UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM **ENGENHARIA DE COMPUTAÇÃO**
SÉTIMO SEMESTRE

QXD0143 - MICROCONTROLADORES

Relatório 01: Conceitos Introdutórios de Sistemas Embarcados e Microcontroladores *ARM*

PEDRO HENRIQUE **MAGALHÃES BOTELHO**

QUIXADÁ - CE
2022

471047 — PEDRO HENRIQUE **MAGALHÃES BOTELHO**

Relatório 01: Conceitos Introdutórios de Sistemas Embarcados e Microcontroladores

Orientador: Prof. Dr. Thiago Werlley Bandeira da Silva

Primeiro relatório escrito para a disciplina de Microcontroladores, no curso de graduação em Engenharia de Computação, pela Universidade Federal do Ceará (UFC), campus em Quixadá.

QUIXADÁ - CE
2022

Conteúdo

Lista de Figuras

1 Introdução

Esse é o primeiro relatório da disciplina de **Microcontroladores**, ministrada pelo professor Thiago W. Bandeira. Nesse relatório são discutidos temas introdutórios da disciplina, como a área de sistemas embarcados, termos usados na área, bem como uma visão geral dos processadores baseados na arquitetura ARM®.

O documento contém informações e imagens retiradas do livro "The Definitive Guide to ARM®Cortex®-M0 and Cortex-M0+ Processors". Ao final, é realizada um fechamento da discussão, na sessão "Conclusão".

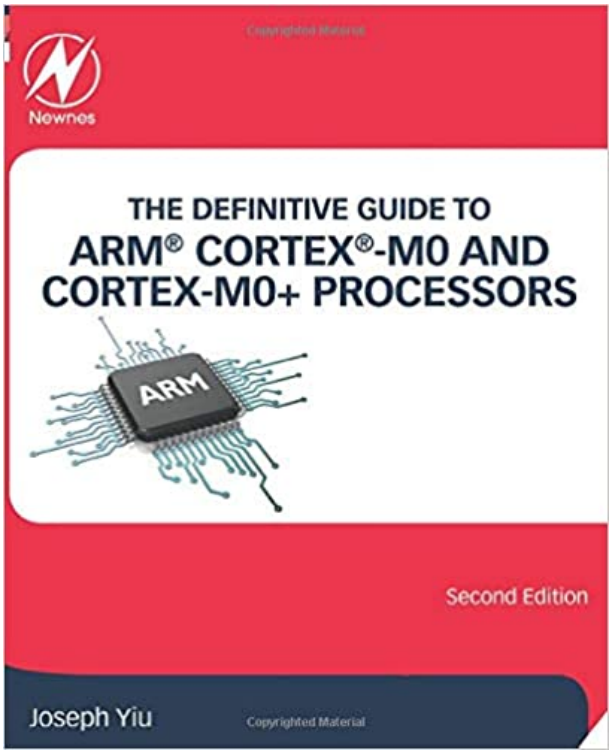

Figura 1: Capa do Livro de Referência

2 A Área de Sistemas Embarcados e suas Terminologias

Desde o advento das pesquisas na área da eletricidade sistemas elétricos ficaram cada vez mais populares. Com o avanço dos estudos sobre os computadores o acesso a essa máquinas foi muito facilitado. A dissipação dos computadores pelo globo e a sua utilização para diversos fins mesclou-se com a ideia de circuitos pequenos que realizam atividades específicas. Daí surge a ideia de construir computadores específicos para realizar pequenas tarefas muito específicas e restritas, a invés de grandes computadores que realizam diversas tarefas, sendo eles muito menores e mais baratos que um computador dito de **propósito geral**.

Temos então o surgimento dos primeiros sistemas embarcados, computadores sendo utilizados para controlar circuitos específicos e facilitar a vida dos projetistas. Claro, a história não se restringe a isso, mas o importante a ressaltar é a ideia de que os sistemas embarcados são pequenos computadores que estão por toda parte! Carros, microondas, sistemas de segurança, etc...

2.1 Conceito de Sistema Embarcado

Pelo termo **Sistema Embarcado** compreendemos que se trata de um *computador de propósito específico*, ou seja, feito para realizar uma função específica intermitentemente. Isso é a definição mais formal que podemos dar.

Podemos definir um sistema embarcado de forma mais informal, ou seja, como um computador que realiza o controle de um sistema, que é quase que totalmente elétrico, estando [o computador] embutido(embedded) no sistema controlado. Como exemplo temos um pequeno computador, ao qual damos o nome de **microcontrolador**, que está anexado a um sistema elétrico de uma trava eletrônica. O microcontrolador irá, então, controlar esse sistema utilizando-se de duas técnicas: por meio de conexões físicas aos terminais do circuito(os pinos do microcontrolador) e um software, o sistema carregado no pequeno computador, que tem como única tarefa controlar essa trava!

