IMU (Gyroscope & Accelerometer)
Can-Sat

Introduction

This documentation shows the software algorithm for making a small IMU (inertial measurement unit) using a gyroscope and an accelerometer, the final output of that IMU is roll (rotation around x-axis) and pitch (rotation around y-axis).

The reason why using both gyro scope and accelerometer for measuring the same quantities is that every device of them has its disadvantages which make its reading alone inaccurate and that what will be discussed later.

Gyroscope

A gyroscope is a device that measures angular velocity and to obtain angels integration over time is needed which makes some drifts due to approximations made to integrate with microcontrollers and the fact of any error will be accumulated over timem that's why gyroscope results for angels is not sufficient.

IDG500 Breakout Gyroscope was used, it's a two axes gyroscope with analog output with a sensitivity of 2mV/°/s and a reference of 1.35 V which means that if the angular velocity = zero the output would be 1.35V, to be mentioned that this value (1.35) is changeable due to ambient temperature so it's not a fixed value to initialize.

To solve the problem of initialization a reading is taken for each axis in the beginning of the program while gyroscope is in static state to take the zero reference for each axis and any reading taken after this would subtract that reference to get the real reading.

Here is the Arduino code for the gyroscope only with comments to describe every line:

```
unsigned long lastTime;
unsigned long now;
unsigned long timeChange;
float x, y;  //angular velocities
float ax,ay;  //angels
int xzero,yzero;  //zero refrences


void setup()
{
  Serial.begin(9600);      // sets the serial port to 9600
  xzero=analogRead(A0);    //takes the zero reference once in the beginig of the program
  yzero=analogRead(A1);    //takes the zero reference once in the beginig of the program
}

void loop()
{
   now = millis();   //gets time from begining in milli seconds
   timeChange = now - lastTime;  //calculates time sample (delta t) for integeration

  x = (analogRead(A0)-xzero)/0.4092;
  //real reading = reading-refrence , and the 0.4092 is to convert readings into angular velocity
  //as 5v gives 1023 then 2mv gives 1023*.002/5 which is .4092 and this value of reading equals 1 degree/second
 if (x==-1 |x==-2 |x==1 |x==2)
 {
   x=0;   // this to ellimnate small errors which may have effects when integerated
 }

  ax += (x*3.0*timeChange/1000); //integrates angular velocity to give angels
  if (ax>180)
  {
    ax = -360+ax ;  // to keep angels between -180>>180
  }
  else if (ax<-180)
  {
    ax = 360+ax ;  // to keep angels between -180>>180
  }


   // y is same as x
  y = (analogRead(A1)-yzero)/0.4096;
if (y==-1 |y==-2 |y==1 |y==2)
 {
   y=0;
 }
  ay += (y*3.0*timeChange/1000);
if (ay>180)
  {
    ay = -360+ay ;
  }
  else if (ay<-180)
  {
    ay = 360+ay ;
  }

  Serial.print("angels of x, y: ");
  Serial.print(ax, DEC);    // print the rotational rate in the X axis
  Serial.print(" ");        // prints a space between the numbers
  Serial.println(ay, DEC);  // print the rotational rate in the Y axis
     lastTime = now;   //to be used in the next loop to be subtracted from the time get time sample

}
```

Accelerometer

Accelerometer is a device that measures acceleration, and in static state it gives the acceleration of the gravity for the axis pointing to the ground, and by using this property angels could be calculated in static state.

Because accelerometer is sensitive to noise low pass filter is used to illuminate noise produced by dynamic state.

ADXL335 accelerometer was used, it’s a 3 axes accelerometer with analog output with sensitivity of 300mV/g. It has the same initialization problem of gyroscope and was treated by the same way.

Here is the Arduino code for the accelerometer only with comments to describe every line:

```
float x, y, z;  //accelerations
float fx, fy, fz;  //filtered acceleration
int xzero, yzero,zzero;  //initial refrences
const float g=9.81;  //gravity constant
float ax, ay;   //angels
float alpha=.2;  //wieght of smoothing

void setup()
{
  Serial.begin(9600);      // sets the serial port to 9600
  xzero=analogRead(A0);  //zero refrence itialization
   yzero=analogRead(A1);  //zero refrence itialization
    zzero=analogRead(A2)-62;  //considering that at static state when testing the z axis would point to ground
}

void loop()
{
  x = (analogRead(A0)-xzero)*g/61.44;
  //real reading = reading-refrence , and the 61.44 is to convert readings into acceleration in m/s^2
  //as 5v gives 1023 then 300mv gives 1023*.3/5 which is 61.44 and this value of reading equals 1 g whic is 9.8 m/s^2
  y = (analogRead(A1)-yzero)*g/61.44;    //same for x
  z = (analogRead(A2)-zzero)*g/61.44;    //same for x

    fx = fx  + alpha*(x-fx);
    fy = fy + alpha*(y-fy);
    fz = fz +alpha*(z-fz);


 ax = atan2(fx, sqrt(fy*fy + fz*fz))*180.0/3.14;  //this equation is to caculae anges from accelerations
 ay = atan2(fy, sqrt(fx*fx + fz*fz))*180.0/3.14;  //this equation is to caculae anges from accelerations

  Serial.print("angels around x, y are");
  Serial.print(ax, DEC);  // print the acceleration in the Z axis
  Serial.print(" ");       // prints a space between the numbers
  Serial.println(ay, DEC);  // print the acceleration in the Z axis

  delay(100);              // wait 100ms for next reading
}
```

Sensor Fusion

As discussed before each sensor has its disadvantages, for gyroscope it has drifts over time due to integration but it still reliable in short term. And for accerlerometer it’s sensitive to noise and not reliable if not in static state but it’s more reliable than gyroscope in long term as its drifts is not due to time.

For that reason a fusion of the two sensors was done with a complementary filter using a simple algorithm that combines readings from both sensors but with major percent for gyroscope as it’s more reliable in short term but the values for the integration process is accumulated over the filtered values to avoid accumulation over wrong data.

The algorithm was as simple as that:

```
filtered_angel = (filtered_angel + rotational_velocity*timeChange)*0.9 +
angel_from_accelerometer*0.1;
```