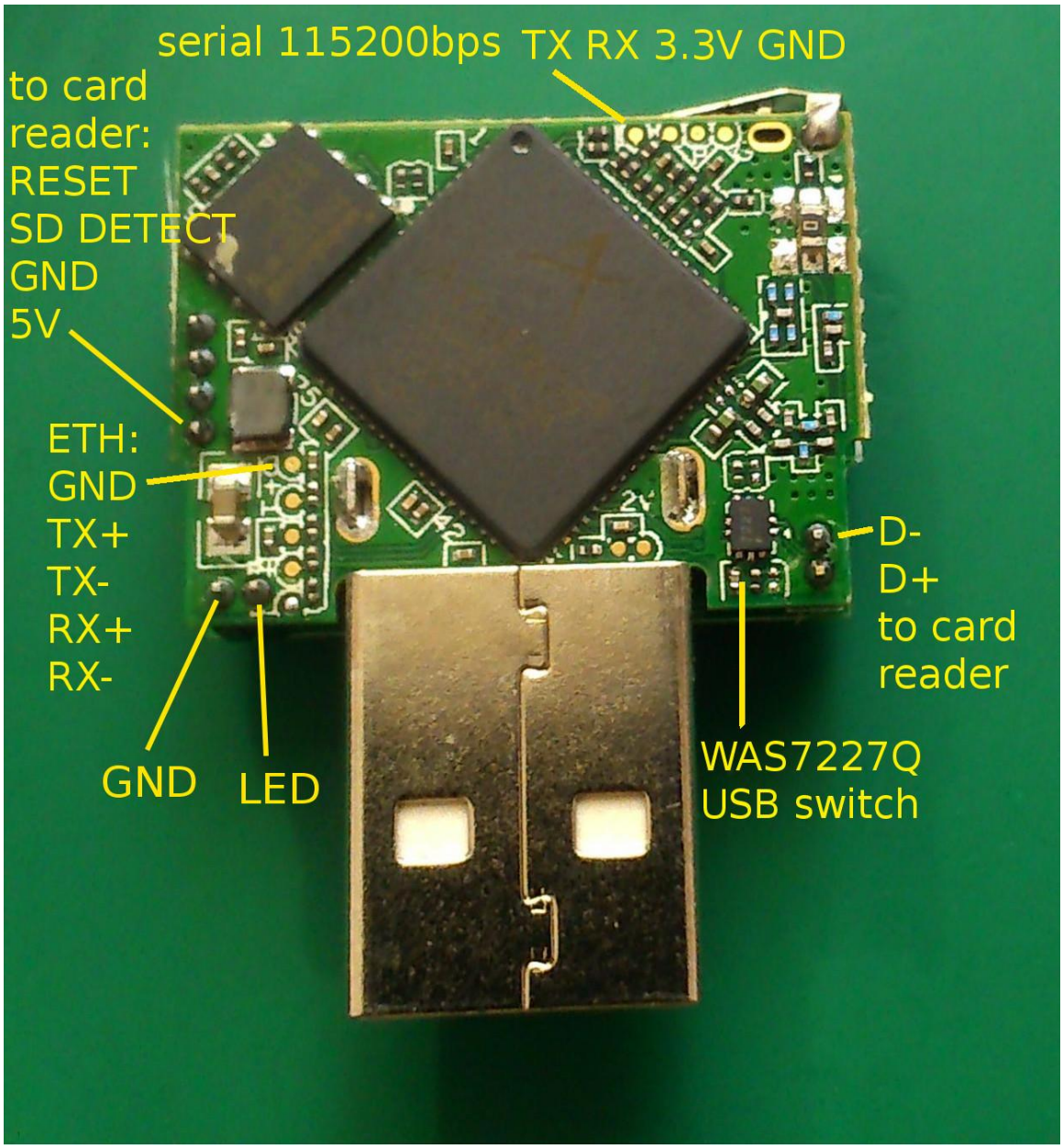

serial 115200bps TX RX 3.3V GND
to card
reader:
RESET
SD DETECT
GND
5V
ETH:
GND
TX+
TX-
RX+
RX-
GND
LED
D-
D+
to card
reader
WAS7227Q
USB switch

ΑΣΚΗΣΗ 3

1.Εισαγωγή

Στόχος της εργασίας είναι να δημιουργηθεί ένα πρόγραμμα που θα τρέχει σε μία ενσωματωμένη συσκευή, στην δικιά μας περίπτωση στο Zsun, οποία θα μαζεύει πληροφορίες για τα κοντινά διαθέσιμα Wifi κάθε μία δοσμένη χρονική στιγμή.

