I’ve been building computer controlled things for more than 35 years. In any large project you always get to the point where strange things are happening and you don’t understand why its doing what its doing. In general my solution to this is to record everything, all sensor inputs, all drive outputs, every thing. While in concept this sound easy, it isn’t. You may have a sensor providing data at 1000Hz and the result of calculation being done every 5 seconds. You may also have hundreds of inputs and outputs with only a small number of them of interest for any one problem. In the past I’ve solved this problem by creating structures to hold logged data, writing them to a memory pool or file record and then writing special tools to decode them.  
  
This leads to several problems:

* Both the encoder and decoder code need to be modified when you change what you are logging.
* If a structure changed (say added a field) its hard to compare results from old/new records together.
* Its real easy to make a clerical error in either the encoding or decoding function.
* When you go back to old files its hard to know what format they are in.

I’ve recreated this code in various forms more times than I could count.  
If C++ supported proper class introspection one could probably solve this problem in a robust way where the logging would record the structure name and data layout as part of saving it. Unfortunately C++ does not presently support introspection. (Tools like Boost do, but that hammer is a bit big fo4r a small embedded device)  
  
The newest C++ standards do however support inline constructor arguments for C++ class/structure member variables, and these can be mutated into a pretty good approximation of c++ class introspection.  
  
For this years SparkFun AVC the very first piece of code I wrote was a robust logging infrastructure and the tools necessary to encode/decode, display it.,

This set of tools allows:

* Logging of arbitrary fixed size data structures .
* Looging different records at different rates
* Automatically stores the date/time revision for every source file into the data record at startup.
* Automatically stores the logged structures names and layout information into the data record.
* Stores the layout stream once and only once the first time any structure is used
* This information is communicated in the data record to the decoding tools.
* Mutiple decoding tools convert the Binary efficient record into:
  + JSON file.
  + CSV file with any easily specified subset of the data fields specified.
  + In Javascrip array of object inside a browser.