Universidade Tecnológica Federal do Paraná
Câmpus Curitiba
Departamento de Engenharia Eletrônica
EL68E - Sistemas embarcados
Prof. Hugo Vieira Neto

Documentação de projeto

Renato Böhler
Everton Barreto
André Cassimiro

Curitiba, PR
14 de junho de 2018

SUMÁRIO

1. Arquitetura funcional do sistema

O diagrama de blocos funcional do sistema está representado na Figura 1.1. O sistema a ser controlado será simulado pelo software SimSE2 v0.94[1], que simula o aspecto físico do sistema de elevadores, num prédio com 15 andares e com 3 elevadores independentes. O simulador estará rodando em um computador convencional, que se comunicará via porta serial COM virtual sobre USB com a placa TIVA TM4C1294, onde será implementada a solução de controle do sistema de elevadores.

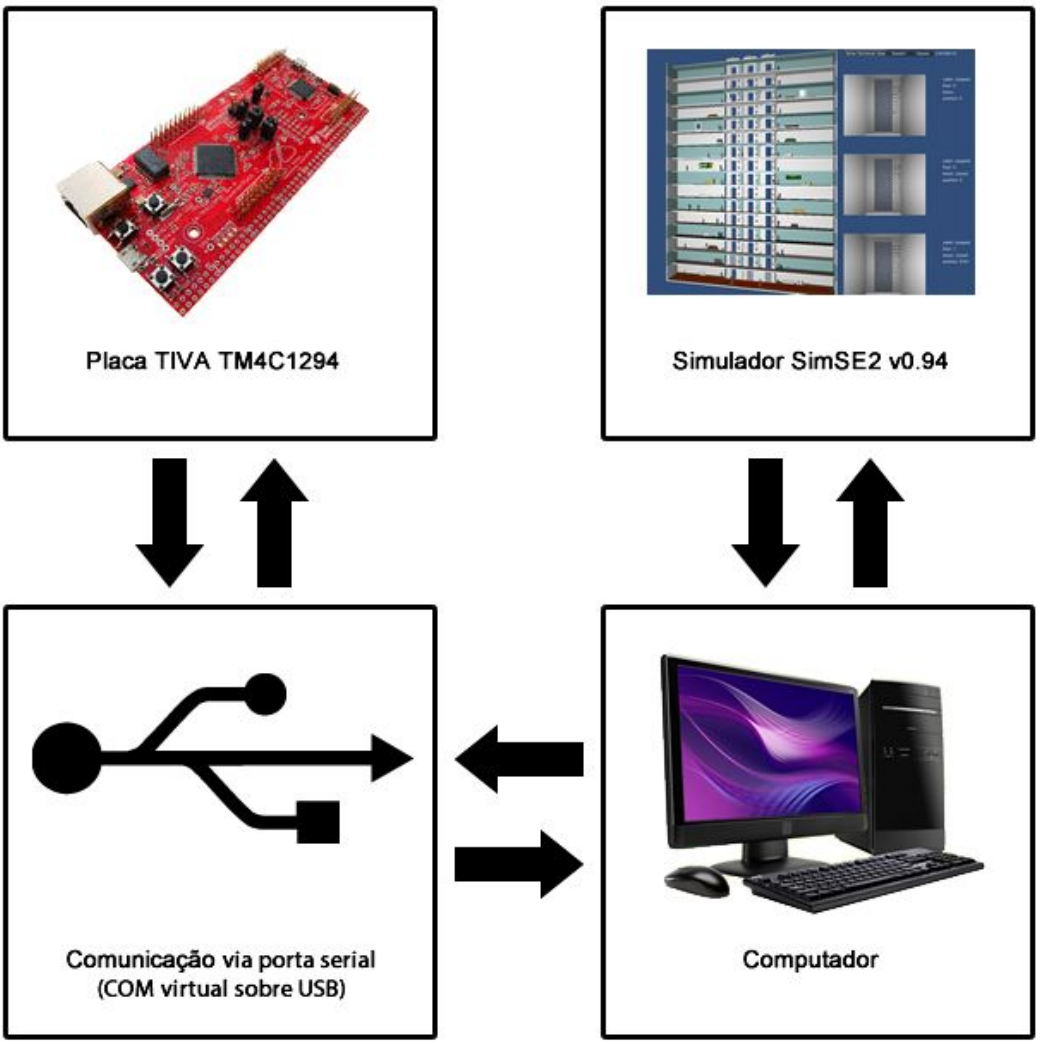

Figura 1.1 - Diagrama funcional do sistema

[1] http://dainf.ct.utfpr.edu.br/~douglas/index_files/Page392.htm

2. Arquitetura física do sistema

O diagrama de blocos do sistema a ser implementado na placa TIVA TM4C1294 está representado na Figura 2.1:

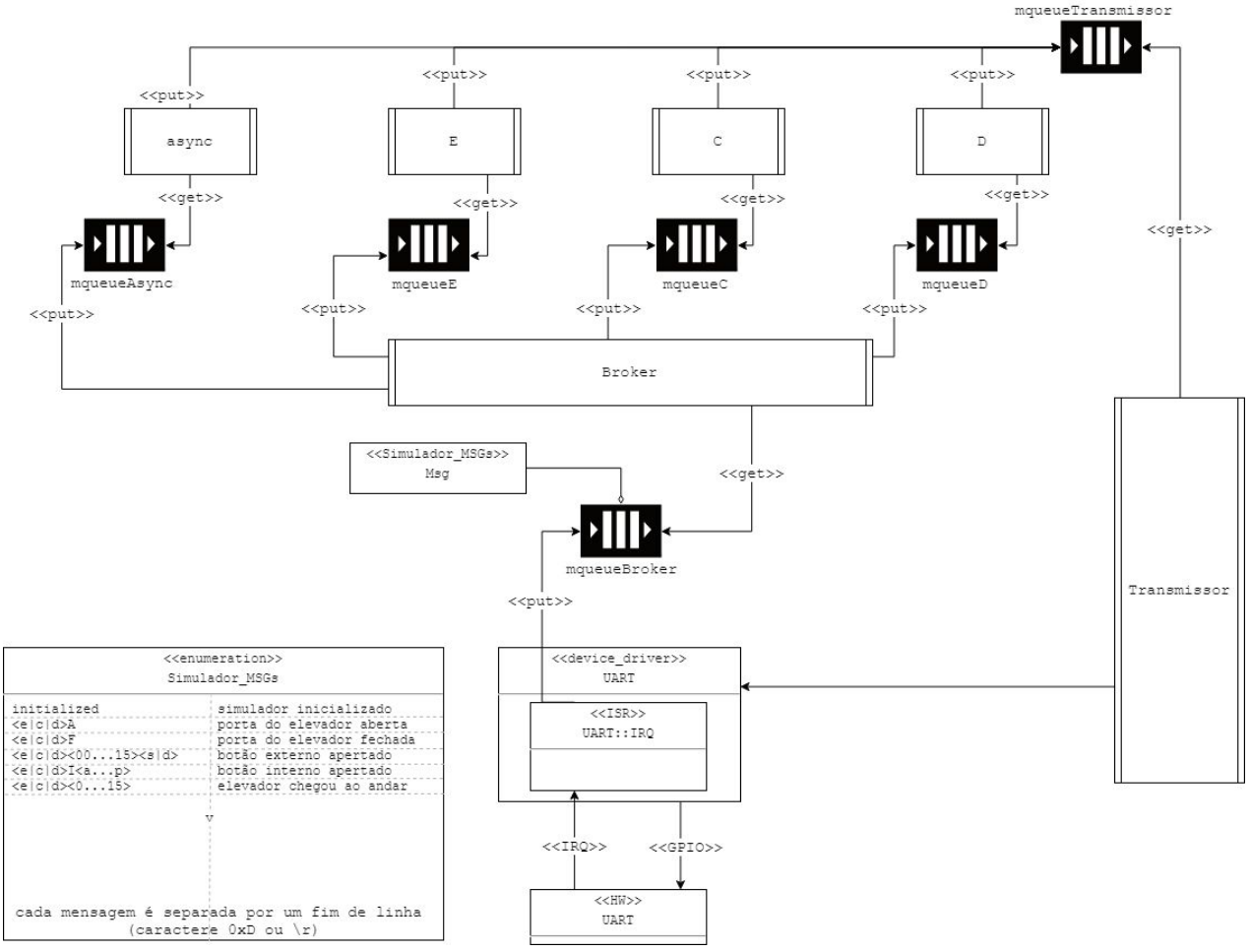

Figura 2.1 - Diagrama de blocos do sistema

Cada bloco do diagrama está relacionado à execução de uma atividade em específico. Neste diagrama também são apresentadas filas de correspondências (mail queues[2]), que serão utilizadas para a comunicação entre as tarefas.

