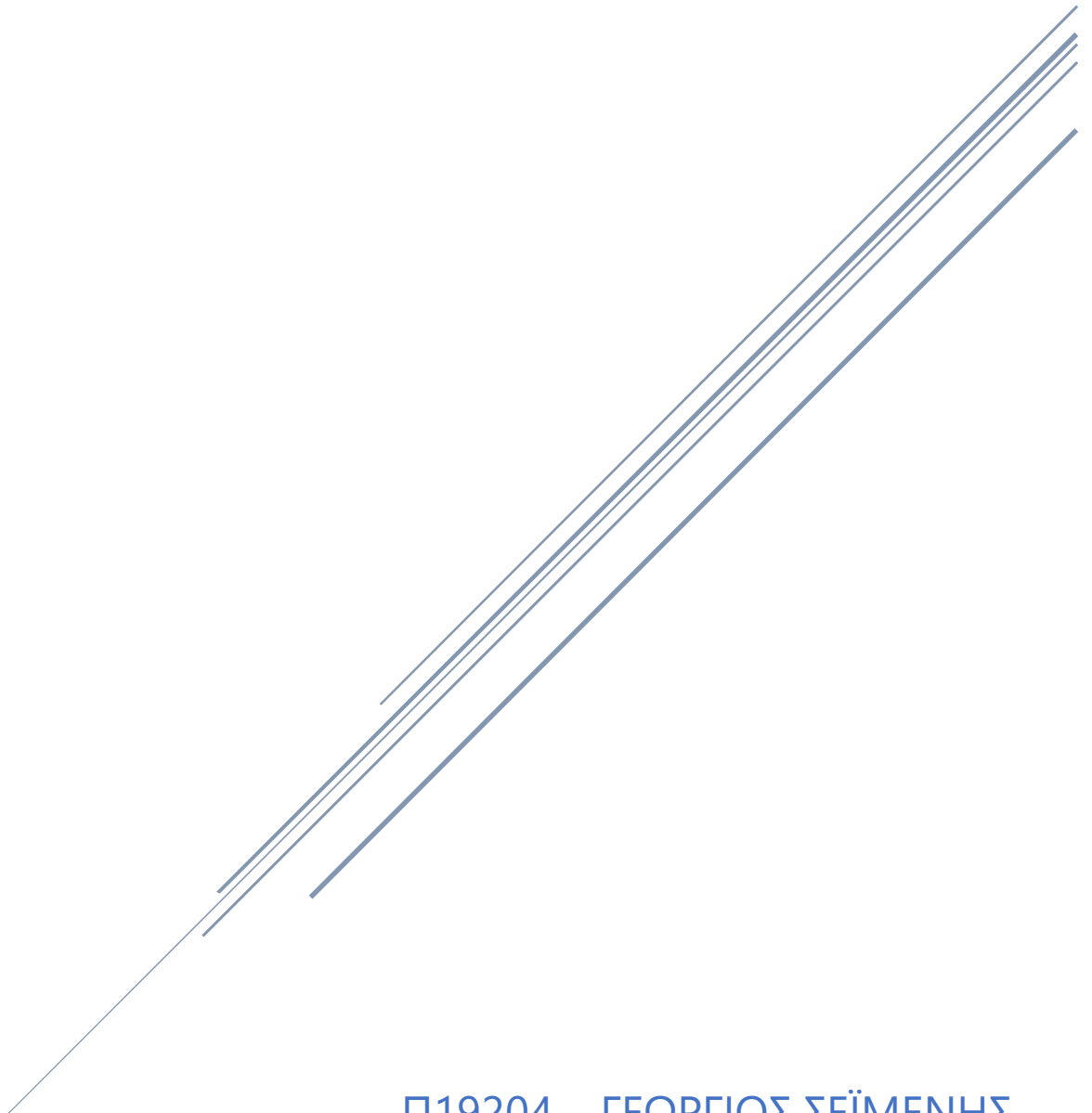


# ΕΡΓΑΣΤΗΡΙΟ Α΄

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΔΙΚΤΥΩΝ ΚΑΙ  
ΚΙΝΗΤΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ



