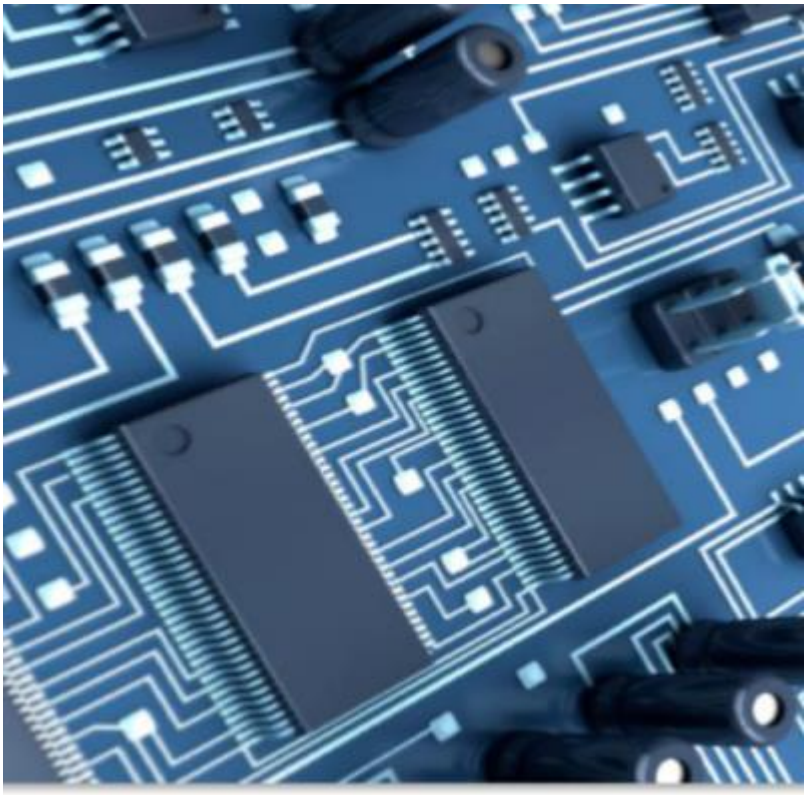
ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ



3η ΕΡΓΑΣΙΑ

ΓΚΟΥΖΙΩΚΑΣ ΙΩΑΝΝΗΣ
ΑΕΜ 8127

ΑΝΑΦΟΡΑ

Στην παρούσα εργασία δημιουργήθηκε πρόγραμμα για μία ενσωματωμένη συσκευή (zSun) που συλλέγει πληροφορίες για τα διαθέσιμα Wi-fi.
Το μοντέλο που χρησιμοποιήθηκε βασίστηκε σε αυτο του producer-consumer που δοθηκε.

Περιγραφη του μοντελου

Ο producer “ξυπνάει” για να βρει διαθεσιμα wi-fi (σκαναρει)

ανάλογα με την περιοδο δειγματοληψίας που δίνει ο χρήστης απο το πληκτρολόγιο βάζοντας τα αποτελέσματα (wi-fi & timestamps) στην ουρα(όσο η ουρά δεν είναι γεμάτη) (fifo) σε ένα δι-σδιάστατο πίνακα (buf) τυπου queue.Ο consumer με το που λαμβάνει signal ότι η ουρά έχει στοιχεία και δεν είναι άδεια(notEmpty) αφαιρεί στοιχεία από την ουρά!

