Μελέτη και εφαρμογή των προτύπων ZigBee και DigiMesh για την υλοποίηση ασύρματου δικτύου αισθητήρων

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κολοτούρος Δημήτριος-Μάριος
Τεχνολόγος Πληροφορικής και Τηλεπικοινωνιών

Επίβλεψη: Επίκουρος Καθηγητής Ιωαννίδης Κωνσταντίνος
Εργαστήριο Πυρηνικής Φυσικής

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΙΣ ΣΥΓΧΡΟΝΕΣ ΗΛΕΚΤΡΟΝΙΚΕΣ ΤΕΧΝΟΛΟΓΙΕΣ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

Ιωάννινα, Δεκέμβριος 2011

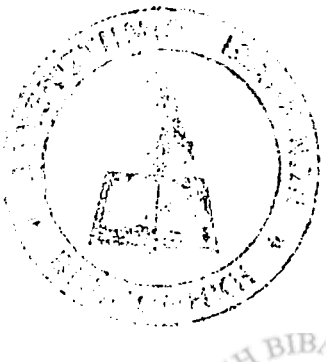
Η έγκριση της παρούσας μεταπτυχιακής διπλωματικής εργασίας από το τμήμα φυσικής του πανεπιστημίου Ιωαννίνων δεν υποδηλώνει αποδοχή του συγγραφέα.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα Επίκουρο Καθηγητή κ. Ιωαννίδη Κωνσταντίνο, μέλος του Εργαστηρίου Πυρηνικής Φυσικής, για την υποστήριξη, την καθοδήγηση και την πολύτιμη βοήθειά του στο διάστημα της εκπόνησης της εργασίας μου, καθώς και τα υπόλοιπα μέλη της τριμελούς εξεταστικής επιτροπής κ. Μάνθο Νικόλαο, Αναπληρωτή Καθηγητή και κ. Φουντά Κωνσταντίνο, Καθηγητή μέλη του Εργαστηρίου Φυσικής Υψηλών Ενεργειών για τις καίριες παρατηρήσεις τους.

Επίσης θα ήθελα να ευχαριστήσω τα μέλη του Εργαστηρίου Φυσικής Υψηλών Ενεργειών για τις συμβουλές, τη βοήθεια και την εμπιστοσύνη που έδειξαν στο πρόσωπό μου.

Τέλος θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου για την υπομονή και την πολύπλευρη συμπαράστασή της



Σύνοψη

Ο ραγδαία ανερχόμενος κλάδος των Ασύρματων Δικτύων Αισθητήρων (Wireless Sensor Networks ή αλλιώς WSNs) είναι ιδιαίτερα διαδεδομένος τα τελευταία χρόνια τόσο στον ακαδημαϊκό όσο και στο βιομηχανικό τομέα. Ένα WSN αποτελείται από ένα σύνολο περιφερειακών κόμβων πυκνά τοποθετημένων μεταξύ τους και ένα κύριο κόμβο για τη συλλογή των πληροφοριών και το συντονισμό του δικτύου. Κάθε ένας από τους περιφερειακούς κόμβους αποτελείται από μια διάταξη αισθητήρων, έναν μικροελεγκτή και έναν πομποδέκτη. Έτσι κάθε κόμβος έχει την δυνατότητα να ανιχνεύει ή και να μετρά ένα φυσικό μέγεθος, να κάνει υπολογισμούς και να μεταφέρει την πληροφορία στο δίκτυο. Οι εφαρμογές που μπορούν να προκύψουν είναι αναρίθμητες και περιορίζονται μόνο από την φαντασία του μηχανικού. Ενδεικτικοί τομείς εφαρμογής είναι η υγεία, η βιομηχανία, το περιβάλλον και άλλοι.

Η συγκεκριμένη διπλωματική εργασία επικεντρώθηκε στη μελέτη των κυριάρχων προτύπων επικοινωνίας για χρήση στα WSN καθώς και στην υλοποίηση ενός WSN, τόσο σε επίπεδο υλικού (hardware), όσο και λογισμικού (software), το οποίο είναι ικανό να καλύψει ένα ευρύ φάσμα εφαρμογών, δίνοντας τη δυνατότητα στο χρήστη να συνδέσει οποιονδήποτε αισθητήρα (συμβατό με κάποιον από τους διαύλους επικοινωνίας SPI, I^2C, 1-Wire ή απλά με μία αναλογική έξοδο) σε κάθε κόμβο. Η υλοποίηση του WSN από πλευράς υλικού περιλαμβάνει τον Κύριο Κόμβο με ενσωματωμένο Web Server ώστε ο χρήστης να μπορεί να διαχειρίζεται το WSN μέσω LAN ή Internet και ενεργειακά αυτόνομους περιφερειακούς κόμβους που κάνουν χρήση της ηλιακής ενέργειας για τη φόρτιση των μπαταριών τους. Από πλευράς λογισμικού, ο κώδικας αναπτύχθηκε στο προγραμματιστικό περιβάλλον Arduino. Ο συνδυασμός του λογισμικού και του υλικού συνθέτουν το σύστημα που μπορεί να χαρακτηριστεί ως ενσωματωμένο (Embedded System). Το προτεινόμενο σύστημα, μετά το πέρας του σχεδιασμού και της υλοποίησής του, δοκιμάστηκε εκτενώς σε πραγματικές συνθήκες. Από τις δοκιμές αυτές, που κράτησαν αρκετές εβδομάδες, παρατηρήθηκε ότι το σύστημα δουλεύει αξιόπιστα. Το σύστημα έχει υλοποιηθεί με την κατασκευή κατάλληλων ηλεκτρονικών κυκλωμάτων.

Abstract

The rapidly ascending branch of Wireless Sensor Networks (or WSNs) is particularly popular the last few years both in academia and in industry. A WSN consists of a set of densely placed nodes, and one master node that gathers the data and coordinates the network. Each node consists of a sensor arrangement, a microcontroller and a radio transceiver. Thus, each node has the ability to sense, process and send data through the network. The applications that can arise are endless and can only be limited by the imagination of the engineer. Indicative areas of application are health, industry, environment etc.

This thesis is focused on the study of the dominant communication standards for use in WSNs as long as the implementation of a WSN, in hardware and also in software, which is capable to cover a wide range of applications, providing the user the ability to connect any sensor (compatible with one of the following buses: SPI, I2C, 1-Wire or with an analog output) on every node. Regarding the hardware, the WSN was designed as a main node attached to a web server offering the user the ease of managing the WSN through a LAN or over the Internet and also energy sufficient peripheral nodes that use solar power for charging their battery. Concerning software, the code was developed on Arduino programming environment. The combination of hardware and software lead to the realization of a system be best described by the term "Embedded System". Following design and implementation, the system was extensively tested in real-life conditions. From these tests, which lasted several weeks, the reliability of the system was assesed.

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε κατά το διάστημα 2010-2011 στο εργαστήριο πυρηνικής φυσικής του τμήματος φυσικής της σχολής θετικών επιστημών, υπό την επίβλεψη του επίκουρου καθηγητή κ. Ιωαννίδη Κωνσταντίνου. Σκοπός αυτής της εργασίας είναι η μελέτη των κυριάρχων προτύπων επικοινωνίας για χρήση στα δίκτυα WSN καθώς και η υλοποίηση ενός δικτύου WSN, τόσο σε επίπεδο υλικού (hardware), όσο και λογισμικού (software) το οποίο είναι ικανό να καλύψει ένα ευρύ φάσμα εφαρμογών. Συγκεκριμένα, στις παραγράφους που ακολουθούν παρουσιάζεται επιγραμματικά η δομή της εργασίας:

Στο πρώτο κεφάλαιο γίνεται εισαγωγή στην έννοια του ασύρματου δικτύου αισθητήρων. Παράλληλα παρουσιάζονται και αναλύονται οι αρχές λειτουργίας του, τα χαρακτηριστικά του, η διαφοροποίησή του σε σχέση με τα παραδοσιακά δίκτυα δεδομένων καθώς και οι πιθανές εφαρμογές του.

Στο δεύτερο κεφάλαιο γίνεται συνοπτική περιγραφή των διαθέσιμων προτύπων επικοινωνίας για χρήση στα ασύρματα δίκτυα αισθητήρων ενώ δίνεται ιδιαίτερη έμφαση στα πρότυπα IEEE 802.15.4, ZigBee και DigiMesh.

Στο τρίτο κεφάλαιο παρουσιάζεται η πλατφόρμα ελεύθερου υλικού και λογισμικού Arduino. Αφού αναλυθούν τα χαρακτηριστικά του υλικού και του λογισμικού, γίνεται αναφορά στις διαφορές και τα πλεονεκτήματα χρήσης της πλατφόρμας έναντι άλλων.

Το τέταρτο κεφάλαιο ασχολείται με το υπο κατασκευή δίκτυο. Αρχικά αναφέρονται οι προδιαγραφές και τα χαρακτηριστικά του και εν συνεχεία η λειτουργία του. Τέλος παρουσιάζεται ο ασύρματος πομποδέκτης που επιλέχθηκε για να χρησιμοποιηθεί τόσο στον κύριο όσο και στους περιφερειακούς κόμβους.

Το πέμπτο κεφάλαιο ασχολείται με τον κύριο κόμβο του υπο κατασκευή δικτύου. Αναλύονται τα δευτερεύοντα και το κύριο κύκλωμα που τον αποτελούν. Παράλληλα γίνεται εκτενής αναφορά στη συνδεσμολογία των ακροδεκτών του μικροελεγκτή με τα περιφερειακά εξαρτήματα.

Το έκτο κεφάλαιο ασχολείται με τους περιφερειακούς κόμβους του υπο κατασκευή δικτύου. Αναλύονται τα επιμέρους κυκλώματα που τον αποτελούν και παρέχονται οδηγίες για τη συναρμολόγησή του.

Στο έβδομο κεφάλαιο παρουσιάζεται η διεπαφή χρήστη (Graphical User Interface ή GUI).

Τέλος, το όγδοο κεφάλαιο αποτελείται από τα συμπεράσματα της παρούσας εργασίας.

Περιεχόμενα

Ακρωνύμια

ADC: Analog to Digital Converter
AODV: Ad-hoc On Demand Distance Vector
API: Application Programming Interface
APS: Application Support sub-layer
ASCII: American Standard Code for Information Interchange
C3I: military Command, Control, Communication, and Intelligence systems
CMOS: Complementary metal–oxide–semiconductor
CS: Chip Select
CSMA-CA: Carrier sense multiple access with collision avoidance
CSV: Comma Separated Value
DAC: Digital to Analog Converter
DDNS: Dynamic Domain Name System
DIN: Digital Input
DOUT: Digital Output
DSSS: Direct-Sequence Spread Spectrum
EEPROM: Electrically Erasable Programmable Read-Only Memory
FET: Field-Effect Transistor
FFD: Full Function Device
FHSS: Frequency-Hopping Spread Spectrum
GND: Ground
GPS: Global Positioning System
GTS: Guaranteed Time Slots
I^2C: Inter-Integrated Circuit
IDE: Integrated Development Environment
I/O: Input/output
IPv6: Internet Protocol (version 6)
ISM band: industrial, scientific and medical band
LR-WPAN: Low Rate Wireless Personal Area Network
MAC: Media Access Control
MANET: Mobile Ad-Hoc Network
MCU: Micro-Controller Unit
MISO: Master In Slave Out
OSI: Open Systems Interconnection
PPTC: Polymeric Positive Temperature Coefficient (device)
QoS: Quality of Service

RF: Radio Frequency

RFD: Reduced Function Device

RSSI: Received Signal Strength Indication

RTC: Real Time Clock

RX: Receive

SCL: Serial Clock

SD: Secure Data

SDA: Serial Data

SMBUS: System Management Bus

SPI: Serial Peripheral Interface

SS: Slave Select

TX: Transmit

UART: Universal Asynchronous Receiver/Transmitter

WBAN: Wireless Body Area Network

WSN: Wireless Sensor Network

ZDO: ZigBee Device Object

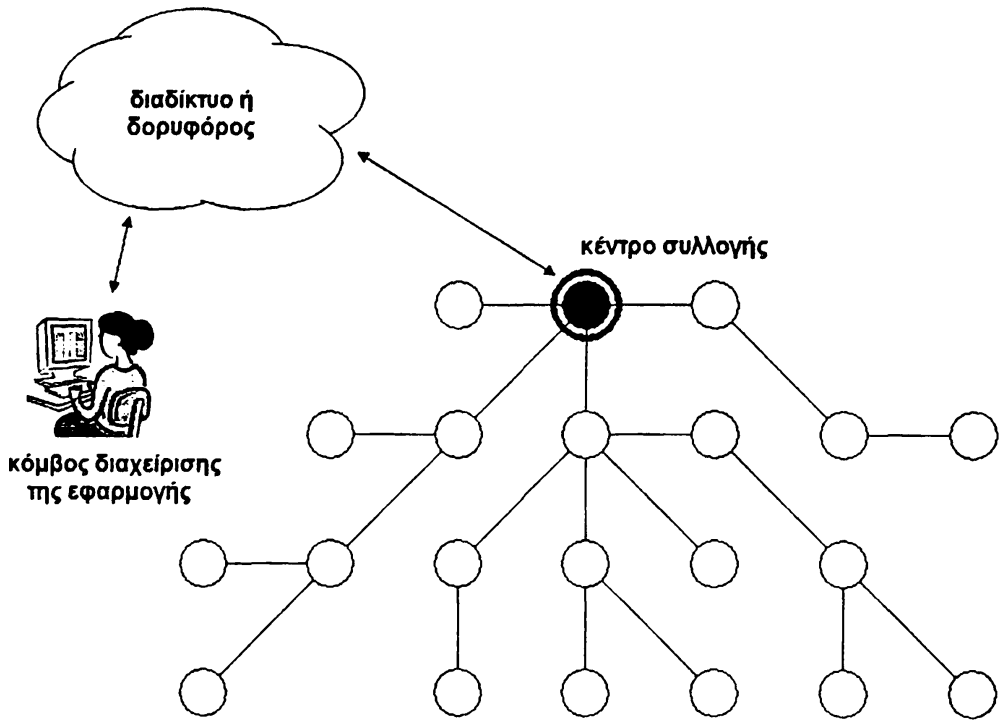
1. Ασύρματα Δίκτυα Αισθητήρων

1.1. Γενικά

Τα ασύρματα δίκτυα αισθητήρων είναι μία αναδυόμενη τεχνολογία η οποία υπόσχεται ένα ευρύ φάσμα δυνητικών εφαρμογών τόσο στον πολιτικό όσο και στο στρατιωτικό τομέα. Ένα ασύρματο δίκτυο αισθητήρων (Wireless Sensor Network=WSN) αποτελείται συνήθως από ένα μεγάλο αριθμό κόμβων, χαμηλού κόστους μεν αλλά και περιορισμένων δυνατοτήτων, ανεπτυγμένων σε μια περιοχή ενδιαφέροντος.

Οι κόμβοι αυτοί, αν και μικροί σε μέγεθος, είναι εξοπλισμένοι με αισθητήρες, μικροελεγκτές ή μικροεπεξεργαστές, ασύρματους πομποδέκτες καθώς και κάποιου είδους συσκευή αποθήκευσης ενέργειας (μπαταρία). Γίνεται αντιληπτό λοιπόν ότι δεν έχουν μόνο τη δυνατότητα «αίσθησης», αλλά έχουν και δυνατότητες επεξεργασίας των δεδομένων, ασύρματης επικοινωνίας καθώς και ενεργειακής αυτονομίας (Zheng, και συν., 2009).

Συνεργαζόμενοι, καταγράφουν φυσικά και κλιματολογικά μεγέθη όπως θερμοκρασία, ήχο, δονήσεις, πίεση, κίνηση, ρυπογόνες ουσίες κλπ σε διάφορες τοποθεσίες. Η μη προκαθορισμένη θέση των κόμβων και η απαίτηση για συλλογική λειτουργία προς εκπλήρωση του σκοπού τους απαιτούν πρωτόκολλα και αλγορίθμους τα οποία παρέχουν στο δίκτυο τη δυνατότητα της αυτό-οργάνωσης (self-organization). Ο τελικός προορισμός των μετρήσεων στα δίκτυα αισθητήρων είναι συνήθως ένα κέντρο συλλογής με μεγάλα ενεργειακά αποθέματα και με δυνατότητα περαιτέρω επεξεργασίας των δεδομένων με απώτερο σκοπό τη λήψη αποφάσεων (Λιακάκος, 2009)



Εικόνα 1: Συνήθης αρχιτεκτονική Δικτύου Αισθητήρων

Τα τελευταία δέκα χρόνια τόσο η ακαδημαϊκή κοινότητα όσο και η βιομηχανία σε όλο τον κόσμο έχουν στρέψει τα βλέμματά τους στα WSN εξαιτίας των μεγάλων δυνατοτήτων τους, προβλέποντας ότι θα φέρουν επανάσταση στον τρόπο με τον οποίο ζούμε, εργαζόμαστε και αλληλεπιδρούμε με το φυσικό κόσμο.

1.2. Χαρακτηριστικά των WSN

Συγκριτικά με τα παραδοσιακά ασύρματα δίκτυα επικοινωνιών όπως τα κυψελωτά ή τα ad-hoc, τα ασύρματα δίκτυα αισθητήρων έχουν μοναδικά χαρακτηριστικά. Τα πιο σημαντικά είναι τα παρακάτω (Zheng, et al., 2009):

- **Υψηλή χωρική πυκνότητα:** Οι κόμβοι είναι πολύ πυκνά διασκορπισμένοι στο χώρο. Ο αριθμός των κόμβων σε ένα WSN μπορεί να ξεπεράσει τάξεις μεγέθους τον αριθμό των κόμβων ενός δικτύου MANET (Mobile Ad-Hoc Network).
- **Μεγάλος περιορισμός ενεργειακής αυτονομίας, υπολογιστικής δύναμης και χώρου αποθήκευσης δεδομένων.**
- **Δυνατότητα αυτό-οργάνωσης:** Οι κόμβοι συνήθως τοποθετούνται τυχαία στο χώρο. Γι' αυτό το λόγο οι κόμβοι πρέπει αυτόνομα να δημιουργήσουν ένα λειτουργικό δίκτυο.
- **Αυτονομία λειτουργίας και προγραμματισμός:** Κάθε κόμβος πρέπει να μπορεί να πάρει αποφάσεις για την εξέλιξη της εφαρμογής, χωρίς την παρέμβαση χειριστή, σε περίπτωση σημαντικών μεταβολών των μετρούμενων μεγεθών (π.χ. αύξηση συχνότητας δειγματοληψίας). Σε περίπτωση αλλαγής στρατηγικής δε, είναι απαραίτητη η δυνατότητα επαναπρογραμματισμού των κόμβων.
- **Προορισμένα για συγκεκριμένη εφαρμογή (application Specific):** Ένα δίκτυο σχεδιάζεται με βάση τις απαιτήσεις μιας συγκεκριμένης εφαρμογής και είναι πολύ δύσκολο να χρησιμοποιηθεί για κάποια άλλη, αν δεν υποστεί σημαντικές αλλαγές τόσο στον αλγόριθμο λειτουργίας, όσο και στη δομή και το χρησιμοποιούμενο υλικό. Για παράδειγμα, για ένα ασύρματο δίκτυο από αισθητήρες που είναι τοποθετημένοι σε κινούμενα ρομπότ που αλληλεπιδρούν μεταξύ τους, ακολουθείται τελείως διαφορετική στρατηγική σε σχέση με ένα στατικό WSN λήψης μετρήσεων από μια τοποθεσία.
- **Χαμηλή αξιοπιστία κόμβων:** Οι κόμβοι συχνά τοποθετούνται σε αφιλόξενα και εχθρικά περιβάλλοντα χωρίς τη δυνατότητα επιτήρησης και συντήρησης. Αυτός ο παράγοντας καθιστά επιρρεπείς του κόμβους σε φθορές και καταστροφές.
- **Χρόνος ζωής:** Στις περισσότερες περιπτώσεις WSN δικτύων η λειτουργία των κόμβων εξαρτάται από μια περιορισμένη πηγή ενέργειας (μπαταρίες). Η προσεκτική διαχείριση ενέργειας κάθε κόμβου παίζει πολύ σημαντικό ρόλο στη διατήρηση του δικτύου για μεγάλο χρονικό διάστημα. Ο χρόνος ζωής είναι μέγεθος αντιστρόφως ανάλογο της ποιότητας λειτουργίας. Ξοδεύοντας περισσότερη ενέργεια έχουμε καλύτερη απόδοση αλλά μικρότερο χρόνο ζωής. Η εξισορρόπηση αυτών των δυο εξαρτάται από το είδος της εφαρμογής.
- **Συχνή αλλαγή τοπολογίας δικτύου:** Η τοπολογία του δικτύου αλλάζει συχνά εξαιτίας απώλειας κόμβων (από εξάντληση ενέργειας ή καταστροφή), ενσωμάτωση νέων, συμφόρησης μέρος δικτύου κλπ.
- **Απλότητα:** Από τη στιγμή που οι κόμβοι διαθέτουν περιορισμένους πόρους, το λειτουργικό σύστημα και ο αλγόριθμος λειτουργίας δεν έχουν ιδιαίτερα υψηλό επίπεδο πολυπλοκότητας.
- **Ποιότητα λειτουργίας (Quality of Service ή QoS):** Η έννοια της ποιότητας επικοινωνίας μπορεί να διαφέρει πολύ απ' ότι σε συμβατικά δίκτυα, καθώς σε ένα WSN δίκτυο πιθανόν να μην παίζει πρωτεύοντα ρόλο η ταχύτητα και ο ρυθμός μετάδοσης, αλλά η αξιόπιστη μετάδοση όλων των μηνυμάτων χωρίς πολλές επανεκπομπές.

1.3. Διαφοροποίηση των WSN σε σχέση με τα υπόλοιπα δίκτυα

Παρόλο που τα WSN αποτελούν δίκτυα υπολογιστικών συσκευών, διαφέρουν από τα παραδοσιακό δίκτυα δεδομένων για τους εξής λόγους (Παπαγεωργακοπούλου, 2009):

- Παρουσιάζουν σε σχέση με τα κλασικά δίκτυα σημαντικούς περιορισμούς στην υπολογιστική ισχύ, στην ενέργεια, την αποθήκευση και το εύρος ζώνης. Στα παραδοσιακά ασύρματα δίκτυα οι λειτουργίες της δρομολόγησης και της διαχείρισης κινητικότητας εκτελούνται με σκοπό τη βελτιστοποίηση της ποιότητας υπηρεσιών Quality of Service-QoS) και της αποτελεσματικότητας του εύρους ζώνης, ενώ η κατανάλωση ενέργειας συνιστά δευτερεύουσα απαίτηση, καθώς η πηγή ενέργειας μπορεί να αντικατασταθεί ή να επαναφορτιστεί οποιαδήποτε στιγμή. Ωστόσο, τα WSN αποτελούνται από κόμβους που έχουν σχεδιαστεί για εφαρμογή σε περιβάλλον λειτουργίας χωρίς ανθρώπινη παρέμβαση. Συνεπώς, μια παρεχόμενη υπηρεσία της δρομολόγησης αποτελεί η βελτιστοποίηση της χρήσης της ενέργειας, με σκοπό τη μεγιστοποίηση της διάρκειας ζωής του δικτύου.
- Στα παραδοσιακά δίκτυα οι χρήστες συνδέονται με ένα κόμβο (ή ένα σύνολο κόμβων) και απαιτούν κάποια υπηρεσία από κάποιο άλλο κόμβο. Αυτό το επικοινωνιακό μοντέλο μεταξύ δύο επί μέρους τμημάτων του δικτύου αποτελεί την πλειοψηφία της συνήθους διακίνησης πληροφοριών και το δίκτυο συνιστά το μέσο διασύνδεσής τους. Επίσης το μοντέλο αλληλεπίδρασης είναι άμεσο, καθόσον ο χρήστης αλληλεπιδρά άμεσα με άλλο χρήστη ή υπηρεσία στο άλλο τερματικό της επικοινωνίας. Εξάλλου, τα WSN προσεγγίζουν περισσότερο τα κατανεμημένα συστήματα (πολλαπλοί Η/Υ συνδεδεμένοι μεταξύ τους με σκοπό την επίτευξη ενός κοινού στόχου), παρά τα τυπικά δίκτυα. Οι κόμβοι συνεργάζονται για την παραγωγή των αποτελεσμάτων, ενώ ο χρήστης σπάνια ενδιαφέρεται για τα αποτελέσματα μεμονωμένων κόμβων. Αντιθέτως, αναζητά παραμέτρους μιας συνολικής, δυναμικής φυσικής διαδικασίας. Συνεπώς, δεν υπάρχουν σαφείς κόμβοι-πηγές ή προορισμοί βασιζόμενοι στις επιθυμίες του χρήστη, μόνο οι χρήστες και το συνολικό δίκτυο. Κατά συνέπεια, το δίκτυο δεν παρέχει σύνδεση μεταξύ διαφορετικών τμημάτων, αλλά υπηρεσίες πληροφοριών στους χρήστες.
- Συνήθως οι κόμβοι ενός WSN παραμένουν στην ίδια θέση μετά την τοποθέτησή τους, με εξαίρεση μικρό αριθμό μετακινούμενων κόμβων.
- Οι κόμβοι ενός WSN δικτύου αποστέλλουν δεδομένα χαμηλού ρυθμού μετάδοσης, με εμφανές το φαινόμενο του πλεονασμού (Στην περίπτωση που μεταδίδονται μεμονωμένα δεδομένα από κάθε κόμβο, αυξάνεται η πιθανότητα οι ενδιάμεσοι κόμβοι να λάβουν το ίδιο μήνυμα από γειτονικούς κόμβους και ο σταθμός βάσης να λάβει τελικά πολλαπλές αντιγραφές του ίδιου μηνύματος και να χρειάζεται να διαγράψει την περίσσεια πληροφορία).

1.4. Εφαρμογές των WSN δικτύων

Οι αισθητήρες μπορούν να χρησιμοποιηθούν για να ανιχνεύουν ή να παρακολουθούν ευρύ φάσμα φυσικών παραμέτρων όπως για παράδειγμα:

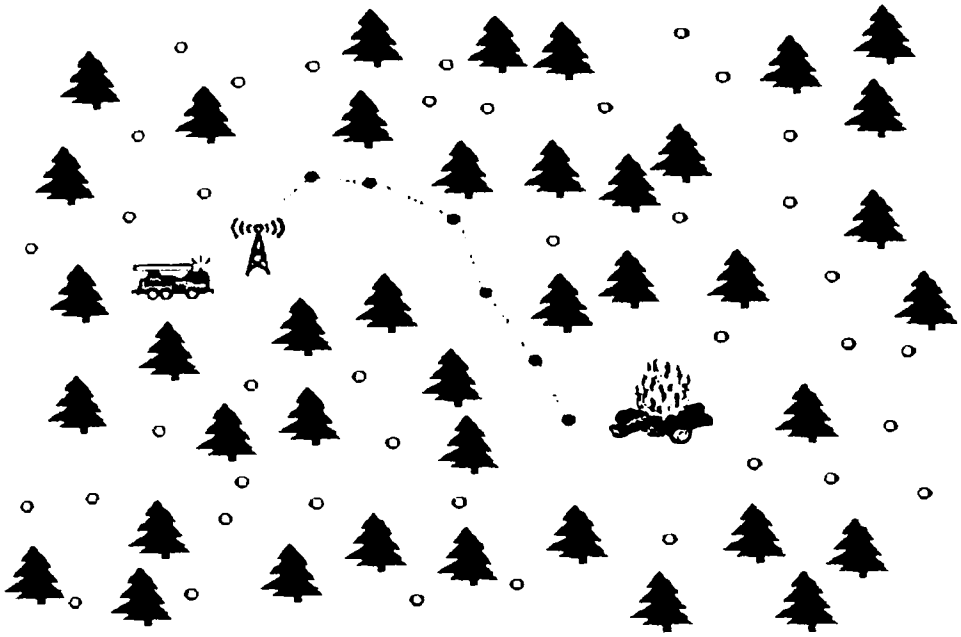
- Φως
- Ήχο
- Υγρασία
- Θερμοκρασία
- Πίεση
- Σύσταση εδάφους
- Ποιότητα αέρα ή νερού
- Χαρακτηριστικά μεγέθη κάποιου αντικειμένου όπως διαστάσεις, βάρος, θέση, ταχύτητα, κατεύθυνση κλπ.

Οι ασύρματοι αισθητήρες έχουν σημαντικά πλεονεκτήματα έναντι αισθητήρων που κάνουν χρήση καλωδίων. Όχι μόνο μπορούν να μειώσουν το κόστος και το χρόνο εγκατάστασής τους, αλλά και να εγκατασταθούν σε σχεδόν οποιοδήποτε εχθρικό και αφιλόξενο περιβάλλον όπως σε πεδία μαχών, στο διάστημα, σε βαθείς ωκεανούς κλπ. Τα ασύρματα δίκτυα αισθητήρων αναπτύχθηκαν αρχικά για στρατιωτικές εφαρμογές και περιελάμβαναν από μεγάλης κλίμακας δίκτυα για την ακουστική παρακολούθηση των ωκεανών, έως πολύ μικρής κλίμακας δίκτυα εδάφους για την ανίχνευση στόχων. Ωστόσο, η διαθεσιμότητα αισθητήρων χαμηλού κόστους και η ανάπτυξη των ασύρματων επικοινωνιών επέτρεψε τη ραγδαία εξέλιξη αυτής της τεχνολογίας τόσο σε πολιτικές όσο και σε στρατιωτικές εφαρμογές. Παρακάτω παρουσιάζονται ορισμένα παραδείγματα εφαρμογών των WSN δικτύων (Zheng, et al., 2009).

1.4.1. Περιβαλλοντική παρακολούθηση

Η περιβαλλοντική παρακολούθηση είναι μία από τις πρώτες εφαρμογές των WSN. Χρησιμοποιούνται αισθητήρες για την παρακολούθηση μιας ποικιλίας περιβαλλοντικών παραμέτρων.

- Παρακολούθηση παραμέτρων περιβάλλοντος διαβίωσης: Χρησιμοποιούνται αισθητήρες για την παρακολούθηση των συνθηκών διαβίωσης της άγριας χλωρίδας και πανίδας (Mainwaring, et al., 2002).
- Παρακολούθηση ποιότητας νερού και αέρα: Αναπτύσσονται αισθητήρες επίγειοι ή υποβρύχιοι ώστε να παρακολουθούν την ποιότητα του νερού και του αέρα. Για παράδειγμα η παρακολούθηση της ποιότητας του νερού μπορεί να χρησιμοποιηθεί στον τομέα της υδροχημείας, ενώ η παρακολούθηση της ποιότητας του αέρα μπορεί να χρησιμοποιηθεί στον έλεγχο της ατμοσφαιρικής ρύπανσης (Ilyas, et al., 2005).
- Ανίχνευση κινδύνου: Οι αισθητήρες χρησιμοποιούνται για την ανίχνευση βιολογικών ή χημικών κινδύνων, για παράδειγμα σε ένα χημικό εργοστάσιο ή στο πεδίο της μάχης (Zheng, et al., 2009).
- Παρακολούθηση καταστροφών: Οι αισθητήρες αναπτύσσονται πυκνά σε μία περιοχή ενδιαφέροντος για την ανίχνευση καταστροφών φυσικών ή μη. Για παράδειγμα μπορούν να διασκορπιστούν αισθητήρες σε ένα δάσος για την ανίχνευση πιθανής φωτιάς ή πλημμύρας. Σεισμικοί αισθητήρες μπορούν να ενσωματωθούν σε κτήρια ώστε να μπορεί να καθοριστεί η κατεύθυνση και το μέγεθος ενός σεισμού και να παρέχουν στοιχεία για την ασφάλεια του κτηρίου (Zheng, et al., 2009).

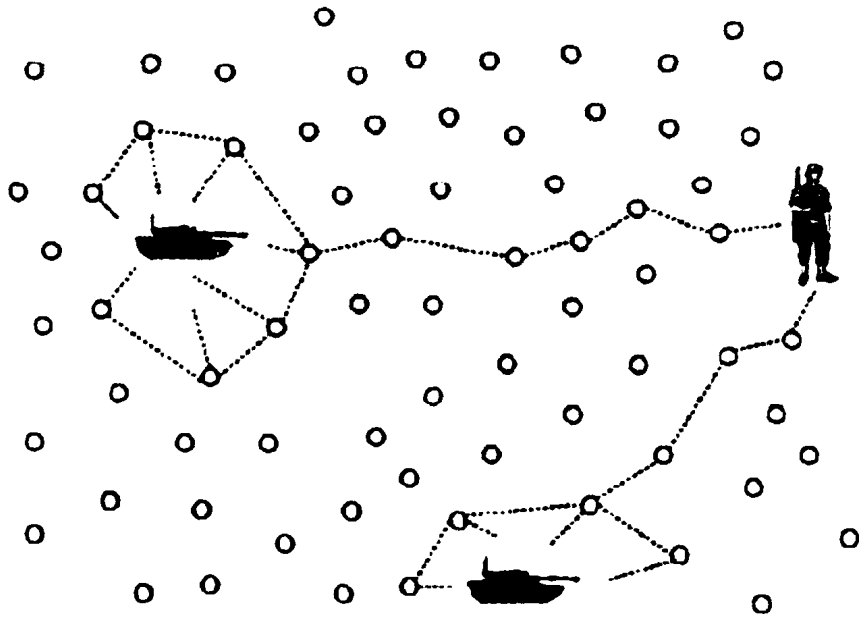


Εικόνα 2: Πυρανίχνευση με ασύρματο δίκτυο αισθητήρων

1.4.2. Στρατιωτικές εφαρμογές

Τα WSN δίκτυα έχουν γίνει αναπόσπαστο κομμάτι των στρατιωτικών συστημάτων C3I. Οι ασύρματοι αισθητήρες μπορούν να αναπτυχθούν γρήγορα στο πεδίο της μάχης ή σε μία αφιλόξενη περιοχή, χωρίς κάποια προϋπάρχουσα υποδομή. Λόγω της ευκολίας εγκατάστασης, της αντοχής, και της ανοχής σε σφάλματα, τα δίκτυα αισθητήρων θα αποτελέσουν σημαντικό κομμάτι των μελλοντικών στρατιωτικών συστημάτων C3I και θα κάνει τους μελλοντικούς πολέμους πιο «έξυπνους» (Ammari, 2009).

- Παρακολούθηση πεδίου μάχης: Αισθητήρες μπορούν να τοποθετηθούν στο πεδίο της μάχης για να παρακολουθούν την παρουσία και τις κινήσεις χερσαίων δυνάμεων και οχημάτων (Εικόνα 3).
- Προστασία χώρων/αντικειμένων: Κόμβοι αισθητήρων μπορούν να αναπτυχθούν γύρω από αντικείμενα ή χώρους μεγάλου ενδιαφέροντος όπως πυρηνικά εργοστάσια, γέφυρες στρατηγικής σημασίας, αγωγούς πετρελαίου και φυσικού αερίου, κέντρα τηλεπικοινωνιών, στρατιωτικά αρχηγεία κλπ.
- Πλοήγηση: Αισθητήρες μπορούν να τοποθετηθούν σε μη-επανδρωμένα ρομποτικά οχήματα, τανκς, μαχητικά αεροσκάφη, υποβρύχια, πυραύλους και τορπίλες ώστε να τα καθοδηγήσουν στους στόχους τους αποφεύγοντας εμπόδια και συντονισμένα μεταξύ τους να καταφέρουν μια πιο αποτελεσματική επίθεση ή άμυνα.
- Τηλεανίχνευση: Αναπτύσσονται αισθητήρες για να ανιχνεύσουν πιθανή χρήση πυρηνικών, βιολογικών ή χημικών όπλων καθώς και πιθανές τρομοκρατικές επιθέσεις.



Εικόνα 3. Αναγνώριση εχθρικών δυνάμεων.

1.4.3. Εφαρμογές στον τομέα της Υγείας

Τα WSN μπορούν επίσης να χρησιμοποιηθούν για να παρακολουθούνται ηλικιωμένοι και ασθενείς για λόγους υγείας, ανακουφίζοντας σημαντικά τη σοβαρή έλλειψη προσωπικού υγειονομικής περίθαλψης και μειώνοντας τις δαπάνες στων συστημάτων υγειονομικής περίθαλψης (Gavrilovska, et al., 2011).

- Παρακολούθηση συμπεριφοράς: Τοποθετούνται αισθητήρες στο σπίτι ενός ασθενή ώστε να παρακολουθείται κατάσταση της υγείας του. Για παράδειγμα μπορεί να ειδοποιηθεί το ΕΚΑΒ, ο γιατρός ή οι οικείοι ενός ασθενούς σε περίπτωση που πέσει ή χρίζει άμεσης ιατροφαρμακευτικής περίθαλψης. Μπορεί επίσης να παρακολουθηθούν οι κινήσεις του ασθενούς και να παρέχονται υπενθυμίσεις και οδηγίες για τη λήψη κάποιου φαρμάκου από την τηλεόραση, το ραδιόφωνο ή τον Η/Υ.
- Ιατρική παρακολούθηση: Αισθητήρες που μπορούν να φορεθούν ενσωματώνονται σε ένα WBAN (Wireless Body Area Network) για την παρακολούθηση ζωτικών σημάτων, περιβαλλοντικών παραμέτρων και γεωγραφικής θέσης. Έτσι παρέχεται η δυνατότητα μακροπρόθεσμης, μη-παρεμβατικής παρακολούθησης ενός ασθενούς ή ηλικιωμένου ατόμου με άμεση ενημέρωση του ιατρικού προσωπικού σε περίπτωση ανάγκης. Ο χρήστης μπορεί να ενημερώνεται οποιαδήποτε στιγμή για την κατάσταση της υγείας του καθώς και για το ιστορικό του.

1.4.4. Έλεγχος βιομηχανικών διεργασιών

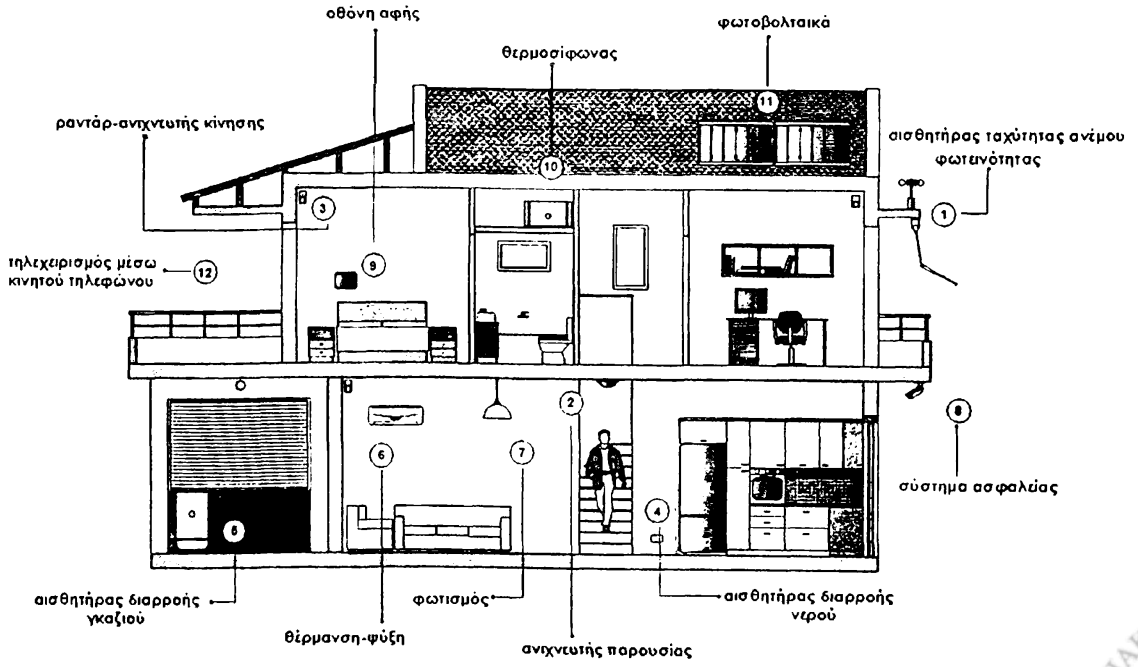
Στη βιομηχανία τα WSN μπορούν να χρησιμοποιηθούν για να παρακολουθούν τη διαδικασία παραγωγής ή την κατάσταση του εξοπλισμού της γραμμής παραγωγής. Για παράδειγμα ασύρματοι αισθητήρες μπορούν να χρησιμοποιηθούν σε χημικά εργοστάσια ή διυλιστήρια για να παρακολουθούν την κατάσταση των αγωγών τους των οποίων το μήκος μπορεί να υπερβαίνει τις μερικές δεκάδες χιλιόμετρα. Μικροσκοπικοί αισθητήρες μπορούν να ενσωματωθούν σε μέρη μίας μηχανής απρόσιτα από τους τεχνικούς ενημερώνοντας για την κατάστασή της ή για τυχόν αποτυχία. Παραδοσιακά, ο εξοπλισμός συντηρείται τακτικά βάσει προγράμματος, για παράδειγμα κάθε δύο μήνες. Η κατάσταση αυτή είναι δαπανηρή, με τη χρήση όμως των WSN, η συντήρηση μπορεί να διεξαχθεί βάσει της κατάστασης του εξοπλισμού, εξοικονομώντας δαπάνες άσκοπης συντήρησης και αυξάνοντας τη διάρκεια ζωής του (Zheng, et al., 2009).

1.4.5. Ασφάλεια και επίβλεψη

Τα ασύρματα δίκτυα αισθητήρων μπορούν να έχουν μία ευρεία χρήση στον τομέα της ασφάλειας και της επίβλεψης. Αισθητήρες ήχου, εικόνας, κίνησης, θραύσης κρυστάλλων κλπ μπορούν να τοποθετηθούν σε κτήρια, αεροδρόμια, σιδηροδρομικούς σταθμούς κλπ, για να εντοπίζουν τυχόν παραβιάσεις (Zheng, et al., 2009).

1.4.6. Το «έξυπνο» σπίτι

Τα WSN δίκτυα μπορούν να συνεισφέρουν σε έξυπνους τρόπους διαβίωσης. Έτσι, ασύρματοι αισθητήρες μπορούν να ενσωματωθούν σε ένα σπίτι και να συνδεθούν μεταξύ τους για να σχηματίσουν ένα αυτόνομο οικιακό δίκτυο. Για παράδειγμα ένα «έξυπνο» ψυγείο θα μπορεί να ετοιμάσει ένα μενού βάσει των τροφίμων που περιέχει (Zheng, et al., 2009). Ακόμα με τη χρήση μετεωρολογικών αισθητήρων θα μπορεί το σπίτι να ελέγχει το άνοιγμα ή το κλείσιμο των παραθύρων, τον κλιματισμό και την θέρμανση διατηρώντας έτσι την επιθυμητή από το χρήστη θερμοκρασία και εξοικονομώντας ενέργεια (Kuorilehto et al., 2007) Ένα παράδειγμα «έξυπνου» σπιτιού φαίνεται στην εικόνα 4.



Εικόνα 4

1.5. Στόχοι σχεδιασμού δικτύων αισθητήρων

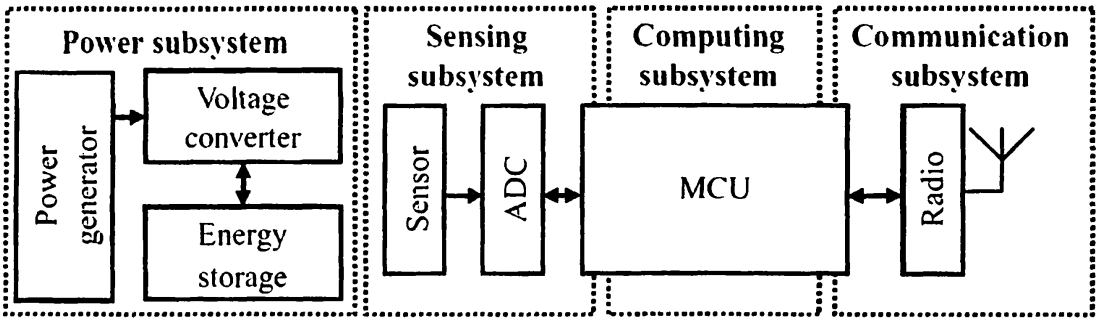
Τα χαρακτηριστικά των ασύρματων δικτύων αισθητήρων και των απαιτήσεων των διάφορων εφαρμογών, έχουν αποφασιστικό αντίκτυπο στους στόχους για το σχεδιασμό του δικτύου. Οι κύριοι στόχοι σχεδιασμού βασίζονται στις ακόλουθες προϋποθέσεις (Zheng, et al., 2009):

- Μικρό μέγεθος κόμβων. Η μείωση του μεγέθους των κόμβων είναι από τους πρωταρχικούς στόχους σχεδίασης. Οι κόμβοι συχνά αναπτύσσονται σε εχθρικά και αφιλόξενα περιβάλλοντα σε μεγάλους αριθμούς. Έτσι, η μείωση του μεγέθους των κόμβων μπορεί να διευκολύνει την ανάπτυξή τους καθώς και να μειώσει το κόστος παραγωγής τους και την κατανάλωσή τους σε ενέργεια.
- Χαμηλό κόστος κόμβων. Η μείωση του κόστους παραγωγής των κόμβων είναι ακόμα ένας βασικός στόχος σχεδίασης. Δεδομένου ότι οι κόμβοι αναπτύσσονται συνήθως σε μεγάλους αριθμούς και δεν μπορούν να επαναχρησιμοποιηθούν, είναι σημαντικό να μειωθεί το κόστος των κόμβων ώστε να μειωθεί κατά συνέπεια το συνολικό κόστος του δικτύου.
- Χαμηλή κατανάλωση ενέργειας. Η μείωση στην κατανάλωση ενέργειας αποτελεί ίσως τον πιο σημαντικό σχεδιαστικό στόχο. Καθόσον οι κόμβοι τροφοδοτούνται από μπαταρίες, και επειδή αρκετά συχνά είναι πολύ δύσκολο έως ακατόρθωτο να επαναφορτίσουν τις μπαταρίες τους, είναι ζωτικής σημασίας να ελαχιστοποιηθεί η κατανάλωση ενέργειας ώστε να μεγιστοποιηθεί η διάρκεια ζωής των κόμβων και κατ' επέκταση του δικτύου.
- Αυτό-οργάνωση. Στα ασύρματα δίκτυα αισθητήρων, οι κόμβοι συνήθως αναπτύσσονται άναρχα, χωρίς κάποιο προσεκτικό σχεδιασμό. Από τη στιγμή που θα αναπτυχθούν θα πρέπει να μπορούν να οργανωθούν αυτόνομα σχηματίζοντας ένα πλήρως λειτουργικό δίκτυο. Το δίκτυο αυτό θα πρέπει να ανταπεξέρχεται σε τυχόν αλλαγές τοπολογίας ή αποτυχίες κόμβων.
- Δυνατότητα επέκτασης. Σε ένα ασύρματο δίκτυο αισθητήρων ο αριθμός των κόμβων μπορεί να ανέρχεται σε δεκάδες, εκατοντάδες ή ακόμα και χιλιάδες. Έτσι, τα δικτυακά πρωτόκολλα που προορίζονται για χρήση σε ασύρματα δίκτυα αισθητήρων θα πρέπει να είναι έτσι σχεδιασμένα ώστε να είναι επεκτάσιμα.
- Προσαρμοστικότητα. Σε ένα ασύρματο δίκτυο αισθητήρων σε λειτουργία, ένας κόμβος μπορεί να αποτύχει, να ενσωματωθεί ή να μετακινηθεί. Οποιαδήποτε από τις παραπάνω ενέργειες θα προκαλέσει αλλαγή στην πυκνότητα των κόμβων και την τοπολογία του δικτύου. Έτσι τα δικτυακά πρωτόκολλα που προορίζονται για χρήση σε ασύρματα δίκτυα αισθητήρων θα πρέπει να είναι προσαρμόσιμα στις όποιες αλλαγές τοπολογίας και πυκνότητας κόμβων.
- Αξιοπιστία. Σε πολλές εφαρμογές των ασύρματων δικτύων αισθητήρων, απαιτείται τα δεδομένα να παραδίδονται αξιόπιστα, ανταγωνιζόμενα θορυβώδη και επιρρεπή σε λάθη ασύρματα κανάλια. Για να ανταπεξέλθουν σε αυτές τις απαιτήσεις, τα δικτυακά πρωτόκολλα που προορίζονται για χρήση σε ασύρματα δίκτυα αισθητήρων θα πρέπει να παρέχουν έλεγχο σφαλμάτων και μηχανισμούς διόρθωσης.
- Ανοχή σε σφάλματα. Οι κόμβοι είναι επιρρεπείς σε αποτυχίες καθώς όπως προαναφέρθηκε συχνά τοποθετούνται σε αφιλόξενα και εχθρικά περιβάλλοντα χωρίς τη δυνατότητα επιτήρησης και συντήρησης. Έτσι, οι κόμβοι θα πρέπει να έχουν ανοχή σε σφάλματα, και επίσης τη δυνατότητα να πραγματοποιούν δοκιμές, να βαθμονομούνται και να επισκευάζονται σε κάποιο βαθμό μόνοι τους.

- Ασφάλεια. Σε πολλές στρατιωτικές εφαρμογές απαιτείται από ένα ασύρματο δίκτυο αισθητήρων ένα επίπεδο ασφάλειας, ώστε να μην είναι ευάλωτο στους αντιπάλους. Έτσι, πρέπει να θεσπιστούν αποτελεσματικοί μηχανισμοί ασφαλείας για τη διαφύλαξη των δεδομένων από κακόβουλες επιθέσεις.
- Αξιοποίηση εύρους ζώνης. Τα ασύρματα δίκτυα αισθητήρων έχουν περιορισμένο εύρος ζώνης. Συνεπώς τα πρωτόκολλα επικοινωνίας θα πρέπει να το αξιοποιούν με το μέγιστο δυνατό τρόπο.
- Ποιότητα Υπηρεσιών (QoS). Στα ασύρματα δίκτυα αισθητήρων, διαφορετικές εφαρμογές μπορεί να έχουν διαφορετικές απαιτήσεις όσον αφορά την καθυστέρηση παράδοσης και απώλεια πακέτων δεδομένων. Για παράδειγμα, σε ένα δίκτυο ανίχνευσης πυρκαγιών, η έγκαιρη παράδοση των πακέτων είναι ζωτικής σημασίας. Αντιθέτως, σε ένα επιστημονικό πείραμα, μια χρονική καθυστέρηση στα πακέτα είναι ανεκτή, όχι όμως και η απώλειά τους. Έτσι, τα δικτυακά πρωτόκολλα θα πρέπει να σχεδιαστούν κατά τέτοιο τρόπο που να λαμβάνεται υπ' όψη η ποιότητα υπηρεσιών της κάθε εφαρμογής.

1.6. Αρχιτεκτονική Κόμβων

Στην αρχιτεκτονική ενός κόμβου, όπως φαίνεται και από την Εικόνα 5 περιλαμβάνονται τα παρακάτω υποσυστήματα: το υποσύστημα αισθητήρων, το υποσύστημα επεξεργασίας, το υποσύστημα επικοινωνιών και το υποσύστημα τροφοδοσίας (Kuorilehto et al., 2007).



Εικόνα 5

1.6.1. Το υποσύστημα αισθητήρων

Το υποσύστημα αισθητήρων αποτελείται από έναν ή και περισσότερους αισθητήρες οι οποίοι διαθέτουν μία αναλογική ή ψηφιακή έξοδο για την ανάγνωση των τιμών τους. Οι αισθητήρες με αναλογική έξοδο απαιτούν τη χρήση ενός ADC (Analog-to-Digital Converter-Μετατροπέας Αναλογικού σε Ψηφιακό) ο οποίος περιλαμβάνεται στο υποσύστημα (Kuorilehto et al., 2007).

Η λειτουργία της αίσθησης εξαρτάται από το είδος της εφαρμογής, υπάρχουν ωστόσο κάποιες γενικές λειτουργίες που εκτελεί ο κόμβος (Παπαγεωργακοπούλου, 2009):

- Καθορισμός σε μια δεδομένη τοποθεσία της τιμής μιας φυσικής παραμέτρου (π.χ. θερμοκρασία, ατμοσφαιρική πίεση, ποσότητα φωτός και σχετική υγρασία σε ένα σύνολο θέσεων μέσω διαφορετικών αισθητήρων, καθένας με διαφορετικό ρυθμό δειγματοληψίας και δυναμική περιοχή τιμών)

- Αντίληψη γεγονότων και εκτίμηση παραμέτρων τους (π.χ. ανίχνευση διέλευσης οχήματος από διασταύρωση και εκτίμηση της ταχύτητας και κατεύθυνσής του)
- Ανίχνευση αντικειμένου (π.χ. ανίχνευση εισβολής στην παρατηρούμενη από το δίκτυο περιοχή) και πιθανότατα κατηγοριοποίηση του αντικειμένου (π.χ. αναγνώριση του είδους ενός οχήματος)

Επιπλέον, τα WSN δίκτυα μπορούν να χαρακτηριστούν ανάλογα με τη συγκέντρωση των απαιτούμενων από την εφαρμογή δεδομένων ως α) συνεχή (όταν οι κόμβοι συλλέγουν αδιαλείπτως δεδομένα), β) ανταποκρινόμενα, (reactive-όταν αποκρίνονται στην απαίτηση ενός παρατηρητή ή συλλέγουν δεδομένα που αφορούν συγκεκριμένα γεγονότα) και γ) περιοδικά (όταν συλλέγουν δεδομένα σύμφωνα με συνθήκες καθοριζόμενες από την εφαρμογή).

1.6.2. Το υποσύστημα επεξεργασίας

Το υποσύστημα επεξεργασίας αποτελείται συνήθως από ένα μικροελεγκτή (MCU) ο οποίος ενσωματώνει έναν επεξεργαστή, μνήμες, χρονοδιακόπτες (timers), ρολόι πραγματικού χρόνου (RTC), ακροδέκτες εισόδου/εξόδου (I/O), ADC και άλλα περιφερειακά υποσυστήματα.

Το υποσύστημα εκτελεί βασικούς υπολογισμούς επεξεργασίας σήματος και πιθανότατα διεργασίες συσχέτισης δεδομένων, όπως είναι η διεργασία Data fusion (συνδυασμός ενός ή περισσότερων πακέτων δεδομένων που έχουν ληφθεί από διαφορετικούς αισθητήρες για τη δημιουργία ενός μοναδικού πακέτου), η οποία οδηγεί στη μείωση της μεταδιδόμενης ποσότητας δεδομένων και συνεπώς της καταναλισκόμενης ενέργειας. Άλλες πιθανές διεργασίες είναι η συμπίεση δεδομένων και η επεξεργασία ασφάλειας (Παπαγεωργακοπούλου, 2009).

1.6.3. Το υποσύστημα επικοινωνιών

Το υποσύστημα επικοινωνιών αποτελείται από ένα πομποδέκτη ραδιοσυχνοτήτων και μία κεραία και είναι υπεύθυνο για όλες τις εισερχόμενες και εξερχόμενες επικοινωνίες του κόμβου.

Σε ένα WSN δίκτυο η επικοινωνία μπορεί να διακριθεί ως α) επικοινωνία υποδομής και β) επικοινωνία εφαρμογών. Η πρώτη κατηγορία αναφέρεται στην επικοινωνία που απαιτείται για τον καθορισμό (configuration), τη διατήρηση και τη βελτιστοποίηση της λειτουργίας του δικτύου, του οποίου η τοπολογία και ο προσδιορισμός ενδέχεται να μεταβάλλονται συχνά, δεδομένου ενός εχθρικού περιβάλλοντος, μεγάλου φόρτου εργασιών και ροπή των κόμβων σε αποτυχία. Σε ένα στατικό WSN απαιτείται μια αρχική προεργασία για τη δημιουργία του δικτύου και η επιπρόσθετη επικοινωνία κρίνεται αναγκαία για την υλοποίηση του επαναπροσδιορισμού του. (Παπαγεωργακοπούλου, 2009).

1.6.4. Το υποσύστημα τροφοδοσίας

Το υποσύστημα τροφοδοσίας παρέχει και ρυθμίζει την τάση τροφοδοσίας του εκάστοτε κόμβου. Τα περιορισμένα ενεργειακά αποθέματα του κάθε κόμβου ορίζουν αυστηρές απαιτήσεις ενεργειακής απόδοσης για το συγκεκριμένο υποσύστημα (Kuorilehto et al., 2007). Παράλληλα μπορεί να διαχειρίζεται μία εξωτερική πηγή ενέργειας (π.χ. φωτοβολταϊκή κυψέλη) εφόσον υπάρχει για την επαναφόρτιση της μπαταρίας του κόμβου.

2. Πρότυπα Ασύρματων Δικτύων Αισθητήρων

2.1. Γενικά

Στο κεφάλαιο αυτό θα γίνει συνοπτική περιγραφή των διαθέσιμων προτύπων για χρήση στα ασύρματα δίκτυα αισθητήρων. Ιδιαίτερη έμφαση θα δοθεί στα πρότυπα IEEE 802.15.4 καθώς αποτελεί τη «βάση» για όλα τα υπόλοιπα πρότυπα, το ZigBee το οποίο είναι ιδιαίτερα διαδεδομένο πρότυπο και το DigiMesh μέσω του οποίου διεξήχθη η επικοινωνία των κόμβων του δικτύου που υλοποιήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας.

2.2. Το πρότυπο WirelessHart

Το πρότυπο WirelessHart παρέχει ένα πρωτόκολλο ασύρματης επικοινωνίας για μετρήσεις διεργασιών και εφαρμογές ελέγχου. Στηρίζεται στο πρότυπο IEEE 802.15.4, λειτουργεί στα 2.4GHz σε χαμηλή ισχύ ενώ είναι συμβατό με όλες τις υπάρχουσες συσκευές, μηχανήματα και συστήματα. Το WirelessHart είναι αξιόπιστο, ασφαλές και ενεργειακά αποδοτικό. Υποστηρίζει τοπολογία δικτύου βρόχου (Mesh), channel hopping (άλματα καναλιού) και μηνύματα συγχρονισμού (Yick, et al., 2008)

2.3. Το πρότυπο ISA100.11a

Το πρότυπο ISA100.11a σχεδιάστηκε για παρακολούθηση διεργασιών σε χαμηλό ρυθμό μετάδοσης και εφαρμογές αυτοματισμού. Ορίζει τις προδιαγραφές για το μοντέλο OSI, την ασφάλεια και τη διαχείριση συστήματος. Παράλληλα εστιάζει στη χαμηλή κατανάλωση ενέργειας, στην επεκτασιμότητα και στη συμβατότητα με άλλες ασύρματες συσκευές. Λειτουργεί στη συχνότητα των 2.4GHz και υποστηρίζει τοπολογίες βρόχου (Mesh) και αστέρα (Star) (Yick, et al., 2008).

2.4. Το πρότυπο 6LoWPAN

Το πρότυπο 6LoWPAN επιτρέπει την ανταλλαγή πακέτων IPv6 σε ένα δίκτυο IEEE 802.15.4, ενώ παρέχει όλα τα πλεονεκτήματα της επικοινωνίας IP (Internet Protocol) και διαχείρισης. Οι συσκευές χαμηλής κατανάλωσης μπορούν να επικοινωνούν απευθείας με συσκευές που διαθέτουν διεύθυνση IP (Yick, et al., 2008). Δεδομένου ότι το μέγεθος των πακέτων IP είναι μεγαλύτερο από αυτό ενός πακέτου IEEE 802.15.4, χρησιμοποιείται ένα επίπεδο (layer) προσαρμογής (Παπαγεωργακοπούλου, 2009).

2.5. Το πρότυπο IEEE 802.15.3

Το πρότυπο IEEE 802.15.3 ορίζει το φυσικό και το επίπεδο MAC (Media Access Control) για υψηλό ρυθμό μετάδοσης δεδομένων. Σχεδιάστηκε για να υποστηρίξει πολυμεσικές εφαρμογές και μετάδοση εικόνας και ήχου συνεχούς ροής σε πραγματικό χρόνο. Λειτουργεί στη συχνότητα των 2.4GHz και ο ρυθμός μετάδοσής του κυμαίνεται από τα 11 Mbps έως τα 55 Mbps (Yick, et al., 2008).

2.6. Το πρότυπο Wibree

Το πρότυπο Wibree σχεδιάστηκε για χαμηλή κατανάλωση ενέργειας, μικρή εμβέλεια και χαμηλό κόστος συσκευών. Το Wibree επιτρέπει την επικοινωνία μεταξύ μικρών συσκευών που τροφοδοτούνται με μπαταρία, όπως ρολόγια, ασύρματα πληκτρολόγια, αισθητήρες για αθλητική χρήση κλπ με συσκευές Bluetooth. Λειτουργεί στη συχνότητα των 2.4GHz και ο ρυθμός μετάδοσής του είναι 1 Mbps. Η εμβέλειά του κυμαίνεται από 5 m έως και 10 m (Yick, et al., 2008).

2.7. Το πρότυπο IEEE 802.15.4

2.7.1. Γενικά

Το πρότυπο 802.15.4 ορίζει τις προδιαγραφές του φυσικού επιπέδου και του επιπέδου MAC, για χαμηλού ρυθμού μετάδοσης ασύρματα δίκτυα (LR-WPAN) που σχηματίζονται από σταθερές ή κινούμενες συσκευές, τροφοδοτούμενες από μπαταρίες ή κάποια άλλη περιορισμένη πηγή ενέργειας (Τζόκας, και συν., 2008).

Ο σχεδιασμός του προτύπου εστιάζει κυρίως στην ελαχιστοποίηση της κατανάλωσης των συσκευών, στην αξιόπιστη μετάδοση των δεδομένων και στην εύκολη εγκατάσταση του δικτύου με βάση απλά και ευέλικτα πρωτόκολλα. Μερικά από τα χαρακτηριστικά του προτύπου είναι τα ακόλουθα (IEEE Standards, 2003):

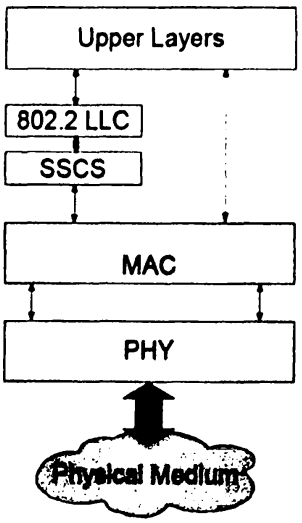
- Ταχύτητες μετάδοσης δεδομένων της τάξης των 250kb/s, 40kb/s και 20kb/s.
- Τοπολογία αστέρα ή peer-to-peer.
- Διευθυνσιοδότηση μεγέθους 16 bit ή 64 bit
- Πρόσβαση καναλιών με τη χρήση CSMA-CA (Carrier sense Multiple Access with Collision Avoidance)
- Λήψη αναφορών παράδοσης για πιο αξιόπιστη μεταφορά δεδομένων.
- Χαμηλή κατανάλωση ενέργειας
- Ανίχνευση ενεργειακών αποθεμάτων
- Ανίχνευση ποιότητας ασύρματης ζεύξης
- 16 κανάλια στη ζώνη των 2.4GHz, 10 κανάλια στη ζώνη των 915MHz και 1 κανάλι στη ζώνη των 868MHz.

2.7.2. Είδη κόμβων

Ένα σύστημα σύμφωνο με το πρότυπο IEEE 802.15.4 μπορεί να αποτελείται από αρκετές συσκευές. Κάθε συσκευή μπορεί να είναι ή του τύπου RFD (Reduced Function Device-συσκευή περιορισμένου αριθμού λειτουργιών) ή του τύπου FFD (Full Function Device-συσκευή πλήρους συνόλου λειτουργιών). Δύο ή περισσότερες συσκευές που επικοινωνούν στο ίδιο φυσικό κανάλι συνιστούν ένα δίκτυο WPAN. Ωστόσο, ένα δίκτυο θα πρέπει να περιλαμβάνει τουλάχιστον μία συσκευή FFD που θα λειτουργεί ως συντονιστής του δικτύου (Buratti, et al., 2011).

2.7.3. Αρχιτεκτονική του Προτύπου IEEE 802.15.4

Η αρχιτεκτονική του προτύπου IEEE 802.15.4 ορίζεται μέσα από ένα σύνολο ξεχωριστών επιπέδων, όπως και στο μοντέλο OSI. Κάθε επίπεδο είναι υπεύθυνο για ένα τμήμα των λειτουργιών του προτύπου και παρέχει υπηρεσίες στα υψηλότερα επίπεδα. Έτσι η αρχιτεκτονική μιας συσκευής αποτελείται από το φυσικό επίπεδο (PHY), που περιλαμβάνει τον πομποδέκτη ραδιοσυχνοτήτων (RF) μαζί με κάποιους χαμηλού επιπέδου μηχανισμούς ελέγχου και το επίπεδο MAC, που παρέχει πρόσβαση στο φυσικό κανάλι για όλους τους τύπους μετάδοσης. Στην εικόνα 6 φαίνεται σχηματικά η αρχιτεκτονική του δικτύου (Τζόκας, et al., 2008).



Εικόνα 6

2.7.3.1. Το επίπεδο PHY

Το επίπεδο PHY (Physical) παρέχει υπηρεσίες όπως ενεργοποίηση και απενεργοποίηση του πομποδέκτη, την αποστολή και λήψη πακέτων δεδομένων στο φυσικό μέσο, την επιλογή καναλιού, την ανίχνευση κίνησης στο κανάλι, την ανίχνευση ποιότητας της ασύρματης ζεύξης καθώς και την ανίχνευση της ισχύος του λαμβανόμενου σήματος (RSSI) (IEEE Standards, 2003).

Ο πομποδέκτης λειτουργεί σε μία από τις ακόλουθες ελεύθερες συχνότητες:

- 868-868.6MHz (Ευρώπη)
- 902-928MHz (Βόρεια Αμερική)
- 2400-2483.5MHz (Παγκοσμίως)

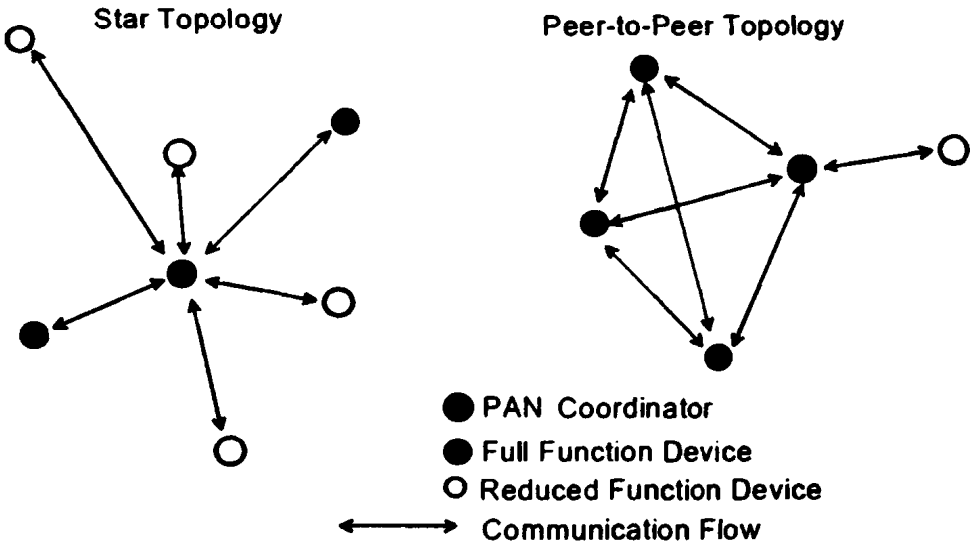
2.7.3.2. Το επίπεδο MAC

Το επίπεδο MAC διαχειρίζεται την πρόσβαση στο προς και από το φυσικό επίπεδο ενώ παράλληλα είναι υπεύθυνο για τις παρακάτω εργασίες (IEEE Standards, 2003):

- Χρήση του μηχανισμού CSMA-CA για την πρόσβαση στο κανάλι
- Ασφάλεια συσκευής
- Υποστήριξη του μηχανισμού GTS (Guaranteed Time Slots)
- Παροχή αξιόπιστης σύνδεσης μεταξύ δύο οντοτήτων MAC
- Σύνδεση και αποσύνδεση στο δίκτυο WPAN
- Εκπομπή μηνυμάτων συγχρονισμού και αναγνώρισης του δικτύου σε περίπτωση που η συσκευή είναι ο συντονιστής.

2.7.4. Τοπολογίες Δικτύου

Ανάλογα με τις απαιτήσεις της εφαρμογής, ένα δίκτυο LR-WPAN μπορεί να λειτουργήσει ακολουθώντας μία από τις ακόλουθες τοπολογίες: Αστέρα και peer-to-peer. Μία σχηματική αναπαράσταση των τοπολογιών φαίνεται στην εικόνα 7:



Εικόνα 7

Στην τοπολογία αστέρα, η επικοινωνία γίνεται μεταξύ των συσκευών και ενός συντονιστή του δικτύου. Η κάθε συσκευή συνήθως «τρέχει» κάποια σχετική εφαρμογή ενώ είναι είτε το σημείο έναρξης είτε το σημείο λήξης της επικοινωνίας. Ο συντονιστής του δικτύου (PAN coordinator) «τρέχει», όπως και οι άλλες συσκευές, κάποια σχετική εφαρμογή, αλλά παράλληλα μπορεί να χρησιμοποιηθεί για την έναρξη, λήξη και δρομολόγηση της επικοινωνίας σε όλο το δίκτυο. Συνήθως σε μία τέτοια τοπολογία, ο συντονιστής τροφοδοτείται από το ηλεκτρικό δίκτυο, ενώ οι υπόλοιπες συσκευές από κάποια μπαταρία. Η τοπολογία αυτή συναντάται κυρίως σε αυτοματισμούς σπιτιού, περιφερειακά Η/Υ, παιχνίδια και συσκευές που έχουν σχέση με τον κλάδο της υγείας (IEEE Standards, 2003).

Η τοπολογία peer-to-peer χρησιμοποιεί και αυτή μία συσκευή ως συντονιστή του δικτύου. Όμως διαφέρει από την τοπολογία αστέρα καθώς κάθε συσκευή μπορεί να επικοινωνήσει με οποιαδήποτε άλλη στο δίκτυο, αρκεί να είναι εντός εμβέλειας. Η τοπολογία αυτή επιτρέπει περίπλοκους σχηματισμούς δικτύου, ακόμα και σχηματισμό νέων τοπολογιών με βάση την τοπολογία peer-to-peer όπως η «mesh». Εφαρμογές όπως ο βιομηχανικός έλεγχος, τα ασύρματα δίκτυα αισθητήρων, η παρακολούθηση και ιχνηλάτηση αντικειμένων, η ευφυής γεωργία και η ασφάλεια μπορούν να επωφεληθούν από τη νέα

τοπολογία. Ένα δίκτυο peer-to-peer μπορεί να είναι ad-hoc και να έχει τη δυνατότητα αυτό-οργάνωσης και αυτό-επισκευής. Μπορεί επίσης να επιτρέψει πολλαπλά άλματα (multi-hop) για τη δρομολόγηση μηνυμάτων από μία συσκευή σε οποιαδήποτε άλλη εντός του δικτύου. Τέτοιες λειτουργίες δεν είναι μέρος του προτύπου, μπορούν όμως να προστεθούν στο επίπεδο του δικτύου όπως θα περιγραφεί στη συνέχεια.

