

VMX-pi Robotics Controller/Motion & Vision Processor User's Guide

Kauai Labs

Copyright — 2019

Table of Contents

Overview	2
VMX-pi	2
Features	3
Technical Specifications	5
Hardware Reference Manual	5
Frequently Asked Questions	6
Installation	9
Assembly with Raspberry Pi	9
RoboRIO Installation	12
Orientation	16
OmniMount	19
Creating an Enclosure	21
Software	23
Software	23
Raspberry Pi	23
RoboRIO	23
Windows	24
Examples	26
Raspberry Pi	26
HAL (C++, Java, C# and Python)	26
Analog Inputs	26
CAN Bus Monitor	26
CAN TX Loopback	26
Digital Inputs	26
Digital Outputs	27
Encoders	27
I2C	27
IMU	27
Interrupts	27
PWM Generation	27
Real-time Clock (RTC)	27
SPI	27
UART	27
VMX-pi Configuration	28
RoboRIO (FRC)	28
Field-Oriented Drive	28
Rotate to Angle	28
Straight-line Driving	29
Automatic Balancing	29
Collision Detection	29
Motion Detection	29
Data Monitor	30
Guidance	31
Guidance	31
Best Practices	31

Terminology	32
Gyro/Accelerometer Calibration	38
Magnetometer Calibration	41
Support	43
Support	43
Forum	43
Updating Firmware	43
Factory Test Procedure	44
Advanced	46
USB Serial Protocol	46
navXUI Customization	51
Technical References	52

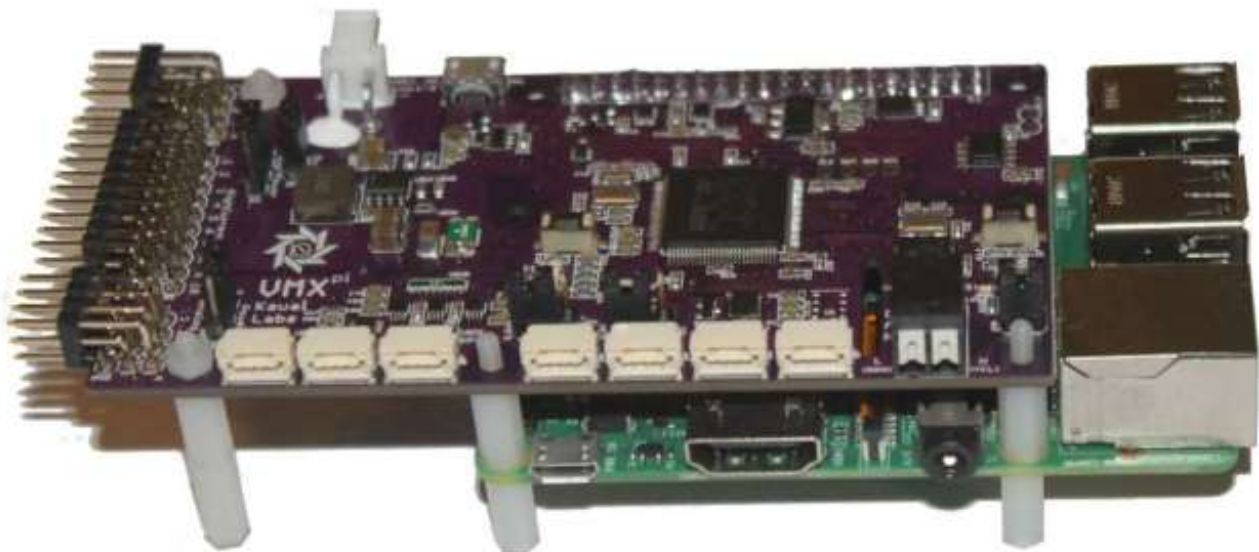
Overview VMX-pi



VMX-pi transforms your Raspberry Pi into a reliable, real-time **Robotics Controller or Vision/Motion Processor with embedded IMU & CAN-bus interface**. VMX-pi plus Raspberry Pi can perform both real-time robotic control and higher-layer Robot Position Tracking, Drivetrain path-planning and kinematics-based control – remotely accessed via Ethernet, Wifi or Bluetooth.

VMX-pi is the heart of an intelligent robot, and includes software [libraries](#), [example code](#) and many more [features](#).

VMX-pi also makes a great Vision/Motion Coprocessor, especially when combined with the [VMX-rtk Robotics Toolkit](#). And VMX-pi CAN Bus connectivity provides an easy way to monitor a robot CAN Bus.



VMX-pi assembled with a Raspberry Pi 3.

Super-charge your robot:

- Switched-mode Power supply for Raspberry Pi and external devices, including under-voltage management and over-current/short-circuit protection
- 30 Digital IOs and 4 Analog Inputs including circuit protection against over-voltage, and locking connectors for many functions
- Digital Communication Interfaces including CAN, SPI, I2C and UART
- navX-technology 9-axis IMU enabling Motion processing features including [Field-Oriented Drive](#), [Auto-balance](#), [Auto-rotate to angle](#), [Collision Detection](#) and more
- Network Time Server w/Battery-backed Real-time clock for synchronizing data and logs on distributed networked robot controllers

Additional Key Features:

- Write code directly on VMX-pi: Functions either as a standalone Development System (via Raspberry Pi inputs for keyboard, mouse, HDMI monitor), or can be used with a remote development environment
- Easily connect to sensors: Breakout boards are available for easily connecting to the Sparkfun [QWIIC Connect System](#) family of I2C sensors.

Power up with VMX Robotics Toolkit:

[VMX Robotics Toolkit \(VMX-rtk\)](#) Raspberry Pi Software Images add many features including Video Streaming, Vision Processing Libraries, NTP Server, Real-Time Linux (PREEMPT_RT), ROS Kinetic, WPI Network Tables and more:

- Vision processing with multiple cameras via Raspberry Pi-based OpenCV libraries and USB or Raspberry Pi Camera interfaces
- Video streaming and Video capture to SD Card from Raspberry Pi
- Robot Localization via Motion/Vision Processing fusion with sensor fusion packages (e.g., via the ROS [robot_pose_ekf](#) package)
- Integration with environment mapping sensors (e.g., stereo cameras & 2D or 3D LIDAR) to implement Simultaneous Localization and Mapping (SLAM) (e.g., via the ROS [gmapping](#) package)

Features

Raspberry Pi Power Management & Communication

- Switched-mode power supply for Raspberry Pi (up to 2.1A, enough to power many USB-connected devices)
- Conforms to [Raspberry Pi "HAT" specification](#) for tight integration w/Raspberry Pi 3B+, Raspberry Pi 3B and Raspberry Pi Zero/W
- High-speed internal communication between VMX-pi realtime IO controller and

Raspberry PI host enables high performance

- Automatic current-limiting ensures Raspberry Pi is powered even when external devices (or short-circuits) exceed current limits

External Device Power Management

- Selectable 5V or 3.3V Power Supply for external sensors and devices, up to .5A
- Voltage Translation for signals to/from 5V and 3.3V external sensors and devices

IMU with Sophisticated Motion Processing

- High Accuracy, Low-latency Yaw, Pitch and Roll Angles, w/update rate from 4-200Hz
- Orientation Quaternions and Gravity-corrected Linear Acceleration
- Tilt-compensated Compass Heading
- Automatic Accelerometer/Gyroscope Calibration
- High-sensitivity Motion Detection
- 9-Axis absolute heading w/Magnetic disturbance detection

30 Digital Interface Channels

- A total of 30 digital channels, each configurable for multiple functions at 3.3V or 5V
- **CAN 2.0b**, **SPI**, **I2C** and **UART** interfaces for communication with external sensors and devices
- Breakout boards for easily connecting to the Sparkfun [QWIC Connect System](#) family of I2C sensors.
- **USB** Interface allows IMU data to be simultaneously transmitted to remote USB-connected devices
- **Digital Outputs** support **PWM** Output for controlling motors and servos
- **Digital Inputs** can also be used for **Interrupts** (handled on the Raspberry Pi) and **PWM Capture**
- Hardware-based **Quadrature Encoder** decoding

4 Analog Interface Channels

- 4-channels of 12-bit digitized analog data
- Analog Triggering and on-board Oversampling/Averaging engine

Synchronization Support

- 1-microsecond resolution system clock for timestamping of sensor data
- Battery-backed Real-time Clock (RTC) allowing use as a Network Time Protocol (NTP) server, supporting ROS data synchronization and accurate log timestamps

Intelligent, Secure Packaging

- Locking connectors for Power, CAN, Digital Communications and Inputs
- A custom enclosure can be created with a 3D printer using provided [Enclosure](#) design files, or purchased [online](#)

Easy to Use

- Power cables for connection to battery and standard WallWart Power supply provided
- Additional connection to external devices (configuration/calibration/monitoring PC) via user-provided USB-micro Cable
- IMU orientation flexibility allowing mounting in horizontal, vertical and even upside-down configurations via [Omnimount](#)
- Tools for Magnetometer calibration
- Available Raspberry Pi SD card images with pre-installed toolstreams and sufficient storage for large amounts of acquired sensor data

Technical Specifications

The VMX-pi circuit board and official firmware provide inertial and magnetic measurements, with a range, accuracy and update rate as described on this page, as well as a broad range of I/O and Digital Communication capabilities and flexible Power Management.

Note that certain IMU performance specifications are only valid after a start-up [Gyroscope/Accelerometer Calibration](#) period, during which time the VMX-pi circuit board must be held still.

Additional details can be found in the [VMX-pi-datasheet](#).

Additionally, the specifications for the connector cables which come with each VMX-pi are also documented on this page.

Electrical Specifications

Input Voltage:	6-16V DC
Output Voltages:	5V, 3.3V
Protection Features:	Input Voltage Clamps; Under-voltage management; Output Current Limiting
Communications Interface:	USB, I2C, SPI, CAN, UART
Power Connector:	2-Pin JST VH Connector
USB Connector:	USB Micro-B

Hardware Reference Manual



The [VMX-pi Hardware Reference Manual](#) describes all VMX-pi hardware components and resources in detail.

Frequently Asked Questions

Which Raspberry Pi board versions are compatible with VMX-pi?

- Raspberry Pi 3 Model B+
- Raspberry Pi 3 Model B
- Raspberry Pi 2 Model B
- Raspberry Pi Zero/W
- Raspberry Pi Zero

Can VMX-pi be used with the National Instruments RoboRIO™?

Yes, in several ways.

1. **Ethernet.** In this configuration, by using the WPI NetworkTables library, vision processing detection events performed on the Raspberry Pi can be exchanged with the RoboRIO. Additionally, VMX-pi's battery-backed real-time clock and Network Time Protocol Daemon (NTPD) can be used to synchronize timestamps between RoboRIO and VMX-pi, ensuring log file timestamps on both systems are the same.
2. **USB.** The VMX-pi USB port is compatible with navX-MXP and navX-Micro; therefore, by connecting a USB cable from VMX-pi to RoboRIO, VMX-pi's IMU data can be accessed via existing navX-FRC Libraries.
3. **CAN.** The VMX-pi can connect to the same CAN bus as the RoboRIO; although there is currently no direct CAN communication with RoboRIO, this connection does allow a FRC CAN bus to be monitored.

Please see the instructions for [installing VMX-pi on a FRC Robot](#) for more details.

Can VMX-pi be used to control devices (e.g., motors, servos and relays) on a FRC competition robot?

VMX-pi is not legal for *competition FRC actuator control*.

However VMX-pi is definitely legal for vision and motion processing, and can also be used to monitor the CAN bus.



And VMX-pi can be used by FRC teams to build off-season robots and other devices such as “T-shirt launchers”.

Aren't the magnetometer (compass heading) readings unreliable when the VMX-pi is used on a Robot with powerful motors?

Yes, this is correct. If the VMX-pi is mounted nearby any energized motors, the magnetometer's ability to measure the (weak) earth's magnetic field is severely diminished.