Π19204 – ΓΕΩΡΓΙΟΣ ΣΕΪΜΕΝΗΣ

# Άσκηση 1

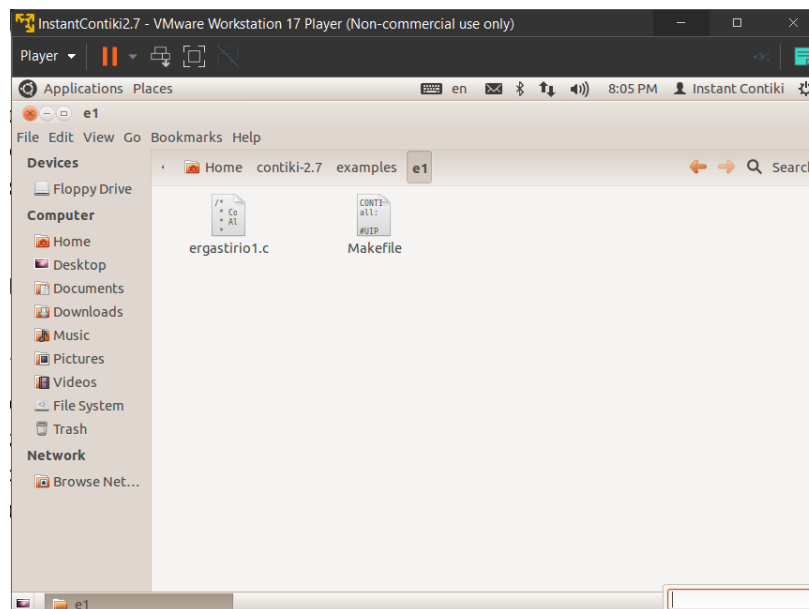
---

## Εισαγωγή

Παρακάτω θα εξηγηθούν οι ενέργειες που ακολουθήθηκαν για να λυθεί η πρώτη εργαστηριακή άσκηση (Άσκηση 1) του μαθήματος. **Στόχος της άσκησης ήταν να τροποποιηθούν οι κατάλληλες γραμμές κώδικα, ώστε να εκτυπώνεται το όνομα του χρήστη, αντί για το «Hello World».**

## Βήμα 1<sup>ο</sup>: Δημιουργία των Κατάλληλων Αρχείων

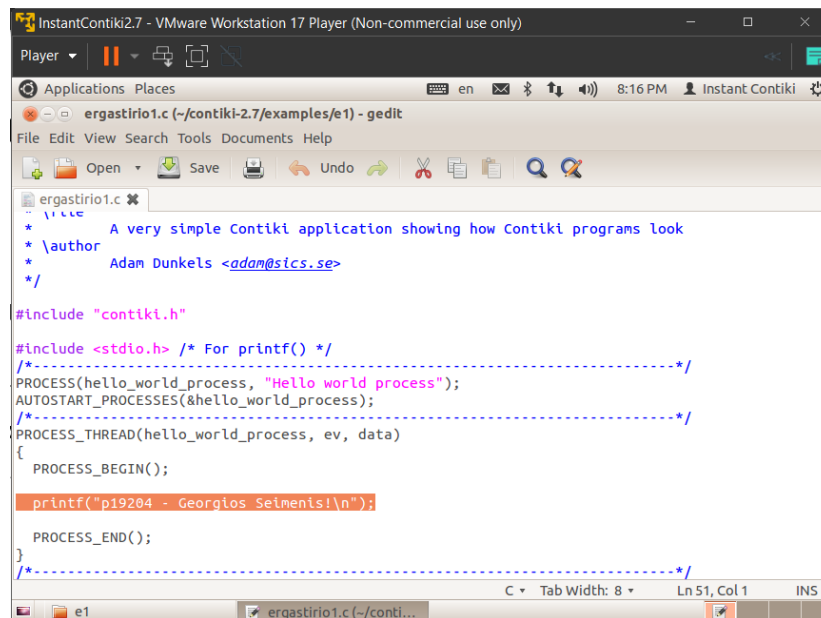
Το πρώτο βήμα που πρέπει να ακολουθήσουμε, προτού ανοίξουμε το Cooja, είναι να τροποποιήσουμε το αρχείο C κώδικα που εκτελείται. **Γι' αυτό, θα πάμε στην διεύθυνση «/home/user/contiki-2.7/examples» και φτιάχνουμε έναν νέο φάκελο, που θα τον πούμε E1. Μέσα σε αυτόν τον φάκελο σιγουρευόμαστε ότι υπάρχουν τα κατάλληλα αρχεία:**



