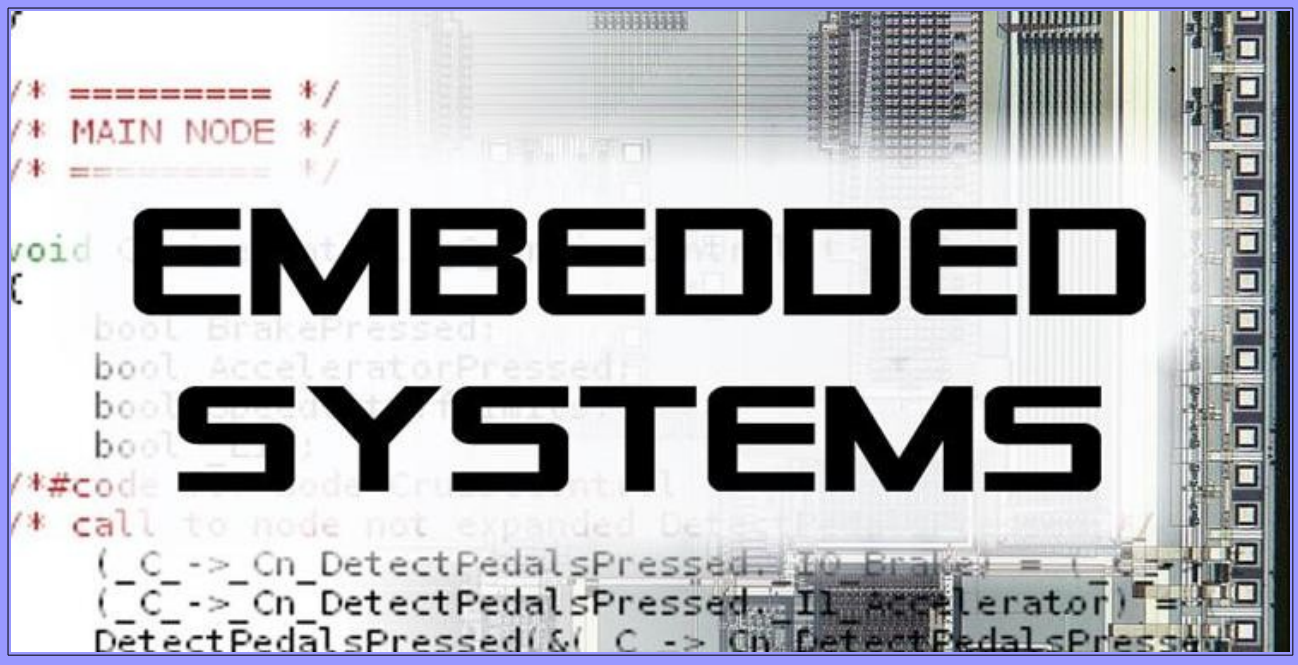
Ονοματεπώνυμο : Χρήστου Φίλιππος
ΑΕΜ : 8276

3η Εργασία στα Ενσωματωμένα συστήματα πραγματικού χρόνου



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης , Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, Τομέα Ηλεκτρονικής & Υπολογιστών

1 : Εισαγωγικά

Έχοντας ήδη εξικοιωθεί με την ιδέα του zSun και της πλατφόρμας OpenWrt , απαιτείται να αναπτυχθεί ένα πρόγραμμα σε γλώσσα C το οποίο αφού το φορτώσουμε στο zSun θα ανιχνεύει τα SSID των wifi γύρω του και θα σημειώνει την ώρα που τα βρήκε σε ένα ειδικά διαμορφωμένο αρχείο.

Για την σάρωση των SSID χρησιμοποιήθηκε το scriptάκι που δόθηκε.

Επισυνάπτοναι επίσης τα αρχεία, με τα οποία λήφθησαν οι μετρήσεις καθώς και το αρχείο timestamp των SSID που παρήχθησε για κάθε ένα από τις περιπτώσεις.

2 : Περιγραφή του τρόπου υλοποίησης

Πρώτα θα παρουσιαστεί γενικά και συνοπτικά και έπειτα πιο αναλυτικά για λεπτομέρεις.

2.1 Γενικά

Ο σκελετός της εργασίας ήταν η προηγούμενη εργασία (2η). Αυτό διότι επιλέχθηκε η παρούσα εργασία να λειτουργεί ως εξής.

