητη του προγράμματος οδήγησης του airmouse, η οποία εξυπηρετεί δύο απαιτήσεις για το σύστημα:

- Δυνατότητα χρήσης διαφορετικών αισθητήρων, δεδομένου ότι κάθε αισθητήρας σε οριζόντια θέση μπορεί να επιστρέφει διαφορετικές τιμές. Σε διαφορετική περίπτωση τόσο ο οδηγός όσο και το airmouse θα έπρεπε να επαναπρογραμματίζονταν με τις καινούργιες τιμές a, b για τους x, y άξονες, που επιστρέφει ο αισθητήρας όταν βρεθεί σε οριζόντια θέση.
- Δυνατότητα επιλογής από το χρήστη διαφορετικής θέσης αναφοράς – θέση κατά την οποία όταν βρεθεί ο αισθητήρας δεν θα υπάρχει κίνηση του κέρσορα στην οθόνη.

Η διαδικασία της βαθμονόμησης[14] τόσο του προγράμματος οδήγησης όσο και του αισθητήρα του ασύρματου airmouse επιτυγχάνεται με την χρήση του calibration tool. Η διεργασία που υλοποιεί το calibration tool:

1. Μπλοκάρει προσωρινά, μέχρι τη ολοκλήρωσή της, την λειτουργία του οδηγού (λήψη μετρήσεων από τον αισθητήρα, αποστολή τιμών στον Xserver για την σχεδίαση της κίνησης του κέρσορα στην οθόνη).
2. Δειγματοληπτεί τον αισθητήρα όταν αυτός βρεθεί στην θέση αναφοράς που επιθυμεί ο χρήστης, ελέγχοντας αν έχει χαθεί η σύνδεση με το airmouse.
3. Βρίσκει την τιμή με την μεγαλύτερη συχνότητα στο δείγμα(τιμές a,b του άξονα x, y) και την στέλνει τόσο στο airmouse όσο και στο πρόγραμμα οδήγησης της συσκευής εισόδου.

Η διαδικασία της δειγματοληψίας κρίνεται αναγκαία δεδομένου του ότι ο αισθητήρας είναι αρκετά ευαίσθητος σε μικρές κινήσεις. Από την στιγμή που τερματίσει με επιτυχία το calibration tool, τόσο το airmouse όσο και το πρόγραμμα οδήγησης υπολογίζουν με βάση τις τιμές που έλαβαν τις καινούργιες αντιστοιχήσεις των τιμών του αισθητήρα σε γωνιακή κλίση της θέσης του.

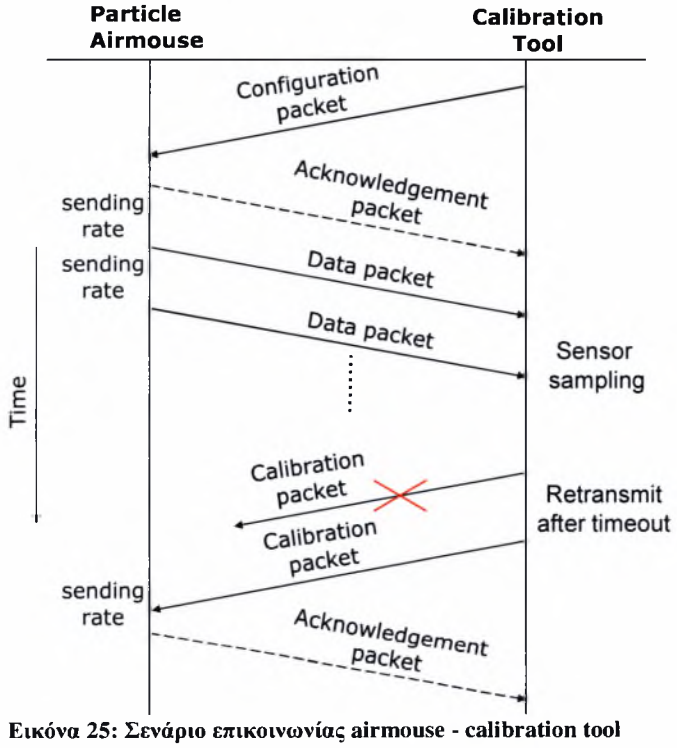
9.1 Επικοινωνία Calibration tool – airmouse

