# A Gyroscopic Data based Pedometer Algorithm

Sampath Jayalath
Department of Electrical and Computer Engineering
Sri Lanka Institute of Information Technology
New Kandy Rd, Malabe, Sri Lanka
adasdjsampath@gmail.com

Nimsiri Abhayasinghe
Department of Electrical and Computer Engineering
Sri Lanka Institute of Information Technology
New Kandy Rd, Malabe, Sri Lanka
nimsiri.a@sliit.lk

*Abstract*—**Accuracy of step counting is one of the main problems that exist in current Pedometers, especially when walking slowly on flat lands and performing different activities, such as climbing up and down stairs and walking on inclined planes. Although accelerometer based pedometers provide a reasonable accuracy when walking at higher speeds, the accuracy of them are not sufficient at slow walking speeds and performing different activities. This paper proposes a novel algorithm to detect steps using single-point gyroscopic sensors embedded in mobile devices. Preliminary analysis of data collected in different environments with the involvement of male and female volunteers indicated that gyroscope alone provides sufficient information necessary for accurate step detection. Algorithm was developed based on the gyroscopic data in conjunction with zero crossing and threshold detection techniques. The results proved that gyroscope based step detection algorithm provide a high accuracy when performing different activities and at slow paced walking.**

***Keywords—Pedometer algorithms; gyroscopic data; single-point sensors; off-the-shelf devices; mobile applications;***

## I. Introduction

Modern medical researches highlight that pedometers support not only to physical body but mental activities of human beings to a greater extent[1],[2],[3]. Low cost pedometers help to improve the motivation of the walker [4], indoor navigation, activity recognition and for various applications in the field of health care. Pedometers can be used to detect steps from vertical acceleration of the human body. This works under two systems of mechanism. One is of mechanical based and other being of the electrical based accelerometers. Modern pedometers are generally based on MEMS (micro-electromechanical systems) accelerometer, mostly 1-axis, but the use of 2-axis and 3-axis accelerometers, gyroscopes and magnetometers improves precision and releases some utilization constraints e.g. positioning of the pedometer. The accuracy of these systems is at an acceptable level but not perfect due to various drawbacks [5]. Applications of pedometer are now upgraded and can be found in mobile devices. It is obvious with the application of pedometers to mobile devices, has now improved the standards of healthcare applications.

The approaches of some pedometer algorithms proposed by researchers are discussed in the background section including their features and drawbacks.

## II. Background

S.E Crouter et al. [6] have compared the accuracy and reliability of 10 pedometers available in the market. These pedometers were based on mechanisms like, accelerometer, metal-on-metal and magnetic reed proximity switch. It is important to notice that all the testing was done at normal walking speeds. Their conclusion was that accuracy of pedometers was highly subjective upon the internal mechanism and sensitivity. But they have failed to measure the accuracy of pedometers when walking slowly and performing different activities like ascending and descending stairs.

Comparative study with commercially available pedometers done by Jerome and Albright [7] has shown accuracies are poor with a minimum average absolute error value of 13%. Their conclusion was that none of the pedometers can be used for research purpose or general usage.

Wasiq Waqar et al. [8] have developed a pedometer based on accelerometer for their “Indoor Positioning System” which consists of a preset threshold based peak detection method to identify a valid step and step cycle pattern detection method to discard invalid steps due to instantaneous readings of the accelerometer. It should be noted that the results of the pedometer may change with different individual walking patterns and speeds due to preset threshold.

Melis Oner et al. [9] have implemented another step detecting algorithm for their “Early detection of the falling event system”. Step detection of this particular algorithm relies on the detecting peaks within a period in the data produced by the accelerometer sensor during walking. They were able to achieve higher accuracies during higher speeds of walking and with the mobile based pedometer placed fixed and loose in the pocket. However, their algorithms failed to count steps accurately during slow paced walking.

Mi-hee Lee et al. [10] were able to achieve 99% accuracy in their portable acceleration sensor module with some advanced processing like FFT, Fuzzy C and statistical calculations. But they have agreed finally that their system doesn’t process data in real time, inability to measure steps during activities like ascending and descending stairs walking and need for an efficient device to carry out processing.

A.M. Cavalcante et al. [11] have developed a pedometer to be used with their research on “Real-time indoor tracking on mobile devices". System compares mean values of acceleration samples in conjunction with a sliding window mechanism to detect steps. Their conclusion was that a proper sampling rate, sliding window and quality sensors embedded in mobile devices are a major requirement to detect steps accurately.

According to Garcia et al. [12] comparative study on the accuracies of mobile phone based pedometers with the commercially available pedometers concluded that mobile phones provide a competitive performance against the commercially available pedometers. Further both indicate less accuracy when at slower speed and high accuracy while in faster walking.

Lim et al. [13] have proposed a pedometer based on gyroscope but not discussed deeply on its practical usage. Zhong et al. [14] has also discussed pedometer based on accelerometer sensor attached to foot with processing done on a PDA and were able to achieve accuracies more than 90%. But none of these pedometers are inconvenient to be used with any application as sensors are attached to the body.

According to G. Boyce et al. [15] comparative study on the accuracies of mobile phone based pedometer technologies which are freely available in the market, concluded that mobile phone based applications lag behind the use of a commercially available pedometer when determining step count. Further manipulation of settings is required to improve the accuracy of step counting for one activity level, but recalibration is required as intensity of activity changes.

M. Ayabe et al. [16] have conducted an important research on accuracies of pedometers during ascending and descending stairs. They have used one spring-levered and two piezo-electric pedometers to test the accuracy. Their conclusion was that pedometers can assess the number of step accurately within an acceptable range of measurement error during the stair climbing activities at a stepping rate of 80 step·min-1 or faster with 18 cm or higher stairs. However, it should be noted that they have highlighted the poor accuracy at slower stepping rates.

Most of the previous researches have identified that pedometers are less valid and reliable during slow walking speeds. This inaccuracy results from smaller vertical movements of the hip which are below the specific pedometer threshold value and are therefore not recorded. Another limitation is that pedometers do not contain an internal clock; therefore it is not possible to determine the intensity or duration of activity performed.

## III. IMPLEMENTED ALGORITHM

The development of this algorithm is based on the experimental results of the research conducted by N. Abhayasinghe et al. on "Indoor Positioning and Indoor Navigation" [17]. This particular algorithm only uses gyroscopic data produced by the mobile device to predict steps. It can be described as follows;

### A. *Initial concept behind the algorithm*

Leg movement during walking shows a sinusoidal behavior. This behavior can be clearly identified by monitoring the angular velocity of the leg. Therefore one axis of the gyroscope provides the information about the movement of the leg depending on the orientation of the device. A small research on "ways of placing the modern mobile devices on a pocket" proved that almost all the users placed their devices vertically in the pocket. Therefore monitoring the gyroscopic x axis data is considered. It is important to notice for different orientations only variable that needed to change is the axis that we are obtaining data from the gyroscope.

### B. *Filtering of raw data*

Initially raw data are filtered using a proper filter that would preserve the properties of walking. A typical walking pace may be around 2 Hz, while running may be double and slow walking may be half of that. A choice of cutoff frequency that accommodates slow-paced activities is a major concern in achieving better accuracy. Therefore choice of the filter was a simple sixth order Butterworth low-pass filter having a cutoff frequency in the range of 0.9 Hz to 3 Hz. Fig. 1 depicts the raw gyro-x value and filtered version (2 Hz) of it.

### C. *Removal of unwanted signal components*

Gyroscopic data takes a sinusoidal behavior after filtering for both steps and instantaneous movement of the device. A sample out technique is used to reduce wrong step counting due to instantaneous movement of the device. According to the experimental results a step occurs on an average of 0.40 to 1.20 seconds depending on the intensity of walking. Therefore once a step is detected, the algorithm can be set to eliminate any signal that can be counted as a valid step before the average step time.

### D. *Identification of key features of the filtered signal*

The main idea for the step detection in this algorithm relies on detecting zero crosses in the data produced by the filtered gyro-x. Fig. 2 illustrates the consecutive zero crosses that occurs during a single stride. In addition to zero crossing an adaptive peak threshold is used to validate a step. An individual can train the algorithm to learn the minimum signal peak that can be used to realize a valid step, especially when moving slowly as possible and descending stairs. So this peak threshold is used as a parameter to validate a step. Threshold peak detection helps to avoid instantaneous and small movements of the device. Further this threshold peak can be adjusted so that the algorithm is capable of detecting steps when the device is placed in different pockets which are loose or tight (Signal strength differs with the environment where the device is placed). Fig. 2, Fig 3 and Fig 4 depict the strength of the gyro-x signal (Upward-peak signal) when walking (Average = 1.3rad/s), ascending (Average = 1.6rad/s) and descending (Average = 0.9rad/s) stairs for an individual. It is important to note that these averages vary with different patterns and intensities of walking.

