Part 2:

The link to the repository:

https://github.com/KevinAbeykoonUniversityAccount/SYSC4001_A1

Based on the simulations that were run, it seems as though the overall execution time relied heavily on the time taken by the ISR process. Since the CPU was relatively faster, the ISR steps slowed the entire process down. This is especially apparent when increasing the ISR activity time to 200 milliseconds. After graphing time taken of the simulation steps, it can be seen that the overhead process takes over twice the time than the CPU work. Increasing the CPU speed will decrease the overall execution time but will be less efficient than increasing the overhead speed.

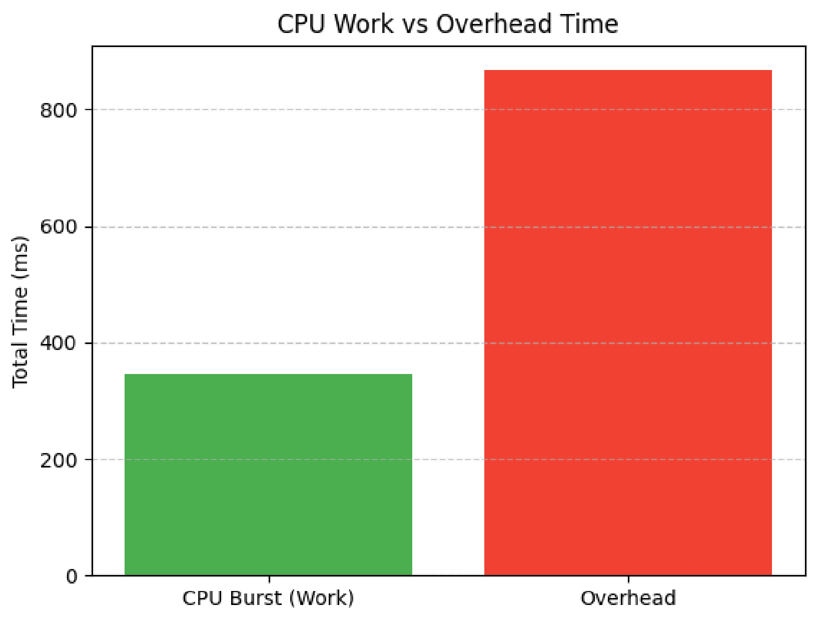
You must run numerous simulation tests and discuss the influence of the different steps of the interrupt process.

1. Change the value of the save/restore context time from 10, to 20, to 30ms. What do you observe?
   Since the save and restore happens very often, minor increases to the context time causes the time to increase by a lot! Compared to the isr time however, it doesn't boost the overall time taken by the program by the thousands. Also, incrementing the context time from 10 to 20 and 20 to 30 have very similar results.

2. Vary the ISR activity time from between 40 and 200, what happens when the ISR execution takes too long?

   The most obvious thing that happens is that the overall execution time is much longer. It starts to bog down the entire process with the ISR taking up more time than the actual CPU computations. Also, by using up all the time that was set out for the devices, it eliminates all the time spent on checking for errors and device status. This could potentially lead to uncaught errors that completely derail the execution.

3. How does the difference in speed of these steps affect the overall execution time of the process? You can process this data using a Python script, Excel spreadsheet or any other tools for separating the overhead from the actual work of your program (i.e., CPU use for processing and actual I/O needed by the program).



As mentioned in previous answers, by increasing the time taken by the ISR process, the overall process takes much more time. This results in a much slower overall execution time. As seen in the graph above, the overhead processes take up over twice the time in milliseconds than the CPU's work. This is a very inefficient use of the resources as very little work is being done throughout the execution. By decreasing the overhead, the save/restore context time and ISR activity time in this case, a much more efficient execution cycle can take place. A graph representing this would have the CPU's Work time be at a similar level or greater than the overhead.

4. Ask yourselves other interesting questions and try to answer them through simulations. For instance: what happens if we have addresses of 4 bytes instead of 2? What if we have a faster CPU

   Without the faster CPU and in the current simulation, nothing would happen. The reason being that the simulation does not take into account the memory accesses, in other words, a longer address doesn't do anything. If the simulator did properly retrieve data from memory addresses, then using 4 bytes instead of 2 would have resulted in the program being able to access much more memory. A 2 byte word can store approximately 65,000 memory addresses. A 4 byte word can store over 4 billion addresses. With a faster CPU, means that the CPU can complete its work much faster. Its time consumption would lower for the same amount of work. As the CPU gets more and more efficient, the overhead processes start to become the bottleneck that slows the efficiency of the entire system down.