Magnetometer readings can be taken before motors are enabled to establish “initial orientation”, which can then be updated during operation.

Why do the Yaw angles provided by the VMX-pi drift over time?

The short answer is that the yaw angle is calculated by integrating reading from a gyroscope which measures changes in rotation, rather than absolute angles. Over time, small errors in the rotation measurements build up over time. VMX-pi features sophisticated digital motion processing and calibration algorithms that limit this error in the yaw angle of ~1 degree per minute. For further details, please see the [Yaw Drift](#) page.

Can the VMX-pi “Displacement” estimates be used for tracking a robot's change in position (dead-reckoning)?

Accelerometer data from the VMX-pi onboard MPU-9250 sensor are double-integrated by the VMX-pi firmware to estimate displacement, *and are accurate to approximately 1 meter of error during a 15 second period.*

While the accuracy of the VMX-pi displacement estimates might be good enough to track the position of an automobile on a road, this accuracy is too low for use in tracking a robot's position with high accuracy.

The root cause of the displacement estimate error rate is *accelerometer noise*. Estimating displacement requires first that each acceleration sample be multiple by itself twice (cubed), and then integrated over time. Practically, if a noisy signal is cubed, the result is very noisy, and when this very noisy value is integrated over time, the total



amount of error grows very quickly.

The current noise levels (approximately 150 micro-g per square-root-hertz) would need to be reduced by a factor of 100 (two orders of magnitude) before displacement estimates with 1 cm of error per 15 seconds can be achieved by double-integration of accelerometers.

MEMS accelerometers which feature these low noise levels are beginning to emerge, but are currently very expensive. KauaiLabs is actively researching these technology developments and projects that MEMS technology that is both (a) low noise (1 micro-g per square root hertz) and (b) available at low cost will be available in approximately 3 years (~2020). KauaiLabs plans to develop a product which can be used for accelerometer-based dead-reckoning at that time.

What's the difference between VMX-pi, navX-MXP and navX-Micro?

- navX-MXP was designed from the ground-up to interface easily with the RoboRIO, making it simple to connect to the robot controller. navX-MXP provides FRC-legal digital inputs and outputs.
- The navX-Micro is a smaller version of the navX-MXP which is designed for use in robots such as those found in FIRST FTC robotics as well as FRC.
- VMX-pi contains navX-technology, in addition to much other functionality needed by a robot controller. The VMX-pi IMU data can be accessed on a RoboRIO by using the existing navX-FRC libraries and connecting to VMX-pi's USB port.

What's the difference between the VMX-pi and the VMX-pi Aero?

VMX-pi and VMX-pi Aero share a single design. VMX-pi Aero adds a pressure sensor (MS5611) providing additional altitude measurements with a resolution of 10 cm.

Since the pressure sensor is an expensive component, this sensor was left off of VMX-pi, decreasing the cost for those not desiring an altitude measurement.

Installation

Assembly with Raspberry Pi

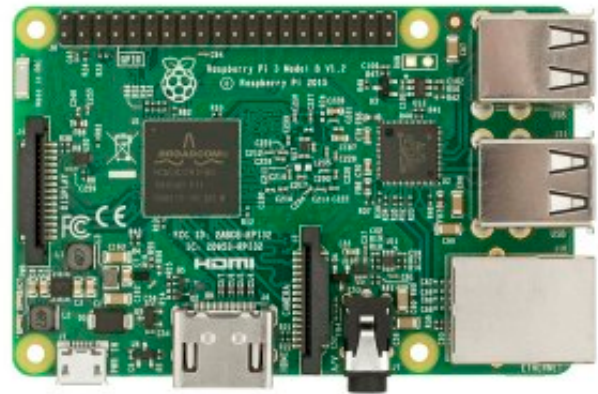
VMX-pi is simple to assemble together with several different Raspberry Pi Circuit boards, including the Raspberry Pi 3 (Models B and B+) and the Raspberry Pi Zero/W.

Raspberry Pi 3 (Models B and B+)



[Raspberry Pi 3 Model B+](#) (top view)

1.4Ghz Quad-Core Processor, 1GB RAM

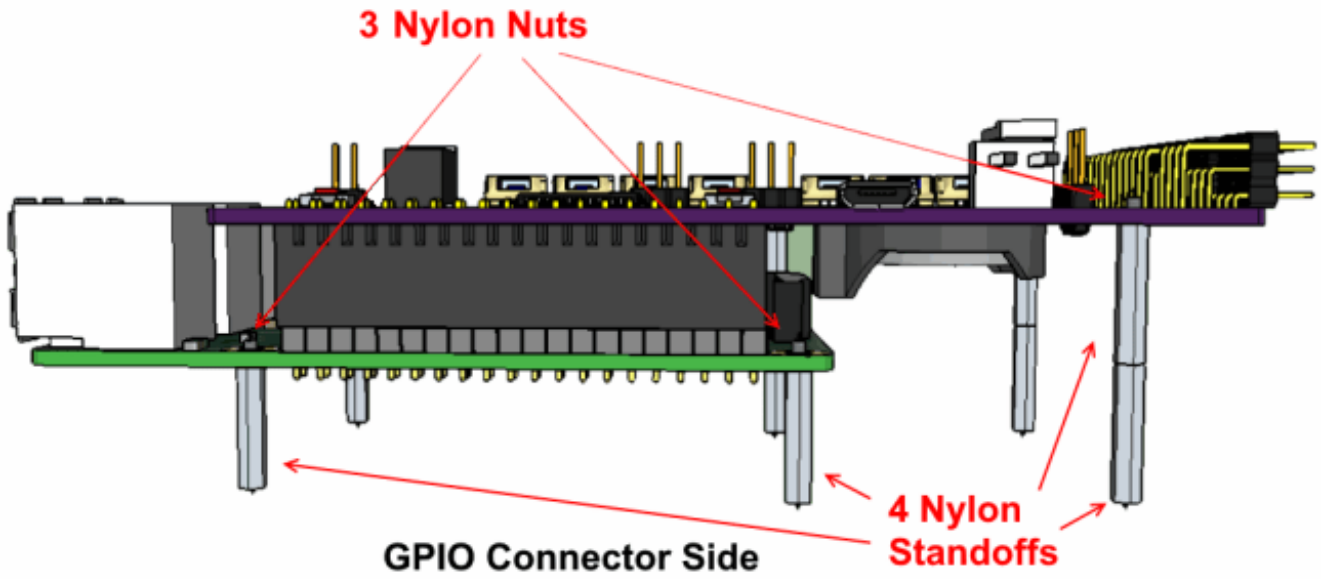


[Raspberry Pi Model 3B](#) (Top View)

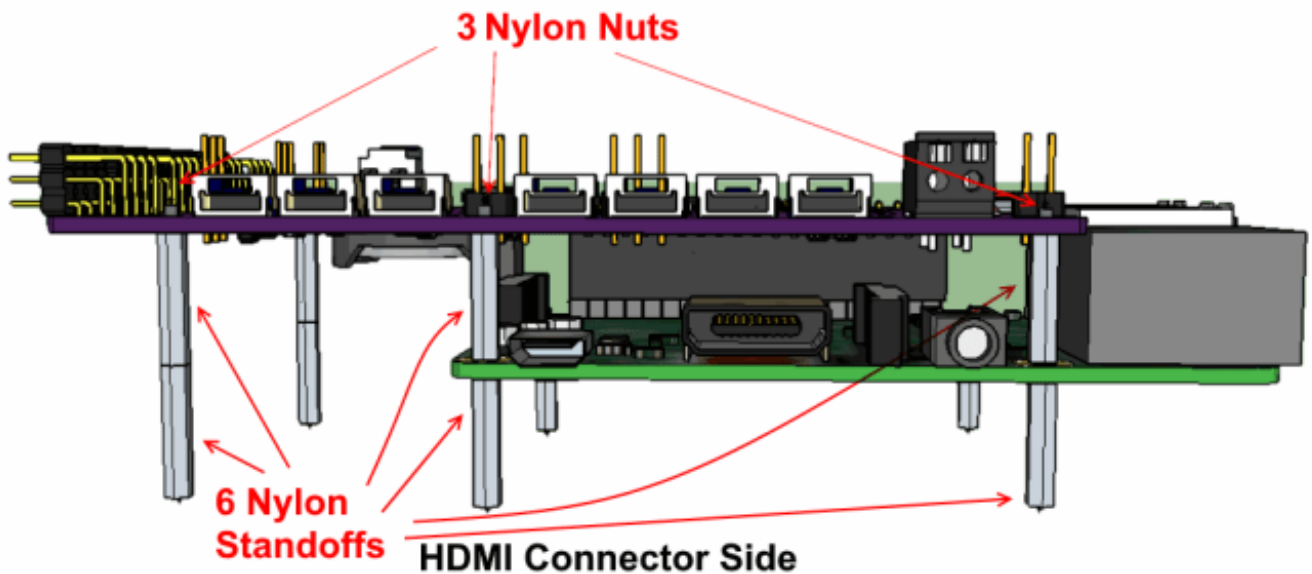
1.2Ghz Quad-Core Processor, 1GB RAM

The Raspberry Pi 3 (purchased separately; retail cost ~\$35) includes a 40-pin GPIO Connector Block – with all 40 pins pointing up – that should be connected to the corresponding 40 pin connector on VMX-pi.

VMX-pi includes ten (10) Nylon Standoffs and six (6) Nylon Nuts which can be used as shown in the video above to fix the two boards relative to each other and to create a set of stable “legs” upon which to mount the assembly onto a Robot, or to rest it on a desktop.



VMX-pi Standoff Installation – GPIO Connector Side



VMX-pi Standoff Installation – HDMI Connector Side

The assembly video also demonstrates the installation of the two provided Power Connectors: A Wall-wart (barrel-jack) connector for desktop use and a Battery connector for on-robot use.

Note: if using the Wall-wart connector, you will need to [purchase separately a AC DC Adapter](#). The adapter must have a center-positive barrel connector, with 2.1mm inner diameter and 5.5mm outer diameter. The recommended adapter has a 12 Volt DC output rated to 3 Amps.

Finally, the video demonstrates how a JST-GH cable and breakout board can be used to connect to external devices.

Note: as an alternative to using standoffs, an [enclosure](#) can be 3D-printed and used instead of the standoffs to create a solid enclosure for protecting and mounting VMX-pi.

Raspberry Pi Zero W



[Raspberry Pi Zero W](#) (top view)

1Ghz Single-Core, 512MB RAM

For those seeking an even lower-cost solution, VMXI-pi can be used with a Raspberry Pi Zero W (purchased separately; retail cost: ~\$10) instead of the Raspberry Pi 3.

By default, the Raspberry Pi Zero W does not have the 40-pin GPIO Pin Header Block soldered to it. Typically, this block is included with the Raspberry Pi Zero W, but can also be purchased separately at sites including [Adafruit](#) and [Pimoroni](#).



40-pin Header (2 rows)

The first step is to solder the 40-pin GPIO Connector Block to the Raspberry Pi Zero W, with the long ends of the header block point upwards from the top of the board.

