Real-Time Shortest-Job-First Scheduling with Preemption on TIVA TM4C Microcontroller

By Caleb Rash and Zachariah Abuelhaj

The Problem

- Implement Shortest Job First scheduling
    - Three different process types, each with a different execution time
    - Each process has critical section

- Preemption
    - User adds process to Queue at any time
    - When a process is added, the Queue adjusts to prioritize

- Starvation
    - With smaller processes in queue, larger ones become lower priority
        - Provides possibility that larger process will never execute

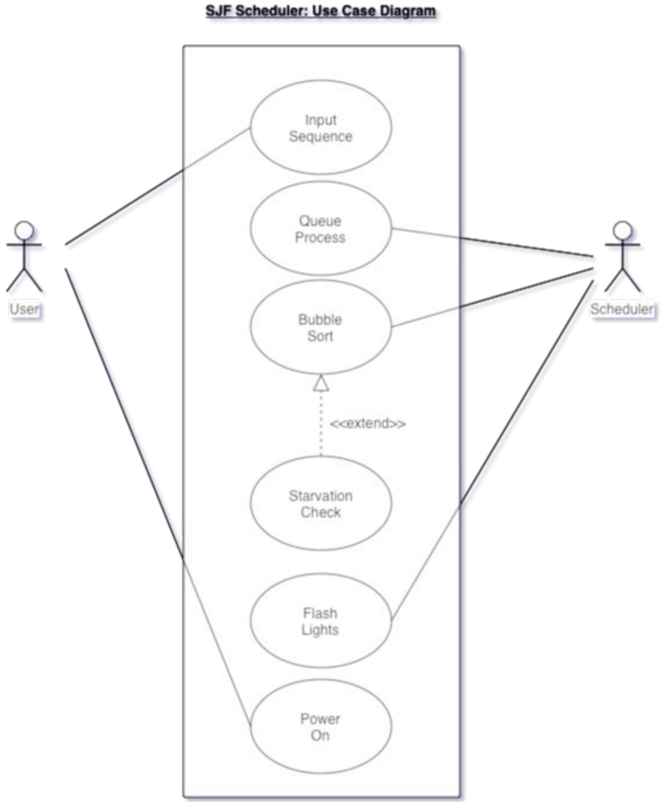
The Solution

- Three different process types
  - P1 - 4 seconds
  - P2 - 8 seconds
  - P3 - 12 seconds
- Defined with struct
  - Process ID: This is used for lighting correct LED of running process
  - Time Waiting: This is used to identify starvation in the system
  - Time Remaining: This is used to determine priority of a process
- Critical sections
  - Each process has a 1 second-long critical section
  - Enters critical section 3 seconds into run-time
  - Process does not get disturbed or preempted in critical section

