About data types in SensorStudio

InvenSense Inc.
1745 Technology Drive, San Jose, CA 95110 U.S.A
+1(408) 988–7339
www.invensense.com

Document Number:
Revision: 1.0
Date: 08/26/2016

TABLE OF CONTENTS

TABLE OF FIGURES

1. DATA TYPING

SensorStudio supports different data types in the data flow, among arrays of doubles and strings for instance. Pins are strongly typed and links can only be created between pins that have the same type.
For example, in Figure 1, the output pin of the ArrayGenerator block and the input pin of the Sliding panel are both arrays of doubles and therefore can be connected together (pin is highlighted in green while creating a link). The input pin of the Label panel is a string, and is not compatible (pin is not highlighted while creating a link).

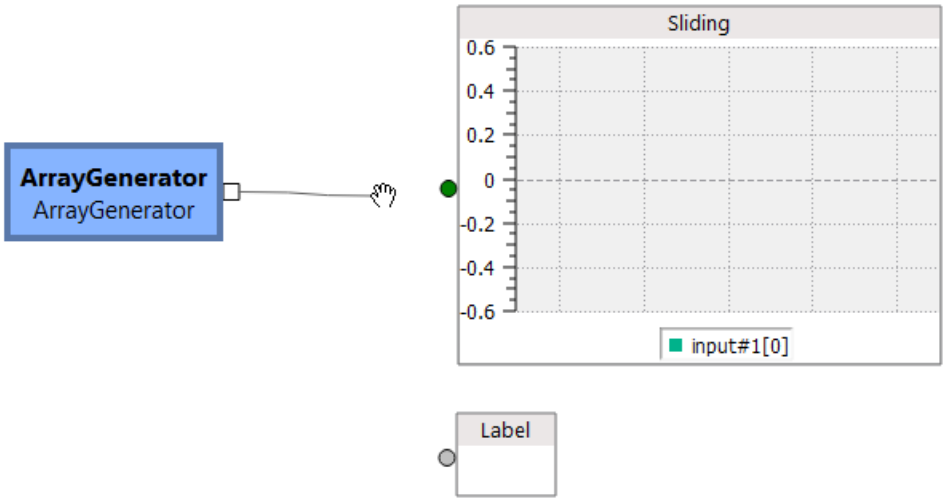


Figure 1. Creating links

The data type of a pin (input pin or output pin) is available in the tooltip of the pin (displayed when hovering the pin with the mouse) (see Figure 2) and in its help (displayed when the block/panel is selected or when the pin itself is selected) (see Figure 3).

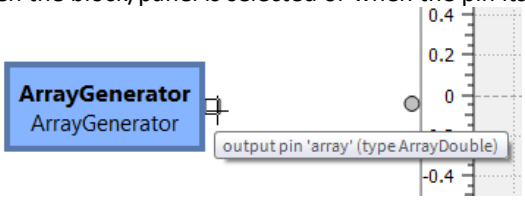


Figure 2. Pin tooltip

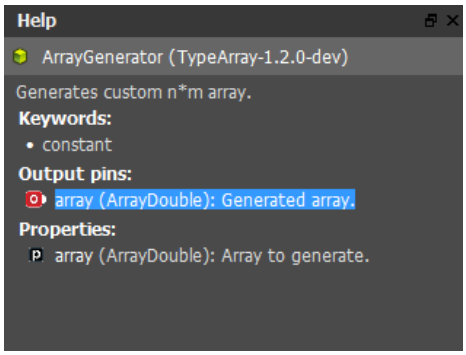


Figure 3. Block help

1.1. DATA: ARRAYDOUBLE

A unique type for handling numbers is used in SensorStudio: ArrayDouble, which are 2-dimensional arrays of dynamic size. For instance, the data produced by the Accelerometer block is an array of size 1x3. The ArrayDouble type can be used for various mathematical objects:

- Scalars (1x1 array),
- Vectors, either horizontal (1xn array) or vertical (nx1),
- Rotation matrices (3x3 array),
- Quaternions (1x4 array),
- Euler angles (1x3 array).