Η πρώτη δουλειά που κάνει το κύριο νήμα είναι να δημιουργήσει ένα δεύτερο νήμα (σε ρόλο consumer), το οποίο είναι υπεύθυνο για την τακτοποίηση των SSID που σαρώθηκαν στο αρχείο. Το νήμα αυτό με το που δημιουργηθεί <<κλειδώνεται>> μέσω μιας εντολής pthread_cond_wait αφού δεν έχει ακόμη παραχθεί κανένα SSID σάρωσης.

Έπειτα το κύριο νήμα (που παίζει ρόλο producer) ενεργοποιεί όπως και στην προηγούμενη εργασία τα σήματα που θα μπορέσουν να το ξυπνήσουν, κάθε και πότε θα γίνεται αυτό το alarm και πώς θα διαικπεραιώνονται αυτή η διακοπή. Αφού κάνει τις παραπάνω αρχικοποιήσεις κοιμάται περιμένοντας την διακοπή.

Όταν η διακοπή φτάσει (μέσω timer), το κύριο νήμα μεταβαίνει στον signal handler , στο οποίο καλώντας το script searchWifi.sh λαμβάνει τα σαρωμένα SSID σε μορφή αρχείου. Έπειτα τα **επεξεργάζεται προκειμένου να προσθέσει στο τέλος κάθε SSID την χρονική στιγμή που έγινε η σάρωση**. Στη συνέχεια μέσω της pthread_cond_signal επιτρέπει το πρώτο thread (consumer) , να αρχίσει να δουλεύει. Τα SSID είναι φορτωμένα σε έναν πίνακα χαρακτήρων, τον buffer. Έχουν ληφθεί προϋποθέσεις ώστε τ**ο γέμισμα του buffer από το κύριο νήμα (producer) να μην επικαλύπτει SSID τιμές που ο consumer δεν έχει προλάβει να χειριστεί**. Η προσπέλαση του buffer πρέπει να γίνεται μέσα σε mutex αφού είναι κοινός και για τα 2 νήματα.

Το νήμα consumer μπορώντας πλέον να συνεχίσει την εκτέλεση, παίρνει γραμμή γραμμή τον buffer (**μέσα σε mutex**) και τοποθετεί την κάθε γραμμή (SSID & Timestamp) στο αρχείο μέσω της arrangeFile(**εκτός mutex**). Επιλέχθηκε η τακτοποίηση σε αρχείο να γίνει εκτός mutex ώστε να μην εμποδίζεται το νήμα-producer εκείνη την στιγμή να κάνει μια προγραμματισμένη σάρωση. Επίσης, για να μη χαθεί καμιά γραμμή του buffer λήφθησαν προϋποθέσεις, μέσω της μεταβλητής offset και της ενέργειας (buffer ← buffer+offset) .

Η τακτοποίηση των SSID γίνεται κάθετα στο αρχείο και κάθε γραμμή έχει τα timestamps οριζόντια που αντιστοιχούν στο SSID. Η τακτική που ακολουθείται είναι σάρωση του αρχείου και αντιγραφή αυτού σε ένα προσωρινό αρχείο, αναζήτηση επιθυμητού SSID, εγγραφή του timestamp ή του SSID & timestamp αν δεν βρεθεί το επιθυμητό SSID και τέλος επανεγγραφή όλων των προηγούμεων στο αρχικό αρχείο από το προσωρινό που δημιουργήθηκε.

2.2 Πιο παραστατικά...

Για καλύτερη κατανόηση συμβαίνει παραστατικά το εξής :
Το scriptάκι βγάζει μια έξοδο της μορφής:

“Fortnet B0-700”
“Wind cf2100”
“LoDi”
“den_moy_klebeis_pote”

Έπειτα ο producer παράγει τον buffer :

“Fortnet B0-700” TS:22:56:8.032106
“Wind cf2100” TS:22:56:8.032106
“LoDi” TS:22:56:8.032106
“den_moy_klebeis_pote” TS:22:56:8.032106

Ο buffer αυτός λαμβάνεται από τον consumer, ο οποίος παίρνει μια γραμμή την φορά (εντός mutex) και την τακτοποιεί μέσα στο αρχείο (εκτός mutex) που έχει την εξής μορφή:

“HOL ZTE” TS:22:50:44.309231 TS:22:45:42.112393
“Fortnet B0-700” TS:22:56:8.032106 TS:22:56:4.128462 TS:22:55:42.001233
“Wind cf2100” TS:22:56:8.032106 TS:22:55:42.001233
“LoDi” TS:22:56:8.032106 TS:22:55:42.001233
“den_moy_klebeis_pote” TS:22:56:8.032106

Τα αριστερότερα timestamps είναι τα πιο πρόσφατα.