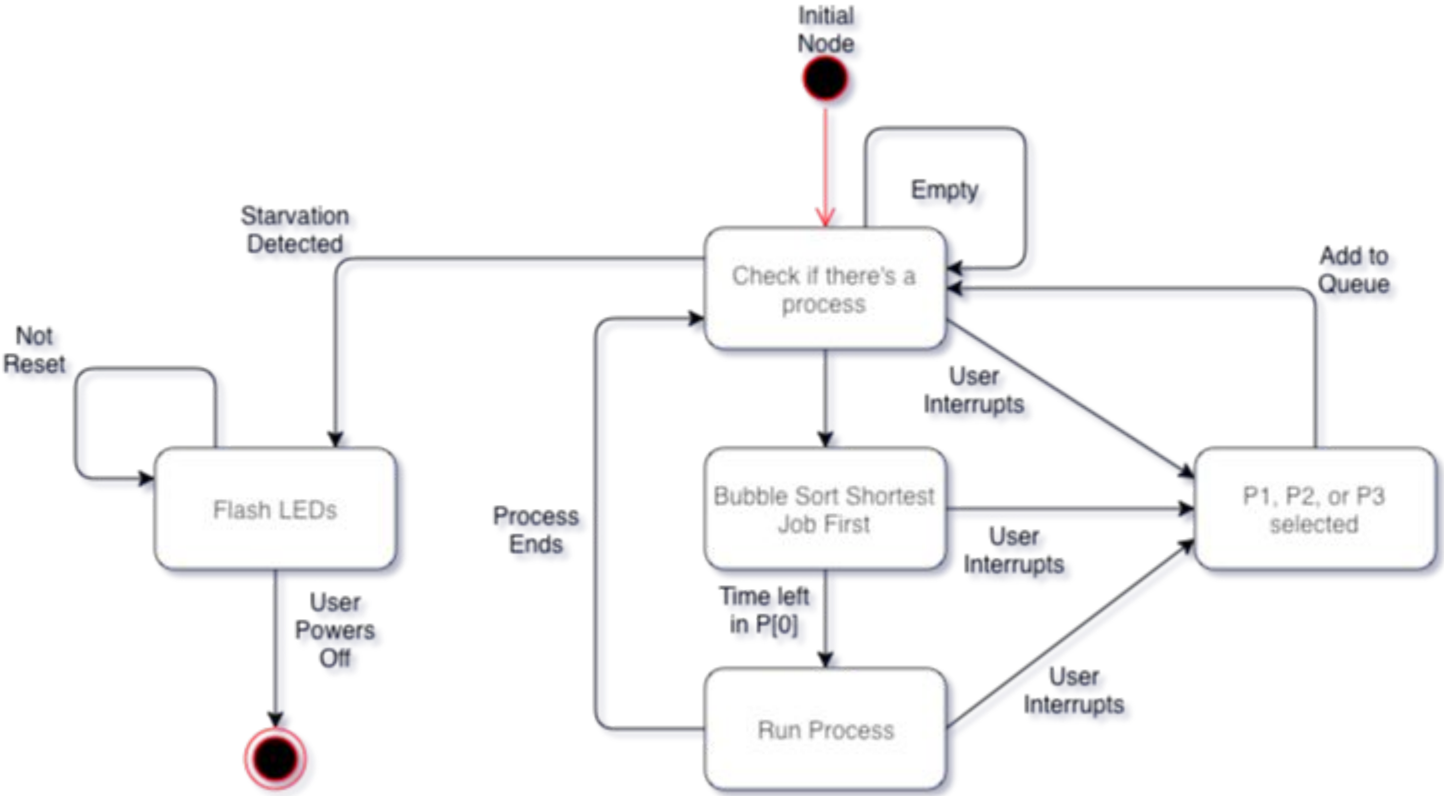
Solution (cont.)

- User adds process at any given time
  - Push-button interrupts to add chosen process to Queue
  - Queue, which is 10 slots wide, will hold each added process
  - Bubble-sort to find shortest process available
    - Based on time left to execute
- Starvation
  - User can input combinations of processes and fill queue
    - i.e. if a P1 gets added repeatedly and a P3 keeps waiting
  - To detect starvation, check how long each process has waited
    - If a process waits longer than 25 seconds, then flash LEDs until reset

