Η ΠΛΑΤΦΟΡΜΑ SunSPOT

Η συσκευή SunSPOT είναι μια μικρή, ασύρματη, πειραματική πλατφόρμα με μπαταρίες. Είναι προγραμματισμένη σχεδόν εξ ολοκλήρου σε Java για να είναι δυνατό σε προγραμματιστές να δημιουργήσουν προγράμματα που απαιτούσαν εξειδικευμένες δεξιότητες προγραμματισμού για ενσωματωμένα συστήματα. Η πλατφόρμα υλικού περιλαμβάνει μια σειρά από ενσωματωμένους αισθητήρες καθώς και 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. Επίσης όλοι οι drivers των συσκευών είναι γραμμένοι σε Java.

Κάθε SunSPOT kit περιέχει δύο πλήρεις, ελευθέρου βεληνεκούς (free-range) SunSpots (με επεξεργαστή, ραδιοσυχνότητα, πίνακα αισθητήρων και μπαταρία) και έναν SunSPOT σταθμό βάσης (base station) (με επεξεργαστή και ραδιοσυχνότητα), ένα USB καλώδιο για την σύνδεση των SPOTs/basestation στον υπολογιστή και ένα cd με όλα τα εργαλεία ανάπτυξης λογισμικού που απαιτούνται για να ξεκινήσει κανείς την ανάπτυξη εφαρμογών για SunSPOT.

Εικόνα 1: Ένα πλήρες SunSPOT kit

Ο σταθμός βάσης (base station) συνδέεται με τον υπολογιστή (PC) και επιτρέπει να γράφουμε προγράμματα που μπορούν να τρέξουν στον υπολογιστή μας και να χρησιμοποιήσουμε τη ραδιοσυχνότητα του σταθμού βάσης (base station) για να επικοινωνήσουμε με απομακρυσμένα SunSPOTs.

Τα εργαλεία ανάπτυξης επίσης μπορούν να κάνουν χρήση του σταθμού βάσης για την ανάπτυξη και τον εντοπισμό σφαλμάτων σε εφαρμογές απομακρυσμένων SunSPOT. Σημειώστε ότι ένα πλήρες SunSPOT μπορεί επίσης να χρησιμοποιηθεί ως ένας σταθμός βάσης, αν και με τον τρόπο αυτό ο πίνακας αισθητήρων του δεν θα μπορούσε να χρησιμοποιηθεί.

Πλατφόρμες ανάπτυξης που υποστηρίζονται αυτήν τη στιγμή

Το αρχικό λογισμικό της Sun SPOT έχει δοκιμαστεί σε Windows XP, Windows 7 (και 32-bit και 64-bit), Macintosh OS X 10.4 + τρέχει και σε PowerPC και σε

Intel-based υπολογιστές, Linux (έχει ελεγχθεί διεξοδικά σε Ubuntu 10.10 32 -bit και 64-bit εκδόσεις), και Solaris x86. Ωστόσο, οι πλατφόρμες ανάπτυξης συνεχώς ενημερώνονται και ανανεώνονται.

Εικόνα 2: Μία συσκευή SunSPOT

Εξομοιωτής ή προσομοιωτής για Sun SPOT

Η έκδοση 6.0 περιλαμβάνει έναν εξομοιωτή ως μέρος του SPOTWorld. Αυτός ο εξομοιωτής μπορεί να εκτελεί μια εφαρμογή SunSPOT στον υπολογιστή. Αυτό επιτρέπει τον έλεγχο ενός προγράμματος πριν από την ανάπτυξη σε ένα πραγματικό SunSPOT, ή αν ένα πραγματικό SunSPOT δεν είναι διαθέσιμο.

Αντί για ένα φυσικό πίνακα αισθητήρων (sensorboard), η SPOTWorld εμφανίζει 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). Η εφαρμογή μας μπορεί να ελέγξει το χρώμα των LEDs που εμφανίζεται στην εικόνα του εικονικού SPOT, όπως ακριβώς θα ήταν ένα πραγματικό SPOT.

Μπορούμε κάνοντας κλικ με το ποντίκι στους διακόπτες στο εικονικό SunSPOT να αλλάξουμε τιμές στους διακόπτες. Υποστηρίζεται επίσης λήψη και αποστολή μέσω ραδιοσυχνότητας. Κάθε εικονικό SPOT έχει εκχωρήσει τη δική του διεύθυνση και μπορεί να μεταδώσει σε ένα ή σε πολλά άλλα

εικονικά SunSPOT. Εάν ένας κοινός σταθμός βάσης είναι διαθέσιμος ένα εικονικό SunSPOT μπορεί επίσης να αλληλεπιδράσει μέσω ραδιοσυχνότητας με πραγματικά SunSPOTs.

Εικόνα 3: Συσκευή SunSPOT σε λειτουργία

4.1 Υλικό (Hardware) του SunSPOT

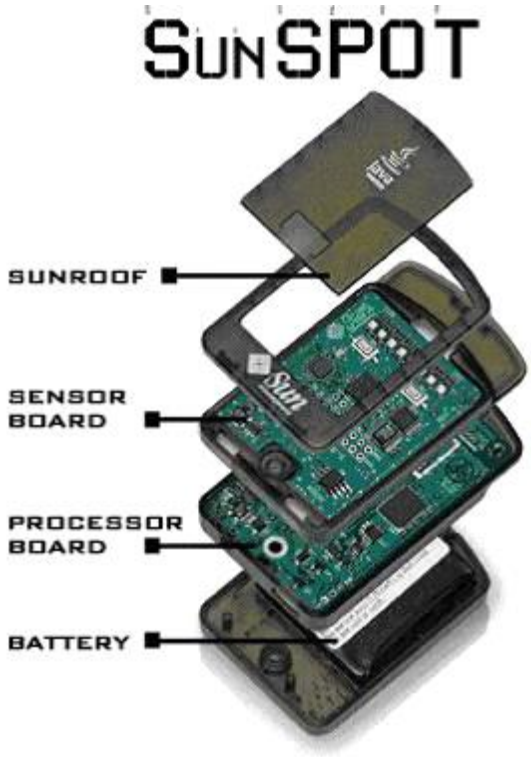
Μια πλήρης, ελευθέρου σήματος (free range) συσκευή SunSPOT αποτελείται από έναν επεξεργαστή SunSPOT με έναν πίνακα αισθητήρων και μία μπαταρία. Είναι συσκευασμένο σε ένα περίβλημα από πλαστικό. Ο μικρότερος σταθμός βάσης SunSPOT αποτελείται από μόλις έναν

επεξεργαστή σε ένα πλαστικό περίβλημα.

-eSPOT - Πρόκειται για την τωρινή έκδοση του SUN SPOT και αποτελείται από ένα κύριο board με μπαταρίες λιθίου, επεξεργαστή, μνήμη, 802.15.4 radio και σύνδεσμο για προσθήκη κάρτας επέκτασης. Στην συγκεκριμένη έκδοση τα SPOTs έχουν μια κάρτα επέκτασης το eDEMO με επιταχυνσιόμετρο, μετατροπέα ADC, ψηφιακές εισόδους/εξόδους(GPIO), 2 κουμπιά και 8 led.

-Basestation – Το basestation είναι μια συσκευή που περιέχει το κύριο board του eSPOT χωρίς μπαταρίες και κάρτα επέκτασης. Η τροφοδοσία παρέχεται από ένα USB καλώδιο που συνδέεται με ένα υπολογιστή. Το basestation χρησιμοποιείται για να επικοινωνούν εφαρμογές που τρέχουν σε ένα υπολογιστή με τα SPOTS.

Η τωρινή διαμόρφωση των SPOTs όπως φαίνεται στην παρακάτω εικόνα περιλαμβάνει το Processor Board (cpu/radio) και το Sensor Board που περιέχει τους αισθητήρες. Σε αυτό το κεφάλαιο θα αναφέρουμε τα βασικά στοιχεία του hardware που χρησιμοποιούνται και μια συνοπτική περιγραφή των χαρακτηριστικών τους. Επίσης θα αναλύσουμε τα σημαντικά υποσυστήματα της πλατφόρμας και την λειτουργικότητα του Sensor Board.



