Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Eletrônica (DAELN)

SISTEMAS EMBARCADOS

Laboratório 5

Prof. André Schneider de Oliveira

andreoliveira@utfpr.edu.br

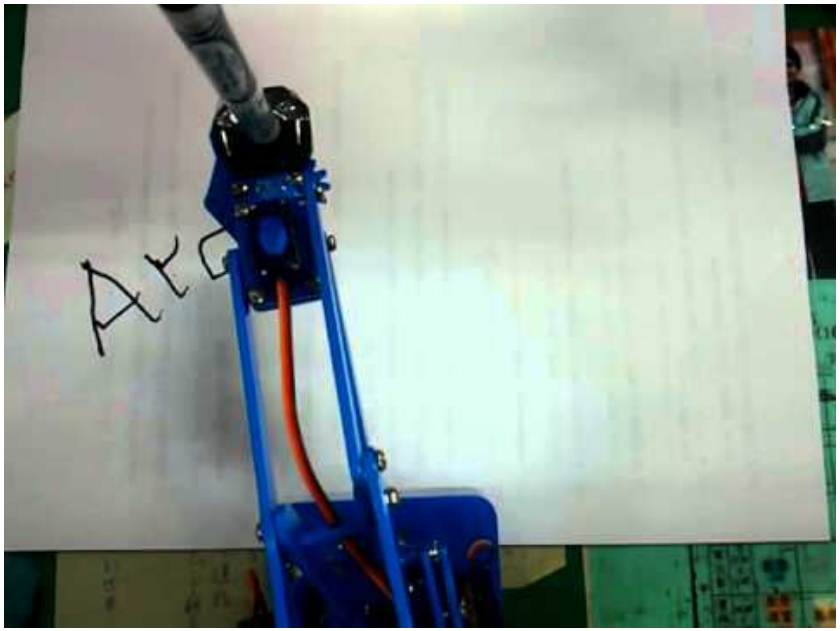
Objetivo

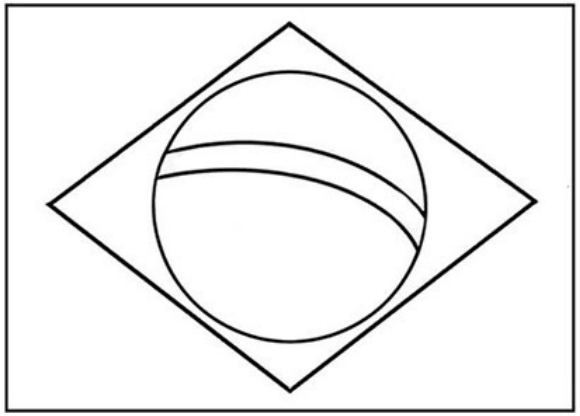
- Escrever uma programa em linguagem C++ Cortex-M, utilizando o CMSIS RTOS, para o escalonamento de múltiplas tarefas com prioridades dinâmicas

Tarefa	Funcionalidade	Prioridade	Deadline	Periodicidade
A	Geração de números primos	**Realtime (-100)**	30%	5Hz
B	Recebimento da UART	**High (-30)**	10%	**interrupção**
C	Geração de pontos para o manipulador	**Low(10)**	70%	10Hz
D	Controle do manipulador	**Normal (0)**	40%	10Hz
E	Geração de números de Fibonacci	**Normal (0)**	50%	1Hz

Especificações

- O sistema regerá um manipulador para a tarefa de desenho, e será controlado via UART, podendo: ***desenhar retângulo, losango, circulo e bandeira do brasil; ou parar o desenho, ou ser acionado pelo teclado.***





https://www.youtube.com/watch?v=VarbJ12r-1A

Especificações

- O sistema deve executar periodicamente as tarefas especificadas de acordo com sua frequência
- Deve-se atribuir um sistema de prioridades onde a de menor valor é a mais alta
- É necessária a inclusão de uma fila de execução que armazene o identificador, prioridade e tempo de relaxamento restante (em ticks) das threads

Especificações

- Toda a tarefa possui o seu deadline especificado, ou seja, o percentual a mais de **ticks** que pode aguardar até a tarefa ser concluída, por exemplo

tarefa leva 20 ticks e possui um deadline 30%

a tarefa deve terminar em até 26 ticks após ser iniciada

***Sugestão: executar a operação em um programa simples (sem RTOS) para determinar o tempo de execução em ticks**

Especificações

- Tarefas com prioridade **realtime** não podem finalizar após o seu deadline. Caso ocorra, o sistema terá falha geral e irá encerrar com a mensagem de erro ***"master fault"***

- Caso alguma tarefa com prioridade **não-realtime** seja finalizada **após o seu deadline**, a mesma deve ter a sua **prioridade incrementada** na próxima execução, gerando a mensagem **"secondary fault"**

- Caso alguma tarefa com prioridade **não realtime** seja finalizada **antes da metade do seu deadline**, a mesma deve ter sua **prioridade decrecida** na próxima execução, gerando a mensagem **"secondary fault"**

Especificações

- As informações de execução das tarefas e do escalonador devem ser apresentadas no display do *Oled*, dentre elas

  - **Prioridade**
  - **Tempo de relaxamento restante (em ticks)**
  - **Estado *(ready, running, waiting)***
  - **Percentual de execução (quanto já foi processada)**
  - **Atraso (em ticks)**
  - **Fila de execução**
  - **Faltas ocorridas**

- Deve ser gerado o diagrama de *Gantt*, por meio de arquivo texto, no formato do site *(https://knsv.github.io/mermaid/live_editor/)*, através da UART, com o objetivo de **demonstrar a funcionalidade do escalonador**

Especificações RTOS

- A implementação deve se basear no CMSIS RTOS, onde cada tarefa deve ser necessariamente uma thread
- A thread principal deve ser convertida em escalonador após a inicialização do ***Kernel***
- Deve ser respeitada a nomenclatura das threads
- **Sugestão:** a utilização de um semáforo no escalonador pode minimizar a ocorrência de execuções não desejadas das tarefas

Cronograma de Avaliação – Laboratório 5

- 12/11
  - Entrega do Diagrama de Estados e Transições (DET) da solução proposta
  - A não execução acarretará na **anulação** do laboratório

- 19/11
  - Apresentação de um desenho simplificado pelo manipulador
  - A não execução acarretará na **perda de 1 ponto** no laboratório

- 26/11
  - Defesa, demonstração e teste