임베디드 시스템

2019년 2학기

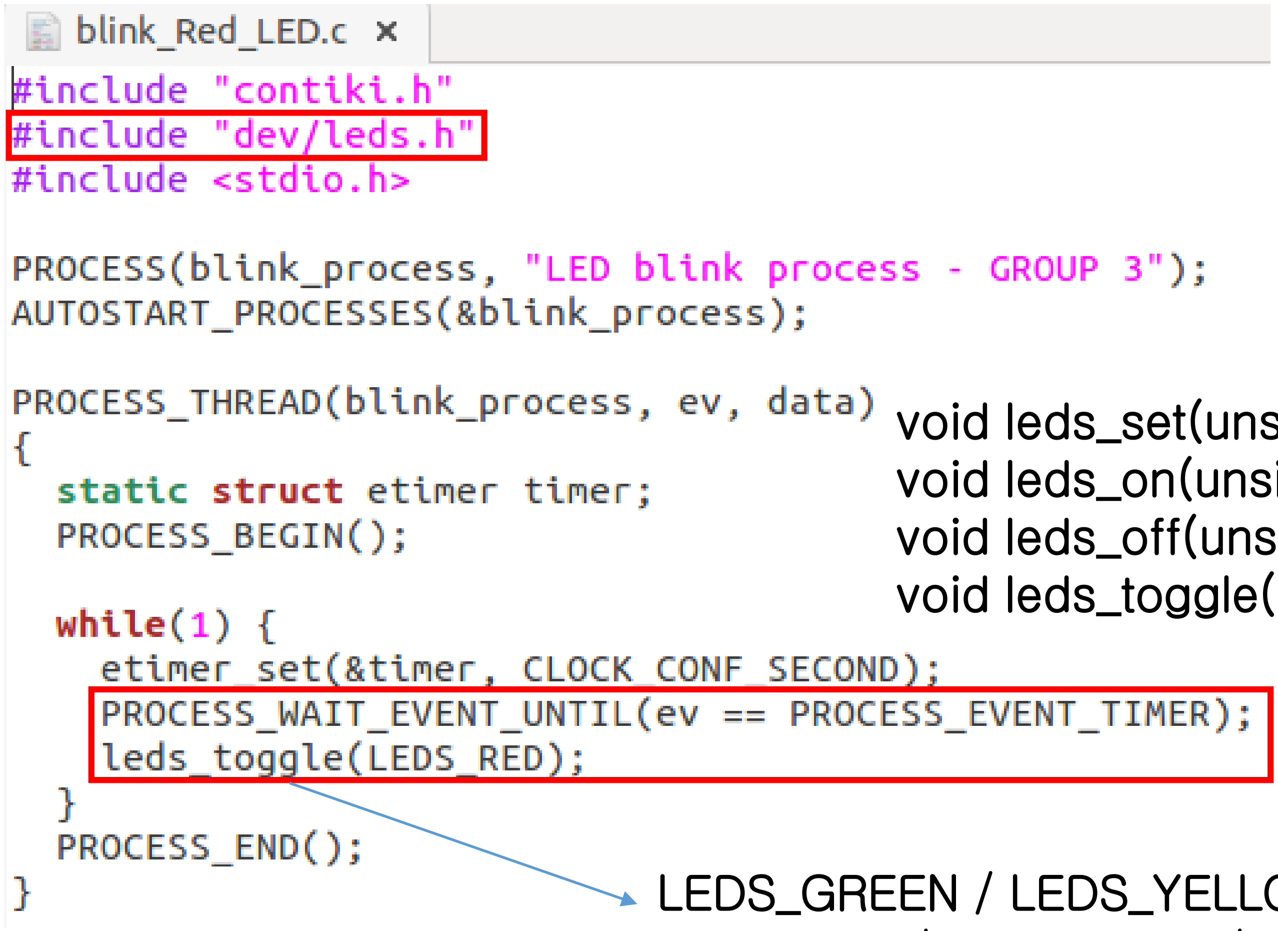
오승민 (smoh@kongju.ac.kr)

