ΤΜΗΜΑ ΦΥΣΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

Ανάπτυξη ασύρματου ενεργειακά αυτόνομου συστήματος έγκαιρης προειδοποίησης με FPGA και IOT

Μεταπτυχιακή Διπλωματική Εργασία

Αθανάσιος Θ. Κολοβός
Φυσικός

Επίβλεψη: Αναπληρωτής Καθηγητής Νικόλαος Μάνθος
Εργαστήριο Φυσικής Υψηλών Ενεργειών

Ιωάννινα, Μάιος 2018

Στην οικογένεια μου
και στους φίλους μου

ΠΕΡΙΛΗΨΗ

Το αντικείμενο της παρούσας μεταπτυχιακής διπλωματικής εργασίας είναι η ανάπτυξη ενός ασύρματου, ενεργειακά αυτόνομου συστήματος έγκαιρης προειδοποίησης. Το σύστημα συλλέγει μετρήσεις φυσικοχημικών παραμέτρων του νερού ενός ποταμού, τις αποθηκεύει και εν' συνεχεία τις μεταδίδει ασύρματα μέσω ενός διακομιστή TFTP. Περιγράφεται η ανάπτυξη ενός συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης των τιμών της θερμοκρασίας και της αγωγιμότητας του νερού ενός ποταμού, η υλοποίηση ενός διακομιστή TFTP για την αποστολή των δεδομένων, το σύστημα συγχρονισμού των σταθμών ασύρματης αναμετάδοσης των παραπάνω μετρήσεων και το σύστημα τροφοδοσίας του συστήματος βασιζόμενο σε φωτοβολταϊκά στοιχεία και υπερπυκνωτές. Συγκεκριμένα, το νερό αντλείται περιοδικά σε ανοξείδωτο δοχείο με αισθητήρα θερμοκρασίας και αγωγιμότητας. Το σήμα του αισθητήρα μετατρέπεται σε ψηφιακό και οδηγείται για αποθήκευση σε μια κάρτα microSD και αποστέλλεται μέσω ενός διακομιστή TFTP. Η όλη διαδικασία αναπτύχθηκε σε ένα προγραμματιζόμενο ολοκληρωμένο FPGA. Το κύκλωμα στο FPGA, περιγράφεται σε VHDL και περιλαμβάνει επιπλέον έναν μικροεπεξεργαστή MicroBlaze. Τα δεδομένα παραλαμβάνονται από ένα σύστημα TFTP που λειτουργεί σε ένα προσωπικό υπολογιστή στο Τμήμα Φυσικής του Πανεπιστημίου Ιωαννίνων και παρουσιάζονται σε ιστοσελίδα.

ABSTRACT

The subject of the present MSc thesis is the development of a wireless, energy efficient early warning system. The system collects measurements of a couple of physicochemical parameters of river water, stores and transmits them wirelessly via a TFTP server to an internet terminal station. In particular, the energy autonomous system for sampling - measuring - storing the temperature and conductivity of the water, the TFTP server, the synchronization of the complete system, the power supply based on photovoltaic panels and superconductors is described. The sampling – measuring process control includes the water pumping, the measurement of the water parameters, the parameter digitization and storage in a microSD card and their transmission to the internet terminal station. The process control circuit is developed in a programmable FPGA. The FPGA circuit is described in VHDL and in addition a MicroBlaze microprocessor is also included. The data is received by a TFTP client on a computer at the Department of Physics of the University of Ioannina and they are presented on a web page.

ΠΡΟΛΟΓΟΣ

Φυσικά φαινόμενα όπως σεισμοί, ηφαιστειακές εκρήξεις, καταιγίδες απειλούσαν πάντα την ζωή στον πλανήτη Γή. Ένας από τους λόγους που έκανε αυτά τα φαινόμενα υπερβολικά επικίνδυνα ήταν η μη δυνατότητα έγκυρης προειδοποίησης της έλευσής τους. Με τη ραγδαία ανάπτυξη των επιστημών τον $20^{ο}$ αιώνα, είναι δυνατό τέτοια φαινόμενα να προβλεφτούν εγκαίρως και να περιοριστούν, τουλάχιστον σε μεγάλο βαθμό, οι επιπτώσεις τους. Το παραπάνω γίνεται δυνατό με την ανάπτυξη συστημάτων έγκαιρης προειδοποίησης. Στην παρούσα μεταπτυχιακή διπλωματική εργασία περιγράφεται και αναλύεται η ανάπτυξη ενός τέτοιου συστήματος.

Στο πρώτο κεφάλαιο γίνεται μια εισαγωγή στα συστήματα έγκαιρης προειδοποίησης καθώς και στα συστήματα συλλογής δεδομένων. Επίσης, γίνεται αναφορά στους αισθητήρες που χρησιμοποιούνται για τη συλλογή μετρήσεων φυσικοχημικών παραμέτρων και στα είδη επεξεργαστών που τους διαχειρίζονται. Τέλος, υπάρχει μία σύντομη περιγραφή του Διαδικτύου των Πραγμάτων (Internet of Things) και του συστήματος που αναπτύχθηκε.

Στο δεύτερο κεφάλαιο, περιγράφεται η πλακέτα Avnet LX9 Microboard, η οποία περιέχει το FPGA Spartan-6 της εταιρίας Xilinx και χρησιμοποιήθηκε ως κεντρική επεξεργαστική μονάδα του συστήματος που αναπτύχθηκε.

Στο τρίτο κεφάλαιο, περιγράφεται το τοπικό σύστημα και τα υδραυλικά μέρη του, ο αισθητήρας και τα ετοιμοπαράδοτα ηλεκτρονικά που χρησιμοποιήθηκαν.

Στο τέταρτο κεφάλαιο, παρουσιάζονται τα σχηματικά διαγράμματα των ηλεκτρονικών πλακετών που σχεδιάστηκαν και περιγράφονται τα υποκυκλώματά τους. Επίσης, γίνεται αναφορά στο σχεδιαστικό πακέτο KiCad που χρησιμοποιήθηκε για τον σχεδιασμό των πλακετών.

Στο πέμπτο κεφάλαιο, περιγράφεται η υλοποίηση του συστήματος και επιπροσθέτως, περιγράφεται η ανάπτυξη αλγορίθμου για την αποθήκευση και αποστολή δεδομένων. Επίσης, αναλύεται το λογισμικό που αναπτύχθηκε.

Στο έκτο κεφάλαιο, περιγράφεται η λειτουργία του συστήματος, η διαδικασία βαθμονόμησης των αισθητήρων, η υλοποίηση του διακομιστή TFTP και η ενεργειακή κατανάλωση του συστήματος.

Στο έβδομο κεφάλαιο, ανακεφαλαιώνεται η ανάπτυξη και αναφέρονται τα συμπεράσματα για το σύστημα έκτακτης προειδοποίησης που κατασκευάστηκε.

Στο Παράρτημα Α, παρατίθενται τα σχέδια των τυπωμένων κυκλωμάτων του συστήματος.

Στα Παραρτήματα Β, Γ, Δ παρουσιάζεται ο κώδικας που αναπτύχθηκε για τη λειτουργία του συστήματος και της διεπαφής χρήστη.

Στο Παράρτημα Ε, περιγράφεται η δημιουργία του μικροεπεξεργαστή MicroBlaze.

Στο Παράρτημα ΣΤ, αναλύεται η δημιουργία του λογισμικού περιφερειακού που δημιουργήθηκε για τον έλεγχο της δειγματοληψίας.

Στο Παράρτημα Ζ, περιγράφεται η διαδικασία προγραμματισμού της μνήμης flash της ηλεκτρονικής πλακέτας Avnet LX9 MicroBoard ως PROM.

Στο Παράρτημα Η, παρατίθεται η λίστα με τα υλικά του που χρησιμοποιήθηκαν για τη δημιουργία του συστήματος.

Στο Παράρτημα Θ, παρατίθεται μία σύντομη περιγραφή των πρωτοκόλλων επικοινωνίας I^2C, SPI και UART που χρησιμοποιήθηκαν.

Στο Παράρτημα Ι, παρατίθεται το εγχειρίδιο χρήστη του συστήματος.

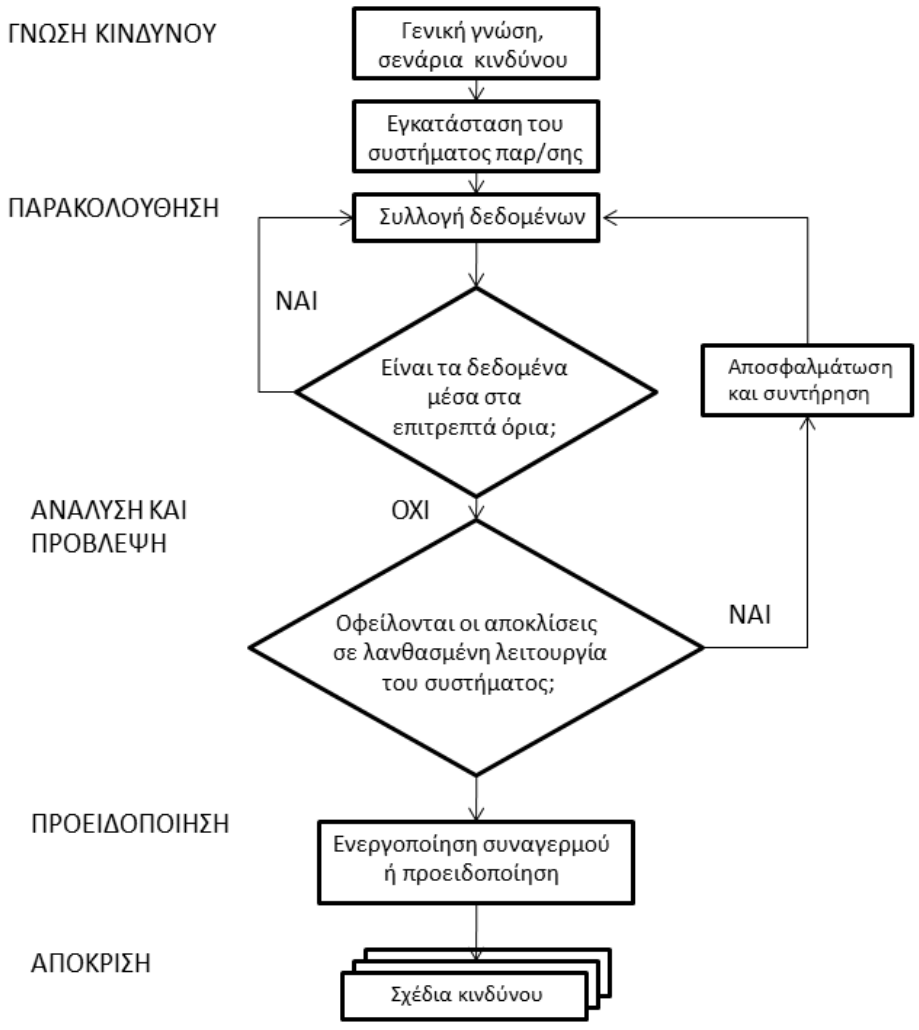
Τέλος, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Νικόλαο Μάνθο για την καθοδήγηση του σε όλη τη διάρκεια εκπόνησης της μεταπτυχιακής διπλωματικής εργασίας μου. Επίσης, θα ήθελα να ευχαριστήσω τον τεχνικό του Εργαστηρίου Φυσικής Υψηλών Ενεργειών του τμήματος Φυσικής, Ευστάθιο Μπλέτσα και όλα τα μέλη του Εργαστηρίου. Επιπροσθέτως, θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου και τους φίλους μου για την ψυχολογική υποστήριξη που μου παρείχαν.

ΠΕΡΙΕΧΟΜΕΝΑ

1.Εισαγωγή

1.1 Το σύστημα έγκαιρης προειδοποίησης

Σύστημα έγκαιρης προειδοποίησης (early warning system "EWS") είναι το σύστημα που ενημερώνει τον διαχειριστή του έγκαιρα για τυχόντα επικείμενο κίνδυνο, με αποτέλεσμα να ληφθούν τα απαραίτητα μέτρα για την πλήρη αποφυγή του ή τη μείωση των βλαβών και φθορών που προκαλεί [1]. Η υλοποίηση του, σε επίπεδο hardware-software περιλαμβάνει ένα συνδυασμό από αισθητήρες, συστήματα ελέγχου και αναμετάδοσης πληροφορίας τα οποία συνδυάζονται μεταξύ τους ώστε να υπάρξει το επιθυμητό αποτέλεσμα, δηλαδή η σωστή και έγκαιρη προειδοποίηση. Στο επίπεδο σχεδιασμού του συστήματος, απαιτείται η εκτενής γνώση του «καταστροφικού» φαινομένου που ενδέχεται να λάβει χώρα, ώστε να ελαχιστοποιηθεί το σφάλμα στη διάγνωση του φυσικού φαινομένου και να αντιμετωπιστεί αποτελεσματικά.

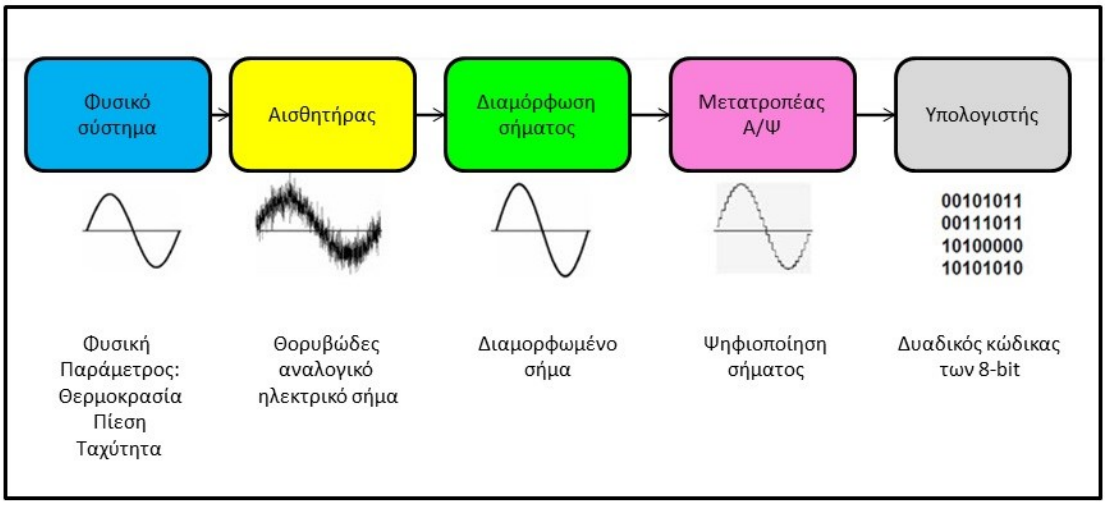


Εικόνα 1. Τυπικό διάγραμμα ροής για ένα σύστημα έγκαιρης προειδοποίησης

Στην εικόνα 1 απεικονίζεται ένα τυπικό διάγραμμα ροής για το σύστημα έγκαιρης προειδοποίησης. Μερικά φυσικά φαινόμενα για τα οποία μπορούν να εφαρμοστούν τέτοια συστήματα είναι οι σεισμοί, μία πυρκαγιά σε ένα δάσος, ένα τσουνάμι ή η εμφάνιση ενός τυφώνα. Επίσης, μπορούν να εφαρμοστούν και σε επικίνδυνα φαινόμενα που οφείλονται στον άνθρωπο, όπως για παράδειγμα η εμφάνιση ενός εμποδίου στην κατεύθυνση του αυτοκινήτου κατά τη διάρκεια της οδήγησης ή σε πολύ μεγαλύτερη κλίμακα επικινδυνότητας, π.χ η εκτόξευση ενός πυραύλου με προορισμό μία πόλη κατά τη διάρκεια εμπόλεμης σύρραξης. Στη συγκεκριμένη διπλωματική εργασία το σύστημα που αναπτύχθηκε μελετά την πιθανή ρύπανση των υδάτων του ποταμού Αράχθου.

Με τον όρο συλλογή δεδομένων ορίζεται η διαδικασία μέτρησης φυσικών παραμέτρων και η μετατροπή των τιμών τους σε αριθμούς που μπορούν να επεξεργαστούν από έναν ηλεκτρονικό υπολογιστή. Τα συστήματα συλλογής – επεξεργασίας δεδομένων (Data Acquisition Systems, DAQ) ως συνέχεια των αισθητήρων, οι οποίοι μετατρέπουν φυσικές παραμέτρους σε ηλεκτρικά σήματα αποτελούνται από:

- Κυκλώματα για τη μετατροπή των ηλεκτρικών σημάτων σε μία μορφή που μπορεί να μετατραπεί σε ψηφιακή τιμή,
- Μετατροπείς αναλογικού-σε-ψηφιακό σήμα (Analog-to-digital converters ή ADCs), που μετατρέπουν τα αναλογικά ηλεκτρικά σήματα των αισθητήρων σε ψηφιακές τιμές [2].



Εικόνα 2: Διάγραμμα δομής (block diagram) συστήματος συλλογής δεδομένων

Στην εικόνα 2 φαίνεται το διάγραμμα ροής για ένα σύστημα συλλογής δεδομένων.

Αισθητήρας, στον πιο γενικό ορισμό του, είναι μία συσκευή, ηλεκτρονικό στοιχείο ή υποσύστημα που ανιχνεύει γεγονότα ή αλλαγές στο φυσικό περιβάλλον και στέλνει τις πληροφορίες που συλλέγει σε άλλα ηλεκτρονικά κυκλώματα [2]. Η χρήση τους στην καθημερινή μας ζωή είναι ευρεία, καθώς χρησιμοποιούνται σε κινητά τηλέφωνα, οικιακές συσκευές, μέσα μεταφοράς όπως επίσης και σε πιο

«προηγμένες» εφαρμογές, για παράδειγμα δορυφόρους, ιατρικά ρομπότ, επιταχυντές σωματιδίων κτλ.

Ανάλογα με το φυσικό φαινόμενο που παρατηρούν και τον τρόπο λειτουργίας τους, οι αισθητήρες μπορούν να ταξινομηθούν στις παρακάτω κατηγορίες:

1. Θερμικοί, πχ. τα θερμόμετρα,
2. Ακτινοβολίας, πχ. αισθητήρες υπέρυθρου φωτός, ακτινών χ, σωματιδίων,
3. Μηχανικοί, πχ. τα γυροσκόπια,
4. Μαγνητικοί, πχ. οι πυξίδες,
5. Ηλεκτρικοί, πχ. τα πολύμετρα,
6. Χημικοί, πχ. αισθητήρες pH και
7. Βιολογικοί.

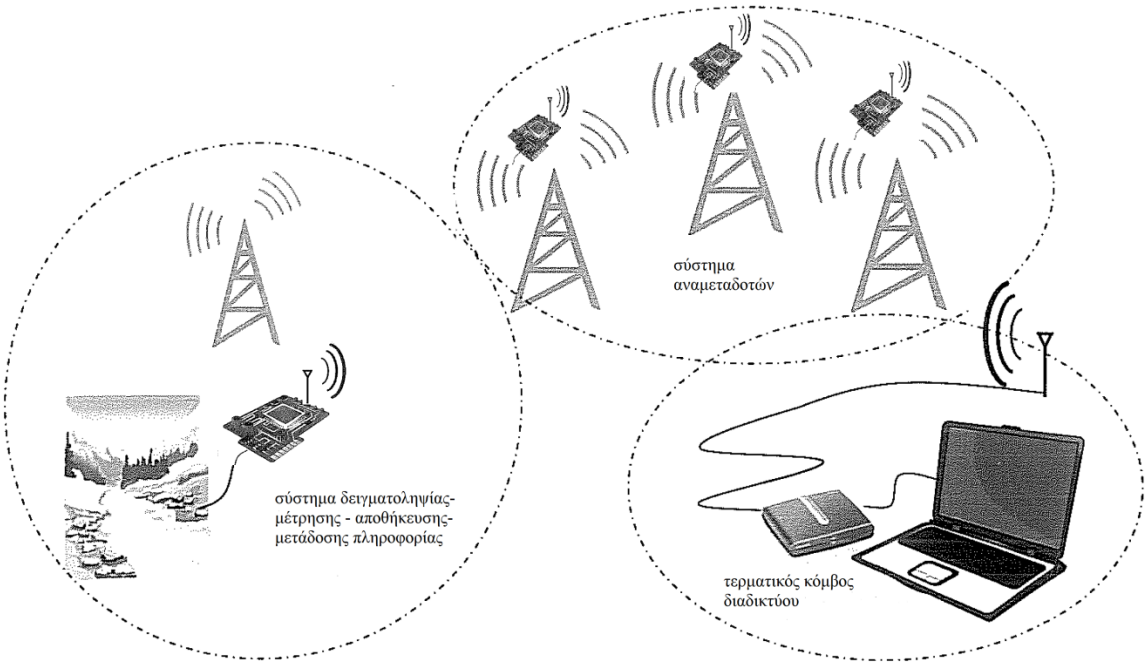
Με την ραγδαία ανάπτυξη της τεχνολογίας τον 20^{ο} αιώνα δημιουργήθηκε η ανάγκη για την κατασκευή συστημάτων τα οποία διευκολύνουν την καθημερινή μας ζωή και συνδυάζουν ένα μεγάλο εύρος από δυνατότητες σε μικρό μέγεθος. Στα πλαίσια αυτής της ανάγκης αναπτύχθηκαν τα πρώτα ενσωματωμένα συστήματα (embedded systems) τη δεκαετία του 60. Ενσωματωμένο σύστημα θεωρείται μία οποιαδήποτε συσκευή η οποία περιλαμβάνει ένα προγραμματιζόμενο σύστημα υπολογισμού, το οποίο δεν είναι υπολογιστής γενικού σκοπού. Προγραμματιζόμενο σύστημα υπολογισμού μπορεί να είναι ένας μικροελεγκτής που είναι ένθετος σε ένα ευρύτερο σύστημα που περιέχει μηχανικά ή ηλεκτρονικά μέρη και έχει μία συγκεκριμένη λειτουργία. Τα πλεονεκτήματα τους σε σχέση με τους υπολογιστές γενικής χρήσης είναι η χαμηλή τους ενεργειακή κατανάλωση, το μικρό μέγεθος και το μικρό κόστος τους. Οι εφαρμογές τους ποικίλουν από απλές συσκευές που χρησιμοποιούμε στην καθημερινή μας ζωή, όπως κινητά τηλέφωνα, ψηφιακά ρολόγια και συσκευές αναπαραγωγής πολυμέσων, έως πολύπλοκα συστήματα, για παράδειγμα μαγνητικούς τομογράφους, ραντάρ και συστήματα καθοδήγησης πυραύλων. Τέλος, εκτός από μικροελεγκτές μπορούν να χρησιμοποιηθούν μικροεπεξεργαστές, FPGAs (Field Programmable Gate Array) ή ολοκληρωμένα κυκλώματα συγκεκριμένης εφαρμογής ASIC.

Πιο πρόσφατα αναπτύχθηκε η ιδέα της δημιουργίας ενός «διαδικτύου των αντικειμένων» (Internet of Things, IoT). Εμφανίστηκε πρώτη φορά το 1982 στο πανεπιστήμιο Carnegie Mellon, όπου κατασκευάστηκε μία μηχανή αυτόματης πώλησης αναψυκτικών που είχε τη δυνατότητα να παρέχει πληροφορίες για τον αριθμό των προϊόντων που περιέχει και για τη θερμοκρασία τους μέσω του διαδικτύου [3]. Ως «διαδίκτυο των αντικειμένων» ορίζεται ένα δίκτυο από ενσωματωμένα συστήματα, μηχανικές ή ηλεκτρονικές συσκευές, μέσα μεταφοράς, αισθητήρες κτλ. που είναι συνδεδεμένα μεταξύ τους, έχοντας μοναδικές «ταυτότητες» και τη δυνατότητα να μεταφέρουν δεδομένα μέσω ενός διαδικτύου χωρίς την ανάγκη αλληλεπίδρασης ανθρώπου με άνθρωπο ή ανθρώπου με υπολογιστή. Ένα αντικείμενο, στο συγκεκριμένο διαδίκτυο, μπορεί να είναι μία συσκευή ενός ατόμου η οποία μετρά τους χτύπους της καρδιάς του, οι αισθητήρες μίας λεωφόρου που μετρούν την κατάσταση της κυκλοφορίας σε αυτή ή οποιαδήποτε

άλλη συσκευή η οποία έχει διεύθυνση IP και τη δυνατότητα να μεταφέρει πληροφορίες μέσω του διαδικτύου. Οι εφαρμογές που έχει η παραπάνω τεχνολογία είναι πάρα πολλές ξεκινώντας από μικρή κλίμακα, για παράδειγμα ένα «έξυπνο» σπίτι στο οποίο οι οικιακές συσκευές μπορούν να επικοινωνήσουν μεταξύ τους για τη δημιουργία του μεσημεριανού γεύματος: ο φούρνος που έχει μία βάση δεδομένων με συνταγές μπορεί να «πει» στο ψυγείο τι υλικά χρειάζονται και αυτό με τη σειρά του να παραγγείλει ότι δεν υπάρχει, έως κλίμακα μίας ολόκληρης περιοχής της οποίας το σύστημα παροχής ενέργειας μπορεί να αυξομειώνει την παραγωγή ανάλογα με τις ανάγκες σε πραγματικό χρόνο.

1.2 Γενική περιγραφή του συστήματος

Το σύστημα που αναπτύχθηκε, είναι ένα σύστημα έγκαιρης προειδοποίησης το οποία μελετά την κατάσταση των φυσικοχημικών παραμέτρων των νερών ενός ποταμού και δίνει τη δυνατότητα αναγνώρισης εάν υπάρχει ρύπανση. Αποτελείται από σταθμούς συλλογής δεδομένων και μετάδοσης πληροφορίας και σταθμούς αναμετάδοσης πληροφορίας. Η τοποθέτηση τους θεωρείται ότι έγινε σε διάφορα σημεία κατά μήκος του ποταμού.



Εικόνα 3: Συνδεσμολογία συστήματος [4]

Στην εικόνα 3 φαίνεται η συνδεσμολογία του συστήματος. Στο σύστημα συλλογής δεδομένων και μετάδοσης της πληροφορίας, υπάρχει ένα τοπικό σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης της πληροφορίας ενώ στο σύστημα αναμεταδοτών τα δεδομένα αναμεταδίδονται μέχρι να φτάσουν στον τελικό τους προορισμό που είναι ένας τερματικός κόμβος διαδικτύου. Όλα τα συστήματα συγχρονίζονται ώστε τα δεδομένα να φτάσουν απρόσκοπτα στον τελικό προορισμό.

Το τοπικό σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης περιλαμβάνει:

- Σύστημα τροφοδοσίας που χρησιμοποιεί φωτοβολταϊκά πάνελ για τη συλλογή ηλεκτρικής ενέργειας και υπερπυκνωτές για την αποθήκευση της [5],
- Σύστημα επιτήρησης – ρύθμισης τάσης προς τους υπερπυκνωτές και από τους υπερπυκνωτές στο σύστημα,
- Αισθητήρα για την μέτρηση της ποιότητας του νερού,
- Σύστημα αυτοματισμού δειγματοληψίας που αποτελείται από δοχείο, αντλία που λειτουργεί με την υδραυλική ενέργεια, ηλεκτρικές βαλβίδες και αισθητήρες στάθμης,
- Σύστημα συγχρονισμού που αποτελείται από ρολόι πραγματικού χρόνου και GPS, για την ακριβή ρύθμιση του ρολογιού.
- Σύστημα καταγραφής και αποθήκευσης της πληροφορίας,
- Ένα διακομιστή TFTP (trivial file transfer protocol) για την αναμετάδοση της πληροφορίας που υλοποιείται από τον επεξεργαστή του συστήματος,
- Ένα μέσου ύψους (4m και 2m ιστός) πύργο κεραίας με κατευθυντική κεραία Wi-Fi [5].

Το σύστημα αναμεταδοτών αποτελείται από:

- Σύστημα τροφοδοσίας που χρησιμοποιεί φωτοβολταϊκά πάνελ για τη συλλογή ηλεκτρικής ενέργειας και υπερπυκνωτές για την αποθήκευση της [5],
- Σύστημα επιτήρησης – ρύθμισης τάσης προς τους υπερπυκνωτές και από τους υπερπυκνωτές στο σύστημα,
- Σύστημα συγχρονισμού που αποτελείται από ρολόι πραγματικού χρόνου και GPS, για την ακριβή ρύθμιση του ρολογιού,
- Ένα μέσου ύψους (4m και 2m ιστός) πύργο κεραίας με κατευθυντική κεραία Wi-Fi [5].

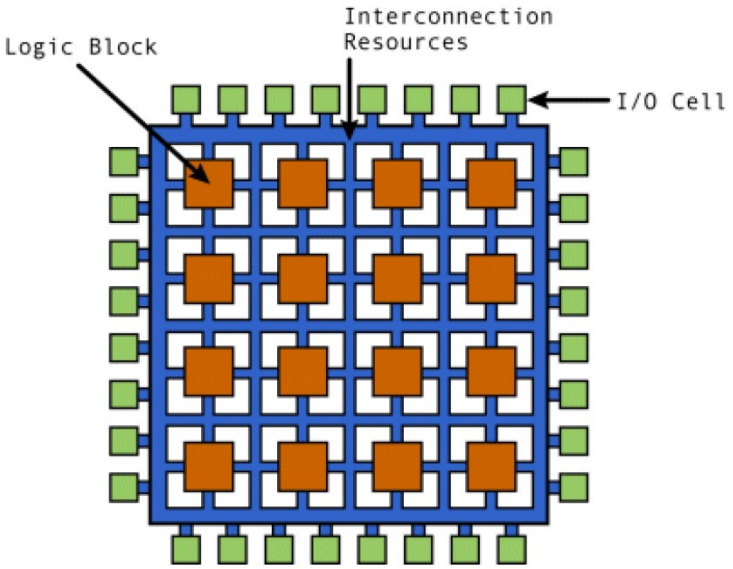
Τέλος, ο τερματικός κόμβος διαδικτύου περιέχει τα παρακάτω στοιχεία:

- Ένα πύργο με κατευθυντική κεραία Wi-Fi [5],
- Ηλεκτρονικό υπολογιστή για τη διαχείριση του δικτύου, την αποθήκευση και ανάλυση των δεδομένων.

2. FPGA

2.1 Γενικές πληροφορίες

Η κεντρική μονάδα επεξεργασίας και ελέγχου των συστημάτων δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης της πληροφορίας και του αναμεταδότη είναι η ηλεκτρονική πλακέτα Spartan 6 LX9 Microboard της εταιρίας Avnet που χρησιμοποιεί ένα Spartan-6 FPGA της εταιρίας Xilinx. Τα FPGA (Field Programmable Gate Array) είναι ολοκληρωμένα κυκλώματα που ο χρήστης τους μπορεί να τα προγραμματίσει χρησιμοποιώντας την γλώσσα προγραμματισμού περιγραφής hardware (Hardware Description Language ή HDL) VHDL ή Verilog. Η πιο κοινή αρχιτεκτονική ενός FPGA αποτελείται από μία διάταξη προγραμματιζόμενων λογικών τμημάτων (configurable logic block ή CLB), I/O pads και κανάλια δρομολόγησης των σημάτων στα λογικά τμήματα.



Εικόνα 4: Διάγραμμα αρχιτεκτονικής ενός FPGA

Στην εικόνα 4 φαίνεται ένα διάγραμμα το οποίο απεικονίζει τις διασυνδέσεις μεταξύ των κύριων τμημάτων ενός FPGA [6]. Ένα από τα πλεονεκτήματα των FPGA σε σχέση με άλλα ολοκληρωμένα κυκλώματα επεξεργασίας, όπως ένας μικροελεγκτής, είναι ότι η διαδικασία, ανάλογα με το ηλεκτρονικό σχέδιο το οποίο εγκαθίσταται στο FPGA, μπορεί να εκτελείται παράλληλα και όχι σειριακά αυξάνοντας έτσι δραματικά την ταχύτητα εκτέλεσης των εφαρμογών για τις οποίες χρησιμοποιούνται. Τέλος, ο χρήστης έχει τη δυνατότητα να υλοποιήσει ένα μικροεπεξεργαστή λογισμικού στο FPGA, σε συνδυασμό με άλλα κυκλώματα. Στη συγκεκριμένη εργασία υλοποιήθηκε ο μικροεπεξεργαστής MicroBlaze της εταιρίας Xilinx.

Στην εικόνα 5 φαίνεται η πλακέτα LX9 Microboard της εταιρίας Avnet που χρησιμοποιήθηκε:

Εικόνα 5: Avnet Spartan-6 LX9 Microboard

Τα κύρια χαρακτηριστικά της είναι τα εξής:

- 10/100 Ethernet PHY,
- μνήμη 128 Mb SPI Flash,
- διακόπτες DIP των 4-bit,
- Μνήμη 64MB LPDDR SDRAM,
- 4 LEDs,
- Ενσωματωμένο κύκλωμα USB JTAG,
- Προγραμματιζόμενο ρολόι χρονισμού,
- FPGA Spartan-6 XC6SLX9-2CSG324C,
- 2 συνδέτες I/O (2x6) συμβατές με DigilentPmod™ [23],
- Υποδοχή USB-to-UART [7].

Οι λόγοι που οδήγησαν στην επιλογή της συγκεκριμένης πλακέτας για την ανάπτυξη του συστήματος είναι:

1. Το Κόστος: Αρκετά πιο φθηνή σε σχέση με άλλες πλακέτες που χρησιμοποιούν το ίδιο FPGA,
2. Το Μέγεθος: Σημαντικό πλεονέκτημα, για συστήματα περιορισμένου χώρου,
3. Τα χαρακτηριστικά της: Έχει όλα τα απαραίτητα χαρακτηριστικά που χρειάζονται για την συγκεκριμένη εφαρμογή.

Η τροφοδοσία της πλακέτας γίνεται μέσω ενός συνδέτη USB micro-b από την υποδοχή USB-to-UART και είναι 5V. Επίσης, οι συνδέτες I/O έχουν την δυνατότητα να παρέχουν τάση 3.3V και γείωση εάν χρειαστεί σε εξωτερικά περιφερειακά. Ο προγραμματισμός του FPGA γίνεται μέσω του συνδέτη USB JTAG από ηλεκτρονικό υπολογιστή.

Όπως αναφέρθηκε παραπάνω στο FPGA υλοποιήθηκε ο μικροεπεξεργαστής MicroBlaze της εταιρίας Xilinx. Ο Microblaze είναι ένας μικροεπεξεργαστής τύπου Reduced Instruction Set Computer (RISC) των 32-bit, προγραμματιζόμενος σε γλώσσα C ή C++, συμβατός με όλες τις νέες οικογένειες FPGA της Xilinx και με αρκετές από τις παλαιότερες. Το πλεονέκτημά του είναι ο μεγάλος βαθμός παραμετροποίησης του, ανάλογα με το που θα χρησιμοποιηθεί. Η παραμετροποίησή του επιτυγχάνεται με το λογισμικό Embedded Development Kit (EDK). Το EDK δίνει τη δυνατότητα, αφαίρεσης ή πρόσθεσης δυνατοτήτων ανάλογα με τις ανάγκες

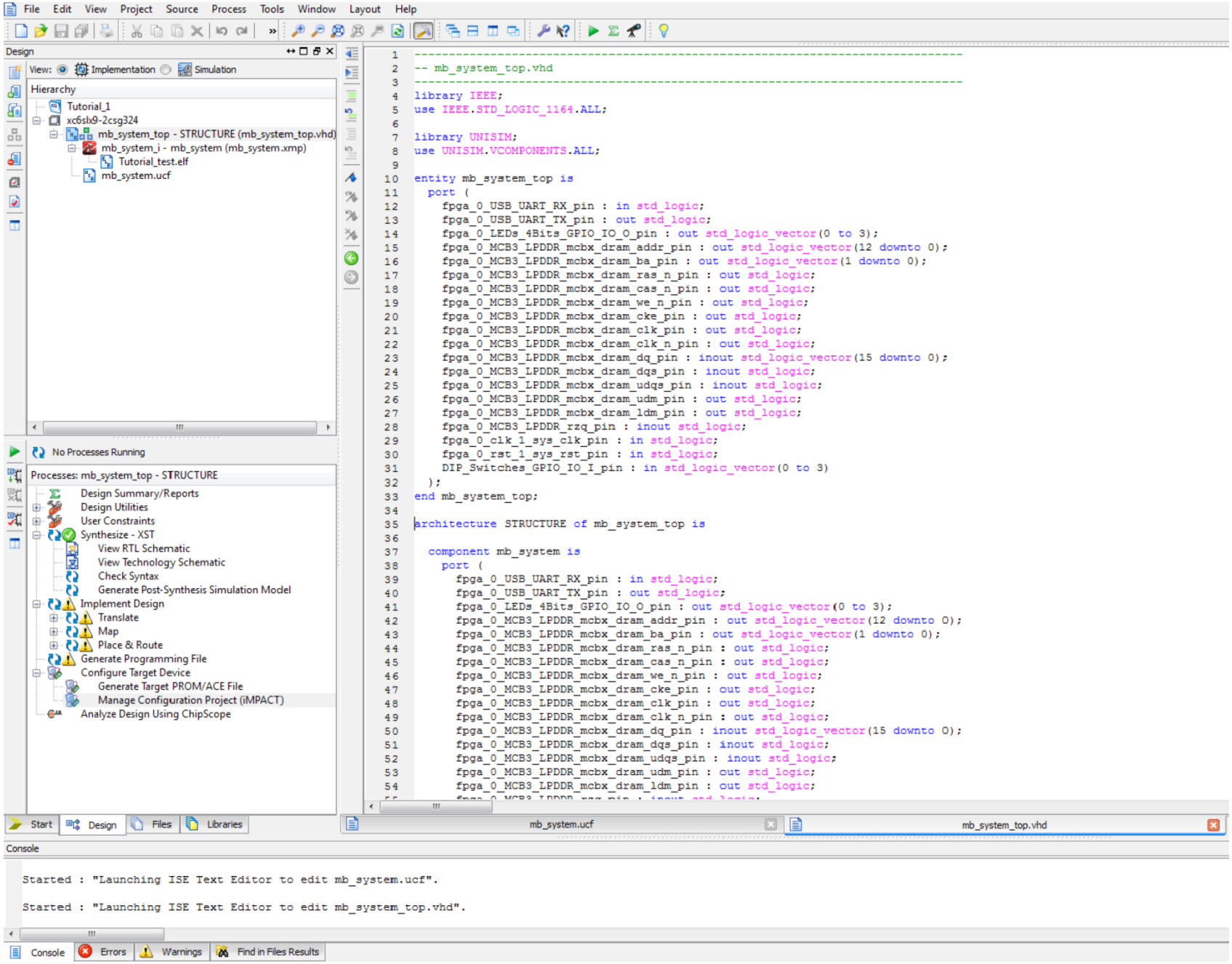
τις εκάστοτε εφαρμογής. Τέλος, ο MicroBlaze μπορεί να χρησιμοποιηθεί χωρίς λειτουργικό σύστημα αλλά είναι συμβατός με το λειτουργικό σύστημα Linux [8].

2.2 Λογισμικό ISE Design Suite

Το λογισμικό πακέτο στο οποίο αναπτύχθηκε ο κώδικας του συστήματος είναι η πλατφόρμα ISE Design Suite 14.7 της εταιρίας Xilinx. Περιέχει το πακέτο λογισμικού ISE για τον προγραμματισμό σε γλώσσα VHDL ή Verilog και το Embedded Development Kit (EDK) για τον MicroBlaze. Το τελευταίο αποτελείται από δύο πακέτα: το Xilinx Platform Studio (XPS) και το Software Development Kit (SDK).

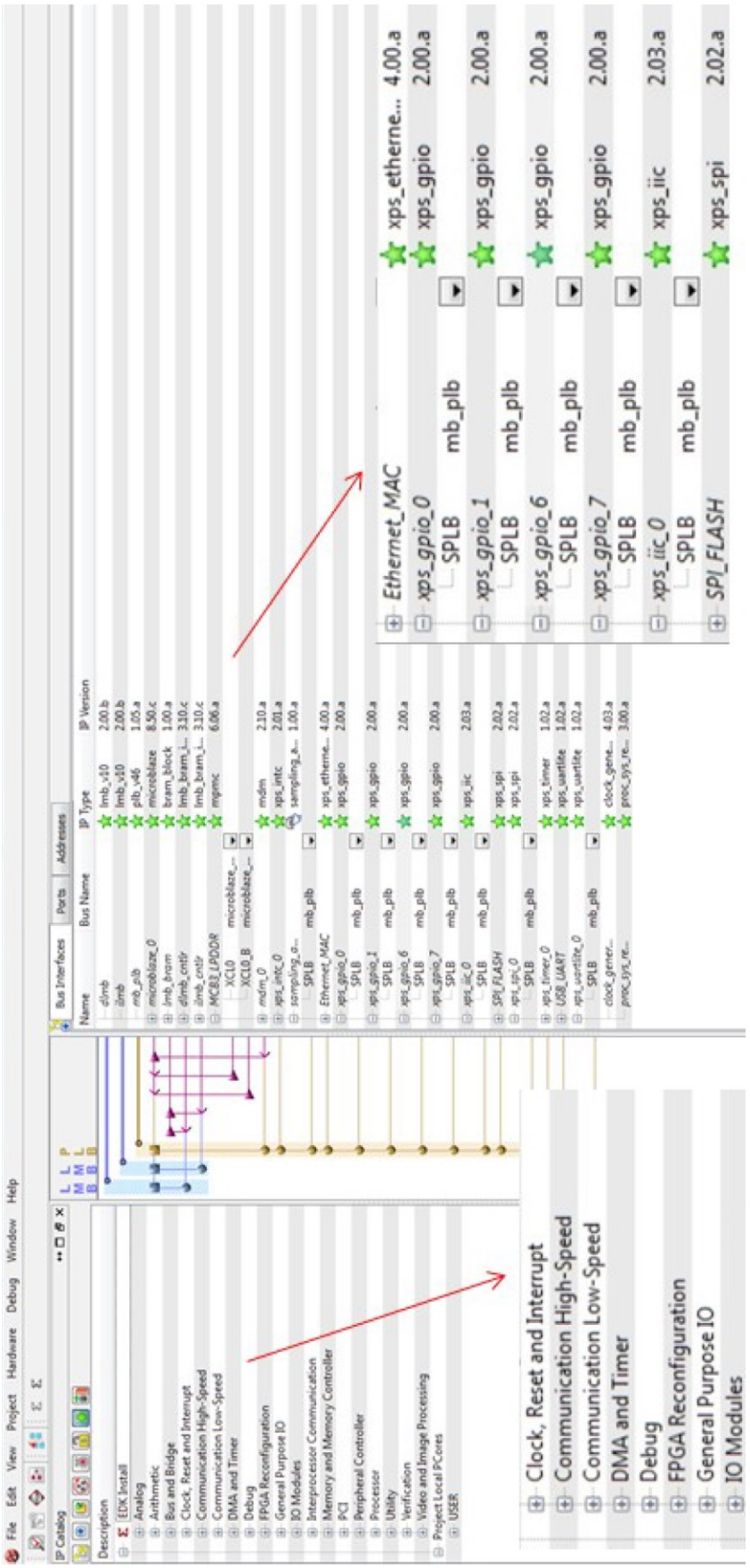
Στην εικόνα 6 παρουσιάζεται η διεπαφή χρήστη της πλατφόρμας ISE 14.7. Είναι ο συνδετικός κρίκος του κώδικα που αναπτύχθηκε για το σύστημα, καθώς ενώνει όλα τα μέρη του σε ένα αρχείο το οποίο φορτώνεται στη μνήμη Flash του FPGA. Στην κεντρική δεξιά καρτέλα φαίνεται ο κώδικας σε γλώσσα VHDL των σημάτων που συνδέονται σε ένα μικροεπεξεργαστή Microblaze. Στην πάνω αριστερή καρτέλα είναι ταξινομημένα, ιεραρχικά, τα αρχεία που απαρτίζουν την κάθε εφαρμογή που δημιουργείται, ενώ στην κάτω αριστερή καρτέλα είναι τα εργαλεία που μεταφράζουν τον κώδικα και προγραμματίζουν τα λογικά block του FGPA δημιουργώντας έτσι το περιγραφόμενο κύκλωμα. Στην εικόνα 7 φαίνεται η διεπαφή χρήστη της πλατφόρμας Xilinx Platform Studio (XPS). Χρησιμοποιείται για την ανάπτυξη - διαμόρφωση του μικροεπεξεργαστή MicroBlaze, ανάλογα με τις ανάγκες του συστήματος που αναπτύσσει ο εκάστοτε χρήστης. Στην αριστερή καρτέλα είναι ο κατάλογος επιλογής των έτοιμων περιφερειακών που μπορούν να εισαχθούν στον μικροεπεξεργαστή. Οι επιλογές είναι αρκετές και περιέχουν, για παράδειγμα οδηγούς (drivers) για πρωτόκολλα επικοινωνίας μεταξύ ηλεκτρονικών (I^2C, SPI κτλ.), εργαλεία αποσφαλμάτωσης (debugging tools) και άλλα. Επίσης, δίνεται η δυνατότητα στο χρήστη να δημιουργήσει δικά του περιφερειακά, όπως έγινε για το σύστημα που πραγματεύεται η παρούσα διπλωματική εργασία. Συγκεκριμένα, δημιουργήθηκε περιφερειακό που ελέγχει τη διαδικασία αυτοματισμού δειγματοληψίας, προγραμματισμένο σε γλώσσα VHDL. Αναλυτική περιγραφή της δημιουργίας του υπάρχει στο Παράρτημα ΣΤ. Στην κεντρική καρτέλα γίνονται οι διασυνδέσεις των διάφορων περιφερειακών στον κεντρικό δίαυλο PLB (processor local bus) του μικροεπεξεργαστή, ώστε να ενωθούν στο γενικό σύστημα.

Τέλος, στην εικόνα 8 απεικονίζεται η διεπαφή χρήστη της εφαρμογής Software Development Kit (SDK), που είναι το κεντρικό περιβάλλον για τον προγραμματισμό του μικροεπεξεργαστή MicroBlaze. Ο κώδικας αναπτύσσεται σε γλώσσα προγραμματισμού C και οι βιβλιοθήκες των περιφερειακών που επιλέχθηκαν για το υπό ανάπτυξη σύστημα παράγονται αυτόματα. Επίσης, υπάρχει η δυνατότητα χρησιμοποίησης προανεπτυγμένου λογισμικού για εφαρμογές όπως υλοποίηση ενός διακομιστή διαδικτύου.




```
1  ----------------------------------------------------------------------
2  -- mb_system_top.vhd
3  ----------------------------------------------------------------------
4  library IEEE;
5  use IEEE.STD_LOGIC_1164.ALL;
6
7  library UNISIM;
8  use UNISIM.VCOMPONENTS.ALL;
9
10 entity mb_system_top is
11   port (
12     fpga_0_USB_UART_RX_pin : in std_logic;
13     fpga_0_USB_UART_TX_pin : out std_logic;
14     fpga_0_LEDs_4Bits_GPIO_IO_O_pin : out std_logic_vector(0 to 3);
15     fpga_0_MCB3_LPDDR_mcbx_dram_addr_pin : out std_logic_vector(12 downto 0);
16     fpga_0_MCB3_LPDDR_mcbx_dram_ba_pin : out std_logic_vector(1 downto 0);
17     fpga_0_MCB3_LPDDR_mcbx_dram_ras_n_pin : out std_logic;
18     fpga_0_MCB3_LPDDR_mcbx_dram_cas_n_pin : out std_logic;
19     fpga_0_MCB3_LPDDR_mcbx_dram_we_n_pin : out std_logic;
20     fpga_0_MCB3_LPDDR_mcbx_dram_cke_pin : out std_logic;
21     fpga_0_MCB3_LPDDR_mcbx_dram_clk_pin : out std_logic;
22     fpga_0_MCB3_LPDDR_mcbx_dram_clk_n_pin : out std_logic;
23     fpga_0_MCB3_LPDDR_mcbx_dram_dq_pin : inout std_logic_vector(15 downto 0);
24     fpga_0_MCB3_LPDDR_mcbx_dram_dqs_pin : inout std_logic;
25     fpga_0_MCB3_LPDDR_mcbx_dram_udqs_pin : inout std_logic;
26     fpga_0_MCB3_LPDDR_mcbx_dram_udm_pin : out std_logic;
27     fpga_0_MCB3_LPDDR_mcbx_dram_ldm_pin : out std_logic;
28     fpga_0_MCB3_LPDDR_rzq_pin : inout std_logic;
29     fpga_0_clk_1_sys_clk_pin : in std_logic;
30     fpga_0_rst_1_sys_rst_pin : in std_logic;
31     DIP_Switches_GPIO_IO_I_pin : in std_logic_vector(0 to 3)
32   );
33 end mb_system_top;
34
35 architecture STRUCTURE of mb_system_top is
36
37   component mb_system is
38     port (
39       fpga_0_USB_UART_RX_pin : in std_logic;
40       fpga_0_USB_UART_TX_pin : out std_logic;
41       fpga_0_LEDs_4Bits_GPIO_IO_O_pin : out std_logic_vector(0 to 3);
42       fpga_0_MCB3_LPDDR_mcbx_dram_addr_pin : out std_logic_vector(12 downto 0);
43       fpga_0_MCB3_LPDDR_mcbx_dram_ba_pin : out std_logic_vector(1 downto 0);
44       fpga_0_MCB3_LPDDR_mcbx_dram_ras_n_pin : out std_logic;
45       fpga_0_MCB3_LPDDR_mcbx_dram_cas_n_pin : out std_logic;
46       fpga_0_MCB3_LPDDR_mcbx_dram_we_n_pin : out std_logic;
47       fpga_0_MCB3_LPDDR_mcbx_dram_cke_pin : out std_logic;
48       fpga_0_MCB3_LPDDR_mcbx_dram_clk_pin : out std_logic;
49       fpga_0_MCB3_LPDDR_mcbx_dram_clk_n_pin : out std_logic;
50       fpga_0_MCB3_LPDDR_mcbx_dram_dq_pin : inout std_logic_vector(15 downto 0);
51       fpga_0_MCB3_LPDDR_mcbx_dram_dqs_pin : inout std_logic;
52       fpga_0_MCB3_LPDDR_mcbx_dram_udqs_pin : inout std_logic;
53       fpga_0_MCB3_LPDDR_mcbx_dram_udm_pin : out std_logic;
54       fpga_0_MCB3_LPDDR_mcbx_dram_ldm_pin : out std_logic;
```

Εικόνα 6: Διεπαφή χρήστη ISE 14.7



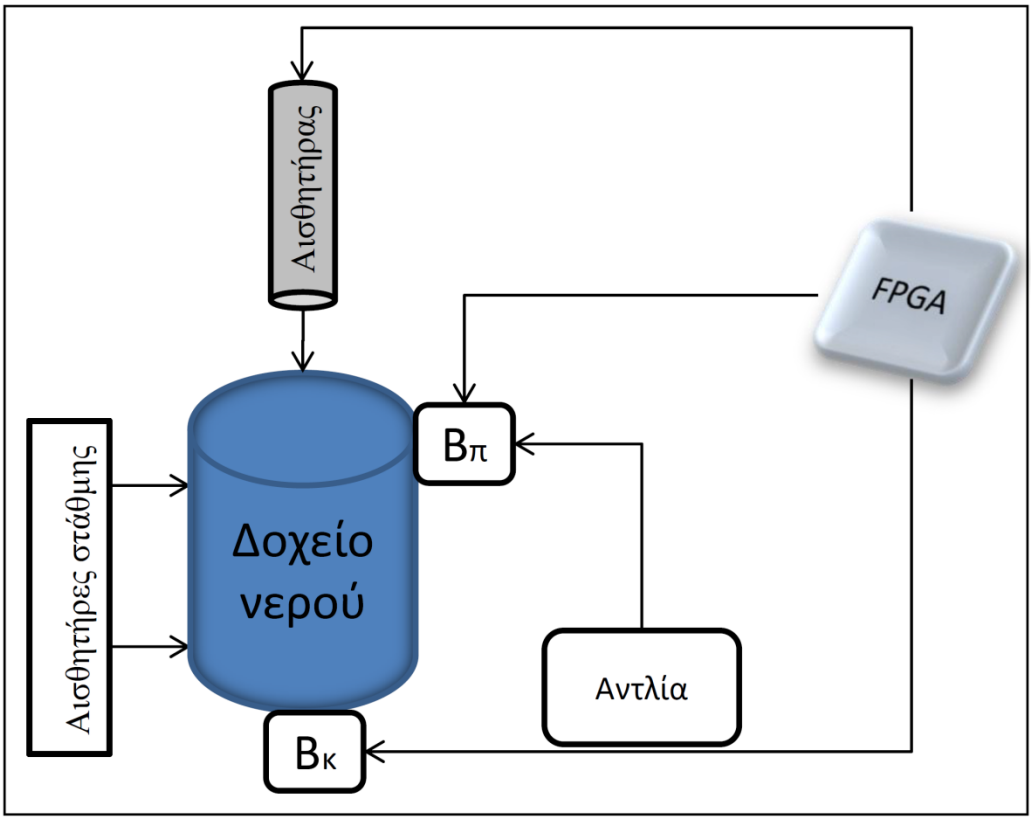
Εικόνα 7: Διεπαφή χρήστη Xilinx Platform Studio (XPS)

Εικόνα 8: Διεπαφή χρήστη Software Development Kit (SDK)

3. Τοπικό σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης και συγχρονισμού

3.1 Περιγραφή

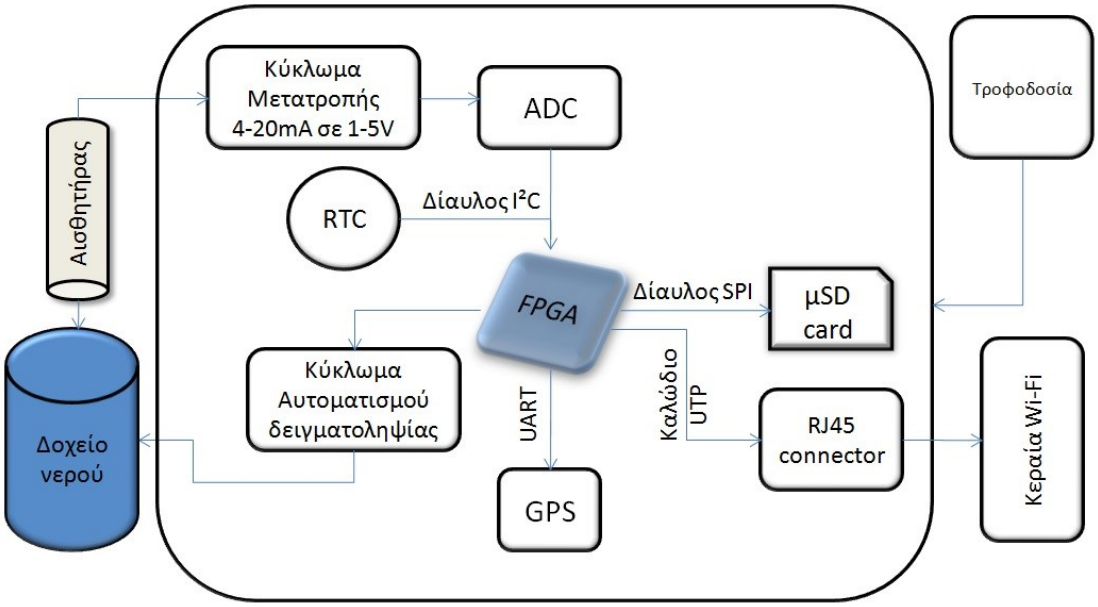
Το τοπικό σύστημα που αναπτύχθηκε για την παρούσα διπλωματική εργασία αποτελείται από το ηλεκτρονικό/ηλεκτρικό τμήμα και το τμήμα δειγματοληψίας. Το τμήμα δειγματοληψίας περιλαμβάνει ένα ανοξείδωτο δοχείο στο οποίο αντλείται το νερό και μία αντλία που τέσσερις φορές τη μέρα ανανεώνει το δείγμα του νερού. Η άντληση στο δοχείο γίνεται με εντολή του ηλεκτρονικού τμήματος του συστήματος.



Εικόνα 9: Block διάγραμμα συστήματος δειγματοληψίας

Στην εικόνα 9 απεικονίζεται το γενικό διάγραμμα του συστήματος δειγματοληψίας. Η διαδικασία που ακολουθείται είναι η εξής: Το δοχείο είναι πάντα γεμάτο με νερό ώστε να μην καταστρέφεται η μεμβράνη του αισθητήρα. Η διαδικασία δειγματοληψίας λαμβάνει χώρα τέσσερις φορές την ημέρα και συγκεκριμένα τις ώρες 00:00, 06:00, 12:00, 18:00 (+2 UTC) οι οποίες σηματοδοτούνται από το ρολόι πραγματικού χρόνου που βρίσκεται στο ηλεκτρονικό μέρος του συστήματος και συνδέεται στο FPGA. Σε κάθε μία από τις τέσσερις φορές, αρχικά ανοίγει η ηλεκτρική βάνα εκκένωσης Βκ ώστε να αδειάσει το νερό που υπήρχε. Όταν το επίπεδο του δείγματος πέσει κάτω από τον αισθητήρα στάθμης εκκένωσης ενεργοποιείται η ηλεκτρική βάνα Βπ πλήρωσης, με ανοιχτή την Βκ για ένα λεπτό, με

αποτέλεσμα να αντλείται φρέσκο νερό και να καθαρίζεται το δοχείο από τυχόν λάσπες ή χώμα που υπήρχαν σε αυτό. Η μείωση της παραμονής χώματος στο δοχείο είναι και η αιτία επιλογής κωνικού και όχι κυλινδρικού δοχείου. Στη συνέχεια, κλείνει η βάνα εκκένωσης και η αντλία γεμίζει το δοχείο με νερό, έως ότου το επίπεδο της επιφάνειας του νερού να περάσει το επίπεδο του αισθητήρα στάθμης πλήρωσης. Τότε η ηλεκτρική βάνα πλήρωσης κλείνει και το ηλεκτρονικό μέρος του συστήματος είναι έτοιμο να ξεκινήσει τη διαδικασία μέτρησης.



