```
/*
 Name:      TeensyQuadcopter.ino
 Created:   5/3/2017 9:52:41 PM
 Author:    Michael Langford
*/

/*

prop1       prop2
|           |
[][] front [][]
[][]   |   [][]
    \  |  /
    {[[[]]}-----microcontroller
    {[[[]]}
    /     \
[][]       [][]
[][]       [][]
|           |
prop3      prop4

*/

#include "WProgram.h"
#include "LIS3DH.h"
#include "L3G4200D.h"
#include "RadioReciever.h"
#include "PID.h"
#include "Motors.h"
#include "Kalman.h"
#include "Debug.h"

#define THROTTLE_THRESHOLD 1.5f

#define THROTTLE_UNLOCK   -90.0f
#define YAW_UNLOCK        90.0f
#define PITCH_UNLOCK      -90.0f
#define ROLL_UNLOCK       -90.0f
#define GEAR_SET          90.0f

#define SAFE              0
#define ARMED             1

//PID loops
PIDClass YawPID;
PIDClass PitchPID;
PIDClass RollPID;

//PID outputs
float yaw_out, pitch_out, roll_out, throttle_out;

//motor outputs
```

```
53  float motor_output[4];
54
55  //debug values
56  int loop_time = 0;
57  int l_loop_time = 0;
58  int count = 0;
59
60  //SAFEing value
61  int safety = SAFE;
62
63  /*//Kalman filters
64  Kalman pitch_kalman;
65  Kalman roll_kalman;*/
66
67  //main angles
68  float yaw_angle = 0.0f;
69  float main_pitch = 0.0f;
70  float main_roll = 0.0f;
71
72  // the setup function runs once when you press reset or power the board
73  void setup() {
74      Serial.begin(115200);
75      Serial.clear();
76
77      //init debug helper
78      init_Debug();
79
80      //delay(1000);
81
82      init_LIS3DH();
83      init_L3G4200D();
84      init_RadioReciever();
85      init_motors();
86
87      //init pid loops
88
89      YawPID.init_PID();
90      PitchPID.init_PID();
91      RollPID.init_PID();
92
93      YawPID.Set_Constants  (0.3f, 0.0f, 900.0f);
94      PitchPID.Set_Constants(0.79f, 0.0f, 950.0f);
95      RollPID.Set_Constants (0.79f, 0.0f, 950.0f);
96
97      /*//init kalman filters
98
99      pitch_kalman.init_Kalman();
100     roll_kalman.init_Kalman();*/
101 }
102
103 int angle_reset_count = 0;
104 int resets = 0;
```

```
105
106 void loop() {
107
108     if (safety == SAFE)
109     {
110         //update debug helper
111         update_Debug_SAFE();
112
113         //zero motors
114         update_motors(0.0f, 0.0f, 0.0f, 0.0f);
115
116         //check ARM
117         arm();
118     }
119     if (safety == ARMED)
120     {
121         loop_time = micros();
122
123         //update debug helper
124         update_Debug_ARMED();
125
126         //update sensors and radio
127         update_LIS3DH();
128         update_L3G4200D();
129         update_RadioReciever();
130
131         /*//Kalman Filter
132         main_pitch = pitch_kalman.Filter(Get_Acc_Pitch(), GetPitchRate() *
              GetGyroElapsedTime());
133         main_roll = roll_kalman.Filter(Get_Acc_Roll(), GetRollRate() *
              GetGyroElapsedTime());
134         */
135
136         /*
137         main_pitch = ((double)(main_pitch + GetPitchRate()*GetGyroElapsedTime())
              * 0.999) + (double)Get_Acc_Pitch()*0.001;
138         main_roll = ((double)(main_roll + GetRollRate()*GetGyroElapsedTime()) *
              0.999) + (double)Get_Acc_Roll() * 0.001;
139         */
140
141
142         main_pitch = GetPitch();// (main_pitch + GetPitchRate()*egyrotime);//
              +Get_Acc_Pitch()*0.000f;
143         main_roll = GetRoll();// (main_roll + GetRollRate()*egyrotime);//
              +Get_Acc_Roll()*0.0001f;
144
145
146         /*angle_reset_count++;
147         if (angle_reset_count > 30)
148         {
149             angle_reset_count = 0;
150             if (abs(GetPitchRate()) < 3.7f && abs(GetRollRate()) < 3.7f)
```

```
151             {
152                 if (abs(Get_Acc_Pitch()) < 10.0f && abs(Get_Acc_Roll()) < 10.0f)
153                 {
154                     resets++;
155                     main_pitch = Get_Acc_Pitch();
156                     main_roll = Get_Acc_Roll();
157                     ClearAngles(GetYaw(), Get_Acc_Pitch(), Get_Acc_Roll());
158                     debug_flash();
159                 }
160             }
161         }*/
162 
163         if (abs(GetPitchRate()*GetGyroElapsedTime()) < 10.0f && abs(GetRollRate()
              *GetGyroElapsedTime()) < 10.0f)
164         {
165             if (abs(Get_Acc_Pitch()) < 4.0f && abs(Get_Acc_Roll()) < 4.0f)
166             {
167                 angle_reset_count++;
168             }
169             else
170             {
171                 angle_reset_count = 0;
172             }
173         }
174         else
175         {
176             angle_reset_count = 0;
177         }
178 
179         if (angle_reset_count > 16)
180         {
181             angle_reset_count = 0;
182             resets++;
183             main_pitch = Get_Acc_Pitch();
184             main_roll = Get_Acc_Roll();
185             ClearAngles(GetYaw(), Get_Acc_Pitch(), Get_Acc_Roll());
186             debug_flash();
