Engenharia de Telecomunicações — Prof. Emerson Ribeiro de Mello

STD29006 – Sistemas Distribuídos — http://docente.ifsc.edu.br/mello/std

Projeto prático 1: Captura de bandeiras com robôs autônomos

14/07/2021

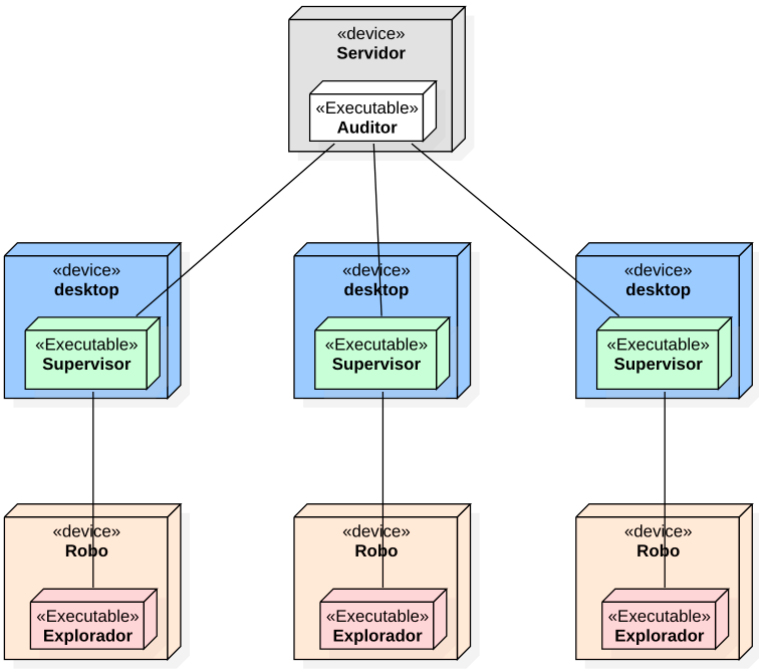
- Na raiz do repositório crie o arquivo `.gitignore` para ignorar arquivos desnecessários para esse projeto. O conteúdo desse arquivo poderá ser gerado por meio do site https://gitignore.io.
- Na raiz do repositório crie o arquivo `Readme.md` e coloque nele uma breve descrição sobre o projeto, as instruções para compilar e para executar as aplicações desse projeto.
- Indique no `Readme.md` se todas funcionalidades foram implementadas e quais não foram.

1 Descrição

A Figura 1 apresenta um diagrama de implantação UML que contém todos os componentes de software, bem como suas disposições, para um jogo de captura de bandeiras com robôs autônomos. A solução conta com um software aplicativo chamado "**Sistema Auditor**" e o mesmo fica em execução em um equipamento servidor na nuvem, com endereço IP público, fixo e conhecido por todos.

Cada competidor é responsável por desenvolver dois softwares aplicativos: (1) **sistema supervisor** – que fica em execução em um *laptop* e que obtém endereço IP dinâmico, por exemplo, por DHCP; (2) **sistema explorador** – que fica em execução em um robô que possui um interface WiFi a qual também obtém IPs dinâmicos por meio do DHCP.

