DEPT. OF AUTOMATION ENGINEERING T.E.
ATEI OF THESSALONIKI

Κατασκευή συστήματος αισθητήρα φωτοδιόδων με μικροϋπολογιστή Beaglebone για χρήση σε μη-παρεμβατικούς βιοϊατρικούς αισθητήρες

Ευστράτιος Γκαγκάνης

Επιβλέπον καθηγητής: Δρ. Μιχαήλ Κιζήρογλου

Θεσσαλονίκη 2019

Abstract

Applications using Photodiode Arrays demand accurate timing to control the duration of exposure and provide accurate frame time-stamping. In this thesis, a real-time programming method used for a new medical laser scattering sensor system is described along with the necessary configurations of the Operating System running on the system's board. In medical devices, the margin of error must be small, so determinism is ensured and the code is optimized by exploitation of all available subsystems of the System on Chip, thereby speeding up various processes. A testing module has been fabricated to debug the driving system and ensure proper operation. Neither the algorithms nor the hardware configuration presented are bound to this particular device and can be used for any application using a sensor with the same operating principle.

Περίληψη

Οι εφαρμογές που χρησιμοποιούν Πίνακες Φωτοδιόδων απαιτούν ακριβή χρονισμό για τον έλεγχο της διάρκειας της έκθεσής τους στο φως, καθώς και ακρίβεια στην χρονική καταγραφή των καρέ. Σε αυτή την πτυχιακή εργασία, παρουσιάζεται μια προγραμματιστική μέθοδος πραγματικού χρόνου που χρησιμοποιείται σε μία νέα ιατρική συσκευή με στόχο την μέτρηση της σκέδασης ακτίνας λέιζερ. Επιπλέον, αναλύονται οι απαραίτητες παραμετροποιήσεις στο Λειτουργικό Σύστημα που τρέχει η πλακέτα του συστήματος. Στις ιατρικές συσκευές το περιθώριο σφάλματος είναι μικρό και για τον λόγο αυτό διασφαλίζεται ο ντετερμινισμός του συστήματος ενώ ο κώδικας είναι βελτιστοποιημένος χρησιμοποιώντας όλα τα διαθέσιμα υποσυστήματα του ολοκληρωμένου. Μία πρωτότυπη μονάδα έχει κατασκευαστεί με σκοπό την αποσφαλμάτωση του συστήματος και την εξακρίβωση της ομαλής του λειτουργίας. Τόσο οι αλγόριθμοι όσο και η διάταξη του υλικού που παρουσιάζονται στην εργασία, μπορούν να χρησιμοποιηθούν σε κάθε εφαρμογή που βασίζεται σε αισθητήρα με την ίδια αρχή λειτουργίας.

Πίνακας περιεχομένων

Επεξήγηση ακρωνύμων

ADC	Analog to Digital Converter
API	Application Programming Interface
V_{AREF}	Analog Reference Voltage
ASIC	Application Specific Integrated Circuit
BBB	BeagleBone Black
CLK	Clock
DDR	Double Data Rate
DTO	Device Tree Overlay
FTP	File Transfer Protocol
GPIO	General Purpose Input / Output
GUI	Graphical User Interface
IoT	Internet of Things
LOC	Lines Of Code
INTC	Interrupt Controller
OS	Operating System
PDA	Photodiode Array
PRU	Programmable Real-Time Unit
PRU-ICSS	Programmable Real-Time Unit and Industrial Communication SubSystem
PRUSS	Programmable Real-Time Sub-System
RAM	Random Access Memory
RPROC	Remote Processor
ROM	Read Only Memory
RTOS	Real Time Operating System
SBC	Single-board Computer
SI	Serial-Input
SoC	System on Chip
SSH	Secure Shell
TI	Texas Instruments
UIO	Userspace I/O

1 Εισαγωγή

Οι ιατρικές συσκευές, σε σχέση με τις περισσότερες άλλες συσκευές που κυκλοφορούν στο εμπόριο, είναι απαραίτητο να προσφέρουν υψηλή ακρίβεια μετρήσεων καθώς και επαναληψιμότητα. Τα χαρακτηριστικά αυτά, είναι ακόμη σημαντικότερο να υπάρχουν στις συσκευές που βρίσκονται ακόμη σε στάδιο ανάπτυξης καθώς οι ερευνητικές ομάδες βασίζονται στα δεδομένα που τους παρέχουν αυτές. Το έργο που παρουσιάζεται στη συγκεκριμένη πτυχιακή εργασία, αφορά την ανάπτυξη των θεμελιωδών λειτουργειών μιας ιατρικής συσκευής, με σκοπό την μέτρηση της σκέδασης ακτίνας λέιζερ στα κύτταρα του αίματος [1]. Η μέτρηση βασίζεται σε οπτικές και μη παρεμβατικές μεθόδους, χρησιμοποιώντας έναν οπτικό αισθητήρα. Κατά τα επόμενα κεφάλαια παρουσιάζεται ολόκληρη η διαδικασία της οδήγησης του πίνακα φωτοδιόδων (photodiode array, PDA) καθώς και οι απαραίτητες τροποποιήσεις του λειτουργικού συστήματος (Operational System, OS) που τρέχει στον μικροϋπολογιστή BeagleBone Black (BBB) [2]. Ένα κεφάλαιο έχει αφιερωθεί στην παρουσίαση της διαδικασίας δοκιμής και αξιολόγησης του συστήματος.

1.1 Οδηγός

Η διασύνδεση ενός αισθητήρα με έναν μικροϋπολογιστή ή μικροεγλεκτή είναι μόνο η αρχή. Για να λειτουργήσει ο αισθητήρας και να επιστρέψει τιμές από τον πραγματικό κόσμο απαιτείται η συγγραφή του «οδηγού». Πρόκειται για πρόγραμμα που έχει ως στόχο την επικοινωνία της CPU με τα περιφερειακά. Για την συγγραφή του οδηγού είναι απαραίτητη η μελέτη του φύλλου δεδομένων του περιφερειακού, καθώς και η γνώση της αρχιτεκτονικής του υπολογιστή. Τα τελευταία χρόνια, η ανάπτυξη οδηγών μεταφέρεται προς γλώσσες υψηλότερου επιπέδου καθώς οι κατασκευαστές υλικού παρέχουν στους προγραμματιστές λογισμικό με πολλαπλά abstraction layers. Ο οδηγός που παρουσιάζεται στη συγκεκριμένη εργασία, είναι γραμμένος κατά βάση σε Assembly. Πρόκειται για γλώσσα χαμηλού επιπέδου και ο λόγος που επιλέχθηκε θα εξηγηθεί στο αντίστοιχο κεφάλαιο της εργασίας.

1.2 Λειτουργία πραγματικού χρόνου

Οι απαιτήσεις των ιατρικών συσκευών σε ακρίβεια, αφορούν τόσο το υλικό όσο και το λογισμικό τους. Οι ανάγκες σε υλικό υψηλής ακρίβειας μπορούν να καλυφθούν χρησιμοποιώντας ηλεκτρονικά εξαρτήματα με χαμηλή ανοχή που προορίζονται για αυτού του είδους τις συσκευές. Οι ανάγκες του λογισμικού συνήθως αφορούν την επεξεργαστική ισχύ, την μνήμη και τους χρονισμούς των διεργασιών. Ενώ η μνήμη και η επεξεργαστική ισχύς καλύπτονται από το υλικό και είναι εύκολο να εξυπηρετηθούν, ο χρονισμός των διεργασιών είναι καθαρά θέμα αρχιτεκτονικής κώδικα και συστήματος. Τα OS γενικής χρήσης που χρησιμοποιούνται κατά κόρον στους υπολογιστές, καθώς και οι διάφορες βιβλιοθήκες χρόνου εκτέλεσης που τρέχουν στο παρασκήνιο, καθιστούν την λειτουργία σε πραγματικό χρόνο αδύνατη. Χάνεται λοιπόν η δυνατότητα για χρονοδρομολόγηση (task scheduling) υψηλής ακρίβειας, των χρονικά κρίσιμων λειτουργιών [3].

Ενώ υπάρχουν OS πραγματικού χρόνου (real time operating systems, RTOS), με αυστηρούς χρονοπρογραμματιστές (schedulers) που ελέγχουν την ροή του προγράμματος και την προτεραιότητα των νημάτων σύμφωνα με τις ρυθμίσεις του προγραμματιστή, δεν επιλέχθηκε κάποιο για την συγκεκριμένη πτυχιακή εργασία, καθώς το BBB περιέχει δύο προγραμματιζόμενες μονάδες πραγματικού χρόνου (Programmable Real-Time Units, PRUs). Τα PRU είναι στην πραγματικότητα ανεξάρτητοι από την CPU, μικροελεγκτές που δεν τρέχουν τίποτα στο παρασκήνιο και προσφέρουν λειτουργία σε πραγματικό χρόνο. Επιπλέον, με την χρήση της γλώσσας Assembly, μπορεί να επιτευχθεί υπολογισμός της χρονικής διάρκειας των ρουτινών με ακρίβεια ενός κύκλου μηχανής, πράγμα που δεν μπορεί να γίνει με την χρήση κάποιου RTOS, ακόμη και αν χρησιμοποιηθεί “inline assembly” [4].

Ο οδηγός του οπτικού αισθητήρα που αναπτύχθηκε στα πλαίσια της εργασίας, τρέχει σε πραγματικό χρόνο χωρίς να μοιράζεται τους PRU με άλλα προγράμματα που τρέχουν στο παρασκήνιο. Το χαρακτηριστικό αυτό, καθιστά τον οδηγό 100% ντετερμινιστικό.

2 Σχεδιαστικές προδιαγραφές

Σε αυτό το κεφάλαιο αναλύονται οι στόχοι της πτυχιακής εργασίας και οι πορεία που ακολουθήθηκε για την διεκπεραίωσή της.

2.1 Προδιαγραφές και στόχοι

Στόχος της παρούσας πτυχιακής εργασίας είναι η κατασκευή ενός συστήματος αισθητήρα φωτοδιόδων, που θα μπορεί να εξάγει και να αποθηκεύει δείγματα με μη-παρεμβατικές μεθόδους από τα ακροδάχτυλα ασθενών. Βασικός ρόλος του συγγραφέα ήταν η δημιουργία ενός αξιόπιστου οδηγού με σκοπό να χρησιμοποιηθούν στη συνέχεια από ερευνητές.

Στα πρώτα στάδια της εργασίας, κάποιες από τις λειτουργικές παραμέτρους της συσκευής, όσον αφορά τον αισθητήρα φωτοδιόδων, δεν είχαν προσδιοριστεί. Αυτό σήμαινε ότι συγκεκριμένα χαρακτηριστικά του οδηγού, όπως μεταξύ άλλων ήταν ο χρόνος έκθεσης του αισθητηρίου στο φως, θα έπρεπε να είναι πλήρως παραμετροποιήσιμα από τους ερευνητές που θα τους χρησιμοποιούσαν. Το σύστημα έπρεπε να είναι σε θέση να εξυπηρετεί χρονισμούς της τάξης των δεκάδων microseconds (μs) μέχρι μερικών εκατοντάδων milliseconds (ms), με σφάλμα μικρότερο του 1%.

Υποθέτοντας ίδιες συνθήκες φωτισμού στο PDA καθώς και είσοδο από τον χρήστη στο σύστημα, ο οδηγός θα έπρεπε πάντα να επιστρέφει τα ίδια δεδομένα για το δείγμα. Εν ολίγοις, η διαδικασία οδήγησης του PDA έπρεπε να είναι ντετερμινιστική και η επανάληψη του ίδιου πειράματος να επιστρέφει τα ίδια δεδομένα. Ενώ στις περισσότερες εφαρμογές δεν δημιουργεί πρόβλημα η χρονική καθυστέρηση που εισάγει ένα OS στην εκτέλεση του κώδικα, στη συγκεκριμένη περίπτωση όπου υπάρχει απαίτηση για ακρίβεια στην τάξη των μs, η ύπαρξη ενός OS και ο διαμοιρασμός του χρόνου της CPU σε διάφορα νήματα θα σήμαινε ανεπιθύμητη καθυστέρηση με αποτέλεσμα την αλλοίωση των δεδομένων.

2.2 Ανάπτυξη συστήματος

Η ανάπτυξη ξεκίνησε με την μελέτη των πόρων του BBB και τον τρόπο που τους χειρίζεται το OS. Σύντομα έγινε ξεκάθαρο πως για την επίτευξη των στόχων της εργασίας, έπρεπε να γίνουν αλλαγές σε αρχεία του συστήματος Linux που χρησιμοποιεί το BBB, λόγω της προκαθορισμένης κατάστασης των συγκεκριμένων ακροδεκτών που ήταν απαραίτητοι για τη λειτουργία του συστήματος. Μέσω δοκιμών, ο συγγραφέας κατέληξε σε ένα πλήρως λειτουργικό Device Tree Overlay (DTO), το οποίο περιγράφει κάθε υποσύστημα του BBB που θα χρησιμοποιούταν από τον οδηγό στο χαμηλό επίπεδο, συμπεριλαμβανομένων των ακροδεκτών εισόδου / εξόδου γενικής χρήσης.

Ο αισθητήρας που χρησιμοποιείται στην πτυχιακή εργασία είναι ο AMS TSL1401CL, αποτελούμενος από 128 φωτοδιόδους σε γραμμική διάταξη. Πριν ξεκινήσει η συγγραφή του οδηγού, ήταν απαραίτητη η μελέτη του φύλλου δεδομένων και η ανάπτυξη ενός βασικού αλγορίθμου που θα έθετε τις βάσεις για τη συνέχεια.

Η ανάπτυξη του κώδικα ξεκίνησε αφού υπήρχε ξεκάθαρη εικόνα της αρχής λειτουργίας του υλικού που θα χρησιμοποιούνταν. Μεγάλη προσοχή δόθηκε στα υποσυστήματα των PRU, ADC και ελεγκτή διακοπών (interrupt controller, INTC), καθώς έπρεπε να δοκιμαστούν και να επαληθευτεί η λειτουργία τους πριν εισαχθούν στον κώδικα. Με την εξακρίβωση της ομαλής λειτουργίας των υποσυστημάτων και την ενσωμάτωση των λειτουργιών τους στον κώδικα, ο οδηγός άρχισε να παίρνει μορφή.

Στα επόμενα κεφάλαια περιγράφονται με περισσότερη λεπτομέρεια όλα τα υποσυστήματα του BeagleBone που χρησιμοποιούνται, καθώς και το υλικό.

3 Υλικό

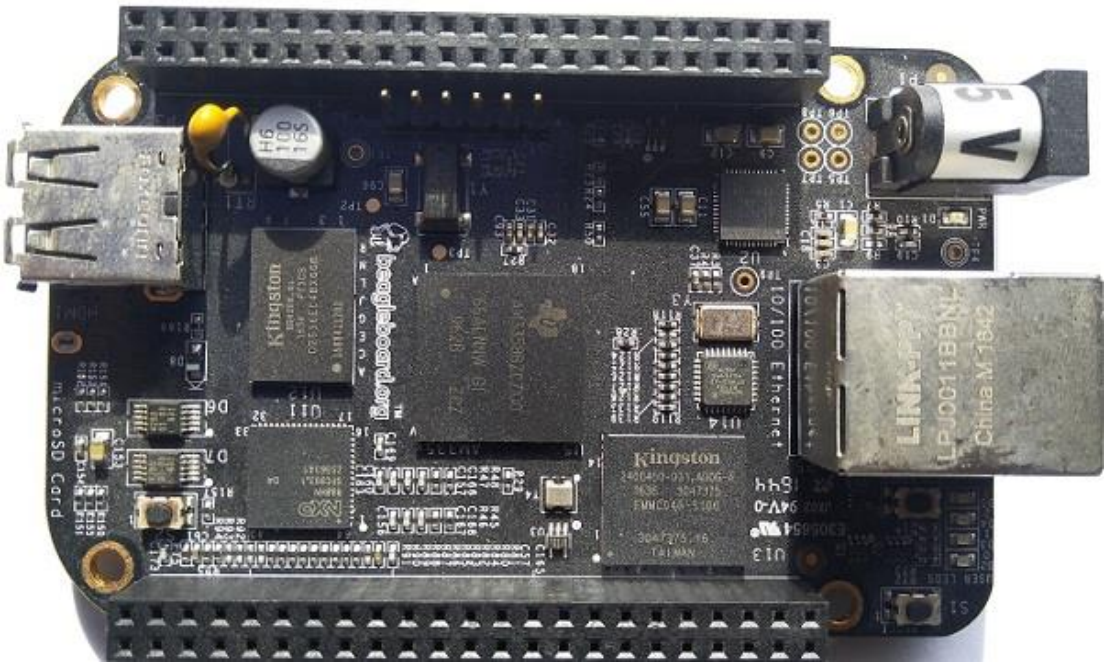
Σε αυτό το κεφάλαιο περιγράφεται η διάταξη του υλικού του συστήματος και τα επιμέρους κομμάτια του. Μεγάλη προσοχή δίνεται στο BeagleBone, καθώς είναι σημαντικό να παρουσιαστεί η αρχή λειτουργίας κάθε υποσυστήματος που χρησιμοποιείται από τον οδηγό. Στη συνέχεια ακολουθεί λεπτομερής ανάλυση του αισθητήρα PDA και του κυκλώματος που αναλαμβάνει την συντήρηση του αναλογικού σήματος του αισθητήρα.

3.1 BeagleBone Black

Κατά την διάρκεια των τελευταίων 8 ετών, αρκετοί υπολογιστές μονής πλακέτας (single-board computers, SBC), έκαναν την εμφάνισή τους στην αγορά . Στόχος τους είναι οι φοιτητές και οι ερασιτέχνες, καθώς είναι χαμηλότερης αξίας και με περισσότερη επεξεργαστική ισχύ από τους κοινούς μικροελεγκτές. Υποστηριζόμενα από τους κατασκευαστές αλλά και από τεράστιες διαδικτυακές κοινότητες, τα SBC χρησιμοποιούνται σε εκατοντάδες διαφορετικές εφαρμογές, από απλές “Internet of Things” (IoT), μέχρι σύνθετες συστοιχίες εξόρυξης κρυπτονομισμάτων. Σε αυτή την κατηγορία υπολογιστών περιλαμβάνεται και το BeagleBone Black που χρησιμοποιείται στην παρούσα εργασία.

Το BBB χρησιμοποιεί τον επεξεργαστή AM3358 Sitara, που κατασκευάζει η Texas Instruments (TI). Ενώ η TI αναφέρεται στο ολοκληρωμένο ως «επεξεργαστή», στην πραγματικότητα διαφέρει από τους συμβατικούς επεξεργαστές καθώς πέρα από την CPU και την μνήμη cache, περιέχει διάφορα υποσυστήματα που συναντώνται συνήθως σε μικροελεγκτές. Η συνήθης ονομασία για αυτού του είδους τα ολοκληρωμένα είναι System on Chip (SoC) [5].

Με σκοπό την διατήρηση του μήκους του εγγράφου σε λογικά πλαίσια, στις επόμενες παραγράφους περιγράφονται μόνο τα υποσυστήματα του SoC που χρησιμοποιούνται από τον οδηγό.



Σχήμα 3.1 Ο μικροϋπολογιστής BeagleBone Black

3.1.1 ARM Cortex A8

Ο βασικός επεξεργαστής του SoC, είναι ένας ARM Cortex A8 με ρολόι χρονισμένο στο 1GHz. Η συγκεκριμένη CPU είναι υπεύθυνη για το OS και την εκκίνηση του οδηγού, ενώ ο ρόλος της κατά την εκτέλεση είναι περισσότερο υποστηρικτικός προς τους PRU:

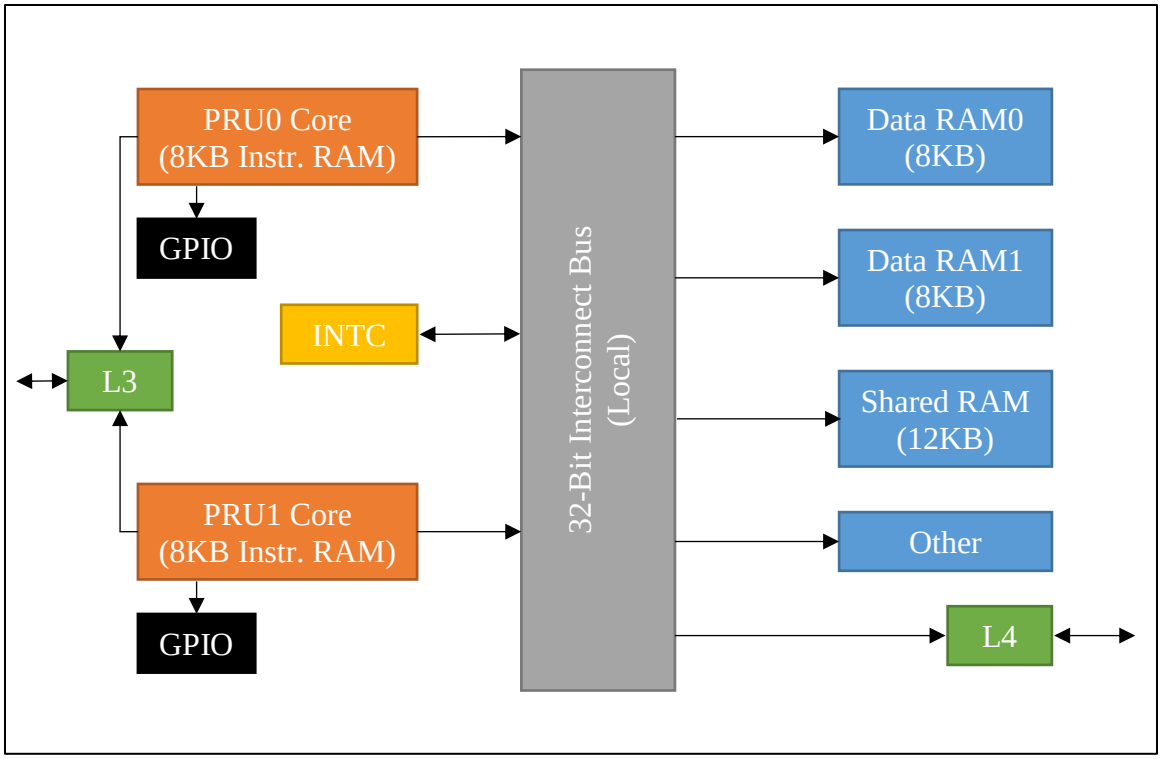
- Φόρτωση του οδηγού κατά την εκκίνηση του BeagleBone.
- Έλεγχος διαθέσιμης μνήμης σε σχέση με την ζητούμενη από τον οδηγό.
- Ενεργοποίηση των PRU και φόρτωση του κώδικα.
- Συγχρονισμός των PRU.
- Συλλογή των δεδομένων από την RAM και αποθήκευση στην κάρτα SD.

