Lab#4 Embedded application design project using a RTOS Part2

Leonardo Fusser (1946995)

Objectives:

- See part 1.
- Define tasks and task's priorities in a RTOS environment.
- Use a Queue management in a RTOS environment.

Hardware: Explorer 16/32 with pic32 PIM, 2 USB cables.

To hand in: See part1.

Multi file system and software requirement: See part1.

Embedded application

By building upon your previous lab, you are to add some features to the vending machine system by respecting the updated requirement document.

Updated requirement document:

The LCD must display "`Out of order-Being Serviced`" whenever the machine is being serviced by a technician.

The refrigerator temperature must be read regularly.

The LCD must display "`Out of order-Temperature issue`" whenever the fridge temperature is lower than 0ºC or higher than 8ºC.

The following transaction examples will clarify the different user interface functionalities.

Examples of user interface transactions:

Whenever a technician services the machine via the serial port the user interface must be disabled automatically:

```
Out of order
Being serviced
```

To re-enable the user interface, the technician must first check out from servicing by typing 'k' through the serial interface. Technician servicing is covered below.

Whenever the temperature is out of range the user interface must be disabled automatically and display:

```
Out of order
Temperat. Issue
```

Technician servicing

The technician can do transactions like load soda bottles, read the soda stock, monitor temperature and more. See menu example below.

```
**********************************************************************
[Explorer 16/32 Vending Machine]
Enter “r” to refresh display.
Enter “o” followed by a 2 digits value between 00 and 99 to load orange bottles.
Enter “c” followed by a 2 digits value between 00 and 99 to load cherry bottles.
Enter “l” followed by a 2 digits value between 00 and 99 to load lemon bottles.
Enter “t” to read fridge temperature.
Enter “e” when the machine's money is removed.
Enter “h” to find out the last transaction time (in seconds).
Enter “m” to read the number of quarters currently in the machine.
Enter “s” to read the soda stock.
Enter “p” to change soda prices.
Enter “k” to check out from servicing.
**********************************************************************
```

The following transaction examples will clarify the technician transaction functionality.

Example of servicing transactions

Adding 12 orange bottles:
```
o12
Orange stock updated
Orange in stock is now: 27
```

Read fridge temperature:
```
t
Fridge temperature: 2 Celsius
```

Read the number of quarters currently in the machine:
```
m
Currently 213 quarters in the machine
```

When the machine's money is removed the technician must select “e” to reset the total credit:
```
e
Currently 0 quarters in the machine
```

Read the soda stock:
```
s
Orange : 27 items
Cherry : 1 items
Limon : 5 items
```

To find out the last transaction time (in seconds):

```
h
The latest transaction happened at 15 seconds
```

Change soda prices:

```
p
*************************************************************************
Enter “op” followed by a 2 digits value to change orange price in quarters.
Enter “cp” followed by a 2 digits value to change cherry price in quarters.
Enter “lp” followed by a 2 digits value to change lemon price in quarters.
Enter “r” to return to main menu.
*************************************************************************
op9
Orange is now at 9 quarters
```

To check out from servicing

```
k
checking out ...
```

After checking out the LCD is available again to the users.

Additional information:

The potentiometer on analog channel 5 will simulate the temperature. You must convert the 0 – 1023 range to a -7 ^{o}C to +20 ^{o}C range.

A heartbeat LED at 2 Hz must be implemented.

A watchdog timer must be implemented.

Lab Work

Clone the previous lab repo.

Part 1: Tasking

On top for the user interface task (see previous lab), a minimum of two extra tasks are required although more tasks can be implemented:

1. A task that handles and processes text streams from a serial interface (current lab).
2. A task that executes periodically. E.g., every second (current lab).

Notes:

- Improve your previous task diagram by adding the new tasks.
- Draw a flowchart or a state machine for each new task.
- Give the list of priority of all tasks and ISRs along with a reason:

Task or ISR	Priority	Justification
vTaskUI	3	System responsiveness needed to ensure smooth user operation.
vDatabase	4	Task runs only once to initialize vending machine drinks (name, quantity, and price), so priority is set to the highest.
vTaskTemperature	1	Least important task due to lack of user interaction.
vTaskTechnician	2	Moderately important task, so system responsiveness is somewhat needed to ensure smooth user operation.

Part 2: Code composing

- The teacher must approve your new diagrams before starting programming.

- Again, your must write your program incrementally by regularly testing your tasks. To save time, you must use the simulator when debugging.

- Make sure you are logged into GitHub and clone the previous lab.

Part 3: Test

- When both the “user interface” AND the “technician servicing” are fully functional, give a demo on the real TARGET.
- At the end of the project, you will be asked to modify/add features in a limited period of time.
- Push it to GitHub.
- If required, revisit all your diagrams to consider the latest version of your code.

Approval and demo required.

