Αλγοριθμικές Θεμελιώσεις Δικτύων Αισθητήρων
Άσκηση 2018

Ομάδα: Κυριακού Ανδρόνικος – ΑΜ: 5806, Ντενέζος Παναγιώτης – ΑΜ: 5853
Έτος: 5ο
Email: kyriakou@ceid.upatras.gr, ntenezos@ceid.upatras.gr

Εισαγωγή

c) Για το ερώτημα αυτό μετατράπηκε ο κώδικας hello-world.c και λήφθηκε το ακόλουθο screenshot.

```
user@instant-contiki:~/contiki/examples/hello-world$ make TARGET=sky login
../../tools/sky/serialdump-linux -b115200 /dev/ttyUSB0
connecting to /dev/ttyUSB0 (115200) [OK]
Rime started with address 0.18.116.0.19.96.239.95
MAC 00:12:74:00:13:60:ef:5f Contiki-2.6-2450-geaa8760 started. Node id is not set.
nullsec CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7400:1360:ef5f
Starting 'Hello world process'
Kyriakou Andronikos-5806
Ntenezos Panagiotis-5853
```

Σχήμα 1: Αποτελέσματα Ερωτήματος c

Ο κώδικας παρατίθεται στο αρχείο *Intro-QuestionC.c*

Μέρος Πρώτο

a) Για το υποερώτημα αυτό προστέθηκαν οι εντολές:

```
leds_on(LEDS_GREEN);
leds_on(LEDS_RED);
```

Όπου και ενεργοποιήθηκε το κόκκινο και το πράσινο led.
Ο κώδικας παρατίθεται στο αρχείο *PartA-QuestionA.c*

b) Στο υποερώτημα αυτό χρησιμοποιήθηκαν μετρητές, Πιο συγκεκριμένα αναπτύχθηκε ο παρακάτω κώδικας για να υλοποιηθεί η λειτουργικότητα που ζητείται.

```
while(1)
{
  leds_on(LEDS_RED);
  printf("Red led ON! \n");
  etimer_set(&et, CLOCK_SECOND * 2);

  PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
  etimer_stop(&et);
  leds_off(LEDS_RED);
  printf("Red led OFF! \n");
  leds_on(LEDS_BLUE);
```

```
  printf("Blue led ON! \n");
  etimer_set(&et, CLOCK_SECOND * 4);

  PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
  etimer_stop(&et);
  leds_on(LEDS_RED);
  printf("Blue and Red leds ON \n");
  etimer_set(&et, CLOCK_SECOND * 1);

  PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
  etimer_stop(&et);
  leds_off(LEDS_RED);
  leds_off(LEDS_BLUE);
  printf("All Leds OFF! \n");
}
```

Ο κώδικας βρίσκεται στο αρχείο *PartA-QuestionB.c* και στο Σχήμα 2 παρατίθεται ένα screenshot από την λειτουργία.

```
Rime started with address 0.18.116.0.19.96.239.95
MAC 00:12:74:00:13:60:ef:5f Contiki-2.6-2450-geaa8760 started. Node id is not set.
nullsec CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7400:1360:ef5f
Starting 'Hello world process'
Red led ON!
Red led OFF!
Blue led ON!
Blue and Red leds ON
All Leds OFF!
Red led ON!
Red led OFF!
Blue led ON!
Blue and Red leds ON
All Leds OFF!
Red led ON!
Red led OFF!
Blue led ON!
```

Σχήμα 2: Αποτελέσματα Ερωτήματος b

c) Στο ερώτημα αυτό έγινε αλληλεπίδραση με το κουμπί.

1) Η έξοδος της σειριακής θύρας:

```
user@instant-contiki:~/contiki/examples/sky$ make TARGET=sky login
../../tools/sky/serialdump-linux -b115200 /dev/ttyUSB0
connecting to /dev/ttyUSB0 (115200) [OK]
Light: 100
Light: 99
Light: 101
Light: 99
Light: 101
Light: 100
Light: 101
```