[2] http://www.keil.com/pack/doc/CMSIS/RTOS/html/group__CMSIS__RTOS__Mail.html

- **UART:** biblioteca que gerencia o periférico UART presente no microcontrolador Cortex e executa a comunicação via serial entre o sistema de elevadores e o sistema de controle.
- **UART::IRQ:** interrupção que será gerada ao receber uma mensagem transmitida via serial. Sua função é receber estas mensagens e colocá-las na fila de correspondências `mqueueBroker` para que o Broker as consuma.
- **Broker:** tarefa cuja função consumir as correspondências da fila `mqueueBroker` e direcioná-las para os blocos que possuem interesse nesta dada mensagem, colocando-as nas respectivas filas de correspondências (`mqueueAsync`, `mqueueE`, `mqueueC` e/ou `mqueueD`).
- **async:** tarefa cuja função é consumir as correspondências da fila `mqueueAsync` e realizar a devida resposta colocando uma correspondência na fila `mqueueTransmissor` para que o Transmissor a consuma. O único caso em que isto se faz necessário é no acionamento do indicador de botão interno.
- **E:** tarefa cuja função é consumir correspondências relacionados ao elevador da esquerda da fila `mqueueE`, e colocar correspondências na fila `mqueueTrasmissor` para que o Transmissor as consuma, de modo a realizar o devido controle do elevador.
- **C:** tarefa cuja função é consumir correspondências relacionados ao elevador do centro da fila `mqueueC`, e colocar correspondências na fila `mqueueTrasmissor` para que o Transmissor as consuma, de modo a realizar o devido controle do elevador.
- **D:** tarefa cuja função é consumir correspondências relacionados ao elevador da direita da fila `mqueueD`, e colocar correspondências na fila `mqueueTrasmissor` para que o Transmissor as consuma, de modo a realizar o devido controle do elevador.
- **Transmissor:** tarefa cuja função é consumir as correspondências colocadas na fila `mqueueTransmissor` pelas tarefas `async`, `E`, `C` e `D` para realizar a transmissão via serial das mensagens, utilizando o bloco da `UART`.

3. Projeto dos componentes

Neste capítulo será apresentada a implementação planejada para os principais componentes do sistema descrito no capítulo anterior.

3.1. UART::IRQ

O manipulador de interrupções vindas da UART terá como responsabilidade a recepção das mensagens vindas do sistema de elevadores através da comunicação serial e a colocação destas mensagens na fila de correspondências `mqueueBroker`. O fluxograma que representa a implementação deste manipulador de interrupções é mostrado na Figura 3.1.1.

