```
/*Name: Darren Smith
/*URN: 6553517
/*Assignment 1: FireAlarm
/*Date: 3th December 2019
/*
/*
 */



#include "contiki.h"
#include "lib/random.h"
#include "net/rime.h"
#include "etimer.h"
#include <stdio.h>
#include "dev/light-sensor.h" //Light Sensor header file
#include "dev/sht11-sensor.h" // Temperature and Humidity Header File
#include "dev/leds.h" // LED Header file
#include "dev/button-sensor.h" //Sensor button
#include "net/rime.h" //Network Unicast test
#include "node-id.h"



/*-------------------------------------*/

static uint8_t blinks;
/*-------------------------------------*/


/*lets define some digits before the decimal point */

unsigned short d1(float f) // Integer part
{

        return((unsigned short)f);
}

unsigned short d2(float f) // Fractional part

{

return(1000*(f-d1(f)));

}




/*---------------------------------------*/



PROCESS(Alarm, "fire Alarm");
```

```
AUTOSTART_PROCESSES(&Alarm);


/* unicast callback
* This method gets called in case that a unicast is received

/*-------------------------------------*/
static void
recv_uc(struct unicast_conn *c, const rimeaddr_t *from)
{
        printf("unicast received from %d.%d\n",
        from->u8[0], from->u8[1]);
}
static const struct unicast_callbacks unicast_callbacks = {recv_uc};
static struct unicast_conn uc;
/*-----------------------------------------------*/



PROCESS_THREAD(Alarm, ev, data)
{


        static struct etimer Alarmtimer; // Create a process which gives a Timer
        static struct etimer et;
        rimeaddr_t addr;

        PROCESS_EXITHANDLER(unicast_close(&uc);)

        PROCESS_BEGIN();

        etimer_set(&Alarmtimer, CLOCK_CONF_SECOND/4); //Configuring Time 1SEC
        SENSORS_ACTIVATE(sht11_sensor);
        SENSORS_ACTIVATE(light_sensor); //Activate the light sensor
        leds_off(LEDS_ALL);
        SENSORS_ACTIVATE(button_sensor);

        unicast_open(&uc, 128, &unicast_callbacks);



        while(1) //start of the while loop
        {
                PROCESS_WAIT_EVENT_UNTIL(ev=PROCESS_EVENT_TIMER); // Wait4Time
                etimer_set(&et, CLOCK_SECOND * 2 + random_rand() % (CLOCK_SECOND
* 2));

                PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));


                //This is where we get the sensor values for light, temp and
humdity

                float temp = 0.01*sht11_sensor.value(SHT11_SENSOR_TEMP)-39.6;
```

```
                float V_sensor =
1.5*light_sensor.value(LIGHT_SENSOR_PHOTOSYNTHETIC)/4096;
                float I = V_sensor/100000;
                float light_lx = 0.625*1e6*I*1000;

                printf("Room Temperature is : %u.%03u C\n" , d1(temp),
d2(temp));
                printf("\n");
                printf("Room Light Temperature is : %u.%03u lux\n",
d1(light_lx), d2(light_lx));
                printf("\n");

/*-------------------------------------------------------------------------
--------------*/
                if(temp<28 && temp > 26) // Implement a threshold
                {
                leds_off(LEDS_ALL);
                printf(" FIRE THREAT RISING!\n");
                printf("\n");
                leds_on(LEDS_GREEN);
                blinks++;
                }

                else if(temp > 28 && temp < 100)
                {
                leds_off(LEDS_ALL);
                printf(" FIRE!\n");
                printf("\n");
                leds_on(LEDS_BLUE);
                blinks++;

                //Address for the Nodes
                packetbuf_copyfrom("Fire",5);
                addr.u8[0] = 0;
                addr.u8[1] = 1;

                        if(!rimeaddr_cmp(&addr, &rimeaddr_node_addr)) {

                        unicast_send(&uc, &addr);
                        printf("Fire: Call the fire Brigade!\n"); //Sending
message to the Server.
                        printf("\n");
                        leds_on(LEDS_RED);
                        blinks++;
                        }
                }
                else{
                leds_off(LEDS_ALL);
                unicast_send(&uc, &addr);
                printf("Threat has passed!\n"); //Sending message to the Server.
                printf("\n");
                }
```

```
                if(ev ==sensors_event);
                {
                        if(data == &button_sensor)
                        {


                        etimer_restart(&Alarmtimer);
                        }

                }


                etimer_reset(&Alarmtimer); //Reset the Timer
        }




        PROCESS_END();
}



/*---------------------------------------------------------------------------*/
```