Registering new resources in CoAP

TP#3 using FIT/IoT-Lab
Lecturer: Keun-Woo Lim
Lecture slides for COMASIC
10-11-2020

Before doing anything,

- **Choose a site with the least interference**
  - Don’t use Paris site too much!

- **Choose 2 nodes (For 120 minutes)**
  - #1 = border router
  - #2 = CoAP server

- **Do the public CoAP tutorial**
  - https://www.iot-lab.info/tutorials/contiki-coap-m3/
  - Make sure the border router recognizes your coAP server

Before doing anything, (cont.)

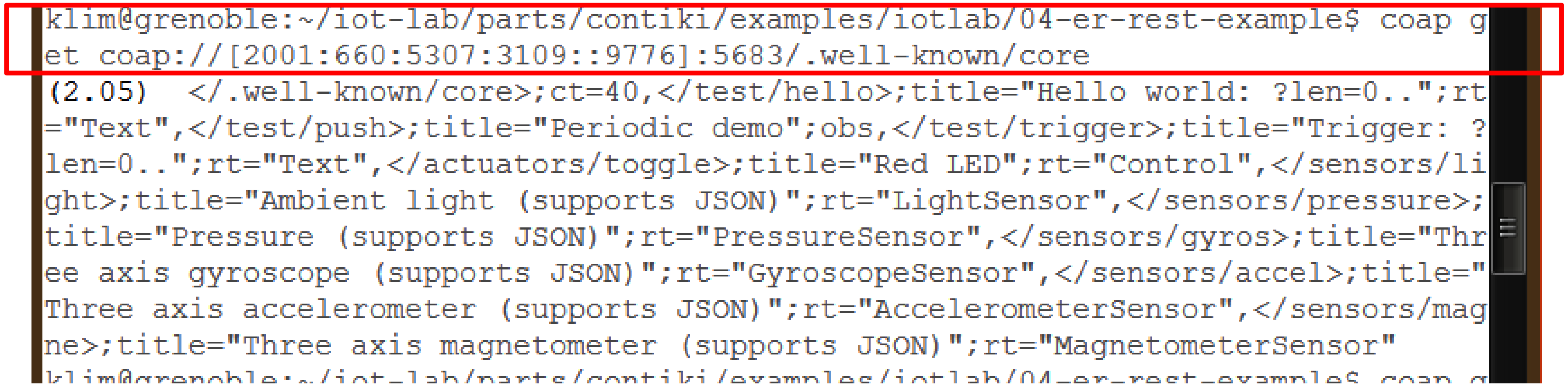
- **Remeber!**
  - Use tunslip and install the original border-router, as explained in the tutorial
  - To confirm RPL link,
    - lynx –dump http://[the node's IP address]

- **All our exercises will be conducted in**
  - ~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example

- **So, every time you change the code:**
  - make TARGET=iotlab-m3
  - iotlab-node -up er-example-server.iotlab-m3 -l XXXXXXXX,m3,YYY

Characteristic of CoAP

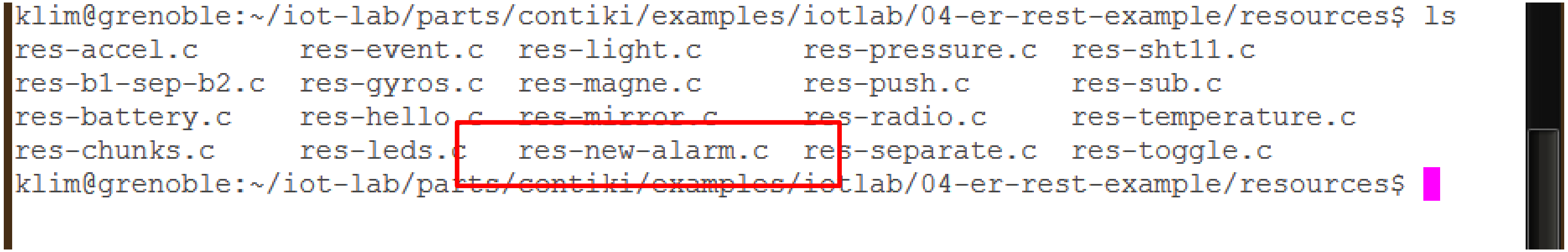
- CoAP server defines specific resources available so that the client can call for it
- Here is an example of resource check, using /.well-known/core



```
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example$ coap g
et coap://[2001:660:5307:3109::9776]:5683/.well-known/core
(2.05)  </.well-known/core>;ct=40,</test/hello>;title="Hello world: ?len=0..";rt
="Text",</test/push>;title="Periodic demo";obs,</test/trigger>;title="Trigger: ?
len=0..";rt="Text",</actuators/toggle>;title="Red LED";rt="Control",</sensors/li
ght>;title="Ambient light (supports JSON)";rt="LightSensor",</sensors/pressure>;
title="Pressure (supports JSON)";rt="PressureSensor",</sensors/gyros>;title="Thr
ee axis gyroscope (supports JSON)";rt="GyroscopeSensor",</sensors/accel>;title="
Three axis accelerometer (supports JSON)";rt="AccelerometerSensor",</sensors/mag
ne>;title="Three axis magnetometer (supports JSON)";rt="MagnetometerSensor"
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example$ coap g
```

New resource?

- **What if you would like to register new functions? New resources?**
  - All you have to do is just register them!



```
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources$ ls
res-accel.c      res-event.c  res-light.c      res-pressure.c  res-sht11.c
res-b1-sep-b2.c  res-gyros.c  res-magne.c      res-push.c      res-sub.c
res-battery.c    res-hello.c  res-mirror.c     res-radio.c     res-temperature.c
res-chunks.c     res-leds.c   res-new-alarm.c  res-separate.c  res-toggle.c
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources$
```

Adding a new resource

- **Start by creating an « alarm » resource**
- **What is it?**
  - A resource that tells you if there is something wrong with the device
- **Why?**
  - The user can make adjustments to the situation, especially if it is urgent

