

Ενσωματωμένα Συστήματα Πραγματικού Χρόνου

Εργασία 2016 Ρολόι - Χρονόμετρο

ΧΑΤΖΗΘΩΜΑ ΑΝΤΡΕΑΣ

AEM: 8026

antreasc@ece.auth.gr

Εισαγωγή

Το ζητούμενο της εργασίας είναι η υλοποίηση ενός ρολογιού – χρονομέτρου το οποίο να τρέχει σε γυμνό νη που τρέχει κάποιο απλό RTOS. Το ρολόι εμφανίζει την τρέχουσα ώρα σε μορφή HH:MM:SS (σε 24ωρο format) ή την τρέχουσα τιμή του χρονομέτρου σε μορφή HH:MM:SS.dd ζωντανά. Αφού ρυθμίστηκε το RTOS που επιλέχθηκε, γράφτηκε ο εκτελέσιμος κώδικας και παράχθηκε ένα binary αρχείο το οποίο έχει τη δυνατότητα να τρέχει σε ένα γυμνό νη.

Επιλογή και Ρύθμιση του RTOS

Αρχικά, επιλέχθηκε το [riot rtos](#) το οποίο φαίνεται ένα πολύ υποσχόμενο RTOS. Παρόλα αυτά, στην πορεία αντιληφθήκαμε ότι δεν υπάρχει κάποιο img το οποίο να μας δίνει τη δυνατότητα να φορτώσουμε το rtos σε γυμνό νη. Έπειτα, επιλέχθηκε το προτεινόμενο από τους διδάσκοντες, [prex](#).

Το prex, δεν έχει ανανεωθεί από το 2009, πράγμα το οποίο έκανε την ρύθμιση του πιο δύσκολη, εφόσον υπήρχαν αρκετά προβλήματα συμβατότητας. Μετά όμως από επιμονή, και σε συνεργασία με συνάδελφους στο thmmy, αλλά και με την βοήθεια του υποψήφιου διδάκτορα Δημήτρη Φλώρου, καταφέραμε να ρυθμίσουμε το prex.

Διαδικασία Ρύθμισης Prex (Συνοπτικά):

Αρχικά, εγκαταστήσαμε τα [Debian](#) - 32 bit στο [Virtual Box](#) το οποίο έτρεχε από περιβάλλον Windows 10 – 64bit.

Έπειτα, εγκαταστήσαμε μια παλαιότερη έκδοση του gcc, συγκεκριμένα την 4.1. Ο λόγος είναι ότι το prex δεν είναι συμβατό με τις νεότερες εκδόσεις.

Ακολούθως, κατεβάσαμε την έκδοση του prex 0.9.0. και ρυθμίσαμε το system target στο οποίο θα τρέχει.

Μετά, έπρεπε να δημιουργήσουμε ένα αρχείο stopwatch.c (ή οτιδήποτε άλλο όνομα) στο οποίο θα υπάρχει ο εκτελέσιμος κώδικας, και αφού δημιουργήσαμε και το Makefile του, ορίσαμε στα tasks του prex το stopwatch. Το αρχείο αυτό μαζί με το makefile, τοποθετήθηκαν στο φάκελο prex/usr/samples.

Επειδή όμως η έκδοση του prex 0.9.0 δεν περιλάμβανε το κατάλληλο αρχείο img, χρειάστηκε να πάρουμε το img από την έκδοση 0.8.0.

Compile & Run:

Για την εκτέλεση του stopwatch, χρησιμοποιήθηκε ο προσομοιωτής qemu. Για τον προσομοιωτή, χρειάστηκε να ορίσουμε το path στο drive του (drive a), τέτοιο ώστε να “βλέπει” το img του prex που κατεβάσαμε από την έκδοση 0.8.0.

Εφόσον ολοκληρώθηκε η ρύθμιση του qemu, κάνουμε compile με τις εντολές

```
make clean
```

```
make.
```

Αυτό παράγει το binary αρχείο με όνομα prexos στο root directory του prex. Ακολούθως, φορτώνουμε το prexos στον προσομοιωτή qemu με την εντολή

```
mcopy prexos a:
```

Τώρα, το μόνο που μένει είναι να τρέξουμε το qemu, χρησιμοποιώντας την εντολή

```
qemu – system – i386 – localtime – fda ./prex – 0.8.0.i386 – pc.img
```

Η εντολή αυτή ανοίγει τον emulator και με αυτόν τον τρόπο μπορούμε να δούμε πως λειτουργεί ο κώδικας του αρχείου stopwatch.c.