3.1.2 Programmable Realtime Units

Αυτό που κάνει το BBB να ξεχωρίζει από τα υπόλοιπα SBC, είναι η συμπερίληψη των δύο PRU στο SoC. Οι δύο επιπλέον αυτοί πυρήνες είναι βασισμένοι σε αρχιτεκτονική 32bit «Load/Store RISC», με ρολόι χρονισμένο στα 200MHz ενώ οι εντολές τους απαιτούν μόνο ένα κύκλο μηχανής για την εκτέλεσή τους, δηλαδή 5 nanoseconds (ns) [6]. Έχουν σχεδιαστεί με σκοπό να είναι ανεξάρτητοι ο ένας από τον άλλον και είναι στο χέρι του προγραμματιστή να επιλέξει αν θα χρησιμοποιήσει τον έναν ή και τους δύο. Κάθε ένας από τους δύο PRU έχει την δική του RAM των 8KB για κώδικα όπως και άλλα 8KB RAM για δεδομένα. Επιπλέον υπάρχει ακόμη μία περιοχή στην μνήμη, μεγέθους 12KB η οποία είναι προσπελάσιμη και από τους δύο πυρήνες. Οι PRU έχουν πρόσβαση στα περισσότερα περιφερειακά του SoC όπως και ο ARM, μέσω διαύλων διασύνδεσης.

Όπως υποδηλώνει και το όνομα, οι PRU λειτουργούν σε πραγματικό χρόνο με ντετερμινιστική συμπεριφορά. Δυστυχώς όμως, αν ένας PRU προσπελάσει κάποιο περιφερειακό του SoC μέσω των διαύλων διασύνδεσης L3 και L4, ο ντετερμινισμός παύει να ισχύει. Αυτό συμβαίνει επειδή είναι πιθανό ο δίαυλος να είναι απασχολημένος από κάποιο άλλο περιφερειακό, οπότε θα δημιουργηθεί καθυστέρηση αγνώστου χρόνου. Αντιθέτως, η επικοινωνία μεταξύ των δύο PRU όπως και με το GPIO, γίνεται άμεσα χωρίς να απαιτείται πρόσβαση σε κάποιο εξωτερικό δίαυλο, οπότε δεν δημιουργούνται χρονικές καθυστερήσεις. Το ίδιο ισχύει και για τον σκανδαλισμό διακοπών μέσω του INTC.

Η αρχιτεκτονική του υποσυστήματος των PRU προκάλεσε σύγχυση στο αρχικό στάδιο της ανάπτυξης καθώς χρησιμοποιούνταν μόνο ο ένας PRU και ήταν απαραίτητη η προσπέλαση περιφερειακών του SoC. Το πρόβλημα λύθηκε εισάγοντας και τον δεύτερο PRU με σκοπό να αναλάβει τις εργασίες που απαιτούν πρόσβαση σε πόρους που βρίσκονται εκτός του υποσυστήματος των PRU (Programmable Real Time Sub-System, PRUSS). Ο αλγόριθμος και η λογική που χρησιμοποιείται στους PRU αναλύεται στο κεφάλαιο 4, ενώ στο σχήμα 3.2 παρουσιάζεται το PRUSS.



Σχήμα 3.2 Απλοποιημένο μπλοκ διάγραμμα του PRU-ICSS

3.1.3 Μνήμη τυχαίας προσπέλασης (RAM)

Ως βασική μνήμη το BBB χρησιμοποιεί μία 512MB DDR3 από την Kingston, χρονισμένη στα 800MHz. Χρησιμοποιείται από το OS για την λειτουργία του αλλά και από τον οδηγό για την προσωρινή αποθήκευση των δεδομένων καθώς εξάγονται από το PDA.

Όπως αναφέρθηκε προηγουμένως, οι PRU έχουν την δική τους RAM η οποία είναι προσπελάσιμη και από τον ARM, γεγονός που την καθιστά κατάλληλη για γρήγορη ανταλλαγή δεδομένων μεταξύ των τριών πυρήνων.

3.1.4 Είσοδοι / Έξοδοι γενικής χρήσης

Ο κατασκευαστής της πλακέτας χωρίζει τους 65 ακροδέκτες εισόδου / εξόδου γενικής χρήσης (General Purpose I/O, GPIO) του SoC σε δύο ομάδες, με ονομασίες P8 και P9. Η τάση στην οποία λειτουργούν οι ακροδέκτες είναι τα 3.3V ενώ κανείς τους δεν μπορεί να απορροφήσει ή να τροφοδοτήσει περισσότερα από 8mA. Ο χρήστης μπορεί να επιλέξει ανάμεσα σε εφτά διαφορετικές λειτουργίες για κάθε ακροδέκτη, όμως πρέπει να γίνει με προσοχή, καθώς η προεπιλεγμένη λειτουργία κάποιων ακροδεκτών αφορά την θύρα HDMI και την μνήμη eMMC.

Οι PRU έχουν πρόσβαση σε συγκεκριμένους ακροδέκτες μέσω των καταχωρητών R30 για εγγραφή και R31 για ανάγνωση, με αντιστοίχιση ενός bit ανά ακροδέκτη. Εκτός από τους PRU, πρόσβαση στο GPIO έχει και ο ARM μέσω ενός αρχείου του OS. Ο χρόνος προσπέλασης ακροδέκτη από τους PRU είναι ένας κύκλος μηχανής για εγγραφή ή δύο κύκλοι για εντοπισμό αλλαγής της κατάστασης του. Οι χρόνοι προσπέλασης για τον ARM είναι σαφώς μεγαλύτεροι και ακαθόριστοι καθώς μεσολαβεί το OS.

Στη συγκεκριμένη εφαρμογή, οι ακροδέκτες που οδηγούν το PDA και τον ADC ελέγχονται από τους PRU. Ακροδέκτες που σχετίζονται με χρονικά μη κρίσιμες λειτουργίες όπως το πλήκτρο και το LED, ελέγχονται από τον ARM.

3.1.5 Μετατροπέας αναλογικού σήματος σε ψηφιακό

Για την ανάγνωση της εξόδου του PDA, χρησιμοποιείται ο ενσωματωμένος μετατροπέας αναλογικού σήματος σε ψηφιακό (Analog to Digital Converter, ADC). Η τάση αναφοράς (Analog Reference Voltage, V_{AREF}) του ADC είναι τα 1.8V, η οποία δεν πρέπει να ξεπεραστεί σε καμία από τις εισόδους, ειδάλλως θα καταστραφεί ο ακροδέκτης. Η μέγιστη συχνότητα του ADC είναι τα 24MHz ενώ μία πλήρης μετατροπή διαρκεί 15 κύκλους ρολογιού. Στα 24MHz ο μέγιστος ρυθμός δειγματοληψίας του ADC είναι τα 1.6Msps.

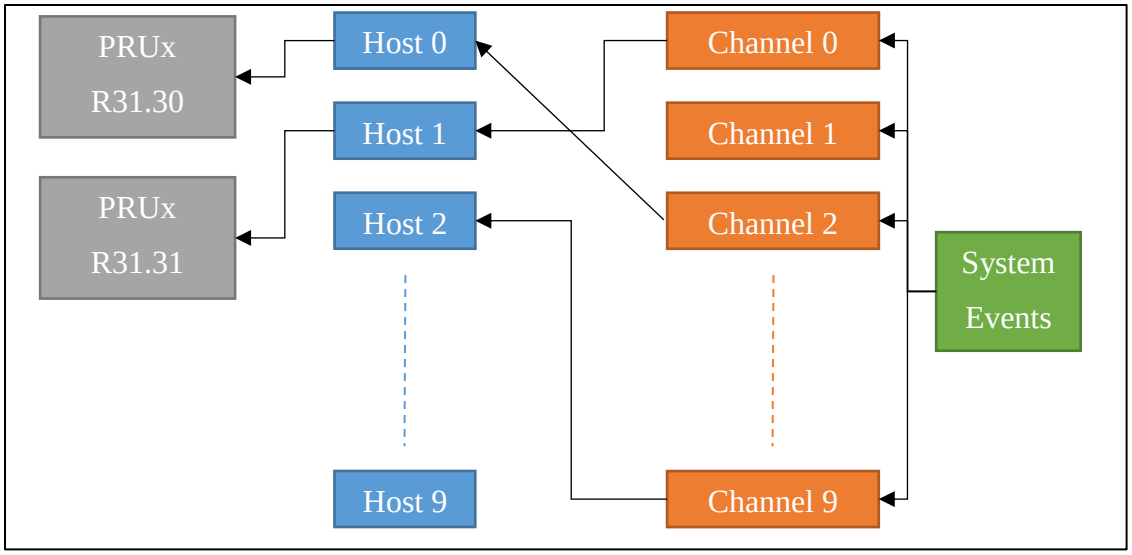
Οι απαραίτητες ενέργειες για την αρχικοποίηση και χρήση του ADC καλύπτονται από το φύλλο δεδομένων που παρέχει η T.I. Η αρχικοποίηση και ο έλεγχος του ADC γίνονται από τον PRU1.

3.1.6 Ελεγκτής διακοπών

Ο ελεγκτής διακοπών που περιέχεται στο SoC, χρησιμοποιείται για τον σκανδαλισμό γεγονότων και την επικοινωνία μεταξύ των υποσυστημάτων. Στη συγκεκριμένη εφαρμογή, η χρήση του περιορίζεται στους PRU και τον ADC. Συγκεκριμένα, ο PRU0 ειδοποιεί μέσω διακοπής τον PRU1 ώστε να ξεκινήσει την διαδικασία της δειγματοληψίας. Με τη σειρά του ο PRU1 εκκινεί τον ADC μέσω διακοπής και περιμένει μέχρι να σηκωθεί η σημαία ολοκλήρωσης της μετατροπής.

Πριν κάποιο γεγονός προκαλέσει διακοπή, πρέπει πρώτα να χαρτογραφηθεί σωστά στο σύστημα “channel – host” που έχει ενσωματώσει η T.I στο SoC. Πρόκειται για έναν περίπλοκο αλλά αποτελεσματικό μηχανισμό συνδέσεων των πόρων του ολοκληρωμένου, του οποίου η ρύθμιση επιτυγχάνεται αρχικά τροποποιώντας αρχεία του OS και κατά την εκτέλεση του προγράμματος μέσω κλήσης ειδικών συναρτήσεων του συστήματος.

Στη συγκεκριμένη περίπτωση, οι εργοστασιακές ρυθμίσεις του INTC εξυπηρετούσαν τις ανάγκες για γρήγορη επικοινωνία μεταξύ των πυρήνων και του ADC, οπότε και δεν χρειάστηκε να γίνει κάποια τροποποίηση στο OS. Είναι σημαντικό να αναφερθεί πως ενώ ο σκανδαλισμός διακοπής από τους PRU διαρκεί ένα κύκλο μηχανής, ο μηδενισμός της σημαίας διακοπής του ADC γίνεται μέσω πρόσβασης σε δίαυλο διασύνδεσης. Αυτή η εργασία έχει ανατεθεί στον PRU1, ο οποίος επιτρέπεται να χάσει τον ντετερμινισμό του κατά την διαδικασία της οδήγησης του αισθητηρίου όπως θα εξηγηθεί αργότερα στο έγγραφο. Στο σχήμα 3.3 φαίνεται το σύστημα “channel – host” σε μορφή απλοποιημένου μπλοκ διαγράμματος.



Σχήμα 3.3 Απλοποιημένο μπλοκ διάγραμμα του Ελεγκτή Διακοπών

3.1.7 Device Tree Overlay

Το Device Tree δεν είναι κομμάτι του υλικού, αλλά μία δομή δεδομένων που περιγράφει το υλικό του SoC ώστε να είναι εντοπίσιμο από το OS [7]. Η διανομή Debian που χρησιμοποιείται κατά κόρον στο BBB, έχει φορτωμένο ένα DTO το οποίο συσχετίζει κάποιους από τους ακροδέκτες του GPIO με τη θύρα HDMI. Κάποιοι από αυτούς τους ακροδέκτες χρειάστηκε να χρησιμοποιηθούν για την οδήγηση του PDA για τους σκοπούς της εργασίας, οπότε ο συγγραφέας επέλεξε την απενεργοποίηση του προ εγκατεστημένου DTO και την συγγραφή ενός καινούριου, το οποίο φορτώνεται από το OS αυτόματα με την εκκίνηση του συστήματος. Ολόκληρο το αρχείο DTO μαζί με σχόλια, βρίσκεται στο παράρτημα του εγγράφου.

3.2 AMS TSL1401CL

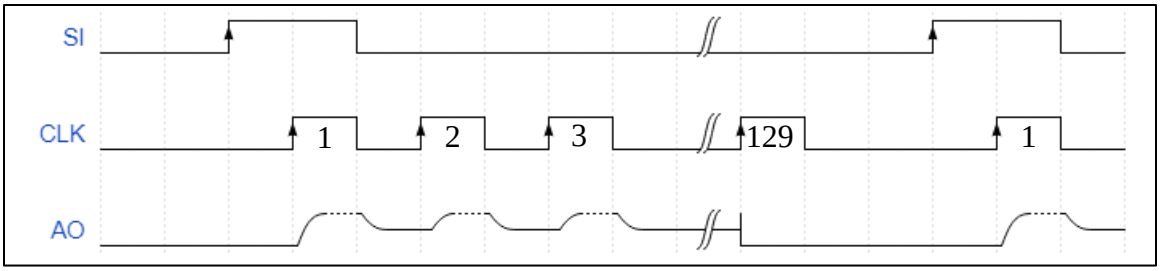
Ο αισθητήρας TSL1401CL είναι ένας γραμμικός πίνακας 128 φωτοδιόδων ή αλλιώς εικονοστοιχεία (pixels) [8]. Ουσιαστικά πρόκειται για μία κάμερα των 128 γραμμικά στοιχισμένων pixels, η οποία μπορεί να καταγράψει μόνο την ένταση του φωτός

χωρίς χρώματα. Στο ίδιο κέλυφος με τον αισθητήρα, ο κατασκευαστής έχει τοποθετήσει κύκλωμα ενισχυτή φόρτισης καθώς και έναν ελεγκτή ολίσθησης bit, για την προώθηση του φορτίου των φωτοδιόδων στην έξοδο. Κάθε φωτοδίοδος αποτελείται από δύο πυκνωτές, ένα κύκλωμα δειγματοληψίας – συγκράτησης και δύο διακόπτες. Η χρονική περίοδος κατά την οποία ο αισθητήρας δειγματοληπτεί την φωτεινότητα ονομάζεται ολοκλήρωση, ενώ το σύνολο των 128 τιμών του ονομάζεται δείγμα.

Η τάση λειτουργίας του αισθητήρα μπορεί να είναι στο εύρος των 3V με 5V και η έξοδος μπορεί να εκμεταλλευτεί πλήρως τη μέγιστη τάση που θα του τροφοδοτηθεί. Για τη λειτουργία του το PDA απαιτεί δύο ψηφιακά σήματα, τα οποία ο κατασκευαστής ονομάζει “serial-input” (SI) και “clock” (CLK) με μέγιστη επιτρεπτή συχνότητα τα 8MHz. Έχει μία μοναδική αναλογική έξοδο μέσω της οποίας εξάγει το φορτίο των φωτοδιόδων, ένα τη φορά, χρησιμοποιώντας τον ελεγκτή ολίσθησης bit.

Το σήμα SI σηματοδοτεί την αρχή της περιόδου ολοκλήρωσης του PDA καθώς και την εξαγωγή των δεδομένων. Στο διάστημα μεταξύ δύο ανερχόμενων παρυφών SI, ο αισθητήρας ολοκληρώνει το τρέχον δείγμα και εξάγει το προηγούμενο. Τα δεδομένα εξάγονται ένα τη φορά, με κάθε ανερχόμενη παρυφή του σήματος CLK. Με το τέλος του παλμού SI, δηλαδή με την πίπτουσα παρυφή του, ο αισθητήρας περιμένει να δει 128 παλμούς στην είσοδο CLK για να εξάγει πλήρως το προηγούμενο δείγμα. Ένας επιπλέον παλμός CLK σηματοδοτεί το τέλος της διαδικασίας, επομένως 129 παλμοί CLK και ένας SI απαιτούνται για ένα δείγμα όπως φαίνεται στο σχήμα 3.4. Η ολοκλήρωση του φορτίου στις φωτοδιόδους ξεκινάει με την 19^{η} ανερχόμενη παρυφή του σήματος CLK. Ο χρόνος των 18 πρώτων παλμών πρέπει να αφαιρεθεί από τον συνολικό κατά τον υπολογισμό του χρόνου έκθεσης του αισθητηρίου.

Πρόκειται για αρκετά σύνθετη διαδικασία η οποία όμως επιτρέπει την κατασκευή του PDA με μικρό κέλυφος και απλό σύστημα εισόδων / εξόδων.



Σχήμα 3.4 Λειτουργική κυματομορφή του PDA

3.3 Μορφοποίηση Σήματος

Όπως αναφέρθηκε προηγουμένως, η V_{AREF} του BBB είναι στα 1.8V, ενώ η ελάχιστη τάση που δέχεται το PDA είναι στα 3.3V. Αυτό σημαίνει ότι δεν μπορεί να υπάρξει άμεση σύνδεση των δύο εξαρτημάτων και απαιτείται ένα ενδιάμεσο κύκλωμα που θα μορφοποιεί το σήμα κατάλληλα, κλιμακώνοντάς το από το εύρος των 0-3.3V σε αυτό των 0-1.8V που μπορεί να δεχτεί το BBB χωρίς να υποστεί ζημιά.

Η μορφοποίηση του σήματος επιτυγχάνεται με τη χρήση τελεστικών ενισχυτών. Οι απαιτήσεις για τα χαρακτηριστικά ενός τελεστικού ενισχυτή συνήθως θέτονται από τις ανάγκες της εκάστοτε εφαρμογής. Στη περίπτωση της παρούσας εργασίας κρίθηκε μεγάλης σημασίας ο ρυθμός μεταβολής της εξόδου του τελεστικού ενισχυτή, καθώς η εξαγωγή των δεδομένων από το PDA γίνεται σε υψηλή συχνότητα. Σε περίπτωση χαμηλού ρυθμού μεταβολής, η έξοδος του PDA θα ενημερώνεται συχνότερα από ότι αυτή του τελεστικού ενισχυτή και ο ADC θα μετατρέπει λάθος δείγμα. Συγκεκριμένα η έξοδος του PDA ενημερώνεται με συχνότητα 100KHz και σε συνδυασμό με το εύρος των 0-3.3V, απαιτείται τελεστικός ενισχυτής με ρυθμό μεταβολής τουλάχιστον 330mV/μs θεωρώντας πάντα ιδανικές συνθήκες.

Δεύτερο σημαντικό κριτήριο ήταν η υποστήριξη “rail to rail” σε είσοδο και έξοδο, δηλαδή η εκμετάλλευση ολόκληρου του εύρους της τάσης λειτουργίας του τελεστικού ενισχυτή χωρίς να υπάρχει σημαντική πτώση της τάσης στο άνω άκρο ή άνωση στο κάτω άκρο [9].

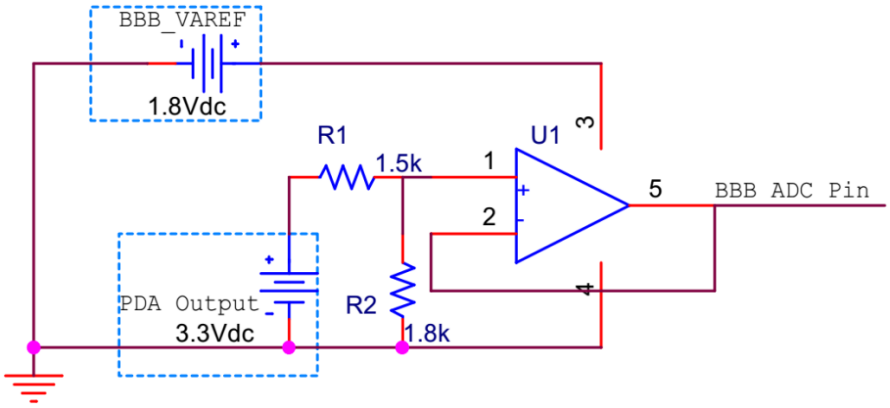
Έχοντας θέσει τα προηγούμενα δύο κριτήρια, επιλέχθηκε ο τελεστικός ενισχυτής LM6142 της T.I. Πρόκειται για ένα σχετικά οικονομικό ολοκληρωμένο, σε DIP κέλυφος που παράγεται μέχρι σήμερα και περιέχει δύο τελεστικούς. Ο ρυθμός μεταβολής του είναι στα 25V/μs και υποστηρίζει rail to rail τόσο σε έξοδο όσο και σε είσοδο.