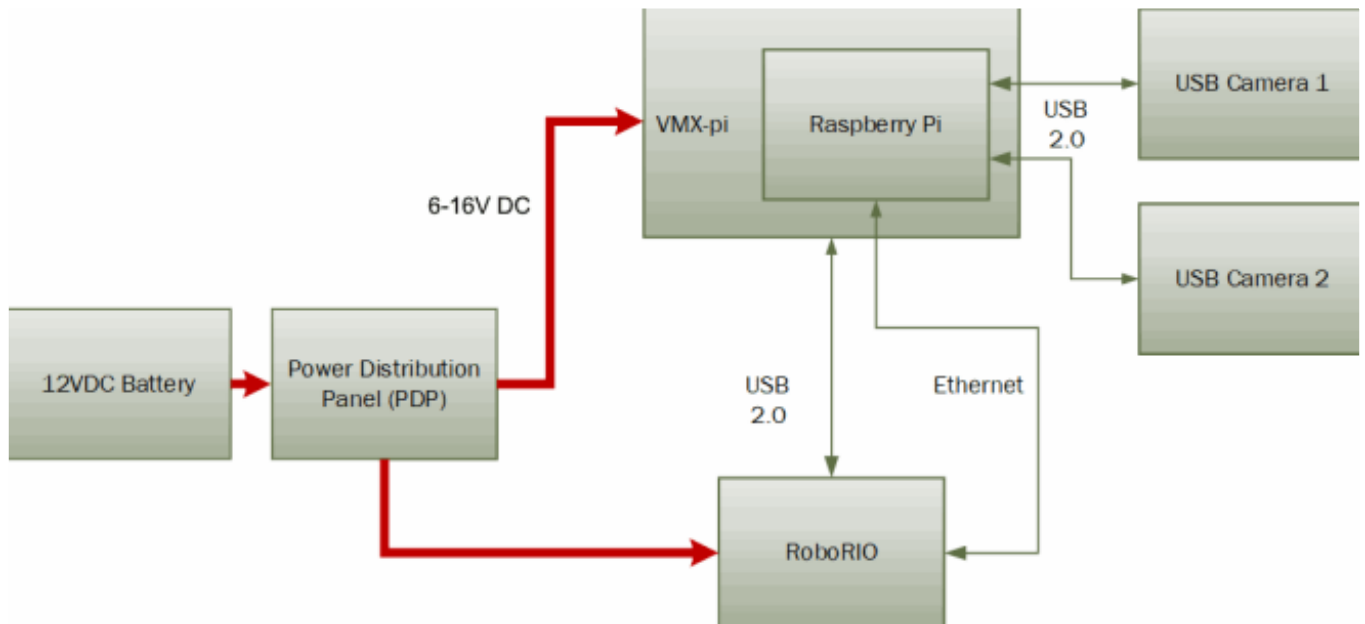
Then, the standoffs included with VMX-pi can be used as shown below to stabilize the two boards relative to each other and to create a set of stable “legs” upon which to mount the assembly onto a Robot.

Alternatively, an enclosure can be 3D-printed and used instead of the standoffs to create an solid [enclosure](#) which can be mounted to the Robot.

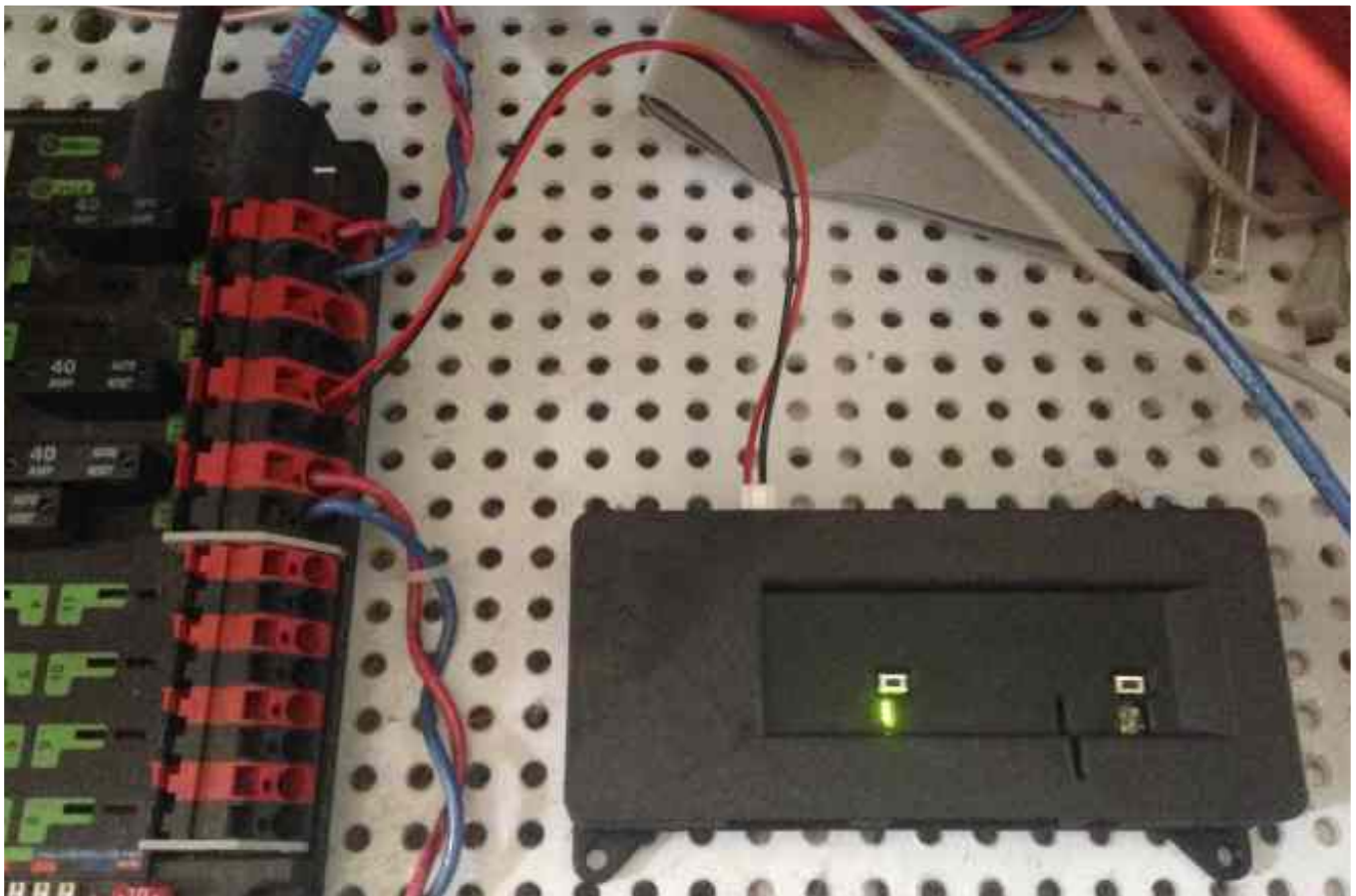
RoboRIO Installation

VMX-pi may be powered by unregulated 12VDC from the Power Distribution Panel (PDP) and communicate with the RoboRIO via several methods:

- Ethernet
- USB
- CAN



Power



VMX-pi [shown in enclosure] powered by 12V battery via Power Distribution Panel (PDP).

As shown in the above picture, the VMX Battery Adapter Cable can connect directly to one of the outputs on the FRC Power Distribution Panel (PDP); the VMX-pi onboard voltage regulator converts the unregulated voltage internally to 5V and 3.3V – and can draw up to a maximum of 3 Amps.

Ethernet

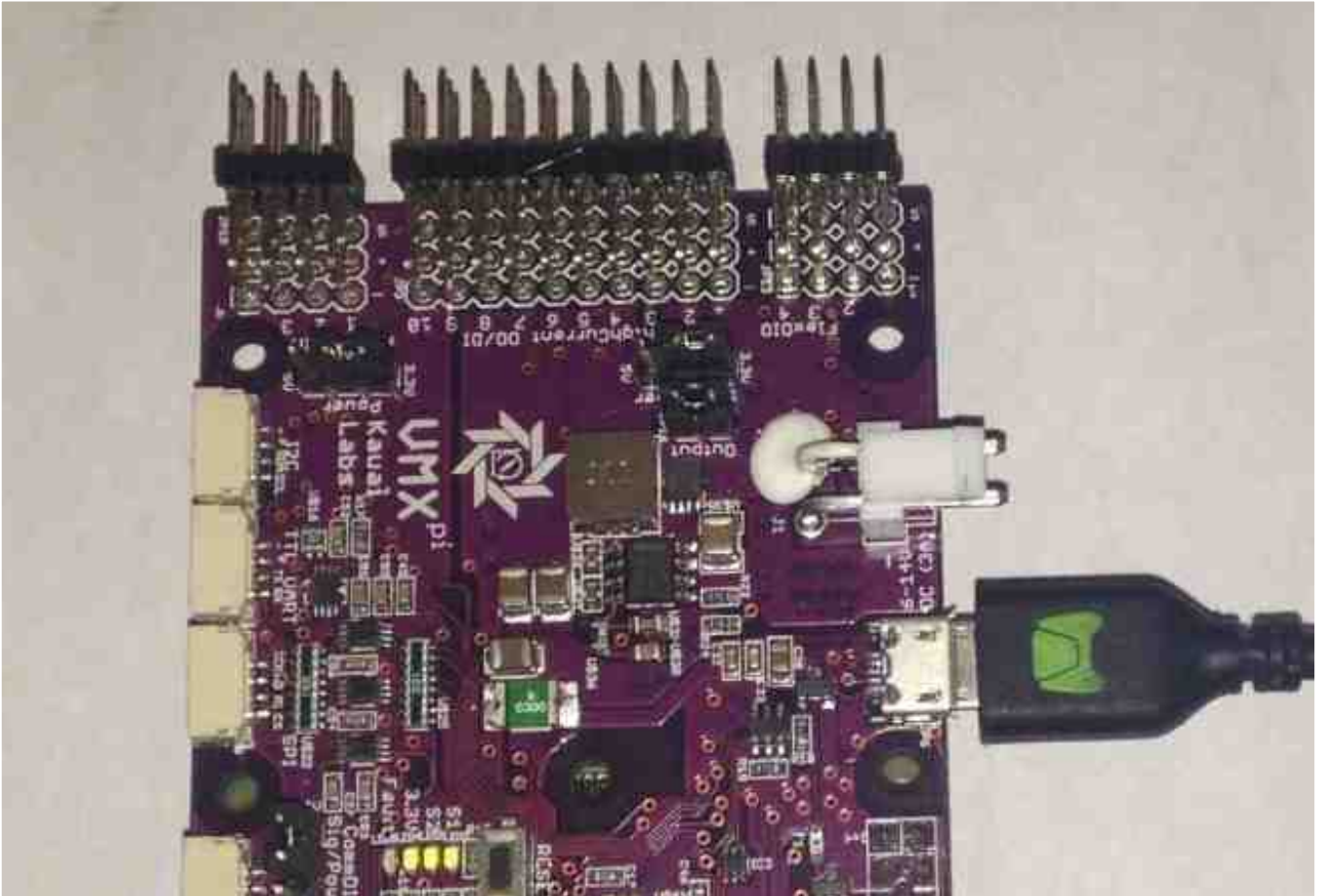
When connected to a Raspberry Pi, the VMX-pi can communicate with the RoboRIO via Ethernet, providing several key features:

- Notification of Video Processing Detection Events
- Streaming of video from one or more Raspberry Pi-connected cameras (via Mjpeg-streamer), and
- Updating the RoboRIO system clock with the battery-backed Real-time Clock via the Network Time Protocol (NTP)



USB

When connected to the RoboRIO via USB, the VMX-pi navX-technology (which is compatible with navX-MXP and navX-Micro sensors) can send real-time orientation data that is accessible via navX-FRC libraries on the RoboRIO for many features.



CAN

When connected to the same CAN 2.0b bus as the RoboRIO, VMX-pi can be used to monitor the CAN bus traffic between all devices on the CAN bus. When combined with real-time logging, this can enable system-level logging of events in CAN-enabled robots.

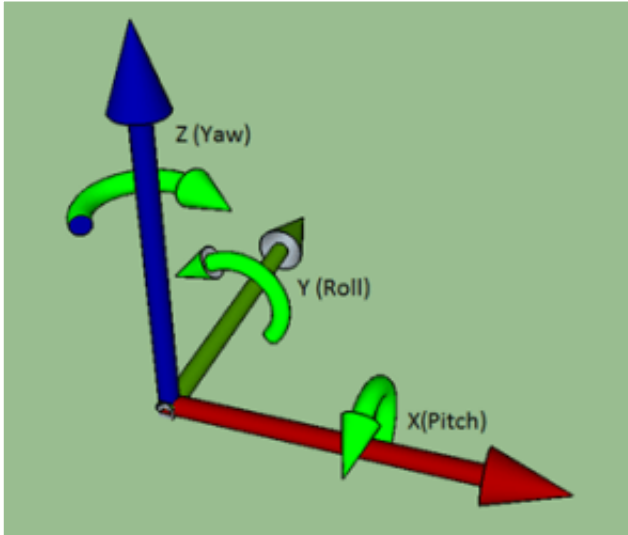


Orientation

VMX-pi measures a total of 9 sensor axes (3 gyroscope axes, 3 accelerometer axes and 3 magnetometer axes) and fuses them into a 3-D coordinate system. In order to effectively use the values reported by VMX-pi, a few key concepts must be understood in order to correctly install VMX-pi on a robot.

