JavaScript is disabled on your browser.

[Skip navigation links](#1fob9te)

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/com/kauailabs/navx/ftc/package-summary.html)
* Class
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-all.html)
* [Help](http://docs.google.com/help-doc.html)
* Prev Class
* [Next Class](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.BoardAxis.html)
* [Frames](http://docs.google.com/index.html?com/kauailabs/navx/ftc/AHRS.html)
* [No Frames](http://docs.google.com/AHRS.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* [Nested](#3znysh7) |
* Field |
* [Constr](#2et92p0) |
* [Method](#tyjcwt)
* Detail:
* Field |
* [Constr](#1t3h5sf) |
* [Method](#2s8eyo1)

com.kauailabs.navx.ftc

## Class AHRS

* java.lang.Object
  + com.kauailabs.navx.ftc.AHRS
* public class AHRS  
  extends java.lang.Object  
  The AHRS class provides an interface to AHRS capabilities of the KauaiLabs navX Robotics Navigation Sensor via I2C on the Android- based FTC robotics control system, where communications occur via the "Core Device Interface Module" produced by Modern Robotics, inc. The AHRS class enables access to basic connectivity and state information, as well as key 6-axis and 9-axis orientation information (yaw, pitch, roll, compass heading, fused (9-axis) heading and magnetic disturbance detection. Additionally, the ARHS class also provides access to extended information including linear acceleration, motion detection, rotation detection and sensor temperature. If used with navX-Aero-enabled devices, the AHRS class also provides access to altitude, barometric pressure and pressure sensor temperature data

### Nested Class SummaryNested Classes

|  |  |
| --- | --- |
| * + Modifier and Type | * + Class and Description |
| * + static class | * + [AHRS.BoardAxis](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.BoardAxis.html) Identifies one of the three sensing axes on the navX sensor board. |
| * + static class | * + [AHRS.BoardYawAxis](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.BoardYawAxis.html) Indicates which sensor board axis is used as the "yaw" (gravity) axis. |
| * + static class | * + [AHRS.DeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DeviceDataType.html) The DeviceDataType specifies the type of data to be retrieved from the sensor. |
| * + class | * + [AHRS.DimI2cDeviceReader](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DimI2cDeviceReader.html) |
| * + class | * + [AHRS.DimI2cDeviceWriter](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DimI2cDeviceWriter.html) |
| * + class | * + [AHRS.DimStateTracker](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DimStateTracker.html) |

### Constructor SummaryConstructors

|  |  |
| --- | --- |
| * + Modifier | * + Constructor and Description |
| * + protected | * + [AHRS](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#AHRS-DeviceInterfaceModule-int-com.kauailabs.navx.ftc.AHRS.DeviceDataType-int-)(DeviceInterfaceModule dim, int dim\_i2c\_port, [AHRS.DeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DeviceDataType.html) data\_type, int update\_rate\_hz) |