Ο τελεστικός χρησιμοποιείται ως ακόλουθος τάσης με έναν διαιρέτη τάσης ανάμεσα στην είσοδό του και την έξοδο του PDA για την κλιμάκωση του σήματος όπως φαίνεται στο σχήμα 3.5. Η συγκεκριμένη διάταξη πετυχαίνει γραμμική κλιμάκωση του σήματος ενώ ταυτόχρονα παρέχει απομόνωση μεταξύ PDA και BBB [10]. Για την επιλογή των αντιστάσεων χρησιμοποιήθηκε η κλασική εξίσωση του διαιρέτη τάσης όπως φαίνεται παρακάτω.

$$V_{out} = \frac{R2}{R1+R2} * V_{in} \leftrightarrow \frac{V_{out}}{V_{in}} = \frac{R2}{R1+R2} \leftrightarrow \frac{3.3}{1.8} = \frac{R2}{R1+R2}$$

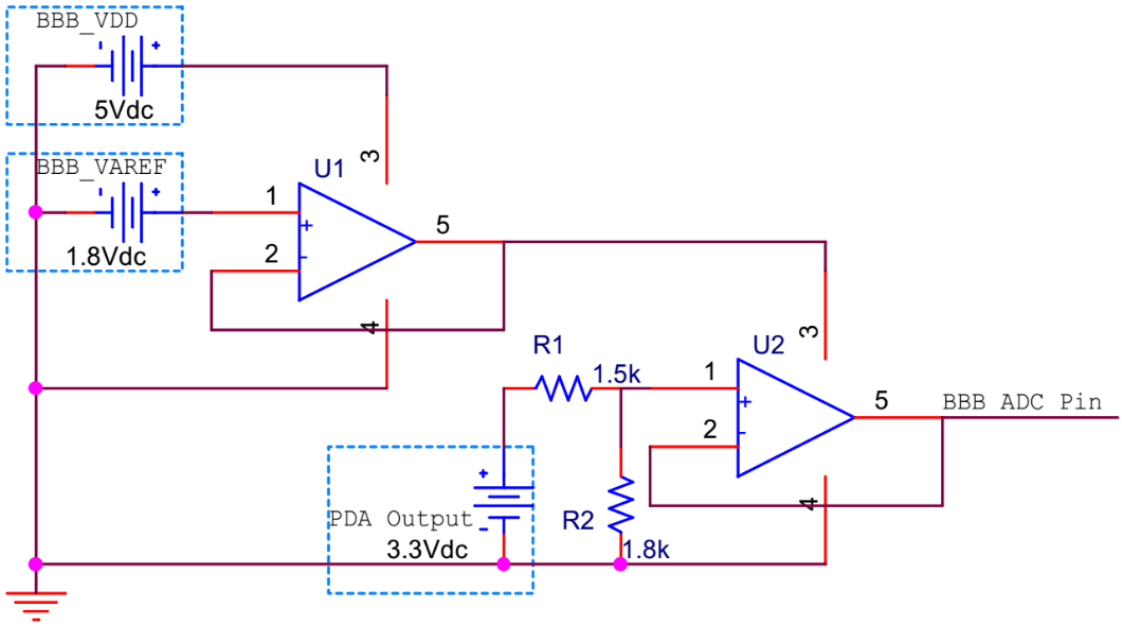
$$\leftrightarrow 0.5454 = \frac{R2}{R1+R2} \leftrightarrow 0.5454 = \frac{1800}{R1+1800}$$

$$\leftrightarrow 0.5454 = \frac{1800}{1500+1800}$$

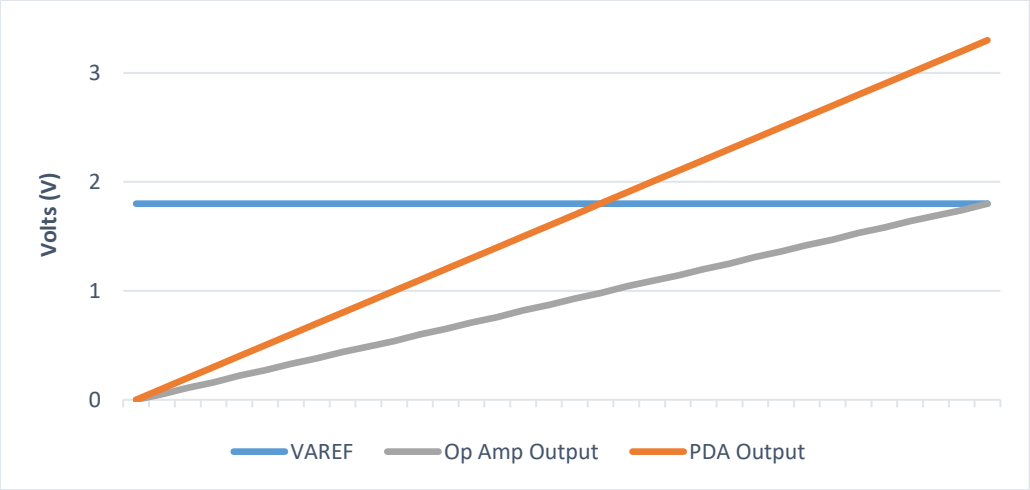


Σχήμα 3.5 Γραμμική κλιμάκωση του σήματος με ακόλουθο τάσης στο OrCAD Capture

Το κύκλωμα του προηγούμενου σχήματος έχει μία αδυναμία. Ο ακροδέκτης V_{AREF} του BBB χρησιμοποιείται ως τροφοδοσία στον τελεστικό ενισχυτή, του οποίου η έξοδος συνδέεται άμεσα με την αναλογική είσοδο του ADC. Αυτό έχει ως αποτέλεσμα την κατανάλωση ισχύος μέσω του ακροδέκτη V_{AREF} και κατ' επέκταση την πτώση τάσης από τα 1.8V. Το πρόβλημα αυτό λύνεται εύκολα τοποθετώντας έναν ακόμη τελεστικό ενισχυτή (U1), με τη μη αναστρέφουσα είσοδο του στα 1.8V και την έξοδο του στη τροφοδοσία του προηγούμενου (U2), όπως φαίνεται στο σχήμα 3.6. Με αυτή τη συνδεσμολογία το φορτίο του κυκλώματος οδηγείται από την πηγή των 5V του BBB ενώ η V_{AREF} παραμένει σταθερή στα 1.8V. Η κλιμάκωση του σήματος φαίνεται στο σχήμα 3.7.



Σχήμα 3.6 Κύκλωμα μορφοποίησης σήματος



Σχήμα 3.7 Γραμμική κλιμάκωση της εξόδου του PDA

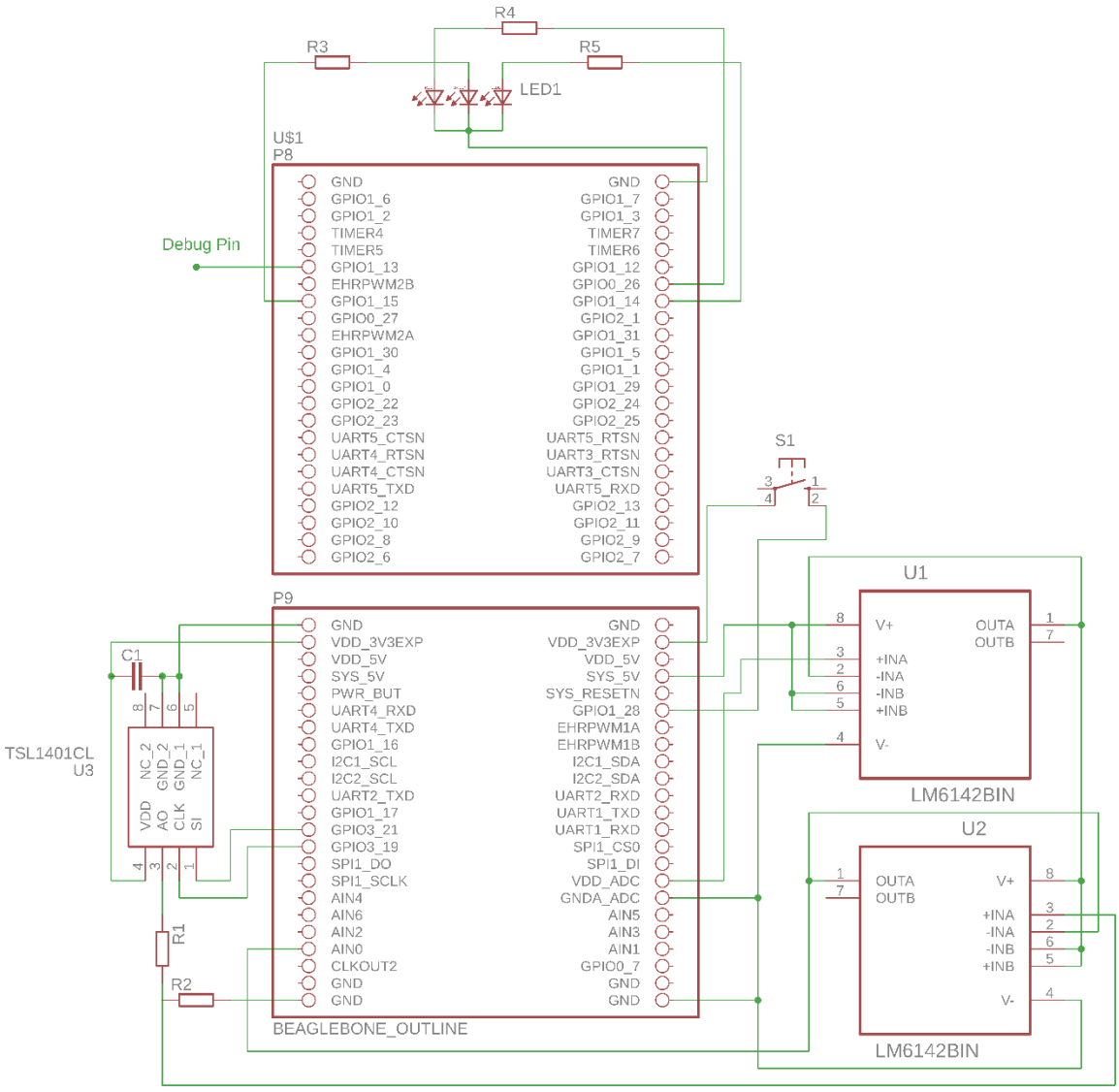
3.4 Υπόλοιπο υλικό και σχηματικό

Πέρα από το υλικό που ήδη περιεγράφηκε, για την λειτουργία του συστήματος χρησιμοποιείται ένα πλήκτρο που εκκινεί την οδήγηση του αισθητηρίου. Επιπλέον, ένα τρίχρωμο LED δείχνει την κατάσταση στην οποία βρίσκεται το σύστημα.

Για το τρίχρωμο LED χρησιμοποιήθηκαν οι παρακάτω τιμές των αντιστάσεων με σκοπό να περιοριστεί το ρεύμα των ακροδεκτών σε ασφαλή επίπεδα.

- Κόκκινο – 480
- Μπλε – 330Ω
- Πράσινο 330Ω

Το σχηματικό της συσκευής φαίνεται στο σχήμα 3.8.



Σχήμα 3.8 Συνολικό σχηματικό του συστήματος

4 Λογισμικό

Το κεφάλαιο αυτό είναι αφιερωμένο στο λογισμικό που γράφτηκε για την λειτουργία του συστήματος. Στις σελίδες που ακολουθούν, παρουσιάζεται η διαδικασία οδήγησης του αισθητήρα τόσο στο χαμηλό όσο και στο υψηλό επίπεδο, μαζί με τα λογικά διαγράμματα.

Μεγαλύτερη βαρύτητα δίνεται στο κομμάτι του αλγορίθμου όπου υπολογίζονται οι κρίσιμοι χρόνοι του συστήματος.

4.1 Απαιτήσεις

Η ιατρική φύση της συσκευής για την οποία έγινε η συγγραφή του οδηγού της εργασίας, θέτει υψηλές απαιτήσεις ως προς την ακρίβεια των χρόνων που υπολογίζονται και μετρούνται κατά την λειτουργία της. Συγκεκριμένα, υπάρχουν τέσσερις συντελεστές που καθορίζουν τη συμπεριφορά του συστήματος, με όλους να έχουν σχεδόν τον ίδιο σημαντικό ρόλο, χωρίς τη δυνατότητα υποτίμησης κάποιου για την καλύτερη εξυπηρέτηση ενός άλλου. Κανένας από τους συντελεστές δεν είχε οριστική τιμή κατά την ανάπτυξη, καθώς πρόκειται για νέα συσκευή της οποίας τα τελικά χαρακτηριστικά θα προέκυπταν ύστερα από πειράματα.

4.1.1 Πλήθος καρέ

Πρώτος συντελεστής είναι το πλήθος των καρέ που πρέπει το σύστημα να καταγράψει. Κάθε καρέ ή δείγμα, αποτελείται από 128 τιμές που λαμβάνονται από τον ADC και αποθηκεύονται προσωρινά στη μνήμη RAM του BeagleBone. Μοναδική πρόκληση του συγκεκριμένου συντελεστή αποτελεί το μέγεθος της μνήμης των PRU η οποία όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, ανέρχεται συνολικά και για τους δύο μαζί με την κοινόχρηστη RAM, στα 28KB. Με το αποτέλεσμα της μετατροπής του ADC

να δεσμεύει 2 bytes και τις 128 τιμές ανά δείγμα, χρειάζονται 256 byte για κάθε καρέ. Μία αρχική εκτίμηση για το πλήθος των καρέ ήταν τα 250, τιμή που εκτοξεύει την ανάγκη για μνήμη στα 64KB και καθιστά την αποθήκευση των δεδομένων στις μνήμες των PRU αδύνατη. Για το λόγο αυτό, χρειάζεται η πρόσβαση στα 512MB της DDR RAM του BeagleBone, η οποία όμως αναιρεί τη ντετερμινιστική ιδιότητα των PRU. Επιπλέον, η συγκεκριμένη μνήμη χρησιμοποιείται από το OS με δυναμικό τρόπο, οπότε μία από τις πρώτες διεργασίες που εκτελεί ο οδηγός, είναι η αίτηση για δέσμευση μνήμης. Σε περίπτωση αποτυχίας της δέσμευσης, τερματίζεται η λειτουργία και επιστρέφεται ο κατάλληλος κωδικός σφάλματος.

4.1.2 Χρόνος ολοκλήρωσης

Το PDA που χρησιμοποιείται για την ανάκτηση των δεδομένων, μπορεί να ολοκληρώσει σε εύρος χρόνου από 33.75μs μέχρι 22.02ms. Υπενθυμίζεται ότι σύμφωνα με την αρχή λειτουργίας του αισθητηρίου, κατά τη διαδικασία ολοκλήρωσης, γίνεται και η εξαγωγή των δεδομένων του προηγούμενου κύκλου, πράγμα που περιπλέκει τη διαδικασία και απαιτείται ειδικός χειρισμός που περιγράφεται σε επόμενη ενότητα.

Αρχικός στόχος ήταν να τηρηθεί μέγιστο επιτρεπτό σφάλμα της τάξης του 1%. Για τον ελάχιστο χρόνο των 33.75μs που υποστηρίζει ως χρόνο ολοκλήρωσης το PDA, το μέγιστο σφάλμα είναι 0.337μs. Πρόκειται για εξαιρετικά χαμηλούς χρόνους που δεν συναντώνται συχνά σε προδιαγραφές συστημάτων.

4.1.3 Καρέ ανά δευτερόλεπτο

Τα καρέ ανά δευτερόλεπτο (frames per second, fps), είναι η ταχύτητα με την οποία το σύστημα καταγράφει την ένταση του φωτός στο οποίο εκτίθεται. Χρησιμοποιείται μία τυπική τιμή των 50fps, η οποία μαζί με το πλήθος των καρέ που είναι στα 250, ορίζει τη συνολική διάρκεια της δειγματοληψίας στα 5s. Με 50fps πρέπει κάθε καρέ να

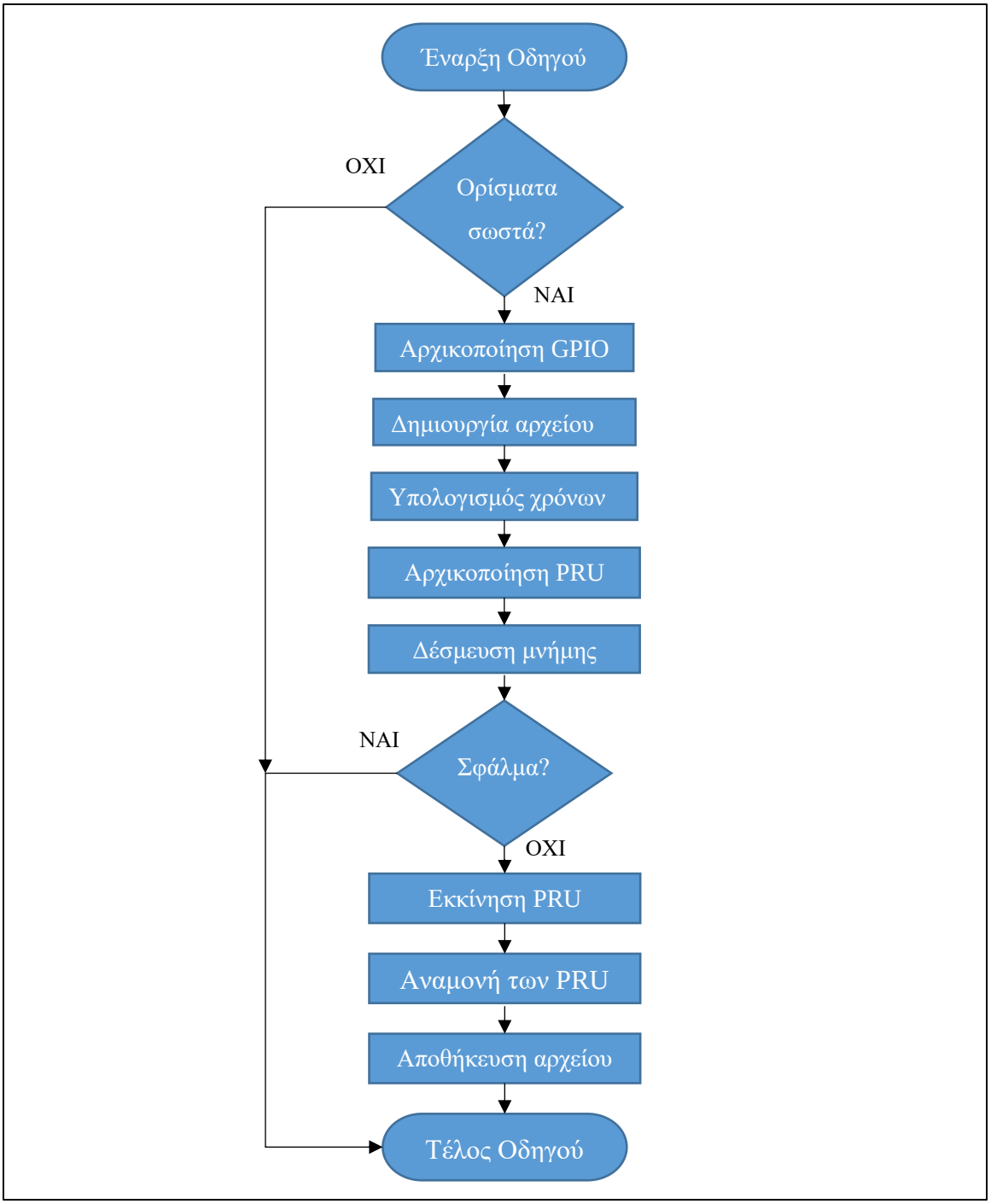
ολοκληρώνεται αλλά και να εξάγεται από το PDA στο χρονικό παράθυρο των 20ms. Ο τρόπος με τον οποίο επιτυγχάνεται παρουσιάζεται σε επόμενη ενότητα του κεφαλαίου.

4.1.4 Συχνότητα ρολογιού αισθητήρα

Η συχνότητα του σήματος CLK που τροφοδοτείται στον αισθητήρα, ορίζει την ταχύτητα με την οποία εξάγει τα δεδομένα αλλά και τον χρόνο ολοκλήρωσης. Για ακόμη μία φορά επαναλαμβάνεται πως οι ταυτόχρονες αυτές εργασίες απαιτούν ειδικό χειρισμό ώστε να εξυπηρετηθούν όλες οι προδιαγραφές με την ίδια ακρίβεια. Η συχνότητα του CLK, είναι ο μοναδικός συντελεστής που παρέμεινε σταθερός από την αρχή της εργασίας, στα 100KHz, συνεχίζει όμως να είναι συντελεστής εισόδου.