Today

- Contiki-OS

LED Control & UDP Communication in Contiki

blink_Red_LED.c



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

PROCESS(blink_process, "LED blink process - GROUP 3");
AUTOSTART_PROCESSES(&blink_process);

PROCESS_THREAD(blink_process, ev, data)
{
  static struct etimer timer;
  PROCESS_BEGIN();

  while(1) {
    etimer_set(&timer, CLOCK_CONF_SECOND);
    PROCESS_WAIT_EVENT_UNTIL(ev == PROCESS_EVENT_TIMER);
    leds_toggle(LEDS_RED);
  }
  PROCESS_END();
}
```

```c
void leds_set(unsigned char leds);
void leds_on(unsigned char leds);
void leds_off(unsigned char leds);
void leds_toggle(unsigned char leds);
```

LEDS_GREEN / LEDS_YELLOW
LEDS_RED / LEDS_BLUE / LEDS_ALL

RPL UDP

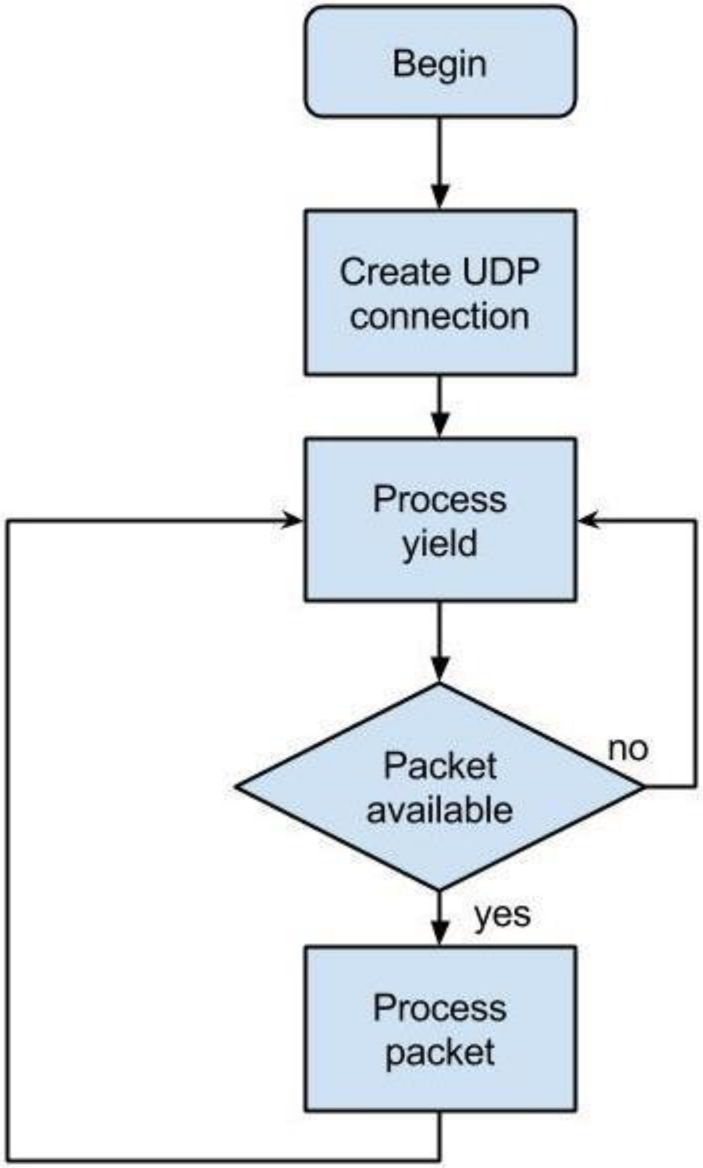
- RPL is the IPv6 Routing Protocol for Low-power and Lossy Networks (LLNs).
