I said I was going to organize this, but that definitely isn't going to happen. This is a gigantic list of random things I wish we had not learned the hard way. Some elements may be nonsensical, some already known, and some not applicable any more. Important ones have been highlighted.

-Brian Truong

Assume the worst.

No seriously, have solutions for when every single piece of hardware fails.

Team Management

* Work harder to involve everyone in development
* Only members with owner status can freely add other members to any team
* Give people responsibility and make sure to tell them when they do something wrong, but in a pleasant way. Don’t just fix it without their knowledge.
* Delegate and then don't leave. Check up on them and help them.
* Talk to other teams
* Use projector to show everything you are doing

Coding Style / Practices

* Peer review system
* Standardize camel case or underscores
* Write obnoxious comments to “set as default”
* Make typedefs in a VI first, then save as file and use elsewhere
* Rewriting things can help you learn, but don't ever lose the most stable version of code
* Make a binder describing programming design
* Actually organize things
  + Comments too

LabVIEW

* Reentrant vi's can't be probed
* Immediately start testing framework on the robot
* Use linked tunnels to auto-connect terminals in case structure
* When making subvi, make a wire terminal required to avoid unexpected events
* ctrl click and drag to make space in that area
* Make a comment and delete it to force Labview to allow you to resave
  + This lets you resave the window's position and size and view port
  + Useful for repositioning dashboard window
* Ctrl space for labview search
* Right click files that have been lost in project explorer
  + Then click "replace" and find the replacement file.
* Right click on subvi and click “relink to subvi” when subvi is deleted
* Mass compile is a thing in Labview
* Update Labview shortcuts sheet sometime
* Labview profiling
  + Tools->Profile->Performance and Memory
* Check begin.vi carefully for references that match
* Experiment with reading errors from Begin.vi
* Explain error thingy
  + Apparently there is some Labview feature to explain errors somewhere? I don’t remember anymore.
* Timers are multi thread so if iteration takes longer than wait time, will immediately move to next iteration
* Consider using conditional disable
  + project->properties->conditional disable
  + Can quickly switch between using different types of sensors, etc
* ctrl-= and ctrl-(minus) to inc/dec font size
* Use sequence frames for timing critical things
* Always consider the bigger picture with your code
  + Just because it does what it should, doesn't mean it will be effective for its purpose. You should try and consider the physical output of your code and whether it considers what the mechanical people need.
* When renaming projects:
  + Be sure to rename the build project thing too. Also check all code for dependency issues.

Hardware / Drive Train

* Make changes to driving based on 1st hand experience not what driver says.
  + Maintain past versions though.
* For counters and encoders, open dio first
* Acceleration limiter based on motor speed
* PID tuning as soon as possible
* Don't change values too quickly
  + Just because it doesn't work a few times doesn't mean it isn't the best values possible. The fact of the matter is that the hardware just won't be that accurate. Do testing before arbitrarily changing values.
* Tune PIDs early
  + Annoy hardware people
* How to make a subvi for tuning PIDs
  + Create chart for speed
  + Display speed
  + Display motor value
  + Able to precisely control set point
  + Able to control PID constants
* Use controls for anything that could possibly be changed
* Try not to use coupled limit switches

Deploying / Router

* Modules have to be in right order
  + Analog, digital, solenoid (digital out)
  + Starting with lan port
* Use port 1 for cRio
* Make a quick sheet
  + Would have things like ip addresses, electrical ports. Printed out for easy access.
  + Standardize ip addresses
* Add flashing to wiki
* Waiting for cRio only indicates problem if it shows twice
* Avoid deployment as much as possible
  + Use dashboard whenever possible

Github / Git

* Github for windows now automatically rebases when it syncs
  + There is an option for it that seems to default to true. There is a small chance this will result in problems, but I don't know whether it is worth telling everyone to turn it off since chances are we will forget.
* Test relying on Git for built robot code
  + Git is faster at loading built code from a commit than Labview at building code
* Make the wiki freakin awesome
* Use double tilde in Github markdown to strikeout text

Dashboard / At Competitions / Other

* Control use of sensors through dashboard
* When code is disabled, inputs from a controller make that controller light up on the driver station
* actually try and use the simulator
* In the pit, take shifts for lunch
* Know autonomous timing in detail
* Use user1 led to display status
* Log viewer for frc
* Have wifi batch script say when it fails
* Make a scouting app
* f1 to reconnect joystick in fms mode
* F1 apparently also enables robot in driver station
  + Don’t press when safety judge walks by
* Use only the two usb ports we always use
  + Otherwise results in various issues with switching controllers
* Know specifics of auto timing for alliance members
  + Only for buiding robot code (assuming you deploy it with that computer)
  + Doesn't seem to work with dashboard (you should build that for the particular laptop)