Ουσιαστικά, για κάθε καινούργιο Wifi, πρέπει να αποθηκεύεται το ssid του (με άλλα λόγια το όνομα του Wifi) και το timestamp του (το οποίο δηλώνει τη χρονική στιγμή που βρέθηκε το κάθε Wifi). Για τα Wifi που βρίσκονται ήδη στη λίστα πρέπει να προστίθεται απλά το καινούργιο timestamp. Επιπλέον η υλοποίηση μας θα προσπαθήσουμε να είναι όσο το δυνατόν πιο ενεργειακά αποδοτική γίνεται.

2.Περιγραφή υλοποίησης

Καταρχήν να επισημάνουμε ότι το πρόγραμμα μας δέχεται την περίοδο μεταξύ των σαρώσεων μέσω της παραμέτρου argv[1]. Στη συνέχεια βασιζόμαστε στην προηγούμενη εργασία για να βάλουμε ένα alarm που θα ξυπνάει την sigsuspend κάθε n δευτερόλεπτα όπου n ο χρόνος της περιόδου μεταξύ των σκαναρισμάτων. Η μόνη τροποποίηση που κάναμε σε σχέση με την προηγούμενη εργασία είναι ότι ο handler που διαχειρίζεται την διακοπή της sigsuspend είναι κενός. Κάναμε αυτή την αλλαγή για να επωφεληθούμε από μία ιδιότητα της sigsuspend η οποία αν μπεί στον handler επιστρέφει τιμή -1 και με αυτόν τον τρόπο μπορούμε να δούμε κάθε πότε περνάει η περίοδος και να κάνουμε τις κατάλληλες διεργασίες. Στη δικιά μας περίπτωση κάθε φορά που περνούσε ο χρόνος μιας περιόδου εξαπολύαμε ένα Thread, το Thread του consumer ώστε να πάει να σκανάρει τα διαθέσιμα Wifi και να τα περάσει σε μία ουρά για να πάει στη συνέχεια ο consumer να τα διαβάσει και να τα γράψει σε ένα αρχείο. Το κομμάτι του κώδικα που υλοποιεί αυτό που προαναφέραμε εμφανίζεται παρακάτω.

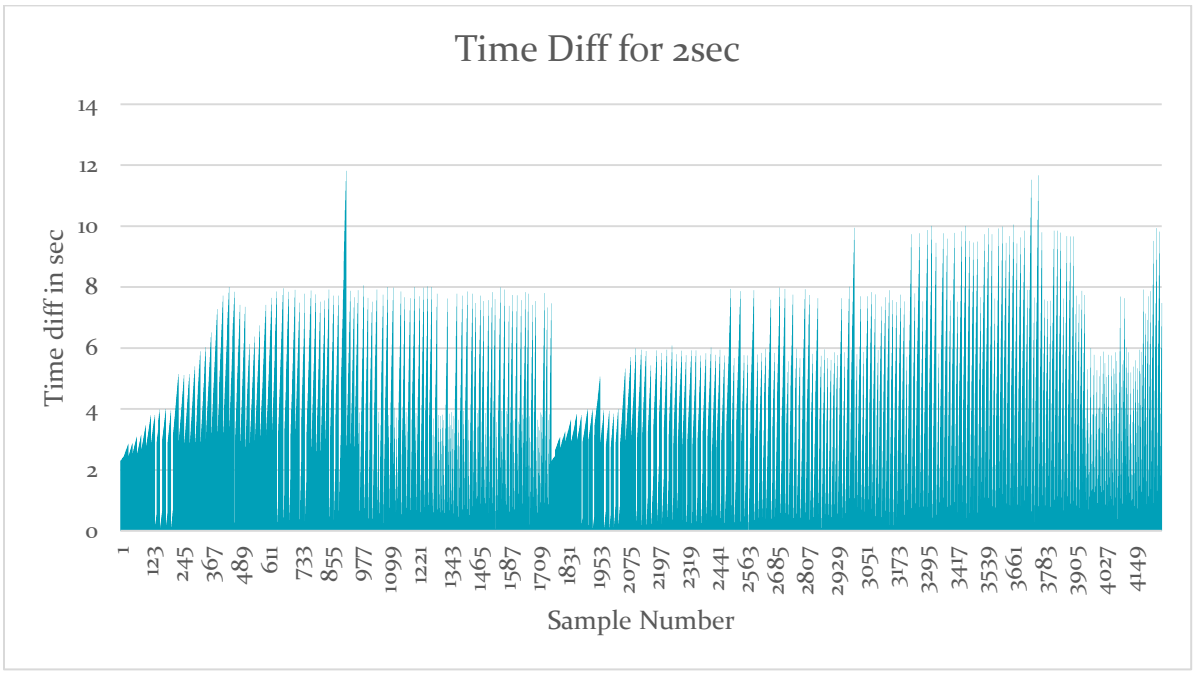
```
    for(;;){
        if(sigsuspend(&mask)==-1){
          pthread_create (&pro, NULL, producer, fifo);
          pthread_join (pro, NULL);
          gettimeofday(&stop,NULL);
          end=clock();
          total=(double)(end - start) / CLOCKS_PER_SEC;
          double elapsed=(stop.tv_sec- go.tv_sec + (stop.tv_usec -
go.tv_usec)/1000000.0); // i estimate the elapsed time
          double percentage=((elapsed-total)/elapsed) *100;
          CLOCK=fopen("Clock.txt","a");
          fprintf(CLOCK, "Idle_percentage:%.4lf\n",percentage);
          fclose(CLOCK);
          firsttime=0;
        }
```