<u>Περιγραφη κώδικα</u>

Το πρόγραμμα είναι σχεδιασμένο να τρέχει για πάντα...καθώς και ο producer και ο consumer βρίσκονται σε ατέρμων βρόγχο με τη διαφορά ότι ο consumer ζητάει συνεχώς στοιχεία ενώ ο producer ξυπνάει να παράγει ανάλογα με την περίοδο δειγματοληψίας.
Τα δύο νήματα δημιουργούνται στη main....ο producer ξυπνάει για να παράγει και καταγραφει τα αποτελέσματα του searchwifi (δηλαδή τα SSIDs) μάζι με τα timestamps σε ένα δισδιάστατο πίνακα buf τυπου queue (που είναι το struct που δημιουργήθηκε για την υλοποίηση του μοντέλου) ενώ ταυτόχρονα κλειδώνει το mutex καθώς ο δισδιάστατος πίνακας buf προσπελάζεται και από τον consumer…στη συνέχεια καλείται η queueAdd η οποία αυξάνει το tail (δλδ αλλάζει γραμμή στον πίνακα buf) ώστε να καταγραφεί το επόμενο ssid από το searchwifi ενώ ταυτόχρονα τσεκάρει αν γέμισε η ουρά. Μετά την καταγραφή ενός στοιχείου στον πίνακα και την κλήση της queueAdd ο producer ξεκλειδώνει το mutex ενώ ταυτόχρονα στέλνει σήμα στον consumer μέσω της cond_signal ότι η ουρά δεν είναι άδεια πλέον και ότι μπορέι να καταναλώσει βγαίνοντας έτσι από την wait που βρισκόταν
Ο πίνακας buff μετά την πρώτη σάρωση θά είναι κάπως έτσι:
1η γραμμή: “Hol-Zte-4” Timestamp= (χρόνος)
2η γραμμή: “Ote-Wifi” Timestamp=(χρόνος)
3η γραμμή: “katoxospito” Timestamp=(χρόνος)
……..
Και ούτο καθεξής ανάλογα και πόσα wifi βρήκε το script…

Εάν ο consumer έχει πάρει κάποιο στοιχείο από τον πίνακα τότε ο πίνακας θα είναι:

“Ote-Wifi” Timestamp=(χρόνος)
“katoxospito” Timestamp=(χρόνος)
……
(Δηλαδή ο consumer είναι ελεύθερος οποιαδήποτε στιγμή υπάρχει κάποιο στοιχείο στην ουρά να το καταναλώσει)
Αφού ο producer καταγράψει τα SSID και τα timestamp της σάρωσης μέσω της usleep κοιμάται (το χρόνο της περιόδου δειγματοληψίας – το χρόνο (eltime) που πέρασε ώστε να κάνει όλες αυτες τις ενέργειες(σκαν και καταγραφη)) ώστε να ξυπνήσει για την επόμενη σάρωση!!!
*****Το eltime αφαιρείται από τον χρόνο της περιόδου ώστε να μην υπάρχει συσώρευση σφάλματος και να μη χάνονται δείγματα**