### Method SummaryAll Methods Static Methods Instance Methods Concrete Methods

|  |  |
| --- | --- |
| * + Modifier and Type | * + Method and Description |
| * + void | * + [close](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#close--)() |
| * + boolean | * + [deregisterCallback](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#deregisterCallback-com.kauailabs.navx.ftc.IDataArrivalSubscriber-)([IDataArrivalSubscriber](http://docs.google.com/com/kauailabs/navx/ftc/IDataArrivalSubscriber.html) callback) Deregisters a previously registered callback interface. |
| * + byte | * + [getActualUpdateRate](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getActualUpdateRate--)() Returns the navX-Model device's currently configured update rate. |
| * + float | * + [getAltitude](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getAltitude--)() Returns the current altitude, based upon calibrated readings from a barometric pressure sensor, and the currently-configured sea-level barometric pressure [navX Aero only]. |
| * + [AHRS.BoardYawAxis](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.BoardYawAxis.html) | * + [getBoardYawAxis](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getBoardYawAxis--)() Returns information regarding which sensor board axis (X,Y or Z) and direction (up/down) is currently configured to report Yaw (Z) angle values. |
| * + double | * + [getByteCount](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getByteCount--)() Returns the count in bytes of data received from the sensor. |
| * + float | * + [getCompassHeading](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getCompassHeading--)() Returns the current tilt-compensated compass heading value (in degrees, from 0 to 360) reported by the sensor. |
| * + int | * + [getCurrentTransferRate](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getCurrentTransferRate--)() Returns the current number of data samples being transferred from the navX-Model device in the last second. |
| * + [AHRS.DeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DeviceDataType.html) | * + [getDeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getDeviceDataType--)() |
| * + int | * + [getDimI2cPort](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getDimI2cPort--)() |
| * + [AHRS.DimStateTracker](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DimStateTracker.html) | * + [getDimStateTrackerInstance](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getDimStateTrackerInstance--)() |
| * + int | * + [getDuplicateDataCount](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getDuplicateDataCount--)() |
| * + java.lang.String | * + [getFirmwareVersion](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getFirmwareVersion--)() Returns the version number of the firmware currently executing on the sensor. |
| * + float | * + [getFusedHeading](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getFusedHeading--)() Returns the "fused" (9-axis) heading. |
| * + static [AHRS](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html) | * + [getInstance](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getInstance-DeviceInterfaceModule-int-com.kauailabs.navx.ftc.AHRS.DeviceDataType-)(DeviceInterfaceModule dim, int dim\_i2c\_port, [AHRS.DeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DeviceDataType.html) data\_type) Returns the single instance (singleton) of the AHRS class. |
| * + static [AHRS](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html) | * + [getInstance](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getInstance-DeviceInterfaceModule-int-com.kauailabs.navx.ftc.AHRS.DeviceDataType-byte-)(DeviceInterfaceModule dim, int dim\_i2c\_port, [AHRS.DeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DeviceDataType.html) data\_type, byte update\_rate\_hz) Returns the single instance (singleton) of the AHRS class. |
| * + static boolean | * + [getLogging](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getLogging--)() |
| * + float | * + [getPitch](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getPitch--)() Returns the current pitch value (in degrees, from -180 to 180) reported by the sensor. |
| * + float | * + [getPressure](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getPressure--)() Returns the current barometric pressure (in millibar) [navX Aero only]. |
| * + float | * + [getQuaternionW](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getQuaternionW--)() Returns the imaginary portion (W) of the Orientation Quaternion which fully describes the current sensor orientation with respect to the reference angle defined as the angle at which the yaw was last "zeroed". |
| * + float | * + [getQuaternionX](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getQuaternionX--)() Returns the real portion (X axis) of the Orientation Quaternion which fully describes the current sensor orientation with respect to the reference angle defined as the angle at which the yaw was last "zeroed". |
| * + float | * + [getQuaternionY](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getQuaternionY--)() Returns the real portion (X axis) of the Orientation Quaternion which fully describes the current sensor orientation with respect to the reference angle defined as the angle at which the yaw was last "zeroed". |
| * + float | * + [getQuaternionZ](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getQuaternionZ--)() Returns the real portion (X axis) of the Orientation Quaternion which fully describes the current sensor orientation with respect to the reference angle defined as the angle at which the yaw was last "zeroed". |
| * + float | * + [getRawAccelX](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRawAccelX--)() Returns the current raw (unprocessed) X-axis acceleration rate (in G). |
| * + float | * + [getRawAccelY](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRawAccelY--)() Returns the current raw (unprocessed) Y-axis acceleration rate (in G). |
| * + float | * + [getRawAccelZ](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRawAccelZ--)() Returns the current raw (unprocessed) Z-axis acceleration rate (in G). |
| * + float | * + [getRawGyroX](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRawGyroX--)() Returns the current raw (unprocessed) X-axis gyro rotation rate (in degrees/sec). |
| * + float | * + [getRawGyroY](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRawGyroY--)() Returns the current raw (unprocessed) Y-axis gyro rotation rate (in degrees/sec). |
| * + float | * + [getRawGyroZ](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRawGyroZ--)() Returns the current raw (unprocessed) Z-axis gyro rotation rate (in degrees/sec). |
| * + float | * + [getRawMagX](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRawMagX--)() Returns the current raw (unprocessed) X-axis magnetometer reading (in uTesla). |
| * + float | * + [getRawMagY](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRawMagY--)() Returns the current raw (unprocessed) Y-axis magnetometer reading (in uTesla). |
| * + float | * + [getRawMagZ](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRawMagZ--)() Returns the current raw (unprocessed) Z-axis magnetometer reading (in uTesla). |
| * + float | * + [getRoll](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRoll--)() Returns the current roll value (in degrees, from -180 to 180) reported by the sensor. |
| * + float | * + [getTempC](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getTempC--)() Returns the current temperature (in degrees centigrade) reported by the sensor's gyro/accelerometer circuit. |
| * + double | * + [getUpdateCount](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getUpdateCount--)() Returns the count of valid updates which have been received from the sensor. |
| * + float | * + [getWorldLinearAccelX](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getWorldLinearAccelX--)() Returns the current linear acceleration in the X-axis (in G). |
| * + float | * + [getWorldLinearAccelY](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getWorldLinearAccelY--)() Returns the current linear acceleration in the Y-axis (in G). |
| * + float | * + [getWorldLinearAccelZ](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getWorldLinearAccelZ--)() Returns the current linear acceleration in the Z-axis (in G). |
| * + float | * + [getYaw](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getYaw--)() Returns the current yaw value (in degrees, from -180 to 180) reported by the sensor. |
| * + boolean | * + [isAltitudeValid](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#isAltitudeValid--)() Indicates whether the current altitude (and barometric pressure) data is valid. |
| * + boolean | * + [isCalibrating](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#isCalibrating--)() Returns true if the sensor is currently performing automatic gyro/accelerometer calibration. |
| * + boolean | * + [isConnected](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#isConnected--)() Indicates whether the sensor is currently connected to the host computer. |
| * + boolean | * + [isMagneticDisturbance](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#isMagneticDisturbance--)() Indicates whether the current magnetic field strength diverges from the calibrated value for the earth's magnetic field by more than the currently- configured Magnetic Disturbance Ratio. |
| * + boolean | * + [isMagnetometerCalibrated](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#isMagnetometerCalibrated--)() Indicates whether the magnetometer has been calibrated. |
| * + boolean | * + [isMoving](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#isMoving--)() Indicates if the sensor is currently detecting motion, based upon the X and Y-axis world linear acceleration values. |
| * + boolean | * + [isRotating](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#isRotating--)() Indicates if the sensor is currently detecting yaw rotation, based upon whether the change in yaw over the last second exceeds the "Rotation Threshold." |
| * + boolean | * + [registerCallback](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#registerCallback-com.kauailabs.navx.ftc.IDataArrivalSubscriber-)([IDataArrivalSubscriber](http://docs.google.com/com/kauailabs/navx/ftc/IDataArrivalSubscriber.html) callback) Registers a callback interface. |
| * + static void | * + [setLogging](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#setLogging-boolean-)(boolean enabled) |
| * + void | * + [zeroYaw](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#zeroYaw--)() Sets the user-specified yaw offset to the current yaw value reported by the sensor. |

