现在通用的姿态传感器是 MPU6050，这种传感器能输出三个加速度 acc 和三个转动 gyro。
有了这几个数据就可以通过数学解算得到飞行器的姿态。

欧拉角

欧拉角 wiki

> 对于任何参考系，一个刚体的取向，是依照顺序，从这参考系，做三个欧拉角的旋转而设定的。所以，刚体的取向可以用三个基本旋转矩阵来决定。换句话说，任何关于刚体旋转的旋转矩阵是由三个基本旋转矩阵复合而成的。
>
> 对于在三维空间里的一个参考系，任何坐标系的取向，都可以用三个欧拉角来表现。参考系又称为实验室参考系，是静止不动的。而坐标系则固定于刚体，随着刚体的旋转而旋转。

四元数

参考：http://www.crazepony.com/book/wiki/quaternions.html

四元数介绍

> 我们知道在平面(x,y)中的旋转可以用复数来表示，同样的三维中的旋转可以用单位四元数来描述。我们来定义一个四元数：
>
> $$\mathbf{q} = a + \overrightarrow{u} = q_0 + q_1 i + q_2 j + q_3 k$$
>
> 我们可以把它写成
>
> $$[w, \mathbf{v}]$$
>
> ，其中
>
> $$\mathbf{v} = q_1 i + q_2 j + q_3 k$$

，

$$w = a = q_0$$

。那么 V 是矢量，表示三维空间中的旋转轴。w 是标量，表示旋转角度。那么

$$[w, \mathbf{v}]$$

就是绕轴 V 旋转 w 度，所以一个四元数可以表示一个完整的旋转。只有单位四元数才可以表示旋转，至于为什么，因为这就是四元数表示旋转的约束条件。

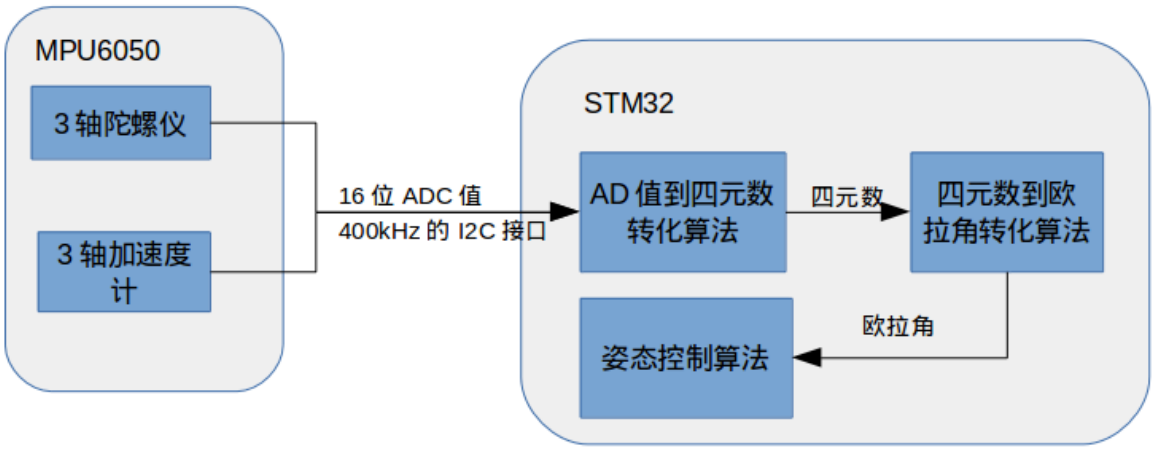
相对于另几种旋转表示法（矩阵，欧拉角，轴角），四元数具有某些方面的优势，如速度更快、提供平滑插值、有效避免万向锁问题、存储空间较小等等。

姿态解算的核心在于旋转，一般旋转有 4 种表示方式：矩阵表示、欧拉角表示、轴角表示和四元数表示。矩阵表示适合变换向量，欧拉角最直观，轴角表示则适合几何推导，而在组合旋转方面，四元数表示最佳。因为姿态解算需要频繁组合旋转和用旋转变换向量，所以采用四元数保存组合姿态、辅以矩阵来变换向量的方案。

