#include "CONTROL.h"

#include "IMU1.h"

#include "moto.h"

#include "RFdate.h"

#include <math.h>

extern T\_RC\_Data Rc\_D; //遥控通道数据;

extern u8 txbuf[4]; //发送缓冲

extern u8 rxbuf[4]; //接收缓冲

extern u16 test1[3]; //接收到NRf24L01数据

extern S\_INT16\_XYZ ACC\_F,GYRO\_F;

PID PID\_ROL,PID\_PIT,PID\_YAW;

extern S\_INT16\_XYZ MPU6050\_ACC\_LAST,MPU6050\_GYRO\_LAST;

int Motor\_Ele=0; //俯仰期望

int Motor\_Ail=0; //横滚期望

//u8 ARMED = 0;

//float rol\_i=0,pit\_i=0,yaw\_p=0;

float thr=0;

S\_FLOAT\_XYZ EXP\_ANGLE ,DIF\_ANGLE;

PID1 PID\_Motor;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float Pitch\_i,Roll\_i,Yaw\_i; //积分项

float Pitch\_old,Roll\_old,Yaw\_old; //角度保存

float Pitch\_d,Roll\_d,Yaw\_d; //微分项

float RC\_Pitch,RC\_Roll,RC\_Yaw; //姿态角

float Pitch\_shell\_out,Roll\_shell\_out,Yaw\_shell\_out;//外环总输出

//外环PID参数

float Pitch\_shell\_kp=280;//30 140

float Pitch\_shell\_kd=0;//

float Pitch\_shell\_ki=0;//

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float Roll\_shell\_kp=250;//30

float Roll\_shell\_kd=0;//10

float Roll\_shell\_ki=0;//0.08

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float Yaw\_shell\_kp=1.5;//10;//30

float Yaw\_shell\_kd=0;//10

float Yaw\_shell\_ki=0;//0.08;//0.08

float Gyro\_radian\_old\_x,Gyro\_radian\_old\_y,Gyro\_radian\_old\_z;//陀螺仪保存

float pitch\_core\_kp\_out,pitch\_core\_kd\_out,Roll\_core\_kp\_out,Roll\_core\_kd\_out,Yaw\_core\_kp\_out,Yaw\_core\_kd\_out;//内环单项输出

float Pitch\_core\_out,Roll\_core\_out,Yaw\_core\_out;//内环总输出

//内环PID参数

//float Pitch\_core\_kp=0.040;

//float Pitch\_core\_kd=0.008;////0.007;//0.07;

float Pitch\_core\_kp=0.040;

float Pitch\_core\_kd=0.002;////0.007;//0.07;

float Roll\_core\_kp=0.040;//;

float Roll\_core\_kd=0.002;////0.007;//06;//0.07;

float Yaw\_core\_kp=0.046;//;

float Yaw\_core\_kd=0.012;////0.007;//06;//0.07;

int16\_t moto1=0,moto2=0,moto3=0,moto4=0;

float tempjd=0;

void CONTROL(float rol, float pit, float yaw)

{

RC\_Pitch=(Rc\_D.PITCH-1500)/20;

////////////////////////外环角度环(PID)///////////////////////////////

Pitch\_i+=(Q\_ANGLE.Pitch-RC\_Pitch);

//-------------Pitch积分限幅----------------//

if(Pitch\_i>300) Pitch\_i=300;

else if(Pitch\_i<-300) Pitch\_i=-300;

//-------------Pitch微分--------------------//

Pitch\_d=Q\_ANGLE.Pitch-Pitch\_old;

//-------------Pitch PID-------------------//

Pitch\_shell\_out = Pitch\_shell\_kp\*(Q\_ANGLE.Pitch-RC\_Pitch) + Pitch\_shell\_ki\*Pitch\_i + Pitch\_shell\_kd\*Pitch\_d;

//角度保存

Pitch\_old=Q\_ANGLE.Pitch;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

RC\_Roll=(Rc\_D.ROLL-1500)/20;

Roll\_i+=(Q\_ANGLE.Rool-RC\_Roll);

//-------------Roll积分限幅----------------//

if(Roll\_i>300) Roll\_i=300;

else if(Roll\_i<-300) Roll\_i=-300;

//-------------Roll微分--------------------//

Roll\_d=Q\_ANGLE.Rool-Roll\_old;

//-------------Roll PID-------------------//

Roll\_shell\_out = Roll\_shell\_kp\*(Q\_ANGLE.Rool-RC\_Roll) + Roll\_shell\_ki\*Roll\_i + Roll\_shell\_kd\*Roll\_d;

//------------Roll角度保存------------------//

Roll\_old=Q\_ANGLE.Rool;