Εικόνα 1: Παρουσία των κατάλληλων αρχείων

## Βήμα 2ο: Τροποποίηση Κώδικα

Τροποποιούμε κατάλληλα τον κώδικα για να εμφανίζονται μονάχα το όνομά μας κατά την εκτέλεσή του. Αρκεί να αλλάξουμε την γραμμή στην οποία γράφεται το `printf()`;



```
ergastirio1.c
/*
 * A very simple Contiki application showing how Contiki programs look
 * \author Adam Dunkels <adam@sics.se>
 */

#include "contiki.h"
#include <stdio.h> /* For printf() */
/*-----*/
PROCESS(hello_world_process, "Hello world process");
AUTOSTART_PROCESSES(&hello_world_process);
/*-----*/
PROCESS_THREAD(hello_world_process, ev, data)
{
    PROCESS_BEGIN();

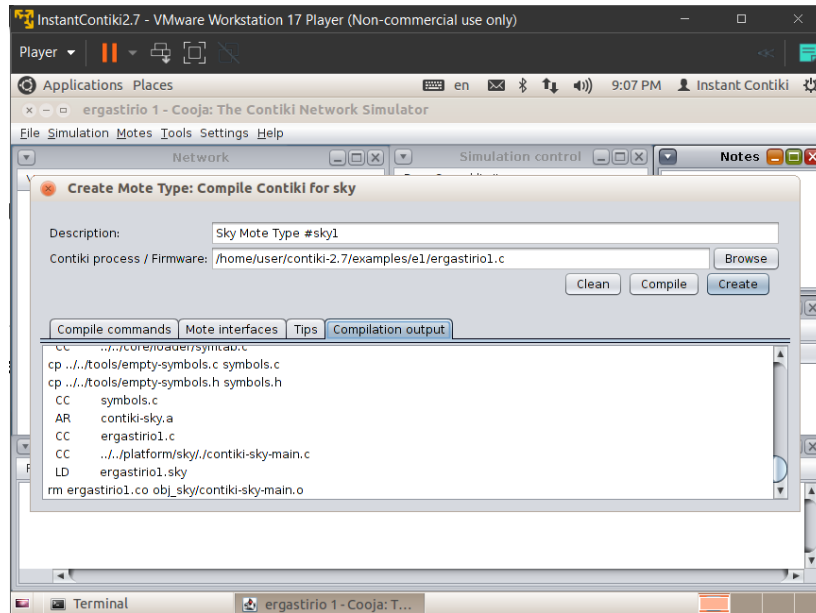
    printf("p19204 - Georgios Selmenis!\\n");

    PROCESS_END();
}
/*-----*/
```

Εικόνα 2: Τροποποίηση της κατάλληλης γραμμής κώδικα.

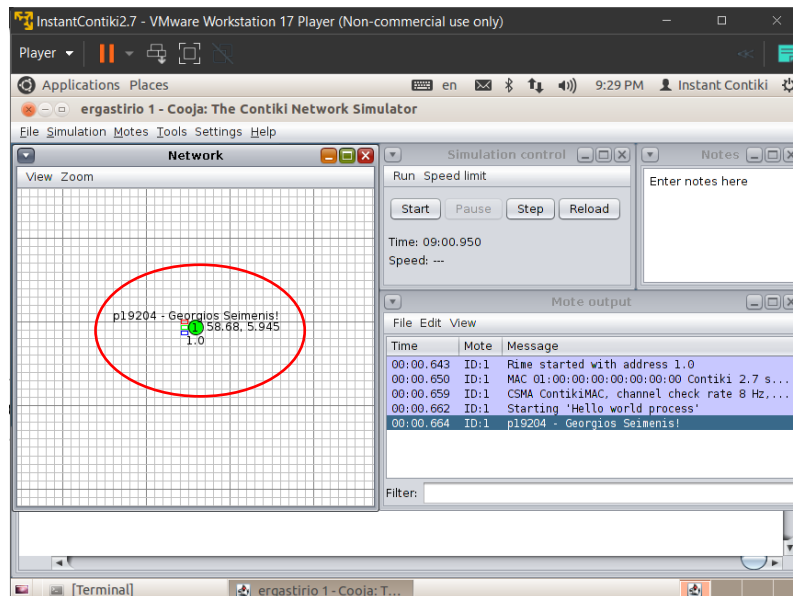
## Βήμα 3ο: Άνοιγμα Cooja και Προσθήκη Κόμβου