Εικόνα 10: Γενικό διάγραμμα του ηλεκτρονικού μέρους

Στην εικόνα 10 φαίνεται το γενικό διάγραμμα του ηλεκτρονικού μέρους του συστήματος. Όπως αναφέρθηκε παραπάνω το σύστημα ενεργοποιείται τέσσερις φορές την ημέρα, όμως ένα μέρος του ηλεκτρονικού κομματιού ενεργοποιείται μία ακόμα φορά, στις 14:00 ώστε να συγχρονίσει τους σταθμούς με τους αναμεταδότες. Η τελευταία διαδικασία περιλαμβάνει την ενεργοποίηση του GPS που είναι συνδεδεμένο με το FPGA. Το GPS δίνει ακριβείς τιμές ώρας και ημερομηνίας στο ρολόι πραγματικού χρόνου και έπειτα το σύστημα κλείνει. Τις άλλες τέσσερις φορές που το σύστημα περιοδικά τροφοδοτείται η διαδικασία περιλαμβάνει: Το μόνο μέρος του συστήματος που τροφοδοτείται είναι το RTC και είναι υπεύθυνο για την ενεργοποίηση του FPGA και την έναρξη της διαδικασίας δειγματοληψίας που περιγράφηκε παραπάνω. Μόλις τελειώσει η δειγματοληψία, το σήμα από τον αισθητήρα, αναλογικό 4-20mA, περνάει από το κύκλωμα μετατροπής 4-20mA σε 1-5V και εν συνεχεία ψηφιοποιείται στον ADC του συστήματος. Οι ψηφιακές τιμές λαμβάνονται από το FPGA, αποθηκεύονται στην κάρτα μνήμης uSD και ανεβαίνουν στον διακομιστή TFTP ώστε να μεταφερθούν στο τερματικό κόμβο διαδικτύου που βρίσκεται στο Πανεπιστήμιο Ιωαννίνων και το σύστημα κλείνει. Η τροφοδοσία του συστήματος γίνεται από φωτοβολταϊκά πάνελ που αποθηκεύουν την ενέργεια που

συλλέγουν σε υπερπυκνωτές [5]. Τέλος, το σύστημα συνδέεται σε ένα πομποδέκτη με κεραία Wi-Fi.

3.2 Υδραυλικό μέρος

Εικόνα 11: Μηχανική αντλία RP-100

Στην εικόνα 11 απεικονίζεται μία αντλία για την άντληση του νερού στο δοχείο. Είναι η αντλία ποταμού RIFE RP-100 της εταιρίας Rife Hydraulic Engine Mfg. Co. [9] και είναι ένα αυτόνομο σύστημα άντλησης, εξ' ολοκλήρου μηχανικό χωρίς να χρειάζεται ηλεκτρικό ρεύμα ή καύσιμο για τη λειτουργία της. Η δύναμη για την ώθηση παρέχεται από το τρεχούμενο νερό. Υπάρχει μόνο ένα κινούμενο μέρος, η περιστρεφόμενη σύζευξη, και είναι υδατολιπαντική [4]. Τα χαρακτηριστικά της παρουσιάζονται στον πίνακα 1:

Βάρος	15.88 kg
Διάμετρος σωλήνα εισόδου	12 mm
Διάμετρος σωλήνα εξόδου	16 mm
Μέγιστο ύψος άντλησης	8 m
Ελάχιστο απαιτούμενο βάθος	0.31 m
Διαστάσεις	38.1cm * 38.1cm* 76.2cm

Πίνακας 1: Χαρακτηριστικά ηλιακής αντλίας RP-100

3.3 Αισθητήρες

Για τον έλεγχο της ποιότητας του νερού χρησιμοποιήθηκε ο αισθητήρας αγωγιμότητας - θερμοκρασίας WQ-Cond της εταιρίας Global Water [10]. Οι αισθητήρες ποιότητας νερού της Global Water είναι πλήρως στεγανοί και συνοδεύονται από αδιάβροχα καλώδια. Σύμφωνα με τις υποδείξεις τις εταιρίας, δεν επιτρέπεται η τοποθέτηση των αισθητήρων σε εφαρμογές που περιέχουν διαλύτες, διότι με την πάροδο του χρόνου μπορεί να προκληθεί φθορά στο καλώδιο ή το αισθητήριο στοιχείο [4]. Η έξοδος τους είναι ένα αναλογικό σήμα 4 – 20 mA. Το 4 – 20 mA αποτελεί βιομηχανικό πρότυπο σήματος, επειδή έχει το πλεονέκτημα ότι το σήμα μπορεί να μεταφέρεται σε μεγάλη απόσταση χωρίς εξασθένηση λόγω της αντίστασης του καλωδίου.

Στην εικόνα 12 φαίνεται ο αισθητήρας αγωγιμότητας - θερμοκρασίας WQ-Cond. Για τη σύνδεσή του στο σύστημα χρησιμοποιούνται τέσσερα καλώδια. Το κόκκινο

καλώδιο συνδέεται στο κύκλωμα τροφοδοσίας του συστήματος (12V), το μαύρο στη γείωση, το λευκό είναι το σήμα εξόδου μέτρησης αγωγιμότητας και το πράσινο το σήμα εξόδου μέτρησης θερμοκρασίας, τα οποία συνδέονται στο κύκλωμα μετατροπής 4-20 mA σε 1-5V. Η σύνδεση γίνεται χωρίς την ύπαρξη τροφοδοσίας. Η μέτρηση της αγωγιμότητας γίνεται με τη χρήση τεσσάρων ηλεκτροδίων, αξιολογώντας την ικανότητα του νερού να οδηγήσει ηλεκτρικό ρεύμα μεταξύ αυτών.

Εικόνα 12: Αισθητήρας WQ-Cond

Η αγωγιμότητα του νερού εξαρτάται από τη συγκέντρωση ιόντων σε αυτό. Η αύξηση της συγκέντρωσής τους έχει αποτέλεσμα την αύξηση της αγωγιμότητας και είναι δείκτης ρύπανσης καθώς δείχνει την ύπαρξη διαλυμένων ανόργανων χημικών σε αυτό [11]. Κάθε υδρόβιο περιβάλλον έχει σταθερό εύρος τιμών αγωγιμότητας, το οποίο όταν οριστικοποιηθεί μπορεί να χρησιμοποιηθεί σαν σημείο αναφοράς. Αποκλίσεις των μετρήσεων αγωγιμότητας από το παραπάνω σημείο αναφοράς υποδεικνύουν ότι βιομηχανικά απόβλητα ή κάποιας άλλης μορφής ρύπανση έχει εισέλθει στο υδρόβιο περιβάλλον. Επίσης, η αγωγιμότητα του νερού εξαρτάται και από τη θερμοκρασία. Με την αύξηση της θερμοκρασίας του διαλύματος, μειώνεται το ιξώδες, αυξάνεται η κινητικότητα των ιόντων και δημιουργούνται νέα ιόντα, με αποτέλεσμα την αύξηση της αγωγιμότητας. Γι' αυτό το λόγο, ο αισθητήρας παρέχει αυτόματη αντιστάθμιση λόγω θερμοκρασίας, με αποτέλεσμα την παροχή ακριβούς μέτρησης αγωγιμότητας ανεξάρτητα από την θερμοκρασία που βρίσκεται το δείγμα. Στον πίνακα 2 φαίνονται τα χαρακτηριστικά του:

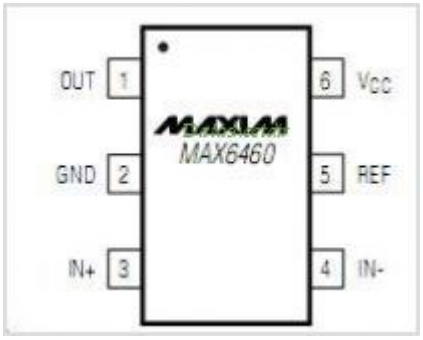
Σήμα εξόδου	4-20 mA
Εύρος αγωγιμότητας	200-2000μS/cm
Εύρος θερμοκρασίας	-5^{o}C έως +70^{o}C
Ακρίβεια μέτρησης αγωγιμότητας	0.5% της κλίμακας
Ακρίβεια μέτρησης θερμοκρασίας	±0.2 ^{o}C
Αντιστάθμιση θερμοκρασίας	2%/ ^{o}C (κανονικοποιημένη στους 25 ^{o}C)
Μέγιστη πίεση	35 psi
Τάση λειτουργίας	10-36V
Κατανάλωση ρεύματος	20mA + άθροισμα τιμών σημάτων εξόδου
Χρόνος προθέρμανσης	12secs minimum
Θερμοκρασία Λειτουργίας	-5^{o}C έως +70^{o}C

Μέγεθος αισθητήρα	Διάμετρος 22 mm Μήκος 202 mm
Αντιστάθμιση θερμοκρασίας	2% ανά ^{o}C
Ηλεκτρόδια	Ανοξείδωτο ατσάλι

Πίνακας 2: Χαρακτηριστικά αισθητήρα αγωγιμότητας WQ-Cond

3.4 **Χρήση ετοιμοπαράδοτων ηλεκτρονικών στοιχείων**

Στις επόμενες παραγράφους παρουσιάζονται τα κύρια ηλεκτρονικά στοιχεία που χρησιμοποιήθηκαν στο ηλεκτρονικό μέρος του συστήματος που αναπτύχθηκε.

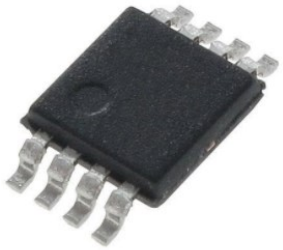


Εικόνα 13: Max 6460

Α. Το σύστημα που αναπτύχθηκε απαιτεί τάση λειτουργίας ίση ή μεγαλύτερη από 12V. Για την επιτήρηση της τάσης τροφοδοσίας, που παρέχεται από τους υπερπυκνωτές, επιλέχθηκε το ολοκληρωμένο MAX 6460 [12] (εικόνα 13) της εταιρίας MAXIM Integrated. Το MAX 6460 είναι ένας υψηλής τάσης εισόδου, μικρής κατανάλωσης ρεύματος επιτηρητής τάσης. Το MAX 6460 μπορεί να χρησιμοποιηθεί για την διακοπή παροχής σε περίπτωση χαμηλής τάσης ή προστασία ενός κυκλώματος από υψηλή τάση. Στο συγκεκριμένο σύστημα χρησιμοποιείται για να ελέγχεται εάν η τάση που παρέχουν οι υπερπυκνωτές είναι μεγαλύτερη από 14.5V, διότι ο ρυθμιστής τάσης (voltage regulator) των 12V χρειάζεται τάση εισόδου Vin=(Vo +2.5V). Εάν είναι μεγαλύτερη, παρέχεται ηλεκτρική ενέργεια στους ρυθμιστές τάσης του κυκλώματος τροφοδοσίας και το σύστημα λειτουργεί.

Τα βασικά χαρακτηριστικά του MAX 6460 είναι:

- Τάση τροφοδοσίας: 4V έως 28V,
- Εσωτερική τάση αναφοράς: 2.25V,
- Κατανάλωση ρεύματος: 3.5μA στα 12V,
- Περίβλημα SOT32 των 6 pin,
- Εύρος θερμοκρασίας περιβάλλοντος: -40^{o}C έως +125^{o}C.



Εικόνα 14: DS1337

Β. Στην εικόνα 14 φαίνεται το σειριακό ρολόι πραγματικού χρόνου (RTC) DS1337 της εταιρίας Maxim Integrated. Διαθέτει καταχωρητές ώρας/ημερομηνίας και δύο προγραμματιζόμενους συναγερμούς. Η μετάδοση δεδομένων και ο προγραμματισμός του γίνονται μέσω ενός διαύλου I^2C (Inter-integrated Circuit). Οι καταχωρητές ώρας και ημερομηνίας παρέχουν δεδομένα για δευτερόλεπτα, λεπτά, ώρες, ημέρα, μήνα και έτος.

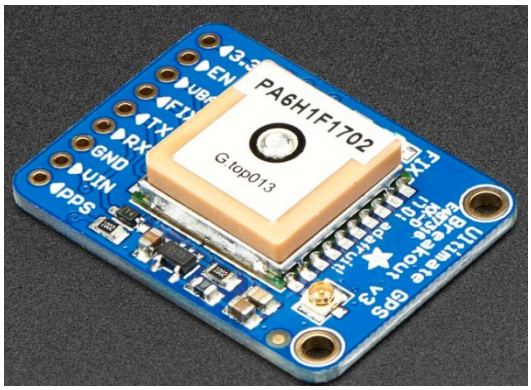
Οι καταχωρητές ημερομηνίας ρυθμίζονται αυτόματα με το πέρας των μηνών με λιγότερες από 31 ημέρες καθώς και τα δίσεκτα έτη. Το ρολόι λειτουργεί σε 24ώρη ή σε 12ώρη μορφή και έχει δείκτη AM/PM [13]. Η τροφοδοσία του γίνεται από μπαταρία και είναι πλήρως λειτουργικό για τάσεις από 1.8V έως 5.5V. Η εφαρμογή του στο σύστημα που αναπτύχθηκε, είναι να ενεργοποιεί το σύστημα πέντε φορές την ημέρα και να παρέχει ημερολογιακές πληροφορίες. Για την περιοδική ενεργοποίηση του συστήματος που αναπτύχθηκε επιλέχθηκε η ενεργοποίηση του συναγερμού 2 (alarm2) όταν ταυτίζονται οι καταχωρητές ώρας και λεπτών του RTC με τους καταχωρητές ώρας και λεπτών του alarm2. Ο προγραμματισμός των καταχωρητών του alarm 2 έγινε σύμφωνα με τον πίνακα 3.

DY/(/DT)	Bit 7 των καταχωρητών του alarm 2			Ενεργοποίηση alarm 2
	Καταχ. Ημέρας	Καταχ. Ώρας	Καταχ. Λεπτών	
X	1	1	1	Μία φορά ανά λεπτό
X	1	1	0	Όταν ταυτίζονται οι καταχ. λεπτών
X	1	0	0	Όταν ταυτίζονται οι καταχ. ώρας, λεπτών

Πίνακας 3: Προγραμματισμός alarm 2

Τα χαρακτηριστικά του είναι τα παρακάτω:

- Τάση τροφοδοσίας: 1.8V-5.5V (πλήρης λειτουργία),
- Κατανάλωση ρεύματος: 150μA (πλήρης λειτουργία), 1.5μA(αναμονή),
- Πακέτο μSOP των 8 pin,
- Εύρος θερμοκρασίας περιβάλλοντος: -40^{o}C έως +85^{o}C.



Εικόνα 15: Adafruit Ultimate GPS – v3

Γ. Για τον ακριβή συγχρονισμό του RTC χρησιμοποιήθηκε η ηλεκτρονική πλακέτα με GPS της εταιρίας Adafruit [14], που φαίνεται στην εικόνα 15. Βασίζεται στο ολοκληρωμένο GPS MTK3339 το οποίο μπορεί να επικοινωνεί με έως και 22 δορυφόρους. Η ηλεκτρονική πλακέτα μπορεί να τροφοδοτηθεί με τάση 3.3V ή 5V και παρέχει τη δυνατότητα χρήσης μπαταρίας ώστε να ελαχιστοποιείται ο χρόνος λήψης έγκυρης πληροφορίας. Έχει εσωτερική κεραία και υπάρχει υποδοχή για εξωτερική. Η συμβολοσειρά εξόδου NMEA που επιλέχθηκε είναι η RMC καθώς είναι η μόνη συμβολοσειρά εξόδου που παρέχει χρονικές και ημερολογιακές πληροφορίες. Οι πληροφορίες που περιέχει παρουσιάζονται στον πίνακα 4:

Παράδειγμα συμβολοσειράς εξόδου:
$GPRMC,064951.000,A,2307.1256,N,12016.4438,E,0.03,165.48,260406,3.05,W,A*2C

Όνομα	Παράδειγμα	Μονάδες	Περιγραφή
Ταυτότητα	$GPRMC		Επικεφαλίδα πρωτοκόλλου RMC
Χρόνος UTC	064951.000		ΩΩΛΛΔΔ.ΔΔΔ
Κατάσταση δεδομένων	A		A= Έγκυρα δεδομένα ή V= μη έγκυρα δεδομένα
Γεωγραφικό πλάτος	2307.1256		ΜΜΠΠ.ΠΠΠΠ
Δείκτης Β/Ν	N		N= Βορράς ή S= Νότος
Γεωγραφικό μήκος	12016.4438		ΜΜΜΠΠ.ΠΠΠΠ
Δείκτης Α/Δ	E		E= Ανατολή ή W=Δύση
Ταχύτητα	0.03	Κόμβοι	
Κατεύθυνση πορείας (course over ground)	165.48	Μοίρες	Πραγματική
Ημερομηνία	260406		ΗΗΜΜΕΕ
Μαγνητική απόκλιση	3.05, W	Μοίρες	E= Ανατολή ή W= Δύση
Λειτουργία	A		A= Αυτόνομη D= Διαφορική E= Εκτιμώμενη
Checksum	*2C		
<CR><LF>			Τέλος συμβολοσειράς

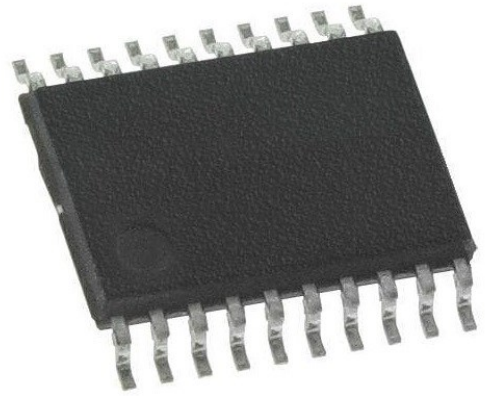
Πίνακας 4: Υπόμνημα συμβολοσειράς εξόδου

Τα βασικά του χαρακτηριστικά είναι τα εξής:

- Τάση τροφοδοσίας: 3.3V ή 5V,
- Κατανάλωση ρεύματος: έως 25mA,
- Δορυφόροι: 22,
- Ρυθμός ανανέωσης: 1 έως 10 Hz,
- Ακρίβεια θέσης: <3 μέτρα,
- Έξοδος: Πρότυπο NMEA 0183
- Εύρος θερμοκρασίας περιβάλλοντος: -40^{o}C έως +85^{o}C.

Εικόνα 16: Κάρτα μνήμης μSD

Δ. Στο κύκλωμα αποθήκευσης πληροφορίας του συστήματος, χρησιμοποιήθηκε κάρτα μνήμης microSD τύπου SDHC και χωρητικότητας 16GB της εταιρίας SanDisk [15] (εικόνα 16). Οι κάρτες μνήμης μSD είναι ευρέως διαδεδομένες ως μέσο αποθήκευσης, σε συσκευές όπως κινητά τηλέφωνα, ψηφιακές φωτογραφικές μηχανές κτλ. Πλεονεκτήματά τους είναι το μικρό μέγεθος, το χαμηλό κόστος αγοράς, η μόνιμη αποθήκευση δεδομένων (δεν σβήνονται με την διακοπή τροφοδοσίας), το μεγάλο εύρος διαθέσιμων χωρητικοτήτων που παρέχουν οι κατασκευάστριες εταιρίες και η συμβατότητα τους με όλα τα λειτουργικά συστήματα ηλεκτρονικών υπολογιστών και έξυπνων συσκευών. Τέλος, υποστηρίζουν πληθώρα πρωτοκόλλων επικοινωνίας, με το πιο διαδεδομένο να είναι το πρωτόκολλο SPI (Serial Peripheral Interface), το οποίο χρησιμοποιήθηκε για την επικοινωνία της κάρτας με το FPGA στο σύστημα που αναπτύχθηκε. Περιγραφή της λειτουργίας του διαύλου SPI υπάρχει στο παράρτημα Θ .



Εικόνα 17: AD7998 12-bit ADC

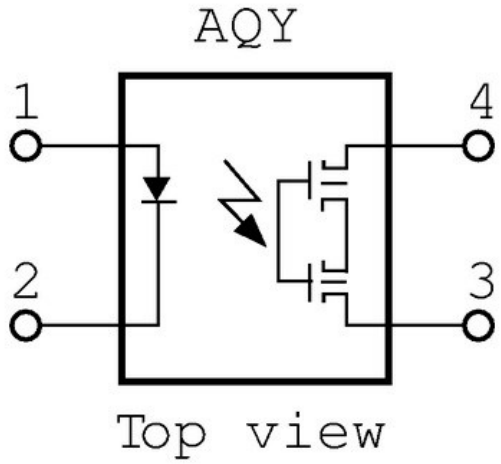
Ε. Για την μετατροπή των μετρήσεων που λαμβάνονται από τους αισθητήρες του συστήματος από αναλογικό σε ψηφιακό σήμα χρησιμοποιήθηκε ο μετατροπέας αναλογικού σήματος σε ψηφιακό (analog-to-digital converter) AD7998 της εταιρίας Analog Devices [16] (εικόνα 17). Η μετατροπή είναι απαραίτητη για την επεξεργασία των μετρήσεων από το FPGA. Η επικοινωνία του ADC με το FPGA γίνεται μέσω του πρωτοκόλλου επικοινωνίας I^2C το οποίο περιγράφεται στο παράρτημα Θ. Τα βασικά του χαρακτηριστικά παρουσιάζονται παρακάτω:

- Ανάλυση: 12-bit,
- 8 αναλογικές είσοδοι,
- Τάση τροφοδοσίας: 2.7 έως 5.5V,
- Τάση αναφοράς: 1.2V έως την τάση τροφοδοσίας,
- Απόλυτη ακρίβεια: ± 1 LSB,
- Χρόνος μετατροπής: 2μS,
- Συχνότητα σήματος εισόδου: έως 11MHz,
- Κατανάλωση ρεύματος: 0.5μA σε λειτουργία αναμονής, 0.4mA σε πλήρη λειτουργία,
- Περίβλημα 20-pin TSSOP,
- Εύρος θερμοκρασίας περιβάλλοντος: -40^{o}C έως +85^{o}C.

Ζ. Για τον έλεγχο της τροφοδοσίας των περιφερειακών του συστήματος που δεν χρειάζεται να είναι πάντα σε κατάσταση λειτουργίας χρησιμοποιήθηκαν οι ηλεκτρονόμοι στερεάς κατάστασης AQY282S της εταιρίας Panasonic [17] (εικόνα 18). Τα πλεονεκτήματα τους σε σχέση με τους μηχανικούς ηλεκτρονόμους είναι τα εξής:

Εικόνα 18: Ηλεκτρονόμος στερεάς κατάστασης

Μικρότερο μέγεθος, απολύτως αθόρυβη λειτουργία και αρκετά μεγαλύτερος χρόνος ζωής, καθώς δεν υπάρχουν κινητά μέρη. Η λειτουργία τους βασίζεται στην αρχιτεκτονική PhotoMos και φαίνεται στην εικόνα 19.



Εικόνα 19: Κύκλωμα PhotoMos solid state relay

Στους ακροδέκτες 1 και 2 είναι συνδεδεμένη μία δίοδος εκπομπής φωτός LED (light emitting diode) η οποία οδηγείται από την τάση ελέγχου που συνδέεται στους ακροδέκτες 1 και 2 (Vcc και γείωση αντίστοιχα). Στους ακροδέκτες 3 και 4 συνδέονται δύο MosFET με την συνδεσμολογία που φαίνεται στο σχήμα 19. Στις πύλες τους υπάρχουν ανιχνευτές φωτός. Όταν η τάση ελέγχου πάρει τιμή που θέτει το LED σε λειτουργία, εκπέμπεται φώς το οποίο όταν ανιχνευτεί ενεργοποιεί τα MosFET με αποτέλεσμα η τάση στον ακροδέκτη 4 να περάσει μέσα από αυτά και να βγει από τον ακροδέκτη 3 [18]. Τα χαρακτηριστικά του AQY282S είναι τα παρακάτω:

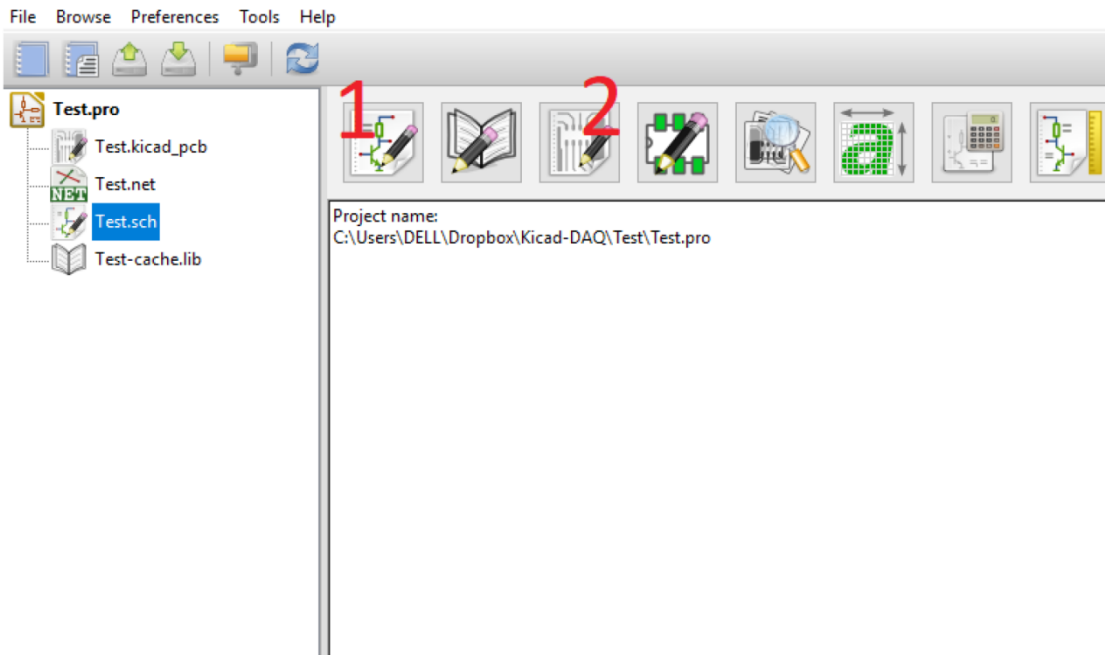
- Τάση ελέγχου: 1.5 έως 5V,
- Μέγιστη τάση εισόδου: 60V,
- Μέγιστο ρεύμα φορτίου: 500mA,
- Μέγιστος χρόνος αλλαγής κατάστασης: 3ms,
- Περίβλημα: SOP,
- Εύρος θερμοκρασίας περιβάλλοντος: -40^{o}C έως +85^{o}C.

4. Σχεδίαση συστήματος

4.1 Σχεδιαστικό πακέτο KiCad

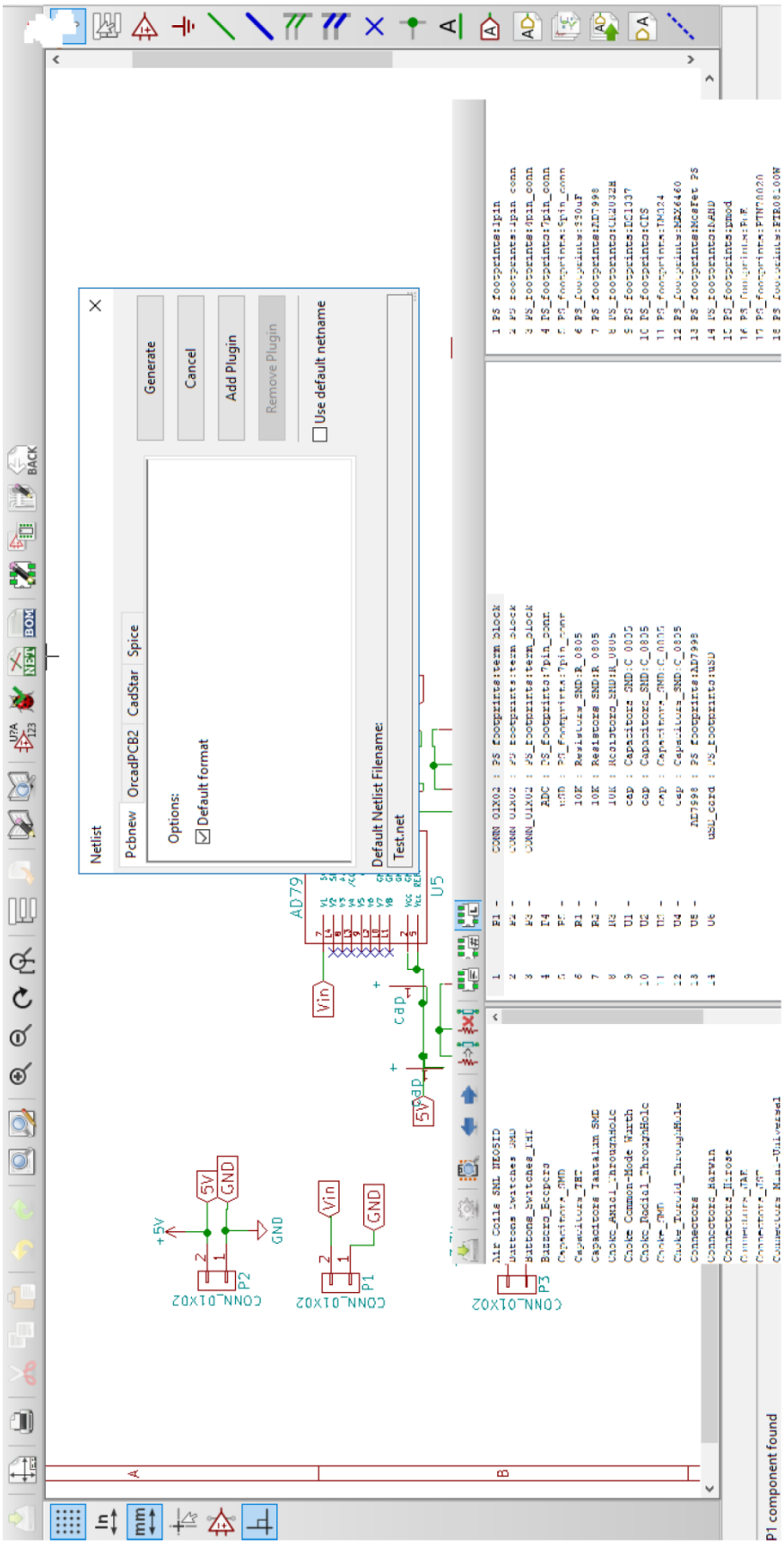
Για τη σχεδίαση του σχηματικού διαγράμματος του συστήματος καθώς και για την σχεδίαση του τυπωμένου κυκλώματος χρησιμοποιήθηκε το σχεδιαστικό πακέτο KiCad, έκδοση 4.0.5 [25]. Το KiCad είναι ένα πακέτο λογισμικού για σχεδίαση ηλεκτρονικών αυτοματισμών, ανοιχτού κώδικα και συμβατό με όλα τα ευρέως χρησιμοποιούμενα λειτουργικά συστήματα (Windows, macOS, Linux). Η σχεδίαση των τυπωμένων κυκλωμάτων βασίζεται σε δύο εφαρμογές του KiCad: το Eeschema (Electronic schematic editor) και το Pcbnew (Printed circuit board editor) ακολουθώντας την παρακάτω διαδικασία:

1. Σχεδιασμός του σχηματικού διαγράμματος με το Eeschema,
2. Αντιστοίχηση των στοιχείων του σχηματικού με τα footprints των τυποποιημένων ηλεκτρονικών που θα χρησιμοποιηθούν στο pcb, μέσω του εργαλείου CvPcb του Eeschema,
3. Δημιουργία λίστας διασυνδέσεων του κυκλώματος (netlist), από το Eeschema,
4. Σχεδίαση του τυπωμένου κυκλώματος, μέσω του Pcbnew και τέλος,
5. Εκτύπωση του τυπωμένου κυκλώματος.



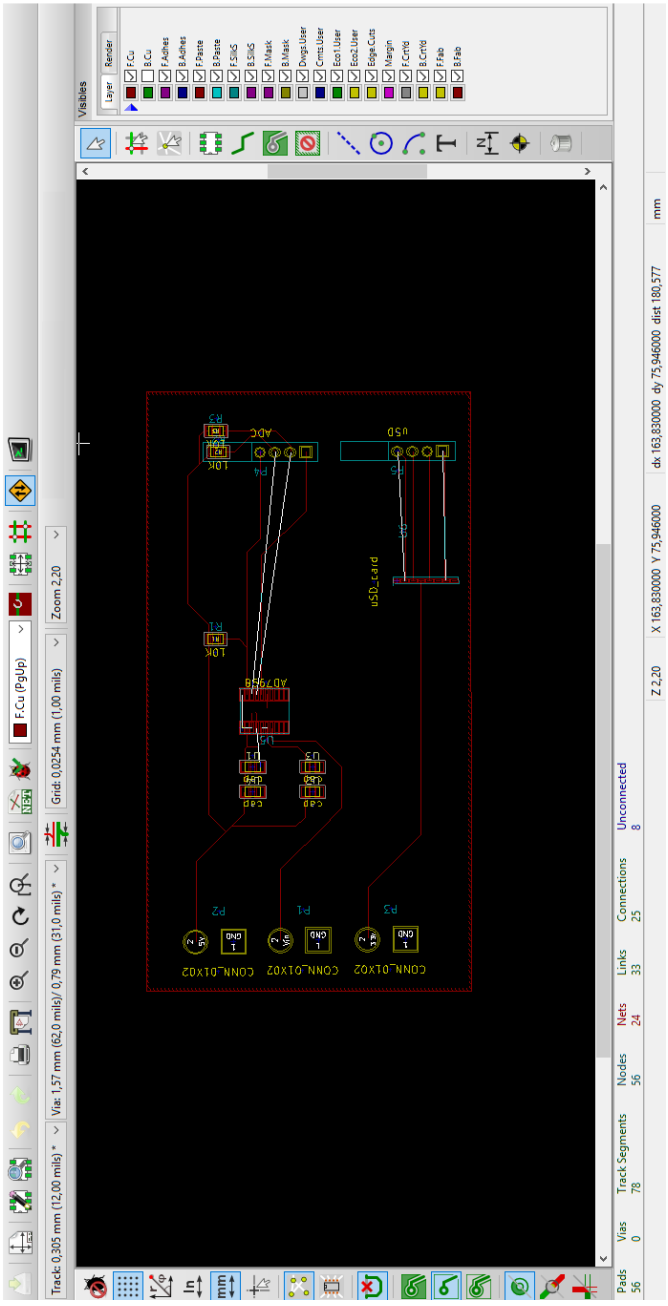
Εικόνα 20 : Αρχική οθόνη του KiCad.

Στην εικόνα 20 φαίνεται η επιφάνεια εργασίας του KiCad. Το εικονίδιο 1 είναι η συντόμευση για το Eeschema, το εικονίδιο 2 είναι η συντόμευση για το Pcbnew, ενώ στην αριστερή οθόνη βρίσκεται η λίστα των αρχείων που περιέχει το σχέδιο που αναπτύσσεται.



Εικόνα 21 : Υποπρόγραμμα Eeschema.

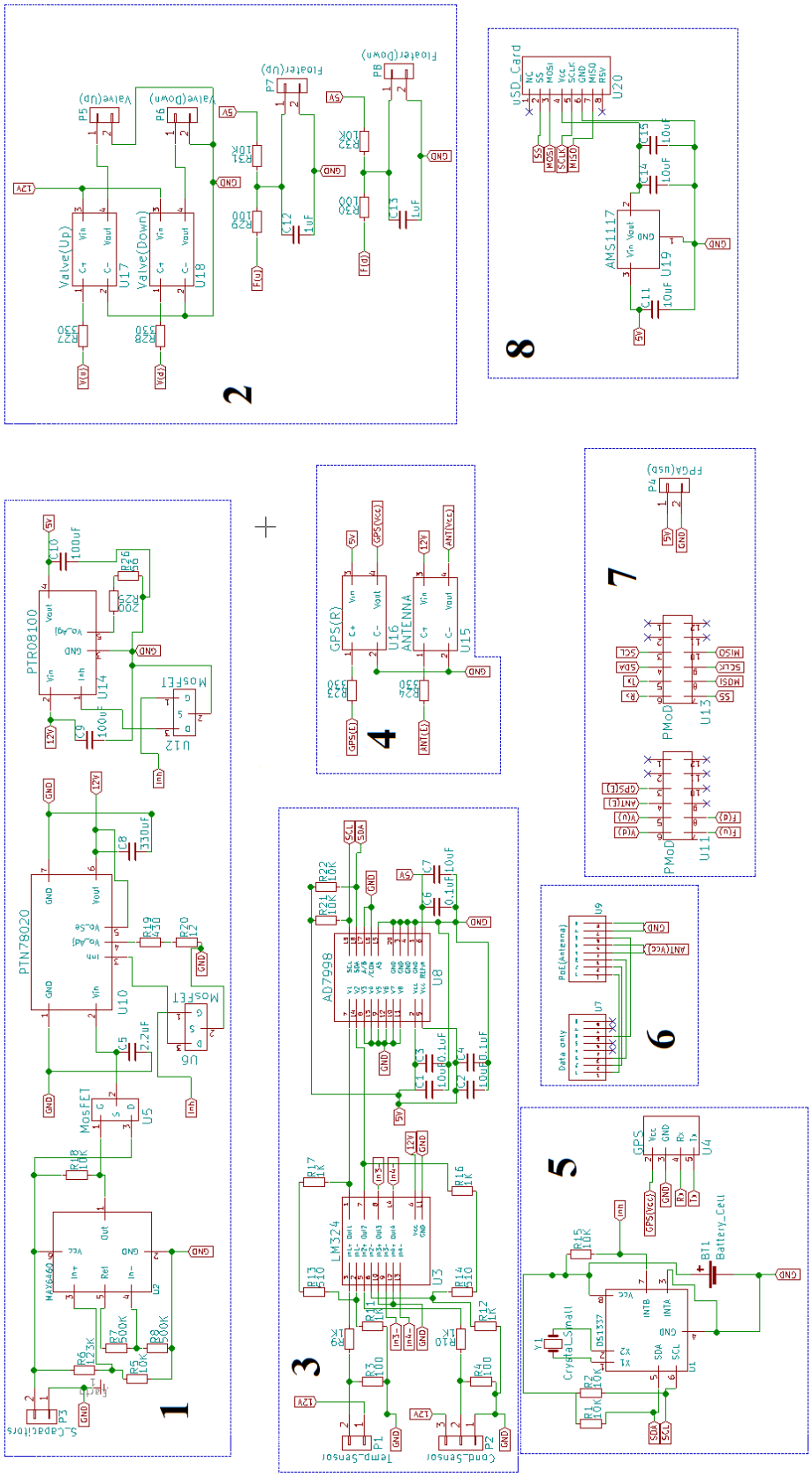
Στην εικόνα 21 φαίνεται η επιφάνεια εργασίας της εφαρμογής του KiCad Eeschema. Στο παράθυρο φαίνεται είναι η επιφάνεια σχεδιασμού του σχηματικού διαγράμματος, στο ένθετο πάνω παράθυρο είναι το εργαλείο δημιουργίας διασυνδέσεων και στο ένθετο κάτω παράθυρο φαίνεται το εργαλείο αντιστοίχησης των στοιχείων με τα footprints τους.



Εικόνα 22: Υποπρόγραμμα Pcbnew.

Στην εικόνα 22 φαίνεται η επιφάνεια εργασίας της εφαρμογής Pcbnew. Στην αριστερή πλευρά βρίσκονται τα εργαλεία ελέγχου του κυκλώματος, στην δεξιά τα εργαλεία τοποθέτησης ηλεκτρονικών στοιχείων και στο κέντρο είναι ο «καμβάς» σχεδίασης του τυπωμένου κυκλώματος.

4.2 Σχηματικό διάγραμμα συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας



Εικόνα 23 : Σχηματικό διάγραμμα συστήματος

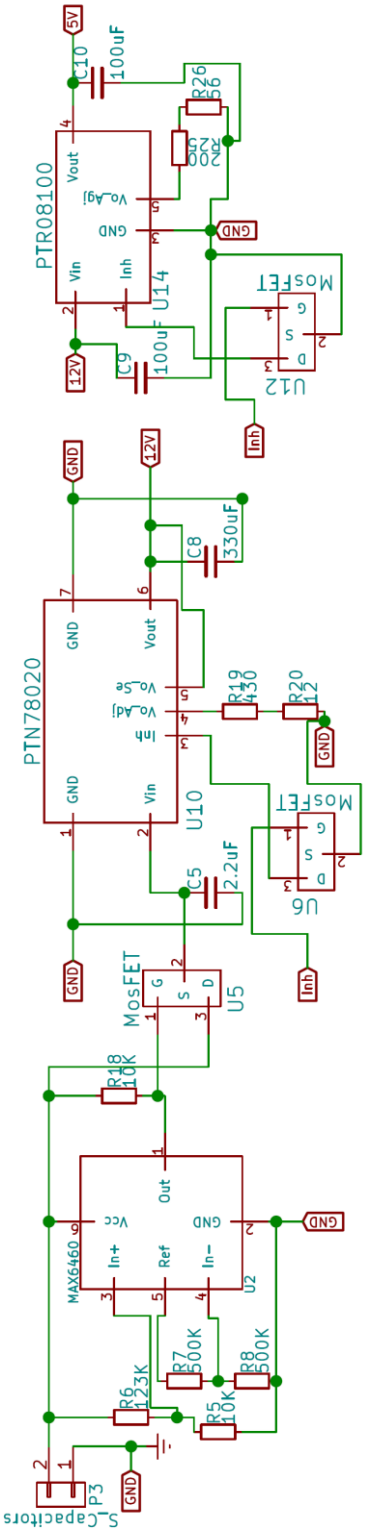
Στην εικόνα 23 φαίνεται το σχηματικό διάγραμμα του συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας, και μέσα στα πλαίσια τα υποκυκλώματα του συστήματος:

1. Κύκλωμα επιτήρησης και ρύθμισης τάσης,
2. Κύκλωμα αυτοματισμού δειγματοληψίας,
3. Κύκλωμα μετατροπής 4-20mA σε 1-5V και αναλογικού σήματος σε ψηφιακό,
4. Κύκλωμα ελέγχου τροφοδοσίας περιφερειακών,
5. Κύκλωμα συγχρονισμού,
6. Κύκλωμα μεταφοράς δεδομένων – ηλεκτρικής ενέργειας στην κεραία,
7. Κύκλωμα διασύνδεσης FPGA,
8. Κύκλωμα αποθήκευσης δεδομένων.

4.2.1 Κύκλωμα επιτήρησης και ρύθμισης τάσης

Στην εικόνα 24 απεικονίζεται το σχηματικό διάγραμμα του κυκλώματος επιτήρησης και ρύθμισης τάσης. Αποτελείται από το ολοκληρωμένο MAX6460 (U2) το οποίο επιτηρεί την τιμή της τάσης τροφοδοσίας που παρέχεται από το σύστημα που βασίζεται σε ηλιακές κυψέλες [5] και δύο ρυθμιστές τάσης (regulators), τους PTN78020 (U10) και PTR08100 (U14) της εταιρίας Texas Instruments [19][20]. Αυτοί είναι διακοπτικοί ρυθμιστές τάσης (switching regulators), υψηλής απόδοσης (έως 96%), προσφέρουν ρεύμα εξόδου έως 6A και 10A αντίστοιχα, έχουν προγραμματιζόμενη τάση εξόδου με τη χρήση αντιστάσεων και μεγάλο εύρος τάσης εισόδου. Επίσης, έχουν λειτουργία on/off(inhibit) ελαχιστοποιώντας έτσι την κατανάλωση ηλεκτρικής ενέργειας όταν δεν είναι απαραίτητη η λειτουργία τους. Χρησιμοποιούνται στο κύκλωμα για να μετατρέψουν την τάση τροφοδοσίας σε 12V και 5V αντίστοιχα. Η επιτήρηση της τάσης τροφοδοσίας από το MAX6460 γίνεται ως εξής: Η έξοδός του (ακροδέκτης Out), είναι έξοδος ανοιχτού-απαγωγού που βρίσκεται σε στάθμη “HIGH” όταν η τιμή της τάσης στον ακροδέκτη IN+ γίνει μεγαλύτερη από αυτή στον ακροδέκτη IN-, τα επίπεδα των τάσεων ρυθμίζονται από τους διαιρέτες τάσης που δημιουργήθηκαν στις εισόδους IN+ και IN-. Ο διαιρέτης τάσης που αποτελείται από τις αντιστάσεις R7, R8 διαιρεί την τάση 2.16V του ακροδέκτη Ref σε 1.08V στον ακροδέκτη IN-. Ο διαιρέτης τάσης που αποτελείται από τις αντιστάσεις R5, R6 έχει τάση εισόδου την τάση που παρέχεται από τους υπερπυκνωτές. Όταν η τιμή της είναι μεγαλύτερη ή ίση με 14.5V, η τάση στον ακροδέκτη IN+ έχει τιμή μεγαλύτερη ή ίση με 1.09V, δηλαδή μεγαλύτερη τιμή από την τάση στον ακροδέκτη IN-. Άρα ο ακροδέκτης Out μεταβαίνει σε στάθμη “HIGH”. Τότε ενεργοποιείται το τύπου-n MosFET (U5) και παρέχει την τάση τροφοδοσίας στους ρυθμιστές τάσης. Η τιμή της τάσης που ενεργοποιεί το MAX6460 είναι $V \geq 14.5V$, διότι για να μετατρέψει ο PTN78020 την τάση εισόδου του Vin σε τάση εξόδου 12V, πρέπει Vin=Vout + 2.5V. Η είσοδος Vin (ακροδέκτης 2) του PTR08100 τροφοδοτείται από την τάση εξόδου Vout (ακροδέκτης 6) του PTR78020, γιατί το εύρος τάσεων εισόδου του είναι από 4.5V έως και 14V. Για τον έλεγχο της λειτουργίας on/off (inhibit), συνδέθηκε στους ακροδέκτες INH των

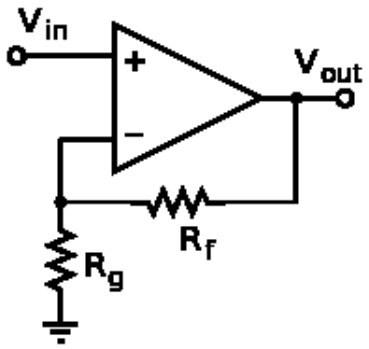
ρυθμιστών τάσης MosFET τύπου-n (U6 και U12) και στην πύλη τους συνδέθηκε ο ακροδέκτης INTB του RTC (παράγραφος 4.5). Όταν το INTB είναι λογικό ‘1’ τα MosFET ενεργοποιούνται και άγουν λογικό ‘0’ από την πηγή S στον απαγωγό D, απενεργοποιώντας έτσι τους ρυθμιστές τάσης, με αποτέλεσμα σημαντική μείωση στην κατανάλωση ηλεκτρικής ενέργειας από αυτούς.



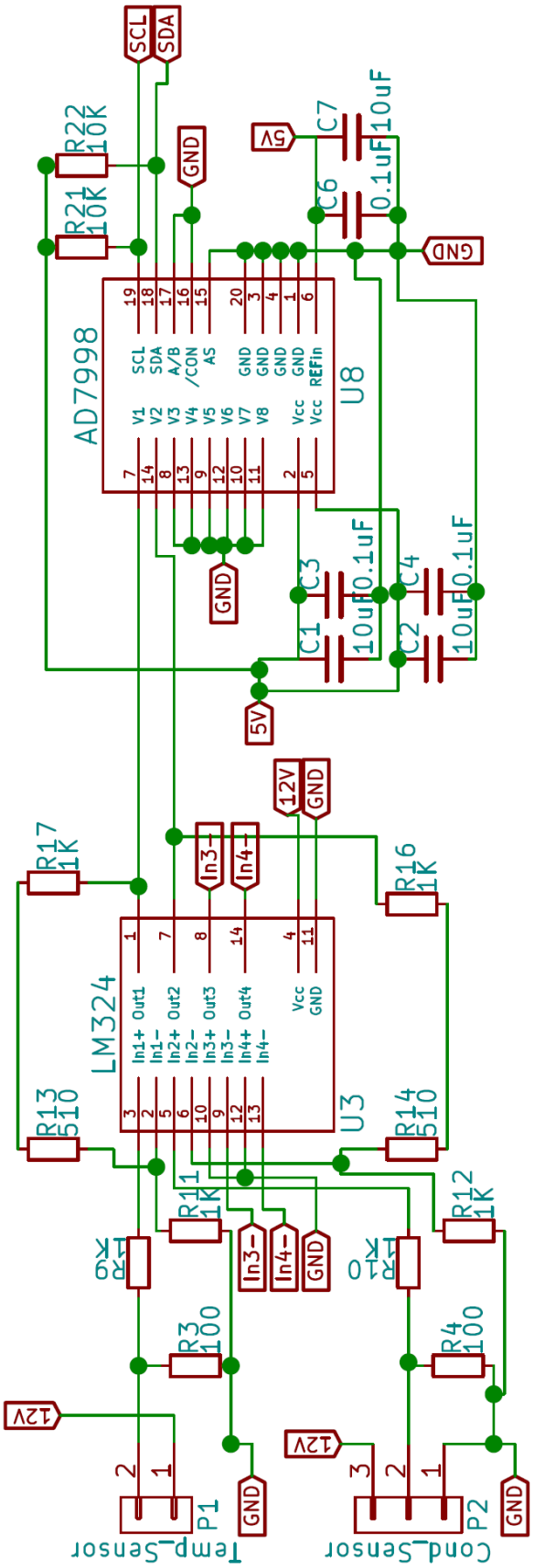
Εικόνα 24: Κύκλωμα επιτήρησης και ρύθμισης τάσης

4.2.2 Κύκλωμα μετατροπής 4-20mA σε 1-5V και αναλογικού σήματος σε ψηφιακό

Στην εικόνα 26 φαίνεται το σχηματικό διάγραμμα του βασικού κυκλώματος λήψης μετρήσεων από το σύστημα. Το κύκλωμα περιλαμβάνει την μετατροπή του ρεύματος 4-20mA σε τάση 1-5V η οποία στη συνέχεια μετατρέπεται σε ψηφιακό σήμα. Κύρια στοιχεία του κυκλώματος είναι ο τελεστικός ενισχυτής (operational amplifier) LM324AM (U3) [21] της εταιρίας Fairchild και ο μετατροπέας αναλογικού σήματος σε ψηφιακό (ADC) AD7998 (U8) της εταιρίας Analog Devices. Σκοπός του κυκλώματος είναι να μετατρέψει το αναλογικό σήμα ρεύματος 4-20mA, που παράγεται από τον αισθητήρα, αρχικά σε αναλογικό σήμα τάσης 1-5V στον τελεστικό ενισχυτή και έπειτα σε ψηφιακό σήμα στον ADC. Ο αισθητήρας συνδέεται στο σύστημα μέσω των συνδετών P1 και P2 και τροφοδοτείται από τάση 12V. Τα σήματά εξόδου του συνδέονται στις μη-ανάστροφες εισόδους, In1+ και In2+, του ενισχυτή μέσω μίας αντίστασης 1ΚΩ και μίας 100Ω ως προς τη γείωση, η οποία λειτουργεί ως φορτίο για το σήμα εισόδου, ώστε να εμφανίζεται τάση 0.4V για ρεύμα 4mA στην μη-ανάστροφη είσοδο του τελεστικού ενισχυτή. Σε κάθε μία από τις ανάστροφες εισόδους, συνδέεται μία αντίσταση 1ΚΩ και σε συνδυασμό με την αντίσταση ανάδρασης 1.5ΚΩ (1ΚΩ και 510Ω σε σειρά) ρυθμίζουν την απολαβή του τελεστικού ενισχυτή σε 2.5. Η απολαβή του ενισχυτή παίρνει αυτή την τιμή διότι: Ο τελεστικός ενισχυτής συνδέεται στο κύκλωμα ακολουθώντας συνδεσμολογία που τον καθιστά μη – αναστρέφων τελεστικό ενισχυτή, εικόνα 25. Η απολαβή ενός μη – αναστρέφων τελεστικού ενισχυτή δίνεται από το λόγο της τάσης εξόδου του διά την τάση εισόδου, A=Vout/Vin. Μετά από ανάλυση του κυκλώματος της εικόνας 25 προκύπτει ότι $A=1+(R_f / R_g)$, όπου R_f = 1.5ΚΩ και R_g = 1ΚΩ στο κύκλωμα που υλοποιήθηκε. Άρα $A=1 + (1.5ΚΩ / 1ΚΩ) \rightarrow A=2.5$. Οπότε Vout= A* Vin. Αυτό έχει ως αποτέλεσμα, τα 4mA να μετατρέπονται σε 1V και τα 20mA να μετατρέπονται σε 5V. Στη συνέχεια του κυκλώματος ο ADC δέχεται το διαμορφωμένο σήμα που εξάγεται από τους ακροδέκτες Out1 (μέτρηση θερμοκρασίας) και Out2 (μέτρηση αγωγιμότητας) του ενισχυτή στις εισόδους του, ακροδέκτες V1 και V2. Με αντίστοιχη εντολή στο λογισμικό, ξεκινάει η μετατροπή της τάσης σε ψηφιακό σήμα και μετά το πέρας της διαδικασίας οι μετρήσεις λαμβάνονται από το FPGA μέσω του πρωτοκόλλου επικοινωνίας I^2C, ακροδέκτες SDA και SCL.