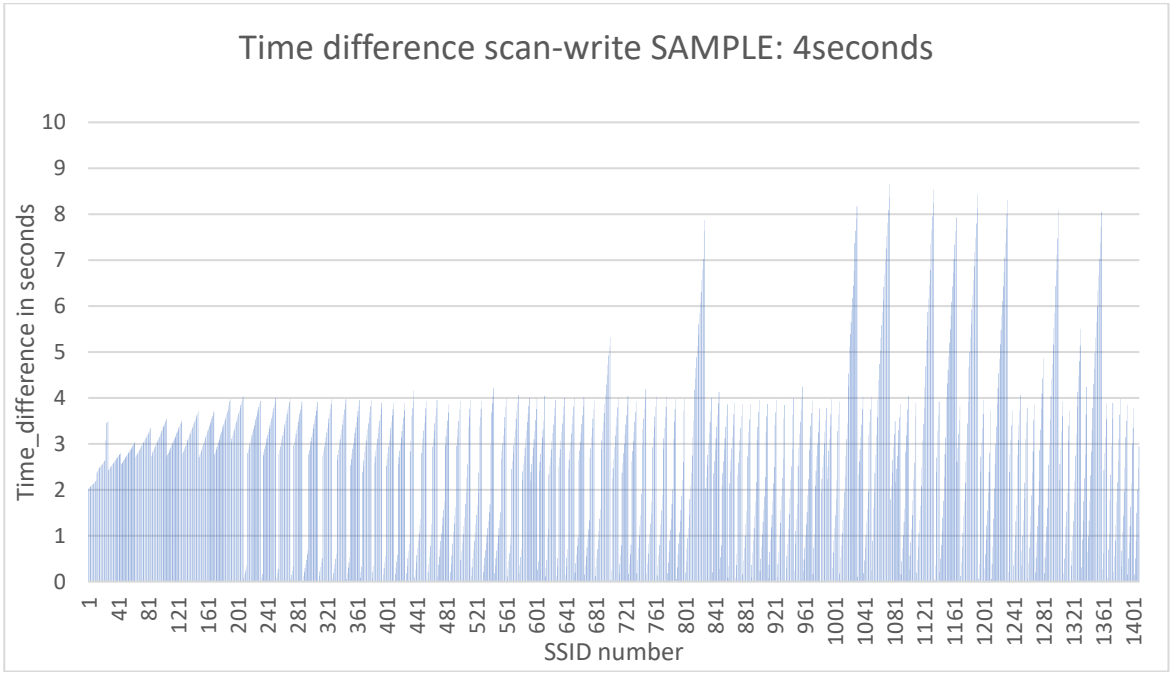
Ο consumer μέσω της cond_wait περιμένει από τον producer (καθώς η ουρά είναι άδεια) να βάλει στοιχεία στον πίνακα ώστε μέσω της signal από τον producer να ξεκλειδώσει και να καταναλώσει...αντιγράφει σε ένα temporary πίνακα μία γραμμή κάθε φορά από τον buf (ουρά) καλώντας την queueDel ώστε να αυκήσει το head για να τσεκάρει εάν έφτασε στο τέλος και ξεκλειδώνει το mutex (τον buf τον διαχειρίζεται και ο producer) ενώ στέλνει παράλληλα και σήμα μέσω της cond_signal ότι η ουρά σίγουρα δεν είναι γεμάτη αφού αφαίρεσε ένα στοιχείο...έτσι ώστε ο producer να ξεκλειδώσει από την δική του wait και να παράξει και άλλα στοιχεία για την ουρά!
τον temporary πίνακα(μία γραμμή από τον buf) τον στέλνει σαν όρισμα στην συνάρτηση Wifiwriteargs στην οποία σαρώνεται το ήδη υπάρχον αρχείο και αντιγράφεται σε ένα αντίγραφο αρχείο έπειτα απομονώνεται το όνομα(SSid) και ο χρόνος (timestamp) γίνεται σύγκριση εάν υπάρχει το (SSid) τότε γράφεται μόνο το timestamp από την γραμμή που στάλθηκε σαν όρισμα από τον