For any ArrayDouble output pin, info on the size of the data is provided in its help; for ArrayDouble input pins, the data consumed can have restrictions (on size for instance), which are displayed in its help (see Figure 4).

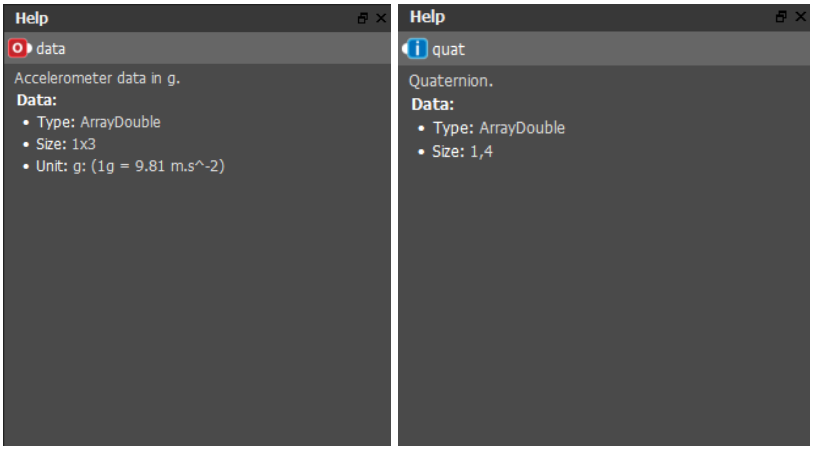


Figure 4. The help of a pin displays additional info – example of data output pin from Accelerometer block and quat input pin from QuaternionToRotMatrix block

1.2. DEVICE-SENSOR LINKS: BLOCKDEVICEHANDLE

SensorStudio uses a specific type for connecting sensors to a device block, BlockDeviceHandle (Figure 6). This type does not convey data and it cannot be visualized.

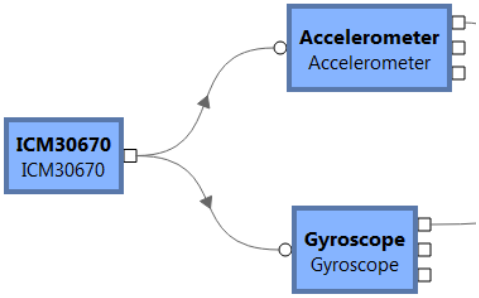


Figure 5. Connecting sensors to a device

1.3. SENSOR-CUSTOMSENSORDESIGNER LINKS: VSENSOREVENT

SensorStudio uses a specific type for connecting sensors to a CustomSensorDesigner, VSensorEvent type (Figure 6). This type cannot be visualized.

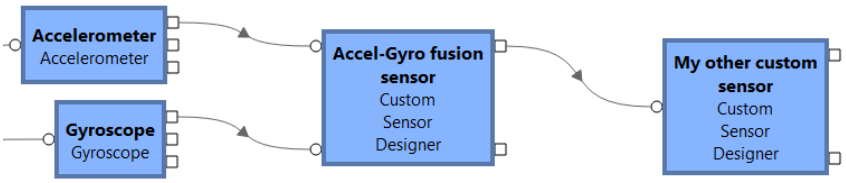


Figure 6. Connecting sensors to CustomSensorDesigner blocks

2. VISUALIZING DATA

The Console panel can display any ArrayDouble as text (Figure 7).

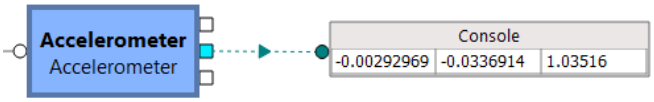


Figure 7. The Console panel can be used to vizualize arrays

2.1. SCALARS (1X1 ARRAYDOUBLE)

Sliding and Oscilloscope panels can plot scalars as curves, X-axis being the time (Figure 8).

