Αριστοτελειο Πανεπιστημιο Θεσσαλονικης (Α.Π.Θ.)

ΗΥ3604 Ενσωματωμενα Συστηματα Πραγματικου Χρονου

Εργασία Raspberry Pi Zero

Αντωνιάδης Δημήτριος (8462): akdimitri@auth.gr

23 Σεπτεμβρίου 2019

Περιεχόμενα

1 Εισαγωγή.

Το παρόν έγγραφο αποτελεί την αναφορά της τελικής εργασίας που πραγματοποιήθηκε στο πλαίσιο του μαθήματος *Ενσωματωμένα Συστήματα Πραγματικού Χρόνου*. Η εργασία αυτή υλοποιεί τη λειτουργία ενός ενσωματωμένου συστήματος παραγωγής και αναδρομολόγησης απλών μηνυμάτων.

Παραπάνω, έγινε μία σύντομη περιγραφή της εργασίας. Στην ακόλουθη $(2_η)$ ενότητα παρουσιάζονται τα αρχεία που περιλαμβάνονται εντός του παραδοτέου φακέλου της εργασία. Στην τρίτη $(3_η)$ ενότητα παρουσιάζεται ο τρόπος μεταγλώττισης και εκτέλεσης του προγράμματος, ενώ στην τέταρτη $(4_η)$ περιγράφεται ο τρόπος υλοποίησης του προγράμματος. Στην πέμπτη $(5_η)$ ενότητα παρουσιάζονται τα αποτελέσματα και τέλος στην έκτη $(6_η)$ ενότητα συνοψίζονται τα συμπεράσματα της εργασίας.

2 Περιεχόμενα εργασίας.

Το σύνολο των αρχείων της εργασίας μπορεί να βρεθεί στον παρακάτω σύνδεσμο:

https://github.com/akdimitri/Raspberry-Pi-0-Ad-hoc-Communication-System

```
Repository
├── code
├── executable files
├── matlab
├── misc
└── results
```

Οι παραπάνω φάκελοι περιλαμβάνονται εντός του Repository με τα αρχεία της εργασία.

Ο φάκελος code περιλαμβάνει τον πηγαίο κώδικα της εργασίας.

```
Repository
└── code
    ├── client
    │   ├── client.c
    │   └── client.h
    ├── server
    │   ├── server.c
    │   └── server.h
    ├── message_generator
    │   ├── message_generator.c
    │   └── message_generator.h
    ├── circular_buffer
    │   ├── circular_buffer.c
    │   └── circular_buffer.h
    └── main.c
```

Εντός του φακέλου executable files περιλαμβάνονται δύο υποφάκελοι. Κάθε υποφάκελος περιέχει ένα εκτελέσιμο αρχείο για τη συσκευή Raspberry Pi 0 και ένα αρχείο AEM.txt το οποίο είναι η λίστα με τα AEM-Παραλήπτες των παραγώμενων μηνυμάτων. Τα αρχεία αυτά χρησιμοποιήθηκαν στις δοκιμές με σκοπό την παρουσίαση της ορθής λειτουργίας του προγράμματος.

Στο φάκελο matlab περιέχεται το αρχείο script.m το οποίο αναλύει το log file που παράγεται ύστερα από μία δοκιμή σε μία συσκευή και παρουσιάζει τα αποτελέσματα.

Ακόμη, ο φάκελος misc περιλαμβάνει ένα αρχείο, το αρχείο rc.local. Το αρχείο αυτό αντικαθιστά

το αρχείο /etc/rc.local της συσκευής. Η λειτουργία του είναι κάθε φορά που εκκινεί η συσκευή να τη ρυθμίζει κατάλληλα ώστε να απαντάει σε ICMP Echo Messages. Επιπλέον, μέσω του αρχείου αυτού είναι δυνατή η εκκίνηση του προγράμματος κατα την εκκίνηση της συσκευής.

Τέλος, ο φάκελος results περιέχει δύο υποφακέλους, τους test_final_8462 και test_final_8535. Κάθε υποφάκελος περιλαμβάνει δύο αρχεία, τα αρχεία circular_buffer.txt και log.txt. Το πρώτο αρχείο περιλαμβάνει την κατάσταση του circular buffer τη στιγμή που διακόπηκαν οι μετρήσεις ,ενώ το δεύτερο αρχείο περιλαμβάνει το σύνολο των γεγονότων που πραγματοποιήθηκαν κατα τη διάρκεια των μετρήσεων.

3 Compilation και Execution.

Για τη μεταγλώττιση του προγράμματος απαιτείται η εκτέλεση της παρακάτω εντολής εντός του φακέλου code:

arm-linux-gnueabihf-gcc main.c ./server/server.c ./client/client.c ./message_generator/message_generator.c ./circular_buffer/circular_buffer.c -o main -march=armv6 -mfloat-abi=hard -mfpu=vfp -pthread

Πριν τη μεταγλώττιση πρέπει να ρυθμιστεί το AEM της συσκευής. Πιο συγκεκριμένα, πρέπει να ορισθεί η μεταβλητή myAEM εντός των αρχείων message_generator.c και server.c

Ακόμη, το πρόγραμμα κάνει χρήση της broadcast IP, συνεπώς εαν θέλει κανείς να χρησιμοποιήσει διαφορετικό υποδίκτυο θα πρέπει να αλλάξει την broadcast IP εντός του αρχείο client.c

Επιπλέον, το μέγεθος του circular buffer έχει ορισθεί στις 2000 θέσεις και μπορεί να αλλαχθεί μέσω της μεταβλητής BUFFER_SIZE εντός του αρχείου main.c.

Τέλος, η λίστα με τις IP που έχει συναντήσει η συσκευή έχει ορισθεί στις 50 συσκευές μέσω της μεταβλητής ARRAY_SIZE εντός του αρχείου main.c

Σε ο,τι αφορά την εκτέλεση του προγράμματος, δεν απαιτούνται ορίσματα.

- ./main

Για να εκτελεσθεί το πρόγραμμα και να συνεχίσει να εκτελείται στο background αρκεί η εντολή:

- nohup ./main &

Τέλος, για την εκτέλεση του προγράμματος και την καταγραφή των γεγονότων σε αρχείο log.txt αρκεί η εντολή:

- nohup ./main > log.txt 2>&1 &

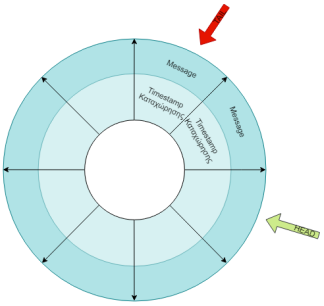
Η εντολή nohup εγγυάται τη συνέχιση της λειτουργίας του προγράμματος ύστερα από την έξοδο της συνεδρίας ssh.

