D Y PATIL
RAMRAO ADIK
INSTITUTE OF
TECHNOLOGY
NAVI MUMBAI

Q.1) Write a program in conitki os using Coojimulator for broadcasting data from sensors.

→
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "contiki.h"
#include "contiki_net.h"
#include "erbiom.h"
#include "dev/shtll -sensor.h"
#if with _COAP_ 3
#include "er -coap-07.h"
#elif WITH _COAP - 7
# include "er -coap - 07.h"
#elif WITH - COAP = 12
# include "er_coap_12.h"
#elif  WITH - COAP = 13
#include  "er- coap =13.h"
#else
#warning "Erbium example without COAP- specific functionality"
#endif
# define DEBUG 0
# if DEBUG
# define PRINTF(...) printf(_VA_ ARGS_)
# define PRINTGADDR (addr) PRINTF ("[%02 x %002 x %02 x %02 x %2 x %02
                                x.%02 x %02]". (uint 8-t)*)
     addr) [0],((unit 8-t *) addr (1) ... [15])
# define PRINTLLADDR (lladdr) PRINTF ("[ %002 x %02 x...]", (lladdr)...)
     -> addr (0), (lladdr) -> addr (1)... [5])
#else
# define PRINTF(....)
# define PRINTGADDR(addr)
# define PRINTLLADDR (addr)
# endif
  PERIODIC_RESOURCE(temperature, METHOD_GET, "sensors / temperature",
              "title =\"Temperature \" :obs", 120* clock_SECOND);
  Void temperature_handler (void* request, void * response, unit8_t* buffer,
   unit 6-t preferred size, int32_t * offset)
 {
   REST.set_header_content_type (response, REST.type.const char* msg
                         ="Temperature, periodic !" ; TEXT_PLAIN);
   REST.set_response.payload (response, msg, strlen (msg));
  }

   Void temperature_periodic_handler (resource_t* r)
  {

     static char content (50);
     unit16-t temperature = sht11-sensor.value (SHT11_SENSOR_TEMP);
     temperature = temperature/10;
     PRINTF ("%u/n" temperature);
     Coap_packet_t notification (1]);
     coap_init_message (notification, COAP_TYPE_NON, REST.status.OK,0);
     coap_set_payload (notification, content, snprintf (content, size of content),
                              "%u", temperature));

     REST.notify_subscribers (r, temperature, notification);
   }
```

Q.2) Explain and write a program for COAP protocol in Cooja Simulator.

→ I. The coap client establishes a connection with a server on the coap port 61616 & sets the et timer to a particular value. Everytime the et timer is expired the send-data(void) function is called, when it receives the response from the server for its request, the handle incoming data() function is called.

```
. ettimer_set (& set, 5* (CLOCK_SECOND );
  while (1) {
          PROCESS_YIELD();
      if (etimer_expired (&et)) {
             send_data ();
          etimer_result (& et);
        } else if (ev == tcpsp_event)
        { handle_incoming_data();
        }
    }
```

II. The coap client runs a timer which when resets, the client randomly selects a service id (resource) using randon_read() function & sets a request to the REST server.

```
send_data (void) function
int data size = 0;
int service_id = randon_read () % number of urls;
coap_packet_t * request = (coap_packet_t*)
allocate_buffer (size of (coap_packet_t ));
init_packet ( request );
coap_set_method (request, COAP_GET);
request ->id = xact_id ++;
request -> type = MESSAGE_TYPE_CON;
coap_set_header_url ( request, service_url's [service_id]);
```

when the server response returns back to the client it runs the handle_incoming_data() function which takes the packet, parses the message & prints the payload that it receives from the response :-

```
static void
handle_incoming_data()
{
   PRINTF("incoming packet size : %u\n", (unit 16_t) uip_datalen();
   if (init_buffer (COAP_DATA_BUFF_SIZE))
   {
      if (uip_newdata()) {
            coap_packet_t *response = (coap_packet_t *)
            allocate_buffer (size of (coap_packet_t));
            if (response) {
                  parse_message (response, uip_appdata, uip_datalen());
            response_handler (response);
            }
      }
      delete_buffer();
   }
```

Q.3) Explain the method for uploading the sensor data from gateway to cloud.

→ Thing speak is an open IOT platform for monitoring your data online. In thingspeak channel you can set the data as private or public according to your choice. Thing speak takes minimum of 15 seconds to update your readings. It's a great & very easy to use platform for building IOT projects.

Step 1 : Sign up for Thing speak

Step 2: Create a channel for your data.
ⓐ Once you sign in after your account verification, create a new channel by clicking "new channel" button.
ⓑ After clicking on new channel enter the name & description of data you want to upload on channel. Now enter name of your data in field section. After this click on save channel button to save

Step 3: Getting API key in thingspeak.
ⓐ click on API keys button to get your unique API key for uploading your CPU data.
ⓑ Now copy your "Write API key". we will use in this code.

Step 4: Code for Raspberry Pi
In code, just mention the API key of your channel and after that run the program. Also install the thingspeak library

Step 5: Check thingspeak site for data logging.
After completing these steps open your channel & you will see the data is updating into thingspeak website

like this you can send any sensor data connected with Raspberry Pi to the thingspeak cloud.