# Cleanflight flight controller: A nose dive into the Frequency Domain

Anam Farrukh

2019

# Contents

# List of Figures

# 0 Introduction

Cleanflight's critical flight control comprises of attitude(3D orientation in space: roll, pitch and yaw) and position(x,y and z) control. The complete flight control loop can be approximated as a classic linear time invariant system that can be expressed in time domain as a convolution of its input signal(reference set point) with the impulse response of the system. The impulse response is the natural response of a system to a unit impulse that corresponds to the system's transfer function(transfer gain from input to output) in frequency domain(Z transform). The transfer function completely characterizes a system through its poles and zeros.

In this work we conduct a detailed theoretical analysis of CF's critical flight control loop and derive its system level transfer function. This would enable us to characterize our system and determine its properties in a principled way. It would also allow us to conduct a sensitivity analysis on the various system parameters; in particular the operating frequencies of various component subsystems(tasks) and the PID controller gains, that affect the overall stability of the system.

For this analysis, we only consider attitude control(orientation in 3D space represented as rotations about the roll, pitch and yaw axis of the sensor frame relative to the earth frame). Our current Cleanflight implementation on SPRACINGF3 micro-controller board has an active IMU(accelerometer, gyroscope and magnetometer) sensor chip that communicates discretized as well as quantized sensor data values(rates of rotation, angle of rotation and heading information) to the flight controller through an $I^2C$ bus. For position control we would need a GPS and a barometer to estimate the x, y and z values in 3D space and will be considered in future implementations. For the rest of this document attitude control and flight control are thus interchangeably used.

We express Cleanflight's attitude control as a feedback control loop where the feedback path incorporates an Attitude and Heading Reference sub-System(AHRS) namely Madwick&Mahony's complimentary filter algorithm for state estimation, to estimate the current attitude of the system from sampled sensor(IMU+Magnetometer) data. The forward path computes the error in current attitude from the desired attitude(reference signal) and feeds it to the PID controller. The output of the controller is mixed with the throttle input and fed to the motors as standard PWM signals via the ESCs. The block diagram in Figure 1 represents the overall attitude control system where various critical component subsystems are marked in orange circles. We analyse each critical subsystem in turn and derive their representative transfer functions in frequency domain.

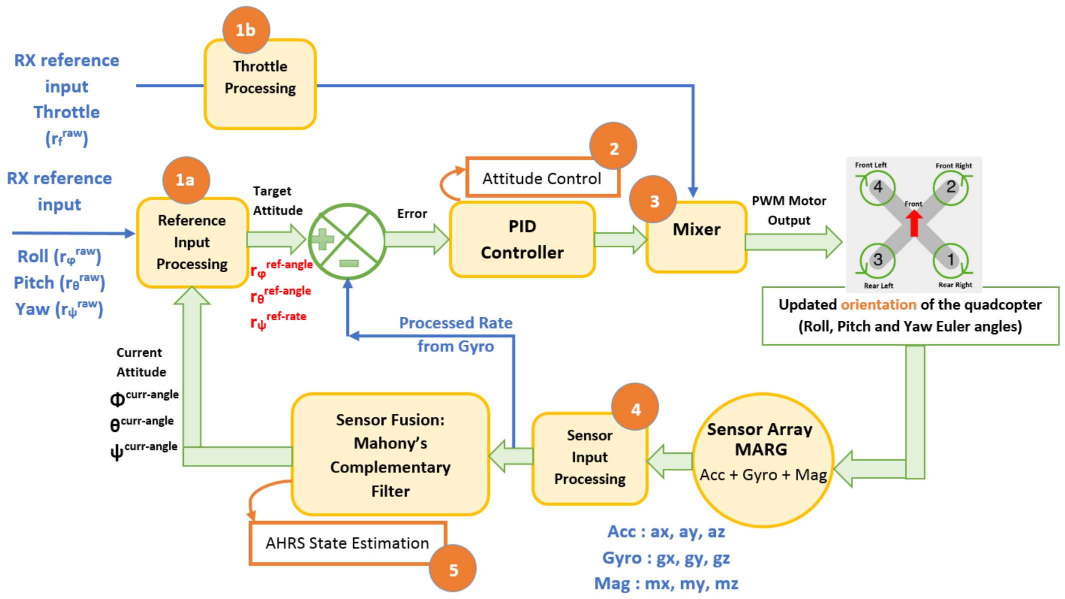


**Figure 1:** Cleanflight Flight/Attitude Control Loop

## 1 Sub-System 1: Receiver Input Processing

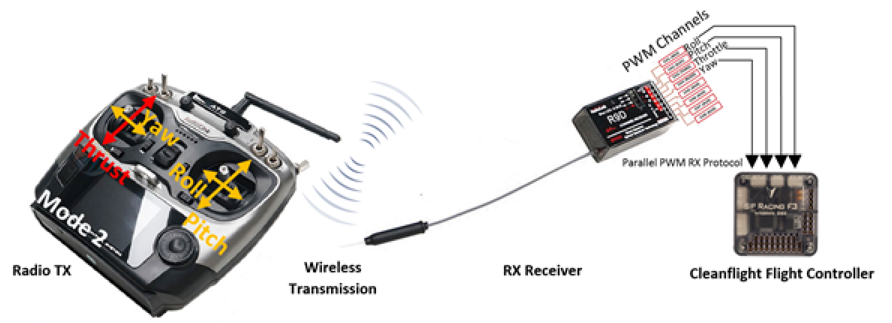


**Figure 2:** Transmitter-Receiver System with Parallel PWM communication protocol used between the four main receiver channels(roll:$\phi$, pitch:$\theta$, yaw:$\psi$ and throttle:$f$) and the Cleanflight flight controller running on top of an SPRACINGF3 board.

Cleanflight receives user stick inputs as radio control signals from the remote transmitter for each of the Roll($r_{\phi}^{raw}$), Pitch($r_{\theta}^{raw}$), Yaw($r_{\psi}^{raw}$) and Throttle($r_{f}^{raw}$) commands and converts them into reference set-point angles and rates for the PID controller. The throttle input($r_{f}^{raw}$), after initial processing, is sent directly to the mixer while the roll, pitch and yaw inputs are processed further and used in determining the attitude error for the overall flight control loop (figure 1).
Processing of RX data within Cleanflight depends on the type of receiver protocol used. Receiver protocols determine and control the transfer of data between the hardware receiver and the flight controller(figure:2). Cleanflight supports three major classes of protocols: 1. Parallel PWM(1 channel per input pin), 2. PPM and 3. Serial. For the sake of simplicity in the data analysis we consider the Parallel PWM receiver protocol, where each receiver channel is connected to a dedicated input pin of the flight controller. Data for each axis is read from a pre-assigned channel by sampling the corresponding GPIO pin at a fixed rate(= RX task rate).
We consider our quad to be operating under ANGLE MODE(self-level mode). Under this mode stick deflections directly correspond to the target angle, the drone should be inclined at, in each of the three axes; roll, pitch and yaw.
After the raw data is sampled($r_{\{\phi,\theta,\psi,f\}}^{raw}$), an average value is calculated for

each axis according to the equation:

$$r^{data}_{\{\phi,\theta,\psi,f\}}[n] = \frac{1}{N}\sum_{k=0}^{N-1} r^{raw}_{\{\phi,\theta,\psi,f\}}[n-k] \tag{1}$$

where $N = 3$ are the number of samples. The above equation represents a moving average filter in time domain. Taking the z transform on both sides of the equation results in:

$$R^{data}_{\{\phi,\theta,\psi,f\}}(z) = \left(\frac{1}{N}\sum_{k=0}^{N-1} z^{-k}\right) R^{raw}_{\{\phi,\theta,\psi,f\}}(z), \quad \text{ROC}(H_{avg}) \cap \text{ROC}(R^{raw}) \tag{2}$$

The transfer function for the filter thus becomes:

$$H_{avg}(z) = \frac{R^{data}_{\{\phi,\theta,\psi,f\}}(z)}{R^{raw}_{\{\phi,\theta,\psi,f\}}(z)} = \frac{1}{N}\sum_{k=0}^{N-1} z^{-k} \tag{3}$$

With N=3 the transfer function can be simplified to a rational form:

$$\boxed{H_{avg}(z) = \frac{1}{3}\left(\frac{z^2+z+1}{z^2}\right), \quad |z| > 0} \tag{4}$$

The pole zero plot is shown in figure 3. Since the averaging system is causal and stable, the Region of Convergence(ROC) for the filter extends outwards from the two poles at z=0 and includes the unit circle.

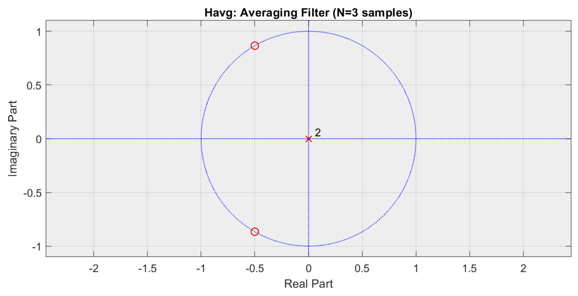


**Figure 3:** Pole zero plot of the Averaging Filter on the z-plane. The two conjugate zeros lie on the unit circle.

Further processing of each channel is detailed below:

- Roll & Pitch:
  - Time Domain:

    $$r^{cmd}_{\{\phi,\theta\}}[n] = r^{data}_{\{\phi,\theta\}}[n] - C_{midrc} \tag{5}$$

    where:

    * $C_{midrc} = 1500$ = configurable constant, fixed to this value for the analysis to hold.

    We do not define any deadband for the roll and pitch axis, implying that the above equation holds for the entire range of rx values: 1000 to 2000.
  - Z Domain:

    $$R^{cmd}_{\{\phi,\theta\}}(z) = R^{data}_{\{\phi,\theta\}}(z) - C_{midrc}\left(\frac{z}{z-1}\right), \quad \text{ROC}(R^{data}) \cap (|z| > 1) \tag{6}$$

- Yaw:
  - Time Domain:

    $$r^{cmd}_{\psi}[n] = C_{rev-dir}(r^{data}_{\psi}[n] - C_{midrc}) \tag{7}$$

    where:

    * $C_{midrc} = 1500$ = configurable constant
    * $C_{rev-dir}$ is the unity gain to control the yaw direction

    We do not define any dead band for the yaw axis as well implying that the above equation holds for the entire rx range.
    For our analysis we assume the roll, pitch and yaw sticks to be completely centered on the radio controller. This translates to a target attitude corresponding to a hover mode with parallel alignment to the earth's xy plane. The flight controller thus detects the yaw command($r^{cmd}_{\psi}$) to lie below a certain threshold and processes it further to incorporate corrections for any deviations in the current measured value of the yaw angle($\psi[n]$) from its last

value. A proportional controller is used for this purpose. The time equation is as follows:

$$r_{\psi}^{cmd-corr}[n] = r_{\psi}^{cmd}[n] - \left(C_{rev-dir}K_{p_{mag}}(\psi[n] - \psi[n-k])\right) \quad (8)$$

where:

* $K_{p_{mag}}$ = Configurable constant for yaw proportional control
* $\psi$ = Measured yaw angle from the AHRS sub-system (equ: 5.6), k is assumed to be 1 for simplicity.

– Z domain:

$$R_{\psi}^{cmd}(z) = C_{rev-dir}(R_{\psi}^{data}(z) - C_{midrc}) \quad (9a)$$

$$R_{\psi}^{cmd-corr}(z) = R_{\psi}^{cmd}(z) - \Psi(z)\left[C_{rev-dir}K_{p_{mag}}\left(\frac{z-1}{z}\right)\right] \quad (9b)$$

- Throttle:

Throttle stick deflections are mapped to data values according to a throttle curve. The curve plots the stick deflection over the entire range, on the (horizontal) independent axis and the throttle percentage values on the (vertical) dependent axis. Cleanflight provides two configurable parameters: throttle expo(exponential): $C_{thr-expo}$ and throttle mid: $C_{thr-mid}$, to control the shape of the throttle curve. Throttle expo($C_{thr-expo}$) controls the stick sensitivity around the throttle mid position($C_{thr-mid}$) by flattening the curve around that point. It thus allows a softer throttle response at maximum stick resolution around the throttle mid position. Throttle Mid($C_{thr-mid}$) on the other hand, controls the position where throttle expo is applied. For our analysis we consider a straight throttle curve($C_{thr-expo} = 0$) with no inflection region around the throttle mid-point; $C_{thr-mid} = 50\%$ (figure: 4).

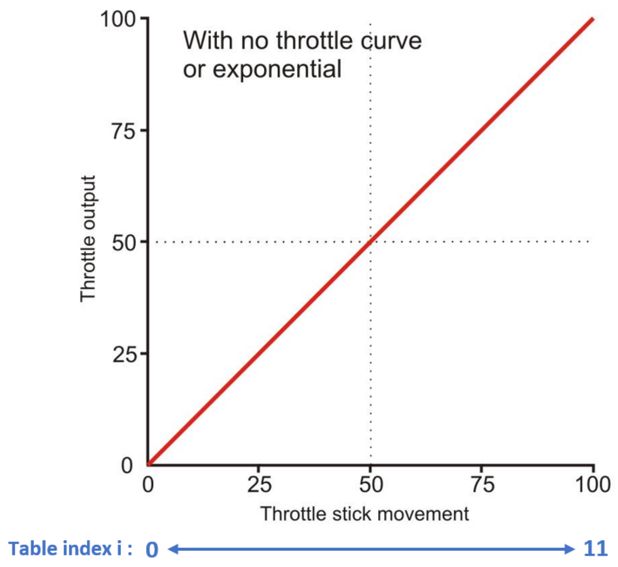


**Figure 4:** Cleanflight's Throttle curve.

Sampled values are mapped according to the curve during the throttle processing stage. A static look up table is used with the following mapping equation for the straight line graph(figure:4):

$$r_f^i[n] = C_{PWM-min} + 100i[n] \tag{10}$$

where:

- i = index into the table representative of the throttle stick position on the radio controller. The whole range of stick deflection is divided into 12 distinct steps thus i ranges from 0 to 11.
- $r_f^i$ = static throttle value at table index i
- $C_{PWM-min} = 1000$ = minimum throttle constant which adds a bias to all throttle values calculated from the graph. The bias brings throttle values within ranges that cleanflight understands: $C_{PWM-range} = C_{PWM-max} - C_{PWM-min} = 2000 - 1000 = 1000$.
- The slope of the curve is 100 if the dependent variable is i. In terms of percentage stick movement, the slope is 1 as shown in the graph in the figure.

The frequency response of $r_f^i[n]$ is:

$$R_f^i(z) = C_{PWM-min} + 100I(z) \tag{11}$$

The stick deflection and hence the table index, i, is calculated at run-time using $r_f^{data}[n]$ from equation 1:

$$i[n] = \frac{1}{100}\left(\frac{r_f^{data}[n] - C_{mincheck}}{C_{PWM-max} - C_{mincheck}}\right)C_{PWM-min} \tag{12}$$

where:

- $C_{mincheck} > C_{PWM-min}$ = minimum throttle constant above which the quad is armed. Once the quad arms, all RX values are measured/calculated with respect to $C_{mincheck}$.
- $C_{PWM-max} = 2000$ = maximum throttle constant

Equation 12 thus maps the sampled and averaged throttle data ($r_f^{data}[n]$) into a valid index, i for indexing into the throttle look up table, where $i[n]$ corresponds to the stick deflection at time step n.
The z transform is:

$$I(z) = C_k(R_f^{data}(z) - C_{mincheck}) \tag{13}$$

where $C_k = \frac{C_{PWM-min}}{100(C_{PWM-max} - C_{mincheck})}$.