**Σημείωση: το πρόγραμμα δεν μπορεί να εκτελεσθεί σε υπολογιστή διότι διαβάζει τη θερμοκρασία του CPU του Raspberry Pi. Προκειμένου να εκτελεσθεί σε υπολογιστή απαιτούνται ορισμένες τροποποιήσεις στο αρχείο message_generator.c.

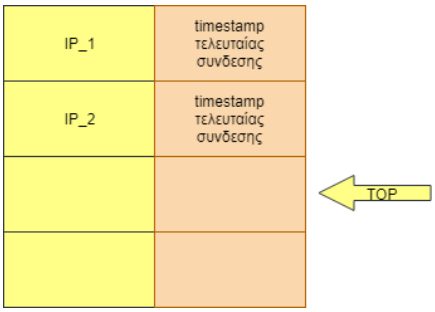
4 Περιγραφή προγράμματος.

4.1 Main function.

Το πρόγραμμα που υλοποιήθηκε εκκινεί με το main thread. Το main thread είναι υπεύθυνο για την αρχικοποίηση του κυκλικού buffer, της λίστας με τις IP των συνδεδεμένων συσκευών και των αντίστοιχων timestamps σύνδεσης. Προφανώς κατά την εκκίνηση, οι πίνακες αυτοί είναι όλοι άδειοι. Υπάρχει, ωστόσο, η δυνατότητα αρχικοποίησης των πινάκων αυτών με συγκεκριμένες τιμές με σκοπό την εκτέλεση προσομοιώσεων. Οι ιδιότητες των παρακάτω δομών θα αναλυθούν στο τμήμα που περιγράφει τον circular buffer.



(α′) Παράλληλος Κυκλικός Buffer. Στον έναν κύκλο περιλαμβάνονται τα μηνύματα και στον άλλο στην αντίστοιχη θέση περιλαμβάνεται το timestamp καταχώρησης.



(β′) Παράλληλος Πίνακας. Στον έναν πίνακα περιλαμβάνονται οι διευθύνσεις IP των συσκευών με τις οποίες έχει πραγματοποιηθεί σύνδεση και στον άλλον το timestamp της τελευταίας σύνδεσης που πραγματοποιήθηκε με τη συσκευή αυτή.

Μετά την αρχικοποίηση των δομών αυτών το main thread δημιουργεί 3 ακόμη threads. Αυτά είναι τα:

- message generator
- client
- server

Ύστερα από τη δημιουργία των παραπάνω threads, το main thread παραμένει αδρανές.

Algorithm 1: MAIN function.

Input: *execTime*: execution time in hours
Output: The circular buffer at the end of experiment

```
1 initialize circular buffer
2 initialize IP list
3 pthread_create(message_generator)
4 pthread_create(client)
5 pthread_create(server)
6 pthread_exit()                    /* wait until the above threads are done */
```

4.2 Circular buffer.

Πριν προχωρήσουμε στην περιγραφή των τριών άλλων threads, κρίνεται σκόπιμη η περιγραφή της δομής circular buffer. Η δομή αυτή, για λόγους ευκολίας, πέραν του κυκλικού buffer υλοποιεί και τη λίστα με τις IP των συνδεδεμένων συσκευών. Αυτό έγινε διότι, τα pthreads επιτρέπουν τη χρήση μόνο ενός ορίσματος και εφόσον η επεξεργασία του circular buffer προυποθέτει και την διερεύνηση της λίστας με τις IP των συσκευών που έχουν συνδεθεί στο παρελθόν, προτιμήθηκε η λίστα με της IP να τοποθετηθεί εντός της δομής αυτής.

```
1  struct circular_buf_t {
2      uint64_t *timestamp;
3      char **message;
4      size_t head;
5      size_t tail;
6      size_t max;
7      bool full;
8      pthread_mutex_t mutex;
9
10
11     char **IPlist;
12     uint64_t *connectionTimestamp;
13     size_t top;
14     size_t listSize;
15 };
```

Παραπάνω φαίνεται η δομή αυτή. Η μεταβλητή **timestamp** και η μεταβλητή **message** συνιστούν τον κυκλικό buffer όπως αυτός παρουσιάστηκε στην εικόνα 1α΄.

Η μεταβλητή **head** δείχνει πάντα μία θέση μπροστά από το πιο πρόσφατο στοιχείο που προστέθηκε στον buffer, ενώ η μεταβλητή **tail** δείχνει πάντα στη θέση του παλαιότερου στοιχείου στον buffer. Η μεταβλητή **max** ισούται με την χωρητικότητα του buffer και η μεταβλητή **full** δείχνει εάν ο circular buffer είναι γεμάτος ή όχι. Τέλος, σε ο,τι αφορά τον κυκλικό buffer, επειδή διαβάζουν και γράφουν σε αυτόν 3 threads ταυτόχρονα, η δομή περιλαμβάνει μία μεταβλητή **mutex** για την ασφαλή ανάγνωση/εγγραφή από και προς τον buffer.

Στο πλαίσιο της εργασίας αυτής ο κυκλικός buffer ορίσθηκε στις 2000 θέσεις. Όταν ο buffer είναι γεμάτος συνεχίζεται η προσθήκη νέων μηνυμάτων με την αντικατάσταση των παλαιότερων.

Αναφορικά με την **IPlist**, σε αυτήν τοποθετούνται οι διευθύνσεις IP των συσκευών με τις οποίες έχει πραγματοποιηθεί επιτυχής σύνδεση στο παρελθόν. Παράλληλα στη λίστα αυτή δημιουργείται μία ακόμη λίστα, η λίστα **connectionTimestamp** η οποία περιλαμβάνει το timestamp της τελευταίας

φοράς που πραγματοποιήθηκε σύνδεση, όπως φαίνεται στο σχήμα 1β΄. Τέλος, η μεταβλητή **top** δίνει τον αριθμό των αποθηκευμένων IPs και επομένως η μεταβλητή αυτή δείχνει μία θέση μετά από τη θέση της τελευταίας(index) καταχώρησης στη λίστα, ενώ η μεταβλητή **listSize** ισούται με τη μέγιστη χωρητικότητα της λίστας.

