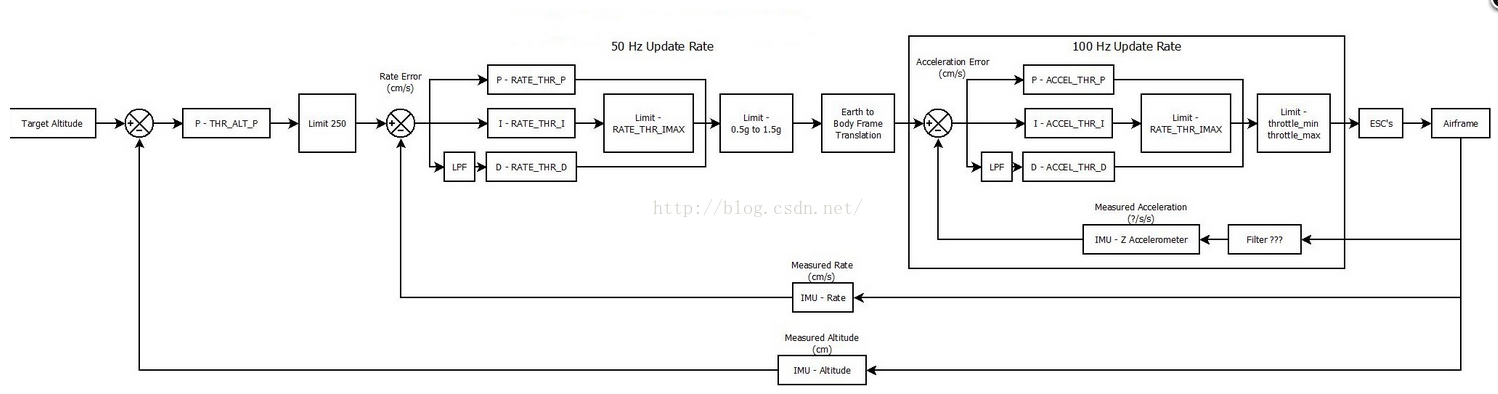
## 涉及到数学知识

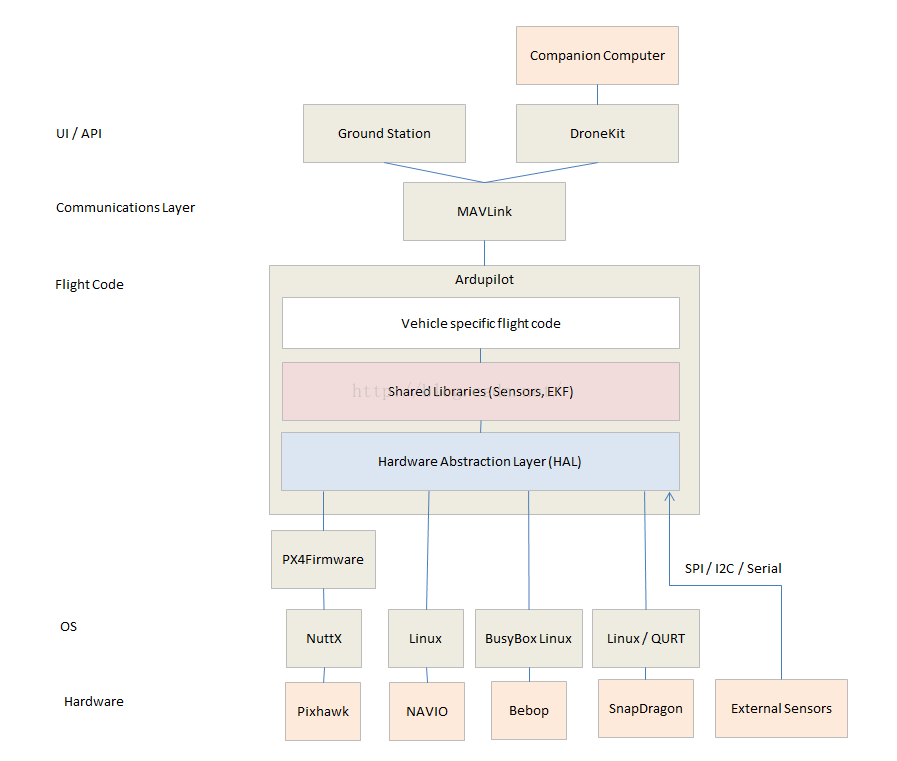
数据融合、姿态结算、卡尔曼滤波、PID控制、模糊控制。

## 定高三级反馈控制原理图，

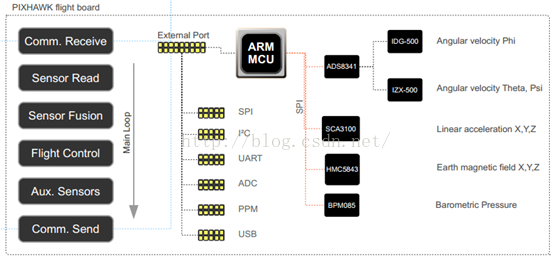
定高三级反馈控制原理图，很多都是基于这种方式进行控制的

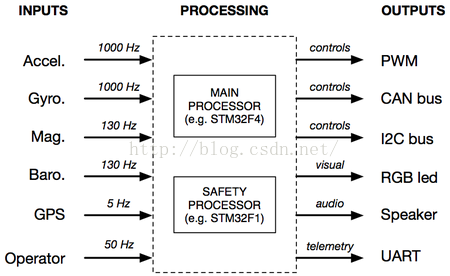


## pixhawk代码框架

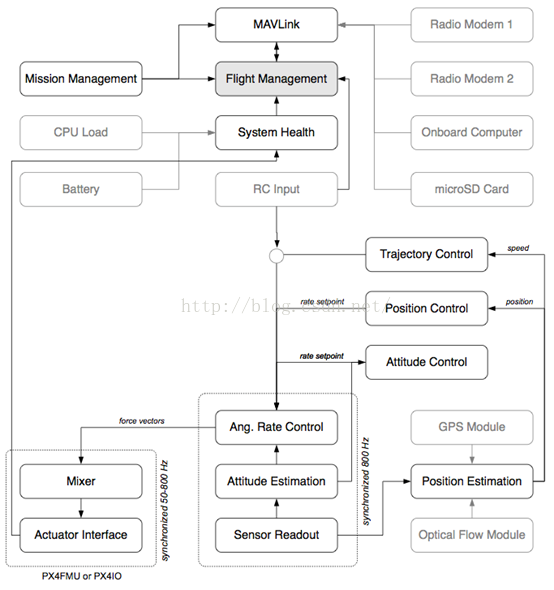


pixhawk的HAL



各组件的执行周期

pixhawk的控制图



整套系统的组件

在PX4Firmware/src/modules中的mc\_att\_control：姿态控制器和mc\_pos\_control位置控制器（mc：multicopter），整个系统都是围绕着这两个控制器。

        mc\_att\_control – Multirotor attitude controller

        mc\_pos\_control – Multirotor position controller

        The PX4 firmware is organized in priority bands:

1) (interrupt level) fast sensordrivers

2) watchdog/system state monitors

3) actuator outputs (PWM outputdriver thread, IO comms sender thread)

4) attitude controller(s)

5) slow/blocking sensor drivers(must not block attitude controllers)

6) destination/positioncontroller(s)

7) default priority - generic usercode, shell commands, random crap, all RR scheduled

8) logger, parameter syncer

9) idle

位置估计的有2套代码，lpe的请看[Lacal\_position\_estimator数据流](http://blog.csdn.net/czyv587/article/details/51814417)和[卡尔曼算法的理解](http://blog.csdn.net/czyv587/article/details/52061523)，inav的请看[position\_estimator\_inav.cpp思路整理及数据流](http://blog.csdn.net/czyv587/article/details/51884052)和[pixhawk position\_estimator\_inav.cpp再分析](http://blog.csdn.net/czyv587/article/details/52514041)

关于位置控制的请看[mc\_pos\_control.cpp源码解读](http://blog.csdn.net/czyv587/article/details/51728079)和[mc\_pos\_control.cpp思路整理](http://blog.csdn.net/czyv587/article/details/51878546)