3-D Coordinate System

When controlling a robot in 3 dimensions a set of 3 axes are combined into a 3-D coordinate system, as depicted below:



In the diagram above, the green rounded arrows represent Rotational motion, and the remaining arrows represent Linear motion.

Axis	Orientation	Linear motion	Rotational Motion
X (Pitch)	Left/Right	- Left / + Right	+ Tilt Backwards
Y (Roll)	Forward/Backward	+ Forward / - Backward	+ Roll Left
Z (Yaw)	Up/Down	+ Up / - Down	+ Clockwise/ - Counter-wise

More discussion on the 3-D Coordinate system is on the [Terminology](#) page.

Reference Frames

Note that the 3-axis coordinate system describes relative motion and orientation; it doesn't specify the orientation with respect to any other reference. For instance, what does "left" mean once a robot has rotated 180 degrees?

To address this, the concept of a [reference frame](#) was invented. There are three separate three-axis "reference frames" that should be understood:

Coordinate System	Technical Term	X Axis	Y Axis
Field	World Frame	Side of Field	Front (Head) of Field
Robot	Body Frame	Side of Robot	Front (Head) of Robot
navX-Micro	Board Frame	See diagram Below	See diagram below

Joysticks and Reference Frames



Since a three-axis joystick is typically used to control a robot, the robot designer must select upon which Reference Frame the driver joystick is based. This selection of Reference Frame typically depends upon the drive mode used:

Drive mode	Reference Frame	Coordinate Orientation
Standard Drive	Body Frame	Forward always points to the front (head) of the robot
Field-oriented Drive	World Frame	Forward always points to the front (head) of the field

VMX-pi Board Orientation (Board Frame)

Aligning Board Frame and Body Frame

In order for the VMX-pi orientation sensor readings to be easily usable by a robot control application, the VMX-pi Coordinate System (Board Frame) must be aligned with the Robot Coordinate system (Body Frame).

Aligning the Yaw (Z) axis and Gravity

The VMX-pi motion processor takes advantage of the fact that gravity can be measured with its onboard accelerometers, fusing this information with the onboard gyroscopes to yield a very accurate yaw reading with a low rate of drift. In order to accomplish this, ***the yaw (Z) axis must be aligned with the “gravity axis” (the axis that points directly up and down with respect to the earth).***

When installing VMX-pi on a robot, the VMX-pi yaw (Z) axis and the gravity axis must be aligned.

Default VMX-pi Board Orientation

The default VMX-pi circuit board orientation is with the VMX-pi logo on the Front *Right*, with the top of the circuit board pointing up (with respect to the earth).

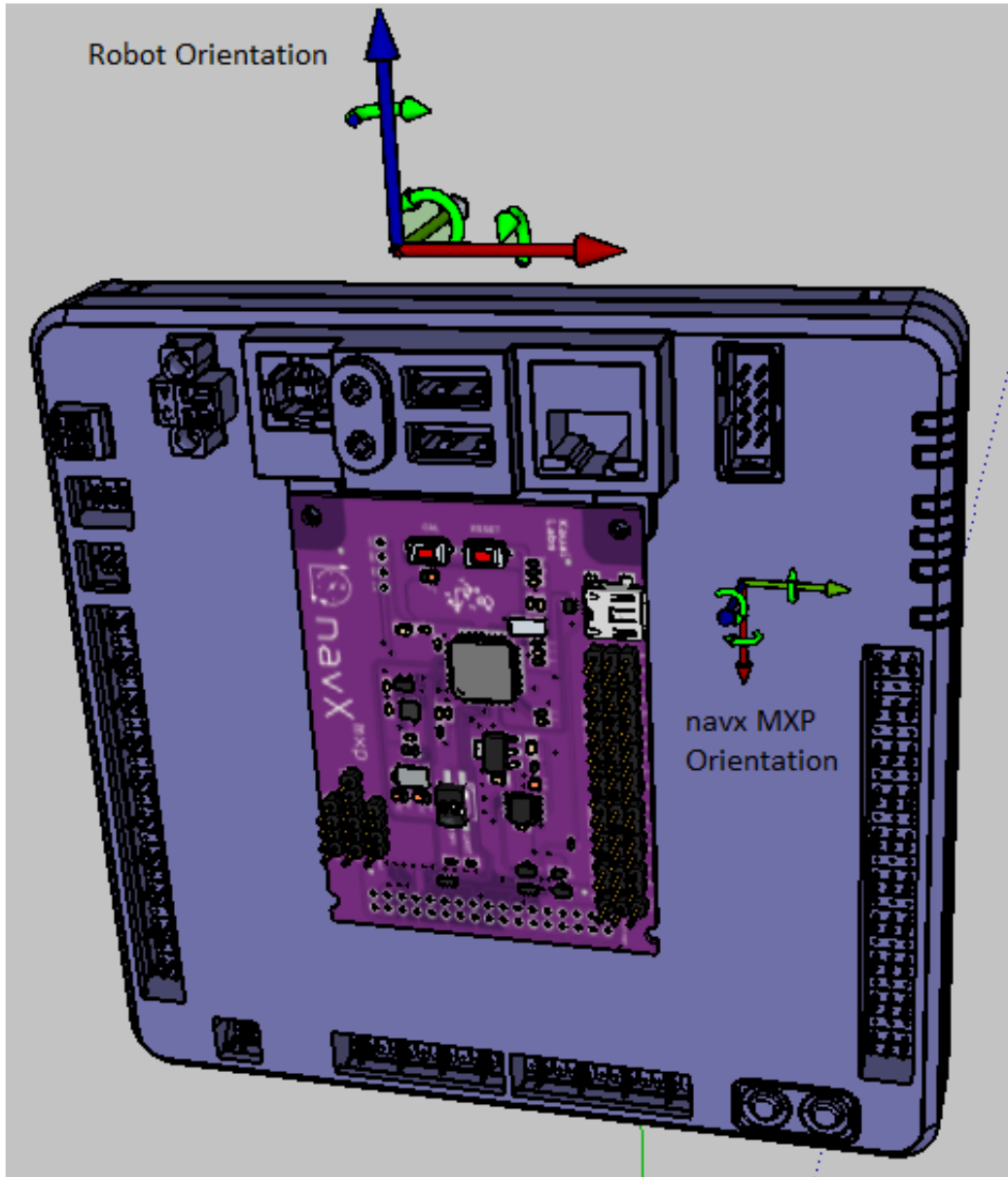
Since Body Frame and Board Frame coordinates should be aligned, and because the Yaw axis must be aligned with gravity, by default you must orient the navX-Micro with the top of the board facing up, and with the Y axis (on the circuit board) pointing to the front of the robot.

If you need to mount the VMX-pi circuit board in a different orientation (vertically, horizontally, or upside down), you can use the [OmniMount](#) feature to transform the orientation.

OmniMount

If the VMX-pi [default yaw axis orientation](#) isn't sufficient for your needs, you can use the **OmniMount** feature to re-configure the VMX-pi yaw axis, allowing high-accuracy yaw axis readings when VMX-pi is mounted horizontally, vertically, or even upside down.

In certain cases, the VMX-pi axes (Board Frame) may not be oriented exactly as that of the Robot (Body Frame). For instance, if the VMX-pi circuit board is mounted sideways, the navX-Micro axes will not be oriented identically to the Robot. This configuration is shown in the diagram below:



Transforming VMX-pi Board Frame to Body Frame with OmniMount

VMX-pi's "OmniMount" feature can transform the VMX-pi X, Y and Z axis sensor data (Board Frame) into Robot Orientation (Body Frame) by detecting which of its three axes is perpendicular to the earth's surface.

This is similar to how a modern smart phone will rotate the display based upon the phone's orientation. However unlike a smart phone, the OmniMount detection of orientation does not happen all the time – since the orientation should not change while the robot is moving. Rather, each time OmniMount configuration occurs, VMX-pi records this transformation in persistent flash memory, and will continue to perform this transformation until the transform is reconfigured.

To configure OmniMount, follow these simple steps:

- Install VMX-pi onto your robot. ENSURE that one of the VMX-pi axes (as shown on the VMX-pi circuit board) is perpendicular to the earth's surface. This axis will become the yaw (Z) axis. Note that this axis can either be pointing away from the earth's surface, or towards the earth's surface.
- Press the 'CAL' button on the VMX-pi Circuit board AND HOLD THE BUTTON DOWN FOR AT LEAST 5 SECONDS.
- Release the 'CAL' button, and verify that the orange 'CAL' light flashes for 1 second and then turns off.
- Press the 'RESET' button on the VMX-pi circuit board, causing it to restart.
- The VMX-pi circuit board will now begin OmniMount auto-calibration. During this auto-calibration period, the orange 'CAL' LED will flash repeatedly. This process takes approximately 15 seconds, and requires two things:
 - 1. During auto-calibration, **one of the VMX-pi axes MUST be perpendicular to the earth's surface.**
 - 2. During auto-calibration, **the VMX-pi must be held still.**
 - If either of the above conditions is not true, the 'CAL' LED will be flashing quickly, indicating an error. To resolve this error, you must ensure that conditions 1 and 2 are met, at which point the 'CAL' LED will begin flashing slowly, indicating calibration is underway.
- Once the VMX-pi auto-calibration is complete, the Board Frame to Body Frame Transform will be stored persistently into VMX-pi flash memory and used until auto-calibration is run once again.

Creating an Enclosure

The VMX-pi circuit board contains sensitive circuitry, and should be handled carefully.

An enclosure is recommended to protect the VMX-pi circuit board from excessive handling, ["swarf"](#), electro-static discharge (ESD) and other elements that could potentially damage the VMX-pi circuitry.

An enclosure for the VMX-pi is available.

[Build vs. Buy](#)

Those who prefer to print the enclosure using their personal 3D printer, enclosure design files is available in the "enclosure" directory of the .

Those who prefer to purchase the enclosure can order it from . The price will be approximately \$65 plus shipping, depending upon the type of material used.



[Design Files](#)

The enclosure design files include:

File	Description
Enclosure Bottom 5 Print.STEP	Enclosure Base (Solidworks Design File)
Enclosure Top 5 Print.STEP	Enclosure Base (Solidworks Design File)
Enclosure Bottom 5 Print.STL	Enclosure Base (STL File for 3D Printing)
Enclosure Top 5 Print.STL	Enclosure Base (STL File for 3D Printing)

Software Software



VMX-pi includes software which makes VMX-pi easy to understand, integrate and use with a robot for navigation and control. This software includes the following components:

- Libraries and Tools for developing [Raspberry pi](#)-based and ROS-based robot applications.
- Libraries for accessing VMX-pi from a [RoboRIO](#)-based FIRST FRC robot.
- Tools for managing/configuring VMX-pi and access VMX-pi data from [Windows](#).

Raspberry Pi

Several libraries and tools are available for use with VMX-pi.

These tools allow lower-level and higher-level robotics applications to be developed, and also include a large toolkit of tools, services and libraries useful for robotics software development.

- [Getting Started on Raspberry Pi](#) provides guidance on configuring your Raspberry Pi correctly to work with VMX-pi, based on your usage model.
- The [VMX-pi Hardware Abstraction \(HAL\) Library](#) provides low-level access to all VMX-pi functions on a Raspberry Pi using C++, Java, C# or Python.
- The [VMX-rtk Robotics Toolkit](#) is a suite of pre-configured software for building robotics applications including tools and libraries for ROS and FIRST FRC.
- The [VMX-pi ROS Node](#) allows ROS-based robotics applications to access VMX-pi capabilities either directly from the host Raspberry Pi, or remotely across a network.

