Hardware classes:

Notes:

* Each of the real and simulated hardware classes inherits an abstract device class, the most basic of which being AbstractDevice, and perhaps metaclasses thereof, for example as AbstractSensor and AbstractBalance in a hierarchical pattern.
* The purpose of these classes is to define the methods and attributes that external classes can count on a certain type of device to have, without worrying about the implementation details for a specific model of device.
* This allows for a new hardware class for one type of device to be developed according to the abstract template and swapped out easily, ideally changing only the line of code where the device is imported.
* In Manager, notice that the type hints for devices are the name of the abstract class. For example, “Oscilloscope: AbstractOscilloscope” This allows the IDE to give type hints and code autofill suggestions for the classes’ abstract template.
* Although the type hints say abstract, keep in mind that the instances of the device classes in Manager will be a subclass, either for a real device or a simulated device.
* To see the documentation for the underlying abstract class, mouse over the method name in PyCharm. If a docstring for the abstract method exists, It will show up in the mouseover interface, unless it is overridden by the docstring of the child class
* One drawback to this approach is that if you use ctrl-click to step into a hardware method from the manager class, it will bring you to the abstract class when you may instead want to see the actual hardware class. To get around this, just go to the desired class in the hardware folder and use ctrl-f to find the desired method.

Manager

Important note for programming manager:

Even though manager is a QThread object, not all code run in or from the manager class runs in that separate thread. **Only code originating from manager’s run method runs in the background thread.** The proper way to direct the manager thread to do long-running tasks from the UI thread is to connect a command\_signal(str) to the manager’s exec\_command slot. This method simply receives the command and stores it until the manager gets around to acting on it in the core event loop. In the core event loop, there is an if/else tree that turns the command into the proper function call. Additional commands can be added to this tree in the future.

* Manager begins the script with a step index of -1, indicating no script is being run. A script begins when manager’s begin\_script\_slot receives metadata from the MainWindow.
* This sets the scripting flag to true, which is the que for the manager’s core event loop to begin advancing the script. It will continue advancing until the variable scripting is false, which will happen if the end of the script is reached, or the script is terminated early.
* If the scripting variable changes from true to false, the manager will show a script complete dialog, even if the script ended prematurely. This occurs in the core run() loop
* The MainWindow includes buttons to abort the script after the current step, or to abort the script immediately. these buttons emit signals which are connected to the corresponding slots in Manager.

RFB data acquisition procedure:

1. Set the AWG to the desired frequency and turn it off.
2. Initiate a reading from the balance, forward power meter, and reflected meter all at once at t = 0 seconds.
3. As soon as all three are finished, record the time and the status of the AWG, and initiate another capture of all three sensors simultaneously. This takes roughly 1/20th of a second but may vary slightly. The sensors always initiate capture at the same time.
4. Repeat step three for the duration of the test.
5. After a specified amount of time (the off time parameter), turn the AWG on
6. After a specified amount of time (the on time parameter), turn the AWG off
7. Repeat steps 5 and 6 a specified number of times (the # cycles parameter), and repeat step five to finish with an off interval.

RFB data analysis procedure: Timeline

Description automatically generated with medium confidence

1. Take the on/off intervals taken by cross-referencing the awg\_on\_ray and the time\_s array and add a predetermined delay to them slightly to allow for sensor transition time (1.5 seconds is recommended). Also remove a fixed number of samples from the end of each interval (3 are recommended).

\*This was previously done by extrapolating from the data with an unknown approach, but I think this method is simpler, perhaps more repeatable, and likely be significantly different.

1. Acoustic power is calculated from the balance reading using the formula:

balance reading (kg) \* g (m/s^2) \* speed of sound in water (m/s) = acoustic power (W)

1. Take the standard deviation of the forward power, reflected power, and acoustic power within each interval. If it exceeds the limit from the config file (for example .05 Watts) the test is considered invalid. The software will stop the analysis and depending on the “Interrupt action” setting in the config file, the software may prompt the user to retry or retry automatically.
2. Random and total uncertainty are relics from the LabVIEW software design and their formulas are not known. If a test with the new python software outputs a file it has determined the standard deviation of the data is acceptable.
3. As a comparable replacement for the output file, I propose the following measurements of uncertainty. These are subject to change and can be viewed or modified in the file /Utilities/formulas.py
   1. Random uncertainty % of an interval: (max - min) / 2 / mean \* 100
   2. Total uncertainty % of an interval: random uncertainty + sensor uncertainty
      1. According to files from the previous system the uncertainty percentage of the balance is 6%
4. Average the forward power, reflected power, and acoustic powers across each on interval
5. Efficiency percentage is calculated using the following formula (from design specifications)
   1. Eff % = Acoustic Power / (Forward power – Reflected power) \* 100
6. Reflected power percent is given by the following formula (from design specifications)
   1. Ref % = Reflected Power / Forward power \* 100
7. The calculated forward power to deliver the target acoustic power (for example 4 Watts) is determined by the following formula:
   1. Target acoustic power / Average acoustic on power \* Average Forward on power
8. The element fails the test if reflected power is above the limit (typically 50%) or if the extrapolated forward power is above the limit (typically 12W)

Sources:  
Acceleration due to gravity

*Taylor, Barry N.; Thompson, Ambler, eds. (March 2008).*[*The international system of units (SI)*](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication330e2008.pdf)*(PDF) (Report).*[*National Institute of Standards and Technology*](https://en.wikipedia.org/wiki/National_Institute_of_Standards_and_Technology)*. p. 52. NIST special publication 330, 2008 edition.*

Speed of sound in water

Engineering ToolBox, (2004). *Water - Speed of Sound vs. Temperature*. [online] Available at: https://www.engineeringtoolbox.com/sound-speed-water-d\_598.html [Accessed Day Mo. Year].