Εικόνα 4: Τα μέρη από τα οποία αποτελείται ένα SunSPOT: οθόνη, πίνακας αισθητήρων, πίνακας επεξεργαστή, μπαταρία

Αρχιτεκτονική του Πίνακα Επεξεργαστή (Processor board) του SunSPOT

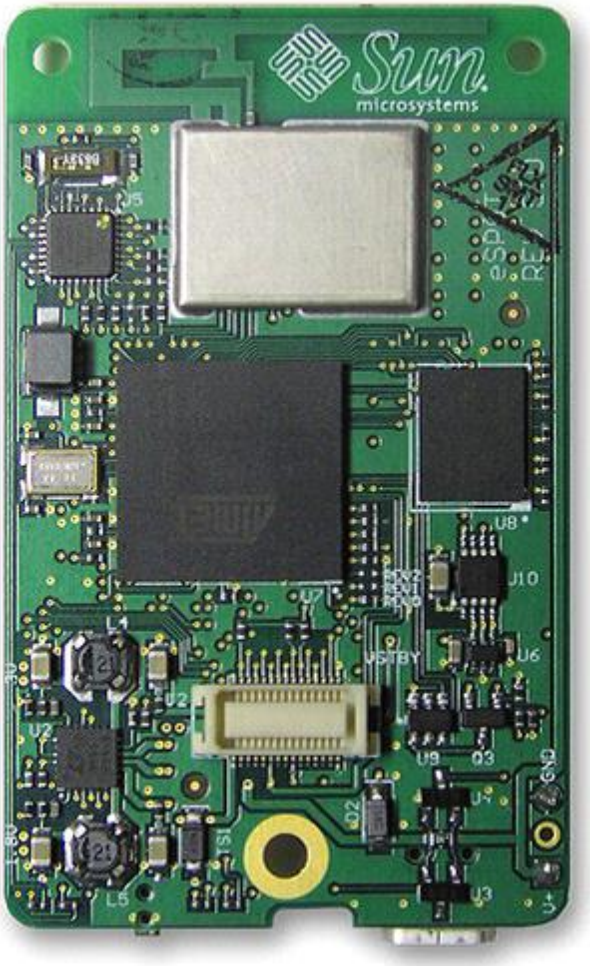
Επεξεργαστής

Κάθε SunSPOT διαθέτει έναν επεξεργαστή 400MHz 32-bit με πυρήνα ARM920T με 1M μνήμη RAM και 8M μνήμη Flash. Ο επεξεργαστής του SunSPOT έχει 2.4GHz ραδιοσυχνότητα με ενσωματωμένη κεραία στην κάρτα. Η ραδιοσυχνότητα είναι ένα TI CC2420 (πρώην ChipCon) και είναι συμβατό με IEEE 802.15.4.

Κάθε κάρτα επεξεργαστή έχει μία θύρα USB (που χρησιμοποιείται για τη σύνδεση με υπολογιστή). Υπάρχουν δύο LED, ένα κόκκινο και ένα πράσινο.

Τέλος, υπάρχει ένας 8-bit μικροελεγκτής Atmel Atmega88 που χρησιμοποιείται ως ελεγκτής ισχύος.

Ο επεξεργαστής ARM920T ARM Thumb processor της ATMEL περιέχεται σε ένα SOC (System On Chip) κύκλωμα το AT91RM9200. Σε κανονική λειτουργία καταναλώνει 44mW και η μέγιστη ταχύτητα του ρολογιού φτάνει τα 180MHz. Το SOC ενσωματώνει 16Kbyte cache εντολών, και 64-way associative 16Kbyte cache δεδομένων. Η MMU (ARMv4) έχει ένα TLB buffer 64 στοιχείων για δεδομένα και άλλον ένα TLB 64 στοιχείων για μετάφραση εντολών. Η πρόσβαση στην εξωτερική μνήμη(flash, pSRAM) γίνεται από το EBI δίαυλο, ο ελεγκτής του διαύλου είναι ρυθμισμένος ώστε να να εκκινεί το σύστημα(διαδικασία boot) από την flash όπου βρίσκεται η Squawk VM. Επίσης το SOC περιλαμβάνει μια μεγάλη συλλογή από interfaces για περιφερειακές συσκευές όπως θύρες USB host/devive , ethernet MAC, προγραμματιζόμενος ελεγκτής I/O (PIO), ελεγκτές SPI/USART/I2C/I2S, και 3 16-bit χρονιστές/μετρητές. Επιπλέον ενσωματώνεται και ένας DMA controller (PDC) για άμεσες και γρήγορες εγγραφές στην μνήμη και στους διαύλους USART/I2S/SPI. Λόγο του μικρού μεγέθους της συσκευής οι USB host και η μια USART θύρες δεν χρησιμοποιούνται όπως και τα TWI/I2S/Ethernet MAC interfaces. Επειδή όμως όλα τα σήματα υπάρχουν στο βύσμα του Processor board που το διασυνδέει με την κάρτα επέκτασης(Sensor board), μπορούμε να χρησιμοποιήσουμε τα παραπάνω interfaces αν προσθέσουμε τις κατάλληλες φυσικές διασυνδέσεις και γράψουμε τους αντίστοιχους drivers.



Εικόνα 5: Ο επεξεργαστής ενός SunSPOT

Μπαταρία

Η μπαταρία που χρησιμοποιείται στα πλήρη SunSPOTs είναι επαναφορτιζόμενη ιόντων λιθίου Li-ION στα 3.7V με χωρητικότητα 720mAH. Η μπαταρία ενσωματώνει κυκλώματα για την προστασία της από πλήρη αποφόρτιση, από υπερφόρτιση και από υψηλή τάση. Η φόρτιση μπορεί να γίνει κάθε φορά που η διασύνδεση USB έχει συνδεθεί σε ένα PC ή USB hub είτε χρησιμοποιώντας ένα USB καλώδιο με βύσμα τύπου B είτε από οποιαδήποτε πηγή 5Volt (+/- 10%). Όταν δεν χρησιμοποιείται χάνει περίπου 2% της χωρητικότητας κάθε μήνα και σε περιπτώσεις υψηλής θερμοκρασίας ο ρυθμός αυτός αυξάνει. Τα κυκλώματα φόρτισης και διαχείρισης ρεύματος είναι ρυθμισμένα με ακρίβεια για να λειτουργούν με τον συγκεκριμένο τύπο μπαταρίας και για αυτό δεν πρέπει να αντικατασταθεί από άλλου τύπου.

Σημειωτέον ότι ο σταθμός βάσης (basestation) SunSpot δεν έχει καμία μπαταρία, τροφοδοτείται μέσω της σύνδεσης USB με τον υπολογιστή υποδοχής.

Ασύρματος πομποδέκτης (wireless radio)

Τα SPOT για την ασύρματη μετάδοση δεδομένων ενσωματώνει τον ασύρματο πομποδέκτη CC2420. Το CC2420 συμμορφώνεται με το πρότυπο IEEE 802.15.4 και λειτουργεί σε συχνότητες από 2.4GHz ως 2.4835GHz(οι συχνότητες φαίνονται στο παρακάτω πίνακα), οι συχνότητες αυτές ανήκουν στην ISM ζώνη και εξαιρούνται αδειοδότησης στην Ελλάδα σύμφωνα με τον νόμο 399/3-4-2006. Το κύκλωμα CC2420 εκτός από τον πομποδέκτη περιέχει δυο 128byte FIFOs για τα TX και RX δεδομένα, δυνατότητα για μέτρηση RSSI (received signal strength indication) με ευαισθησία 100db και ρύθμιση ισχύς του πομπού από -24dBm ως 0dBm(οι τιμές φαίνονται στο παρακάτω πίνακα). Ο πρακτικός ρυθμός μετάδοσης δεδομένων φτάνει τα 250Kbit/s ενώ η ευαισθησία του δέκτη είναι -90dBm. Για τα σήματα ελέγχου και δεδομένων από και προς το CC2420 στο Processor Board χρησιμοποιούνται PIO θύρες και ο δίαυλος SPI. Στις PIO θύρες συνδέονται τα σήματα ελέγχου όπως reset, power down, start of frame(SFD) και σήματα κατάστασης όπως FIFO και FIFOP που ενημερώνουν αν η ουρά δεδομένων είναι άδεια ή αν έχουν ληφθεί δεδομένα. Ο δίαυλος SPI χρησιμοποιείται για την μεταβίβαση δεδομένων προς το CC2420. Το κύκλωμα καταναλώνει 20mA όταν ο δέκτης λαμβάνει δεδομένα και 18mA κατά την διάρκεια μετάδοσης με ισχύ 0dBm.