  - LLNs are a class of network in which both the routers and their interconnect are constrained.
    - LLN routers typically operate with constraints on processing power, memory, and energy.
  - RPL provides a mechanism whereby multipoint-to-point traffic from devices inside the LLN towards a central control point as well as point-to-multipoint traffic from the central control point to the devices inside the LLN are supported.

UDP example

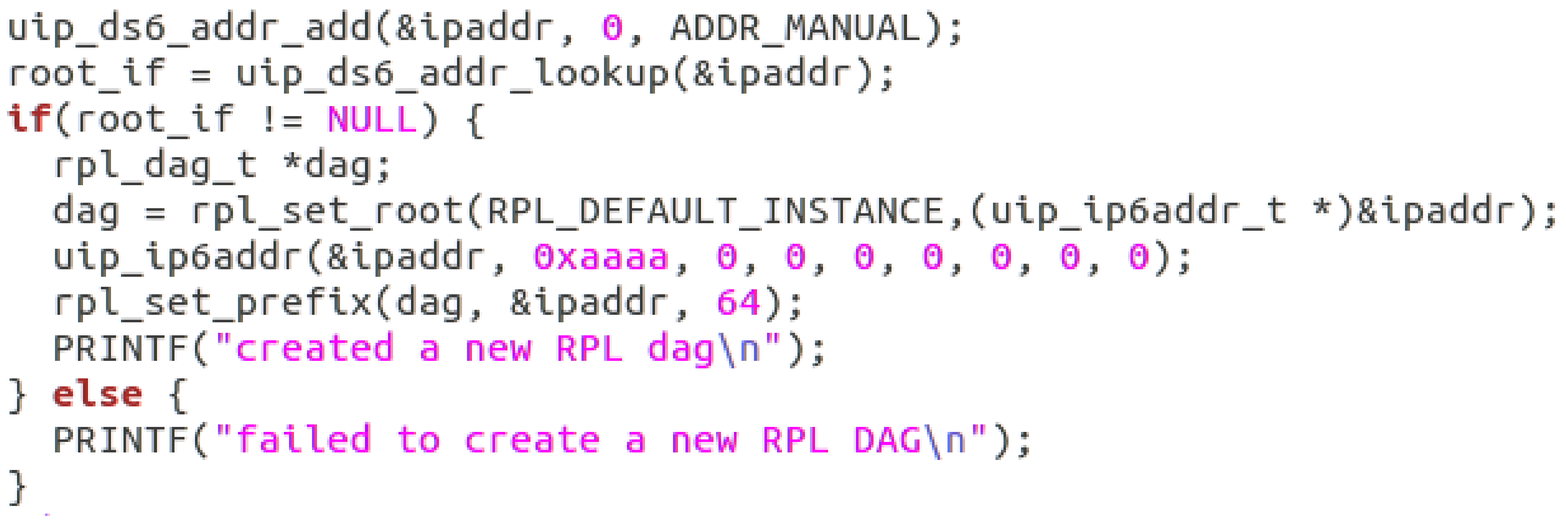
- Location: <contiki_folder>/examples/ipv6/rpl-upl/
- In this example, UDP is implemented on top of RPL.
  - A LLN is comprised of **a UDP server**, which accepts available packets, and **several UDP clients**, which send packets periodically to server through single-hop or multi-hops.

UDP Server