Behavior of new resource

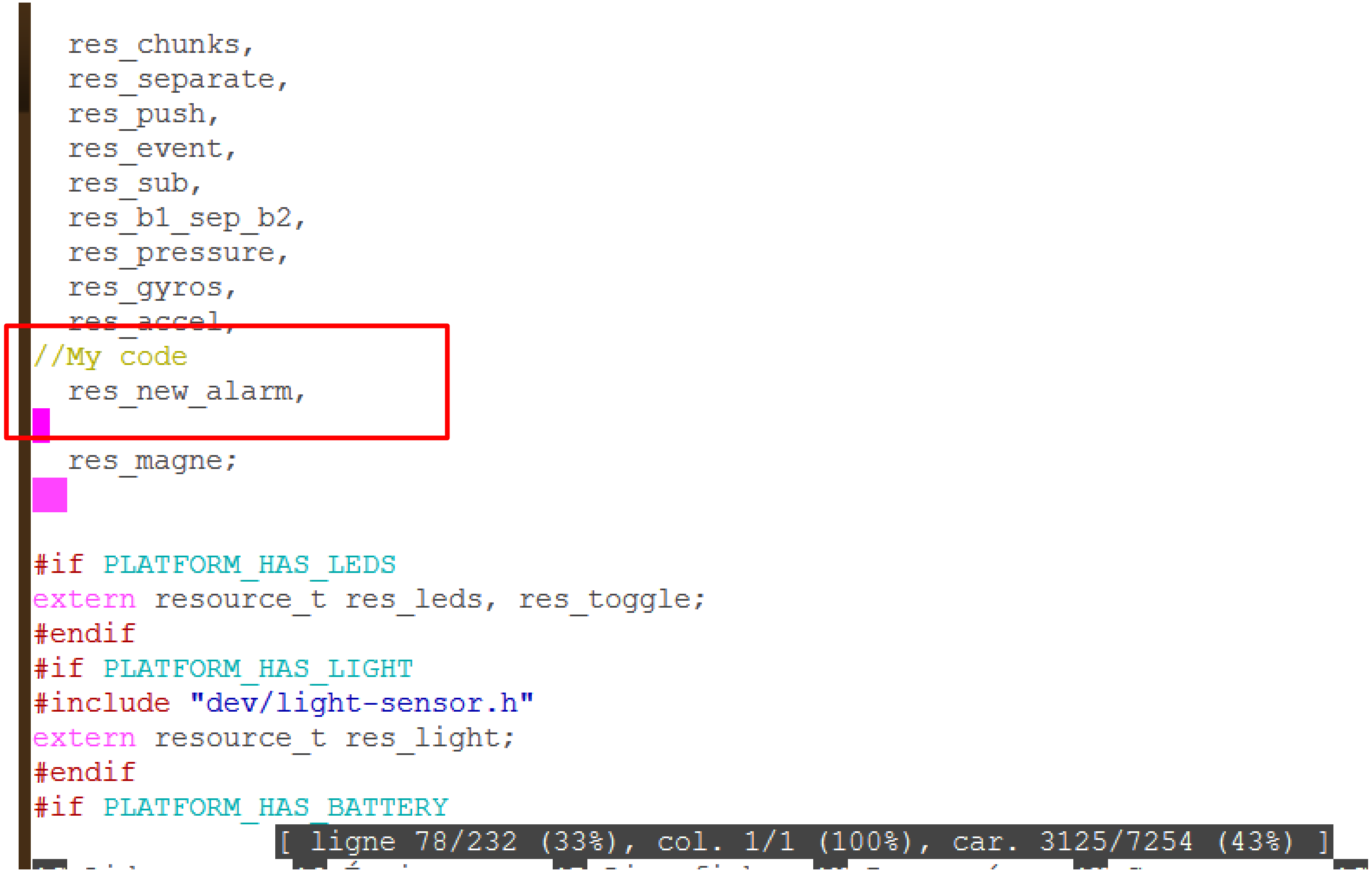
- **Firstly, we create a « On-demand alarm »**
  - Client (front-end) periodically calls the alarm to see if there is a problem.
  - The server will return:
    - 0 if no problem
    - 1 if problem
  - Client will act if there is a problem
  - Problem?
    - A randomized 0/1 generator (For now we use randomness)

1. Change er-example-server.c

- **Goto**
  - ~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example
  - Nano er-example-server.c

- **Declare a new resource name**
  - Line 76
  - Register new resource:
    - res_new_alarm,

Screenshot of (1)

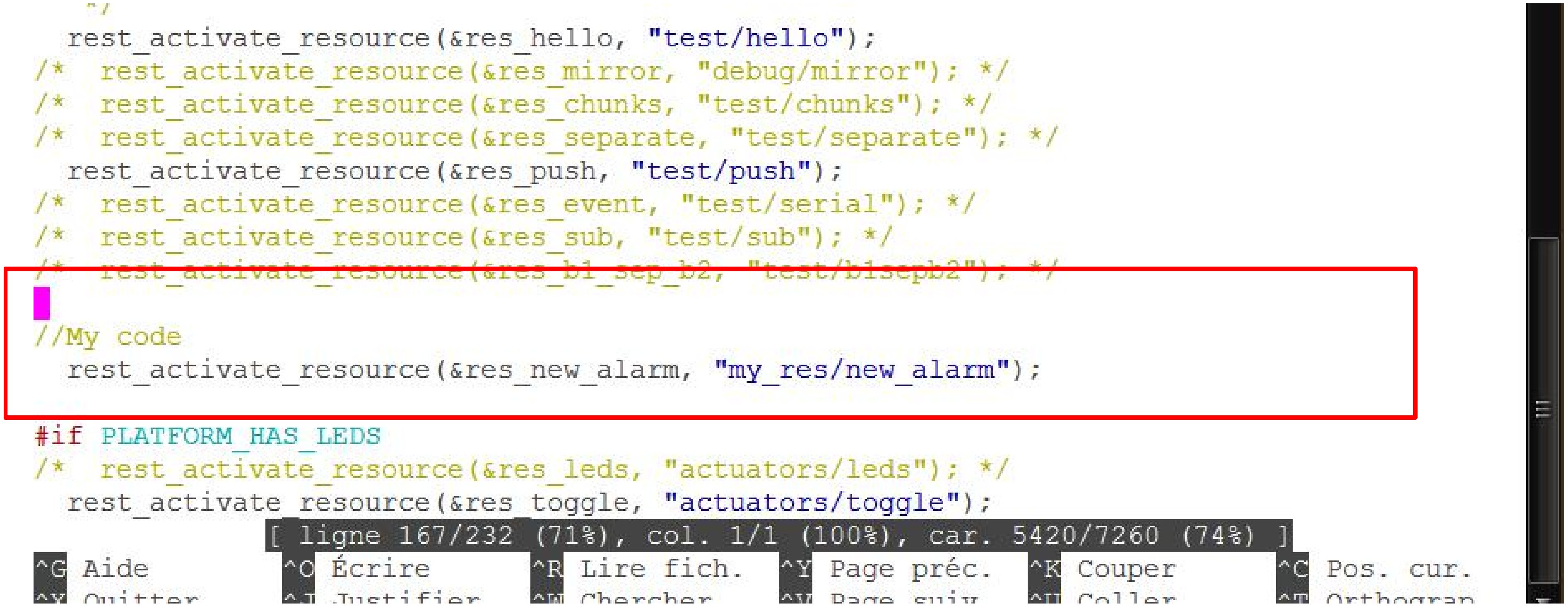


```
  res_chunks,
  res_separate,
  res_push,
  res_event,
  res_sub,
  res_b1_sep_b2,
  res_pressure,
  res_gyros,
  res_accel,
//My code
  res_new_alarm,

  res_magne;


#if PLATFORM_HAS_LEDS
extern resource_t res_leds, res_toggle;
#endif
#if PLATFORM_HAS_LIGHT
#include "dev/light-sensor.h"
extern resource_t res_light;
#endif
#if PLATFORM_HAS_BATTERY
                    [ ligne 78/232 (33%), col. 1/1 (100%), car. 3125/7254 (43%) ]
```

