Ενσωματωμένα Συστήματα & Συστήματα Πραγματικού Χρόνου
Εργασία 1
Πετρίδης Περικλής 8896

1. Εισαγωγή

Στην εργασία αυτή ζητήθηκε να υλοποιήσουμε δύο προγράμματα δειγματοληψίας σε C. Συγκεκριμένα, τα δύο πρόγραμματα θα πρέπει να λαμβάνουν και να αποθηκεύουν τα **timestamps** κάθε **dt δευτερόλεπτα**, και για συνολικά **t δευτερόλεπτα**. Τα **t** και **dt** δίνονται ως **είσοδοι** στα προγράμματα.
Επιπλέον, τα δύο προγράμματα θα πρέπει να είναι οικονομικά ως προς την CPU και την ενέργεια που καταναλώνουν, οπότε είναι σκόπιμο να παραμένουν ανενεργά ανάμεσα στις δειγματοληψίες. Αυτό γίνεται με την χρήση μιας παραλλαγής της συνάρτησης sleep.
Η διαφορά ανάμεσα στα δύο προγράμματα είναι ότι το ένα θα μπορεί να χρησιμοποιεί την τιμή του timestamp ώστε να **διορθώνει** την χρονική ολίσθηση που προκαλεί η ανακρίβεια του sleep, ενώ το δεύτερο πρόγραμμα όχι.

2. Υλοποίηση

Και στα δύο προγράμματα χρησιμοποιείται μια παραλλαγή της συνάρτησης **sleep** ενδιάμεσα από δειγματοληψίες του timestamp και αποθηκεύεται εν τέλει η απόσταση μεταξύ των δειγμάτων, καθώς και η απόκλιση από την θεωρητική, σε δύο αρχεία.

Το ένα αρχείο (complete) είναι “πλήρες” και έχει όλες τις πληροφορίες και text ενώ το άλλο (lean) περιέχει μόνο τις τιμές των αποκλίσεων από την θεωρητική τιμή της χρονικής απόστασης μεταξύ δύο δειγμάτων.

Τελικά, όλα τα αποτελέσματα αναλύονται με matlab που βγάζει τα στατιστικά και σχεδιάζει τα γραφήματα.

2.1 Είσοδοι

Οι είσοδοι στα προγράμματα είναι
1) συνολικός χρόνος **t**
2) χρονικό βήμα δειγματοληψίας **dt**
3) το επιθυμητό όνομα των αρχείων στα οποία θα αποθηκευτούν τα αποτελέσματα.

2.2 Δειγματοληψία και Συνάρτηση Sleep

Επέλεξα να χρησιμοποιήσω sleep αντί για interrupt επειδή θεώρησα ότι τα αποτέλεσματα ήταν αρκετά ικανοποιητικά.

```
void sleep_ms(int milliseconds){
struct timespec ts;
    ts.tv_sec = milliseconds / 1000;
    ts.tv_nsec = (milliseconds % 1000) * 1000000;
    nanosleep(&ts, NULL);
}
```

Για μείωση της κατανάλωσης ισχύος χρησιμοποιήθηκε η nanosleep από την βιβλιοθήκη time.h.
Συγκεκριμένα υλοποίησα την βοηθητική συνάρτηση sleep_ms η οποία ως όρισμα παίρνει μόνο πόσα millisecond θέλουμε να γίνει sleep.

2.3 Διόρθωση time shift

Στο πρόγραμμα στο οποίο μπορούσαμε να χρησιμοποιήσουμε διόρθωση της δειγματοληψίας, χρησιμοποίησα τον αλγόριθμο που φαίνεται στην εικόνα:

```
//CORRECTION -- this is the ONLY difference with timestampsWithout.c
if(time_shift_accum*1000>1){
  correction = 1-(time_shift_accum*1000);
}else if(time_shift_accum*1000<-1){
  correction = -(time_shift_accum*1000);
}//End CORRECTION
```

Συγκεκριμένα, υπολογίζοντας την διαφορά του χρόνου που πέρασε (μέσω timestamps) από τον θεωρητικό χρόνο που θα έπρεπε να έχει περάσει (αριθμος δειγμάτων * βήμα δειγματοληψίας), αν η διαφορά τους ήταν πάνω από 1 millisecond, τότε γινόταν επέμβαση ώστε η επόμενη περίπτωση sleep_ms να κοιμηθεί για περισσότερο ή λιγότερο χρόνο.