- In the example, UDP server does three tasks primarily.
    1. Initializes RPL DAG (Directed Acyclic Graph);
    2. Sets up UDP connection;
    3. Waits for packets from client, receives and print them on stdout.

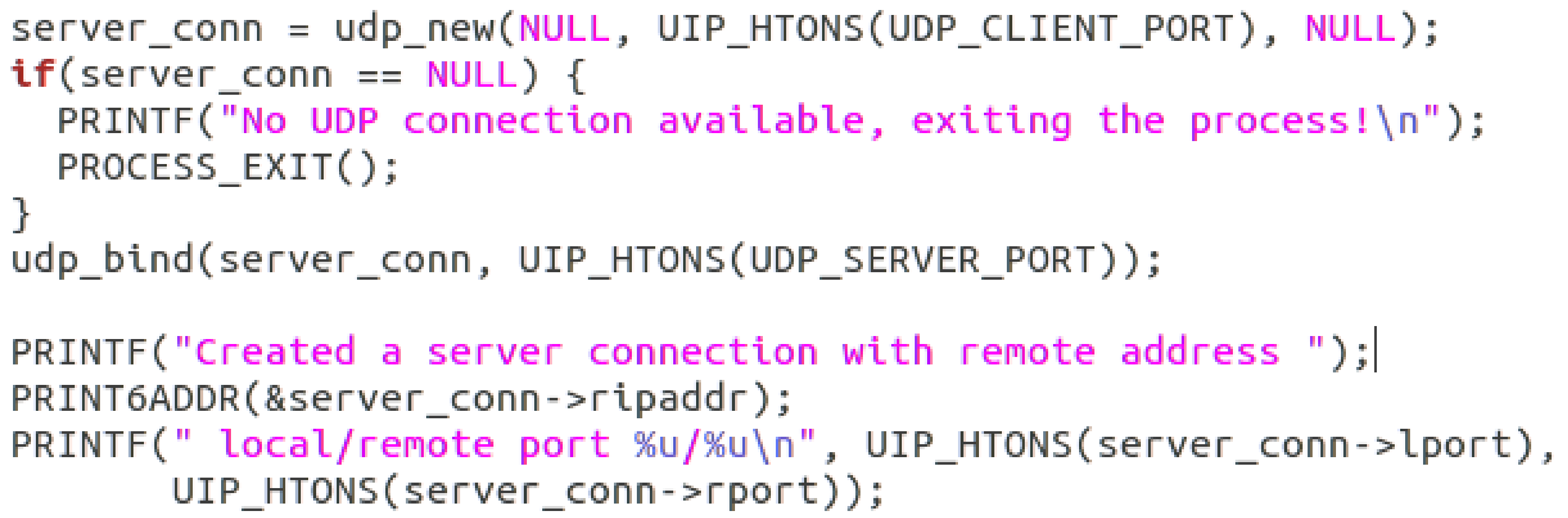


UDP Server - Initialize RPL DAG



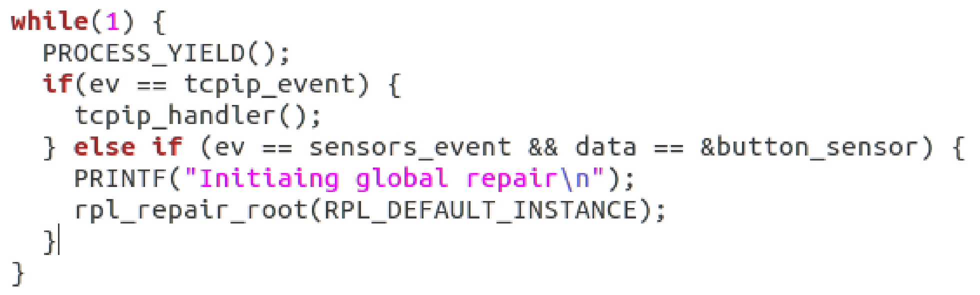
```
uip_ds6_addr_add(&ipaddr, 0, ADDR_MANUAL);
root_if = uip_ds6_addr_lookup(&ipaddr);
if(root_if != NULL) {
  rpl_dag_t *dag;
  dag = rpl_set_root(RPL_DEFAULT_INSTANCE,(uip_ip6addr_t *)&ipaddr);
  uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 0);
  rpl_set_prefix(dag, &ipaddr, 64);
  PRINTF("created a new RPL dag\n");
} else {
  PRINTF("failed to create a new RPL DAG\n");
}
```

UDP Server – Set up a UDP Connection

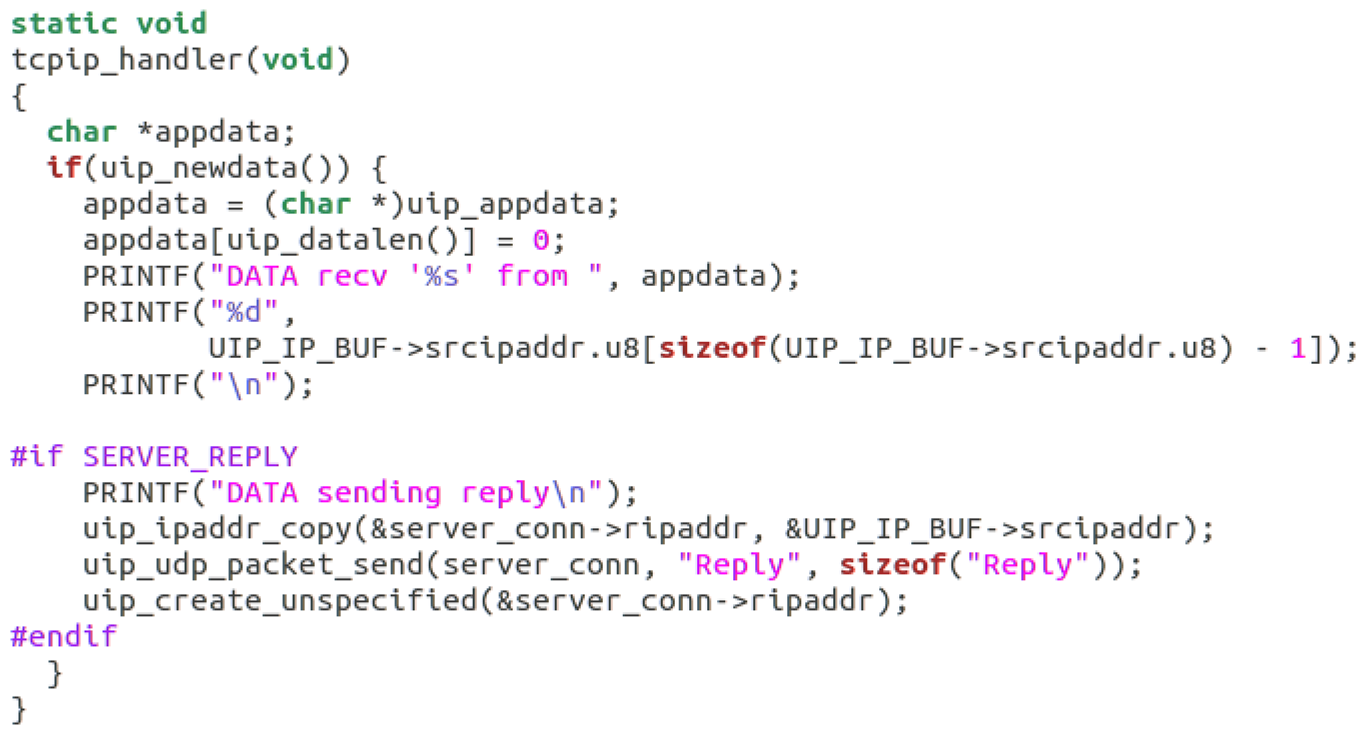