Como o computador irá controlar o sistema irá depender de vários fatores, mas podemos categorizar-los em componentes de **hardware** e **software**. O componente principal para a coordenação das tarefas de um computador é o **processador**. Ele que irá processar os dados necessários para realizar a tarefa necessária, seja executando uma instrução ou obtendo dados dos pinos do microcontrolador.

2.2 Microcontroladores e os Sistemas Embarcados

Sabendo que os sistemas embarcados estão espalhados pelo globo temos então a ideia de termos processadores sendo utilizados na maioria dos eletrônicos. Geralmente, os processadores nesses sistemas são "encapsulados" em chips chamados de microcontroladores, que

também possuem sistemas de memória, para persistência de dados e código(memória flash), assim como para utilização de dados pelo programa(memória SRAM), bem como dispositivos internos para interface de hardware(conversores A/D, D/A, UART, etc, ou seja, pequenos periféricos).

Os microcontroladores estão disponíveis nos mais diversos tipos: diferentes processadores, tamanhos de memória, periféricos internos, bem como diferentes encapsulamentos(forma como os componentes internos são empacotados) também, como mostra a figura 2.

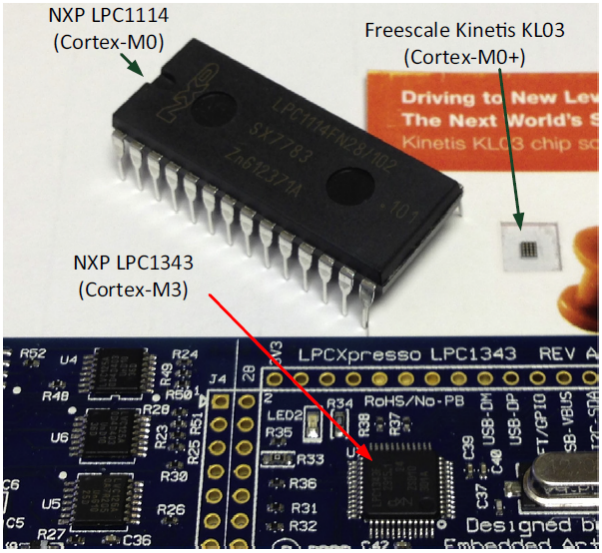

Figura 2: Diferentes Encapsulamentos de Microcontroladores

Vários microcontroladores também são fabricados para serem de propósito geral, significado que podem ser utilizados para uma vasta gama de aplicações. Mas não confunda: o sistema embarcado é de propósito específico, o que significa que ele será projetado para fazer apenas uma, ou algumas, operação, sendo ela suportada pelo microcontrolador.

Existem alguns termos relacionados usados para denominar microcontroladores que o autor do livro emprega. Esses termos são muito utilizados na área da eletrônica, portanto, muito importantes de se conhecer. Vejamos alguns deles:

- **Application-Specific Integrated Circuits (ASICs)**: Circuitos projetados especificamente e unicamente para um conjunto muito pequeno de produtos ou para um propósito muito específico.
- **Application Specific Standard Products (ASSPs)**: Circuitos projetados para realizar funções específicas, porém com maiores aplicações, sendo utilizados em vários tipos de produtos.
- **System-on-a-Chip (SoC)**: Significando literalmente "sistema em um chip" é um chip que comporta quase todos, se não todos, os componentes de um computador. Se

assemelha muito a um microcontrolador, porém sendo mais versátil: desde um pequeno componente até um potente e diversificado computador, podendo ser utilizado para realizar mais tarefas e até mesmo alcançar um "nível" de propósito geral.

Há mais um termo que vale a pena mencionar: **Cortex microcontroller software interface standard (CMSIS)**. Trata-se de uma camada de abstração padronizada para microcontroladores baseados em ARM. Ou seja, uma grande biblioteca de funções em C!

2.3 O ARM Cortex-M0+

Um processador oferece a capacidade de controlar o sistema via software, o que permite uma configuração muito customizada do sistema. O microprocessador, ao contrário do microcontrolador, não tem todo o aparato necessário para o funcionamento do computador em seu interior, precisando ser conectado à memórias e dispositivos de E/S.

A linha de processadores Cortex®-M surgiu como o principal padrão para microcontroladores. O Cortex-M0+ estabelece-se como um processador de baixo custo e baixo consumo, porém muito poderoso e que tomou o mercado dominado pelos microcontroladores de 8-bit e 16-bit, já que até então os microprocessadores PIC tinham domínio sobre o mercado.

