Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Μικροϋπολογιστων και Ψηφιακων Συστηματων

Αλληλεπίδραση και απομακρυσμένη παρακολούθηση συστήματος

Διπλωματικη Εργασια

του

ΑΝΑΓΝΩΣΤΟΥΛΗ ΒΑΣΙΛΕΙΟΥ

Επιβλέπων: Δημήτριος Σούντρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Μικροϋπολογιστων και Ψηφιακων Συστηματων

Αλληλεπίδραση και απομακρυσμένη παρακολούθηση συστήματος

Διπλωματικη Εργασια

του

ΑΝΑΓΝΩΣΤΟΥΛΗ ΒΑΣΙΛΕΙΟΥ

Επιβλέπων: Δημήτριος Σούντρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10 Ιουλίου 2022.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Δημήτριος Σούντρης	Παναγιώτης Τσανάκας	Σωτήριος Ξυδής
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Χ.Π.Α.

Αθήνα, Ιούλιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Μικροϋπολογιστων και Ψηφιακων Συστηματων

(Υπογραφή)

.....
Αναγνωστούλης Βασίλειος

10 Ιουλίου 2022

Περίληψη

Η δημιουργία ενός ολοκληρωμένου συστήματος για την αγορά αποτελεί ένα πολυδιάστατο πρόβλημα με δύο βασικές πτυχές (i) τον καθορισμό του τρόπου αλληλεπίδρασης του χρήστη με το σύστημα, ώστε να αποκτήσει το σύστημα λειτουργική συμπεριφορά, ενημερώνοντας κατάλληλα το χρήστη (ii) την παρακολούθηση των συνθηκών λειτουργίας των επιμέρους υποσυστημάτων, με σκοπό την προστασία και την αύξηση της αξιοπιστίας του συστήματος. Για το πρώτο πρόβλημα θα πρέπει να υπάρχει ένα υλικολογισμικό (firmware) που επιτρέπει στο χρήστη να εκκινήσει / διακόψει τη λειτουργία του συστήματος, ενώ για το δεύτερο, πέρα από το σωστό σχεδιασμό του υλικού, απαιτείται και η περιοδική παρακολούθηση δεδομένων περιβάλλοντος, ώστε να προστατευτεί η χρηστική αξία των υποσυστημάτων.

Σε αυτήν την εργασία κληθήκαμε να λύσουμε αυτά τα προβλήματα για ένα σύστημα που αποτελείται από τέσσερα διαφορετικά υποσυστήματα. Χρησιμοποιήθηκαν τέσσερις μικροελεγκτές, οι οποίοι επικοινωνούν μέσω ενός δίαυλου I^2C τόσο μεταξύ τους, όσο με εξωτερικούς αισθητήρες, ώστε να επιτευχθεί η παρακολούθηση των σωστών συνθηκών λειτουργίας του συνολικού συστήματος. Η πληροφορία αυτή, αφού συλλεχθεί από ένα μικροελεγκτή, ο οποίος δρα σαν master και είναι υπεύθυνος να προσδώσει τη λειτουργική συμπεριφορά στο σύστημα, προωθείται μέσω της σειριακής μονάδας UART σε μία ΚΜΕ, η οποία με τη σειρά της μεταφέρει την πληροφορία σε μία διαδικτυακή εφαρμογή από την οποία ενημερώνεται ο χρήστης.

Λέξεις Κλειδιά

Παρακολούθηση συστήματος, Αλληλεπίδραση με σύστημα, Προστασία συστήματος, ATxmega128D4, ATmega2560, UART, SPI, I^2C, ADC, mcp9808, ina260, MAX-M8W, cdcm6208, Microchip Studio, Buildroot, Linux kernel

Σε αυτούς που το επέτρεψαν,
σε αυτά που περάσαμε,
σε όσα έπονται.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Κατά τη σχεδίαση ενός οποιουδήποτε υπολογιστικού συστήματος, ο προσδιορισμός του τρόπου αλληλεπίδρασης και των επιτρεπτών συνθηκών λειτουργίας αποτελούν βασικές πτυχές για τη σωστή και αξιόπιστη λειτουργία του συστήματος. Στην παρούσα εργασία επιχειρείται η δημιουργία και κατασκευή ενός συστήματος, που με τη χρήση μικροελεγκτών, αισθητήρων και μίας ΚΜΕ, προσδίδεται η λειτουργική συμπεριφορά σε ένα σύστημα τηλεπικοινωνιών, προστατεύοντάς το από αλλαγές που μπορεί να προκληθούν στις συνθήκες λειτουργίας, ενημερώνοντας ο χρήστη για τις περιπτώσεις αυτές.

Αρχικά, παρουσιάζεται το θεωρητικό υπόβαθρο το οποίο αξιοποιείται στην υλοποίηση. Ξεκινώντας, ορίζονται κάποιες χρήσιμες έννοιες για την ομαλή ανάγνωση. Έπειτα, αναφέρονται τα βασικά στοιχεία της θεωρίας αναφορικά με τις σειριακές επικοινωνίες και αναλύονται τα πρωτόκολλα επικοινωνίας, τα οποία χρησιμοποιούνται στο προς υλοποίηση σύστημα παρακολούθησης.

Έπειτα, παρουσιάζονται οι λειτουργικές και μη λειτουργικές απαιτήσεις που διέπουν τη σχεδίαση μας. Με βάση τις απαιτήσεις αυτές, παρουσιάζεται η αρχιτεκτονική της προτεινόμενης λύσης. Αυτή αποτελείται από τρία διαφορετικά συστατικά: το μικροελεγκτή που διαχειρίζεται όλο το σύστημα (master), την ΚΜΕ και τους υπόλοιπους μικροελεγκτές (slave). Ο μικροελεγκτής (master) είναι υπεύθυνος για την εκκίνηση και διακοπή της λειτουργίας του συστήματος και οφείλει να συλλέξει και να προωθήσει στην ΚΜΕ όλες τις πληροφορίες από τους αισθητήρες και τους μικροελεγκτές. Οι μικροελεγκτές (slave) διαχειρίζονται και συλλέγουν πληροφορίες για το τοπικό τους υποσύστημα. Η ΚΜΕ λειτουργεί ως εξυπηρετητής, επιτρέποντας τη σύνδεση χρηστών σε μία διαδικτυακή εφαρμογή με σκοπό την ενημέρωση και τη διαχείριση του συστήματος απο την πλευρά του χρήστη.

Στη συνέχεια, παρουσιάζονται κάποια βασικά κομμάτια της ανάπτυξης του λογισμικού που αφορούν: το χειρισμό των μονάδων των μικροελεγκτών, τον προγραμματισμό των αισθητήρων και τον τρόπο ανάπτυξης μίας μινιμαλιστικής διανομής Linux που εξυπηρετεί τις ανάγκες της εφαρμογής του συστήματός μας.

Τέλος, παρουσιάζεται η διαδικτυακή εφαρμογή, δηλαδή η διεπαφή του χρήστη με το σύστημα, που αποτελεί τον τελικό στόχο της εργασίας. Η ορθή λειτουργική συμπεριφορά της εφαρμογής, προϋποθέτει τη σωστή επικοινωνία μεταξύ των μικροελεγκτών, των αισθητήρων, της ΚΜΕ και του χρήστη. Σε όλες αυτές τις περιπτώσεις, διαπιστώνεται ότι η επικοινωνία διεξάγεται ομαλά και κατά τον αναμενόμενο τρόπο.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Παρακολούθηση συστήματος

Με τον όρο παρακολούθηση συστήματος ορίζεται μία διαδικασία κατά την οποία παρέχονται στο χρήστη χρήσιμες πληροφορίες σχετικά με το σύστημα. Υπάρχουν δύο σκοπιές στην παρακολούθηση ενός συστήματος *(i)* Η παρακολούθηση του λογισμικού, *(ii)* Η παρακολούθηση του υλικού.

Στο επίπεδο του λογισμικού εμφανίζονται στο χρήστη πληροφορίες σχετικά με τις διάφορες διεργασίες που εκτελούνται στο σύστημα, όπως είναι ο όγκος δεδομένων ενός δικτύου, η χρησιμοποίηση (utilization) των πόρων του συστήματος, οι καταγραφές των προγραμμάτων (logs), το αποτύπωμα που αφήνει ένας χρήστης [1]. Στόχος όλων αυτών είναι συνήθως η αποσφαλμάτωση.

Στη δεύτερη περίπτωση, προωθείται στο χρήστη η πληροφορία της κατάστασης του περιβάλλοντος του υλικού όπως η θερμοκρασία, οι τάσεις τροφοδοσίας διαφόρων συστημάτων ή/και τα ρεύματα που το διαρρέουν, η ύπαρξη ή μη συγκεκριμένων υποσυστημάτων και περιφερειακών. Για την υλοποίηση αυτού, θα πρέπει οι μονάδες που εκτελούν την παρακολούθηση να είναι τοποθετημένες επάνω στο υλικό, ώστε να μπορούν να αλληλεπιδρούν άμεσα με αυτό. Λόγω της καταστροφικής επίπτωσης στο σύστημα από ορισμένα σφάλματα υλικού, συνήθως, ορίζεται και η λειτουργική απαίτηση, πέραν της παρακολούθησης, να δύναται το πρόγραμμα να «δράσει» για την προστασία του συστήματος. Τα προγράμματα που υλοποιούν επεξεργασία σε χαμηλό επίπεδο (επίπεδο υλικού), ονομάζονται υλικολογισμικά (firmware) και αναπτύσσονται συνεχώς στα ενσωματωμένα συστήματα.

2.2 Αλληλεπίδραση ανθρώπου-υπολογιστή

Το πάτημα ενός διακόπτη έχει σαν αποτέλεσμα τη μεταβολή στη διέλευση ηλεκτρικού ρεύματος στο κύκλωμα. Αυτό, σε μικρά κυκλώματα, είναι ικανοποιητική συνθήκη για την έναρξη και τον τερματισμό της λειτουργίας ενός κυκλώματος. Σε πολύπλοκα συστήματα, όπως ένας ηλεκτρονικός υπολογιστής, η απλοποίηση της ενεργοποίησης/απενεργοποίησης του συστήματος προσθέτει ένα επίπεδο αφαίρεσης, κατά το οποίο αποκρύπτεται η πραγματική συμπεριφορά του κυκλώματος. Πιο συγκεκριμένα, αποκρύπτεται πως κατά την εκκίνηση η ρευματοδότηση των επιμέρους μονάδων υλικού γίνεται με συγκεκριμένο ακολουθιακό τρόπο (power up/down sequence) και επιβάλλονται αυστηροί έλεγχοι στις τάσεις και στα

ρεύματα που διαρρέουν το κύκλωμα. Επίσης, αποκρύπτονται πληροφορίες όπως ο χρονισμός του κυκλώματος, ο συντονισμός των επιμέρους μονάδων και τα πρωτόκολλα επικοινωνίας αυτών ώστε να εκκινήσει ορθά το σύστημα.

Η απομακρυσμένη ένδειξη σφάλματος ενός υποσυστήματος αποτελεί μία πολύ χρήσιμη πληροφορία για την κατανόηση της λειτουργικής συμπεριφοράς ενός συστήματος για έναν απομακρυσμένο χρήστη. Η παροχή τέτοιων δεδομένων δημιουργεί ένα πολυδιάστατο πρόβλημα: το υλικό θα πρέπει να αναγνωρίσει το πρόβλημα που υπάρχει, να μεταφέρει την πληροφορία στο λογισμικό, αυτό με τη σειρά του να την προωθήσει στο δίκτυο, το λογισμικό του χρήστη να παραλάβει την πληροφορία σφάλματος, και έτσι να φτάσει στο χρήστη σε ευανάγνωστη μορφή.

Στα δύο παραδείγματα που παρουσιάστηκαν παραπάνω, υπάρχουν πολλά επίπεδα αφαίρεσης. Αποκρύπτοντας την υλοποίηση και εστιάζοντας στις ανάγκες χρήσης, μπορεί κανείς να ορίσει και να βελτιώσει τον τρόπο με τον οποίο ο χρήστης παρέχει και αντλεί πληροφορίες από ένα σύστημα. Έτσι, ορίζεται η αλληλεπίδραση ανθρώπου-υπολογιστή ως ένα σύνολο μεθόδων, συσκευών και προγραμμάτων, που έχουν ως στόχο να παρέχουν ευχρηστία στις διεπαφές (user interface) μεταξύ του συστήματος και του χρήστη και να είναι προσανατολισμένα στις ανθρώπινες ανάγκες [2].

2.3 Ενσωματωμένα συστήματα

Ενσωματωμένο σύστημα είναι ένα υπολογιστικό σύστημα ειδικά σχεδιασμένο με σκοπό να υλοποιεί πολύ συγκεκριμένες εργασίες. Το ενσωματωμένο σύστημα διαφέρει από τον υπολογιστή γενικής χρήσης, ο οποίος είναι σχεδιασμένος για να υλοποιεί ένα μεγάλο φάσμα διαφορετικών εργασιών επεξεργασίας. Είναι ενσωματωμένο, ως μέρος ενός γενικότερου ολοκληρωμένου συστήματος ή προϊόντος. Αποτελεί συνδυασμό του υλικού και του λογισμικού, αφού συνήθως διαχειρίζεται του φυσικούς πόρους του συστήματος, αλληλεπιδρώντας παράλληλα με διάφορους αισθητήρες.

Τα ενσωματωμένα συστήματα έχουν σχεδιαστεί για να εκτελούν μόνο συγκεκριμένες εργασίες. Αυτό έχει ως συνέπεια να υπάρχουν αυξημένοι περιορισμοί στους πόρους που διαχειρίζονται. Αυτοί οι περιορισμοί αφορούν το μέγεθος, αφού αποτελούν υποσυστήματα σε μεγαλύτερα συστήματα, το κόστος, καθώς κατασκευάζονται και χρησιμοποιούνται μαζικά, τη κατανάλωση ενέργειας, διότι συνήθως λειτουργούν αδιάκοπα και δίχως να ψύχονται όπως μια ΚΜΕ και την απόδοση, γιατί στις περισσότερες περιπτώσεις ο σκοπός τους είναι η αλληλεπίδραση με το χρήστη σε πραγματικό χρόνο. Οι παραπάνω περιορισμοί αποτελούν και στοιχεία που διέπουν ένα ενσωματωμένο σύστημα.

Σήμερα τα συστήματα αυτά βρίσκουν εφαρμογή σε ένα ευρύ φάσμα προϊόντων. Παραδείγματα στα οποία η ανάπτυξη ενός προϊόντος βασίζεται στα ενσωματωμένα συστήματα εντοπίζονται στους τομείς της υγείας [3], της αυτοκινητοβιομηχανίας [4], στις οικιακές ηλεκτρονικές συσκευές [5], στην αμυντική βιομηχανία [6], καθώς επίσης και σε εφαρμογές του Internet of Things (IoT) [7].

Τα σύγχρονα ενσωματωμένα συστήματα βασίζονται στη χρήση των μικροελεγκτών που αποτελούν ένα ολοκληρωμένο υπολογιστικό σύστημα, το οποίο σε ένα μόνο τσιπ ενσωματώνει μία κεντρική μονάδα επεξεργασίας, μνήμη (EEPROM, SRAM, FLASH), περιφερειακά

(UART, Timer, SPI) και ακροδέκτες για ψηφιακή και αναλογική επεξεργασία (ADC, DAC).

2.4 Αισθητήρες

Αισθητήρες ονομάζονται οι συσκευές οι οποίες είναι ικανές να ανιχνεύσουν χαρακτηριστικά του περιβάλλοντος όπως είναι η υγρασία, η θερμοκρασία, το υψόμετρο και η φωτεινότητα, με σκοπό να εξάγουν αυτή τη μέτρηση σε έναν ή πολλούς παραλήπτες. Οι παραλήπτες μπορεί να είναι είτε άνθρωποι, είτε υπολογιστικά συστήματα. Τα δεδομένα μεταδίδονται με τη μορφή αναλογικού (αναλογικοί αισθητήρες) ή διακριτού σήματος (ψηφιακοί αισθητήρες). Οι διαφορές αυτών των δύο κατηγοριών επικεντρώνονται στις διαφορές μεταξύ διακριτών και αναλογικών σημάτων. Για παράδειγμα, τα αναλογικά σήματα χρησιμοποιούνται στη μουσική και τα βίντεο, καθώς ο όγκος και η ακρίβεια της πληροφορίας ενός αναλογικού σήμα2τος είναι υψηλότερος από αυτή του διακριτού. Από την άλλη πλευρά, το διακριτό σήμα μπορεί να μεταφέρει την πληροφορία σε μεγαλύτερες αποστάσεις, δίχως ο θόρυβος να επηρεάζει τη μετάδοση. Ένα άλλο χαρακτηριστικό των διακριτών σημάτων είναι η δυνατότητα κρυπτογράφησης του σήματος, προσφέροντας έτσι μεγαλύτερη ασφάλεια. Η επιλογή των αισθητήρων εξαρτάται από την εφαρμογή και τους πόρους που καλείται ο σχεδιαστής να διαχειριστεί.

Σε ότι αφορά τις εφαρμογές των ενσωματωμένων συστημάτων, οι μικροελεγκτές σχεδόν πάντοτε διαχειρίζονται διακριτά σήματα. Παρόλα αυτά, υπάρχουν ορισμένες περιπτώσεις που ενδείκνυται η χρήση αναλογικών σημάτων, όπως για παράδειγμα η μέτρηση μίας τάσης ή της έντασης του ρεύματος που διαρρέει ένα κύκλωμα. Δημιουργείται έτσι η ανάγκη ύπαρξης ενός μετατροπέα αναλογικού-σήματος σε ψηφιακό. Για το λόγο αυτό, σε όλους τους σύγχρονους μικροελεγκτές υπάρχει η αντίστοιχη υποστήριξη από μονάδες υλικού που είναι ειδικά σχεδιασμένες για τις μετατροπές αυτές ADC (Analog to Digital Converter και DAC (Digital-to-Analog Converter).

2.5 Σειριακές επικοινωνίες

Τα ενσωματωμένα ηλεκτρονικά παίζουν έχουν σημαντική αξία για την ανάπτυξη ενός σύγχρονου συστήματος, καθώς είναι συνυφασμένα με τη διασύνδεση των διαφόρων επιμέρους ολοκληρωμένων, τα οποία προσδιορίζουν τη λειτουργική αξία του συστήματος. Για το λόγο αυτό καθίσταται αναγκαίος ο καθορισμός ενός συνόλου κανόνων (πρωτόκολλο επικοινωνίας) που εξυπηρετούν τη δυνατότητα ανταλλαγής πληροφοριών μεταξύ των μονάδων υλικού. Για την επίτευξη της επικοινωνίας αυτής έχουν αναπτυχθεί διαφορετικά πρωτόκολλα, τα οποία εντάσσονται σε δύο ευρύτερες κατηγορίες: στα πρωτόκολλα σειριακής και στα πρωτόκολλα παράλληλης επικοινωνίας.

Οι βασικές διαφορές των δύο κατηγοριών εντοπίζονται σε θέματα ταχύτητας, κόστους και πολυπλοκότητας σχεδίασης. Οι παράλληλες διασυνδέσεις προσφέρουν υψηλότερες ταχύτητες μετάδοσης, αφού σε κάθε κύκλο ρολογιού μεταφέρονται πολλαπλά bits δεδομένων. Για την επίτευξη όμως αυτού χρειάζονται περισσότερες φυσικές καλωδιώσεις, αυξάνοντας έτσι την πολυπλοκότητα και το κόστος σχεδίασης. Αντιθέτως, στις σειριακές διασυνδέσεις μεταφέρεται 1-bit πληροφορίας σε κάθε κύκλο ρολογιού, απαιτώντας το μέγιστο τέσσερα

καλώδια. Για το λόγο αυτό, σε μικροελεγκτές και περιφερειακά στα οποία οι γραμμές εισόδου/εξόδου είναι περιορισμένες και πολύτιμες, γίνεται ο συμβιβασμός στην ταχύτητα με σκοπό την εξοικονόμηση γραμμών εισόδου/εξόδου.

Τα σειριακά πρωτόκολλα χωρίζονται κι αυτά σε δύο κατηγορίες στις σύγχρονες και στις ασύγχρονες επικοινωνίες [;]. Η σύγχρονη επικοινωνία επιτυγχάνεται μέσω ενός κοινού σήματος ρολογιού που συγχρονίζει τις συσκευές. Έτσι η επικοινωνία γίνεται αξιόπιστη και απλή έχοντας ως κόστος τη χρήση ενός επιπλέον καλωδίου, του σήματος του ρολογιού. Αντιθέτως, οι ασύγχρονες επικοινωνίες δεν βασίζονται σε ένα κοινό ρολόι κατά τη μετάδοση, αλλά οι συσκευές πρέπει να συμφωνήσουν σε ένα προκαθορισμένο σύνολο από παραμέτρους για να διασφαλίσουν την αλάνθαστη μετάδοση δεδομένων.

2.5.1 Ασύγχρονη σειριακή επικοινωνία - Μονάδα UART

Όπως αναφέρθηκε παραπάνω, η ασύγχρονη σειριακή επικοινωνία καθιστά αναγκαία την ύπαρξη ενός μηχανισμού συγχρονισμού της μετάδοσης. Για το λόγο αυτό δημιουργήθηκε ένα σύνολο από μεταβλητές, που εάν προ-συμφωνηθούν από πομπό και δέκτη είναι εφικτή η μετάδοση πληροφοριών. Αυτές οι παράμετροι είναι:

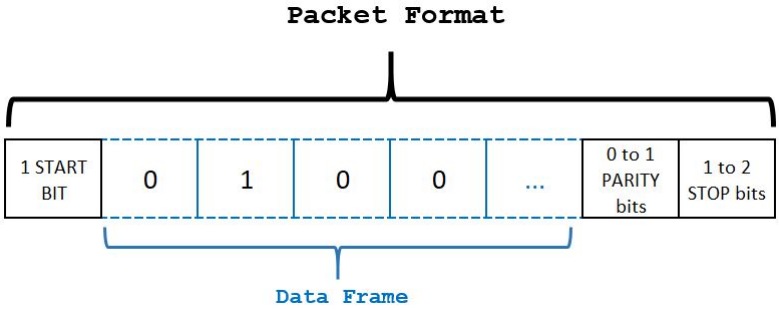
- **Η ταχύτητα μετάδοσης (baudrate)**
  Το baudrate εκφράζει το πόσο γρήγορα μεταδίδεται η πληροφορία μιας σειριακής γραμμής και συνήθως περιγράφεται σε bits per seconds (bps). Όσο μεγαλύτερη η τιμή, τόσο πιο γρήγορα μεταδίδεται η πληροφορία. Παρόλα αυτά, η δίχως σφάλματα επικοινωνία σε μεγάλες ταχύτητες δεν είναι πάντα εφικτή, καθώς υπάρχει περιορισμένο εύρος στις συχνότητες των ρολογιών των συσκευών, όπως επίσης και περιορισμένη επεξεργαστική ισχύς κατά την επεξεργασία και την αποστολή - λήψη των δεδομένων.

- **Τα bits δεδομένων (data bits)**
  Τα data bits εκφράζουν το μήκος της χρήσιμης πληροφορίας, δηλαδή τα πόσα bit μεταφέρονται στο δίαυλο. Στις μέρες μας σχεδόν πάντα χρησιμοποιούμε 8-bit πληροφορίας, παρόλα αυτά στην ιστορία υπήρχαν φορές που χρησιμοποιήθηκαν 5-bit και 7-bit πληροφορίας. Η τιμή που μπορεί να πάρει είναι από 5 έως 9 bits.

- **Το bit ισοτιμίας (parity bit)**
  Το parity bit αποτελεί ένα μηχανισμό εντοπισμού λαθών κατά τη μετάδοση και αποτελεί μία ειδική μορφή του cyclic redundancy check (CRC). Η χρήση αυτού του μηχανισμού είναι προαιρετική, καθώς επιβραδύνεται ο χρόνος αποστολής και λήψης της πληροφορίας λόγω του υπολογισμού και της επαλήθευσης του parity bit αντίστοιχα. Για την παραγωγή του parity bit ο αποστολέας προσθέτει όλα τα data bits και ανάλογα με το εάν το αποτέλεσμα είναι άρτιος ή περιττός αριθμός, το parity bit παίρνει τιμή '0' ή '1' αντίστοιχα (even parity bit) και το μεταδίδει μαζί με το πακέτο δεδομένων. Ο παραλήπτης με τη σειρά του υπολογίζει το άθροισμα όλων των data bit του πακέτου, υπολογίζει το parity bit με την ίδια μέθοδο, συγκρίνει τις τιμές με το parity bit του αποστολέα και εάν η τιμή αυτή διαφέρει από το δικό του αποτέλεσμα, γνωρίζει ότι υπήρξε σφάλμα στη μετάδοση του πακέτου.

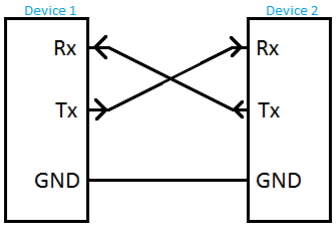
- **Τα bits συγχρονισμού (start bit, stop bit)**
  Τα start και stop bits σηματοδοτούν την έναρξη και τον τερματισμό μιας μετάδοσης πληροφορίας. Όταν το κανάλι είναι σε κατάσταση αδράνειας (idle), δηλαδή δεν μεταφέρει δεδομένα, τότε βρίσκεται σε λογικό '1'. Η μετάδοση ξεκινά με ένα start bit, το οποίο βρίσκεται σε λογικό '0' και ολοκληρώνεται με ένα ή δύο stop bits να βρίσκονται σε λογικό '1'.

Συνοψίζοντας τα παραπάνω, η συνολική πληροφορία που αποστέλλεται αποτελείται από πακέτα τα οποία έχουν προσαρτημένα τη χρήσιμη πληροφορία, τα start/stop bits και το parity bit. Στο σχήμα 2.1 φαίνεται η δομή ενός τέτοιου πακέτου.



Σχήμα 2.1: *Δομή πακέτου ασύγχρονης σειριακής επικοινωνίας*

Υλοποίηση της σειριακής επικοινωνίας αποτελούν τα κυκλώματα UART - Asynchronous Receiver-Transmitter. Η λειτουργία των κυκλωμάτων UART βασίζεται σε δύο ακροδέκτες Rx και Tx (Σχήμα 2.2), οι οποίοι αποτελούν τις γραμμές λήψης και εκπομπής των πακέτων αντίστοιχα. Τα κυκλώματα αυτά μπορεί να διαφέρουν για κάθε χρήση, τηρούν όμως τις βασικές αρχές που περιγράφηκαν παραπάνω.



Σχήμα 2.2: *Διασύνδεση UART επικοινωνίας μεταξύ δύο συσκευών*

2.5.2 SPI

Το SPI (Serial Peripheral Interface) [8] είναι ένας δίαυλος σύγχρονής σειριακής επικοινωνίας, που επιτρέπει πολλαπλά ψηφιακά ολοκληρωμένα που λειτουργούν ως slaves να επικοινωνούν με μία συσκευή τύπου master. Ο master παράγει το σήμα του ρολογιού που συγχρονίζει τη μεταφορά των δεδομένων. Κατά την επικοινωνία μεταξύ ενός master και ενός slave είναι εφικτή η ταυτόχρονη λήψη και αποστολή πληροφοριών και από τις δύο συσκευές (full-duplex). Αυτό είναι εφικτό, καθώς οι δύο συσκευές διασυνδέονται με 4 ακροδέκτες:

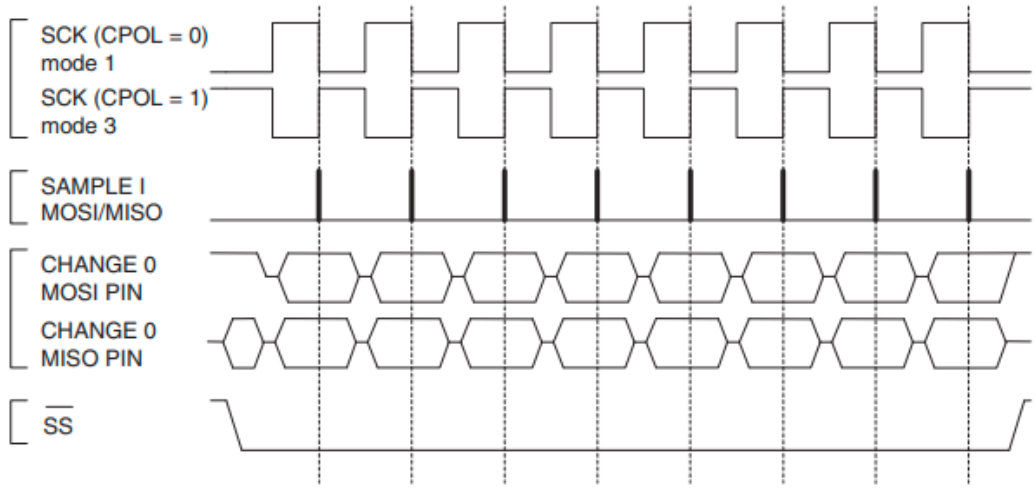
- **SCLK (Serial Clock)**: Το σήμα του ρολογιού που παράγει η master συσκευή και συγχρονίζει την επικοινωνία.
- **CS (Chip Select)**: Θέτοντας το σήμα αυτό σε χαμηλή λογική στάθμη, καθορίζεται ποια slave συσκευή επιλέγεται από τον master για ανταλλαγή δεδομένων.
- **MISO (Master In / Slave Out)**: Η γραμμή στην οποία η master συσκευή δέχεται δεδομένα από τη slave συσκευή.
- **MOSI (Master Out / Slave In)**: Η γραμμή στην οποία η master συσκευή στέλνει δεδομένα προς τη slave συσκευή.

Εκτός από την ταχύτητα ρολογιού, η master συσκευή έχει τη δυνατότητα να επιλέξει τη φάση (CPHA) και την πόλωση (CPOL) του σήματος του ρολογιού. Έχουμε έτσι τις εξής τέσσερις λειτουργίες (SPI modes) για τη ρύθμιση της γεννήτριας χρονισμού, όπως περιγράφονται στον πίνακα 2.1. Τα σήματα των ακροδεκτών για τα διαφορετικά SPI modes φαίνονται στα Σχήματα 2.3 , 2.4. Για την ορθή επικοινωνία master και slave θα πρέπει να έχουν κοινό SPI mode.

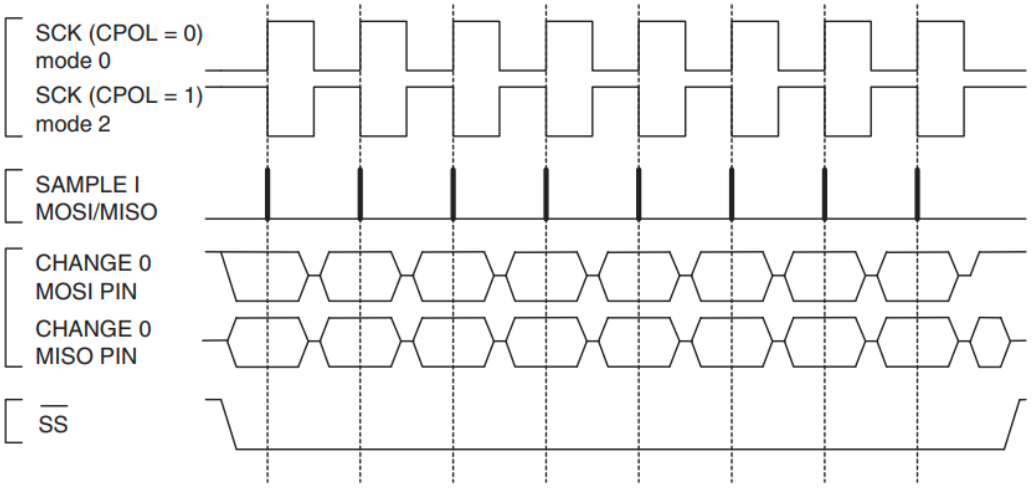
SPI Mode	CPHA	CPOL	Δειγματοληψία
0	0	0	Ακμή ανόδου στην αρχή του παλμού
1	0	1	Ακμή καθόδου στην αρχή του παλμού
2	1	0	Ακμή καθόδου στο τέλος του παλμού
3	1	1	Ακμή ανόδου στο τέλος του παλμού

Πίνακας 2.1: *Δειγματοληψία MISO/MOSI για τα διάφορα SPI Modes*

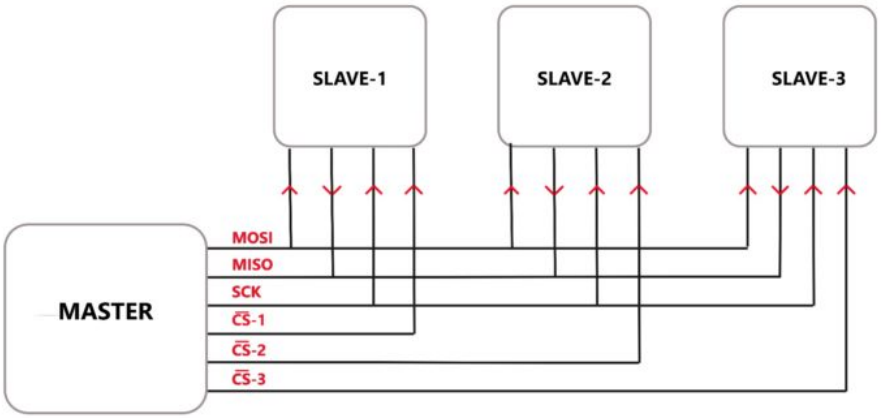
Το πρωτόκολλο επικοινωνίας SPI χρησιμοποιείται πολύ συχνά, καθώς υποστηρίζεται από πολύ απλές μονάδες υλικού και επιτρέπει μεγάλες ταχύτητες (της τάξης των δεκάδων MHz) ταυτόχρονης, αμφίδρομης επικοινωνίας και μετάδοση ωφέλιμης πληροφορίας μήκους μεγαλύτερης των 8-bit. Παρόλα αυτά, απαιτεί μεγαλύτερη καλωδίωση από τα υπόλοιπα πρωτόκολλα, όπως φαίνεται και στο Σχήμα 2.5, όπου φαίνεται πως η πολυπλοκότητα σχεδίασης και μεγέθους του PCB είναι ανάλογα ποσά με την αύξηση των slave συσκευών στο σύστημα. Ένα άλλο μειονέκτημα του SPI σε σχέση με τα υπόλοιπα πρωτόκολλα, αποτελεί το γεγονός πως δεν υπάρχει μηχανισμός ανίχνευσης λαθών και επαναποστολής κατά τη μετάδοση. Παρόλα αυτά, προσφέρει μεγαλύτερη αξιοπιστία σε πολύ μεγάλες ταχύτητες κατά τη μετάδοση και αυτό το καθιστά χρήσιμο εργαλείο κατά το σχεδιασμό ενός σύγχρονου συστήματος στο οποίο η γρήγορη απόκριση είναι βασική απαίτηση.