RoboRIO

Libraries for accessing VMX-pi IMU data from a National Instruments RoboRIO®-based FRC Robot are now available.

To access VMX-pi from a FRC RoboRIO-based system, select the **USB interface option** available in the navX-MXP RoboRIO libraries.

Note: the link below leads to the navX-MXP website.

[RoboRIO Libraries](#)

Windows

VMX-pi Tools for Windows provide several useful tools which help update VMX-pi firmware, configure the real-time motion-processing algorithms, perform advanced calibration of the Magnetometer.

These tools are not required to use VMX-pi, but can be useful in certain situations. Kauai Labs recommends all VMX-pi customers install the VMX-pi Tools for Windows.

navXUI

Additionally, VMX-pi Tools for Windows includes the [navXUI](#) user interface application provides a simple way to visualize the data provided by VMX-pi.

Configuration/Calibration Tools

VMX-pi software includes several tools for [magnetometer calibration](#) and [advanced configuration](#). These tools run on a Windows PC and communicate with VMX-pi via USB.

NOTE: The Configuration/Calibration tools are provided for use by advanced users; please carefully read the tool descriptions before using them.

Installing/Running VMX-pi Tools for Windows

- Make sure Java 7 (version 1.7) or Java 8 (version 1.8) is installed on your computer. The 64-bit version of Java is recommended. To tell which version of java is currently “Active”, open up a command window, and type this command:

```
java -version
```

- Download the and unzip the contents to your local computer.
- To run the various tools, VMX-pi must be connected to a PC running Windows by connecting the VMX-pi micro USB connector to a PC USB connector.
- Run the setup.exe program, which will install navXUI, as well as all necessary device

drivers for communicating over USB with VMX_pi, as well as some additional tools.

Examples Raspberry Pi

Many examples are available for the Raspberry Pi, when using VMX-pi as a Robotics Controller or a Robotics Co-processor.

[HAL \(C++, Java, Python, C#\)](#): The HAL Examples demonstrate how to access/control all VMX-pi functionality from applications written in various programming languages running directly on the VMX-pi host Raspberry Pi.

ROS (C++, Python): The ROS Examples demonstrate how to access/control all VMX-pi functionality from a C++ or Python application running either directly on the VMX-pi host Raspberry Pi, or remotely with communication via Network.

HAL (C++, Java, C# and Python)

The Raspberry Pi HAL examples demonstrate how applications running directly on the VMX-pi host Raspberry Pi – written in a variety of programming languages – can access all VMX-pi functionality for use in Robotics Controller and Robotics Co-processor applications.

[table id=3 /]

Analog Inputs

CAN Bus Monitor

CAN TX Loopback

Digital Inputs

Digital Outputs

Encoders

I2C

IMU

Interrupts

PWM Generation

Real-time Clock (RTC)

SPI

UART

VMX-pi Configuration

RoboRIO (FRC)

To access VMX-pi from a FRC RoboRIO-based system, select the USB interface option available in the navX-MXP RoboRIO libraries.

Each example includes C++, Java and LabVIEW samples.

Note: each example section links to a corresponding page code on the navX-MXP website.

Field-Oriented Drive

Note: the link below leads to the navX-MXP website.

[Field-Oriented Drive \(FRC\)](#)

To access VMX-pi from a FRC RoboRIO-based system, select the USB interface option available in the navX-MXP RoboRIO libraries.

Rotate to Angle

Note: the link below leads to the navX-MXP website.

[Rotate to Angle \(FRC\)](#)

To access VMX-pi from a FRC RoboRIO-based system, select the USB interface option

available in the navX-MXP RoboRIO libraries.

Straight-line Driving

Driving a robot in a straight line using navX-Micro is very similar in nature to the Rotate to Angle example, the only difference being that the robot will move in a direction, and continually adjust the motors, just as in the Rotate to Angle example, to ensure that angular heading is maintained.

This example also includes the functionality to “reset” the “yaw” angle, so that the heading when forward motion starts is equivalent to zero degrees.

The PID Controller coefficients defined in the example code will need to be tuned for your drive system, and you may wish to adjust the update rate from the sensor to match the speed at which you wish to update the motors with new motor speeds.

FTC Android Example

Automatic Balancing

Note: the link below leads to the navX-MXP website.

[Automatic Balancing \(FRC\)](#)

Collision Detection

Note: the link below leads to the navX-MXP website.

[Collision Detection \(FRC\)](#)



Motion Detection

Note: the link below leads to the navX-MXP website.

[Motion Detection \(FRC\)](#)

Data Monitor

Note: the link below leads to the navX-MXP website.

[Data Monitor \(FRC\)](#)

Guidance Guidance



To ensure the highest possible accuracy and reliability, several recommendations regarding installation and use are provided.

This includes a set of [best practices](#) which summarize key factors which help ensure successful operation.

VMX-pi also features several calibration processes to ensure easy-to-use, high-reliability operation. This includes automatic [gyroscope/accelerometer calibration](#), as well as manual [magnetometer calibration](#). This section discusses these calibration processes and the theory behind them, including how to ensure minimal [yaw drift](#).

Best Practices

This page summarizes the recommended best practices when integrating VMX-pi with a robot, such as a FIRST FRC robot. Following these best practices will help ensure high reliability and consistent operation.

[1\) Secure VMX-pi circuit board to the Robot Chassis](#)

Excessive vibration will reduce the quality of VMX-pi orientation sensor measurements. The VMX-pi circuit board should be [mounted](#) in such a way that it is firmly attached to the robot chassis.

2) Understand and Plan for Calibration

[Gyro/Accelerometer Calibration](#) is vital to achieving high-quality VMX-pi IMU readings. Be sure to understand this process, and ensure that it completes successfully each time you use the robot.

If your robot moves during calibration, or if noticeable temperature changes occur during calibration, the calibration process may take longer than normal.

Using the VMX-pi yaw angle before calibration completes may result in errors in robot control. To avoid this situation, your robot software should verify that calibration has completed before using VMX-pi IMU data.

3) Protect the Circuitry

VMX-pi contains sensitive circuitry. The VMX-pi circuit board should be handled carefully.

An [enclosure](#) is recommended to protect the VMX-pi circuit board from excessive handling, “swarf”, electro-static discharge (ESD) and other elements that could potentially damage VMX-pi circuitry.

5) Provide a “Zero Yaw” feature (for Field-Oriented Drive)

The VMX-pi gyro “yaw” angle will [drift](#) over time (approximately 1 degree/minute). While this does not normally impact the robot during a typical FRC match, if using field-oriented drive during extended practice sessions it may be necessary to periodically “zero” the yaw. Drivers should be provided a simple way (e.g., a joystick button) with which to zero the yaw.

6) If possible, mount VMX-pi near the center of rotation

Since VMX-pi measures rotation, errors in the measured angles can occur if VMX-pi is mounted at a point not near the robot center of rotation. For optimal results, VMX-pi should be mounted at the robot’s center of rotation. If VMX-pi cannot be mounted near the robot’s center of rotation, the offset from the center of rotation can be used to correct the yaw angle.

7) Use [OmniMount](#) if VMX-pi is not mounted horizontally

By default, VMX-pi’s motion processing requires the unit be mounted horizontally, parallel to the earth’s surface; the yaw (Z) axis should be perpendicular to the earths surface.

If you need to mount VMX-pi vertically or upside-down, you will need to enable the [“OmniMount”](#) feature in order to get reliable, accurate yaw (Z) axis readings.

8) Learn how the sensor behaves by using the navXUI

The [navXUI](#) provides insight into the key VMX-pi IMU features, and can help debug issues you may encounter when integrating VMX-pi onto your robot. Running this user interface is highly recommended for anyone using VMX-pi.

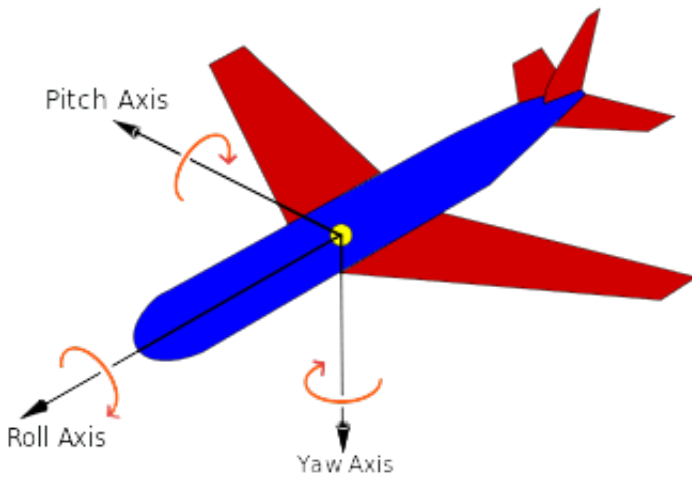
Terminology

Several terms used throughout the VMX-pi libraries and documentation may not be commonly understood and are defined herein.

Basic Terminology

A working knowledge of the following Basic Terminology is highly recommended when working with VMX-pi or any other Inertial Measurement Unit (IMU).

Pitch, Roll and Yaw



Pitch, Roll and Yaw are measures of angular rotation about an object’s center of mass, and together provide a measure of “orientation” of that object with respect to an “at rest” position. When units of degrees are used, their range is from -180 to 180 degrees, where 0 degrees represents the “at rest” position of each axis.

Axis	Orientation relative to object’s center of mass	Rotational Motion
X (Pitch)	Left/Right	+ Tilt Backwards
Y (Roll)	Forward/Backward	+ Roll Left
Z (Yaw)	Up/Down	+ Clockwise/ – Counter-wise

Important Note: Pitch, Roll and Yaw angles represent rotation from the “origin” (0,0,0) of a 3-axis coordinate system. VMX-pi Pitch and Roll angles are referenced to earth’s gravity – so when VMX-pi is flat, Pitch and Roll angles should be 0.

The Yaw angle is different – Yaw is not referenced to anything external. When VMX-pi startup calibration completes, the Yaw angle is automatically set to 0 – thus at this point, 0 degrees represents where the “head” of the VMX-pi circuit board is pointing. The Yaw angle can be reset at any time after calibration completes if a new reference direction is desired.

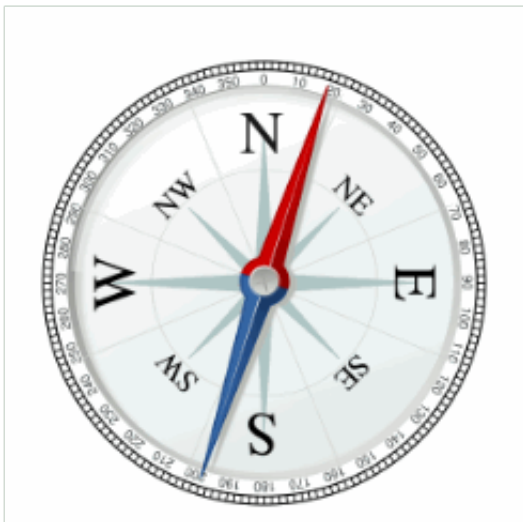
Linear Acceleration

Linear Acceleration is a measure of the change in velocity in a specific direction. For example, when a car starts from a standstill (zero relative velocity) and travels in a straight line at increasing speeds, it is accelerating in the direction of travel.

Axis	Orientation	Linear motion
X	Left/Right	– Left / + Right
Y	Forward/Backward	+ Forward / – Backward
Z	Up/Down	+ Up / – Down

Because the gyroscope and accelerometer axes are aligned, VMX-pi measures linear acceleration in the same set of 3 axes used to measure Pitch, Roll and Yaw. However unlike Pitch, Roll and Yaw, acceleration measures linear motion rather than rotation, and is measured in units of G, with a range of +/- 2.0.

