RMIT University

EEET2490 – Embedded System: OS and Interfacing, Semester 2024-1

ASSESSMENT 2 – INDIVIDUAL ASSIGNMENT

OBJECTIVES

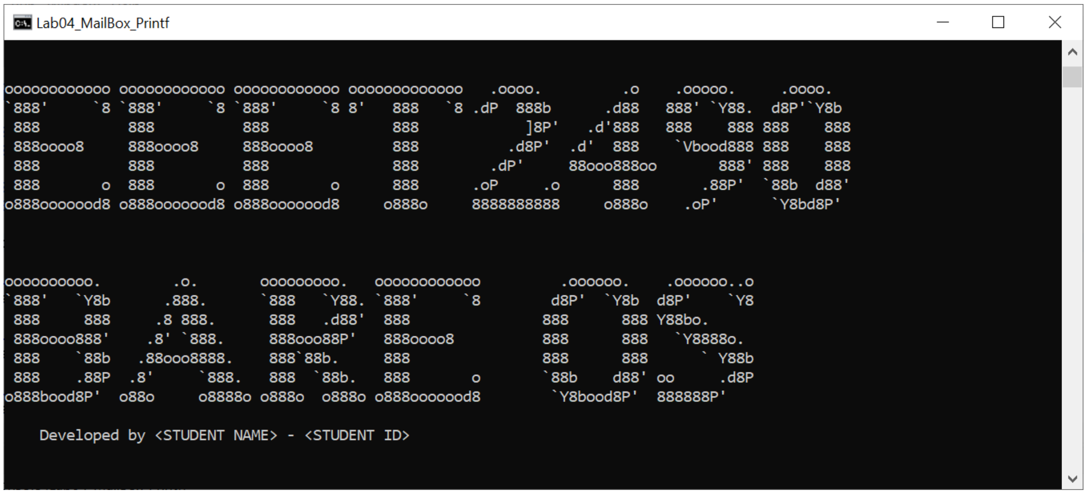
Bare Metal Development of an Operating System is an important topic to enhance learners' practical skills. In this particular assignment, students will further strengthen development skills and gain practical experience of embedded OS through implementation of additional features such as command line interpreter (CLI), ANSI code for terminal formatting, standard printf function, and variable arguments handling.

DESCRIPTION

Students should read all of the tasks below before starting of implementation, since one task may support another task.

1. Welcome Message and Command Line Interpreter (CLI)

Normally when an OS successfully boot up, it will show a **Welcome Message** (text/ screen/ sound). For your Bare Metal OS, you should also have a welcome text like below.

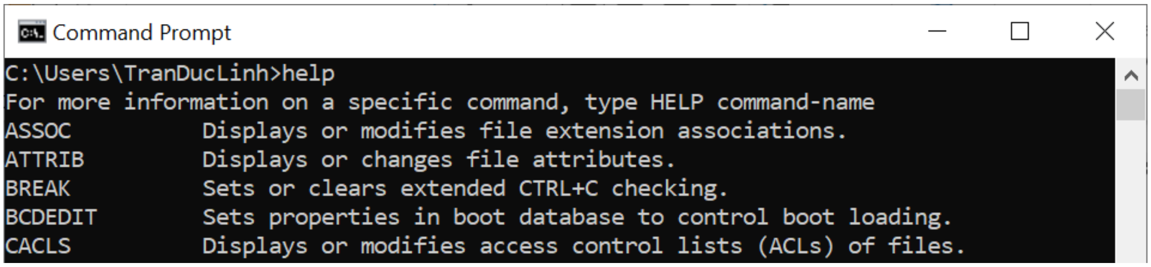


```
oooooooooooo oooooooooooo oooooooooooo ooooooooooooo   .oooo.         .o    .ooooo.     .oooo.
`888'     `8 `888'     `8 `888'     `8 8'   888   `8 .dP  888b      .d88   888' `Y88.  d8P'`Y8b
 888          888          888               888             ]8P'   .d'888   888    888 888    888
 888oooo8     888oooo8     888oooo8          888           .d8P'  .d'  888    `Vbood888 888    888
 888          888          888               888         .dP'     88ooo888oo       888' 888    888
 888       o  888       o  888       o       888       .oP     .o      888       .88P'  `88b  d88'
o888ooooood8 o888ooooood8 o888ooooood8     o888o     8888888888     o888o    .oP'       `Y8bd8P'


oooooooooo.        .o.       ooooooooo.   oooooooooooo         .oooooo.    .oooooo..o
`888'   `Y8b      .888.      `888   `Y88. `888'     `8        d8P'  `Y8b  d8P'    `Y8
 888     888     .8 888.      888   .d88'  888               888      888 Y88bo.
 888oooo888'    .8' `888.     888ooo88P'   888oooo8          888      888  `Y8888o.
 888    `88b   .88ooo8888.    888`88b.     888               888      888      ` Y88b
 888    .88P  .8'     `888.   888  `88b.   888       o       `88b    d88' oo     .d8P
o888bood8P'  o88o     o8888o o888o  o888o o888ooooood8        `Y8bood8P'  888888P'

    Developed by <STUDENT NAME> - <STUDENT ID>
```

