Arquitetura HIL para teste de sistemas embarcados como *vehicle interface* de veículos autônomos baseados no Autoware

Projeto – Etapa 2

Gabriel Toffanetto França da Rocha
g289320@dac.unicamp.br

Juan Luis Barraza Ramirez
j272583@dac.unicamp.br

Professor Dr. Rodrigo Moreira Bacurau

IM420X – Projeto de Sistemas Embarcados de Tempo Real

Faculdade de Engenharia Mecânica
Universidade Estadual de Campinas

22 de outubro de 2024

Schedule

Introdução

Proposta

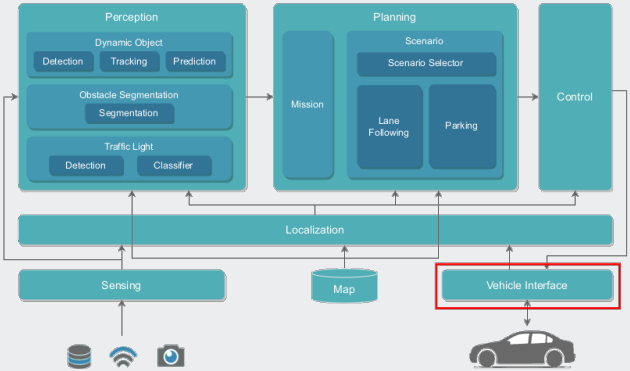


Figura 1: Escopo do projeto na arquitetura Autoware.

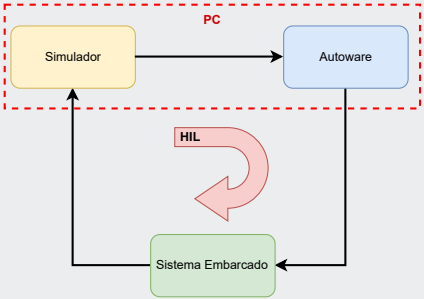


Figura 2: Arquitetura de teste do *hardware*.

Estados do sistema

Estados do sistema

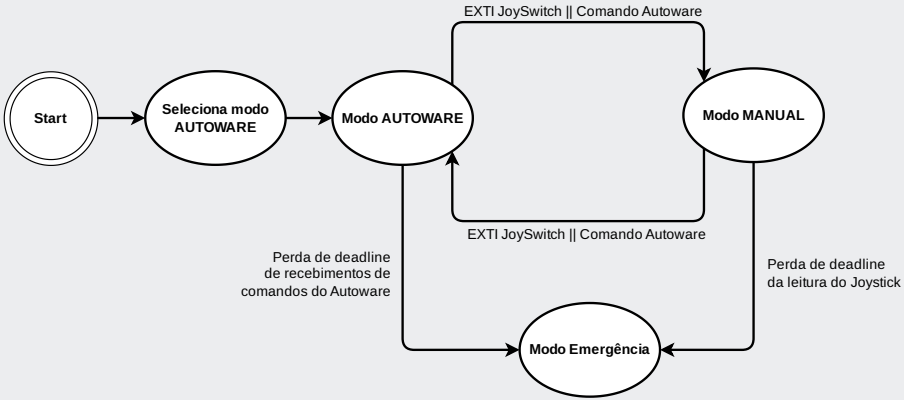


Figura 3: Máquina de estados do sistema.

