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