### Methods inherited from class java.lang.Objectclone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

#### AHRS protected AHRS(DeviceInterfaceModule dim, int dim\_i2c\_port, [AHRS.DeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DeviceDataType.html) data\_type, int update\_rate\_hz)

### Method Detail

#### registerCallback public boolean registerCallback([IDataArrivalSubscriber](http://docs.google.com/com/kauailabs/navx/ftc/IDataArrivalSubscriber.html) callback) Registers a callback interface. This interface will be called back when new data is available, based upon a change in the sensor timestamp. Note that this callback will occur within the context of the device IO thread, which is not the same thread context the caller typically executes in.

#### deregisterCallback public boolean deregisterCallback([IDataArrivalSubscriber](http://docs.google.com/com/kauailabs/navx/ftc/IDataArrivalSubscriber.html) callback) Deregisters a previously registered callback interface. Be sure to deregister any callback which have been previously registered, to ensure that the object implementing the callback interface does not continue to be accessed when no longer necessary.

#### close public void close()

#### getInstance public static [AHRS](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html) getInstance(DeviceInterfaceModule dim, int dim\_i2c\_port, [AHRS.DeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DeviceDataType.html) data\_type) Returns the single instance (singleton) of the AHRS class. If the singleton does not alrady exist, it will be created using the parameters passed in. The default update rate will be used. If the singleton already exists, the parameters passed in will be ignored.Returns: The singleton AHRS class instance.