Για να διασφαλιστεί η ακεραιότητα του προγράμματος υλοποιήθηκε η βιβλιοθήκη circular_buffer.h όπου ορίστηκαν όλες οι συναρτήσεις πρόσβασης στις μεταβλητές της δομής. Αξίζει να αποσαφηνιστεί οτι ο τύπος **cbuf_handle_t** αποτελεί pointer της δομής **circular_buf_t**.

Το σύνολο των συναρτήσεων που δημιουργήθηκαν αποτελείται από συναρτήσεις δημιουργίας/καταστροφής μίας δομής circular_buf_t, προσθήκης, αφαίρεσης και ενημέρωσης στοιχείων. Δύο συναρτήσεις που αξίζει να αναλυθούν περαιτέρω είναι οι:

- void circular_buf_lock(cbuf_handle_t cbuf)
- void circular_buf_unlock(cbuf_handle_t cbuf)

Οι δύο αυτές συναρτήσεις υλοποιούν το lock/unlock της mutex μεταβλητής. Θα μπορούσε κανείς να τοποθετήσει τη λειτουργία αυτή εντός των συναρτήσεων επεξεργασίας/προσθήκης/ενημέρωσης των στοιχείων του circular buffer, τότε όμως θα υπήρχαν πάρα πολλά locks και unlocks τα οποία θα καθυστερούσαν το πρόγραμμα. Συνεπώς με τη χρήση των δύο αυτών συναρτήσεων κλειδώνεται ο buffer κάθε φορά που ένα thread επιθυμεί να εκτελέσει κάποιες διεργασίες στον circular buffer. Με τον τρόπο αυτό πραγματοποιούνται μαζικά οι ενέργειες που επιθυμεί το κάθε thread πολύ ταχύτερα. Αφού ολοκληρώθουν οι εργασίες που εκτελεί το thread που την κλείδωσε, απελευθερώνει τον buffer για να τον δεσμεύσει κάποια άλλη λειτουργία.

Το σύνολο των συναρτήσεων διαχείρισής του circular buffer μπορεί να βρεθεί στο αρχείο **circular_buffer.h**

4.3 Message generator.

Το thread που υλοποιεί τον message generator ουσιαστικά εκτελεί την πιο απλή λειτουργία. Το thread αυτό εισέρχεται σε ένα ατέρμονο loop το οποίο σε κάθε επανάληψη δημιουργεί ένα τυχαίο μήνυμα και το τοποθετεί στον κυκλικό buffer.

Πιο αναλυτικά, ένα μήνυμα ορίζεται ως εξής:

- ΑΕΜαποστολέα_ΑΕΜπαραλήπτη_creationTimestamp_Message

Επομένως, ένα μήνυμα θα αποτελεί ένα String 277 χαρακτήρων. Αυτό προκύπτει ως εξής:

- ΑΕΜπαραλήπτη = 4 χαρακτήρες
- ΑΕΜαποστολέα = 4 χαρακτήρες
- creationTimestamp(timestamp δημιουργίας μηνύματος σε δευτερόλεπτα) = 10 χαρακτήρες
- Message = 256 χαρακτήρες
- 3 χαρακτήρες underscore (_)

Για να έχει νόημα ενσωματωμένου συστήματος, στην παρούσα υλοποίηση επιλέχθηκε το μήνυμα να μεταδίδει μία τιμή από το σύστημα. Συγκεκριμένα, η τιμή αυτή είναι θερμοκρασία του επεξεργαστή του Raspberry pi. Επομένως, ένα τυχαίο μήνυμα θα ήταν:

8721_8462_1562310900_CPU temperature is: 44.96000

Αφού δημιουργηθεί το μήνυμα αυτό τοποθετείται στον κυκλικό buffer προκειμένου να μεταδοθεί σε μελλοντική σύνδεση με άλλη συσκευή ενώ παράλληλα αποθηκεύεται και το timestamp καταχώρησης στον buffer.

Μετά τη δημιουργία του μηνύματος το thread αυτό κοιμάται για τυχαίο χρονικό διάστημα [1, 5] λεπτών.

Ο παραλήπτης επιλέγεται τυχαία από τη λίστα που παρέχεται από το αρχείο AEM.txt.

Algorithm 2: MESSAGE GENERATOR function - random recipient from AEM list.

Input: $cbuf$ pointer to circular buffer struct
Output: $New\_Message$[277] in circular buffer

```
1  AEM_LIST = read(AEM.txt)
2  while TRUE do
3      circular_buf_lock( cbuf)                          /* Lock circular buffer */
4      T = read_CPU_temperature()
5      toAEM = AEM_LIST[rand()]                          /* random AEM from AEM list */
6      gettimeofday(timestamp)
7      New_Message = myAEM_toAEM_temistamp.sec_CPUtemperatureis : T
8      circular_buf_put( cbuf, timestamp.sec, New_Message) /* Add New_Message to
        buffer */
9      circular_buf_unlock(cbuf)                         /* Un-Lock circular buffer */
10     sleep([1, 5] minutes);
```

4.4 Client.

Στην υποενότητα αυτή παρουσιάζεται η λειτουργία του thread που υλοποιεί τον client.

Προκειμένου να έχει χαρακτήρα ενσωματωμένου συστήματος, η συνάρτηση αυτή σκανάρει και εντοπίζει τις συσκευές που βρίσκονται στο τοπικό δίκτυο. Εαν εντοπίσει συσκευές επιχειρεί να επικοινωνήσει μαζί τους. Ύστερα από το πέρας της παραπάνω διαδικασίας, η συσκευή μπαίνει σε sleep mode για 5 δευτερόλεπτα. Η παραπάνω διαδικασία επαναλαμβάνεται διαρκώς.

Όταν η συσκευή μας επιχειρεί να επικοινωνήσει με κάποια άλλη συσκευή, δεν αποστέλλονται όλα τα μηνύματα αλλά αποστέλλονται μόνο τα μηνύματα που δεν έχουν σταλεί στο παρελθόν προς τη συσκευή για επικοινωνία. Πιο αναλυτικά, όταν εντοπίζεται μία συσκευή, ο client αναζητά τη διεύθυνση IP της συνδεδεμένης συσκευής στον πίνακα IPlist. Αν βρεθεί η IP εντός της λίστας αυτής σημαίνει ότι η συσκευή μας έχει συνδεθεί στο παρελθόν ξανά με τη συσκευή αυτή. Τότε, ο client διαβάζει το timestamp κατά το οποίο πραγματοποιήθηκε η τελευταία σύνδεση και αποστέλλει μόνο τα μηνύματα τα οποία προστέθηκαν στον buffer με μεταγενέστερη χρονική στιγμή(timestamp) από αυτή της τελευταίας σύνδεσης. Διαφορετικά, αν είναι η πρώτη φορά που πραγματοποιείται σύνδεση με την νέα συσκευή αποστέλλονται όλα τα μηνύματα του buffer.