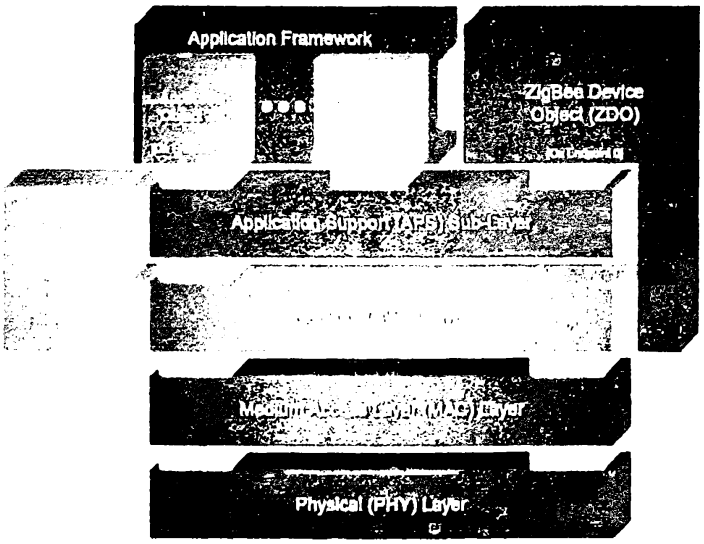
2.8. Το πρότυπο ZigBee

2.8.1. Γενικά

Το πρότυπο ZigBee αποτελεί μία από τις νεότερες τεχνολογίες στο χώρο των ασύρματων δικτύων προσωπικού χώρου (WPANs). Προέκυψε από τη συνεργασία πολλών εταιριών, οι οποίες σχημάτισαν την κοινοπραξία «ZigBee Alliance». Παρέχει τη δυνατότητα διασύνδεσης συσκευών με χαμηλό ρυθμό μετάδοσης, χαμηλό κόστος και χαμηλή κατανάλωση ενέργειας (Κριτσωτάκη, 2010).

2.8.2. Αρχιτεκτονική Προτύπου ZigBee

Ουσιαστικά το πρότυπο ZigBee αποτελεί επέκταση της στοίβας πρωτοκόλλων του IEEE 802.15.4, καθώς υλοποιεί τα επίπεδα δικτύου και εφαρμογών, βασιζόμενο στις υπηρεσίες που παρέχουν το φυσικό (PHY) και το επίπεδο MAC του προτύπου IEEE 802.15.4. Η στοίβα πρωτοκόλλων του προτύπου ZigBee βασίζεται και αυτή στο μοντέλο OSI. Μια λεπτομερής απεικόνισή παρουσιάζεται στην εικόνα 8. Τα πρώτα δυο επίπεδα ορίζονται από το 802.15.4 ενώ τα υπόλοιπα από το ZigBee (Τζόκας, et al., 2008).



Εικόνα 8

2.8.2.1. Το επίπεδο NWK (Network)

Το επίπεδο NWK είναι υπεύθυνο για τη δικτύωση τοπολογίας «mesh». Αυτό περιλαμβάνει αποστολή πακέτων κατά μήκος του δικτύου, και καθορισμό διαδρομής πακέτων για κάθε κόμβο. Παράλληλα, το επίπεδο NWK, διαθέτει ένα σύνολο εντολών για θέματα ασφάλειας, περιλαμβάνοντας ασφαλή σύνδεση, αποσύνδεση και επανασύνδεση ενός κόμβου. (Gislason, 2008)

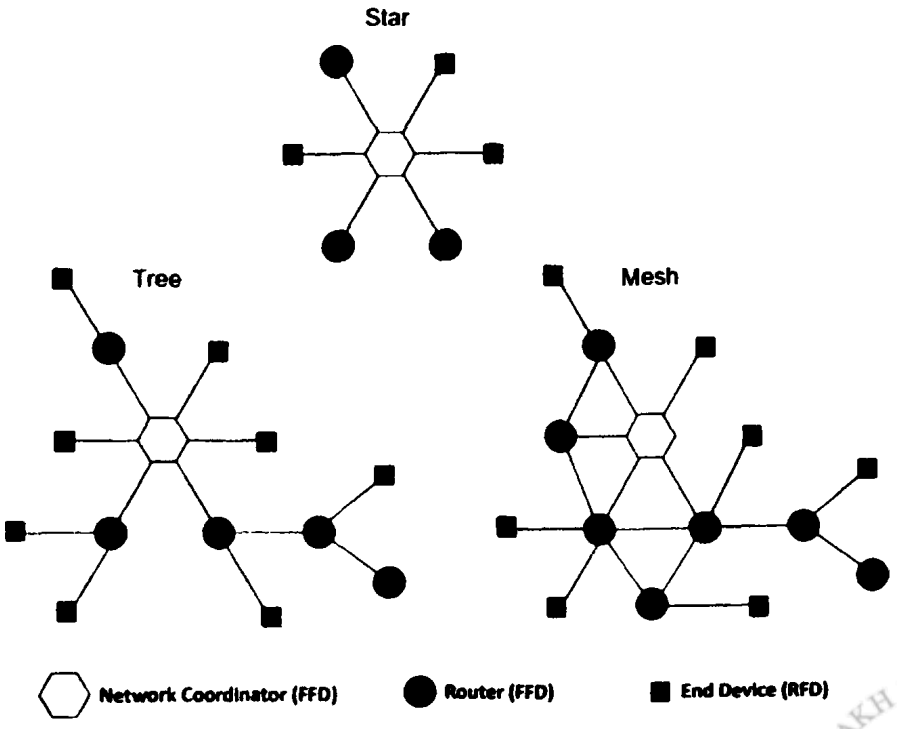
2.8.2.2. Το επίπεδο APL (Application Layer)

Το επίπεδο APL (επίπεδο εφαρμογών) είναι το τελευταίο ιεραρχικά επίπεδο και το πιο περίπλοκο από όλα τα υπόλοιπα. Αποτελείται από τα υπο-επίπεδα Application Framework, ZigBee Device Object (ZDO) και το Application Support Sub-layer (APS) (Χαρτουμπέκης, 2010).

- Application Framework (Πλαίσιο εφαρμογών). Πρόκειται για το περιβάλλον στο οποίο «φιλοξενούνται» αντικείμενα εφαρμογών. Μπορούν να οριστούν έως και 240 διακριτά αντικείμενα εφαρμογών σε διαδοχικές διευθύνσεις με τιμές 1-240. Ορίζονται ακόμα δύο διευθύνσεις. Η μία με τιμή «0» για τη διεπαφή των δεδομένων με το υπο-επίπεδο ZDO, και μία με τιμή «255» για τη διεπαφή των δεδομένων όταν πρόκειται για μετάδοση αυτών σε όλα τα αντικείμενα εφαρμογών. Οι διευθύνσεις με τιμές 241-254 προορίζονται για μελλοντική χρήση.
- Application Support (APS-Υποστήριξη εφαρμογών). Το υπο-επίπεδο APS εξυπηρετεί στην ανταλλαγή δεδομένων μεταξύ δύο ή περισσότερων αντικειμένων εφαρμογών.
- ZigBee device objects (ZDO). Το συγκεκριμένο υπο-επίπεδο προσφέρει τη διεπαφή μεταξύ των αντικειμένων των εφαρμογών, του προφίλ της συσκευής και του υπο-επιπέδου APS. Βρίσκεται μεταξύ του πλαισίου Application Framework και του APS, και ικανοποιεί αιτήματα από όλες τις εφαρμογές που «τρέχουν» στο πρωτόκολλο ZigBee.

2.8.3. Τοπολογίες Δικτύου

Το πρότυπο ZigBee υποστηρίζει τοπολογίες τύπου αστέρα, συστάδας (Cluster Tree) και βρόχων (Mesh). Μία σχηματική αναπαράσταση των τοπολογιών φαίνεται στην εικόνα 9 (Zheng, et al., 2009).



Εικόνα 9

Στην τοπολογία αστέρα, όπως και στο πρότυπο 802.15.4, η επικοινωνία γίνεται μεταξύ των συσκευών και ενός συντονιστή του δικτύου. Ο συντονιστής παίζει επίσης το ρόλο του δρομολογητή για τη μεταφορά δεδομένων μεταξύ των άλλων συσκευών, καθώς αυτές δεν έχουν τη δυνατότητα της απευθείας επικοινωνίας (Eady, 2007).

Στις τοπολογίες Mesh και Cluster Tree ο συντονιστής ευθύνεται για τη λειτουργία του δικτύου και την επιλογή ορισμένων παραμέτρων του δικτύου. Το δίκτυο μπορεί επίσης να επεκταθεί χρησιμοποιώντας δρομολογητές (Routers) ZigBee, οι οποίοι πρέπει να είναι πάντα σε λειτουργία (σε αντίθεση με τις End Devices οι οποίες μπορούν να τεθούν σε κατάσταση αναμονής για εξοικονόμηση ενέργειας). Οι δρομολογητές εξυπηρετούν στη δρομολόγηση μηνυμάτων με πολλαπλά άλματα (multi-hop) από μία συσκευή σε οποιαδήποτε άλλη εντός του δικτύου (Eady, 2007).

Στην τοπολογία Cluster Tree, οι δρομολογητές προωθούν δεδομένα και εντολές χρησιμοποιώντας μία ιεραρχική στρατηγική δρομολόγησης, ενώ η ανακάλυψη των κόμβων και η δρομολόγηση γίνεται με τον αλγόριθμο Ad-hoc On Demand Distance Vector (AODV). Στα πλεονεκτήματα της τοπολογίας συγκαταλέγονται η πολύ χαμηλή κατανάλωση ενέργειας στις τελικές συσκευές (End Devices) και η πολύ αποδοτική δρομολόγηση, ενώ μειονεκτήματα αποτελούν η υψηλή κατανάλωση ενέργειας των δρομολογητών, η μεγάλη δικτυακή κίνηση σε περίπτωση αλλαγών στο δίκτυο που μπορεί να προκαλέσει συγκρούσεις και απώλεια δεδομένων και η αδυναμία λειτουργίας του δικτύου σε περίπτωση βλάβης του συντονιστή (ZigBee Alliance, 2008).

Στην τοπολογία Mesh υποστηρίζεται πλήρως η επικοινωνία peer-to-peer. Σε αντίθεση με την τοπολογία Cluster Tree δεν υπάρχουν ιεραρχικές σχέσεις, κάθε συσκευή μπορεί να επικοινωνήσει με οποιαδήποτε άλλη (Farahani, 2008). Κάθε συσκευή εγκαθιστά συνδέσεις με τις γειτονικές της συσκευές που βέβαια βρίσκονται εντός εμβέλειας. Με τη χρήση αλγόριθμων δρομολόγησης, όπως και στην τοπολογία Cluster Tree, μία συσκευή μπορεί να στείλει δεδομένα σε κάποια άλλη μη-γειτονική (Zheng, et al., 2009).

2.9. Το πρότυπο DigiMesh

2.9.1. Γενικά

Κατά καιρούς πολλά διαφορετικά πρωτόκολλα τοπολογίας mesh έχουν παρουσιαστεί από διάφορες εταιρίες που υλοποιούνται με βάση το πρότυπο 802.15.4 MAC. Ένα από τα πρότυπα αυτά είναι το πρότυπο DigiMesh της εταιρίας Digi International. Είναι ένα πλήρως κατανεμημένο πρότυπο τοπολογίας Mesh όπου όλοι οι κόμβοι μπορούν να εισέλθουν σε κατάσταση αναμονής για εξοικονόμηση ενέργειας καθώς και να προωθούν τα γειτονικά πακέτα (δεδομένων) (David Gascón, 2009).

Το πρότυπο, που προορίζεται για χρήση σε ασύρματα δίκτυα αισθητήρων, εφαρμόζει αρχιτεκτονική peer-to-peer και είναι εξοπλισμένο με προηγμένες δυνατότητες δικτύωσης, συμπεριλαμβανομένης της υποστήριξης κατάστασης αναμονής (sleep mode) στους δρομολογητές (Routers) και πυκνή ανάπτυξη του δικτύου. Η δομή των πακέτων δεδομένων έχει βελτιστοποιηθεί για καλύτερη απόδοση του δικτύου ενώ έχει απλουστευθεί και η διευθυνσιοδότηση (Digi International, 2008).

Για τη δρομολόγηση των πακέτων και την ανακάλυψη νέων κόμβων χρησιμοποιεί μία παραλλαγή του αλγορίθμου AODV, έτσι οι πίνακες δρομολόγησης συμπληρώνονται μόνο για τους προορισμούς που χρειάζεται. Οι κόμβοι που δεν χρησιμοποιούνται δεν καταχωρούνται στους πίνακες δρομολόγησης, ενώ αντίθετα, κόμβοι που χρησιμοποιούνται συχνά ανανεώνουν επίσης συχνά τις καταχωρήσεις στους πίνακες για βελτιστοποίηση της αποδοτικότητάς τους. Για αυτόν το λόγο το πρότυπο αναφέρεται ως peer-to-peer mesh αντί για cluster-tree. Η πρόσβαση στο κανάλι γίνεται με ένα είδος συγχρονισμένου CSMA

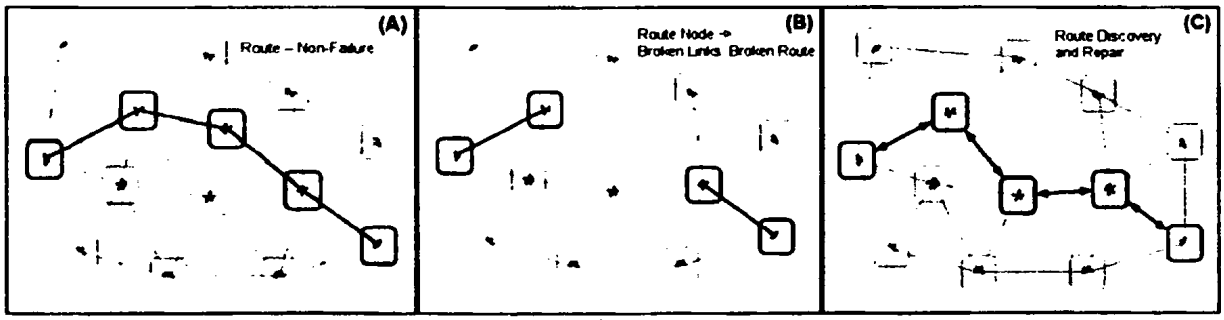
(Carrier sense multiple access) διευκολύνοντας την καταιγιστική εκπομπή πακέτων με κόστος κάποιες συγκρούσεις. (Young, 2008).

Το πρότυπο DigiMesh αναπτύχθηκε για να αντιμετωπίσει τις ανάγκες μιας ευρείας κατηγορίας εφαρμογών ασύρματης δικτύωσης. Τα σημαντικότερα πλεονεκτήματά του συνοψίζονται στα εξής χαρακτηριστικά (Digi International, 2008):

- Ισχυρή δικτύωση τοπολογίας «Mesh».
- Ενεργειακά βελτιστοποιημένο πρότυπο με δρομολογητές (Routers) που υποστηρίζουν κατάσταση αναμονής (sleep mode) για εφαρμογές με περιορισμένα ενεργειακά αποθέματα.
- Εύκολο στη χρήση, απλοποιεί τη δικτύωση τοπολογίας «Mesh» καθώς όλες οι συσκευές είναι δρομολογητές (Routers).
- Μεγαλύτερη αυτονομία δικτύου καθώς δεν υπάρχει διαχειριστής δικτύου (Coordinator) και το συγχρονισμό πραγματοποιεί ένας δρομολογητής με διαδικασία «επιλογής».
- Οι συσκευές διατίθενται σε δύο εκδόσεις, μία στα 900MHz FHSS και μία στα 2.4GHz DSSS.

2.9.2. Δρομολόγηση

Η εικόνα 10 παρουσιάζει την αντιμετώπιση μιας αποτυχημένης δρομολόγησης. Στο τμήμα (A) της εικόνας φαίνεται η αρχική διαμόρφωση του δικτύου όπου έχει εγκαθιδρυθεί μία διαδρομή από ένα σημείο σε ένα άλλο. Το τμήμα (B) της εικόνας παρουσιάζει την αποτυχία δρομολόγησης καθώς ένας από του κόμβους έχει αφαιρεθεί από άγνωστη αιτία, καταργώντας έτσι τη σύνδεση στο κέντρο της διαδρομής. Τελικά, στο τμήμα (C) της εικόνας παρουσιάζεται πως η διαδρομή αναδιαμορφώνεται χρησιμοποιώντας ένα δρόμο που δεν υπήρχε πριν. Οι συσχετισμοί των κόμβων υπήρχαν αλλά αφού δεν είχαν χρησιμοποιηθεί ποτέ ανακαλύφθηκαν εκ νέου με τη χρήση του αλγορίθμου AODV μετά την αποτυχία (Young, 2008).



Εικόνα 10

2.9.3. Πλεονεκτήματα και μειονεκτήματα του προτύπου DigiMesh

Στις επόμενες δύο παραγράφους αναφέρονται συνοπτικά τα πλεονεκτήματα και τα μειονεκτήματα του νέου αυτού προτύπου δικτύωσης.

Πλεονεκτήματα: Κάθε κόμβος είναι δρομολογητής χαμηλής κατανάλωσης. Εφόσον για κάθε πακέτο αποστέλλεται αναφορά παράδοσης και οι διαδρομές καθορίζονται μόνο όπου χρειάζεται, το δίκτυο δεν κατακλύζεται με περιττή κίνηση, πολύ σημαντικό αν οι δρομολογητές τροφοδοτούνται από μπαταρία και μπαίνουν σε κατάσταση αναμονής. Η ασφάλεια ανταποκρίνεται επίσης στις απαιτήσεις της κρυπτογράφησης και της

αυθεντικοποίησης. Η αξιοπιστία φτάνει μέχρι και το 99.99%. Τέλος, το σύστημα υποστηρίζει μεγαλύτερου μεγέθους πακέτα δεδομένων με υποστήριξη κατακερματισμού μηνυμάτων.

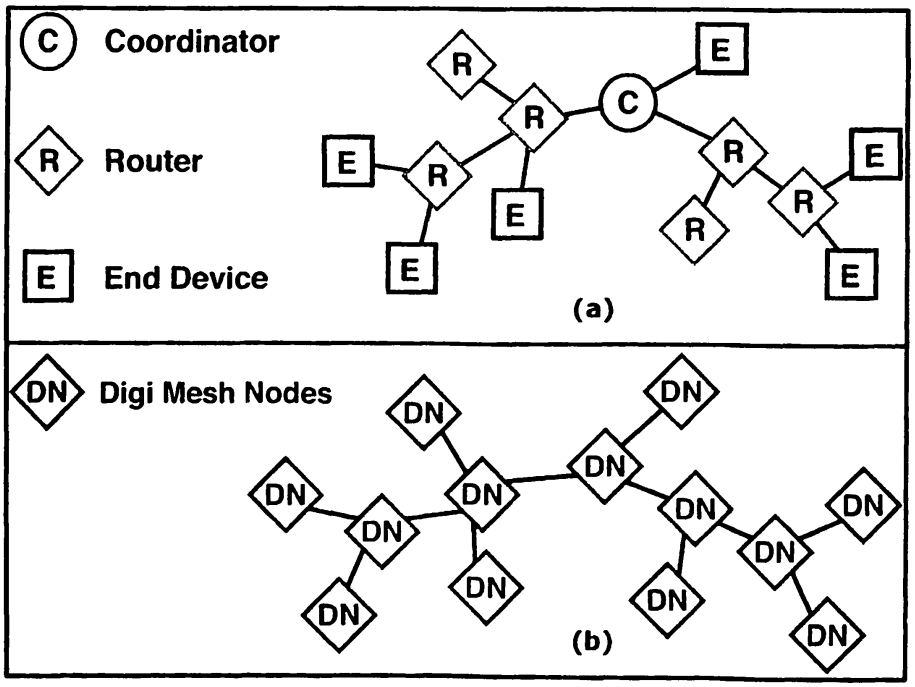
Μειονεκτήματα: Δυστυχώς, η επαρκής διαχείριση ενέργειας σημαίνει μεγάλη και μη ντετερμινιστική καθυστέρηση. Αν και η ρυθμοαπόδοση (μέσος ρυθμός επιτυχών μεταδόσεων στο κανάλι επικοινωνίας, αγγλικός όρος:throughput) δεν περιορίζεται από τα time slots, είναι περιορισμένη, ανάλογα με το φόρτο του δικτύου και τις ανακαλύψεις νέων διαδρομών. Το δίκτυο μπορεί να κλιμακωθεί σε μέτριο μέγεθος περίπου 500+ κόμβων και μπορεί να είναι πολύ μεγαλύτερο εάν η κίνηση είναι μικρή και η κίνηση των μηνυμάτων δεν αλλάζει πολύ (Young, 2008).

2.10. Σύγκριση των προτύπων DigiMesh και ZigBee

Αν και τα δύο πρότυπα έχουν πολλά κοινά χαρακτηριστικά, όπως για παράδειγμα ότι και τα δύο βασίζονται στο επίπεδο 802.15.4 MAC, υπάρχουν σημαντικές διαφορές μεταξύ τους, οι οποίες θα πρέπει να ληφθούν σοβαρά υπ' όψη, όταν σχεδιάζεται ένα δίκτυο και επιλέγεται ποιο πρότυπο θα χρησιμοποιηθεί.

Το πρότυπο ZigBee ορίζει τρείς τύπους κόμβων: Το συντονιστή του δικτύου ή αλλιώς Coordinator (FFD), κόμβους που λειτουργούν ως δρομολογητές (Routers-FFD) και απλούς κόμβους (End Devices-RFD). Σε κάθε δίκτυο μπορεί να υπάρχει μόνο ένας συντονιστής. Οι απλοί κόμβοι αν και μπορούν να επικοινωνήσουν με το συντονιστή και τους δρομολογητές δεν μπορούν να προωθήσουν πακέτα γειτονικών κόμβων. Κατά συνέπεια το κόστος των απλών κόμβων είναι μειωμένο (Εικόνα 11a).

Το πρότυπο DigiMesh χρησιμοποιεί μόνο έναν τύπο κόμβου. Ως ομοιογενές δίκτυο, κάθε κόμβος μπορεί να δρομολογήσει γειτονικά πακέτα από και προς οποιοδήποτε κόμβο. Καθώς δεν υπάρχουν σχέσεις «γονέα-παιδιού» μπορεί να τοποθετηθεί οπουδήποτε χωρίς να χρειάζεται κάποιος προσεκτικός σχεδιασμός του δικτύου (Εικόνα 11b) (Digi International Inc., 2008).



Εικόνα 11: Τοπολογία δικτύου με το πρότυπο ZigBee (a) και με το πρότυπο DigiMesh (b)

Οι σημαντικότερες διαφορές των δύο προτύπων μπορούν να συνοψιστούν στον παρακάτω πίνακα (πίνακας 1) (Digi International Inc., 2008).

	ZigBee	DigiMesh
Τύποι κόμβων, πλεονεκτήματα	Coordinator, Routers και End Devices. Μειωμένο κόστος εξαιτίας της περιορισμένης λειτουργικότητας των End Devices.	Ένας τύπος κόμβου. Μεγαλύτερη ευελιξία στην επέκταση και παραμετροποίηση του δικτύου.
Κατάσταση αναμονής	Μόνο τα End Devices μπορούν να μπουν σε κατάσταση αναμονής	Όλοι οι κόμβοι μπορούν να μπουν σε κατάσταση αναμονής.
Over-the-Air αναβάθμιση firmware	Ναι	Όχι
Εμβέλεια	Έως 3,2km	Έως 64km (XTend™)
Μέγεθος πακέτου	80 bytes	Έως 256 bytes
Συχνότητες λειτουργίας και ρυθμός μετάδοσης	2.4GHz (250kbps), 900MHz (40kbps) και 868MHz (20kbps)	2.4GHz (250kbps) 900MHz (10, 125, 150kbps)
Ασφάλεια	AES κρυπτογράφηση	AES κρυπτογράφηση
Συμβατότητα	Πιθανή συμβατότητα με προϊόντα ZigBee άλλων εταιριών	Συμβατότητα μόνο με προϊόντα DigiMesh (της εταιρίας Digi)
Ανοχή σε παρεμβολές	Direct-Sequence Spread Spectrum (DSSS)	900 MHz: Frequency-Hopping Spread Spectrum (FHSS). 2.4GHz: Direct-Sequence Spread Spectrum (DSSS)
Διευθυνσιοδότηση	Δύο layer: MAC διεύθυνση (64bit) και Network διεύθυνση (16bit).	MAC διεύθυνση μόνο (64bit).
Συντήρηση	Υπάρχουν αρκετά διαγνωστικά εργαλεία στην αγορά	Δεν υπάρχουν αρκετά διαγνωστικά εργαλεία στην αγορά, παρόλα αυτά, η απλούστερη διευθυνσιοδότηση βοηθά στον εντοπισμό προβλημάτων.

Πίνακας 1

3. Η πλατφόρμα ανοιχτού υλικού/λογισμικού Arduino

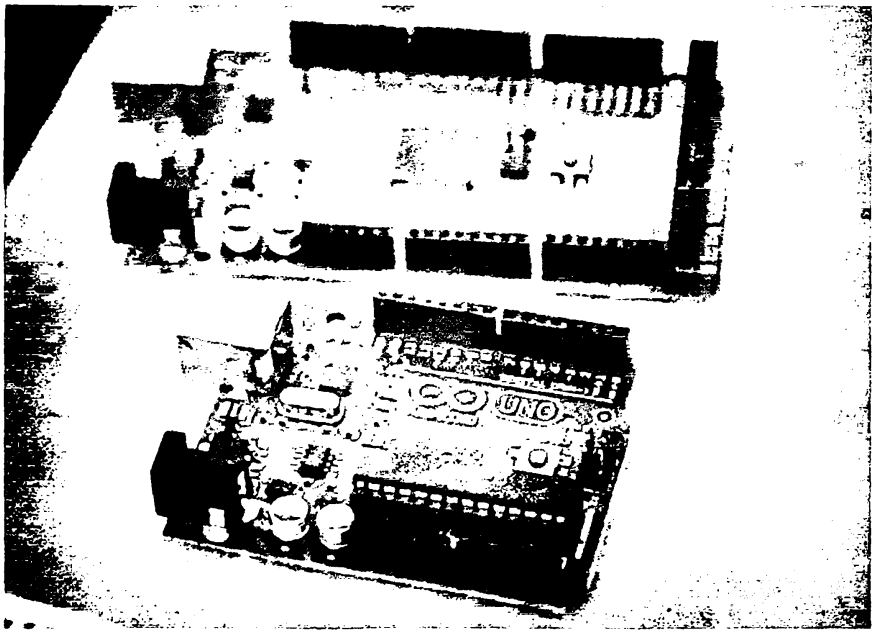
3.1. Γενικές πληροφορίες για την πλατφόρμα Arduino

Το Arduino είναι μια ελεύθερη (open-source) υπολογιστική πλατφόρμα βασισμένη σε μι απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή ο οποίος φέρει εισόδους/ εξόδου και η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τ C++ με κάποιες μετατροπές) (Banzi, 2008).

Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη αυτόνομων διαδραστικώ εφαρμογών, ανιχνεύοντας το φυσικό περιβάλλον με τη χρήση αισθητήρων ή και διακοπτώ και αλληλεπιδρώντας με αυτό με τη χρήση κινητήρων, φώτων και άλλων φυσικών εξόδων Παράλληλα μπορεί να επικοινωνεί με κάποιο άλλο υπολογιστικό σύστημα, όπως γι παράδειγμα με έναν Η/Υ, ανταλλάσοντας δεδομένα (Arduino, 2011). Το λογότυπο το Arduino φαίνεται στην εικόνα 12.

Εικόνα 12: Το λογότυπο του Arduino

Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες Τα σχηματικά διαγράμματα καθώς και πληροφορίες για το υλικό είναι ελεύθερα διαθέσιμ για όσους θέλουν να συναρμολογήσουν το Arduino μόνοι τους ή ακόμα και να σχεδιάσου μία δική τους τροποποιημένη έκδοση. Οι δύο επίσημες εκδόσεις φαίνονται στην εικόνα 13.



Εικόνα13: Arduino Uno (κάτω), Arduino Mega2560 (επάνω).

3.2. Πλεονεκτήματα του συστήματος Arduino

Υπάρχουν στο εμπόριο διαθέσιμες αρκετές άλλες υπολογιστικές πλατφόρμες με τις ίδιες δυνατότητες του Arduino, όπως Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard κλπ. Όλες αυτές οι πλατφόρμες, συμπεριλαμβανομένου του Arduino, απλοποιούν τον προγραμματισμό του μικροελεγκτή. Παρόλα αυτά, το σύστημα Arduino έχει ορισμένα πλεονεκτήματα ακόμα έναντι των άλλων, όπως:

- Μικρό κόστος. Η υπολογιστική πλατφόρμα Arduino είναι σχετικά φθηνή σε σχέση με άλλες. Στην πιο φθηνή εκδοχή του, κοστίζει λιγότερο από 25€.
- Συμβατότητα. Το Arduino προσφέρει συμβατότητα με όλα τα λειτουργικά συστήματα όπως Windows, Macintosh OSX και Linux, σε αντίθεση με τις περισσότερες πλατφόρμες οι οποίες «τρέχουν» μόνο σε Windows.
- Απλουστευμένο προγραμματιστικό περιβάλλον. Το προγραμματιστικό περιβάλλον του Arduino είναι αρκετά απλό στη χρήση για ένα αρχάριο χρήστη, αλλά και πολύ ευέλικτο για κάποιον πιο προχωρημένο.
- Λογισμικό ανοιχτού κώδικα. Το λογισμικό του Arduino είναι δημοσιευμένο στο διαδίκτυο ως ελεύθερο λογισμικό ανοιχτού κώδικα, διαθέσιμο για τροποποίηση από πιο έμπειρους προγραμματιστές. Η γλώσσα προγραμματισμού μπορεί επίσης να επεκταθεί με τη χρήση βιβλιοθηκών C++.
- «Ανοιχτό υλικό». Οι αναπτυξιακές πλακέτες του Arduino στηρίζονται κυρίως στους μικροελεγκτές ATmega328 και ATmega1280. Τα σχηματικά διαγράμματα είναι ελεύθερα διαθέσιμα στο διαδίκτυο υπό την άδεια «Creative Commons Attribution-Share Alike 3.0» καθιστώντας εφικτό στους έμπειρους χρήστες και σχεδιαστές υλικού να τροποποιήσουν και να δημιουργήσουν τη δική τους έκδοση (Cre11).

3.3. Υλικό

Η πλακέτα του Arduino αποτελείται από ένα μικροελεγκτή Atmel AVR(*) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωσή του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης (voltage regulator) 5V και ένα κρυσταλλικό ταλαντωτή 16MHz. Ο μικροελεγκτής προγραμματίζεται με ένα πρόγραμμα έναρξης λειτουργίας (bootloader), ώστε να μην χρειάζεται εξωτερικός προγραμματιστής. Ο προγραμματισμός του μικροελεγκτή πραγματοποιείται με τη χρήση της σειριακής θύρας.

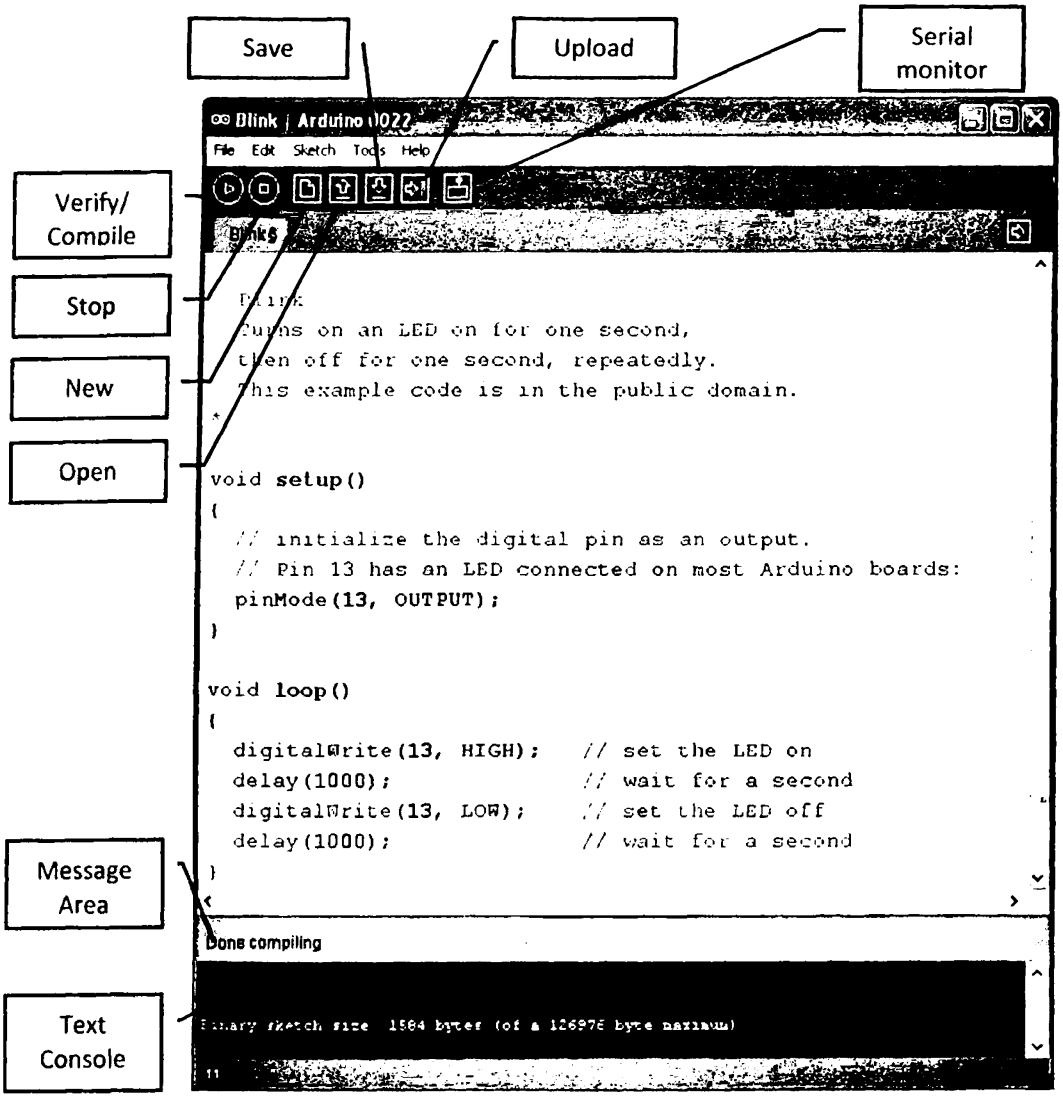
Η πλακέτα του Arduino διαθέτει αρκετές επαφές εισόδου/εξόδου για συνδυασμό με άλλα κυκλώματα. Υπάρχουν επίσης έτοιμες για χρήση (plug-in) πλακέτες εφαρμογών, γνωστές και σαν «shields», διαθέσιμες στο εμπόριο.

3.4. Λογισμικό

Το ολοκληρωμένο περιβάλλον διαχείρισης IDE (Integrated Development Environment) του Arduino είναι γραμμένο στη γλώσσα προγραμματισμού Java και μπορεί να τρέξει σε πολλαπλές πλατφόρμες. Το περιβάλλον ανάπτυξης είναι βασισμένο στη γλώσσα προγραμματισμού Processing, η οποία προέρχεται από τη γλώσσα Wiring. Περιλαμβάνει έναν κειμενογράφο, μια περιοχή μηνυμάτων, μία κονσόλα διαχείρισης κειμένου, μια γραμμή εργαλείων και μια σειρά από μενού (Εικόνα 14).

Τα προγράμματα που γράφονται με το προγραμματιστικό περιβάλλον του Arduino ονομάζονται sketches. Η περιοχή μηνυμάτων εμφανίζει τα λάθη που υπάρχουν στον κώδικα. Η κονσόλα διαχείρισης κειμένου εμφανίζει τις εξόδους που θα έχει το κείμενο που έχει γραφτεί στο αναπτυξιακό περιβάλλον συμπεριλαμβανομένων μηνυμάτων λαθών και άλλων πληροφοριών. Η γραμμή εργαλείων περιλαμβάνει κουμπιά που επιτρέπουν την

επαλήθευση και τη φόρτωση (upload) των προγραμμάτων, την δημιουργία, το άνοιγμα και την αποθήκευση των προγραμμάτων (sketches) καθώς και το άνοιγμα ενός σειριακού παραθύρου. Επιπλέον επιλογές υπάρχουν στα εξής πέντε μενού: File, Edit, Sketch, Tools και Help (McRoberts, 2010).



Εικόνα 14: Το αναπτυξιακό περιβάλλον του Arduino

3.4.1. Sketch

Το μενού περιέχει τις εξής επιλογές:

- Verify/Compile: ελέγχει το πρόγραμμα για λάθη.
- Import Library: Εισάγει μία νέα βιβλιοθήκη στο πρόγραμμα εισάγοντας την εντολή #include και μετά το όνομα της βιβλιοθήκης που επιλέξαμε.
- Show Sketch Folder: ανοίγει τον φάκελο με το sketch που επιθυμούμε
- Add File: Εισάγει ένα ήδη υπάρχον αρχείο στο project.

3.4.2. Tools

Το μενού εργαλείων (Tools) περιέχει τις εξής επιλογές:

- Auto Format: διαμορφώνει αυτόματα τον κώδικα.
- Board: μπορούμε να επιλέξουμε την έκδοση του Arduino κυκλώματος που χρησιμοποιούμε.

- Serial port (Σειριακή θύρα): περιέχει όλες τις σειριακές συσκευές που είναι συνδεδεμένες στον υπολογιστή παρέχοντας την δυνατότητα επιλογής.
- Burn Bootloader: εγγραφή bootloader

3.4.3. Περιοχή Sketchbook

Το sketchbook είναι περιοχή στην οποία αποθηκεύονται τα προγράμματα ή τα sketches που έχουν δημιουργηθεί. Επιτρέπει επίσης τη διαχείριση των sketches που περιέχουν περισσότερα από ένα αρχεία. Αυτά μπορεί να είναι τυπικά αρχεία με την επέκταση του Arduino (.pde), αρχεία σε C (.c), αρχεία σε C++ (.cpp) ή αρχεία header (.h).

3.4.4. Uploading

Για να γίνει σωστά η μεταφορά ενός προγράμματος στο μικροελεγκτή θα πρέπει πρώτα να επιλέξουμε την σωστή έκδοση του Arduino board από το μενού των εργαλείων (Tools) καθώς και τη σειριακή θύρα στην οποία έχουμε συνδεδεμένο το Arduino. Μόλις γίνουν οι σωστές ρυθμίσεις «πιέζουμε» το κουμπί Upload από τη γραμμή εργαλείων.

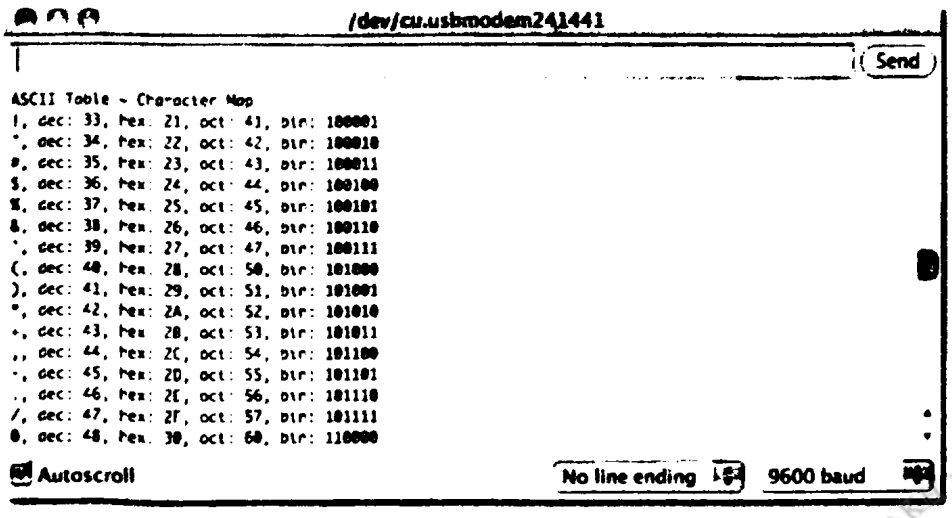
Όταν ανεβάζουμε ένα πρόγραμμα (sketch) χρησιμοποιούμε το πρόγραμμα bootloader, ένα μικρό πρόγραμμα που έχει ήδη φορτωθεί στο μικροελεγκτή από εξωτερικό προγραμματιστή. Το πρόγραμμα bootloader μας επιτρέπει να ανεβάζουμε κώδικα χωρίς να χρησιμοποιήσουμε επιπλέον συσκευές. Είναι ενεργό για μερικά δευτερόλεπτα και εκτελεί κάθε φορά το πρόγραμμα που έχει φορτωθεί στο μικροελεγκτή την τελευταία φορά.

3.4.5. Βιβλιοθήκες

Οι βιβλιοθήκες προσφέρουν επιπλέον λειτουργίες στο πρόγραμμα. Η εισαγωγή τους γίνεται με την εντολή #include και μετά ακολουθεί το όνομα της βιβλιοθήκης. Όταν οι βιβλιοθήκες φορτώνονται στο μικροελεγκτή μαζί με το πρόγραμμα, αυξάνουν τη μνήμη που καταλαμβάνεται. Κάποιες βιβλιοθήκες περιλαμβάνονται στο περιβάλλον του Arduino κάποιες άλλες όμως μπορούν να βρεθούν σε διάφορους άλλους ιστοτόπους.

3.4.6. Serial Monitor

Εμφανίζει τα δεδομένα από τη σειριακή θύρα που λαμβάνονται από το Arduino. Για να στείλουμε δεδομένα στο Arduino γράφουμε το κείμενο και «πιέζουμε» send ή enter. Για να γίνει σωστή λήψη και αποστολή δεδομένων θα πρέπει το ρυθμός μετάδοσης (baud rate) να είναι ίδιος με τον αντίστοιχο ρυθμό που έχουμε ορίσει στο πρόγραμμα στο Serial.begin(baud rate) (Margolis, 2011).



Εικόνα 15:Το Serial Monitor του Arduino IDE

4. Διάρθρωση Δικτύου

4.1. Γενικά

Τα προηγούμενα κεφάλαια επικεντρώθηκαν στα χαρακτηριστικά και τις απαιτήσεις με σκοπό την υλοποίηση ενός Ασύρματου Δικτύου Αισθητήρων. Παράλληλα ορίστηκαν τα πλεονεκτήματα και τα μειονεκτήματα των πιο διαδεδομένων προτύπων επικοινωνίας. Σε αυτό το κεφάλαιο θα ασχοληθούμε με το υπο κατασκευή δίκτυο. Θα γίνει αναφορά αρχικά στις προδιαγραφές-χαρακτηριστικά του δικτύου και εν συνεχεία στη λειτουργία του. Τέλος, θα ακολουθήσει η περιγραφή του ασύρματου πομποδέκτη (Xbee) που επιλέχθηκε για να χρησιμοποιηθεί στο υπό κατασκευή δίκτυο.

4.2. Προδιαγραφές Συστήματος

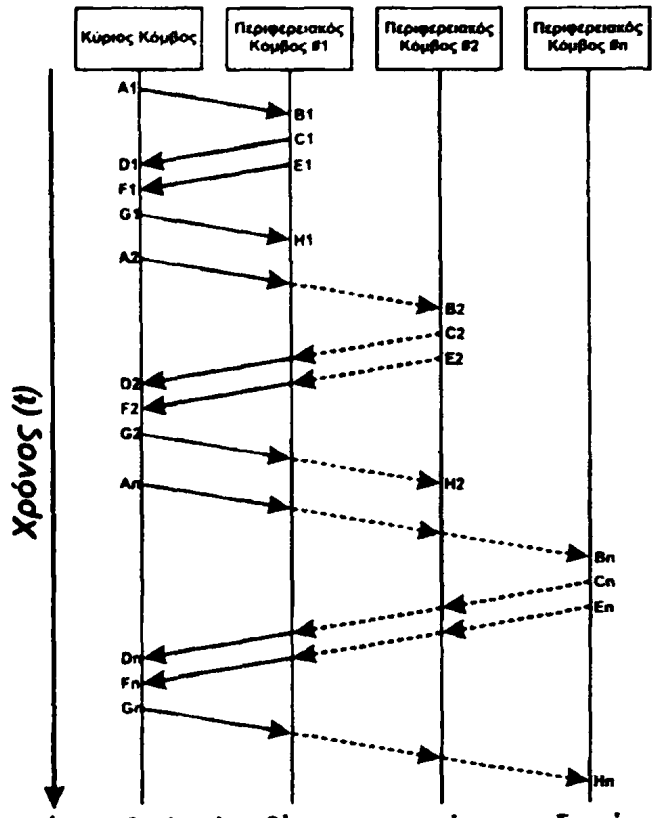
Απώτερος στόχος της παρούσας διπλωματικής εργασίας ήταν η κατασκευή ενός Ασύρματου Δικτύου Αισθητήρων σε επίπεδο υλικού (hardware), αλλά και λογισμικού (software) με δυνατότητα χρήσης του σε πολλές και διαφορετικές εφαρμογές. Στην πράξη, δίνεται η δυνατότητα στο χρήστη να μετρά το φυσικό μέγεθος που τον ενδιαφέρει προσαρμόζοντας κάθε φορά τον κατάλληλο αισθητήρα. Συγκεκριμένα, το εν λόγω σύστημα επιτυγχάνει:

- Συμβατότητα μεταξύ διαφορετικών λειτουργικών συστημάτων καθώς και λειτουργία χωρίς την ανάγκη εγκατάστασης κάποιου προγράμματος ή οδηγού (driver). Για την ανάγκη αυτή, η διεπαφή χρήστη (user interface) θα πρέπει να είναι βασισμένη σε ιστοσελίδα (Web-based User Interface) και να «τρέχει» μέσω ενός Web Browser.
- Δυνατότητα απομακρυσμένης πρόσβασης στο WSN (δεδομένα και διαχείριση) μέσω Internet. Αυτό μπορεί να πραγματοποιηθεί με την εκχώρηση μιας πραγματικής διεύθυνσης IP στον Κύριο Κόμβο ή κάνοντας προώθηση (Port Forward) της θύρας και της τοπικής διεύθυνσης IP που έχει εκχωρηθεί στον Κύριο Κόμβο από το δρομολογητή (Router) του LAN.
- Αποθήκευση των μετρήσεων σε αρχεία με τέτοια μορφή ώστε να είναι προσβάσιμα αλλά και επεξεργάσιμα από προγράμματα πέραν της διεπαφής χρήστη, όπως για παράδειγμα η μορφή .CSV. Η συγκεκριμένη μορφή αρχείου μπορεί να εισαχθεί στο Microsoft Office Excel ή το Open Office Calc.
- Δυνατότητα επαναπρογραμματισμού των συσκευών (Κύριος και περιφερειακοί κόμβοι) χωρίς τη χρήση κάποιας εξωτερικής συσκευής (programmer).
- Δυνατότητα αλλαγής βασικών παραμέτρων λειτουργίας χωρίς την ανάγκη επαναπρογραμματισμού.
- Αξιόπιστη λειτουργία χωρίς «κολλήματα». Τόσο στον Κύριο Κόμβο όσο και στους περιφερειακούς, ο κώδικας που «τρέχουν» δεν θα πρέπει να «κολλάει». Στην περίπτωση που κολλήσει όμως θα πρέπει το σύστημα να επανεκκινηθεί αυτόνομα.
- Απλοποιημένη δομή δικτύου με χρήση ενός μόνο τύπου κόμβων.
- Αντοχή των περιφερειακών κόμβων σε καιρικά φαινόμενα με χρήση κουτιών που είναι αεροστεγή και υδατοστεγή.

- Αντοχή των περιφερειακών κόμβων σε υψηλές και χαμηλές θερμοκρασίες με την προσεκτική επιλογή των εξαρτημάτων και των ολοκληρωμένων που τους αποτελούν ώστε να ικανοποιούν τα βιομηχανικά πρότυπα (-40°C ~ +125°C).
- Ενεργειακή αυτονομία των περιφερειακών κόμβων με χρήση ανανεώσιμων πηγών ενέργειας όπως για παράδειγμα οι φωτοβολταϊκές κυψέλες.
- Συμβατότητα του κάθε περιφερειακού κόμβου με τους πιο διαδεδομένους διαύλους επικοινωνίας όπως I2C, SPI, 1-Wire και δυνατότητα μέτρησης αναλογικού σήματος για σύνδεση με τους αισθητήρες.
- Κατάτμηση των σχηματικών διαγραμμάτων της κάθε συσκευής (Κύριος και περιφερειακοί κόμβοι) σε δευτερεύοντα κυκλώματα. Αυτός ο τρόπος σχεδίασης εξυπηρετεί στην εύκολη εύρεση σφαλμάτων και οικονομική αντικατάσταση ελαττωματικών κυκλωμάτων.

4.3. Λειτουργία Συστήματος

Κατά την εκκίνηση του συστήματος, ο Κύριος Κόμβος αφού εκτελέσει κάποιες διαγνωστικές εργασίες, αναζητά συμβατούς περιφερειακούς κόμβους, ώστε να τους ενσωματώσει στο ασύρματο δίκτυο αισθητήρων. Η αναζήτηση πραγματοποιείται με την αποστολή μίας εντολής ταυτοποίησης. Ο κάθε περιφερειακός κόμβος που λαμβάνει την εντολή αυτή απαντά στέλνοντας ένα μήνυμα το οποίο περιέχει το όνομά του (ορίζεται από το χρήστη) και τη διεύθυνσή του. Τα στοιχεία αυτά αποθηκεύονται σε μία βάση δεδομένων στον κύριο κόμβο. Όταν ολοκληρωθεί και αυτή η εργασία, το σύστημα επικοινωνεί με τους περιφερειακούς κόμβους για τη λήψη των μετρήσεων από τους αισθητήρες που φέρει ο κάθε περιφερειακός κόμβος. Το διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών μετρήσεων μπορεί να οριστεί από τη διεπαφή χρήστη (κεφάλαιο 7) σε δευτερόλεπτα. Στην εικόνα 16 παρουσιάζεται γραφικά η ακολουθία που εφαρμόζεται κατά την επικοινωνία του δικτύου.



Εικόνα 16: Ακολουθία επικοινωνίας του δικτύου

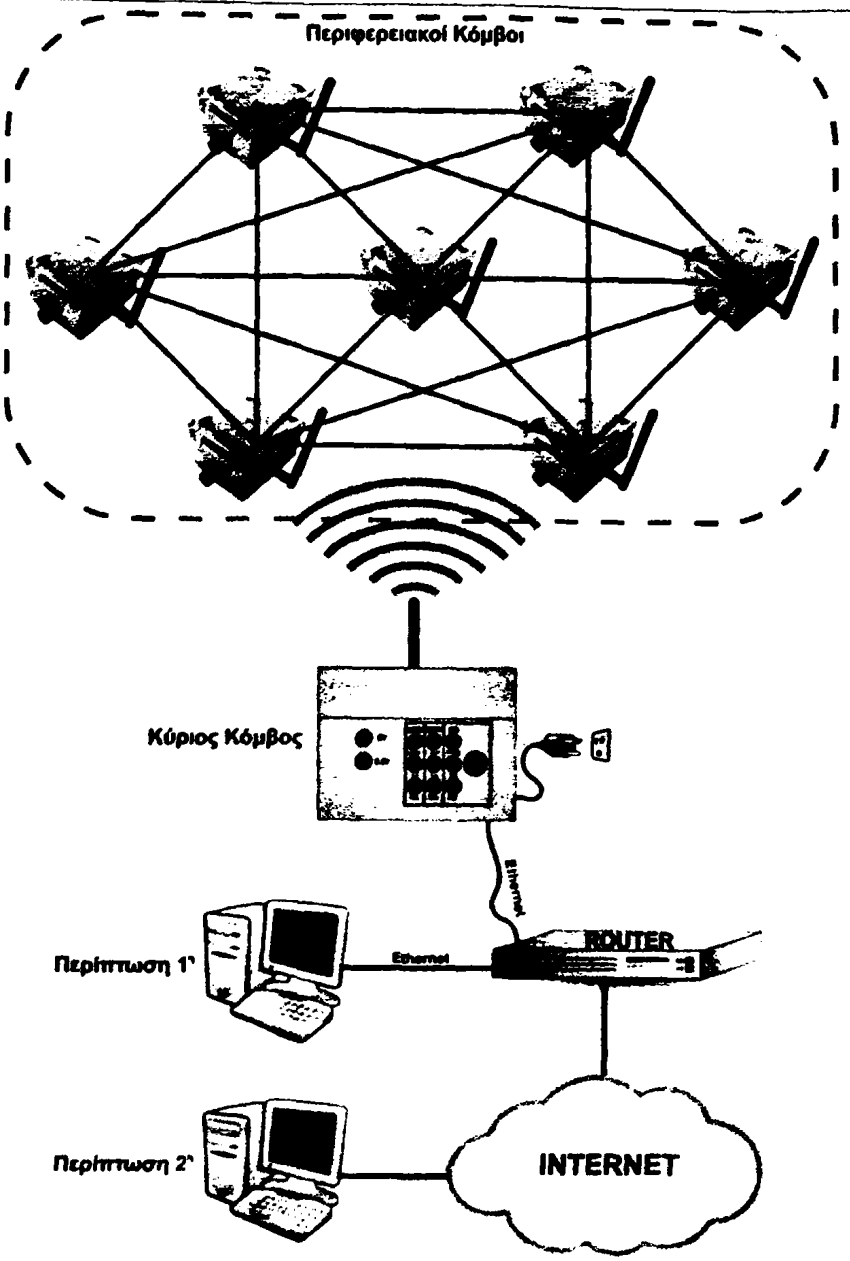
Στη συνέχεια περιγράφεται αναλυτικά η ακολουθία επικοινωνίας του δικτύου:

- **An:** Ο κύριος κόμβος αποστέλλει στη διεύθυνση του κόμβου n τον χαρακτήρα «|» ο οποίος σηματοδοτεί την έναρξη μίας νέας μέτρησης και δύο ακόμα bytes τα οποία αντιπροσωπεύουν το διάστημα μεταξύ δύο διαδοχικών μετρήσεων. Τα δύο αυτά bytes ανασυνδέονται από τον περιφερειακό κόμβο n σε έναν ακέραιο αριθμό (integer μεγέθους 2^{16}). Η δεκαδική αναπαράσταση του αριθμού αυτού αντιπροσωπεύει το διάστημα μεταξύ δύο διαδοχικών μετρήσεων. Ο περιφερειακός κόμβος χρησιμοποιεί τον αριθμό για να υπολογίσει πόσο χρόνο πρέπει να είναι ενεργός και πόσο χρόνο πρέπει να είναι σε κατάσταση αναμονής ώστε να εξοικονομεί ενέργεια.
- **Bn:** Ο περιφερειακός κόμβος n λαμβάνει το χαρακτήρα «|» και παίρνει μία νέα μέτρηση από τον αισθητήρα που είναι συνδεδεμένος σ' αυτόν. Τα δύο επόμενα bytes που λαμβάνει ανασυνδέονται σε έναν ακέραιο αριθμό (integer μεγέθους 2^{16}). Η δεκαδική αναπαράσταση του αριθμού αυτού αντιπροσωπεύει το διάστημα μεταξύ δύο διαδοχικών μετρήσεων σε δευτερόλεπτα. Ο περιφερειακός κόμβος χρησιμοποιεί τον αριθμό για να υπολογίσει πόσο χρόνο πρέπει να είναι ενεργός και πόσο χρόνο πρέπει να είναι σε κατάσταση αναμονής ώστε να εξοικονομεί ενέργεια.
- **Cn:** Ο περιφερειακός κόμβος αποστέλλει μία αναφορά παράδοσης στον κύριο κόμβο για το μήνυμα που μόλις έλαβε.
- **Dn:** Ο κύριος κόμβος λαμβάνει την αναφορά παράδοσης από τον περιφερειακό κόμβο για το μήνυμα που έστειλε στο στάδιο An.
- **En:** Ο περιφερειακός κόμβος αποστέλλει τα δεδομένα της μέτρησης που πραγματοποίησε ακολουθούμενα από ένα byte. Η δεκαδική αναπαράσταση του byte (0-2^8) πληροφορεί το σύστημα για την επί τοις εκατό (%) διαθέσιμη ενέργεια στη μπαταρία του περιφερειακού κόμβου.
- **Fn:** Ο κύριος κόμβος λαμβάνει το μήνυμα, αποθηκεύει τα δεδομένα στην κάρτα μνήμης microSD και προβάλει την επί τοις εκατό διαθέσιμη ενέργεια του περιφερειακού κόμβου στη διεπαφή χρήστη.
- **Gn:** Ο κύριος κόμβος αποστέλλει μία αναφορά παράδοσης στον περιφερειακό κόμβο n για το μήνυμα που μόλις έλαβε.
- **Hn:** Ο περιφερειακός κόμβος λαμβάνει την αναφορά παράδοσης από τον κύριο κόμβο για το μήνυμα που έστειλε στο στάδιο En και τίθεται σε κατάσταση αναμονής.

Η ακολουθία που μόλις περιγράφηκε, επαναλαμβάνεται για όσο διάστημα είναι ενεργό το σύστημα. Οι διακεκομμένες γραμμές συμβολίζουν τη δρομολόγηση μηνυμάτων στο δίκτυο με πολλαπλά άλματα (multi-hop).

4.4. Συνδεσμολογία συστήματος

Στην εικόνα 17 παρουσιάζεται γραφικά η διάρθρωση του υπό κατασκευή δικτύου. Αποτελείται από έναν Κύριο Κόμβο (κεφάλαιο 5) και ένα σύμπλεγμα πολλών περιφερειακών κόμβων (κεφάλαιο 6). Ο Κύριος Κόμβος λειτουργεί ως συντονιστής του συστήματος και του δικτύου. Είναι υπεύθυνος για την εγκαθίδρυση του δικτύου για την αποδοτικότερη δρομολόγηση των δεδομένων (από πλευράς ρυθμοαπόδοσης και εξοικονόμησης ενέργειας), την αποθήκευση των μετρήσεων, τη φιλοξενία της διεπαφής χρήστη (κεφάλαιο 7) καθώς και την εκτέλεση βασικών διαγνωστικών λειτουργιών.



Εικόνα 17: Διάρθρωση δικτύου

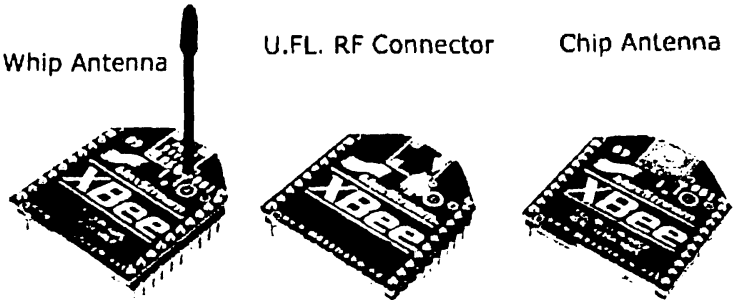
Στην πρώτη περίπτωση που παρουσιάζεται στην εικόνα 17, ο χρήστης συνδέεται στο σύστημα με τη χρήση ενός Η/Υ μέσω του τοπικού δικτύου (LAN). Η προεπιλεγμένη διεύθυνση IP που θα πρέπει να πληκτρολογήσει στο πρόγραμμα περιήγησης στο διαδίκτυο ώστε να εμφανιστεί η διεπαφή χρήστη είναι η 192.168.2.100.

Στη δεύτερη περίπτωση που παρουσιάζεται στην εικόνα 16, ο χρήστης συνδέεται στο σύστημο με τη χρήση ενός Η/Υ απομακρυσμένα μέσω του διαδικτύου (Internet). Αυτό προϋποθέτει ότι ο δρομολογητής στον οποίο ο χρήστης έχει συνδέσει τον Κύριο Κόμβο του συστήματος μπορεί να εκχωρήσει μία στατική διεύθυνση IP. Σε αντίθετη περίπτωση (π.χ. οικιακή σύνδεση με δυναμική διεύθυνση IP) κρίνεται απαραίτητη η προώθηση θύρας (port forwarding) και η εγγραφή σε υπηρεσία DDNS (Dynamic Domain Name System).

4.5. Ασύρματος πομποδέκτης XBee

4.5.1. Γενικά

Ο ασύρματος πομποδέκτης XBee (μοντέλο XBP24-DMUIT-250) της εταιρίας Digi International Inc. ο οποίος χρησιμοποιήθηκε στην παρούσα εργασία, αποτελεί ένα ενσωματωμένο RF σύστημα, που παρέχει συνδεσιμότητα μεταξύ διαφόρων συσκευών, κάνοντας χρήση ενός ασύρματου πομποδέκτη στα 2.4GHz. Οι συγκεκριμένες μονάδες χρησιμοποιούν το δικτυακό πρωτόκολλο DigiMesh. Το καινοτόμο αυτό peer-to-peer δίκτυο, προσφέρει στους χρήστες αυξημένη σταθερότητα, δυνατότητα αυτό-ίασης σε περίπτωση αποτυχίας ορισμένου αριθμού κόμβων καθώς και κατάσταση αναμονής λειτουργίας (sleep mode) για όλους τους κόμβους, επιμηκύνοντας έτσι τη διάρκεια ζωής των δικτύων που λειτουργούν με μπαταρίες (Digi International Inc., 2011a). Στην εικόνα 18 παρουσιάζεται ο ασύρματος πομποδέκτης Xbee σε τρεις διαφορετικές εκδόσεις.



Εικόνα 18: Ο ασύρματος πομποδέκτης XBee

4.5.2. Προδιαγραφές

Οι προδιαγραφές του ασύρματου πομποδέκτη Xbee φαίνονται στον πίνακα 2.

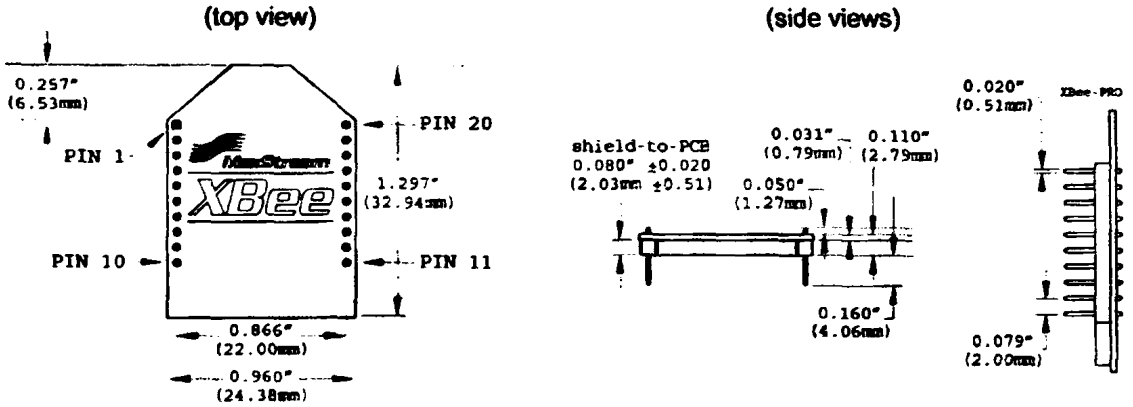
Performance	
Indoor/urban range	90m
Outdoor range (line of sight)	1500m
Transmit Output Power	63mW
RF Data Rate	250kbps
Serial Interface Data Rate	1200bps-250kbps
Receiver Sensitivity	-100dBm (1% packet error rate)
Power Requirements	
Voltage Supply	2.8-3.4 V
Transmit Current (typical)	PL=0 (10dBm): 137mA(@3.3V), 139mA(@3.0V) PL=1 (12dBm): 155mA(@3.3V), 153mA(@3.0V) PL=2 (14dBm): 170mA(@3.3V), 171mA(@3.0V) PL=3 (16dBm): 188mA(@3.3V), 195mA(@3.0V) PL=4 (18dBm): 215mA(@3.3V), 227mA(@3.0V)
Idle/Receive Current (typical)	55mA (@3.3V)
Power-down Current	<50μA
General	
Operating Frequency	ISM 2.4GHz
Dimensions	2.438cm X 3.294cm
Operating Temperature	-40°C to +85°C (Industrial)
Antenna Options	Integrated Whip, Chip or U.FL connector

Networking & Security	
Supported Network Topologies	Point-to-Point, Point-to-Multipoint, Peer-to-Peer
Number of Channels	12 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Addresses

Πίνακας 2

4.5.3. Διαστάσεις

Οι διαστάσεις της μονάδας Xbee φαίνονται στην εικόνα 19.



Εικόνα 19

4.5.4. Οι ακροδέκτες και η λειτουργία τους

Στον πίνακα 3 φαίνονται οι ακροδέκτες της μονάδας Xbee και οι λειτουργίες τους.

Pin#	Όνομα	Διεύθυνση	Περιγραφή
1	VCC	-	Power Supply
2	DOUT	Output	UART Data Out
3	DIN/CONFIG	Input	UART Data In
4	DIO12	Either	Digital I/O 12
5	RESET	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0/RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR/SLEEP_RQ/DI8	Either	Pin Sleep Control Line or Digital I/O 8
10	GND	-	Ground
11	AD4/DIO4	Either	Analog Input 4 or Digital I/O 4
12	CTS/DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON/SLEEP	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate/AD5/DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5

16	RTS/AD6/DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3/DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2/DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1/DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0/DIO0	Either	Analog Input 0 or Digital I/O 0

Πίνακας 3

Οι μονάδες Xbee δεν απαιτούν κάποια συγκεκριμένα ή περίπλοκα εξωτερικά κυκλώματα για τη σωστή λειτουργία τους. Παρ' όλα αυτά, υπάρχουν κάποιες γενικές κατευθυντήριες γραμμές σχεδιασμού που συνιστώνται για την αποφυγή προβλημάτων και τη βέλτιστη λειτουργία των μονάδων.

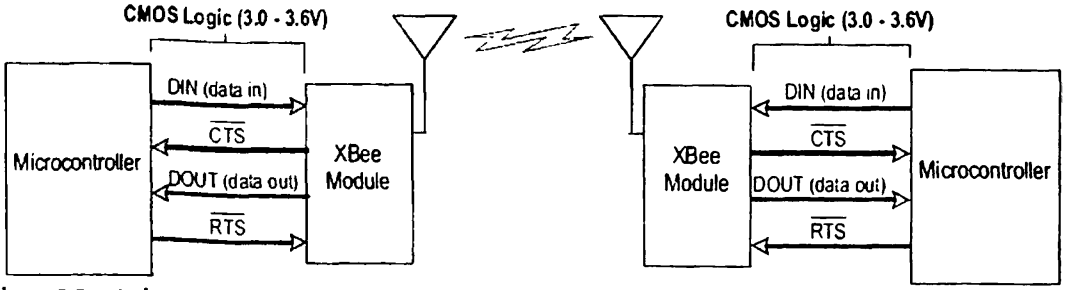
Ο κακός σχεδιασμός στην παροχή ηλεκτρικού ρεύματος μπορεί να οδηγήσει σε φτωχή απόδοση στο τμήμα RF, ειδικά αν η τάση τροφοδοσίας δεν διατηρείται μέσα στα όρια ανοχής ή είναι υπερβολικά θορυβώδης. Για να μειωθεί ο θόρυβος συνιστάται να τοποθετηθούν δύο πυκνωτές, ένας 1μF και ένας 8,2pF όσο το δυνατό πιο κοντά στον ακροδέκτη 1.

Οι μόνοι ακροδέκτες που είναι απολύτως απαραίτητοι για τη λειτουργία μιάς μονάδας είναι οι ακροδέκτες VCC, GND, DIN και DOUT, ενώ για την αναβάθμιση του υλικολογισμικού (firmware) απαιτούνται και οι RTS και DTR. Όλοι οι μη χρησιμοποιούμενοι ακροδέκτες θα πρέπει να παραμείνουν αποσυνδεδεμένοι. Σε όλες τις αχρησιμοποίητες εισόδους της μονάδας μπορεί να εφαρμοστεί η τάση τροφοδοσίας μέσω μίας εσωτερικής αντίστασης pull-up. Αυτό πραγματοποιείται προγραμματιστικά με τη χρήση της εντολής «PR». Οι αχρησιμοποίητες έξοδοι της μονάδας δεν απαιτούν κάποια ειδική μεταχείριση. Όλοι οι υπόλοιποι ακροδέκτες μπορούν να συνδεθούν σε εξωτερικά κυκλώματα για ευκολία στη χρήση της μονάδας, συμπεριλαμβανομένου του LED σύνδεσης (Associate LED, ακροδέκτης 15) και του κουμπιού ανάθεσης (Commissioning button, ακροδέκτης 20). Το LED σύνδεσης αναβοσβήνει ανάλογα με την κατάσταση της μονάδας, ενώ το κουμπί ανάθεσης μπορεί να βοηθήσει στην αντιμετώπιση προβλημάτων χωρίς τη χρήση εντολών μέσω της UART.

Συνδυαστικά, οι δυνατότητες της μονάδας τροφοδοσίας και συλλογής ρεύματος, περιορίζονται στα 120 mA για όλους τους ακροδέκτες. Συγκεκριμένα, οι ακροδέκτες 11 και 15 μπορούν τα τροφοδοτήσουν/συλλέξουν έως και 2 mA, οι ακροδέκτες 9, 6 και 13 έως και 16 mA, ενώ όλοι οι υπόλοιποι έως και 8 mA. Εάν απαιτείται αναλογική δειγματοληψία θα πρέπει στον ακροδέκτη 14 να συνδεθεί μία τάση αναφοράς.

