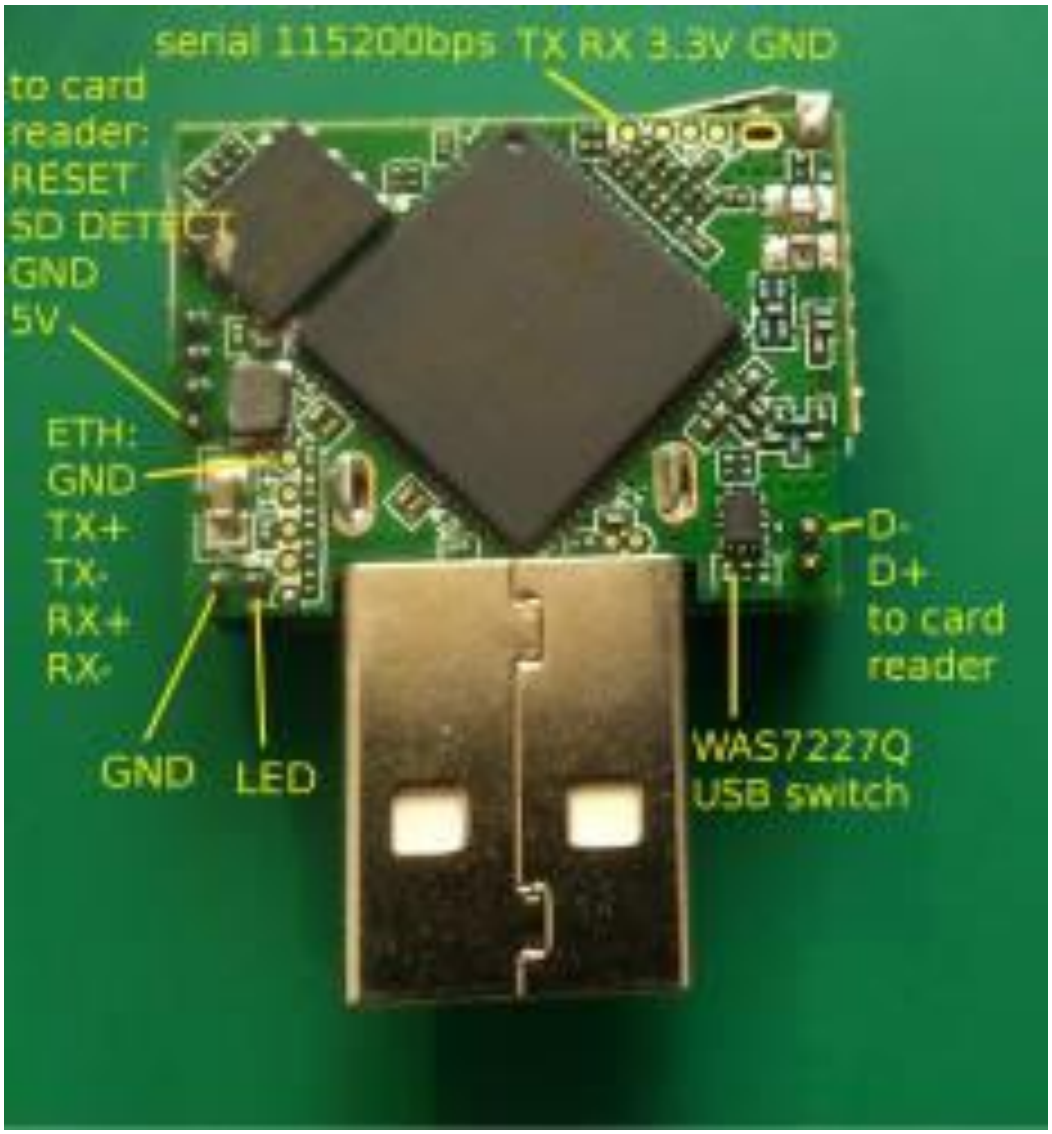
ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ

ΑΝΑΦΟΡΑ-ΑΣΚΗΣΗ 2



ΑΝΤΩΝΙΑΔΗΣ ΣΤΑΥΡΟΣ

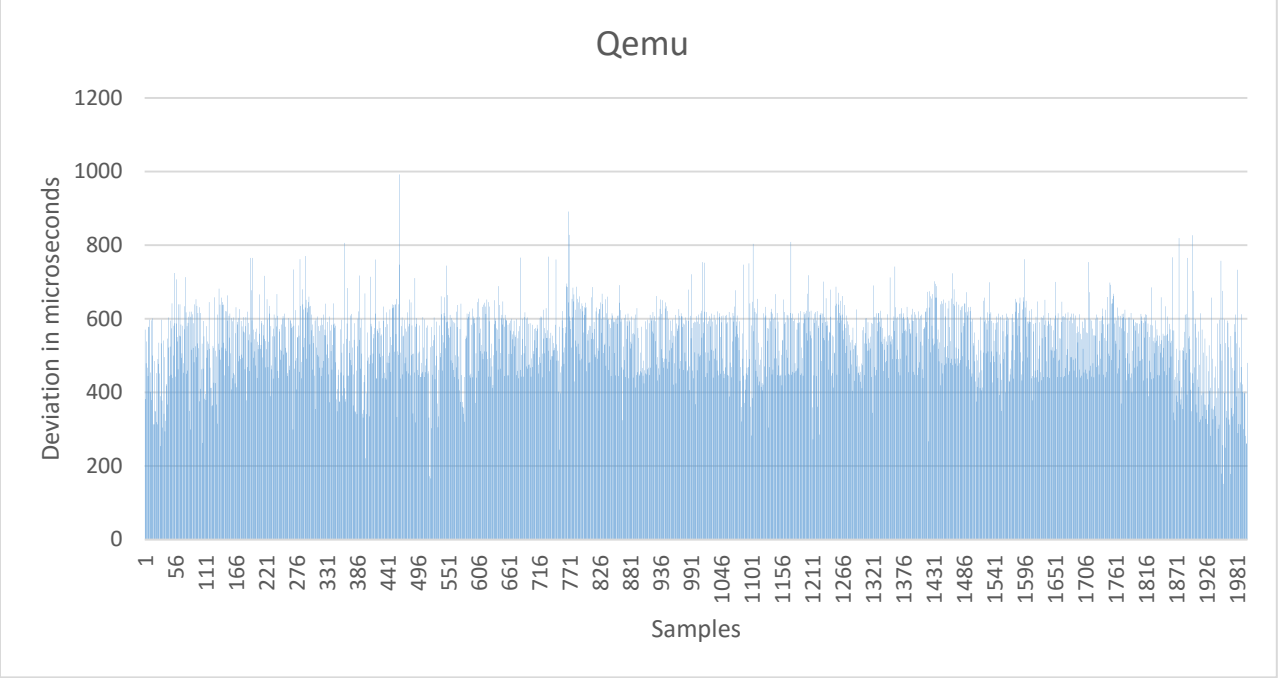
ΑΕΜ:8279

Σκοπός αυτής της εργασίας ήταν η υλοποίηση ενός χρονομέτρου που εμφανίζει την μικρότερη δυνατή απόκλιση από τον πραγματικό χρόνο. Στην συνέχεια ακολουθεί ο σχολιασμός του πηγαίου κώδικα που υλοποιήσαμε.