Η κεραία του SPOT είναι τύπου inverted-F, τυπωμένη στην άνω επιφάνεια του PCB(Printed Circuit Board) του Main Board. Είναι σχεδιασμένη για να συντονίζεται στη συχνότητα 2450MHz με ωμική αντίσταση 115Ω. Λόγο της θέσης την κεραίας θα πρέπει να αποφεύγουμε την τοποθέτηση μεταλλικών αντικειμένων ή γραμμών τροφοδοσίας κοντά σε αυτήν. Σε εξωτερικό χώρο, κάτω από καλές καιρικές συνθήκες η εμβέλεια φτάνει τα 100m ενώ σε εσωτερικούς χώρους περιορίζεται στα 30m.

Ελεγκτής τροφοδοσίας

Πρόκειται για τον 8-bit μικροελεγκτή Atmega88 της Atmel. Έχει ενσωματωμένο firmware που είναι υπεύθυνο για την λειτουργία του 64-bit ρολογιού, την επαναφορά της συσκευής σε περίπτωση που δεχτεί εξωτερικό interrupt και την επαναφορά ή είσοδό σε Deep-Sleep όταν πιεστεί το attention κουμπί. Η επικοινωνία με τον επεξεργαστή γίνεται μέσω του SPI διαύλου, από τον οποίο μεταφέρονται εντολές και δεδομένα κατάστασης από και προς τον Atmega88. Επίσης ο ελεγκτής μετράει και παρακολουθεί το φορτίο της μπαταρίας και τις τάσεις της USB, της μπαταρίας, και των εσωτερικών υποσυστημάτων χρησιμοποιώντας ένα 10-bit ACD. Ακόμα ο Atmega88 ελέγχει το power LED και δηλώνει διαφορετικές καταστάσεις(προβληματικές ή όχι) του SPOT μέσω ενδείξεων αυτού του LED. Για παράδειγμα όταν ανιχνεύσει ότι η μπαταρία έχει σχεδόν αποφορτιστεί ο ελεγκτής θα θέσει το power LED μόνιμα κόκκινο. Όλες οι πιθανές ενδείξεις παρουσιάζονται στον παρακάτω πίνακα.

Δίαυλοι επικοινωνίας του Main Board

Η επικοινωνία μεταξύ των SPOTs και workstation γίνεται κυρίως μέσω του διαύλου USB και για την διασύνδεση υπάρχει μια υποδοχή mini USB τύπου B. Η USB client συσκευή στα SPOTs είναι συμβατή με τα πρότυπα USB 1.1 και USB 2.0 και υποστηρίζει ACM modem για την σειριακή μετάδοση. Για την επικοινωνία ανάμεσα σε εσωτερικές συσκευές του Processor board και μεταξύ του Processor board και του Sensor board(κάρτα επέκτασης/αισθητήρων) χρησιμοποιείται το SPI και το PIO. Το SPI είναι ένας σειριακός δίαυλος για την επικοινωνία με τον ασύρματο πομποδέκτη IC CC2420, τον power controller και τον έλεγχο των LEDs του eDEMO board. Το PIO interface ελέγχει το activity LED που βρίσκεται αριστερά της mini USB υποδοχής καθώς και τα σήματα ελέγχου και κατάστασης του ασύρματου πομποδέκτη, όπως για παράδειγμα ότι το κανάλι είναι ελεύθερο για μετάδοση ή ότι η RX ουρά είναι πλήρης. Τέλος μέσω του PIO μεταφέρονται τα σήματα ελέγχου του κυκλώματος που ρυθμίζει την τροφοδοσία ρεύματος στην USB θύρα.

Μνήμη

Η μνήμη στο Processor Board είναι η Spansion S71PL032J40, και αποτελείται από 4Mbyte NOR flash και 512Kbyte pSRAM(pseudo-SRAM) που βρίσκονται στο ίδιο chip. Ο χρόνος πρόσβασης(access time) για την pSRAM είναι 70nsec και για την Flash 65nsec και έχουν 16- bit δίαυλο δεδομένων. Και οι δυο χρησιμοποιούν τροφοδοσία 3Volt και σε κανονικές συνθήκες λειτουργίας η κατανάλωση είναι 25ma για την pSRAM και 22ma για την Flash. Τα δεδομένα στην pSRAM διατηρούνται όσο το SPOT είναι συνδεδεμένο σε κάποια τροφοδοσία ή μπαταρία. Όταν στο SPOT είναι σε κατάσταση deep-sleep, που τα περισσότερα υποσυστήματα δεν τροφοδοτούνται για εξοικονόμηση ενέργειας η pSRAM καταναλώνει περίπου 8μA για την διατήρηση των δεδομένων της ενώ η flash απενεργοποιείται. Η flash είναι προγραμματισμένη ήδη από το εργοστάσιο και περιέχει τον bootloader, την Squawk VM, τις βασικές βιβλιοθήκες και μια προ εγκατεστημένη εφαρμογή (bounce demo).

Κύκλωμα τροφοδοσίας

Το SPOT μπορεί να λειτουργήσει χρησιμοποιώντας οποιοδήποτε συνδυασμό από της εξής πηγές: την επαναφορτιζόμενη μπαταρία, USB Host ,είτε εξωτερική τροφοδοσία. Το κύκλωμα τροφοδοσίας είναι υπεύθυνο για να φορτίζει την ενσωματωμένη μπαταρία, να ρυθμίζει το ρεύμα που παρέχεται στα υποσυστήματα του Processor Board και του Sensor Board είτε το SPOT βρίσκεται σε κανονική λειτουργία είτε σε deep-sleep. Το κύκλωμα αποτελείται από δύο τμήματα-κυκλώματα, κάθε ένα με διαφορετική λειτουργία LTC3455 και το TPS79730. Το LTC3455 έχει ενσωματωμένο, ένα κύκλωμα για την φόρτιση της μπαταρίας Li-ION ένα διαχειριστή ρεύματος για την USB και ένα διπλό σταθεροποιητή τάσης. Το LTC3455 διαχειρίζεται το ρεύμα που λαμβάνεται από την USB. Ανάλογα με τις απαιτήσεις της συσκευής, ο επεξεργαστής επιτρέπει την κατανάλωση περισσότερου ρεύματος από την USB. Το TPS79730 είναι ένας σταθεροποιητής τάσης και παρέχει μικρή ποσότητα ρεύματος στα 3Volt στην περίπτωση που το SPOT εισέλθει σε κατάσταση stand-by, επίσης παρέχει σταθερό ρεύμα στον Atmega88 και στην pSRAM και σε περίπτωση που η τάση πέσει κάτω από τα ασφαλή όρια λειτουργίας του επεξεργαστή τον απενεργοποιεί. Τα SPOT έχουν ειδικό firmware για εξοικονόμηση ενέργειας που μπορεί να θέσει την συσκευή σε τρεις καταστάσεις λειτουργίας:

- Run – Είναι η βασική κατάσταση στην οποία όλοι οι επεξεργαστές και το radio τροφοδοτούνται και λειτουργούν κανονικά. Η κατανάλωση σε αυτήν την κατάσταση φτάνει κυμαίνεται από 70mA ως 120mA, ενώ η κάρτα επέκτασης μπορεί να καταναλώνει μέχρι 400mA.
- Idle - Σε αυτή την κατάσταση το ρολόι του επεξεργαστή σταματάει και το radio απενεργοποιείται ενώ η κατανάλωση πέφτει στα 24mA.
- Deep-Sleep - Σχεδόν όλα τα κυκλώματα τροφοδοσίας απενεργοποιούνται εκτός από το κύκλωμα που δίνει ελάχιστο ρεύμα για την διατήρηση των δεδομένων της pSRAM. Η επαναφορά της συσκευής από την κατάσταση Deep-Sleep διαρκεί περίπου 2msec με 10msec.

