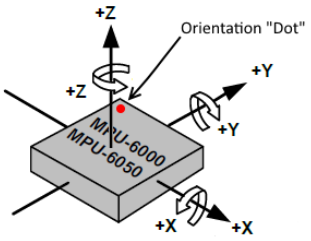
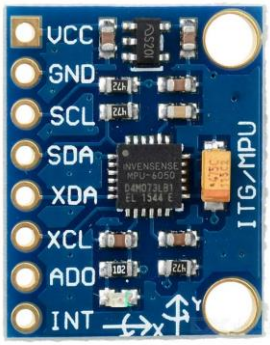
Balancing Robot Maths !

The maths behind the balancing robot is simpler and more interesting than you might think !

Firstly let's have a quick look at the MPU-6050 accelerometer and gyroscope sensor. The x, y and z directions of the MPU-6050 have an accelerometer and gyroscope senor as shown in the diagram – note also the x and y directions are marked on the board.





Accelerometers are devices that measure acceleration - the rate of change of the velocity of an object, measured in units of m/s^2 (meters per second per second) or sometimes expressed in G-forces (g). The G-force for us here on Earth is 9.8 m/s^2. So, when the MPU-6050 is lying on the desk the accelerometer in the z direction will measure 9.8 m/s^2 (or 1 g), and 0 m/s^2 (or 0 g) in the x and y directions.

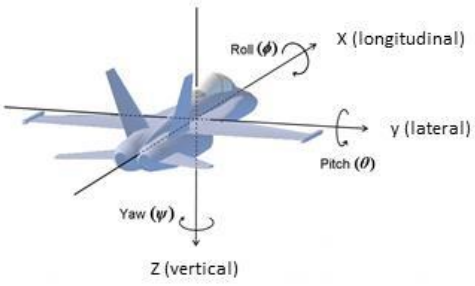
Gyro sensors, also known as angular velocity sensors, are devices that measure angular velocity. Angular velocity is the change in rotational angle per unit of time, generally measured in units of deg/s (degrees per second).

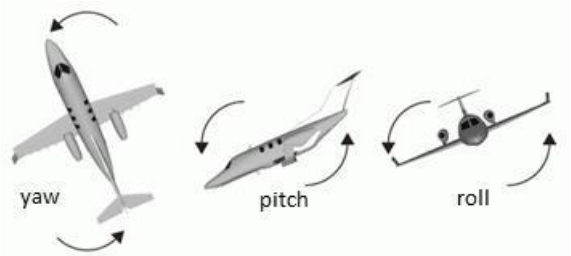
Accelerometers give an absolute way of finding angles (because gravity can be used as a reference direction), but are sensitive to noise (vibration). Gyroscopes are less sensitive to vibrations, but tend to drift over time. We need to combine the two to obtain stable measurements (called fusion).

Maths for calculating pitch angle from accelerometers

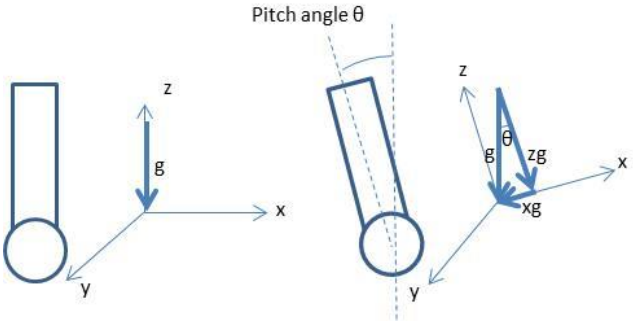
Vehicles that are free to move in three dimensions can change their direction around three axes:

- longitudinal (x) axis (the axis the vehicle is moving in) - motion about this axis is called roll
- lateral (y) axes - motion about this axis is called pitch
- vertical (z) axis - motion about this axes is called yaw





On earth the acceleration due to gravity g (g=9.8m/s^2) is always acting downwards, so we can use this as a reference direction to calculate pitch, roll and yaw angles from the accelerometers.



To keep our robot stable we are only interested in the pitch angle (we use this angle to drive the robot wheels to keep the robot upright).

When the robot is upright and stationary, the accelerometer in the z direction measures the full g value and the x and y accelerators measure zero. However when the robot tilts (pitches), the z and x accelerators each measure a component of g (zg and xg). So to calculate the pitch angle θ we can use trigonometry:

$$xg = g \times Sin(\theta)$$
$$\theta = \sin^{-1}(\frac{xg}{g})$$

In Arduino code this is:

```
acc_x_data = Wire.read()<<8|Wire.read();  // read the x accelerometer from MPU-6050
pitch_angle_acc = asin(acc_x_data/9.8) * 57.29578 ;  // pitch angle in degrees
```

Note: asin() is the Arduino function for $\sin^{-1}$(), it returns results in Radians. To convert to degrees multiply by $\frac{360}{2\pi}$ = 57.29578.

Maths for calculating pitch angle from gyros

We said earlier that accelerometers are sensitive to noise (vibration), and although gyros are less sensitive to noise, they tend to drift over time. So if we combine the two we obtain a stable pitch angle measurement.

Therefore we also need to calculate the pitch angle from the gyros. Again we only need to measure the pitch angle (not roll and yaw), and we can do this by taking gyro measurements about the y axis.

