

Mutirão C - 5s - Eng. da Computação - Aula 1

Rafael Corsi

Fevereiro 2018

Table of Contents

- ▶ Visão Geral
 - ▶
- ▶ Primeiros passos
 - ▶ Primeiros passos, conectando.
 - ▶ Primeiros passos, rodando o exemplo.
 - ▶ Atmel Studio
 - ▶ Programando
- ▶ Funcionamento
 - ▶ Como isso funciona ?
 - ▶ Sistema detalhado
 - ▶ microcontrolador
- ▶ Firmware
 - ▶ main {.fragile}
 - ▶ imgShow
 - ▶ SuperLoop
- ▶ Clock
 - ▶ Energia em sistemas MOSFET
 - ▶ Frequência de operação 4.5Mhz
 - ▶ Alterando para 300 Mhz

Visão Geral

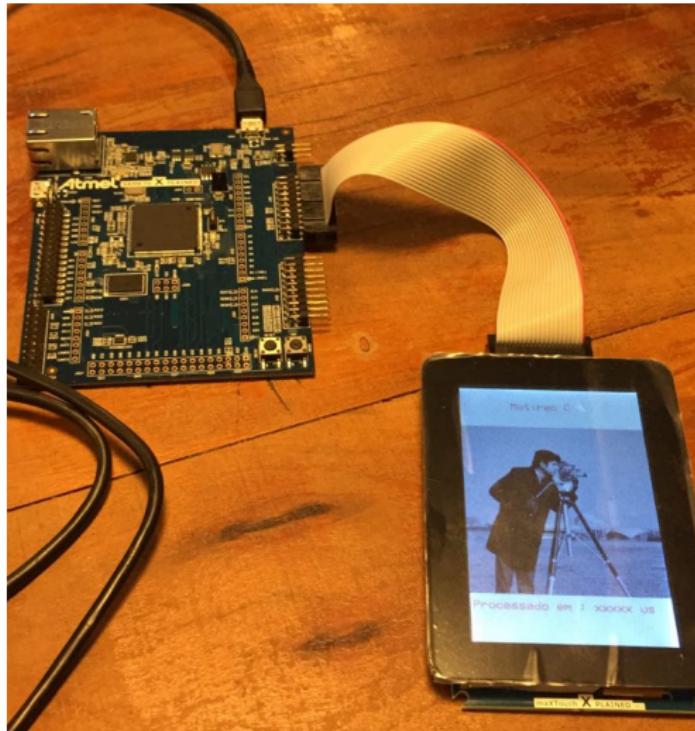


Figure 1: Resultado

Primeiros passos

Primeiros passos, conectando.

- ▶ Tome cuidado ao manusear a placa, não coloque ela sobre outros materiais.

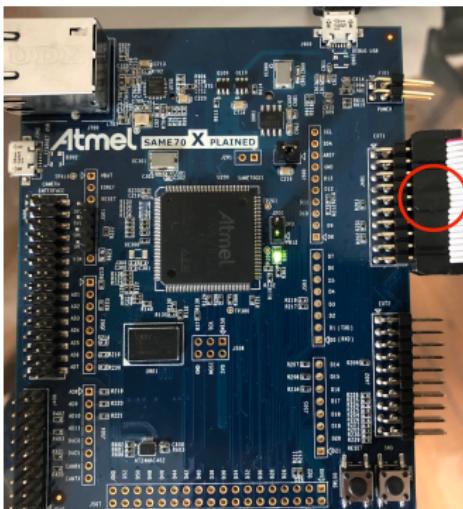
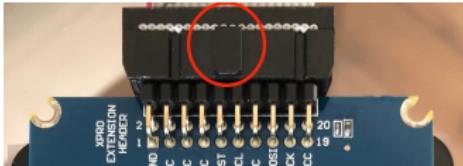


Figure 2: Ligação SAME70 Explained



Primeiros passos, rodando o exemplo.

1. Conectar o USB do programador no computador



Figure 4: USB DEBUG

2. Abra o projeto exemplo (SAME70-MutiraoC) localizado no repositório do mutirão :
 - ▶ github.com/insper/MutiraoC/dia-19-02/SAEM70-MutiraoC