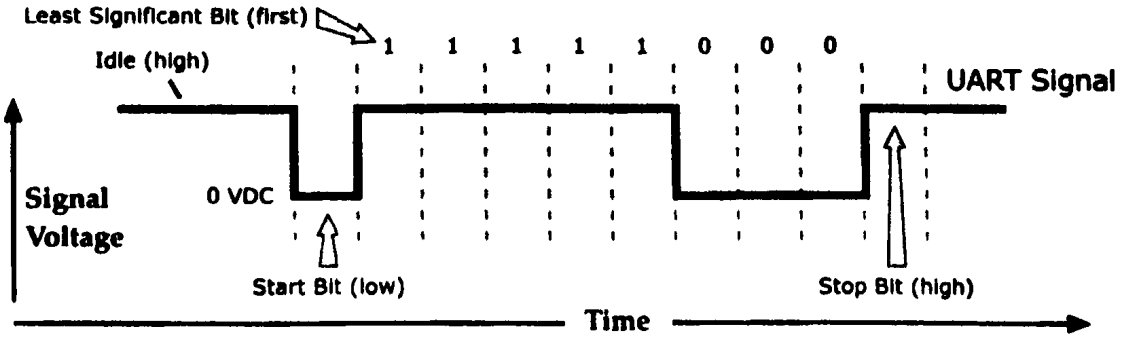
4.5.5. Σειριακή επικοινωνία

Μία μονάδα Xbee μπορεί να συνδεθεί με μία συσκευή μέσω ασύγχρονης σειριακής σύνδεσης. Με τη χρήση της σειριακής θύρας, η μονάδα μπορεί να επικοινωνήσει με οποιαδήποτε UART με συμβατή τάση λειτουργίας, όπως φαίνεται στην εικόνα 20.



Εικόνα 20: Διάγραμμα ροής δεδομένων συστήματος σε περιβάλλον διεπαφής UART.

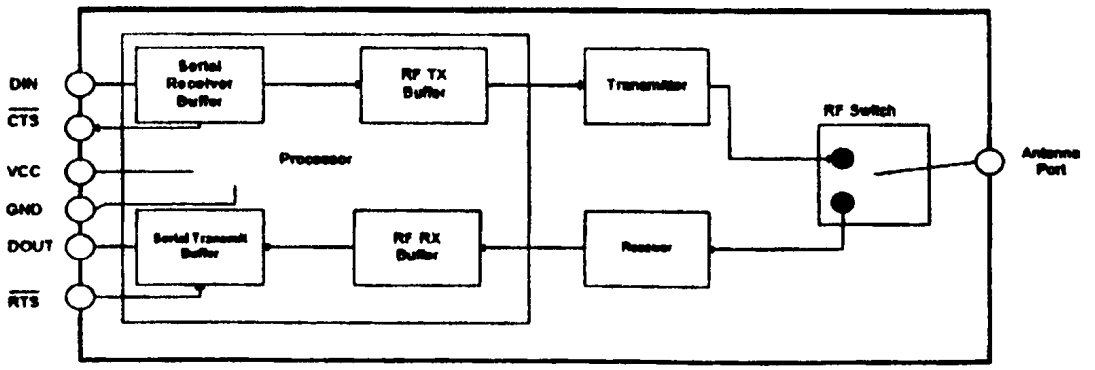
Τα δεδομένα εισέρχονται στη μονάδα μέσω του ακροδέκτη DIN (ακροδέκτης 3) ως ένα ασύγχρονο σειριακό σήμα. Όταν το σήμα είναι αδρανές και δεν υπάρχει καμία μετάδοση θα πρέπει να είναι σε υψηλή στάθμη (λογικό «1»). Κάθε byte δεδομένων αποτελείται από ένα bit εκκίνησης (λογικό «0»), οκτώ bit δεδομένων (το LSB μεταδίδεται πρώτο) και ένα bit τερματισμού. Στην εικόνα 21 μπορούμε να δούμε την ακολουθία των bit που διέρχονται από τη μονάδα.



Εικόνα 21: Πακέτο δεδομένων 0x1F (δεκαδικός αριθμός «31») όπως μεταδίδεται από τη μονάδα σε format 8- N-1 (data bits – parity - #of stop bits)

Ο πομποδέκτης UART της μονάδας επίσης πραγματοποιεί εργασίες ελέγχου χρονισμού και ισοτιμίας, οι οποίες είναι απαραίτητες για τη μετάδοση δεδομένων.

Η μονάδα διατηρεί ενδιάμεσες μνήμες (buffers) για να συλλέγει τα εισερχόμενα σειριακά και RF δεδομένα. Μία σχηματική αναπαράσταση των buffers φαίνεται στην εικόνα 22. Ο σειριακός buffer λήψης συλλέγει το εισερχόμενα δεδομένα και τα διατηρεί μέχρι την επεξεργασία τους. Ο σειριακός buffer εκπομπής συλλέγει τα ληφθέντα δεδομένα από την σύνδεση RF τα οποία θα αποσταλούν από το UART (Εικόνα 22).



Εικόνα 22: Εσωτερικό διάγραμμα ροής δεδομένων

4.5.6. Σειριακά πρωτόκολλα διεπαφής

Οι μονάδες Xbee υποστηρίζουν δύο πρωτόκολλα διεπαφής, το transparent και το API (Application Programming Interface).

4.5.6.1. Πρωτόκολλο Transparent

Όταν έχει επιλεχθεί ο τρόπος λειτουργίας transparent, η μονάδα ενεργεί σαν μία ενσύρματη σειριακή σύνδεση. Όλα τα δεδομένα που λαμβάνονται μέσω του UART από τον ακροδέκτη DIN μπαίνουν στη σειρά για να μεταδοθούν ασύρματα. Αντίθετα, όλα τα δεδομένα που λαμβάνονται ασύρματα, προωθούνται μέσω του UART στον ακροδέκτη DOUT. Η τροποποίηση των παραμέτρων γίνεται χρησιμοποιώντας εντολές «AT» (η μονάδα πρέπει να εισέλθει σε τρόπο λειτουργίας "command mode"). Τα δεδομένα αποθηκεύονται προσωρινά στη σειριακή μνήμη (buffer) λήψης έως ότου συμβεί ένα από τα ακόλουθα γεγονότα που θα προκαλέσει τη δημιουργία ενός πακέτου δεδομένων και αποστολή του:

- Κανένας χαρακτήρας δεν έχει ληφθεί μέσα σε ένα προκαθορισμένο διάστημα. Το διάστημα αυτό τροποποιείται από την εντολή RO (Packetization Timeout). Αν RO=0, τότε η δημιουργία του πακέτου πραγματοποιείται κάθε φορά που λαμβάνεται ένας χαρακτήρας.
- Λήψη ακολουθίας εντολών GT+CC+GT. Θα αποσταλούν όλοι οι χαρακτήρες που ελήφθησαν στη σειριακή μνήμη (buffer) λήψης πριν την ανωτέρω ακολουθία.
- Ο μέγιστος αριθμός χαρακτήρων από τους οποίους απαρτίζεται ένα πακέτο RF έχει ληφθεί.

4.5.6.2. Πρωτόκολλο API

Το πρωτόκολλο API είναι μία εναλλακτική λύση στο πρωτόκολλο Transparent, το οποίο επεκτείνει τις δικτυακές δυνατότητες της μονάδας. Όταν η μονάδα βρίσκεται σε τρόπο λειτουργίας API όλα τα δεδομένα που εισέρχονται ή εξέρχονται εμπεριέχονται σε δομημένα πλαίσια (frames), τα οποία ορίζουν λειτουργίες ή γεγονότα στη μονάδα. Πλαίσια προς αποστολή (ληφθέντα πλαίσια από τον ακροδέκτη DIN) περιλαμβάνουν:

- Πλαίσια αποστολής δεδομένων
- Πλαίσια αποστολής εντολών (ισοδύναμα με command mode)
- Ληφθέντα πλαίσια (πλαίσια προς αποστολή από τον ακροδέκτη DOUT) περιλαμβάνουν:
  - Ληφθέντα πλαίσια δεδομένων
  - Απάντηση σε απεσταλμένη εντολή
  - Ειδοποιήσεις συμβάντων όπως reset, associate, disassociate κλπ

Το πρωτόκολλο API παρέχει επίσης εναλλακτικούς τρόπους παραμετροποίησης της λειτουργίας των μονάδων καθώς και δρομολόγησης των δεδομένων σε επίπεδο εφαρμογής. Αυτό σημαίνει στην πράξη ότι η εφαρμογή μπορεί να στείλει στη μονάδα πλαίσια δεδομένων τα οποία περιέχουν διεύθυνση και μέγεθος πλαισίου αντί να εισέρχεται σε τρόπο λειτουργίας command mode κάθε φορά για να τροποποιεί τη διεύθυνση παραλήπτη.

Αντίθετα η μονάδα θα στείλει πλαίσια δεδομένων στην εφαρμογή τα οποία περιλαμβάνουν πληροφορίες κατάστασης, διεύθυνση αποστολέα και μέγεθος πλαισίου. Παράλληλα το πρωτόκολλο αυτό διευκολύνει πολλές λειτουργίες όπως:

- Αποστολή δεδομένων σε πολλαπλούς προορισμούς χωρίς τη χρήση του command mode.
- Λήψη αναφορών επιτυχών/ανεπιτυχών μεταδόσεων για κάθε RF πακέτο.
- Ταυτοποίηση αποστολέα για κάθε πακέτο που λαμβάνεται.

Από τα παραπάνω συμπεραίνεται ότι το πρωτόκολλο transparent είναι απλό στη χρήση του αλλά με πολλούς περιορισμούς. Από την άλλη πλευρά, το πρωτόκολλο API είναι πιο περίπλοκο, αλλά διευκολύνει πολλές λειτουργίες της μονάδας. Για την παρούσα εργασία, αφού δοκιμάστηκαν και τα δύο πρωτόκολλα στην πράξη, κρίθηκε καταλληλότερο για τις ανάγκες της το πρωτόκολλο API.

4.5.7. Καταστάσεις Λειτουργίας (Modes)

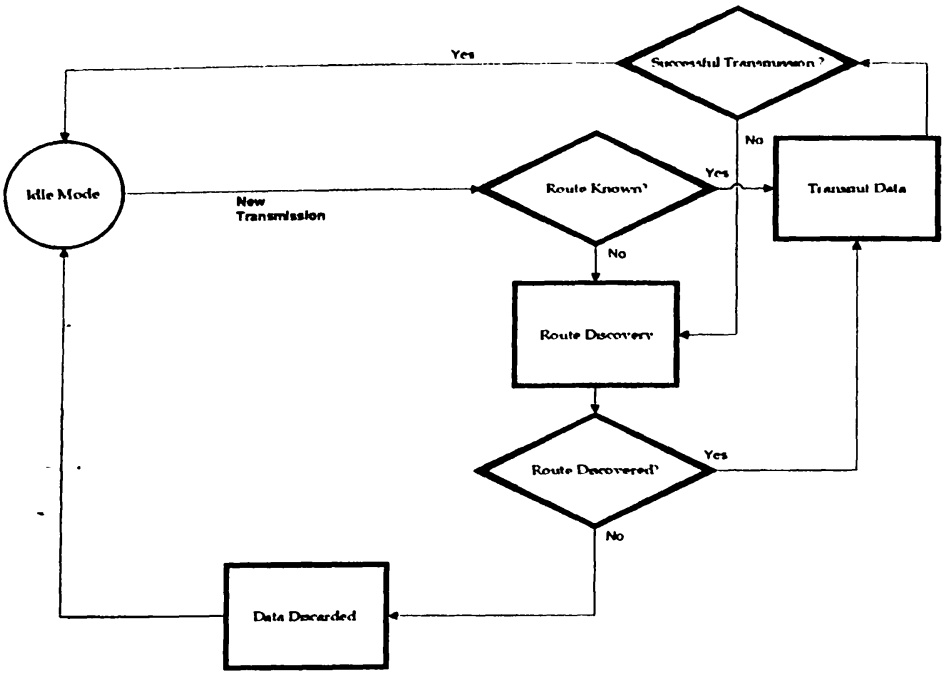
4.5.7.1. Κατάσταση αδράνειας (Idle Mode)

Όταν η μονάδα Xbee δεν λαμβάνει ούτε αποστέλλει δεδομένα για κάποιο χρονικό διάστημα, εισέρχεται σε Idle mode (κατάσταση αδράνειας). Σε αυτή την κατάσταση η μονάδα ελέγχει επίσης για έγκυρα δεδομένα RF. Η μονάδα μπορεί να αλλάξει κατάσταση λειτουργίας στις ακόλουθες καταστάσεις:

- Transmit Mode. Θα μεταβεί σε αυτήν την κατάσταση στην περίπτωση που τα δεδομένα στη σειριακή μνήμη (buffer) λήψης είναι έτοιμα να σχηματίσουν ένα πακέτο.
- Receive Mode. Θα μεταβεί σε αυτήν την κατάσταση στην περίπτωση που έγκυρα δεδομένα RF ληφθούν μέσω της κεραίας της μονάδας.
- Command Mode. Θα μεταβεί σε αυτήν την κατάσταση στην περίπτωση που αποσταλεί στη μονάδα η ακολουθία μετάβασης σε Command Mode (συνήθως «+++»).
- Sleep Mode. Θα μεταβεί σε αυτήν την κατάσταση στην περίπτωση που αποσταλεί η ακολουθία μετάβασης σε Sleep Mode.

4.5.7.2. Κατάσταση αποστολής δεδομένων (Transmit Mode)

Όταν η μονάδα δεχθεί αρκετά δεδομένα ώστε να μπορεί να σχηματίσει πακέτο, εξέρχεται από την κατάσταση Idle Mode και προσπαθεί να στείλει τα δεδομένα. Η διεύθυνση του παραλήπτη καθορίζει ποιος κόμβος ή ποιοι κόμβοι θα δεχτούν τα δεδομένα. Για μία επιτυχή επικοινωνία όλες οι μονάδες θα πρέπει να είναι ρυθμισμένες στο ίδιο κανάλι και ID δικτύου. Εάν η διαδρομή του πακέτου δεν είναι γνωστή, τότε η «ανεύρεση διαδρομής» θα πραγματοποιηθεί αυτόματα από τη μονάδα. Στην περίπτωση που δεν βρεθεί κάποια μονάδα τα δεδομένα απορρίπτονται. Στην εικόνα 23 φαίνεται η ακολουθία μετάδοσης δεδομένων υπό τη μορφή διαγράμματος ροής.



Εικόνα 23: Ακολουθία μετάδοσης δεδομένων

Όταν ένας κόμβος στέλνει δεδομένα σε κάποιον άλλο αποστέλλεται πίσω στον αποστολέα μία αναφορά παράδοσης. Σε περίπτωση που ο αποστολέας δεν λάβει την αναφορά παράδοσης, ξαναστέλνει τα δεδομένα.

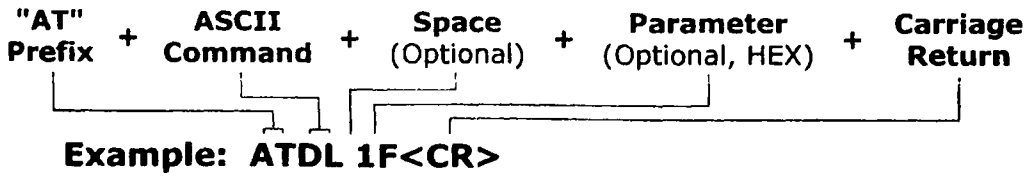
4.5.7.3. Κατάσταση λήψης δεδομένων (Receive Mode)

Όταν η μονάδα Xbee δεχτεί ένα έγκυρο πακέτο RF, τότε τα δεδομένα μεταφέρονται στη σειριακή ενδιάμεση μνήμη (buffer) εκπομπής για περαιτέρω μετάδοση και επεξεργασία.

4.5.7.4. Κατάσταση εντολών (Command Mode)

Για να διαβαστούν ή να τροποποιηθούν οι ρυθμίσεις της μονάδας, αυτό θα πρέπει να εισέλθει στην κατάσταση command mode. Σε αυτή την κατάσταση, όλοι οι εισερχόμενοι χαρακτήρες από τη σειριακή θύρα ερμηνεύονται ως εντολές.

Για να εισέλθει η μονάδα σε command mode ο χρήστης ή η εφαρμογή θα πρέπει να στείλει στον ακροδέκτη DIN την ακολουθία χαρακτήρων «+++». Αν η μονάδα εισέλθει επιτυχώς σε command mode θα στείλει πίσω «OK\r» (\r=carriage return) από τον ακροδέκτη DOUT. Από αυτό το σημείο και έπειτα μπορούμε να στέλνουμε εντολές στη μονάδα. Ένα παράδειγμα σύνταξης μιας εντολής φαίνεται στην εικόνα 24.



Εικόνα 24: Παράδειγμα σύνταξης εντολής

4.5.7.5. Κατάσταση αναμονής (Sleep Mode)

Όταν η μονάδα Xbee μπαίνει σε κατάσταση sleep mode εισέρχεται επίσης σε κατάσταση χαμηλής κατανάλωσης. Όλες οι μονάδες Xbee DigiMesh 2.4 υποστηρίζουν αυτή τη λειτουργία θέτοντας όλο το δίκτυο, συγχρονισμένα, σε κατάσταση sleep mode επιτρέποντας έτσι την εξοικονόμηση ενέργειας σε όλους της κόμβους του δικτύου. Ο πομποδέκτης μπορεί να μεταβεί σε κατάσταση αναμονής με τη χρήση της εντολής «ATSM» η οποία μπορεί να δεχθεί τις εξής παραμέτρους:

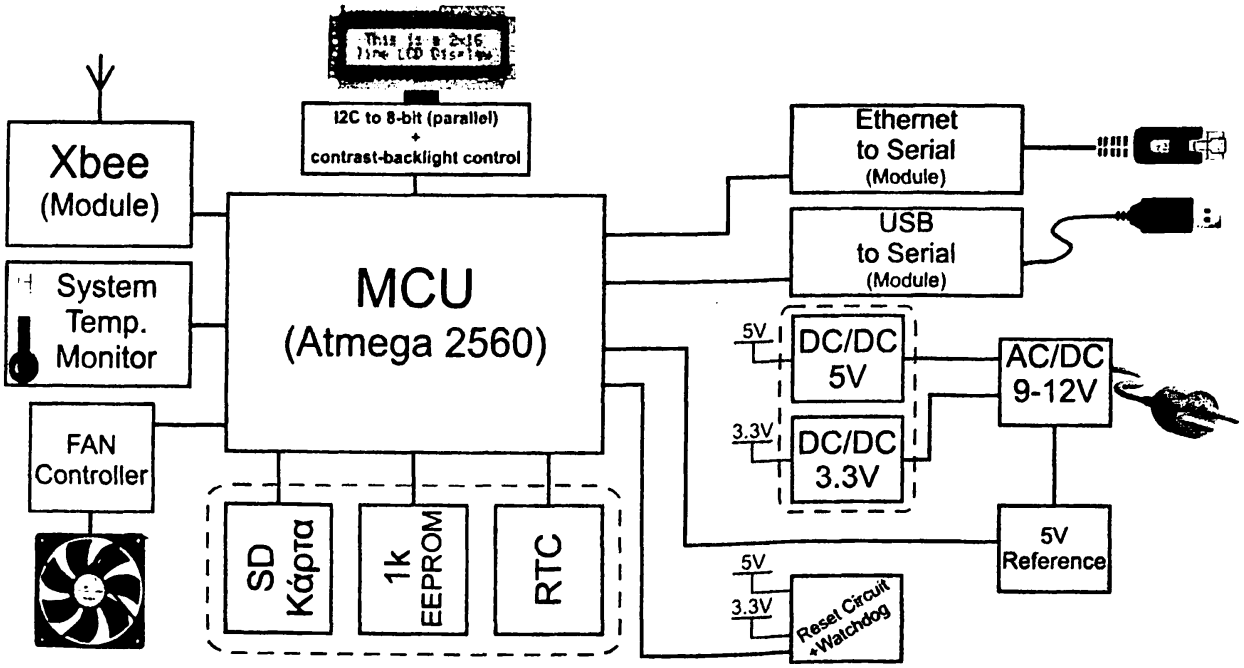
- **SM=0 (Normal Mode):** Σε αυτήν την κατάσταση ο πομποδέκτης δεν μεταβαίνει σε κατάσταση αναμονής.
- **SM=1 (Asynchronous Pin Sleep Mode):** Σε αυτήν την κατάσταση ο πομποδέκτης μπορεί να μεταβεί σε κατάσταση αναμονής ανάλογα με την κατάσταση του ακροδέκτη 9 (Sleep_RQ). Όταν στον ακροδέκτη εφαρμοστεί λογικό «1», τότε ο πομποδέκτης θα τελειώσει ότι διεργασίες αποστολής ή λήψης εκτελεί και ύστερα θα μεταβεί σε κατάσταση αναμονής. Αντίθετα, όταν εφαρμοστεί λογικό «0» στον ακροδέκτη, ο πομποδέκτης μεταβαίνει σε κατάσταση αδράνειας.
- **SM=4 (Cyclic Sleep Mode):** Σε αυτήν την κατάσταση ο πομποδέκτης μεταβαίνει περιοδικά σε κατάσταση αναμονής και αδράνειας για προκαθορισμένους χρόνους. Μετά τη διέλευση του χρόνου αυτού, ο πομποδέκτης μεταβαίνει σε κατάσταση αδράνειας. Ως ένδειξη της κατάστασης του πομποδέκτη, ο ακροδέκτης 13 (On_/Sleep) παράγει ένα λογικό «1» όταν βρίσκεται σε κατάσταση αδρανείας και ένα λογικό «0» όταν βρίσκεται σε κατάσταση αναμονής. Σε αυτήν την κατάσταση ο πομποδέκτης αγνοεί μηνύματα του δικτύου αλλά ανταποκρίνεται σε επικοινωνία μέσω της σειριακής θύρας.
- **SM=5 (Asynchronous Cyclic Sleep with Pin Wake Up Mode):** Πρόκειται για μία παραλλαγή της κατάστασης SM=4. Ο πομποδέκτης μεταβαίνει σε κατάσταση αδράνειας είτε μόνιμα με τη χρήση του ακροδέκτη 9, είτε προσωρινά όταν εξαντληθεί ο προκαθορισμένος χρόνος.
- **SM=7 (Synchronous Sleep Support Mode):** Σε αυτήν την κατάσταση ο πομποδέκτης θα συγχρονιστεί με ένα δίκτυο του οποίου οι κόμβοι είναι σε κατάσταση αναμονής αλλά ο ίδιος δεν θα μεταβεί σε αυτήν την κατάσταση (αναμονής). Η κατάσταση αυτή ενδείκνυται για πομποδέκτες που λειτουργούν ως συντονιστές του δικτύου.
- **SM=8 (Synchronous Cyclic Sleep Mode):** Σε αυτήν την κατάσταση ο πομποδέκτης μεταβαίνει περιοδικά σε κατάσταση αναμονής και αδράνειας για προκαθορισμένους χρόνους. Μετά τη διέλευση του χρόνου αυτού, ο πομποδέκτης μεταβαίνει σε κατάσταση αδράνειας. Ως ένδειξη της κατάστασης του πομποδέκτη, ο ακροδέκτης 13 (On_/Sleep) παράγει ένα λογικό «1» όταν βρίσκεται σε κατάσταση αδρανείας και ένα λογικό «0» όταν βρίσκεται σε κατάσταση αναμονής. Σε αυτήν την κατάσταση ο πομποδέκτης αγνοεί όλα τα μηνύματα, είτε προέρχονται από το δίκτυο, είτε προέρχονται από τη σειριακή θύρα.

5. Κύριος Κόμβος

Ο κύριος κόμβος αποτελεί την «καρδιά» του συστήματος. Μέσα σ' αυτόν γίνεται η επικοινωνία με τον χρήστη, αποθηκεύονται τα δεδομένα από της μετρήσεις και εκτελούνται εργασίες συγχρονισμού και συντήρησης του δικτύου. Τα κύρια στοιχεία της συσκευής είναι:

- Ο μικροελεγκτής,
- Το κύκλωμα με την μνήμη EEPROM, το RTC και την κάρτα SD,
- Ο ασύρματος πομποδέκτης XBee,
- Η μονάδα μετατροπής Ethernet σε σειριακή επικοινωνία,
- Η μονάδα μετατροπής USB σε σειριακή επικοινωνία,
- Το κύκλωμα τροφοδοσίας,
- Το κύκλωμα επανεκκίνησης (Reset),
- Το κύκλωμα οδήγησης της οθόνης,
- Το κύριο κύκλωμα (πλακέτα) που φιλοξενεί όλα τα παραπάνω.

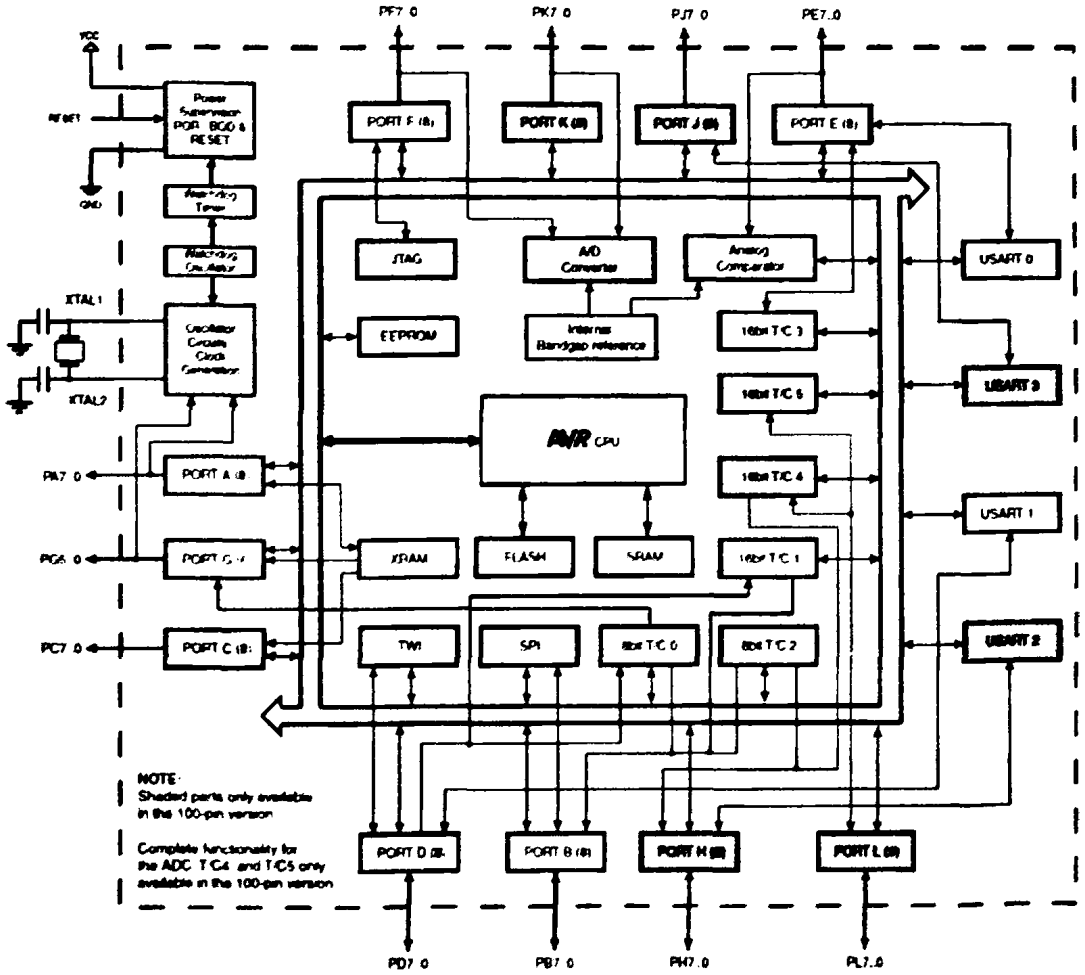
Η σχεδίαση του κύριου κόμβου στηρίχθηκε στην ανάπτυξη ξεχωριστών κυκλωμάτων τα οποία συνδέονται σε έναν κεντρικό, όπως ακριβώς συμβαίνει στους Η/Υ με τη μητρική πλακέτα και τις περιφερειακές μονάδες(RAM, κάρτα γραφικών, κάρτα ήχου, τροφοδοτικό κλπ). Στην εικόνα 25 παρουσιάζεται το λειτουργικό διάγραμμα με τα σημαντικότερα στοιχεία του κύριου κόμβου.



Εικόνα 25: Διάγραμμα του κύριου κόμβου

5.1. Ο μικροελεγκτής ATmega2560

Ο μικροελεγκτής που χρησιμοποιήθηκε στον κύριο κόμβο είναι ο ATmega2560 της εταιρίας Atmel. Πρόκειται για ένα μικροελεγκτή CMOS 8-bit χαμηλής κατανάλωσης, βασισμένο στη βελτιωμένη αρχιτεκτονική AVR RISC. Ένα από τα κορυφαία χαρακτηριστικά του είναι ότι επιτυγχάνει εκτέλεση μιας εντολής σε ένα μόνο κύκλο ρολογιού. Αυτό πρακτικά σημαίνει ότι επιτυγχάνει ταχύτητες που αγγίζουν το 1 MIPS/MHz. Στην εικόνα 26. φαίνεται το διάγραμμα του μικροελεγκτή (Atmel Corporation, 2007).



Εικόνα 26: Διάγραμμα του μικροελεγκτή ATmega2560

Η κεντρική μονάδα επεξεργασίας (CPU) AVR συνδυάζει ένα αρκετά πλούσιο σύνολο εντολών (Instruction Set) με 32 καταχωρητές (registers) γενικής χρήσης. Οι καταχωρητές συνδέονται απευθείας με την αριθμητική λογική μονάδα (ALU), επιτρέποντας την ταυτόχρονη προσπέλαση δύο ανεξάρτητων καταχωρητών με μία εντολή σε έναν κύκλο ρολογιού όπως ήδη προαναφέρθηκε. Αυτό έχει ως αποτέλεσμα η αρχιτεκτονική του AVR να είναι πιο αποτελεσματική και οι ρυθμοαποδόσεις που επιτυγχάνονται να είναι έως και 10 φορές ταχύτερες από εκείνες της συμβατικής αρχιτεκτονικής CISC.

Τα κυριότερα χαρακτηριστικά του ATmega2560 είναι τα εξής:

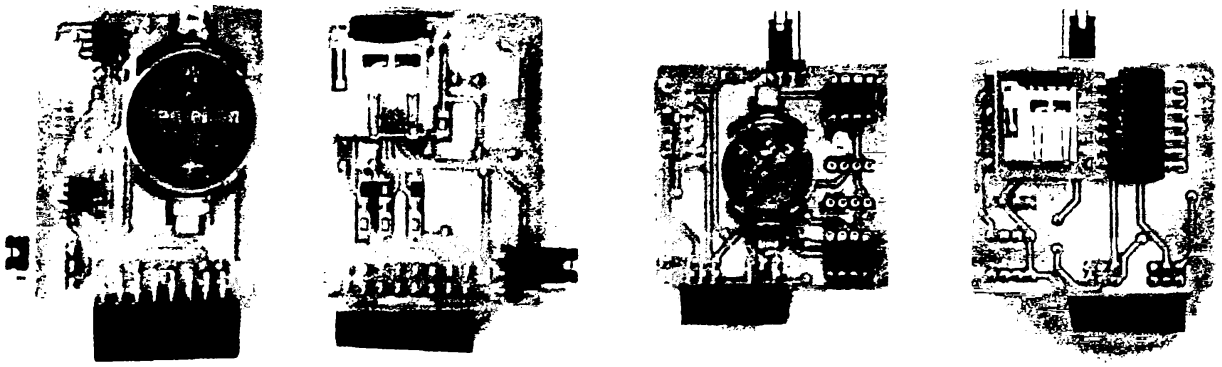
- 256kB μνήμη FLASH με δυνατότητα ταυτόχρονης ανάγνωσης και εγγραφής
- 4kB μνήμη EEPROM
- 8kB μνήμη SRAM
- Έως 16MIPS (@16MHz)
- 86 I/O γενικής χρήσης

- 32 καταχωρητές γενικής χρήσης
- 6 Timers/Counters με δυνατότητα σύγκρισης και PWM
- 4 USARTs
- 2-Wire Serial Interface (TWI ή αλλιώς I^2C)
- SPI serial port (Master ή Slave)
- 10-Bit ADC 16 καναλιών (προαιρετικά με διαφορική είσοδο και προγραμματιζόμενο GAIN)
- IEEE std. 1149.1 compliant JTAG test interface
- 6 Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, και Extended Standby
- Εσωτερικό, προγραμματιζόμενο Watchdog Timer

Για την παρούσα εφαρμογή επιλέχθηκε η διάταξη ολοκληρωμένου κυκλώματος (footprint) TQFP-100. Στον μικροελεγκτή φορτώθηκε το κατάλληλο πρόγραμμα εκκίνησης (bootloader) ώστε να είναι συμβατός με την πλατφόρμα Arduino.

5.2. Το κύκλωμα μνήμης EEPROM, ρολογιού RTC, κάρτας SD

Ένα περιφερειακό κύκλωμα του κύριου κόμβου είναι αυτό που διαθέτει δύο μνήμες EEPROM, το Real Time Clock (RTC) και μία υποδοχή (socket) η οποία φιλοξενεί την κάρτα μνήμης microSD όπου αποθηκεύτηκαν οι μετρήσεις του δικτύου. Στην εικόνα 27 μπορούμε να δούμε την τελική μορφή του κυκλώματος.



Εικόνα 27: Το κύκλωμα με την EEPROM το RTC και την SD σε δύο διαφορετικές εκδόσεις. Χρησιμοποιήθηκε το κύκλωμα που βρίσκεται στο δεξί μέρος της εικόνας

5.2.1. Μνήμες EEPROM

Το ολοκληρωμένο 24LC1025 κατασκευασμένο από την εταιρία Microchip αποτελεί μία σειριακή μνήμη EEPROM (Electronically Erasable Programmable Read-Only Memory) μεγέθους 1024 kbit (128k X 8). Η μνήμη αυτή μπορεί να προγραμματιστεί, να διαγραφεί και να επαναπρογραμματιστεί από το χρήστη. Η διεπαφή της μνήμης με το μικροελεγκτή πραγματοποιείται δια μέσω του διαύλου I^2C. Εκτός των κλασικών ακροδεκτών για το δίαυλο (SDA, SCL) και τους ακροδέκτες τάσης και γείωσης, η συσκευή διαθέτει επίσης έναν ακροδέκτη με το όνομα WP (Write Protect), ο οποίος όταν ενεργοποιηθεί, αποτρέπει την λειτουργία εγγραφής στη μνήμη ενώ αφήνει ανεπηρέαστη τη λειτουργία ανάγνωσης.

Παράλληλα διαθέτει δύο ακόμα ακροδέκτες διευθυνσιοδότησης κάνοντας εφικτή τη λειτουργία έως και τεσσάρων ίδιων μνημών στον ίδιο δίαυλο (2^2=4 διαφορετικοί συνδυασμοί). Στο κύκλωμα υπάρχουν δύο ολοκληρωμένα 24LC1025, το ένα με διεύθυνση 0x50 και το άλλο με διεύθυνση 0x53. (Microchip Corporation, 2011).

5.2.2. Ολοκληρωμένο κύκλωμα RTC (Real Time Clock)

Το ολοκληρωμένο DS1307 της εταιρίας Maxim, είναι ένα χαμηλής κατανάλωσης BCD ρολόι/ημερολόγιο. Η διεύθυνση και τα δεδομένα μεταφέρονται σειριακά μέσω του διαύλου I^2C. Το ολοκληρωμένο παρέχει πληροφορίες για τα δευτερόλεπτα, τα λεπτά, τις ώρες, τις ημέρες, τους μήνες και τη χρονιά. Το τέλος του μήνα προσαρμόζεται αυτόματα για τους μήνες που έχουν λιγότερες από 31 ημέρες. Επίσης υπάρχει αυτόματη διόρθωση για τα δίσεκτα έτη. Το ρολόι λειτουργεί είτε σε 24ωρη λειτουργία είτε σε 12ωρη με δείκτη AM/PM. Το DS1307 έχει ενσωματωμένο κύκλωμα το οποίο ανιχνεύει τις αποτυχίες ηλεκτροδότησης και αυτόματα μεταφέρεται σε παροχή από μπαταρία σε περίπτωση που η κύρια παροχή πέσει κάτω από 1,25 X Vbat. Λειτουργεί ως υποτελής (slave) συσκευή στο σειριακό δίαυλο (Maxim Integrated Products, Inc., 2008).

5.2.3. Κάρτα Μνήμης microSD

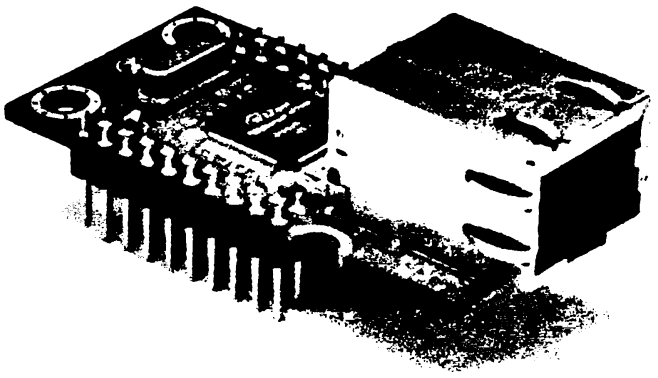
Η κάρτα μνήμης microSD ανήκει στην οικογένεια των καρτών SD που αποτελούν βιομηχανικό πρότυπο. Η συγκεκριμένη κάρτα χρησίμευσε, όπως προαναφέρθηκε, στην αποθήκευση των μετρήσεων του δικτύου. Η διασύνδεση με το μικροελεγκτή έγινε μέσω του πρωτοκόλλου SPI, παρεμβάλλοντας ένα διαιρέτη τάσης καθώς ο μικροελεγκτής δουλεύει στα 5 V ενώ η κάρτα μνήμης στα 3,3 V, ενώ το σύστημα αρχείων που μπορεί να διαχειριστεί το λογισμικό είναι FAT16 και FAT32.

5.3. Ο ασύρματος πομποδέκτης XBee

Η μονάδα, της οποίας η περιγραφή προηγήθηκε στο κεφάλαιο 4.5.

5.4. Μονάδα "Ethernet to Serial"

Για τη διασύνδεση του μικροελεγκτή με το τοπικό δίκτυο χρησιμοποιήθηκε μονάδα Ethernet-to-Serial, το WIZ812MJ της εταιρίας Wiznet (Wiznet, 2009). Η καρδιά της μονάδας είναι το ολοκληρωμένο W5100 της ίδιας εταιρίας (Wiznet, 2011). Μέσα στο ολοκληρωμένο, υλοποιούνται τα πρωτόκολλα TCP/IP, MAC και Physical (PHY). Η διασύνδεση με το μικροελεγκτή έγινε μέσω του πρωτοκόλλου SPI.

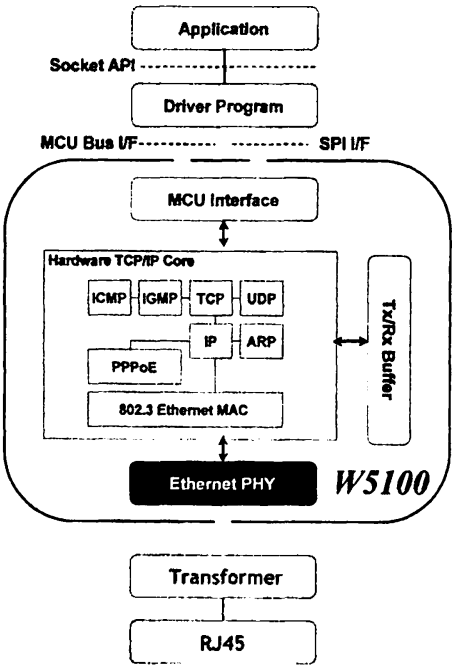


Εικόνα 28: Μονάδα WIZ812MJ

Στα χαρακτήριστικά της μονάδας συγκαταλέγονται τα εξής:

- Υποστήριξη 10/100 Base-T
- Υποστήριξη λειτουργίας ταυτόχρονης αποστολής και λήψης δεδομένων (half/full duplex)
- Υποστήριξη αυτόματης διαπραγμάτευσης με το δίκτυο
- Αυτόματη ανίχνευση σύνδεσης καλωδίου Cross-Over
- Συμβατότητα με IEEE 802.3 και IEEE 802.3u
- Υλοποίηση επίσης των πρωτοκόλλων UDP, ICMP, ARP, PPPoE, IGMP και DLC.
- Υποστήριξη έως και 4 συνδέσεων ταυτόχρονα.

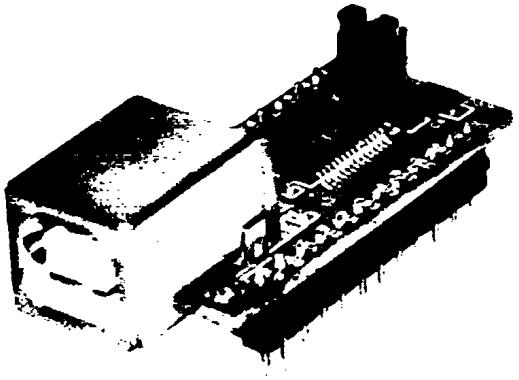
Στην εικόνα 29 παρουσιάζεται το απλοποιημένο σχηματικό διάγραμμα της μονάδας.



Εικόνα 29: Απλοποιημένο διάγραμμα της μονάδας WIZ812MJ

5.5. Μονάδα "USB to Serial"

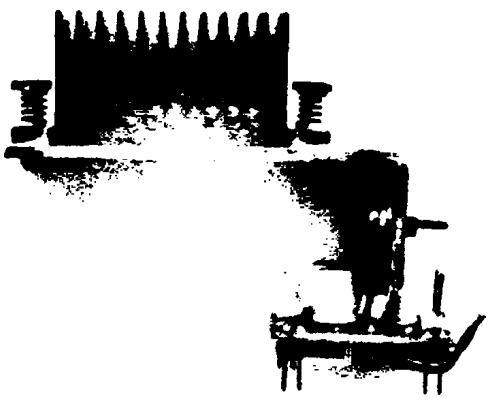
Για τον προγραμματισμό του μικροελεγκτή, καθώς για την επικοινωνία του με έναν Η/Υ θα πρέπει να χρησιμοποιηθεί η σειριακή του θύρα (συγκεκριμένα η UART0). Καθώς οι περισσότεροι Η/Υ σήμερα δεν διαθέτουν σειριακή θύρα, η λύση στο πρόβλημα δόθηκε με τη χρήση ενός ενδιάμεσου κυκλώματος που κάνει μετατροπή των σημάτων από USB σε Serial και ανάποδα από Serial σε USB. Το ενδιάμεσο αυτό κύκλωμα, ονομάζεται UM232R (FTDI, 2011a) της εταιρίας FTDI, το οποίο ενσωματώνει το ολοκληρωμένο FT232R (FTDI, 2011b) της ίδιας εταιρίας. Αντίθετα με προηγούμενες εκδόσεις, το FT232R, απλοποιεί τη σχεδίαση ενός συστήματος με δυνατότητα επικοινωνίας USB, ενσωματώνοντας τη μνήμη EEPROM, τον ταλαντωτή καθώς και τις αντιστάσεις ανάμεσα στο συνδετήρα USB και το ολοκληρωμένο. Το συγκεκριμένο ολοκληρωμένο, εισάγει επίσης την τεχνολογία FTDIChip-ID™, όπου ένας μοναδικός αριθμός είναι προεγγεγραμμένος στη μνήμη του κατά τη διάρκεια της κατασκευής του, ο οποίος μπορεί να διαβαστεί κατευθείαν μέσω του διαύλου USB, θέτοντας τη βάση για τη δημιουργία μιας δικλείδας ασφαλείας για το συνοδευτικό λογισμικό της συσκευής που θα φέρει το Chip. Στην εικόνα 30 μπορούμε να δούμε τη μονάδα.



Εικόνα 30: Ενδιάμεσο κύκλωμα UM232R

5.6. Κύκλωμα τροφοδοσίας

Το σύστημα για να λειτουργήσει χρειάζεται δύο τάσεις τροφοδοσίας, μια των 5V και μία των 3,3V. Το κύκλωμα της τροφοδοσίας είναι σχετικά απλό. Τροφοδοτείται από έναν εξωτερικό μετασχηματιστή τοίχου με μία τάση που μπορεί να κυμαίνεται από 9V έως 12V εντάσεως από 500mA έως 1A. Το κύκλωμα παίρνει αυτή την τροφοδοσία και μέσα από μία γέφυρα διόδων (για προστασία αντίστροφης πολικότητας) (Vishay, 2011) και μία επαναφερόμενη ασφάλεια PPTC (1A) για προστασία από βραχυκυκλώματα (Bourns, 2010), τη διοχετεύει σε δύο ρυθμιστές τάσης, στον KA78T05 της εταιρίας Fairchild Semiconductor που παρέχει τα 5V (Fairchild Semiconductor, 2002), και στον LD1117V33 της εταιρίας ST Microelectronics που παρέχει τα 3,3V (ST Microelectronics, 2005). Το κύκλωμα επίσης παρέχει δύο βύσματα όπου μπορούν να συνδεθούν δύο LEDs για την ένδειξη λειτουργίας των ρυθμιστών τάσης. Στην εικόνα 31 φαίνεται το κύκλωμα τροφοδοσίας του συστήματος μαζί με τη ψήκτρα.



Εικόνα 31:Κύκλωμα τροφοδοσίας

5.7. Κύκλωμα Επανεκκίνησης (Reset)

Το κύκλωμα επανεκκίνησης εξυπηρετεί στη σωστή επανεκκίνηση του συστήματος σε διάφορες περιπτώσεις:

- Πτώση οποιασδήποτε από τις δύο τάσεις τροφοδοσίας που έχει ως αποτέλεσμα το «κρέμασμα» του μικροελεγκτή,
- Χειροκίνητη επανεκκίνηση από το χρήστη,
- «Κόλλημα» του προγράμματος που τρέχει ο μικροελεγκτής,
- Απομακρυσμένη επανεκκίνηση με τη χρήση λογισμικού
- Επανεκκίνηση Power-On Reset (POR)

Για την επανεκκίνηση σε περίπτωση πτώσης κάποιας από τις δύο τάσεις, Power-On Reset ή της χειροκίνητης επανεκκίνησης από το χρήστη, χρησιμοποιήθηκε το ολοκληρωμένο DS1834 της εταιρίας Maxim (Maxim Integrated Products, Inc., 2000). Το ολοκληρωμένο είναι ένας επιτηρητής τάσης (Voltage Supervisor) που παρακολουθεί τις τάσεις τροφοδοσίας 5 V και 3,3 V και στην περίπτωση που οι τάσεις πέσουν πάνω από 10%, δηλαδή κάτω από τα 4,5 V και 2,97 V αντίστοιχα, τότε ο ακροδέκτης reset# αλλάζει την έξοδό του από λογικό «1» σε λογικό «0». Όταν οι τάσεις επανέλθουν στα αποδεκτά επίπεδα, το ολοκληρωμένο θα κρατήσει την έξοδο του σε λογικό «0» για 350 ms ακόμα πριν την αλλάξει σε λογικό «1». Με τον ίδιο τρόπο επιτυγχάνεται και το Power-On Reset. Για τη χειροκίνητη επανεκκίνηση, ένας διακόπτης (push button) συνδέεται στο ολοκληρωμένο και τη γείωση. Όταν ο χρήστης τον πιέσει, το ολοκληρωμένο εξαλείφει τυχόν αιχμές τάσης μέσω ενός εσωτερικού κυκλώματος αποφυγής θορύβου (debounce) και στη συνέχεια ο ακροδέκτης reset# αλλάζει την έξοδό του από λογικό «1» σε λογικό «0» για 350 ms.

Για την επανεκκίνηση του συστήματος από «κόλλημα» του προγράμματος, χρησιμοποιήθηκε ένα Watchdog Timer, το MAX6371 της εταιρίας Maxim (Maxim Integrated Products, Inc., 2011). Ένας ακροδέκτης I/O του μικροελεγκτή συνδέθηκε στον ακροδέκτη WDI (WatchDog Input) του MAX6371. Το πρόγραμμα που «τρέχει» ο μικροελεγκτής θα πρέπει να αλλάζει τη λογική κατάσταση του ακροδέκτη που συνδέεται με το MAX6371 τακτικά, ώστε να μηδενίζει ο μετρητής του. Σε περίπτωση που ο μετρητής «γεμίσει» θα προκαλέσει υπερχείληση (overflow) προκαλώντας έτσι επανεκκίνηση στο σύστημα. Η έξοδος του Watchdog Timer μπορεί να απενεργοποιηθεί αφαιρώντας το βραχυκυκλωτήρα (jumper) που βρίσκεται στο πίσω μέρος του κυκλώματος.

Η απομακρυσμένη επανεκκίνηση με τη χρήση λογισμικού δεν εμπίπτει σε καμία από τις παραπάνω περιπτώσεις, ούτε απαντάται τακτικά στη βιβλιογραφία. Πρόκειται για μία εντολή επανεκκίνησης που δίνει ο χρήστης απομακρυσμένα με τη χρήση λογισμικού και απαιτείται από το μικροελεγκτή να την πραγματοποιήσει. Η πιο απλή υλοποίηση θα ήταν να συνδεθεί ένα I/O του μικροελεγκτή στο reset και να μεταβεί σε λογικό «0». Αυτό δυστυχώς δεν είναι εφικτό καθώς με το που θα γίνει η μετάβαση όλοι οι ακροδέκτες θα μεταβούν σε κατάσταση high-Z, συμπεριλαμβανομένου του ακροδέκτη που έδωσε την εντολή. Για να ολοκληρωθεί μία επανεκκίνηση θα πρέπει ο ακροδέκτης που δίνει την εντολή και κατά συνέπεια και ο ακροδέκτης reset να παραμείνει σε λογικό «0» για κάποιο διάστημα, πράγμα που δεν είναι δυνατό, καθώς είναι ήδη σε high-Z κατάσταση. Συνέπεια όλων αυτών είναι να «κρεμάσει» ο μικροελεγκτής. Για το λόγο αυτό σχεδιάστηκε ένα κύκλωμα με τη χρήση ενός χρονιστή 556 σε μονοσταθή λειτουργία (δύο χρονιστές 555 συνδεδεμένοι σε σειρά) της εταιρίας Texas Instruments (Texas Instruments, 1997). Ο σκανδαλισμός (trigger) του πρώτου συνδέεται σε ένα I/O του μικροελεγκτή και η έξοδός του στο trigger του δεύτερου. Η έξοδος του δεύτερου χρονιστή 555 καταλήγει στο reset μέσω μίας πύλης NOT (Texas Instruments, 2011). Στην εικόνα 32 φαίνεται το κύκλωμα επανεκκίνησης.

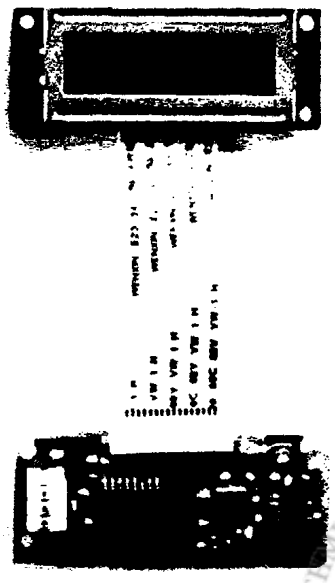
Εικόνα 32: Κύκλωμα επανεκκίνησης.

5.8. Το κύκλωμα οδήγησης της οθόνης

Το κύκλωμα οδήγησης της οθόνης, μαζί με την οθόνη, εξυπηρετεί στην προβολή κάποιων παραμέτρων λειτουργίας. Συγκεκριμένα προβάλλει εναλλάσσοντας κάθε 5 δευτερόλεπτα τις εξής παραμέτρους:

- Διεύθυνση IP με την οποία είναι συνδεδεμένο το σύστημα στο τοπικό δίκτυο.
- Τρέχουσα ημερομηνία και ώρα.
- Μέτρηση των δύο τάσεων λειτουργίας του κύριου κόμβου (μέσω του ενσωματωμένου ADC του μικροελεγκτή).
- Θερμοκρασία κύριου κόμβου.
- Παρουσία ή όχι της κάρτας μνήμης microSD.
- Χρονικό διάστημα μεταξύ δύο διαδοχικών μετρήσεων.

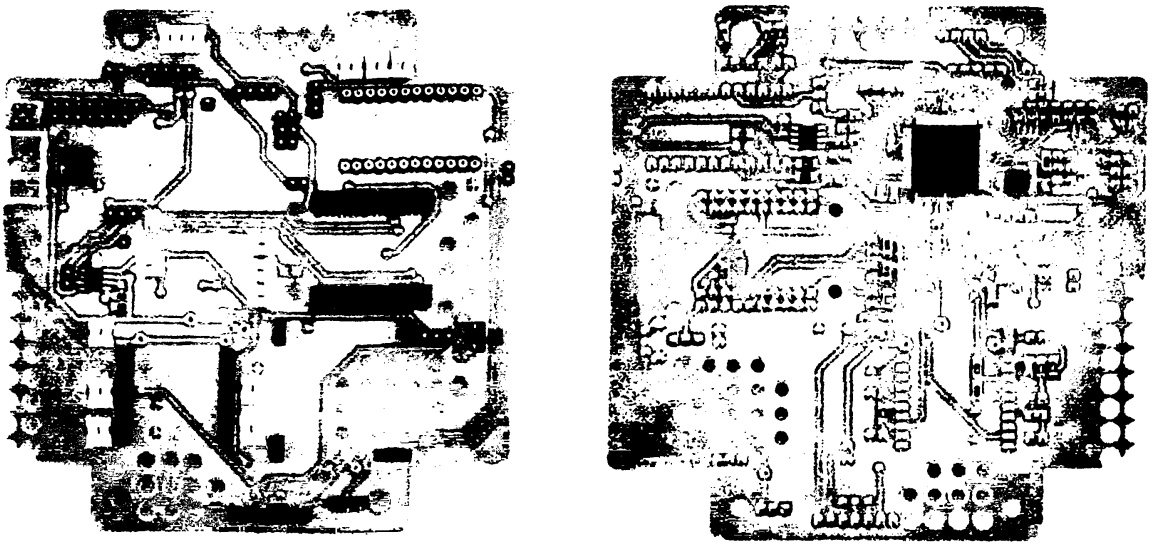
Το κύκλωμα συνδέεται με το υπόλοιπο σύστημα μέσω του διαύλου I^2C. Αποτελείται από το ολοκληρωμένο PCF8574 της εταιρίας Texas Instruments το οποίο μετατρέπει τα σήματα του διαύλου I^2C σε μία παράλληλη θύρα εύρους 8 bit για τη διασύνδεση των σημάτων της οθόνης (Texas Instruments, 2001), το ολοκληρωμένο OPA2350 της εταιρίας Texas Instruments (Texas Instruments, 2005) το οποίο παρέχει την κατάλληλη ενίσχυση ρεύματος για το φωτισμό της οθόνης και το ολοκληρωμένο MAX5392 της εταιρίας Maxim (Maxim Integrated Products, Inc., 2010) το οποίο είναι ένα ψηφιακό ποτενσιόμετρο. Ελέγχεται μέσω του ίδιου διαύλου για τη ρύθμιση της φωτεινότητας και της αντίθεσης της οθόνης. Τέλος, στο κύκλωμα συνδέεται μία οθόνη υγρών κρυστάλλων 2 γραμμών και εύρους 16 χαρακτήρων η κάθε μία (2X16) της εταιρίας Hantronix (Hantronix, 2010). Στην εικόνα 33 παρουσιάζεται η οθόνη η οποία είναι συνδεδεμένη στο κύκλωμα οδήγησής της.



Εικόνα 33: Οθόνη και κύκλωμα οδήγησης

5.9. Το κύριο κύκλωμα

Το κύριο κύκλωμα είναι το εφάμιλλο μίας μητρικής κάρτας σε έναν Η/Υ. Είναι το κύκλωμα στο οποίο συνδέονται όλα τα δευτερεύοντα κυκλώματα που περιγράφηκαν στα προηγούμενα υποκεφάλαια και επίσης φιλοξενεί τον μικροελεγκτή. Στην εικόνα 34 φαίνεται τελειωμένο το κύριο κύκλωμα.



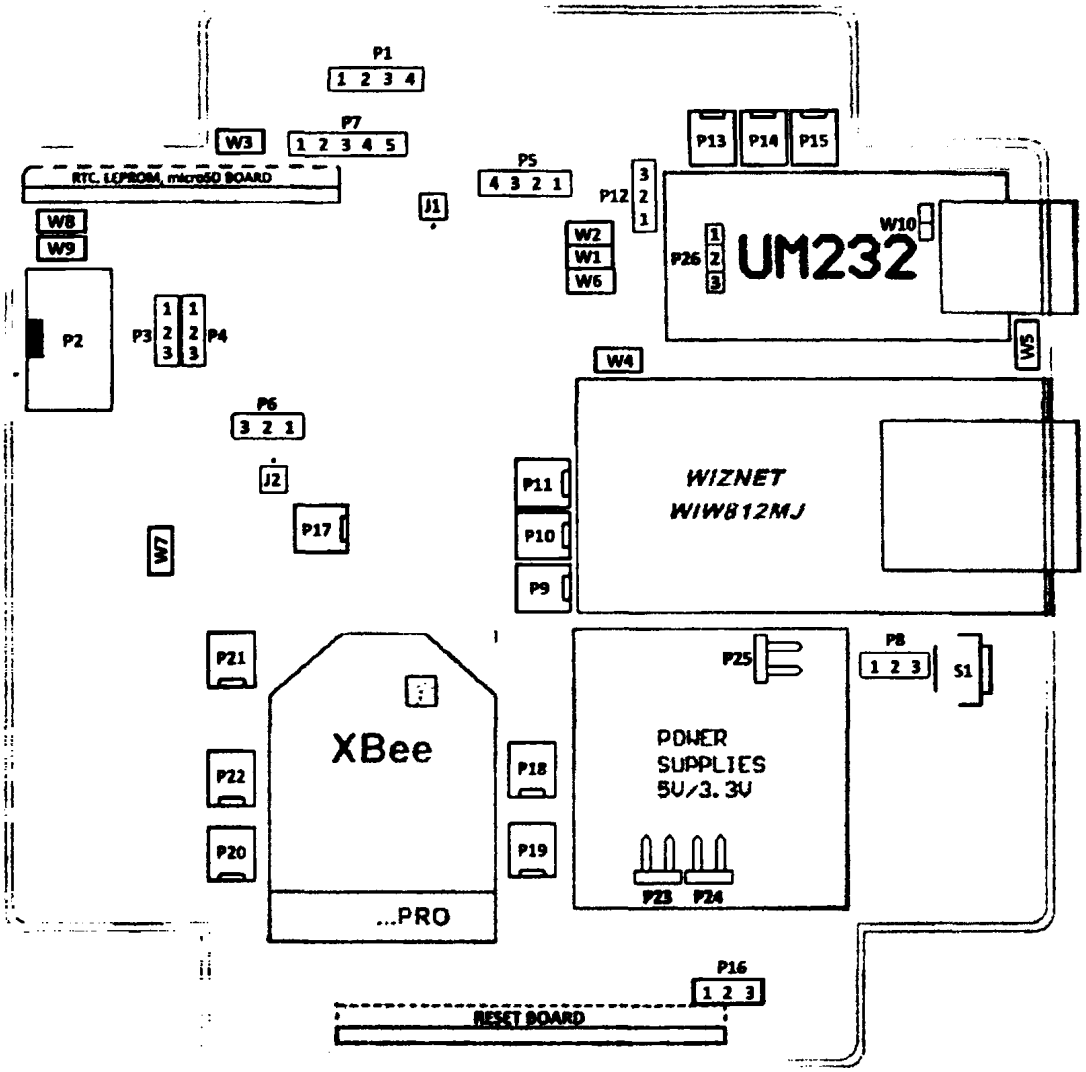
Εικόνα 34: Κύριο κύκλωμα (Top layer αριστερά, Bottom Layer δεξιά)

Αξίζει να αναφερθεί πως στο κύριο κύκλωμα έχει ενσωματωθεί το ολοκληρωμένο MAX6665 της εταιρίας Maxim (Maxim Integrated Products, Inc., 2005), το οποίο είναι ένας ελεγκτής ανεμιστήρα και ένας ανεμιστήρας εξαερισμού στο κουτί μεγέθους, 2cm X 2cm για προστασία από υπερθέρμανση των κυκλωμάτων. Ο εξαερισμός λειτουργεί όταν ξεπεραστεί ένα προκαθορισμένο κατώφλι (45°C) ή κατ' απαίτηση του χρήστη. Το ολοκληρωμένο διαθέτει επίσης τη δυνατότητα να ενημερώνει την εφαρμογή αν ο ανεμιστήρας λειτουργεί και αν έχει ξεπεραστεί το κατώφλι.

Παράλληλα, έχει ενσωματωθεί το ολοκληρωμένο MAX6350 (Maxim Integrated Products, Inc., 2001) το οποίο παράγει με μεγάλη ακρίβεια μία τάση αναφοράς 5 V για τους μετατροπείς ADC του μικροελεγκτή. Σε δύο από αυτούς έχουν συνδεθεί οι τάσεις τροφοδοσίας 5 V και 3.3 V ώστε η εφαρμογή να μπορεί να τις παρακολουθεί ως διαγνωστικό έλεγχο.

5.10. Συναρμολόγηση και ρυθμίσεις λειτουργίας Κύριου Κόμβου

Όπως προαναφέρθηκε, ο Κύριος Κόμβος αποτελείται από το κύριο κύκλωμα και δευτερεύοντα κυκλώματα που συνδέονται πάνω σ' αυτό. Στην εικόνα 35 φαίνεται ένα τοπογραφικό διάγραμμα του κύριου κυκλώματος με τις θέσεις και τον προσανατολισμό των δευτερευόντων κυκλωμάτων (ανοιχτό γκρι χρώμα). Το διάγραμμα επίσης περιέχει και τις θέσεις διαφόρων βυσμάτων των οποίων η λειτουργία θα εξηγηθεί παρακάτω.



Εικόνα 35: Τοπογραφικό διάγραμμα κύριου κυκλώματος

- **P1:** Βύσμα με το πρωτόκολλο επικοινωνίας I2C. Ο ακροδέκτης «1» παρέχει τάση 5V, ο «2» την έξοδο ρολογιού από τον «master» του διαύλου (SCL), ο «3» την είσοδο/έξοδο δεδομένων (SDA) και ο «4» γείωση.
- **P2:** Θύρα προγραμματισμού ISP (In-System Programming). Σε αυτό το βύσμα καταλήγουν τα σήματα του πρωτοκόλλου SPI, το RESET, η τάση και η γείωση. Με τη σύνδεση ενός εξωτερικού προγραμματιστή περνάει το πρόγραμμα στη μνήμη flash του μικροελεγκτή. Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε μόνο μία φορά για την εγγραφή του προγράμματος εκκίνησης του Arduino. Στη συνέχεια, το πρόγραμμα πέρασε στη μνήμη flash χωρίς τη χρήση κάποιας εξωτερικής συσκευής, μέσω της μονάδας USB-to-Serial.
- **P3:** Βύσμα με το πρωτόκολλο επικοινωνίας 1-Wire. Ο ακροδέκτης «1» παρέχει τάση 5V, ο «2» είσοδο/έξοδο δεδομένων και ο «3» γείωση. Σε αυτό το βύσμα

τοποθετήθηκε ο αισθητήρας θερμοκρασίας DS18B20 της εταιρίας Maxim για την παρακολούθηση της θερμοκρασίας συστήματος.

- **P4:** Βύσμα, όπως και το P3, με το πρωτόκολλο επικοινωνίας 1-Wire.
- **P5:** Θύρα προγραμματισμού JTAG (Joint Test Action Group). Σε αυτό το βύσμα καταλήγουν τα σήματα του πρωτοκόλλου JTAG, το TDI (Test Data In, ακροδέκτης «4»), το TDO (Test Data Out, ακροδέκτης «3»), το TMS (Test Mode Select, ακροδέκτης «2») και το TCK (Test Clock, ακροδέκτης «1»). Η θύρα αυτή χρησιμεύει επίσης στην αποσφαλμάτωση.
- **P6:** Βύσμα με τα σήματα PB5 (ακροδέκτης «1»), PB6 (ακροδέκτης «2»), και PB7 (ακροδέκτης «3»). Τα σήματα αυτά είναι χρήσιμα σε περίπτωση που χρειαστεί στο μέλλον να προγραμματιστεί ο μικροελεγκτής με τη χρήση παράλληλης συνδεσμολογίας.
- **P7:** Βύσμα με τα σήματα PD2 (ακροδέκτης «1»), PD3 (ακροδέκτης «2»), PD4 (ακροδέκτης «3»), PD5 (ακροδέκτης «4») και PD6 (ακροδέκτης «5»). Τα σήματα αυτά είναι χρήσιμα σε περίπτωση που χρειαστεί στο μέλλον να προγραμματιστεί ο μικροελεγκτής με τη χρήση παράλληλης συνδεσμολογίας.
- **P8:** Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη /RESET του WIZ812MJ (Ethernet-to-Serial) είτε με το κύκλωμα RESET είτε με τον ακροδέκτη PE3 του μικροελεγκτή. Αν ο βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «1» και «2» τότε το /RESET του WIZ812MJ συνδέεται με τον ακροδέκτη PE3 του μικροελεγκτή, ενώ αν τοποθετηθεί στους «2» και «3» συνδέεται με το κύκλωμα RESET.
- **P9:** Ethernet LINK LED, βύσμα στο οποίο συνδέθηκε ένα μπλε LED του κουτιού. Ανάβει όταν υπάρχει σύνδεση Ethernet με ένα Switch ή ένα Router.
- **P10:** Ethernet TX LED, βύσμα στο οποίο συνδέθηκε ένα πράσινο LED του κουτιού. Ανάβει όταν ο μικροελεγκτής αποστέλλει δεδομένα μέσω της θύρας Ethernet.
- **P11:** Ethernet RX LED, βύσμα στο οποίο συνδέθηκε ένα κόκκινο LED του κουτιού. Ανάβει όταν ο μικροελεγκτής λαμβάνει δεδομένα μέσω της θύρας Ethernet.
- **P12:** Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη /RESET του UM232R (USB-to-Serial) είτε με την τάση 5V, είτε με κύκλωμα RESET ώστε να επανεκκινείται κάθε φορά που επανεκκινείται το σύστημα. Αν ο βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «1» και «2» τότε το /RESET του UM232R συνδέεται με την τάση, ενώ αν τοποθετηθεί στους «2» και «3» συνδέεται με το κύκλωμα RESET.
- **P13:** USB TX LED, βύσμα στο οποίο συνδέθηκε ένα πράσινο LED του κουτιού. Ανάβει όταν ο μικροελεγκτής αποστέλλει δεδομένα μέσω της θύρας USB.
- **P14:** USB RX LED, βύσμα στο οποίο συνδέθηκε ένα κόκκινο LED του κουτιού. Ανάβει όταν ο μικροελεγκτής λαμβάνει δεδομένα μέσω της θύρας USB.
- **P15:** USB LED, βύσμα στο οποίο συνδέθηκε ένα μπλε LED του κουτιού. Ανάβει όταν υπάρχει USB σύνδεση με έναν Η/Υ.
- **P16:** Βραχυκυκλωτήρας που συνδέει το κύκλωμα Auto-Reset από το UM232R είτε με την έξοδο του κυκλώματος RESET (ακροδέκτες «1» και «2»), είτε με την είσοδο του ίδιου κυκλώματος (ακροδέκτες «2» και «3»). Κατά τον προγραμματισμό του μικροελεγκτή συνιστάται να τοποθετηθεί στους ακροδέκτες «1» και «2».
- **P17:** Fan Output, Βύσμα στο οποίο συνδέθηκε ο ανεμιστήρας εξαερισμού του κουτιού.
- **P18:** κουμπί Commission Xbee, βύσμα στο οποίο συνδέθηκε το κουμπί Commission. Το κουμπί αυτό παρέχει κάποιες λειτουργίες που βοηθούν στην παραμετροποίηση του δικτύου. Οι λειτουργίες αυτές φαίνονται στον πίνακα 4.

Πατήματα κουμπιού	Ρύθμιση κατάστασης αναμονής	Λειτουργία
1	Χωρίς ρύθμιση για αναμονή λειτουργίας	Στέλνει την εντολή Node Identification (NI) σε όλο το δίκτυο. Κάθε μονάδα που θα λάβει την εντολή θα ανάβει και θα σβήνει το Associate LED γρήγορα για 1 δευτερόλεπτο.
1	Με ρύθμιση για αναμονή λειτουργίας	«Ξυπνάει» τη μονάδα για 30 δευτερόλεπτα. Στον επόμενο κύκλο λειτουργίας του δικτύου (αφού βγει από την κατάσταση αναμονής) κάθε μονάδα θα πρέπει να στείλει Node Identification ενώ παράλληλα θα ανάβει και θα σβήνει το Associate LED γρήγορα για 1 δευτερόλεπτο.
2	Χωρίς ρύθμιση για αναμονή λειτουργίας	Δεν υποστηρίζεται.
2	Με ρύθμιση για αναμονή λειτουργίας	Αναγκάζει τη μονάδα να λειτουργήσει σαν Network Sleep Coordinator.
4	Με ή χωρίς ρύθμιση για αναμονή λειτουργίας	Εκδίδει την εντολή ATRE για επαναφορά των προεπιλεγμένων ρυθμίσεων.

Πίνακας 4

Το πάτημα του κουμπιού μπορεί να αντικατασταθεί με τη χρήση της εντολής «ATCB"X"» (όπου "X" μπαίνει ο αριθμός των πατημάτων του κουμπιού).

- **P19:** Xbee Associate LED, βύσμα στο οποίο συνδέθηκε το πράσινο LED που βρίσκεται ενσωματωμένο στο κουμπί «Commission». Το LED αυτό μπορεί να παρέχει πληροφορίες για την κατάσταση αναμονής λειτουργίας (sleep mode) της μονάδας, καθώς και διαγνωστικές λειτουργίες. Για να λειτουργήσει το LED σαν Associate LED θα πρέπει να τεθεί η εντολή D5 σε «1». Στον πίνακα 5 φαίνεται η συμπεριφορά του LED στις διάφορες περιπτώσεις.

Sleep Mode	Κατάσταση LED	Επεξήγηση
Όχι	Ανάβει και σβήνει	Η μονάδα λειτουργεί κανονικά.
Ναι	Μόνιμα αναμμένο	Η μονάδα δεν έχει συγχρονιστεί ή έχει χάσει το συγχρονισμό του με το δίκτυο.
Ναι	Ανάβει και σβήνει αργά (500ms αναμμένο-500ms σβηστό)	Η μονάδα λειτουργεί σαν συντονιστής κατάστασης αναμονής του δικτύου (network sleep coordinator) και λειτουργεί κανονικά.
Ναι	Ανάβει και σβήνει γρήγορα (250ms αναμμένο-250ms σβηστό)	Η μονάδα είναι σωστά συγχρονισμένο με το δίκτυο.

