```
1   #include "control.h"
2   #include "display.h"
3   #include "range_finder.h"
4   #include "switch.h"
5   #include "to_7seg.h"
6
7
8   // extended state ------------------------------------------------------
9     // "basic" state
10  typedef enum {CTRL_START, CTRL_LED, CTRL_WAIT, CTRL_OFF} ctrl_state_t;
11  static ctrl_state_t   g_ctrl_state;
12
13    // externally reachable objects
14  bool volatile         gb_ctrl_can_sleep;        // this FSM can sleep
15
16    // hardware resources
17  static DigitalOut     *gp_ctrl_ldl;             // LEDS
18  static InterruptIn    *gp_ctrl_swm;             // switch
19  static AnalogIn       *gp_ctrl_lit;             // LDR
20
21  Timeout to;
22
23    // internal objects
24  static bool           gb_ctrl_initd;            // true after call to ctrl_init()
25  static bool volatile  gb_ctrl_to_evnt;          // timeout evnt
26  static int32_t        g_dist;
27  static int32_t        g_delay_us;
28  static int16_t        luz;
29  static bool volatile to_evnt;
30
31  // end of extended state  ----------------------------------------------
32
33  // ISRs ----------------------------------------------------------------
34  static void to_isr(void){
35    to_evnt = true;
36  }
37
38  // end of ISRs  --------------------------------------------------------
39
40  // FSM  ----------------------------------------------------------------
41  void ctrl_fsm (void) {
42    if (gb_ctrl_initd) {  // protect against calling ctrl_fsm() w/o a previous call to
    ctrl_init()
43
44      switch(g_ctrl_state){
45
46        case CTRL_LED:
47
48        gb_swm_long_msg = false; // irrelevante
49        gb_rf_done_msg = false;
50
51        if(to_evnt){
52          to_evnt = false;
53          to.detach();
54          *gp_ctrl_ldl = 0;
55
56          g_delay_us = 1000;
57
58          if(g_dist > 0){
59            g_delay_us =  g_delay_us + (1420* g_dist);
60          }
61          g_delay_us = (g_delay_us > 1300000  ? 1300000 : g_delay_us);
62
63          to.attach_us(to_isr,g_delay_us);
64
65           g_ctrl_state = CTRL_WAIT;
66
67         }
68        break;
69
70
71        case CTRL_WAIT:
72
73        gb_swm_long_msg = false; // irrelevante
74
75        if(to_evnt){
76
77          if( 0 == *gp_ctrl_swm){
78          to_evnt = false;
79          to.detach();
80
81          gb_rf_start_msg = true;
82          gb_display_update_msg = true;
83
```

```
84            if(g_dist > 99){
85
86            g_display_segs = 0x4040;
87
88            }else if(g_dist > 0){
89
90              g_display_segs = (to_7seg(g_dist/10)<<8)  | to_7seg(g_dist%10);
91
92            }else if(-8 == g_dist){
93               g_display_segs = 0x7950;
94
95            }else{
96               g_display_segs = 0;
97            }
98            g_ctrl_state = CTRL_START;
99
100            gb_display_brightness_msg = true;
101            luz = gp_ctrl_lit -> read_u16()/656;
102            g_display_brightness = 0.39 * luz +1;
103
104         }else{
105           gb_display_off_msg = true;
106            g_ctrl_state = CTRL_OFF;
107         }
108       }
109         break;
110
111
112
113
114
115         case CTRL_OFF:
116
117         to_evnt = false; // evento irrelevante
118         gb_rf_done_msg = false;
119
120         if(gb_swm_long_msg){
121           gb_swm_long_msg = false;
122           gb_rf_start_msg = true;
123           gb_display_on_msg = true;
124           gb_display_brightness_msg = true;
125           g_display_segs = 0x5454;
126           g_display_brightness = 100;
127           g_ctrl_state = CTRL_START;
128         }
129         break;
130
131
132         default: //CTRL_START
133           to_evnt = false; // evento irrelevante
134           gb_swm_long_msg = false; // irrelevante
135
136
137         if(gb_rf_done_msg){
138           *gp_ctrl_ldl = 1;
139           g_dist = g_rf_range_cm-7;
140           to.attach_us(to_isr,200000);
141           g_ctrl_state = CTRL_LED;
142         }
143
144         break;
145
146       }// fin switch
147
148       //dormir micro
149       __disable_irq();
150       if(!to_evnt && !gb_rf_done_msg && !gb_swm_long_msg){
151        gb_ctrl_can_sleep = true;
152       }
153       __enable_irq();
154
155     } // if (gb_ctrl_initd)
156   }
157   // end of FSM ------------------------------------------------------------
158
159   // initialize FSM machinery  ---------------------------------------
160   void ctrl_init (DigitalOut *ldl, AnalogIn *lit, InterruptIn *swm) {
161     if (!gb_ctrl_initd) {
162       gb_ctrl_initd = true;    // protect against multiple calls to ctrl_init
163
164       g_dist = 0;
165       g_delay_us = 0;
166       to_evnt = false;
167
```

```
168       gp_ctrl_ldl = ldl;
169       gp_ctrl_lit = lit;
170       gp_ctrl_swm = swm;
171       *gp_ctrl_ldl = 0;
172
173       gb_display_off_msg = true;
174       g_ctrl_state = CTRL_OFF;
175
176     }
177   }
178   // end of FSM initialization  ------------------------------------------
179
```