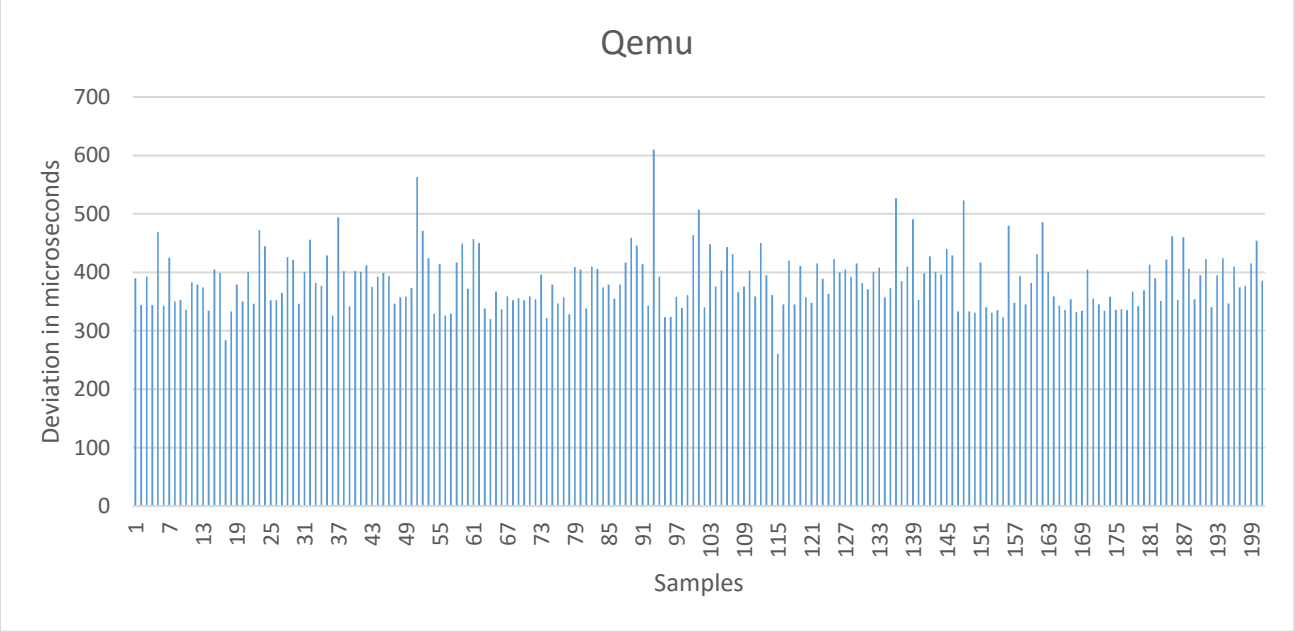
Αρχικά, το πρόγραμμα μας δέχεται σαν παραμέτρους μέσω του argv[](πρώτα το διάστημα και μετά τον αριθμό των δειγματοληψιών) τα δύο ορίσματα που ζητούνται, τα οποία είναι το διάστημα δειγματοληψίας σε δευτερόλεπτα και το πλήθος των δειγμάτων που θα πάρουμε. Έπειτα, θα ρυθμίζουμε το χρονόμετρο μας σύμφωνα με το διάστημα δειγματοληψίας μετά θα αδρανοποιούμε το σύστημα μας και κάθε φορά που θα λήγει το χρονόμετρο θα κάνουμε ένα interrupt στη συνάρτηση που το αδρανοποιεί θα κάνουμε τις μετρήσεις για το χρόνο που θέλουμε. Αυτό θα επαναλαμβάνεται για αριθμό ίσο με τον αριθμό δειγμάτων που εισάγαμε. Για να ξεκινήσουμε το χρονόμετρο όμως, πρέπει προηγουμένως να έχουμε δημιουργήσει ένα διαχειριστή σημάτων(handler) για να γνωρίζει η διαδικασία που θα σταματήσει, πως να διαχειριστεί το εκάστοτε σήμα. Το τελευταίο επιτυγχάνεται με την συνάρτηση sigaction(). Η συνάρτηση αυτή δέχεται τρία ορίσματα. Το πρώτο όρισμα αναφέρεται στο σήμα το οποίο θέλουμε να διαχειριστούμε που στην περίπτωση μας είναι το SIGALRM. Το δεύτερο όρισμα είναι ένα struct μέσω του οποίου προσδιορίζουμε κάποιες πληροφορίες για τη διαχείριση του σήματος. Το struct αυτό περιέχει τρεις μεταβλητές. Η πρώτη μεταβλητή έχει να κάνει με τον τρόπο με τον οποίο θα γίνει η διαχείριση του σήματος και μπορεί να γίνει μέσω κάποιων προκαθορισμένων διαδικασιών αλλά και μέσω μίας συνάρτησης που μπορεί να ορίσει ο προγραμματιστής, αυτό συνέβη και στην περίπτωση μας. Θα εξηγήσουμε παρακάτω τί περιέχει αυτή η συνάρτηση-handler. Η δεύτερη μεταβλητή είναι μία μάσκα η οποία προσδιορίζει ποια σήματα είναι μπλοκαρισμένα όταν τρέχει ο handler,με άλλα λόγια σε ποια σήματα δεν θα δίνει σημασία η διαδικασία ενώ τρέχει η συνάρτηση. Και η τελευταία μεταβλητή προσδιορίζει διάφορες flags που μπορεί να επηρεάσουν την συμπεριφορά του σήματος. Τέλος το τελευταίο όρισμα της συνάρτησης sigaction() ένα ίδιο struct που σκοπό έχει να μας επιστρέψει πληροφορίες για το παλιό τρόπο διαχείρισης του σήματος. Αφού ορίσαμε λοιπόν το διαχειριστή του σήματος ήρθε η ώρα να ξεκινήσουμε το χρονόμετρο. Αυτό θα γίνει μέσω της συνάρτησης setitimer(). Η συνάρτηση αυτή δέχεται τρία ορίσματα το πρώτο προσδιορίζει το timer που θα ρυθμιστεί σύμφωνα με τις τιμές του δεύτερου ορίσματος. Στην περίπτωση μας θέλουμε να

