The Vending Machine Controller - RTOS

By
Umang Deshpande and Akshay Hegde

June 2017

1 Objectives

- Understanding the RTOS implementation
- Implementing a Vending Machine Model

2 Prerequisites

- Basics of RTOS, familiarization with tasks, semaphores and task scheduling.
- Drawing up of a statechart from the problem statement. (Please go through Resources in the Resources section for the same)
- Understanding of Statemachine implementation of switch debouncing.

3 Problem Statement

The overall objective is to design a vending machine controller.

1. The system has five digital inputs and three digital outputs. You can simulate the system with five switches and three LEDs.
2. The inputs are quarter , dime , nickel , soda and diet . The quarter input will go high, then go low when a 25¢ coin is added to the machine. The dime and nickel inputs work in a similar manner for the 10¢ and 5¢ coins.
3. The sodas cost 35¢ each. The user presses the soda button to select a regular soda and the diet button to select a diet soda.

4. The GiveSoda output will release a regular soda if pulsed high, then low. Similarly, the GiveDiet output will release a diet soda if pulsed high, then low. The Change output will release a 5¢ coin if pulsed high, then low.

So from user perspective, user inserts desired amount of coins and select Diet or Regular soda for taking out soda from vending machine.

4 Relevant Theory

4.1 Statechart Implementation

Vending Machine has 4 states of operation. *INITM, COIN, SELECT* and *DELIVERY*.

4.1.1 INITM:

- Initially controller will be in INITM state.
- Perform reset/initialization of requisite variables(coins entered, number of soda cans selected, etc.).
- Transition immediately to the next state, i.e.,*COIN*.

4.1.2 COIN:

- Instructions for coin entry and mapped switch presses should be displayed on screen.(As in Fig 1)
- Depending upon which switch is pressed, increment the total sum. Ex: for *DIME* coin entered, $sum = sum + 5$.
- Display total money entered till now. (As in Fig 2, Fig 3 and Fig 4)
- Switch press is used to transition to next mode. Remain in *COIN* mode till mode transition switch is pressed.

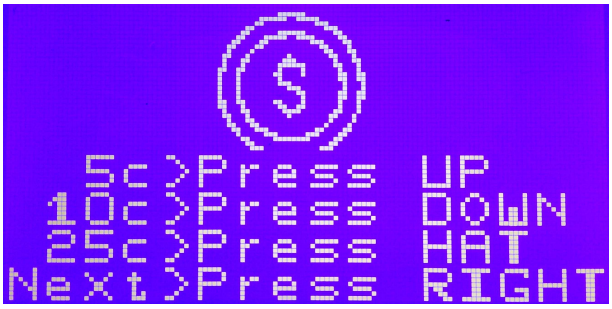

Fig 1: Put Coin Screen(Initial)

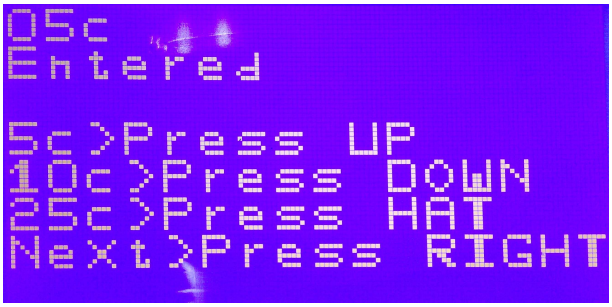

Fig 2: Entered Dime Screen

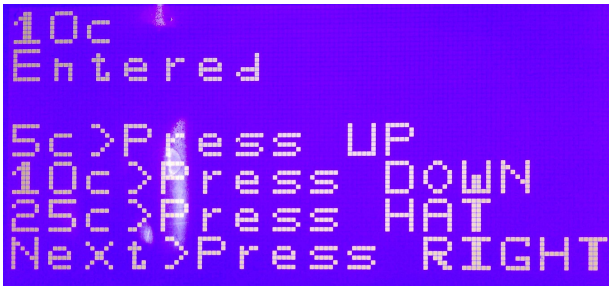

Fig 3: Entered Nickel Screen

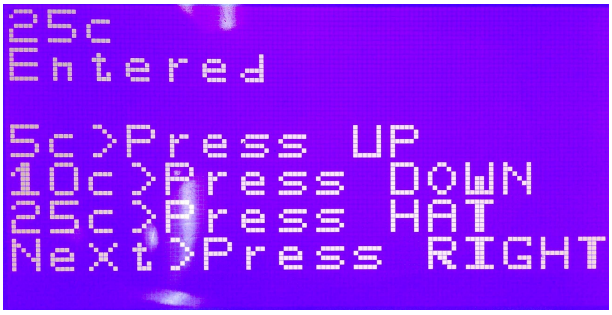

Fig 4: Entered Quarter Screen

4.1.3 SELECT:

- In *SELECT* mode, initially, sode selection screen should be displayed with instructions to select each soda with appropriate switch press.

- Once the requisite switch is pressed, the soda is selected.(And appropriate screen displayed as in Fig 6 and Fig 7). In case, multiple soda selection is allowed, increment each soda count variable.
- If the user changes his/her mind, he/she should be able to opt out. Hence, a cancel transaction option should be provided.(Fig 5 displays the cancel option)
- If cancel switch is pressed, ask for confirmation(as in Fig 8) and then move to *CHANGE* substate in DELIVERY state.
- Also, a transition switch should be provided to move on to *DELIVERY* state.