Compass Heading



A compass measures the earth’s magnetic field and indicates the current direction (heading) relative to magnetic north (N). Compass Heading is measured in degrees and is similar to Yaw, but has a few key differences:

- Compass Heading has a range of 0-360 (where magnetic north is 0).
- Compass Heading is absolute – it is referenced to magnetic north, and thus Compass Heading does not drift over time

Important Note 1: Compass Heading relies upon being able to measure the earth’s magnetic field. Since the earth’s magnetic field is weak, Compass Heading may not be able to measure earth’s magnetic field when the compass is near a strong magnetic field such as that generated by a motor.

Important Note 2: [Magnetic North is not exactly the same as True North](#). Your robot can calculate True North given a Magnetic North reading, as long as the current declination is known. Declination is a measure of the difference in angle between Magnetic North and True North, and changes depending upon your location on earth, and also changes over time at that same location. An [online calculator](#) is provided allowing one to calculate declination for a given earth location and date.

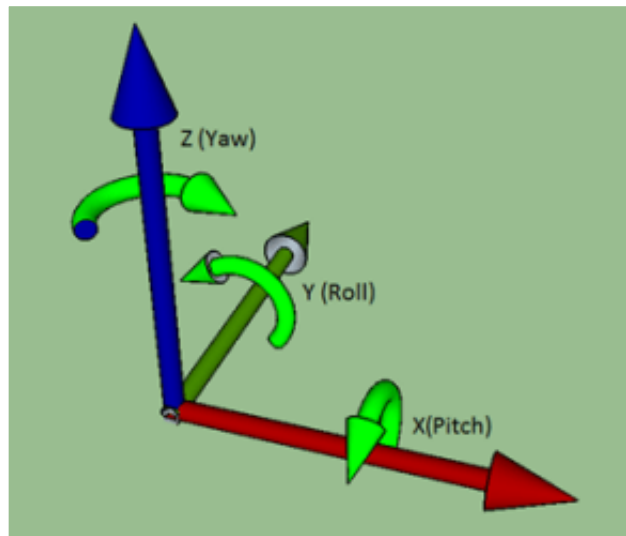
Altitude

Altitude is a measure of distance in the “up” direction from a reference; VMX-pi (Aero Edition) calculates altitude above sea-level using a pressure sensor.

VMX-pi (Aero Edition) altitude has a range of 0 to 25,000 meters.

Important Note: Altitude is calculated based upon barometric pressure. In order to accurately estimate altitude above the earth, VMX-pi should be configured with the sea-level barometric pressure in the surrounding area. This setting can be configured via the VMX-pi [Advanced Configuration Tool](#).

3-D Coordinate System



VMX-pi 3-D Coordinate System

A 3-D Coordinate System uses one or more numbers (coordinates), often used to uniquely determine the position of a point within a space measured by that system. The origin of a 3-D coordinate system has a value of (0, 0, 0).

VMX-pi features gyroscopes, accelerometers and magnetometers which are all aligned with each other in a 3-D coordinate system. Each sensor type measures values with respect to that coordinate system, as follows:

Gyroscopes: measure rotation (as shown in the green arrows) about each axis. The coordinate system origin represents the center of the VMX-pi circuit board.

Accelerometers: measure acceleration, where the origin represents the position in space at which the previous acceleration sample was acquired.

Magnetometers: measure earth's magnetic field, where the origin represents the center of the VMX-pi circuit board.

Important Note: Because the VMX-pi Gyroscopes, Accelerometers and Magnetometers are all aligned to this 3-D Coordinate System, VMX-pi's motion processor can also use Sensor Fusion to provide additional information and processing including Tilt Correction, "Fused Heading", a Gravity Vector, World Reference Frame-based Linear Acceleration and Quaternions, as discussed in the Motion Processing section below.

Motion Processing

Users should also have a working knowledge of the terms defined in the Motion Processing Terminology.

Tilt Correction

Without correction, the compass heading calculated by a 3-axis magnetometer will only be accurate if the magnetometers are held "flat" with respect to the earth. To ensure the compass heading is valid even in cases when the sensor is "pitched" (Pitch angle $\neq 0$) or "rolled" (Roll angle $\neq 0$), VMX-pi performs "tilt correction" fusing the reading from the magnetometers with the pitch and roll angles from the accelerometers. Once corrected, the compass heading is aligned with the VMX-pi Z axis, which ensures that the Yaw angle and the Compass Heading measure rotation similarly.

"Fused" Heading

Given the gravity-referenced orientation provided by the Yaw angle, as well as the absolute compass heading angle which has been aligned to the VMX-pi 3-D coordinate system, both angles can be fused together. As shown in the table below, over a period of several minutes this can minimize the drift inherent in the Yaw angle, as well as provide an absolute reference for the Yaw angle – as long as the magnetometer is calibrated and a valid magnetometer reading is available every minute or so.

Value	Accuracy	Update Rate	Drift
Yaw	.01 degrees	Up to 66 Hz	~1 degree/minute

Compass Heading	2 degrees	1 Hz (if not magnetically disturbed)	None
Fused Heading	2 degrees (as long as a valid magnetometer reading is received in the last minute or so)	Up to 66Hz	None (~1 degree/minute, during periods of magnetic disturbance)

Like the Compass Heading, the Fused Heading has a range from 0-360 degrees.

Important Note: If the Compass Heading is not valid, the Fused Heading origin is the same as the Yaw angle. When valid (magnetically undisturbed) compass readings are received, the Fused Heading's origin shifts to magnetic north (0 degrees on the Compass).

Gravity Vector

Accelerometers measure both acceleration due to gravity, as well as acceleration due to linear acceleration. This fact makes using raw accelerometer data difficult. VMX-pi's automatic accelerometer calibration determines the component of measured acceleration which corresponds to gravity, and uses this information together with gyroscope readings to calculate a gravity vector, which represents acceleration due to gravity. Pitch and Roll angles are derived from this gravity vector.

Once the gravity vector is understood, this value is then subtracted from the raw accelerometer data to yield the acceleration due to linear motion.

Velocity and Displacement

Acceleration is defined as the change in Velocity. Therefore, linear velocity can be calculated by integrating linear acceleration over time.

Velocity is defined as the change in Position, otherwise known as Displacement. Therefore, linear displacement can be calculated by integrating linear velocity over time.

Important Note: Using currently-available MEMS-based accelerometers to calculate linear velocity and displacement is subject to large amounts of error primarily due to accelerometer "noise" (a difference between the actual acceleration and the measured acceleration inherent with MEMS sensors). This noise not only accumulates, but is also squared in the case of velocity, and is cubed in the case of displacement. Therefore, the resulting estimated velocity and displacement values are not typically useful for robotic navigation. The amount of error in displacement estimation can be several feet per second. As MEMS sensors improve in the coming years and accelerometer noise is reduced by approximately 100 times its currently value, this technique will become more useful for robotics navigation.

If you would like to experiment with using the VMX-pi to calculate displacement and velocity, you can use the [navXUI](#)'s "Experimental" button to bring up a dialog which displays the

integrated velocity and displacement values calculated in real-time by the VMX-pi.

World Reference Frame

Raw acceleration data measures acceleration along the corresponding sensor axis. This measurement occurs in a reference frame known as “Body Reference Frame”. This works well as long as the VMX-pi circuit board is in its original orientation. However as the VMX-pi circuit board rotates, the X and Y accelerometer axes no longer point “forward/back” and “left/right” with respect to the original orientation. To understand this more clearly, consider how the meaning of the term “left” changes once a robot has rotated 180 degrees? Introducing a World Reference Frame solves this issue by providing a reference upon which to measure “leftness”.

To account for this, VMX-pi’s motion processing adjusts each linear acceleration value by rotating it in the opposite direction of the current yaw angle. The result is an acceleration value that represents acceleration with respect to the area in which VMX-pi operates, which is known as “World Reference Frame”. This world-frame linear acceleration value is much simpler to use for tracking motion of an object, like a robot, which might rotate while it moves.

Important Note: VMX-pi Linear Acceleration values are in World Reference Frame.

Advanced

Advanced users may require knowledge of the following terminology.

Quaternions

A [quaternion](#) is a four-element vector that can be used to encode any rotation in a 3D coordinate system. This single 4-element vector value can describe not only rotation about a reference frame’s origin (Pitch, Roll and Yaw) but also the rotation of that entire reference frame with respect to another. Furthermore, when Pitch, Roll and Yaw measures to perform certain calculations, it is not possible to clearly ascertain orientation when two axes are aligned with each other; this condition is referred to as “Gimbal Lock”. For robotics applications, Pitch, Roll and Yaw are sufficient, however for certain aerospace applications, Quaternions may be required to handle all possible orientations.

VMX-pi uses Quaternions internally, and also provides the 4 quaternion values for use by those who might need them.

Gyro/Accelerometer Calibration

VMX-pi onboard orientation sensors require calibration in order to yield optimal results. We highly recommend taking the time to understand this calibration process – successful calibration is vital to ensure optimal performance.

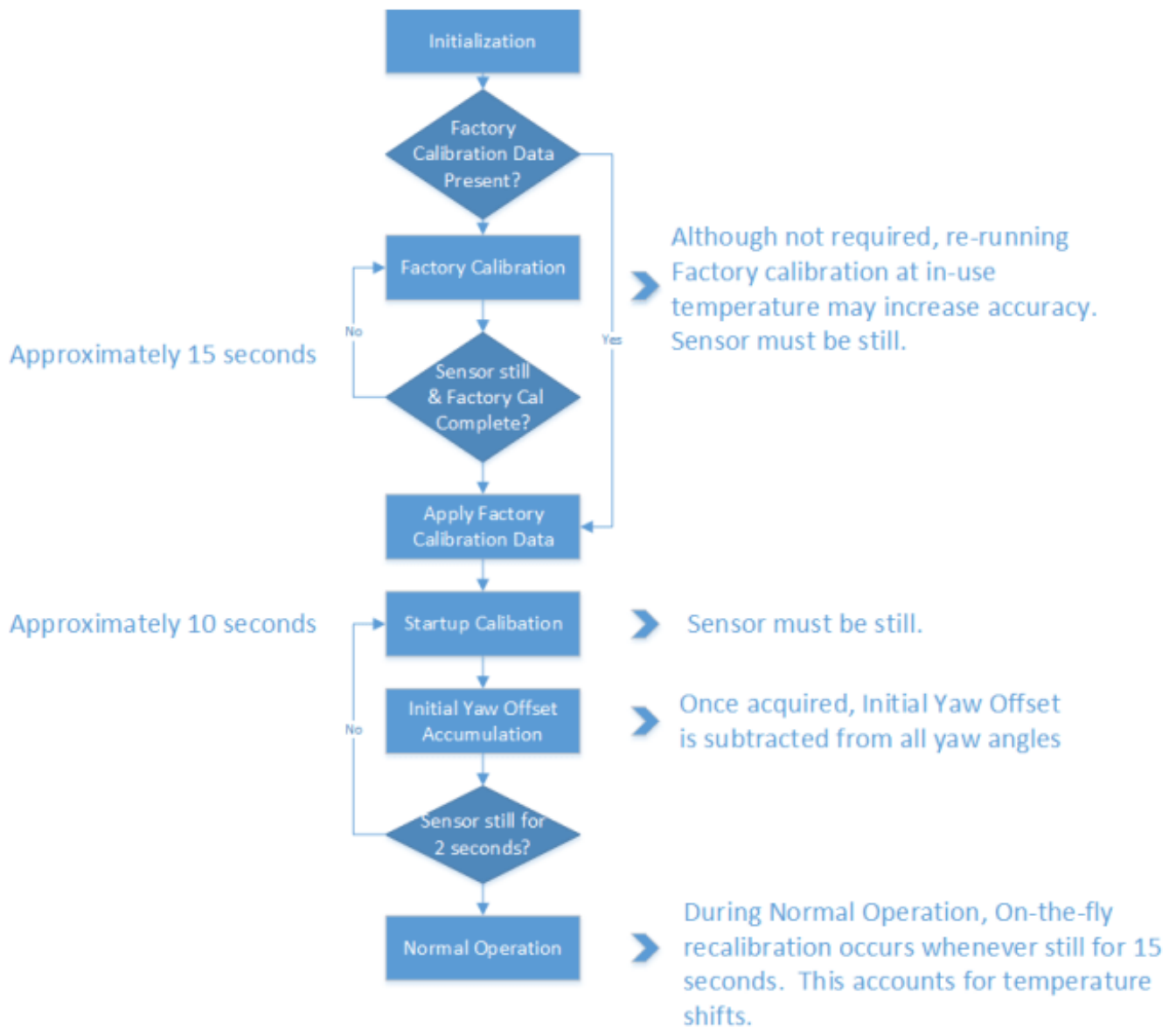
Accurate Gyroscope Calibration is crucial in order to yield valid yaw angles. Although this process occurs automatically, understanding how it works is required to obtain the best results.