Πίνακας 5

- **P20:** Xbee RSSI (Received Signal Strength Indicator) LED, βύσμα στο οποίο συνδέθηκε ένα μπλε LED του κουτιού. Επειδή η έξοδος RSSI της μονάδας Xbee είναι PWM, η ένταση που ανάβει το LED είναι ανάλογη με τη ισχύ του ληφθέντος σήματος.
- **P21:** Xbee TX LED, βύσμα στο οποίο συνδέθηκε ένα πράσινο LED του κουτιού. Ανάβει όταν ο μικροελεγκτής αποστέλλει δεδομένα μέσω της μονάδας Xbee.
- **P22:** Xbee RX LED, βύσμα στο οποίο συνδέθηκε ένα κόκκινο LED του κουτιού. Ανάβει όταν ο μικροελεγκτής λαμβάνει δεδομένα μέσω της μονάδας Xbee.
- **P23:** 5V OK, βύσμα στο οποίο συνδέθηκε ένα κόκκινο LED του κουτιού. Ανάβει όταν υπάρχει η τάση των 5V.
- **P24:** 3,3V OK, βύσμα στο οποίο συνδέθηκε ένα κόκκινο LED του κουτιού. Ανάβει όταν υπάρχει η τάση των 3,3V.
- **P25:** Vin, βύσμα πάνω στο δευτερεύον κύκλωμα τροφοδοσίας στο οποίο συνδέθηκε το εξωτερικό βύσμα τροφοδοσίας του κουτιού. Η προτεινόμενη τροφοδοσία που θα πρέπει να συνδεθεί είναι τάσεως από 9V έως 12V και εντάσεως από 500mA έως 1A. Η πολικότητα δεν παίζει καμία σημασία, καθώς το κύκλωμα τροφοδοσίας είναι εφοδιασμένο με μία γέφυρα διόδων.
- **P26:** Βραχυκυκλωτήρας πάνω στη μονάδα USB-to-Serial. Χρησιμεύει στην επιλογή τάσης στα I/O της μονάδας. Εάν ο βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «1» και «2» τότε τα I/O της μονάδας θα χρησιμοποιούν 5 V. Εάν τοποθετηθεί στους «2» και «3» τότε θα χρησιμοποιούν 3,3 V. Για τη συνδεσμολογία που χρησιμοποιείται στον κύριο κόμβο θα πρέπει να τοποθετηθεί στους ακροδέκτες «1» και «2».
- **J1:** Βύσμα με το σήμα AD0. Το σήμα αυτό είναι χρήσιμο σε περίπτωση που χρειαστεί στο μέλλον να προγραμματιστεί ο μικροελεγκτής με τη χρήση παράλληλης συνδεσμολογίας.
- **J2:** Βύσμα με το σήμα PB0. Το σήμα αυτό είναι χρήσιμο σε περίπτωση που χρειαστεί στο μέλλον να προγραμματιστεί ο μικροελεγκτής με τη χρήση παράλληλης συνδεσμολογίας.
- **W1:** Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί συνδέει την τάση τροφοδοσίας 3.3 V με τον εσωτερικό ADC1 του μικροελεγκτή για διαγνωστικό έλεγχο.
- **W2:** Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί συνδέει την τάση τροφοδοσίας 5V με τον εσωτερικό ADC3 του μικροελεγκτή για διαγνωστικό έλεγχο.
- **W3:** Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί, συνδέει δύο 10kΩ pull-up αντιστάσεις στις γραμμές SDA και SCL του διαύλου I2C.
- **W4:** Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί συνδέει το Interrupt Output του Ethernet-to-Serial με το Interrupt 4 του μικροελεγκτή. Ενεργοποιείται παράγοντας ένα λογικό «0» μετά από κάποια αποστολή ή λήψη κάποιου πακέτου δεδομένων μέσω της Ethernet ζητώντας την προσοχή του μικροελεγκτή. Προς το παρόν δεν χρησιμοποιείται.
- **W5:** Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί ενεργοποιεί την αυτόματη επανεκκίνηση του μικροελεγκτή κατά τον προγραμματισμό του. Σε περίπτωση που δεν τοποθετηθεί απαιτείται η χειροκίνητη επανεκκίνηση για την ολοκλήρωση του προγραμματισμού.
- **W6:** Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί συνδέει τον ακροδέκτη AREF του μικροελεγκτή την τάση αναφοράς που παράγεται από το ολοκληρωμένο MAX6350.
- **W7:** Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί συνδέει τον ακροδέκτη HYST του Fan Controller (MAX6665) με τον ακροδέκτη «24» του μικροελεγκτή. Για να μην ξεκινά και σταματά στιγμιαία ο ανεμιστήρας εξαερισμού του κουτιού από

μικρές μεταβολές της θερμοκρασίας, μπορεί να οριστεί μια καθυστέρηση σε oC πάνω από το Threshold. Αν εφαρμόσουμε στον ακροδέκτη λογικό «0» η καθυστέρηση ορίζεται στους 4oC ενώ αν εφαρμόσουμε λογικό «1» η καθυστέρηση ορίζεται στους 8oC. Στην περίπτωση που δεν τοποθετηθεί ο βραχυκυκλωτήρας ο ακροδέκτης HYST μεταβαίνει σε high-Z κατάσταση και η καθυστέρηση ορίζεται αυτόματα στον 1oC.

- **W8:** Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί συνδέει τάση 5V στο κύκλωμα με την EEPROM το RTC και τη microSD κάρτα.
- **W9:** Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί συνδέει τάση 3,3V στο κύκλωμα με την EEPROM το RTC και τη microSD κάρτα.
- **W10:** Βραχυκυκλωτήρας πάνω στο USB-to-Serial ο οποίος αν τοποθετηθεί παρέχει τροφοδοσία από το δίαυλο USB στο δευτερεύον κύκλωμα. Για τη συνδεσμολογία που χρησιμοποιείται στον κύριο κόμβο θα πρέπει να τοποθετηθεί.

Στον πίνακα 6 παρατίθενται οι ακροδέκτες του μικροελεγκτή ATmega2560 με τα ονόματά τους και τη συνδεσμολογία τους. Ο πίνακας αυτός είναι ιδιαίτερα χρήσιμος κατά τη *συγγραφή ή τροποποίηση κώδικα*.

Α/Α	Όνομα	Περιγραφή
1	PG5(OC0B)	Χωρίς σύνδεση.
2	PE0(**RXD0**/PCINT8)	Ακροδέκτης λήψης δεδομένων της σειριακής θύρας (Hardware Serial0). Συνδέεται με τον ακροδέκτη αποστολής δεδομένων (TX) του UM232R
3	PE1(**TXD0**)	Ακροδέκτης αποστολής δεδομένων της σειριακής θύρας (Hardware Serial0). Συνδέεται με τον ακροδέκτη λήψης δεδομένων (RX) του UM232R
4	PE2(XCK0/AIN0)	Χωρίς σύνδεση.
5	**PE3**(OC3A/AIN1)	Ακροδέκτης για την επανεκκίνηση του WIZ812MJ. (Σε συνδυασμό με το βραχυκυκλωτήρα P8).
6	**PE4**(OC3B/INT4)	Ακροδέκτης για τη μετάβαση της μονάδας Xbee σε κατάσταση αναμονής. Όταν ο μικροελεγκτής θέσει τον ακροδέκτη σε λογικό «1», η μονάδα λειτουργεί σε sleep mode. Αντίθετα, σε λογικό «0» η μονάδα λειτουργεί κανονικά.
7	**PE5**(OC3C/INT5)	Ακροδέκτης ένδειξης λειτουργίας της μονάδας Xbee. Συνδέεται με τον ακροδέκτη ON/SLEEP της μονάδας Xbee. Όταν διαβάζει ο μικροελεγκτής λογικό «1» η μονάδα λειτουργεί κανονικά. Όταν διαβάζει λογικό «0» η μονάδα είναι σε sleep mode.
8	PE6(T3/INT6)	Χωρίς σύνδεση.
9	PE7(CLKO/ICP3/INT7)	Χωρίς σύνδεση.
10	**VCC**	Τάση τροφοδοσίας 5V.
11	**GND**	Γείωση.
12	PH0(**RXD2**)	Ακροδέκτης λήψης δεδομένων της σειριακής

		θύρας (Hardware Serial2). Συνδέεται με τον ακροδέκτη αποστολής δεδομένων (TX) της μονάδας Xbee.
13	PH1(**TXD2**)	Ακροδέκτης αποστολής δεδομένων της σειριακής θύρας (Hardware Serial0). Συνδέεται με τον ακροδέκτη λήψης δεδομένων (RX) της μονάδας Xbee
14	PH2(XCK2)	Χωρίς σύνδεση.
15	PH3(OC4A)	Χωρίς σύνδεση.
16	**PH4**(OC4B)	Ο ακροδέκτης αυτός συνδέεται με την είσοδο του WatchDog Timer που βρίσκεται στο κύκλωμα Reset (WDI).
17	**PH5**(OC4C)	Αυτός ο ακροδέκτης συνδέεται με τον ακροδέκτη /WARN του Fan Controller (MAX6665). Είναι ένδειξη υπερθέρμανσης. Παράγει ένα λογικό «0» η θερμοκρασία που μετράει το ολοκληρωμένο είναι 15°C πάνω από το Threshold. Σε κανονική λειτουργία παράγει λογικό «1».
18	**PH6**(OC2B)	Αυτός ο ακροδέκτης συνδέεται με τον ακροδέκτη /OT του Fan Controller (MAX6665). Είναι ένδειξη υπερθέρμανσης. Παράγει ένα λογικό «0» η θερμοκρασία που μετράει το ολοκληρωμένο είναι 30°C πάνω από το Threshold. Σε κανονική λειτουργία παράγει λογικό «1».
19	PB0((/SS)/PCINT0)	Χωρίς σύνδεση.
20	PB1(**SCK**/PCINT1)	Ακροδέκτης SCK (Serial Clock) του διαύλου SPI.
21	PB2(**MOSI**/PCINT2)	Ακροδέκτης MOSI (Master Out-Slave In) του διαύλου SPI.
22	PB3(**MISO**/PCINT3)	Ακροδέκτης MISO (Master In-Slave Out) του διαύλου SPI.
23	**PB4**(OC2A/PCINT4)	Ακροδέκτης SS (Slave Select) του διαύλου SPI για τη μονάδα Ethernet-to-Serial (WIZ812MJ)
24	**PB5**(OC1A/PCINT5)	Αυτός ο ακροδέκτης συνδέεται με τον ακροδέκτη /FORCEON του Fan Controller (MAX6665). Αν εφαρμόσουμε στον ακροδέκτη λογικό «0» ο ανεμιστήρας εξαερισμού ξεκινά ανεξαρτήτως θερμοκρασίας. Αντίθετα αν εφαρμόσουμε λογικό «1» ο ανεμιστήρας επιστρέφει στην κανονική του λειτουργία. Συνδέεται επίσης με το βύσμα P6 (ακροδέκτης «1»)
25	**PB6**(OC1B/PCINT6)	Αυτός ο ακροδέκτης συνδέεται με τον ακροδέκτη HYST του Fan Controller (MAX6665). Αν εφαρμόσουμε στον ακροδέκτη λογικό «0» η καθυστέρηση ορίζεται στους 4°C ενώ αν εφαρμόσουμε λογικό «1» η καθυστέρηση ορίζεται στους 8°C. Στην περίπτωση που δεν συνδέσουμε τον ακροδέκτη η καθυστέρηση ορίζεται αυτόματα στον 1°C. Συνδέεται επίσης με το βύσμα P6 (ακροδέκτης «2»).

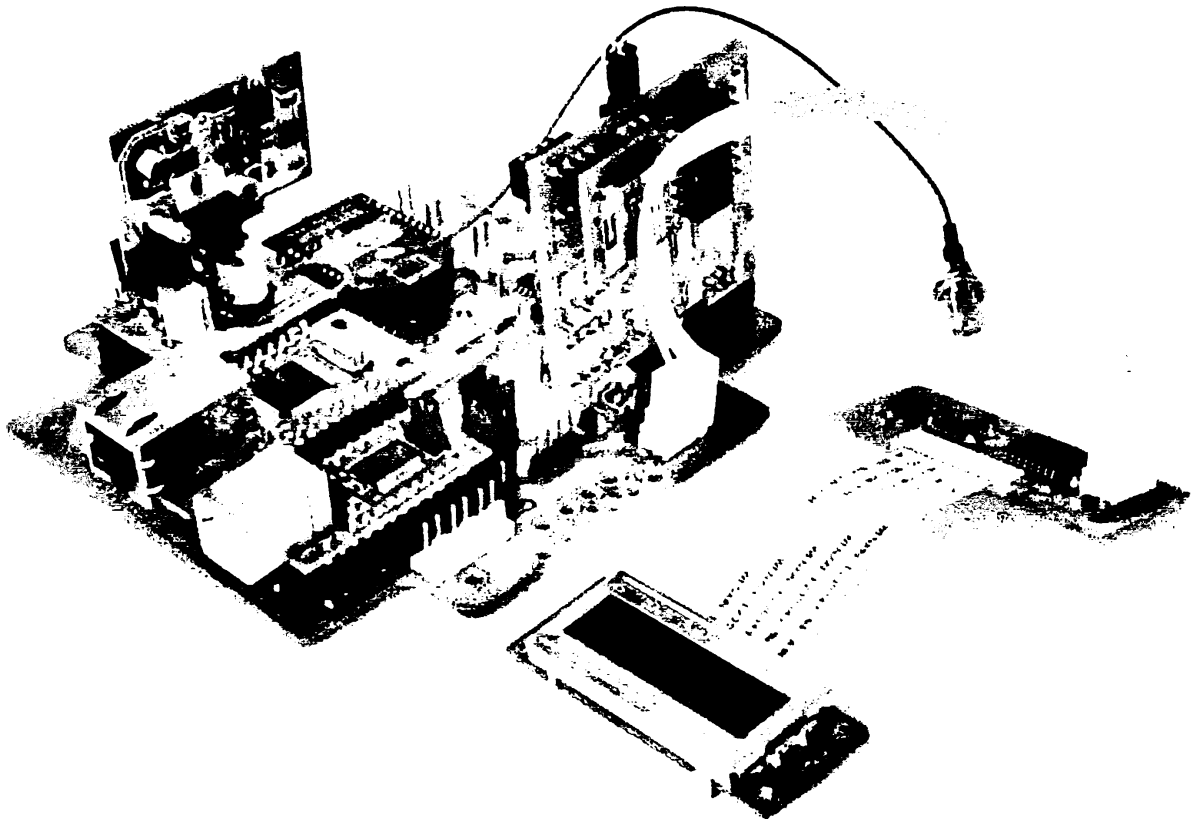
26	**PB7**(OC0A/OC1C/PCINT7)	Αυτός ο ακροδέκτης συνδέεται με τον ακροδέκτη FANON του Fan Controller (MAX6665). Είναι ένδειξη λειτουργίας του ανεμιστήρα εξαερισμού. Αν παράγει ένα λογικό «1» ο ανεμιστήρας λειτουργεί. Αν παράγει λογικό «0» ο ανεμιστήρας δεν λειτουργεί. Συνδέεται με το βύσμα P6 (ακροδέκτης «3»).
27	PH7(T4)	Χωρίς σύνδεση.
28	PG3(TOSC2)	Χωρίς σύνδεση.
29	PG4(TOSC1)	Χωρίς σύνδεση.
30	**/RESET**	Reset του μικροελεγκτή. Μπορεί να ενεργοποιηθεί από το UM232R κατά τον προγραμματισμό (αν είναι τοποθετημένος ο βραχυκυκλωτήρας W5), ή από το κύκλωμα Reset.
31	**VCC**	Τάση τροφοδοσίας 5V.
32	**GND**	Γείωση.
33	**XTAL2**	Σύνδεση με ταλαντωτή 16MHz.
34	**XTAL1**	Σύνδεση με ταλαντωτή 16MHz.
35	**PL0**(ICP4)	Στον ακροδέκτη αυτό θα πρέπει να υλοποιηθεί προγραμματιστικά το πρωτόκολλο 1-wire. Έχει συνδεθεί επάνω του μια pull-up αντίσταση 4,7kΩ για υποστήριξη παρασιτικής τροφοδοσίας.
36	**PL1**(ICP5)	Ακροδέκτης που καταλήγει στο κύκλωμα με την EEPROM, το RTC και την κάρτα μνήμης microSD. Write Protect της EEPROM με διεύθυνση 0x50.
37	PL2(T5)	Χωρίς σύνδεση.
38	PL3(OC5A)	Χωρίς σύνδεση.
39	PL4(OC5B)	Χωρίς σύνδεση.
40	PL5(OC5C)	Χωρίς σύνδεση.
41	**PL6**	Ακροδέκτης ανίχνευσης εισαγωγής κάρτας μνήμης microSD. Συνδέεται με το microSD socket το οποίο παράγει ένα λογικό «1» όταν τοποθετηθεί μια κάρτα microSD ή ένα λογικό «0» όταν το socket είναι άδειο.
42	**PL7**	Ακροδέκτης SS (Slave Select) του διαύλου SPI για την κάρτα μνήμης microSD.
43	PD0(**SCL**/INT0)	Ακροδέκτης SCL του διαύλου I^2C. Έχει συνδεθεί επάνω του μια pull-up αντίσταση 10kΩ.
44	PD1(**SDA**/INT1)	Ακροδέκτης SDA του διαύλου I^2C. Έχει συνδεθεί επάνω του μια pull-up αντίσταση 10kΩ.
45	**PD2**(RXD1/**INT2**)	Interrupt του μικροελεγκτή (Input) που συνδέεται μέσω του βραχυκυκλωτήρα W4 στο Interrupt (active low) της μονάδας Ethernet-to-Serial (WIZ812MJ). Συνδέεται και με το βύσμα P7 (ακροδέκτης «1»)
46	**PD3**(TXD1/INT3)	Χωρίς σύνδεση με το κύκλωμα. Συνδέεται με το

		βύσμα P7 (ακροδέκτης «2»)
47	**PD4**(ICP1)	Χωρίς σύνδεση με το κύκλωμα. Συνδέεται με το βύσμα P7 (ακροδέκτης «3»)
48	**PD5**(XCK1)	Χωρίς σύνδεση με το κύκλωμα. Συνδέεται με το βύσμα P7 (ακροδέκτης «4»)
49	**PD6**(T1)	Χωρίς σύνδεση με το κύκλωμα. Συνδέεται με το βύσμα P7 (ακροδέκτης «5»)
50	**PD7**(T0)	Ακροδέκτης που συνδέεται με τον ακροδέκτη SIG του κυκλώματος RESET. Χρησιμεύει στο να γίνει triggered ο πρώτος χρονιστής 555 του κυκλώματος RESET. Ο ακροδέκτης θα πρέπει να οριστεί σαν έξοδος και να παράγει μόνιμα λογικό «1». Τη στιγμή που θα πρέπει να γίνει επανεκκίνηση ο ακροδέκτης από λογικό «1» θα πρέπει να παράγει ένα λογικό «0» για τουλάχιστον 500ms.
51	PG0(/WR)	Χωρίς σύνδεση.
52	PG1(/RD)	Χωρίς σύνδεση.
53	PC0(A8)	Χωρίς σύνδεση.
54	PC1(A9)	Χωρίς σύνδεση.
55	PC2(A10)	Χωρίς σύνδεση.
56	PC3(A11)	Χωρίς σύνδεση.
57	PC4(A12)	Χωρίς σύνδεση.
58	PC5(A13)	Χωρίς σύνδεση.
59	PC6(A14)	Χωρίς σύνδεση.
60	PC7(A15)	Χωρίς σύνδεση.
61	**VCC**	Τάση τροφοδοσίας 5V.
62	**GND**	Γείωση.
63	PJ0(RXD3/PCINT9)	Χωρίς σύνδεση.
64	PJ1(TXD3/PCINT10)	Χωρίς σύνδεση.
65	PJ2(XCK3/PCINT11)	Χωρίς σύνδεση.
66	PJ3(PCINT12)	Χωρίς σύνδεση.
67	PJ4(PCINT13)	Χωρίς σύνδεση.
68	PJ5(PCINT14)	Χωρίς σύνδεση.
69	PJ6(PCINT15)	Χωρίς σύνδεση.
70	PG2(ALE)	Χωρίς σύνδεση.
71	PA7(AD7)	Χωρίς σύνδεση.
72	PA6(AD6)	Χωρίς σύνδεση.
73	PA5(AD5)	Χωρίς σύνδεση.
74	PA4(AD4)	Χωρίς σύνδεση.
75	PA3(AD3)	Χωρίς σύνδεση.

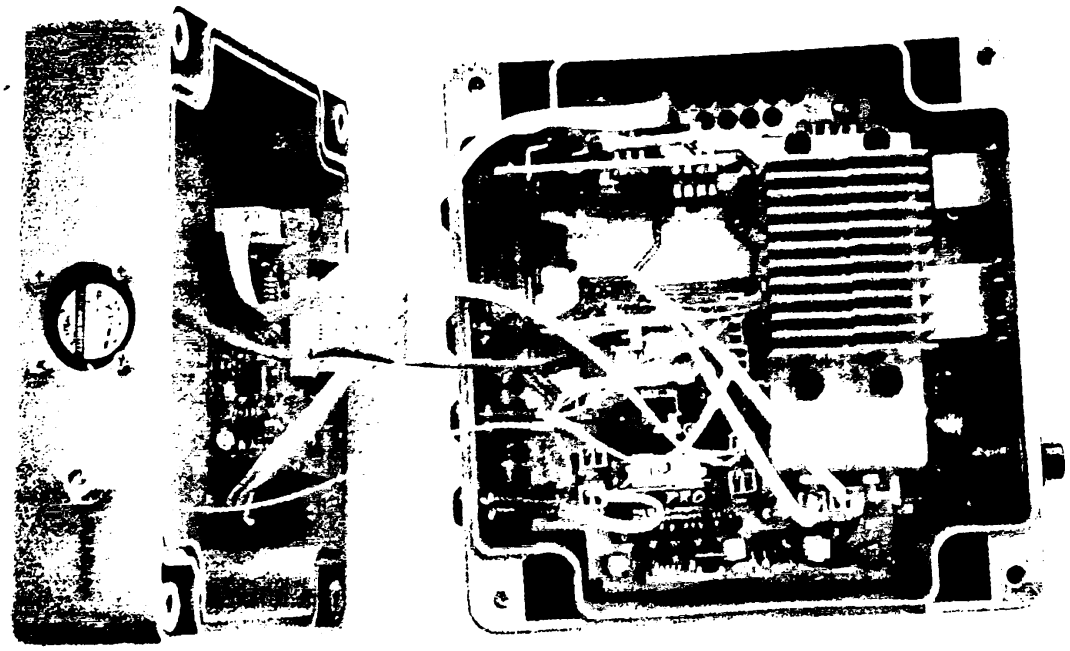
76	PA2(AD2)	Χωρίς σύνδεση.
77	PA1(AD1)	Χωρίς σύνδεση.
78	**PA0**(AD0)	Χωρίς σύνδεση με το κύκλωμα. Συνδέεται με το βύσμα J1
79	PJ7	Χωρίς σύνδεση.
80	**VCC**	Τάση τροφοδοσίας 5V.
81	**GND**	Γείωση.
82	PK7(ADC15/PCINT23)	Χωρίς σύνδεση.
83	PK6(ADC14/PCINT22)	Χωρίς σύνδεση.
84	PK5(ADC13/PCINT21)	Χωρίς σύνδεση.
85	PK4(ADC12/PCINT20)	Χωρίς σύνδεση.
86	PK3(ADC11/PCINT19)	Χωρίς σύνδεση.
87	PK2(ADC10/PCINT18)	Χωρίς σύνδεση.
88	PK1(ADC9/PCINT17)	Χωρίς σύνδεση.
89	PK0(ADC8/PCINT16)	Χωρίς σύνδεση.
90	PF7(ADC7/**TDI**)	Χωρίς σύνδεση με το κύκλωμα. Συνδέεται με το βύσμα P5 (ακροδέκτης «4»)
91	PF6(ADC6/**TDO**)	Χωρίς σύνδεση με το κύκλωμα. Συνδέεται με το βύσμα P5 (ακροδέκτης «3»)
92	PF5(ADC5/**TMS**)	Χωρίς σύνδεση με το κύκλωμα. Συνδέεται με το βύσμα P5 (ακροδέκτης «2»)
93	PF4(ADC4/**TCK**)	Χωρίς σύνδεση με το κύκλωμα. Συνδέεται με το βύσμα P5 (ακροδέκτης «1»)
94	PF3(**ADC3**)	Ακροδέκτης ο οποίος συνδέεται με την τάση τροφοδοσίας 5V για διαγνωστικό έλεγχο
95	PF2(ADC2)	Χωρίς σύνδεση.
96	PF1(**ADC1**)	Ακροδέκτης ο οποίος συνδέεται με την τάση τροφοδοσίας 3.3V για διαγνωστικό έλεγχο
97	PF0(ADC0)	Χωρίς σύνδεση.
98	AREF	Τάση αναφοράς για τον εσωτερικό 10-bit ADC. Συνδέεται στη γείωση μέσω ενός πυκνωτή 100nF σε σειρά ή με την τάση αναφοράς που παράγεται από το ολοκληρωμένο MAX6350. Η επιλογή γίνεται μέσω του βραχυκυκλωτήρα 16.
99	GND	Γείωση.
100	AVCC	Ακροδέκτης τροφοδοσίας των εσωτερικών 10-bit ADC. Συνδέεται με την τάση.

Πίνακας 6

Στις εικόνες που ακολουθούν παρουσιάζεται ο κύριος κόμβος.

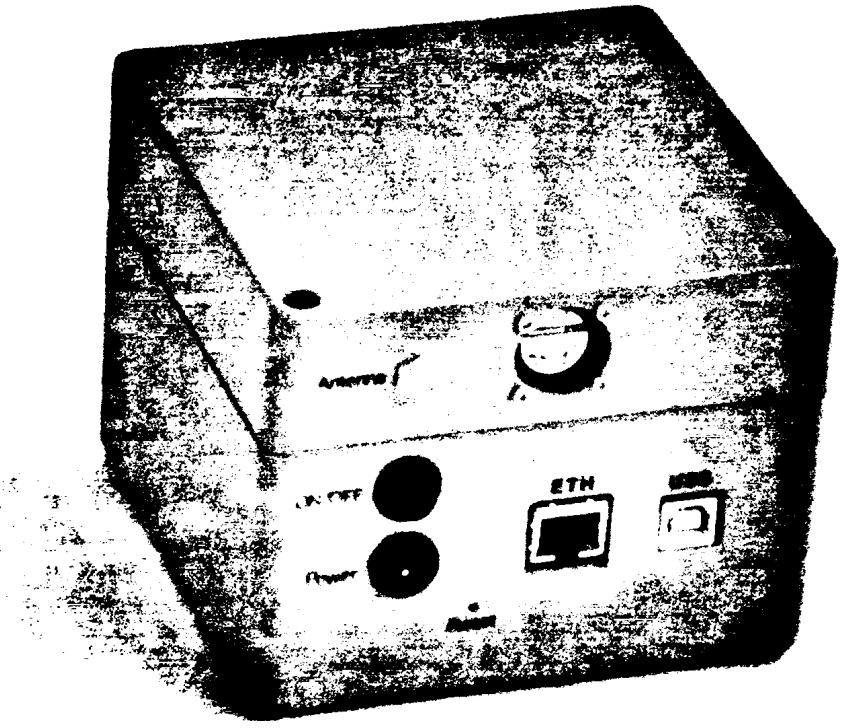


Εικόνα 36: Ο κύριος κόμβος συναρμολογημένος εκτός κουτιού.



Εικόνα 37: Ο κύριος κόμβος συναρμολογημένος μέσα στο κουτί του με ανοιχτό καπάκι

Εικόνα 38: Ο κύριος κόμβος (εμπρός όψη)



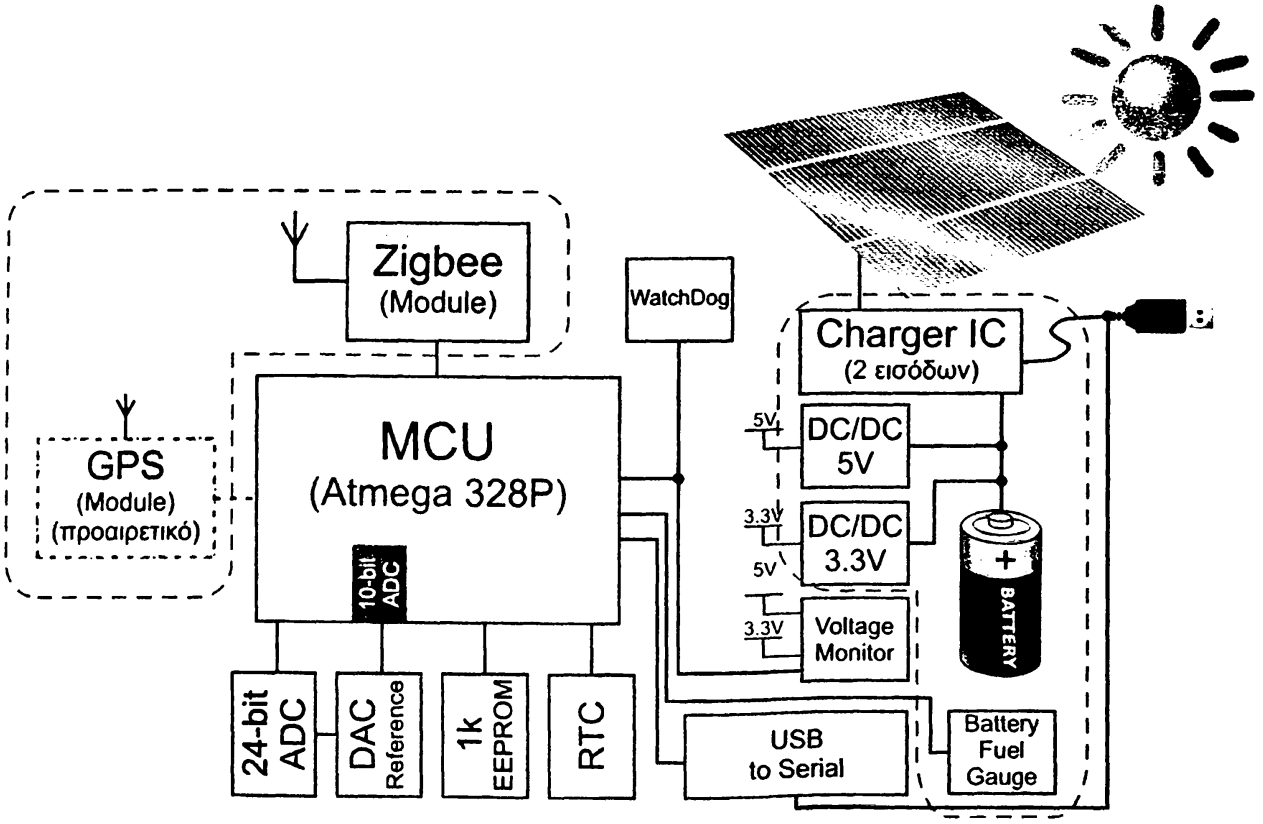
Εικόνα 39: Ο κύριος κόμβος (πίσω όψη)

6. Περιφερειακοί Κόμβοι

Η σχεδίαση των περιφερειακών κόμβων στηρίχθηκε στην ανάπτυξη ξεχωριστών κυκλωμάτων, όπως ακριβώς και ο κύριος κόμβος. Κάθε ένας από τους περιφερειακούς κόμβους αποτελείται από τρία κυκλώματα:

- Το κύκλωμα του μικροελεγκτή,
- το κύκλωμα RF,
- και το κύκλωμα τροφοδοσίας.

Τα κυκλώματα αυτά τοποθετούνται το ένα επάνω στο άλλο και συνδέονται μεταξύ τους με βύσματα. Οι περιφερειακοί κόμβοι σχεδιάστηκαν με αυτόν τον τρόπο για εξοικονόμηση χώρου, εκμετάλλευση όλου του όγκου που προσφέρει το κουτί καθώς και εύκολη εύρεση και αντικατάσταση κάποιου ελαττωματικού κυκλώματος. Η τροφοδοσία του κάθε κόμβου γίνεται από μία ενσωματωμένη μπαταρία ιόντων λιθίου. Στην εικόνα 40 παρουσιάζεται το λειτουργικό διάγραμμα με τα σημαντικότερα στοιχεία ενός περιφερειακού κόμβου.

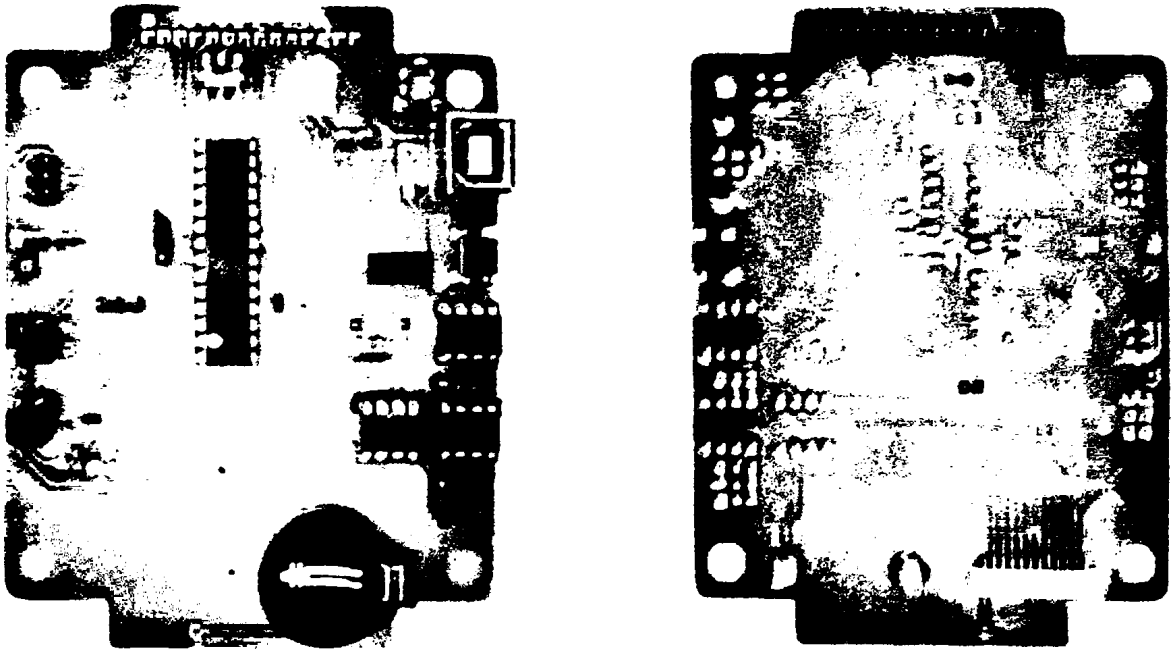


Εικόνα 40: Μπλοκ διάγραμμα περιφερειακού κόμβου

Παράλληλα, ένας από τους βασικούς στόχους σχεδίασης των περιφερειακών κόμβων, ήταν η προστασία όλων των σημαντικών εξαρτημάτων από την έκθεσή τους σε διάφορα καιρικά φαινόμενα, καθώς οι περιφερειακοί κόμβοι αναπτύσσονται συνήθως σε εξωτερικό περιβάλλον. Για το λόγο αυτό, το κουτί και τα βύσματα του κάθε κόμβου επιλέχθηκαν με προσοχή ώστε να είναι αεροστεγή και υδατοστεγή. Οι βαθμοί προστασίας (IP ratings) του κουτιού και των βυσμάτων είναι IP66 και IP67 αντίστοιχα.

5.1. Κύκλωμα Μικροελεγκτή

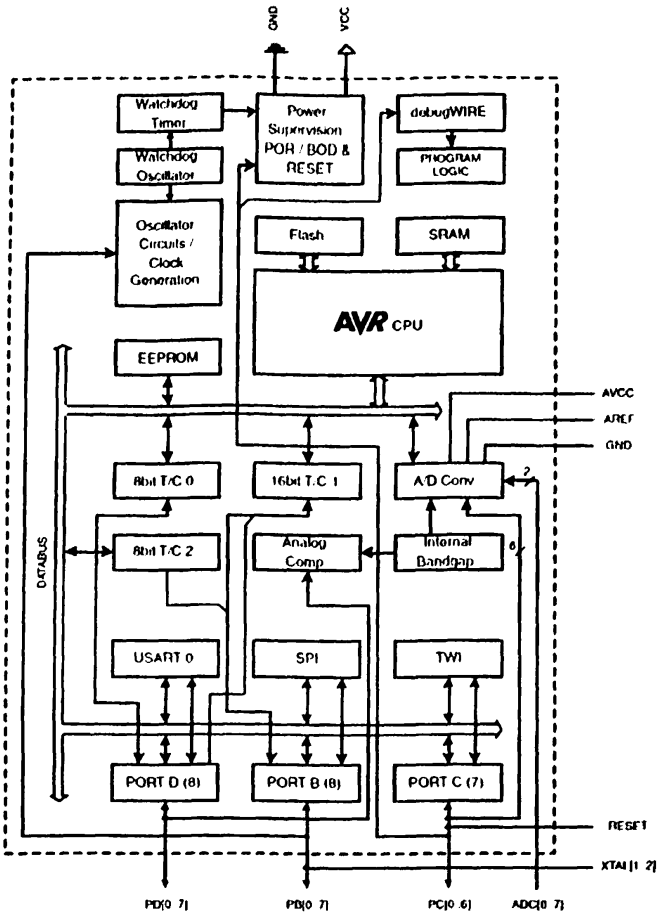
Το κύκλωμα του μικροελεγκτή (Εικόνα 41) σχεδιάστηκε με βάση το ελεύθερο (open-source) σχέδιο του αναπτυξιακού κυκλώματος Arduino Duemilanove ώστε να υπάρχει συμβατότητα με την πλατφόρμα Arduino. Όπως φαίνεται και από το διάγραμμα της εικόνας 40, το κύκλωμα του μικροελεγκτή αναπτύχθηκε αρχίζοντας από το βασικό σχέδιο του Arduino Duemilanove με την προσθήκη των επιμέρους κυκλωμάτων Real Time Clock, εξωτερικής EEPROM, Voltage Supervisor, εξωτερικού WatchDog Timer, εξωτερικού 24-bit ADC καθώς και ενός εξωτερικού DAC. Το κύκλωμα προγραμματισμού με το ολοκληρωμένο FT232R παρέμεινε ίδιο με το αρχικό σχέδιο. Οι επιμέρους συνιστώσες περιγράφονται στη συνέχεια.



Εικόνα 41: Το κύκλωμα του μικροελεγκτή.

5.1.1. Ο μικροελεγκτής ATmega328P

Ο μικροελεγκτής που χρησιμοποιήθηκε για τον περιφερειακό κόμβο είναι ο ATmega328P της εταιρίας Atmel. Πρόκειται για ένα μικροελεγκτή CMOS 8-bit εξαιρετικά χαμηλής κατανάλωσης, βασισμένος στη βελτιωμένη αρχιτεκτονική AVR RISC. Ένα από τα κορυφαία χαρακτηριστικά του, όπως και του μικροελεγκτή ATmega2560 που ήδη έχει περιγραφεί στο προηγούμενο κεφάλαιο, είναι ότι επιτυγχάνει εκτέλεση μιας εντολής σε ένα μόνο κύκλο ρολογιού. Αυτό πρακτικά σημαίνει ότι επιτυγχάνει ταχύτητες που αγγίζουν το 1 MIPS/MHz. Στην εικόνα 42 φαίνεται το διάγραμμα του μικροελεγκτή (Atmel Corporation, 2011).



Εικόνα 42: Διάγραμμα του μικροελεγκτή ATmega328P

Όπως η κεντρική μονάδα επεξεργασίας (CPU) AVR του μικροελεγκτή ATmega2560, έτσι και αντίστοιχη μονάδα του ATmega328P συνδυάζει ένα αρκετά πλούσιο σύνολο εντολών (Instruction Set) με 32 καταχωρητές (Registers) γενικής χρήσης. Όλοι οι καταχωρητές συνδέονται απευθείας με την αριθμητική λογική μονάδα (ALU), επιτρέποντας την ταυτόχρονη προσπέλαση δύο ανεξάρτητων καταχωρητών με μία εντολή σε έναν κύκλο ρολογιού όπως ήδη προαναφέρθηκε.

Τα κυριότερα χαρακτηριστικά του ATmega328P είναι τα εξής:

- 32kB μνήμη FLASH με δυνατότητα ταυτόχρονης ανάγνωσης και εγγραφής
- 1kB μνήμη EEPROM
- 2kB μνήμη SRAM
- Ταχύτητα ως 16MIPS (@16MHz)
- 23 I/O γενικής χρήσης
- 32 καταχωρητές γενικής χρήσης
- 2 Timers/Counters με δυνατότητα σύγκρισης και PWM
- 1 USARTs
- 2-Wire Serial Interface (TWI ή αλλιώς I^2C)
- SPI serial port (Master ή Slave)
- 10-Bit ADC 8 καναλιών
- 6 Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, και Extended Standby
- Εσωτερικό, προγραμματιζόμενο Watchdog Timer

Για την εφαρμογή μας επιλέχθηκε το footprint PDIP-28. Στον μικροελεγκτή περάστηκε το κατάλληλο bootloader ώστε να είναι συμβατός με την πλατφόρμα Arduino.

6.1.2. RTC (Real Time Clock)

Χρησιμοποιήθηκε το ολοκληρωμένο DS1307 η περιγραφή του οποίου έγινε στο κεφάλαιο 5.2.2.

6.1.3. Μνήμη EEPROM

Χρησιμοποιήθηκε το ολοκληρωμένο 24AA1025 που περιγράφηκε στο κεφάλαιο 5.2.1.

6.1.4. Voltage Supervisor

Χρησιμοποιήθηκε το ολοκληρωμένο DS1834 που περιγράφηκε στο κεφάλαιο 5.7.

6.1.5. WatchDog Timer

Χρησιμοποιήθηκε το ολοκληρωμένο MAX6371 που περιγράφηκε στο κεφάλαιο 5.7.

6.1.6. Το ολοκληρωμένο κύκλωμα FT232R (USB-to-Serial)

Για τον προγραμματισμό του μικροελεγκτή, καθώς για την επικοινωνία του με έναν Η/Υ χρησιμοποιήθηκε η σειριακή του θύρα. Όπως στον κύριο κόμβο, έτσι και στους περιφερειακούς έχει χρησιμοποιηθεί ένα ενδιάμεσο κύκλωμα που κάνει μετατροπή των σημάτων από το δίαυλο USB σε σειριακό και αντίστροφα από σειριακό σε USB. Το ενδιάμεσο αυτό κύκλωμα, σχεδιάστηκε και υλοποιήθηκε στο κύκλωμα του μικροελεγκτή με βάση το ελεύθερο σχέδιο του αναπτυξιακού κυκλώματος Arduino Duemilanove όπως προαναφέρθηκε, με τη χρήση του ολοκληρωμένου FT232R της εταιρίας FTDI (FTDI, 2011b).

6.1.7. 24-bit ADC

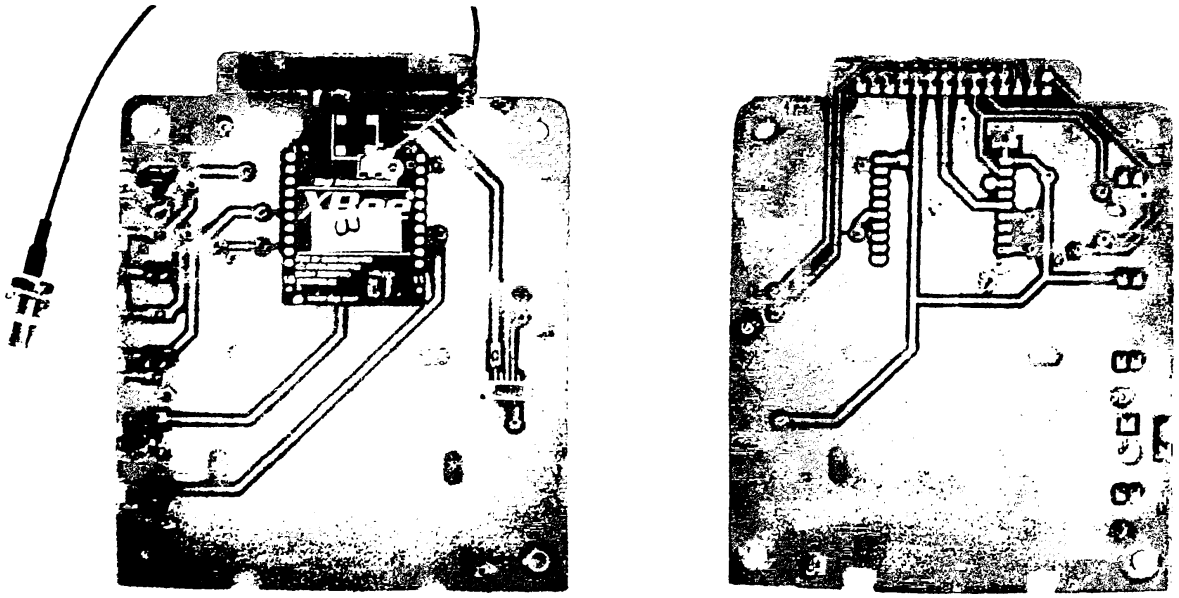
Ο κάθε περιφερειακός κόμβος είναι εξοπλισμένος με ένα μετατροπέα αναλογικού σήματος σε ψηφιακό (Analog-to-Digital ή ADC) με ανάλυση 24 bit. Ο μετατροπέας αυτός είναι ο LTC2400 της εταιρίας Linear (Linear Technology Corporation, 1998). Η τάση τροφοδοσίας του μπορεί να κυμαίνεται από τα 2,7V έως και τα 5,5V. Η επικοινωνία με το μικροελεγκτή γίνεται μέσω του διαύλου SPI. Ως τάση αναφοράς χρησιμοποιείται η τάση τροφοδοσίας ή η τάση που παράγεται από το DAC για το οποίο θα γίνει αναφορά παρακάτω. Η επιλογή επιτυγχάνεται με τη χρήση ενός βραχυκυκλωτήρα (jumper).

6.1.8. 12-bit DAC

Όπως αναφέρθηκε στο κεφάλαιο 6.1.7 ως τάση αναφοράς της μονάδας ADC μπορεί να χρησιμοποιηθεί η τάση από την έξοδο της μονάδας DAC (Digital-to-Analog) με ανάλυση 12 bit που έχει τοποθετηθεί στον κάθε περιφερειακό κόμβο. Ο DAC που χρησιμοποιήθηκε είναι ο MCP4725 της εταιρίας Microchip (Microchip Corporation, 2009). Η τάση τροφοδοσίας του μπορεί να κυμαίνεται από τα 2,7V ως και τα 5,5V. Η επικοινωνία με το μικροελεγκτή γίνεται μέσω του διαύλου I^2C. Στη συγκεκριμένη εφαρμογή η μονάδα DAC παίρνει την τάση τροφοδοσίας και να την επιμερίζει σε 4096 τμήματα. Σε μία τάση τροφοδοσίας 5V, για παράδειγμα, κάθε τμήμα θα αντιστοιχεί σε 0,001220703125V.

6.2. Κύκλωμα RF

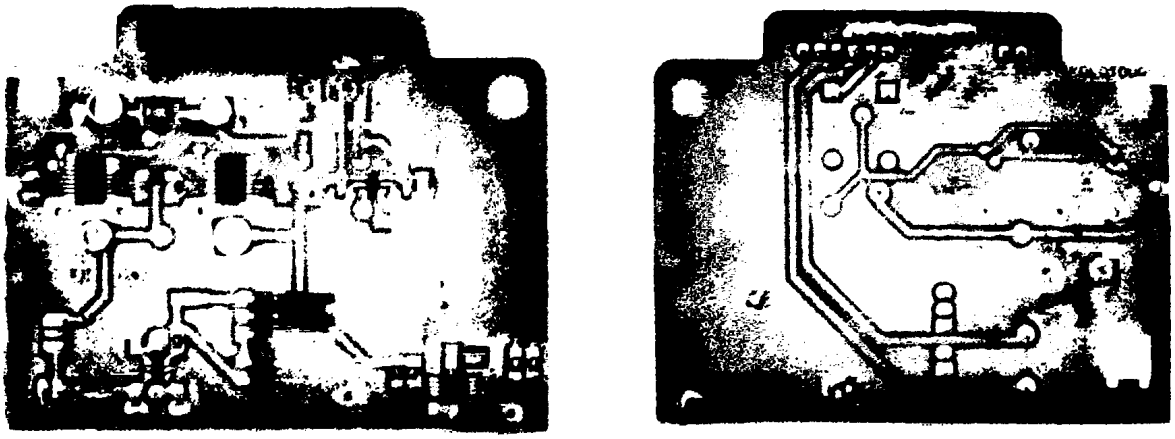
Το κύκλωμα RF φιλοξενεί τα τμήματα του περιφερειακού κόμβου που ασχολούνται με ραδιοσυχνότητες. Συγκεκριμένα στο κύκλωμα RF φιλοξενείται ο ασύρματος πομποδέκτης XBee και προαιρετικά ο δέκτης GPS (EM-408 της εταιρίας USGlobalSat (USGlobalSat Inc., 2011)). Δεν θα γίνει αναλυτική περιγραφή του XBee εδώ επειδή ήδη έχει γίνει στο κεφάλαιο 4.5 (ο ασύρματος πομποδέκτης XBee). Το κύκλωμα είναι εφοδιασμένο με φωτοδιόδους (LEDs) και έναν πιεστικό διακόπτη (push-button) συνδεδεμένα στα σήματα 3,3 V Power, DIN, DOUT, RSSI, Associate και Commission για αποσφαλμάτωση (debugging). Όταν το κύκλωμα τοποθετηθεί στο κουτί μαζί με τα άλλα κυκλώματα, οι φωτοδίοδοι μπορούν να απενεργοποιηθούν αν αφαιρεθούν όλοι οι βραχυκυκλωτήρες (jumpers) για λόγους κατανάλωσης. Στην εικόνα 43 φαίνεται το κύκλωμα RF στο οποίο έχει τοποθετηθεί η μονάδα XBee.



Εικόνα 43: Το κύκλωμα RF

6.3. Κύκλωμα τροφοδοσίας

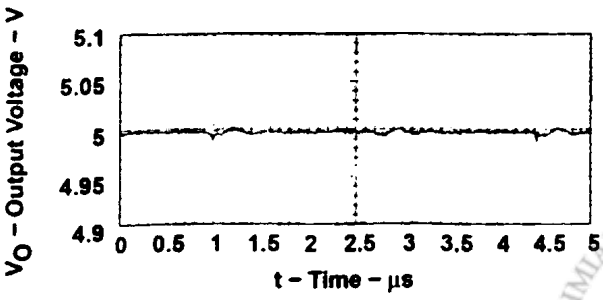
Το κύκλωμα τροφοδοσίας πέρα από την προφανή λειτουργία του, να τροφοδοτεί τα υπόλοιπα κυκλώματα με σταθερές τάσεις 5V και 3,3V, είναι υπεύθυνο και για τη φόρτιση μιας ενσωματωμένης στον κόμβο μπαταρίας λιθίου χωρητικότητας 2000mAh. Επίσης το κύκλωμα είναι εφοδιασμένο με ένα ολοκληρωμένο κύκλωμα το οποίο παρακολουθεί την κατανάλωση του κόμβου (battery fuel-gauge) καθώς και τα αποθέματα της μπαταρίας σε πραγματικό χρόνο. Παράλληλα, το κύκλωμα προστατεύει όλα τα ηλεκτρονικά εξαρτήματα του κόμβου σε περίπτωση βραχυκυκλώματος με τη χρήση μιας επαναφερόμενης ασφάλειας PPTC (500mA). Στην εικόνα 44 παρουσιάζεται το κύκλωμα τροφοδοσίας.



Εικόνα 44: Το κύκλωμα τροφοδοσίας

6.3.1. TPS60110 Step-up Regulator

Για την παροχή των 5V χρησιμοποιήθηκε το ολοκληρωμένο TPS60110 της εταιρίας Texas Instruments (Texas Instruments, 2008). Πρόκειται για ένα κύκλωμα άντλησης φορτίου (Step-up charge pump) το οποίο παράγει τάση 5V±4% εντάσεως έως και 300mA από μια είσοδο 2,7V έως 5,4V. Το συγκεκριμένο ολοκληρωμένο επιλέχθηκε για την υψηλή απόδοση λειτουργίας του που φτάνει έως και το 90% και το χαμηλό κυματισμό εξόδου ο οποίος δεν ξεπερνά τα $10mV_{pp}$. Για τις ανάγκες της εφαρμογής χρησιμοποιήθηκαν δύο ολοκληρωμένα του τύπου αυτού, συνδεδεμένα παράλληλα σύμφωνα με τις οδηγίες και τις προδιαγραφές του τεχνικού εγχειριδίου, εφαρμόζοντας στην τελική έξοδο ένα φίλτρο LC για περαιτέρω μείωση του κυματισμού εξόδου (Εικόνα 45). Το συνολικό ρεύμα εξόδου είναι 600 mA. Το ολοκληρωμένο ενεργοποιείται εφαρμόζοντας μία τάση ίση με τη V_{IN} και απενεργοποιείται με μια τάση 0V (γείωση). Η ενεργοποίηση/απενεργοποίηση του ολοκληρωμένου γίνεται με τη χρήση του διακόπτη S2 που βρίσκεται στο κύκλωμα του μικροελεγκτή για ευκολότερη πρόσβαση.



Εικόνα 45: Ο κυματισμός της τάσης εξόδου συναρτήσει του χρόνου

6.3.2. Γραμμικός ρυθμιστής τάσης MIC5219 Linear Regulator

Για την παροχή των 3,3 V χρησιμοποιήθηκε το ολοκληρωμένο MIC5219 της εταιρίας Micrel (Micrel, 2009b). Πρόκειται για ένα γραμμικό ρυθμιστή τάσης (Linear Regulator) ο οποίος παράγει τάση 3,3V±1% εντάσεως έως και 500mA από μια είσοδο 4,3 V έως 12 V. Η πτώση τάσης δεν ξεπερνά τα 10mV σε χαμηλά φορτία, ενώ αγγίζει τα 500mV σε πλήρη φόρτο. Η τροφοδοσία του παρέχεται από το ρυθμιστή TPS60110. Η ενεργοποίηση/ απενεργοποίηση του ρυθμιστή MIC5219 γίνεται όπως ακριβώς γίνεται με το ρυθμιστή TPS60110.

6.3.3. Ο φορτιστής διπλής εισόδου MAX1555

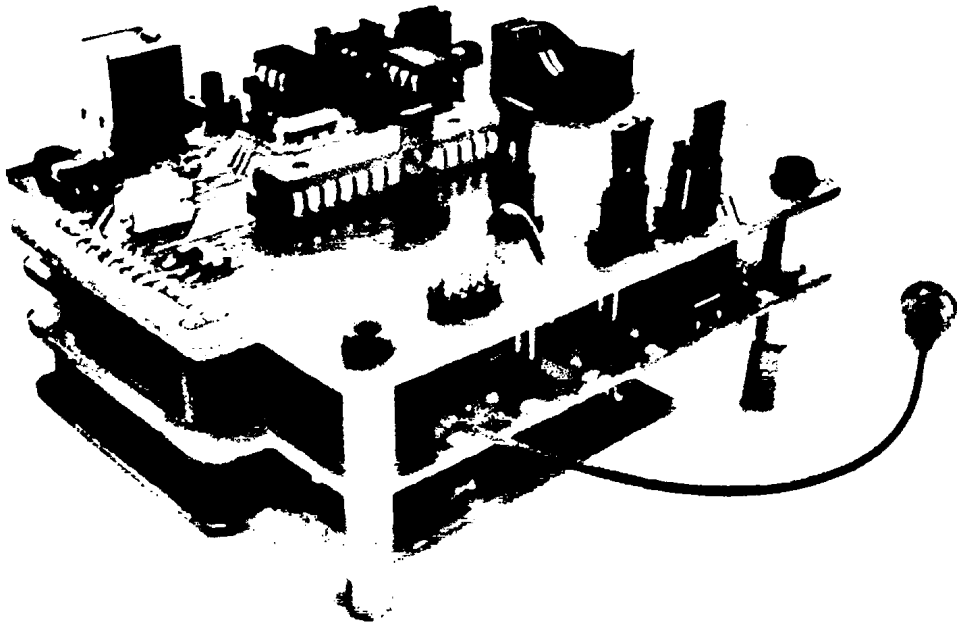
Το ολοκληρωμένο MAX1555 της εταιρίας Maxim (Maxim Integrated Products, Inc., 2003) είναι ένας φορτιστής μονών μπαταριών ιόντων λιθίου (Single-Cell Li+ ή Li-Ion). Ο φορτιστής μπορεί να δεχθεί δύο εισόδους, μία από θύρα USB και μία από μία άλλη πηγή τάσης DC έως 7V. Όταν εφαρμοστεί τάση από τη θύρα USB (5V) τότε το μέγιστο ρεύμα φόρτισης περιορίζεται στα 100mA. Στην περίπτωση που συνδεθεί μια άλλη πηγή τάσης DC (η τάση από τη θύρα USB και η τάση από την άλλη πηγή DC συνδέονται σε διαφορετικούς ακροδέκτες του ολοκληρωμένου) είτε ταυτόχρονα με την τάση από τη USB είτε ξεχωριστά, τότε επιλέγεται να φορτιστεί η μπαταρία από την πηγή φόρτισης DC. Σε αυτήν την περίπτωση το μέγιστο ρεύμα φόρτισης περιορίζεται στα 340mA. Στην παρούσα εφαρμογή ο φορτιστής συνδέθηκε με τη θύρα USB στο κύκλωμα του μικροελεγκτή και μία φωτοβολταϊκή κυψέλη ισχύος ενός Watt, τοποθετημένη στο διαφανές καπάκι του κουτιού από την εσωτερική του πλευρά για προστασία. Η συνδεσμολογία του είναι εξαιρετικά απλή χωρίς την ανάγκη για χρήση εξωτερικών FETs και διόδων. Ο φορτιστής επίσης διαθέτει ένδειξη φόρτισης με τη σύνδεση ενός LED στο ακροδέκτη /CHG. Μπορεί να χρησιμοποιηθεί είτε το LED που βρίσκεται στο κύκλωμα, είτε κάποιο εξωτερικό LED τοποθετημένο στο κουτί.

6.3.4. Ο δείκτης DS2782

Το ολοκληρωμένο κύκλωμα DS2782 της εταιρίας Maxim (Maxim Integrated Products, Inc., 2009) λειτουργεί ως «δείκτης» (fuel gauge) για τα ενεργειακά αποθέματα της μπαταρίας που φέρει ο κάθε περιφερειακός κόμβος μετρώντας την τάση και τη θερμοκρασία της μπαταρίας και του ρεύματος που έχει καταναλωθεί. Τα χαρακτηριστικά της μπαταρίας και οι παράμετροι της εφαρμογής που χρησιμοποιούνται για τους υπολογισμούς αποθηκεύονται στην ενσωματωμένη EEPROM που διαθέτει το ολοκληρωμένο. Μέσω του πρωτοκόλλου I^2C το DS2782 μπορεί να αναφέρει τη διαθέσιμη ισχύ της μπαταρίας σε mAh ή/και ποσοστό επί της εκατό (%) μιας πλήρως φορτισμένης μπαταρίας.

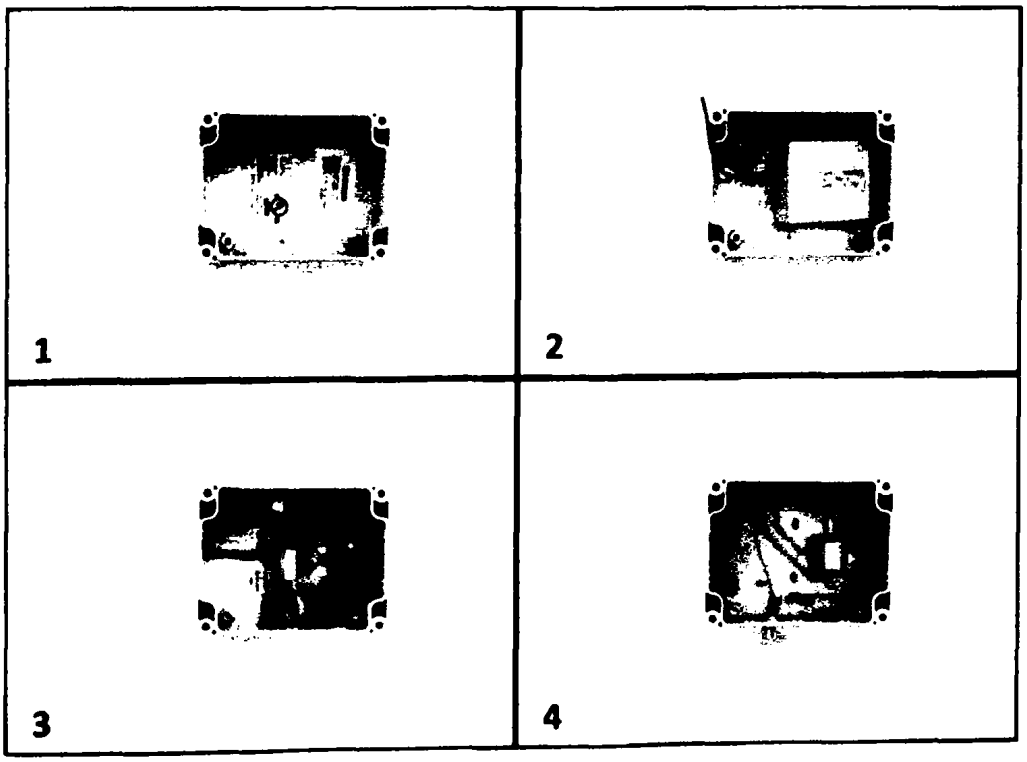
Συναρμολόγηση περιφερειακού κόμβου

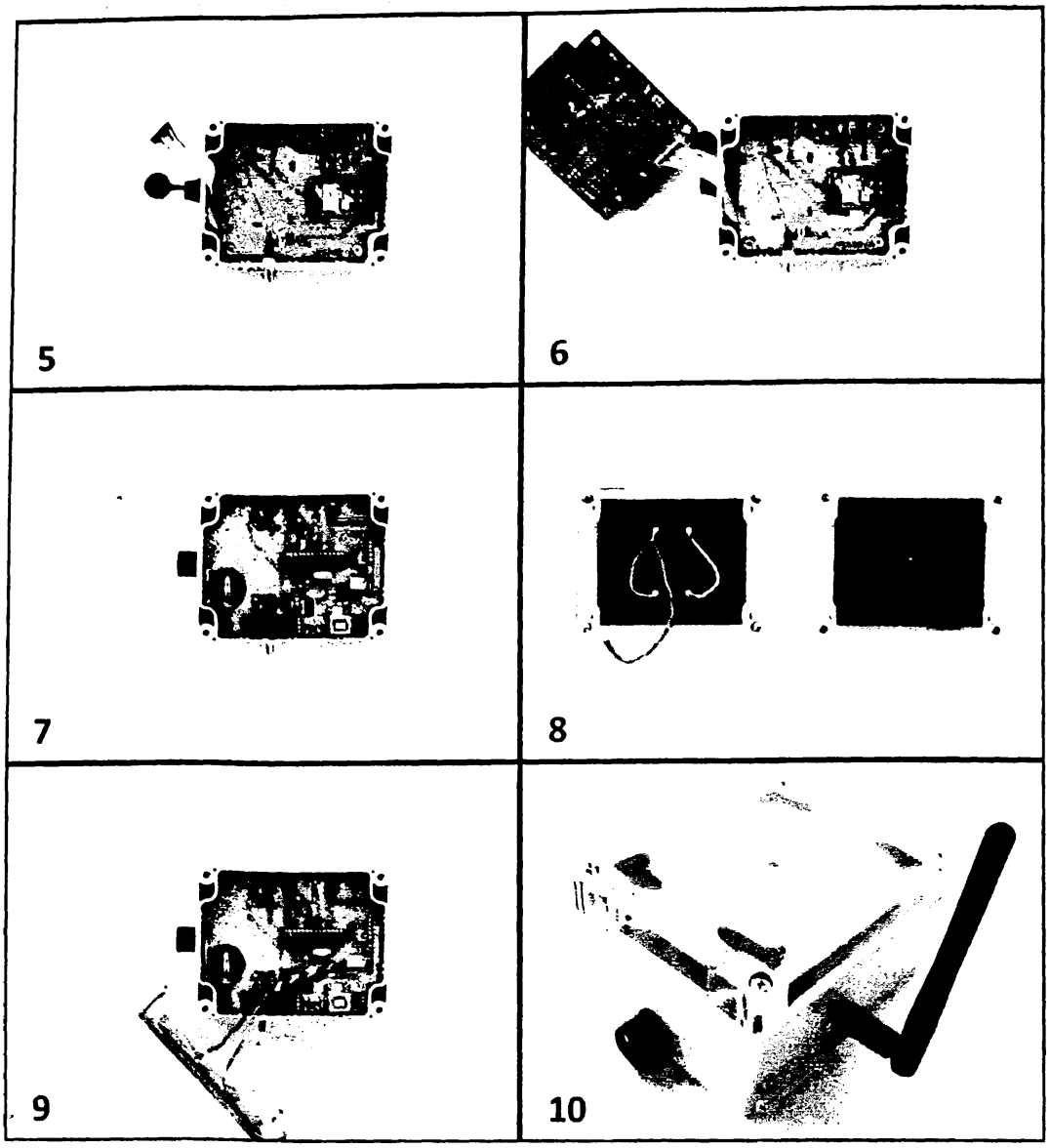
ην εικόνα 46 φαίνεται ένας περιφερειακός κόμβος συναρμολογημένος εκτός κουτιού. ούμε να διακρίνουμε από πάνω προς τα κάτω το κύκλωμα του μικροελεγκτή, το μα RF και το κύκλωμα τροφοδοσίας.



Εικόνα 46: Συναρμολογημένος κόμβος εκτός κουτιού.

ην εικόνα που ακολουθεί (Εικόνα 47) παρουσιάζονται τα στάδια συναρμολόγησης ενός ερειακού κόμβου εντός κουτιού.





Εικόνα 47: Στάδια συναρμολόγησης ενός περιφερειακού κόμβου εντός κουτιού

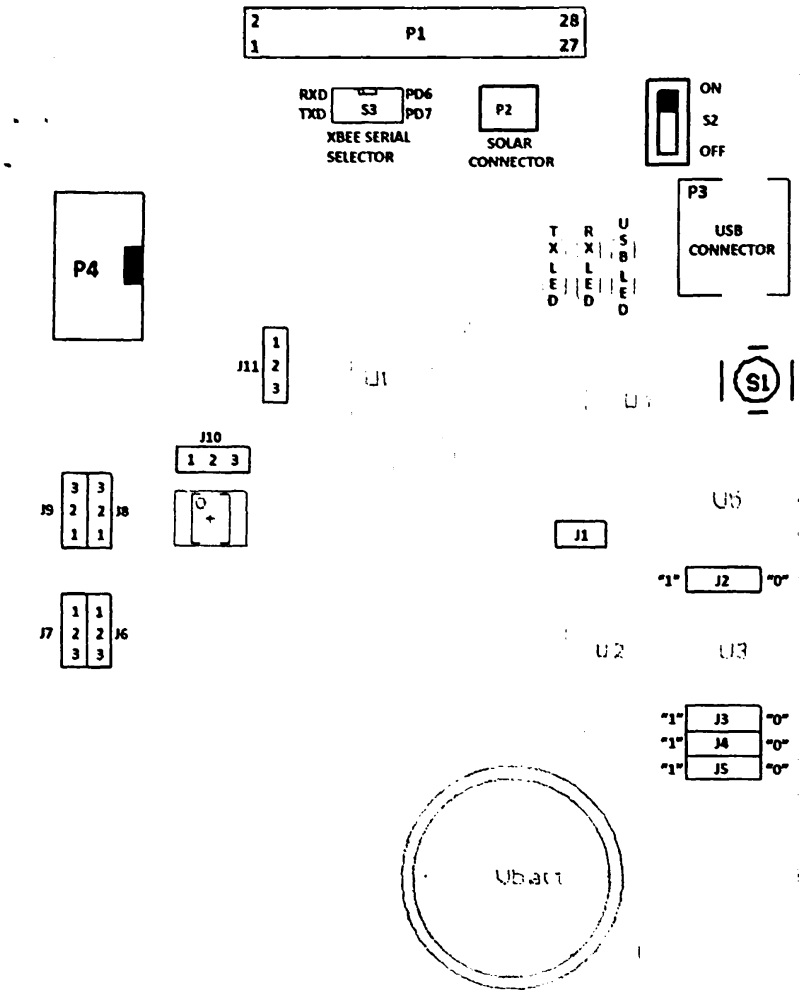
- **Εικόνα 47, Τμήμα 1:** Το κουτί ενός περιφερειακού κόμβου. Αφού **τρυπηθεί** στα σωστά σημεία για την τοποθέτηση των βυσμάτων είναι έτοιμο για να δεχθεί τα κυκλώματα.
- **Εικόνα 47, Τμήμα 2:** Στο κάτω μέρος του κουτιού τοποθετήθηκε η μπαταρία ιόντων λιθίου (Li-ion) και οι μεταλλικοί αποστάτες στο αριστερό μέρος του κουτιού μήκους 10mm.
- **Εικόνα 47, Τμήμα 3:** Ακριβώς επάνω από την μπαταρία, τοποθετείται το κύκλωμα τροφοδοσίας και ασφαλίζεται με μεταλλικούς αποστάτες μήκους 8mm. Επίσης τοποθετείται στο κουτί η φωτοδίοδος ένδειξης φόρτισης και το κάλυμμά του. Σειρά έχει η σύνδεση της φωτοδιόδου στο βύσμα με τους δύο ακροδέκτες, η τοποθέτηση του βραχυκυκλωτήρα στους δύο δεξιότερους ακροδέκτες του βύσματος δίπλα στο βύσμα της φωτοδιόδου και η σύνδεση της μπαταρίας στο λευκό βύσμα που βρίσκεται στην επάνω αριστερή γωνία του κυκλώματος.
- **Εικόνα 47, Τμήμα 4:** Σε αυτό το στάδιο τοποθετείται το κύκλωμα RF. Το κύκλωμα θα πατήσει επάνω στους μεταλλικούς αποστάτες ενώ οι ηλεκτρικές συνδέσεις με τα

υπόλοιπα κυκλώματα γίνεται από το βύσμα (αρσενικό-θηλυκό) που βρίσκεται στη δεξιά πλευρά του κυκλώματος. Το αρσενικό τμήμα του βύσματος (bottom layer) συνδέεται με το θηλυκό βύσμα του κυκλώματος τροφοδοσίας. Στο θηλυκό τμήμα του βύσματος (top layer) συνδέθηκε στη συνέχεια με το αρσενικό βύσμα του κυκλώματος του μικροελεγκτή. Ακολούθως τοποθετήθηκε η μονάδα Xbee, βιδώθηκε το βύσμα RP-SMA της κεραίας του στο κουτί και οι μεταλλικοί αποστάτες στις τέσσερις γωνίες του κυκλώματος για να πατήσει το κύκλωμα του μικροελεγκτή.