RC\_Yaw=(Rc\_D.YAW-1500)\*10;

//-------------Yaw微分--------------------//

Yaw\_d=MPU6050\_GYRO\_LAST.Z-Yaw\_old;

//-------------Roll PID-------------------//

Yaw\_shell\_out = Yaw\_shell\_kp\*(MPU6050\_GYRO\_LAST.Z-RC\_Yaw) + Yaw\_shell\_ki\*Yaw\_i + Yaw\_shell\_kd\*Yaw\_d;

//------------Roll角度保存------------------//

Yaw\_old=MPU6050\_GYRO\_LAST.Z;

////////////////////////内环角速度环(PD)///////////////////////////////

pitch\_core\_kp\_out = Pitch\_core\_kp \* (Pitch\_shell\_out + MPU6050\_GYRO\_LAST.Y \* 3.5);

pitch\_core\_kd\_out = Pitch\_core\_kd \* (MPU6050\_GYRO\_LAST.Y - Gyro\_radian\_old\_y);

Roll\_core\_kp\_out = Roll\_core\_kp \* (Roll\_shell\_out + MPU6050\_GYRO\_LAST.X \*3.5);

Roll\_core\_kd\_out = Roll\_core\_kd \* (MPU6050\_GYRO\_LAST.X - Gyro\_radian\_old\_x);

Yaw\_core\_kp\_out = Yaw\_core\_kp \* (Yaw\_shell\_out + MPU6050\_GYRO\_LAST.Z \* 1);

Yaw\_core\_kd\_out = Yaw\_core\_kd \* (MPU6050\_GYRO\_LAST.Z - Gyro\_radian\_old\_z);

Pitch\_core\_out = pitch\_core\_kp\_out + pitch\_core\_kd\_out;

Roll\_core\_out = Roll\_core\_kp\_out + Roll\_core\_kd\_out;

Yaw\_core\_out = Yaw\_core\_kp\_out + Yaw\_core\_kd\_out;

Gyro\_radian\_old\_y = MPU6050\_GYRO\_LAST.X;

Gyro\_radian\_old\_x = MPU6050\_GYRO\_LAST.Y;

Gyro\_radian\_old\_z = MPU6050\_GYRO\_LAST.Z; //储存历史值

//--------------------将输出值融合到四个电机--------------------------------//

if(Rc\_D.THROTTLE>1020)

{

thr=Rc\_D.THROTTLE- 1000;

// if(Rc\_D.THROTTLE<=2000)

// {

// moto1=(int16\_t)(thr - Pitch\_core\_out);//- yaw);

// moto2=(int16\_t)(thr - Pitch\_core\_out);//+ yaw);

// moto3=(int16\_t)(thr + Pitch\_core\_out);// - yaw);

// moto4=(int16\_t)(thr + Pitch\_core\_out);//+ yaw);

// moto1=(int16\_t)(thr - Roll\_core\_out);//- yaw);

// moto2=(int16\_t)(thr + Roll\_core\_out);//+ yaw);

// moto3=(int16\_t)(thr + Roll\_core\_out);// - yaw);

// moto4=(int16\_t)(thr - Roll\_core\_out);//+ yaw);

// moto1=(int16\_t)(thr - Yaw\_core\_out);//- yaw);

// moto2=(int16\_t)(thr + Yaw\_core\_out);//+ yaw);

// moto3=(int16\_t)(thr - Yaw\_core\_out);// - yaw);

// moto4=(int16\_t)(thr + Yaw\_core\_out);//+ yaw);

//moto1=(int16\_t)(thr - Roll\_core\_out - Pitch\_core\_out);

//moto2=(int16\_t)(thr + Roll\_core\_out - Pitch\_core\_out);

//moto3=(int16\_t)(thr + Roll\_core\_out + Pitch\_core\_out);

//moto4=(int16\_t)(thr - Roll\_core\_out + Pitch\_core\_out);

//

moto1=(int16\_t)(thr - Roll\_core\_out - Pitch\_core\_out- Yaw\_core\_out);

moto2=(int16\_t)(thr + Roll\_core\_out - Pitch\_core\_out+ Yaw\_core\_out);

moto3=(int16\_t)(thr + Roll\_core\_out + Pitch\_core\_out- Yaw\_core\_out);

moto4=(int16\_t)(thr - Roll\_core\_out + Pitch\_core\_out+ Yaw\_core\_out);

// }

}

else

{

moto1 = 0;

moto2 = 0;

moto3 = 0;

moto4 = 0;

}

MOTO\_PWMRFLASH(moto1,moto2,moto3,moto4);// Moto\_PwmRflash(moto1,moto2,moto3,moto4);

}