ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανίχνευση Συμβάντων και Δειγματοληψία Αποθέματος σε Ασύρματα Δίκτυα Αισθητήρων

Μάρκος Π. Μπερσίμης

Επιβλέπων: Ευστάθιος Χατζηευθυμιάδης, Αναπληρωτής Καθηγητής

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2015

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανίχνευση Συμβάντων και Δειγματοληψίας Αποθέματος σε Ασύρματα Δίκτυα Αισθητήρων

Μάρκος Π. Μπερσίμης

Α.Μ.: 1115200800080

ΕΠΙΒΛΕΠΩΝ: **Ευστάθιος Χατζηευθυμιάδης,** Αναπληρωτής Καθηγητής

ΠΕΡΙΛΗΨΗ

Στόχος της παρούσας πτυχιακής εργασίας είναι η ανάλυση των τιμών θερμοκρασίας που αποστέλλει ένα ασύρματο δίκτυο αισθητήρων, χρησιμοποιώντας δύο διαφορετικές υλοποιήσεις. Η πρώτη, αφορά στην εφαρμογή του αλγορίθμου Shewhart για κάθε απεσταλμένη τιμή, σε πραγματικό χρόνο, ενώ η δεύτερη αφορά στη συσσώρευση τιμών και την εξαγωγή δείγματος, μέσω ενός Αλγορίθμου Αποθέματος και έπειτα την τροφοδότηση του μεγίστου (ή μέσου όρου) αυτού του δείγματος, ως είσοδο στον αλγόριθμο Shewhart. Τέλος, η αποτελεσματικότητα των δύο υλοποιήσεων συγκρίνεται με τη βοήθεια της απόστασης Hamming.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ:Ασύρματα Δίκτυα Αισθητήρων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: π.χ. αισθητήρες,αλγόριθμος Shewhart,ανίχνευση συμβάντος, αλγόριθμος αποθέματος, απόσταση Hamming

ABSTRACT

The main aim of this thesis is to analyze the temperature values sent over a wireless sensor network using two different implementations. The first one implements Shewhart algorithm processing every value sent in real time, whereas the second one, accumulates the values and exports a sample using a reservoir algorithm. From that sample, called reservoir, the max or the average value is extracted and then forwarded as input in the Shewhart algorithm. Finally, the efficiency of the two outputs is compared by means of Hamming distance.

SUBJECT AREA: wireless sensor networks

KEYWORDS: sensors, Shewhart algorithm, event detection, reservoir algorithm, Hamming distance

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία εκπονήθηκε στο πλαίσιο του προπτυχιακού τμήματος Πληροφορικής και Τηλεπικοινωνιών της Σχολής Θετικών Επιστημών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών. Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα αναπληρωτή καθηγητή κ. Ευστάθιο Χατζηευθυμιάδη, για την άριστη συνεργασία και την καθοδήγησή του, συμβάλλοντας να υλοποιηθεί επιτυχώς η παρούσα εργασία.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1. SUNSPOT

1.1. Περιγραφή της πλατφόρμας

Η προσομοίωση υλοποιήθηκε μέσω της πλατφόρμας SunSPOT (Sun Small Programable Object Technology (SPOT)), η οποία είναι μια μικρού μεγέθους, ασύρματη, πειραματική πλατφόρμα που λειτουργεί με μπαταρίες. Είναι προγραμματισμένη σχεδόν εξ ολοκλήρου σε Java. Αυτό δίνει τη δυνατότητα σε προγραμματιστές να δημιουργήσουν προγράμματα που απαιτούσαν εξειδικευμένες δεξιότητες προγραμματισμού για ενσωματωμένα συστήματα. Η πλατφόρμα υλικού περιλαμβάνει μια σειρά από ενσωματωμένους αισθητήρες και παρέχει τη δυνατότητα εύκολης διεπαφής με εξωτερικές συσκευές.

Τα SunSPOTs (ή SPOTs) χρησιμοποιούν μια υλοποίηση της Java ME που λέγεται Squawk και υποστηρίζει CLDC 1.1 και MIDP 1.0. Τα SPOTs δεν έχουν κάποιο λειτουργικό σύστημα, αλλά τρέχουν την Squawk VM απευθείας πάνω στον επεξεργαστή, και η VM παρέχει τις βασικότερες λειτουργίες ενός Λειτουργικού Συστήματος. Επιπλέον σε Java είναι γραμμένοι όλοι οι drivers των συσκευών.
Κάθε SunSPOT kit περιέχει :

- Δύο πλήρεις, ελευθέρου βεληνεκούς (free-range) SunSpots (με επεξεργαστή, ραδιοσυχνότητα, πίνακα αισθητήρων και μπαταρία)
- ‘Εναν SunSPOT σταθμό βάσης (base station) (με επεξεργαστή και ραδιοσυχνότητα)
- Ένα USB καλώδιο για την σύνδεση των SPOTs/basestation με τον υπολογιστή
- Ένα cd με όλα τα εργαλεία ανάπτυξης λογισμικού που απαιτούνται προκειμένου να ξεκινήσει η ανάπτυξη εφαρμογών για SunSPOT.

Εικόνα 1: Ένα πλήρες SunSPOT kit

Ο σταθμός βάσης (base station) συνδέεται με τον υπολογιστή (PC). Η σύνδεση αυτή μας επιτρέπει να γράφουμε προγράμματα που μπορούν να τρέξουν στον υπολογιστή μας και ακόμη να χρησιμοποιήσουμε τη ραδιοσυχνότητα του σταθμού βάσης (base station) ώστε να επικοινωνήσουμε με απομακρυσμένα SunSPOTs. Ο σταθμός βάσης μπορεί ακόμα να χρησιμοποιηθεί από τα εργαλεία ανάπτυξης με στόχο τον εντοπισμό σφαλμάτων που αφορούν σε εφαρμογές απομακρυσμένων SunSPOT.

Ως ένας σταθμός βάσης, θα μπορούσε να χρησιμοποιηθεί και ένα πλήρες SunSPOT, με δεδομένο ότι δε θα μπορούσε να χρησιμοποιηθεί ο πίνακας αισθητήρων του.

Πλατφόρμες ανάπτυξης που υποστηρίζονται

Το αρχικό λογισμικό της Sun SPOT έχει δοκιμαστεί επιτυχώς σε

- Windows XP
- Windows7 (και 32-bit και 64-bit),
- Macintosh OS X 10.4
- Linux (έχει ελεγχθεί διεξοδικά σε Ubuntu 10.10 32-bit και 64-bit εκδόσεις),
- Solaris x86.
- PowerPC και
- Intel-based υπολογιστές,

Φυσικά οι πλατφόρμες ανάπτυξης ενημερώνονται και ανανεώνονται συνέχεια [1] [2].

Εικόνα 2: Μια συσκευή SunSPOT

Εξομοιωτής ή προσομοιωτής για Sun SPOT

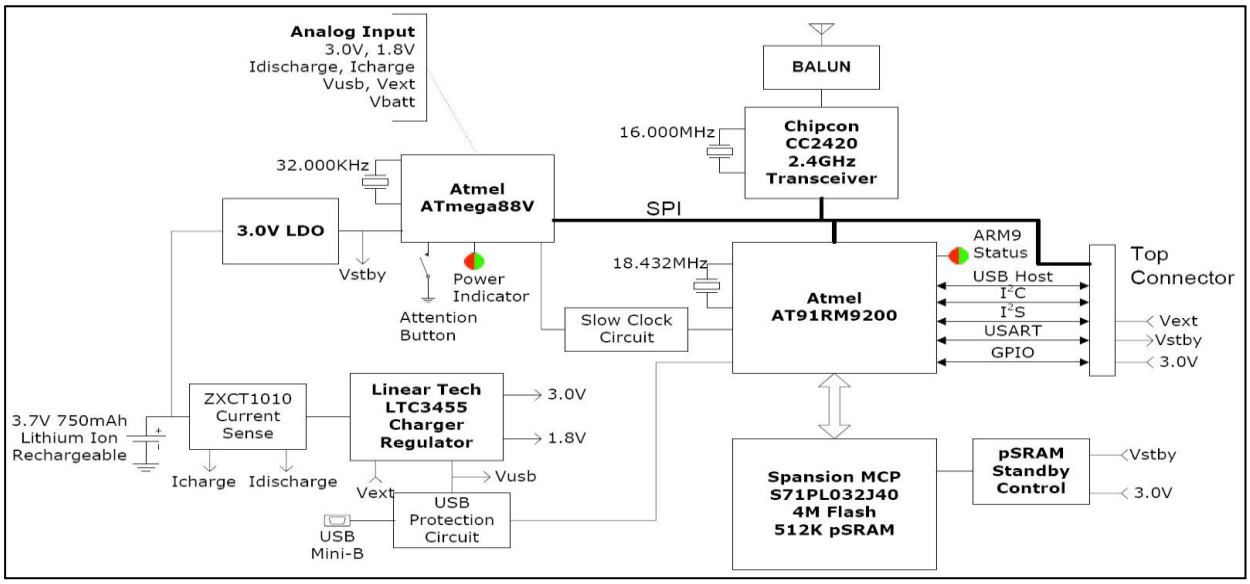
Η έκδοση 6.0 περιλαμβάνει έναν εξομοιωτή ως μέρος του SPOTWorld. Μ αυτόν τον εξομοιωτή εκτελείται μια εφαρμογή SunSPOT στον υπολογιστή. Με αυτό επιτυγχάνεται ο έλεγχος ενός προγράμματος πριν από την ανάπτυξη σε ένα πραγματικό SunSPOT, ή στην περίπτωση που ένα πραγματικό SunSPOT δεν είναι διαθέσιμο.

Η SPOTWorld επιτρέπει να εμφανιστεί ένα εικονικό SPOT με έναν πίνακα ελέγχου, όπου μπορούμε να ορίσουμε οποιαδήποτε από τις πιθανές εισόδους αισθητήρων (π.χ. επίπεδο φωτός, θερμοκρασία, ψηφιακές εισόδους pin, αναλογικές τάσεις εισόδου, τις τιμές και επιταχυνσιόμετρο), αντικαθιστώντας κατ' αυτόν τον τρόπο ένα φυσικό πίνακα αισθητήρων (sensorboard), Η εφαρμογή μας μπορεί να ελέγξει το χρώμα τω LEDs που εμφανίζεται στην εικόνα του εικονικού SPOT, όπως ακριβώς θα συνέβαινε αν ήταν ένα πραγματικό SPOT. Κάνοντας κλικ, με το ποντίκι στους διακόπτες, στο εικονικό SunSPOT μπορούμε να αλλάξουμε τις τιμές τους. Επίσης λήψη και αποστολή μπορεί να επιτευχθεί μέσω ραδιοσυχνότητας. Έχοντας εκχωρήσει τη δική του διεύθυνση, κάθε εικονικό SPOT, μπορεί να μεταδώσει σε ένα ή σε πολλά άλλα εικονικά SunSPOT. Ένα εικονικό SunSPOT μπορεί επίσης να αλληλεπιδράσει μέσω ραδιοσυχνότητας με πραγματικά SunSPOTs, στην περίπτωση που είναι διαθέσιμος ένας κοινός σταθμός βάσης [2].

1.2. Περιγραφή του Υλικού (Hardware)

Οι διαστάσεις ενός Sun SPOT είναι οι εξής: μήκος 71 mm, πλάτος 42.40 mm και ύψος 18mm και μπορεί να χωρέσει στην παλάμη ενός ενήλικα. Ένα Sun SPOT αποτελείται από τα παρακάτω στοιχεία:

- Έναν κύριο επεξεργαστή, πρόκειται για ένα ολοκληρωμένο κύκλωμα 180 MHz 32-bit ARM920T.
- Μνήμη 4MB Flash και 512K RAM.
- Έναν μικροεπεξεργαστή 8-bit Atmel Atmega 88,που χρησιμοποιείται σαν ελεγκτής ισχύος.
- Μία εσωτερική επαναφορτιζόμενη μπαταρία, που φορτίζεται μέσω θύρας USB, (3.7V rechargeable, 750 mAh lithium – ion battery). Η μπαταρία αυτή, μπορεί να διαρκέσει μέχρι και 7 ώρες με ενεργό τον επεξεργαστή και την ασύρματη επικοινωνία. Δύναται, ο χρόνος αυτός να επεκταθεί αν καμία λειτουργία δε λαμβάνει χώρα, οπότε και το Sun SPOT τίθεται σε κατάσταση αδρανοποίησης. Σε συνεχή χρήση των LEDS του πίνακα αισθητήρων, η διάρκεια της μπαταρίας μπορεί να είναι μέχρι και 3 ώρες.
- Αισθητήρες θερμοκρασίας και έντασης φωτισμού, 8 τρίχρωμους LEDS, 6 αναλογικές εισόδους που διαβάζονται από έναν αναλογικό-σε-ψηφιακό μετατροπέα, ένα τριαξωνικό επιταχύμετρο,2 διακόπτες, 5 γενικής χρήσης I/O pins και 4 υψηλής τάσης εξόδους pins.
- Έναν πομποδέκτη TI CC2420, (παλαιότερα ChipCon), με ενσωματωμένη κεραία, ο οποίος χρησιμοποιεί το πρωτόκολλο IEEE 802.15.4 για την ασύρματη επικοινωνία λειτουργώντας στη συχνότητα των 2.4GHz [3] [4] [5].



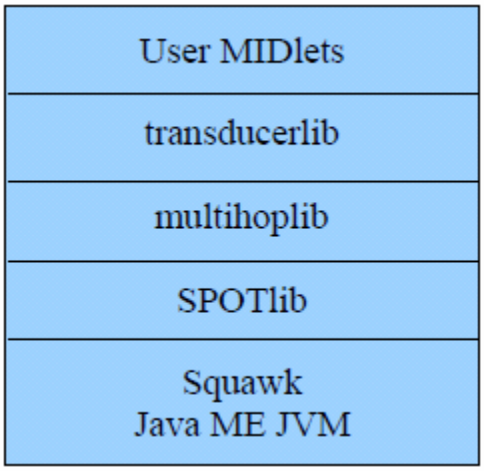
Σχήμα 1: Δομή πλακέτας επεξεργαστή SunSPOT

1.3. Περιγραφή της Αρχιτεκτονικής

Προκειμένου να αναπτυχθεί λογισμικό για την πλατφόρμα SunSPOT είναι απαραίτητο να γνωρίζει κανείς την αρχιτεκτονική τους.

1.3.1. Αρχιτεκτονική απλού free range SunSPOT.

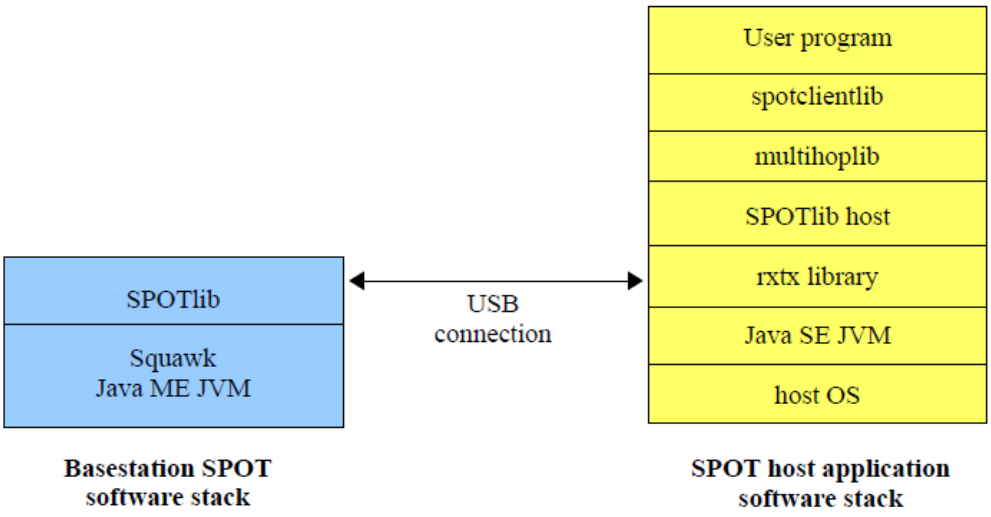
- Ξεκινώντας από την κορυφή, βρίσκεται η εφαρμογή που έγραψε ο χρήστης για το SPOT, που επεκτείνει την Java ME MIDlet class.
- Στο κατώτερο επίπεδο είναι η Squawk JVM. Δεν υπάρχει λειτουργικό σύστημα, η Squawk τρέχει αυτοτελώς.
- Ενδιάμεσα βρίσκονται οι διάφορες βιβλιοθήκες του SPOT μιας και η πρόσβαση στη συσκευή του SPOT και οι βασικές λειτουργίες εισόδου/εξόδου παρέχονται από την SPOTlib. Τούτο περιλαμβάνει πρόσβαση στο χαμηλού επιπέδου MACradio protocol. Η βιβλιοθήκη multihoplib παρέχει υψηλότερου επιπέδου radio protocols όπως το Radiogram και το Radiostream και ταυτόχρονα φροντίζει τη δρομολόγηση όσων πακέτων δεν είναι σε άμεση επαφή με αυτό. Η βιβλιοθήκη transducerlib παρέχει ένα τρόπο πρόσβασης του υλικού στον πίνακα αισθητήρων του SPOT eDemo όπως π.χ. το επιταχυνσιόμετρο, τα LEDs, η ψηφιακή είσοδος/έξοδος, οι διακόπτες, οι αναλογικές είσοδοι κτλ [1] [6].



Εικόνα 3: Επίπεδα λογισμικού ενός free-range SunSPOT

1.3.2. Αρχιτεκτονική host – application SunSPOT

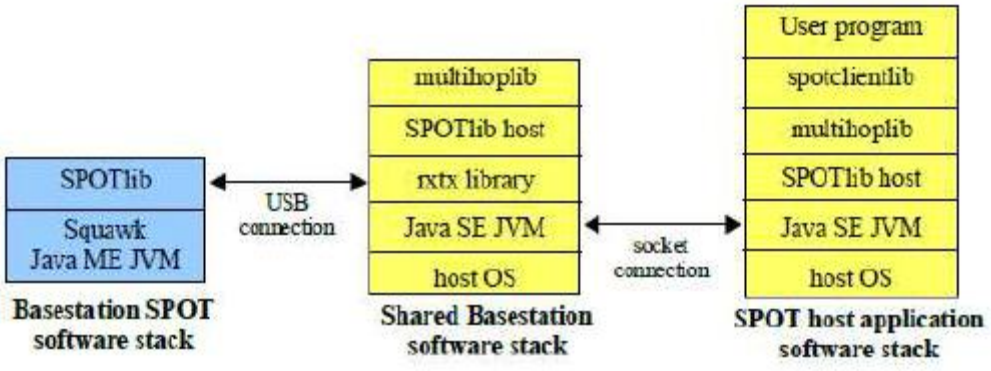
Ομοίως, στην κορυφή υπάρχει μία εφαρμογή γραμμένη από το χρήστη host SPOT, πρόκειται για ένα απλό πρόγραμμα σε Java SE. Δύναται να εκτελέσει όλες τις λειτουργίες που εκτελεί ένα σύνηθες πρόγραμμα σε JAVA:
είσοδο/έξοδο σε αρχείο, απεικόνιση Swing GUI's κτλ. Δύναται επίσης να στείλει και να λάβει μηνύματα μέσω ραδιοσυχνότητας με free-range SPOTs όταν ένας σταθμός βάσης basestation είναι σε σύνδεση με τον υπολογιστή. Στο κατώτερο επίπεδο βρίσκεται το λειτουργικό σύστημα του host: Linux, Windows, Mac OS X ή Solaris. Επάνω από το Λειτουργικό Σύστημα είναι η Java SE JVM με όλες τις βιβλιοθήκες της Java. Ενδιαμέσως υπάρχουν οι διάφορες βιβλιοθήκες των SPOT. Πρόσβαση στη συσκευή του SPOT και βασική είσοδος/έξοδος παρέχεται από την host έκδοση της SPOTlib . Αυτή περιλαμβάνει πρόσβαση στο χαμηλού επιπέδου πρωτόκολλο MAC radio, που είτε χρησιμοποιεί σύνδεση USB για να έχει πρόσβαση στο ράδιο του σταθμού βάσης είτε χρησιμοποιεί συνδέσεις socket για να επικοινωνεί με άλλες host applications. Η βιβλιοθήκη multihoplib και εδώ παρέχει υψηλότερου επιπέδου radio protocols όπως το Radiogram και το Radiostream και ταυτοχρόνως φροντίζει τη δρομολόγηση των πακέτων που δεν είναι σε άμεση επαφή με αυτό. Με τη βιβλιοθήκη spotclientlib παρέχεται πρόσβαση σε ένα πλήθος εντολών που μπορούν να αποσταλούν σε ένα ελευθέρου βεληνεκούς (free-range) SPOT. Αυτό περιλαμβάνει την εντολή "Hello" που χρησιμοποιείται για την ανακάλυψη SPOTs εντός της ραδιοσυχνότητας. Η βιβλιοθήκη rxtx χρησιμοποιείται για σειριακή είσοδο/έξοδο μέσω της usb σύνδεσης με το σταθμό βάσης basestation. Το Solarium και όλες οι SPOT SDK ant εντολές αποτελούν SPOT host applications [1] [6].