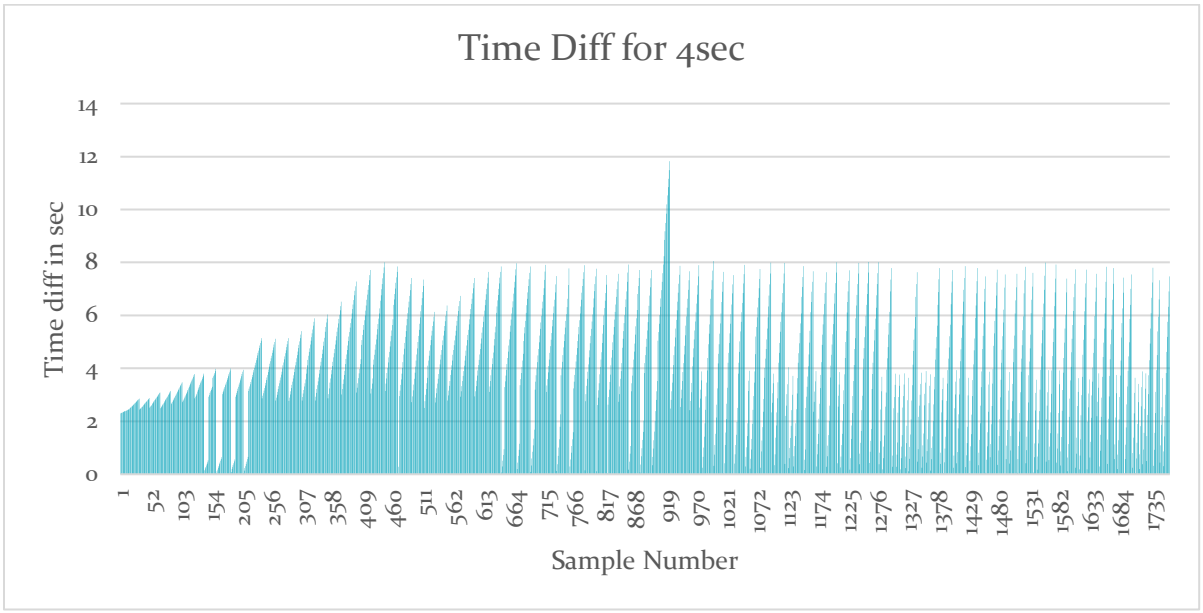
Εδώ να τονιστεί ότι βασιστήκαμε στο μοντέλο producer-consumer που μας δόθηκε με κάποιες μικρές παραλλαγές, που είχαν να κάνει με την προσθήκη κάποιων μεταβλητών, π.χ αλλάξαμε τον buffer από μονοδιάστατο σε δισδιάστατο για να αποθηκεύουμε σε κάθε θέση ένα string που περιέχει το όνομα του Wifi και το timestamp της στιγμής που βρέθηκε, όπως και μία flag για το αν το Wifi βρέθηκε στην πρώτη σάρωση για να το καταχωρήσουμε απευθείας στο αρχείο που περιέχει τα Wifi με τα timestamps τους. Ακόμα αλλάξαμε τις συναρτήσεις queueAdd και queueDel. Ουσιαστικά απλά βγάλαμε της εντολές τροποποίησης του buffer εκτός και τις βάλαμε αντίστοιχα στον producer και στον consumer.