关于姿态的估计和控制，写的比较水，不嫌弃的可以看[pixhawk姿态与控制部分的记录](http://blog.csdn.net/czyv587/article/details/51548377)

关于位置估计和控制、姿态估计和控制串联起来的可以看[pixhawk光流--位置估计--姿态估计--位置控制--姿态控制](http://blog.csdn.net/czyv587/article/details/51558612)(这篇博客写的时候有些概念还不是很清楚，也是边分析边写的，所以看起来比较混乱，请注意前面大概1/3的位置都是APM的，后面2/3的才是px4原生固件的)

关于调试的请看[为实际调试做的准备](http://blog.csdn.net/czyv587/article/details/51839978)、[调试记录（一）---关于参数设置的问题](http://blog.csdn.net/czyv587/article/details/51922787)、[调试记录(二)pixracer 进入光流模式](http://blog.csdn.net/czyv587/article/details/52119338)

关于飞行模式切换请看[commander.cpp的飞行模式切换解读](http://blog.csdn.net/czyv587/article/details/51777392)、[\_control\_mode如何产生的](http://blog.csdn.net/czyv587/article/details/52119059)

关于任务规划请看[commander--navigator--modules之间的联系](http://blog.csdn.net/czyv587/article/details/52120876)(未完待续)

关于px4驱动的请看[pixhawk px4 字符型设备驱动](http://blog.csdn.net/czyv587/article/details/53783004)，[pixhawk px4 spi设备驱动](http://blog.csdn.net/czyv587/article/details/53817154)

关于commander的请看[pixhawk px4 commander.cpp](http://blog.csdn.net/czyv587/article/details/77621011)

经常使用的两个官网地址

       1）[http://copter.ardupilot.com](http://copter.ardupilot.com/wiki/assembly-instructions/)

        2）[http://dev.ardupilot.com](http://dev.ardupilot.com/wiki/where-to-get-the-code/)

# [Pixhawk控制电机命令pwm源码详解](http://blog.csdn.net/wangjiaweiwei/article/details/49638369)

标签： [源码](http://www.csdn.net/tag/%e6%ba%90%e7%a0%81)[终端](http://www.csdn.net/tag/%e7%bb%88%e7%ab%af)[飞控](http://www.csdn.net/tag/%e9%a3%9e%e6%8e%a7)

2015-11-04 15:13 2517人阅读 [评论](http://blog.csdn.net/wangjiaweiwei/article/details/49638369#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/wangjiaweiwei/article/details/49638369#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

C语言（24） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg 飞控（2） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

在用USB转串口连接好飞控板之后，可以在终端使用命令pwm来进行设置pwm的输出值

例如：命令

nsh>pwm test –c 13 –p 1200

该命令用来测试通道1和3，并将他们的输出值设定为1200us。

pwm命令的源文件存储路径为：

originalcode/PX4Firmware/src/systemcmds/pwm/pwm.c

pwm命令的参数‘c’的源代码为：

case 'c':

                            /\*Read in channels supplied as one int and convert to mask: 1234 -> 0xF \*/

                            channels= strtoul(optarg, &ep, 0);

                            while((single\_ch = channels % 10)) {

                                     set\_mask|= 1<<(single\_ch-1);

                                     channels/= 10;

                            }

                            break;

设channels从命令行参数optarg得到的值为1234，则

single\_ch =1234 % 10=4；

set\_mask |= 1<<(single\_ch-1)=1<<(4-1)=1<<3

{set\_mask的初始值为0，所以，现在set\_mask=0|1<<3=0|1000=1000}

channels/=10

{channels = 1234/10 = 123}

此段源代码一直运行，直到single\_ch得到的值为0,此时set\_mask的值为1111，

即四个通道的掩码值都被设置为1，参数‘c’用于选择打开哪几个通道

pwm命令的参数‘p’的源代码为：

case 'p':

           pwm\_value = strtoul(optarg, &ep, 0);

                            if(\*ep != '\0')

                                     usage("BADPWM VAL");