Σχήμα 2.3: *Μεταφορά δεδομένων στα SPI Modes 1 , 3 (CPHA - 1)*



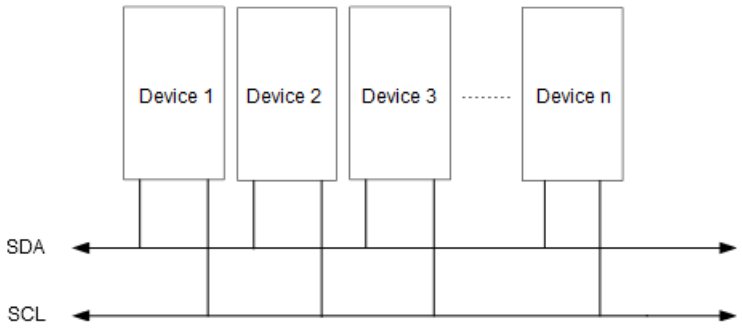
Σχήμα 2.4: *Μεταφορά δεδομένων στα SPI Modes 0 , 2 (CPHA - 0)*



Σχήμα 2.5: *Συνδεσμολογία ενός SPI master με 3 slave συσκευές*

2.5.3 I²C

Το I²C (Inter-Integrated Circuit) [9] είναι ένας δίαυλος σύγχρονης σειριακής επικοινωνίας, που κάνοντας χρήση μόνο δύο καλωδίων (SCL, SDA), επιτρέπει πολλαπλά ψηφιακά ολοκληρωμένα που λειτουργούν ως slaves να επικοινωνούν με μία ή περισσότερες συσκευές τύπου master (Σχήμα 2.6).



Σχήμα 2.6: *Διασύνδεση I²C διαύλου*

Ο δίαυλος I²C εξοικονομεί χώρο και μειώνει το γενικό κόστος. Η χρήση μόνο δύο γραμμών μειώνει τη συνολική διαδικασία αποσφαλμάτωσης και ευνοεί την ύπαρξη μικρότερου PCB. Σε ένα σύστημα που αναπτύσσεται συνεχώς, οι συσκευές I²C μπορούν εύκολα να προστεθούν ή να αφαιρεθούν χωρίς να δημιουργηθούν προβλήματα στο υπόλοιπο σύστημα. Για τους λόγους αυτούς χρησιμοποιείται ευρέως στα ενσωματωμένα συστήματα.

Ο γενικός τρόπος λειτουργίας είναι ο εξής: Η master συσκευή παράγει το σήμα του ρολογιού που συγχρονίζει την επικοινωνία και είναι υπεύθυνη για την έναρξη και τον τερματισμό μίας μεταφοράς δεδομένων. Η slave συσκευή μπορεί να ελέγξει το σήμα του ρολογιού μόνο για να καθυστερήσει την επικοινωνία (clock stretching), ώστε να υλοποιήσει τυχόν υπολογισμούς πριν τη μετάδοση της πληροφορίας. Καθε slave συσκευή έχει μία διεύθυνση μήκους 7 ή 10 bit, την οποία ο master χρησιμοποιεί για να αναφερθεί μοναδικά. Εκτός από την επιλογή της συσκευής, ο master καλείται να επιλέξει και μία από τις δύο διαδικασίες προς εκτέλεση: ανάγνωση ή εγγραφή (half-duplex). Έπειτα, τα δεδομένα μεταφέρονται σε μορφή byte με μετάδοση πρώτα του πιο σημαντικού ψηφίου. Στην περίπτωση που υπάρχουν περισσότερες από μία συσκευή τύπου master, στο δίαυλο εφαρμόζονται τεχνικές διαιτησίας (arbitration) και συγχρονισμού των ρολογιών μεταξύ των masters.

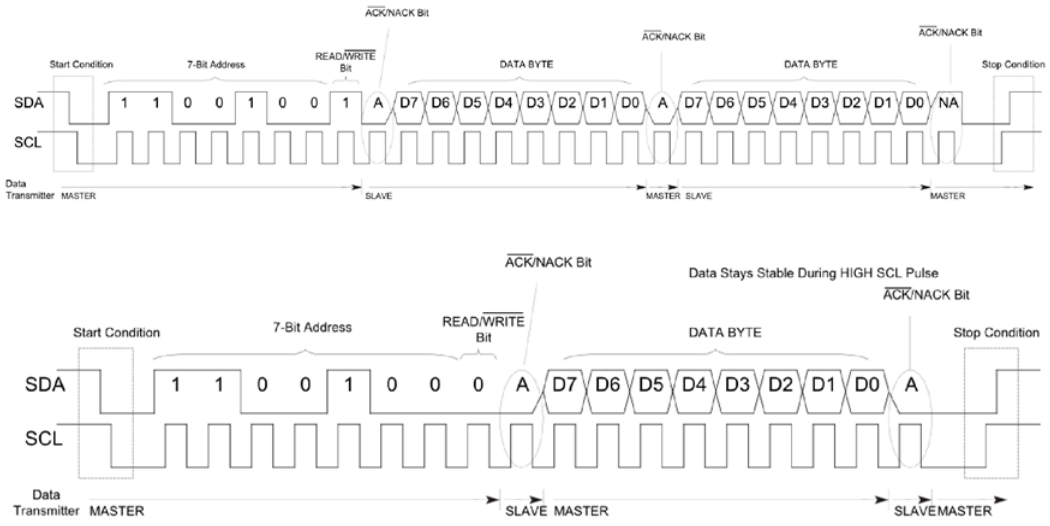
Οι ταχύτητες μετάδοσης που υποστηρίζονται είναι 100KBps (Standard mode), 400KBps (Fast mode), 3,4 MBps (High-Speed mode) και 5MBps (I²C Ultra Fast-Mode). Ωστόσο, η επικοινωνία σε Ultra Fast Mode διαφέρει σημαντικά από τις υπόλοιπες.

Η επικοινωνία μέσω του I²C είναι πολυπλοκότερη σε σχέση με τις UART και SPI. Η σηματοδότηση των SCL και SDA πρέπει να τηρεί ένα συγκεκριμένο πρωτόκολλο για τις συσκευές του δίαυλου, ώστε να αναγνωρίζεται η έγκυρη επικοινωνία. Ορίζονται έτσι οι παρακάτω καταστάσεις:

- **START**: Η έναρξη μίας μεταφοράς δεδομένων. Ορίζεται από τον master, ο οποίος έχοντας το σήμα SCL σε υψηλή στάθμη εναλλάσσει το σήμα SDA από υψηλή σε χαμηλή στάθμη.

- **ADDR**: Η διεύθυνση της συσκευής με την οποία θέλει να επικοινωνήσει ο master. Αποτελείται από την 7-bit πληροφορία που μεταφέρεται στο κανάλι SDA στους επόμενους 7 θετικούς παλμούς του σήματος ρολογιού (SCL), μετά την αποστολή του START condition.
- **R/W bit**: Η λειτουργία που θέλει να υλοποιήσει ο master (Read/Write). Για εγγραφή στη slave συσκευή το bit έχει την τιμή '0', ενώ για ανάγνωση την τιμή '1'.
- **DATA**: Η τουλάχιστον 8-bit πληροφορία που μεταδίδεται είτε από τον master προς τον slave (εγγραφή), είτε από τον slave στον master (ανάγνωση).
- **STOP**: Ο τερματισμός μίας μεταφοράς δεδομένων. Ορίζεται από τον master, ο οποίος έχοντας το σήμα SCL σε υψηλή στάθμη, εναλλάσσει το σήμα SDA από χαμηλή σε υψηλή στάθμη.
- **ACK**: Η 1-bit πληροφορία που σηματοδοτεί την επιτυχημένη μεταφορά δεδομένων. Χρησιμοποιείται και από τη slave συσκευή όταν αναγνωρίσει πως ο master αναφέρθηκε στη δική του διεύθυνση.
- **NACK**: Η 1-bit πληροφορία που σηματοδοτεί την αποτυχημένη μεταφορά δεδομένων. Χρησιμοποιείται και από τον master κατά την ανάγνωση για να γνωστοποιήσει πως δεν χρειάζεται περισσότερα δεδομένα.

Στο σχήμα 2.7 [10] φαίνονται οι μεταβολές των δύο σημάτων κατά τις δύο λειτουργίες εγγραφής και ανάγνωσης σε ένα δίαυλο I²C.



Σχήμα 2.7: *Σήματα στους ακροδέκτες ενός I²C διαύλου στις λειτουργίες ανάγνωσης (πάνω) και εγγραφής (κάτω)*

Κεφάλαιο 3

Σχεδιασμός Συστήματος

3.1 Προδιαγραφές απαιτήσεων συστήματος

Η δημιουργία ενός ολοκληρωμένου υπολογιστικού συστήματος για την αγορά αποτελεί ένα πολυδιάστατο πρόβλημα με δύο βασικές πτυχές *(i)* τον καθορισμό του τρόπου αλληλεπίδρασης του συστήματος με το χρήστη, *(ii)* την παρακολούθηση των συνθηκών λειτουργίας των επιμέρους υποσυστημάτων. Σε ένα ιδανικό περιβάλλον υπάρχουν πολύ τρόποι προσέγγισης των παραπάνω προβλημάτων, αλλά όταν πρόκειται για ένα προϊόν της αγοράς, θα πρέπει να πληρούνται αυστηρές προϋποθέσεις. Ο καθορισμός αυτών των συνθηκών υλοποιείται κατά την αρχική σχεδίαση του συστήματος και είναι δύσκολο να τροποποιηθεί μετά το στάδιο της ανάπτυξης, καθώς αυτό απαιτεί τη διαφοροποίηση του υλικού και του λογισμικού που διαχειρίζεται το σύστημα. Σκοπός του σταδίου της προδιαγραφής των απαιτήσεων είναι η λεπτομερής περιγραφή των λειτουργιών, των υπηρεσιών και των λειτουργικών περιορισμών του συστήματος.

3.1.1 Λειτουργικές απαιτήσεις

Το πρώτο στάδιο της σχεδίασης αφορά τις λειτουργικές απαιτήσεις, οι οποίες ορίζουν το πώς θα πρέπει ένα σύστημα να αντιδρά σε συγκεκριμένες εισόδους και το πώς θα πρέπει να συμπεριφέρεται σε συγκεκριμένες καταστάσεις, ορίζουν δηλαδή το λειτουργικό χαρακτήρα που διέπει το σύστημα. Στην εργασία μας, ορίζουμε τις εξής λειτουργικές απαιτήσεις:

- **Power on**: Ο χρήστης δύναται να ενεργοποιήσει το σύστημα με το πάτημα ενός κουμπιού, τοποθετημένο στην πρόσοψη του συστήματος.

- **Login**: Ο χρήστης κατά τη διάρκεια λειτουργίας του συστήματος δύναται να συνδεθεί στη διαδικτυακή εφαρμογή μέσω φυλλομετρητή, κάνοντας χρήση διαπιστευτηρίων που του παρέχονται. Δύναται επίσης να συνδεθεί μέσω του πρωτοκόλλου Secure Socket Shell στο μηχάνημα που αποτελεί τον εξυπηρετητή της εφαρμογής (server), κάνοντας χρήση των διαπιστευτηρίων του διαχειριστή του συστήματος (root credentials).

- **Power off**: Ο χρήστης δύναται να απενεργοποιήσει το σύστημα με το παρατεταμένο πάτημα ενός κουμπιού, τοποθετημένο στην πρόσοψη του συστήματος. Εάν ο χρήστης είναι εξουσιοδοτημένος μπορεί να απενεργοποιήσει το σύστημα είτε μέσω διαδικτυακής εφαρμογής, είτε μέσω του τερματικού.

- **Reboot**: Ο εξουσιοδοτημένος χρήστης δύναται να επανεκκινήσει το σύστημα μέσω της διαδικτυακής εφαρμογής ή μέσω του τερματικού.

- **Ενημέρωση κατάστασης συστήματος μέσω LED**: Ο μη απομακρυσμένος χρήστης ενημερώνεται για την κατάσταση που βρίσκεται το σύστημα μέσω System LED στην πρόσοψη του συστήματος.

- **Παρακολούθηση συστήματος**: Ο εξουσιοδοτημένος χρήστης δύναται καθ' όλη τη διάρκεια λειτουργίας του συστήματος να ενημερώνεται σχετικά με τη θερμοκρασία των υποσυστημάτων, καθώς επίσης και για τις τάσεις και τα ρεύματα που διαρρέουν τα κυκλώματά τους. Τα αποτελέσματα της παρακολούθησης εμφανίζονται στη διαδικτυακή εφαρμογή.

- **Προστασία συστήματος**: Το σύστημα παρακολούθησης δύναται να κλείσει τις τοπικές τροφοδοσίες των υποσυστημάτων, με σκοπό να προστατεύσει το υλικό στις περιπτώσεις που η θερμοκρασία, οι τάσεις ή τα ρεύματα είναι εκτός των επιτρεπτών ορίων καλής λειτουργίας των ολοκληρωμένων, όπως περιγράφονται στα εγχειρίδια χρήσης.

- **Ενημέρωση σφάλματος - εξουσιοδοτημένος χρήστης**: Ο εξουσιοδοτημένος χρήστης ενημερώνεται ασύγχρονα για πιθανή δυσλειτουργία στο σύστημα, που προκύπτει από την παρακολούθηση του συστήματος μέσω της διαδικτυακής εφαρμογής.

- **Ενημέρωση σφάλματος - μη απομακρυσμένος**: Ο μη απομακρυσμένος χρήστης στην περίπτωση σφάλματος ενημερώνεται μέσω του Alarm LED που είναι τοποθετημένο στην πρόσοψη του συστήματος.

- **Ενημέρωση ασφαλείας**: Ο εξουσιοδοτημένος χρήστης ενημερώνεται όταν ανιχνευθεί παραβίαση του συστήματος. Η παραβίαση αφορά το φυσικό μέρος του συστήματος.

- **Ενημέρωση τοποθεσίας**: Ο εξουσιοδοτημένος χρήστης δύναται να λάβει την πληροφορία που μεταδίδεται μέσω του GPS ώστε να γνωρίζει τη γεωγραφική θέση του συστήματος.

3.1.2 Μη λειτουργικές απαιτήσεις

Έπειτα καθορίζονται οι λειτουργικοί περιορισμοί που αποτελούν μέρος των μη λειτουργικών απαιτήσεων και των απαιτήσεων πεδίου. Οι μη λειτουργικές απαιτήσεις ορίζουν τους περιορισμούς και τις ιδιότητες ενός συστήματος, όπως για παράδειγμα τις απαιτήσεις για γρήγορο χρόνο απόκρισης και τον περιορισμό στις δυνατότητες των συσκευών εισόδου-εξόδου. Οι απαιτήσεις πεδίου προέρχονται από το πεδίο εφαρμογής του συστήματος και περιγράφουν χαρακτηριστικά και δυνατότητες του συστήματος, οι οποίες αντανακλούν το πεδίο. Στη συνέχεια αναλύονται οι προαναφερθείσες απαιτήσεις:

- **Ενεργειακή κατανάλωση**: Τα ολοκληρωμένα του συστήματος θα πρέπει να έχουν χαμηλή ενεργειακή κατανάλωση, δεδομένου ότι το σύστημα μπορεί να βρίσκεται σε μεταβλητό περιβάλλον και να τροφοδοτείται από μπαταρία.

- **Θερμοκρασία**: Το σύστημα θα πρέπει να μην αναπτύσσει μεγάλη θερμοκρασία, τόσο για την προστασία, όσο και για την καλύτερη απόδοση του συστήματος. Οι αποδεκτές θερμοκρασίες περιβάλλοντος ορίζονται από -30°C έως +65°C.

- **Ακρίβεια μετρήσεων**: Τα ολοκληρωμένα θα πρέπει να παρέχουν μετρήσεις υψηλής ακρίβειας με σκοπό να διασφαλιστεί η ασφάλεια του συστήματος κατά την παρακολούθηση.

- **Χρόνος απόκρισης**: Το σύστημα θα πρέπει να έχει υψηλούς χρόνους απόκρισης καθώς γίνεται χρήση από ανθρώπους.

- **Modularity**: Τα επιμέρους υποσυστήματα θα πρέπει να αποτελούν αυτόνομες μονάδες που θα λειτουργούν ανεξάρτητα με τη διασύνδεση μεταξύ τους. Έτσι, θα είναι εφικτή η δημιουργία ενός εύκολα διαμορφώσιμου συστήματος που θα ανταποκρίνεται στις ανάγκες του πελάτη.

- **Απόδοση**: Το σύστημα θα πρέπει να σχεδιαστεί βέλτιστα, με τρόπο που να μη σπαταλάει επεξεργαστικούς πόρους, υλικά και ενέργεια προκειμένου να υλοποιήσει τις λειτουργικές απαιτήσεις.

- **Ευχρηστία**: Ο χρήστης θα μπορεί να κάνει χρήση των λειτουργιών του συστήματος εύκολα, δίχως να έχει καμία γνώση σχετικά με αυτό.

- **Αξιοπιστία**: Η λειτουργική συμπεριφορά του συστήματος θα πρέπει να παραμείνει η ίδια σε βάθος χρόνου, δηλαδή τα ολοκληρωμένα που χρησιμοποιούνται να μη φθαρούν στο διάβα του χρόνου.

- **Ανταγωνιστικές τιμές**: Το σύστημα θα πρέπει να σχεδιαστεί βέλτιστα, κρατώντας χαμηλό το κόστος της πρώτης ύλης με σκοπό να μπορεί να είναι ανταγωνιστικό στην αγορά προϊόντων.

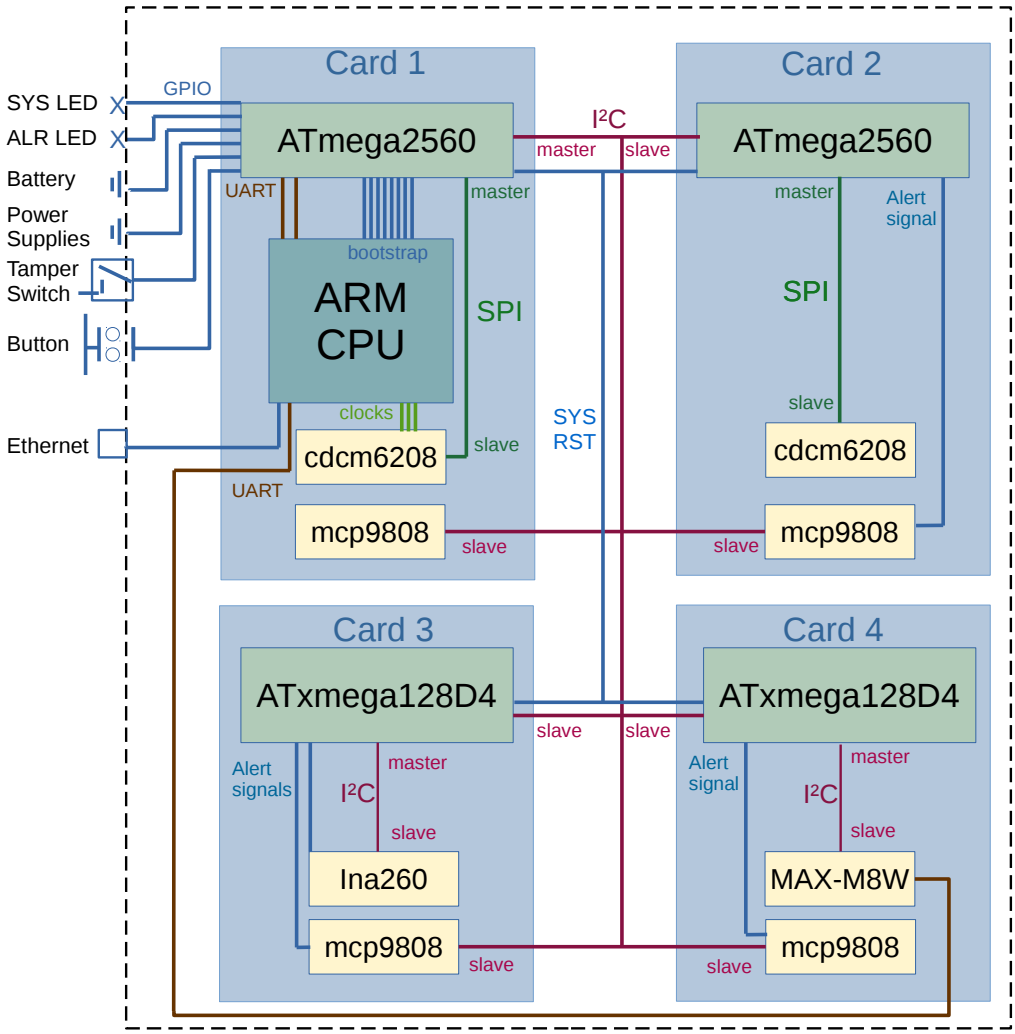
- **Συμβατότητα**: Τα ολοκληρωμένα που θα χρησιμοποιηθούν στις υλοποιήσεις, θα πρέπει να είναι συμβατά μεταξύ τους, να έχουν κοινά πρωτόκολλα επικοινωνίας και να μπορούν τα σήματα ρολογιού να λειτουργούν σε ίδιες συχνότητες.

- **Αξιοποίηση αποθέματος**: Είναι θεμιτό τα συστατικά μέρη του συστήματος να υπάρχουν ήδη σε απόθεμα, ώστε να πραγματοποιηθεί γρηγορότερη η ανάπτυξη του υλικού και να μειωθεί το κόστος και ο χρόνος απόκτησης νέων ολοκληρωμένων.

3.2 Αρχιτεκτονική

Έχοντας ως στόχο την υλοποίηση των λειτουργικών απαιτήσεων και με γνώμονα τις μη λειτουργικές απαιτήσεις, προτείνεται η αρχιτεκτονική που παρουσιάζεται στο σχήμα 3.1.

Στο σχήμα παρατηρούμε πως το υπό μελέτη σύστημα αποτελείται από τέσσερα διαφορετικά υποσυστήματα, τα οποία από εδώ και στο εξής θα αναφέρονται ως κάρτες. Ο λόγος ύπαρξης τεσσάρων διαφορετικών καρτών δεν εμπίπτει στα πλαίσια ενδιαφέροντος της διπλωματικής και θεωρείται ως δεδομένο, αφού αποτελεί αποτέλεσμα της σχεδίασης του υλικού.



Σχήμα 3.1: *Σχεδιάγραμμα του προτεινόμενου συστήματος*

Τα στοιχεία που αναλύονται στην παρούσα διπλωματική είναι όλα όσα αφορούν τη συνδεσμολογία και λειτουργία του υλικού, καθώς επίσης και την ευχρηστία αλληλεπίδρασής του με το χρήστη.

Σημειώνεται πως *(i)* οι κάρτες είναι διαφορετικές μεταξύ τους, άρα απαιτείται υλοποίηση διαφορετικών κύκλων λειτουργίας FSM για το εκάστοτε υλικό, *(ii)* οι κάρτες θα πρέπει να λειτουργούν αυτόνομα, δίχως να αποτελούν μέρη του συνολικού συστήματος. Για τους λόγους αυτούς κάθε κάρτα έχει το δικό της μικροελεγκτή που είναι υπεύθυνος για διάφορες λειτουργίες, όπως η παροχή και η διακοπή της τροφοδοσίας στην κάρτα, η συλλογή δεδομένων των τάσεων, της θερμοκρασίας και της έντασης του ρεύματος και η προστασία της κάρτας στις περιπτώσεις που οι συνθήκες λειτουργίας δεν είναι οι επιθυμητές.

Εκτός από τους κύκλους λειτουργίας των επιμέρους καρτών, για να λειτουργήσει το σύστημα θα πρέπει να τηρηθεί μία σειρά εκκίνησης των καρτών, δηλαδή θα πρέπει να υπάρχει μία κάρτα «συντονιστής». Δεδομένης της ανάγκης αυτής και θέλοντας η πληροφορία που αποκτάται από όλες τις κάρτες να μη φτάνει με άναρχο τρόπο στο χρήστη, προτιμήθηκε

το μοντέλο του Master-Slave για την υλοποίηση της επικοινωνίας του υλικού. Καθώς η ανάπτυξη υλικού επιβάλει τη συνεχή προσθήκη νέων στοιχείων ή την εξαγωγή παλαιών, επιλέχθηκε το I^2C σαν δίαυλος διασύνδεσης των καρτών και αισθητήρων, έτσι ώστε να μη δημιουργείται κανένα πρόβλημα στο υπόλοιπο σύστημα.

Η τοποθέτηση θερμομέτρων υψηλής ακρίβειας σε σημεία όπου προβλέπεται αυξημένη θερμότητα αποτελεί ένα εργαλείο για την απόκτηση μετρήσεων που αφορά το υλικό. Την ανάγκη αυτή, σε συνδυασμό με τις απαιτήσεις της ενεργειακής κατανάλωσης, της υψηλής απόκρισης και αντοχής, καλύπτει η σχεδιαστική επιλογή του αισθητήρα mcp9808 [11].

Η μέτρηση του ρεύματος μέσω της μονάδας ADC των μικροελεγκτών αποτελεί ένα σύνθετο πρόβλημα από τη σκοπιά της σχεδίασης υλικού και η λύση του δεν αποτελεί καλή πρακτική. Σαν λύση θα προσθέταμε μία πολύ μικρή αντίσταση σε σειρά με το φορτίο που θέλουμε να μετρήσουμε. Έπειτα, θα προσθέταμε στα άκρα της αντίστασης έναν πολύ χαμηλού θορύβου ενισχυτή για να μετατρέψει το διερχόμενο ρεύμα σε τάση, τάση η οποία θα φτάσει στους ακροδέκτες του μικροελεγκτή. Όπως γίνεται αντιληπτό, αυτό το κύκλωμα έχει πολύ μεγάλο κόστος, καταναλώνει πολύ χώρο και δεν παρέχει καμία ενημέρωση για τιμές κατωφλίου. Αντί αυτού, χρησιμοποιούμε τον αισθητήρα ina260 [12] για μετρήσεις υψηλής ακρίβειας της τάσης και του ρεύματος το οποίο είναι εύκολα προγραμματίσιμο.

Για να μπορεί ο χρήστης να έχει ακριβή μέτρηση του χρόνου στο σύστημα, ακόμη και όταν αυτό είναι κλειστό, απαιτείται η ύπαρξη ενός Real Time Clock (RTC). Για να μπορέσει να τροφοδοτηθεί το RTC, χρειάζεται εξωτερική τροφοδοσία από μία μπαταρία, όπως συμβαίνει και στους προσωπικούς υπολογιστές. Έτσι τοποθετήθηκε η μπαταρία στο σύστημα και προστέθηκε μία ακίδα που επιτρέπει τη μέτρηση της τάσης αυτής από την κάρτα συντονιστή, ώστε ο χρήστης να γνωρίζει το πότε αυτή τείνει να αποφορτιστεί. Για τη μέτρηση της τάσης, οι μικροελεγκτές θα πρέπει να ενσωματώνουν τη μονάδα μετατροπής αναλογικού σήματος σε ψηφιακό Analog to Digital Converter (ADC).

Για να μπορέσει ο χρήστης να συνδεθεί στη διαδικτυακή εφαρμογή και να αλληλεπιδράσει με το σύστημα, θα πρέπει να υπάρχει ένας server που να εξυπηρετεί τα αιτήματά του. Για τον παραπάνω λόγο επιλέχθηκε μία CPU αρχιτεκτονικής ARM. Κατά την εκκίνηση η CPU θα πρέπει να γνωρίζει σε ποια μνήμη θα αναζητήσει τον bootloader, ώστε να εκκινήσει το λειτουργικό σύστημα. Η μετάδοση αυτής της πληροφορίας γίνεται μέσω της κατάλληλης προσαρμογής των bootstrap pins. Για την εύκολη και ευέλικτη επιλογή μνήμης, συνδέσαμε τους ακροδέκτες του μικροελεγκτή με τα bootstrap pins της CPU. Επίσης, για τη λειτουργία της η CPU χρειάζεται υψηλών συχνοτήτων διαφορικά σήματα ρολογιού, τα οποία παρέχονται από τον clock generator cdcm6208 [13]. Η ανάγκη αυτή προκύπτει από την απαίτηση του κατασκευαστή να μειωθεί ο θόρυβος και να αυξηθεί η ακεραιότητα του σήματος του ρολογιού. Η επιλογή του συγκεκριμένου ολοκληρωμένου έγινε λόγω ευκολίας διαθεσιμότητας και συμβατότητας με την KME. Ενώ δίνεται η επιλογή για τον προγραμματισμό μέσω I^2C, διαλέξαμε το SPI για λόγους ταχύτητας.

Για την υλοποίηση της απαίτησης της ενημέρωσης ασφαλείας γίνεται χρήση των tamper switches. Υπάρχουν διακόπτες τοποθετημένοι στη μέση των επιμέρους μηχανικών μερών του συστήματος και κατά την απελευθέρωσή τους διακόπτεται η τροφοδοσία της μπαταρίας και ενημερώνεται ο χρήστης σχετικά με τη φυσική παραβίαση του συστήματος. Για τη μετάδοση αυτής της πληροφορίας, καθώς επίσης και για τη μετάδοση των πληροφοριών που

αποκτώνται κατά την παρακολούθηση του συστήματος, δημιουργείται η ανάγκη χρήσης της μονάδας UART, ώστε η πληροφορία που συλλέγεται και εξάγεται να είναι ευανάγνωστη.

Οι ανάγκες χρήσης των μονάδων UART, SPI, I^2C, ADC καλύπτονται είτε από τη χρήση του ATxmega128D4 [14] [15], είτε του ATmega2560 [16], οι οποίοι είναι εύκολα διαθέσιμοι. Οι δύο μικροελεγκτές έχουν αρκετά κοινά όπως περιγράφεται στο Κεφάλαιο 4, παρόλα αυτά, η ειδοποιός τους διαφορά εντοπίζεται στον αριθμό των ακροδεκτών Εισόδου/Εξόδου, στην ταχύτητα εκτέλεσης των εντολών, στη χωρητικότητα της μνήμης, καθώς και στο κόστος αγοράς. Στις Κάρτες 1 και 2 υπάρχουν οι ανάγκες για μεγαλύτερη ταχύτητα εκτέλεσης, για ανάληψη μεγαλύτερου φόρτου εργασίας, για περισσότερες θύρες Εισόδου/Εξόδου και για μεγαλύτερα μεγέθη μνημών. Συνεπώς, επιλέχθηκε ο ATmega2560 για να ανταποκριθεί στις ανάγκες αυτών των καρτών. Αντιθέτως, στις κάρτες 3 και 4 οι λειτουργίες είναι ελάχιστες και μπορούμε να εξοικονομήσουμε χρήματα και κατανάλωση ενέργειας χρησιμοποιώντας τον ATxmega128D4.

3.3 Διαδικτυακή διεπαφή χρήστη

Οι δυνατότητες ενός χρήστη στην εφαρμογή περιγράφουν επακριβώς το σύνολο των λειτουργικών απαιτήσεων. Έτσι, κατά τη σύνδεση ένας χρήστης θα μπορεί *(i)* να αναζητήσει μέσα από ένα ευδιάκριτο μενού τις λειτουργίες απενεργοποίησης και επανεκκίνησης του συστήματος, *(ii)* να του εμφανιστούν τα δεδομένα της παρακολούθησης του συστήματος με ευανάγνωστο τρόπο κατά τον οποίο να μπορεί να διακρίνει εάν η τιμές είναι στο επιτρεπτό εύρος τιμών και *(iii)* να εμφανίζονται τα γεγονότα στα οποία το σύστημα δεν λειτουργεί κάτω από τις φυσιολογικές συνθήκες, γνωστοποιώντας την αιτία της δυσλειτουργίας.

