New for 2017

Significant Changes

There have been a number of significant changes to WPILib for the 2017 FRC season. This list highlights the most important changes but you should look in the rest of this section for details on these and other updates.

- Most of the 3rd party devices have been unbundled from the core WPILib. There is now a standard procedure for adding support for these devices back into WPILib. As a result the **CANTalon support has moved from WPILib to the** CTRE web site and released by CTRE. To use the CANTalon you must download the installer from the CTRE web site. Similarly **CANJaguar has moved from the library to a separate package**.
- We have moved away from the NIVision libraries in favor of **OpenCV as a fully integrated solution**. There is now significant additions to the WPILib Suite to make it much easier and constant to use vision in your robot programs.
- **GRIP**, the graphical program generator for OpenCV now can either run on the Driver Station or it **can generate code in C++, Java or Python** for incorporation into your robot programs. We no longer recommend deploying GRIP into the roboRIO or other low resource processors.
- **WPILib for C++ and Java are now Open Source projects** collectively called WPILib Suite and have moved to GitHub. You can view, download or clone any of the repositories in the WPILib Suite of repositories.

Computer vision and camera support

For 2017 the most significant features added to WPILib Suite have been in the area of computer vision. First and formost, we have moved from the NIVision libraries to OpenCV. OpenCV is an open source computer vision library widely used through academia and industry. It is available in many languages, we spefically support C++, Java and Python. There is a tremendous wealth of documentation, videos, tutorials, and books on using OpenCV in a wide ranging set of applications with much emphasis on robotics.

- OpenCV libraries are now bundled with WPILib and will be downloaded to the roboRIO without the need for teams to locate and download it themselves.
- There is complete support for USB and Axis cameras in the form of a CameraServer class and specific camera classes that will produce OpenCV images that can be used for further processing. You can either let the CameraServer automatically stream camerea video to the SmartDashboard or you can add processing steps on the robot between capture and

sending to the Dashboard. All the example programs in eclipse have been updated to show how the new Camera server is used.
- GRIP, the graphical vision pipeline generator can be used to quickly and easily create and test computer vision algorithms that can run standalone on your Driver Station computer sending results back to the robot via NetworkTables. New for 2017, GRIP can generate code in either C++, Java or Python for your vision algorithm that can easily be incorporated into robot programs.
- The NIVision libraries have been removed from WPILib to a separately installable package.

All WPILib languages

- New commands were added to reduce boilerplate code. TimedCommand finishes after a timeout. InstantCommand executes once then finishes.
- DriverStation::WaitForData() is safer and handles spurious wakeups. However, the function is now only safe for a single thread to call. Additional threads calling it will change the behavior and the threads will most likely not get called correctly.
- The DigitalSource Interface has changed.
- AnalogTriggers now allocate the AnalogInput they are using. If you need multiple AnalogTriggers on one AnalogInput, you must use the constructor with an AnalogInput reference parameter.
- All instances of floats in the user interface have been replaced with doubles since it's more consistent and there's no measured performance impact.
- The unimplemented function I2C::Broadcast() has been removed.
- The Joystick hierarchy was redesigned and an Xbox controller class was added. Added getPort to Joystick class.
- New enums (kUSB1 and kUSB2) have been added to SerialPort. kUSB aliases to kUSB1. When only one USB serial device is connected, kUSB1 will connect to it, and properly reconnect on open. If 2 USB devices are connected, kUSB1 will be the top USB port on the RoboRIO (closest to the edge) and kUSB2 will be the bottom USB port.
- Fixed Digital output PWM on MXP ports

C++ specific changes

- The SpeedController abstract base class Set() function now only takes one parameter (the optional syncGroup parameter has been removed). The only class which used syncGroup, CANJaguar, now has both one-parameter and two-parameter Set() functions. Any custom classes you derived from SpeedController should be updated.
- Definition of REAL has been removed from wpilib.h.
- The Task class has been deprecated. Use std::thread instead, which provides the same functionality. The new Threads.h header provides functions for setting thread priority.