O microcontrolador KL25Z128, o qual contém o processador Cortex-M0+, utilizado na disciplina, por exemplo, possui configurações modestas: 48MHz de frequência de clock, 128kB de memória flash e 16kB memória SRAM. Para computadores de propósito geral e servidores essa configurações são insuficientes, porém, para a maioria dos sistemas embarcados, essa configurações são mais do que suficientes! A placa pode ser visualizada na figura 3.

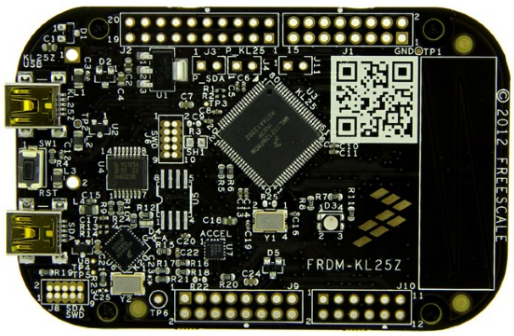

Figura 3: A Placa de Desenvolvimento Freedom FRDM-KL25Z

2.4 Programação dos Microcontroladores

A programação dos microcontroladores, os μC, é muito diferente da programação de computadores comuns. Isso porque nos nossos computadores nos programamos em cima de um Sistema Operacional que abstrai o hardware para nós. Em um microcontrolador temos que programar basicamente em cima do hardware, no que chamamos de ***bare metal***. Podemos usar sistemas operacionais de tempo real, chamados de **RTOS**, para facilitar nossa vida ou para aplicações com restrições de tempo. Esses sistemas operacionais tem menos requisições de memória e de processamento que sistemas operacionais comuns e oferecem gerenciamento e escalonamento de processos, as *tasks*.

A programação *bare metal*, em resumo, consiste em manipular endereços que referenciam os pinos do microcontrolador, bem como tratar as interrupções que irão invocar os métodos baseados em eventos externos. Os programas são, geralmente, feitos em C, C++ e até mesmo Assembly, sendo C o mais comum. A programação é feita em um computador de proposito geral, e o sistema final é passado ao microcontrolador por meio de alguma interface de comunicação, como a serial. O software, constituído das instruções do sistema, é salvo na memória flash do sistema.

Os sistemas embarcados geralmente entregam uma interface de usuário muito simples, como botões e LEDs, ou displays LCD simples. O mesmo pode ser dito de recursos internos. Recursos como sistema de arquivos e multitasking dificilmente são implementados em sistemas embarcados comuns. Os sistemas embarcados são projetados para serem simples, porém de forma a realizar suas tarefas com robustez.

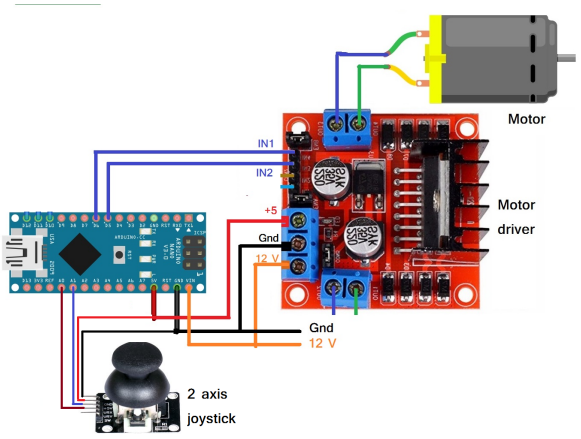

Figura 4: Protótipo de um Controlador de Motor DC com Joystick

Como exemplo de sistema embarcado simples temos o protótipo da figura 4. O microcontrolador da placa Arduino Nano irá utilizar controlar a velocidade e direção do motor DC baseado no movimento realizado no joystick. Esse sistema é muito simples, realizando apenas uma tarefa, portanto, podemos utilizar programação *bare metal*, ao invés de um **RTOS** robusto. Dessa forma iremos gravar um **firmware** na memória flash do μC para realizar as operações, ou seja, um software para realizar a operação do sistema.

3 Processadores de Sistemas Embarcados

Para cada aplicação existe um processador diferente, e existem muitos tipos de processadores. Para servidores temos processadores de alto desempenho, com altas frequências de clocks e que gastam muita energia. Já em sistemas embarcados temos sistemas menores, que devem gastar pouquíssima energia e que não precisam de muita performance.