Η επικοινωνία της ασύρματης συσκευής εισόδου και του calibration tool γίνεται μέσω του USB Bridge και βασίζεται στο πρωτόκολλο AwareCon. Η διαδικασία είναι απλή και φαίνεται στην Εικόνα 24. Το calibration tool αρχικά στέλνει ένα configuration πακέτο, ορίζοντας τον ρυθμό αποστολής των δεδομένων για την δειγματοληψία του αισθητήρα. Το airmouse απαντά με ένα acknowledgement πακέτο, πιστοποιώντας ότι αρχικοποιήθηκε με αυτή την νέα παράμετρο και αρχίζει να στέλνει τις μετρήσεις του αισθητήρα του περιοδικά. Στην περίπτωση που το calibration tool δεν λάβει την απάντηση μέσα σε κάποιο χρονικό διάστημα τότε ξαναστέλνει το configuration πακέτο. Η επικοινωνία διαρκεί μέχρι το calibration tool να ολοκληρώσει την δειγματοληψία(δείγμα 40 τιμών). Αφού η διαδικασία ολοκληρωθεί, το calibration tool δημιουργεί ένα πακέτο calibration και το στέλνει στο airmouse περιμένοντας για πακέτο acknowledgement. Στην περίπτωση που δεν ληφθεί απάντηση μέσα σε κάποιο χρονικό διάστημα το πακέτο ξανά-αποστέλλεται (Εικόνα 25). Η επικοινωνία calibration tool – airmouse βασίζεται στην ανταλλαγή τριών διαφορετικού τύπου πακέτων:

1. Το configuration πακέτο αποτελείται από δύο ACL tuples με τα ακόλουθα subject:
   - CAC Address Control Field (102, 64): για το handshake μεταξύ airmouse – calibration tool και την αποστολή πακέτου acknowledgement.

- CRS Control Remote Sensor (204, 232): για το configuration του airmouse, καθορίζοντας τον ρυθμό αποστολής των δεδομένων του airmouse.

2. Κάθε data πακέτο αποτελείται από δύο ACL tuples με τα ακόλουθα subjects:
   - SGX Sensor GravityX (234,128): για την 16bit τιμή της επιτάχυνσης της βαρύτητας ως προς τον άξονα X.
   - SGY Sensor GravityY (240,192): για την 16bit τιμή της επιτάχυνσης της βαρύτητας ως προς τον άξονα Y.
3. Κάθε calibration πακέτο αποτελείται από τρία ACL tuples με τα ακόλουθα subjects:
   - CAC Address Control Field (102, 64): για την αποστολή πακέτου acknowledgement.
   - SGX Sensor GravityX (234,128): για την 16bit τιμή της επιτάχυνσης της βαρύτητας ως προς τον άξονα X. Τιμή που προέρχεται από την δειγματοληψία του αισθητήρα και αναφέρεται στην θέση ισορροπίας του αισθητήρα για το άξονα X.
   - SGY Sensor GravityY (240,192): για την 16bit τιμή της επιτάχυνσης της βαρύτητας ως προς τον άξονα Y. Τιμή που προέρχεται από την δειγματοληψία του αισθητήρα και αναφέρεται στην θέση ισορροπίας του αισθητήρα για το άξονα Y.

