* 1. **ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΠΛΑΤΦΟΡΜΑΣ**
* υλοποίηση της προσοµοίωσης έγινε µέσω της πλατφόρµας SunSPOT

(Sun Small Programable Object Technology (SPOT)).

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 ραδιoσυχνότητα), 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.

33



**Εικόνα 9: 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

34

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 10: Μία συσκευή SunSPOT**

Εξοµοιωτής ή προσοµοιωτής για Sun SPOT

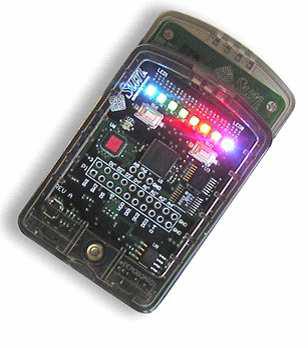
Η έκδοση 6.0 περιλαµβάνει έναν εξοµοιωτή ως µέρος του SPOTWorld. Αυτός ο εξοµοιωτής µπορεί να εκτελεί µια εφαρµογή SunSPOT στον υπολογιστή. Αυτό επιτρέπει τον έλεγχο ενός προγράµµατος πριν από την ανάπτυξη σε ένα πραγµατικό SunSPOT, ή αν ένα πραγµατικό SunSPOT δεν είναι διαθέσιµο.

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

35

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 11: Συσκευή SunSPOT σε λειτουργία**

36

**4.1 Υλικό (Hardware) του SunSPOT**

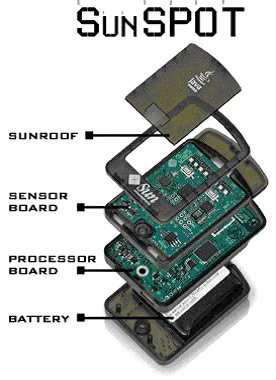
Μια πλήρης, ελευθέρου σήµατος (free range) συσκευή SunSPOT αποτελείται από έναν επεξεργαστή SunSPOT µε έναν πίνακα αισθητήρων και µία µπαταρία. Είναι συσκευασµένο σε ένα περίβληµα από πλαστικό. Ο µικρότερος σταθµός βάσης SunSPOT αποτελείται από µόλις έναν επεξεργαστή σε ένα πλαστικό περίβληµα.

-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.

37

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 12: Τα µέρη από τα οποία αποτελείται ένα SunSPOT: οθόνη, πίνακας**

**αισθητήρων, πίνακας επεξεργαστή, µπαταρία**

38

Αρχιτεκτονική του Πίνακα Επεξεργαστή (Processor board) του SunSPOT Επεξεργαστής

Κάθε SunSPOT διαθέτει έναν επεξεργαστή 400MHz 32-bit µε πυρήνα ARM920T µε 1Μ µνήµη RAM και 8Μ 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. H MMU (ΑRMv4) 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 Ι/Ο (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.

39



**Εικόνα 13: 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.

40

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.

41

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.

∆ίαυλοι επικοινωνίας του Main Board

* επικοινωνία µεταξύ των SPOTs και workstation γίνεται κυρίως µέσω του διαύλου USB και για την διασύνδεση υπάρχει µια υποδοχή mini USB τύπου B.
* 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. To 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. To 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

42

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

43

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. H παρακάτω εικόνα δείχνει τις µεταβάσεις που µπορεί να γίνουν µεταξύ των διαφορετικών καταστάσεων λειτουργίας.

Στοιχεία του Sensor Board (Πίνακας αισθητήρων)

To 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

44

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 SPΙ 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

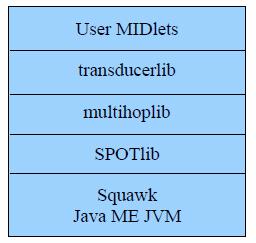
Iled theLed = Spot.getInstance().getGreenLed();

και για τον χειρισµό του LED: theLed.setOn(); theLed.setOff();

45

**4.2 Αρχιτεκτονική των SunSPOTs**

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

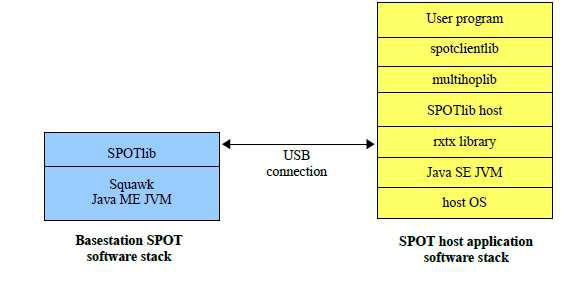


**Εικόνα 14: Επίπεδα λογισµικού ενός free-range SPOT**

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. ∆εν 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.

