```
#include "flight_control.h"
#include "rc.h"
#include <math.h>

float pid_x_integ1 = 0;
float pid_y_integ1 = 0;
float pid_z_integ1 = 0;
float pid_x_integ2 = 0;
float pid_y_integ2 = 0;
float pid_z_integ2 = 0;
float pid_x_pre_error2 = 0;
float pid_y_pre_error2 = 0;
float pid_z_pre_error2 = 0;
float pid_x_pre_deriv = 0;
float pid_y_pre_deriv = 0;

extern int16_t gTHR;
int16_t motor_thr;
float dt_recip;

void PIDControlInit(P_PI_PIDControlTypeDef *pid)
{
  pid->ts = PID_SAMPLING_TIME;

  pid->x_kp1 = PITCH_PID_KP1;
  pid->x_ki1 = PITCH_PID_KI1;
  pid->x_i1_limit = PITCH_PID_I1_LIMIT;
  pid->x_kp2 = PITCH_PID_KP2;
  pid->x_ki2 = PITCH_PID_KI2;
  pid->x_kd2 = PITCH_PID_KD2;
  pid->x_i2_limit = PITCH_PID_I2_LIMIT;
  pid->x_s1 = 0;
  pid->x_s2 = 0;

  pid->y_kp1 = ROLL_PID_KP1;
  pid->y_ki1 = ROLL_PID_KI1;
  pid->y_i1_limit = ROLL_PID_I1_LIMIT;
  pid->y_kp2 = ROLL_PID_KP2;
  pid->y_ki2 = ROLL_PID_KI2;
  pid->y_kd2 = ROLL_PID_KD2;
  pid->y_i2_limit = ROLL_PID_I2_LIMIT;
  pid->y_s1 = 0;
  pid->y_s2 = 0;

  pid->z_kp1 = YAW_PID_KP1;
  pid->z_ki1 = YAW_PID_KI1;
  pid->z_i1_limit = YAW_PID_I1_LIMIT;
  pid->z_kp2 = YAW_PID_KP2;
  pid->z_ki2 = YAW_PID_KI2;
  pid->z_kd2 = YAW_PID_KD2;
```

```
  pid->z_i2_limit = YAW_PID_I2_LIMIT;
  pid->z_s1 = 0;
  pid->z_s2 = 0;
}

void FlightControlPID(EulerAngleTypeDef *euler_rc, EulerAngleTypeDef *euler_ahrs, Gyro_Rad *gyro_rad, AHRS_State_TypeDef *ahrs, P_PI_PIDControlTypeDef *pid, MotorControlTypeDef *motor_pwm)
{
  float error, deriv;

  if(gTHR<MIN_THR)
  {
    pid_x_integ1 = 0;
    pid_y_integ1 = 0;
    pid_z_integ1 = 0;
    pid_x_integ2 = 0;
    pid_y_integ2 = 0;
    pid_z_integ2 = 0;
  }


  //x-axis pid
  error = euler_rc->thx - euler_ahrs->thx;
  pid_x_integ1 += error*pid->ts;
  if(pid_x_integ1 > pid->x_i1_limit)
    pid_x_integ1 = pid->x_i1_limit;
  else if(pid_x_integ1 < -pid->x_i1_limit)
    pid_x_integ1 = -pid->x_i1_limit;
  pid->x_s1 =  pid->x_kp1*error + pid->x_ki1*pid_x_integ1;

  error = euler_rc->thx - gyro_rad->gx;
  pid_x_integ2 += error*pid->ts;
  if(pid_x_integ2 > pid->x_i2_limit)
    pid_x_integ2 = pid->x_i2_limit;
  else if(pid_x_integ2 < -pid->x_i2_limit)
    pid_x_integ2 = -pid->x_i2_limit;
  deriv = error - pid_x_pre_error2;
  pid_x_pre_error2 = error;
  pid->x_s2 = pid->x_kp2*error + pid->x_ki2*pid_x_integ2 + pid->x_kd2*deriv;

  if(pid->x_s2 > MAX_ADJ_AMOUNT)  pid->x_s2 = MAX_ADJ_AMOUNT;
  if(pid->x_s2 < -MAX_ADJ_AMOUNT)  pid->x_s2 = -MAX_ADJ_AMOUNT;


  //y-axis pid
  error = euler_rc->thy - euler_ahrs->thy;
  pid_y_integ1 += error*pid->ts;
  if(pid_y_integ1 > pid->y_i1_limit)
    pid_y_integ1 = pid->y_i1_limit;
```

