ΟΛΟΚΛΗΡΩΜΕΝΟ ΣΥΣΤΗΜΑ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΠΕΡΙΒΑΛΛΟΝΤΙΚΩΝ ΣΥΝΘΗΚΩΝ ΚΑΙ ΤΗΛΕ-ΕΙΔΟΠΟΙΗΣΗΣ ΣΕ ΓΕΩΡΓΙΚΕΣ ΕΦΑΡΜΟΓΕΣ

Σπηλιόπουλος Χρήστος

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ : Καθηγητής Γεώργιος Σταμούλης
2ος ΒΑΘΜΟΛΟΓΗΤΗΣ : Δρ. Παναγιώτης Κίκιρας

Περίληψη

Η εργασία αυτή έχει ως σκοπό την παρουσίαση ενός ολοκληρωμένου συστήματος με δυνατότητες παρακολούθησης περιβαλλοντικών συνθηκών και τηλεειδοποίησης σε γεωργικές εφαρμογές. Αναλύει τα επιμέρους στοιχεία που απαρτίζουν το σύστημα και το πως αυτά αλληλεπιδρούν προγραμματιστικά. Περιέχει αναλυτικές οδηγίες για το πως γίνεται η εγκατάσταση του συστήματος από την αρχή και η επικοινωνία του με τον χρήστη μέσω κινητού τηλεφώνου.

Abstract

The purpose of this paper is to demonstrate a complete system that has capabilities of environmental conditions observation and sms notification on agricultural appliances. It analyzes the several components that the system is consisted of and how they do interact programmatically. It contains detailed instructions on how to install such a system from scratch and how to communicate with it via a mobile phone.

Πίνακας περιεχομένων

1.
Εισαγωγή

1.1 Αντικείμενο της εργασίας

Η εργασία αυτή έχει ως σκοπό την δημιουργία ενός συστήματος το οποίο θα καταγράφει τις περιβαλλοντικές συνθήκες που επικρατούν σε κάποιο χώρο ανά πάσα στιγμή. Οι συνθήκες που ενδιαφέρουν την περίπτωσή μας (γεωργικές εφαρμογές) είναι η θερμοκρασία, η υγρασία και η ακτινοβολία. Το σύστημά μας έχει την δυνατότητα να ειδοποιεί μέσω sms τον ενδιαφερόμενο για τις συνθήκες αυτές αλλά υπάρχει ακόμα η δυνατότητα και το σύστημα να ειδοποιηθεί μέσω sms και να αλλάξει κάποιες παραμέτρους του και άρα την λειτουργικότητά του. Ένα τέτοιο σύστημα απαρτίζεται από έναν κεντρικό κόμβο-υπολογιστή ο οποίος λειτουργεί ως σταθμός βάσης συλλέγοντας μετρήσεις από ένα δίκτυο αισθητήρων και μέσω ενός GSM modem στέλνει και δέχεται μηνύματα. Ως εκ τούτου κρίνεται ιδανικό για γεωργικές εφαρμογές τύπου θερμοκηπίων, μικρής ή μεγάλης κλίμακας. Εμείς, για χάρη απλότητας, θα ασχοληθούμε με σύστημα ενός αισθητήρα και όχι με δίκτυο αισθητήρων. Περαιτέρω κλιμάκωση-αναβάθμιση είναι εύκολο να γίνει όπως θα αναφέρουμε και αργότερα, με βάση τις εγγενείς ιδιότητες τέτοιου τύπου δικτύων.

1.2 Επισκόπηση κεφαλαίων

Τα περιεχόμενα των κεφαλαίων της εργασίας περιγράφονται εδώ συνοπτικά.

Το πρώτο κεφάλαιο παρουσιάζει το αντικείμενο της εργασίας περιληπτικά.

Στο δεύτερο κεφάλαιο παρουσιάζονται τα επιμέρους τμήματα σε επίπεδο υλικού που απαρτίζουν το σύστημά μας και αναλύονται τα χαρακτηριστικά τους, το λειτουργικό σύστημα το οποίο χρησιμοποιούν και τα υπόλοιπα προγραμματιστικά εργαλεία που είναι απαραίτητα για την επίτευξη της λειτουργικότητάς μας.

Το τρίτο κεφάλαιο δείχνει την δομή της εφαρμογής (αρχιτεκτονική) καθώς και διαγράμματα λειτουργίας και καταστάσεων.

Το τέταρτο κεφάλαιο ουσιαστικά είναι ένα εγχειρίδιο εγκατάστασης μιας συγκεκριμένης Linux διανομής στο σύστημά μας, καθώς αυτή γίνεται μόνο μέσω δικτύου. Επίσης υπάρχει αναλυτικός οδηγός εγκατάστασης όλων των προγραμματιστικών εργαλείων που παρουσιάζονται στο δεύτερο κεφάλαιο.

Το πέμπτο κεφάλαιο αναλύει τον κώδικα με τον οποίο οι αισθητήρες επικοινωνούν με τον σταθμό βάσης και με την βάση δεδομένων.

Το έκτο κεφάλαιο αναλύει τον κώδικα με τον οποίο στέλνονται και λαμβάνονται τα sms μέσω του modem.

Το έβδομο κεφάλαιο περιγράφει τα υπόλοιπα κομμάτια κώδικα που απαρτίζουν την εφαρμογή.

Το όγδοο κεφάλαιο περιγράφει έναν οδηγό χρήσης αφού το σύστημα έχει εγκατασταθεί και είναι έτοιμο για λειτουργία.

Στο ένατο κεφάλαιο γίνεται μια αποτίμηση του αποτελέσματος καθώς και αναφορές σε πιθανές μελλοντικές προσθήκες και επεκτάσεις.

Στο δέκατο και τελευταίο κεφάλαιο βρίσκεται η βιβλιογραφία από την οποία αντλήθηκε γνώση για την υλοποίηση του συστήματος.

Τέλος, στο ενδέκατο κεφάλαιο παραθέτουμε τον πηγαίο κώδικα της εφαρμογής.

2.
Υλικό

2.1 Κεντρικός κόμβος

Ως κεντρικό κόμβο ορίζουμε το βασικό “μηχάνημα” πάνω στο οποίο θα χτιστεί το σύστημά μας. Πρόκειται ουσιαστικά για έναν υπολογιστή μικρών επιδόσεων, που όμως είναι υπεραρκετός για εφαρμογές που έχουν ελάχιστες απαιτήσεις από άποψη τεχνικών χαρακτηριστικών, όπως Firewalls, VPN routers, Internet Gateways κ.α. Για τον σκοπό μας θα χρησιμοποιήσουμε το net5501 της εταιρίας SOEKRIS, το οποίο βλέπουμε παρακάτω στις εικόνες 2.1.1. και 2.1.2.

εικόνα 2.2.1

Τα χαρακτηριστικά του net5501 είναι τα εξής :

- 500 Mhz CPU,
- 512 Mbyte DDR-SDRAM
- 4 Ethernet ports
- 2 Serial ports
- USB connector
- CF socket
- 44 pins IDE connector
- SATA connector
- 1 Mini-PCI socket
- 3.3V PCI connector.

εικόνα 2.1.2

Το net5501 είναι υπεραρκετό για να του εγκαταστήσουμε μια διανομή Linux με τις απαραίτητες εφαρμογές και να τρέχει απρόσκοπτα με πολύ χαμηλή κατανάλωση ενέργειας. Το net5501 δεν έχει γραφική υποστήριξη και δεν κάνει boot από usb, οπότε θα καταφύγουμε στα άδυτα της κονσόλας και στην εγκατάσταση λειτουργικού μέσω δικτύου. Αναλυτικές οδηγίες παραθέτουμε στο κεφάλαιο 3.

2.2 Modem

Το modem είναι το στοιχείο εκείνο που δίνει νόημα στον όρο τηλε-ειδοποίηση που χαρακτηρίζει το σύστημά μας. Μέσω αυτού στέλνονται και λαμβάνονται τα sms που είναι απαραίτητα για την λειτουργικότητά του.

Στην εργασία χρησιμοποιήσαμε ένα USB modem και συγκεκριμένα το ModemUSB G10 της εταιρίας TELTONIKA (εικ. 2.2.1).

Το συγκεκριμένο modem είναι σχεδιασμένο για μεταφορά δεδομένων μέσω GSM δικτύων. Συνδέεται στον υπολογιστή μέσω μιας USB θύρας και έχει υποδοχή για μια κάρτα SIM κινητών τηλεφώνων. Από την στιγμή που έχει συνδεθεί με τον υπολογιστή έχει την δυνατότητα να στείλει sms ή να συνδεθεί στο διαδίκτυο μέσω κάποιου προτύπου που υποστηρίζεται από το πρωτόκολλο GSM, όπως GPRS, CSD και SMS.

Ενώ υπάρχουν drivers για λειτουργικό σύστημα windows, καθώς και GUI για εύκολη πρόσβαση και χρησιμοποίηση των λειτουργιών, εμείς θα κάνουμε χρήση μιας Linux διανομής. Σε μια τέτοια διανομή που έχει πυρήνα (kernel) έκδοσης 2.6.9 και πάνω (την στιγμή που γράφονται αυτές οι γραμμές έχουμε φτάσει αισίως στην έκδοση 2.6.31) υποστηρίζονται οι λεγόμενοι FTDI serial converter drivers καθώς το modem μας τους χρησιμοποιεί για να λειτουργήσει.

Από την στιγμή που αναγνωριστεί το modem μας μπορούμε μέσω κάποιου εργαλείου (όπως το θρυλικό minicom) να λειτουργήσουμε το modem στέλνοντάς του AT εντολές (περισσότερα στο κεφάλαιο 6). Όμως εμείς θα προγραμματίσουμε το modem σε λίγο πιο χαμηλό και straight επίπεδο με χρήση της γλώσσας python και χρησιμοποιώντας το module python-serial. Το συγκεκριμένο module ενδείκνυται για επικοινωνία με σειριακές συσκευές και επειδή έχουμε υποστήριξη FTDI USB-to serial converter όλα λειτουργούν κανονικά και σύμφωνα με τις προδιαγραφές.

GUI για αλληλεπίδραση με τον χρήστη δεν μπορούμε εκ των πραγμάτων να έχουμε, καθώς το Soekris δεν υποστηρίζει γραφικό περιβάλλον. Αντί αυτού θα φτιάξουμε ένα υποτυπώδες μενού που θα φένεται μέσα στην κονσόλα του Linux.

Εξάλλου αυτό θα χρησιμοποιηθεί κατά την εγκατάσταση και αρχική παραμετροποίηση του συστήματος, καθώς μετέπειτα το σύστημα θα “τρέχει” μόνο του και οι όποιες αλλαγές θα γίνονται μέσω sms και πολύ σπάνια μέσα απο την κονσόλα τοπικά (π.χ. σε περίπτωση επαννεκίνησης ή αναβάθμισης...).

Εικ 2.2.1

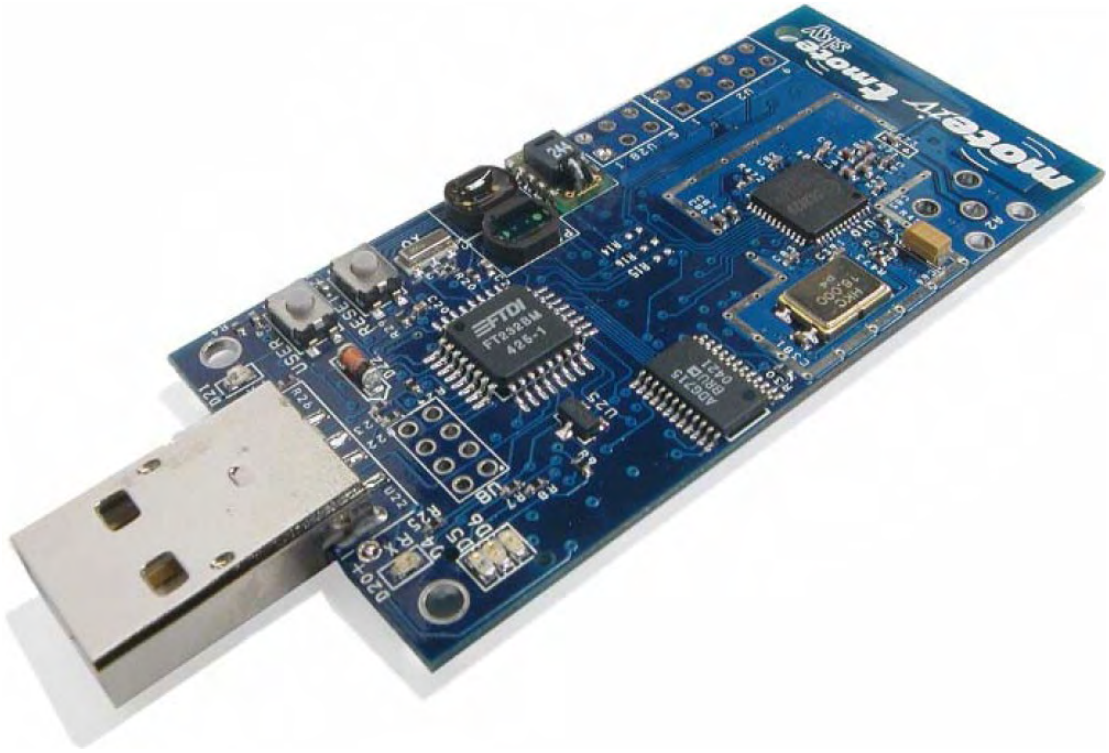
2.3 Αισθητήρες

Ο αισθητήρας μας αναλαμβάνει να πάρει τις μετρήσεις από το περιβάλλον και να τις προωθήσει στην εφαρμογή μας. Στην εργασία αυτή θα χρησιμοποιήσουμε τον αισθητήρα Tmote sky της εταιρίας Moteiv (εικόνα 2.3.1). Χαρακτηρίζεται για τις εξαιρετικά χαμηλές απαιτήσεις του σε ενέργεια και για την χρήση του σε δίκτυα αισθητήρων με υψηλές απαιτήσεις σε μετάδοση δεδομένων. Συμπεριλαμβάνει ολοκληρωμένους αισθητήρες για μέτρηση θερμοκρασίας, υγρασίας και ηλιακής ακτινοβολίας, κεραία, μικροελεγκτές και προγραμματιστικές δυνατότητες.

Υποστηρίζει πρότυπα όπως USB και IEEE 802.15.4 για εύκολη επικοινωνία με άλλες συσκευές.

Αναλυτικά τα χαρακτηριστικά του :

- 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver.
- Σύνδεση με άλλες IEEE 802.15.4 συσκευές.
- 8MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash)
- Ενσωματωμένο ADC, DAC, Supply Voltage Supervisor, and DMA Controller
- Ενσωματωμένη onboard κεραία με 50m εμβέλεια σε εσωτερικούς σώρους / 125m
  εμβέλεια σε εξωτερικούς.
- Ενσωματωμένα sensors για μέτρηση υγρασίας, θερμοκρασίας και φωτός.
- Εξαιρετικά χαμηλή κατανάλωση ρεύματος.
- Γρήγορο ξύπνημα από ύπνο (<6κs).
- Hardware link-layer κρυπτογράφηση και πιστοποίηση.
- Προγραμματισμός και συλλογή δεδομένων μέσω USB καλωδίου.
- 16-pin υποδοχή επέκτασης και δυνατότητα επιλογής SMA σύνδεσης κεραίας.
- TinyOS υποστήριξη: mesh networking and communication implementation.
- Συμμορφώνεται με το FCC Part 15 και τους κανονισμούς Βιομηχανίας του Καναδά.



Εικόνα 2.3.1

Όπως φαίνεται και από τα χαρακτηριστικά το λειτουργικό σύστημα ενός Tmote sky είναι το TinyOS. Το TinyOS είναι ένα ελεύθερο λειτουργικό σύστημα ανοιχτού κώδικα και βασίζεται σε συστατικά – components που χρησιμοποιείται ευρέως σε ασύρματα δίκτυα αισθητήρων. Ενσωματώνεται σε κάποιο άλλο λειτουργικό σύστημα και είναι γραμμένο σε nesC σαν ένα σύνολο συνεργαζόμενων διεργασιών. Ξεκίνησε σαν συνεργασία μεταξύ του πανεπιστημίου Berkeley και της Intel Research.

Τι είναι η nesc?

Η nesc είναι ακρωνύμιο του network embedded systems C. Βασίζεται σε components και σε ένα μοντέλο προγραμματισμού καθοδηγούμενο απο γεγονότα (event driven). Χρησιμοποιείται για το χτίσιμο εφαρμογών πάνω σε TinyOS πλατφόρμες. Ουσιαστικά είναι μια επέκταση της C γλώσσας προγραμματισμού που χρησιμοποιεί components συνεργαζόμενα μεταξύ τους για να τρέξουν οι εφαρμογές στο TinyOS.

2.4 Λογισμικό

2.4.1 Λειτουργικό

Το λειτουργικό σύστημα που θα χρησιμοποιήσουμε στην εργασία μας υπόκειται ουσιαστικά σε περιορισμούς από το υλικό μας. Το Soekris net5501 είναι σχεδιασμένο να τρέχει μόνο Unix-οειδή λειτουργικά, δηλαδή FreeBSD, NetBSD, OpenBSD, Debian GNU/Linux και Gentoo Linux. Όλα τα παραπάνω είναι προφανώς “ελαφριά” λειτουργικά συστήματα, χωρίς πολλές απαιτήσεις και επειδή επιπλέον δεν έχουμε γραφικό περιβάλλον πετυχαίνουμε εξοικονόμηση μνήμης για τις εφαρμογές μας.

Θα εγκαταστήσουμε την διανομή Debian 5.0.2 (με την κωδική ονομασία Lenny), μια πασίγνωστη διανομή που χαρακτηρίζεται για την αξιοπιστία, την σταθερότητα και την πληθώρα εφαρμογών και πακέτων που αναπτύσονται γι' αυτήν. Έχει χαρακτηριστεί ως Universal Operating System, στοχεύοντας σε ένα πλήθος συστημάτων, από servers και desktops μέχρι embedded μικροσυστήματα, υποστηρίζοντας ένα τεράστιο σύνολο αρχιτεκτονικών. Πολλές πασίγνωστες και μη διανομές έχουν ως βάση το Debian (Ubuntu, Mepis, Sidux κ.α.) Το πλεονέκτημα αυτής της διανομής είναι ότι έχει στα αποθετήριά της το TinyOS ως “πακέτο” έτοιμο για εγκατάσταση. Έτσι γλιτώνουμε κόπο και χρόνο, αποφεύγοντας μια “manual” εγκατάσταση του TinyOS (με rpms) που μπορεί να μας οδηγήσει σε διάφορα προβλήματα εγκατάστασης (εξ'άλλου συνιστάται και από τους χρήστες Tmote Sky σε Linux περιβάλλοντα, για αποφυγή προβλημάτων ακόμα και έπειτα από μια φαινομενικά επιτυχή εγκατάσταση).

2.4.2 Βάση δεδομένων

Βάση δεδομένων θα χρειαστούμε για την αποθήκευση των τιμών και άλλων πληροφοριών που προέρχονται από τις μετρήσεις των αισθητήρων. Θα χρησιμοποιήσουμε την πασίγνωστη MySQL (My Structured Query Language), η οποία είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Η MySQL τρέχει ως εξυπηρετητής που παρέχει πρόσβαση σε βάσεις δεδομένων από

πολλούς χρήστες ταυτόχρονα. Είναι λογισμικό ανοιχτού κώδικα υπό την άδεια GNU General Public License, και χρησιμοποιείται ευρέως λόγω υψηλής ευχρηστίας, αξιοπιστίας και απόδοσης, όντας η δημοφιλέστερη στον κόσμο.

2.4.3 Προγραμματιστικά εργαλεία

Για τον προγραμματισμό των αισθητήρων θα βασιστούμε στην έτοιμη πλατφόρμα του TinyOS που θα εγκαταστήσουμε μαζί με το λειτουργικό. Θα χρειαστούμε όμως και κώδικα για τον προγραμματισμό του modem καθώς και γενικά της εφαρμογής η οποία θα τρέχει απρόσκοπτα σε κάποιον ατέρμονα βρόχο και θα χρησιμοποιεί – συντονίζει τα υπόλοιπα προγραμματιστικά και μη κομμάτια του συστήματος. Γι' αυτόν τον σκοπό θα χρησιμοποιήσουμε την γλώσσα python, η οποία μας παρέχει όλα τα παραπάνω, εύκολα και με αποδοτικό τρόπο. Επίσης, επειδή το TinyOS συνεργάζεται με java εφαρμογές για την επικοινωνία των αισθητήρων με τον υπολογιστή, θα κάνουμε χρήση java και θα τροποποιήσουμε “ελαφρώς” κάποιο κομμάτι έτοιμου κώδικα για να πετύχουμε τον στόχο μας.