Για να εισέλθει η συσκευή σε κατάσταση χαμηλής κατανάλωσης(Deep-Sleep) πρέπει το radio να είναι απενεργοποιημένο, να μην παρέχεται ρεύμα από εξωτερική συσκευή και να μην είναι ενεργοποιημένη η USB. Το SPOT εισέρχεται στις καταστάσεις Deep-Sleep και idle καλώντας κατάλληλες συναρτήσεις της βιβλιοθήκης. Επιπλέον μπορούσαμε να θέσουμε την συσκευή σε Deep-Sleep πατώντας το attention κουμπί για περισσότερα από 3 δευτερόλεπτα. Για να εξέλθει η συσκευή από Deep-Sleep πρέπει να χρησιμοποιήσουμε κάποιο εξωτερικό interrupt ή να πιέσουμε το attention κουμπί. Η παρακάτω εικόνα δείχνει τις μεταβάσεις που μπορεί να γίνουν μεταξύ των διαφορετικών καταστάσεων λειτουργίας.

Στοιχεία του Sensor Board (Πίνακας αισθητήρων)

Το Sensor Board είναι η κάρτα επέκτασης(daughterboard) του eSPOT. Αυτή είναι ενσωματωμένη στα SPOT που υπάρχουν στο αναπτυξιακό της SUN και προσφέρει μια ποικιλία από αισθητήρες και I/O θύρες. Στο eSPOT Processor Board μπορούν να συνδεθούν και διαφορετικές κάρτες επέκτασης και αυτή την στιγμή είναι υπό σχεδίαση αρκετές κάρτες με πιο προηγμένες δυνατότητες και πιο ευαίσθητα όργανα. Προϋπόθεση για την προθήκη μιας κάρτας επέκτασης στο eSPOT είναι να συνδέεται με τον Processor Board μέσω ενός βύσματος Hirose DF17-30, να υποστηρίζει το SPI interface αφού μέσω αυτού του διαύλου γίνεται η επικοινωνία και να περιέχει μια SPI flash για την

αποθήκευση πληροφορίας σχετικά με τις παραμέτρους λειτουργίας της. Στο παρακάτω σχήμα βλέπουμε την διασύνδεση του eDEMO με τα υπόλοιπα στοιχεία του SPOT.

Ο πίνακας αισθητήρων αποτελείται από:

2G/6G επιταχυνσιόμετρο τριών αξόνων

αισθητήρα θερμοκρασίας

αισθητήρα φωτός

8 τρι-χρώμα LEDs

6 αναλογικές εισόδους

2 στιγμιαίους διακόπτες

5 γενικού σκοπού I / O pins και 4 υψηλού ρεύματος εξόδου pins

Για τον χειρισμό των παραπάνω στοιχείων η SUN έχει κατάλληλους drivers και βιβλιοθήκες με κλάσεις για τον χειρισμό τους. Για παράδειγμα αν θέλουμε να ελέγξουμε το πράσινο LED πρέπει να χρησιμοποιήσουμε την κλάση Iled της βιβλιοθήκης

Iled theLed = Spot.getInstance().getGreenLed();

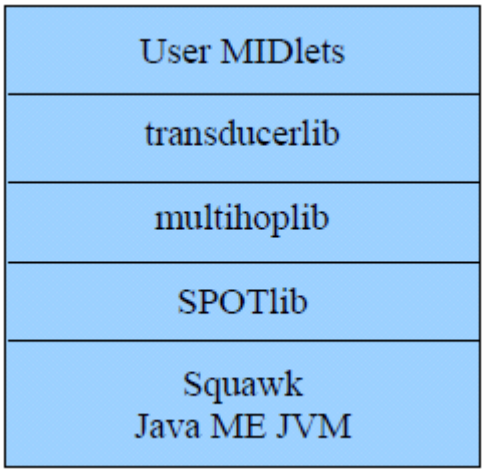
και για τον χειρισμό του LED:

theLed.setOn();

theLed.setOff();

4.2 Αρχιτεκτονική των SunSPOTs

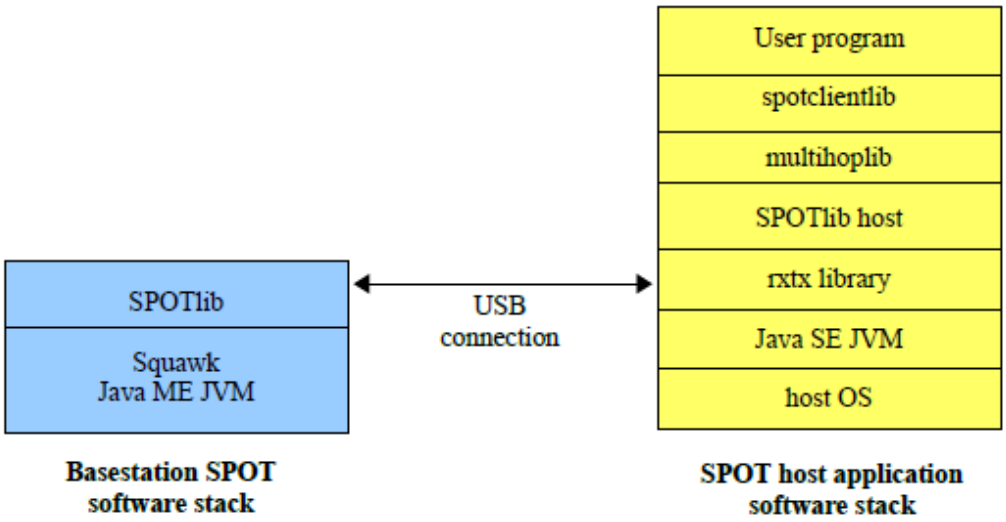
Για την ανάπτυξη λογισμικού για την πλατφόρμα SunSPOT είναι χρήσιμο να γνωρίζει κανείς την αρχιτεκτονική τους.



Εικόνα 6: Επίπεδα λογισμικού ενός free-range SPOT

Αρχιτεκτονική απλού free range SunSPOT

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



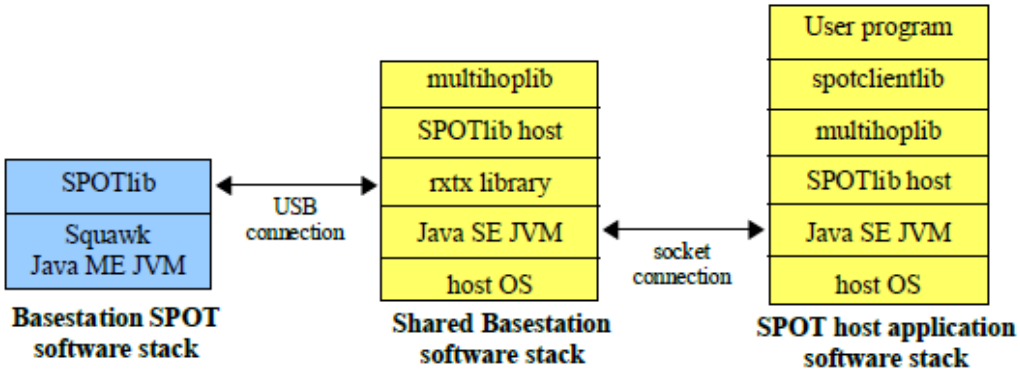
Εικόνα 7: Αρχιτεκτονική SPOT host application

Αρχιτεκτονική host-application SunSPOT

Πάλι στην κορυφή βρίσκεται μία εφαρμογή γραμμένη από το χρήστη host SPOT, που είναι ένα απλό πρόγραμμα σε Java SE. Μπορεί να εκτελέσει όλες τις λειτουργίες που εκτελεί ένα συνηθισμένο πρόγραμμα σε JAVA: είσοδο/έξοδο σε αρχείο, απεικόνιση Swing GUI's κτλ. Μπορεί επίσης να στείλει και να λάβει μηνύματα μέσω ραδιοσυχνότητας με free-range SPOTs αν ένας σταθμός βάσης basestation είναι συνδεδεμένος με τον υπολογιστή.