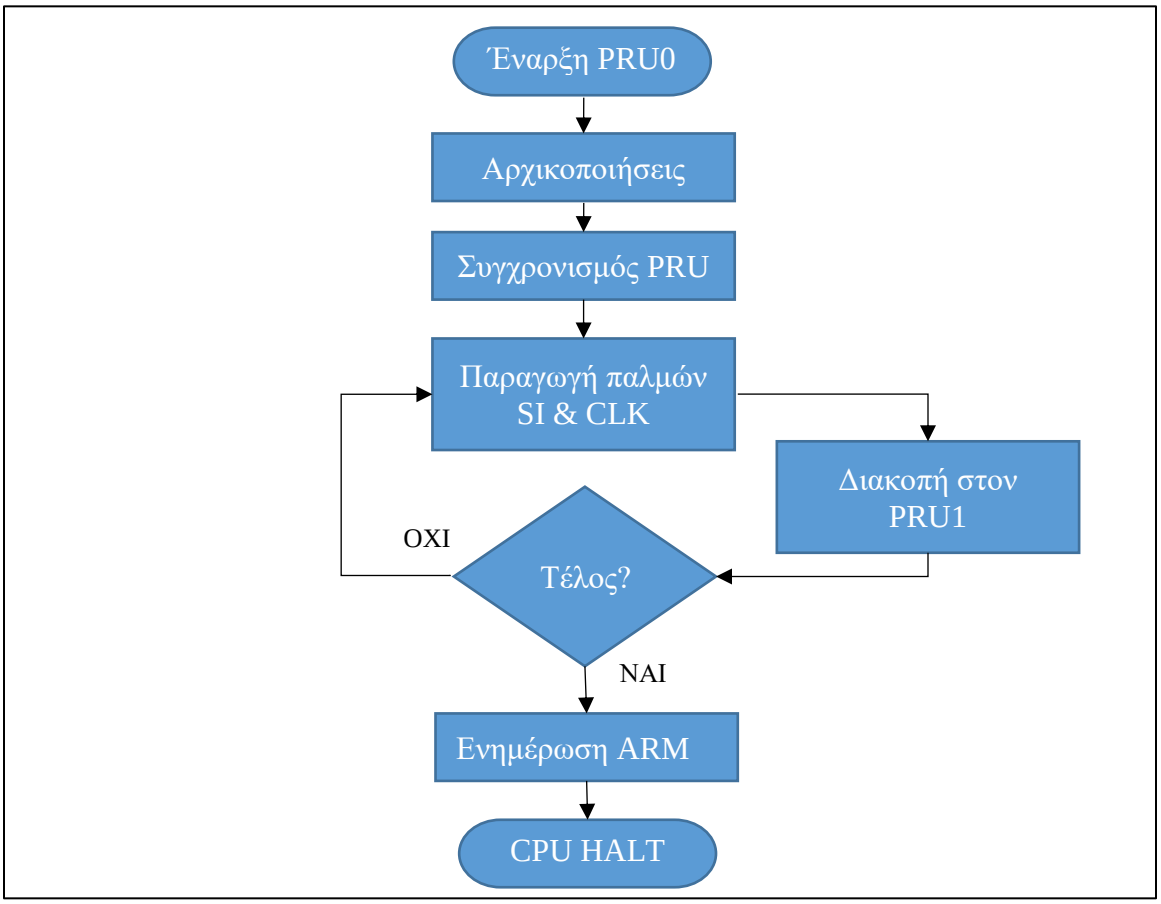
4.2 Γενική εικόνα αλγορίθμου

Ως δεδομένα εισόδου για τον αλγόριθμο, ορίζονται οι τέσσερις συντελεστές της προηγούμενης ενότητας. Κατά την εκκίνηση, γίνεται έλεγχος ορθότητας στα ορίσματα με σκοπό τον τερματισμό του προγράμματος σε περίπτωση που έχει δοθεί κάποια λάθος τιμή. Στη συνέχεια ακολουθεί η αρχικοποίηση του GPIO που χρησιμοποιείται από τον ARM και αφορά τους ακροδέκτες του τρίχρωμου LED και αυτόν του πλήκτρου. Ακολουθεί η δημιουργία του αρχείου όπου θα αποθηκευτούν τα δεδομένα, ο υπολογισμός των χρονισμών από τα ορίσματα του χρήστη, η αρχικοποίηση των PRU και η δέσμευση μνήμης μέσω του OS μαζί με την τοποθέτηση των απαραίτητων δεδομένων στις μνήμες των PRU για να τα χρησιμοποιήσουν κατά την λειτουργίας τους. Σε αυτό το σημείο, ο ARM συγχρονίζει τους δύο PRU μέσω μίας διαδικασίας χειραψίας και κολλάει η εκτέλεση του κώδικα του, μέχρι να ειδοποιηθεί από τον PRU1 πως ολοκληρώθηκε η διαδικασία συλλογής των δεδομένων. Στο σχήμα 4.1 παρουσιάζεται το απλοποιημένο διάγραμμα ροής του αλγορίθμου που εκτελεί ο επεξεργαστής ARM.



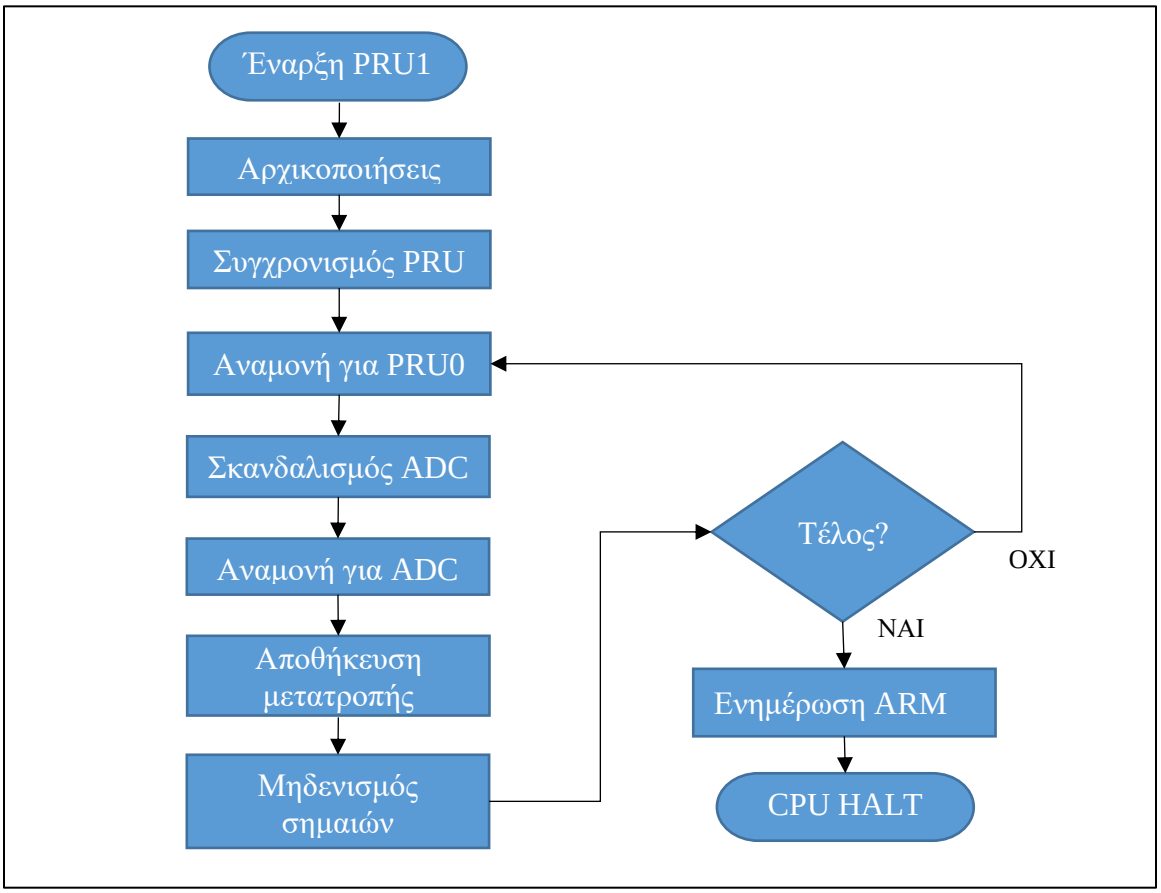
Σχήμα 4.1 Απλοποιημένο διάγραμμα ροής του αλγορίθμου του ARM

Ο PRU0 αναλαμβάνει εξ ολοκλήρου την παραγωγή των σημάτων που οδηγούν τον αισθητήρα χωρίς να χρησιμοποιεί κάποιον από τους διάδρομους διασύνδεσης L3 ή L4. Έτσι διατηρεί την ντετερμινιστική του συμπεριφορά και σε συνδυασμό με την χρήση της γλώσσας Assembly για τον προγραμματισμό του, δίνεται η δυνατότητα υπολογισμού του χρόνου εκτέλεσης των ρουτινών του με ακρίβεια ενός κύκλου μηχανής, δηλαδή 5ns. Την κατάλληλη χρονική στιγμή, ειδοποιεί μέσω του INTC, τον PRU1 ώστε να ξεκινήσει τη διαδικασία μετατροπής του ADC. Αυτή η διαδικασία επαναλαμβάνεται συνεχώς, μέχρι να συμπληρωθεί ο απαιτούμενος αριθμός επαναλήψεων που ορίζεται από το γινόμενο του πλήθους των καρέ και των φωτοδιόδων. Οι χρονικές καθυστερήσεις που παρουσιάστηκαν στην προηγούμενη ενότητα, υλοποιούνται με χρήση ρουτινών καθυστέρησης και θα εξηγηθούν σε επόμενη ενότητα του παρόντος κεφαλαίου. Στο σχήμα 4.2 φαίνεται το απλοποιημένο διάγραμμα ροής του αλγορίθμου που τρέχει ο PRU0.



Σχήμα 4.2 Απλοποιημένο διάγραμμα ροής του αλγόριθμου του PRU0

Κατά την εκκίνηση του, ο PRU1 αναλαμβάνει την αρχικοποίηση του ADC και του INTC για να μπορέσουν να χρησιμοποιηθούν από το PRUSS. Μετά την χειραψία με τον ARM, πέφτει σε έναν βρόχο αναμονής και περιμένει μέχρι να τον ειδοποιήσει ο PRU0. Στη συνέχεια, σκανδαλίζει τον ADC και ελέγχει ασταμάτητα την σημαία ολοκλήρωσης μετατροπής μέχρι να γίνει '1', όπου την καθαρίζει και αποθηκεύει το αποτέλεσμα στη μνήμη DDR του BeagleBone. Ακολουθεί ο μηδενισμός σημαιών INTC και ADC ενώ στο τέλος, μειώνει τον μετρητή επαναλήψεων και ελέγχει αν ολοκληρώθηκε η διαδικασία. Αν δεν ήταν η τελευταία μετατροπή, επιστρέφει στο βρόχο αναμονής μέχρι να τον ειδοποιήσει ο PRU0 και να επαναλάβει την εργασία του. Σε περίπτωση όπου ήταν η τελευταία του επανάληψη, ειδοποιεί τον ARM μέσω διακοπής και κολλάει την εκτέλεση του μέχρι να απενεργοποιηθεί πλήρως από τον ARM. Το απλοποιημένο διάγραμμα ροής του PRU1 παρουσιάζεται στο σχήμα 4.3.

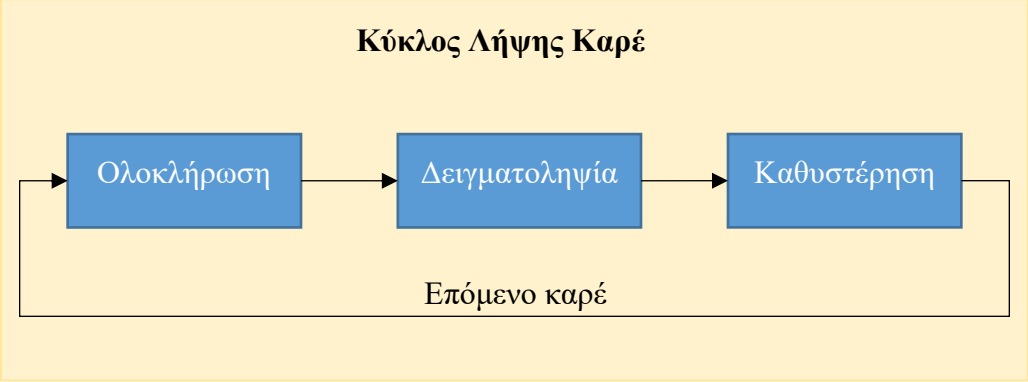


Σχήμα 4.3 Απλοποιημένο διάγραμμα ροής του αλγόριθμου του PRU1

4.3 Σύλληψη των καρέ

Η ουσία της παρούσας πτυχιακής εργασίας βρίσκεται στον τρόπο σύλληψης των καρέ, διατηρώντας τα χρονικά σφάλματα σε χαμηλά επίπεδα. Η μεγαλύτερη δυσκολία εισάγεται από την αρχή λειτουργίας του PDA και συγκεκριμένα τις ταυτόχρονες διαδικασίες ολοκλήρωσης και εξαγωγής δεδομένων. Είναι αδύνατον να λειτουργήσει σωστά ο οδηγός παραμένοντας πιστός στους συντελεστές που δόθηκαν από τον χρήστη εάν ολοκληρώνει και εξάγει δεδομένα παράλληλα. Για αυτό το λόγο επιλέχθηκε ο διαχωρισμός των διαδικασιών και η δημιουργία του «κύκλου λήψης καρέ».

Ως κύκλος λήψης καρέ ορίζεται η συνολική διαδικασία που απαιτείται για την ολοκλήρωση και τη δειγματοληψία 128 δεδομένων που απαρτίζουν ένα καρέ. Αποτελείται από τρεις διαφορετικές φάσεις, οι οποίες προκύπτουν από τους συντελεστές που εισάγει ο χρήστης. Στο σχήμα 4.4 παρουσιάζεται η ροή ενός κύκλου λήψης καρέ.



Σχήμα 4.4 Μπλοκ διάγραμμα ενός κύκλο λήψης καρέ

Η πρώτη φάση ονομάζεται ολοκλήρωση και είναι η ίδια ακριβώς με αυτή του PDA, όπου το αισθητήριο συλλέγει την ένταση του φωτός στις φωτοδιόδους του. Η χρονική διάρκεια της ολοκλήρωσης δίνεται άμεσα από τον χρήστη. Τα δεδομένα που εξάγονται από το αισθητήριο αγνοούνται και δεν συλλέγονται.

Δειγματοληψία ονομάζεται η φάση στην οποία εξάγονται από το αισθητήριο τα δεδομένα που συλλέχθηκαν κατά την ολοκλήρωση. Τη χρονική διάρκεια της δειγματοληψίας καθορίζει κυρίως η συχνότητα του παλμού CLK που δίνει ο χρήστης. Τα δεδομένα της ολοκλήρωσης που πραγματοποιεί το αισθητήριο κατά αυτή τη φάση δεν συλλέγονται αλλά αγνοούνται καθώς δεν είναι χρήσιμα.

Είναι φανερό πως οι δύο πρώτες φάσεις είναι ουσιαστικά ο χρονικός διαχωρισμός των δύο παράλληλων εργασιών του PDA. Με την οδήγηση του αισθητηρίου δύο φορές, εστιάζοντας κάθε φορά στην ολοκλήρωση ή την εξαγωγή των δεδομένων, επιτυγχάνεται το ίδιο αποτέλεσμα.

Το άθροισμα των χρόνων που διαρκούν οι δύο αυτές φάσεις, πρέπει να ισούται ή να είναι μικρότερο του χρόνου που προκύπτει από τα καρέ ανά δευτερόλεπτο (fps) που ζητάει ο χρήστης. Αν για παράδειγμα ο χρήστης ζητάει 50 fps τότε πρέπει η ολοκλήρωση και η δειγματοληψία να διαρκούν συνολικά έως και 20ms. Οι πιθανότητες να διαρκούν ακριβώς 20ms είναι ελάχιστες οπότε είναι απαραίτητη μία τρίτη φάση, αυτή της καθυστέρησης κατά την οποία δεν εκτελείται καμία απολύτως ενέργεια και χρησιμοποιείται μόνο για την συμπλήρωση του συνολικού χρόνου.

Στη συνέχεια ακολουθούν οι εξισώσεις μέσω των οποίων υπολογίζεται η χρονική διάρκεια κάθε φάσης, μαζί με τη σημασία κάθε όρου στον πίνακα 4.1.

$$T_{integration} = (intgr - 20) * 10^{-6} \quad (1)$$

$$T_{sampling} = \frac{1}{CLK * 10^3} * 129 \quad (2)$$

$$T_{delay} = \frac{1}{fps} - T_{sampling} - T_{integration} \quad (3)$$

CLK	Συχνότητα σήματος CLK σε Hz από τον χρήστη
fps	Καρέ ανά δευτερόλεπτο από τον χρήστη
intgr	Χρόνος ολοκλήρωσης από τον χρήστη σε μs
$T_{integration}$	Διάρκεια ολοκλήρωσης σε s
$T_{sampling}$	Διάρκεια δειγματοληψίας σε s
T_{delay}	Διάρκεια καθυστέρησης σε s

Πίνακας 4.1 Οι μεταβλητές των εξισώσεων

4.3.1 Μετατροπή του χρόνου σε κύκλους μηχανής

Οι χρόνοι που προκύπτουν με τη χρήση των εξισώσεων πρέπει να μετατραπούν σε κύκλους μηχανής για να είναι χρήσιμοι στον PRU0. Κάθε εντολή του PRU διαρκεί 5ns, ενώ για να δημιουργηθεί μία ρουτίνα χρονικής καθυστέρησης χρειάζονται τουλάχιστον δύο εντολές, οπότε η ελάχιστη χρονική καθυστέρηση που μπορεί να υλοποιηθεί είναι 10ns. Αν για παράδειγμα προέκυπτε χρόνος καθυστέρησης 1287,394μs, σε κύκλους μηχανής γίνεται:

$$Cycles_{delay} = \frac{1287,394}{0.01} = 128.739 \quad (4)$$

Τα 4ns φυσικά χάνονται καθώς η ακρίβεια είναι στα 10ns. Επομένως, πρέπει η ρουτίνα καθυστέρησης να «σπαταλήσει» από τη CPU του PRU0 αυτόν ακριβώς τον αριθμό κύκλων μηχανής. Η μορφή της ρουτίνας που θα υλοποιούσε αυτή τη καθυστέρηση είναι:

```
1.  .macro DelayPhase
2.      LDI R1, 128739          //Φόρτωσε το 128739 στον καταχωρητή R1
3.      SUB R1, R1, 1           //Αφαίρεσε το 1 από το 128739
4.  DelayLoop:
5.      SUB R1, R1, 1           //Αφαίρεσε το 1 από το 129739
6.      QBNE DelayLoop, R1, 0   //Αν ο R1 δεν είναι ίσος με 0, πήδα στο DelayLoop
7.  .endm
```

Η καθυστέρηση πραγματοποιείται από την ετικέτα DelayLoop και ύστερα. Η πρώτη εντολή είναι απαραίτητη για τη φόρτωση του αριθμού επαναλήψεων, εισάγει όμως μία καθυστέρηση των 5ns. Προσθέτοντας μία εντολή αφαίρεσης αμέσως μετά, εισάγονται άλλα 5ns καθυστέρησης αλλά ταυτόχρονα αφαιρούνται 10ns, οπότε ο βρόχος θα διαρκέσει ακριβώς το χρόνο που πρέπει.

Το παραπάνω κομμάτι κώδικα είναι απλά ένα παράδειγμα επίτευξης χρονικής καθυστέρησης με τους PRU. Ο πραγματικός κώδικας δεν θα μπορούσε να έχει κωδικοποιημένο τον αριθμό των κύκλων μηχανής καθώς αυτός προκύπτει από τους συντελεστές που εισάγει ο χρήστης. Επιπλέον, υπάρχουν και άλλες εντολές πριν και μετά την κλήση της ρουτίνας, των οποίων η διάρκεια πρέπει να αφαιρεθεί από το σύνολο των κύκλων ώστε ο τελικός χρόνος της καθυστέρησης να είναι όσο πιο κοντά γίνεται στον ζητούμενο. Αυτή η ακρίβεια μπορεί να επιτευχθεί μόνο με χρήση της γλώσσας Assembly.

Διορθώσεις στους χρόνους που αφορούν τις προδιαγραφές του αισθητηρίου γίνονται από τον κώδικα που είναι γραμμένος σε C, ενώ αυτές που αφορούν κύκλους μηχανής γίνονται από τους PRU σε Assembly.

4.3.2 Υπολογισμός διάρκειας παλμών

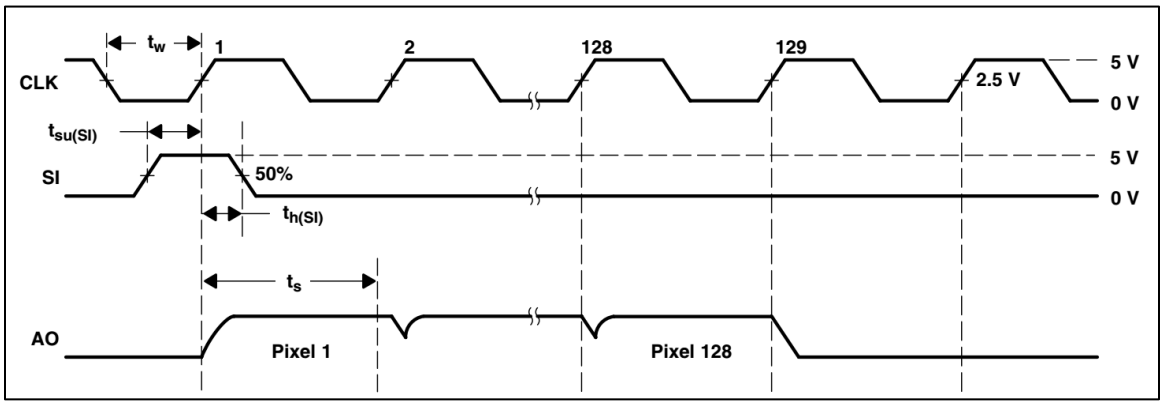
Κατά τη διάρκεια των δύο πρώτων φάσεων ενός κύκλου λήψης καρέ, απαιτείται η δημιουργία παλμών SI και CLK για την οδήγηση του PDA. Αυτοί οι παλμοί δυστυχώς δεν μπορούν να δημιουργηθούν από το υλικό του BeagleBone χωρίς πρόσβαση σε διαδρόμους διασύνδεσης οπότε η δημιουργία των παλμών, προστίθεται στις εργασίες του PRU0.