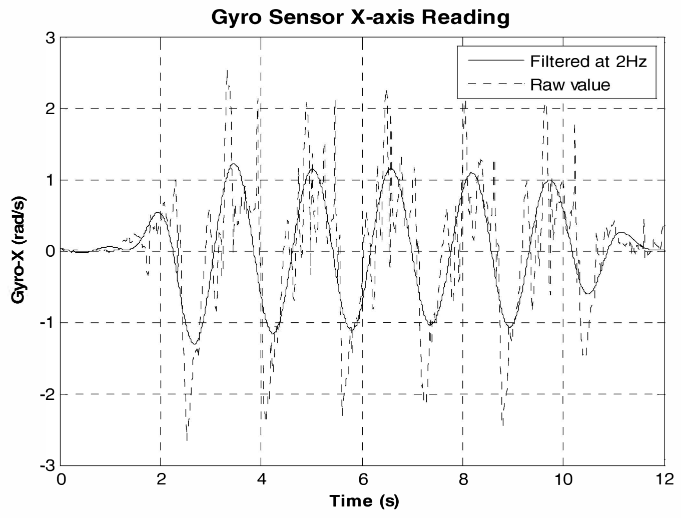


Figure 1. Gyroscopic X Axis reading (Raw and Filtered Value at 2 Hz)

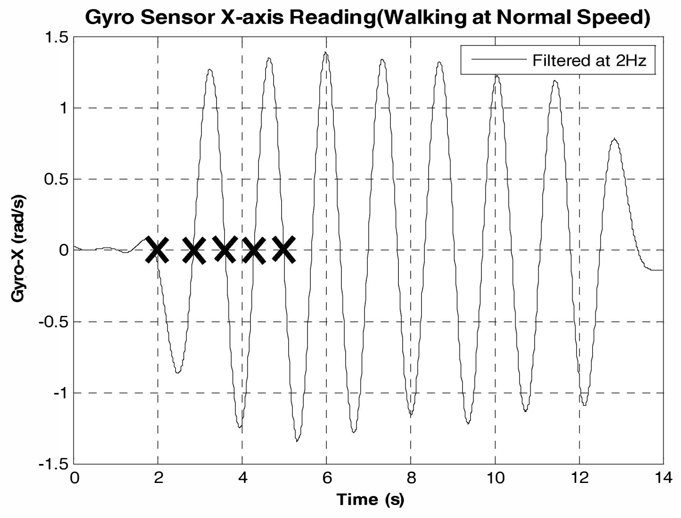


Figure 2. Strength of the gyro-x signal when walking at a medium pace

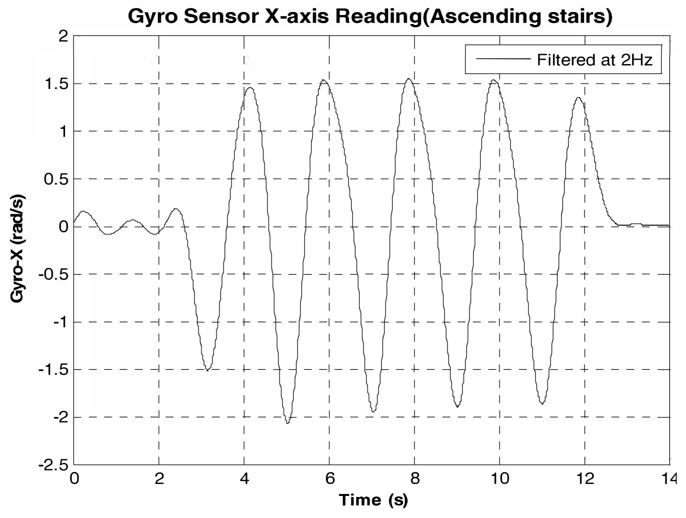


Figure 3. Strength of the gyro-x signal when ascending stairs

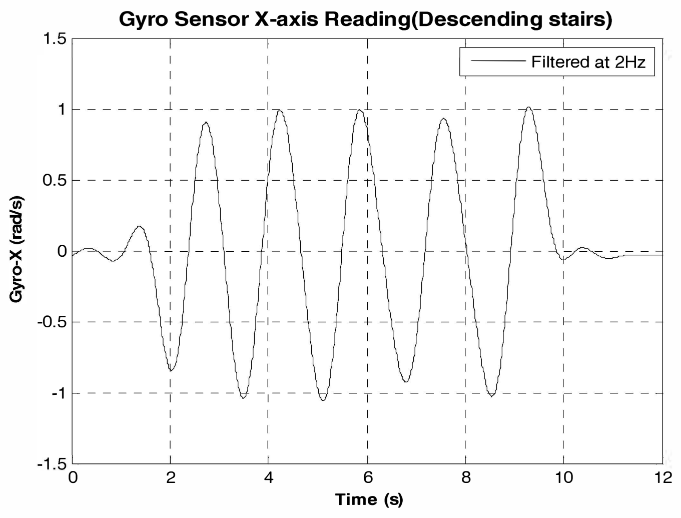


Figure 4. Strength of the gyro-x signal when descending stairs

### E. *Experimental test and result*

Apple devices are used to test the Pedometer algorithm, which provide a higher data rate and accuracy when extracting data from the embedded sensor. The proposed algorithm was implemented on iPhone 4S and iPod 4G.

Testing of the algorithm was done with the involvement of 5 male and 5 female members with random heights and weights. Since this algorithm aimed to detect steps irrespective of age, sex and various physical aspect of human being, we were not much concerned of varieties of the people involved sex etc. When the application starts we assume that the initial orientation of the device is closer to zero degrees. Then users were instructed to perform different activities, such as climbing up and down stairs and walking on inclined planes. Sample of counted steps for an individual is tabulated in Table I.

TABLE I. EXPERIMENTAL RESULTS OF THE PROPOSED ALGORITHM FOR AN INDIVUAL PERFORMING DIFFERENT ACTIVITIES

| Activity | **Actual number of steps** | **Calculated by the algorithm** |
|---|---|---|
| Walking slowly on flat lands | 27 | 26 |
| Walking faster on flat lands | 49 | 49 |
| Climbing up stairs | 11 | 11 |
| Climbing down stairs | 11 | 11 |
| Walking on inclined plane(up) | 40 | 40 |
| Walking on inclined planes(down) | 43 | 41 |

A Similar reading profile was maintained for each individual who participate the testing process. Combined results were tabulated in Table II.

TABLE II. EXPERIMENTAL RESULTS OF THE PROPOSED ALGORITHM FOR ALL PARTICIPANTS

| Activity | Actual number of steps | Calculated by the algorithm | Accuracy as a percentage |
|---|---|---|---|
| Walking slowly on flat lands | 285 | 276 | 96.84 |
| Walking faster on flat lands | 491 | 485 | 98.77 |
| Climbing up stairs | 110 | 107 | 97.27 |
| Climbing down stairs | 110 | 104 | 94.54 |
| Walking on inclined planes(up) | 433 | 425 | 98.15 |
| Walking on inclined planes(down) | 422 | 410 | 97.15 |

Further this algorithm was tested for accuracy at different walking speeds and stepping speeds with the involvement of multiple male and female volunteers. Table III shows the variation of the accuracies at different walking speeds on a flat land while accuracies at different stepping speeds are tabulated in Table IV and Table V.

TABLE III. ACCUARACY OF THE ALGORITHM AT DIFFERENT WALKING SPEEDS

| Speeds of walking | Actual number of steps | Calculated by the algorithm | Accuracy as a percentage |
|---|---|---|---|
| 50 steps/min | 259 | 255 | 98.45 |
| 75 steps/min | 378 | 370 | 97.88 |
| 100 steps/min | 510 | 499 | 97.84 |
| 125 steps/min | 625 | 620 | 99.20 |
| 150 steps/min | 745 | 739 | 99.19 |

TABLE IV. ACCUARACY OF THE ALGORITHM AT DIFFERENT STEPPING UP SPEEDS

| Stepping up speeds | Actual number of steps | Calculated by the algorithm | Accuracy as a percentage |
|---|---|---|---|
| 50 steps/min | 110 | 108 | 98.18 |
| 75 steps/min | 110 | 106 | 96.36 |
| 100 steps/min | 110 | 108 | 98.18 |
| 125 steps/min | 110 | 106 | 96.36 |
| 150 steps/min | 110 | 109 | 99.09 |

TABLE V. ACCUARACY OF THE ALGORITHM AT DIFFERENT STEPPING DOWN SPEEDS

| Stepping down speeds | Actual number of steps | Calculated by the algorithm | Accuracy as a percentage |
|---|---|---|---|
| 50 steps/min | 110 | 105 | 95.45 |
| 75 steps/min | 110 | 107 | 97.27 |
| 100 steps/min | 110 | 106 | 96.36 |
| 125 steps/min | 110 | 105 | 95.45 |
| 150 steps/min | 110 | 108 | 98.18 |

Overall accuracy of the algorithm was above 94%. It is evident from the results that climbing down stairs registered a low percentage of accuracy. This is mainly due to the weak signal strength by the user. Apart from that other errors were due to weak signal strength of different individuals at the start and end of the travel.

