# ECE 443 – Project 3

## Objective

Receive messages from PuTTY via a UART, save the messages in I2C EEPROM, and then retrieve and display on the LCD in response to a BTN1 press.

## Specifications

#### UART

* The UART will be configured to generate an interrupt every time a character is received
* Messages will be no more than 80 characters and terminated by a carriage return (‘\r’). Only printable characters and “spaces” – no other control characters
* Carriage return will not be hit a second time until the previous message has been stored in the EEPROM, i.e., no “pending” messages to store, although characters can be typed for a second message
* LEDA will be lit to indicate “Ready for new message”, i.e., “NOT full”, at which time the carriage return can be hit. It should go off when the carriage return cannot be pressed.
* UART should be configured for 19200 Baud, odd parity, eight data bits, and one stop bit
* puTTY should be configured to force off local echo. Instead, the application should echo received characters as they are typed

#### EEPROM

* Messages are stored and retrieved according to a first-in/first-out ordering
* No more than five messages will be stored in the EEPROM, pending display

#### Button

* Each time BTN1 is pressed a message will be retrieved from the EEPROM and displayed on the LCD
* Multiple button presses (not bouncing) are treated as “pending” retrievals. That is a second retrieve request can be issued while a message is being retrieved and displayed on the LCD
* LEDB is lit when there are no additional messages to retrieve, i.e., “empty”. Subsequent presses while the EEPROM is empty are ignored.

#### LCD

* Messages are displayed in a “rolling” fashion using the two LCD lines and broken across lines at spaces. No truncated words! One word per line is considered less than ideal.
* Each message will start on the second line of the LCD, and then scroll upwards in a “rolling” manner until the LCD is blank.
* The LCD will scroll at a rate of one line per second and the screen will remain blank for a full second before any new message is displayed.

#### “Heartbeat”

## Include a task that toggles LEDC every millisecond at the highest task priority in your system.

Lastly, take advantage of the FreeRTOS mechanisms for cooperation amongst tasks, limiting your use of global variables as much as possible and utilizing “information hiding.” Utilize Tracealyzer to characterize the behavior and performance of your project. Document your code using Doxygen

## Deliverables

* Share your project with me via OneDrive before 10 pm PT on the specified due date
* Demo to the instructor during the next class period
* Upload to Canvas a 2-3 page professional report in PDF format that documents your implementation and, in particular, the communication (control and data) between software modules and the module functionality. In addition, explain *in detail* what is working (and not working). Include in your report output from your Tracealyzer experiments and your Doxygen PDF.

## Bonus

I’ll give an extra 10% if you utilize only task notifications and stream/message buffers instead of semaphores and queues. 😊