Gyros measure angular velocity (in degrees/sec). So to calculate an angle through which an object has rotated we need to integrate the gyro measurement over time (just like integrating velocity to obtain distance travelled).

To do this integration numerically we sample (take measurements) the y gyro on the MPU6050 at regular intervals (at period = T secs, e.g. at 4 milliseconds), add them all up and multiply by the period T:

$$Angle_{rotated} = T \times \sum_{t=start}^{current\ t} Gyro_{measurement}$$

A very simple way to do this calculation is to add the previous value $of\ Angle_{rotated}$ to the new $Gyro_{measurement} \times T$. In Arduino code this is (note the += in the second line of code):

```
gyro_y_data = Wire.read()<<8|Wire.read();      // read the y gyro from MPU-6050
pitch_angle_gyro += gyro_y_data * T;           // integrate to obtain an angle
```

Putting it all together - Obtaining a stable pitch angle

We have now calculated the pitch angle from both the accelerometers and the gyros. We now need to combine these to obtain a stable pitch angle, called fusion.

Over the short term, we want to use the gyro data, because it is not susceptible to noise (vibration), but over the long term we want to use the accelerometer data as it does not drift.

There is an optimal way to do this called Kalman filtering, but this is complicated to tune and requires a lot of code. A simpler method which is nearly as good for our robot is called a Complementary filter.

A complementary filter weights the pitch angles from the acc and gyro and adds them together:

$$pitch\_angle = (1 - \alpha) \times pitch\_angle\_gyro + \alpha \times pitch\_angle\_acc$$

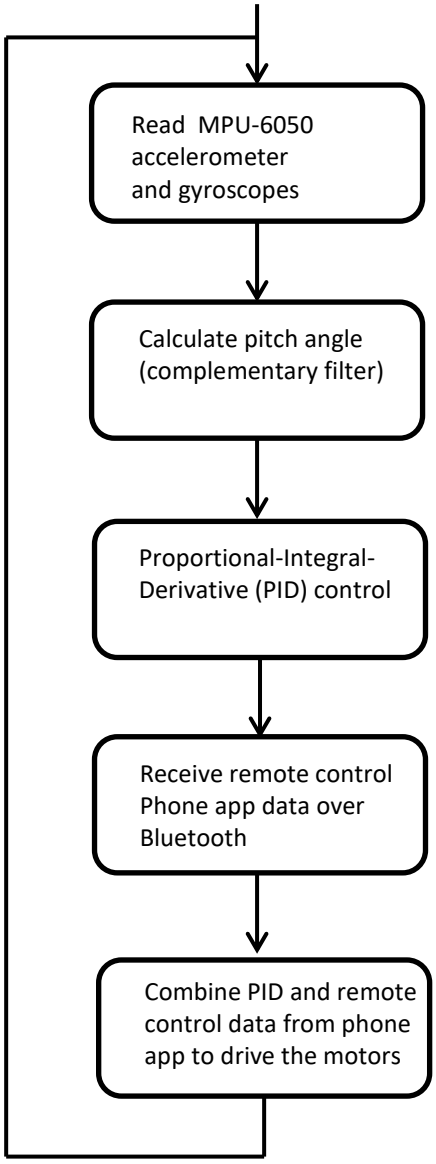
and then sets the *pitch_angle_gyro* to the new filtered *pitch_angle*, so the drift in the gyro pitch angle is corrected over time. The weighting factor α is made small so that individual measurement noise in the acc derived pitch angle does not have a large affect and averages out over time.

Putting all the acc and gyro pitch angle Arduino code together:

```
// calculate pitch angle using acc data
acc_x_data = Wire.read()<<8|Wire.read();  // read the x accelerometer from MPU-6050
pitch_angle_acc = asin(acc_x_data/9.8) * 57.29578;   // pitch angle in degrees

// calculate pitch angle using gyro data
gyro_y_data = Wire.read()<<8|Wire.read(); // read the y gyro from MPU-6050
pitch_angle_gyro += gyro_y_data * T;      // integrate to obtain an angle

// complementary filter
pitch_angle = pitch_angle_gyro * 0.9996 + pitch_angle_acc * 0.0004;

pitch_angle_gyro = pitch_angle;           // corrects the pitch_angle_gyro drift
```

Balancing Robot Program Blocks

The main program blocks for the balancing robot are shown here. If you look carefully at your Arduino sketch you might be able to make out these blocks in the loop() function.



This code runs every 4ms (250 times per second).

This blocks reads the three accelerometers and three gyroscopes (along the x, y, and z axises) from the MPU-6050 sensor.

This block calculates the pitch angle (how much the robot is leaning) of the robot. It combines angle calculations from the accelerometers and the gyroscopes to obtain a stable pitch angle, through a process called a complementary filter. Trigometry and integration is used to calculate the angle – see the maths section.

Using the angle calculated above, this block determines how the motors should be controlled to keep the robot upright. It uses an algorithm called Proportional-Integral-Derivative (PID) control. It is a common algorithm used in many control situations.

This block receives the remote control data from the phone app joystick (and buttons). The x and y coordinates of the joystick position are converted into a length (using Pythagoras) and an angle (using the trig function Tan) which are then used to determine the speed and direction of the robot.

This block combines the PID motor control (which keeps the robot upright) with the controls received from the phone app and drives the motors so that the robot balances and moves according to the phone app joystick. The motors are controlled using Pulse Width Modulation (PWM).