

```

1  /*****
2  *   Virtual hoop with 2D G sensor on PIC16F88
3  *   C source with CCS
4  *   File Name: 201103_virtual_hoop.c
5  *   Version: 1.0
6  *   Description: check function, start, zero and G sensor
7  *   Finally turn on 3 LEDs and beep
8  *   COPYRIGHT 2011 MYCOMKITS.COM, owned by CNET LIMITED
9  *   当プログラムの著作権は、製作者「マイコンキットドットコム運営 有限会社クネット」に帰属します。
10  *   著作権を放棄していませんが、当プログラムを使った学習の中でプログラムを自由に変更してお使いください。
11  *****/
12 // include header file
13 #include <16f88.h>
14 #device ADC=8 //this is required
15 #fuses INTRC IO, NOWDT, NOMCLR, NOBROWNOUT, PUT, NOLVP, NOPROTECT
16 #fuses NOCPD, NOWRT
17
18 #use delay(CLOCK=4000000)
19 #use fast_io(a)
20 #use fast_io(b)
21
22 #ROM 0x2100 = {0x01, 0x01}
23
24 #define saved_address 0
25 #define Bmode 0x00 //port B all output
26 #define Amode 0xFF //port A all input
27
28 // port define for output and function
29 #define green_led PIN_B7 //up and set switch active low
30 #define yellow_led PIN_B6 //function switch active low
31 #define red_led PIN_B5 //up and set switch active low
32 #define func_switch PIN_A4 //function switch active low
33 #define zero_switch PIN_A5 //zero calibration switch active low
34 #define start_switch PIN_A3 //start switch active low
35
36 ///// define variables
37 float Zaxis_G=0; //z axis G value
38 float Xaxis_G=0; //x axis G value
39 float vector_G=0; //
40 float vector_G_temp=0;
41 float vector_G_sum=0;
42 float vector_G_integ=0;
43 float Xoffset=0; //x offset
44 float Zoffset=0; //z offset
45 float Gmax =0.1; //G max default 0.1G
46 int Gmax_i =1; //G max integer
47 float Gmax_old; //old value
48 long int count1=0;
49 long int n;
50 int statusB;
51 float vector_norm;
52 int vector_norm_i;
53 int moving_ave=20; // 20 times
54
55 int Prev_cal_sw = 1; //previous status for cal switch
56 int Prev_f_sw = 1; //previous status for function switch
57 int Prev_s_sw = 1; //previous status for start switch
58
59 int dntimer=30;
60 int mode=1; // hoop mode = 1
61 int mode_old;
62 //
63 // prototyping
64 //
65 #separate
66 void get_adc();
67
68 //
69 // optional analog output with 4 bit DAC
70 //
71 #separate
72 void optional_out()
73 {
74     statusB = 0xE1 & input_B();
75     if(vector_G < 2020) vector_norm = (vector_G * 16)/2025; // normalize
76     else vector_norm=0x0F;
77     vector_norm_i = vector_norm;
78     vector_norm_i = vector_norm_i << 1;
79     statusB = statusB + vector_norm_i;
80     output_B(statusB);

```

```

81  }
82  //
83  // read both ADC
84  //
85  #separate
86  void check_adc() {
87      set_adc_channel(2);          //A2 channel
88      delay_us(50);                //over 31u Require ADx10+15u after changing channel
89      Zaxis_G = read_adc();
90      set_adc_channel(0);          //A0 channel
91      delay_us(50);                //over 31u Require ADx10+15u after changing channel
92      Xaxis_G = read_adc();
93      //
94      Xaxis_G = Xaxis_G - 128; //normalize -128 to 0 to 128
95      Zaxis_G = Zaxis_G - 128; //normalize -128 to 0 to 128
96      //
97  }
98  //
99  // save x, z Gmax data
100 //
101 #separate
102 void save_data()
103 {
104     Gmax_i=10*Gmax; // convert it into integer
105     WRITE_EEPROM(1, Gmax_i);
106 }
107 //
108 // save mode
109 //
110 #separate
111 void save_mode()
112 {
113     WRITE_EEPROM(0, mode);
114 }
115 //
116 // calculate LED location with actual max value
117 //
118 #separate
119 void LED_location(float n) {
120     if(n==0.1) output_B(0x80);
121     if(n==0.5) output_B(0xC0);
122     if(n==1.0) output_B(0xE0);
123 }
124 //
125 // set & display max front G
126 //
127 #separate
128 void set_level() {;
129     disable_interrupts(INT_TIMER1); // disable interrupt timer 1
130     downtimer=300; //wait 3 sec
131     while(downtimer>0) {;
132         if(!input(PIN_A4)) { ; // A4 function sw
133             delay_ms(10);
134             while(!input(PIN_A4)) {};
135             Gmax=Gmax+0.5;
136             if(Gmax==0.6) Gmax=0.5;
137             else if(Gmax>1.0) Gmax=0.1;
138             LED_location(Gmax);
139             downtimer=300; //
140         }
141         delay_ms(10);
142         downtimer--;
143     }
144     output_B(0x00); // all LED off
145     enable_interrupts(INT_TIMER1); // enable interrupt timer 1
146 }
147 //
148 // set & display max front G
149 //
150 #separate
151 void disp_confirm() {;
152     disable_interrupts(INT_TIMER1); // disable interrupt timer 1
153     output_B(0xE0); //turn on all LED
154     delay_ms(500);
155     output_B(0x00); //turn off all LED
156     enable_interrupts(INT_TIMER1); // enable interrupt timer 1
157 }
158 //
159 // setting interrupt timer 1
160 // timer1 counts down for virtual hoop training at every 30m sec.