```
server_conn = udp_new(NULL, UIP_HTONS(UDP_CLIENT_PORT), NULL);
if(server_conn == NULL) {
  PRINTF("No UDP connection available, exiting the process!\n");
  PROCESS_EXIT();
}
udp_bind(server_conn, UIP_HTONS(UDP_SERVER_PORT));

PRINTF("Created a server connection with remote address ");
PRINT6ADDR(&server_conn->ripaddr);
PRINTF(" local/remote port %u/%u\n", UIP_HTONS(server_conn->lport),
       UIP_HTONS(server_conn->rport));
```

UDP Server – Receiving and Processing Incoming Packet



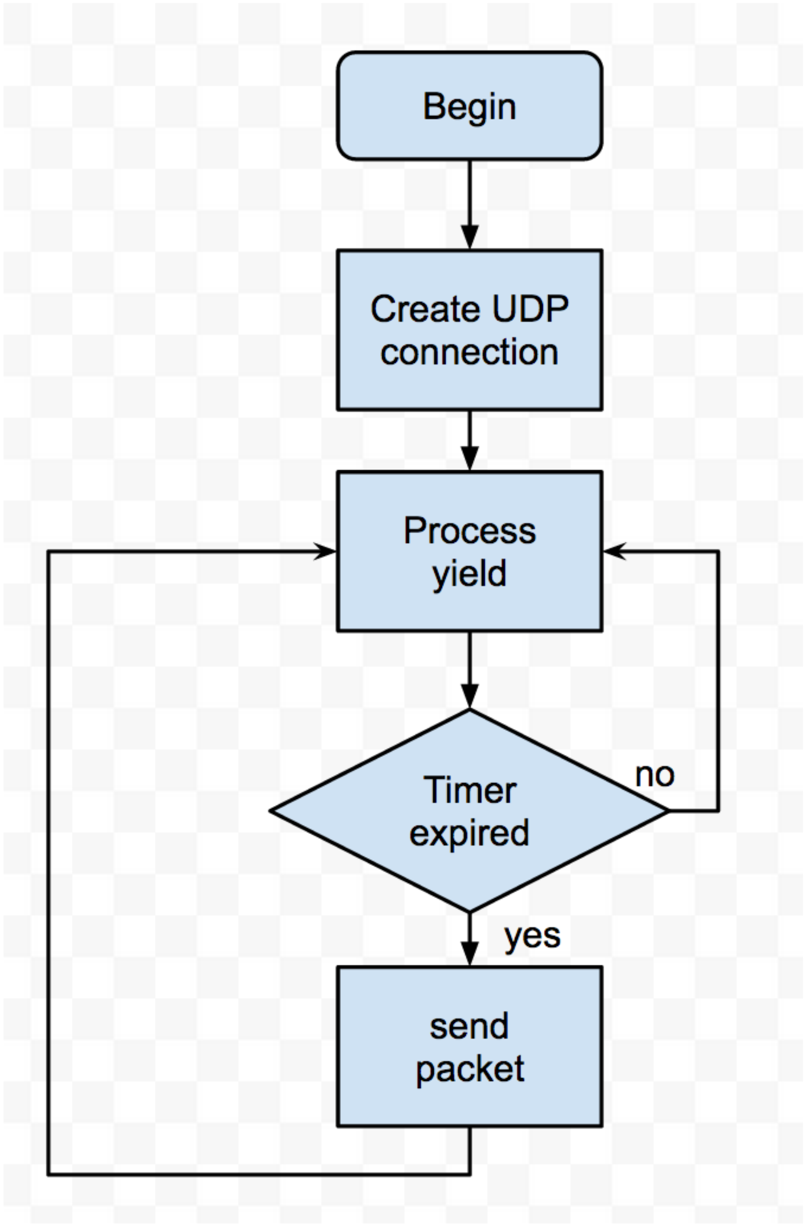
```
while(1) {
