

**ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΔΙΚΤΥΩΝ ΚΑΙ ΚΙΝΗΤΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΕΡΓΑΣΙΑ 3**

ΑΝΑΠΤΥΞΗΣ ΕΡΓΑΣΙΑΣ ΑΠΟ:

ΙΩΑΝΝΗΣ ΙΑΣΩΝ ΜΠΟΪΔΑΝΗΣ – Π19217

*Μάιος 2023*

# ΠΕΡΙΕΧΟΜΕΝΑ

[ΠΕΡΙΕΧΟΜΕΝΑ .2](#_Toc37047)

[ΘΕΜΑ ΕΡΓΑΣΙΑΣ .2](#_Toc37048)

[2. Μέρος Α1 .](#_Toc37049)3

[3. Μέρος Α2 9](#_Toc37050)

3.1 Α2.1………………………………………………………………………………………9

3.2 Α2.2…………………………………………………………………………….12

3.3 Α2.3………………………………………………………………………………13

3.4 Α2.4………………………………………………………………………………13

3.5 Α2.5…………………………………………………………………………………..14

Θέμα Εργασίας

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, έγγραφο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, έγγραφο, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, έγγραφο

Περιγραφή που δημιουργήθηκε αυτόματα

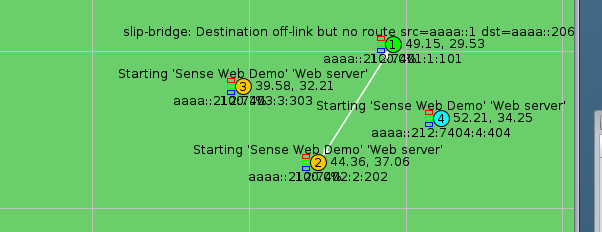
Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, ιστοσελίδα

Περιγραφή που δημιουργήθηκε αυτόματα

2. Μέρος Α1

Στην αρχή δημιουργώ ένα simulation με όνομα erg\_ask3.

Έπειτα βάζουμε έναν κόμβο(κόμβος 1) με το script border-router και άλλους τρεις κόμβους με το script sky-websence(κόμβοι 2-4).



Αφού έχουμε εκκινήσει το πρόγραμμα και έχουμε βάλει τους κόμβους μας ζητείται να κατευθυνθούμε στο terminal όπου αναζητούμε το παρακάτω directory. Στην συνέχεια εκτελούμε την εντολή cd contiki-2.7/examples/ipv6/rpl-border-router make connect-router-cooja αφού έχουμε εκκινήσει το simulation.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, κατάλογος

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Έπειτα αντιγράφουμε την πρώτη διεύθυνση(ή την δεύτερη) του κόμβου και την κάνουμε ping από ένα νέο terminal.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Γραφικά από μία ιστοσελίδα μπορούμε να ελέγξουμε ότι ο κόμβος είναι ενεργός

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Μετά από λίγη ώρα βλέπει και τους απομακρυσμένους κόμβους.

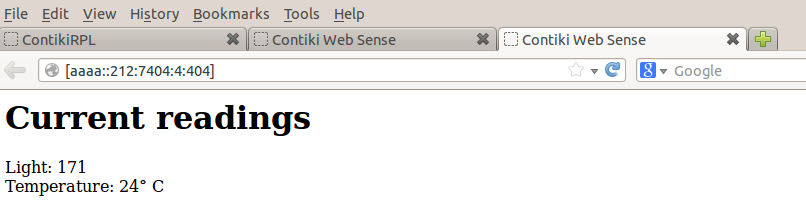
Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Ας δούμε για παράδειγμα τις μετρήσεις των υπόλοιπων κόμβων