Στο κάτω-κάτω επίπεδο βρίσκεται το λειτουργικό σύστημα του host: Linux, Windows, Mac OS X ή Solaris. Ακριβώς πάνω από το Λειτουργικό Σύστημα είναι η Java SE JVM μαζί με όλες τις βιβλιοθήκες της Java. Ενδιάμεσα βρίσκονται οι διάφορες βιβλιοθληκες των SPOT. Η πρόσβαση στη συσκευή του SPOT και η βασική είσοδος/έξοδος παρέχεται από την host έκδοση της SPOTlib . Αυτή περιλαμβάνει πρόσβαση στο χαμηλού επιπέδου πρωτόκολλο MAC radio, το οποίο χρησιμοποιεί είτε σύνδεση USB για να έχει πρόσβαση στο ράδιο του σταθμού βάσης είτε χρησιμοποιεί συνδέσεις socket για να επικοινωνεί με άλλες host applications. Η βιβίοθήκη multihoplib και εδώ παρέχει υψηλότερου επιπέδου radio protocols όπως το Radiogram και το Radiostream και ταυτόχρονα φροντίζει τη δρομολόγηση των πακέτων σε που δεν είναι σε άμεση επαφή με αυτό. Η βιβλίοθήκη spotclientlib δίνει πρόσβαση σε ένα πλήθος εντολών που μπορούν να σταλούν σε ένα ελευθέρου

βεληνεκούς (free-range) SPOT. Αυτό περιλαμβάνει την εντολή "Hello" που χρησιμοποιείται για να ανακαλύψει SPOTs εντός της ραδιοσυχνότητας. Η βιβλιοθήκη rxtx χρησιμοποιείται για σειριακή είσοδο/έξοδο μέσω της usb σύνδεσης με το σταθμό βάσης basestation. Σημειωτέον ότι το Solarium και όλες οι SPOT SDK ant εντολές αποτελούν SPOT host applications.

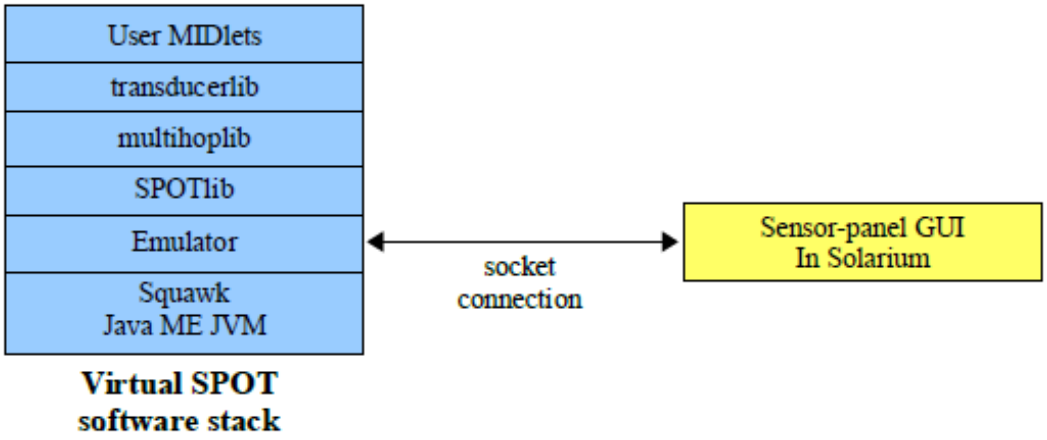


Εικόνα 8: Αρχιτεκτονική SPOT-basestation

Αρχιτεκτονική SPOT-basestation

Ο σταθμός βάσης παρέχει έναν τρόπο επικοινωνίας μεταξύ των host applications με τα free-range SPOTs χρησιμοποιώντας τη ραδιοσυχνότητα του.

Αξιοσημείωτο είναι ότι η host application τρέχει αποκλειστικά στον host υπολογιστή δεν τρέχει καθόλου κώδικας του χρήστη στο σταθμό βάσης. Τα πακέτα στέλνονται μέσω USB στο basestation το οποίο τα στέλνει έξω μέσω της ραδιοσυχνότητάς του. Παρομοίως 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. Τα πακέτα που παραλαμβάνονται από άλλες host applications μέσω μίας σύνδεσης socket προωθούνται στα SPOTs χρησιμοποιώντας τη ραδιοσυχνότητα του basestation.



Εικόνα 9: Αρχιτεκτονική Virtual SPOT

Αρχιτεκτονική Virtual SPOT

Όταν δημιουργείται ένα νέο virtual SPOT στο Solarium, μία νέα διαδικασία εκκινείται 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. Για παράδειγμα, όταν η εφαρμογή του SPOT αλλάζει την RGB τιμή του ενός LED, αυτή η πληροφορία περνάει στον κώδικα του virtual SPOT GUI που ενημερώνει την απεικόνιση αυτού του LED με τη νέα RGB τιμή. Παρομοίως όταν ο χρήστης κλικάρει έναν από τους διακόπτες του virtual SPOT με το ποντίκι, το Solarium στέλνει ένα μήνυμα στον κώδικα του προσομοιωτή ότι ο διακόπτης έχει κλικαριστεί πράγμα που γίνεται αντιληπτό από την SPOT application.

Κάθε virtual SPOT έχει τη δική του Squawk VM που τρέχει ως ξεχωριστή διεργασία στον host υπολογιστή. Κάθε Squawk VM περιέχει μία στοίβα ραδιοσυχνοτήτων σαν μέρος της βιβλιοθήκης SPOT που επιτρέπει στη SPOT εφαρμογή να επικοινωνεί με άλλες εφαρμογές SPOT που τρέχουν στον ίδιο υπολογιστή όπως άλλα virtual SPOTs χρησιμοποιώντας sockets ή πραγματικά SPOTs μέσω ραδιοσυχνότητας αν τρέχει/χρησιμοποιείται ένας σταθμός βάσης.

4.3 Λογισμικό

Δεν υπάρχει λειτουργικό σύστημα πάνω στο οποίο τρέχει το SunSpot . Το

SunSpot τρέχει σε Java VM.

Τα SunSpot χρησιμοποιούν μια εφαρμογή Java ME, που ονομάζεται Squawk,

που υποστηρίζει CLDC 1.1 και MIDP 1.0, και επιπρόσθετα παρέχει τη βασική

λειτουργικότητα ενός OS. Η VM εκτελείται απευθείας από τη μνήμη flash. Όλα τα προγράμματα οδήγησης συσκευών (device drivers), είναι επίσης, γραμμένα σε Java.

Το λογισμικό του SunSpot είναι λογισμικό ανοιχτού κώδικα. Όλο το λογισμικό SPOT Sun έχει κυκλοφορήσει ως open source υπό την GNU General Public License (GPL κατά 2,0). Ανάπτυξη του κώδικα πραγματοποιείται σε java.net.

Για να βρούμε τον πηγαίο κώδικα Sun SPOT, αφού εγκαταστήσουμε το SunSPOT SDK θα υπάρξει ένας "src" κατάλογος που περιέχει διάφορα αρχεία jar. Επεκτείνοντάς τα αποκτούμε πρόσβαση στη βιβλιοθήκη με τον πηγαίο κώδικα SunSPOT.

