Caso de uso: Detecção de Objetos com Ultrassom

|  |  |
| --- | --- |
| **Ação do Usuário** | **Reação do Sistema** |
| Detectar objetos | Solicitar dados da serial,  Solicitar dados de posicionamento,  Processar objs e coordenadas |
| Obter dados brutos | Ler serial,  Formatar pacote, |
| Calcular posicionamento | Definir margem de mapeamento,  Calcular velocidade do som,  Calcular distancia,  Solicitar identificação de objetos |
| Identificar objetos | Aproximar coordenadas,  Rotular coordenadas |
| Fechar processo | - |

Tabela WARC

|  |  |  |
| --- | --- | --- |
| **Workflow: Detecção de objetos com ultrassom** | | |
| **Ação/Reação** | **Responsável** | **Colaboradores** |
| Detectar objetos | Classe\_Controlador | - |
| Solicitar dados da serial | Classe\_Controlador | Classe\_Comunicacao |
| Solicitar dados de posicionamento | Classe\_Controlador | Classe\_Posicionamento |
| Processar objs e coordenadas | Classe\_Controlador | - |
| Obter dados brutos | Classe\_Comunicacao | Classe\_Serial |
| Ler serial | Classe\_Comunicacao | Classe\_Serial |
| Formatar pacote | Classe\_Comunicacao | - |
| Calcular posicionamento | Classe\_Posicionamento | - |
| Definir margem de mapeamento | Classe\_Posicionamento | - |
| Calcula velocidade do som | Classe\_Posicionamento | - |
| Calcula distancia | Classe\_Posicionamento | - |
| Solicitar identificação de objetos | Classe\_Posicionamento | Classe\_OBJ\_Detec |
| Identificar objetos | Classe\_OBJ\_Detec | - |
| Aproximar coordenadas | Classe\_OBJ\_Detec | - |
| Rotular coordenadas | Classe\_OBJ\_Detec | - |

Contratos - Abaixo segue os contratos de cada ação e reação definida anteriormente

|  |
| --- |
| **Contrato:** Detectar objetos |
| **Pré-requisitos: -** |
| **Termos:** Esta ação será responsável por efetuar as chamadas dos métodos responsáveis pela leitura e processamento dos dados. No final deste processo, este método deve efetuar a chamada de outro método que possibilite a visualização do resultado final. É necessário definir um método dentro da Classe\_Controlador para corresponder a esta ação. |
| **Pós-requisitos: -** |

|  |
| --- |
| **Contrato:** Solicitar dados brutos da serial |
| **Pré-requisitos:** Porta serial utilizada para a comunicação e *Baud rate*. |
| **Termos:** Consiste em um método dentro da Classe\_Controlador que será responsável pela obtenção dos dados brutos vindo dos sensores. Esta operação deve criar um objeto da Classe\_Comunicacao e solicitar a leitura dos dados brutos da serial. |
| **Pós-requisitos:** Dados brutos no formato de lista. |

|  |
| --- |
| **Contrato:** Solicitar dados de posicionamento |
| **Pré-requisitos:** Dados brutos dos sensores. |
| **Termos:** Esta operação consiste em instanciar um objeto da Classe\_Posicionamento que deve ter como parâmetros de entrada os dados brutos dos sensores. Essa reação consiste em um método dentro da Classe\_Controlador que deve solicitar o tratamento dos dados brutos dos sensores para a Classe\_Posicionamento. |
| **Pós-requisitos:** Dados já processados. |

|  |
| --- |
| **Contrato:** Processar objs e coordenadas |
| **Pré-requisitos:** Dados dos sensores processados pela Classe\_Posicionamento. |
| **Termos:** Esta operação consiste em criar uma forma de visualização dos dados, preferencialmente com alguma biblioteca como o matplotlib ou Kivy. Deve possuir uma visualização limpa, dinâmica e estar presente como um método da Classe\_Controlador. |
| **Pós-requisitos: -** |

|  |
| --- |
| **Contrato:** Obter dados brutos |
| **Pré-requisitos:** Objeto da Classe\_Comunicacao iniciado com a porta serial utilizada para a comunicação e *Baud rate*. |
| **Termos:** Esta operação deve conter a chamada dos metodos de leitura da serial e pré-processamento. |
| **Pós-requisitos:** Dados pré-processados para a Classe\_Controlador. |