### *F. Advantages of the Proposed Algorithm*

There are several advantages in gyroscopic data based step detection algorithm. One main advantage is the ability to detect steps at slow speeds for both walking and stepping up and down. This is achieved by setting a proper filtering process along with an adaptive peak threshold set by the user. It is important to notice that accuracies of pedometers at slow speeds of activity intensities were a major requirement for both research activities and consumers of pedometers.

In addition to those, the algorithm depends only on the data of a single axis of the gyroscope. This provides lesser computations and ability to integrate this function in research areas like Indoor Navigation.

Further algorithm is implemented in currently available high end mobile devices, which is more convenient to be used with any application as sensors are not attached to the body.

### *G. Future Work*

In this paper we discuss the algorithm used to detect steps, even at slow walking speeds. In this particular algorithm user may need to change threshold to be consistent with accurate step counting as the environment that places the device changes. So designing a neural network that learns different thresholds with the changing environments can solve this inconvenience.

The error percentage of the algorithm is due to the fact that it is detecting the movement of the phone to validate a step. So we will be working on an improved solution to avoid such false counts.

Integration of the Barrow meter with the proposed algorithm may provide a better solution to inaccuracies found during stepping down due to weak signal strength.

Further integrating an activity recognition algorithm with the proposed algorithm may increase the accuracy during different activities.

## IV. CONCLUSIONS

It is evident from the results of past researches that most of the pedometer algorithms perform very poorly at slow speeds of walking (50-70 steps/min) and when performing different activities at varying speeds. Therefore the need of developing a reliable algorithm which performs at any walking speed and with different walking patterns was a major requirement among modern researchers and consumers of pedometers. Key points of the algorithm presented in this paper consist of engineering techniques like noise removal, zero crossing and threshold detection. Further it makes use of the natural rotational motion of the leg of the human being to predict steps rather than the linear motion of the body. Overall accuracy of the algorithm was above 94%, in which accuracies at slow speeds of walking was remarkably high. It is also important to notice that this accuracy can be further improved with proposed engineering techniques.

## ACKNOWLEDGMENT

The authors would like to thank Dr.Iain Murray from Curtin University, Australia for providing all hardware and software accessories required to accomplish this task and all the volunteer participants (students of Sri Lanka Institute of Information Technology, Malabe, Sri Lanka) for their participation in the experiments conducted.

## REFERENCES

[1] A.Z. LaCroix, S.G. Leveille, J.A. Hecht, L.C. Grothaus and E.H. Wagner, "Does walking decrease the risk of cardiovascular disease hospitalizations and death in older adults?," Journal of the *American Geriatrics Society*, vol. 44, no.2, pp.113–120, Feb 1996.

[2] E.M. Simonsick, J.M. Guralnik, S. Volpato, J. Balfour, and L.P. Fried, "Just get out of the door! Importance of walking outside the home for maintaining mobility: Findings from the women's health and aging study," Journal of the American Geriatrics Society, vol.53, no.2, pp.198-203, Feb 2005.

[3] Y. Hatano, "Use of the pedometer for promoting daily walking exercise," International Council for Health, Physical Education, and Recreation, vol. 29, pp.4-8, 1993.

[4] E. Garcia, Hang Ding, A. Sarela and M. Karunanithi, "Can a mobile phone be used as a pedometer in an outpatient cardiac rehabilitation program?," in IEEE/ICME International Conference on Complex Medical Engineering (CME) 2010., 2010, pp.250-253.

[5] D.S. Michaud, E. Giovannucci, W.C. Willett, G.A. Colditz, M.J. Stampfer and C.S. Fuchs, "Physical activity, obesity, height, and the risk of pancreatic cancer," Journal of the American Medical Association, vol. 286, no.8, pp. 921–929, Aug 2001.

[6] S.E. Crouter, E. Scott, P.L. Schneider, M. Karabulut and D.R. Bassett, "Validity of 10 electronics pedometers for measuring steps, distance and energy cost," Medicine and Science in Sports and Exercise, vol. 35, no.8, pp.1455–1460, 2003.

[7] G. J. Jerome and C. Albright. (2011, June). "Accuracy of Five Talking Pedometers under Controlled Conditions." The Journal of Blindness Innovation and Research. [On-line]. 1(2). Available: www.nfb-jbir.org/index.php/JBIR/article/view/17/38 [Oct. 27, 2011].

[8] W. Waqar, A. Vardy and Y. Chen. "Motion Modelling using Smart phones for Indoor Mobilephone Positioning," presented at the 20th Newfoundland Electrical and Computer Engineering Conference, Newfoundland, Canada, 2011.

[9] M. Oner, J.A. Pulcifer-Stump, P. Seeling and T. Kaya. "Towards the Run and Walk Activity Classification through Step Detection- An Androi Application,"in 34th Annual International Conference of the IEEE Engineering in Medicine and Biology, 2012, pp.1980-1983.

[10] M. Lee, J. Kim, K. Kim, I. Lee, S.H. Jee and S.K. Yoo. "Physical Activity Recognition Using a Single Tri-Axis Accelerometer," in Proceedings of the World Congress on Engineering and Computer Science, vol 1, pp.14-17, October 2009.

[11] A.M. Cavalcante, E.B. Souza, J.J. Bazzo, N. Bezerra, A. Pontesand and R.D. Vieira. " A Pedometer-Based System for Real-time Indoor Tracking on Mobile Devices," in XXIX Brazilian Symposium on Telecommunications, Curitiba, Brazil, Oct,2011. [Online]. Available: http://mwsl.unb.br/index.php/pt/publicacoes [Apr. 12,2012].

[12] E. Garcia, H. Ding, A. Sarela and M. Karunanithi, "Can a mobile phone be used as a pedometer in an outpatient cardiac rehabilitation program?," in IEEE/ICME International Conference on Complex Medical Engineering (CME) 2010., 2010, pp.250-253.

[13] Y.P. Lim, I.T. Brown and J.C.T. Khoo, "An accurate and robust gyroscope-gased pedometer," in 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2008. EMBS 2008., 2008, pp.4587-4590.

[14] S. Zhong, L. Wang, A.M. Bernardos and M. Song, "An accurate and adaptive pedometer integrated in mobile health application," in IET International Conference on Wireless Sensor Network, 2010. IET-WSN., 2010, pp.78-83.

[15] G. Boyce, G. Padmasekara, M. Blum. "Accuracy of Mobile Phone Pedometer Technology". Journal of Mobile Technology in Medicine, vol. 1, pp.16-22, June 2012.

[16] M. Ayabe, J. Aoki, K. Ishii, K. Takayama, H. Tanaka "Pedometer accuracy during stair climbing and bench stepping exercises". Journal of Sports Science and Medicine, vol. 7, pp.249-254, June 2008.

[17] N. Abhayasinghe, I. Murray, "A Novel Approach for Indoor Localization Using Human Gait Analysis with Gyroscopic Data," in International Conferenc on Indoor Positioning and Indoor Navigation,November 2012, [Online]. Available : www.surveysurveying.unsw.edu.au/ipin2012/papers.php [Dec.10,2012].

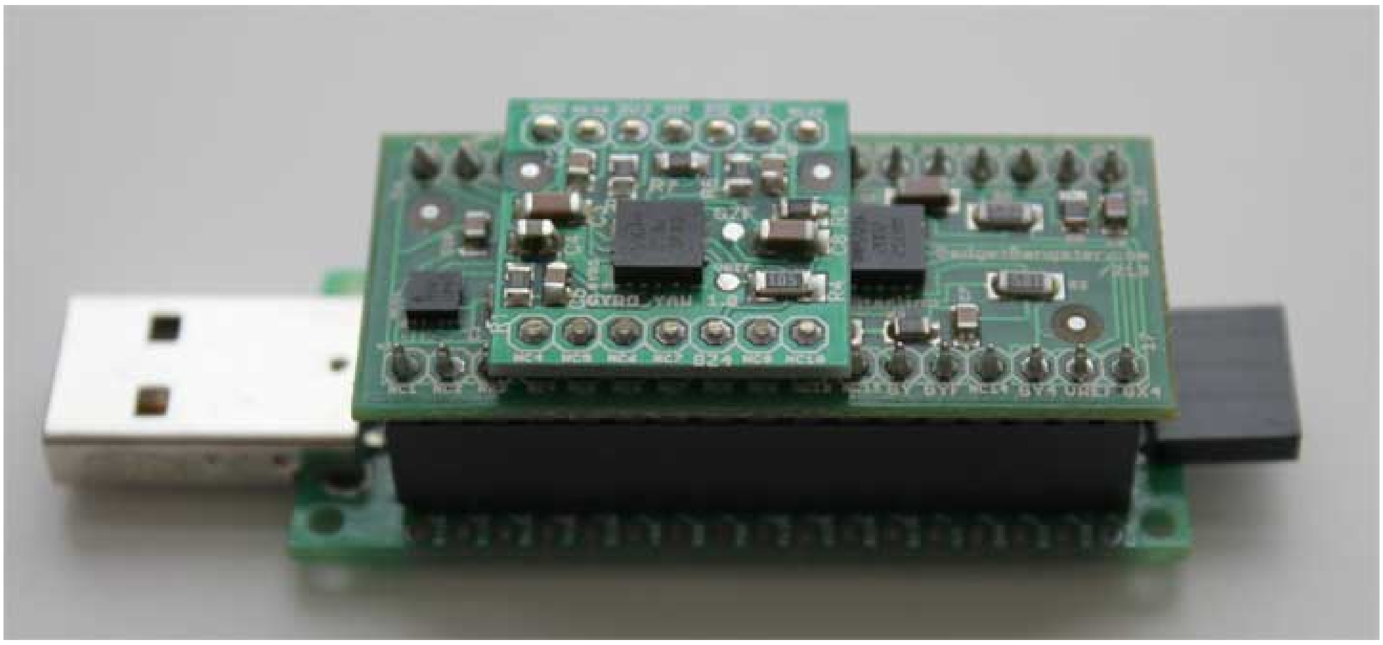
# A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications.