3.4 Λειτουργίες μονάδων επεξεργασίας

Στο σημείο αυτό χρειάζεται να αναφερθεί πως η ελάχιστη και μέγιστη θερμοκρασία εντός του συστήματος ορίζεται από το λιγότερο ανεκτικό σε θερμοκρασία ολοκληρωμένο, δηλαδή τους μικροελεγκτές μας, οι οποίοι έχουν όρια λειτουργίας σύμφωνα με το εγχειρίδιο από -40°C έως +85°C.

Όπως έχει ήδη αναφερθεί, η κάθε κάρτα υλοποιεί το δικό της κύκλο λειτουργίας. Παρακάτω παρουσιάζονται εκτενέστερα και συγκεντρωτικά οι λειτουργίες της κάθε κάρτας.

3.4.1 Κάρτα 1

- **Αλληλεπίδραση με το χρήστη μέσω ενός διακόπτη ενεργοποίησης, απενεργοποίησης του συστήματος.**
  Ο χρήστης για να ενεργοποιήσει το σύστημα θα πρέπει να πατήσει το κουμπί για περισσότερο από 50msec, χρόνος που είναι ικανοποιητικός για να αποφύγουμε φαινόμενα αναπήδησης αλλά και για να θεωρηθεί έγκυρη η κίνηση του. Ο χρήστης για να απενεργοποιήσει το σύστημα θα πρέπει να πατήσει το κουμπί για περισσότερο από 5sec. Η πληροφορία της κατάστασης του συστήματος μεταβιβάζεται στο χρήστη μέσω του SYS LED της πρόσοψης, όπως περιγράφεται στον Πίνακα 3.1.

Κατάσταση LED	Κατάσταση συστήματος
Σταθερά πράσινο	Σύστημα ενεργοποιημένο
Σταθερά σβηστό	Σύστημα απενεργοποιημένο
Αναβοσβήνει πράσινο	Αναβάθμιση λογισμικού σύστηματος
Αναβοσβήνει κίτρινο	Λήφθηκε εντολή για απενεργοποίηση του συστήματος

Πίνακας 3.1: *Ερμηνεία λειτουργίας του System LED*

- **Ενεργοποίηση και απενεργοποίηση της κάρτας**.
  Η κάρτα κατά την εκκίνηση του συστήματος θα πρέπει να ενεργοποιήσει τις τοπικές τροφοδοσίες και έπειτα να βγάλει από reset state τα ολοκληρωμένα του κυκλώματος. Αυτό γίνεται σειριακά καθώς υπάρχουν αλληλοεξαρτώμενες μονάδες.

- **Ενεργοποίηση και απενεργοποίηση του συστήματος**.
  Η κάρτα κατά την εκκίνηση του συστήματος θα πρέπει να ενεργοποιήσει τις τροφοδοσίες του συνολικού συστήματος και να βγάλει από reset state τις υπόλοιπες κάρτες.

- **Εκκίνηση της CPU**.
  Κατά την εκκίνηση της CPU ο μικροελεγκτής θα πρέπει να προγραμματίσει σωστά τον clock generator (cdcm6208) μέσω SPI, ώστε να παράξει τα απαιτούμενα για την CPU διαφορικά σήματα ρολογιού. Έπειτα, θα πρέπει να ορίσει σωστά τα bootstrap pins για να γνωρίζει η CPU σε ποία ROM μνήμη θα ψάξει να βρει τον bootloader.

- **Αλληλεπίδραση με το χρήστη μέσω του τερματικού του λειτουργικού συστήματος της CPU**.
  Ο εξουσιοδοτημένος χρήστης, εκτελώντας την εντολή poweroff στο τερματικό απενεργοποιεί το λειτουργικό σύστημα, θέτει σε reset τις υπόλοιπες κάρτες και τα ολοκληρωμένα της κάρτας και απενεργοποιεί τις τροφοδοσίες όλου του συστήματος. Εκτελώντας την εντολή reset, επανεκκινείται το λειτουργικό σύστημα, μπαίνουν σε reset όλα τα ολοκληρωμένα και οι κάρτες, αλλά οι τροφοδοσίες του συστήματος παραμένουν ανοιχτές.

- **Αλληλεπίδραση με το χρήστη μέσω διαδικτυακής εφαρμογής**.
  Ο εξουσιοδοτημένος χρήστης μπορεί με το πάτημα ενός πλήκτρου στην εφαρμογή να απενεργοποιήσει και να επανεκκινήσει το σύστημα όπως περιγράφεται παραπάνω.

- **Προγραμματισμός αισθητήρων του συστήματος**
  Ο μικροελεγκτής κατά την εκκίνηση προγραμματίζει όλους τους αισθητήρες θερμοκρασίας για να έχουν ως αποδεκτά όρια τις τιμές (-40,+85)°C και στην περίπτωση μη τήρησης αυτών να παράγουν σήμα ALERT, το οποίο θα λαμβάνουν οι εκάστοτε κάρτες.

- **Παρακολούθηση κάρτας**
  Ο μικροελεγκτής καθ' όλη τη διάρκεια λειτουργίας του, παρακολουθεί την υγιή κατάσταση των τροφοδοτικών μέσω σημάτων Power Good (PG). Επιπροσθέτως, διαβάζει την τιμή της θερμοκρασίας του τοπικού θερμόμετρου και ελέγχει ότι είναι στα αποδεκτά όρια (-40,+85)°C.

- **Συσσώρευση πληροφορίας συστήματος και εξαγωγή στο χρήστη**.
  Ο μικροελεγκτής της κάρτας αφού αρχικοποιείται ως Master στο I^2C bus, συλλέγει τις πληροφορίες των τάσεων και ρευμάτων από τους υπόλοιπους slave-μικροελεγκτές και τις θερμοκρασίες από τους αισθητήρες των καρτών. Επιπλέον, συλλέγονται οι πληροφορίες για τις εκδόσεις του υλικολογισμικού και της τοπολογίας των μικροελεγκτών των καρτών. Έπειτα στέλνει τα δεδομένα σε ευανάγνωστη μορφή μέσω UART στο λειτουργικό σύστημα. Στο λειτουργικό σύστημα, ένα πρόγραμμα που εκτελείται στο παρασκήνιο, διαβάζει τις διαθέσιμες πληροφορίες της σειριακής θύρας, τις μετατρέπει σε ένα JSON string, το οποίο με τη σειρά του προωθείται στο front-end της διαδικτυακής εφαρμογής. Η διαδικασία αυτή εκτελείται περιοδικά, κάθε 5sec. Το πρόγραμμα αυτό είναι υπεύθυνο για τη μετατροπή οποιουδήποτε σήματος παραγόμενο από το χρήστη (π.χ poweroff button στο web application) σε μορφή κατανοητή για το μικροελεγκτή μέσω της σειριακής θύρας.

- **Χειρισμός σφαλμάτων**
  Στην περίπτωση ανίχνευσης εσφαλμένης τροφοδοσίας ή θερμοκρασίας εκτός ορίων στην κάρτα, ο μικροελεγκτής αφού ενημερώσει το χρήστη μέσω της σειριακής απενεργοποιεί άμεσα το σύστημα σε διάστημα μικρότερου των 2 δευτερολέπτων. Αυτή η λειτουργία είναι ζωτικής σημασίας για την προστασία του συστήματος. Σε περίπτωση ανίχνευσης τιμών εκτός ορίων από τις υπόλοιπες κάρτες, ο Master το μόνο που κάνει είναι να ενημερώσει το χρήστη μέσω της διαδικτυακής εφαρμογής και του ALR LED της πρόσοψης, όπως περιγράφεται στον Πίνακα 3.2.

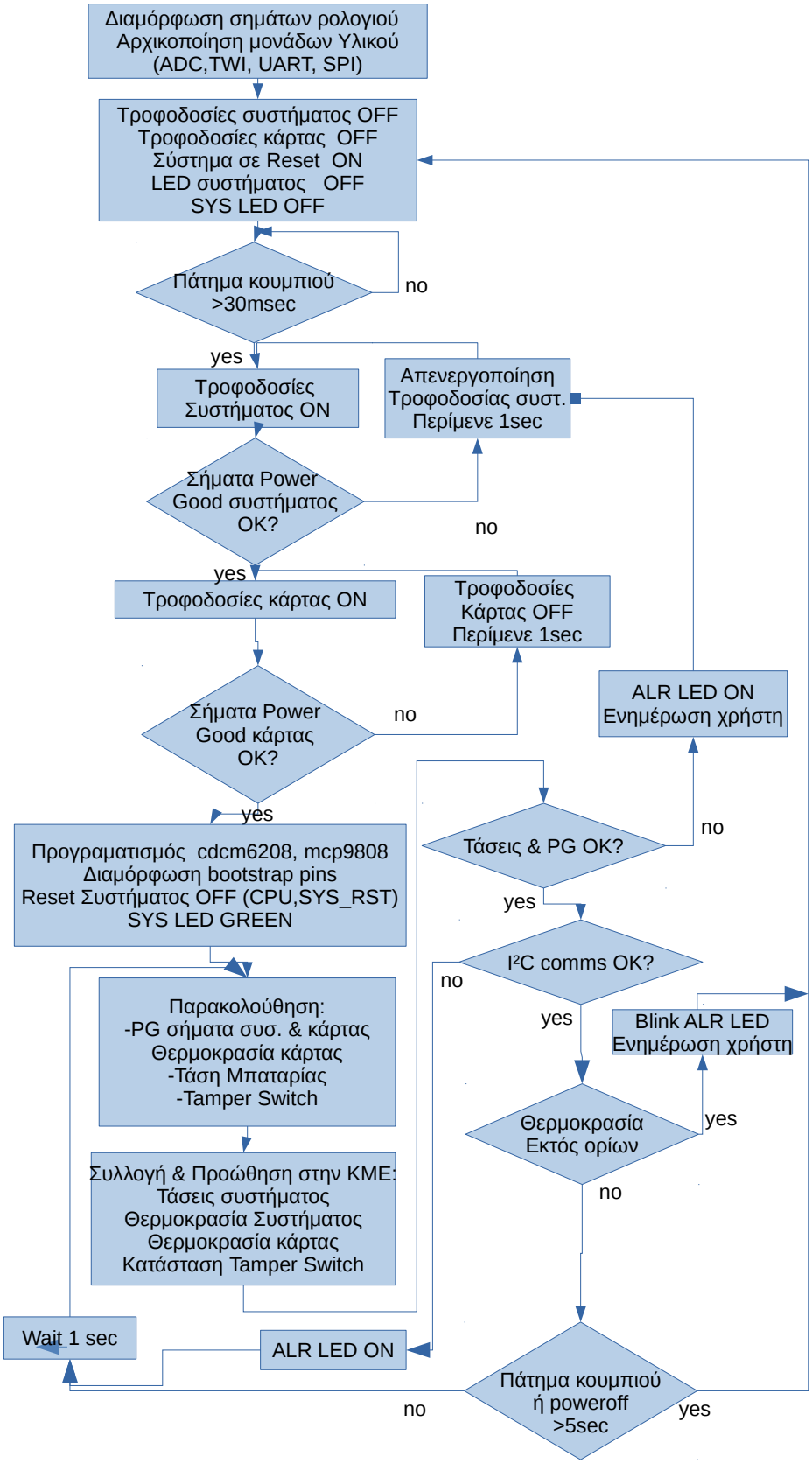
Κατάσταση LED	**Σφάλμα**
Σταθερά κόκκινο	Πρόβλημα στην επικοινωνία μεταξύ του υλικού ή/και τάση εκτός ορίων
Σταθερά σβηστό	Σύστημα σε κανονικές συνθήκες λειτουργίας
Αναβοσβήνει κόκκινο	Θερμοκρασία εκτός επιτρεπτών ορίων

Πίνακας 3.2: *Ερμηνεία λειτουργίας του Alarm LED*

- **Έλεγχος των tamper switches**.
  Σε περίπτωση που δημιουργηθεί θετικός παλμός χρονικής διάρκειας μεγαλύτερης των 20msec, ο μικροελεγκτής ενημερώνει το χρήστη μέσω UART για την παραβίαση του συστήματος. Το πρόγραμμα που περιγράφηκε παραπάνω μετατρέπει την πληροφορία αυτή σε JSON και αποστέλλοντάς την, ενημερώνεται ο χρήστης.

- **Έλεγχος της τάσης μπαταρίας του συστήματος**.
  Ο μικροελεγκτής της κάρτας μετράει την τάση της μπαταρίας του συστήματος και ενημερώνει το χρήστη για την τιμή αυτή.

Συνοπτικά τα παραπάνω εμφανίζονται στο διάγραμμα ροής της κάρτας 1 του σχήματος 5.9.



Σχήμα 3.2: *Διάγραμμα ροής του μικροελεγκτή της κάρτας 1*

3.4.2 Κάρτα 2

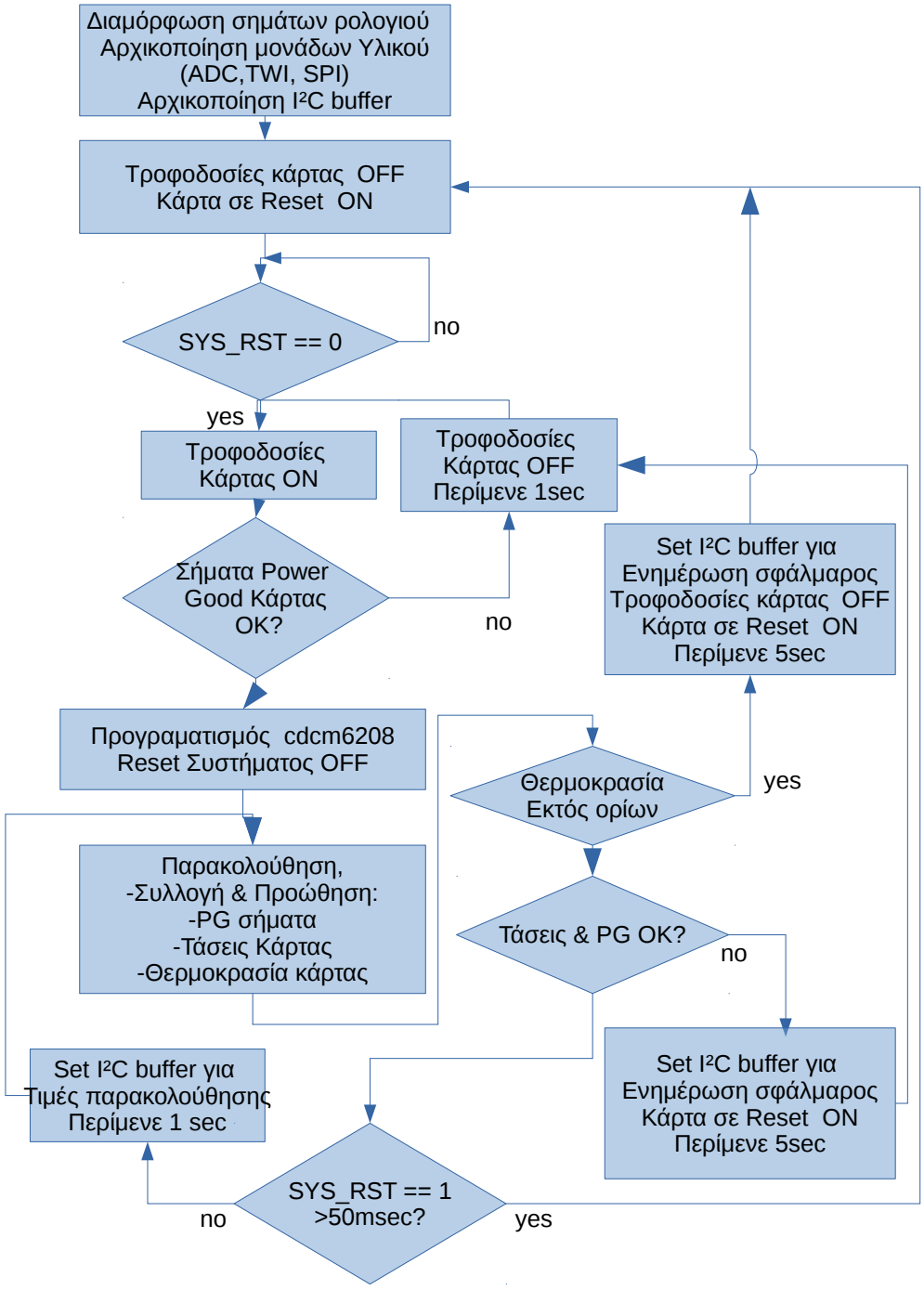
- **Ενεργοποίηση και απενεργοποίηση της κάρτας**.
  Η κάρτα κατά την εκκίνηση του συστήματος θα πρέπει να ενεργοποιήσει τις τοπικές τροφοδοσίες και στη συνέχεια να βγάλει από reset state τα ολοκληρωμένα του κυκλώματος.

- **Προγραμματισμός clock generator**.
  Ο μικροελεγκτής θα πρέπει να προγραμματίσει σωστά τον clock generator (cdcm6208) μέσω SPI, ώστε να παράξει τα απαιτούμενα για το υπόλοιπο κύκλωμα διαφορικά σήματα ολογιού.

- **Παρακολούθησή κάρτας**
  Ο μικροελεγκτής καθ' όλη τη διάρκεια λειτουργίας του παρακολουθεί την υγιή κατάσταση των τροφοδοτικών μέσω σημάτων Power Good (PG), μετρώντας ταυτόχρονα τις τάσεις στους ακροδέκτες του. Επίσης ελέγχει έμμεσα πως η κάρτα λειτουργεί στα αποδεκτά όρια θερμοκρασίας (-40,+85)˚C μέσω του σήματος ALERT που παράγει ο αισθητήρας.

- **Μετάδοση πληροφορίας στην Master κάρτα**
  Η κάρτα αφού συλλέξει τις τιμές τάσεων θα πρέπει να τις αποθηκεύσει σε τοπικούς καταχωρητές, ώστε όταν ο Master ζητήσει να διαβάσει τις τιμές αυτές η πληροφορία να είναι άμεσα διαθέσιμη.

Συνοπτικά τα παραπάνω εμφανίζονται στο διάγραμμα ροής της κάρτας 2 στο σχήμα 3.3.



Σχήμα 3.3: *Διάγραμμα ροής του μικροελεγκτή της κάρτας 2*

3.4.3 Κάρτα 3

- **Ενεργοποίηση και απενεργοποίηση της κάρτας**.
  Η κάρτα κατά την εκκίνηση του συστήματος θα πρέπει να ενεργοποιήσει τις τοπικές τροφοδοσίες και να βγάλει από reset state τα ολοκληρωμένα του κυκλώματος. Η ενεργοποίηση των τροφοδοτικών γίνεται σειριακά καθώς υπάρχουν αλληλοεξαρτώμενες μονάδες.

- **Προγραμματισμός περιφερειακού ina260**
  Κατά την εκκίνηση ο μικροελεγκτής προγραμματίζει το περιφερειακό ina260, ώστε να μπορεί να διαβάζει τις πληροφορίες της τάσης στα άκρα του καθώς επίσης και του ρεύματος που το διαρρέει. Τέλος, για την προστασία του συστήματος, στην περίπτωση που η ένταση του ρεύματος ξεπεράσει τα 0.5A, το περιφερειακό αποστέλλει σήμα ALERT στους ακροδέκτες του μικροελεγκτή, ο οποίος όταν το λάβει θα κλείσει τις τροφοδοσίες.

- **Παρακολούθηση κάρτας**
  Ο μικροελεγκτής καθ' όλη τη διάρκεια λειτουργίας του παρακολουθεί την υγιή κατάσταση των τροφοδοτικών μέσω σημάτων Power Good (PG), μετρώντας ταυτόχρονα τις τάσεις στους ακροδέκτες του. Επίσης ελέγχει έμμεσα πως η κάρτα λειτουργεί στα αποδεκτά όρια θερμοκρασίας (-40,+85)°C μέσω του σήματος ALERT που παράγει ο αισθητήρας. Αντίστοιχα παρακολουθεί το σήμα που προέρχεται από το περιφερειακό ina260. Η εναλλαγή τουλάχιστον ενός εκ των δύο σημάτων θα πρέπει να ληφθεί ασύγχρονα και όχι κατά το monitor cycle ώστε ο μικροελεγκτής να δράσει γρήγορα υπέρ της προστασίας της κάρτας.

- **Μετάδοση πληροφορίας στην Master κάρτα**
  Η κάρτα αφού συλλέξει τις τιμές τάσεων, θα πρέπει να τις αποθηκεύσει σε τοπικούς καταχωρητές, ώστε όταν ο Master ζητήσει να διαβάσει τις τιμές αυτές η πληροφορία να είναι άμεσα διαθέσιμη.

Συνοπτικά τα διαγράμματα ροής των καρτών 3 και 4 παρατίθενται στο σχήμα 3.4.

3.4.4 Κάρτα 4

- **Ενεργοποίηση και απενεργοποίηση της κάρτας**.
  Η κάρτα κατά την εκκίνηση του συστήματος θα πρέπει να ενεργοποιήσει τις τοπικές τροφοδοσίες και να βγάλει από reset state τα ολοκληρωμένα του κυκλώματος. Αυτό γίνεται σειριακά καθώς υπάρχουν αλληλοεξαρτώμενες μονάδες.

- **Παρακολούθηση κάρτας**
  Ο μικροελεγκτής καθ' όλη τη διάρκεια λειτουργίας του παρακολουθεί την υγιή κατάσταση των τροφοδοτικών μέσω σημάτων Power Good (PG), μετρώντας ταυτόχρονα τις τάσεις στους ακροδέκτες του. Επίσης ελέγχει έμμεσα πως η κάρτα λειτουργεί στα αποδεκτά όρια θερμοκρασίας (-40,+90) μέσω του σήματος ALERT που παράγει ο αισθητήρας.

- **Μετάδοση πληροφορίας στην Master κάρτα**
  Η κάρτα αφού συλλέξει τις τιμές τάσεων θα πρέπει να τις αποθηκεύσει σε τοπικούς καταχωρητές , ώστε όταν ο Master ζητήσει να διαβάσει τις τιμές αυτές, η πληροφορία να είναι άμεσα διαθέσιμη.

- **Προγραμματισμός του GPS**
  Η κάρτα κατά την εκκίνηση θα πρέπει να προγραμματίζει το GPS, ώστε να ενσωματώνει στον κύκλο λειτουργίας του, τη λειτουργία της ανίχνευσης την εξωτερικής κεραίας.

Συνοπτικά τα διαγράμματα ροής των καρτών 3 και 4 παρατίθενται στο σχήμα 3.4.

3.4.5 Λειτουργία ΚΜΕ

Η κεντρική μονάδα επεξεργασίας πρέπει να παρέχει τις εξής υπηρεσίες:

- Να υπάρχει μία βάση δεδομένων στην οποία θα αποθηκεύονται τα στοιχεία των χρηστών που μπορούν να συνδεθούν στην εφαρμογή. Επίσης στη βάση θα αποθηκεύονται τα σφάλματα υλικού που εντοπίζονται κατά τη διαδικασία παρακολούθησης και έπειτα θα εξάγονται στην εφαρμογή.

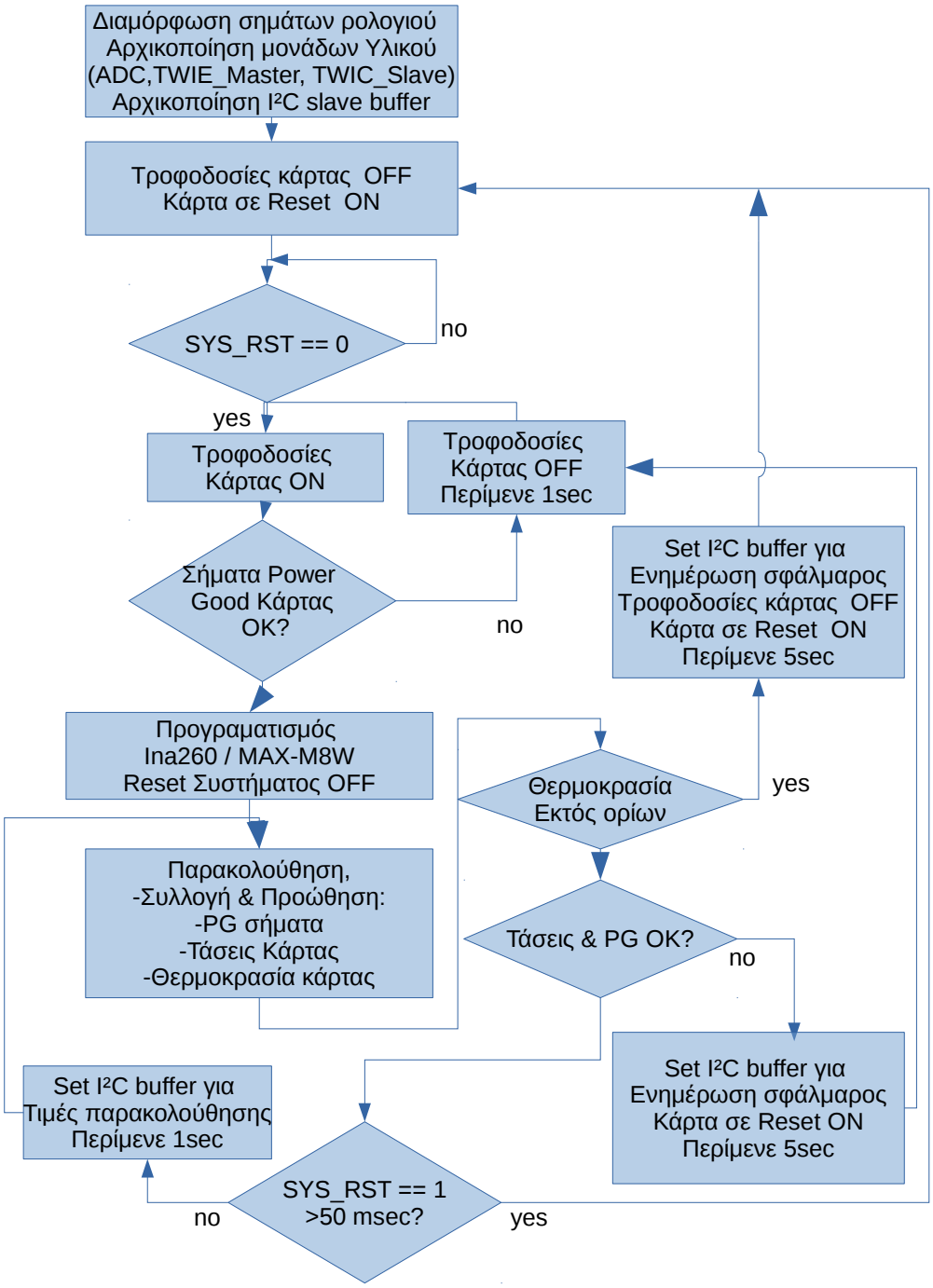
- Να υπάρχει η δυνατότητα να συνδέεται ο system administrator μέσω του πρωτοκόλλου SSH για να μπορεί να διαχειριστεί το σύστημα.

- Να λειτουργεί ως εξυπηρετητής της διαδικτυακής εφαρμογής.

- Να λαμβάνει μέσω σειριακής θύρας τα δεδομένα της παρακολούθησης και τα δεδομένα του συστήματος και έπειτα να τα εξάγει στη διαδικτυακή εφαρμογή.

- Να λαμβάνει μέσω σειριακής θύρας τα δεδομένα του GPS receiver και να ενημερώνει το χρήστη κατάλληλα.

- Θα πρέπει να κάνει resolve την public IP ή οποία αλλάζει συνεχώς, ώστε να μπορεί ο χρήστης να συνδεθεί μέσω του Internet.

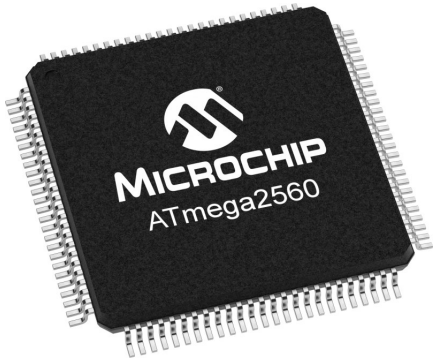


Σχήμα 3.4: *Διάγραμμα ροής του μικροελεγκτή της κάρτας 3 (ina260) & 4 (MAX-M8W)*

Κεφάλαιο 4

Περιγραφή Υλικού

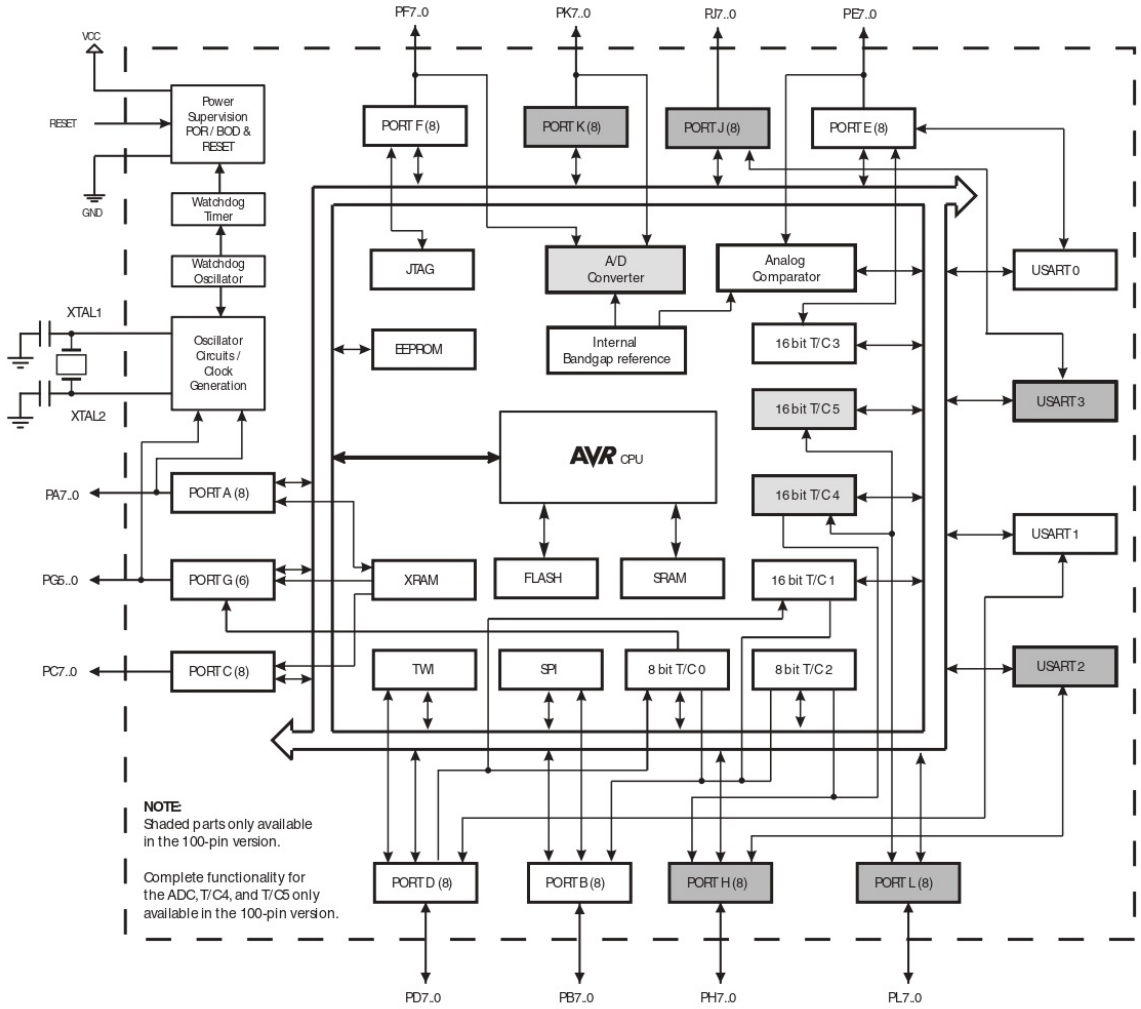
4.1 ATmega2560



Ο ATmega2560 είναι ένας 8bit μικροελεγκτής αρχιτεκτονικής RISC, χαμηλής κατανάλωσης (10-14mA σε ενεργή κατάσταση στα 8MHz με 5V τροφοδοσίας), ο οποίος εκτελεί τις περισσότερες εντολές σε μόνο έναν κύκλο ρολογιού, πετυχαίνοντας έτσι ταχύτητα έως και 1 Million Instructions Per Second (MIPS) ανά MHz. Διαθέτει υψηλής αντοχής Non-volatile μνήμες: μία 256KBytes Flash, η οποία μπορεί να προγραμματίζεται μέσω του μικροελεγκτή, μία 4KBytes EEPROM και μία 8KBytes SRAM. Διαθέτει εσωτερικό ταλαντωτή για τα σήματα ρολογιού αλλά επιτρέπει και τη χρήση εξωτερικών κρυστάλλων. Επιπροσθέτως, έχει 86 γραμμές Εισόδου/Εξόδου δίνοντας μία ελευθερία κατά το σχεδιασμό. Οι θερμοκρασίες λειτουργίας είναι -40°C έως +85°C. Για τον προγραμματισμό των μνημών του μικροελεγκτή, καθώς και για την αποσφαλμάτωση αυτού, διατίθεται μονάδα JTAG.