έναν real-time timer ο οποίος θα υπολογίζει τον χρόνο που έχει περάσει (elapsed time) και αυτό επιτυγχάνεται με την τιμή `ITIMER_REAL`. Το δεύτερο όρισμα είναι ένα struct μέσω του οποίου όπως προαναφέραμε προσδιορίζεται το χρονικό διάστημα που θα τρέξει το χρονόμετρο και ουσιαστικά είναι η τιμή που πήραμε ως είσοδο που αφορούσε το διάστημα δειγματοληψίας. Και το τρίτο όρισμα είναι ένα ίδιο struct και επιστρέφει πληροφορίες για το προηγούμενο άληκτο σήμα. Να αναφέρουμε εδώ ότι κάθε φορά που το χρονόμετρο λήγει παράγεται το σήμα SIGALRM. Στην συνέχεια αμέσως μετά το κάλεσμα της συνάρτησης setitimer() καλούμε την συνάρτηση gettimeofday() για να ξέρουμε πότε ξεκίνησε το πρώτο χρονόμετρο και άρα στην συνέχεια να μπορέσουμε να υπολογίσουμε πόσο χρόνο έκανε να λήξει. Τέλος και πριν σχολιάσουμε την συνάρτηση διαχείρισης σήματος, τρέχουμε σε μία λούπα για n φορές , οπού n είναι ο αριθμός των δειγμάτων που θέλουμε να πάρουμε, την συνάρτηση sigsuspend() η οποία αδρανοποιεί το σύστημα μας και περιμένει να διακοπεί κάθε φορά που λήγει το χρονόμετρο. Η συνάρτηση αυτή παίρνει ένα όρισμα, το οποίο είναι μια μάσκα τύπου sigset_t και ουσιαστικά προσδιορίζει ποια σήματα μπορούν να δεν σταματήσουν τη λειτουργία της ή με άλλα λόγια ποια σήματα είναι μπλοκαρισμένα . Να πούμε σε αυτό το σημείο ότι η μάσκα γέμισε με την συνάρτηση sigfillset() και στην συνέχεια αφαιρέθηκε το σήμα(SIGALRM) που θέλουμε να μπορεί να της κάνει interrupt με την συνάρτηση sigdelset(). Όσον αφορά τον handler τώρα, ο μόνος σκοπός που επιτελούσε ήταν η μέτρηση του χρόνου που τελειώνει κάθε δειγματοληψία και η αποθήκευση αυτού του timestamp σε ένα πίνακα. Αυτό το υλοποιήσαμε μέσω μίας αφαίρεσης του χρόνου που είχαμε μόλις ξεκίνησε το πρώτο χρονόμετρο μείον του χρόνου που παίρναμε κάθε φορά που μπαίναμε στον handler, δηλαδή κάθε φορά που έληγε ένα χρονόμετρο. Παρακάτω παρουσιάζονται κάποια διαγράμματα όπου φαίνεται η απόκλιση που είχε εν τέλει το χρονόμετρο πρώτα στο Qemu, το οποίο είναι ένας προσομοιωτής real-time συσκευής, και στη συνέχεια στο Zsun. Έπειτα ακολουθούν κάποιες παρατηρήσεις για τα διαγράμματα και τέλος το μπόνους κομμάτι της εργασίας που αφορά την ενεργειακή απόδοση του προγράμματος μας.