The final processed throttle data: $r_f^{cmd}[n]$ after lookup is calculated as:

$$\begin{aligned} r_f^{cmd}[n] &= r_f^i[n] + \left(\frac{r_f^{i+1}[n] - r_f^i[n]}{100}\right)(100i[n] - \lfloor 100i[n] \rfloor) \\ &= r_f^i[n] + (100i[n] - \lfloor 100i[n] \rfloor) \\ &= r_f^i[n] + 100i[n] - i^{'}[n] \end{aligned} \tag{14}$$

where:

- $\frac{r_f^{i+1} - r_f^i}{100} = 1$ = slope of the straight line graph
- $r_f^i[n]$ = value read from the table at index i
- $i[n]$ = index at time step n

Taking the z transform on both sides of the equation yields:

$$R_f^{cmd}(z) = R_f^i(z) + 100I(z) - \mathcal{Z}\{\lfloor 100i[n] \rfloor\} \tag{15a}$$
$$= C_{PWM-min} + 200I(z) - I'(z) \tag{15b}$$

$r_f^{cmd}[n]$ is then sent directly on to the mixer.

## 1.1 Reference set-points for Roll($\phi$), Pitch($\theta$) and Yaw($\psi$)

Processed command values from the previous subsection, for each of the three rotational axes ($\phi, \theta, \psi$) are normally passed through a linear smoothing filter. Cleanflight controls the level of smoothness by a configurable step-size parameter. For this work, we fix the RC commands at values corresponding to centered stick positions which basically makes smoothing non-essential. We thus turn off the smoothing filter.

As the final processing step in determining the reference set-points ($r_{\phi,\theta,\psi}^{ref-rate}$) for the roll, pitch and yaw axis, Cleanflight first normalizes the command values to get corresponding stick deflections that range from -1 to 1:

$$r_{\phi,\theta}^{deflection}[n] = \frac{r_{\phi,\theta}^{cmd}[n]}{500} \tag{16a}$$
$$r_{\psi}^{deflection}[n] = \frac{r_{\psi}^{cmd-corr}[n]}{500} \tag{16b}$$

The z transforms are:

$$R_{\phi,\theta}^{deflection}(z) = \frac{R_{\phi,\theta}^{cmd}(z)}{500}, \qquad \text{ROC}(R^{cmd}) \tag{17a}$$
$$R_{\psi}^{deflection}(z) = \frac{R_{\psi}^{cmd-corr}(z)}{500}, \qquad \text{ROC}(R^{cmd-corr}) \tag{17b}$$

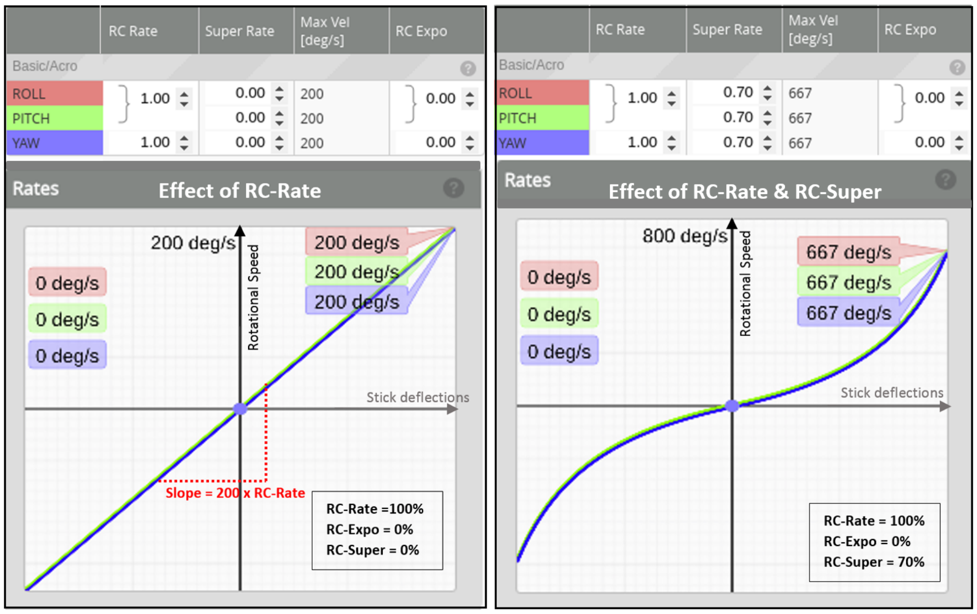
Deflection values are then mapped to reference set-point rates according to rotational rate v/s stick position curve(figure:5) for each axis. The shape of each curve is controlled by 3 configurable parameters that are borrowed from Betaflight:

1. *RC-Rate*($C_{rc-rate}^{\phi,\psi} = 100\%$): Controls the sensitivity of the sticks. A higher rate would make the quad more responsive since a small stick

deflection would translate to a large rotation of the quad about the corresponding axis. This parameter controls the slope of the curve: $slope = 200C^{\phi,\psi}_{rc-rate}$ where the slope for roll and pitch axis is controlled by the same constant:$C^{\phi}_{rc-rate}$.
The left half of figure 5 shows the effect of RC-Rate(= 100%) on each of the three curves for roll, pitch and yaw. RC-Expo and RC-Super are set to 0 hence giving a straight line through the mid-stick position corresponding to 0 rotational speed.

2. *RC-Expo*(exponential)($C^{\phi,\psi}_{rc-expo} = 0\%$): Reduces the sensitivity of the sticks near the center point while retaining maximum rotation speed at the min and max points of the deflection range(horizontal axis of the graph). A 0% value implies a linear relation with no inflection region around the mid-stick position.

3. *RC-Super*$(C^{\phi,\theta,\psi}_{rc-super}) = 70\%$: A hybrid combination of RC-Rate and RC-Expo. This parameter changes the full deflection rate as well as the center stick precision. A higher value would result in a moderate stick sensitivity around the mid stick position and high rotational speeds near stick end points. The right half of figure 5 shows the combined effect of RC-Super with RC-Rates. This is the curve Cleanflight uses by default to map stick deflections to set-point rates.



| Basic/Acro | RC Rate | Super Rate | Max Vel [deg/s] | RC Expo |
|---|---|---|---|---|
| ROLL | 1.00 | 0.00 | 200 | 0.00 |
| PITCH | | 0.00 | 200 | |
| YAW | 1.00 | 0.00 | 200 | 0.00 |

| Basic/Acro | RC Rate | Super Rate | Max Vel [deg/s] | RC Expo |
|---|---|---|---|---|
| ROLL | 1.00 | 0.70 | 667 | 0.00 |
| PITCH | | 0.70 | 667 | |
| YAW | 1.00 | 0.70 | 667 | 0.00 |

**Figure 5:** Left:Effect of RC Rates, Right: Combined effect of RC-Rates and RC-Super on the rotational speed v/s stick deflection curves for Roll, Pitch and Yaw channels. [3]

The time equations for the reference rates are:

$$r_{\phi,\theta}^{ref-rate}[n] = \frac{(200C_{rc-rate}^{\phi})r_{\phi,\theta}^{deflection}[n]}{1-(C_{rc-super}^{\phi,\theta}\left\|r_{\phi,\theta}^{deflection}[n]\right\|)} \tag{18a}$$

$$\boxed{r_{\psi}^{ref-rate}[n]} = \frac{(200C_{rc-rate}^{\psi})r_{\psi}^{deflection}[n]}{1-(C_{rc-super}^{\psi}\left\|r_{\psi}^{deflection}[n]\right\|)} \tag{18b}$$

For the sake of simplicity of our analysis, we assume the transfer functions to be:

$$H_{\phi,\theta}^{ref-rate} = \frac{R_{\phi,\theta}^{ref-rate}(z)}{R_{\phi,\theta}^{deflection}(z)} = M_{\phi,\theta}^{rate-curve} \tag{19a}$$

$$\boxed{H_{\psi}^{ref-rate}} = \frac{R_{\psi}^{ref-rate}(z)}{R_{\psi}^{deflection}(z)} = M_{\psi}^{rate-curve} \tag{19b}$$

Yaw control, irrespective of the mode of operation, is purely rate based. This implies that the reference rate from equation 18b is directly sent as input to the PID controller(sec:2) for yaw control. However, for Roll and Pitch, the the flight controller's mode of operation determines the set-point for each axis's respective PID controller. For the default ACRO mode (rate based mode), the reference rate is determined according to equation 18a but for ANGLE mode, a set-point angle is calculated instead. This angle is based on the 1. roll and pitch stick deflection value(equ:16a) and the 2. error in the current attitude angle from the target inclination in the corresponding axis. The calculation proceeds in the following sequence of steps:

1. Current roll and pitch angle is determined from the stick deflections:

$$\phi^{target}[n] = C_{level-angle} r_{\phi}^{deflection}[n] \tag{20a}$$
$$\theta^{target}[n] = C_{level-angle} r_{\theta}^{deflection}[n] \tag{20b}$$

   where $C_{level-angle} = 55°$ is a configurable constant that linearly maps the stick deflection onto the target roll and pitch angle. $r_{\phi,\theta}^{deflection}$ lies within the range $\left[-1, 1\right]$. Calculating the z transforms yields:

$$\Phi^{target}(z) = C_{level-angle} R_{\phi}^{deflection}(z), \qquad \text{ROC}(R_{\phi}^{deflection}) \tag{21a}$$
$$\Theta^{target}(z) = C_{level-angle} R_{\theta}^{deflection}(z), \qquad \text{ROC}(R_{\theta}^{deflection}) \tag{21b}$$

2. Next the error in angle is calculated:

$$e_{\phi}[n] = \phi^{target}[n] - C_{att-scale}(\phi[n] - C_{\phi}^{trim}) \tag{22a}$$
$$e_{\theta}[n] = \theta^{target}[n] - C_{att-scale}(\theta[n] - C_{\theta}^{trim}) \tag{22b}$$

   where:

   - $C_{att-scale}$ : $\frac{1}{10}$ Scaling constant for converting deci-degrees to degrees for the angle
   - $\phi[n]$ & $\theta[n]$ : Current measured value of the roll and pitch angle. The current value is determined by the AHRS sub-system(sec:5) in the feedback path(figure:1).
   - $C_{\phi,\theta}^{trim}$ : Configurable angle trim values. These values can be configured at run-time through a specific combination of roll and pitch stick inputs. After every change they are saved in the flight controller's EEPROM. For our purposes we are only interested in

hover mode, so we intend to keep the roll and pitch sticks centered and do not intend to change the input during the entire duration of flight. This means we only need to set the angle trims once during the configuration stage after system startup. Updating the trims involves a memory write and read cycle which has a much higher latency than the loop times for our flight control. We thus consider the trim values as constants and disconnect them from the main run-time flight controller loop.
The Z transform of the time equations is:

$$E_\phi(z) = \Phi^{target}(z) - C_{att-scale}\Phi(z) + C_{att-scale}C_\phi^{trim}\left(\frac{z}{z-1}\right), \tag{23a}$$

$$\textit{where:} \quad \text{ROC}(\Phi^{target}) \cap \text{ROC}(\Phi) \cap (|z| > 1)$$

$$E_\theta(z) = \Theta^{target}(z) - C_{att-scale}\Theta(z) + C_{att-scale}C_\theta^{trim}\left(\frac{z}{z-1}\right), \tag{23b}$$

$$\textit{where:} \quad \text{ROC}(\Theta^{target}) \cap \text{ROC}(\Theta) \cap (|z| > 1)$$

- As a final step, a proportional gain of a P controller is applied to the error to calculate the final angle set-point:

$$\boxed{r_{\phi,\theta}^{ref-angle}[n] = P_{level-gain}e_{\phi,\theta}[n]} \tag{24}$$

  where $P_{level-gain} = 5$ is a proportional constant for P controller. The transfer function is:

$$\boxed{H_{\phi,\theta}^{ref-angle} = \frac{R_{\phi,\theta}^{ref-angle}(z)}{E_{\phi,\theta}(z)} = P_{level-gain}, \quad \text{ROC:}\textit{All } z} \tag{25}$$

The reference angle values for roll and pitch are then sent onto the PID Controller. The next subsection shows the overall RX processing system.

## 1.2 Block Diagram of the Overall Receiver Sub-system

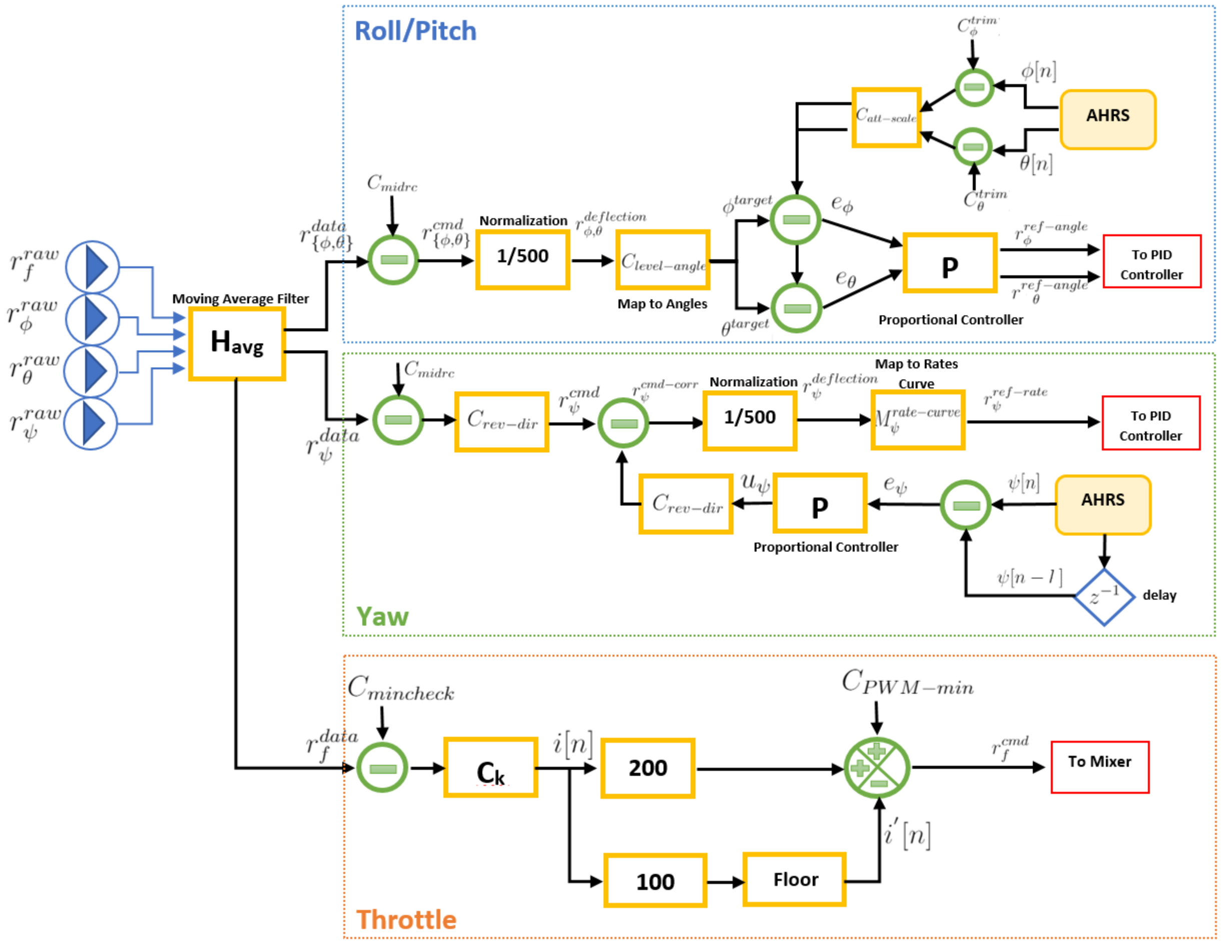


**Figure 6:** Processing of received data for Throttle, Roll, Pitch and Yaw user commands

The final equations are:

$$\begin{aligned} R_{\phi}^{ref-angle}(z) = &\left(\frac{C_{level-angle}P_{level-gain}}{1500}\right)\left(\frac{z^2+z+1}{z^2}\right)R_{\phi}^{raw}(z) \\ &- \frac{P_{level-gain}}{500}\left(\frac{z}{z-1}\right)\left(C_{level-angle}C_{midrc} - 500C_{att-scale}C_{\phi}^{trim}\right) \\ &- \left(P_{level-gain}C_{att-scale}\right)\Phi(z), \end{aligned} \tag{26}$$

The Region of Convergence of the above equation is an intersection of each individual term's ROC.

