|  |  |
| --- | --- |
| **Actuators Module**  **Module Description** | **ID: DC-DOC-3** |

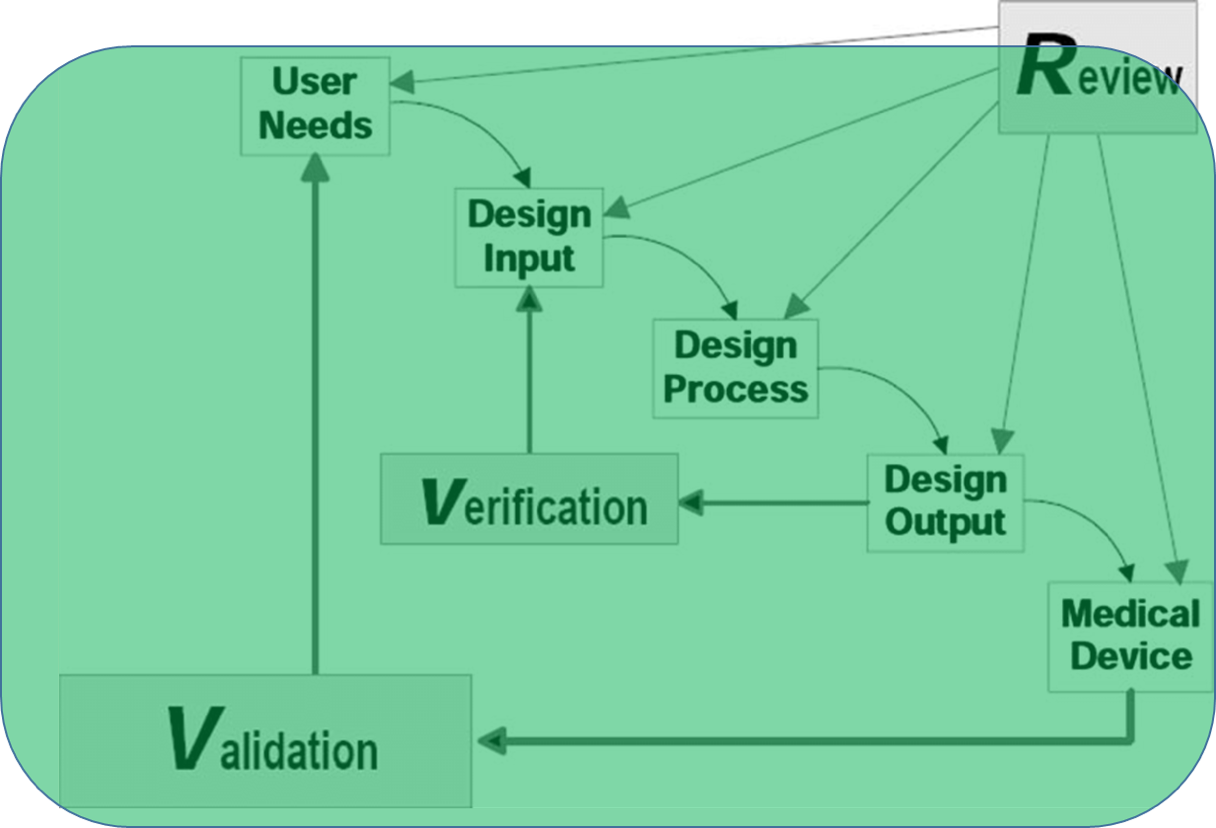


Figure : Waterfall Diagram, Food and Drug Administragion Design Control Guidence Document, March 1997

**Learning Objectives**

|  |  |  |
| --- | --- | --- |
| **Design Thinking** | **Skills** | **Tools** |
| * Quantitative decision making to solve open-ended problem * Design Outputs * Writing Requirements * Requirement verification / validation | * Arduino C++/ANSIC programming * Functional Programming * Stepper motor control * Microcontroller inputs and debounce | * Arduino * Breadboard * Stepper Motor Driver Pololu A4988 |