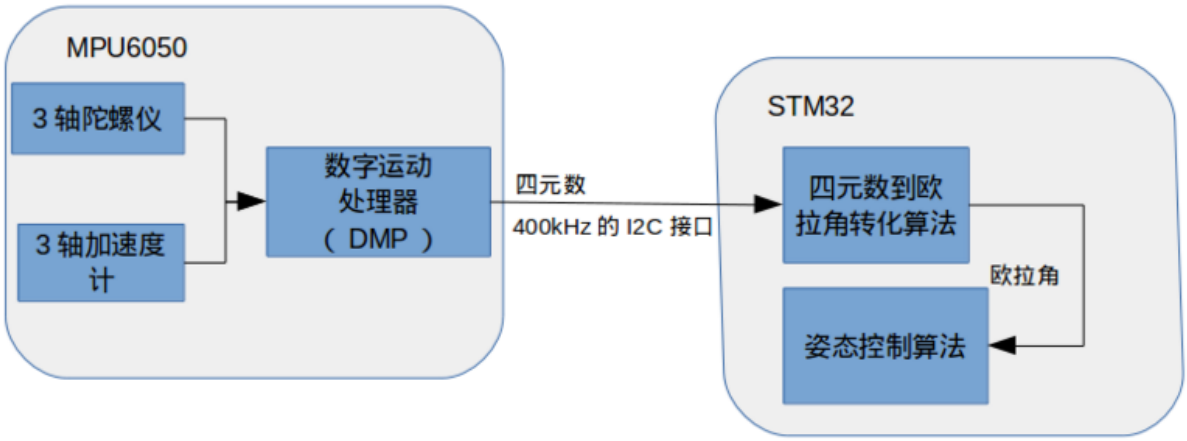
总结来说，姿态解算中使用四元数来保存飞行器的姿态，包括旋转和方位。在获得四元数之后，会将其转化为欧拉角，然后输入到姿态控制算法中。

姿态控制算法的输入参数必须要是欧拉角。AD 值是指 MPU6050 的陀螺仪和加速度值，3 个维度的陀螺仪值和 3 个维度的加速度值，每个值为 16 位精度。AD 值必须先转化为四元数，然后通过四元数转化为欧拉角。这个四元数可能是软解，主控芯片（STM32）读取到 AD 值，用软件从 AD 值算得，也可能是通过 MPU6050 中的 DMP 硬解，主控芯片（STM32）直接读取到四元数。

MPU6050 中四元数与欧拉角的关系



软解姿态示意图



硬解姿态

数学推导其关系

旋转一个角度的坐标变换：

上面仅仅是绕一根轴的旋转，如果三维空间中的欧拉角旋转要转三次：

$$O-X_nY_nZ_n \xrightarrow{\text{绕}-Z_n\text{轴旋转}\ \psi} O-X_1Y_1Z_1 \xrightarrow{\text{绕}X_1\text{轴旋转}\ \theta} O-X_2Y_2Z_2 \xrightarrow{\text{绕}Y_2\text{轴旋转}\ \gamma} O-X_bY_bZ_b$$

$$C_n^b = C_2^b C_1^2 C_n^1 = \begin{bmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & -\sin\phi & 0 \\ \sin\phi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\gamma\cos\psi + \sin\gamma\sin\psi\sin\theta & -\cos\gamma\sin\psi + \sin\gamma\cos\psi\sin\theta & -\sin\gamma\cos\theta \\ \sin\psi\cos\theta & \cos\psi\cos\theta & \sin\theta \\ \sin\gamma\cos\psi - \cos\gamma\sin\psi\sin\theta & -\sin\gamma\sin\psi - \cos\gamma\cos\psi\sin\theta & \cos\gamma\cos\theta \end{bmatrix}$$

坐标变换

欧拉角旋转一个角度

$$\begin{bmatrix} \dot{\gamma} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \frac{1}{\cos\theta} \begin{bmatrix} \cos\theta & \sin\gamma\sin\theta & \cos\gamma\sin\theta \\ 0 & \cos\theta\cos\gamma & -\sin\gamma\cos\theta \\ 0 & \sin\gamma & \cos\gamma\cos\theta \end{bmatrix}^{-1} \bullet \begin{bmatrix} \omega_{EbX}^b \\ \omega_{EbY}^b \\ \omega_{EbZ}^b \end{bmatrix}$$

引入四元数后，刚才用欧拉角描述的方向余弦矩阵用四元数描述则为：

$$C_n^b = \begin{bmatrix} q_1^2 + q_0^2 - q_3^2 - q_2^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_2^2 - q_3^2 + q_0^2 - q_1^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_3^2 - q_2^2 - q_1^2 + q_0^2 \end{bmatrix}$$

所以在软件解算中，我们要首先把加速度计采集到的值(三维向量)转化为单位向量,即向量除以模，传入参数是陀螺仪 x,y,z 值和加速度计 x,y,z 值：
首先由三个加计的值，转化为单位向量（值除以模）：

```
void IMUupdate(float gx, float gy, float gz, float ax, float ay, float az) {
float norm;
```

```
float vx, vy, vz;
float ex, ey, ez;

norm = sqrt(ax*ax + ay*ay + az*az);
ax = ax / norm;
ay = ay / norm;
az = az / norm;
```

由角速度得到四元数，其中 qn 的初始值是 0

```
// integrate quaternion rate and normalise
q0 = q0 + (-q1*gx - q2*gy - q3*gz)*halfT;
q1 = q1 + (q0*gx + q2*gz - q3*gy)*halfT;
q2 = q2 + (q0*gy - q1*gz + q3*gx)*halfT;
q3 = q3 + (q0*gz + q1*gy - q2*gx)*halfT;
```