1. Change er-example-server.c (cont)

Activate my new resource

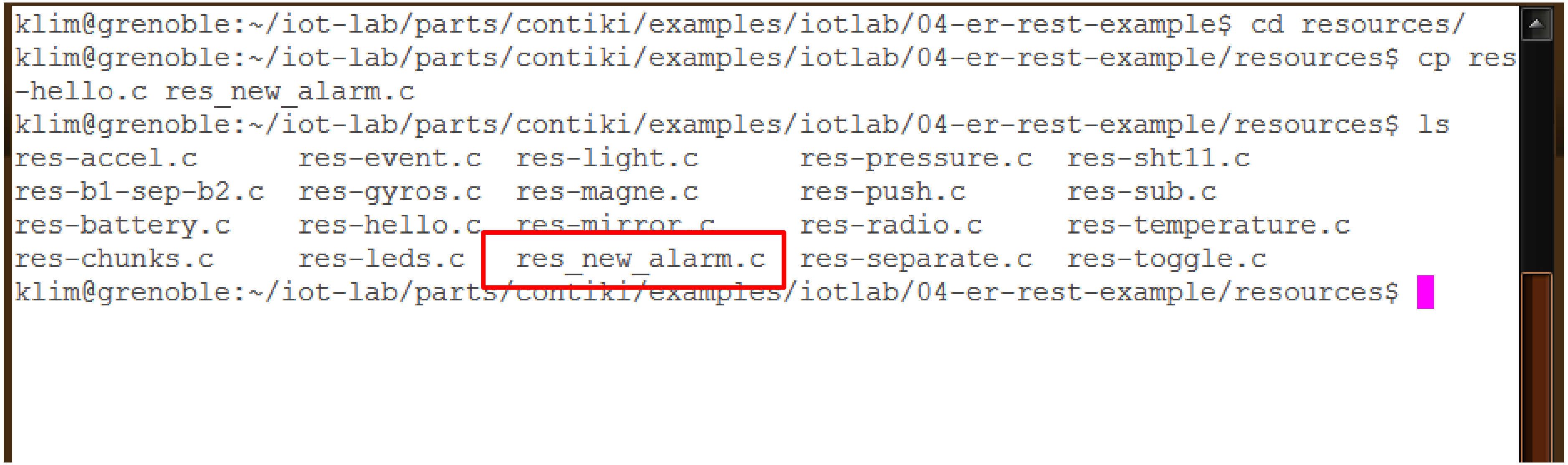
- Line 167-170
- activate new resource:



```
  */
  rest_activate_resource(&res_hello, "test/hello");
/*  rest_activate_resource(&res_mirror, "debug/mirror"); */
/*  rest_activate_resource(&res_chunks, "test/chunks"); */
/*  rest_activate_resource(&res_separate, "test/separate"); */
  rest_activate_resource(&res_push, "test/push");
/*  rest_activate_resource(&res_event, "test/serial"); */
/*  rest_activate_resource(&res_sub, "test/sub"); */
/*  rest_activate_resource(&res_b1_sep_b2, "test/b1sepb2"); */

//My code
  rest_activate_resource(&res_new_alarm, "my_res/new_alarm");

#if PLATFORM_HAS_LEDS
/*  rest_activate_resource(&res_leds, "actuators/leds"); */
  rest_activate_resource(&res_toggle, "actuators/toggle");
                    [ ligne 167/232 (71%), col. 1/1 (100%), car. 5420/7260 (74%) ]
^G Aide          ^O Écrire        ^R Lire fich.    ^Y Page préc.    ^K Couper        ^C Pos. cur.
^X Quitter       ^J Justifier     ^W Chercher      ^V Page suiv.    ^U Coller        ^T Orthograp
```

2. Create new resource

- **Goto**
  - ~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources
  - cp res_hello.c res_new_alarm.c



```
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example$ cd resources/
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources$ cp res
-hello.c res_new_alarm.c
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources$ ls
res-accel.c       res-event.c  res-light.c      res-pressure.c  res-sht11.c
res-b1-sep-b2.c   res-gyros.c  res-magne.c      res-push.c      res-sub.c
res-battery.c     res-hello.c  res-mirror.c     res-radio.c     res-temperature.c
res-chunks.c      res-leds.c   res_new_alarm.c  res-separate.c  res-toggle.c
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources$
```

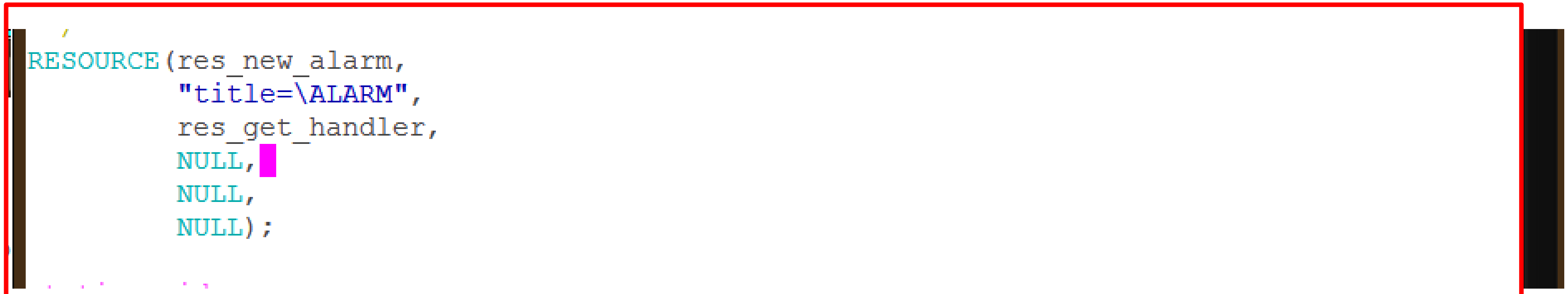
3. Change contents of new resource

- **Goto**
  - nano res_new_alarm.c

- **Change (because it is still res-hello)**
  - Let’s analyze the code a bit!

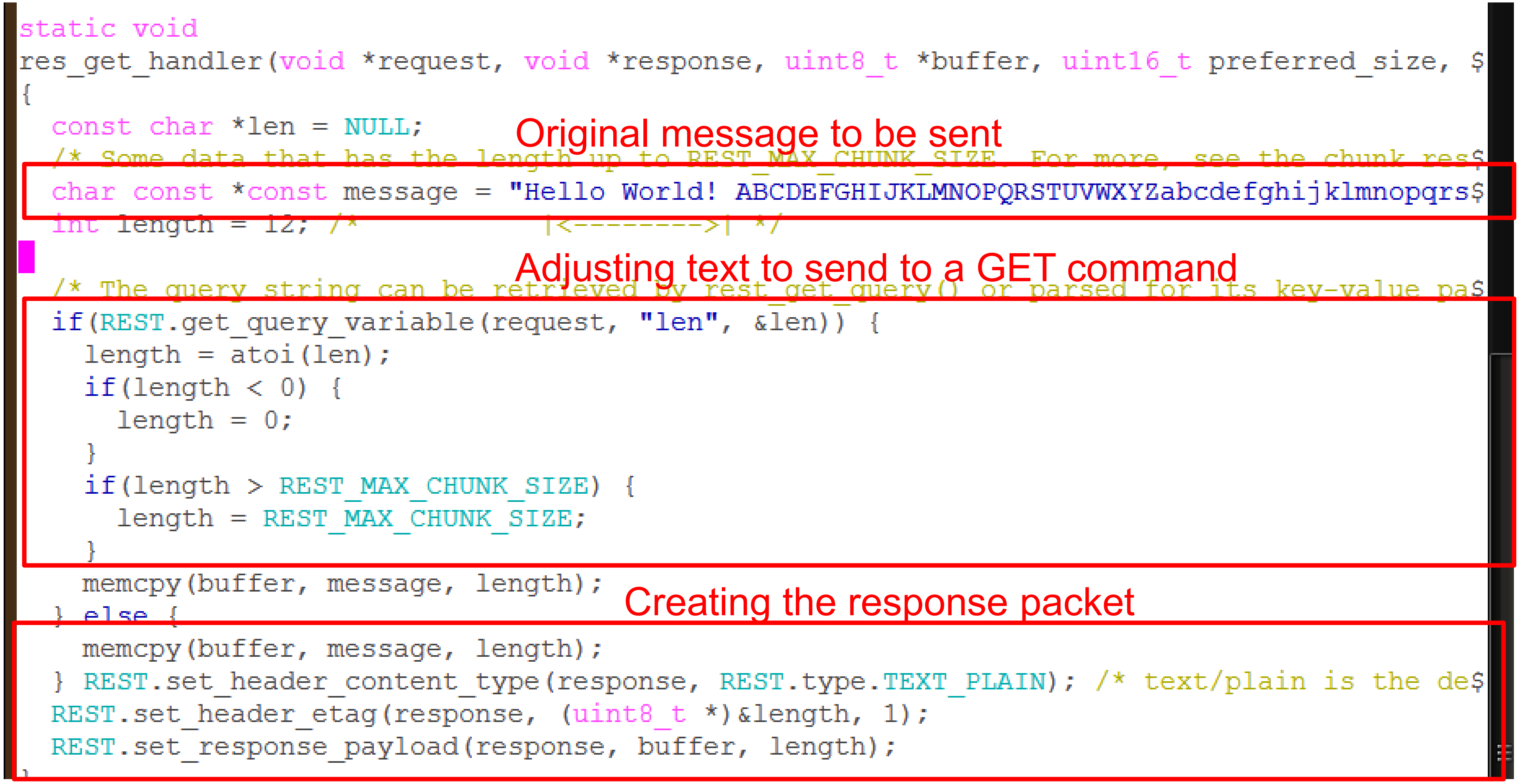
3. Change contents of new resource (cont)