#### getInstance public static [AHRS](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html) getInstance(DeviceInterfaceModule dim, int dim\_i2c\_port, [AHRS.DeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DeviceDataType.html) data\_type, byte update\_rate\_hz) Returns the single instance (singleton) of the AHRS class. If the singleton does not alrady exist, it will be created using the parameters passed in, including a custom update rate. Use this function if an update rate other than the default is desired. NOTE: The range of valid requested update rates is from 4 to 66. However, the actual update does not always match the requested update rate. The following table summarizes the requested to actual update rate lookup table: Actual Requested 66.6 58-66 50 45-57 40 37-44 33.3 31-36 28.57 27-30 25 25-26 22.22 22-23 20 20-21 18.18 18-19 16.67 17 15.38 15-16 14.28 14 Requested values below 14Hz result in an actual rate which is accurate to within 1Hz. The reason for this difference is that the update rate must be an even multiple of a 200Hz clock (5ms). So an actual of 66.6 (15ms/sample) can be calculated as follows: actual\_rate = 200 / (200 / requested\_rate) The getActualUpdateRate() can be used to calculate this value.Returns: The singleton AHRS class instance.

#### setLogging public static void setLogging(boolean enabled)

#### getLogging public static boolean getLogging()

#### getDimI2cPort public int getDimI2cPort()

#### getDeviceDataType public [AHRS.DeviceDataType](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DeviceDataType.html) getDeviceDataType()

#### getPitch public float getPitch() Returns the current pitch value (in degrees, from -180 to 180) reported by the sensor. Pitch is a measure of rotation around the X Axis.Returns: The current pitch value in degrees (-180 to 180).

#### getRoll public float getRoll() Returns the current roll value (in degrees, from -180 to 180) reported by the sensor. Roll is a measure of rotation around the X Axis.Returns: The current roll value in degrees (-180 to 180).

#### getYaw public float getYaw() Returns the current yaw value (in degrees, from -180 to 180) reported by the sensor. Yaw is a measure of rotation around the Z Axis (which is perpendicular to the earth). Note that the returned yaw value will be offset by a user-specified offset value; this user-specified offset value is set by invoking the zeroYaw() method.Returns: The current yaw value in degrees (-180 to 180).

#### getCompassHeading public float getCompassHeading() Returns the current tilt-compensated compass heading value (in degrees, from 0 to 360) reported by the sensor. Note that this value is sensed by a magnetometer, which can be affected by nearby magnetic fields (e.g., the magnetic fields generated by nearby motors). Before using this value, ensure that (a) the magnetometer has been calibrated and (b) that a magnetic disturbance is not taking place at the instant when the compass heading was generated.Returns: The current tilt-compensated compass heading, in degrees (0-360).

#### zeroYaw public void zeroYaw() Sets the user-specified yaw offset to the current yaw value reported by the sensor. This user-specified yaw offset is automatically subtracted from subsequent yaw values reported by the getYaw() method.

#### isCalibrating public boolean isCalibrating() Returns true if the sensor is currently performing automatic gyro/accelerometer calibration. Automatic calibration occurs when the sensor is initially powered on, during which time the sensor should be held still, with the Z-axis pointing up (perpendicular to the earth). NOTE: During this automatic calibration, the yaw, pitch and roll values returned may not be accurate. Once calibration is complete, the sensor will automatically remove an internal yaw offset value from all reported values. Returns: Returns true if the sensor is currently automatically calibrating the gyro and accelerometer sensors.

#### isConnected public boolean isConnected() Indicates whether the sensor is currently connected to the host computer. A connection is considered established whenever communication with the sensor has occurred recently. Returns: Returns true if a valid update has been recently received from the sensor.

#### getActualUpdateRate public byte getActualUpdateRate() Returns the navX-Model device's currently configured update rate. Note that the update rate that can actually be realized is a value evenly divisible by the navX-Model device's internal motion processor sample clock (200Hz). Therefore, the rate that is returned may be lower than the requested sample rate. The actual sample rate is rounded down to the nearest integer that is divisible by the number of Digital Motion Processor clock ticks. For instance, a request for 58 Hertz will result in an actual rate of 66Hz (200 / (200 / 58), using integer math.Returns: Returns the current actual update rate in Hz (cycles per second).

#### getCurrentTransferRate public int getCurrentTransferRate() Returns the current number of data samples being transferred from the navX-Model device in the last second. Note that this number may be greater than the sensors update rate.Returns: Returns the count of data samples received in the last second.