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα



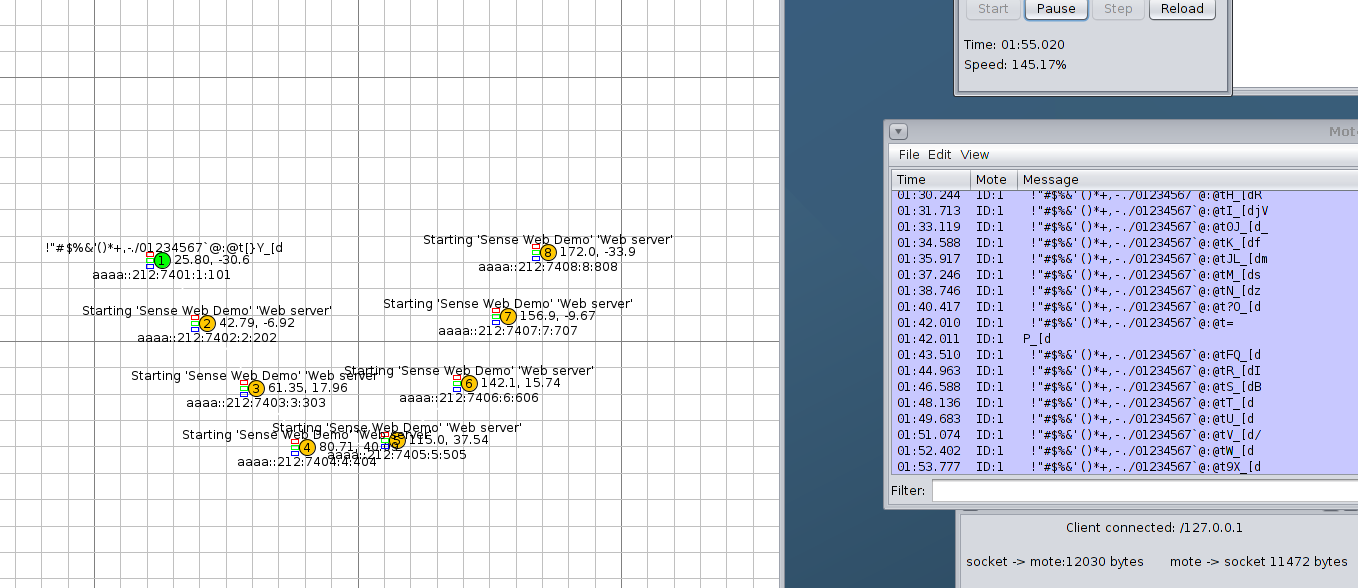
Εικόνα που περιέχει κείμενο, λογισμικό, γραμματοσειρά, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

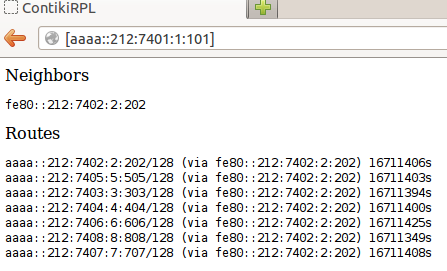
3. Μέρος Α2

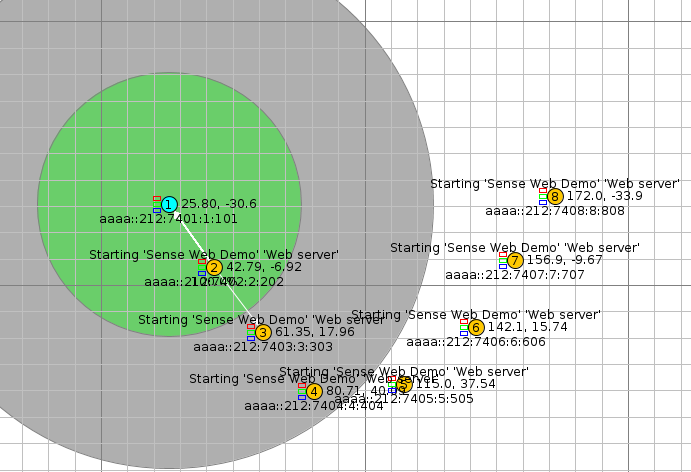
3.1 Α2.1

Προσθέτουμε τους ζητούμενους κόμβους και φτιάχνουμε μία ελλειψοειδή δομή έτσι ώστε να δημιουργηθεί ένα δίκτυο με βάθος τουλάχιστον 2 hops.



Όπως βλέπουμε ο ρούτερ μας επικοινωνεί με όλους τους κόμβους παρόλο που βλέπει μόνο μέχρι τον 2ο κόμβο.





Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, κύκλος, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Πράγματι βλέπουμε την ροή της επικοινωνίας μεταξύ των γωνιακών κόμβων.