Define the handler first!



```
RESOURCE(res_new_alarm,
         "title=\ALARM",
         res_get_handler,
         NULL,
         NULL,
         NULL);
```

3. Analyze the response (cont)



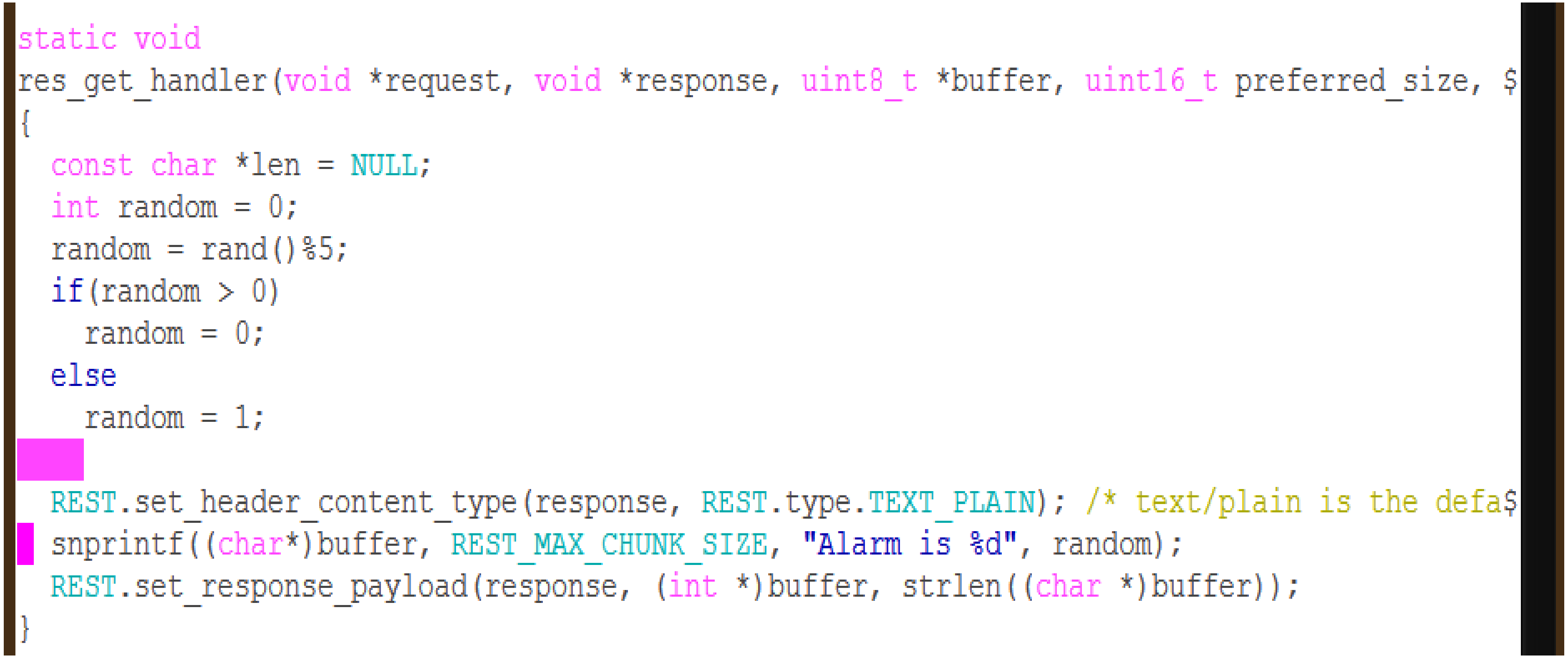
```
static void
res_get_handler(void *request, void *response, uint8_t *buffer, uint16_t preferred_size, $
{
  const char *len = NULL;
  /* Some data that has the length up to REST_MAX_CHUNK_SIZE. For more, see the chunk res$
  char const *const message = "Hello World! ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrs$
  int length = 12; /*           |<-------->| */

  /* The query string can be retrieved by rest_get_query() or parsed for its key-value pa$
  if(REST.get_query_variable(request, "len", &len)) {
    length = atoi(len);
    if(length < 0) {
      length = 0;
    }
    if(length > REST_MAX_CHUNK_SIZE) {
      length = REST_MAX_CHUNK_SIZE;
    }
    memcpy(buffer, message, length);
  } else {
    memcpy(buffer, message, length);
  } REST.set_header_content_type(response, REST.type.TEXT_PLAIN); /* text/plain is the de$
  REST.set_header_etag(response, (uint8_t *)&length, 1);
  REST.set_response_payload(response, buffer, length);
}
```

Original message to be sent

Adjusting text to send to a GET command

Creating the response packet

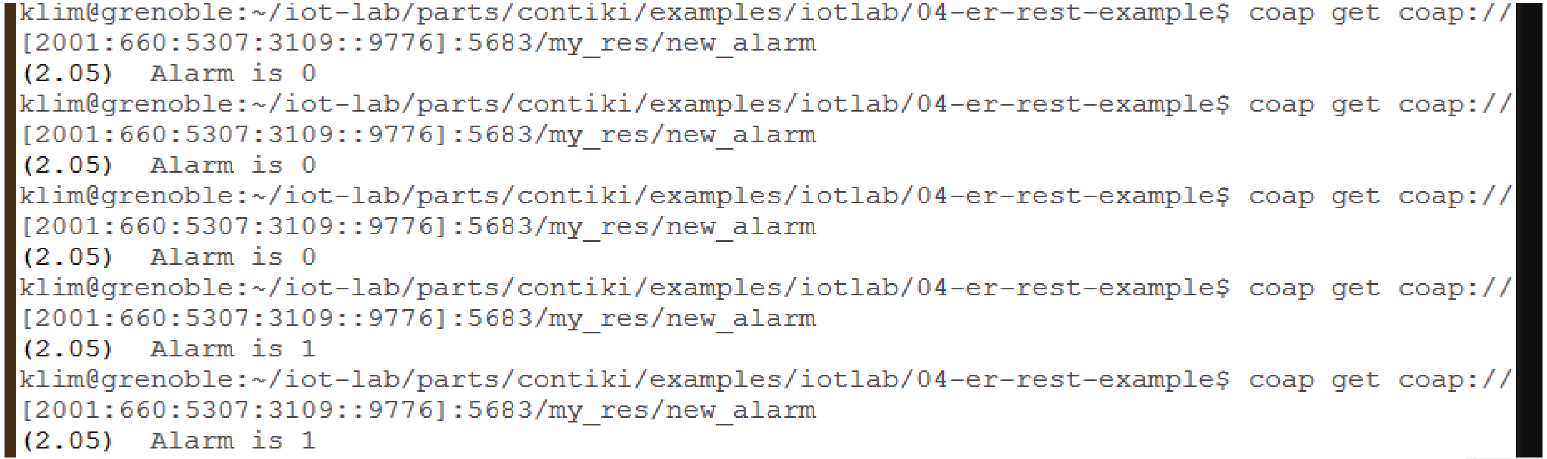
3. My code



```
static void
res_get_handler(void *request, void *response, uint8_t *buffer, uint16_t preferred_size, $
{
  const char *len = NULL;
  int random = 0;
  random = rand()%5;
  if(random > 0)
    random = 0;
  else
    random = 1;

  REST.set_header_content_type(response, REST.type.TEXT_PLAIN); /* text/plain is the defa$
  snprintf((char*)buffer, REST_MAX_CHUNK_SIZE, "Alarm is %d", random);
  REST.set_response_payload(response, (int *)buffer, strlen((char *)buffer));
}
```

4. Compile and Flash

- **Do**
  - cd ..
  - make TARGET=iotlab-m3
  - iotlab-node -up er-example-server.iotlab-m3 -l XXXXXXXX,m3,YYY
  - coap get coap://[IPv6 address of your site::XXXX]: 5683/my_res/new_alarm



```
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example$ coap get coap://
[2001:660:5307:3109::9776]:5683/my_res/new_alarm
(2.05)  Alarm is 0
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example$ coap get coap://
[2001:660:5307:3109::9776]:5683/my_res/new_alarm
(2.05)  Alarm is 0
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example$ coap get coap://
[2001:660:5307:3109::9776]:5683/my_res/new_alarm
(2.05)  Alarm is 0
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example$ coap get coap://
[2001:660:5307:3109::9776]:5683/my_res/new_alarm
(2.05)  Alarm is 1
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example$ coap get coap://
[2001:660:5307:3109::9776]:5683/my_res/new_alarm
(2.05)  Alarm is 1
```

Share information between resources

Why two resources should share data?