|  |
| --- |
| **Contrato:** Ler Serial |
| **Pré-requisitos: -** |
| **Termos:** Esta operacao consiste em efetuar a leitura dos dados brutos vindo dos sensores. Deve estar contido dentro da Classe\_Comunicacao e iniciar uma comunicacao com a serial. |
| **Pós-requisitos:** Dados brutos para pré-processamento. |

|  |
| --- |
| **Contrato:** Formatar pacote |
| **Pré-requisitos:** Deve possuir como entrada os dados brutos obtidos. |
| **Termos:** Esta operação está contida na Classe\_Comunicacao. Ela realiza o pré-processamento dos dados agrupando-os em um dicionario e separando os dados brutos por sensores. Ex: {“Sensor 1”: [{“angulo”: 89.0, “temp”: 25.0}]}. |
| **Pós-requisitos: -** |

|  |
| --- |
| **Contrato:** Calcular posicionamento |
| **Pré-requisitos:** Definir a distância entre os sensores. Deve ter como parâmetro os dados brutos pré-processados. |
| **Termos:** Esta operação pertence a Classe\_Posicionamento, e será responsável pelo cálculo das coordenadas dos objetos com base nas distâncias entre os sensores. |
| **Pós-requisitos:** Pares ordenados de cada objeto no formato de lista. |

|  |
| --- |
| **Contrato:** Definir margem de mapeamento |
| **Pré-requisitos:** Novas distâncias entre sensores |
| **Termos:** Esta operação pertence a Classe\_Posicionamento, sendo responsável por alterar as distâncias entre os sensores caso haja necesssidade. |
| **Pós-requisitos: -** |

|  |
| --- |
| **Contrato:** Calcula velocidade do som |
| **Pré-requisitos:** Deve receber como parâmetro a temperatura do ambiente. |
| **Termos:** Esta operação pertence a Classe\_Posicionamento e consiste no cálculo da velocidade do som de acordo com a temperatura do meio de propagação. |
| **Pós-requisitos:** Velocidade do som no formato float. |

|  |
| --- |
| **Contrato:** Calcula distancia |
| **Pré-requisitos:** Velocidade do som calculada pelo método calcula\_velocidade\_do\_som() e o tempo de resposta registrado em segundos. |
| **Termos:** Esta operação pertence a Classe\_Posicionamento e consiste em calcular a distância entre o sensor e o objeto detectado. |
| **Pós-requisitos:** Retornar a distância em metros no formato float entre o sensor e o objeto detectado. |

**CHECKPOINT**

|  |
| --- |
| **Contrato:** Solicitar identificação de objetos |
| **Pré-requisitos:** Lista de coordenadas. |
| **Termos:** Esta operação deve pertencer a Classe\_Posicionamento e instanciar um objeto da Classe\_OBJ\_Detec, onde sera responsavel pela chamada do metodo Identificar Objetos pertencente ao objeto da Classe\_OBJ\_Detec e em seguida deve retornar esses dados. |
| **Pós-requisitos:** Retornar uma lista com os objetos identificados. |

|  |
| --- |
| **Contrato:** Identificar objetos |
| **Pré-requisitos:** Este metodo deve receber como parametro as coordenadas ja processadas. |
| **Termos:** Esta operacao pertence a Classe\_OBJ\_Detec e deve ser responsavel por chamar os metodos aproximar\_coordenadas() e rotular\_coordenadas() e depois retornar o resultado. |
| **Pós-requisitos:** Deve retornar uma lista contendo o ID de cada objeto e suas respectivas coordenadas. |

|  |
| --- |
| **Contrato:** Aproximar Coordenadas |
| **Pré-requisitos:** Deve receber como parametro as coordenadas. |
| **Termos:** Este metodo pertence a Classe\_OBJ\_Detec, sendo responsavel por unir pontos que estejam muito proximos ao tirar a media entre eles, considerando-os como coordenadas de um so objeto. |
| **Pós-requisitos:** Retornar dados no formato lista de dicionario com as coordenadas ja processadas e sem pontos muito proximos. |

|  |
| --- |
| **Contrato:** Rotular coordenadas |
| **Pré-requisitos:** Deve receber como parametro as coordenadas ja processadas pelo metodo identificar\_pontos\_proximos() |
| **Termos:** Este metodo pertence a Classe\_OBJ\_Detec, sendo responsavel por rotular cada ponto identificado. |
| **Pós-requisitos:** Retornar as coordenadas ja rotuladas, no formato lista de dicionarios. |