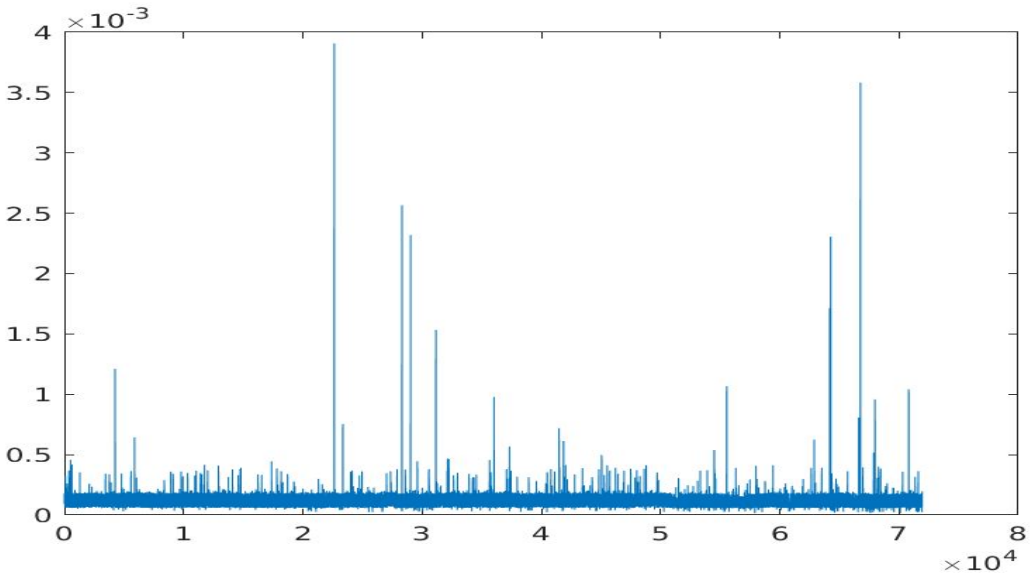
3. Αποτελέσματα

Τα αποτελέσματα βγήκαν μέσω matlab script, από το .csv αρχείο με τις χρονικές αποκλίσεις μεταξύ των δειγμάτων.

3.1 Time-shift μεταξύ διαδοχικών δειγμάτων

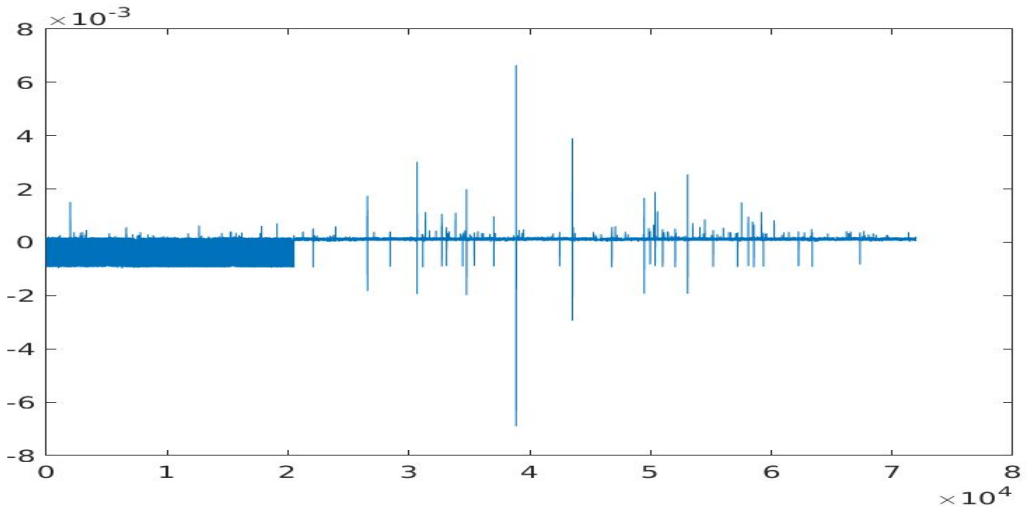
Στα παρακάτω γραφήματα φαίνεται το time-shift από το θεωρητική χρονικό βήμα μεταξύ των δειγμάτων.

Χωρίς διόρθωση:
Βλέπουμε πως το time-shift είναι πάντα θετικό, δηλαδή στην πραγματικότητα περνάει περισσότερος χρόνος μεταξύ δειγμάτων από το θεωρητικό βήμα **dt**.



Με διόρθωση:
Παρατηρούμε δύο πράγματα εδώ.
Καταρχάς εδώ το time-shift παίρνει και αρνητικές τιμές. Αυτό οφείλεται στην διόρθωση του time-shift, η οποία είναι πάντα αρνητική εφόσον όπως είδαμε στο προηγούμενο γράφημα, χωρίς διόρθωση έχουμε πάντα μια μικρή επιπρόσθεση καθυστέρηση ανάμεσα στα δείγματα.

Το άλλο που παρατηρούμε είναι ότι μερικές φορές η διόρθωση μπορεί να είναι σχετικά μεγάλη (~3-4 millisecond), και προκαλεί μία φθίνουσα ταλάντωση εώς ότου ο “εικονικός” χρόνος συμπίπτει με τον πραγματικό χρόνο εντός <1 millisecond.