46



**Εικόνα 15: Αρχιτεκτονική SPOT host application**

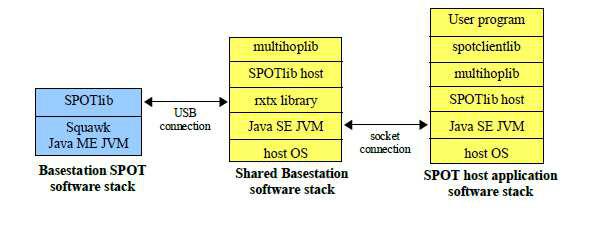
Αρχιτεκτονική host-application SunSPOT

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

47

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.



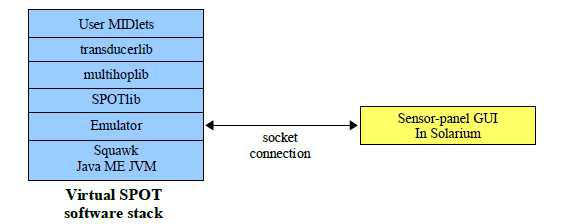
**Εικόνα 16: Αρχιτεκτονική SPOT-basestation**

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.

48



**Εικόνα 17: 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.

49

**4.3 Λογισµικό**

∆εν 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. Επεκτείνoντάς 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:

* O 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.

50

* + library suite – που περιλαµβάνει την βιβλιοθήκη µε τις κλάσεις σχετικές µε το SUN SPOT.
* 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:

51

* 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 {

}

52

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 o 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

53

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>").

54

**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**

H 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. H 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

55

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>

");

To <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:

Program 1 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();

56

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

57

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**

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

58

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

RadiogramConnection conn =

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

H <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>");

To <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://0014.4F01.0000.0006:1 00");

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");

59

} 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. ∆εν 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

60

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

61

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/”.

62

**4.5 Ανάπτυξη Πρότυπης Εφαρµογής**

∆ηµιουργία των κατάλληλων φακέλων του project

Αν έχουµε εγκαταστήσει το NetBeans IDE µπορούµε να δηµιουργήσουµε µια νέα εφαρµογή για τα SUN SPOT από το µενού File -> New Project και επιλέγοντας “Sun SPOT Application” από την κατηγορία “General”. Το

NetBeans θα δηµιουργήσει τους κατάλληλους φακέλους και αρχεία , και ένα βασικό MIDlet για να ξεκινήσουµε. Αν δεν έχουµε NetBeans µπορούµε να χρησιµοποιήσουµε σαν πρότυπο (template) τον κώδικα που µας δίνεται από το SUN SPOT SDK στον φάκελο

“C:\Sun\SunSPOT\Demos\CodeSamples\SunSpotApplicationTemplate”,

αντιγράφοντας τον κατάλογο αυτό σε µια άλλη θέση που θα αναπτύξουµε την εφαρµογή µας. Αν επιλέξουµε την δεύτερη επιλογή, θα πρέπει να αλλάξουµε στο αρχείο MANIFEST.MF τις τιµές που σχετίζονται µε το όνοµα της εφαρµογής µας.

Στη συνέχεια, χρησιµοποιώντας σαν πρότυπο τον κώδικα του SunSpotApplicationTemplate, ο χρήστης µπορεί να το επεκτείνει γράφοντας τον δικό του κώδικα σε Java.

