```
#include "control.h"
#include "display.h"
#include "range_finder.h"
#include "switch.h"
#include "to_7seg.h"


// extended state ------------------------------------------------------
  // "basic" state
typedef enum {CTRL_START, CTRL_LED, CTRL_WAIT} ctrl_state_t;
static ctrl_state_t   g_ctrl_state;

  // externally reachable objects
bool volatile         gb_ctrl_can_sleep;        // this FSM can sleep

  // hardware resources
static DigitalOut     *gp_ctrl_ldl;             // LEDS
//static InterruptIn    *gp_ctrl_swm;             // switch
static AnalogIn       *gp_ctrl_lit;             // LDR

Timeout to;

  // internal objects
static bool           gb_ctrl_initd;            // true after call to ctrl_init()
static bool volatile  gb_ctrl_to_evnt;          // timeout evnt
static int32_t        g_dist;
static int32_t        g_delay_us;
static int16_t        luz;
static bool volatile to_evnt;

// end of extended state  ----------------------------------------------

// ISRs ----------------------------------------------------------------
static void to_isr(void){
  to_evnt = true;
}

// end of ISRs  --------------------------------------------------------

// FSM  ----------------------------------------------------------------
void ctrl_fsm (void) {
  if (gb_ctrl_initd) {  // protect against calling ctrl_fsm() w/o a previous call to
ctrl_init()

    switch(g_ctrl_state){

      case CTRL_LED:

      if(to_evnt){
        to_evnt = false;
        to.detach();
        *gp_ctrl_ldl = 0;

        g_delay_us = 1000;

        if(g_dist > 0){
          g_delay_us =  g_delay_us + (1420* g_dist);
        }
        g_delay_us = (g_delay_us > 1300000  ? 1300000 : g_delay_us);

        to.attach_us(to_isr,g_delay_us);

         g_ctrl_state = CTRL_WAIT;

       }
      break;


      case CTRL_WAIT:

      if(to_evnt){
        to_evnt = false;
        to.detach();

        gb_rf_start_msg = true;
        gb_display_update_msg = true;

        if(g_dist > 99){

        g_display_segs = 0x4040;

        }else if(g_dist > 0){

          g_display_segs = (to_7seg(g_dist/10)<<8) | to_7seg(g_dist%10);
```

```
84
85            }else if(-8 == g_dist){
86               g_display_segs = 0x7950;
87
88            }else{
89               g_display_segs = 0;
90            }
91            g_ctrl_state = CTRL_START;
92
93              gb_display_brightness_msg = true;
94             luz = gp_ctrl_lit -> read_u16()/656;
95             g_display_brightness = 0.39 * luz +1;
96          }
97          break;
98
99          default: //CTRL_START
100           to_evnt = false; // evento irrelevante
101
102
103         if(gb_rf_done_msg){
104           *gp_ctrl_ldl = 1;
105           g_dist = g_rf_range_cm-7;
106           to.attach_us(to_isr,200000);
107           g_ctrl_state = CTRL_LED;
108         }
109
110         break;
111
112       }// fin switch
113
114       //dormir micro
115       __disable_irq();
116       if(!to_evnt && !gb_rf_done_msg){
117         gb_ctrl_can_sleep = true;
118       }
119       __enable_irq();
120
121     } // if (gb_ctrl_initd)
122   }
123   // end of FSM ----------------------------------------------------------
124
125   // initialize FSM machinery  -------------------------------------------
126   void ctrl_init (DigitalOut *ldl, AnalogIn *lit, InterruptIn *swm) {
127     if (!gb_ctrl_initd) {
128       gb_ctrl_initd = true;    // protect against multiple calls to ctrl_init
129
130       g_dist = 0;
131       g_delay_us = 0;
132       to_evnt = false;
133
134       gp_ctrl_ldl = ldl;
135       gp_ctrl_lit = lit;
136       *gp_ctrl_ldl = 0;
137
138       gb_rf_start_msg = true;
139
140       gb_display_on_msg = true;
141       g_display_segs = 0x5454;
142       gb_display_brightness_msg = true;
143       luz = gp_ctrl_lit -> read_u16()/656;
144       g_display_brightness = 0.39 * luz +1;
145       g_ctrl_state = CTRL_START;
146
147     }
148   }
149   // end of FSM initialization  ------------------------------------------
150
```