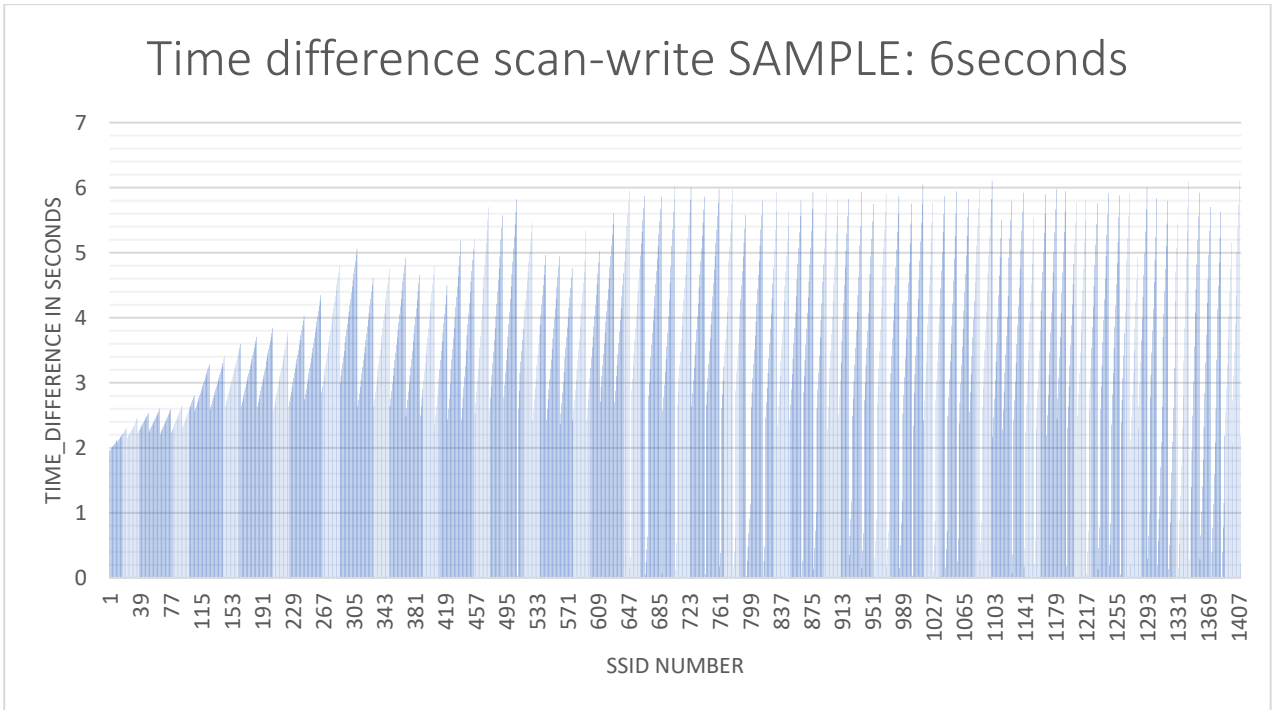
consumer αλλιώς γράφονται και τα δύο Ssid & Timestamp...και τέλος ξαναγράφεται όλο το προηγούμενο αρχείο στο αρχικό μεσω του αντίγραφου αρχείου!