If you are tempted to ignore this information, please read the section entitled “The Importance of Stillness” at the end of this page.

Calibration Process

The VMX-pi Calibration Process is comprised of three calibration phases:

- Factory Calibration
- Startup Calibration
- On-the-fly Calibration



[Factory Calibration](#)

Before VMX-pi units are shipped, the accelerometers and gyroscopes are initially calibrated at the factory; this calibration data is stored in flash memory and applied automatically to the accelerometer and gyroscope data each time the navX-Micro circuit board is powered on.

Note that the onboard gyroscopes are sensitive to temperature changes. Therefore, since the average ambient temperature at the factory (on the island of Kauai in Hawaii) may be different than in your environment, you can optionally choose to re-calibrate the gyroscope by pressing and holding the “CAL” button for at least 10 seconds. When you release the “CAL” button, ensure that the “CAL” Led flashes briefly, and then press the “RESET” button to restart navX-Micro. When VMX-pi is re-started, it will perform the Initial Gyro Calibration – the same process that occurs at our factory. *NOTE: It is very important to hold VMX-pi still, and parallel to the earth’s surface, during this Initial Gyro Calibration period.* You might consider performing this process before using your robot the first time it is used within a new environment (e.g., when you arrive at a FTC competition event).

The value of re-running Factory Calibration at the same temperature VMX-pi will be operated at is potentially increased yaw accuracy as well as faster Startup Calibration. If a significant temperature shift has occurred since the last Factory Calibration, the Startup Calibration time may take longer than normal, and it’s possible that yaw accuracy will be diminished until the next On-the-fly Gyro Calibration completes.

[Startup Calibration](#)

Startup Calibration occurs each time the VMX-pi is powered on, and requires that the sensor be held still in order to complete successfully. Using the Factory Calibration as a starting point, the sensor calibrates the accelerometers and adjusts the gyroscope calibration data as well based upon current temperature conditions.

If the sensor continues to move during startup calibration, Startup Calibration will eventually timeout – and as a result, the VMX-pi yaw angle may not be as accurate as expected.

[Initial Yaw Offset Calibration](#)

Immediately after Startup Calibration, an **Initial Yaw Offset** is automatically calculated. The purpose of the Initial Yaw Offset is to ensure that whatever direction the “front” of the VMX-pi circuit board is pointed to at startup (after initial calibration is applied) will be considered “0 degrees”.

Yaw Offset Calibration requires that VMX-pi be still for approximately 2 seconds after Startup Calibration completes. After approximately 2 seconds of no motion, VMX-pi will acquire the current yaw angle, and will subtract it from future yaw measurements automatically. The VMX-pi protocol and libraries provide a way to determine the yaw offset value it is currently using.



NOTE: If VMX-pi is moving during startup, this Yaw Offset Calibration may take much longer than 2 seconds, and may not be calculated at all if the sensor continues moving long enough. Therefore it is highly-recommended to keep VMX-pi still until initial calibration and Initial Yaw Offset calibration completes.

[On-the-fly Gyro Calibration](#)

In addition to Startup Calibration, during normal operation VMX-pi will automatically re-calibrate the gyroscope (e.g., to account for ongoing temperature changes) during operation, whenever it detects 8 seconds of no motion. This process completes after about 7-8 more seconds, and is completely transparent to the user. Therefore each time VMX-pi is still for approximately 15 seconds, the gyroscopes are re-calibrated “on-the-fly”. The purpose of On-the-fly Gyro re-calibration is to help maintain yaw accuracy when shifts in ambient temperature occur during operation.

This On-the-fly Gyro Calibration can help deal with cases where the sensor was moving during Startup Calibration, but note that the yaw is not zeroed at the completion of On-the-fly Calibration. So once again, it’s important to keep the sensor still during Startup Calibration.

[Runtime Yaw Zeroing](#)

Your robot software can optionally provide the robot operator a way to reset the yaw angle to Zero at any time. Please see the documentation for the [VMX-pi libraries](#) for more details.

The importance of stillness

This is the most important takeaway from this discussion: It is highly-recommended that VMX-pi be held still during the above Initial Gyro and Initial Yaw Offset calibration periods. In support of this, VMX-pi indicates when it is calibrating; we recommend you incorporate this information into your software. Please see the discussion of the [navXUI](#), and the [VMX-pi libraries](#) for more details on this indication.

Magnetometer Calibration

VMX-pi onboard orientation sensors require calibration in order to yield optimal results. We highly recommend taking the time to understand this calibration process – successful calibration is vital to ensure optimal performance.

Careful and accurate Magnetometer Calibration is crucial in order to yield valid compass heading, 9-axis heading and magnetic disturbance detection.

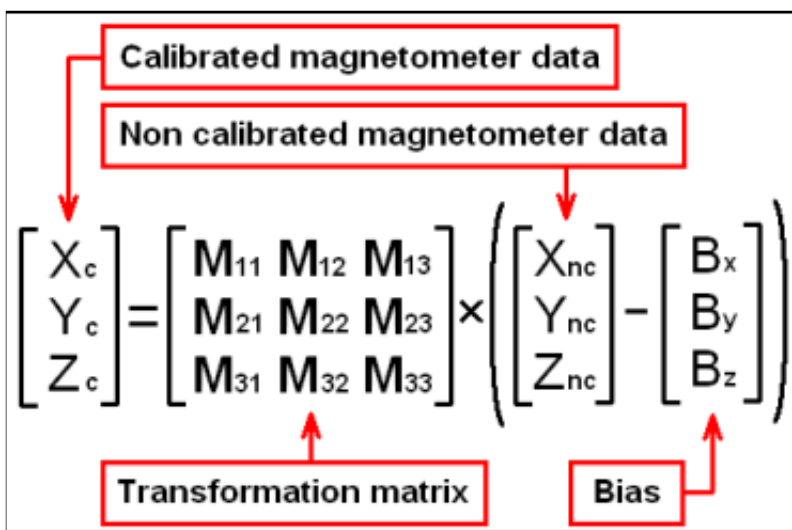
Magnetometer Calibration is not typically required for use in many robotics applications,

including Field-oriented drive. Magnetometer Calibration is a manual process and is recommended for advanced users who need to calculate absolute heading.

Calibration Process

The magnetometer calibration encompasses three areas: (a) hard-iron calibration, (b) soft-iron calibration and (c) magnetic disturbance calibration.

Hard and soft-iron calibration allows the following equation to be used, and corrects for hard and soft-iron effects due to nearby ferrous metals and magnetic fields. This calibration is necessary in order to achieve valid compass heading readings:



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \times \left(\begin{bmatrix} X_{nc} \\ Y_{nc} \\ Z_{nc} \end{bmatrix} - \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} \right)$$

In addition, using the same calibration data the strength of the Earth’s Magnetic Field is determined. Whenever the data from the magnetometer indicates the current magnetic field differs from the calibrated Earth’s Magnetic Field strength by more than the “Magnetic Disturbance Ratio”, a Magnetic Anomaly is declared.

Therefore, careful and accurate Magnetometer Calibration is crucial in order to yield valid compass heading, 9-axis heading and magnetic disturbance detection.

Magnetometer Calibration can be accomplished with a single, simple calibration process through the use of the [Magnetometer Calibration tool](#). This tool is designed to run on a Windows computer, and communicate to the VMX-pi circuit board via a USB cable.

Support Support



Please visit [VMX Support](#) if you are experiencing difficulty or trouble.

In addition, some common needs are addressed:

- Instructions for [updating the VMX-pi Firmware](#)
- The VMX-pi Discussion [Forum](#)
- A [“factory test” procedure](#) which can verify the VMX-pi circuit board is functioning properly

Forum

Updating Firmware

Requirements

- VMX-pi Circuit Board (rev. 5.35 or higher)
- PC with USB 2.0 port running Windows 7 or greater.
- Micro-USB Cable

[Updating the Firmware](#)

- Download the VMX-pi Tools for Windows [latest build](#).
- Unpack the contents of the vmx-pi.zip file and run the setup.exe program, which will install the tools as well as all necessary device drivers for communicating over USB with the VMX-pi, as well as some additional tools. In addition, the setup program will install the latest firmware at the following location:

<HomeDirectory>\vmx-pi\firmware

For example, if your user name is Robot, the directory name will be C:\Users\Robot\navx-micro\firmware.

- Within that directory, the firmware file will be named using this pattern:

vmx-pi_X.Y.ZZZ.hex

(X = Major Version Number Y = Minor Version Number Z = Revision Number)

1. Press and hold down the “CAL” button on the VMX-pi circuit board. While holding this button down, connect a USB-micro cable from a Windows PC to the VMX-pi circuit board. Use the micro-usb connector immediately to the left of the VMX-pi power connector. Applying power when the “CAL” button is held down places the board into “bootloader” mode, at which point the firmware can be loaded.
2. From your Start Menu, select “Kauai Labs” and then click on the VMXFirmwareUpdater menu item, and follow the directions included in the program.
3. Once you have downloaded the firmware, you can use the “Currently-loaded Firmware Version” tab of the VMXFirmwareUpdater to verify the version number you have just installed.

Factory Test Procedure

The VMX-pi Factory Test Procedure verifies correct operation of the circuit board and it’s key components. The VMX-pi Factory Test Procedure is performed in the factory to verify initial correct operation, and may be run at any later point in time to re-verify correct operation.

Test Procedure

1. Press the “Reset” button on the board to begin executing the firmware self-tests
 - **Test1 (Reset Button Test):** Verify that the “RESET” button successfully causes the software to restart
 - *Failure indicates a problem w/the “RESET” button or associated pull-up resistor.*
 - **Test2 (Orange/Green LED Test):** Verify all LEDs are working. The Orange “CAL” Led and the two Green “S1” and “S2” LEDs should turn on briefly after the firmware restarts.
 - *Failure indicates a problem w/one or more of the LEDs or their corresponding current-limiting resistors.*
 - **Test3 (MPU-9250 Sensor Selftest):** Sensor Selftest. **NOTE: The circuit must be still, and it must have the top of the circuit board pointing directly up (away from the earth), in order to pass successfully.** The first time (and only the first time) the board is started after firmware is reloaded, a self-test will run (for approximately 5 seconds). If this succeeds, proceed to Test 8. If this fails, the “CAL” Led will continue to flash quickly, and the selftest will be run again until it passes. If it succeeds, the software will proceed automatically to Test 8 (see below).
 - *There are two possible reasons for failure of the self test:*



- *Communication Failure over I2C bus to the MPU-9250. This case is identified by both green “S1” and “S2” LEDs being off while the orange “CAL” LED is flashing quickly.*
- *Sensor not Still or not Flat – or Sensor Failure. This case is identified by the green “S2” LED being on while the orange “CAL” LED is flashing quickly. Be sure to hold the board still, and be sure the top of the circuit board points directly up (away from the earth). If the self-test still fails after verifying the board is still and flat for several seconds, this indicates a problem w/one or more of the sensors on the MPU-9250.*
- **Test4 (MPU-9250 Calibration):** Inertial Sensor Calibration. The first time the board is started after firmware is reloaded, and after the selftest has successfully passed, the firmware will perform inertial sensor calibration. Inertial sensor calibration executes for approximately 20 seconds. During this time, the sensor must be held still, and should be held flat, and the orange “CAL” LED will flash slowly. Once the calibration is complete, the orange “CAL” LED will turn off.
 - *Failure of this test is due to the board not being held still. Re-run the test and be sure to hold the board still.*
- **Test5 (Normal State):** Once the MPU-9250 Selftest and MPU-9250 Calibration are complete, the Orange calibration LED should be OFF, and the S1 and S2 status LEDs should be on.