POSTED ON DECEMBER 29, 2009 BY STARLINO

## Introduction

*There's now a FRENCH translation of this article in PDF. Thanks to Daniel Le Guern!*

This guide is intended to everyone interested in inertial MEMS (Micro-Electro-Mechanical Systems) sensors, in particular Accelerometers and Gyroscopes as well as combination IMU devices (Inertial Measurement Unit).



***Example IMU unit:*** *Acc_Gyro_6DOF on top of MCU processing unit UsbThumb providing USB/Serial connectivity*

I'll try try to cover few basic but important topics in this article:

– what does an accelerometer measure
– what does a gyroscope (aka gyro) measure
– how to convert analog-to-digital (ADC) readings that you get from these sensor to physical units (those would be g for accelerometer, deg/s for gyroscope)
– how to combine accelerometer and gyroscope readings in order to obtain accurate information about the inclination of your device relative to the ground plane

Throughout the article I will try to keep the math to the minimum. If you know what Sine/Cosine/Tangent are then you should be able to understand and use these ideas in your project no matter what platform you're using Arduino, Propeller, Basic Stamp, Atmel chips, Microchip PIC, etc. There are people out there who believe that you need co
math in order to make use of an IMU unit (complex FIR or IIR filters such as Kalman filters, Parks-McClellan filter
You can research all those and achieve wonderful but complex results. My way of explaining things require just

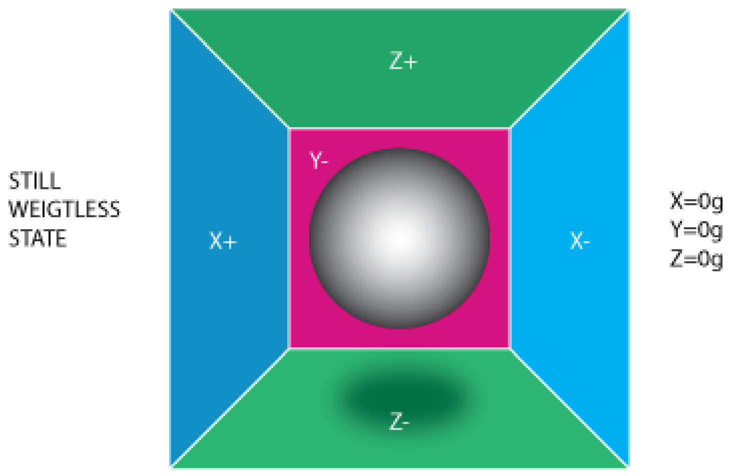
I'll use as an example a new IMU unit that I designed – the Acc_Gyro Accelerometer + Gyro IMU. We'll use parameters of this device in our examples below. This unit is a good device to start with because it consists of 3 devices:

– LIS331AL (datasheet) – analog 3-axis 2G accelerometer
– LPR550AL (datasheet) – a dual-axis (Pitch and Roll), 500deg/second gyroscope
– LY550ALH (datasheet) – a single axis (Yaw) gyroscope (this last device is not used in this tutorial but it becomes relevant when you move on to DCM Matrix implementation)

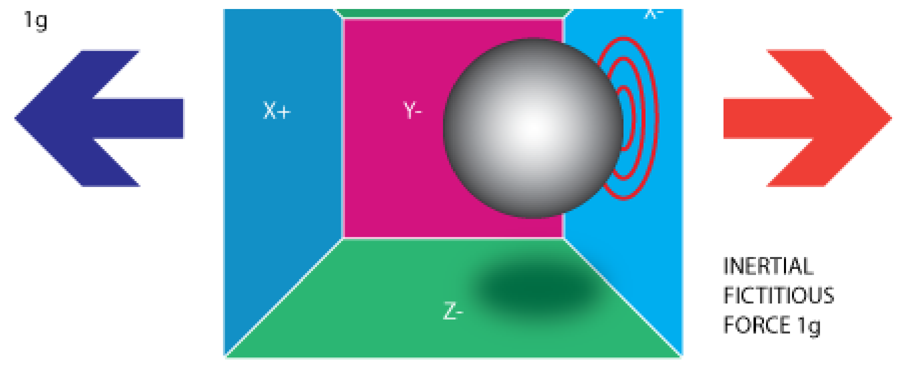
Together they represent a 6-Degrees of Freedom Inertial Measurement Unit. Now that's a fancy name! Nevertheless, behind the fancy name is a very useful combination device that we'll cover and explain in detail below.

## Part 1. Accelerometer

To understand this unit we'll start with the accelerometer. When thinking about accelerometers it is often useful to image a box in shape of a cube with a ball inside it. You may imagine something else like a cookie or a donut , but I'll imagine a ball:

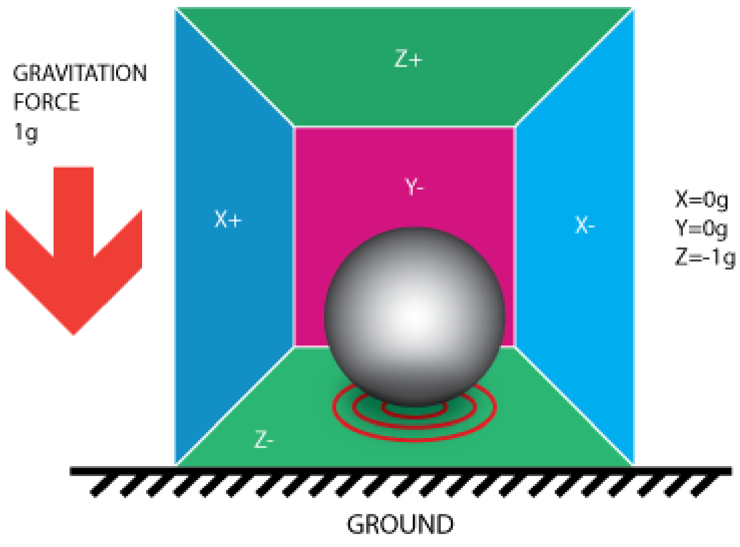


If we take this box in a place with no gravitation fields or for that matter with no other fields that might affect the ball's position – the ball will simply float in the middle of the box. You can imagine the box is in outer-space far-far away from any cosmic bodies, or if such a place is hard to find imagine at least a space craft orbiting around the planet where everything is in weightless state . From the picture above you can see that we assign to each axis a pair of walls (we removed the wall Y+ so we can look inside the box). Imagine that each wall is pressure sensitive. If we move suddenly the box to the left (we accelerate it with acceleration 1g = 9.8m/s^2), the ball will hit the wall X-. We then measure the pressure force that the ball applies to the wall and output a value of -1g on the X axis.



Please note that the accelerometer will actually detect a force that is directed in the opposite direction from the acceleration vector. This force is often called Inertial Force or Fictitious Force . One thing you should learn from this is that an accelerometer measures acceleration indirectly through a force that is applied to one of it's walls (according to our model, it might be a spring or something else in real life accelerometers). This force can be caused by the acceleration , but as we'll see in the next example it is not always caused by acceleration.