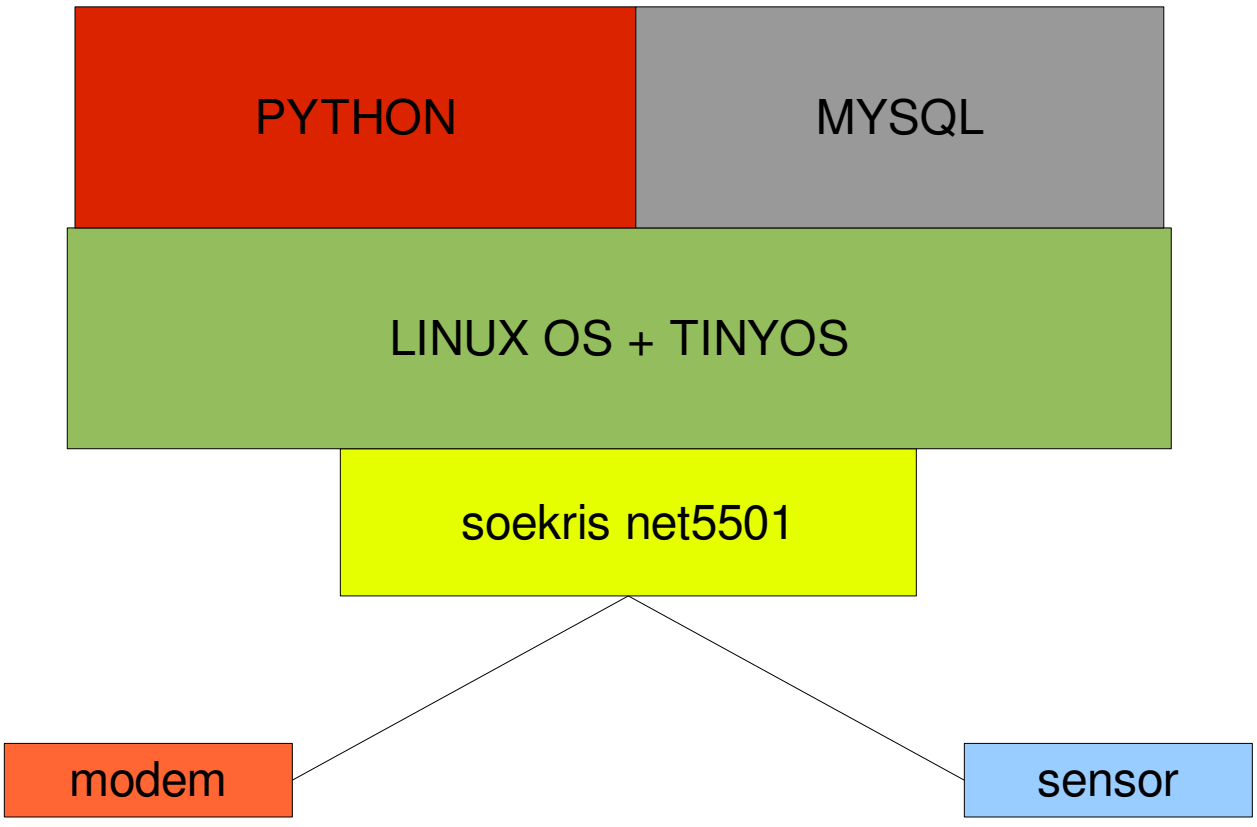
Γιατί Python?

Η python (εμπνευσμένη ετυμολογικά από τους Monty Pythons) είναι μια γενικού σκοπού γλώσσα προγραμματισμού υψηλού επιπέδου. Η σχεδιαστική της φιλοσοφία δίνει έμφαση στην εύκολη ανάγνωση, οργάνωση, αποσφαλμάτωση και διατήρηση του κώδικα. Χαρακτηρίζεται επίσης από τις πανίσχυρες δυνατότητες της σε οποιδήποτε πεδίο εφαρμογής και μάλιστα με σημαντική εξοικονόμηση γραμμών κώδικα. Ενδεικτικό παράδειγμα η υιοθέτηση και χρησιμοποίησή της από την NASA.

3.
Αρχιτεκτονική και διαγράμματα

3.1 Αρχιτεκτονική

Μια γενική αρχιτεκτονική του συστήματός μας φαίνεται παρακάτω :

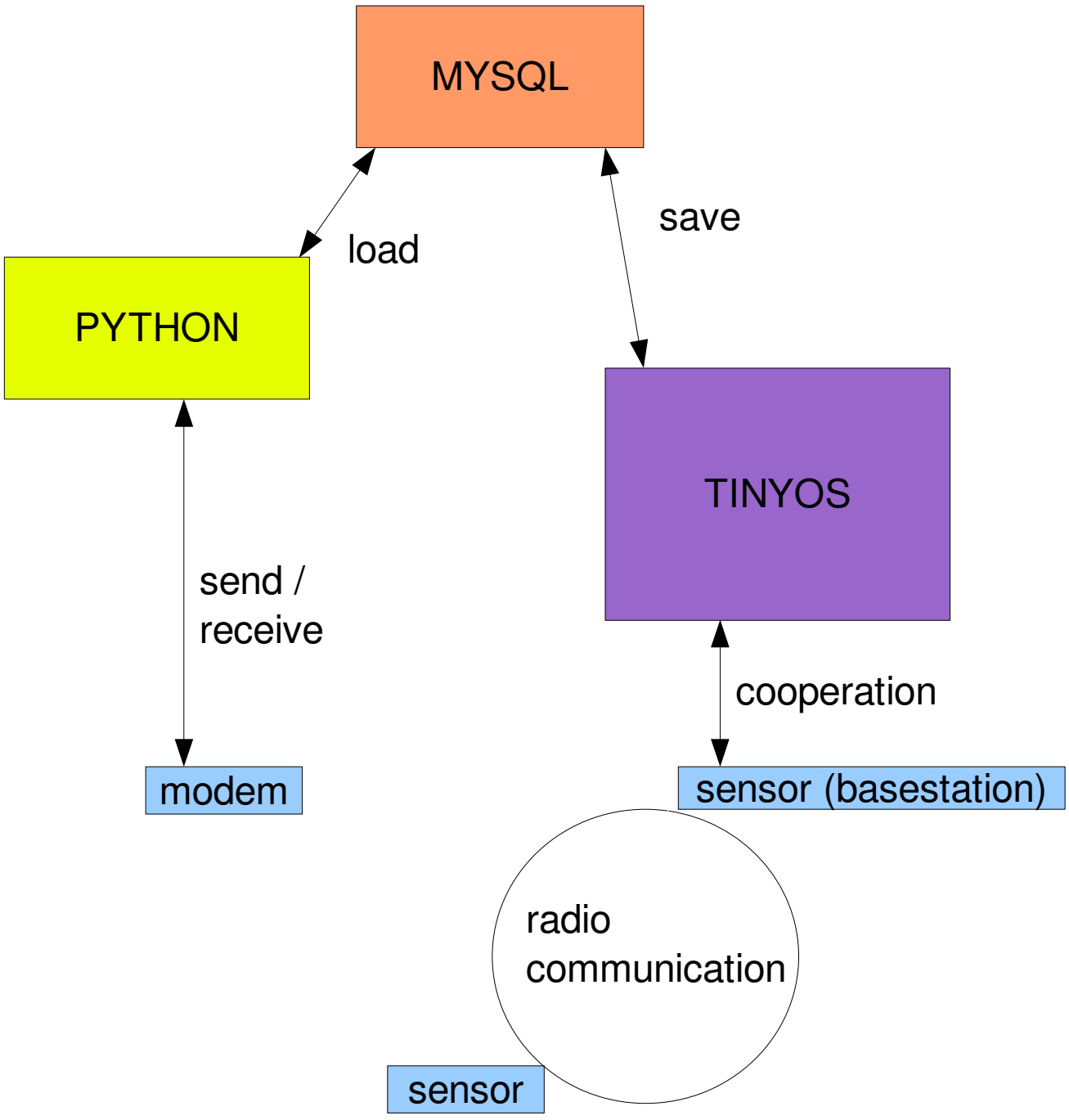


Έχουμε έναν υπολογιστή (soekris), το modem και τον αισθητήρα σε επίπεδο υλικού, και το λειτουργικό σύστημα με την βάση δεδομένων και την εφαρμογή στο παραπάνω επίπεδο λογισμικού.

Οι μετρήσεις που παίρνει ο αισθητήρας και μας ενδιαφέρουν είναι :

- θερμοκρασία
- υγρασία
- συνολική ηλιακή ακτινοβολία (TSR)
- ενεργή φωτοσυνθετική ακτινοβολία (PAR)

Πιο συγκεκριμένα και λίγο πιο αναλυτικά :



Δηλαδή ένας αισθητήρας στέλνει τις μετρήσεις στον σταθμό βάσης. Αυτός μέσω εργαλείων που παρέχει το TinyOS φέρνει τα δεδομένα στον υπολογιστή (soekris), αυτα αποθηκεύονται στην βάση δεδομένων και από εκεί και πέρα η εφαρμογή (python) αναλαμβάνει όλα τα υπόλοιπα, όπως παραλλαβή των μετρήσεων, διάφορες επεξεργασίες και επικοινωνία με το modem.

Η δομή της βάσης δεδομένων, η αλλιώς το σχήμα (schema) μας είναι το ακόλουθο :

nodeid	result_time	temp	TSR	PAR	humidity	counter

όπου :

- **nodeid** το αναγνωριστικό του αισθητήρα
- **result_time** η χρονική στιγμή, την οποία γίνεται η αποθήκευση των μετρήσεων της συγκεκριμένης εγγραφής.
- **temp** η μέτρηση της θερμοκρασίας
- **TSR** η μέτρηση της συνολικής ηλιακής ακτινοβολίας
- **PAR** η μέτρηση της ενεργής φωτοσυνθετικής ακτινοβολίας
- **humidity** η μέτρηση της υγρασίας
- **counter** ο μετρητής του μηνύματος

Ως πρωτεύων κλειδί έχουμε ορίσει το result_time. Όλες οι ιδιότητες των πεδίων του σχήματος είναι :

FIELD	TYPE	NULL	KEY	DEFAULT
nodeid	int	no		0
result_time	timestamp	no	PRI	0
temp	float	yes		null
TSR	double	yes		null
PAR	double	yes		null
humidity	float	yes		null
counter	int	yes		null

3.2 Διαγράμματα και σενάρια

Το σύστημά μας ακολουθεί κάποιο σενάριο εκτέλεσης, κατά την επιθυμία του χρήστη και για κάποια συγκεκριμένη λειτουργικότητα που πρέπει να παρέχεται. Τα σενάρια εκτέλεσης είναι τρία, θα τα περιγράψουμε συνοπτικά και θα παραθέσουμε για το καθένα ένα (ψευδό) διάγραμμα δραστηριότητας στα πρότυπα των UML activity diagrams. Το σενάριο καθορίζεται κατά την εκκίνηση της εφαρμογής αλλά μπορεί να αλλάξει δυναμικά απλά με την αποστολή ενός sms που περιέχει την κατάλληλη εντολή.

ΣΕΝΑΡΙΟ Α

Το πρώτο σενάριο ειδοποιεί τον χρήστη με sms εφόσον κάποια τιμή ενός χαρακτηριστικού ξεπεράσει μια τιμή κατωφλίου. Δηλαδή το σύστημα αρχικοποιείται με μια λίστα όπου υπάρχουν ζευγάρια χαρακτηριστικών και αντίστοιχων τιμών κατωφλίου (η τιμή 0 καθορίζει το να μη λάβουμε υπόψιν κάποιο συγκεκριμένο χαρακτηριστικό). Οπότε το σύστημα απλά διαβάζει τις τιμές από την βάση δεδομένων και εφόσον παρατηρήσει κάποια ανωμαλία, ξεκινά ο μηχανισμός ενημέρωσης του χρήστη. Επιπλέον, ελέγχεται η τυχόν λήψη κάποιου sms-οδηγίας, και επεξεργάζεται κατάλληλα για τυχόν αλλαγή του σεναρίου (εικ. 3.1).

ΣΕΝΑΡΙΟ Β

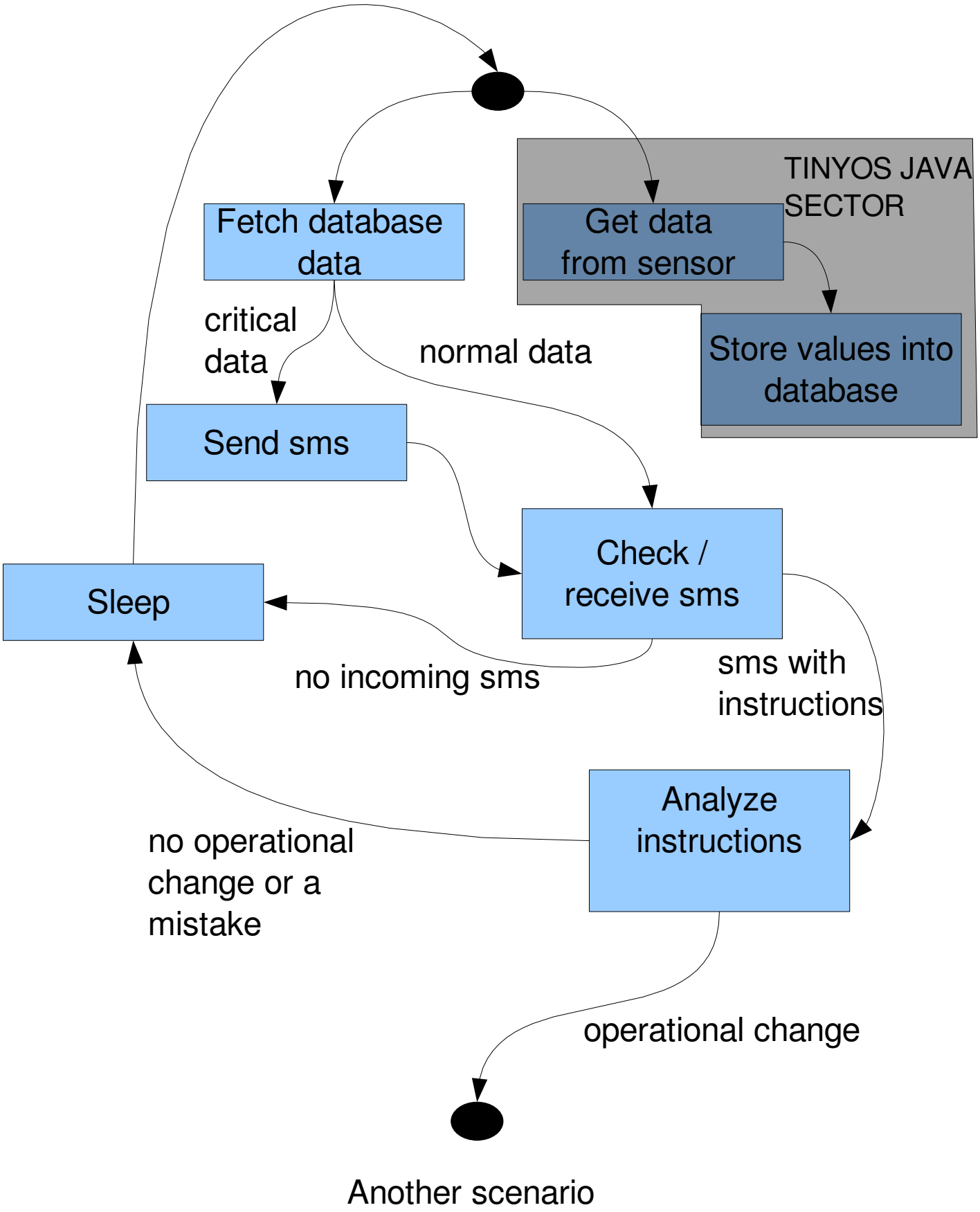
Σε αυτό το σενάριο οι μετρήσεις αποθηκεύονται κανονικά στην βάση δεδομένων (κάτι το οποίο γίνεται σε όλα τα σενάρια και μάλιστα (ψευδο)παράλληλα). Το κυρίως πρόγραμμα όμως απλά περιμένει να λάβει κάποιο sms, το οποίο πιθανώς θα αλλάξει και το σενάριο λειτουργίας (εικ. 3.2).

ΣΕΝΑΡΙΟ Γ

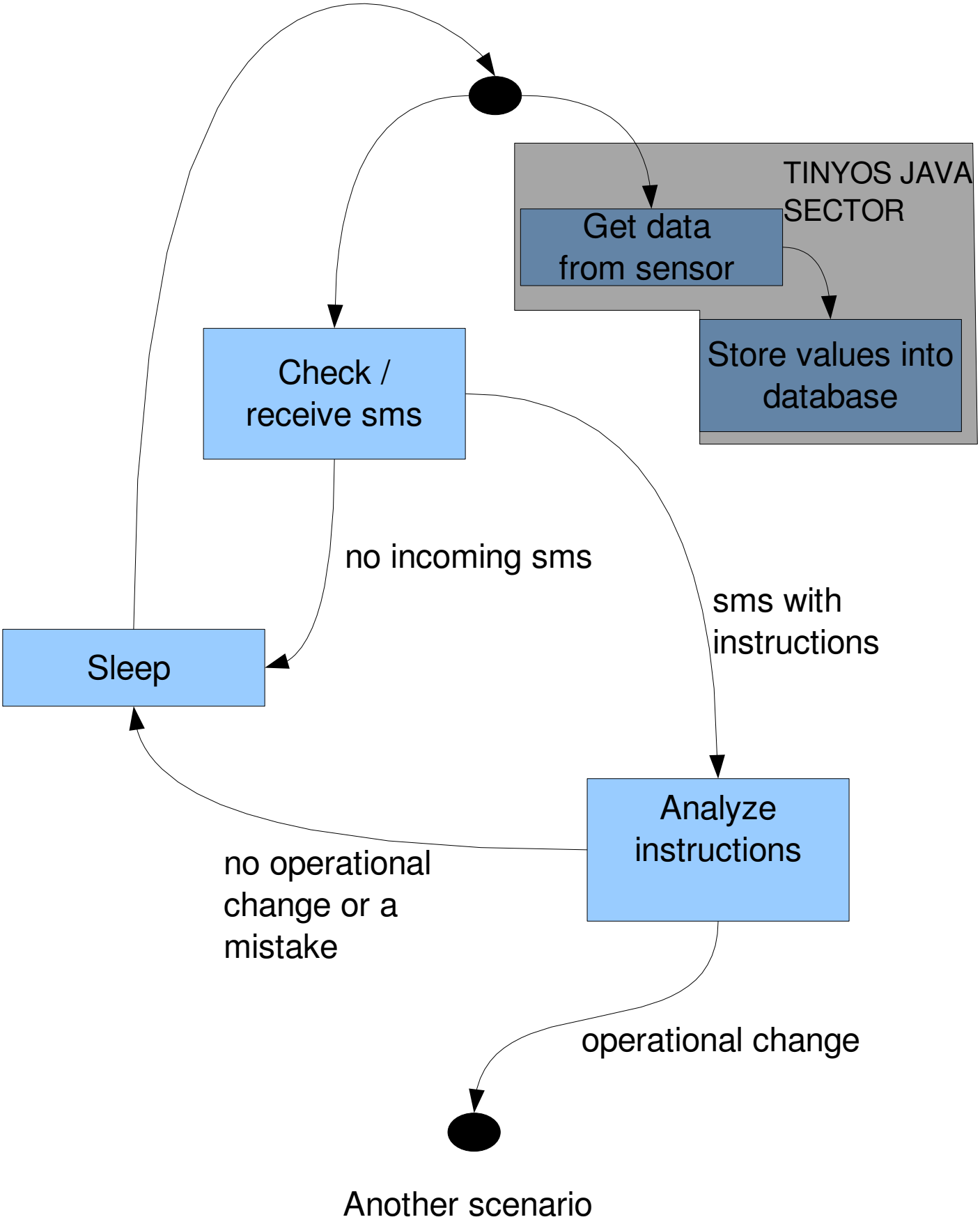
Στο σενάριο αυτό το σύστημα αδρανοποιείται για κάποιο χρονικό διάστημα και ενημερώνει τον χρήστη στέλνοντάς του sms μετά το πέρας του διαστήματος. Ο χρόνος αυτός υπολογίζεται δίνοντας μια μελλοντική ημερομηνία όπου θέλουμε να ενημερωθούμε. Στο τέλος του σεναρίου αυτού το σύστημα επανέρχεται στο σενάριο Β (εικ. 3.3).