#### getDuplicateDataCount public int getDuplicateDataCount()

#### getByteCount public double getByteCount() Returns the count in bytes of data received from the sensor. This could can be useful for diagnosing connectivity issues. If the byte count is increasing, but the update count (see getUpdateCount()) is not, this indicates a software misconfiguration.Returns: The number of bytes received from the sensor.

#### getUpdateCount public double getUpdateCount() Returns the count of valid updates which have been received from the sensor. This count should increase at the same rate indicated by the configured update rate.Returns: The number of valid updates received from the sensor.

#### getWorldLinearAccelX public float getWorldLinearAccelX() Returns the current linear acceleration in the X-axis (in G). World linear acceleration refers to raw acceleration data, which has had the gravity component removed, and which has been rotated to the same reference frame as the current yaw value. The resulting value represents the current acceleration in the x-axis of the body (e.g., the robot) on which the sensor is mounted. Returns: Current world linear acceleration in the X-axis (in G).

#### getWorldLinearAccelY public float getWorldLinearAccelY() Returns the current linear acceleration in the Y-axis (in G). World linear acceleration refers to raw acceleration data, which has had the gravity component removed, and which has been rotated to the same reference frame as the current yaw value. The resulting value represents the current acceleration in the Y-axis of the body (e.g., the robot) on which the sensor is mounted. Returns: Current world linear acceleration in the Y-axis (in G).

#### getWorldLinearAccelZ public float getWorldLinearAccelZ() Returns the current linear acceleration in the Z-axis (in G). World linear acceleration refers to raw acceleration data, which has had the gravity component removed, and which has been rotated to the same reference frame as the current yaw value. The resulting value represents the current acceleration in the Z-axis of the body (e.g., the robot) on which the sensor is mounted. Returns: Current world linear acceleration in the Z-axis (in G).

#### isMoving public boolean isMoving() Indicates if the sensor is currently detecting motion, based upon the X and Y-axis world linear acceleration values. If the sum of the absolute values of the X and Y axis exceed a "motion threshold", the motion state is indicated. Returns: Returns true if the sensor is currently detecting motion.

#### isRotating public boolean isRotating() Indicates if the sensor is currently detecting yaw rotation, based upon whether the change in yaw over the last second exceeds the "Rotation Threshold." Yaw Rotation can occur either when the sensor is rotating, or when the sensor is not rotating AND the current gyro calibration is insufficiently calibrated to yield the standard yaw drift rate. Returns: Returns true if the sensor is currently detecting motion.

#### getAltitude public float getAltitude() Returns the current altitude, based upon calibrated readings from a barometric pressure sensor, and the currently-configured sea-level barometric pressure [navX Aero only]. This value is in units of meters. NOTE: This value is only valid sensors including a pressure sensor. To determine whether this value is valid, see isAltitudeValid(). Returns: Returns current altitude in meters (as long as the sensor includes an installed on-board pressure sensor).

#### isAltitudeValid public boolean isAltitudeValid() Indicates whether the current altitude (and barometric pressure) data is valid. This value will only be true for a sensor with an onboard pressure sensor installed. If this value is false for a board with an installed pressure sensor, this indicates a malfunction of the onboard pressure sensor. Returns: Returns true if a working pressure sensor is installed.

#### getFusedHeading public float getFusedHeading() Returns the "fused" (9-axis) heading. The 9-axis heading is the fusion of the yaw angle, the tilt-corrected compass heading, and magnetic disturbance detection. Note that the magnetometer calibration procedure is required in order to achieve valid 9-axis headings. The 9-axis Heading represents the sensor's best estimate of current heading, based upon the last known valid Compass Angle, and updated by the change in the Yaw Angle since the last known valid Compass Angle. The last known valid Compass Angle is updated whenever a Calibrated Compass Angle is read and the sensor has recently rotated less than the Compass Noise Bandwidth (~2 degrees).Returns: Fused Heading in Degrees (range 0-360)

#### isMagneticDisturbance public boolean isMagneticDisturbance() Indicates whether the current magnetic field strength diverges from the calibrated value for the earth's magnetic field by more than the currently- configured Magnetic Disturbance Ratio. This function will always return false if the sensor's magnetometer has not yet been calibrated; see isMagnetometerCalibrated().Returns: true if a magnetic disturbance is detected (or the magnetometer is uncalibrated).