Ο μικροελεγκτής ενσωματώνει πολύ χρήσιμα περιφερειακά, κάποια εκ των οποίων είναι: 4 x 8-bit Timer/Counter, 4 x 16-bit Timer/Counter, 10-bit ADC μονάδα με 16 κανάλια, 4 x Επαναπρογραμματιζόμενες σειριακές UART, SPI μονάδα με δυνατότητα επιλογής μεταξύ Master/Slave, I^2C μονάδες (TWI) με δυνατότητα επιλογής μεταξύ Master/Slave, Αναλογικό συγκριτή, Μονάδα εξυπηρέτησης διακοπών,Watchdog Timer.

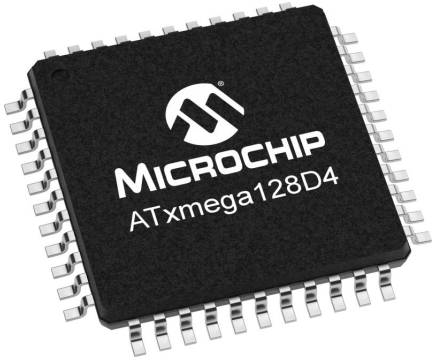
Το πλήρες σχηματικό διάγραμμα του μικροελεγκτή, που φαίνονται όσα περιγράφηκαν παραπάνω, βρίσκεται στο σχήμα 4.1



Σχήμα 4.1: *Σχηματικό διάγραμμα ATmega2560*

4.2 ATxmega128D4

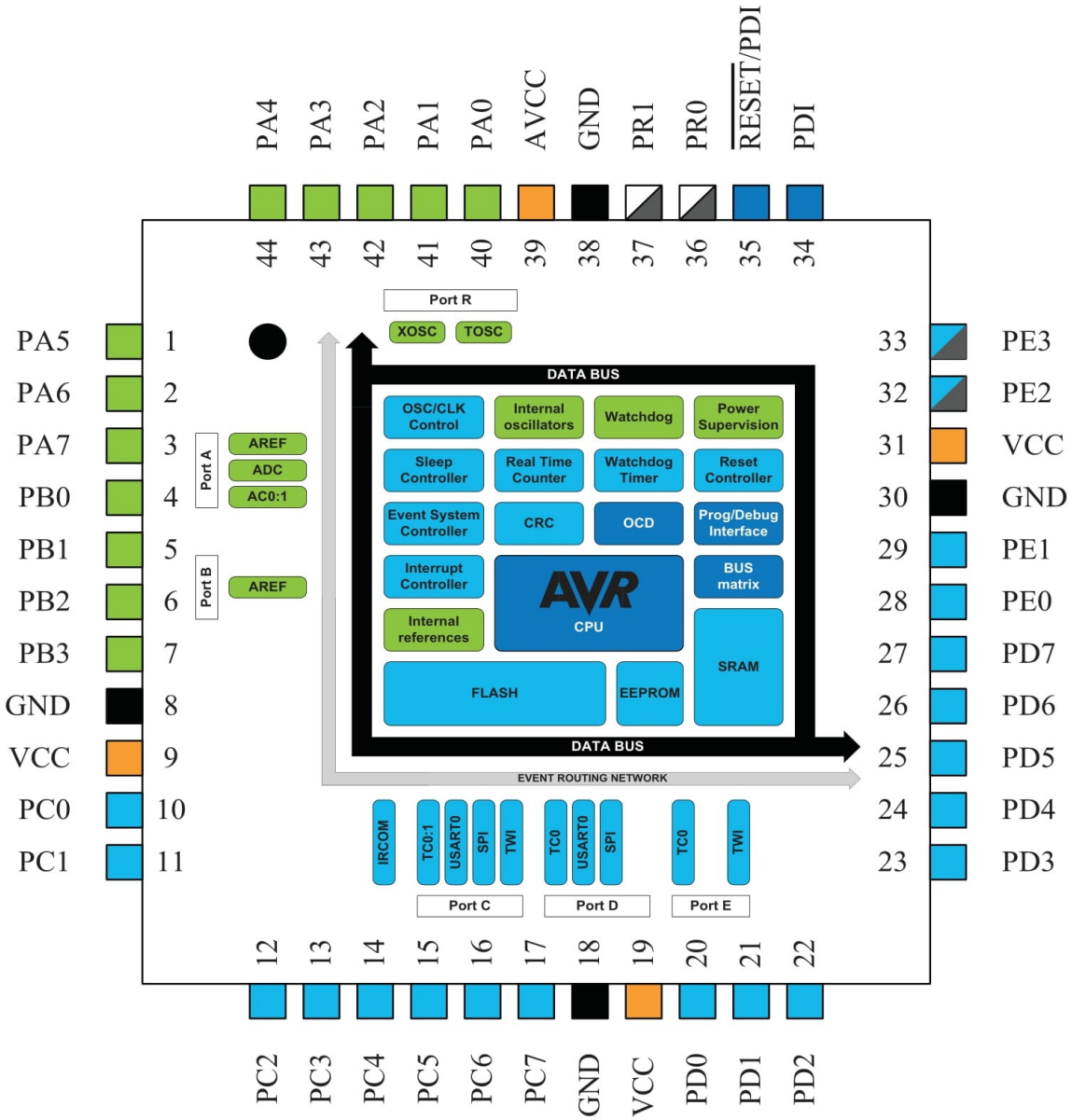
Ο ATxmega128D4 είναι ένας χαμηλής κατανάλωσης (0.75-1.4mA σε ενεργή κατάσταση στα 2MHz), υψηλής απόδοσης, 8-bit μικροελεγκτής αρχιτεκτονικής RISC. Είναι σχεδιασμένος να εκτελεί εντολές σε έναν κύκλο ρολογιού, επιτυγχάνοντας έτσι ταχύτητα έως και 1 Million Instructions Per Second (MIPS) ανά MHz. Διαθέτει εσωτερικό ταλαντωτή για τα σήματα ρολογιού αλλά επιτρέπει και τη χρήση εξωτερικών κρυστάλλων. Για τον προγραμμα-



τισμό των μνημών του μικροελεγκτή, καθώς και για την αποσφαλμάτωση αυτού, διατίθεται μονάδα PDI.

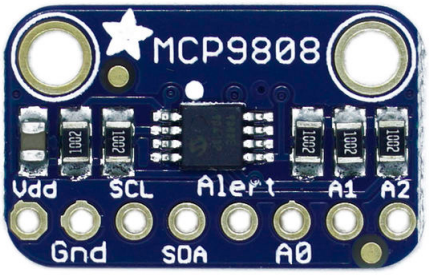
Ο μικροελεγκτής παρέχει τις παρακάτω δυνατότητες: ολοκληρωμένα/περιφερειακά: μία 128KBytes Flash η οποία μπορεί να προγραμματίζεται μέσω του μικροελεγκτή, μία 2KBytes EEPROM και μία 8KBytes SRAM, interrupt controller τριών επιπέδων (high, medium, low), 34 γραμμές E/E, τέσσερις timers/μετρητές των 16-bit, δύο προγραμματίσιμες σειριακές μονάδες (UART), δύο TWI μονάδες που υλοποιούν την επικοινωνία I^2C σε Master/Slave mode, 12-bit μονάδα μετατροπής αναλογικού σε ψηφιακό σήμα (ADC), αναλογικούς συγκριτές και προγραμματιζόμενο watchdog και επιτρέπει 5 διαφορετικές καταστάσεις εξοικονόμησης ενέργειας.

Το πλήρες σχηματικό διάγραμμα του μικροελεγκτή, που φαίνονται όσα περιγράφηκαν παραπάνω, βρίσκεται στο σχήμα 4.2



Σχήμα 4.2: *Σχηματικό διάγραμμα ATxmega128D4*

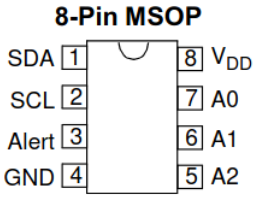
4.3 mcp9808



Ο mcp9808 είναι ένας ψηφιακός αισθητήρας θερμοκρασίας χαμηλής κατανάλωσης (200μA σε πλήρη λειτουργία στα 5V) που μετατρέπει φυσικές τιμές θερμοκρασίας από -20°C έως +100°C.

Ο mcp9808 δίνει την επιλογή στο χρήστη να προγραμματίσει τον αισθητήρα.μέσω των καναχωρητών του, παρέχοντας ευελιξία ανάλογα με την εφαρμογή στην οποία χρησιμοποιείται. Ο χρήστης μπορεί να ορίσει την ακρίβεια της μέτρησης έχοντας ως επιτρεπτές τιμές τις 0.0625°C, 0.125°C, 0.25°C, 0.5°C. Επιπλέον ο χρήστης μπορεί να επιλέξει μεταξύ των low-power και shutdown modes, ώστε να μειώσει ακόμη περισσότερο την κατανάλωση ενέργειας σε περιόδους που δεν γίνεται χρήση του αισθητήρα. Τέλος, ο αισθητήρας μπορεί να προγραμματιστεί ώστε να εξάγει σήμα ALERT, επιτρέποντας την επιλογή των χαρακτηριστικών του σήματος αυτού σε περιπτώσεις που η θερμοκρασία ξεπεράσει ένα user-defined εύρος τιμών ή μία critical τιμή θερμοκρασίας.

Ο αισθητήρας είναι I^2C compatible με ταχύτητες έως και 400KHz και δίνεται η επιλογή μεταξύ 16 διαφορετικών διευθύνσεων μέσω των ακροδεκτών A0, A1, A2 (σχήμα 4.3) όπως περιγράφεται στον πίνακα 4.1, προσφέροντας έτσι τη δυνατότητα να ελέγξουμε σε ένα μόνο bus 16 αισθητήρες.

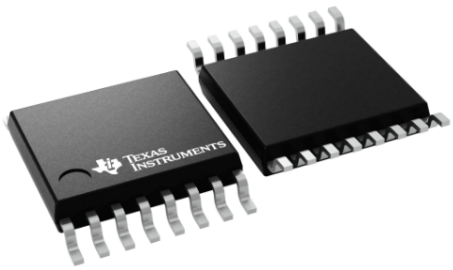


Σχήμα 4.3: *Διάγραμμα ακροδεκτών του mcp9808*

Device	Address Code				Slave Address		
	A6	A5	A4	A3	A2	A1	A0
MCP9808	0	0	1	1	x	x	x
MCP9808	1	0	0	1	x	x	x

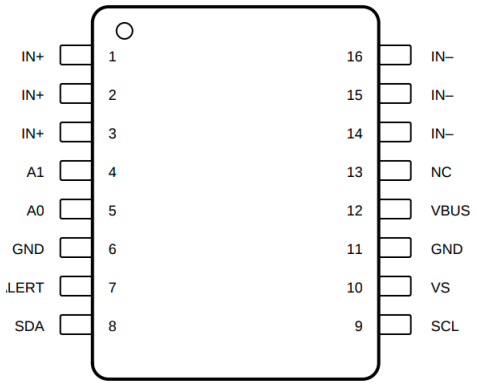
Πίνακας 4.1: *mcp9808 Byte διεύθυνσης*

4.4 ina260



Το ina260 είναι ένα ολοκληρωμένο χαμηλής ενεργειακής κατανάλωσης (κατανάλωση 310μA σε πλήρη λειτουργία από 2.7V έως 5.5V), το οποίο επιτρέπει την παρακολούθηση των τιμών του ρεύματος, ενέργειας και τάσεων του κυκλώματος. Επιτρέπει μετρήσεις υψηλής ακρίβειας (μέγιστο σφάλμα 0.15%), καθώς επίσης και ανίχνευσης ρευμάτων υψηλής έντασης. Οι μετρήσεις εξάγονται μέσω των ρευμάτων που διαρρέουν έναν current-sensing resistor, προσφέροντας μετρήσεις με πολύ χαμηλή απώλεια απόδοσης λόγω της θερμοκρασίας.

Το ολοκληρωμένο προγραμματίζεται μέσω I^2C και διαθέτει έως και 16 διαφορετικές διευθύνσεις οι οποίες ορίζονται από το σχεδιαστή υλικού μέσω των ακροδεκτών A1 και A2, όπως περιγράφονται στο σχήμα 4.4.



Σχήμα 4.4: *Διάγραμμα ακροδεκτών του cdcm6208*

4.5 cdcm6208

Το cdcm6208 είναι ένας χαμηλής κατανάλωσης (0.5W σε κανονική λειτουργία) και ευέλικτος συνθέτης συχνοτήτων, ο οποίος δημιουργεί χαμηλού θορύβου παλμούς ρολογιού που χρησιμοποιούνται σε πληθώρα εφαρμογών, όπως στις υποδομές ασύρματων και ενσύρματων δικτύων, στους υπολογιστές, στην ακτινοδιαγνωστική, σε servers αποθήκευσης δεδομένων κ.α. Το cdcm6208 παρέχει μία αρχιτεκτονική η οποία μπορεί να παράξει οποιαδήποτε συχνότητα με ακρίβεια 1ppm (Parts per Million) [17]. Αυτό σημαίνει πως η μεταβολή της συχνότητας του παλμού εξαιτίας τον εξωτερικών συνθηκών είναι μικρότερη από 0.0001%. Τέλος, το ολοκληρωμένο προγραμματίζεται μέσω I^2C και SPI και στην περίπτωση που ο

σχεδιαστής δεν θέλει να εφαρμόσει κάποιο σειριακό πρωτόκολλο επικοινωνίας, δίνεται η δυνατότητα προγραμματισμού μέσω pin σε ένα εκ των 32 έτοιμων configuration.

4.6 MAX-M8W

Το MAX-M8W [18] [19] είναι μία μονάδα υλικού που προσφέρει μία αξιόπιστη και αποδοτική μηχανή λήψης σημάτων δορυφόρων, όπως αυτών του GLONASS, QZSS, SBAS. Προσφέρει υψηλή ευαισθησία, προστασία της ακεραιότητας της πληροφορίας του μηνύματος και ανίχνευση spoofing. Ενσωματώνει τις παραπάνω λειτουργίες κρατώντας χαμηλά την κατανάλωση ενέργειας και υποστηρίζοντας διαφορετικά power save modes. Οι επιτρεπτές θερμοκρασίες λειτουργίας κυμαίνονται από -40°C έως +85°C. Επίσης προσφέρει ευκολία στην ενσωμάτωσή του σε συστήματα, καθώς είναι εύκολα προγραμματίσιμο μέσω I^2C (αναφέρεται ως DDC στην u-blox). Επιτρέπει ακόμη την προσθήκη εξωτερικής κεραίας λήψης. Είναι συμβατό με το Fast-Mode του I^2C αφού η μέγιστη ταχύτητα SCL είναι 400KHz. Η πληροφορία που αποκτάται από τους δορυφόρους μεταδίδεται στη μορφή NMEA ή UBX μέσω της σειριακής μονάδας UART. Ο χρήστης μπορεί να προγραμματίσει την ταχύτητα μετάδοσης της σειριακής (baudrate).

Κεφάλαιο 5

Υλοποίηση

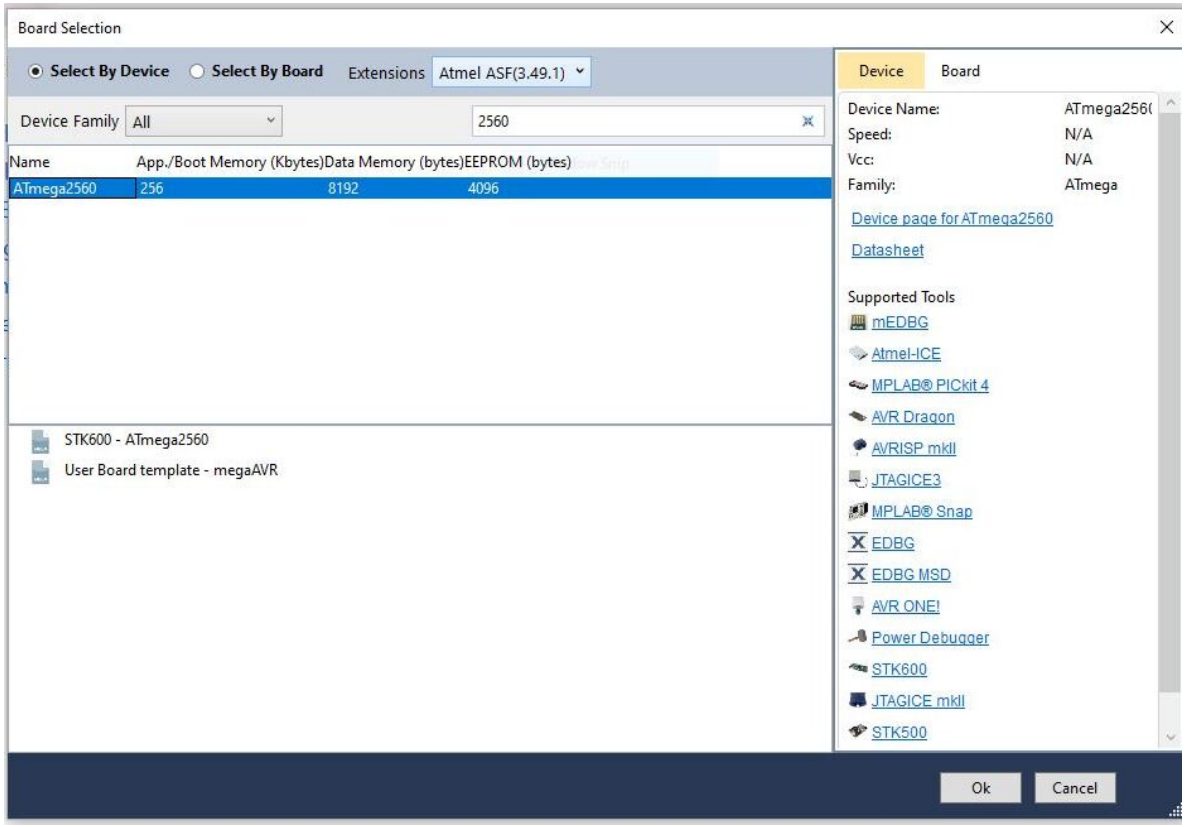
5.1 Περιβάλλον ανάπτυξης Microchip Studio

Το ολοκληρωμένο περιβάλλον ανάπτυξης Microchip Studio [20] είναι ένα πρόγραμμα το οποίο παρέχει στον προγραμματιστή υποστήριξη σε όλους τους τύπους AVR μικροελεγκτών, τόσο για τη συγγραφή κώδικα σε C/C++ [21], όσο και για τη μεταγλώττιση και αποσφαλμάτωσή του. Είναι συμβατό με τον Atmel ICE, ο οποίος αποτελεί τον debugger που χρησιμοποιήσαμε για τον προγραμματισμό των μνημών των μικροελεγκτών και για την αποσφαλμάτωση του κώδικα.

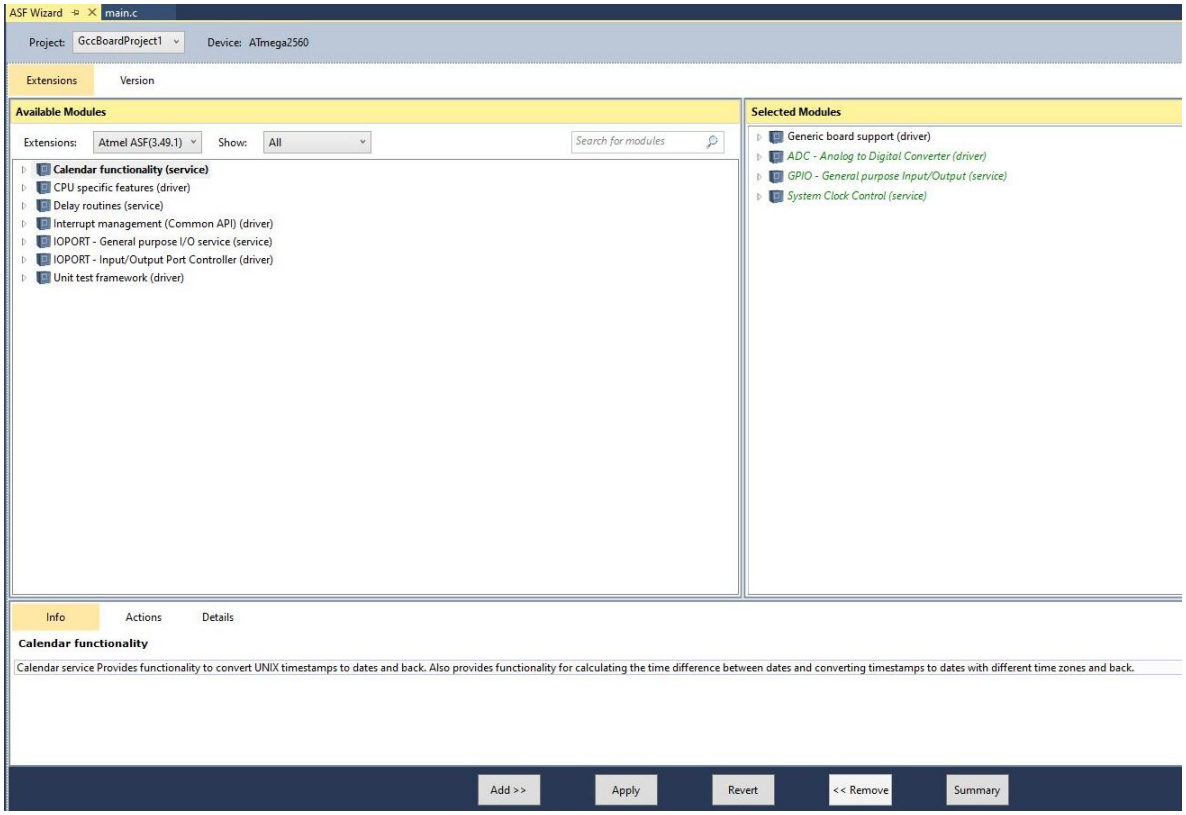
Για να ξεκινήσουμε την υλοποίηση ενός καινούριου project επιλέγουμε *File* → *New* → *Project*. Στη συνέχεια διαλέγουμε GCC C ASF Board Project. Το ASF ερμηνεύεται ως Advanced Software Framework και αποτελεί μία open-source βιβλιοθήκη από drivers και κομμάτια κώδικα, τα οποία έχουν συγγραφεί από επαγγελματίες και βοηθούν πολύ στην ανάπτυξη ενός λογισμικού. Παρόλα αυτά, οι βιβλιοθήκες έχουν εφαρμογή σε λειτουργίες γενικής χρήσης και μπορεί να εκτελούν πολύ περισσότερες εντολές και να χρησιμοποιούν περισσότερους πόρους σε σχέση με μία custom σχεδίαση. Για το λόγο αυτό, πολλές φορές τα κομμάτια κώδικα έχουν συμβουλευτική χρήση και δεν ενσωματώνονται αυτούσια.

Αφού επιλέξουμε το GCC C ASF Board Project, επόμενο βήμα είναι η επιλογή του μοντέλου του μικροελεγκτή που θα χρησιμοποιήσουμε. Διαλέγοντας τον ATmega2560 εμφανίζεται και μία λίστα από τα υποστηριζόμενα εργαλεία, καθώς επίσης και διάφορα χρήσιμα links όπως φαίνεται στην εικόνα 5.1. Επιλέγοντας User Board template - megaAVR και πατώντας το πλήκτρο OK δημιουργείται ένα ολόκληρο σύνολο από αρχεία και φακέλους τα οποία φαίνονται επιλέγοντας *View* → *Solution Explorer*. Το Microchip Studio έχει συνδέσει τα αρχεία που έχει δημιουργήσει, έτσι ώστε χτίζοντας την εφαρμογή *(Project* → *Build)* να μην εξάγεται σφάλμα. Οι μεταγλωττιστές, τα μονοπάτια των αρχείων, οι επιλογές μεταγλώττισης, οι συσκευές αποσφαλμάτωσης και πολλές άλλες χρήσιμες λειτουργίες βρίσκονται στο μενού *File* → *Project Properties*.

Για την ενσωμάτωση κώδικα από το ASF, διαλέγουμε *ASF* → *ASF Wizard*. Το ASF Wizard είναι ένας βοηθός ενσωμάτωσης χρήσιμων driver στο project μας. Στην περίπτωσή μας, αφού διαλέξουμε το project από το drop down menu του βοηθού, συμπεριλαμβάνουμε στην υλοποίηση μας τους drivers για τη μονάδα ADC, τη διαχείριση του ρολογιού και τη διαχείριση των ακροδεκτών Εισόδου/Εξόδου. Αυτό γίνεται διαλέγοντας τα παραπάνω από τη λίστα αριστερά, πατώντας το κουμπί *ADD* και στη συνέχεια το κουμπί *Apply*.



Εικόνα 5.1: *Δημιουργία GCC C ASF Board Project για τον ATmega2560 στο Microchip Studio*



Εικόνα 5.2: *Προσθήκη drivers μέσω του ASF Wizard*

Αφού γίνει η ανάπτυξη του κώδικα μας, συνδέουμε τον Atmel ICE με τις διεπαφές JTAG για τον ATmega2560 και PDI για τον ATxmega128D4, προσέχοντας πάντα την αντιστοίχιση του pinout. Τέλος, για να προγραμματίσουμε τις Flash μνήμες από τις οποίες οι μικροελεγκτές διαβάζουν τον κώδικα προς εκτέλεση πατάμε το πράσινο πλήκτρο *Run*. Κατά το Running mode δίνεται η δυνατότητα αποσφαλμάτωσης του κώδικα με τη χρήση breakpoints (*Debug* sub-menu).

5.2 Αρχικοποίηση μικροελεγκτών

Κάθε φορά που ένας μικροελεγκτής τροφοδοτείται από τάση και εκκινεί θα πρέπει να αρχικοποιήσει τις μονάδες υλικού, καθώς οι ρυθμίσεις προηγούμενης λειτουργίας δεν αποθηκεύονται σε κάποια non-volatile μνήμη, με αποτέλεσμα έπειτα από κάθε διακοπή τροφοδοσίας να χάνονται οι επιλογές του προγραμματιστή.

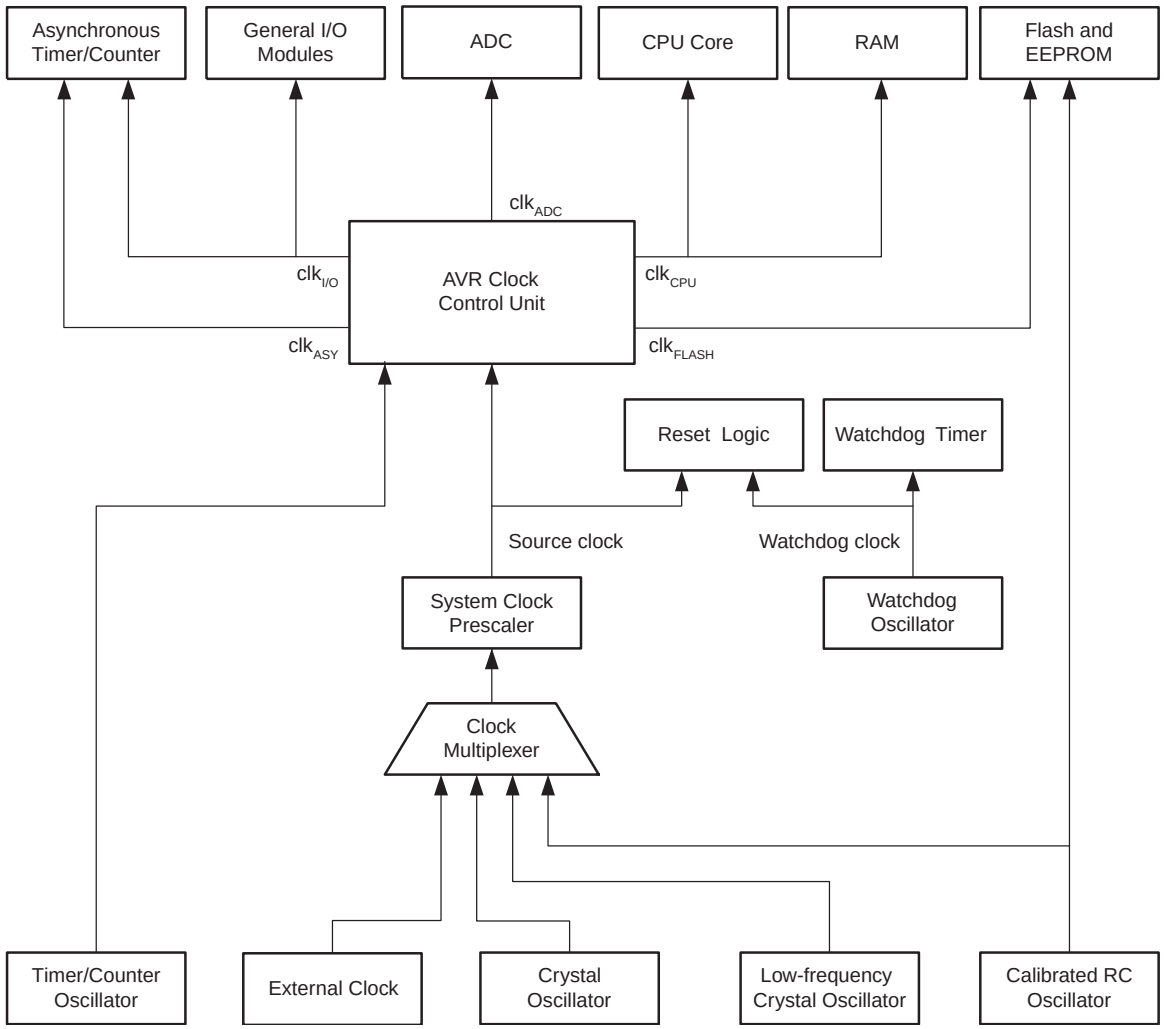
Μία ρύθμιση που πρέπει να γίνει κατά την εκκίνηση είναι η αρχικοποίηση των ακροδεκτών αναλόγως τη χρήση τους. Στην περίπτωση μας, επειδή χειριζόμαστε πολλά σήματα Εισόδου/Εξόδου, τα οποία πρακτικά ενεργοποιούν τροφοδοσίες και επιτρέπουν τη λειτουργία των ολοκληρωμένων, πρέπει κατά την εκκίνηση τα σήματα αυτά να έχουν συγκεκριμένες τιμές. Για το λόγο αυτό, στα ζωτικής σημασίας σήματα προσθέτουμε pull-up, pull-down αντιστάσεις, ώστε τα σήματα να παραμείνουν σε υψηλή η χαμηλή στάθμη αντίστοιχα μέσω του υλικού. Σε επίπεδο λογισμικού προσαρμόζουμε τους ακροδέκτες είτε ως εισόδους, είτε ως εξόδους με τη συνάρτηση *ioport_set_pin_dir(pin, direction)*. Στην περίπτωση που ένα σήμα αποτελεί έξοδο του μικροελεγκτή, πρέπει να καθορίσουμε και τη στάθμη στην οποία θα βρίσκεται με την εντολή *arch_ioport_set_pin_level(pin , level)* έτσι ώστε να μην επιτρέψουμε τη λειτουργία υποσυστήματος σε λάθος συνθήκες περιβάλλοντος.

Για να είμαστε σίγουροι πως θα έχουμε την επιθυμητή συμπεριφορά σε κάθε εκκίνηση, αρχικοποιούμε τους μικροελεγκτές ρυθμίζοντας την ταχύτητα του ρολογιού. Το ρολόι είναι η καρδιά του μικροελεγκτή, καθώς αυτό παράγει και συγχρονίζει όλα τα περιφερειακά και τις μονάδες υλικού. Έτσι χρειάζεται ιδιαίτερη προσοχή στον τρόπο ρύθμισης του. Στα επόμενα δύο υποκεφάλαια περιγράφεται ο τρόπος προγραμματισμού των μονάδων χρονισμού των δύο μικροελεγκτών που χρησιμοποιούμε.

5.2.1 ATmega2560

Για τον προγραμματισμό της μονάδας χειρισμού των σημάτων ρολογιού (σχήμα 5.1) χρησιμοποιούμε το clock service που παρέχει το ASF. Συγκεκριμένα, τρέχοντας τη συνάρτηση *sysclk_init()* κατά την εκκίνηση του μικροελεγκτή αρχικοποιούμε τη μονάδα και απενεργοποιούμε τα σήματα ρολογιού σε όλες τις υπό-μονάδες υλικού εκτός των σημάτων clk_{FLASH} και του clk_{CPU}. Έτσι μπορούμε να είμαστε σίγουροι πως δεν είναι ενεργοποιημένοι πόροι που δεν χρειάζονται και πως εάν ενεργοποιηθούν θα έχουν πρώτα ρυθμιστεί από εμάς σύμφωνα με τις ανάγκες της εφαρμογής.

Η ταχύτητα που επιλέξαμε στους δύο μικροελεγκτές μας είναι 7.3728MHz για δύο λόγους. Αρχικά γιατί θέλουμε να έχουμε 0% σφάλμα κατά τη μετάδοση πακέτων στην ασύγχρονη σειριακή επικοινωνία στα 115.2K baudrate, όπως περιγράφεται στο Table 22-10



Σχήμα 5.1: *Διανομή σημάτων ρολογιού του μικροελεγκτή ATmega2560*

του datasheet. Εν συνεχεία, διότι όπως περιγράφεται στο κεφάλαιο *24.5.2* του εγχειριδίου, η ταχύτητα του $\text{clk}_{I/O}$ σε slave mode θα πρέπει να είναι 16 φορές μεγαλύτερη από την ταχύτητα του σήματος SCL του I^2C bus, δηλαδή 400KHz, άρα 400K $*16$ = 6.4MHz. Έτσι επιλέξαμε την προσθήκη εξωτερικού ταλαντωτή ταχύτητας 7.3728MHz, τον οποίο συνδέσαμε στον ακροδέκτη XTAL1 αφήνοντας τον ακροδέκτη XTAL2 αποσυνδεδεμένο. Η ταχύτητα του $\text{clk}_{I/O}$ είναι η ίδια με την ταχύτητα του clk_{CPU} με μόνη διαφορά ότι το $\text{clk}_{I/O}$ μπορεί να απενεργοποιηθεί για μείωση κατανάλωσης.

