WPILib Long Term Roadmap

Updated for 2024

WPILib Mission

- Enable *FIRST* teams to focus on writing game-specific software rather than on hardware details
- Make new robotics technologies more approachable to teams: "raise the floor, don't lower the ceiling"
    - Enable teams with limited programming knowledge or mentor experience to be more successful
    - Enable teams with intermediate programming knowledge to use powerful tools to improve their robot performance
    - Enable teams with advanced programming knowledge to use the full power of the system
- Support the Kit of Parts control system hardware (e.g. controllers, sensors)
- Provide parity across all officially supported languages
    - Enable teams to pick the language of their choice without worrying about supported features
- To this end, the library and associated tools need to be robust, reliable, maintainable, and understandable!

FRC Control System: Who Is Responsible for What?

WPILib Team (Volunteers)
C++, Java, Python libraries
NetworkTables, CameraServer
SmartDashboard, Shuffleboard
Simulation GUI, infrastructure
OutlineViewer, Glass
RobotBuilder
AdvantageScope
C++, Java, Python build + deploy infrastructure
Romi and XRP compatibility

National Instruments
roboRIO image / libraries
FRC NetComm on roboRIO
LabVIEW libraries
FRC Driver Station
LabVIEW Dashboard
FPGA code and interface library
roboRIO Imaging Tool

Vendors
Vendor libraries

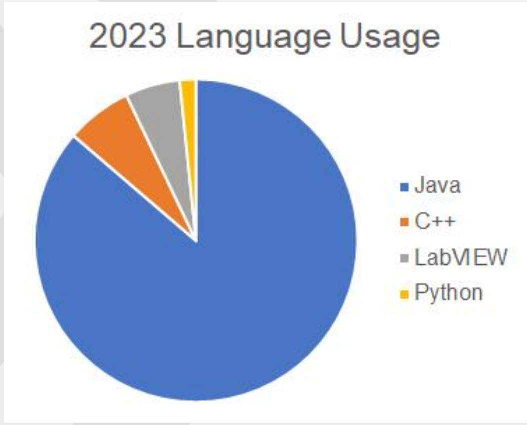
FIRST
Field Mgmt System (FMS)
DS-Robot-FMS interfaces
Field Network Configuration
Robot Radio
Robot and Game Rules

WPILib History (notable events)

- Pre-2009: C language only (one of several library/template options)
- 2009: NI cRIO; WPILib C++ becomes an “official” library/language
- 2010: Java added (Java 6 ME); community RobotPy project starts (integrated in 2024)
- 2015: RoboRIO; Java 8; C++11; move to GitHub; volunteer contributions 🚀
- 2016: NetworkTables 3
- 2017: CameraServer
- 2018: Shuffleboard; Timed Robot
- 2019: VS Code; GradleRIO (integration); Java 11; C++17; Pi image; PathWeaver; installer
- 2020: ReadTheDocs; Sim GUI; Commands v2; Geometry+Trajectory
- 2021: Glass; Romi
- 2022: SysId; DataLog; first long-term roadmap
- 2023: NT4; Java 17; C++20
- 2024: AdvantageScope (integration); XRP; Python (integration); Java units; Struct/Protobuf

Official Language Options Today

- 4 officially supported languages: C++, Java, Python, and LabVIEW
    - Roughly equivalent feature sets
- Primary driver of language selection is network effect
    - School classes (AP CS Java)
    - Mentor experience
    - What other teams in your area use
    - Online support base (teams worldwide)
    - What other teams at competitions use
- In 2023, 86% of FRC teams used Java
    - Python was unofficial in 2023; expecting to see future growth as an officially supported language