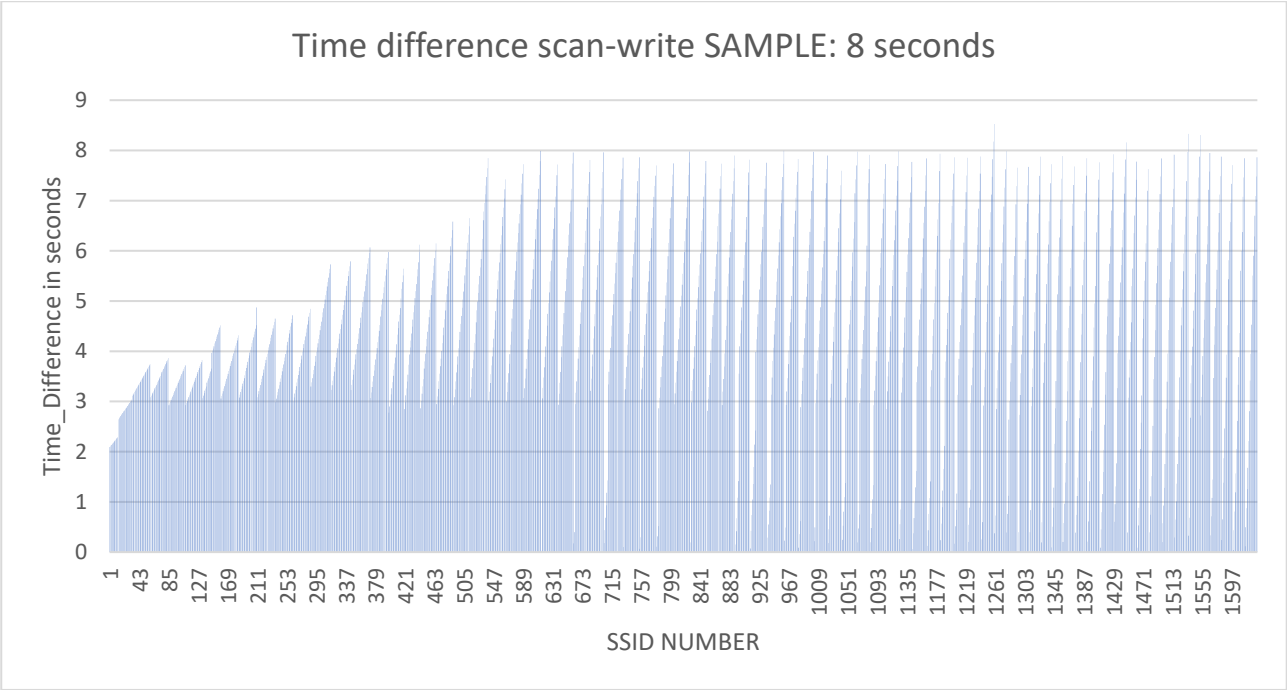
***ΣΗΜΕΙΩΣΗ (eltime):** Τα δείγματα που πάρθηκαν ήταν για 4s , 6s , 8s και όχι για μικρότερο χρόνο δειγματοληψίας γιατί όσο περνάει η ώρα ο το (eltime) που είναι ο χρόνος που περνάει μέσα στον producer (σάρωση searchwifi καταγραφή στον buf) γίνεται όλο και μεγαλύτερος (λόγω αρχείου) με αποτέλεσμα κάποια στιγμή να γίνει μεγαλύτερος από την περίοδο δειγματοληψίας και έτσι το όρισμα της usleep (period-eltime) να βγει αρνητικό....στην περίπτωση αυτή σημειώνεται ότι έχει χάνεται ένα δείγμα και ο producer κοιμάται για τον χρόνο της περιόδου ώστε να πάρει το επόμενο δείγμα(σάρωση)....Όσο πιο μικρή ήταν η περίοδος που ξυπνάει ο producer τόσο πιο πολλά δείγματα χανόταν και μάλιστα για περίοδο 1s δεν ήταν δυνατό να πραγματοποιηθεί η σάρωση καθώς μέσα στον producer καταναλωνόταν μονίμως χρόνος μεγαλύτερος από 1s!!!

Διαφορά χρόνου μεταξύ της σάρωσης και της εγγραφής

Παρακάτω παρουσιάζονται τα διαγράμματα διαφοράς χρόνου δηλ από τη στιγμη που έγινε η σάρωση στον producer μέχρι την εγγραφή των αποτελεσμάτων στο αρχείο στη συνάρτηση Wifiwriteargs....χρησιμοποιήθηκε η gettimeofday πριν την σάρωση στον producer και ο χρόνος αυτός αφαιρέθηκε από τον χρόνο που έδωσε η gettimeofday πριν την εγγραφή των στοιχείων(timestamp) στο αρχείο
Τα διαγράμματα είναι για 4 6 και 8 second περίοδο αντίστοιχα!!!


Time difference scan-write SAMPLE: 4seconds
Time_difference in seconds
SSID number


Time difference scan-write SAMPLE: 6seconds
TIME_DIFFERENCE IN SECONDS
SSID NUMBER



<u>Παρατηρήσεις!!!</u> Από το διάγραμμα διαφορών χρόνου φαίνεται ότι ο τιμή του χρόνου **αυξάνεται σταδιακά** πράγμα λογικό καθώς αυξάνεται και το μέγεθος του αρχείου με συνέπεια να παίρνει μεγαλύτερο χρόνο η όλη διαδικασία(αναζήτηση-αντιγραφή)