Tarefas

Diagrama do sistema

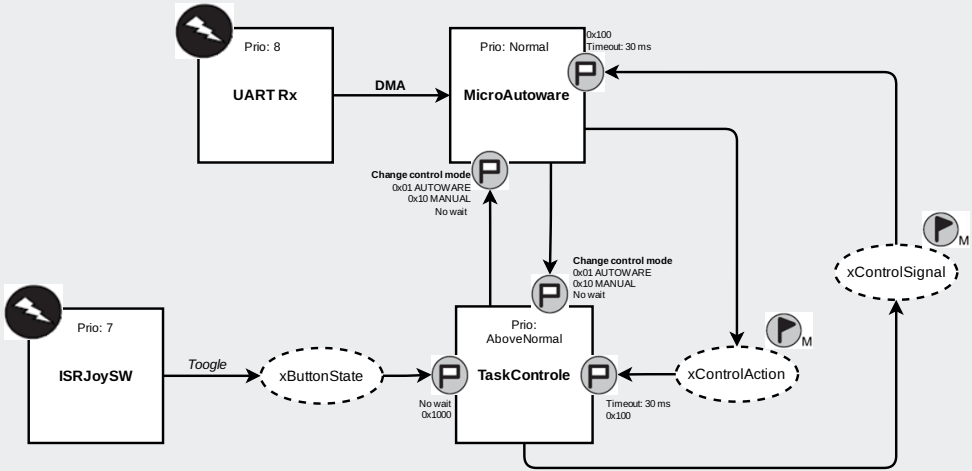


Figura 4: Diagrama do sistema.

Descrição das tarefas

Tarefa	
Nome	MicroAutoware
Prioridade	Normal
Tamanho da stack	3500 kB
Detalhes	Leitura dos *subscribers* Autoware, leitura dos *subscribers* CARLA, envio das informações de controle e modo de operação para a TaskControle, recebimentos das informações de controle da TaskControle, escrita dos *publishers* Autoware, escrita dos *publishers* CARLA.

Descrição das tarefas

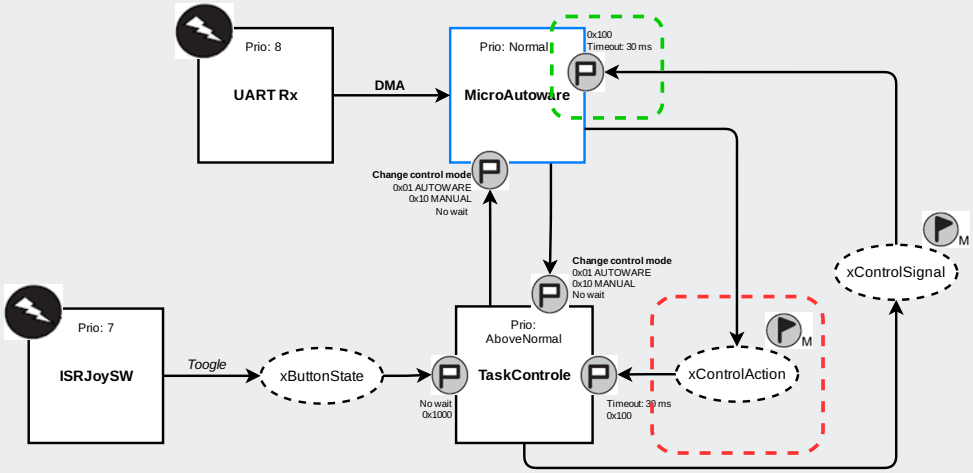


Figura 5: Elementos de bloqueio e desbloqueio da tarefa MicroAtuoware.

Descrição das tarefas

Tarefa

Nome	TaskControle
Prioridade	AboveNormal
Tamanho da stack	500 kB
Detalhes	Realiza o controle do veículo utilizando a referência dada pelo *joystick* ou pelo Autoware, dado o modo de operação, podendo ser MANUAL ou AUTOWARE, respectivamente. A alteração do modo é feita por *ThreadFlag*, gerada por ISR ou pelo Autoware. Em caso do modo de operação AUTOWARE, os sinais de controle são recebidos por variável global e sincronizados por *ThreadFlag*, com tempo de 30 ms, onde caso não receba, entra em algum modo de segurança. Em caso de operação MANUAL, o *joystick* é lido por DMA, aguardando 20 ms antes de cada leitura, convertendo os valores analógicos em sinais de controle, onde também caso haja algum erro, o modo de emergência é acionado. O sinal de controle é enviado para o MicroAutoware por uma variável global e sincronizado por *ThreadFlag*.

