Lab 3: Monitor, Tasks, Synchronization, Semaphore, Mutex

Farhang Nemati

In this lab you will implement a monitor that wraps a data structure and synchronization facilities that are hidden from tasks using the data structure.

A. Monitor

Implement a monitor that contains a data structure (an array) which can contain a limited number of integer numbers. Adding and reading the numbers form the data structure has to be in queue manner, i.e., First In First Out (FIFO). Notice that it is not allowed to use FreeRTOS queues.

- The monitor has to provide 3 functions; *init()*, *read()* and *add()*.
- *init()* initializes the monitor, e.g., initialize semaphores.
- *read()* reads and removes the number from the top of the queue if it is not empty. If the queue is empty the caller task will block.
- *add()* adds a number to the end of the queue if it is not full. If the queue is full the caller task will block.
- No more than one task is allowed to add to or read from the queue at the same time.
- The protection and synchronization has to be wrapped in read() and add() functions.

B. Producer and Consumer Tasks

Implement 4 periodic tasks; 2 producers and 2 consumers. The producer tasks will periodically add random numbers to the monitor and the consumer tasks will read the numbers from the monitor and print them out onto the Serial port.

Notice: To generate random numbers use the following code:

```
#include <time.h>
#include <stdlib.h>

srand(time(NULL));   // Initialization, should only be called once, i.e., in setup()
int r = rand();      // Returns a pseudo-random integer between 0 and RAND_MAX.
```

Examination

To pass the lab hand your code (only in Blackboard!). You also need to demonstrate the programs for the lab assistant in one of the lab sessions. The lab assistant may ask you questions related to lab task.