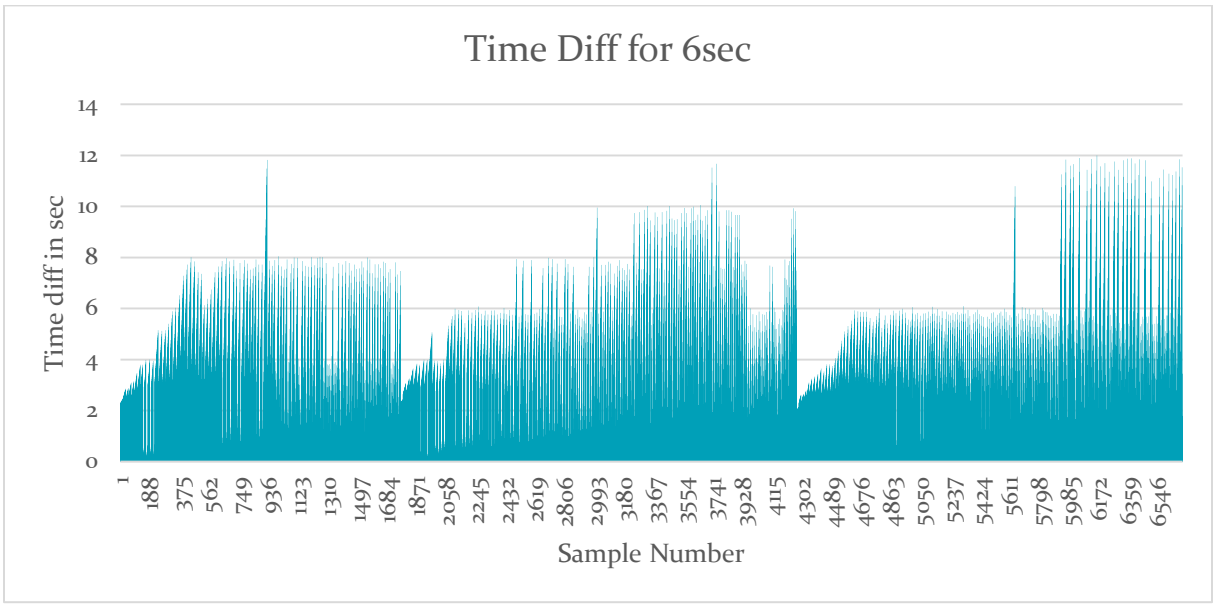
Στη συνέχεια θα σχολιάσουμε λίγο την λειτουργία του producer και του consumer. Όσον αφορά τον producer αυτό που κάνει είναι να τρέχει ένα batch file για την σάρωση των διαθέσιμων Wifi και να καταχωρεί το όνομα του κάθε Wifi που βρίσκει μαζί με τον χρόνο της στιγμής που το βρήκε σε μία δομή ουράς. Από την πλευρά του ο consumer πάει και διαβάζει από τον buffer τα στοιχεία που καταχώρησε ο producer και τα γράφει σε ένα αρχείο ελέγχοντας πρώτα αν έχει ξαναγραφτεί το αντίστοιχο Wifi ώστε σε αυτή την περίπτωση απλά να πάει να προσθέσει το καινούργιο timestamp που βρέθηκε δίπλα στο όνομα του. Όλα τα παραπάνω φυσικά γίνονται με συνέπεια και προσοχή ως προς την διαχείριση των δεδομένων, που υλοποιείται μέσω σημάτων (pthread_condition_wait) και mutexes (ή αλλιώς κλειδαριών).