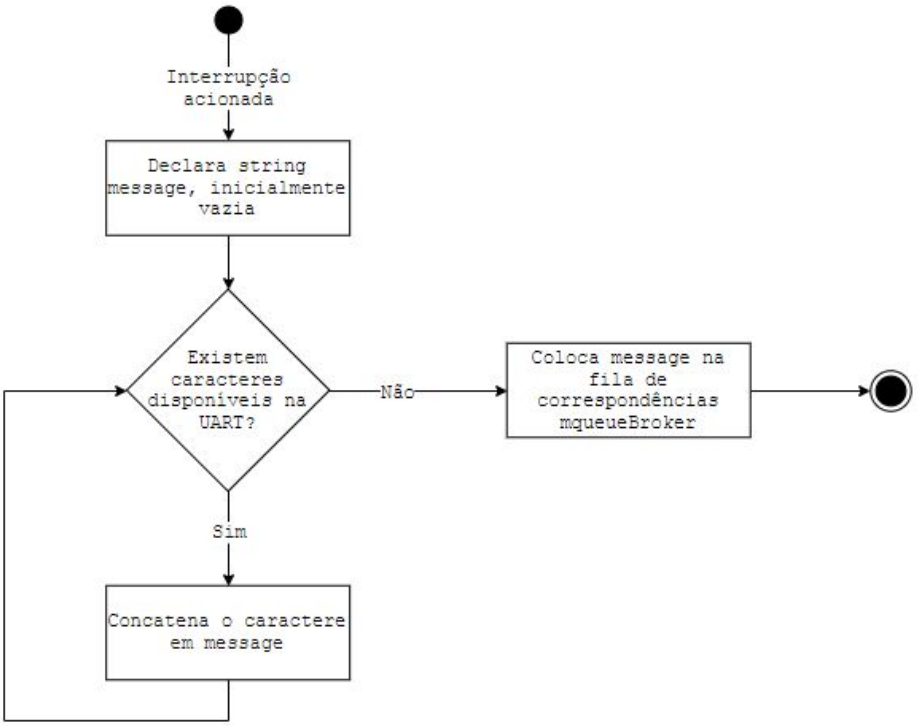

Figura 3.1.1 - Fluxograma da implementação de `UART::IRQ`

3.2. Broker

Esta tarefa é responsável pelo correto endereçamento das correspondências da fila `mqueueBroker` para as filas `mqueueAsync`, `mqueueE`, `mqueueC` e `mqueueD`. As correspondências vindas de `mqueueBroker` também podem precisar ser separadas em duas ou mais, pois cada caractere `0xD` recebido pela UART divide duas mensagens distintas. O fluxograma que representa a implementação desta atividade é mostrado na Figura 3.2.1.