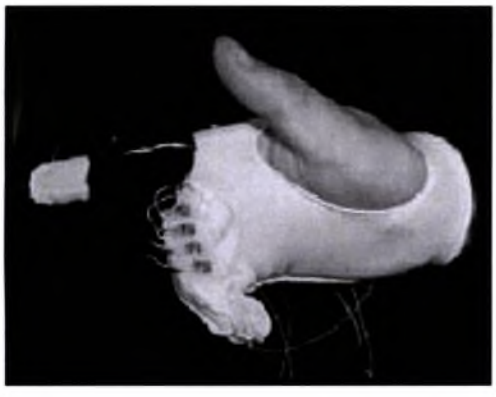


Εικόνα 25: Σενάριο επικοινωνίας airmouse - calibration tool

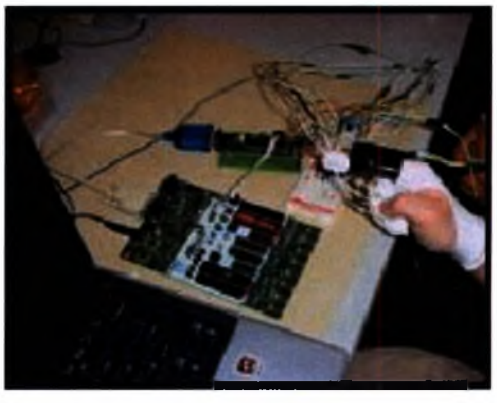
10 Σχετική Έρευνα

ATmega32-based Airmouse

Οι Andrew Sawchuk και Joseph Tanen ασχολήθηκαν με την δημιουργία ενός motion-sensing glove[18] (Εικόνα 26) παρέχοντας τον έλεγχο την κίνηση του κέρσορα της οθόνης του υπολογιστή, χωρίς την χρήση ποντικιού. Σκοπός ήταν η δημιουργία μιας σειριακής συσκευής εισόδου (Εικόνα 27) βασισμένη σε κλισιόμετρο που θα παρείχε αντίστοιχη λειτουργικότητα με αυτή ενός συμβατικού ποντικιού (tilt mouse). Βασισμένο στον ATmega32 microcontroller και στο ADXL2303 accelerometer, το σύστημα παρέχει υποστήριξη δεξιού και αριστερού κλικ καθώς και κάθετη ανακύλιση/κατακύλιση περιεχομένων οθόνης H/Y (vertical scrolling). Στο airmouse γίνεται χρήση τεσσάρων κουμπιών που καθορίζουν την λειτουργικότητα του: Output On/Off, Left Click, Right Click και Scroll Enabled. Η σχεδίαση βασίζεται σε δυο πρότυπα: την χρήση του Microsoft serial mouse πρωτοκόλλου και της RS 232 σειριακής επικοινωνίας. Το Microsoft serial mouse πρωτόκολλο ορίζει ότι κάθε πακέτο θα πρέπει να έχει την δομή που φαίνεται στην Εικόνα 28.



Εικόνα 26: motion-sensing glove.



Εικόνα 27: Σειριακή συσκευή εισόδου.

Byte	D7	D6	D5	D4	D4	D2	D1	D0
1	1	1	L	R	Y7	Y6	X7	X6
2	1	0	X5	X4	X3	X2	X1	X0
3	1	0	Y5	X4	Y3	Y2	Y1	Y0
4	1	0	0	M	Z3	Z2	Z1	Z0

Εικόνα 78: Δομή Microsoft serial mouse πακέτου.

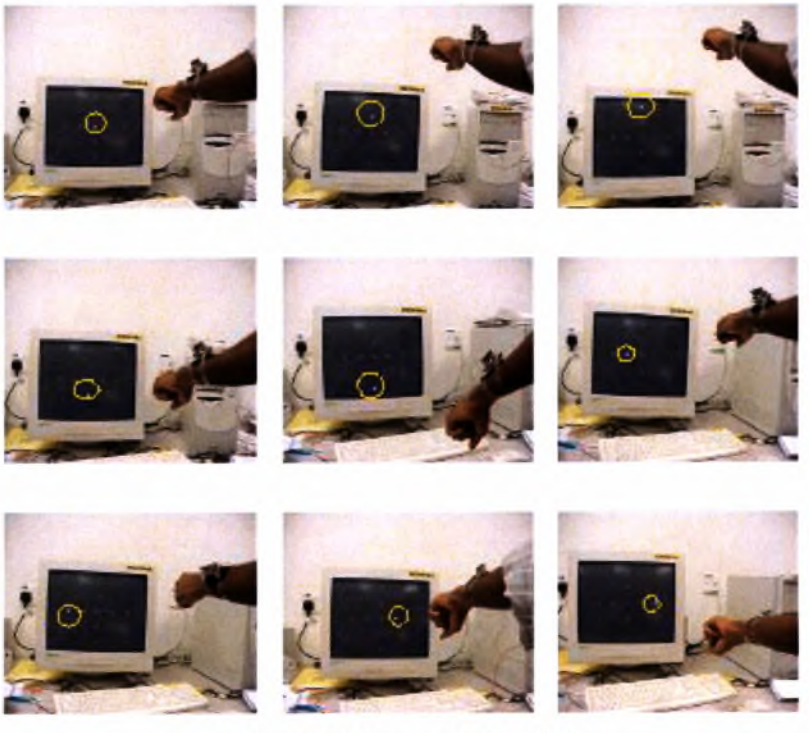
Η επιλογή κάθε κουμπιού του airmouse καθορίζει το περιεχόμενο των τεσσάρων bytes του Microsoft serial mouse πακέτου που αποστέλλεται:

- Output On/Off: ελέγχει την διαδικασία αποστολής δεδομένων
- Left Button: καθορίζει την τιμή του L bit: 0/1.
- Right Button: καθορίζει την τιμή του R bit: 0/1.
- Scroll Enabled Button: καθορίζει την λειτουργία του airmouse για κάθετη περιήγηση (vertical scrolling), την τιμή των Z3...Z0 bits.

Η RS 232 σειριακής επικοινωνίας επιτυγχάνεται μέσω του MAX233A RS-232 driver/receiver chip. Η ιδέα πάνω στην οποία βασίζεται η υλοποίηση του συστήματος είναι απλή. Υπολογίζεται δειγματοληπτικά η επιτάχυνση του airmouse που οφείλεται στην βαρύτητα, με ένα ADC και στην συνέχεια πολλαπλασιάζεται με μια σταθερά έτσι ώστε να γίνει μια περαιτέρω διαβάθμιση των τιμών.

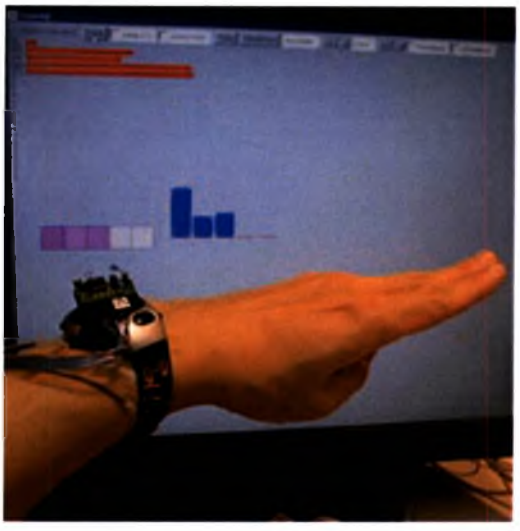
Gesture Wrist

Το Gesture wrist[17] του Jun Rekimoto μπορεί να φορεθεί στον καρπό και να αποτελέσει μια πρωτότυπη συσκευή εισόδου (Εικόνα 29). Αποτελείται από ένα Analog Devices ADXL202 επιταχυνσίμετρο που μετράει συνεχώς την κλίση του αντιβραχίονα, και μεταδίδει τις τιμές στον υπολογιστή. Η κλίση στον άξονα x μετακινεί τον κέρσορα κάθετα και η κλίση στον άξονα y μετακινεί τον κέρσορα οριζόντια. Η λειτουργία της επιλογής επιτυγχάνεται με ένα διακόπτη πάνω στην πλατφόρμα. Το Gesture wrist επιπλέον μπορεί να αναγνωρίσει και χειρονομίες του χεριού μετρώντας τις αλλαγές στο σχήμα του καρπού (Εικόνα 30, Εικόνα 31). Κάτι τέτοιο επιτυγχάνεται με την χρήση δύο ηλεκτροδίων(transmitter/receiver) που τοποθετούνται στο εσωτερικό του περιβραχιόνιου του Gesture wrist και μετρούν αλλαγές στην μορφή του μυ του καρπού κατά το άνοιγμα και κλείσιμο τις παλάμη του χεριού του χρήστη που φοράει την συσκευή. Η λειτουργικότητα αυτή στηρίζεται στην τεχνική του capacitance sensing: τεχνική για τον υπολογισμό της απόστασης κοντινών αγώγιμων αντικειμένων μετρώντας τη χωρητικότητα μεταξύ του αισθητήρα και του αντικειμένου.