この関数は使われない.
代わりに, FlightControlPID_OuterLoop() と FlightControlPID_innerLoop()が使われる

```
 99  else if(pid_y_integ1 < -pid->y_i1_limit)
100    pid_y_integ1 = -pid->y_i1_limit;
101  pid->y_s1 =  pid->y_kp1*error + pid->y_ki1*pid_y_integ1;
102
103  error = euler_rc->thy - gyro_rad->gy;
104  pid_y_integ2 += error*pid->ts;
105  if(pid_y_integ2 > pid->y_i2_limit)
106    pid_y_integ2 = pid->y_i2_limit;
107  else if(pid_y_integ2 < -pid->y_i2_limit)
108    pid_y_integ2 = -pid->y_i2_limit;
109  deriv = error - pid_y_pre_error2;
110  pid_y_pre_error2 = error;
111  pid->y_s2 = pid->y_kp2*error + pid->y_ki2*pid_y_integ2 + pid->y_kd2*deriv;
112
113  if(pid->y_s2 > MAX_ADJ_AMOUNT)  pid->y_s2 = MAX_ADJ_AMOUNT;
114  if(pid->y_s2 < -MAX_ADJ_AMOUNT)  pid->y_s2 = -MAX_ADJ_AMOUNT;
115
116
117  //z-axis pid
118  error = euler_rc->thz - gyro_rad->gz;
119  pid_z_integ2 += error*pid->ts;
120  if(pid_z_integ2 > pid->z_i2_limit)
121    pid_z_integ2 = pid->z_i2_limit;
122  else if(pid_z_integ2 < -pid->z_i2_limit)
123    pid_z_integ2 = -pid->z_i2_limit;
124  deriv = error - pid_z_pre_error2;
125  pid_z_pre_error2 = error;
126  pid->z_s2 = pid->z_kp2*error + pid->z_ki2*pid_y_integ2 + pid->z_kd2*deriv;
127
128  if(pid->z_s2 > MAX_ADJ_AMOUNT)  pid->z_s2 = MAX_ADJ_AMOUNT;
129  if(pid->z_s2 < -MAX_ADJ_AMOUNT)  pid->z_s2 = -MAX_ADJ_AMOUNT;
130
131  #ifdef MOTOR_DC
132
133    motor_thr = 0.33333f*gTHR + 633.333f;           //Devo7E >> 630 to 1700
134
135  #endif
136
137  #ifdef MOTOR_ESC
138
139    //motor_thr = 0.28f*gTHR + 750.0f;                  //TGY-i6 remocon and external ESC
    STEVAL-ESC001V1
140    //motor_thr = 0.28f*gTHR + 850.0f;                  //TGY-i6 remocon and external ESC
    Afro12A
141    motor_thr = 0.32f*gTHR + 900.0f;                  //TGY-i6 remocon and external ESC
    Afro12A
142
143  #endif
144
145
```

```
146   motor_pwm->motor1_pwm = motor_thr - pid->x_s2 - pid->y_s2 + pid->z_s2 + MOTOR_OFF1;
147   motor_pwm->motor2_pwm = motor_thr + pid->x_s2 - pid->y_s2 - pid->z_s2 + MOTOR_OFF2;
148   motor_pwm->motor3_pwm = motor_thr + pid->x_s2 + pid->y_s2 + pid->z_s2 + MOTOR_OFF3;
149   motor_pwm->motor4_pwm = motor_thr - pid->x_s2 + pid->y_s2 - pid->z_s2 + MOTOR_OFF4;
150
151
152 }
153
154 void     FlightControlPID_OuterLoop(EulerAngleTypeDef     *euler_rc,     EulerAngleTypeDef
    *euler_ahrs, AHRS_State_TypeDef *ahrs, P_PI_PIDControlTypeDef *pid)
155 {
156   float error;
157
158   if(gTHR<MIN_THR)
159   {
160     pid_x_integ1 = 0;
161     pid_y_integ1 = 0;
162     pid_z_integ1 = 0;
163   }
164
165   //x-axis pid
166   error = euler_rc->thx - euler_ahrs->thx;
167   pid_x_integ1 += error*pid->ts;
168   if(pid_x_integ1 > pid->x_i1_limit)
169     pid_x_integ1 = pid->x_i1_limit;
170   else if(pid_x_integ1 < -pid->x_i1_limit)
171     pid_x_integ1 = -pid->x_i1_limit;
172   pid->x_s1 =  pid->x_kp1*error + pid->x_ki1*pid_x_integ1;
173
174   //y-axis pid
175   error = euler_rc->thy - euler_ahrs->thy;
176   pid_y_integ1 += error*pid->ts;
177   if(pid_y_integ1 > pid->y_i1_limit)
178     pid_y_integ1 = pid->y_i1_limit;
179   else if(pid_y_integ1 < -pid->y_i1_limit)
180     pid_y_integ1 = -pid->y_i1_limit;
181   pid->y_s1 =  pid->y_kp1*error + pid->y_ki1*pid_y_integ1;
182
183   //z-axis pid
184   error = euler_rc->thz - euler_ahrs->thz;
185   pid_z_integ1 += error*pid->ts;
186   if(pid_z_integ1 > pid->z_i1_limit)
187     pid_z_integ1 = pid->z_i1_limit;
188   else if(pid_z_integ1 < -pid->z_i1_limit)
189     pid_z_integ1 = -pid->z_i1_limit;
190   pid->z_s1 =  pid->z_kp1*error + pid->z_ki1*pid_z_integ1;
191 }
192
193 void    FlightControlPID_innerLoop(EulerAngleTypeDef    *euler_rc,    Gyro_Rad    *gyro_rad,
    AHRS_State_TypeDef *ahrs, P_PI_PIDControlTypeDef *pid, MotorControlTypeDef *motor_pwm)
```