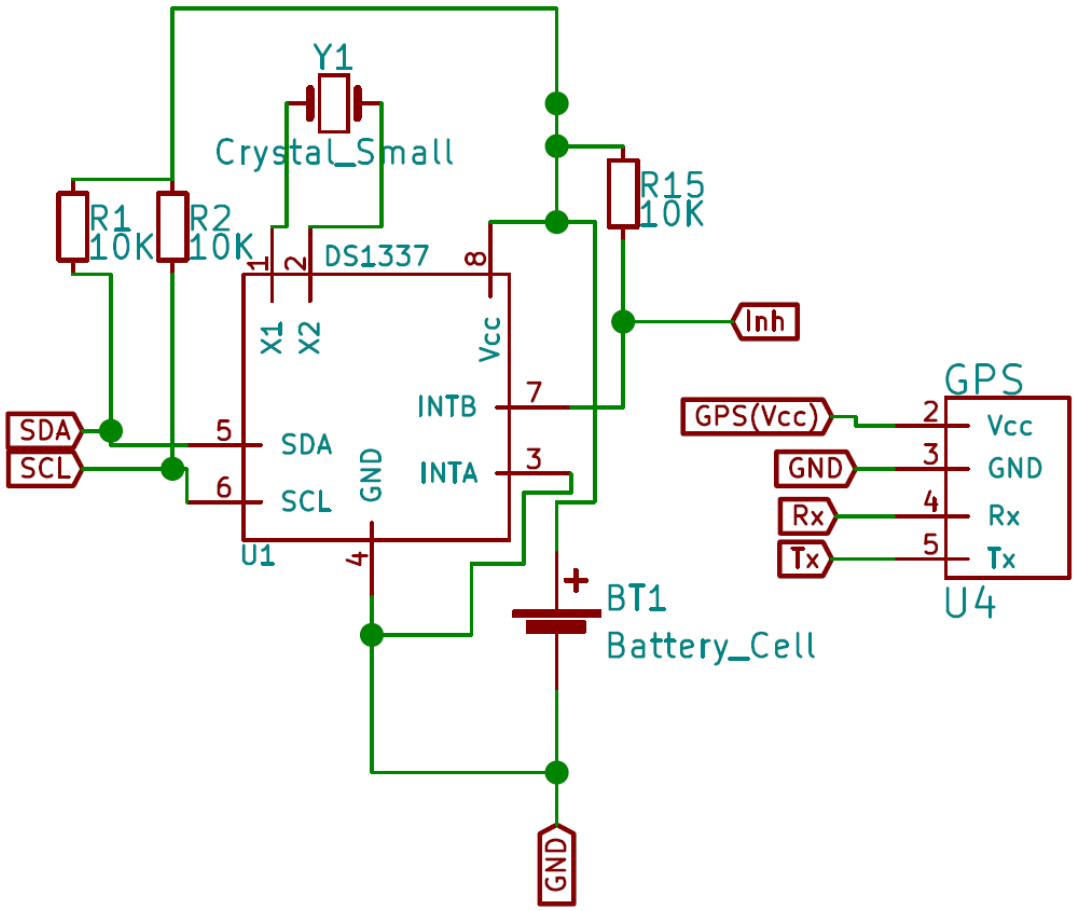


Εικόνα 25: Μη – αναστρέφων ενισχυτής



Εικόνα 26: Κύκλωμα μετατροπής 4-20mA σε 1-5V και αναλογικού σε ψηφιακό

4.2.3 Κύκλωμα συγχρονισμού

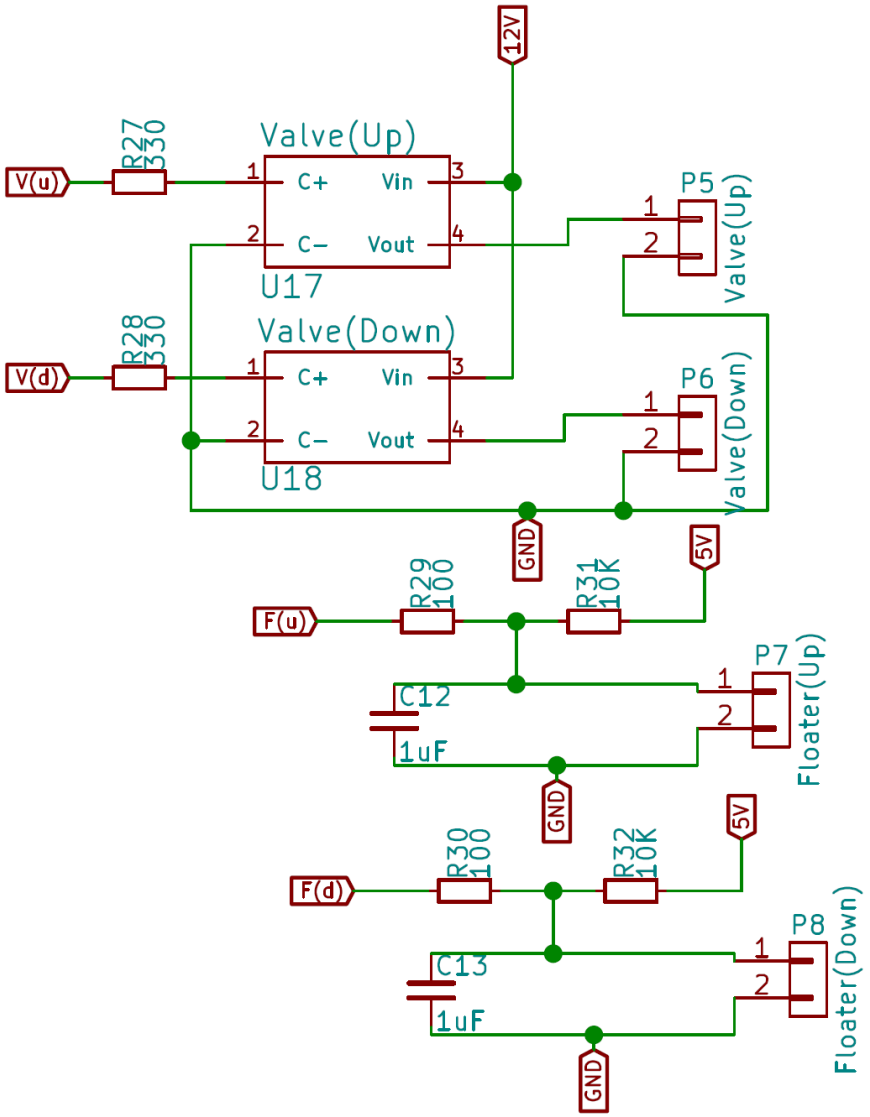


Εικόνα 27: Κύκλωμα συγχρονισμού

Στην εικόνα 27 παρουσιάζεται το σχηματικό διάγραμμα του κυκλώματος συγχρονισμού, το οποίο είναι υπεύθυνο για την αφύπνιση του συστήματος και για τον έλεγχο της ορθότητας της ώρας και ημερομηνίας του ρολογιού πραγματικού χρόνου (RTC). Αποτελείται από την ηλεκτρονική πλακέτα Adafruit Ultimate GPS (U4) και το ρολόι πραγματικού χρόνου DS 1337 της Maxim (U1), που περιγράφηκαν στην παράγραφο 3.4. Το RTC τροφοδοτείται πάντα με τάση από μία μπαταρία τύπου 18650 που συνδέεται στο κύκλωμα συγχρονισμού μέσω του κυκλώματος φόρτισης μπαταρίας (παράγραφος 4.2.9) και παρέχει τάση 3.7V, ενώ η τροφοδοσία του GPS είναι 5V και ελέγχεται από τον ηλεκτρονόμο στερεάς κατάστασης (U16) που βρίσκεται στο κύκλωμα ελέγχου τροφοδοσίας περιφερειακών (παράγραφος 4.2.8). Στο RTC (ακροδέκτες 1,2) έχει συνδεθεί ταλαντωτής 32768 KHz (Y1). Το ρολόι είναι εξυπηρετούμενη συσκευή στον δίαυλο I^2C του συστήματος και επικοινωνεί με το FPGA, που είναι ο εξυπηρετητής του διαύλου, μέσω των ακροδεκτών SDA και SCL στους οποίους έχουν συνδεθεί αντιστάσεις pull-up, διότι είναι έξοδοι ανοιχτού απαγωγού (open drain outputs). Ο ακροδέκτης INTA έχει συνδεθεί στη γείωση καθώς δεν χρησιμοποιείται. Ο ακροδέκτης INTB συνδέεται με τις πύλες των MosFET του κυκλώματος επιτήρησης-ρύθμισης τάσης και έχει δύο λειτουργίες: παραγωγή τετραγωνικού παλμού προγραμματιζόμενης συχνότητας και λειτουργία

παραγωγής παλμού διακοπής (interrupt), όταν οι τιμές στους καταχωρητές ώρας συμπίπτουν με την τιμή των καταχωρητών του συναγερμού (Alarm 2) του RTC. Στο σύστημα που αναπτύχθηκε χρησιμοποιείται η δεύτερη λειτουργία με τον εξής τρόπο: Ο ακροδέκτης INTB, όταν δεν συμπίπτουν οι τιμές των καταχωρητών ώρας και του συναγερμού, παράγει σήμα στάθμης λογικού '1' με αποτέλεσμα την ενεργοποίηση των MosFET, τη σύνδεση του ακροδέκτη INH των ρυθμιστών τάσης, μέσω των MosFET, στη γείωση του τυπωμένου κυκλώματος, απενεργοποιώντας έτσι τους ρυθμιστές τάσης. Όταν υπάρχει ταύτιση των τιμών, παράγεται σήμα στάθμης λογικού '0', τα MosFET απενεργοποιούνται, ξεκινάει η λειτουργία των ρυθμιστών τάσης και το σύστημα ενεργοποιείται. Στο κύκλωμα συνδέονται μόνο τέσσερις από τους εννιά ακροδέκτες της πλακέτας με GPS διότι οι άλλοι δεν χρειάζονται. Τέλος, το GPS επικοινωνεί με το FPGA μέσω των ακροδεκτών Rx και Tx, μέσω του πρωτοκόλλου επικοινωνίας UART (universal asynchronous receiver-transmitter) που περιγράφεται στο παράρτημα Θ.

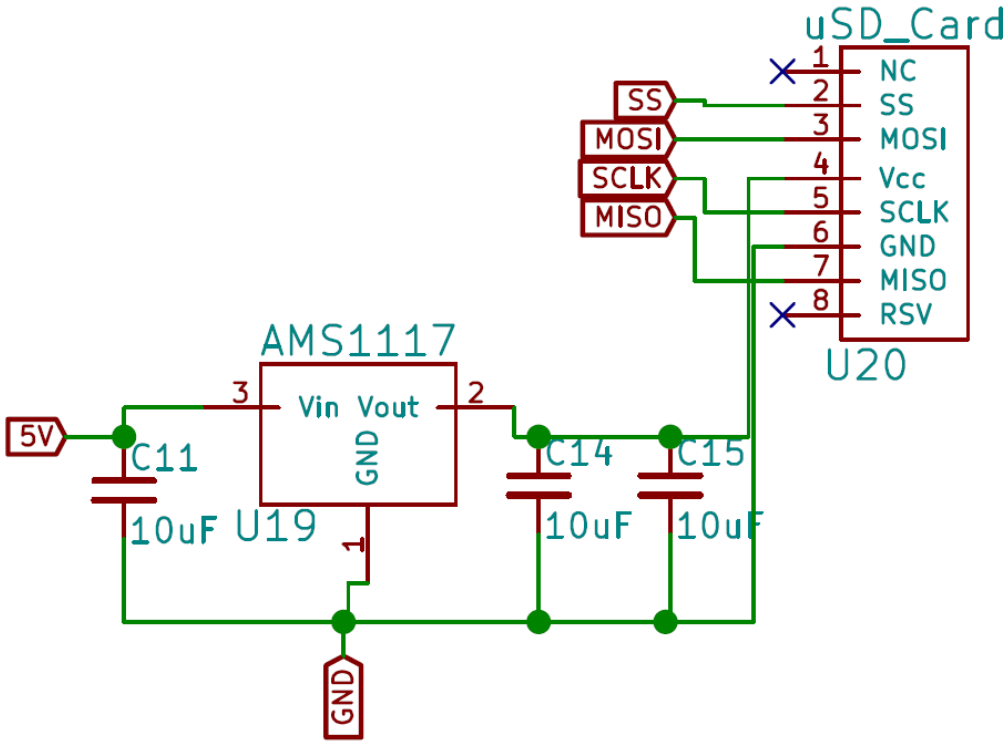
4.2.4 Κύκλωμα αυτοματισμού δειγματοληψίας



Εικόνα 28: Κύκλωμα αυτοματισμού δειγματοληψίας

Στην εικόνα 28 απεικονίζεται το σχηματικό διάγραμμα του κυκλώματος αυτοματισμού δειγματοληψίας. Είναι υπεύθυνο για το γέμισμα και άδειασμα του δοχείου δειγματοληψίας με νερό, μέσω του ελέγχου της τροφοδοσίας των ηλεκτρικών βαλβίδων που είναι προσαρτημένες στο δοχείο δειγματοληψίας και συνδέονται με το κύκλωμα μέσω των συνδετών - κλεμών P5 και P6, καθώς και τον έλεγχο της στάθμης του νερού στο δοχείο μέσω των αισθητήρες στάθμης, που βρίσκονται μέσα σε αυτό και συνδέονται με το κύκλωμα μέσω των συνδετών - κλεμών P7 και P8. Η τάση τροφοδοσίας των ηλεκτρικών βαλβίδων είναι 12V και ο έλεγχος της γίνεται μέσω των ηλεκτρονόμων στερεάς κατάστασης (U17 και U18), που συνδέονται στο FPGA και ελέγχονται από συνεχή τάση 3.3V. Οι ηλεκτρικές βαλβίδες χρησιμοποιούν διασύνδεση τριών καλωδίων, ένα για την τροφοδοσία τους, ένα για την ενεργοποίηση της λειτουργίας τους και ένα για σύνδεση με την γείωση του τυπωμένου κυκλώματος. Οι αισθητήρες στάθμης ελέγχουν την στάθμη του νερού συνδέοντας ή αποσυνδέοντας το κύκλωμα το οποίο τίθεται σε λογική στάθμη ‘0’ ή ‘1’ με τη βοήθεια αντιστάσεων pull-up, τις R31, R32.

4.2.5 Κύκλωμα αποθήκευσης δεδομένων

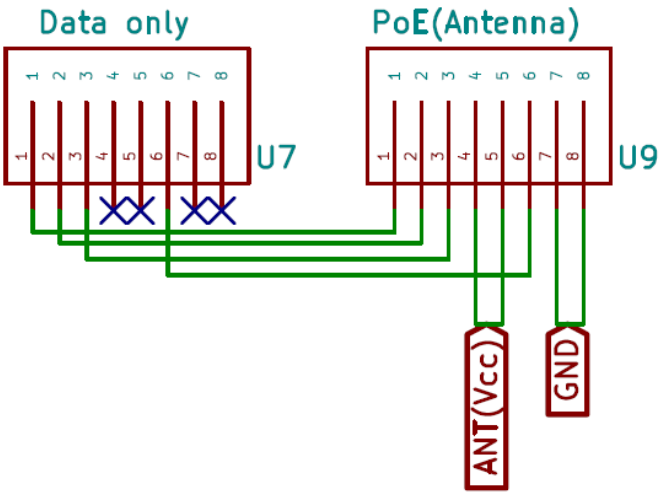


Εικόνα 29: Κύκλωμα αποθήκευσης δεδομένων

Στην εικόνα 29 απεικονίζεται το σχηματικό διάγραμμα του κυκλώματος αποθήκευσης δεδομένων. Τα κύρια στοιχεία του είναι μία κάρτα μνήμης μSD της SanDisk και ο ρυθμιστής τάσης AMS1117 της Advanced Monolithic Systems [22] (U19), ο οποίος ρυθμίζει την τάση τροφοδοσίας (Vcc) της κάρτας μνήμης σε 3.3V. Η

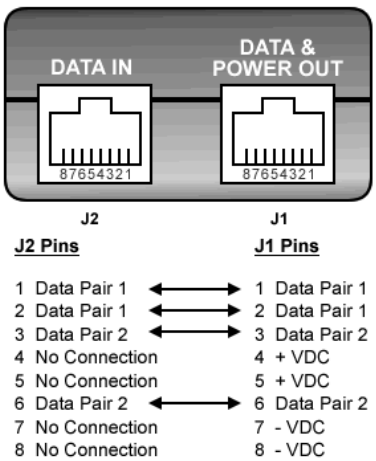
κάρτα επικοινωνεί με το FPGA μέσω των ακροδεκτών SS, MOSI, SCLK και MISO, μέσω του πρωτοκόλλου επικοινωνίας SPI. Σε αυτή γίνεται η αποθήκευση των τιμών των φυσικοχημικών παραμέτρων του νερού που λαμβάνονται από τον αισθητήρα του συστήματος.

4.2.6 Κύκλωμα μεταφοράς δεδομένων – ηλεκτρικής ενέργειας στην κεραία



Εικόνα 30: Κύκλωμα μεταφοράς δεδομένων – ηλεκτρικής ενέργειας στην κεραία

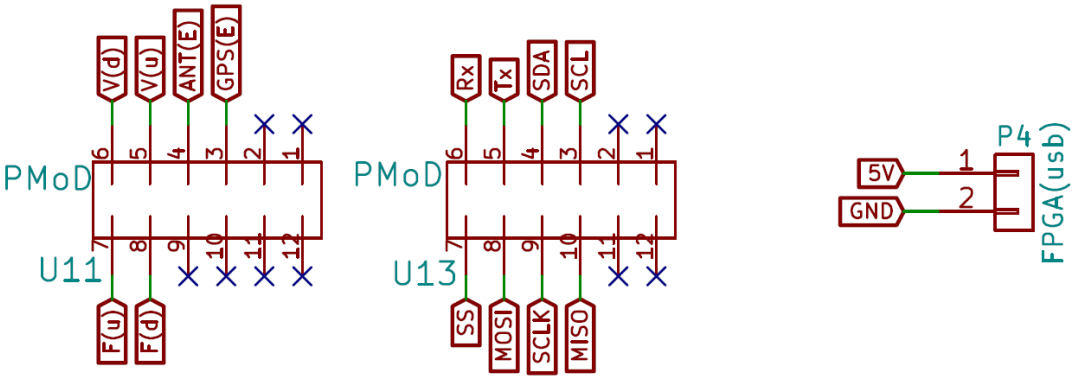
Στην εικόνα 30 φαίνεται το σχηματικό διάγραμμα του κυκλώματος μεταφοράς δεδομένων και ηλεκτρικής ενέργειας στην κεραία. Είναι υπεύθυνο για την προώθηση των αποθηκευμένων μετρήσεων μέσω του διακομιστή TFTP και την απαιτούμενη τροφοδοσία στην κεραία Wi-Fi που συνδέει τον σταθμό με τον τερματικό κόμβο διαδικτύου που βρίσκεται στο Πανεπιστήμιο Ιωαννίνων. Η τάση τροφοδοσίας της κεραίας ελέγχεται από ηλεκτρονόμο στερεάς κατάστασης που περιγράφεται στην παράγραφο 4.2.8 και έχει τιμή 12V.



Εικόνα 31: Διασύνδεση υποδοχών RJ-45

Το κύκλωμα αποτελείται από δύο υποδοχείς RJ-45 (που αντιστοιχούν στα U7και U9 της εικόνας 30) και η διασύνδεση τους έγινε σύμφωνα με την εικόνα 31. Είναι διασύνδεση τύπου PoE (Power over Ethernet) και επιτρέπει την μεταφορά, όχι μόνο δεδομένων αλλά και τη μεταφορά ηλεκτρικής ενέργειας μέσω ενός καλωδίου UTP (unshielded twisted pair).

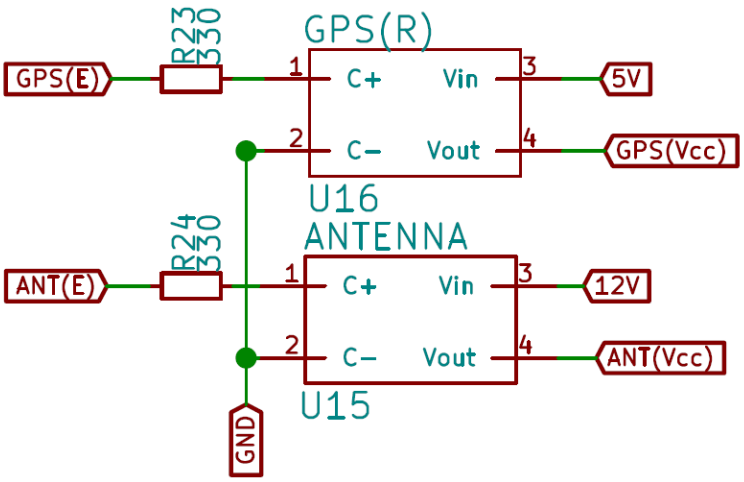
4.2.7 Κύκλωμα διασύνδεσης FPGA



Εικόνα 32: Κύκλωμα ελέγχου τροφοδοσίας περιφερειακών

Στην εικόνα 32 παρουσιάζεται το σχηματικό διάγραμμα του κυκλώματος διασύνδεσης του FPGA με το υπόλοιπο σύστημα. Αποτελείται από τους συνδέτες PMoD (U11 και U13) της εταιρίας Digilent [24] και μία κλέμα – συνδέτη (P4) που τροφοδοτεί το FPGA με τάση 5V, μέσω ενός καλωδίου τύπου USB micro-b. Οι ακροδέκτες 1, 2, 11, 12 των δύο συνδετών έχουν μείνει ασύνδετοι διότι δεν χρησιμοποιούνται στο κύκλωμα. Οι ακροδέκτες 3, 4, 5, 6 του U11 ελέγχουν τους ηλεκτρονόμους στερεάς κατάστασης του συστήματος και οι 7, 8 είναι οι είσοδοι για το σήμα των αισθητήρων στάθμης του νερού στο δοχείο δειγματοληψίας (παράγραφος 4.2.4). Τέλος, οι ακροδέκτες του U13 είναι υπεύθυνοι για τη διασύνδεση του FPGA με τους ακροδέκτες των περιφερειακών του συστήματος που χρησιμοποιούν πρωτόκολλα επικοινωνίας.

4.2.8 Κύκλωμα ελέγχου τροφοδοσίας περιφερειακών

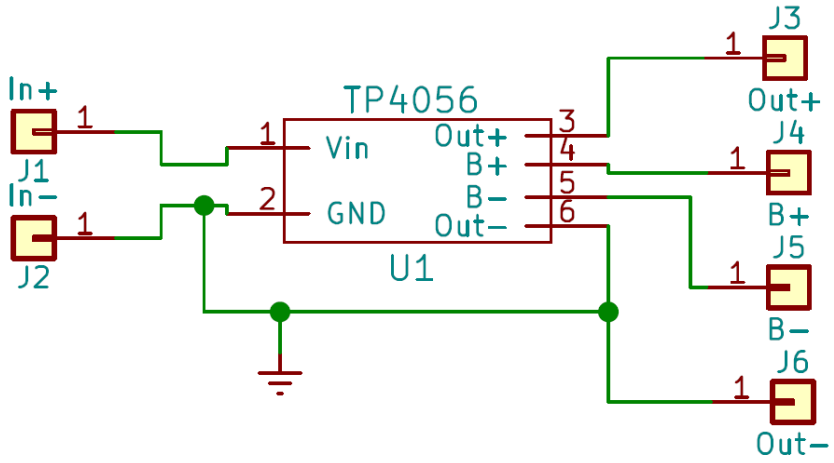


Εικόνα 33: Κύκλωμα ελέγχου τροφοδοσίας περιφερειακών

Τέλος, στην εικόνα 33 απεικονίζεται το σχηματικό διάγραμμα του κυκλώματος τροφοδοσίας περιφερειακών. Τα βασικά του στοιχεία είναι οι ηλεκτρονόμοι στερεάς κατάστασης AQY282S της Panasonic, οι οποίοι ελέγχουν την τροφοδοσία του GPS (U16) και την τροφοδοσία της κεραίας (U15).

4.2.9 Κύκλωμα φόρτισης μπαταρίας

Για την τροφοδοσία του ρολογιού πραγματικού χρόνου του κυκλώματος συγχρονισμού χρησιμοποιείται η μπαταρία ICR 18650 της εταιρίας Samsung. Η μπαταρία είναι τύπου 18650 και έχει χωρητικότητα 2400mAh, η οποία επαρκεί για την συνεχή τροφοδοσία του RTC με ηλεκτρική ενέργεια για 316 ημέρες. Για να μην είναι απαραίτητη η αλλαγή της μπαταρίας υλοποιήθηκε το κύκλωμα φόρτισης μπαταρίας που παρουσιάζεται στην εικόνα 34. Αποτελείται από τη μπαταρία και ηλεκτρονική πλακέτα η οποία περιέχει το ολοκληρωμένο TP4056 [34]. Το TP4056 είναι ένας γραμμικός φορτιστής συνεχούς τάσης/ρεύματος για μπαταρίες ιόντων λιθίου. Ο θετικός πόλος της μπαταρίας συνδέεται στον ακροδέκτη B+ και ο αρνητικός στον ακροδέκτη B-. Η τάση της μπαταρίας παρέχεται στο RTC μέσω των ακροδεκτών OUT+ και OUT-, οι οποίοι συνδέονται μέσω καλωδίων στο κύκλωμα συγχρονισμού στους ακροδέκτες Battery Cell +/-. Για την λειτουργία του TP4056 απαιτείται τάση 5V, η οποία παρέχεται στο κύκλωμα μέσω του συνδέτη P4 (εικόνα 32, παράγραφος 4.2.7).



Εικόνα 34: Κύκλωμα φόρτισης μπαταρίας

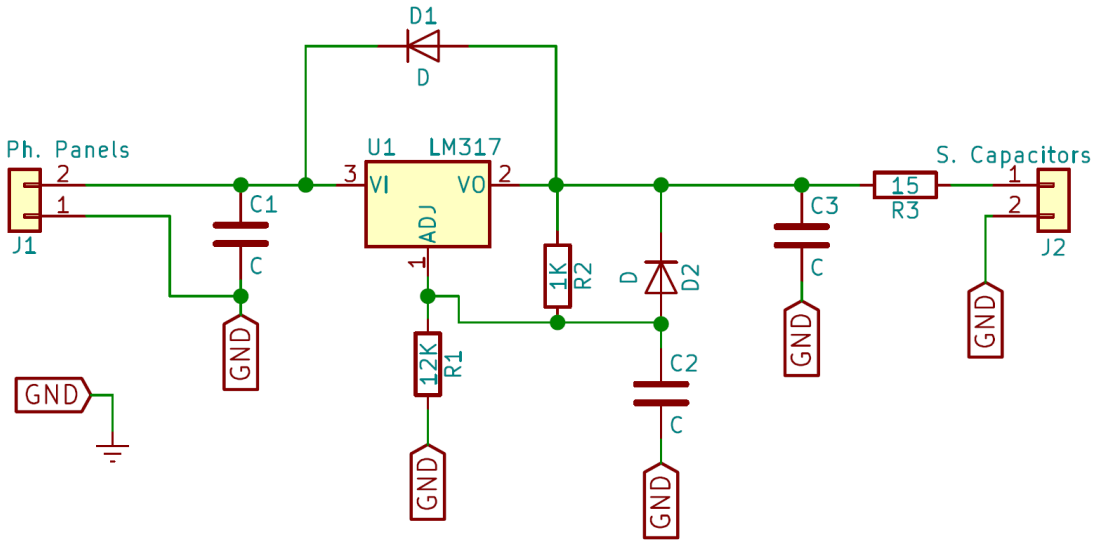
Στον συνδέτη P4 συνδέονται οι ακροδέκτες IN+, IN- μέσω καλωδίων. Η φόρτιση της μπαταρίας γίνεται κάθε φορά που ενεργοποιείται το σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας, δηλαδή πέντε φορές την ημέρα και διαρκεί για όσο χρόνο το σύστημα παραμένει σε λειτουργία. Το ρεύμα φόρτισης της μπαταρίας κυμαίνεται από 200mA – 300mA και η τάση φόρτισης είναι 5V.

4.2.10 Κύκλωμα σταθεροποίησης τάσης φόρτισης των υπερπυκνωτών

Η σύνδεση των φωτοβολταϊκών στοιχείων με τους υπερπυκνωτές γίνεται μέσω του κυκλώματος σταθεροποίησης τάσης φόρτισης των υπερπυκνωτών, του οποίου το σχηματικό διάγραμμα φαίνεται στην εικόνα 35. Η τάση των φωτοβολταϊκών στοιχείων, δηλαδή η τάση φόρτισης των υπερπυκνωτών, ανάλογα με την ηλιοφάνεια κυμαίνεται από 5V έως 22V. Όταν η τάση είναι μικρότερη των 17V το ρεύμα που παρέχεται είναι μικρό, ενδεικτικά στα 7V μόνο 5mA, και η απόδοση της διαδικασίας φόρτισης είναι ασήμαντη. Επίσης, για να μην προκληθεί βλάβη στους υπερπυκνωτές η τάση φόρτισης δεν πρέπει να υπερβαίνει τα 16.2 V. Για το λόγο αυτό χρησιμοποιείται ο ρυθμιστής τάσης LM317 (U1) της εταιρίας Texas Instruments [35]. Με χρήση των αντιστάσεων R1=12KΩ, R2=1KΩ η τάση εισόδου Vin ρυθμίζεται σε τάση εξόδου Vout:

- 16.2 V, όταν η τάση εισόδου είναι μεγαλύτερη από 16.2 V,
- Ίση με την τάση εισόδου, όταν η τάση εισόδου είναι μικρότερη από 16.2V.

Για να μην υπάρξει διαρροή ηλεκτρικού ρεύματος από τους υπερπυκνωτές στα φωτοβολταϊκά στοιχεία χρησιμοποιήθηκε η δίοδος αντεπιστροφής D1. Διαρροή ηλεκτρικού ρεύματος από τους υπερπυκνωτές προς τα φωτοβολταϊκά στοιχεία μπορεί να υπάρξει, διότι: Όσο η τάση των φωτοβολταϊκών είναι μεγαλύτερη από την τάση στα άκρα των υπερπυκνωτών η φορά του ρεύματος είναι από τα φωτοβολταϊκά προς τους υπερπυκνωτές. Όταν όμως η τάση των φωτοβολταϊκών γίνει μικρότερη από αυτή των υπερπυκνωτών, για παράδειγμα κατά τη διάρκεια της νύχτας, το ρεύμα



Εικόνα 35: Κύκλωμα σταθεροποίησης τάσης φόρτισης των υπερπυκνωτών

θα προσπαθήσει να αλλάξει τη φορά του χωρίς όμως αυτό να είναι δυνατό γιατί εμποδίζεται από την πολωμένη ανάστροφα δίοδο D1.

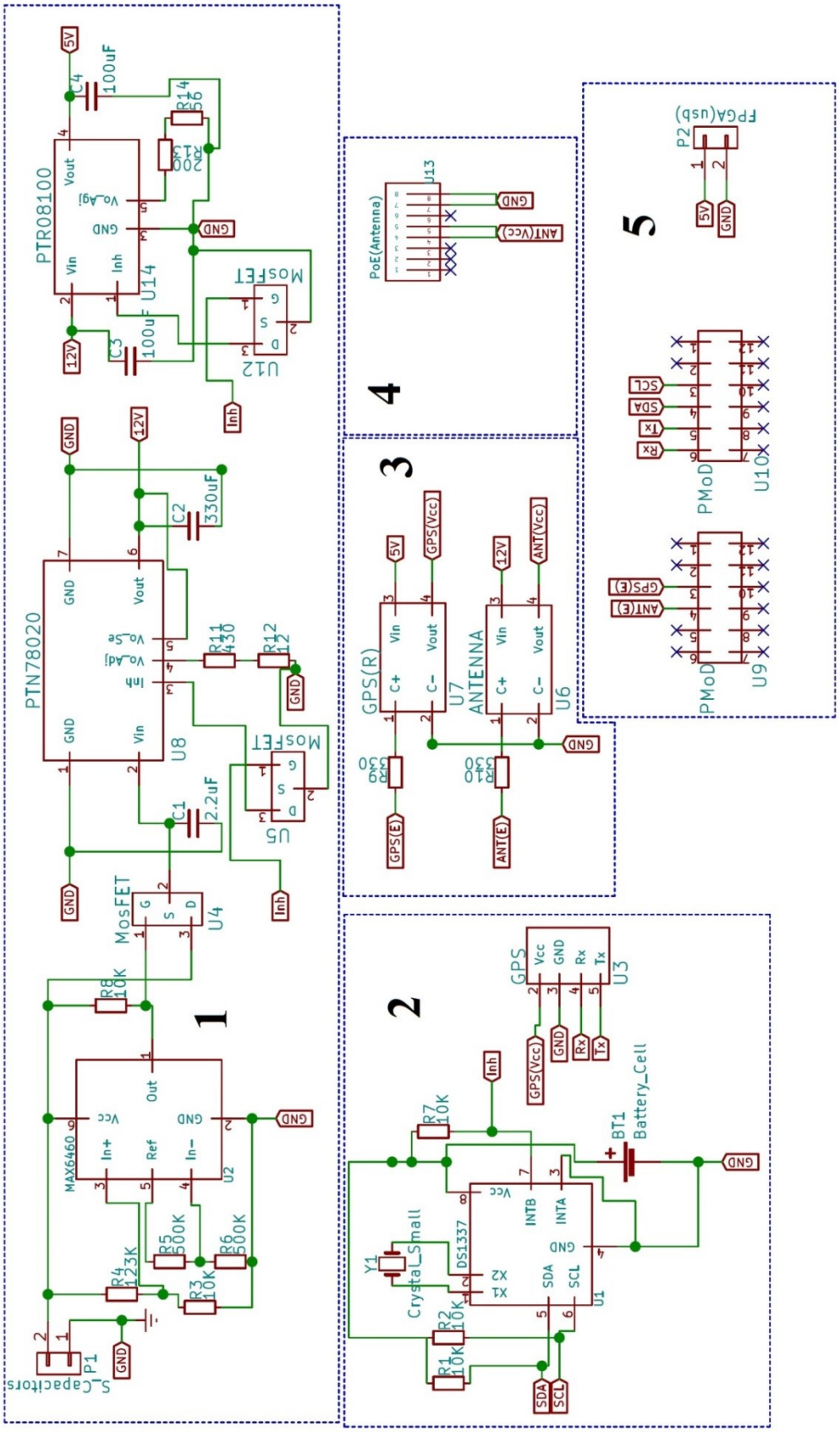
4.3 Σχηματικό διάγραμμα συστήματος αναμετάδοσης πληροφορίας

Όπως αναφέρθηκε στην παράγραφο 1.2 το σύστημα έγκαιρης προειδοποίησης που αναπτύχθηκε, εκτός από το σύστημα δειγματοληψίας- μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας περιλαμβάνει και σύστημα αναμεταδοτών . Στην εικόνα 36 παρουσιάζεται το σχηματικό διάγραμμα του ηλεκτρονικού μέρους του συστήματος αναμετάδοσης πληροφορίας και μέσα στα πλαίσια τα υποκυκλώματα του συστήματος:

1. Κύκλωμα επιτήρησης και ρύθμισης τάσης,
2. Κύκλωμα συγχρονισμού,
3. Κύκλωμα ελέγχου τροφοδοσίας περιφερειακών,
4. Κύκλωμα μεταφοράς ηλεκτρικής ενέργειας στην κεραία,
5. Κύκλωμα διασύνδεσης FPGA.

Τα παραπάνω κυκλώματα είναι ίδια με αυτά του συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – αναμετάδοσης πληροφορίας και περιγραφή τους υπάρχει στις αντίστοιχες παραγράφους. Διαφορές υπάρχουν στα κυκλώματα με αριθμό 4 και 5. Συγκεκριμένα:

- Κύκλωμα 4: Δεν υλοποιείται διακομιστής TFTP, οπότε δεν χρειάζεται μεταφορά δεδομένων από το FPGA στην κεραία. Γι’ αυτό υπάρχει μία μόνο υποδοχή RJ45 που μεταφέρει ηλεκτρική ενέργεια στην κεραία μέσω καλωδίου UTP.
- Κύκλωμα 5: Δεν υλοποιείται δίαυλος SPI οπότε οι ακροδέκτες 7,8,9,10 του U10 είναι ασύνδετοι. Επίσης οι ακροδέκτες 5,6,7,8 του U9 είναι ασύνδετοι διότι δεν συνδέονται ηλεκτρικές βαλβίδες και αισθητήρες στάθμης στο σύστημα.



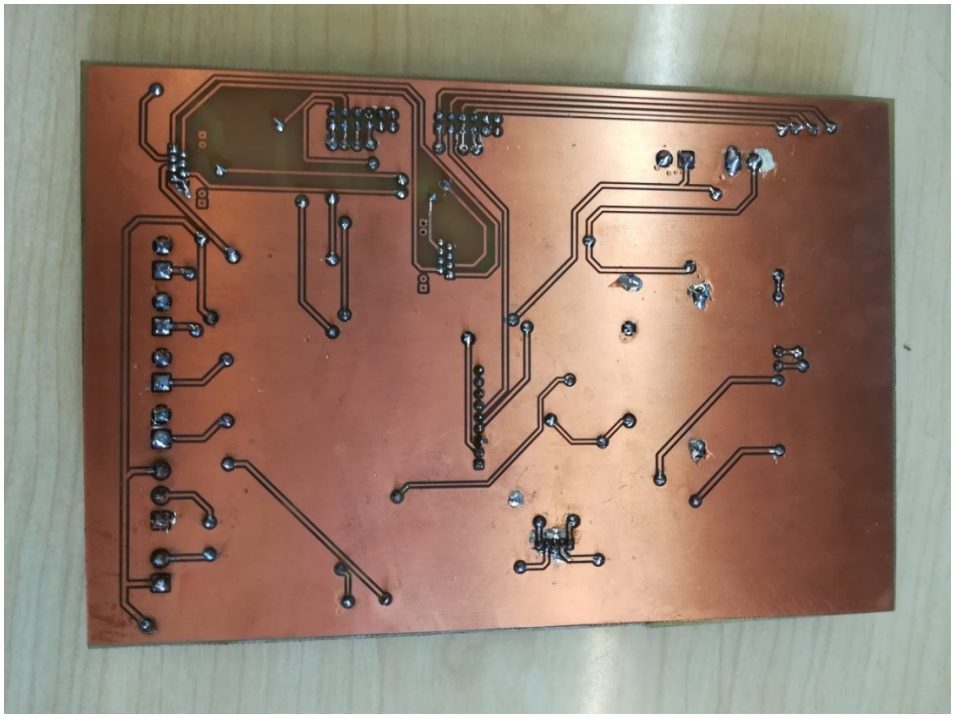
Εικόνα 36: Σχηματικό διάγραμμα συστήματος

5. Υλοποίηση συστήματος

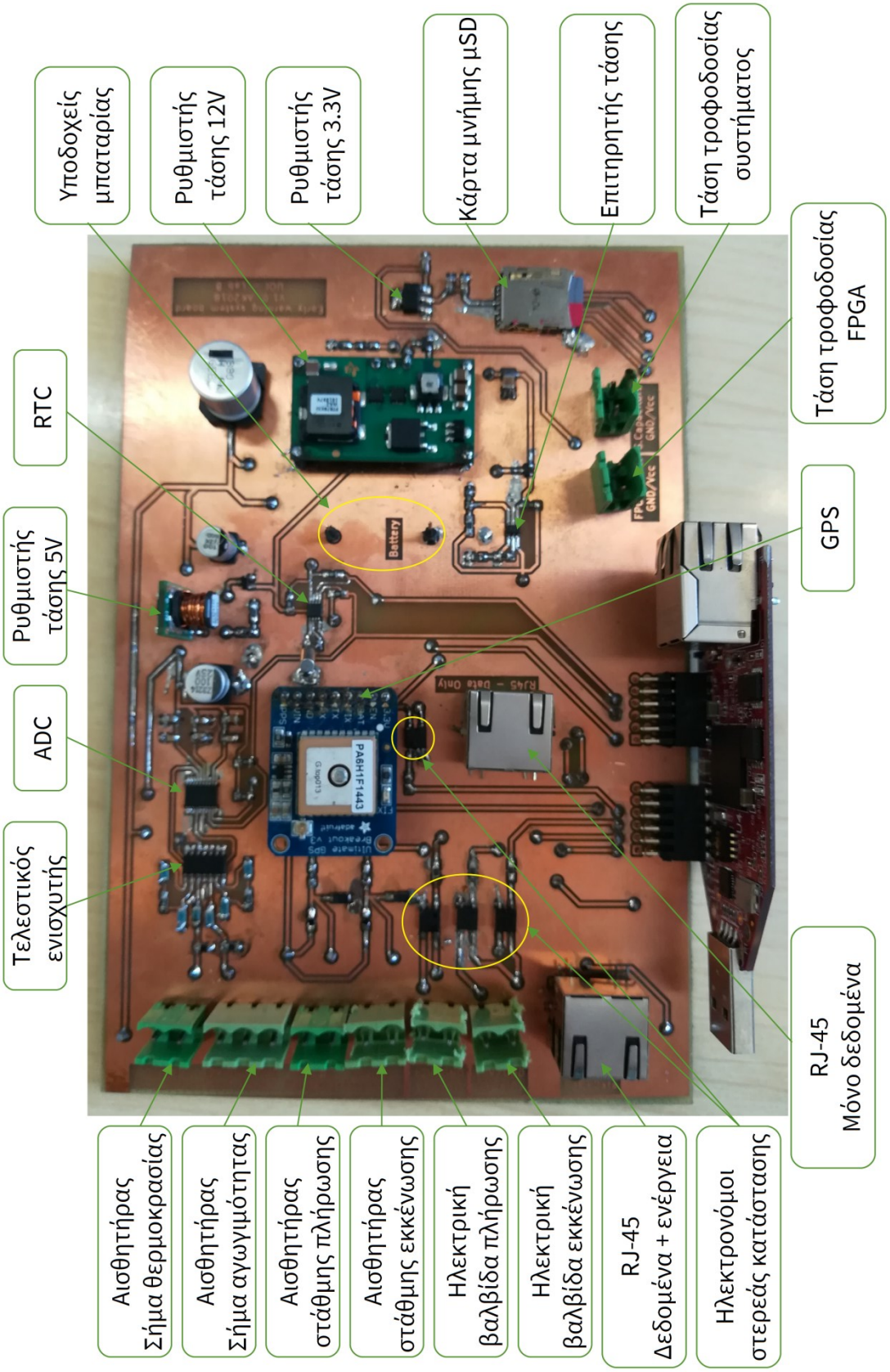
5.1 Ηλεκτρονικές πλακέτες συστήματος

Το σύστημα δειγματοληψίας-μέτρησης-αποθήκευσης-μετάδοσης υλοποιήθηκε σε μία ηλεκτρονική κάρτα διπλής όψης. Για την σχεδίαση της χρησιμοποιήθηκε η εφαρμογή Pcbnew του σχεδιαστικού πακέτου KiCad. Η εκτύπωσή της έγινε στο Εργαστήριο Υψηλών Ενεργειών του Τμήματος Φυσικής ακολουθώντας την εξής διαδικασία: Αρχικά, οι δύο όψεις του σχεδίου του τυπωμένου κυκλώματος εκτυπώθηκαν σε δύο διαφάνειες, μεγέθους Α4 οι οποίες τοποθετήθηκαν στις δύο πλευρές φωτοευαίσθητης πλακέτας. Με χρήση σαρωτή υπεριώδους ακτινοβολίας το κύκλωμα αποτυπώθηκε στην φωτοευαίσθητη πλακέτα. Η εμφάνιση του κυκλώματος έγινε με χρήση υδατικού διαλύματος υπεροξειδίου του καλίου (K_2O_2). Τέλος, η αποχάλκωση της πλακέτας έγινε με διάλυμα αποτελούμενο από: 350ml νερό, 120ml υδροχλωρικό οξύ (HCl) και 80ml υπεροξείδιο του υδρογόνου (H_2O_2). Οι διαστάσεις της ηλεκτρονικής πλακέτας που κατασκευάστηκε είναι 17.9x11.9x0.2 cm. Οι διαστάσεις του ηλεκτρονικού μέρους του συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης, δηλαδή της ηλεκτρονικής πλακέτας Avnet Spartan-6 LX9 Microboard προσαρτημένης στην ηλεκτρονική πλακέτα που κατασκευάστηκε, είναι 17.9x11.9x3.5 cm.

Στην εικόνα 37 φαίνεται σε φωτογραφία η κάτω όψη της ηλεκτρονικής πλακέτας και στην εικόνα 38 σε φωτογραφία η πάνω όψη.

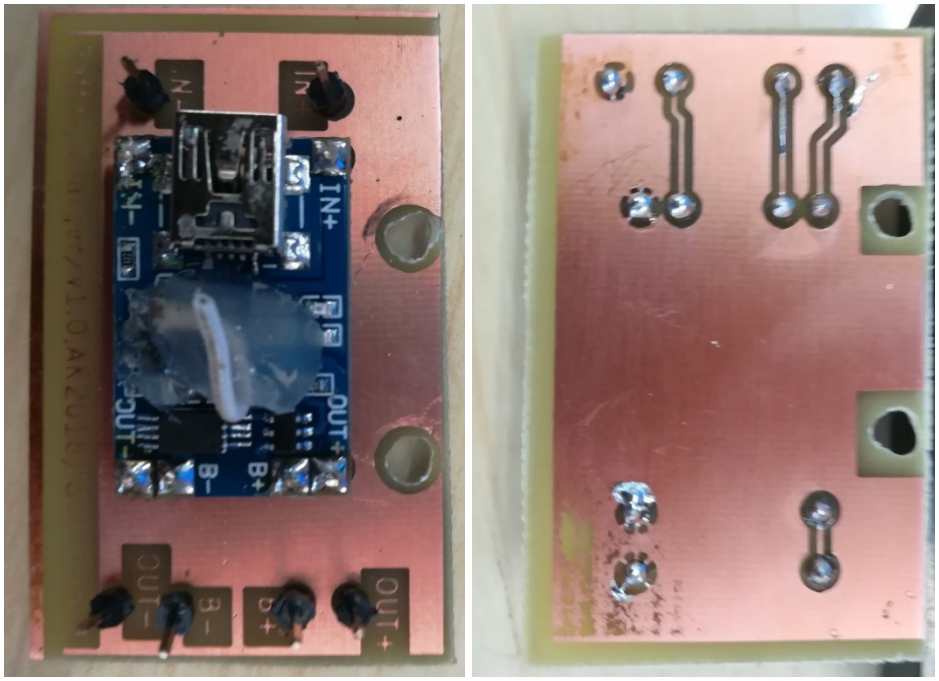


Εικόνα 37: Ηλεκτρονική πλακέτα συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας – κάτω όψη

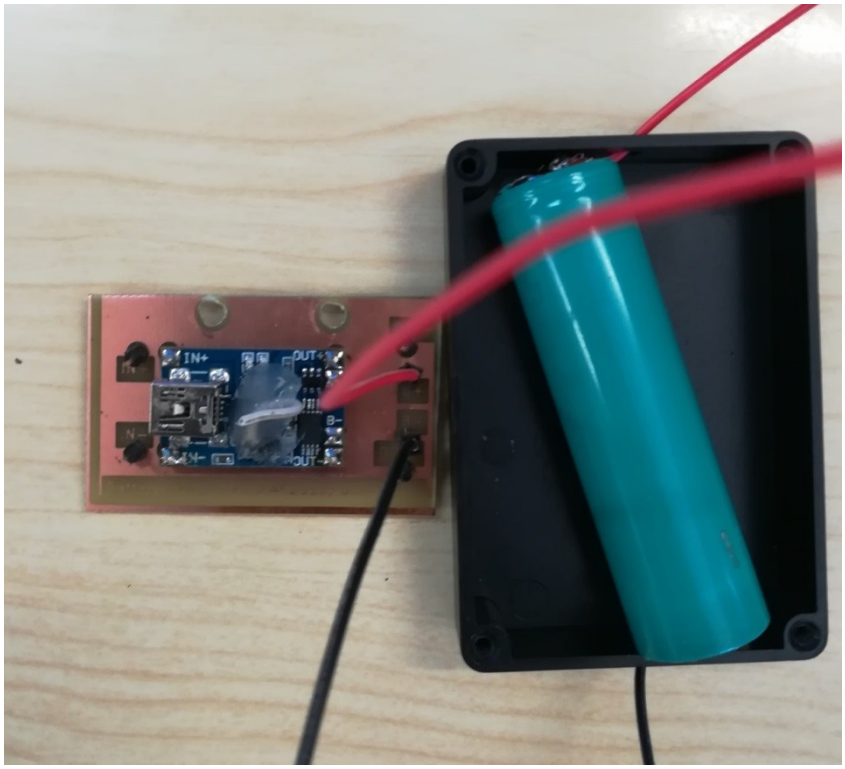


Εικόνα 38: Ηλεκτρονική πλακέτα συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας – πάνω όψη

Για την φόρτιση της μπαταρίας που τροφοδοτεί το ρολόι πραγματικού χρόνου δημιουργήθηκε η ηλεκτρονική πλακέτα διπλής όψης που φαίνεται στην εικόνα 39 (πάνω/κάτω όψη) και στην εικόνα 40 μαζί με την μπαταρία τοποθετημένη σε θήκη. Οι διαστάσεις της πλακέτας είναι 4.93x3.04x1.2 cm.

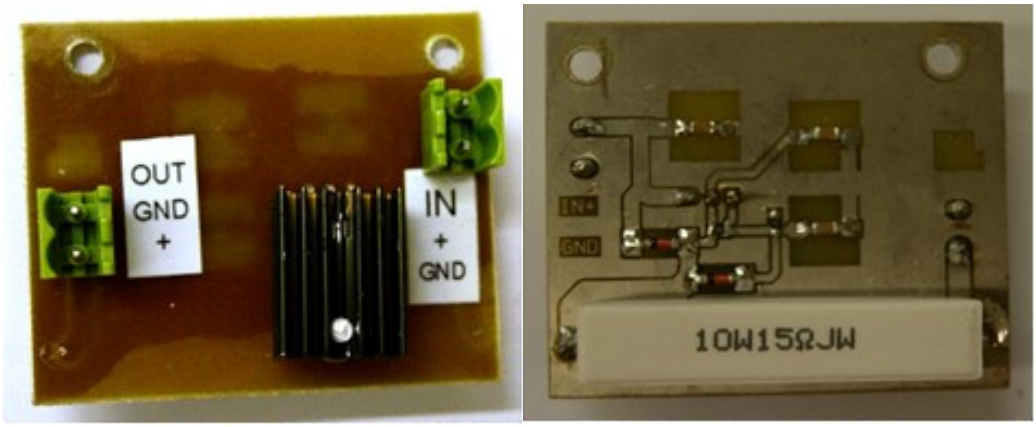


Εικόνα 39: Ηλεκτρονική πλακέτα φόρτισης μπαταρίας - πάνω/κάτω όψη



Εικόνα 40: Ηλεκτρονική πλακέτα φόρτισης μπαταρίας μαζί με τη μπαταρία

Για την σταθεροποίηση της τάσης φόρτισης των υπερπυκνωτών κατασκευάστηκε η ηλεκτρονική πλακέτα μονής όψης που φαίνεται στην εικόνα 42 (πάνω/κάτω όψη). Οι διαστάσεις της είναι: 5.9x4.7x0.7 cm.



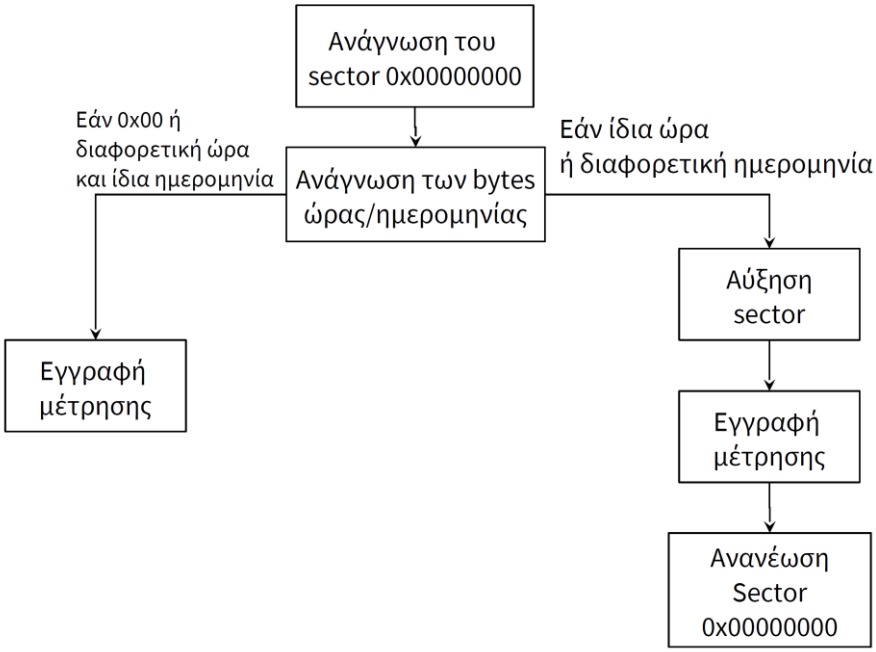
Εικονα 42: Ηλεκτρονική πλακέτα σταθεροποίησης τάσης φόρτισης των υπερπυκνωτών – πάνω/κάτω όψη

Στην εικόνα 43 φαίνεται το συνολικό σύστημα τοποθετημένο σε κουτί μαζί με δύο συστοιχίες των έξι υπερπυκνωτών συνδεδεμένες παράλληλα. Οι έξι πυκνωτές είναι συνδεδεμένοι σε σειρά ώστε η συνολική τάση να είναι 16.2V και η συνολική χωρητικότητα είναι 500F, ενώ η χωρητικότητα του κάθε πυκνωτή είναι 3000F. Το κουτί είναι το PP3004 της εταιρίας Plastim [36] και ακολουθεί τις προδιαγραφές IP65. Τα σχέδια των τυπωμένων κυκλωμάτων παρατίθενται στο Παράρτημα Α και η λίστα των υλικών που χρησιμοποιήθηκαν για την κατασκευή του συστήματος παρατίθεται στο Παράρτημα Η.

Εικόνα 43: Σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας τοποθετημένο σε κουτί

5.2 Ανάπτυξη αλγορίθμου για την αποθήκευση και αποστολή των δεδομένων

Για την αποθήκευση των δεδομένων στην κάρτα μνήμης μSD δεν δημιουργήθηκε σύστημα αρχείων (NTFS, FAT32 κτλ.), συμβατό με την κάρτα μνήμης μSD, επειδή σε αυτή την περίπτωση το λογισμικό θα ήταν αρκετά εκτεταμένο και πολύπλοκο. Αναπτύχθηκε λογισμικό υλικού με το οποίο οι μετρήσεις που λαμβάνονται καταχωρούνται διακριτά ανάλογα με την ημερομηνία που λαμβάνονται. Η κάρτα μνήμης μSD έχει χωρητικότητα 16GB και είναι τύπου SDHC, οπότε ο αποθηκευτικός χώρος της χωρίζεται σε τομείς (sectors) των 512B και έχει περίπου 31 εκατομμύρια sectors. Τα δεδομένα αποθηκεύονται στην κάρτα σε μορφή «ακατέργαστων» δεδομένων (raw data) και ακολουθείται η διαδικασία που απεικονίζεται στην εικόνα 44:



`Εικόνα 44: Διάγραμμα ροής αλγορίθμου για αποθήκευση στην κάρτα microSD

Στον τομέα 0x00000000 (hexadecimal μορφή) της μSD αποθηκεύεται η πληροφορία σχετικά με τον ποιο sector έχουν γραφεί οι τελευταίες μετρήσεις. Στους υπόλοιπους sectors η πληροφορία αποθηκεύεται με την εξής σειρά:

- Bytes 1, 9, 17, 25: Ώρα μέτρησης (μεταβλητή y),
- Bytes 2, 10, 18, 26: Ημέρα μέτρησης
- Bytes 3, 11, 19, 27: Μήνας μέτρησης
- Bytes 4, 12, 20, 28: Έτος μέτρησης
- Bytes 5, 13, 21, 29: Θερμοκρασία (8 bits)
- Bytes 6, 14, 22, 30: Θερμοκρασία (8 bits)
- Bytes 7, 15, 23, 31: Αγωγιμότητα (8 bits)
- Bytes 8, 16, 24, 32: Αγωγιμότητα (8 bits)

Κατά την διαδικασία αποθήκευσης των μετρήσεων, με την ανάγνωση του τομέα 0x00000000, ανακτάται η πληροφορία του τελευταίου καταλυμένου τομέα του οποίου διαβάζεται η τιμή του (μόνο τα τέσσερα πρώτα bytes).

- Εάν τα τέσσερα bytes είναι 0x00, είναι δηλαδή άγραφος, ή η ώρα μέτρησης είναι διαφορετική αλλά η ημερομηνία είναι ίδια, δηλαδή μέτρηση ίδιας ημέρας αλλά διαφορετικής ώρας, καταχωρείται η μέτρηση στη σωστή θέση εγγραφής και η διαδικασία τελειώνει.
- Εάν η ώρα είναι ίδια (δεν μπορεί να υπάρξουν δύο μετρήσεις με την ίδια ώρα στον ίδιο sector) ή αν τα bytes ημερομηνίας είναι διαφορετικά, δηλαδή υπάρχει ήδη μέτρηση διαφορετικής ημερομηνίας στον τομέα, αυξάνεται η τιμή του τομέα κατά 1 και αποθηκεύεται η μέτρηση στον τομέα που προκύπτει από την εν λόγω πρόσθεση. Τέλος, αυξάνεται η τιμή του τομέα 0x00000000 κατά 1 και η διαδικασία τελειώνει.

Από τον κάθε τομέα χρησιμοποιούνται 32 bytes και όλα τα υπόλοιπα έχουν τιμή 0xFF. Για την ανάγνωση των δεδομένων της κάρτας microSD σε ηλεκτρονικό υπολογιστή χρησιμοποιήθηκε το πρόγραμμα ανοιχτού κώδικα HxD [24].

Για την αποστολή των αποθηκευμένων μετρήσεων μέσω του διακομιστή TFTP, απαιτείται η δημιουργία συστήματος αρχείων στη μνήμη RAM του FPGA. Το αρχείο προς αποστολή δημιουργείται κάθε φορά που ενεργοποιείται ο διακομιστής και το όνομα του είναι file1.txt. Το μέγεθος του αρχείου είναι 33 Byte και τα δεδομένα του είναι σε μορφή χαρακτήρων ASCII. Στην περίπτωση, που η ηλεκτρική ενέργεια των υπερπυκνωτών δεν επαρκεί για την λειτουργία του συστήματος, δεν πραγματοποιείται η διαδικασία μέτρησης – αποθήκευσης των μετρήσεων. Επομένως, υπάρχει πιθανότητα να απουσιάζουν μετρήσεις στην μSD. Στη διαδικασία αποστολής υπάρχει προστασία για την εν λόγω απουσία:

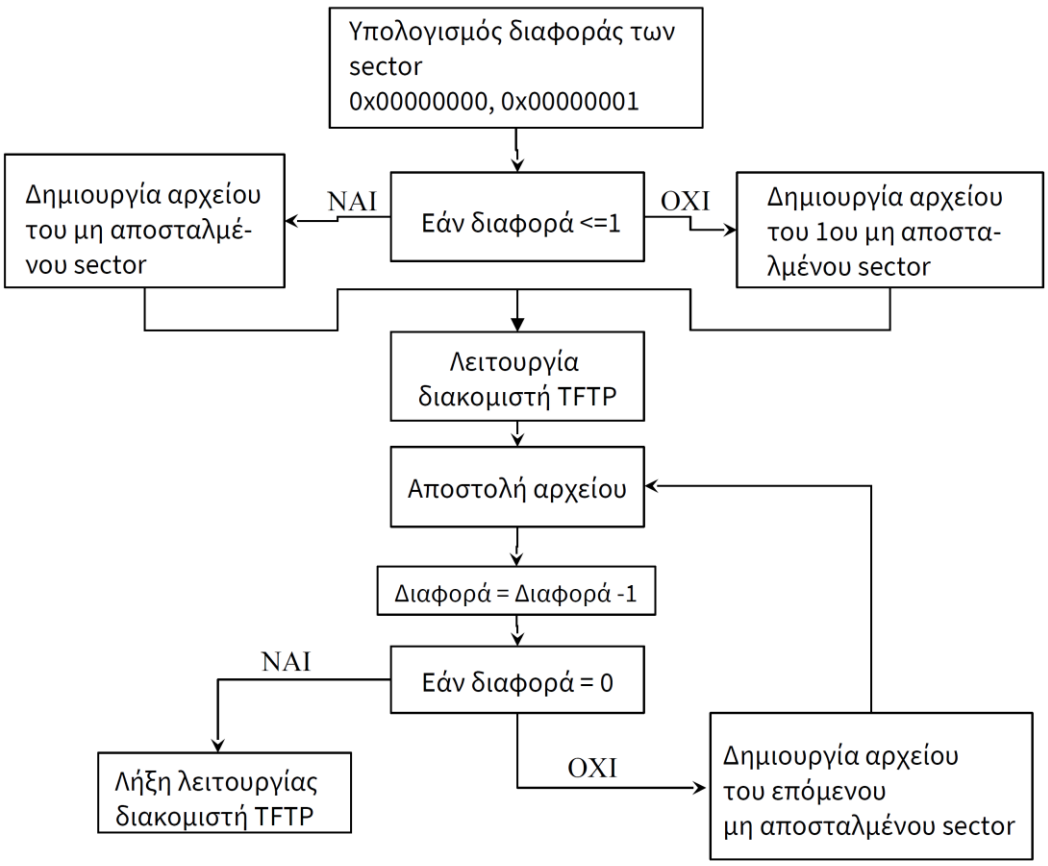
Στον τομέα 0x00000001 της μSD αποθηκεύεται η πληροφορία σχετικά με το ποιος ήταν ο τελευταίος sector του οποίου τα δεδομένα έχουν αποσταλεί. Στον διακομιστή TFTP, αρχικά ελέγχεται η διαφορά των τιμών των τομέων 0x00000000 και 0x00000001.

