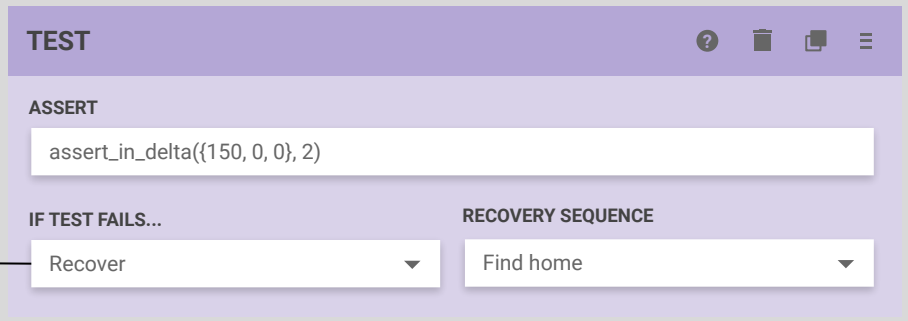
The **ASSERT** input should allow the checking of basically anything in the FBOS state tree (locations, peripheral states, wifi signal strength, encoders, hardware settings, FBOS version, etc) using Lua assertions



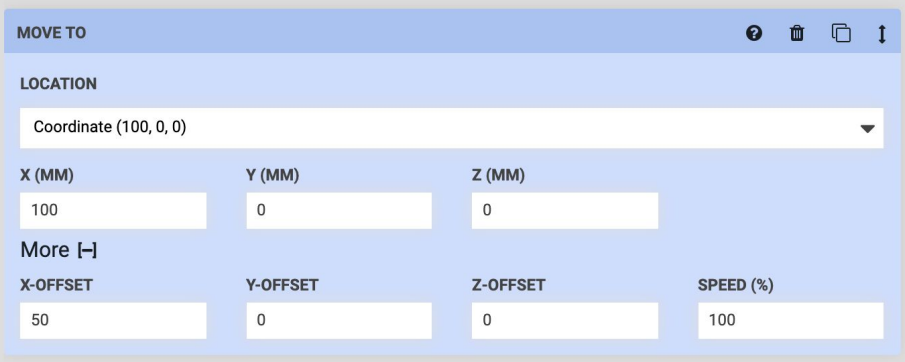
Continue
Recover
E-stop

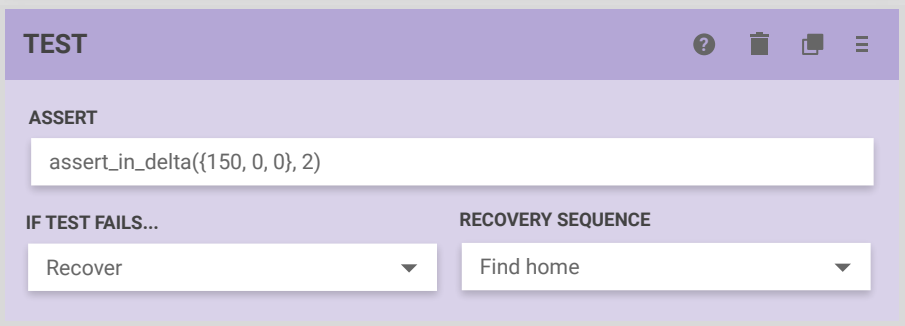
Test results (pass or fail) are always sent as a log.

The **RECOVERY SEQUENCE** field only shows if needed (when the Recover option is chosen for if the test fails). This would be useful, for example, to go back to the home position and then send an email, or to try a test again.

Example uses

This example checks that the bot moved to the correct position (within 2 mm of tolerance). Specifically, it tests positive coordinate x-axis movements and positive x-offsets. Further movements and test commands could be added to check all possible combinations of +/- coordinate, offset, and axis combinations for the **MOVE TO** command. Could easily create tests using variables as well.





What types of bugs would this automate catching?

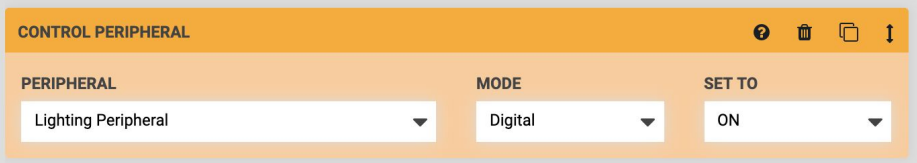
With several beta releases of the v8 `next` branch, basic movements like this were executing incorrectly and it took a few tries and in-person checking of a bot to iron them out. One bug caused an incorrect axis to be moving. Another bug was subtracting offsets instead of adding them. Both of these bugs made it past the FBOS test suite, but would be quickly caught with this method of testing.