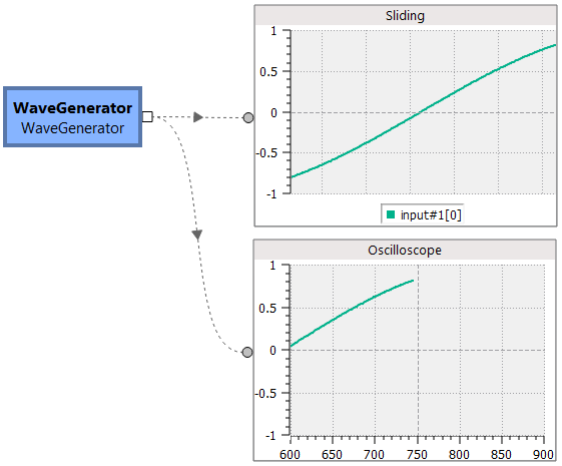


Figure 8. Plotting scalars

Color, Image, Sound and Text panels can be used to visualize the data based on ranges.

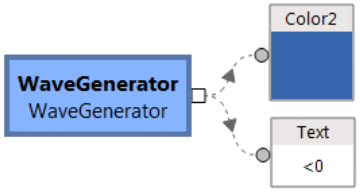


Figure 9. Using range-based panels to display scalars

2.2. VECTORS (1XN ARRAYDOUBLE)

Plot1D panel can be used to plot a vector as a curve. The X-axis represent the index of the data in the vector (y[i] = f(i)).
Sliding and Oscilloscope panels can also plot vectors as curves, X-axis being the time; they will create one curve per data in the vector.

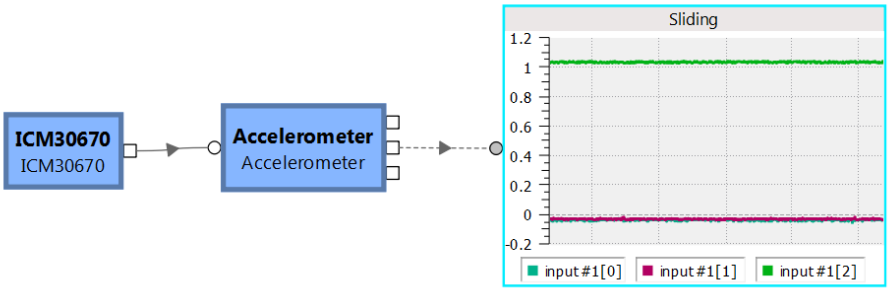


Figure 10. Using Sliding panel to display vectors

2.3. ROTATION MATRICES (3X3 ARRAYDOUBLE)

Cube panel can display a rotation matrix (Figure 11). Blocks to convert quaternions or other representations of an orientation to a rotation matrix are provided in the OrientationUtils plugin.
The next section explains the concept behind each representation of an orientation.

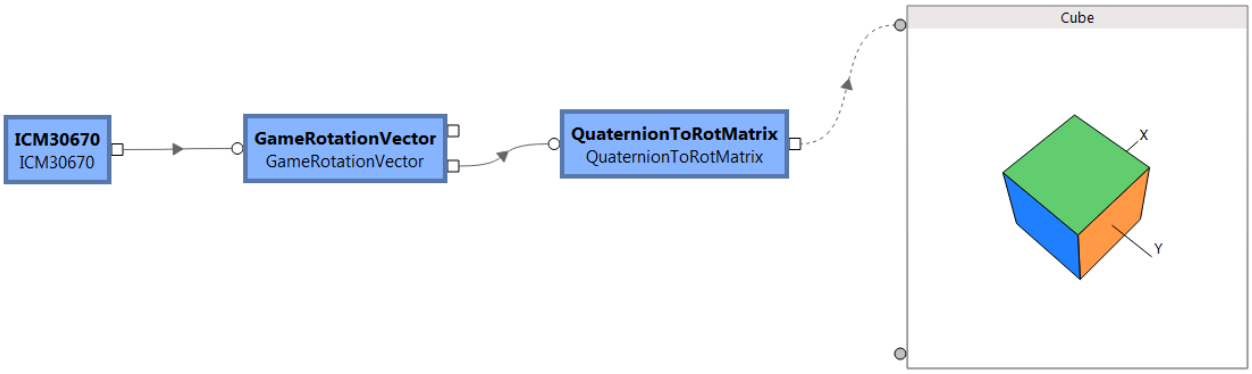


Figure 11. Using Cube panel to display rotation matrices

3. INTRODUCTION TO ROTATIONS

Available representations of an orientation are:

- Rotation matrix
- Quaternion
- Angle Axis
- Euler Angles

Any 3D rotation can be characterized by an angle ϕ and a unit vector $N(n_x,n_y,n_z)$ which is invariant by this rotation (using -ϕ and $N'(-n_x,-n_y,-n_z)$ gives the same rotation). There are several ways to represent this rotation. Rotation matrix, quaternion and axis angle represent the real 3D rotation whereas the Euler angles gives a representation using angles in projection planes (2D angles).

Euler Angles representation has the advantage of being easily understood since it refers to 2D angles. We are used to them (planar geometry, 2D radiography, books of anatomy, etc). Unfortunately, things become tricky when we want to represent a 3D orientation because they are not relevant when the orientation is perpendicular to the observation plane (projection will be close to origin and small displacement will induce big angles, see Figure 12 below).

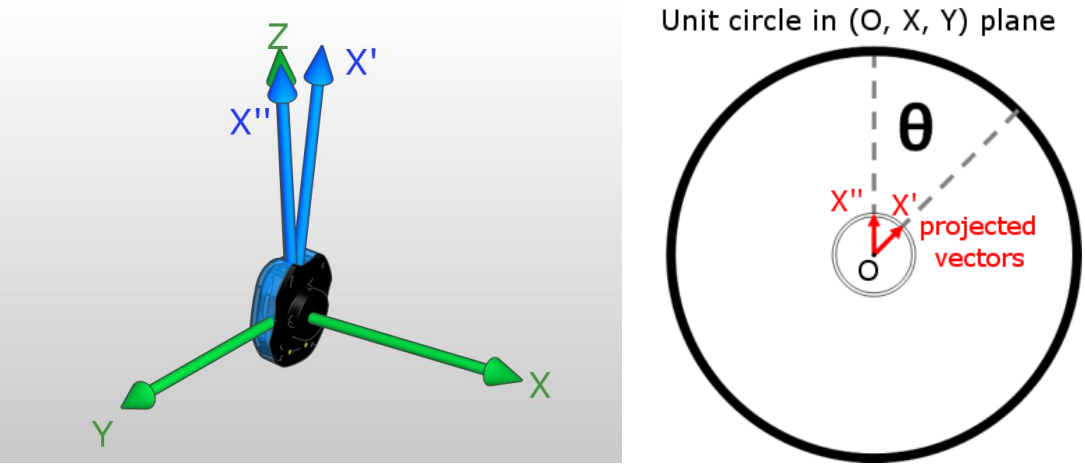


Figure 12. Euler angles instability problem (in that case, pitch is applied first, then yaw, then roll).

If the projected vector (X' or X'') is almost perpendicular to the projection plane, a small 3D rotation (right) may yield big changes in the observed angle θ as you can see in the (O, X, Y) plane (left).

If you want to use Euler Angles and avoid these problems, make sure your measures stay in a correct range (see Euler Angles section). For general 3D movements, use other representations.

Note:
All blocks that convert any representation of rotations into Euler Angles have an additional output pin, gimbalLock, to indicate if the measures are in the correct range or not.

3.1. ROTATION MATRIX

The rotation matrix is the matrix of the transformation applied to the reference frame to get the current frame. As a direct consequence:

- its first column is the coordinates of the new X axis expressed in the reference frame.
- its second column is the coordinates of the new Y axis expressed in the reference frame.
- its third column is the coordinates of the new Z axis expressed in the reference frame.