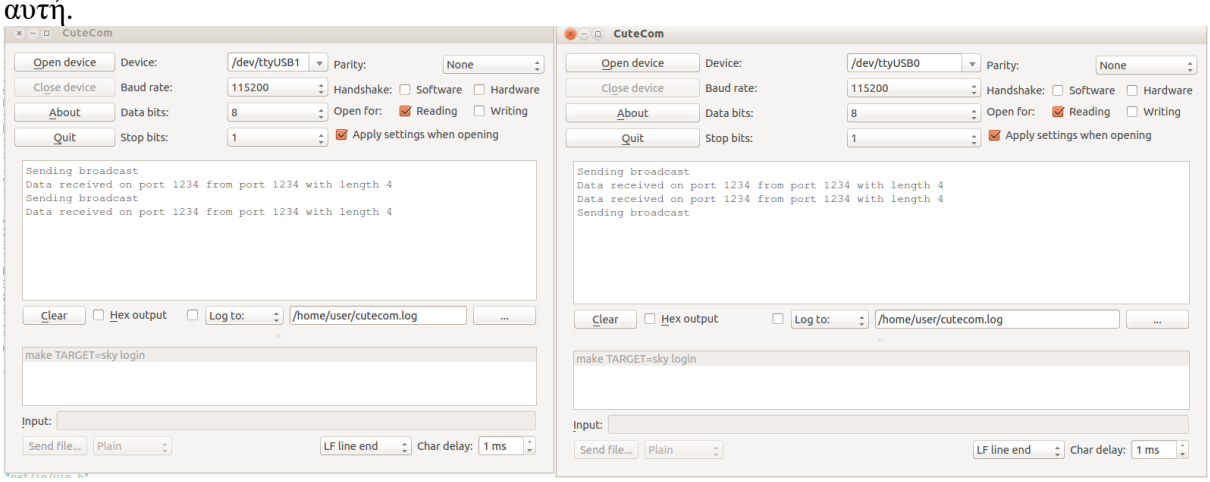
Σχήμα 3: Εκτέλεση Ερωτήματος c

2) Ο κώδικας που αναπτύχθηκε για το ερώτημα αυτό βρίσκεται στο αρχείο *PartA-QuestionC.c* και παρατίθεται παρακάτω:

```
static struct etimer et;
 PROCESS_BEGIN();
 active = 0;
 SENSORS_ACTIVATE(button_sensor);

 while(1){
  etimer_set(&et, CLOCK_SECOND * 10);
  PROCESS_WAIT_EVENT_UNTIL((ev == sensors_event &&
                        data                                    ==
        &button_sensor)||(etimer_expired(&et)));
  leds_toggle(LEDS_ALL);
  etimer_stop(&et);
  SENSORS_ACTIVATE(light_sensor);
  printf("Light: %d\n", light_sensor.value(0));
  SENSORS_DEACTIVATE(light_sensor);
 }
 PROCESS_END();
```

Μέρος Δεύτερο

a) Στο ερώτημα αυτό παρατηρούμε ότι οι αισθητήρες ανταλλάσσουν UDP πακέτα μεταξύ τους ανά τακτά χρονικά διαστήματα. Όπως είναι αναμενόμενο, λόγω πρωτοκόλλου κάποια πακέτα χάνονται. Παρακάτω παρουσιάζεται ένα screenshot με την λειτουργία αυτή.



Σχήμα 4: Ανταλλαγή UDP πακέτων

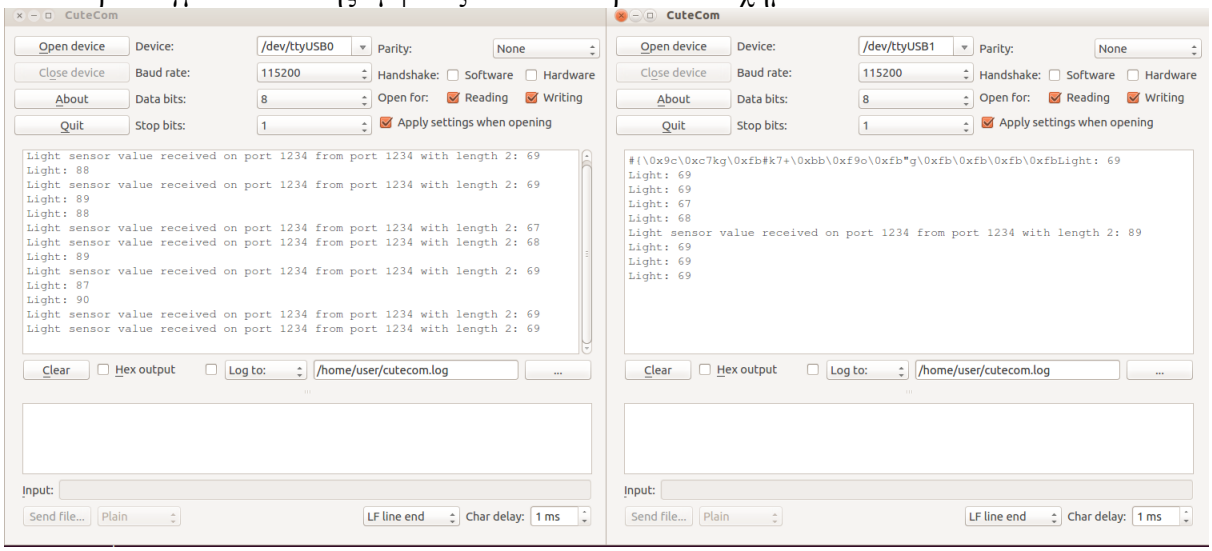
b) Η τροποποίηση που χρειάζεται για να ανταλλάσσεται η τιμή του αισθητήρα φωτεινότητας περιγράφεται στις επόμενες γραμμές:

```
SENSORS_ACTIVATE(light_sensor);
int *mydata = light_sensor.value(0);
printf("Light: %d\n", mydata);
simple_udp_sendto(&broadcast_connection, &mydata,
sizeof(light_sensor.value(0)), &addr);
SENSORS_DEACTIVATE(light_sensor);
```

```
static void
receiver(struct simple_udp_connection *c,
         const uip_ipaddr_t *sender_addr,
         uint16_t sender_port,
         const uip_ipaddr_t *receiver_addr,
         uint16_t receiver_port,
         const uint8_t *data,
         uint16_t datalen)
{
  printf("Light sensor value received on port %d from port %d
with length %d: %d\n",
         receiver_port, sender_port, datalen,*data);
}
```

