|  |  |
| --- | --- |
| CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC | |
| (X) PRÉ-PROJETO     (     ) PROJETO | ANO/SEMESTRE: 2023/1 |

IMPLEMENTAÇÃO DA M+++ NO ESP-32 UTILIZANDO O SISTEMA OPERACIONAL NUTTX

Gustavo Felipe Soares

Prof. Danton Cavalcanti Franco Junior – Orientador

Prof. Miguel Alexandre Wisintainer – Coorientador

# Introdução

Sistemas embarcados são importantes na atualidade e podem ser encontrados em uma ampla gama de dispositivos que utilizados no dia a dia. Esses sistemas são compostos por hardware e software e possuem periféricos de entrada e saída de dados, além de um microcontrolador que é o principal componente de um sistema embarcado. O microcontrolador é um computador com capacidade de processamento reduzida, responsável por executar e gerenciar uma aplicação. Tanto o ESP-32 quanto a M+++ são exemplos de microcontroladores utilizados em sistemas embarcados, sendo que a M+++ foi desenvolvida na Fundação Universidade Regional de Blumenau (FURB) em 2003 e vem sendo aperfeiçoada desde então.

Para auxiliar no gerenciamento e na eficiência dos sistemas embarcados, é possível instalar um Sistema Operacional (S.O.) compatível com o microcontrolador. Porém, é importante que esses sistemas operacionais sejam leves e otimizados para o processamento, a fim de executar a aplicação de forma eficiente. Um dos S.O.s que vem ganhando destaque no mercado é o NuttX, um sistema operacional recente e voltado para microcontroladores, que será abordado nesse estudo.

O NuttX é um sistema operacional de tempo real, ou seja, executa uma aplicação por vez. Pode ser usado em microcontroladores de 8 a 64 bits e é compatível com os padrões Portable Operating System Interface (POSIX) e American National Standards Institute (ANSI) e diversos hardwares. Lançado em 2007 por Gregory Nutt e passado para a Apache em 2019, possui uma grande comunidade de desenvolvedores e se tornou uma ótima opção para uso em microcontroladores.

No entanto, apesar da sua crescente popularidade, o NuttX possui pouca documentação unificada que detalhe o passo a passo para o desenvolvimento e disponibilização de aplicações a serem usadas no próprio NuttX. Diante desse contexto, esse trabalho propõe o desenvolvimento da M+++ em um ESP-32 utilizando o NuttX como sistema operacional, visando gerar documentação para ele. O objetivo é facilitar e auxiliar os futuros projetos e aplicações de sistemas embarcados.

## OBJETIVOS

O objetivo principal é disponibilizar uma aplicação web capaz de interpretar código *assembly* e rodar em um ESP-32 utilizando o sistema operacional NuttX visando a geração de documentação de todo o processo.

Os objetivos específicos são:

1. possibilitar a inserção e interpretação de código *assembly* em componentes físicos;
2. adicionar uma aplicação ao NuttX;
3. gerar documentação da configuração do sistema operacional NuttX;
4. avaliar a eficácia do compilador em sala de aula com alunos.

# trabalhos correlatos

Nessa etapa serão apresentados trabalhos semelhantes aos principais objetivos do estudo. Na seção 2.1 é apresentado o trabalho de Klann (2017) que desenvolveu uma aplicação *desktop* para simular o microcontrolador M+++. Na seção 2.2 é apresentado o trabalho de Bieging (2018) que implementou a M++ em um Field-Programmable Gate Array (FPGA) e na seção 2.3 é apresentado o artigo de Assis e Jerpelea (2023) que apresentou o NuttX, um sistema operacional POSIX voltado para sistemas embarcados.

## SOFTWARE SIMULADOR DO MICROCONTROLADOR M+++

O software desenvolvido em 2017 por Klann, ex-aluno do curso de Ciências da Computação da FURB, é uma ferramenta didática de simulação virtual da M++, um microcontrolador criado na FURB em 2003 usado nas aulas de Arquitetura de Computadores I. Nele é possível codificar e depurar *assembly*, salvar e carregar projetos, construir circuitos eletrônicos e visualizar sinais internos do microcontrolador a partir de uma interface gráfica. Além disso, o software também possui um montador *assembly* que valida o código criado ao rodá-lo. A Figura 1 mostra uma das versões das telas de codificação, de depuração e do circuito.