- **Εικόνα 47, Τμήμα 5:** Σειρά είχε η τοποθέτηση του καλωδίου με τα δύο βύσματα. Το καλώδιο από τη μια πλευρά διαθέτει ένα βύσμα JST 12 θέσεων το οποίο συνδέεται στο κύκλωμα του μικροελεγκτή και από την άλλη πλευρά διαθέτει ένα αδιάβροχο βύσμα 12 θέσεων το οποίο τοποθετείται στο κουτί. Με αυτόν τον τρόπο, ο μικροελεγκτής μπορεί να επικοινωνήσει με αισθητήρες και συσκευές που κάνουν χρήση ενός από τα πρωτόκολλα επικοινωνίας I2C, SPI και 1-Wire ή απλά διαθέτουν μία αναλογική έξοδο. Το βύσμα μπορεί επίσης να τροφοδοτήσει τη συσκευή ή τον αισθητήρα που θα συνδεθεί με τόση 3,3V και 5V.
- **Εικόνα 47, Τμήμα 6:** Αφού το ένα βύσμα του καλωδίου βιδωθεί και ασφαλιστεί με τις κατάλληλες φλάντζες αδιαβροχοποίησης στο κουτί, θα πρέπει να συνδεθεί το άλλο βύσμα καλωδίου (JST) στο κύκλωμα του μικροελεγκτή. Το αντίστοιχο βύσμα του κυκλώματος βρίσκεται στο bottom layer.
- **Εικόνα 47, Τμήμα 7:** Στη συνέχεια τοποθετήθηκε το κύκλωμα του μικροελεγκτή επάνω από το κύκλωμα RF πατώντας επάνω στου μεταλλικούς αποστάτες. Το κύκλωμα ασφαλίζει με την υπόλοιπη κατασκευή με τη χρήση μεταλλικών βιδών πάχους 3mm οι οποίες βιδώνουν στους μεταλλικούς αποστάτες στις τέσσερις γωνίες. Οι ηλεκτρικές συνδέσεις με τα υπόλοιπα κυκλώματα γίνεται από το βύσμα (αρσενικό) που βρίσκεται στη δεξιά πλευρά του κυκλώματος στο bottom layer.
- **Εικόνα 47, Τμήμα 8:** Σε αυτό το τμήμα της εικόνας φαίνονται οι δύο όψεις από το καπάκι του κουτιού το οποίο είναι διάφανο. Από την εσωτερική του πλευρά έχουν κολληθεί με εποξική ρητίνη δύο φωτοβολταϊκές κυψέλες, ισχύος 0,5Watt η κάθε μία, συνδεδεμένες παράλληλα. Στους δύο πόλους των φωτοβολταϊκών συνδέθηκε ένα καλώδιο με ένα βύσμα JST δύο θέσεων στην άλλη του άκρη.
- **Εικόνα 47, Τμήμα 9:** Στο σημείο αυτό, συνδέθηκε το JST βύσμα των φωτοβολταϊκών στο αντίστοιχο βύσμα του κυκλώματος του μικροελεγκτή.
- **Εικόνα 47, Τμήμα 10:** Αυτό είναι το τελευταίο στάδιο συναρμολόγησης. Ο περιφερειακός κόμβος είναι σχεδόν έτοιμος. Το μόνο που μένει είναι να κλείσει και να βιδωθεί το καπάκι και να βιδωθεί η κεραία της μονάδας Xbee.

6.5. Ρυθμίσεις λειτουργίας περιφερειακού κόμβου

Στην εικόνα 48 φαίνεται ένα τοπογραφικό διάγραμμα του κυκλώματος του μικροελεγκτή το οποίο περιέχει τις θέσεις διαφόρων βυσμάτων και βραχυκυκλωτήρων των οποίων η λειτουργία θα εξηγηθεί παρακάτω. Παράλληλα στο τέλος του κεφαλαίου θα παρατεθεί ένας πίνακας με τη συνδεσμολογία των ακροδεκτών του μικροελεγκτή ATmega328P (Πίνακας 8). Ο πίνακας αυτός είναι ιδιαίτερα χρήσιμος κατά τη συγγραφή ή τροποποίηση κώδικα.



Εικόνα 48: Τοπογραφικό διάγραμμα κυκλώματος μικροελεγκτή

- P1: Βύσμα επικοινωνίας με τα υπόλοιπα κυκλώματα. Όπως γίνεται αντιληπτό, τα υπόλοιπα δύο κυκλώματα διαθέτουν από ένα ίδιο βύσμα, συνεπώς όλα τα σήματα του βύσματος είναι κοινά σε όλα τα κυκλώματα. Τα σήματα του βύσματος φαίνονται στον πίνακα 7.

Α/Α	Όνομα	Λειτουργία	Α/Α	Όνομα	Λειτουργία
1	I^2C_SDA	Μεταφορά δεδομένων διαύλου I^2C	15	RESET(3,3V)	Σήμα Reset για 3,3V συσκευές
2	ZB_SLEEP	Ακροδέκτης για τη μετάβαση της μονάδας Xbee σε κατάσταση αναμονής	16	GND	Γείωση

3	I^2C_SCL	Ρολόι I^2C	17	5V	Τάση 3,3V
4	NC	Χωρίς Λειτουργία (Not Connected)	18	GND	Γείωση
5	GPS_RX	Ακροδέκτης λήψης δεδομένων του δέκτη GPS	19	3,3V	Τάση 5V
6	NC	Χωρίς Λειτουργία (Not Connected)	20	GND	Γείωση
7	GPS_TX	Ακροδέκτης αποστολής δεδομένων του δέκτη GPS	21	SOLAR	Ισχύς από τη φωτοβολταϊκή κυψέλη
8	GND	Γείωση	22	GND	Γείωση
9	XBEE_RX	Ακροδέκτης λήψης δεδομένων της μονάδας Xbee	23	USB	Τάση 5V από τη θύρα USB
10	GND	Γείωση	24	NC	Χωρίς Λειτουργία (Not Connected)
11	XBEE_TX	Ακροδέκτης λήψης δεδομένων της μονάδας Xbee	25	ON-OFF	Ο ακροδέκτης Enable των Voltage Regulators
12	GND	Γείωση	26	NC	Χωρίς Λειτουργία (Not Connected)
13	GPS_EN	Ο ακροδέκτης Enable του GPS	27	PACK+	Ο θετικός ακροδέκτης της μπαταρίας
14	GND	Γείωση	28	NC	Χωρίς Λειτουργία (Not Connected)

Πίνακας 7

- **P2: Βύσμα τροφοδοσίας από τη φωτοβολταϊκή κυψέλη.**
- **P3: USB βύσμα.** Χρησιμοποιείται για τον προγραμματισμό του μικροελεγκτή (Αφού προγραμματιστεί πρώτα με το κατάλληλο bootloader μέσω της θύρας προγραμματισμού) και τη φόρτιση της μπαταρίας.
- **P4: Θύρα προγραμματισμού ISP (In-System Programming).** Σε αυτό το βύσμα καταλήγουν τα σήματα του πρωτοκόλλου SPI, το RESET, η τάση και η γείωση. Με τη σύνδεση ενός εξωτερικού προγραμματιστή περνάει το πρόγραμμα στη μνήμη flash του μικροελεγκτή. Στη συγκεκριμένη περίπτωση θα χρησιμοποιηθεί μόνο μία φορά για να «καεί» ο Arduino bootloader. Αφού γίνει αυτό, το πρόγραμμα μπορεί να περνάει στη μνήμη flash χωρίς τη χρήση κάποιας εξωτερικής συσκευής, μέσω της μονάδας USB-to-Serial.
- **S1**: Διακόπτης Reset.
- **S2**: Διακόπτης ON-OFF. Συνδέει τους ακροδέκτες Enable των Voltage Regulators είτε με την τάση της μπαταρίας (ON), είτε με την γείωση (OFF).
- **S3**: Ο διακόπτης αυτός συνδέει τους ακροδέκτες RX/TX της μονάδας Xbee είτε με τους ακροδέκτες PD0/PD1 (Hardware Serial) του μικροελεγκτή, είτε με τους ακροδέκτες PD6/PD7. Στην περίπτωση που επιλεγεί η σύνδεση με τους ακροδέκτες PD0/PD1 κατά τη συγγραφή του λογισμικού χρησιμοποιείται κανονικά η σειριακή θύρα του μικροελεγκτή (Hardware Serial). Κατά τον προγραμματισμό όμως του μικροελεγκτή η σειριακή θύρα θα πρέπει να απομονωθεί από τη μονάδα Xbee γυρνώντας το διακόπτη δεξιά στο PD6/PD7. Στην περίπτωση που επιλεγεί η

σύνδεση με τους ακροδέκτες PD6/PD7 κατά τη συγγραφή του λογισμικού θα πρέπει να χρησιμοποιηθεί η βιβλιοθήκη «Software Serial» η οποία θα «μετατρέψει» τους δύο απλούς ακροδέκτες I/O (PD6 και PD6) σε μία επιπλέον σειριακή θύρα.

- **USB LED**: Πράσινο LED. Ανάβει όταν υπάρχει USB σύνδεση με έναν Η/Υ.
- **RX LED**: Κόκκινο LED. Ανάβει όταν ο μικροελεγκτής λαμβάνει δεδομένα μέσω της θύρας USB.
- **TX LED**: Πράσινο LED. Ανάβει όταν ο μικροελεγκτής αποστέλλει δεδομένα μέσω της θύρας USB
- **J1**: Βραχυκυκλωτήρας ο οποίος αν τοποθετηθεί, συνδέει την έξοδο του WatchDog Timer με το Reset
- **J2**: Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη Write Protect της μνήμης EEPROM είτε με την τάση (λογικό «1»), είτε με τη γείωση (λογικό «0»).Όταν ενεργοποιηθεί (λογικό «1»), αποτρέπει την λειτουργία εγγραφής στη μνήμη ενώ αφήνει ανεπηρέαστη τη λειτουργία ανάγνωσης. Όταν απενεργοποιηθεί (λογικό «0») επιτρέπει την εγγραφή και ανάγνωση της μνήμης.
- **J3**: Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη διευθυνσιοδότησης A0 της μνήμης EEPROM είτε με την τάση (λογικό «1»), είτε με τη γείωση (λογικό «0»). Η διεύθυνση της μνήμης που χρησιμοποιήθηκε στην εφαρμογή (24LC1025) είναι «1010B0A1A0». Το bit B0 χωρίζει τη μνήμη σε δύο ίσα τμήματα των 512Kbits το κάθε ένα. Ορίζοντάς το «0» ή «1» προγραμματιστικά μπορεί να εγγραφεί ή να αναγνωσθεί το αντίστοιχο τμήμα.
- **J4**: Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη διευθυνσιοδότησης A1 της μνήμης EEPROM είτε με την τάση (λογικό «1»), είτε με τη γείωση (λογικό «0»).
- **J5**: Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη διευθυνσιοδότησης A2 της μνήμης EEPROM είτε με την τάση (λογικό «1»), είτε με τη γείωση (λογικό «0»). Στη συγκεκριμένη περίπτωση που χρησιμοποιείται η μνήμη 24LC1025, ο ακροδέκτης A2 πρέπει να είναι πάντα συνδεδεμένος με την τάση.
- **J6**: Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη διευθυνσιοδότησης A0 του DAC MCP4725 με την τάση (λογικό «1»), είτε με τη γείωση (λογικό «0»). Η διεύθυνση του MCP4725 είναι «110000A0».
- **J7**: Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη VCC του MCP4725 με την τάση 5V του κυκλώματος. Αν ο Βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «1» και «2» τότε το ολοκληρωμένο τροφοδοτείται με ισχύ και λειτουργεί. Ο ακροδέκτης «3» δεν συνδέεται πουθενά, συνεπώς αν τοποθετηθεί στους «2» και «3» το ολοκληρωμένο δεν λειτουργεί. Συνιστάται να τοποθετείται ο βραχυκυκλωτήρας στους ακροδέκτες «2» και «3» αν δεν χρησιμοποιείται για εξοικονόμηση ενέργειας.
- **J8**: Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη V_{REF} του εξωτερικού 24-bit ADC (LTC2400) είτε με την τάση 5V του κυκλώματος, είτε με την έξοδο του DAC. Αν ο βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «1» και «2» τότε η V_{REF} βραχυκυκλώνεται με την τάση 5V. Αν ο βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «2» και «3» τότε η V_{REF} βραχυκυκλώνεται με την έξοδο του DAC.
- **J9**: Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη VCC του εξωτερικού 24-bit ADC (LTC2400) με την τάση 5V του κυκλώματος. Αν ο βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «1» και «2» τότε το ολοκληρωμένο τροφοδοτείται με ισχύ και λειτουργεί. Ο ακροδέκτης «3» δεν συνδέεται πουθενά, συνεπώς αν τοποθετηθεί στους «2» και «3» το ολοκληρωμένο δεν λειτουργεί. Συνιστάται να τοποθετείται ο βραχυκυκλωτήρας στους ακροδέκτες «2» και «3» αν δεν χρησιμοποιείται για εξοικονόμηση ενέργειας.

- **J10**: Βραχυκυκλωτήρας που συνδέει την αναλογική είσοδο του περιφερειακού κόμβου (ένα από τα σήματα του αδιάβροχου βύσματος κουτιού) είτε με τον ακροδέκτη V_{IN} του εξωτερικού ADC, είτε με τον ακροδέκτη ADC0 του μικροελεγκτή. Αν ο Βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «1» και «2» τότε η αναλογική είσοδος του κόμβου θα μετατραπεί σε ψηφιακό σήμα από τον εξωτερικό 24-bit ADC. Αν ο βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «2» και «3» τότε η αναλογική είσοδος του κόμβου θα μετατραπεί σε ψηφιακό σήμα από τον εσωτερικό 10-bit ADC του μικροελεγκτή.
- **J11**: Βραχυκυκλωτήρας που συνδέει τον ακροδέκτη A_{REF} του εσωτερικού 10-bit ADC του μικροελεγκτή είτε με την τάση 5V του κυκλώματος, είτε με την έξοδο του DAC. Αν ο Βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «1» και «2» τότε η A_{REF} βραχυκυκλώνεται με την τάση 5V. Αν ο Βραχυκυκλωτήρας τοποθετηθεί στους ακροδέκτες «2» και «3» τότε η A_{REF} βραχυκυκλώνεται με την έξοδο του DAC.

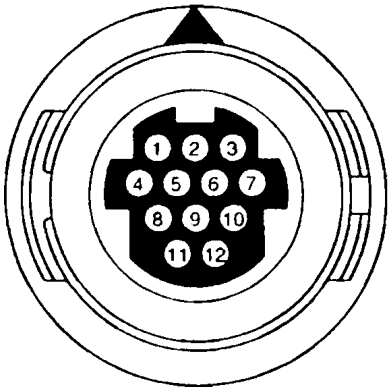
Α/Α	Όνομα Ακροδέκτη	Λειτουργία-Σύνδεση
1	PC6/**Reset**	Reset του μικροελεγκτή. Μπορεί να ενεργοποιηθεί από το FT232R κατά τον προγραμματισμό, το Watchdog Timer, το Voltage Supervisor και χειροκίνητα.
2	PD0/RXD	Ακροδέκτης λήψης δεδομένων της σειριακής θύρας (Hardware Serial). Συνδέεται με τον ακροδέκτη αποστολής δεδομένων (TX) του FT232R μόνιμα και με τον ακροδέκτη αποστολής δεδομένων (TX) της μονάδας Xbee αν ο διακόπτης S3 γυριστεί αριστερά.
3	PD1/TXD	Ακροδέκτης αποστολής δεδομένων της σειριακής θύρας (Hardware Serial). Συνδέεται με τον ακροδέκτη λήψης δεδομένων (RX) του FT232R μόνιμα και με τον ακροδέκτη λήψης δεδομένων (RX) της μονάδας Xbee αν ο διακόπτης S3 γυριστεί αριστερά.
4	PD2/INT0	-
5	PD3/INT1	Ο ακροδέκτης αυτός συνδέεται με την είσοδο του WatchDog Timer.
6	PD4	Ακροδέκτης αποστολής (TX) σειριακών δεδομένων προς το δέκτη GPS. Πρόκειται για Software Serial που υλοποιείται με τη χρήση κατάλληλης βιβλιοθήκης. Συνδέεται με τον ακροδέκτη αποστολής δεδομένων (RX) του δέκτη GPS.
7	VCC	Τάση τροφοδοσίας 5V.
8	GND	Γείωση.
9	PB6/OSC1	Σύνδεση με ταλαντωτή.
10	PB7/OSC2	Σύνδεση με ταλαντωτή.
11	PD5	Ακροδέκτης λήψης (RX) σειριακών δεδομένων από το δέκτη GPS. Πρόκειται για Software Serial που υλοποιείται με τη χρήση κατάλληλης βιβλιοθήκης. Συνδέεται με τον ακροδέκτη αποστολής δεδομένων (TX) του δέκτη GPS.
12	PD6	Ακροδέκτης αποστολής (TX) σειριακών δεδομένων προς τη μονάδα Xbee. Πρόκειται για Software Serial που υλοποιείται με τη χρήση κατάλληλης βιβλιοθήκης. Συνδέεται με τον ακροδέκτη αποστολής δεδομένων (RX) της μονάδας Xbee.

13	PD7	Ακροδέκτης λήψης (RX) σειριακών δεδομένων της μονάδας Xbee. Πρόκειται για Software Serial που υλοποιείται με τη χρήση κατάλληλης βιβλιοθήκης. Συνδέεται με τον ακροδέκτη αποστολής δεδομένων (TX) της μονάδας Xbee.
14	PB0	Ο ακροδέκτης αυτός συνδέεται με τον ακροδέκτη Enable του δέκτη GPS. Για να τεθεί σε λειτουργία ο δέκτης GPS θα πρέπει ο ακροδέκτης PB0 να δώσει λογικό «1». Αντίστοιχα θα πρέπει να δώσει λογικό «0» για να τον απενεργοποιήσει.
15	PB1	Στον ακροδέκτη αυτό θα πρέπει να υλοποιηθεί προγραμματιστικά το πρωτόκολλο 1-wire. Έχει συνδεθεί επάνω του μια pull-up αντίσταση 4,7kΩ για υποστήριξη παρασιτικής τροφοδοσίας και συνδέεται απ' ευθείας με το αδιάβροχο βύσμα κουτιού.
16	PB2/SS	Ακροδέκτης SS (Slave Select) του διαύλου SPI. Συνδέεται με το αδιάβροχο βύσμα κουτιού.
17	PB3/MOSI	Ακροδέκτης MOSI (Master Out-Slave In) του διαύλου SPI. Συνδέεται με το αδιάβροχο βύσμα κουτιού.
18	PB4/MISO	Ακροδέκτης MISO (Master In-Slave Out) του διαύλου SPI. Συνδέεται με το αδιάβροχο βύσμα κουτιού.
19	PB5/SCK	Ακροδέκτης SCK (Serial Clock) του διαύλου SPI. Συνδέεται με το αδιάβροχο βύσμα κουτιού.
20	AVCC	Ακροδέκτης τροφοδοσίας του εσωτερικού 10-bit ADC. Συνδέεται με την τάση.
21	AREF	Τάση αναφοράς για τον εσωτερικό 10-bit ADC. Συνδέεται είτε με την τάση 5V, είτε με τον DAC MCP4725 με τη χρήση του Βραχυκυκλωτήρα J11.
22	GND	Γείωση.
23	PC0/ADC0	Αναλογική είσοδος. Συνδέεται με την αναλογική είσοδο του περιφερειακού κόμβου (αδιάβροχο βύσμα κουτιού) βραχυκυκλώνοντας τους ακροδέκτες «2» και «3» στο βραχυκυκλωτήρα J10.
24	PC1/ADC1	Ακροδέκτης ένδειξης λειτουργίας της μονάδας Xbee. Συνδέεται με τον ακροδέκτη ON/SLEEP της μονάδας Xbee. Όταν διαβάζει ο μικροελεγκτής λογικό «1» η μονάδα λειτουργεί κανονικά. Όταν διαβάζει λογικό «0» η μονάδα είναι σε sleep mode.
25	PC2/ADC2	-
26	PC3/ADC3	Ακροδέκτης SS (Slave Select) του εξωτερικού 24-bit ADC.
27	PC4/ADC4/SDA	Ακροδέκτης SDA του διαύλου I^2C. Έχει συνδεθεί επάνω του μια pull-up αντίσταση 2,2kΩ και συνδέεται με το αδιάβροχο βύσμα κουτιού.
28	PC5/ADC5/SCL	Ακροδέκτης SCL του διαύλου I^2C. Έχει συνδεθεί επάνω του μια pull-up αντίσταση 2,2kΩ και συνδέεται με το αδιάβροχο βύσμα κουτιού.

Πίνακας 8

6.6. Σύνδεση με Αισθητήρες

Όπως προαναφέρθηκε, ο μικροελεγκτής μπορεί να επικοινωνήσει με αισθητήρες και συσκευές που βρίσκονται έξω από το κουτί του περιφερειακού κόμβου οι οποίοι κάνουν χρήση ενός από τα πρωτόκολλα επικοινωνίας I2C, SPI και 1-Wire ή απλά διαθέτουν μία αναλογική έξοδο. Αυτό καθίσταται εφικτό με τη χρήση ενός αδιάβροχου βύσματος 12 θέσεων που έχει τοποθετηθεί στο κουτί (HR30-7R-12P της εταιρίας HIROSE). Το βύσμα μπορεί επίσης να τροφοδοτήσει τη συσκευή ή τον αισθητήρα που θα συνδεθεί με τάση 3,3V και 5V. Στην εικόνα 49 φαίνεται η διάταξη των ακροδεκτών του βύσματος όπως αυτή φαίνεται από το εξωτερικό μέρος του κουτιού, καθώς και η λειτουργία του κάθε ακροδέκτη στον πίνακα που ακολουθεί (Πίνακας 9).



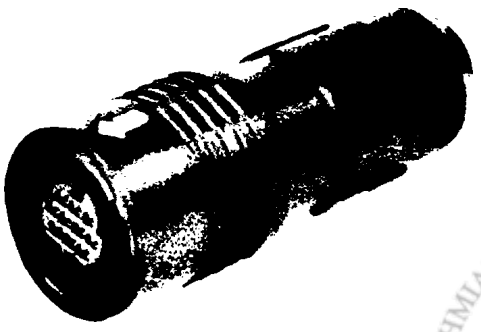
Εικόνα 49

A/A	Λειτουργία
1	Γείωση
2	+5V
3	/RESET
4	Αναλογική Είσοδος
5	1-Wire πρωτόκολλο
6	Slave Select (SS) SPI πρωτοκόλλου
7	Master Out-Slave In (MOSI) SPI πρωτοκόλλου
8	Master In-Slave Out (MISO) SPI πρωτοκόλλου
9	Serial Clock (SCK) SPI πρωτοκόλλου
10	SDA I^2C πρωτοκόλλου
11	SCL I^2C πρωτοκόλλου
12	+3.3V

Πίνακας 9

Το αντίστοιχο βύσμα καλωδίου που θα πρέπει να χρησιμοποιηθεί με το βύσμα του κουτιού του περιφερειακού κόμβου είναι το HR30-7P-12S της εταιρίας HIROSE (Εικόνα 50).

Εικόνα 50: Το βύσμα καλωδίου HR30-7P-12S



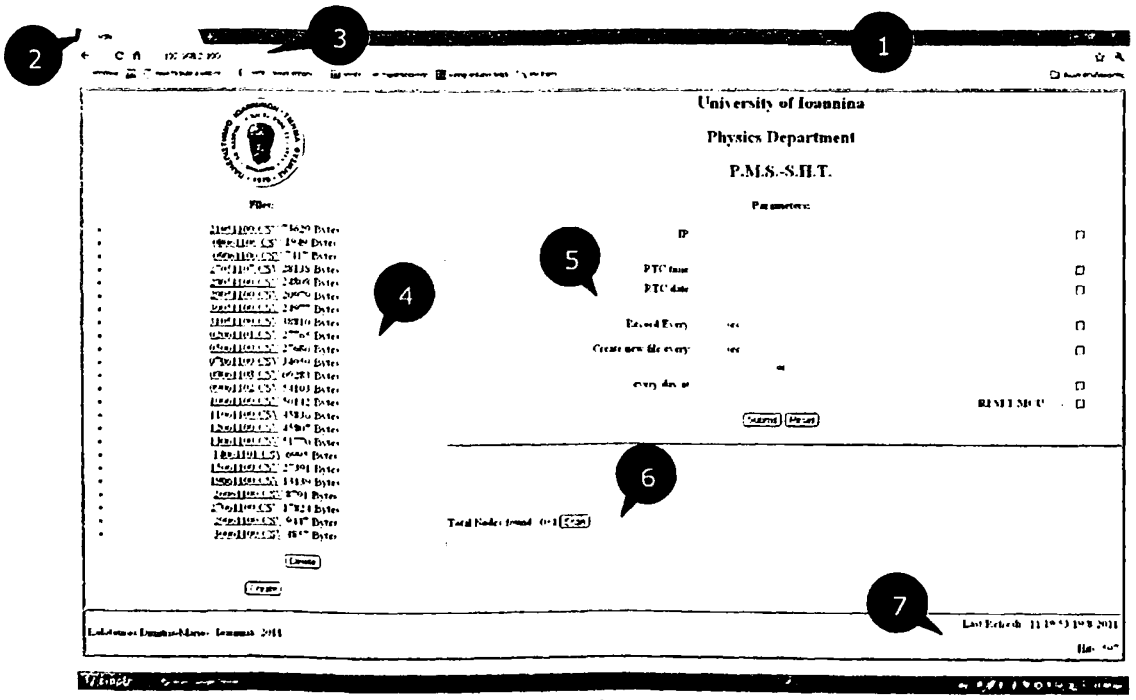
7. Διεπαφή Χρήστη (User Interface)

7.1. Γενικά

Ο σχεδιασμός διεπαφής χρήστη είναι ένα υποσύνολο του αντικειμένου μελέτης με θέμα την αλληλεπίδραση ανθρώπου-μηχανής. Έχει ως σκοπό τη μελέτη, το σχεδιασμό και την υλοποίηση ενός τρόπου συνεργασίας ανθρώπων και Η/Υ ώστε να ικανοποιούνται οι ανάγκες των ανθρώπων με τον πιο αποτελεσματικό τρόπο. Οι σχεδιαστές διεπαφών χρήστη πρέπει να λάβουν υπόψη αρκετούς παράγοντες όπως το τι θέλουν και τι περιμένουν οι χρήστες, η αντίληψη των χρηστών πάνω στο αντικείμενο, τεχνικές δυσκολίες και περιορισμοί υλικού/λογισμικού κ.α. Η διεπαφή χρήστη είναι το τμήμα εκείνο ενός Η/Υ, και του λογισμικού του, που ο χρήστης μπορεί να δει, να ακούσει, και να αλληλεπιδράσει μαζί του. Τα πλέον βασικά χαρακτηριστικά μιας διεπαφής χρήστη είναι η είσοδος και η έξοδος δεδομένων. Η είσοδος δεδομένων εξυπηρετεί στο να διατυπώσει ο χρήστης στον Η/Υ τις απαιτήσεις του. Οι πιο διαδεδομένες συσκευές εισόδου δεδομένων είναι τα πληκτρολόγια, ποντίκια, trackball, οθόνες αφής και μικρόφωνα (για φωνητικές εντολές). Η έξοδος δεδομένων εξυπηρετεί στην μετάδοση των αποτελεσμάτων από τον Η/Υ στο χρήστη. Ο πιο διαδεδομένος τρόπος εξόδου δεδομένων σήμερα είναι οι οθόνες και τα ηχεία (Galitz, 2002).

7.2. Περιβάλλον Διεπαφής

Για την παρούσα διπλωματική εργασία αναπτύχθηκε μία διεπαφή χρήστη βασισμένη σε ιστοσελίδα (Web-based User Interface). Ο λόγος που επιλέχθηκε αυτός ο τύπος διεπαφής είναι η απλότητα σε συνδυασμό με την ευκολία καθώς δεν χρειάζεται να γίνει εγκατάσταση κάποιου προγράμματος, και η συμβατότητα καθώς με τη χρήση ενός προγράμματος περιήγησης στο διαδίκτυο (Web Browser), η διεπαφή μπορεί να τρέξει στις περισσότερες υπολογιστικές πλατφόρμες (Windows, Mac και Linux). Στην εικόνα 51 μπορούμε να δούμε το περιβάλλον της διεπαφής που αναπτύχθηκε:

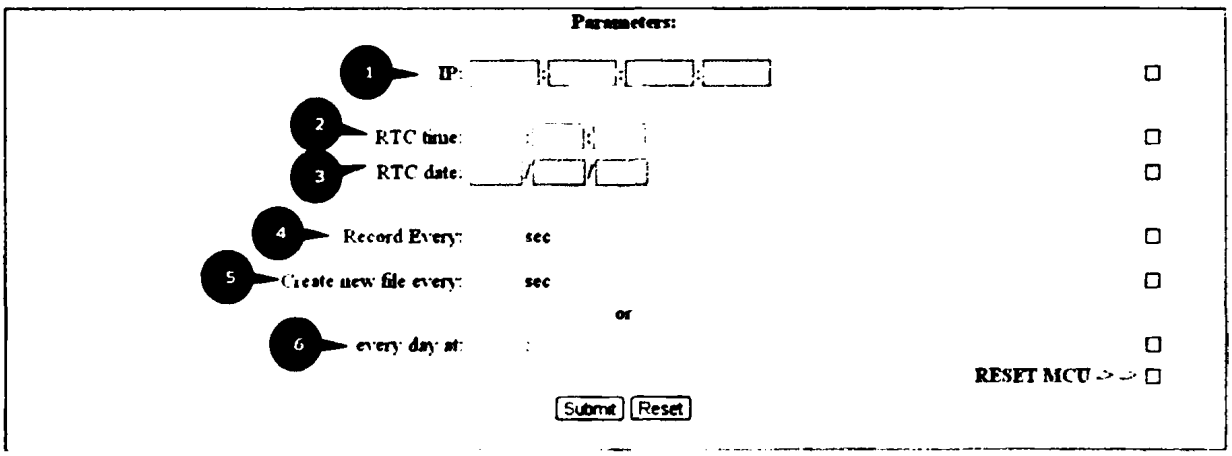


Εικόνα 51

- Web browser: Στην εικόνα 51, στο σημείο 1 βλέπουμε το παράθυρο ενός Web browser. Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε ο Google Chrome. Η διεπαφή επίσης δοκιμάστηκε για την ορθή λειτουργία της σε Internet Explorer και Mozilla Firefox.
- Καρτέλα του Browser: Στο σημείο 2 βλέπουμε μία καρτέλα του browser η οποία περιέχει το όνομα της σελίδας (WSN).
- Γραμμή διεύθυνσης: Στο σημείο 3 βλέπουμε τη γραμμή διεύθυνσης του browser στην οποία πληκτρολογούμε τη διεύθυνση που «ακούει» η διεπαφή ώστε να εμφανιστεί η σελίδα.
- Αρχεία: Στο σημείο 4 βλέπουμε το χώρο όπου εμφανίζονται τα αποθηκευμένα αρχεία στα οποία περιέχονται οι μετρήσεις από το ασύρματο δίκτυο αισθητήρων.
- Παραμετροποίηση: Στο σημείο 5 βλέπουμε το χώρο όπου μπορούμε να αλλάξουμε βασικές παραμέτρους του συστήματος
- Πλήθος κόμβων: Στο σημείο 6 βλέπουμε το πλήθος των κόμβων που λειτουργούν κάθε στιγμή στο σύστημα. Πατώντας το κουμπί «Scan» μπορούμε να σαρώσουμε το δίκτυο για τυχόν αλλαγές.
- Πληροφορίες Σελίδας: Στο σημείο 7 βλέπουμε πληροφορίες που αφορούν τη σελίδα όπως αριθμό προσπελάσεων (Hits) και ημερομηνία/ώρα τελευταίας ανανέωσης.

7.2.1. Παραμετροποίηση (τροποποίηση παραμέτρων λειτουργίας)

Όπως είδαμε, στο σημείο 5 της διεπαφής, μπορούν να αλλαχθούν κάποιες από τις βασικές παραμέτρους του συστήματος. Οι παράμετροι που μπορούν να αλλαχθούν φαίνονται στην εικόνα 52:

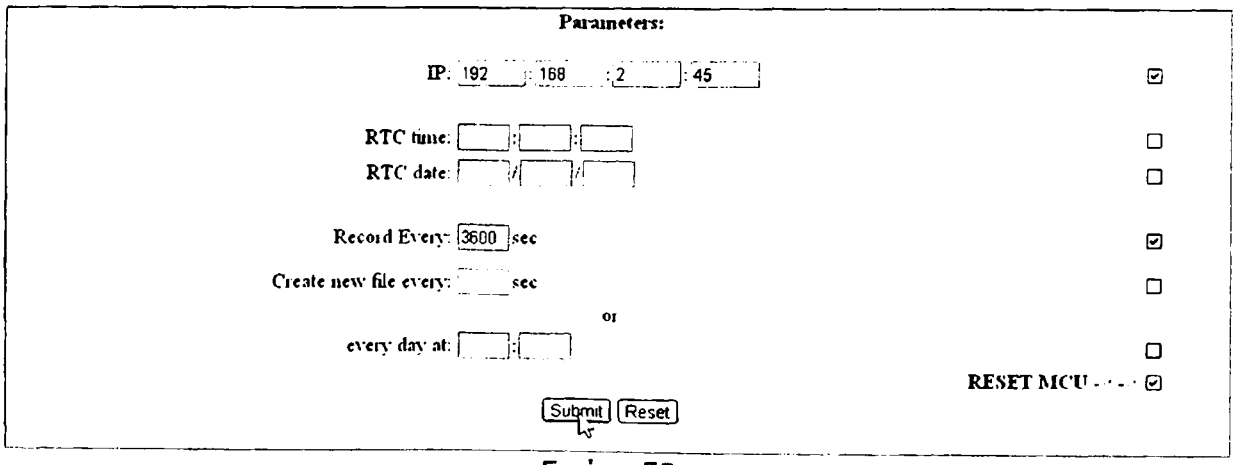


Εικόνα 52

- Διεύθυνση IP: Σε αυτό το πεδίο μπορεί να αλλάξει η διεύθυνση IP που χρησιμοποιεί το σύστημα για να συνδεθεί στο ενσύρματο δίκτυο. Η διεύθυνση που θα δηλωθεί θα πρέπει να είναι μέσα στο εύρος τιμών που ορίζει ο κατασκευαστής του Router ή του Switch. Για παράδειγμα εάν το εύρος διευθύνσεων ενός Router είναι 192.168.2.xxx και εμείς δώσουμε στο σύστημα 192.168.1.xxx τότε το σύστημα δεν θα καταφέρει να συνδεθεί με το δίκτυο. Η διεύθυνση που δηλώνεται κάθε φορά είναι στατική και δεν μπορεί να αλλάξει με δυναμικό τρόπο (π.χ. DHCP).

- Ώρα συστήματος: Σε αυτό το πεδίο μπορεί να αλλάξει η ώρα του συστήματος σε περίπτωση αλλαγής από θερινή σε χειμερινή ώρα και αντίστροφα ή σε περίπτωση αλλαγής της μπαταρίας του RTC ρολογιού.
- Ημερομηνία Συστήματος: Σε αυτό το πεδίο μπορεί να αλλάξει η ώρα του συστήματος σε περίπτωση αλλαγής της μπαταρίας του RTC ρολογιού.
- Διάστημα μεταξύ μετρήσεων: Σε αυτό το πεδίο μπορεί να αλλάξει το διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών μετρήσεων σε δευτερόλεπτα. Το εύρος τιμών που μπορεί να δεχθεί το συγκεκριμένο πεδίο είναι 0-65.535 (περίπου 0-18,2 ώρες).
- Δημιουργία νέου αρχείου μετρήσεων (Α' τρόπος): Σε αυτό το πεδίο μπορεί να αλλάξει το διάστημα που μεσολαβεί για τη δημιουργία ενός νέου αρχείου σε δευτερόλεπτα. Το εύρος τιμών που μπορεί να δεχθεί το συγκεκριμένο πεδίο είναι 0-65.535 (περίπου 0-18,2 ώρες).
- Δημιουργία νέου αρχείου μετρήσεων (Β' τρόπος): Σε αυτό το πεδίο μπορεί να αλλάξει η ώρα που θα δημιουργείται ένα νέο αρχείο, μία φορά την ημέρα. Μπορεί να χρησιμοποιείται μόνο ο ένας από τους δύο τρόπους κάθε φορά. Για παράδειγμα, Θα πρέπει να ρυθμιστεί η δημιουργία νέου αρχείου να πραγματοποιείται είτε κάθε 3600 sec (1 ώρα), είτε μία φορά την ημέρα στις 15.30. Υπάρχει και Γ' τρόπος δημιουργίας νέου αρχείου, τον οποίο θα δούμε στο υποκεφάλαιο 6.2.2, και δεν επηρεάζει τη λειτουργία των δύο παραπάνω τρόπων.

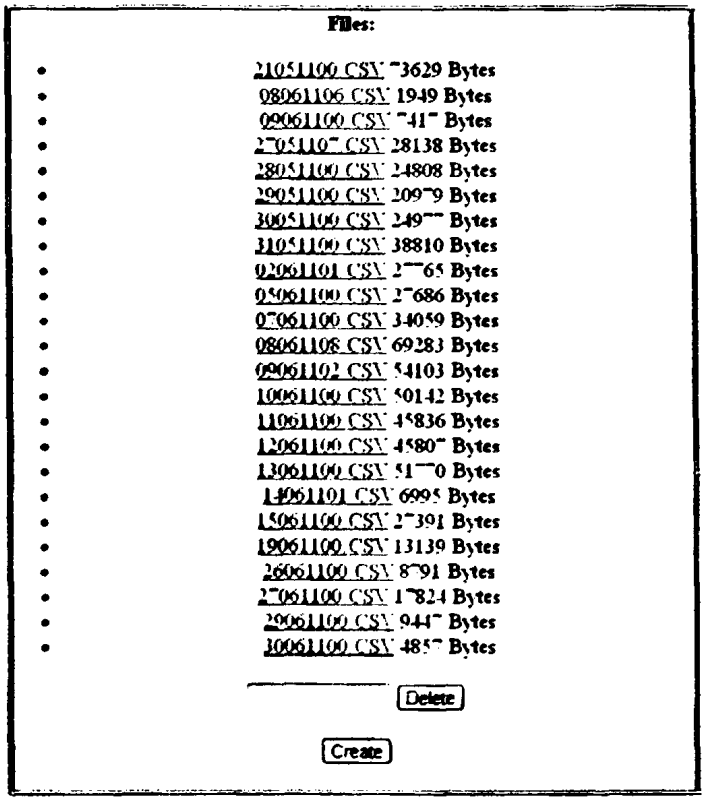
Για να αλλαχθεί μία παράμετρος θα πρέπει να συμπληρωθούν τα πεδία της με τις επιθυμητές τιμές, να ενεργοποιηθεί το checkbox δεξιά του πεδίου, καθώς και το checkbox «RESET MCU->->» ώστε να γίνει επανεκκίνηση του συστήματος και να ισχύσουν οι αλλαγές. Στη συνέχεια πρέπει να πατηθεί το κουμπί «Submit». Το κουμπί «Reset» καθαρίζει όλα τα πεδία της φόρμας. Στην εικόνα 53 μπορούμε να δούμε ένα παράδειγμα αλλαγής διεύθυνσης IP σε 192.168.2.45 και αλλαγής του διαστήματος μεταξύ δύο διαδοχικών μετρήσεων σε 3600 sec. Το checkbox «RESET MCU ->->» μπορεί να χρησιμοποιηθεί και ανεξάρτητα για να επανεκκινηθεί χειροκίνητα το σύστημα (πάντα με τη χρήση του κουμπιού «Submit»).



Εικόνα 53

7.2.2. Αρχεία

Οι μετρήσεις που πραγματοποιούνται από το δίκτυο αποθηκεύονται σε αρχεία τύπου .CSV μέσα στην κάρτα μνήμης microSD. Αρχεία τέτοιου τύπου διαχωρίζουν τις τιμές τους με το λατινικό κόμμα («;») ενώ είναι προσπελάσιμα από το Microsoft Excell και άλλα προγράμματα λογιστικών φύλλων για περαιτέρω επεξεργασία. Τα αρχεία που δημιουργούνται με οποιονδήποτε από τους τρεις τρόπους ακολουθούν το format 8.3. Αυτό σημαίνει ότι το όνομα του κάθε αρχείου μπορεί να περιλαμβάνει το πολύ 8 χαρακτήρες ακολουθουμένη από μία τελεία και την κατάληξη του τύπου αρχείου. Στην συγκεκριμένη εφαρμογή, οι 6 πρώτοι χαρακτήρες αφιερώνονται στην ημερομηνία και οι χαρακτήρες 7 και 8 σε έναν αύξοντα αριθμό. Για παράδειγμα εάν η τρέχουσα ημερομηνία είναι 25/5/2011 τότε το πρώτο αρχείο που θα δημιουργηθεί εκείνη την ημέρα θα ονομάζεται 25051100.CSV ενώ το δεύτερο 25051101.CSV κλπ. Πρακτικά, στη διάρκεια μίας ημέρας μπορούν να δημιουργηθούν έως και 100 αρχεία (0-99) χωρίς να δημιουργηθεί αστάθεια στο σύστημα. Στην εικόνα 54 βλέπουμε πως προβάλλονται τα αποθηκευμένα αρχεία

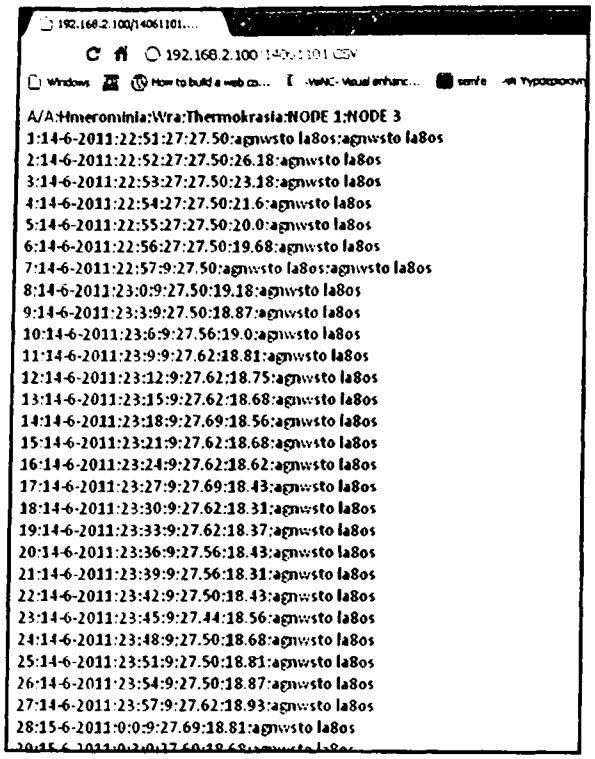


Εικόνα 54

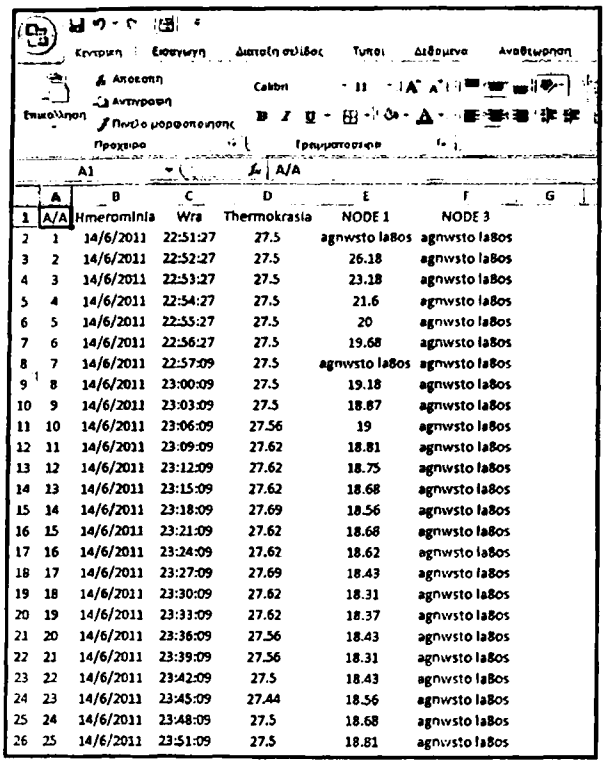
Σε αυτό το τμήμα του interface δίνεται η δυνατότητα δημιουργίας (Γ' τρόπος), ανάγνωσης, αποθήκευσης και διαγραφής των αρχείων. Ο Γ' τρόπος δημιουργίας ενός αρχείου αφορά την άμεση δημιουργία του χωρίς χρονοπρογραμματισμό. Για να δημιουργήσουμε άμεσα ένα νέο αρχείο αρκεί να πατήσουμε το κουμπί «Create».

Όπως φαίνεται και από την εικόνα 54 τα αρχεία εμφανίζονται με τη μορφή υπερσυνδέσμου (hyperlink). Με ένα πάτημα επάνω στο αρχείο που μας ενδιαφέρει ο browser το προβάλει. Επίσης μπορούμε να το προβάλουμε σε νέα καρτέλα του browser πατώντας δεξί κλικ στο αρχείο που μας ενδιαφέρει και ύστερα πατώντας στην επιλογή «άνοιγμα συνδέσμου σε νέα καρτέλα».

Με την ίδια διαδικασία, κάνοντας δηλαδή δεξί κλικ στο αρχείο και πατώντας στην επιλογή «αποθήκευση συνδέσμου ως...» μπορούμε να κατεβάσουμε το αρχείο στον υπολογιστή μας για περαιτέρω επεξεργασία.



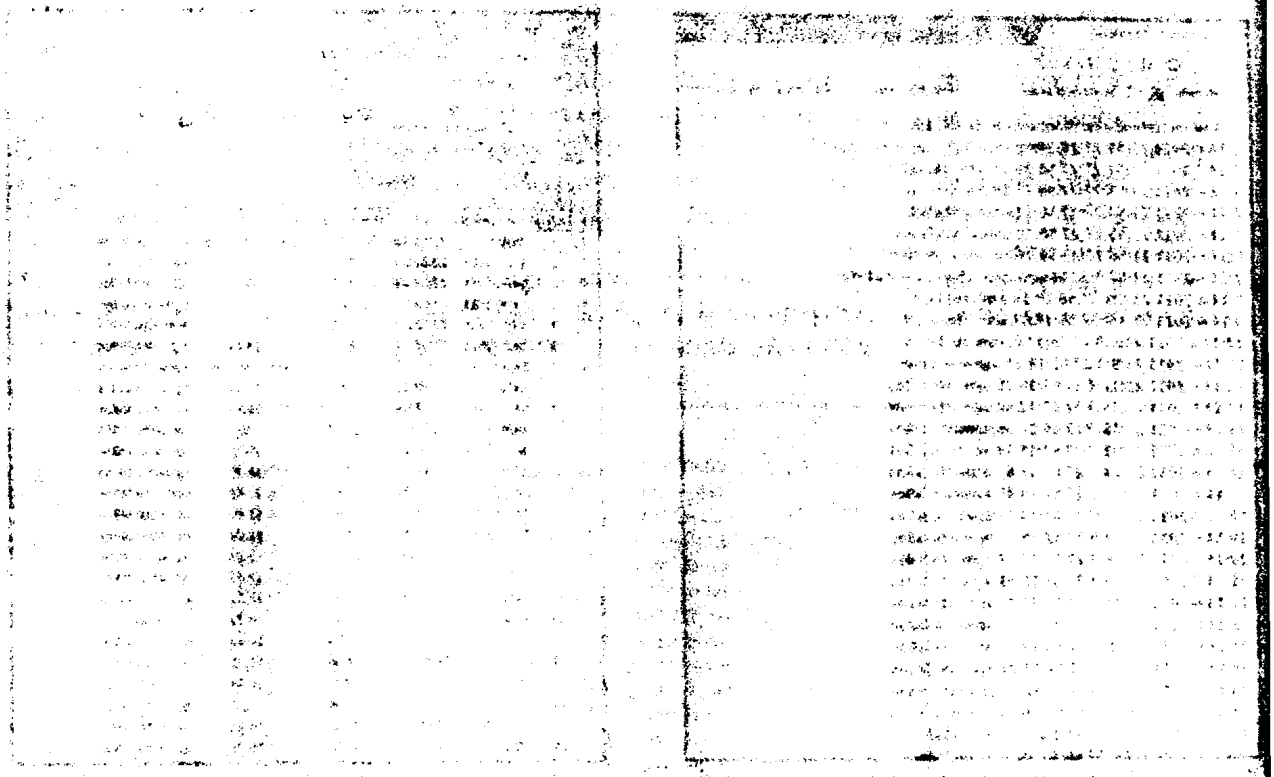
A/A:Hmeromlnia:Wra:Thermokrasia:NODE 1:NODE 3
1:14-6-2011:22:51:27:27.50:agnwsto la8os:agnwsto la8os
2:14-6-2011:22:52:27:27.50:26.18:agnwsto la8os
3:14-6-2011:22:53:27:27.50:23.18:agnwsto la8os
4:14-6-2011:22:54:27:27.50:21.6:agnwsto la8os
5:14-6-2011:22:55:27:27.50:20.0:agnwsto la8os
6:14-6-2011:22:56:27:27.50:19.68:agnwsto la8os
7:14-6-2011:22:57:9:27.50:agnwsto la8os:agnwsto la8os
8:14-6-2011:23:0:9:27.50:19.18:agnwsto la8os
9:14-6-2011:23:3:9:27.50:18.87:agnwsto la8os
10:14-6-2011:23:6:9:27.56:19.0:agnwsto la8os
11:14-6-2011:23:9:9:27.62:18.81:agnwsto la8os
12:14-6-2011:23:12:9:27.62:18.75:agnwsto la8os
13:14-6-2011:23:15:9:27.62:18.68:agnwsto la8os
14:14-6-2011:23:18:9:27.69:18.56:agnwsto la8os
15:14-6-2011:23:21:9:27.62:18.68:agnwsto la8os
16:14-6-2011:23:24:9:27.62:18.62:agnwsto la8os
17:14-6-2011:23:27:9:27.69:18.43:agnwsto la8os
18:14-6-2011:23:30:9:27.62:18.31:agnwsto la8os
19:14-6-2011:23:33:9:27.62:18.37:agnwsto la8os
20:14-6-2011:23:36:9:27.56:18.43:agnwsto la8os
21:14-6-2011:23:39:9:27.56:18.31:agnwsto la8os
22:14-6-2011:23:42:9:27.50:18.43:agnwsto la8os
23:14-6-2011:23:45:9:27.44:18.56:agnwsto la8os
24:14-6-2011:23:48:9:27.50:18.68:agnwsto la8os
25:14-6-2011:23:51:9:27.50:18.81:agnwsto la8os
26:14-6-2011:23:54:9:27.50:18.87:agnwsto la8os
27:14-6-2011:23:57:9:27.62:18.93:agnwsto la8os
28:15-6-2011:0:0:9:27.69:18.81:agnwsto la8os



	A	B	C	D	E	F	G
1	A/A	Hmeromlnia	Wra	Thermokrasia	NODE 1	NODE 3	
2	1	14/6/2011	22:51:27	27.5	agnwsto la8os	agnwsto la8os	
3	2	14/6/2011	22:52:27	27.5	26.18	agnwsto la8os	
4	3	14/6/2011	22:53:27	27.5	23.18	agnwsto la8os	
5	4	14/6/2011	22:54:27	27.5	21.6	agnwsto la8os	
6	5	14/6/2011	22:55:27	27.5	20	agnwsto la8os	
7	6	14/6/2011	22:56:27	27.5	19.68	agnwsto la8os	
8	7	14/6/2011	22:57:09	27.5	agnwsto la8os	agnwsto la8os	
9	8	14/6/2011	23:00:09	27.5	19.18	agnwsto la8os	
10	9	14/6/2011	23:03:09	27.5	18.87	agnwsto la8os	
11	10	14/6/2011	23:06:09	27.56	19	agnwsto la8os	
12	11	14/6/2011	23:09:09	27.62	18.81	agnwsto la8os	
13	12	14/6/2011	23:12:09	27.62	18.75	agnwsto la8os	
14	13	14/6/2011	23:15:09	27.62	18.68	agnwsto la8os	
15	14	14/6/2011	23:18:09	27.69	18.56	agnwsto la8os	
16	15	14/6/2011	23:21:09	27.62	18.68	agnwsto la8os	
17	16	14/6/2011	23:24:09	27.62	18.62	agnwsto la8os	
18	17	14/6/2011	23:27:09	27.69	18.43	agnwsto la8os	
19	18	14/6/2011	23:30:09	27.62	18.31	agnwsto la8os	
20	19	14/6/2011	23:33:09	27.62	18.37	agnwsto la8os	
21	20	14/6/2011	23:36:09	27.56	18.43	agnwsto la8os	
22	21	14/6/2011	23:39:09	27.56	18.31	agnwsto la8os	
23	22	14/6/2011	23:42:09	27.5	18.43	agnwsto la8os	
24	23	14/6/2011	23:45:09	27.44	18.56	agnwsto la8os	
25	24	14/6/2011	23:48:09	27.5	18.68	agnwsto la8os	
26	25	14/6/2011	23:51:09	27.5	18.81	agnwsto la8os	

Εικόνα 55: Αρχείο με μετρήσεις όπως φαίνεται στο browser Google Chrome (αριστερά) και όπως φαίνεται στο Microsoft Office Excell.

Για να διαγράψουμε ένα αρχείο θα πρέπει να πληκτρολογήσουμε το όνομά του μαζί με την κατάληξη του στη φόρμα αριστερά του κουμπιού «Delete» και στη συνέχεια να πατήσουμε το κουμπί «Delete».



8. Συμπεράσματα-Συζήτηση

Ο σκοπός της παρούσας διπλωματικής εργασίας ήταν η μελέτη των ασύρματων δικτύων αισθητήρων, η σύγκριση μεταξύ των δημοφιλέστερων προτύπων δικτύωσης που χρησιμοποιούνται σήμερα καθώς και η υλοποίηση ενός ασύρματου δικτύου αισθητήρων.

Ο σκοπός της εργασίας επετεύχθη. Αρχικά έγινε σύγκριση των προτύπων δικτύωσης και έπειτα από επιλογή του κατάλληλου προτύπου υλοποιήθηκε ένα ασύρματο δίκτυο αισθητήρων σε επίπεδο υλικού αλλά και λογισμικού το οποίο είναι ικανό να καλύψει ένα ευρύ φάσμα εφαρμογών. Αφού τέθηκαν οι προδιαγραφές που θα πρέπει να πληρεί και τα χαρακτηριστικά του συστήματος, υλοποιήθηκε το απαιτούμενο λογισμικό με τη βοήθεια αναπτυξιακού περιβάλλοντος. Ύστερα από επανειλημμένες δοκιμές και αποσφαλμάτωση του κώδικα, σχεδιάστηκαν τα κατάλληλα κυκλώματα για τις συσκευές στην τελική τους μορφή. Ακολούθησε η κατασκευή των κυκλωμάτων και ο προγραμματισμός των μικροελεγκτών.

Συμπερασματικά αναφέρεται ότι οι δοκιμές του συστήματος που ακολούθησαν ήταν επιτυχείς και απέδειξαν ότι λειτουργεί όπως αναμενόταν. Κατά τη διάρκεια της λειτουργίας του συστήματος παρατηρήθηκε ότι ο κύριος κόμβος καταναλώνει 3,82 W ενώ ο κάθε περιφερειακός κόμβος καταναλώνει σε κατάσταση αναμονής 0,099 W και 0,8 W κατά την αποστολή δεδομένων. Επίσης έγινε δοκιμή του κάθε περιφερειακού κόμβου υπό βροχή και αποδείχθηκε ότι είναι υδατοστεγής.

8.1. Πλεονεκτήματα

Συγκριτικά με άλλες εργασίες που έχουν γίνει πάνω στο αντικείμενο και εμπορικά διαθέσιμα ασύρματα δίκτυα αισθητήρων, η κατασκευή που πραγματοποιήθηκε υπερτερεί στα εξής:

- Ο κάθε περιφερειακός κόμβος είναι έτοιμος για χρήση σε πραγματικές συνθήκες και έντονα καιρικά φαινόμενα.
- Ο κάθε περιφερειακός κόμβος είναι συμβατός με πληθώρα ανιχνευτών.
- Το προγραμματιστικό περιβάλλον με το οποίο αναπτύχθηκε το λογισμικό του συστήματος είναι ανοιχτού κώδικα.
- Η διεπαφή χρήστη (User Interface) είναι βασισμένη σε ιστοσελίδα, γεγονός που εξαλείφει την ανάγκη εγκατάστασης κάποιου προγράμματος.

8.2. Μειονεκτήματα-Προβλήματα

Στα μειονεκτήματα συγκαταλέγονται τα εξής:

- Υποστήριξη έως 10 περιφερειακών κόμβων. Αν και το πρότυπο υποστηρίζει έως και 65.000 κόμβους συνδεδεμένους στο ίδιο δίκτυο, το δίκτυο που υλοποιήθηκε έχει προγραμματιστεί να υποστηρίζει έως 10 λόγω μικρής χωρητικότητας της μνήμης RAM του μικροελεγκτή. Το πρόβλημα μπορεί να λυθεί με την προσθήκη επιπλέον μνήμης RAM.
- Αδυναμία δημιουργίας αρχείου για την αποθήκευση των μετρήσεων κατά το χρονικό διάστημα μεταξύ ώρας 00.00 και 00.03 σε καθημερινή βάση. Το σύστημα

το δεδομένο χρονικό διάστημα δουλεύει σωστά, ανταποκρίνεται και παίρνει μετρήσεις, όμως αδυνατεί να δημιουργήσει ένα καινούργιο αρχείο. Το πρόβλημα εντοπίζεται στη βιβλιοθήκη του αναπτυξιακού περιβάλλοντος που διαχειρίζεται τις χρονικές διακοπές (time interrupts).

8.3. Μελλοντική Εξέλιξη

Παρόλο που οι συσκευές οι οποίες κατασκευάστηκαν έχουν ικανοποιητική απόδοση και καλύπτουν τους σκοπούς της εργασίας, υπάρχουν τομείς στους οποίους θα μπορούσε να υπάρξει κάποια εξέλιξη.

- Προσθήκη ενός δέκτη GPS σε κάθε περιφερειακό κόμβο (στη θέση και στο βύσμα που έχει προβλεφθεί και ήδη υπάρχει). Παράλληλα θα πρέπει να τροποποιηθεί το λογισμικό του κύριου κόμβου ώστε να προβάλλει με τη βοήθεια της υπηρεσίας Google maps τη θέση του κάθε περιφερειακού κόμβου.
- Τροποποίηση του λογισμικού στον κύριο κόμβο και στους περιφερειακούς για την καλύτερη ενεργειακή διαχείριση των περιφερειακών κόμβων. Στην παρούσα κατάσταση, οι ασύρματοι πομποδέκτες είναι ενεργοί για 10 δευτερόλεπτα στο διάστημα μεταξύ δύο διαδοχικών μετρήσεων. Με τη χρήση του νέου υλικολογισμικού (firmware) των ασύρματων πομποδεκτών (806x έναντι του 804x που χρησιμοποιείται) και τον κατάλληλο προγραμματισμό, ο χρόνος αυτός μπορεί να μειωθεί σε μερικά δέκατα του δευτερολέπτου.
- Προβολή γραφημάτων των μετρήσεων μέσω της διεπαφής χρήστη, είτε με την τροποποίηση του λογισμικού και την προσθήκη εντολών JavaScript, είτε με τη βοήθεια της υπηρεσίας Google charts.
- Προσθήκη ασύρματου πομποδέκτη GSM/GPRS για την ενημέρωση του χρήστη σε περίπτωση κάποιου σημαντικού γεγονότος. Π.χ. ξέσπασμα φωτιάς σε περίπτωση που η εφαρμογή χρησιμοποιείται για πυρανίχνευση.

Βιβλιογραφία

Ammari Habib M. Challenges and Opportunities of Connected k-CoveredWireless Sensor Networks [Βιβλίο]. - Warsaw : Springer, 2009.

Amol B. Bakshi Viktor K. Prasanna ARCHITECTURE-INDEPENDENT PROGRAMMING FORWIRELESS SENSOR NETWORKS [Βιβλίο]. - Hoboken NJ : JOHN WILEY & SONS, 2008.

Arduino Main Site [Ηλεκτρονικό]. - 2011. - 25 2 2011. - http://www.arduino.cc/.

Atmel Corporation ATmega328P 8-bit Atmel Microcontroller with 32K Bytes In-System Programmable Flash datasheet [Ηλεκτρονικό]. - 5 2011. - Rev. D. - 6 6 2011. - http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf.

Atmel Corporation ATmega640/1280/1281/2560/2561 Preliminary [Ηλεκτρονικό]. - 8 2007. - 12 2010. - http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf.

Atmel Corporation AVR151: Setup And Use of The SPI [Ηλεκτρονικό]. - 7 2008. - Rev.2585C. - 10 5 2011. - http://atmel.com/dyn/resources/prod_documents/doc2585.pdf.

Banzi Massimo Getting Started with Arduino [Βιβλίο]. - [s.l.] : O'Reilly, 2008.

Bourns PTC Resettable Fuse datasheet [Ηλεκτρονικό]. - 12 2010. - 15 6 2011. - http://www.bourns.com/data/global/pdfs/mfmsmf.pdf.

Boyer Etienne The Secrets of I2C [Άρθρο] // Elektor Electronics. - Brentford : [s.n.], 2008. - 375.

Buratti Chiara [και συν.] Sensor Networks with IEEE 802.15.4 Systems. Distributed Processing, MAC, and Connectivity [Βιβλίο]. - Berlin : Springer, 2011.

Creative Commons [Ηλεκτρονικό]. - 2011. - http://creativecommons.org/licenses/by-sa/3.0/.

David Gascón 802.15.4 vs ZigBee [Ηλεκτρονικό]. - Libelium, 28 4 2009. - 10 3 2011. - http://www.libelium.com/development/articles/091811815000.

Digi International Inc. Product Manual: XBee / XBee-PRO DigiMesh 2.4 OEM RF Modules [Ηλεκτρονικό]. - 2011a. - 17 2 2011. - http://ftp1.digi.com/support/documentation/90000991_C.pdf.

Digi International Inc. Users Guide: XCTU Configuration & Test Utility Software [Ηλεκτρονικό]. - 2011. - 5 1 2011. - http://ftp1.digi.com/support/documentation/90001003_A.pdf.

Digi International Inc. Wireless Mesh Networking ZigBee® vs. DigiMesh™ [Ηλεκτρονικό]. - 2008. - 9 2 2011. - http://www.digi.com/pdf/wp_zigbeevsdigimesh.pdf.

Digi International The DigiMesh™ Networking Protocol [Ηλεκτρονικό]. - 23 9 2008. - 10 3 2011. - http://www.digi.com/technology/digimesh/.

Eady Fred Hands-On ZigBee: Implementing 802.15.4 with Microcontrollers [Βιβλίο]. - [s.l.] : Newnes, 2007.

Fairchild Semiconductor KA78TXX 3-Terminal 3A Positive Voltage Regulator [Ηλεκτρονικό]. - 2002. - Rev. 1.0.1. - 13 9 2011. - http://www.fairchildsemi.com/ds/KA/KA78T05.pdf.

Faludi Robert Building Wireless Sensor Networks [Βιβλίο]. - Sebastopol : O'Reilly, 2011.

Farahani Shahin ZigBee Wireless Networks and Transceivers [Βιβλίο]. - [s.l.] : Newnes, 2008.

Freescale Semiconductor MCF51AC256 ColdFire® Integrated Microcontroller Reference Manual [Ηλεκτρονικό]. - 9 2010. - Rev.5. - 10 5 2011. - http://cache.freescale.com/files/32bit/doc/ref_manual/MCF51AC256RM.pdf.

FTDI FT232R USB UART IC datasheet [Ηλεκτρονικό]. - 4 2011b. - Ver. 2.0.9. - 21 6 2011. - http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.

FTDI UM232R USB - Serial UART Development Module datasheet [Ηλεκτρονικό]. - 23 5 2011a. - Ver. 1.0.5. - 21 6 2011. - http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UM232R.pdf.

Galitz Wilbert O. The Essential Guide to User Interface Design [Βιβλίο]. - [s.l.] : John Wiley & Sons, Inc., 2002.

Gavrilovska Liljana [και συν.] Application and Multidisciplinary Aspects of Wireless Sensor Networks. Concepts, Integration, and Case Studies [Βιβλίο]. - London : Springer, 2011.

Gislason Drew Zigbee Wireless Networking [Βιβλίο]. - [s.l.] : Newnes, 2008.

H. Labiod H. Afifi, C. De Santis WI-FI BLUETOOTH ZIGBEE and WIMAX [Βιβλίο]. - Dordrecht : Springer, 2007.

Hantronix 16 Character x 2 Lines, Very Small Size, LED Backlight HDM16216L-S [Ηλεκτρονικό]. - 2010. - 21 7 2011. - http://www.hantronix.com/down/16216ls.pdf.

IEEE Standards Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs) [Άρθρο]. - New York : The Institute of Electrical and Electronics Engineers (IEEE), Inc., 2003.

Ilyas Mohammad και Mahgoub Imad Handbook of Sensor Networks:Compact Wireless and Wired Sensing Systems [Βιβλίο]. - Boca Raton : CRC Press, 2005.

Intersil SPI Protocol and Bus Configuration of Multiple DCPs [Ηλεκτρονικό]. - 20 8 2007. - 10 5 2011. - http://www.intersil.com/data/an/an1340.pdf.

Kuorilehto et al. Mauri Ultra-Low Energy Wireless Sensor Networks in Practice:Theory, Realization and Deployment [Βιβλίο]. - West Sussex : John Wiley & Sons, Ltd, 2007. - ISBN 978-0-470-05786-5.

Linear Technology Corporation LTC2400 24-Bit mPower No Latency DSTM ADC in SO-8 datasheet [Ηλεκτρονικό]. - 1998. - Rev A. - 28 12 2010. - http://cds.linear.com/docs/Datasheet/2400fa.pdf.

Mainwaring Alan [και συν.] Wireless Sensor Networks for Habitat Monitoring [Βιβλίο]. - Atlanta : 1st ACM International Workshop, 2002.

Margolis Michael Arduino Cookbook [Βιβλίο]. - [s.l.] : O'Reilly, 2011.

Maxim Integrated Products, Inc. DS1307 64 x 8 Serial, I2C Real-Time Clock datasheet [Ηλεκτρονικό]. - 16 10 2008. - Rev. 5. - 20 1 2011. - http://datasheets.maxim-ic.com/en/ds/DS1307.pdf.

Maxim Integrated Products, Inc. DS1834/A/D Dual EconoReset with Pushbutton datasheet [Ηλεκτρονικό]. - 17 7 2000. - Rev. 1. - 8 1 2011. - http://datasheets.maxim-ic.com/en/ds/DS1834-DS1834D.pdf.

Maxim Integrated Products, Inc. DS2782 Stand-Alone Fuel Gauge IC datasheet [Ηλεκτρονικό]. - 5 2009. - Rev. 4. - 25 10 2010. - http://datasheets.maxim-ic.com/en/ds/DS2782.pdf.

Maxim Integrated Products, Inc. MAX1551/MAX1555 Dual-Input USB/AC Adapter 1-Cell Li+ Battery Chargers datasheet [Ηλεκτρονικό]. - 7 2003. - Rev. 0. - 25 10 2010. - http://datasheets.maxim-ic.com/en/ds/MAX1551-MAX1555.pdf.

Maxim Integrated Products, Inc. MAX5392 Dual, 256-Tap, Volatile, Low-Voltage, Linear Taper Digital Potentiometer [Ηλεκτρονικό]. - 11 2010. - Rev.2. - 25 7 2011. - http://datasheets.maxim-ic.com/en/ds/MAX5392.pdf.

Maxim Integrated Products, Inc. MAX6325/MAX6341/MAX6350 1ppm/°C, Low-Noise, +2.5V/+4.096V/+5V Voltage References datasheet [Ηλεκτρονικό]. - 1 2001. - Rev. 1. - 5 9 2011. - http://datasheets.maxim-ic.com/en/ds/MAX6325-MAX6350.pdf.

Maxim Integrated Products, Inc. MAX6369–MAX6374 Pin-Selectable Watchdog Timers datasheet [Ηλεκτρονικό]. - 1 2011. - Rev. 5. - 18 4 2011. - http://datasheets.maxim-ic.com/en/ds/MAX6369-MAX6374.pdf.

Maxim Integrated Products, Inc. MAX6665 Fan Controller -Driver with Factory-Programmed Temperature Thresholds datasheet [Ηλεκτρονικό]. - 5 2005. - Rev 0. - 5 9 2011. - http://datasheets.maxim-ic.com/en/ds/MAX6665.pdf.

McRoberts Michael Beginning Arduino [Βιβλίο]. - [s.l.] : Apress, 2010.

Micrel MIC5219 500mA-Peak Output LDO Regulator datasheet [Ηλεκτρονικό]. - 18 6 2009b. - 2 2 2011. - http://www.micrel.com/_PDF/mic5219.pdf.

Microchip Corporation 24AA1025/24LC1025/24FC1025, 1024K I2C™ CMOS Serial EEPROM [Ηλεκτρονικό]. - 1 2011. - Revision H. - 23 5 2011. - http://ww1.microchip.com/downloads/en/DeviceDoc/21941J.pdf.

Microchip Corporation MCP4725 12-Bit Digital-to-Analog Converter with EEPROM Memory [Ηλεκτρονικό]. - 6 2009. - Rev. D. - 8 1 2011. - http://ww1.microchip.com/downloads/en/DeviceDoc/22039d.pdf.

Noble Joshua Programming Interactivity [Βιβλίο]. - Sebastopol : O'Reilly, 2009.

Pavel Ripka Alois Tipek Modern Sensor Handbook [Βιβλίο]. - ISTE : London, 2007.

Philips Semiconductors THE I 2C-BUS SPECIFICATION [Ηλεκτρονικό]. - 1 2000. - VERSION 2.1. - 10 5 2011. - http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf.

ST Microelectronics LD1117 SERIES LOW DROP FIXED AND ADJUSTABLE POSITIVE VOLTAGE REGULATORS [Ηλεκτρονικό]. - 12 2005. - Rev.19. - 13 9 2011. - http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00000544.pdf.

Texas Instruments OPA2350 High-Speed, Single-Supply, Rail-to-Rail OPERATIONAL AMPLIFIERS [Ηλεκτρονικό]. - 1 2005. - 25 7 2011. - http://www.ti.com/lit/ds/sbos099c/sbos099c.pdf.

Texas Instruments PCF8574 REMOTE 8-BIT I/O EXPANDER FOR I2C BUS [Ηλεκτρονικό]. - 6 2001. - 12 8 2011. - http://www.ti.com/lit/ds/symlink/pcf8574.pdf.