#### isMagnetometerCalibrated public boolean isMagnetometerCalibrated() Indicates whether the magnetometer has been calibrated. Magnetometer Calibration must be performed by the user. Note that if this function does indicate the magnetometer is calibrated, this does not necessarily mean that the calibration quality is sufficient to yield valid compass headings. Returns: Returns true if magnetometer calibration has been performed.

#### getQuaternionW public float getQuaternionW() Returns the imaginary portion (W) of the Orientation Quaternion which fully describes the current sensor orientation with respect to the reference angle defined as the angle at which the yaw was last "zeroed". Each quaternion value (W,X,Y,Z) is expressed as a value ranging from -2 to 2. This total range (4) can be associated with a unit circle, since each circle is comprised of 4 PI Radians. For more information on Quaternions and their use, please see this [definition](https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation).Returns: Returns the imaginary portion (W) of the quaternion.

#### getQuaternionX public float getQuaternionX() Returns the real portion (X axis) of the Orientation Quaternion which fully describes the current sensor orientation with respect to the reference angle defined as the angle at which the yaw was last "zeroed". Each quaternion value (W,X,Y,Z) is expressed as a value ranging from -2 to 2. This total range (4) can be associated with a unit circle, since each circle is comprised of 4 PI Radians. For more information on Quaternions and their use, please see this [description](https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation).Returns: Returns the real portion (X) of the quaternion.

#### getQuaternionY public float getQuaternionY() Returns the real portion (X axis) of the Orientation Quaternion which fully describes the current sensor orientation with respect to the reference angle defined as the angle at which the yaw was last "zeroed". Each quaternion value (W,X,Y,Z) is expressed as a value ranging from -2 to 2. This total range (4) can be associated with a unit circle, since each circle is comprised of 4 PI Radians. For more information on Quaternions and their use, please see: https://en.wikipedia.org/wiki/Quaternions\_and\_spatial\_rotationReturns: Returns the real portion (X) of the quaternion.

#### getQuaternionZ public float getQuaternionZ() Returns the real portion (X axis) of the Orientation Quaternion which fully describes the current sensor orientation with respect to the reference angle defined as the angle at which the yaw was last "zeroed". Each quaternion value (W,X,Y,Z) is expressed as a value ranging from -2 to 2. This total range (4) can be associated with a unit circle, since each circle is comprised of 4 PI Radians. For more information on Quaternions and their use, please see: https://en.wikipedia.org/wiki/Quaternions\_and\_spatial\_rotationReturns: Returns the real portion (X) of the quaternion.

#### getTempC public float getTempC() Returns the current temperature (in degrees centigrade) reported by the sensor's gyro/accelerometer circuit. This value may be useful in order to perform advanced temperature- correction of raw gyroscope and accelerometer values. Returns: The current temperature (in degrees centigrade).

#### getBoardYawAxis public [AHRS.BoardYawAxis](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.BoardYawAxis.html) getBoardYawAxis() Returns information regarding which sensor board axis (X,Y or Z) and direction (up/down) is currently configured to report Yaw (Z) angle values. NOTE: If the board firmware supports Omnimount, the board yaw axis/direction are configurable. For more information on Omnimount, please see: http://navx-mxp.kauailabs.com/navx-mxp/installation/omnimount/ Returns: The currently-configured board yaw axis/direction.

#### getFirmwareVersion public java.lang.String getFirmwareVersion() Returns the version number of the firmware currently executing on the sensor. To update the firmware to the latest version, please see: http://navx-mxp.kauailabs.com/navx-mxp/support/updating-firmware/ Returns: The firmware version in the format [MajorVersion].[MinorVersion]

#### getRawGyroX public float getRawGyroX() Returns the current raw (unprocessed) X-axis gyro rotation rate (in degrees/sec). NOTE: this value is un-processed, and should only be accessed by advanced users. Typically, rotation about the X Axis is referred to as "Pitch". Calibrated and Integrated Pitch data is accessible via the [getPitch()](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getPitch--) method. Returns: Returns the current rotation rate (in degrees/sec).

#### getRawGyroY public float getRawGyroY() Returns the current raw (unprocessed) Y-axis gyro rotation rate (in degrees/sec). NOTE: this value is un-processed, and should only be accessed by advanced users. Typically, rotation about the T Axis is referred to as "Roll". Calibrated and Integrated Pitch data is accessible via the [getRoll()](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getRoll--) method. Returns: Returns the current rotation rate (in degrees/sec).