Descrição das tarefas

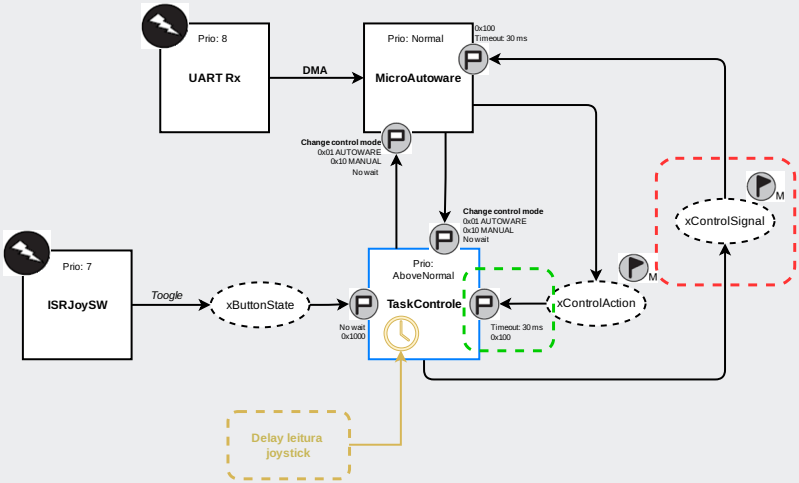


Figura 6: Elementos de bloqueio e desbloqueio da tarefa TaskControle.

Fluxograma MicroAutoware

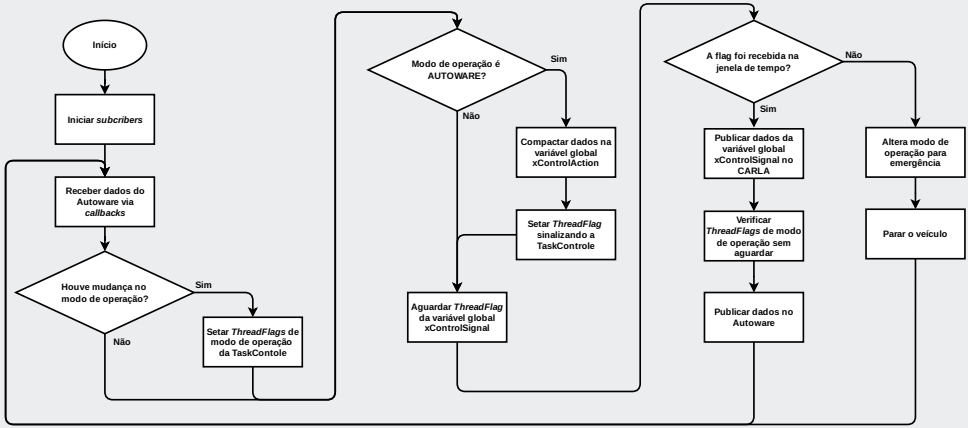


Figura 7: Fluxograma da tarefa MicroAutoware.

Fluxograma TaskControl

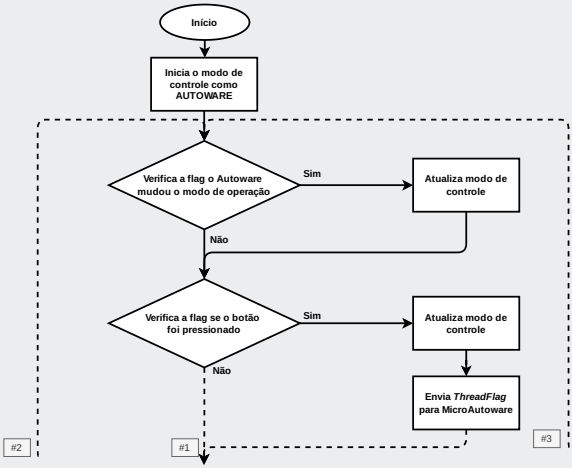


Figura 8: Fluxograma da tarefa TaskControle (Parte 1/2).