If we take our model and put it on Earth the ball will fall on the Z- wall and will apply a force of 1g on the bottom wall, as shown in the picture below:

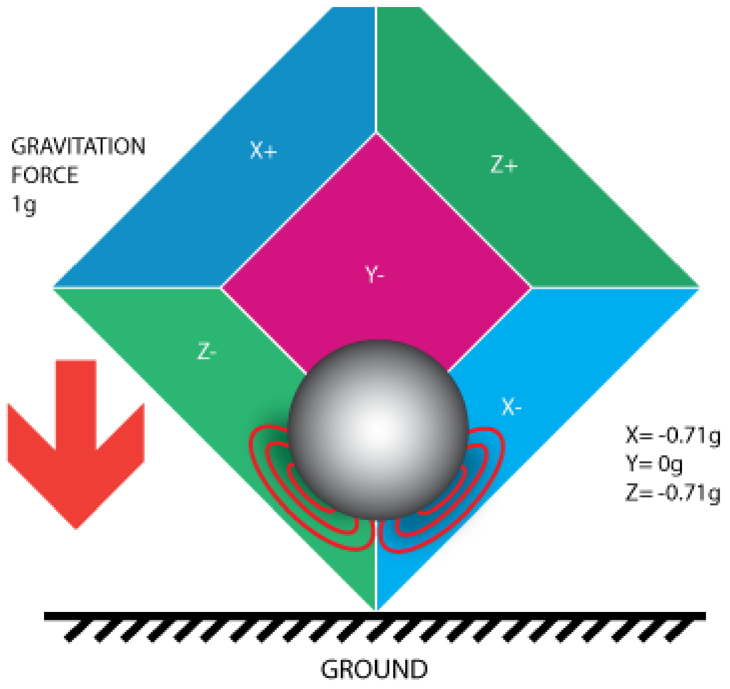


In this case the box isn't moving but we still get a reading of -1g on the Z axis. The pressure that the ball has applied on the wall was caused by a gravitation force. In theory it could be a different type of force – for example, if you imagine that our ball is metallic, placing a magnet next to the box could move the ball so it hits another wall. This was said just to prove that in essence accelerometer measures force not acceleration. It just happens that acceleration causes an inertial force that is captured by the force detection mechanism of the accelerometer.

While this model is not exactly how a MEMS sensor is constructed it is often useful in solving accelerometer related problems. There are actually similar sensors that have metallic balls inside, they are called tilt switches, however they are more primitive and usually they can only tell if the device is inclined within some range or not, not the extent of inclination.

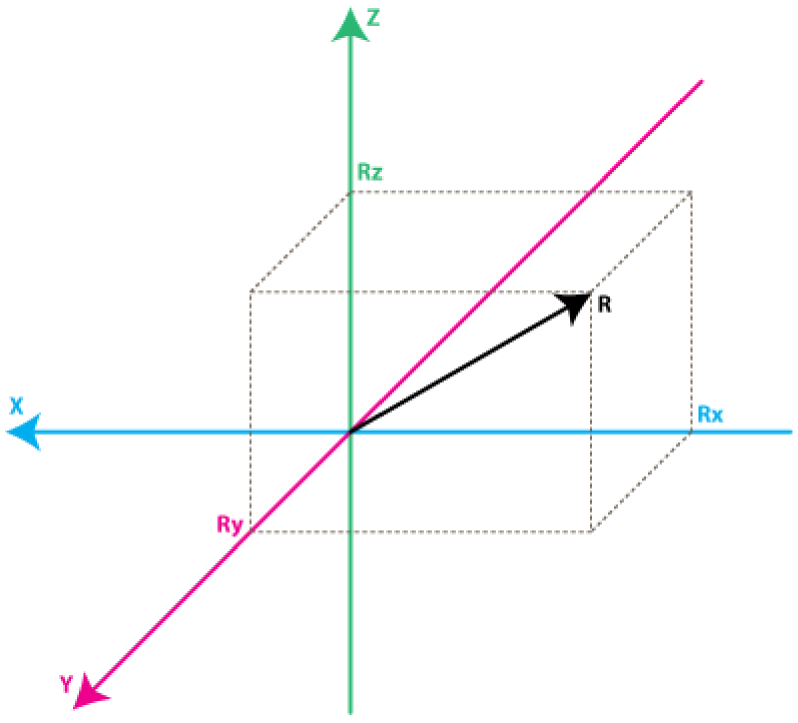
So far we have analyzed the accelerometer output on a single axis and this is all you'll get with a single axis accelerometers. The real value of triaxial accelerometers comes from the fact that they can detect inertial force three axes. Let's go back to our box model, and let's rotate the box 45 degrees to the right. The ball will touch 2

隐私权 - 使用条款



The values of 0.71 are not arbitrary, they are actually an approximation for SQRT(1/2). This will become more clear as we introduce our next model for the accelerometer.

In the previous model we have fixed the gravitation force and rotated our imaginary box. In last 2 examples we have analyzed the output in 2 different box positions, while the force vector remained constant. While this was useful in understanding how the accelerometer interacts with outside forces, it is more practical to perform calculations if we fix the coordinate system to the axes of the accelerometer and imagine that the force vector rotates around us.



Please have a look at the model above, I preserved the colors of the axes so you can make a mental transition f
previous model to the new one. Just imagine that each axis in the new model is perpendicular to the respective
the box in the previous model. The vector R is the force vector that the accelerometer is measuring (it could be

which is basically the equivalent of the Pythagorean theorem in 3D.

Remember that a little bit earlier I told you that the values of SQRT(1/2) ~ 0.71 are not random. If you plug them in the formula above, after recalling that our gravitation force was 1 g we can verify that:

1^2 = (-SQRT(1/2) )^2 + 0 ^2 + (-SQRT(1/2))^2

simply by substituting R=1, Rx = -SQRT(1/2), Ry = 0 , Rz = -SQRT(1/2) in **Eq.1**

After a long preamble of theory we're getting closer to real life accelerometers. The values Rx, Ry, Rz are actually linearly related to the values that your real-life accelerometer will output and that you can use for performing various calculations.

Before we get there let's talk a little about the way accelerometers will deliver this information to us. Most accelerometers will fall in two categories: digital and analog. Digital accelerometers will give you information using a serial protocol like I2C , SPI or USART, while analog accelerometers will output a voltage level within a predefined range that you have to convert to a digital value using an ADC (analog to digital converter) module. I will not go into much detail about how ADC works, partly because it is such an extensive topic and partly because it is different from one platform to another. Some microcontroller will have a built-in ADC modules some of them will need external components in order to perform the ADC conversions. No matter what type of ADC module you use you'll end up with a value in a certain range. For example a 10-bit ADC module will output a value in the range of 0..1023, note that 1023 = 2^10 -1. A 12-bit ADC module will output a value in the range of 0..4095, note that 4095 = 2^12-1.

Let's move on by considering a simple example, suppose our 10bit ADC module gave us the following values for the three accelerometer channels (axes):

AdcRx = 586
AdcRy = 630
AdcRz = 561

Each ADC module will have a reference voltage, let's assume in our example it is 3.3V. To convert a 10bit adc value to voltage we use the following formula:

VoltsRx = AdcRx * Vref / 1023

A quick note here: that for 8bit ADC the last divider would be 255 = 2 ^ 8 -1 , and for 12bit ADC last divider would be 4095 = 2^12 -1.

Applying this formula to all 3 channels we get:

VoltsRx = 586 * 3.3V / 1023 =~ 1.89V (we round all results to 2 decimal points)
VoltsRy = 630 * 3.3V / 1023 =~ 2.03V
VoltsRz = 561 * 3.3V / 1023 =~ 1.81V

Each accelerometer has a zero-g voltage level, you can find it in specs, this is the voltage that corresponds to 0g. To get a signed voltage value we need to calculate the shift from this level. Let's say our 0g voltage level is VzeroG = 1.65V. We calculate the voltage shifts from zero-g voltage as follows::

DeltaVoltsRx = 1.89V – 1.65V = 0.24V
DeltaVoltsRy = 2.03V – 1.65V = 0.38V
DeltaVoltsRz = 1.81V – 1.65V = 0.16V

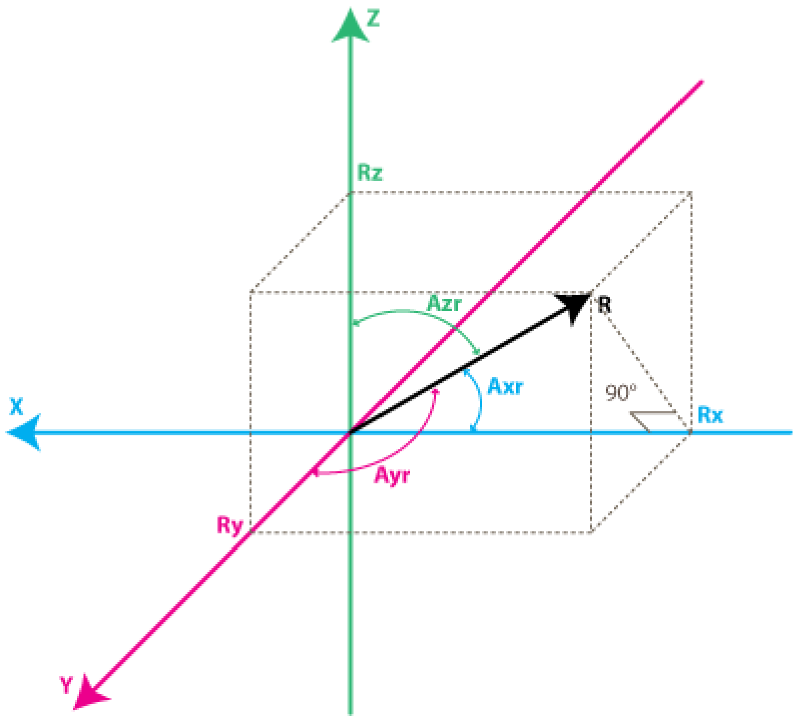
Rx = DeltaVoltsRx / Sensitivity

Rx = 0.24V / 0.4785V/g =~ 0.5g
Ry = 0.38V / 0.4785V/g =~ 0.79g
Rz = 0.16V / 0.4785V/g =~ 0.33g

We could of course combine all steps in one formula, but I went through all the steps to make it clear how you go from ADC readings to a force vector component expressed in g.