Ο προγραμματισμός στην πλατφόρμα SUN SPOT γίνεται με την γλώσσα Java. Συγκεκριμένα οι εφαρμογές ακολουθούν τις προδιαγραφές του MIDP(Mobile Information Device Profile) που χτίζεται πάνω στο CLDC και προσθέτει ένα επιπλέον API για εφαρμογές σε embedded συσκευές. Το MIDP χρησιμοποιείται σε πολλά ενσωματωμένα (embedded) συστήματα και συσκευές όπως για παράδειγμα τα κινητά τηλέφωνα. Τα συγκεκριμένα προγράμματα που ακολουθούν τις παραπάνω προδιαγραφές καλούνται MIDlets και έχουν συγκεκριμένη δομή και περιορισμούς. Τα MIDlets τρέχουν σε μια μικρή Java ME(J2ME) virtual machine (VM) που λέγεται Squawk VM. Όπως προαναφέραμε τα SPOT δεν έχουν λειτουργικό σύστημα, αλλά τον ρόλο του OS τον αναλαμβάνει η Squawk VM. Μαζί με τα MIDlet του χρήστη αλλά σε “χαμηλότερο” επίπεδο τρέχουν και μια πλειάδα άλλων εφαρμογών που δεν είναι άμεσα “ορατές” στον χρήστη:

- Ο bootloader – που είναι υπεύθυνος για τη USB σύνδεση, εκκινεί τις εφαρμογές και επικοινωνεί με τα ant scripts του PC που είναι συνδεδεμένο το SPOT.
- bootstrap suite – που περιλαμβάνει τις πρότυπες κλάσεις της Java ME.

- library suite – που περιλαμβάνει την βιβλιοθήκη με τις κλάσεις σχετικές με το SUN SPOT.

Η Squawk VM χρησιμοποιεί ανεξάρτητες περιοχές για εκτέλεση εφαρμογών, τα isolates. Κάθε isolate συνιστά ένα διαφορετικό σύνολο από threads και αντικείμενα που σχετίζονται με αυτά. Το SPOT έχει πάντα ένα master isolate, στο οποίο τρέχουν deamon threads που διαχειρίζονται βασικές λειτουργίες του. Αυτά τα threads είναι τμήμα της βασικής βιβλιοθήκης του SunSPOT και φροντίζουν για την διαχείριση ενέργειας (απενεργοποιώντας υποσυστήματα που δεν χρησιμοποιούνται), παρακολουθούν την κατάσταση της USB και αποτελούν μέρος της υλοποίησης του radiostack. Τα υπόλοιπα isolates που μπορεί να δημιουργηθούν καλούνται child isolates. Η προκαθορισμένη (default) συμπεριφορά του SPOT είναι οι εφαρμογές του χρήστη να τρέχουν στο master isolate αν και αυτό δεν είναι υποχρεωτικό. Τα threads χρησιμοποιούνται στα MIDlet μέσω της κλάσης Thread, και υπάρχει η δυνατότητα να ρυθμιστεί η προτεραιότητά τους από 1 (Thread.MIN_PRIORITY) ως 10 (Thread.MAX_PRIORITY). Επίσης υπάρχει η δυνατότητα να δοθούν και υψηλότερες προτεραιότητες όπως οι “προτεραιότητες συστήματος” (system priorities), αυτές όμως αφορούν ορισμένα deamon threads και δεν προορίζονται για χρήση από τα MIDlets. Εδώ πρέπει να αναφέρουμε ότι για την σωστή λειτουργία των threads των βιβλιοθηκών του SPOT, οι προγραμματιστές θα πρέπει να αναθέτουν προτεραιότητα χαμηλότερη από 5 (Thread.NORMAL) όταν οι εφαρμογές τους είναι cpu-bounded. Υψηλές προτεραιότητες μπορεί να προκαλέσουν προβλήματα στα threads της SPOT library, όπως την απώλεια broadcast μηνυμάτων.

Στην Java SE μια εφαρμογή θα πρέπει να περιέχει μία main() μέθοδο ή να υλοποιεί το Applet interface αν πρόκειται να εκτελεστεί από έναν browser. Όμως στην Java ME, που υλοποιεί η Squawk VM, κάθε εφαρμογή που υλοποιούμε πρέπει να είναι συμβατή με το πρότυπο MIDlet. Όλες οι εφαρμογές για τα SPOT πρέπει να κληρονομούν (extend) τα στοιχεία της κλάσης MIDlet και να υλοποιούν τις μεθόδους:

- startApp() - Η μέθοδος αυτή καλείται όταν πρόκειται να εκτελεστεί το MIDlet.

- PauseApp() - Η μέθοδος αυτή καλείται όταν πρόκειται να ανασταλεί η εκτέλεση του MIDlet.
- destroyApp() - Η μέθοδος αυτή καλείται όταν το MIDlet τερματίζεται από το σύστημα, όπως σε περιπτώσεις που το isolate που εκτελείται το MIDlet καταστραφεί με την μέθοδο ”Isolate.exit()” είτε τερματίσει η VM με την μέθοδο VM.stopVM() είτε το MIDlet προκαλέσει μια εξαίρεση εκτός της MIDletStateChangeException.

Προαιρετικά μπορεί να υπάρχει μια μέθοδος δημιουργός (constructor) χωρίς ορίσματα. Για τον τερματισμό ενός MIDlet από τον προγραμματιστή πρέπει να χρησιμοποιείται η μέθοδος notifyDestroyed(), οι εφαρμογές δεν πρέπει ποτέ να καλούν την μέθοδο System.exit(). Όλες λοιπόν οι εφαρμογές για τα SPOT έχουν την παρακάτω βασική δομή (εισάγουμε όλες τις κλάσεις που χρειαζόμαστε προκειμένου να έχουμε πλήρη λειτουργικότητα):

```
import com.sun.spot.peripheral.Spot;

import com.sun.spot.io.j2me.radiostream.*;

import com.sun.spot.io.j2me.radiogram.*;

import com.sun.spot.sensorboard.peripheral.ITriColorLED;

import com.sun.spot.sensorboard.EDemoBoard;

import com.sun.spot.peripheral.radio.IRadioPolicyManager;

import com.sun.spot.util.*;

import javax.microedition.io.*;

import javax.microedition.midlet.MIDlet;

import javax.microedition.midlet.MIDletStateChangeException;


public class SunSpotApplication extends MIDlet {


protected void startApp() throws MIDletStateChangeException {

    }
```

```
    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }
    protected void destroyApp(boolean unconditional) throws MIDletStateChangeException {
    }
}
```

Η δομή των φακέλων όταν αναπτύσσουμε μια εφαρμογή για τα SPOT πρέπει να είναι συγκεκριμένη. Στο φάκελο κάθε εφαρμογής πρέπει να υπάρχουν 2 αρχεία, ένα αρχείο build.xml και ένα build.properties που χρησιμοποιούνται από τα ant scripts για την μετάφραση και την εκτέλεση των MIDlet. Επιπλέον πρέπει να έχουμε και 3 υποφακέλους, έναν με όνομα src που τοποθετούμε τους καταλόγους με τα αρχεία πηγαίου κώδικα και έναν με το όνομα resources/META-INF που περιγράφεται στην επόμενη παράγραφο. Τέλος αν η εφαρμογή μας χρησιμοποιεί το NetBeans IDE τότε θα υπάρχει και ένας τρίτος φάκελος με όνομα nbproject, με τα περιεχόμενα του NetBeans project. Στο φάκελο resources/META-INF υπάρχει το αρχείο MANIFEST.MF που περιέχει πληροφορίες που χρησιμοποιούνται από την Squawk VM για την εκκίνηση των εφαρμογών. Συγκεκριμένα περιέχει τα όνομα των αρχικών κλάσεων των MIDlets και ορισμένες ιδιότητες αυτών. Επιπλέον ο φάκελος μπορεί να περιλαμβάνει αρχεία που ορίζει ο προγραμματιστής και είναι διαθέσιμα στην εφαρμογή όταν εκτελείται. Η δομή ενός τυπικού MANIFEST.MF είναι:

```
MIDlet-Name: Air Text demo
MIDlet-Version: 1.0.0
MIDlet-Vendor: Oracle
MIDlet-1: AirText, , org.sunspotworld.demo.AirTextDemo
MicroEdition-Profile: IMP-1.0
MicroEdition-Configuration: CLDC-1.1
SomeProperty: some value
```