Fluxograma TaskControl

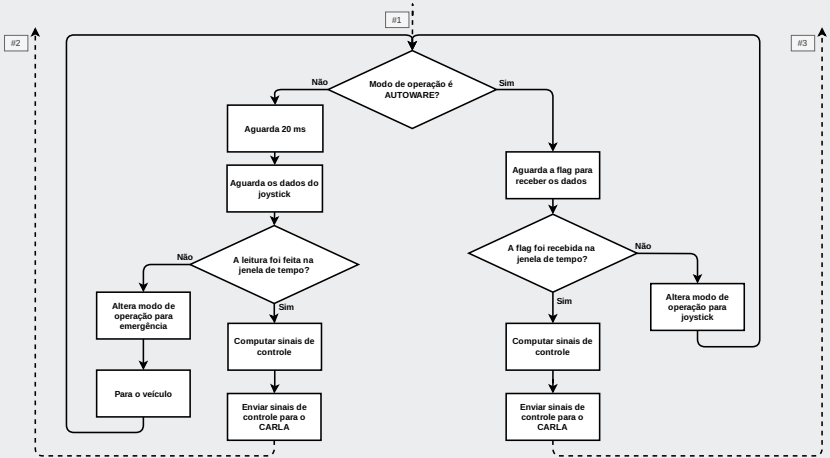


Figura 9: Fluxograma da tarefa TaskControle (Parte 2/2).

Fluxograma ISR JoySW

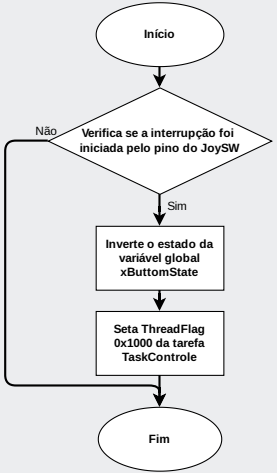


Figura 10: Fluxograma da ISR JoySW.

Sincronização e comunicação entre tarefas

Sincronização entre tarefas

Sinalização `xButtonState`

- **Objeto:** *ThreadFlag*
- **Flag:** 0x1000
- **Modo:** *No wait*
- **Descrição:** Sinaliza ocorrência da interrupção do botão JoySW.

Sincronização entre tarefas

Sinalização `xButtonState`

- **Objeto:** *ThreadFlag*
- **Flag:** 0x1000
- **Modo:** *No wait*
- **Descrição:** Sinaliza ocorrência da interrupção do botão JoySW.

Sinalização `xControlAction`

- **Objeto:** *ThreadFlag*
- **Flag:** 0x0100
- **Modo:** *Timeout* 30 ms
- **Descrição:** Sinaliza o recebimento de dados pela variável global `xControlAction`.

Sincronização entre tarefas

Sinalização `xButtonState`

- **Objeto:** *ThreadFlag*
- **Flag:** 0x1000
- **Modo:** *No wait*
- **Descrição:** Sinaliza ocorrência da interrupção do botão JoySW.

Sinalização `xControlAction`

- **Objeto:** *ThreadFlag*
- **Flag:** 0x0100
- **Modo:** *Timeout* 30 ms
- **Descrição:** Sinaliza o recebimento de dados pela variável global `xControlAction`.

Sinalização `xControlSignal`

- **Objeto:** *ThreadFlag*
- **Flag:** 0x0100
- **Modo:** *Timeout* 30 ms
- **Descrição:** Sinaliza o recebimento de dados pela variável global `xControlSignal`.

Comunicação entre tarefas

Alteração do modo de condução por interrupção JoySW

- **Objeto:** *ThreadFlag*
- **Flags:**
  - Modo de controle alterado para AUTOWARE: 0x01
  - Modo de controle alterado para MANUAL: 0x10
- **Modo:** *No wait*
- **Descrição:** Realiza a sincronização do modo de operação da tarefa TaskControle para a MicroAutoware.