Προκειμένου να πραγματοποιηθεί η διαδικασία σκαναρίσματος του τοπικού δικτύου απαιτείται η χρήση ορισμένων εντολών του συστήματος UNIX. Χωρίς, την επιτυχή διαδικασία σκαναρίσματος, ο client δεν μπορεί να προχωρήσει στη διαδικασία αποστολής μηνυμάτων. Οι εντολές που χρησιμοποιούνται για τη διαδικασία σκαναρίσματος είναι οι:

- ip neigh flush all
- ping -c10 -b 10.255.255.255 > /dev/null
- arp -n

Η κύρια εντολή που χρησιμοποιείται για την ανακάλυψη συσκευών στο τοπικό δίκτυο είναι η **arp**. [1] Η εντολή αυτή εμφανίζει τον πίνακα με τις *αποθηκευμένες* διευθύνσεις IP που έχουν αντιστοιχηθεί σε φυσική διεύθυνση χρησιμοποιώντας το πρωτόκολλο ARP (Address Resolution Protocol).

Προκειμένου να τυπώνονται σίγουρα κάθε φορά μόνο οι συσκευές οι οποίες είναι συνδεδεμένες τη δεδομένη χρονική στιγμή στο δίκτυο, απαιτείται ο πίνακας αυτός να καθαρίζεται πριν από κάθε σκανάρισμα. Αυτό επιτυγχάνεται με την εντολή **ip neigh flush all**.

Η εντολή arp από μόνη της δεν είναι αρκετή για να ανακαλυφθεί μία νέα συσκευή στο δίκτυο. Προκειμένου να ανακαλυφθεί μία νέα συσκευή χρησιμοποιείται η εντολή **ping**. Η εντολή αυτή χρησιμοποιείται για την επιβεβαίωση της ικανότητας ενός πόρου να επικοινωνήσει μέσω του δικτύου. Πιο συγκεκριμένα, με την εκτέλεση της εντολής αποστέλλονται ICMP (Internet Control Message Protocol) Echo Messages. Στη συνέχεια, οι συσκευές που βρίσκονται στο τοπικό δίκτυο απαντούν στα πακέτα αυτά και κατ΄ επέκταση είναι πλέον αναγνωρίσιμες από την εντολή arp.

Στην υλοποίηση της εργασίας αυτής αποστέλλονται 10 πακέτα στην broadcast IP του υποδικτύου των διευθύνσεων των Raspberry Pi, δηλαδή στη διεύθυνση 10.255.255.255.

Συσκευές όπως τα Raspberry Pi δεν απαντούν σε πακέτα αυτού του τύπου. Προκειμένου να απαντούν στα πακέτα της εντολής ping οι συσκευές Raspberry Pi απαιτείται να εκτελέσουν την ακόλουθη εντολή πριν την εκτέλεση του προγράμματος:

- echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

Με την παραπάνω εντολή το πρόγραμμα είναι έτοιμο να εκτελέσει τη διαδικασία του σκαναρίσματος. Τώρα με την εκτέλεση της εντολής arp -n, κάθε γραμμή αποτελέσματος περιλαμβάνει μία IPv4. Από κάθε γραμμή εξάγεται η IP αυτή και αποθηκεύεται. Όταν έχουν εξαχθεί όλες οι IPs ο client γνωρίζει όλες τις συσκευές που είναι συνδεδεμένες στο τοπικό δίκτυο του, δηλαδή στο WiFi του και μπορεί να πραγματοποιήσει τη διαδικασία αποστολής μηνυμάτων.

Algorithm 3: CLIENT function.

Input: *cbuf* pointer to circular buffer struct

```
1 while TRUE do
2     IPaddresses = scan()                          /* find connected devices */
3     for i in IPaddreses do
4         sendMessage(IPaddresses[i])   /* Call function to send buffer's messages
           to i_th IP */
5     sleep( 5 seconds)
```

4.4.1 Αποστολή μηνυμάτων του κυκλικού buffer προς μία IP.

Για τη διαδικασία αποστολής μηνυμάτων προς μία IP, ο client καλεί τη συνάρτηση sendMessage(), όπως φαίνεται παραπάνω. Επομένως, κρίνεται σκόπιμο στη συνέχεια της υποενότητας αυτής να παρουσιαστεί η συνάρτηση *sendMessage()*.

Αρχικά, η συνάρτηση αυτή δημιουργεί ένα νέο socket για την επίτευξη της επικοινωνίας με την IP της συσκευής που δόθηκε ως όρισμα στη συνάρτηση. Στη συνέχεια επιδιώκει η sendMessage() να συνδεθεί με τη νέα συσκευή, αν αποτύχει ή αν δε συνδεθεί εντός ενός συγκεκριμένου χρονικού διαστήματος επιστρέφει με το κατάλληλο σφάλμα και τερματίζει την επικοινωνία με τη συσκευή αυτή. Η διαδικασία σύνδεσης των sockets επιτυγχάνεται με τη χρήση της συνάρτησης *connect_ wait()*.

Αν η σύνδεση είναι επιτυχής τότε η συνάρτηση sendMessage() προχωράει. Το επόμενο βήμα είναι να ελέγξει τη λίστα με τις IP των συσκευών με τις οποίες έχει συνδεθεί στο παρελθόν. Η συνάρτηση που πραγματοποιεί τον έλεγχο αυτό είναι η *checkIPlist()*.

Αν η checkIPlist βρει την IP της συνδεδεμένης συσκευής στη λίστα, τότε επιστρέφει τη θέση της στη λίστα και το timestamp κατά το οποίο πραγματοποιήθηκε η τελευταία σύνδεση με τη συσκευή αυτή. Διαφορετικά επιστρέφει αρνητική τιμή η οποία υποδεικνύει ότι η συσκευή αυτή εμφανίζεται για πρώτη φορά.

Στο σημείο αυτό, ξεκινάει η ουσιαστική επικοινωνία που αφορά τη μετάδοση μηνυμάτων και εφεξής οι συναρτήσεις απαιτούν πρόσβαση στον κυκλικό buffer. Συνεπώς, στο σημείο αυτό καλείται η συνάρτηση circular_buf_lock(), η οποία κλειδώνει την πρόσβαση στον κυκλικό buffer.