```

```

161 //
162 #int_timer1
163 void isr1(void) // 8 bit AD converter
164 {
165     set_timer1(0xF000); // set timer1 again, around 32.8ms
166     if(count1>0) count1=count1-1;
167 }
168 //
169 // system intializing
170 //
171 #separate
172 void initializing()
173 {
174     // ADC set them up analog inputs are RA0 RA2
175     setup_adc(ADC_CLOCK_DIV_32);
176     setup_adc_ports(sAN0 | sAN2 | VSS_VDD); //ADC ports are 0 and 2
177     set_tris_a(Amode); //A0 to A2 input, a3 to 5 output
178     set_tris_b(Bmode); //all output
179     output_B(0x00); //all LED off
180     //
181     // set G sensor mode
182     // mode = 1 for hoop mode, 0 for G sensor mode
183     // set mode=0 if pressing function sw when turning on
184     if(input(PIN_A4)==0) // PIN_A4 is function sw
185     {
186         mode=0; // set G sensor mode if pressing function sw when starting up
187         Gmax=0.5; // set default gmax value = 0.5
188         save_mode();
189         disp_confirm();
190         while(input(PIN_A4)==0) {} //halt until releasing sw
191     }
192     else
193     {
194         mode_old = read_EEPROM (0);
195         if(mode_old == 0xFF) mode_old = 1;
196         mode = mode_old;
197     }
198     //
199     // set default virtual hoop mode, and maxG = 1 (default)
200     // mode = 1 for hoop mode, 0 for G sensor mode
201     // set mode=1 if pressing start sw when turning on
202     if(input(PIN_A3)==0) // PIN_A3 is start sw
203     {
204         mode=1; // set virtual hoop mode if pressing start sw when starting up
205         Gmax=0.1; // set default gmax value = 0.1
206         save_mode();
207         save_data();
208         disp_confirm();
209         while(input(PIN_A3)==0) {} //halt until releasing sw
210     }
211     else
212     {
213         Gmax_old = read_EEPROM (1); // convert into float
214         if(Gmax_old == 0xFF) Gmax_old = 0x01;
215         Gmax_old = Gmax_old/10;
216         Gmax = Gmax_old;
217     }
218     //
219     // initializing timer1 interrupt for testing actual temerature by ADC
220     //
221     setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
222     set_timer1(0xF000); // setting timer1 at almost 32m sec
223     enable_interrupts(INT_TIMER1); // enable timer 1 interrupt
224     enable_interrupts(GLOBAL); // enable global interrupt
225     //
226     // PWM control initializing with timer2
227     //
228     setup_ccp1(ccp_pwm); //port B0
229     setup_timer_2(T2_DIV_BY_1,0xFF,1);
230     set_pwm1_duty(0x00);
231 }
232 //
233 // get adc value, get moving average, turn on LEDs and beep
234 //
235 #separate
236 void get_adc()
237 {
238     if(moving_ave<20)
239     { //calculating moving average 5 times then updating vector G
240         check_adc();