Rx = (AdcRx * Vref / 1023 – VzeroG) / Sensitivity **(Eq.2**)
Ry = (AdcRy * Vref / 1023 – VzeroG) / Sensitivity
Rz = (AdcRz * Vref / 1023 – VzeroG) / Sensitivity

We now have all 3 components that define our inertial force vector, if the device is not subject to other forces other than gravitation, we can assume this is the direction of our gravitation force vector. If you want to calculate inclination of device relative to the ground you can calculate the angle between this vector and Z axis. If you are also interested in per-axis direction of inclination you can split this result into 2 components: inclination on the X and Y axis that can be calculated as the angle between gravitation vector and X / Y axes. Calculating these angles is more simple than you might think, now that we have calculated the values for Rx,Ry and Rz. Let's go back to our last accelerometer model and do some additional notations:



The angles that we are interested in are the angles between X,Y,Z axes and the force vector R. We'll define these angles as Axr, Ayr, Azr. You can notice from the right-angle triangle formed by R and Rx that:

cos(Axr) = Rx / R , and similarly :
cos(Ayr) = Ry / R
cos(Azr) = Rz / R

We can deduct from **Eq.1** that R = SQRT( Rx^2 + Ry^2 + Rz^2).

We can find now our angles by using arccos() function (the inverse cos() function ):

We've gone a long way to explain the accelerometer model, just to come up to these formulas. Depending on your applications you might want to use any intermediate formulas that we have derived. We'll also introduce the gyroscope model soon, and we'll see how accelerometer and gyroscope data can be combined to provide even more accurate inclination estimations.

But before we do that let's do some more useful notations:

cosX = cos(Axr) = Rx / R
cosY = cos(Ayr) = Ry / R
cosZ = cos(Azr) = Rz / R

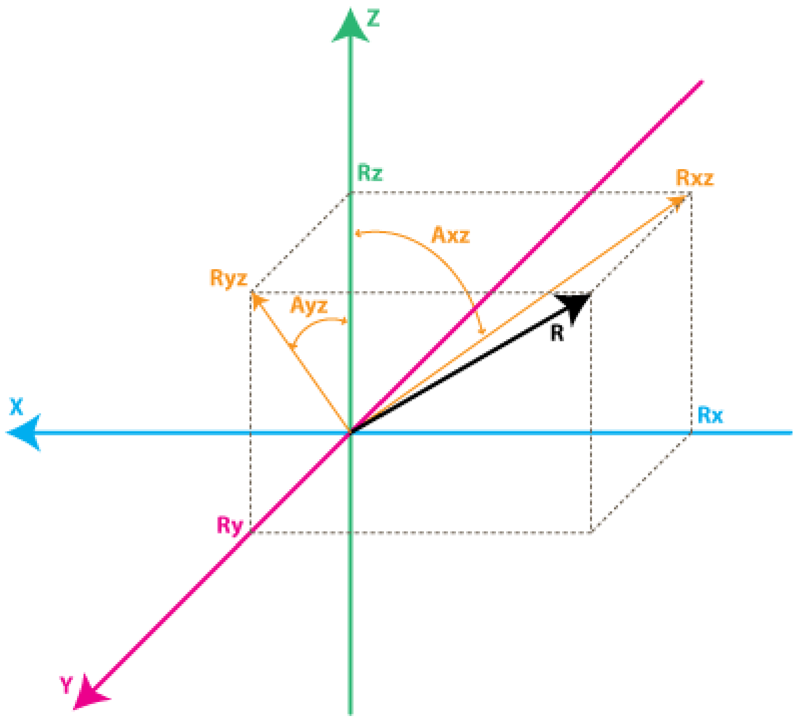
This triplet is often called Direction Cosine , and it basically represents the unit vector (vector with length 1) that has same direction as our R vector. You can easily verify that:

SQRT(cosX^2 + cosY^2 + cosZ^2) = 1

This is a nice property since it absolve us from monitoring the modulus(length) of R vector. Often times if we're just interested in direction of our inertial vector, it makes sense to normalize it's modulus in order to simplify other calculations.

## Part 2. Gyroscope

We're not going to introduce any equivalent box model for the gyroscope like we did for accelerometer, instead we're going to jump straight to the second accelerometer model and we'll show what does the gyroscope measure according to this model.



Each gyroscope channel measures the rotation around one of the axes. For instance a 2-axes gyroscope will measure the rotation around (or some may say "about") the X and Y axes. To express this rotation in numbers let's do some notations. First let's define:

Rxz^2 = Rx^2 + Rz^2 , and similarly:
Ryz^2 = Ry^2 + Rz^2

also note that:

R^2 = Rxz^2 + Ry^2 , this can be derived from **Eq.1** and above equations, or it can be derived from right-angle triangle formed by R and Ryz
R^2 = Ryz^2 + Rx^2

We're not going to use these formulas in this article but it is useful to note the relation between all the values in our model.

Instead we're going to define the angle between the Z axis and Rxz, Ryz vectors as follows:

Axz – is the angle between the Rxz (projection of R on XZ plane) and Z axis
Ayz – is the angle between the Ryz (projection of R on YZ plane) and Z axis

Now we're getting closer to what the gyroscope measures. Gyroscope measures the rate of changes of the angles defined above. In other words it will output a value that is linearly related to the rate of change of these angles. To explain this let's assume that we have measured the rotation angle around axis Y (that would be Axz angle) at time t0, and we define it as Axz0, next we measured this angle at a later time t1 and it was Axz1. The rate of change will be calculated as follows:

RateAxz = (Axz1 – Axz0) / (t1 – t0).

If we express Axz in degrees, and time in seconds , then this value will be expressed in deg/s . This is what a gyroscope measures.

In practice a gyroscope(unless it is a special digital gyroscope) will rarely give you a value expressed in deg/s. Same as for accelerometer you'll get an ADC value that you'll need to convert to deg/s using a formula similar to **Eq. 2** that we have defined for accelerometer. Let's introduce the ADC to deg/s conversion formula for gyroscope (we assume we're using a 10bit ADC module , for 8bit ADC replace 1023 with 255, for 12bit ADC replace 1023 with 4095).

RateAxz = (AdcGyroXZ * Vref / 1023 – VzeroRate) / Sensitivity **Eq.3**
RateAyz = (AdcGyroYZ * Vref / 1023 – VzeroRate) / Sensitivity

AdcGyroXZ, AdcGyroYZ – are obtained from our adc module and they represent the channels that measure the rotation of projection of R vector in XZ respectively in YZ planes, which is the equivalent to saying rotation was done around Y and X axes respectively.

Vref – is the ADC reference voltage we'll use 3.3V in the example below

VzeroRate – is the zero-rate voltage, in other words the voltage that the gyroscope outputs when it is not subject to any rotation, for the Acc_Gyro board it is for example 1.23V (you can find this values in the specs – but don't trust the specs most gyros will suffer slight offset after being soldered so measure VzeroRate for each axis output using a voltmeter, usually this value will not change over time once the gyro was soldered, if it variates – write a calibration routine to measure it before device start-up, user must be instructed to keep device in still position upon start-up for gyros to calibrate).

Sensitivity – is the sensitivity of your gyroscope it is expressed in mV / (deg / s) often written as mV/deg/s , it bas
tells you how many mV will the gyroscope output increase , if you increase the rotation speed by one deg/s. The
sensitivity of Acc_Gyro board is for example 2mV/deg/s or 0.002V/deg/s

隐私权 - 使用条款

Using the above formula, and using the specs parameters of Acc_Gyro board we'll get:

RateAxz = (571 * 3.3V / 1023 – 1.23V) / ( 0.002V/deg/s) =~ 306 deg/s
RateAyz = (323 * 3.3V / 1023 – 1.23V) / ( 0.002V/deg/s) =~ -94 deg/s

In other words the device rotates around the Y axis (or we can say it rotates in XZ plane) with a speed of 306 deg/s and around the X axis (or we can say it rotates in YZ plane) with a speed of -94 deg/s. Please note that the negative sign means that the device rotates in the opposite direction from the conventional positive direction. By convention one direction of rotation is positive. A good gyroscope specification sheet will show you which direction is positive, otherwise you'll have to find it by experimenting with the device and noting which direction of rotation results in increasing voltage on the output pin. This is best done using an oscilloscope since as soon as you stop the rotation the voltage will drop back to the zero-rate level. If you're using a multimeter you'd have to maintain a constant rotation rate for at least few seconds and note the voltage during this rotation, then compare it with the zero-rate voltage. If it is greater than the zero-rate voltage it means that direction of rotation is positive.