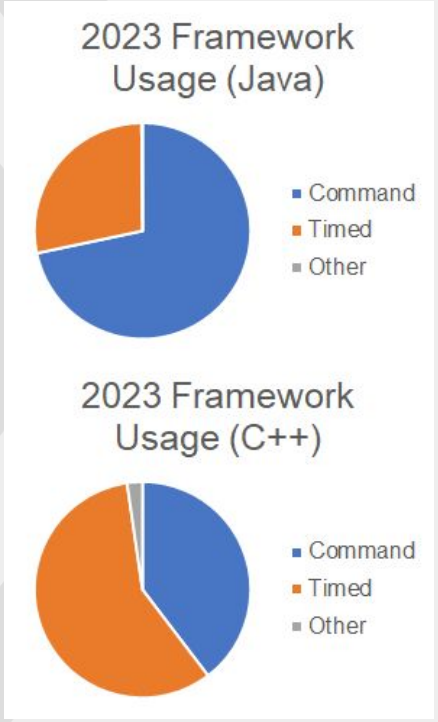
WPI Lib

WPILib Frameworks Today

WPILib offers two primary frameworks

- Timed Robot
  - Runs a function (based on mode) every 20 ms
  - Easy to get started with, but unwieldy for complex programs
- Command Based
  - Built on top of Timed Robot
  - Genesis was team approaches to adding structure
  - Structures code into “subsystems” and “commands”
  - Execution model is cooperative multitasking
  - Enables more complex programs, but has steeper learning curve

In 2023, 62% of *all* FRC teams used command-based Java for their competition robot programs



Overall WPILib Direction

- Continue open source, volunteer community development
- Support 2027 MRC
- Add support for FTC users, including on-bot development environment
- Help teams discover WPILib capabilities and be more successful with them
- Embrace essential community tools and integrate/support them where possible
- Prioritize “quality of life” improvements over new features
- Enhance debugging/diagnosis tools
- Add problem-focused documentation
- Improve integration of smart motor controllers and coprocessors
- Provide tools to enable enhanced robot autonomy
- Create tools and documentation demonstrating best practices for end-to-end workflow

Guide to These Charts

- [Community] denotes community-provided tools/software not (currently) bundled with, supported, or documented by WPILib
- [Integrated] denotes community-originated tools/software that WPILib now bundles and documents
- ✅ denotes 2022 long-term roadmap items available today
- 👥 denotes where we would love more help, particularly people who would be willing to maintain for the long term / build a community of maintainers
  - In general, we love more help, even in small ways (bug reports, small bug fixes, etc), but these are more major projects. WPILib exists in the form it does today due to many contributions from the community–thank you!

Documentation

2022 and Today:

- Open source (read the docs)
- Translations to multiple languages
- Lots of solution space documentation

Future:

- FTC focused docs, MRC updates
- Problem-focused documentation 👥
- Improved integration with vendor docs
- Docs overall reorganization

Maybe?

- Click-to-run (open & run example project from docs link) 👥

WPI Lib

Robot Languages

2022:

- Java 11
- C++17 (with GCC 8 for Rio)
- [Community] Python (RobotPy)

Today:

- Java 17 ✅
- C++20 (with GCC 12 for roboRIO) ✅
- [Integrated] Python (RobotPy) ✅

Future:

- Java 21+ (more closely track future evolution)
- C++2x (more closely track future evolution)
- Python (RobotPy) continued increased integration and support

Maybe?

- C# (no earlier than 2027, would likely look for unofficial uptake first, similar to RobotPy)
- C++ modules support (no earlier than 2027)

Telemetry and Data Logging and Analysis

2022 and Today:

- NetworkTables, LiveWindow (telemetry), SmartDashboard+Shuffleboard classes
- DataLog+DataLogTool (late 2022), Glass real-time analysis

Today:

- Pub/sub (NT4) ✅
- LiveWindow telemetry: disabled by default ✅
- [Community] Annotation-based telemetry (Monologue) ✅
- [Integrated] Unified offline data analysis tool (AdvantageScope) ✅

Future:

- [Integrated] Annotation-based telemetry
- Remove LiveWindow telemetry
- HAL telemetry

Maybe?

- NetComm telemetry
- ROS2 interoperability

Driver Dashboards

2022 and Today:

- 2 common "driver" dashboards (SmartDashboard, Shuffleboard)👥
  - Venn diagram of feature sets

Today:

- [Community] Web-tech Shuffleboard equivalent (Elastic) ✅
- [Community] Web-tech dashboard components (FRC Web Components)

Future:

- SmartDashboard retirement
- [Integrated] Web-tech Shuffleboard equivalent 👥
- [Integrated] Web-tech dashboard components 👥

Team Code Structure

2022 and Today:

- Command Based v2, TimedRobot

Today:

- Mature Command Based v2 “fluent” (chained decorator methods) tutorials and examples
- [Community] Start commands based on location while path following (PathPlanner) ✅
- [Community] AdvantageKit
- Command Based “quality of life” improvements (e.g. named joystick buttons that are more natural to hook to commands) ✅

Future:

- [Integrated] Start commands based on location while path following 👥
- Trigger-based autonomous structure (tutorials and examples) 👥

Maybe?