Figura 1 – Telas do software de simulação da M+++



Fonte: Klann (2017, p.44).

Por se tratar de uma evolução, traz uma série de melhorias em relação à versão anterior como a performance e a possibilidade de adicionar pontos de parada, embora também possua desvantagens, pois há uma quantidade menor de componentes disponíveis. Segundo o autor, essa solução apresentou resultados positivos nas pesquisas realizadas por ele, auxiliando na compreensão do funcionamento de sistemas embarcados, sendo bastante utilizado nas aulas.

## IMPLEMENTAÇÃO DA M++ EM FPGA

O projeto desenvolvido em 2018 por Bieging, também ex-aluno do curso de Ciências da Computação da FURB, tem como objetivo trazer a M++ para um FPGA funcionando em um *clock* superior a 10KHz. Para isso, é possível carregar programas para a memória do microprocessador para serem lidos e interpretados. Com a finalidade de reproduzir a M++, seis módulos foram criados:

1. módulo de controle: conecta todos os módulos;
2. Unidade Lógica e Aritmética (ULA): implementa a ULA da M++;
3. endereçador de memória de programa;
4. banco de registradores: vetor para armazenar conteúdo dos registradores;
5. endereçador da memória RAM externa: contador de incremento e decremento;
6. memórias ROM.

Segundo o autor, a frequência máxima obtida foi de 90MHz e utilizou apenas 129 das mais de 41.000 portas lógicas disponíveis. O Quadro 1 mostra um comparativo entre seu projeto e outras três implementações de processadores:

Quadro 1 - Comparativo dos resultados do Bieging (2018)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trabalhos Correlatos  Características | AYEH et al., (2008) | PABLO et al., (2016) | ZALAVA et al., (2015) | M++ FPGA |
| fabricante | Xilinx | Xilinx | Xilinx | Altera |
| tecnologia | FPGA | FPGA | FPGA | FPGA |
| número de instruções | 4 | 29 | 29 | 14 |
| arquitetura | Própria | Harvard | Harvard | Harvard |
| set de instruções | Próprio | RISC | RISC | Próprio |
| frequência | ~95MHz | ~40MHz | Não inf. | 90MHz |

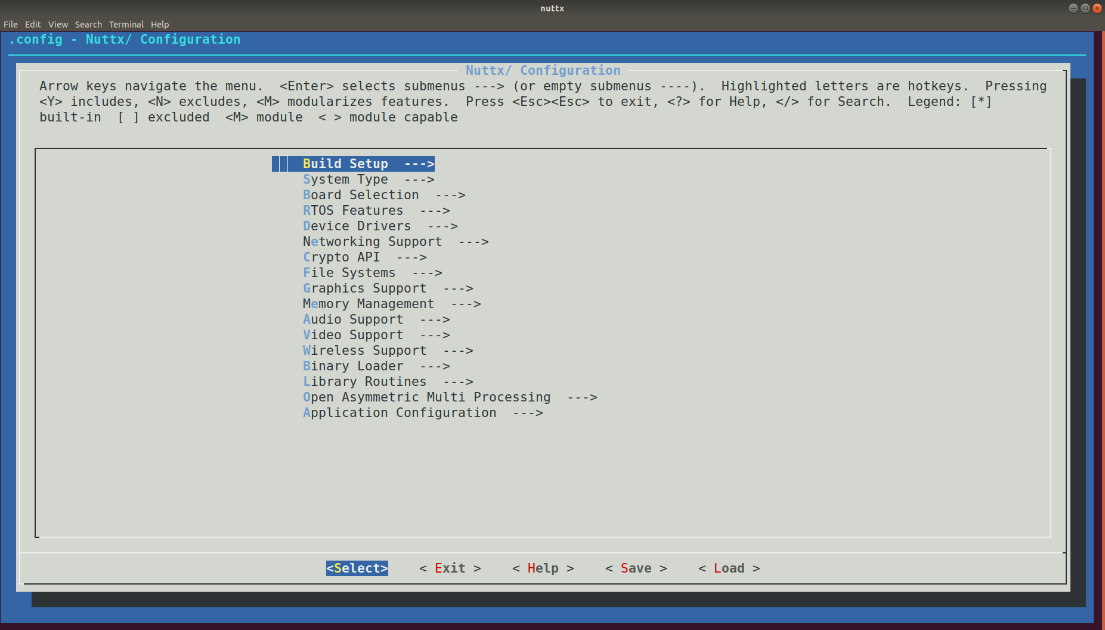
Fonte: Bieging (2018, p.71).