Εάν ο χρήστης επιλέξει να χρησιµοποιήσει τη γραµµή εντολών αντί της πλατφόρµας ανάπτυξης του κώδικα, η τυπική διαδικασία δηµιουργίας και εκτέλεσης της εφαρµογής περιλαµβάνει τα εξής στάδια:

1. ∆ηµιουργία ενός καταλόγου για την αποθήκευση της εφαρµογής
2. Σύνταξη του κώδικα
3. Χρησιµοποίηση του παρεχόµενου template για σωστή σύνταξη του κώδικα της και δηµιουργίας ενός µεµονωµένου έτοιµου προς φόρτωση και εκτέλεση αρχείου από τα επιµέρους αρχεία.
4. Χρησιµοποίηση του παρεχόµενου template για τη φόρτωση και εκτέλεση της εφαρµογής.

Παρακάτω περιγράφουµε αναλυτικά τη διαδικασία για κάθε βήµα:

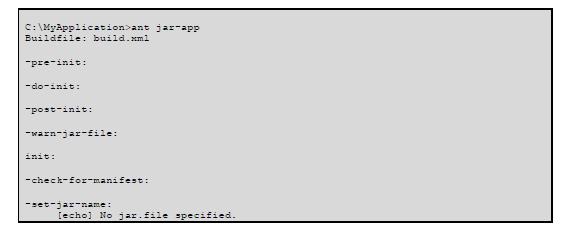
1. Ο κατάλογος Demos/CodeSamples/SunSpotApplicationTemplate περιέχει µία πολύ απλή εφαρµογή που µπορεί να χρησιµοποιηθεί σαν πρότυπο για τη συγγραφή του δικού µας κώδικα. Αντιγράφουµε τα περιεχόµενα αυτού του καταλόγου στον κατάλογο που επιθυµούµε να δουλέψουµε, τον οποίο

63

αποκαλούµε κατάλογο root. Οποιοσδήποτε κατάλογος µπορεί να χρησιµοποιηθεί σαν κατάλογος root. Όλοι οι root κατάλογοι των εφαρµογών έχουν την ίδια µορφή. Ο αρχικός κατάλογος περιέχει δύο αρχεία το build.xml και build.properties που ελέγχουν το ant script που χρησιµοποιείται για την κατασκευή (build) και εκτέλεση (run) των εφαρµογών. Επίσης, ο κατάλογος περιέχει 3 υποκαταλόγους. Ο πρώτος λέγεται src, είναι ο κατάλογος-ρίζα γι'αυτή την εφαρµογή. Ο δεύτερος, που λέγεται nbproject περιέχει αρχεία του project αν η πλατφόρµα που χρησιµοποιείται είναι το Netbeans. Ο τρίτος λέγεται resources, περιέχει το manifest file που προσδιορίζει την εφαρµογή καθώς και ο,τι άλλο αρχείο τύπου χρειάζεται η εφαρµογή κατά την εκτέλεσή της. 'Αλλοι κατάλογοι θα εµφανιστούν κατά τη διαδικασία του build.

2. Κάνουµε compile το πρότυπο παράδειγµα και δηµιουργούµε ένα αρχείο jar που το περιέχει µε την εντολή ant jar-app . Το jar δηµιουργείται στον φάκελο και ονοµάζεται ανάλογα µε τις τιµές των δύο ιδιοτήτων του αρχείου manifest.

Το όνοµα θα είναι της µορφής <MIDlet-Name>\_<MIDlet-Version>.jar



64



3.Συνδέουµε το Sun SPOT στον υπολογιστή χρησιµοποιώντας ένα καλώδιο

mini-USB.

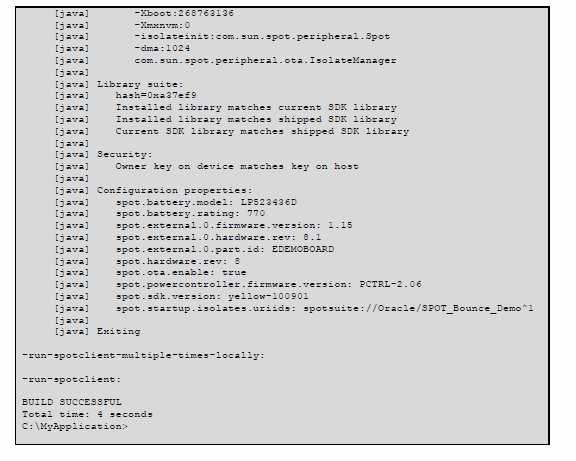
4. Ελέγχουµε την επικοινωνία µε το Sun SPOT χρησιµοποιώντας την εντολή ant info που δείχνει πληροφορίες για τη συσκευή.