Υλοποίηση Ρολογιού/Χρονόμετρου

Αρχικά, θα αναφέρουμε τις μεταβλητές και τις συναρτήσεις που χρησιμοποιήθηκαν για την ολοκλήρωση του προγράμματος.

Μεταβλητές που αφορούν το **ρολόι**:

- **sec, min, hour**: μεταβλητές με τις οποίες κρατάμε τον χρόνο του ρολογιού
- **sub_sec, add_min, add_hour**: μεταβλητές για την προσθήκη λεπτών/ωρών και τον μηδενισμό των δευτερολέπτων του ρολογιού.

Μεταβλητές που αφορούν το χρονόμετρο:

- ***pause_flag, reset_flag, stop_flag, stop_flag_lock***: Αυτές οι μεταβλητές έχουν τον ρόλο των σημαιών για το pause/reset/stop/start του ρολογιού.
- ***stopped_time, t***: Μεταβλητές για την αποθήκευση του χρόνου που είναι σταματημένο το χρονόμετρο καθώς και του χρόνου που θα εμφανίζεται στην οθόνη.
- ***stop_A, stop_B***: Μεταβλητές για τον υπολογισμό του χρόνου που είναι σταματημένο το χρονόμετρο.
- ***tic = 0, tac = 0***: Μεταβλητές για τον υπολογισμό του χρόνου που είναι ξεκινημένο το χρονόμετρο.

Συναρτήσεις:

- ***struct _timeval getTime(void)***: η συνάρτηση αυτή επικοινωνεί με το rtc και επιστρέφει το epoch σε δευτερόλεπτα. Το νούμερο το οποίο επιστρέφει είναι είτε *ms* είτε *μs* (*.tv_sec* και *.tv_usec* αντίστοιχα) που έχουν περάσει από τις 00:00:00 01/01/1970, γνωστό και ως [unix epoch](#).
- ***int tcsetattr(..), int tcgetattr(..), void mode(..)***: οι συναρτήσεις αυτές τροποποιήθηκαν και εγγράφηκαν εκ νέου στο αρχείο του κώδικα μας, διότι οι ήδη υλοποιημένες συναρτήσεις που βρίσκονται στη βιβλιοθήκη *termios* δεν επιτρέπουν την ανάγνωση χαρακτήρα χωρίς να πατηθεί το κουμπί *enter*, με αποτέλεσμα την εισαγωγή καινούριας γραμμής που οδηγούσε στο scroll της οθόνης.
- ***void displayClock(void)***: η συνάρτηση αυτή είναι υπεύθυνη για την ορθή προβολή του ρολογιού στην οθόνη. Μέσα στη συνάρτηση αυτή, υπολογίζεται η εκάστοτε χρονική στιγμή χρησιμοποιώντας την συνάρτηση *getTime* που αναφέρθηκε πιο πάνω. Επίσης, μέσω των μεταβλητών *add_min* και *add_hours* προσθέτουμε στο ρολόι τα λεπτά και τις ώρες ανάλογα με το πόσες φορές πατήθηκαν τα κουμπιά 'M' και 'H' αντίστοιχα, και μέσω της μεταβλητής *sub_sec*

αφαιρούμε τα δευτερόλεπτα που έχουν περάσει από την αρχή του λεπτού, ούτως ώστε να μηδενίζονται τα δευτερόλεπτα με το πάτημα του κουμπιού 'Z'. Οι ώρες μηδενίζονται στο 24 και τα λεπτά στο 60.

- ***void displayStopwatch(void)***: η συνάρτηση αυτή είναι υπεύθυνη για την ορθή προβολή του χρονόμετρου στην οθόνη, ανάλογα με το ποια κουμπιά έχουν πατηθεί και σε τι κατάσταση βρίσκεται το χρονόμετρο όταν πατηθούν συγκεκριμένα κουμπιά.

