

```

1  #include "mbed.h"
2  #include "pinout.h"
3  #include "hardware.h"
4  #include "to_7seg.h"
5  #include "range_finder.h"
6  #include "switch.h"
7
8  // mux stuff
9  static Ticker      g_mux_tick;
10 static bool volatile gb_mux_evnt;
11
12 static void mux_isr (void) {
13     gb_mux_evnt = true;
14 }
15
16 // range_finder automatic start
17 static Ticker      g_meas_tick;
18 static bool volatile gb_meas_evnt;
19
20 static void meas_isr (void) {
21     gb_meas_evnt = true;
22 }
23
24 //variable en curso
25 bool volatile en_curso;;
26
27 int main (void) {
28     //variable en curso
29     en_curso = false;
30
31     // right display on?
32     bool      b_right = false;
33     // auto measure?
34     bool      b_auto_measure = false;
35     // the 4 MSB of this variable hold the symbol to be displayed at the
36     // left display, the 4 LSB that to be displayed at the right one
37     uint8_t   disp = 0;
38
39     hw_init();
40
41     g_seven_seg = 0;
42     g_dsr = b_right;
43     g_dsl = !b_right;
44     g_mux_tick.attach_us(mux_isr, 4000);      // 250 Hz
45
46     // initialize the range finder FSM
47     rf_init(&g_trg, &g_ech);
48
49     // initialize the middle switch FSM
50     swm_init(&g_swm);
51
52     // -----
53     for (;;) {
54
55         // the range finder FSM
56         rf_fsm();
57
58         // the middle switch FSM
59         swm_fsm();
60
61         //codigo
62         if(gb_swm_long_msg){
63             gb_swm_long_msg = false;
64
65             if(!en_curso){
66                 en_curso = true;
67                 g_meas_tick.attach_us(meas_isr, 100000);
68
69             }else{
70                 en_curso = false;
71                 g_meas_tick.detach();
72             }
73         }
74
75         if(gb_swm_msg && !en_curso){
76             gb_swm_msg = false;
77             en_curso = false;
78             meas_isr();
79         }
80         // -----
81
82         // start a new range measurement every 100 ms
83         if (gb_meas_evnt) {
84             gb_meas_evnt = false;

```

```

85     gb_rf_start_msg = true;
86 }
87
88 // when the measurement is complete, update variable disp
89 if (gb_rf_done_msg) {
90     gb_rf_done_msg = false;
91     if (g_rf_range_cm > 99) {
92         disp = 0xBB; // --
93     } else if (-1 == g_rf_range_cm) {
94         disp = 0xEC; // Er
95     } else {
96         disp = (g_rf_range_cm / 10) << 4 | (g_rf_range_cm % 10);
97     }
98 }
99
100 // display multiplex
101 if (gb_mux_evnt) {
102
103     gb_mux_evnt = false;
104     b_right = !b_right;
105     if(b_right){
106         g_dsr = 1;
107         g_dsl= 0;
108         g_seven_seg = (g_rf_range_cm > 99 ) ? disp : to_7seg(disp%10);
109
110     }else{
111         g_dsr = 0;
112         g_dsl= 1;
113         g_seven_seg = (g_rf_range_cm > 99 ) ? disp : to_7seg(disp/10);
114
115     }
116 }
117
118 // sleep
119 __disable_irq();
120 if (!gb_meas_evnt && !gb_rf_done_msg && !gb_mux_evnt
121     && !gb_rf_start_msg && gb_rf_can_sleep
122     && !gb_swm_msg && !gb_swm_long_msg && gb_swm_can_sleep) {
123     __WFI();
124 }
125 __enable_irq();
126 } // forever
127 } // main()
128
129

```