You can use the rotation matrix, M, as the change of basis matrix from new frame to reference frame.
If you know the coordinate of a given point are (x, y, z) in the rotated frame, its coordinates (x',y',z') in the reference frame will be:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = [M] \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Use its inverse, M^{-1} which is also its transpose matrix, to do the reverse transformation.

Note:
There is a relation between the matrix's coefficients and the rotation's parameters, ϕ and its unit vector $N(n_x,n_y,n_z)$:

$$M = \cos\varphi \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + (1-\cos\varphi) \begin{bmatrix} n_x^2 & n_x n_y & n_x n_z \\ n_x n_y & n_y^2 & n_y n_z \\ n_x n_z & n_y n_z & n_z^2 \end{bmatrix} + \sin\varphi \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$

3.2. QUATERNION

Quaternions are often used to represent rotations because of their compactness (4 coefficients compared to 9 for rotation matrix) and their efficiency for most operations on rotation (composition, etc). They are also very useful to interpolate rotations (which cannot be done with other representations).
There is a simple relation between the quaternion $q = (q_0, q_1, q_2, q_3)$ also noted q = (w, x, y, z) and the rotation's parameters, ϕ and its unit vector $N(n_x,n_y,n_z)$:

$w = q_0 = \cos(\phi/2)$
$x = q_1 = n_x.\sin(\phi/2)$
$y = q_2 = n_y.\sin(\phi/2)$
$z = q_3 = n_z.\sin(\phi/2)$

However if you need ϕ and $N(n_x,n_y,n_z)$, you do not have to use quaternions because they are straightly ginven by the Angle Axis representation.

3.3. ANGLE AXIS

The angle axis represents the rotation directly by its parameters, ϕ and its unit vector $N(n_x,n_y,n_z)$. Here ϕ is always positive, in [0, 360°] and the direction of rotation is chosen so that as you look in the direction of $N(n_x,n_y,n_z)$, the rotation is counterclockwise around the origin.

angle = ϕ
$x = n_x$
$y = n_y$
$z = n_z$

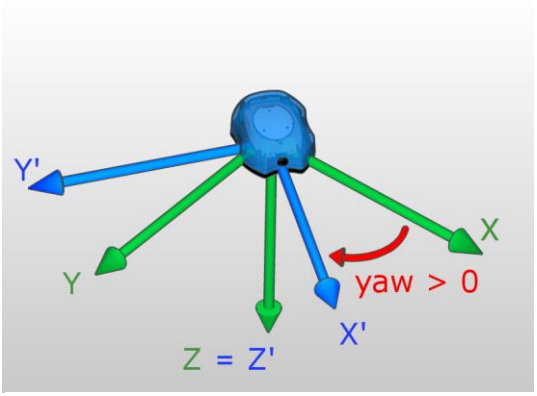
Note:

- if ϕ = 0 then the axis is arbitrary, we have chosen (1, 0, 0),
- if you rotate counterclockwise around Z (yaw > 0), you will get ϕ = yaw and N(0,0,1),
- if you rotate clockwise around Z (yaw < 0), you will get ϕ = -yaw and N(0,0,-1).

3.4. EULER ANGLES

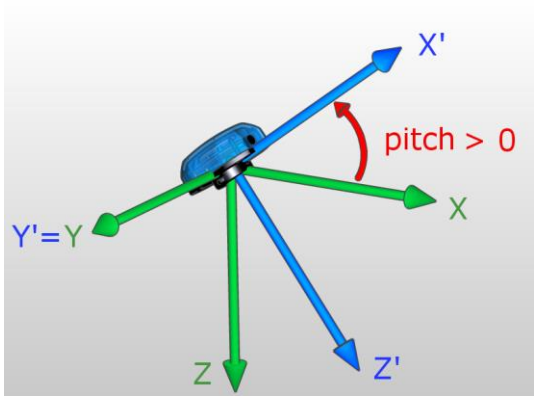
3.4.1. Definition of yaw, pitch and roll