- Coroutines for autonomous sequences
- [Integrated] AdvantageKit style structure option

Cameras, Image Processing, Coprocessors

2022 and Today:

- CameraServer library: USB camera access, MJPEG streaming
- WPILibPi: Raspberry Pi image w/included streaming and vision processing examples

Today:

- Integrated support for AprilTag markers ✅
- [Community] PhotonVision

Future:

- Lower latency video streaming (WebRTC) 👥
- More efficient codecs (h.264 or similar, currently limited by licensing and platform capability) 👥
- Web-based persistent camera configuration 👥
- Improved picam support 👥
- Standalone app for performant camera stream viewing

Maybe?

- Support for RealSense or other 3D cameras
- ML

Vendor Integration

2022 and Today:

- Vendor deps - standalone installers/downloads
  - WPILib examples don't use any smart motor controller features

Future:

- Centralized catalog of vendor deps in IDE (for discovery purposes)
- Integrated downloader/installer, including vendors and community tools
- Linux driver model for open source vendor libs
  - Built/distributed by WPILib, maintained/supported by vendors

Maybe?

- Vendor-agnostic wrappers/interfaces for smart motor controller most-used functionality (e.g. integrated PID controls)

Training/Education

2022 and Today:

- Simulation, Romi
- RobotBuilder

Today:

- XRP

Future:

- RobotBuilder v2.0
- Blockly-style builder

Maybe?

- Tutorial-based VS Code plugin akin to Jupyter Notebook

IDE & Build System

2022 and Today:

- Visual Studio Code & GradleRIO

Today:

- WPILib installer for Raspberry Pi ✅

Future:

- On-robot development w/ web environment (Chromebook-friendly) 👥
- IDE support for deploying coprocessor programs (at least to WPILibPi) 👥
- Use industry standard build tools for each language (e.g. CMake for C++) 👥
- Create build-system-agnostic deployment tool

Maybe?

- IntelliJ
- Bazel

Simulation

2022 and Today:

- Sim support integrated w/ SimGUI frontend

Today:

- [Integrated] 3D environment and more viz tools (AdvantageScope) ✅

Future:

- Integrate simulation into team development workflow (sim first?) 👥
- More actuator models 👥
- Vendor-specific simulation docs
- Simulation code included in all examples 👥
- Improved simulation support in vendor libraries
- Swerve simulation 👥

Maybe?

Drivetrain Trajectory Planning Tool

2022:

- Nonholonomic trajectory support (PathWeaver)
- JSON import of PathWeaver files

Today:

- [Community] Holonomic (e.g. swerve) support (PathPlanner) ✅
- [Community] Constraints and path command events (PathPlanner) ✅
- [Community] Trajectory optimization (Choreo) ✅

Future:

- [Integrated] Swerve support, constraints, path command events
- [Integrated] Trajectory optimization

Maybe?

- Obstacle avoidance

WPI Lib

Controls: Localization and State Estimation

2022:

- Encoder/gyro odometry fused with computer vision pose measurements
- State estimation with Kalman filters (KF, EKF, UKF)

Today:

- Improved numerical stability of UKF with square-root form ✅
- 3D geometry classes ✅
- [Community] AprilTag pose estimation (PhotonVision) ✅

Future:

- Improved pose estimation accuracy with GTSAM (see #5669)

Controls: Controllers

2022:

- Arm, elevator, and flywheel feedforward controllers
- PID and LQR feedback controllers
- Ramsete unicycle feedback controller

Today:

- Differential drive feedforward controller ✅
- Linear time-varying unicycle and differential drive feedback controllers ✅

Future:

- Remove PIDSubsystem, PIDCommand, RamseteCommand

WPI Lib

Controls: Motion Profiles

2022:

- Trapezoid profile

Today:

- Exponential profile ✅

Future:

- Replace ProfiledPIDController with user-composed motion profile and PID

Maybe?

- Spline-based trajectories for multi-DOF mechanisms (e.g., 2-DOF arms)

Controls: System Modeling

2022:

- Linear system identification with SysId

Today:

- Improved stability of SysId via user-side logging ✅

Future:

- SysId ease of use (e.g. clarifying units, links to docs)