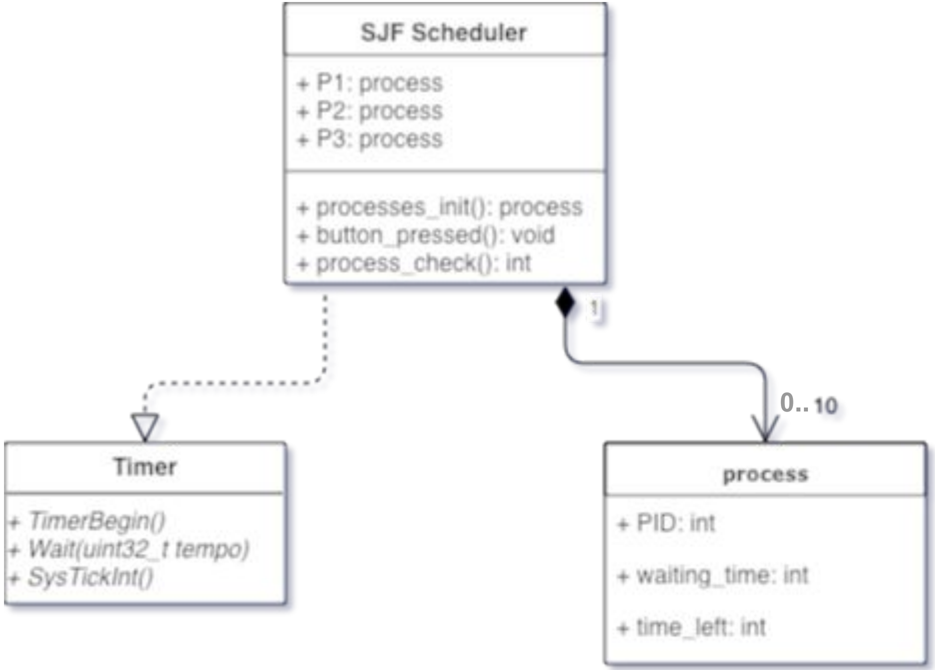
Implementation: UML Use-Case Diagram



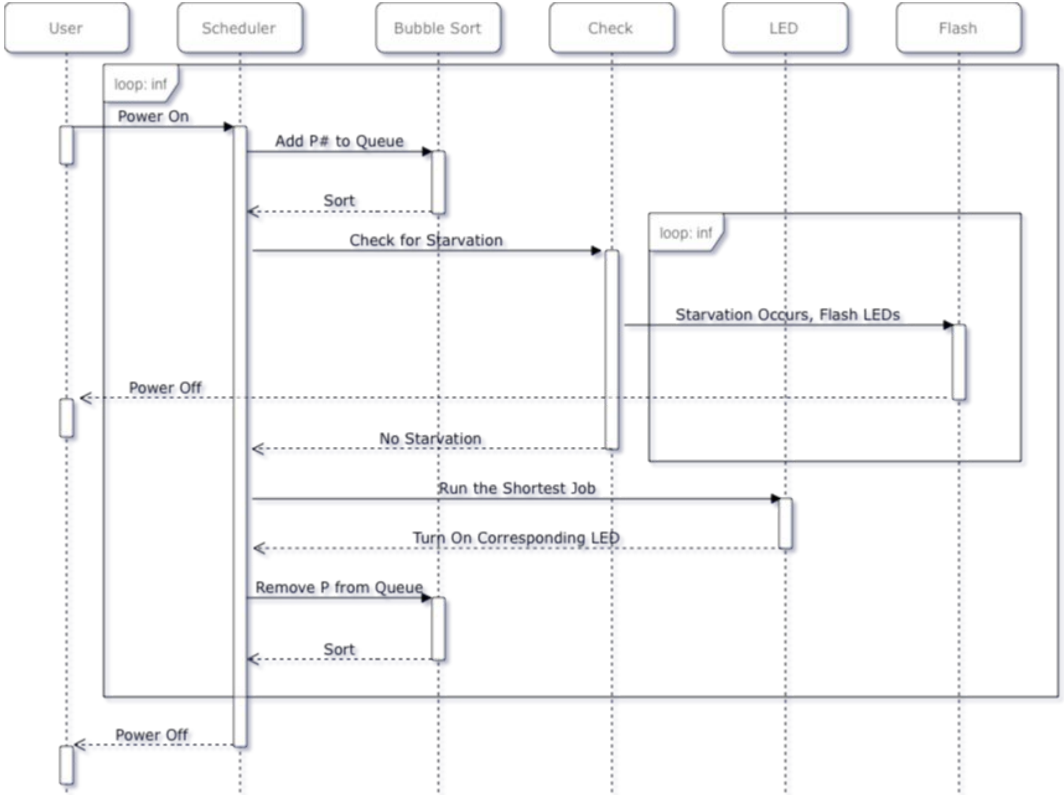
Implementation: State Diagram



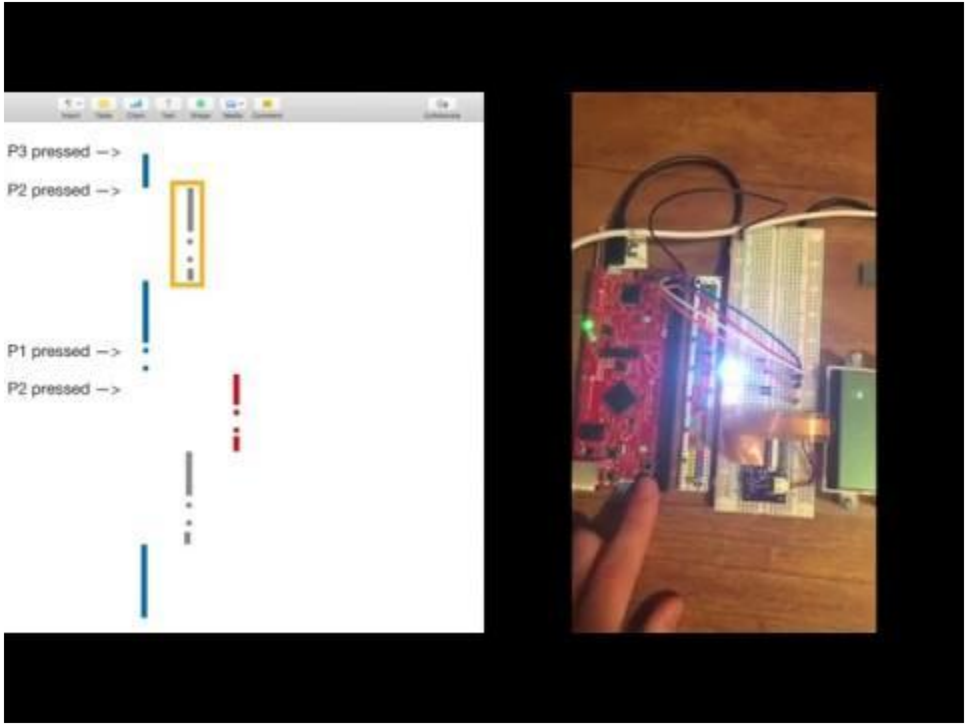
Implementation: UML Class Diagram



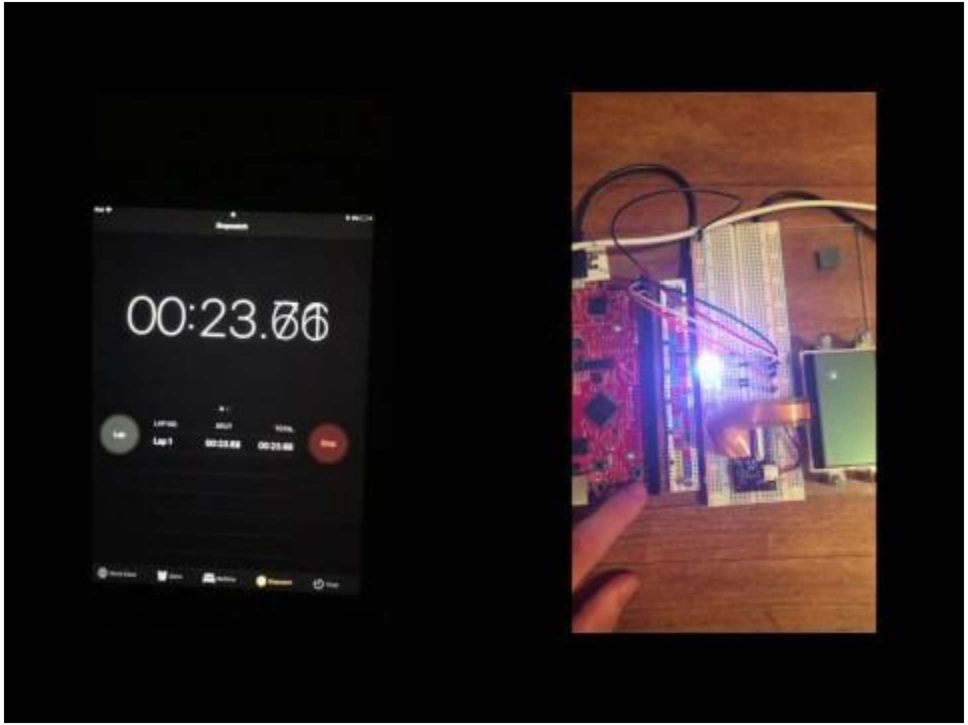
Implementation: Sequence Diagram



Results (Example Operation)



Results (Example Starvation)



Conclusions and Future Work

- Successfully implemented Shortest Job First, which includes preemption, on a Tiva TM4C microcontroller
    - Ran in real-time
    - Process struct used with timers to simulate actual processes/tasks to be ran
    - Lightweight and easily modifiable
- Takeaways:
    - Could be converted to a library for implementation on simpler embedded systems
        - Replace processes with different I/O tasks for the microcontroller to schedule
    - Not as complex as TI-RTOS, easier to use understand
- Future Work:
    - Could improve starvation detection to run process in queue rather than exit system

Questions