Για να δούμε εν δράσει την τροποποίηση της γραμμής αυτής, θα πρέπει να ανοίξουμε το Cooja και να προσθέσουμε έναν κόμβο με το Script αυτό. **Έχοντας φτιάξει, λοιπόν, ένα νέο simulation στο Cooja, πατάμε την προσθήκη ενός Sky Mote. Πάνω σε αυτό το Sky Mote, βάζουμε το Script «ergastirio1.c».** Όταν βρει το αρχείο, μπορούμε να πατήσουμε "compile" για να γίνει η μεταγλώττιση του αρχείου.



**Εικόνα 3: Αποτέλεσμα πατήματος του κουμπιού compile.**

Αφού προσθέσουμε τον κόμβο, μπορούμε να δούμε την εκτέλεση του προγράμματος, στην οποία, πράγματι, εμφανίζεται το μήνυμα που θέλουμε. Για να δούμε την εκτέλεση του προγράμματος, πατάμε το κουμπί «start».



**Εικόνα 4: Επιτυχής εκτέλεση του προγράμματός μας.**

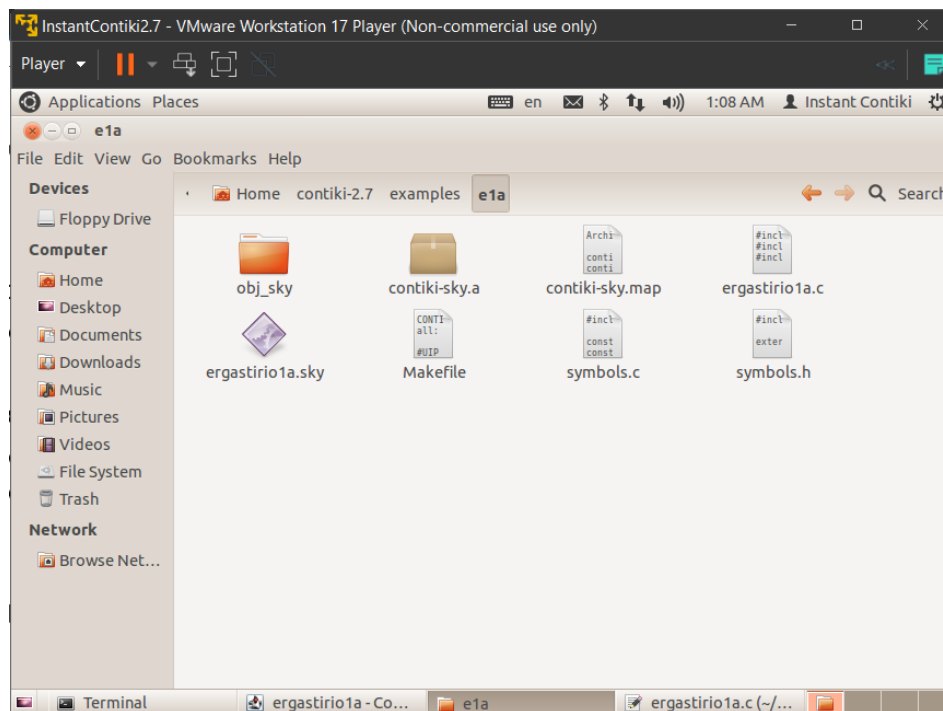