Comunicação entre tarefas

Alteração do modo de condução por interrupção JoySW

- **Objeto:** *ThreadFlag*
- **Flags:**
  - Modo de controle alterado para AUTOWARE: 0x01
  - Modo de controle alterado para MANUAL: 0x10
- **Modo:** *No wait*
- **Descrição:** Realiza a sincronização do modo de operação da tarefa TaskControle para a MicroAutoware.

Alteração do modo de condução pelo Autoware

- **Objeto:** *ThreadFlag*
- **Flags:**
  - Modo de controle alterado para AUTOWARE: 0x01
  - Modo de controle alterado para MANUAL: 0x10
- **Modo:** *No wait*
- **Descrição:** Realiza a sincronização do modo de operação da tarefa MicroAutoware para a TaskControle.

Proteção de recursos

Proteção de recursos

Variável global `xControlSignal`

- Protegida por MUTEX.
  - `MutexControlSignal`

Proteção de recursos

Variável global `xControlSignal`

- Protegida por MUTEX.
  - `MutexControlSignal`

Variável global `xControlAction`

- Protegida por MUTEX.
  - `MutexControlAction`

Padronização de projeto

Padronização de projeto

Domínio ROS $\times$ Domínio FreeRTOS

Padronização de projeto

Domínio ROS $\times$ Domínio FreeRTOS

Domínio ROS

- *Vehicle interface*;
- Padronização de código do ROS.

Padronização de projeto

Domínio ROS $\times$ Domínio FreeRTOS

Domínio ROS

- *Vehicle interface*;
- Padronização de código do ROS.

Domínio FreeRTOS

- Padronização padrão da disciplina.

Padronização de projeto

Domínio ROS $\times$ Domínio FreeRTOS

Domínio ROS

- *Vehicle interface*;
- Padronização de código do ROS.

Domínio FreeRTOS

- Padronização padrão da disciplina.

Padronização de código ROS

- Subscriber: `nome_subscriber_sub_`
- Publisher: `nome_subscriber_pub_`
- Service server: `nome_subscriber_server_`
- Mensagem: `nome_mensagem_msg_`
- Node: `NomeDoNode`
- Callback: `nome_do_topico_callback`

Padronização de projeto

Domínio ROS $\times$ Domínio FreeRTOS

Domínio ROS

- *Vehicle interface*;
- Padronização de código do ROS.

Domínio FreeRTOS

- Padronização padrão da disciplina.

Padronização de código ROS

- Subscriber: `nome_subscriber_sub_`
- Publisher: `nome_subscriber_pub_`
- Service server: `nome_subscriber_server_`
- Mensagem: `nome_mensagem_msg_`
- Node: `NomeDoNode`
- Callback: `nome_do_topico_callback`

Considera-se que as padronizações não irao se misturar!

Cronograma

Cronograma

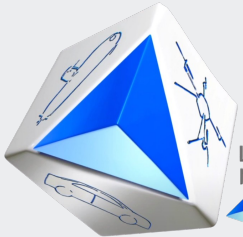
Atividade/Semana	1	**2**	3	**4**	5	6	**7**	8	**9**
Proposta do projeto	■								
Projeto de *hardware* e *software*		■	■						
Integração do STM com o micro-ROS		■							
Integração do micro-ROS com o Autoware			■	■	■				
Implementação das tarefas do sistema embarcado				■	■	■	■		
Construção do ambiente de testes					■	■	■		
Realização dos testes							■	■	■
Escrita do relatório		■	■	■	■	■	■	■	■

Tabela 1: Cronograma de atividades.

- **Semana 2:** Apresentação Etapa 1
- **Semana 4:** Apresentação Etapa 2
- **Semana 7:** Apresentação Etapa 3
- **Semana 9:** Apresentação Final

Obrigado!

Dúvidas?



LABORATÓRIO DE MOBILIDADE AUTÔNOMA