Supporting Resource: You may use this tool https://onlineasciitools.com/convert-text-to-ascii-art to help convert any text into an ASCII art string, then just print it out to the console. You are free to select any font and customize the welcome message.

Furthermore, **Command Line Interpreter (CLI)** is an indispensable feature of every operating system. For example, Windows 10 provides two CLIs for their users - Command Prompt and PowerShell, whereas Linux users use Shell on a regular basis.

Example of some commands in Windows' Command Prompt



```
C:\Users\TranDucLinh>help
For more information on a specific command, type HELP command-name
ASSOC          Displays or modifies file extension associations.
ATTRIB         Displays or changes file attributes.
BREAK          Sets or clears extended CTRL+C checking.
BCDEDIT        Sets properties in boot database to control boot loading.
CACLS          Displays or modifies access control lists (ACLs) of files.
```

For your OS, implement a **command line interpreter** that reads keypresses into a buffer and then executes the commands when the key "Enter" is pressed. Implement some feature below for your CLI:

- An **OS name (e.g. MyOS)** as *initial text that is always displayed in the console while waiting for user to type commands*
- **Auto-completion:** user can use TAB for auto-completion of the command.
- **Command history:** to browse within command history. Typically the user use UP and DOWN arrow to do so. However, these special keys are not read by the UART, thus, please use _ key as UP arrow and **+** key as DOWN arrow for this feature.

Implement **some commands as below** to test your CLI:

No.	Comand Name	Usage
1	`help`	- Show brief information of all commands - Example: **`MyOS> help`**
	`help <command_name>`	- Show full information of the command - Example: **`MyOS> help hwinfo`**
2	`clear`	- Clear screen *(in our terminal it will scroll down to current position of the cursor).* - Example: **`MyOS> clear`**

3	`setcolor`
	`-t <text color>`
	`-b <background color>`
		- Set text color, and/or background color of the console to one of the following colors: BLACK, RED, GREEN, YELLOW, BLUE, PURPLE, CYAN, WHITE
		- Examples:
		`MyOS> setcolor -t yellow`
		`MyOS> setcolor -b yellow -t white`

4	`showinfo`	- Show **board revision** and **board MAC address** in correct format/ meaningful information - Example: **`MyOS> showinfo`**

Supporting Resource:

- **ANSI Escape Characters and Color Codes**

  In order to set color for a terminal, you may need to use ANSI Escape Codes and Color Codes.

For example, put a code “**\033[42m**” before you print out “HelloWorld” text and observe your terminal. A good reference about ANSI Codes could be found here:

https://gist.github.com/fnky/458719343aabd01cfb17a3a4f7296797
https://chrisyeh96.github.io/2020/03/28/terminal-colors.html

- **Information for Board Revision** can be found here:

  https://www.raspberrypi-spy.co.uk/2012/09/checking-your-raspberry-pi-board-version/

2. Further Development of UART driver

Further develop the UART driver (PL011 UART) so that it can support the following configuration via the **Command Line Interpreter (CLI).** Develop and provide the commands that users can use to do the UART configuration.