Σύμφωνα με το φύλλο δεδομένων του PDA, απαιτούνται 129 παλμοί CLK και ένας SI για ένα ολόκληρο καρέ. Αφήνοντας για λίγο τον παλμό SI, οι κύκλοι μηχανής που απαιτούνται για τη δημιουργία ενός παλμού CLK προκύπτουν από τη σχέση 5.

$$Cycles_{pulse\_half} = \frac{T_{integration}}{129 * 2} \quad (5)$$

Ο διπλασιασμός του παρονομαστή προκύπτει από τις δύο καταστάσεις που έχει ένας παλμός.

Στη προσπάθεια να τηρηθεί πλήρως το διάγραμμα χρονισμού του φύλλου δεδομένων του PDA, εισάγεται μία ακόμη δυσκολία. Ο πρώτος παλμός CLK ξεκινάει ενώ ο SI είναι ήδη σε υψηλή κατάσταση. Επιπλέον, ο SI θα αλλάξει σε χαμηλή κατάσταση και θα παραμείνει σε αυτή πριν ο παλμός CLK πάει σε χαμηλή κατάσταση. Στο σχήμα 4.5 φαίνεται ο χρονισμός των σημάτων όπως τον δίνει ο κατασκευαστής στο φύλλο δεδομένων.



Σχήμα 4.5 Το λειτουργικό διάγραμμα όπως δίνεται από την ams στο φύλλο δεδομένων του PDA

Για την υψηλότερη δυνατή ακρίβεια, πραγματοποιήθηκε διαίρεση της διάρκειας του πρώτου παλμού CLK σε τέσσερα κομμάτια αντί για δύο. Με αυτό τον τρόπο, δίνεται η δυνατότητα στον PRU0, να αλλάζει την κατάσταση των ακροδεκτών SI και CLK τη κατάλληλη χρονική στιγμή. Χρησιμοποιώντας τη σχέση 6, υπολογίζεται το πλήθος κύκλων μηχανής που χρειάζονται για χρονική καθυστέρηση ίση με το ένα τέταρτο της διάρκειας ενός παλμού CLK. Το "+1" στον παρονομαστή προκύπτει από το δεύτερο μισό του τελευταίου παλμού CLK που είναι σε χαμηλή κατάσταση.

$$Cycles_{pulse\_quarter} = \frac{T_{integration}}{(129 * 4) + 1} = \frac{T_{integration}}{517} \qquad (6)$$

Αφού υπολογιστούν οι κύκλοι μηχανής για κάθε μία από τις τρεις φάσεις, γράφονται σε συγκεκριμένες θέσεις μνήμης της κοινόχρηστης RAM των PRU. Μαζί γράφονται και πληροφορίες όπως είναι ο αριθμός των pixel και το πλήθος των καρέ που πρέπει να καταγραφούν.

Στον πίνακα 4.2 παρουσιάζονται οι μετρήσεις που έγιναν με τη χρήση λογικού αναλυτή Tektronix TLA 612, ιδιοκτησίας του τμήματος Αυτοματισμού. Η περίοδος δειγματοληψίας του οργάνου μέτρησης ήταν ρυθμισμένη στα 4ms.

$T_{integration}$			**Καρέ ανά δευτερόλεπτο**			
Ζητούμενος χρόνος (μs)	**Μέτρηση (μs)**	**Σφάλμα %**	**Ζητούμενα fps**	**T_{frame}(ms)**	**Μέτρηση T_{frame}(ms)**	**Σφάλμα %**
80.000	80.000	0.000	25	40.000	39.998	0.005
100.000	99.996	0.004	35	28.571	28.570	0.004
150.000	149.996	0.003	50	20.000	19.999	0.004
200.000	199.992	0.004	100	10.000	10.000	0.000
800.000	799.960	0.005	250	4.000	4.000	0.009
1600.000	1599.910	0.006	300	3.333	3.334	0.012
3500.000	3499.790	0.006	500	2.000	2.000	0.024

Πίνακας 4.2 Αποτελέσματα μετρήσεων που έγιναν με χρήση λογικού αναλυτή

Τα αποτελέσματα των μετρήσεων είναι ιδιαίτερα ενθαρρυντικά καθώς στη χειρότερη περίπτωση, το ποσοστιαίο σφάλμα κυμαίνεται στο δεύτερο δεκαδικό ψηφίο.

4.4 Αρχιτεκτονική κώδικα

Ο κώδικας αποτελείται από πολλές μικρές βοηθητικές ρουτίνες, στις οποίες χωρίζονται οι λειτουργίες του προγράμματος με σκοπό την ευκολότερη κατανόηση του. Αυτές οι ρουτίνες είναι μοιρασμένες σε διαφορετικά source αρχεία σύμφωνα με την λειτουργικότητα τους, ενώ κάθε ένα από αυτά τα αρχεία, συνοδεύεται από το αντίστοιχο header και περιέχει τις δηλώσεις των ρουτινών, των global μεταβλητών και των macro. Ο πίνακας 4.3 περιέχει τις ονομασίες των source file και την περιγραφή τους.

pda_drivers.c	Περιέχει τη ρουτίνα main() καθώς και όλες τις βοηθητικές που δεν μπορούσαν να μπουν σε κάποιο από τα άλλα αρχεία.
gpio.c	Περιέχει τις ρουτίνες που χειρίζονται τους ακροδέκτες του πλήκτρου και του τρίχρωμου LED.
sample_file.c	Περιέχει τις ρουτίνες που δημιουργούν και επεξεργάζονται το αρχείο με τα δεδομένα του δείγματος.
debug.c	Περιέχει τη ρουτίνα εκτύπωσης για την αποσφαλμάτωση του προγράμματος.

Πίνακας 4.3 Τα source αρχεία του προγράμματος

Η ρουτίνα Debug_Print() που βρίσκεται στο αρχείο debug.c, χρησιμοποιήθηκε για την αποσφαλμάτωση του κώδικα κατά την ανάπτυξή του. Για την άμεση απενεργοποίηση της εκτύπωσης δεδομένων στη κονσόλα, αρκεί η αλλαγή της δήλωσης "#define DEBUG_MODE 1" στο αρχείο debug.h, σε "#define DEBUG_MODE 0". Κατά τη διαδικασία του build του κώδικα, ο preprocessor του compiler θα αφήσει εκτός τις εντολές που βρίσκονται ανάμεσα στις οδηγίες για αποσφαλμάτωση:

```
#if DEBUG_MODE == 1
        /* Κώδικας που εκτελείται μόνο κατά τη διαδικασία αποσφαλμάτωσης. */
#endif
```

Με αυτό τον τρόπο επιτυγχάνεται γρήγορη ενσωμάτωση μηχανισμού αποσφαλμάτωσης χωρίς ιδιαίτερη παρέμβαση στον κώδικα σε περίπτωση που παρατηρηθεί λανθασμένη λειτουργία.

Κάθε ρουτίνα που καλείται και έχει πιθανότητα αποτυχίας, όπως για παράδειγμα είναι η δυναμική δέσμευση μνήμης ή το άνοιγμα και το κλείσιμο αρχείων, ελέγχεται στο τέλος της εκτέλεσης της και σε περίπτωση που έχει αποτύχει τερματίζει η εκτέλεση του προγράμματος.

Με τον τερματισμό του προγράμματος, επιστρέφεται ένας αριθμός τύπου unsigned int. Αν επιστρέψει τιμή στο εύρος των αριθμών 1 – 99.999, τότε πρόκειται για τον αριθμό του αρχείου στο οποίο αποθηκεύτηκε το δείγμα. Σε περίπτωση όπου ο αριθμός που επιστρέφεται είναι από 100.000 και πάνω, πρόκειται για κάποιο από τα αριθμημένα σφάλματα. Η συγκεκριμένη υλοποίηση παρέχει στον χρήστη τη δυνατότητα της ενσωμάτωσης της κλήσης του οδηγού στο δικό του πρόγραμμα και τον χειρισμό των δεδομένων χωρίς παύσεις για χειροκίνητη φόρτωση.

5 Έλεγχος λειτουργίας

Σε αυτό το κεφάλαιο παρουσιάζονται τα πειράματα μέσω των οποίων επαληθεύτηκε η λειτουργικότητα του συστήματος. Κάθε υποσύστημα του BeagleBone που χρειάστηκε να χρησιμοποιηθεί, δοκιμάστηκε και εξακριβώθηκε η αρχή λειτουργίας του πριν ενσωματωθεί στον οδηγό. Ακολουθώντας αυτή τη λογική κατά την ανάπτυξη, ήταν εύκολο να εντοπιστεί η πηγή των σφαλμάτων που προέκυπταν σε σύντομο χρονικό διάστημα.

5.1 Χρήση των PRU

Κατά την εκκίνηση του BeagleBone, οι PRU είναι ανενεργοί. Για την ενεργοποίηση και τη χρήση τους, μελετήθηκε ο οδηγός του API που διατίθεται από την T.I [11] και στη συνέχεια έγινε συγγραφή ενός δοκιμαστικού κώδικα C και ενός δοκιμαστικού κώδικα Assembly. Ο κώδικας C χρησιμοποιούσε το API των PRU για να εκκινήσει και να φορτώσει τον δοκιμαστικό κώδικα Assembly στον PRU0. Η μοναδική εργασία που εκτελούσε ο κώδικας Assembly, ήταν η δημιουργία ενός παλμού με σταθερή περίοδο ενός δευτερολέπτου στον ακροδέκτη που αργότερα χρησιμοποιήθηκε για την παραγωγή του παλμού SI. Χρησιμοποιώντας ένα LED σε σειρά με μία αντίσταση, ήταν εύκολο να εξακριβωθεί η λειτουργία του PRU0.

Σειρά είχε η επικοινωνία του PRU0 με τον ARM μέσω της μνήμης. Έγιναν δοκιμές με ανταλλαγές δεδομένων που είτε χρησιμοποιούνταν από τον PRU0 για την αλλαγή της περιόδου του δοκιμαστικού παλμού, είτε τυπώνονταν από τον ARM στη κονσόλα των Linux. Δόθηκε σημασία στη στοίχιση των δεδομένων στη μνήμη καθώς και στον τρόπο χρήσης των καταχωρητών στο χαμηλό επίπεδο στις περιπτώσεις όπου τα δεδομένα ήταν σε διαφορετικό μήκος από αυτό των 32bit.

Ο έλεγχος του ADC από τον PRU δοκιμάστηκε χρησιμοποιώντας ένα ποτενσιόμετρο αλλά και της τεχνικής ανταλλαγής δεδομένων μέσω τις μνήμης. Ο PRU0

διάβαζε συνεχώς την τιμή του ποτενσιόμετρου και την έγγραφη στη κοινόχρηστη RAM του. Ο ARM διάβαζε την ίδια θέση μνήμης δύο φορές το δευτερόλεπτο και τύπωνε την τιμή στη κονσόλα. Περιστρέφοντας τον άξονα του ποτενσιόμετρου, άλλαζε ανάλογα και η τιμή που εμφανιζόταν στη κονσόλα.

Η προσθήκη του INTC αλλά και του δεύτερου PRU, έγινε σε μεταγενέστερο χρόνο όπου υπήρχε πλέον εξοικείωση με το BeagleBone και δεν χρειάστηκε κάποιου είδους πείραμα.

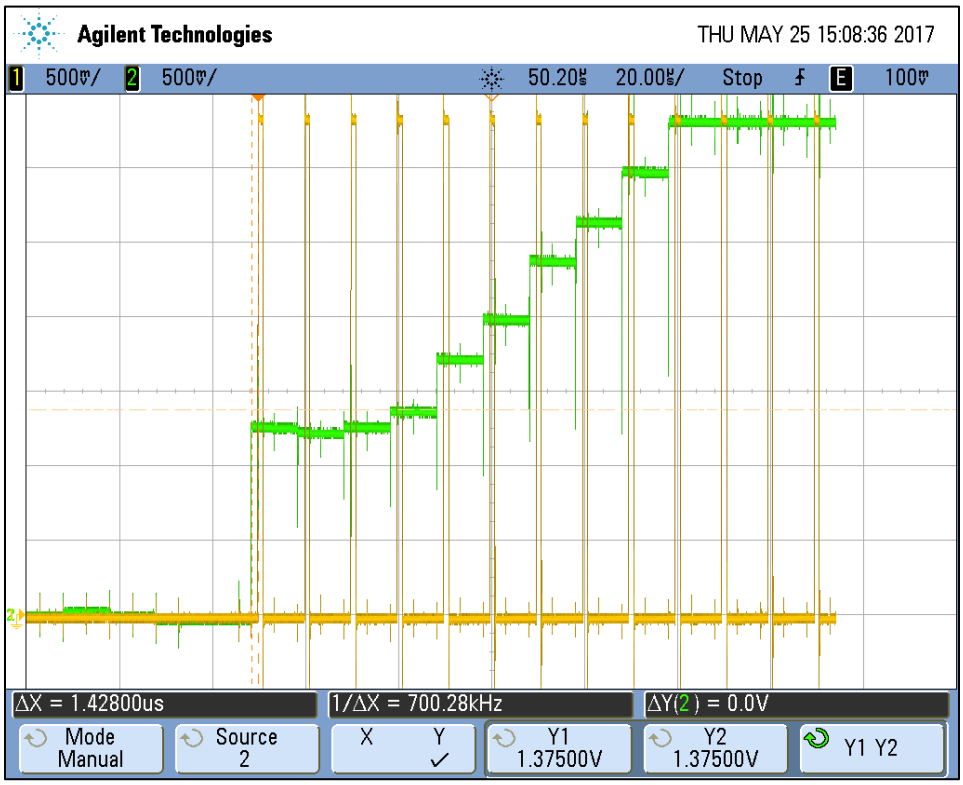
Για τον έλεγχο των παλμών χρησιμοποιήθηκε ένας λογικός αναλυτής χαμηλού κόστους που είναι συμβατός με το λογισμικό της Saleae. Στο σχήμα 5.1 παρουσιάζεται ένα στιγμιότυπο από τις πρώτες συλλήψεις που έγιναν κατά τα πρώτα στάδια της ανάπτυξης του συστήματος.

Σχήμα 5.1 Επαλήθευση των παλμών με χρήση λογικού αναλυτή

5.2 Οδήγηση του PDA

Η περίπλοκη αρχή λειτουργίας του αισθητηρίου που περιεγράφηκε στα δύο προηγούμενα κεφάλαια δημιούργησε αρκετές φορές την ανάγκη για χρήση εξοπλισμού όπως είναι ο παλμογράφος και ο λογικός αναλυτής. Στο σχήμα 5.2 φαίνεται το αποτέλεσμα της πρώτης δοκιμής με το PDA όπως καταγράφηκε από ψηφιακό παλμογράφο. Ο ακροδέκτης του παλμογράφου συνδέθηκε με την αναλογική έξοδο του PDA με σκοπό την καταγραφή των φορτίων που είχαν καταφέρει να συσσωρεύσουν οι φωτοδίοδοι. Τοποθετήθηκε ένα μικρό κομμάτι πλαστικό στις πρώτες φωτοδιόδους του PDA με σκοπό να προκαλέσει σκίαση, ενώ ο φωτισμός στον χώρο ήταν αρκετά έντονος. Αυτό που

παρατηρήθηκε ήταν, πως στα πρώτα δεδομένα που εξήγαγε το PDA, οι τάσεις ήταν χαμηλές και σταδιακά αυξάνονταν. Αυτό σήμαινε πως ο οδηγός παρήγαγε σωστά τους παλμούς και το αισθητήριο αποκρινόταν.

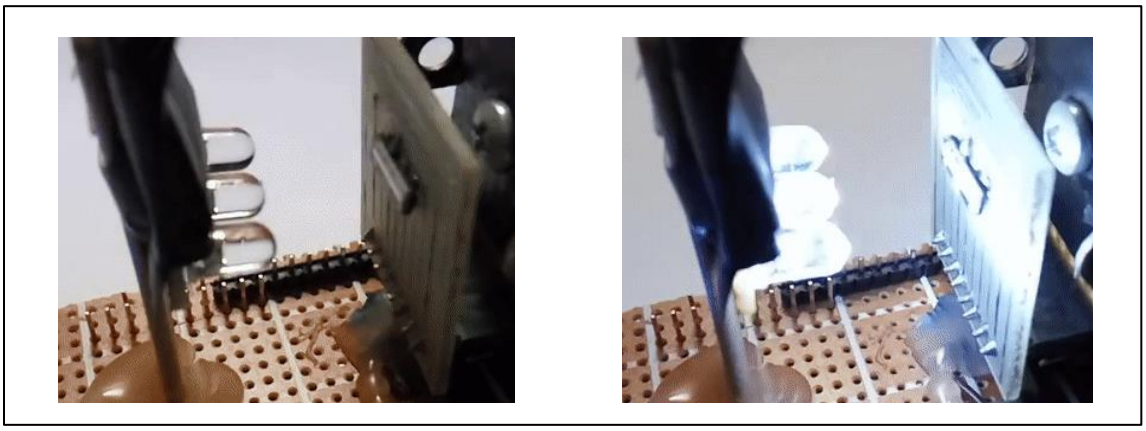


Σχήμα 5.2 Η έξοδος του PDA κατά τη διαδικασία του πειράματος.

Με την ανάπτυξη και τον εμπλουτισμό του κώδικα, έπρεπε να γίνεται συνεχώς έλεγχος της λειτουργίας του συστήματος. Το επόμενο πείραμα είχε ως σκοπό τον έλεγχο των ακριανών pixel του PDA, καθώς το σύστημα λαμβάνει συνεχόμενα καρέ. Ο λόγος που το συγκεκριμένο πείραμα κρίθηκε απαραίτητο, είναι η περίπλοκη φύση του οδηγού όσον αφορά τους χρονισμούς των παλμών.

Δημιουργήθηκε μία δοκιμαστική πλακέτα με τρία λευκά LED τοποθετημένα σε απόσταση περίπου ενός εκατοστού από το PDA. Τα LED οδηγούνταν από ένα Arduino το οποίο είχε συνδεδεμένη μία είσοδο του στην έξοδο SI του Beaglebone. Σε κάθε δεύτερο

παλμό SI το Arduino άλλαζε κατάσταση στα LED με σκοπό την λήψη διαδοχικών καρέ με αντίθετες συνθήκες φωτός όπως φαίνεται στο σχήμα 5.3. Αν το πρώτο καρέ ήταν σκοτεινό, το επόμενο θα ήταν στον κορεσμό, οπότε τα δεδομένα που θα γράφονταν στο αρχείο .txt, θα έπρεπε να αλλάζουν ανά 128 σε ακραίες τιμές. Οι τιμές που θα αντιστοιχούσαν στα σκοτεινά καρέ θα έπρεπε να είναι πολύ κοντά στο 0, ενώ αυτές των φωτεινών θα έπρεπε να είναι κοντά στο 4095.



Σχήμα 5.3 Τα LED τοποθετημένα μπροστά από το PDA, στις δύο διαφορετικές καταστάσεις

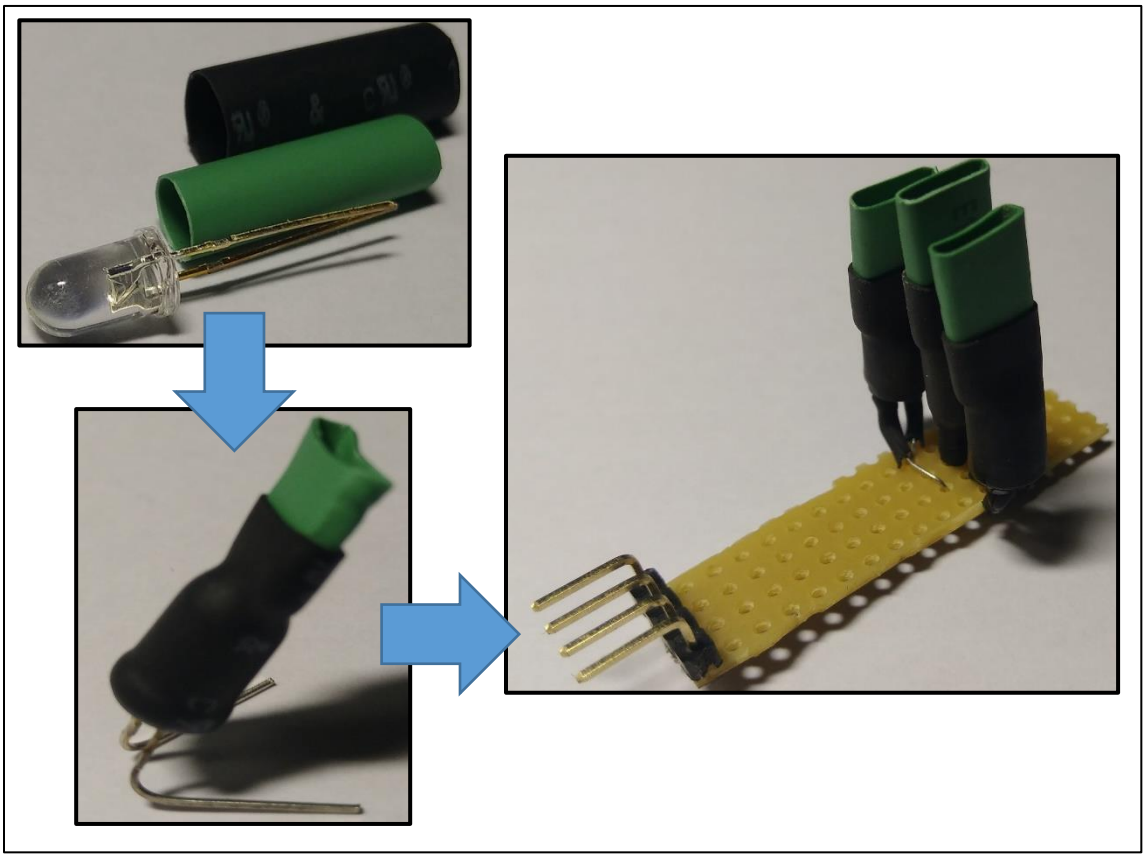
Το αποτέλεσμα του πειράματος αρχικά επέστρεψε ανησυχητικά δεδομένα. Στα φωτεινά καρέ, οι τρεις πρώτες τιμές ήταν πολύ μικρότερες του 4095, ενώ στα σκοτεινά συνέβαινε το αντίθετο, με τις πρώτες τιμές να είναι πολύ μεγαλύτερες του 0.