Εικόνα που περιέχει κείμενο, γραμμή, γράφημα, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

3.2 Α2.2

Η εφαρμογή sky-websense αποτελείται από δύο διαφορετικά threads που τρέχουν παράλληλα: το web server thread και το sensing thread. Το web server thread είναι υπεύθυνο για την εξυπηρέτηση αιτημάτων από πελάτες μέσω του πρωτοκόλλου HTTP, ενώ το sensing thread χρησιμοποιεί διάφορους αισθητήρες για τη μέτρηση της φωτεινότητας και της θερμοκρασίας, και σε δική μας τροποποίηση μπορεί να μετρήσει και την υγρασία.

Αρχικά, το sensing thread αρχικοποιεί τους αισθητήρες και δημιουργεί έναν πίνακα ιστορικού για την αποθήκευση των μετρήσεων. Στη συνέχεια, ξαναμετράει συνεχώς τον κύκλο μέτρησης, αποθηκεύοντας τις τιμές στον πίνακα ιστορικού και ενημερώνοντας τις τελευταίες τιμές που έχουν μετρηθεί.

Το web server thread είναι ένας διακομιστής ιστού που λειτουργεί σε έναν από τους αισθητήρες (sensor nodes) του δικτύου αισθητήρων και χρησιμοποιεί το πρωτόκολλο HTTP για να εξυπηρετεί αιτήσεις από τους πελάτες (clients) που συνδέονται στο δίκτυο. Ο κώδικας του web server βρίσκεται στη συνάρτηση process\_http, όπου παραλαμβάνεται και επεξεργάζεται οποιαδήποτε αίτηση HTTP καταχωρείται από τους πελάτες.

Επιπλέον, υπάρχουν δύο σελίδες στον ιστότοπο που ελέγχουν τα LEDs που υπάρχουν στη συσκευή.

3.3 Α2.3

Το script border-router.c υλοποιεί έναν δρομολογητή για ένα ασύρματο δίκτυο αισθητήρων. Βασικός στόχος της είναι να διασυνδέσει τα ασύρματα δίκτυα αισθητήρων με το Internet. Για να επιτευχθεί αυτό, η εφαρμογή χρησιμοποιεί το πρωτόκολλο IPv6 για τη διασύνδεση των ασύρματων δικτύων και του Internet.

Ο δρομολογητής αυτός λειτουργεί σαν μια πύλη (gateway) μεταξύ του ασύρματου δικτύου αισθητήρων και του ενσύρματου Internet. Πιο συγκεκριμένα, λαμβάνει τα πακέτα από το ασύρματο δίκτυο αισθητήρων, τα μετατρέπει σε IPv6 πακέτα και τα δρομολογεί στο ενσύρματο Internet, και αντίστροφα. Επίσης, παρέχει υπηρεσίες διαμοιρασμού διευθύνσεων, διαχείρισης δικτύου και αποστολής εντολών στους αισθητήρες.

Η υλοποίηση του script border-router βρίσκεται στο αρχείο border-router.c. To script εκκινείται με την κλήση της συνάρτησης "border\_router\_init()". Κατά την εκκίνηση, δημιουργείται μια νέα διεπαφή δικτύου, εκκινεί τον IPv6 δρομολογητή και συνδέεται με το Internet. Επίσης μπορεί να γίνει τη δυναμική ανάθεση διευθύνσεων IPv6 στις συσκευές του IoT και υποστηρίζει τη λειτουργία του αυτόματου δρομολογητή.

3.4 Α2.4

Ο webserver ενός κόμβου websense έχει τις εξής δυνατές υπηρεσίες :

Ενημέρωση των ρυθμίσεων του κόμβου, όπως η αλλαγή του μετρητή φωτεινότητας ή θερμοκρασίας για την εκκίνηση του sensing thread.

Ελέγχους και διαχείριση του δικτύου, όπως η αναζήτηση και η προβολή των διαθέσιμων δικτυακών κόμβων ή η διαμόρφωση του δικτύου μέσω της αποστολής κατάλληλων πακέτων.

Εμφάνιση της κατάστασης του κόμβου, όπως η εμφάνιση των τρεχουσών μετρήσεων της φωτεινότητας και της θερμοκρασίας, του επιπέδου μπαταρίας κ.λπ.

Επικοινωνία με άλλους κόμβους στο δίκτυο, όπως η αποστολή μηνυμάτων και η λήψη δεδομένων από άλλους κόμβους.