Αν τώρα η IP ανήκει σε μία IP που έχει βρεθεί στο παρελθόν καλείται η συνάρτηση *set_ start_ index()*. Με δεδομένη την τιμή του timestamp της τελευταίας σύνδεσης που πραγματοποιήθηκε με τη συσκευή, η set_start_index() σκανάρει όλη τη λίστα με τα μηνύματα που βρίσκονται στον buffer έως ότου βρει ένα μήνυμα με timestamp καταχώρησης νεότερο του timestamp της τελευταίας σύνδεσης που πραγματοποιήθηκε με τη συνδεδεμένη συσκευή και επιστέφει τη θέση του μηνύματος στον buffer. Αν δε βρεθεί νεότερο μήνυμα σημαίνει ότι η συσκευή που συνδέθηκε έχει λάβει ήδη όλα τα μηνύματα που υπάρχουν στον buffer μας και η επικοινωνία τερματίζεται. Η τιμή της θέσης που επέστρεψε η set_start_index() δείχνει τη θέση του πρώτου μηνύματος που θα αποσταλεί από τον κυκλικό buffer προς τη συνδεδεμένη συσκευή. Επίσης, ενημερώνεται και το timestamp της τελευταίας σύνδεσης της συσκευής αυτής με το τωρινό timestamp.

Αν τώρα η συνδεδεμένη συσκευή συνδέεται για πρώτη φορά με τη συσκευή μας, το πρώτο μήνυμα που θα αποσταλεί προς τη συσκευή αυτή θα είναι το αρχαιότερο χρονικά, δηλαδή αυτό που βρίσκεται στη θέση *tail* του buffer.

Στο σημείο αυτό έχει ορισθεί το πρώτο μήνυμα που θα αποσταλεί προς τη συνδεδεμένη συσκευή και αυτό που απομένει είναι να εκκινήσει η διαδικασία αποστολής.

Με την εκκίνηση της διαδικασίας της αποστολής αποστέλλονται ένα ένα τα μηνύματα από το πρώτο μήνυμα που ορίσθηκε προς αποστολή (*αρχαιότερο χρονικά*) έως το μήνυμα που βρίσκεται στη θέση head του buffer (νεότερο χρονικά).

*Σημείωση: στον πηγαίο κώδικα έχουν ληφθεί υπόψιν και οι υποπεριπτώσεις της διακοπής της αποστολής λόγω κάποιου προβλήματος(ERROR) ή λόγω μη απόκρισης του server (TIMEOUT) οι οποίες δεν παρουσιάζονται στον ψευδοκώδικα προκειμένου αυτός να είναι πιο κατανοητός.

Ύστερα από την ολοκλήρωση των αποστολών, ξεκλειδώνεται ο buffer και επιστρέφει η λειτουργία στη συνάρτηση client(), η οποία είτε συνεχίζει με την επόμενη συσκευή ή κοιμάται αν η συσκευή αυτή ήταν η τελευταία.

Algorithm 4: SEND MESSAGE function.

Input: $cbuf$ pointer to circular buffer struct, IP address of the connected device

```
1  create socket
2  status = connect_wait()
3  if status == -1 then
4  |   error
5  |   return -1
6  else if status == 1 then
7  |   timeout
8  |   return -1
9  checkIPlist(IP)              /* check whether IP connects for the 1st time.  */
10 circular_buf_lock(cbuf)
11 if IP exists then
12 |   index = set_start_index(IP_last_connection_Timestamp)
13 |   update(IP_last_connection_Timestamp)
14 else
15 |   index = circular_buf_get_tail(cbuf)
16 |   add_to_IPlsit( IP, timestamp.sec)
17 for i from index until reaches head do
18 |   send(cbuf.message[i])
19 circular_buf_unlock(cbuf)
20 return
```

4.5 Server.

Στην υποενότητα αυτή περιγράφεται η λειτουργία του Server thread. Η συνάρτηση server είναι και αυτή ένα ατέρμονο loop το οποίο αναμένει διαρκώς νέα μηνύματα. Αρχικά δημιουργεί ένα socket και αναμένει κάποια συσκευή να επικοινωνήσει μαζί του.

Από τη στιγμή που κάποια συσκευή συνδεθεί επιτυχώς στο server τότε ξεκινάει η διαδικασία λήψης μηνυμάτων, επομένως πρέπει να κλειδωθεί ο buffer.

Για κάθε νέο μήνυμα που λαμβάνεται, ελέγχεται με τη συνάρτηση checkForDuplicate() αν το μήνυμα αυτό υπάρχει ήδη στον buffer. Αν υπάρχει ήδη απορρίπτεται, διαφορετικά ελέγχεται αν εγώ είμαι ο τελικός αποδέκτης του μηνύματος, εξετάζοντας το δεύτερο μέρος του μηνύματος που περιλαμβάνει το AEM του παραλήπτη. Εαν είμαι εγώ ο παραλήπτης δεν αποθηκεύω το μήνυμα στον buffer. Διαφορετικά αποθηκεύω το μήνυμα στον buffer και το timestamp που λήφθηκε το μήνυμα.

Εάν κλείσει το socket από την πλευρά της άλλης συσκευής. η λήψη μηνυμάτων θεωρείται επιτυχής. Διαφορετικά ελέγχεται αν υπήρξε κάποιο σφάλμα ή κάποιο timeout στην επικοινωνία.

Algorithm 5: SERVER function.

```
   Input: cbuf pointer to circular buffer struct
 1 create socket
 2 listen to socket
 3 while TRUE do
 4     incoming_socket = accept()          /* New device granted communication */
 5     circular_buf_lock(cbuf)
 6     while TRUE do
 7         status = read()                                        /* read message */
 8         if status < 0 then
 9             handle error
10             circular_buf_unlock(cbuf)
11             break
12         else if status == 0 then
13             socket closed successfully
14             circular_buf_unlock(cbuf)
15             break
16         else
17             if checkForDuplicate() > 0 then
                                                  /* If no duplicate was found  */
18                 if amIrecipient() then
19                     i received the message
20                 else
21                     circular_buf_put( message timestamp, sec)
22             else
23                 message has already been received
```

5 Αποτελέσματα.

Στο πλαίσιο των δοκιμών που πραγματοποιήθηκαν, χρησιμοποιήθηκαν 2 Raspberry Pi. Στη μία συσκευή δόθηκε το AEM 8462 και κατ΄ επέκταση η IP 10.0.84.62 και στην άλλη συσκευή το AEM 8535 και η IP 10.0.85.35.