65



66



5. Για να φορτώσουµε την πρότυπη εφαρµογή πληκτρολογούµε την εντολή

ant jar-deploy.

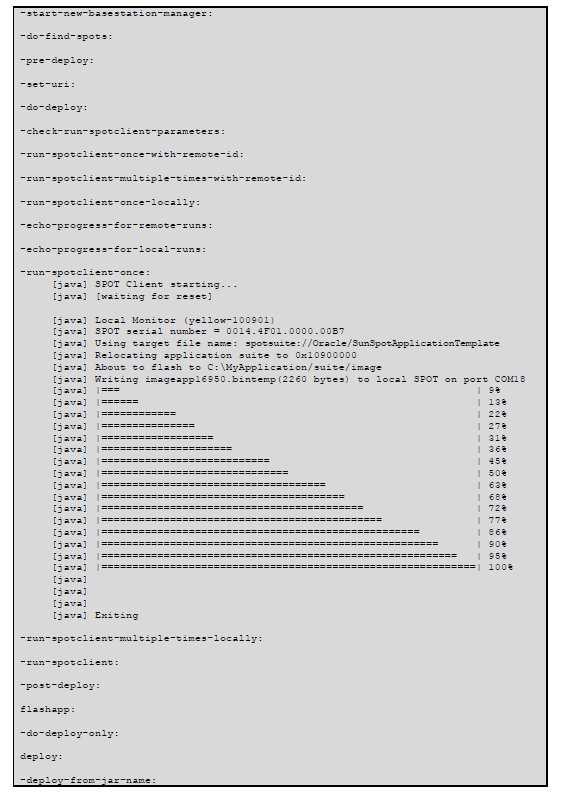
67



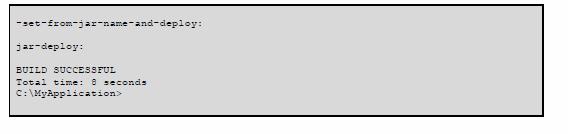
68



69



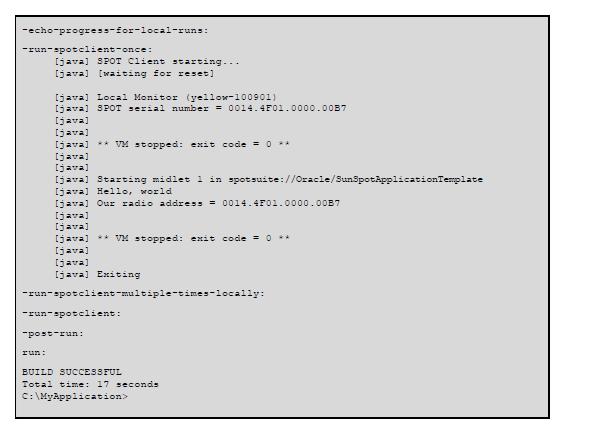
70



6. Τρέχουµε την εφαρµογή , µε την εντολή ant run.



71



Τέλος, χρησιµοποιώντας το ant script µπορούµε να συντάξουµε σωστά τον κώδικα και συνδυάζοντας όλα τα αρχεία κώδικα να δηµιουργήσουµε ένα αρχείο που µπορεί να ενσωµατωθεί/φορτωθεί σε ένα SunSPOT. Χρησιµοποιώντας αυτό το αρχείο θα το φορτώσoυµε και θα το τρέξουµε.

Ανάπτυξη εφαρµογής σε εικονικό SPOT

Το εργαλείο SPOTManager tool , στην τωρινή του µορφή διαθέτει τις εξής οκτώ επιλογές:

Sun SPOTs: επιτρεπει να αναζητήσουµε µεµονωµένα, να φορτώσουµε και να διαµορφώσουµε το λογισµικό τους.

SDKs: επιτρέπει να φορτώσουµε και να εγκαταστήσουµε τις διαθέσιµες εκδόσεις του SunSPOT SDK από τον ιστότοπο του SunSPOT.