Bieging ainda comenta que seu projeto obteve um avanço, uma vez que com o simulador da M++ da época só era possível alcançar 50Hz. Seu trabalho atingiu todos os objetivos esperados, além de ter gerado documentação sobre a arquitetura.

## (Apache) NuttX A Linux-like RTOS for Microcontrollers

Criado em 2007 por Gregory Nutt e doado para o Apache em 2019, o NuttX é um sistema operacional POSIX voltado para sistemas embarcados por ser bastante pequeno. “Pensando em ser uma alternativa de ser o que o Linux deveria ser para microcontroladores” (Assis e Jerpelea, 2023, p.2, tradução nossa), ele possui uma série de características fundamentais para esses dispositivos: é altamente configurável, roda em tempo real, é determinístico, possui suporte para hierarquia de prioridade etc. Além de ter diversas funcionalidades e *drivers* como sistema de arquivos, protocolos de internet, suporte gráfico, suporte à USB e áudio, também possui compatibilidade com vários outros microcontroladores como o ESP-32. Na Figura 2 é mostrado o menu de configuração do NuttX.

Figura 2 – Tela de configuração do NuttX



Fonte: Apache Software Foundation, 2020.

O NuttX, como os próprios autores comentam, “foi desenvolvido desde o começo para ser compatível com POSIX. [...] Tornando melhor a transição para o NuttX para os desenvolvedores” (Assis e Jerpelea, 2023, p.4, tradução nossa), uma vez que poderão assimilar soluções Linux para microcontroladores rodando NuttX.

# proposta DO SOFTWARE

A seguir serão apresentadas a proposta e a justificativa, assim como as principais características dos trabalhos correlatos, os requisitos e a metodologia utilizada junto ao cronograma.

## JUSTIFICATIVA

No Quadro 2 são comparados os três trabalhos correlatos acima a partir das principais características buscadas.

Quadro 2 - Comparativo dos trabalhos correlatos

|  |  |  |  |
| --- | --- | --- | --- |
| Trabalhos Correlatos  Características | Klann (2017) | Bieging (2018) | Assis e Jerpelea (2023) |
| Interpreta código assembly | X |  |  |
| Simula a M+++ | X | X |  |
| Possui sistema operacional NuttX |  |  | X |
| Código gerado é executável em microcontroladores |  |  | X |
| É executado em microcontroladores |  |  | X |

Fonte: elaborado pelo autor.

Conforme mostrado no Quadro 2, tanto a aplicação desenvolvida por Klann (2017) quanto a desenvolvida por Bieging (2018) simulam a M+++, porém não rodam em microcontroladores e no NuttX. O software *desktop* implementado por Klann (2017) tem como propósito simular virtualmente, possuindo uma série de componentes para acoplar ao microcontrolador e podendo interpretar código *assembly* e visualizar o comportamento dos componentes de forma virtual.

Já o projeto do Bieging (2018) tem como objetivo rodar a simulação em FPGA que, diferentemente de um microcontrolador que se utiliza de software e já possui alguns componentes, possui diversas portas lógicas programáveis e precisa de periféricos externos.

O NuttX, apresentado por Assis e Jerpelea (2023) tem a proposta de rodar como um Linux em microcontroladores e permite a criação de aplicações novas nele.

Para a implementação do estudo se faz necessário o entendimento sobre o NuttX, a M+++ e compiladores. No primeiro trabalho correlato, os principais aspectos técnicos da M+++ e como ela funciona são mostrados, assim como a definição de um compilador e seus principais analisadores: léxico, sintático e semântico.

Esse trabalho tem a intensão de mostrar e documentar todos os passos para disponibilizar uma aplicação compatível com o NuttX visando a geração de um guia de desenvolvimento. Além disso, o produto gerado pelo estudo permitirá visualizar a execução de sistemas embarcados em meio físico em que, aplicado de forma didática, possibilitará uma maior imersão por parte dos alunos.

## REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os principais Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) são:

1. permitir a inserção de código *assembly* (RF);
2. salvar e carregar código *assembly* (RF);
3. exibir erros retornados pelo compilador (RF);
4. executar um *script* por vez (RF);
5. ser compatível com o sistema operacional NuttX (RNF);
6. ser desenvolvido em C (RNF).

## METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

1. levantamento bibliográfico: buscar por trabalhos relacionados à M+++ e ao NuttX, assim como artigos e outras fontes voltadas ao desenvolvimento em NuttX;
2. levantamento de requisitos: detalhar os requisitos da aplicação com base nas bibliografias e definições dadas pelo orientador e coorientador;
3. fundamentação teórica: estudo sobre o NuttX e a linguagem *assembly*;
4. levantamento de ferramentas a serem usadas: definir quais ferramentas (software e hardware) são necessárias para a execução e desenvolvimento da aplicação;
5. documentação: detalhar todos os passos necessários para a configuração do sistema operacional NuttX, desenvolvimento da aplicação e disponibilização dela;
6. desenvolvimento: implementação da aplicação a partir dos requisitos levantados, sendo ela desenvolvida em C utilizando o Visual Studio Code;
7. testes: validar as principais funcionalidades do software tais como o montador *assembly* e o servidor web;
8. validação com alunos: testar a aplicação e a documentação com alunos em sala para avaliar o funcionamento.

As etapas serão realizadas nos períodos relacionados no Quadro 3 abaixo.

Quadro 3 - Cronograma

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | 2023 | | | | | | | | | | | | | | | | |  | | | |  | | | |  | | | |  | | | |  | |
|  | fev. | | mar. | | | abr. | | | maio | | | jun. | | | jul. | | | ago. | | | | set. | | | | out. | | | | nov. | | | | dez. | | | |
| etapas / quinzenas | 1 | 2 | | 1 | 2 | | 1 | 2 | | 1 | 2 | | 1 | 2 | | 1 | 2 | | 1 | | 2 | | 1 | | 2 | | 1 | | 2 | | 1 | | 2 | | 1 | | 2 |
| levantamento bibliográfico |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| levantamento de requisitos |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| fundamentação teórica |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| levantamento de ferramentas a serem usadas |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| documentação |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| desenvolvimento |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| testes |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| validação com alunos |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |

Fonte: elaborado pelo autor.

# REVISÃO BIBLIOGRÁFICA

Este capítulo descreve brevemente os assuntos que fundamentarão o estudo a ser realizado: aplicações NuttX para sistemas embarcados, a M++ e a linguagem *assembly*.

Sistemas embarcados são úteis para “executar uma tarefa específica em um sistema maior. [...]. Eles também precisam ser capazes de lidar com restrições de espaço e recursos como memória e processamento limitados.” (Souza, 2023). Por isso, sistemas operacionais como Windows e Linux não são boas opções para serem utilizados nesses dispositivos.

Para solucionar isso, existem diversos sistemas operacionais modelados para essas circunstâncias como é o caso do NuttX, que começou a ter visibilidade nos últimos anos, podendo rodar com apenas 32K de memória. Esse sistema operacional já possui uma série de aplicações prontas para o uso e a cada dia novas funcionalidades são adicionadas a ele pela comunidade de desenvolvedores.

A M++ é um microcontrolador criado na FURB e desenvolvido no software Logisim. De acordo com Jung (2014), ela possui:

1. memória RAM de 8 bits;
2. memória de pilha de 8 bits;
3. memória ROM de 16 bits;
4. 4 registradores de entrada;
5. 4 registradores de saída;
6. operações da Unidade de Lógica e Aritmética (ULA);
7. 5 operações de salto;
8. linguagem semelhante ao *assembly*;
9. as flags End Of Instruction (EOI), Carry e Zero;
10. 4 registradores de 8 bits (B – E) + acumulador de 8 bits (A).

Para se adequar à M++, o componente PCA9555 deverá ser adicionado ao ESP-32, uma vez que a M++ possui 32 entradas/saídas e o ESP-32 utilizado tem apenas 16. Esse componente é um expansor de 16 portas e será integrado ao microcontrolador a partir do protocolo I2C, que possui dois canais de comunicação: *serial data*, para transmissão de dados e o *serial clock*, para manter a sincronia entre os dois dispositivos conectados.