Εικόνα 4: Αρχιτεκτονική SunSPOT host - application

1.3.3. Αρχιτεκτονική SunSPOT – basestation

Ο σταθμός βάσης, χρησιμοποιώντας τη ραδιοσυχνότητα του, παρέχει έναν τρόπο επικοινωνίας μεταξύ των host applications με τα free-range SPOTs. Σημειώνεται ότι η host application τρέχει αποκλειστικά στον host υπολογιστή και δεν τρέχει καθόλου κώδικας του χρήστη στο σταθμό βάσης. Τα πακέτα αποστέλλονται μέσω USB στο basestation, το οποίο στη συνέχεια τα στέλνει έξω μέσω της ραδιοσυχνότητάς του. Ομοίως, όταν ο σταθμός βάσης δέχεται ένα πακέτο το προωθεί στο host application. Όταν χρησιμοποιείται ένα διαμοιραζόμενο basestation δεν φορτώνεται ο κώδικας στο SPOT, αντιθέτως τρέχει ο κώδικας της Basestation class στη βιβλιοθήκη του SPOT. Πακέτα τα οποία παραλαμβάνονται από άλλες host applications μέσω μίας σύνδεσης socket προωθούνται στα SPOTs χρησιμοποιώντας τη ραδιοσυχνότητα του basestation [1] [6].

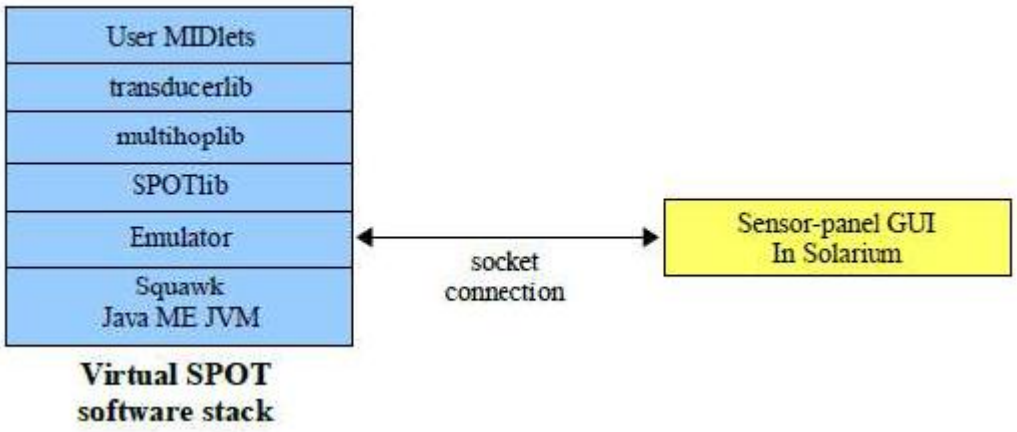


Εικόνα 5: Αρχιτεκτονική SunSPOT - basestation

1.3.4. Αρχιτεκτονική Virtual SunSPOT

Με κάθε νέα δημιουργία ενός νέου virtual SPOT στο Solarium, ξεκινάει μία νέα διαδικασία προκειμένου να τρέξει τον κώδικα του προσομοιωτή σε μία Squawk VM. Ο κώδικας του προσομοιωτή επικοινωνεί μέσω μίας σύνδεσης socket με τον κώδικα ενός virtual SPOT GUI στο Solarium. Σαν παράδειγμα μπορεί να αναφερθεί ότι , όταν η εφαρμογή του SPOT αλλάζει την RGB τιμή του ενός LED, η πληροφορία αυτή περνάει στον κώδικα του virtual SPOT GUI που ενημερώνει την απεικόνιση αυτού του LED με τη νέα RGB τιμή. Ομοίως, κλικάρωντας ο χρήστης, με το ποντίκι, έναν από τους διακόπτες του virtual SPOT, το Solarium στέλνει ένα μήνυμα στον κώδικα του προσομοιωτή ότι ο διακόπτης έχει κλικαριστεί γεγονός που γίνεται αντιληπτό από την SPOT application.

Κάθε virtual SPOT έχει τη δική του Squawk VM η οποία τρέχει ως ξεχωριστή διεργασία στον host υπολογιστή. Κάθε Squawk VM περιέχει μία στοίβα ραδιοσυχνοτήτων που είναι μέρος της βιβλιοθήκης SPOT και επιτρέπει στη SPOT εφαρμογή να επικοινωνεί με άλλες εφαρμογές SPOT που τρέχουν στον ίδιο υπολογιστή, όπως άλλα virtual SPOTs, χρησιμοποιώντας sockets πραγματικά SPOTs μέσω ραδιοσυχνότητας αν χρησιμοποιείται σταθμός βάσης [1] [6].



Εικόνα 6: Αρχιτεκτονική Virtual SunSPOT

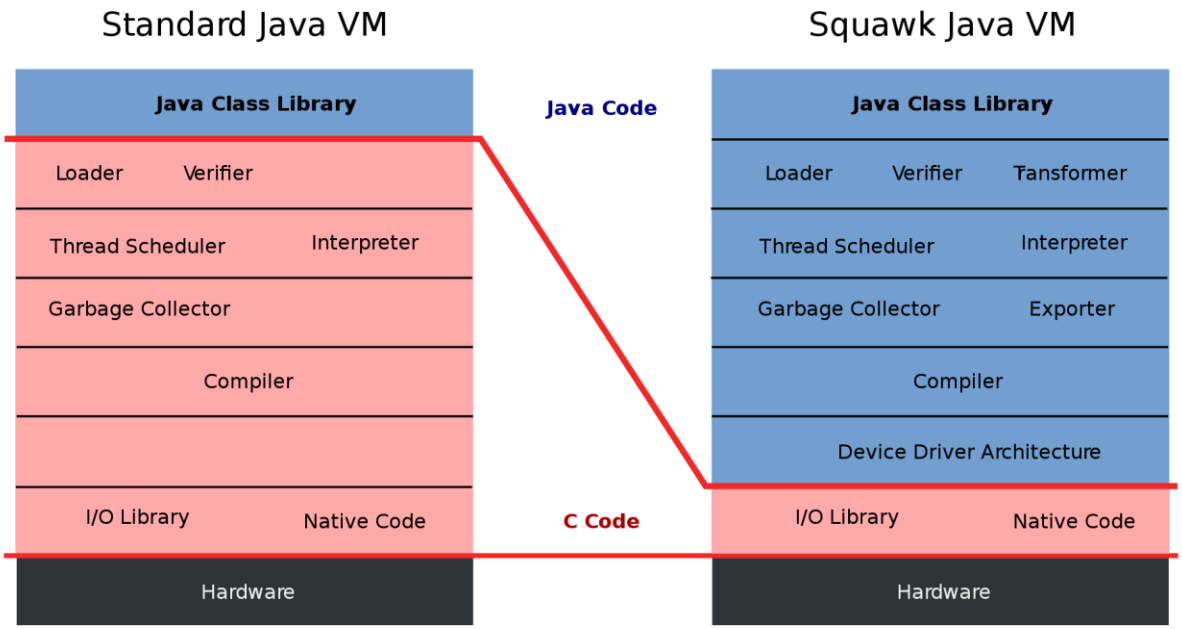
1.4. Περιγραφή Λογισμικού

1.4.1. Squawk Java Virtual Machine

Πρόκειται για ένα μηχάνημα εικονικής πραγματικότητας που κατασκεύασε η εταιρεία πληροφορικής Sun Microsystems για Java εφαρμογές σε ενσωματωμένα συστήματα και μικροσυσκευές, και αποτελεί το βασικό λογισμικό των Sun SPOTs. Η δημιουργία του προέκυψε από την ανάγκη για ένα μηχάνημα εικονικής πραγματικότητας σε Java συμβατό με τη λειτουργία CLDC, (Connected Limited Device Configuration), που δε θα χρειάζεται την ύπαρξη λειτουργικού συστήματος για να εκτελεστεί. Επομένως, το Squawk JVM σε αντίθεση με τα περισσότερα μηχανήματα εικονικής πραγματικότητας που έχουν γραφτεί σε χαμηλού επιπέδου γλώσσες, όπως C/C++ ή assembly, έχει γραφτεί σε Java και δεν είναι απαραίτητη η ύπαρξη λειτουργικού συστήματος για να εκτελεστεί. Το Squawk JVM προσφέρει πολλά πλεονεκτήματα όπως ασφάλεια, συλλογή απορριμμάτων και χειρισμό εξαιρέσεων.

Επιπλέον απαιτεί μικρή μνήμη, διευκολύνει τη μεταγλώττιση, (compiling), και εκτέλεση του κώδικα και παρέχει τη δυνατότητα ταυτόχρονης εκτέλεσης πολλαπλών εφαρμογών, με την κάθε εφαρμογή να είναι πλήρως ανεξάρτητη και απομονωμένη από όλες τις υπόλοιπες [3] [7] [8].

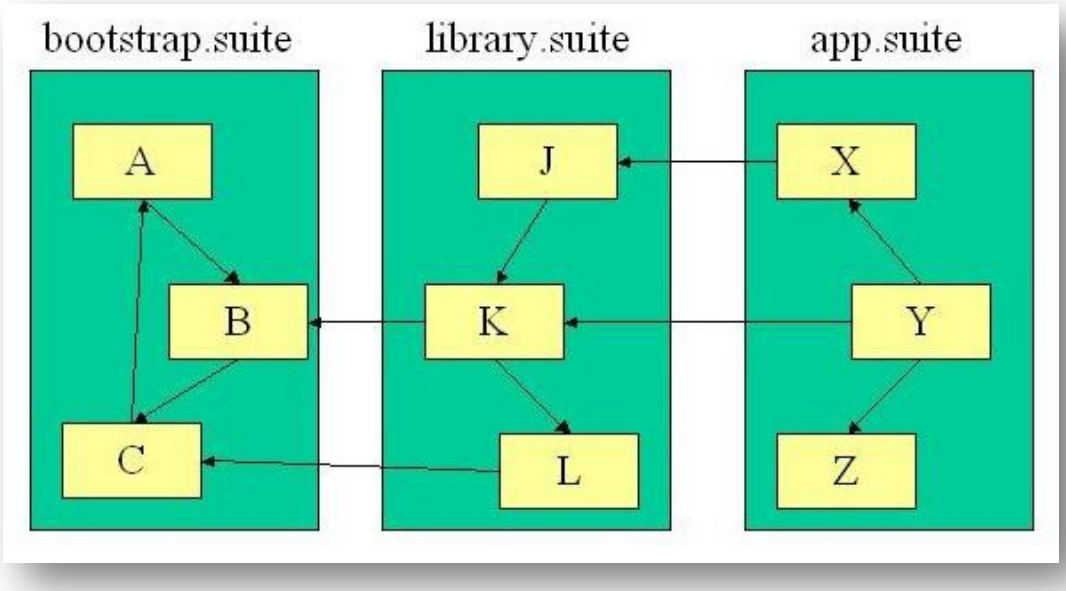
Το Squawk JVM αποτελεί λογισμικό ανοιχτού κώδικα. Στο Squawk JVM ο μεταγλωττιστής έχει γραφτεί σε C, ενώ ο συλλέκτης απορριμμάτων, (garbage collector), μεταφράζεται από την Java στην C. Η αρχιτεκτονική του Squawk JVM παρουσιάζεται στην εικόνα 7.



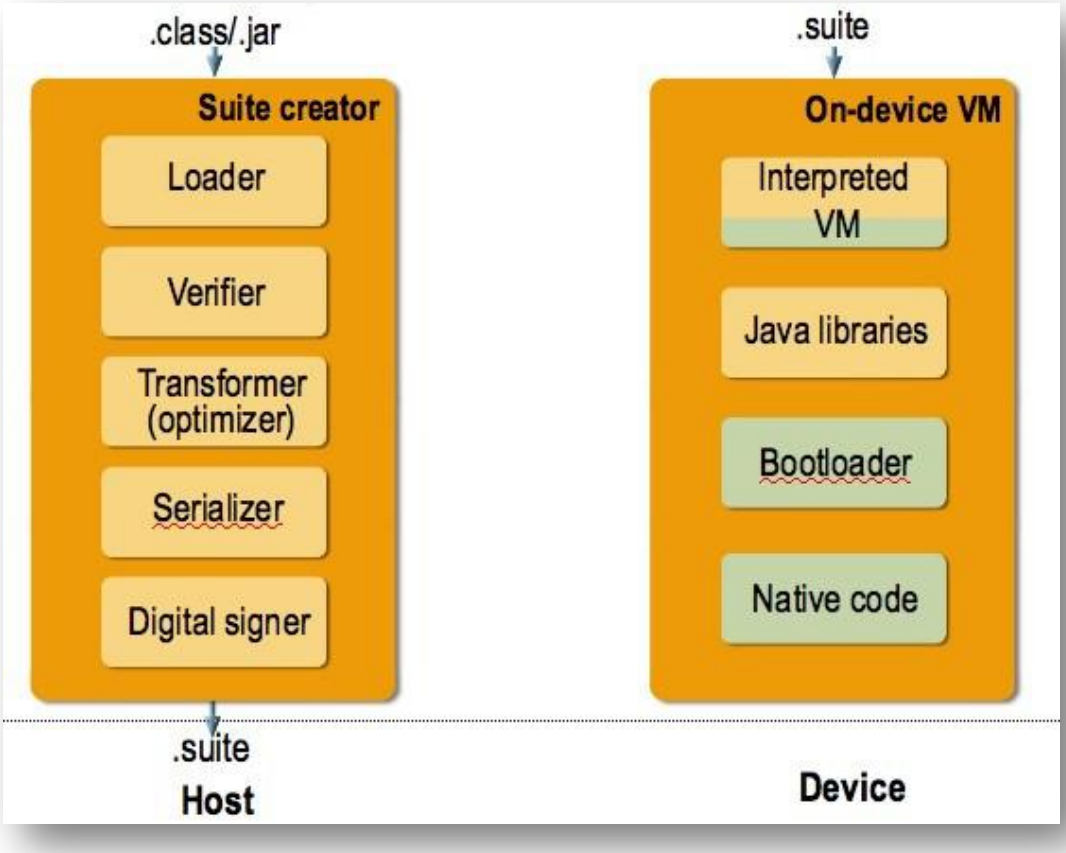
Εικόνα 7: Αρχιτεκτονική Squawk JVM

Εμφανίζει τα εξής χαρακτηριστικά:

- Περιέχει ένα αμετάβλητο και συμπαγές σύνολο εντολών με κατά 35 – 45% μικρότερο μέγεθος από αυτό των αντίστοιχων J2ME αρχείων, (Java 2 Platform Micro – Edition).
- Προσφέρει εσωτερική μετάφραση των κλάσεων σε ένα προσυνδεδεμένο συμπαγές σχήμα και έτσι επιτρέπει την αποδοτική εκτέλεση των bytecodes τα οποία βρίσκονται στη μνήμη μόνο για ανάγνωση.
- Διευκολύνει τη συλλογή απορριμμάτων, (garbage collection), δεδομένου ότι οι μεταβλητές διακρίνονται σε δείκτες και σε καθορισμένες από την Java μεταβλητές και συνεπώς απαιτείται μόνο ένας χάρτης δεικτών για κάθε εφαρμογή. Ενώ καθώς δεν υπάρχει τίποτα στη στοίβα κατά τη διάρκεια της εκτέλεσης του κώδικα, δεν υπάρχει λόγος για στατική μετάφραση μεθόδων.
- Περιλαμβάνει suites. Μία suite είναι ένα σύνολο κλάσεων. Κάθε κλάση μέσα σε μία suite μπορεί να αναφέρεται σε κλάσεις της ίδιας suite ή σε κλάσεις της γονικής suite. Δημιουργούνται έτσι αλυσίδες από suites, μοντέλο πολύ πρακτικό γιατί οι suites απαιτούν το 1/3 της μνήμης που απαιτούν οι κλάσεις. Στην εικόνα 8 παρουσιάζεται μία αλυσίδα από suites.
- Προκειμένου να εξοικονομηθεί μνήμη οι κλάσεις φορτώνονται σε ένα desktop μηχάνημα, όπου μετατρέπονται σε suites και στη συνέχεια φορτώνονται σε μικροσυσκευές και μεταγλωττίζονται από το Squawk JVM των μικροσυσκευών. Τούτο επιτρέπει μικρότερο μέγεθος για το εικονικό μηχάνημα στη μικροσυσκευή και ταχύτερη εκτέλεση των εφαρμογών. Στην εικόνα 9 παρουσιάζεται η αρχιτεκτονική μεταγλώττισης του Squawk JVM [3] [7].



Εικόνα 8: Αλυσίδα από suites



Εικόνα 9: Αρχιτεκτονική μεταγλώττισης εικονικής μηχανής

- Παρέχει application isolation. Στο Squawk JVM κάθε εφαρμογή αναπαρίσταται από ένα Java αντικείμενο. Αυτό το Java αντικείμενο είναι ένα στιγμιότυπο της κλάσης Isolate και χρησιμοποιείται για να μπορεί να ενημερωθεί για την κατάσταση της σχετιζόμενης εφαρμογής ή να επηρεάσει απευθείας την εφαρμογή μέσω συναρτήσεων όπως start(), pause(), resume() και exit(). Το Squawk JVM δίνει τη δυνατότητα εκτέλεσης πολλαπλών εφαρμογών στο ίδιο μηχάνημα. Με την χρήση των suites, ο isolate μηχανισμός του Squawk JVM επιτρέπει την κοινή χρήση αυτών ανάμεσα σε διαφορετικές εφαρμογές αλλά με την κάθε εφαρμογή να μην επηρεάζεται από την εκτέλεση των υπολοίπων. Καταυτό τον τρόπο απαιτείται λιγότερη μνήμη σε κάθε εφαρμογή. Ο isolate μηχανισμός του Squawk JVM υποστηρίζει τα περισσότερα χαρακτηριστικά του JSR 121 Isolate Application Programming Interface με τη διαφορά ότι χρησιμοποιεί ένα πιό απλό και λιγότερο αυστηρό μοντέλο.
- Παρέχει isolation migration. Κάθε isolate εφαρμογή που εκτελείται σε ένα Squawk JVM μηχάνημα δύναται να διακοπεί, να μετατραπεί σε ένα stream και να αποθηκευτεί σε ένα αρχείο και στη συνέχεια να επανεκκινηθεί σε ένα άλλο μηχάνημα Squawk JVM. Επομένως, μπορούν οι εκτελούμενες εφαρμογές να μετακινηθούν από ένα Sun SPOT σε ένα άλλο ή από ένα desktop μηχάνημα σε ένα άλλο ή και από έναν εξυπηρετητή, (server) σε έναν άλλο κ.ο.κ
- Υποστηρίζει πράσινα νήματα, (green threads). Τα πράσινα νήματα προσομοιώνουν πολυνηματικά περιβάλλοντα χωρίς τη συνδρομή κάποιου λειτουργικού συστήματος. Με το σταμάτημα της λειτουργίας ενός πράσινου νήματος παραδίδεται ο έλεγχος σε άλλα νήματα, π.χ. με την Thread.yield(), ή όταν πραγματοποιεί μια εντολή που αποκλείει άλλα νήματα, π.χ. read().
- Υποστηρίζει χειρισμό εξαιρέσεων. Με την πραγματοποίηση μιας εξαίρεσης, προστίθεται ένα bit στην Interrupt Status Word, (ISW), και απενεργοποιείται η εξαίρεση ώστε να μην επαναληφθεί. Κάθε φορά ο χρονοπρογραμματιστής ελέγχει το ISW, στέλνει σήμα στο Squawk JVM, το νήμα συνεχίζει την εκτέλεσή του από το σημείο που είχε διακοπεί και η εξαίρεση επανενεργοποιείται [3] [7] [9].

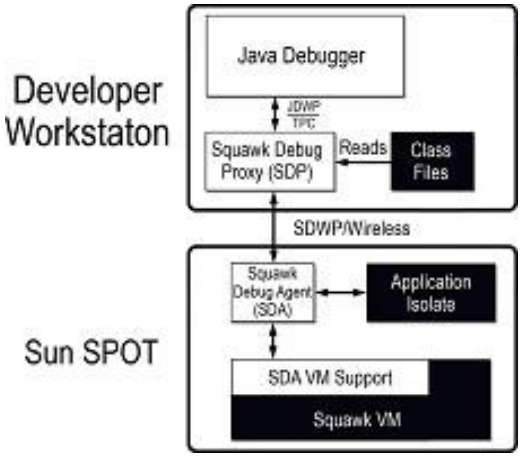
1.4.1.1. Εκσφαλμάτωση

Το Squawk JVM δίνει τη δυνατότητα οι εφαρμογές στα Sun SPOTs να εκσφαλματώνονται με τη χρήση Java περιβαλλόντων που υποστηρίζουν το Java Debug Wire Protocol, (JDWP), όπως το NetBeans και το JDB, (Java Debugger). Λόγω των περιορισμών που υπάρχουν στη μνήμη των SunSPOTs, το Squawk JVM δεν εφαρμόζει το JDWP απευθείας πάνω στο Sun SPOT αλλά μοιράζει τη διαδικασία της εκσφαλμάτωσης σε τρία στοιχεία.