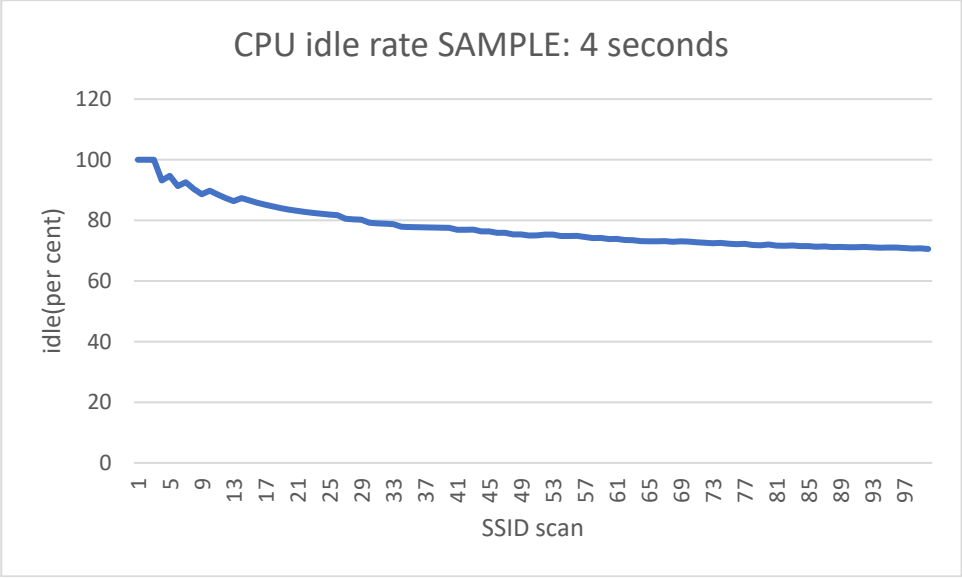
<u>Χρόνος που μένει αδρανής η CPU</u>

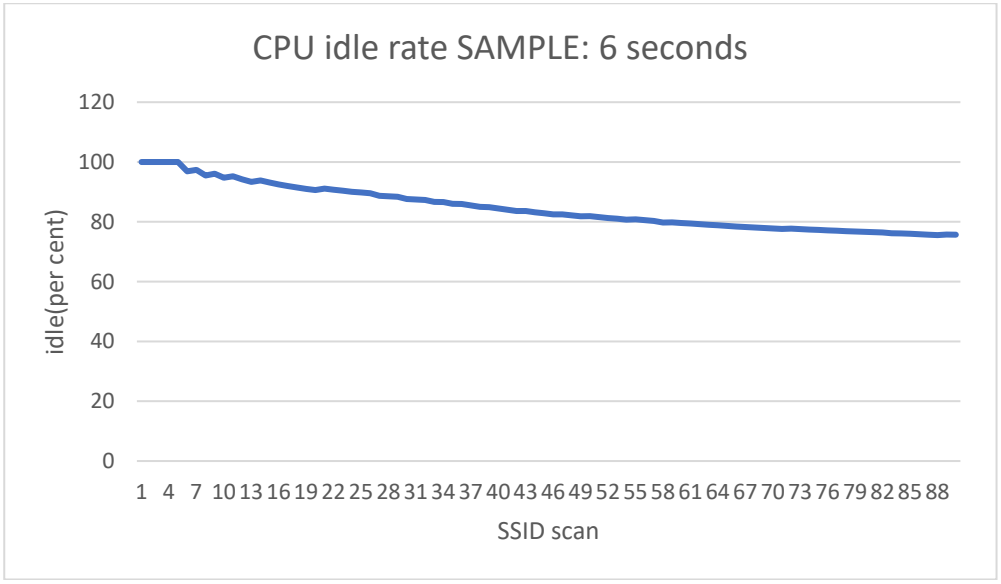
<u>Για την μέτρηση του ποσοστού χρόνου που παραμένει αδρανής η CPU:</u>