- Εάν η διαφορά είναι μικρότερη ή ίση του 1, οι μετρήσεις που δεν έχουν αποσταλεί είναι τα δεδομένα του τελευταίου τομέα που έχει γραφεί. Διαβάζεται ο τομέας και δημιουργείται το αρχείο με τα δεδομένα που περιέχει.
- Εάν η διαφορά είναι μεγαλύτερη του 1, πρέπει αν αποσταλούν πάνω από ένα αρχεία. Αυξάνεται η τιμή του τομέα 0x00000001 κατά 1 και δημιουργείται αρχείο στο οποίο καταχωρούνται τα δεδομένα που είναι αποθηκευμένα στον τομέα που προκύπτει από την παραπάνω πρόσθεση.

Έπειτα λειτουργεί ο διακομιστής TFTP και αποστέλλεται το αντίστοιχο αρχείο. Όταν αποσταλεί ένα αρχείο η διαφορά μειώνεται κατά 1. Στη συνέχεια:

- Εάν η διαφορά είναι ίση με 0, δεν υπάρχουν άλλα αρχεία που πρέπει να αποσταλούν και σταματάει η λειτουργία του διακομιστή TFTP.
- Εάν η διαφορά είναι μεγαλύτερη ή ίση του 1, η διαδικασία αποστολής επαναλαμβάνεται.

Στην εικόνα 45 παρουσιάζεται το διάγραμμα ροής της λογικής ελέγχου αποστολής αρχείων:

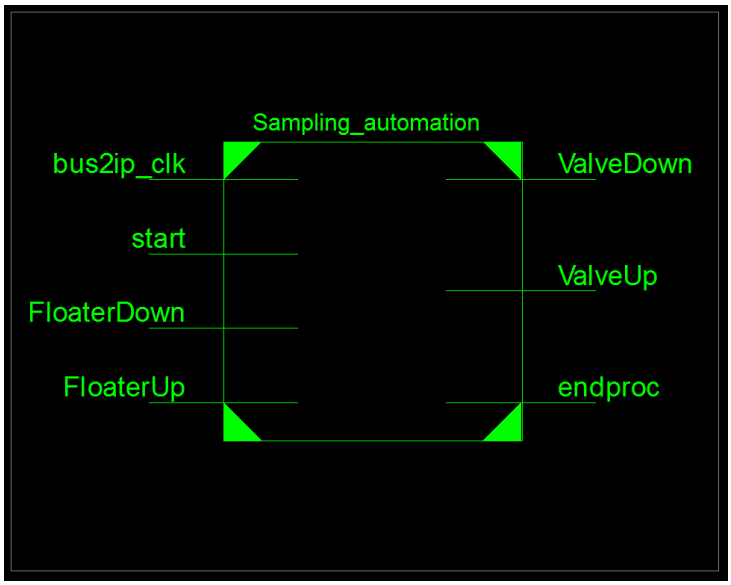


Εικόνα 45: Διάγραμμα ροής λογικής ελέγχου αποστολής αρχείων

5.3 Κώδικας VHDL περιγραφής ηλεκτρονικού κυκλώματος του FPGA για τον αυτοματισμό της δειγματοληψίας

Για την περιγραφή του ηλεκτρονικού κυκλώματος του περιφερειακού αυτοματισμού δειγματοληψίας χρησιμοποιήθηκε η γλώσσα περιγραφής hardware VHDL [26]. Η διαδικασία που ακολουθείται για τη δειγματοληψία αναλύεται στην παράγραφο 3.1 και η ανάπτυξη του περιφερειακού αυτοματισμού δειγματοληψίας στο Παράρτημα ΣΤ. Ο κώδικας περιγραφής του περιφερειακού αποτελείται από τρείς διαδικασίες (processes) που εκτελούνται παράλληλα. Αρχικά, δηλώνονται τα σήματα του κυκλώματος που έχουν διασυνδέσεις με το περιβάλλον του, έπειτα τα εσωτερικά σήματα και οι σταθερές που χρησιμοποιήθηκαν. Η πρώτη διαδικασία χρησιμοποιώντας ρολόι με δύο απαριθμητές (counters), δημιουργεί χρονική καθυστέρηση διάρκειας ενός λεπτού, για τη λειτουργία της ηλεκτρικής βαλβίδας εκκένωσης. Η δεύτερη, όπως και η πρώτη, χρησιμοποιώντας ρολόι με δύο απαριθμητές δημιουργεί χρονόμετρο χρονικής διάρκειας πέντε λεπτών για τον τερματισμό της διαδικασίας δειγματοληψίας. Υλοποιήθηκαν δύο απαριθμητές σε

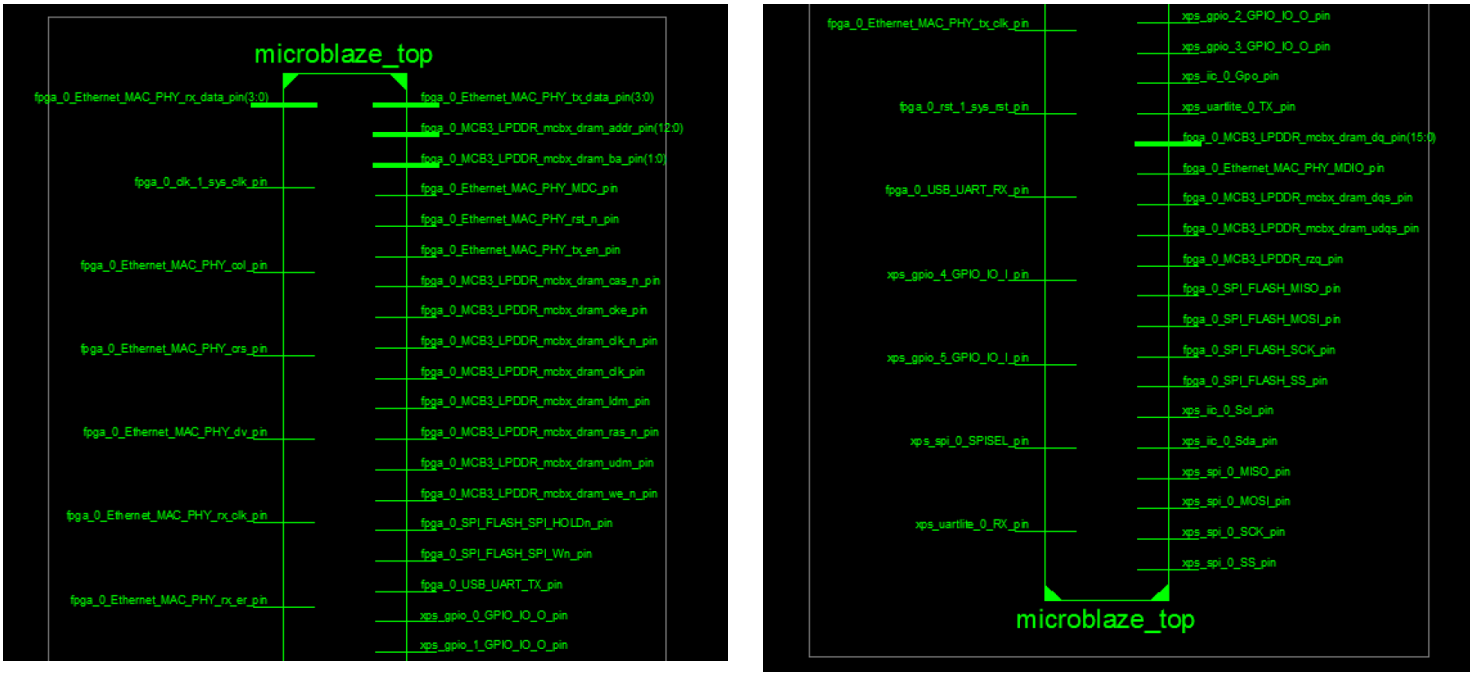
κάθε διαδικασία για τη δημιουργία χρονικής καθυστέρησης, επειδή: η συχνότητα χρονισμού του FPGA είναι 66.7MHz, δηλαδή η περίοδος του ψηφιακού παλμού χρονισμού είναι 15ns. Για την δημιουργία χρονικής καθυστέρησης ενός λεπτού χρειάζονται 4 δισεκατομμύρια παλμοί. Η μεγαλύτερη τιμή ενός ακεραίου στη VHDL είναι 2^{31}= 2.147.483.648, αριθμός μικρότερος από τις περιόδους που χρειάζονται. Στην πρώτη process για τη δημιουργία χρονικής καθυστέρησης διάρκειας 1 λεπτού δημιουργήθηκαν δύο απαριθμητές, ένας που μετρά μέχρι το 400.000.000 δημιουργώντας χρονική καθυστέρηση 6s και ένας που μετρά μέχρι το 10, αυξάνοντας την τιμή του κατά 1 όταν ο πρώτος απαριθμητής πάρει την μέγιστη τιμή του. Αντίστοιχα, στη δεύτερη process υλοποιήθηκαν δύο απαριθμητές, ένας που μετρά μέχρι το 400.000.000 και ένας μέχρι το 50, δημιουργώντας έτσι χρονική καθυστέρηση πέντε λεπτών. Η τρίτη process έχει είσοδο τα σήματα των αισθητήρων στάθμης του νερού και ανάλογα με την τιμή τους ελέγχει τη λειτουργία της ηλεκτρικής βαλβίδας πλήρωσης του δοχείου δειγματοληψίας. Όταν η διαδικασία ολοκληρωθεί ή περάσουν πέντε λεπτά από την έναρξή της το σήμα εξόδου του κυκλώματος endproc γίνεται λογικό ‘1’, από λογικό ‘0’. Ο κώδικας που περιγράφει το περιφερειακό αυτοματισμού παρατίθεται στο παράρτημα Γ. Στην εικόνα 46 φαίνεται το ηλεκτρονικό κύκλωμα για τον αυτοματισμό δειγματοληψίας που υλοποιείται εντός του FPGA.



Εικόνα 46: Ηλεκτρονικό κύκλωμα αυτοματισμού δειγματοληψίας

5.4 Ανάπτυξη λογισμικού που λειτουργεί στον MicroBlaze

Σε αυτή την παράγραφο παρατίθενται οι συναρτήσεις που αναπτύχθηκαν σε γλώσσα C [27] και λειτουργούν στον μBlaze. Το ηλεκτρονικό κύκλωμα του μBlaze που χρησιμοποιείται από το σύστημα φαίνεται στην εικόνα 47.



Εικόνα 47: Ηλεκτρονικό κύκλωμα MicroBlaze

Οι συναρτήσεις των οποίων η γραμματοσειρά είναι σε μορφή italic δημιουργήθηκαν αυτόματα από την εφαρμογή Software Development Kit.

- **Συναρτήσεις που περιλαμβάνονται στο αρχείο spi_f.c:**
  - **u32 spi_send_byte (u8 Byte)**

  Χρησιμοποιείται για να σταλούν δεδομένα στην κάρτα μνήμης μSD με το πρωτόκολλο SPI. Με την συνάρτηση XIo_Out32((Register Address) + (offset), data) της βιβλιοθήκης XIo.h, προγραμματίζονται οι καταχωρητές του περιφερειακού SPI, ενώ με την συνάρτηση XIo_In32 ((Register address) + (offset)) της βιβλιοθήκης XIo.h, δέχεται δεδομένα από τους παραπάνω καταχωρητές. Το όρισμα u8 Byte είναι τα δεδομένα προς αποστολή.
  - **u32 spi_read_byte (void)**

  Χρησιμοποιείται για την λήψη δεδομένων που είναι αποθηκευμένα στην κάρτα μνήμης μSD.

- **Συναρτήσεις που περιλαμβάνονται στο αρχείο init.c:**
  - **int sd_init(void)**

  Περιγράφει τη διαδικασία αρχικοποίησης, σύμφωνα με τις οδηγίες της SD Association [28], της κάρτας μνήμης μSD ώστε να προγραμματιστεί σαν εξυπηρετούμενο σύστημα στο δίαυλο του πρωτοκόλλου SPI. Ο εξυπηρετητής του διαύλου SPI επικοινωνεί με την κάρτα μνήμης μSD μέσω εντολών (CMDx) μεγέθους 6 bytes. Το πρώτο byte είναι ο αριθμός της εντολής, τα επόμενα τέσσερα bytes είναι τα ορίσματα της εντολής και το έκτο byte περιέχει το byte ελέγχου CRC (cyclic redundancy check), το οποίο ελέγχει την ορθή αποστολή

των 5 bytes. Αρχικά, αποστέλλεται η εντολή CMD0 η οποία βάζει την κάρτα σε κατάσταση αναμονής ή την επαναφέρει σε αυτή. Έπειτα, ακολουθεί η εντολή CMD8 οι οποία ελέγχει την διεπαφή στην οποία είναι συνδεδεμένη η κάρτα και αν μπορεί να λειτουργήσει με την τάση τροφοδοσίας που της παρέχεται. Τέλος, αποστέλλεται το ζεύγος εντολών CMD55 και ACMD41. Η CMD55 ειδοποιεί την κάρτα ότι η επόμενη εντολή έχει το πρόθεμα Α, έχει δηλαδή συγκεκριμένη εφαρμογή (application specific command). Τέλος, η ACMD41 που ακολουθεί ενεργοποιεί τη διαδικασία αρχικοποίησης της κάρτας μνήμης μSD και τη θέτει σε κατάσταση πλήρους λειτουργίας.

- **Συναρτήσεις που περιλαμβάνονται στο αρχείο tftpserver.c:**
  - **void print_tftp_app_header( )**

  Τυπώνει πληροφορίες σχετικά με την υλοποίηση του διακομιστή, δηλαδή διεύθυνση ip και την θύρα διαδικτύου στην οποία λειτουργεί ο διακομιστής.
  - **int start_tftp_application( )**

  Ενεργοποιεί τον διακομιστή TFTP.
  - **int transfer_tftp_data( )**

  Περιγράφει τις λειτουργίες αποστολής και λήψης αρχείων που υπάρχουν στον διακομιστή TFTP.

- **Συναρτήσεις που περιλαμβάνονται στη βιβλιοθήκη lwip.h:**
  - **void lwip_raw_init( )**

  Ξεκινάει την εφαρμογή ανοιχτού κώδικα lwIP [29], η οποία περιλαμβάνει την στοίβα UDP που χρησιμοποιείται από τον διακομιστή TFTP.

- **Συναρτήσεις που περιλαμβάνονται στο αρχείο encoder.c:**
  - **u8 SecondsEncoder(u8 LSB, u8 MSB)**
  - **u8 MinutesEncoder(u8 LSB, u8 MSB)**
  - **u8 HoursEncoder(u8 LSB, u8 MSB)**
  - **u8 DayEncoder(u8 LSB, u8 MSB)**
  - **u8 MonthEncoder(u8 LSB, u8 MSB)**
  - **u8 YearEncoder(u8 LSB, u8 MSB)**

  Οι παραπάνω συναρτήσεις χρησιμοποιούνται για να μετατρέψουν τα δεδομένα που λαμβάνονται από το GPS και είναι κωδικοποιημένα σε χαρακτήρες ASCII σε δεκαεξαδική (hexadecimal) μορφή, ώστε να βαθμονομηθεί το RTC, δηλαδή να προγραμματιστούν οι καταχωρητές του χρονομέτρου και της ημερομηνίας του RTC. Το GPS αποστέλλει δύο bytes για κάθε στοιχείο της ώρας / ημερομηνίας, δηλαδή τα δεδομένα έχουν μορφή:
  Ώρα → SSMMHH,
  Ημερομηνία → DDMMYY, όπου SS,MM,HH,DD,MM,HH δύο bytes.
  Το όρισμα u8 MSB είναι ο πρώτος χαρακτήρας ASCII κάθε στοιχείου και το όρισμα u8 LSB ο δεύτερος.

Επίσης, χρησιμοποιήθηκαν οι παρακάτω συναρτήσεις για τη λειτουργία των πρωτοκόλλων επικοινωνίας του συστήματος. Δημιουργούνται αυτόματα και βρίσκονται στις βιβλιοθήκες που δηλώθηκαν στην αρχή του λογισμικού. Συγκεκριμένα:

- ***Βιβλιοθήκη xiic_l.h (πρωτόκολλο I^2C), συναρτήσεις:***
  - ***XIic_Send (u32 BaseAddress, u8 Address, u8 *BufferPtr, unsigned ByteCount , u8 Option)***
  - ***Xiic_Recv (u32 BaseAddress, u8 Address, u8 *BufferPtr, unsigned ByteCount , u8 Option)***

  Οι δύο συναρτήσεις χρησιμοποιούνται για αποστολή και λήψη δεδομένων στον δίαυλο I^2C που υλοποιήθηκε. Το όρισμα u32 BaseAddress είναι η διεύθυνση του περιφερειακού I^2C στον μικροεπεξεργαστή MicroBlaze, που είναι ο εξυπηρετητής του διαύλου, και το u8 Address η διεύθυνση του περιφερειακού (RTC ή ADC) που είναι το ενεργό εξυπηρετούμενο σύστημα στο δίαυλο. Το όρισμα u8 *BufferPtr είναι ο δείκτης της μεταβλητή που περιέχει τα δεδομένα προς αποστολή ή καταχωρούνται τα δεδομένα που λαμβάνονται. Το όρισμα u8 ByteCount είναι ο αριθμός των bytes που θα σταλούν/ληφθούν και το όρισμα u8 Option ελέγχει την κατάσταση του διαύλου μόλις ολοκληρωθεί η διαδικασία λήψης/αποστολής.

- ***Βιβλιοθήκη xuart_lite_l.h (πρωτόκολλο UART), συναρτήσεις:***
  - ***XUartLite_SendByte (u32 BaseAddress, u8 Data)***
  - ***XUartLite_RecvByte (u32 BaseAddress)***

  Συναρτήσεις για αποστολή και λήψη δεδομένων μέσω του πρωτοκόλλου επικοινωνίας UART. Χρησιμοποιούνται για την επικοινωνία του FPGA με το σύστημα GPS του συστήματος. Το όρισμα u32 BaseAddress είναι η διεύθυνση του περιφερειακού UART στον μικροεπεξεργαστή Microblaze και το όρισμα u8 Data τα δεδομένα προς αποστολή.

Βιβλιοθήκη xgpio_l.h (general purpose input/output), συναρτήσεις:

- ***XGpio_WriteReg (u32 BaseAddress, u8 RegOffset, u8 Data)***
- ***XGpio_ReadReg (u32 BaseAddress, u8 RegOffset)***

Συναρτήσεις για αποστολή και λήψη δεδομένων μέσω των ακροδεκτών γενικού σκοπού. Στη συγκεκριμένη εφαρμογή χρησιμοποιούνται για τον έλεγχο των ηλεκτρονόμων στερεάς κατάστασης, την ανάγνωση των αισθητήρων στάθμης και την έναρξη / λήξη της δειγματοληψίας. Το όρισμα u32 BaseAddress είναι η διεύθυνση του περιφερειακού gpio στον μικροεπεξεργαστή Microblaze, το όρισμα u8 RegOffset είναι η διαφορά της διεύθυνσης ενός καταχωρητή του περιφερειακού από τη διεύθυνση του περιφερειακού και το όρισμα u8 Data τα δεδομένα προς αποστολή.

Τέλος, ακολουθούν οι συναρτήσεις που χρησιμοποιήθηκαν για την υλοποίηση συστήματος αρχείων στη μνήμη RAM (random access memory) του FPGA:

- ***Βιβλιοθήκη xilmfs.h (Xilinx memory file system), συναρτήσεις:***
  - ***mfs_init_fs (int numbytes, char* address, int init_type)***
  - ***mfs_create_dir (char* newdir)***
  - ***mfs_file_open (const char* filename, int mode)***
  - ***mfs_file_write (int fd, const char* buf, int buflen)***

  Η συνάρτηση mfs_init_fs δημιουργεί το σύστημα αρχείων με τα ορίσματά της να δηλώνουν το μέγεθος του συστήματος, την διεύθυνση του στον μικροεπεξεργαστή MicroBlaze και τον τύπο του συστήματος. Η συνάρτηση mfs_create_dir δημιουργεί τον κατάλογο στον οποίο αποθηκεύονται τα αρχεία που δημιουργούνται. Το όρισμα της είναι το όνομα του νέου καταλόγου. Η συνάρτηση mfs_file_open δημιουργεί ένα νέο ή ανοίγει ένα ήδη υπάρχον αρχείο. Με το όρισμα const char* δίνεται το όνομα του αρχείου και το όρισμα int mode δηλώνει τη λειτουργία της συνάρτησης, δηλαδή δημιουργία ή άνοιγμα αρχείου. Η συνάρτηση mfs_file_write εισάγει δεδομένα σε ένα αρχείο. Το όρισμα int fd είναι ο αύξων αριθμός του αρχείου, στη μνήμη του FPGA, στο οποίο θα εισαχθούν δεδομένα, το όρισμα const char* buf είναι ο δείκτης στη μεταβλητή που περιέχει τα δεδομένα προς εισαγωγή και το όρισμα int buflen το μέγεθος της μεταβλητής.

Εκτός από τις συναρτήσεις που αναφέρθηκαν αναπτύχθηκε και η κύρια συνάρτηση:

- **int main (void)**

Στην εικόνα 48 φαίνεται το διάγραμμα ροής της κύριας συνάρτησης main: Δηλώνονται οι τοπικές μεταβλητές και έπειτα ακολουθεί ο κυρίως κώδικας. Αρχικά, προγραμματίζονται οι καταχωρητές του RTC που είναι υπεύθυνοι για τον προγραμματισμό του συναγερμού alarm 2, σύμφωνα με τον πίνακα 3 που βρίσκεται στην παράγραφο 3.4, και η τιμή τους είναι ίδια για κάθε ώρα ενεργοποίησης του συστήματος, δηλαδή προγραμματίζονται οι καταχωρητές ημέρας και λεπτών με τις τιμές 0x80,0x00 αντίστοιχα. Έπειτα, ελέγχεται ο καταχωρητής ώρας του RTC ώστε να δοθεί τιμή στην μεταβλητή y, που οι τιμές της είναι οι ώρες δειγματοληψίας – συγχρονισμού των συστημάτων του ασύρματου συστήματος. Οι τιμές και οι αντίστοιχες ώρες φαίνονται στον πίνακα 5:

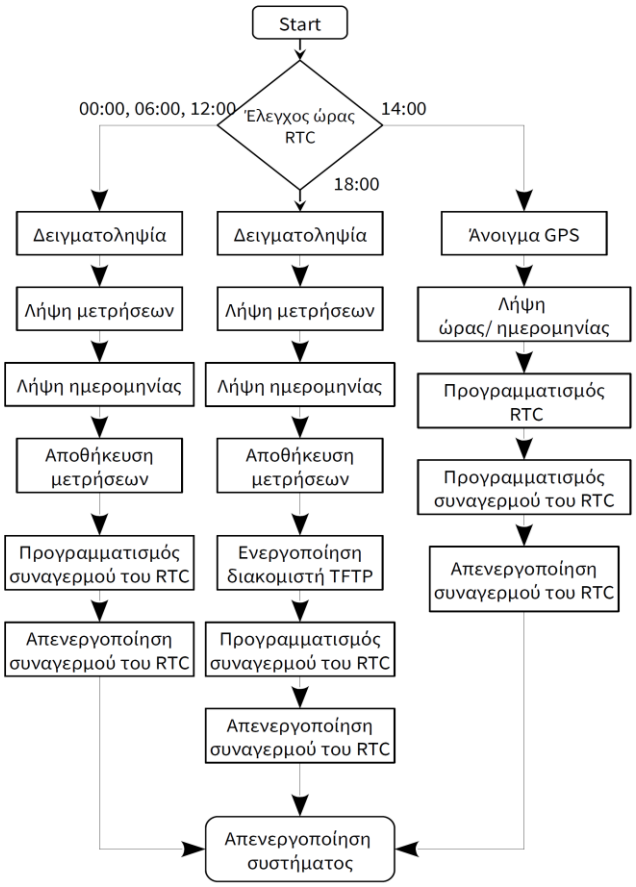
Μεταβλητή y	**Ώρα ενεργοποίησης συστήματος**
0x01	00:00
0x02	06:00
0x03	12:00
0x04	18:00
0x05	14:00

Πίνακας 5: Τιμές της μεταβλητής y

Ελέγχεται η τιμή της μεταβλητής y και ακολουθείται η αντίστοιχη διαδικασία λειτουργίας του συστήματος. Εάν, η τιμή αντιστοιχεί σε ώρα που πρέπει να γίνει

δειγματοληψία - μέτρηση, ξεκινά η διαδικασία δειγματοληψίας. Όταν τελειώσει, μέσω του πρωτοκόλλου I^2C λαμβάνονται οι μετρήσεις του ADC στο FPGA. Επειδή το μέγεθος τους είναι 16-bit (2 bytes), διαχωρίζονται σε δύο μεταβλητές των 8-bit, ώστε να μπορούν να αποθηκευθούν στη κάρτα μνήμης μSD. Ακολουθεί η λήψη των τιμών των καταχωρητών ημερομηνίας του RTC και έτσι έχει συλλεχθεί όλη η απαραίτητη πληροφορία για μία μέτρηση. Στη συνέχεια αποθηκεύεται η μέτρηση στην κάρτα μνήμης μSD ακολουθώντας την διαδικασία που περιγράφηκε στην παράγραφο 5.2. Εάν η μεταβλητή y έχει τιμή 0x04 δημιουργείται το σύστημα των αρχείων και ενεργοποιείται ο διακομιστής TFTP. Τέλος, προγραμματίζονται οι καταχωρητές του alarm 2, ώστε να ενεργοποιηθεί το σύστημα την αμέσως επόμενη ώρα λειτουργίας, και το σύστημα απενεργοποιείται, μέσω του προγραμματισμού του καταχωρητή κατάστασης του RTC.

Εάν, η τιμή της μεταβλητής y αντιστοιχεί στην ώρα 2μμ, το GPS του συστήματος ενεργοποιείται και όταν ληφθεί έγκυρη πληροφορία από τους δορυφόρους, δηλαδή το byte 18 της μεταβλητής GpsData [i] είναι ίσο με τον χαρακτήρα ‘A’, προγραμματίζονται οι καταχωρητές χρονομέτρησης/ημερομηνίας του RTC, αφού πρώτα τα δεδομένα που έχουν ληφθεί από το GPS μετατραπούν σε δεκαεξαδική μορφή. Προγραμματίζονται οι καταχωρητές του alarm 2 και το σύστημα απενεργοποιείται, μέσω του προγραμματισμού του καταχωρητή κατάστασης του RTC.



Εικόνα 48: Διάγραμμα ροής συνάρτησης main

5.5 Προγραμματισμός του συστήματος

Η ηλεκτρονική πλακέτα Avnet LX9 Microboard διαθέτει θύρα USB – JTAG μέσω της οποίας γίνεται ο προγραμματισμός του FPGA και της μνήμης Flash που υπάρχει στην πλακέτα. Επειδή όταν το FPGA χάσει την τάση τροφοδοσίας του το πρόγραμμα που εκτελείται σε αυτό χάνεται, το λογισμικό αποθηκεύεται στην μνήμη Flash, η οποία έχει προγραμματιστεί σαν PROM. Η μνήμη προγραμματίζεται με το πρόγραμμα iMPACT της πλατφόρμας ISE Design suite και η διαδικασία προγραμματισμού περιγράφεται στο Παράρτημα Ε. Το πρόγραμμα καταλαμβάνει χώρο 2014KB από το μέγιστο των 16MB, δηλαδή χρησιμοποιείται το 12.58% της συνολικής χωρητικότητας της μνήμης Flash.

6. Αποτίμηση του συστήματος

6.1 Περιγραφή λειτουργίας συστήματος

Το σύστημα έγκαιρης προειδοποίησης που αναπτύχθηκε ενεργοποιείται πέντε φορές την ημέρα και ανάλογα με την ώρα ενεργοποίησης, εκτελείται η αντίστοιχη διαδικασία λειτουργίας. Υπεύθυνο για την ενεργοποίηση του συστήματος είναι το ρολόι πραγματικού χρόνου, που έχει προγραμματιστεί να παράγει το σήμα που ενεργοποιεί τους ρυθμιστές τάσης ώστε να τροφοδοτείται το σύστημα. Επίσης, σημαντικό ρόλο παίζει ο επιτηρητής τάσης MAX6460 καθώς, εάν η τάση που παρέχεται από τους υπερπυκνωτές [5] δεν επαρκεί για την λειτουργία του συστήματος, δηλαδή είναι μικρότερη από 14.5V, δεν επιτρέπει τη λειτουργία του. Οι ώρες που το σύστημα ενεργοποιείται είναι 12πμ, 6πμ, 12μμ, 2μμ, 6μμ, και οι διαδικασίες που ακολουθούνται κάθε φορά παρουσιάζονται παρακάτω:

- Ώρες 6πμ, 12μμ, 12πμ: Με την παραγωγή του σήματος από το ρολόι, οι ρυθμιστές τάσης τροφοδοτούν το σύστημα και το FPGA ξεκινάει τη λειτουργία του. Αρχικά, ενεργοποιείται το περιφερειακό σύστημα αυτοματισμού δειγματοληψίας ώστε να ανανεωθεί το νερό που υπάρχει στο δοχείο δειγματοληψίας. Μετά το πέρας της διαδικασίας, αρχίζει η διαδικασία μέτρησης των φυσικοχημικών παραμέτρων του νερού. Το αναλογικό σήμα ρεύματος 4-20mA, που παράγεται από τους αισθητήρες, μετατρέπεται σε αναλογικό σήμα τάσης 1-5V και έπειτα ο ADC το ψηφιοποιεί. Το FPGA μέσω του διαύλου I^2C επικοινωνεί με το ADC, από το οποίο λαμβάνει τις μετρήσεις, και με το RTC, από το οποίο λαμβάνει πληροφορίες για την ημερομηνία της μέτρησης. Τέλος, οι μετρήσεις και η ημερομηνία αποθηκεύονται στην κάρτα μνήμης μSD, προγραμματίζονται οι καταχωρητές του δεύτερου συναγερμού (alarm 2) του RTC και το σύστημα απενεργοποιείται.
- Ώρα 6μμ: Ακολουθείται η διαδικασία που περιγράφηκε για τις ώρες 6πμ, 12μμ, 12μμ με την πρόσθετη λειτουργία της ενεργοποίησης του διακομιστή TFTP και της κεραία μετάδοσης πληροφορίας. Συγκεκριμένα, μετά το τέλος της μέτρησης και την αποθήκευση των δεδομένων στην κάρτα μνήμης μSD, τροφοδοτείται η κεραία ώστε να σταλεί το αρχείο που δημιουργήθηκε και περιλαμβάνει τα δεδομένα των μετρήσεων της συγκεκριμένης μέρας. Έπειτα, προγραμματίζεται ο συναγερμός 2 και το σύστημα απενεργοποιείται.
- Ώρα 2μμ: Ενεργοποιείται μόνο το GPS του συστήματος, ώστε να ανανεωθούν οι καταχωρητές ώρας και ημερομηνίας του ρολογιού από τις πληροφορίες ημερομηνίας και ώρας των δορυφόρων και το σύστημα να παραμείνει συγχρονισμένο.

Τις ώρες 12πμ, 6πμ και 12μμ το σύστημα παραμένει ανοιχτό για χρονικό διάστημα ανάλογο με το χρόνο που θα χρειαστεί η αντλία να γεμίσει το δοχείο δειγματοληψίας με νερό, καθώς ο κυρίως κώδικας δεν εκτελείται εάν δεν τελειώσει η διαδικασία δειγματοληψίας και ο χρόνος εκτέλεσής του είναι αρκετά μικρός (περίπου 10 δευτερόλεπτα). Ο διαθέσιμος χρόνος που έχει δοθεί στο σύστημα για την

δειγματοληψία είναι πέντε λεπτά. Σε περίπτωση που χρειαστούν πάνω από πέντε λεπτά για να ολοκληρωθεί η διαδικασία, το περιφερειακό αυτοματισμού δειγματοληψίας σταματάει τη λειτουργία του και εκτελείται ο κυρίως κώδικας. Το σύστημα αναμετάδοσης της πληροφορίας παραμένει απενεργοποιημένο. Την ώρα 6μμ, το σύστημα έχει την ίδια λειτουργία με τις παραπάνω ώρες, με τη διαφορά ότι μόλις ολοκληρωθεί η διαδικασία αποθήκευσης των μετρήσεων, ενεργοποιείται η κεραία αναμετάδοσης πληροφορίας και ενεργοποιείται ο διακομιστής TFTP. Ο διαθέσιμος χρόνος συνολικά για την διαδικασία στις 18:00 που έχει δοθεί στο σύστημα είναι 10 λεπτά. Με το πέρας των 10 λεπτών το σύστημα απενεργοποιείται.

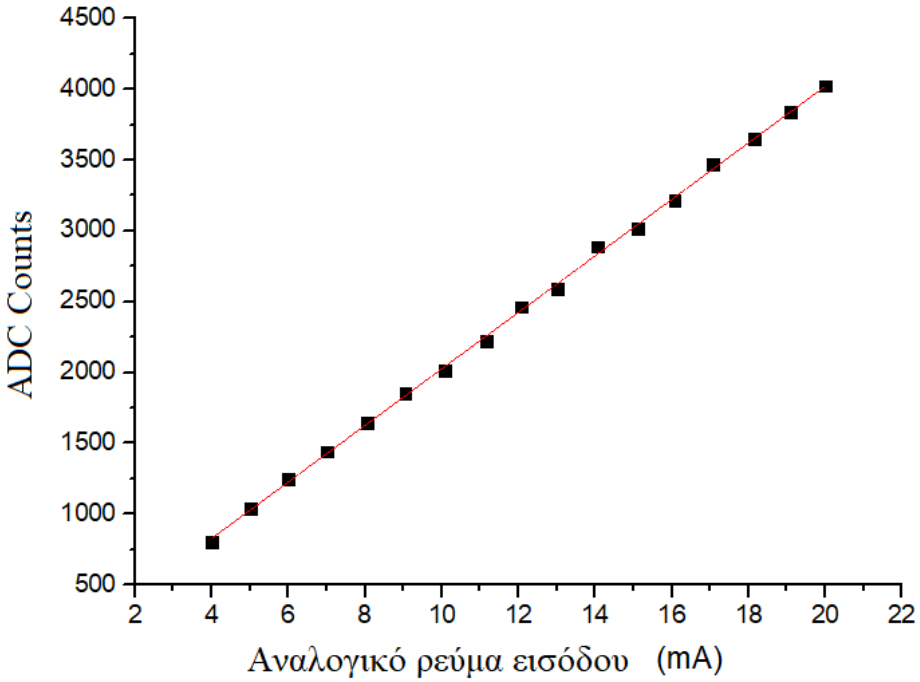
6.2 Βαθμονόμηση αισθητήρων

Όπως αναφέρθηκε στο κεφάλαιο 3 στο σύστημα που αναπτύχθηκε χρησιμοποιείται αισθητήρας της εταιρίας Global Water. Ο αισθητήρας πριν χρησιμοποιηθεί στο σύστημα βαθμονομήθηκε. Βαθμονόμηση είναι η διαδικασία κατά την οποία εξετάζεται η ορθότητα των μετρήσεων. Στην περίπτωση που υπάρχει απόκλιση οι μετρήσεις πολλαπλασιάζονται με έναν συντελεστή απόκλισης ώστε να διορθωθούν. Μετράτε δηλαδή μία φυσικοχημική παράμετρος με όργανο μέτρησης μεγάλης ακρίβειας και έπειτα με τον προς βαθμονόμηση αισθητήρα. Με την σύγκριση των δύο τιμών προσδιορίζεται ο συντελεστής διόρθωσης των μετρήσεων του αισθητήρα.

Για τη μετατροπή του σήματος του αισθητήρα από αναλογικό σήμα ρεύματος 4-20mA σε αναλογικό σήμα τάσης 1-5V και έπειτα σε ψηφιακό σήμα, κατασκευάστηκε το κύκλωμα μετατροπής και ψηφιοποίησης σήματος που περιγράφεται στην παράγραφο 4.2.2. Καθίσταται έτσι, απαραίτητος ο έλεγχος της ορθής λειτουργίας του κυκλώματος πριν τη διαδικασία βαθμονόμησης του αισθητήρα. Το στοιχείο του κυκλώματος που ελέγχθηκε, είναι ο μετατροπέας αναλογικού σήματος σε ψηφιακό (ADC) του συστήματος. Για τη βαθμονόμηση κατασκευάστηκε μία πηγή ρεύματος, με τη χρήση τροφοδοτικού τάσης και δύο μεταβλητών αντιστάσεων (ροοστατών) και για τιμές ρεύματος εύρους 4-20mA μετρήθηκαν τα ADC counts και η αντίστοιχη τάση εξόδου. Ο ADC έχει ανάλυση 12bit, οπότε το εύρος των ADC counts είναι από 0 έως 2^{12} = 4096 για τάση μεταξύ των 0V και της τάσης αναφοράς, 5.15V στο κύκλωμα που κατασκευάστηκε. Επομένως, η ευαισθησία της εισόδου του ADC είναι 5.15V/4096=1.2mV. Επίσης, επειδή η είσοδος του ADC παίρνει τιμές από 1-5V και όχι από 0-5V το εύρος των τιμών των ADC counts μειώνεται κατά: 4096/(5.15-0)=795. Άρα το νέο εύρος των ADC counts είναι: 4096-795=3301 τιμές. Στον πίνακα 6 παρουσιάζονται οι μετρήσεις που καταγράφηκαν και στην εικόνα 49 το διάγραμμα του ρεύματος συναρτήσει των ADC counts (μέση τιμή δέκα μετρήσεων για κάθε τιμή ρεύματος):

Αναλογικό ρεύμα εισόδου (mA)	Αναλογική τάση εξόδου (V)	ADC counts (Decimal)
4,01	1.00	809
5,03	1.30	1008
6	1.56	1212
7,05	1.81	1416
8,04	2.07	1623
9,02	2.33	1833
10,05	2.53	2030
11,04	2.79	2233
12	3.09	2422
13,07	3.25	2643
14,05	3.63	2845
15,06	3.79	3045
16,08	4.04	3243
17,01	4.36	3436
18,06	4.58	3642
19,04	4.86	3869
20,03	5.05	4058

Πίνακας 6: Μετρήσεις ελέγχου του κυκλώματος μετατροπής



Εικόνα 49: Διάγραμμα ρεύματος συναρτήσει των ADC counts

Όπως φαίνεται στην εικόνα 49, η σχέση του αναλογικού ρεύματος εισόδου με τα ADC counts είναι γραμμική, κάτι που ήταν αναμενόμενο, οπότε το κύκλωμα λειτουργεί ικανοποιητικά.

Στην εικόνα 49 η γραφική παράσταση του ρεύματος εισόδου συναρτήσει των ADC counts είναι ευθεία, δηλαδή περιγράφεται από μία εξίσωση y=ax+b. Για να υπολογιστούν οι παράμετροι της εξίσωσης χρησιμοποιήθηκε η μέθοδος προσαρμογής των ελαχίστων τετραγώνων. Οι παράμετροι a και b υπολογίζονται από τις σχέσεις:

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{D},$$

$$b = \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{D},$$

όπου $D = n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2,$

όπου x_i είναι οι τιμές του ρεύματος εισόδου, y_i οι τιμές των ADC counts και n το πλήθος των μετρήσεων. Οι τιμές που προκύπτουν είναι:
α=202.57 mA^{-1}
β=29.99
Τα σφάλματα τους υπολογίζονται από τις παρακάτω σχέσεις:

$$\sigma_y = \sqrt{\frac{\sum_{i=1}^{n}(y_i - b - ax_i)^2}{n-2}},$$

$$\sigma_a = \sigma_y\sqrt{\frac{n}{D}},$$

$$\sigma_b = \sigma_y\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{D}},$$

Οι τιμές που προκύπτουν είναι:
σ_a= 0.31 mA^{-1}
σ_b= 4.07
Άρα, a= 202.57 ± 0.31 mA^{-1}
b= -5.42 ± 4.07

Επομένως, η σχέση μεταξύ των ADC counts και του ρεύματος εισόδου είναι ευθεία και περιγράφεται από την εξίσωση: Co=202.57 * I – 5.42 $\Rightarrow$ $I = \frac{C_o+5.42}{202.57}$.

Η εταιρία κατασκευής του αισθητήρα Global Water έχει ήδη πραγματοποιήσει τη βαθμονόμηση του και αναγράφει πάνω σε αυτόν τα αποτελέσματα, που παρουσιάζονται στον πίνακα 7:

Αισθητήρας	**Βαθμονομημένη τιμή**
WQ-Cond Temperature	-5°C→4.0mA +70°C→20.0mA
WQ-Cond Conductivity	200 μS/cm→4.0mA 2000μS/cm→20.0mA

Πίνακας 7: Αποτελέσματα βαθμονόμησης από την Global Water

Οπότε θεωρώντας την σχέση y=κx+λ, όπου x η τιμή του ρεύματος που παράγεται από τον αισθητήρα και y η τιμής της μετρούμενης φυσικοχημικής παραμέτρου, χρησιμοποιώντας τις τιμές του πίνακα 4 μπορεί να υπολογιστεί η σχέση της παραμέτρου με το ρεύμα. Για την αγωγιμότητα, με χρήση των τιμών του πίνακα 4 υπολογίζεται ότι:

κ=112,5 μS/cm*mA

λ=-250 μS/cm.

Άρα C=112.5*I – 250 $\Rightarrow$

$$C = 112{,}5\frac{Co + 5{,}42}{202{,}57} - 250 \Rightarrow$$

$$C = \frac{112{,}5}{202{,}57}Co - 250 - \frac{112{,}5 * (-5{,}42)}{202{,}57}$$

$$C = 0.55Co - 247.$$

Το σφάλμα της αγωγιμότητας υπολογίζεται ως εξής:

$$\sigma_C = \sqrt{(\frac{\partial C}{\partial a})^2\sigma_a^2 + (\frac{\partial C}{\partial b})^2\sigma_b^2 + (\frac{\partial C}{\partial Co})^2\sigma_{Co}^2}$$

$$\sigma_C = \sqrt{(\frac{-\kappa * Co}{a^2} + \frac{\kappa * b}{a^2})^2\sigma_\alpha^2 + \left(\frac{-\kappa}{\alpha}\right)^2\sigma_b^2 + \left(\frac{\kappa}{a}\right)^2\sigma_{Co}^2} \ ,$$

όπου σ_{Co}=2 επειδή, η ακρίβεια του ADC είναι ±1 LSB.
Μετά από αντικατάσταση των τιμών προκύπτει ότι το σφάλμα της αγωγιμότητας είναι:

$\sigma_C = 3.06$ μS/cm.

Επομένως, η εξίσωση που περιγράφει την αγωγιμότητα συναρτήσει των ADC counts είναι:

C=0.55 Co -247 σε μS/cm με σφάλμα σ_C= ±3 μS/cm.

Αντίστοιχα προκύπτει ότι η εξίσωση που περιγράφει την θερμοκρασία συναρτήσει των ADC counts είναι:

T=0.023 Co -23.04 σε ^oC με σφάλμα σ_T=±0.8 ^oC.

Ακολούθως πραγματοποιήθηκε έλεγχος της βαθμονόμησης του αισθητήρα. Για τον έλεγχο της μέτρησης αγωγιμότητας κατασκευάστηκε διάλυμα γνωστής τιμής αγωγιμότητας 500μS/cm [38] (εικόνα 50) και η θερμοκρασία του, μετρήθηκε με θερμόμετρο υψηλής ακρίβειας, ήταν 17.03 ^oC. Στη συνέχεια, το ίδιο δείγμα μετρήθηκε με σύστημα και οι τιμές θερμοκρασίας και αγωγιμότητας που λήφθηκαν ήταν:

- Θερμοκρασία →1744 Counts που αντιστοιχούν σε 17.1 ^oC ±0.8 ^oC,
- Αγωγιμότητα → 1357 Counts που αντιστοιχούν σε 499 μS/cm ±3 μS/cm.

Οι τιμές που μετρήθηκαν με το σύστημα είναι εντός της τυπικής απόκλισης. Το σύστημα είναι βαθμονομημένο ικανοποιητικά.

Εικόνα 50: Υδατικό διάλυμα γνωστής αγωγιμότητας 500 μS/cm

6.3 Διακομιστής TFTP – Διεπαφή χρήστη

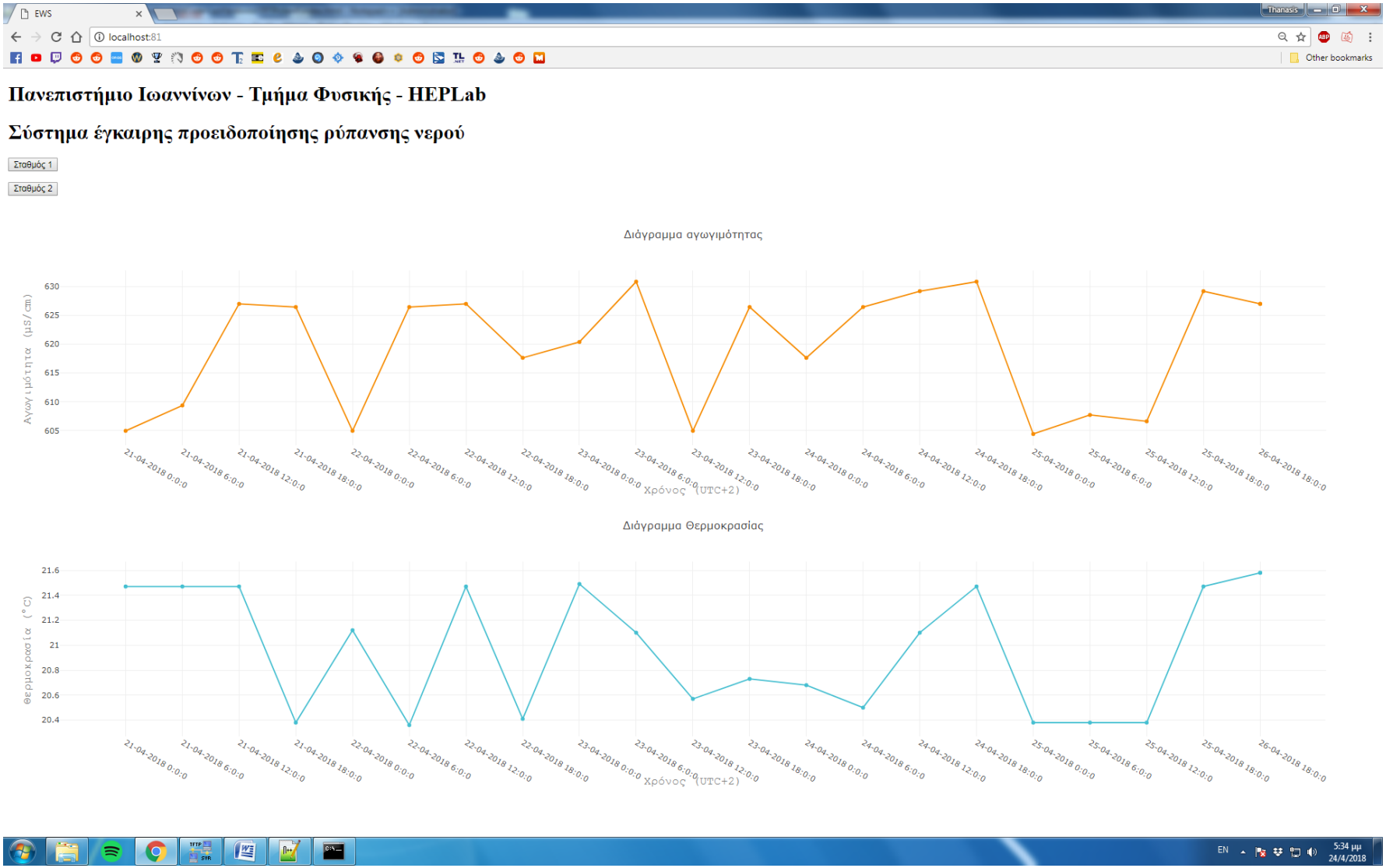
Για την μετάδοση των μετρήσεων που λαμβάνονται από το σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας στον τερματικό κόμβο διαδικτύου, που βρίσκεται στο Πανεπιστήμιο Ιωαννίνων, υλοποιήθηκε, ένας διακομιστής TFTP. Το ελάχιστο επίπεδο πολυπλοκότητας ενός συστήματος συμβατού με το Διαδίκτυο των Αντικειμένων (Internet of things, IoT), περιλαμβάνει ένα αντικείμενο το οποίο έχει μοναδική ταυτότητα και τα δεδομένα που είναι αποθηκευμένα σε αυτό μπορούν να αναγνωσθούν από οπουδήποτε και οποτεδήποτε μέσω μίας εφαρμογής διαδικτύου [34]. Το σύστημα που αναπτύχθηκε έχει μοναδική ταυτότητα, την διεύθυνση IP του διακομιστή TFTP που υλοποιείται και τα δεδομένα που είναι αποθηκευμένα μπορούν να αναγνωσθούν οποιαδήποτε στιγμή και από οπουδήποτε μέσω μίας ιστοσελίδας, η οποία φιλοξενείται σε διακομιστή Web. Ο διακομιστής Web που υλοποιείται σε ηλεκτρονικό υπολογιστή στον κόμβο διαδικτύου που βρίσκεται στο Πανεπιστήμιο Ιωαννίνων. Δηλαδή, το σύστημα καλύπτει τις ελάχιστες απαιτήσεις συμβατότητας με το Διαδίκτυο των Αντικειμένων.

Το πρωτόκολλο Trivial File Transfer Protocol (TFTP) είναι ένα πρωτόκολλο για αποστολή αρχείων μέσω διαδικτύου. Οι λειτουργίες που υποστηρίζει είναι λήψη αρχείων ή αποστολή αρχείων σε ένα διακομιστή. Επιλέχθηκε λόγω της περιορισμένης μνήμης που καταλαμβάνει στο σύστημα που υλοποιείται. Μία διαδικασία (session) λήψης αρχείου μέσω του πρωτοκόλλου TFTP ακολουθεί τα παρακάτω στάδια:

1. Το εξυπηρετούμενο σύστημα TFTP (TFTP client) στέλνει στον διακομιστή TFTP (TFTP Server) μία αίτηση λήψης αρχείου RRQ (read request), εντολή GET, μαζί με το όνομα του αρχείου προς λήψη.
2. Στη συνέχεια, ο διακομιστής TFTP αποστέλλει το αρχείο στο εξυπηρετούμενο σύστημα.
3. Όταν ολοκληρωθεί η αποστολή του αρχείου η σύνδεση τερματίζεται [31].

Στο σύστημα που αναπτύχθηκε ο διακομιστής TFTP υλοποιείται στον μικροεπεξεργαστή Microblaze του συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας και ο εξυπηρετούμενος TFTP στον ηλεκτρονικό υπολογιστή του τερματικό κόμβο διαδικτύου στο Πανεπιστήμιο Ιωαννίνων.

Τα δεδομένα που λαμβάνονται από το εξυπηρετούμενο σύστημα TFTP, αποθηκεύονται σε ηλεκτρονικό υπολογιστή στον κεντρικό κόμβο διαδικτύου. Εκεί αναλύονται και `στη συνέχεια παρουσιάζονται, σε μορφή διαγράμματος, σε ιστοσελίδα που φιλοξενείται σε διακομιστή Web. Η πρόσβαση σε αυτή γίνεται πληκτρολογώντας σε ένα φυλλομετρητή διαδικτύου, όπως ο Google Chrome, τη διεύθυνση http://195.115.130.180:81. Η ιστοσελίδα αυτή αποτελεί τη διεπαφή χρήστη του συστήματος. Στην εικόνα 51 παρουσιάζεται η ιστοσελίδα που αναπτύχθηκε:



Εικόνα 51: Διεπαφή χρήστη

Με επιλογή του αντίστοιχου πλήκτρου, π.χ σταθμός 1, της ιστοσελίδας εμφανίζονται τα γραφήματα των τιμών αγωγιμότητας και θερμοκρασίας του νερού που έχουν ληφθεί από το σύστημα, στον συγκεκριμένο σταθμό. Η δημιουργία του διακομιστή Web που φιλοξενεί την ιστοσελίδα, η οποία αποτελεί την διεπαφή χρήστη του συστήματος έγινε σε γλώσσα JavaScript. Ο κώδικας του διακομιστή εκτελείται στην εφαρμογή ανοιχτού λογισμικού Node.js [32].

Η διεπαφή χρήστη υλοποιήθηκε ακολουθώντας τα παρακάτω βήματα:

1. Δημιουργείται στον υπολογιστή του τερματικού κόμβου διαδικτύου ο φάκελος με όνομα «DiepafiXristi», που περιέχει τα αρχεία από τα οποία αποτελείται η διεπαφή χρήστη. Δημιουργούνται τα αρχεία TFTPclient.bat, slicing.js, server.js, plot.js, index.html και οι φάκελοι Plot, Received Files, που περιγράφονται παρακάτω.
2. Ενεργοποιείται η εφαρμογή TFTP client των Microsoft Windows.

Το εξυπηρετούμενο σύστημα TFTP ενεργοποιείται με ένα αρχείο εντολών (batch), το αρχείο με όνομα TFTPclient.bat, το οποίο εκτελεί τον παρακάτω κώδικα:

:start
tftp -i 192.168.1.10 get file1.txt
timeout 10
goto start

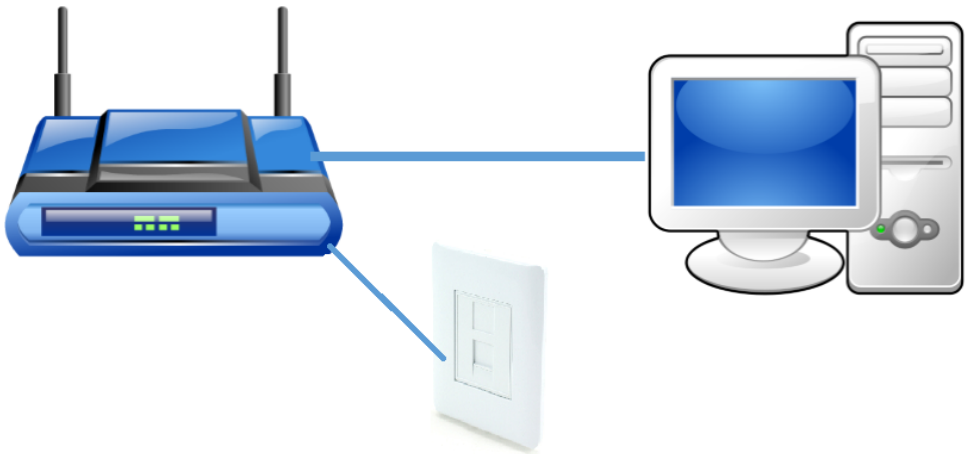
Οι εντολές του αρχείου δημιουργούν έναν ατέρμων βρόγχο, στον οποίο το εξυπηρετούμενο σύστημα TFTP στέλνει μία εντολή λήψης αρχείου στο διακομιστή TFTP,ο οποίος έχει διεύθυνση IP την διεύθυνση IP του συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης του εκάστοτε σταθμού. Όταν ο διακομιστής TFTP δεν λειτουργεί ή το αρχείο με όνομα file1.txt υπάρχει στο φάκελο με τα αρχεία της διεπαφής χρήστη, δεν λαμβάνεται το αρχείο και η εκτέλεση του κώδικα καθυστερεί για 10 δευτερόλεπτα μέσω της εντολής καθυστέρησης timeout. Όταν ο διακομιστής TFTP λειτουργεί και το αρχείο δεν υπάρχει στο φάκελο, το αρχείο αποστέλλεται από τον διακομιστή στο εξυπηρετούμενο σύστημα TFTP.

3. Το αρχείο με όνομα slicing.js (αρχείο JavaScript) εκτελεί μία δέσμη ενεργειών του εξυπηρετητή και είναι υπεύθυνο για τον έλεγχο ύπαρξης του αρχείου file1.txt και τη μετατροπή των δεδομένων του σε δεδομένα μορφής συμβατής με αρχείο κατάληξης .csv. Επίσης, επειδή οι τιμές αγωγιμότητας και θερμοκρασίας που περιέχονται στο αρχείο file1.txt είναι τα ADC σε μορφή μετρήσεων (counts), το αρχείο slicing.js με χρήση των σχέσεων που υπολογίστηκαν κατά τη βαθμονόμηση του αισθητήρα (παρ. 6.2) μετατρέπει τα ADC counts σε πραγματικές τιμές αγωγιμότητας και θερμοκρασίας. Σε περίπτωση που αλλάξει ο αισθητήρας του συστήματος οι σχέσεις που μετατρέπουν τα ADC counts σε πραγματικές τιμές αγωγιμότητας και θερμοκρασίας πρέπει να αντικατασταθούν από τις νέες σχέσεις που θα προκύψουν από τη βαθμονόμηση του νέου αισθητήρα. Μετά την μετατροπή και την μορφοποίηση των δεδομένων εξάγονται δύο αρχεία τύπου .csv με όνομα Data.csv και DD_MM_YYYY.csv, όπου DD η ημέρα, MM ο μήνας, YYYY το έτος της

λήψης του αρχείου από τον διακομιστή TFTP. Το Data.csv αποθηκεύεται στο φάκελο Plot και το DD_MM_YYYY.csv στο φάκελο Received Files. Έπειτα, διαγράφεται το αρχείο file1.txt από το φάκελο «DiepafiXristi».