## 2 Sub-System 2: PID Attitude Controller

Roll, Pitch and Yaw are represented by Euler angles $\phi$, $\theta$ and $\psi$ respectively. Error in rate for each of these axis is:

$$e^{rate}_{\phi,\theta,\psi}[n] = r_{\phi,\theta,\psi}[n] - y^{rate}_{\phi,\theta,\psi}[n] \tag{27}$$

where n is a discrete time step. $r_{\phi,\theta,\psi}[n]$ is the setpoint rate of rotation(reference signal) of the quad in each axis and $y^{rate}_{\phi,\theta,\psi}[n]$ is the current rate of rotation as sampled from the gyroscope.
The corresponding Z transform of the error signal is:

$$E^{rate}_{\phi,\theta,\psi}(z) = R_{\phi,\theta,\psi}(z) - Y^{rate}_{\phi,\theta,\psi}(z) \tag{28}$$

The output of the PID controller for each axis is the sum of P, I and D terms of the corresponding axis where P, I and D represent Proportional, Integral and Derivative control outputs respectively for each axis:

$$u_{\phi,\theta}[n] = P_{\phi,\theta}[n] + I_{\phi,\theta}[n] + D_{\phi,\theta}[n] \tag{29}$$

$$u_{\psi}[n] = P_{\psi}[n] + I_{\psi}[n] \tag{30}$$

The overall transfer function of the PID controller can be determined by taking the Z transform of equation [29] and equation [30] and using the property of linearity of Z transforms:

$$\begin{aligned} U_{\phi,\theta}(z) &= P_{\phi,\theta}(z) + I_{\phi,\theta}(z) + D_{\phi,\theta}(z) \\ &= TF^{P}_{\phi,\theta}(z)E^{rate}_{\phi,\theta}(z) \\ &+ TF^{I}_{\phi,\theta}(z)E^{rate}_{\phi,\theta}(z) \\ &+ TF^{D}_{\phi,\theta}(z)E^{rate}_{\phi,\theta}(z) \end{aligned} \tag{31}$$

$$\begin{aligned} U_{\psi}(z) &= P_{\psi}(z) + I_{\psi}(z) \\ &= TF^{P}_{\psi}(z)E^{rate}_{\psi}(z) \\ &+ TF^{I}_{\psi}(z)E^{rate}_{\psi}(z) \end{aligned} \tag{32}$$

where $TF^{P,I,D}(z)$ are the transfer functions of each of the P,I and D terms of the PID controller per axis. The overall PID transfer function thus becomes:

$$\boxed{\frac{U_{\phi,\theta}(z)}{E^{rate}_{\phi,\theta}(z)} = TF^{P}_{\phi,\theta}(z) + TF^{I}_{\phi,\theta}(z) + TF^{D}_{\phi,\theta}(z)} \tag{33}$$

and:

$$\boxed{\frac{U_\psi(z)}{E^{rate}_\psi(z)} = TF^P_\psi(z) + TF^I_\psi(z)} \tag{34}$$

A block diagram of the PID controller with the corresponding input and output signals for the Roll($\phi$) axis is as shown in figure 7. The diagram for pitch($\theta$) and yaw($\psi$) axis would be similar.

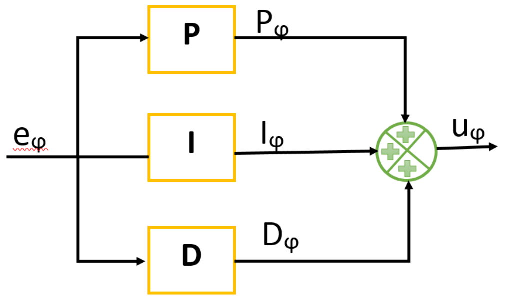


**Figure 7:** PID Controller for Roll Axis

## 2.1 Matlab's 2DoF PID Controller

Betaflight/Cleanflight PID controller uses Matlab's 2DoF PID Controller as reference design.

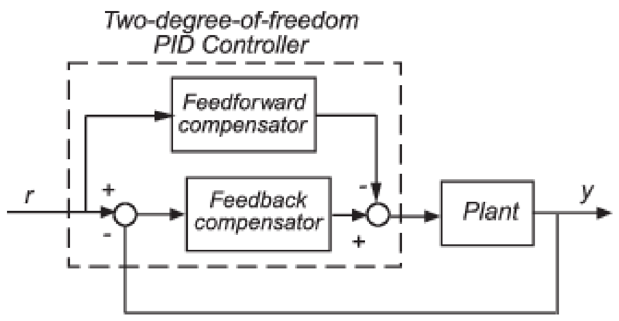


**Figure 8:** 2DoF PID Controller with feedforward(PD) and feedback(PID) compensator [1]

It has the following discrete time transfer function [1]:

$$u = K_p(br - y) + K_i \frac{T_s}{z-1}(r-y) + \left(\frac{K_d}{T_f}\right)\left(\frac{z-1}{z + (\frac{T_s}{T_f} - 1)}\right)(cr - y) \quad (35)$$

where

- $K_p$ =proportional gain
- $K_i$ =integrator gain
- $K_d$ =derivative gain
- $T_s$ =sampling time = $\frac{1}{samplingfreq}$
- $T_f$ =first order derivative filter time constant
- $b$ =setpoint weight on the proportional term
- $c$ =setpoint weight on the derivative term

Setpoint weights b and c determine the strength of the proportional and derivative action in the feedforward compensator. For Cleanflight's PID controller, $b = 1$ and c is a tunable value set to a constant during system configuration. For our case, since we are operating under ANGLE MODE, c is set to 0.

## 2.2 Time Equations and Transfer Functions

### 2.2.1 Proportional Control

- Roll & Pitch:

$$P_{\phi,\theta}[n] = C_{tpa} K_{p,\{\phi,\theta\}} e_{\phi,\theta}[n] \quad (36)$$

where

  - $K_{p,\{\phi,\theta\}}$ = proportional gain for roll($\phi$) and pitch($\theta$) axis
  - $C_{tpa}$ = TPA(Throttle PID Attenuation) factor = constant if applied throttle < TPA Threshold
  - $e_{\phi,\theta}[n]$ = error signal at time step n

The Z transform is as follows:

$$P_{\phi,\theta}(z) = C_{tpa}K_{p,\{\phi,\theta\}}E_{\phi,\theta}(z) \tag{37}$$

The transfer function for P term for roll and pitch is:

$$\boxed{TF^{P}_{\phi,\theta}(z) = \frac{P_{\phi,\theta}(z)}{E_{\phi,\theta}(z)} = C_{tpa}K_{p,\{\phi,\theta\}}} \tag{38}$$

- Yaw:

$$P^{unfiltered}_{\psi}[n] = C_{tpa}K_{p,\psi}e_{\psi}[n] \tag{39}$$

  The corresponding Z transform is:

$$P^{unfiltered}_{\psi}(z) = C_{tpa}K_{p,\psi}E_{\psi}(z) \tag{40}$$

  The transfer function of unfiltered P term for yaw is:

$$\boxed{\frac{P^{unfiltered}_{\psi}(z)}{E_{\psi}(z)} = C_{tpa}K_{p,\psi}} \tag{41}$$

  For the yaw term, a first order, low-pass PT1 filter is applied to unfiltered $P_{\psi}[n]$.

$$P^{LPF}_{\psi}[n] = LPfilter(P^{unfiltered}_{\psi}[n]) \tag{42}$$

  The filter for a generic input signal($in$[n]) is defined as follows:

$$\begin{aligned} in^{LPF}[n] &= LPfilter(in[n]) \\ &= \left\{(\frac{T_s}{RC + T_s})(in[n] - in^{LPF}[n-1])\right\} + in^{LPF}[n-1] \end{aligned} \tag{43}$$

  where

  - $in[n] = P_{\psi}[n]$ = input signal to be filtered at time step n
  - $in^{LPF}[n-1]$ = filtered value from previous time step
  - $in^{LPF}$ = filtered output
  - $T_s$ = sampling Period = constant PID looptime in seconds set at system configuration time = $\frac{1}{PID-sampling-frequency}$

– $RC$ = filter time constant = $\frac{1}{2\pi f_{cut-off}}$, $f_{cut-off}$ is set at configuration time

Thus the final proportional term for yaw:

$$\begin{aligned} P_\psi^{LPF}[n] &= \left\{ (\frac{T_s}{RC+T_s})(P_\psi^{unfiltered}[n] - P_\psi^{LPF}[n-1]) \right\} + P_\psi^{LPF}[n-1] \\ &= C_{LPF}(P_\psi^{unfiltered}[n] - P_\psi^{LPF}[n-1]) + P_\psi^{LPF}[n-1] \\ &= C_{LPF}P_\psi^{unfiltered}[n] - (C_{LPF}+1)P_\psi^{LPF}[n-1] \end{aligned} \tag{44}$$

where $C_{LPF}$ is a constant term = $\frac{T_s}{RC+T_s}$. Using the linearity and time delay property of Z transforms we get the following Z transform for the filtered proportional term of yaw axis:

$$P_\psi^{LPF}(z) = C_{LPF}P_\psi^{unfiltered}(z) - (C_{LPF}+1)z^{-1}P_\psi^{LPF}(z) \tag{45}$$

The block diagram representation of the above filter equation and the corresponding graphical simplification is presented in figure 9.

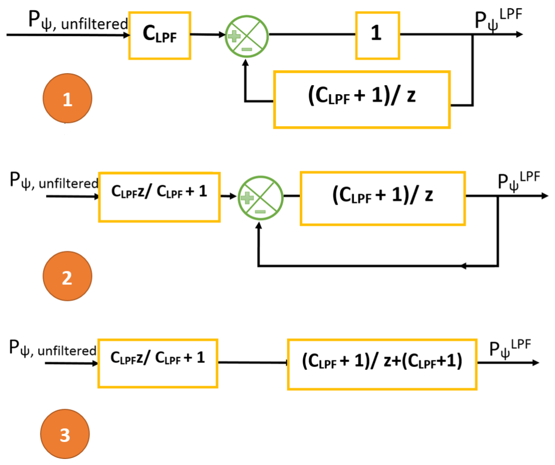


**Figure 9:** Block Diagram Representation of Proportional term's filter function for Yaw($\psi$) axis. The graphical simplification steps are numbered in orange with step 1 depicting equation 45 and step 3 giving the final simplified form.

From figure 9 step 3 we can determine the final gain for the filter as follows:

$$\boxed{\frac{P_\psi^{LPF}(z)}{P_\psi^{unfiltered}(z)} = \frac{C_{LPF}z}{z + (C_{LPF}+1)}} \tag{46}$$

Figure 10 shows the overall transfer function block diagram for yaw axis's P term. Thus the final transfer function for yaw is:

$$\boxed{\begin{aligned} TF_\psi^P(z) = \frac{P_\psi^{LPF}(z)}{E_\psi(z)} &= \frac{C_{tpa}K_{p,\psi}C_{LPF}z}{z + (C_{LPF}+1)} \\ &= \frac{C_{DC}z}{z + (C_{LPF}+1)} \end{aligned}} \tag{47}$$

where $C_{DC} = C_{tpa}K_{p,\psi}C_{LPF}$ is the DC gain.

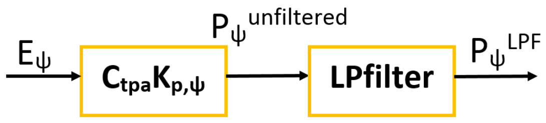


**Figure 10:** Transfer Function of Proportional term for Yaw($\psi$) axis

### 2.2.2 Integral Control

The I term considers the history of the error signal values($e_{\phi,\theta,\psi}$) accumulated over time. Normally for continuous time this corresponds to the following formula:

$$I(t) = K_i \int_0^t e(\tau)d\tau \tag{48}$$

In discrete domain we convert the integral into a sum of distinct non-overlapping areas under the error signal v/s time graph for each sampling period. This approximation is done by cleanflight using rectangles to represent accumulated error per sampling period $T_s$. This sort of approximation is known as Forward Euler and is represented by the following discrete time formula:

$$I[n] = I[n-1] + K_iT_se[n] \tag{49}$$

The first part of the equation refers to the accumulated error till [n-1] time step and the second part gives the approximated accumulated error between time step [n] and [n-1] i.e for one sampling time period, $T_s$. The equation for the integral term for each axis is thus as follows:

$$I_{\phi,\theta,\psi}[n] = I_{\phi,\theta,\psi}[n-1] + K_{i,\{\phi,\theta,\psi\}} C_{Idyn} e_{\phi,\theta,\psi}[n] \tag{50}$$

Assumption: For integral control the above equation holds if the motor mixer output is not saturated and is within range which implies that motor mix range is $< 1.0$. The maximum allowable motor mix range($max\ mix - min\ mix$) by the Cleanflight mixer for an X configured quadcopter is 3.0 i.e -1.5 to 1.5. The mix range depends on the final mix value for each motor. Details of mixing are presented in the next section(section:3). For simplification of analysis we assume that the motor mix range(max mix - min mix) remains $< 1.0$ and consequently the integration term never reaches the windup point.

The sampling period for the PID controller in Cleanflight is

$$T_s = \frac{1}{PID - loop - frequency} = \frac{1}{2(PID - Nyquist - frequency)}$$

For the purposes of our analysis we assume $C_{Idyn}$ is a constant that is directly proportional to $T_s$. The complete equation for $C_{Idyn}$ is as follows:

$$\begin{aligned} C_{Idyn} &= min\left(\left[1 - C_{mix-range}\right]\left[\frac{1}{1 - C_{windup-pt}}\right], 1.0\right) C_{accelerator-gain} T_s \\ &= (T_s)(C_{idyn}) \end{aligned} \tag{51}$$

where

- $C_{mix-range}$ = motor mix range assumed to be constant and $< 1.0$
- $C_{windup-pt}$ = integral windup point is set during system configuration hence constant, default value = 50%. Cleanflight dynamically scales the I term if the motor mix is saturated and if the quad cannot adjust the mix to reduce the error in rate of rotation from the target setpoint rate.
- $C_{accelerator-gain}$ = constant. This encapsulates the rate of change of throttle input which directly affects the change in setpoint rate.
- $C_{idyn}$ = product of all the constant terms with $T_s$ factored out.