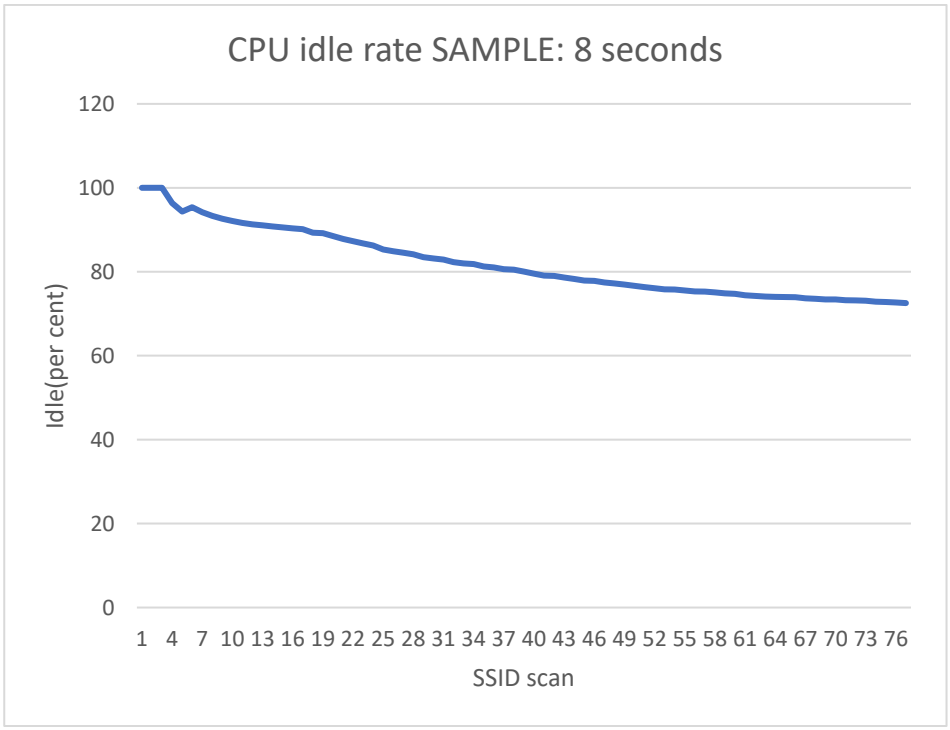
Χρησιμοποίησα την gettimeofday(&startforCPU,NULL) για να παρω το χρόνο στην αρχή της main καθώς και την

start_t = clock() που μου επιστρεφει τους χτυπους....και επίσης την gettimeofday(&endforCPU,NULL) για να πάρω το χρόνο στο τέλος του producer δλδ πριν κοιμηθεί με την usleep για την επόμενη σάρωση καθώς και την end_t=clock() για τους χτύπους εκείνη τη στιγμή....με την total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC; Μετέτρεψα τους χτύπους σε second για την CPU μέχρι εκείνο το σημείο(πριν κοιμηθεί ο producer για την επόμενη σάρωση)
στην tempCPU αποθήκευσα τον χρόνο που έχει μεσολαβήσει από την αρχή του προγράμματος μέχρι εκείνο το σημείο (χρόνος απο τις gettimeofday) και στην totaltime αποθήκευσα το ποσοστό αδράνειας όπως φαίνεται παρακάτω....
 totaltime = (tempCPU - total_t)/tempCPU * 100;

Τα αποτελέσματα τα έγραψα σε ένα αρχείο (totalCPU) και στη συνέχεια έκανα τα διαγράμματα!!!







ΠΑΡΑΤΗΡΗΣΕΙΣ!!!

Επειδή μιλάμε για ενσωματωμένο σύστημα τα δείγματα έχουν παρθεί στο zsun θέλουμε όσο το δυνατόν πιο μεγάλη αδράνεια για την CPU….η αδράνεια αυτήν μειώνεται σταδιακά όπως φαίνεται και στα διαγράμματα καθώς είναι λογικό όσο μεγαλώνει το αρχείο να απαιτεί περισσότερη δουλειά απο την CPU….τα καλά ποσοστά

αδράνειας τα πετυχαίνουμε μέσω της usleep που ουσιαστικά αδρανοποιεί τον producer για όσο χρόνο δεν τον χρειαζόμαστε και ξυπνάει ανάλογα με την περίοδο δειγματοληψίας για να σκανάρει!

ΤΕΛΟΣ!!!