Για την αναπαράσταση του χρονόμετρου, χρησιμοποιήθηκε η συνάρτηση *sys_time(&time)* της οποίας η υλοποίηση βρέθηκε στα παραδείγματα του *prex*, και συγκεκριμένα στο παράδειγμα με το όνομα *alarm*. Ο λόγος που δεν χρησιμοποιήθηκε η *getTime*, είναι επειδή σε κάθε καινούριο δευτερόλεπτο που περνάει, τα χιλιοστά του δευτερολέπτου μηδενίζονται και μετράνε ξανά από την αρχή μέχρι το επόμενο δευτερόλεπτο. Οπότε, ανάλογα με την χρονική στιγμή που πατάμε το 'S' για να ξεκινήσουμε το χρονόμετρο, τα χιλιοστά του δευτερολέπτου θα βρίσκονται μεταξύ [0,999] και το χρονόμετρο θα ξεκινάει από λάθος στιγμή, δείχνοντας πολλές φορές ένα δευτερόλεπτο μπροστά μετά από μόλις μερικά κλάσματα δευτερολέπτου. Μια λύση θα ήταν να κρατούσαμε σταθερή την ώρα έναρξης του χρονομέτρου και σε κάθε επανάληψη να την αφαιρούσαμε από την εκάστοτε ώρα, δηλαδή *current_time - start_time*, παρόλα αυτά αποδείχτηκε λάθος σκεπτική διότι εφόσον το *start_time* είναι σταθερό και το *current_time* αλλάζει συνεχώς στο διάστημα [0,99], τότε θα προκύπτουν αρνητικές τιμές. Έτσι, χρησιμοποιήθηκε η *sys_time* η οποία λειτουργεί με πανομοιότυπο τρόπο με την *tic/toc* του Matlab.

Μόλις καλούμε τη συνάρτηση *displayStopwatch*, γίνεται έλεγχος αν έχει πατηθεί το 'S' για stop του χρονόμετρου και αν ναι, παγώνουμε την οθόνη στη συγκεκριμένη μέτρηση και κρατάμε τον χρονική στιγμή που πατήθηκε το κουμπί από την *sys_time* στην μεταβλητή *tic*. Μόλις ξαναπατηθεί το 'S' για εκκίνηση του χρονόμετρου, κρατάμε από την *sys_time* τον χρόνο στην μεταβλητή *tac*. Αφαιρώντας *tac - tic* βρίσκουμε την χρονική διάρκεια που το χρονόμετρο ήταν σταματημένο, και έτσι αφαιρώντας τον χρόνο που ήταν σταματημένο από τον χρόνο που θα βρισκόταν αν δεν το σταματούσαμε, το χρονόμετρο συνεχίζει κανονικά από εκεί που σταμάτησε.

Στην περίπτωση του κουμπιού 'P', τότε παγώνουμε την συγκεκριμένη χρονική στιγμή που δείχνει η οθόνη, ενώ το χρονόμετρο συνεχίζει να τρέχει, ούτως ώστε όταν ξαναπατηθεί το κουμπί να συνεχίσει από τον τρέχων χρόνο.

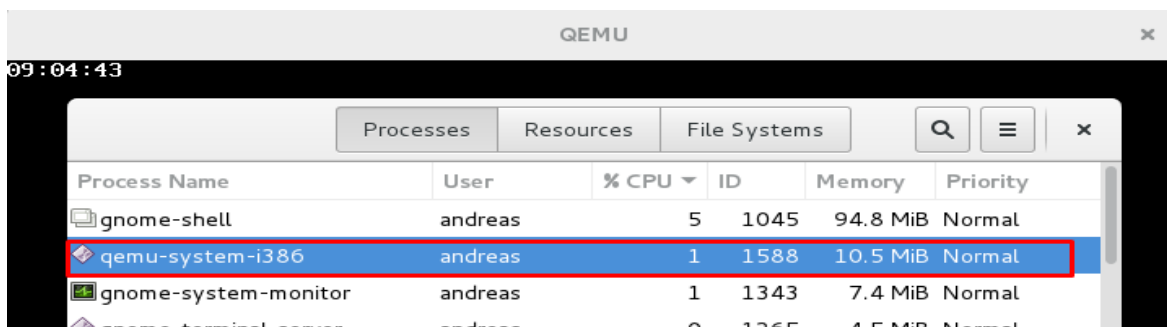
Τέλος, όταν πατηθεί το 'R' μηδενίζουν οι μετρητές και οι μεταβλητές του χρόνου, και οι σχετικές σημαίες (*flags*) επανέρχονται στις αρχικές τους τιμές, εκτός από τις σημαίες του Start/Stop και του Pause. Έτσι, το Reset μπορεί να γίνει ανά