ΣΕΝΑΡΙΟ Δ

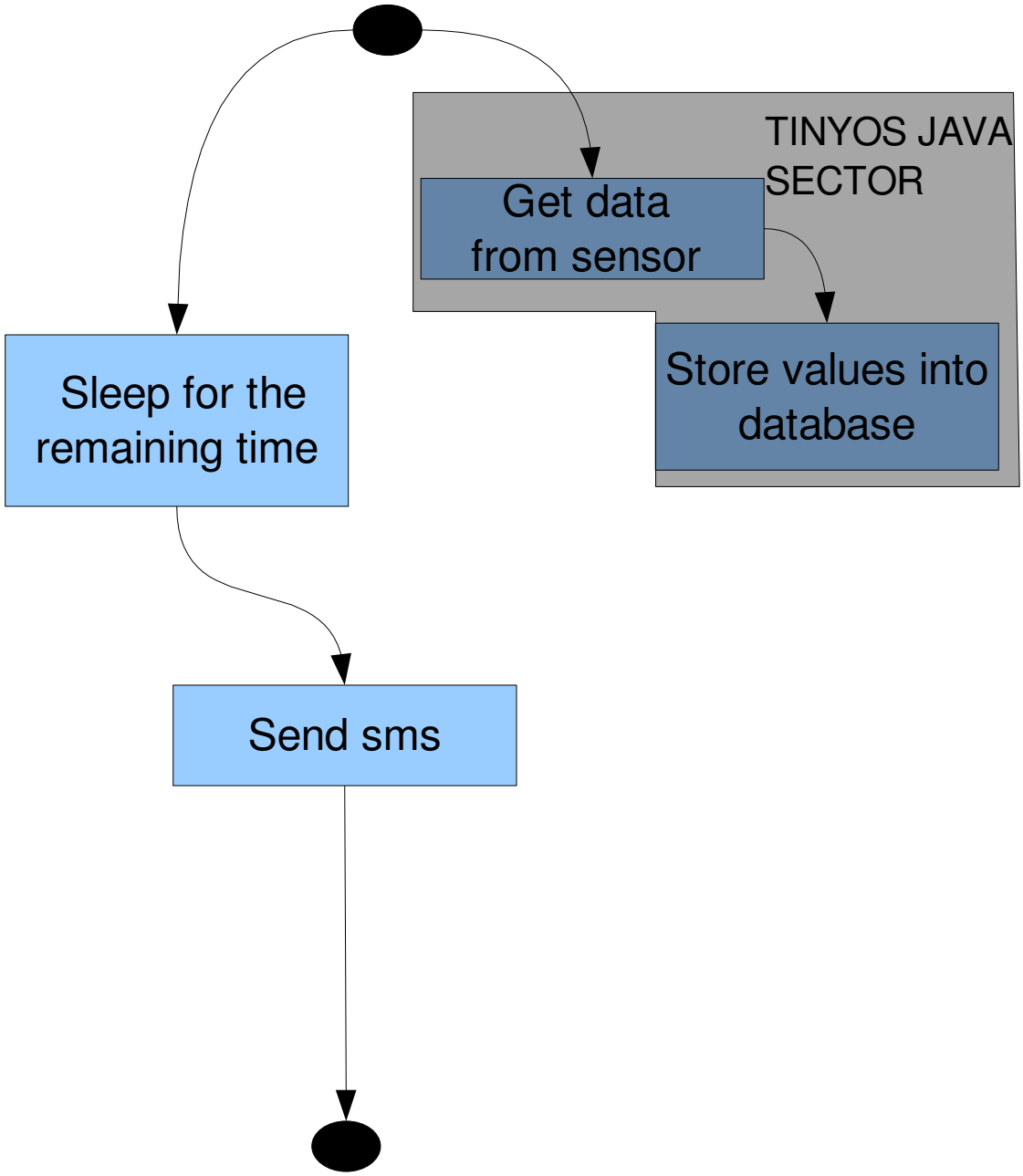
Εδώ έχουμε να κάνουμε με μια παραλλαγή του σεναρίου Γ. Ο χρήστης ορίζει μια μελλοντική χρονική στιγμή ειδοποίησης και μια σταθερά επανάληψης, ώστε να ειδοποιείται περιοδικά όπως επιθυμεί (σε επαναλαμβανόμενα σταθερά χρονικά διαστήματα)(εικ. 3.4).


TINYOS JAVA SECTOR
Fetch database data
Get data from sensor
Store values into database
critical data
normal data
Send sms
Check / receive sms
Sleep
no incoming sms
sms with instructions
Analyze instructions
no operational change or a mistake
operational change
Another scenario

εικ. 3.1

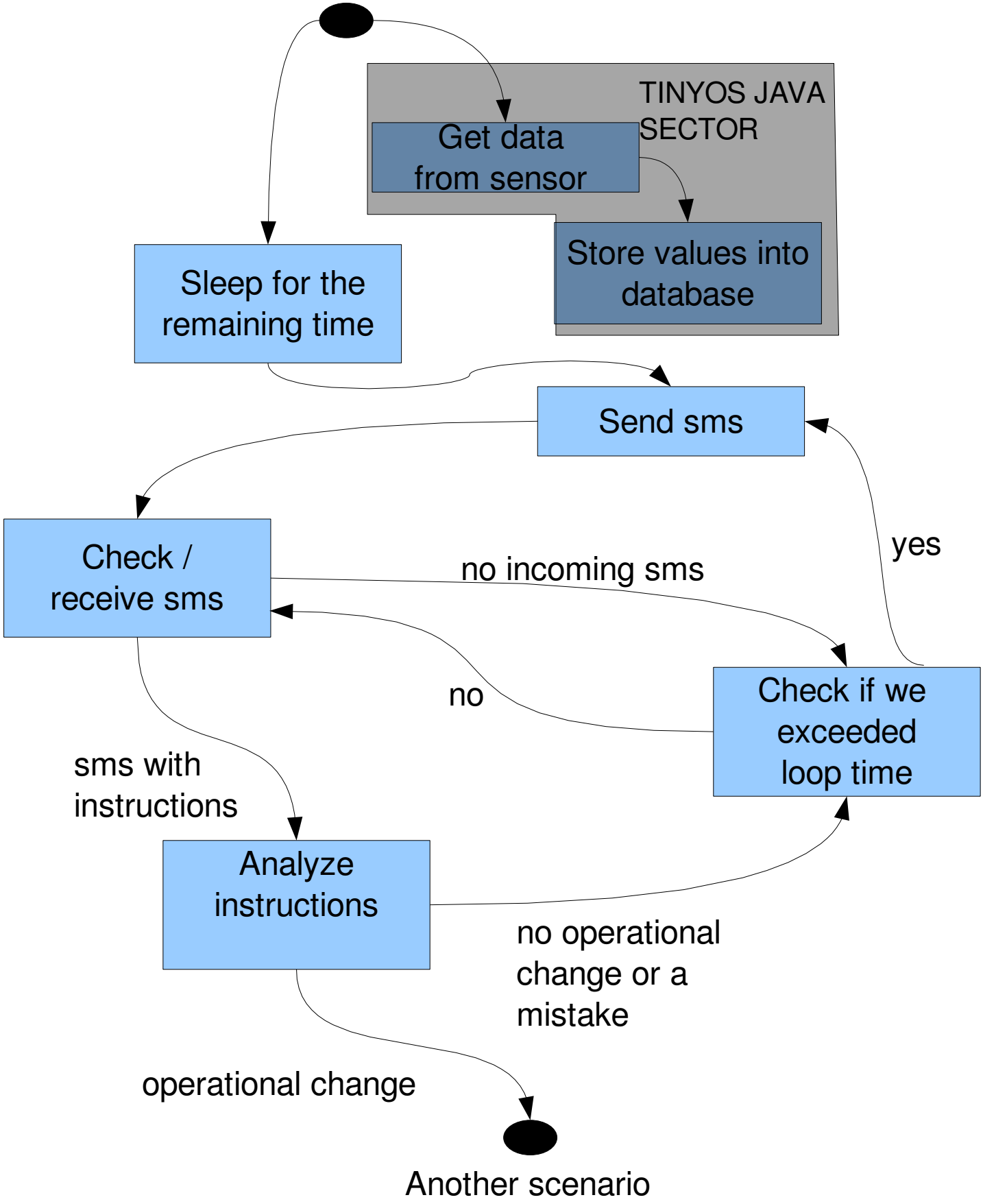

TINYOS JAVA SECTOR
Get data from sensor
Store values into database
Check / receive sms
no incoming sms
sms with instructions
Sleep
Analyze instructions
no operational change or a mistake
operational change
Another scenario

εικ. 3.2



Scenario B

εικ. 3.3


TINYOS JAVA SECTOR
Get data from sensor
Store values into database
Sleep for the remaining time
Send sms
Check / receive sms
no incoming sms
yes
no
Check if we exceeded loop time
sms with instructions
Analyze instructions
no operational change or a mistake
operational change
Another scenario

εικ. 3.4

4.

Εγκατάσταση Λειτουργικού Συστήματος

4.1 Εισαγωγικά

Η εγκατάσταση του λειτουργικού συστήματος θα γίνει μέσω δικτύου, ή αλλιώς μέσω ενός PXE (Preboot Execution Environment) server. Δηλαδή θα στήσουμε ένα μηχάνημα-server που θα αναλάβει να κάνει την εγκατάσταση μέσω δικτύου (LAN) σε ένα μηχάνημα-client, το Soekris net5501. Ακολουθούν λεπτομερείς οδηγίες για το πως θα γίνει η διεκπεραίωση όλων των παραπάνω.

4.2 Λεπτομερής ανάλυση

Εδώ παραθέτουμε τα βήματα της εγκατάστασης. Ο PXE server μας “τρέχει” λειτουργικό σύστημα Ubuntu 9.04. Τα πακέτα που θα χρειαστούμε είναι τα εξής : inetutils-inetd, tftp-hpa και dhcp3-server.
Ανοίγουμε μια κονσόλα (terminal) και γινόμαστε υπερχρήστες, καθώς από εδώ και πέρα όλες οι εντολές θα γίνονται από τον root λογαριασμό μας. Αν δεν έχουμε δημιουργήσει ένα root login, γράφουμε :

```
sudo passwd root
```

και βάζουμε τον κωδικό του root. Έπειτα συνδεόμαστε ως root :

```
su
```

κάνουμε εγκατάσταση των απαραίτητων πακέτων για το στήσιμο ενός PXE server με την εντολή

```
apt-get install inetutils-inetd tftpd-hpa dhcp3-server
```

έπειτα θα τροποποιήσουμε το αρχείο tftpd-hpa ώστε να αρχίσει τον daemon του (κάτι το οποίο δεν γίνεται by default)

```
gedit etc/default/tftp-hpa
```

εκεί θα δούμε οτι το αρχείο περιέχει τις ακόλουθες γραμμές :

#Defaults for tftpd-hpa
RUN_DAEMON="no"
OPTIONS="-l -s /var/lib/tftpboot"

αλλάζουμε το “no” με “yes” και κάνουμε save. Μετά κάνουμε επαννεκίνηση το script αυτό για να σηκώσουμε τον daemon με την εντολή :

```
/etc/default/tftpd-hpa restart
```

Τώρα θα ρυθμίσουμε το DHCP. Αν δεν έχουμε ρυθμίσει το network interface μας (π.χ.το eth0) θα το κανουμε τώρα. Ανοίγουμε το αρχείο /etc/network/interfaces

```
gedit /etc/network/interfaces
```

και προσθέτουμε τις παρακάτω γραμμές κώδικα :

auto eth0
iface eth0 inet static
address 192.168.0.0
netmask 255.255.255.0

κάνουμε save και ανοίγουμε το etc/default/dhcp3-server

```
gedit /etc/default/dhcp3-server
```

και σιγουρευόμαστε οτι λέει :

INTERFACES="eth0"

έπειτα ανοίγουμε το etc/dhcp3/dhcpd.conf

```
gedit /etc/dhcp3/dhcpd.conf
```

και του προσθέτουμε (στο τέλος) :

```
subnet 192.168.2.0 netmask 255.255.255.0 {
        option domain-name-servers 208.67.222.222;
         option routers 192.168.2.1;
        range 192.168.2.50 192.168.2.200;
        filename "pxelinux.0";
}
```

Μετά επαννεκινούμε τον DHCP server μας με την εντολή

```
/etc/init.d/dhcp3-server restart
```

ο DHCP server είναι έτοιμος! Τώρα στο αρχείο /etc/inetd.conf θα πρέπει να υπάρχει η παρακάτω γραμμή :

```
tftp dgram udp wait root /usr/sbin/in.tftpd

/usr/sbin/in.tftpd -s /var/lib/tftpboot
```

που σημαίνει οτι είμαστε εντάξει και μπορούμε να κάνουμε εγκατάσταση κάποιας διανομής από κάποιο αποθετήριο στο internet αντιγράφοντας το netboot αρχείο της διανομής στο /var/lib/tftpboot.

Εγκαθιστούμε το πρόγραμμα minicom:

```
apt-get install minicom
```

καθώς θα συνδέσουμε το soekris με ένα καλώδιο USB-to-serial-converter στο μηχάνημα με τον PXE server, και θα βλέπουμε την έξοδό του στο minicom. Αφού ρυθμίσουμε το minicom να “ακούει” στην θύρα που το λειτουργικό μας αναγνώρισε το soekris (π.χ. /dev/ttyUSB0, που είναι και η default θύρα του minicom), ρυθμίζουμε επίσης το baud rate στα 19200 8N1, το οποίο είναι το default του soekris. Τώρα θα χρειαστεί να τροποποιήσουμε κάποιες παραμέτρους του netboot, ώστε να δουλέψει με την σειριακή κονσόλα (minicom). Ανοίγουμε το αρχείο /var/lib/tftpboot/debian-installer/i386/boot-screens/txt.cfg και συμπληρώνουμε ότι χρειαστεί ώστε να έχουμε τελικά:

```
default install
label install
        menu label ^Install
        menu default
        kernel debian-installer/i386/linux
        append vga=normal initrd=debian-installer/i386/initrd.gz --
console=ttyUSB0,19200 earlyprint=serial,ttyUSB0,19200
```

όπου αντικαθιστούμε το ttyUSB0 με ttyUSBx ανάλογα πώς έχει αναγνωριστεί το soekris μέσω του καλωδίου USB-to-serial-converter.

Επίσης στο /var/lib/tftpboot/debian-installer/i386/boot-screens/syslinux.cfg φροντίζουμε να έχουμε:

```
# D-I config version 1.0
CONSOLE 0
SERIAL 0 19200 0
include debian-installer/i386/boot-screens/menu.cfg
default debian-installer/i386/boot-screens/vesamenu.c32
prompt 0
timeout 0
```

και είμαστε έτοιμοι!!

Συνδέουμε το soekris (την θύρα LAN0) στο δίκτυό μας με κάποιο καλώδιο ftp. Το συνδέουμε με την τροφοδοσία και βλέπουμε την έξοδό μας στο minicom. Όπως μας προτρέπει η ίδια η κονσόλα, πατάμε Ctrl + P και δίνουμε boot f0, προτρέποντας το soekris να boot-άρει από το δίκτυο. Στη συνέχεια, κάνουμε μια φυσιολογική εγκατάσταση του Debian, την οποία για λόγους χώρου δεν θα παρουσιάσουμε εδώ.

4.3 Προσθήκη απαραίτητων εργαλείων

Τώρα πλέον όποτε ανοίγουμε το soekris θα ξεκινάμε σε Debian περιβάλλον κονσόλας (αυτό που θα βλέπουμε στο minicom). Θα πρέπει να εγκαταστήσουμε κάποια επιπλέον εργαλεία, όπως TinyOS, Mysql, Java κ.α., καθώς το απαιτεί η εφαρμογή μας. Να σημειώσουμε ότι "μπαίνουμε" προς το παρόν στο σύστημα ως root χρήστες. Αναλυτικότερα:

4.3.1 TinyOS

Έχουμε το πλεονέκτημα να υπάρχει έτοιμο το TinyOS στα αποθετήρια του Debian. Οπότε προσθέτουμε στο /etc/apt/sources.list αφού το ανοίξουμε:

```
nano /etc/apt/sources.list
```

τα παρακάτω "αποθετήρια":

deb http://ftp.se.debian.org/debian lenny main contrib non-free
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu feisty main

έπειτα:

```
apt-get update
apt-get install tinyos-2.0.2
```

το οποίο εγκαθιστάτα παρακάτω πακέτα:

avr-binutils-tinyos, avr-gcc-tinyos, avr-libc-tinyos, avr-optional-tinyos, avr-tinyos, avr-tinyos-base, avrdude-tinyos, deputy-tinyos, graphviz, gsfonts-x11, java-common, libgraphviz4, libnss-mdns, msp430-binutils-tinyos, msp430-gcc-tinyos, msp430-libc-tinyos, msp430-optional-tinyos, msp430-tinyos, msp430-tinyos-base, nesc, odbcinst1debian1, python-serial, sun-java5-bin, sun-java5-demo, sun-java5-jdk, sun-java5-jre, tinyos-2.0.2, tinyos-base, tinyos-required-all, tinyos-required-avr, tinyos-required-msp430, tinyos-tools, ttf-liberation, unixodbc

και το TinyOS πλέον βρίσκεται εγκατεστημένο στο σύστημά μας. Δεν μένει παρά να προσθέσουμε μια γραμμή στα αρχεία .bashrc και /etc/profile:

```
nano /etc/profile
nano /home/user/.bashrc
```

στο τέλος από τα παραπάνω αρχεία προσθέτουμε:

source /opt/tinyos-2.0.2/tinyos.sh

το οποίο ουσιαστικά είναι ένα bash script που κάνει export κάποιες μεταβλητές περιβάλλοντος που είναι απαραίτητες για την ορθή λειτουργία του TinyOS. Βάζοντάς το στα παραπάνω αρχεία κάθε shell που ξεκινάει θα “βλέπει” αυτές τις μεταβλητές και δεν θα μας απασχολήσει με “not found” μηνύματα.

Θα επανέλθουμε στο TinyOS στο κεφάλαιο 4.3.4, καθώς κάναμε απλά την εγκατάστασή του και δεν το ρυθμίσαμε ώστε να λειτουργεί απρόσκοπτα.

4.3.2 Java

Εάν ρίξουμε μια ματιά στα πακέτα που εγκαταστάθηκαν μαζί με το TinyOS, θα δούμε ότι συμπεριλαμβάνεται και η 5η έκδοση της java. Θα περίμενε κανείς ότι είμαστε εντάξει, αλλά προς κακή μας τύχη στην πορεία διαπιστώνουμε ότι έχουμε να αντιμετωπίσουμε ένα bug. Δηλαδή, όταν (όπως θα δούμε στην συνέχεια) ρυθμίσουμε το σύστημα μας για να λειτουργήσει απρόσκοπτα, η συνεργασία java – TinyOS απλά δεν είναι δυνατή. Η λύση βρίσκεται στην εγκατάσταση της 6ης έκδοσης της java, που αποδείχτηκε λειτουργική. Οπότε δίνουμε:

```
apt-get install sun-java6-jdk
```

και αφού τελειώσει η εγκατάσταση

```
update-alternatives –config java
```

και με βάση τις οδηγίες που θα δούμε στην κονσόλα καθιστούμε ενεργή την 6η έκδοση της java έναντι της 5ης. Ανάλογα πράττουμε και για τον compiler (javac):

```
update-alternatives -config javac
```

Τέλος, εγκαθιστούμε και το πακέτο libmysql-java, ώστε να συνεργαστεί η βάση δεδομένων μας με την java.

```
apt-get install libmysql-java
```

4.3.3 MySQL

Θα εγκαταστήσουμε το πακέτο mysql-server:

```
apt-get install mysql-server
```

Κατά την διάρκεια της εγκατάστασης θα μας ζητηθεί να δώσουμε ένα συνθηματικό για τον root user της mysql. Αφού έχουμε εγκαταστήσει τον mysql-server, θα πρέπει να δημιουργήσουμε μια βάση δεδομένων και κάποιον χρήστη με δικαιώματα πρόσβασης σε αυτή. Μπαίνουμε ως root στην mysql:

```
mysql -u root
```

και αφού βάλουμε τον root κωδικό δίνουμε:

```
CREATE DATABASE my_db;

USE my_db;

GRANT USAGE ON *.* TO user@localhost IDENTIFIED BY
'12345';

GRANT ALL PRIVILEGES ON my_db.* TO  soekris@localhost;
```

με τις παραπάνω εντολές φτιάξαμε μια βάση δεδομένων με όνομα my_db και έναν χρήστη user με πλήρη δικαιώματα πάνω σε αυτή. Έπειτα θα "μορφοποιήσουμε" την βάση μας και θα της εισάγουμε τα κατάλληλα πεδία όπου θα αποθηκεύονται οι μετρήσεις μας :

```
CREATE TABLE sensor (
  result_time TIMESTAMP NOT NULL PRIMARY KEY,
  temp DOUBLE,
  tsr DOUBLE,
  par DOUBLE,
  humidity DOUBLE,
  nodeid INTEGER,
  counter INTEGER
)ENGINE = MYISAM ;
```

και η βάση δεδομένων μας πλέον είναι έτοιμη να εκετελέσει τα καθήκοντά της!

4.3.4 Διάφορα

Θα χρειαστούμε να εγκαταστήσουμε το πακέτο make, καθώς χρειάζεται για να κάνουμε compile σε TinyOS εφαρμογές:

```
apt-get install make
```

Επίσης, θα χρειαστούμε το πακέτο-εργαλείο sudo, για να μπορεί ο χρήστης (που τον ορίσαμε στην εγκατάσταση του Debian) να εκτελεί ως root κάποιες εντολές:

```
apt-get install sudo
```

Θα πρέπει να ορίσουμε σε ποιόν χρήστη θα δώσουμε δικαιώματα sudo:

```
nano /etc/sudoers
```

και προσθέτουμε:

our_user ALL=(ALL) ALL

όπου our_user το όνομα του χρήστη που θέλουμε.

4.4 Τελικές ρυθμίσεις

Θα προσθέσουμε τις τελευταίες “πινελιές” ώστε να φέρουμε το σύστημα στην επιθυμητή κατάσταση.