Όταν ο consumer πάρει μια γραμμή ο buffer θα γίνει έτσι : (λείπει η πρώτη γραμμή)

“Wind cf2100” TS:22:56:8.032106
“LoDi” TS:22:56:8.032106
“den_moy_klebeis_pote” TS:22:56:8.032106

Και τούτο το κάθε εξής. Έτσι αν ο consumer έχει προλάβει να κάνει μόνο μια γραμμή όταν ο producer κληθεί (μέσω του σήματος alarm) να ξανασαρώσει θα παράγει έναν buffer εξής μορφής (κρατώντας ίδιες τις 3 πρώτες γραμμές):

“Wind cf2100” TS:22:56:8.032106
“LoDi” TS:22:56:8.032106
“den_moy_klebeis_pote” TS:22:56:8.032106
“New Wifi1” TS:22:59:08.103251
“New Wifi2” TS:22:59:08.103251
“New Wifi3” TS:22:59:08.103251
“New Wifi4” TS:22:59:08.103251
“New Wifi5” TS:22:59:08.103251

3 Περί Μέτρησης Διαφοράς Σάρωση Εγγραφής

Όπως παρατηρήθηκε από παραπάνω , το timestamp που γράφεται στο αρχείο είναι το ακριβές timestamp που συνέβη την ώρα της σάρωσης του SSID. Αυτό κατορθώθηκε αφού προστέθηκε μέσα στον ίδιο τον buffer, που περιέχει τα SSID.

Συνεπώς η μέτρηση διαφοράς από την στιγμή της σάρωσης μέχρι την στιγμή της εγγραφής δεν θα είχε άλλο νόημα, παραμόνο για να δούμε την εξέλιξη του συστήματος και πώς αυτό συμπεριφέρεται.

Για την πληρότητα της άσκησης, λοιπόν, μετρήθηκε το παραπάνω διάστημα. Κατά την εγγραφή στο αρχείο έγινε μια κλήση στην gettimeofday και συγκρίθηκε το αποτέλεσμα με το timestamp που φέρει ο buffer.