πάσα στιγμή χωρίς όμως να αλλάξει η λειτουργία του χρονόμετρου, παρά μόνο να το μηδενίσει. Επίσης τα Start/Stop και Pause δεν μπορούν να εκτελεστούν ταυτόχρονα διότι θεωρήσαμε ότι δεν έχει νόημα κάτι τέτοιο, αλλά το Start μπορεί να ενεργοποιηθεί ενώσω βρισκόμαστε σε λειτουργία ρολογιού, διότι θα είχε νόημα κάποιος να θέλει να ξεκινήσει το χρονόμετρο βάσει συγκεκριμένης ώρας.

- ***void main(void)***: Είναι η κύρια συνάρτηση στην οποία τρέχει το πρόγραμμα. Εντός της *main* γίνονται όλες οι αρχικοποιήσεις μεταβλητών, και ακολούθως εισέρχεται σε μια ατέρμονη λούπα. Εντός της λούπας, γίνεται έλεγχος για το πάτημα κάποιου κουμπιού και σε περίπτωση που πατηθεί κάποιο κουμπί αλλάζουν κατάσταση οι εκάστοτε σημαίες που αφορούν το κουμπί που πατήθηκε. Επίσης, εντός της λούπας «κοιμίζουμε» τον επεξεργαστή εισάγοντας την εντολή ***timer_sleep(10,0)***, όπου το 10 αντιστοιχεί σε *ms*. Με αυτό τον τρόπο, πετυχαίνουμε λιγότερη κατανάλωση του συστήματος και χαμηλότερη χρήση της *cpu*.

Κατανάλωση Ενέργειας

Αρχικά, όταν ελέγξαμε το πρόγραμμα ως προς την κατανάλωση ενέργειας βλέποντας την χρήση του επεξεργαστή, διαπιστώσαμε πως το πρόγραμμα χρησιμοποιούσε τον επεξεργαστή 100%. Αυτό οφειλότανε στον ατέρμονο βρόχο, όπου το πρόγραμμα έκανε συνεχώς πράξεις υπολογίζοντας τον χρόνο είτε του ρολογιού είτε του χρονόμετρου. Πρακτικά όμως, όλες αυτές οι πράξεις είναι αχρείαστες εφόσον πολλές από αυτές ήταν ίδιες. Επίσης, όσον αφορά τα *ms* του χρονόμετρου, τυπώνουμε στην οθόνη μόνο 2 ψηφία αντί 3, επομένως ο υπολογισμός της εκατοντάδας (του ενός ψηφίου που δεν εκτυπώνουμε) είναι αχρείαστος. Έτσι, όπως αναφέρθηκε και πιο πάνω στην επεξήγηση της *main*, τοποθετήθηκε η εντολή *timer_sleep* για 10 *ms* σε κάθε επανάληψη του βρόχου. Με αυτό τον τρόπο, η χρήση της *cpu* έπεσε κατακόρυφα τείνοντας προς το μηδέν. Αυτό μπορεί να φανεί και από το παρακάτω screenshot.



Process Name	User	% CPU	ID	Memory	Priority
gnome-shell	andreas	5	1045	94.8 MiB	Normal
qemu-system-i386	andreas	1	1588	10.5 MiB	Normal
gnome-system-monitor	andreas	1	1343	7.4 MiB	Normal
gnome-terminal-server	andreas	0	1265	4.5 MiB	Normal

Προκύπτει λοιπόν, πως οι κατανάλωση ενέργειας είναι αρκετά χαμηλή και είναι επίσης ανάλογη με τον χρόνο στον οποίο είναι ενεργό το ρολόι/χρονόμετρο μας, καθιστώντας το, ίσως, ιδανικό για μικρές συσκευές με μικρή αποθηκευτική ενέργεια.