72

Solarium: Παρέχει ένα εργαλείο για τη διαχείριση µεµονωµένων SunSPOTs και την εγκατάσταση λογισµικού σε αυτά. Επίσης διαθέτει έναν προσοµοιωτή SunSPOT για τον έλεγχο του λογισµικού µιάς εφαρµογής SunSPOT και την εκτέλεσή της.

Tutorial: Επιτρέπει να διαβάσουµε το εγχειρίδιο του

Docs: επιτρέπει την πλοήγηση στις οδηγίες για τα ενεργά SDK

Console: δείχνει την στάνταρτ έξοδο της χρήσης των εργαλείων του SPOTManager. Αν υπάρχει κάποιο νέο σφάλµα εκτέλεσης, το αποτέλεσµα είναι κόκκινο, αλλίώς είναι µαύρο.

Share: επιτρέπει την ανάγνωση και αξιολόγηση κοµµατιών κώδικα καθώς και τη δηµοσίευση δικών µας κοµµατιών κώδικα

Preferences: ∆ιαµορφώνει το δίκτυο και τα χαρακτηριστικά του ίδιου του SPOTManager tool. ∆ιαθέτει επίσης τα εξής τρία κουµπιά στο κάτω µέρος του παραθύρου, ανεξαρτήτως της επιλογής µας:

Support: Επιτρέπει να αναφέρουµε κάποιο σφάλµα στα SunSPOTs στο λογισµικό του SunSPOT ή στον SPOTManager. Εάν επιθυµούµε, συλλέγει και περικλείει πληροφορίες για τα συνδεδεµένα SunSPOTs την ώρα της αναφοράς.

Quit: Απενεργοποίηση του εργαλείου

Forums: Ανοίγει ένα πλοηγό του διαδικτύου στην ιστοσελίδα του επίσηµου φόρουµ του SunSPOT www.sunspotworld.com/forums/.

73



**Εικόνα 18: Το εργαλείο SPOTManager tool**

Επιλογή SunSPOTs:

Η επιλογή Sun SPOTS έχει ένα µενού για να επιλέξουµε µεταξύ των συνδεδεµένων µε USB Sun SPOTs, µία µεγάλης έκτασης περιοχή για έξοδο κειµένου και οκτώ κουµπιά συγκεκριµένα για αυτή την επιλογή.

Το µενού επάνω από την περιοχή της κονσόλας ονοµάζεται: “Select a Sun SPOT” και εµφανίζει µία λίστα όλων των συνδεδεµένων SunSPOTs µέσω USB στο σταθµό εργασίας, το καθένα µε τον IEEE αριθµό δικτύου του. Τα οκτώ πρώτα ψηφία του είναι πάντα 0014.4F01, ενώ τα τελευταία οκτώ ψηφία είναι γραµµένα σε ένα αυτοκόλλητο ορατό µέσω του πλαστικού περιβλήµατος της κεραίας του SunSPOT.

Τα οκτώ κουµπιά που διαθέτει το είναι τα εξής:

Upgrade: Εφαρµόζει την εντολή ant upgrade στο επιλεγµένο SunSPOT φορτώνοντας το µε το λογισµικό που απαιτείται για την τρέχουσα έκδοση του

SDK.

74

Properties: ∆είχνει τις τρέχουσες ιδιότητες του συστήµατος του επιλεγµένου SunSPOT και επιτρέπει την αλλαγή τους.

SPOT Info: Τρέχει την εντολή ant info στο επιλεγµένο SunSPOT και δείχνει τις πληροφορίες που αφορούν το λογισµικό του στην περιοχή της εξόδου. Echo: Εµφανίζει τα αποτελέσµατα της εκτέλεσης του επιλεγµένου SunSPOT. Basestation: Εµφανίζει ένα µενού όπου µπορεί ο χρήστης να ενεργοποιήσει ή να απενεργοποιήσει το επιλεγµένο SPOT ως σταθµό βάσης basestation. Αυτό κατά κανόνα γίνεται σε SunSPOT που δεν έχει µπαταρία ή πίνακα αισθητήρων. Υπάρχει η δυνατότητα επιλογής απλού 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: Ενεργοποιεί η 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, εκτελέσουµε και εκσφαλµατώσουµε εφαρµογές σε ένα Sun SPOT χρησιµοποιώντας τη ραδιοσυχνότητα µεταξύ του basestation και του επιλεγµένου Sun SPOT. Σηµειωτέον ότι το OTA command server θα πρέπει να ενεργοποιείται µόνο σε free-range SPOTs.