Σε κάθε συσκευή δόθηκε και μία λίστα AEM.txt με παραλήπτες. Οι παραλήπτες της συσκευής 8462 ήταν:

- 8535
- 8796
- 8931
- 8006

ενώ οι παραλήπτες της συσκευής 8535 ήταν:

- 8462
- 8963
- 8526
- 8478

Εσκεμμένα χρησιμοποιήθηκε ως παραλήπτης η συσκευή 8535 από την 8462 και αντίστροφα προκειμένου να είναι εμφανής η λειτουργία επιτυχούς παράδοσης μηνύματος στον παραλήπτη.

Οι συσκευές αφέθηκαν για μία ώρα να λειτουργούν. Στη συνέχεια τερματίστηκε η λειτουργία τους και μελετήθηκαν τα παραγόμενα αρχεία. Κάθε συσκευή παρήγαγε ένα αρχείο log.txt και ένα αρχείο circular_buffer.txt.

Το αρχείο log.txt της συσκευής 8462 αναλύθηκε στο matlab και έδωσε τα παρακάτω αποτελέσματά:

Ο MESSAGE GENERATOR παρήγαγε **20** μηνύματα:

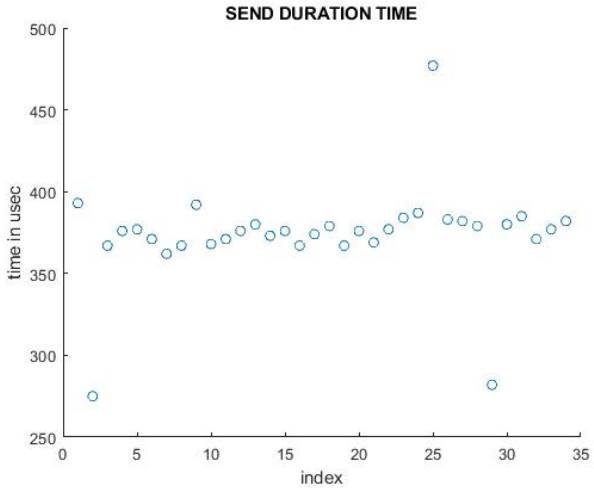
```
1  8462_8796_1554444962_CPU temperature is : 40.084000
2  8462_8796_1554445104_CPU temperature is : 41.160000
3  8462_8535_1554445286_CPU temperature is : 43.312000
4  8462_8006_1554445515_CPU temperature is : 44.388000
5  8462_8931_1554445591_CPU temperature is : 44.926000
6  8462_8535_1554445839_CPU temperature is : 45.464000
7  8462_8535_1554446063_CPU temperature is : 44.926000
8  8462_8796_1554446234_CPU temperature is : 45.464000
9  8462_8931_1554446302_CPU temperature is : 46.002000
10 8462_8006_1554446412_CPU temperature is : 45.464000
11 8462_8006_1554446689_CPU temperature is : 46.540000
12 8462_8796_1554446909_CPU temperature is : 46.540000
13 8462_8006_1554447071_CPU temperature is : 46.540000
14 8462_8796_1554447328_CPU temperature is : 47.078000
15 8462_8796_1554447409_CPU temperature is : 46.540000
16 8462_8931_1554447679_CPU temperature is : 47.616000
17 8462_8535_1554447946_CPU temperature is : 47.078000
18 8462_8006_1554448189_CPU temperature is : 47.616000
19 8462_8006_1554448468_CPU temperature is : 47.078000
20 8462_8796_1554448533_CPU temperature is : 47.616000
```

SERVER:

- Ο SERVER επέτρεψε **257** φορές σε ξένη συσκευή να επικοινωνήσει μαζί του.
- Έλαβε **36** μηνύματα εκ των οποίων τα **22** ήταν νέα μηνύματα.
- Από τα 22 μηνύματα **6** μηνύματα είχαν ως τελικό αποδέκτη τη συσκευή 8462.
- Από τα 36 μηνύματα που λήφθηκαν τα **14** απορρίφθηκαν διότι υπήρχαν ήδη στον circular buffer.
- Κατά τη διάρκεια των δοκιμών δεν παρουσιάστηκαν ERRORS ή TIMEOUTS. 257 φορές το socket της επικοινωνίας έκλεισε επιτυχώς.

CLIENT:

- Ο CLIENT απέστειλε επιτυχώς **34** μηνύματα.
- 223 φορές ο CLIENT δεν απέστειλε μήνυμα διότι τα είχε όλα ήδη αποστείλει.
- Η μέση διάρκεια αποστολής μηνύματος ήταν **373.58** usecs.
- Η τυπική απόκλισή ήταν **30.52** usecs.



Σχήμα 2: Διάρκεια αποστολής μηνυμάτων της συσκευής 8462.

Ο circular buffer και τα αντίστοιχο timestamp καταχώρησης στην τελική τους κατάσταση ύστερα από μία ώρα δοκιμών φαίνονται παρακάτω:

```
8462_8796_1554444962_CPU temperature is : 40.084000     1554444962
8462_8796_1554445104_CPU temperature is : 41.160000     1554445104
8535_8963_1554456601_CPU temperature is : 39.008000     1554445202
8462_8535_1554445286_CPU temperature is : 43.312000     1554445286
8535_8526_1554456772_CPU temperature is : 39.008000     1554445371
8462_8006_1554445515_CPU temperature is : 44.388000     1554445515
8462_8931_1554445591_CPU temperature is : 44.926000     1554445591
```

```
8535_8478_1554457132_CPU temperature is : 39.008000     1554445723
8462_8535_1554445839_CPU temperature is : 45.464000     1554445839
8535_8478_1554457346_CPU temperature is : 39.008000     1554445935
8462_8535_1554446063_CPU temperature is : 44.926000     1554446063
8535_8963_1554457582_CPU temperature is : 39.546000     1554446174
8462_8796_1554446234_CPU temperature is : 45.464000     1554446234
8462_8931_1554446302_CPU temperature is : 46.002000     1554446302
8535_8963_1554457730_CPU temperature is : 39.546000     1554446329
8462_8006_1554446412_CPU temperature is : 45.464000     1554446412
8462_8006_1554446689_CPU temperature is : 46.540000     1554446689
8535_8478_1554458214_CPU temperature is : 39.546000     1554446808
8462_8796_1554446909_CPU temperature is : 46.540000     1554446909
8535_8478_1554458463_CPU temperature is : 39.546000     1554447062
8462_8006_1554447071_CPU temperature is : 46.540000     1554447071
8535_8526_1554458581_CPU temperature is : 39.546000     1554447174
8462_8796_1554447328_CPU temperature is : 47.078000     1554447328
8535_8963_1554458759_CPU temperature is : 39.008000     1554447358
8462_8796_1554447409_CPU temperature is : 46.540000     1554447409
8535_8478_1554458827_CPU temperature is : 39.546000     1554447428
8535_8963_1554458985_CPU temperature is : 39.008000     1554447583
8462_8931_1554447679_CPU temperature is : 47.616000     1554447679
8535_8963_1554459246_CPU temperature is : 39.546000     1554447836
8462_8535_1554447946_CPU temperature is : 47.078000     1554447946
8462_8006_1554448189_CPU temperature is : 47.616000     1554448189
8535_8526_1554459642_CPU temperature is : 40.084000     1554448231
8535_8526_1554459826_CPU temperature is : 39.008000     1554448414
8462_8006_1554448468_CPU temperature is : 47.078000     1554448468
8535_8478_1554459892_CPU temperature is : 40.084000     1554448484
8462_8796_1554448533_CPU temperature is : 47.616000     1554448533
```