Thus the discrete time equation [50] then becomes:

$$I_{\phi,\theta,\psi}[n] = I_{\phi,\theta,\psi}[n-1] + K_{i,\{\phi,\theta,\psi\}} C_{idyn} T_s e_{\phi,\theta,\psi}[n] \tag{52}$$

Taking the Z transform:

$$\begin{aligned} I_{\phi,\theta,\psi}(z) &= z^{-1} I_{\phi,\theta,\psi}(z) + K_{i,\{\phi,\theta,\psi\}} C_{idyn} T_s E_{\phi,\theta,\psi}(z) \\ I_{\phi,\theta,\psi}(z)(1 - z^{-1}) &= K_{i,\{\phi,\theta,\psi\}} C_{idyn} T_s E_{\phi,\theta,\psi}(z) \end{aligned} \tag{53}$$

The transfer function for I term for all 3 axis is:

$$\boxed{TF^{I}_{\phi,\theta,\psi}(z) = \frac{I_{\phi,\theta,\psi}(z)}{E_{\phi,\theta,\psi}(z)} = (K_{i,\{\phi,\theta,\psi\}} C_{idyn} T_s)\left(\frac{z}{z-1}\right)} \tag{54}$$

### 2.2.3 Derivative Control

Derivative control is only applied to the Roll and Pitch axis. The D term depends on the rate of change of error signal (equation:27). Since the D term is sensitive to the presence of high frequency noise in the error signal, Cleanflight uses two cascaded biquad filters; a notch(band-stop) and low pass filter, to further filter out the noise from the current gyro data($y_{\phi,\theta}$ in equation:27) for roll and pitch axis. The overall block diagram of the cascaded filters is as shown in figure [11].

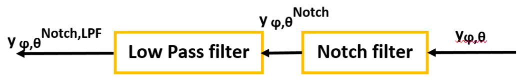


**Figure 11:** PID D term cascaded filters for roll and pitch axis gyroscope data

We now derive the transfer functions of each filter and compute the overall transfer function for the combined fourth-order filter. A discrete time biquad filter's transfer function(2 poles, 2 zeros) with normalized coefficients($a_0 = 1$) is as follows:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{55}$$

**Notch Filter** The specifications of the *notch* filter and the corresponding coefficients for the transfer function are:

- Notch Specification:
  - lower cut-off freq = $f_L$ : constant and configurable on system startup
  - center/notch freq = $f_c$ : constant and configurable on system startup
  - $w_c = 2\pi \frac{f_c}{F_s}$ : where $F_s = \frac{1}{T_s}$ is the PID loop sampling frequency.
  - upper cut-off freq = $f_H = \frac{f_c^2}{f_L}$
  - Quality factor - Q = $\frac{f_c}{f_H - f_L} = \frac{f_c}{f_{BW}}$ where $f_{BW}$ is the bandwidth of the notch and the quality factor describes how under-damped an oscillator/filter is. A higher Q value implies lower rate of energy loss, less damping and sharper=narrower notch.
  - $\alpha = \frac{sin(w_c)}{2Q}$ : intermediate variable used in coefficient calculation.
- Notch normalized coefficients:
  - $b_0 = \frac{1}{1+\alpha}$
  - $b_1 = \frac{-2cos(w_c)}{1+\alpha}$
  - $b_2 = \frac{1}{1+\alpha}$
  - $a_0 = 1$
  - $a_1 = \frac{-2cos(w_c)}{1+\alpha}$
  - $a_2 = \frac{1-\alpha}{1+\alpha}$

**Low pass Filter** Similarly the specifications of the *low pass* filter(LPF) and the corresponding coefficients for the transfer function are:

- LPF Specification:
  - cut-off/corner freq = $f_0$ : constant and configurable on system startup
  - $w_0 = 2\pi \frac{f_0}{F_s}$ : where $F_s = \frac{1}{T_s}$ is the PID loop sampling frequency.

    - $Q = \frac{1}{\sqrt{2}}$ for a Butterworth approximation (maximally flat frequency pass-band gain)
    - $\alpha = \frac{sin(w_0)}{2Q}$ : intermediate variable used in coefficient calculation.

- LPF normalized coefficients:
    - $b_0 = \frac{1-cos(w_0)}{2(1+\alpha)}$
    - $b_1 = \frac{1-cos(w_0)}{1+\alpha}$
    - $b_2 = \frac{1-cos(w_0)}{2(1+\alpha)}$
    - $a_0 = 1$
    - $a_1 = \frac{-2cos(w_0)}{1+\alpha}$
    - $a_2 = \frac{1-\alpha}{1+\alpha}$

Cleanflight implements each biquad filter in transposed direct form II topology to save on memory storage for previous input values. Figure 12 shows the topology for the notch filter. Low Pass filter would have a similar topology. The corresponding time equations and the transfer function of both the filters are as follows:

- **Notch**: Time equations:

$$\begin{aligned} y_{\phi,\theta}^{notch}[n] &= b_0 y_{\phi,\theta}[n] + s_1[n-1] \\ s_1[n] &= b_1 y_{\phi,\theta}[n] - a_1 y_{\phi,\theta}^{notch}[n] + s_2[n-1] \\ s_2[n] &= b_2 y_{\phi,\theta}[n] - a_2 y_{\phi,\theta}^{notch}[n] \end{aligned} \tag{56}$$

  where $s_1[n]$ and $s_2[n]$ are intermediate signals initially set to 0.(figure 12)

- **Notch**: Transfer Function after taking the Z transform of the time

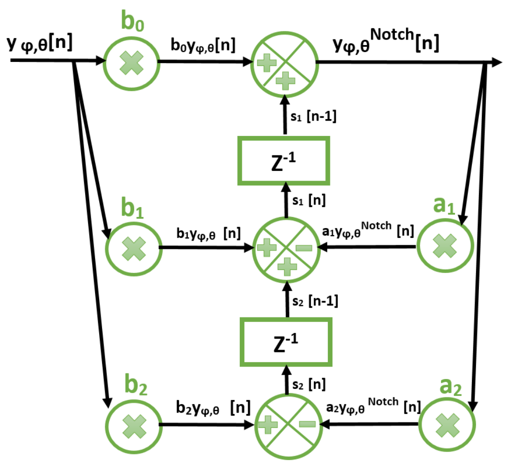


**Figure 12:** PID D term Biquad Notch filter for gyroscope data in direct form II topology

equation:

$$\begin{aligned} H_{\phi,\theta}^{notch}(z) &= \frac{Y_{\phi,\theta}^{notch}(z)}{Y_{\phi,\theta}(z)} \\ &= \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2} \\ &= \frac{\frac{1}{1+\alpha} z^2 - \frac{2cos(w_c)}{1+\alpha} z + \frac{1}{1+\alpha}}{z^2 - \frac{2cos(w_c)}{1+\alpha} z + \frac{1-\alpha}{1+\alpha}} \\ &= \boxed{\left(\frac{1}{1+\alpha}\right)\left(\frac{z^2 - 2cos(w_c)z + 1}{z^2 - \frac{2cos(w_c)}{1+\alpha} z + \frac{1-\alpha}{1+\alpha}}\right)} \end{aligned} \tag{57}$$

- **LPF**: Time equations:

$$\begin{aligned} y_{\phi,\theta}^{notch,LPF}[n] &= b_0 y_{\phi,\theta}^{notch}[n] + s_1[n-1] \\ s_1[n] &= b_1 y_{\phi,\theta}^{notch}[n] - a_1 y_{\phi,\theta}^{notch,LPF}[n] + s_2[n-1] \\ s_2[n] &= b_2 y_{\phi,\theta}^{notch}[n] - a_2 y_{\phi,\theta}^{notch,LPF}[n] \end{aligned} \tag{58}$$

- **LPF**: Transfer Function after taking the Z transform of the time equation:

$$\begin{aligned} H^{LPF}_{\phi,\theta}(z) &= \frac{Y^{notch,LPF}_{\phi,\theta}(z)}{Y^{notch}_{\phi,\theta}(z)} \\ &= \frac{b_0z^2 + b_1z + b_2}{z^2 + a_1z + a_2} \\ &= \frac{\frac{1-cos(w_0)}{2(1+\alpha)}z^2 + \frac{1-cos(w_0)}{1+\alpha}z + \frac{1-cos(w_0)}{2(1+\alpha)}}{z^2 - \frac{2cos(w_0)}{1+\alpha}z + \frac{1-\alpha}{1+\alpha}} \\ &= \boxed{\left(\frac{1-cos(w_0)}{2(1+\alpha)}\right)\left(\frac{z^2+2z+1}{z^2 - \frac{2cos(w_0)}{1+\alpha}z + \frac{1-\alpha}{1+\alpha}}\right)} \end{aligned} \tag{59}$$

The final transfer function of the cascaded filters is:

$$\boxed{\begin{aligned} H^{notch,LPF}_{\phi,\theta} &= \frac{Y^{notch,LPF}_{\phi,\theta}(z)}{Y_{\phi,\theta}(z)} \\ &= H^{notch}_{\phi,\theta}(z) \times H^{LPF}_{\phi,\theta}(z) \end{aligned}} \tag{60}$$

The D term in discrete time is represented as a difference equation which is an approximation to continuous time differential equation. A generalized continuous time differential term is presented in equation:(61).

$$D(t) = K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{61}$$

The discrete time equivalent, as implemented for Cleanflight PID controller is:

$$\begin{aligned} D_{\phi,\theta}[n] &= C_{tpa}K_{d,\{\phi,\theta\}}\frac{(e_{\phi,\theta}[n] - e_{\phi,\theta}[n-1])}{\Delta t} \\ &= C_{tpa}K_{d,\{\phi,\theta\}}\frac{(e_{\phi,\theta}[n] - e_{\phi,\theta}[n-1])}{T_s} \end{aligned} \tag{62}$$

where

- $C_{tpa}$ = TPA(Throttle PID Attenuation) factor = constant if applied throttle < TPA Threshold - same as in proportional term (equation:36)
- $K_{d,\{\phi,\theta\}}$ = derivative gain for roll and pitch axis

- $e_{\phi,\theta}[n] = C_{Ddyn} r_{\phi,\theta} - y_{\phi,\theta}^{notch,LPF}[n]$ = error signal at time step n for roll and pitch axis, $C_{Ddyn}$ = constant set point weight for the derivative term(equation:35). This is set to zero in ANGLE MODE $\implies e_{\phi,\theta}[n] = -y_{\phi,\theta}^{notch,LPF}[n]$
- $T_s$ = sampling time period

Taking Z transform of the equation yields:

$$D_{\phi,\theta}(z) = \left(\frac{C_{tpa} K_{d,\{\phi,\theta\}}}{T_s}\right)\left(E_{\phi,\theta}(z) - z^{-1} E_{\phi,\theta}(z)\right) \tag{63}$$

The transfer function for D term for roll and pitch axis is:

$$\boxed{TF_{\phi,\theta}^{D}(z) = \frac{D_{\phi,\theta}(z)}{E_{\phi,\theta}(z)} = \left(\frac{C_{tpa} K_{d,\{\phi,\theta\}}}{T_s}\right)\left(\frac{z-1}{z}\right)} \tag{64}$$

where:

$$\begin{aligned} E_{\phi,\theta}(z) &= C_{Ddyn} R_{\phi,\theta}(z) - Y_{\phi,\theta}^{notch,LPF}(z) \\ &= C_{Ddyn} R_{\phi,\theta}(z) - \left(H_{\phi,\theta}^{notch,LPF}(z) \times Y_{\phi,\theta}(z)\right) \end{aligned} \tag{65}$$

## 2.3 PID Attitude Controller Transfer function

The overall PID transfer function for roll, pitch and yaw from equation:33 & 34 is reproduced below for reference:

- Roll($\phi$) & Pitch($\theta$)

$$TF_{\phi,\theta}^{PID}(z) = \frac{U_{\phi,\theta}(z)}{E_{\phi,\theta}^{rate}(z)} \tag{66}$$

$$= TF_{\phi,\theta}^{P}(z) + TF_{\phi,\theta}^{I}(z) + TF_{\phi,\theta}^{D}(z) \tag{67}$$

$$\begin{aligned} &= C_{tpa} K_{p,\{\phi,\theta\}} && + \\ &\left(K_{i,\{\phi,\theta,\psi\}} C_{idyn} T_s\right)\left(\frac{z}{z-1}\right) && + \\ &\left(\frac{C_{tpa} K_{d,\{\phi,\theta\}}}{T_s}\right)\left(\frac{z-1}{z}\right) \end{aligned} \tag{68}$$

- Yaw($\psi$)

$$
\begin{aligned}
TF_{\psi}^{PID}(z) &= \frac{U_{\psi}(z)}{E_{\psi}^{rate}(z)} && (69)\\
&= TF_{\psi}^{P}(z) + TF_{\psi}^{I}(z) && (70)\\
&= \Big(C_{tpa} K_{p,\psi} C_{LPF}\Big) \frac{z}{z + (C_{LPF} + 1)} \qquad + \\
\Big(K_{i,\{\phi,\theta,\psi\}} C_{idyn} T_s\Big)&\left(\frac{z}{z-1}\right) && (71)
\end{aligned}
$$

## 3 Sub-System 3: Quadcopter Mixer X

This section presents detailed time and frequency domain analysis for the mixer(plant) in the feedforward loop that takes its input from the PID controller's output($u_{\phi,\theta,\psi}$) for each axis, mixes them according to the predefined mixing configuration constants(depending on the frame type used), adds the throttle input($r_f[n]$) uniformly and feeds the output PWM signal to each of the 4 electronic speed controllers(ESCs) of the quadcopter.