# Άσκηση 1α

## Εισαγωγή

Μετά την πραγματοποίηση της άσκησης 1, δόθηκε μία άλλη άσκηση με ελάχιστα διαφορετικό στόχο. Σε αυτήν την άσκηση θα πρέπει να γράψουμε κώδικα ώστε να ανάψουν LEDs μέσα στους κόμβους μας. **Με την χρήση χρονομετρητών, θα πρέπει να ανάψουν μετά από ένα χρονικό διάστημα τα δύο LED με διαφορετικά χρώματα.** Συγκεκριμένα, για 2 δευτερόλεπτα μόνο το κόκκινο LED, ύστερα για 4 δευτερόλεπτα μόνο το μπλε LED, και τέλος, για ένα δευτερόλεπτο και τα δύο (παράλληλα θα εμφανίζονται και μηνύματα).

## Βήμα 1<sup>ο</sup>: Δημιουργία των Κατάλληλων Αρχείων

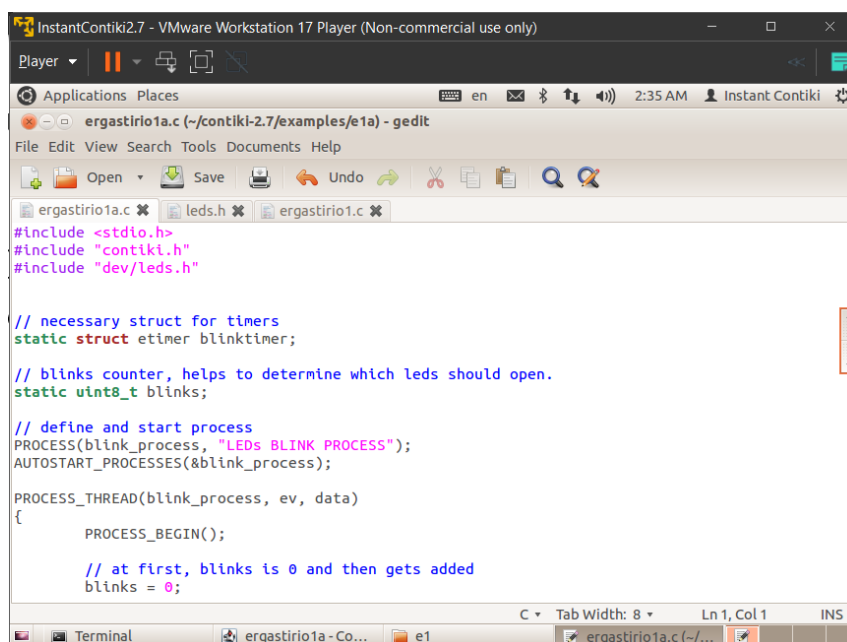
Θα ακολουθήσουμε την ίδια διαδικασία με πριν, κάνοντας τα δύο απαραίτητα αρχεία, όπως και πριν. Αυτήν τη φορά, ας το ονομάσουμε «ergastirio1a.c».



Εικόνα 5: Τα κατάλληλα αρχεία για την εκτέλεση.