Η σύνταξη κάθε γραμμής είναι <property-name>:<space><property-value. Η πιο σημαντική γραμμή για κάθε πρόγραμμα είναι η MIDlet-1:<όρισμα 1>,<όρισμα 2>,<όρισμα 3>. Το πρώτο όρισμα είναι μια περιγραφή της εφαρμογής και το τρίτο όρισμα είναι η κύρια κλάση του MIDlet. Το δεύτερο όρισμα είναι μια εικόνα που σχετίζεται με το MIDlet, αλλά στην παρούσα έκδοση δεν υποστηρίζεται αυτή η επιλογή. Μέσα από την εφαρμογή μπορούμε να διαβάσουμε τις τιμές για τις παραπάνω ιδιότητες χρησιμοποιώντας την μέθοδο <όνομα MIDlet>.getAppProperty("<property-name>"). Όλα τα αρχεία μέσα στον φάκελο resources είναι διαθέσιμα στην εφαρμογή κατά την διάρκεια που εκτελείται, μπορούμε να ανοίξουμε ένα stream εισόδου για να διαβάσουμε τα περιεχόμενά τους μέσω της μεθόδου InputStream is = getClass().getResourceAsStream("/<όνομα αρχείου>").

4.4 Βιβλιοθήκες του Συστήματος

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. Επίσης θα δώσουμε παραδείγματα για το πως μπορούμε να χειριστούμε διαφορετικά υποσυστήματα του SUN SPOT και του sensor board μέσω κλάσεων αυτών των βιβλιοθηκών.

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 παρακάτω συσκευές:

Radio Communication Library

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

- radiostream – Το radiostream παρέχει αξιόπιστη μετάδοση δεδομένων μεταξύ δυο κόμβων, επίσης χρησιμοποιεί buffers για καλύτερη απόδοση. Το πρωτόκολλο αυτό είναι stream-based.
- Radiogram – Στο radiogram πρωτόκολλο η μεταφορά δεδομένων γίνεται με datagrams, και 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 να μην φτάσει με την σειρά που στάλθηκε. Ενώ στην περίπτωση που τα datagrams

στέλνονται σε γειτονικό κόμβο(ένα hop), η μόνη περίπτωση είναι κάποια να παραλειφθούν περισσότερες από μια φορές.

Παράδειγμα χρήσης του radiostream: για να συνδεθούμε με ένα κόμβο με το πρωτόκολλο radiostream χρησιμοποιούμε τον παρακάτω κώδικα για να “ανοίξουμε” μια σύνδεση:

```
RadiostreamConnection conn =
(RadiostreamConnection)Connector.open("radiostream:<destAddr>:<portNo>
");
```

Το <destinationAddr> είναι η 64-bit διεύθυνση του προορισμού και η <portNo> είναι ένας αριθμός από 1 ως 255 που χαρακτηρίζει μοναδικά την συγκεκριμένη σύνδεση. Ο προγραμματιστής μπορεί να επιλέξει οποιοδήποτε αριθμό port θέλει από την παραπάνω περιοχή εκτός των ports 1 ως 31 που είναι δεσμευμένα για χρήση από το σύστημα. Προκειμένου να μπορούν δυο κόμβοι να χρησιμοποιήσουν ένα radiostream για να επικοινωνήσουν μεταξύ τους πρέπει και οι δυο να ανοίξουν ένα RadiostreamConnection στην την ίδια port και με τις αντίστοιχες διευθύνσεις. Αφού έχουμε ανοίξει μια radiostream σύνδεση μπορούμε να πάρουμε το stream εισόδου και εξόδου αυτός της σύνδεσης για να μεταδώσουμε δεδομένα, για παράδειγμα:

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

Παρακάτω ακολουθεί να απλό παράδειγμα με δυο προγράμματα:

<u>Program 1</u>

```
RadiostreamConnection conn =
(RadiostreamConnection)Connector.open("radiostream://0014.4F01.0000.000
6:100");
DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos = conn.openDataOutputStream();
try {
dos.writeUTF("Hello up there");
```

```
dos.flush();
System.out.println ("Answer was: " + dis.readUTF());
} catch (NoRouteException e) {
System.out.println ("No route to 0014.4F01.0000.0006");
} finally {
dis.close();
dos.close();
conn.close();
}
```

Program 2

```
RadiostreamConnection conn =
(RadiostreamConnection)Connector.open("radiostream://0014.4F01.0000.000
7:100");
DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos = conn.openDataOutputStream();
try {
String question = dis.readUTF();
if (question.equals("Hello up there")) {
dos.writeUTF("Hello down there");
} else {
dos.writeUTF("What???");
}
dos.flush();
} catch (NoRouteException e) {
System.out.println ("No route to 0014.4F01.0000.0007");
```

```
} finally {
dis.close();
dos.close();
conn.close();
}
```

Σε αυτό το παράδειγμα τα δυο προγράμματα ανοίγουν ένα radiostream connection και τα αντίστοιχα stream εισόδου/εξόδου. Μετά το πρόγραμμα 2 αναμένει να λάβει δεδομένα από το stream και το πρόγραμμα 1 στέλνει το μήνυμα "Hello up there". Αν το μήνυμα παραληφθεί σωστά το πρόγραμμα 2 απαντά “Hello down there” και η απάντηση τυπώνεται στο System.out stream από το πρόγραμμα 1. Τα δεδομένα στέλνονται ασύρματα όποτε γεμίσει το buffer του radiostream, αν θέλουμε να σταλούν τα δεδομένα άμεσα πρέπει να χρησιμοποιήσουμε την εντολή flush(). Επίσης αν το πρόγραμμα 2 ξεκινήσει πριν το 1 υπάρχει περίπτωση να χαθεί το μήνυμα "Hello up there", αφού η αποστολή θα γίνει πριν το πρόγραμμα 1 ζητήσει δεδομένα. Για να είμαστε σίγουροι ότι θα εκτελεστούν με την σωστή σειρά πρέπει να εφαρμόσουμε κάποιο είδος handshake ή να εισάγουμε μια καθυστέρηση πριν το πρόγραμμα 2 στείλει το μήνυμα. Σε περίπτωση που δεν μπορεί να βρεθεί μια διαδρομή προς τον προορισμό προκαλείται μια εξαίρεση NoRouteException.

Σε κάθε μετάδοση ενός πακέτου το MAC layer περιμένει την αποστολή ενός ACK(πακέτο επιβεβαίωσης), που δηλώνει την επιτυχή λήψη του πακέτου. Σε περίπτωση που ο τελικός προορισμός βρίσκεται σε απόσταση ενός hop, το MAC-level ack είναι αρκετό για να διασφαλίσουμε την σωστή λήψη. Σε διαφορετική περίπτωση το radiostream θα ζητήσει από τον προορισμό να στείλει ένα πακέτο επιβεβαίωσης πίσω στον αρχικό αποστολέα. Για την βελτίωση της απόδοσης του πρωτοκόλλου όταν στέλνουμε δεδομένα, το output stream δεν περιμένει το ACK από τον προορισμό αλλά επιστρέφει άμεσα. Σε περίπτωση που δεν παραλάβουμε το πακέτο επιβεβαίωσης σε κάποιο προκαθορισμένο χρονικό διάστημα τότε ξαναστέλνεται αυτόματα. Μετά από ένα αριθμό αποτυχημένων προσπαθειών προκαλείται μια εξαίρεση NoMeshLayerAckException στην επόμενη προσπάθεια για αποστολή δεδομένων. Σε αυτή την περίπτωση δεν μπορούμε να ξέρουμε πόσα από τα

δεδομένα που έχουμε στείλει έχουν πράγματι φτάσει στον προορισμό. Τέλος μια άλλη εξαίρεση που μπορεί να προκληθεί και στα στα δυο πρωτόκολλα (radiostream και radiogram), είναι η ChannelBusyException. Αυτή η εξαίρεση είναι ένδειξη ότι το κανάλι είναι απασχολημένο, δηλαδή ότι μεταδίδουν άλλες συσκευές.