Ο λόγος που επιλέχθηκε το 10 ms είναι διότι για περισσότερα ms (>10) χάνεται πληροφορία στην εκτύπωση των ms του χρονόμετρου. Εναλλακτικά, για περισσότερη εξοικονόμηση ενέργειας θα μπορούσαμε να «κοιμίζουμε» τον επεξεργαστή στα 10 ms όταν βρισκόμαστε σε λειτουργία χρονομέτρου και στα 100 ms όταν βρισκόμαστε σε λειτουργία ρολογιού. Επειδή όμως το αποτέλεσμα της κατανάλωσης ενέργειας για 10 ms γενικά ήταν ικανοποιητικό, κρατήθηκε το 10 ms για όλο το πρόγραμμα. Έτσι και αλλιώς, δεν έχουμε κάποια συσκευή (ή τουλάχιστον τις προδιαγραφές της) η οποία θα τρέχει το σύστημα μας έτσι ώστε να μπορούμε να πούμε με σιγουριά αν αυτό είναι εντάξει ή όχι από άποψη κατανάλωσης ενέργειας.

Απόκριση Συστήματος

Η απόκριση του συστήματος μπορεί να θεωρηθεί 10 ms λόγω του *sleep_timer* που αναφέραμε πιο πάνω. Επιπλέον, όταν τρέχουμε το πρόγραμμα και πατάμε συνεχόμενα διάφορα κουμπιά, δεν παρατηρείται καμία καθυστέρηση.

Για να το ελέγξουμε σίγουρα όμως, βάζουμε μια *sys_time(&t1)*, μόλις πατηθεί κάποιο κουμπί (π.χ. toggle), και μια *sys_time(&t2)* μόλις επιστρέψει στην αρχή της while μετά την κλήση της συνάρτησης *displayStopwatch* (ή *displayClock*), όπως φαίνεται και στην παρακάτω εικόνα.

```
/* entering loop */
while(1){
    device_open( "tty", 0, &ttyDevice );
    device_ioctl( ttyDevice, TIOCINQ, &input);
    device_close( ttyDevice);

    if (toggle_flag){
        sys_time(&t2);
        printf("\r\n%d",t2-t1);
        break;
    }

    if( input > 0 ){
        button = getchar();
        switch(button){
            case 't':
                toggle_flag=!toggle_flag;
                sys_time(&t1);
                break;
            case 'i':
```

Όταν τρέξουμε λοιπόν το πρόγραμμα και πατήσουμε 'T', παίρνουμε την ακόλουθο χρόνο:



Όντως σωστά υποθέσαμε, ο χρόνος απόκρισης μπορεί να θεωρηθεί 10ms λόγω του *sleep_timer*.

Παρατήρηση: Αφήνοντας το ρολόι να προχωρήσει για περίπου ένα λεπτό, παρατηρήθηκε ότι «μένει πίσω» κατά 1 δευτερόλεπτο σε σύγκριση με το κάποιο άλλο ρολόι, π.χ. κινητό τηλέφωνο.

Αντίστοιχα για το χρονόμετρο, όταν μετρήθηκε δίπλα-δίπλα με το χρονόμετρο του κινητού τηλεφώνου, παρατηρήθηκε ότι όταν σταματήσαμε το κινητό τηλέφωνο και το χρονόμετρο ταυτόχρονα, το κινητό βρισκόταν στο 00:01:00.26 ενώ το χρονόμετρο που υλοποιήσαμε ήταν στο 00:00:59.29. Αυτό πιθανότατα να συμβαίνει επειδή τρέχουμε το vm σε περιβάλλον Debian, το οποίο με τη σειρά του τρέχει σε περιβάλλον Windows μέσω του προγράμματος Virtual Box.