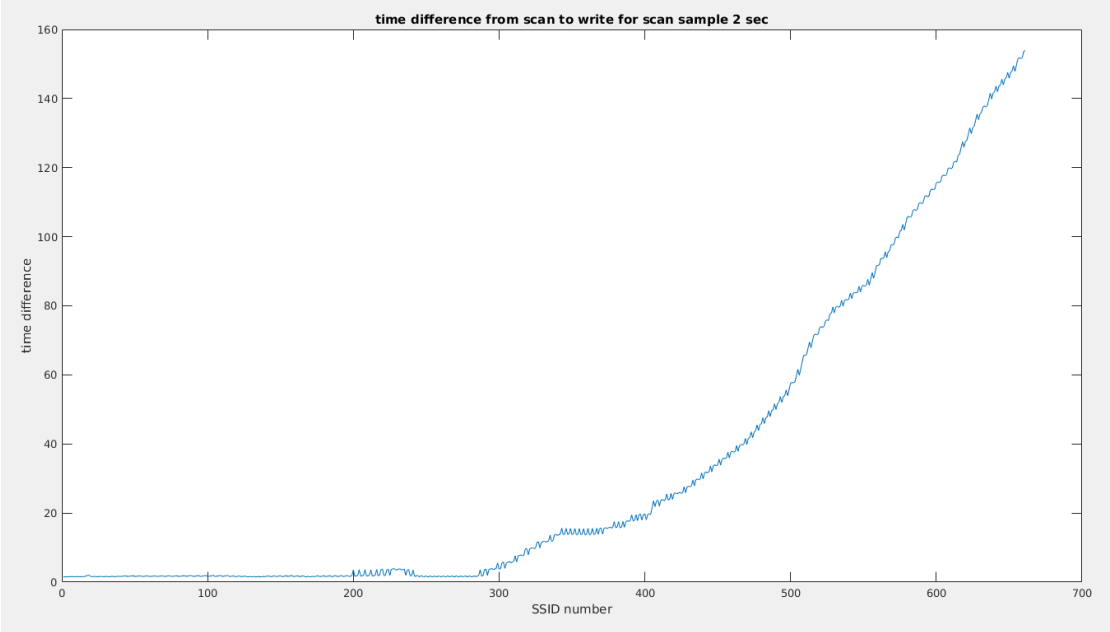
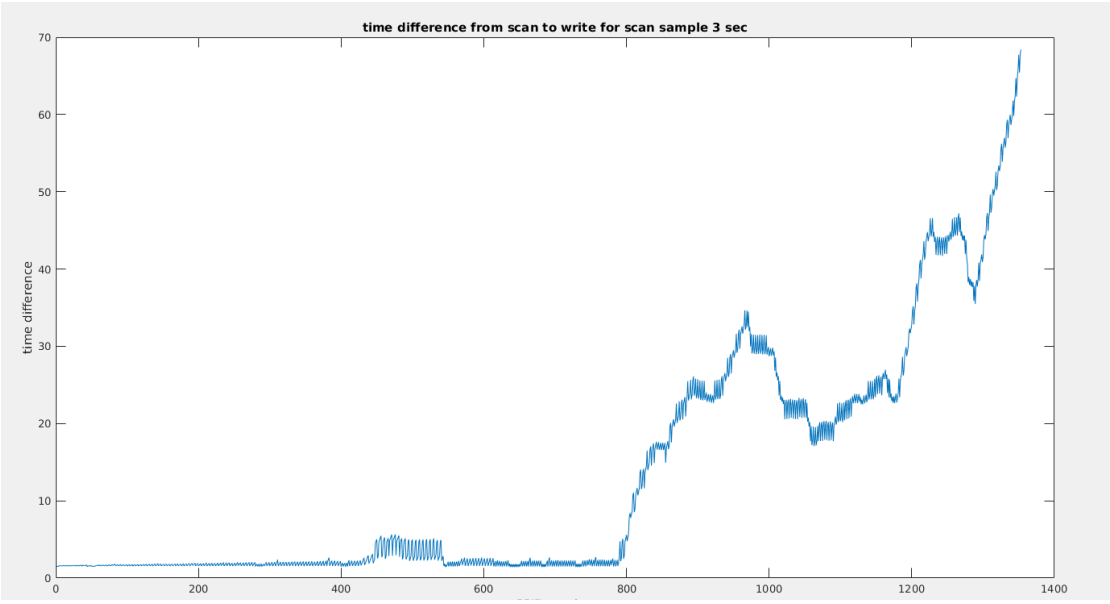