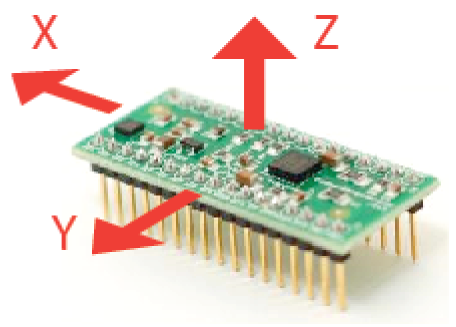
## Part 3. Putting it all together. Combining accelerometer and gyroscope data.

If you're reading this article you probably acquired or are planning to acquire a IMU device, or probably you're planning to build one from separate accelerometer and gyroscope devices.

**NOTE:** FOR PRACTICAL IMPLEMENTATION AND TESTING  OF THIS ALGORITHM PLEASE READ THIS ARTICLE:

http://starlino.com/imu_kalman_arduino.html

The first step in using a combination IMU device that combines an accelerometer and a gyroscope is to align their coordinate systems. The easiest way to do it is to choose the coordinate system of accelerometer as your reference coordinate system. Most accelerometer data sheets will display the direction of X,Y,Z axes relative to the image of the physical chip or device. For example here are the directions of X,Y,Z axes as shown in specifications for the Acc_Gyro board:



Next steps are:

– identify the gyroscope outputs that correspond to RateAxz , RateAyz values discussed above.
– determine if these outputs need to be inverted due to physical position of gyroscope relative to the acceleron

隐私权 - 使用条款

– start from placing the device in horizontal position. Both X and Y outputs of accelerometer would output the zero-g voltage (for example for Acc_Gyro board this is 1.65V)
– next start rotating the device around the Y axis, another way to say it is that you rotate the device in XZ plane, so that X and Z accelerometer outputs change and Y output remains constant.
– while rotating the device at a constant speed note which gyroscope output changes, the other gyroscope outputs should remain constant
– the gyroscope output that changed during the rotation around Y axis (rotation in XZ plane) will provide the input value for AdcGyroXZ, from which we calculate RateAxz
– the final step is to ensure the rotation direction corresponds to our model, in some cases you may have to invert the RateAxz value due to physical position of gyroscope relative to the accelerometer
– perform again the above test, rotating the device around the Y axis, this time monitor the X output of accelerometer (AdcRx in our model). If AdcRx grows (the first 90 degrees of rotation from horizontal position), then AdcGyroXZ should decrease. This is due to the fact that we are monitoring the gravitation vector and when device rotates in one direction the vector will rotate in oposite direction (relative to the device coordonate system, which we are using). So, otherwise you need to invert RateAxz , you can achieve this by introducing a sign factor in **Eq.3**, as follows:

RateAxz = InvertAxz * (AdcGyroXZ * Vref / 1023 – VzeroRate) / Sensitivity , where InvertAxz is 1 or -1

same test can be done for RateAyz , by rotating the device around the X axis, and you can identify which gyroscope output corresponds to RateAyz, and if it needs to be inverted. Once you have the value for InvertAyz, you should use the following formula to calculate RateAyz:

RateAyz = InvertAyz * (AdcGyroYZ * Vref / 1023 – VzeroRate) / Sensitivity

If you would do these tests on Acc_Gyro board you would get following results:

– the output pin for RateAxz is GX4 and InvertAxz = 1
– the output pin for RateAyz is GY4 and InvertAyz = 1

From this point on we'll consider that you have setup your IMU in such a way that you can calculate correct values for Axr, Ayr, Azr (as defined Part 1. Accelerometer) and RateAxz, RateAyz (as defined in Part 2. Gyroscope). Next we'll analyze the relations between these values that turn out useful in obtaining more accurate estimation of the inclination of the device relative to the ground plane.

You might be asking yourself by this point, if accelerometer model already gave us inclination angles of Axr,Ayr,Azr why would we want to bother with the gyroscope data ? The answer is simple: accelerometer data can't always be trusted 100%. There are several reason, remember that accelerometer measures inertial force, such a force can be caused by gravitation (and ideally only by gravitation), but it might also be caused by acceleration (movement) of the device. As a result even if accelerometer is in a relatively stable state, it is still very sensitive to vibration and mechanical noise in general. This is the main reason why most IMU systems use a gyroscope to smooth out any accelerometer errors. But how is this done ? And is the gyroscope free from noise ?

The gyroscope is not free from noise however because it measures rotation it is less sensitive to linear mechanical movements, the type of noise that accelerometer suffers from, however gyroscopes have other types of problems like for example drift (not coming back to zero-rate value when rotation stops). Nevertheless by averaging data that comes from accelerometer and gyroscope we can obtain a relatively better estimate of current device inclination than we would obtain by using the accelerometer data alone.

In the next steps I will introduce an algorithm that was inspired by some ideas used in Kalman filter, however it more simple and easier to implement on embedded devices. Before that let's see first what we want our algorit[illegible] calculate. Well , it is the direction of gravitation force vector R = [Rx,Ry,Rz] from which we can derive other values like confusion let's re-define the accelerometer measurements as follows:

Racc – is the inertial force vector as measured by accelerometer, that consists of following components (projections on X,Y,Z axes):

RxAcc = (AdcRx * Vref / 1023 – VzeroG) / Sensitivity
RyAcc = (AdcRy * Vref / 1023 – VzeroG) / Sensitivity
RzAcc = (AdcRz * Vref / 1023 – VzeroG) / Sensitivity

So far we have a set of measured values that we can obtain purely from accelerometer ADC values. We'll call this set of data a "vector" and we'll use the following notation.

Racc = [RxAcc,RyAcc,RzAcc]

Because these components of Racc can be obtained from accelerometer data , we can consider it an input to our algorithm.

Please note that because Racc measures the gravitation force you'll be correct if you assume that the length of this vector defined as follows is equal or close to 1g.

|Racc| = SQRT(RxAcc^2 +RyAcc^2 + RzAcc^2),

However to be sure it makes sense to update this vector as follows:

Racc(normalized) = [RxAcc/|Racc| , RyAcc/|Racc| , RzAcc/|Racc|].

This will ensure the length of your normalized Racc vector is always 1.

Next we'll introduce a new vector and we'll call it

Rest = [RxEst,RyEst,RzEst]

This will be the output of our algorithm , these are corrected values based on gyroscope data and based on past estimated data.

Here is what our algorithm will do:
– accelerometer tells us: "You are now at position Racc"
– we say "Thank you, but let me check",
– then correct this information with gyroscope data as well as with past Rest data and we output a new estimated vector Rest.
– we consider Rest to be our "best bet" as to the current position of the device.

Let's see how we can make it work.

We'll start our sequence by trusting our accelerometer and assigning:

Rest(0) = Racc(0)

By the way remember Rest and Racc are vectors , so the above equation is just a simple way to write 3 sets of equations, and avoid repetition:

RxEst(0) = RxAcc(0)
RyEst(0) = RyAcc(0)
RzEst(0) = RzAcc(0)

Suppose we're at step n. We have two known sets of values that we'd like to use:

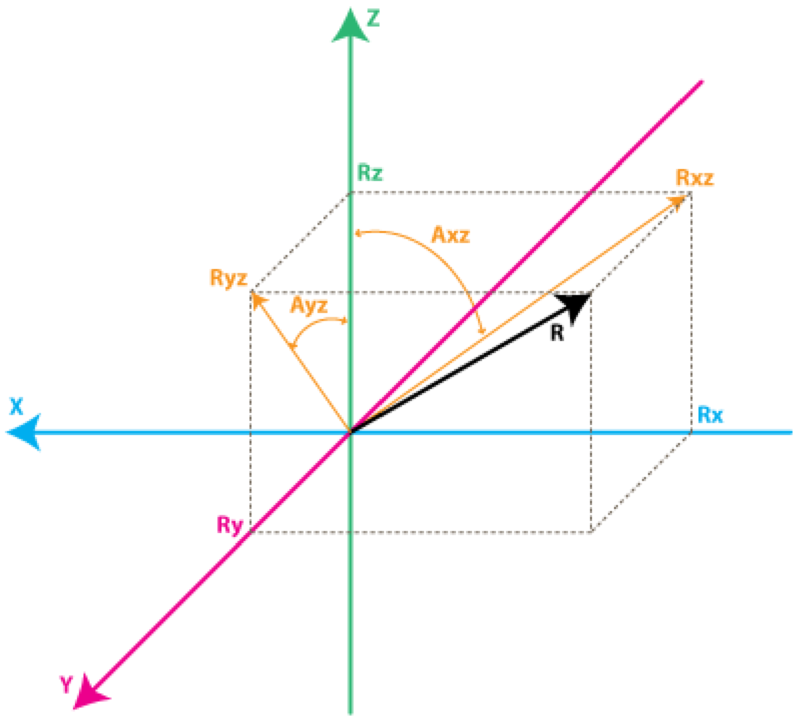
Rest(n-1) – our previous estimate, with Rest(0) = Racc(0)
Racc(n) – our current accelerometer measurement

Before we can calculate Rest(n) , let's introduce a new measured value, that we can obtain from our gyroscope and a previous estimate.