Για να είναι συμβατή η υλοποίησή μας με τους drivers του ASF πρέπει να ορίσουμε την τιμή BOARD_EXTERNAL_CLK. Αυτό γίνεται διότι κατά την αρχικοποίηση διαφορετικών μονάδων υλικού ο μικροελεγκτής χρειάζεται να γνωρίζει την ταχύτητα στην οποία είναι χρονισμένος, ώστε να ρυθμίσει μέσω των prescalers τα ρολόγια των περιφερειακών. Ορίζουμε έτσι στον κώδικα μας το εξής:

```
#define BOARD_EXTERNAL_CLK 7372800
```

Με την παραπάνω γραμμή κώδικα ορίζουμε την ταχύτητα του εξωτερικού ρολογιού, δεν έχουμε όμως ενημερώσει το μικροελεγκτή πού θα βρει αυτήν την πηγή ταλάντωσης. Αυτό

γίνεται με την εγγραφή στα Fuse Bits. Έτσι, γράφοντας την τιμή 0xFD στο Fuse low byte επιτυγχάνουμε τα εξής: Αρχικά, αφαιρούμε το CKDIV8 το οποίο διαιρεί την ταχύτητα του ρολογιού με 8. Δεύτερον, διαλέγουμε μέσω του CKSEL τον εξωτερικό ταλαντωτή που θα χρησιμοποιήσουμε (ουσιαστικά ρυθμίζουμε τον Clock Multiplexer του σχήματος 5.1). Η επιλογή της πηγής ρολογιού δεν θα αλλάξει ακόμη κι αν διακόψουμε την τροφοδοσία διότι τα Fuse bytes βρίσκονται στη Flash η οποία είναι μνήμη non volatile.

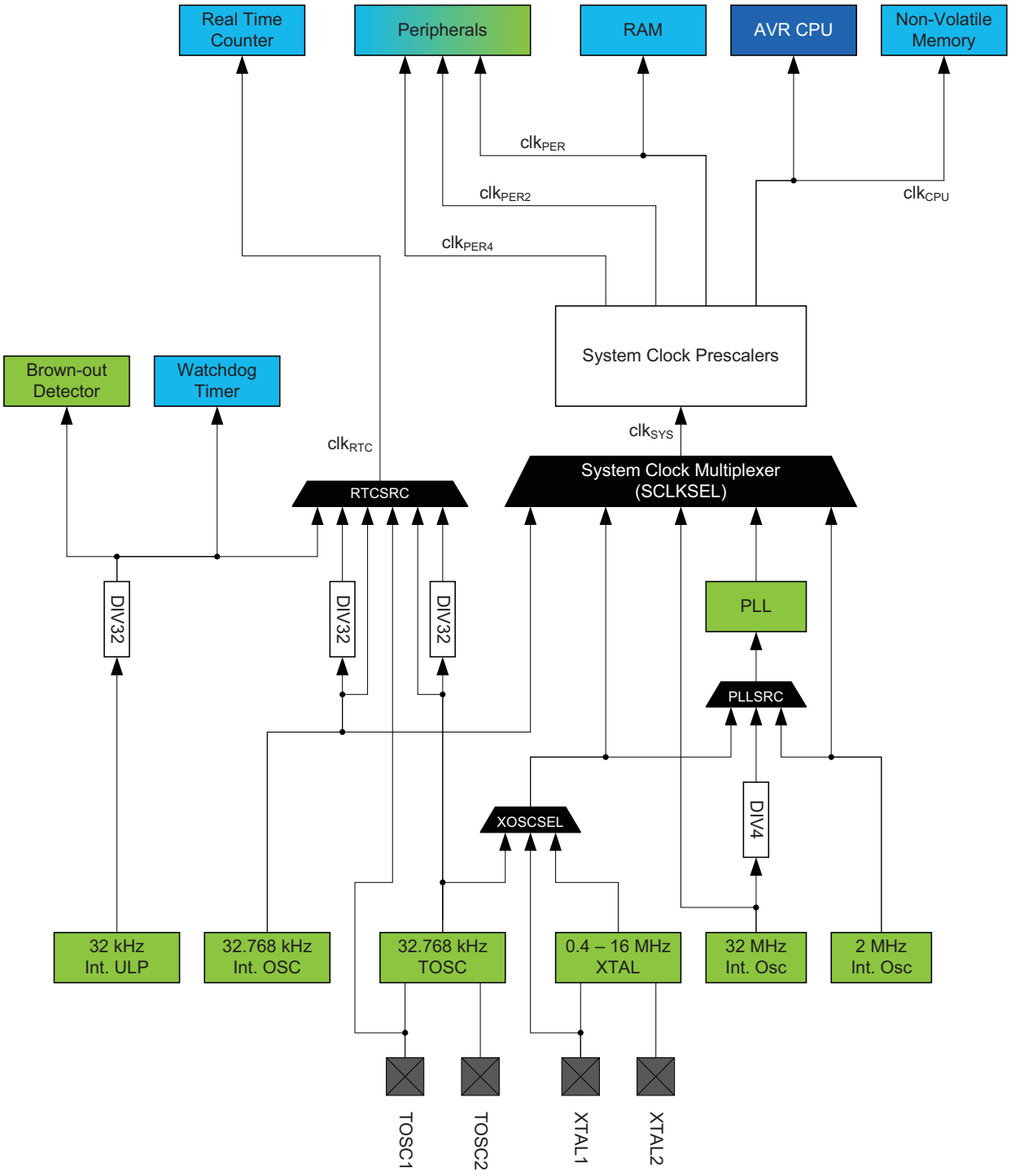
5.2.2 ATxmega128D4

Όπως φαίνεται στο σχήμα 5.2, η διανομή των σημάτων ρολογιού δεν διαφέρει πολύ στους δύο μικροελεγκτές. Η σημαντική διαφορά έγκειται στο γεγονός ότι στον ATxmega128D4 η ρύθμιση επιλογής της πηγής της ταλάντωσης δεν αποθηκεύεται μόνιμα σε μνήμη Flash και θα πρέπει να προγραμματίζεται κάθε φορά κατά την τροφοδοσία του μικροελεγκτή.

Στις εφαρμογές μας οι απαιτήσεις σχετικά με τα σήματα ρολογιού έχουν να κάνουν μόνο με τη μονάδα ADC, την I^2C επικοινωνία και τη μονάδα του Timer. Όλες αυτές οι ανάγκες καλύπτονται με τη χρήση των εσωτερικών ταλαντωτών που παρέχονται. Η ταχύτητα στην οποία ρυθμίστηκε προέκυψε έπειτα από πειραματική μελέτη και είναι αυτή των 32MHz. Σε χαμηλότερες ταχύτητες δεν μπορούσε ο μικροελεγκτής να χειριστεί την επικοινωνία I^2C παράλληλα με την παρακολούθηση.

Κάνοντας χρήση της βιβλιοθήκης ASF μπορούμε εύκολα να ρυθμίσουμε την ταχύτητα ρολογιού έχοντας πρώτα ορίσει τη μεταβλητή CONFIG_SYSCLK_SOURCE και τρέχοντας τη συνάρτηση *sysclk_init()*. Για 32MHz ορίζουμε:

```
#define CONFIG_SYSCLK_SOURCE SYSCLK_SRC_RC32MHZ
```



Σχήμα 5.2: *Διανομή σημάτων ρολογιού του μικροελεγκτή ATxmega128D4*

5.3 Ενεργοποίηση συστήματος

Η ύπαρξη υλικού σε ένα οποιοδήποτε σύστημα δεν προσδίδει καμία λειτουργική συμπεριφορά δίχως την ύπαρξη ενός μηχανισμού ενεργοποίησης και απενεργοποίησης του συστήματος. Στην υλοποίηση μας ο τρόπος με τον οποίο το σύστημα μας εκκινεί είναι μέσω ενός κουμπιού, όπως συνηθίζεται στους υπολογιστές γενικής χρήσης.

Όταν το σύστημα τροφοδοτείται παρέχεται στους μικροελεγκτές των καρτών τάση 3.3Volt, η οποία προκαλεί την εκκίνηση της εκτέλεσης του κώδικα που είναι αποθηκευμένος στην Flash μνήμη του κάθε μικροελεγκτή. Αυτή η κατάσταση ονομάζεται idle και διαφέρει από την ενεργοποίηση του συστήματος που περιγράφεται παρακάτω. Έχοντας το κουμπί της πρόσοψης συνδεδεμένο με το μικροελεγκτή της κάρτας 1, μέσω του σήματος POWER_BUTTON (ακροδέκτης PK3) ο μικροελεγκτής μπορεί να ανιχνεύσει το πάτημα του κουμπιού.

5.3.1 Διακοπές σημάτων Εισόδου/Εξόδου (GPIO interrupts)

Το σήμα POWER_BUTTON είναι active high, που σημαίνει πως κατά το πάτημα του κουμπιού (active) το σήμα που παράγεται και διαβάζει ο μικροελεγκτής βρίσκεται σε υψηλή στάθμη (high). Ρυθμίζουμε το μικροελεγκτή μας ώστε η λήψη του σήματος του κουμπιού να φτάνει ως interrupt, ώστε να παρέχουμε real time λειτουργία στο χρήστη. Γράφοντας στον καταχωρητή PCICR την τιμή 1 στο πεδίο PCIE2 ενεργοποιούμε τις διακοπές για οποιαδήποτε εναλλαγή στάθμης στους ακροδέκτες με διακοπές PCINT23:16, οι οποίες εξυπηρετούνται από τον PCINT2 interrupt vector. Διαβάζοντας το εγχειρίδιο βρίσκουμε πως η εναλλαγή της στάθμης του ακροδέκτη PK3 προκαλεί τη διακοπή PCINT19. Για να την ενεργοποιήσουμε γράφουμε στον καταχωρητή PCMSK2 την τιμή 1 στο πεδίο PCINT19. Έπειτα τρέχοντας την εντολή *cpu_irq_enable()* ενεργοποιούμε τη δυνατότητα του μικροελεγκτή να λαμβάνει διακοπές υλικού. Όταν προκαλείται μία διακοπή στον PCINT2_vector δεν γνωρίζουμε ποιό από τα οκτώ σήματα (PCINT23:16) προκάλεσαν τη διακοπή, οπότε θα πρέπει να γίνει έλεγχος μέσα στη ρουτίνα εξυπηρέτησης της διακοπής. Συνοπτικά ο κώδικας ενεργοποίησης των διακοπών είναι:

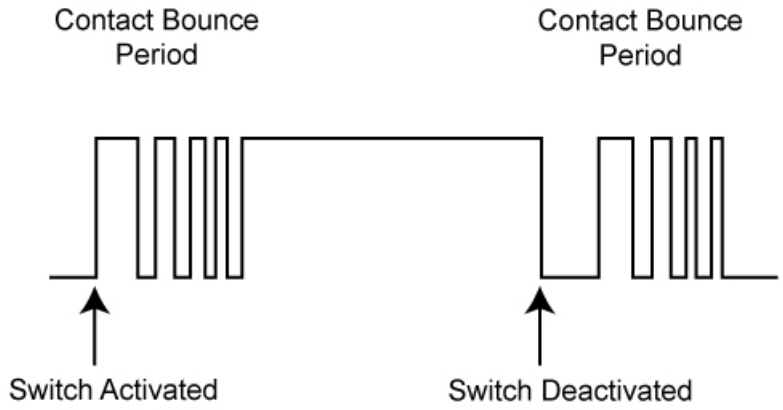
```
PCICR |= 1 << (PCIE2);
PCMSK2 |= 1 << (PCINT19);
cpu_irq_enable();

ISR(PCINT2_vect){ read_gpio_state(POWER_BUTTON); ...}
```

5.3.2 Αναπήδηση διακοπτών (Pushbutton Debounce) - Μονάδα μετρητή

Οι διακόπτες (κουμπιά) από τη σκοπιά της ηλεκτρονικής έχουν ένα πρόβλημα: κατά το πάτημα τους η επαφή των δύο μηχανικών μερών δεν γίνεται ακαριαία, όπως πιστεύει ο χρήστης. Στην πραγματικότητα η επαφή τους εναλλάσσεται σε πολλαπλές καταστάσεις φυσικής σύνδεσης και φυσικής αποσύνδεσης. Έτσι σε μικρό χρονικό διάστημα παράγονται πολλαπλά σήματα εναλλαγής από χαμηλή σε υψηλή στάθμη, όπως φαίνεται στο σχήμα 5.4. Το φαινόμενο αυτό ονομάζεται αναπήδηση διακόπτη [22].

Από την πλευρά του λογισμικού λύνουμε αυτό το πρόβλημα με τη βοήθεια της μονάδας Timer/Counter. Μετράμε αρχικά την περίοδο αναπήδησης του διακόπτη με χρήση παλμογράφου στα 1.2msec. Έπειτα, η λογική είναι η εξής: Κατά τη λήψη του interrupt που προκαλείται από την εναλλαγή της στάθμης του σήματος POWER_BUTTON και όταν αυτό είναι ενεργό (POWER_BUTTON == 1), αρχικοποιούμε ένα μετρητή να παράξει διακοπή σε χρόνο μεγαλύτερο από την περίοδο αναπήδησης και μικρότερο από το χρόνο που ένας



Σχήμα 5.3: *Σήμα εισόδου στο μικροελεγκτή κατά την αναπήδηση ενός διακόπτη*

χρήστης κρατάει πατημένο το κουμπί, δηλαδή από 2msec μέχρι 40msec. Οι χρόνοι αυτοί έχουν μετρηθεί με παλμογράφο. Κατά την εξυπηρέτηση της διακοπής που προκαλείται από το μετρητή γίνεται έλεγχος εάν το σήμα POWER_BUTTON είναι ακόμη ενεργό, δηλαδή 1. Εάν ισχύει η συνθήκη, είμαστε σίγουροι πως ο διακόπτης πατήθηκε από ένα χρήστη. Σε αντίθετη περίπτωση ερμηνεύτηκε λανθασμένα μία εναλλαγή στο σήμα POWER_BUTTON ως πάτημα του διακόπτη και απορρίπτεται.

Για να υλοποιήσουμε την παραπάνω προτεινόμενη λύση, προγραμματίζουμε τη μονάδα Timer1 ως counter ο οποίος αυξάνει την τιμή κατά 1 σε κάθε clock tick του clk_{timer} και παράγει διακοπή όταν γίνει overflow στον buffer που αποθηκεύεται η τιμή της μέτρησης. Έτσι οι μόνοι παράγοντες που επηρεάζουν το χρόνο που προκαλείται η διακοπή είναι η ταχύτητα που αυξάνεται ο μετρητής, δηλαδή το clock tick και η αρχική τιμή του buffer. Για παράδειγμα, εάν η ταχύτητα του ρολογιού είναι 7.37MHz και η αρχική τιμή του buffer είναι 0, ένας μετρητής θα πάρει την τιμή 10 σε $10 * \frac{1}{7.37MHz} = 10 * 13.5nsec = 135nsec$. Στην προκειμένη περίπτωση που ο buffer είναι 16-bit και αρχικοποιημένος στο 0, γνωρίζουμε πως το overflow συμβαίνει στην τιμή $2^{16} = 65536$ και για ταχύτητα 7.37MHz συμβαίνει κάθε 0.884msec. Επειδή όμως αυτό σε πολλές περιπτώσεις είναι πολύ μικρό χρονικό διάστημα, δίνεται η επιλογή να ρίξουμε την ταχύτητα του ρολογιού μέσω των prescalers, έτσι ώστε να δημιουργήσουμε διακοπές υλικού σε χρόνους μεγαλύτερους του δευτερολέπτου. Έχουμε δηλαδή διαίρεση του $clk_{I/O}$:

$$clk_{TIMER}\,(Hz) = \frac{clk_{I/O}(Hz)}{prescaler} \qquad (1)$$

Η συνάρτηση υπολογισμού του επιθυμητού χρόνου παραγωγής διακοπών δίνεται:

$$Interrupt\_Interval\,(sec) = \frac{2^{16} - buffer\_initial\_value}{clk_{TIMER}(Hz)} \qquad (2)$$

Λύνοντας ως προς την τιμή buffer_initial_value και κάνοντας χρήση της εξίσωσης (1) έχουμε:

$$buffer\_initial\_value = 2^{16} - \frac{Interrupt\_Interval * clk_{I/O}}{prescaler}$$

Συνεπώς για την παραγωγή διακοπών 40msec έπειτα από τη λήψη POWER_BUTTON το ρολόι 7.37MHz δεν μας αρκεί, αφού ο μέγιστος χρόνος που παράγεται διακοπή είναι 0.884msec. Εύκολα υπολογίζουμε την τιμή του ικανοποιητικού prescaler για Interrupt Interval 40msec μέσω της (2)

$$(40 * 7372800)/2^{16} = \mathbf{4.5}$$

sec

Άρα θα χρειαστούμε διαίρεση του $clk_{I/O}$ κατά 8, αφού είναι η επόμενη ακέραια επιτρεπτή τιμή μετά το 4,5. Η τιμή που θα πρέπει να αρχικοποιούμε τον buffer σε κάθε overflow είναι

$$buffer\_initial\_value = 2^{16} - \frac{40msec * 7372800}{8} = \mathbf{28672}$$

Για να γίνει αυτό, γράφουμε στον καταχωρητή TCCR1A την τιμή 0 για να θέσουμε τη μονάδα σε λειτουργία μετρητή (counter). Στη συνέχεια γράφουμε στον καταχωρητή TCCR1B την τιμή 010 στα CS12:0 bits, αφού αυτά καθορίζουν την τιμή του prescaler στη μονάδα με βάση τον πίνακα 5.1. Τέλος, αρχικοποιούμε την τιμή του buffer (TCNT1) σε 28672 και ενεργοποιούμε το overflow interrupt στον καταχωρητή TIMSK1 θέτοντας 1 το πεδίο TOIE1.

CSn2	**CSn1**	**CSn0**	**Περιγραφή**
0	0	0	Καμία πηγή ρολογιού (Διακοπή λειτουργίας)
0	0	1	$clk_{I/O}$ / 1
0	1	0	$clk_{I/O}$ / 8
0	1	1	$clk_{I/O}$ / 64
1	0	0	$clk_{I/O}$ / 256
1	0	1	$clk_{I/O}$ / 1024
1	1	0	Εξωτερικό Ρολόι αρνητικά ακμοπυροδότητο
1	1	1	Εξωτερικό Ρολόι θετικά ακμοπυροδότητο

Πίνακας 5.1: *Επιλογή prescaler για τη μονάδα Timer/Counter*

Συνοπτικά ο κώδικας για την ενεργοποίηση του μετρητή:

```
power_timer1_enable();
// Normal mode (Counter)
TCCR1A = 0;
//timer runs at (clk_speed)/8
TCCR1B |= (1 << CS11);
//Initialise counter value
TCNT1 = 15000;
// Enable Timer1 overflow interrupt
TIMSK1 = (1 << TOIE1);
```

Συνοπτικά ο κώδικας για την απενεργοποίηση του μετρητή:

```
// disable overflow interrupts
TIMSK1 &= _BV(TOIE1);
// stop timer
TCCR1B = 0;
power_timer1_disable();
```

5.3.3 Τροφοδοσία συστήματος

Αφού λάβει το σήμα ενεργοποίησης, ο μικροελεγκτής της κάρτας 1 ενεργοποιεί τις τροφοδοσίες του συστήματος θέτοντας το σήμα PS_ON (PK6) ίσο με 1. Το σήμα αυτό είναι συνδεδεμένο με τα σήματα enable των τροφοδοτικών. Τα τροφοδοτικά του συστήματος ενεργοποιούνται και ελέγχουμε εάν τα σήματα Power Good γίνονται 1. Η συνθήκη αυτή δηλώνει πως έχει παραχθεί αξιόπιστη τροφοδοσία που δε θα βλάψει το σύστημα. Στην περίπτωση που τα σήματα είναι όλα high, η κάρτα 1 ενεργοποιεί με τον ίδιο τρόπο τα τοπικά τροφοδοτικά της κάρτας υλοποιώντας τον αντίστοιχο έλεγχο μέσω των Power Good signals. Όταν όλες οι τροφοδοσίες σταθεροποιηθούν, τότε η κάρτα 1 βγάζει από Reset τα υπόλοιπα υποσυστήματα θέτοντας τον ακροδέκτη SYS_RST (PJ4) low. Επίσης, ανάβει το SYS_LED με πράσινο χρώμα θέτοντας τον ακροδέκτη PJ0 ίσο με 1.

Όλες οι υπόλοιπες κάρτες κατά την εκκίνηση εκτέλεσης του κώδικα, αφού υλοποιούν τις αρχικοποιήσεις που περιγράφονται στο κεφάλαιο 5.2, περιμένουν να λάβουν το σήμα SYS_RST που σηματοδοτεί την επιτυχημένη τροφοδοσία του συστήματος και άρα τις ικανοποιητικές συνθήκες για την τοπική εκκίνηση του υλικού. Έπειτα η κάθε μία ενεργοποιεί τα τοπικά της τροφοδοτικά και ελέγχει τα αντίστοιχα σήματα Power Good.

5.4 Απενεργοποίηση συστήματος

Σε ότι αφορά την απενεργοποίηση του συστήματος μέσω του διακόπτη της πρόσοψης, εφαρμόζουμε την ίδια λογική για την αντιμετώπιση της αναπήδησης όπως περιγράφηκε σε προηγούμενο κεφάλαιο. Η μόνη διαφορά εντοπίζεται στο γεγονός ότι το κουμπί θα πρέπει να παραμείνει πατημένο για τουλάχιστον 5 δευτερόλεπτα. Έτσι, ο prescaler του μετρητή θα πρέπει να είναι 1024 και η αρχική τιμή του buffer να είναι 29536. Όταν ανιχνευθεί παρατεταμένο πάτημα στο κουμπί ο μικροελεγκτής πρώτα βάζει σε reset όλα τα ολοκληρωμένα θέτοντας το σήμα SYS_RST ίσο με 1, μετά κλείνει τις τοπικές τροφοδοσίες της κάρτας και τέλος, κλείνει τις τροφοδοσίες του συστήματος. Λαμβάνοντας το σήμα SYS_RST, οι κάρτες αφού βάλουν σε reset τα ολοκληρωμένα απενεργοποιούν τα τοπικά τροφοδοτικά τους.

Εκτός του πατήματος του κουμπιού, το σύστημα μπορεί να απενεργοποιηθεί και μέσω της λήψης της εντολής *'AVRpwr'*, η οποία αποστέλλεται μέσω σειριακής από την ARM CPU στο τέλος εκτέλεσης της εντολής *poweroff* του λειτουργικού συστήματος. Στα επόμενα κεφάλαια θα αναλυθεί περισσότερο ο τρόπος που πραγματοποιείται αυτό, όταν γίνει αναφορά στη σειριακή επικοινωνία μεταξύ CPU και ATmega2560. Αφού λάβει την εντολή, ο μικροε-

λεγκτής ακολουθεί τη διαδικασία που περιγράφηκε παραπάνω για την απενεργοποίηση της τροφοδοσίας.

Τέλος, η απενεργοποίηση του συστήματος μπορεί να υλοποιηθεί από οποιοδήποτε διαπιστευμένο χρήστη έχει πρόσβαση στην εφαρμογή πατώντας το πλήκτρο *'Power Off'*. Έτσι, η εφαρμογή του front-end αποστέλλει ένα μήνυμα στο back-end και αυτό με τη σειρά του αποστέλλει την εντολή *'AVRpwr'* μέσω σειριακής στο μικροελεγκτή της κάρτας 1.

5.5 Επανεκκίνηση συστήματος

Όταν ένας χρήστης θέλει να κάνει soft reset οι τροφοδοσίες του συστήματος δεν απενεργοποιούνται, οπότε δεν ελέγχεται ξανά η σταθεροποίηση των τάσεων. Παρόλα αυτά η κάρτα αναγκάζει τις κάρτες 2, 3, 4 να μπουν σε Reset, κάνοντας το σήμα SYS_RST high. Παράλληλα το λειτουργικό σύστημα στην KME επανεκκινεί.

Αφού κρατήσουμε την reset κατάσταση για 5 δευτερόλεπτα ώστε οι μικροελεγκτές των καρτών να έχουν κλείσει τα τοπικά τροφοδοτικά (power down sequence) και να επανέλθουν στην αρχική τους κατάσταση, επαναφέρουμε το σήμα SYS_RST σε χαμηλή στάθμη και το σύστημα εκκινεί κανονικά.

5.6 Εκκίνηση και λειτουργία της KME

5.6.1 Προγραμματισμός γεννήτριας cdcm6208

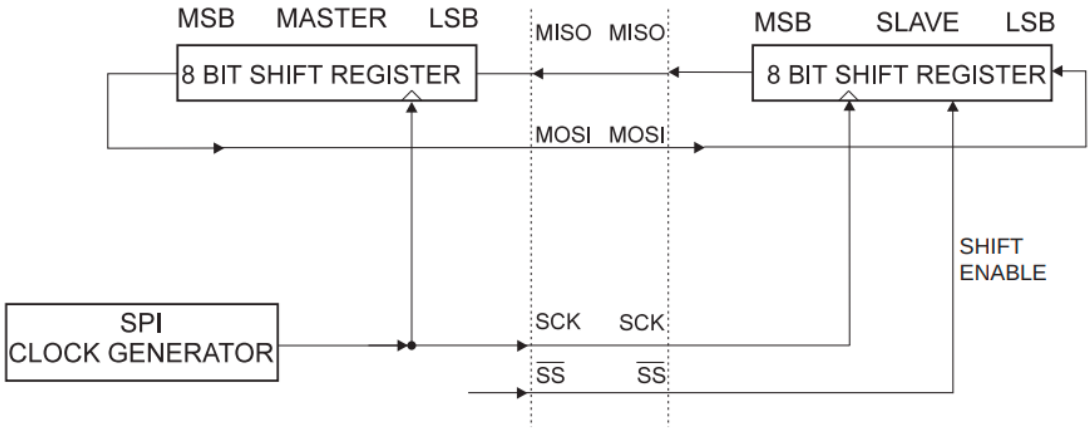
Κάθε μονάδα επεξεργασίας χρειάζεται, όπως είδαμε και παραπάνω, ένα σήμα ρολογιού για να λειτουργήσει. Στις μονάδες επεξεργασίας των AVRs αυτό το σήμα παραγόταν από έναν ταλαντωτή ο οποίος βρισκόταν ήδη στο τσιπ του μικροελεγκτή. Σε πολυεπεξεργαστικά συστήματα όπως η KME δεν επιτρέπεται μία πηγή ταλάντωσης να οδηγεί πολλαπλές εισόδους ρολογιών, διότι αυξάνεται ο θόρυβος και μειώνεται η ακεραιότητα του σήματος ρολογιού. Για αυτόν τον λόγο, προσθέσαμε τη γεννήτρια σημάτων ρολογιού cdcm6208,

5.6.1.1 ATmega2560 SPI Driver

Για τον προγραμματισμό του περιφερειακού γίνεται χρήση της μονάδας SPI. Το ASF παρέχει τους drivers για τη διαχείριση της αντίστοιχης μονάδας.

Στο σημείο αυτό να σημειωθεί πως οι drivers δεν είναι διαθέσιμοι επιλέγοντας το συγκεκριμένο επεξεργαστή στο ASF Wizard. Αντιθέτως, χρειάστηκε αναζήτηση από την πλευρά του προγραμματιστή. Διαλέγοντας μικροελεγκτή ίδιας οικογένειας και αναζητώντας στα example projects που παρέχονται μέσω του ASF (*File → New → Example Project*), βρέθηκαν τα αρχεία *spi_master.h*, *spi_master.c*, *spi_mega.h*, *spi_mega.c* τα οποία ήταν συμβατά με το μικροελεγκτή μας και ενσωματώθηκαν στην υλοποίησή μας. Παρόλα αυτά δεν υπήρχε η συνάρτηση στην οποία μπορούμε να διαβάζουμε και να γράφουμε στον ίδιο κύκλο ρολογιού, πράγμα αναγκαίο για να μπορέσουμε να διαβάσουμε κάποιο καταχωρητή του clock generator. Δεδομένου ότι η μονάδα μας υλοποιείται με έναν καταχωρητή ολίσθησης SPDR, όπως φαίνεται στο σχήμα 5.2, μία ανάγνωση από τον Master πρακτικά μεταφράζεται σε μία οποιαδήποτε εγγραφή. Αυτό ακριβώς υλοποιεί η συνάρτηση *spi_read_packet* του driver αλλά με

dummy πακέτο. Με μικρή τροποποίηση επιτυγχάνουμε να γράφουμε στον slave χρήσιμη πληροφορία και να γράφουμε εξίσου χρήσιμα πράγματα.

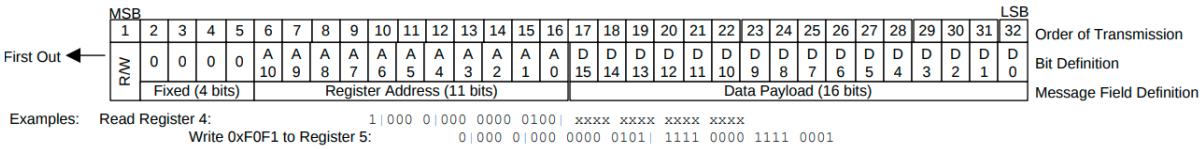


Σχήμα 5.4: *Συνδεσμολογία SPI μεταξύ ATmega2560 και cdcm6208*

5.6.1.2 Serial Peripheral Interface

Για να ορίσουμε πως ο προγραμματισμός του cdcm6208 θα γίνεται μέσω SPI, γειώνουμε τους SI_MODE[1:0] ακροδέκτες του ολοκληρωμένου.

Στην επικοινωνία το περιφερειακό είναι πάντα slave και περιμένει να διαβάσει ένα μήνυμα που περιέχει το Read(1) /Write (0) bit ανάλογα με τη λειτουργία που θέλει ο master να υλοποιήσει, την εσωτερική διεύθυνση του καταχωρητή στην οποία θα υλοποιηθεί η λειτουργία και τα δεδομένα που μεταφέρονται, όπως φαίνεται στο σχήμα 5.5.



Σχήμα 5.5: *Μορφή SPI πακέτου του cdcm6208*

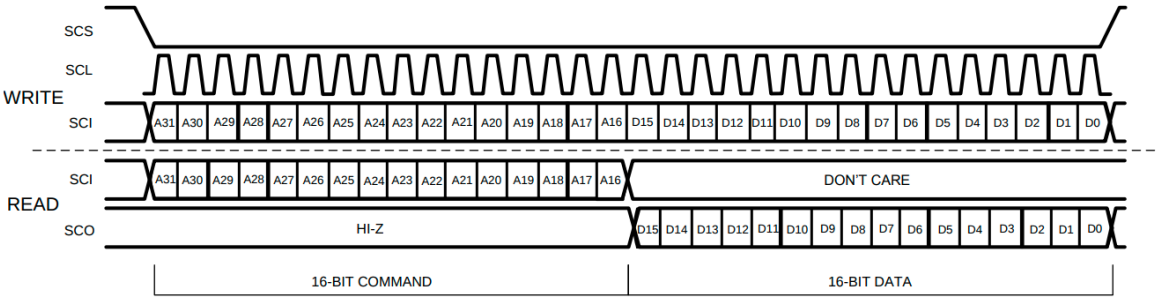
Εγγραφή

Αφού κάνουμε το σήμα του ακροδέκτη PB0 (SCS στο ολοκληρωμένο) του μικροελεγκτή ίσο με 0, μεταδίδουμε τα μηνύματα της μορφής 5.5. Έτσι για την εγγραφή ενός καταχωρητή θέτουμε το πρώτο bit ίσο με 0. Τα επόμενα 4 bit είναι όλα ίσα με 0. Στη συνέχεια, στα υπόλοιπα 11 bit γράφουμε τη διεύθυνση του καταχωρητή στον οποίο θέλουμε να γράψουμε τα τελευταία 16bit δεδομένων. Τέλος, ο μικροελεγκτής τερματίζει τη μεταφορά δεδομένων θέτοντας το σήμα του ακροδέκτη PB0 ίσο με 1.

Ανάγνωση

Αφού κάνουμε το σήμα του ακροδέκτη PB0 (SCS στο ολοκληρωμένο) του μικροελεγκτή ίσο με 0, μεταδίδουμε τα μηνύματα της μορφής 5.5. Έτσι για την εγγραφή ενός καταχωρητή θέτουμε το πρώτο bit ίσο με 1. Τα επόμενα 4 bit είναι όλα ίσα με 0. Στη συνέχεια, στα υπόλοιπα 11 bit γράφουμε τη διεύθυνση του καταχωρητή τον οποίο θέλουμε να διαβάσουμε. Στα υπόλοιπα 16 bit ο μικροελεγκτής διαβάζει τα δεδομένα που του παρέχει το ολοκληρωμένο, τα οποία αποτελούν την πληροφορία που περιέχεται στο ζητούμενο καταχωρητή. Λόγω αυτής της λειτουργίας έγινε η τροποποίηση στους drivers που περιγράφηκε στην αρχή του κεφαλαίου.