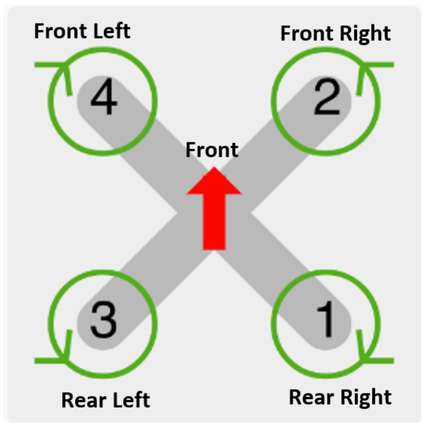


**Figure 13:** QuadX motor configuration and direction of rotation

The most popular motor mixing configuration is an "X" configuration where the four motors on the arms of the quadcopter are mounted in an "X" formation(figure: 13). The mixing constants for a QuadX as defined in Cleanflight are presented in table:1.

| **Motor Number** | **Motor Position** | **Motor Direction** | **Throttle($f$)** | **Roll($\phi$)** | **Pitch($\theta$)** | **Yaw($\psi$)** |
|---|---|---|---|---|---|---|
| ① | Rear Right | CW | 1 | -1 | 1 | -1 |
| ② | Front Right | ACW | 1 | -1 | -1 | 1 |
| ③ | Rear Left | ACW | 1 | 1 | 1 | 1 |
| ④ | Front Left | CW | 1 | 1 | -1 | -1 |

**Table 1:** QuadX mixing constants

The time equation for the mixer for each motor, $i$ where $i \in \{1, 2, 3, 4\}$

is:

$$y_i[n] = C_{mix}\begin{pmatrix} C_{u_\phi}u_\phi[n]\times m_{\{i,\phi\}} & + \\ C_{u_\theta}u_\theta[n]\times m_{\{i,\theta\}} & + \\ C_{u_\psi}u_\psi[n]\times m_{\{i,\psi\}} & \end{pmatrix} + \left(C_f r_f[n]\times m_{\{i,f\}}\right) \tag{72}$$

where

- $y_i[n]$ = PWM motor output signal fed to each motor, i.
- $C_{mix}$ = scaling constant for the entire mix sum to restrict the range of the mix
- $C_{u_{\{\phi,\theta,\psi\}}}$ = scaling constant for each axis's pid output
- $u[n]$ pid output for each axis
- $m$ = mixing constant from table:1
- $C_f$ = throttle scaling factor
- $r_f[n]$ = throttle input from the receiver

Note for this system, there are 4 time dependent inputs to the mixer i.e. 1. $r_f[n]$: throttle input from the receiver, 2. $u_\phi[n]$: control output from the roll axis PID controller, 3. $u_\theta[n]$ control output from the pitch axis PID controller and 4. $u_\psi[n]$ control output from the yaw axis PID controller. When we take the z-transform of the above mixing equation:

$$Y_i(z) = C_{mix}\begin{pmatrix} C_{u_\phi}U_\phi(z)\times m_{\{i,\phi\}} & + \\ C_{u_\theta}U_\theta(z)\times m_{\{i,\theta\}} & + \\ C_{u_\psi}U_\psi(z)\times m_{\{i,\psi\}} & \end{pmatrix} + \left(C_f R_f(z)\times m_{\{i,f\}}\right) \tag{73}$$

The above equation implies that the mixer plant is composed of 4 sub-plants, the output of which is summed up to get the final PWM motor output. Figure: 14 gives a pictorial representation of the mixer system of equation: 73.

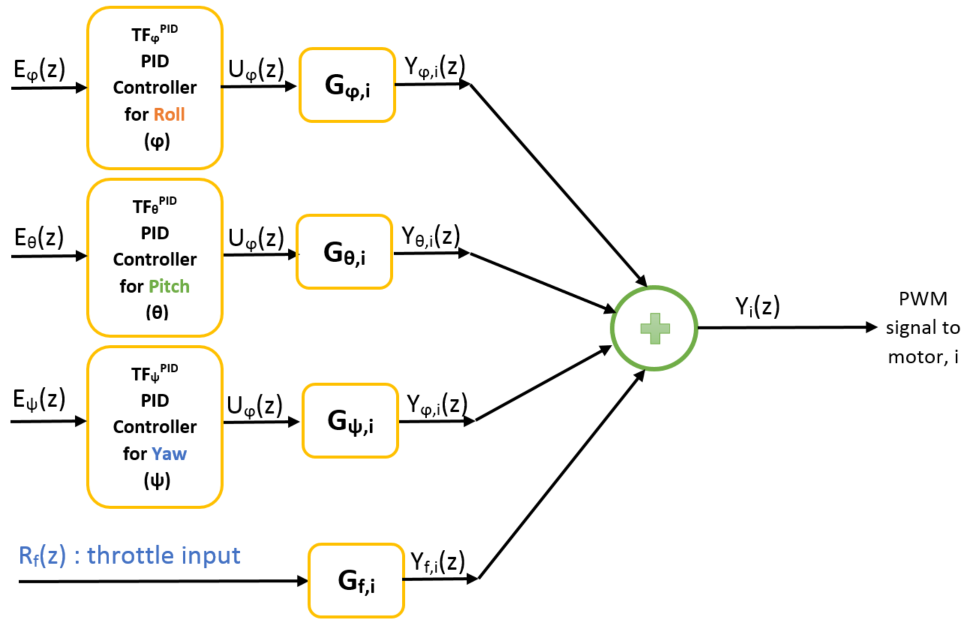


**Figure 14:** CF Quad-X mixer system for a PWM motor output for one motor

From the diagram, the transfer function for each sub-plant becomes:

$$G_{\{\phi,i\}}(z) = \frac{Y_{\{\phi,i\}}(z)}{U_\phi(z)} = C_{mix}C_{u_\phi}m_{\{i,\phi\}} \tag{74a}$$

$$G_{\{\theta,i\}}(z) = \frac{Y_{\{\theta,i\}}(z)}{U_\theta(z)} = C_{mix}C_{u_\theta}m_{\{i,\theta\}} \tag{74b}$$

$$G_{\{\psi,i\}}(z) = \frac{Y_{\{\psi,i\}}(z)}{U_\psi(z)} = C_{mix}C_{u_\psi}m_{\{i,\psi\}} \tag{74c}$$

$$G_{\{f,i\}}(z) = \frac{Y_{\{f,i\}}(z)}{R_f(z)} = C_f m_{\{i,f\}} \tag{74d}$$

Thus the mixer output equation becomes:

$$\begin{aligned} Y_i(z) = \ & G_{\{\phi,i\}}(z)U_\phi(z) + && (75a)\\ & G_{\{\theta,i\}}(z)U_\theta(z) + && (75b)\\ & G_{\{\psi,i\}}(z)U_\psi(z) + && (75c)\\ & G_{\{f,i\}}(z)R_f(z) && (75d)\\ = \ & Y_{\{\phi,i\}} + Y_{\{\theta,i\}} + Y_{\{\psi,i\}} + Y_{\{f,i\}} && (75e) \end{aligned}$$

# 4 Sub-System 4: Sensor Processing

## 4.1 Accelerometer

The accelerometer measures static and dynamic acceleration forces in g units in each of the three axis; x, y and z of the sensor's frame i.e $a[n] = \big(a_x[n], a_y[n], a_z[n]\big)$. Measurement of the static forces like gravity allows the flight controller to compute the angle of inclination of the sensor frame with respect to the earth frame. Details of how this is achieved by Cleanflight is discussed in the next section(sec:5). This section considers processing of the raw data from the accelerometer before it is input to the AHRS subsystem. To be consistent with Euler angle notation used in the previous sections, we will consider x, y and z axis of the sensor frame to correspond to the roll($\phi$), pitch($\theta$) and yaw($\psi$) axis respectively. Cleanflight samples raw accelerometer readings at a constant sampling rate ($F_s^{acc}$) and filters high frequency noise from the measured signals by passing them through a biquad low pass filter(LPF) as can be seen in figure 15.

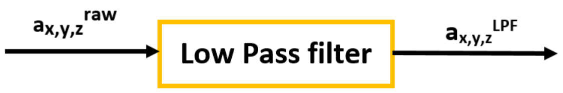


**Figure 15:** Low pass filter for measured raw accelerometer data for all three axes.

The low pass filter is a second order biquad filter similar to the one presented in section 2.2.3. Equation 55 (reproduced below) is the discrete time biquad filter's transfer function with normalized coefficients($a_0 = 1$).

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{76}$$

The specifications of the LPF and the corresponding coefficients are:

- LPF Specifications:
  - cut-off/corner freq $= f_0 = 10Hz$ : constant and configurable on system startup

- $w_0 = 2\pi \frac{f_0}{F_s^{acc}}$ : where $F_s^{acc} = \frac{1}{T_s^{acc}}$ is the accelerometer sampling frequency derived from gyroscope's target loop time. $w_0$ represents the product of the radian cut-off frequency and the sampling time period($T_s^{acc}$) of the accelerometer.
- $Q = \frac{1}{\sqrt{2}}$ for a Butterworth filter approximation with maximally flat pass-band frequency gain
- $\alpha = \frac{sin(w_0)}{2Q}$ : intermediate variable used in coefficients' calculation.

- For the LPF normalized coefficient ($a_1, a_2, b_0, b_1$ *and* $b_2$) equations, please refer to section 2.2.3 under the header: "Low pass Filter".

The filter has similar biquad topology as the Notch and Low pass Biquad filters presented in section 2.2.3. The time equations are:

$$\begin{aligned} a_{x,y,z}^{LPF}[n] &= b_0 a_{x,y,z}^{raw}[n] + s_1[n-1] \\ s_1[n] &= b_1 a_{x,y,z}^{raw}[n] - a_1 a_{x,y,z}^{LPF}[n] + s_2[n-1] \\ s_2[n] &= b_2 a_{x,y,z}^{raw}[n] - a_2 a_{x,y,z}^{LPF}[n] \end{aligned} \tag{77}$$

The transfer function of the LPF filter is:

$$\begin{aligned} H_{acc}^{LPF}(z) &= \frac{A_{x,y,z}^{LPF}(z)}{A_{x,y,z}^{raw}(z)} \\ &= \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2} \\ &= \frac{\frac{1-cos(w_0)}{2(1+\alpha)} z^2 + \frac{1-cos(w_0)}{1+\alpha} z + \frac{1-cos(w_0)}{2(1+\alpha)}}{z^2 - \frac{2cos(w_0)}{1+\alpha} z + \frac{1-\alpha}{1+\alpha}} \\ &= \boxed{\left(\frac{1-cos(w_0)}{2(1+\alpha)}\right)\left(\frac{z^2+2z+1}{z^2 - \frac{2cos(w_0)}{1+\alpha} z + \frac{1-\alpha}{1+\alpha}}\right)} \end{aligned} \tag{78}$$

The filtered readings ($a_{x,y,z}^{LPF}$) are then manipulated to align with the flight controller hardware's forward orientation ($a_{x,y,z}^{LPF,align}$) as shown in the diagram:

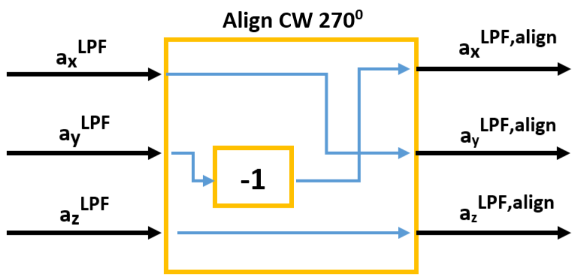


**Figure 16:** Accelerometer Alignment.

Since changing the alignment has no effect on the frequency response other than changing the signs and rerouting the input signals, we consider the transfer functions for the measured signals in each of the three axes as having a constant gain of 1 or -1 as shown in the following z domain frequency response equations:

$$H_x^{align} = \frac{A_x^{LPF,align}(z)}{A_y^{LPF}(z)} = -1 \tag{79a}$$

$$H_y^{align} = \frac{A_y^{LPF,align}(z)}{A_x^{LPF}(z)} = 1 \tag{79b}$$

$$H_z^{align} = \frac{A_z^{LPF,align}(z)}{A_z^{LPF}(z)} = 1 \tag{79c}$$

The aligned accelerometer values are then compared with the reference trim values. A difference or error($e_{x,y,z}^{acc} = a_{x,y,z}^{LPF,align} - a_{x,y,z}^{trim}$) is calculated and fed to the moving average accumulator(refer to figure 17).

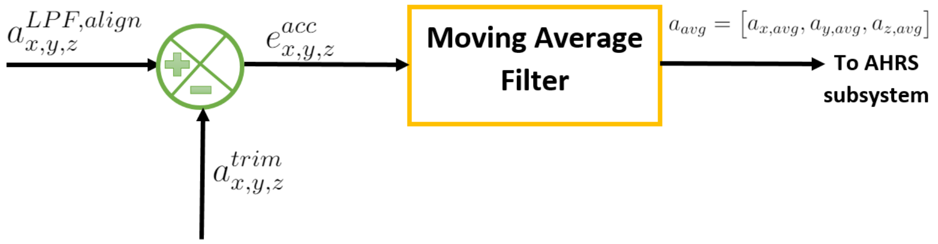


**Figure 17:** Accelerometer trimming and smoothing filter.

The trim value for each axis($a_{x,y,z}^{trim}$) is calculated during the accelerometer's configuration phase at the start of the system as an average value of a certain number of sampled readings for that axis. In the configuration phase, the accelerometer is held in a horizontal still position, parallel to the ground such that the sensor frame is completely aligned with the earth frame. This orientation ensures that the accelerometer measures the reference field of gravity in the earth frame. Trim values thus provide reference acceleration forces in each axis.
The difference between measured in-flight values and the pre-calculated trim values gives the increase/decrease in accelerometer values from the reference values. The error is then smoothed out or averaged over a fixed number of samples($N$) before being fed to the AHRS subsystem(sec:5). AHRS module then determines the drone's angle of inclination in real time.
The moving average filter is a simple low-pass finite impulse response(FIR) filter. The time equations for the filter are:

$$\begin{aligned} a_{x,avg}[n] &= \frac{1}{N}\sum_{k=0}^{N-1} e_x^{acc}[n-k] \\ a_{y,avg}[n] &= \frac{1}{N}\sum_{k=0}^{N-1} e_y^{acc}[n-k] \\ a_{z,avg}[n] &= \frac{1}{N}\sum_{k=0}^{N-1} e_z^{acc}[n-k] \end{aligned} \tag{80}$$

where

$$N = \frac{\text{ACCELEROMETER } \textit{task rate}}{\text{ATTITUDE } \textit{task rate}} \tag{81}$$

Taking the corresponding Z transforms of the above equations yield:

$$\begin{aligned} A_{x.avg}(z) &= E_x^{acc}(z)\left(\frac{1}{N}\sum_{k=0}^{N-1} z^{-k}\right) \\ A_{y.avg}(z) &= E_y^{acc}(z)\left(\frac{1}{N}\sum_{k=0}^{N-1} z^{-k}\right) \\ A_{z.avg}(z) &= E_z^{acc}(z)\left(\frac{1}{N}\sum_{k=0}^{N-1} z^{-k}\right) \end{aligned} \tag{82}$$

The transfer function of the moving average filter thus becomes:

$$H_{avg}(z) = \frac{A_{avg}(z)}{E^{acc}_{x,y,z}(z)} = \frac{1}{N}\sum_{k=0}^{N-1} z^{-k} \tag{83}$$

In summary, the overall control flow of accelerometer processing is shown in figure 18 where each sub-component's transfer function, calculated according to the forgoing equations in this section, is shown inside the yellow blocks.