#### getRawGyroZ public float getRawGyroZ() Returns the current raw (unprocessed) Z-axis gyro rotation rate (in degrees/sec). NOTE: this value is un-processed, and should only be accessed by advanced users. Typically, rotation about the T Axis is referred to as "Yaw". Calibrated and Integrated Pitch data is accessible via the [getYaw()](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getYaw--) method. Returns: Returns the current rotation rate (in degrees/sec).

#### getRawAccelX public float getRawAccelX() Returns the current raw (unprocessed) X-axis acceleration rate (in G). NOTE: this value is unprocessed, and should only be accessed by advanced users. This raw value has not had acceleration due to gravity removed from it, and has not been rotated to the world reference frame. Gravity-corrected, world reference frame-corrected X axis acceleration data is accessible via the [getWorldLinearAccelX()](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getWorldLinearAccelX--) method. Returns: Returns the current acceleration rate (in G).

#### getRawAccelY public float getRawAccelY() Returns the current raw (unprocessed) Y-axis acceleration rate (in G). NOTE: this value is unprocessed, and should only be accessed by advanced users. This raw value has not had acceleration due to gravity removed from it, and has not been rotated to the world reference frame. Gravity-corrected, world reference frame-corrected Y axis acceleration data is accessible via the [getWorldLinearAccelY()](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getWorldLinearAccelY--) method. Returns: Returns the current acceleration rate (in G).

#### getRawAccelZ public float getRawAccelZ() Returns the current raw (unprocessed) Z-axis acceleration rate (in G). NOTE: this value is unprocessed, and should only be accessed by advanced users. This raw value has not had acceleration due to gravity removed from it, and has not been rotated to the world reference frame. Gravity-corrected, world reference frame-corrected Z axis acceleration data is accessible via the [getWorldLinearAccelZ()](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getWorldLinearAccelZ--) method. Returns: Returns the current acceleration rate (in G).

#### getRawMagX public float getRawMagX() Returns the current raw (unprocessed) X-axis magnetometer reading (in uTesla). NOTE: this value is unprocessed, and should only be accessed by advanced users. This raw value has not been tilt-corrected, and has not been combined with the other magnetometer axis data to yield a compass heading. Tilt-corrected compass heading data is accessible via the [getCompassHeading()](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getCompassHeading--) method. Returns: Returns the mag field strength (in uTesla).

#### getRawMagY public float getRawMagY() Returns the current raw (unprocessed) Y-axis magnetometer reading (in uTesla). NOTE: this value is unprocessed, and should only be accessed by advanced users. This raw value has not been tilt-corrected, and has not been combined with the other magnetometer axis data to yield a compass heading. Tilt-corrected compass heading data is accessible via the [getCompassHeading()](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getCompassHeading--) method. Returns: Returns the mag field strength (in uTesla).

#### getRawMagZ public float getRawMagZ() Returns the current raw (unprocessed) Z-axis magnetometer reading (in uTesla). NOTE: this value is unprocessed, and should only be accessed by advanced users. This raw value has not been tilt-corrected, and has not been combined with the other magnetometer axis data to yield a compass heading. Tilt-corrected compass heading data is accessible via the [getCompassHeading()](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.html#getCompassHeading--) method. Returns: Returns the mag field strength (in uTesla).

#### getPressure public float getPressure() Returns the current barometric pressure (in millibar) [navX Aero only]. This value is valid only if a barometric pressure sensor is onboard.Returns: Returns the current barometric pressure (in millibar).

#### getDimStateTrackerInstance public [AHRS.DimStateTracker](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.DimStateTracker.html) getDimStateTrackerInstance()

[Skip navigation links](#3cqmetx)

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/com/kauailabs/navx/ftc/package-summary.html)
* Class
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-all.html)
* [Help](http://docs.google.com/help-doc.html)
* Prev Class
* [Next Class](http://docs.google.com/com/kauailabs/navx/ftc/AHRS.BoardAxis.html)
* [Frames](http://docs.google.com/index.html?com/kauailabs/navx/ftc/AHRS.html)
* [No Frames](http://docs.google.com/AHRS.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* [Nested](#3znysh7) |
* Field |
* [Constr](#2et92p0) |
* [Method](#tyjcwt)
* Detail:
* Field |
* [Constr](#1t3h5sf) |
* [Method](#2s8eyo1)