- Έναν debug proxy,που τρέχει στο περιβάλλον του χρήστη
- Έναν debug agent, που επικοινωνεί με τον debug proxy και ελέγχει την εφαρμογή που εκσφαλματώνεται
- Μία μικρή debug agent υποστήριξη που παρέχεται από το ίδιο το Squawk JVM.

Ο debug proxy και ο debug agent επικοινωνούν μέσω ενός υποεπιπέδου του JDWP που ονομάζεται Squawk Debug Wire Protocol, (SDWP).

Αυτή η αρχιτεκτονική δίνει τη δυνατότητα σε στοιχεία της Java Platform Debugger Architecture, (JPDA), που απαιτεί αρκετή μνήμη, να αποθηκεύονται στο περιβάλλον ανάπτυξης του χρήστη και όχι στο Squawk JVM, ελαττώνοντας έτσι το overhead της μνήμης. Συγκεκριμένα ο debug proxy στο περιβάλλον ανάπτυξης του χρήστη έχει πρόσβαση στις αρχικές κλάσεις που εισήχθησαν σε suites στη συσκευή, συνεπώς έχει πρόσβαση και στους πίνακες, στις μεθόδους, στα πεδία και στα τοπικά ονόματα μεταβλητών που άλλαξαν σε μία suite.



Εικόνα 10: Αρχιτεκτονική εκσφαλμάτωσης Squawk JVM

Το Squawk JVM -από την κατασκευή του- περιλαμβάνει και ένα δεύτερο χαμηλότερου επιπέδου debugger για την εκσφαλμάτωση του υλικού των συσκευών. Καθένας από αυτούς τους debuggers αθροίζει ένα 10% overhead στο βρόχο μεταγλώττισης ελέγχοντας αν ένα breakpoint έχει οριστεί ή όχι [3] [8].

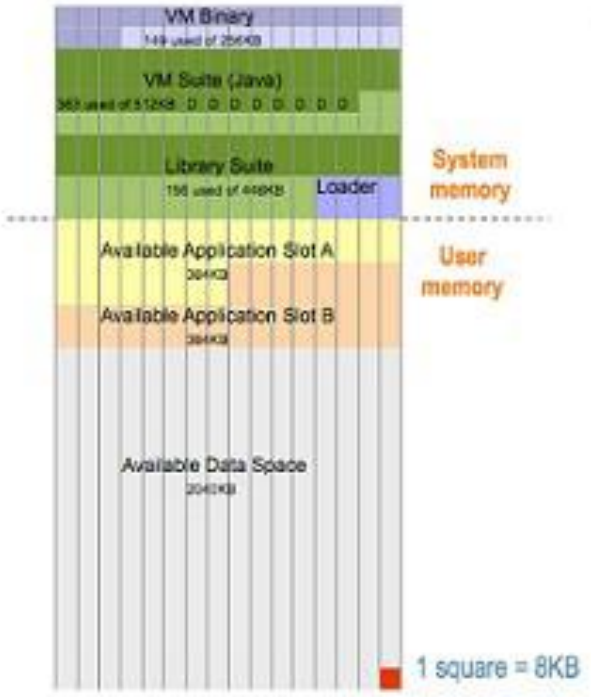
1.4.1.2. Flash μνήμη

Στα Sun SPOTs, τα 4MB flash μνήμης έχουν χαμηλή ισχύ και μπορούν να εκτελέσουν το μέγιστο ένα εκατομμύριο κύκλους ανά τομέα. Για κώδικα συστήματος διατηρείται το 1/3 αυτής της μνήμης χωρίς να χρησιμοποιείται ολόκληρο, ενώ τα 2/3 διατηρούνται για εφαρμογές και δεδομένα. Η εικόνα 11 δείχνει την κατανομή της μνήμης για τα διάφορα στοιχεία του Squawk JVM και τις σχετιζόμενες βιβλιοθήκες.

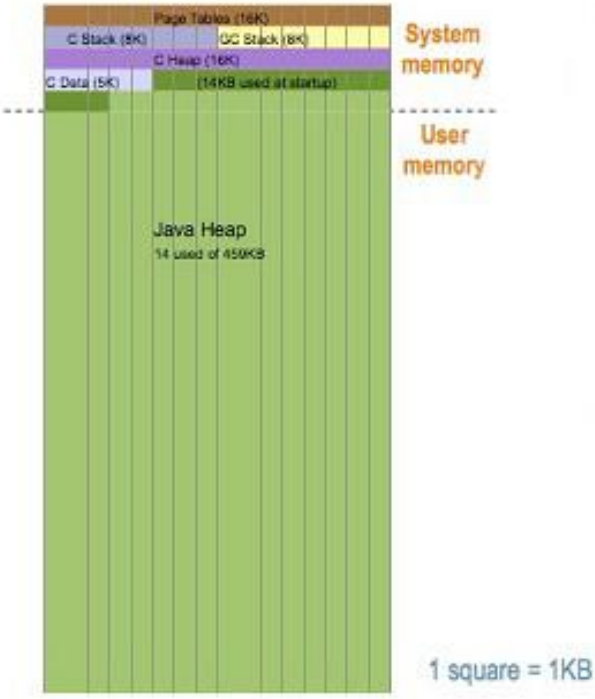
Αναλυτικά η κατανομή της μνήμης του συστήματος γίνεται ως εξής: 256 KB διατηρούνται για το δυαδικό Squawk JVM και από αυτά τα 149 KB βρίσκονται σε χρήση, 512 KB διατηρούνται για τις suites του Squawk JVM και από αυτά τα 363 KB βρίσκονται σε χρήση, εκ των οποίων τα 64 KB διατηρούνται για τον debugger, 448 KB διατηρούνται για τις βιβλιοθήκες των suites και από αυτά τα 156 KB βρίσκονται στη διάθεση του χρήστη και 64 KB διατηρούνται και χρησιμοποιούνται από τον bootloader. Η μνήμη που διατηρείται για το χρήστη αποτελείται από δύο μέρη, έκαστο εκ των οποίων αποτελείται από 384 KB, και από 2040 KB που διατίθενται για τις εφαρμογές [3] [10].

1.4.1.3. RAM μνήμη

Κάθε Sun SPOT έχει μνήμη 512 KB SRAM. Για τη μνήμη συστήματος διατηρείται λιγότερο από το 20% της SRAM και η υπόλοιπη μνήμη διατίθεται για τα αντικείμενα των εφαρμογών. Η εικόνα 12 δείχνει την κατανομή της μνήμης RAM. Αναλυτικά η κατανομή της μνήμης του συστήματος έχει ως εξής: 16 KB διατηρούνται για τους πίνακες σελίδων, 8 KB διατηρούνται για τη C στοίβα, 8 KB διατηρούνται για τον συλλέκτη απορριμμάτων, 16 KB διατηρούνται για τον C σωρό, 5 KB διατηρούνται για τα C δεδομένα και 14 KB διατηρούνται για την εκκίνηση. Ο Java σωρός διατηρεί 459 KB [3] [10].



Εικόνα 11: Κατανομή flash μνήμης



Εικόνα 12: Κατανομή RAM μνήμης

1.4.2. Βιβλιοθήκες

Στην ενότητα αυτή θα δούμε τα περιεχόμενα και την λειτουργία των βασικών βιβλιοθηκών του SUN SPOT, αυτές είναι: η βιβλιοθήκη συσκευών (device library), η βιβλιοθήκη για ασύρματη επικοινωνία (radio communication library) και η βιβλιοθήκη για τον έλεγχο του sensor board.

Device Library
Η βιβλιοθήκη συσκευών βρίσκεται στο spotlib_device.jar και στο spotlib_common.jar. Ο πηγαίος κώδικας της βιβλιοθήκης (των δυο παραπάνω jar) βρίσκεται στο spolib_source.jar στο φάκελο "Sun\SunSPOT\sdk\src" του sdk και περιέχει drivers για τις συσκευές.

Radio Communication Library
Η βιβλιοθήκη αυτή είναι υπεύθυνη για την δημιουργία και τον έλεγχο όλων των συνδέσεων και πρωτοκόλλων, όπως το radiostream και το radiogram. Οι κλάσεις που υλοποιούν τμήματα του radio stack πάνω από το MAC layer βρίσκονται στο multihoplib_rt.jar και ο αντίστοιχος πηγαίος κώδικας στο multihoplib_source.jar. Η παρούσα έκδοση του SUN SPOT SDK χρησιμοποιεί το GCF(Generic Connection Framework) για την δημιουργία συνδέσεων και την ασύρματη επικοινωνία μεταξύ των SPOTs χρησιμοποιώντας τα εξής δυο πρωτόκολλα:

- radiostream – Το radiostream παρέχει αξιόπιστη μετάδοση δεδομένων μεταξύ δυο κόμβων, επίσης χρησιμοποιεί buffers για καλύτερη απόδοση. Αποτελεί ένα stream-based πρωτόκολλο.
- Radiogram – Στο radiogram πρωτόκολλο η μεταφορά δεδομένων γίνεται με datagrams, και θεωρητικά δεν παρέχει καμία εγγύηση ότι τα πακέτα θα παραλειφθούν σωστά ή ότι θα φτάσουν στον προορισμό με την σειρά που στάλθηκαν. Όταν ένα πακέτο στέλνεται μέσω άλλων κόμβων(περισσότερα του ενός hop), υπάρχει περίπτωση να χαθεί χωρίς να παρέχεται κάποια ειδοποίηση, είτε να παραλειφθεί από τον προορισμό περισσότερες από μια φορές, είτε ακόμα να μην φτάσει με την σειρά που στάλθηκε. Στην περίπτωση όμως που τα datagrams στέλνονται σε κάποιο γειτονικό κόμβο (ένα hop), η μόνη περίπτωση είναι κάποια απλά να παραλειφθούν περισσότερες από μια φορές.

Sensor Board library
Η βιβλιοθήκη αυτή περιέχει όλες τις κλάσεις και τα interfaces που χρειαζόμαστε για τη διαχείριση των συστημάτων και των αισθητήρων του eDEMO Board. Αυτές βρίσκονται στο αρχείο transducerlib_rt.jar. Οι κλάσεις αυτής της βιβλιοθήκης μπορούν να χρησιμοποιηθούν για αλληλεπίδραση με το επιταχυνσιόμετρο, τα LED, τον αισθητήρα φωτεινότητας, τον αισθητήρα θερμοκρασίας, τις ψηφιακές θύρες εισόδου/εξόδου καθώς και τους διακόπτες. Για περισσότερες λεπτομέρειες προτρέπουμε τον αναγνώστη να διαβάσει τα αντίστοιχα τμήματα του javadoc [1] [2].

1.5. Προγραμματίσμός των SunSPOTs

1.5.1. MIDlets

MIDlets ονομάζονται οι εφαρμογές στα Sun SPOTs. Τα MIDlets χρησιμοποιούν τις προδιαγραφές του MIDP, (Mobile Information Device Profile), που ορίζονται από το πρωτόκολλο CLDC, (Connected Limited Device Configuration), και προορίζονται για

Java περιβάλλοντα σε μικροσυσκευές. Οι πλέον τυπικές εφαρμογές των MIDlets είναι τα παιχνίδια στα κινητά τηλέφωνα, τα οποία υποστηρίζουν γραφικά, χρήση πληκτρολογίου για επικοινωνία με το χρήστη και περιορισμένη σύνδεση στο Διαδίκτυο μέσω.
Στον κύκλο ζωής ενός MIDlet υπάρχουν τρεις πιθανές καταστάσεις:
1. **paused** – το MIDlet έχει δημιουργηθεί και είναι ανενεργό
2. **active** – το MIDlet είναι ενεργό
3. **destroyed** – το MIDlet έχει καταστραφεί και αναμένεται η αποκατάστασή του από τον συλλέκτη απορριμμάτων

Κάθε MIDlet είναι ένα σύνολο από κλάσεις που έχει σχεδιαστεί ώστε να ελέγχεται και να εκτελείται από το εκάστοτε σύστημα ελέγχου λογισμικού της κινητής συσκευής, δηλαδή στην περίπτωση των Sun SPOTs από το Squawk JVM. Κάθε MIDlet πρέπει να περιέχει μία κλάση, η οποία θα επεκτείνει την javax.microedition.midlet.MIDlet κλάση. Σε αυτήν την κλάση οι μέθοδοι θα επιτρέπουν στο σύστημα ελέγχου λογισμικού της κινητής συσκευής να δημιουργεί, να εκκινεί, να διακόπτει και να καταστρέφει την εφαρμογή. Παρομοίως, όλα τα MIDlets που εκτελούνται στα Sun SPOTs περιέχουν τις συναρτήσεις startApp(), pauseApp() και destroyApp().

Εκ κατασκευής το MIDlet βρίσκεται στην paused κατάσταση. Με την startApp() δεσμεύονται οι απαραίτητοι πόροι για την εκτέλεση της εφαρμογής και ενεργοποιείται το MIDlet. Κάθε φορά που η εκτέλεση γίνει ανενεργή ή συμβεί μία εξαίρεση, καλείται η pauseApp() και απελευθερώνονται προσωρινά οι αρχικά δεσμευμένοι πόροι. Με την ολοκλήρωση της εκτέλεσης της εφαρμογής καλείται η destroyApp(), η οποία αποδεσμεύει τους δεσμευμένους πόρους του MIDlet και καταστρέφει το MIDlet, δηλαδή αποκαθίσταται από τον συλλέκτη απορριμμάτων.

Τα MIDlets ενσωματώνονται σε suites. Μία MIDlet suite αποτελείται από ένα .jad αρχείο, (Java Application Descriptor), και ένα .jar αρχείο, (Java Archive). Το .jad αρχείο περιγράφει τη MIDlet suite, κοινώς περιλαμβάνει το όνομα της, απαιτήσεις και χαρακτηριστικά της λειτουργίας της, τη θέση και το μέγεθος του .jar αρχείου καθώς και προδιαγραφές που ορίζονται από το MIDP ή και από τον προγραμματιστή των MIDlets. Παρόμοια είναι η σύνταξη του .jad αρχείου με αυτή της java.util.Properties κλάσης.Επίσης το .jad αρχείο χρησιμοποιείται για να φορτώσει τις εφαρμογές στα Sun SPOTs μέσω OTA, (over-the-air). Στη μέθοδο αυτή τα .jad και .jar αρχεία μεταφορτώνονται από το Sun SPOT μέσω 67 σε έναν εξυπηρετητή δικτύου. Στη συνέχεια ο χρήστης κατεβάζει το .jad αρχείο και εγκαθιστά στο Sun SPOT τα MIDlets που απαιτούνται.

Το .jar αρχείο περιλαμβάνει ένα MANIFEST.MF αρχείο, το οποίο περιέχει τις απαραίτητες πληροφορίες για την εκτέλεση της εφαρμογής από το Squawk JVM,ουσιαστικά καθορίζει ποιες κλάσεις χρησιμοποιούν ποιά MIDlets. Όπως ειπώθηκε πριν, κάθε .jar αρχείο έχει αποκλειστικά και μόνο ένα MANIFEST.MF αρχείο και αυτό βρίσκεται στον υποκατάλογο META – INF [3] [11] [12] [13].

1.5.1.1. Μορφή του MIDlet στα SunSPOTs

Ένα MIDlet πρέπει να έχει συγκεκριμένη μορφή για να μπορεί να φορτωθεί και να εκτελεστεί σε ένα Sun SPOT. Συγκεκριμένα, ο γονικός κατάλογος που περιέχει το MIDlet, θα πρέπει να περιέχει και τα αρχεία build.xml και build.properties, που ελέγχουν το ant script τα οποία θα μεταγλωττίσει, εκσφαλματώσει και εκτελέσει την εφαρμογή. Ο γονικός αυτός κατάλογος θα πρέπει επίσης να περιέχει και τρεις υποκαταλόγους, τον src, τον nbproject και τον resources. Ο src θα περιέχει όλο τον πηγαίο κώδικα του MIDlet. Ο nbproject θα περιέχει αρχεία που έχουν δημιουργηθεί στην περίπτωση που ο κώδικας γραφτεί με χρήση του IDE Netbeans, (Integrated Development Environment Netbeans), και ο resources θα περιέχει το αρχείο MANIFEST.MF, το οποίο καθορίζει

χαρακτηριστικά του MIDlet. Προκειμένου να φορτωθεί σε Sun SPOTs και να εκτελεστεί ένα MIDlet θα πρέπει να έχει την παραπάνω μορφή [3] [6].

1.5.2. Apache Ant

Το Apache Ant, (Another Neat Tool), είναι μία scripting γλώσσα που έχει σχεδιαστεί για τη ταχύτερη και ευκολότερη εκσφαλμάτωση, μεταγλώττιση και εκτέλεση του κώδικα, αυτοματοποιώντας όλη αυτή τη διαδικασία, (build processing). Δηλαδή αποτελεί λογισμικό ανοιχτού κώδικα.

Για την εκτέλεση του το Ant χρησιμοποιεί την Java και απαιτεί Java περιβάλλοντα. Παρομοιάζει με το Make με τη διαφορά ότι για την περιγραφή της build διαδικασίας παράγει xml αρχεία. Από την κατασκευή του κάθε αρχείο ονομάζεται build.xml.

Αρχικά αναπτύχθηκε ως εργαλείο που θα αυτοματοποιούσε την build διαδικασία στον Apache Tomcat με χρήση xml αρχείων, ανεξαρτήτως πλατφόρμας. Ωστόσο, στη συνέχεια διαμορφώθηκε ως build εργαλείο για όλα σχεδόν τα Java projects.

Τα Sun SPOTs χρησιμοποιούν το Ant κατά την build διαδικασία. Δηλαδή κάθε φορά που φορτώνεται ένα MIDlet δημιουργείται ένα build.xml, το οποίο εκτελείται και τελικά εκτελείται και η εφαρμογή του MIDlet [3] [14] [15].

1.5.3. Πρωτόκολλα radiostream – radiogram

Το radiostream πρωτόκολλο παράγει ένα αξιόπιστο συνεχές ρεύμα εισόδου – εξόδου μεταξύ δύο συσκευών με χρήση sockets.
Με την εντολή

RadiostreamConnection conn =
(RadiostreamConnection)Connector.open(“radiostream:<destAddr>:<portNo>”);

ανοίγει μία σύνδεση. destAddr είναι η 64-bit διεύθυνση της συσκευής προορισμού και portNo είναι ο αριθμός της θύρας που χρησιμοποιείται ως αναγνωριστικό της συγκεκριμένης σύνδεσης και κυμαίνεται μεταξύ 0 και 255. Το 0 δε θεωρείται έγκυρη IEEE διεύθυνση σε αυτή την εφαρμογή. Η σύνδεση που ανοίγει χρησιμοποιεί εκ κατασκευής το κανάλι 26 και το PAN identifier 3. Για να εδραιωθεί η σύνδεση θα πρέπει και οι δύο συσκευές της σύνδεσης να χρησιμοποιήσουν το ίδιο portNo και τις αντίστοιχες IEEE διευθύνσεις. Όταν εδραιωθεί η σύνδεση μπορεί καθεμία από τις δύο συσκευές να λάβει και να αποστείλει δεδομένα με τις παρακάτω εντολές.

DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos = conn.openDataOutputStream();

Το radiogram πρωτόκολλο χρησιμοποιεί το μοντέλο πελάτη – εξυπηρετητή, (client – server), και παράγει datagrams μεταξύ των δύο συσκευών. Το πρωτόκολλο αυτό δεν παρέχει καμία εγγύηση για την παράδοση των datagrams. Τα datagrams ενδέχεται να μην παραδοθούν καθόλου ή να παραδοθούν με λάθος σειρά ή να παραδοθούν περισσότερες από μία φορές.

Με την εντολή

RadiogramConnection conn =
(RadiogramConnection)Connector.open(“radiogram://:<portNo>”);

ανοίγει ο εξυπηρετητή μία σύνδεση. portNo είναι ο αριθμός της θύρας που

χρησιμοποιείται ως αναγνωριστικό της συγκεκριμένης σύνδεσης και κυμαίνεται μεταξύ 0 και 255. Εκ κατασκευής χρησιμοποιούνται από τη σύνδεση το κανάλι 26 και το PAN identifier 3.
Με την εντολή

```
RadiogramConnection conn =
(RadiogramConnection)Connector.open("radiogram://<serveraddr>:<portNo>");
```

ανοίγει ο πελάτης μία σύνδεση. serveraddr είναι η 64-bit IEEE διεύθυνση του εξυπηρετητή και portNo είναι ο αριθμός της θύρας που χρησιμοποιείται ως αναγνωριστικό της συγκεκριμένης σύνδεσης και κυμαίνεται μεταξύ 0 και 255. Το 0 δε θεωρείται έγγυρη IEEE διεύθυνση σε αυτή την εφαρμογή. Ομοίως εξυπηρετητής και πελάτης πρέπει να χρησιμοποιούν το ίδιο portNo.
Με την εντολή

```
Datagram dg = conn.newDatagram(conn.getMaximumLength());
```

δημιουργείται ένα άδειο datagram από τη σύνδεση.