4. Το αρχείο με όνομα plot.js (αρχείο JavaScript) εκτελεί μία δέσμη ενεργειών εξυπηρετητή web, είναι υπεύθυνο για τη δημιουργία των γραφημάτων στην ιστοσελίδα, χρησιμοποιώντας την βιβλιοθήκη JavaScript Plotly.js [33].

5. Τέλος, τα αρχεία με όνομα server.js και index.html είναι υπεύθυνα για τη λειτουργία του διακομιστή Web και τη μορφοποίηση της ιστοσελίδας που φιλοξενείται στο διακομιστή. Το αρχείο server.js περιέχει κώδικα σε γλώσσα JavaScript και το αρχείο index.html σε γλώσσα HTML. Ο διακομιστής Web υλοποιείται στη θύρα 81 του υπολογιστή του τερματικού κόμβου διαδικτύου και αρχικά είναι προσβάσιμος μόνο από συσκευές που συνδέονται στο τοπικό δίκτυο. Η επιλογή της θύρας 81 δεν εξυπηρετεί κάποιο συγκεκριμένο σκοπό, μπορεί να χρησιμοποιηθεί οποιαδήποτε άλλη θύρα που δεν χρησιμοποιείται από κάποια εφαρμογή. Για να είναι εφικτή η πρόσβαση στην ιστοσελίδα από το διαδίκτυο η θύρα 81 προωθήθηκε μέσω του δρομολογητή του τερματικού κόμβου διαδικτύου. Στην εικόνα 52 φαίνεται η συνδεσμολογία του τερματικού κόμβου διαδικτύου. Ο κώδικας των παραπάνω αρχείων παρατίθεται στο παράρτημα Δ και η εκτέλεσή τους στον υπολογιστή που βρίσκεται στον τερματικό κόμβο διαδικτύου πρέπει να γίνει μία φορά, όταν δημιουργείται ο διακομιστής Web. Η αρχική εκτέλεση του αρχείου .bat γίνεται με επιλογή του αρχείου και έπειτα επιλογή του **Open** στη γραμμή εργαλείων. Η αρχική εκτέλεση των υπόλοιπων αρχείων γίνεται ανοίγοντας την εφαρμογή command prompt των Microsoft Windows και στη συνέχεια πληκτρολογείται η εντολή: node ονομα_αρχείου.js για κάθε ένα από τα αρχεία με κατάληξη .js, εκτός από το αρχείο Plot.js το οποίο εκτελείται αυτόματα μέσω του αρχείου server.js όταν ο επισκέπτης της ιστοσελίδας επιθυμεί να δει το γράφημα των μετρήσεων κάποιου σταθμού.



Εικόνα 52: Συνδεσμολογία του τερματικού κόμβου διαδικτύου

6.4 Ενεργειακή κατανάλωση

Η τροφοδοσία του συστήματος που αναπτύχθηκε γίνεται από υπερπυκνωτές που φορτίζονται από δύο φωτοβολταϊκά στοιχεία. Χρησιμοποιούνται δύο συστοιχίες των έξι υπερπυκνωτών χωρητικότητας 3000F της εταιρίας Maxwell [37], συνδεδεμένες παράλληλα. Οι υπερπυκνωτές κάθε συστοιχίας συνδέονται σε σειρά μεταξύ τους και παρέχουν, όταν είναι πλήρως φορτισμένοι, μέγιστη τάση εξόδου στα άκρα τους 16.2V. Η συνολική χωρητικότητα $C_{ολ}$ κάθε συστοιχίας δίνεται από την σχέση: $\frac{1}{C_{ολ}} = \frac{1}{C_1} + \frac{1}{C_2} + \cdots + \frac{1}{C_6}$ και είναι 500F. Η συνολική χωρητικότητα των δύο συστοιχιών είναι 1000F, διότι συνδέονται παράλληλα και το αποτέλεσμα προκύπτει από την σχέση: $C_{ολ} = C_1 + C_2 \rightarrow C_{ολ} = 500F + 500F = 1000F$.

Η κατανάλωση ρεύματος του συστήματος δειγματοληψίας - μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας μετρήθηκε στο εργαστήριο και τα αποτελέσματα παρουσιάζονται στον πίνακα 8:

Λειτουργία συστήματος	**Κατανάλωση (mA)**
Αναμονή	15
Δειγματοληψία	365-568
Λήψη – αποθήκευση μετρήσεων/ Υλοποίηση διακομιστή TFTP	365
Συγχρονισμός (μέσω GPS)	365

Πίνακας 8: Κατανάλωση ρεύματος

Η διαφορά στην τιμή κατανάλωσης κατά τη διάρκεια της δειγματοληψίας οφείλεται στον τρόπο λειτουργίας των ηλεκτρικών βαλβίδων. Η μικρότερη από τις δύο τιμές αντιστοιχεί στην κατάσταση αναμονής των βαλβίδων, ενώ η μεγαλύτερη όταν περιστρέφονται και οι δύο βαλβίδες. Η περιστροφή των βαλβίδων διαρκεί 26.2 δευτερόλεπτα.

Η ωφέλιμη ηλεκτρική ενέργεια που παρέχουν οι υπερπυκνωτές στο σύστημα είναι 26095 J υπολογισμένη από την σχέση:

$E = \frac{1}{2} C (V_{αρχ}^2 - V_{τελ}^2)$, όπου $V_{αρχ}$=16.2V, $V_{τελ}$=14.5V και C=1000F.

E = 26095 J.

Επιλέχθηκαν αυτές οι τιμές τάσης τροφοδοσίας γιατί, 16.2V είναι η μέγιστη τάση που παρέχουν οι υπερπυκνωτές και 14.5 η ελάχιστη τάση που απαιτείται από το σύστημα για την ορθή λειτουργία του. Η ενεργειακή κατανάλωση του συστήματος κατά το χρονικό διάστημα που δεν υπάρχει ηλιοφάνεια, δηλαδή τις ώρες 18:00, 00:00, και 06:00 παρουσιάζεται στον πίνακα 9:

Ώρα λειτουργίας	**Χρονικό διάστημα λειτουργίας (λεπτά)**	**Ενεργειακή κατανάλωση (J)**
18:00	10	5348

00:00	6.2	2199
06:00	6.2	2199

Πίνακας 9: Ενεργειακή κατανάλωση συστήματος δειγματοληψίας - μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας

Στην ενεργειακή κατανάλωση της ώρας 18:00 έχει προστεθεί η ενέργεια που καταναλώνει η κεραία αναμετάδοσης πληροφορίας των 6W για χρονικό διάστημα πέντε λεπτών, και είναι 1800 J. Για το παραπάνω χρονικό διάστημα 12 ωρών το σύστημα καταναλώνει επιπλέον 10171 J γιατί καταναλώνει ρεύμα 15mA στην κατάσταση αναμονής. Η συνολική ενεργειακή κατανάλωση του συστήματος για το παραπάνω χρονικό διάστημα είναι 19917 J. Άρα η ωφέλιμη ενέργεια που είναι αποθηκευμένη στους υπερπυκνωτές καλύπτει τις ενεργειακές ανάγκες του συστήματος για το χρονικό διάστημα που οι υπερπυκνωτές δεν φορτίζονται. Επιπλέον, ο ρυθμιστής τάσης στο κύκλωμα σταθεροποίησης τάσης φόρτισης των υπερπυκνωτών (παράγραφος 4.2.10) έχει απόδοση 94%, οπότε οι υπερπυκνωτές φορτίζονται πλήρως, δηλαδή από 14.5V στα 16.2V, σε:

- 11.4 λεπτά, όταν τα φωτοβολταϊκά στοιχεία αποδίδουν τη μέγιστη ισχύ τους δηλαδή 20W+20W=40W,
- 22.8 λεπτά, όταν αποδίδουν ισχύ 20W.

Τις ώρες 12:00, 14:00 η ενεργειακή κατανάλωση του συστήματος είναι 2199 J για κάθε κύκλο λειτουργίας.

Το σύστημα αναμετάδοσης πληροφορίας λειτουργεί δύο φορές τη μέρα, τις ώρες 14:00, 18:00. Την ώρα 14:00 εκτελεί τη διαδικασία συγχρονισμού και την ώρα 18:00 ενεργοποιεί την κεραία αναμετάδοσης. Η ενεργειακή του κατανάλωση για μία μέρα λειτουργίας φαίνεται στον πίνακα 10:

Ώρα λειτουργίας	**Χρονικό διάστημα λειτουργίας (λεπτά)**	**Ενεργειακή κατανάλωση (J)**
14:00	6.2	904
18:00	5	1800
Αναμονή	1429	20835

Πίνακας 10: Ενεργειακή κατανάλωση συστήματος αναμετάδοσης πληροφορίας

7. Συμπεράσματα

Το ασύρματο ενεργειακά αυτόνομο σύστημα έγκαιρης προειδοποίησης που αναπτύχθηκε στο Εργαστήριο Υψηλών Ενεργειών του Τμήματος Φυσικής για την παρούσα μεταπτυχιακή διπλωματική εργασία, είναι ικανό να μετρά τις φυσικοχημικές παραμέτρους του νερού ενός ποταμού, να τις αποθηκεύει σε κάρτα μνήμης μSD και να τις αποστέλλει σε τερματικό κόμβο διαδικτύου για περαιτέρω ανάλυση. Για τον συγχρονισμό του συστήματος με τους σταθμούς αναμετάδοσης χρησιμοποιήθηκε ρολόι πραγματικού χρόνου και ηλεκτρονική πλακέτα με δέκτη gps που χρησιμοποιείται για ενδεχόμενη διόρθωση της ώρας του RTC. Η δειγματοληψία γίνεται ελέγχοντας, μέσω ηλεκτρονόμων στερεάς κατάστασης, δύο ηλεκτρικές βαλβίδες οι οποίες σε συνδυασμό με μία μηχανική αντλία είναι υπεύθυνες για την άντληση νερού στο δοχείο δειγματοληψίας και το άδειασμα του. Το σύστημα τροφοδοτείται με ηλεκτρική ενέργεια από σύστημα που περιλαμβάνει φωτοβολταϊκά στοιχεία και υπερπυκνωτές. Για την πλήρη λειτουργία του συστήματος η τάση τροφοδοσίας του, που παρέχεται από τους υπερπυκνωτές, πρέπει να είναι μεγαλύτερη από 14.5V, με μέγιστη τιμή τα 16.2V. Με χρήση διακοπτικών ρυθμιστών τάσης μετατρέπεται σε 12V, 5V, 3.3V, που απαιτούνται για την ορθή λειτουργία των τυποποιημένων ηλεκτρονικών και ηλεκτρικών στοιχείων του συστήματος. Η τάση 12V χρειάζεται για τη λειτουργία του επιτηρητή τάσης MAX6460, του τελεστικού ενισχυτή LM324, του αισθητήρα WQ-Cond, των ηλεκτρικών βανών και της κεραίας αναμετάδοσης πληροφορίας. Η τάση 5V χρειάζεται για τη λειτουργία του FPGA, του μετατροπέα αναλογικού σήματος σε ψηφιακό AD7998, του GPS και των αισθητήρων στάθμης. Η τάση 3.3V χρειάζεται για τη λειτουργία της κάρτας μνήμης μSD. Η κεντρική μονάδα επεξεργασίας του συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης της πληροφορίας και του αναμεταδότη είναι η ηλεκτρονική πλακέτα Avnet Spartan-6 LX9 Microboard, η οποία χρησιμοποιεί ένα FPGA Spartan-6 της εταιρίας Xilinx. Στο FPGA υλοποιήθηκε ο μικροεπεξεργαστής MicroBlaze της εταιρίας Xilinx, στον οποίο λειτουργεί το ακολουθιακό τμήμα της συγκεκριμένης εφαρμογής.

Οι διαστάσεις του ηλεκτρονικού μέρους του συστήματος είναι 24 x 11.9 x 3.5 cm και τοποθετήθηκε σε κουτί για προστασία.

Οι φυσικοχημικές παράμετροι που μετρούνται είναι η θερμοκρασία και η αγωγιμότητα. Η αγωγιμότητα είναι δείκτης ρύπανσης του νερού καθώς η ύπαρξη αποβλήτων, φυτοφαρμάκων ή άλλων ρυπογόνων ουσιών στο νερό αυξάνουν την τιμή της αγωγιμότητας του. Η αποστολή τους στον τερματικό κόμβο διαδικτύου, που βρίσκεται στο Εργαστήριο Υψηλών Ενεργειών του Τμήματος Φυσικής, γίνεται μέσω ενός διακομιστή TFTP που υλοποιήθηκε στο FPGA. Επίσης, αναπτύχθηκε ιστοσελίδα διεπαφή χρήστη – συστήματος στην οποία εμφανίζονται οι γραφικές παραστάσεις των μετρήσεων. Για την μείωση της κατανάλωσης ηλεκτρικής ενέργειας, το σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης ενεργοποιείται 5 φορές την ημέρα, τις ώρες 12πμ,06πμ,12μμ,02μμ,06μμ και ο αναμεταδότης μία φορά στις 06μμ.

Η αυτονομία του συστήματος χωρίς ηλιοφάνεια είναι περίπου 18 ώρες. Θα μπορούσε να βελτιωθεί με την προσθήκη κυκλώματος διακόπτη, αποτελούμενο από τύπου n-MosFET που λειτουργεί ως λογική πύλη OR ανάμεσα από τους υπερπυκνωτές και το σύστημα Δ-Μ-Α-Μ, καθώς θα μειωθεί σημαντικά η κατανάλωση ενέργειας από το σύστημα στην κατάσταση αναμονής, θεωρητικά θα είναι περίπου 2731 J για χρονικό διάστημα 1405 λεπτών και η νέα αυτονομία του συστήματος θα είναι 37 ώρες.

Αναφορές

[1] **ISDR,** Basics of early warning, https://www.unisdr.org/2006/ppew/whats-ew/basics-ew.htm

[2] **Κ. Καλαϊτζάκης, Ε. Κουτρούλης,** Ηλεκτρικές μετρήσεις και αισθητήρες: Αρχές λειτουργίας και σχεδιασμός των ηλεκτρονικών συστημάτων μέτρησης, Εκδόσεις Κλειδάριθμος, Αθήνα 2010.

[3] **IEEE,** Towards a definition for the Internet of Things (IoT), Revision 1, 27 May 2015.

[4] **Χ. Αυδίκου,** Ανάπτυξη συστήματος μέτρησης της ποιότητας νερού με απομακρυσμένη διαχείριση μέσω παγκόσμιου ιστού, Μεταπτυχιακή Διπλωματική Εργασία, Ιωάννινα 2013.

[5] **Α. Μπόχλος,** Ανάπτυξη ενεργειακά αυτόνομου ασύρματου συστήματος αναμετάδοσης ψηφιακής πληροφορίας, Μεταπτυχιακή Διπλωματική Εργασία, Ιωάννινα 2013.

[6] **S. Brown**, **Z. Vranesic,** Σχεδίαση ψηφιακών συστημάτων με τη γλώσσα VHDL, Εκδόσεις Τζιόλα, 3η έκδοση, Θεσσαλονίκη 2011.

[7] **Avnet,** LX9 Microboard - https://www.avnet.com/shop/us/products/avnet-engineering-services/aes-s6mb-lx9-g-3074457345628965461/

[8] **Xilinx,** MicroBlaze Soft Processor Core - https://www.xilinx.com/products/design-tools/microblaze.html

[9] **Rife Hydraulic Engine Mfg. Co.,** Rife River Pump RP-100 http://www.riferam.com/xcart/product.php?productid=17520

[10] **Global Water Instrumentation.Inc,** WQ-COND CONDUCTIVITY SENSORS - http://www.globalw.com/products/WQ-cond.html

[11] **Fundamentals of environmental measurements,** Conductivity, Salinity & Total Dissolved Solids – http://www.fondriest.com/environmental-measurements/parameters/water-quality/conductivity-salinity-tds/

[12] **Maxim,** MAX 6460 - https://datasheets.maximintegrated.com/en/ds/MAX6457-MAX6460.pdf

[13] **Maxim,** DS 1337 - https://datasheets.maximintegrated.com/en/ds/DS1337-DS1337C.pdf

[14] **Adafruit,** Adafruit Ultimate GPS Breakout – https://www.adafruit.com/product/746

.

[15] **SanDisk,** microSDHC – https://www.sandisk.com/home/memory-cards/microsd-cards/sandisk-microsd

[16] **Analog Devices,** AD7998, 8-Channel, 12-Bit ADC with I^2C Compatible Interface in 20-Lead TSSOP – http://www.analog.com/media/en/technical-documentation/data-sheets/AD7997_7998.pdf

[17] **Mouser,** Panasonic PhotoMOS Relays – https://www.mouser.com/ds/2/315/AQY28XXEH-Datasheet-1196596.pdf

[18] **Panasonic,** How a PhotoMOS relay works – https://na.industrial.panasonic.com/blog/how-photomos-solid-state-relay-works

[19] **Texas Instruments,** PTN78020W - 6-A, Wide-Input adjustable switching regulator - http://www.ti.com/lit/ds/symlink/ptn78020w.pdf

[20] **Texas Instruments,** PTR08100W – 10-A, 4.5-V to 14-V input, non – isolated, adjustable wide – output, switching regulator – http://www.ti.com/lit/ds/slts284f/slts284f.pdf

[21] **Fairchild Semiconductor,** LM324 Quad Operational Amplifier – https://www.mouser.com/ds/2/308/LM324A-1120979.pdf

[22] **Advanced Monolithic Systems,** AMS1117 1A LOW DROPOUT VOLTAGE REGULATOR – http://www.advanced-monolithic.com/pdf/ds1117.pdf

[23] **Digilent,** Digilent Pmod™ Interface Specification – https://www.digilentinc.com/Pmods/Digilent-Pmod_%20Interface_Specification.pdf

[24] **HxD - Freeware Hex Editor and Disk Editor,** https://mh-nexus.de/en/hxd/

[25] **KiCad EDA,** http://kicad-pcb.org/

[26] **V. Pedroni,** Σχεδιασμός κυκλωμάτων με τη VHDL, 2007, Εκδόσεις Κλειδάριθμος, Αθήνα, 1η έκδοση.

[27] **H. Schildt,** Οδηγός της C, Εκδόσεις Μ. Γκιούρδας, Αθήνα 2006, 4η έκδοση.

[28] **SD Association,** Simplified specifications, https://www.sdcard.org/downloads/pls/

[29] **lwIP - A Lightweight TCP/IP stack – Summary,** https://savannah.nongnu.org/projects/lwip/

[30] **Ν. Λούκας,** Ανάπτυξη διεπαφών αισθητήρων – μικροελεγκτή για ασύρματο σύστημα μετάδοσης ψηφιακής πληροφορίας, Μεταπτυχιακή Διπλωματική Εργασία, Ιούλιος 2009

[31] **K. Sollins,** The TFTP protocol (Revision 2), MIT, July 1992, https://tools.ietf.org/html/rfc1350

[32] **Node.js,** https://nodejs.org/en/

[33] **Plotly.js,** https://plot.ly/javascript/

[34] **NanJing Top Power ASIC Corp,** TP4056, https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf

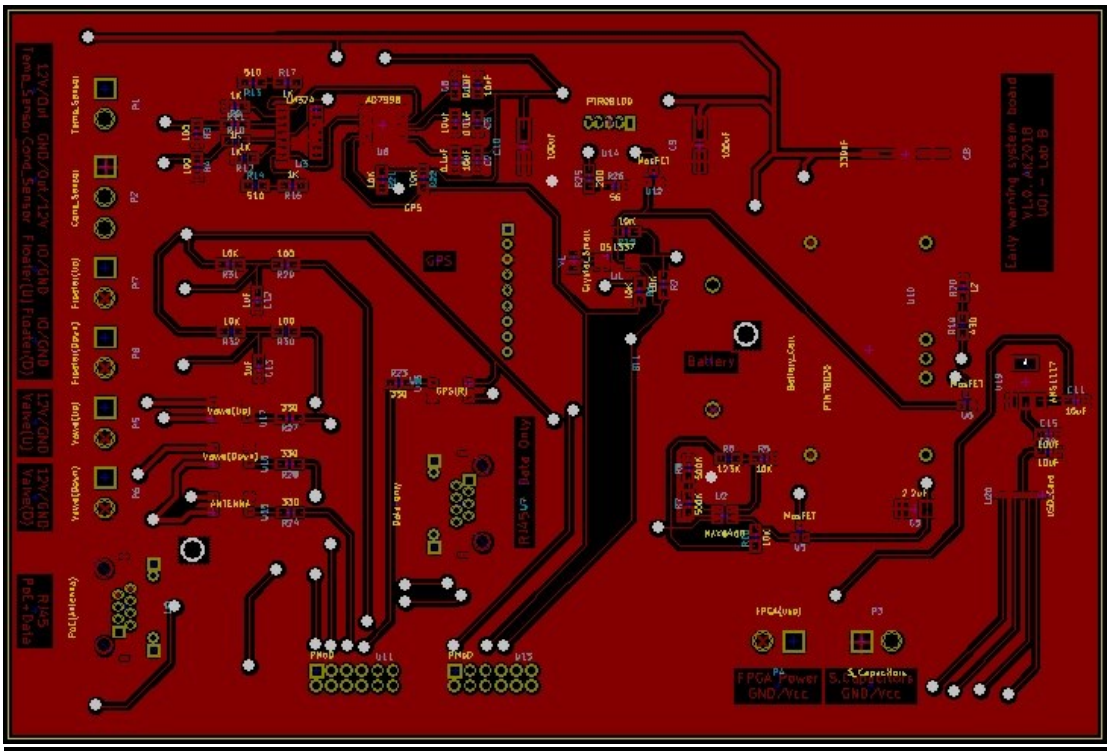
[35] **Texas Instruments,** LM317 3-Terminal Adjustable Regulator, http://www.ti.com/lit/ds/symlink/lm317.pdf

[36] **Plastim PP3004,** http://plastim.com.tr/en/plastic-distribution-boards-grey-door/

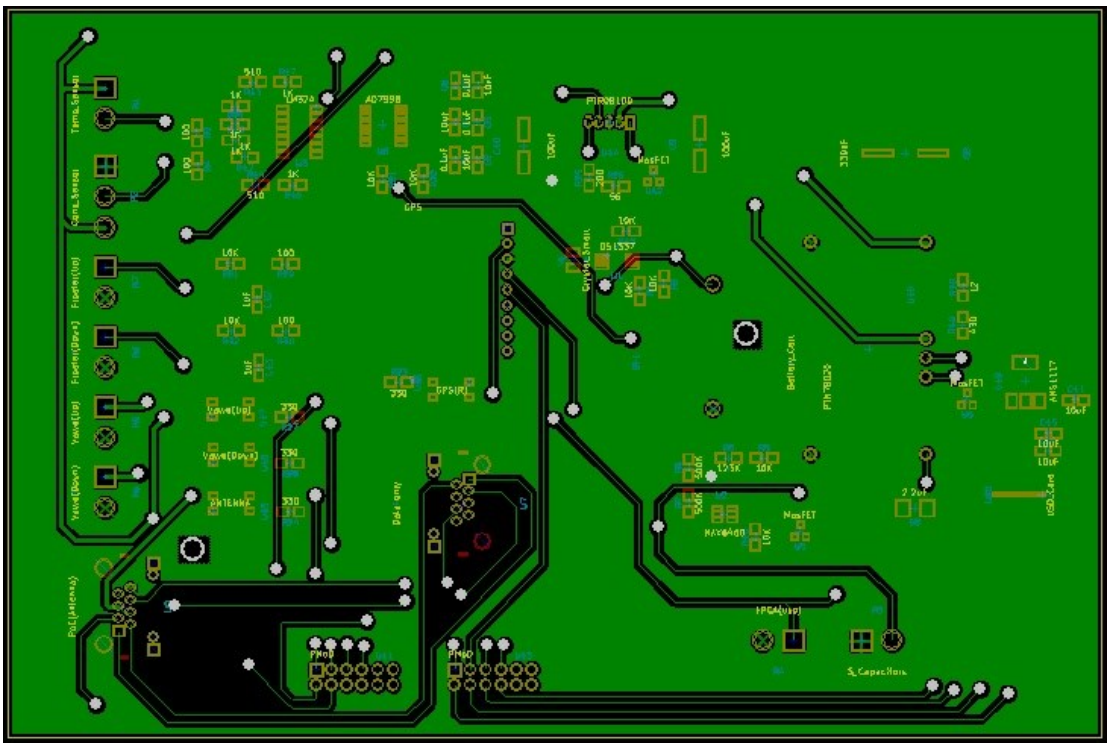
[37] **Maxwell Technologies,** K2 2.7V Series, http://www.maxwell.com/products/ultracapacitors/k2-series/documents

[38] **Μαργαρίτα Καψή,** Εργαστήριο Ελέγχου Προστασίας και Τεχνολογίας Περιβάλλοντος, Τμήμα Χημείας, Πανεπιστήμιο Ιωαννίνων.

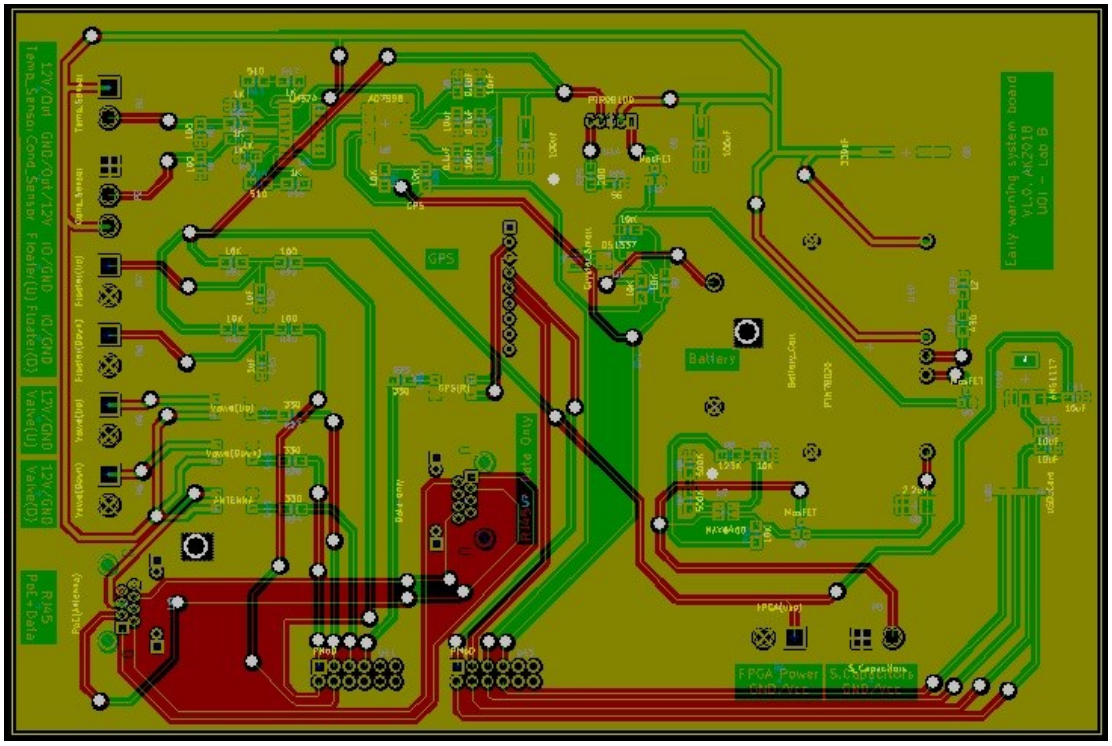
ΠΑΡΑΡΤΗΜΑ Α.: <u>Σχέδια τυπωμένων κυκλωμάτων</u>



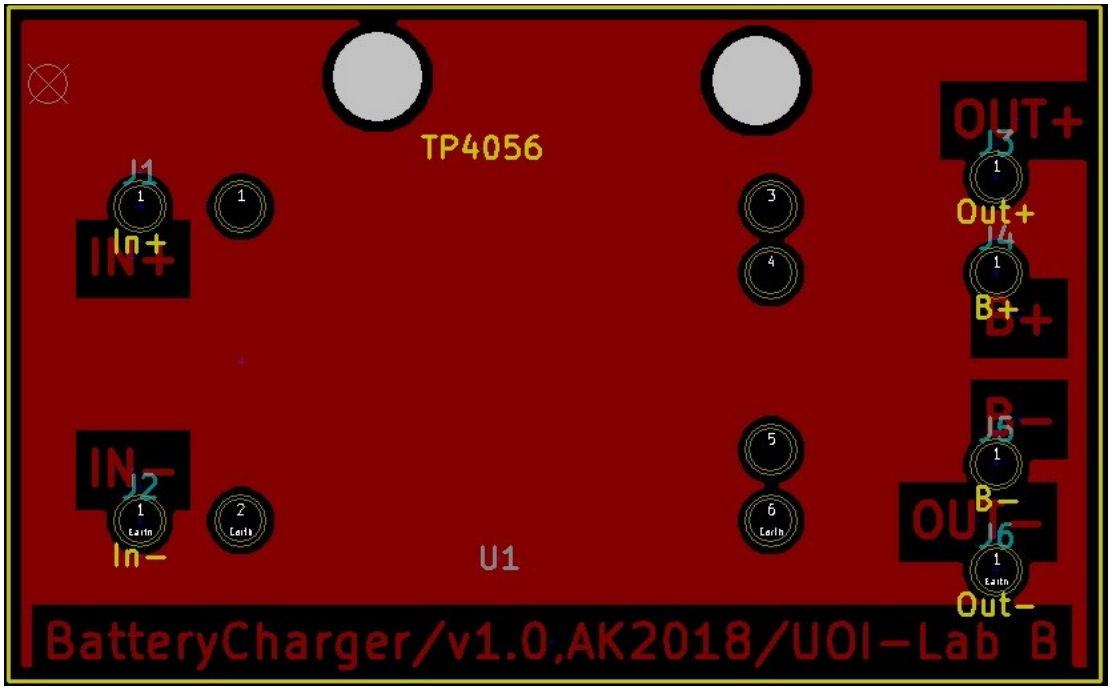
Εικόνα Α.1: Pcbnew KiCad, σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας, πάνω όψη



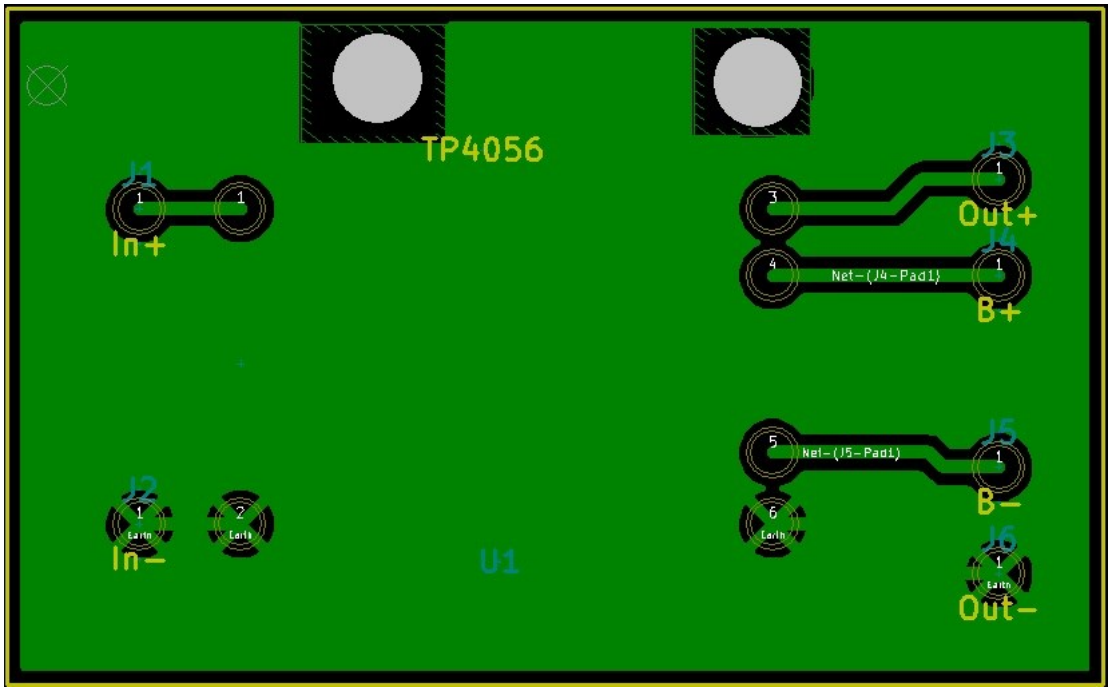
Εικόνα Α.2: Pcbnew KiCad, σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας, κάτω όψη



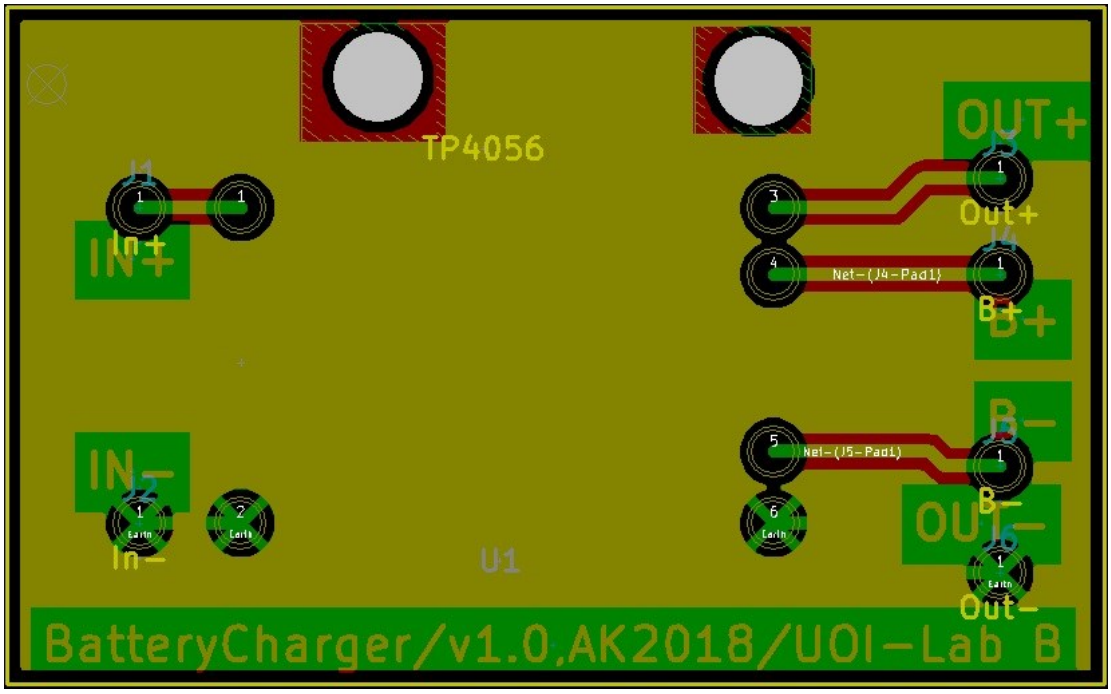
Εικόνα Α.3: Pcbnew KiCad, σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας , πάνω και κάτω όψεις



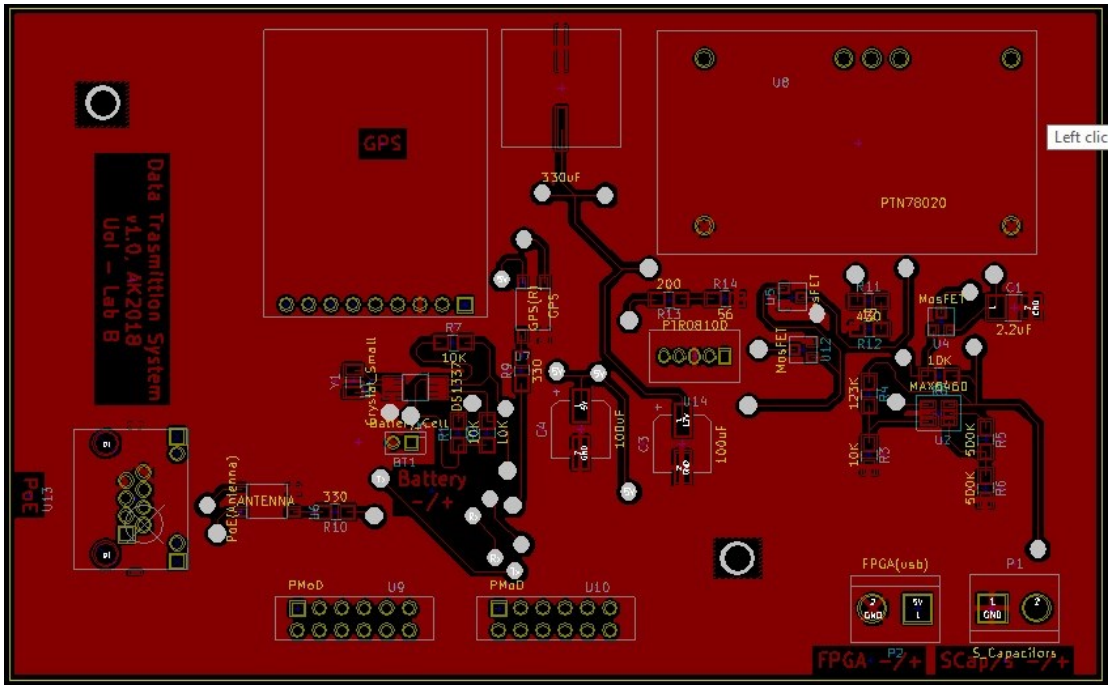
Εικόνα Α.4: Pcbnew KiCad, κύκλωμα φόρτισης μπαταρίας, πάνω όψη



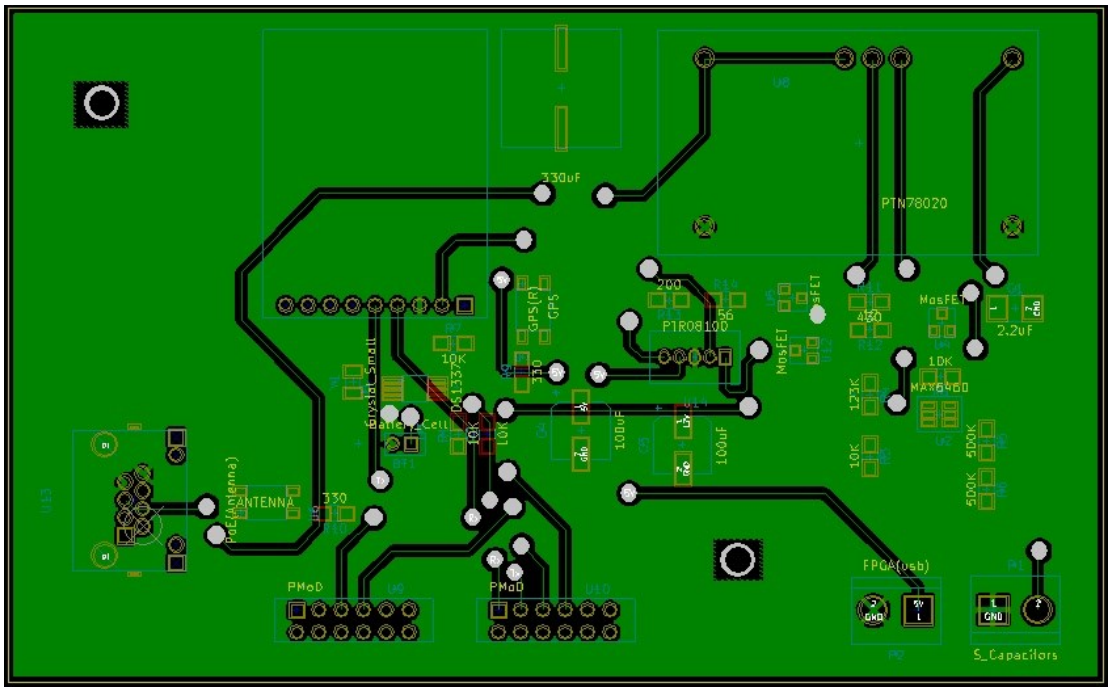
Εικόνα Α.5: Pcbnew KiCad, κύκλωμα φόρτισης μπαταρίας, κάτω όψη



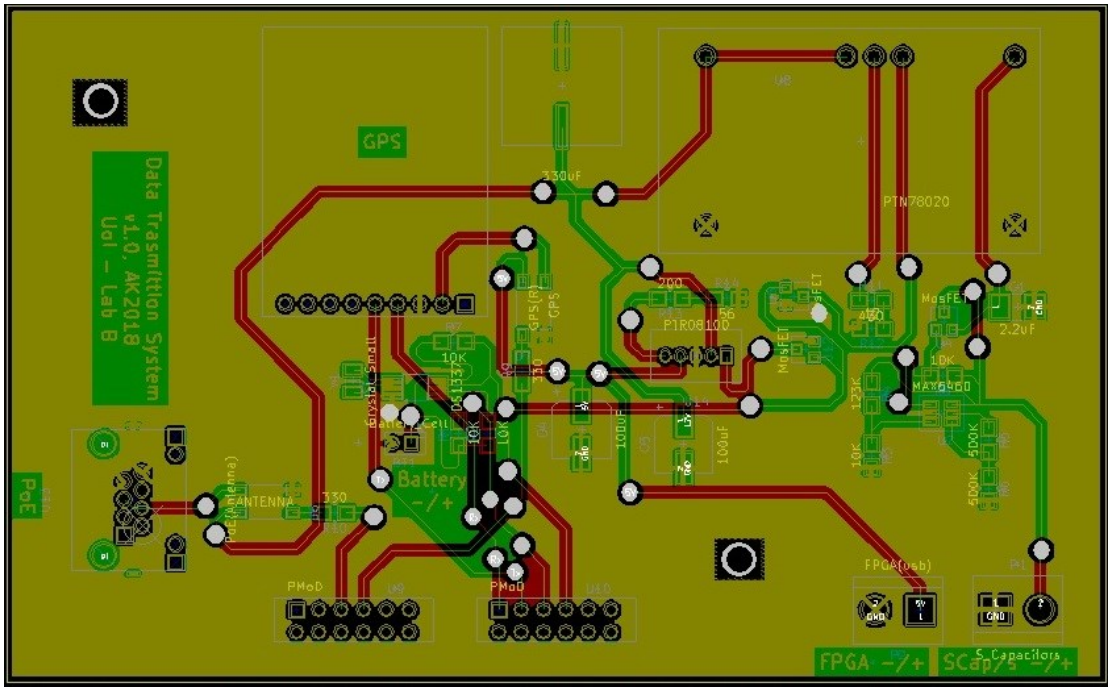
Εικόνα Α.6: Pcbnew KiCad, κύκλωμα φόρτισης μπαταρίας, πάνω και κάτω όψεις



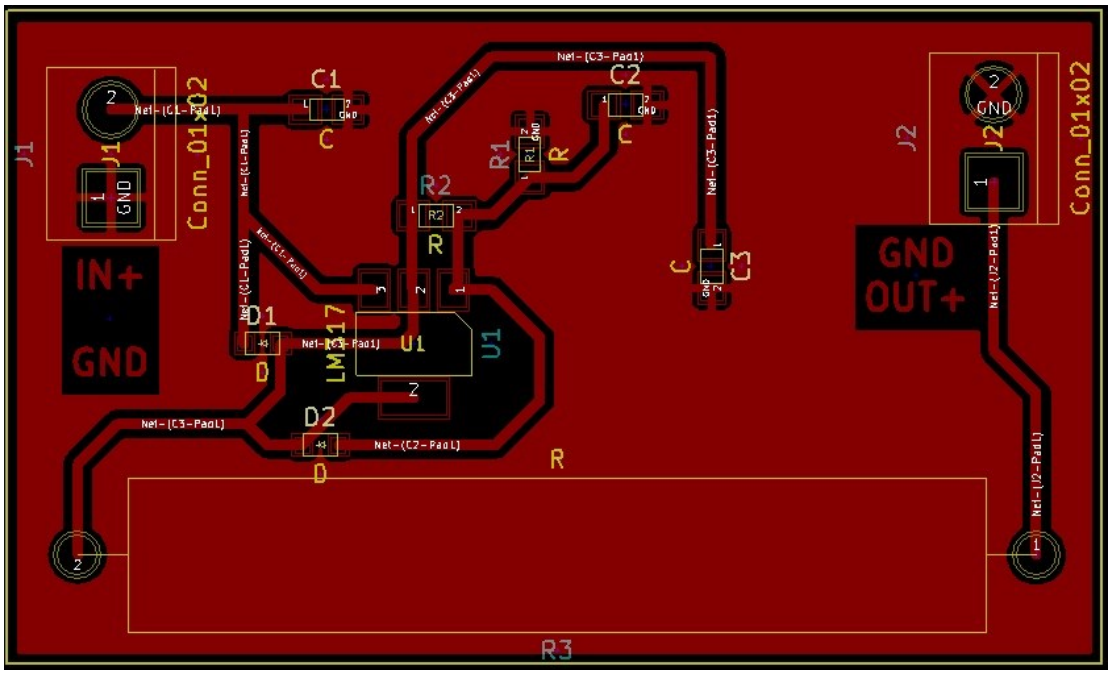
Εικόνα Α.7: Pcbnew KiCad, σύστημα αναμετάδοσης πληροφορίας, πάνω όψη



Εικόνα Α.8: Pcbnew KiCad, σύστημα αναμετάδοσης πληροφορίας, πάνω όψη



Εικόνα Α.9: Pcbnew KiCad, σύστημα αναμετάδοσης πληροφορίας, πάνω και κάτω όψεις



Εικόνα Α.10: Pcbnew KiCad, κύκλωμα σταθεροποίησης τάσης φόρτισης των υπερπυκνωτών, πάνω όψη

ΠΑΡΑΡΤΗΜΑ Β.: <u>Κώδικας σε γλώσσα C</u>

Ο προγραμματισμός του συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας έγινε με χρήση της γλώσσας προγραμματισμού C. Στον πίνακα Β.1 παρουσιάζονται οι μεταβλητές που χρησιμοποιήθηκαν στη συνάρτηση int main (void). Περιγραφή του κώδικα υπάρχει στην παράγραφο 5.3.1.

Όνομα μεταβλητής	**Περιγραφή**	**Αρχική τιμή**
Y	Οι τιμές που αντιστοιχούν στις ώρες ενεργοποίησης του συστήματος: 0x01 → 00:00 0x02 → 06:00 0x03 → 12:00 0x04 → 18:00 0x05 → 14:00	0x00
SecondsReg	Διεύθυνση του καταχωρητή δευτερολέπτων του RTC	0x00
MinutesReg	Διεύθυνση του καταχωρητή λεπτών του RTC	0x01
HoursReg	Διεύθυνση του καταχωρητή ώρας του RTC	0x02
DateReg	Διεύθυνση του καταχωρητή ημέρας του RTC	0x04
MonthReg	Διεύθυνση του καταχωρητή μήνα του RTC	0x05
YearReg	Διεύθυνση του καταχωρητή έτους του RTC	0x06
ControlReg	Διεύθυνση του καταχωρητή ελέγχου του RTC	0x0E
StatusReg	Διεύθυνση του καταχωρητή κατάστασης του RTC	0x0F
SecondsSetup[2]	Καταχώρηση στον καταχωρητή δευτερολέπτων του RTC. Το πρώτο στοιχείο του πίνακα είναι η διεύθυνση του καταχωρητή και το δεύτερο η τιμή που	{0x00,0x00}

	καταχωρείται	
MinutesSetup[2]	Καταχώρηση στον καταχωρητή λεπτών του RTC	{0x01,0x00}
HoursSetup[2]	Καταχώρηση στον καταχωρητή ώρας του RTC	{0x02,0x00}
DaySetup[2]	Καταχώρηση στον καταχωρητή ημέρας του RTC	{0x04,0x00}
MonthSetup[2]	Καταχώρηση στον καταχωρητή μήνα του RTC	{0x05,0x00}
YearSetup[2]	Καταχώρηση στον καταχωρητή έτους του RTC	{0x06,0x00}
Alarm2MinsSetup[2]	Καταχώρηση στον καταχωρητή λεπτών του Alarm2 του RTC	{0x0B,0x00}
Alarm2HoursSetup1[2]	Καταχώρηση στον καταχωρητή ώρας του Alarm2 του RTC, ώστε το Alarm2 να ενεργοποιηθεί στις 06:00	{0x0C,0x46}
Alarm2HoursSetup2[2]	Καταχώρηση στον καταχωρητή ώρας του Alarm2 του RTC, ώστε το Alarm2 να ενεργοποιηθεί στις 12:00	{0x0C,0x72}
Alarm2HoursSetup3[2]	Καταχώρηση στον καταχωρητή ώρας του Alarm2 του RTC, ώστε το Alarm2 να ενεργοποιηθεί στις 14:00	{0x0C,0x62}
Alarm2HoursSetup4[2]	Καταχώρηση στον καταχωρητή ώρας του Alarm2 του RTC, ώστε το Alarm2 να ενεργοποιηθεί στις 00:00	{0x0C,0x52}
Alarm2HoursSetup5[2]	Καταχώρηση στον καταχωρητή ώρας του Alarm2 του RTC, ώστε το Alarm2 να ενεργοποιηθεί στις 18:00	{0x0C,0x66}
Alarm2DaySetup[2]	Καταχώρηση στον καταχωρητή λεπτών του Alarm2 του RTC	{0x0D,0x80}
CtrlRegSetup[2]	Καταχώρηση στον καταχωρητή ελέγχου του RTC. Ρυθμίζει τον	{0x0E,0x06}

	ακροδέκτη INTB ως ακροδέκτη εξόδου του Alarm2	
StatusRegSetup[2]	Καταχώρηση στον καταχωρητή κατάστασης του RTC. Απενεργοποιεί το Alarm2	{0x0F,0x00}
ADCregT	Προγραμματίζει τον ADC να ψηφιοποιήσει το σήμα στην αναλογική είσοδο $V_{IN}1$ (μέτρηση θερμοκρασίας)	0x80
ADCregC	Προγραμματίζει τον ADC να ψηφιοποιήσει το σήμα στην αναλογική είσοδο $V_{IN}2$ (μέτρηση αγωγιμότητας)	0x90
Temp_data	Για ανάγνωση της ψηφιοποιημένης μέτρησης θερμοκρασίας	0x00
Cond_data	Για ανάγνωση της ψηφιοποιημένης μέτρησης αγωγιμότητας	0x00
Temp_DataM	Για ανάγνωση των πρώτων 4 bits της μέτρησης θερμοκρασίας	0x00
Temp_DataL	Για ανάγνωση των υπόλοιπων 8 bits της μέτρησης θερμοκρασίας	0x00
Cond_DataM	Για ανάγνωση των πρώτων 4 bits της μέτρησης αγωγιμότητας	0x00
Cond_DataL	Για ανάγνωση των υπόλοιπων 8 bits της μέτρησης αγωγιμότητας	0x00
GpsData[75]	Πίνακας για τα δεδομένα που λαμβάνονται από το GPS	Όλα τα στοιχεία του πίνακα έχουν τιμή 0x00
GpsCommand[51]	Προγραμματίζει το GPS να στέλνει μόνο την συμβολοσειρά εξόδου RMC.	Φαίνεται στο κάτω μέρος της σελίδας 97
SecMSB	Για ανάγνωση του πρώτου από τα δύο bytes της τιμής των δευτερολέπτων που λαμβάνεται από το GPS (χαρακτήρας ASCII)	0x00
SecLSB	Για ανάγνωση του δεύτερου από τα δύο bytes της τιμής των	0x00

	δευτερολέπτων που λαμβάνεται από το GPS (χαρακτήρας ASCII)	
MinMSB	Για ανάγνωση του πρώτου byte των λεπτών	0x00
MinLSB	Για ανάγνωση του δεύτερου byte των λεπτών	0x00
HourMSB	Για ανάγνωση του πρώτου byte της ώρας	0x00
HourLSB	Για ανάγνωση του δεύτερου byte της ώρας	0x00
DayMSB	Για ανάγνωση του πρώτου byte της ημέρας	0x00
DayLSB	Για ανάγνωση του δεύτερο byte την ημέρας	0x00
MonthMSB	Για ανάγνωση του πρώτου byte του μήνα	0x00
MonthLSB	Για ανάγνωση του δεύτερου byte του μήνα	0x00
YearMSB	Για ανάγνωση του πρώτου byte του έτους	0x00
YearLSB	Για ανάγνωση του δεύτερου byte του έτους	0x00
SecondsEnc	Για ανάγνωση της τιμής των δευτερολέπτων σε δεκαεξαδική μορφή	0x00
MinutesEnc	Για ανάγνωση της τιμής των λεπτών σε δεκαεξαδική μορφή	0x00
HoursEnc	Για ανάγνωση της τιμής της ώρας σε δεκαεξαδική μορφή	0x00
DayEnc	Για ανάγνωση της τιμής της ημέρας σε δεκαεξαδική μορφή	0x00
MonthEnc	Για ανάγνωση της τιμής του μήνα σε δεκαεξαδική μορφή	0x00
YearEnc	Για ανάγνωση της τιμής του έτους σε δεκαεξαδική μορφή	0x00
sector_data[3]	Για ανάγνωση της τιμής του sector 0x00000000 της κάρτας μνήμης μSD	{0x00,0x00,0x00}
sent_sector_data[3]	Για ανάγνωση της τιμής του sector 0x00000001 της κάρτας μνήμης μSD	{0x00,0x00,0x00}
Control_info[5]	Για ανάγνωση των bytes ενός sector που περιέχουν	{0x00,0x00,0x00, 0x00,0x00}

	ημερολογιακές πληροφορίες. Ελέγχονται από το λογισμικό και ανάλογα με την τιμή τους ακολουθείται διαφορετική διαδικασία	
sData[512]	Πίνακας για τα δεδομένα που αποστέλλονται από την κάρτα μνήμης μSD	Όλα τα στοιχεία του πίνακα έχουν τιμή 0x00
data	Για έλεγχο της ολοκλήρωσης των διαδικασιών αποστολής ή λήψης δεδομένων από την κάρτα μνήμης μSD	0x00
sd_data[33]	Πίνακας για τις μετρήσεις που περιέχονται σε ένα sector σε δεκαεξαδική μορφή	Όλα τα στοιχεία του πίνακα έχουν τιμή 0x00
cdata[33]	Πίνακας για τα δεδομένα του πίνακα sd_data[33] σε μορφή χαρακτήρων ASCII.	Όλα τα στοιχεία του πίνακα έχουν τιμή 0x00
k	Για έλεγχο της κατάστασης της διαδικασίας αρχικοποίησης της κάρτας μSD. Οι τιμές που παίρνει είναι: 0: Διαδικασία σε εξέλιξη 1: Η διαδικασία ολοκληρώθηκε	0
diafora[2]	Αποθηκεύεται η διαφορά των bytes των sectors 0x00,0x01. Στο πρώτο στοιχείο του πίνακα αποθηκεύεται η διαφορά του πρώτου byte των sectors και στο δεύτερο στοιχείο του πίνακα η διαφορά του δεύτερου	{0x00,0x00}
Diafora	Αποθηκεύεται το άθροισμα των στοιχείων του πίνακα diafora[2], ώστε να υπολογιστεί η συνολική διαφορά των sectors 0x00,0x01	0x00
o	Εάν ένας τομέας της κάρτας μνήμης είναι άγραφος γίνεται 1, αλλιώς παραμένει 0	0

x	Εάν σε ένα sector υπάρχει μέτρηση διαφορετικής ημερομηνίας γίνεται 1, αλλιώς παραμένει 0	0
l	Αποθηκεύεται η τιμή που αντιστοιχεί στο αρχείο που δημιουργείται στη μνήμη RAM του FPGA	0
A	Καθολική μεταβλητή για τον έλεγχο αποστολής ενός αρχείου μέσω του διακομιστή TFTP. 0: Αναμένεται αποστολή 1: Το αρχείο έχει σταλεί	0
endproc	Ελέγχει τη λειτουργία του αυτοματισμού δειγματοληψίας. 0: Η διαδικασία είναι σε εξέλιξη 1: Η διαδικασία έχει τελειώσει	0x00

Πίνακας Β.1: Μεταβλητές που χρησιμοποιούνται στην int main(void)

Στη συνέχεια παρατίθεται ο κύριος κώδικας του λογισμικού σε C του αρχείου main.c και κώδικες από τα αρχεία: encoder.c, init.c, spi_f.c, tftpserver.c, bootloader.c.