Illustration 1



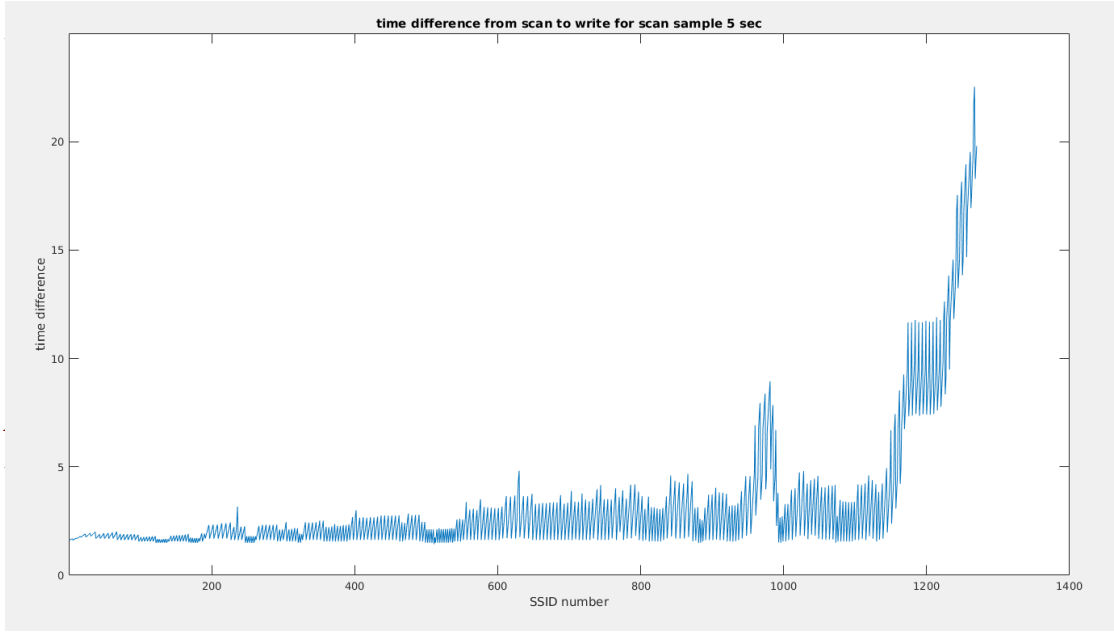


Illustration 3

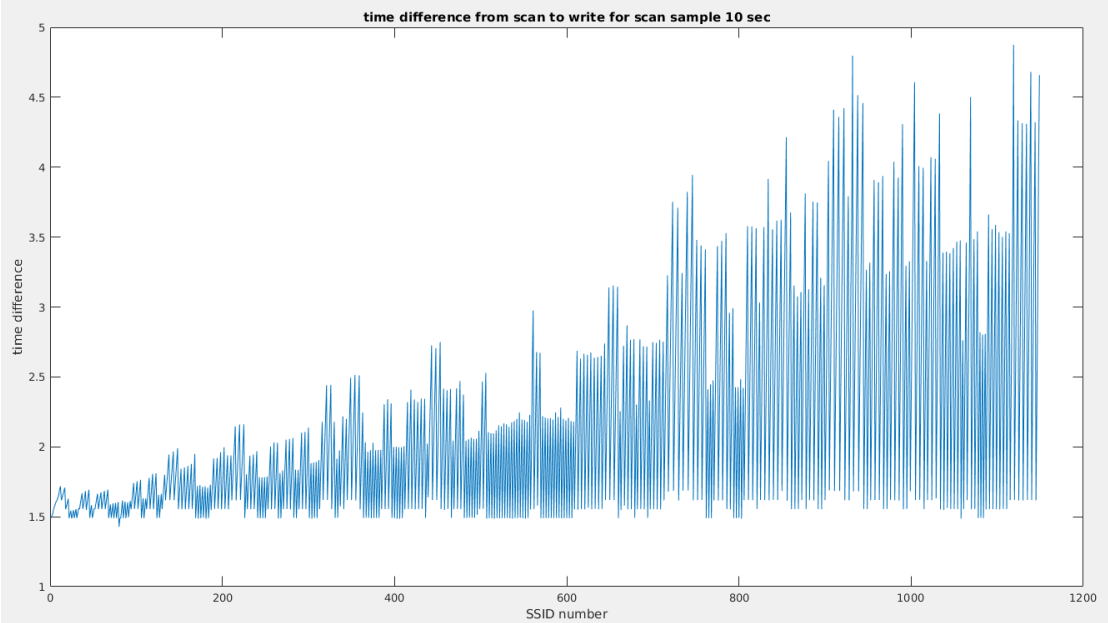
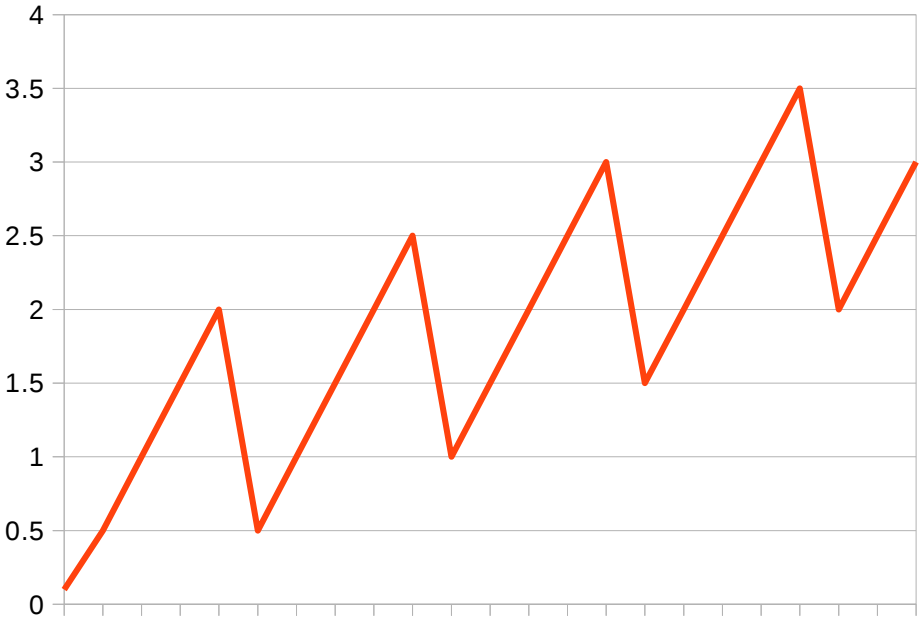