- **Well, normally alarms are not random**
  - They should react to the information given by another sensor
  - They must be able to read the information from other sensors

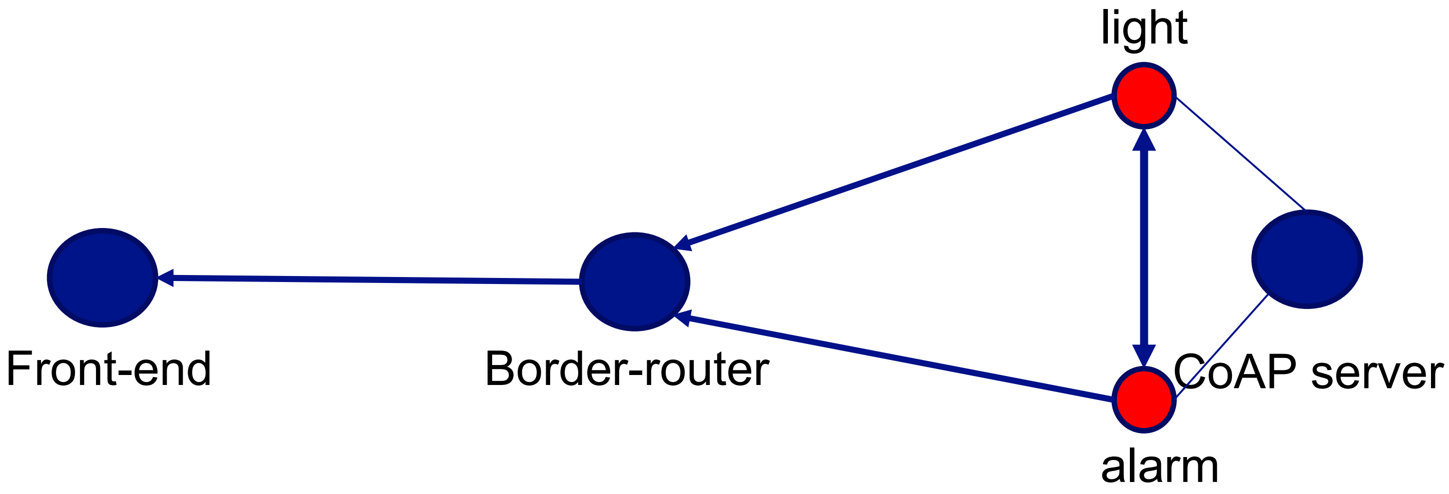
- **If resources can also share data between them, we can make the alarm look much more appealing**

The problem?

- **The structure of the current codes are not very friendly**
  - Separate sources
  - Separate variables
  - We need to combine

- **Therefore, this will be more of a coding problem**
  - Using extern variables
  - Globalizing values

Overview of the architecture



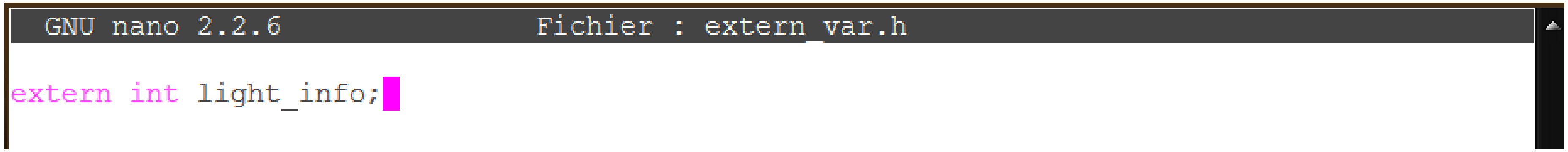
Just a moment

- **Do you really need this architecture?**
  - I mean, the front-end can just read the light information and determine the alarm
  - Then, the alarm resource is not really needed

- **This is true for this architecture, but if we consider two sensor nodes sharing with each other**
  - On-demand alarming isn't really efficient, too much data
  - A sensor must be able to POST an alarm to another server – in this case, an alarm resource is indeed needed
  - So we do this first for practice

1. Declare header

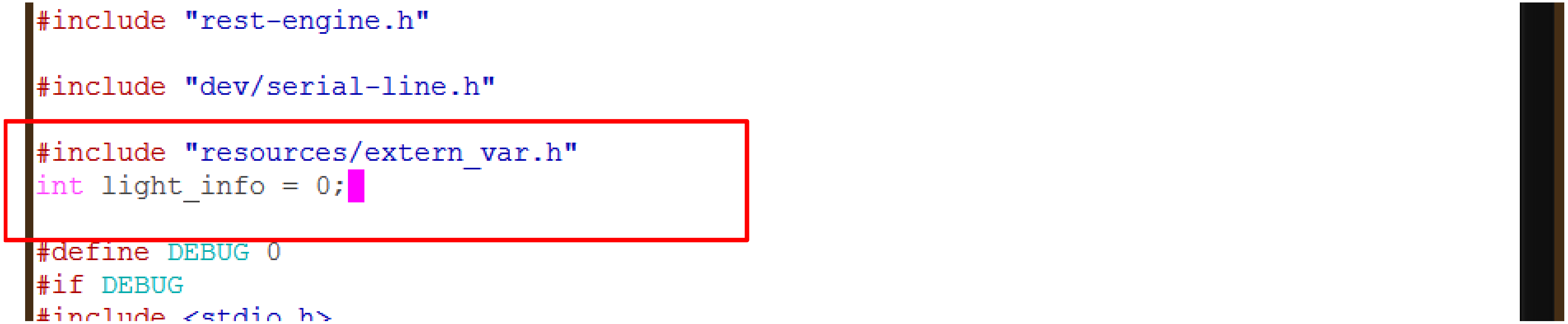
- **Goto**
  - ~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources
  - Nano extern_var.h
- **Declare new extern variable**



```
  GNU nano 2.2.6                    Fichier : extern_var.h

extern int light_info;
```

2. Declare global variable in server

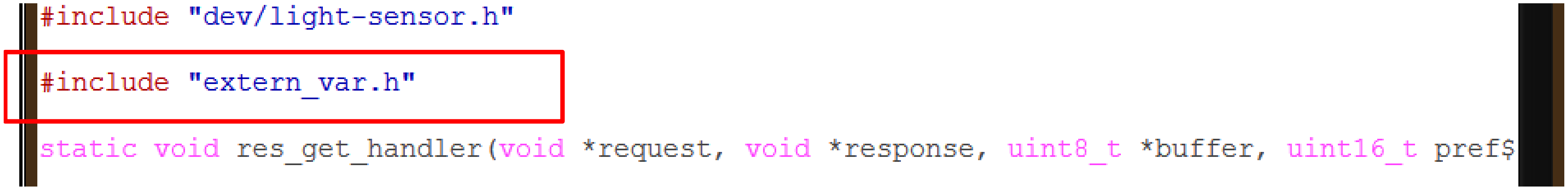
- **Goto**
  - ~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example
  - Nano er-example-server.c



```
#include "rest-engine.h"

#include "dev/serial-line.h"

#include "resources/extern_var.h"
int light_info = 0;

#define DEBUG 0
#if DEBUG
#include <stdio.h>
```