## Βήμα 2ο: Τροποποίηση Κώδικα

Πάμε να δούμε την λογική πίσω από τα ζητούμενα της άσκησης. Για αρχή, θέλουμε τις κατάλληλες βιβλιοθήκες του Contiki, ώστε να έχουμε χρήσιμα εργαλεία για την εκτέλεση του προγράμματος. **Από την στιγμή λοιπόν, που έχουμε να κάνουμε με LEDs, χρειαζόμαστε την "leds.h" και για τους χρονομετρητές, το struct "etimer". Μαζί με αυτά, θα προσθέσουμε και μία ακέραια μεταβλητή, την «blinks», η οποία θα μας βοηθήσει να ανοίξουμε LEDs με την σειρά.**



```
#include <stdio.h>
#include "contiki.h"
#include "dev/leds.h"

// necessary struct for timers
static struct etimer blinktimer;

// blinks counter, helps to determine which leds should open.
static uint8_t blinks;

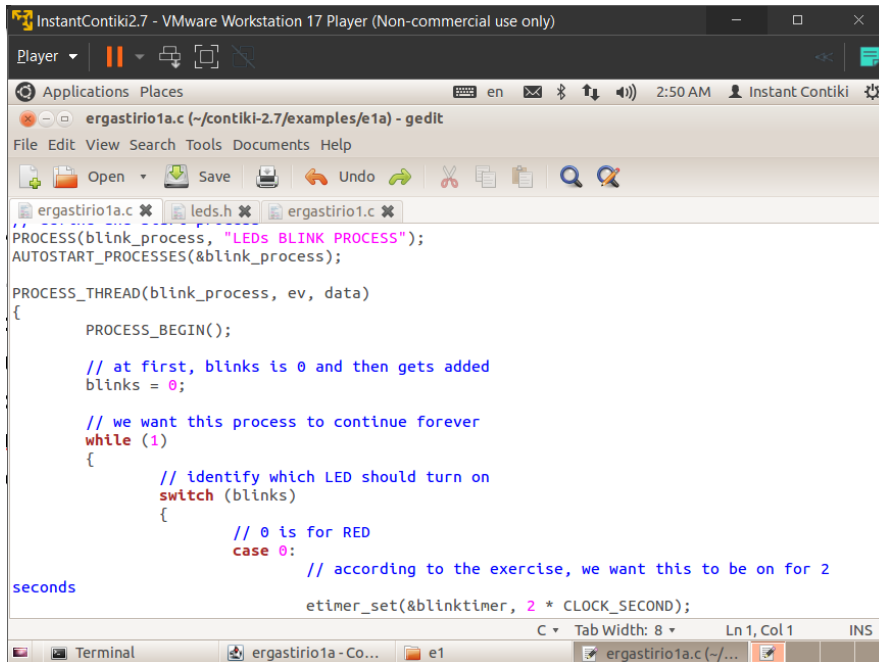
// define and start process
PROCESS(blink_process, "LEDS BLINK PROCESS");
AUTOSTART_PROCESSES(&blink_process);

PROCESS_THREAD(blink_process, ev, data)
{
    PROCESS_BEGIN();

    // at first, blinks is 0 and then gets added
    blinks = 0;
```

Εικόνα 6: Εισαγωγή των κατάλληλων μεταβλητών και βιβλιοθηκών.

Έχοντας, λοιπόν, τις κατάλληλες προϋποθέσεις για το πρόγραμμα, πάμε να γράψουμε τις συνθήκες για το πώς θα ανάβουν με τη σειρά τα LEDs. Γι' αρχή, **αρχικοποιούμε με την τιμή 0 την μεταβλητή blinks.** Θέλουμε, να επαναλαμβάνεται η διαδικασία αλλαγής των LED, όπως το εξηγήσαμε στην εισαγωγή. **Άρα, θα ξεκινάμε από το 0, θα προσθέτουμε με τη σειρά μέχρις ότου να φτάσουμε στο 2, και μετά ξαναμηδενίζουμε την μεταβλητή.** Από τη στιγμή που η διαδικασία θα συνεχίζεται επ' άπειρον, θα το βάλουμε μέσα σε μία "while (1)" επανάληψη.



Εικόνα 7: Κώδικας της επανάληψης.

Τώρα, έρχεται το σημείο στο οποίο θα πρέπει να δούμε τι θα κάνουμε σε κάθε περίπτωση. Χρησιμοποιώντας την «switch», μπορούμε να ελέγχουμε σε ποια περίπτωση είμαστε και να εκτελούμε τον κατάλληλο κώδικα. **Στην περίπτωση 0, θα θέλουμε ο κώδικάς μας να ανάβει το κόκκινο φως, στην περίπτωση 1 να ανάβει μόνο το μπλε, και στην τελευταία περίπτωση και τα δύο. Ακολουθεί ο κώδικας που γράφτηκε στο script:**

```
while (1)
{
    // identify which LED should turn on
    switch (blinks)
    {
        // 0 is for RED
        case 0:
            // according to the exercise, we want this to be on for 2 seconds
            etimer_set(&blinktimer, 2 * CLOCK_SECOND);

            // turn all leds off except for the red.
            leds_off(LEDS_ALL);
            leds_on(LEDS_RED);

            printf("Now turning on RED.\n");