3.5 Α2.5

Για δική μας ευκολία αν μελετήσουμε το αρχείο κώδικα sky-websense υπάρχει μέσα το αρχείο sht11-sensor μπορούμε εύκολα να το χρησιμοποιήσουμε για να προσθέσουμε την λειτουργία της μέτρησης της θερμοκρασίας.

Συγκεκριμένα αν προσθέσουμε τις εξής γραμμες:

static int

get\_humidity(void)

{

return sht11\_sensor.value(SHT11\_SENSOR\_HUMIDITY) / 100;

}

Και:

"Humidity: %u<br>"

"Temperature: %u&deg; C",

get\_light(),get\_humidity(), get\_temp());

θα μπορέσουμε να εμφανίσουμε εύκολα τις μετρήσεις της υγρασίας. Όπου αντίστοιχα με την μέτρηση της θερμοκρασίας δεν αλλάζει συχνά. Στην δική μου υλοποίηση δεν αποθηκεύω την μέτρηση σε έναν πίνακα ιστορικού κι απλά την εμφανίζω. Επίσης διαιρώ με το 100 για να απεικονίζει το ποσοστό υγρασίας γύρω του κόμβου. Πχ 41%

Ο τροποποιημένος κώδικας:

#include "contiki.h"

#include "httpd-simple.h"

#include "dev/sht11-sensor.h"

#include "dev/light-sensor.h"

#include "dev/leds.h"

#include <stdio.h>

PROCESS(web\_sense\_process, "Sense Web Demo");

PROCESS(webserver\_nogui\_process, "Web server");

PROCESS\_THREAD(webserver\_nogui\_process, ev, data)

{

PROCESS\_BEGIN();

httpd\_init();

while(1) {

PROCESS\_WAIT\_EVENT\_UNTIL(ev == tcpip\_event);

httpd\_appcall(data);

}

PROCESS\_END();

}

AUTOSTART\_PROCESSES(&web\_sense\_process,&webserver\_nogui\_process);

#define HISTORY 16

static int temperature[HISTORY];

static int light1[HISTORY];

static int sensors\_pos;

/\*---------------------------------------------------------------------------\*/

static int

get\_light(void)

{

return 10 \* light\_sensor.value(LIGHT\_SENSOR\_PHOTOSYNTHETIC) / 7;

}

/\*---------------------------------------------------------------------------\*/

static int

get\_temp(void)

{

return ((sht11\_sensor.value(SHT11\_SENSOR\_TEMP) / 10) - 396) / 10;

}

static int

get\_humidity(void)

{

return sht11\_sensor.value(SHT11\_SENSOR\_HUMIDITY) / 100;

}

/\*---------------------------------------------------------------------------\*/

static const char \*TOP = "<html><head><title>Contiki Web Sense</title></head><body>\n";

static const char \*BOTTOM = "</body></html>\n";

/\*---------------------------------------------------------------------------\*/

/\* Only one single request at time \*/

static char buf[256];

static int blen;

#define ADD(...) do { \

blen += snprintf(&buf[blen], sizeof(buf) - blen, \_\_VA\_ARGS\_\_); \

} while(0)

static void

generate\_chart(const char \*title, const char \*unit, int min, int max, int \*values)

{

int i;

blen = 0;

ADD("<h1>%s</h1>\n"

"<img src=\"http://chart.apis.google.com/chart?"

"cht=lc&chs=400x300&chxt=x,x,y,y&chxp=1,50|3,50&"

"chxr=2,%d,%d|0,0,30&chds=%d,%d&chxl=1:|Time|3:|%s&chd=t:",

title, min, max, min, max, unit);

for(i = 0; i < HISTORY; i++) {

ADD("%s%d", i > 0 ? "," : "", values[(sensors\_pos + i) % HISTORY]);

}

ADD("\">");

}

static

PT\_THREAD(send\_values(struct httpd\_state \*s))

{

PSOCK\_BEGIN(&s->sout);

SEND\_STRING(&s->sout, TOP);

if(strncmp(s->filename, "/index", 6) == 0 ||

s->filename[1] == '\0') {

/\* Default page: show latest sensor values as text (does not

require Internet connection to Google for charts). \*/

blen = 0;

ADD("<h1>Current readings</h1>\n"

"Light: %u<br>"