Με την εντολή

```
conn.send(dg);
```

αποστέλλεται ένα datagram και με την παρακάτω εντολή λαμβάνεται ένα datagram.

```
conn.receive(dg);
```

Ακολουθούν κάποια αξιοσημείωτα χαρακτηριστικά των datagrams:

- Datagrams που λαμβάνονται από μία σύνδεση δεν μπορούν να χρησιμοποιηθούν σε άλλη σύνδεση.
- Η διεύθυνση στην οποία πραγματοποιείται μία σύνδεση είναι χαρακτηριστική για τη σύνδεση. Προσπάθειες να αποσταλούν datagrams σε διαφορετική διεύθυνση από αυτή της σύνδεσης θα καταλήγουν σε εξαίρεση.
- Επιτρέπεται να ανοιχθούν συνδέσεις πελάτη και διακομιστή στο ίδιο μηχάνημα με χρήση του ίδιου αριθμού θύρας. Όλα τα εισερχόμενα σε αυτή τη θύρα datagrams θα δρομολογούνται στη σύνδεση του διακομιστή.
- Εάν κλείσει η σύνδεση του διακομιστή, κλείνουν και όλες οι συνδέσεις του πελάτη σε αυτό το διακομιστή.

Η broadcast επικοινωνία πραγματοποιείται με το radiogram πρωτόκολλο. Δεν είναι αξιόπιστη καθώς ενδέχεται τα datagrams να μην παραδοθούν καθόλου στους προορισμούς τους, αφού τα Sun SPOTs αγνοούν τις επιβεβαιώσεις των broadcast μηνυμάτων που τους στάλθηκαν ή που έστειλαν τα ίδια.

Με την εντολή

```
DatagramConnection conn =
(DatagramConnection)Connector.open("radiogram://broadcast:<portnum>");
```

ανοίγει μία radiogram σύνδεση για broadcast επικοινωνία. portnum είναι ο αριθμός της θύρας που θα χρησιμοποιηθεί. Οι broadcast συνδέσεις δεν είναι αμφίδρομες, χρησιμοποιούνται δηλαδή μόνο για να σταλθούν πακέτα και όχι να ληφθούν. Για να ληφθούν πακέτα από μία broadcast σύνδεση θα πρέπει να ανοιχθεί μία σύνδεση διακομιστή στον ίδιο αριθμό θύρας [3] [6] [11].

1.5.4. Ανάπτυξη εφαρμογής σε εικονικό SunSPOT

Το εργαλείο SPOTManager tool , στη σημερινή του μορφή προσφέρει οκτώ επιλογές:

1. Sun SPOTs: μας δίνει τη δυνατότητα μεμονωμένης αναζήτησης, φόρτωσης και διαμόρφωσης του λογισμικού τους.
2. SDKs: μας επιτρέπει την φόρτωση και εγκατάσταση των διαθέσιμων εκδόσεων του SunSPOT SDK από τον ιστότοπο του SunSPOT.
3. Solarium: προσφέρει ένα εργαλείο για τη διαχείριση μεμονωμένων SunSPOTs και την εγκατάσταση λογισμικού σε αυτά. Επιπλέον έχει έναν προσομοιωτή SunSPOT ο οποίος χρησιμοποιείται για τον έλεγχο του λογισμικού μιάς εφαρμογής SunSPOT καθώς και για την εκτέλεσή της.
4. Tutorial: επιτρέπει το διάβασμα του εγχειριδίου του.
5. Docs: προσφέρει δυνατότητα πλοήγησης στις οδηγίες για τα ενεργά SDK
6. Console: υποδικνύει την στάνταρτ έξοδο της χρήσης των εργαλείων του SPOTManager. Σε περίπτωση νέου σφάλματος εκτέλεσης, το αποτέλεσμα είναι κόκκινο, ενώ σε αντίθετη περίπτωση είναι μαύρο.
7. Share: επιτρέπει την ανάγνωση αλλά και την αξιολόγηση τμημάτων κώδικα καθώς επίσης και τη δημοσίευση δικών μας τμημάτων κώδικα.
8. Preferences: δύναται να διαμορφώνει το δίκτυο και τα χαρακτηριστικά του ίδιου του SPOTManager tool.

Επίσης διαθέτει τρία κουμπιά, στο κάτω μέρος του παραθύρου, ανεξαρτήτως από την επιλογή μας
1. Support: Επιτρέπει την αναφορά κάποιου σφάλματος στα SunSPOTs στο λογισμικό του SunSPOT ή στον SPOTManager. Μπορεί να συλλέγει και να περικλείει πληροφορίες για τα συνδεδεμένα SunSPOTs την ώρα της αναφοράς, εάν αυτό είναι επιθυμητό.
2. Quit: Απενεργοποίηση του εργαλείου
3. Forums: Ανοίγει ένα πλοηγό του διαδικτύου στην ιστοσελίδα του επίσημου φόρουμ του SunSPOT η οποία είναι η εξής: www.sunspotworld.com/forums/.

Εικόνα 13: Το εργαλείο SunSPOT Manager Tool

Επιλογή SunSPOTs:

Η επιλογή Sun SPOTS προσφέρει ένα μενού προκειμένου να επιλέξουμε μεταξύ των συνδεδεμένων με USB Sun SPOTs, μία μεγάλης έκτασης περιοχή για έξοδο κειμένου και οκτώ κουμπιά ειδικά για αυτή την επιλογή. Το μενού που υπάρχει επάνω από την περιοχή της κονσόλας λέγεται: “Select a Sun SPOT” και εμφανίζει μία λίστα με όλα τα συνδεδεμένα SunSPOTs μέσω USB στο σταθμό εργασίας, όπου το καθένα έχει τον IEEE αριθμό δικτύου του. Τα οκτώ πρώτα ψηφία του είναι πάντα τα ίδια, τα εξής 0014.4F01, ενώ τα τελευταία οκτώ ψηφία εμφανίζονται μέσα σε ένα αυτοκόλλητο το οποίο είναι ορατό μέσω του πλαστικού περιβλήματος της κεραίας του SunSPOT.

Τα οκτώ κουμπιά που διαθέτει είναι τα παρακάτω:

- Upgrade: Εκτελεί την εντολή ant upgrade στο επιλεγμένο SunSPOT και του φορτώνει το λογισμικό που απαιτείται για την τρέχουσα έκδοση του SDK.
- Properties: Δείχνει τις τρέχουσες ιδιότητες του συστήματος του επιλεγμένου SunSPOT και επιτρέπει την αλλαγή τους.
- SPOT Info: Τρέχει την εντολή ant info στο επιλεγμένο SunSPOT και εμφανίζει πληροφορίες οι οποίες αφορούν το λογισμικό του στην περιοχή της εξόδου.
- Echo: Εμφανίζει τα αποτελέσματα που απορρέουν από τη εκτέλεση του επιλεγμένου SunSPOT.
- Basestation: Εμφανίζει ένα μενού δίνοντας στο χρήστη τη δυνατότητα να ενεργοποιήσει ή να απενεργοποιήσει το SPOT που έχει επιλεχθεί ως σταθμό βάσης basestation. Αυτό κατά κανόνα συμβαίνει σε SunSPOT που δε διαθέτει

μπαταρία ή πίνακα αισθητήρων. Υπάρχει η δυνατότητα να επιλεχθεί απλό basestation (πιο αποτελεσματικό, μπορεί να επικοινωνεί με ένα τη φορά) ή του διαμοιραζόμενου (shared basestation), που μπορεί να επικοινωνεί με περισσότερα free-range SPOTs.

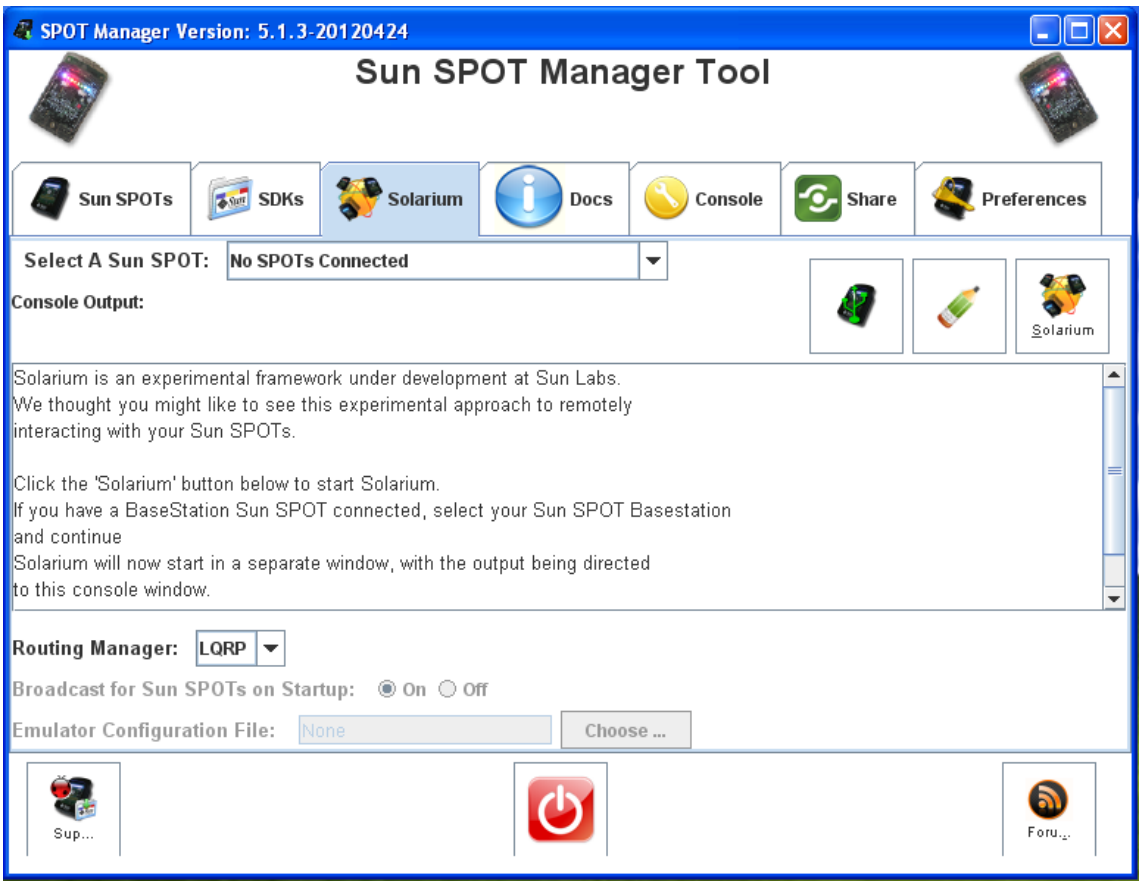
- OTA Command: Δύναται να ενεργοποιεί ή να απενεργοποιεί την εντολή Over The Air (OTA) επεξεργάζοντας το επιλεγμένο SPOT. Εμφανίζει το μενού τις επιλογές Enable OTA ή Disable OTA. Η επιλογή Enable OTA επιτρέπει να φορτώσουμε, να εκτελέσουμε και να διορθώσουμε τα σφάλματα σε εφαρμογές σε ένα Sun SPOT με τη χρήση της ραδιοσυχνότητας μεταξύ του basestation και του Sun SPOT που έχει επιλεχθεί. Σημειωτέον ότι το OTA command server θα πρέπει να ενεργοποιείται μόνο σε free-range SPOTs.
- Restore: Εκτελεί την εντολή ant restore στο Sun SPOT που επιλέχθηκε και το επαναφέρει στις αρχικές εργοστασιακές SDK ρυθμίσεις του.
- Clear: Διαγράφει όλο το κείμενο από το παράθυρο εξόδου [1] [2].

1.5.5. Επιλογή Solarium

Αυτή η επιλογή επιτρέπει την εκκίνηση μια εφαρμογής Solarium, δηλαδή μία εφαρμογή Java που χρησιμοποιείται για να εκτελέσει μια σειρά ενεργειών στα Sun SPOTs. Δίνει τις παρακάτω δυνατότητες:

- Ανακάλυψη και εμφάνιση Sun SPOTs συνδεδεμένων στον υπολογιστή μέσω usb ή μέσω ασύρματης επικοινωνίας.
- Αλληλεπίδραση με τα SPOTs: Δύναται να φορτώσει και να ξεφορτώσει λογισμικό από τα Sun SPOTs, να εκκινήσει, να διακόψει, να επανεκκινήσει, να τερματίσει εφαρμογές και να ανιχνεύσει την κατάσταση μίας συσκευής. Το πλαίσιο Radio View προσφέρει έναν γραφικό τρόπο αναπαράστασης της ασύρματης συνδεσιμότητας των Sun SPOTs.
- Διαχείριση ενός δικτύου από SPOTs: Δύναται να ελέγξει ένα δίκτυο από SunSPOTs με χρήση του πλαισίου Deployment View. Με το πλαίσιο αυτό καθορίζεται ποιά εφαρμογή πρέπει να φορτωθεί σε ποιά συσκευή και διευκολύνεται η διαδικασία φόρτωσης.
- Προσομοίωση SPOTs: Περιλαμβάνει έναν προσομοιωτή, που δύναται να φορτώσει και να εκτελέσει εικονικές εφαρμογές σε εικονικά Sun SPOTs.

Το Solarium δίνει τη δυνατότητα σε ένα basestation SPOT να επικοινωνήσει με πραγματικά SunSPOTs. Αν θέλουμε να επικοινωνήσουμε μόνο με προσομοιωμένα SPOTs δεν είναι απαραίτητο ένα basestation. Αν θέλουμε προσομοιωμένα SPOTs να επικοινωνήσουν ασύρματα με πραγματικά SPOTs απαιτείται ένα διαμοιραζόμενο basestation. Αν εκκινήσουμε το Solarium με το κουμπί με την ένδειξη Solarium ο SPOTManager εμφανίζει ένα κουτί διαλόγου στο οποίο πρέπει να ορίσουμε ποιό από τα συνδεδεμένα με USB SPOTs θέλουμε να είναι ο basestation.



Εικόνα 14: Το εργαλείο Solarium

Παρακάτω, θα αναλύσουμε δύο από τις βασικές λειτουργίες του Solarium, την ανακάλυψη και εμφάνιση των SPOTs και την προσομοίωση των SPOTs.

Ανακάλυψη και εμφάνιση των SPOTs

Το Solarium δύναται να εντοπίσει και να εμφανίσει τα SPOTs που είναι συνδεδεμένα στην επιφάνεια εργασίας μέσω USB όπως επίσης και τα συνδεδεμένα ασύρματα κοντινά SPOTs. Για να είναι ορατά μέσω ασύρματης επικοινωνίας, τα SPOTs θα πρέπει να έχουν ενεργοποιημένο το OTA (OTA command server) τους, που κανονικά είναι ενεργοποιημένο από μόνο του. Στην περίπτωση που αυτό δεν ισχύει, ο χρήστης δύναται να το ενεργοποιήσει συνδέοντας το SPOT μέσω USB και είτε πληκτρολογώντας την εντολή ant enableota σε μία γραμμή εντολών είτε πατώντας το κουμπί με την ένδειξη "Enable OTA" του SPOT Manager. Η ανακάλυψη και εμφάνιση των SPOTs εμπεριέχει και δυνατότητες όπως: ο έλεγχος των SPOT που εντοπίζονται, ο έλεγχος του τρόπου με τον οποίο εμφανίζονται, ο έλεγχος του ποιά θα εμφανίζονται, κ.ά.

Προσομοίωση SPOTs

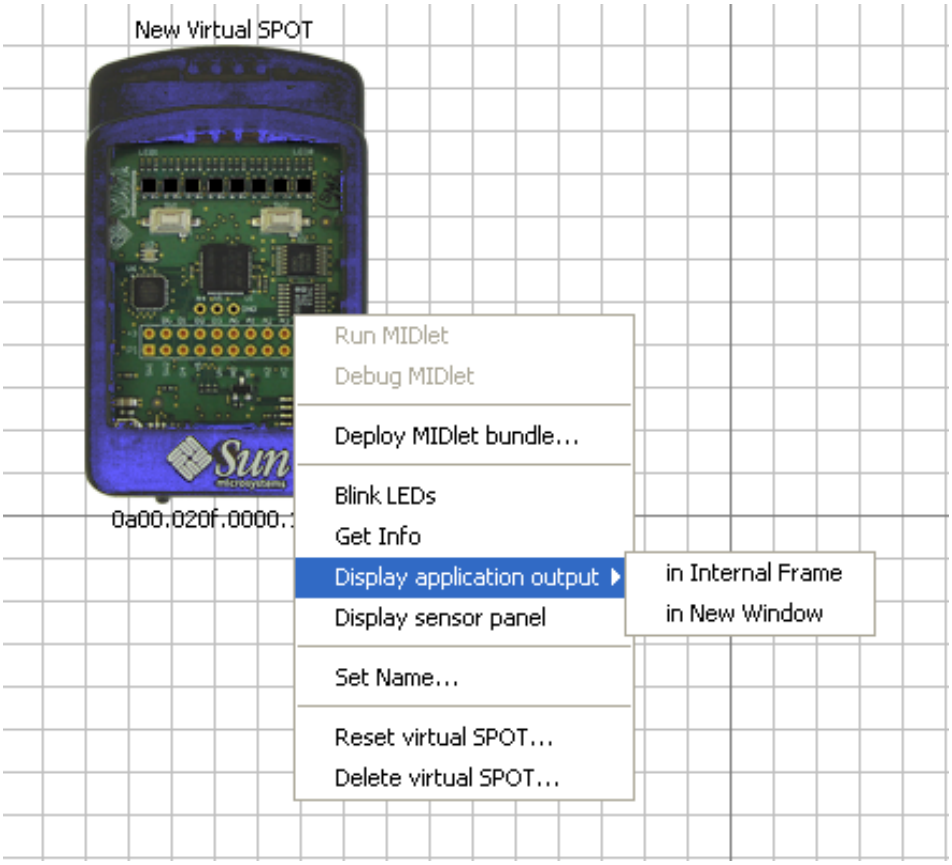
Το Solarium περιλαμβάνει έναν προσομοιωτή, δίνοντας τη δυνατότητα να εκτελεί εφαρμογές για SunSPOT στον υπολογιστή μας. Μας επιτρέπει έτσι να ελέγχουμε την ορθότητα ενός προγράμματος πριν το φορτώσουμε σε ένα SPOT ή να προσομοιώσουμε ένα SPOT αν δεν διαθέτουμε ένα. Τα εικονικά SPOT διαθέτουν μία οθόνη αισθητήρων, αντί για πίνακα αισθητήρων, που μπορεί να χρησιμοποιηθεί για την εισαγωγή των επιδιωκομένων τιμών των αισθητήρων (π.χ. επίπεδο φωτός, θερμοκρασία, ψηφιακές είσοδοι, αναλογικές τιμές ρεύματος εισόδου, και τιμές του επιταχυνσιόμετρου). Η εφαρμογή δύναται να ελέγξει το χρώμα των LEDs, όπως θα

έκανε ένα πραγματικό SPOT. Με το πάτημα των κουμπιών στους διακόπτες της εικόνας μπορεί κανείς να αναβοσβήσει τους διακόπτες. Επίσης παρέχεται η δυνατότητα αποστολής και παραλαβής μηνυμάτων μέσω ραδιοσυχνότητας, δεδομένου ότι κάθε SPOT διαθέτει δική του διεύθυνση και μπορεί να μεταδώσει σε ένα ή περισσότερα SPOT πληροφορίες.

Διαχείριση των virtual SPOTs

Έχοντας ανοίξει το γραφικό περιβάλλον του Solarium, επιλέγοντας από τη γραμμή εργαλείων το View > SPOT View, έπειτα, από το μενού File-> New virtual Spot δημιουργούμε ένα νέο virtual Spot. Κάνοντας δεξί κλικ επάνω εμφανίζεται ένα μενού με τις διαθέσιμες επιλογές:

- Set Name: μπορούμε να δώσουμε ένα όνομα στο virtual SPOT για να εμφανίζεται από κάτω μαζί με την IEEE διεύθυνση δικτύου.
- Deploy MIDlet bundle: μπορούμε να φορτώσουμε μια εφαρμογή για το virtual SPOT επιλέγοντας ένα jar αρχείο που έχει δημιουργηθεί κατά τη διαδικασία του build.
- Run MIDlet : έχοντας φορτώσει κάποια MIDlets μπορούμε να χρησιμοποιήσουμε αυτή την εντολή που τα εμφανίζει όλα και να επιλέξουμε ποιό θα εκτελέσουμε. Το MIDlet που είναι υπό εκτέλεση θα εμφανίζεται σε ένα κουτί δίπλα στο virtual SPOT. Κλικάροντας επάνω του εμφανίζεται η επιλογή διακοπής της εκτέλεσής του.
- Debug MIDlet : παρουσιάζει όλα τα MIDlets που περιέχονται στο αρχείο με το φορτωμένο jar και επιτρέπει να συνδέσουμε έναν εξωτερικό Java εκσφαλματωτή.
- Reset virtual SPOT: σταματάει όλα τα MIDlets που εκτελούνται και επανεκκινεί την Squawk VM. Έχουμε έτσι τη δυνατότητα να επαναφορτώσουμε οποιοδήποτε αρχείο jar έχει δημιουργηθεί νωρίτερα και να εκτελέσουμε όποιο από τα MIDlets επιθυμούμε.
- Display application output : εμφανίζει ένα νέο παράθυρο όπου τυπώνεται η έξοδος του SPOT application στο System.out ή στο System.err.
- Get info : εμφανίζει ένα παράθυρο πληροφοριών για το virtual SPOT: την IEEE διεύθυνσή του, το αρχείο jar που έχει φορτωθεί και τα ονόματα των διαθέσιμων MIDlets.
- Delete virtual SPOT : διαγράφει ένα virtual SPOT.