We'll call it Rgyro , and it is also a vector consisting of 3 components:

Rgyro = [RxGyro,RyGyro,RzGyro]

We'll calculate this vector one component at a time. We'll start with RxGyro.



Let's start by observing the following relation in our gyroscope model, from the right-angle triangle formed by Rz and Rxz we can derive that:

tan(Axz) = Rx/Rz => Axz = atan2(Rx,Rz)

Atan2 might be a function you never used before, it is similar to atan, except it returns values in range of (-PI,PI) as opposed to (-PI/2,PI/2) as returned by atan, and it takes 2 arguments instead of one. It allows us to convert the two values of Rx,Rz to angles in the full range of 360 degrees (-PI to PI). You can read more about atan2 here.

So knowing RxEst(n-1) , and RzEst(n-1) we can find:

Axz(n-1) = atan2( RxEst(n-1) , RzEst(n-1) ).

Remember that gyroscope measures the rate of change of the Axz angle. So we can estimate the new angle Axz(n) as follows:

Axz(n) = Axz(n-1) + RateAxz(n) * T

Axz(n) = Axz(n-1) + RateAxzAvg * T

The same way we can find:

Ayz(n) = Ayz(n-1) + RateAyz(n) * T

Ok so now we have Axz(n) and Ayz(n). Where do we go from here to deduct RxGyro/RyGyro ? From **Eq. 1** we can write the length of vector Rgyro as follows:

|Rgyro| = SQRT(RxGyro^2 + RyGyro^2 + RzGyro^2)

Also because we normalized our Racc vector, we may assume that it's length is 1 and it hasn't changed after the rotation, so it is relatively safe to write:

|Rgyro| = 1

Let's adopt a temporary shorter notation for the calculations below:

x =RxGyro , y=RyGyro, z=RzGyro

Using the relations above we can write:

x = x / 1 = x / SQRT(x^2+y^2+z^2)

Let's divide numerator and denominator of fraction by SQRT(x^2 + z^2)

x = ( x / SQRT(x^2 + z^2) ) / SQRT( (x^2 + y^2 + z^2) / (x^2 + z^2) )

Note that x / SQRT(x^2 + z^2) = sin(Axz), so:

x = sin(Axz) / SQRT (1 + y^2 / (x^2 + z^2) )

Now multiply numerator and denominator of fraction inside SQRT by z^2

x = sin(Axz) / SQRT (1 + y^2 * z ^2 / (z^2 * (x^2 + z^2)) )

Note that z / SQRT(x^2 + z^2) = cos(Axz) and y / z = tan(Ayz), so finally:

x = sin(Axz) / SQRT (1 + cos(Axz)^2 * tan(Ayz)^2 )

Going back to our notation we get:

RxGyro = sin(Axz(n)) / SQRT (1 + cos(Axz(n))^2 * tan(Ayz(n))^2 )

same way we find that

RyGyro = sin(Ayz(n)) / SQRT (1 + cos(Ayz(n))^2 * tan(Axz(n))^2 )

***Side Note: it is possible to further simplify this formula. By dividing both parts of the fraction by sin(Axz(n)) you get:***
*RxGyro = 1 / SQRT (1/* ***sin(Axz(n))^2*** *+ cos(Axz(n))^2 /* ***sin(Axz(n))^2*** ** tan(Ayz(n))^2 )*
*RxGyro = 1 / SQRT (1/ sin(Axz(n))^2 + cot(Axz(n))^2 ** ***sin(Ayz(n))^2 / cos(Ayz(n))^2*** *)*
*now add and substract cos(Axz(n))^2/sin(Axz(n))^2 = cot(Axz(n))^2*
*RxGyro = 1 / SQRT (1/ sin(Axz(n))^2* ***– cos(Axz(n))^2/sin(Axz(n))^2*** *+ cot(Axz(n))^2 * sin(Ayz(n))^2 / cos(Ayz(n))^2* ***+ cot(Axz(n))^2*** *)*
*and by grouping terms 1&2 and then 3&4 we get*
***RxGyro = 1 / SQRT (1 + cot(Axz(n))^2 * sec(Ayz(n))^2 ), where cot(x) = 1 / tan(x) and sec(x) = 1 / cos(x)***

隐私权 - 使用条款

Now, finally we can find:

RzGyro  =  Sign(RzGyro)*SQRT(1 – RxGyro^2 – RyGyro^2).

Where Sign(RzGyro) = 1 when RzGyro>=0 , and Sign(RzGyro) = -1 when RzGyro<0.

One simple way to estimate this is to take:

Sign(RzGyro) = Sign(RzEst(n-1))

*In practice be careful when RzEst(n-1) is close to 0. You may skip the gyro phase altogether in this case and assign:  Rgyro = Rest(n-1). Rz is used as a reference for calculating Axz and Ayz angles and when it's close to 0, values may overflow and trigger bad results. You'll be in domain of large floating point numbers where tan() / atan() function implementations may lack precision.*

So let's recap what we have so far, we are at step **n** of our algorithm and we have calculated the following values:

Racc – current readings from our accelerometer
Rgyro – obtained from Rest(n-1) and current gyroscope readings

Which values do we use to calculate the updated estimate Rest(n) ? You probably guessed that we'll use both. We'll use a weighted average, so that:

Rest(n) = (Racc * w1 + Rgyro * w2 ) / (w1 + w2)

We can simplify this formula by dividing both numerator and denominator of the fraction by w1.

Rest(n) = (Racc * w1/w1 + Rgyro * w2/w1 ) / (w1/w1 + w2/w1)

and after substituting w2/w1 = wGyro we get:

Rest(n) = (Racc + Rgyro * wGyro ) / (1 + wGyro)

In the above formula wGyro tells us how much we trust our gyro compared to our accelerometer. This value can be chosen experimentally usually values between 5..20 will trigger good results.

The main difference of this algorithm from Kalman filter is that this weight is relatively fixed , whereas in Kalman filter the weights are permanently updated based on the measured noise of the accelerometer readings. Kalman filter is focused at giving you "the best" theoretical results, whereas this algorithm can give you results "good enough" for your practical application. You can implement an algorithm that adjusts wGyro depending on some noise factors that you measure, but fixed values will work well for most applications.

We are one step away from getting our updated estimated values:

RxEst(n) = (RxAcc + RxGyro * wGyro ) / (1 + wGyro)
RyEst(n) = (RyAcc + RyGyro * wGyro ) / (1 + wGyro)
RzEst(n) = (RzAcc + RzGyro * wGyro ) / (1 + wGyro)

Now let's  normalize this vector again:

R = SQRT(RxEst(n) ^2 + RyEst(n)^2 +  RzEst(n)^2 )

RxEst(n) = RxEst(n)/R
RyEst(n) = RyEst(n)/R
RzEst(n) = RzEst(n)/R

http://starlino.com/imu_kalman_arduino.html

**Other Resources on Accelerometer and Gyroscope IMU Fusion:**

http://www.mikroquad.com/pub/Research/ComplementaryFilter/filter.pdf

http://stackoverflow.com/questions/1586658/combine-gyroscope-and-accelerometer-data

http://www.dimensionengineering.com/accelerometers.htm

//starlino//

POSTED IN FEATURED, IMU THEORY AND EXPERIMENTS
TAGGED ACCELEROMETER, ACC_GYRO, GYROSCOPE, IMU, MOTION

## 327 THOUGHTS ON "A GUIDE TO USING IMU (ACCELEROMETER AND GYROSCOPE DEVICES) IN EMBEDDED APPLICATIONS."

**GLENJOY** APRIL 9, 2010 / REPLY

Your tutorial is helpful, but some of the equations are quite confusing, as I am more of expecting so what is now the formula to get the mixed output of the accelerometer and the gyroscope?

I will be using a single axis gyro and accelerometer. So if ever I'll take one of your formulas, say RxEst(n) = (RxAcc + RxGyro * wGyro ) / (1 + wGyro);

and from my understanding from your tutorial,

RxAcc – ADC reading of the acclerometer
RxGyro – is the previous reading of the Gyro, so it means, RXGyro(n-1), as n is the current.

So does this mean that RxGyro is a direct ADC reading from the gyroscope or is there a mathematical computation still needed to get the value of RxGyro?

Thank you.

**STARLINO** APRIL 9, 2010 / REPLY