Το πρόβλημα εντοπίστηκε στο ρυθμό μεταβολής του τελεστικού ενισχυτή που χρησιμοποιούνταν εκείνη την περίοδο, τον MCP6002. Με μόλις 0.6V/μs, ήταν πολύ αργός σε σχέση με την ταχύτητα που μεταβάλλεται η έξοδος του PDA, με αποτέλεσμα την ανάγνωση λανθασμένων δεδομένων, τα οποία υπό άλλες συνθήκες φαινόντουσαν φυσιολογικά.

Με την αλλαγή του τελεστικού ενισχυτή στον LM6142, ο ρυθμός μεταβολής αυξήθηκε στα 25V/μs και το πρόβλημα λύθηκε. Επιπλέον, προστέθηκε μία ρουτίνα σύντομης χρονικής καθυστέρησης στον PRU1, πριν τον σκανδαλισμό του ADC ώστε να

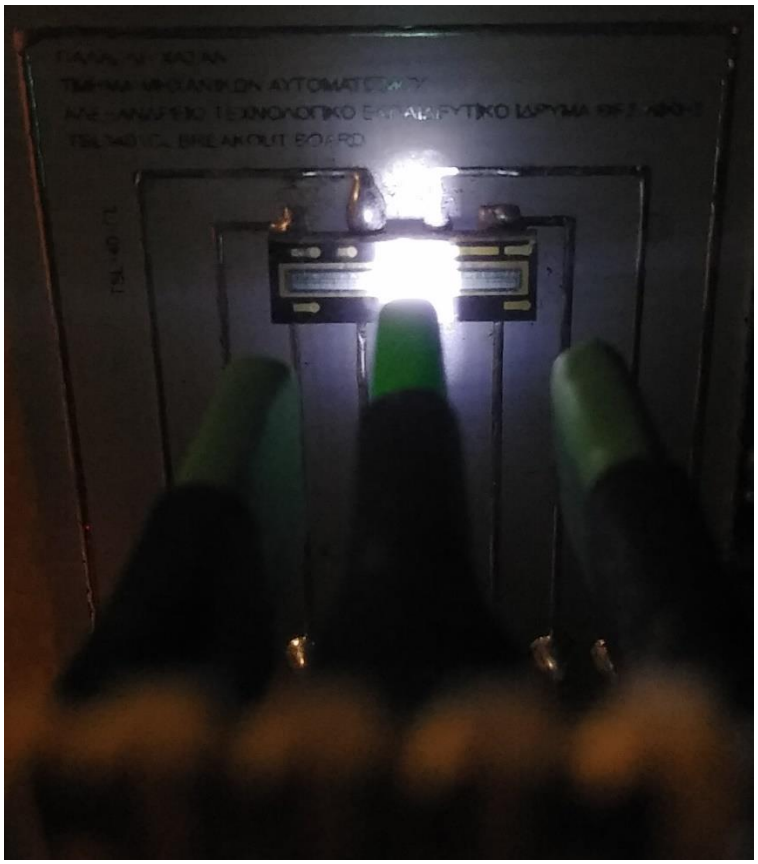
δίνεται στην έξοδο του τελεστικού ενισχυτή χρόνος για την σταθεροποίηση της τάσης του. Η χρονική καθυστέρηση είναι στα 150ns, τιμή αρκετά μικρότερη των 10μs που διαρκεί ένας παλμός CLK στα 100KHz.

Για λόγους επίδειξης σχεδιάστηκε και πραγματοποιήθηκε ένα ακόμη πείραμα χρησιμοποιώντας το Arduino και τα LED. Στόχος του πειράματος είναι η λήψη διαδοχικών καρέ καθώς ανάβει πάντα μόνο το ένα από τα τρία LED. Κάθε ένα από αυτά, στοχεύει ένα συγκεκριμένο τμήμα του PDA, ενώ έχουν καλυφθεί με θερμοσυστελλόμενο μονωτικό υλικό ώστε να περιορίζεται η κατεύθυνση του φωτός ευθεία μπροστά τους όπως φαίνεται στο σχήμα 5.4. Ξεκινώντας με το αριστερό LED αναμμένο, με κάθε δεύτερο παλμό SI το Arduino ολισθαίνει το ενεργό LED μία θέση δεξιά.



Σχήμα 5.4 Τροποποίηση των LED με σκοπό την εστίαση του φωτός σε συγκεκριμένα σημεία

Στο στιγμιότυπο που παρουσιάζεται στο σχήμα 5.5, φαίνεται ο τρόπος που είναι τοποθετημένα τα LED καθώς και η περιοχή του PDA που φωτίζεται έχοντας μόνο το μεσαίο LED ενεργό.

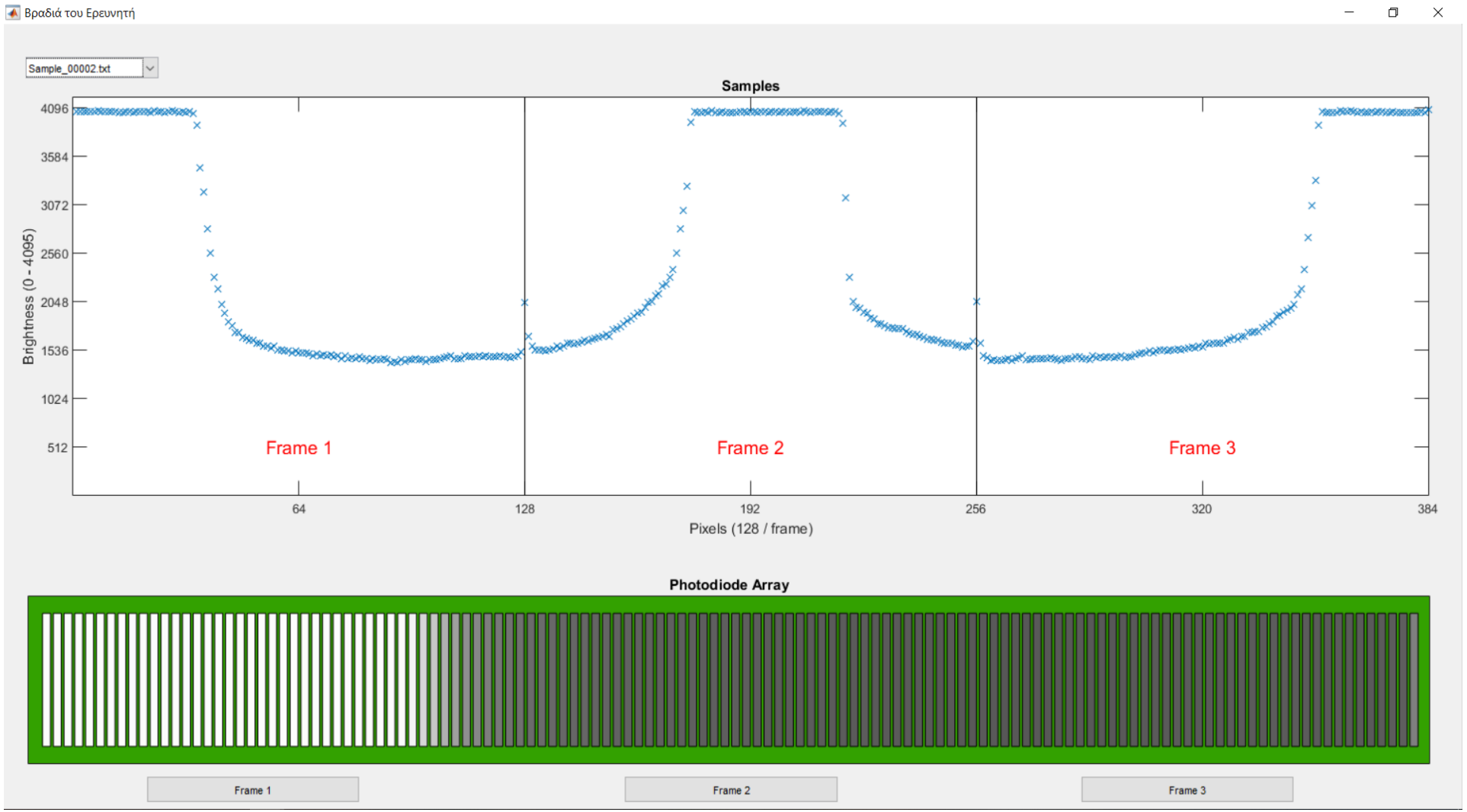


Σχήμα 5.5 Στιγμιότυπο του πειράματος καθώς είναι μόνο το κεντρικό LED ενεργό.

Απεικονίζοντας γραφικά τα δείγματα, φαίνεται ξεκάθαρα πως οι φωτοδίοδοι με το ενεργό LED έχουν τιμές πολύ κοντά στον κορεσμό ενώ οι φωτοδίοδοι που ήταν στα σκοτεινά σημεία επέστρεψαν τιμές στα επίπεδα φωτεινότητας του περιβάλλοντος. Για γρήγορο έλεγχο των δειγμάτων, δημιουργήθηκε ένα γραφικό περιβάλλον διεπαφής χρήστη (graphical user interface, GUI) στο Matlab, το οποίο ανοίγει το αρχείο του δείγματος και δημιουργεί μία δισδιάστατη γραφική παράσταση των τριών πρώτων καρέ. Επιπλέον, παρουσιάζει γραφικά την ένταση της φωτεινότητας που δέχτηκε κάθε φωτοδίοδος

ξεχωριστά, χρησιμοποιώντας την τιμή της στο χρωματικό μοντέλο RGB. Στο σχήμα 5.6 αποτυπώνεται ένα στιγμιότυπου του GUI καθώς έχει φορτώσει ένα από τα αρχεία με δείγματα που λήφθηκαν κατά την διεξαγωγή του δεύτερου πειράματος με τα LED.

Οι διαδικασίες που παρουσιάστηκαν σε αυτό το κεφάλαιο ήταν αρκετές για να εξακριβωθεί η ομαλή λειτουργία του οδηγού ώστε να παραδοθεί για χρήση σε μετρήσεις από μέλος της ερευνητικής ομάδας που έχει αναλάβει την εργασία της ανάλυσης των δεδομένων της έρευνας που αφορά τον καθορισμό του μεγέθους των κυττάρων του αίματος με μη παρεμβατικό τρόπο. Τα δεδομένα που εξάγονται είναι ενθαρρυντικά και συνεχώς επαληθεύουν τη σωστή λειτουργία του οδηγού καθώς κατά την γραφική τους απεικόνιση, εμφανίζονται αναμενόμενα φαινόμενα όπως οι καρδιακοί παλμοί και το κλείσιμο της αορτικής βαλβίδας [12].



Σχήμα 5.6 Το GUI που δημιουργήθηκε για τη γραφική παράσταση των δειγμάτων

6 Εγκατάσταση και ρύθμιση του Debian

Το κεφάλαιο που παρουσιάζεται στις επόμενες σελίδες έχει τον ρόλο οδηγού για την εγκατάσταση του Debian OS στο BeagleBone, καθώς και την εφαρμογή των απαραίτητων ρυθμίσεων για την ομαλή λειτουργία του οδηγού. Στους αναγνώστες που επιθυμούν να χρησιμοποιήσουν τον οδηγό, προτείνεται η ανάγνωση ολόκληρου του κεφαλαίου πριν επιχειρηθεί κάποια ενέργεια. Τα περισσότερα βήματα που παρουσιάζονται παρακάτω, απαιτούν πρόσβαση root στο OS, για το λόγο αυτό, είναι απαραίτητο ο χρήστης να προβεί προσεκτικά στις επόμενες κινήσεις του.

Επίσης είναι σημαντικό να μην διακοπεί η λειτουργία της πλακέτας κατά την εκτέλεση των βημάτων που ακολουθούν. Ο σωστός τρόπος απενεργοποίησης του συστήματος είναι μέσω της εντολής “sudo shutdown -h now”, ενώ για την επανεκκίνηση χρησιμοποιείται η εντολή “sudo reboot”. Η απότομη απώλεια τάσης μπορεί να προκαλέσει καταστροφή αρχείων του OS ή απροσδιόριστη συμπεριφορά.

6.1 Λειτουργικό σύστημα Debian

Η περισσότερο διαδεδομένη διανομή Linux για το BeagleBone είναι η Debian [13]. Πρόκειται για έναν από τους παλαιότερους πυρήνες Linux, γνωστός για την φιλοσοφία του ανοιχτού κώδικα και του δωρεάν λογισμικού.

Η παρούσα εργασία βασίζεται στην έκδοση “Debian 8.7 2017-03-19 4GB SD IOT”, η οποία διανέμεται δωρεάν στην επίσημη σελίδα του BBB [14]. Μεταγενέστερες εκδόσεις από αυτήν που παρατίθεται παραπάνω, είναι πιθανό να έχουν σοβαρές τροποποιήσεις στη Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface, API) των PRU με αποτέλεσμα την απώλεια της συμβατότητας του οδηγού. Προτείνεται η χρήση της συγκεκριμένης έκδοσης για την διατήρηση της συμβατότητας.

6.1.1 Εγκατάσταση του OS

Μετά τα λήψη της εικόνας του OS, ο χρήστης πρέπει να την γράψει σε SD κάρτα, μεγέθους τουλάχιστον 4GB. Προτείνεται το εργαλείο "Win32 Disk Imager" [15] για την εγγραφή της εικόνας, καθώς είναι δωρεάν και εύκολο στη χρήση. Εφόσον έχει εισαχθεί η κάρτα μνήμης στον Η/Υ, ο χρήστης πρέπει να επιλέγει το αρχείο προς εγγραφή και μετά να πατήσει το πλήκτρο "Write". Η διαδικασία ολοκληρώνεται μετά από λίγα λεπτά και η κάρτα είναι έτοιμη για τοποθέτηση στο BeagleBone. Σε περίπτωση διακοπής της εγγραφής, η διαδικασία πρέπει να επαναληφθεί από την αρχή.

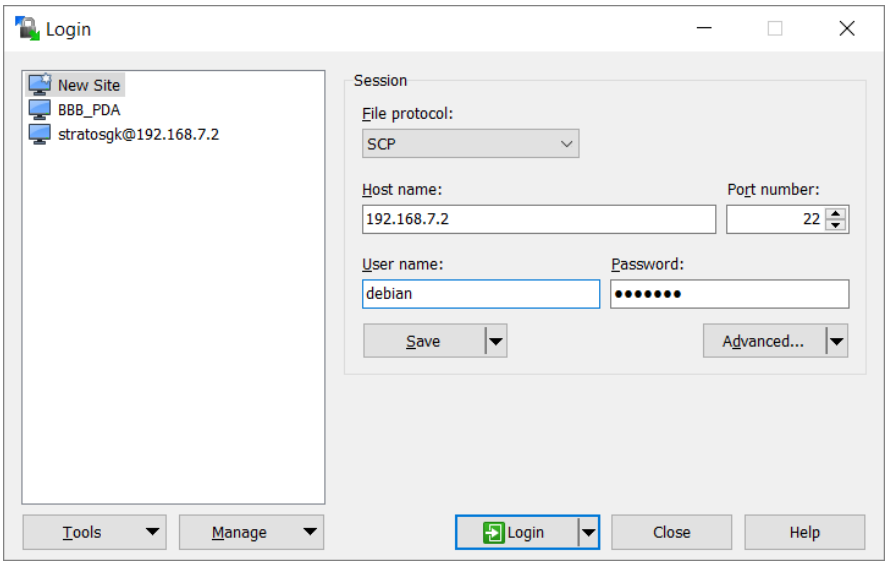
Εφόσον η εγγραφή ολοκληρώθηκε και η κάρτα μεταφέρθηκε στο BeagleBone, πρέπει να συνδεθεί μέσω καλωδίου USB με τον Η/Υ και στη συνέχεια να ενεργοποιηθεί. Ο χρήστης δεν πρέπει να ξεχνάει ότι το BeagleBone είναι ουσιαστικά ένας Η/Υ και απαιτούνται μερικά δευτερόλεπτα μέχρι αυτό να είναι έτοιμο για χρήση. Μία ασφαλής τακτική είναι η αναμονή μέχρι ο Η/Υ να ανοίξει παράθυρο περιήγησης του BeagleBone, καθώς όταν αυτό είναι πλέον έτοιμο για χρήση, τα Windows το εντοπίζουν σαν μέσο αποθήκευσης.

6.1.2 Σύνδεση με το BeagleBone

Για να εδραιωθεί σύνδεση μεταξύ Η/Υ και BeagleBone ώστε να ελέγχεται το δεύτερο μέσω εντολών, απαιτείται ή χρήση του πρωτοκόλλου "Secure Shell" (SSH). Υπάρχουν διάφορα προγράμματα που υποστηρίζουν το συγκεκριμένο πρωτόκολλο, αυτό που προτείνεται όμως είναι το "PuTTY" [16]. Ακόμη, είναι απαραίτητη και η χρήση κάποιου προγράμματος FTP client για την μεταφορά αρχείων και για τις δύο κατευθύνσεις. Το πρόγραμμα που προτείνεται για τη συγκεκριμένη εργασία, είναι το "WinSCP" [17], το οποίο είναι δωρεάν και περιέχει το PuTTY στην εγκατάσταση καθώς συχνά χρειάζονται παράλληλα αυτά τα δύο προγράμματα.

Κατά την εκκίνηση του WinSCP εμφανίζεται το παράθυρο εισόδου, το οποίο πρέπει να συμπληρωθεί όπως φαίνεται στο σχήμα 6.1. Στο πεδίο του κωδικού πρέπει να

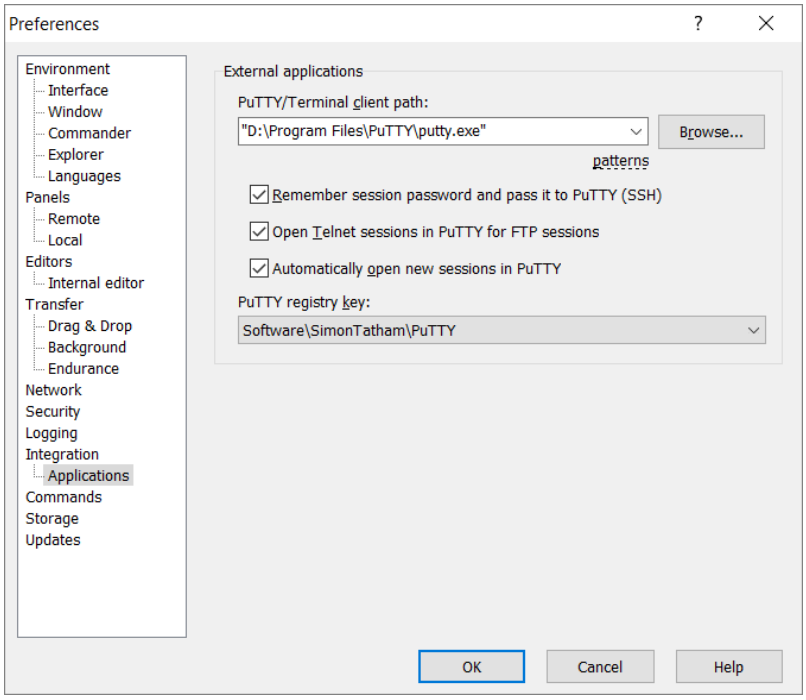
εισαχθεί το "*temppwd*", όπου είναι και ο προεπιλεγμένος κωδικός του χρήστη debian. Αφού συμπληρωθούν σωστά τα πεδία και αποθηκευτούν οι αλλαγές, ο χρήστης πρέπει να πατήσει το πλήκτρο "login" και να περιμένει μέχρι να εμφανιστεί ένα παράθυρο. Επιλέγοντας "Add" στο παράθυρο, εδραιώνεται η σύνδεση μεταξύ Η/Υ και BeagleBone.



Σχήμα 6.1 Παράθυρο εισόδου του WinSCP

Εάν το PuTTY δεν ξεκινήσει αυτόματα με την σύνδεση, μπορεί να ρυθμιστεί από το μενού "Options" επιλέγοντας "Preferences", στη συνέχεια "Integration" και τέλος "Applications". Πρέπει να εισαχθεί το πλήρες path μέχρι το putty.exe και να συμπληρωθούν τα κουτιά όπως φαίνεται στο σχήμα 6.2. Κάνοντας επανεκκίνηση του WinSCP, θα πρέπει πλέον να λειτουργεί η αυτόματη ενεργοποίηση του PuTTY.