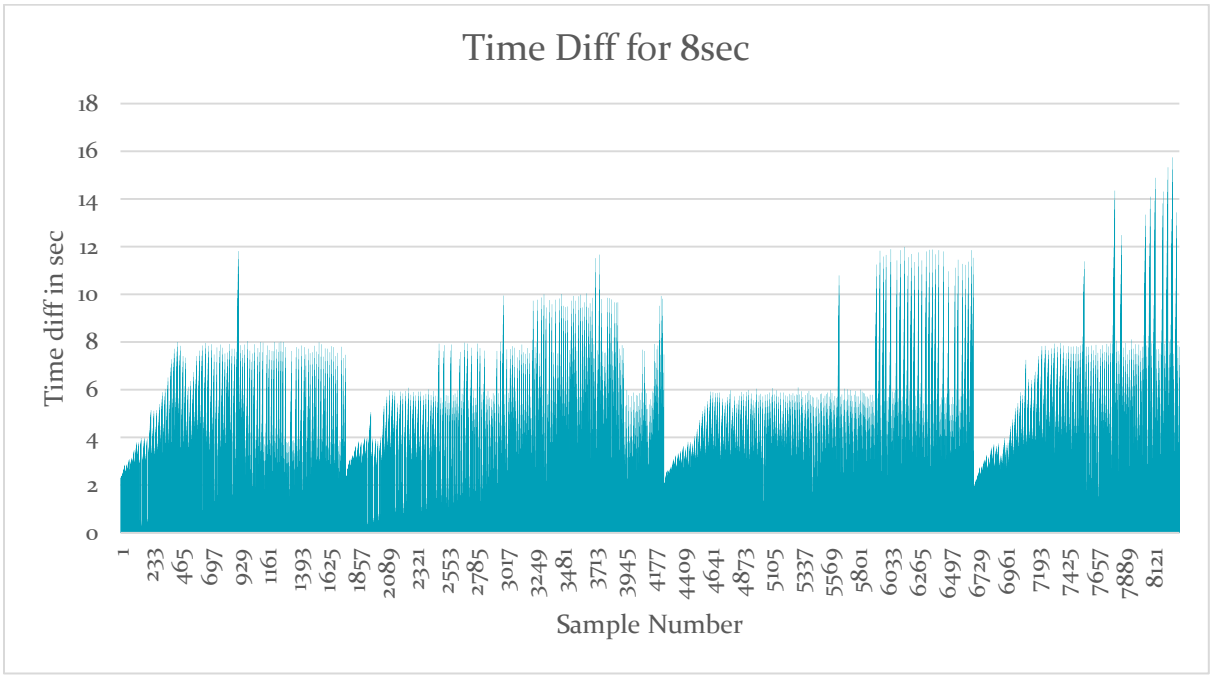
3.Μέτρηση Διαφορών

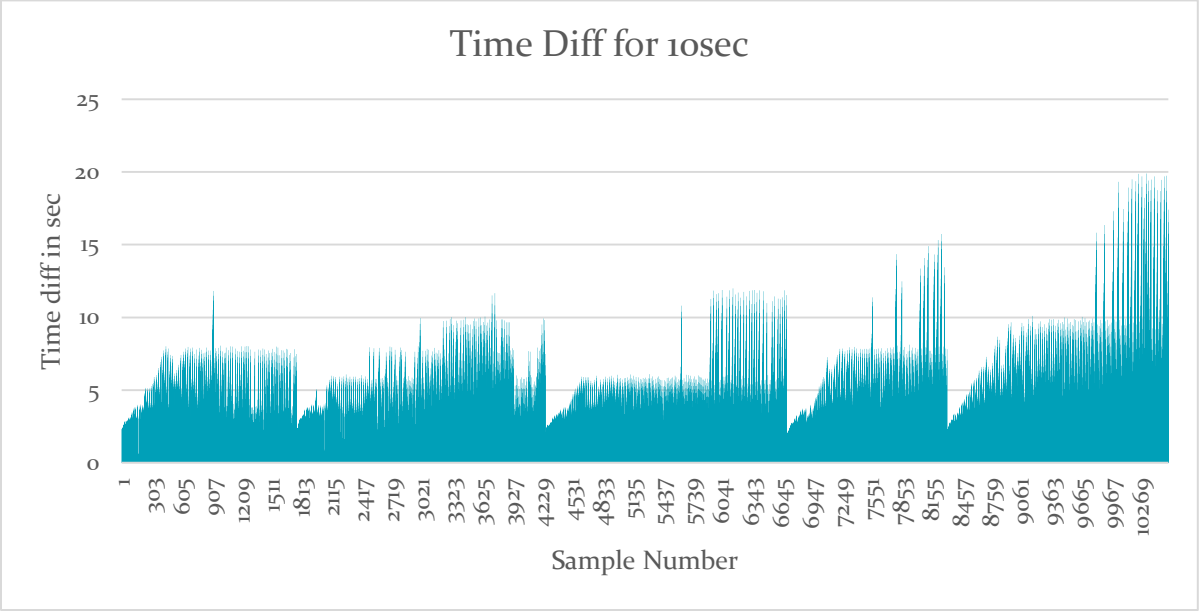
Μας ζητήθηκε να βρούμε την διαφορά μεταξύ του χρόνου σάρωσης ενός Wifi μέχρι τον χρόνο που τελικά καταγράφτηκε στο αρχείο μας και να την παραστήσουμε μέσω διαγραμμάτων. Οι μετρήσεις που πάρθηκαν ήταν για περίοδο μεταξύ δύο σαρώσεων 2,4,6,8 και 10 δευτερολέπτων και τα διαγράμματα και κάποια σχόλια παρουσιάζονται παρακάτω:


Time Diff for 2sec
Time diff in sec
0
2
4
6
8
10
12
14
1
123
245
367
489
611
733
855
977
1099
1221
1343
1465
1587
1709
1831
1953
2075
2197
2319
2441
2563
2685
2807
2929
3051
3173
3295
3417
3539
3661
3783
3905
4027
4149
Sample Number


Time Diff for 4sec
Time diff in sec
0
2
4
6
8
10
12
14
1
52
103
154
205
256
307
358
409
460
511
562
613
664
715
766
817
868
919
970
1021
1072
1123
1174
1225
1276
1327
1378
1429
1480
1531
1582
1633
1684
1735
Sample Number


Time Diff for 6sec
Time diff in sec
0
2
4
6
8
10
12
14
1
188
375
562
749
936
1123
1310
1497
1684
1871
2058
2245
2432
2619
2806
2993
3180
3367
3554
3741
3928
4115
4302
4489
4676
4863
5050
5237
5424
5611
5798
5985
6172
6359
6546
Sample Number


Time Diff for 8sec
Time diff in sec
0
2
4
6
8
10
12
14
16
18
1
233
465
697
929
1161
1393
1625
1857
2089
2321
2553
2785
3017
3249
3481
3713
3945
4177
4409
4641
4873
5105
5337
5569
5801
6033
6265
6497
6729
6961
7193
7425
7657
7889
8121
Sample Number



Παρατηρούμε ότι καθώς αυξάνονται τα δείγματα που παίρνουμε δηλαδή καθώς περνάει ο χρόνος αυξάνεται με μία γραμμικότητα ο χρόνος που κάνει να γράψει τα Wifi σε σχέση με το χρόνο που σαρώθηκαν. Αυτό συμβαίνει γιατί το αρχείο συνεχώς μεγαλώνει οπότε ο consumer κάνει περισσότερη ώρα να το διατρέξει σε σχέση με μία προηγούμενη φορά. Επίσης, βλέπουμε μία περιοδικότητα, αυτή η περιοδικότητα έχει να κάνει μάλλον με το γεγονός ότι τα πρώτα Wifi είναι πιο εύκολα προσβάσιμα οπότε κάνει ο consumer πιο λίγη ώρα να τα βρεί σε σχέση με τα τελευταία οπότε γράφει πιο γρήγορα τα δεδομένα.

4.Ποσοστό χρόνου αδράνειας

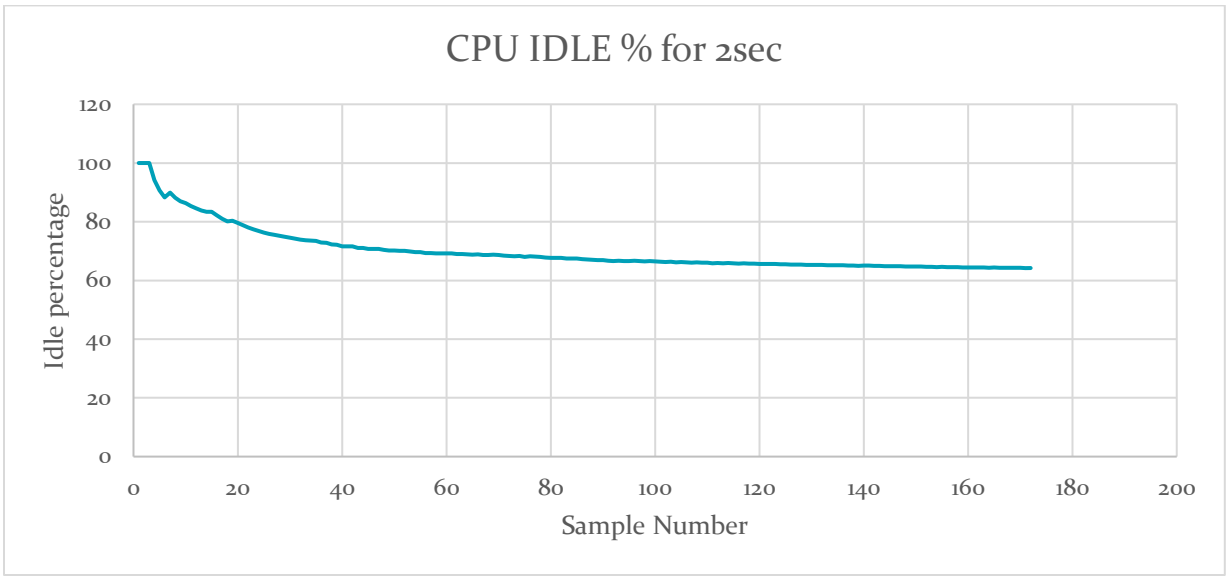
Για την μέτρηση του χρόνου αδράνειας της CPU χρησιμοποιήθηκαν οι συναρτήσεις clock() και gettimeofday(). Ουσιαστικά αυτό που κάναμε ήταν να μετράμε πόσοι χτύποι ρολογιού έχουν περάσει από την αρχή μέχρι το πέρας μίας σάρωσης, να μετατρέπουμε τους χτύπους σε δευτερόλεπτα χρήσης της CPU και ενώ έχουμε μετρήσει ταυτόχρονα τον πραγματικό χρόνο που πέρασε ως τότε να παίρνουμε τον παρακάτω τύπο και να προκύπτει το επιθυμητό ποσοστό.