```

```

241     Zaxis_G = Zaxis_G-Zoffset;
242     Xaxis_G = Xaxis_G-Xoffset;
243     vector_G temp = Zaxis_G*Zaxis_G+Xaxis_G*Xaxis_G;
244     vector_G_temp = vector_G_temp/Gmax; //normalize into full scale
245     vector_G_sum = vector_G_sum + vector_G_temp;
246     moving_ave++;
247 }
248 else
249 {
250     moving_ave = 0;
251     vector_G = vector_G_sum/20;
252     vector_G_sum=0;
253 }
254 //
255 if(mode==1)
256 { // hoop mode then integrate the value
257     if(vector_G_integ < vector_G)
258     {
259
260         vector_G_integ = vector_G;
261     }
262     else
263     {
264         vector_G = vector_G_integ;
265         vector_G_integ = vector_G_integ - 500;
266     }
267 }
268 //
269 //
270 optional out(); // for optional external analog out
271 if(vector_G>1800)
272 {
273     output_high(green_led); // green LED on
274     output_high(yellow_led); // yello LED on
275     output_high(red_led); // red LED on
276     if(mode==1) set_pwm1_duty(0x00); // beep off
277     else set_pwm1_duty(0x0F); // beep high on
278 }
279 else if(vector_G>800)
280 {
281     output_high(green_led); // green LED on
282     output_high(yellow_led); // yellow LED on
283     output_low(red_led); // red LED off
284     if(mode==1) set_pwm1_duty(0x03); // beep low on
285     else set_pwm1_duty(0x03); // beep low on
286 }
287 else if(vector_G>225)
288 {
289     output_high(green_led); // green LED on
290     output_low(yellow_led); // yellow LED off
291     output_low(red_led); // red LED off
292     if(mode==1) set_pwm1_duty(0x07); // beep middle on
293     else set_pwm1_duty(0x00); // beep off
294 }
295 else
296 {
297     output_low(green_led); // green LED off
298     output_low(yellow_led); // yellow LED off
299     output_low(red_led); // red LED off
300     if(mode==1) set_pwm1_duty(0x0F); // beep high on
301     else set_pwm1_duty(0x00); // beep off
302 }
303 }
304 //
305 // beep one seconds
306 //
307 #separate
308 void beep1000()
309 {
310     set_pwm1_duty(0x0F); // beep high
311     delay_ms(1000);
312     set_pwm1_duty(0x00); // beep off
313 }
314 //
315 // main program
316 //
317 void main()
318 {
319     //
320     // initialzing

```

```

321     initializing();
322     //
323     Xoffset=0;
324     Zoffset=0;
325     //
326     while(1)
327     { //endless loop
328         get_adc();
329         //
330         // set level
331         //
332         if(!input(func_switch)) { //function PIN A4 active low
333             delay_ms(50); //check it again for chattaling
334             if(!input(func_switch)) { ;
335                 if(Prev_f_sw == 1)
336                 {
337                     set_level();
338                     Prev_f_sw = 0;
339                 }
340             }
341         }
342         else Prev_f_sw = 1;
343         //
344         // zero cal
345         //
346         if(!input(zero_switch))
347         { //zero adjust is PIN A5 active low
348             delay_ms(50); //check it again for chattaling
349             if(!input(zero_switch))
350             {
351                 if(Prev_cal_sw == 1)
352                 { //hold until releasing
353                     check_adc();
354                     Xoffset=Xaxis_G; //
355                     Zoffset=Zaxis_G; //
356                     disp_confirm();
357                     Prev_cal_sw = 0;
358                 }
359             }
360         }
361         else Prev_cal_sw = 1;
362         //
363         // start
364         //
365         if(!input(start_switch))
366         { //start PIN A3 active low
367             delay_ms(50); //check it again for chattaling
368             if(!input(start_switch))
369             {
370                 if(Prev_s_sw == 1)
371                 //disable interrupts(INT_TIMER1); // disable timer 1
372                 if(mode != mode_old) save_mode();
373                 if(Gmax != Gmax_old) save_data();
374                 if(mode==1) // only for virtual hoop mode
375                 {
376                     disable_interrupts(INT_TIMER1); // disable timer 1
377                     count1=5400; //around 3 minutes
378                     enable_interrupts(INT_TIMER1); // enable timer 1
379                     while(count1>0)
380                     {
381                         get_adc();
382                         //
383                         Prev_s_sw = 0;
384                     }
385                     beep1000(); // beep 1 seconds
386                     delay_ms(300);
387                     beep1000(); // beep 1 seconds
388                     delay_ms(300);
389                     //
390                 }
391                 else Prev_s_sw = 0;
392             }
393         }
394         else Prev_s_sw = 1;
395     }
396 }
397

```