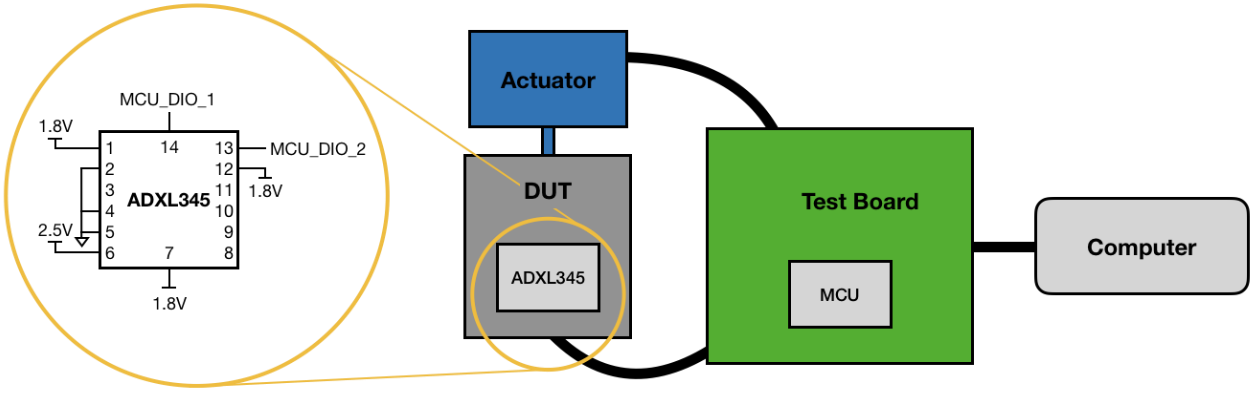
SW Test Challenge: Accelerometer Test Script



Overview

For this exercise you will implement a Python test script for a made-up DUT (device under test) that contains an accelerometer IC.
As you can see in the diagram above the accelerometer IC is an ADXL345 and it is connected to the Test Board from which it receives power and communications. The Test Board also controls an actuator which you can command during the test to move the DUT through various configurations. The Test Board is then connected to the computer where your script will be run and there you can import the *test_board_fwk* module. **Assume this module is already completely implemented and you do not need to write any of the functionality.** The module has a driver class *TestBoard* with the following functionality:

__init__()	Finds and connects to board, raising *ConnectionError* exception if it fails
turn_on_ps(supply)	Turns on power supply for provided string arg (e.g. *"1V8"*, *"2V5"*, *"3V3"*, *"5V"*)
turn_off_ps(supply)	Turns off power supply for provided string arg (e.g. *"1V8"*, *"2V5"*, *"3V3"*, *"5V"*)
i2c_setup(sda, scl, freq)	Sets up a 2-wire I2C bus with provided string sda and scl pins as seen in diagram (e.g. *"MCU_DIO1"*) and integer frequency in Hz
i2c_cmd(addr, data, resp_len=0)	Writes to the provided device address *addr* the provided bytes list *data* and then waits for a response from the device of byte length *resp_len* (where 0 means no response is expected). Returns list of read bytes of length specified. Raises *I2CError* exception if it fails.
actuator_move(config)	Moves the actuator according to the provided string config. This function is **blocking** during motion so it returns only after it is completed (~10 sec). Config options are *"slow_climb"*, *"sharp_turn"*, and *"quick_drop".*

	Raises *ActuatorError* exception if it fails.

The Challenge

With the information provided above and the ADXL345 datasheet write a test script that when run will go through the following high-level test procedure, failing the test as soon as any of the individual checks fail:

- Configures the accelerometer to output valid x,y,z measurements at the maximum data rate possible given communications and sensor constraints
- Run accelerometer self test and check results are within datasheet range
- Move actuator through "*slow_climb*" config and check throughout the motion that the y axis is between -1g and 1g, and that the z axis is between 6g and 8g
- Move actuator through "*sharp_turn*" config and check throughout the motion that the x axis and y axis are both greater than 5g
- Move actuator through "*quick_drop*" config and check throughout the motion that the z axis is less than -8g

Regardless of any exceptions that occur during the test the script must always finish by logging the test result in the following format:

If test passed	TEST PASSED in *[insert time elapsed]* sec
If test failed	TEST FAILED in *[insert time elapsed]* sec due to *[insert exception and any other relevant details]*

To support your test function you can write any number of methods and classes and these can be separated into different files as you see fit and please try to stick to core python libraries for your solution.
Feel free to add any comments in code when you find you need to make assumptions about the problem.