MPU-6050
OBTAINING ANGLE VALUES USING RAW MEASUREMENTS FROM THE SENSOR

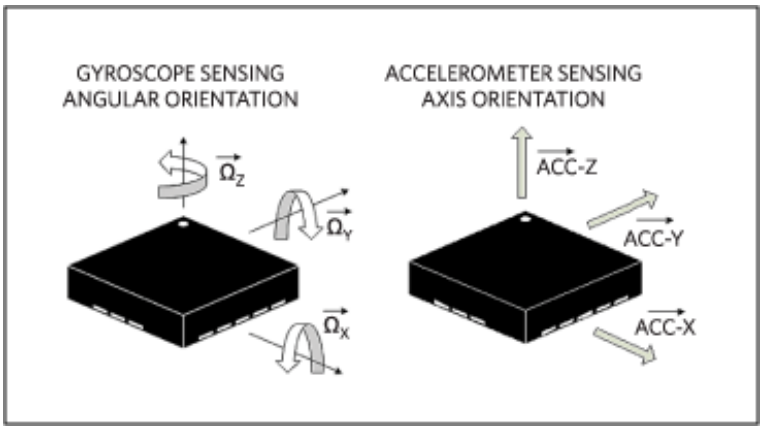
Fryderyk Kowalczyk

Contents

1 MPU-6050

The MPU-6050 is a sensor module that combines a three-axis accelerometer and a three-axis gyroscope into a single integrated circuit.

1. Accelerometer: Measures acceleration along the x, y, and z axes. It detects changes in velocity or movement in any direction.
2. Gyroscope: Measures angular velocity or rotation rate around the x, y, and z axes. It detects the rate of rotation or changes in orientation.



2 Accelerometer

Transformation of Raw Accelerometer Data into RPY (Roll, Pitch, Yaw)

Raw accelerometer data can be represented as an acceleration vector $\mathbf{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$.

The accelerometer model can be expressed by an equation describing the measured acceleration as the sum of various components. Below is the general accelerometer model:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \ddot{x}_{\text{linear}} \\ \ddot{y}_{\text{linear}} \\ \ddot{z}_{\text{linear}} \end{bmatrix} + \begin{bmatrix} \ddot{x}_{\text{rotation}} \times y_{\text{rot}} - \ddot{y}_{\text{rotation}} \times x_{\text{rot}} \\ \ddot{x}_{\text{rotation}} \times x_{\text{rot}} + \ddot{y}_{\text{rotation}} \times y_{\text{rot}} \\ \ddot{z}_{\text{rotation}} \end{bmatrix} + \begin{bmatrix} g \times \sin(\theta) \\ -g \times \sin(\phi) \\ g \times \cos(\phi) \times \cos(\theta) \end{bmatrix} + \begin{bmatrix} \text{bias}_x \\ \text{bias}_y \\ \text{bias}_z \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

where:

- a_x, a_y, a_z - measured acceleration in the x, y, z axes respectively,
- $\ddot{x}_{\text{linear}}, \ddot{y}_{\text{linear}}, \ddot{z}_{\text{linear}}$ - linear accelerations,
- $\ddot{x}_{\text{rotation}}, \ddot{y}_{\text{rotation}}, \ddot{z}_{\text{rotation}}$ - rotation-induced accelerations,
- $x_{\text{rot}}, y_{\text{rot}}, z_{\text{rot}}$ - rotation vector coordinates,
- g - gravitational acceleration,
- ϕ, θ, ψ - Roll, Pitch, Yaw angles,
- $\text{bias}_x, \text{bias}_y, \text{bias}_z$ - systematic error (bias) components,
- n_x, n_y, n_z - noise components.

This model helps understand how raw accelerometer data is composed of different components affecting the final acceleration measurement. However, to read the RPY angles, it is necessary to simplify the model to a static model to reduce the calculations only to the gravitational acceleration, resulting in:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} g \times \sin(\theta) \\ -g \times \sin(\phi) \\ g \times \cos(\phi) \times \cos(\theta) \end{bmatrix}$$

This allows deriving the Pitch and Roll angles as follows:

$$\hat{\theta} = \tan^{-1}\left(\frac{a_y}{a_z}\right)$$

$$\hat{\phi} = \sin^{-1} = \left(\frac{a_x}{g}\right)$$

Reducing the model to a static form means the accelerometer does not guarantee accurate measurements when the system is in motion.

3 Gyroscope

$$\begin{cases} \dot{\phi} = p + b_\phi + n_\phi \\ \dot{\theta} = q + b_\theta + n_\theta \\ \dot{\psi} = r + b_\psi + n_\psi \end{cases}$$

where:

- $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$ are the angular velocities around the Roll, Pitch, and Yaw axes respectively,
- p, q, r are the actual gyroscope angular velocities,
- b_ϕ, b_θ, b_ψ are the biases in angular velocities,
- n_ϕ, n_θ, n_ψ are white noise (uncorrelated, stationary Gaussian noise).

The given velocities can be represented using Euler angle representations:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

To obtain the necessary angles from the obtained measurements, integration is required.

$$\hat{\phi} = \int_0^T (p + \sin\phi\tan\theta q + \cos\phi\tan\theta r)\, dt + \phi_0 = \int_0^T \dot{\phi}\, dt + \phi_0$$

$$\hat{\theta} = \int_0^T (\cos\phi q - \sin\phi r)\, dt + \theta_0 = \int_0^T \dot{\theta}\, dt + \theta_0$$

Integration over time introduces the need to apply a constant and the most precise sampling frequency. For this purpose, timers are used in the program, the period of which can be calculated as follows.

$$T = \frac{Prescaler\_Value \times Auto\_Reload\_Register}{f_{HCLK}}$$

For an 80MHz clock, in the case of the microcontroller being used, the period will be very accurate. Upon the overflow of the ARR register (after the period elapses), HAL allows the triggering of an interrupt that will start the measurement. Such a solution will allow achieving the maximum possible precision.