Ανοίγουμε το αρχείο tinyos.sh:

```
nano /opt/tinyos-2.0.2/tinyos.sh
```

και προσθέτουμε (αυτά που φαίνονται με κόκκινο):

```
echo "Setting up for TinyOS 2.0.2"
export TOSROOT=
export TOSDIR=
export MAKERULES=

TOSROOT="/opt/tinyos-2.0.2"
TOSDIR="$TOSROOT/tos"
CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java
CLASSPATH=$CLASSPATH:/usr/share/java/mysql-5.1.6.jar
CLASSPATH=$CLASSPATH:/opt/tinyos-2.0.2/apps/Sensing/
MAKERULES="$TOSROOT/support/make/Makerules"

export TOSROOT
export TOSDIR
export CLASSPATH
export MAKERULES
```

Έτσι, όπως είδαμε και προηγουμένως στο κεφάλαιο 4.3.1, κάθε shell που ανοίγουμε θα ξέρει ακριβώς που υπάρχουν κάποιες συγκεκριμένες παράμετροι-μεταβλητές του συστήματος. Έπειτα:

```
cd /opt/tinyos-2.0.2/support/sdk/java
make
```

και μετά:

```
tos-install-jni
```

και τέλος!! Πλέον έχουμε ένα πλήρες λειτουργικό soekris, με όλα τα εργαλεία που χρειαζόμαστε εγκατεστημένα και ρυθμισμένα ώστε να τρέξουν τις εφαρμογές μας.

5.
ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΑΙΣΘΗΤΗΡΩΝ

5.1 Εισαγωγή

Οι αισθητήρες είναι συσκευές που συνδέονται μέσω USB με τον υπολογιστή. Τα προγράμματα που τρέχουν σε αυτά είναι TinyOS εφαρμογές γραμμένες σε γλώσσα NesC. Υπάρχουν αρκετές τέτοιες εφαρμογές-προγράμματα έτοιμα προς χρήση στην πλατφόρμα του TinyOS. Εμείς φτιάξαμε μια τέτοια εφαρμογή που να ταιριάζει στα χαρακτηριστικά της εργασίας, δηλαδή να μετράει θερμοκρασία, υγρασία, συνολική ηλιακή ακτινοβολία (TSR) και ενεργή φωτοσυνθετική ακτινοβολία (PAR). Το “υλικό” έχει τις προδιαγραφές να μετράει τις τιμές των παραπάνω χαρακτηριστικών. Αυτό που απομένει είναι ο σωστός προγραμματισμός σε NesC των συστατικών-components ώστε να επιτευχθεί η λειτουργικότητα που θέλουμε.

Η αρχιτεκτονική του TinyOS είναι συνοπτικά η εξής: Οι εφαρμογές μεταγλωτίζονται και παράγουν ένα μοναδικό binary image που έχει τον πλήρη έλεγχο του υλικού (hardware) του αισθητήρα. Έτσι ένας αισθητήρας τρέχει ένα και μοναδικό τέτοιο image κάθε φορά. Μια εφαρμογή γραμμένη σε NesC αποτελείται από ένα ή περισσότερα components. Αυτά καθορίζουν δύο τινά, την προδιαγραφή τους η οποία περιέχει τα ονόματα των interfaces και την υλοποίησή τους. Ένα component δηλαδή προδιαγράφει (παρέχει) και χρησιμοποιεί interfaces. Τα interfaces που παρέχονται αναπαριστούν την λειτουργικότητα που το component παρέχει στον χρήστη με βάση τις προδιαγραφές. Με άλλα λόγια τα χρησιμοποιούμενα interfaces αναπαριστούν την λειτουργικότητα που το component χρειάζεται για να πραγματοποιήσει τον σκοπό του μέσα από την υλοποίησή του.

Τα interfaces έχουν διπλή σημασία. Προδιαγράφουν ένα σύνολο εντολών και ένα σύνολο γεγονότων τα οποία είναι συναρτήσεις προς υλοποίηση. Για να καλέσει ένα component τις εντολές ενός interface, πρέπει να υλοποιήσει τις συναρτήσεις χειρισμού των γεγονότων του interface.

Υπάρχουν δυο τύποι components στην NesC: τα modules και τα configurations. Τα modules παρέχουν την υλοποίηση ενός ή περισσοτέρων interfaces. Τα configurations χρησιμοποιούνται για να ενώσουν άλλα components μαζί, ενώνοντας interfaces που χρησιμοποιούνται από components με interfaces που παρέχονται από άλλους. Κάθε εφαρμογή NesC περιγράφεται από ένα top-level configuration που “δένει” (wires) τα εσωτερικά components.

5.2 Η εφαρμογή Basestation

Η εφαρμογή Basestation έρχεται έτοιμη μαζί με την εγκατάσταση του TinyOS. Λειτουργεί σαν γέφυρα μεταξύ της σειριακής θύρας USB και του ραδιο-δικτύου. Το ραδιο-δίκτυο (radio network) είναι ουσιαστικά ο τρόπος επικοινωνίας των αισθητήρων, κάτι το οποίο είναι εγγενές χαρακτηριστικό τους. Η εφαρμογή Basestation αναλαμβάνει να προωθεί ή να δέχεται πακέτα σε/από αυτό το δίκτυο σε συνεργασία με την σειριακή θύρα. Πρακτικά αυτό σημαίνει ότι ένας υπολογιστής με συνδεδεμένο αισθητήρα (σε κάποια θύρα USB) που τρέχει αυτή την εφαρμογή έχει πλήρη πρόσβαση σε και αλληλεπίδραση με δίκτυα αισθητήρων.

Χρησιμοποιούμε την εφαρμογή Basestation για να λαμβάνουμε τις μετρήσεις από αισθητήρες που τρέχουν την εφαρμογή Sensing, όπως θα δούμε παρακάτω.

5.3 Η εφαρμογή Sensing

Η εφαρμογή Sensing είναι μια εφαρμογή TinyOS/NesC που αναλαμβάνει την παρακολούθηση και καταγραφή των χαρακτηριστικών που μας ενδιαφέρουν. Χρησιμοποιώντας τους εσωτερικούς μικροαισθητήρες του αισθητήρα Tmote Sky, αποτυπώνει την τρέχουσα θερμοκρασία, υγρασία κ.τ.λ., και στέλνει τις μετρήσεις στο ραδιο-δίκτυο ώστε να παραληφθούν από τον σταθμό βάσης (τον κόμβο που “τρέχει” την εφαρμογή Basestation) για επεξεργασία. Ας δούμε συνοπτικά την λογική και κάποια κομμάτια κώδικα της εφαρμογής:

Η εφαρμογή αποτελείται από τέσσερα αρχεία:

- SensingC.nc
- SensingAppC.nc
- Sensing.h
- Makefile

Το Sensing.h είναι η “βιβλιοθήκη” της εφαρμογής. Δηλώνει τις σταθερές και τους τύπους δεδομένων της:

```
#ifndef SENSING_H
#define SENSING_H

enum {
  AM_SENSINGMSG = 6,
  TIMER_PERIOD_MILLI = 2500
};

typedef nx_struct SensingMsg {
 nx_uint16_t nodeid;
 nx_uint16_t par;
 nx_uint16_t tsr;
 nx_uint16_t temp;
 nx_uint16_t hum;
 nx_uint16_t counter;
} SensingMsg;

#endif
```

Συνοπτικά βλέπουμε ο τύπος του πακέτου AM είναι 6 (AM_SENSINGMSG) και ο χρόνος που μεσολαβεί μεταξύ των αποστολών πακέτων είναι 2500 milliseconds (TIMER_PERIOD_MILLI). Επίσης η δομή του πακέτου είναι έξι στοιχεία τύπου int16 που το καθένα είναι και κάποιο χαρακτηριστικό που μας ενδιαφέρει.

Το SensingC.nc παρέχει την λογική υλοποίησης του προγράμματος (τα modules και την υλοποίησή τους):

```
.....
  uses interface Timer<TMilli> as Timer0;
  uses interface Packet;
  uses interface AMPacket;
  uses interface AMSend;
  uses interface Read<uint16_t> as ReadPAR;
  uses interface Read<uint16_t> as ReadTSR;
.....
```

δηλαδή ορίζουμε πια interfaces θα χρησιμοποιήσουμε (για χειρισμό πακέτων, για χρονοπρογραμματισμό, για μέτρηση χαρακτηριστικών κ.α.). Έπειτα ακολουθεί η υλοποίηση:

```
implementation {
.....................
}
```

όπου προγραμματιστικά πλέον υλοποιούμε συναρτήσεις που πυροδοτούν γεγονότα ή πυροδοτούνται από αυτά, και μας παρέχουν την απαιτούμενη λειτουργικότητα (βλ. Κεφάλαιο 11).

Το SensingAppC.nc παρέχει το απαραίτητο “δέσιμο” (wiring) μεταξύ των components της εφαρμογής και των interfaces (είναι δηλαδή το configuration της εφαρμογής):

```
 #include <Timer.h>
#include "Sensing.h"

configuration SensingAppC {
}
implementation {
  components MainC;
  components LedsC;
  components SensingC as App;
  components new TimerMilliC() as Timer0;
  components ActiveMessageC;
  components new HamamatsuS1087ParC() as PAR;
  components new HamamatsuS10871TsrC() as TSR;
  components new SensirionSht11C() as SensorSht;
  components new AMSenderC(AM_SENSINGMSG);
  components new AMReceiverC(AM_SENSINGMSG);



  App.Boot -> MainC;
  App.Leds -> LedsC;
  App.Timer0 -> Timer0;
  App.ReadPAR -> PAR;
  App.ReadTSR -> TSR;
  App.ReadTemperature -> SensorSht.Temperature;
  App.ReadHumidity -> SensorSht.Humidity;
  App.Packet -> AMSenderC;
  App.AMPacket -> AMSenderC;
  App.AMSend -> AMSenderC;
  App.AMControl -> ActiveMessageC;
  App.Receive -> AMReceiverC;

}
```

Υπάρχουν δηλαδή οι δηλώσεις των components που χρησιμοποιεί η εφαρμογή και τα οποία παρέχουν τα interfaces που επιλέχθηκαν στο SensingC.nc. Επίσης γίνεται το wiring των interfaces στα components που τα παρέχουν.

Τέλος το Makefile είναι ένα αρχείο για να μεταγλωτιστεί σωστά η εφαρμογή με το εργαλείο make.

5.4 Εγκατάσταση

Η εγκατάσταση των εφαρμογών Basestation και Sensing μπορεί να γίνει σε οποιονδήποτε υπολογιστή που έχει σωστά εγκατεστημένο το περιβάλλον TinyOS.

Αφού τοποθετήσουμε τους αισθητήρες Tmote Sky σε κάποιες USB θύρες και βρούμε που αντιστοιχούν αυτές οι συσκευές σε επίπεδο λειτουργικού (δίνοντας την εντολή 'motelist' ή την καθαρά unix-οειδή 'dmesg | tail'), μπαίνουμε διαδοχικά στους φακέλους που βρίσκονται οι εφαρμογές, τις μεταγλωτίζουμε και τις εγκαθιστούμε στους αισθητήρες. Για παράδειγμα:

Βάζουμε το πρώτο Tmote Sky σε μια θύρα. Δίνουμε σε ένα τερματικό:

```
motelist
```

Βλέπουμε ως έξοδο:

```
Reference  CommPort   Description
---------- ---------- --------------------------------------
UCC89MXV   /dev/ttyUSB1       Telos (Rev B 2004-09-27)
```

Άρα ξέρουμε ότι το Tmote Sky μας έχει αναγνωριστεί ως /dev/ttyUSB1.Ενδύκνειται να αλλάξουμε τα permissions της συσκευής:

```
sudo chmod 666 /dev/ttyUSB1
```

Πηγαίνουμε στον φάκελο της εφαρμογής που μας ενδιαφέρει:

```
cd /opt/tinyos-2.0.2/apps/BaseStation
```

και δίνουμε:

```
make telosb reinstall, 1 bsl,/dev/ttyUSB1
```

Η παραπάνω εντολή μεταγλωτίζει την εφαρμογή Basestation και την εγκαθιστά στην συσκευή /dev/ttyUSB1 προσδίδοντάς της και την ονομασία “κόμβος 1” (reinstall ***1***), ένα αναγνωριστικό κόμβου δηλαδή για λόγους διάκρισης.

Την ίδια ακριβώς διαδικασία ακολουθούμε για να εγκαταστήσουμε την εφαρμογή Sensing σε κάποιο άλλο Tmote Sky.

Πρέπει να τονίσουμε το γεγονός ότι αυτή η διαδιακασία αρκεί να γίνει μια φορά, καθώς μετά το επιτυχές πέρας της ο κάθε αισθητήρας έχει εγκαταστημένη την ανάλογη εφαρμογή “για πάντα”, η οποία τίθεται σε λειτουργία κάθε φορά που παρέχουμε ρεύμα στον αισθητήρα, είτε μέσω USB θύρας είτε μέσω μπαταριών. Μπορούμε φυσικά να κάνουμε εγκατάσταση οποιασδήποτε εφαρμογής “σβήνοντας” την παλιά.

6.
ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ MODEM

6.1 Γενικά

Το modem μας συνδέεται μέσω USB με το Soekris net5501. Ο προγραμματισμός του εναπόκειται στην σωστή χρησιμοποίηση των κατάλληλων AT εντολών, που είναι οι κατ' εξοχήν αρμόδιες για να “μιλήσει” κανείς σε σχεδόν όλους τους τύπους modem. Οι AT εντολές, ή αλλιώς το σύνολο εντολών του Hayes, είναι μια αλληλουχία από μικρές κωδικές λέξεις οι οποίες συνεργάζονται με το modem ώστε να ολοκληρώσουν λειτουργίες όπως κλήση, αναμονή ή αλλαγή παραμέτρων της σύνδεσης.

Στην περίπτωσή μας επικοινωνούμε με το modem μέσω της γλώσσας python, και συγκεκριμένα με την βοήθεια του module python-serial, το οποίο ενθυλακώνει την πρόσβαση σε σειριακές θύρες. Η οργάνωση του συνόλου του κώδικα python έχει γίνει σε modules, όπου κάθε ένα από αυτά υλοποιεί και κάποια διαφορετική λειτουργικότητα. Πέρα από την διευθέτηση της λήψης/αποστολής sms έχουμε εμπλουτίσει την εφαρμογή με επιπλέον χαρακτηριστικά. Παραθέτουμε αναλυτικά τα modules της python που χρησιμοποιούνται για την διαδραστικότητα με το modem.

6.2 Sms python modules

Ας δούμε όμως αναλυτικά τα modules της εφαρμογής.

gsmmodem.py

Το συγκεκριμένο module ουσιαστικά είναι η υλοποίηση μιας κλάσης, η οποία περιγράφει το modem μας. Χρησιμοποιεί το module serial, γι' αυτό και η πρώτη γραμμή είναι:

```
import serial
```

Αρχικοποιεί το modem το οποίο η linux διανομή μας το βλέπει ως /dev/ttyUSBx (όπως παρατηρούμε από τον κώδικα αυτό περνιέται ως παράμετρος, η οποία “ανακαλύπτεται” με κάποιο installation script το οποίο θα παραθέσουμε και θα αναλύσουμε αργότερα) και του θέτει baud rate (symbols or pulses per second) 115200 και timeout 1 δευτερόλεπτο (δηλαδή 1 δευτερόλεπτο αναμονή για ανάγνωση/γράψιμο). Επίσης εκτελεί μια πράξη flush σε είσοδο και έξοδο ώστε σε κάθε περίπτωση να καθαρίσουν τα αντίστοιχα buffers από τυχόν δεδομένα:

```
def __init__(self, dev):
    self.ser = serial.Serial("%s" % dev, 115200, timeout=1)
    self.ser.flushInput()
    self.ser.flushOutput()
```

Παρέχει συναρτήσεις για αποστολή εντολών στο modem, παραλαβή απλών, μονών ή διπλών αποτελεσμάτων (ουσιαστικά την ανταπόκριση του modem κάθε AT εντολή):

receiveChar(), receiveSingleResult(), receiveDualResult(), receiveLine(), sendCommand()

sendsms.py

Βασίζεται στο gsmmodem.py εξ' ου και το:

```
from gsmmodem import modem
```

Υλοποιεί την συνάρτηση sendSms οποία στέλνει στο modem AT εντολές μέσω της sendCommand() :

```
gsm = modem(dev)

command = 'ATZ'
gsm.sendCommand(command)
```

Αναλυτικά οι AT εντολές που πρέπει να σταλούν στο modem για να στείλει ένα sms :

ATZ : αρχικοποίηση με βάση προεπιλεγμένες ρυθμίσεις(default settings)

AT+CMGF=1 : επιλογή μορφής μυνήματος (κείμενο)

AT+CMGS=' + chr(34) + phone_no + chr(34) : στείλε το μύνημα στον συγκεκριμένο αριθμό τηλεφώνου (τα chr(34) είναι ο χαρακτήρας “ σε ascii μορφή)

chr(26) : ειδικός χαρακτήρας τερματισμού, substitude σε ascii, κοινώς Ctrl + z

και τέλος σταματάμε την σειριακή μας επικοινωνία :

```
gsm.close()
```

recvsms.py

Βασίζεται στο gsmmodem.py αλλά εισάγει και μια βιβλιοθήκη για τον χειρισμό strings :

```
from gsmmodem import modem
import string
```

στέλνει τις εξής AT εντολές για την παραλαβή όλων των μυνημάτων από την μνήμη sim (μέσω της sendCommand()) :

AT : αρχικοποίηση του modem

AT+CMGF=1 : επιλογή μορφής μυνήματος (κείμενο)

AT+CMGL=\"ALL\" : φέρε όλα τα μυνήματα

και επιστρέφει μια λίστα με δυο στοιχεία, το πρώτο εκ των οποίων είναι το τελευταίο μύνημα που παραλήφθηκε και το δεύτερο το id του μυνήματος στην μνήμη sim. Σε περίπτωση που δεν υπάρχει sms επιστρέφει μια κενή λίστα.

delsms.py

Αυτό το module βασίζεται στο gsmmodem.py και ουσιαστικά διαγράφει από την μνήμη sim το μήνυμα με κάποιο συγκεκριμένο id, το οποίο περνιέται σαν παράμετρος.

Υπεύθυνη AT εντολή για το σβήσιμο sms είναι η:

AT+CMGD=x : όπου x το id του μυνήματος.

Ο λόγος για τον οποίο σβήνουμε το μήνυμα από την μνήμη γίνεται για λόγους σωστής διαχείρισης. Από το να υλοποιήσουμε έναν ελεγκτικό μηχανισμό που να κοιτάει την πληρότητα της μνήμης sim και να σβήνει μηνύματα μετά από κάποιο χρονικό διάστημα, σβήνουμε το μήνυμα ακριβώς μετά την παραλαβή και επεξεργασία του. Έτσι γλιτώνουμε κώδικα, χρόνο και λάθη.

sms_format.py

Αυτό το module ουσιαστικά δημιουργεί το sms το οποίο θα φτάσει στον χρήστη. Δέχεται δυο παραμέτρους: την attributes και την dictionary. Η attributes είναι μια λίστα που περιέχει την ονομασία του χαρακτηριστικού που μας ενδιαφέρει (π.χ. θερμοκρασία, εφόσον έχει περάσει κάποιο όριο η τιμή της) και η dictionary είναι ένα λεξικό (built-in python type) το οποίο περιέχει όλα τα ζευγάρια χαρακτηριστικών-τιμών. Μετά από κατάλληλη επεξεργασία, δημιουργείται ενα ευανάγνωστο μύνημα που περιέχει ποιες τιμές έχουν τυχόν ξεπεράσει τα δοθέντα όρια, καθώς και τις τρέχουσες τιμές όλων των χαρακτηριστικών. Όλα αυτά γίνονται μέσω της συνάρτησης sms_report(attributes,dictionary).

Εδώ υλοποιείται επίσης η συνάρτηση sms_error() η οποία επιστρέφει απλά ένα μήνυμα λάθους στον χρήστη σε περίπτωση που το σύστημα έλαβε από τον χρήστη κάποιο μήνυμα (εντολή) η οποία ήταν συντακτικά λάθος, προτρέποντάς τον να ξαναστείλει την εντολή αλλά σωστά αυτήν την φορά.