3.1 Relação Custo X Performance

Quanto mais complexo o sistema mais recursos ele deve ter, como um sistema operacional ou periféricos adicionais. O preço, obviamente, aumenta de forma proporcional. Ao introduzir elementos ao chip aumenta-se a área de silício necessária e o gasto de energia. Sistemas Embarcados, em sua essência, possuem o mínimo de recursos necessários para operar.

Caso, por exemplo, seja necessário aumentar o *clock* para melhorar a performance tanto o custo do dispositivo quanto o gasto energético irão aumentar. A relação **Custo X Performance** está explicitada na figura 5:

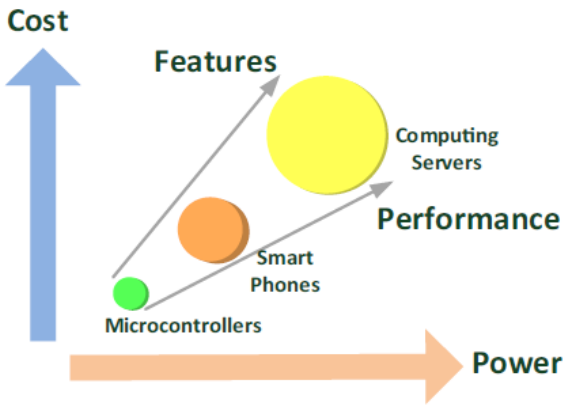

Figura 5: Relação Custo X Performance no Projeto de Processadores

Para cada projeto há um processador específico que irá atender suas necessidades. A escolha o tipo de processador no projeto do sistema é crítica. Os processadores ARM oferecem um equilíbrio entre performance, consumo de energia e recursos.

A segmentação dos tipos de processadores baseados em mercados facilita a busca do processador ideal para o projeto, onde cada processador oferece recursos que se adéquam melhor a um mercado específico. Cada processador tem suas vantagens e cabe ao projetista saber qual melhor atende as necessidades do projeto.

3.2 Visão Geral dos Processadores ARM

A ARM vem projetando processadores por mais de 20 anos, em sua maioria de 32-bits, que vem tomando o mercado de sistemas embarcados. O ARM7TDMI foi um marco na história dos processadores ARM, sendo eficiente do ponto de vista energético, com boa performance e alta densidade de código no modo Thumb®, o conjunto de instruções de 16-bits da ARM.

Um fato muito interessante que o livro trás é que, por volta de 2003, a ARM começou a segmentar o mercado dos processadores, criando três perfis específicos e adotando a marca Cortex®para nomear os novos processadores. Esses perfis viriam a ser essenciais no mercado de sistemas embarcados, já que cada perfil tem requisitos específicos de aplicações específicas.

Na figura 6 é possível ver um panorama geral dos microcontroladores ARM, desde 2003. Fica evidenciado na figura os três tipos de processadores ARM, os quais falaremos a seguir.

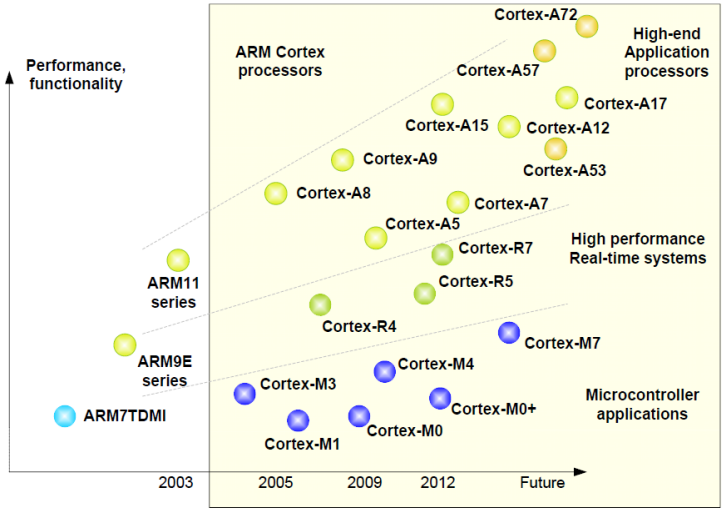

Figura 6: Visão Geral dos Processadores ARM

3.2.1 Linha de Processadores Cortex-A®

Os processadores dessa linha focam em alta performance, suportando sistemas operacionais avançados, como o Linux ou o Android. Dizemos que esses processadores são de **Aplicação**, já que podem ser usados para executar aplicações de usuário em um SO, ao invés de um simples firmware.

Fazendo jus à sua denominação, entregam longos pipelines e altas frequências de clock(mais de 1 GHz). Além disso, possuem MMU(Unidade de Gerenciamento de Memória), suportando memória virtual. São usados tipicamente em telefones móveis, tablets e até mesmo servidores de baixo consumo energético.