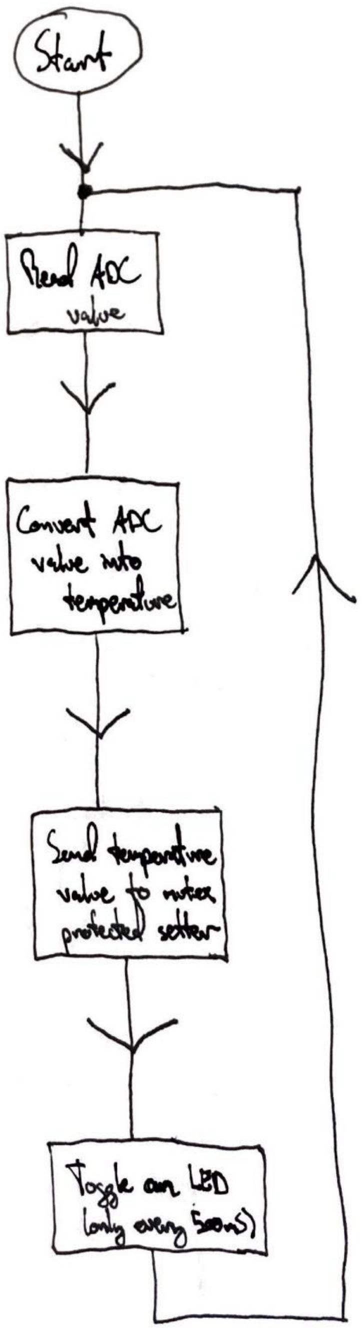
After lab questions:

1- Explain what you learned in this lab. What went wrong and what did you learn from your mistakes. Give specifics please.

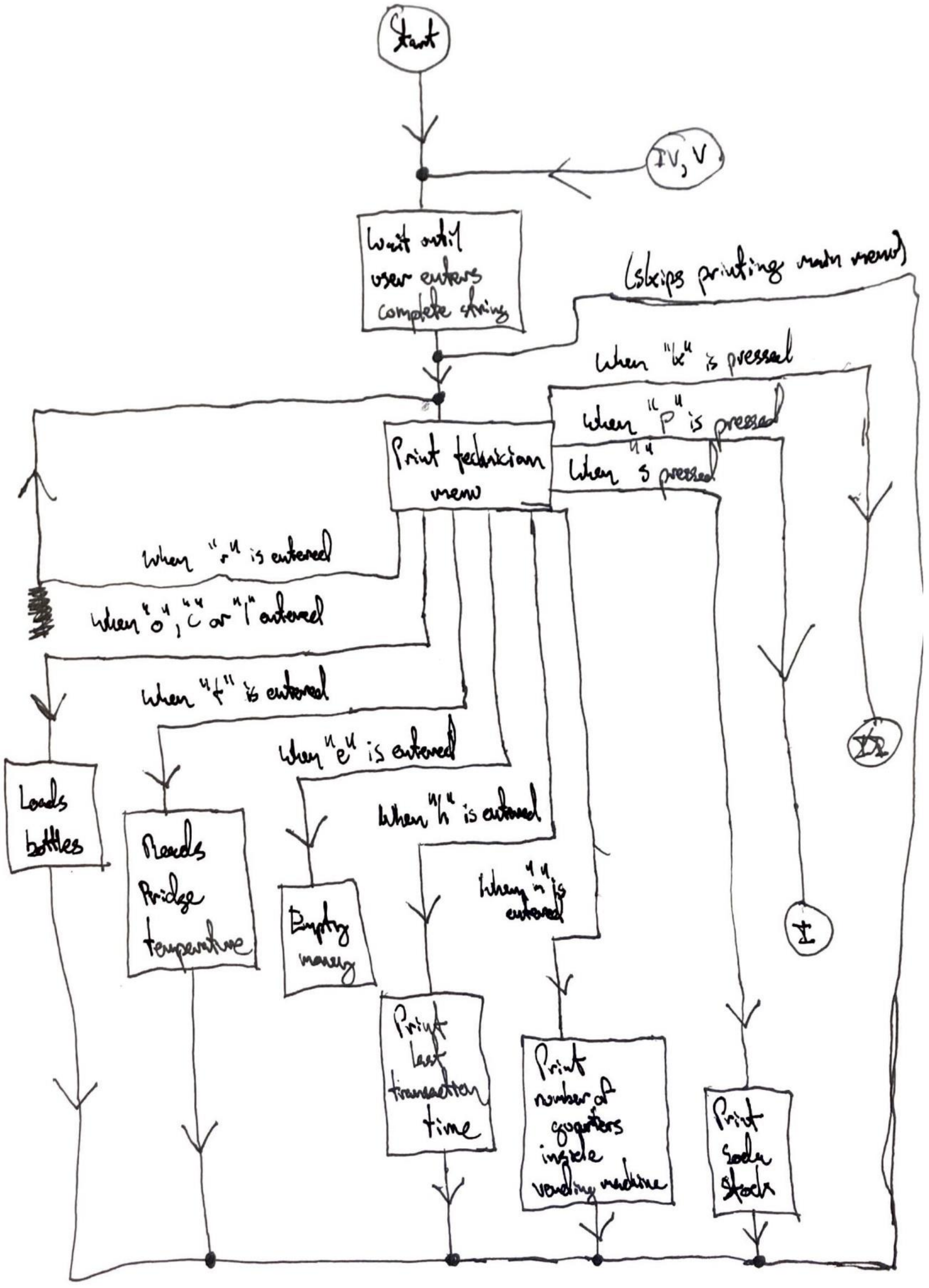
- ➢ The main takeaway from the past two labs were the utilization of multiple tasks with different priorities and using state machines inside the tasks. The tasks needed to be prioritized for some of the reasons mentioned previously in these lab sheets. Furthermore, some of the tasks required the use of state machines in order to reduce code complexity.

