Future of EmWiz

EmWiz current state

Component	Dependencies
FastScan	C++&Qt4, DOOCS, TINE
EmCalc	C++&Qt4
RootPlot	C++ & QtROOT, ROOT
MemoryWatcher	C++

Component	Name	End of life
Operating system	SL 7	June 2024
Programming language	C++	-
API	Qt4	2015
API	DOOCS	-
API	TINE	2030 (successor: DOOCS)
API	QtROOT	2015
API	ROOT	-

EmWiz possible upgrades

Component	Name	End of life	Name	End of life
Operating system	SL 7	June 2024	AlmaLinux 9	2032
Programming language	C++	-	C++ & Qt6, Matlab(+ C++lib?), Python	-
API	Qt4	2015	Qt6, Matlab, Python	-
API	DOOCS	-	DOOCS	-
API	TINE	2030	DOOCS	-
API	QtROOT	2015	QtCharts, Matlab, Python	-

EmWiz current state

Component	Implementation	Estimated time (100% load)
FastScan	C++ & Qt6	2 months
	Matlab (+ C++lib)	2 months
	Python	? months
EmCalc + RootPlot	C++ & Qt6 & QtCharts	4 months
	Matlab (+ C++lib)	2 months
	Python	? months
MemoryWatcher	C++	1 week

Pros and Cons (Qt, Matlab, Python):

Matlab: everybody can do - easier to add new/modify things; Qt and Python: only few people can do

Qt & Python: native multithreading (must for fastscan); Matlab: multithreading is tricky

Current EmWiz implementation: hard wo maintain (code quality) -> needs refactoring in any case

Cleanup code and repository

Implement replacement classes for QtRoot.

Migrate from Qt4 to Qt6

Code refactoring to make it more maintainable. -> Not urgent can be done step by step

- There is a lot of standard implemented stuff (part of them are in standard recently) that should be removed from the code
- A better style for outsourcing
- Try to get rid of parallel working applications or separate them at all