3.2.2 Linha de Processadores Cortex-R®

Estes focam em alta performance na modalidade de tempo real, podendo entregar até 1GHz de frequência de clock, sendo muito rápidos ao responder eventos de hardware. Estes processadores tem memórias caches, bem como memórias fortemente acopladas(memória interna ao chip com conexão dedicada), o que permite um comportamento determinístico para o tratamento de interrupção.

Contém, ainda, recursos que permitem uma confiabilidade melhor, podendo ser usadas em sistemas críticos, como automotivos e industriais.

3.2.3 Linha de Processadores Cortex-M®

Indicados para sistemas de baixo consumo energético, que não necessitem de muito processamento. Os processadores dessa linha tem um pipeline muito curto e baixas frequências de clock, decorrente das otimizações para poupar energia.

Os processadores dessa linha provém respostas à interrupções determinísticas, utilizando-se de um NVIC(Nested Vectored Interrupt Controller), um controlador de interrupções junto ao controlador de execução da CPU. O NVIC fornece gerenciamento de interrupções simples de programar, porém poderosas.

No geral, o Cortex-M é muito simples de usar, podendo ser programado quase inteiramente em C!

3.2.4 Resumo da Linha de Processadores ARM

Enquanto os processadores Cortex-A tem alta performance, eles não foram projetados para lidar com as rápidas respostas à interrupções, necessárias para sistemas de tempo real. Aí que entra o Cortex-R que, porém, consume muita energia e tem um projeto muito complexo, o que não acontece no Cortex-M, que é de baixo consumo, projeto simples e oferece performance aceitável.

Temos, ainda, processadores projetados especificamente para produtos de segurança, da linha SecurCore®. Um exemplo que o livro oferece é o SC000™, baseado no Cortex-M0, achado em chips de telefonia.

4 Conclusão

Com esse trabalho discutimos alguns conceitos iniciais de processadores ARM, microcontroladores e o mundo dos sistemas embarcados. Ficou evidenciado a grande importância, e impacto, que os processadores baseados em ARM tem no mercado.

Devido à seus vários produtos de sucesso que trouxeram grandes evoluções para o mundo dos sistemas embarcados os microcontroladores baseados em ARM, sobretudo na linha ARM Cortex-M, tornaram-se referência e padrão a ser seguido, como explicita o autor em [1].

Para exemplificar um sistema embarcado simples segue abaixo um código muito simples para a placa usada na disciplina, a FRDM-KL25Z, que pisca o LED azul embutido na placa:

```
1  #include <MKL25Z4.h>
2
3  #define BitSet(valor, bit) ((valor) |= (1 << bit)) // Aciona o bit em valor.
4  #define BitClr(valor, bit) ((valor) &= ~(1 << bit)) // Limpa o bit em valor.
5
6  int main(void) {
7    BitSet(SIM->SCGC5, 12);     // Habilita o clock para PORTD.
8    BitSet(PORTD->PCR[1], 8); // Seleciona a funcao de I/O para D1,
9    BitSet(PTD->PDDR, 1);       // Configura D1 como saida de dados.
10   while(1) {
11     BitSet(PTD->PDOR, 1);     // Liga o LED.
12     delayMs(500);             // Espera 0,5s.
13     BitClr(PTD->PDOR, 1);     // Desliga o LED.
14     delayMs(500);             // Espera 0,5s.
15   }
16 }
17
18 void delayMs(int ms) {
19   int i, j;
20   for(i = 0; i < ms; i++) {
21     for(j = 0; j < 7000; j++);
22   }
23 }
```

Listing 1: Código em C para piscar o LED Azul da placa FRDM-KL25Z

Temos então um microcontrolador, um computador, que tem como única função piscar um LED, porque foi para isso que o programei.

O código acima implementa dois macros para facilitar a manipulação de bits: **BitSet** aciona um bit e **BitClr** limpa um bit. Com essas duas operações podemos ligar o LED, com 1 no bit correspondente, e desliga-lo, com 0 no bit. O LED está localizado na porta D, no pino 1, portanto, **D1**. E é nisso que se baseia a programação de μC: **operações em bits**!

Na função **main()** configuramos alguns bits de controle e depois realizamos a operação de piscar o LED, no laço infinito. Manipulamos o bit correspondente do LED nas linhas 11 e 13 com os macros. Para espaçar os "estados" do LED usamos uma função de **delay de tempo**.

Referências

[1] Joseph Yiu. *The Definitive Guide to ARM(r) Cortex(r)-M0 and Cortex-M0+ Processors Second Edition.* 2015.