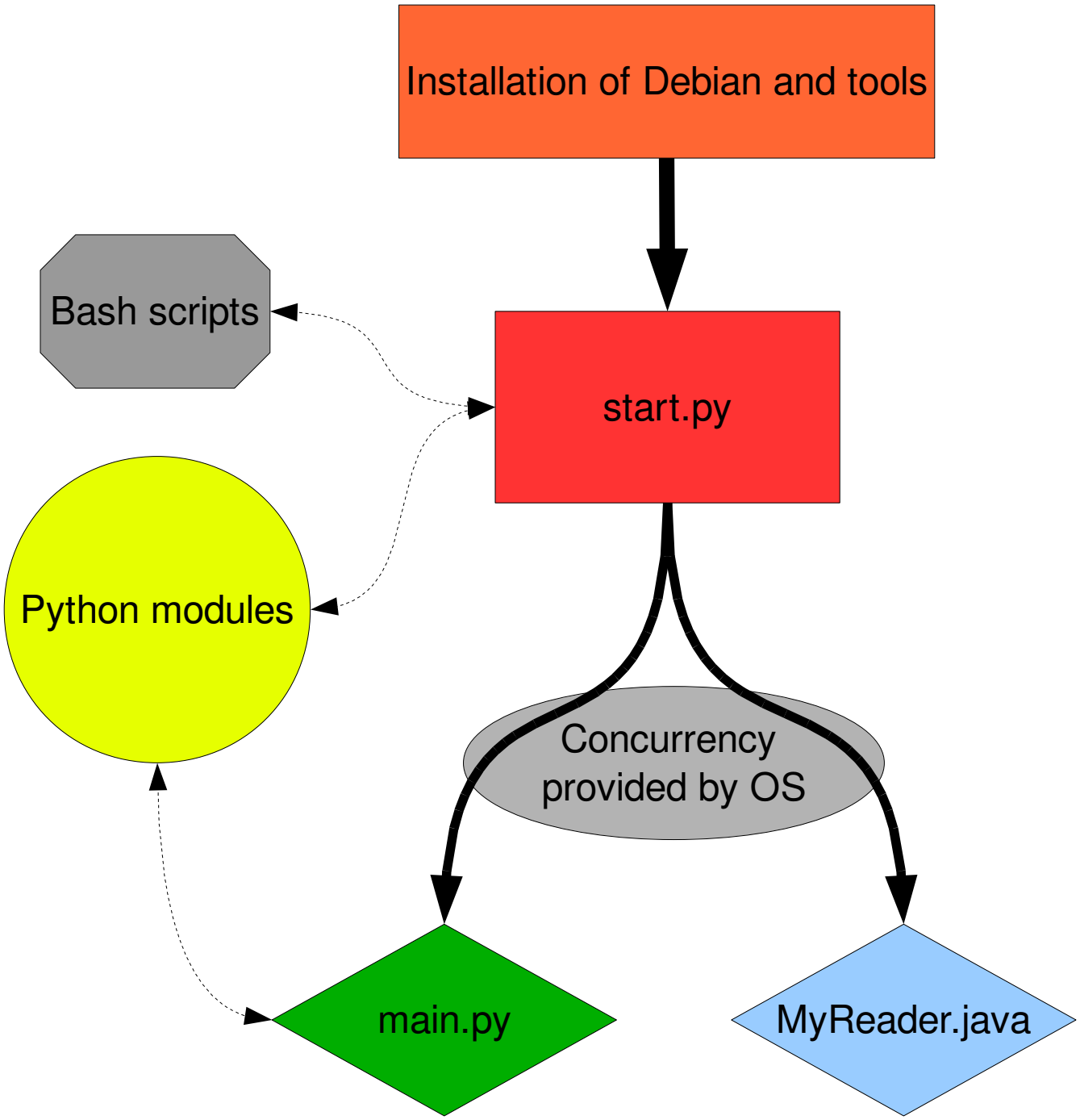
"Received unknown instruction! Please refer to the manual and sent an sms again".

7.
ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

6.1 Οργάνωση

Ο κώδικας της κύριας εφαρμογής που “τρέχει” το σύστημά μας είναι κατά το μεγαλύτερο ποσοστό γραμμένος σε python. Υπάρχουν κάποια bash scripts που εκτελούνται κατά την αρχικοποίηση του συστήματος, καθώς και μια τροποποιημένη έκδοση ενός java προγράμματος που υπάρχει ήδη ως εργαλείο του TinyOS.

Στην πραγματικότητα, η ροή της εκτέλεσης του κώδικα είναι η εξής: Μετά την εγκατάσταση του λειτουργικού και όλων των απαιτούμενων εργαλείων, τρέχουμε το αρχείο start.py. Αυτό το αρχείο κάνει κάποιες κατάλληλες αρχικοποιήσεις στο σύστημα, αλληλεπιδρώντας και με τον χρήστη. Στην πορεία καλώνται κάποια bash scripts για να πάρουμε και να επεξεργαστούμε την “είσοδο” από τον χρήστη. Στο τέλος και εφόσον όλα είναι έτοιμα για να “τρέξουν”, το start.py δίνει την σκυτάλη σε δυο άλλα προγράμματα, τα οποία τρέχουν ταυτόχρονα, το Myreader.java και το main.py. Το Myreader.java διαβάζει τα δεδομένα από τον σταθμό βάσης, τα τροποποιεί αναλόγως και τα αποθηκεύει στην βάση δεδομένων. Το main.py κάνει όλα τα υπόλοιπα. Είναι ουσιαστικά το κυρίως πρόγραμμα που τρέχει σε ατέρμονα βρόγχο και ελέγχει την συμπεριφορά του συστήματος.


Installation of Debian and tools
Bash scripts
start.py
Python modules
Concurrency
provided by OS
main.py
MyReader.java

Όλος ο κώδικας βρίσκεται μέσα στον φάκελο /opt/tinyos-2.0.2/apps/Sensing. Στον φάκελο python βρίσκονται όλα τα modules της εφαρμογής, οργανωμένα σε φακέλους, καθώς και τα start.py, main.py. Δηλαδή:

/opt/tinyos-2.0.2/apps/Sensing/python

start.py
main.py

/db
sql.py
backup.py

/misc
parser.py
scenario.py
scheduler.py

/scripts
get_phone.sh
get_report_time.sh
get_threshold.sh
get_loop_time.sh
params.txt

/sms
delsms.py
gsmmodem.py
recvsms.py
sendsms.py
sms_format.py

Επίσης στον φάκελο /opt/tinyos-2.0.2/support/sdk/java/net/tinyos/ tools βρίσκεται το αρχείο MyReader.java, που είναι μια παραλλαγή του MsgReader.java.

Θα εξηγήσουμε συνοπτικά στη συνέχεια τι κάνει κάθε αρχείο-πρόγραμμα από τα παραπάνω (εκτός από το /sms που το είδαμε στο προηγούμενο κεφάλαιο).

6.2 Περιγραφή

start.py

Το start.py όπως προαναφέραμε είναι η αφετηρία της εφαρμογής. Η χρήση python και bash scripting μας γλιτώνει από κόπο και χρόνο, αυτοματοποιώντας την διαδικασία προεργασίας για το τρέξιμο της εφαρμογής. Έχει ένα πρωτόγονο γραφικό interface που βασίζεται στην βιβλιοθήκη curses της python. Κάνει import τα εξής:

```
import os
from subprocess import Popen, PIPE
import curses.wrapper
from misc.parser import extract_date
```

Τα os και subprocess modules είναι απαραίτητα για την συνεργασία του κώδικα python με εντολές του λειτουργικού συστήματος (μπορούμε δηλαδή να τρέξουμε εντολές shell μέσα από κώδικα python, και να έχουμε απόλυτο έλεγχο της κατάστασης που αυτές επιφέρουν). Το curses.wrapper είναι ένα wrapper module για το curses, το οποίο είναι υπεύθυνο για το στοιχειώδες γραφικό περιβάλλον, και είναι σχεδιασμένο ώστε να αποκρύπτει τις τυχόν ασυνέπειες / λάθη που προκύπτουν από την όχι και τόσο καλή (documented) χρήση του curses module. Εισάγουμε επίσης και μια συνάρτηση extract_date από ένα module που γράψαμε και θα παρουσιάσουμε παρακάτω.

Το start.py αναλαμβάνει τα εξής:

- Αλλάζει τα permissions σε φακέλους και αρχεία του συστήματος, καθώς αυτό είναι απαραίτητο για την σωστή λειτουργία του TinyOS και όχι μόνο.
- Προτρέπει τον χρήστη να τοποθετήσει στο σύστημα το modem και τον αισθητήρα-σταθμό βάσης, βρίσκει που αντιστοιχούν οι συσκευές αυτές σε επίπεδο λειτουργικού συστήματος και αλλάζει τα permissions ώστε να έχουμε πλήρη έλεγχο.
- Καθορίζει το σενάριο εκτέλεσης του συστήματος, μέσα από ένα απλό γραφικό περιβάλλον, με πλήρη καθοδήγηση.
- Ξεκινά τα MyReader.java και main.py με τις σωστές παραμέτρους, θέτωντας το σύστημα σε λειτουργία.

Μια ματιά στον πηγαίο κώδικα του κεφαλαίου 11 και στα σχόλια που υπάρχουν εκεί μπορούν να αποκαλύψουν περισσότερα.

main.py

Το main.py ουσιαστικά είναι το κυρίως πρόγραμμα. Τρέχει σε ατέρμονα βρόγχο και εκτελεί το τρέχων σενάριο λειτουργίας, ελέγχωντας παράλληλα την λήψη κάποιου μηνύματος και αλλάζοντας το σενάριο εφόσον πληρούνται οι προδιαγραφές. Κάνει import τα εξής:

```
import sys
import time
from db.sql import *
from misc.scenario import *
from sms import *
from misc.scheduler import *
from misc.parser import remove_white
```

Το sys module χρειάζεται για την λήψη παραμέτρων κατά την κλήση του προγράμματος. Το time module χρειάζεται για τον εσωτερικό “χρονοπρογραμματισμό”, ή αν κάτι τέτοιο ακούγεται βαρυσήμαντο, για να μπορεί το πρόγραμμα να αδρανοποιείται (sleep) εφόσον το σενάριο προϋποθέτει κάτι τέτοιο. Τα υπόλοιπα modules θα τα αναλύσουμε αργότερα.

Το main.py ξεκινάει επεξεργαζόμενο τις παραμέτρους με τις οποίες εκλήθη. Αρχικοποιεί την βάση δεδομένων, και εφόσον οι παράμετροι προδιαγράφουν ένα σωστό σενάριο λειτουργίας, εκτελεί το αντίστοιχο σενάριο (βλ. Κεφάλαιο 3). Εκεί εκτελεί όσους υπολογισμούς χρειάζονται, ενημερώνει τον χρήστη στέλνοντας κατάλληλο sms όταν αυτό πρέπει να γίνει, ελέγχει αν έχει έρθει κάποιο καινούργιο μύνημα και αλλάζει δυναμικά το σενάριο λειτουργίας.

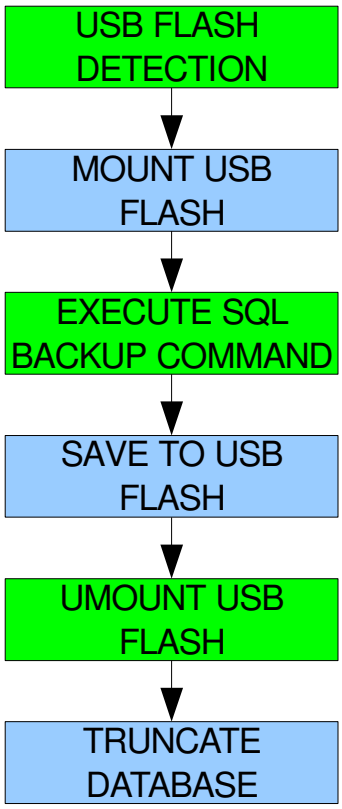
sql.py

Αυτό το module παρέχει συναρτήσεις για εκτέλεση πράξεων στην mysql βάση δεδομένων μας. Η βασική συνάρτηση η οποία χρησιμοποιούμε είναι η fetch_last(), η οποια και επιστρέφει την τελευταία μέτρηση.

backup.py

Εδώ παρέχεται η συνάρτηση που είναι απαραίτητη για την αποθήκευση του σχήματος της βάσης δεδομένων μας σε κάποιο usb flash (backup). Με την χρήση των os και subprocess modules, όπως στο start.py, επεμβαίνουμε στο λειτουργικό σύστημα και εντοπίζουμε πότε κάποιο usb flash drive συνδέεται στο

σύστημά μας. Έπειτα εκτελούμε όλες τις λειτουργίες που χρειάζονται για να γίνει το backup. (εικ. 7.1)



εικ. 7.1

parser.py

Αυτό το module παρέχει δυο συναρτήσεις οι οποίες είναι απαραίτητες για την επεξεργασία του κειμένου του μηνύματος που στέλνει ο χρήστης στο σύστημα, την remove_white και την extract_date. Η πρώτη αφαιρεί από την συμβολοσειρά-μήνυμα τα περιττά κενά, δηλαδή επιστρέφει την συμβολοσειρά με ένα κενό (whitespace) ανάμεσα στις λέξεις. Αυτό γίνεται για να παρέχουμε μια ανοχή σε λάθη που μπορεί να γίνουν κατά το γράψιμο ενός μυνήματος , καθώς 2 κενά μεταξύ των λέξεων είναι και συχνό και δυσδιάκριτο. Η δεύτερη συνάρτηση κάνει χρήση regular expressions για να αποφανθεί αν μια ημερομηνία είναι σε σωστή μορφή (format). Δηλαδή δοθείσης μιας ημερομηνίας σε μορφή συμβολοσειράς xx/yy/zz aa:bb αποφαίνεται αν είναι σε έγκυρη μορφή για μετέπειτα χρησιμοποίηση από άλλες συναρτήσεις, και επιστρέφει την ημερομηνία. Υπάρχει και εδώ μια ανοχή σε λάθη για προφανείς λόγους αποφυγής αυστηρότητας. Για παράδειγμα μια ημερομηνία της μορφής 1/12/9 1:2 σωστά γίνεται αποδεκτή και αντιστοιχεί στην ορθότερη συντακτικά ημερομηνία 01/12/09 01:02.

scenario.py

Ένα πολύ σημαντικό module είναι το scenario.py. Παρέχει στην εφαρμογή κρίσιμες συναρτήσεις οι οποίες επεξεργάζονται την είσοδο της εφαρμογής, ή αλλιώς τις παραμέτρους που δίνονται στο πρόγραμμα. Αυτές οι συναρτήσεις είναι η modem_device, η set_phone, η scenario_mapping και η dict_scenario.

Η modem_device και η set_phone ψάχνουν στις παραμέτρους λέξεις κλειδιά (“-comm” και “-phone”) και επιστρέφουν τις αντίστοιχες τιμές που τα συνοδεύουν, δηλαδή το που αντιστοιχεί σε επίπεδο λειτουργικού το modem μας και ποιο είναι το τηλέφωνο που θα χρησιμοποιηθεί για αποστολή των μηνυμάτων από την μεριά του συστήματος.

Η scenario_mapping “σαρώνει” τις παραμέτρους και επιστρέφει μια λίστα με το συγκεκριμένο σενάριο λειτουργίας. Αυτή η λίστα χρησιμοποιείται από το κυρίως πρόγραμμα main.py για να καθοριστεί η λειτουργικότητα. Οι παράμετροι που ελέγχονται είναι οι πιθανές τιμές κατωφλίου που δεν πρέπει να ξεπεραστούν (και προφανώς σε πιο χαρακτηριστικό αντιστοιχούν), η ώρα που πρέπει να ειδοποιηθεί ο χρήστης στο μέλλον, και προφανώς, το σενάριο αυτό καθ'αυτό. Γίνονται όλοι οι απαραίτητοι έλεγχοι ώστε η λίστα σεναρίου που θα επιστραφεί να είναι σε σωστή μορφή και να υλοποιεί ακριβώς την λειτουργικότητα που επιθυμούμε.

Τέλος, η συνάρτηση dict_scenario επιστρέφει την λίστα σεναρίου ως λεξικό (built-in python type), σε ζεύγος χαρακτηριστικών και τιμών. Αυτό γίνεται για πιο εύκολη προγραμματιστική διαχείριση από άλλες συναρτήσεις.

scheduler.py

Αυτό το module παρέχει την συνάρτηση calcRemainingTime() η οποία κάνει χρήση της βιβλιοθήκης datetime και υπολογίζει τον υπολοιπόμενο χρόνο σε δευτερόλεπτα από μια δοθείσα ημερομηνία και ώρα της μορφής "21/11/06 16:30". Έτσι μπορούμε να υλοποιήσουμε την λειτουργικότητα του σεναρίου ειδοποίησης σε κάποια μελλοντική στιγμή, π.χ. να ειδοποιηθούμε σε 20 λεπτά για τις συνθήκες που μας ενδιαφέρουν. Προφανώς η συνάρτηση επιστρέφει κωδικό λάθους προς ανάλογη επεξεργασία εφόσον η ημερομηνία ανήκει στο παρελθόν.

scripts

Τα bash scripts που υπάρχουν στον φάκελο /scripts είναι υποτυπώδη και

απλά λαμβάνουν είσοδο από τον χρήστη, πραγματοποιώντας και κάποιο απλό έλεγχο. Αποθηκεύουν την είσοδο στο αρχείο params.txt, από όπου το start.py τα παίρνει για επεξεργασία. Τα scripts είναι τρία : get_phone.sh, get_report_time.sh, get_threshold.sh, get_report_loop.sh. Το πρώτο προτρέπει τον χρήστη να υποδείξει το νούμερο του τηλεφώνου όπου θα αποστέλλονται τα sms από το σύστημα, το δεύτερο την μελλοντική ημερομηνία που θέλει να ειδοποιηθεί ο χρήστης (εφόσον για σενάριο εκκίνησης επιλεγεί κάτι τέτοιο), το τρίτο τις τιμές κατωφλίου για τα χαρακτηριστικά που μας ενδιαφέρουν (πάντα με την εκκίνηση του αντίστοιχου σεναρίου) και το τελευταίο τον χρόνο περιοδικής ειδοποίησης (σενάριο Δ).

MyReader.java

Το πρόγραμμα MyReader.java είναι μια παραλλαγή του MsgReader.java, το οποίο είναι ένα έτοιμο εργαλείο που έρχεται μαζί με το TinyOS. Ουσιαστικά μιλάμε για έναν packet sniffer, ο οποίος πηγαίνει πακέτο με την εφαρμογή BaseStation, και σκοπό έχει την παρουσίαση στον χρήστη των πακέτων που λαμβάνονται από τον σταθμό βάσης. Π.χ.

```
1152232617609: Message
  [nodeid=0x2]
  [counter=0x1049]
```

Εδώ για παράδειγμα έχουμε ένα μήνυμα όπου υποδηλώνεται ο κόμβος (το id του αισθητήρα) και ο μετρητής του μηνύματος. Στα μηνύματα της εφαρμογής μας έχουμε και τις μεταβλητές par, tsr, temp και hum, που μετράνε αντίστοιχα την ενεργή φωτοσυνθετική ακτινοβολία, την συνολική ηλιακή ακτινοβολία, την θερμοκρασία και την υγρασία. Επειδή όμως αυτές οι τιμές είναι σε δεκαεξαδική και μη κανονικοποιημένη μορφή, το εργαλείο MyReader.java αναλαμβάνει την κατάλληλη επεξεργασία τους. Επίσης είναι υπεύθυνο για την αποθήκευση των τροποποιημένων τιμών στην βάση δεδομένων, για την μετέπειτα λήψη τους από την κυρίως εφαρμογή. Αυτό γίνεται με την χρήση του jdbc API, το οποίο είναι υπεύθυνο για την εκτέλεση πράξεων σε μια βάση δεδομένων.

Όλος ο πηγαίος κώδικας του MyReader.java με εκτενή σχόλια είναι διαθέσιμος στο κεφάλαιο 11.

8.
ΟΔΗΓΟΣ ΧΡΗΣΗΣ

8.1 Εγχειρίδιο εντολών

Σε αυτό το κεφάλαιο παραθέτουμε έναν οδηγό χρήσης ο οποίος καλύπτει όλα όσα πρέπει να ξέρει κάποιος για να χειριστεί το σύστημά μας. Για την ακρίβεια, εδώ αναλύουμε την μορφή των εντολών που μπορεί να στείλει κάποιος από το κινητό του. Οι εντολές ξεκινούν πάντα με:

-rep
ή
-repat
ή
-reploop

Ας εξετάσουμε την πρώτη: με την εντολή *-rep* εκτελούμε πάντα το σενάριο Α. Η *-rep* ακολουθείται πάντα από μία ή περισσότερες εντολές του τύπου (χωρισμένες με κενά):

-temp x
-hum x
-psr x
-tsr x

όπου x κάποια αριθμητική τιμή, π.χ. 30. Οι παραπάνω εντολές απλά καθορίζουν τις τιμές κατωφλίου των αντίστοιχων χαρακτηριστικών. Για παράδειγμα, αν στείλουμε :

-rep -temp 32

λέμε στο σύστημα να μας ειδοποιήσει με sms όταν και αν η τιμή της θερμοκρασίας υπερβεί τους 32 βαθμούς Κελσίου. Επίσης αν στείλουμε:

-rep -hum 21 -par 12 -tsr 45

λέμε στο σύστημα να μας ειδοποιήσει με sms όταν και αν η υγρασία υπερβεί την τιμή 21 **ή** η ενεργή φωτοσυνθετική ακτινοβολία υπερβεί την τιμή 12 **ή** η συνολική ηλιακή ακτινοβολία υπερβεί την τιμή 45.

Με την εντολή *-repat* πηγαίνουμε στα σενάρια Β και Γ. Αν στείλουμε μόνο *-repat*, πυροδοτούμε το σενάριο Β, όπου και δεν ειδοποιούμαστε με sms μέχρι να ξαναστείλουμε sms με κάποια άλλη εντολή.

Την εντολή *-repat* μπορεί να την ακολουθήσει η εντολή

-time day/month/year hour:minute

όπου με *day/month/year hour:minute* καθορίζουμε την ακριβή μελλοντική ημερομηνία που θέλουμε να ειδοποιηθούμε (σενάριο Γ). Για παράδειγμα:

-repat -time 22/01/10 13:00

σημαίνει πως το σύστημα θα μας ειδοποιήσει για τις τιμές που επικρατούν στις 22/01/10 13:00, **και μετά θα εκτελέσει το σενάριο Β**.