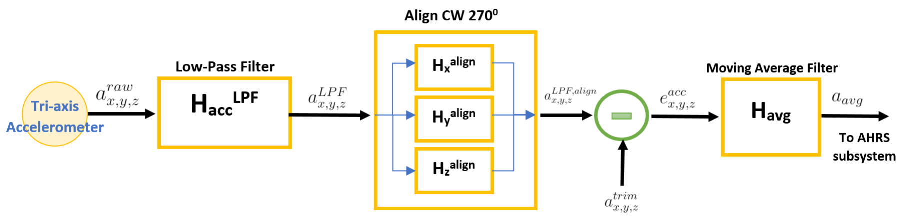


**Figure 18:** Accelerometer Data Processing.

## 4.2 Magnetometer

The magnetometer measures the direction and magnitude of Earth's magnetic fields in each of the three axes of the sensor frame; $m[n] = \big(m_x[n], m_y[n], m_z[n]\big)$, in units of gauss(Ga) (1 gauss = 100$\mu$ Tesla). Measurement of Earth's Magnetic North provides an absolute reference field in addition to the gravitational field measured by the accelerometer(refer to section: 4.1), to accurately determine the orientation of the sensor in the earth's frame. Detailed analysis of the sensor fusion and attitude estimation algorithm is presented in the next section(sec:5).
Cleanflight samples raw magnetometer data($m^{raw}_{x,y,z}$) at a constant sampling rate($F^{mag}_s$ = COMPASS task execution rate) and then aligns the raw data in the sensor frame to the quadcopter's frame, via a clockwise rotation of 270°. This alignment is similar to the accelerometer data alignment shown in figure 16. Finally the aligned values($m^{align}_{x,y,z}$) are compared with the reference field values($m^{ref}_{x,y,z}$) that are determined during the configuration phase at system startup. The difference between the measured and the reference values($m^{error}_{x,y,z} = m^{align}_{x,y,z} - m^{ref}_{x,y,z}$) gives the error or offset in the magnetometer readings which is then fed to the AHRS sub-system as input. To calculate

the reference field values($m^{ref}_{x,y,z}$), Cleanflight allows 30s of configuration time, by the end of which, the midrange(mean of the highest and lowest measured values) of the minimum and the maximum values, sampled for each axis during configuration, is calculated. Control flow of the magnetometer data processing is presented in the following figure(fig:19). The transfer functions for the alignment module are:

$$H^{align}_x = \frac{M^{align}_x(z)}{M^{raw}_y(z)} = -1 \tag{84a}$$

$$H^{align}_y = \frac{M^{align}_y(z)}{M^{raw}_x(z)} = 1 \tag{84b}$$

$$H^{align}_z = \frac{M^{align}_z(z)}{M^{raw}_z(z)} = 1 \tag{84c}$$

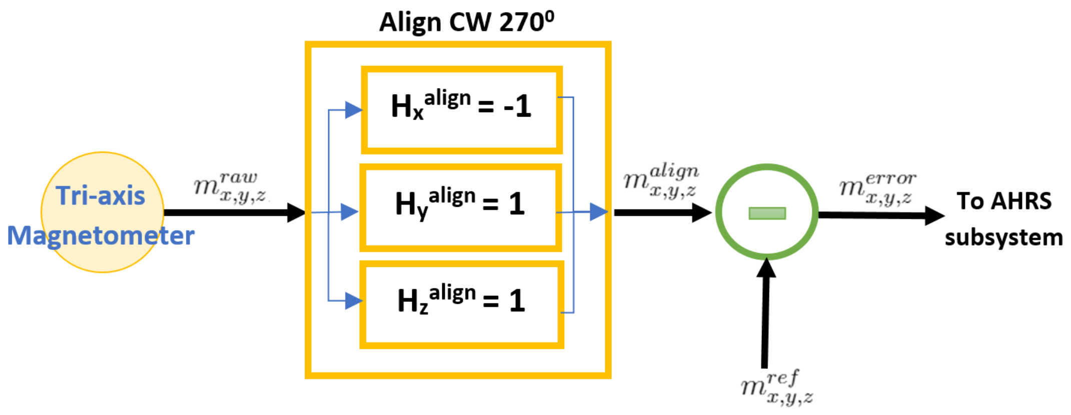


**Figure 19:** Magnetometer Data Processing.

## 4.3 Gyroscope

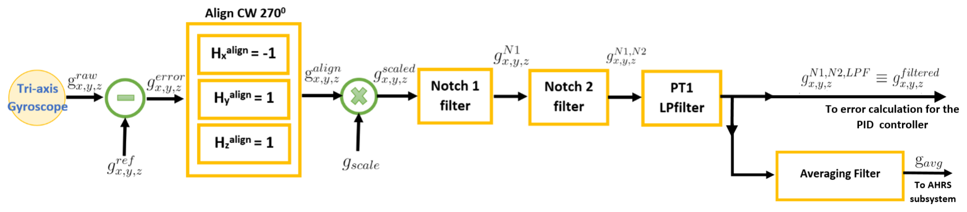


**Figure 20:** Gyroscope Data Processing.

Figure 20 gives a high level diagrammatic overview of the control flow of gyroscope's data in the processing pipeline of the sensor. Gyroscope measures the angular velocity of the quadcopter in each of the three axis of the sensor frame($g[n] = \big(g_x[n], g_y[n], g_z[n]\big)$) in units of degrees per second(°/s). Cleanflight samples raw gyroscope data($g_{x,y,z}^{raw}$) at a sampling frequency of $F_s^{gyro} = \frac{1}{T_s^{gyro}} = \frac{1}{GyroLoopTime}$.The maximum sampling rate for the MPU6050 gyroscope, as stated in the datasheet, is 8kHz. Cleanflight allows further division of the sampling rate and calculates the actual sampling period as:

$$T_s^{gyro} = GyroLoopTime = (gyro\_sync\_denom) \times (max\_sampling\_period) \tag{85}$$

where *gyro_sync_denom* is a configurable parameter. An angular velocity offset(error) is determined by subtracting the zero reference values in each axis from the raw measurements in the corresponding axis i.e $g_{x,y,z}^{error} = g_{x,y,z}^{raw} - g_{x,y,z}^{ref}$. Reference values are determined during the gyro calibration phase and represent an average of raw sampled values over a time period of 3s after system startup. The sensor must be held relatively still during the calibration phase. The offset values($g_{x,y,z}^{error}$) in the sensor frame are then aligned with the quadcopter frame via a clockwise rotation of 270°in the same way as alignment was done for the accelerometer(figure:16) and magnetometer readings. Transfer functions of the alignment module are:

$$H_x^{align} = \frac{G_x^{align}(z)}{G_y^{error}(z)} = -1 \tag{86a}$$

$$H_y^{align} = \frac{G_y^{align}(z)}{G_x^{error}(z)} = 1 \tag{86b}$$

$$H_z^{align} = \frac{G_z^{align}(z)}{G_z^{error}(z)} = 1 \tag{86c}$$

The aligned values are subsequently scaled with gyroscope's sensitivity parameter ($\frac{1}{16.4\ lsb/dps} \equiv \frac{1}{16.4\ counts/^\circ/s}$) to convert the digital representation(lsb) to units of angular velocity(°/s) and then fed to the first notch filter. The scale factor is thus represented as constant gain: $H_{scale}$ in the frequency domain.

$$\boxed{H_{scale} = \frac{G_{x,y,z}^{scaled}}{G_{x,y,z}^{align}}} \tag{87}$$

Gyroscope data is always susceptible to unwanted interference from low frequency motor vibrations that add themselves as noise to the angular velocity measurements and thus need to be filtered out. For this purpose Cleanflight implements two cascaded biquad notch filters and a PT1 low pass filter to clean the data signal from the noise on each axis. The general form of a biquad($2^{nd}$ order) filter was presented in equation 55 in subsection 2.2.3. The specifications of the notch and low-pass PT1 filter and the expressions for the coefficients($a_1, a_2, b_0, b_1$ & $b_2$) of those filters were discussed in detail and presented for the PID controller module in section 2. Gyro filters use the same specifications, the only difference being in the cut-off frequency/corner frequency and the center notch frequency for each filter(figure 20). The time equations and corresponding z domain transfer functions for each filter are:

- Notch 1 Filter:
  - Time Equation:

$$\begin{aligned} g_{x,y,z}^{N1}[n] &= b_0 g_{x,y,z}^{scaled}[n] + s_1[n-1] \\ s_1[n] &= b_1 g_{x,y,z}^{scaled}[n] - a_1 g_{x,y,z}^{N1}[n] + s_2[n-1] \\ s_2[n] &= b_2 g_{x,y,z}^{scaled}[n] - a_2 g_{x,y,z}^{N1}[n] \end{aligned} \tag{88}$$

– Z-domain:

$$\begin{aligned}
H^{N1}_{gyro}(z) &= \frac{G^{N1}_{x,y,z}(z)}{G^{scaled}_{x,y,z}(z)}\\
&= \frac{b_{01}z^2 + b_{11}z + b_{21}}{z^{21} + a_{11}z + a_{21}}\\
&= \frac{\frac{1}{1+\alpha_1}z^2 - \frac{2cos(w_{c1})}{1+\alpha_1}z + \frac{1}{1+\alpha_1}}{z^2 - \frac{2cos(w_{c1})}{1+\alpha_1}z + \frac{1-\alpha_1}{1+\alpha_1}}\\
&= \boxed{\left(\frac{1}{1+\alpha_1}\right)\left(\frac{z^2 - 2cos(w_{c1})z + 1}{z^2 - \frac{2cos(w_{c1})}{1+\alpha_1}z + \frac{1-\alpha_1}{1+\alpha_1}}\right)}
\end{aligned} \tag{89}$$

where:

* $w_{c1} = 2\pi\frac{f_{c1}}{F^{gyro}_s}$ : where $F^{gyro}_s = \frac{1}{T^{gyro}_s}$ and $f_{c1}$ is the configurable center/notch frequency
* $\alpha_1 = \frac{sin(w_{c1})}{2Q_1}$ : where $Q_1$ is the quality factor for the notch 1 filter

• Notch 2 Filter: The time equation($g^{N1,N2}_{x,y,z}$) and the transfer function equation($H^{N2}_{gyro} = \frac{G^{N1,N2}_{x,y,z}(z)}{G^{N1}_{x,y,z}}$) are the same as Notch 1 with different parameters that are specific to Notch 2:

– Notch 2 Parameters:

* $w_{c2} = 2\pi\frac{f_{c2}}{F^{gyro}_s}$
* $\alpha_2 = \frac{sin(w_{c2})}{2Q_2}$ where $Q_2$ is the quality factor for the notch 2 filter

• PT1 Low Pass Filter:

– Time Equation:

$$\begin{aligned}
g^{N1,N2,LPF}_{x,y,z}[n] &= \left\{C_{LPF}(g^{N1,N2}_{x,y,z}[n] - g^{N1,N2,LPF}_{x,y,z}[n-1])\right\} + g^{N1,N2,LPF}_{x,y,z}[n-1]\\
&= C_{LPF}g^{N1,N2}_{x,y,z}[n] - (C_{LPF}+1)g^{N1,N2,LPF}_{x,y,z}[n-1]
\end{aligned} \tag{90}$$

where $C_{LPF}$ is a constant term $= \frac{T^{gyro}_s}{RC+T^{gyro}_s}$ and $RC = \frac{1}{2\pi f_{cut-off}}$ is the filter time constant. $f_{cut-off}$ is the configurable cutoff frequency for the low pass filter.

– Z domain: Simplified transfer function of the PT1 low pass filter:

$$\boxed{H_{gyro}^{LPF}(z) = \frac{G_{x,y,z}^{N1,N2,LPF}(z)}{G_{x,y,z}^{N1,N2}(z)} = \frac{C_{LPF}z}{z + (C_{LPF} + 1)}} \tag{91}$$

The transfer function of the combined filters thus becomes:

$$\boxed{H_{gyro}^{filter}(z) = H_{gyro}^{N1}(z) \times H_{gyro}^{N2}(z) \times H_{gyro}^{LPF}(z)} \tag{92}$$

The filtered gyro readings($g_{x,y,z}^{N1,N2,LPF} \equiv g_{x,y,z}^{filtered}$) are output directly from the gyro processing pipeline in the feedback path of the overall flight controller system(figure: 1), to calculate the error rate from the set-point rate. This error is then fed to the PID controller in the feed forward path.
The filtered readings are also numerically integrated within the gyro processing pipeline, in a separate branch and passed through a time averaging filter before being fed to the AHRS sub-system(refer to the system diagram in Figure:1). The AHRS subsystem(presented in the next section) makes use of the average gyroscope readings in each axis to determine the attitude of the quadcopter with respect to the earth's frame of reference.
The number of accumulated, filtered gyro samples; N, for the average, depends on the ratio of the GYRO update task rate to the ATTITUDE task rate. Cleanflight approximates integration of the filtered gyroscope readings with the trapezoidal rule. The time equations are:

$$g_{avg}[n] = \frac{g_{x,y,z}^{sum}[n]}{NT_s^{gyro}} \tag{93a}$$

*where*

$$g_{x,y,z}^{sum}[n] = g_{x,y,z}^{sum}[n-1] + \frac{1}{2}\Big(g_{x,y,z}^{filtered}[n] + g_{x,y,z}^{filtered}[n-1]\Big)T_s^{gyro} \tag{93b}$$

Taking the corresponding Z transform of the equations yield:

$$G_{avg}(z) = \left(\frac{1}{NT_s^{gyro}}\right)G_{x,y,z}^{sum}(z) \tag{94a}$$

*and*

$$G_{x,y,z}^{sum}(z) = \left(\frac{T_s^{gyro}}{2}\right)\left(\frac{z+1}{z-1}\right)G_{x,y,z}^{filtered} \tag{94b}$$

Representing $G_{avg}(z)$ in terms of $G^{filtered}_{x,y,z}$:

$$G_{avg}(z) = \left(\frac{1}{2N}\right)\left(\frac{z+1}{z-1}\right)G^{filtered}_{x,y,z} \tag{95}$$

The transfer function of the averaging filter thus becomes:

$$\begin{aligned} H^{avg}_{gyro} &= \frac{G_{avg}(z)}{G^{filtered}_{x,y,z}} \\ &= \boxed{\left(\frac{1}{2N}\right)\left(\frac{z+1}{z-1}\right)} \end{aligned} \tag{96}$$

In summary the control flow of the gyro data processing with all the corresponding sub-component transfer functions are shown in the following figure:

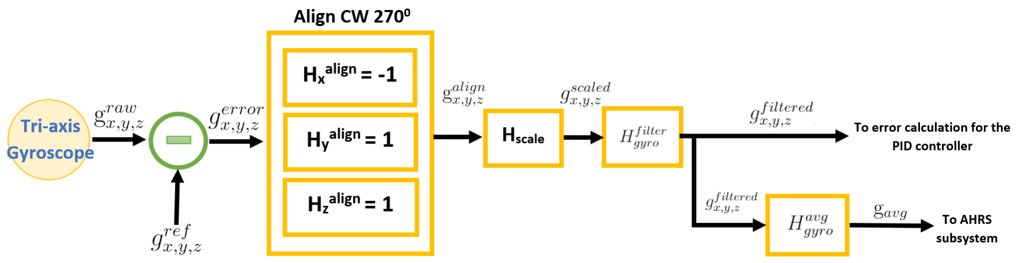


**Figure 21:** Gyroscope Data Processing with sub-component transfer functions presented as gains in yellow blocks.

# 5 Sub-System 5: Attitude and Heading Reference System

Cleanflight's state estimation theorem accurately determines the current attitude(state/orientation) of the quadcopter with respect to the Earth's frame of reference($E$) using measured sensor data that is sampled in the sensor frame($S$)(the system diagram in Figure:1 shows the state estimation module in the feedback path). Raw data is obtained from the MARG(Magnetic, Angular rate and Gravity) sensor array that includes the IMU(tri-axis gyroscope + tri-axis accelerometer) and the tri-axis magnetometer. The data is then processed and averaged over a certain number of sampling periods (exact number of samples depends on the ratio between the rate of execution of the ATTITUDE task and the sampling rate of the respective sensor task i.e ACCELEROMETER, GYRO, and COMPASS) before it is input to the Attitude and Heading Reference sub-system(AHRS). Figure 22 represents the high-level diagram of the AHRS sub-system.