アウタ・ループ姿勢角度制御

スロットルを下げているとき（着陸中など）は追従偏差の積分値をリセットする

166～172 行目はピッチ姿勢角の制御

追従偏差を求める

追従偏差の積分を求める

追従偏差の積分値が過大にならないようリミッタをかけて飽和させる

PI 制御により角速度目標値を求める

比例(P)制御

積分(I)制御

ロール姿勢角の制御

ヨー角の制御

インナ・ループ角速度制御

```
194 {
195   float error, deriv;
196
197   if(gTHR<MIN_THR)
198   {
199     pid_x_integ2 = 0;
200     pid_y_integ2 = 0;
201     pid_z_integ2 = 0;
202   }
203
204   dt_recip = 1/pid->ts;
205
206   //X Axis
207   error = pid->x_s1 - gyro_rad->gx;
208   pid_x_integ2 += error*pid->ts;
209   if(pid_x_integ2 > pid->x_i2_limit)
210     pid_x_integ2 = pid->x_i2_limit;
211   else if(pid_x_integ2 < -pid->x_i2_limit)
212     pid_x_integ2 = -pid->x_i2_limit;
213   deriv = (error - pid_x_pre_error2)*dt_recip;
214   pid_x_pre_error2 = error;
215   deriv = pid_x_pre_deriv + (deriv - pid_x_pre_deriv)*D_FILTER_COFF;
216   pid_x_pre_deriv = deriv;
217   pid->x_s2 = pid->x_kp2*error + pid->x_ki2*pid_x_integ2 + pid->x_kd2*deriv;
218
219   if(pid->x_s2 > MAX_ADJ_AMOUNT)  pid->x_s2 = MAX_ADJ_AMOUNT;
220   if(pid->x_s2 < -MAX_ADJ_AMOUNT)  pid->x_s2 = -MAX_ADJ_AMOUNT;
221
222   //Y Axis
223   error = pid->y_s1 - gyro_rad->gy;
224   pid_y_integ2 += error*pid->ts;
225   if(pid_y_integ2 > pid->y_i2_limit)
226     pid_y_integ2 = pid->y_i2_limit;
227   else if(pid_y_integ2 < -pid->y_i2_limit)
228     pid_y_integ2 = -pid->y_i2_limit;
229   deriv = (error - pid_y_pre_error2)*dt_recip;
230   pid_y_pre_error2 = error;
231   deriv = pid_y_pre_deriv + (deriv - pid_y_pre_deriv)*D_FILTER_COFF;
232   pid_y_pre_deriv = deriv;
233   pid->y_s2 = pid->y_kp2*error + pid->y_ki2*pid_y_integ2 + pid->y_kd2*deriv;
234
235   if(pid->y_s2 > MAX_ADJ_AMOUNT)  pid->y_s2 = MAX_ADJ_AMOUNT;
236   if(pid->y_s2 < -MAX_ADJ_AMOUNT)  pid->y_s2 = -MAX_ADJ_AMOUNT;
237
238   //Z Axis
239   error = pid->z_s1 - gyro_rad->gz;
240   pid_z_integ2 += error*pid->ts;
241   if(pid_z_integ2 > pid->z_i2_limit)
242     pid_z_integ2 = pid->z_i2_limit;
243   else if(pid_z_integ2 < -pid->z_i2_limit)
```