- All classes have been moved into an frc namespace. There is a compatibility shim that has been added to make this a non-breaking change, which is planned to be removed for 2018.
- The Semaphore class at HAL/cpp/Semaphore.h has been deprecated. Replace with a std::mutex and a std::condition_variable combination.
- All unsigned integers in the user interface were replaced with signed integers.
- SerialPort::Write(string&, int) has been deprecated, and is heavily suggested to not be used anymore. Instead use the 2 new overloads (const char*, int) or (StringRef). The StringRef overload will take a std::string directly, with no need for a length parameter. Construct a StringRef with a custom length in order to pass 0's in the buffer.
- delayTicks(), delayMillis(), delaySeconds(), HAL_NO_WAIT, HAL_WAIT_FOREVER, niTimestamp32(), and niTimestamp64() were removed in favor of std::chrono.
- DriverStation::IsSysBrownedOut() was renamed to DriverStation::IsBrownedOut() to match Java.
- The syntax for using the SendableChooser has changed somewhat. See the updated examples at the bottom of this page: http://wpilib.screenstepslive.com/s/4485/m/26401/l/255419-choosing-an-autonomous-program-from-smartdashboard?id=255419-choosing-an-autonomous-program-from-smartdashboard

Java specific changes

- The SpeedController interface set() function now only takes one parameter; the overload which provided a syncGroup parameter has been removed. The only class which used syncGroup, CANJaguar, still implements both one-parameter and two-parameter set() functions. Any custom subclasses of SpeedController must be updated.
- DriverStation.waitForData(timeout) now takes a value in seconds for a timeout rather than milliseconds.
- Classes containing a series of static constants were replaced with Java enums.

Hardware Abstraction Layer (HAL)

The Hardware Abstraction Layer of WPILib is the low-level code that interfaces directly with the hardware or the external APIs in the roboRIO. It is written in C++ and shared between the user-facing C++ and Java libraries. The HAL API has seen significant changes this year. Since the HAL is not considered user-facing, these changes have been deemed to not affect teams. If you use the HAL, most functionality still exists, with a few changes that cannot be replicated.

- All HAL functions have been prepended HAL_ to clean up the exported function calls.
- The HAL now uses handles instead of opaque pointers for passing around variables. This increases type safety, and allows for cleaner errors when an invalid variable is passed to the HAL.

- An extension to this is that all resource counting has been moved to the HAL. This cleans up the WPILib classes a lot, and allows future HALs to change parameters without needing to modify the WPILib upstream.
- Port and other constants have been moved to the HAL. This enables the WPILib to get information on the hardware from the HAL directly, which will make future HALs and hardware easier to implement.
- The analog gyro class has been moved to the HAL. This change enables easier simulation support in the future and reduces code duplication between C++ and Java.
- The Encoder WPILib class has been moved down to the HAL. The HAL now detects a 4x encoder vs a 2x or 1x encoder and properly selects the FPGA class to use.
- PWM bounds math is now performed at the HAL level. This change enables adding easier simulation support in the future and makes the HAL intake real world values.
- Joysticks now return -1 to 1 from the HAL, which again is a change to real world values.
- Waiting for new DriverStation data now happens at the HAL level, which will make it easier in the future to handle the DriverStation data.
- The DigitalSource API has been changed upstream to make the HAL code for encoders, counters and interrupts cleaner and easier to write and understand.

SmartDashboard

- New widget for viewing video streams including frame rate from the CameraServer object in robot programs.
- Improved connection roboustness.

GRIP

The most significant change to GRIP is the addition of **code generation for C++, Java, and Python**. After developing your pipeline using the interactive user interface and everything is working you can now generate code to implement that pipeline using OpenCV. In the past the only way to get the GRIP pipeline to run on a roboRIO or small co-processor was to deploy a headless version of GRIP to that device. This proved to be difficult because of the resources required by GRIP and limited resources on the processor. Now you can generate a class that implements the pipeline and call it from your robot program. See the ScreenSteps documentation for GRIP.

For the full list of updates to GRIP view the release notes for recent releases.

We no longer recommend using the GRIP deploy tool for roboRIO or Raspberry PI processors due to issues seen by many teams running out of resources.

Network Tables

- DeleteAll now no longer deletes persistent variables, but only deletes non-persistent variables.
- ConnectionInfo::remote_name has been changed to remote_ip to match its actual contents.

RobotBuilder

- Added support for new commands: TimedCommand, InstantCommand, and ConditionalCommand
- Allow command extensions to use a custom base class

WPILib Suite open source project

The WPILib project and it's associated projects have moved to a new organization on GitHub called WPILib Suite.

All the source code, issue reporting, and development activities associated with the C++ and Java languge support for FRC are located in that organization. You are free and encouraged to browse, clone, or just download the source code from there. If you have any issues using the WPILib Suite please file an issue against the appropriate repository.

We welcome contributions to WPILib Suite, with anything from bug fixes to major improvements. But before starting a project please review the contributing instructions before starting something that you would like to see merged into the suite.

We are in the process of moving as much of the documentation as we can to ScreenSteps so if you are just looking for instructions on using WPILib Suite look there.