  PROCESS_YIELD();
  if(ev == tcpip_event) {
    tcpip_handler();
  } else if (ev == sensors_event && data == &button_sensor) {
    PRINTF("Initiaing global repair\n");
    rpl_repair_root(RPL_DEFAULT_INSTANCE);
  }
}
```



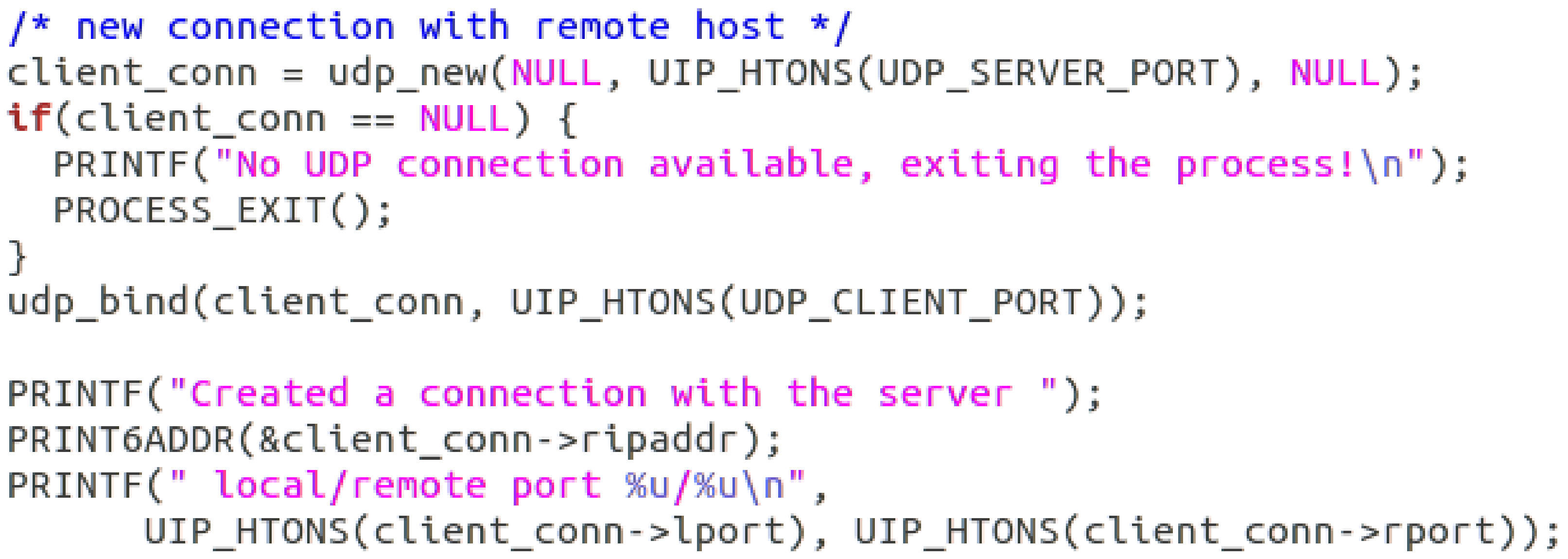
```
static void
tcpip_handler(void)
{
  char *appdata;
  if(uip_newdata()) {
    appdata = (char *)uip_appdata;
    appdata[uip_datalen()] = 0;
    PRINTF("DATA recv '%s' from ", appdata);
    PRINTF("%d",
           UIP_IP_BUF->srcipaddr.u8[sizeof(UIP_IP_BUF->srcipaddr.u8) - 1]);
    PRINTF("\n");