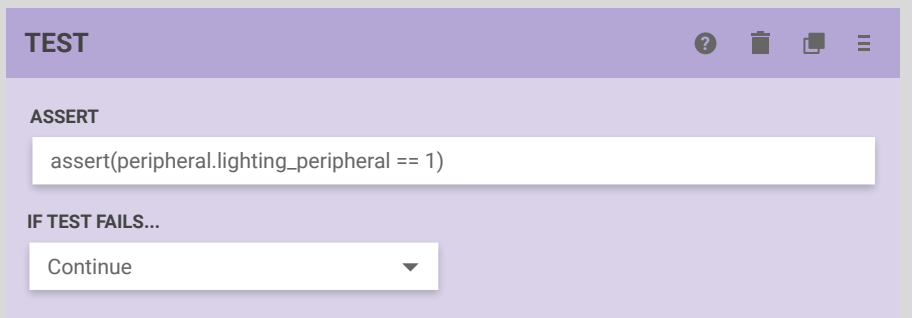
While it is fairly easy to catch bugs for simple examples like this in-person, the Test command could allow us to automate such tests, and be significantly more comprehensive to cover more permutations and more complex scenarios. Comprehensiveness will be especially critical with the launch of variables, multi-variables, multi-location variables, variables of different types, etc because the possibilities of how one can use FarmBot are going to dramatically increase.

Does the system work as expected with a multi-axis movement, traversing from positive coordinates into negative coordinates, with a destination location being a variable with multiple offsets?

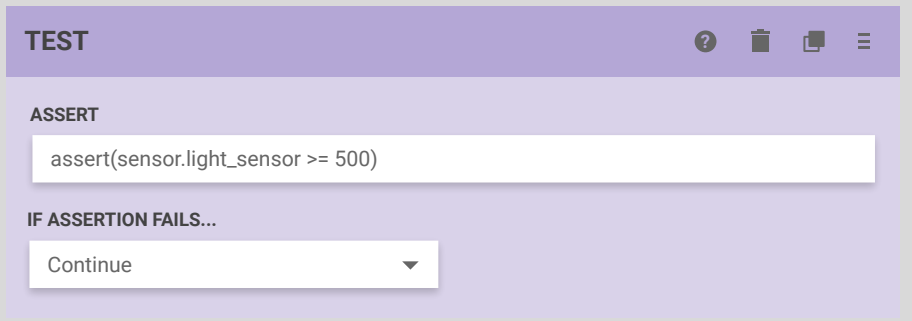
That would be pretty dang easy to test with the Test command, either with several smaller unit-like tests or with a more integrated one-fell-swoop kind of test. Such examples of that complexity are exhausting to test manually and thoroughly with each release, let alone with each beta, but they would be trivial tests to create (and re-use) with the Test command.

This example checks that the bot turned on the lighting. Again, while it might be easy to surface-level check things like this now by simply looking at a bot, it's not so fast and easy to make sure this command works in all the ways it can be used: with Peripherals (named pins) vs raw pins, with digital vs analog mode, turning ON vs OFF, etc. And imagine where we're heading: Variable Peripherals, Variable Values, SPI based peripherals(?), etc.





We could also use the Test command with an instrumented FarmBot to do even more comprehensive testing. For example, if a bot had an ambient light sensor hooked up to it, we could check the value of the light sensor instead of the state of the lighting peripheral to see if the lights are actually on. This would test not only FBOS, but the firmware and hardware as well as more of an integrated test.



We could use an instrumented FarmBot to test out most of the common uses of FarmBot's hardware:

- **Movements**: Limit switches and/or hall effect sensors could determine if FarmBot is at a physical location or not
- **Solenoid valve**: A flow meter could determine if the solenoid valve is open or closed
- **Vacuum pump**: A microphone could determine if the vacuum pump is on
- **Lighting**: Light sensor as described above

Diagnostics/Test Page

To take this testing approach one step further, Test command logs could be sent out with a new log type, `Test`, and/or on a dedicated AMQP channel. These could be intercepted by the FE and used to provide glanceable/easy-to-read test results on a dedicated page, or the logs page with filtering.

Imagine a bot being set up to run a suite of several hundred tests every night between 1am and 4am. You could login to that bot's account and see what's passing or failing right away without having to do much manual QA or sift through logs.

It would also be great to build up an idea of how hardware performs and fails over time in a better-than-anecdotal way.

We could develop Test Suites tailored to our various bot models (provided to users via account seeding or something) which would help customers diagnose hardware issues, and help us help them.