VMX-pi LED States

Condition	S1 (Green)	S2 (Green)	3.3V (Green)	Fault (Red)	CAL (Orange)	CAN Status
Startup (1 second)	On	On	On	Off	On	Off
Selftest/Accelerometer Calibration	Off	On	On	Off	Fast Flash	On
Gyro Calibration	On	On	On	Off	Slow Flash	On
Normal	On	On	On	Off	Off	On

Note: If the S1 LED is off during Gyro Calibration or Normal State, this indicates interrupts are not being received from the MPU-9250.

Note: If the S2 LED is off at any time except briefly after Startup, this indicates a problem communicating to the MPU-9250 from the STM32F4 over the internal I2C bus.

Note: If the Fault LED is on at any time, this indicates a short between one of the external power and ground pins.

Advanced USB Serial Protocol

In addition to direct communication with the Raspberry Pi, VMX-pi can simultaneously communicate sensor data to a client (e.g., a RoboRio robot controller) via a custom protocol which communicates over USB Serial connection. This protocol defines messages sent between the VMX-pi and the client over the serial USB interface, and includes an error detection capability to ensure corrupted data is not used by the client.

The VMX-pi Serial protocol uses two message types, the legacy ASCII messages initially introduced in the nav6 sensor, and the modern binary messages introduced in the navX-MXP.

Source code that implements the VMX-pi ASCII and binary protocols in [C/C++](#) and [Java](#) are available online to simplify adding support for the VMX-pi protocol to a software project.

[Message Structure](#)

ASCII Protocol Messages

Each VMX-pi Serial ASCII protocol message has the following structure:

Start of Message	Message ID	Message Body	Message Termination
1 byte	1 byte	length is message-type dependent	4 bytes

Binary Protocol Messages

Each VMX-pi Serial Binary protocol message has the following structure:

Start of Message	Binary Message Indicator	Binary Message Length	Message ID	Message Body	Message Termination
1 byte	1 byte	1 byte	1 byte	length is message-type dependent	4 bytes

[Data Type Encoding \(ASCII\)](#)

Base16 encoding is used for ASCII message elements, as follows:

Data Type	Encoding	Example
Float	(Sign)(100s)(10s)(1s).(10ths)(100ths)	'-132.96'. ' 257.38'

8-bit Integer	(HighNibble)(LowNibble)	'E9'
16-bit Integer	(HighByte,HighNibble)(HighByte,LowNibble)(LowByte,HighNibble)(LowByte,LowNibble)	'1A0F'

Data Type Encoding (Binary)

Binary encoding is used for all Binary message elements. All Binary-formatted data types that are signed are encoded as [2's complement](#). All multi-byte data types are in [little-endian](#) format. Certain non-standard 'packed' data types are used to increase storage efficiency.

Data Type	Range	Byte Count
Unsigned Byte	0 to 255	1
Unsigned Short	0 to 65535	2
Signed Short	-32768 to 32768	2
Signed Hundredths	-327.68 to 327.67	2
Unsigned Hundredths	0.0 to 655.35	2
Signed Thousandths	-32.768 to 32.767	2
Signed Pi Radians	-2 to 2	2
Q16.16	-32768.9999 to 32767.9999	4
Unsigned Long	0 to 4294967295	4

*Unsigned Hundredths: original value * 100 rounded to nearest integer

*Signed Hundredths: original value * 100 rounded to nearest integer

*Signed Thousandths: original value * 1000 rounded to nearest integer

*Signed Pi Radians: original value * 16384 rounded to nearest integer

Start of Message

Each message begins with "start of message" indicator (a '!' character), which indicates that the following bytes contain a message.

Binary Message Indicator

Each binary message includes a "binary message" indicator (a '#' character), which indicates that the following bytes contain a binary message.

Binary Message Length

Each Binary message contains a length value (a value from 0-255), which indicates that the number of bytes which follow in the Message Body and Message Termination.

Message ID

The Message ID indicates the type of message, which may be one of the following:

ID	Message Type	Encoding
'y'	Yaw/Pitch/Roll/Compass Heading Update	ASCII
'g'	Raw Data Update	ASCII
'p'	AHRS + Position Data Update	Binary
'S'	Stream Configuration Command	ASCII
's'	Stream Configuration Response	ASCII
'l'	Integration Control Command	Binary
'j'	Integration Control Response	Binary

[Message Body](#)

The message body differs depending upon the Message Type; the various Message Body specifications are listed below.

[Message Termination](#)

The final four bytes of each Serial protocol message contain a Base16 unsigned 8-bit checksum (encoded in 2 bytes as an ASCII 8-bit integer) followed by a carriage return and then a line feed character.

[Checksum](#)

The checksum is calculated by adding each byte of the message except the bytes within the Message Termination itself. The checksum is accumulated within an 8-bit unsigned byte.

[New Line](#)

The [carriage return](#) (0x10) and [newline](#) characters (0x13) are present at the end of the message so that when the message is displayed in a console window, a new line will be inserted in the console at the end of the message.

[Message Body Definitions](#)

Yaw/Pitch/Roll/Compass Heading Update Message

The Yaw/Pitch/Roll/Compass Heading Update message indicates the VMX-pi current orientation and heading, in units of degrees, as follows:

Byte Offset	Element	Data Type	Unit
0	Yaw	Float	Degrees (-180 to 180)
7	Pitch	Float	Degrees (-180 to 180)
14	Roll	Float	Degrees (-180 to 180)
21	Compass Heading	Float	Degrees (0 to 360)

Raw Data Update Message

The Raw Data update message communicates the raw gyro, accelerometer, magnetometer and temperature data. This data bypasses the Digital Motion Processor, and allows the individual sensors to be used directly without any intervening processing. This can allow the following types of use:

- Access to instantaneous measures of angular velocity in each of the X, Y and Z axes, provided by the tri-axial gyroscopes. Note that the accelerometer and gyroscope data has already had bias calibration applied.
- Additionally, raw magnetometer data is provided. Note that the raw magnetometer data may have already had soft/hard iron calibration applied, if the VMX-pi magnetometer calibration procedure has already been completed.

Byte Offset	Element	Data Type
0	Gyro X (15-bits, signed)	16-bit Integer
4	Gyro Y (15-bits, signed)	16-bit Integer
8	Gyro Z (15-bits, signed)	16-bit Integer
12	Acceleration X (16-bits, signed)	16-bit Integer
16	Acceleration Y (16-bits, signed)	16-bit Integer
20	Acceleration Z (16-bits, signed)	16-bit Integer
24	Magnetometer X (12 bits, signed)	16-bit Integer
28	Magnetometer Y (12 bits, signed)	16-bit Integer
32	Magnetometer Z (12 bits, signed)	16-bit Integer
36	Temperature (Centigrade degrees)	Float

Gyro Device Units: value in deg/sec * gyro full scale range

Accelerometer Device Units: value in G * accelerometer full scale range

Magnetometer Device Units: value in uTesla * .15

AHRS / Position Data Update

Byte Offset	Element	Data Type	Unit
-------------	---------	-----------	------

0	Yaw	Signed Hundredths Degrees	
2	Pitch	Signed Hundredths Degrees	
4	Roll	Signed Hundredths Degrees	
6	Compass Heading	Unsigned Hundredths Degrees	
8	Altitude	Signed 16:16	Meters
12	Fused Heading	Unsigned Hundredths Degrees	
14	Linear Accel X	Signed Thousandths	G
16	Linear Accel Y	Signed Thousandths	G
18	Linear Accel Z	Signed Thousandths	G
20	Velocity X	Signed 16:16	Meters/Sec
24	Velocity Y	Signed 16:16	Meters/Sec
28	Velocity Z	Signed 16:16	Meters/Sec
32	Displacement X	Signed 16:16	Meters
36	Displacement Y	Signed 16:16	Meters
40	Displacement Z	Signed 16:16	Meters
44	Quaternion W	Signed Pi Radians	Pi Radians
46	Quaternion X	Signed Pi Radians	Pi Radians
48	Quaternion Y	Signed Pi Radians	Pi Radians
50	Quaternion Z	Signed Pi Radians	Pi Radians
52	MPU Temp	Signed Hundredths	Centigrade
54	Op. Status	UInt8	_NAVX_OP_STATUS
55	Sensor Status	UInt8	_NAVX_SENSOR_STATUS
56	Cal. Status	UInt8	_NAVX_CAL_STATUS
57	Selftest Status	UInt8	_NAVX_SELFTEST_STATUS

[Stream Configuration Command](#)

By default, VMX-pi begins transmitting YPR Updates upon power up. The Stream Configuration Command is sent in order to change the type of VMX-pi Streaming Update transmitted to the client.

Byte Offset	Element	Data Type
0	Stream Type	8-bit ASCII Character
1	Update Rate (Hz) – Valid range: 4-60	8-bit Integer
Stream Type	Description	
'y'	Yaw, Pitch, Roll & Compass Heading Update	
'g'	Gyro (Raw) Data Update	
'p'	AHRS + Position Data Update	

[Stream Configuration Response](#)

Whenever a Stream Configuration Command is received, VMX-pi responds by sending a Stream Configuration Response message, which is formatted as follows:

Byte Offset	Element	Data Type
0	Stream Type	8-bit ASCII Character
1	Gyroscope Full Scale Range (Degrees/sec)	16-bit Integer
5	Accelerometer Full Scale Range (G)	16-bit Integer
9	Update Rate (Hz)	16-bit Integer
13	Calibrated Yaw Offset (Degrees)	Float
20	Reserved	16-bit Integer
24	Reserved	16-bit Integer
28	Reserved	16-bit Integer
32	Reserved	16-bit Integer
36	Flags	16-bit Integer
Flag value		Description
0, 1		Startup Gyro Calibration in progress
2		Startup Gyro Calibration complete

[Integration Control Command](#)

The Integration Control Command is sent in order to cause certain values being integrated on the VMX-pi to be reset to 0.

Byte Offset	Element	Data Type
0	Action	UInt8 (NAVX INTEGRATION_CTL)
1	Parameter	UInt32

Integration Control Response

The Integration Control Response is sent in response to an Integration Control Command, confirming that certain values being integrated on the VMX-pi have been reset to 0.

Byte Offset	Element	Data Type
0	Action	UInt8 (NAVX INTEGRATION_CTL)
1	Parameter	UInt32

navXUI Customization

The navXUI Source Code is Open-Source and can be customized using the following

instructions:

- Download and install the free . NOTE: the current navXUI code is compatible with version 3.0beta5 of the Processing development environment.
- Checkout the [navXUI source code](#).
- Copy the contents of the navX source code's 'processing' directory to <User Directory>\Processing directory.
- Open the Processing IDE and then open the navXUI sketch via the File->Sketchbook menu.
- Compile/Run the navXUI by selecting the Sketch->Run menu.

If your computer has more than one serial port, you will need to select the appropriate serial port (corresponding to the USB serial port the navX-Micro is connected to) from the COM port selection drop-down list in the top-right of the navXUI display.

Technical References

The references on this page are provided to help students gain a deeper understanding of the algorithms, technologies and tools used within the VMX-pi and other Inertial Measurement Unit (IMU) and Attitude/Heading Reference System (AHRS) products. Additionally, links to other notable open-source works which could perhaps be adapted to work on the VMX-pi are included.

[Algorithms](#)

[Technology](#)