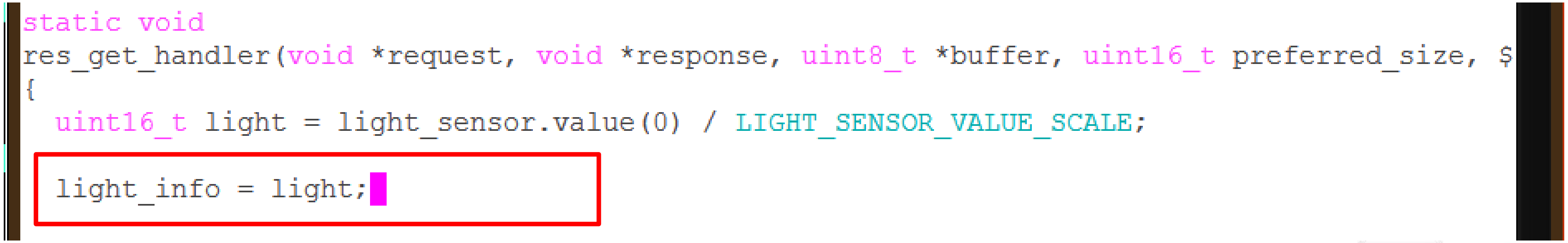
3. Declare global variable in res-light.c

- **Goto**
  - ~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources
  - Nano res-light.c



```
#include "dev/light-sensor.h"

#include "extern_var.h"

static void res_get_handler(void *request, void *response, uint8_t *buffer, uint16_t pref$
```

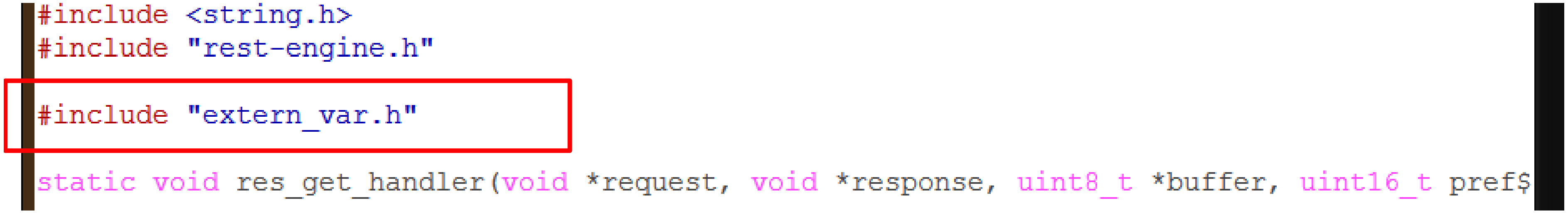
- Record light value in global variable



```
static void
res_get_handler(void *request, void *response, uint8_t *buffer, uint16_t preferred_size, $
{
  uint16_t light = light_sensor.value(0) / LIGHT_SENSOR_VALUE_SCALE;

  light_info = light;
```

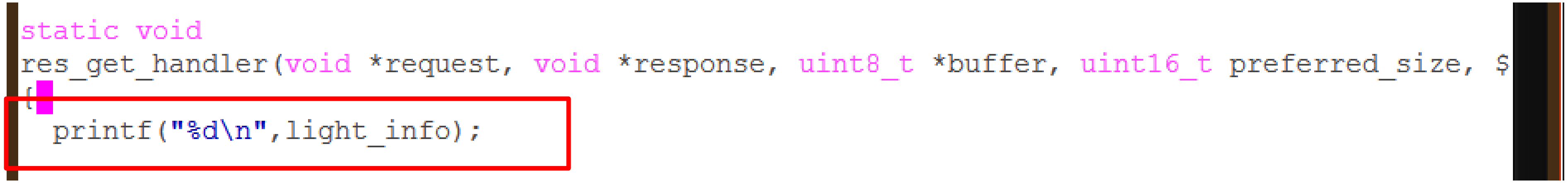
4. read global variable in res_new_alarm.c

- **Goto**
  - Nano res_new_alarm.c



```
#include <string.h>
#include "rest-engine.h"

#include "extern_var.h"

static void res_get_handler(void *request, void *response, uint8_t *buffer, uint16_t pref$
```

  - Read light value from global variable



```
static void
res_get_handler(void *request, void *response, uint8_t *buffer, uint16_t preferred_size, $
{
  printf("%d\n",light_info);
```

5. Compile and Flash

Do

- cd ..
- iotlab-node -up er-example-server.iotlab-m3 -l XXXXXXXX,m3,YYY

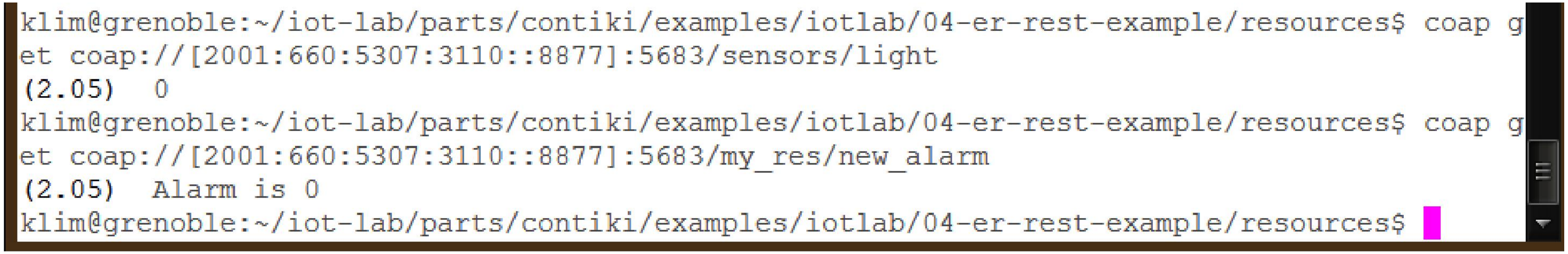
Open another console

- Nc m3-XXX(Your coap server) 20000

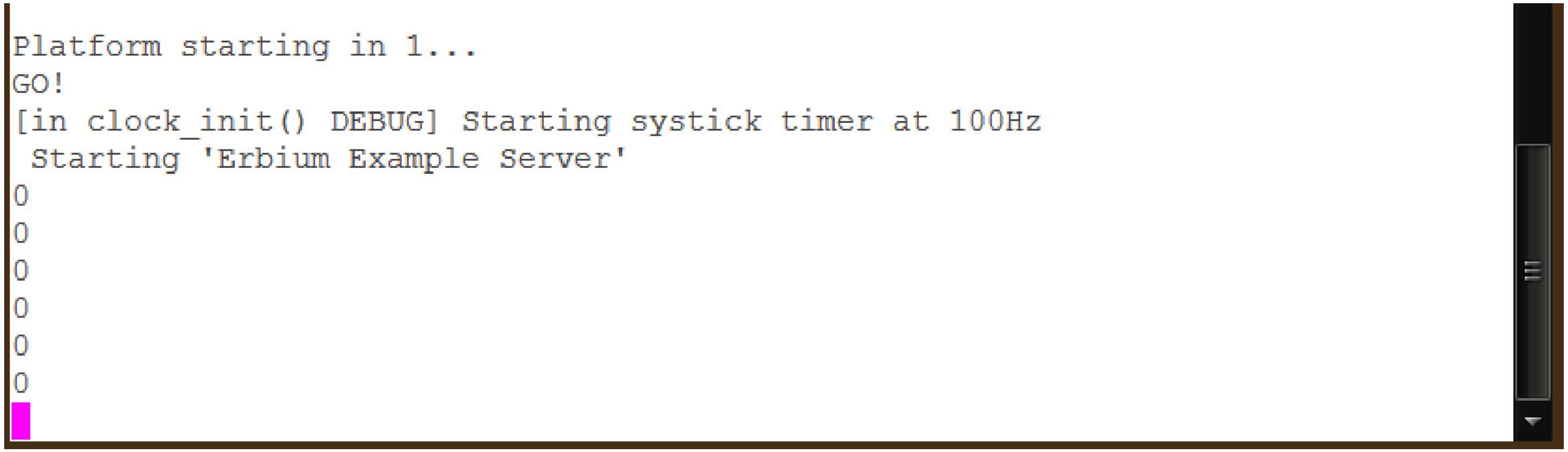
Run CoAP

- coap get coap://[IPv6 address of your site::XXXX]:5683/sensors/light
- coap get coap://[IPv6 address of your site::XXXX]:5683/my_res/new_alarm

Do you get these results?