Ο κώδικας βρίσκεται στο αρχείο *PartB-QuestionB.c*

Ένα παράδειγμα εκτέλεσης εμφανίζεται στο παρακάτω Σχήμα.



Σχήμα 5: Ανταλλαγή τιμών αισθητήρων φωτεινότητας.

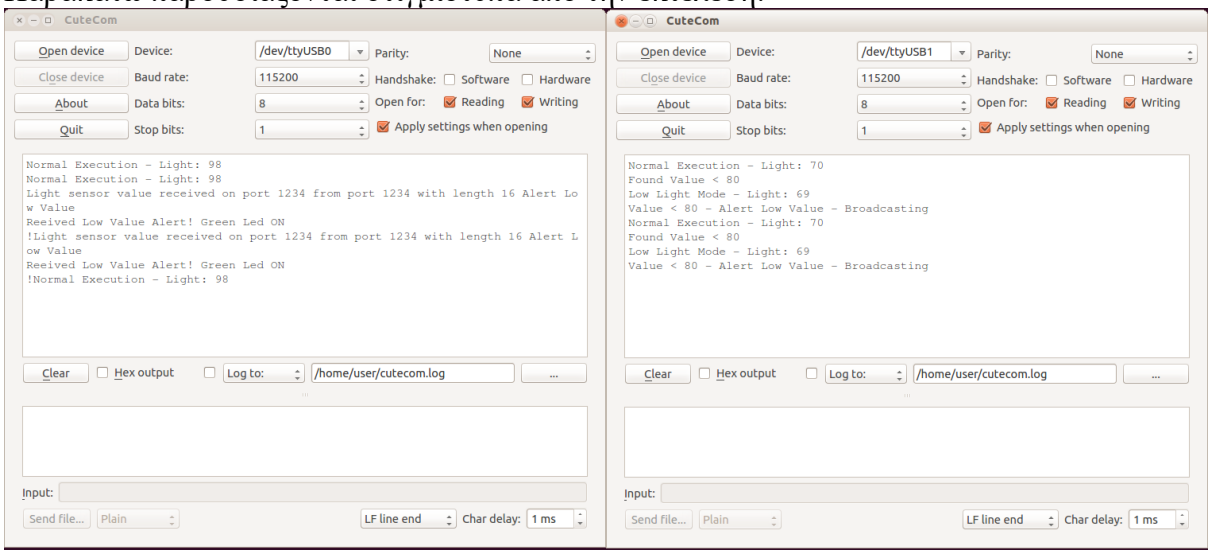
c) Για το ερώτημα αυτό αναπτύχθηκε ο κώδικας που βρίσκεται στο αρχείο *PartB-QuestionC.c* και παρατίθεται παρακάτω. Παρατηρήθηκε ότι ο ένας αισθητήρας παίρνει μετρήσεις στο διάστημα 85-95 και ο άλλος στο διάστημα 67-70 άρα ορίστηκε η τιμή κατωφλίου στο 80.

```
PROCESS_THREAD(broadcast_example_process, ev, data)
{
  static struct etimer et,et2;
```