Restore: εκτελεί την εντολή ant restore στο επιλεγµένο Sun SPOT επαναφέροντάς το στις αρχικές εργοαστασιακές του SDK ρυθµίσεις.

Clear: ∆ιαγράφει όλο το κείµενο από το παράθυρο εξόδου.

Επιλογή Solarium

Η επιλογή Solarium επιτρέπει να εκκινησουµε µια εφαρµογή Solarium, δηλαδή µία Java εφαρµογή που χρησιµοποιείται για να εκτελέσει µια σειρά από ενέργειες στα Sun SPOTs. Έχει τις παρακάτω δυνατότητες:

Ανακάλυψη και εµφάνιση SPOTs: Ανακαλύπτει και εµφανίζει Sun SPOTs που είναι συνδεδεµένα στον υπολογιστή µέσω usb ή ασύρµατης επικοινωνίας. Αλληλεπίδραση µε τα SPOTs: Μπορεί να φορτώσει και να ξεφορτώσει λογισµικό από τα Sun SPOTs, να εκκινήσει, διακόψει, επανεκκινήσει και τερµατίσει εφαρµογές και να ανιχνεύσει την κατάσταση µίας συσκευής. Το πλαίσιο Radio View παρέχει έναν γραφικό τρόπο αναπαράστασης της ασύρµατης συνδεσιµότητας των Sun SPOTs.

75

∆ιαχείριση ενός δικτύου από SPOTs: Μπορεί να ελέγξει ένα δίκτυο από Sun SPOTs µε χρήση του πλαισίου Deployment View. Το πλαίσιο αυτό καθορίζει ποια εφαρµογή πρέπει να φορτωθεί σε ποια συσκευή και διευκολύνει τη διαδικασία φόρτωσης τους.

Προσοµοίωση SPOTs: Περιλαµβάνει έναν προσοµοιωτή, ο οποίος µπορεί 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. Ένα basestation δεν είναι απαραίτητο αν θέλουµε να επικοινωνήσουµε µόνο µε προσοµοιωµένα SPOTs. Ένα διαµοιραζόµενο basestation απαιτείται αν θέλουµε προσοµοιωµένα SPOTs να επικοινωνήσουνε ασύρµατα µε πραγµατικά SPOTs. Αν εκκινήσουµε το Solarium µε το κουµπί µε την ένδειξη Solarium ο SPOTManager εµφανίζει ένα κουτί διαλόγου όπου πρέπει να ορίσουµε ποιό από τα συνδεδεµένα µε USB SPOTs θέλουµε να είναι ο basestation.



**Εικόνα 19: Το εργαλείο Solarium**

76

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

Aνακάλυψη και εµφάνιση των SPOTs

To Solarium µπορει να εντοπισει και να εµφανίσει τα SPOTs που είναι συνδεδεµένα στην επιφάνεια εργασίας µέσω USB καθώς και τα κοντινά SPOTs που ειναι συνδεδεµένα ασύρµατα.

Προκειµένου να είναι ορατά µέσω ασύρµατης επικοινωνίας, τα SPOTs θα πρέπει να έχουν ενεργοποιηµένο το ΟΤΑ (OTA command server) τους, το οποίο κανονικά είναι ενεργοποιηµένο από µόνο του. Σε περίπτωση που κάτι τέτοιο δεν ισχύει, ο χρήστης µπορεί 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 περιλαµβάνει και κάποιες άλλες 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 των SPOTs