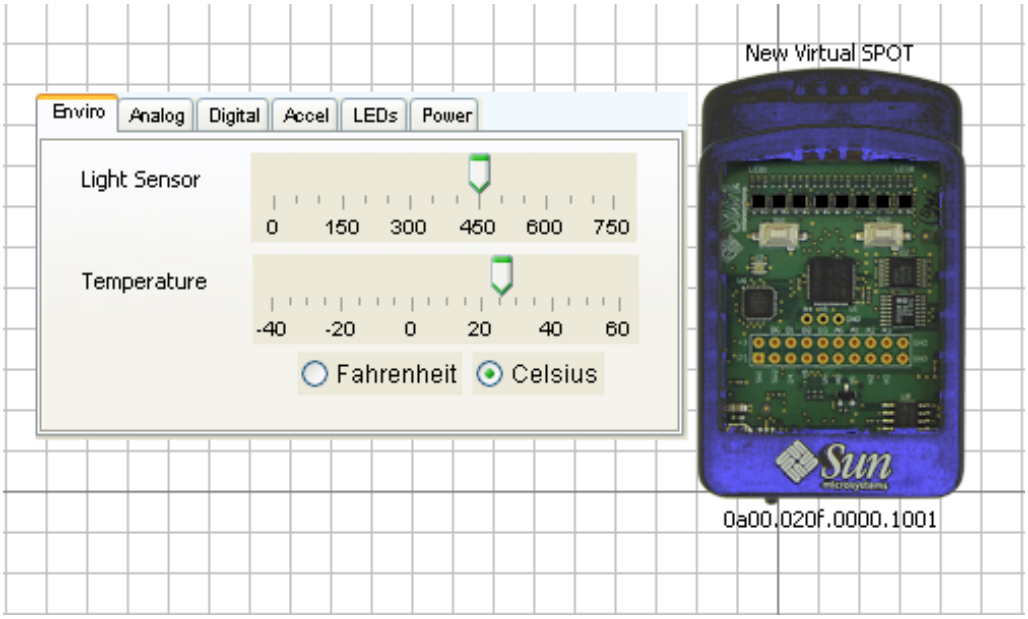
Εικόνα 15: Επιλογές Διαχείρισης ενός Virtual SPOT

Μέσω της κύριας γραμμής εργαλείων παρέχεται η δυνατότητα να αποθηκεύσει κανείς την κατάσταση στην οποία βρίσκονται όλα τα virtual SPOTs που δημιούργησε πηγαίνοντας Emulator->Save virtual configuration... .Τοιουτοτρόπως αποθηκεύονται, το όνομα και η διεύθυνση του κάθε virtual SPOT, ποιό jar αρχείο χρησιμοποιεί, ποιά MIDlets εκτελούνται, που είναι τοποθετημένο το virtual SPOT καθώς και μία περιγραφή.

Η επιλογή Open virtual SPOT... επιτρέπει την επιλογή του αρχείου που σώσαμε προηγουμένως και την επαναφόρτωσή του. Παράλληλα εμφανίζεται και το κείμενο περιγραφής που είχαμε σώσει, το οποίο μπορεί να επανεμφανιστεί με την επιλογή Display virtual configuration description. Τέλος, με τη χρήση της εντολής Delete all virtual SPOTs... μπορούμε να διαγράψουμε όλα τα virtual SPOTs που έχουμε καθορίσει στο Solarium.

Οθόνη πίνακα αισθητήρων

Αξίζει να σημειωθεί ότι τα Virtual SPOTs διαθέτουν μία οθόνη πίνακα αισθητήρων ως εναλλακτικό μηχανισμό για να παίρνουν τιμές οι αισθητήρες, αντί για πίνακα αισθητήρων που έχουν τα πραγματικά SPOTs. Έτσι, επιλέγοντάς το εμφανίζεται ένα παράθυρο με έξι επιλογές: Enviro, Analog In, Accel, Digital, LEDs και Power. Καθεμία δε, αντιστοιχεί σε εισόδους των αισθητήρων του virtual SPOT και μας επιτρέπει να δώσουμε τις ανάλογες τιμές.



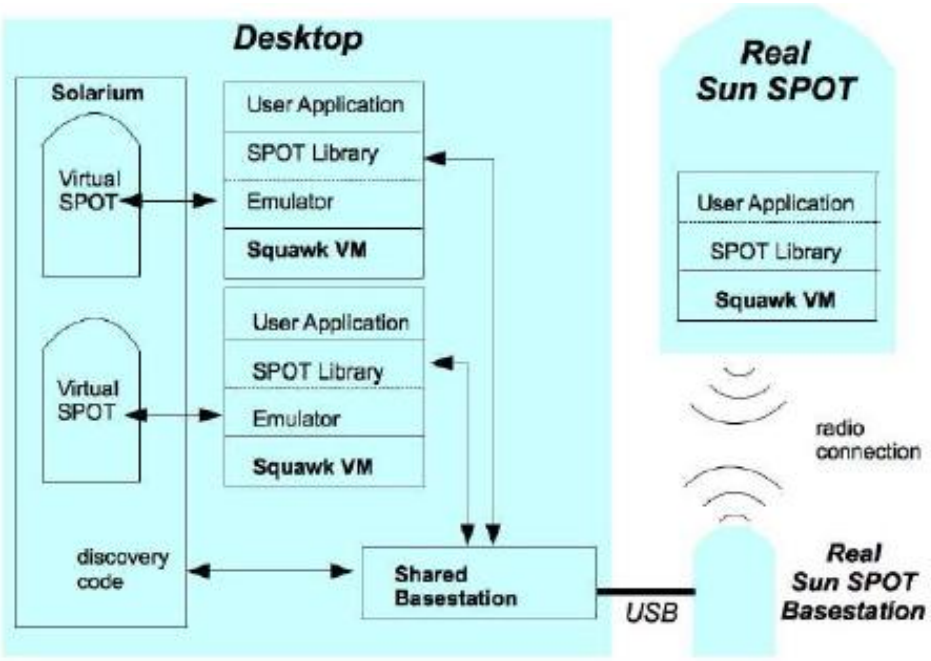
Εικόνα 16: Οθόνη πίνακα αισθητήρων

Χρησιμοποιώντας τη ραδιοσυχνότητα

Τα Virtual SPOTs δύνανται να επικοινωνούν μεταξύ τους εγκαθιδρύοντας ραδιοσυνδέσεις είτε broadcast είτε point-to-point. Όταν είναι συνδεδεμένος ένας σταθμός βάσης (basestation SPOT) στον υπολογιστή και εκτελείται ένας διαμοιραζόμενος σταθμός βάσης, (shared basestation), τα virtual SPOTs μπορούν να επικοινωνούν και με πραγματικά κάνοντας χρήση της ραδιοσυχνότητας του basestation. Υπάρχει το πλεονέκτημα ότι πολλές host applications μπορούν να έχουν πρόσβαση στη ραδιοσυχνότητα. Από την άλλη υπάρχει το μειονέκτημα ότι η επικοινωνία μεταξύ ενός host application και του SPOT στόχου κοστίζει δύο radio hops (βήματα ραδιοσυχνότητας), σε αντίθεση με το ένα radio hop (βήμα ραδιοσυχνότητας) που χρειάζεται ένας αποκλειστικός σταθμός βάσης (dedicated basestation).Κατά την εκτέλεση προκύπτει ένα άλλο μειονέκτημα ότι δεν είναι δυνατή η διαχείριση της ραδιοκεραίας του basestation SPOT, του pan id ή της ισχύος της εξόδου. Για να χρησιμοποιούμε πάντα shared basestation αρκεί να προσθέσουμε την ακόλουθη γραμμή στο αρχείο .sunspot.properties:basestation.shared=true

Τρόπος λειτουργίας του προσομοιωτή

Δημιουργώντας ένα νέο virtual SPOT στο Solarium, ξεκινάει μία νέα διαδικασία προκειμένου να εκτελέσει τον κώδικα του προσομοιωτή σε μία Squawk VM. Ο κώδικας του επικοινωνεί μέσω μίας σύνδεσης socket με τον κώδικα του virtual SPOT GUI στο Solarium. Π.χ., όταν μια SPOT εφαρμογή αλλάζει την τιμή RGB ενός LED, αυτή η πληροφορία διοχετεύεται στον κώδικα του SPOT GUI ο οποίος ανανεώνει την εμφάνιση αυτού του LED δίνοντάς του τη νέα RGB τιμή. Παρομοίως, κλικάροντας ο χρήστης έναν από τους διακόπτες του virtual SPOT, το Solarium στέλνει ένα μήνυμα στον κώδικα του προσομοιωτή ότι ο διακόπτης έχει πατηθεί, πράγμα που μπορεί να παρατηρηθεί από την εφαρμογή του SPOT.
Ακολουθεί το διάγραμμα της αρχιτεκτονικής του προσομοιωτή:



Εικόνα 17: Αρχιτεκτονική ενός προσομοιωτή

Κάθε virtual SPOT έχει τη δική του Squawk VM να εκτελείται ως μία ξεχωριστή διεργασία στον host υπολογιστή. Κάθε Squawk VM περιέχει μία πλήρη στοίβα σαν μέρος της βιβλιοθήκης του SPOT, επιτρέποντας την επικοινωνία της εφαρμογής του SPOT με άλλες εφαρμογές για SPOT που εκτελούνται στον υπολογιστή όπως π.χ. άλλα virtual SPOTs χρησιμοποιώντας sockets ή πραγματικά SPOTs μέσω ραδιοσυχνότητας εάν εκτελείται ένας διαμοιραζόμενος σταθμός βάσης (shared basestation) [3] [16].

1.5.6. Αισθητήρες

Οι αισθητήρες που περιλαμβάνονται στα Sun SPOTs είναι αισθητήρες φωτισμού, θερμοκρασίας και κίνησης. Για να χρησιμοποιηθεί οποιοσδήποτε από τους παραπάνω αισθητήρες αλλά και οτιδήποτε υπάρχει πάνω στην κεντρική πλακέτα του Sun SPOT, όπως LEDs, αναλογικές είσοδοι κ.α. πρέπει να χρησιμοποιηθεί το πακέτο com.sun.spot.sensorboard.EdemoBoard. Πρέπει επίσης να χρησιμοποιηθεί το πακέτο com.sun.spot.sensorboard.peripheral, το οποίο περιέχει διεπαφές για συγκεκριμένες εφαρμογές διάφορων περιφερειακών μονάδων που συνδέονται με την eDemo πλακέτα [3] [17] [18].

1.5.6.1. Αισθητήρας φωτισμού

Ο αισθητήρας φωτισμού ανιχνεύει την ένταση του φωτισμού και επιστρέφει έναν ακέραιο που κυμαίνεται από το 0 μέχρι το 750. Το 0 αντικατοπτρίζει το απόλυτο σκοτάδι. Η μέγιστη ευαισθησία του αισθητήρα αγγίζει μήκη κυμάτων 600 nm. Για να δημιουργήσουμε ένα αντικείμενο αισθητήρα φωτισμού χρησιμοποιείται ο παρακάτω κώδικας.

```
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.ILightSensor;
ILightSensor ourLightSensor = EdemoBoard.getInstance().getLightSensor();
```

Για να πάρουμε τιμές από τον αισθητήρα φωτισμού γίνεται χρήση του παρακάτω κώδικα [3] [17] [18].

```
int lightSensorReading = ourLightSensor.getValue();
```

1.5.6.2. Αισθητήρας θερμοκρασίας

Ο αισθητήρας θερμοκρασίας ανιχνεύει την ένταση της θερμοκρασίας περιβάλλοντος και επιστρέφει έναν ακέραιο που κυμαίνεται μεταξύ 0 και 1023. Ο αριθμός αυτός δεν αντιπροσωπεύει τη θερμοκρασία ούτε σε Celsius ούτε σε Fahrenheit βαθμούς, είναι ένας ακατέργαστος αριθμός. Το μοντέλο του αισθητήρα φωτισμού είναι ADT7411 με έναν 10-bit αναλογικό-σε-ψηφιακό μετατροπέα ο οποίος μπορεί να ανιχνεύσει θερμοκρασία περιβάλλοντος που κυμαίνεται από -40 σε +125 βαθμούς Celsius. Για να δημιουργηθεί ένα αντικείμενο αισθητήρα θερμοκρασίας χρησιμοποιείται ο παρακάτω κώδικας.

```
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.io.ITemperatureInput;
ITemperatureInput ourTempSensor = EdemoBoard.getADCTemperature();
```

Για να πάρουμε τιμές από τον αισθητήρα θερμοκρασίας χρησιμοποιείται ο παρακάτω κώδικας.

```
// ακατέργαστη τιμή θερμοκρασίας
int temp = tempSensor.getValue();
// τιμή θερμοκρασίας σε Celsius
double celsiusTemp = ourTempSensor.getCelsius();
// τιμή θερμοκρασίας σε Fahrenheit
double fahrenheitTemp= ourTempSensor.getFahrenheit();
```

Για πηγές φωτισμού, όπως οι λάμπες φθορισμού, οι οποίες φαίνονται στο ανθρώπινο μάτι σταθερές αλλά στην πραγματικότητα η έντασή τους αλλάζει συνέχεια, χρησιμοποιείται η παρακάτω εντολή,

```
int getAverageValue(int n);
```

όπου λαμβάνονται n μετρήσεις, καθεμία κάθε 1 msec και υπολογίζεται η μέση ένταση φωτισμού τους.

Ο αισθητήρας θερμοκρασίας των Sun SPOTs βρίσκεται αναπόφευκτα κοντά σε πηγές θερμότητας πάνω στην πλακέτα των Sun SPOTs και ιδεατά πιο έγκυρες μετρήσεις θα προέκυπταν από τη χρήση ενός εξωτερικού αισθητήρα θερμοκρασίας που θα συνδεόταν στις θύρες εισόδου – εξόδου της πλακέτας των Sun SPOTs [3] [17] [18].

1.5.6.3. Αισθητήρας κίνησης (Επιταχυνσιόμετρο)

Το επιταχυνσιόμετρο μετράει την επιτάχυνση της βαρύτητας, ανιχνεύει δηλαδή την κίνηση σε τρεις διαστάσεις. Το μοντέλο του επιταχυνσιόμετρου είναι LIS3L02AQ και αποτελείται από έναν μικροαισθητήρα MEMS, (Micro-Electro-Mechanical System), ο οποίος μετατοπίζεται από την ονομαστική του θέση όταν εφαρμόζεται μία γραμμική επιτάχυνση, προκαλώντας έτσι μία ηλεκτρική δυσαναλογία, η οποία διαβάζεται τελικά από τον αναλογικό-σε-ψηφιακό μετατροπέα της κεντρικής πλακέτας των Sun SPOTs. Διαθέτει δύο τρόπους λειτουργίας που καθορίζουν το εύρος των τιμών που θα μετρήσουν, από -2g έως 2g και από -6g έως 6g. Η ακατέργαστη τιμή που λαμβάνει το επιταχυνσιόμετρο είναι ένας αριθμός που κυμαίνεται από το 0 μέχρι το 1023 και μετατρέπεται στη συνέχεια σε g δυνάμεις. Η μηδενική βαρύτητα βρίσκεται περίπου στο 466,5 [3] [17] [18].

Το επιταχυνσιόμετρο μετράει την επιτάχυνση σε τρεις διαστάσεις (x, y, z).

Για να δημιουργηθεί ένα αντικείμενο του επιταχυνσιόμετρου χρησιμοποιείται ο παρακάτω κώδικας.

```
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.IAccelerometer3d;
IAccelerometer3D ourAccel = EdemoBoard.getInstance().getAccelerometer();
```

Για να λάβει δεδομένα το επιταχυνσιόμετρο, θα πρέπει να οριστεί το εύρος τιμών στο οποίο θα μετρήσει. Το 0 θέτει το εύρος τιμών μεταξύ -2g και 2g, και το 1 θέτει το εύρος τιμών μεταξύ -6g και 6g. Στη συνέχεια το επιταχυνσιόμετρο θα πρέπει να μείνει ανενεργό για 100 milliseconds ώστε οι μετρήσεις του να είναι έγκυρες. Ακολουθεί ο σχετικός κώδικας.

```
// 0 είναι 2g, 1 είναι 6g
ourAccel.setRange(0); //χρησιμοποίησε 2g εύρος τιμών
// παύση για 100 milliseconds ώστε οι μετρήσεις να είναι έγκυρες
try{
Thread.sleep(100);
} catch (InterruptedException ex) {
// αγνόησε εξαιρέσεις
}
```

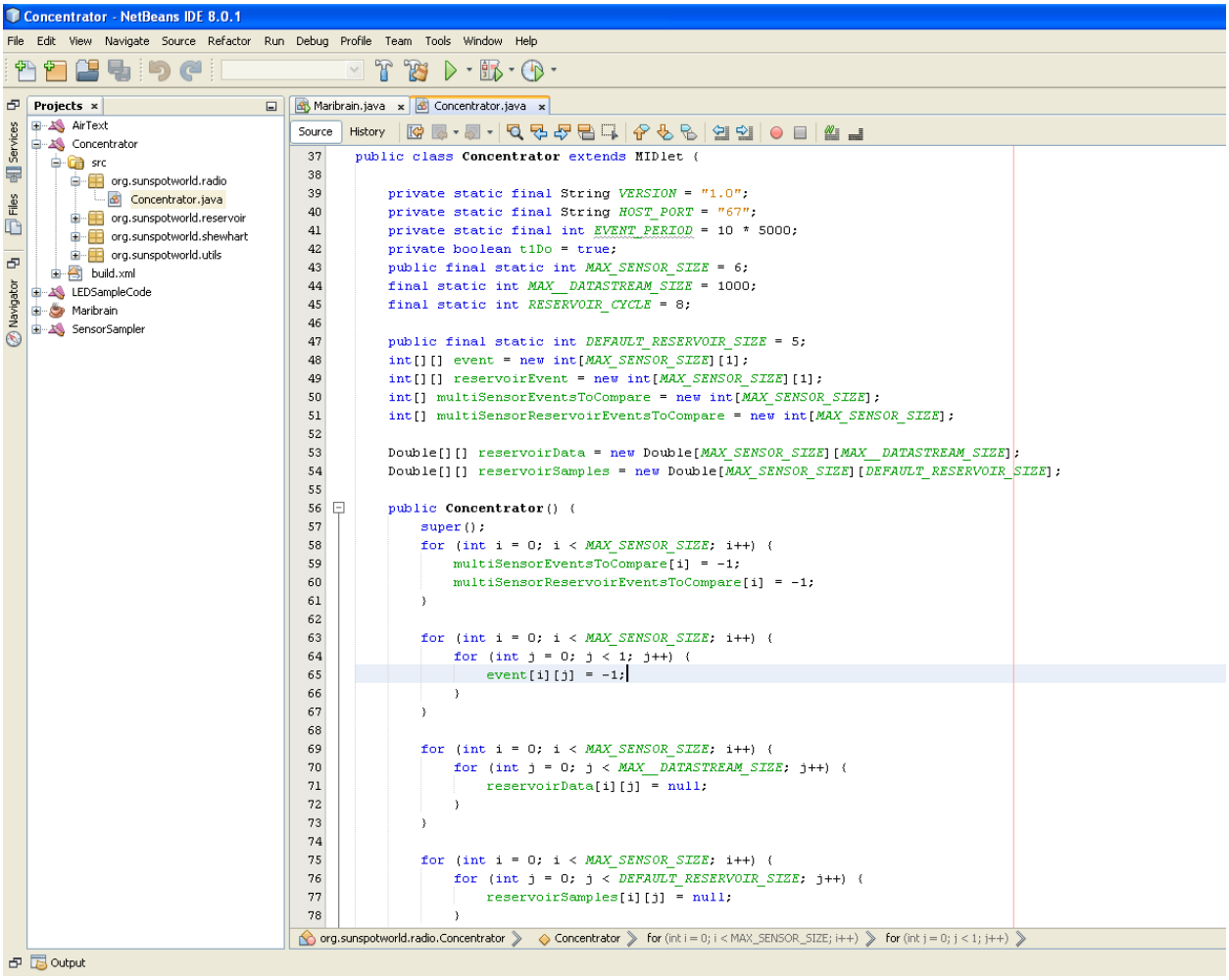
Για να πάρουμε τιμές από το επιταχυνσιόμετρο χρησιμοποιείται ο παρακάτω κώδικας.

```
int x-accel = ourAccel.getXaxis().getValue();
int y-accel = ourAccel.getYaxis().getValue();
int z-accel = ourAccel.getZaxis().getValue();
```

1.5.7. NetBeans

Η υλοποίηση του λογισμικού πραγματοποιήθηκε στο περιβάλλον ανάπτυξης NetBeans IDE, (Integrated Development Environment). Το NetBeans είναι μία ανοιχτή πλατφόρμα ανάπτυξης λογισμικού για εφαρμογές που έχουν γραφτεί κυρίως σε Java αλλά υποστηρίζει και άλλες γλώσσες, ιδίως PHP, C/C++ και HTML5.