Illustration 4

Πρώτα θα παρατηρήσουμε της μορφή που τον envelope που έχει το κάθε διάγραμμα και θα δούμε ότι πάει κάπως έτσι προσεγγιστικά :



Αυτό συμβαίνει γιατί όταν πάει να τακτοποιήσει την πρώτη γραμμή του buffer θα έχει περάσει λιγότερος χρόνος απ`ότι όταν πάει να τακτοποιήσει την δεύτερη γραμμή του buffer και πόσο μάλιστα όταν πάει να τακτοποιήσει την τελευταία γραμμή του buffer που θα έχουν μεσολαβήσει τόσες αντιγραφές και αναζητήσεις κειμένου που κοστίζουν χρονικά.

Όμως η ιστορία δεν τελειώνει εκεί , αφού οι τιμές διαφορών δεν είναι περιοδικές αλλά σταδιακά αυξάνονται. Αυτό συμβαίνει γιατί αυξάνεται το μέγεθος του αρχείου σιγά σιγά και έτσι η αναζήτηση και η αντιγραφή έχουν μεγαλύτερο χρονικό κόστος. Η αποκορύφωση του σφάλματος γίνεται όταν ο ρυθμός σάρωσης γίνεται συγκρίσιμος με τον χρόνο διευθέτησης του buffer στο αρχείο.

Η παραπάνω πρόταση εξηγεί και γιατί όσο μεγαλώνει ο ρυθμός σάρωσης φτάνουμε πιο γρήγορα σε μεγαλύτερες τιμές διαφοράς. Γιατί πλέον δεν είναι ανεπαίσθητη η προσπέλαση του αρχείου και η εγγραφή αυτού σε σχέση με τον ρυθμό σάρωσης SSID.

4 Περί χρόνου αδράνειας της CPU

Για την μέτρηση του χρόνου αδράνειας της CPU έγιναν τα εξής :

Τοποθετήθηκαν οι εντολές *start_cl=clock();* & *gettimeofday(&cpuTime_1, NULL);* στην αρχή του προγράμματος και έπειτα μετά την κάθε διαικπεραίωση διακοπής (παραγωγή buffer με SSID & timestamps) γράφηκαν οι εντολές *gettimeofday(&cpuTime_2, NULL); & end_cl=clock();* παρόμοια.

Έχουμε ότι

cpu_time_used = ((double) (end_cl - start_cl))/CLOCKS_PER_SEC ;
realTimePassed = ((cpuTime_2.tv_usec – cpuTime_1.tv_usec)/1.0e6+ cpuTime_2.tv_sec - cpuTime_1.tv_sec) ;

και άρα

cpu_idle_rate = (realTimePassed-cpu_time_used)/realTimePassed * 100 ;

που μας δείχνει το ποσοστό αδράνειας της CPU μετά από κάθε διαικπεραίωση διακοπής.