Εικόνα 29: Χρήση του Gesture Wrist.

Εικόνα 30: Διαφορετικές θέσεις καρπού αναγνωρίσιμες από το Gesture Wrist.



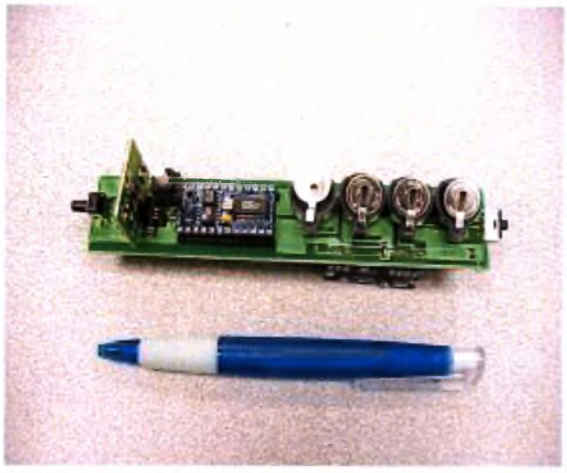
Εικόνα 31

Wear pen

Το Wear pen[6] είναι μια συσκευή εισόδου όμοια με ένα συμβατικό στυλό (Εικόνα 32). Η κίνηση αναγνωρίζεται μετρώντας την επιτάχυνση και η στιγμιαία θέση βρίσκεται αφού γίνει φιλτράρισμα της επιτάχυνση. Το wear pen έχει ένα Basic stamp BS2SX micro-controller και δύο ADXL202E επιταχυνσίμετρα(accelerometers) τοποθετημένα κάθετα το ένα ως προς το άλλο παρέχοντας τρισδιάστατη 3D πληροφορία για τη θέση του. Η επικοινωνία με τον υπολογιστή γίνεται μέσω του TX3 wireless transmitter ασύρματα. Η μετάδοση γίνεται στα 914 Mhz και η μορφή των πακέτων φαίνεται στον Πίνακα 4.

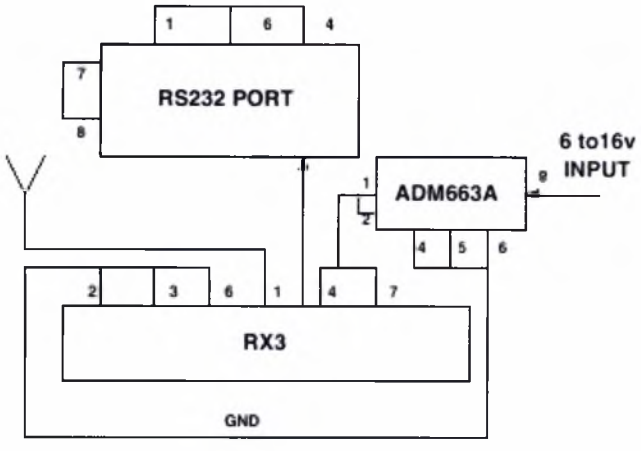
Byte Number	Description
1	1(decimal)
2	X Axis
3	Y Axis of accelerometer 1
4	Y Axis of accelerometer 2

Πίνακας 4

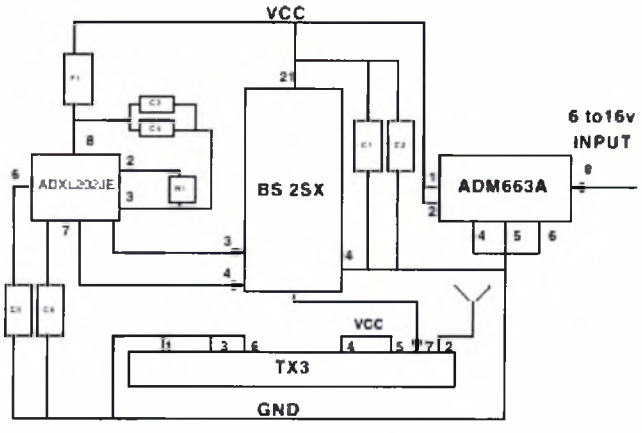


Εικόνα 32: Wear pen.