```

```

        // continue to the next one after.
        blinks++;
        break;

// 1 is for BLUE
case 1:
    // according to the exercise, we want this to be on for 4
seconds.

    etimer_set(&blinktimer, 4 * CLOCK_SECOND);

    // turn all LEDs except blue.
    leds_off(LEDS_ALL);
    leds_on(LEDS_BLUE);

    printf("Now turning on BLUE.\n");

    // continue to the next one.
    blinks++;
    break;
// 2 is for RED and BLUE
case 2:
    // lastly, this one must be on for just one second
    etimer_set(&blinktimer, 1 * CLOCK_SECOND);

    // turn on blue and red, using the logical OR operator.
    leds_off(LEDS_ALL);
    leds_on(LEDS_BLUE | LEDS_RED);

    printf("Now turning on both.\n");

    // add one to the blinks
    blinks++;
    break;
default:
    break;
}

// we don't want blinks to be more than two, so reset it.
if (blinks > 2)
{
    blinks = 0;
}

```

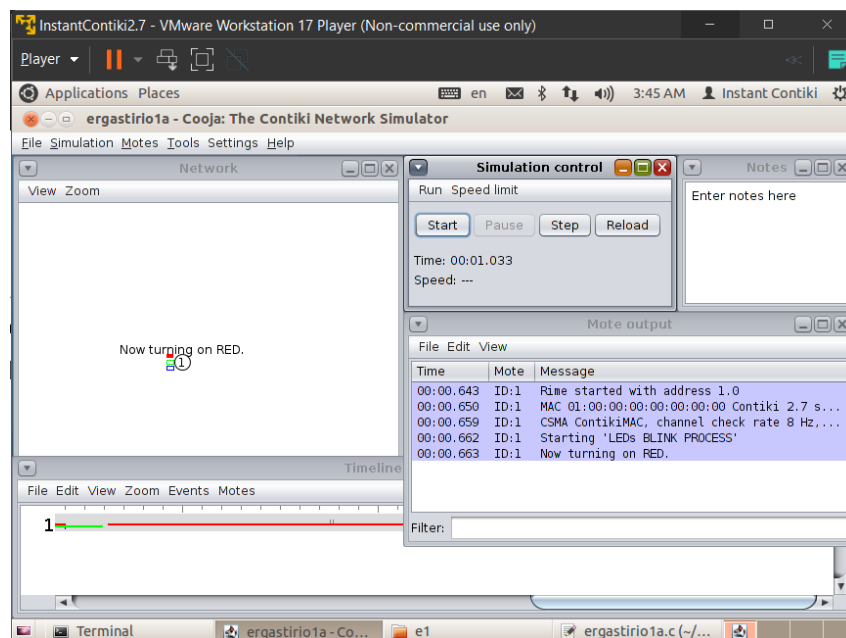


```
// this is for the timer. It waits until the processes are done.  
PROCESS_WAIT_EVENT_UNTIL(ev == PROCESS_EVENT_TIMER);  
}
```

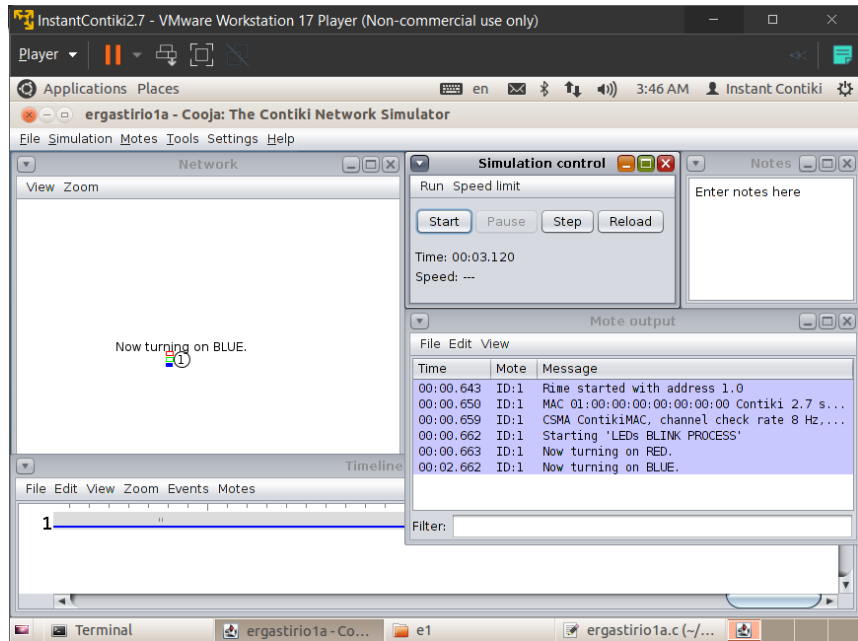
**Παράδειγμα 1: Ο κώδικας στην επανάληψη τύπου while.**