                            break;

pwm\_value是pwm的数值，pwm\_value的值从命令行参数optarg得到，并且命令行参数optarg在输入pwm的数值之后必须要有结束符，参数‘p’用于改变pwm的数值

pwm命令中“test”参变量的源代码详解：

else if (!strcmp(argv[1],"test")) {

//用来判断参变量是否是“test”命令

                   if(set\_mask == 0) {

                            usage("nochannels set");

                   }

//如果set\_mask值为0，说明没有设置选通通道

                   if(pwm\_value == 0)

                            usage("noPWM provided");

//如果pwm\_value值为0，说明pwm的数值没有设定

                   /\*get current servo values \*/

                   structpwm\_output\_values last\_spos;

//last\_spos结构体用于存储当前电机的值

                   for(unsigned i = 0; i < servo\_count; i++) {

                            ret= ioctl(fd, PWM\_SERVO\_GET(i), (unsigned long)&last\_spos.values[i]);

//读取电机的当前数值

                            if(ret != OK)

                                     err(1,"PWM\_SERVO\_GET(%d)", i);

                   }

                   /\*perform PWM output \*/

                   /\*Open console directly to grab CTRL-C signal \*/

                   structpollfd fds;

                   fds.fd= 0; /\* stdin \*/

                   fds.events= POLLIN;

//设置控制台终端来接收CTRL-C信号，以终止程序

                   warnx("PressCTRL-C or 'c' to abort.");

                   while(1) {

                            for(unsigned i = 0; i < servo\_count; i++) {

//依次选中电机

                                     if(set\_mask & 1<<i) {

//set\_mask与1的左移i位后的值进行相与，使得只有相应的位为1，其他的位为0，例如,i=0,

//时，set\_mask的值为1,i=1时，set\_mask的值为2，i=2时，set\_mask的值为4，i=3时，set\_mask

//的值为8,根据set\_mask值得不同就可以设置，相应的电机的状态

                                               ret= ioctl(fd, PWM\_SERVO\_SET(i), pwm\_value);

//设置相应电机的pwm的值

                                               if(ret != OK)

                                                        err(1,"PWM\_SERVO\_SET(%d)", i);

                                     }

                            }

                            /\*abort on user request \*/

//以下代码用来在终端接收到指定的字符之后，重新恢复电机原来的转速

                            charc;

                            ret= poll(&fds, 1, 0);

                            if(ret > 0) {

                            read(0,&c, 1);

                                     if(c == 0x03 || c == 0x63 || c == 'q') {

//如果接收到的字符是EXT、C、Q几个字符，则会退出用户设定模式

                                               /\*reset output to the last value \*/

                                               for(unsigned i = 0; i < servo\_count; i++) {

                                                                                    if(set\_mask & 1<<i) {

                                                                                             ret= ioctl(fd, PWM\_SERVO\_SET(i), last\_spos.values[i]);

//将所有电机的油门值设为电机原来的油门值

                                                                                             if(ret != OK)

                                                                                                       err(1,"PWM\_SERVO\_SET(%d)", i);

                                                                                    }

                                                                           }

                                               warnx("Userabort\n");

//发出信息说明用户已经终止设置自定义油门参数

                                               exit(0);

                                     }

                            }

                            usleep(2000);

//睡眠2000毫秒

                   }

                   exit(0);

**1）名词解释**

惯性测量单元IMU(InertialMeasurementUnit)

姿态航向参考系统AHRS(Attitudeand Heading Reference System)

地磁角速度重力MARG(Magnetic,Angular Rate, and Gravity)

微机电系统MEMS(MicroElectrical Mechanical Systems)

自由度维数DOF(Dimension OfFreedom)

无人驾驶飞行器UAV(UnmannedAerial Vehicle)

扩展卡尔曼滤波EKF(ExtendedKalman Filter)

无损卡尔曼滤波UKF(UnscentedKalman Filter)

惯性导航系统INS(InertialNavigation System)

全球导航卫星系统GNSS(GlobalNavigation Satellite System)

天文导航系统CNS(CelestialNavigation System)

可垂直起降VTOL(VerticalTake-off and Landing)