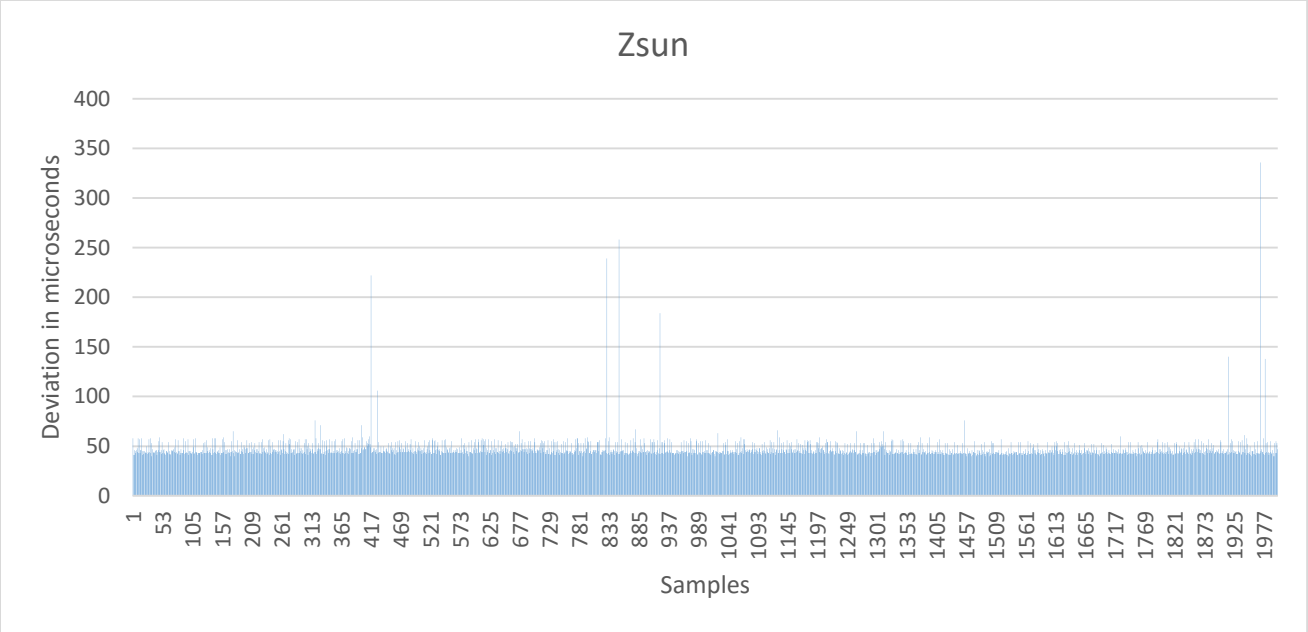
Διάστημα δειγματοληψίας 1 sec και αριθμός δειγμάτων 20000. Η μέση τιμή της απόκλισης που προέκυψε είναι 535.8181 microseconds.