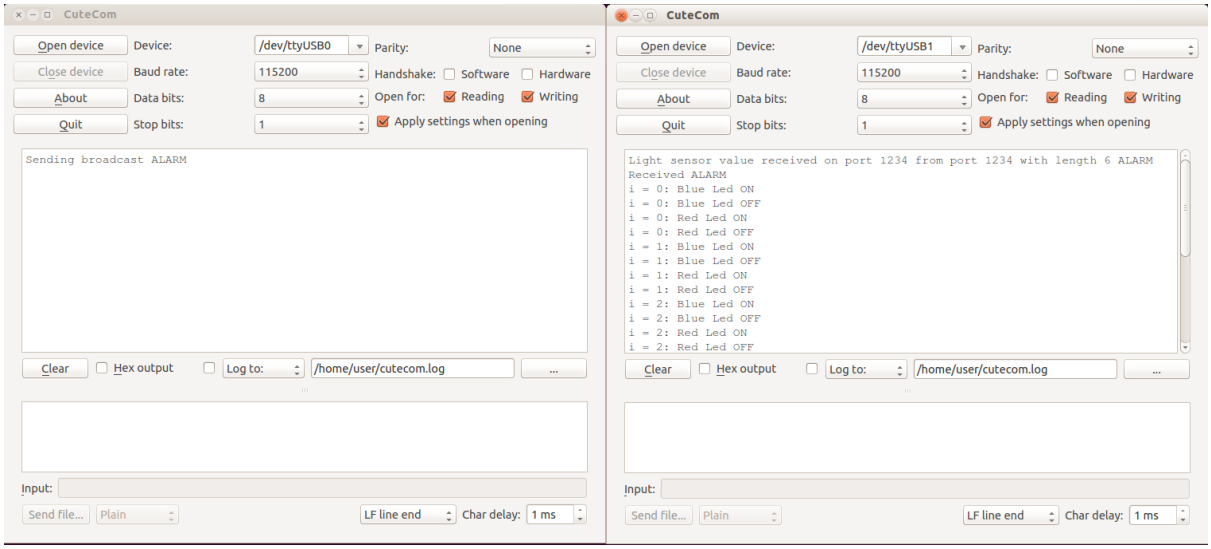
```
  uip_ipaddr_t addr;

  PROCESS_BEGIN();
  SENSORS_ACTIVATE(button_sensor);
  simple_udp_register(&broadcast_connection, UDP_PORT,
                      NULL, UDP_PORT,
                      receiver);

  etimer_set(&et, 10*CLOCK_SECOND);
  int value;
  while(1) {

  PROCESS_WAIT_EVENT_UNTIL((ev == sensors_event &&
                     data == &button_sensor)||(etimer_expired(&et)));

      if(data == &button_sensor)
      {
            printf("Sending broadcast ALARM\n");
            uip_create_linklocal_allnodes_mcast(&addr);
            simple_udp_sendto(&broadcast_connection, "ALARM", sizeof("ALARM"),
&addr);
            etimer_set(&et, 10*CLOCK_SECOND);
            flag = 0;
      }
      else if(flag == 0)
      {
            SENSORS_ACTIVATE(light_sensor);
            printf("Normal Execution - Light: %d\n",value=light_sensor.value(0));
             SENSORS_DEACTIVATE(light_sensor);
            if (value < 80)
            {
                  printf("Found Value < 80 \n");
                  flag = 1;
                  etimer_set(&et,3*CLOCK_SECOND);
            }
           else
            {
                        etimer_set(&et, 10*CLOCK_SECOND);
               }
            }
      else if (flag == 1)
      {
            SENSORS_ACTIVATE(light_sensor);
            printf("Low Light Mode - Light: %d\n",value=light_sensor.value(0));
            SENSORS_DEACTIVATE(light_sensor);
                  if (value < 80)
                  {
                        printf ("Value < 80 - Alert Low Value - Broadcasting \n");
                        uip_create_linklocal_allnodes_mcast(&addr);
```

```
                    simple_udp_sendto(&broadcast_connection, "Alert Low Value",
sizeof("Alert Low Value"), &addr);
                }
                flag = 0;
                etimer_set(&et, 10*CLOCK_SECOND);
        }
    }
    PROCESS_END();
}
```

Παρακάτω παρουσιάζονται στιγμιότυπα από την εκτέλεση:



Σχήμα 6: Κανονική Λειτουργία



Σχήμα 7: Λειτουργία Συναγερμού