*Event-Driven Setup*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam non lacus malesuada, dignissim mi ac, fringilla mi. Aliquam vestibulum metus commodo urna blandit congue. Fusce ac dolor viverra, semper eros eu, ornare ante. Nulla dignissim, ex eu vehicula viverra, nunc ligula faucibus arcu, sit amet mattis urna augue vitae mauris. Phasellus placerat, ligula non aliquet scelerisque, sapien metus interdum erat, ut suscipit urna libero consequat risus. Donec libero ipsum, volutpat sit amet enim et, aliquet luctus orci.



*Advantages*

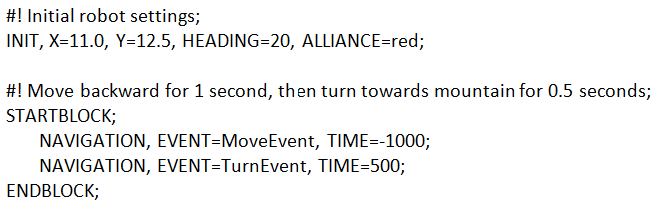
* Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam non lacus malesuada, dignissim mi ac, fringilla mi.
* Lorem ipsum dolor sit amet, consectetur adipiscing elit.

*Disadvantages*

* Lorem ipsum dolor sit amet, consectetur adipiscing elit.
* Phasellus placerat, ligula non aliquet scelerisque, sapien metus interdum erat, ut suscipit urna libero consequat risus. Donec libero ipsum, volutpat sit amet enim et, aliquet luctus orci.

*Autonomous Event Configurations*

Our autonomous event configuration system (or “event configs” for short) is a mini-programming language developed by us. It allows us to easily create, modify, and manage many different autonomous variations, with as little change to our main code as possible. Event configs, like the example below, are set up to work with how our event-driven system works. For more info, see the “Event Config” section under the “Misc. Documentation” tab.



*Advantages*

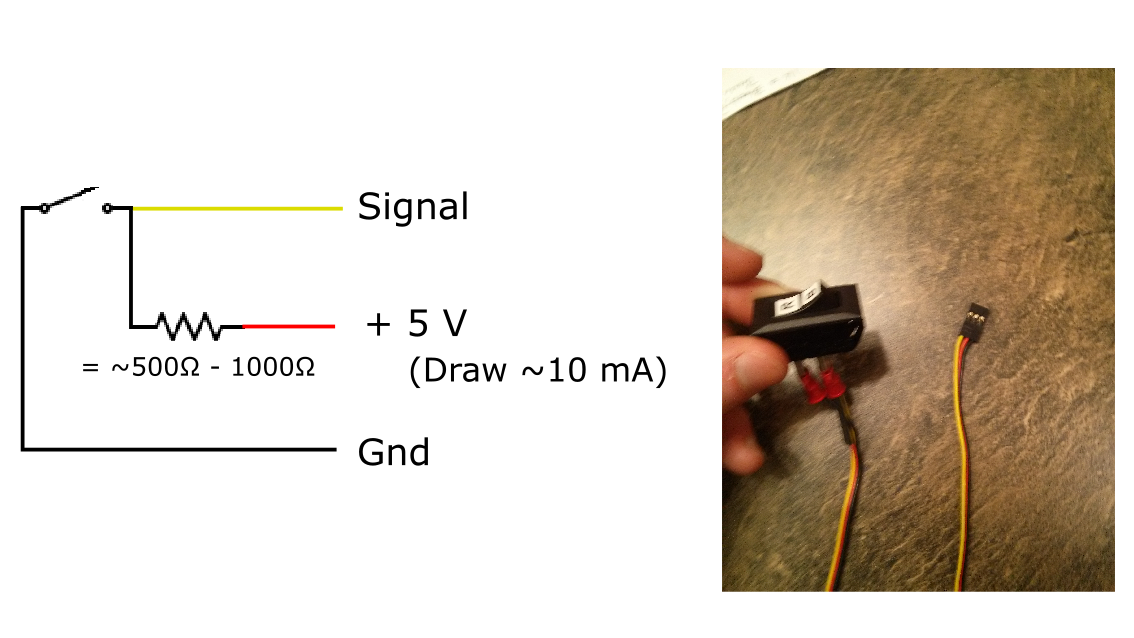
* Makes setting up, modifying, and managing many different autonomous variations significantly easier & quicker (only a few additional lines in the actual code for each event config)
* Minimal programming knowledge is needed to create an event config