Texas Instruments SN74LVC1G04 SINGLE INVERTER GATE datasheet [Ηλεκτρονικό]. - 30 6 2011. - Rev. X. - 11 8 2011. - http://www.ti.com/lit/gpn/sn74lvc1g04.

Texas Instruments TLC556, TLC556Y DUAL LinCMOS TIMERS datasheet [Ηλεκτρονικό]. - 25 9 1997. - Rev. B. - 2 12 2010. - http://www.ti.com/lit/gpn/tlc556.

Texas Instruments TPS60110 Regulated 5V 300mA Low-Noise Charge Pump DC/DC Converter datasheet [Ηλεκτρονικό]. - 7 8 2008. - Rev. C. - 2 7 2011. - http://www.ti.com/lit/gpn/tps60110.

USGlobalSat Inc. EM-408 GPS Engine Board User Guide [Ηλεκτρονικό]. - 2011. - Ver. 1.4.1. - 25 5 2011. - http://www.usglobalsat.com/store/download/47/em408_ug.pdf.

Vishay DF06SA Miniature Glass Passivated Single-Phase Bridge Rectifier datasheet [Ηλεκτρονικό]. - 11 3 2011. - 1 4 2011. - http://www.vishay.com/docs/88573/dfs.pdf.

Wiznet W5100 Datasheet [Ηλεκτρονικό]. - 16 2 2011. - Ver. 1.2.3. - 28 6 2011. - http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/W5100_Datasheet_v1.2.3[0].pdf.

Wiznet WIZ812MJ Datasheet [Ηλεκτρονικό]. - 28 1 2009. - Rev. 1.1. - 2 11 2010. - http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/WIZ812MJ_Datasheet_V_1.1[1].pdf.

Yick Jennifer, Mukherjee Biswanath και Ghosal Dipak Wireless sensor network survey [Βιβλίο] / επιμ. Ekici E.. - Davis, California : Elsevier, 2008.

Young Joel K. What a Mesh! Part 2-Networking Architectures and Protocols [Ηλεκτρονικό]. - Sensors Magazine, 1 12 2008. - 10 12 2010. - http://www.sensorsmag.com/networking-communications/wireless-sensor/what-a-mesh-part-2-networking-architectures-and-protocols-1544.

Zheng Jun και Jamalipour Abbas WIRELESS SENSOR NETWORKS, A Networking Perspective [Βιβλίο]. - Hoboken NJ : John Wiley & Sons, 2009.

ZigBee Alliance ZigBee Specification [Ηλεκτρονικό]. - 17 1 2008. - 4 12 2010. - http://www.zigbee.org/zigbee/en/spec_download/spec_download.asp?AccessCode=1528055807.

Καλόμοιρος Ι., Μπουλταδάκης Σ. και Πεταλάς Ι. Έλεγχος Κυκλωμάτων και Μετρήσεων με Η/Υ [Βιβλίο]. - Θεσσαλονίκη : Τζιόλας, 2002.

Κριτσωτάκη Αγγελική Ανάπτυξη Ασύρματου Δικτύου Προσωπικής Εμβέλειας για Εφαρμογές Τηλεϊατρικής [Βιβλίο]. - Αθήνα : Ε.Μ.Π.-Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, 2010.

Λιακάκος Δημήτριος Συλλογή και Συνάθροιση Δεδομένων σε Ασύρματα Δίκτυα Αισθητήρων με χρήση Ενεργειακά Αποδοτικού Πρωτοκόλλου σε Επίπεδο Πρόσβασης Μέσου [Βιβλίο]. - Αθήνα : Εθνικό Μετσόβιο Πολυτεχνείο (Διπλωματική Εργασία), 2009.

Παπαγεωργακοπούλου Ελένη-Αντιγόνη Έλεγχος Συνδεσιμότητας Ασύρματου Δικτύου Αισθητήρων [Βιβλίο]. - Πάτρα : Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών, 2009.

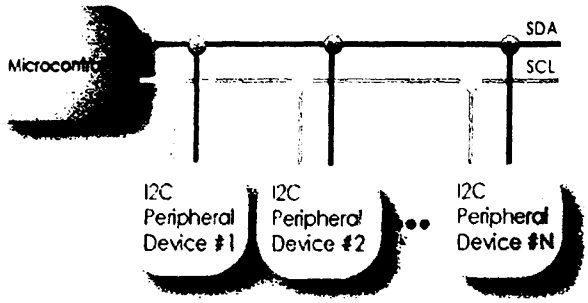
Τζόκας Δημήτριος και Φρόνιμος Θεόδωρος Ασύρματα Δίκτυα Αισθητήρων για παρακολούθηση περιβαλλοντικών μεγεθών [Βιβλίο]. - Θεσσαλονίκη : Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, , 2008.

Χαρτουμπέκης Γεώργιος Ασύρματα Δίκτυα Αισθητήρων:Ανάπτυξη Γραφικού Περιβάλλοντος με το Matlab για τη λήψη Μετρήσεων με χρήση του MTS400/420 board της Crossbow [Βιβλίο]. - Πάτρα : Πανπιστήμιο Πατρών-Τμήμα Φυσικής, 2010.

ΠΑΡΑΡΤΗΜΑ Α: Δίαυλοι επικοινωνίας

Π.Α.1 Ο Δίαυλος I^2C

Ο Δίαυλος I^2C (Inter Integrated Circuit) προτάθηκε από την Phillips την εποχή που πρωτοστατούσε στις εξελίξεις των συστημάτων ήχου. Τα κύρια πεδία εφαρμογών που είχε κλιθεί να καλύψει τότε (αρχές δεκαετίας του '80), αφορούσαν τους οικιακούς αυτοματισμούς και τις οικιακές καταναλωτικές συσκευές. Την εποχή εκείνη είχαν αρχίσει να εμφανίζονται οι πρώτοι μικροελεγκτές στις τηλεοράσεις και έπρεπε απαραιτήτως να βρεθεί μια λύση για το πώς θα «μιλούσαν» με τα διάφορα υποσυστήματα. Ο Δίαυλος I^2C είναι στην πραγματικότητα ένας σύγχρονος σειριακός δίαυλος υλοποιούμενος με τη βοήθεια τριών αγωγών: τον Serial Data (SDA), τον Serial Clock (SCL) και GND (Γείωση) (Boyer, 2008).



Εικόνα Π.Α.-1

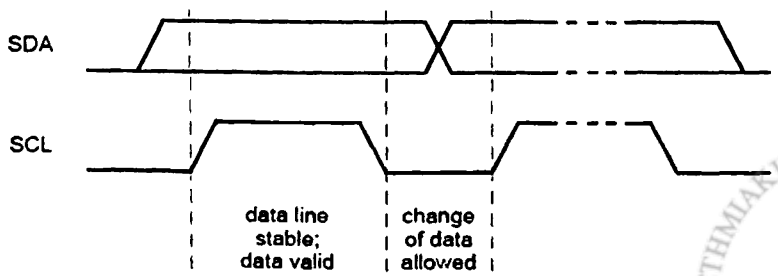
Η «φιλοσοφία» γύρω από τον δίαυλο I^2C είναι απλή. Η Κύρια Συσκευή, ή αλλιώς Master (περιφερειακό που έχει τον έλεγχο του διαύλου) εκπέμπει μηνύματα προς την εξαρτώμενη συσκευή ή αλλιώς Slave (περιφερειακό που λαμβάνει και επιβεβαιώνει τη λήψη μέσω του ομώνυμου σήματος). Αυτό το φαινομενικά «απλοϊκό» πρωτόκολλο δεν είναι μειωμένων δυνατοτήτων, κάθε άλλο. Με τη βοήθειά του μπορούν να επικοινωνήσουν πολλές συσκευές μεταξύ τους, αρκεί βέβαια να πληρούνται ορισμένοι κανόνες και προϋποθέσεις.

Οι τέσσερις σημαντικότερες καταστάσεις στις οποίες μπορεί να βρίσκεται ανά πάσα στιγμή ο δίαυλος είναι οι εξής:

Π.Α.1.1 Μεταφορά ενός ψηφίου πάνω στο δίαυλο I^2C

Το σήμα χρονισμού δεν «μοιάζει» πολύ με τα σήματα χρονισμού όπως τα ξέραμε ως τώρα. Αιτία γι' αυτό αποτελεί ο μεταβλητός λόγος διάρκειας προς περίοδο, ο οποίος μπορεί να παίρνει τιμές μέσα σε πολύ μεγάλα περιθώρια (Boyer, 2008).

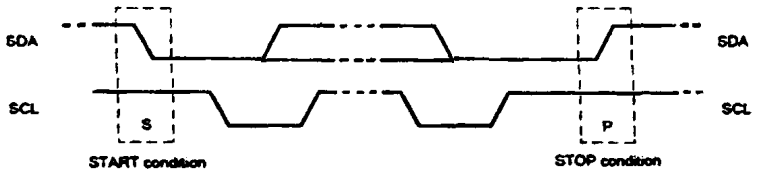
Εικόνα Π.Α.-2



Π.Α.1.2 Ακολουθίες Έναρξης και Λήξης

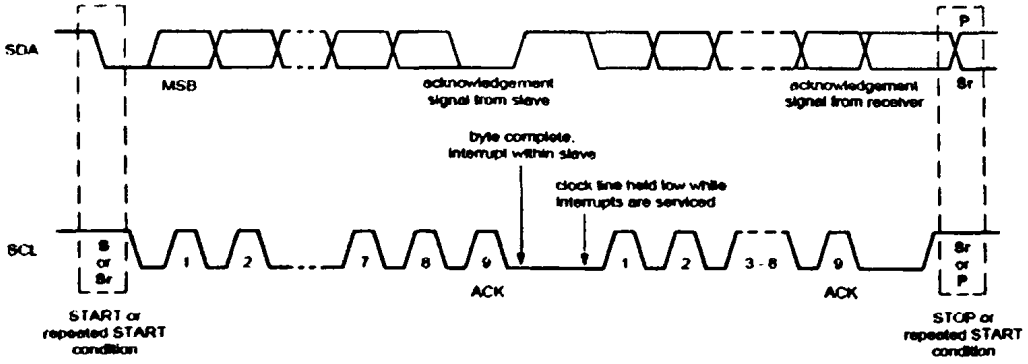
Προτού η Κύρια συσκευή επιχειρήσει να στείλει ένα μήνυμα, παράγει την ακολουθία Έναρξης η οποία περιγράφεται από την αρνητική μετάβαση της γραμμής SDA, τη στιγμή που η γραμμή SCL βρίσκεται σε χαμηλή στάθμη. Μετά τη λήξη της αποστολής, η γραμμή SDA επιστρέφει σε υψηλή στάθμη, έχοντας πρώτα εξασφαλίσει ότι η SCL βρίσκεται σε επίσης υψηλή στάθμη. Ο συνδυασμός των δύο αυτών σημάτων δηλώνει την ακολουθία Λήξης (Boyer, 2008).

Εικόνα Π.Α.-3



Π.Α.1.3 Μεταφορά Δεδομένων μέσω του Διαύλου I^2C

Το πρώτο ψηφίο του εκπεμπόμενου byte είναι το περισσότερο σημαντικό (Most Significant Bit ή MSB). Μόλις εφαρμοστεί η στάθμη του πάνω στη γραμμή SDA, η Κύρια συσκευή «ανεβάζει» για λίγο το SCL σε υψηλή στάθμη προκειμένου να γνωστοποιήσει στην Εξαρτώμενη συσκευή την παρουσία του ψηφίου δεδομένων. Η μετάδοση συνεχίζεται με τον ίδιο τρόπο και για τα υπόλοιπα ψηφία του byte. Με τον ένατο παλμό του σήματος SCL, η Εξαρτώμενη συσκευή οφείλει να δημιουργήσει το *Σήμα Επιβεβαίωσης του Διαύλου I^2C* (Philips Semiconductors, 2000).

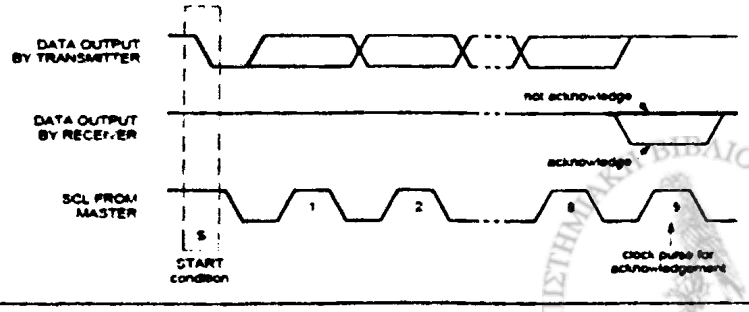


Εικόνα Π.Α.-4

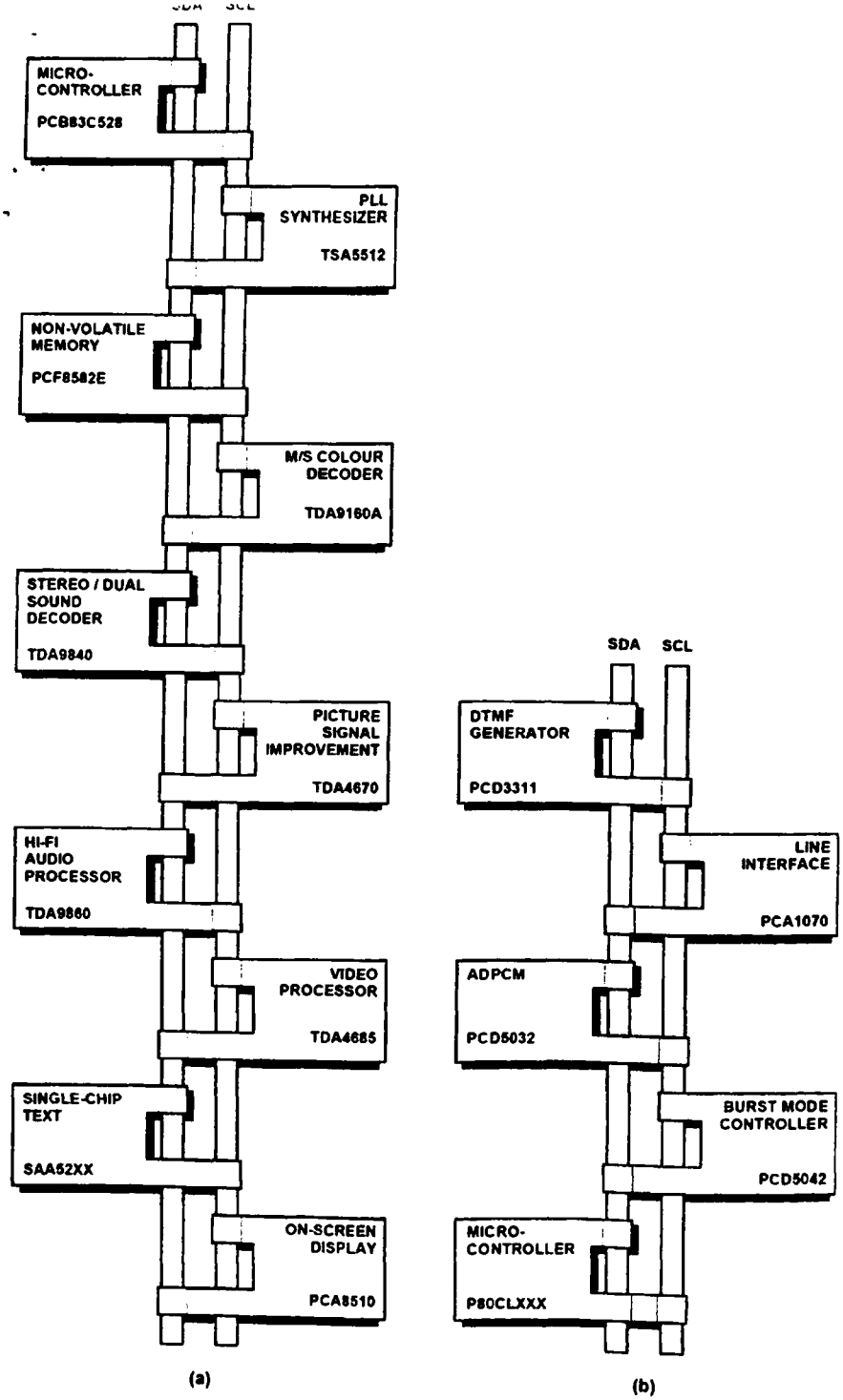
Π.Α.1.4 Σήμα Επιβεβαίωσης του διαύλου I^2C

Αν η εξαρτώμενη συσκευή έχει λάβει σωστά τα ψηφία του μηνύματος, οφείλει να «κατεβάσει» τη γραμμή SDA για χρονικό διάστημα ίσο με αυτό που αντιστοιχεί στο χρόνο ενός ψηφίου (Philips Semiconductors, 2000).

Εικόνα Π.Α.-5



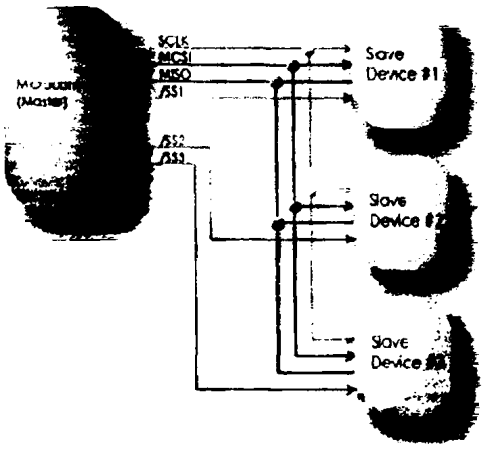
Αν και, όπως σημειώσαμε προηγουμένως, ο δίαυλος I^2C υλοποιείται πολύ εύκολα και ελέγχεται από ένα σχετικά απλό πρωτόκολλο, είναι σε θέση να ανταποκριθεί σε αρκετά πολύπλοκες εφαρμογές. Τα παλμογραφήματα στα οποία γίνεται εκτενής αναφορά εξηγούν με πολύ παραστατικό τρόπο το πώς γίνονται πράξη όλα τα παραπάνω. Αξίζει να σημειωθεί πως η Κύρια συσκευή εκτός από το να εκπέμπει, μπορεί το ίδιο εύκολα να λαμβάνει δεδομένα από την Εξαρτώμενη (Philips Semiconductors, 2000).



Εικόνα Π.Α.-6

Π.Α.2 Ο Δίαυλος SPI

Ο SPI είναι ένας σύγχρονος σειριακός δίαυλος δεδομένων που επινοήθηκε από την Motorola και λειτουργεί με full-duplex τρόπο. Οι συσκευές επικοινωνούν με τρόπο κύρια/εξαρτώμενη (master/slave) όπου η κύρια συσκευή αρχίζει την αποστολή του πακέτου των δεδομένων. Πολλαπλές εξαρτώμενες συσκευές μπορούν να λειτουργήσουν ταυτόχρονα χρησιμοποιώντας ξεχωριστές γραμμές επιλογής (εξαρτώμενης συσκευής) (Intersil, 2007).



Εικόνα Π.Α.-7

Π.Α.2.1 Λειτουργία

Ο δίαυλος SPI μπορεί να λειτουργήσει με μία κύρια συσκευή και μία ή περισσότερες εξαρτώμενες. Εάν χρησιμοποιείται μόνο μία βοηθητική, τότε το SS pin (Slave Select ή Chip Select) θα πάρει την τιμή του λογικού «0» αν το επιτρέψει η εξαρτώμενη συσκευή. Με πολλαπλές εξαρτώμενες συσκευές σε λειτουργία, απαιτείται από την κύρια συσκευή ένα ξεχωριστό σήμα SS για κάθε εξαρτώμενη. Οι περισσότερες συσκευές ενσωματώνουν tri-state (high, low, floating) εξόδους με αποτέλεσμα ο ακροδέκτης MISO να αποκτά μεγάλη αντίσταση και συμπεριφέρεται σαν «αποσυνδεδεμένος» όταν η συσκευή δεν επιλέγεται. Συσκευές χωρίς tri-state εξόδους δεν μπορούν να συνυπάρξουν στο δίαυλο SPI με άλλες συσκευές. Μόνο μία εξαρτώμενη συσκευή μπορεί να επικοινωνήσει με την κύρια, και μόνο η δική της chipselect μπορεί να ενεργοποιηθεί (Freescale Semiconductor, 2010).

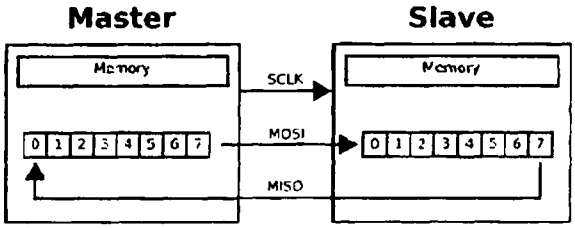
Π.Α.2.2 Μετάδοση Δεδομένων

Για να ξεκινήσει η επικοινωνία, η κύρια συσκευή θα πρέπει πρώτα να διαμορφώσει το ρολόι, χρησιμοποιώντας μια συχνότητα μικρότερη ή ίση από τη μέγιστη συχνότητα που υποστηρίζει η εξαρτώμενη συσκευή. Το εύρος συχνοτήτων είναι συνήθως 1-70 MHz. Στη συνέχεια, η κύρια συσκευή φέρνει το Slave Select της εξαρτώμενης συσκευής που θέλει να επικοινωνήσει σε λογικό «0». Εάν απαιτείται μια περίοδος αναμονής (π.χ. για τη μετατροπή από αναλογικό σε ψηφιακό) τότε η κύρια συσκευή πρέπει να περιμένει, πριν αρχίσει να ασχολείται με τους κύκλους του ρολογιού (Intersil, 2007).

Κατά τη διάρκεια ενός κύκλου ρολογιού του διαύλου SPI, λαμβάνει χώρα μια μετάδοση δεδομένων τύπου full-duplex:

- Η κύρια συσκευή στέλνει ένα bit στη γραμμή MOSI και η εξαρτώμενη συσκευή το διαβάζει.
- Η εξαρτώμενη συσκευή στέλνει ένα bit στη γραμμή MISO και η κύρια συσκευή το διαβάζει.

Αυτές οι λειτουργίες δεν είναι απαραίτητες σε κάθε μετάδοση αλλά πραγματοποιούνται. Οι μεταδόσεις συνήθως περιλαμβάνουν δύο καταχωρητές ολίσθησης (shift registers) κάποιου δεδομένου μεγέθους λέξης (για παράδειγμα οχτώ bits), ένα στην κύρια συσκευή και ένα στη εξαρτώμενη. Τα δεδομένα συνήθως στέλνονται με το πιο σημαντικό bit (MSB) πρώτο. Μετά από αυτό ο καταχωρητής έχει πλέον αποσταλεί και η κύρια και η εξαρτώμενη συσκευή έχουν ανταλλάξει τις τιμές των καταχωρητών. Κατόπιν κάθε συσκευή παίρνει την τιμή αυτή και κάνει κάτι, όπως για παράδειγμα να την γράψει μνήμη. Εάν υπάρχουν περισσότερα στοιχεία για ανταλλαγή, οι καταχωρητές φορτώνονται με τα νέα στοιχεία και η διαδικασία επαναλαμβάνεται. Οι μεταδόσεις μπορούν να περιλάβουν οποιοδήποτε αριθμό κύκλων ρολογιών. Όταν δεν υπάρχουν άλλα στοιχεία για να είναι διαβιβασθούν, η κύρια σταματά το ρολόι της. Κανονικά, έπειτα αποδεσμεύει την εξαρτώμενη συσκευή. Οι μεταδόσεις συνήθως αποτελούνται από λέξεις των 8-bit, ωστόσο και άλλα μεγέθη λέξεων μπορούν να χρησιμοποιηθούν ανάλογα με την εφαρμογή ή/και την εξαρτώμενη συσκευή, όπως για παράδειγμα 16-bit (touchscreen controller, audio codec) ή 12-bit (digital-to-analog, analog-to-digital).

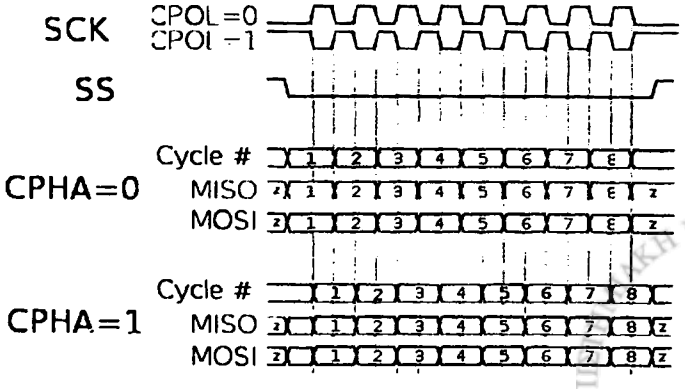


Εικόνα Π.Α.-8: Συνδεσμολογία χρησιμοποιώντας δύο καταχωρητές ολίσθησης

Κάθε εξαρτώμενη συσκευή του διαύλου που δεν έχει ενεργοποιηθεί χρησιμοποιώντας την γραμμή επιλογής (chipselect) πρέπει να απορρίπτει το ρολόι και τα δεδομένα από τη γραμμή MOSI, καθώς και να μην θέσει σε λειτουργία τη γραμμή MISO. Η κύρια συσκευή επιλέγει κάθε φορά μόνο μία βοηθητική συσκευή.

Π.Α.2.3 Πόλωση ρολογιού και φάση

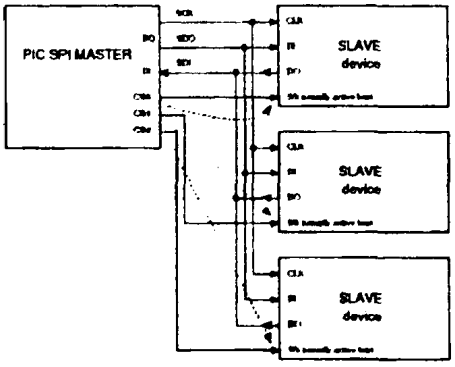
Η κύρια συσκευή εκτός από το να ρυθμίζει τη συχνότητα του ρολογιού, θα πρέπει επίσης να ρυθμίζει την πόλωση και φάση του ρολογιού αναφορικά με τα δεδομένα. Η εταιρία Freescale ονόμασε αυτές τις δύο ρυθμίσεις ως CPOL και CPHA αντίστοιχα. Οι περισσότερες εταιρίες έχουν υιοθετήσει αυτούς τους όρους (Freescale Semiconductor, 2010).



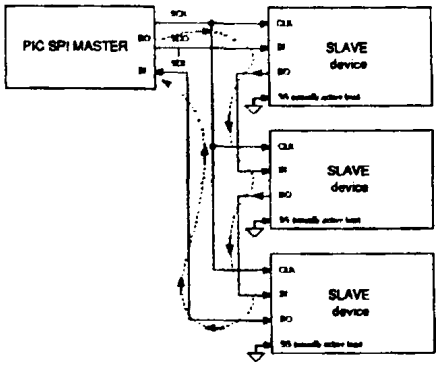
Εικόνα Π.Α.-9: Διάγραμμα χρονισμού, πόλωση ρολογιού και φάση

Π.Α.2.4 Συνδεσμολογίες

Τα περισσότερα προϊόντα με δυνατότητα επικοινωνίας μέσω του διαύλου SPI χρησιμοποιούν τη συνδεσμολογία «αρτηρίας», που είναι και η πιο κοινή. Μερικά άλλα όμως, σχεδιάζονται για να είναι ικανά να συνδεθούν σε μια συνδεσμολογία «αστέρα», όπου η πρώτη εξαρτώμενη έξοδος συνδέεται με τη δεύτερη εξαρτώμενη είσοδο κλπ (Intersil, 2007).



Εικόνα Π.Α.-10: Συνδεσμολογία Αρτηρίας



Εικόνα Π.Α.-11: Συνδεσμολογία Αστέρα

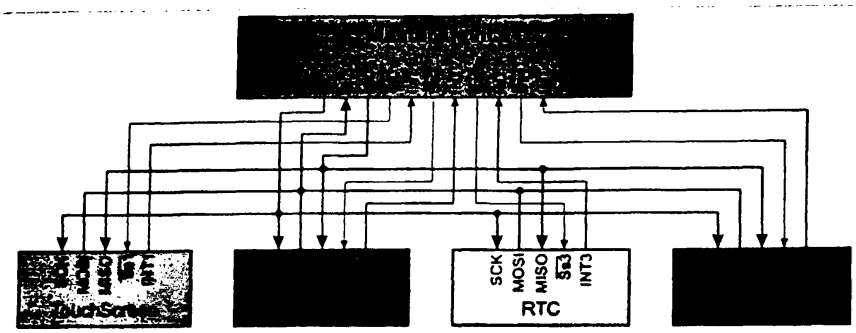
Η SPI θύρα κάθε εξαρτώμενης συσκευής έχει σχεδιαστεί έτσι ώστε να στέλνει κατά τη διάρκεια της δεύτερης ομάδας παλμών του ρολογιού ένα ακριβές αντίγραφο αυτού που έλαβε κατά τη διάρκεια της πρώτης ομάδας παλμών. Ολόκληρη η αλυσίδα ενεργεί ως ένας καταχωρητής ολίσθησης επικοινωνίας SPI, ο «αστέρας» γίνεται συχνά με καταχωρητές ολίσθησης για να παρέχει μια «αποθήκη» εισόδων ή εξόδων μέσω του διαύλου SPI. Ένα τέτοιο χαρακτηριστικό γνώρισμα απαιτεί μόνο μια ενιαία γραμμή SS από την κύρια συσκευή, παρά μια χωριστή γραμμή SS για κάθε εξαρτώμενη συσκευή.

Π.Α.2.5 «Έγκυρη» Επικοινωνία

Μερικές εξαρτώμενες συσκευές σχεδιάζονται με τέτοιο τρόπο ώστε να αγνοούν οποιεσδήποτε επικοινωνίες SPI των οποίων ο αριθμός παλμών του ρολογιού είναι μεγαλύτερος από κάποιο συγκεκριμένο. Άλλες συσκευές δεν ενδιαφέρονται, αγνοώντας τους πρόσθετους παλμούς και συνεχίζουν να στέλνουν το ίδιο bit εξόδου. Είναι σύνηθες για τις διαφορετικές συσκευές να χρησιμοποιούν επικοινωνίες SPI με διαφορετικά μήκη, όπως, παραδείγματος χάριν, όταν χρησιμοποιείται ο SPI για να έχουν πρόσβαση στην scan chain ενός ολοκληρωμένου θέτοντας μια λέξη-εντολή συγκεκριμένου μεγέθους (ίσως 32 bit) και έπειτα να παίρνει απάντηση διαφορετικού μεγέθους (ίσως 153 bit, ένα για κάθε διαφορετικό pin στην scan chain) (Intersil, 2007).

Π.Α.2.6 Διακοπές (Interrupts)

Οι συσκευές SPI μερικές φορές χρησιμοποιούν μια άλλη, ξεχωριστή γραμμή σημάτων για να στείλουν ένα σήμα διακοπής στην κύρια συσκευή. Για παράδειγμα, όταν η γραφίδα ακουμπήσει ένα touchscreen, όταν η θερμοκρασία φτάσει ή ξεπεράσει ένα προκαθορισμένο όριο από έναν αισθητήρα θερμοκρασίας, η εισαγωγή ακουστικών σε ένα κινητό τηλέφωνο κλπ (Atmel Corporation, 2008).



Εικόνα Π.Α.-12: Παράδειγμα SPI επικοινωνίας με interrupts

Π.Α.2.7 Πλεονεκτήματα

- Full duplex επικοινωνία.
- Υψηλότερη ρυθμοαπόδοση από τους διαύλους I^2C και SMBus.
- Πλήρης ευελιξία πρωτοκόλλου για τα bits που μεταφέρονται.
- Εξαιρετικά απλή διασύνδεση υλικού.
- Χαρακτηριστικά χαμηλότερες απαιτήσεις ισχύος από τους διαύλους I^2C και SMBus λόγω των λιγότερων στοιχείων κυκλώματος (συμπεριλαμβανομένων των pullups αντιστάσεων).
- Χρησιμοποιεί πολύ λιγότερους ακροδέκτες στα ολοκληρωμένα, και τα καλώδια στα σχεδιαγράμματα.
- Στη χειρότερη περίπτωση κάθε εξαρτώμενη συσκευή καταλαμβάνει ένα μόνο επιπλέον σήμα (chipselect), τα υπόλοιπα είναι κοινά.

Π.Α.2.8 Μειονεκτήματα

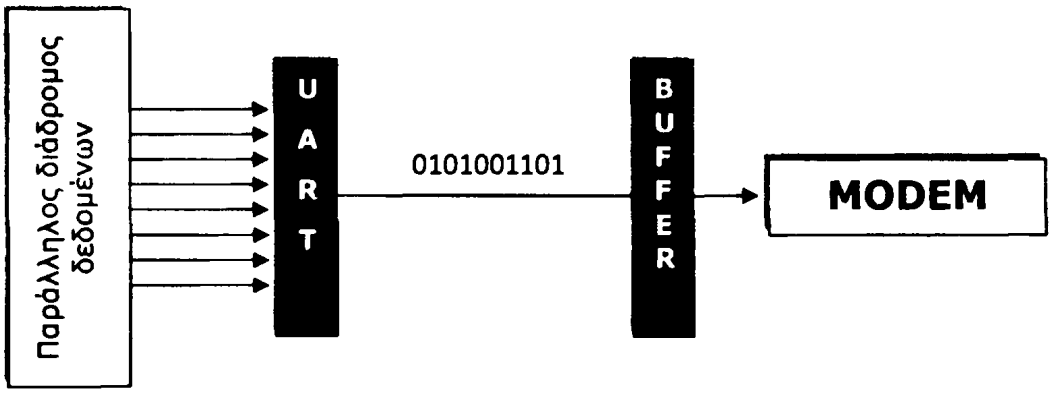
- Απαιτεί περισσότερους ακροδέκτες στα ολοκληρωμένα συγκριτικά με τους διαύλους I^2C και SMBus, ακόμη και στην υλοποίηση «3-wire»
- Δεν υποστηρίζεται διευθυνσιοδότηση
- Δεν υπάρχει έλεγχος ροής υλικού
- Δεν υποστηρίζεται αναγνώριση εξαρτώμενων συσκευών.
- Υποστηρίζει μόνο μια κύρια συσκευή.
- Δεν υπάρχουν επίσημα πρότυπα.
- Χειρίζεται μόνο τις σύντομες αποστάσεις εν αντιθέσει με RS-232 , RS-485 CAN-bus

Π.Α.3 Σειριακή Επικοινωνία

Π.Α.3.1 Γενικά

Ένας εναλλακτικός και πολύ διαδεδομένος τρόπος μετάδοσης πληροφοριών είναι η σειριακή επικοινωνία. Με τον τρόπο αυτό τα bits της πληροφορίας μεταδίδονται ένα κάθε φορά, στη σειρά, μέσα από έναν αγωγό μεταφοράς των δεδομένων. Στην απλούστερη περίπτωση τέτοιας επικοινωνίας χρειαζόμαστε τρεις συνολικά αγωγούς, έναν για την αποστολή δεδομένων (TX), ένα για την λήψη(RX) και ένα που θα βρίσκεται στο δυναμικό αναφοράς των μεταδιδόμενων σημάτων (GND).

Είναι προφανές ότι για να αποσταλούν με σειριακό τρόπο κάποια δεδομένα από την παράλληλη μορφή, με την οποία εμφανίζονται στον διάδρομο δεδομένων, πρέπει πρώτα να μετατραπούν σε σειριακή μορφή. Τη λειτουργία αυτή αναλαμβάνει ένα κύκλωμα που ονομάζεται UART (Universal Asynchronous Receiver/Transmitter), το οποίο υπάρχει σε ολοκληρωμένη μορφή μέσα στον μικροελεγκτή. Η λειτουργία του κυκλώματος αυτού στηρίζεται στη λειτουργία του καταχωρητή ολίσθησης, ο οποίος αφού λάβει κάποια δεδομένα και τα καταχωρήσει στα flip-flops που διαθέτει, ολισθαίνει τα bits της ψηφιολέξης που έχει καταχωρήσει ένα-ένα προς τα δεξιά ή προς τα αριστερά. Οι βασικές ιδέες της σειριακής μετάδοσης που περιγράφηκαν παραπάνω, παραθέτονται στην παρακάτω εικόνα.



Εικόνα Π.Α.-13

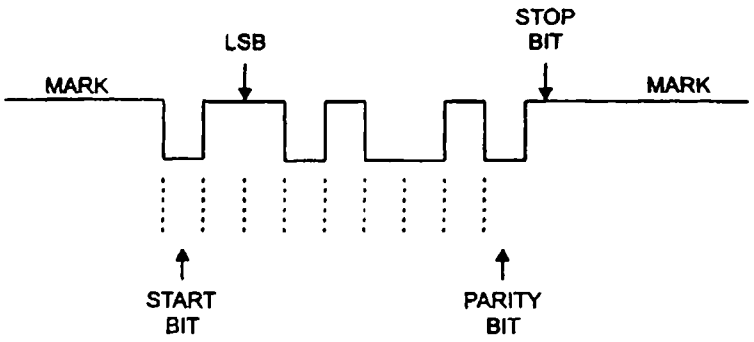
Το κυριότερο πλεονέκτημα της σειριακής επικοινωνίας είναι ο μικρότερος αριθμός καλωδίων διασύνδεσης που απαιτείται, σε σχέση με την παράλληλη επικοινωνία. Επιπλέον, τα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται στην σειριακή επικοινωνία επιτρέπουν μεγάλες στάθμες σημάτων σε σχέση με τα πρωτόκολλα της παράλληλης επικοινωνίας, οπότε οι απώλειες του σήματος δημιουργούν μικρότερο πρόβλημα και η μετάδοση σε μεγάλη απόσταση είναι εφικτή. Εξάλλου με την σειριακή επικοινωνία είναι πολύ ευκολότερη η ασύρματη μετάδοση, ειδικά μέσω διατάξεων υπέρυθρης ακτινοβολίας, που είναι πολύ διαδεδομένες. Τέλος, η σειριακή μετάδοση είναι πιο κατάλληλη για χρήση με μικροελεγκτές, που επίσης έχουν διαδοθεί πολύ τα τελευταία χρόνια. Ο λόγος είναι ότι οι διάφορες διατάξεις, όπως μετατροπείς A/D, μνήμες κλπ καταλαμβάνουν πολύ λιγότερους ακροδέκτες του μικροελεγκτή όταν επικοινωνούν σειριακά με αυτόν, παρά όταν επικοινωνούν παράλληλα. Εξάλλου, μερικά συστήματα μικροελεγκτών έχουν ενσωματωμένες θύρες σειριακής διασύνδεσης με το εξωτερικό περιβάλλον κάτι που καθιστά απλή τη σειριακή διασύνδεσή τους με περιφερειακές συσκευές (Καλόμοιρος, et al., 2002).

Π.Α.3.2 Ασύγχρονη Σειριακή Επικοινωνία

Η ασύγχρονη σειριακή επικοινωνία εξυπηρετεί τη μετάδοση χαρακτήρων που εκπέμπονται από κάποιο πομπό χωρίς κανένα συγχρονισμό, όπως συμβαίνει με τους αλφαριθμητικούς χαρακτήρες που δημιουργούνται όταν πιέζουμε τα πλήκτρα ενός πληκτρολογίου.

Κάθε τέτοιος χαρακτήρας μετατρέπεται σε ψηφιολέξη μέσω του κώδικα ASCII και η ακολουθία bits που αντιστοιχεί σε αυτόν εμφανίζεται στη γραμμή μετάδοσης. Ο δέκτης πρέπει να είναι σε θέση να αναγνωρίζει ότι έφτασε ένας χαρακτήρας και να δέχεται τα bits του χαρακτήρα με τη σωστή σειρά χωρίς απώλειες.

Για τον παραπάνω σκοπό, τα bits της ασύγχρονης σειριακής μετάδοσης οργανώνονται σε ομάδες των εννέα έως δώδεκα bits συνολικά, οι οποίες περιέχουν κάποιους χαρακτήρες έναρξης και λήξης. Το πρώτο bit κάθε πλαισίου είναι το λεγόμενο START BIT, το οποίο αντιστοιχεί σε λογικό «0». Ακολουθεί η σειρά των ψηφίων του χαρακτήρα που αποστέλλεται. Για παράδειγμα αν αποστέλλεται το κεφαλαίο γράμμα Α, τότε η ακολουθία των ψηφίων 01001011. Μετά από τα bits του χαρακτήρα ακολουθεί ένα bit άρτιας ή περιττής ισοτιμίας (parity), το οποίο ενεργοποιεί μια διαδικασία ελέγχου σφαλμάτων, για να ανιχνεύσει τυχόν λάθη που συνέβησαν κατά τη μετάδοση. Το πλαίσιο κλείνει με ένα ή δύο STOP BITS, που υποδηλώνουν το τέλος του χαρακτήρα και την κατάσταση αναμονής για τον επόμενο. Η λογική κατάσταση των ψηφίων λήξης (STOP BITS) είναι το λογικό «1».



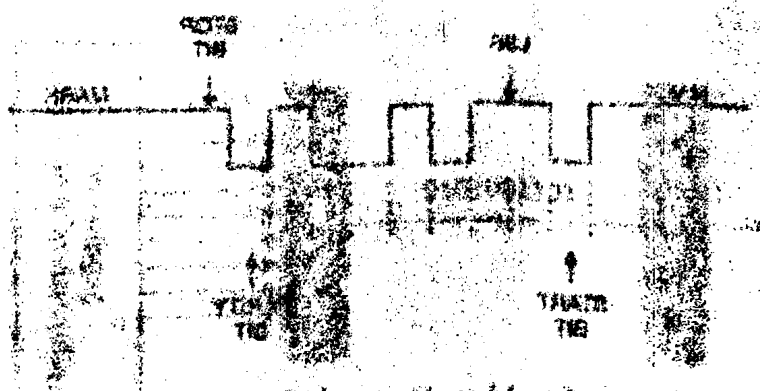
Εικόνα Π.Α.-14: Μορφή του σήματος στην ασύγχρονη μετάδοση χαρακτήρα

Όταν δεν μεταδίδεται κάποιος χαρακτήρας, τότε λέμε ότι η σύνδεση είναι ανενεργή, οπότε η γραμμή βρίσκεται σε λογικό «1». Η κατάσταση αυτή ονομάζεται «συνθήκη MARK». Όταν μεταδοθεί το bit έναρξης και φθάσει στο δέκτη, ο δέκτης καταλαβαίνει ότι ακολουθούν τα bits του χαρακτήρα που αποστέλλεται, οπότε ενεργοποιεί το σύστημα χρονισμού του και διαβάζει με τη σειρά τα επόμενα bits μέχρι τα bits λήξης. Στη συνέχεια τίθεται και πάλι στην κατάσταση αναμονής.

Είναι φανερό ότι η διάρκεια του κάθε εκπεμπόμενου bit στην ασύγχρονη σειριακή μετάδοση πρέπει να είναι αυστηρά η ίδια, ώστε να μπορεί ο δέκτης με βάση κάποιο σύστημα χρονισμού, να μπορεί να διακρίνει τα bits μεταξύ τους. Συνεπώς ο πομπός και ο δέκτης πρέπει να συμφωνούν ως προς την ταχύτητα της σειριακής μετάδοσης των bits. Η ταχύτητα αυτή ορίζει το λεγόμενο «ρυθμό μετάδοσης» (baud rate), που μετριέται σε bits ανά δευτερόλεπτο (bits/sec ή bps). Συνήθεις ρυθμοί στις ασύγχρονες σειριακές επικοινωνίες είναι 2400, 4800, 9600, 14400, 19200, 28800, και 33600 bits/sec. Η μέγιστη ταχύτητα που υποστηρίζει μια θύρα επικοινωνίας είναι 115.2kbps.

Σαν παράδειγμα αναφέρουμε ότι, για να έχουμε ρυθμό μετάδοσης 9600bps, η διάρκεια του κάθε bit πρέπει να είναι 104μs. Κάθε χαρακτήρας θα διαρκεί στη γραμμή 1,14ms.

Το βασικό μειονέκτημα της ασύγχρονης σειριακής μετάδοσης είναι η ανάγκη που προκύπτει για START και STOP bits στην αρχή και στο τέλος κάθε χαρακτήρα. Με τον τρόπο αυτό επιβαρύνεται σημαντικά η διαδικασία της μετάδοσης με επιπλέον bits που δεν αντιπροσωπεύουν χρήσιμη πληροφορία (Καλόμοιρος, et al., 2002).



Παράρτημα Β: Κώδικας

Π.Β.1 Κώδικας Κύριου Κόμβου

```
1   #include <SPI.h>
2   #include <Wire.h>
3   #include <OneWire.h>
4   #include <SdFat.h>
5   #include <SdFatUtil.h>
6   #include <Time.h>
7   #include <TimeAlarms.h>
8   #include <Ethernet.h>
9   #include <DS1307RTC.h>
10  #include <DallasTemperature.h>
11  #include <LiquidCrystal_I2C.h>
12  #include <I2C_eeprom.h>
13
14  #include <XBeeMega.h>
15
16  LiquidCrystal_I2C lcd(0x20,16,2);  // dieu8insi LCD 0x20 16 chars, 2 grammes
17
18  #define BUFSIZ 256
19
20  I2C_eeprom ee(0x57);
21    ,
22  #define ONE_WIRE_BUS 49
23  #define TEMPERATURE_PRECISION 12
24  OneWire oneWire(ONE_WIRE_BUS);
25  DallasTemperature sensors(&oneWire);
26  DeviceAddress insideThermometer;
27
28
29  /************* ETHERNET STUFF ************/
30  byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
31  Server server(80);
32
33  /************* SDCARD STUFF ************/
34  Sd2Card card;
35  SdVolume volume;
36  SdFile root;
37  SdFile file;
38
39  /************* RTC STUFF ************/
40  time_t  prevEventTime=0;     // metabliti xronou
```

```
41  time_t duration;
42  time_t timeNow = now();
43  time_t newfile_sec;
44
45  /************* XBEE STUFF *************/
46  XBee xbee = XBee();  // dimiourgia XBee object
47  uint8_t payload[] = { 0, 0, 0};
48  ZBTxStatusResponse txStatus = ZBTxStatusResponse();
49  XBeeResponse response = XBeeResponse();
50  ZBRxResponse rx = ZBRxResponse();
51  ModemStatusResponse msr = ModemStatusResponse();
52  int statusLed = 18;
53  int errorLed = 43;
54  int dataLed = 45;
55  uint8_t ndCmd[] = { 'N','D'};
56  uint8_t nmCmd[] = { 'N','D'};
57  AtCommandRequest atRequest = AtCommandRequest(ndCmd);
58  AtCommandResponse atResponse = AtCommandResponse();
59
60  typedef struct
61  {
62    uint32_t highbyte;
63    uint32_t lowbyte;
64    int NodeIdentifier;
65    int level;
66  }
67  module;
68  module all_modules[10]={};
69
70
71  ////////////////////////////////////////////////////////////////////////////
72
73  void flashLed(int pin, int times, int wait)
74  {
75    for (int i = 0; i < times; i++)
76    {
77      digitalWrite(pin, HIGH);
78      delay(wait);
79      digitalWrite(pin, LOW);
80
81      if (i + 1 < times)
82      {
83        delay(wait);
84      }
85    }
```

```
86  }
87
88  ////////////////////////////////////////////////////////////////////////////
89
90  #define error(s) error_P(PSTR(s)) //apo8ikeusi la8ous sti flash
91  void error_P(const char* str)
92  {
93    //Serial3.print("error: ");     //gia debugging
94    //Serial3.println(str);        //gia debugging
95    lcd.clear();
96    lcd.setCursor(0, 0);
97    lcd.print("error: ");
98    lcd.setCursor(6, 0);
99    lcd.print(str);
100   Alarm.delay(500);
101
102   if (card.errorCode())
103   {
104     //Serial3.print("SD error: ");             //gia debugging
105     //Serial3.print(card.errorCode(), HEX);    //gia debugging
106     //Serial3.print(",");                      //gia debugging
107     //Serial3.println(card.errorData(), HEX); //gia debugging
108     lcd.setCursor(0, 1);
109     lcd.print("SD error: ");
110     lcd.setCursor(9, 1);
111     lcd.print(card.errorCode(), HEX);
112     lcd.setCursor(12, 1);
113     lcd.print(",");
114     lcd.setCursor(13, 1);
115     lcd.print(card.errorData(), HEX);
116     for (int i = 0; i < 10; i++)
117     {
118       lcd.noBacklight();
119       Alarm.delay(500);
120       lcd.backlight();
121     }
122   }
123 }
124
125 ////////////////////////////////////////////////////////////////////////////
126
127 int k;
128 int kl;
129 int kh;
130 int hits;
```

```
131  int hitsh;
132  int hitsl;
133  int g;
134  int gh;
135  int gl;
136  int NODES=-1;
137  byte othoni=0;
138  byte ip01=0;
139  byte ip02=0;
140  byte ip03=0;
141  byte ip04=0;
142
143
144  ////////////////////////////////////////////////////////////////////////////
145
146  void setup()
147  {
148    analogReference(EXTERNAL);
149    pinMode(38, OUTPUT);                    //self reset
150    digitalWrite(38, HIGH);                 //self reset
151    pinMode(11, OUTPUT);                    //forceon fan
152    digitalWrite(11, HIGH);                 //forceon fan
153    pinMode(7, OUTPUT);                     // watchdog
154    digitalWrite(7, HIGH);                  // watchdog
155    Alarm.delay(1);                         // watchdog
156    digitalWrite(7, LOW);                   // watchdog
157    pinMode(13, INPUT);                     //FAN STATUS
158    pinMode(43, INPUT);                     //card STATUS
159    //XMCRA = B10001111;              // initialize external SRAM
160    //XMCRB = B00000001;              // initialize external SRAM
161    Alarm.delay(0);
162    lcd.init();                       // initialize the lcd
163    lcd.clear();
164    lcd.backlight();
165    Wire.beginTransmission(40);
166    delay(10);
167    Wire.send(0x11);
168    Wire.send(0xd9);    // fotismos lcd
169    delay(10);
170    Wire.endTransmission();
171    Wire.beginTransmission(40);
172    delay(10);
173    Wire.send(0x12);
174    Wire.send(0x1f);
175    delay(10);
```

```
176     Wire.endTransmission();
177
178     lcd.print("Initializing.....");
179     //Alarm.delay(500);
180     Serial2.begin(9600);
181     //Serial3.begin(9600);   //gia debugging
182     Serial.begin(9600);
183     pinMode(statusLed, OUTPUT);
184     pinMode(errorLed, OUTPUT);
185     pinMode(dataLed,  OUTPUT);
186     xbee.begin(9600);
187     Alarm.delay(5000);
188     flashLed(statusLed, 3, 50);
189     scan();
190     setSyncProvider(RTC.get);   // the function to get the time from the RTC
191     if(timeStatus()!= timeSet)
192     {
193       lcd.clear();
194       lcd.setCursor(0, 1);
195       lcd.print("RTC sync failed");
196       //Serial3.println("RTC sync failed");      //gia debugging
197       //Alarm.delay(500);                        //gia debugging
198     }
199     else
200     {
201       lcd.clear();
202       lcd.setCursor(0, 1);
203       lcd.print("RTC sync OK!!!");
204       //Serial3.println("RTC sync OK!!!");       //gia debugging
205       //Alarm.delay(500);                        //gia debugging
206     }
207     setSyncInterval(60);      //ka8e pote kanei sync me RTC
208     setTime(hour(), minute(), second(), day(), month(), year());
209     Alarm.timerRepeat(5, othoni2x16);
210     byte ip[] = {192, 168, 2, 100};
211     byte u = ee.readByte(20);   // ip flag
212     if (u==49)  //AN EINAI 1
213     {
214       ip[0] = ee.readByte(16);
215       ip[1] = ee.readByte(17);
216       ip[2] = ee.readByte(18);
217       ip[3] = ee.readByte(19);
218       ee.writeByte(38, ip[0]);
219       ee.writeByte(39, ip[1]);
220       ee.writeByte(40, ip[2]);
```

```
221       ee.writeByte(41, ip[3]);
222     }
223
224     else
225     {
226       ip[0] = ee.readByte(38);
227       ip[1] = ee.readByte(39);
228       ip[2] = ee.readByte(40);
229       ip[3] = ee.readByte(41);
230     }
231
232     if ((ip[0] ==0) && (ip[1] ==0) && (ip[2] ==0) && (ip[3] ==0))
233   {
234       ip[0] = 192;
235       ip[1] = 168;
236       ip[2] = 2;
237       ip[3] = 100;
238
239       ee.writeByte(38, ip[0]);
240       ee.writeByte(39, ip[1]);
241       ee.writeByte(40, ip[2]);
242       ee.writeByte(41, ip[3]);
243
244       ee.writeByte(16, ip[0]);
245       ee.writeByte(17, ip[1]);
246       ee.writeByte(18, ip[2]);
247       ee.writeByte(19, ip[3]);
248   }
249
250     ip01=ip[0];
251     ip02=ip[1];
252     ip03=ip[2];
253     ip04=ip[3];
254
255     ee.writeByte(20, '0');
256     u = ee.readByte(24);   // HR flag
257     if (u==49)  //AN EINAI 1
258     {
259       int hr =  ee.readByte(21);
260       int min = ee.readByte(22);
261       int sec = ee.readByte(23);
262       setTime(hr, min, sec, day(), month(), year());
263       time_t timeNow1 = now();
264       RTC.set(timeNow1);
265     }
```

```
266     ee.writeByte(24, '0');
267
268     u = ee.readByte(28);   // DATE flag
269     if (u==49)  //AN EINAI 1
270     {
271       time_t t_day =  ee.readByte(25);
272       time_t t_month = ee.readByte(26);
273       time_t t_year = ee.readByte(27);
274       setTime(hour(), minute(), second(), t_day, t_month, t_year);
275       time_t timeNow2 = now();
276       RTC.set(timeNow2);
277     }
278     ee.writeByte(28, '0');   //ekswteriki eeprom
279
280     u = ee.readByte(31);   // rec_sec flag
281     if (u==49)  //AN EINAI 1
282     {
283       gh = ee.readByte(29);
284       gl = ee.readByte(30);
285       g = ((int)gh << 8) | gl;
286       duration=g;
287       ee.writeByte(42, gh);
288       ee.writeByte(43, gl);
289     }
290     else
291     {
292       gh = ee.readByte(42);
293       gl = ee.readByte(43);
294       g = ((int)gh << 8) | gl;
295       duration=g;
296     }
297     ee.writeByte(31, '0');   //ekswteriki eeprom
298
299
300     u = ee.readByte(46);   // NEW FILE flag
301     if (u==49)  //AN EINAI 1
302     {
303       u = ee.readByte(34);   //NEW FILE flag
304       if (u==49)  //AN EINAI 1
305       {
306         gh = ee.readByte(32);
307         gl = ee.readByte(33);
308         g = ((int)gh << 8) | gl;
309         newfile_sec=g;
310         Alarm.timerRepeat(g, RepeatTask);        // timer alarm
```

```
311         ee.writeByte(44, gh);
312         ee.writeByte(45, gl);
313       }
314       else
315       {
316         gh = ee.readByte(44);
317         gl = ee.readByte(45);
318         g = ((int)gh << 8) | gl;
319         newfile_sec=g;
320         Alarm.timerRepeat(g, RepeatTask);   // timer alarm
321       }
322       ee.writeByte(34, '0');   //ekswteriki eeprom
323     }
324     else
325     {
326       u = ee.readByte(37);   // NEW FILE2 flag
327       if (u==49)  //AN EINAI 1
328       {
329         int qw = ee.readByte(35);
330         int qe = ee.readByte(36);
331         Alarm.alarmRepeat(qw,qe,0, RepeatTask);  // every day
332         ee.writeByte(47, gh);
333         ee.writeByte(48, gl);
334       }
335       else
336       {
337         int qw = ee.readByte(47);
338         int qe = ee.readByte(48);
339         Alarm.alarmRepeat(qw,qe,0, RepeatTask);  // every day
340       }
341       ee.writeByte(34, '0');   //ekswteriki eeprom
342     }
343     sensors.begin();
344     sensors.getAddress(insideThermometer, 0);
345     sensors.setResolution(insideThermometer, 12);  //12= posa bit resolution
346
347     digitalClockDisplay();
348
349     lcd.clear();
350     lcd.setCursor(0, 1);
351     lcd.print("Free RAM: ");
352     lcd.setCursor(9, 1);
353     lcd.print(FreeRam());
354     //Serial3.print("Free RAM: "); //gia debugging
355     //Serial3.println(FreeRam());  //gia debugging
```

```
356     pinMode(10, OUTPUT);          // SS pin eksodos (aparaitito!)
357     digitalWrite(10, HIGH);       // apenergopoiisi W5100 chip!
358     Alarm.delay(500);
359     if (!card.init(SPI_HALF_SPEED, 42)) error("card.init failed!");
360     if (!volume.init(&card)) error("vol.init failed!");//initialize FATvolume
361
362     lcd.clear();
363     lcd.setCursor(0, 1);
364     lcd.print("Volume is FAT");
365     lcd.setCursor(13, 1);
366     lcd.print(volume.fatType(),DEC);
367     //Serial3.print("Volume is FAT");         //gia debugging
368     //Serial3.println(volume.fatType(),DEC); //gia debugging
369     //Alarm.delay(500);                      //gia debugging
370
371     if (!root.openRoot(&volume)) error("openRoot failed");//anoigma tou root
372
373     Ethernet.begin(mac, ip);
374     server.begin();
375     digitalWrite(7, HIGH);                    // watchdog
376     Alarm.delay(1);                           // watchdog
377     digitalWrite(7, LOW);                     // watchdog
378     //Serial3.print("MILLIS:");               //gia debugging
379     //Serial3.println(millis());              //gia debugging
380     digitalWrite(11, LOW);                    //forceon fan
381   }
382
383   ////////////////////////////////////////////////////////////////////////////////
384   void RepeatTask()
385   {
386     scan();
387     k=1;
388     kl=lowByte(k);
389     kh=highByte(k);
390     ee.writeByte(12, kh);
391     ee.writeByte(13, kl);
392
393     char name[] = "LOGGER00.CSV";
394
395     int o = day();
396     if (o<10)
397     {
398       name[0] = '0';
399       name[1] = o % 10 + '0';
400     }
```

```
401     else
402     {
403       name[0] = (o-(o%10))/10 + '0';
404       name[1] = o % 10 + '0';
405     }
406
407     int p = month();
408     if (p<10)
409     {
410       name[2] = '0';
411       name[3] = p % 10+ '0';
412     }
413     else
414     {
415       name[2] = ((p-(p%10))/10)+ '0';
416       name[3] = p % 10+ '0';
417     }
418     int u = year();
419     u=u-2000;
420     if (u<10)
421     {
422       name[4] = '0';
423       name[5] = u % 10+ '0';
424     }
425     else
426     {
427       name[4] = ((u-(u%10))/10)+'0';
428       name[5] = u % 10+ '0';
429     }
430
431     for (uint8_t i = 0; i < 100; i++)
432     {
433       name[6] = i/10 + '0';
434       name[7] = i%10 + '0';
435       if (file.open(&root, name, O_CREAT | O_WRITE | O_EXCL | O_APPEND |
436       O_SYNC)) break;
437     }
438     file.timestamp(T_WRITE, year(), month(), day(), hour(), minute(),
439     second());
440     file.print("A/A");
441     file.print(";Hmerominia");
442     file.print(";Wra");
443     file.print(";Thermokrasia sustimatos");
444
445     for (byte i = 0; i < NODES; i++)
```

```
446     {
447       file.print(";NODE ");
448       file.print(all_modules[i].NodeIdentifier,DEC);
449       //byte h=i+1;
450       //file.print(h,DEC);
451     }
452
453     file.println();
454     if (file.writeError || !file.sync())
455     {
456       error("write header failed");
457     }
458     file.close();
459
460     for (int q = 0; q < 12; q++)
461     {
462       ee.writeByte(q, name[q]);    //ekswteriki eeprom
463     }
464
465     lcd.clear();
466     lcd.setCursor(0, 0);
467     lcd.print("New file created");
468     lcd.setCursor(0, 1);
469     lcd.print(name);
470     //Serial3.print("New file created:"); //gia debugging
471     //Serial3.println(name);              //gia debugging
472     //Alarm.delay(500);                   //gia debugging
473   }
474
475   ////////////////////////////////////////////////////////////////////////////
476
477   void othoni2x16()
478   {
479     switch (othoni)
480     {
481     case 0:
482       {
483         lcd.clear();
484         lcd.setCursor(0, 0);
485         lcd.print("IP Address:");
486         lcd.setCursor(0, 1);
487         lcd.print(ip01,DEC);
488         lcd.print(".");
489         lcd.print(ip02,DEC);
490         lcd.print(".");
```

```
491         lcd.print(ip03,DEC);
492         lcd.print(".");
493         lcd.print(ip04,DEC);
494       }
495       break;
496
497     case 1:
498       {
499         lcd.clear();
500         lcd.setCursor(3, 0);
501         printDigits(day());
502         lcd.setCursor(5, 0);
503         lcd.print("/");
504         lcd.setCursor(6, 0);
505         printDigits(month());
506         lcd.setCursor(8, 0);
507         lcd.print("/");
508         lcd.setCursor(9, 0);
509         lcd.print(year());
510
511         lcd.setCursor(5, 1);
512         printDigits(hour());
513         lcd.setCursor(7, 1);
514         lcd.print(":");
515         lcd.setCursor(8, 1);
516         printDigits(minute());
517       }
518       break;
519
520     case 2:
521       {
522         lcd.clear();
523         lcd.setCursor(0, 0);
524         lcd.print("Vcc1: ");
525         float vcc1=analogRead(A3)*0.0048828125;
526         lcd.print(vcc1);
527         lcd.setCursor(11, 0);
528         lcd.print("Volts");
529
530         lcd.setCursor(0, 1);
531         lcd.print("Vcc2: ");
532         float vcc2=analogRead(A1)*0.0048828125;
533         lcd.print(vcc2);
534         lcd.setCursor(11, 1);
535         lcd.print("Volts");
```

```
536       }
537       break;
538
539     case 3:
540       {
541         lcd.clear();
542         lcd.setCursor(0, 0);
543         lcd.print("System Temp");
544         lcd.setCursor(11, 0);
545         lcd.print(0xFF, BYTE);
546         lcd.print("FAN");
547         lcd.setCursor(2, 1);
548         lcd.print(sensors.getTempC(insideThermometer));
549         lcd.print(223,BYTE);
550         lcd.print("C");
551         lcd.setCursor(11, 1);
552         lcd.print(0xFF, BYTE);
553         lcd.setCursor(12, 1);
554         byte buttonState = digitalRead(13);
555         if (buttonState == HIGH)
556         {
557           lcd.print("ON");
558         }
559         else
560         {
561           lcd.print("OFF");
562         }
563       }
564       break;
565
566     case 4:
567       {
568         lcd.clear();
569         lcd.setCursor(2, 0);
570         lcd.print("microSD card:");
571         byte cardState = digitalRead(43);
572         if (cardState == LOW)
573         {
574           lcd.setCursor(6, 1);
575           lcd.print("PRESENT");
576         }
577         else
578         {
579           lcd.setCursor(2, 1);
580           lcd.print("NOT PRESENT");
```

```
581         }
582       }
583       break;
584
585     case 5:
586       {
587         lcd.clear();
588         lcd.setCursor(0, 0);
589         lcd.print("Record Every:");
590         lcd.setCursor(2, 1);
591         lcd.print(duration,DEC);
592         lcd.setCursor(7, 1);
593         lcd.print("sec.");
594       }
595       break;
596
597     }
598     othoni=othoni+1;
599     if (othoni>5)
600     {
601       othoni=0;
602     }
603   }
604
605   ////////////////////////////////////////////////////////////////////////////////
606   void printDigits(int digits)
607   {
608     if(digits < 10)
609       lcd.print('0');
610     lcd.print(digits);
611   }
612
613   ////////////////////////////////////////////////////////////////////////////////
614
615   void ListFiles(Client client, uint8_t flags)
616   {
617     dir_t p;
618     root.rewind();
619     client.println("<ul>");
620     while (root.readDir(p) > 0)
621     {
622       if (p.name[0] == DIR_NAME_FREE) break;
623       if (p.name[0] == DIR_NAME_DELETED || p.name[0] == '.') continue;
624       if (!DIR_IS_FILE_OR_SUBDIR(&p)) continue;
625       client.print("<li><a href=\"");
```

```
626       for (uint8_t i = 0; i < 11; i++)
627       {
628         if (p.name[i] == ' ') continue;
629         if (i == 8)
630         {
631           client.print('.');
632         }
633         client.print(p.name[i]);
634       }
635       client.print("\">");
636
637
638       for (uint8_t i = 0; i < 11; i++)
639       //print file name with possible blank fill
640       {
641         if (p.name[i] == ' ') continue;
642         if (i == 8)
643         {
644           client.print('.');
645         }
646         client.print(p.name[i]);
647       }
648       client.print("</a>");
649
650       if (DIR_IS_SUBDIR(&p))
651       {
652         client.print('/');
653       }
654       if (flags & LS_DATE) // print modify date/time if requested
655       {
656         root.printFatDate(p.lastWriteDate);
657         client.print(' ');
658         root.printFatTime(p.lastWriteTime);
659       }
660       if (!DIR_IS_SUBDIR(&p) && (flags & LS_SIZE))  //print size if requested
661       {
662         client.print(' ');
663         client.print(p.fileSize);
664         client.print(" Bytes");
665       }
666       client.println("</li>");
667     }
668     client.println("</ul>");
669   }
670   ////////////////////////////////////////////////////////////////////////////////
```

```
671
672  void digitalClockDisplay()
673  {
674    lcd.clear();
675    lcd.setCursor(0, 0);
676    lcd.print(hour());
677    lcd.setCursor(2, 0);
678    lcd.print(":");
679    lcd.setCursor(3, 0);
680    lcd.print(minute());
681    lcd.setCursor(5, 0);
682    lcd.print(":");
683    lcd.setCursor(6, 0);
684    lcd.print(second());
685    lcd.setCursor(0, 1);
686    lcd.print(day());
687    lcd.setCursor(2, 1);
688    lcd.print("-");
689    lcd.setCursor(3, 1);
690    lcd.print(month());
691    lcd.setCursor(5, 1);
692    lcd.print("-");
693    lcd.setCursor(6, 1);
694    lcd.print(year());
695  }
696
697  ////////////////////////////////////////////////////////////////////////////
698
699  void sendAtCommand()
700  {
701    //Serial3.println("Sending command to the XBee");   //gia debugging
702    xbee.send(atRequest);  // send the command
703    if (xbee.readPacket(15000))
704    // wait up to 5 seconds for the status response
705    { // got a response!
706      if (xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE)
707      // should be an AT command response
708      {
709        xbee.getResponse().getAtCommandResponse(atResponse);
710        if (atResponse.isOk())
711        {
712          //Serial3.print("Command [");                 //gia debugging
713          //Serial3.print(atResponse.getCommand()[0]); //gia debugging
714          //Serial3.print(atResponse.getCommand()[1]); //gia debugging
715          //Serial3.println("] was successful!");       //gia debugging
```

```
716           if (atResponse.getValueLength() > 0)
717           {
718             //Serial3.print("Command value length is ");   //gia debugging
719             //Serial3.println(atResponse.getValueLength(), DEC); //debugging
720             //Serial3.print("Command value: ");             //gia debugging
721             for (int i = 0; i < atResponse.getValueLength(); i++)
722             {
723               //Serial3.print(atResponse.getValue()[i], HEX);  //gia debugging
724               //Serial3.print(" ");                            //gia debugging
725             }
726             //Serial3.println("");                            //gia debugging
727           }
728         }
729         else
730         {
731           //Serial3.print("Command return error code: ");     //gia debugging
732           //Serial3.println(atResponse.getStatus(), HEX);     //gia debugging
733         }
734       }
735       else
736       {
737         //Serial3.print("Expected AT response but got ");     //gia debugging
738         //Serial3.print(xbee.getResponse().getApiId(), HEX);  //gia debugging
739       }
740     }
741     else
742     {
743       if (xbee.getResponse().isError()) // at command failed
744       {
745         //Serial3.print("Error reading packet.  Error code: ");//gia debugging
746         //Serial3.println(xbee.getResponse().getErrorCode());  //gia debugging
747       }
748       else
749       {
750         //Serial3.println("No response from radio");           //gia debugging
751       }
752     }
753   }
754
755   ////////////////////////////////////////////////////////////////////////////
756
757
758   void scan()
759   {
760     digitalWrite(45, HIGH);
```

```
761     int NETWORKStringLength=-1;
762     Alarm.delay(1500);
763     Serial2.print("+++");
764     Alarm.delay(1500);
765     String NETWORK = String();
766     Serial2.flush();
767     Serial2.println("ATND");
768     byte index = 0;
769     byte qq=0;
770
771     timeNow = now();
772     while (now()-timeNow<=15)
773     {
774       while(Serial2.available() > 0)
775       {
776         if (qq==3)
777         {
778           goto telos_scanning;
779         }
780         char c = Serial2.read();
781
782         NETWORK += c;
783
784         index++;
785         if (c == '\r' && qq<3)
786         {
787           qq++;
788         }
789         else
790         {
791           qq=0;
792         }
793       }
794       digitalWrite(45, LOW);
795     }
796   telos_scanning:
797     NETWORKStringLength = NETWORK.length();
798     NODES=((NETWORKStringLength-1)/54);
799
800     digitalWrite(45, LOW);
801
802     int z=0;
803     int offset=0;
804     for (int i=0;i<NODES;i++)
805     {
```

```
806       int Pos_a = 4+z;
807       int Pos_b = Pos_a + 9;
808       String high_byte1 = NETWORK.substring(Pos_a,Pos_b);
809       char high_byte2[high_byte1.length()+1];
810       high_byte1.toCharArray(high_byte2, sizeof(high_byte2));
811       unsigned long n = strtoul(high_byte2,NULL,16);
812
813       int Pos_c = Pos_a + 10;
814       int Pos_d = Pos_c + 9;
815       String low_byte1 = NETWORK.substring(Pos_c,Pos_d);
816       char low_byte2[low_byte1.length()+1];
817       low_byte1.toCharArray(low_byte2, sizeof(low_byte2));
818       unsigned long l = strtoul(low_byte2,NULL,16);
819
820
821
822       int Pos_e = Pos_d + 1;
823       Pos_e = Pos_e + 6; // se periptwsi pou einai "MODULE02"....KLP
824       //Pos_e = Pos_e + 7; // se periptwsi pou einai "MODULE 2"....KLP
825       int Pos_f = Pos_e + 1;
826       String NodeIdentifier1 = NETWORK.substring(Pos_e,Pos_f);
827       char NodeIdentifier2[NodeIdentifier1.length()+1];
828       NodeIdentifier1.toCharArray(NodeIdentifier2, sizeof(NodeIdentifier2));
829       int y = atoi(NodeIdentifier2);
830       //Serial3.println(y);                    //gia debugging
831       //Serial3.println(y,DEC);                //gia debugging
832       //Serial3.println(NodeIdentifier2);      //gia debugging
833       //Serial3.println();                     //gia debugging
834
835       z=z+54;
836
837       //Serial3.flush();                       //gia debugging
838       Serial2.flush();
839
840       all_modules[i].highbyte=n;
841       all_modules[i].lowbyte=l;
842       all_modules[i].NodeIdentifier=y;
843     }
844     //Serial3.println(all_modules[0].NodeIdentifier);     //gia debugging
845     //Serial3.println(all_modules[1].NodeIdentifier);     //gia debugging
846     //Serial3.println();                                  //gia debugging
847   }
848
849   ////////////////////////////////////////////////////////////////////////////
850
```

```
851  void loop()
852  {
853    Alarm.delay(0);
854    //lcd.clear();
855    //lcd.setCursor(0, 0);
856    //lcd.print("Main Program");
857    digitalWrite(7, HIGH);                    // watchdog
858    Alarm.delay(1);                           // watchdog
859    digitalWrite(7, LOW);                     // watchdog
860
861    timeNow = now();
862    if (timeNow-prevEventTime >= duration)
863    {
864      char name [] = "00000000.000";
865      for (int s = 0; s < 12; s++)
866      {
867        name[s] = ee.readByte(s);   //ekswteriki
868      }
869      kh = ee.readByte(12);
870      kl = ee.readByte(13);
871      k = ((int)kh << 8) | kl;
872      file.open(&root, name, O_WRITE | O_APPEND);
873      Alarm.delay(10);
874      file.writeError = 0;
875      file.print(k);
876      file.print(';');
877      file.print(day());
878      file.print('-');
879      file.print(month());
880      file.print('-');
881      file.print(year());
882      file.print(';');
883      file.print(hour());
884      file.print(':');
885      file.print(minute());
886      file.print(':');
887      file.print(second());
888      file.print(';');
889      sensors.requestTemperatures(); // entoli gia 8ermokrasia sustimatos
890      file.print(sensors.getTempC(insideThermometer));
891
892      for (int i=0;i<NODES;i++)
893      {
894        uint32_t nl=all_modules[i].highbyte;
895        uint32_t ll=all_modules[i].lowbyte;
```

```
896         payload[0] = '|';
897         int dur1=highByte(duration);
898         int dur2=lowByte(duration);
899         payload[1]=dur1;
900         payload[2]=dur2;
901
902         XBeeAddress64 addr64 = XBeeAddress64(n1,l1);
903         ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
904
905         xbee.send(zbTx);
906         Serial.println("stelnei");
907         if (xbee.readPacket(500))
908         {
909           Serial.println("read packet");
910           if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE)
911           {
912             Serial.println("get responce");
913             xbee.getResponse().getZBTxStatusResponse(txStatus);
914           }
915         }
916         delay(3000); //me 3000 douleuei!!!!
917         xbee.readPacket();
918         if (xbee.getResponse().isAvailable())
919         {      // got something
920           Serial.println("got sthing");
921           if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
922           {        // got a zb rx packet
923             xbee.getResponse().getZBRxResponse(rx);
924             if (rx.getOption() == ZB_PACKET_ACKNOWLEDGED)
925             {             // the sender got an ACK
926               Serial.println("packet acknowledged");
927             }
928             else
929             {
930               Serial.println("packet not acknowledged");
931             }
932
933             file.print(';');
934             file.print(rx.getData()[0], DEC);
935             file.print(rx.getData()[1], BYTE);
936             file.print(rx.getData()[2], DEC);
937             all_modules[i].level= rx.getData(3);
938             Serial.print("data apo xbee:");
939             Serial.println(rx.getDataLength());
940
```

```
941             // for (int p = 0; p < rx.getDataLength(); p++)
942             //  {////// GIA POLLA DEDOMENA BOLEUEI KALYTERA...//////
943             //    file.print(rx.getData()[p], DEC);
944             //  }
945           }
946         }
947         else if (xbee.getResponse().isError())
948         {
949           Serial.println("error");
950           file.print(';');
951           file.print("error code:");
952           file.println(xbee.getResponse().getErrorCode());
953         }
954         else
955         {
956           file.print(';');
957           file.print("agnwsto la8os");
958           Serial.println("agnwsto la8os");
959         }
960       }
961
962       file.println();
963       if (!file.sync()) error("sync failed_metriseis");
964       if (file.writeError) error("write data failed_metriseis");
965
966       file.close();
967       prevEventTime = timeNow;
968       k=k+1;
969       kl=lowByte(k);
970       kh=highByte(k);
971       ee.writeByte(12, kh);
972       ee.writeByte(13, kl);
973     }
974
975     String client_line = String(BUFSIZ);
976     char clientline[BUFSIZ];
977     int index = 0;
978
979     Client client = server.available();
980     if (client)
981     {
982       boolean current_line_is_blank = true;
983       // an http request ends with a blank line
984
985         index = 0;  // reset input buffer
```

```
986       while (client.connected())
987       {
988         if (client.available())
989         {
990           char c = client.read();
991           if (c != '\n' && c != '\r')
992           // If it isn't a new line, add the character to the buffer
993           {
994             clientline[index] = c;
995             client_line += c;
996             index++;
997             if (index >= BUFSIZ)
998             // An einai pio megalo apo buffer, petame dedomenaa
999               index = BUFSIZ -1;
1000
1001            continue;
1002          }
1003          clientline[index] = 0;
1004          // got a \n or \r new line, which means the string is done
1005
1006          lcd.clear();
1007          //lcd.setCursor(0, 0);
1008          //lcd.print("Main Program");
1009          lcd.setCursor(0, 1);
1010          lcd.print(clientline);
1011
1012          if (strstr(clientline, "GET / ") != 0)
1013          // Look for substring such as a request to get the root file
1014          {
1015            client.println("HTTP/1.1 200 OK");
1016            // send a standard http response header
1017            client.println("Content-Type: text/html");
1018            client.println();
1019
1020  selida:
1021
1022            client.print("<head>");
1023            client.print("<meta http-equiv=""Content-Type""
1024            content=""text/html; charset=iso-8859-1"" />");
1025            client.print("<title>WSN</title>");
1026            client.print("<style type=""text/css"">");
1027            client.print("<!--");
1028            client.print(".style1 {color: #FF0000;font-weight: bold;}");
1029            client.print("-->");
1030            client.print("</style>");
```

```
1031            client.print("</head>");
1032            client.print("<body>");
1033            client.print("<table width=""100%"" border=""2"">");
1034            client.print("<tr><th scope=""col""><img
1035            src=""http://www.physics.uoi.gr/home/sites/default/
1036            files/garland_logo.png"" width=""153"" height=""156"" border=""0""
1037            align=""middle"" /></th>");
1038            client.print("<th scope=""col""><h2><strong>University of
1039            Ioannina</strong></h2>");
1040            client.print("<h2><strong>Physics Department</strong></h2>");
1041            client.print("<h2><strong>P.M.S.-S.H.T.</strong></h2></th></tr>");
1042            client.print("<tr><td rowspan=""2""><div align=""center"">");
1043            client.print("<p><strong>Files:</strong></p>");
1044
1045            ListFiles(client, LS_SIZE);
1046
1047            client.print("<form name=""diagrafi"">");
1048            client.print("<input name=""delete_file"" type=""text""
1049            size=""12"" maxlength=""12"">");
1050            client.print("<input type=""submit"" name=""delete""
1051            value=""Delete""></p></form>");
1052            client.print("<form name=""create""><input name=""Create""
1053            type=""submit"" id=""Create""
1054            value=""Create""></form></div></td>");
1055            client.print("<td><div align=""center"">");
1056            client.print("<form name=""parametroi"">");
1057            client.print("<p><strong>Parameters:</strong></p>");
1058            client.print("<table width=""100%"" border=""0"">");
1059            client.print("<tr><td width=""37%""><div
1060            align=""right"">IP:</div></td>");
1061            client.print("<td width=""56%"">");
1062            client.print("<input name=""ip1"" type=""text"" id=""ip1""
1063            size=""3"" maxlength=""3"">:");
1064            client.print("<input name=""ip2"" type=""text"" id=""ip2""
1065            size=""3"" maxlength=""3"">:");
1066            client.print("<input name=""ip3"" type=""text"" id=""ip3""
1067            size=""3"" maxlength=""3"">:");
1068            client.print("<input name=""ip4"" type=""text"" id=""ip4""
1069            size=""3"" maxlength=""3""></td>");
1070            client.print("<td width=""7%""> ");
1071            client.print("<input type=checkbox name=ip value=1></td></tr>");
1072            client.print("<tr><td colspan=""3""> </td></tr>");
1073            client.print("<tr><td><div align=""right"">RTC time:</div></td>");
1074            client.print("<td><input name=""hour"" type=""text"" size=""1""
1075            maxlength=""2"">:");
```

```
1076          client.print("<input name=""min"" type=""text"" size=""1""
1077          maxlength=""2"">:");
1078          client.print("<input name=""sec"" type=""text"" size=""1""
1079          maxlength= "2""></td>");
1080          client.print("<td><input type=""checkbox"" name=""time""
1081          value=""1""></td></tr>");
1082          client.print("<tr><td><div align=""right"">RTC date:</div></td>");
1083          client.print("<td><input name=""day"" type=""text"" size=""1""
1084          maxlength=""2"">/");
1085          client.print("<input name=""month"" type=""text"" size=""1""
1086          maxlength=""2"">/");
1087          client.print("<input name=""year"" type=""text"" size=""1""
1088          maxlength=""4""></td>");
1089          client.print("<td><input type=""checkbox"" name=""date""
1090          value=""1""></td></tr>");
1091          client.print("<tr><td colspan=""3""> </td></tr>");
1092          client.print("<tr><td><div align=""right"">Record
1093          Every:</div></td>");
1094          client.print("<td><input name=""rec_sec"" type=""text"" size=""1""
1095          maxlength=""5"">sec</td>");
1096          client.print("<td><input type=""checkbox"" name=""rec_sec2""
1097          value=""1""></td></tr>");
1098          client.print("<tr><td colspan=""3""><div
1099          align=""center""></div></td></tr>");
1100          client.print("<tr><td colspan=""3""></td></tr>");
1101          client.print("<tr><td><div align=""right"">Create new file every:
1102          </div></td>");
1103          client.print("<td><input name=""newfile_sec"" type=""text""
1104          size=""1"" maxlength=""5"">sec</td>");
1105          client.print("<td><input type=""checkbox"" name=""newfile_sec2""
1106          value=""1""></td></tr>");
1107          client.print("<tr><td colspan=""3""><div
1108          align=""center"">or</div></td></tr>");
1109          client.print("<tr><td><div align=""right"">every day
1110          at:</div></td>");
1111          client.print("<td><input name=""newfile_daily_hour"" type=""text""
1112          size=""1"" maxlength=""2"">:");
1113          client.print("<input name=""newfile_daily_min"" type=""text""
1114          size=""1"" maxlength=""2""></td>");
1115          client.print("<td><input type=""checkbox"" name=""newfile_daily""
1116          value=""1""></td></tr>");
1117          client.print("<tr><td colspan=""2""><div align=""center""
1118          class=""style1"">");
1119          client.print("<div align=""right""> RESET MCU -&gt;  -&gt;
1120          </div>");
```

```
1121          client.print("</div></td>");
1122          client.print("<td><input type=""checkbox"" name=""reset_mcu""
1123          value=""1""></td></tr>");
1124          client.print("<tr><td colspan=""3""><div align=""center"">");
1125          client.print("<input type=""submit"" "" value=""Submit"">");
1126          client.print("<input name=""Reset MCU"" type=""reset"" id=""Reset
1127          MCU"" value=""Reset""></div></td></tr>
1128          </table></form></div></td></tr>");
1129          client.print("<tr><td><form name=""scan"">Total Nodes found : ");
1130          client.print(NODES);
1131          client.print("+1");
1132          client.print("<input name=""Scan"" type=""submit"" id=""Scan"" ""
1133          value=""Scan""></form>");
1134
1135          for (byte i = 0; i < NODES; i++)
1136          {
1137            client.print("<p>Battery level, Node ");
1138            client.print(all_modules[i].NodeIdentifier,DEC);
1139            client.print(": ");
1140            client.print(all_modules[i].level, DEC);
1141            client.print("%");
1142            client.print("</p>");
1143          }
1144          client.print("</td></tr>");
1145          client.print("<tr><td colspan=""2""><table width=""100%""
1146          border=""0"">");
1147          client.print("<tr><td>Kolotouros Dimitris-Marios, Ioannina,
1148          2011</td>");
1149          client.print("<td><p align=""right"">Last Refresh : ");
1150          client.print(hour());
1151          client.print(':');
1152          client.print(minute());
1153          client.print(':');
1154          client.print(second());
1155          client.print("  ");
1156          client.print(day());
1157          client.print('/');
1158          client.print(month());
1159          client.print('/');
1160          client.print(year());
1161          client.print("</p>");
1162          client.print("<p align=""right"">Hits: ");
1163
1164          hitsh = ee.readByte(14);
1165          hitsl = ee.readByte(15);
```

```
1166            hits = ((int)hitsh << 8) | hitsl;
1167            client.print(hits);
1168
1169            client.print("</p></td></tr></table></td></tr></table>");
1170            client.print("</body>");
1171            client.print("</html>");
1172
1173            hits = hits + 1;
1174            hitsl=lowByte(hits);
1175            hitsh=highByte(hits);
1176            ee.writeByte(14, hitsh);
1177            ee.writeByte(15, hitsl);
1178
1179            digitalWrite(7, HIGH);                    // watchdog
1180            Alarm.delay(1);                           // watchdog
1181            digitalWrite(7, LOW);                     // watchdog
1182          }
1183
1184
1185          else if (strstr(clientline, "GET /?") != 0)
1186          {
1187            if (client_line.indexOf("?") > -1)
1188            {
1189              if (client_line.indexOf("ip=1") > -1)
1190              {
1191                int Pos_a = client_line.indexOf("ip1=");
1192                int Pos_b = client_line.indexOf("&ip2=");
1193                Pos_a = Pos_a + 4;
1194                String ip1a =client_line.substring(Pos_a,Pos_b);
1195                char ip1b[ip1a.length()+1];
1196                ip1a.toCharArray(ip1b, sizeof(ip1b));
1197                int n = atoi(ip1b);
1198                ee.writeByte(16, n);   //ekswteriki eeprom
1199
1200                Pos_a = client_line.indexOf("ip2=");
1201                Pos_b = client_line.indexOf("&ip3=");
1202                Pos_a = Pos_a + 4;
1203                String ip2a =client_line.substring(Pos_a,Pos_b);
1204                char ip2b[ip2a.length()+1];
1205                ip2a.toCharArray(ip2b, sizeof(ip2b));
1206                n = atoi(ip2b);
1207                ee.writeByte(17, n);
1208
1209                Pos_a = client_line.indexOf("ip3=");
1210                Pos_b = client_line.indexOf("&ip4=");
```

```
1211              Pos_a = Pos_a + 4;
1212              String ip3a =client_line.substring(Pos_a,Pos_b);
1213              char ip3b[ip3a.length()+1];
1214              ip3a.toCharArray(ip3b, sizeof(ip3b));
1215              n = atoi(ip3b);
1216              ee.writeByte(18, n);   //ekswteriki eeprom
1217
1218              Pos_a = client_line.indexOf("ip4=");
1219              Pos_b = client_line.indexOf("&ip=1");
1220              Pos_a = Pos_a + 4;
1221              String ip4a =client_line.substring(Pos_a,Pos_b);
1222              char ip4b[ip4a.length()+1];
1223              ip4a.toCharArray(ip4b, sizeof(ip4b));
1224              n = atoi(ip4b);
1225              ee.writeByte(19, n);   //ekswteriki eeprom
1226
1227              ee.writeByte(20, '1');   //ekswteriki eeprom
1228            }
1229
1230            if (client_line.indexOf("time=1") > -1)
1231            {
1232              int Pos_a = client_line.indexOf("&hour=");
1233              int Pos_b = client_line.indexOf("&min=");
1234              Pos_a = Pos_a + 6;
1235              String houra =client_line.substring(Pos_a,Pos_b);
1236              char hourb[houra.length()+1];
1237              houra.toCharArray(hourb, sizeof(hourb));
1238              int n = atoi(hourb);
1239              ee.writeByte(21, n);   //ekswteriki eeprom
1240
1241              Pos_a = client_line.indexOf("&min=");
1242              Pos_b = client_line.indexOf("&sec=");
1243              Pos_a = Pos_a + 5;
1244              String mina =client_line.substring(Pos_a,Pos_b);
1245              char minb[mina.length()+1];
1246              mina.toCharArray(minb, sizeof(minb));
1247              n = atoi(minb);
1248              ee.writeByte(22, n);   //ekswteriki eeprom
1249
1250              Pos_a = client_line.indexOf("&sec=");
1251              Pos_b = client_line.indexOf("&time=");
1252              Pos_a = Pos_a + 5;
1253              String seca =client_line.substring(Pos_a,Pos_b);
1254              char secb[seca.length()+1];
1255              seca.toCharArray(secb, sizeof(secb));
```

```
1256              n = atoi(secb);
1257              ee.writeByte(23, n);   //ekswteriki eeprom
1258
1259              ee.writeByte(24, '1');
1260            }
1261
1262            if (client_line.indexOf("date=1") > -1)
1263            {
1264              int Pos_a = client_line.indexOf("&day=");
1265              int Pos_b = client_line.indexOf("&month=");
1266              Pos_a = Pos_a + 5;
1267              String daya =client_line.substring(Pos_a,Pos_b);
1268              char dayb[daya.length()+1];
1269              daya.toCharArray(dayb, sizeof(dayb));
1270              int n = atoi(dayb);
1271              ee.writeByte(25, n);   //ekswteriki eeprom
1272
1273              Pos_a = client_line.indexOf("&month=");
1274              Pos_b = client_line.indexOf("&year=");
1275              Pos_a = Pos_a + 7;
1276              String montha =client_line.substring(Pos_a,Pos_b);
1277              char monthb[montha.length()+1];
1278              montha.toCharArray(monthb, sizeof(monthb));
1279              n = atoi(monthb);
1280              ee.writeByte(26, n);   //ekswteriki eeprom
1281
1282              Pos_a = client_line.indexOf("&year=");
1283              Pos_b = client_line.indexOf("&date=");
1284              Pos_a = Pos_a + 6;
1285              String yeara =client_line.substring(Pos_a,Pos_b);
1286              char yearb[yeara.length()+1];
1287              yeara.toCharArray(yearb, sizeof(yearb));
1288              n = atoi(yearb);
1289              ee.writeByte(27, n);   //ekswteriki eeprom
1290
1291              ee.writeByte(28, '1');
1292            }
1293
1294            if (client_line.indexOf("rec_sec2=1") > -1)
1295            {
1296              int Pos_a = client_line.indexOf("rec_sec=");
1297              int Pos_b = client_line.indexOf("&rec_sec2");
1298              Pos_a = Pos_a + 8;
1299              String rec_seca =client_line.substring(Pos_a,Pos_b);
1300              char rec_secb[rec_seca.length()+1];
```

```
1301              rec_seca.toCharArray(rec_secb, sizeof(rec_secb));
1302              g = atoi(rec_secb);
1303              gl=lowByte(g);
1304              gh=highByte(g);
1305              ee.writeByte(29, gh);
1306              ee.writeByte(30, gl);
1307              ee.writeByte(31, '1');
1308            }
1309
1310            if (client_line.indexOf("newfile_sec2=1") > -1)
1311            {
1312              int Pos_a = client_line.indexOf("&newfile_sec=");
1313              int Pos_b = client_line.indexOf("&newfile_sec2=");
1314              Pos_a = Pos_a + 13;
1315              String newfile_seca =client_line.substring(Pos_a,Pos_b);
1316              char newfile_secb[newfile_seca.length()+1];
1317              newfile_seca.toCharArray(newfile_secb, sizeof(newfile_secb));
1318              g = atoi(newfile_secb);
1319              gl=lowByte(g);
1320              gh=highByte(g);
1321              ee.writeByte(32, gh);
1322              ee.writeByte(33, gl);
1323              ee.writeByte(34, '1');
1324              ee.writeByte(46, '1');
1325            }
1326
1327            if (client_line.indexOf("newfile_daily=1") > -1)
1328            {
1329              int Pos_a = client_line.indexOf("&newfile_daily_hour=");
1330              int Pos_b = client_line.indexOf("&newfile_daily_min=");
1331              Pos_a = Pos_a + 20;
1332              String newfile_daily_houra
1333              =client_line.substring(Pos_a,Pos_b);
1334              char newfile_daily_hourb[newfile_daily_houra.length()+1];
1335              newfile_daily_houra.toCharArray(newfile_daily_hourb,
1336              sizeof(newfile_daily_hourb));
1337              int n = atoi(newfile_daily_hourb);
1338              ee.writeByte(35, n);
1339
1340              Pos_a = client_line.indexOf("&newfile_daily_min=");
1341              Pos_b = client_line.indexOf("&newfile_daily=1");
1342              Pos_a = Pos_a + 19;
1343              String newfile_daily_mina =client_line.substring(Pos_a,Pos_b);
1344              char newfile_daily_minb[newfile_daily_mina.length()+1];
1345              newfile_daily_mina.toCharArray(newfile_daily_minb,
```

```
1346              sizeof(newfile_daily_minb));
1347              n = atoi(newfile_daily_minb);
1348              ee.writeByte(36, n);
1349              ee.writeByte(37, '1');
1350              ee.writeByte(46, '0');
1351            }
1352
1353            if (client_line.indexOf("reset_mcu=1") > -1)
1354            {
1355              digitalWrite(38, LOW);
1356              Alarm.delay(500);
1357              digitalWrite(38, HIGH);
1358            }
1359
1360            if (client_line.indexOf("delete_file") > -1)
1361            {
1362              int Pos_a = client_line.indexOf("delete_file=");
1363              int Pos_b = client_line.indexOf("&delete");
1364              Pos_a = Pos_a + 12;
1365              String dela =client_line.substring(Pos_a,Pos_b);
1366              char delb[dela.length()+1];
1367              dela.toCharArray(delb, sizeof(delb));
1368              file.remove(&root, delb);
1369            }
1370
1371            if (client_line.indexOf("Scan") > -1)
1372            {
1373              scan();
1374            }
1375
1376            if (client_line.indexOf("Create") > -1)
1377            {
1378              RepeatTask();
1379            }
1380          }
1381          goto selida;
1382        }
1383
1384        else if (strstr(clientline, "GET /") != 0) {
1385              char *filename;
1386
1387          filename = clientline + 5;
1388          (strstr(clientline, " HTTP"))[0] = 0;
1389
1390          lcd.clear();
```

```
1391            //lcd.setCursor(0, 0);
1392            //lcd.print("Main Program");
1393            lcd.setCursor(0, 1);
1394            lcd.print(filename);  // print the file we want
1395
1396
1397            if (! file.open(&root, filename, O_READ)) {
1398              client.println("HTTP/1.1 404 Not Found");
1399              client.println("Content-Type: text/html");
1400              client.println();
1401              client.println("<h2>File Not Found!(1)</h2>");
1402              break;
1403            }
1404
1405            lcd.clear();
1406            //lcd.setCursor(0, 0);
1407            //lcd.print("Main Program");
1408            lcd.setCursor(0, 1);
1409            lcd.print("Opened!");
1410
1411            client.println("HTTP/1.1 200 OK");
1412            client.println("Content-Type: text/plain");
1413            client.println();
1414
1415            int16_t c;
1416            while ((c = file.read()) > 0)
1417            {
1418              client.print((char)c);
1419            }
1420            file.close();
1421          }
1422          else
1423          {
1424            client.println("HTTP/1.1 404 ");  // everything else is a 404
1425            client.println("Content-Type: text/html");
1426            client.println();
1427            client.println("<h2>File Not Found!(2)</h2>");
1428          }
1429          break;
1430        }
1431      }
1432      Alarm.delay(10);    // give the web browser time to receive the data
1433      client.stop();
1434    }
1435    digitalWrite(7, HIGH);                    // watchdog
```

```
1436    Alarm.delay(1);                    // watchdog
1437    digitalWrite(7, LOW);              // watchdog
1438  }
```

