Hello Everybody, I've been working on an arduino based quadcopter for the last few months. A key component to make the quadcopter balance is an orientation sensor that periodically reports the yaw/pitch/roll which can be used as input to a PID controller that adjusts the RPM of the quadcopter motors. The Invensense MPU6050 is a popular and cheap sensor that has an accelerometer, gyroscope and a temperature sensor on board. The accelerometer reports the components of the acceleration vector along the local coordinate system of the sensor. If the sensor itself is not accelerating, the only accelerating force that the sensor is subjected to is gravity. Since the gravity vector points straight down, the components of the gravity vector along the axis of the local coordinate system of the sensor can be used to estimate the orientation of the sensor (in the absence of other accelerating forces).

The gyroscope reports the angular velocity along the axes which can be integrated to obtain the current orientation. As has been reported in numerous articles on the web, gyro measurements are subject to drift, while accelerometer measurements are noisy. In order to obtain optimal orientation measurements, one must combine the results of the gyroscope and the accelerometer. This is typically done using a complementary filter or a kalman filter. The complimentary filter is much simpler to implement and produces results that are very close to that of the kalman filter.

The MPU6050 also has a MPU (Motion Processing Unit) that performs sensor fusion on-board (using some unknown algorithm) and reports the orientation in yaw/pitch/roll or quaternion format.

When I started working on the quadcopter, I read a lot of articles about using the MPU6050 about determining orientation. Some of the articles used the raw accelerometer and gyroscope readings and performed their own sensor fusion. Others used the data from the MPU directly. I was not sure if one approach is better than the other. In the last couple of weeks, I've developed an oscilloscope like application using QT and plot the data from the MPU and the results of the sensor fusion. In this article, I'd report my results and also provide a link to the code for my oscilloscope application and the Arduino code.

Performing Sensor Fusion:

Sensor fusion involves combining the data obtained from the accelerometer and the gyroscope. The accelerometer reports the components of the acceleration vector along the axis of the coordinate system attached to the sensor. The angle of rotation along the x/y axis can be obtained from these components. The accelerometer can't be used to obtain the angle about the z axis (yaw). There is confusion about the trigonometry that should be used to obtain the pitch and roll. Some people recommend simply dividing the x and y components by the z component. I found that this doesn't produce results that agree with the pitch/roll reported by the MPU. The formula that worked for me is:

angle\_x =  atan(ay/(ax[sup]2[/sup] + az[sup]2[/sup])\*RAD\_TO\_DEGREES

angle\_y =  atan(ax/(ay[sup]2[/sup] + az[sup]2[/sup])\*RAD\_TO\_DEGREES

The accelerometer produces instantaneous estimates of the pitch/roll, but these estimates are subject to a lot of noise. Therefore, the pitch/roll angles from the accelerometer can’t be used directly without some filtering.

The gyroscope measures the angular velocity measured in degrees per second. In order to obtain angle of rotation about x/y/z axes, the angular velocity measurements need to be integrated over time. This is easily done by multiplying the current angular velocity by the time elapsed since the last measurement and accumulating the product. While gyro measurements are very stable and noise free, they have a tendency to drift over time. This can be seen in the figure below.

Fortunately, the accelerometer and gyroscope measurements can be combined using a kalman filter or a complimentary filter. There are numerous excellent tutorials about the Kalman filter available on the web, so refer to any of those. The complimentary filter is much simpler and involves a linear combination of the accelerometer and the accumulated gyroscope data with most of the weight being placed on the gyroscope measurements. This tends to produce measurements that closely follow the ones produced by the on-board MPU.

Note that the gyroscope and accelerometer measurements must be scaled before being used. This is not so important for the accelerometer measurements as they are divided and the scale factor cancels out, however for the gyroscope measurements, this is quite important. The scale factor can be obtained as follows:

uint8\_t READ\_FS\_SEL = mpu.getFullScaleGyroRange();

GYRO\_FACTOR = 131.0/(READ\_FS\_SEL + 1);

This can be seen in the table on page 12 of the MPU6050 user manual available from Invensense.