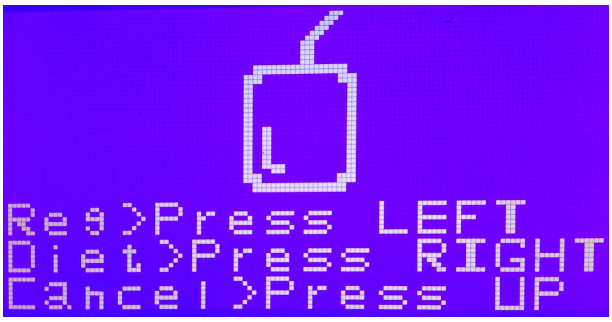

Fig5: Initial Soda Selection Screen

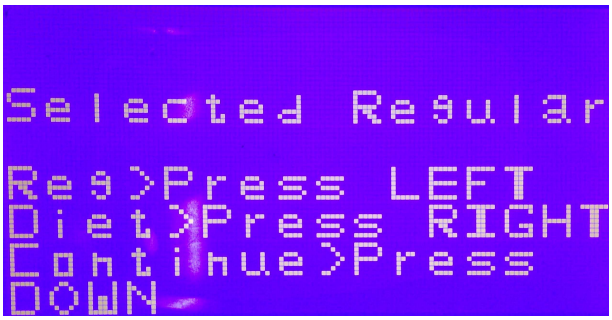

Fig6: Regular Soda Selected Screen

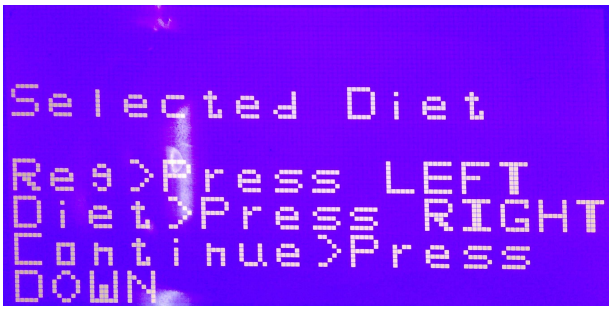

Fig7: Diet Soda Selected Screen

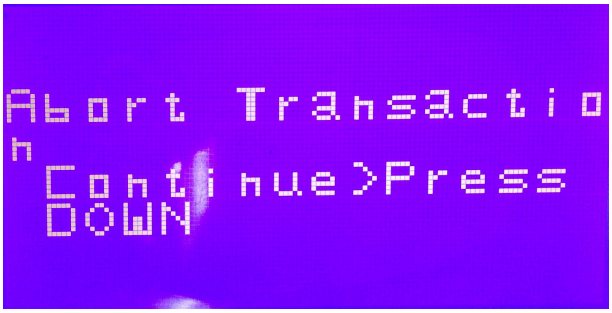

Fig8: Abort Transaction Screen

4.1.4 DELIVERY:

- Based on count of each soda selected, dispense corresponding soda through LED blink, and display the soda delivery screen(as in Fig 9).
- Once soda is delivered, return change such that each LED blink corresponds to 5¢. *Number of LED Blinks = Sum/5*
- Notify the user of coin dispense using a screen(as in Fig 10)

Fig 9: Soda Dispense Screen

Fig 10: Change Dispense Screen

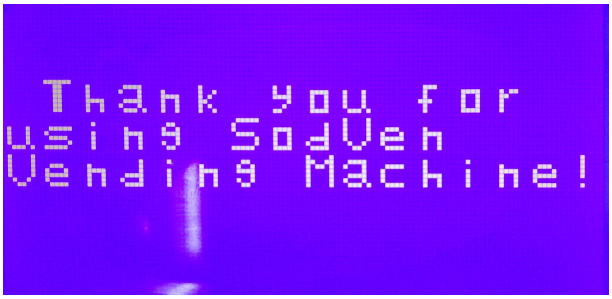

Fig 11: Thank You Screen

4.2 RTOS Implementation

Vending Machine can be implemented using two tasks *readInput()* and *displayOutput()*

4.2.1 readInput() Task

Handles all input switch press detection. Use Switch debouncing to accurately detect switch press. Since switch behaviour is different in different states, define switch behaviour accordingly in different states. Ex: UP Switch denotes Dime Enter in *COIN* state(Fig 1), but cancels transaction in *SELECT* state(Fig 5).

4.2.2 displayOutput() Task

Handles output screens to be displayed on GLCD corresponding to different states. Also handles LED output in *DELIVERY* state. Ex: Coin Enter Screen

in *COIN* state(Fig 1), and Select Soda Screen in *SELECT* state(Fig 5) are state-dependent.

5 Procedure

1. Include all the relevant header files in your code. Ensure that the following header files are present:

```
#include "Console/console.h"
#include "Console/glcd.h"
#include "Images/images.h"
```

   The above libraries contain direct-to-GLCD font libraries and certain GUI graphic premade. The libraries can be downloaded from here. Direct functions may be used to display for example, a coin, onto the screeen. Go through the headers to know more.

2. All necessary initialization, including Timers, Interrupts, Clock, GPIO, etc. can be performed using function _init_() from console.h. Edit console.h to initialize other peripherals in addition to the ones provided.

3. Initialize each of the various states using enumeration as in the following example:

```
enum vm_modes{

    // States for Vending Machine

    INITM,
    COIN,
    SELECT,
    DELIVERY
};
// Initialization
enum vm_modes vm_mode = INITM;
```

4. Declare two tasks(empty), with corresponding semaphores *readInput()* and *displayOutput()*.

5. Use Timer 2A Interrupt for task scheduling and post semaphores at appropriate instances in time.

6. Now, in each task, implement the overall Vending Machine state machine, as in the statechart description given in Section 4(Switch Case or State Table Implementation can be used).

6 Demo and Submission:

- Implement appropriate statechart within each task and use appropriate task scheduling in RTOS.
- Show the output to TA and explain the working.

All The Best