Το NetBeans IDE έχει γραφτεί σε Java και μπορεί να εκτελέσει εφαρμογές σε οποιοδήποτε λειτουργικό που βρίσκεται προεγκατεστημένο κάποιο JVM (Java Virtual Machine). Για την ανάπτυξη Java προγραμμάτων είναι απαραίτητη και η εγκατάσταση JDK, (Java Development Kit) [19] [20]. Στην παρούσα εφαρμογή εγκαταστάθηκε και χρησιμοποιήθηκε η έκδοση NetBeans IDE 8.0.1.



Εικόνα 18: Εργαλείο ανάπτυξης λογισμικού NetBeans

2. Αναλυτική περιγραφή των αλγορίθμων

2.1. Shewhart algorithm (Control Chart)

Το διάγραμμα ελέγχου Shewhart έχει μια βάση αναφοράς καθώς και ανώτατα και κατώτατα όρια, που εμφανίζονται ως διακεκομμένες γραμμές, οι οποίες είναι συμμετρικές γύρω από τη βασική γραμμή. Οι μετρήσεις απεικονίζονται στο γράφημα έναντι μιας γραμμής χρόνου. Οι μετρήσεις που βρίσκονται έξω από τα όρια που θεωρούνται εκτός ελέγχου.

Τα άνω (Upper Control Limit = UCL) και κάτω (Lower Control Limit = LCL) όρια ελέγχου είναι:

UCL = αποδεκτή τιμή + k * *διαδικασία τυπικής απόκλισης*

LCL = αποδεκτή τιμή - k * *διαδικασία τυπικής απόκλισης*

όπου η *διαδικασία τυπικής απόκλισης* είναι η *τυπική απόκλιση* που υπολογίζεται από τον βασικό έλεγχο της βάσης δεδομένων.

Αυτή η διαδικασία είναι ένα διάγραμμα ελέγχου μεμονωμένων παρατηρήσεων. Τα προηγουμένως περιγραφέντα διαγράμματα ελέγχου εξαρτώνται από ορθολογικά υποσύνολα, τα οποία χρησιμοποιούν τις τυπικές αποκλίσεις που υπολογίζονται από ορθολογικά υποσύνολα για τον υπολογισμό των ορίων ελέγχου. Για μια διαδικασία μέτρησης, οι υποομάδες θα αποτελούνταν από βραχυπρόθεσμες επαναλήψεις οι οποίες μπορεί να χαρακτηρίζουν την ακρίβεια του οργάνου, αλλά όχι την μακροπρόθεσμη μεταβλητότητα της διαδικασίας. Στην επιστήμη της μέτρησης, το ενδιαφέρον επικεντρώνεται στην αξιολόγηση μεμονωμένων μετρήσεων (ή μέσων όρων των βραχυπρόθεσμων επαναλήψεων). Έτσι, η τυπική απόκλιση με την πάροδο του χρόνου είναι το κατάλληλο μέτρο της μεταβλητότητας.

Για την επίτευξη αυστηρού ελέγχου της διαδικασίας μέτρησης, θέτουμε

k = 2

όπου σε αυτή την περίπτωση περίπου 5% των μετρήσεων από μια διαδικασία που έχει τον έλεγχο θα παράγουν εκτός-ελέγχου σήματα. Αυτό προϋποθέτει ότι υπάρχει ένας αρκετά μεγάλος αριθμός βαθμών ελευθερίας (> 100) για την εκτίμηση της διαδικασίας τυπικής απόκλισης.
Για να επισημάνουμε μόνο εκείνες οι μετρήσεις οι οποίες είναι κατάφωρα εκτός ελέγχου, θέτουμε

k = 3

όπου σε αυτή την περίπτωση περίπου 1% των μετρήσεων από μια διαδικασία ελέγχου σε θα παράγει εκτός-ελέγχου σήματα [21].

Διαφορετικά, στο διάγραμμα ελέγχου Shewhart, μια μεταβλητή x_k διαπιστώνεται ότι αποκλίνει από τις κανονικές τιμές σε κάποια δεδομένη χρονική στιγμή με βάση τα δύο όρια ελέγχου: το ανώτατο όριο ελέγχου (UCL) και το κατώτατο όριο ελέγχου (LCL). Τα όρια ελέγχου ορίζονται ως η απόσταση από την τρέχουσα μέση τιμή της ακολουθίας $x_1,\ldots,x_k$ η οποία είναι:

- $\bar{x}$ + $\alpha\sigma_k$ for UCL, and
- $\bar{x}$ - $\alpha\sigma_k$ for LCL

Δηλαδή, η τιμή x_k ανιχνεύεται ότι θα «χτυπήσει» έναν συναγερμό ή αλλιώς θα σημάνει ένα συμβάν αν $x_k > UCL_k$ ή $x_k < LCL_k$. Οι τιμές UCL_k and LCL_k ορίζονται ως εξής (με σταδιακά αυξανόμενες μεθόδους):

$\bar{x}_k = \bar{x}_{k-1} + (x_k - \bar{x}_{k-1})/k$

και

$\sigma^2_k = ((k-1)\,\sigma^2_{k-1} + (x_k - \bar{x}_k)\,(x_k - \bar{x}_{k-1}))/k$

Το διάγραμμα ελέγχου Shewhart επιστρέφει +1 αν $x_k > UCL_k$, -1 αν $x_k < LCL_k$ and ομαλότητα δηλαδή 0, αν $x_k \in (LCL, UCL)$. Η παράμετρος α ορίζεται να είναι a = 3. (Στην δική μας υλοποίηση το α συμβολίζεται με κ) [22].

2.2. Αλγόριθμος Αποθέματος (Reservoir Algorithm)

Η τυχαία δειγματοληψία είναι βασική σε πολλές υπολογιστικές εφαρμογές στην επιστήμη των υπολογιστών, στη στατιστική και στην μηχανική. Το πρόβλημα έγκειται στην επιλογή χωρίς αντικατάσταση ενός τυχαίου δείγματος μήκους n από ένα σύνολο N. Για παράδειγμα, μπορεί να θέλουμε ένα τυχαίο δείγμα n εγγραφών από μια ομάδα N εγγραφών, ή μπορεί να θέλουμε ένα τυχαίο δείγμα n ακεραίων αριθμών από το σύνολο {1,2,…,N}.

Πολλοί αλγόριθμοι έχουν αναπτυχθεί για αυτό το πρόβλημα όταν η τιμή του N είναι γνωστή εξαρχής. Σε αυτή την εργασία μελετάμε ένα πολύ διαφορετικό πρόβλημα: δειγματοληψία όταν το N είναι άγνωστο και δεν μπορεί να προσδιοριστεί αποτελεσματικά. Μία λύση είναι να προσδιορίσουμε την τιμή του N σε ένα «πέρασμα» και μετά να χρησιμοποιήσουμε άλλη μέθοδο κατά τη διάρκεια του δεύτερου «περάσματος». Ωστόσο, προκαθορίζοντας το N μπορεί να μην είναι πάντα πρακτικό ή δυνατόν [23] [24-31].

Για αυτόν τον λόγο, θα περιοριστούμε στην επεξεργασία του αρχείου εγγραφών με ένα διαδοχικό πέρασμα και χωρίς γνώση του N.

Αποδεικνύεται ότι όλες οι δειγματοληπτικές μέθοδοι που επεξεργάζονται το αρχείο με ένα «πέρασμα» μπορούν να χαρακτηριστούν ως αλγόριθμοι αποθέματος(reservoir algorithms).

Η βασική ιδέα πίσω από κάθε αλγόριθμο αποθέματος είναι να διαλέξει ένα δείγμα μεγέθους ≥ n, από το οποίο ένα τυχαίο δείγμα μεγέθους n μπορεί να παραχθεί. Ένας αλγόριθμος αποθέματος ορίζεται ως εξής:

Ορισμός 1. Το πρώτο βήμα κάθε αλγορίθμου αποθέματος είναι να τοποθετήσει τις πρώτες n εγγραφές του αρχείου σε μια δεξαμενή δειγματοληψίας (reservoir). Οι υπόλοιπες εγγραφές επεξεργάζονται σειριακά και επιλέγονται από τη δεξαμενή δειγματοληψίας μόνο όταν φτάσει η σειρά τους για επεξεργασία. Ένας αλγόριθμος καλείται αλγόριθμος αποθέματος αν καταφέρνει να επιτύχει μετά από την επεξεργασία της κάθε εγγραφής, ότι ένα πραγματικά τυχαίο δείγμα μεγέθους n μπορεί να εξαχθεί από την τρέχουσα κατάσταση της δεξαμενής δειγματοληψίας.

Στο τέλος της σειριακής σάρωσης του αρχείου, πρέπει να εξαχθεί το τελικό τυχαίο δείγμα από την δεξαμενή δειγματοληψίας. Η δεξαμενή δειγματοληψίας ενδέχεται να είναι αρκετά μεγάλο και έτσι η διαδικασία αυτή να είναι «ακριβή». Οι πιο αποδοτικοί αλγόριθμοι αποθέματος αποφεύγουν αυτό το βήμα με το να διατηρούν πάντα ένα σύνολο από n υποψήφιες εγγραφές που έχουν αναδειχθεί στην δεξαμενή δειγματοληψίας, οι οποίες αποτελούν ένα πραγματικά τυχαίο δείγμα των εγγραφών που έχουν επεξεργαστεί μέχρι εκείνη τη στιγμή. Όταν μια εγγραφή επιλεχθεί για την δεξαμενή δειγματοληψίας, γίνεται υποψήφια και αντικαθιστά μία από τις πρώην υποψήφιες εγγραφές. Στο τέλος της σειριακής σάρωσης του αρχείου, το τρέχον σύνολο των n υποψηφίων είναι το παραγόμενο αποτέλεσμα ως το τελικό δείγμα.

Ο Αλγόριθμος R(που είναι αλγόριθμος αποθέματος λόγω του Alan Waterman) λειτουργεί ως εξής: Όταν, για t ≥ n, η (t+1)στή εγγραφή του αρχείου είναι υπό επεξεργασία, τότε οι n υποψήφιες αποτελούν ένα τυχαίο δείγμα των πρώτων t εγγραφών. Η (t+1)στή εγγραφή έχει πιθανότητα n/(t+1) να βρίσκεται σε ένα τυχαίο δείγμα μεγέθους n των πρώτων t+1 εγγραφών και κατ' αυτόν τον τρόπο γίνεται υποψήφια με πιθανότητα n/(t+1). Η υποψήφια εγγραφή που αντικαθιστά επιλέγεται τυχαία από τις n υποψήφιες. Είναι αρκετά εύκολο να διαπιστώσει κανείς ότι το τελικό σύνολο n υποψηφίων αποτελεί ένα τυχαίο δείγμα των πρώτων t+1 εγγραφών.

Ο πλήρης αλγόριθμος δίνεται παρακάτω. Το τρέχον σύνολο των n υποψηφίων αποθηκεύεται στον πίνακα C, στις θέσεις C[0], C[1],…, C[n-1]. Η ενσωματωμένη Boolean συνάρτηση eof επιστρέφει true αν φτάσουμε στο τέλος του αρχείου. Η γεννήτρια τυχαίων αριθμών RANDOM επιστρέφει έναν πραγματικό αριθμό στη μονάδα του χρόνου. Η κλήση της συνάρτησης READ_NEXT_RECORD(R) διαβάζει την επόμενη εγγραφή από το αρχείο και την αποθηκεύει στην εγγραφή R. Η κλήση συνάρτησης SKIP_RECORDS(k) παραλείπει τις επόμενες k εγγραφές του αρχείου.

```
{Make the first n records candidates for the sample}
for j := 0 to n − 1 do READ_NEXT_RECORD(C[j]);
t := n;                                   {t is the number of records processed so far}
while not eof do                                    {Process the rest of the records}
  begin
  t := t + 1;
  M := TRUNC (t × RANDOM( ));          {M is random in the range 0 ≤ M ≤ t − 1}
  if M < n then          {Make the next record a candidate, replacing one at random}
    READ_NEXT_RECORD(C[M])
  else                                                  {Skip over the next record}
    SKIP_RECORDS(1)
  end;
```

Όταν φτάσει στο τέλος του αρχείου, οι n υποψήφιες εγγραφές που είναι αποθηκευμένες στον πίνακα C αποτελούν ένα πραγματικά τυχαίο δείγμα των N εγγραφών του αρχείου.

Οι περιορισμοί στους αλγορίθμους για αυτό το πρόβλημα δειγματοληψίας είναι ότι οι εγγραφές πρέπει να διαβαστούν σειριακά και το πολύ μια φορά. Αυτό πρακτικά σημαίνει ότι κάθε αλγόριθμος για αυτό το πρόβλημα πρέπει να διατηρεί μια δεξαμενή δειγματοληψίας η οποία περιέχει ένα τυχαίο δείγμα μεγέθους n των εγγραφών που έχουν επεξεργαστεί μέχρι τη δεδομένη χρονική στιγμή. Αυτό μας παρέχει την ακόλουθη γενίκευση.

Θεώρημα 1. Κάθε αλγόριθμος για αυτό το πρόβλημα δειγματοληψίας είναι ένας τύπος αλγόριθμου αποθέματος.

Ορισμός 2. Η τυχαία μεταβλητή $\mathscr{S}$(n, t) ορίζεται ως ο αριθμός των εγγραφών στο αρχείο που παραλείπονται πριν η επόμενη εγγραφή επιλεχθεί για την δεξαμενή δειγματοληψίας, όπου n είναι το μέγεθος του δείγματος και t είναι ο αριθμός των εγγραφών που έχουν επεξεργαστεί μέχρι τη δεδομένη στιγμή. Για σημειογραφική ευκολία, συχνά γράφουμε $\mathscr{S}$ αντί για $\mathscr{S}$ (n, t), όπου σε αυτή την περίπτωση οι παράμετροι n και t θα εννοούνται.

Η βασική ιδέα για τους αλγόριθμους αποθέματος είναι να παράγουν επαναλαμβανόμενα το $\mathscr{S}$, να παραλείπουν αυτόν τον αριθμό εγγραφών, και να επιλέγουν την επόμενη εγγραφή για την δεξαμενή δειγματοληψίας. Όπως στον Αλγόριθμο R, διατηρούμε ένα σύνολο n υποψηφίων στη δεξαμενή δειγματοληψίας. Αρχικά, οι πρώτες n εγγραφές επιλέγονται ως υποψήφιες και θέτουμε t := n. Οι αλγόριθμοι αποθέματος έχουν κατά βάση την παρακάτω γενική περιγραφή:

```
while not eof do                                   {Process the rest of the records}
  begin
  Generate an independent random variate 𝒮(n, t);
  SKIP_RECORDS(𝒮);                                 {Skip over the next 𝒮 records}
  if not eof then
    begin                  {Make the next record a candidate, replacing one at random}
    𝒦 := TRUNC (n × RANDOM( ));        {𝒦 is uniform in the range 0 ≤ 𝒦 ≤ n − 1}
    READ_NEXT_RECORD(C[𝒦])
    end
  t := t + 𝒮 + 1;
  end;
```

Οι αλγόριθμοι δειγματοληψίας που παρουσιάζονται διαφέρουν μεταξύ τους στον τρόπο που παράγεται το $\mathscr{S}$. Ο Αλγόριθμος R μπορεί να μπει σε αυτό το πλαίσιο: παράγει το $\mathscr{S}$ σε O($\mathscr{S}$) χρόνο, χρησιμοποιώντας O($\mathscr{S}$) κλήσεις στην RANDOM. Οι τρεις αλγόριθμοι που παρουσιάζονται σε αυτή την εργασία (Αλγόριθμοι X, Y και Z) παράγουν το $\mathscr{S}$ ταχύτερα απ' ότι ο Αλγόριθμος R. Ο αλγόριθμος Z που θα αναλυθεί στη συνέχεια, παράγει το $\mathscr{S}$ σε σταθερό χρόνο, συνήθως, και κατά συνέπεια εκτελείται σε μέσο χρόνο O(n(1+log(N+n))).

Το φάσμα του $\mathscr{S}$ (n, t) είναι το σύνολο των μη αρνητικών ακεραίων. Η συνάρτηση κατανομής F(s) = Prob{$\mathscr{S} \leq$s}, για s $\geq$ 0 μπορεί να εκφραστεί ως εξής:

$$F(s) = 1 - \frac{t^{\overline{n}}}{(t+s+1)^{\overline{n}}} = 1 - \frac{(t+1-n)^{\overline{s+1}}}{(t+1)^{\overline{s+1}}}.$$

Εξίσωση 1

Με βάση το παραπάνω μπορούμε να παράγουμε μια ανεξάρτητη ομοιόμορφη τυχαία μεταβλητή $\mathscr{U}$ και στη συνέχεια να θέσουμε την $\mathscr{S}$ στην μικρότερη τιμή s ≥ 0 έτσι ώστε $\mathscr{U} \leq$ F(s). Αντικαθιστώντας στην Εξίσωση 1, η συνθήκη τερματισμού ισοδυναμεί με:

$$\frac{(t+1-n)^{\overline{s+1}}}{(t+1)^{\overline{s+1}}} \leq 1 - \mathscr{U}.$$

Εξίσωση 2

Επειδή η $\mathscr{U}$ είναι ανεξάρτητη και ομοιόμορφα κατανεμημένη, έτσι είναι και η ποσότητα 1- $\mathscr{U}$. Επομένως μπορούμε να αντικαταστήσουμε το 1- $\mathscr{U}$ σε αυτή την ανισότητα με μια ανεξάρτητη ομοιόμορφη τυχαία μεταβλητή, την οποία ονομάζουμε $\mathscr{V}$. Η νέα συνθήκη τερματισμού είναι η:

$$\frac{(t+1-n)^{\overline{s+1}}}{(t+1)^{\overline{s+1}}} \leq \mathscr{V}.$$

Εξίσωση 3

Ο Αλγόριθμος Χ βρίσκει την μικρότερη τιμή του s ≥ 0 ικανοποιώντας την Εξίσωση 3 με απλή διαδοχική αναζήτηση, ως ακολούθως:

```
V := RANDOM( );                                   {V is uniform on the unit interval}
Search sequentially for the minimum s ≥ 0 such that (t + 1 − n)^(s+1) / (t + 1)^(s+1) ≤ V;
S := s;
```

Οι χρόνοι εκτέλεσης και των δύο Αλγορίθμων R και Χ είναι O(N), αλλά ο Αλγόριθμος Χ είναι πολλαπλά ταχύτερος, διότι καλεί την RANDOM μόνο μια φορά κάθε φορά που παράγεται το $\mathscr{S}$, αντί για O($\mathscr{S}$) φορές.

Ένας εναλλακτικός τρόπος να βρούμε το ελάχιστο s που να ικανοποιεί την Εξίσωση 3 είναι να χρησιμοποιήσουμε δυαδική αναζήτηση, ή ακόμα καλύτερα, μια παραλλαγή της μεθόδου παρεμβολής του Newton (Newton's interpolation method). Η τελευταία προσέγγιση είναι η βάση για τον Αλγόριθμο Υ.

Ο Αλγόριθμος Z ακολουθεί την γενική μορφή που περιγράψαμε παραπάνω. Η μεταβλητή τυχαίας παράληψης $\mathscr{S}$(n, t) παράγεται σε συνεχή χρόνο, κατά μέσο όρο, από μια τροποποίηση της μεθόδου απόρριψης-αποδοχής του von Neumann. (rejection-acceptance method). Η κύρια ιδέα είναι ότι μπορούμε να παράγουμε το γρήγορα, παράγοντας μια γρήγορη προσέγγιση και «διορθώνοντάς» την έτσι ώστε η καταληκτική της κατανομή να είναι η επιθυμητή κατανομή F(s).

Η γενική περιγραφή του πώς ο Αλγόριθμος Z παράγει το $\mathscr{S}$(n, t) παρουσιάζεται παρακάτω:

```
if t ≤ T × n then Use the inner loop of Algorithm X to generate 𝒮
else begin
  repeat
    Generate an independent random variable 𝒳 with density function g(x);
    𝒰 := RANDOM( );                          {𝒰 is uniform on the unit interval}
    if 𝒰 ≤ h(⌊𝒳⌋)/cg(𝒳) then break loop
  until 𝒰 ≤ f(⌊𝒳⌋)/cg(𝒳);
  𝒮 := ⌊𝒳⌋
  end;
```

Το περίπλοκο κομμάτι στην ανάπτυξη του Αλγορίθμου Z ήταν να βρεθούν κατάλληλες επιλογές για τις παραμέτρους g(x)(η συνάρτηση πυκνότητας του $\mathscr{X}$), c και h(s) που να δίνουν γρήγορες τιμές εκτέλεσης. Οι παρακάτω επιλογές φαίνεται να δουλεύουν καλύτερα:

$$g(x) = \frac{n}{t+x}\left(\frac{t}{t+x}\right)^n, \qquad x \geq 0;$$

$$c = \frac{t+1}{t-n+1};$$

$$h(s) = \frac{n}{t+1}\left(\frac{t-n+1}{t+s-n+1}\right)^{n+1}, \qquad x \geq 0.$$

Βελτιστοποιήσεις Αλγορίθμου Z: Παρακάτω παρουσιάζονται τρεις βελτιστοποιήσεις που βελτιώνουν δραματικά το χρόνο εκτέλεσης της απλής έκδοσης του Αλγορίθμου Z. Αυτές οι τροποποιήσεις ενσωματώνονται στην τελική υλοποίηση του Αλγορίθμου Z.