  The main issue encountered was that the state machines were a little tricky to debug logical errors. Switching between the simulator and target did not help matters. Some issues occurred when going from the simulator to target and need some more fine tuning. For example, printing on the console worked fine the first time around but going to the target showed otherwise. Lesson learned here is to not always rely on the simulator!

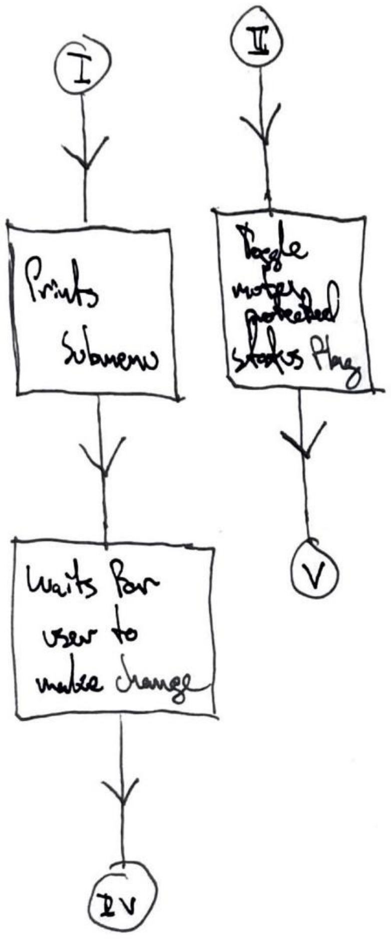
[vTaskTemperature flowchart]


Start
Read ADC value
Convert ADC value into temperature
Send temperature value to mutex protected setter
Toggle an LED (only every 5secs)

[vTaskTechnician flowchart I]



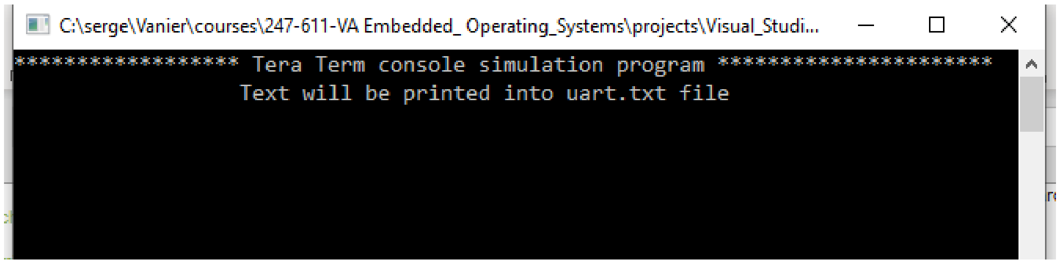
[vTaskTechnician flowchart II]


I
Prints Submenu
Waits for user to make change
IV
II
Toggle motor protected status flag
V

Appendix

The following application will simulate the Tera Term console:

console.exe

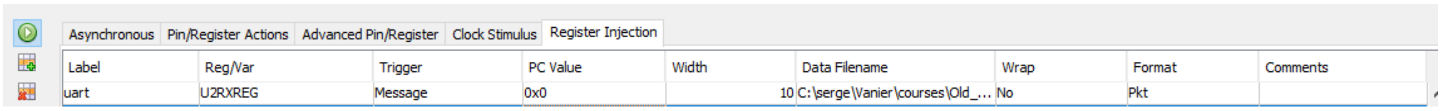


C:\serge\Vanier\courses\247-611-VA Embedded_ Operating_Systems\projects\Visual_Studi...

```
****************** Tera Term console simulation program **********************
                   Text will be printed into uart.txt file
```

All values entered in the console will be saved into uart.txt file.

Serial reception buffer using MPLABX stimulus:



Asynchronous | Pin/Register Actions | Advanced Pin/Register | Clock Stimulus | Register Injection

Label	Reg/Var	Trigger	PC Value	Width	Data Filename	Wrap	Format	Comments
uart	U2RXREG	Message	0x0	10	C:\serge\Vanier\courses\Old_...	No	Pkt	

Data filename should target the uart.txt file