下面把四元数换算成方向余弦中的第三行的三个元素。刚好 vx,vy,vz 其实就是上一次的欧拉角（四元数）的机体坐标参考系换算出来的重力的单位向量。

```
// estimated direction of gravity
vx = 2*(q1*q3 - q0*q2);
vy = 2*(q0*q1 + q2*q3);
vz = q0*q0 - q1*q1 - q2*q2 + q3*q3;
```

- 向量间的误差，可以用向量叉积（也叫向量外积、叉乘）来表示，exyz 就是两个重力向量的叉积。
  这个叉积向量仍旧是位于机体坐标系上的，而陀螺积分误差也是在机体坐标系，而且叉积的大小与陀螺积分误差成正比，正好拿来纠正陀螺。（你可以自己拿东西想象一下）由于陀螺是对机体直接积分，所以对陀螺的纠正量会直接体现在对机体坐标系的纠正。
  用叉积误差来做 PI 修正陀螺零偏

```
// integral error scaled integral gain
exInt = exInt + ex*Ki;
eyInt = eyInt + ey*Ki;
ezInt = ezInt + ez*Ki;

// adjusted gyroscope measurements
gx = gx + Kp*ex + exInt;
gy = gy + Kp*ey + eyInt;
gz = gz + Kp*ez + ezInt;
```

- 欧拉角:

  ```
  Q_ANGLE.Yaw = atan2(2 * q1 * q2 + 2 * q0 * q3, -2 * q2*q2 - 2 * q3* q3 + 1)* 57.3; // yaw
  Q_ANGLE.Pitch  = asin(-2 * q1 * q3 + 2 * q0* q2)* 57.3; // pitch
  Q_ANGLE.Roll = atan2(2 * q2 * q3 + 2 * q0 * q1, -2 * q1 * q1 - 2 * q2* q2 + 1)* 57.3; // roll
  ```

整理思路:

- 对加速度进行单位化处理:

  ```
  //加速度单位化处理
  norm = sqrt(ax*ax + ay*ay + az*az);
  if(norm==0) return 0;
  ax = ax / norm;
  ay = ay / norm;
  az = az / norm;
  ```

- 把飞行器上次计算得到的姿态（四元数）换算成“方向余弦矩阵”中的第三列的三个元素。根据余弦矩阵和欧拉角的定义，地理坐标系的重力向量，转到机体坐标系，正好是这三个元素。

```
vx = 2*(q1q3 - q0q2);
vy = 2*(q0q1 + q2q3);
vz = q0q0 - q1q1 - q2q2 + q3q3;
```

- 在机体坐标系上，加速度计测出来的重力向量是 ax、ay、az。
  由上次姿态解算的姿态（可以简单认为是陀螺积分）来推算出的重力向量是 vx、vy、vz。
  它们之间的误差向量，就是上次姿态解算（可以认为是陀螺仪积分）后的姿态和加速度计测出来的姿态之间的误差。

```
//坐标系和重力叉积运算
ex = (ay*vz - az*vy);
ey = (az*vx - ax*vz);
ez = (ax*vy - ay*vx);
//
//误差
exInt = exInt + ex*bs004_quad_Ki;
eyInt = eyInt + ey*bs004_quad_Ki;
ezInt = ezInt + ez*bs004_quad_Ki;
```

- 用叉乘误差来做 PI 修正陀螺零偏，通过调节 Kp，Ki 两个参数，可以控制加速度计修正陀螺仪积分姿态的速度。

```
//消除误差
gx = gx + bs004_quad_Kp*ex + exInt;
gy = gy + bs004_quad_Kp*ey + eyInt;
gz = gz + bs004_quad_Kp*ez + ezInt;
```

- 最后得到四元数&欧拉角：

```
//最终四元数的值
q0 = q0 + (-q1*gx - q2*gy - q3*gz)*bs004_quad_halfT;
q1 = q1 + (q0*gx + q2*gz - q3*gy)*bs004_quad_halfT;
q2 = q2 + (q0*gy - q1*gz + q3*gx)*bs004_quad_halfT;
q3 = q3 + (q0*gz + q1*gy - q2*gx)*bs004_quad_halfT;
//
//四元数单位化
norm = sqrt(q0*q0 + q1*q1 + q2*q2 + q3*q3);
if(norm==0) return 0;
q0 = q0 / norm;
q1 = q1 / norm;
q2 = q2 / norm;
q3 = q3 / norm;
//
//欧拉角转换
bs004_imu_roll=asin(-2*q1q3 + 2*q0q2)*57.30f;
bs004_imu_pitch=atan2(2*q2q3 + 2*q0q1, -2*q1q1-2*q2q2 + 1)*57.30f;
bs004_imu_yaw=bs004_imu_yaw+2*gz_input/bs004_mpu6050_gyro_scale;
```