1. *Βελτιστοποίηση Κατωφλιού (Threshold Optimization).*
   Έχει ήδη συμπεριληφθεί ως μέρος του βασικού αλγορίθμου. Χρησιμοποιήσαμε μια σταθερή παράμετρο T για να προσδιορίσουμε πως θα παραχθεί το $\mathscr{S}$: αν t ≤ Tn, τότε χρησιμοποιείται η εσωτερική επανάληψη του Αλγορίθμου X για να παραχθεί το $\mathscr{S}$, αλλιώς χρησιμοποιείται η τεχνική της απόρριψης – αποδοχής. Η τιμή του T συνήθως ορίζεται στο εύρος 10 – 40. Η επιλογή του T ≈ 22 λειτούργησε καλύτερα από όλες στην εφαρμογή και ο τρέχων χρόνος

περιορίστηκε από έναν παράγοντα περίπου 4. Υπάρχει ένας πολύ βασικός λόγος που εξηγεί γιατί αυτή η βελτιστοποίηση είναι σημαντική(γι' αυτό εξάλλου χρησιμοποιήθηκε και ως μέρος του βασικού αλγορίθμου): Προστατεύει από υπερχείλιση υποδιαστολής κινητού μήκους! (floating-point overflow)

Οι βελτιστοποιήσεις που παρουσιάζονται στα σημεία 2 και 3 αφορούν τεχνικές για την απαλοιφή δαπανηρών κλήσεων σε μαθηματικές υπορουτίνες.

2. *Βελτιστοποίηση της RANDOM (RANDOM Optimization).*
   Η βελτιστοποίηση αυτή μας επιτρέπει μειώσουμε τις κλήσεις στην RANDOM περισσότερο από το ένα τρίτο. Κάθε φορά που παράγεται η $\mathscr{X}$ απαιτείται να παραχθεί και μια ανεξάρτητη ομοιόμορφη τυχαία μεταβλητή όπου εδώ την καλούμε $\mathscr{V}$. Μόνο που την πρώτη φορά που παράγεται η $\mathscr{X}$, μπορεί να αποφευχθεί μια κλήση στην RANDOM και να υπολογιστεί η $\mathscr{V}$(και κατά συνέπεια η $\mathscr{X}$) με έναν ανεξάρτητο τρόπο χρησιμοποιώντας τις τιμές των $\mathscr{U}$ και $\mathscr{X}$ από την προηγούμενη επανάληψη.

3. *Βελτιστοποίηση Υπορουτινών (Subroutine Optimization).*
   Μπορούμε να αυξήσουμε την ταχύτητα του Αλγορίθμου Z σχεδόν στο διπλάσιο, μειώνοντας στο μισό τον αριθμό των πράξεων της μορφής x^y, όπου x και y είναι είτε πραγματικοί αριθμοί (με υποδιαστολή κινητού μήκους) είτε μεγάλοι ακέραιοι. Ο υπολογισμός του x^y = exp(ylnx) περιλαμβάνει μια έμμεση κλήση στις δύο υπορουτίνες της μαθηματικής βιβλιοθήκης: EXP (exponential – εκθετική) kai LOG (logarithm – λογαριθμική). Ο υπολογισμός παίρνει σταθερό χρόνο, αλλά η σταθερά είναι πολύ μεγαλύτερη από τον χρόνο για ένα πολλαπλασιασμό ή διαίρεση. Μετά από πράξεις, θέτοντας $\mathscr{W}$ τον ένα όρο (έχει την ίδια διασπορά με την $\mathscr{V}^{-1/n}$) παρατηρούμε ότι μπορούμε να υπολογίσουμε την $\mathscr{X}$ χωρίς πράξη της μορφής x^y θέτοντας $\mathscr{X}$:=t($\mathscr{W}$ – 1).

<u>Υλοποίηση του Αλγορίθμου Z</u>

Αρχικά εκτελούμε τον Αλγόριθμο Z για να βρούμε ένα τυχαίο δείγμα μεγέθους n_0. Στη συνέχεια παράγουμε ένα τυχαίο δείγμα μεγέθους n από το τυχαίο δείγμα μεγέθους n_0 χρησιμοποιώντας την ακόλουθη απλή διαδικασία:

```
for j := 0 to n0 − 1 do already_selected := false;
num_selected = 0;
while num_selected < n do
  begin
  M := TRUNC(n0 × RANDOM( ));
  if not already_selected[M] then
        begin
        OUTPUT(C[M]);
        already_selected[M] := true;
        num_selected := num_selected + 1
        end
  end
```

Ο βελτιστοποιημένος κώδικας για τον Αλγόριθμο Z παρουσιάζεται παρακάτω. Το πρώτο κομμάτι του προγράμματος είναι ουσιαστικά ο κώδικας για τον Αλγόριθμο X, ο οποίος εκτελεί την δειγματοληψία μέχρι το t να είναι αρκετά μεγάλο [23].

```
{Make the first n records candidates for the sample}
for j := 0 to n − 1 do READ_NEXT_RECORD(C[j]);
t := n;                                   {t is the number of records processed so far}
{Process records using the method of Algorithm X until t is large enough}
thresh := T × n;
num := 0;                                 {num is equal to t − n}
while not eof and (t ≤ thresh) do
    begin
    V := RANDOM( );                       {Generate V}
    S := 0;
    t := t + 1; num := num + 1;
    quot := num/t;
    while quot > V do                     {Find min S satisfying (4.1)}
        begin
        S := S + 1;
        t := t + 1; num := num + 1;
        quot := (quot × num)/t
        end;
    SKIP_RECORDS(S);                      {Skip over the next S records}
    if not eof then
        begin          {Make the next record a candidate, replacing one at random}
        M := TRUNC(n × RANDOM( ));  {M is uniform in the range 0 ≤ M ≤ n − 1}
        READ_NEXT_RECORD(C[M])
        end
    end;

{Process the rest of the records using the rejection technique}
W := EXP(− LOG(RANDOM( ))/n);             {Generate W}
term := t − n + 1;                        {term is always equal to t − n + 1}
while not eof do
    begin
    loop
        {Generate U and X}
        U := RANDOM( );
        X := t × (W − 1.0);
        S := TRUNC(X);                    {S is tentatively set to ⌊X⌋}
        {Test if U ≤ h(S)/cg(X) in the manner of (6.3)}
        lhs := EXP(LOG(((U × (((t + 1)/term) ↑ 2)) × (term + S))/(t + X))/n);
        rhs := (((t + X)/(term + S)) × term)/t;
        if lhs ≤ rhs then
            begin W := rhs/lhs; break loop end;
        {Test if U ≤ f(S)/cg(X)}
        y := (((U × (t + 1))/term) × (t + S + 1))/(t + X);
        if n < S then begin denom := t; numer_lim := term + S end
        else begin denom := t − n + S; numer_lim := t + 1 end;
        for numer := t + S downto numer_lim do
            begin y := (y × numer)/denom; denom := denom − 1 end;
        W := EXP(− LOG(RANDOM( ))/n);     {Generate W in advance}
        if EXP(LOG(y)/n) ≤ (t + X)/t then break loop
    end loop;
    SKIP_RECORDS(S);                      {Skip over the next S records}

if not eof then
    begin          {Make the next record a candidate, replacing one at random}
    M := TRUNC (n × RANDOM( ));  {M is uniform in the range 0 ≤ M ≤ n − 1}
    READ_NEXT_RECORD(C[M])
    end;
t := t + S + 1;
term := term + S + 1
end;
```

Πίνακας 1:Χρόνοι για Αλγορίθμους R, X, Y, Z (σε δευτερόλεπτα)

Table II. Timings for Algorithms R, X, and Z (in seconds)

	Algorithm R	Algorithm X	Algorithm Z
$N = 10^6$			
$n = 10^1$	171	41	0.1
$n = 10^2$	153	35	0.8
$n = 10^3$	155	37	5
$n = 10^4$	158	47	31
$n = 10^5$	176	95	96
$N = 10^7$			
$n = 10^1$	1595	430	0.1
$n = 10^2$	1590	356	1
$n = 10^3$	1553	371	8
$n = 10^4$	1560	387	55
$n = 10^5$	1604	500	303

2.3. Απόσταση Hamming (Hamming Distance)

Στη θεωρία της πληροφορίας, η απόσταση Hamming μεταξύ δύο συμβολοσειρών ίσου μήκους είναι ο αριθμός των θέσεων στις οποίες τα αντίστοιχα σύμβολα είναι διαφορετικά. Με άλλα λόγια, μετρά τον ελάχιστο αριθμό των αντικαταστάσεων που απαιτούνται για να αλλάξει μία συμβολοσειρά στην άλλη, ή τον ελάχιστο αριθμό των σφαλμάτων που θα μπορούσαν να έχουν μετατρέψει την μία συμβολοσειρά στην άλλη.

Η απόσταση Hamming χρησιμοποιείται για να καθορίσει ορισμένες βασικές έννοιες στη θεωρία κωδικοποίησης, όπως τον εντοπισμό σφαλμάτων και κώδικες διόρθωσης λαθών. Ειδικότερα, ένας κώδικας C λέγεται ότι είναι ανίχνευσης k-λαθών εάν οποιεσδήποτε δύο κωδικές λέξεις C1 και C2 από το C που έχουν απόσταση Hamming μικρότερη από k συμπίπτουν. Διαφορετικά, ένας κώδικας είναι ανίχνευσης k-σφαλμάτων, αν και μόνο αν η ελάχιστη απόσταση Hamming μεταξύ δύο οποιωνδήποτε κωδικών λέξεων του είναι τουλάχιστον k + 1.

Ένας κώδικας C λέγεται ότι είναι διορθωτής k-σφάλματων αν για κάθε λέξη w του υποκείμενου χώρου Hamming H υπάρχει το πολύ μία κωδική λέξη c (από το C), έτσι ώστε η απόσταση Hamming μεταξύ w και γ να είναι μικρότερη από k. Με άλλα λόγια, ένας κώδικας είναι διορθωτής k-σφάλματων αν και μόνο αν η ελάχιστη απόσταση Hamming μεταξύ οποιωνδήποτε δύο από τις κωδικές λέξεις του είναι τουλάχιστον 2k + 1.

Έτσι, ένας κώδικας με ελάχιστη απόσταση Hamming d μεταξύ των κωδικών λέξεων του μπορεί να ανιχνεύσει το πολύ d-1 λάθη και μπορεί να διορθώσει ⌊ (d-1) / 2⌋ λάθη.

Ορισμός: Η απόσταση Hamming d(x,y) ανάμεσα σε δύο διανύσματα ή μονοδιάστατους πίνακες x,y είναι ο αριθμός των συντελεστών στους οποίους διαφέρουν [32] π.χ.

$F_2^{(5)}$ d(00111,11001) = 4

$F_3^{(4)}$ d(0122,1220) = 3.

3. Δομή του προγράμματος

Το πρόγραμμα της πτυχιακής εργασίας απαρτίζεται από δύο επιμέρους προγράμματα. Το πρώτο είναι το SensorSampler και το δεύτερο το Concentrator.

3.1. SensorSampler

Το SensorSampler αποτελείται ουσιαστικά από ένα μόνο java package, το sensor. Η κύρια κλάση του sensor είναι η SensorSampler. Σε αυτήν ορίζεται το port όπου και θα γίνει η σύνδεση με τον Concentrator καθώς και η περίοδος δειγματοληψίας σε milliseconds. Αρχικά λαμβάνει την τιμή της θερμοκρασίας σε βαθμούς Κελσίου από τον slider του Solarium, την οποία αποθηκεύει σε μορφή μεταβλητής double. Δημιουργεί και ανοίγει την σύνδεση για να σταλούν οι τιμές αυτές μέσω του radiogram. Επίσης δημιουργείται το datagram στο οποίο γράφεται η θερμοκρασία την δεδομένη χρονική στιγμή και αποστέλλεται στον Concentrator. Τέλος, ο αισθητήρας μπαίνει σε “sleep mode” για να εξοικονομήσει ενέργεια για όσο χρόνο έχουμε ορίσει ανάλογα με την συχνότητα δειγματοληψίας.

3.2. Concentrator

Το **Concentrator** αποτελείται από τέσσερα java packages. Πιο συγκεκριμένα:

3.2.1. Package radio

Κλάση Concentrator

Κάνει override την run η οποία εκτελεί την συνάρτηση concentrate. Σε αυτή την κλάση ορίζονται οι περισσότερες βασικές μεταβλητές. Αρχικά ορίζεται η θύρα επικοινωνίας (HOST_PORT = 67) των δύο προγραμμάτων SensorSampler kai Concentrator ώστε να ληφθούν οι μετρήσεις θερμοκρασίας και να επεξεργαστούν καταλλήλως. Ορίζονται επίσης πόσοι θα είναι οι αισθητήρες (MAX_SENSOR_SIZE = 6) που θα αποστέλλουν τιμές θερμοκρασίας στο σύστημα. Ακόμα ορίζεται το μέγεθος του datastream (MAX__DATASTREAM_SIZE = 1000 ή αλλιώς απλά ένας μεγάλος αριθμός), καθώς και η συχνότητα (RESERVOIR_CYCLE = 8)με την οποία η δειγματοληψία δεξαμενής αποστέλλει το δείγμα των υποψηφίων. Τέλος ορίζεται και ο αριθμός των θέσεων της δειγματοληψίας δεξαμενής (DEFAULT_RESERVOIR_SIZE = 5).

Η κλάση αυτή περιλαμβάνει επίσης τους ακόλουθους πίνακες, σημαντικούς στην υλοποίηση της εφαρμογής. Στον πίνακα event αποθηκεύονται τα events για κάθε ροή/αισθητήρα από την υλοποίηση που χρησιμοποιεί μόνο τον αλγόριθμο Shewhart. Ο πίνακας αυτός έχει μια θέση για κάθε ροή, που σημαίνει ότι οι μετρήσεις γίνονται overwrite και αυτή που θα επεξεργαστεί εν τέλει είναι η τελευταία που θα αποθηκευτεί. Στον πίνακα reservoirEvent αποθηκεύονται τα events της δεύτερης υλοποίησης με την δειγματοληψία δεξαμενής. Έχει επίσης μία θέση για κάθε ροή και αποθηκεύει είτε το μέσο είτε το μέγιστο όρο των υποψηφίων της δειγματοληψίας δεξαμενής. Οι πίνακες multiSensorEventsToCompare και multiSensorReservoirEventsToCompare έχουν

αποθηκευμένα για μια δεδομένη στιγμή τα συμβάντα (events) προς σύγκριση για την πρώτη και για την δεύτερη υλοποίηση αντίστοιχα. Η αρχικοποίηση των παραπάνω τεσσάρων πινάκων γίνεται στον Constractor του Concentrator.

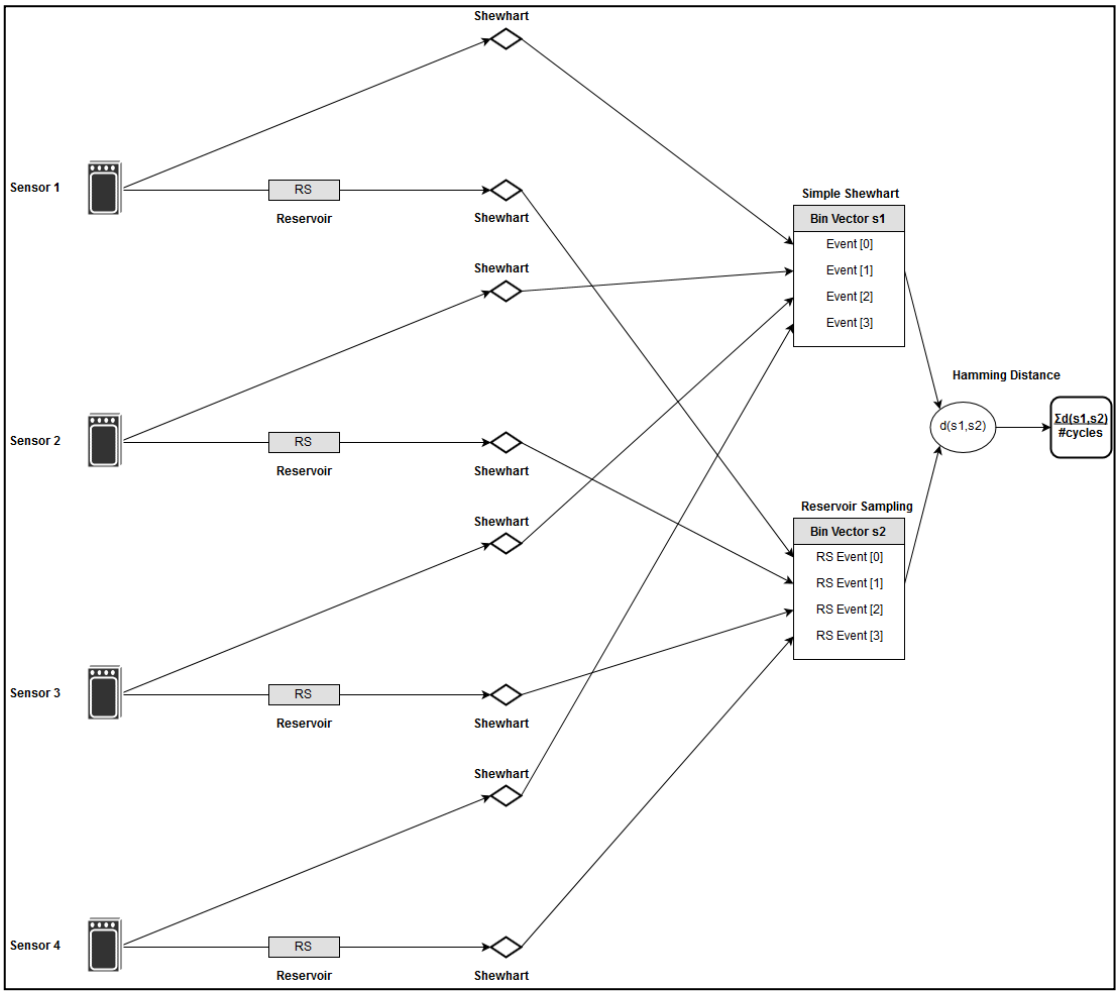
Συνάρτηση Run

Η συνάρτηση αυτή δημιουργεί ένα νήμα (thread), το οποίο καλεί την συνάρτηση concentrate.

Συνάρτηση private concentrate()

Αυτή η συνάρτηση είναι υπεύθυνη για τη βασική λειτουργία του Concentrator, δηλαδή τη συλλογή των απεσταλμένων τιμών θερμοκρασίας από τους αισθητήρες. Ανοίγουμε το connection (rCon) ώστε να λάβει τις τιμές μέσω του Radiogram (dg).

Φτιάχνουμε δύο ShewartChDet αντικείμενα, ένα για κάθε υλοποίηση, δηλαδή ένα αντικείμενο για τα αποτελέσματα της δειγματοληψίας δεξαμενής και ένα για την απλή υλοποίηση μόνο με τον αλγόριθμο Shewhart. Στη συνέχεια, αρχίζουμε να λαμβάνουμε δεδομένα, κάνουμε parse το αναγνωριστικό (id) του κάθε πομπού και διαπιστώνουμε αν έχει υπάρξει συμβάν (event) μέσω του αλγορίθμου Shewhart. Έπειτα, αποθηκεύουμε τις τιμές στους πίνακες event και reservoirData. Είναι σημαντικό να τονίσουμε, ότι ο πίνακας event έχει μία θέση η οποία αντικαθιστάται σε κάθε κύκλο, ενώ ο πίνακας reservoirData, είναι πίνακας πολλαπλών θέσεων και αποθηκεύει πρακτικά όλες τις ληφθείσες τιμές θερμοκρασίας που αποστέλλουν οι αισθητήρες. Αφού αποθηκεύσουμε τις τιμές στους δύο διακριτούς αυτούς πίνακες, ελέγχουμε αν έχει ολοκληρωθεί ένας κύκλος ώστε να αρχίσει το reservoir sampling. Σημειώνεται ότι ως ένας κύκλος ορίζεται οι οχτώ επαναλήψεις του while loop, δηλαδή η λήψη οχτώ τιμών θερμοκρασίας από τους αισθητήρες. Αν έχει περάσει ένας κύκλος δημιουργούμε έναν πίνακα υποψηφίων από τα ως τώρα συγκεντρωμένα δεδομένα στον πίνακα, χρησιμοποιώντας τον αλγόριθμο Αποθέματος. Ύστερα, παίρνουμε τη μέση ή τη μέγιστη τιμή του δείγματος των υποψηφίων (ανάλογα την τεχνική υλοποίησης που θέλουμε να χρησιμοποιήσουμε) και τη δίνουμε ως είσοδο στον αλγόριθμο Shewhart, ο οποίος μας επιστρέφει τον εντοπισμό ή μη, ενός συμβάντος. Αυτή την τιμή και την τιμή που έχει αποθηκεύσει ο πίνακας event, τις αποθηκεύουμε στους δύο ξεχωριστούς πίνακες (multiSensorEventsToCompare και multiSensorReservoirEventsToCompare αντίστοιχα), στις ίδιες θέσεις. Αυτοί είναι και οι τελικοί πίνακες που θα συγκριθούν για την εξαγωγή των συμπερασμάτων μας. Τέλος, όταν γεμίσουν αυτοί οι δύο πίνακες, συγκρίνονται μέσω της απόστασης Hamming και εκτυπώνεται η διαφορά τους. Η διαδικασία επαναλαμβάνεται εξ αρχής. Στο Σχήμα 2 παρουσιάζεται μια απεικόνιση της υλοποίσης.