3.2 Στατιστικές δειγμάτων

Από matlab (μεγέθη σε τάξη millisecond):

	1 Mean	2 StandardDeviation	3 Median	4 Max	5 Min
1 Without timestamps	0.1137	0.0482	0.1050	3.9070	0.0160
2 With timestamps	0.0730	0.2136	0.0990	6.6450	-6.9060

Συνολικό σφάλμα με διόρθωση: 0.5~ milliseconds

```
virtual_time = 7199.973145 ---- real_time = 7199.973633
```

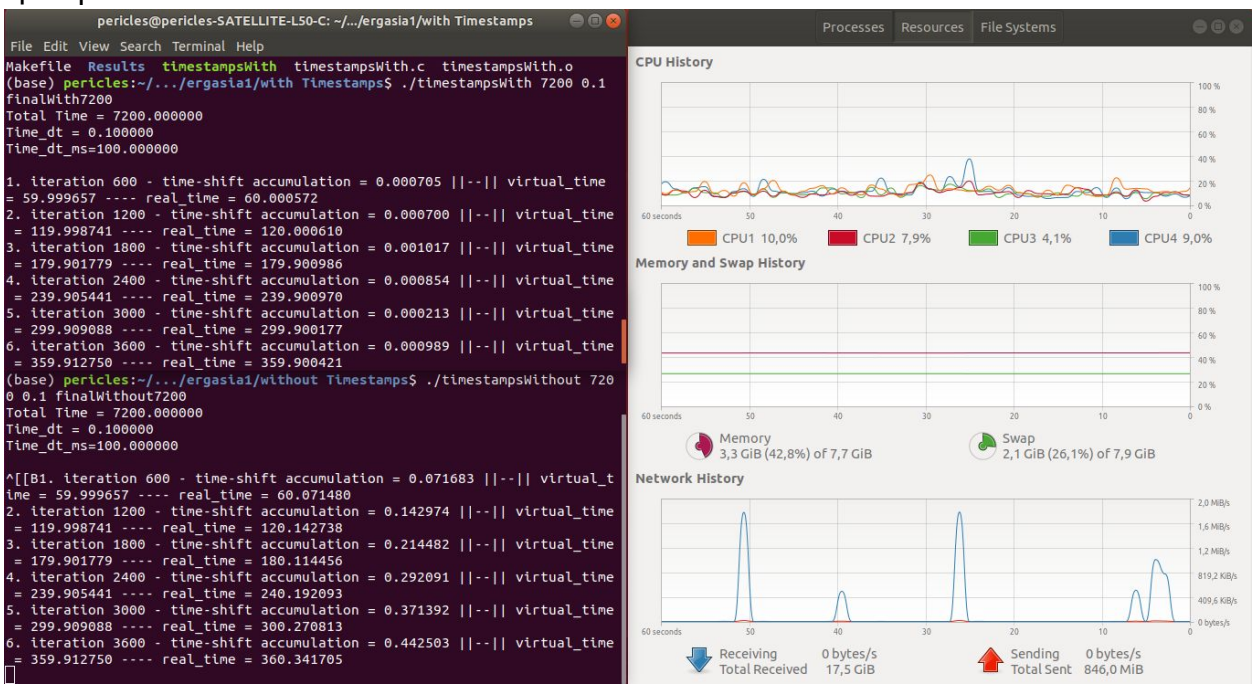
Συνολικό σφάλμα χωρίς διόρθωση: ~2.88 seconds

```
virtual_time = 7199.973145 ---- real_time = 7202.876953
```

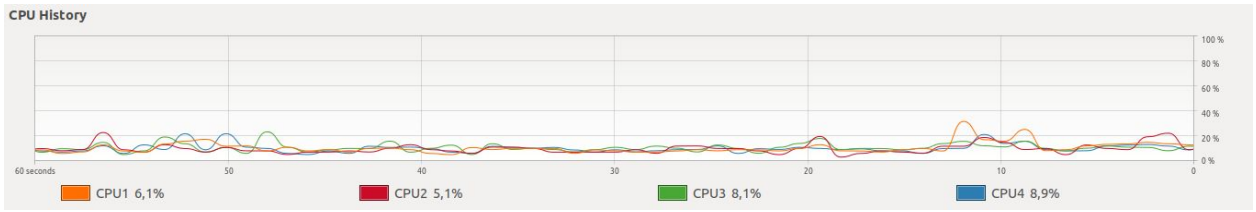
3.3 Χρήση CPU

Στις παρακάτω εικόνες φαίνεται η χρήση της CPU καθώς τρέχουν και τα 2 πειράματα (με και χωρίς χρήση timestamp) ήδη για 6 λεπτά, και όταν δεν τρέχει τίποτα για σύγκριση.

Είναι εύλογο να θεωρήσω πως η χρήση της CPU είναι ικανοποιητικά μικρή εώς αμελητέα.



Ο ίδιος υπολογιστής χωρίς να τρέχουν τα 2 προγράμματα για σύγκριση:



Github: https://github.com/PericlesPet/RTES-Sampling