To Solarium περιλαµβάνει έναν προσοµοιωτή που δίνει τη δυνατότητα να εκτελεί εφαρµογές για SunSPOT στον υπολογιστή µας. Αυτό µας επιτρέπει να ελέγχουµε την ορθότητα ενός προγράµµατος προτού το φορτώσουµε σε ένα SPOT ή να προσοµοιώσουµε ένα SPOT αν δεν διαθέτουµε ένα. Αντί για πίνακα αισθητήρων, τα εικονικά SPOT διαθέτουν µία οθόνη αισθητήρων, που µπορεί να χρησιµοποιηθεί για να εισαχθούν οι επιδιωκώµενες τιµές των αισθητήρων (π.χ. επίπεδο φωτός, θερµοκρασία, ψηφιακές είσοδοι, αναλογικές τιµές ρεύµατος εισόδου, και τιµές του επιταχυνσιόµετρου). Η εφαρµογή µπορεί να ελέγξει το χρώµα των LEDs, όπως θα έκανε ένα πραγµατικό SPOT. Πατώντας στα κουµπιά των διακοπτών στην εικόνα µπορεί κανείς να αναβοσβήσει τους διακόπτες. Υποστηρίζεται επίσης η αποστολή και παραλαβή µηνυµάτων µέσω ραδιοσυχνότητας, καθώς κάθε

77

SPOT διαθέτει δική του διεύθυνση και µπορεί να µεταδώσει σε ένα ή περισσότερα SPOT πληροφορίες.

∆ιαχείριση των virtual SPOTs

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

Set Name: µπορούµε να δώσουµε ένα όνοµα στο virtual SPOT για να εµφανίζεται απο κάτω µαζί µε την ΙΕΕΕ 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 : Αφότου έχουµε 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. Κάνοντας κλικ επάνω του εµφανίζεται η επιλογή να διακόψουµε την εκτέλεσή του.

Debug MIDlet : εµφανίζει όλα τα MIDlets που περιέχονται στο αρχείο µε το φορτωµένο jar και επιτρέπει να συνδέσουµε έναν εξωτερικό Java εκσφαλµατωτή.

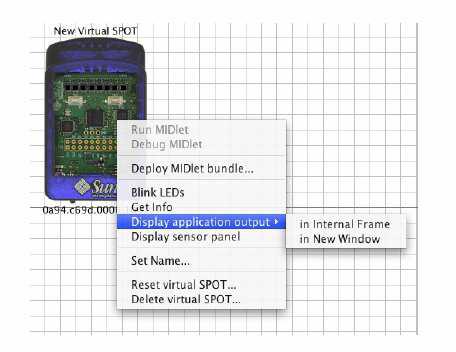
Reset virtual SPOT: σταµατάει όλα τα MIDlets που εκτελούνται και επανεκκινεί την Squawk VM. Μπορούµε έτσι να επαναφορτώσουµε οποιοδήποτε αρχείο jar έχει δηµιουργηθεί νωρίτερα και να εκτελέσουµε όποιο από τα MIDlets που περιέχει επιθυµούµε.

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 µε πληροφορίες για το virtual SPOT: την ΙΕΕΕ διεύθυνσή του, το αρχείο jar που έχει φορτωθεί και τα ονόµατα των διαθέσιµων MIDlets.

Delete virtual SPOT : διαγράφει ένα virtual SPOT.

78



**Εικόνα 20: Επιλογές ∆ιαχείρησης ενός Virtual SPOT**

Από την κύρια γραµµή εργαλείων υπάρχει επίσης η δυνατότητα να αποθηκεύσει κανείς την κατάσταση στην οποία βρίσκονται όλα τα virtual

SPOTs που δηµιούργησε πηγαίνοντας Emulator->Save virtual configuration... .Έτσι, 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 να επιλέξουµε το αρχείο που σώσαµε προηγουµένως και να το επαναφορτώσουµε. Ταυτόχρονα θα εµφανιστεί και το κείµενο περιγραφής που είχαµε σώσει, το οποίο µπορεί να επανεµφανιστεί µε την επιλογή Display virtual configuration description.

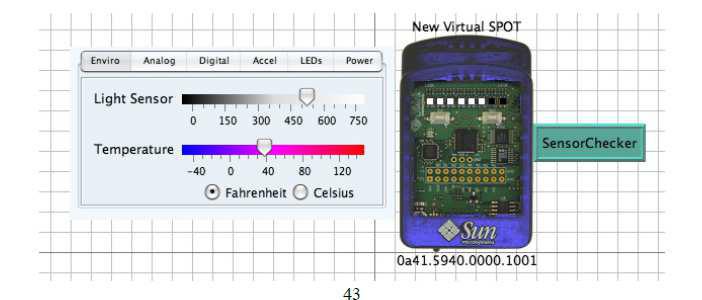
Τέλος, µε την εντολή Delete all virtual SPOTs... µπορούµε να διαγράψουµε όλα τα virtual SPOTs που έχουµε καθορίσει στο Solarium.