Σχήμα 2: Σχηματική απεικόνιση υλοποίησης

3.2.2. Package reservoir

Κλάση Reservoir

Συνάρτηση public Double[] getSample()

Η συνάρτηση αυτή περιλαμβάνεται στην παραπάνω κλάση. Παίρνει το position της δειγματοληψίας δεξαμενής, παραλείπει τον αριθμό των τιμών που ορίζει το αποτέλεσμα της συνάρτησης getItemsToSkip και συνεχίζει με τον ίδιο τρόπο.

Κλάση Z

Η κλάση Z περιέχει την συνάρτηση public int getItemsToSkip, η οποία είναι υπεύθυνη για την επιλογή του αριθμό των στοιχείων που θα παραληφθούν σε κάθε βήμα.

3.2.3. Package shewhart

Κλάση ShewAlg

Κυρίως υλοποίηση αλγορίθμου Shewhart όπως περιγράφηκε στην προηγούμενη ενότητα.

Κλάση ShewartChDet

Για να λειτουργήσει ορθά ο αλγόριθμος Shewhart, πρέπει να δημιουργήσουμε ένα object τύπου ShewAlg, καθώς και να γνωρίζουμε στον εκάστοτε υπολογισμό την κατάσταση (state) του κάθε τέτοιου τύπου object, το οποίο αντιστοιχεί σε έναν αισθητήρα. Για αυτό το λόγο πριν αρχίσει η συλλογή και επεξεργασία των απεσταλμένων πληροφοριών δημιουργούμε τόσα objects, όσοι και οι αισθητήρες.

Κλάση ShewState

Αρχικοποίηση της μέσης τιμής (mean) σε 0.0, διακύμανσης (variance) σε 0.0 και k σε 3.

3.2.4. Package utils

Κλάση SuperMath

Λόγω του ότι η έκδοση SDK που χρησιμοποιήθηκε (1.4) δεν περιελάμβανε την Java.lang.Math.pow() υπήρξε η ανάγκη δημιουργίας της παρούσας κλάσης, δηλαδή της δύναμης ενός αριθμού double υψωμένος σε έναν άλλο αριθμό double.

Κλάση SuperUtils

Η συνάρτηση appendToArray προσθέτει αριθμούς int στον πίνακα.

Η συνάρτηση appendToArray προσθέτει στον αντίστοιχο πίνακα αριθμούς double.

Οι συναρτήσεις print χρησιμοποιήθηκαν για logging.

Η συνάρτηση isarrayfull ελέγχει αν ο εν λόγω πίνακας είναι γεμάτος.

Η συνάρτηση getAverage υπολογίζει τον μέσο όρο των τιμών της δειγματοληψίας δεξαμενής.

Η συνάρτηση getMax βρίσκει τον μέγιστο όρο των τιμών της δειγματοληψίας

Η συνάρτηση getHammingDistance υπολογίζει την απόσταση Hamming των δυο πινάκων που αντιστοιχούν στις δύο διαφορετικές υλοποιήσεις.

Κλάση SuperString

Σε αυτήν την κλάση γίνεται ο διαχωρισμός (split) του string ώστε να διαπιστωθεί από ποιόν αισθητήρα ήρθαν οι εκάστοτε μετρήσεις θερμοκρασίας.

4. ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Για να έχουμε αξιόπιστα αποτελέσματα, η εφαρμογή τροποποιήθηκε ελαφρώς ώστε να δεχτεί σαν είσοδο μεγάλο όγκο δεδομένων σε μορφή csv (comma – separated values) αντί για τιμές από το slider του Solarium. Πιο συγκεκριμένα το πρόγραμμα τροφοδοτήθηκε με περίπου 21.000 μετρήσεις ανά αισθητήρα/ροή (stream). Οι ροές αυτές επιλέχθηκε να είναι 6, αριθμός που παραμένει σταθερός καθ' όλη τη διάρκεια των εκτελέσεων του προγράμματος. Οι διαφορετικές εκτελέσεις έγιναν για τις διαφορετικές τιμές των παρακάτω μεταβλητών:

- Reservoir_Cycle: Δηλώνει ανά πόσες μετρήσεις/κύκλους γίνεται η δειγματοληψία. Δοκιμάστηκαν οι τιμές 5, 10 και 15.
- Reservoir_Size: Το μέγεθος της δειγματοληψίας δεξαμενής, εκεί όπου πρακτικά αποθηκεύονται οι υποψήφιοι από τους οποίους θα βγει το μέγιστο (Max) ή ο μέσος όρος (Average) για να συγκριθεί με το αποτέλεσμα της απλής υλοποίησης χωρίς δειγματοληψία δεξαμενής. Δοκιμάστηκαν οι τιμές 5, 10, 20,και 30.
- MAX: Η πρώτη εκ των δύο επιλογών για την εκλογή αντιπροσώπου της δειγματοληψίας δεξαμενής, που θα συγκριθεί με το αποτέλεσμα της απλής υλοποίησης χωρίς δειγματοληψία δεξαμενής. Επιλέγει την μεγαλύτερη τιμή του πίνακα.
- AVERAGE: Η δεύτερη εκ των δύο επιλογών για την εκλογή αντιπροσώπου της δειγματοληψίας δεξαμενής που θα συγκριθεί με το αποτέλεσμα της απλής υλοποίησης χωρίς δειγματοληψία δεξαμενής. Επιλέγει τον μέσο όρο των τιμών του πίνακα.

Σημείωση: Λόγω του ότι οι μετρήσεις στο σύνολο τους παρουσίαζαν μεγαλύτερες αποστάσεις Hamming, δηλαδή χειρότερα αποτελέσματα για την επιλογή του AVERAGE, οι εκτελέσεις του προγράμματος για το AVERAGE έγιναν μόνο για τις ακραίες τιμές των Reservoir_Cycle και Reservoir_Size.

Τα αποτελέσματα της εκτέλεσης παρουσιάζονται στον Πίνακα 2.

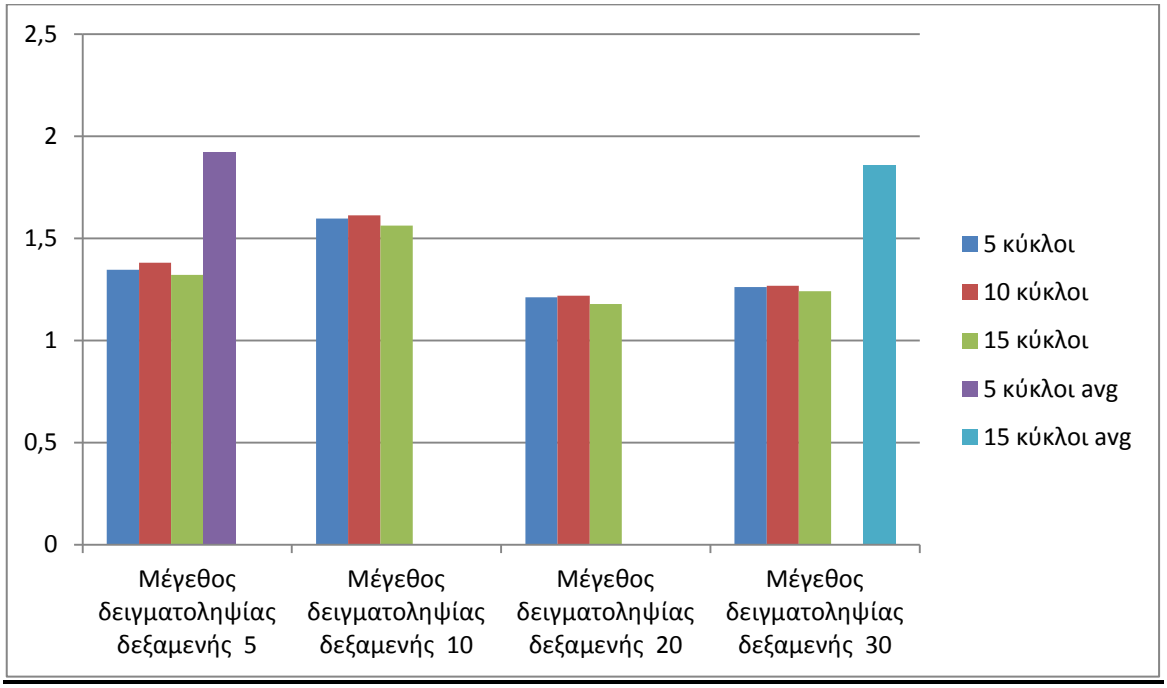
α : Reservoir size/Μέγεθος δειγματοληψίας δεξαμενής
β: Reservoir cycles/Κύκλοι δειγματοληψία

Πίνακας 2: Αποτελέσματα εκτέλεσης προγράμματος για διαφορετικούς συνδυασμούς τιμών μεγέθους δειγματοληψίας δεξαμενής, κύκλων δειγματοληψίας και επιλογής αντιπροσώπου δειγματοληψίας δεξαμενής

α / β	5	10	20	30
5	1,3457687849 1,9203288017	1,5964801348	1,2114026767	1,2619875645
10	1,3797174783	1,6118490406	1,2190596668	1,2681846932
15	1,3200506008	1,5613535736	1,1774193548	1,2413029728 1,8551549652

Καλύτερη τιμή Χειρότερη τιμή Τιμές Average

Το διάγραμμα στο Σχήμα 3 παρουσιάζει τα αποτελέσματα του Πίνακα 2 σχηματικά.



Σχήμα 3:Συσχετισμός αριθμού κύκλων και μεγέθους δειγματοληψίας

5. ΣΥΜΠΕΡΑΣΜΑΤΑ

Όσον αφορά στην εκλογή αντιπροσώπου της δειγματοληψίας δεξαμενής υπήρχαν δύο επιλογές: επιλογή της μέγιστης τιμής της δειγματοληψίας δεξαμενής (max) και επιλογή του μέσου όρου των τιμών της δειγματοληψίας δεξαμενής (average). Τα αποτελέσματα των εκτελέσεων του προγράμματος, δείχνουν καθαρά ότι η επιλογή της μέγιστης τιμής εμφανίζει σαφώς καλύτερα αποτελέσματα. Το εύρος των τιμών αυτών κυμαίνεται από περίπου 1,18 μέχρι 1,60 σε αντίθεση με την επιλογή του μέσου όρου όπου οι μετρήσεις είναι όλες άνω του 1,85, συγκλίνοντας στο 2. Γι' αυτό το λόγο οι μετρήσεις με την επιλογή του μέσου όρου έγιναν μόνον ενδεικτικά για τις ακραίες τιμές των μεταβλητών και όχι για όλο το εύρος των μετρήσεων. Εκτός των αποτελεσμάτων των παραπάνω μετρήσεων, υπάρχει και η πρακτική προσέγγιση του προβλήματος αυτού που ενισχύει την εγκυρότητα των αποτελεσμάτων. Για παράδειγμα, το εικονικό μας περιβάλλον αποτελείται από μια συστάδα αισθητήρων που αποστέλλουν δεδομένα, τα οποία είναι μετρήσεις θερμοκρασίας του συστήματος αυτών. Όταν παρατηρηθεί αύξηση της θερμοκρασίας, η οποία είναι δυνητικά επικίνδυνη προς τη λειτουργία του συστήματος, η μέθοδος δειγματοληψίας που εφαρμόζουμε, θα πρέπει να εντοπίσει μια τέτοια αυξημένη τιμή θερμοκρασίας ανάμεσα σε πολλές διαφορετικές τιμές. Εφαρμόζοντας, το μέσο όρο της δειγματοληψίας δεξαμενής, είναι πιθανό να μην εντοπιστεί αυτή η αύξηση της θερμοκρασίας με αποτέλεσμα την μη διαπίστωση του συμβάντος στο σύστημα με τις όποιες επιπτώσεις αυτό επιφέρει. Επιλέγοντας όμως τη μέγιστη τιμή της δειγματοληψίας δεξαμενής, η μέτρηση αυτή εντοπίζεται με αποτέλεσμα τον εντοπισμό της αύξησης θερμοκρασίας και άρα την αποφυγή ενός τέτοιου συμβάντος.

Από τα αποτελέσματα των μετρήσεων παρατηρούμε επίσης, ότι όσο αυξάνεται το μέγεθος της δειγματοληψίας δεξαμενής, η απόσταση Hamming συνεχώς ελαττώνεται συγκλίνοντας στο 1 και πιθανόν, αυξανομένων των τιμών, και στο 0. Αυτό συνάδει με τη λογική σκέψη, ότι όσο μεγαλύτερο είναι ένα δείγμα, τόσο πιο ακριβές είναι. Για παράδειγμα, αυξάνοντας το μέγεθος της δειγματοληψίας δεξαμενής στο 50, το τελικό αποτέλεσμα κυμαίνεται μεταξύ 1,05 και 1,06. Δεδομένου του αρκετά μεγάλου μεγέθους των πειραματικών δεδομένων (~21.000 μετρήσεις), το αποτέλεσμα αυτό είναι εντυπωσιακά ακριβές.

Τέλος, παρατηρώντας προσεχτικά τον Πίνακα 2 διαπιστώνουμε ότι παρά τη μεταβολή της συχνότητας (5, 10, 15) σε σχέση πάντα με το μέγεθος της δειγματοληψίας δεξαμενής, οι επιμέρους τιμές ελάχιστα διαφοροποιούνται, δεδομένο το οποίο μας οδηγεί στο συμπέρασμα ότι η συχνότητα της δειγματοληψίας ελάχιστα επηρεάζει τα αποτελέσματά μας.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος Όρος	Ελληνικός Όρος
reservoir algorithm	αλγόριθμος αποθέματος
reservoir	δειγματοληψία δεξαμενής
Hamming Distance	απόσταση Hamming
Wireless Sensor Network (WSN)	Ασύρματο Δίκτυο Αισθητήρων
event	συμβάν
Graphical User Interface (GUI)	γραφικό περιβάλλον
virtual machine	εικονικό μηχάνημα/μηχανή
compiling	μεταγλώττιση
free-range	ελευθέρου βεληνεκούς
basestation	σταθμός βάσης
virtual SPOT	εικονικό SPOT
sockets	υποδοχές
garbage collector	συλλέκτης απορριμμάτων
application isolation	απομόνωση εφαρμογών
thread	νήμα
debugging	εκσφαλμάτωση
client	πελάτης
server	εξυπηρετητής, διακομιστής
stream	ροή

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

WSN	Wireless Sensor Network
SunSPOT	Sun Small Programmable Object Technology
RAM	Random Access Memory
USB	Universal Serial Bus
GUI	Graphical User Interface
MAC	Media Access Control
JVM	Java Virtual Machine
RGB	Red Green Blue
CLDC	Connected Limited Device Configuration
HTTP	Hypertext Transfer Protocol
ISW	Interrupt Status Word
JDWP	Java Debug Wire Protocol
JPDA	Java Platform Debugger Architecture
GCF	Generic Connection Framework
MIDP	Mobile Information Device Profile
JAD	Java Application Descriptor
OTA	Over The Air
XML	Extensible Markup Language
IDE	Integrated Development Environment
ANT	Another Neat Tool
IEEE	Institute of Electrical and Electronics Engineers
SDK	Software Development Kit
MEMS	Micro-Electro-Mechanical System

UCL	Upper Control Limit
LCL	Lower Control Limit
CSV	Comma – Separated Values

ΑΝΑΦΟΡΕΣ

[1] Πατρώνη Μαρίκα, «Ανάπτυξη συστήματος έξυπνου φωτισμού με χρήση αισθητήρων Sun SPOT», Πτυχιακή εργασία, Εθνικό και Καποδιστριακό Πανεπιστήμιο, Σχολή Θετικών Επιστημών,Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Ιούλιος 2010
[2] SunSPOT-Programmers-Manual
[3] Καλυβίνου Λουκιανή, «Συμπίεση Δεδομένων Πλαισίου: Χρησιμοποίηση της Ανάλυσης Κύριων Συνιστωσών για Αποτελεσματικές Ασύρματες Επικοινωνίες», Πτυχιακή εργασία, Εθνικό και Καποδιστριακό Πανεπιστήμιο, Σχολή Θετικών Επιστημών,Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Μάρτιος 2013
[4] Introduction To SPOT, http://www.slideshare.net/pauldeng/introduction-to-spot
[5] Animesh Pathak, Qunzhi Zhou, Luca Mottola, Amol Bakshi, Viktor K. Prasanna, and Gian Pietro Picco, "End-to-end Toolkit for Developing a Class of WSN Applications on Sun SPOT Nodes", Διαθέσιμο: https://who.rocq.inria.fr/Animesh.Pathak/papers/pathakdcoss07poster.pdf
[6] Sun Microsystems, *Sun SPOT Developer's Guide: Red Release 5.0*, Sun Microsystems, California, 2009
[7] The Squawk Project, http://labs.oracle.com/projects/squawk/
[8] Doug Simon, Cristina Cifuentes, Dave Cleal, John Daniels and Derek White, *Java™ on the Bare Metal of Wireless Sensor Devices: The Squawk Java Virtual Machine*, Proceedings of the 2nd international conference on Virtual Execution Environments, June 14-16, 2006, Ottawa, Ontario, Canada
[9] Doug Simon and Cristina Cifuentes, *The Squawk Virtual Machine: Java™ on the Bare Metal*, Extended abstract, OOPSLA, October 16-20, 2005, San Diego, California, USA
[10] Angela Caicedo, *The Sun SPOT Technology: Java™ Technology – Based Wireless Sensor Networks*, presented at Austin Java Users Group – Meeting, 05 December, 2006
[11] Lee Chuk Munn, *Fun with Sun SPOTs*, presented at Sun Tech Days 2007-2008, Hyderabad, India, 2008
[12] Wikipedia, MIDlet, http://en.wikipedia.org/wiki/MIDlet
[13] Learning Path: MIDlet Life Cycle, http://developers.sun.com/mobility/learn/midp/lifecycle
[14] Wikipedia, Apache Ant, http://en.wikipedia.org/wiki/Apache_Ant
[15] Apache Ant Organisation, http://ant.apache.org/
[16] Sun Laboratories, *Solarium User's Guide: Red Release 5.0*, Sun Microsystems, California, July 2009
[17] Sun Laboratories, *Sun SPOT Owner's Manual: Red Release 5.0*, Sun Microsystems, California, June 2009
[18] Sun Laboratories, *Sun SPOT Theory of Operation: Red Release 5.0*, Sun Microsystems, California, June 2009
[19] NetBeans Organisation, http://netbeans.org/
[20] Wikipedia, NetBeans, https://en.wikipedia.org/wiki/NetBeans
[21] http://www.itl.nist.gov/div898/handbook/mpc/section2/mpc221.htm
[22] Shewhart (more from algorithms.pdf)
[23] Random Sampling with a Reservoir Jeffrey Scott Vitter Brown University
[24] BENTLEY, J. L. Personal communication, Apr. 1983; see [ll].
[25] ERNVALL, J. AND NEVALAINEN, 0. An algorithm for unbiased random sampling. Cornput. J. 25, 1 (Jan. 1982), 45-47.
[26] FAN, C. T., MULLER, M. E., AND REZUCHA, I. Development of sampling plans by using sequential (item by item) selection techniques and digital computers. Am. Stat. Assoc. J. 57 (June 1962), 387-402.
[27] JONES, T. G. A note on sampling a tape file. Commun. ACM 5,6 (June 1962), 343.
[28] KAWARASAKI, J., AND BBUYA, M. Random numbers for simple random sampling without replacement. Keio Math. Sem. Rep. No. 7 (1982), l-9.
[29] KNUTH, D. E. The Art of Computer Programming. vol. 2: Seminumerical Algorithms, 2nd ed. Addison-Wesley, Reading, Mass., 1981.
[30] VITTER, J. S. Faster methods for random sampling. Commun. ACM 27, 7 (July 1984). 703-718.
[31] VI'ITER, J. S. Optimum algorithms for two random samplingproblems. In Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science (Tucson, AZ., Nov. 7-9), IEEE, New York, 1983 pp. 65-75
[32] http://www.maths.manchester.ac.uk/~pas/code/notes/part2.pdf