Π.Β.2 Κώδικας Περιφερειακών Κόμβων

```
1   #include <XBee.h>
2   #include <Wire.h>
3   #include <OneWire.h>
4   #include <DallasTemperature.h>
5   #include <LiquidCrystal_I2C.h>
6
7   //LiquidCrystal_I2C lcd(0x20,16,2); //gia debbuging
8   XBee xbee = XBee();
9   uint8_t payload[] = { 0, 0, 0,0 };
10  XBeeResponse response = XBeeResponse();
11  // create reusable response objects for responses we expect to handle
12  ZBRxResponse rx = ZBRxResponse();
13  ModemStatusResponse msr = ModemStatusResponse();
14  XBeeAddress64 addr64 = XBeeAddress64(0x0013a200, 0x405D870F);
15  ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
16  ZBTxStatusResponse txStatus = ZBTxStatusResponse();
17
18  #define ONE_WIRE_BUS 2
19  OneWire oneWire(ONE_WIRE_BUS);
20  DallasTemperature sensors(&oneWire);
21  DeviceAddress insideThermometer;
22
23  char c;
24  long previousMillis = 0;
25  long currentMillis = 0;
26  boolean sleep = false;
27  long t=0;
28  char t1;
29  char t2;
30
31
32  void setup()
33  {
34    //////////Akolou8ei i arxikopoiisi tis lcd, ////////////////
35    /////////o kanonikos kombos den exei, mono gia debugging////
```

```
36    //lcd.init();
37    //lcd.backlight();
38    //lcd.print("Hello, world!");
39    //lcd.setCursor(7, 1);
40    //lcd.print("t=");
41    //lcd.setCursor(9, 1);
42    //lcd.print(t,DEC);
43
44    //lcd.setCursor(7, 0);
45    //lcd.print(t1,DEC);
46    //lcd.setCursor(11, 0);
47    //lcd.print(t2,DEC);
48
49    Wire.begin();                 // join i2c bus (address optional for master)
50
51    ////////////////////ENARKSI ARXIKOPOIISIS DS2782////////////////////////////
52
53    Wire.beginTransmission(52);
54    Wire.send(0x69);              // sends instruction byte
55    Wire.send(5);
56    Wire.endTransmission();
57    delay(50);
58
59    Wire.beginTransmission(52);
60    Wire.send(0x64);              // sends instruction byte
61    Wire.send(214);
62    Wire.endTransmission();
63    delay(50);
64
65    Wire.beginTransmission(52);
66    Wire.send(0x65);              // sends instruction byte
67    Wire.send(20);
68    Wire.endTransmission();
69    delay(50);
70
71    Wire.beginTransmission(52);
72    Wire.send(0x66);              // sends instruction byte
73    Wire.send(128);
74    Wire.endTransmission();
75    delay(50);
76
77    Wire.beginTransmission(52);
78    Wire.send(0x67);              // sends instruction byte
79    Wire.send(5);
80    Wire.endTransmission();
```

```
81      delay(50);
82
83      Wire.beginTransmission(52);
84      Wire.send(0x62);            // sends instruction byte
85      Wire.send(25);
86      Wire.endTransmission();
87      delay(50);
88
89      Wire.beginTransmission(52);
90      Wire.send(0x63);            // sends instruction byte
91      Wire.send(0);
92      Wire.endTransmission();
93      delay(50);
94
95      Wire.beginTransmission(52);
96      Wire.send(0x78);            // sends instruction byte
97      Wire.send(3);
98      Wire.endTransmission();
99      delay(50);
100
101     Wire.beginTransmission(52);
102     Wire.send(0x79);            // sends instruction byte
103     Wire.send(216);
104     Wire.endTransmission();
105     delay(50);
106
107     Wire.beginTransmission(52);
108     Wire.send(0x7a);            // sends instruction byte
109     Wire.send(2);
110     Wire.endTransmission();
111     delay(50);
112
113
114     Wire.beginTransmission(52);
115     Wire.send(0x61);            // sends instruction byte
116     Wire.send(0);
117     Wire.endTransmission();
118     delay(50);
119
120     Wire.beginTransmission(52);
121     Wire.send(0x10);            // sends instruction byte
122     Wire.send(25);
123     Wire.endTransmission();
124     delay(50);
125
```

```
126     Wire.beginTransmission(52);
127     Wire.send(0x11);              // sends instruction byte
128     Wire.send(0);
129     Wire.endTransmission();
130     delay(50);
131
132
133     ////////////////////TELOS ARXIKOPOIISIS DS2782////////////////////////////
134
135
136     sensors.begin();
137     xbee.begin(9600);
138     delay(5000);
139     sensors.getAddress(insideThermometer, 0);
140     sensors.setResolution(insideThermometer, 12);
141
142     pinMode(A1, OUTPUT);                        //xbee sleep
143     digitalWrite(A1, LOW);
144     pinMode(4, OUTPUT);
145     digitalWrite(4, HIGH);
146
147     pinMode(3, OUTPUT);                          // watchdog
148     digitalWrite(3, HIGH);                       // watchdog
149     delay(1);                                    // watchdog
150     digitalWrite(3, LOW);                        // watchdog
151     lcd.clear();
152   }
153
154   void loop()
155   {
156     digitalWrite(3, HIGH);                       // watchdog
157     delay(1);                                    // watchdog
158     digitalWrite(3, LOW);                        // watchdog
159
160     currentMillis=millis();
161
162     while (t==0 || sleep==false)
163     {
164       //lcd.setCursor(0, 0);                     //mono gia debugging
165       //long xronos = 0;                         //mono gia debugging
166       //xronos=currentMillis - previousMillis;   //mono gia debugging
167       //lcd.print(xronos,DEC);                   //mono gia debugging
168       //lcd.setCursor(0, 1);                     //mono gia debugging
169       //lcd.print("WAKE");                       //mono gia debugging
170       //lcd.setCursor(7, 1);                     //mono gia debugging
```

```
171     //lcd.print("t=");                          //mono gia debugging
172     //lcd.setCursor(9, 1);                      //mono gia debugging
173     //lcd.print(t,DEC);                         //mono gia debugging
174
175     digitalWrite(3, HIGH);                    // watchdog
176     delay(1);                                 // watchdog
177     digitalWrite(3, LOW);                     // watchdog
178
179     xbee.readPacket();
180     if (xbee.getResponse().isAvailable())
181     {      // got something
182       if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
183       {        // got a zb rx packet
184         xbee.getResponse().getZBRxResponse(rx); // now fill our zb rx class
185         c=rx.getData(0);
186         t1=rx.getData(1);
187         t2=rx.getData(2);
188         //lcd.setCursor(7, 0);                  //mono gia debugging
189         //lcd.print(c,HEX);                     //mono gia debugging
190         //lcd.setCursor(10, 0);                 //mono gia debugging
191         //lcd.print(t1,HEX);                    //mono gia debugging
192         //lcd.setCursor(13, 0);                 //mono gia debugging
193         //lcd.print(t2,HEX);                    //mono gia debugging
194         t=t1;
195         t=t<<8;
196         t |= t2;
197         t=t-10;
198         t=t*1000;
199       }
200       else if (xbee.getResponse().getApiId() == MODEM_STATUS_RESPONSE)
201       {
202         xbee.getResponse().getModemStatusResponse(msr);
203         // the local XBee sends this response
204         //on certain events, like association/dissociation
205       }
206
207       if (c==124)
208       {
209         sensors.requestTemperatures();
210         int q=entoli(0x06);
211         float c=sensors.getTempC(insideThermometer);
212         int d=sensors.getTempC(insideThermometer);
213         int e=(c-d)*100;
214         payload[0] = d;
215         payload[1] = ('.');
```

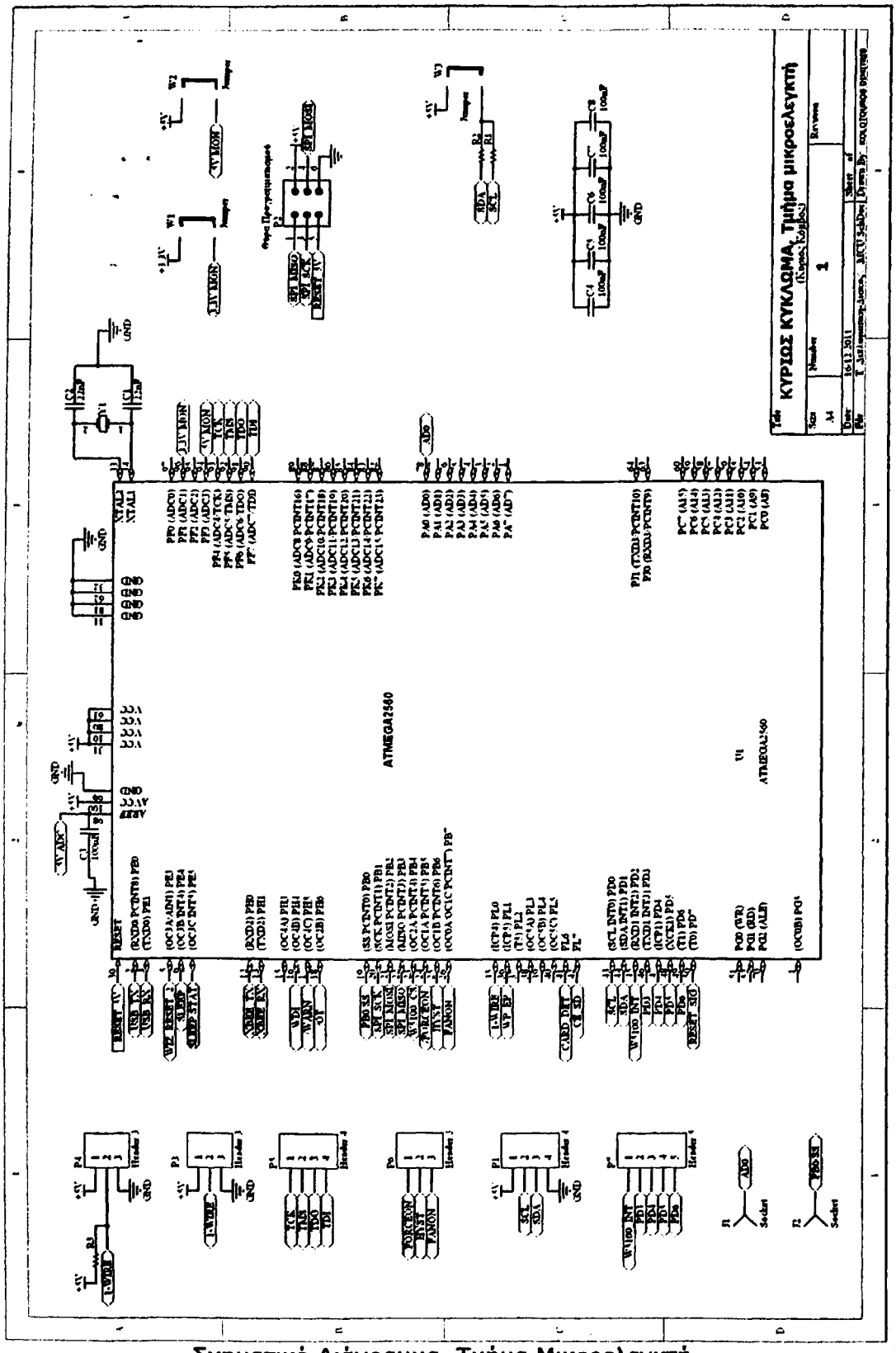
```
216         payload[2] = e;
217         payload[3] = q;   //apo ds2782 %mpataria
218         xbee.send(zbTx);
219         if (xbee.readPacket(500))
220         {     // got a response!
221           if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE)
222           // should be a znet tx status
223           {
224             xbee.getResponse().getZBTxStatusResponse(txStatus);
225             // get the delivery status, the fifth byte
226             delay(5000);
227             digitalWrite(A1, HIGH);  //GO TO SLEEP
228             digitalWrite(4, LOW);
229             sleep = true;
230           }
231         }
232       }
233     }
234     currentMillis=millis();
235   }
236
237   previousMillis=currentMillis;
238   currentMillis=millis();
239
240
241   if (sleep==true)
242   {
243     while ((currentMillis-previousMillis)<=t)
244     {
245       //lcd.setCursor(0, 0);                          //mono gia debugging
246       //long xronos = 0;                              //mono gia debugging
247       //xronos=currentMillis - previousMillis;        //mono gia debugging
248       //lcd.print(xronos,DEC);                        //mono gia debugging
249       //lcd.setCursor(0, 1);                          //mono gia debugging
250       //lcd.print("SLEEP");                           //mono gia debugging
251       //lcd.setCursor(7, 1);                          //mono gia debugging
252       //lcd.print("t=");                              //mono gia debugging
253       //lcd.setCursor(9, 1);                          //mono gia debugging
254       //lcd.print(t,DEC);                             //mono gia debugging
255
256       digitalWrite(3, HIGH);                          // watchdog
257       delay(1);                                       // watchdog
258       digitalWrite(3, LOW);                           // watchdog
259
260       currentMillis=millis();
```

```
261       }
262     }
263
264     previousMillis=currentMillis;
265     digitalWrite(A1, LOW);  //WAKE
266     digitalWrite(4, HIGH);
267     sleep = false;
268   }
269
270   int entoli(int w)
271   {
272     Wire.beginTransmission(52);
273     delay(1);
274     Wire.send(w);
275     delay(1);
276     Wire.endTransmission();
277.    delay(1);
278     Wire.requestFrom(52, 1);
279     delay(1);
280     int apotelesma = Wire.receive();
281     return apotelesma;
282   }
```

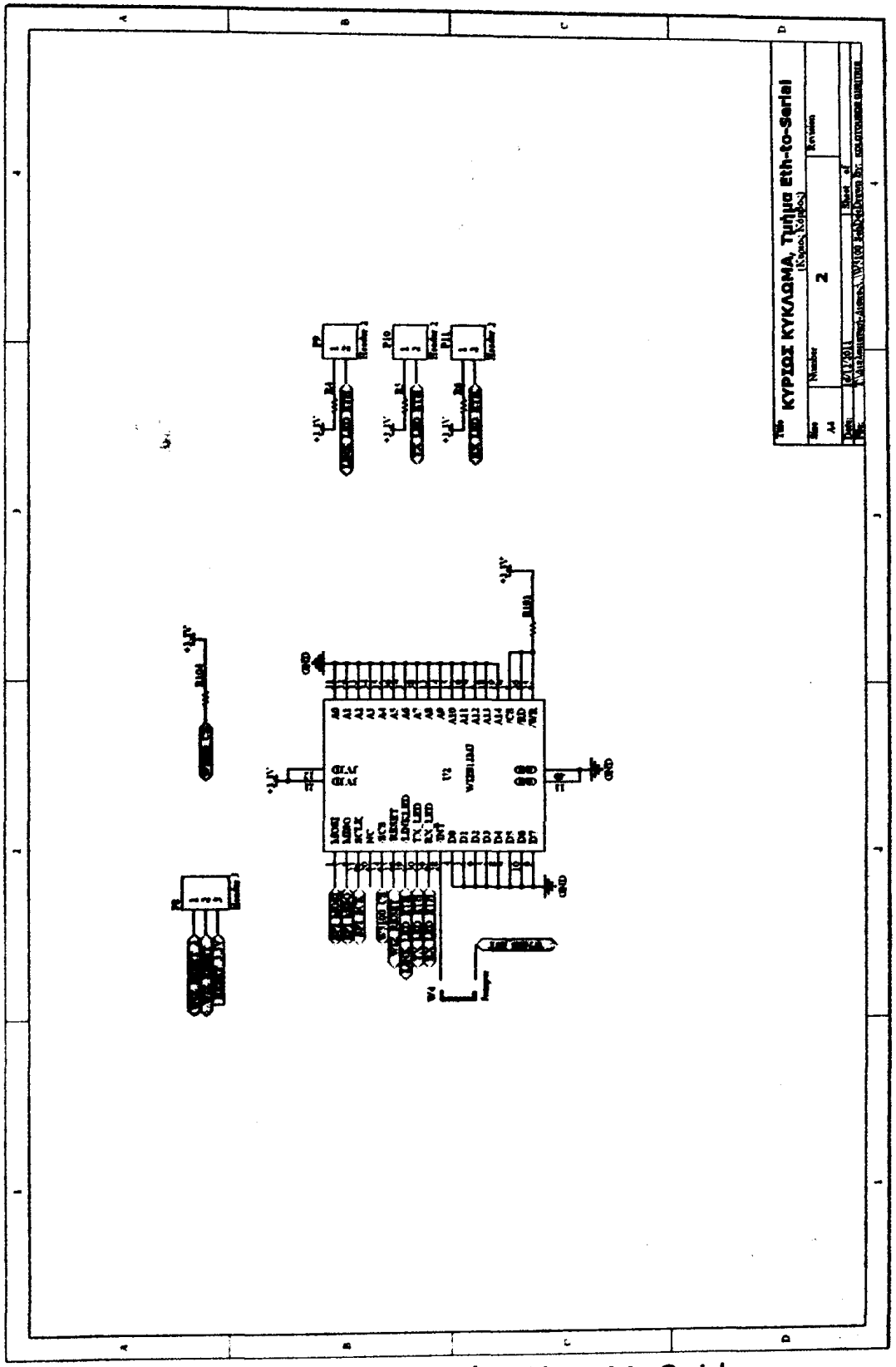
Παράρτημα Γ: Σχέδια Τυπωμένων Κυκλωμάτων

Π.Γ.1 Σχέδια Κύριου Κόμβου

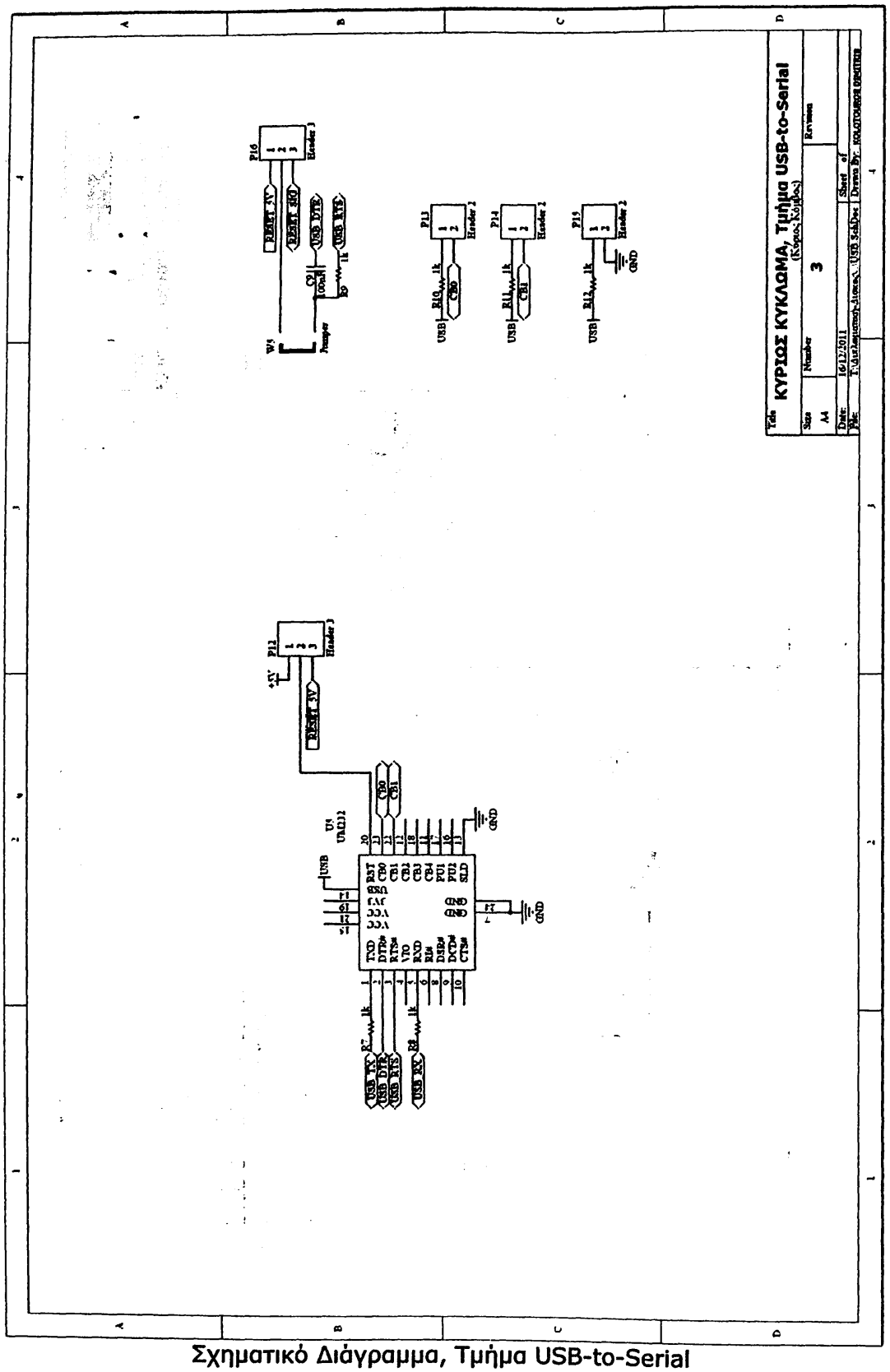
Π.Γ.1.1 Κυρίως Κύκλωμα



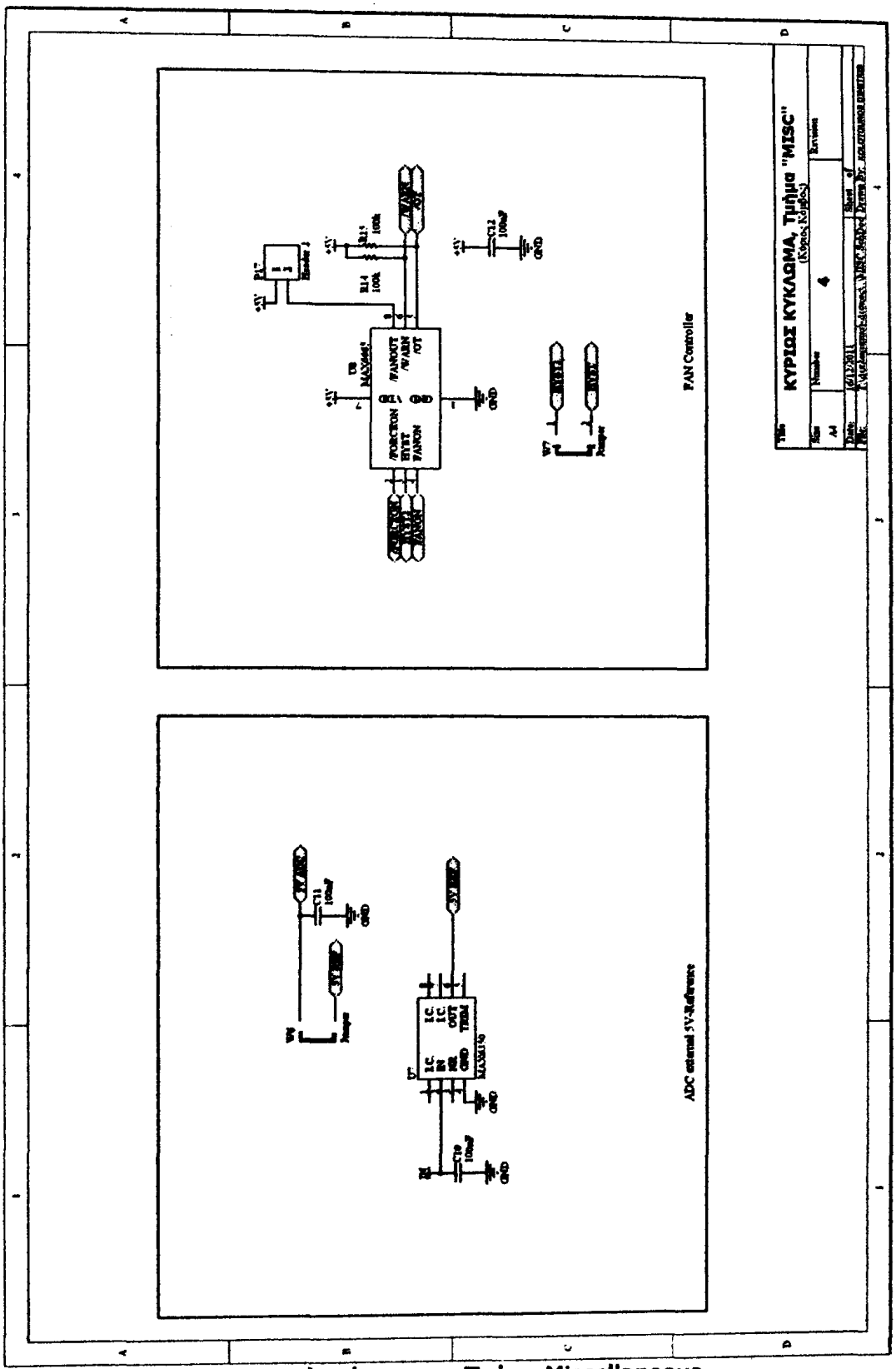
Σχηματικό Διάγραμμα, Τμήμα Μικροελεγκτή



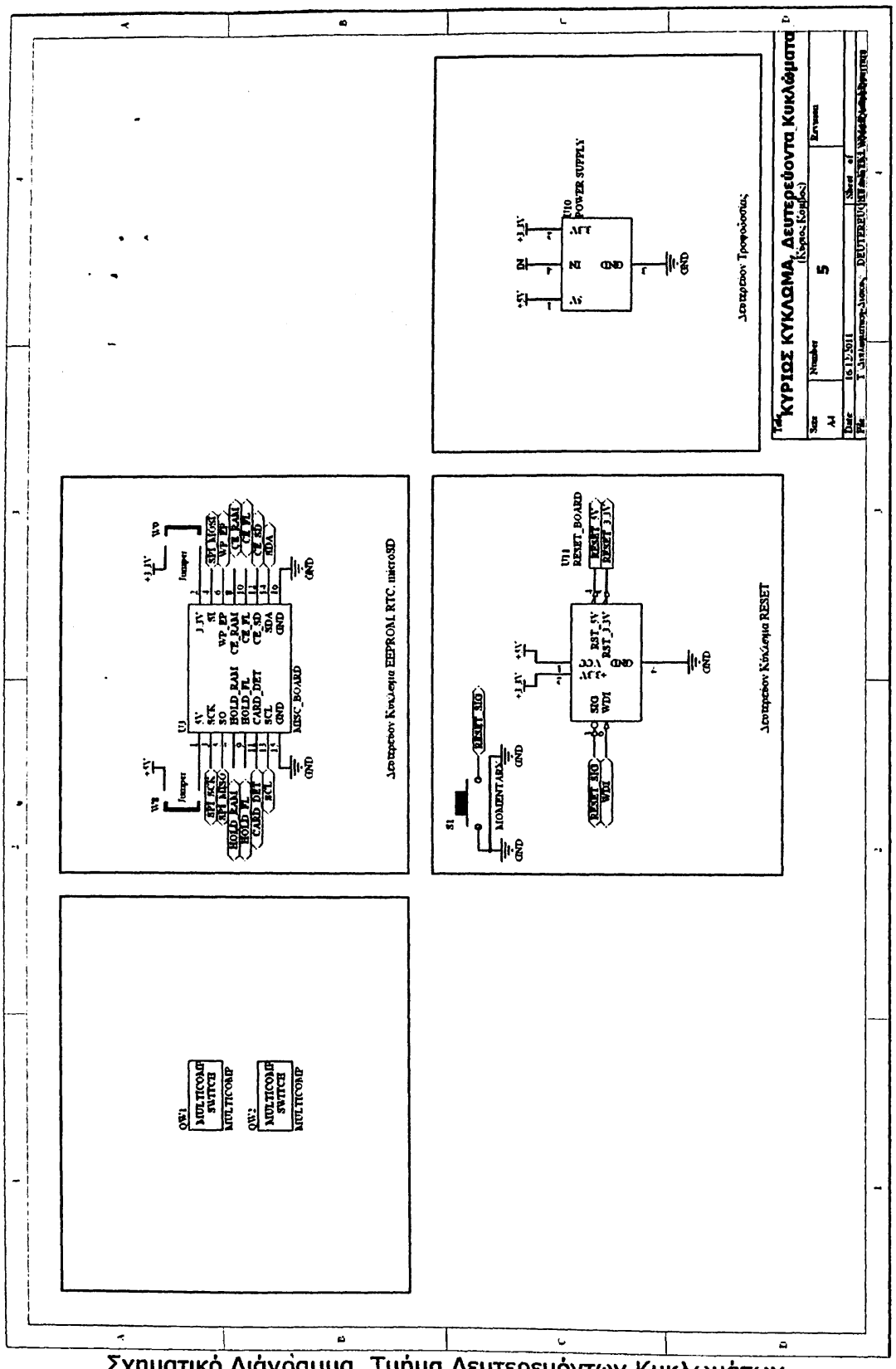
Σχηματικό Διάγραμμα, Τμήμα Ethernet-to-Serial



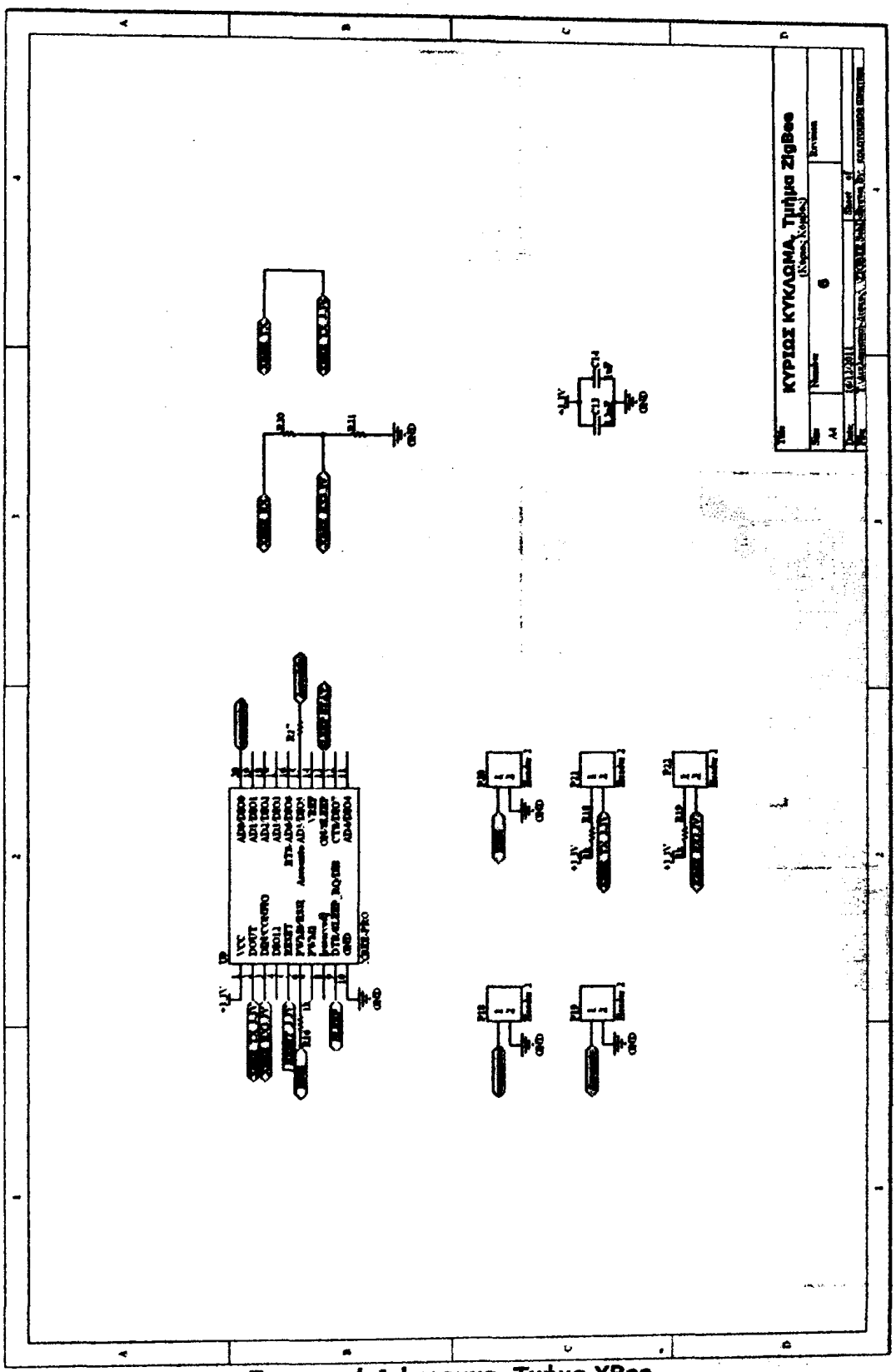
Σχηματικό Διάγραμμα, Τμήμα USB-to-Serial



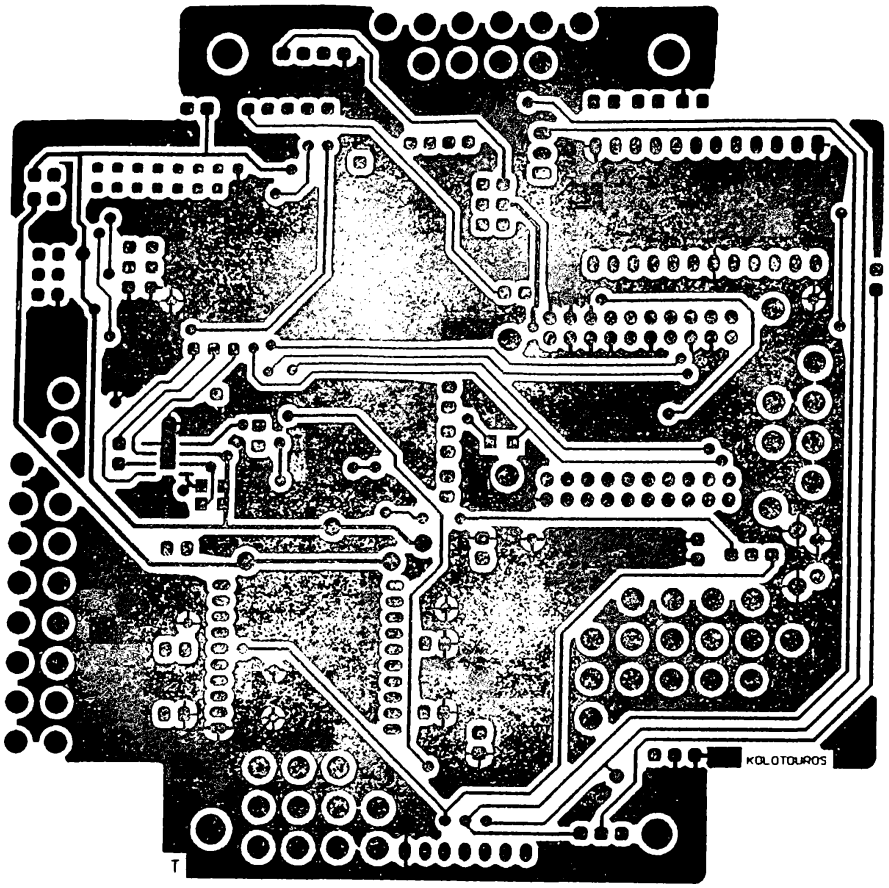
Σχηματικό Διάγραμμα, Τμήμα Miscellaneous



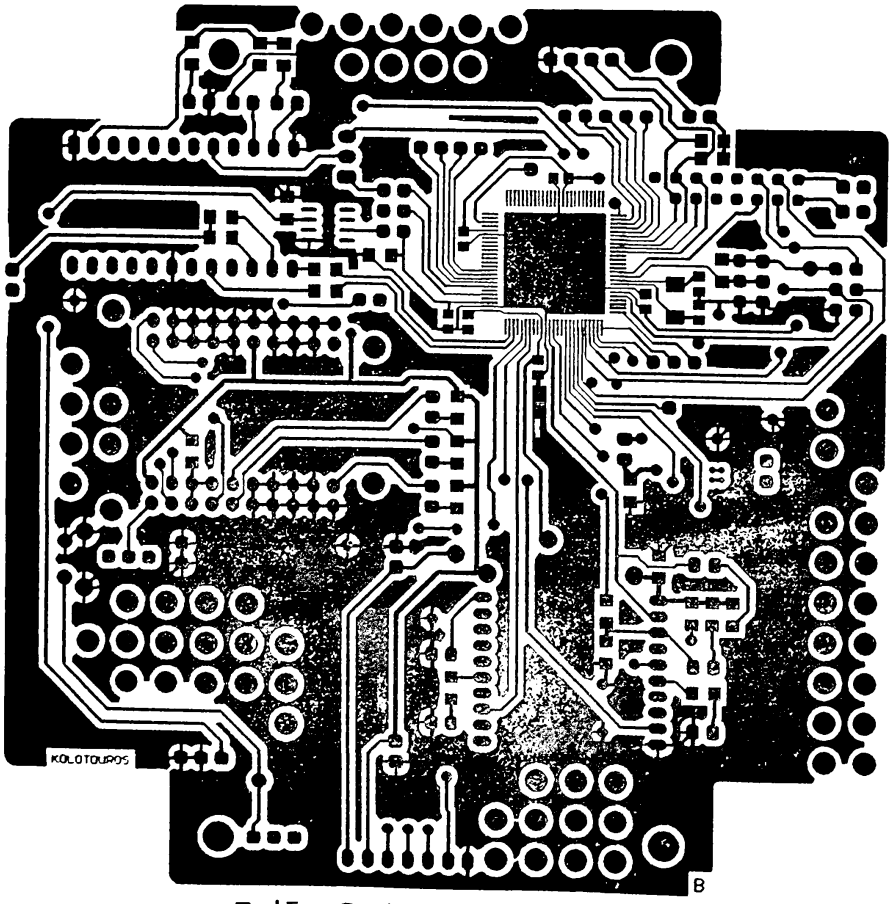
Σχηματικό Διάγραμμα, Τμήμα Δευτερευόντων Κυκλωμάτων



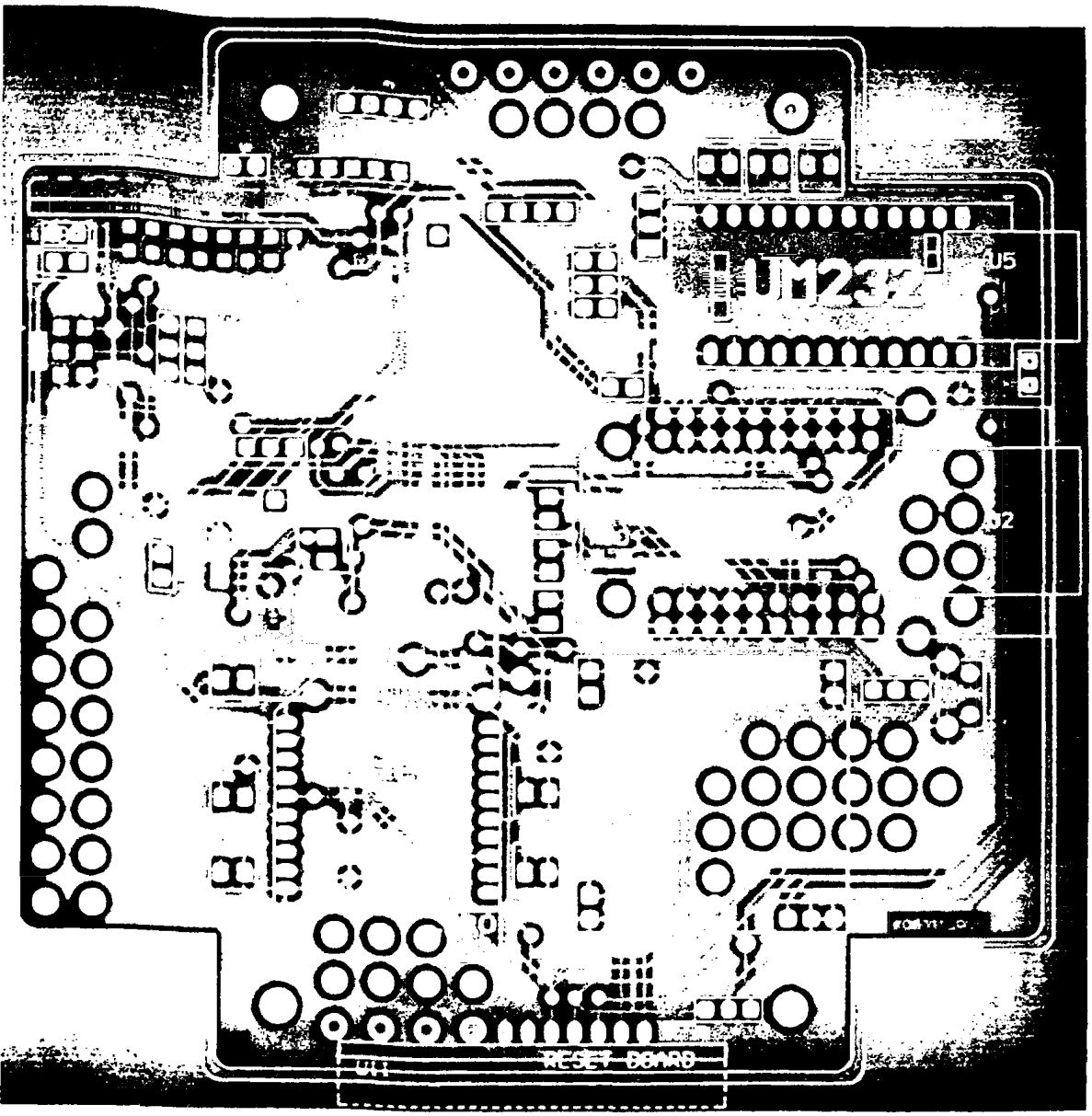
Σχηματικό Διάγραμμα, Τμήμα XBee



Σχέδιο Gerber, Top Layer



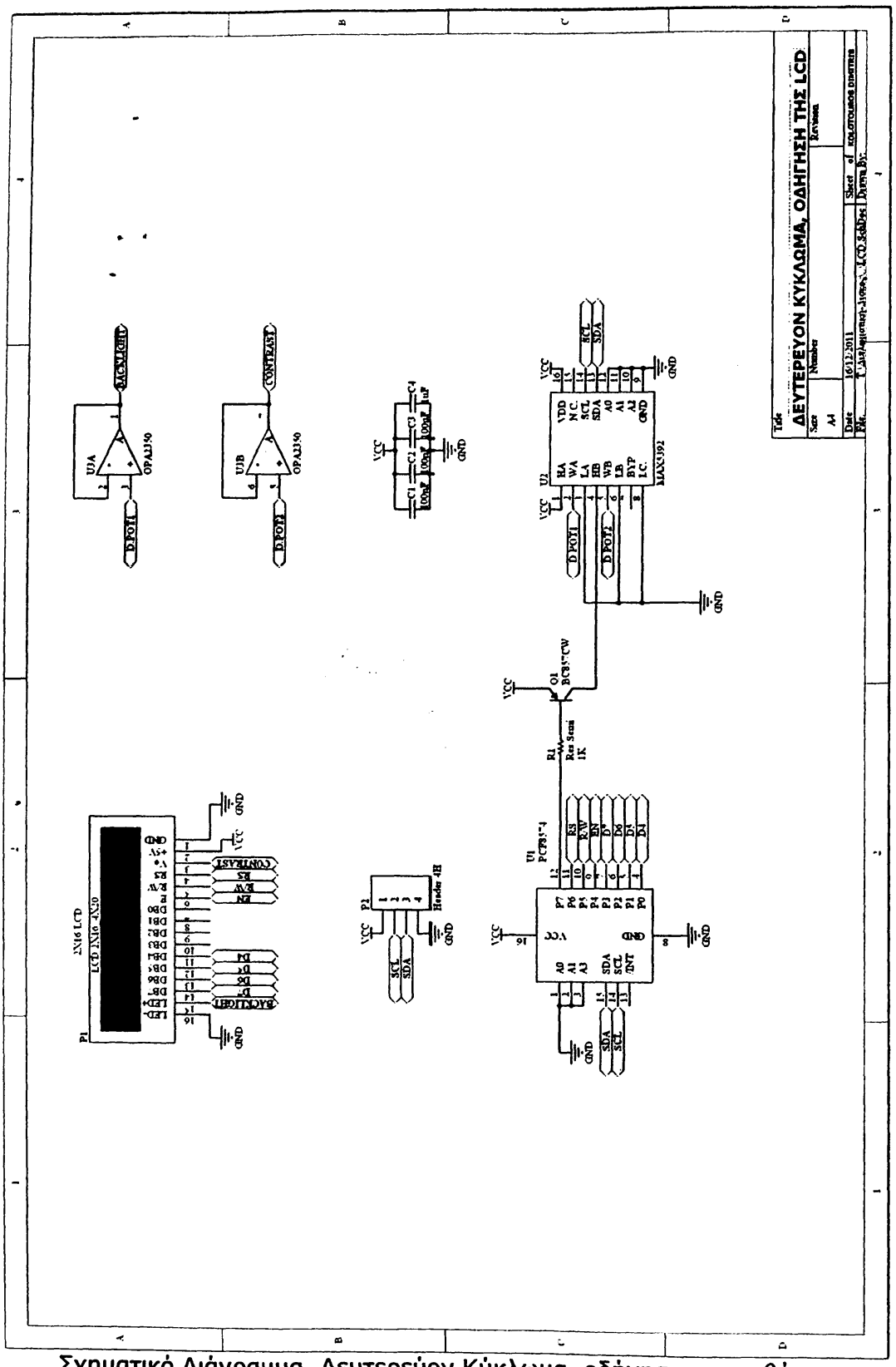
Σχέδιο Gerber, Bottom Layer



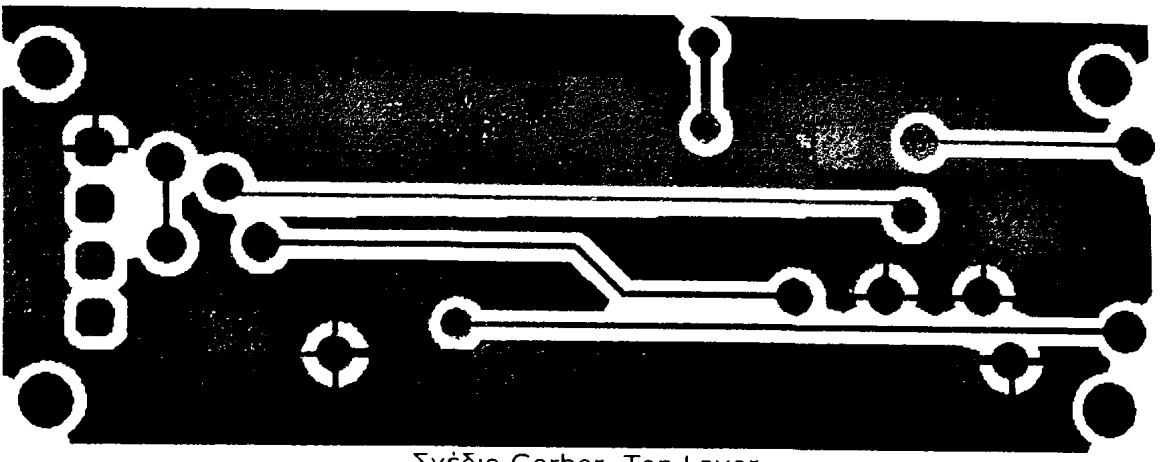
Layout Κυκλώματος με όλα τα layer ορατά
(Top, Top Overlay, Bottom, Bottom Overlay, Keep-Out, Multilayer)