#if SERVER_REPLY
    PRINTF("DATA sending reply\n");
    uip_ipaddr_copy(&server_conn->ripaddr, &UIP_IP_BUF->srcipaddr);
    uip_udp_packet_send(server_conn, "Reply", sizeof("Reply"));
    uip_create_unspecified(&server_conn->ripaddr);
#endif
  }
}
```

UDP Client

- In the example, UDP client does two tasks primarily.
    1. Sets up UDP connection;
    2. Sends packet to UDP server periodically.

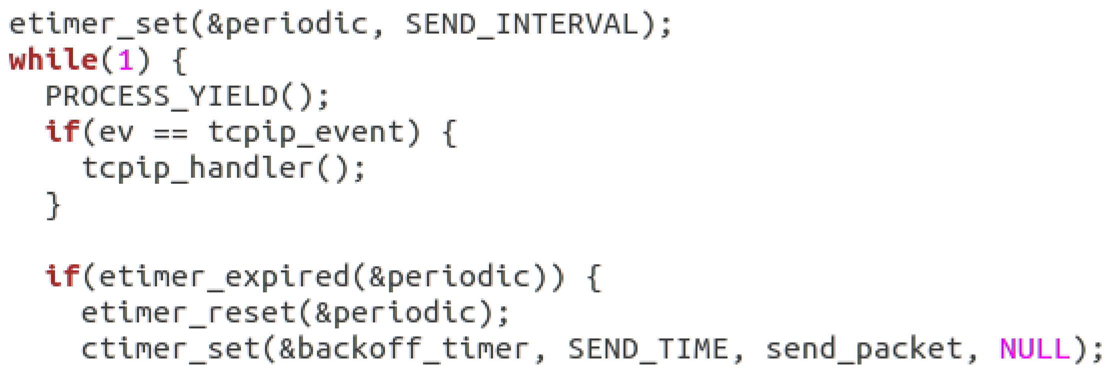


UDP Client – Set up UDP Connection

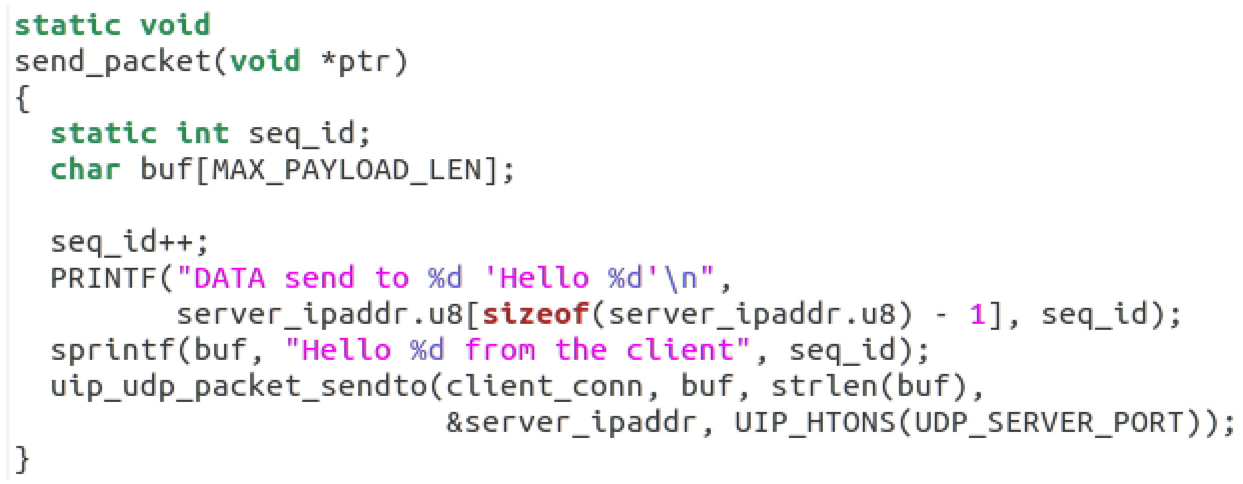