*Disadvantages*

* A system to parse/translate the event config text into events in the code needed to be programmed
* A new type of event added to the navigation or action system will also require the addition of code to the event config parser/translator to be able to fully use the new event

*Custom Alliance Switch*

We created a switch that allows us to tell the robot which alliance we are on in each match. This is important to the autonomous code, because it allows us to create a new autonomous variation without having to create 2 versions for the red and blue alliance sides of the field. We had to wire this switch up to work correctly and safely with the robot’s device interface module. The robot reads this switch only once and stores its state, just in case the switch is hit during autonomous. The parts of the code that vary for each side of the field read this stored value, and if necessary they reflect using equations we found. For more info, see the “Grid Reflection” section under the “Development” tab.



*Advantages*

* Literally halves the number of autonomous variations we have to manage

*Disadvantages*

* If the drivers or coach forget to flip the switch to the right alliance, autonomous would likely fail, and the robot could interfere with our alliance partner’s program or get penalized

*Modular Design*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam non lacus malesuada, dignissim mi ac, fringilla mi. Aliquam vestibulum metus commodo urna blandit congue. Fusce ac dolor viverra, semper eros eu, ornare ante. Nulla dignissim, ex eu vehicula viverra, nunc ligula faucibus arcu, sit amet mattis urna augue vitae mauris. Phasellus placerat, ligula non aliquet scelerisque, sapien metus interdum erat, ut suscipit urna libero consequat risus. Donec libero ipsum, volutpat sit amet enim et, aliquet luctus orci.



*Advantages*

* Great opportunity for expansion
* As long as the parts of a ‘module’ used by other parts of the program remain, that entire module can be rewritten with little or no impact on the rest of the program
* Allows for many different parts of the software to be developed simultaneously

*Disadvantages*

* Modular design is generally less efficient when it is run (for our team though this is not significant enough to be a problem)
* Modular design usually requires more code to set it up initially

*Separate Navigation and Action Systems*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam non lacus malesuada, dignissim mi ac, fringilla mi. Aliquam vestibulum metus commodo urna blandit congue. Fusce ac dolor viverra, semper eros eu, ornare ante. Nulla dignissim, ex eu vehicula viverra, nunc ligula faucibus arcu, sit amet mattis urna augue vitae mauris. Phasellus placerat, ligula non aliquet scelerisque, sapien metus interdum erat, ut suscipit urna libero consequat risus. Donec libero ipsum, volutpat sit amet enim et, aliquet luctus orci.



*Advantages*

* Allows the robot to move around on the field while doing other actions, this allows for the limited autonomous time to be used more effectively

*Disadvantages*

* Since the two systems are running at the same time, there are only a few ways to guarantee the order of events between the two systems (we added a way to group events together that alleviates this issue)

*A Few Other Features*

* A “debug hook” which is used to locate where in the program serious errors/”exceptions” in the code are occurring
* A safety feature of our code, where if an error/exception occurs while autonomous is running, the program basically stops running our code. This helps prevent issues (penalties, interfering with our alliance partner, etc.) that could result from a bug, and aids in fixing the bug (prevents a “cascade” of bugs caused by the same code)
* Nulla dignissim, ex eu vehicula viverra, nunc ligula faucibus arcu, sit amet mattis urna augue vitae mauris.
* Phasellus placerat, ligula non aliquet scelerisque, sapien metus interdum erat, ut suscipit urna libero consequat risus. Donec libero ipsum, volutpat sit amet enim et, aliquet luctus orci.