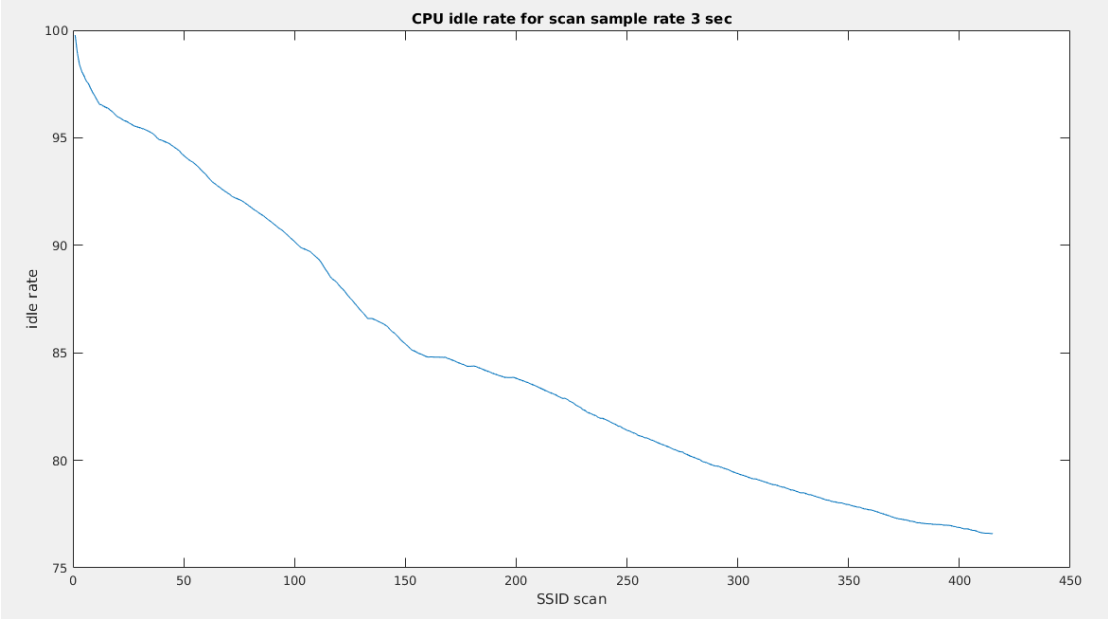


Illustration 5

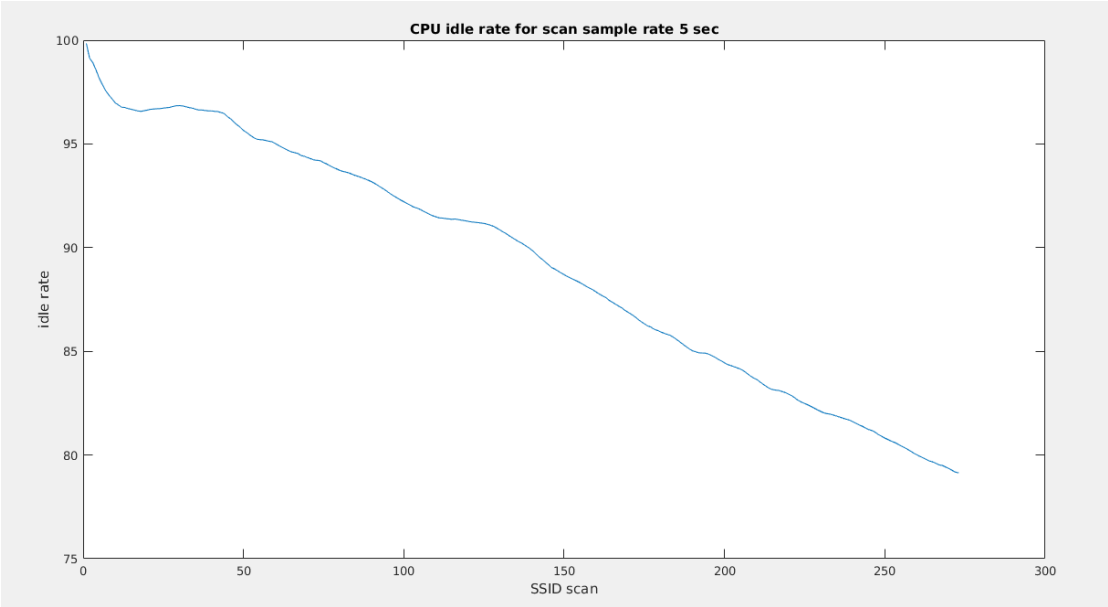


Illustration 6

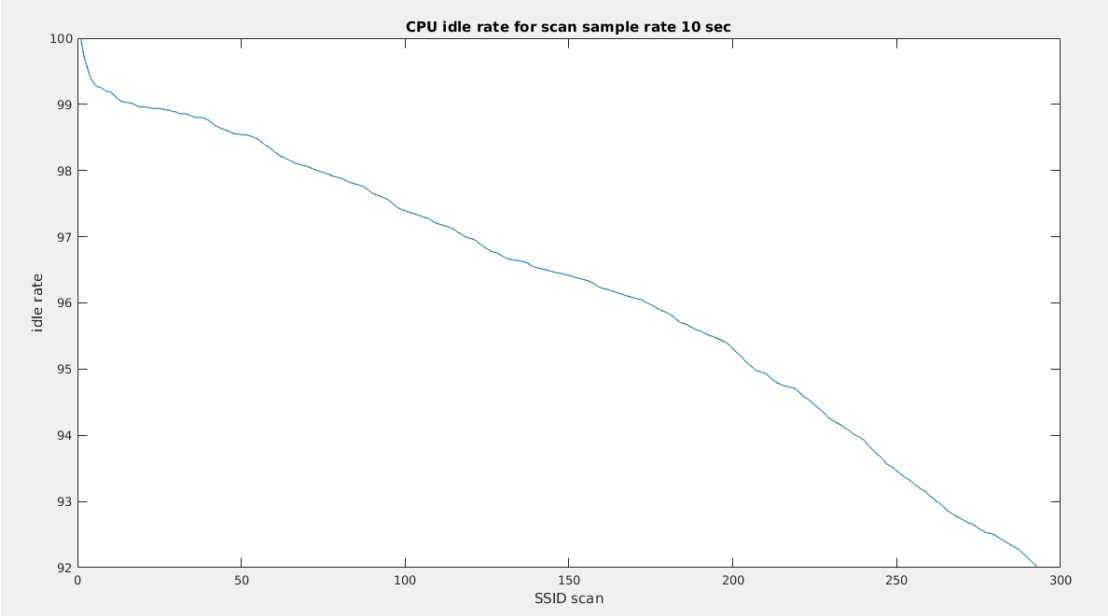


Illustration 7

Επιθυμητό είναι εφόσον το πρόγραμμά μας έχει εφαρμογή σε ενσωματωμένα συστήματα να έχουμε υψηλό ποσοστό αδράνειας της CPU, ωστε να μην χαλάει πολύ ενέργεια και συνεπώς μπαταρία. Αυτό το κατορθώνουμε κυρίως μέσω των σημάτων , με το να κοιμάται η CPU τον περισσότερο χρόνο και να περιμένει να ξυπνήσει από ένα σήμα για να κάνει τη δουλειά μια στο τόσο.

Το consumer thread που έχει εξαπολύσει από την αρχή θα <<κοιμηθεί>> εφόσον έχει διευθετήσει όλα τα διαθέσιμα SSID με τα timestamps τους. Αυτός ο ύπνος θα έρθει μέσω της εντολής *pthread_cond_wait*.

Είναι λογικό όσο περνάει ο χρόνος να αυξάνεται η δουλειά της CPU καθώς αυξάνεται σε μέγεθος το αρχείο και η δουλειά που πρέπει να γίνει για την εγγραφή σε αυτό (σάρωση , αναζήτηση, αντιγραφή σε προσωρινό αρχείο και επανεγγραφή στο αρχικό).