Πρέπει να σημειωθεί πως υπάρχει η πιθανότητα να μπλοκάρει το τείχος προστασίας του υπολογιστή το PuTTY ή το WinSCP με αποτέλεσμα να μην γίνεται σύνδεση. Σε αυτή τη περίπτωση, ο χρήστης πρέπει να προσθέσει τα δύο αυτά προγράμματα στις εξαιρέσεις του τείχους προστασίας.



Σχήμα 6.2 Ρύθμιση του PuTTY ως προεπιλεγμένη εφαρμογή telnet

6.2 Τροποποίηση του OS

Εφόσον τα βήματα των προηγούμενων παραγράφων έχουν ολοκληρωθεί, ο χρήστης είναι τώρα σε θέση να ξεκινήσει την τροποποίηση του OS που βρίσκεται εγκατεστημένο στην SD κάρτα. Για την εκτέλεση συγκεκριμένων εντολών, είναι απαραίτητη η είσοδος με δικαιώματα root, οπότε η πρώτη εντολή που πρέπει να δοθεί είναι η “sudo su”. Με την εκτέλεση της εντολής, ζητείται από τον χρήστη να εισάγει ξανά τον κωδικό που είχε δοθεί στο login παράθυρο, δηλαδή τον *temppwd*.

Κατά την διάρκεια της συνεδρίας με δικαιώματα root, ο χρήστης πρέπει να είναι ιδιαίτερα προσεκτικός, καθώς μπορεί να προκαλέσει μόνιμη ζημιά στο OS, με μοναδική λύση την επανεγκατάστασή του. Ενώ υπάρχει τρόπος ώστε ο χρήστης να μην έχει συνεχώς δικαιώματα root, δεν παρουσιάζεται στην συγκεκριμένη εργασία, καθώς ξεφεύγει από τον σκοπό της.

6.2.1 Ρύθμιση του systemd-journald

Το πρώτο αρχείο που πρέπει να τροποποιηθεί είναι το jourland.conf [18]. Είναι κομμάτι της υπηρεσιας system-journald του συστήματος, το οποίο ουσιαστικά είναι μηχανισμός καταγραφής δεδομένων για τα διάφορα υποσυστήματα του OS [19].

Κατά την ανάπτυξη του οδηγού και της συσκευής γενικότερα, προέκυψε ένα σφάλμα στο σύστημα το οποίο απέτρεπε την μεταφορά αρχείων προς το BeagleBone. Μετά από μία σύντομη αναζήτηση, έγινε αντιληπτό πως έπρεπε να αδειάσει το αρχείο καταγραφής που διατηρεί η υπηρεσία system-journald, καθώς είχε γεμίσει και το OS απαγόρευε την λήψη αρχείων επειδή δεν μπορούσε πλέον να τις καταγράψει.

Για την τροποποίηση των ρυθμίσεων πρέπει να ανοίξει το αρχείο journald.conf με κάποιον επεξεργαστή κειμένου. Ο ευκολότερος τρόπος είναι να χρησιμοποιηθεί το “vi” [20], απλά πληκτρολογώντας την εντολή “vi /etc/systemd/journald.conf” στην κονσόλα.

Με το πάτημα του πλήκτρου “i” ενεργοποιείται η δυνατότητα τροποποίησης του αρχείου. Μετακινώντας τον κέρσορα στο κατάλληλο σημείο κάθε γραμμής τη φορά, πρέπει οι τρεις γραμμές που παρουσιάζονται παρακάτω να έχουν την ίδια ακριβώς κατάσταση. Αυτές οι επιλογές αφορούν το μέγιστο μέγεθος του αρχείου καταγραφής, καθώς και το επίπεδο των γεγονότων που θα καταγράφονται. Για αποθήκευση και έξοδο από το vi, πρέπει να πατηθεί το πλήκτρο escape και να δοθεί η εντολή “:wq!”. Πλέον το σύστημα έχει στη διάθεση του περισσότερο διαθέσιμο χώρο, ενώ θα κρατάει και λιγότερα γεγονότα.

- #SystemMaxUse=16M
- #MaxLevelStore=warning
- #MaxLevelSyslog=warning

6.2.2 Απενεργοποίηση του Universal Cape

Για τον έλεγχο του GPIO μέσω κώδικα υπάρχουν διάφορες επιλογές. Όπως όμως αναφέρθηκε προηγουμένως, έχει συγγραφεί DTO για την συγκεκριμένη εφαρμογή, το οποίο ενημερώνει το OS για τους ακροδέκτες του GPIO που χρησιμοποιούνται. Για λόγους ευκολίας προς τους αρχάριους, το BeagleBone έχει ένα προεπιλεγμένο DTO που ονομάζεται "Universal Cape" και παρέχει πρόσβαση σε όλους τους ακροδέκτες που είναι ελεύθεροι προς χρήση. Ο κατασκευαστής το έχει ρυθμίσει να φορτώνεται αυτόματα κατά την εκκίνηση του συστήματος, απαγορεύοντας έτσι την λειτουργία του DTO της πτυχιακής εργασίας που παρουσιάστηκε στο προηγούμενο κεφάλαιο. Στις παραγράφους που ακολουθούν, εξηγείται η διαδικασία απενεργοποίησης του Universal Cape.

Αρχικά, εκτελώντας την εντολή "cat /sys/devices/platform/bone_capemgr/slots", εμφανίζεται η λίστα με τα ενεργά "capes" που έχει το OS εκείνη τη στιγμή. Στο σχήμα 6.3 φαίνεται το αποτέλεσμα της εντολής, με την τελευταία γραμμή να αφορά το Universal Cape.

```
root@beaglebone:/home/debian# cat /sys/devices/platform/bone_capemgr/slots
 0: PF----  -1
 1: PF----  -1
 2: PF----  -1
 3: PF----  -1
 4: P-O-L-   0 Override Board Name,00A0,Override Manuf,univ-all
```

Σχήμα 6.3 Η λίστα με τα ενεργά capes

Οι PRU είναι σχεδιασμένοι να έχουν πρόσβαση σε συγκεκριμένους ακροδέκτες του GPIO. Κάποιοι από αυτούς τους ακροδέκτες χρησιμοποιούνται για την μετάδοση ήχου και εικόνας από την έξοδο HDMI ενώ κάποιοι άλλοι συνδέονται με το υποσύστημα της μνήμης eMMC. Για τους σκοπούς της εργασίας δεν απαιτείται η χρήση των HDMI και eMMC, οπότε με την κατάργηση του Universal Cape αποδεσμεύονται οι ακροδέκτες και απενεργοποιούνται τα εν λόγω υποσυστήματα.

Με χρήση της εντολής “vi /boot/uEnv.txt”, ανοίγει στο περιβάλλον vi το αρχείο που περιέχει την πληροφορία για το Universal Cape. Πατώντας για ακόμη μία φορά το πλήκτρο “i” ο επεξεργαστής κειμένου μπαίνει σε λειτουργία τροποποίησης.

Το HDMI/eMMC cape απενεργοποιείται διαγράφοντας τον χαρακτήρα “#”, από την αρχή της γραμμής “#dtb=am335x-boneblack-overlay.dtb”. Το αποτέλεσμα πρέπει να μοιάζει με αυτό του σχήματος 6.4.

```
##BeagleBone Black: HDMI (Audio/Video)/eMMC disabled:
dtb=am335x-boneblack-overlay.dtb
```

Σχήμα 6.4 Απενεργοποίηση του HDMI(AudioVideo/eMMC Cape

Μερικές γραμμές αργότερα, η οδηγία “cmline-coherent_pool=1M net.ifname=0 quiet cape_universal=enable” πρέπει να τροποποιηθεί εισάγοντας τον χαρακτήρα “#” has ανάμεσα στις λέξεις “quiet” και “cape_universal”. Ιδανικά, το “#cape_universal=enable” πρέπει να τοποθετηθεί στην επόμενη γραμμή όπως φαίνεται στο σχήμα 6.5. Με το πάτημα του πλήκτρου escape και στη συνέχεια εισάγοντας την εντολή “:wq!”, αποθηκεύονται οι αλλαγές και κλείνει ο επεξεργαστής κειμένου. Απαιτείται επανεκκίνηση του συστήματος για την εφαρμογή των αλλαγών.

```
cmdline=coherent_pool=1M net.ifnames=0 quiet
#cape_universal=enable
```

Σχήμα 6.5 Απενεργοποίηση του Universal Cape

Εκτελώντας ξανά την εντολή “cat /sys/devices/platform/bone_capemgr/slots” πρέπει να εμφανίζεται κενή η λίστα με τα capes.

6.2.3 Αλλαγή του Kernel

Ως τελευταίο βήμα, μένει η αλλαγή του kernel του OS, σε κάποιον περισσότερο κατάλληλο, καθώς αυτός που συνοδεύει την διανομή Debian που χρησιμοποιείται, δεν επιτρέπει πρόσβαση στους PRU σε χαμηλό επίπεδο. Τόσο ο αρχικός kernel, όσο και ο αντικαταστάτης του, βασίζονται σε αυτόν των Linux.

Η διανομή Debian που έχει φορτωθεί στην κάρτα SD, χρησιμοποιεί τον kernel 4.54-ti-r93 της T.I. Το χαρακτηριστικό που τον καθιστά ακατάλληλο, είναι η διεπαφή που προσφέρει μεταξύ Linux και περιφερειακών του SoC μέσω του framework Remote Processor framework (RPROC) [21], γνωστό και ως remoteproc. Επάνω σε αυτό το framework, η T.I έχει ενσωματώσει έναν μηχανισμό ανταλλαγής μηνυμάτων που τον ονομάζει RPMsg [22], το οποίο αναλαμβάνει την επικοινωνία μεταξύ του Linux kernel (επεξεργαστής ARM) και των PRU, χρησιμοποιώντας memory buffers.

Ο λόγος για τον οποίο το RPMsg δεν εξυπηρετεί τον σκοπό του οδηγού, είναι η έλλειψη της δυνατότητας για άμεση διευθυνσιοδότηση της μνήμης, μια απαραίτητη λειτουργία για την αλληλεπίδραση με το υλικό στο χαμηλό επίπεδο.

Τα Linux είναι εξοπλισμένα με τον δικό τους μηχανισμό για επικοινωνία με το υλικό και ονομάζεται Userspace I/O (UIO) [23]. Προσφέρει επικοινωνία σε «ωμή» μορφή χωρίς σύνθετους μηχανισμούς μεταξύ του user space και του υλικού. Είναι εύκολο στη λειτουργία για τις περισσότερες των περιπτώσεων και χρησιμοποιείται αρκετό καιρό ώστε να έχει εξελιχθεί σε αξιόπιστο μηχανισμό. Για την χρήση του UIO, απαιτείται η λήψη και εγκατάσταση του kernel “bone”.

Για την λήψη του kernel όμως, πρέπει το BeagleBone να συνδεθεί στο διαδίκτυο και να ρυθμιστεί η εσωτερική του ώρα και ημερομηνία στην τρέχουσα. Για να γίνει αυτό υπάρχουν δύο επιλογές, είτε χρησιμοποιώντας καλώδιο Ethernet απευθείας στο BeagleBone, είτε μέσω της θύρας USB να χρησιμοποιηθεί η σύνδεση στο διαδίκτυο του H/Y [24]. Ενώ ο πρώτος τρόπος φαίνεται να είναι και ο ευκολότερος, στην πραγματικότητα παρουσιάζει προβλήματα με την σύνδεση στους servers του github, από

όπου πρέπει να γίνει η λήψη του kernel. Η σύνδεση μέσω USB φαίνεται να λειτουργεί περισσότερο αξιόπιστα.

Για την ρύθμιση της ημερομηνίας πρέπει να εισαχθεί η εντολή “date --set yyyy-mm-dd”, ενώ για την ώρα η σωστή εντολή είναι η “date --set hh:mm:ss”. Λόγω έλλειψης κυκλώματος τροφοδοσίας του ρολογιού, σε περίπτωση όπου χαθεί η τάση πριν ολοκληρωθεί η λήψη του kernel, οι παραπάνω εντολές πρέπει να επαναληφθούν.

Έχοντας κάνει τις παραπάνω ρυθμίσεις και με ενεργή σύνδεση στο διαδίκτυο, οι επόμενες τρεις εντολές κατεβάζουν και εγκαθιστούν τον νέο kernel.

- cd /opt/scripts/tools/
- git pull
- sudo ./update_kernel.sh --bone-kernel --lts4_4

Μετά από επανεκκίνηση, με την εκτέλεση της εντολής “uname -a”, επιστρέφεται η τρέχουσα έκδοση του kernel, η οποία πρέπει να περιέχει την λέξη “bone” ώστε η διαδικασία να ήταν επιτυχής. Τα ακριβής νούμερα της έκδοσης δεν έχουν σημασία, καθώς με τον καιρό θα αυξάνονται και θα πάψουν να είναι ίδια με αυτά που υπήρχαν κατά την συγγραφή της πτυχιακής εργασίας.

6.3 Εκτέλεση του οδηγού

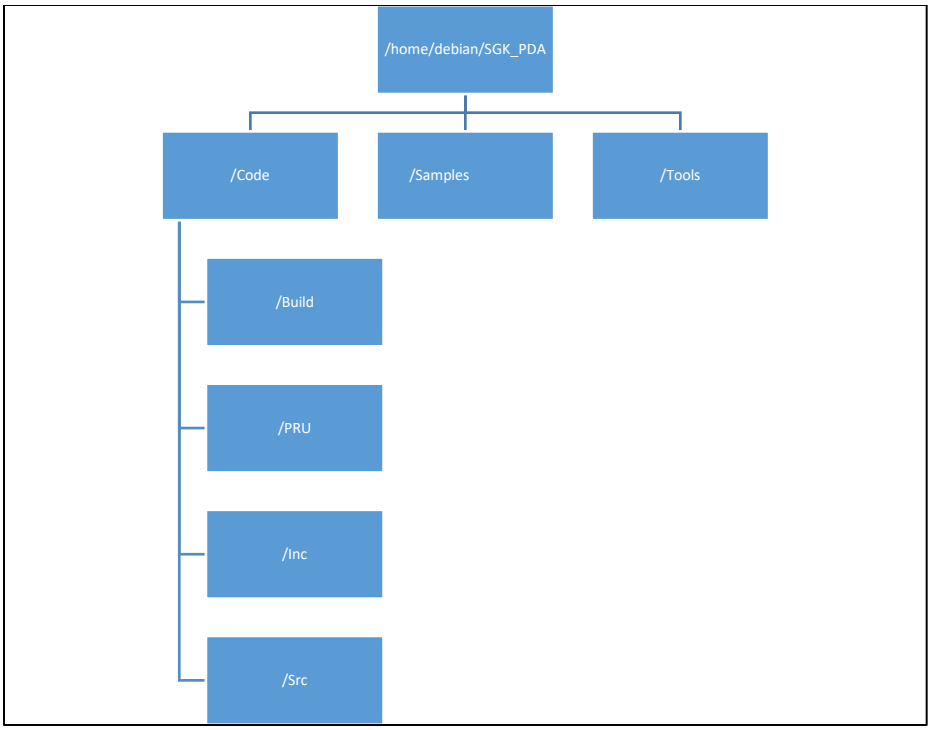
Το μόνο που απομένει για την εκτέλεση του οδηγού, είναι η δημιουργία των φακέλων, η μεταφορά του πηγαίου κώδικα της εργασίας στο BeagleBone και η δημιουργία των εκτελέσιμων αρχείων. Επιπλέον, είναι απαραίτητη η ενεργοποίηση του υποσυστήματος των PRU πριν την εκτέλεση του κώδικα. Στις παραγράφους που ακολουθούν, παρουσιάζονται οι παραπάνω διαδικασίες.

6.3.1 Δημιουργία φακέλων

Ο κώδικας του οδηγού μαζί με όλα τα συνοδευτικά αρχεία πρέπει να τοποθετηθούν σε συγκεκριμένους φακέλους για να λειτουργήσει το σύστημα. Στο σχήμα 6.6 παρουσιάζεται η δομή των φακέλων. Για την δημιουργία της δομής, πρέπει να εκτελεστούν οι παρακάτω εντολές:

- cd /home/debian
- mkdir -m 777 SGK_PDA
- mkdir -m 777 SGK_PDA/Code
- mkdir -m 777 SGK_PDA/Samples
- mkdir -m 777 SGK_PDA/Tools
- mkdir -m 777 SGK_PDA/Code/Build
- mkdir -m 777 SGK_PDA/Code/Src
- mkdir -m 777 SGK_PDA/Code/Inc
- mkdir -m 777 SGK_PDA/Code/PRU

Στη συνέχεια, χρησιμοποιώντας το γραφικό περιβάλλον του WinSCP, μπορούν να μεταφερθούν τα αρχεία από τους αντίστοιχους φακέλους του Η/Υ σε αυτούς που μόλις δημιουργήθηκαν στο BeagleBone.



Σχήμα 6.6 Η δομή των φακέλων του project

6.3.2 Δημιουργία εκτελέσιμων αρχείων

Το πρώτο αρχείο που πρέπει να φορτώνεται κάθε φορά είναι αυτό του DTO. Στο φάκελο /home/debian/SGK_PDA/Tools υπάρχει το αρχείο με ονομασία SGK-PDA.dts, το οποίο πρέπει να εισαχθεί στον device tree compiler για την παραγωγή του τελικού αρχείου. Για να επιτευχθεί αυτός ο σκοπός πρέπει να εκτελεστούν οι παρακάτω τρεις εντολές:

- cd /home/debian/SGK_PDA/Tools
- dtc -O dtb -o SGK-PDA-00A0.dtbo -b 0 -@ SGK-PDA.dts
- sudo cp SGK-PDA-00A0.dtbo /lib/firmware

Με την ολοκλήρωση της δεύτερης εντολής, πρέπει να εμφανιστεί το αρχείο με όνομα SGK-PDA-00A0.dtbo μέσα στον φάκελο Tools. Η τελευταία εντολή μεταφέρει το τελικό αρχείο στο κατάλληλο φάκελο του συστήματος.

Στη συνέχεια, πρέπει να παραχθούν τα binary αρχεία των PRU, των οποίων ο πηγαίος κώδικας βρίσκεται στο φάκελο /home/debian/SGK_PDA/Code/PRU. Η εκτέλεση των παρακάτω εντολών θα πρέπει να έχει ως αποτέλεσμα την δημιουργία δύο νέων αρχείων μέσα στον ίδιο φάκελο, με ονόματα PRU0.bin και PRU1.bin.

- cd /home/debian/SGK_PDA/Code/PRU
- pasm -b PRU0.p
- pasm -b PRU1.p

Η εκτέλεση των εντολών pasm ενεργοποιεί τον assembler και πρέπει να έχει ως αποτέλεσμα μηδενικά error και warnings.

Στο τέλος έμεινε η χρήση του C Compiler για την παραγωγή του εκτελέσιμου αρχείου του C κώδικα. Τα πηγαία αρχεία βρίσκονται στο φάκελο /home/debian/SGK_PDA/Code/Src ενώ τα headers βρίσκονται στο φάκελο /home/debian/SGK_PDA/Code/Inc. Για τη δημιουργία του εκτελέσιμου αρχείου, έχει γραφεί ένα Makefile [25] με οδηγίες για τον Compiler του οποίου το περιεχόμενο παρουσιάζεται παράρτημα του εγγράφου. Εκτελώντας τις παρακάτω εντολές, παράγεται το αρχείο pda_drivers_exec στον φάκελο /home/debian/SGK_PDA/Code.

- cd /home/debian/SGK_PDA/Code
- make

Σε αυτό το σημείο, έχουν δημιουργηθεί όλα τα εκτελέσιμα αρχεία και ο οδηγός είναι έτοιμος για χρήση.

6.3.3 Εκκίνηση του οδηγού

Εκτελώντας τις παρακάτω δύο εντολές, τα Linux τυπώνουν τους ακροδέκτες που βρίσκονται σε χρήση ενώ είναι χωρισμένοι σε γκρουπ σύμφωνα με την λειτουργία τους.

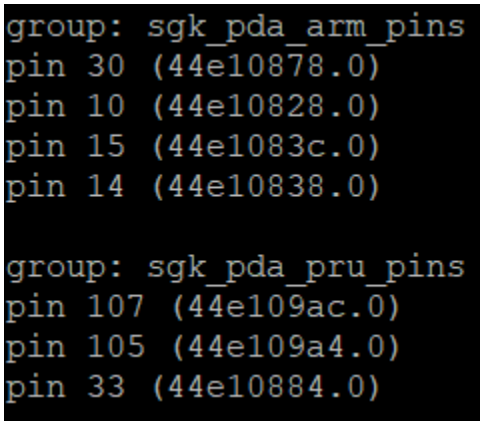
- cd /sys/kernel/debug/pinctrl/44e10800.pinmux

- cat pingroups

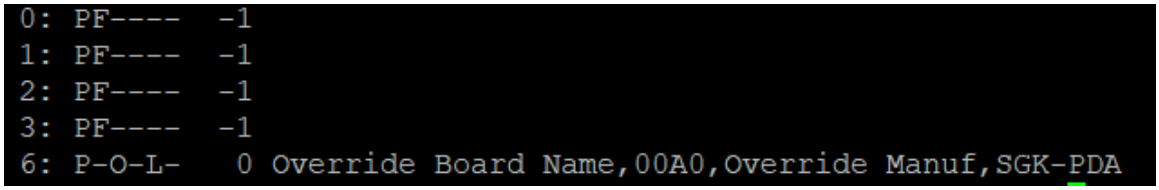
Για να εισαχθεί στη λίστα το γκρουπ των ακροδεκτών που χρησιμοποιείται από την παρούσα εργασία, πρέπει να φορτωθεί το DTO. Αρχικά όμως πρέπει να ενεργοποιηθούν οι PRU, καθώς στο DTO υπάρχει οδηγία που τους συσχετίζει με κάποιους ακροδέκτες. Για να επιτευχθεί αυτό, πρέπει να γίνει χρήση του προγράμματος modprobe των Linux, το οποίο είναι υπεύθυνο για την φόρτωση μονάδων στον kernel. Έτσι εκτελώντας τις επόμενες δύο εντολές, γίνεται εκκίνηση των PRU καθώς και φόρτωση του DTO.