Αν αντί του -time *day/month/year hour:minute* δώσουμε την εντολή -now, ειδοποιούμαστε άμεσα. Δηλαδή:

-repat -now

σημαίνει πώς το σύστημα μας στένει άμεσα sms με τις μετρήσεις, **και στη συνέχεια εκτελεί το σενάριο Β**

Με την εντολή -reploop εκτελούμε το σενάριο Δ. Μετά την εντολή -reploop ακολουθούν οι ίδιες εντολές της εντολής -repat, δηλαδή:

-time *day/month/year hour:minute*

ή

-now

ακολουθούμενες **πάντα** από την εντολή:

-loop minutes

όπου ορίζουμε τον χρόνο επαναλαμβανόμενης ειδοποίησης σε λεπτά. Παραδείγματα:

-reploop -time 22/01/10 13:00 -loop 3

σημαίνει πως το σύστημα θα μας ειδοποιήσει στις *22/01/10 13:00* για τις συνθήκες που επικρατούν και έπειτα κάθε **3 λεπτά** θα μας ειδοποιεί παρομοίως.

-reploop -now -loop 60

σημαίνει ότι το σύστημα μας ειδοποιεί άμεσα και έπειτα μας ειδοποιεί κάθε **60 λεπτά**.

Παραπάνω από ένα κενά μεταξύ των παραπάνω λέξεων κλειδιών καθώς και μη “ορθή” αναπαράσταση της ημερομηνίας (π.χ. αν αντί για 01:03 γράψουμε 1:3) συγχωρούνται. Συντακτικά λάθη και παρελθοντικές ημερομηνίες απλά **δεν θα αλλάξουν το σενάριο**, και θα λάβουμε ένα sms το οποίο θα περιέχει:

"Received unknown instruction! Please refer to the manual and sent an sms again".

Οπότε θα πρέπει να ξαναστείλουμε ένα sms-εντολή χωρίς κάποιο από τα παραπάνω λάθη.

8.1 Λειτουργία backup

Μια πολύ σημαντική και χρήσιμη λειτουργία που παρέχει το σύστημά μας είναι η αποθήκευση/σώσιμο (backup) της βάσης δεδομένων μας, με όλες τις τιμές των μετρήσεων που έχει μέχρι εκείνη την στιγμή. Αυτό μπορεί να είναι θεμιτό για λόγους εξοικονόμησης χώρου στον δίσκο του συστήματος ή για πιο διεξοδική ανάλυση όλων των δεδομένων μέχρι στιγμής (π.χ. χρήση στατιστικής στο σύνολο των δεδομένων).

Για να γίνει αυτό αρκεί να έχουμε ένα δίσκο usb flash, και να το συνδέσουμε

με την θύρα usb του συστήματος. Όλα τα υπόλοιπα τα αναλαμβάνει ο κώδικας (μπορούμε να δούμε τι γίνεται και πότε τελειώνει η διαδικασία από την κονσόλα που βλέπουμε την έξοδο του συστήματος).

Ως αποτέλεσμα θα έχουμε ένα αρχείο .sql αποθηκευμένο στο usb flash (το οποίο μπορούμε να το κάνουμε import οπουδήποτε αλλού αρκεί να “υπάρχει” mysql) και μια άδεια βάση δεδομένων (truncated) στο σύστημά μας.

9.
ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΕΚΤΑΣΕΙΣ

9.1 Περί επεκτασιμότητας του συστήματος

Το σύστημά μας επιτελεί παρακολούθηση σε συγκεκριμένα περιβαλλοντικά χαρακτηριστικά και έχει μηχανισμό ειδοποίησης με sms, καθώς και λήψη οδηγιών από αυτά. Όμως, λόγω νέων τεχνολογιών καθώς και μη αξιοποίησης κάποιων λειτουργιών-υπηρεσιών που υπάρχουν ήδη, θα μπορούσαμε να προσθέσουμε πολλά ακόμα στοιχεία που θα εμπλούτιζαν την λειτουργικότητα σε μεγάλο βαθμό. Κάποια από αυτά είναι:

- Σύνδεση του συστήματος με τον παγκόσμιο ιστό (για παράδειγμα με mobile internet) και ανάπτυξη μιας εφαρμογής (web interface) για πλήρη απομακρυσμένη πρόσβαση σε πραγματικό χρόνο και έλεγχο κάθε περαιτέρω λειτουργίας.

- Περισσότερο έλεγχο σε στοιχεία λειτουργίας, όπως πρωτόκολλα αλλαγής συμπεριφοράς του δικτύου αισθητήρων, περισσότερο “βάθος” σε πράξεις στην βάση δεδομένων, περισσότερες επιλογές αλ έλεγχος αυτών. Οι δυνατότητες είναι μάλλον άπειρες και αφήνονται εν μέρει στην φαντασία.ληλεπίδρασης με τον χρήστη κ.α.

- Προσθήκη στοιχείων αυτοματισμού και για άλλες λειτουργίες που επαφύονται των (γεωργικών και όχι μόνο) εφαρμογών, και πλήρης έλεγχος αυτών. Οι δυνατότητες είναι μάλλον άπειρες και αφήνονται εν μέρει στην φαντασία.

9.2 Συμπεράσματα

Η εργασία αυτή περιγράφει ένα σύστημα που μπορεί να χρησιμοποιηθεί σε γεωργικές εφαρμογές. Για παράδειγμα στον έλεγχο των τιμών κάποιων χαρακτηριστικών σε ένα θερμοκήπιο, μικρής ή μεγάλης κλίμακας και σε εξωτερικούς χώρους. Προφανώς, και όχι μόνο. Η παρακολούθηση περιβαλλοντικών αλλαγών και η δυνατότητα τηλε-ειδοποίησης μπορούν, πέρα από γεωργικές εφαρμογές, να εφαρμοστούν σε

- Μικρούς μετεωρολογικούς σταθμούς
- Περιβαλλοντική έρευνα

και γενικότερα σε κάθε είδους εφαρμογή, από παραγωγή προϊόντων ειδικών προδιαγραφών μέχρι ιδιωτική χρήση, αρκεί να υπάρχει γενικότερα η ανάγκη και η απαίτηση μέτρησης μεγεθών και χαρακτηριστικών όπως αυτά για τα οποία κατασκευάστηκε να παρακολουθεί το σύστημα.

10.
ΒΙΒΛΙΟΓΡΑΦΙΑ

Εδώ παραθέτουμε την βιβλιογραφία από την οποία αντλήθηκε γνώση για την εκπόνηση της εργασίας:

- http://docs.tinyos.net/index.php/
- http://wiki.soekris.info/Installing_Debian_Linux_5.0
- http://lover2linux.blogspot.com/2009/07/how-to-install-tinyos-in.html
- http://designbuildtestrepeat.wordpress.com/2008/06/26/sms-over-3g-and-bluetooth-from-python/
- http://docs.python.org/
- http://www.kitebird.com/articles/jdbc.html
- http://wiki.ubuntu.org.cn/UbuntuWiki:PXEInstallServer
- http://www.kitebird.com/articles/pydbapi.html
- https://help.ubuntu.com/community/SerialConsoleHowto
- http://www.teltonika.lt

11.
ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ

Σε αυτό το κεφάλαιο παραθέτουμε τον πηγαίο κώδικα της εφαρμογής.

/opt/tinyos-2.0.2/apps/Sensing/python/**start.py**

```
from os import system
import os
from subprocess import Popen, PIPE
import curses.wrapper
from misc.parser import extract_date
import time

def execute_cmd(cmd_string):
    system("clear")
    a = system(cmd_string)
    print ""
    if a == 0:
        print "Command executed correctly"
    else:
        print "Command terminated with error"
    print ""

def execute_cmd_enter(cmd_string, show = False):
    system("clear")
    a = system(cmd_string)
    print ""
    if a == 0 and show:
        print "Command %s executed correctly" %cmd_string

    if a!=0:
        print "Command terminated with error"
    raw_input("Press enter")
    print ""


def get_param(prompt_string):

    input = s.getstr(10, 10, 60)
    return input
```

```
def mapping(cur_line):

    a = 1

    while cur_line > 4:
        a +=1
        cur_line -= 2
    return a


def main(screen):

    height = 24
    width = 80
    start_x = 0
    start_y = 0

    x=0
    line = 4

    parameters = None # parameters to call the main.py , initiated to None



    curses.endwin()
        # The commands below are pure dark art
    # find the current user of the system and redirect to stdout
    (stdout, stderr) = Popen(["who"], stdout=PIPE).communicate()
    curses.endwin()
    # trick to filter the word
    first_gap = stdout.find(" ")
    # which is the string before the first gap
    USER = stdout[:first_gap]

    # GREET THE USER THAT DARED TO DIVE INTO GRAPHICLESS ENVIRONMENT IN YEAR 2010
    execute_cmd_enter("echo \"%s WELCOME TO OUR SYSTEM SETUP PROCESS. PLEASE INSPECT CAREFULLY
THE MESSAGES THAT POP OUT IN THIS STEP BY STEP INSTALLATION SCRIPT.\""%USER)
    curses.endwin()
    # First of all change owner of /opt folder and its subfolders
    execute_cmd_enter("echo \"STEP 1) CHANGING PERMISSIONS ON /opt FOLDER AND ITS SUBFOLDERS, AS
WELL AS MAKING EXECUTABLES SOME ESSENTIAL BASH SCRIPTS ---- YOU WILL BE PROMPTED FOR
ROOT PASSWORD\"")
    curses.endwin()
    execute_cmd_enter("sudo chown -R %s /opt" % USER, True)
    curses.endwin()
    execute_cmd_enter("sudo chmod 777 -R /opt/tinyos-2.0.2/apps/Sensing/python/scripts/", True)
    curses.endwin()
    (stdout, stderr) = Popen(["ls /dev | grep ttyUSB"], stdout=PIPE,shell=True).communicate()
    curses.endwin()

    present_usb_devs = stdout.split("\n") # find the system currents usb devices

    execute_cmd_enter("echo \"STEP 4) PLEASE INSERT THE MODEM IN A USB PORT.\"")
    curses.endwin()
```

```
    # The commands below are pure dark art
  ok = True
  while True:
    (stdout, stderr) = Popen(["ls /dev | grep ttyUSB"], stdout=PIPE,shell=True).communicate()
        curses.endwin()
    try:
      MODEM = "/dev/%s" % [val for val in stdout.split("\n") if val not in present_usb_devs][0] # magic python
          ok = True
    except IndexError:
      execute_cmd_enter("echo \"STEP 4) PLEASE INSERT THE MODEM IN A USB PORT.\"")
      curses.endwin()
          ok = False
          time.sleep(2)
        if ok == True:
          break

  present_usb_devs = stdout.split("\n") # find the system currents usb devices

  execute_cmd_enter("echo \"%s AS YOUR MODEM. THIS WILL NOW BE CHMOD-ED AND DECLARED LATER
AS ENVIROMENTAL VARIABLE MODEM\""%MODEM)
  curses.endwin()
  execute_cmd_enter("sudo chmod 777 %s"%MODEM, True)
  curses.endwin()
  curses.endwin()

  # Now mount the sensor with BaseStation into a usb port, only if not present
  execute_cmd_enter("echo \"STEP 2) PLEASE INSERT THE SENSOR THAT RUNS THE BASESTATION
APPLICATION IN A USB PORT.\"")
  curses.endwin()

  ok = True
  while True:
    (stdout, stderr) = Popen(["ls /dev | grep ttyUSB"], stdout=PIPE,shell=True).communicate()
        curses.endwin()
    try:
          BASE_SENSOR = "/dev/%s" % [val for val in stdout.split("\n") if val not in present_usb_devs][0] # magic
python
          ok = True
        except:
          execute_cmd_enter("echo \"STEP 2) PLEASE INSERT THE SENSOR THAT RUNS THE BASESTATION
APPLICATION IN A USB PORT.\"")
        curses.endwin()
          ok = False
          time.sleep(0.5)
        if ok == True:
          break

  present_usb_devs = stdout.split("\n") # find the system currents usb devices

  execute_cmd_enter("echo \"%s AS YOUR BASE_SENSOR. THIS WILL NOW BE CHMOD-ED AND DECLARED
LATER AS ENVIROMENTAL VARIABLE BASE_SENSOR\""%BASE_SENSOR)
  curses.endwin()
  execute_cmd_enter("sudo chmod 777 %s"%BASE_SENSOR, True)
  curses.endwin()
```

```
    while x != ord("q"):

        s = curses.newwin(height,width,start_x,start_y)
        s.keypad(1)

        xx=2 # initialize the position in x axis
        s.border(0)
        s.addstr(xx, 1, "ADMINISTRATION PANEL (q or ESC for quit)")
        xx+=1
        s.addstr(xx, 1, "---------------------------------")
        xx+=1
        s.addstr(xx, 6, "1- SYSTEM CONFIGURATION")
        xx+=1
        s.addstr(xx, 1, "---------------------------------")
        xx+=1
        s.addstr(xx, 6, "2- RUN")
        xx+=1
        s.addstr(xx, 1, "---------------------------------")
        xx+=1
        s.addstr(line,2,"-->")
        s.refresh()
        x = s.getch()

#moveology

        if x == curses.KEY_DOWN:
            if line < xx-2:
                line +=2

        if x == curses.KEY_UP:
            if line > 4:
                line -=2

        if x == 10: #actually the ENTER key
            k = mapping(line)
            kk = str(k)
            x = ord(kk)

        if x == 27:
            x = ord("q")


        if x == ord("1"):

            x = 0
            line = 4

            while x != ord("b"):

                s = curses.newwin(height,width,start_x,start_y)

                s.keypad(1)


                xx=2 # initialize the position in x axis
                s.border(0)
```

```
            s.addstr(xx, 1, "SYSTEM CONFIGURATION , (b or ESC for back)")
            xx+=1
            s.addstr(xx, 1, "-----------------------------------------------------------------")
            xx+=1
            s.addstr(xx, 6, "1- SCENARIO A: SEND SMS WHEN A VALUE EXCEEDS A THRESHOLD")
            xx+=1
            s.addstr(xx, 1, "-----------------------------------------------------------------")
            xx+=1
            s.addstr(xx, 6, "2- SCENARIO B: AWAIT INSTRUCTIONS FROM SMS")
            xx+=1
            s.addstr(xx, 1, "-----------------------------------------------------------------")
            xx+=1
            s.addstr(xx, 6, "3- SCENARIO C: SEND SMS AT GIVEN TIME")
            xx+=1
            s.addstr(xx, 1, "-----------------------------------------------------------------")
                    xx+=1
            s.addstr(xx, 6, "4- SCENARIO D: SEND SMS AT GIVEN TIME AND THEN REPORT EVERY x TIME")
            xx+=1
            s.addstr(xx, 1, "-----------------------------------------------------------------")
            xx+=1
            s.addstr(line,2,"-->")

            s.refresh()
            x = s.getch()

            if x == curses.KEY_DOWN:
               if line < xx-2:
                  line +=2

            if x == curses.KEY_UP:
               if line > 4:
                  line -=2

            if x == 10: #actually the ENTER key
               k = mapping(line)
               kk = str(k)
               x = ord(kk)

            if x == 27:
               line = 4
               x = ord("b")

            if x == ord("1"):


               while True: # until we meet the requirements
                  curses.endwin()
                  execute_cmd_enter("/opt/tinyos-2.0.2/apps/Sensing/python/scripts/get_threshold.sh")
                  curses.endwin()
                  (stdout, stderr) = Popen(["cat","/opt/tinyos-2.0.2/apps/Sensing/python/scripts/params.txt"],
stdout=PIPE).communicate()
                  curses.endwin()
                  # parsing the redirected output from file
                  split_values = stdout.split() # split at whitespace
                  max_temp = split_values[0]
```

```
                max_hum = split_values[1]
                max_psr = split_values[2]
                max_tsr = split_values[3]
                if (max_temp.isdigit() and max_hum.isdigit() and max_psr.isdigit() and max_tsr.isdigit()):
                    curses.endwin()
                    execute_cmd_enter("echo YOU CAN NOW PROCEED AND RUN THE APPLICATION, OR DEFINE
ANOTHER SCENARIO")
                    break # if all the above are numbers, continue
                else:
                    curses.endwin()
                    execute_cmd_enter("echo some values you entered are not numeric, please retry")

            # gather all the above and form the parameters
            parameters = "-rep -temp %s -hum %s -psr %s -tsr %s" % (max_temp,max_hum,max_psr,max_tsr)

        if x == ord("2"):
            parameters = "-repat"
            curses.endwin()
            execute_cmd_enter("echo YOU CAN NOW PROCEED AND RUN THE APPLICATION, OR DEFINE
ANOTHER SCENARIO")

        if x == ord("3"):

            while True: # until we meet the requirements
                curses.endwin()
                execute_cmd_enter("/opt/tinyos-2.0.2/apps/Sensing/python/scripts/get_report_time.sh")
                curses.endwin()
                (stdout, stderr) = Popen(["cat","/opt/tinyos-2.0.2/apps/Sensing/python/scripts/params.txt"],
stdout=PIPE).communicate()
                curses.endwin()
                # parsing the redirected output from file
                            print stdout
                            if stdout[0] == "0":
                                date = "-now"
                            else:
                    date = extract_date(str(stdout)) # check if the date given is in valid format

                if (date != "FORMAT ERROR"):
                    curses.endwin()
                    execute_cmd_enter("echo YOU CAN NOW PROCEED AND RUN THE APPLICATION, OR DEFINE
ANOTHER SCENARIO")
                    break # if all the above are numbers, continue
                else:
                    curses.endwin()
                    execute_cmd_enter("echo the time provided is not in a valid format, please retry")

            # gather all the above and form the parameters
                    if date == "-now":
                parameters = "-repat %s" % date
                    else:
                        parameters = "-repat -time %s" % date

        if x == ord("4"):

            while True: # until we meet the requirements
```

```
                curses.endwin()

                execute_cmd_enter("/opt/tinyos-2.0.2/apps/Sensing/python/scripts/get_report_time.sh")
                curses.endwin()
                (stdout, stderr) = Popen(["cat","/opt/tinyos-2.0.2/apps/Sensing/python/scripts/params.txt"],
stdout=PIPE).communicate()
                curses.endwin()
                # parsing the redirected output from file
                        print stdout
                        if stdout[0] == "0":
                            date = "-now"
                        else:
                   date = extract_date(str(stdout)) # check if the date given is in valid format

                if (date != "FORMAT ERROR"):
                   curses.endwin()
                   execute_cmd_enter("echo PROCEED TO THE NEXT STEP")
                   break # if all the above are numbers, continue
                else:
                   curses.endwin()
                   execute_cmd_enter("echo the time provided is not in a valid format, please retry")

             while True: # until we meet the requirements
                curses.endwin()
                execute_cmd_enter("/opt/tinyos-2.0.2/apps/Sensing/python/scripts/get_loop_time.sh")
                curses.endwin()
                (stdout, stderr) = Popen(["cat","/opt/tinyos-2.0.2/apps/Sensing/python/scripts/params.txt"],
stdout=PIPE).communicate()
                curses.endwin()
                # parsing the redirected output from file

                if (int(stdout[0]) > 0):
                   curses.endwin()
                   execute_cmd_enter("echo YOU CAN NOW PROCEED AND RUN THE APPLICATION, OR DEFINE
ANOTHER SCENARIO")
                   break # if all the above are numbers, continue
                else:
                   curses.endwin()
                   execute_cmd_enter("echo you must enter a value greater than zero, please retry")

             # gather all the above and form the parameters

                    if date == "-now":
                parameters = "-reploop %s -loop %s" % (date, stdout[0])
                    else:
                       parameters = "-reploop -time %s -loop %s" % (date, stdout[0])

          curses.endwin()

      if x == ord("2"):

         if parameters == None:
            curses.endwin()
            execute_cmd_enter("echo \"PLEASE CONFIGURE THE SYSTEM FIRST\"")
```

```
            else:
                curses.endwin()
                execute_cmd_enter("/opt/tinyos-2.0.2/apps/Sensing/python/scripts/get_phone.sh") # take the phone number
                curses.endwin()
                (stdout, stderr) = Popen(["cat","/opt/tinyos-2.0.2/apps/Sensing/python/scripts/params.txt"],
stdout=PIPE).communicate()
                curses.endwin()
                    phone_num = stdout
            # copy the SensingMsg.class into python folder due to "have tried a thousand ways to do it otherwise" but
            # the java command below crashed.
            execute_cmd("cp /opt/tinyos-2.0.2/apps/Sensing/SensingMsg.class /opt/tinyos-2.0.2/apps/Sensing/python")
            #invoke the java packet reader
            curses.endwin()
                    execute_cmd("java net.tinyos.tools.MyReader -comm serial@%s:telos SensingMsg &" %
BASE_SENSOR)
                    execute_cmd("python main.py %s -comm %s -phone %s" % (parameters,MODEM,phone_num))
                    break

    curses.endwin()

curses.wrapper(main)
```