A rotation can be expressed with 3 angles: yaw, pitch and roll which are respectively rotations around X, Y and Z axis. They are illustrated below:



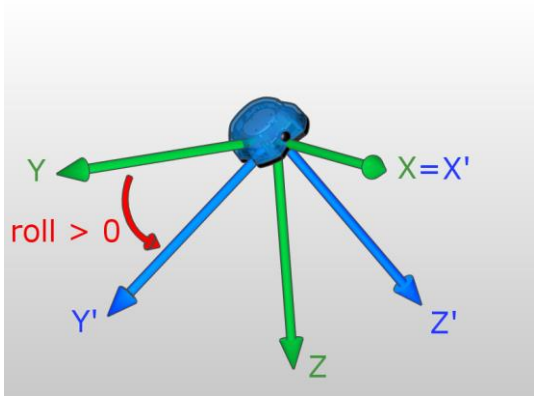
Rotation Matrix associated with yaw:

$$Q_{\mathbf{z}}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$



Rotation Matrix associated with pitch:

$$Q_{\mathbf{y}}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},$$



Rotation Matrix associated with roll:

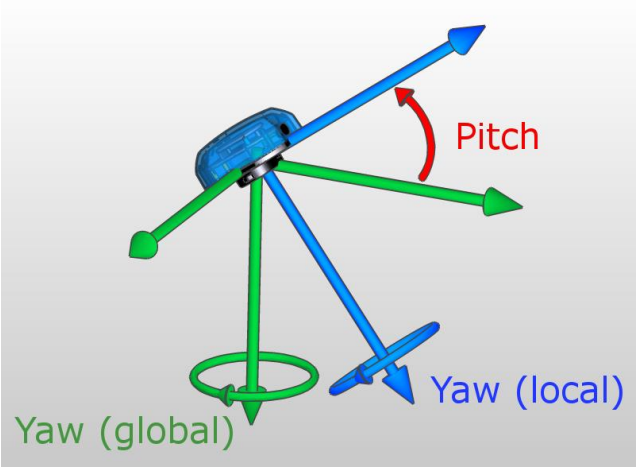
$$Q_{\mathbf{x}}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix},$$

3.4.2. Local or global mode?

There are two ways to deal with successive rotations:

- global rotations, they are done around the reference axis.
- local rotations, they are done around local axis of the device which may have been already affected by other rotations.

As you can see on the figure below, applying local yaw (around Z') or global yaw (around Z) axis is different if the device has already been rotated around Y (pitch).



You might want to choose global or local mode depending on your application.

3.4.3. Different orders

There are 6 different orders in which to apply yaw pitch and roll (see table below). Depending on the movement you want to study, you might want to choose one or the other.

You have to keep in mind that the angles will be relevant if the movement belongs to a plane or is close to it.
This is because instabilities occur if the secondly applied angle is close to + or - 90°.
So the second angle has to be small (absolute value smaller than 70° is recommended). It means that the movement should be close to the plane perpendicular to the axis around which the second rotation is done.

n°	Rotation order	Range for yaw	Range for pitch	Range for roll	Reliable range
0	yaw then pitch then roll	[-180°, 180°]	[-90°, 90°]	[-180°, 180°]	-70° < pitch < 70°
1	pitch then yaw then roll	[-90°, 90°]	[-180°, 180°]	[-180°, 180°]	-70° < yaw < 70°
2	roll then pitch then yaw	[-180°, 180°]	[-90°, 90°]	[-180°, 180°]	-70° < pitch < 70°
3	yaw then roll then pitch	[-180°, 180°]	[-180°, 180°]	[-90°, 90°]	-70° < roll < 70°
4	pitch then roll then yaw	[-180°, 180°]	[-180°, 180°]	[-90°, 90°]	-70° < roll < 70°
5	roll then yaw then pitch.	[-90°, 90°]	[-180°, 180°]	[-180°, 180°]	-70° < yaw < 70°

4. REVISION HISTORY

REVISION DATE	REVISION	DESCRIPTION
WIP	1.0	Initial Release

InvenSense