# 3. Software used and the reason for using this software

The software that we CS students will be using will be the SIMAVR. The reason being that it is the only one out of the 3 software that allows for substantial coding and for implementing a viable program to use the Arduino.

The Multisim and the Tinker card simulators are great tools for experimenting with the actual hardware and testing with various inputs from the Arduino, however they are not entirely viable options for extensive coding with the Arduino.

These 2 simulators can be used to visualise the effects of the Arduino on the bioreactor and should be used in line with the SIMAVR to develop a project that covers both the software and hardware aspects.

We would also be using the MIT app inventor to develop an Android app to pair with the Arduino so users can use mobile phones to control the bioreactor, adding an extra layer of convenience.

# 4. Key milestones (that you need to meet for the project)

1. Develop a working program for all 3 of the subsystems (heating, stirring, pH) that correctly adjusts the bioreactor to the corresponding settings.

a. Based on the specifications of the subsystem, write programs to allow the system to read the sensor values correctly (e.g., heating is at x degrees)

b. The program should then output control values that sets the state of the bioreactor (e.g., heating is set to y degrees higher/lower)

c. The bioreactor reacts correctly to the changes and the sensor reads the adjusted value (e.g., heating shows the adjusted x + y degrees)

d. Improve the program by writing different working scenarios for the subsystems or changing their efficiencies (e.g., by changing the speed of heating)

2. Test the program extensively to ensure the program is working and the subsystems are well integrated with one another. (by this time should have received the working plan for all 3 subsystems) Also, run debug tests to prevent the system from failing under certain circumstances.

a. Test the program within the range that it is supposed to work in and fix the bugs that are produced in the tests.

b. Test the system outside the range to check how the system reacts to a reading that is outside of its usual limits. If it doesn’t work, improve the code such that it deals with extreme values correctly (producing error messages etc.).

c. Run full pilot tests on the bioreactor to check if it works in strained conditions, such as prolonged periods of time, extreme environmental conditions, fluctuating power supply or when large amounts of information are being fed into the system.

3. Review the parts that produce significant problems and work on those parts. Continual process of coding, testing and bug-fixing.

# 5. Difficulties and challenges (What are the constraints and limitations of your proposed solution?)

1. As we are not working with an actual Arduino or a bioreactor, we do not get to test out our systems in real time. As a result, the programs that we write are harder to test effectively and may not operate as smoothly.

2. We do not have a simulator that encompasses the Arduino, the electronics, and the physical system. This means that we must use different simulators for each branch of our project, which may be difficult to integrate smoothly into our plan.

# 6. Ways to overcome difficulties (What are the different options and approaches that a designer could consider and follow when tackling this challenge?)

1. Extensive testing throughout every stage of the assembly and coding process. This technique helps us eliminate potential bugs in each component before we integrate then together. Making sure each component does exactly what it is meant to do under each condition.
2. After each component and their software is functioning smoothly, integrating the multiple different systems together and continuing to test and fix bugs until we have a whole, synced, integrated system which we can further test. This method fixes bugs at every stage making the integration of the different subsystems a simpler task at the end.

# 7. References (at least 2)

1. MIT Android App Inventor <https://appinventor.mit.edu/>
2. Software development cycle <https://youtu.be/i-QyW8D3ei0>