- modprobe uio_pruss
- echo SGK-PDA > /sys/devices/platform/bone_capemgr/slots

Με την επανάληψη των εντολών που τυπώνουν τα ενεργά γκρουπ ακροδεκτών, επιστρέφεται η λίστα που περιέχει πλέον και τους ακροδέκτες που αφορούν την εργασία και ελέγχονται από τις CPU, όπως φαίνεται στο σχήμα 6.7. Μπορεί να γίνει ένας επιπλέον έλεγχος, δίνοντας την εντολή “cat /sys/devices/platform/bone_capemgr/slots” που παρουσιάστηκε στην αρχή του κεφαλαίου για την εμφάνιση των ενεργών capes του συστήματος. Η σωστή εικόνα του αποτελέσματος φαίνεται στο σχήμα 6.8.



Σχήμα 6.7 Τα γκρουπ ακροδεκτών που χρησιμοποιούν οι CPU



Σχήμα 6.8 Η λίστα με τα ενεργά capes αφού φορτώθηκε αυτό της εργασίας

Μία τελική παρέμβαση στο σύστημα του BeagleBone, είναι η απενεργοποίηση των user LED που αναβοσβήνουν κατά την λειτουργία της πλακέτας, για να αποφευχθεί ο θόρυβος στο PDA. Για την μόνιμη απενεργοποίησή τους, ακόμη και μετά από reboot, πρέπει να δοθούν οι παρακάτω εντολές.

- echo none > /sys/class/leds/beaglebone\:green\:usr0/trigger
- echo none > /sys/class/leds/beaglebone\:green\:usr1/trigger
- echo none > /sys/class/leds/beaglebone\:green\:usr2/trigger
- echo none > /sys/class/leds/beaglebone\:green\:usr3/trigger

Το μόνο που απομένει για την εκτέλεση του οδηγού, είναι η χρήση της εντολής που τον χρησιμοποιεί. Όπως αναφέρεται και στο κεφάλαιο του λογισμικού, η κλήση του οδηγού πρέπει να συνοδεύεται από 4 ορίσματα, εκ των οποίων το καθένα έχει ένα εύρος αποδεκτών τιμών. Σε αυτό το σημείο, για λόγους δοκιμής προτείνεται η χρήση των ορισμάτων που φαίνεται παρακάτω.

- cd /home/debian/SGK_PDA/Code
- ./sgk_pda_exec 10 80 50 100

Εάν όλα έχουν γίνει σωστά, θα πρέπει να μην εμφανιστούν σφάλματα στην κονσόλα καθώς και να έχει δημιουργηθεί αρχείο δείγματος στο φάκελο /home/debian/SGK_PDA/Samples με όνομα Sample_00001.txt. Το αρχείο αυτό δεν πρέπει να περιέχει καμία τιμή μεγαλύτερη του 4095, δεδομένου ότι ο ADC έχει 12bit ανάλυση, άρα η μέγιστη τιμή που μπορεί να δώσει για μία μετατροπή είναι το 4095.

7 Συμπεράσματα και προτάσεις βελτίωσης

Η υψηλή ακρίβεια στο χρονισμό των διαδικασιών που παρουσιάστηκαν σε αυτή τη πτυχιακή εργασία, είναι αποτέλεσμα του παράλληλου προγραμματισμού στο χαμηλό επίπεδο. Είναι φανερό πως ακόμη και αν το υλικό είναι χαμηλής αξίας, μετά από ενδελεχή μελέτη του και σωστή χρήση των πόρων, μπορεί να επιτευχθεί ποιοτικό αποτέλεσμα.

Περιπτώσεις όπως αυτή της παρούσας εργασία, αποδεικνύουν πως ο προγραμματισμός στο χαμηλό επίπεδο, κοντά στη μηχανή, είναι μέχρι σήμερα απαραίτητος για την παραγωγή αξιόπιστων αποτελεσμάτων σε χρονικά κρίσιμα συστήματα.

7.1 Εξέλιξη του υπάρχον συστήματος

Μοναδική πρόταση για την εξέλιξη του συστήματος που αναπτύχθηκε για την εργασία χωρίς την τροποποίηση του υλικού, είναι η εισαγωγή ενός αλγορίθμου πακεταρίσματος των δειγμάτων του ADC με σκοπό την εξοικονόμηση μνήμης.

Στη παρούσα φάση, ο PRU1 γράφει στη DDR RAM τα δεδομένα αμέσως μόλις τα πάρει από τον ADC, καταναλώνοντας 2 byte για κάθε ένα δείγμα καθώς η ανάλυση του ADC είναι 12bit. Με αυτό τον τρόπο «χάνονται» 4bit για κάθε δείγμα, δηλαδή ένα ολόκληρο byte σε κάθε δύο δείγματα.

Προτείνεται λοιπόν το πακετάρισμα των δειγμάτων χρησιμοποιώντας τρία byte αντί για τέσσερα, για κάθε δύο δείγματα. Για λόγους ευκολίας κατά τη προσπέλαση της μνήμης, τα δεδομένα θα συλλέγονται αρχικά στη RAM του PRU1 και θα γράφονται στη DDR RAM κάθε φορά που θα συμπληρώνονται τέσσερα bytes, δηλαδή κάθε 2,66 δείγματα. Με αυτό το τρόπο δεν υπάρχει σπατάλη στη RAM ενώ μένει ίδιος ο χρόνος που απαιτείται για πρόσβαση σε διάδρομο διασύνδεσης από τον PRU1. Στο σχήμα 7.1 παρουσιάζεται με γραφικό τρόπο η μέθοδος πακεταρίσματος που προτείνεται.

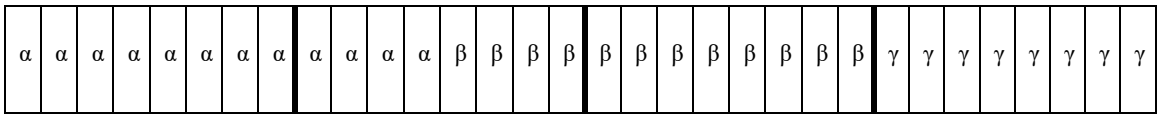


Figure 7.1 Πακετάρισμα 2,66 δειγμάτων σε 4 byte

Με το τέλος του οδηγού, κατά την συλλογή των δεδομένων από τη RAM, ο ARM θα τρέχει την αντίστροφη διαδικασία για το διαχωρισμό των δειγμάτων πριν τα γράψει στο αρχείο .txt.

Για δοκιμή της μεθόδου θα τροποποιηθεί ελάχιστα ο κώδικας του PRU1 ώστε να μην πακετάρει τα δεδομένα του ADC, αλλά το περιεχόμενο ενός software μετρητή ο οποίος θα αυξάνεται σε γνωστό εύρος τιμών. Στο τέλος της διαδικασίας με έναν έλεγχο του περιεχομένου του .txt που σώζεται, μπορεί να διαπιστωθεί εύκολα αν η μέθοδος λειτουργεί ή όχι.

Μία ακόμη προσθήκη στο υπάρχον σύστημα θα μπορούσε να είναι η προσθήκη ενός πέμπτου συντελεστή εισόδου που θα αφορά τον ρυθμό μεταβολής του τελεστικού ενισχυτή για την ρύθμιση της χρονικής στιγμής κατά την οποία σκανδαλίζεται ο ADC.

7.2 Αλλαγή του υλικού

Το σύστημα θα μπορούσε να αναβαθμιστεί και με τη χρήση διαφορετικού αισθητήρα. Με την αλλαγή του PDA σε κάποιο άλλο με παρόμοιο τρόπο λειτουργίας, το οποίο όμως θα προσφέρει περισσότερα pixel ή με χρήση ενός εντελώς διαφορετικού αισθητήρα που απαιτεί νέο οδηγό.

Για λόγους μείωσης κόστους αλλά και για την απλούστευση του συστήματος, θα μπορούσε να επιλεχθεί διαφορετικός επεξεργαστής. Υπάρχουν μικροελεγκτές στην αγορά, με προγραμματιζόμενη λογική εσωτερικά του ολοκληρωμένου. Πρόκειται για ψηφιακά

και αναλογικά μπλοκ, τα οποία λειτουργούν ανεξάρτητα της CPU και είναι πλήρως προγραμματιζόμενα από τον χρήστη.

Η σειρά PSoC της Cypress [26] προσφέρει ολοκληρωμένα με επεξεργαστές ARM Cortex M και προγραμματιζόμενη λογική. Το κόστος των ολοκληρωμένων είναι πολύ χαμηλότερο από αυτό του AM3358 που χρησιμοποιείται στο BeagleBone ενώ οι αναπτυξιακές πλακέτες είναι περίπου στη μισή τιμή.

Αρκετά δημοφιλής είναι ο μικροελεγκτής ATMega4809 της Microchip που χρησιμοποιείται στο Arduino Uno Wifi Rev 2. Ο συγκεκριμένος είναι αρκετά πιο αδύναμος από το SoC του BeagleBone αλλά και από τους PSoC, όμως έχει πολύ χαμηλότερο κόστος. Χρησιμοποιεί την τεχνολογία “Configurable Custom Logic” (CCL) της Microchip, η οποία ουσιαστικά είναι παρόμοια με αυτή των PSoC, δηλαδή προγραμματιζόμενη λογική ανεξάρτητη του πυρήνα.

Κοινό στοιχείο των ολοκληρωμένων που αναφέρθηκαν είναι η προγραμματιζόμενη λογική. Ανεξάρτητα από την ονομασία που δίνει ο κάθε κατασκευαστής στην υλοποίηση του, η λογική είναι ίδια σε κάθε περίπτωση. Πρόκειται για προγραμματιζόμενο υλικό το οποίο λειτουργεί ανεξάρτητα από την CPU, χωρίς καθυστερήσεις στην εκτέλεση των εργασιών του, ενώ στις περισσότερες των περιπτώσεων, παρέχεται η δυνατότητα επικοινωνίας με την CPU για την τροποποίηση της λειτουργίας του.

8 Παραρτήματα

Το κεφάλαιο των παραρτημάτων χρησιμοποιείται για την τοποθέτηση των αρχείων κώδικα που χρησιμοποιούνται στην παρούσα πτυχιακή εργασία.

8.1 Device Tree Overlay

```
/*
 *Stratos Gkagkanis
 *Thesis Project:
 *"Fabrication of a photodiode sensor system using the Beaglebone
 *microcomputer for use in non-invasive biomedical sensors."
 *Complete Device Tree Overlay file.
 *Hardware: TSC-ADC Module, PRU0, PRU1.
 *I/O:  AIN0 (AO), GPIO3_19 (CLK), GPIO3_21 (SI), GPIO1_13 (DEBUG)
 */

/dts-v1/;
/plugin/;

/{
    compatible = "ti,beaglebone", "ti,beaglebone-black";
    part-number = "SGK-PDA";
    version = "00A0";

    exclusive-use =
        //P8 Header
        "P8.20",    //Debug Pin (GPIO1_13  // GPIO 45)
        "gpio1_13",
        "P8.14",    //LED RED (GPIO0_26)
        "gpio0_26",
        "P8.15",    //LED GREEN (GPIO1_15)
        "gpio1_15",
        "P8.16",    //LED BLUE (GPIO1_14)
        "gpio1_14",

        //P9 Header
        "P9.25",    //SI Pin (GPIO3_21 // GPIO 117)
        "gpio3_21",
        "P9.27",    //Clock Generator Pin (GPIO3_19  // GPIO 115)
        "gpio3_19",
        "P9.39",    //AIN0 Pin
        "AIN0",
        "P9.12",    //Button (GPIO1_28)
        "gpio1_28",


  //Hardware
        "pru0",     //PRU0
        "pru1",     //PRU1
        "tscadc";   //Touchsrceen and Analog to Digital Converter
```

```
    fragment@0{
            target = <&am33xx_pinmux>;
            __overlay__{
            pru_pru_pins: sgk_pda_pru_pins {
                pinctrl-single,pins = <
                    0x1ac 0x05  // P9_25 MODE5 |PRU0 SI
                    0x1a4 0x05  // P9_27 MODE5 |PRU0 - CLK
                    0x084 0x05  // P8_20 MODE5 |PRU1 - DEBUG
                >;
                };
            arm_contr_pins: sgk_pda_arm_pins {
                pinctrl-single,pins = <
                    0x078 0x27  // P9_12 gpio1[28],       MODE7 |BUTTON
                    0x028 0x07  // P8_14 gpio0[26],       MODE7 |LED RED
                    0x03c 0x07  // P8_15 gpio1[15],       MODE7 |LED GREEN
                    0x038 0x07  // P8_16 gpio1[14],       MODE7 |LED BLUE
                >;
                };
            };
    };

    fragment@1 {
      target = <&pruss>;
      __overlay__ {
         status = "okay";
         pinctrl-names = "default";
         pinctrl-0 = <&pru_pru_pins>;
        SGK_pins{
                pin-names = "pin319","pin321","pin131";
                gpios     = <&gpio3 17 0
                             &gpio3 15 0
                             &gpio1 63 0
                             >;
                };
      };
   };
fragment@2 {          // Enable the GPIOs
      target = <&ocp>;
      __overlay__ {
         arm_contr_pins_pinmux {
            //compatible = "gpio-of-helper";
            compatible = "bone-pinmux-helper";
            status = "okay";
            pinctrl-names = "default";
            pinctrl-0 = <&arm_contr_pins>;
        SGK_pins{
                pin-names = "pin128","pin026","pin115","pin114";
                gpios     = <&gpio1 60 0
                             &gpio0 26 0
                             &gpio1 47 0
                             &gpio1 46 0
                             >;
                      };
```

```
                };
            };
        };

        fragment@3{
                target = <&tscadc>;
                __overlay__{
                status = "okay";
                adc {
                    ti,adc-channels = <0>;
                    ti,chan-step-avg=<0x0>;
                    ti,chan-step-opendelay=<0x0>;
                    ti,chan-step-sampledelay=<0x0>;
                    };
                };
         };
    };
```

8.2 Makefile

```
INCDIR = Inc
SRCDIR = Src
OBJDIR = Build
CC=gcc
CFLAGS=-I$(INCDIR)

_DEPS = pda_drivers.h gpio.h sample_file.h debug.h
DEPS = $(patsubst %,$(INCDIR)/%,$(_DEPS))

_OBJ = pda_drivers.o gpio.o sample_file.o debug.o
OBJ = $(patsubst %,$(OBJDIR)/%,$(_OBJ))

LIBS = -lpthread -lprussdrv -lm

$(OBJDIR)/%.o: $(SRCDIR)/%.c $(DEPS)
    $(CC) -c -o $@ $< $(CFLAGS)

pda_drivers_exec: $(OBJ)
    $(CC) -o $@ $^ $(CFLAGS) $(LIBS)

.PHONY: clean

clean:
    rm -f $(OBJDIR)/*.o *~ core
    @echo Done cleaning....
```

8.3 Κώδικας

Ο πηγαίος κώδικας της εργασίας μαζί με τα υπόλοιπα αρχεία που απαιτούνται για τη λειτουργία του οδηγού, μπορούν να βρεθούν στο github του συγγραφέα της πτυχιακής εργασίας ακολουθώντας τον παρακάτω σύνδεσμο.

https://github.com/StratosGK

8.4 Λίστα υλικών κατασκευής

Στον πίνακα 8.1 παρατίθενται τα υλικά που χρησιμοποιήθηκαν για την κατασκευή του συστήματος μαζί με τα επιπλέον υλικά που χρησιμοποιήθηκαν για την αποσφαλμάτωση και τα πειράματα. Οι αριθμοί παραγγελίας τους αφορούν τον διαδικτυακό προμηθευτή ηλεκτρονικού υλικού, Mouser Electronics.

BeagleBone Black Rev C	958-BBB01-SC-505
TSL1401CL	856-TSL1401CL
LM6142 BIN/NOPB (x2)	926-LM6142BIN/NOPB
AVR Arduino Nano V3	992-ARD-NANO30
Πλήκτρο	611-PTS645SK952
Τρίχρωμο LED	743-HV-5RGB60
Κεραμικός πυκνωτής 0.1μF	-
Αντίσταση 1.5KΩ 1%	-
Αντίσταση 1.8KΩ 1%	-
Αντίσταση 330Ω (x4) 1%	-
Αντίσταση 480Ω (x2) 1%	-

Βιβλιογραφία

[1] C. Iosifidis, K. Katsaliaki, P. A. Kollenspenger και M. E. Kiziroglou, «Design of an embedded sensor system for measuring laser scattering blood cells,» σε *SPIE Microtechnologies*, 2017.

[2] BeagleBoard.org Foundation, «beagleboard.org,» BeagleBoard.org Foundation, 2013. [Ηλεκτρονικό]. Available: https://beagleboard.org/black.

[3] K. G. Shin και R. Parameswaran, «Real-Time Computing: a New Discipline of Computer Science and Engineering,» 1994.

[4] B. B. Rao, «Inline assembly for x86 in Linux,» 1 March 2001. [Ηλεκτρονικό]. Available: http://web.archive.org/web/20041210030000/http://www-106.ibm.com/developerworks/library/l-ia.html.

[5] P. Christensson, «SoC Definition,» 12 November 2015. [Ηλεκτρονικό]. Available: https://techterms.com/definition/soc.

[6] Texas Instruments, «PRU Assembly Instructions,» Texas Instruments, 9 April 2018. [Ηλεκτρονικό]. Available: http://processors.wiki.ti.com/index.php/PRU_Assembly_Instructions.

[7] eLinux, «Device Tree What It Is,» 11 December 2016. [Ηλεκτρονικό]. Available: https://elinux.org/Device_Tree_What_It_Is.

[8] AMS, «ams.com,» 30 August 2016. [Ηλεκτρονικό]. Available: https://ams.com/documents/20143/36005/TSL1401CL_DS000136_3-00.pdf/c4ec4eb0-212b-b8d0-467b-21ce2a0365fa.

[9] A. Hahn, «Application of Rail-to-Rail Operational Amplifiers,» December 1999. [Ηλεκτρονικό]. Available: http://www.ti.com/lit/an/sloa039a/sloa039a.pdf.

[10] J. Chang, «Voltage Reference Design,» 2017. [Ηλεκτρονικό]. Available: http://www.ti.com/lit/ml/slyc147/slyc147.pdf.

[11] Texas Instruments, «Texas Instruments Wiki,» Texas Instruments, 26 August 2016. [Ηλεκτρονικό]. Available: http://processors.wiki.ti.com/index.php/PRU_Linux_Application_Loader_API_Guide.

[12] K. K. e. al., «Portable system development for Mie scattering analysis, to determine the size of blood cells in in-vivo and in-vitro studies,» σε *ECESCON 11*, Thessaloniki, 2019.

[13] Software in the Public Interest, Inc, «Debian, the universal operating system,» [Ηλεκτρονικό]. Available: https://www.debian.org/.

[14] BeagleBoard.org Foundation, «BeagleBone Latest Images,» [Ηλεκτρονικό]. Available: https://beagleboard.org/latest-images.

[15] Slashdot Media, «SOURCEFORGE,» [Ηλεκτρονικό]. Available: https://sourceforge.net/projects/win32diskimager/.

[16] S. Tatham, «Download PuTTY,» [Ηλεκτρονικό]. Available: https://www.putty.org/.

[17] WinSCP.net, «WinSCP,» [Ηλεκτρονικό]. Available: https://winscp.net/eng/download.php.

[18] Software in the Public Interest, Inc, «JOURNALD.CONF,» [Ηλεκτρονικό]. Available: https://manpages.debian.org/stretch/systemd/journald.conf.5.en.html.

[19] Software in the Public Interest, Inc, «SYSTEMD-JOURNALD.SERVICE,» [Ηλεκτρονικό]. Available: https://manpages.debian.org/stretch/systemd/systemd-journald.service.8.en.html.

[20] Colorado State University, Computer Science Department, «Basic vi Commands,» [Ηλεκτρονικό]. Available: https://www.cs.colostate.edu/helpdocs/vi.html.

[21] Linux Kernel Organization, «Remote Processor Framework,» [Ηλεκτρονικό]. Available: https://www.kernel.org/doc/Documentation/remoteproc.txt.

[22] Texas Instruments, «PRU-ICSS Remoteproc and RPMsg,» [Ηλεκτρονικό]. Available: http://processors.wiki.ti.com/index.php/PRU-ICSS_Remoteproc_and_RPMsg#RPMsg.

[23] H.-J. Koch, «The Userspace I/O HOWTO,» 11 December 2006. [Ηλεκτρονικό]. Available: https://www.kernel.org/doc/html/v4.18/driver-api/uio-howto.html.

[24] MAKER.IO STAFF, «How to Connect a BeagleBone Black to the Internet Using USB,» MAKER.IO, 6 October 2016. [Ηλεκτρονικό]. Available: https://www.digikey.com/en/maker/blogs/how-to-connect-a-beaglebone-black-to-the-internet-using-usb.

[25] F. S. Foundation, «GNU make,» Free Software Foundation, [Ηλεκτρονικό]. Available: https://www.gnu.org/software/make/manual/make.html.

[26] Cypress Semiconductor Corporation, «MCU and PSoC Products,» Cypress Semiconductor Corporation, [Ηλεκτρονικό]. Available: https://www.cypress.com/products/microcontrollers-mcus.