Κύριος κώδικας (main.c)

```
#include "xparameters.h"
#include "xil_printf.h"
#include "xiic_l.h"
#include "xuartlite_l.h"
#include "xgpio_l.h"
#include "xil_types.h"
#include "xio.h"
#include "netif/xadapter.h"
#include "xilmfs.h"
#include "platform.h"
#include "platform_config.h"
#define SPI_ADDR XPAR_XPS_SPI_0_BASEADDR
#define GPIO_0_ADDR XPAR_XPS_GPIO_0_BASEADDR
#define GPIO_1_ADDR XPAR_XPS_GPIO_1_BASEADDR
#define GPIO_6_ADDR XPAR_XPS_GPIO_6_BASEADDR
#define GPIO_7_ADDR XPAR_XPS_GPIO_7_BASEADDR
#define IIC_ADDR XPAR_XPS_IIC_0_BASEADDR
```

```
#define RTC_ADDR 0x68
#define GPS_ADDR XPAR_UARTLITE_2_BASEADDR
#define ADC_ADDR 0x21
//Global variable declaration
int A=0;
//Function Declaration
u32 spi_send_byte (u8 Byte);
/*****************************************************************
* Name: u32 spi_send_byte(u8 Byte)
* File: spi_f.c
* Description: Sends data through the spi bus
* Parameter: u8 Byte > Data to be sent
*****************************************************************/
u32 spi_read_byte(void);
/*****************************************************************
* Name: u32 spi_read_byte(void)
* File: spi_f.c
* Description: Receives data through the spi bus
*****************************************************************/
int sd_init(void);
/*****************************************************************
* Name: int sd_init(void)
* File: init.c
* Description: Initializes the uSD memory card
*****************************************************************/
int start_tftp_application();
/*****************************************************************
* Name: int start_tftp_application()
* File: tftpserver.c
* Description: Starts the TFTP Server
*****************************************************************/
int transfer_tftp_data();
/*****************************************************************
* Name int transfer_tftp_data()
* File: tftpserver.c
* Description: Allows data transmission from/to the tftp server
*****************************************************************/
void print_tftp_app_header();
/*****************************************************************
* Name: void print_tftp_app_header()
* File: tftpserver.c
* Description: Prints the tftp server ip address, port number etc.
```

```
*****************************************************************/
void lwip_raw_init();
/*****************************************************************
* Name void lwip_raw_init()
* Header file: lwip.h
* Description: Starts the lwip application
*****************************************************************/
static struct netif server_netif;
struct netif *netif;
void print_ip(char *msg, struct ip_addr *ip)
{
print(msg);
xil_printf("%d.%d.%d.%d\n\r", ip4_addr1(ip), ip4_addr2(ip),
ip4_addr3(ip), ip4_addr4(ip));
}
void
print_ip_settings(struct ip_addr *ip, struct ip_addr *mask, struct ip_addr *gw)
{
print_ip("Board IP: ", ip);
print_ip("Netmask : ", mask);
print_ip("Gateway : ", gw);
}
u8 SecondsEncoder(u8 LSB, u8 MSB);
u8 MinutesEncoder(u8 LSB, u8 MSB);
u8 HoursEncoder(u8 LSB, u8 MSB);
u8 DayEncoder(u8 LSB, u8 MSB);
u8 MonthEncoder(u8 LSB, u8 MSB);
u8 YearEncoder(u8 LSB, u8 MSB);
/*****************************************************************
* Name: u8 _________ Encoder(u8 LSB, u8 MSB)
* File: encoder.c
* Description: Transforms the data, coded in ASCII characters, that are received
*from the gps module to hexadecimal numbers
* Parameter: u8 LSB > Second ASCII character
* Parameter: u8 MSB > First ASCII character
*****************************************************************/
int main(void){
// Variables declaration
u8 SecondsReg=0x00;
u8 MinutesReg=0x01;
u8 HoursReg=0x02;
u8 ControlReg=0x0E;

u8 StatusReg=0x0F;
```

```
u8 Alarm2MinsSetup[2]={0x0B,0x00};
u8 Alarm2HoursSetup1[2]={0x0C,0x46};
u8 Alarm2HoursSetup2[2]={0x0C,0x72};
u8 Alarm2HoursSetup3[2]={0x0C,0x62};
u8 Alarm2HoursSetup4[2]={0x0C,0x52};
u8 Alarm2HoursSetup5[2]={0x0C,0x66};
u8 Alarm2DaySetup[2]={0x0D,0x80};
u8 CtrlRegSetup[2]={0x0E,0x06};
u8 StatusRegSetup[2]={0x0F,0x00};
u8 HourMSB;
u8 HourLSB;
u8 MinMSB;
u8 MinLSB;
u8 SecMSB;
u8 SecLSB;
u8 DayMSB;
u8 DayLSB;
u8 MonthMSB;
u8 MonthLSB;
u8 YearMSB;
u8 YearLSB;
u8 SecondsEnc;
u8 MinutesEnc;
u8 HoursEnc;
u8 DayEnc;
u8 MonthEnc;
u8 YearEnc;
u8 MinutesData;
u8 HoursData;
u8 yearData;
u8 monthData;
u8 dateData;
u8 SecondsSetup[2]={0x00,0x00};
u8 MinutesSetup[2]={0x01,0x00};
u8 HoursSetup[2]={0x02,0x00};
u8 DaySetup[2]={0x04,0x00};
u8 MonthSetup[2]={0x05,0x00};
u8 YearSetup[2]={0x06,0x00};
u8 DateReg=0x04;
u8 MonthReg=0x05;
u8 YearReg=0x06;
u8 endproc=0x00;
u8 GpsData[75];
```

```
u8
GpsCommand[51]={0x24,0x50,0x4D,0x54,0x4B,0x33,0x31,0x34,0x2C,0X30,0X2C,
0X31,0x2C,0X30,0x2C,0X30,0x2C,0X30,0x2C,0X30,0x2C,0X30,0x2C,0X30,0x2C,
0X30,0x2C,0X30,0x2C,0X30,0x2C,0X30,0x2C,0X30,0x2C,0X30,0x2C,0X30,0x2C,
0X30,0x2C,0X30,0x2C,0X30,0x2C,0X30,0X2A,0X32,0X39,0XD,0XA};
int i=0;

int o=0;
int x=0;
int z=0;
u8 ADCregT=0x80;
u8 ADCregC=0x90;
u16 Temp_data;
u16 Cond_data;
u8 Temp_dataM;
u8 Temp_dataL;
u8 Cond_dataM;
u8 Cond_dataL;
int k;
u8 data;
u8 sData[512];
int l;
int r;
char cdata[33];
u8 y=0x00;
u8 sector_data[3];
u8 sent_sector_data[3];
u8 diafora[2];
u8 Diafora;
u8 control_info[5]={0x00,0x00,0x00,0x00,0x00,0x00,0x00};
u8 sd_data[33];
//Code start
//Alarm2 programming (same code for all the hours)
XIic_Send(IIC_ADDR,RTC_ADDR,&CtrlRegSetup[i++], 2,XIIC_STOP);
/*****************************************************************
* Name: Xiic_Send(u32 BaseAddress, u8 Address, u8 *BufferPtr, unsigned Byte
* Count, u8 Option)
* Header file: xiic_l.h
* Description: Sends data through the IIC bus
* Parameter: u32 BaseAddress > Master's address
* Parameter: u8 Address > Slave's address
* Parameter: u8 *BufferPtr > Pointer to the variable that contains the data to be sent
* Parameter: unsigned ByteCount: Size in bytes of the data
* Parameter u8 Option: Controls the bus operation when the data transmission is done
```

```
**********************************************************************/
i=0;
XIic_Send(IIC_ADDR,RTC_ADDR,&Alarm2MinsSetup[i++], 2,XIIC_STOP);
i=0;
XIic_Send(IIC_ADDR,RTC_ADDR,&Alarm2DaySetup[i++], 2,XIIC_STOP);
i=0;

//Reading the hour register
for (i=0;i<10;i++){
XIic_Send(IIC_ADDR,RTC_ADDR,&HoursReg, 1,XIIC_STOP);
XIic_Recv(IIC_ADDR,RTC_ADDR,&HoursData, 1,XIIC_STOP);
/*********************************************************************
* Name: Xiic_Recv(u32 BaseAddress, u8 Address, u8 *BufferPtr, unsigned Byte
* Count, u8 Option)
* Header file: xiic_l.h
* Description: Receives data through the IIC bus
* Parameter: u32 BaseAddress > Master's address
* Parameter: u8 Address > Slave's address
* Parameter: u8 *BufferPtr > Pointer to the variable that stores the received data
* Parameter: unsigned ByteCount: Size in bytes of the data
* Parameter u8 Option: Controls the bus operation when the data transmission is done
**********************************************************************/
}
// Checking the hour data, in order to pick the right system procedure
if (HoursData==0x52) y=0x01;
if (HoursData==0x46) y=0x02;
if (HoursData==0x72) y=0x03;
if (HoursData==0x62) y=0x05;
if (HoursData==0x66) y=0x04;
if ((y==0x01||y==0x02)||(y==0x03||y==0x04)){
//Sampling automation start
XGpio_WriteReg (GPIO_6_ADDR, 0, 0x01);
/*********************************************************************
* Name: XGpio_WriteReg (u32 BaseAddress, u8 Regoffset, u8 Data)
* Header file: xgpio_l.h
* Description: Sends data through the gpio
* Parameter: u32 BaseAddresss > Address of the gpio peripheral
* Parameter: u8 Regoffset > Register offset from the peripheral's address
* Parameter: u8 Data > Data to be sent
**********************************************************************/
while (endproc!=0x01){
endproc=XGpio_ReadReg(GPIO_7_ADDR, 0);
/*********************************************************************
* Name: XGpio_ReadReg (u32 BaseAddress, u8 RegOffset)
```

```
* Header file: xgpio_l.h
* Description: Receives data through the gpio
100
* Parameter: u32 BaseAddresss > Address of the gpio peripheral
* Parameter: u8 Regoffset > Register offset from the peripheral's address
*****************************************************************/
}
// Measurement reception
// Temperature measurement
XIic_Send(IIC_ADDR,ADC_ADDR,&ADCregT,1,XIIC_STOP);
XIic_Recv(IIC_ADDR,ADC_ADDR,&Temp_data, 2,XIIC_STOP);
// Conductivity measurement
XIic_Send(IIC_ADDR,ADC_ADDR,&ADCregC, 1,XIIC_STOP);
XIic_Recv(IIC_ADDR,ADC_ADDR,&Cond_data, 2,XIIC_STOP);
// Only the last 12-bit of the measurement are importnant
Cond_data=Cond_data & 0x0fff;
Temp_dataM=Temp_data >>8;
Temp_dataL=Temp_data & 0x00FF;
Cond_dataM=Cond_data >>8;
Cond_dataL=Cond_data & 0x00FF;
// Data reception from the date registers
XIic_Send(IIC_ADDR,RTC_ADDR,&DateReg, 1,XIIC_STOP);
XIic_Recv(IIC_ADDR,RTC_ADDR,&dateData, 1,XIIC_STOP);
XIic_Send(IIC_ADDR,RTC_ADDR,&MonthReg, 1,XIIC_STOP);
XIic_Recv(IIC_ADDR,RTC_ADDR,&monthData, 1,XIIC_STOP);
XIic_Send(IIC_ADDR,RTC_ADDR,&YearReg, 1,XIIC_STOP);
XIic_Recv(IIC_ADDR,RTC_ADDR,&yearData, 1,XIIC_STOP);
// uSD memory card initialization
k=sd_init();
while(k!=1){
}
// Reading sector 0x00 to receive info about the last written sector
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);
for (i=0;i<512;i++){
```

```
sData[i]=spi_read_byte();
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
for (i=0;i<3;i++){
sector_data[i]=sData[i];
xil_printf("sec %d %x\n",i,sector_data[i]);
}
// Reading the last written sector to receive info about the measurement's date or if a measurement exists
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);
spi_send_byte(sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);
for (i=0;i<512;i++){
sData[i]=spi_read_byte();
}
for (i=0;i<5;i++){
control_info[i]=sData[i];\
xil_printf("ctrl %d %x\n",i,control_info[i]);
}
if (y==0x02){
control_info[5]=sData[9];
xil_printf("ctrl 9 %x\n",control_info[5]);
}
if (y==0x03){
control_info[6]=sData[17];
xil_printf("ctrl 17 %x\n",control_info[6]);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
// Sector write procedure if the sector is empty
if ((control_info[1]==0x00)&&(control_info[2]==0x00)&&(control_info[3]==0x00))
{

o=1;
```

```
XIo_Out32(SPI_ADDR +0x70, 0x00); // Writing the data
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);
spi_send_byte(sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
// xil_printf("aoa %x\n",data);
}
spi_send_byte(0xFE);
spi_send_byte(y);
spi_send_byte(dateData);
spi_send_byte(monthData);
spi_send_byte(yearData);
spi_send_byte(Temp_dataM);
spi_send_byte(Temp_dataL);
spi_send_byte(Cond_dataM);
spi_send_byte(Cond_dataL);
for (i=0;i<504;i++){
spi_send_byte(0xFF);
}
while (data != 0x03){
data=spi_read_byte();
xil_printf("+1agr %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
}
// Sector write procedure if the sector contains a measurement from a different date
if (((control_info[1]==y) || (control_info[2]!=dateData) ||
(control_info[3]!=monthData) || (control_info[4]!=yearData)) && (o!=1)){
//Sector increment
if (sector_data[1]==0xff) {
sector_data[2]=sector_data[2] + 0x01;
sector_data[1]=0x00;}

sector_data[1]=sector_data[1] + 0x01;
x=1;
```

```
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);
spi_send_byte(sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
// xil_printf("aoa %x\n",data);
}
spi_send_byte(0xFE);
spi_send_byte(y);
spi_send_byte(dateData);
spi_send_byte(monthData);
spi_send_byte(yearData);
spi_send_byte(Temp_dataM);
spi_send_byte(Temp_dataL);
spi_send_byte(Cond_dataM);
spi_send_byte(Cond_dataL);
for (i=0;i<504;i++){
spi_send_byte(0xFF);
}
while (data != 0x03){
data=spi_read_byte();
xil_printf("+1d %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);

spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
```

```
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
// xil_printf("aoa %x\n",data);
}
spi_send_byte(0xFE);
spi_send_byte(sector_data[1]);
spi_send_byte(sector_data[2]);
while (data != 0x03){
data=spi_read_byte();
xil_printf("+1s %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
}
// Sector write procedure if the sector contains a measurement of the same date
if((control_info[1]!=y)&&(control_info[2]==dateData)&&
(control_info[3]==monthData)&&(x!=1)) {
if (y==0x02){
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);
spi_send_byte(sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);
for (i=0;i<512;i++){
sData[i]=spi_read_byte();
}
for (i=1;i<9;i++){
sd_data[i]=sData[i];
}
sd_data[9]=y;
sd_data[10]=dateData;
sd_data[11]=monthData;

sd_data[12]=yearData;
sd_data[13]=Temp_dataM;
sd_data[14]=Temp_dataL;
```

```
sd_data[15]=Cond_dataM;
sd_data[16]=Cond_dataL;
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);
spi_send_byte(sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
// xil_printf("aoa %x\n",data);
}
spi_send_byte(0xFE);
for (i=1;i<17;i++){
spi_send_byte(sd_data[i]);
}
while (data != 0x03){
data=spi_read_byte();
xil_printf("y=2 %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
}
if (y==0x03){
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);

spi_send_byte(sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);
```

```
for (i=0;i<512;i++){
sData[i]=spi_read_byte();
}
for (i=1;i<17;i++){
sd_data[i]=sData[i];
}
sd_data[17]=y;
sd_data[18]=dateData;
sd_data[19]=monthData;
sd_data[20]=yearData;
sd_data[21]=Temp_dataM;
sd_data[22]=Temp_dataL;
sd_data[23]=Cond_dataM;
sd_data[24]=Cond_dataL;
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);
spi_send_byte(sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
// xil_printf("aoa %x\n",data);
}
spi_send_byte(0xFE);
for (i=1;i<25;i++){
spi_send_byte(sd_data[i]);
}
while (data != 0x03){
data=spi_read_byte();
xil_printf("y=3 %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);

spi_send_byte(0xff);
}
if (y==0x04){
XIo_Out32(SPI_ADDR +0x70, 0x00);
```

```
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);
spi_send_byte(sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);
for (i=0;i<512;i++){
sData[i]=spi_read_byte();
}
for (i=1;i<25;i++){
sd_data[i]=sData[i];
}
sd_data[25]=y;
sd_data[26]=dateData;
sd_data[27]=monthData;
sd_data[28]=yearData;
sd_data[29]=Temp_dataM;
sd_data[30]=Temp_dataL;
sd_data[31]=Cond_dataM;
sd_data[32]=Cond_dataL;
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);
spi_send_byte(sector_data[1]);
spi_send_byte(0x00);

data=spi_read_byte();
while (data !=0x00){
// xil_printf("aoa %x\n",data);
}
spi_send_byte(0xFE);
```

```
for (i=1;i<33;i++){
spi_send_byte(sd_data[i]);
}
while (data != 0x03){
data=spi_read_byte();
xil_printf("y=3 %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
}
}
// TFTP Server start
if (y==0x04){
XIo_Out32(SPI_ADDR +0x70, 0x00); // Reading data from the last written sector
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);
for (i=0;i<512;i++){
sData[i]=spi_read_byte();
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
for (i=0;i<3;i++){
sector_data[i]=sData[i];
xil_printf("sec %d %x\n",i,sector_data[i]);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
XIo_Out32(SPI_ADDR +0x70, 0x00); // Reading data from the last sent sector
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);

spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
```

```
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x01);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);
for (i=0;i<512;i++){
sData[i]=spi_read_byte();
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
for (i=0;i<3;i++){
sent_sector_data[i]=sData[i];
xil_printf("sector %d %x\n",i,sent_sector_data[i]);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
//Comparing the last written sector to the last sent sector
diafora[1]=sector_data[2]-sent_sector_data[2];
diafora[0]=sector_data[1]-sent_sector_data[1];
Diafora=diafora[0]+diafora[1];
if (Diafora<=0x01){
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sector_data[2]);
spi_send_byte(sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);
for (i=0;i<512;i++){
sData[i]=spi_read_byte();

}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
for (i=1;i<33;i++){
cdata[i]=sData[i];
```

```
xil_printf("%d %x\n",i,cdata[i]);
}
//file creation
mfs_init_fs(100000,0x45000000, MFSINIT_NEW);
mfs_create_dir("dir1");
l=mfs_file_open("file1.txt", 3);
mfs_file_write(l,cdata,33);
mfs_file_close(l);
}
if (Diafora>0x01){
if (sent_sector_data[1]==0xff) {
sent_sector_data[2]=sent_sector_data[2] + 0x01;
sent_sector_data[1]=0x00;}
sent_sector_data[1]=sent_sector_data[1] + 0x01;
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sent_sector_data[2]);
spi_send_byte(sent_sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);
for (i=0;i<512;i++){
sData[i]=spi_read_byte();
xil_printf("%d %x\n",i,sData[i]);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
for (i=1;i<33;i++){
cdata[i]=sData[i];
xil_printf("%d %x\n",i,cdata[i]);
}

mfs_init_fs(100000,0x45000000, MFSINIT_NEW);
/*****************************************************************
* Name: mfs_init_fs (int numbytes, char* address, int init_type)
* Header file: xilmfs.h
* Description: Creates a file system at the FPGA's RAM
```

```
* Parameter: int numbytes > Size of the file system
* Parameter: char* address > The file system's location at the FPGA's RAM
* Parameter: int init_type: Type of the file system
*****************************************************************/
mfs_create_dir("dir1");
/*****************************************************************
* Name: mfs_create_dir (char* newdir)
* Header file: xilmfs.h
* Description: Creates a new directory
* Parameter: char* newdir > Name of the new directory
*****************************************************************/
l=mfs_file_open("file1.txt", 3);
/*****************************************************************
Name: mfs_file_open (const char* filename, int mode)
* Header file: xilmfs.h
* Description: Creates a new file or opens an existing file
* Parameter: const char* filename> Name of the new directory
* Parameter: int mode > Sets the function's mode
*****************************************************************/
mfs_file_write(l,cdata,33);
/*****************************************************************
* Name: mfs_file_write (int fd, const char* buf, int buflen)
* Header file: xilmfs.h
* Description: Writes data to a file
* Parameter: int fd > The file's number at the FPGA's RAM
* Parameter: const char* buf > Data to be written
* Parameter: int buflen > The data size in bytes
*****************************************************************/
mfs_file_close(l);
}
struct ip_addr ipaddr, netmask, gw;
/* the mac address of the board. this should be unique per board */
unsigned char mac_ethernet_address[] =
{ 0x00, 0x0a, 0x35, 0x00, 0x01, 0x02 };
netif = &server_netif;
init_platform();

/* initliaze IP addresses to be used */
IP4_ADDR(&ipaddr, 192, 168, 1, 10);
IP4_ADDR(&netmask, 255, 255, 255, 0);
IP4_ADDR(&gw, 192, 168, 1, 1);
//print_app_header();
print_ip_settings(&ipaddr, &netmask, &gw);
lwip_raw_init();
```

```
if (!xemac_add(netif, &ipaddr, &netmask,
&gw, mac_ethernet_address,
PLATFORM_EMAC_BASEADDR)) {
xil_printf("Error adding N/W interface\n\r");
return -1;
}
netif_set_default(netif);
/* Create a new DHCP client for this interface.
* Note: you must call dhcp_fine_tmr() and dhcp_coarse_tmr() at
* the predefined regular intervals after starting the client.
*/
/* dhcp_start(echo_netif); */
/* now enable interrupts */
platform_enable_interrupts();
/* specify that the network if is up */
netif_set_up(netif);
print_tftp_app_header();
/* start the application (web server, rxtest, txtest, etc..) */
start_tftp_application();
while (MinutesData<= 0x10) {
XIic_Send(IIC_ADDR,RTC_ADDR,&MinutesReg,1,XIIC_STOP);
XIic_Recv(IIC_ADDR,RTC_ADDR,&MinutesData,1,XIIC_STOP);
while(Diafora!=0x00){
xemacif_input(netif);
transfer_tftp_data();
if (A==1){

Diafora=Diafora-0x01;
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x01);
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
// xil_printf("aoa %x\n",data);
}
spi_send_byte(0xFE);
```

```
spi_send_byte(sent_sector_data[1]);
spi_send_byte(sent_sector_data[2]);
while (data != 0x03){
data=spi_read_byte();
xil_printf("+1ssector %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
if (Diafora==0x00){
A=0;
}
if (diafora>=0x01){
if (sent_sector_data[1]==0xff) {
sent_sector_data[2]=sent_sector_data[2] + 0x01;
sent_sector_data[1]=0x00;}
sent_sector_data[1]=sent_sector_data[1] + 0x01;
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0x51);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(sent_sector_data[2]);
spi_send_byte(sent_sector_data[1]);
spi_send_byte(0x00);
data=spi_read_byte();
xil_printf("sd %x\n",data);

for (i=0;i<512;i++){
sData[i]=spi_read_byte();
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
for (i=1;i<33;i++){
cdata[i]=sData[i];
xil_printf("%d %x\n",i,cdata[i]);
}
mfs_init_fs(100000,0x45000000, MFSINIT_NEW);
mfs_create_dir("dir1");
l=mfs_file_open("file1.txt", 3);
mfs_file_write(l,cdata,33);
mfs_file_close(l);
A=0;
}
}
```

```
}
}
cleanup_platform();}
// Alarm 2 programming
i=0;
if    (y==0x01){    XIic_Send(IIC_ADDR,RTC_ADDR,&Alarm2HoursSetup1[i++],
2,XIIC_STOP);}
if    (y==0x02)    {XIic_Send(IIC_ADDR,RTC_ADDR,&Alarm2HoursSetup2[i++],
2,XIIC_STOP); }
if    (y==0x03)    {XIic_Send(IIC_ADDR,RTC_ADDR,&Alarm2HoursSetup3[i++],
2,XIIC_STOP);}
if    (y==0x04)    {XIic_Send(IIC_ADDR,RTC_ADDR,&Alarm2HoursSetup4[i++],
2,XIIC_STOP);}
i=0;
//Unflagging the Alarm2 bit in the status register in order to disable the system
XIic_Send(IIC_ADDR,RTC_ADDR,&StatusRegSetup[i++], 2,XIIC_STOP);
}
// Synchronization procedure
if (y==0x05){
XGpio_WriteReg(GPIO_0_ADDR,0,0x01);
while(x<7){
GpsData[i]=XUartLite_RecvByte(GPS_ADDR);

/*****************************************************************
* Name: XUartLite_RecvByte (u32 BaseAddress)
* Header file: xuartlite.h
* Description: Receives data through the Uart bus
* Parameter: u32 BaseAddress > Address of the uartlite peripheral
*****************************************************************/
if (GpsData[i]==0x24){
x++;
}
if (x==6){
for (i=0;i<51;i++){
XUartLite_SendByte(GPS_ADDR,GpsCommand[i]);
/*****************************************************************
* Name: XUartLite_SendByte (u32 BaseAddress, u8 Data)
* Header file: xuartlite.h
* Description: Sends data through the Uart bus
* Parameter: u32 BaseAddress > Address of the uartlite peripheral
* Parameter: u8 Data > Data to be sent
*****************************************************************/
}
i=0;
```

```
}
}
r=0;
while(r<11){
GpsData[i]=XUartLite_RecvByte(GPS_ADDR);
if (GpsData[i]==0x24){
i=0;
}
if (GpsData[18]==0x41){
HourMSB=GpsData[7];
HourLSB=GpsData[8];
MinMSB=GpsData[9];
MinLSB=GpsData[10];
SecMSB=GpsData[11];
SecLSB=GpsData[12];
SecondsEnc = SecondsEncoder (SecLSB, SecMSB);
SecondsSetup[1]=SecondsEnc;
MinutesEnc = MinutesEncoder (MinLSB, MinMSB);
MinutesSetup[1]=MinutesEnc;
HoursEnc = HoursEncoder (HourLSB, HourMSB);
HoursSetup[1]=HoursEnc;
xil_printf("\n");

r++;}
if (GpsData[i]==0x2a){
DayMSB=GpsData[i-10];
DayLSB=GpsData[i-9];
MonthMSB=GpsData[i-8];
MonthLSB=GpsData[i-7];
YearMSB=GpsData[i-6];
YearLSB=GpsData[i-5];
xil_printf("\n");
DayEnc=DayEncoder (DayLSB, DayMSB);
DaySetup[1]=DayEnc;
MonthEnc=MonthEncoder(MonthLSB, MonthMSB);
MonthSetup[1]=MonthEnc;
YearEnc= YearEncoder(YearLSB, YearMSB);
YearSetup[1]=YearEnc;
}
i++;
}
// Programming of the date and time registers
z=0;
XIic_Send(IIC_ADDR,RTC_ADDR,&SecondsSetup[z++], 2,XIIC_STOP);
```

```
z=0;
XIic_Send(IIC_ADDR,RTC_ADDR,&MinutesSetup[z++], 2,XIIC_STOP);
z=0;
XIic_Send(IIC_ADDR,RTC_ADDR,&HoursSetup[z++], 2,XIIC_STOP);
z=0;
XIic_Send(IIC_ADDR,RTC_ADDR,&DaySetup[z++], 2,XIIC_STOP);
z=0;
XIic_Send(IIC_ADDR,RTC_ADDR,&MonthSetup[z++], 2,XIIC_STOP);
z=0;
XIic_Send(IIC_ADDR,RTC_ADDR,&YearSetup[z++], 2,XIIC_STOP);
z=0;
// Alarm 2 programming
if(y==0x05)XIic_Send(IIC_ADDR,RTC_ADDR,&Alarm2HoursSetup5[i++],
2,XIIC_STOP);
i=0;
// System deactivation
XIic_Send(IIC_ADDR,RTC_ADDR,&StatusRegSetup[i++], 2,XIIC_STOP);
i=0;
}

return 0;
}
```

Κώδικας για κωδικοποίηση ASCII σε HEX (encoder.c)

```
#include "xil_types.h"

// Seconds encoding function
u8 SecondsEncoder (u8 SecLSB ,u8 SecMSB ){
u8 Seconds;
if (SecMSB==0x30){
if (SecLSB==0x30) Seconds=0x00;
if (SecLSB==0x31) Seconds=0x01;
if (SecLSB==0x32) Seconds=0x02;
if (SecLSB==0x33) Seconds=0x03;
if (SecLSB==0x34) Seconds=0x04;
if (SecLSB==0x35) Seconds=0x05;
if (SecLSB==0x36) Seconds=0x06;
if (SecLSB==0x37) Seconds=0x07;
if (SecLSB==0x38) Seconds=0x08;
if (SecLSB==0x39) Seconds=0x09;
}
if (SecMSB==0x31){
```

```
if (SecLSB==0x30) Seconds=0x10;
if (SecLSB==0x31) Seconds=0x11;
if (SecLSB==0x32) Seconds=0x12;
if (SecLSB==0x33) Seconds=0x13;
if (SecLSB==0x34) Seconds=0x14;
if (SecLSB==0x35) Seconds=0x15;
if (SecLSB==0x36) Seconds=0x16;
if (SecLSB==0x37) Seconds=0x17;
if (SecLSB==0x38) Seconds=0x18;
if (SecLSB==0x39) Seconds=0x19;
}
if (SecMSB==0x32){
if (SecLSB==0x30) Seconds=0x20;
if (SecLSB==0x31) Seconds=0x21;
if (SecLSB==0x32) Seconds=0x22;
if (SecLSB==0x33) Seconds=0x23;
if (SecLSB==0x34) Seconds=0x24;
if (SecLSB==0x35) Seconds=0x25;
if (SecLSB==0x36) Seconds=0x26;
if (SecLSB==0x37) Seconds=0x27;
if (SecLSB==0x38) Seconds=0x28;
if (SecLSB==0x39) Seconds=0x29;
}
if (SecMSB==0x33){
if (SecLSB==0x30) Seconds=0x30;
if (SecLSB==0x31) Seconds=0x31;
if (SecLSB==0x32) Seconds=0x32;
if (SecLSB==0x33) Seconds=0x33;
if (SecLSB==0x34) Seconds=0x34;
if (SecLSB==0x35) Seconds=0x35;
if (SecLSB==0x36) Seconds=0x36;
if (SecLSB==0x37) Seconds=0x37;
if (SecLSB==0x38) Seconds=0x38;
if (SecLSB==0x39) Seconds=0x39;
}
if (SecMSB==0x34){
if (SecLSB==0x30) Seconds=0x40;
if (SecLSB==0x31) Seconds=0x41;
if (SecLSB==0x32) Seconds=0x42;
if (SecLSB==0x33) Seconds=0x43;
if (SecLSB==0x34) Seconds=0x44;
if (SecLSB==0x35) Seconds=0x45;
if (SecLSB==0x36) Seconds=0x46;
if (SecLSB==0x37) Seconds=0x47;
```

```
if (SecLSB==0x38) Seconds=0x48;
if (SecLSB==0x39) Seconds=0x49;
}
if (SecMSB==0x35){
if (SecLSB==0x30) Seconds=0x50;
if (SecLSB==0x31) Seconds=0x51;
if (SecLSB==0x32) Seconds=0x52;
if (SecLSB==0x33) Seconds=0x53;
if (SecLSB==0x34) Seconds=0x54;
if (SecLSB==0x35) Seconds=0x55;
if (SecLSB==0x36) Seconds=0x56;
if (SecLSB==0x37) Seconds=0x57;
if (SecLSB==0x38) Seconds=0x58;
if (SecLSB==0x39) Seconds=0x59;
}
return (u8) Seconds;
}

// Minutes encoding function
u8 MinutesEncoder (u8 MinLSB, u8 MinMSB){
u8 Minutes;
if (MinMSB==0x30){
if (MinLSB==0x30) Minutes=0x00;
if (MinLSB==0x31) Minutes=0x01;
if (MinLSB==0x32) Minutes=0x02;
if (MinLSB==0x33) Minutes=0x03;
if (MinLSB==0x34) Minutes=0x04;
if (MinLSB==0x35) Minutes=0x05;
if (MinLSB==0x36) Minutes=0x06;
if (MinLSB==0x37) Minutes=0x07;
if (MinLSB==0x38) Minutes=0x08;
if (MinLSB==0x39) Minutes=0x09;
}
if (MinMSB==0x31){
if (MinLSB==0x30) Minutes=0x10;
if (MinLSB==0x31) Minutes=0x11;
if (MinLSB==0x32) Minutes=0x12;
if (MinLSB==0x33) Minutes=0x13;
if (MinLSB==0x34) Minutes=0x14;
if (MinLSB==0x35) Minutes=0x15;
if (MinLSB==0x36) Minutes=0x16;
if (MinLSB==0x37) Minutes=0x17;
if (MinLSB==0x38) Minutes=0x18;
if (MinLSB==0x39) Minutes=0x19;
```

```
}
if (MinMSB==0x32){
if (MinLSB==0x30) Minutes=0x20;
if (MinLSB==0x31) Minutes=0x21;
if (MinLSB==0x32) Minutes=0x22;
if (MinLSB==0x33) Minutes=0x23;
if (MinLSB==0x34) Minutes=0x24;
if (MinLSB==0x35) Minutes=0x25;
if (MinLSB==0x36) Minutes=0x26;
if (MinLSB==0x37) Minutes=0x27;
if (MinLSB==0x38) Minutes=0x28;
if (MinLSB==0x39) Minutes=0x29;
}
if (MinMSB==0x33){
if (MinLSB==0x30) Minutes=0x30;
if (MinLSB==0x31) Minutes=0x31;
if (MinLSB==0x32) Minutes=0x32;
if (MinLSB==0x33) Minutes=0x33;
if (MinLSB==0x34) Minutes=0x34;
if (MinLSB==0x35) Minutes=0x35;
if (MinLSB==0x36) Minutes=0x36;
if (MinLSB==0x37) Minutes=0x37;
if (MinLSB==0x38) Minutes=0x38;
if (MinLSB==0x39) Minutes=0x39;
}
if (MinMSB==0x34){
if (MinLSB==0x30) Minutes=0x40;
if (MinLSB==0x31) Minutes=0x41;
if (MinLSB==0x32) Minutes=0x42;
if (MinLSB==0x33) Minutes=0x43;
if (MinLSB==0x34) Minutes=0x44;
if (MinLSB==0x35) Minutes=0x45;
if (MinLSB==0x36) Minutes=0x46;
if (MinLSB==0x37) Minutes=0x47;
if (MinLSB==0x38) Minutes=0x48;
if (MinLSB==0x39) Minutes=0x49;
}
if (MinMSB==0x35){
if (MinLSB==0x30) Minutes=0x50;
if (MinLSB==0x31) Minutes=0x51;
if (MinLSB==0x32) Minutes=0x52;
if (MinLSB==0x33) Minutes=0x53;
if (MinLSB==0x34) Minutes=0x54;
if (MinLSB==0x35) Minutes=0x55;
```

```
if (MinLSB==0x36) Minutes=0x56;
if (MinLSB==0x37) Minutes=0x57;
if (MinLSB==0x38) Minutes=0x58;
if (MinLSB==0x39) Minutes=0x59;
}
return (u8) Minutes;
}

// Hour encoding function
u8 HoursEncoder (u8 hourLSB, u8 HourMSB){
u8 Hours;
if (HourMSB==0x30){
if (hourLSB==0x30) Hours=0x43;
if (hourLSB==0x31) Hours=0x44;
if (hourLSB==0x32) Hours=0x45;
if (hourLSB==0x33) Hours=0x46;
if (hourLSB==0x34) Hours=0x47;
if (hourLSB==0x35) Hours=0x48;
if (hourLSB==0x36) Hours=0x49;
if (hourLSB==0x37) Hours=0x50;
if (hourLSB==0x38) Hours=0x51;
if (hourLSB==0x39) Hours=0x72;
}
if (HourMSB==0x31){
if (hourLSB==0x30) Hours=0x61;
if (hourLSB==0x31) Hours=0x62;
if (hourLSB==0x32) Hours=0x63;
if (hourLSB==0x33) Hours=0x64;
if (hourLSB==0x34) Hours=0x65;
if (hourLSB==0x35) Hours=0x66;
if (hourLSB==0x36) Hours=0x67;
if (hourLSB==0x37) Hours=0x68;
if (hourLSB==0x38) Hours=0x69;
if (hourLSB==0x39) Hours=0x70;
}
if (HourMSB==0x32){
if (hourLSB==0x30) Hours=0x71;
if (hourLSB==0x31) Hours=0x52;
if (hourLSB==0x32) Hours=0x41;
if (hourLSB==0x33) Hours=0x42;
}
return (u8) Hours;
}
```

```
// Day encoding function
u8 DayEncoder (u8 dayLSB, u8 dayMSB){
u8 day;
if (dayMSB==0x30){
if(dayLSB==0x31) day=0x01;
if(dayLSB==0x32) day=0x02;
if(dayLSB==0x33) day=0x03;
if(dayLSB==0x34) day=0x04;
if(dayLSB==0x35) day=0x05;
if(dayLSB==0x36) day=0x06;
if(dayLSB==0x37) day=0x07;
if(dayLSB==0x38) day=0x08;
if(dayLSB==0x39) day=0x09;
}
if (dayMSB==0x31){
if(dayLSB==0x30) day=0x10;
if(dayLSB==0x31) day=0x11;
if(dayLSB==0x32) day=0x12;
if(dayLSB==0x33) day=0x13;
if(dayLSB==0x34) day=0x14;
if(dayLSB==0x35) day=0x15;
if(dayLSB==0x36) day=0x16;
if(dayLSB==0x37) day=0x17;
if(dayLSB==0x38) day=0x18;
if(dayLSB==0x39) day=0x19;
}
if (dayMSB==0x32){
if(dayLSB==0x30) day=0x20;
if(dayLSB==0x31) day=0x21;
if(dayLSB==0x32) day=0x22;
if(dayLSB==0x33) day=0x23;
if(dayLSB==0x34) day=0x24;
if(dayLSB==0x35) day=0x25;
if(dayLSB==0x36) day=0x26;
if(dayLSB==0x37) day=0x27;
if(dayLSB==0x38) day=0x28;
if(dayLSB==0x39) day=0x29;
}
if (dayMSB==0x33){
if(dayLSB==0x30) day=0x30;
if(dayLSB==0x31) day=0x31;
}
return (u8)day;
}
```

```
// Month encoding function
u8 MonthEncoder (u8 monthLSB, u8 monthMSB){
u8 month;
if (monthMSB==0x30){
if (monthLSB==0x31) month=0x01;
if (monthLSB==0x32) month=0x02;
if (monthLSB==0x33) month=0x03;
if (monthLSB==0x34) month=0x04;
if (monthLSB==0x35) month=0x05;
if (monthLSB==0x36) month=0x06;
if (monthLSB==0x37) month=0x07;
if (monthLSB==0x38) month=0x08;
if (monthLSB==0x39) month=0x09;
}
if (monthMSB==0x31){
if (monthLSB==0x30) month=0x10;
if (monthLSB==0x31) month=0x11;
if (monthLSB==0x32) month=0x12;
}
return (u8)month;
}

// Year encoding function
u8 YearEncoder (u8 yearLSB, u8 yearMSB){
u8 year;
if (yearMSB==0x31){
if (yearLSB==0x38) year=0x18;
if (yearLSB==0x39) year=0x19;
}
if (yearMSB==0x32){
if (yearLSB==0x30) year=0x20;
if (yearLSB==0x31) year=0x21;
if (yearLSB==0x32) year=0x22;
if (yearLSB==0x33) year=0x23;
if (yearLSB==0x34) year=0x24;
if (yearLSB==0x35) year=0x25;
if (yearLSB==0x36) year=0x26;
if (yearLSB==0x37) year=0x27;
if (yearLSB==0x38) year=0x28;
if (yearLSB==0x39) year=0x29;
}
if (yearMSB==0x33){
if (yearLSB==0x30) year=0x30;
if (yearLSB==0x31) year=0x31;
```

```
if (yearLSB==0x32) year=0x32;
if (yearLSB==0x33) year=0x33;
if (yearLSB==0x34) year=0x34;
if (yearLSB==0x35) year=0x35;
if (yearLSB==0x36) year=0x36;
if (yearLSB==0x37) year=0x37;
if (yearLSB==0x38) year=0x38;
if (yearLSB==0x39) year=0x39;
}
return (u8) year;
}
```

Κώδικας εντολών πρωτοκόλλου SPI (spi_f.c)

```
#include "xil_types.h"
#include "xparameters.h"
#include "xil_printf.h"
#include "xio.h"

#define SPI_ADDR XPAR_XPS_SPI_0_BASEADDR
#define SPI_Status_Mask_Empty 0x01

u32 spi_send_byte (u32 Byte){
u32 ReadRx;
u32 Return;

XIo_Out32(SPI_ADDR+0x68,(u32)Byte);
ReadRx=XIo_In32(SPI_ADDR +0x64);
while((ReadRx & SPI_Status_Mask_Empty)==SPI_Status_Mask_Empty){
        ReadRx=XIo_In32(SPI_ADDR +0x64);
}
Return=XIo_In32(SPI_ADDR +0x6C);
return (u32)Return;
}

u32 spi_read_byte(void){
u32 Return;
int i;

for (i=0;i<=8;i++){
        Return=spi_send_byte(0xFF);
                if (Return !=0xFF){
                        return Return;
```

```
                    }
}
i=0;
return Return;
}
```

Κώδικας υλοποίησης διακομιστή TFTP (tftpserver.c)

```
#include <stdio.h>
#include <string.h>
#include "lwip/inet.h"
#include "lwip/err.h"
#include "lwip/udp.h"
#include "xilmfs.h"
#include "tftpserver.h"
#include "tftputils.h"
#include "prot_malloc.h"
#include "platform_fs.h"

/* tftp_errorcode error strings */
char *tftp_errorcode_string[] = {
"not defined",
"file not found",
"access violation",
"disk full",
"illegal operation",
"unknown transfer id",
"file already exists",
"no such user",
};

int A=0;
err_t tftp_send_message(struct udp_pcb *upcb, struct ip_addr *to_ip, int to_port, char
*buf, int buflen)
{
err_t err;
struct pbuf *p;
/* form a pbuf */
p = pbuf_alloc(PBUF_TRANSPORT, buflen, PBUF_POOL);
if (!p) {
xil_printf("error allocating pbuf\n\r");
return ERR_MEM;
}
```

```
memcpy(p->payload, buf, buflen);
/* send message */
err = udp_sendto(upcb, p, to_ip, to_port);
/* free pbuf */
pbuf_free(p);
return err;
}

/* construct an error message into buf using err as the error code */
int tftp_construct_error_message(char *buf, tftp_errorcode err)
{
int errorlen;
tftp_set_opcode(buf, TFTP_ERROR);
tftp_set_errorcode(buf, err);
tftp_set_errormsg(buf, tftp_errorcode_string[err]);
errorlen = strlen(tftp_errorcode_string[err]);
/* return message size */
return 4 + errorlen + 1;
}

/* construct and send an error message back to client */
int tftp_send_error_message(struct udp_pcb *upcb, struct ip_addr *to, int to_port,
tftp_errorcode err)
{
char buf[512];
int n;
n = tftp_construct_error_message(buf, err);
return tftp_send_message(upcb, to, to_port, buf, n);
}

/* construct and send a data packet */
int tftp_send_data_packet(struct udp_pcb *upcb, struct ip_addr *to, int to_port, int
block, char *buf, int buflen)
{
char packet[TFTP_MAX_MSG_LEN];
tftp_set_opcode(packet, TFTP_DATA);
tftp_set_block(packet, block);
tftp_set_data_message(packet, buf, buflen);
return tftp_send_message(upcb, to, to_port, packet, buflen + 4);
}

int tftp_send_ack_packet(struct udp_pcb *upcb, struct ip_addr *to, int to_port, int
block)
{
```

```
char packet[TFTP_MAX_ACK_LEN];
tftp_set_opcode(packet, TFTP_ACK);
tftp_set_block(packet, block);
return tftp_send_message(upcb, to, to_port, packet, TFTP_MAX_ACK_LEN);
}


void tftp_cleanup(struct udp_pcb *upcb, tftp_connection_args *args)
{
/* cleanup the args */
mfs_file_close(args->fd);
free(args);
/* close the connection */
udp_remove(upcb);
}

void tftp_send_next_block(struct udp_pcb *upcb, tftp_connection_args *args,
struct ip_addr *to_ip, u16_t to_port)
{
        args->data_len = mfs_file_read(args->fd, args->data,
TFTP_DATA_PACKET_MSG_LEN);
if (args->data_len <= 0) {
xil_printf("closing connection, ret = %d\n\r", args->data_len);
/* we are done */
return tftp_cleanup(upcb, args);
}
/* send the data */
tftp_send_data_packet(upcb, to_ip, to_port,
args->block, args->data, args->data_len);
}

void rrq_recv_callback(void *_args, struct udp_pcb *upcb,
                       struct pbuf *p, struct ip_addr *addr, u16_t port)
{
tftp_connection_args *args = (tftp_connection_args *)_args;
if (tftp_is_correct_ack(p->payload, args->block)) {
/* increment block # */
 ((tftp_connection_args *)args)->block++;
}
else {
/* we did not receive the expected ACK, so
   do not update block #, thereby resending current block */
xil_printf("incorrect ack received, resending last block\n\r");
}
```

```
/* if the last read returned less than the requested number of bytes,
* then we've sent the whole file and so we can quit
*/
if (args->data_len < TFTP_DATA_PACKET_MSG_LEN)
return tftp_cleanup(upcb, args);



tftp_send_next_block(upcb, args, addr, port);
}

int tftp_process_read(struct udp_pcb *upcb, struct ip_addr *to, int to_port, char *fname)
{
int fd;
tftp_connection_args *args;
/* first make sure the file exists and we can read the file */
if (mfs_exists_file(fname) != 1) {
xil_printf("unable to open file: %s\n\r", fname);
tftp_send_error_message(upcb, to, to_port, TFTP_ERR_FILE_NOT_FOUND);
udp_remove(upcb); // close the connection - added by jwdonal to fix Xilinx version memory leak
return -1;
}
fd = mfs_file_open(fname, MFS_MODE_READ);
/* this function is called from a callback => interrupts are disabled
* => we can use regular malloc
*/
args = mem_malloc(sizeof *args);
xil_printf("aso %d\n");
if (!args) {
xil_printf("unable to allocate memory for tftp args\n\r");
tftp_send_error_message(upcb,to,to_port, TFTP_ERR_FILE_NOT_FOUND);
udp_remove(upcb); // close the connection - added by jwdonal to fix Xilinx version memory leak
return -1;
}

args->op = TFTP_RRQ;
args->to_ip.addr = to->addr;
args->to_port = to_port;
args->fd = fd;

/* set callback for receives on this UDP PCB (Protocol Control Block) */
```

```
udp_recv(upcb, rrq_recv_callback, args);
/* initiate the transaction by sending the first block of data
* further blocks will be sent when ACKs are received
*         - the receive callbacks need to get the proper state
*/
args->block = 1;
tftp_send_next_block(upcb, args, to, to_port);

return 0;
}

/* write callback */
void wrq_recv_callback(void *_args, struct udp_pcb *upcb,
                       struct pbuf *p, struct ip_addr *addr, u16_t port)
{
tftp_connection_args *args = (tftp_connection_args *)_args;
if (p->len != p->tot_len) {
xil_printf("ERROR: tftp server does not support chained pbuf's\n\r");
pbuf_free(p);
return;
}

/* make sure data block is what we expect */
if ((p->len > 4) && (tftp_extract_block(p->payload) == (args->block + 1))) {
/* write the received data to the file */
int n = mfs_file_write(args->fd, p->payload+TFTP_DATA_PACKET_HDR_LEN,
                                  p->len-TFTP_DATA_PACKET_HDR_LEN);
if (n != 1) {
xil_printf("write to file error\n\r");
tftp_send_error_message(upcb, addr, port, TFTP_ERR_FILE_NOT_FOUND);
udp_remove(upcb);
pbuf_free(p);
return;
}
args->block++;
}

tftp_send_ack_packet(upcb, addr, port, args->block);
/* if the last read returned less than the requested number of bytes,
* then we've sent the whole file and so we can quit
*/
if (p->len < TFTP_DATA_PACKET_MSG_LEN)
return tftp_cleanup(upcb, args);
}
```

```
/* write data coming via sd to file *fname */
int tftp_process_write(struct udp_pcb *upcb, struct ip_addr *to, int to_port, char
*fname)
{
int fd;
tftp_connection_args *args;
/* we do not allow overwriting files */
if (mfs_exists_file(fname)) {
printf("file %s already exists\n", fname);
tftp_send_error_message(upcb,to, to_port, TFTP_ERR_FILE_ALREADY_EXISTS);
udp_remove(upcb);
return -1;
}

fd = mfs_file_open(fname, MFS_MODE_CREATE);
if (fd < 0) {
xil_printf("unable to open file %s for writing\n", fname);
tftp_send_error_message(upcb, to, to_port, TFTP_ERR_DISKFULL);
udp_remove(upcb);
return -1;
}

/* this function is called from a callback => interrupts are disabled
* => we can use regular malloc
*/
args = mem_malloc(sizeof *args);
if (!args) {
xil_printf("unable to allocate memory for tftp args\n\r");
tftp_send_error_message(upcb, to, to_port, TFTP_ERR_FILE_NOT_FOUND);
udp_remove(upcb);
return -1;
}
args->op = TFTP_WRQ;
args->to_ip.addr = to->addr;
args->to_port = to_port;
args->fd = fd;
args->block = 0;

/* set callback for receives on this UDP PCB (Protocol Control Block) */
udp_recv(upcb, wrq_recv_callback, args);
/* initiate the transaction by sending the first ack */
tftp_send_ack_packet(upcb, to, to_port, args->block);
return 0;
```

```
}

/* for each new request (data in p->payload) from addr:port,
 * create a new port to serve the response, and start the response
 * process
 */
void process_tftp_request(struct pbuf *p, struct ip_addr *addr, u16_t port)
{
tftp_opcode op = tftp_decode_op(p->payload);
char fname[512];
struct udp_pcb *upcb;
err_t err;
/* create new UDP PCB structure */
upcb = udp_new();
if (!upcb) {
xil_printf("Error creating PCB. Out of Memory\n\r");
return;
}

/* bind to port 0 to receive next available free port */
err = udp_bind(upcb, IP_ADDR_ANY, 0);
if (err != ERR_OK) {
xil_printf("Unable to bind to port %d: err = %d\n\r", port, err);
return;
}

switch (op) {
case TFTP_RRQ:
tftp_extract_filename(fname, p->payload);
printf("TFTP RRQ (read request): %s\n", fname);
tftp_process_read(upcb, addr, port, fname);
break;
case TFTP_WRQ:
tftp_extract_filename(fname, p->payload);
printf("TFTP WRQ (write request): %s\n", fname);
tftp_process_write(upcb, addr, port, fname);
break;
default:
/* send a generic access violation message */
tftp_send_error_message(upcb, addr, port, TFTP_ERR_ACCESS_VIOLATION);
printf("TFTP unknown request op: %d\n", op);
udp_remove(upcb);
break;
}
```

```
}

/* the recv_callback function is called when there is a packet received
 * on the main tftp server port (69)
 */
void recv_callback(void *arg, struct udp_pcb *upcb,
struct pbuf *p, struct ip_addr *addr, u16_t port)
{
/* process new connection request */
process_tftp_request(p, addr, port);
pbuf_free(p);
}

int start_tftp_application()
{

struct udp_pcb *upcb;
err_t err;
unsigned port = 69;

/* create new UDP PCB structure */
upcb = udp_new();
if (!upcb) {
xil_printf("Error creating PCB. Out of Memory\n\r");
return -1;
}

/* bind to @port */
err = udp_bind(upcb, IP_ADDR_ANY, port);
if (err != ERR_OK) {
xil_printf("Unable to bind to port %d: err = %d\n\r", port, err);
return -2;
}
udp_recv(upcb, recv_callback, NULL);
xil_printf("TFTP server started @ port %d\n\r", port);
   return 0;
}

void
print_tftp_app_header()
{
xil_printf("\n\r\n\r-----lwIP TFTP Server ------\n\r");
xil_printf("Upload or download files from the Memory File System to your
localhost\n\r");
```

```
xil_printf("e.g.: upload from WinXP host to the board:\n\r");
xil_printf("          tftp -i 192.168.1.10 PUT source [destination]\n\r");
}

void
transfer_tftp_data()
{}
```

Κώδικας bootloader για προγραμματισμό της μνήμης flash (bootloader.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "blconfig.h"
#include "portab.h"
#include "errors.h"
#include "srec.h"
#include "xparameters.h"
#include <xilisf.h>                /* Serial Flash Library header file */

/* Defines */
#define CR      13

/* Comment the following line, if you want a smaller and faster bootloader which
will be silent */
//#define VERBOSE

/* Declarations */
static void display_progress (uint32_t lines);
static uint8_t load_exec ();
static uint8_t flash_get_srec_line (uint8_t *buf);
extern void init_stdout();

/* Declarations for ISF/SPI */
#define SPI_DEVICE_ID                   XPAR_SPI_FLASH_DEVICE_ID
#define ISF_SPI_SELECT          0x01
#define ISF_PAGE_SIZE           256

static XIsf Isf;
static XSpi Spi;
XIsf_ReadParam ReadParam;
volatile static int TransferInProgress; /* State of Spi Transfer */
```

```
u8 IsfWriteBuffer[ISF_PAGE_SIZE + XISF_CMD_SEND_EXTRA_BYTES];
u8 ReadBuffer[ISF_PAGE_SIZE + XISF_CMD_SEND_EXTRA_BYTES]; /* Read
Buffer */

extern int srec_line;

#ifdef __cplusplus
extern "C" {
#endif

extern void outbyte(char c);

#ifdef __cplusplus
}
#endif

/* Data structures */
static srec_info_t srinfo;
static uint8_t sr_buf[SREC_MAX_BYTES];
static uint8_t sr_data_buf[SREC_DATA_MAX_BYTES];

static uint8_t *flbuf;

#ifdef VERBOSE
static int8_t *errors[] = {
   "",
   "Error while copying executable image into RAM",
   "Error while reading an SREC line from flash",
   "SREC line is corrupted",
   "SREC has invalid checksum."
};
#endif

/* We don't use interrupts/exceptions.
  Dummy definitions to reduce code size on MicroBlaze */
#ifdef __MICROBLAZE__
void _interrupt_handler () {}
void _exception_handler () {}
void _hw_exception_handler () {}
#endif


int main()
{
```

```
uint8_t ret;

/*
* Initialize the SPI driver so that it's ready to use,
* specify the device ID that is generated in xparameters.h.
*/
XSpi_Initialize(&Spi, SPI_DEVICE_ID);

/*
 * Start the SPI driver so that interrupts and the device are enabled.
 */
XSpi_Start(&Spi);

/*
* Disable Global interrupt to use polled mode operation
*/
XSpi_IntrGlobalDisable(&Spi);

/*
* Initialize the Serial Flash Library.
*/
XIsf_Initialize(&Isf, &Spi, ISF_SPI_SELECT, IsfWriteBuffer);
init_stdout();

#ifdef VERBOSE
print ("\r\nSREC Bootloader\r\n");
print ("Loading SREC image from flash @ address: ");
putnum (FLASH_IMAGE_BASEADDR);
print ("\r\n");
#endif

flbuf = (uint8_t*)FLASH_IMAGE_BASEADDR;
ret = load_exec ();

/* If we reach here, we are in error */

#ifdef VERBOSE
if (ret > LD_SREC_LINE_ERROR) {
print ("ERROR in SREC line: ");
putnum (srec_line);
print (errors[ret]);
} else {
print ("ERROR: ");
print (errors[ret]);
```

```
}
#endif

return ret;
}

#ifdef VERBOSE
static void display_progress (uint32_t count)
{
/* Send carriage return */
outbyte (CR);
print  ("Bootloader: Processed (0x)");
putnum (count);
print (" S-records");
}
#endif

static uint8_t load_exec ()
{
uint8_t ret;
void (*laddr)();
int8_t done = 0;

srinfo.sr_data = sr_data_buf;

while (!done) {
if ((ret = flash_get_srec_line (sr_buf)) != 0)
return ret;

if ((ret = decode_srec_line (sr_buf, &srinfo)) != 0)
return ret;

#ifdef VERBOSE
      display_progress (srec_line);
#endif
switch (srinfo.type) {
case SREC_TYPE_0:
break;
case SREC_TYPE_1:
case SREC_TYPE_2:
case SREC_TYPE_3:
memcpy ((void*)srinfo.addr, (void*)srinfo.sr_data, srinfo.dlen);
break;
case SREC_TYPE_5:
```

```
break;
case SREC_TYPE_7:
case SREC_TYPE_8:
case SREC_TYPE_9:
laddr = (void (*)())srinfo.addr;
done = 1;
ret = 0;
break;
}
}

#ifdef VERBOSE
print ("\r\nExecuting program starting at address: ");
putnum ((uint32_t)laddr);
print ("\r\n");
#endif

(*laddr)();

/* We will be dead at this point */
return 0;
}


static uint8_t flash_get_srec_line (uint8_t *buf)
{
uint8_t c;
int count = 0;

while (1) {
/*
* Set the
* - Address in the Serial Flash where the data is to be read from.
* - Number of bytes to be read from the Serial Flash.
* - Read Buffer to which the data is to be read.
*/
TransferInProgress = TRUE;
ReadParam.Address = flbuf++;
ReadParam.NumBytes = 1;
ReadParam.ReadPtr = ReadBuffer;

XIsf_Read(&Isf, XISF_READ, (void*) &ReadParam);
IsfWaitForFlashNotBusy();
c  = ReadBuffer[XISF_CMD_SEND_EXTRA_BYTES];
```

```
if (c == 0xD) {
/* Eat up the 0xA too */
TransferInProgress = TRUE;
ReadParam.Address = flbuf++;
ReadParam.NumBytes = 1;
ReadParam.ReadPtr = ReadBuffer;
XIsf_Read(&Isf, XISF_READ, (void*) &ReadParam);
IsfWaitForFlashNotBusy();
c  = ReadBuffer[XISF_CMD_SEND_EXTRA_BYTES];

return 0;
}

*buf++ = c;
count++;
if (count > SREC_MAX_BYTES)
return LD_SREC_LINE_ERROR;
}
}

int IsfWaitForFlashNotBusy(void)
{
int Status;
u8 StatusReg;

while(1) {

/*
* Get the Status Register.
*/
TransferInProgress = TRUE;
Status = XIsf_GetStatus(&Isf, ReadBuffer);
if(Status != XST_SUCCESS) {
return XST_FAILURE;
}

/*
* Check if the Serial Flash is ready to accept the next
* command. If so break.
*/
StatusReg = ReadBuffer[BYTE2];
if((StatusReg & XISF_SR_IS_READY_MASK) == 0) {
break;
```