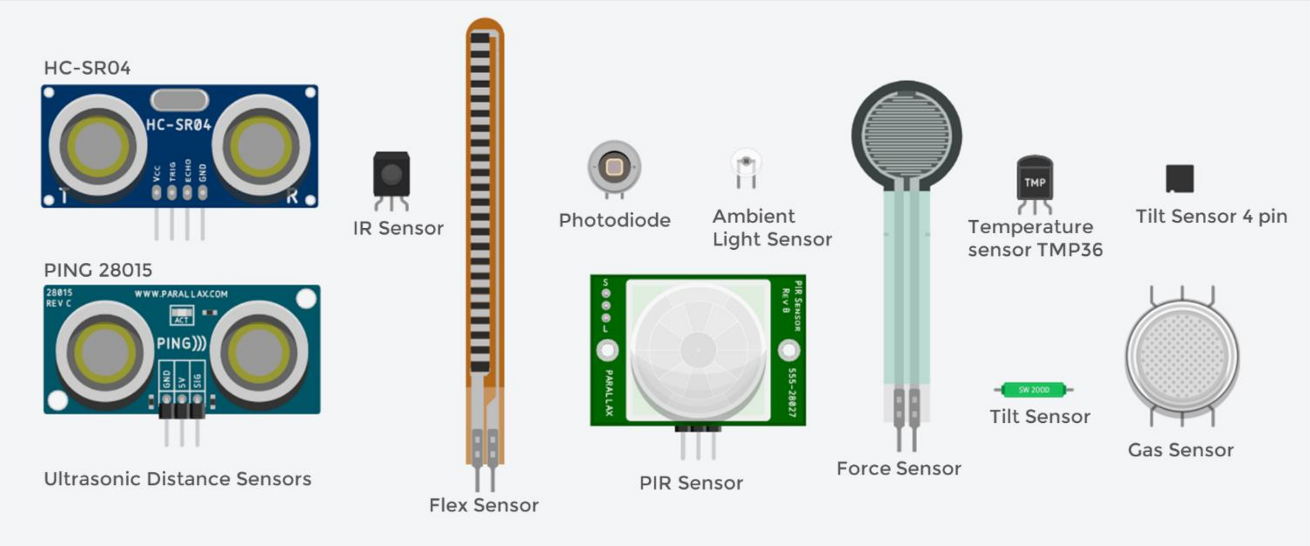
- Configurable **Baud Rate**: The driver must support various baud rates, including but not limited to 9600, 19200, 38400, 57600, and 115200 bits per second.
- **Data Bits**: Support for configuring the number of data bits (5, 6, 7, or 8).
- **Stop Bits**: Ability to select between one or two stop bits.
- **Parity**: Support for none, even, and odd parity configurations.
- **Handshaking control**: CTS/RTS handshaking.

UPDATE: For Task 1 and 2, it is required to test with both QEMU emulation and the real Raspberry Pi board.

3. Get Used with Some Common Sensors

TinkerCad Circuits is a good environment for simple quick protopying which includes microcontroller (Arduino/ Microbit), sensors, and variety of supporting devices. It can be accessed at this link: https://www.tinkercad.com/learn/circuits

The figure below is a list of common sensor that are available in TikerCad Circuits:



Choose at least **03 different types of sensors above and discuss in detail,** including how it works, function of each pins and possible applications with example circuits demonstrated on TikerCad, also compare with real devices available on the market if there are any differences.

4. Individual Reflection

Besides academic knowledge, it is always good to have a look at the practical world. For this section, first of all, search for yourself and explore some job descriptions of Embedded Software Engineers.

With your international accredited degree, you can work all around the world, not only in Vietnam. For example, LinkedIn is a good source to start your search:

https://www.linkedin.com/jobs/embedded-software-engineer-jobs/?originalSubdomain=sg

After that, write an individual reflection that reflect on your learning:

- What are the challenges and opportunities you met while working on bare metal OS development (labs and assignments)?
- What are the new things that you have learned? Are they related with skills/ experience required from job descriptions listed above?
- What are the skills or knowledge that you want to further explore/ improve?

Note: The length of this part should not be too long (recommended length is around 1-2 page).

Report and Presentation

Complete your **report** with discussion of all parts. You should provide some background information, explanation for your work, and also include screenshots of output as evidence of a successful outcome.

For presentation and demonstration, please record a **video** of maximum 10 mins that present, explain and demonstrate your work.