Σε ο,τι αφορά τη συσκευή 8535 τα αντίστοιχα αποτελέσματα φαίνονται παρακάτω.

Ο MESSAGE GENERATOR παρήγαγε **23** μηνύματα:

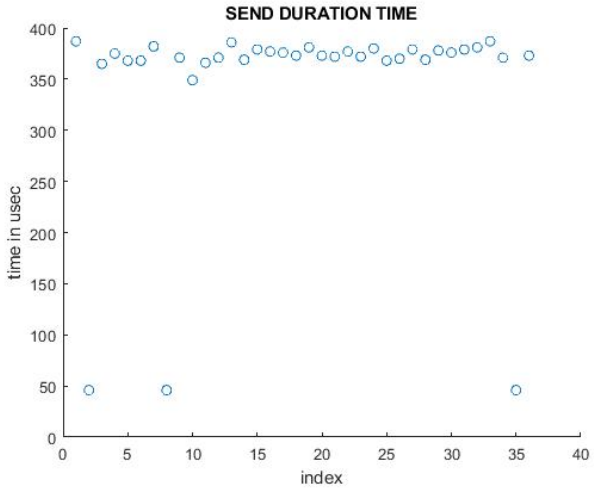
```
8535_8462_1554456366_CPU temperature is : 38.470000
8535_8963_1554456601_CPU temperature is : 39.008000
8535_8526_1554456772_CPU temperature is : 39.008000
8535_8462_1554457007_CPU temperature is : 39.008000
8535_8478_1554457132_CPU temperature is : 39.008000
8535_8478_1554457346_CPU temperature is : 39.008000
8535_8963_1554457582_CPU temperature is : 39.546000
8535_8963_1554457730_CPU temperature is : 39.546000
8535_8462_1554457954_CPU temperature is : 39.008000
8535_8462_1554458042_CPU temperature is : 39.008000
8535_8462_1554458154_CPU temperature is : 39.546000
8535_8478_1554458214_CPU temperature is : 39.546000
8535_8478_1554458463_CPU temperature is : 39.546000
8535_8526_1554458581_CPU temperature is : 39.546000
8535_8963_1554458759_CPU temperature is : 39.008000
8535_8478_1554458827_CPU temperature is : 39.546000
8535_8963_1554458985_CPU temperature is : 39.008000
8535_8963_1554459246_CPU temperature is : 39.546000
8535_8462_1554459383_CPU temperature is : 39.546000
8535_8526_1554459642_CPU temperature is : 40.084000
8535_8526_1554459826_CPU temperature is : 39.008000
8535_8478_1554459892_CPU temperature is : 40.084000
8535_8963_1554459999_CPU temperature is : 39.546000
```

SERVER:

- Ο SERVER επέτρεψε **258** φορές σε ξένη συσκευή να επικοινωνήσει μαζί του.
- Έλαβε **34** μηνύματα εκ των οποίων τα **18** ήταν νέα μηνύματα.
- Από τα 18 μηνύματα **4** μηνύματα είχαν ως τελικό αποδέκτη τη συσκευή 8535.
- Από τα 34 μηνύματα που λήφθηκαν τα **16** απορρίφθηκαν διότι υπήρχαν ήδη στον circular buffer.
- Κατά τη διάρκεια των δοκιμών δεν παρουσιάστηκαν ERRORS ή TIMEOUTS.
- 258 φορές το socket της επικοινωνίας έκλεισε επιτυχώς.

CLIENT:

- Ο CLIENT απέστειλε επιτυχώς **36** μηνύματα.
- 224 φορές ο CLIENT δεν απέστειλε μήνυμα διότι τα είχε όλα ήδη αποστείλει.
- Η μέση διάρκεια αποστολής μηνύματος ήταν **346.83** usecs.
- Η τυπική απόκλιση αποστολής μηνύματος ήταν **92.26** usecs.



Σχήμα 3: Διάρκεια αποστολής μηνυμάτων της συσκευής 8535.