/opt/tinyos-2.0.2/apps/Sensing/python/**main.py**

```
import sys
import time
from datetime import datetime
from db.sql import *
from db.backup import *
from misc.scenario import *
from sms import *
from misc.scheduler import *
from misc.parser import remove_white

class Main:

    def __init__(self):
        # initialize by taking arguments
        self.scenario = scenario_mapping(sys.argv[1:]) # now scenario contains all the arguments passed and defines the ....
execution scenarios
        self.modem_comm = modem_device(sys.argv[1:]) # the physical modem device, passed once
            self.phone_num = set_phone(sys.argv[1:]) # the phone number, passed once
            print "phone" , self.phone_num

    def run(self): # main loop

        while True:

            action = backup_on_mount() # check if a usb stick is attached on our system so to backup our database
                if action == "truncate": # if backup successfull, truncate the sensor table
                    while True: # try to truncate the db
                        self.database = db() # initialize connection to database
```

```
                status = self.database.truncator() # truncate it
                self.database.close() # close the connection
                if status == "truncated":
                        break
                time.sleep(2)


        print "scenario : ", self.scenario
        print

        time.sleep(1)
#################################################################################################################
################################################

        if self.scenario[0] == "BREAK": # case where the program exits due to insufficient arguments
        # actually its a pretty way to handle situations where at REPORT_AT mode the instructions are not in an acceptable format
        # so the scenario_mapping function fails to parse the received sms and says "BREAK"
           print
           print " THE PROGRAM EXITS DUE TO INSUFFICIENT OR WRONG PARAMETERS "
           print

           break
#################################################################################################################
################################################

        if self.scenario[0] == "REPORT_ONLY": # case where we only report when a value is higher than the threshold given as argument


                while True:  # the while here goes for stability, if no measurements is present due to truncate, the program crashes
                   self.database = db() # initialize connection to database
                   measurement = self.database.fetch_last() #measurement has nodeid, par etc....
                   self.database.close() # close the connection
                   if measurement != None:
                        break
             time.sleep(2)

           cur_measurement = dict_scenario(measurement) # dictionary containing the latest measurement
           at_what = [(v, k) for v,k in cur_measurement.items()] # create list from dictionary


           print
           print "this measurement contains : " , cur_measurement
           print
           time.sleep(1)

           SENDSMS = False # flag to know whether or not to send sms in "REPORT_ONLY" scenario
           where = [] # dont know at what attributes

           for i in range(1,5): # 1-5 contains the threshold values passed as parameters, as well as the corresponded in measurement
           # now check if a value from measurement is higher than the corresponded threshold given by scenario
```

```
            if float(self.scenario[i]) < measurement[i] and self.scenario[i] !=0: # if we pass the threshold (a non zero one), send sms

                SENDSMS = True # raise the send sms flag cause we are into the wolfs lair


                for j in range(len(at_what)): # this finds the attribute that corresponds to value/threshold
                    if float(at_what[j][1]) == round(measurement[i],2):
                        where.append(at_what[j][0]) # name your attribute


                print "Sending SMS because %s of %s is greater than %s " % (where[-1], round(measurement[i],2),self.scenario[i])

        print "lista : " ,where

        if SENDSMS: # the long awaited time , send sms

            sms_to_send = sms_format.sms_report(where,cur_measurement) # formulate the sms to be sent
            print "sms to send = " , sms_to_send
            print "sms length = " , len(sms_to_send)
            sendsms.sendSms(self.modem_comm,'%s'% self.phone_num, '%s' % sms_to_send) # send it
            time.sleep(4)
                    self.scenario = [] # init scenario
            self.scenario.append("REPORT_AT") # if we have sent an sms, we are ok, so switch to REPORT AT scenario

        # NOW CHECK IF AN SMS ARRIVES AND SPECIFIES AN INSTRUCTION

        print "AWAITING...."

        a = recvsms.recvSms(self.modem_comm) # its the last sms, and its a string

        cur_scenario = self.scenario # keep the current scenario in case the next is a "BREAK" ... we won't break at this time

        if a != []: # if there is an sms in memory

            print "message : " , a[0]
            print "with id : " , a[1]

            a[0] = remove_white(a[0]) # remove the more than one whitespaces in the message

            new_arg = a[0][0:].split(" ") # so pass the new argument into scenario below but as a list
            print new_arg

            self.scenario = scenario_mapping(new_arg) # and change the scenario

            delsms.delSms(self.modem_comm,a[1])# delete the sms you just received

        if self.scenario[0] == "BREAK": # if the sms was not in acceptable format after "parsed" by scenario_mapping

            # INFORM THE USER BY SENDING SMS THAT THE SMS HE SENT WAS NOT IN ACCEPTABLE FORMAT
            sms_to_send = sms_format.sms_error() # send an error warning sms
            sendsms.sendSms(self.modem_comm,'%s'% self.phone_num, '%s' % sms_to_send) # send it
```

```
                self.scenario = cur_scenario # old good days, aka bring back the old scenario, and continue...

#               for i in a[1]: # delete all sms
#                   delsms.delSms(self.modem_comm,i)
#################################################################################################################
################################################

        if self.scenario[0] == "REPORT_AT" and len(self.scenario)==1 : # case where we await for an sms to come and
change the scenario

            print "AWAITING...."

            a = recvsms.recvSms(self.modem_comm) # its the last sms, and its a string

            cur_scenario = self.scenario # keep the current scenario in case the next is a "BREAK" ... we won't break at this
time

            if a != []: # if there is an sms in memory

                print "message : " , a[0]
                print "with id : " , a[1]

                new_arg = a[0][0:].split(" ") # so pass the new argument into scenario below but as a list
                print new_arg

                a[0] = remove_white(a[0]) # remove the more than one whitespaces in the message

                self.scenario = scenario_mapping(new_arg) # and change the scenario

                delsms.delSms(self.modem_comm,a[1])# delete the sms you just received

            if self.scenario[0] == "BREAK": # if the sms was not in acceptable format after "parsed" by scenario_mapping

                # INFORM THE USER BY SENDING SMS THAT THE SMS HE SENT WAS NOT IN ACCEPTABLE
FORMAT
                sms_to_send = sms_format.sms_error() # send an error warning sms
                sendsms.sendSms(self.modem_comm,'%s'% self.phone_num, '%s' % sms_to_send) # send it

                self.scenario = cur_scenario # old good days, aka bring back the old scenario, and continue...

#               for i in a[1]: # delete all sms
#                   delsms.delSms(self.modem_comm,i)

            time.sleep(4)

#################################################################################################################
################################################

        if self.scenario[0] == "REPORT_AT" and len(self.scenario)>1 : # case where we have to report in a well specified
future time

            print "GONNA SLEEP"
            time.sleep(2)
```

```
                if self.scenario[2] == "0": # report now
                    time_to_sleep = 0 # just a nap
                else:
            time_to_sleep = calcRemainingTime(self.scenario[2]) # calculate time to sleep in seconds

        print "sleeping for : " ,time_to_sleep

        if time_to_sleep == "error": # some error occured in calculation

            # INFORM THE USER BY SENDING SMS THAT SOMETHING WENT WRONG
            sms_to_send = sms_format.sms_error() # send an error warning sms
            sendsms.sendSms(self.modem_comm,'%s'% self.phone_num, '%s' % sms_to_send) # send it
            self.scenario = [] # init scenario
            self.scenario.append("REPORT_AT") # don't forget to keep consistency and flow by setting the scenario to
REPORT_AT
        else: # who doens't love some sleep

            time.sleep(time_to_sleep) # being jealous of the possibilities :-)


                while True:  # the while here goes for stability, if no measurements is present due to truncate, the
program crashes
                    self.database = db() # initialize connection to database
                    measurement = self.database.fetch_last() #measurement has nodeid, par etc....
                    self.database.close() # close the connection
                    if measurement != None:
                        break
            time.sleep(2)

            cur_measurement = dict_scenario(measurement) # dictionary containing the latest measurement
            print "Sending SMS.."

            sms_to_send = sms_format.sms_report(0,cur_measurement) # formulate the sms to be sent, with 0 flag
            print "sms to send = " , sms_to_send
            print "sms length = " , len(sms_to_send)
                    sendsms.sendSms(self.modem_comm,'%s'% self.phone_num, '%s' % sms_to_send) # send it
            time.sleep(2)
            self.scenario = [] # init scenario
            self.scenario.append("REPORT_AT") # don't forget to keep consistency and flow by setting the scenario to
REPORT_AT


#############################################################################################################
################################################

    if self.scenario[0] == "REPORT_LOOP": # case where we have to report at given time intervals

        print "REPORT AT GIVEN INTERNAL TIMES"
        time.sleep(2)

                if self.scenario[2] == "0": # report now
                    time_to_sleep = 0 # just a nap
                else:
            time_to_sleep = calcRemainingTime(self.scenario[2]) # calculate time to sleep in seconds
```

```
            print "sleeping for : " ,time_to_sleep

            if time_to_sleep == "error": # some error occured in calculation

                # INFORM THE USER BY SENDING SMS THAT SOMETHING WENT WRONG
                sms_to_send = sms_format.sms_error() # send an error warning sms
                sendsms.sendSms(self.modem_comm,'%s'% self.phone_num, '%s' % sms_to_send) # send it
                self.scenario = [] # init scenario
                self.scenario.append("REPORT_AT") # don't forget to keep consistency and flow by setting the scenario to
REPORT_AT
            else: # entering disturbed sleep at time intervals

                time.sleep(time_to_sleep) # being jealous of the possibilities :-)



                    interval = int(self.scenario[4])*60 # fetch the loop interval at seconds
                    print "interval is %s and type %s"  % (interval,type(interval))

                    cur_scenario = self.scenario # keep the current scenario in case the next is a "BREAK" ... we won't
break at this time

                    while self.scenario == cur_scenario: # outer while loop, that runs every internal times

                        sec_counter = 0 # a seconds' counter

                        print "OUTER WHILE LOOP"

                        while True:  # the while here goes for stability, if no measurements is present due to truncate, the
program crashes
                          self.database = db() # initialize connection to database
                          measurement = self.database.fetch_last() #measurement has nodeid, par etc....
                          self.database.close() # close the connection
                          if measurement != None:
                              break
                       time.sleep(2)

                    cur_measurement = dict_scenario(measurement) # dictionary containing the latest measurement
                    print "Sending SMS.."

                    sms_to_send = sms_format.sms_report(0,cur_measurement) # formulate the sms to be sent, with 0 flag
                    print "sms to send = " , sms_to_send
                    print "sms length = " , len(sms_to_send)
                         sendsms.sendSms(self.modem_comm,'%s'% self.phone_num, '%s' % sms_to_send) # send it

                        while sec_counter < interval: # while we haven't reached the interval time

                          print "INNER WHILE LOOP"
                          print
                          print "sec_counter is %s"  %sec_counter

                          now = datetime.now() # start the timer

                          sec_counter += 5 # increase it by 5 because we gonna sleep for 5 secs later plus the time for the
calculations below
```

```
                        a = recvsms.recvSms(self.modem_comm) # its the last sms, and its a string

                        if a != []: # if there is an sms in memory

                                print "message : " , a[0]
                                print "with id : " , a[1]

                                new_arg = a[0][0:].split(" ") # so pass the new argument into scenario below but as a list
                                print new_arg

                                a[0] = remove_white(a[0]) # remove the more than one whitespaces in the message

                                self.scenario = scenario_mapping(new_arg) # and change the scenario

                                delsms.delSms(self.modem_comm,a[1])# delete the sms you just received

                        if self.scenario[0] == "BREAK": # if the sms was not in acceptable format after "parsed" by
scenario_mapping

                        # INFORM THE USER BY SENDING SMS THAT THE SMS HE SENT WAS NOT IN
ACCEPTABLE FORMAT
                                sms_to_send = sms_format.sms_error() # send an error warning sms
                                sendsms.sendSms(self.modem_comm,'%s'% self.phone_num, '%s' % sms_to_send) #
send it

                                self.scenario = cur_scenario # old good days, aka bring back the old scenario, and
continue...

                        if self.scenario != cur_scenario: # if a new scenario arrived, exit this while loop
                           break

                        then = datetime.now() # stop the timer

                        duration = "%.2f" % ((then - now).seconds + (then - now).microseconds*0.000001) # make a
string of the difference in seconds

                        time.sleep(5 + float(duration)) # sleep for a while so to increase performance, with a pretty
good accuracy


                    if self.scenario != cur_scenario: # if a new scenario arrived, exit AND this while loop
                        break

mainprog = Main()
mainprog.run()
```

/opt/tinyos-2.0.2/apps/Sensing/python/db/**sql.py**

```
import MySQLdb

class db:

   def __init__(self):
```

```
        self.conn = MySQLdb.connect(host = "localhost",user = "user",passwd = "12345",db = "my_db") #initialize the
connection
        self.cursor = self.conn.cursor ()

    def insert(self, nodeid, par, tsr, temp, humidity, counter): # inserts measurements into db

        self.cursor.execute("""INSERT INTO `sensor` (nodeid, par, tsr, temp, humidity, counter)
                            VALUES (%s, %s, %s, %s, %s, %s)
           """,(nodeid, par, tsr, temp, humidity, counter))

    def fetch_last(self): # fetches the latest measurement

        self.cursor.execute ("SELECT nodeid, par, tsr, temp, humidity, counter FROM sensor ORDER BY result_time DESC
LIMIT 1") # our query
        row = self.cursor.fetchone ()
        return row

    def close(self): # close the database connection

        self.cursor.close ()
        self.conn.close ()
```

/opt/tinyos-2.0.2/apps/Sensing/python/db/**backup.py**

```
from os import system
from subprocess import Popen, PIPE
import time


def backup_on_mount():

    (stdout, stderr) = Popen(["ls /dev | grep sd"], stdout=PIPE,shell=True).communicate()

    usb_stick = [i for i in stdout.split("\n") if len(i)==4] # this should suffice


    if usb_stick != []: # if there is one, mount it

        print " detected usb stick !!!"

        system("clear")

        while True:
            mkdir = system("sudo mkdir /media/usb_stick") # create a temporary directory
                if mkdir == 0: # if command executed succesfully
                    print "created directory /media/usb_stick"
                        break
                else: # retry later
                    time.sleep(2)

        system("clear")
             while True:
                mount = system("sudo mount -t vfat /dev/%s /media/usb_stick" % usb_stick[0]) # mount the usb stick to /media/
usb_stick folder
```

```
        if mount == 0: # if command executed succesfully
           print "usb stick succesfully mounted on /media/usb_stick folder"
                   break
             else: # retry later
                time.sleep(3)



          while True:
        (stdout, stderr) = Popen(["/opt/tinyos-2.0.2/apps/Sensing/python/scripts/backup.sh /media/usb_stick"],
stdout=PIPE,shell=True).communicate()

        if stderr == None: # everything is ok

           system("clear")

               while True:
              b = system("sudo umount /media/usb_stick") # un-mount the usb stick$
              if b == 0: # if command executed succesfully
                 print "Unplug your usb stick, backup completed succesfully!!"
                         break
              else: # retry later
                 time.sleep(4)

           system("clear")

               while True:
              rm = system("sudo rm /dev/%s" % usb_stick[0]) # destroy usb node
                        if rm == 0: # if command executed succesfully
                           break
                        else:# retry later
                           time.sleep(2)

                     system("clear")

               while True:
              rmdir = system("sudo rmdir /media/usb_stick") # delete the temporary /media/usb_stick folder
                        if rmdir == 0: # if command executed succesfully
                           break
                        else:# retry later
                           time.sleep(2)

               break # we are ok

        else:
           time.sleep(5)

          return "truncate"   # and then main.py knows the deed

   else:
      print "no usb sticks detected"
      return "proceed" # and then main.py knows the deed
```

/opt/tinyos-2.0.2/apps/Sensing/python/misc/**parser.py**

```
import re
import string

def remove_white(a): # function to remove possible double or triple or... whitespaces from a string

    a= re.sub(r'(\s{1}\s+)'," ",a) # replace in a all the double or triple or n-ipple :-) whitespaces with one whitespace
    return a

def extract_date(a): # function to match via regular expression a valid datetime format
                # and return a string in this format : 23/12/09 12:23
    date = a
#####  DAY  ##############################################
    valid_day = re.compile(r'(\d+)/') # regular expression, every digit one or more times followed by /
    m = valid_day.match(date) # is it in the date?

    if not m: # if not , report error
            return 'FORMAT ERROR'

    day = string.atoi(m.group().rstrip('/')) # else , make the match an integer after removing the trailing /

    if ((day>31) | (day<1)): # as to apply date logic and check for errors (day is in range 1-31)
            return 'FORMAT ERROR'
    else:
            day = m.group() # so day is the match above , for example 21/
            date = date.replace(m.group(),",1) # spit out the remaining date....
#####   MONTH  ############################################
    valid_month = re.compile(r'(\d+)/') # and the process goes on, on the same spirit
    m = valid_month.match(date)

    if not m:
            return 'FORMAT ERROR'

    month = string.atoi(m.group().rstrip('/'))

    if ((month>12) | (month<1)):
            return 'FORMAT ERROR'
    else:
            month = m.group()
            date = date.replace(m.group(),",1)
#####   YEAR  #############################################
    valid_year = re.compile(r'(\d+)\W')
    m = valid_year.match(date)

    if not m:
            return 'FORMAT ERROR'

    year = string.atoi(m.group().rstrip(' '))

    if year==0:
            return 'FORMAT ERROR'
    else:
```

```
            year = m.group()
            date = date.replace(m.group(),'',1)
#####    HOUR  ##############################################
    valid_hour = re.compile(r'(\d+):')
    m = valid_hour.match(date)

    if not m:
            return 'FORMAT ERROR'

    hour = string.atoi(m.group().rstrip(':'))

    if hour>23:
            return 'FORMAT ERROR'
    else:
            hour = m.group()
            date = date.replace(m.group(),'',1)
#####    MINUTE  ############################################
    valid_min = re.compile(r'(\d+)')
    m = valid_min.match(date)

    if not m:
            return 'FORMAT ERROR'

    min = string.atoi(m.group().rstrip(':'))

    if min>59:
            return 'FORMAT ERROR'
    else:
            min = m.group()
            date = date.replace(m.group(),'',1)

    return day + month + year + hour + min
```

/opt/tinyos-2.0.2/apps/Sensing/python/misc/**scenario.py**

```
import parser

def modem_device(args): # returns the physical modem device

    if "-comm" in args:
            index = args.index("-comm")
            return args[index+1]

def set_phone(args): # returns the physical modem device

    if "-phone" in args:
            index = args.index("-phone")
            return args[index+1]

def scenario_mapping(args):

    scenario = []; # the returned scenario is in the format [scenario,par,tsr,temp,hum]
```

```
if "-rep" in args: # normal scenario : gathers messages and reports at threshold values
                   # e.g ["A",0,1,0,2] means that we report at TSR greater than 1 or at humidity greater than 2
    scenario.append("REPORT_ONLY")


    if "-par" in args: #set par threshold in the scenario
        index = args.index("-par") # find par in arguments
        scenario.append(args[index+1]) # par threshold is the next argument
    else:
        scenario.append(0)

    if "-tsr" in args: #set tsr threshold in the scenario
        index = args.index("-tsr")
        scenario.append(args[index+1])
    else:
        scenario.append(0)

    if "-temp" in args: #set temp threshold in the scenario
        index = args.index("-temp")
        scenario.append(args[index+1])
    else:
        scenario.append(0)

    if "-hum" in args: #set hum threshold in the scenario
        index = args.index("-hum")
        scenario.append(args[index+1])
    else:
        scenario.append(0)



elif "-repat" in args: # report on receipt of sms (actually awaiting instructions from sms)

    scenario.append("REPORT_AT")

    if "-time" in args:
        index = args.index("-time") # find time in arguments
        #HERE GOES THE PARSER
        print " to pass : " ,args[index+1] + " " + args[index+2]
        date = parser.extract_date(args[index+1]+ " " + args[index+2]) # check if the date given is in valid format
        print " date is " , date
        if date=="FORMAT ERROR": # if not, say "BREAK"
            scenario = []
            scenario.append("BREAK")
        else:
            scenario.append("TIME")
            scenario.append(date) # else append the date

          if "-now" in args:
              scenario.append("TIME")
              scenario.append("0") # trigger the report right now option

elif "-reploop" in args: # report at given time and then loop-report at given time intervals

    scenario.append("REPORT_LOOP")
```

```
    if "-time" in args:
        index = args.index("-time") # find time in arguments
        #HERE GOES THE PARSER
        print " to pass : " ,args[index+1] + " " + args[index+2]
        date = parser.extract_date(args[index+1]+ " " + args[index+2]) # check if the date given is in valid format
        print " date is " , date
        if date=="FORMAT ERROR": # if not, say "BREAK"
            scenario = []
            scenario.append("BREAK")
        else:
            scenario.append("TIME")
            scenario.append(date) # else append the date

          if "-now" in args:
            scenario.append("TIME")
            scenario.append("0") # trigger the report right now option

          if "-loop" in args:
            index = args.index("-loop") # find loop in arguments
            loop = args[index+1] # the loop time
            if loop == None: # if no loop there
                scenario = []
                    scenario.append("BREAK") # say "BREAK"
            else:
                scenario.append("LOOP")
                    scenario.append(loop) # append the loop time

          if len(scenario) < 2: # reploop only means nothing
            scenario = []
            scenario.append("BREAK") # say "BREAK"


    else: # wrong arguments

        scenario.append("BREAK")


    return scenario


def dict_scenario(scenario): #returns the scenario values as a dictionary for visionary pleasure and programmatic ease

    dictscenario = dict(nodeid = scenario[0], par=scenario[1], tsr = scenario[2], temp = '%.2f' % float(scenario[3]), hum =
'%.2f' % float(scenario[4]))
    return dictscenario
```