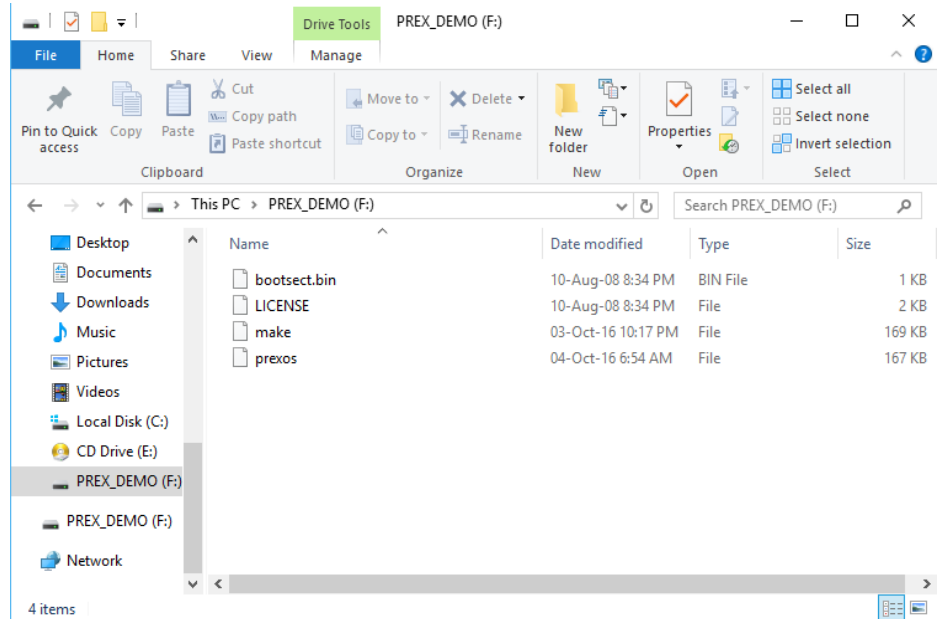
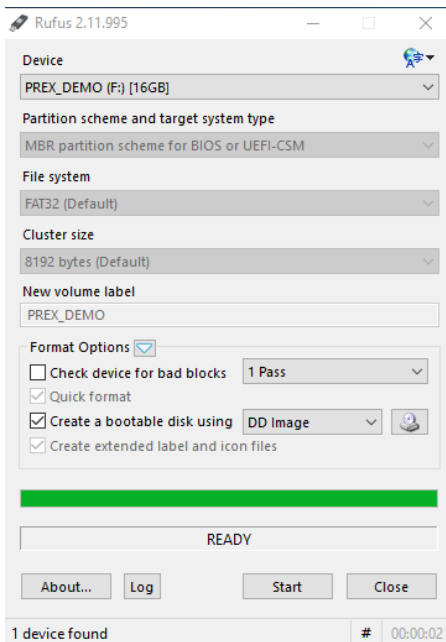
Εντούτοις, όταν τρέξαμε το ρολόι/χρονόμετρο κάνοντας boot από ένα usb stick (η διαδικασία εξηγείται αμέσως παρακάτω), παρατηρήθηκε ότι η διαφορά του πραγματικού ρολογιού από το δικό μας μειώθηκε προς το 0, με τις διαφορές να οφείλονται, ίσως, σε ανθρώπινους παράγοντες (πάτημα start/stop σχεδόν ταυτόχρονα).

Bootable Disk

Για να δημιουργήσουμε το bootable disk, όπως αναφέρθηκε και στις πρώτες σελίδες, χρειαζόμαστε το img αρχείο. Ακολούθως, χρειαζόμαστε και ένα πρόγραμμα το οποίο να έχει τη δυνατότητα να φορτώσει το img σε ένα flash drive και να δημιουργήσει το bootable disk. Για καλή μας τύχη, το πρόγραμμα το οποίο χρησιμοποιούμε και για άλλα bootable disks (.iso), έχει τη δυνατότητα να δημιουργήσει bootable disk και από .img αρχεία, αναγνωρίζοντας τα ως DD OS, δημιουργώντας και το bootsector αρχείο που χρειάζεται για να γίνει το boot. Το πρόγραμμα αυτό είναι το [rufus](#). Έτσι, δημιουργώντας το bootable disk, μπορέσαμε να bootάρω τον υπολογιστή μας από το usb το οποίο είχε το ρολόι/χρονόμετρο που φτιάξαμε.

Το αρχείο img που προέκυψε είχε μέγεθος: 1.40 MB (1,474,560 bytes)

Παρακάτω, παρουσιάζονται κάποια screenshots ως απόδειξη της δημιουργίας του bootable disk.



ΕΥΧΑΡΙΣΤΩ