"Humidity: %u<br>"

"Temperature: %u&deg; C",

get\_light(),get\_humidity(), get\_temp());

SEND\_STRING(&s->sout, buf);

} else if(s->filename[1] == '0') {

/\* Turn off leds \*/

leds\_off(LEDS\_ALL);

SEND\_STRING(&s->sout, "Turned off leds!");

} else if(s->filename[1] == '1') {

/\* Turn on leds \*/

leds\_on(LEDS\_ALL);

SEND\_STRING(&s->sout, "Turned on leds!");

} else {

if(s->filename[1] != 't') {

generate\_chart("Light", "Light", 0, 500, light1);

SEND\_STRING(&s->sout, buf);

}

if(s->filename[1] != 'l') {

generate\_chart("Temperature", "Celsius", 15, 50, temperature);

SEND\_STRING(&s->sout, buf);

}

}

SEND\_STRING(&s->sout, BOTTOM);

PSOCK\_END(&s->sout);

}

/\*---------------------------------------------------------------------------\*/

httpd\_simple\_script\_t

httpd\_simple\_get\_script(const char \*name)

{

return send\_values;

}

/\*---------------------------------------------------------------------------\*/

PROCESS\_THREAD(web\_sense\_process, ev, data)

{

static struct etimer timer;

PROCESS\_BEGIN();

sensors\_pos = 0;

etimer\_set(&timer, CLOCK\_SECOND \* 2);

SENSORS\_ACTIVATE(light\_sensor);

SENSORS\_ACTIVATE(sht11\_sensor);

while(1) {

PROCESS\_WAIT\_EVENT\_UNTIL(etimer\_expired(&timer));

etimer\_reset(&timer);

light1[sensors\_pos] = get\_light();;

temperature[sensors\_pos] = get\_temp();

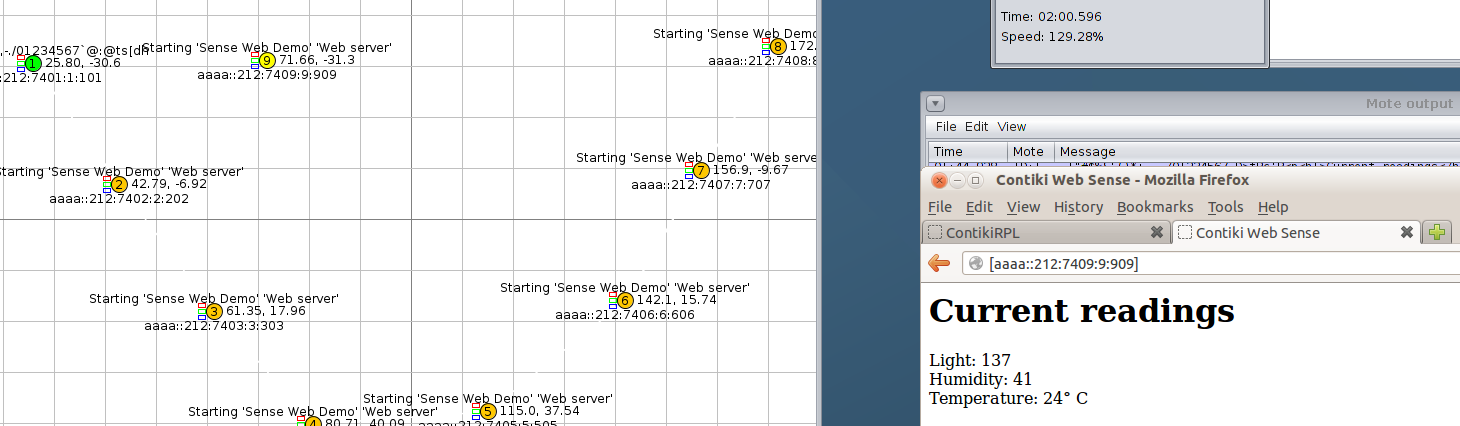
sensors\_pos = (sensors\_pos + 1) % HISTORY;

}

PROCESS\_END();

}

Στην συνέχεια προσθέτουμε έναν κόμβο(κόμβος 9).



Πράγματι αν επικοινωνήσουμε με τον 9ο κόμβο μέσω του url θα μπορέσουμε να δούμε τις μετρήσεις της υγρασίας.