Συνοπτικά οι λειτουργίες εγγραφής και ανάγνωσης περιγράφονται στο σχεδιάγραμμα 5.6, από το οποίο παρατηρούμε πως το SPI Mode που πρέπει να χρησιμοποιήσουμε είναι το 0.



Σχήμα 5.6: *Εγγραφή και ανάγνωση μέσω SPI για τη γεννήτρια cdcm6208*

Έπειτα από την περάτωση του προγραμματισμού της γεννήτριας παλμών, οι τιμές του οποίου αποτελούν αποτέλεσμα της σχεδίασης του υλικού, ο μικροελεγκτής περιμένει να λάβει το σήμα PLL_LOCK, γνωστοποιώντας έτσι πως ο προγραμματισμός ήταν επιτυχής. Στην περίπτωση που δεν το λάβει θα πρέπει να επαναλάβει τον προγραμματισμό από την αρχή. Σε αντίθετη περίπτωση συνεχίζει στην εκκίνηση της ΚΜΕ.

5.6.2 Εκκίνηση Κεντρικής Μονάδας Επεξεργασίας

Στο σημείο στο οποίο οι τροφοδοσίες, η θερμοκρασία και τα σήματα ρολογιού είναι τα επιθυμητά η ΚΜΕ μπορεί να εκκινήσει τη λειτουργία της.

Υπάρχουν δύο ειδών εκκινήσεις:

- **Cold Boot (Hard Boot)**: Η πρώτη εκκίνηση της μονάδας κατά την οποία το κύκλωμα Power On Reset (POR) αναγκάζει την CPU να εκκινήσει από την Boot ROM μνήμη.
- **Warm Boot (Soft Boot)**. Αφορά την επανεκκίνηση του λογισμικού. Στην περίπτωση αυτή μπορεί ίσως να τροποποιηθεί το power-on self test (POST) ώστε να αποφευχθούν αρκετοί έλεγχοι. Συνήθως όμως και εδώ υλοποιείται το Cold Boot.

Σε μία πολυεπεξεργαστική μονάδα πρώτα εκκινεί ο πρώτος (primary) πυρήνας και στη συνέχεια μέσω του software ενεργοποιούνται και οι υπόλοιποι. Ο primary πυρήνας έπειτα από την ενεργοποίησή του μέσω των σημάτων RESET, πάσχει από αμνησία· δεν υπάρχει

τίποτε στη μνήμη ώστε να εκτελέσει. Ο πρώτος κώδικας προς εκτέλεση βρίσκεται σε συγκεκριμένη διεύθυνση (συνήθως η 0x00000000), στην οποία βρίσκεται ένα μικρό κομμάτι κώδικα που ονομάζεται Boot ROM. Μέσω των bootstrap pins ο μικροελεγκτής βοηθάει την Boot ROM να εντοπίσει και να εκτελέσει τον first stage boot-loader, U-Boot. Ο boot-loader έχει γνώση του υλικού που διαχειρίζεται και είναι υπεύθυνος για την αρχικοποίηση των περιφερειακών, των μνημών και γενικότερα του υλικού και της φόρτωσης του λειτουργικού συστήματος στη μνήμη. Από τη στιγμή που φορτωθεί σωστά το λειτουργικό σύστημα, αυτό πλέον αναλαμβάνει τη διαχείριση του συστήματος.

Για τη δημιουργία του λειτουργικού μας συστήματος χρησιμοποιήσαμε το Buildroot [23] για να παράξουμε το root file system και το cross-compilation toolchain. Ο πυρήνας Linux [24] χτίζεται αυτόνομα, χρησιμοποιώντας τον cross compiler που δημιούργησε το Buildroot για καλύτερο έλεγχο και ευκολότερη ενσωμάτωση αλλαγών σε αυτόν.

5.6.3 Buildroot

Το Buildroot είναι ένα εργαλείο ανάπτυξης το οποίο αυτοματοποιεί και απλοποιεί τη δημιουργία ενός ενσωματωμένου Linux [25] περιβάλλοντος. Το πετυχαίνει αυτό αφού είναι ικανό μέσω ενός menu να παράξει εύκολα:

- **Το cross-compilation toolchain**: Τα εργαλεία που χρειάζεται ο προγραμματιστής για να δημιουργήσει τα εκτελέσιμα που θα είναι ικανά να τρέξουν στην επιλεγμένη target συσκευή.
- **Το root file system**: Το θεμελιώδες σύστημα αρχείων. Χωρίς αυτό δεν μπορεί να εκκινήσει ο πυρήνας του Linux. Περιέχει τον ίδιο τον πυρήνα στο directory /boot, καθώς επίσης και πολλά σημαντικά αρχεία και scripts τα οποία ορίζουν τη λειτουργία του συστήματος π.χ. inittab, rcS, fstab, profile, interfaces και πολλά άλλα.
- **Τον πυρήνα Linux**: Η καρδιά του λειτουργικού συστήματος. Είναι υπεύθυνος για τη διαχείριση των διεργασιών, της μνήμης, της ασφάλειας και τη γενικότερη διεπαφή των διεργασιών με το υλικό.
- **Τον boot-loader**: Το πρόγραμμα που είναι υπεύθυνο για την αρχικοποίηση του υλικού και τη μεταβίβαση της ροής του προγράμματος στο λειτουργικό σύστημα.

Αφού κατεβάσουμε το buildroot φτιάχνουμε ένα directory για το δικό μας board. Μέσα σε αυτό το directory τρέχουμε την εντολή make menuconfig και εμφανίζεται το μενού με πληθώρα επιλογών.

Η πιο βασική ρύθμιση βρίσκεται στο υπο-μενού Target Options, στο οποίο διαλέγουμε την αρχιτεκτονική (arm) της target συσκευής μας.

Στη συνέχεια, στο μενού toolchain διαλέγουμε τις βιβλιοθήκες της C (glibc) που χρησιμοποιούμε, την έκδοση των binutils (2.35) [26] και του compiler (gcc 9), καθώς επίσης και τα header files του πυρήνα (5.15.43) που θα χρησιμοποιήσουμε στη σχεδίασή μας.

Τέλος επιλέγουμε τα πακέτα τα οποία θέλουμε να ενσωματώσουμε στο περιβάλλον μας στο μενού *Target Packages*. Διαλέγουμε το Busybox, [27] το οποίο αποτελεί ένα minimal bash και παρέχει όλες τις χρήσιμες εντολές όπως ls, echo, cd, ping, stat, fsck, su, stty

και άλλες, σε ένα μόνο εκτελέσιμο. Στη συνέχεια, ανάλογα με την εφαρμογή που υλοποιεί η target συσκευή μας προσθέτουμε τα πακέτα, δημιουργώντας έτσι πρακτικά το δικό μας distribution. Για την ιστοσελίδα μας και τη δυνατότητα σύνδεσης στο περιβάλλον linux, προσθέτουμε τα πακέτα **nodejs** [28] ώστε να υποστηρίζεται η πλατφόρμα node.js για τη διαδικτυακή εφαρμογή, **ssh** [29] ώστε να επιτρέψουμε τη σύνδεση του διαχειριστή του συστήματος στο τερματικό, **gspd** ώστε να ενσωματώσουμε την open source server, gpsd [30] ώστε να μας παρέχει με εύκολο τρόπο την πληροφορία της GPS συσκευής, καθώς επίσης και πακέτα αποσφαλμάτωσης δικτύωσης και λογισμικού όπως τα tcpdump, net-tools, binutils. Τέλος, προσθέτουμε το πακέτο **dcron** [31] το οποίο χρησιμοποιούμε σε συνδυασμό με την υπηρεσία Dynamic DNS (DDNS) που παρέχει το Duck DNS [32].

Για να έχουμε πρόσβαση στο σύστημα μας απο το Internet θα πρέπει είτε να έχουμε ένα domain name, είτε να γνωρίζουμε συνεχώς την public IP του router που είναι συνδεδεμένο το σύστημα. Αυτή η IP δεν είναι σταθερή και αλλάζει σχεδόν καθημερινά. Για να μπορέσουμε να βρίσκουμε τον server μας από οποιοδήποτε σημείο θέλουμε, θα πρέπει να έχουμε πάντα την ενημερωμένη public IP. Αυτό το υλοποιούμε χρησιμοποιώντας το Duck DNS για να δημιουργήσουμε ένα domain name για τον server και με χρήση του dcron, κάθε 5 λεπτά να γίνεται ενημέρωση της public IP στο domain name που έχουμε ορίσει. Τέλος, ενεργοποιώντας το port forwarding στον router που είναι συνδεδεμένο το σύστημα μπορεί ένας χρήστης να συνδεθεί στην εφαρμογή μέσω Internet.

Επιλέγοντας save δημιουργούμε το δικό μας configuration (αρχείο .config), το αποθηκεύουμε και εκτελούμε την εντολή make. Το buildroot πρέπει να έχει πρόσβαση στο διαδίκτυο ώστε να βρει και να κατεβάσει τα χρήσιμα πακέτα κι έτσι να παράξει το root file system και τον cross compiler.

5.6.4 Linux Kernel

Αφού χτίσουμε τον cross compiler, κατεβάζουμε τον πυρήνα που θέλουμε να χτίσουμε και θέτουμε (μέσω export σε linux περιβάλλον) τις εξής μεταβλητές:

- **ARCH** Η αρχιτεκτονική του target μας (arm).
- **CROSS_COMPILER** Το πρόθεμα όλων των παραγόμενων από το buildroot εκτελέσιμων (arm-thesis-linux-gnu-).
- **USER_CROSS_COMPILER** Το ίδιο με το CROSS_COMPILER.
- **KERNEL_PATH** Το path που βρίσκεται ο πυρήνας (/kernel/linux-5.15.43).
- **KSRC** Το ίδιο με το KERNEL_PATH.
- **KVER** Η έκδοση του πυρήνα (5.15.43).
- **INSTALL_MOD_PATH** Το directory που θα γίνουν install τα modules του πυρήνα που θα χτίσουμε.
- **COMPILER_DIR** Το directory που βρίσκεται το board μας και θα χτιστεί ο cross-compiler.

- **CROSS_SYSROOT** Ορίζουμε το system root του target toolchain να χτιστεί στο directory. ${COMPILER_DIR}/host/usr/arm-thesis-linux-gnu/sysroot.

Έπειτα μπαίνουμε στο directory του πυρήνα και εκτελούμε make menuconfig για να ρυθμίσουμε τον πύρήνα μας.

Ένας πυρήνας γενικής χρήσης ενσωματώνει πάρα πολλά kernel modules για να υποστηρίξει συσκευές όλων των ειδών. Σε μία εφαρμογή που ο προγραμματιστής γνωρίζει το υλικό που καλείται να διαχειριστεί στην εφαρμογή του, όλα αυτά τα modules δεν θα χρειαστούν ποτέ. Με αυτόν τον τρόπο, κατά το χτίσιμο του πυρήνα επιλέγουμε όλους τους οδηγούς συσκευών οι οποίοι είναι χρήσιμοι, απενεργοποιώντας τους υπόλοιπους. Έτσι, γλιτώνουμε χρόνο στο χτίσιμο και βελτιστοποιήσουμε το μέγεθος του πυρήνα και του παραγόμενου τελικού root file system, αφού σε αυτό περιέχονται ελάχιστα modules. Βελτιώνουμε με αυτόν τον τρόπο τη διαδικασία του προγραμματισμού στην παραγωγή ενός συστήματος.

Στην περίπτωση μας, κάποιες συσκευές που πρέπει να υποστηρίζονται από το λειτουργικό μας σύστημα είναι: η συσκευή GPS (MAX-M8W), το Real Time Clock, τα LEDs, οι συσκευές USB και η οθόνη.

Χρήσιμα εργαλεία για την ανίχνευση του υλικού και την ενσωμάτωση των αντίστοιχων kernel modules, είναι οι εντολές lsusb, lspci, lshw, lsscsi. Τρέχοντας αυτές στο τερματικού της target συσκευής μπορούμε να δούμε τι συσκευές ανιχνεύονται και στη συνέχεια να ενσωματώσουμε τους drivers κατά το χτίσιμο του πυρήνα είτε ως module, είτε ως built-in κώδικα του πυρήνα.

Κάποια κομμάτια κώδικα, λόγω της σπουδαίας λειτουργίας που επιτελούν, επιλέγονται ως built-in καθώς δεν υπάρχει περίπτωση να μην αξιοποιηθούν, όπως για παράδειγμα η υποστήριξη των ext4, VFAT, NTFS filesystems. Built-in σημαίνει ότι εμπεριέχονται στο εκτελέσιμο του πυρήνα. Με τον τρόπο αυτόν, αυξάνεται το μέγεθος του πυρήνα και η αναβάθμιση τους προϋποθέτει αναβάθμιση όλου του πυρήνα. Παρόλα αυτά, Σε αντίθετη περίπτωση, το χτίσιμο ενός κώδικα ως module δίνει τη δυνατότητα της εύκολης αναβάθμισης και μη χρήσης αυτού σε περιπτώσεις που δεν απαιτείται.

Αποθηκεύουμε το configuration file (.config) και εκτελούμε make; make modules_install ώστε να χτίσουμε τον πυρήνα, τα modules και να τα εγκαταστήσουμε στο επιθυμητό directory που ορίσαμε με τη μεταβλητή INSTALL_MOD_PATH.

Το buildroot μας δίνει τη δυνατότητα να τροποποιήσουμε το τελικό root file system κατά τη δημιουργία του. Η δυνατότητα αυτή περιγράφεται ως overlay directories. Πρακτικά δημιουργούμε ένα δικό μας root file system τροποποιώντας ή προσθέτοντας αρχεία, τα οποία θέλουμε να συμπεριλάβουμε στο τελικό περιβάλλον. Το buildroot σαν τελευταία διαδικασία μετά την παραγωγή του δικού του αρχικού filesystem αντιγράφει τα αρχεία των overlay directories, συμπεριλαμβάνοντάς τα στο τελικό σύστημα αρχείων. Για τη ρύθμιση αυτή διαλέγουμε (*System Configuration → Root file system overlay directories*) γράφοντας το path του δικού μας file system.

Στο δικό μας overlay file system προσθέτουμε τον πυρήνα που χτίσαμε στο */boot* directory μαζί με τα modules */lib/modules/5.15.43* directory

5.7 Παρακολούθηση συστήματος

Κατά την παρακολούθηση δύο τιμές είναι πολύ σημαντικές για την προστασία του υλικού, οι τάσεις και οι θερμοκρασίες που αναπτύσσονται.

5.7.1 Μέτρηση Θερμοκρασίας

Η μέτρηση της θερμοκρασίας γίνεται μέσω των αναλογικών αισθητήρων mcp9808, οι οποίοι είναι τοποθετημένοι σε σημεία που αναμένεται να αναπτυχθεί μεγάλη θερμοκρασία, όπως για παράδειγμα δίπλα από την κεντρική μονάδα επεξεργασίας. Ο τρόπος ανάγνωσης των θερμομέτρων προϋποθέτει τη σωστή χρήση της μονάδας TWI, η οποία υλοποιεί την I^2C επικοινωνία.

5.7.1.1 Χρήση μονάδας TWI του ATmega2560 σε master mode

Για να κάνουμε χρήση της μονάδας αρχικά ενεργοποιούμε το σήμα ρολογιού clk_{TWI} μέσω του Power Reduction Register 0 (PRR0), γράφοντας την τιμή 0 στο πεδίο PRTWI. Στη συνέχεια, ενεργοποιούμε και τις διακοπές υλικού, καθώς δεν μπορεί να υλοποιηθεί η επικοινωνία δίχως αυτές, εάν δεν θέλουμε να δεσμεύουμε το μικροελεγκτή μόνο σε αυτή τη λειτουργία.

Η μονάδα αρχικοποιείται μέσω της συνάρτησης *twi_master_init* που παρέχει το ASF. Όρισμα της είναι η επιθυμητή ταχύτητα επικοινωνίας (400KHz). Έπειτα κάνει τις αντίστοιχες εγγραφές στους καταχωρητές TWCR, TWSR, TWBR, αφού υπολογίσει την τιμή για τον καταχωρητή TWBR μέσω της macro TWI_CLOCK_RATE, λύνοντας δηλαδή την παρακάτω εξίσωση ως προς TWBR.

$$SCL\ Frequency = \frac{clk_{CPU}}{16 + 2 * TWBR * 4^{TWPS}}$$

Η μονάδα έχει τέσσερις λειτουργίες μετάδοσης: master πομπός, master δέκτης, slave πομπός, slave δέκτης. Με την εγγραφή και ανάγνωση των καταχωρητών TWINT, TWEA, TWSTA, TWSTO, TWWC, TWEN, TWIE, η μονάδα παράγει τα σήματα όπως περιγράφηκαν στο κεφάλαιο 2 και βρίσκεται στις παραπάνω τέσσερις καταστάσεις λειτουργίας.

Η πολύπλοκη υλοποίηση πραγματοποιείται από τους οδηγούς που παρέχει το ASF, θεμέλια των οποίων αποτελούν οι δύο συναρτήσεις *twi_master_read(), twi_master_write()*. Πριν την εκτέλεση των παραπάνω εντολών το μόνο που πρέπει να καθοριστεί από τον προγραμματιστή είναι τα παραμετροποιήσιμα πεδία του πακέτου. Αυτό γίνεται με τη δομή *twi_package_t* που περιέχει τα πεδία:

- **chip** Σε ποια διεύθυνση βρίσκεται ο slave.
- **addr_length** Υπάρχει εσωτερική διεύθυνση καταχωρητή του slave στην οποία θέλουμε να απευθυνθούμε; Αν ναι, ποιο είναι το μήκος του σε bytes;

- **addr** Σε ποια εσωτερική διεύθυνση καταχωρητή του slave θέλουμε να εκτελέσουμε την εγγραφή/ανάγνωση.
- **buffer** Πού βρίσκονται τα δεδομένα που θα γραφτούν στον slave ή πού θα αποθηκευτούν τα δεδομένα κατά την ανάγνωση του slave.
- **length** Πόσα bytes δεδομένων θέλουμε να γράψουμε στον slave.

Για παράδειγμα, για να διαβάσουμε τον καταχωρητή $T_{AMBIENT}(0x05)$ του ολοκληρωμένου mcp9808 το οποίο έχει διεύθυνση 0x34, ορίζουμε τις τιμές:
chip=0x34
addr_length=1
addr=0x05
length=0x02
buffer=&master_read_buffer

Το αποτέλεσμα της ανάγνωσης αποθηκεύεται στον buffer master_read_buffer που δίνουμε ως όρισμα στη δομή.

5.7.1.2 Προγραμματισμός mcp9808

Ο αισθητήρας mcp9808 έχει αρκετούς καταχωρητές στους οποίους μπορεί να αναφερθεί ο προγραμματιστής. Για την επιλογή του εσωτερικού καταχωρητή γίνεται χρήση ενός καταχωρητή ¨δείκτη¨ (pointer register), στον οποίο αρχικά θα πρέπει ο προγραμματιστής να γράψει την τιμή του καταχωρητή που θέλει να διαβάσει/γράψει στην επόμενη μετάδοση. Αν δηλαδή θέλει να διαβάσει τον καταχωρητή θερμοκρασίας (0b0101), θα πρέπει πρώτα να γράψει στον pointer register την τιμή 0101 και στη συνέχεια να διαβάσει 2 bytes μέσω του διαύλου I^2C.

Η ανάγνωση της θερμοκρασίας του αισθητήρα περιγράφεται σαν διαδικασία παραπάνω, αλλά η τιμή που διαβάζουμε από τον καταχωρητή θερμοκρασίας $T_{AMBIENT}$ χρειάζεται μετάφραση. Για τη μετάφραση αυτή διαβάζουμε αρχικά το πρόσημο της τιμής της θερμοκρασίας (bit 12) και στη συνέχεια υπολογίζουμε τη θερμοκρασία ως εξής:

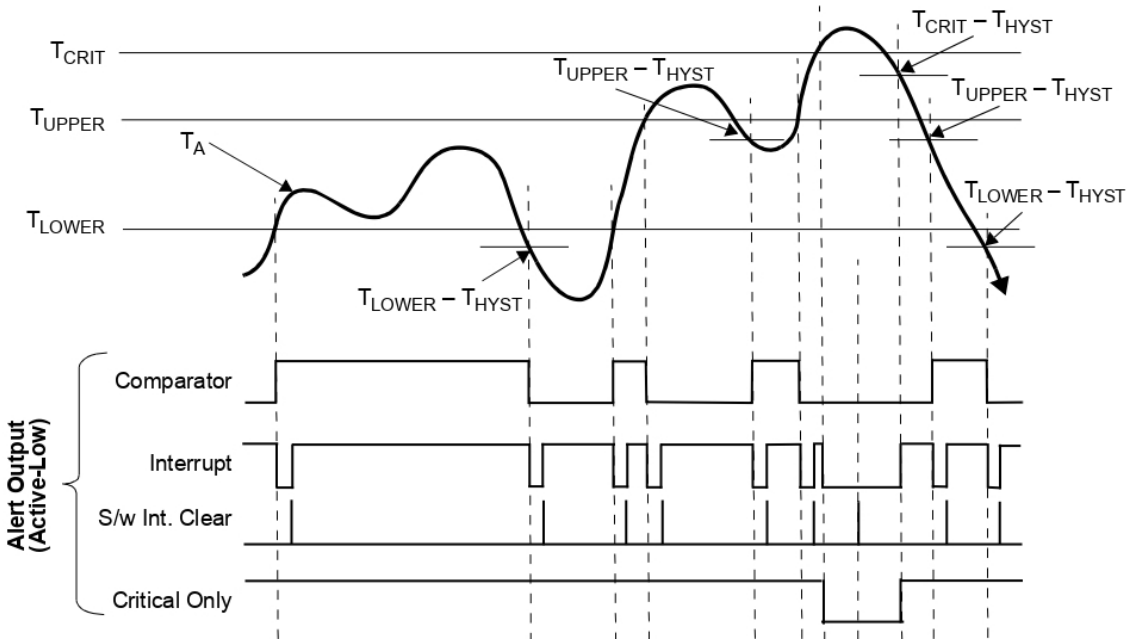
```
U16 Temp = ((bit15:8) & 0x1F) << 8 + (bit7:0);
if (SIGN == 1)
{
Temp -=8192; //13 bit one's complement
}
TempCelcius = Temp * 0.0625; // LSB
```

bit 15	14	13	12	11	10	9	bit 8
T_A *vs* T_{CRIT}	T_A *vs* T_{UPPER}	T_A *vs* T_{LOWER}	SIGN	$2^{7\circ}C$	$2^{6\circ}C$	$2^{5\circ}C$	$2^{4\circ}C$

bit 7	6	5	4	3	2	1	bit 0
$2^{3\circ}C$	$2^{2\circ}C$	$2^{1\circ}C$	$2^{0\circ}C$	$2^{-1\circ}C$	$2^{-2\circ}C$	$2^{-3\circ}C$	$2^{-4\circ}C$

Καταχωρητής Θερμοκρασίας $T_{Ambient}$

Τα 3 πιο σημαντικά bit (15:13) αφορούν τις συγκρίσεις μεταξύ θερμοκρασίας και των αποδεκτών ορίων που μπορούμε να προγραμματίζουμε τον αισθητήρα να λειτουργεί. Παρόλα αυτά, δεν ελέγχουμε αυτά τα bits διότι προγραμματίζουμε τον αισθητήρα να το κάνει για εμάς και να εξάγει το σήμα alert όταν οι συνθήκες $T_{LOWER} < T_{Ambient}$, $T_{Ambient} < T_{UPPER}$, $T_{Ambient} < T_{CRITICAL}$ δεν ικανοποιούνται. Αυτό γίνεται μέσω της εγγραφής της τιμής 0x18 στον configuration register, ενεργοποιώντας έτσι την εξαγωγή του σήματος alert μέσω της μονάδας συγκριτή (comparator). Όταν η θερμοκρασία δεν είναι στα επιτρεπτά όρια το σήμα αυτό ενεργοποιείται σε χαμηλή στάθμη (active low), γνωστοποιώντας έτσι στο μικροελεγκτή μας πως μία από τις συνθήκες σταμάτησε να ισχύει. Στην περίπτωση του (comparator) δεν υπάρχει διαφορά στις συνθήκες $T_{Ambient} < T_{UPPER}$ και $T_{Ambient} < T_{CRITICAL}$. Η κυματομορφή της εξόδου alert φαίνεται στο σχήμα 5.7, όπου το T_{HYST} το θέτουμε ίσο με 0 μέσω του configuration register.



Σχήμα 5.7: *Κυματομορφή σήματος εξόδου alert του αισθητήρα mcp9808 στις διάφορες συνθήκες λειτουργίας*

Τα όρια θερμοκρασιών T_{LOWER}, T_{UPPER}, $T_{CRITICAL}$ βρίσκονται στους αντίστοιχους καταχωρητές T_{UPPER}, T_{LOWER}, $T_{CRITICAL}$, στους οποίους προγραμματίζουμε τις τιμές -40°C, +85°C, +85°C αντίστοιχα έπειτα από τη μετάφραση.

bit 15	14	13	12	11	10	9	bit 8
-	-	-	SIGN	$2^{7}°C$	$2^{6}°C$	$2^{5}°C$	$2^{4}°C$

bit 7	6	5	4	3	2	1	bit 0
$2^{3}°C$	$2^{2}°C$	$2^{1}°C$	$2^{0}°C$	$2^{-1}°C$	$2^{-2}°C$	-	-

Καταχωρητές Θερμοκρασίας $T_{UPPER}/T_{CRITICAL}/T_{LOWER}$

Τον προγραμματισμό όλων των αισθητήρων για την εξαγωγή των σημάτων alert σε συγκεκριμένα όρια, καθώς επίσης και την ανάγνωση των θερμοκρασιών από τον καταχωρητή

θερμοκρασίας, τα αναλαμβάνει η κάρτα 1. Έπειτα, μέσω των σημάτων alert, οι κάρτες 2, 3, 4 μπορούν και λαμβάνουν την πληροφορία ότι η θερμοκρασία στην κάρτα τους βρίσκεται μέσα στα αποδεκτά όρια λειτουργίας.

5.7.2 Μέτρηση τάσεων

5.7.2.1 Χρήση μονάδας ADC του μικροελεγκτή ATmega2560

Η μονάδα ADC μετατρέπει ένα αναλογικό σήμα σε ψηφιακή τιμή μεγέθους 10-bit, όπου η μικρότερη τιμή αναπαριστά τη γείωση και η μεγαλύτερη τιμή αναπαριστά την τάση αναφοράς $V_{REFERENCE}$. Ορίζουμε ως τάση αναφοράς την εσωτερική τάση 1.1V και προσαρμόζουμε τα σήματα εισόδου με τους αντίστοιχους διαιρέτες, ώστε οι τάσεις να έρχονται στο εύρος τιμών 0 έως 1.1V. Αυτό γίνεται γράφοντας στο πεδίο REFS1:0 του καταχωρητή ADMUX την τιμή 10.

Η μονάδα μπορεί να μετατρέπει έως και 16 διαφορετικά αναλογικά σήματα, τα οποία είναι συνδεδεμένα είτε στην PORTF είτε στην PORTK. Η επιλογή του καναλιού που θα αποτελέσει την αναλογική είσοδο στη μονάδα γίνεται μέσω των καταχωρητών ADMUX, ADCSRB. Στην περίπτωσή μας, όπου χρειάζεται να ελέγξουμε 5 διαφορετικές τιμές τάσεων στα πρώτα 5 κανάλια (ADC0-ADC5), θα πρέπει πριν από κάθε μέτρηση να αλλάξουμε το κανάλι εισόδου. Αυτό γίνεται γράφοντας στον καταχωρητή ADMUX στα πεδία MUX4:0 την τιμή του καναλιού που επιλέγουμε σύμφωνα με τον πίνακα 5.2.

MUX5:0	Input	MUX5:0	Input
000000	ADC0	100000	ADC8
000001	ADC1	100001	ADC9
000010	ADC2	100010	ADC10
000011	ADC3	100011	ADC11
000100	ADC4	100100	ADC12
000101	ADC5	100101	ADC13
000110	ADC6	100110	ADC14
000111	ADC7	100111	ADC15

Πίνακας 5.2: *Επιλογή αναλογικής εισόδου στη μονάδα ADC του ATmega2560*

Έτσι παίρνουμε το επιθυμητό αποτέλεσμα γράφοντας:

```
ADMUX |= input_channel
```

Για παράδειγμα για το κανάλι 3 (PF3) γράφουμε ADMUX |= 3. Εδώ θα πρέπει να σημειωθεί πως η εκτέλεση της παραπάνω εντολής είναι εφικτή μόνο στα πρώτα 8 κανάλια, ενώ για τα υπόλοιπα 8 χρειάζεται να ορίσουμε την τιμή 1 στο πεδίο MUX5 του καταχωρητή ADCSRB σύμφωνα με τον πίνακα 5.2 και οι τιμές των MUX4:0 αλλάζουν.

Για τη λειτουργία της μονάδας πρέπει να γράψουμε την τιμή 0 στο πεδίο PRADC του καταχωρητή Power Reduction Register 0 (PRR0) προκειμένου να ενεργοποιήσουμε τα σήματα ρολογιού και να γράψουμε την τιμή 1 στο πεδίο ADEN του καταχωρητή ADCSRA, ώστε να ενεργοποιήσουμε τη μονάδα.

Για τη σωστή λειτουργία του κυκλώματος απαιτούνται συχνότητες ρολογιού μεταξύ 50KHz και 200KHz. Για να πετύχουμε τις αποδεκτές ταχύτητες, η μονάδα παρέχει prescalers με

συχνότητα παραγόμενου ρολογιού clk_{ADC} έως και 128 φορές κάτω από το clk_{CPU}. Διαλέγουμε τον παράγοντα διαίρεσης της ταχύτητας του ρολογιού να είναι 128 γράφοντας στα πεδία ADPS2:0 του ADCSRA την τιμή 0b111.

Διαλέγουμε τάση αναφοράς την εσωτερική τάση 1.1V γράφοντας στα πεδία REFS1:0 του καταχωρητή ADMUX την τιμή 0b10.

Έπειτα για την εκκίνηση της μετατροπής γράφουμε στο πεδίο ADSC του καταχωρητή ADCSRA την τιμή 1 και περιμένουμε μέχρι η τιμή να γυρίσει σε 0, πράγμα που σημαίνει ότι η μετατροπή έχει τελειώσει.

Το αποτέλεσμα της μέτρησης αποθηκεύεται στους καταχωρητές ADCH, ADCL. Η τάση που μετρήθηκε σαν αναλογική είσοδος προκύπτει από την εξίσωση:

$$V_{INPUT} = \frac{ADC * V_{REFERENCE}}{1024}$$

Χρήσιμοι καταχωρητές

bit	7	6	5	4	3	2	1	0
ADCSRB	-	ACME	-	-	MUX5	ADTS2	ADTS1	ADTS0

bit	7	6	5	4	3	2	1	0
ADCSRA	ADEN	ADSC	ADATE:	ADIF	ADIE	ADPS2	ADPS1	ADPS0

bit	7	6	5	4	3	2	1	0
ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

5.7.2.2 Χρήση μονάδας ADC του μικροελεγκτή ATxmega128D4

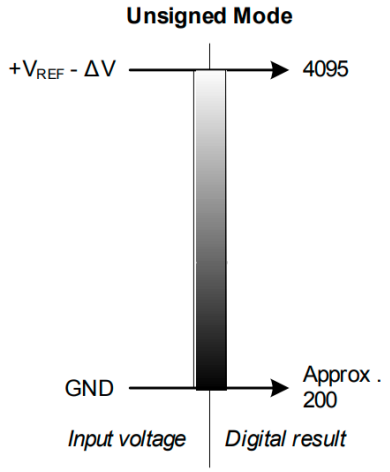
Η μονάδα του μικροελεγκτή ATxmega128D4 παρέχει περισσότερες δυνατότητες από αυτές του ATmega2560. Αρχικά διότι η πληροφορία εκφράζεται στη μορφή 12-bit δεδομένων, παρέχοντας έτσι μεγαλύτερη ακρίβεια. Επιπλέον η μονάδα αυτή υποστηρίζει τη μέτρηση και αρνητικών τιμών, πράγμα που δεν χρησιμοποιούμε στην περίπτωση μέτρησης της τάσης. Τέλος, η μονάδα δύναται να εκτελέσει μέχρι και 300000 μετατροπές το δευτερόλεπτο, σε σχέση με τη μονάδα του ATmega2560 που μπορεί να μετατρέψει 77000.

Στην προκειμένη περίπτωση, για μετρήσεις τάσεων έχουμε σήματα εισόδου single-ended στις PORTA, PORTB.

Η ενεργοποίηση της μονάδας προκύπτει έπειτα από την εγγραφή της τιμής 0 στο πεδίο ADC στον καταχωρητή Power Reduction Port A (PRPA).

Ρυθμίζουμε τη μονάδα μας ώστε να μας παρέχει 12-bit μη αρνητική πληροφορία, γράφοντας στον καταχωρήτη CTRLB την τιμή 0 στο πεδίο CONVMODE και την τιμή 00 στα πεδία RESOLUTION1:0.

Τα όρια τάσεων που μπορούν να μετρηθούν μέσω της μονάδας είναι μεταξύ 0 και V_{REF}-ΔV, όπως φαίνεται στο σχήμα 5.8.