Ο circular buffer και τα αντίστοιχο timestamp καταχώρησης στην τελική τους κατάσταση ύστερα από μία ώρα δοκιμών φαίνονται παρακάτω:

```
8535_8462_1554456366_CPU temperature is : 38.470000     1554456366
8462_8796_1554444962_CPU temperature is : 40.084000     1554456384
8462_8796_1554445104_CPU temperature is : 41.160000     1554456525
8535_8963_1554456601_CPU temperature is : 39.008000     1554456601
8535_8526_1554456772_CPU temperature is : 39.008000     1554456772
8462_8006_1554445515_CPU temperature is : 44.388000     1554456934
8462_8931_1554445591_CPU temperature is : 44.926000     1554457004
8535_8462_1554457007_CPU temperature is : 39.008000     1554457007
8535_8478_1554457132_CPU temperature is : 39.008000     1554457132
8535_8478_1554457346_CPU temperature is : 39.008000     1554457346
8535_8963_1554457582_CPU temperature is : 39.546000     1554457582
8462_8796_1554446234_CPU temperature is : 45.464000     1554457652
8462_8931_1554446302_CPU temperature is : 46.002000     1554457722
8535_8963_1554457730_CPU temperature is : 39.546000     1554457730
8462_8006_1554446412_CPU temperature is : 45.464000     1554457835
8535_8462_1554457954_CPU temperature is : 39.008000     1554457954
8535_8462_1554458042_CPU temperature is : 39.008000     1554458042
8462_8006_1554446689_CPU temperature is : 46.540000     1554458103
8535_8462_1554458154_CPU temperature is : 39.546000     1554458154
8535_8478_1554458214_CPU temperature is : 39.546000     1554458214
8462_8796_1554446909_CPU temperature is : 46.540000     1554458329
8535_8478_1554458463_CPU temperature is : 39.546000     1554458463
8535_8526_1554458581_CPU temperature is : 39.546000     1554458581
8462_8796_1554447328_CPU temperature is : 47.078000     1554458751
8535_8963_1554458759_CPU temperature is : 39.008000     1554458759
8535_8478_1554458827_CPU temperature is : 39.546000     1554458827
8535_8963_1554458985_CPU temperature is : 39.008000     1554458985
8462_8931_1554447679_CPU temperature is : 47.616000     1554459103
8535_8963_1554459246_CPU temperature is : 39.546000     1554459246
8535_8462_1554459383_CPU temperature is : 39.546000     1554459383
8462_8006_1554448189_CPU temperature is : 47.616000     1554459610
8535_8526_1554459642_CPU temperature is : 40.084000     1554459642
8535_8526_1554459826_CPU temperature is : 39.008000     1554459826
8535_8478_1554459892_CPU temperature is : 40.084000     1554459892
8462_8006_1554448468_CPU temperature is : 47.078000     1554459892
```

```
8462_8796_1554448533_CPU temperature is : 47.616000      1554459948
8535_8963_1554459999_CPU temperature is : 39.546000      1554459999
```

Παρακάτω φαίνεται ενδεικτικά μέρος του log.txt του 8462.

```
WARNING: pinging broadcast address
MESSAGE GENERATOR: 4 AEMs READ from file.
MUTEX: LOCK
MESSAGE GENERATOR: 8462_8796_1554444962_CPU temperature is : 40.084000
CIRCULAR BUFFER PRINTED TO FILE
MUTEX: UNLOCK
CLIENT: ACTIVE CONNECTIONS: 1
CLIENT: NEW CONNECTION WITH 10.0.85.35  TIMESTAMP: 1554444972 secs.
MUTEX: LOCK
CLIENT: FIRST TIME MET IP: 10.0.85.35
CLIENT: MESSAGE 8462_8796_1554444962_CPU temperature is : 40.084000 SENT
    SUCCESSFULLY in 393 usecs.
MUTEX: UNLOCK
CLIENT: MESSAGES SENT TO 10.0.85.35 : 1
SERVER: 10.0.85.35 GRANTED COMMUNICATION
MUTEX: LOCK
SERVER: MESSAGE RECEIVED: 8535_8462_1554456366_CPU temperature is : 38.470000
SERVER: NEW MESSAGE RECEIVED
SERVER: NEW MESSAGE ARRIVED AT ITS DESTINATION
SERVER: MESSAGE RECEIVED: 8462_8796_1554444962_CPU temperature is : 40.084000
SERVER: MESSAGE HAS ALREADY BEEN MET
SERVER: SOCKET HAS BEEN CLOSED SUCCESSFULLY
CIRCULAR BUFFER PRINTED TO FILE
MUTEX: UNLOCK
```

Παρακάτω φαίνεται το αντιστοιχο μέρος του log.txt του 8535.

```
WARNING: pinging broadcast address
MESSAGE GENERATOR: 4 AEMs READ from file.
MUTEX: LOCK
MESSAGE GENERATOR: 8535_8462_1554456366_CPU temperature is : 38.470000
CIRCULAR BUFFER PRINTED TO FILE
MUTEX: UNLOCK
WARNING: pinging broadcast address
CLIENT: ACTIVE CONNECTIONS: 1
CLIENT: NEW CONNECTION WITH 10.0.84.62  TIMESTAMP: 1554456375 secs.
CLIENT: ERROR in sendMessage(). DID NOT CONNECT TO SERVER. SERVER DOES NOT EXIST
    ON THE OTHER SIDE.
CLIENT: COMMUNICATION TERMINATED UNSUCCESSFULLY WITH IP: 10.0.84.62
SERVER: 10.0.84.62 GRANTED COMMUNICATION
MUTEX: LOCK
SERVER: MESSAGE RECEIVED: 8462_8796_1554444962_CPU temperature is : 40.084000
SERVER: NEW MESSAGE RECEIVED
SERVER: SOCKET HAS BEEN CLOSED SUCCESSFULLY
CIRCULAR BUFFER PRINTED TO FILE
MUTEX: UNLOCK
SERVER: MESSAGES RECEIVED SUCCESSFULLY FROM 10.0.84.62: 1. MESSAGES SAVED: 1
WARNING: pinging broadcast address
CLIENT: ACTIVE CONNECTIONS: 1
MUTEX: LOCK
CLIENT: FIRST TIME MET IP: 10.0.84.62
CLIENT: MESSAGE 8535_8462_1554456366_CPU temperature is : 38.470000 SENT
    SUCCESSFULLY in 387 usecs.
CLIENT: MESSAGE 8462_8796_1554444962_CPU temperature is : 40.084000 SENT
    SUCCESSFULLY in 46 usecs.
MUTEX: UNLOCK
CLIENT: MESSAGES SENT TO 10.0.84.62 : 2
```

Το σύνολο των αποτελεσμάτων και των παραγόμενων αρχείων μπορούν να βρεθούν εντός των γακέλων matlab και results.

6 Επίλογος.

Στο έγγραφο αυτό παρουσιάστηκε η εργασία εξαμήνου του μαθήματος Ενσωματωμένα Συστήματα Πραγματικού Χρόνου. Στο πλαίσιο της εργασίας αυτής κατασκευάστηκε ένα ενσωματωμένο σύστημα διαχείρισης μηνυμάτων. Το σύστημα αυτό διαχωρίζεται σε τρία κομμάτια. Το ένα κομμάτι αφορά την παραγωγή τυχαίων μηνυμάτων. Το δεύτερο κομμάτι αφορά τη λήψη μηνυμάτων και τη σωστή διαχείριση τους. Τέλος, το τρίτο κομμάτι αφορά τη σωστή αποστολή μηνυμάτων προς μία συνδεδεμένη συσκευή.

Το πρόγραμμα που υλοποιήθηκε δοκιμάσθηκε και τα αποτελέσματα των δοκιμών παρουσίασαν θετικά αποτελέσματα λειτουργίας.

Αναφορές

[1] Michael Kerrisk, 'Linux System Administrator's Manual.' `http://man7.org/`.