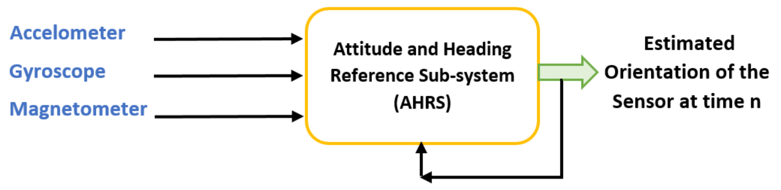

Accelometer
Gyroscope
Magnetometer
Attitude and Heading Reference Sub-system (AHRS)
Estimated Orientation of the Sensor at time n

**Figure 22:** Attitude and Heading Reference Sub-system of Cleanflight with measured inputs from the MARG sensor array

Equations 97, 98 & 99 represent the corresponding tri-axis components of the sampled input vectors(bold face) from the 3 sensors at time step n.

- Accelerometer Input:

$$^{S}\hat{\mathbf{a}}_{avg}[n] = [a_x, a_y, a_z] \quad \textit{where} \quad ^{S}\hat{\mathbf{a}}_{avg}[n] = \frac{\mathbf{a}_{avg}[n]}{\sqrt{a_x^2 + a_y^2 + a_z^2}} \tag{97}$$

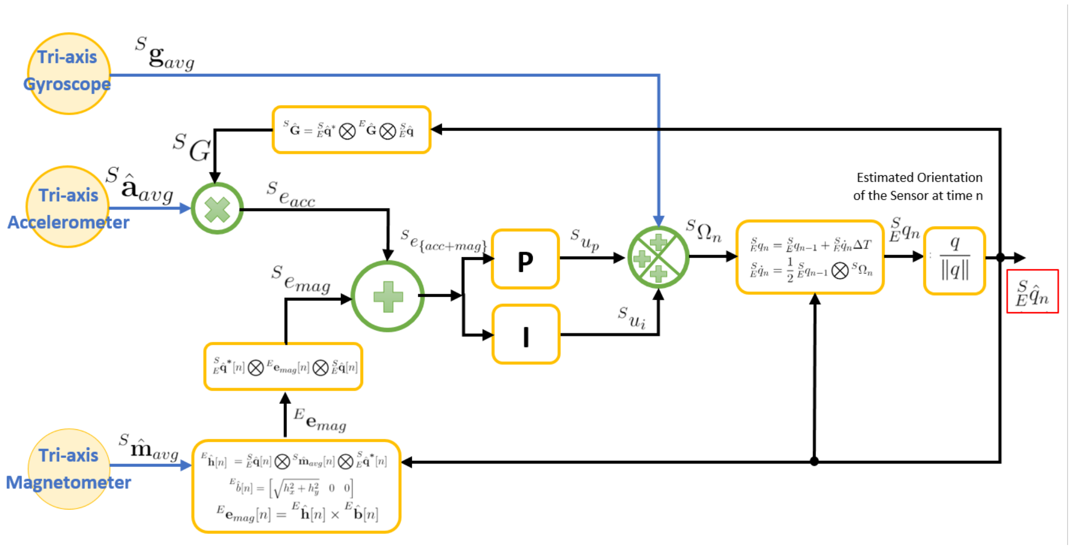
- Magnetometer Input:

$$^{S}\hat{\mathbf{m}}[n] = [m_x, m_y, m_z] \quad \textit{where} \quad ^{S}\hat{\mathbf{m}}[n] = \frac{\mathbf{m}[n]}{\sqrt{m_x^2 + m_y^2 + m_z^2}} \tag{98}$$

- Gyroscope Input:
$$^{S}\mathbf{g}_{avg}[n] = [g_x, g_y, g_z] \tag{99}$$

Note:

- The leading superscript $(S)$ denotes the frame, measurements are with reference to i.e. the sensor frame in which the sensor measurements were sampled.
- The hat symbol: $\hat{}$ denotes a normalized vector. Accelerometer and magnetometer values are normalized with their corresponding magnitudes.
- The vector subscript: *avg* represents averaged values over a fixed number of samples for each axis. The subscript is omitted for each individual vector component for a neater representation.



Tri-axis Gyroscope
$^{S}\mathbf{g}_{avg}$
$^{S}\hat{\mathbf{G}} = {}^{S}_{E}\hat{\mathbf{q}}^{*} \otimes {}^{E}\hat{\mathbf{G}} \otimes {}^{S}_{E}\hat{\mathbf{q}}$
$^{S}G$
Tri-axis Accelerometer
$^{S}\hat{\mathbf{a}}_{avg}$
$^{S}e_{acc}$
$^{S}e_{\{acc+mag\}}$
P
I
$^{S}u_p$
$^{S}u_i$
$^{S}\Omega_n$
$^{S}_{E}q_n = {}^{S}_{E}q_{n-1} + {}^{S}_{E}\dot{q}_n \Delta T$
$^{S}_{E}\dot{q}_n = \frac{1}{2}\, {}^{S}_{E}q_{n-1} \otimes {}^{S}\Omega_n$
$^{S}_{E}q_n$
$\frac{q}{\|q\|}$
Estimated Orientation of the Sensor at time n
$^{S}_{E}\hat{q}_n$
$^{S}e_{mag}$
$^{S}_{E}\hat{\mathbf{q}}^{*}[n] \otimes {}^{E}\mathbf{e}_{mag}[n] \otimes {}^{S}_{E}\hat{\mathbf{q}}[n]$
$^{E}\mathbf{e}_{mag}$
Tri-axis Magnetometer
$^{S}\hat{\mathbf{m}}_{avg}$
$^{E}\hat{\mathbf{h}}[n] = {}^{S}_{E}\hat{\mathbf{q}}[n] \otimes {}^{S}\hat{\mathbf{m}}_{avg}[n] \otimes {}^{S}_{E}\hat{\mathbf{q}}^{*}[n]$
$^{E}\hat{b}[n] = \left[\sqrt{h_x^2 + h_y^2} \quad 0 \quad 0\right]$
$^{E}\mathbf{e}_{mag}[n] = {}^{E}\hat{\mathbf{h}}[n] \times {}^{E}\hat{\mathbf{b}}[n]$

**Figure 23:** AHRS subsystem for Cleanflight:Implementation of Madgewick & Mahony's sensor fusion algorithm

The AHRS sub-system (figure:23) in CF implements *Mahony's Nonlinear Direct Complementary Filter Algorithm*. The algorithm performs low-pass filtering on a low-frequency estimate obtained from the accelerometer and magnetometer measurements, and high-pass filtering on a high-frequency estimate obtained from integration of gyroscope(angular velocity) measurements, and optimally fuses the two to obtain an all-frequency attitude estimate of the sensor [4]. The filter uses earth's gravitational and magnetic fields to provide an absolute reference of the sensor's orientation. An error is determined between the reference and the measured fields of gravity and magnetic north and subsequently compensated via proportional and integral feedback compensation before being fused with the gyroscopic data. Proportional feedback decides the cross-over frequency between the low and high pass filters and the integral feedback corrects for time dependent zero-bias drift in the gyroscope data. The zero bias drifts over time due to the changes in the physical properties of the sensor. This adds a DC component to the measured angular velocity which thus dominates the low frequency content making the attitude estimation from gyroscope data unreliable at low angular frequencies. In contrast, the accelerometer and magnetometer are susceptible to high frequency noise that invalidates attitude estimations using these sensors at higher angular frequencies.

In addition magnetic distortion can result in the measured direction of Earth's magnetic field due to the presence of ferromagnetic materials in the vicinity of the magnetometer. This is corrected by defining the reference direction of Earth's magnetic field in the Earth frame($^{E}\hat{b}[n] = [b_x, b_y, b_z]$) in terms of the measured direction of the field in the Sensor frame($^{S}\hat{m}[n] = [m_x, m_y, m_z]$) which is mapped to the Earth frame($^{E}\hat{h}[n] = [h_x, h_y, h_z]$). Cleanflight assumes that the reference direction of magnetic field is strictly perpendicular to the direction of gravity i.e. lies only in the horizontal plane $\perp$ to the Z axis i.e. $^{E}\hat{b}[n] = [b_x, 0, 0]$. Similarly the measured direction in the Earth frame is assumed to have no vertical component i.e. $^{E}\hat{h}[n] = [h_x, h_y, 0]$. The normalized reference direction of magnetic North in earth frame is thus calculated as:

$$\boxed{^{E}\hat{b}[n] = \begin{bmatrix} \sqrt{h_x^2 + h_y^2} & 0 & 0 \end{bmatrix}} \tag{100}$$

## 5.1 Quaternion Algebra

Cleanflight models the flight controller system as a first order kinematic system and implements Madgwick's C code interpretation of Mahony's algorithm in quaternion form(4D complex number) for its AHRS subsystem.

- The normalized quaternion representation; ${}^{E}_{S}\hat{q}[n] = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}$ is used to represent orientation of the sensor frame(leading subscript $S$) with respect to the earth frame(leading superscript $E$) in 3D space at time step n.
- Quaternion product is represented by $\bigotimes$.
- The quaternion conjugate denoted by $^*$ is used to swap relative frames for an

orientation i.e. ${}^E_S\hat{q}[n]$(orientation of S frame wrt E frame) is the conjugate of ${}^S_E\hat{q}[n]$(orientation of E frame wrt S frame).

- The time equations of the AHRS algorithm use the quaternion rotation operator(equation:101) to map measured field vectors(gravity and magnetic north) in the sensor frame coordinates to earth frame coordinates.
- Quaternion algebra avoids singularity points in rotation calculations, that would otherwise exist in Euler angle(roll:$\phi$, pitch:$\theta$, yaw:$\psi$ ) rotations, by representing such rotations about all three of the axes(X,Y and Z) in a tightly coupled and compact manner.

The quaternion rotation in time domain for a vector ${}^S v$ in the sensor frame results in a rotated vector ${}^E v$ in the earth frame:

$$\begin{aligned}
{}^E\mathbf{v}[n] &= {}^S_E\hat{\mathbf{q}}[n] \bigotimes {}^S\mathbf{v}[n] \bigotimes {}^S_E\hat{\mathbf{q}}^*[n] && \text{(101a)}\\
&= {}^S_E Q\, {}^S\mathbf{v}[n] && \text{(101b)}\\
&\textit{where} \\
{}^S_E Q[n] &= \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2(q_xq_y-q_wq_z) & 2(q_xq_z+q_wq_y)\\ 2(q_xq_y+q_wq_z) & 1-2q_x^2-2q_z^2 & 2(q_yq_z-q_wq_x)\\ 2(q_xq_z-q_wq_y) & 2(q_yq_z+q_wq_x) & 1-2q_x^2-2q_y^2 \end{bmatrix} && \text{(101c)}
\end{aligned}$$

Similarly the inverse(conjugate) quaternion rotation for a vector ${}^E v$ in the earth frame results in a rotated vector ${}^S v$ in the sensor frame:

$$\begin{aligned}
{}^S\mathbf{v}[n] &= {}^S_E\hat{\mathbf{q}}^*[n] \bigotimes {}^E\mathbf{v}[n] \bigotimes {}^S_E\hat{\mathbf{q}}[n] && \text{(102a)}\\
&= {}^S_E Q^t\, {}^E\mathbf{v}[n] && \text{(102b)}\\
&= {}^E_S Q\, {}^E\mathbf{v}[n] && \text{(102c)}\\
&= {}^E_S\hat{\mathbf{q}}[n] \bigotimes {}^E\mathbf{v}[n] \bigotimes {}^E_S\hat{\mathbf{q}}^*[n] && \text{(102d)}\\
&\textit{where} \\
{}^S_E Q^t[n] &= \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2(q_xq_y+q_wq_z) & 2(q_xq_z-q_wq_y)\\ 2(q_xq_y-q_wq_z) & 1-2q_x^2-2q_z^2 & 2(q_yq_z+q_wq_x)\\ 2(q_xq_z+q_wq_y) & 2(q_yq_z-q_wq_x) & 1-2q_x^2-2q_y^2 \end{bmatrix} && \text{(102e)}
\end{aligned}$$

## 5.2 Error from Accelerometer Measurements

Reference field of gravity in the earth frame is parallel to the Z axis i.e. ${}^E\hat{G} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$. Rotation of the reference field of gravity to sensor frame is:

$${}^S\hat{\mathbf{G}}[n] = {}^S_E\hat{\mathbf{q}}^*[n] \bigotimes {}^E\hat{\mathbf{G}} \bigotimes {}^S_E\hat{\mathbf{q}}[n] \qquad (103)$$

The error is calculated as the cross product between the reference gravity field in the sensor frame; ${}^{S}\hat{\mathbf{G}}[n]$ and the measured gravity field; ${}^{S}\hat{\mathbf{a}}_{avg}[n]$(equation :97):

$$
{}^{S}\mathbf{e}_{acc}[n] = {}^{S}\hat{\mathbf{a}}_{avg}[n] \times {}^{S}\hat{\mathbf{G}}[n] \tag{104}
$$

## 5.3 Error from Magnetometer Measurements

From equation 100 we have the reference direction of the magnetic field in the earth frame(${}^{E}\hat{\mathbf{b}}[n]$), represented in terms of measured direction of the field in the earth frame; ${}^{E}\hat{\mathbf{h}}[n]$. The measured field in the earth frame is obtained via a quaternion rotation of the measured field vector in the sensor frame (${}^{S}\hat{\mathbf{m}}[n]$) using equation 101.

$$
\begin{aligned}
{}^{E}\hat{\mathbf{h}}[n] &= \begin{bmatrix} h_x & h_y & 0 \end{bmatrix} \\
&= {}^{S}_{E}\hat{\mathbf{q}}[n] \bigotimes {}^{S}\hat{\mathbf{m}}[n] \bigotimes {}^{S}_{E}\hat{\mathbf{q}}^{*}[n]
\end{aligned} \tag{105}
$$

The error vector wrt the earth frame is calculated as the cross product between reference and measured magnetic fields in the earth frame:

$$
\begin{aligned}
{}^{E}\mathbf{e}_{mag}[n] &= {}^{E}\hat{\mathbf{h}}[n] \times {}^{E}\hat{\mathbf{b}}[n] \\
&= \begin{bmatrix} 0 & 0 & e_z \end{bmatrix}
\end{aligned} \tag{106}
$$

Error in the sensor frame is computed as follows (refer to equation: 102):

$$
{}^{S}\mathbf{e}_{mag}[n] = {}^{S}_{E}\hat{\mathbf{q}}^{*}[n] \bigotimes {}^{E}\mathbf{e}_{mag}[n] \bigotimes {}^{S}_{E}\hat{\mathbf{q}}[n] \tag{107}
$$

## 5.4 Error Compensation: PI feedback control

The total error is fed into the PI controller. The corresponding time equations are:

$$
\boxed{{}^{S}\mathbf{e}_{total}[n] = {}^{S}\mathbf{e}_{acc}[n] + {}^{S}\mathbf{e}_{mag}[n]} \tag{108}
$$

- **Proportional Control:**

$$
{}^{S}\mathbf{u}_{p,att}[n] = K_{p,att}\, {}^{S}\mathbf{e}_{total}[n] \tag{109}
$$

  where

  - $K_{p,att}$ = proportional gain for total error in the attitude($att$) estimate

- **Integral Control:**

$$]\,{}^{S}\mathbf{u}_{i,att}[n] = {}^{S}\mathbf{u}_{i,att}[n-1] + K_{i,att}\,{}^{S}\mathbf{e}_{total}[n]T_{s,att} \tag{110}$$

  where

  - $K_{i,att}$ = integral gain for error

  The equation gives accumulated error till time step [n-1] summed with error between time step [n] and [n-1] i.e for one sampling time period; $T_{s,att}$ (1/task rate) for the ATTITUDE task.