/opt/tinyos-2.0.2/apps/Sensing/python/misc/**scheduler.py**

```
from datetime import datetime

def calcRemainingTime(future_time): # in such a format: "21/11/06 16:30"
```

```
    try:
        now = datetime.now() # take the time now
        future = datetime.strptime(future_time, "%d/%m/%y %H:%M") # bring the future time to  format similar to datetime

        if future < now:
            return 'error' # sorry, past can't be reached

        else:
            diff = future-now # datetime difference
            return diff.days*86400 + diff.seconds # returns the total seconds

    except ValueError: #our error case
        return 'error'
```

/opt/tinyos-2.0.2/apps/Sensing/python/sms/**gsmmodem.py**

```
import serial

class modem(object):

    def __init__(self, dev): #initialize the device
        self.ser = serial.Serial("%s" % dev, 115200, timeout=1)
        self.ser.flushInput()
        self.ser.flushOutput()

    def close(self): #close the serial connection
        self.ser.close()

    def receiveChar(self, chars = 1): # fetch a received character from output
        return self.ser.read(chars)

    def receiveDualResult(self): # fetch two responses from output
        blank = self.receiveLine()
        ir = self.receiveLine()
        blank = self.receiveLine()
        frc = self.receiveLine()

        return ir, frc

    def receiveLine(self):  # fetch a whole line from output
        return self.ser.readline().replace("\r", "").replace("\n", "")

    def receiveSingleResult(self): # fetch a single response from output
        blank = self.receiveLine()
        frc = self.receiveLine()
        return frc

    def sendCommand(self, command, newline = True): # send a command to device
        self.ser.write(command)

        if newline:
            self.ser.write("\r")
```

/opt/tinyos-2.0.2/apps/Sensing/python/sms/**sendsms.py**

```
from gsmmodem import modem

def sendSms(dev,phone_no,message): # a function to send sms

    gsm = modem(dev) # initialize the device

    command = 'ATZ' # first AT command (initialization)
    gsm.sendCommand(command)
    print gsm.receiveSingleResult()

    command = 'AT+CMGF=1' # text mode
    gsm.sendCommand('AT+CMGF=1')
    print gsm.receiveSingleResult()

    command = 'AT+CMGS=' + chr(34) + phone_no + chr(34) # define the phone number to send
    gsm.sendCommand(command)
    print gsm.receiveSingleResult()

    command = message # the message itself
    gsm.sendCommand(command)
    print gsm.receiveSingleResult()

    command = chr(26) # ascii control character, actually the Ctrl + z, needed for termination
    gsm.sendCommand(command,False)
    print gsm.receiveSingleResult()

    gsm.close()
```

/opt/tinyos-2.0.2/apps/Sensing/python/sms/**recvsms.py**

```
from gsmmodem import modem
import string

def recvSms(dev): # returns the last message received and its id

    gsm = modem(dev) # initialize the device

    command = 'AT' # AT initialization
    gsm.sendCommand(command)
    print gsm.receiveSingleResult()

    command = 'AT+CMGF=1' # text mode
    gsm.sendCommand(command)
    print gsm.receiveSingleResult()

    command = 'AT+CMGL=\"ALL\"' # fetch the sms
    gsm.sendCommand(command)
```

```
    gsm.receiveLine()

    messages = []
    last_ids = []
    criterion = []

    while True:
        output = gsm.receiveLine()

            if output == "OK":
                break

            else:
                if output.startswith("+CMGL: "): # calculations and parsing of what we take
                    header = output.replace("\"", "").split(": ")[1].split(",")
                    message = gsm.receiveLine()

                    if header[1].startswith("REC "): # in the main message sector
                        new_message = message
                        this_criterion = string.atoi(header[4].replace('/','')+header[5].replace(':','').replace('+','')) # create integer
from date and time
                        criterion.append(this_criterion)
                        criterion.sort() # sort datetime
                        pos = criterion.index(this_criterion)
                        messages.insert(pos,new_message) # insert message with a sorted datetime way
                        last_ids.insert(pos,header[0]) # insert id of message, sorted

    gsm.close()

    if messages == []: # if no messages return []
        return []

    else:
        return [messages[-1], last_ids[-1]] # return the last message and its id
```

/opt/tinyos-2.0.2/apps/Sensing/python/sms/**delsms.py**

```
from gsmmodem import modem

def delSms(dev,id): # deletes the sms with this id

    gsm = modem(dev)
    command = 'AT' #initialize
    gsm.sendCommand(command)
    print gsm.receiveSingleResult()

    command = 'AT+CMGF=1' # text mode
    gsm.sendCommand(command)
    print gsm.receiveSingleResult()

    command = 'AT+CMGD=%s' % id # delete
    gsm.sendCommand(command)
    print gsm.receiveSingleResult()
```

/opt/tinyos-2.0.2/apps/Sensing/python/sms/**sms_format.py**

```
from datetime import datetime

def sms_report(attributes,dictionary): #actually this is the sms that will be sent

    message = "" # init to nothing
    now = datetime.now() # take this.time :-)
    date = str(now.year) + "-" + str(now.month) + "-" + str(now.day) + " " + str(now.hour) + ":" + str(now.minute) + " " #
express it with this format
    keyz = [k for k, v in dictionary.items()] # fetch keys of dictionary
    values = [v for k, v in dictionary.items()] # fetch values of dictionary

    for i in range(len(values)): # and formulate a nice string of those pairs
        message += str(keyz[i]) + "=" + str(values[i]) + " "

# now bring nodeid=something to the front of the string, so
    nodeid_start_index = message.find("nodeid=") # find index of nodeid in message
    nodeid_stop_index = message.find(" ",nodeid_start_index) # find the index where nodeid=xx stops, eg when you face a "
" after the start index
    nodeid_substring = message[nodeid_start_index:nodeid_stop_index] # extract the substring
    message = message[:nodeid_start_index] + message[nodeid_stop_index+1:] # replace it with "", actually erase it
    message = nodeid_substring + " " + message # and place it in front of the message

    if attributes != 0: # attach a header that reports which attributes exceed the threshold
        anomalous_attributes = "Attributes that exceeded threshold are "
        for i in range(len(attributes)): # attach the attributes that exceeded thresholds
            anomalous_attributes += attributes[i] + ", "
        return date + anomalous_attributes + "Measurements: " + message # a human readable sms

    else: # simply report the measurements
        return date + "Measurements: " + message # a human readable sms

def sms_error(): # when an error occurs
    message = "Received unknown instruction! Please refer to the manual and sent an sms again"
    return message
```

/opt/tinyos-2.0.2/apps/Sensing/python/scripts/**get_phone.sh**

```
# Bash script that prompts for a phone number, then saves them in params.txt

echo "PLEASE DEFINE THE PHONE NUMBER IN WHICH SMS ARE GOING TO BE SENT. Example +30694xxxxx"

# delete contents of params.txt
cat /dev/null > /opt/tinyos-2.0.2/apps/BlinkToRadio/python/scripts/params.txt

while [ "$phone" == "" ]
do
echo -n "Enter phone and press [ENTER]: "
read phone
done
```

```
# write to params.txt
echo "$phone" >> /opt/tinyos-2.0.2/apps/BlinkToRadio/python/scripts/params.txt
```

/opt/tinyos-2.0.2/apps/Sensing/python/scripts/**get_report_time.sh**

```
# Bash script that prompts for a date, then saves them in params.txt

echo "PLEASE DEFINE THE TIME THAT YOU WANT TO BE NOTIFIED BY SMS. TIME MUST FOLLOW THE FORMAT day/month/year hour:minute. Example 12/9/09 16:34"

# delete contents of params.txt
cat /dev/null > /opt/tinyos-2.0.2/apps/BlinkToRadio/python/scripts/params.txt

while [ "$report_time" == "" ]
do
echo -n "Enter time and press [ENTER]: "
read report_time
done

# write to params.txt
echo "$report_time" >> /opt/tinyos-2.0.2/apps/BlinkToRadio/python/scripts/params.txt
```

/opt/tinyos-2.0.2/apps/Sensing/python/scripts/**get_threshold.sh**

```
# Bash script that prompts for some values, then saves them in params.txt

echo "PLEASE SET THE VALUES THAT WHEN EXCEEDED, THE SYSTEM SHALL INFORM VIA SMS. IF YOU SET A VALUE FOR 0, IT WILL NOT BE INSPECTED AND TAKEN UNDER CONSIDERATION FOR THE REST OF THE PROCESS"

# delete contents of params.txt
cat /dev/null > /opt/tinyos-2.0.2/apps/BlinkToRadio/python/scripts/params.txt

while [ "$max_temp" == "" ]
do
echo -n "Enter the temperature threshold value (0 otherwise) and press [ENTER]: "
read max_temp
done

while [ "$max_hum" == "" ]
do
echo -n "Enter the humidity threshold value (0 otherwise) and press [ENTER]: "
read max_hum
done

while [ "$max_psr" == "" ]
do
echo -n "Enter the psr threshold value (0 otherwise) and press [ENTER]: "
read max_psr
done
```

```
while [ "$max_tsr" == "" ]
do
echo -n "Enter the tsr threshold value (0 otherwise) and press [ENTER]: "
read max_tsr
done

# write to params.txt
echo "$max_temp $max_hum $max_psr $max_tsr" >> /opt/tinyos-2.0.2/apps/BlinkToRadio/python/scripts/params.txt
```

opt/tinyos-2.0.2/support/sdk/java/net/tinyos/tools/MyReader.java

```
/*                                                                      tab:4
 * "Copyright (c) 2000-2005 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and
 * its documentation for any purpose, without fee, and without written
 * agreement is hereby granted, provided that the above copyright
 * notice, the following two paragraphs and the author appear in all
 * copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY
 * PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
 * DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS
 * DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN
 * ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 * Copyright (c) 2002-2005 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */
/* Authors:         Phil Levis <pal@cs.berkeley.edu>
 * Date:        December 1 2005
 * Desc:        Generic Message reader
 *
 */

/**
 * @author Phil Levis <pal@cs.berkeley.edu>
 */
```

```
package net.tinyos.tools;

import java.util.*;
import java.text.*;
import net.tinyos.message.*;
import net.tinyos.packet.*;
import net.tinyos.util.*;

import java.sql.*;

public class MyReader implements net.tinyos.message.MessageListener {

  private MoteIF moteIF;

  public MyReader(String source) throws Exception {
    if (source != null) {
      moteIF = new MoteIF(BuildSource.makePhoenix(source, PrintStreamMessenger.err));
    }
    else {
      moteIF = new MoteIF(BuildSource.makePhoenix(PrintStreamMessenger.err));
    }
  }

  public void start() {
  }

  public void messageReceived(int to, Message message) {
    long t = System.currentTimeMillis();
    //    Date d = new Date(t);
    System.out.print("" + t + ": ");
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// mine code (parsing the message received serially....and doing some database insertions)
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    String a = message.toString(); // convert it to string

    int nodeid_start_index = a.indexOf("nodeid"); // index of nodeid
    int par_start_index = a.indexOf("par"); // index of par
    int tsr_start_index = a.indexOf("tsr"); // index of tsr
    int temp_start_index = a.indexOf("temp"); // index of temp
    int hum_start_index = a.indexOf("hum"); // index of hum
    int counter_start_index = a.indexOf("counter"); // index of counter

    int nodeid_stop_index = par_start_index - 5; // empirical rule
    int par_stop_index = tsr_start_index - 5; // empirical rule
    int tsr_stop_index = temp_start_index - 5; // empirical rule
    int temp_stop_index = hum_start_index - 5; // empirical rule
    int hum_stop_index = counter_start_index - 5; //empirical rule
    int counter_stop_index = a.indexOf("]",counter_start_index); //empirical rule

    String hex_string_nodeid = a.substring(nodeid_start_index + 9,nodeid_stop_index); //see below
    String hex_string_par = a.substring(par_start_index + 6,par_stop_index); // the +6 here means that it will remove the
par=0x and shall leave the hexademical string only
    String hex_string_tsr = a.substring(tsr_start_index + 6,tsr_stop_index); // the same for hum
    String hex_string_temp = a.substring(temp_start_index + 7,temp_stop_index); // the same for hum
    String hex_string_hum = a.substring(hum_start_index + 6,hum_stop_index); // the same for hum
```

```
    String hex_string_counter = a.substring(counter_start_index + 10, counter_stop_index); // the same for counter

    int nodeid = Integer.valueOf(hex_string_nodeid, 16).intValue();    //integer value of PAR
    int par = Integer.valueOf(hex_string_par, 16).intValue();    //integer value of PAR
    int tsr = Integer.valueOf(hex_string_tsr, 16).intValue();    //integer value of TSR
    int temp = Integer.valueOf(hex_string_temp, 16).intValue();   //integer value of temperature !!NOT IN CELSIOUS!!
    int hum = Integer.valueOf(hex_string_hum, 16).intValue();    //integer value of humidity
    int cnt = Integer.valueOf(hex_string_counter, 16).intValue();    //integer value of counter

    double celsius_temp = 0.01*temp - 39.60 ; // in CELSIOUS format
    double humidity = -4 + 0.0405*hum + (-2.8*Math.pow(10,-6))*Math.pow(hum,2); // humidity, not temperature
compensated
    double humidity_norm = (celsius_temp - 25) * (0.01 + 0.00008*hum) + humidity; // calculate humidity compensated by
temperature

    DecimalFormat df = new DecimalFormat("#.###"); // more than  two decimal points are rather useless

    float humidity_normalized = Float.parseFloat(df.format(humidity_norm)); // now they are well represented e.g. 30.32
    float temp_normalized = Float.parseFloat(df.format(celsius_temp));

    System.out.println("node id is : " + nodeid);
    System.out.println("PAR is : " + par);
    System.out.println("TSR is : " + tsr);
    System.out.println("temperature is : " + temp_normalized);
    System.out.println("humidity is : " + humidity_normalized);
    System.out.println("counter is : " + cnt);
    System.out.println();

  // DATABASE
    Connection conn = null;

    try{
//some initialization stuff
    String url = "jdbc:mysql://localhost/my_db?user=user&password=12345";
    Class.forName ("com.mysql.jdbc.Driver").newInstance ();
    conn = DriverManager.getConnection (url);
    System.out.println ("Database connection established");

// insert the message parsed values into db

    PreparedStatement s;

    s = conn.prepareStatement("INSERT INTO sensor (nodeid, par, tsr, temp, humidity, counter) VALUES(?,?,?,?,?,?)");

    s.setInt(1, nodeid);
    s.setDouble(2, par);
    s.setDouble(3, tsr);
    s.setFloat(4, temp_normalized);
    s.setFloat(5, humidity_normalized);
    s.setInt(6, cnt);
    s.executeUpdate();

    }
      catch (Exception e)
          {
```

```
            System.err.println ("Cannot connect to database server");
        }

        finally
        {
            if (conn != null)
            {
                try
                {
                    conn.close ();
                    System.out.println ("Database connection terminated");
                }
                catch (Exception e) { /* ignore close errors */ }
            }
        }
}

private static void usage() {
  System.err.println("usage: MsgReader [-comm <source>] message-class [message-class ...]");
}

private void addMsgType(Message msg) {
  moteIF.registerListener(msg, this);
}

public static void main(String[] args) throws Exception {
  String source = null;

  Vector v = new Vector();
  if (args.length > 0) {
    for (int i = 0; i < args.length; i++) {
          if (args[i].equals("-comm")) {
            source = args[++i];
          }
          else {
            String className = args[i];
            try {
              Class c = Class.forName(className);
              Object packet = c.newInstance();
              Message msg = (Message)packet;
              v.addElement(msg);
            }
            catch (Exception e) {
              System.err.println(e);
            }
          }
    }
  }
  else if (args.length != 0) {
    usage();
    System.exit(1);
  }

  MyReader mr = new MyReader(source);
  Enumeration msgs = v.elements();
```

```
    while (msgs.hasMoreElements()) {
      Message m = (Message)msgs.nextElement();
      mr.addMsgType(m);
    }
    mr.start();
  }

}
```

/opt/tinyos-2.0.2/apps/Sensing/**Sensing.h**

```
#ifndef SENSING_H
#define SENSING_H

enum {
  AM_SENSINGMSG = 6, // AM type of AMSenderC
  TIMER_PERIOD_MILLI = 2500 // timeout for sending the message to radio
};

typedef nx_struct SensingMsg { // our message struct which carries the messages
 nx_uint16_t nodeid;
 nx_uint16_t par;
 nx_uint16_t tsr;
 nx_uint16_t temp;
 nx_uint16_t hum;
 nx_uint16_t counter;
} SensingMsg;

#endif
```

/opt/tinyos-2.0.2/apps/Sensing/**SensingAppC.nc**

```
#include <Timer.h>
#include "Sensing.h"

configuration SensingAppC {
}
implementation { // here we define the components of our application
  components MainC;
  components LedsC;
  components SensingC as App;
  components new TimerMilliC() as Timer0;
  components ActiveMessageC;
  components new HamamatsuS1087ParC() as PAR; // for example this component can offer psr measurement
  components new HamamatsuS10871TsrC() as TSR;
  components new SensirionSht11C() as SensorSht;
  components new AMSenderC(AM_SENSINGMSG);
  components new AMReceiverC(AM_SENSINGMSG);

 // here goes the wiring
  App.Boot -> MainC;
```

```
  App.Leds -> LedsC;
  App.Timer0 -> Timer0;
  App.ReadPAR -> PAR;
  App.ReadTSR -> TSR;
  App.ReadTemperature -> SensorSht.Temperature;
  App.ReadHumidity -> SensorSht.Humidity;
  App.Packet -> AMSenderC;
  App.AMPacket -> AMSenderC;
  App.AMSend -> AMSenderC;
  App.AMControl -> ActiveMessageC;
  App.Receive -> AMReceiverC;

}
```

/opt/tinyos-2.0.2/apps/Sensing/**SensingAppC.nc**

```
#include <Timer.h>
#include "Sensing.h"

module SensingC { // the interfaces that we shall use
  uses interface Boot;
  uses interface Leds;
  uses interface Timer<TMilli> as Timer0;
  uses interface Packet;
  uses interface AMPacket;
  uses interface AMSend;
  uses interface Read<uint16_t> as ReadPAR;
  uses interface Read<uint16_t> as ReadTSR;
  uses interface Read<uint16_t> as ReadTemperature;
  uses interface Read<uint16_t> as ReadHumidity;
  uses interface SplitControl as AMControl;
  uses interface Receive;
 }


implementation { // the implementation logic
//some variables declarations
  bool busy = FALSE;
  message_t pkt;

  uint16_t counter = 0;
  uint16_t par_measurement;
  uint16_t tsr_measurement;
  uint16_t temperature;
  uint16_t humidity;

  event void Boot.booted() { // start the radio when the system has booted
    call AMControl.start();
  }

 event void AMControl.startDone(error_t err) { // handler to indicate that we are ready to fire the timer
   if (err == SUCCESS) {
    call Timer0.startPeriodic(TIMER_PERIOD_MILLI); //start the timer if the radio started succesfully
```

```
  }
  else {
    call AMControl.start(); // else start again the radio
  }
 }

 event void AMControl.stopDone(error_t err) { //nothing here
 }


 event void Timer0.fired() { // when the timer fires
   counter++; // increase the message counter
   call ReadPAR.read(); // try to read the things we want to read
   call ReadTSR.read();
   call ReadTemperature.read();
   call ReadHumidity.read();
   call Leds.set(counter);
   if (!busy) { // if a message transmision is not in progress
     SensingMsg* btrpkt = (SensingMsg*)(call Packet.getPayload(&pkt, NULL)); // form the message packet
     btrpkt->nodeid = TOS_NODE_ID;
     btrpkt->par = par_measurement;
     btrpkt->tsr = tsr_measurement;
     btrpkt->temp = temperature;
     btrpkt->hum = humidity;
     btrpkt->counter = counter;
     if (call AMSend.send(AM_BROADCAST_ADDR, &pkt, sizeof(SensingMsg)) == SUCCESS) { // transmission accepted
       busy = TRUE;
     }
    }

 }


event void ReadPAR.readDone(error_t result, uint16_t data) // handler to indicate the par was read successfully
   {
     if (result == SUCCESS){
       par_measurement = data;
     }
   }

event void ReadTSR.readDone(error_t result, uint16_t data) // handler to indicate that tsr was read successfully
   {
     if (result == SUCCESS){
       tsr_measurement = data;
     }
   }
event void ReadTemperature.readDone(error_t result, uint16_t data) // handler to indicate that temperature was read
successfully
   {
     if (result == SUCCESS){
       temperature = data;
     }
   }

event void ReadHumidity.readDone(error_t result, uint16_t data) // handler to indicate that humidity was read successfully
```

```
  {
    if (result == SUCCESS){
      humidity = data;
    }
  }


 event void AMSend.sendDone(message_t* msg, error_t error) { // handler to indicate that the message buffer can be reused after we sent a message
   if (&pkt == msg) {
     busy = FALSE;
   }
 }

 event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len) {  // handler to indicate a receipt of a message
   if (len == sizeof(SensingMsg)) {
     SensingMsg* btrpkt = (SensingMsg*)payload;
     call Leds.set(btrpkt->counter);
    }
   return msg;
 }

 }
```

/opt/tinyos-2.0.2/apps/Sensing/**Makefile**

```
COMPONENT=SensingAppC
include $(MAKERULES)
```