Estudo de caso: Controlador de uma máquina de venda de refrigerantes

Rebecca Quintino Do Ó[1]

[1]UFSC

October 4, 2023

1 Introdução

O projeto seguinte apresenta a implementação de uma máquina de refrigerante descrita em linguagem de programação C++ utilizando uma máquina de estados finita (FSM). A máquina fornece dois tipos de refrigerantes o MEET e ETIRPS, estes estão disponíveis para a escolha do usuário seguindo a seguinte a tabela mostrada na Figure 1. Esta tabela representa o comportamento desejado da máquina de estados, sendo os estados atuais, o estado em que a pessoa se encontra, definido pelo número de moedas que a pessoa inseriu. Além disso, temos os comandos que representam o valor da moeda inserida (botões) ou alguma chave ativada (sendo essas de confirmação ou devolução). Alguns dos comportamentos relevantes são:

- Ambos refrigerantes custam R$ 1.50 e caso o valor seja ultrapassado (i.e. inserido mais moedas), a quantidade é automaticamente estornada;
- Existe um botão para devolução do dinheiro inserido na máquina como também de confirmação do produto escolhido;
- O usuário só pode inserir moedas de: R$ 0.25, R$ 0.50 e R$ 1.00;
- O programa deve ser executado em ambiente Linux e no processador LEON3;
- Para o ambiente do LEON3 as mensagens do usuário serão visualizadas através do display OLED e os comandos serão enviados através dos botões e chaves que se encontram na placa. Para o ambiente Linux, os comandos de entrada e saída serão realizados através do terminal.

Estado atual	Comandos							
	Nada	M025	M050	M075	M100	DEV	MEET	ETIRPS
S000	S0	S025	S050	S075	S100	S000, D000	S0	S0
S025	S025	S050	S075	S100	S125	S000, D025	S025	S025
S050	S050	S075	S100	S125	S150	S000, D050	S050	S050
S075	S075	S100	S125	S150	S150, D025	S000, D075	S075	S075
S100	S100	S125	S150	S150, D025	S150, D050	S000, D100	S100	S100
S125	S125	S150	S150, D025	S150, D050	S150, D075	S000, D125	S125	S125
S150	S150	S150, D025	S150, D050	S150, D075	S150, D100	S000, D150	S150	S150

Figure 1: Diagrama de estados da máquina de refrigerante

2 Funcionamento

A seguir estarão descritos os comandos que podem ser ativos na placa para realizar o funcionamento da máquina de refrigerantes:

- Adicione moedas de R$ 0.25 pressionando o botão BTNU;
- Adicione moedas de R$ 0.50 pressionando o botão BTNL;
- Adicione moedas de R$ 1.00 pressionando o botão BTND;
- Selecione a opção ETIRPS com o botão BTNR e a opção MEET com o botão BTNC;
- Para confirmar o refrigerante ative a chave SW0, caso já tenha selecionado o refrigerante e tenha um saldo de R$ 1.50;
- Se quiser efetuar a devolução ative a chave SW1;

3 Arquivos

Estão presentes os seguintes arquivos no projeto:

- maquina_refrigerante.cpp e maquina_refrigerate.h
- estados.cpp e estados.h

- comandos.cpp e comandos.h
- Oled_classes.cpp e Oled_classes.h
- Oled.cpp e Oled.h
- main.cpp
- Makefile

3.1 maquina_refrigerante

Os arquivos *maquina_refrigerante* é a implementação de mais alto nível, que representará a maquina de estados presente na Figure 1, por meio da classe Maquina descrita no digrama de classes da Figura 2.

3.2 comandos

Os arquivos *command* contém a declaração da classe *Comando*, como também a enumeração dos "Comandos" presentes na Figure 1 para posteriormente relacionar esses comandos com os botões e chaves da placa. Primeiro, ao ser inicializado um objeto Comando ele começará com o comando "NADA". Posteriormente, no método *comando_string()* teremos a implementação das mensagens para cada comando escolhido. Por outro lado, o método *wait_for_comando* será o responsável por mapear os botões e chaves com seus respectivos comandos.