The controller output is thus:

$$\boxed{{}^{S}\mathbf{u}_{att}[n] = {}^{S}\mathbf{u}_{p,att}[n] + {}^{S}\mathbf{u}_{i,att}[n]} \tag{111}$$

## 5.5 Integration with Gyroscope Measurements

The gyroscope input(equation:99) is fused with the controller's output:

$${}^{S}\Omega[n] = {}^{S}\mathbf{g}_{avg}[n] + {}^{S}\mathbf{u}_{att}[n] \tag{112}$$

The quaternion derivative describes the rate of change of orientation of the earth frame with respect to the sensor frame and is calculated as:

$${}^{S}_{E}\dot{\mathbf{q}}[n] = \frac{1}{2}\,{}^{S}_{E}\hat{\mathbf{q}}[n-1] \bigotimes {}^{S}\Omega[n] \tag{113}$$

The sensor orientation is computed by taking the numerical integral of the quaternion derivative from equation 113, with known initial value of the estimate.

$$\boxed{{}^{S}_{E}\mathbf{q}[n] = {}^{S}_{E}\hat{\mathbf{q}}[n-1] + {}^{S}_{E}\dot{\mathbf{q}}[n]T_{s,att}} \tag{114}$$

where

- ${}^{S}_{E}\mathbf{q}[n]$ = attitude estimate of orientation
- ${}^{S}_{E}\hat{\mathbf{q}}[n-1]$ = previous attitude estimate of orientation
- $T_{s,att}$ = sampling period of the ATTITUDE task

Finally the estimate is normalized:

$${}^{S}_{E}\hat{\mathbf{q}}[n] = \frac{{}^{S}_{E}\mathbf{q}[n]}{\left\|{}^{S}_{E}\mathbf{q}[n]\right\|} \tag{115}$$

## 5.6 Attitude in terms of Roll, Pitch and Yaw Angles

The quaternion estimate is used to compute attitude values for each individual Euler angle:

- **Roll angle in degrees:**

$$\phi[n] = \tan^{-1}\left(\frac{2(q_y q_z + q_w q_x)}{1 - 2q_x^2 - 2q_y^2}\right) \tag{116}$$

- **Pitch angle in degrees:**

$$\theta[n] = \pi/2 - \cos^{-1}\Bigg(-2(q_x q_z - q_w q_y)\Bigg) \tag{117}$$

- **Yaw angle in degrees:**

$$\psi[n] = -\tan^{-1}\left(\frac{2(q_x q_y + q_w q_z)}{1 - 2q_y^2 - 2q_z^2}\right) \tag{118}$$

## 5.7 Frequency domain

Each sensor input to the AHRS sub-system has three real valued, time dependent components corresponding to the measured sensor value in each of the three axis; ${}^{S}X$, ${}^{S}Y$ and ${}^{S}Z$, in the sensor frame. Their corresponding Z transforms are:

- Accelerometer:

$$\mathbf{a}[n] = \begin{bmatrix} a_x[n] & a_y[n] & a_z[n] \end{bmatrix} \tag{119a}$$
$$a_x[n] \longleftrightarrow A_x(z)$$
$$a_y[n] \longleftrightarrow A_y(z)$$
$$a_z[n] \longleftrightarrow A_z(z)$$
$$\mathbf{a}[n] \longleftrightarrow \mathbf{A}(z) \tag{119b}$$

- Gyroscope:

$$\mathbf{g}[n] = \begin{bmatrix} g_x[n] & g_y[n] & g_z[n] \end{bmatrix} \tag{120a}$$
$$g_x[n] \longleftrightarrow G_x(z)$$
$$g_y[n] \longleftrightarrow G_y(z)$$
$$g_z[n] \longleftrightarrow G_z(z)$$
$$\mathbf{g}[n] \longleftrightarrow \mathbf{G}(z) \tag{120b}$$

- Magnetometer:

$$\mathbf{m}[n] = \begin{bmatrix} m_x[n] & m_y[n] & m_z[n] \end{bmatrix} \tag{121a}$$
$$m_x[n] \longleftrightarrow M_x(z)$$
$$m_y[n] \longleftrightarrow M_y(z)$$
$$m_z[n] \longleftrightarrow M_z(z)$$
$$\mathbf{m}[n] \longleftrightarrow \mathbf{M}(z) \tag{121b}$$

The state estimate of the sensor, represented by a unit quaternion(refer to time domain analysis in the previous sub-sections) has four real-valued, time dependent component signals; ${}^S_E\hat{q}[n] = \begin{bmatrix} \hat{q}_w[n] & \hat{q}_x[n] & \hat{q}_y[n] & \hat{q}_z[n] \end{bmatrix}$. The frequency domain analysis of quaternions is an active area of research and hence out of scope of this work. We thus represent the Z transform of a quaternion as: $\mathcal{Z}\{q\}$. For a detailed analysis of Z transforms of 4-tuple complex numbers(quaternions), refer to 2016's SIBCON paper [2].

The AHRS sub-system is a MIMO (Multiple Input and Multiple Output) system with 3 different inputs from the three sensors namely accelerometer, magnetometer and gyroscope and three Euler angle outputs namely roll($\phi[n]$), pitch($\theta[n]$) and yaw($\psi[n]$) that are derived from the quaternion estimate. Each Euler angle is independent of the other two Euler angles and can be computed individually(refer to sub-section 5.6). The Euler angles describe an orientation of the sensor frame achieved by independent sequential rotations, from alignment with the earth frame, of $\psi$ around ${}^SZ$, $\theta$ around ${}^SY$ and $\phi$ around ${}^SX$. From the previous discussion of the AHRS algorithm we can safely assume that the attitude estimate depends linearly on the gyroscope input and the error calculated from the accelerometer and magnetometer inputs. The following frequency analysis elaborates on this:

$$\mathcal{Z}\{q[n]\} = \mathcal{Z}\{q[n-1]\} + \mathcal{Z}\{\dot{q}[n]\}T_{s,att} \qquad \leftarrow \textit{(equation 114)} \tag{122a}$$
$$\mathcal{Z}\{\dot{q}[n]\} = \frac{1}{2}\mathcal{Z}\{q[n-1] \bigotimes \Omega[n]\} \qquad \leftarrow \textit{(equation 113)} \tag{122b}$$
$$\mathcal{Z}\{\Omega[n]\} = \mathcal{Z}\{g_{avg}[n]\} + \mathcal{Z}\{u_{att}[n]\} \qquad \leftarrow \textit{(equation 112)}$$
$$= \mathbf{G}(z) + U_{att}(z) \tag{122c}$$
$$U_{att}(z) = U_p(z) + U_i(z) \quad \textit{where} \qquad \leftarrow \textit{(equation 111)} \tag{122d}$$
$$U_p(z) = K_p E_{total}(z) \quad \textit{and} \qquad \leftarrow \textit{(equation 109)}$$
$$U_i(z) = K_i T_{\{s,att\}}\left(\frac{z}{z-1}\right)E_{total}(z) \qquad \leftarrow \textit{(equation 110)}$$
$$E_{total}(z) = E_{acc}(z) + E_{mag}(z) \tag{122e}$$

This implies:

$$\frac{U_{att}(z)}{E_{total}(z)} = K_p + K_i T_{\{s,att\}} \left( \frac{z}{z-1} \right) \tag{123}$$

Thus:

$$\mathcal{Z}\{\Omega[n]\} = \mathbf{G}(z) + \left[ K_p + K_i T_s \left( \frac{z}{z-1} \right) \right] E_{acc}(z) + \left[ K_p + K_i T_s \left( \frac{z}{z-1} \right) \right] E_{mag}(z) \tag{124}$$

We assume the transfer function of ${}^S_E\hat{q}[n]$ with respect to the input $\Omega[n]$ is:

$$\begin{gathered} \frac{\mathcal{Z}\{\hat{q}[n]\}}{\mathcal{Z}\{\Omega[n]\}} = D \implies \\ \mathcal{Z}\{\hat{q}[n]\} = D\Big( \mathcal{Z}\{\Omega[n]\} \Big) \end{gathered} \tag{125}$$

The frequency relation between accelerometer and magnetometer inputs; $\hat{a}[n]$ and $\hat{m}[n]$, and the corresponding error terms is non linear and much too complicated. To keep the complexity down we consider the entire transfer function of the AHRS subsystem as a blackbox with respect to the accelerometer and magnetometer inputs. The transfer functions with respect to each sensor input and their corresponding block diagram depictions are presented as follows:

- Gyroscope:

$$\frac{\mathcal{Z}\{\hat{q}[n]\}}{\mathbf{G}(z)} = D_{gyro} = D \tag{126}$$

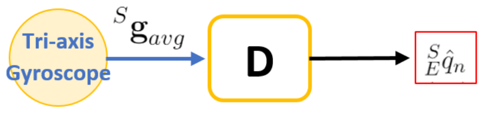


**Figure 24:** Control flow between gyroscope input and the attitude estimate output.

- Accelerometer:

$$\frac{\mathcal{Z}\{\hat{q}[n]\}}{\mathbf{A}(z)} = D_{acc} \tag{127}$$

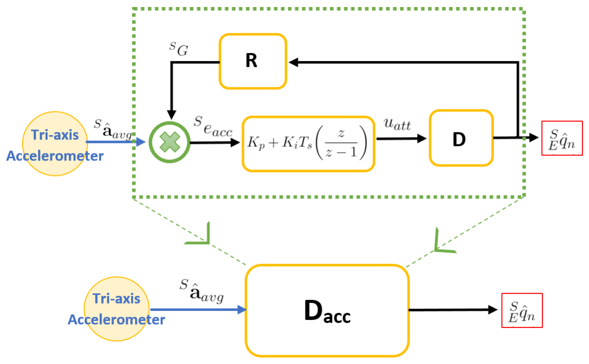


**Figure 25:** Control flow between accelerometer input and the attitude estimate output.

- Magnetometer:

$$\frac{\mathcal{Z}\{\hat{q}[n]\}}{\mathbf{M}(z)} = D_{mag} \tag{128}$$

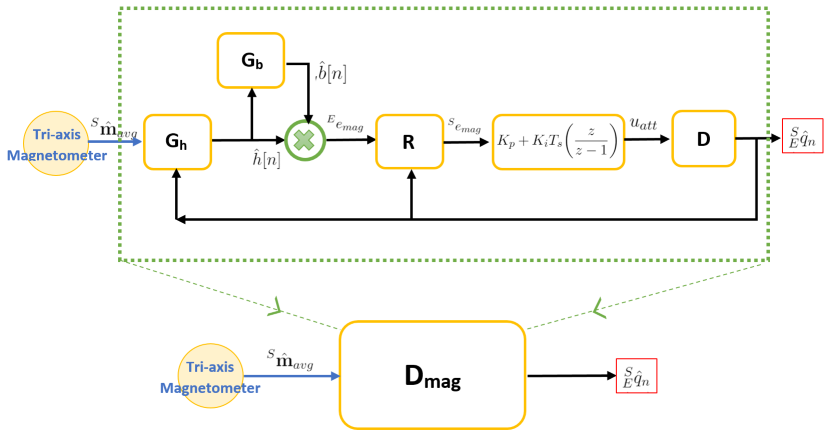


**Figure 26:** Control flow between magnetometer input and the attitude estimate output.

- Summarized linear relation between the attitude estimate in quaternion form and each sensor input can be thus represented as:

$$\mathcal{Z}\{\hat{q}[n]\} = D_{gyro}\mathbf{G}(z) + D_{acc}\mathbf{A}(z) + D_{mag}\mathbf{M}(z) \tag{129}$$

Individual Euler angle output for roll, pitch and yaw can be derived from the quaternion state estimate using time equations presented in sub-section:5.6. We consider the transfer function between each Euler angle and the state estimate in quaternion form as:

$$\frac{\phi(z)}{\mathcal{Z}\{q[n]\}} = D_\phi \qquad \frac{\theta(z)}{\mathcal{Z}\{q[n]\}} = D_\theta \qquad \frac{\psi(z)}{\mathcal{Z}\{q[n]\}} = D_\psi \tag{130a}$$

The overall simplified AHRS sub-system is shown in Figure:27:

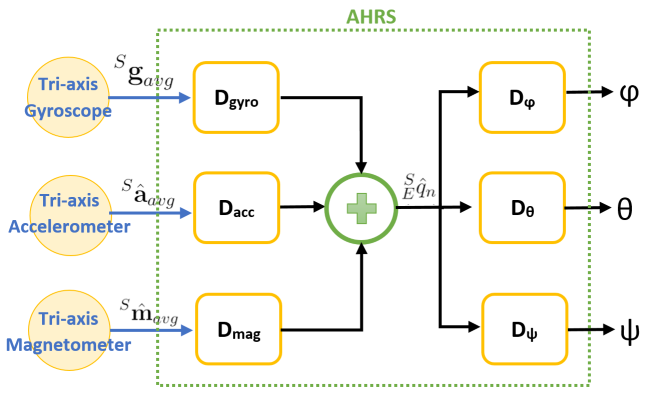


**Figure 27:** Control flow between sensor inputs and attitude estimate in terms of Euler angles.

### 5.7.1 Sampling Frequency of the AHRS sub-system

We observe that CF's ATTITUDE task's sampling frequency information is implicitly contained within the individual sub-module transfer functions of the AHRS sub-system. As a side effect of avoiding complexity of Z transforms for hypercomplex numbers, we lose control of the tunable task sampling frequency parameter. Thus for the purposes of our analysis of the overall CF flight controller loop, we explicitly choose a range of sampling frequencies for the AHRS sub-system (Cleanflight by default uses 100Hz).

The minimum sampling frequency is set based on Madgewick's findings [5] from his investigation into the effect of sampling rate on the performance of the orientation filter. According to [5], static and dynamic performance, measured as the root mean square values of the error in Euler angles between the actual and estimated values from the filter, remains constant for sampling frequencies above 50Hz. Also a tolerable error in attitude results at frequencies as low as 10Hz. Thus using Madgwick's results as baseline, we choose our minimum frequency as 10Hz.

On the other hand the maximum frequency is restricted by the highest sampling frequency of the sensor hardware. An accurate estimate of the attitude would need atleast one measured value from each of the three sensors hence we take the minimum of all three sensor tasks'(GYROSCOPE, ACCELEROMETER and COMPASS) sampling frequency as our maximum frequency bound for the ATTITUDE task.

## References

[1] MathWorks Matlab-R2018b Documentation. Two-degree-of-freedom pid controllers. https://www.mathworks.com/help/control/ug/two-degree-of-freedom-2-dof-pid-controllers.html.

[2] A. Uncini. F. Ortolani. Quaternion digital signal processing: a hypercomplex approach to information processing. 2016.

[3] Home. Cleanflight flight controller. https://www.cleanflight.com.

[4] R. Mahony J. Kim T. Hamel. M. Euston, P. Coote. A complementary failter for attitude estimation of a fixed-wing uav. pages 340–345, 2008.

[5] A. J. L. Harrison S. O. H. Madgwick and R. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. 2011.