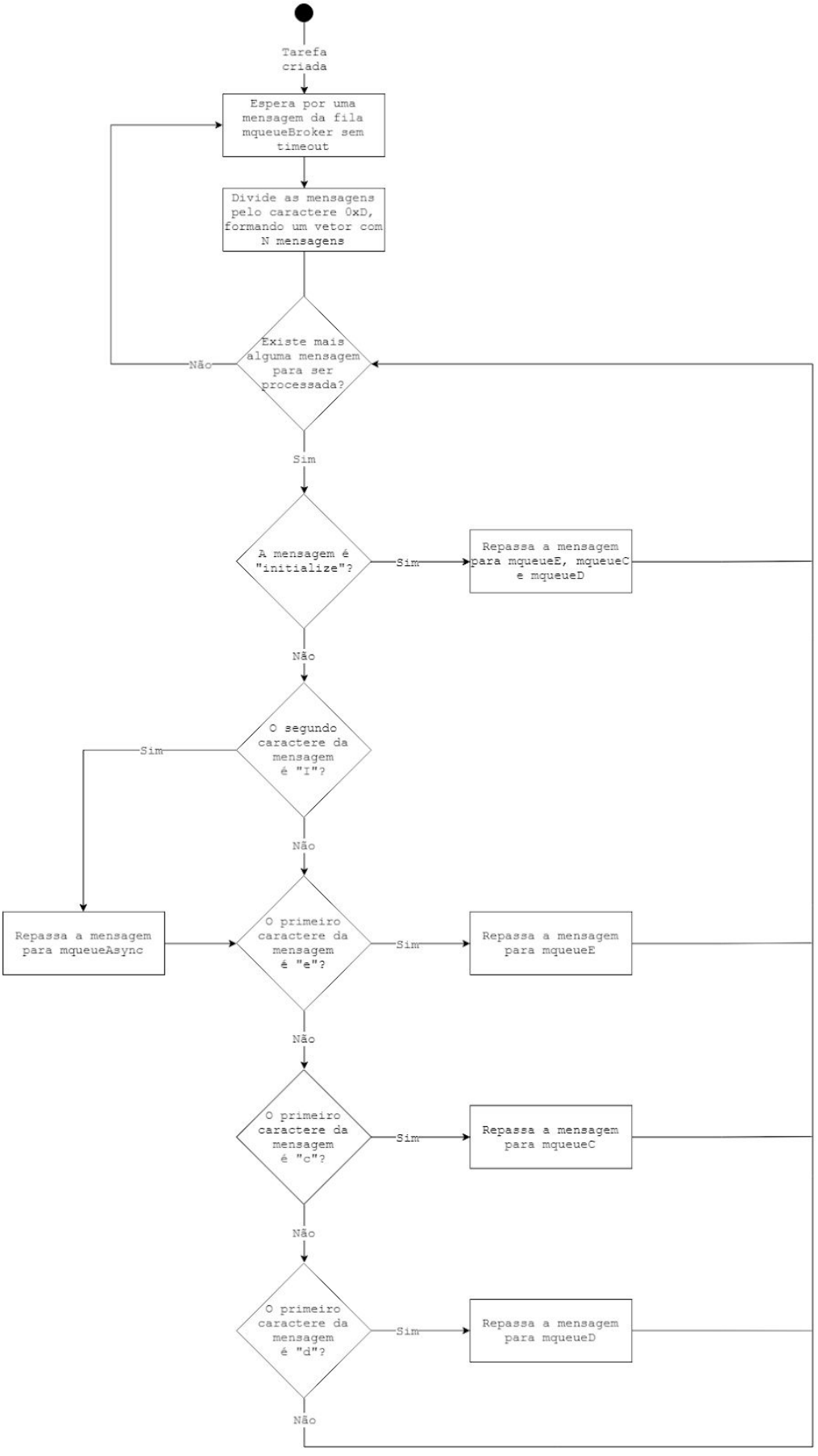

Figura 3.2.1 - Fluxograma da implementação de `Broker`

3.3. async

Esta tarefa é responsável por consumir as correspondências da fila `mqueueAsync` e repassar a correspondência correta para `mqueueTransmissor`. O fluxograma que representa a implementação desta atividade é mostrado na Figura 3.3.1.

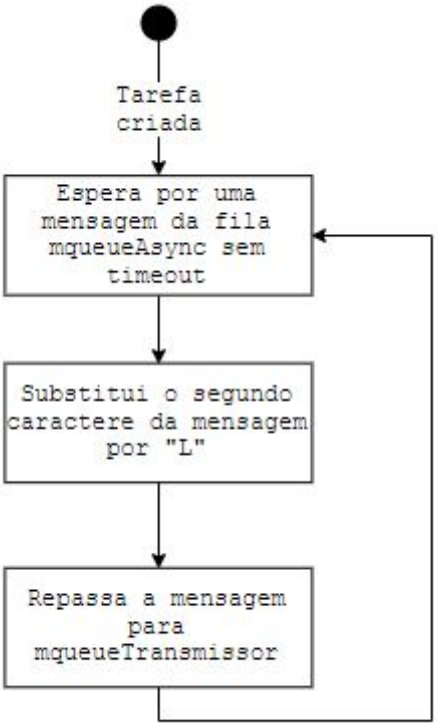

Figura 3.3.1 - Fluxograma da implementação de `async`

3.4. Tarefas dos elevadores

As tarefas dos elevadores possuirão a mesma lógica, sendo 3 instâncias de uma mesma *thread*. Esta tarefa será responsável por consumir as mensagens da fila do elevador correspondente (`mqueueE`, `mqueueC` ou `mqueueD`), manter variáveis de estado do elevador atualizadas e manter atualizada uma máquina de estados para poder enviar a `mqueueTransmissor` as mensagens corretas para o cumprimento dos requisitos do sistema.

A implementação desta tarefa se dará em 3 etapas:

1. Espera por uma mensagem vinda da fila correspondente;
2. Atualização das variáveis de estado do elevador;
3. Atualização da máquina de estados do elevador.

A primeira etapa é autoexplicativa: a tarefa estará esperando por correspondências na sua fila.

A segunda etapa consiste na análise da mensagem recebida e na atualização das variáveis de estado do elevador. As variáveis de estado a serem mantidas atualizadas são:

- `andar`: de 0 a 15, corresponde ao andar no qual o elevador está;
- `porta`: indica se a porta está aberta ou fechada;
- `destinos_internos:` indica quais destinos foram requisitados pelo painel de botões internos e ainda não foram atendidos;
- `destinos_sobe:` indica quais destinos foram requisitados pelos painéis de botões externos de cada andar, com o objetivo de subida, e ainda não foram atendidos;
- `destinos_desce:` indica quais destinos foram requisitados pelos painéis de botões externos de cada andar, com o objetivo de descida, e ainda não foram atendidos;
- `chegou_em_andar:` indica que a última mensagem recebida informou que o elevador chegou em algum andar.