Η δομή του πομπού - wear pen και του δέκτη – υπολογιστή διακρίνεται στις Εικόνες 33 και 34.



Εικόνα 33



Εικόνα 34

Βιβλιογραφία

[1] Analog Devices Inc. ADXL210 Datasheet, 2001.

[2] Baudel Thomas, Beaudouin-Lafon Michel, Charade: Remote Control of Objects using free-hand gestures, Communication of the ACM, 36(7):28-35, July 1993.

[3] Beigl Michael, Gellersen Hans, Smart-Its: Ambedded Platform for Smart Objects, Smart Objects Conference 2003, May 15-17, Grenoble, France.

[4] Beigl Michael, Krohn Albert, Zimmer Tobias, Decker Christian, Typical Sensors needed in Ubiquitous and Pervasive Computing, First International Workshop on Networked Sensing Systems (INSS) 2004, Tokyo, Japan, June 22-23. 2004, pp 153-158.

[5] Chapweske Adam, PS/2 Mouse Interface, http://www.computer-engineering.org/index.php?title=PS/2_Mouse_Interface.

[6] Cheok Adrian David, Kumar Krishnamoorthy Ganesh, Prince Simon, Micro-Accelerometer based Hardware Interfaces for Wearable Computer Mixed Reality Applications, National University of Singapore, Singapore 117576

[7] Compatibility of Type 2 Pointing Devices with MS IntelliMouse, Updated: December 4, 2001, http://www.microsoft.com/whdc/device/input/mcompat.mspx.

[8] Configuring Xorg, the X server configuration howto, http://www.gentoo.org/doc/en/xorg-config.xml#doc_chap3.

[9] Faruqui Faisal, Introduction to Interprocess Communication Using Named Pipes, Sun Article, July 2002.

[10] Kitchin Charles, Using Accelerometer in Low g Applications, Analoge Devices Inc. Application Note.

[11] Krohn Albert, Beigl Michael, Decker Christian, Kochendorfer Uwe, Robinson Philip, Zimmer Tobias, Inexpensive and Automatic Calibration for Acceleration Sensors, Telecooperation Office (TECO), University of Karlsruhe.

[12] Krohn Albert, Beigl Michael, Decker Christian, Robinson Philip, Zimmer Tobias, AwareCon: Situation Aware Context Communication, Ubicomp 2003, Oct. 12-15, Seattle, USA.

[13] Krohn Albert, Beigl Michael, Decker Christian, Robinson Philip, Zimmer Tobias, ConCom – A language and Protocol for Communication of Context, Telecooperation Office (TECO), University of Karlsruhe.

[14] Matson Joe, Calibrating the ADXL210 Accelerometer, December 1999.

[15] Papadimitriou Kyriakos, Dollas Apostolos, Sotiropoulos N. Stamatios, Low-Cost Real-Time 2-D Motion Detection Based on Reconfigurable Computing, IEE Transaction on Instrumentation and Measurement, VOL. 55, NO 6, Dec 2006.

[16] Particle Web Site, University of Kurlsruhe, http://particle.teco.edu.

[17] Rekimoto Jun, Gesture Wrist and GesturePad: Unobtrusive Wearable Interaction Devices.

[18] Sawchuk Andrew, Tanen Joseph, Nontraditional Cursor Control – Atmega32 Based Motion Sensing, Circuit Cellar, the Magazine for Computer Applications.

[19] The X Window System, http://en.wikipedia.org/wiki/X_Window_System.

[20] Vaught Andy, Introduction to Named Pipes, Linux Journal, Article No. 18, ISSN:1075-3583, September 1997.

[21] Zimmer Tobias, Towards a Better Understanding of Context Attributes, PerCom 2004, Orlando Florida USA, pp. 23-28, ISBN 0-7695-2106-1, IEEE, March 2004.