Παράδειγμα χρήσης του radiogram

Το πρωτόκολλο radiogram ακολουθεί το μοντέλο client-server, και παρέχει επικοινωνία μεταξύ δυο συσκευών μέσω datagrams.

Για να “ανοίξουμε” μια σύνδεση από την πλευρά του server χρησιμοποιούμε τον ακόλουθο κώδικα:

RadiogramConnection conn =

(RadiogramConnection) Connector.open("radiogram://:<portNo>");

Η <portNo> είναι ένας αριθμός από 1 ως 255 που χαρακτηρίζει μοναδικά την συγκεκριμένη σύνδεση. Ο προγραμματιστής μπορεί να επιλέξει οποιοδήποτε αριθμό port θέλει από την παραπάνω περιοχή εκτός των ports 1 ως 31 που είναι δεσμευμένα για χρήση από το σύστημα. Για να “ανοίξουμε” μια σύνδεση από την πλευρά του client χρησιμοποιούμε τον ακόλουθο κώδικα:

RadiogramConnection conn =

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

Το <destinationAddr> είναι η 64-bit διεύθυνση του server και η <portNo> είναι η port που έχει ανοίξει το radiogram connection ο server που θέλουμε να συνδεθούμε. Τα δεδομένα μεταξύ του client και του server στέλνονται ως datagrams, προκειμένου να στείλουμε δεδομένα πρέπει να κατασκευάσουμε ένα αντικείμενο datagram από το connection που έχουμε ανοίξει, να γράψουμε σε αυτό τα δεδομένα που θέλουμε και μετά να το στείλουμε στον προορισμό μέσω του connection. Παρακάτω παραθέτουμε ένα παράδειγμα που έχει την ίδια λειτουργία με τα προγράμματα 1 και 2 της προηγούμενης ενότητας:

Client end

```
RadiogramConnection conn =
(RadiogramConnection)Connector.open("radiogram://0014.4F01.0000.0006:100");
Datagram dg = conn.newDatagram(conn.getMaximumLength());
try {
dg.writeUTF("Hello up there");
conn.send(dg);
conn.receive(dg);
System.out.println ("Received: " + dg.readUTF());
} catch (NoRouteException e) {
System.out.println ("No route to 0014.4F01.0000.0006");
} finally {
conn.close();
}
```

Server end

```
RadiogramConnection conn = (RadiogramConnection)Connector.open("radiogram://:100");
Datagram dg = conn.newDatagram(conn.getMaximumLength());
Datagram dgreply = conn.newDatagram(conn.getMaximumLength());
try {
conn.receive(dg);
String question = dg.readUTF();
dgreply.reset(); // reset stream pointer
dgreply.setAddress(dg); // copy reply address from input
if (question.equals("Hello up there")) {
dgreply.writeUTF("Hello down there");
```

```
} else {

dgreply.writeUTF("What???");

}

conn.send(dgreply);

} catch (NoRouteException e) {

System.out.println ("No route to " + dgreply.getAddress());

} finally {

conn.close();

}
```

Ορισμένα χαρακτηριστικά των datagrams που θα πρέπει να αναφέρουμε είναι:

- Τα datagrams που στέλνονται μέσω ενός datagram connection πρέπει να έχουν κατασκευαστεί από αυτό, χρησιμοποιώντας την μέθοδο newDatagram() του ανοιχτού connection. Δεν μπορούμε να στείλουμε datagrams σε ένα connection, τα οποία έχουν προέλθει από ένα άλλο.
- Ένα datagram connection που έχει ανοιχτεί για μια συγκεκριμένη διεύθυνση δεν μπορεί να χρησιμοποιηθεί για να σταλούν πακέτα σε άλλο προορισμό. Αν προσπαθήσουμε να στείλουμε ένα datagram σε διαφορετικό προορισμό θα προκληθεί εξαίρεση.
- Είναι δυνατό να έχουμε στην ίδια συσκευή να έχουμε ανοικτό ένα server και ένα client connection στην ίδια port. Τα datagrams που λαμβάνονται και προορίζονται για την συγκεκριμένη port θα προωθούνται στο server connection.
- Στην τωρινή υλοποίηση του πρωτοκόλλου στα SPOT όταν ο server κλείσει το connection, τότε κλείνει αυτόματα και το connection του client.

Τα radiograms connections μπορούν να χρησιμοποιηθούν για την αποστολή broadcast πακέτων. Η προκαθορισμένη συμπεριφορά είναι να αποστέλλονται τα broadcast σε ακτίνα 2 hop για να μπορούν να λαμβάνονται από κοντινές συσκευές που όμως βρίσκονται εκτός ακτίνας μετάδοσης. Ο κώδικας για τη αποστολή broadcasts πακέτων είναι ο εξής:

```
DatagramConnection conn =

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

Για την λήψη broadcasts πακέτων δεν μπορεί να χρησιμοποιηθεί το παραπάνω connection αλλά πρέπει να ορίσουμε ένα client connection ως εξής:

RadiogramConnection serverConn =

(RadiogramConnection)Connector.open("radiogram://");

Αν θέλουμε να τροποποιήσουμε τον αριθμό των hops που θα μεταβεί το broadcast μήνυμα τότε μπορούμε να χρησιμοποιήσουμε την παρακάτω μέθοδο στο “ανοιχτό” connection θέτοντας το n στον αριθμό των hops που θέλουμε :

((RadiogramConnection)conn).setMaxBroadcastHops(n);

Εδώ πρέπει να προσθέσουμε ότι ο μηχανισμός μετάδοσης broadcast δεν παρέχει καμία εγγύηση για σωστή παραλαβή, αφού δεν χρησιμοποιεί πακέτα επιβεβαίωσης, ούτε πραγματοποιούνται αναμεταδώσεις broadcast πακέτων. Στην περίπτωση που το μέγεθος των broadcast πακέτων είναι μεγάλο, τότε στέλνεται σε τμήματα (fragments) και αυξάνεται η πιθανότητα να συμβεί κάποιο σφάλμα σε τουλάχιστον ένα από αυτά. Το μέγιστο μέγεθος δεδομένων (payload) των broadcast πακέτων που μπορούμε να στείλουμε είναι 1260 bytes, ενώ το payload των πακέτων του 802.15.4 radio είναι περίπου 100 bytes. Στην περίπτωση που στέλνονται πολλά fragments τα SPOTs αντιμετωπίζουν και ένα επιπλέον πρόβλημα που οδηγεί σχεδόν σίγουρα στην απώλεια δεδομένων. Το πρόβλημα είναι ότι η συσκευή που δέχεται τα πακέτα έχει ένα περιορισμό στο πόσο γρήγορα μπορεί να αδειάζει τον packet buffer, κάτι που μπορεί να επιδεινώνεται σε περιπτώσεις που ο δέκτης έχει μεγάλο αριθμό από ενεργά threads ή τυχαίνει να καλεί συχνά τον gc (garbage collector). Για αυτό τον λόγο προτείνεται να μην στέλνουμε broadcast radiograms με περισσότερα από 200 bytes δεδομένων, δηλαδή το πολύ 2 radio packets για κάθε broadcast και να εισάγουμε μια καθυστέρηση 20ms μεταξύ διαδοχικών broadcast ώστε να προλαβαίνει ο δέκτης να τα παραλαμβάνει.

Sensor Board library

Πρόκειται για την βιβλιοθήκη που 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. Σε αυτή την ενότητα θα παρουσιάσουμε συνοπτικά με παραδείγματα πως μπορούμε να χρησιμοποιήσουμε τις κλάσεις αυτής της βιβλιοθήκης για να χειριστούμε 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 αντιστοιχία τμήματα του javadoc και να μελετήσει τα παραδείγματα που βρίσκονται στον φάκελο “[SpotSDKdirectory]/doc/AppNotes/”.