スロットルを下げているとき（着陸中など）は追従偏差の積分値をリセットする

206～220 行目は X 軸（ピッチ）角速度の制御

追従偏差を求める

追従偏差の積分を求める

追従偏差の積分値が過大にならないようリミッタをかけて飽和させる

追従偏差の微分を求める

追従偏差の微分値をノイズ・フィルタに通す

PID 制御の計算を行う

制御出力が過大にならないようリミッタをかけて飽和させる

比例(P)制御

積分(I)制御

微分(D)制御

Y 軸（ロール）角速度の制御

Z 軸（ヨー）角速度の制御（～250 行目）

```
244     pid_z_integ2 = -pid->z_i2_limit;
245   deriv = (error - pid_z_pre_error2)*dt_recip;
246   pid_z_pre_error2 = error;
247   pid->z_s2 = pid->z_kp2*error + pid->z_ki2*pid_z_integ2 + pid->z_kd2*deriv;
248
249   if(pid->z_s2 > MAX_ADJ_AMOUNT_YAW)  pid->z_s2 = MAX_ADJ_AMOUNT_YAW;
250   if(pid->z_s2 < -MAX_ADJ_AMOUNT_YAW)  pid->z_s2 = -MAX_ADJ_AMOUNT_YAW;
251
252
253#ifdef MOTOR_DC
254
255   motor_thr = 0.33333f*gTHR + 633.333f;           //Remocon Devo7E >> 630 to 1700
256
257#endif
258
259#ifdef MOTOR_ESC
260
261   //motor_thr = 0.28f*gTHR + 750.0f;                    //TGY-i6 remocon and external ESC
    STEVAL-ESC001V1
262   //motor_thr = 0.28f*gTHR + 850.0f;                    //TGY-i6 remocon and external ESC
    Afro12A
263   motor_thr = 0.32f*gTHR + 900.0f;                    //TGY-i6 remocon and external ESC
    Afro12A
264
265
266#endif
267
268   motor_pwm->motor1_pwm = motor_thr - pid->x_s2 - pid->y_s2 + pid->z_s2 + MOTOR_OFF1;
269   motor_pwm->motor2_pwm = motor_thr + pid->x_s2 - pid->y_s2 - pid->z_s2 + MOTOR_OFF2;
270   motor_pwm->motor3_pwm = motor_thr + pid->x_s2 + pid->y_s2 + pid->z_s2 + MOTOR_OFF3;
271   motor_pwm->motor4_pwm = motor_thr - pid->x_s2 + pid->y_s2 - pid->z_s2 + MOTOR_OFF4;
272
273}
274
275void PIDOuterLoopFrameTrans(P_PI_PIDControlTypeDef *pid, EulerAngleTypeDef *euler_ahrs)
276{
277   float cosx;
278
279   cosx = cos(euler_ahrs->thx);
280   pid->y_s1 = cosx*pid->y_s1;
281
282}
```

推力指令値を求める

推力指令値および角速度制御器の出力をコマンド分配則に通して，4 つのモータの PWM 値へ変換する