

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROJETO E DESENVOLVIMENTO DE SISTEMAS EMBARCADOS - DAS410056

Projeto em AADL de Câmera Estabilizada com Gimbal para Aeronave Remotamente Pilotada

Autores:

Afonso da Fonseca Braga

Angela Crepaldi

Rafael Azambuja da Silva

Professor:

Leandro Buss Becker, Dr.

Florianópolis, 7 de outubro de 2019.

# Introdução

# REQUISITOS

1- O sistema da câmera estabilizada deve se comunicar com o controlador da aeronave remotamente pilotada (ARP).

2- A posição da câmera deve ser remotamente controlada.

3- Capturas de imagens devem salvar em um cartão SD.

# FUNÇÕES

1- Ler a posição da câmera.

2- Movimentar a câmera nos três eixos ortogonais (*roll*, *pitch*, *yaw*) a partir de comandos recebidos pelo rádio do ARP.

3- Estabilizar a imagem da câmera.

4- Salvar as imagens da câmera.

5- Enviar imagens de baixa resolução para a ARP.

6- Monitorar funcionamento do sistema.

# SISTEMA

A screenshot of a cell phone

Description automatically generated

Figura 1: Sistema de controle da câmera estabilizada.

## Componentes

Processador

Processador

barramento para o controlador do ARP: USART

memória compartilhável

3 servo-motores

3 controladores PWM

Câmera USB

IMU (ângulos de rolagem, arfagem e guinada, aceleração nos três eixos)

I2C

SD Card

# Processo

O sistema possui apenas um processo chamado de *sw* e pode ser visto na Figura 2 com as threads, as conexões e os fluxos das threads. Na Figura 3, são mostrados os fluxos do processo.

A screenshot of a social media post

Description automatically generated

Figura 2: Processo

A picture containing map, text

Description automatically generated

Figura 3:Fluxos do processo.

As funções do sistema foram divididas em quatro threads: *get\_attitude*, *control\_servos*, *process\_video* e *health\_mgr*. As características de tempo das threads são mostradas na Tabela 1, onde a tarefa de mais alta prioridade é a de maior valor, quatro.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Threads** | **Período** | **Tempo de execução** | **Deadline** | **Prioridade** |
| get\_attitude | 40 ms | 2 ms a 4 ms | 40 ms | 4 |
| control\_servos | 100 ms | 4 ms | 100 ms | 3 |
| process\_video | 50 ms | 10 ms a 20 ms | 50 ms | 2 |
| health\_mgr | 1000 ms | 2 ms | 1000 ms | 1 |

Tabela 1: Características temporais das threads do sistema.

## get\_attitude

A thread *get\_attitude* é responsável por receber os dados dos sensores inerciais da IMU (Attitude\_IMU) e o comando de posicionamento do operador remoto (Radio\_Command\_in) que é recebido pelo controlador da ARP. Essa thread gera um valor de posicionamento para o comando dos servo-motores (Move\_Servos\_Out) e envia dados de monitoramento de falha de software e hardware (health).

Foi considerada como restrição temporal que o período da thread *get\_attitude* fosse pelo menos metade do período da thread que usa os dados de saída da get\_attitude, respeitando o Teorema de Nyquist.

## control\_servos

A thread *control\_servos* trata os dados de posicionamento (Move\_Servos\_In) e comanda os servo-motores por PWM (Move\_Roll\_Out, Move\_Pitch\_Out, Move\_Yaw\_Out). Essa thread também envia dados de monitoramento de falha de software e hardware relativos ao seu funcionamento (health).

## process\_video

A thread *process\_video* captura os frames da câmera de vídeo (video\_from\_camera), faz o processamento de imagem, envia a imagem processada para armazenar no cartão de memória (video\_to\_sd\_card) e envia uma versão da imagem com baixa resolução para o controlador do ARP (video\_to\_uav) para que o operador tenha uma amostra da imagem. Essa thread também envia dados de monitoramento de falhas do software e dos dispositivos de hardware relacionados a ela.

O período da thread de 50 ms foi baseado numa taxa de 20 frames por segundo para captura das imagens da câmera.

## health\_mgr

A thread *health\_mgr* executa o monitoramento do sistema e das demais threads (attitude, vídeo, servos) e envia o status para o operador (status\_to\_uav) a fim de que ele possa tomar uma decisão caso haja alguma falha no sistema prejudicando a missão.

Julgou-se que um período de 1 segundo é suficiente para atualizar os dados de saúde do sistema.

# Resultados

Os resultados foram obtidos através do AADL Inspector.

## Simulação

A Figura 4 mostra a linha de tempo com as tarefas e as suas entradas e saídas. O período em que as tarefas estão em execução é mostrado em preto e o período em que as tarefas estão aguardando execução ou são preemptadas estão em laranja. A troca de mensagens é mostrada com as setas em cinza.

A screenshot of a video game

Description automatically generated

Figura 4: Simulação do sistema.

## Análise Temporal

A screenshot of a cell phone

Description automatically generated

Figura :

A screenshot of a social media post

Description automatically generated

Figura :

A screenshot of a cell phone

Description automatically generated

Figura :

Apesar de incluir fluxo de dados ponta-a-ponta, não se conseguiu analisar o resultado.

# Conclusão

# Referências Bibliográficas