O *assembly* é uma linguagem de programação de baixo nível que funciona como uma abstração do código de máquina, tornando mais fácil seu entendimento e bastante útil quando se quer trabalhar byte a byte e/ou com menos memória, além de ser mais rápido. Porém, pode ser demorado escrever o código por ser uma linguagem mis complexa.

Algumas das instruções encontradas no *assembly* x86, conforme o *website* GitBook (2022), estão listadas abaixo:

1. matemáticas: ADD, SUB, INC, DEC, MUL e DIV;
2. lógicas: AND, OR, XOR, CMP, NEG e NOT;
3. saída/atribuição: MOV, POP e PUSH.

Também possui os seguintes registradores:

1. AX – Acumulador;
2. BX – Endereço base;
3. CX – Contador;
4. DX – Dado;
5. SP – Ponteiro para o topo da pilha;
6. BP – Ponteiro para o início da pilha;
7. SI – Endereço de origem dos dados;
8. DI – Endereço de destino dos dados.

Já o *assembly* da M++ possui algumas diferenças. Abaixo estão listadas as instruções disponibilizadas por Borges (2014):

1. matemáticas: ADD, SUB e INC;
2. lógicas: AND, OR, XOR e NOT;
3. saída/atribuição: MOV, POP e PUSH.

Os seguintes registradores são encontrados na M++: B, C, D e E.

Referências

APACHE SOFTWARE FOUNDATION. **Configuring**. 2020. Disponível em: https://nuttx.apache.org/docs/latest/quickstart/configuring.html. Acesso em: 13 mar. 2023.

ASSIS, Alan Carvalho de; JERPELEA, Alin. **(Apache) NuttX**: a linux-like rtos for microcontrollers. 2023. 6 f. Apache, 2023. Disponível em: https://pt.scribd.com/document/628557282/NuttX-RTOS. Acesso em: 13 mar. 2023.

BIEGING, André Leonardo. **IMPLEMENTAÇÃO DA M++ EM FPGA**. 2018. 75 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2018. Disponível em: https://www.furb.br/dsc/tcc/index.php?cd=6&tcc=1937. Acesso em: 20 mar. 2023.

GITBOOK. **Registradores de propósito geral**. 2022. Disponível em: https://mentebinaria.gitbook.io/assembly/a-base/registradores-de-proposito-geral. Acesso em: 10 abr. 2023.

JUNG, Jean. **M+++**. 2014. Disponível em: https://github.com/jejung/maquina-plus-plus/blob/master/README.md. Acesso em: 13 mar. 2023.

KLANN, Jean Carlos. **SOFTWARE SIMULADOR DO MICROCONTROLADOR M+++**. 2017. 61 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2017. Disponível em: https://www.furb.br/dsc/tcc/index.php?cd=6&tcc=1837. Acesso em: 13 mar. 2023.

SOUZA, Fábio. **O que são sistemas embarcados?** 2023. Disponível em: https://embarcados.com.br/o-que-sao-sistemas-embarcados/. Acesso em: 21 mar. 2023.

FORMULÁRIO DE avaliação BCC – PROFESSOR AVALIADOR – Pré-projeto

Avaliador(a): **Francisco Adell Péricas**

Atenção: quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ASPECTOS AVALIADOS | | Atende | atende parcialmente | não atende |
| ASPECTOS TÉCNICOS | 1. INTRODUÇÃO   O tema de pesquisa está devidamente contextualizado/delimitado? | X |  |  |
| O problema está claramente formulado? |  | X |  |
| 1. OBJETIVOS   O objetivo principal está claramente definido e é passível de ser alcançado? | X |  |  |
| Os objetivos específicos são coerentes com o objetivo principal? |  | X |  |
| 1. TRABALHOS CORRELATOS   São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos? |  | X |  |
| 1. JUSTIFICATIVA   Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada? | X |  |  |
| São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta? | X |  |  |
| São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta? | X |  |  |
| 1. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO   Os requisitos funcionais e não funcionais foram claramente descritos? | X |  |  |
| 1. METODOLOGIA   Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC? |  | X |  |
| Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta? | X |  |  |
| 1. REVISÃO BIBLIOGRÁFICA   Os assuntos apresentados são suficientes e têm relação com o tema do TCC? |  | X |  |
| As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)? | X |  |  |
| ASPECTOS METODOLÓGICOS | 1. LINGUAGEM USADA (redação)   O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica? | X |  |  |
| A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)? | X |  |  |