

**April 2nd, 2018**

**Embedded Systems I**

Third Practice: I2C-based UART-controlled system

*Celis Tapia Daniel* - ie689705

*Araiza Obeso Gustavo* - ie699416

Introduction

*The following report shows the development of a UART-controlled system with an external memory and a real-time clock module to save and retrieve data throughout a user interface from the computer terminal and a Bluetooth terminal. It also allows the user to chat between the two controlling terminals.*

*A main menu allows the users to select from the different options to execute, mainly to set data into the peripherals and retrieve it. After every operation, the user may continue through other menus.*

Functional Analysis: General State machine

For the development of the practice, we need to create a series of tasks that allow us to properly execute the sequences. Basically, we need tasks that communicate what we need to do on the hardware side, and a series of sequences that call tasks and wait or their completion to execute the necessary steps to work properly.

On the diagram, we can see as the key point our escape sequence, since it is the one with the highest priority. There are some initialization sequences that create the corresponding tasks for each terminal, but they are eliminated as soon as the menus are started up.

Development:

Resources are protected by a mutex to avoid any priority issues. Most calls are made through event groups, which in turn are the ones that allow the sequences to work properly, as they keep track of the enabled tasks and the corresponding done flag. The main sequences used for this practice are:

* ***ESC sequence:*** The escape sequence is called whenever the terminals receive the escape character (# due to the Bluetooth terminal limitations). It breaks up the current task and recalls the menu to allow new user input.
* ***Menu sequence:*** Menu sequence is the one that calls the other corresponding sequences. Once finished, they return to the menu sequence as it is the base for the application.
* ***Write memory sequence:*** Waits for an address and the data to place in the external memory.
* ***Read memory sequence:*** Waits for an address and a data length to return the saved data on the external memory.
* ***I2C task:*** Task that receives a transfer handle indicating the parameters to operate the I2C hardware components.
* ***Receive task:*** Receives the user input to the system, queues it or breaks up if it receives a return character (end of message) or an escape character (calling the escape sequence).
* ***Transmit task:***  Empties its queue and sends it to the corresponding terminal.

Conclusions

This practice gave us an insight into the implementation of the communication protocols and their interaction to allow a user to get and set information from different peripherals. The biggest challenge was to understand the core functionalities of the FreeRTOS platform and how to use its structure to place the corresponding sequences to perform properly and without any issues between two terminals.

Personally, I found challenging to really understand what were the issues that arose during the development of the practice. Some of them were not explicitly caught by the IDE, so they required a careful approach to see where the issue stemmed.