Ώστόσο όταν το χρονικό διάστημα σάρωσης είναι μεγαλύτερο , υπάρχει μεγαλύτερη άνεση από την πλευρά του συστήματος. Ουσιαστικά ο CPU έχει να κάνει την ίδια δουλειά που θα έκανε σε ένα μικρό χρονικό διάστημα σε ένα μεγάλο χρονικό διάστημα. Οπότε αμέσως καταλαβαίνουμε ότι όσο μεγαλύτερο είναι το χρονικό διάστημα σάρωσης τόσο μεγαλύτερο θα είναι και το ποσοστό αδράνειας του CPU.

Έχοντας ενδιαφέρον να δω μέχρι πού θα φτάσει το γράφημα , επέλεξα να το τρέξω στα linux με αντίστοιχο script εύρεσης SSID και με χρονικό διάστημα σάρωση 0.5 sec (καθώς έπαιρνε πάρα πολύ χρόνο για διαστήματα σάρωσης άνω των 3 sec). Το γράφημα έφτασε μέχρι ποσοστό αδράνειας περίπου 15% και έπειτα το πρόγραμμα

κόλλησε. Επίσης από τα 19% και μετά ο buffer του συστήματος συνεχώς αυξάνονταν αντί να μειώνεται . Αυτό δείχνει ότι το πρόγραμμα είχε φτάσει σε στάδιο να παράγει περισσότερα SSID απ`ότι μπορεί να τακτοποιήσει και συνεπώς crashαρε.