```
/* new connection with remote host */
client_conn = udp_new(NULL, UIP_HTONS(UDP_SERVER_PORT), NULL);
if(client_conn == NULL) {
  PRINTF("No UDP connection available, exiting the process!\n");
  PROCESS_EXIT();
}
udp_bind(client_conn, UIP_HTONS(UDP_CLIENT_PORT));

PRINTF("Created a connection with the server ");
PRINT6ADDR(&client_conn->ripaddr);
PRINTF(" local/remote port %u/%u\n",
      UIP_HTONS(client_conn->lport), UIP_HTONS(client_conn->rport));
```

UDP Client – Sends Packet



```
etimer_set(&periodic, SEND_INTERVAL);
while(1) {
  PROCESS_YIELD();
  if(ev == tcpip_event) {
    tcpip_handler();
  }

  if(etimer_expired(&periodic)) {
    etimer_reset(&periodic);
    ctimer_set(&backoff_timer, SEND_TIME, send_packet, NULL);
```



```
static void
send_packet(void *ptr)
{
  static int seq_id;
  char buf[MAX_PAYLOAD_LEN];

  seq_id++;
  PRINTF("DATA send to %d 'Hello %d'\n",
         server_ipaddr.u8[sizeof(server_ipaddr.u8) - 1], seq_id);
  sprintf(buf, "Hello %d from the client", seq_id);
  uip_udp_packet_sendto(client_conn, buf, strlen(buf),
                        &server_ipaddr, UIP_HTONS(UDP_SERVER_PORT));
}
```

Today’s mission

- Run blink_Red_LED.c to control the LEDs of motes.
- Toggle the LEDs of a UDP server according to the UDP sender IDs.
  - For example,
    - If sender is Node 2, toggle the RED led.
    - If sender is Node 3, toggle the BLUE led.
    - If sender is Node 4, toggle the GREEN led.