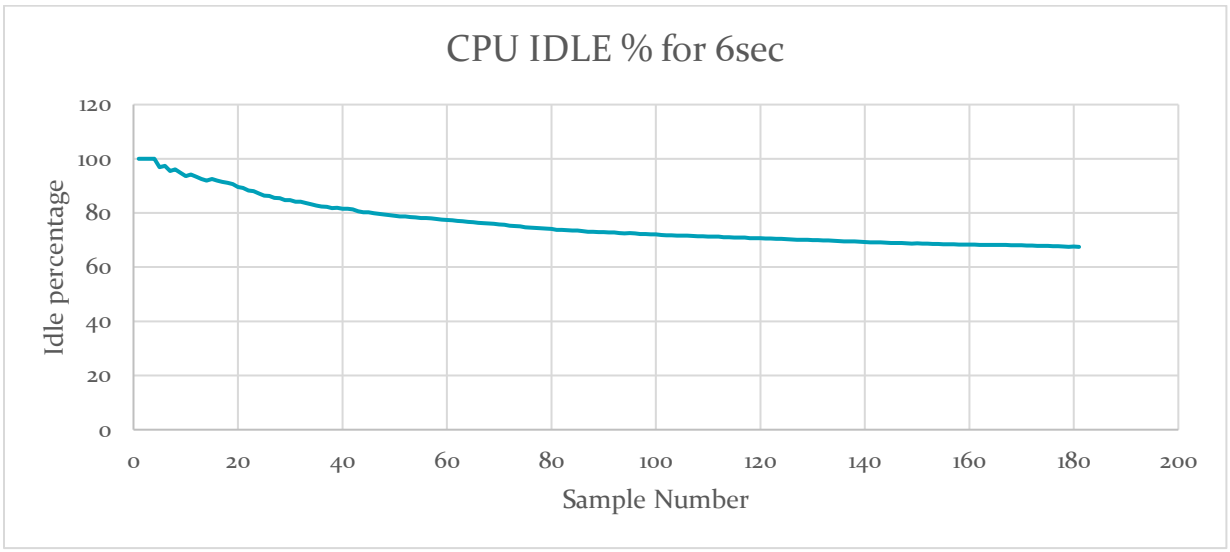
```
double percentage=((elapsed-total)/elapsed) *100;
```