```
}
}

return XST_SUCCESS;
}


#ifdef __PPC__

#include <unistd.h>

/* Save some code and data space on PowerPC
   by defining a minimal exit */
void exit (int ret)
{
_exit (ret);
}
#endif
```

Παράρτημα Γ.: Κώδικας σε γλώσσα VHDL

Η περιγραφή του ηλεκτρονικού σχεδίου του περιφερειακού αυτοματισμού δειγματοληψίας έγινε σε γλώσσα VHDL. Στον πίνακα Γ.1 περιγράφονται τα σήματα που χρησιμοποιήθηκαν και έχουν διασυνδέσεις εκτός του κυκλώματος που περιγράφεται. Στον πίνακα Γ.2 περιγράφονται τα εσωτερικά σήματα του κυκλώματος.

Όνομα σήματος	Τύπος σήματος	Περιγραφή
Bus2IP_Clk	in std_logic	Σήμα εισόδου παλμού χρονισμού
start	in std_logic	Σήμα εισόδου εκκίνησης της διαδικασίας δειγματοληψίας
endproc	out std_logic	Σήμα εξόδου της προόδου της διαδικασίας δειγματοληψίας. '0'→ Διαδικασία σε εξέλιξη '1'→ Τέλος διαδικασίας
ValveDown	out std_logic	Σήμα εξόδου που ελέγχει τη λειτουργία της ηλεκτρικής βάνας εκκένωσης
ValveUp	out std_logic	Σήμα εξόδου που ελέγχει τη λειτουργία της ηλεκτρικής βάνας πλήρωσης
FloaterDown	in std_logic	Σήμα εισόδου του αισθητήρα στάθμης εκκένωσης
FloaterUp	in std_logic	Σήμα εισόδου του αισθητήρα στάθμης πλήρωσης

Πίνακας Γ.1: Σήματα με εξωτερικές διασυνδέσεις

Όνομα σήματος	Τύπος σήματος	Περιγραφή	Αρχική τιμή
temp	std_logic	Για έλεγχο της τιμής του σήματος vd. Όταν περάσει ένα λεπτό από την	'0'

		έναρξη της διαδικασίας δειγματοληψίας παίρνει την τιμή ‘1’	
temp2	std_logic	Για έλεγχο της προόδου της διαδικασίας δειγματοληψίας και δίνει τιμή στο σήμα εξόδου endproc. Όταν η διαδικασία ολοκληρωθεί παίρνει την τιμή ‘1’	‘0’
temp3	std_logic	Για έλεγχο της προόδου της διαδικασίας δειγματοληψίας και δίνει τιμή στο σήμα εξόδου endproc. Εάν η διαδικασία δεν έχει ολοκληρωθεί σε πέντε λεπτά παίρνει την τιμή ‘1’	‘0’
vu	std_logic	Για καταχώρηση τιμής στο σήμα ValveUp. ‘0’→Ηλεκτρική βάνα κλειστή ‘1’→Ηλεκτρική βάνα ανοιχτή	‘0’
vd	std_logic	Για καταχώρηση τιμής στο σήμα ValveDown. ‘0’→Ηλεκτρική βάνα κλειστή ‘1’→Ηλεκτρική βάνα ανοιχτή	‘0’
ticks1	integer	Για καταχώρηση της μέγιστης τιμής του count 1	400000000
ticks2	integer	Για καταχώρηση της μέγιστης τιμής του count 2	10
ticks3	integer	Για καταχώρηση της μέγιστης τιμής του count3	400000000
ticks4	integer	Για καταχώρηση της μέγιστης τιμής του count4	50

count1 - 4	integer	Απαριθμητές	0

Πίνακας Γ.2: Εσωτερικά σήματα

Στη συνέχεια παρατίθεται ο κώδικας VHDL που περιγράφει το ηλεκτρονικό κύκλωμα του συστήματος αυτοματισμού δειγματοληψίας.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

library proc_common_v3_00_a;
use proc_common_v3_00_a.proc_common_pkg.all;

entity user_logic is
  generic
  (
    -- ADD USER GENERICS BELOW THIS LINE ---------------
    --USER generics added here
    -- ADD USER GENERICS ABOVE THIS LINE ---------------

    -- DO NOT EDIT BELOW THIS LINE ----------------------
    -- Bus protocol parameters, do not add to or delete
    C_SLV_DWIDTH                : integer             := 32;
    C_NUM_REG                   : integer             := 1
    -- DO NOT EDIT ABOVE THIS LINE ----------------------
  );
  port
  (
    -- ADD USER PORTS BELOW THIS LINE ------------------
          start : in std_logic;
    ValveDown: out std_logic;
          ValveUp : out std_logic;
          FloaterDown : in std_logic;
          FloaterUp : in std_logic;
          endproc : out std_logic;
    -- ADD USER PORTS ABOVE THIS LINE ------------------

    -- DO NOT EDIT BELOW THIS LINE ----------------------
    -- Bus protocol ports, do not add to or delete
    Bus2IP_Clk                  : in  std_logic;
    Bus2IP_Reset                : in  std_logic;
    Bus2IP_Data                 : in  std_logic_vector(0 to C_SLV_DWIDTH-1);
```

```
    Bus2IP_BE                  : in  std_logic_vector(0 to C_SLV_DWIDTH/8-1);
    Bus2IP_RdCE                : in  std_logic_vector(0 to C_NUM_REG-1);
    Bus2IP_WrCE                : in  std_logic_vector(0 to C_NUM_REG-1);
    IP2Bus_Data                : out std_logic_vector(0 to C_SLV_DWIDTH-1);
    IP2Bus_RdAck               : out std_logic;
    IP2Bus_WrAck               : out std_logic;
    IP2Bus_Error               : out std_logic
    -- DO NOT EDIT ABOVE THIS LINE ----------------------
  );

  attribute MAX_FANOUT : string;
  attribute SIGIS : string;

  attribute SIGIS of Bus2IP_Clk    : signal is "CLK";
  attribute SIGIS of Bus2IP_Reset  : signal is "RST";

end entity user_logic;

------------------------------------------------------------------------------
-- Architecture section
------------------------------------------------------------------------------

architecture IMP of user_logic is

        signal temp: std_logic :='0';
        signal temp2: std_logic :='0';
        signal temp3: std_logic :='0';
        signal vu:std_logic :='0';
        signal vd:std_logic :='0';
        signal ticks1:integer :=400000000;
        signal ticks2:integer :=10;
        signal ticks3:integer :=400000000;
        signal ticks4:integer :=50;
        signal count1:integer := 0;
        signal count2:integer := 0;
        signal count3:integer := 0;
        signal count4:integer := 0;
  ---------------------------------------
  -- Signals for user logic slave model s/w accessible register example
  ---------------------------------------
  signal slv_reg0                 : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg_write_sel            : std_logic_vector(0 to 0);
  signal slv_reg_read_sel             : std_logic_vector(0 to 0);
  signal slv_ip2bus_data              : std_logic_vector(0 to C_SLV_DWIDTH-1);
```

```
  signal slv_read_ack                : std_logic;
  signal slv_write_ack               : std_logic;

begin

-- delay 1min --
  process (Bus2IP_Clk) is
  begin
  if start='1' then
          if (Bus2IP_Clk'event and Bus2IP_Clk='1') then
                  if count1<=ticks1 then
                          count1<=count1 + 1;
                                  else
                                  count2<=count2 + 1;
                                  count1<=0;
                  end if;
                  if count2>=ticks2 then
                          temp <= '1';
                          count1<=ticks1;
                  end if;
          end if;
   end if;
  end process;
 -- delay 5mins --
  process (Bus2IP_Clk) is
  begin
  if start='1' then
          if (Bus2IP_Clk'event and Bus2IP_Clk='1') then
                  if count3<=ticks3 then
                          count3<=count3 + 1;
                                  else
                                  count4<=count4 + 1;
                                  count3<=0;
                  end if;
                  if count4>=ticks4 then
                          temp3<= '1';
                          count4<=ticks4;
                  end if;
          end if;
   end if;
  end process;
-- sampling automation --
   process (Bus2IP_Clk) is
   begin
```

```
if start='1' then
                vd<='1';
                        if FloaterDown='0' then
                                vu<='1';
                        end if;
                        if temp='1' then
                                vd<='0';
                        end if;
                        if FloaterUp='1' and temp='1' then
                                vu<='0';
                                temp2<='1';

                        end if;
                        if temp3='1' then
                                vd<='0';
                                vu<='0';
                        end if;
        end if;
        end process;

        ValveDown<=vd;
        ValveUp<=vu;
        endproc<='1' when temp2='1' else
                         '1' when temp3='1' else
                         '0';

slv_reg_write_sel <= Bus2IP_WrCE(0 to 0);
slv_reg_read_sel  <= Bus2IP_RdCE(0 to 0);
slv_write_ack     <= Bus2IP_WrCE(0);
slv_read_ack      <= Bus2IP_RdCE(0);

-- implement slave model software accessible register(s)
SLAVE_REG_WRITE_PROC : process( Bus2IP_Clk ) is
begin

 if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
  if Bus2IP_Reset = '1' then
   slv_reg0 <= (others => '0');
  else
   case slv_reg_write_sel is
    when "1" =>
     for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
      if ( Bus2IP_BE(byte_index) = '1' ) then
```

```
            slv_reg0(byte_index*8 to byte_index*8+7) <= Bus2IP_Data(byte_index*8
to byte_index*8+7);
          end if;
        end loop;
      when others => null;
    end case;
  end if;
 end if;

 end process SLAVE_REG_WRITE_PROC;

 -- implement slave model software accessible register(s) read mux
 SLAVE_REG_READ_PROC : process( slv_reg_read_sel, slv_reg0 ) is
 begin

  case slv_reg_read_sel is
   when "1" => slv_ip2bus_data <= slv_reg0;
   when others => slv_ip2bus_data <= (others => '0');
  end case;

 end process SLAVE_REG_READ_PROC;

 ---------------------------------------
 -- Example code to drive IP to Bus signals
 ---------------------------------------
 IP2Bus_Data  <= slv_ip2bus_data when slv_read_ack = '1' else
              (others => '0');

 IP2Bus_WrAck <= slv_write_ack;
 IP2Bus_RdAck <= slv_read_ack;
 IP2Bus_Error <= '0';

end IMP;
```

ΠΑΡΑΡΤΗΜΑ Δ.: Κώδικας σε γλώσσα JavaScript και HTML

Παρακάτω παρατίθεται ο κώδικας που έχει αναπτυχθεί σε γλώσσα JavaScript και HTML. Είναι υπεύθυνος για την υλοποίηση της διεπαφής χρήστη του συστήματος στον τερματικό κόμβο διαδικτύου που βρίσκεται στο Πανεπιστήμιο Ιωαννίνων.

- **Αρχείο slicing.js**

```
var fs = require ('fs');
var Str1;


var i;
while(true){

var day =new Date();
var x;
x=day.getDate();
y=day.getMonth()+1;
z=day.getFullYear();

if (fs.existsSync('ReceivedFiles/'+x+'_'+y+'_'+z+'.csv')) {
}else {
    fs.appendFileSync('ReceivedFiles/'+x+'_'+y+'_'+z+'.csv',
'Date,Temp,Conductivity\n');
}
if (fs.existsSync('Plot/Data.csv')) {
}else {
    fs.appendFileSync('Plot/Data.csv', 'Date,Temp,Conductivity\n');
}

filereadName="file1.txt"
fileWriteName1='ReceivedFiles/'+x+'_'+y+'_'+z+'.csv'
fileWriteName2='Plot/Data.csv'
if (fs.existsSync(filereadName)){
    console.log("yparxei");
    ReadStr="";
            Str1="OLD ONE:";
            console.log('old Str: ' + Str1);
            Str1=readFileToStr(filereadName);
            console.log('Read File is   :'+filereadName);
            console.log('ReadStr  :'+Str1);
```

```
                for (i = 0; i < (Str1.length-16); i=i+16) {
                time1=Str1.slice(i+2,i+4);
                date1=Str1.slice(i+4,i+10);
                temp1=Str1.slice(i+10,i+14);
                cond1=Str1.slice(i+14,i+18);

                cond1Int1=parseInt(cond1,16)
                tempInt1=parseInt(temp1,16);
                time1=parseInt(time1,16);
if ((time1==1)||(time1==2)||(time1==3)||(time1==4)){
if (time1==1){
time1=00+':'+00+':'+00;
}
else if (time1==2){
    time1=06+':'+00+':'+00;
}
else if (time1==3){
    time1=12+':'+00+':'+00;
}
else {
    time1=18+':'+00+':'+00;
}
date2=date1.slice(0,2);
date3=date1.slice(2,4);
date4=date1.slice(4,6);

date4='20'+date4;
cond=0.55*cond1Int1-247;
tempa=0.023*tempInt1-23.04;
cond=cond.toFixed(2);
tempa=tempa.toFixed(2);

    var      sampleStr= date2+'-'+date3+'-'+date4+' '+time1+','+tempa+','+cond;




        console.log('Str : ' + sampleStr);
        writeDataToFile(fileWriteName1,sampleStr );
              writeDataToFile(fileWriteName2,sampleStr );}}
    fs.unlinkSync(filereadName);
}else{
    console.log("oxi");
```

```
}



function readFileToStr(fileReadName){
var data=fs.readFileSync(fileReadName,'hex');
console.log('break4');
    console.log(data);
     return data
}


function writeDataToFile(fileName,Str1){
     fs.appendFileSync(fileName,Str1+'\n', (err) => {
   if (err) throw err;
   console.log('The file is updated!');
});
}
}
```

- **Αρχείο plot.js**

```
var socket = io.connect('http://localhost:8080');
          socket.on('message', function(message) {
             alert('The server has a message for you: ' + message);
          })


$('#button3').click(function () {
             socket.emit('message3', 'Plot');

Plotly.d3.csv('Plot/Data.csv', function(err, rows){

  function unpack(rows, key) {
  return rows.map(function(row) { return row[key]; });
}


var trace1 = {
  type: "scatter",
  mode: "lines+markers",
  name: 'Temp',
  x: unpack(rows, 'Date'),
  y: unpack(rows, 'Temp'),
```

```
  line: {color: '#17BECF'}
}

var trace2 = {
  type: "scatter",
  mode: "lines+markers",
  name: 'Cond1',
  x: unpack(rows, 'Date'),
  y: unpack(rows, 'Conductivity'),
  line: {color: '#FF8C00'}
}

var data2 = [trace2];
var data1 = [trace1];


var layout1 = {
  title: 'Διάγραμμα Θερμοκρασίας',

  xaxis: {
    title: 'Χρόνος (UTC+2)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#7f7f7f'
    }
  },
  yaxis: {
    title: 'Θερμοκρασία (°C)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#000000'
    }
  }

};

var layout2 = {
  title: 'Διάγραμμα αγωγιμότητας',
xaxis: {


    title: 'Χρόνος (UTC+2)',
```

```
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#7f7f7f'
    }
  },
  yaxis: {
    title: 'Αγωγιμότητα (μS/cm)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#000000'
    }
  }

};
Plotly.newPlot('Conductivitydiv', data1, layout1);
Plotly.newPlot('Temperaturediv', data2, layout2);
})
//////////////////////////https://plot.ly/javascript/subplots/

          })
// Τέλος γραφήματος

$('#button4').click(function () {
              socket.emit('message4', 'Plot');

Plotly.d3.csv('Plot/Data2.csv', function(err, rows){

  function unpack(rows, key) {
  return rows.map(function(row) { return row[key]; });
}


var trace1 = {
  type: "scatter",
  mode: "lines+markers",
  name: 'Temp',
  x: unpack(rows, 'Date'),
  y: unpack(rows, 'Temp'),
  line: {color: '#17BECF'}
}

var trace2 = {
```

```
  type: "scatter",
  mode: "lines+markers",
  name: 'Cond1',
  x: unpack(rows, 'Date'),
  y: unpack(rows, 'Conductivity'),
  line: {color: '#FF8C00'}
}

var data2 = [trace2];
var data1 = [trace1];


var layout1 = {
  title: 'Διάγραμμα Θερμοκρασίας',

  xaxis: {
    title: 'Χρόνος (UTC+2)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#7f7f7f'
    }
  },
  yaxis: {
    title: 'Θερμοκρασία (°C)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#000000'
    }
  }

};

var layout2 = {
  title: 'Διάγραμμα αγωγιμότητας',
xaxis: {


    title: 'Χρόνος (UTC+2)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#7f7f7f'
```

```
    }
  },
  yaxis: {
    title: 'Αγωγιμότητα (μS/cm)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#000000'
    }
  }

};
Plotly.newPlot('Conductivitydiv', data1, layout1);
Plotly.newPlot('Temperaturediv', data2, layout2);
})
//////////////////////////https://plot.ly/javascript/subplots/

          })
// Τέλος γραφήματος
```

- **Αρχείο server.js**

```
// HTTP Portion
var http = require('http');
// Path module
var path = require('path');

// Using the filesystem module
var fs = require('fs');

var server = http.createServer(handleRequest);
server.listen(81);

console.log('Server started on port 8080');

function handleRequest(req, res) {
  // What did we request?
  var pathname = req.url;

  // If blank let's ask for index.html
  if (pathname == '/') {
    pathname = '\\index.html';
  }

  // Ok what's our file extension
```

```
 var ext = path.extname(pathname);

 // Map extension to file type
 var typeExt = {
  '.html': 'text/html',
  '.js':   'text/javascript',
  '.css':  'text/css'
 };
 // What is it?  Default to plain text

 var contentType = typeExt[ext] || 'text/plain';

 var htmlfilename = path.join(__dirname, '..\\TFTPClient', pathname);

 // User file system module
 fs.readFile(htmlfilename,
  // Callback function for reading
  function (err, data) {
   // if there is an error
   if (err) {
    res.writeHead(500);
    return res.end('Error loading ' + htmlfilename);
   }
   // Otherwise, send the data, the contents of the file
   res.writeHead(200,{ 'Content-Type': contentType });
   res.end(data);
  }
 );

}
var io = require('socket.io').listen(server);
io.sockets.on('connection', function (socket) {
   socket.emit('message', 'You are connected!');
 socket.on('message3', function (message) {
       console.log('Message3: ' + message);
              });
              socket.on('message4', function (message) {
       console.log('Message4: ' + message);
              });
     // Νέο κουμπί
     });
```

- **Αρχείο index.html**

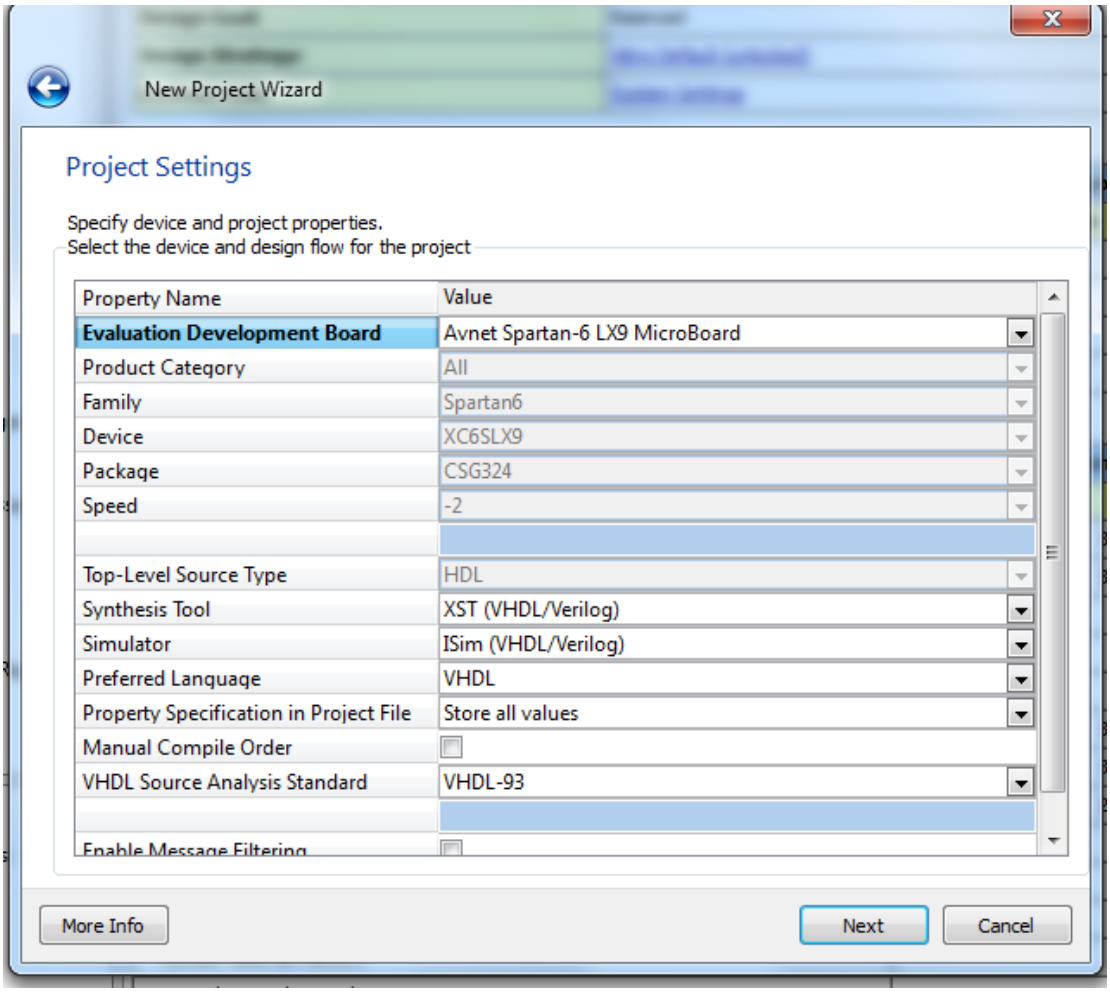
```
<!DOCTYPE html>
<html>
  <head>


      <meta charset="utf-8" />
      <title>EWS</title>
            <!-- Plotly.js -->
 <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
  </head>

  <body>
     <h1>Πανεπιστήμιο Ιωαννίνων - Τμήμα Φυσικής - HEPLab</h1>
           <h1>Σύστημα έγκαιρης προειδοποίησης ρύπανσης νερού</h1>

           <p><input type="button" value="Σταθμός 1" id="button3" /></p>
           <p><input type="button" value="Σταθμός 2" id="button4" /></p>
           <!--Νέο κουμπί

           <!--<p><input type="button" value="Σταθμός2" id="button4" /></p>
           <!-- Plotly chart will be drawn inside this DIV -->
     <div id="Temperaturediv"></div>
     <div id="Conductivitydiv"></div>
      <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
      <script src="/socket.io/socket.io.js"></script>

           <script type="text/javascript" src="testplot2.js" ></script>
  </body>
</html>
```

ΠΑΡΑΡΤΗΜΑ Ε.: Μικροεπεξεργαστής MicroBlaze και παραμετροποίησή του

Στο παράρτημα περιγράφεται αναλυτικά η δημιουργία του μικροεπεξεργαστή MicroBlaze (παραμετροποιημένο για το σύστημα έγκαιρης προειδοποίησης που αναπτύχθηκε) στο FPGA Spartan-6 της εταιρίας Xilinx, που βρίσκεται στην πλακέτα Avnet LX9 Microboard, με την αναπτυξιακή πλατφόρμα ISE Design Suite 14.7.

1. Αρχικά, με την εκκίνηση του προγράμματος Xilinx ISE Project Navigator: **Start→All Programs→Xilinx Design Tools→ISE Design Suite 14.7→ISE Design Tools→Project Navigator** δημιουργείται μία νέα εφαρμογή: **File →New Project.**

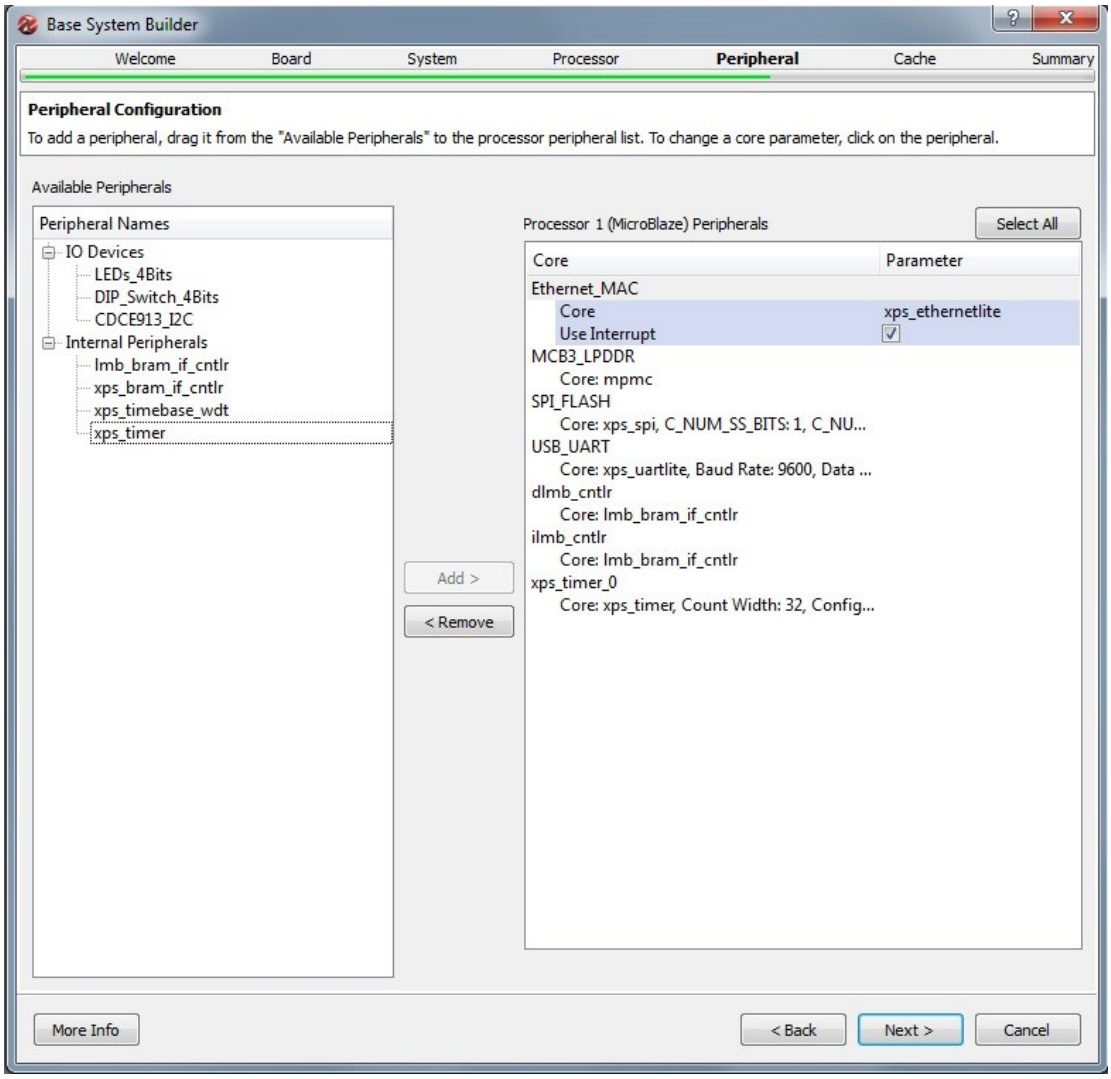
2. Στην πρώτη καρτέλα που εμφανίζεται δίνεται όνομα στην εφαρμογή και επιλέγεται η τοποθεσία αποθήκευσης από το χρήστη. Στο επόμενο παράθυρο (με next), γίνονται οι απαραίτητες επιλογές, που φαίνονται στην εικόνα Ε.1 και ακολούθως **next** και **finish.**



Εικόνα Ε.1 : Επιλογή ηλεκτρονικής πλακέτας

3. Στην εφαρμογή που δημιουργήθηκε, εισάγεται ο μικροεπεξεργαστής MicroBlaze μέσω της διαδρομής **Project→New Source→Embedded processor→Next→Finish** και έπειτα ανοίγει αυτόματα το πρόγραμμα Xilinx Platform Studio (XPS) εμφανίζοντας την καρτέλα δημιουργίας νέας εφαρμογής χρησιμοποιώντας τον BSB Wizard. **Yes**, και επιλογή του **PLB System** και **OK.**

4. Next μέχρι την καρτέλα processor όπου η τοπική μνήμη επιλέγεται να είναι 32kB: **Local Memory→32kB.** Στην καρτέλα **Peripheral** αφαιρούνται τα περιφερειακά CDCE913_I^{2}C, LEDs_4Bits και DIP_Switch_4Bits. Προστίθεται το περιφερειακό xps_timer και ενεργοποιούνται τα interrupts για τα περιφερειακά Ethernet_MAC και xps_timer_0 όπως φαίνεται στην εικόνα Ε.2.

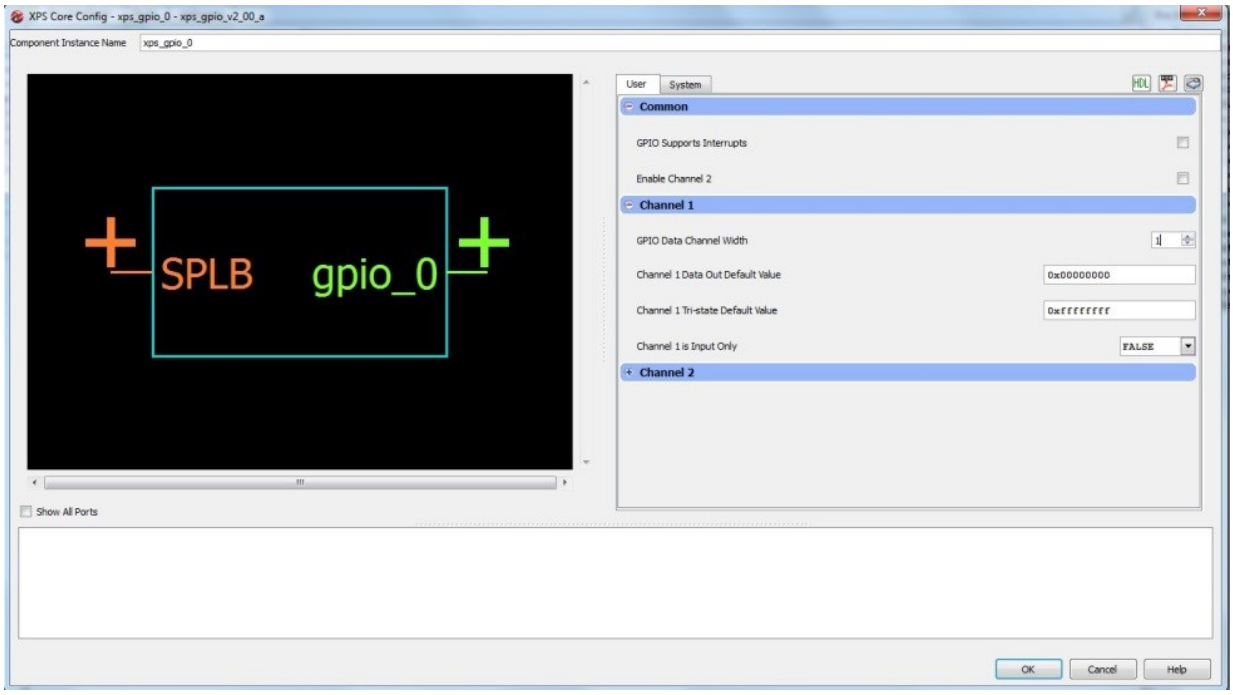


Εικόνα Ε.2: Επιλογή περιφερειακών

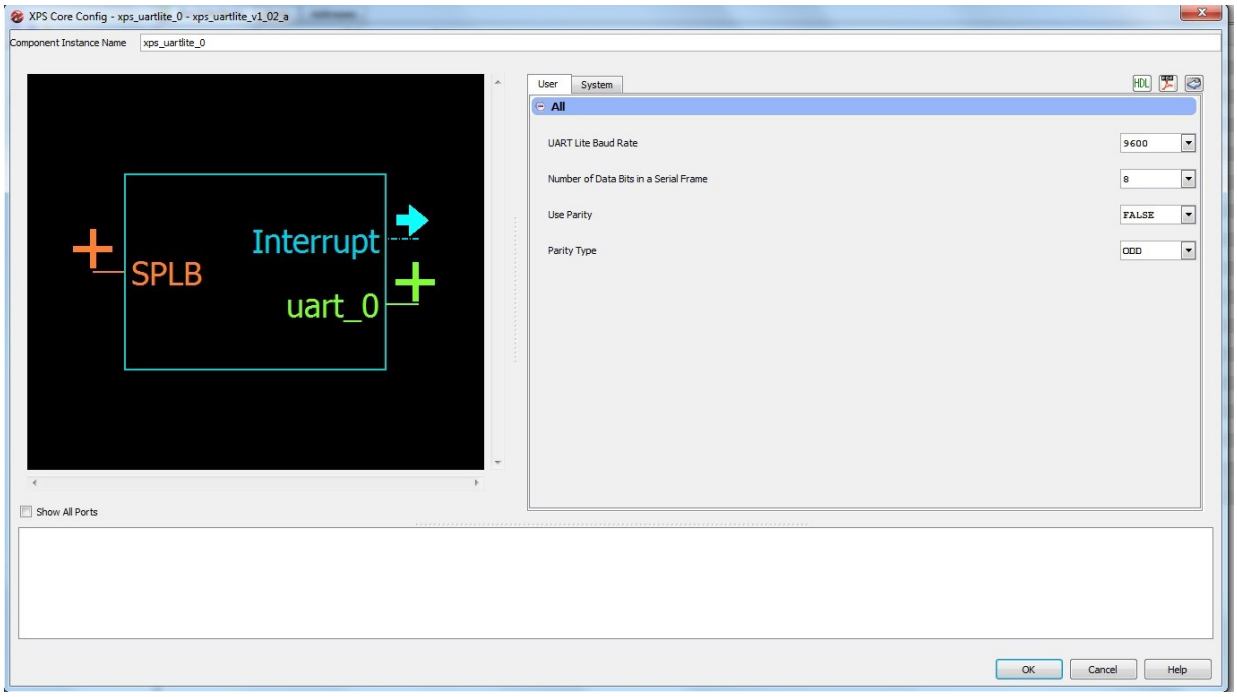
Next και στην επόμενη καρτέλα **Cache** ενεργοποιούνται τα **Instruction Cache** και **Data Cache** και το μέγεθος τους ορίζεται σε 8kB. **Next→finish** και έτσι η δημιουργία του base system τελειώνει.

5. Εμφανίζεται η βασική επιφάνεια εργασίας του προγράμματος. Από την καρτέλα **IP catalog** προστίθενται στο σύστημα τα εξής περιφερειακά: **8x XPS General Purpose**

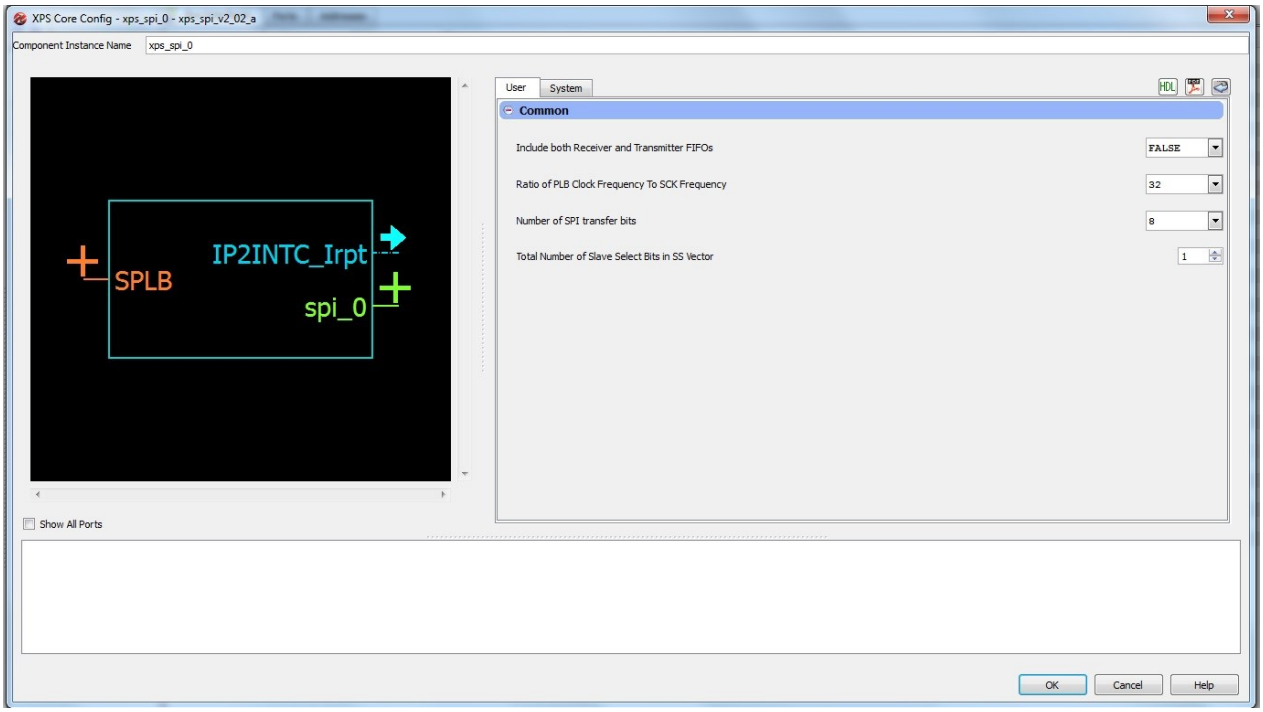
IO, 1x XPS IIC Interface, 1x XPS UART (lite) και **1x XPS SPI Interface**. Επιπλέον, εισάγεται και το περιφερειακό αυτοματισμού δειγματοληψίας, του οποίου η δημιουργία περιγράφεται στο παράρτημα Δ. Με επιλογή των περιφερειακών στην καρτέλα **Bus Interfaces** ανοίγει η καρτέλα παραμετροποίησης τους και γίνονται οι επιλογές που φαίνονται στις εικόνες Ε.3 (gpio), Ε.4 (uart) και Ε.5 (spi): Για τα **gpios** στο channel 1, αλλάζει το **gpio data channel width→1**. Για το **uart**, **parity→false** και για το **spi**, **include both receiver and transmitter FIFOs→false.**



Εικόνα Ε.3 : Παραμετροποίηση περιφερειακού gpio

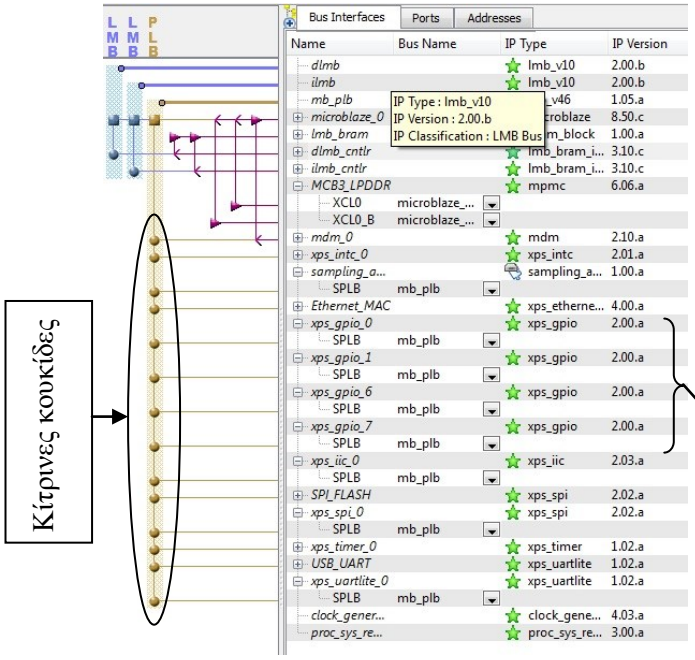


Εικόνα Ε.4 : Παραμετροποίηση περιφερειακού UART



Εικόνα Ε.5 : Παραμετροποίηση περιφερειακού SPI

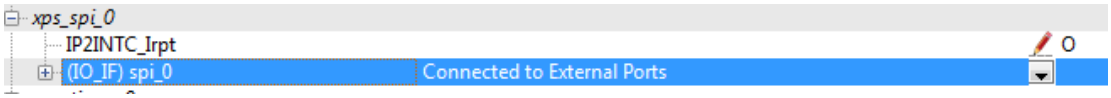
Έπειτα, με επιλογή στις κίτρινες κουκίδες συνδέονται όλα τα περιφερειακά στο **PLB.** Οι απαραίτητες διασυνδέσεις φαίνονται στην εικόνα Ε.6 :



Παρόλο που έχουν προστεθεί 8 gpios φαίνονται μόνο 4 από αυτά, για λόγο που θα εξηγηθεί παρακάτω. Όταν συνδεθούν όλα τα περιφερειακά, στην καρτέλα **Addresses** δίνονται σε αυτά διευθύνσεις στον δίαυλο PLB μέσω του κουμπιού **Generate Addresses** στην πάνω δεξιά γωνία της επιφάνειας χρήστη.

Εικόνα Ε.6 : Διασυνδέσεις περιφερειακών

6. Στην καρτέλα **Ports** γίνεται η διασύνδεση των ακροδεκτών των περιφερειακών με το υπόλοιπο σύστημα. Για τα περιφερειακά I^2C και SPI οι ακροδέκτες συνδέονται στα **external ports** (εικόνα Ε.7):



Εικόνα Ε.7 : Σύνδεση στα external ports

Για τα gpio_0 έως gpio_3, συνδέεται ο ακροδέκτης **GPIO_IO_O** και για τα gpio_4, gpio_5 ο ακροδέκτης **GPIO_IO_I**. Για το περιφερειακό αυτοματισμού δειγματοληψίας γίνονται οι παρακάτω διασυνδέσεις:

Ακροδέκτης	**Port**
start	xps_gpio_6:GPIO_IO_O
endproc	xps_gpio_7:GPIO_IO_I
ValveDown	External Ports:xps_gpio_3:_GPIO_IO_O
ValveUp	External Ports:xps_gpio_2:_GPIO_IO_O
FloaterDown	External Ports:xps_gpio_5:_GPIO_IO_I
FloaterUp	External Ports:xps_gpio_4:_GPIO_IO_I

Πίνακας Ε.1 : Διασύνδεση περιγερειακού αυτοματισμού δειγματοληψίας

Έπειτα, στην καρτέλα **Bus Interfaces**, με επιλογή των gpio_2 έως και gpio_5 γίνεται η διαγραφή τους, με πίεση του πλήκτρου Delete και επιλέγοντας **Delete instance but do not remove the net** στο παράθυρο που εμφανίζεται, έτσι ώστε να πάρουν τη θέση τους οι ακροδέκτες του περιφερειακού αυτοματισμού δειγματοληψίας.

7. Στην καρτέλα **project** στο κάτω αριστερά μέρος της επιφάνειας χρήστη, με επιλογή του **UCF File** ανοίγει το αρχείο και προστίθενται οι παρακάτω γραμμές κώδικα:

Net xps_gpio_0_GPIO_IO_O_pin LOC=C18 | IOSTANDARD = LVCMOS33;
Net xps_gpio_1_GPIO_IO_O_pin LOC=C17 | IOSTANDARD = LVCMOS33;
Net xps_gpio_2_GPIO_IO_O_pin LOC=F16 | IOSTANDARD = LVCMOS33;
Net xps_gpio_3_GPIO_IO_O_pin LOC=F15 | IOSTANDARD = LVCMOS33;
Net xps_gpio_4_GPIO_IO_I_pin LOC=F14 | IOSTANDARD = LVCMOS33;
Net xps_gpio_5_GPIO_IO_I_pin LOC=G14 | IOSTANDARD = LVCMOS33;
Net xps_iic_0_Scl_pin LOC=E18 | IOSTANDARD = LVCMOS33 | PULLUP;
Net xps_iic_0_Sda_pin LOC=E16 | IOSTANDARD = LVCMOS33 | PULLUP;
Net xps_spi_0_MISO_pin LOC=F18 | IOSTANDARD = LVCMOS33 | PULLUP;
Net xps_spi_0_MOSI_pin LOC=K13 | IOSTANDARD = LVCMOS33 | PULLUP;
Net xps_spi_0_SCK_pin LOC=F17 | IOSTANDARD = LVCMOS33;
Net xps_spi_0_SS_pin LOC=K12 | IOSTANDARD = LVCMOS33;
Net xps_uartlite_0_RX_pin LOC=G13 | IOSTANDARD = LVCMOS33;
Net xps_uartlite_0_TX_pin LOC=H12 | IOSTANDARD = LVCMOS33;

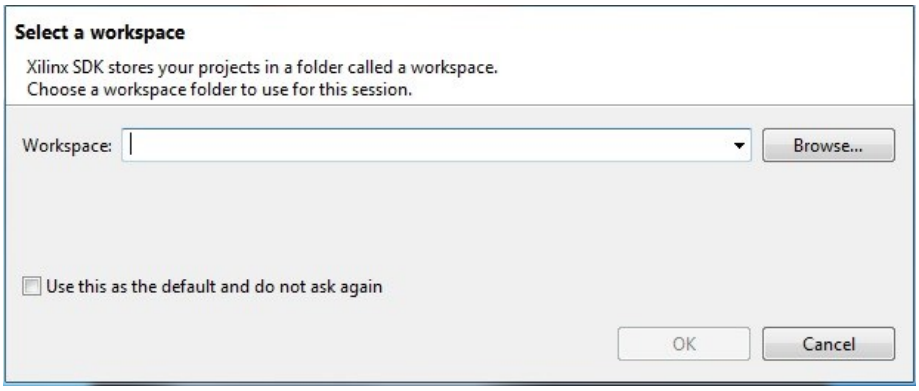
Με την παραπάνω συνδεσμολογία οι ακροδέκτες των περιφερειακών συνδεόνται στα PMoDs της ηλεκτρονικής πλακέτας. Μέσω της διαδρομής **File→Save** το αρχείο αποθηκεύεται.

8. Τέλος, εξάγεται το σύστημα που δημιουργήθηκε: **Project→Export Hardware Design to SDK→Export only.** Όταν η διαδικασία τελειώσει κλείνει το XPS.

9. Στο πρόγραμμα ISE Project Navigator προστίθεται το αρχείο ucf: **Project→add copy of source→project directory→embedded processor name→data→ucf file** και έπειτα με επιλογή του **Generate Top HDL Source** δημιουργείται ο κώδικας VHDL που περιγράφει τον μικροεπεξεργαστή που αναπτύχθηκε. Τέλος, με επιλογή του **Export Hardware Design To SDK with Bitstream** δημιουργείται το αρχείο .bit για προγραμματισμό του FPGA (εικόνα Ε.8):

Εικόνα Ε.8: Δημιουργία αρχείου .bit

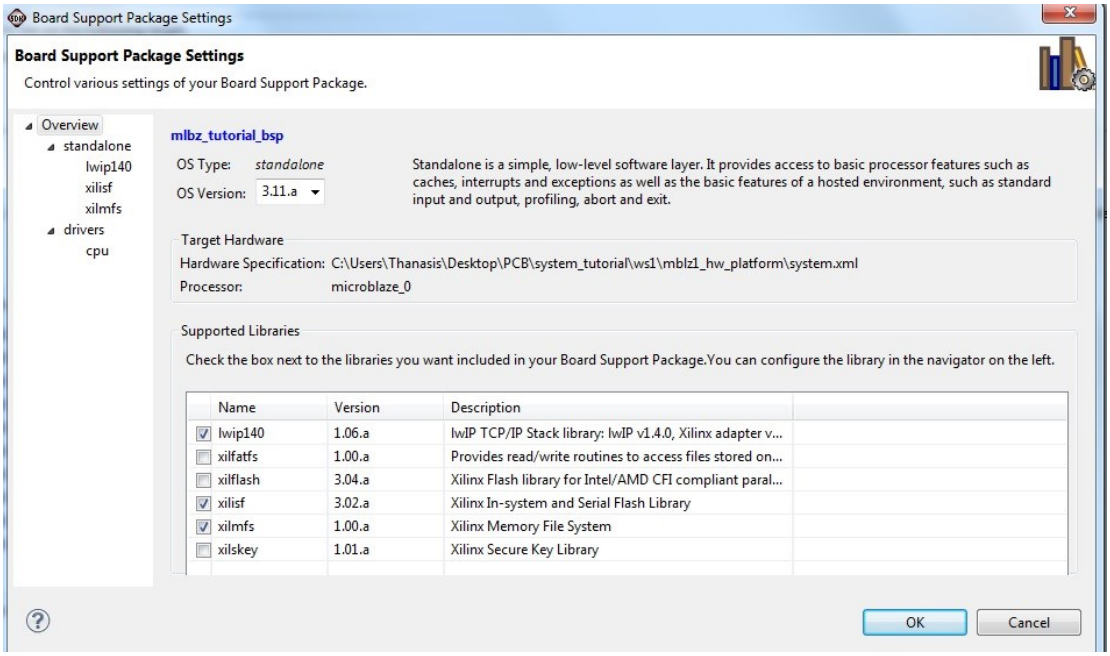
10. Όταν η διαδικασία του βήματος 9 τελειώσει, ανοίγει αυτόματα το πρόγραμμα Software Development Kit (SDK) και εμφανίζεται το παράθυρο επιλογής **workspace (εικόνα Ε.9)**:



Εικόνα Ε.9: Επιλογή workspace

Προτείνεται η δημιουργία της workspace στον φάκελο που δημιουργήθηκε στα βήματα 1,2 και περιέχει τα αρχεία του μικροεπεξεργαστή που αναπτύχθηκε. Αριστερό κλικ στο κουμπί **OK**.

11. Αρχικά, γίνεται η δημιουργία ενός καινούργιου bsp: **File→New→Board Support Package**. Μετά την ονοματοδοσία με κλικ στο κουμπί **Finish** γίνεται η δημιουργία του νέου bsp. Στην καρτέλα **Board Support Package Settings** επιλέγονται οι βιβλιοθήκες **lwip140, xilisf** και **xilmfs** (εικόνα Ε.10):



Εικόνα Ε.10: Απαραίτητες βιβλιοθήκες

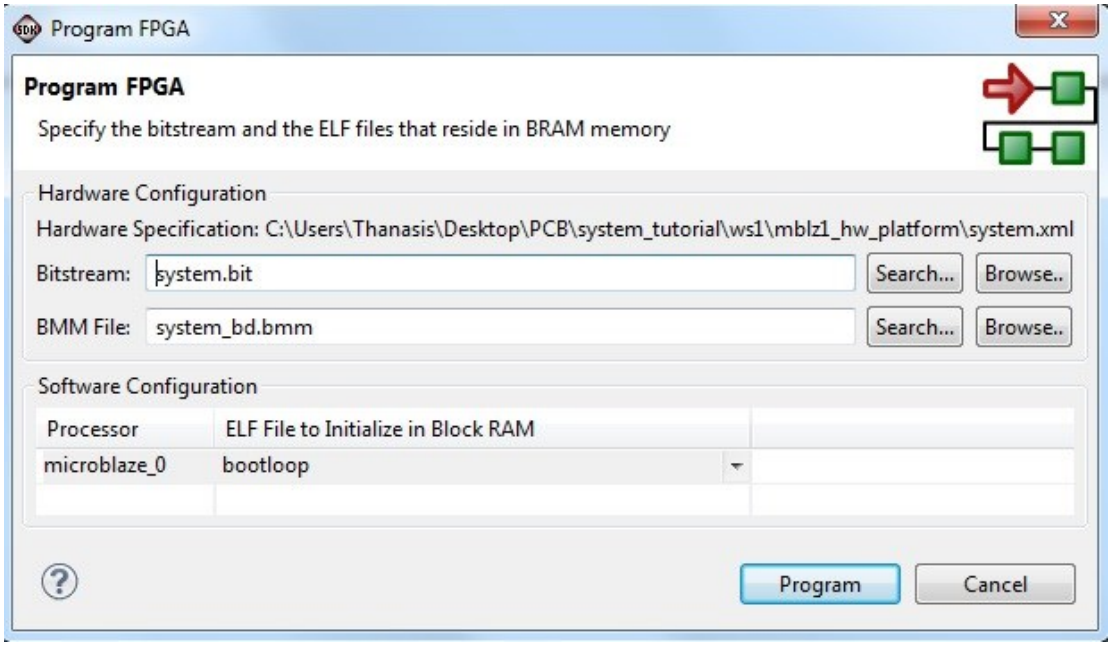
Επιλέγοντας τη βιβλιοθήκη xilmfs, πρέπει να αλλάξει η τιμή **base_address** από 0x100000 σε 0x45000000, ώστε να μην υπάρχει κώδικας στις θέσεις της μνήμης RAM που θα αποθηκευθούν αρχεία. Επίσης πρέπει να αλλάξει η τιμή **serial_flash_family** της βιβλιοθήκης xilisf από **1→3,** για να επιλεχθεί η κατασκευάστρια εταιρία της μνήμης flash **.** Κλικ στο κουμπί **OK** και ξεκινάει η διαδικασία δημιουργίας του bsp.

12. Όταν τελειώσει η διαδικασία του βήματος 11, δημιουργούνται τα projects που περιέχουν τον κυρίως κώδικα του συστήματος: **File→New→Application Project.** Πρέπει να δημιουργηθούν δύο project, το ένα θα περιέχει τον κώδικα που περιγράφει το σύστημα και το άλλο τον κώδικα του bootloader που θα φορτώσει τον κώδικα που υπάρχει στη μνήμη flash της ηλεκτρονικής πλακέτας στην μνήμη RAM του FPGA κατά την εκκίνηση της λειτουργίας του. Η εκτέλεση του κώδικα από τη μνήμη RAM του FPGA είναι απαραίτητη καθώς το μέγεθος του κώδικα είναι μεγαλύτερο από τη χωρητικότητα της μνήμης bram του FPGA. Και για τα δύο επιλέγεται **Board Support Package→Use existing.** Πατώντας το κουμπί **next** εμφανίζεται η καρτέλα με τα **templates**. Για το project που περιέχει τον κυρίως κώδικα επιλέγεται άδεια εφαρμογή (**empty application**), ενώ για τον bootloader → **SREC Bootloader.** Η διαδικασία του βήματος 12 τελειώνει με **finish**.

13. Επιλέγοντας τους φακέλους που δημιουργήθηκαν εισάγονται τα αρχεία που περιέχουν τον κώδικα: **File→Import→General→File System→Επιλογή του φακέλου που περιέχει τον κώδικα→Select all→Finish.** Ο φάκελος που περιέχει τον κυρίως κώδικα ονομάζεται "complete code" και ο φάκελος που περιέχει τον κώδικα του bootloader ονομάζεται "bootloader_code".

14. Έπειτα, πριν γίνει ο προγραμματισμός του FPGA, πρέπει πρώτα να ρυθμιστεί η διασύνδεση JTAG, ώστε να αναγνωρίζει το καλώδιο USB-to-JTAG που παρέχει η κατασκευάστρια εταιρία. Η παραμετροποίηση γίνεται μέσω της διαδρομής: **Xilinx Tools→ Configure JTAG settings.** Στο πεδίο **type** επιλέγεται **3rd party cable, Xilinx Plug-in** και στο πεδίο **other options** πρέπει να πληκτρολογηθεί: **-cable type xilinx_plugin modulename digilent_plugin.**

15. Ο προσωρινός προγραμματισμός του FPGA για τη δημιουργία του αρχείου με κατάληξη **.elf**, που είναι απαραίτητο για τον προγραμματισμό της μνήμης flash, ή για αποσφαλμάτωση του κώδικα που αναπτύσσεται γίνεται με αριστερό κλικ στο εικονίδιο **Program FPGA** που βρίσκεται στην κεντρική επιφάνειας εργασίας του SDK (εικόνα Ε.11).



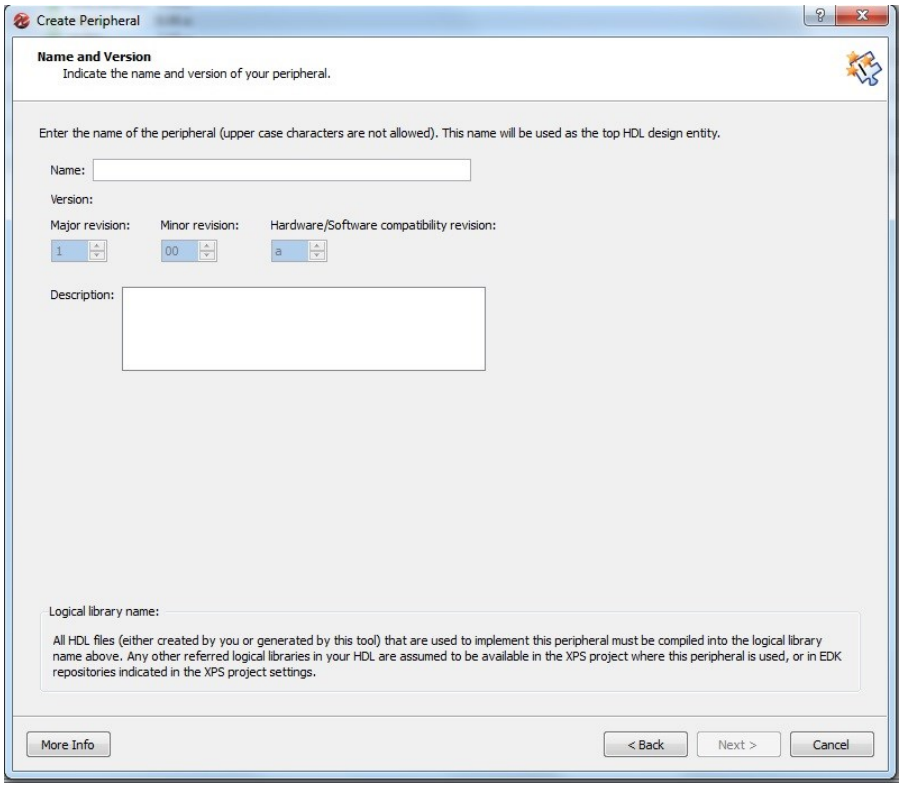
Εικόνα Ε.11: Προγραμματισμός FPGA

Στο πεδίο **bitstream** εισάγεται το αρχείο με κατάληξη **.bit** και στο πεδίο **BMM File** εισάγεται το αρχείο με κατάληξη **.bmm.** Τα παραπάνω αρχεία βρίσκονται στον φάκελο που περιέχει τα αρχεία της εφαρμογής που δημιουργήθηκε. Με επιλογή του **Program** το FPGA προγραμματίζεται. Όταν τελειώσει η διαδικασία προγραμματισμού, επιλέγοντας **Run** στην γραμμή εργαλείων της επιφάνειας εργασίας και ακολουθώντας τη διαδρομή **Run → Run configurations…** εμφανίζεται το παράθυρο παραμετροποίησης της εκτέλεσης του κώδικα που έχει προγραμματιστεί στο FPGA. Στην καρτέλα **STDIO Connection** ενεργοποιείται η επιλογή **Connect STDIO to console** ώστε να συνδεθεί η έξοδος του FPGA με το πρόγραμμα SDK. Τέλος, επιλέγοντας **Run**, στο παράθυρο που έχει εμφανιστεί, ξεκινάει η εκτέλεση του κώδικα και ολοκληρώνεται η διαδικασία δημιουργίας και προγραμματισμού του μικροεπεξεργαστή Microblaze.