**Designette**

**User Story**

Many engineered tissues require some type of bioreactor to allow cells to proliferate and grow or establish communication pathways, remodel the underlying hydrogel matrix, and even begin producing vasculature through angiogenesis. These bioreactors can differ pretty broadly, but many of them use some form of microfluidics or small environmental chambers as the bioreactor. These microfluidics usually require pumps. These pumps can be syringe pumps, gas-pressure pumps or peristaltic “roller” pumps. We often end up using these roller-style pumps in my lab to support tissue constructs in small microfluidic systems. Overall these pumps are inexpensive, and they get the job done but they could always be improved. For one thing, it would be ideal for several of these pumps to run off one motor with a way to detach or stop individual pumps when necessary. So, we could have multiple pump heads without the need to replicate the stepper motor, driver, Arduino etc. Also, these stepper motors can get pretty hot when they are run at low RPM, and these motors are often inside of an incubator. This can cause the roller pump assembly to warp. So, finding some way to avoid that from happening would be good. We could also use a really good user interface so the user could input some variables regarding tubing type and desired volumetric flowrate and the Arduino code would automatically set the step speed. Lastly, we could use a well-designed PCB or board layout for soldering or assembling all of the small electrical components, so we don’t have things like breadboards in the incubator. Or fixing the Solidworks CAD files to add configurations so the pumps could be resized based on using different tubes, and having a set of engineering drawings would probably be good. Or really any other advancement on the user interface or function of these pumps would be great.

**For this designette**

* Begin by building a working micropump as currently designed.
  + Make a video and upload to canvas.
  + This is an individual assignment.
* Generate UI that accepts volumetric flowrate and tubing geometry as inputs and sets the appropriate control conditions in software.
  + Design a test to show that the user inputs correctly set the pump flowrate.
  + This is an individual assignment.
* Improve the design. Tell us what you are going to improve on the design in the Deliverable Worksheet. Then in some meaningful way verify / validate that you accomplished your design goals. This is very open-ended. You should state how you are going to make it better, say how you will test that you met those goals then run those tests. Use any of the templates provided to present your prototyping and testing. If you need to change a template, delete a table that is not being used, add a new table, add a plot, etc… You are free to make those changes to the templates. They are only a guidance document to help you with formatting.
* You do not necessarily need to fabricate and test a physical object, though you may if it makes sense and would meet the previous criteria. However, if you made the changes to the Solidworks files to make a configuration-driven design, you could show in CAD drawings that the configurations would work for differing tubing sizes. Or if you designed a PCB and had a well laid out wiring diagram and component selections with associated datasheets, this would work also. If you redesigned the UI, you could show this with screenshots and the base code.
* You should include any Arduino code that you wrote. This code should use comments for all variables, functions and section headings. The code should also be functionalized so that the main loop in reasonably short and all computations are done by subfunctions.
* You will submit the Design Controls Worksheet, one Prototyping Protocol, and one Test Protocol. I would rather you spend your time being creative and doing engineering than filling out documentation. Make these documents short. I just need to know what you are doing, how you are doing it and how you plan to prove that you did what you set out to do.

**Deliverables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Deliverables** | | **Points** | **File ID** |
| 1. | Completed Design Controls Worksheet | 30 | DC-W-3 |
| 2. | Prototyping Protocol | 30 | P-P-3 |
| 3. | Test Protocol | 15 | T-P-3 |
| **Individual Deliverables** | | **Points** |  |
| 4. | Arduino Code incl UI upgrades | 15 |  |
| 5. | Functional Micropump (Individual Grade and Upload) | 10 |  |
|  |  |  |  |
| **Due Dates** | |  |  |
| Team Deliverables | | 3/28/2021 | 11:59PM |
| Individual Deliverables | | 3/28/2021 | 11:59PM |
|  |  |  |  |

**Additional information for individual assignment deliverables**

Video upload - Please include in your video

* Talk through your video and say what you are doing
* Evidence that pump turns and light blinks when triggered by pushbutton.
* Evidence that UI defaults to some value that results in pump turning.
* Evidence that lowering flowrate results in slower pump rotation for equivalent diameter.
* Evidence that increasing diameter results in slower pump rotation for equivalent flowrate.
* Write a brief statement outlining your tests above as a comment in the header of your Arduino Code.
  + So others can test your code at a later time and ensure functionality.
* Example video here: [DemonstrationVideo\_WorkingPump.mp4](https://ucdenver.instructure.com/courses/449997/files/12769729/download?wrap=1)