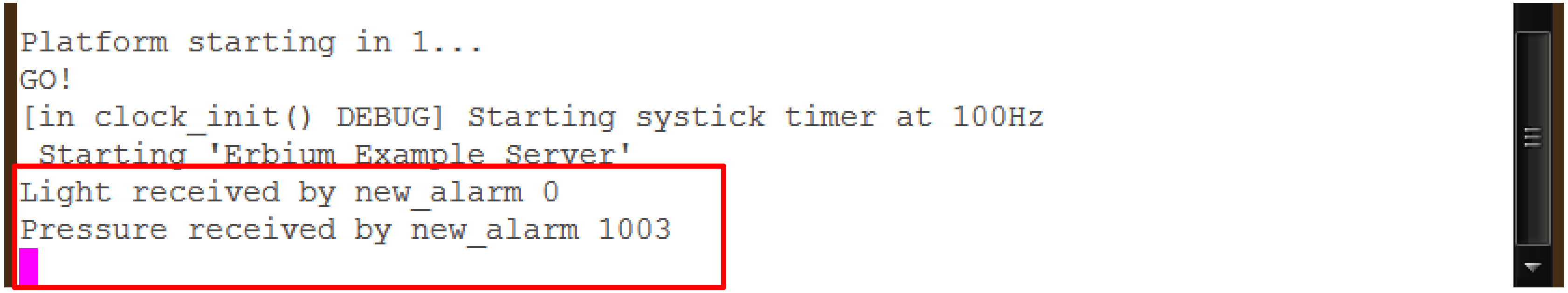
```
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources$ coap g
et coap://[2001:660:5307:3110::8877]:5683/sensors/light
(2.05)  0
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources$ coap g
et coap://[2001:660:5307:3110::8877]:5683/my_res/new_alarm
(2.05)  Alarm is 0
klim@grenoble:~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example/resources$
```



```
Platform starting in 1...
GO!
[in clock_init() DEBUG] Starting systick timer at 100Hz
 Starting 'Erbium Example Server'
0
0
0
0
0
0
```

Challenges #1

- **Try using other sensor values**
  - Easy to add them, just add more variables in the header



```
Platform starting in 1...
GO!
[in clock_init() DEBUG] Starting systick timer at 100Hz
 Starting 'Erbium Example Server'
Light received by new_alarm 0
Pressure received by new_alarm 1003
```

Challenges #2

- **Trigger the alarm based on the light/pressure value**
  - Using thresholds, detect the change in the light/pressure value
  - If there is a change, trigger the alarm

- **When the Front-end calls for the alarm data, send this trigger to the front-end**

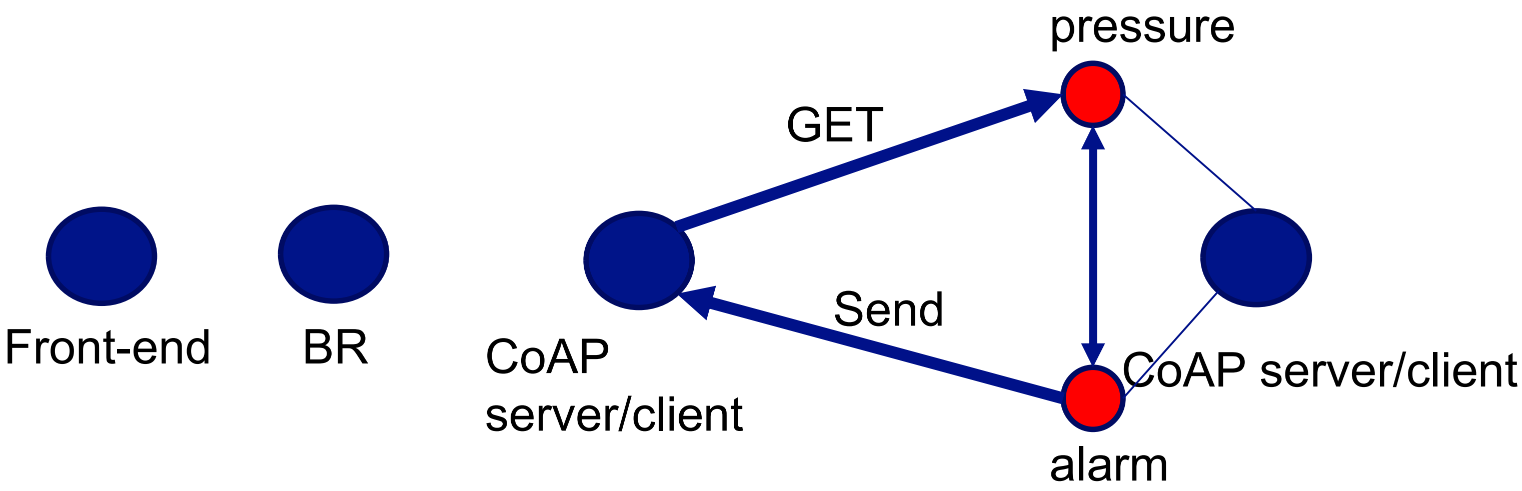
Proactive alarming

Our previous two experiments

- **Based on on-demand alarming**
  - It is needed, if the front-end does not have a server
    - It can only request for data
  - However, this is inefficient
    - Short periods to catch alarms in real-time, will cause energy usage
    - So the alarms should be proactive

Our final experiment

- **Proactive alarming**
  - Three sensors needed:
    - Two CoAP client/servers
    - BR (Just for the routing)



Ultimate challenge

- **Use three nodes, one BR, and two CoAP client/servers**
  - Create a proactive alarm reaction system
- **Procedure**
  1. CoAP node #1 (C#1) gets pressure data from CoAP node #2 (C#2) at a period of 5 seconds
  2. With a 20% chance, C#2 generates an alarm and sends to C#1
  3. When C#1 receives alarm, it reduces its period to 1 second
  4. C#1 continues until it receives another POST, returning to period of 5 seconds