ΠΑΡΑΡΤΗΜΑ ΣΤ.: <u>Δημιουργία περιφερειακού αυτοματισμού δειγματοληψίας</u>

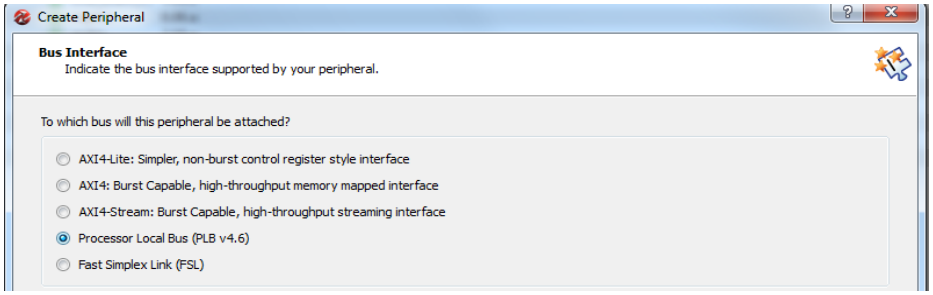
Η δημιουργία του περιφερειακού αυτοματισμού δειγματοληψίας έγινε με χρήση του προγράμματος Xilinx Platform Studio (XPS).

1. Αρχικά, γίνεται εκκίνηση του προγράμματος και στην κεντρική επιφάνεια εργασίας, μέσω της διαδρομής **Hardware→Create or import peripheral** ξεκινά η διαδικασία δημιουργίας του περιφερειακού. Επιλογή **Next** μέχρι να εμφανιστεί η καρτέλα με όνομα **Name and version** (εικόνα ΣΤ.1), όπου γίνεται η ονοματοδοσία του περιφερειακού.



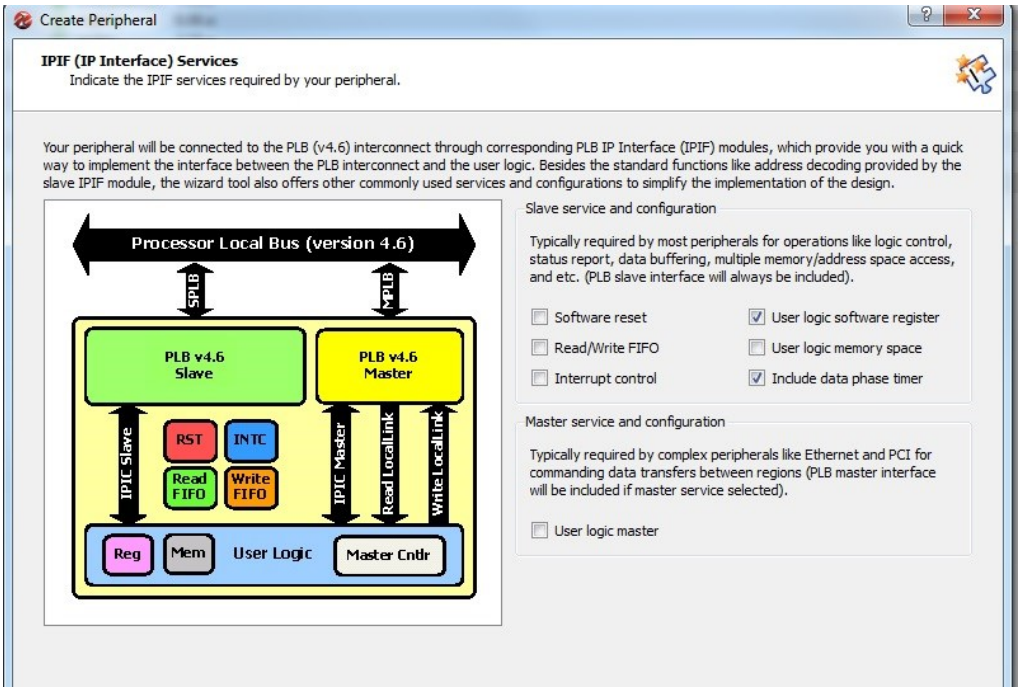
Εικόνα ΣΤ.1: Καρτέλα ονοματοδοσίας περιφερειακού

Επιλογή **Next** μέχρι να εμφανιστεί η καρτέλα με όνομα **Bus Interface** (εικόνα ΣΤ.2). Επιλέγεται ο δίαυλος **Processor Local Bus** και **Next.**



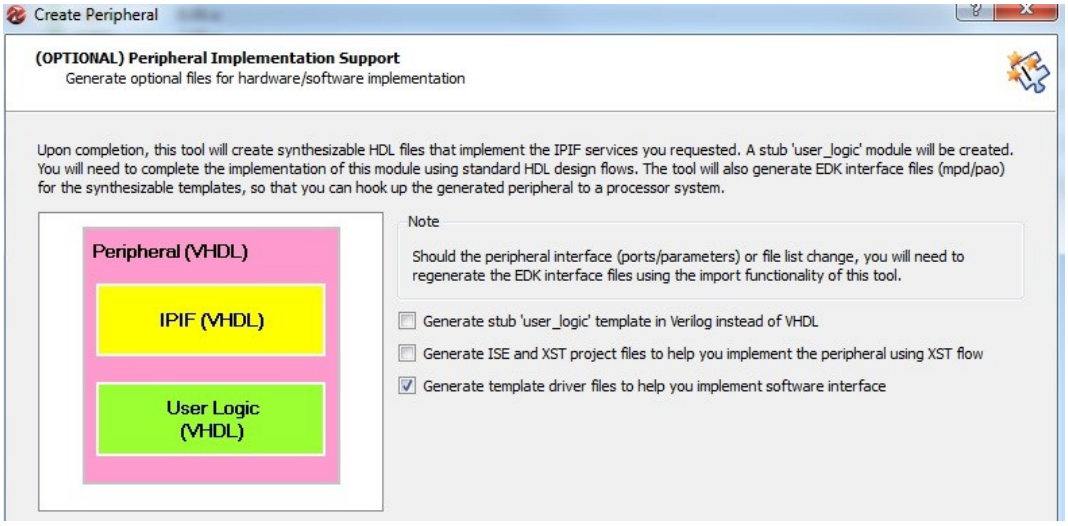
Εικόνα ΣΤ.2: Καρτέλα επιλογής διαύλου

Έπειτα, στην καρτέλα με όνομα **IPIF (IP interfaces) Services** γίνονται οι επιλογές που φαίνονται στην εικόνα ΣΤ.3:



Εικόνα ΣΤ.3: Καρτέλα IPIF Services

Στη συνέχεια, επιλογή **Next** μέχρι να εμφανιστεί η καρτέλα με όνομα **Peripheral Implementation Support** και γίνεται η επιλογή που φαίνεται στην εικόνα ΣΤ.4:



Εικόνα ΣΤ.4: Καρτέλα Peripheral Implementation Support

Επιλογή **Finish** και η διαδικασία δημιουργίας του νέου περιγερειακού τελειώνει. Ακολουθεί η παραμετροποίηση του.

2. Η παραμετροποίηση του περιφερειακού γίνεται με χρήση της γλώσσας προγραμματισμού VHDL, μέσω της προσθήκης κώδικα στα παρακάτω αρχεία: **user_logic.vhd**, **peripheral_name.vhd** και **peripheral_name.mpd**. Τα αρχεία δημιουργούνται αυτόματα από το πρόγραμμα XPS και βρίσκονται στο φάκελο που δημιουργήθηκε στα βήματα 1,2 του Π.Ε και περιέχει τα αρχεία του μικροεπεξεργαστή για τον οποίο αναπτύσσεται το περιφερειακό. Στο αρχείο **user_logic.vhd** προστίθεται ο κώδικας VHDL που περιγράφει τη λειτουργία του περιφερειακού. Περιγραφή των σημάτων που χρησιμοποιήθηκαν υπάρχει στον πίνακα Γ.1 του παραρτήματος Γ. Συγκεκριμένα:

- Δηλώνονται τα σήματα εισόδου/εξόδου:

```
-- ADD USER PORTS BELOW THIS LINE ------------------
    start : in std_logic;
    ValveDown: out std_logic;
    ValveUp : out std_logic;
    FloaterDown : in std_logic;
    FloaterUp : in std_logic;
    endproc : out std_logic;
-- ADD USER PORTS ABOVE THIS LINE ------------------
```

- Δηλώνονται τα εσωτερικά σήματα:

```
--USER signal declarations added here, as needed for user logic
    signal temp: std_logic :='0';
    signal temp2: std_logic :='0';
    signal temp3: std_logic :='0';
    signal vu:std_logic :='0';
    signal vd:std_logic :='0';
    signal ticks1:integer :=400000000;
    signal ticks2:integer :=10;
    signal ticks3:integer :=400000000;
    signal ticks4:integer :=10;
    signal ticks5:integer :=5;
    signal count1:integer := 0;
    signal count2:integer := 0;
    signal count3:integer := 0;
    signal count4:integer := 0;
    signal count5:integer := 0;
```

- Προστίθεται ο κώδικας που περιγράφει το περιφερειακό:

```
--USER logic implementation added here
 -- delay 1min --
process (Bus2IP_Clk) is
begin
if start='1' then
if (Bus2IP_Clk'event and Bus2IP_Clk='1') then
```

```
        if count1<=ticks1 then
                count1<=count1 + 1;
                else
                count2<=count2 + 1;
                count1<=0;
        end if;
        if count2>=ticks2 then
                temp <= '1';
                count1<=ticks1;
        end if;
        end if;
 end if;
 end process;
-- delay 5mins --
 process (Bus2IP_Clk) is
 begin
 if start='1' then
        if (Bus2IP_Clk'event and Bus2IP_Clk='1') then
                if count3<=ticks3 then
                        count3<=count3 + 1;
                                else
                                count4<=count4 + 1;
                                count3<=0;
                end if;
                if count4>=ticks4 then
                        temp3<= '1';
                        count4<=ticks4;
                end if;
        end if;
 end if;
 end process;
-- sampling automation --
  process (Bus2IP_Clk) is
  begin
  if start='1' then
        vd<='1';
  if FloaterDown='0' then
        vu<='1';
  end if;
  if temp='1' then
        vd<='0';
  end if;
  if FloaterUp='1' and temp='1' then
        vu<='0';
```

```
        temp2<='1';

  end if;
  if temp3='1' then
        vd<='0';
        vu<='0';
  end if;
  end if;
  end process;
  ValveDown<=vd;
  ValveUp<=vu;
  endproc<='1' when temp2='1' else
              '1' when temp3='1' else
              '0';
```

Το αρχείο με όνομα **peripheral_name.vhd** είναι το ιεραρχικά πρώτο αρχείο του περιγερειακού που αναπτύχθηκε. Σε αυτό δηλώνονται οι βιβλιοθήκες που χρησιμοποιούνται και τα σήματα εισόδου/εξόδου. Προστίθενται οι παρακάτω γραμμές κώδικα:

- Δηλώνονται τα σήματα εισόδου/εξόδου:

```
-- ADD USER PORTS BELOW THIS LINE ------------------
        start: in std_logic;
        ValveDown: out std_logic;
        ValveUp : out std_logic;
        FloaterDown : in std_logic;
        FloaterUp : in std_logic;
        endproc:out std_logic;
   -- ADD USER PORTS ABOVE THIS LINE ------------------
```

- Ενώνονται τα σήματα εισόδου/εξόδου που δηλώθηκαν παραπάνω με τα σήματα που δηλώθηκαν στο αρχείο **user_logic.vhd**:

```
-- MAP USER PORTS BELOW THIS LINE ------------------
         ValveDown => ValveDown,
         ValveUp => ValveUp,
         FloaterDown => FloaterDown,
         FloaterUp => FloaterUp,
         start => start,
         endproc => endproc,
-- MAP USER PORTS ABOVE THIS LINE ------------------
```

Τέλος, στο αρχείο με όνομα **peripheral_name.mpd** προστίθεται ο κώδικας που εμφανίζει τα σήματα εισόδου/εξόδου στην καρτέλα **Ports** του XPS:

```
## Ports
PORT start = "", DIR = I
PORT ValveDown = "", DIR = O
```

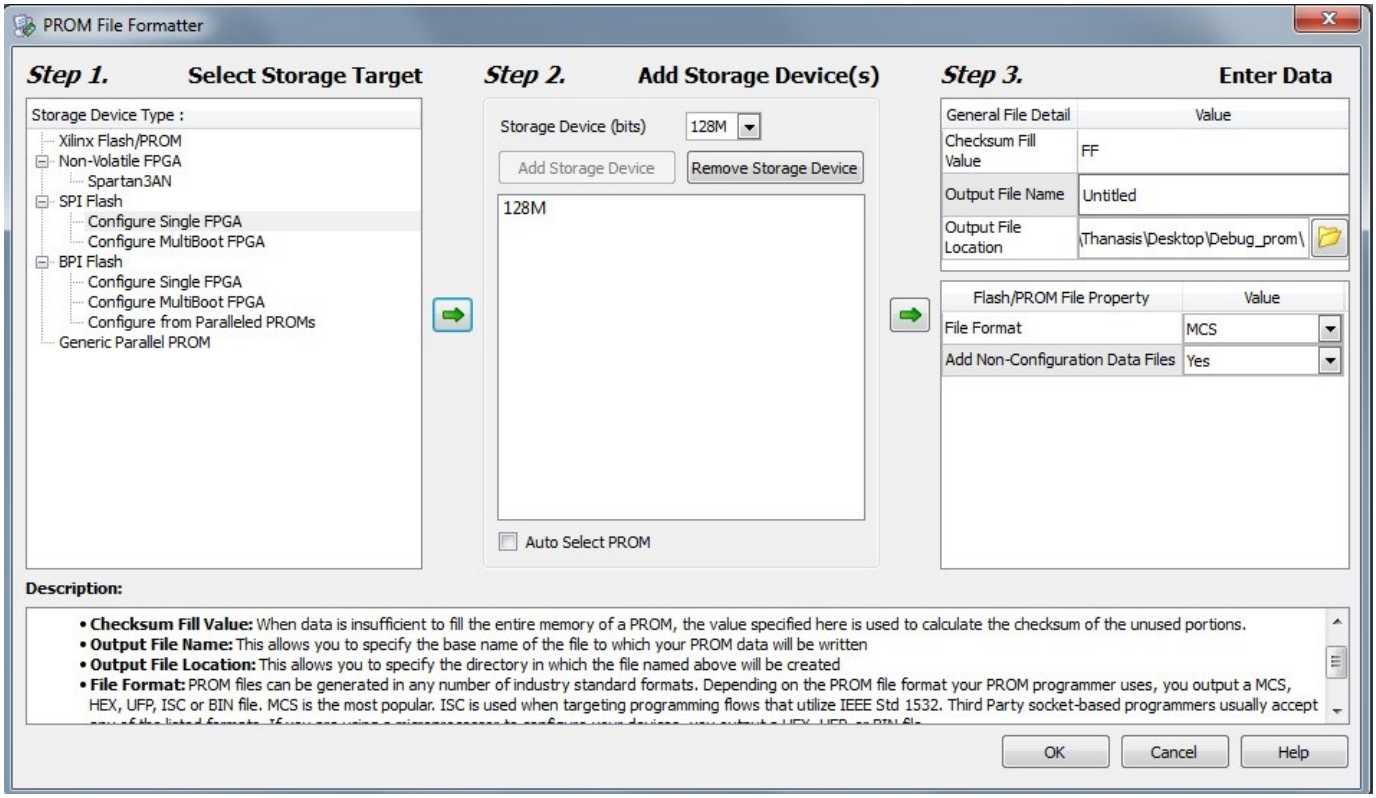
PORT ValveUp = "", DIR = O
PORT FloaterDown = "", DIR = I
PORT FloaterUp = "", DIR = I
PORT endproc ="", DIR = O

3. Μετά την προσθήκη του παραπάνω κώδικα το περιφερειακό είναι έτοιμο για σύνδεση στον μικροεπεξεργαστή. Στην κεντρική επιφάνεια εργασίας του XPS μέσω της διαδρομής **Project→Rescan user repositories**, το περιφερειακό εμφανίζεται στην καρτέλα **IP Catalog** στην κατηγορία **Project Local Pcores**. Τέλος, ακολουθώντας τα βήματα 5 – 7 του παραρτήματος Ε γίνονται οι απαραίτητες ενέργειες ώστε το περιφερειακό να συνδεθεί με τον μικροεπεξεργαστή.

ΠΑΡΑΡΤΗΜΑ Ζ.: <u>Προγραμματισμός μνήμης flash</u>

Ο προγραμματισμός της μνήμης flash, της ηλεκτρονικής πλακέτας Avnet LX9 Microboard, ως PROM (Programmable read only memory) γίνεται με χρήση του προγράμματος iMPACT και η διαδικασία προγραμματισμού περιγράφεται στα παρακάτω βήματα:

1. Αρχικά γίνεται εκκίνηση του προγράμματος και στην αριστερή πλευρά της κεντρικής επιφάνειας εργασίας με την επιλογή **Create PROM file** ξεκινά η δημιουργία του αρχείου, με κατάληξη .mcs, που προγραμματίζει την μνήμη flash σαν PROM. Έπειτα εμφανίζεται το παράθυρο με τις επιλογές παραμετροποίησης του αρχείου και γίνονται οι επιλογές που φαίνονται στην εικόνα Ζ.1. Στην καρτέλα με όνομα **Step 1** γίνεται η επιλογή του τύπου της μνήμης flash που διαθέτει η ηλεκτρονική πλακέτα. Επιλέγεται **SPI Flash→Configure Single FPGA**. Στην καρτέλα με όνομα **Step 2** επιλέγεται η χωρητικότητα της μνήμης flash. Στο πεδίο **Storage Device (bits)** επιλέγονται **128M**. Στην καρτέλα με όνομα **Step 3** γίνεται η παραμετροποίηση του αρχείου .mcs. Στο πεδίο **Output File Name** εισάγεται το όνομα του αρχείου που θα δημιουργηθεί και στο πεδίο **Output File Location** ο φάκελος που θα αποθηκευθεί. Στο πεδίο **Flash/Prom File property** γίνονται οι επιλογές που φαίνονται στην εικόνα Ζ.1.



Εικόνα Ζ.1: Επιλογή παραμέτρων αρχείου προγραμματισμού μνήμης flash

2. Με το πέρας της διαδικασίας του παραπάνω βήματος εμφανίζεται το παράθυρο προσθήκης αρχείου ρυθμίσεων. Επιλογή **OK** και στη συνέχεια επιλέγεται το αρχείο

με κατάληξη .bit, που βρίσκεται στο φάκελο που δημιουργήθηκε στα βήματα 1,2 του παραρτήματος Ε. Στο επόμενο παράθυρο που εμφανίζεται αυτόματα (add another device) επιλογή **NO.** Έπειτα, εμφανίζεται το παράθυρο εισαγωγής αρχείου δεδομένων και εισάγεται το αρχείο με κατάληξη .srec, το οποίο βρίσκεται στον ίδιο φάκελο με το αρχείο ρυθμίσεων. Στο παράθυρο εισαγωγής πρόσθετου αρχείου δεδομένων επιλογή **NO.** Επιλογή του **Generate File** στην αριστερή πλευρά της κεντρικής επιφάνειας εργασίας του προγράμματος, το αρχείο είναι έτοιμο για εισαγωγή στη μνήμη flash.

3. Επιλογή του **Boundary Scan** και έπειτα επιλογή του **File → Initialize Chain.** Με την παραπάνω επιλογή το πρόγραμμα συνδέεται με το FPGA μέσω του καλωδίου JTAG και είναι έτοιμο για προγραμματισμό. Επιλογή του εικονιδίου του FPGA και επιλογή του **Add SPI/BPI Device**. Εισάγεται το αρχείο με κατάληξη .mcs, που δημιουργήθηκε στα παραπάνω βήματα και στο παράθυρο που εμφανίζεται γίνονται οι επιλογές που φαίνονται στην εικόνα Z.2:

Εικόνα Z.2: Επιλογή μνήμης flash

Επιλογή **OK.** Επιλογή του εικονιδίου της PROM και επιλογή του **Program.** Ξεκινάει έτσι η διαδικασία προγραμματισμού της μνήμης flash. Όταν τελειώσει η παραπάνω διαδικασία η μνήμη flash είναι πλέον PROM και προγραμματίζει το FPGA κάθε φορά ξεκινάει η λειτουργία του.

ΠΑΡΑΡΤΗΜΑ Η: <u>Λίστα υλικών συστήματος</u>

Όνομα στοιχείου	Ποσό-τητα	Package
Capacitor 0.1uF	6	SMD0805
Capacitor 1uF	2	SMD0805
Capacitor 2.2uF	1	SMD1210
Capacitor 10uF	6	SMD0805
Capacitor 100uF	2	SMD
Capacitor 330uF	1	SMD
Resistor 12Ω	1	SMD0805
Resistor 10W – 15Ω	1	Thruhole
Resistor 56Ω	1	SMD0805
Resistor 100Ω	4	SMD0805
Resistor 200Ω	1	SMD0805
Resistor 330Ω	4	SMD0805
Resistor 430Ω	1	SMD0805
Resistor 510Ω	2	SMD0805
Resistor 1KΩ	7	SMD0805
Resistor 10KΩ	9	SMD0805
Resistor 12KΩ	1	SMD0805
Resistor 123KΩ	1	SMD0805
Resistor 500KΩ	2	SMD0805
Diode 1N4148	2	SMD0805
1-pin male header	14	Thruhole
2-pin female terminal block	8	Thruhole
2-pin male terminal block	8	Thruhole
3-pin female terminal block	1	Thruhole
3-pin male terminal block	1	Thruhole
12-pin male header	2	Thruhole
Avnet LX9 Microboard	1	-
MAX6460-Maxim Integrated	1	6 SOT23
LM324-Fairchild Semiconductors	1	TSSOP 14
Adafruit Ultimate GPS breakout	1	Thruhole
NVR4003N	3	SOT23
RJ-45	2	Thruhole
AD7998-Analog Devices	1	TSSOP-20
PTN78020-Texas Instruments	1	Thruhole

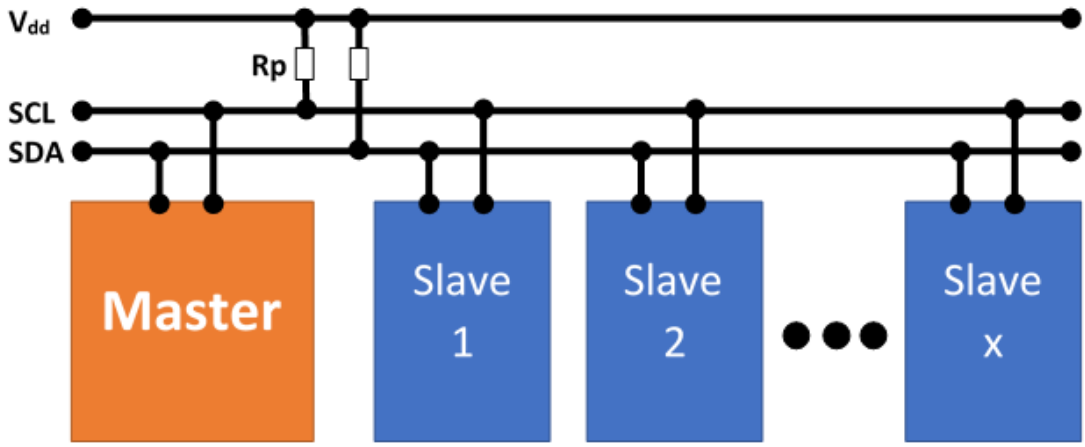
PTR08100-Texas Instruments	1	Thruhole
AQY282S-Panasonic	4	SOP
AMS1117-Advanced Monolithic Systems	1	SOT23
Crystal 32.768 KHz	1	Thruhole
DS1337-Maxim Integrated	1	8 uSOP
microSD 16GB-SDHC-SanDisk	1	-
DM3AT-SF-PEJM5-Hirose Electric	1	SMD
WQ Cond-Global Water	1	-
TP4056-NanJing Top Power ASIC Corp	1	Thruhole
LM314-Texas Instruments	1	SOT23
Samsung ICR 18650	1	-
Maxwell K2 2.7 supercapacitors	12	-
Maxwell K2 integration kit	6	-
BSL-SE16C-120M Photovoltaic panel	2	
Φωτοευαίσθητη πλακέτα	2	-
Κουτί Plastim PP3004	1	-
Βίδα μήκους 13.5cm	4	-
Βίδα μήκους 1.5cm	4	-
Βίδα μήκους 3 cm	6	-
Παξιμάδι διαμέτρου 1 cm	12	-
Παξιμάδι διαμέτρου 0.5 cm	14	-
Στυπιοθλίπτης διαμέτρου 1.7cm	4	-
Στυπιοθλίπτης διαμέτρου 3.8cm	2	-

ΠΑΡΑΡΤΗΜΑ Θ: Πρωτόκολλα επικοινωνίας

Για τη διασύνδεση και την επικοινωνία του FPGA με τα υπόλοιπα ολοκληρωμένα του συστήματος χρησιμοποιήθηκαν τα πρωτόκολλα επικοινωνίας I^2C, SPI και UART. Ο δίαυλος I^2C χρησιμοποιείται από το ρολόι πραγματικού χρόνου (RTC) και τον μετατροπέα αναλογικού σήματος σε ψηφιακό (ADC), ο δίαυλος SPI για επικοινωνία με την κάρτα μνήμης microSD και η διεπαφή UART από το GPS του συστήματος.

Π.Θ.1 Δίαυλος I^2C

Το I^2C (inter-integrated circuit) είναι ένας σύγχρονος σειριακός δίαυλος επικοινωνίας που χρησιμοποιείται για τη διασύνδεση ολοκληρωμένων που βρίσκονται πάνω στο ίδιο τυπωμένο κύκλωμα ή σε διαφορετικά, με την προϋπόθεση ότι υπάρχει μεταξύ τους διασύνδεση με καλώδια [30]. Χρησιμοποιεί δύο καλώδια για την μεταφορά δεδομένων, διπλής κατεύθυνσης, τα SDA και SCL. Η γραμμή SCL (serial clock line) χρησιμοποιείται για την μετάδοση του παλμού συγχρονισμού του εξυπηρετητή του διαύλου με τους εξυπηρετούμενους, ενώ η γραμμή SDA (serial data line) για τη μεταφορά δεδομένων.



Εικόνα Θ.1: Διασύνδεση διαύλου I^2C

Η εικόνα Θ.1 απεικονίζει το διάγραμμα μίας τυπικής διασύνδεσης ενός διαύλου I^2C. Οι γραμμές πρέπει να είναι συνδεδεμένες μέσω αντιστάσεων pull-ups στην γραμμή τάσης τροφοδοσίας, η οποία συνήθως έχει τιμές 3.3 ή 5V. Ο μέγιστος αριθμός των στοιχείων που μπορούν να συνδεθούν στο δίαυλο περιορίζεται από τον αριθμό των διαθέσιμων διευθύνσεων, καθώς χρησιμοποιούνται 7-bits για αυτό τον σκοπό. Ένας τρόπος λύσης αυτού του προβλήματος, είναι η δημιουργία, από τους κατασκευαστές των ολοκληρωμένων, ακροδέκτη που επιτρέπει την επιλογή διεύθυνσης ανάλογα με τη σύνδεσή του στο κύκλωμα. Τέλος, κάθε συσκευή μπορεί να χρησιμοποιηθεί σαν εξυπηρετητής ή εξυπηρετούμενος στο δίαυλο. Ο εξυπηρετητής αποφασίζει για τις

λειτουργίες που γίνονται στο δίαυλο, ενώ ο εξυπηρετούμενος υπακούει στις εντολές του master.

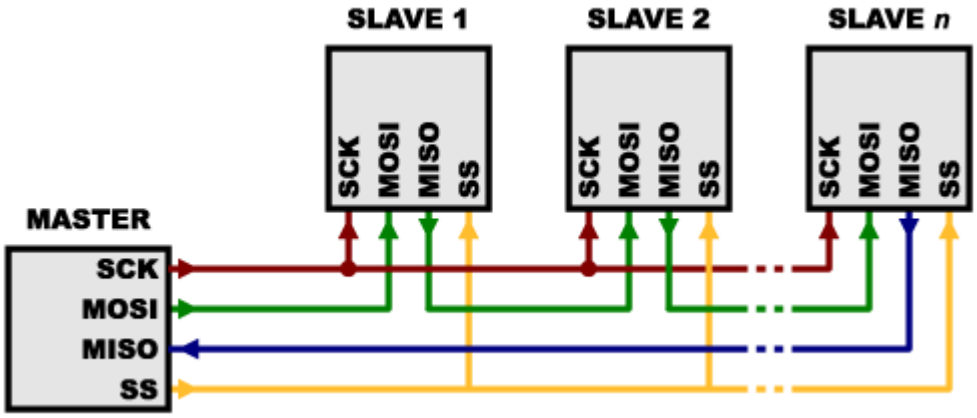
Π.Ι.2 Δίαυλος SPI

Το SPI (Serial Peripheral Interface bus) είναι ένας σύγχρονος σειριακός δίαυλος επικοινωνίας που χρησιμοποιείται για την διασύνδεση ολοκληρωμένων. Η λειτουργία του είναι πλήρως αμφίδρομη και για την υλοποίηση του χρειάζονται τέσσερις γραμμές σήματος [30]. Οι ονομασίες των ακροδεκτών του πρωτοκόλλου μπορεί να διαφέρουν ανάλογα με τον κατασκευαστή του εκάστοτε ολοκληρωμένου. Οι συνήθεις ονομασίες των ακροδεκτών σύνδεσης στον δίαυλο φαίνονται στον πίνακα Θ.1:

SCK	Σειριακό ρολόι χρονισμού, που ελέγχεται από τον master
/SS	Active-low ακροδέκτης, για την επιλογή slave από τον master
MISO	Ακροδέκτης για μεταφορά δεδομένων από τον slave στον master
MOSI	Ακροδέκτης για μεταφορά δεδομένων από τον master στον slave

Πίνακας Θ.1: Ακροδέκτες δίαυλου SPI

Τα πλεονέκτημά του σε σχέση με άλλα πρωτόκολλα επικοινωνίας, για παράδειγμα το I^2C, είναι η μεγάλη ταχύτητα μετάδοσης δεδομένων, καθώς είναι πλήρως αμφίδρομος δίαυλος, δηλαδή όσο ο master στέλνει δεδομένα στον slave μπορεί να λαμβάνει δεδομένα ταυτόχρονα από αυτόν και ο θεωρητικά απεριόριστος αριθμός από slaves σε ένα δίαυλο, γιατί δεν χρησιμοποιούνται διευθύνσεις για την επιλογή τους αλλά ο ακροδέκτης /SS.

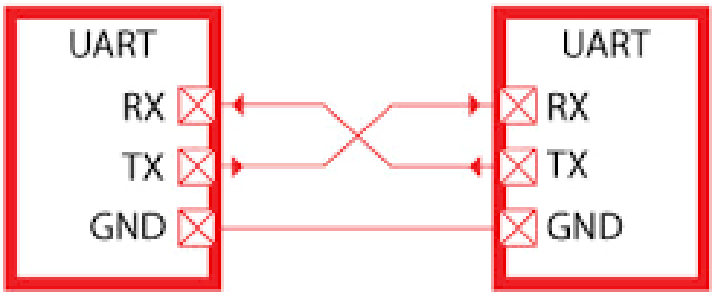


Εικόνα Θ.2: Διάγραμμα διασύνδεσης διαύλου SPI

Στην εικόνα Θ.2 παρουσιάζεται μια τοπολογία διασύνδεσης με το πρωτόκολλο SPI. Τέλος, χρησιμοποιείται για επικοινωνία με ένα μεγάλο εύρος τυποποιημένων ηλεκτρονικών όπως, ρολόγια πραγματικού χρόνου, μετατροπείς αναλογικού σήματος σε ψηφιακό, κάρτες μνήμης SD, μνήμες Flash κτλ.

Π.Θ.3.: Πρωτόκολλο επικοινωνίας UART

Το UART (universal asynchronous receiver transmitter) είναι ένα ασύγχρονο σειριακό πρωτόκολλο επικοινωνίας μεταξύ δύο ολοκληρωμένων, στο οποίο η μορφή των δεδομένων και η ταχύτητα αποστολής είναι προγραμματιζόμενα [30]. Χρησιμοποιούνται δύο ακροδέκτες για την διασύνδεση των συσκευών που το χρησιμοποιούν, όπως φαίνεται στην εικόνα Θ.3. Ο ακροδέκτης Rx (receiver) χρησιμοποιείται για την παραλαβή δεδομένων και ο ακροδέκτης Tx (transmitter) για την αποστολή τους. Τέλος υπάρχει η δυνατότητα ελέγχου της ορθότητας των δεδομένων που λαμβάνονται από κάθε συσκευή με ενεργοποίηση αποστολής του ψηφίου ομοτιμίας (parity bit).



Εικόνα Θ.3: Διασύνδεση διαύλου UART

ΠΑΡΑΡΤΗΜΑ Ι.: Εγχειρίδιο χρήστη

ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΤΗ

ΠΕΡΙΕΧΟΜΕΝΑ

Απαιτήσεις λογισμικού ηλεκτρονικού υπολογιστή του τερματικού κόμβου διαδικτύου

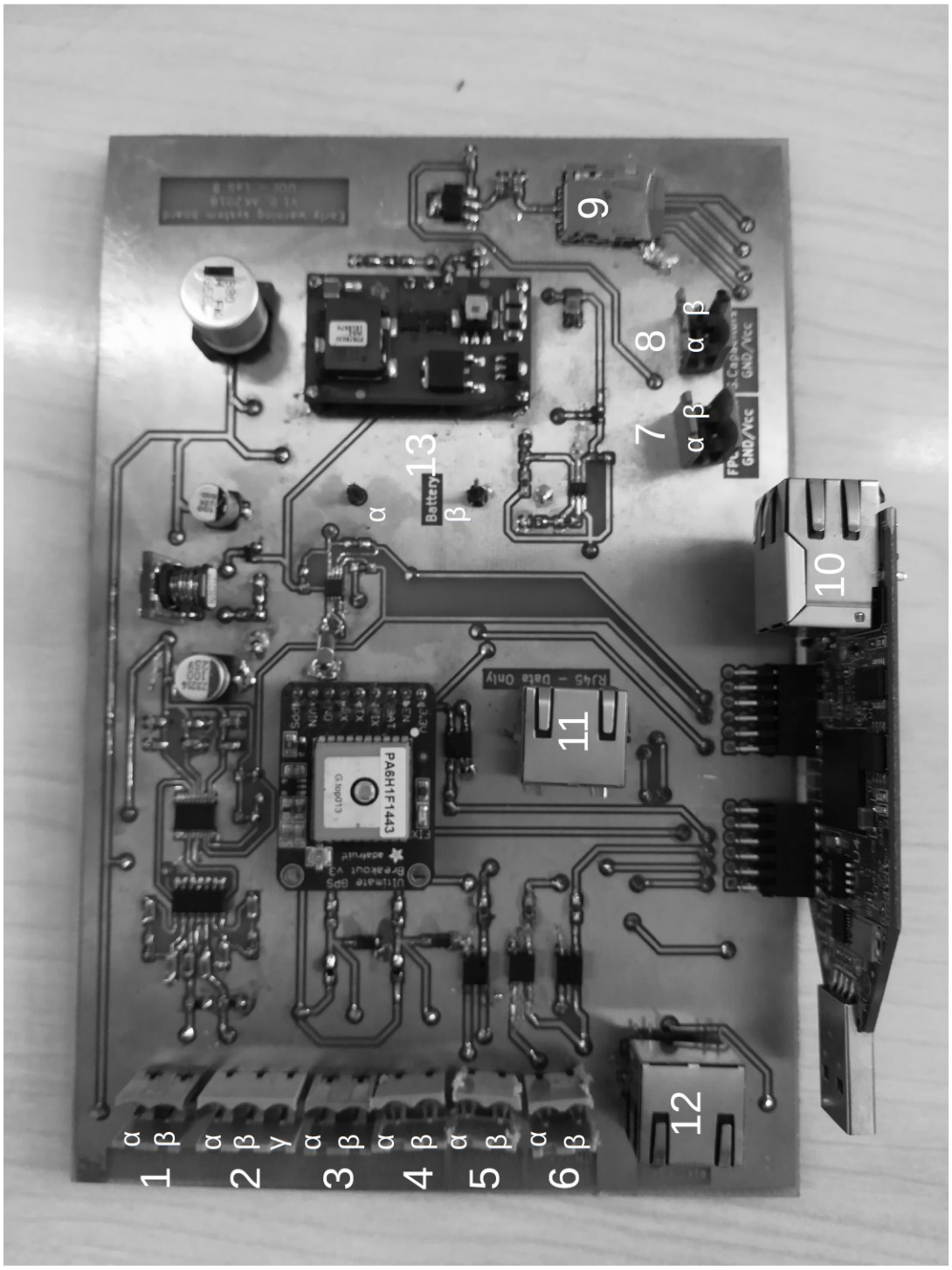
Το λειτουργικό σύστημα που πρέπει να είναι εγκατεστημένο στον ηλεκτρονικό υπολογιστή του τερματικού κόμβου διαδικτύου είναι το Microsoft Windows 7 ή νεότερη έκδοση. Επίσης απαιτείται η εφαρμογή Node.js (έκδοση 8.10 ή νεότερη) για την λειτουργία του διακομιστή Web και ο ηλεκτρονικός υπολογιστής να είναι συνδεδεμένος στο διαδίκτυο.

Απαιτήσεις λογισμικού ηλεκτρονικού υπολογιστή για παραμετροποίηση του συστήματος

Το λειτουργικό σύστημα που πρέπει να είναι εγκατεστημένο στον ηλεκτρονικό υπολογιστή είναι το Microsoft Windows 7 ή νεότερη έκδοση. Απαιτείται η εφαρμογή ISE Design Suite (έκδοση 14.7) της εταιρίας Xilinx για την παραμετροποίηση του λογισμικού του FPGA. Για την σχεδίαση των ηλεκτρονικών πλακετών του συστήματος προτείνεται η χρήση της σχεδιαστικής πλατφόρμας KiCad (έκδοση 4.0.5 ή νεότερη).

1. <u>Διασυνδέσεις ηλεκτρονικής πλακέτας συστήματος δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας</u>

Στην εικόνα Ι.1 παρουσιάζεται η επάνω όψη της ηλεκτρονικής πλακέτας που δημιουργήθηκε για το σύστημα δειγματοληψίας – μέτρησης – αποθήκευσης – μετάδοσης πληροφορίας:



Εικόνα Ι.1: Ηλεκτρονική πλακέτα συστήματος

Συνδέτης 1: Στον ακροδέκτη 1β συνδέεται το πράσινο καλώδιο του αισθητήρα του συστήματος (μέτρηση θερμοκρασίας) και στον ακροδέκτη 1α το κόκκινο καλώδιο των ηλεκτρικών βαλβίδων (τάση τροφοδοσίας 12V).
Συνδέτης 2: Στον ακροδέκτη 2α συνδέεται το μαύρο καλώδιο του αισθητήρα (γείωση), στον ακροδέκτη 2β το λευκό καλώδιο (μέτρηση αγωγιμότητας) και στον ακροδέκτη 2γ το κόκκινο καλώδιο (τάση τροφοδοσίας 12V).
Συνδέτης 3: Συνδέεται ο αισθητήρας στάθμης πλήρωσης.
Συνδέτης 4: Συνδέεται ο αισθητήρας στάθμης εκκένωσης.
Συνδέτης 5: Στον ακροδέκτη 5α συνδέεται το καφέ καλώδιο (ενεργοποίηση περιστροφής) της ηλεκτρικής βαλβίδας πλήρωσης και στον ακροδέκτη 5β το μαύρο καλώδιο (γείωση).
Συνδέτης 6: Στον ακροδέκτη 6α συνδέεται το καφέ καλώδιο (ενεργοποίηση περιστροφής) της ηλεκτρικής βαλβίδας εκκένωσης και στον ακροδέκτη 6β το μαύρο καλώδιο (γείωση).
Συνδέτης 7: Συνδέεται το καλώδιο usb micro-b που τροφοδοτεί την πλακέτα Avnet Spartan-6 LX9 Microboard. Στον ακροδέκτη 7β συνδέεται το καλώδιο που παρέχει την τάση τροφοδοσίας και στον 7α η γείωση.
Συνδέτης 8: Συνδέεται το καλώδιο που τροφοδοτεί το σύστημα με ηλεκτρική ενέργεια που είναι αποθηκευμένη στους υπερπυκνωτές. Στον ακροδέκτη 8β συνδέεται το καλώδιο που παρέχει την τάση τροφοδοσίας και στον 8α η γείωση.
Υποδοχή κάρτας μνήμης μSD 9: Τοποθετείται η κάρτα μνήμης μSD.
Υποδοχές RJ45 10-11: Συνδέεται καλώδιο UTP μεταξύ τους.
Υποδοχή RJ45 12: Συνδέεται καλώδιο UTP, που μεταφέρει δεδομένα και τροφοδοσία στην κεραία.
Υποδοχή 13: Συνδέεται η μπαταρία που τροφοδοτεί το ρολόι πραγματικού χρόνου.

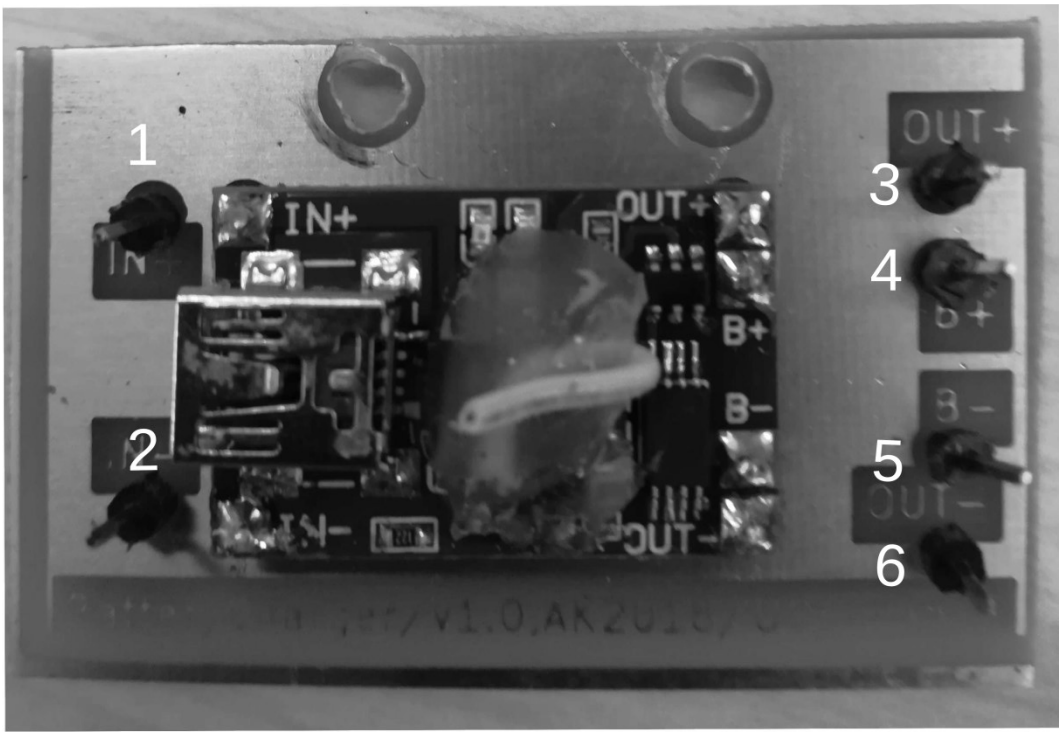
2. Διασυνδέσεις ηλεκτρονικής πλακέτας Avnet Spartan-6 LX9 Microboard

Εικόνα Ι.2: Ηλεκτρονική πλακέτα Avnet Spartan-6 LX9 Microboard

Στην εικόνα 2 φαίνεται η ηλεκτρονική πλακέτα Avnet Spartan-6 LX9 Microboard. Στην υποδοχή usb 1 (υποδοχή USB-to-JTAG) συνδέεται το λευκό καλώδιο usb που παρέχεται από την κατασκευάστρια εταιρία και το άλλο άκρο του καλωδίου συνδέεται σε ηλεκτρονικό υπολογιστή. Στην υποδοχή usb 2 (υποδοχή usb micro-b) συνδέεται το μαύρο καλώδιο που παρέχεται και το άλλο άκρο του καλωδίου συνδέεται σε ηλεκτρονικό υπολογιστή.

3. <u>Διασυνδέσεις ηλεκτρονικής πλακέτας φόρτισης μπαταρίας</u>

Στην εικόνα Ι.3 φαίνεται η ηλεκτρονική πλακέτα φόρτισης της μπαταρίας που τροφοδοτεί το ρολόι πραγματικού χρόνου.

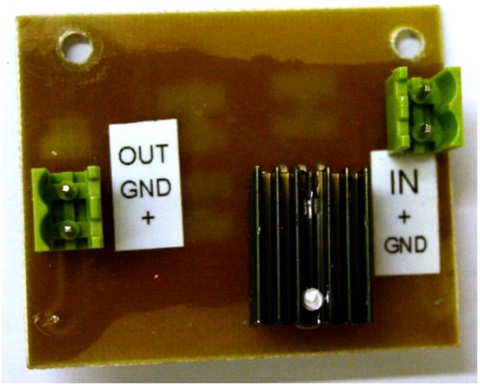


Εικόνα Ι.3: Ηλεκτρονική πλακέτα φόρτισης μπαταρίας

<u>Ακροδέκτης 1</u>: Συνδέεται στον ακροδέκτη 7α της εικόνας Ι.1.
<u>Ακροδέκτης 2</u>: Συνδέεται στον ακροδέκτη 7β της εικόνας Ι.1.
<u>Ακροδέκτης 3</u>: Συνδέεται στον ακροδέκτη 13α της εικόνας Ι.1.
<u>Ακροδέκτης 4</u>: Συνδέεται ο θετικός πόλος της μπαταρίας.
<u>Ακροδέκτης 5</u>: Συνδέεται ο αρνητικός πόλος της μπαταρίας..
<u>Ακροδέκτης 6</u>: Συνδέεται στον ακροδέκτη 13β της εικόνας Ι.1.

4. Διασυνδέσεις ηλεκτρονικής πλακέτας σταθεροποίησης τάσης φόρτισης των υπερπυκνωτών

Στην εικόνα Ι.4 φαίνεται η ηλεκτρονική πλακέτα σταθεροποίησης τάσης φόρτισης των υπερπυκνωτών.



Εικόνα Ι.4: Ηλεκτρονική πλακέτα σταθεροποίησης τάσης φόρτισης των υπερπυκνωτών

Στον ακροδέκτη IN συνδέονται τα φωτοβολταϊκά στοιχεία και στον ακροδέκτη OUT οι υπερπυκνωτές και οι ακροδέκτες 8α,β της εικόνας Ι.1.

5. Δημιουργία και προγραμματισμός MicroBlaze – Δημιουργία περιφερειακού δειγματοληψίας – Προγραμματισμός μνήμης Flash

Η διαδικασίας δημιουργίας και προγραμματισμού του μικροεπεξεργαστή MicroBlaze περιγράφεται στο παράρτημα Ε. Η δημιουργία του περιφερειακού δειγματοληψίας στο παράρτημα ΣΤ και ο προγραμματισμός της μνήμης flash στο παράρτημα Ζ.

6. Κάρτα μνήμης μSD

Πριν μία κάρτα μνήμης μSD τοποθετηθεί στο σύστημα είναι απαραίτητη η μορφοποίηση των τομέων (sectors) της ώστε να είναι συμβατή με τη λογική που αναπτύχθηκε, ακολουθώντας τα παρακάτω βήματα:

1. Αρχικά, εισάγεται σε ηλεκτρονικό υπολογιστή, που διαθέτει υποδοχή ανάγνωσης κάρτας μνήμης SD.
2. Διαγράφονται τα δεδομένα που υπάρχουν στην κάρτα μνήμης, ακολουθώντας τη διαδικασία διαμόρφωσης (format) συσκευής αποθήκευσης του λειτουργικού συστήματος που χρησιμοποιεί ο ηλεκτρονικός υπολογιστής στον οποίο τοποθετήθηκε η κάρτα μνήμης μSD.

3. Στη συνέχεια, η κάρτα τοποθετείται στην υποδοχή 9 της ηλεκτρονικής πλακέτας του συστήματος. Μέσω του προγράμματος SDK πρέπει να προγραμματιστεί το FPGA με τον παρακάτω κώδικα, που βρίσκεται στο αρχείο SDformatting.c:

```
#include "xparameters.h"
#include "xil_printf.h"
#include "xiic_l.h"
#include "xuartlite_l.h"
#include "xgpio_l.h"
#include "xil_types.h"
#include "xio.h"
#define SPI_ADDR XPAR_XPS_SPI_0_BASEADDR

int sd_init(void);
int main (void){
int k=0;
int i;
u8 data;
k=sd_init();
while (k!=1){
}
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
}
spi_send_byte(0xFE);
spi_send_byte(0x03);
for (i=0;i<511;i++){
        spi_send_byte(0x00);
}
while (data != 0x03){
data=spi_read_byte();
xil_printf("sec1 %x\n",data);
}
```

```
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x01);
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
}
spi_send_byte(0xFE);
spi_send_byte(0x02);
for (i=0;i<511;i++){
spi_send_byte(0x00);
}
while (data != 0x03){
data=spi_read_byte();
xil_printf("sec2 %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x06);
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
}
spi_send_byte(0xFE);
for (i=0;i<511;i++){
        spi_send_byte(0x00);
```

```
}
while (data != 0x03){
data=spi_read_byte();
xil_printf("sec6 %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x07);
spi_send_byte(0x00);
data=spi_read_byte();
while (data !=0x00){
}
spi_send_byte(0xFE);
for (i=0;i<511;i++){
spi_send_byte(0x00);
}

while (data != 0x03){
data=spi_read_byte();
xil_printf("sec7 %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
XIo_Out32(SPI_ADDR +0x70, 0x00);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0xff);
spi_send_byte(0x58);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x00);
spi_send_byte(0x0C);
spi_send_byte(0x00);
data=spi_read_byte();
```

```
while (data !=0x00){
}
spi_send_byte(0xFE);
for (i=0;i<511;i++){
        spi_send_byte(0x00);
}
while (data != 0x03){
data=spi_read_byte();
xil_printf("sec12 %x\n",data);
}
XIo_Out32(SPI_ADDR +0x70, 0x01);
spi_send_byte(0xff);
return 0;
}
```

Στο φάκελο της εφαρμογής που δημιουργήθηκε για την εκτέλεση του παραπάνω κώδικα πρέπει να εισαχθούν τα αρχεία init.c και spi_f.c, των οποίων ο κώδικας παρατίθεται στο παράστημα Π.Β. Ο παραπάνω κώδικας γράφει τις τιμές που φαίνονται στον πίνακα Ι.1 στους sectors αντίστοιχους τομείς της κάρτας μνήμης μSD:

Τομέας	**Τιμή**
0x00000000	0x03
0x00000001	0x02
0x00000006	0x00
0x00000007	0x00
0x0000000C	0x00

Πίνακας Ι.1: Τιμές διαμόρφωσης κάρτας μνήμης μSD

Μετά την επιτυχή εκτέλεση του παραπάνω κώδικα η κάρτα μνήμης μSD είναι έτοιμη για την αποθήκευση των μετρήσεων που λαμβάνονται από το σύστημα.

7. Διεπαφή χρήστη – Αλλαγή αισθητήρα – Προσθήκη σταθμών

Η δημιουργία της διεπαφής χρήστη περιγράφεται στο κεφάλαιο 6.3.

Για την αναπαράσταση των σωστών τιμών αγωγιμότητας και θερμοκρασίας στα διαγράμματα της διεπαφής χρήστη σε περίπτωση που γίνει αλλαγή του αισθητήρα του συστήματος, πρέπει να αλλάξουν οι γραμμές 65, 66 του κώδικα που περιέχεται στο αρχείο slicing.js. Στη γραμμή 65, στη μεταβλητή Cond εισάγεται η νέα σχέση που περιγράφει την αγωγιμότητα συναρτήσει των ADC counts και στη γραμμή 66, στη μεταβλητή tempa η νέα σχέση που περιγράφει την θερμοκρασία συναρτήσει των ADC counts.

Εάν προστεθεί σταθμός δειγματοληψίας στο σύστημα, για την προσθήκη του στη διεπαφή χρήστη πρέπει να ακολουθηθούν τα παρακάτω βήματα:

1. Δημιουργία νέου φακέλου που θα περιέχει τα αρχεία TFTPclient.bat, slicing.js καθώς και τον φάκελο Received Files.
2. Στο αρχείο TFTPclient.bat πρέπει να αλλάξει η διεύθυνση IP στη δεύτερη γραμμή του κώδικα. Η νέα διεύθυνση IP είναι η διεύθυνση IP του διακομιστή TFTP που υλοποιείται στο νέο σταθμό.
3. Στο αρχείο slicing.js πρέπει να αλλάξει ο κώδικας στις γραμμές 19-22 σε:

```
if (fs.existsSync('../DiepafiXristi/Plot/DataN.csv')) {
}else {
        fs.appendFileSync('../DiepafiXristi/Plot/DataN.csv','Date,Temp,Conductivity\
n');
```

}, όπου DiepafiXristi το όνομα του φακέλου που περιέχει τα αρχεία της διεπαφής χρήστη και στο DataN.csv, το N είναι ο αύξων αριθμός του νέου σταθμού δειγματοληψίας.

Επίσης, στη γραμμή 26 η μεταβλητή fileWriteName2 πρέπει να είναι ίση με: '../DiepafiXristi/Plot/DataN.csv'. Οι παραπάνω αλλαγές αποθηκεύουν το αρχείο DataN.csv του νέου σταθμού στο φάκελο που περιέχει τα αρχεία που χρησιμοποιούνται για τη δημιουργία των διαγραμμάτων αγωγιμότητας και θερμοκρασίας.

4. Στο αρχείο plot.js πρέπει να προστεθούν οι παρακάτω γραμμές κώδικα:

```
$('#buttonN').click(function () {
            socket.emit('messageN', 'Plot');

Plotly.d3.csv('Plot/DataN.csv', function(err, rows){

  function unpack(rows, key) {
  return rows.map(function(row) { return row[key]; });
}
var trace1 = {
  type: "scatter",
  mode: "lines+markers",
  name: 'Temp',
  x: unpack(rows, 'Date'),
  y: unpack(rows, 'Temp'),
  line: {color: '#17BECF'}
}
var trace2 = {
  type: "scatter",
  mode: "lines+markers",
  name: 'Cond1',
  x: unpack(rows, 'Date'),
  y: unpack(rows, 'Conductivity'),
  line: {color: '#FF8C00'}
```

```
}
var data2 = [trace2];
var data1 = [trace1];
var layout1 = {
  title: 'Διάγραμμα Θερμοκρασίας',
  xaxis: {
    title: 'Χρόνος (UTC+2)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#7f7f7f'
    }
  },
  yaxis: {
    title: 'Θερμοκρασία (°C)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#7f7f7f'
    }
  }
};

var layout2 = {
  title: 'Διάγραμμα αγωγιμότητας',
xaxis: {
    title: 'Χρόνος (UTC+2)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#7f7f7f'
    }
  },
  yaxis: {
    title: 'Αγωγιμότητα (μS/cm)',
    titlefont: {
      family: 'Courier New, monospace',
      size: 18,
      color: '#7f7f7f'
    }
  }
};
Plotly.newPlot('Conductivitydiv', data1, layout1);
Plotly.newPlot('Temperaturediv', data2, layout2);
```

})
})
, όπου Ν ο αύξων αριθμός του νέου σταθμού. Οι παραπάνω γραμμές κώδικα πρέπει να προστεθούν κάτω από τη γραμμή που έχει το σχόλιο: //Τέλος γραφήματος και δημιουργούν τα διαγράμματα των μετρήσεων που λαμβάνονται από το νέο σταθμό.
5. Στο αρχείο server.js εισάγεται κάτω από το σχόλιο //Νέο κουμπί:
socket.on('messageN', function (message) {
console.log('MessageN: ' + message);
});
, όπου Ν ο αύξων αριθμός του νέου σταθμού.

6. Τέλος, στο αρχείο index.html προστίθεται η παρακάτω γραμμή κώδικα κάτω το σχόλιο <!--Νέο κουμπί:
<input type="button" value="Σταθμός Ν" id="buttonN" />
,όπου Ν ο αύξων αριθμός του νέου σταθμού.