## Βήμα 3<sup>ο</sup>: Άνοιγμα Contiki Cooja και Προσθήκη Κόμβου

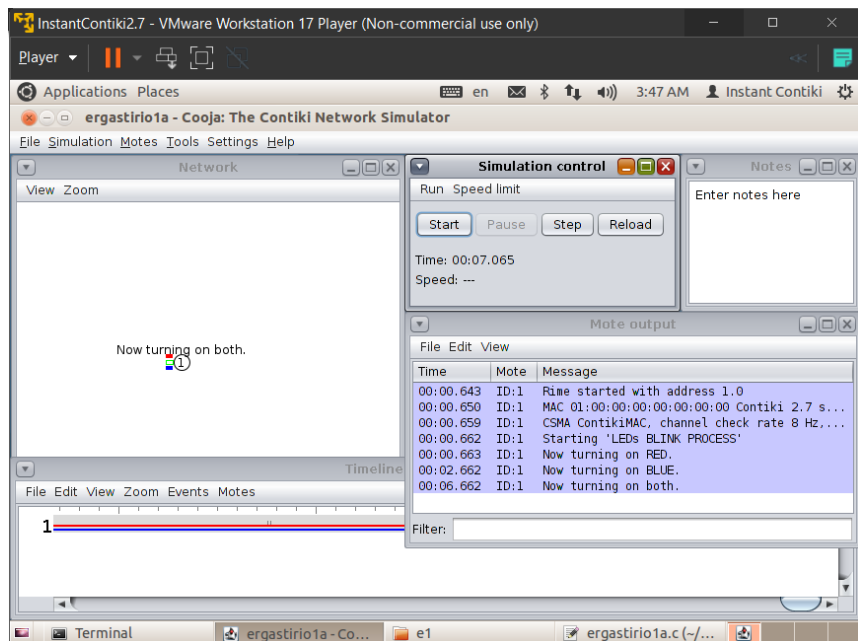
Σε αυτό το σημείο, μένει να δούμε εάν ο κώδικάς μας δουλεύει στο simulator. Ανοίγουμε το simulator, βάζουμε τα Sky Motes (όπως και στην άσκηση 1) και εκτελούμε το Script. Πράγματι, δουλεύουν όπως θα θέλαμε.



**Εικόνα 8: Πρώτη επανάληψη**



Εικόνα 9: Δεύτερη επανάληψη



Εικόνα 10: Τελευταία επανάληψη

Ρίχνοντας μία ματιά στα Mote Output, μπορούμε να δούμε ότι και η χρονομέτρηση γίνεται σωστά, όπως θέλαμε στην αρχή για κάθε διαφορετικό LED. Άρα, το πρόγραμμά μας δουλεύει ακριβώς έτσι όπως θέλουμε.

# Πηγές – Βιβλιογραφία

---

Θερμές ευχαριστίες στα παρακάτω, που βοήθησαν στην συγγραφή και την δημιουργία αυτής της άσκησης. Ο κώδικας των motes μπορεί να βρεθεί και στο προσωπικό [GitHub](#) του φοιτητή.

1. "[LED Programming in Contiki OS – IoT Tutorial 9](#)"
2. [Σημειώσεις Διδάσκοντα](#)

**ΤΕΛΟΣ ΠΑΡΟΥΣΙΑΣΗΣ ~ Π19204 Γεώργιος Σεϊμένης**