3.3 estados

Nos arquivos *estado* é utilizado o conceito de *Classes Bases Virtuais* no qual a classe mãe *Estado* tem suas filhas, que são os "Estados atuais" descritos na Figure 1. A implementação no .cpp é um pouco mais elaborada neste caso, pois para cada "estado filho" foi descrito o funcionamento da máquina de estados para cada "Comando" escolhido. O diagrama de classes na Figure 3 contemplará mais detalhes sobre os métodos e atributos utilizados para essa implementação. Resumidamente, em cada "Estado atual" teremos os métodos *estado_string()*, *init()*, *exit()*, *proximo_estado()*, *estado_instance()*:

- Em *estado_string()* é retornado o valor do saldo.
- Em *init()* Oled será apagado e posteriomente será mostrado o saldo, a moeda escolhida e o troco referentes aquele estado.
- Em *exit()* definirá como será o estado depois dele finalizado.

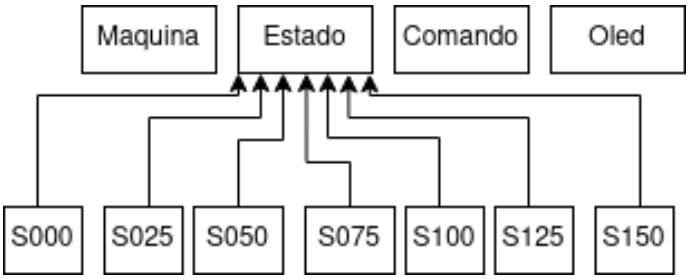

Figure 2: Diagrama de classes da máquina de refrigerante

- Em *proximo_estado()* seŕa onde a implementação da FSM ocorrerá. O próximo estado será definido de acordo com o comando selecionado na placa pelo cliente.
- Em *estado_instance()* é definido um objeto para cada "Estado atual", assim não teremos que criar e excluir um objeto toda vez que mudarmos de estado.

3.4 Oled

Temos os arquivos relacionados ao funcinamento do Oled. Além dos arquivos da biblioteca Oled disponibilizados pelo professor, foi criado outros dois arquivos *Oled_classes* para a definição da classe Oled, afim de facilitar o uso do init(), clear() e print_display() posteriormente.

3.5 main

O *main* começa ativando o init() e o clear() implementados no *Oled_classes*. Posteriormente será feita a atualização da Máquina de refrigerantes, esperando sempre um novo comando quando este for inserido através da placa.

3.6 Makefile

Por fim o *Makefile* foi criado a fim de facilitar a compilação do código na hora dos testes.

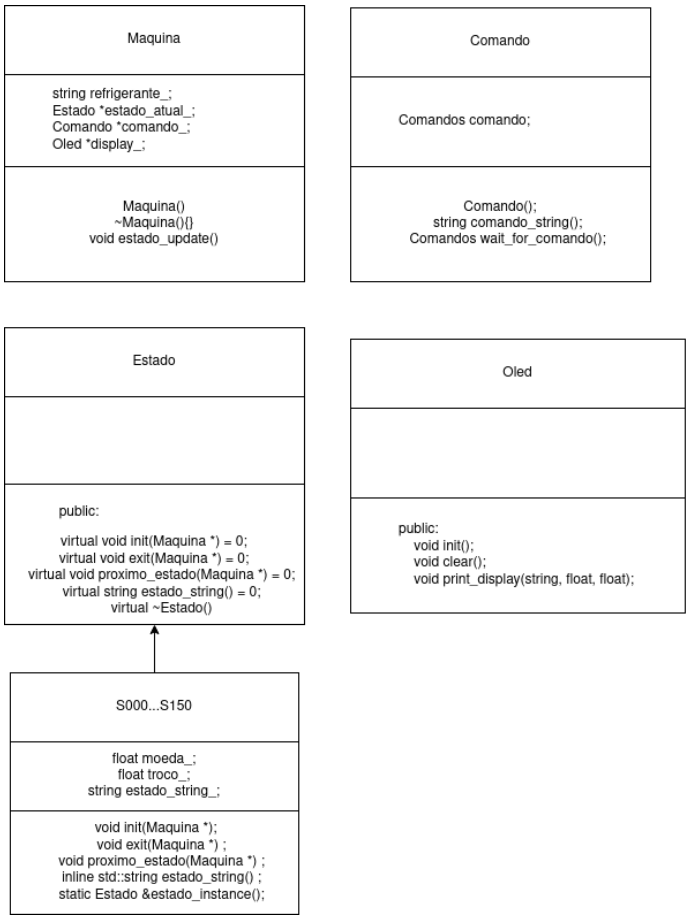

Figure 3: Diagrama de classes da máquina de refrigerante