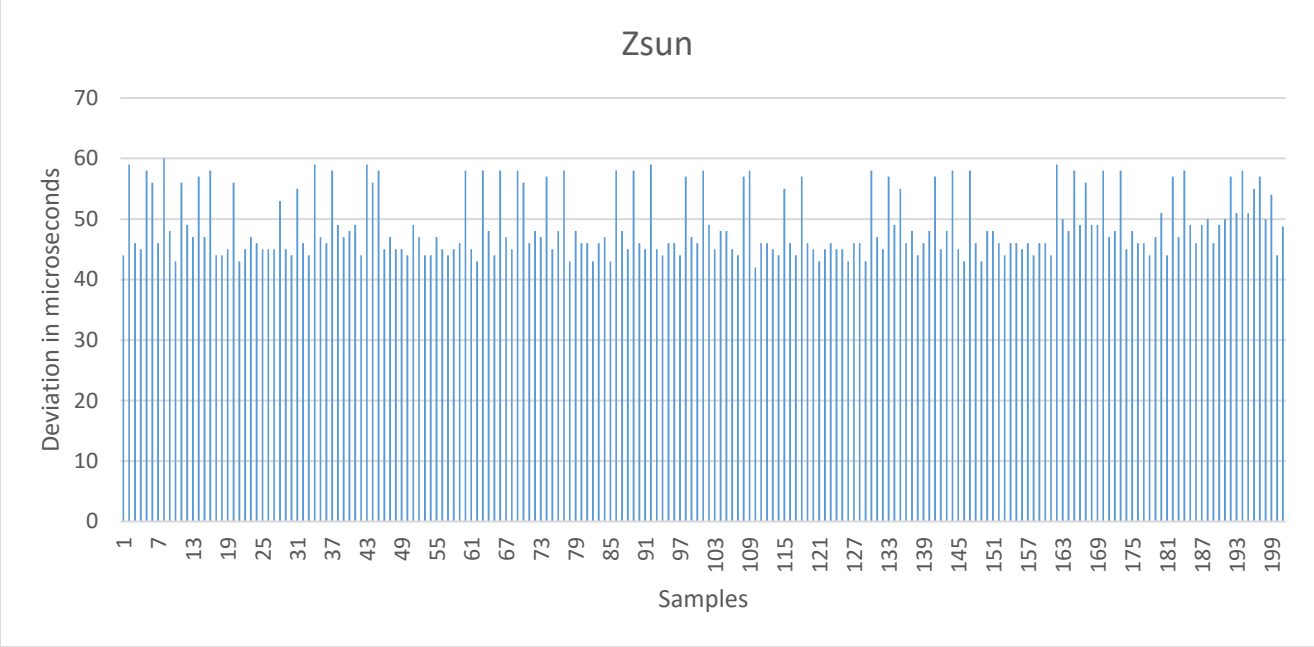
Διάστημα δειγματοληψίας 10 sec και αριθμός δειγμάτων 2000. Η μέση τιμή της απόκλισης που προέκυψε είναι 385.885 microseconds.



Διάστημα δειγματοληψίας 1 sec και αριθμός δειγμάτων 20000. Η μέση τιμή της απόκλισης που προέκυψε είναι 46.8205 microseconds.



Διάστημα δειγματοληψίας 10 sec και αριθμός δειγμάτων 2000. Η μέση τιμή της απόκλισης που προέκυψε είναι 48.78 microseconds.



Παρατηρούμε ότι η απόκλιση στον προσομοιωτή είναι αρκετά μεγαλύτερη σε σχέση με αυτήν του Zsun. Γενικά βλέπουμε ότι η διασπορά δεν είναι πολύ μεγάλη και η διαφορά της απόκλισης μεταξύ των δειγμάτων είναι πολύ μικρή με κάποιες ελάχιστες εξαιρέσεις.

BONUS

Τρέξαμε το πρόγραμμα μας στο Zsun το οποίο ήταν συνδεδεμένο με ένα powerbank με χωρητικότητα μπαταρίας 2500mAh. Το χρονόμετρο μας έτρεξε με διάστημα δειγματοληψίας 1 sec και μετά από 5 ώρες και 30 λεπτά η μπαταρία δεν είχε πέσει οπότε δεν καταφέραμε να βγάλουμε κάποιο συμπέρασμα για την ενεργειακή του απόδοση.