Figura 1: Diagrama de implantação da solução

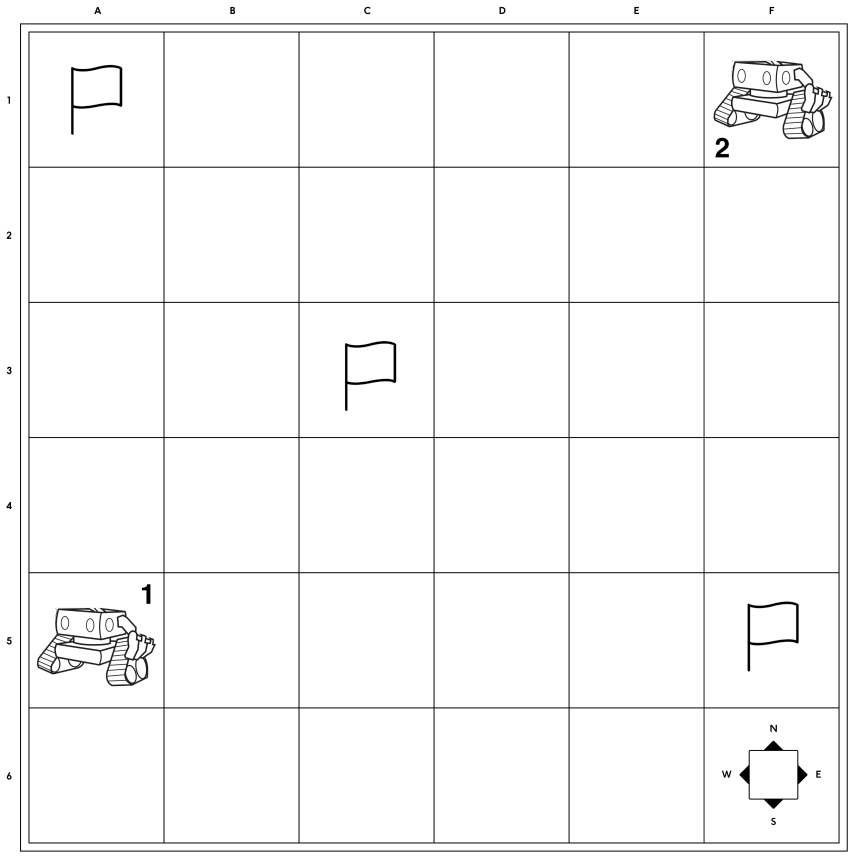


1.1 Dinâmica do jogo

O **sistema auditor** sabe quais são os **sistemas supervisores** que estão ativos, pois em um passo anterior todos os **sistemas supervisores** avisaram-no. Da mesma forma, cada **sistema supervisor** sabe se seu respectivo **robô** está ativo.

No início da partida o **sistema auditor** envia para todos os **sistemas supervisores**, o mapa a ser explorado, o qual indica o tamanho da área, onde está cada bandeira e a posição inicial de cada robô no mapa. O mapa é formado por coordenadas (x,y) e cada robô ou bandeira está em uma coordenada. Na Figura 2 é ilustrada uma instância do jogo com dois robôs exploradores e três bandeiras. O deslocamento do robô sempre ocorre em unidades alterando o valor x ou y a cada vez. Por exemplo, se o robô está na coordenada (A,5) e solicitado que este se desloque para a coordenada (A,1), então o menor caminho para isso consistiria em passar pelas coordenadas (A,4), (A,3) e (A,2), chegando assim na coordenada (A,1).

Figura 2: Instância de um mapa a ser explorado com dois robôs exploradores



Ao receber o mapa, o **sistema supervisor** envia para o **robô** um comando para indicar qual é sua posição inicial na partida. Por exemplo, o robô 1 na Figura 2 saberá que sua posição inicial é (A,5). O **sistema supervisor** avisa o **sistema auditor** que seu robô encontra-se na posição inicial e está pronto para iniciar a partida. O **sistema auditor** aguarda receber a confirmação de todos os **sistemas supervisores** e assim que receber, avisa todos os **sistemas supervisores** que eles já podem começar a explorar o mapa.

No próximo passo, o **sistema supervisor** envia as coordenadas da próxima bandeira a ser capturada para seu robô. Esse inicia a movimentação e avisa seu supervisor sempre que estiver uma nova coordenada e faz isso até chegar na coordenada destino. Por fim, o **supervisor** avisa o **sistema auditor** que seu robô capturou a bandeira na coordenada (X,Y). O **sistema auditor**, avisa todos os **sistemas supervisores** que a bandeira na posição (X,Y) foi capturada pelo `robô N`. Assim,

os demais robôs poderão ir em busca de outras bandeiras.

2 Solução a ser desenvolvida

Desenvolva o sistema auditor, sistema supervisor e sistema do robô de forma a permitir a comunicação explicitada na Subseção 1.1. Deve-se considerar que **sistema supervisor** e **robô** não possuem IPs estáticos e conhecidos. O único equipamento que possui IP público, estático e conhecido é o **servidor** onde o **sistema auditor** ficará em execução. O **equipamento servidor** pode ter outros serviços em execução, se julgar necessário.

O **sistema auditor** não deve possuir uma interface que permita a interação com o usuário e deverá iniciar uma partida automaticamente assim que todos os sistemas supervidores indicarem que estão prontos. Assim, ao iniciar o processo do sistema auditor deve-se informar quantos sistemas supervidores são esperados para a partida (faça por meio de argumentos de linha de comando). O **sistema supervidor** e o **robô explorador** não terão interface interativa com o usuário, porém deverão imprimir em seus respectivos consoles, todas mensagens trocadas e as ações que foram executadas após receber uma mensagem ou para enviar uma mensagem.

A exploração a ser feita pelo **robô** deve ser simulada, isto é, pode-se fazer o **robô** dormir por alguns segundos para simular sua movimentação pelo mapa até chegar na coordenada destino. Assim que todas as bandeiras forem capturadas, o **sistema auditor** envia para todos os **supervisores** o total de bandeiras que cada equipe conseguiu capturar. O supervisor deverá imprimir o resultado da partida e encerrar o processo.

- A solução deverá obrigatoriamente ser construída sobre contêineres Docker. Sendo assim, no repositório é esperado um arquivo `docker-compose.yml` e, se necessário, os arquivos `Dockerfile` para cada sistema (auditor, supervisor e robô);
  - Em um cenário com 2 jogadores serão necessários no mínimo 5 contêineres (auditor, supervisor 1, robô 1, supervidor 2 e robô 2).
- A solução pode ser desenvolvida nas linguagens C, Java ou Python3;
- A solução pode ser desenvolvida com sockets (inclusive o ZMQ), gRPC, RMI ou fila de mensagens (RabbitMQ);
  - Se fizer uso do RabbitMQ, então este deverá ser executado em um contêiner próprio.
- A solução a ser proposta poderá fazer uso do serviço de nomes presente no Docker para descoberta e resolução dos endereços IPs dos equipamentos.

Data para entrega: Até o dia 15/08/2021 via Github Classroom.