79

Οθόνη πίνακα αισθητήρων

Τα Virtual SPOTs αντί για πίνακα αισθητήρων που έχουν τα πραγµατικά SPOTs, διαθέτουν µία οθόνη πίνακα αισθητήρων ως εναλλακτικό µηχανισµό για να παίρνουν τιµές οι αισθητήρες. Έτσι, επιλέγοντας το εµφανίζεται ένα παράθυρο µε έξι επιλογές: Enviro, Analog In, Accel, Digital Pins, LEDs.

Καθεµία από τις οποίες αντιστοιχεί σε εισόδους των αισθητήρων του virtual SPOT και µας επιτρέπει να δώσουµε τις ανάλογες τιµές.



**Εικόνα 21: Οθόνη πίνακα αισθητήρων**

Χρησιµοποιώντας τη ραδιοσυχνότητα

Τα Virtual SPOTs µπορούν 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

µπορούνα να επικοινωνούν και µε πραγµατικά χρησιµοποιώντας τη ραδιοσυχνότητα του basestation. Το πλεονέκτηµα είναι ότι πολλές host applications µπορόυν να έχουν πρόσβαση στη ραδιοσυχνότητα. Το µειονέκτηµα είναι 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

80

σταθµός βάσης (dedicated basestation). Ένα άλλο µειονέκτηµα είναι ότι δεν είναι δυνατή η διαχείριση της ραδιοκεραίας του basestation SPOT, του pan id ή της ισχύος της εξόδου κατά την εκτέλεση. Για να χρησιµοποιούµε πάντα shared basestation αρκεί να προσθέσουµε την ακόλουθη γραµµή στο αρχείο .sunspot.properties:

basestation.shared=true

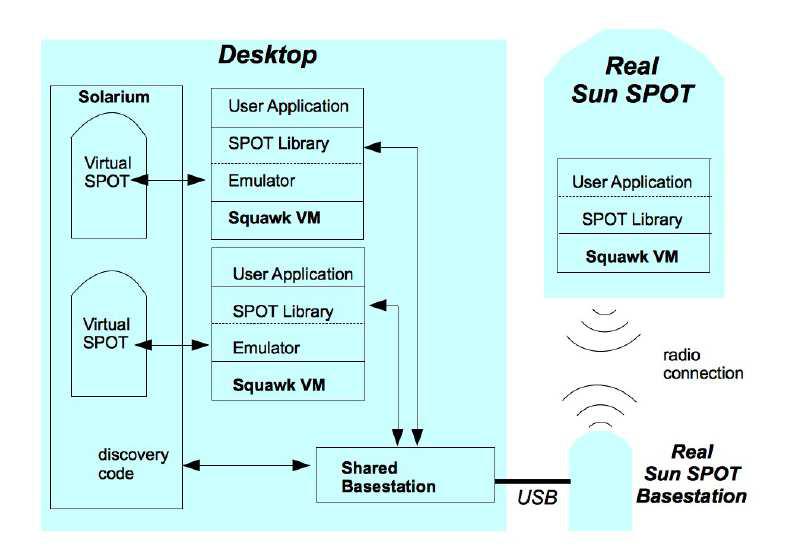
81

Τρόπος λειτουργίας του προσωµοιωτή

Όταν δηµιουργούµε ένα νέο virtual SPOT στο Solarium, µία νέα διδικασία ξεκινά 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. Για παράδειγµα, 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. Οµοίως, όταν ο χρήστης κάνει κλικ µε το ποντίκι έναν από τους διακόπτες του virtual SPOT, το Solarium στέλνει ένα µήνυµα στον κώδικα του προσοµοιωτή ότι ο διακόπτης έχει πατηθεί, το οποίο µετά µπορεί να παρατηρηθεί από την εφαρµογή του SPOT.

Ακολουθεί το διάγραµµα της αρχιτεκτονικής του προσοµοιωτή:



**Εικόνα 22: Αρχιτεκτονική ενός προσοµοιωτή**

82

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 ως µέρος της βιβλιοθήκης του SPOT, που επιτρέπει 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).