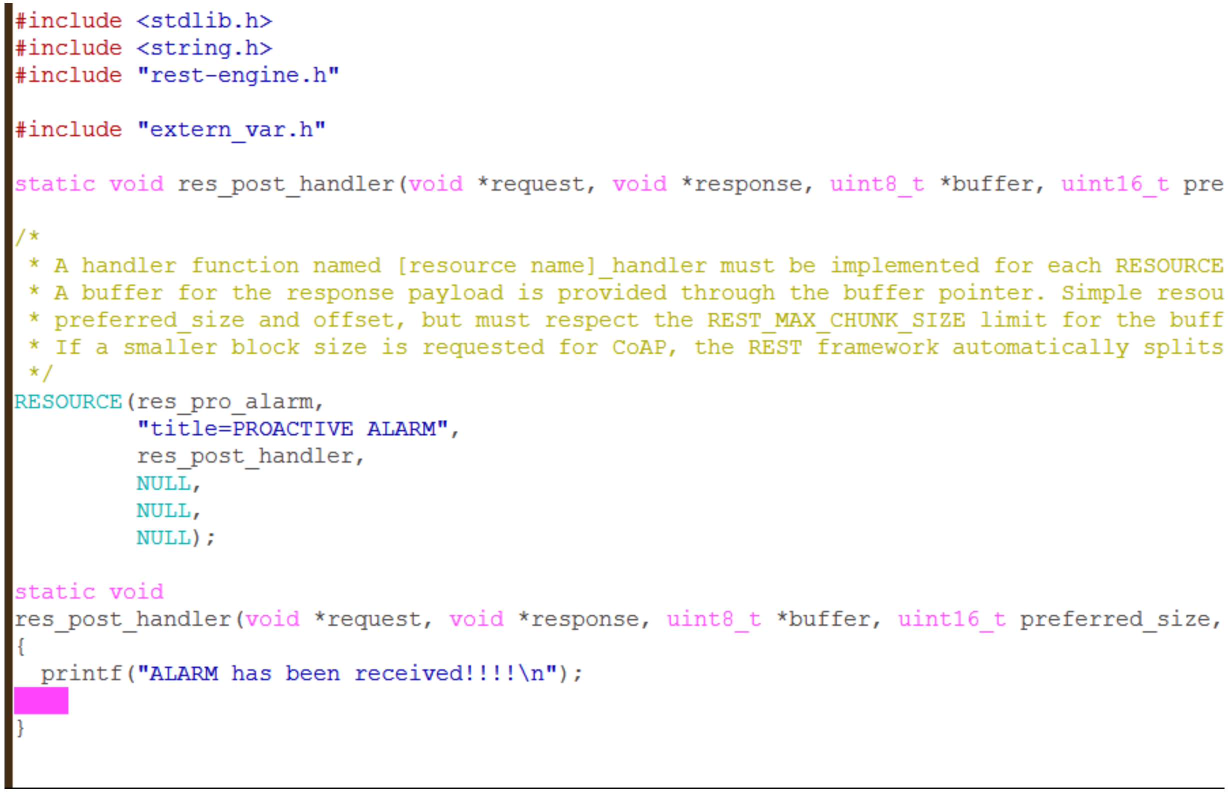
Hints

- **Based on last TPs, you must combine the CoAP client/server code together**
  - I will provide the sample code to everyone on the site

- **TOGGLE_INTERVAL is the one that defines the interval of GET, but it cannot be changed automatically**
  - So, change it to a variable instead
  - Make it a global variable!!(define in extern_var.h)

Hints #2

- **For simplicity, I designed a new resource called pro_alarm**



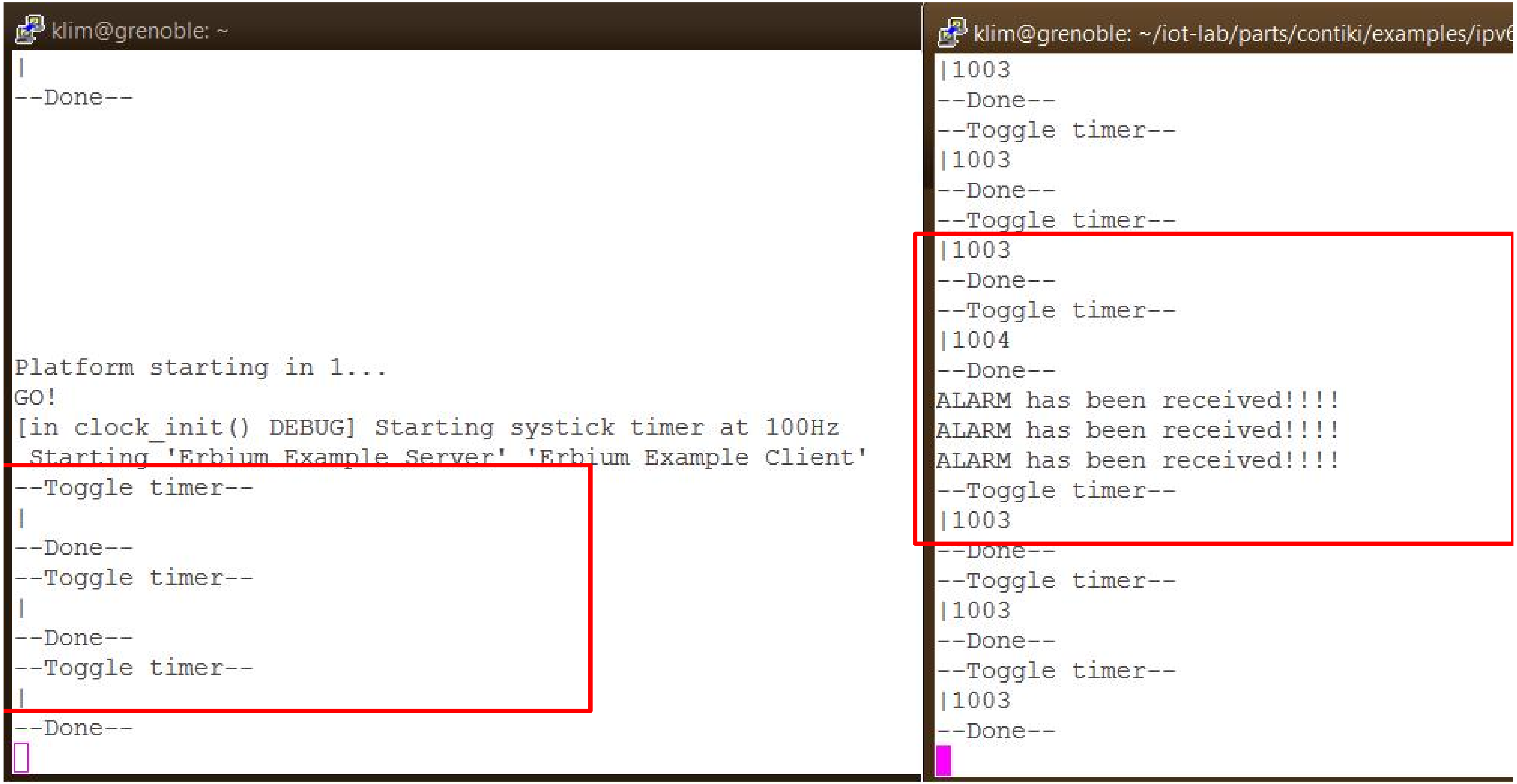
```
#include <stdlib.h>
#include <string.h>
#include "rest-engine.h"

#include "extern_var.h"

static void res_post_handler(void *request, void *response, uint8_t *buffer, uint16_t pre

/*
 * A handler function named [resource name]_handler must be implemented for each RESOURCE
 * A buffer for the response payload is provided through the buffer pointer. Simple resou
 * preferred_size and offset, but must respect the REST_MAX_CHUNK_SIZE limit for the buff
 * If a smaller block size is requested for CoAP, the REST framework automatically splits
 */
RESOURCE(res_pro_alarm,
         "title=PROACTIVE ALARM",
         res_post_handler,
         NULL,
         NULL,
         NULL);

static void
res_post_handler(void *request, void *response, uint8_t *buffer, uint16_t preferred_size,
{
  printf("ALARM has been received!!!!\n");

}
```

Hints #3

- **The codes for two CoAP client/server only needs to be different on:**
  - The address of each other
  - What kind of URL you are calling
    - One node requests for pressure, the other sends alarm
  - The server part and the resources can be identical

- **The answer code**
  - Will be provided to you next week

Anticipated results



klim@grenoble: ~

```
|
--Done--







Platform starting in 1...
GO!
[in clock_init() DEBUG] Starting systick timer at 100Hz
 Starting 'Erbium Example Server' 'Erbium Example Client'
--Toggle timer--
|
--Done--
--Toggle timer--
|
--Done--
--Toggle timer--
|
--Done--
```

klim@grenoble: ~/iot-lab/parts/contiki/examples/ipv6

```
|1003
--Done--
--Toggle timer--
|1003
--Done--
--Toggle timer--
|1003
--Done--
--Toggle timer--
|1004
--Done--
ALARM has been received!!!!
ALARM has been received!!!!
ALARM has been received!!!!
--Toggle timer--
|1003
--Done--
--Toggle timer--
|1003
--Done--
--Toggle timer--
|1003
--Done--
```