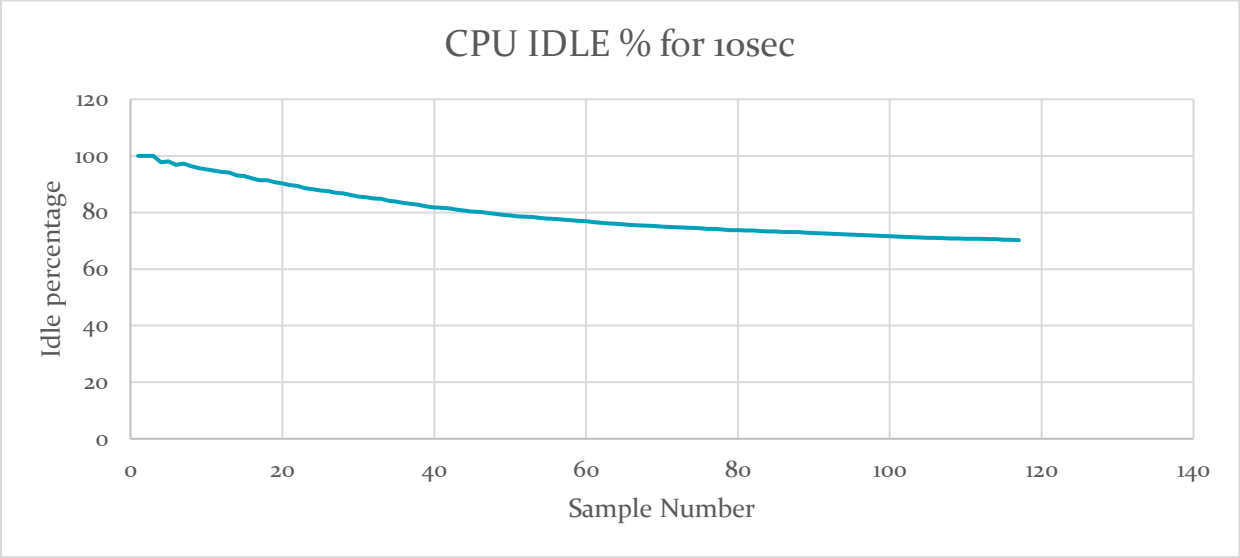
,

Όπου το elapsed είναι ο πραγματικός χρόνος που πέρασε και το total ο χρόνος λειτουργίας της CPU σε δευτερόλεπτα.

Παρουσιάζουμε διαγράμματα των ποσοστών που εμφανίστηκαν σε μετρήσεις περιόδων 2,6 και 10sec.


CPU IDLE % for 2sec
Idle percentage
120
100
80
60
40
20
0
0
20
40
60
80
100
120
140
160
180
200
Sample Number


CPU IDLE % for 6sec
Idle percentage
120
100
80
60
40
20
0
0
20
40
60
80
100
120
140
160
180
200
Sample Number



Σκοπός των προγραμματιστών που κάνουν εφαρμογές/προγράμματα για ενσωματωμένα συστήματα είναι η εφαρμογή/πρόγραμμα να κρατάει την CPU όσο το δυνατόν περισσότερο αδρανή. Αυτό συμβαίνει γιατί να περισσότερα ενσωματωμένα συστήματα πραγματικού χρόνου στις μέρες μας λειτουργούν μέσω μπαταρίας, οπότε όσο πιο λίγο βάζεις την CPU να δουλέψει τόσο πιο λίγη ενέργεια καταναλώνει το σύστημα σου. Άρα έχει και περισσότερη αυτονομία και αυτό είναι μεγάλο ατού σε τέτοιες συσκευές.

Στη δικιά μας περίπτωση τώρα προσπαθήσαμε να πετύχουμε καλή ενεργειακή απόδοση δηλαδή μεγάλο idle time μέσω των signals που στέλνει ο producer στον consumer για το αν η ουρά είναι γεμάτη ή αν είναι άδεια, οπότε αν υπάρχει δουλεία. Στην περίπτωση που δεν υπάρχει δουλεία κοιμούνται ώστε να μην ξοδεύουν ενέργεια αδίκως.(empty ουρά->consumer sleeps, full ουρά->producer sleeps).

5.Παρατήρηση

Παρατηρήσαμε ότι ανάλογα με την περίοδο σάρωσης και τον αριθμό των διαθέσιμων Wifi που βρίσκουμε υπάρχει περίπτωση να χαθούν περίοδοι μετρήσεων καθώς περνάει ο χρόνος. Αυτό συμβαίνει γιατί ο producer κάνει περισσότερη ώρα να βάλει τα Wifi στην ουρά από την ώρα πού έχουμε θέσει για περίοδο σκαναρίσματος με αποτέλεσμα ενώ φτάνει το επόμενο σήμα σκαναρίσματος, ο producer δεν έχει βάλει ακόμα τα δεδομένα από το προηγούμενο σκανάρισμα οπότε δεν λαμβάνει το σήμα (για μικρές περιόδους αυτό συμβαίνει από τις πρώτα κιόλας δείγματα ενώ όσο ανεβαίνουμε σε θέμα περιόδου αργεί να εμφανιστεί κάτι τέτοιο).