Σχήμα 5.8: *Μετρήσεις δίχως πρόσημο της μονάδας ADC του ATxmega128D4*

Για να μπορέσει η μονάδα να αντιληφθεί την εναλλαγή μεταξύ αρνητικού και θετικού σήματος (zero-cross detection) στις μετατροπές δίχως πρόσημο, χρειάζεται η αφαίρεση μίας μικρής ποσότητας ΔV=0.05 * V_{REF}. Αυτό πρακτικά μεταφράζεται στην ψηφιακή τιμή 200, όπου αποτελεί την ψηφιακή τιμή για μηδενική αναλογική τάση εισόδου.

Ορίζουμε τάση αναφοράς την εσωτερική τάση 1V και προσαρμόζουμε τα σήματα εισόδου με τους αντίστοιχους διαιρέτες προκειμένου οι τάσεις να έρχονται στο εύρος τιμών 0 έως 0.95V. Γράφουμε στο πεδίο REFSEL2:0 καταχωρητή REFCTRL την τιμή 000 και την τιμή 1 στο πεδίο BANDGAP.

Γράφουμε την τιμή 0 στο σημείο FREERUN του καταχωρήτη CTRLB και 0 στον καταχωρητή EVCTRL, ώστε κάθε μετατροπή να είναι χειροκίνητη κατά τη ροή του προγράμματος την ώρα που τη χρειαζόμαστε.

Επειδή δεν εκτελούμε πολλές μετατροπές, μπορούμε να θυσιάσουμε την ταχύτητα εκτέλεσης για την καλύτερη απόδοση και τη μικρότερη κατανάλωση. Έτσι, για το χρονισμό της μονάδας γράφουμε στον καταχωρητή PRESCALER την τιμή 111 για να να διαιρέσουμε τη συχνότητα εισόδου 32MHz κατά 512 για να παραγάγουμε συχνότητα 62KHz. Επίσης για τη μείωση της κατανάλωσης γράφουμε την τιμή 11 στο πεδίο CURRLIMIT του καταχωρητή CTRLB1:0 με σκοπό να μειώσουμε κατά το μέγιστο την κατανάλωση ενέργειας από τη μονάδα.

Έπειτα από τη χειροκίνητη έναρξη της μετατροπής ρυθμίζουμε τη μονάδα να παράξει μία διακοπή υλικού κατά την περάτωση της μετατροπής γράφοντας την τιμή 00 στο πεδίο INTMODE του καταχωρητή INTCTRL.

Τέλος ενεργοποιούμε τη λειτουργία της μονάδας γράφοντας την τιμή 1 στο πεδίο ENABLE του καταχωρητή CTRLA.

Για την επιλογή του καναλιού γράφουμε στον καταχωρητή MUXCTRL στο πεδίο MUXPOS την είσοδο την οποία θέλουμε να χρησιμοποιήσουμε, σύμφωνα με τον πίνακα 5.3. Ξεκινάμε τη μετατροπή γράφοντας στον καταχωρητή CTRL στο πεδίο START την τιμή 1.

Το αποτέλεσμα της μετατροπής τοποθετείται στους καταχωρητές RESH, RESL και η τάση

MUX5:0	Input	MUX5:0	Input
0000	ADC0	1000	ADC8
0001	ADC1	1001	ADC9
0010	ADC2	1010	ADC10
0011	ADC3	1011	ADC11
0100	ADC4	1100	ADC12
0101	ADC5	1101	ADC13
0110	ADC6	1110	ADC14
0111	ADC7	1111	ADC15

Πίνακας 5.3: *Επιλογή αναλογικής εισόδου στη μονάδα ADC του ATxmega128D4*

προκύπτει σύμφωνα με την εξίσωση:

$$V_{INPUT} = \frac{RES * V_{REF}}{2048} - 0.05 * V_{REF}$$

Χρήσιμοι καταχωρητές

bit	7	6	5	4	3	2	1	0
CTRLB	-	CURRLIMIT[1:0]		CONVMODE	FREERUN	RESOLUTION[1:0]		-

bit	7	6	5	4	3	2	1	0
REFCTRL	-	REFSEL[2:0]			-	-	BANDGAP	TEMPREF

bit	7	6	5	4	3	2	1	0
PRESCALER	-	-	-	-	-	PRESCALER[2:0]		

bit	7	6	5	4	3	2	1	0
CTRL	START	-	-	GAIN[2:0]			INPUTMODE[1:0]	

bit	7	6	5	4	3	2	1	0
MUXCTRL	-	MUXPOS[3:0]				MUXNEG[2:0]		

5.7.2.3 Χρήση μονάδας TWI του ATxmega128D4 σε master mode

Για τον προγραμματισμό των ολοκληρωμένων ina260 και MAX-M8W χρειάζεται να κάνουμε χρήση της μονάδας TWI σε master mode. Για το λόγο αυτό, ενσωματώνουμε τους οδηγούς της μονάδας TWI που παρέχονται στο ASF.

Η ενεργοποίηση της μονάδας προκύπτει έπειτα από την εγγραφή της τιμής 0 στο πεδίο TWI στον καταχωρητή Power Reduction Port A (PRPC).

Στη συνέχεια αρχικοποιούμε τη μονάδα κάνοντας χρήση της συνάρτησης twi_master_init. η οποία γράφει στους καταχωρητές BAUD, CTRLA, STATUS, PMIC τις απαραίτητες

τιμές προκειμένου η μονάδα να λειτουργεί στα 400KHz, παράγοντας διακοπές υλικού στις λειτουργίες ανάγνωσης και εγγραφής. Θέτουμε μέσω του ορίσματος CONF_TWIM_INTLVL τις διακοπές υλικού να έχουν υψηλή προτεραιότητα.

Η ανάγνωση και εγγραφή σε κάποια συσκευή υλοποιείται με τον ίδιο τρόπο όπως και στον ATmega2560, κάνοντας χρήση των συναρτήσεων *twi_master_read(), twi_master_write()*. Πριν την εκτέλεση παραμετροποιούμε αντίστοιχα τα πεδία του πακέτου *twi_package_t*.

5.7.2.4 Προγραμματισμός ina260

Για τον προγραμματισμό του ολοκληρωμένου καθορίζουμε τις επιθυμητές συνθήκες λειτουργίας και τις συνθήκες ένδειξης σφάλματος.

Για τις συνθήκες λειτουργίας γράφουμε στον καταχωρητή Configuration Register (00h) στο πεδίο AVG12:0 την τιμή 011, ώστε η κάθε μέτρηση να προκύπτει από τη δειγματοληψία 64 δειγμάτων. Οι default τιμές έπειτα από το Power-On reset είναι ικανοποιητικές.

Για τις ρυθμίσεις που αφορούν τις ενδείξεις σφάλματος γράφουμε την τιμή 1 στο πεδίο OCL, διότι θέλουμε να ενεργοποιηθεί το σήμα ALERT στην περίπτωση που το ρεύμα που μετρηθεί κατά την τελευταία μετατροπή ξεπεράσει την τιμή που υπάρχει στον καταχωρητή Alert Limit Register. Επίσης, ορίζουμε το σήμα που εξάγεται κατά την ενεργοποίηση να είναι σε υψηλή στάθμη (active high). Το τελευταίο πράγμα που πρέπει να κάνουμε είναι να γράψουμε την τιμή 0.5 (mA) στον καταχωρητή Alert Limit Register. Για το λόγο αυτό, μετατρέπουμε την τιμή αυτή σε ψηφιακή διαιρώντας την με το 0.00125. Η τιμή αυτή αποτελεί τη μικρότερη μονάδα μέτρησης που χρησιμοποιεί το ολοκληρωμένο και αφορά την αναλογική πληροφορία που μεταφέρει ένα ψηφιακό bit.

Έπειτα, διαβάζουμε τις τιμές των καταχωρητών Current Register, Power Voltage Register και πολλαπλασιάζοντας την ψηφιακή τιμή με 0.00125 προκύπτει η πληροφορία της τάσης και του ρεύματος στην κάρτα μας.

5.8 Συλλογή και εξαγωγή πληροφοριών συστήματος

5.8.1 Μονάδα TWI για το μικροελεγκτή ATmega2560 σε slave mode - Ανάγνωση κάρτας 2

Οι οδηγοί της μονάδας που περιγράφηκαν σε προηγούμενο κεφάλαιο περιέχουν και τη δυνατότητα της λειτουργίας της μονάδας σε slave mode.

Αρχικά ενεργοποιούμε τα σήματα ρολογιού στη μονάδα μέσω ης συνάρτησης *power_twi_enable*. Στη συνέχεια, τρέχουμε την εντολή *twi_slave_init* δίνοντας ως όρισμα τη διεύθυνση στην οποία θα χρησιμοποιεί ο master για να απευθυνθεί στη συσκευή. Τέλος, αρχικοποιούμε τους δύο buffers που θα χρησιμοποιεί ο master στη συσκευή μας, τον πρώτο για την εγγραφή και τον δεύτερο για την ανάγνωση δεδομένων.

Η θέση μνήμης των καταχωρητών για την ανάγνωση και εγγραφή δεδομένων πρέπει να δοθεί σαν όρισμα στη δομή *slave_data_buffer_t*, στα πεδία *tx_buffer* και *rx_buffer* αντίστοιχα. Έτσι κατά την ανάγνωση ο master θα διαβάσει τα περιεχόμενά του *tx_buffer* της συσκευής μας. Για να χρησιμοποιήσει ο driver τη δομή με τους δείκτες στους καταχωρητές, τρέχουμε την εντολή *twi_slave_start* με όρισμα το δείκτη στη δομή αυτή.

Η χρήση του καταχωρητή *tx_buffer* φέρει δύο διαφορετικές πληροφορίες. Κατά την πρώτη ανάγνωση, όταν το σύστημα εκκινήσει, ο μικροελεγκτής «συστήνεται», μεταβιβάζοντας στον master το μοναδικό αναγνωριστικό της κάρτας και την έκδοση του υλικολογισμικού που εκτελείται, όπως φαίνεται στον πίνακα 5.4. Μετά τη δεύτερη ανάγνωση μεταβιβάζει στον καταχωρητή τις πληροφορίες της τοπικής παρακολούθησης της κάρτας, όπως φαίνεται στον πίνακα 5.5.

byte	5	4 3	2	1 0
	LENGTH	NAME	ID	FIRMWARE VERSION

π.χ

LENGTH=5
NAME=C2 -1 εάν ανεπιτυχής probe
ID=2 -1 εάν ανεπιτυχής probe
FW VERSION=1.89 -1 εάν ανεπιτυχής probe

Πίνακας 5.4: *Πακέτο δεδομένων κατά την πρώτη ανάγνωση της κάρτας 2*

byte	11	10 9	8 7	6 5	4 3	2 1	0
	LENGTH	1.2V	1.5V	2.5V	1.8V	1.0V	PG

π.χ

LENGTH=11 (bytes)
1.2V=1.18 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
1.5V=1.49 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
2.5V=2.51 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
1.8V=1.79 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
1.0V=1.01 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
PG=xxx11111 bit4→ $1.2V, ..., bit0 \rightarrow 1.0V$
μηδενική τιμή δηλώνει πως το αντίστοιχο σήμα PG του τροφοδοτικού είναι not active

Πίνακας 5.5: *Πακέτο δεδομένων κατά την ανάγνωση της κάρτας 2*

5.8.2 Μονάδα TWI για το μικροελεγκτή ATxmega128D4 σε slave mode

Οι οδηγοί της μονάδας που περιγράφηκαν σε προηγούμενο κεφάλαιο περιέχουν και τη δυνατότητα της λειτουργίας της μονάδας σε slave mode.

Αρχικά ενεργοποιούμε τη μονάδα μέσω της εντολής *power_twi_enable*. Έπειτα δημιουργούμε ένα στιγμιότυπο της δομής *TWI_Slave_t*. Η δομή αυτή περιέχει όλες τις απαραίτητες πληροφορίες για την υλοποίηση της I^2C επικοινωνίας, αλλά το πιο σημαντικό είναι οι δύο *buffers receivedData* και *sendData*, που αποτελούν τους καταχωρητές στους οποίους θα γράψει/διαβάσει η master κάρτα. Εκτελούμε τη συνάρτηση *TWI_SlaveInitializeDriver*, δίνοντας ως όρισμα την παραπάνω δομή, τους ακροδέκτες που χρησιμοποιούμε για την επικοινωνία (TWIC) και τη συνάρτηση που θα εκτελεί ο driver όταν η master κάρτα υλοποιεί μία εγγραφή στην κάρτα μας.

Η μονάδα μας αρχικοποιείται εκτελώντας τη συνάρτηση *TWI_SlaveInitializeModule*. Ορίσματα αποτελούν η διεύθυνση στην οποία θα απευθύνεται η master συσκευή, η δομή που

περιγράφηκε παραπάνω και η προτεραιότητα των διακοπών υλικού, σύμφωνα με τις οποίες θα εκτελεί η κεντρική μονάδα επεξεργασίας του μικροελεγκτή, τις εντολές στις ρουτίνες διακοπών.

Η χρήση του καταχωρητή *sendData* φέρει δύο διαφορετικές πληροφορίες. Κατά την πρώτη ανάγνωση, όταν το σύστημα εκκινήσει, ο μικροελεγκτής «συστήνεται», μεταβιβάζοντας στον master το μοναδικό αναγνωριστικό της κάρτας και την έκδοση του υλικολογισμικού την οποία εκτελεί. Μετά τη δεύτερη ανάγνωση μεταβιβάζει τις πληροφορίες της τοπικής παρακολούθησης της κάρτας.

5.8.2.1 Πακέτα δεδομένων κατά την ανάγνωση της κάρτας 3

byte	5	4 3	2	1 0
	LENGTH	NAME	ID	FIRMWARE VERSION

π.χ

LENGTH=5
NAME=C3 -1 εάν ανεπιτυχής probe
ID=3 -1 εάν ανεπιτυχής probe
FW VERSION=1.26 -1 εάν ανεπιτυχής probe

Πίνακας 5.6: *Πακέτο δεδομένων κατά την πρώτη ανάγνωση της κάρτας 3*

byte 15	14 13	12 11	10 9	8 7	6 5	4 3	2 1	0
LENGTH	V_{ina260}	I_{ina260}	1.2V	1.8V	2.5V	3.3V	5.0V	PG

π.χ

LENGTH=15 (bytes)
V$_{ina260}$ = 4.99 (μέτρηση), 0 εάν μετρηθεί εκτός ορίων, -1 εάν υπάρχει πρόβλημα κατά την ανάγνωση
I$_{ina260}$ = 0.34(*mA* μέτρηση), 0 εάν μετρηθεί εκτός ορίων, -1 εάν υπάρχει πρόβλημα κατά την ανάγνωση
1.2V=1.20 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
1.8V=1.79 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
2.5V=2.51 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
3.3V=3.29 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
5.0V=4.99 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
PG=xxx11111 bit4→ 1.2*V*, ..., *bit*0 → 5.0*V*
μηδενική τιμή δηλώνει πως το αντίστοιχο σήμα PG του τροφοδοτικού είναι not active

Πίνακας 5.7: *Πακέτο δεδομένων κατά την ανάγνωση της κάρτας 3*

5.8.2.2 Πακέτα δεδομένων κατά την ανάγνωση της κάρτας 4

byte	5	4 3	2	1 0
	LENGTH	NAME	ID	FIRMWARE VERSION

π.χ

LENGTH=5
NAME=C4 -1 εάν ανεπιτυχής probe
ID=4 -1 εάν ανεπιτυχής probe
FW VERSION=1.50 -1 εάν ανεπιτυχής probe

Πίνακας 5.8: *Πακέτο δεδομένων κατά την πρώτη ανάγνωση της κάρτας 4*

byte	7	6	5	4	3	2	1	0
	LENGTH	5.0V		3.3V		3.3V		PG

π.χ

LENGTH=7 Bytes
5.0V=5.01 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
3.3V=3.31 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
3.3V=3.33 (μέτρηση) ή 0 εάν μετρηθεί εκτός ορίων
PG=xxxxx111 bit2→ 5.0V, ..., $bit0 \rightarrow 3.3V$
μηδενική τιμή δηλώνει πως το αντίστοιχο σήμα PG του τροφοδοτικού είναι not active

Πίνακας 5.9: *Πακέτο δεδομένων κατά την ανάγνωση της κάρτας 4*

5.8.3 Μονάδα UART

Πρώτο βήμα για τη λειτουργία της μονάδας είναι να ενεργοποιήσουμε τα σήματα ρολογιού γράφοντας στον καταχωρητή Power Reduction Register 0 (PRR0) την τιμή 0 στο πεδίο PRUSART1.

Η αρχικοποίηση της μονάδας γίνεται με χρήση των οδηγών που παρέχονται. Έτσι δημιουργούμε ένα στιγμιότυπο της δομής usart_rs232_options_t και αρχικοποιούμε τα πεδία της με τις τιμές:

```
.baudrate = 115200
.charlength = USART_CHSIZE_8BIT_gc
.paritytype = USART_PMODE_DISABLED_gc
.stopbits = false
```

Έτσι ορίζουμε πως η ταχύτητα της επικοινωνίας θα είναι στα 115200 baudrate, τα δεδομένα θα είναι 8bite, δεν θα υπάρχει parity bit και τα stop bits θα είναι 1-bit.

Εκτελώντας την εντολή *usart_init_rs232* με πρώτο όρισμα τη μονάδα που θέλουμε να χρησιμοποιήσουμε, στην περίπτωσή μας USART1 (PORTD), και δεύτερο τη δομή που ορίσαμε παραπάνω, προγραμματίζουμε τη μονάδα σύμφωνα με τις ρυθμίσεις της δομής.

Επειδή θέλουμε να λαμβάνουμε και να στέλνουμε ασύγχρονα τα δεδομένα μέσω διακοπών υλικού, ενεργοποιούμε τις διακοπές γράφοντας στον καταχωρητή UCSR1B την τιμή 1 στα πεδία RXCIE1, TXCIE1.

5.8.4 Αποστολή δεδομένων παρακολούθησης

Η κάρτα 1 αφού συλλέξει τα δεδομένα από τις slave συσκευές σε πακέτα όπως φαίνονται στους πίνακες 5.5, 5.7, 5.9 και τους αντίστοιχους αισθητήρες θερμοκρασίας των καρτών, προωθεί τα δεδομένα μέσω της σειριακής στον επεξεργαστή. Μεταβιβάζονται τέσσερα διαφορετικά πακέτα, ένα για κάθε κάρτα, όπως φαίνονται παρακάτω:

Στην αρχή του κάθε δεδομένου παρακολούθησης που αποστέλλεται υπάρχει ένας delimiter (το σύμβολο ¨$¨) για να μπορεί να ξεχωρίζει εύκολα τις τιμές ο agent του Linux συστήματος.

byte 1	2	3	4	5 6	7 8	9 10	11 12	
START BIT	LENGTH	CARD_NAME	ID	PG	V_{BAT}	T_{CARD1}	CRC	STOP BIT

Πίνακας 5.10: *Πακέτο UART των δεδομένων της κάρτας 1*

byte 1	2	3	4	5 ... 15	16 17	18 19	
START BIT	LENGTH	CARD_NAME	ID	I^2C *PACKET*	T_{CARD2}	CRC	STOP BIT

Πίνακας 5.11: *Πακέτο UART των δεδομένων της κάρτας 2*

byte 1	2	3	4	5 ... 19	20 21	22 23	
START BIT	LENGTH	CARD_NAME	ID	I^2C *PACKET*	T_{CARD3}	CRC	STOP BIT

Πίνακας 5.12: *Πακέτο UART των δεδομένων της κάρτας 3*

byte 1	2	3	4	5 ... 11	12 13	14 15	
START BIT	LENGTH	CARD_NAME	ID	I^2C *PACKET*	T_{CARD4}	CRC	STOP BIT

Πίνακας 5.13: *Πακέτο UART των δεδομένων της κάρτας 4*

5.8.5 Ενημέρωση σφάλματος καρτών

Υπάρχουν επτά περιπτώσεις σφάλματος στο υλικό:

1. **Read Sensor Error**: Πρόβλημα στην επικοινωνία με τον αισθητήρα θερμοκρασίας της κάρτας.
2. **Read Card Error**: Πρόβλημα στην επικοινωνία με το μικροελεγκτή της κάρτας.
3. **Voltage Error**: Τάση κάρτας εκτός ορίων.
4. **Current Error**: Ρεύμα κάρτας εκτός ορίων.
5. **Temperature Error**: Θερμοκρασία στην κάρτα εκτός ορίων.

6. **Power Good Error**: Κάποιο σήμα Power Good της κάρτας δεν λήφθηκε σωστά.

7. **Read Ina Error**: Πρόβλημα στην επικοινωνία με το ολοκληρωμένο ina260.

Οι παραπάνω αριθμοί αλλά με αρνητικό πρόσημο αντιπροσωπεύουν το αναγνωριστικό του σφάλματος (ERROR ID). Αυτό θα γίνει πιο ξεκάθαρο στη συνέχεια.

Για να ενημερώσει για τυχόν σφάλματα η κάρτα 1 στέλνει το εξής πακέτο:

byte 1	2	3	4	5	6
START BIT	CARD_NAME	CARD ID	ERROR ID	CRC	STOP BIT

Πίνακας 5.14: *Πακέτο UART σε περίπτωση σφάλματος*

Για παράδειγμα, εάν η κάρτα 1 έχει πρόβλημα να επικοινωνήσει με τον αισθητήρα θερμοκρασίας **(Read Sensor Error)** της κάρτας 2, τότε θα στείλει το εξής μήνυμα μέσω σειριακής

START BIT	C2	2	**-1**	CRC	STOP BIT

Πίνακας 5.15: *Παράδειγμα πακέτου σφάλματος αισθητήρα της κάρτας 2*

Ένα άλλο παράδειγμα: έστω πως η κάρτα 1 έχει μετρήσει τη θερμοκρασία της κάρτας 4 και είναι εκτός των επιτρεπτών ορίων **(Temperature Error)**. Στέλνει τότε:

START BIT	C4	4	**-5**	CRC	STOP BIT

Πίνακας 5.16: *Παράδειγμα πακέτου σφάλματος θερμοκρασίας εκτός ορίων στην κάρτα 4*

Προφανώς τα σφάλματα **Read Ina Error**, **Read Current Error** αφορούν μόνο την κάρτα 3, αφού μόνο στην κάρτα αυτήν έχουν νόημα.

Τέλος, ειδικά για τα σφάλματα τάσης Voltage Error, υπάρχει ακόμη ένα πεδίο που περιγράφει σε ποια τάση παρατηρήθηκε το πρόβλημα. Για παράδειγμα, εάν στην κάρτα 3 η τάση 1.8V δεν μετρηθεί στα επιτρεπτά όρια αποστέλλεται:

START BIT	C3	3	**-3**	**1.8**	CRC	STOP BIT

Πίνακας 5.17: *Παράδειγμα πακέτου σφάλματος τάσης 1.8V στην κάρτα 3*

5.8.6 Ενημέρωση συστήματος

Σχετικά με το σύστημα εξάγονται τρία μηνύματα ως προς το χρήστη:

1. **Tamper switch released**. Ο μικροελεγκτής της κάρτας 1 ανίχνευσε μέσω του ακροδέκτη PJ7 πως το σύστημα παραβιάστηκε. Αποστέλλεται τότε προς την ΚΜΕ το μήνυμα *tamper*.

2. **Battery is dead**. Ο μικροελεγκτής της κάρτας 1 μέτρησε την τάση της μπαταρίας εκτός των αποδεκτών ορίων, δηλαδή κάτω από 2.6V. Αποστέλλεται προς την ΚΜΕ το μήνυμα *bat_off*.

3. **Battery discharged**. Ο μικροελεγκτής της κάρτας 1 μέτρησε την τάση της μπαταρίας πολύ κοντά στα όρια αποδεκτής λειτουργίας [2.6V, 2.8V]. Αποστέλλεται προς την ΚΜΕ το μήνυμα *bat_low*.

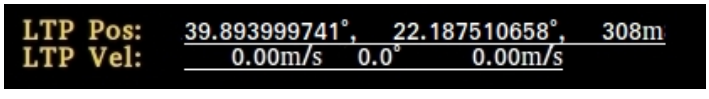
5.8.7 Ενημέρωση τοποθεσίας

Ο δέκτης μας είναι συνδεδεμένος με τις τροφοδοσίες που ενεργοποιεί ο μικροελεγκτής κι έτσι δεν χρειάζεται να στείλουμε εντολή ενεργοποίησης/απενεργοποίησης μέσω I^2C.

Ο δέκτης MAX-M8w επιτρέπει στο σχεδιαστή να υλοποιήσει κύκλωμα ανίχνευσης βραχυκυκλώματος και ανοιχτοκυκλώματος στην ενεργή κεραία λήψης σημάτων δορυφόρου, έτσι ώστε να προστατευτεί η κεραία μέσω της άμεσης διακοπής της τροφοδοσίας και της εναλλαγής τροφοδοσίας όταν αυτό είναι εφικτό. Η πληροφορία για την κατάσταση της τροφοδοσίας της κεραίας μεταβιβάζεται μέσω πακέτων NMEA ή UBX (πακέτα δεδομένων για συσκευές της u-blox).

Για να υλοποιηθεί το παραπάνω, πρέπει ο μικροελεγκτής ως master να στείλει το πακέτο "B5 62 06 13 04 00 1F 00 F0 B5 E1 DE". Τα πρώτα 2 bytes (0xB5 0x62) αποτελούν την επικεφαλίδα κάθε μηνύματος, τα επόμενα δύο (0x06, 0x13) σηματοδοτούν την εντολή UBX-CFG-ANT, έπειτα ακολουθεί το μήκος των δεδομένων (4 bytes), η χρήσιμη πληροφορία ώστε να ρυθμίσουμε τη συσκευή (0x1F 0x00 0xF0 0xB5) και τέλος το checksum εντολής για αποφυγή περιπτώσεων λανθασμένης μετάδοσης.

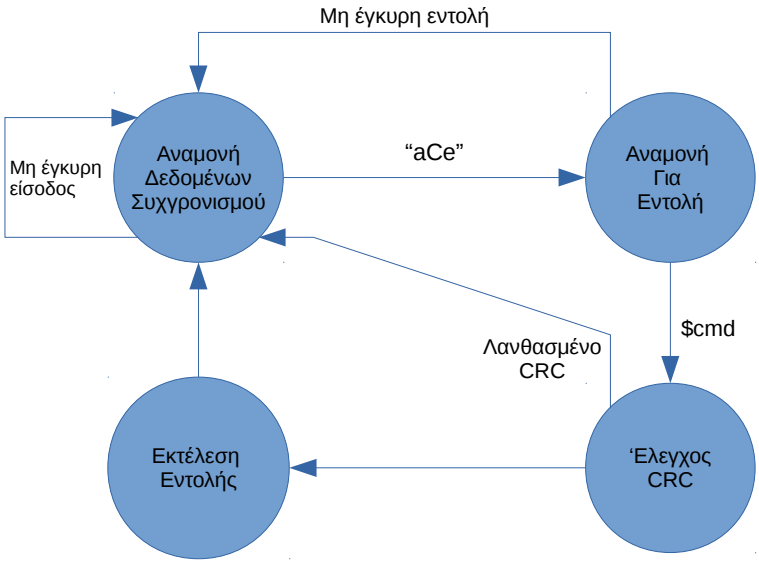
Η ενημέρωση τοποθεσίας είναι εφικτή για το χρήστη μόνο κάνοντας χρήση του Secure Shell Protocol (ssh). Έτσι, για να αποκτήσει τις πληροφορίες που το ολοκληρωμένο MAX-M8W στέλνει μέσω της σειριακής θύρας στην ΚΜΕ, ο χρήστης, αφού συνδεθεί μέσω ssh, εκτελεί την εντολή gpsmon η οποία εμφανίζει τα δεδομένα που έχει συλλέξει ο gpsd μέσω σειριακής από τη συσκευή. Αυτό προϋποθέτει την παραμετροποίηση του gpsd με τη σειριακή θύρα /dev/ttyS0 και την ταχύτητα επικοινωνίας στα 9600. Ύστερα από την εκτέλεση της εντολής gpsmon [33], εμφανίζεται ένας πίνακας με αρκετές πληροφορίες στο χρήστη. Η πληροφορία που περιέχεται στο πεδίο LTP Pos αφορά την τοποθεσία και το υψόμετρο στα οποία βρίσκεται το σύστημα. Για παράδειγμα εκτυπώνεται στο χρήστη το παρακάτω μέρος του μηνύματος:



5.8.8 Λήψη εντολών από το λειτουργικό σύστημα

Η λήψη των δεδομένων στη μονάδα γίνεται 1 byte τη φορά, μέσω εξυπηρέτησης της ρουτίνας διακοπής. Για αυτόν το λόγο, είναι αναγκαίο να οριστεί ένα FSM (Σχήμα 5.9) για να περιγράφει την κατάσταση στην οποία βρίσκεται η μονάδα λήψης δεδομένων, ώστε να γνωρίζουμε πότε περιμένουμε μία νέα εντολή, πότε μπορούμε να υπολογίσουμε το crc του

μηνύματος, και έτσι να αποδεχθούμε και να εκτελέσουμε την εντολή και να αγνοήσουμε θόρυβο που μπορεί να προκληθεί κατά τη λήψη των δεδομένων. Ορίζουμε τις παρακάτω καταστάσεις:



Σχήμα 5.9: *Μηχανή πεπερασμένων καταστάσεων για εισόδους UART*

- **Αναμονή Δεδομένων Συγχρονισμού**: Για να θεωρήσουμε πως μία εντολή έχει σταλεί από ένα χρήστη, θα πρέπει να έχει σαν επικεφαλίδα τους χαρακτήρες ASCII "aCe" ή εκφρασμένο σε δεκαεξαδικό τους αριθμούς 0x614365. Έτσι μπορούμε να απορρίπτουμε κάθε είσοδο που προκλήθηκε από θόρυβο.

- **Αναμονή για Εντολή**: Έπειτα από την αποδοχή της επικεφαλίδας συγχρονισμού, ο μικροελεγκτής σαν επόμενη είσοδο δέχεται ένα string που αντιστοιχεί στην εντολή που θα εκτελέσει. Αυτή μπορεί να είναι μία από τις παρακάτω:

  1. *Χαρτογράφηση κατάστασης συστήματος* **(initSt)**: Εντολή ενημέρωσης που απαιτείται από διεργασίες που τρέχουν στο παρασκήνιο. Μεταφέρει στο λειτουργικό ένα πακέτο δεδομένων που περιέχει πληροφορίες όπως οι κάρτες που ανιχνεύθηκαν (i^2C *probe succeeded*), τις πληροφορίες που περιέχονται στα πρώτα πακέτα δεδομένων των καρτών (Πίνακες 5.2, 5.4, 5.8) και την κατάσταση των Tamper switches.
  2. *Απενεργοποίηση συστήματος* **(AVRpwr)**: Απενεργοποιεί το σύστημα όπως περιγράφηκε στο κεφάλαιο 5.4.
  3. *Επανεκκίνηση συστήματος* **(AVRrbt)**: Προκαλεί την επανεκκίνηση του συστήματος όπως περιγράφεται στο κεφάλαιο 5.5.
  4. *Εκκίνηση Αναβάθμισης υλικολογισμικού* **(fwUpdU)**: Το στέλνει το λειτουργικό σύστημα για να ενημερώσει το μικροελεγκτή πως υλοποιείται κάποια αναβάθ-

μιση λογισμικού. Προκαλεί περιοδικό αναβοσβήσιμο του SYS LED της πρόσοψης σε πράσινο χρώμα (3.1).

5. *Τερματισμός Αναβάθμισης υλικολογισμικού* **(fwUpdD)**: Το στέλνει το λειτουργικό σύστημα για να ενημερώσει το μικροελεγκτή πως η αναβάθμιση λογισμικού υλοποιήθηκε. Ανάβει πράσινο χρώμα στο SYS LED της πρόσοψης, που υποδεικνύει πως το σύστημα είναι ανοιχτό (3.1).

- **Υπολογισμός CRC του πακέτου**: Επιπλέον έλεγχος ώστε να είμαστε σίγουροι πως η εντολή που ζητήθηκε δεν παρερμηνεύθηκε κατά τη μεταφορά.
- **Εκτέλεση εντολής**: Εκτέλεση της εντολής. Σχεδόν όλες οι εντολές κατά την περάτωσή τους δεν επιστρέφουν κάτι, εκτός από την περίπτωση της εντολής 1.

Για παράδειγμα, για να απενεργοποιήσουμε το σύστημα στέλνουμε από μέσω της σειριακής (tty) το πακέτο:

START BIT	aCe	AVRpwr	CRC	STOP BIT

5.9 Διαδικτυακή διεπαφή

Η ανάπτυξη της διαδικτυακής εφαρμογής έγινε με χρήση του Node-RED [34]. Η συγκεκριμένη πλατφόρμα αποτελεί ένα open source περιβάλλον προγραμματισμού, το οποίο εκτελείται σε φυλλομετρητές και παρέχει ευκολία στον προγραμματιστή. Η ευκολία έγκειται στο γεγονός πως ο προγραμματισμός βασίζεται σε «μαύρα κουτιά» ή κόμβους (nodes). Ο κάθε κόμβος εκτελεί μία συγκεκριμένη ενέργεια. Έτσι κάθε κόμβος, και συνεπακόλουθα κάθε ενέργεια, συντηρείται και βελτιστοποιείται ξεχωριστά. Επίσης δίνεται και η δυνατότητα σε έναν προγραμματιστή να αναπτύξει το δικό του κόμβο, γράφοντας κώδικα σε Javascript. Με αυτόν τον τρόπο ο προγραμματισμός υλοποιείται με τη διασύνδεση των επιμέρους κόμβων.