Π.Γ.1.2 Δευτερεύοντα Κυκλώματα

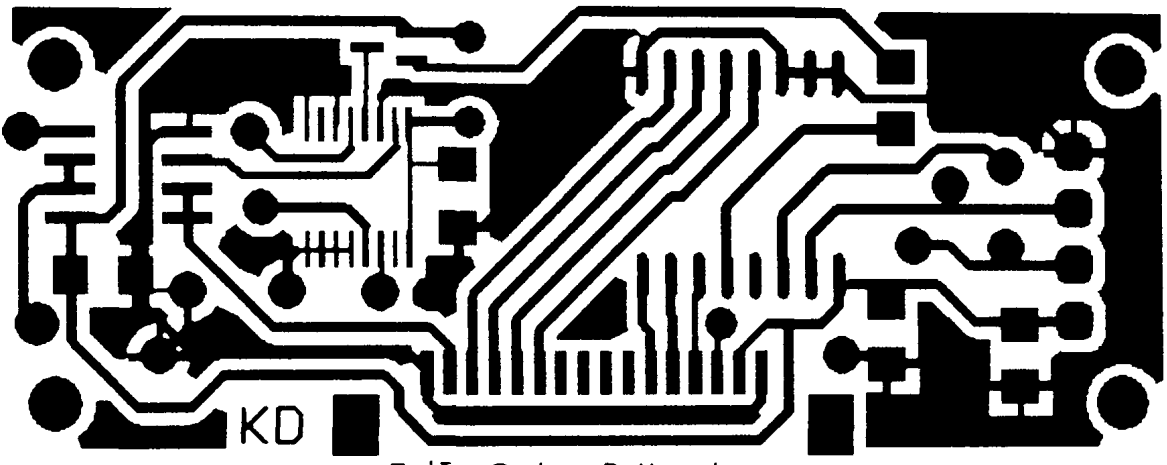
Π.Γ.1.2.1 Κύκλωμα οδήγησης της οθόνης



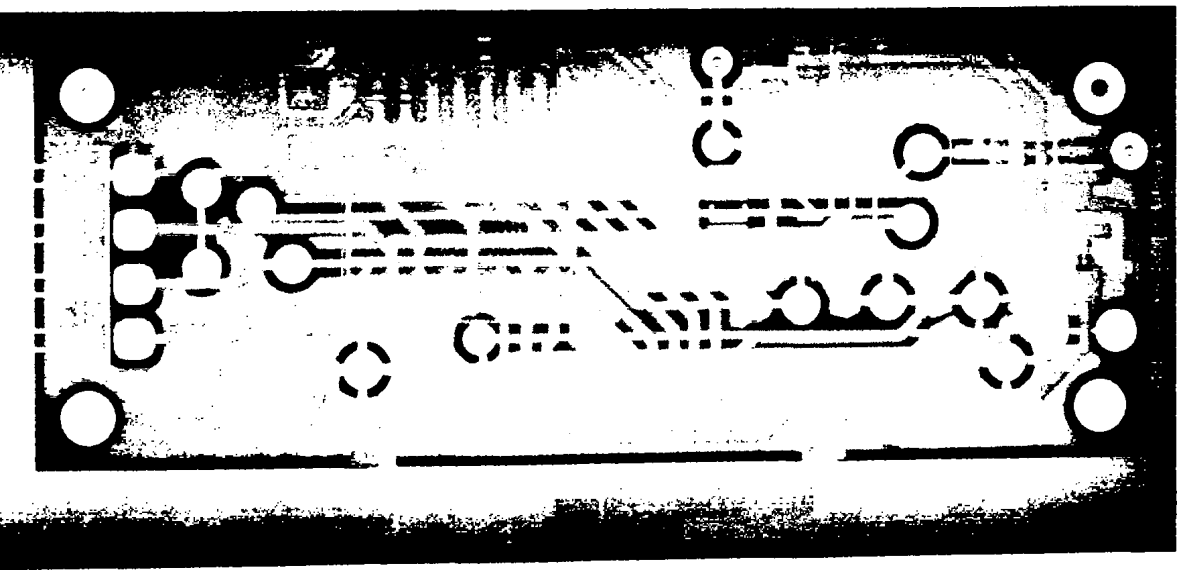
Σχηματικό Διάγραμμα, Δευτερεύον Κύκλωμα, οδήγησης της οθόνης



Σχέδιο Gerber, Top Layer



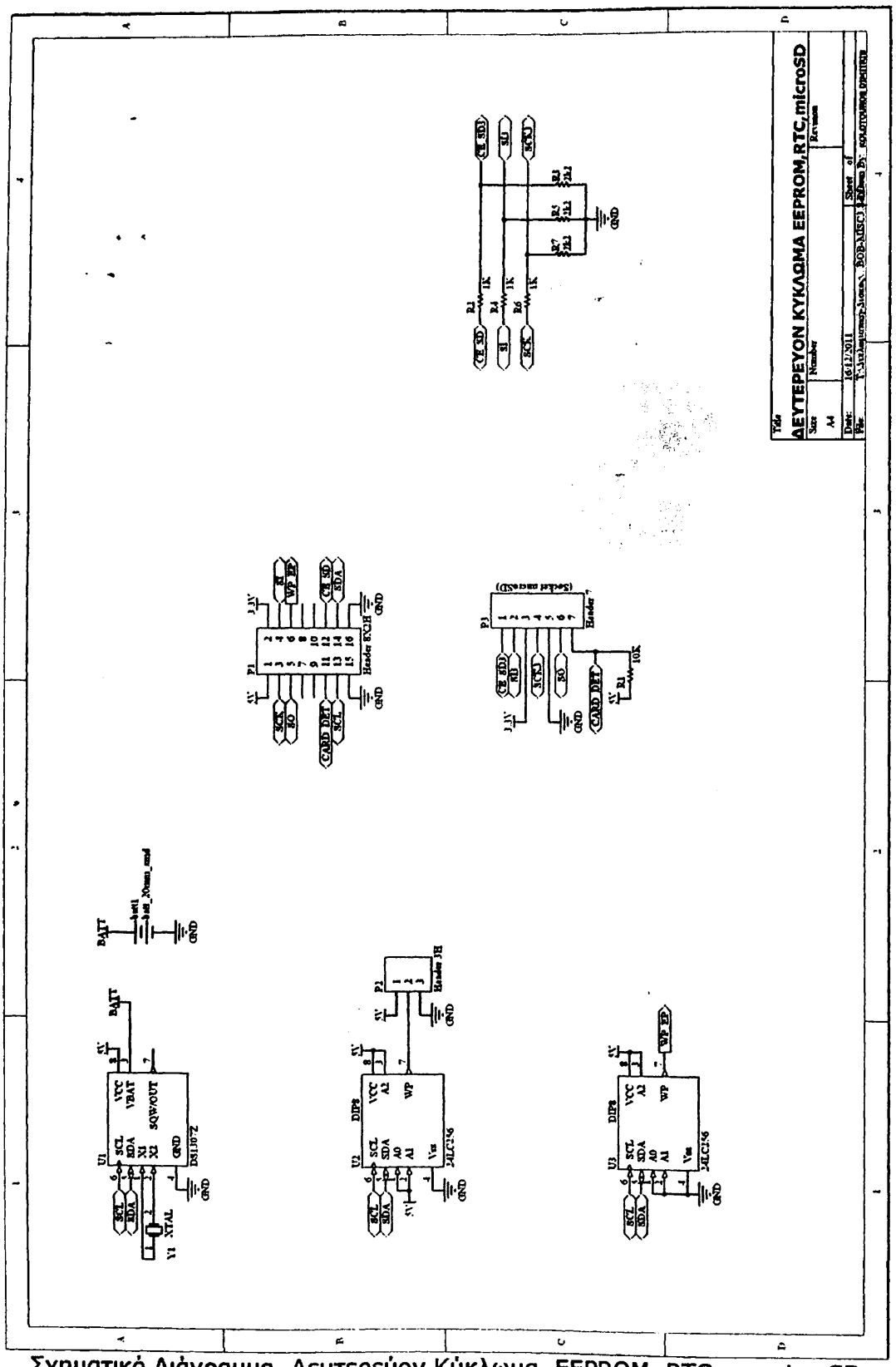
Σχέδιο Gerber, Bottom Layer



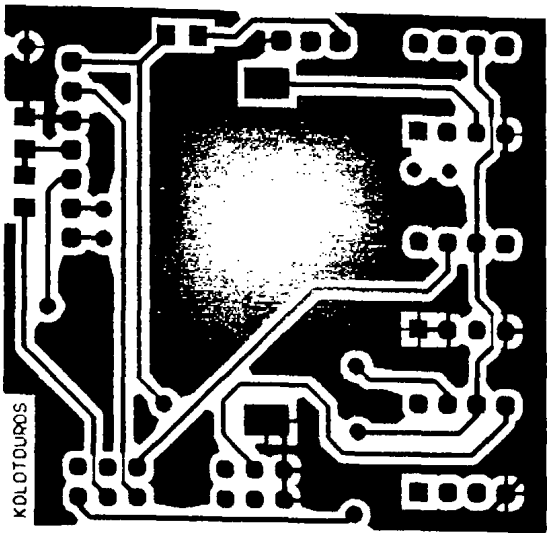
Layout Κυκλώματος με όλα τα layer ορατά

(Top, Top Overlay, Bottom, Bottom Overlay, Keep-Out, Multilayer)

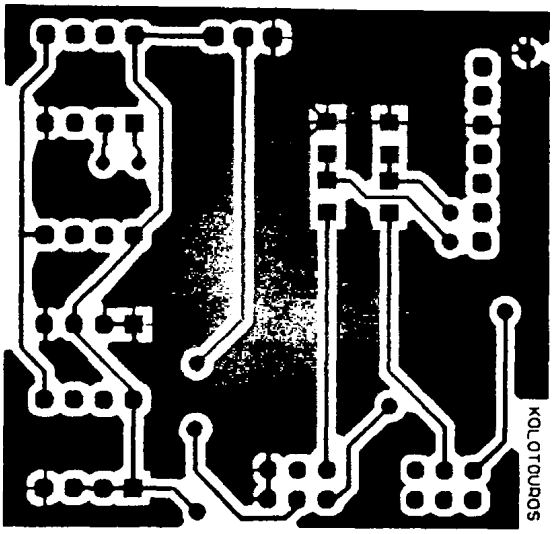
Π.Γ.1.2.2 Κύκλωμα EEPROM, RTC και microSD



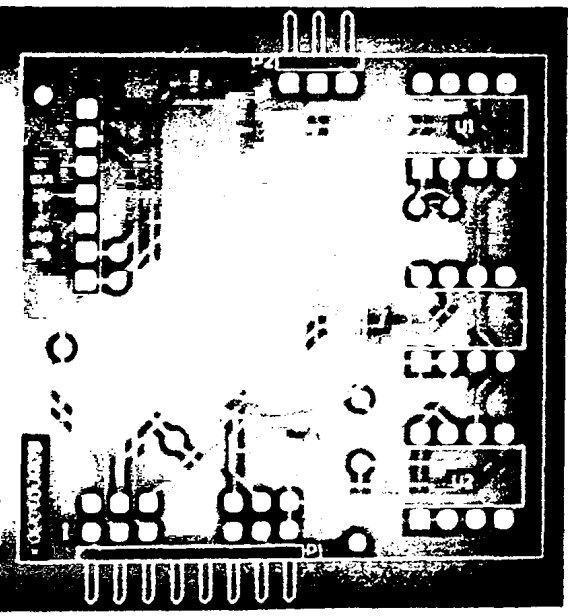
Σχηματικό Διάγραμμα, Δευτερεύον Κύκλωμα, EEPROM, RTC και microSD



Σχέδιο Gerber, Top Layer



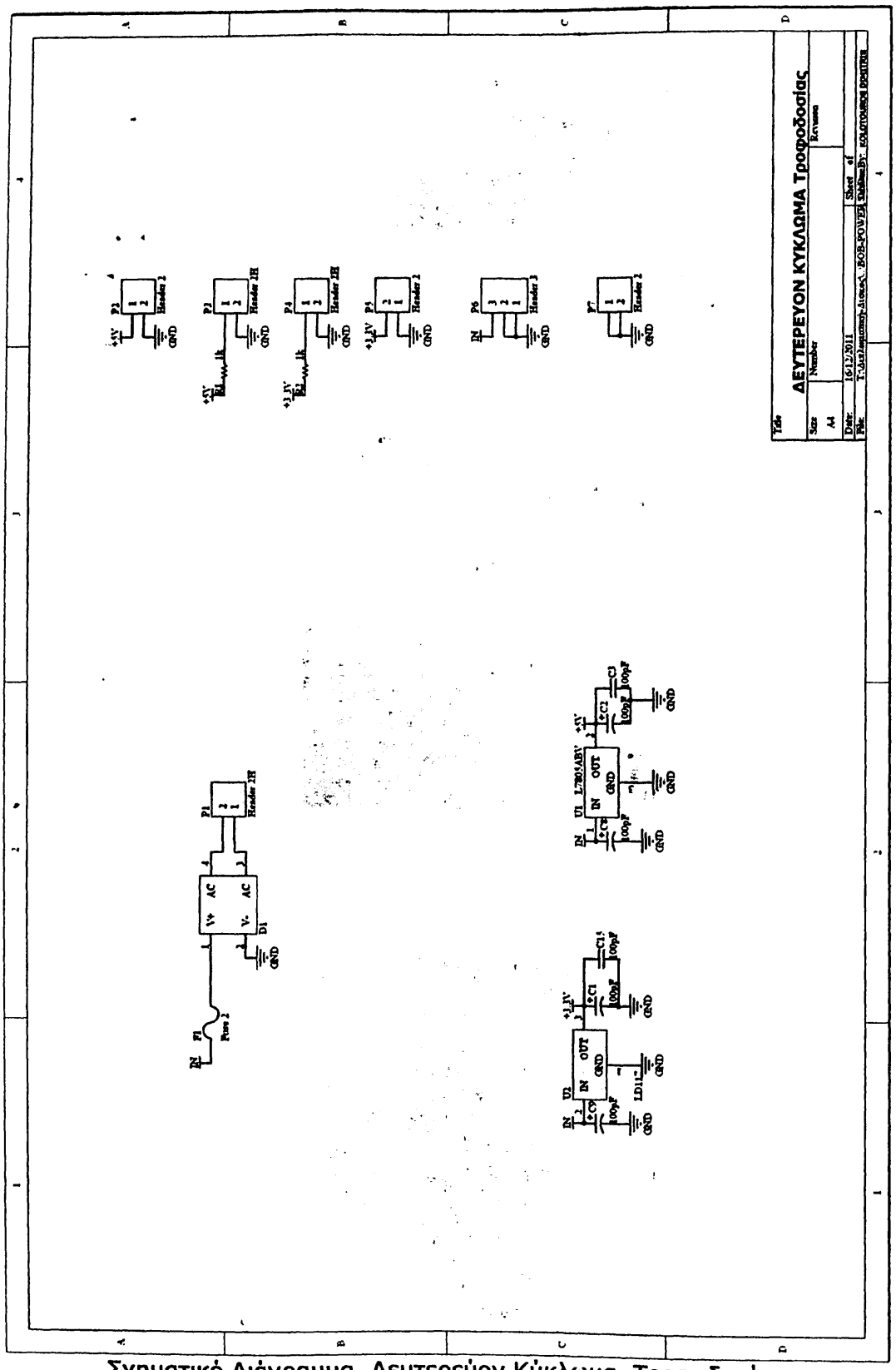
Σχέδιο Gerber, Bottom Layer



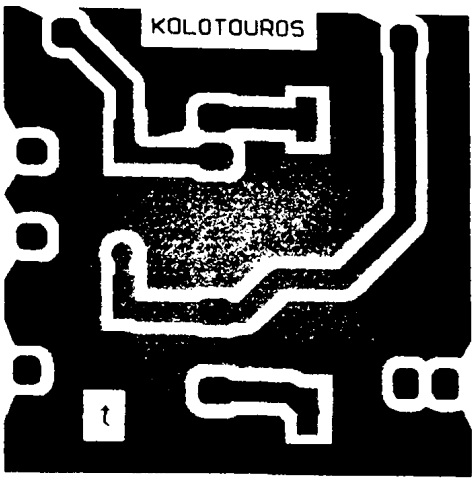
ayout Κυκλώματος με όλα τα layer ορατά

erlay, Bottom, Bottom Overlay, Keep-Out, Multilayer)

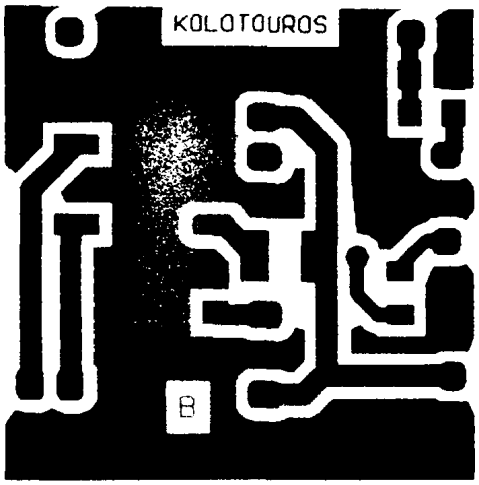
Π.Γ.1.2.3 Κύκλωμα Τροφοδοσίας



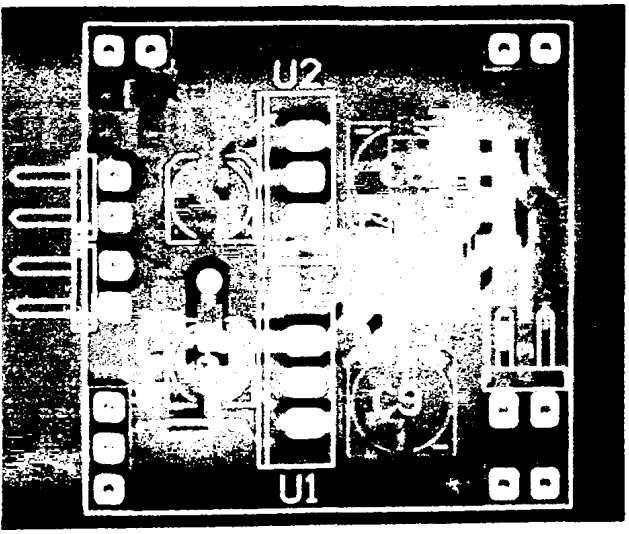
Σχηματικό Διάγραμμα, Δευτερεύον Κύκλωμα, Τροφοδοσίας



Σχέδιο Gerber, Top Layer

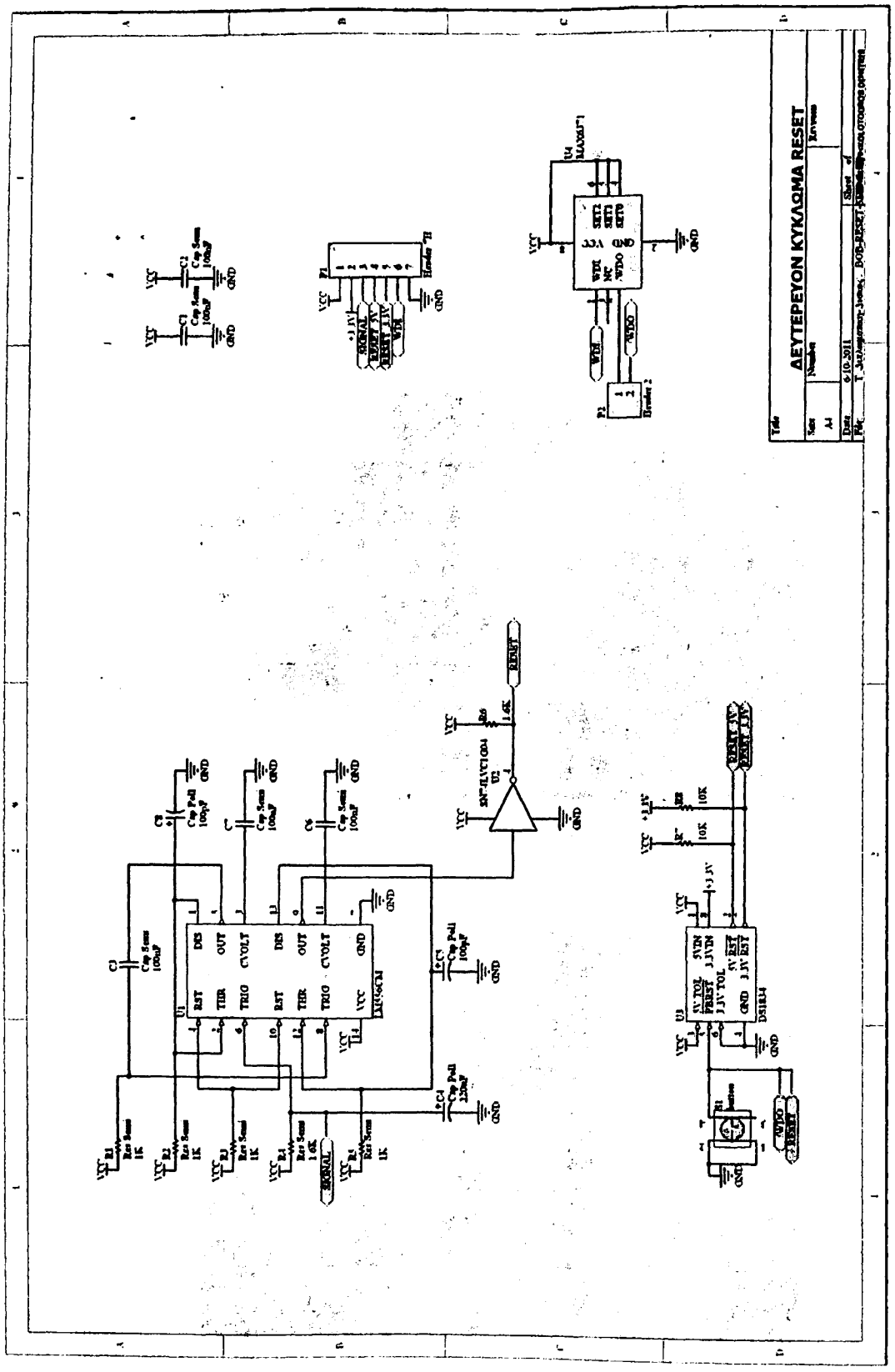


Σχέδιο Gerber, Bottom Layer

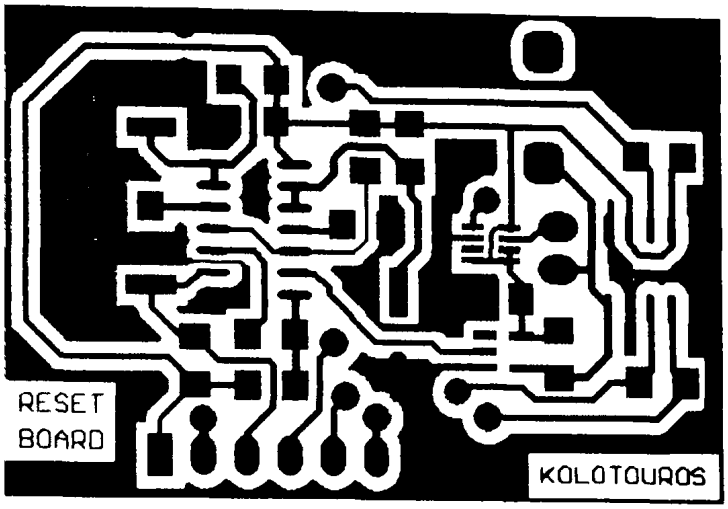


Layout Κυκλώματος με όλα τα layer ορατά
Overlay, Bottom, Bottom Overlay, Keep-Out, Multilayer)

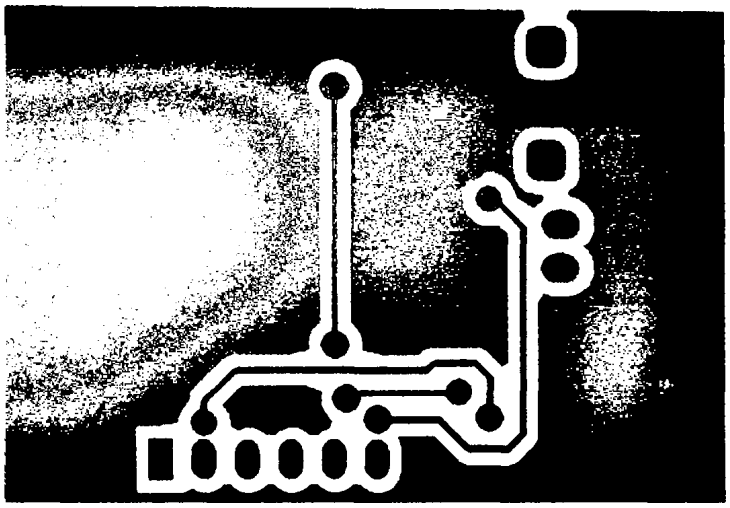
Π.Γ.1.2.4 Κύκλωμα RESET



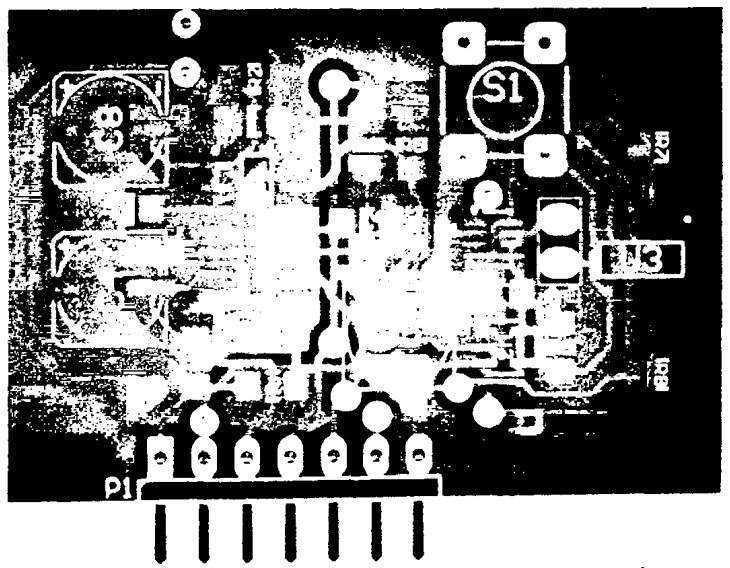
Σχηματικό Διάγραμμα, Δευτερεύον Κύκλωμα, RESET



Σχέδιο Gerber, Top Layer



Σχέδιο Gerber, Bottom Layer

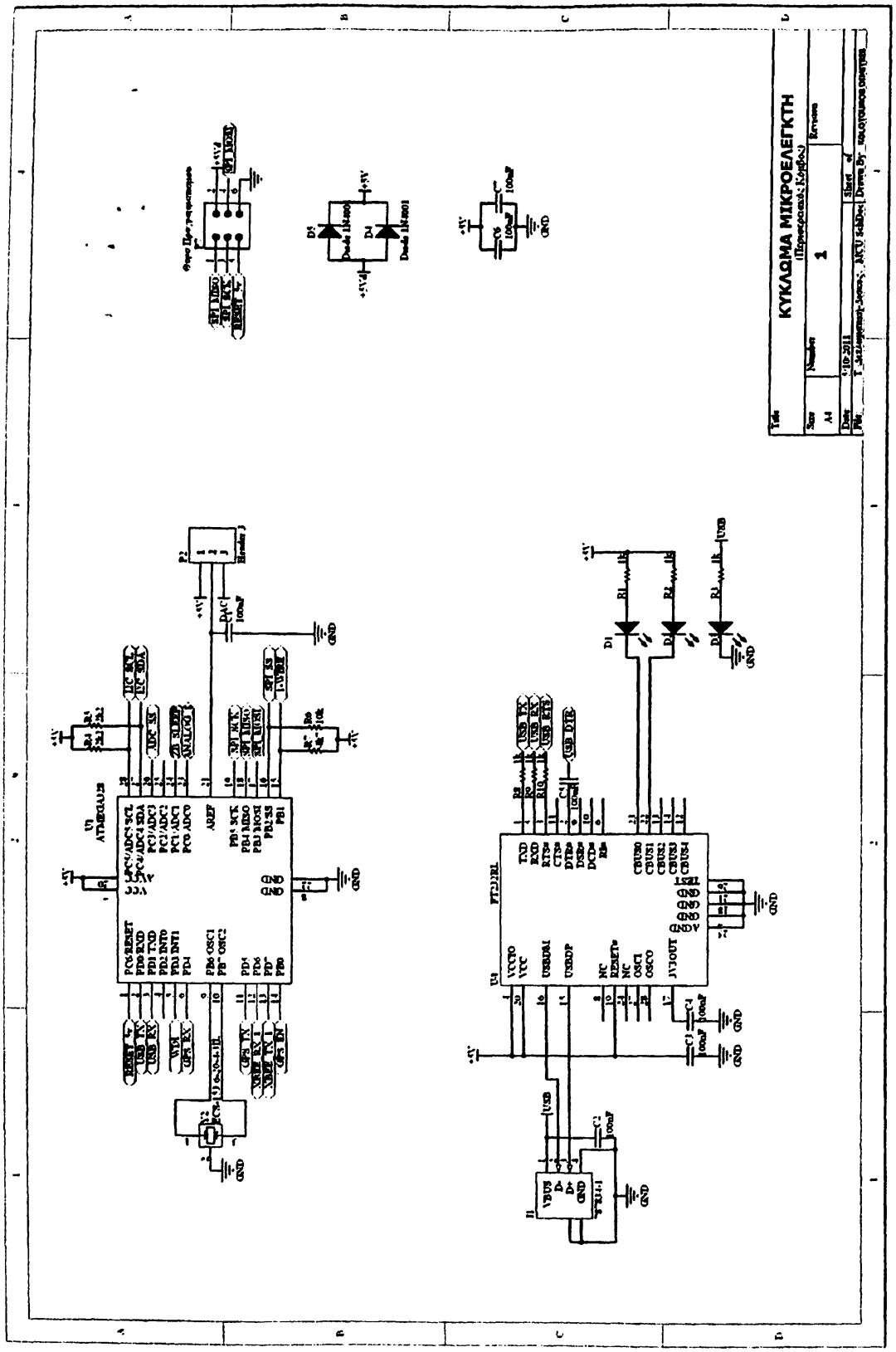


Layout Κυκλώματος με όλα τα layer ορατά

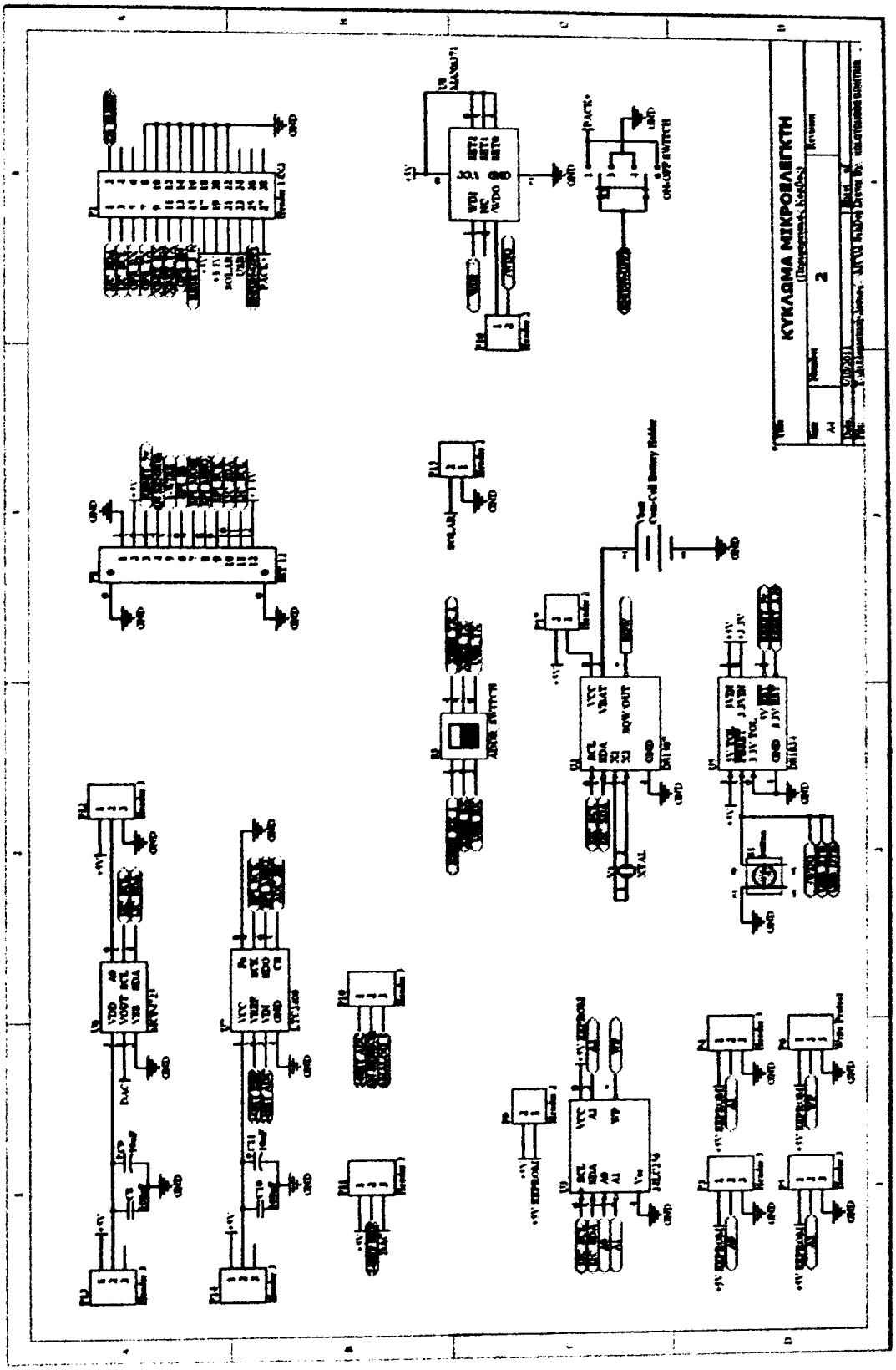
op Overlay, Bottom, Bottom Overlay, Keep-Out, Multilayer)

Π.Γ.2 Σχέδια Περιφερειακού Κόμβου

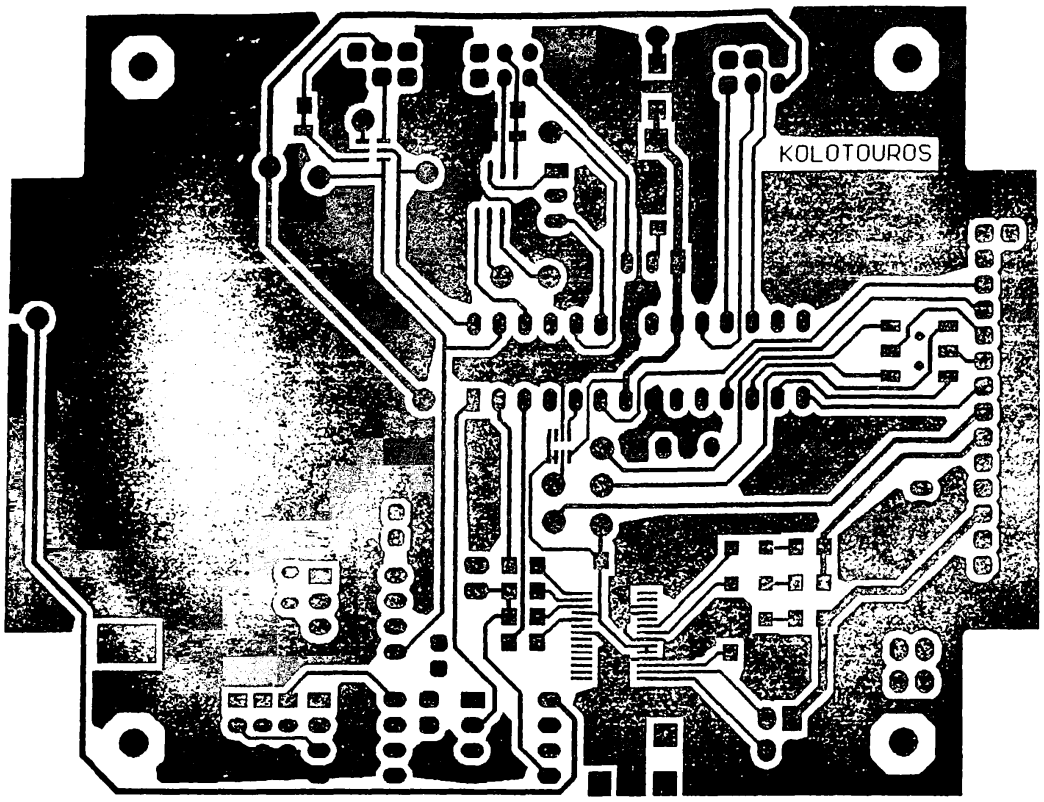
Π.Γ.2.1 Κύκλωμα Μικροελεγκτή



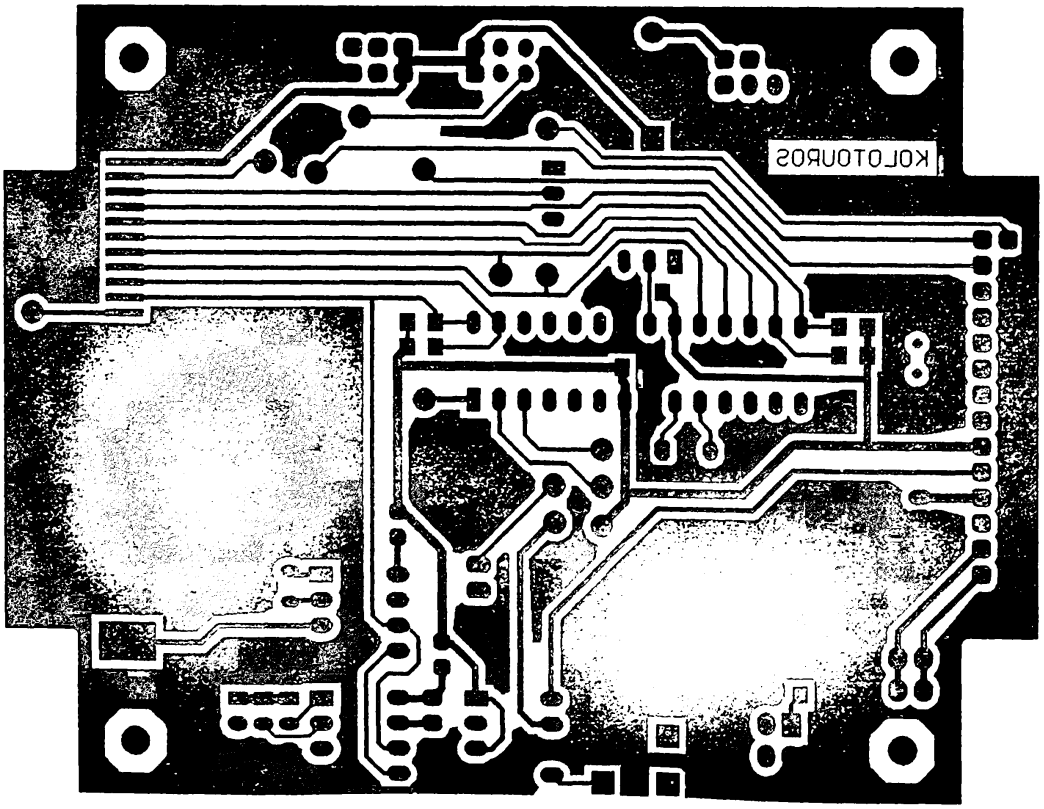
Σχηματικό Διάγραμμα, Κύκλωμα Μικροελεγκτή



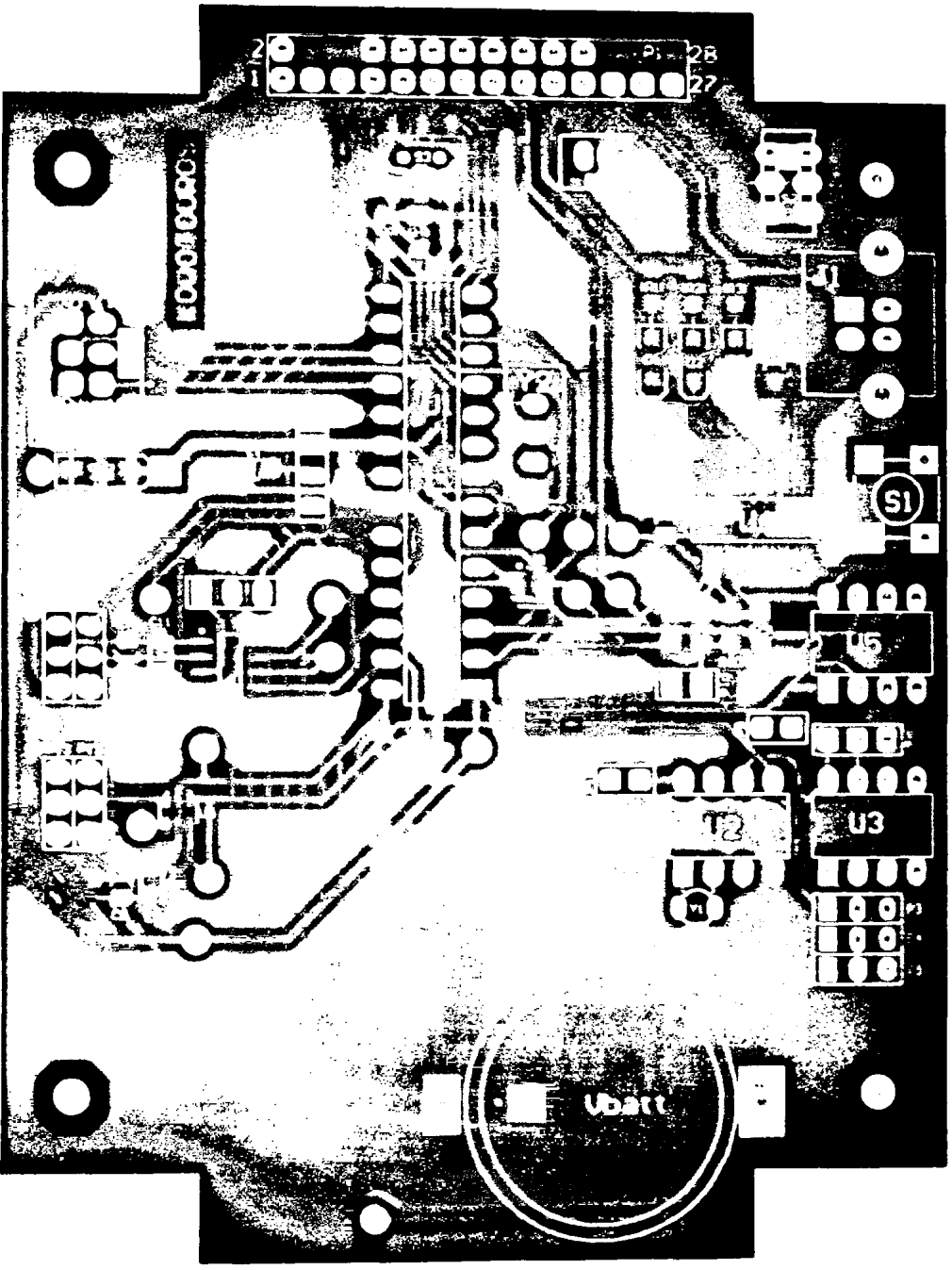
Σχηματικό Διάγραμμα, Κύκλωμα Μικροελεγκτή (Περιφερειακά)



Σχέδιο Gerber, Top Layer

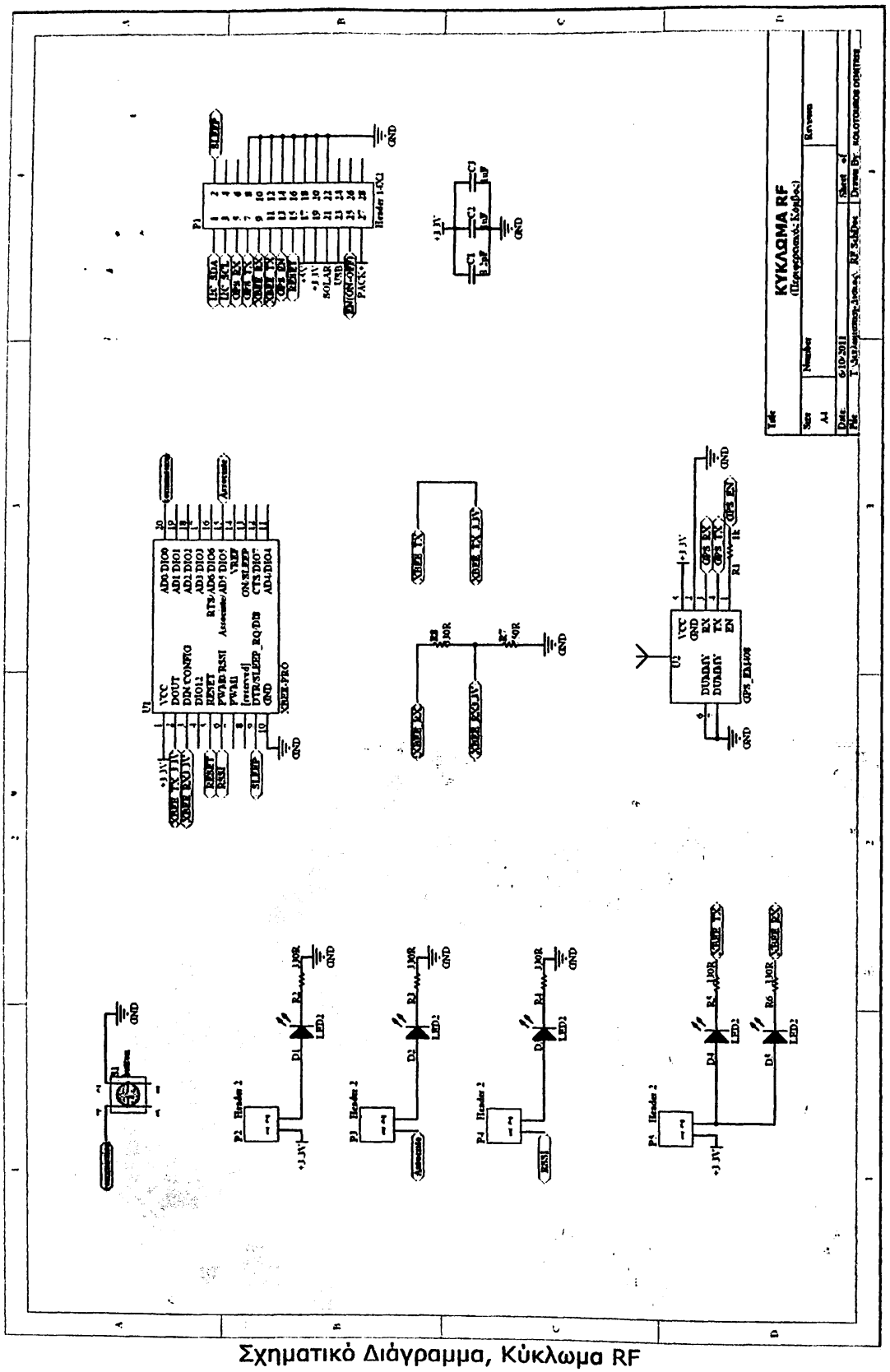


Σχέδιο Gerber, Bottom Layer

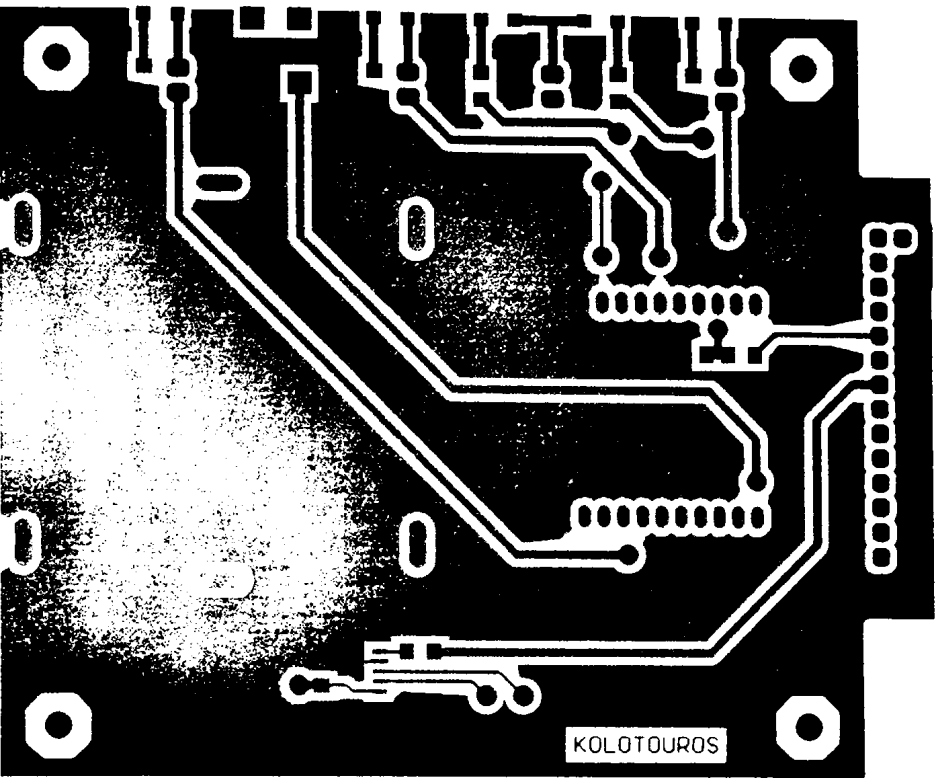


Layout Κυκλώματος με όλα τα layer ορατά
(Top, Top Overlay, Bottom, Bottom Overlay, Keep-Out, Multilayer)

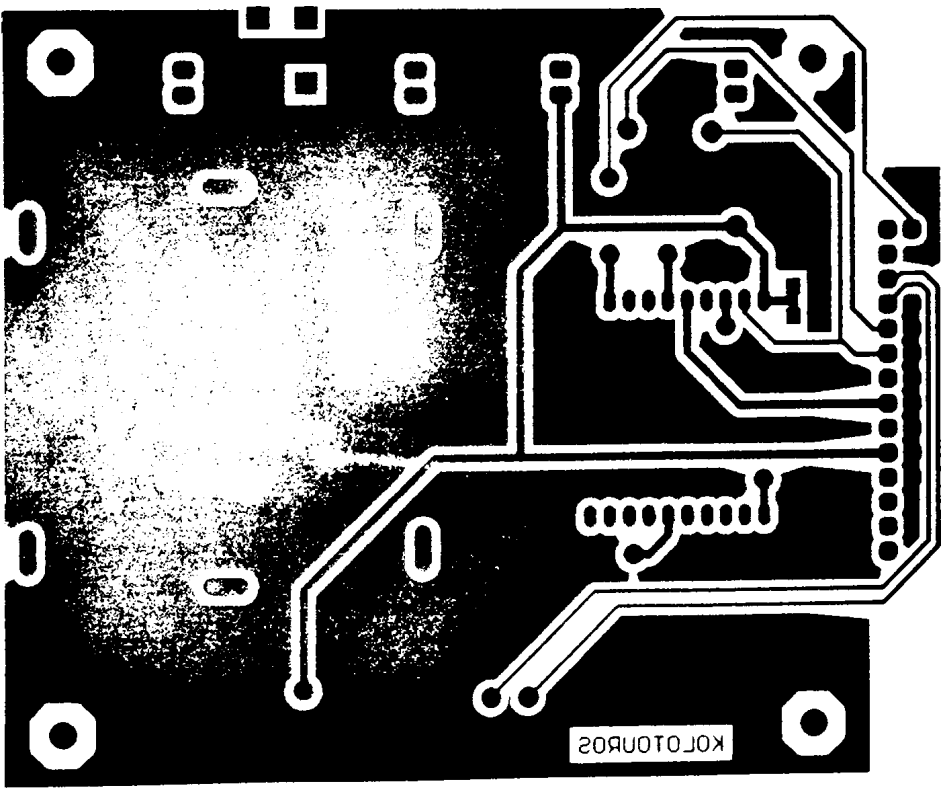
Π.Γ.2.2 Κύκλωμα RF



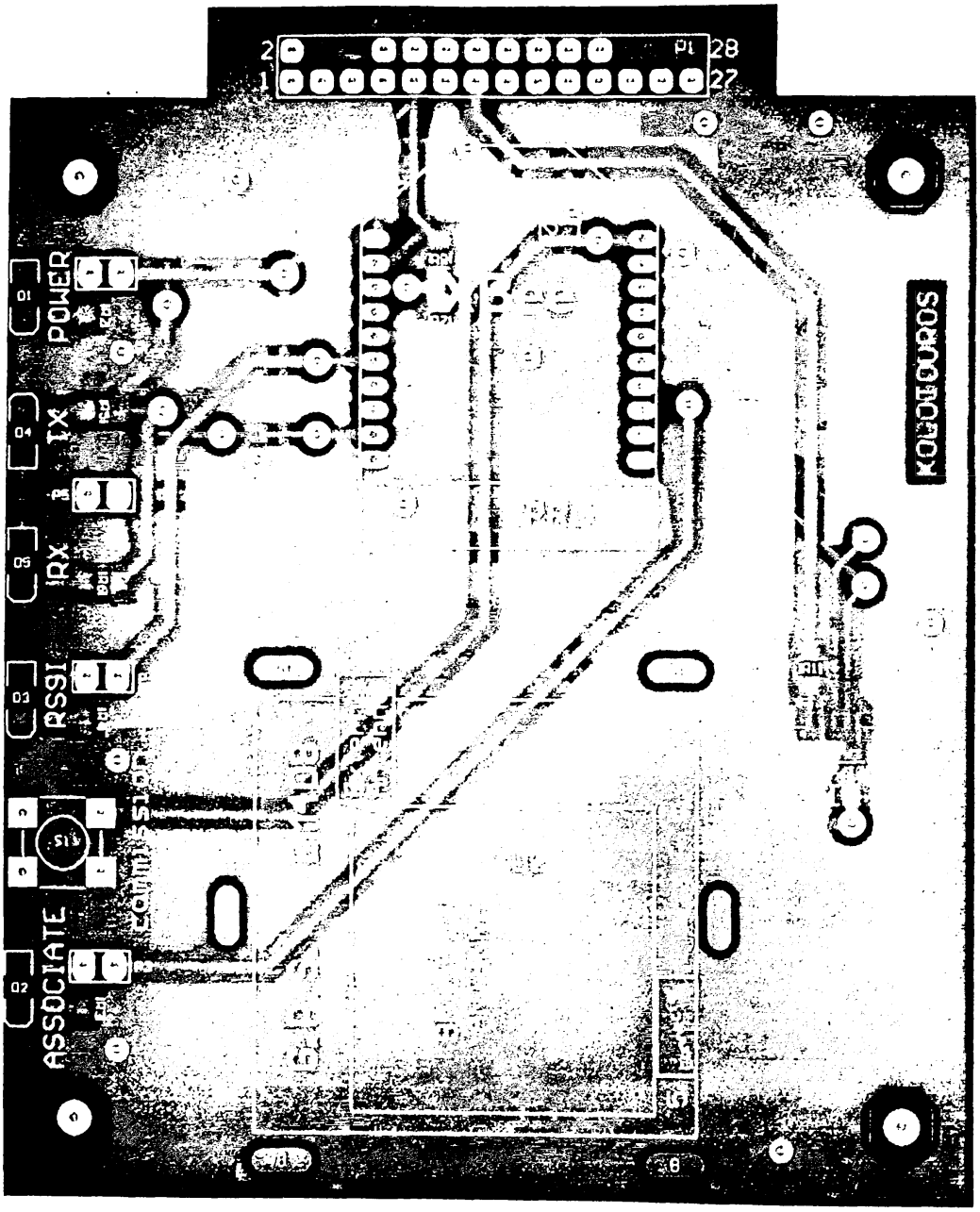
Σχηματικό Διάγραμμα, Κύκλωμα RF



Σχέδιο Gerber, Top Layer

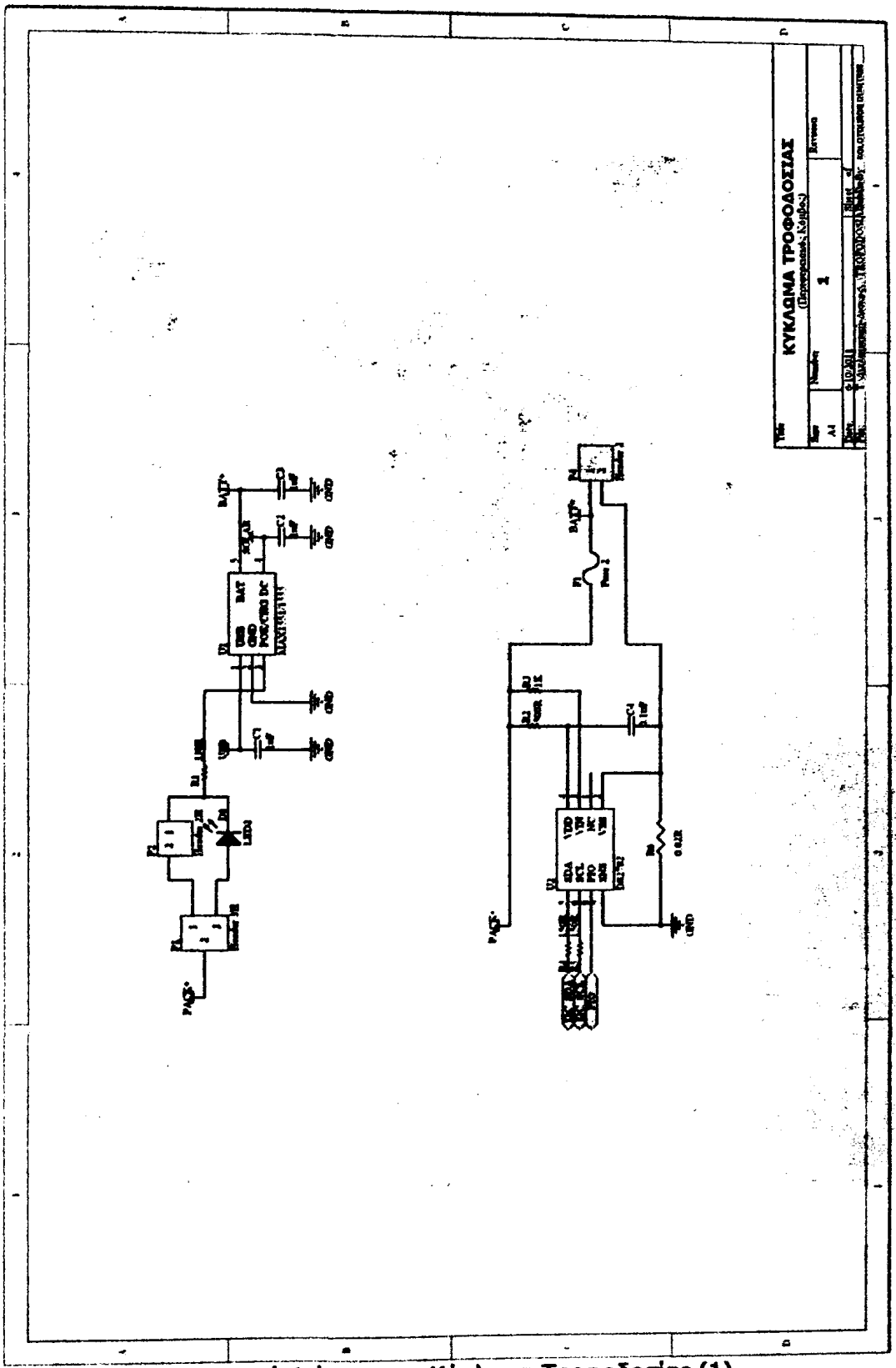


Σχέδιο Gerber, Bottom Layer

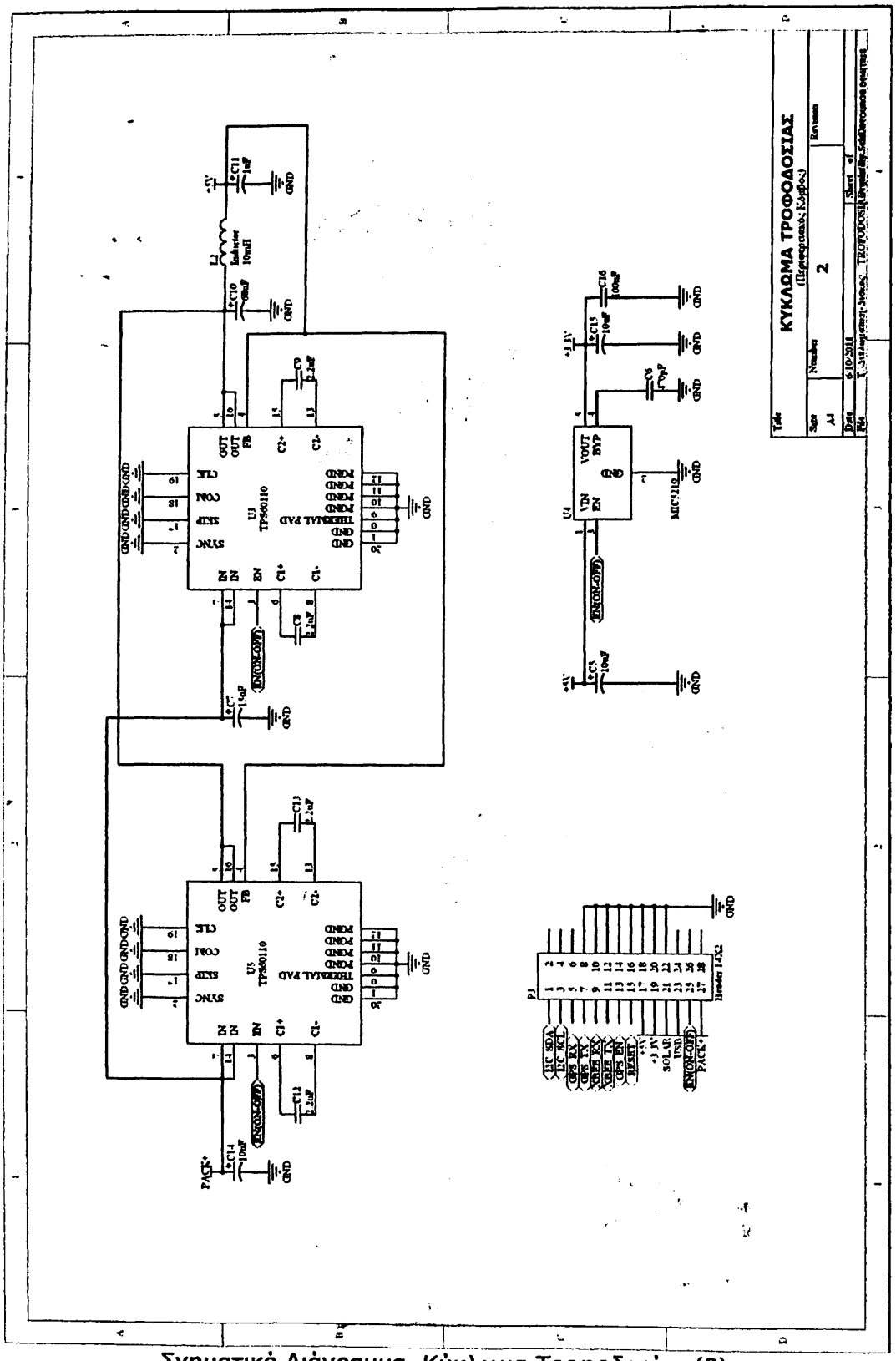


Layout Κυκλώματος με όλα τα layer ορατά
(Top, Top Overlay, Bottom, Bottom Overlay, Keep-Out, Multilayer)

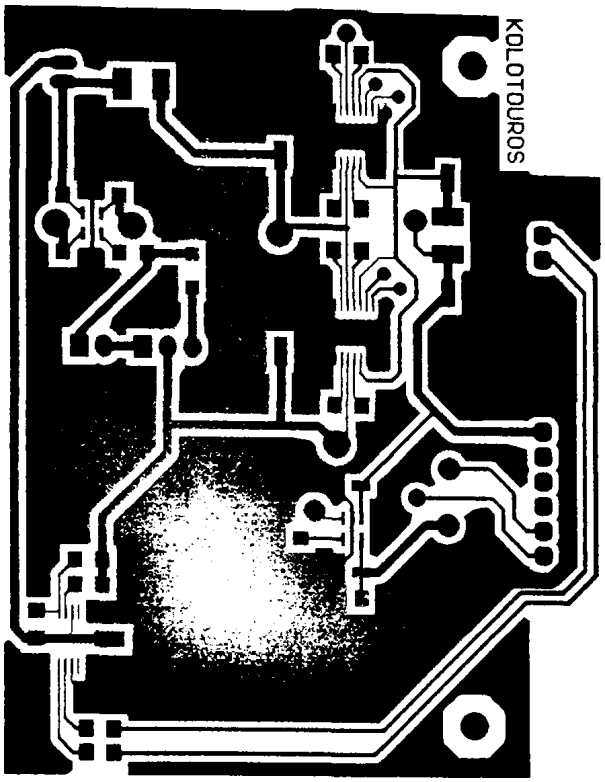
Π.Γ.2.3 Κύκλωμα Τροφοδοσίας



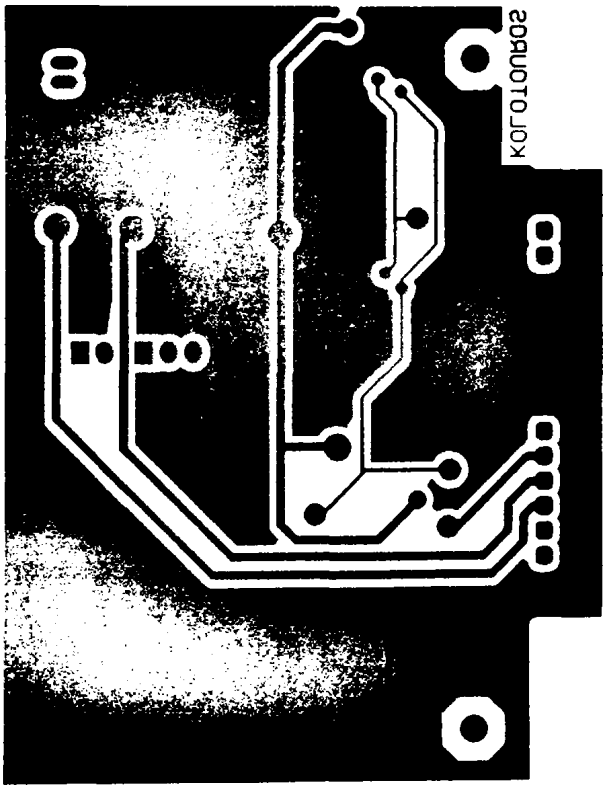
Σχηματικό Διάγραμμα, Κύκλωμα Τροφοδοσίας (1)



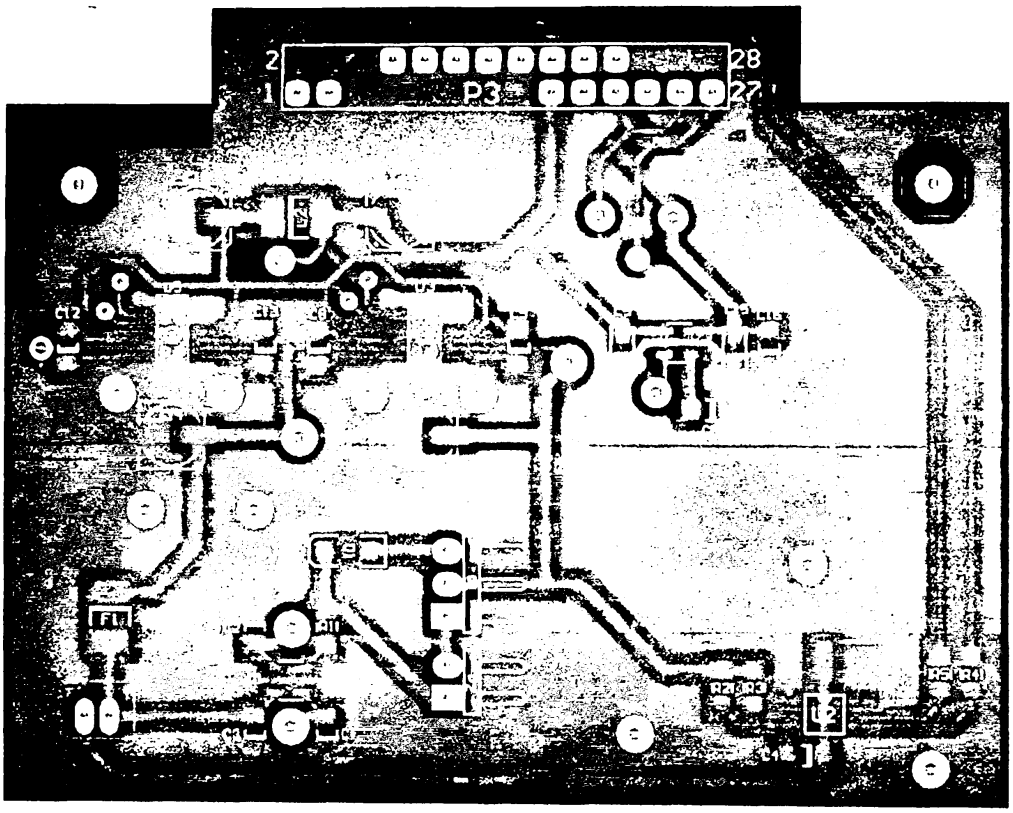
Σχηματικό Διάγραμμα, Κύκλωμα Τροφοδοσίας (2)



Σχέδιο Gerber, Top Layer



Σχέδιο Gerber, Bottom Layer



Layout Κυκλώματος με όλα τα layer ορατά
(Top, Top Overlay, Bottom, Bottom Overlay, Keep-Out, Multilayer)