187         }
188 
189         //update PID loops
190         yaw_angle += GetChannel(RUD_YAW) / 1500.0f;
191         yaw_out = YawPID.update_PID(GetYaw(), yaw_angle);
192         pitch_out = PitchPID.update_PID(main_pitch, GetChannel(ELEV_PITCH) /
              3.0f);
193         roll_out = RollPID.update_PID(main_roll, GetChannel(AIL_ROLL) / 3.0f);
194 
195         //update motor values
196 
197         throttle_out = GetChannel(THROTTLE);
198         throttle_out = (throttle_out + 100.0f) / 2.0f;
199 
200         //check safety value
```

```
201
202         if (GetChannel(GEAR) < GEAR_SET && throttle_out < THROTTLE_THRESHOLD)
203         {
204             safety = SAFE;
205             return;
206         }
207
208         motor_output[0] = throttle_out;
209         motor_output[1] = throttle_out;
210         motor_output[2] = throttle_out;
211         motor_output[3] = throttle_out;
212
213         motor_output[0] += yaw_out;
214         motor_output[1] += -yaw_out;
215         motor_output[2] += -yaw_out;
216         motor_output[3] += yaw_out;
217
218         motor_output[0] += -pitch_out;
219         motor_output[1] += -pitch_out;
220         motor_output[2] += pitch_out;
221         motor_output[3] += pitch_out;
222
223         motor_output[0] += -roll_out;
224         motor_output[1] += roll_out;
225         motor_output[2] += -roll_out;
226         motor_output[3] += roll_out;
227
228         //clamp motor values
229
230         motor_output[0] = max(0.0f, motor_output[0]);
231         motor_output[1] = max(0.0f, motor_output[1]);
232         motor_output[2] = max(0.0f, motor_output[2]);
233         motor_output[3] = max(0.0f, motor_output[3]);
234
235         motor_output[0] = min(100.0f, motor_output[0]);
236         motor_output[1] = min(100.0f, motor_output[1]);
237         motor_output[2] = min(100.0f, motor_output[2]);
238         motor_output[3] = min(100.0f, motor_output[3]);
239
240         if (throttle_out < THROTTLE_THRESHOLD)
241         {
242             motor_output[0] = 0.0f;
243             motor_output[1] = 0.0f;
244             motor_output[2] = 0.0f;
245             motor_output[3] = 0.0f;
246         }
247
248         //send motor values to driver
249         update_motors(motor_output[0], motor_output[1], motor_output[2],
              motor_output[3]);
250         //update_motors(0.0f, 0.0f, 0.0f, 0.0f);
251
```

```
252         //////                        ///////
253         //////       display          ///////
254         //////                        ///////
255         count++;
256         if (count > 40)
257         {
258             count = 0;
259             //Serial.printf("%f, %f, %f, %f", motor_output[0], motor_output[1],
                  motor_output[2], motor_output[3]);
260             //Serial.printf(", %f, %f, %f", GetYaw(), GetPitch(), GetRoll());
261             //Serial.printf(", %f, %f, %f, %f, %f\n", (loop_time - l_loop_time) /
                   1000000.0f, GetChannel(THROTTLE), GetChannel(RUD_YAW), GetChannel
                  (AIL_ROLL), GetChannel(ELEV_PITCH));
262
263             //Serial.printf("%f, %f, %f, 00, %f, %f, %f, 00, %d, %d, %d\n",
                  GetYaw(), GetPitch(), GetRoll(), get_cal_x(), get_cal_y(),
                  get_cal_z(), getX(), getY(), getZ());
264             //Serial.printf("%f, %f, ", Get_Acc_Pitch(), Get_Acc_Roll());
265             Serial.printf("%f, %f, %f, %f, ", GetChannel(THROTTLE), GetChannel
                  (RUD_YAW), GetChannel(AIL_ROLL), GetChannel(ELEV_PITCH));
266             Serial.printf("%f, %f, %d\n", GetRoll(), GetPitch(), resets);
267         }
268         l_loop_time = loop_time;
269     }
270 }
271
272 void arm()
273 {
274     update_RadioReciever();
275     if (GetChannel(THROTTLE) < THROTTLE_UNLOCK)
276     {
277         if (GetChannel(RUD_YAW) > YAW_UNLOCK)
278         {
279             if (GetChannel(AIL_ROLL) < ROLL_UNLOCK)
280             {
281                 if (GetChannel(ELEV_PITCH) < PITCH_UNLOCK)
282                 {
283                     if (GetChannel(GEAR) > GEAR_SET)
284                     {
285                         Calibrate_Accelerometer();
286
287                         ClearAngles(0.0f, 0.0f, 0.0f);
288
289                         Calibrate_Gyro();
290
291                         //delay(100);
292                         yaw_angle = 0.0f;
293
294                         main_pitch = 0.0f;
295                         main_roll = 0.0f;
296
297                         /*pitch_kalman.init_Kalman();
```

```
298                         roll_kalman.init_Kalman();*/
299
300                         ClearAngles(0.0f, 0.0f, 0.0f);
301                         update_L3G4200D();
302                         ClearAngles(0.0f, 0.0f, 0.0f);
303
304                         safety = ARMED;
305                     }
306                 }
307             }
308         }
309     }
310 }
311
```