Το Node-RED είναι χτισμένο σε Node.js. Το Node.js είναι μία open source πλατφόρμα ανάπτυξης λογισμικού σε περιβάλλον Javascript. Χρησιμοποιείται κατά κόρον στην ανάπτυξη διαδικτυακών εφαρμογών, καθώς έχει ως απαίτηση τη χρήση μόνο μίας γλώσσας προγραμματισμού τόσο στον client, όσο και στον server. Η αρχιτεκτονική του node.js βασίζεται στην ασύγχρονη επικοινωνία Εισόδου/Εξόδου, βελτιστοποιώντας έτσι το throughput και την κλιμακωσιμότητα των δικτύων που δέχονται πολλές αιτήσεις. Τέλος, επειδή είναι open source, υπάρχουν πολλές βιβλιοθήκες που μπορούμε να χρησιμοποιήσουμε κατά το στάδιο της ανάπτυξης.

Οι πληροφορίες μεταβιβάζονται υπό τη μορφή json (JavaScript Object Notation). Το JSON είναι μία μορφή κειμένου που είναι ανεξάρτητη από τη γλώσσα προγραμματισμού. Είναι ευανάγνωστο τόσο από τον άνθρωπο, όσο και από τις μηχανές. Χρησιμοποιείται διότι μεταφέρει πληροφορία με πολύ μικρό κόστος σε μέγεθος.

Στο επόμενο κεφάλαιο παρουσιάζεται η διαδικτυακή εφαρμογή από την πλευρά του χρήστη.

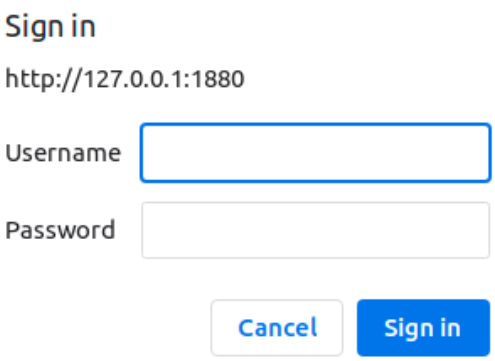
Κεφάλαιο 6

Παρουσίαση Εφαρμογής

6.1 Σύνδεση χρήστη

Για να συνδεθεί ο χρήστης στην εφαρμογή μας θα πρέπει να ανοίξει ένα φυλλομετρητή και να συνδεθεί στο σύνδεσμο *http://127.0.0.1:1880/ui*.

Στη συνέχεια, εμφανίζεται η φόρμα σύνδεσης (εικόνα 6.1) στην οποία συμπληρώνει τα στοιχεία που του έχουν δοθεί.

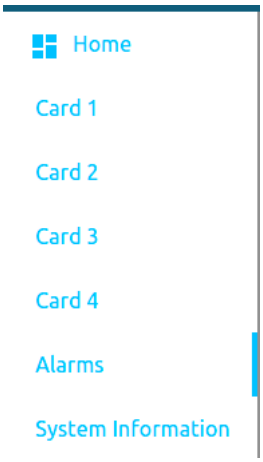


Εικόνα 6.1: *Φόρμα σύνδεσης χρήστη*

Στην αρχική οθόνη ο χρήστης μπορεί να δει την κατάσταση των LEDs της πρόσοψης του κουτιού όπως περιγράφεται στους Πίνακες 3.1, 3.2. Επιπλέον, στην αρχική οθόνη βρίσκονται και τα κουμπιά απενεργοποίησης και επανεκκίνησης του συστήματος, όπως φαίνεται στην εικόνα 6.2.

Εικόνα 6.2: *Αρχική οθόνη εφαρμογής*

Επάνω αριστερά στην αρχική οθόνη βρίσκεται το μενού. Πατώντας τις τρεις κάθετες γραμμές εμφανίζονται οι επιλογές όπως φαίνονται στην εικόνα 6.3

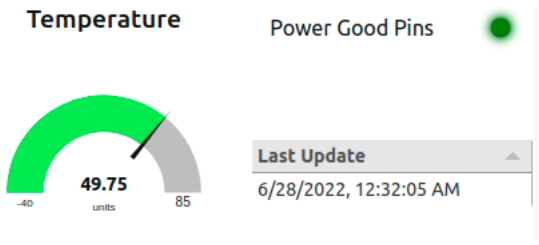


Εικόνα 6.3: *Μενού χρήστη*

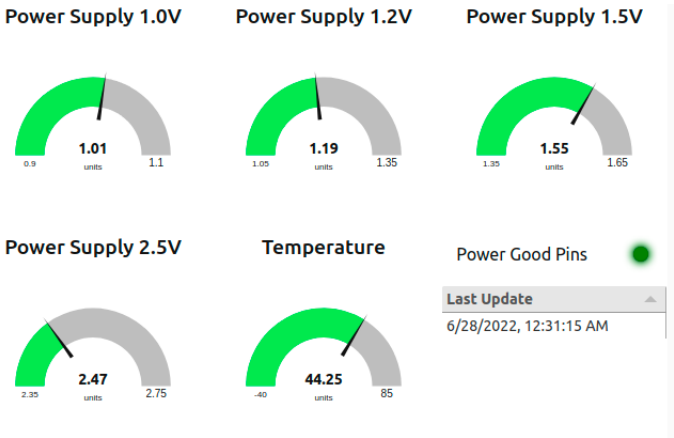
6.2 Εξαγωγή πληροφοριών στο χρήστη

Για να δει ο χρήστης τις πληροφορίες που μετρήθηκαν σε κάθε κάρτα, αρκεί να επιλέξει την αντίστοιχη κάρτα από το μενού. Στη συνέχεια εμφανίζεται μια σελίδα που ανανεώνεται κάθε πέντε δευτερόλεπτα με τις πιο πρόσφατες πληροφορίες που απέκτησε η κάρτα 1.

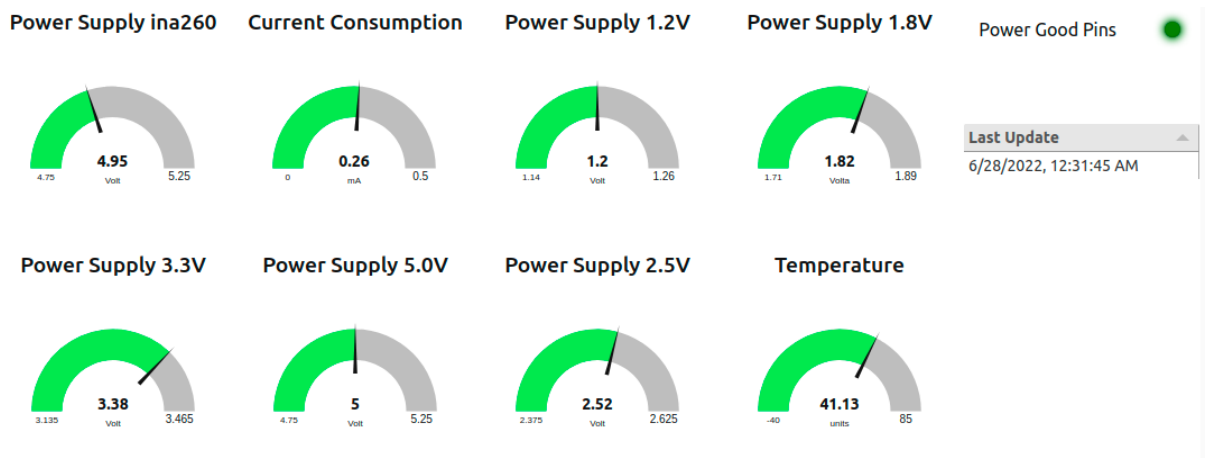
Οι πληροφορίες αυτές για τις κάρτες 1, 2, 3, 4 φαίνονται στις εικόνες 6.4, 6.5, 6.6 και 6.7 αντίστοιχα.



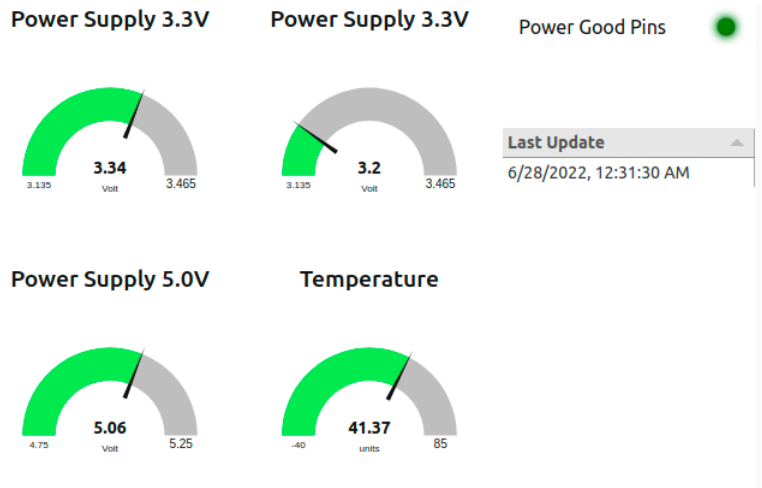
Εικόνα 6.4: *Πληροφορίες κάρτας 1*



Εικόνα 6.5: *Πληροφορίες κάρτας 2*



Εικόνα 6.6: *Πληροφορίες κάρτας 3*



Εικόνα 6.7: *Πληροφορίες κάρτας 4*

6.3 Ενδείξεις σφάλματος

Τα σφάλματα που δημιουργούνται κατά τη διάρκεια λειτουργίας του συστήματος καταγράφονται σε μία βάση δεδομένων MySQL και εμφανίζονται σε έναν πίνακα αναφοράς σφαλμάτων. Τα δεδομένα του πίνακα είναι *(i)* η χρονική στιγμή που το σφάλμα παρουσιάστηκε και *(ii)* η περιγραφή του σφάλματος. Πιθανά σφάλματα είναι τα εξής:

- Card *n* voltage *x* out of range
- Card *n* reading temperature sensor error
- Card *n* temperature out of range
- Card *n* reading card through I^2C *error*
- Card *n* current out of range
- Card *n* Power Good Pins error

Όπου n μπορεί να είναι 1, 2, 3 ή 4 και x να είναι οι τιμές 1.0, 1.2, 1.5, 1.8, 2.5, 3.3, 5.0, αναλόγως τις τροφοδοσίες που φέρει η κάθε κάρτα.

Για παράδειγμα στο σύστημα της εικόνας 6.8 η κάρτα 1 δεν μπόρεσε να επικοινωνήσει με το μικροελεγκτή της κάρτας 3 για να αποκτήσει τα δεδομένα της τοπικής παρακολούθησης, ανιχνεύθηκε πρόβλημα στην τροφοδοσία 3.3 της κάρτας 4 και η κάρτα 1 δεν μπόρεσε να αποκτήσει τα δεδομένα του αισθητήρα θερμοκρασίας της κάρτας 2 μέσω του I^2C *bus*.

Date / Time	Description
6/12/2022, 9:19:56 PM	Card 4 voltage 3.3V I out of range
6/12/2022, 9:19:36 PM	Card 2 reading temperature sensor error
6/12/2022, 9:19:06 PM	Card 3 reading I2C bus error

Εικόνα 6.8: *Πίνακας αναφοράς σφαλμάτων στο χρήστη*

6.4 Ενημέρωση παραβίασης συστήματος

Έπειτα από τη φυσική παραβίαση στο σύστημα, στον πίνακα αναφοράς σφαλμάτων εμφανίζεται ένα διακριτό μήνυμα που αναφέρει την παραβίαση αυτή (εικόνα 6.10).

Date / Time	Description
6/12/2022, 10:12:42 PM	Tamper switch released

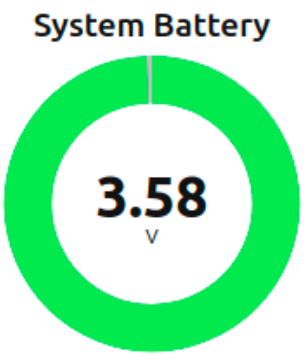
Εικόνα 6.9: *Μήνυμα παραβίασης συστήματος*

6.5 Ενημέρωση κατάστασης συστήματος

Στην τελευταία επιλογή του μενού (System information) εμφανίζεται στο χρήστη η πληροφορία της πρώτης ανάγνωσης των slave μικροελεγκτών. Αυτές οι πληροφορίες είναι τα IDs και η εκδόσεις των υλικολογισμικών των καρτών. Επίσης εμφανίζεται live η κατάσταση του tamper switch. Τέλος, εμφανίζεται η τάση της μπαταρίας που μετριέται από την κάρτα 1.

Card ID	Firmware Version
1	4.2
2	1.4
3	1.1
4	3.8

Tamper switch deactivated. System is safe



Εικόνα 6.10: *Πληροφορίες συστήματος στο χρήστη*

Κεφάλαιο 7

Δοκιμές, Συμπεράσματα & Μελλοντικές επεκτάσεις

Η εξαγωγή των συμπερασμάτων υλοποιήθηκε σε διαφορετικά περιβάλλοντα δοκιμών.

7.1 Περιβάλλον εργαστηρίου

Έχοντας εκτελέσει τον πλήρη έλεγχο της ορθής λειτουργίας του συστήματος μπορεί να συναχθεί το συμπέρασμα πως οι επικοινωνίες μεταξύ των μικροελεγκτών και των αισθητήρων, των μικροελεγκτών, του μικροελεγκτή και της ΚΜΕ και του εξυπηρετητή με το χρήστη λειτουργούν απρόσκοπτα και κατά τον αναμενόμενο τρόπο. Συγκεκριμένα, οι εντολές αποστέλλονται ορθά από την πλευρά του χρήστη προς το μικροελεγκτή της κάρτας 1, με αποτέλεσμα αυτός να τις πραγματοποιεί άμεσα. Αντίστοιχα, οι μετρήσεις και τα σφάλματα υλικού συλλέγονται και αποστέλλονται από το μικροελεγκτή προς το χρήστη σε καθορισμένο χρόνο, απεικονίζοντας ορθά τα δεδομένα των μετρήσεων στο γραφικό τους περιβάλλον. Η αλληλεπίδραση του χρήστη με το σύστημα έχει την επιθυμητή συμπεριφορά κατά την παραβίαση , ενεργοποίηση και απενεργοποίησή του.

Στο περιβάλλον εργαστηρίου η κατανάλωση ενέργειας στις δύο καταστάσεις του συστήματος idle και active ήταν η εξής:

Κατάσταση συστήματος	Τάση εισόδου (V)	Ρεύμα εισόδου (A)	Ισχύς (W)
idle	24	0.08	1.44
active	24	1.59	38.16

Πίνακας 7.1: *Κατανάλωση ενέργειας των μικροελεγκτών σε διαφορετικές καταστάσεις λειτουργίας*

Η αυξημένη κατανάλωση (0.08A) προκύπτει από το γεγονός ότι αρκετά ολοκληρωμένα είναι συνδεδεμένα με την τάση stand by, οπότε τροφοδοτούνται μαζί με τους μικροελεγκτές. Το παραπάνω γίνεται αντιληπτό ελέγχοντας επίσης και τα εγχειρίδια. Από αυτά προκύπτει πως αθροιστικά η κατανάλωση σε idle mode για τους μικροελεγκτές είναι 11.4mA και για active mode είναι 36mA. Η κατανάλωση του κάθε μικροελεγκτή σύμφωνα με τα εγχειρίδια είναι:

1. Για τον ATxmega128D4 στα 32MHz με 3.3Volt τροφοδοσία:

- **Idle**: 4.2mA
- **Active**: 10.5mA

2. Για τον ATmega2560 στα 7.37MHz με 3.3Volt τροφοδοσία:

   - **Idle**: 1.5mA
   - **Active**: 7.5mA

Για την αξιολόγηση του κώδικα χρησιμοποιούμε τη μετρική του μεγέθους του κώδικα, καθώς επίσης και της κατανάλωσης της μνήμης. Παρακάτω παρουσιάζονται τα στοιχεία που μετρήθηκαν:

Κάρτα	Μέγεθος εκτελέσιμου	Κατανάλωση Flash	Μέγεθος δεδομένων	Κατανάλωση SRAM
1	57496 Bytes	21.9%	1831 Bytes	22.4%
2	20702 Bytes	7.8%	262 Bytes	3.2%
3	18506 Bytes	13.3%	295 Bytes	3.5%
4	18408 Bytes	13.2%	321 Bytes	3.9%

Πίνακας 7.2: *Χρησιμοποίηση μνημών μικροελεγκτών*

7.2 Θερμική δοκιμή

Το πρώτο περιβάλλον δοκιμής υλοποιήθηκε μέσα σε ένα φούρνο που είχε ρυθμιστεί στους +65°C, δηλαδή στο ανώτατο όριο της επιτρεπτής θερμοκρασίας περιβάλλοντος για το σύστημα. Σκοπός του θερμικού τεστ είναι η δοκιμή της λειτουργικότητας του συστήματος και η επαλήθευση τήρησης των προδιαγραφών κάθε συσκευής. Οι συσκευές που δεν τηρούν τις θερμικές προδιαγραφές έχουν χαμηλή αξιοπιστία και μπορούν να επηρεάσουν το συνολικό σύστημα, ωθώντας το στη μη τήρηση των προδιαγραφών του.

Κατά τη θερμική δοκιμή παρατηρήθηκαν δύο σφάλματα υλικού:

1. Όταν η θερμοκρασία της κάρτας 3 ξεπερνούσε τους 71°C το τροφοδοτικό των 2.5V εξήγαγε σφάλμα μέσω του Power Good σήματος, ενώ η τάση που μετρούσε ο μικροελεγκτής ήταν σταθερή στα 2.51V. Επίσης το κύκλωμα που ήταν συνδεδεμένο με το τροφοδοτικό ήταν ενεργό, που σημαίνει πως μόνο το Power Good σήμα ήταν λανθασμένο. Το ίδιο παρατηρήθηκε και όταν η θερμοκρασία ξεπερνούσε τους 72°C με το τροφοδοτικό των 1.8V. Η αλλαγή των τροφοδοτικών έλυσε το πρόβλημα αυτό και το σύστημα είχε σταθερή συμπεριφορά στις θερμοκρασίες αυτές.

2. Κατά τη δοκιμή ενεργοποίησης της κάρτας 4 σε θερμοκρασία μεγαλύτερη των 68°C παρατηρήθηκε πρόβλημα λόγω του reverse saturation current. Στο κύκλωμα μας υπήρχε ένα LED σε κάθε τάση για να παρέχεται η ένδειξη σωστής τροφοδότησης. Σε περιβάλλον υψηλής θερμοκρασίας υπήρχε αυξημένη ροή ρεύματος από τη δίοδο του LED, το οποίο ενίσχυε την ένταση του αντίστροφου ρεύματος της διόδου που μέσω ενός συγκριτή, καθόριζε το reset σήμα των τροφοδοτικών. Έτσι κατά την εκκίνηση, τα

τροφοδοτικά λειτουργούσαν στιγμιαία και το reset σήμα δεν επέτρεπε τη λειτουργία τους. Αφαιρώντας τα LEDs το κύκλωμα λειτούργησε όπως αναμενόταν μέχρι τους 76°C.

Έπειτα από τη διόρθωση των παραπάνω παρατηρήσεων, το σύστημα δοκιμάστηκε ξανά έχοντας παρόμοια συμπεριφορά με αυτή στο περιβάλλον εργαστηρίου. Αυτό οφείλεται στο γεγονός πως η μέγιστη θερμοκρασία που αναπτύχθηκε στις κάρτες δεν ξεπέρασε τους 76°C. Η λειτουργική συμπεριφορά του συστήματος δεν επηρεάστηκε και η κατανάλωση του συστήματος δεν άλλαξε αισθητά, κάτι που συμπίπτει με τις πληροφορίες των εγχειριδίων αν λάβει κανείς υπόψιν τη διαφορά της κατανάλωσης σε 25°C και 85°C σε ενεργή κατάσταση. Αυτή η πληροφορία παρουσιάζεται στον παρακάτω πίνακα:

Ολοκληρωμένο	**Κατανάλωση ρεύματος στους 25°C**	**Κατανάλωση ρεύματος στους 85°C**
ATxmega128D4	10.5mA	10.2mA
ATmega2560	12.1mA	12.7mA
mcp0808	160μA	179μA
ina260	16.9μA	16.9μA
MAX-M8W	38mA	38mA
cdcm6208	318mA	318mA

Πίνακας 7.3: *Διαφορά της κατανάλωσης ενέργειας σε διαφορετικές θερμοκρασίες λειτουργίας*

7.3 Δοκιμή δονήσεων

Το δεύτερο περιβάλλον δοκιμής υλοποιήθηκε πάνω σε μία πλατφόρμα που δονείται με τυχαίο τρόπο. Σκοπός της δοκιμής αυτής είναι η προσομοίωση κανονικών συνθηκών λειτουργίας, έτσι ώστε να επαληθευτεί η ακεραιότητα και η σταθερή λειτουργία του συστήματος.

Στη δοκιμή αυτή συνήθως εμφανίζονται προβλήματα στα μηχανικά κομμάτια του συστήματος, όπως για παράδειγμα στους μηχανικούς (HDD) σκληρούς δίσκους που έχουν κινούμενα μέρη, τα οποία κατά τη λειτουργία υπό δονήσεις μπορούν να φθαρούν, καθιστώντας το σύστημα εκτός λειτουργίας.

Ένα άλλο παράδειγμα φθοράς που εντοπίζεται συνήθως είναι η κύρτωση του PCB της κάρτας, η οποία μπορεί να λυγίσει και να φθείρει όλο το σύστημα.

Στη δοκιμή δονήσεων δεν παρουσιάστηκε κανένα πρόβλημα και η λειτουργία του συστήματος ήταν ίδια με τη λειτουργία στο περιβάλλον του εργαστηρίου, επικυρώνοντας έτσι τον ορθό πολυεπίπεδο αρχικό σχεδιασμό του συστήματος.

7.4 Βελτιώσεις

Η λήψη μετρήσεων για την παρακολούθηση σε επίπεδο υλικού αποτελεί ένα αναγκαίο μέσο προστασίας ενός αξιόπιστου συστήματος τόσο από ενδεχόμενες αυξομειώσεις τάσεων, όσο και από την υψηλή θερμοκρασία, που μπορούν να προκαλέσουν μη αναστρέψιμη ζημιά στο σύστημα. Επίσης, η ύπαρξη ενός μέσου διαχείρισης ενός συστήματος από ένα χρήστη αποτελεί πρωταρχικό πρόβλημα που χρειάζεται λύση κατά το σχεδιασμό οποιουδήποτε συ-

στήματος. Συνεπώς, η μελέτη λύσεων υλοποίησης των παραπάνω αποτελεί ζωτικής σημασίας τόσο για τον κλάδο της τεχνολογίας, όσο και για όλους τους τομείς ανάπτυξης προϊόντων.

Η παραπάνω εργασία αποτελεί μία ολοκληρωμένη προσπάθεια ανάπτυξης ενός συστήματος παρακολούθησης που έχει ως στόχο την εύκολη εποπτεία και αποσφαλμάτωση του ίδιου του συστήματος. Παρόλα αυτά, στο πλαίσιο του εγχειρήματος αυτού ήταν απαραίτητο να γίνουν κάποιες απλοποιήσεις και εκ των προτέρων θεωρήσεις, με αποτέλεσμα το αναπτυχθέν σύστημα να επιδέχεται βελτιώσεων.

Η βασικότερη βελτίωση που θα μπορούσε να εφαρμοστεί, είναι να παρέχεται η πληροφορία των ALERT σημάτων των αισθητήρων των καρτών 2-4 στην κάρτα 1, καθώς έτσι θα μπορέσει να απενεργοποιήσει τις τάσεις του συστήματος πολύ γρηγορότερα και όχι κατά τον κύκλο παρακολούθησης που λαμβάνει χώρα κάθε 5 δευτερόλεπτα.

Από τη στιγμή που το σύστημα αποσταλεί στον πελάτη, το σενάριο αναβάθμισης των υλικολογισμικών στις κάρτες είναι δύσκολα υλοποιήσιμο σύμφωνα με την τωρινή σχεδίαση. Αυτό έγκειται στο γεγονός πως δεν υπάρχει πρόσβαση προς τον έξω κόσμο, παρά μόνο μέσω της σειριακής επικοινωνίας μεταξύ της ARM CPU και του μικροελεγκτή της κάρτας 1. Συνεπώς, κατά τη συγγραφή ενός bootloader θα πρέπει να ενσωματώσουμε όλη τη λογική της εφαρμογής στο bootloader section της Flasb μνήμης, πράγμα δύσκολο δεδομένου του μεγέθους της μνήμης, δηλαδή 8KByte. Επιπλέον, για την αναβάθμιση των υπόλοιπων καρτών θα πρέπει η κάρτα 1 να παραλάβει τα δεδομένα αναβάθμισης μέσω UART και να τα προωθήσει μέσω I^2C στις υπόλοιπες κάρτες, κάτι που αυξάνει το περιθώριο λάθους, το χρόνο υλοποίησης, καθώς επίσης και την πολυπλοκότητα της λύσης. Προτείνεται λοιπόν η εισαγωγή μιας σειριακής θύρας UART στην πρόσοψη του συστήματος της κάθε κάρτας, δίνοντας τη δυνατότητα της επιλογής της κάρτας προς αναβάθμιση μέσω ενός πολυπλέκτη κατευθυνόμενο από την κάρτα 1.

Το σύστημα αυτό αποτελεί την καρδιά ενός οποιουδήποτε συστήματος τη σήμερον ημέρα. Η βασική λειτουργία της εργασίας θα μπορούσε να υλοποιηθεί με διαφορετικούς τρόπους και σε διαφορετικά σενάρια λειτουργίας. Για παράδειγμα, εάν η πληροφορία των μετρήσεων αντί να μεταφέρεται ενσύρματα μέσω του I^2C μεταφερόταν μέσω ραδιοσυχνοτήτων, θα είχαμε έναν εξοπλισμό παρακολούθησης που θα μπορούσε να εφαρμοστεί στη γεωργία. Εάν η διασύνδεση μεταξύ των ολοκληρωμένων γινόταν με CAN bus (λόγω της μεγαλύτερης απόστασης που επιτρέπει), θα μπορούσε το σύστημα να αποτελεί ένα υποσύστημα παρακολούθησης μερών αυτοκινήτου. Στα παραπάνω παραδείγματα γίνεται αντιληπτό πως η μελέτη λειτουργίας και οι αρχές που διέπουν ένα τέτοιο σύστημα αποτελούν βάση πολλών σύγχρονων συστημάτων.

Βιβλιογραφία

[1] *30 Linux System Monitoring Tools Every SysAdmin Should Know*. https://www.cyberciti.biz/tips/top-linux-monitoring-tools.html. Ημερομηνία πρόσβασης: 10-06-2022.

[2] *Human-computer interaction*. https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction. Ημερομηνία πρόσβασης: 10-06-2022.

[3] S. Kiruthika K. Meena S. Hari Bala Krishnan M. V. Logeswaran M. Thilagaraj, R. Krishnakumar. *Iot Based Embedded System for Continuous Healthcare Monitoring*. *Annals of RSCB*, σελίδα 4420–4424, 2021.

[4] Jinyou Zhang. *Network communication technology of automobile inspection and control system based on embedded system*. *2021 IEEE 4th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, σελίδες 428–432, 2021.

[5] Yanni Zhai και Xiaodong Cheng. *Design of smart home remote monitoring system based on embedded system*. *2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering*, τόμος 2, σελίδες 41–44, 2011.

[6] Shailendra Singh και Priya Ranjan. *Towards a new low cost, simple implementation using embedded system wireless networking for UAVs*. *2011 Fifth IEEE International Conference on Advanced Telecommunication Systems and Networks (ANTS)*, σελίδες 1–3, 2011.

[7] Yanghao Ye, Qing Wang και Jian Wang. *Green city air monitoring and architectural digital art design based on IoT embedded system*. *Environmental Technology Innovation*, 23:101717, 2021.

[8] *Serial Peripheral Interface*. https://en.wikipedia.org/wiki/Serial_Peripheral_Interface. Ημερομηνία πρόσβασης: 19-12-2021.

[9] *I^2C-bus specification and user manual Rev 7.0*. https://www.nxp.com/docs/en/user-guide/UM10204.pdf. Ημερομηνία πρόσβασης: 10-01-2021.

[10] *I^2Primer: What is I2C? (Part 1)*. https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html. Ημερομηνία πρόσβασης: 10-06-2022.

[11] *MCP9808 0.5C Maximum Accuracy Digital Temperature Sensor*. https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/MCP9808-0.5C-Maximum-Accuracy-Digital-Temperature-Sensor-Data-Sheet-DS20005095B.pdf. Ημερομηνία πρόσβασης: 10-11-2021.

[12] *INA260 Precision Digital Current and Power Monitor With Low-Drift, Precision Integrated Shunt datasheet (Rev. C).* https://www.ti.com/lit/ds/symlink/ina260.pdf?ts=1656151116124&ref_url=https%253A%252F%252Fwww.google.com%252F. Ημερομηνία πρόσβασης: 10-12-2021.

[13] *CDCM6208 2:8 Clock Generator, Jitter Cleaner With Fractional Dividers datasheet (Rev. G).* https://www.ti.com/lit/ds/symlink/cdcm6208.pdf?ts=1656175907067&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCDCM6208. Ημερομηνία πρόσβασης: 14-12-2021.

[14] *Atmel AVR XMEGA D Manual.* https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/Atmel-8210-8-and-16-bit-AVR-Microcontrollers-XMEGA-D_Manual.pdf. Ημερομηνία πρόσβασης: 01-10-2021.

[15] *ATxmega128D4/64D4/32D4/16D4 - Complete Datasheet.* https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8135-8-and-16-bit-AVR-microcontroller-ATxmega16D4-32D4-64D4-128D4_datasheet.pdf. Ημερομηνία πρόσβασης: 01-10-2021.

[16] *ATmega640/1280/1281/2560/2561 - Complete Datasheet.* https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/ATmega640-1280-1281-2560-2561-Datasheet-DS40002211A.pdf. Ημερομηνία πρόσβασης: 01-10-2021.

[17] *What is Frequency Stability?* https://www.everythingrf.com/community/what-is-frequency-stability. Ημερομηνία πρόσβασης: 14-12-2021.

[18] *MAX-8 / MAX-M8 Hardware Integration Manual.* https://www.u-blox.com/docs/UBX-15030059. Ημερομηνία πρόσβασης: 02-05-2022.

[19] *u-blox 8 / u-blox M8 Receiver Description including Protocol Specification (public version).* https://www.u-blox.com/docs/UBX-13003221. Ημερομηνία πρόσβασης: 02-05-2022.

[20] *Microchip Studio for AVR® and SAM Devices.* https://www.microchip.com/en-us/tools-resources/develop/microchip-studio. Ημερομηνία πρόσβασης: 01-10-2021.

[21] RITCHIE M. DENNIS KERNIGHAN W. BRIAN. *Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C.* ΚΛΕΙΔΑΡΙΘΜΟΣ ΕΠΕ ΕΚΔΟΣΕΙΣ, 2η έκδοση, 2011.

[22] *Preventing switch bounce in low-profile keypad assembles.* https://blog.epectec.com/preventing-switch-bounce-in-low-profile-keypad-assemblies. Ημερομηνία πρόσβασης: 10-05-2022.

[23] *Buildroot - Making Embedded Linux Easy.* https://buildroot.org/. Ημερομηνία πρόσβασης: 10-01-2022.

[24] *The Linux Kernel Archives.* https://www.kernel.org/. Ημερομηνία πρόσβασης: 21-02-2022.

[25] Sreekrishnan Venkateswaran. *Essential Linux Device Drivers*. Prentice Hall Open Source Software Development, 1η έκδοση, 2008.

[26] *Binutils - GNU Project - Free Software Foundation*. https://www.gnu.org/software/binutils/. Ημερομηνία πρόσβασης: 01-06-2022.

[27] *BusyBox-Commands*. https://boxmatrix.info/wiki/BusyBox-Commands. Ημερομηνία πρόσβασης: 21-05-2022.

[28] *Node.js*. https://nodejs.org/en/. Ημερομηνία πρόσβασης: 01-06-2022.

[29] *Secure Shell*. https://en.wikipedia.org/wiki/Secure_Shell. Ημερομηνία πρόσβασης: 21-05-2022.

[30] *gpsd — a GPS service daemon*. https://gpsd.gitlab.io/gpsd/. Ημερομηνία πρόσβασης: 21-05-2022.

[31] *DCRON - DILLON'S LIGHTWEIGHT CRON DAEMON*. https://github.com/dubiousjim/dcron. Ημερομηνία πρόσβασης: 01-6-2022.

[32] *Duck DNS free dynamic DNS hosted on AWS*. http://www.duckdns.org/. Ημερομηνία πρόσβασης: 01-05-2022.

[33] *gpsmon(1)*. https://gpsd.gitlab.io/gpsd/gpsmon.html. Ημερομηνία πρόσβασης: 01-06-2022.

[34] *Node-RED*. https://nodered.org/. Ημερομηνία πρόσβασης: 01-06-2022.