EECE 276
Embedded Systems

Task communication:
Queues, mailboxes, pipes
Task communication

- Tasks need to communicate:
  - Shared data structure: Queue
  - Operations: (RTOS services)
    - Queue Initialize(int size): create and set up for size
    - Void Enqueue(Queue q,void* item): block if full, else put
    - Void* Dequeue(Queue q): block if empty, else get
  - Multiple tasks use it: *mutual exclusion* is enforced
Task communication

- Mailboxes:
  » Similar to queues, (usually) has a single element and two states: empty/full
  » If multiple tasks are waiting for a mailbox, the highest priority one gets the message

- Pipes:
  » Streams of data
  » Typically use fwrite()/fread() operations
  » Arbitrary length and data format
Using queues

1-to-1:

Producer Task → Queue → Consumer Task

1-to-many:

Producer Task → Queue

Consumer Task

Highest priority task gets the data
Using queues, mailboxes

Many-to-1:

Producer Task \rightarrow \text{Queue} \rightarrow \text{Consumer Task}

(Non-deterministic) Merge of data

Double buffer:

Producer Task \rightarrow MBox1 \rightarrow Consumer Task

Producer Task \rightarrow MBox2 \rightarrow Consumer Task
Problems with queues, etc.

- If of finite size, task may block/receive error if full/empty
- Passing pointers as queue elements: data becomes shared ("ownership" changes)
- The sender and the receive must agree on the format/meaning of the data passed
  - void* is not a very good idea
- Memory management: running out of memory space leads to disasters (or reboots)