A máquina de estados a ser atualizada na terceira etapa é exibida na Figura 3.4.1. Nos comentários, a função *send* seria equivalente a colocar a mensagem passada por argumento para `mqueueTransmissor`, e o caractere `x` seria o identificador do elevador (`e`, `c` ou `d`). Todos os comentários informam ações que devem ser tomadas em uma transição de um dado estado.

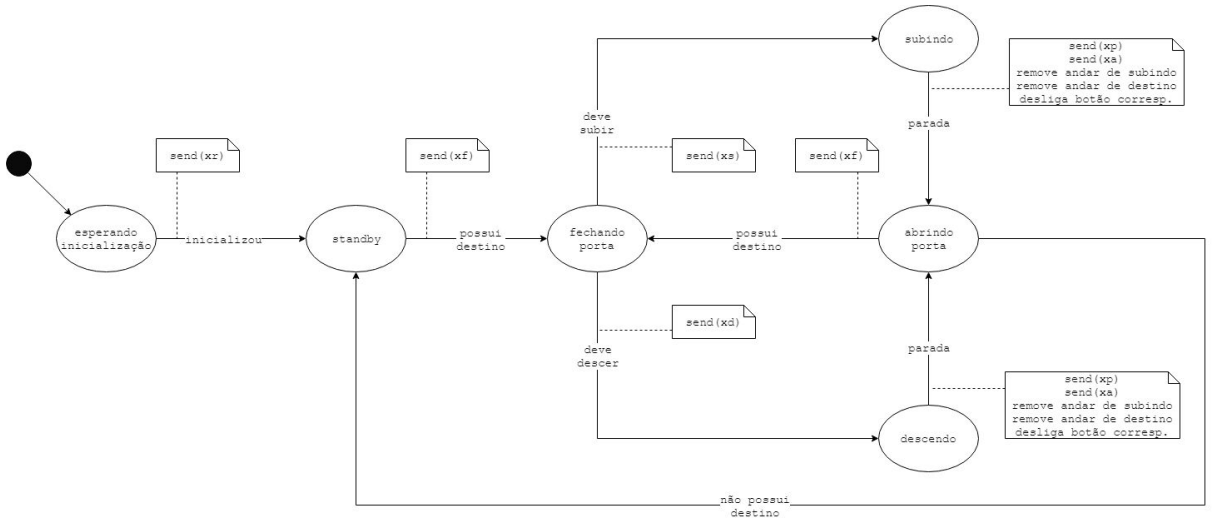

Figura 3.4.1 - Máquina de estados

3.5. Transmissor

Esta tarefa é responsável por consumir as correspondências da fila `mqueueTransmissor` e simplesmente transmiti-las via serial pelo uso da UART. O fluxograma que representa a implementação desta atividade é mostrado na Figura 3.5.1.

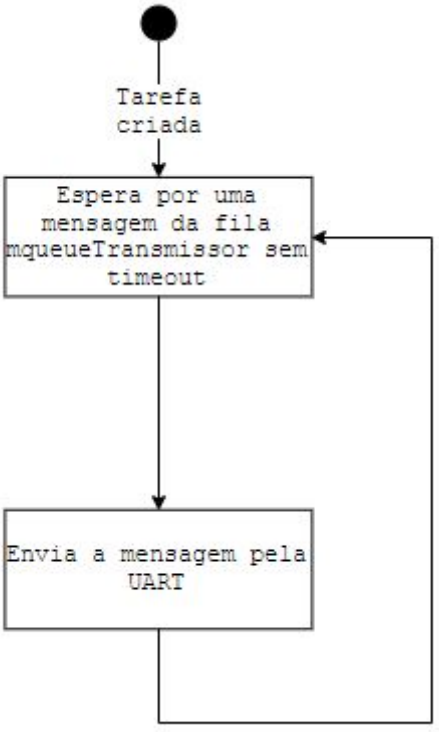

Figura 3.5.1 - Fluxograma da implementação de `Transmissor`