Atmel Studio

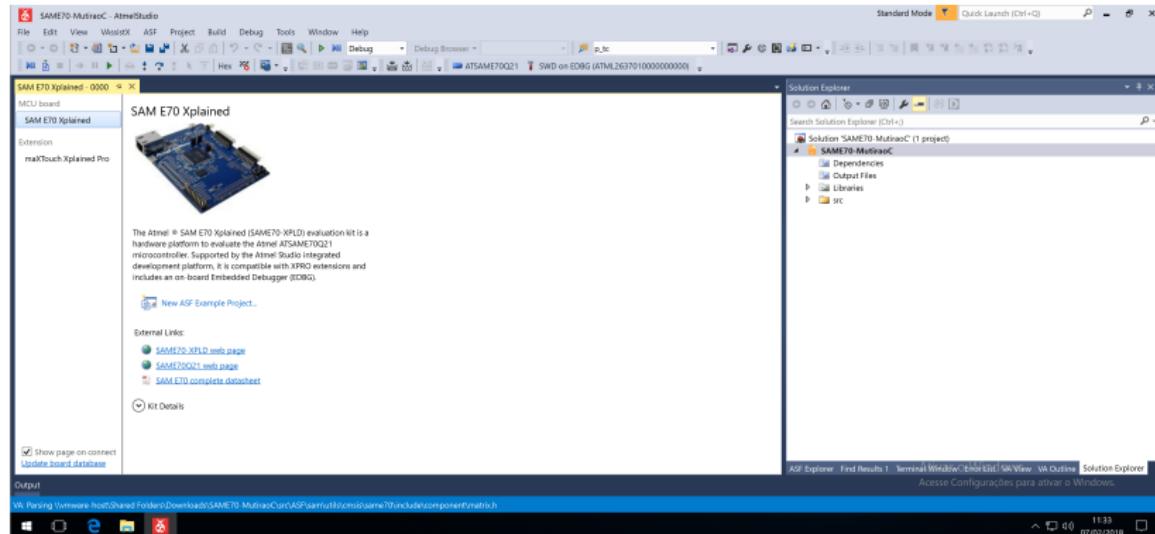


Figure 5: Atmel Studio

Programando

1. A etapa atual será a de embarcar o código exemplo no uC, para isso basta clicar em **Start Without Debug**



Figure 6: Embarcando

2. Uma vez embarcado o exemplo, o LCD deverá exibir uma imagem. A primeira imagem que aparece é a imagem original sem nenhum tipo de modificação, ao apertar o botão **SW0** do kit de desenvolvimento uma função (**imageProcess()**) é chamada e a imagem original é processada e exibida.



F

Funcionamento

Como isso funciona ?

Em uma visão mais geral podemos analisar o sistema como um kit de desenvolvimento e um display LCD :

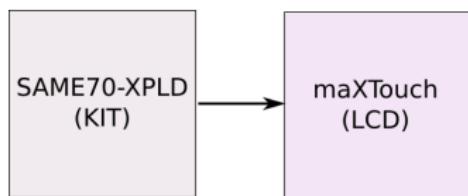


Figure 8: Diagrama simplificado

Sistema detalhado

Uma análise mais detalhada do projeto pode ser visto no diagrama de blocos a seguir :

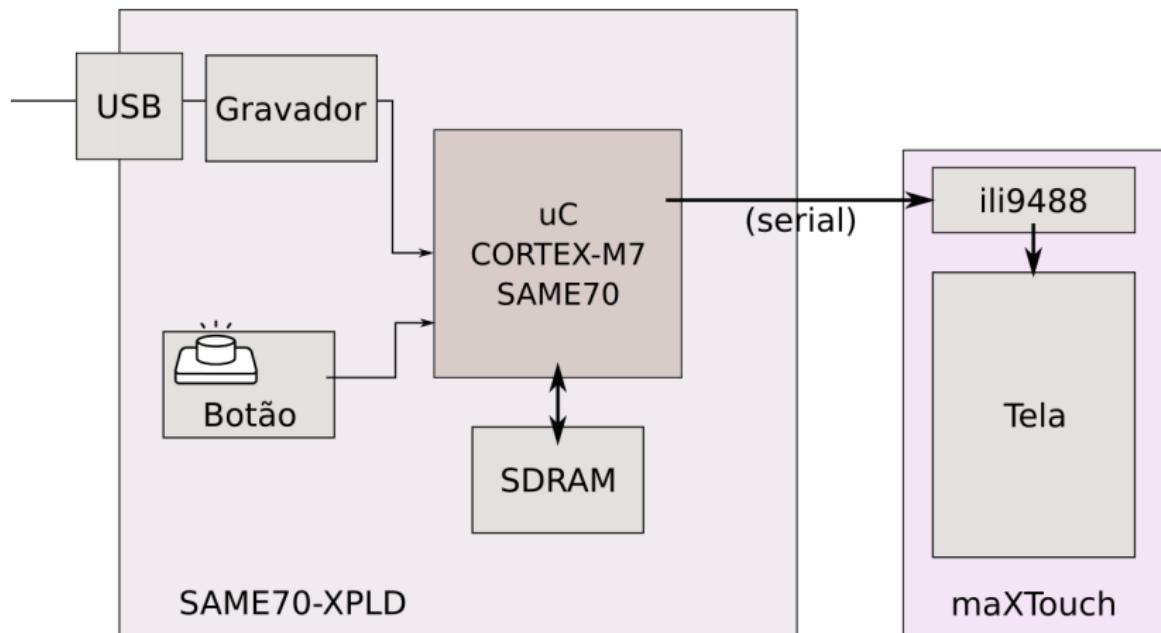
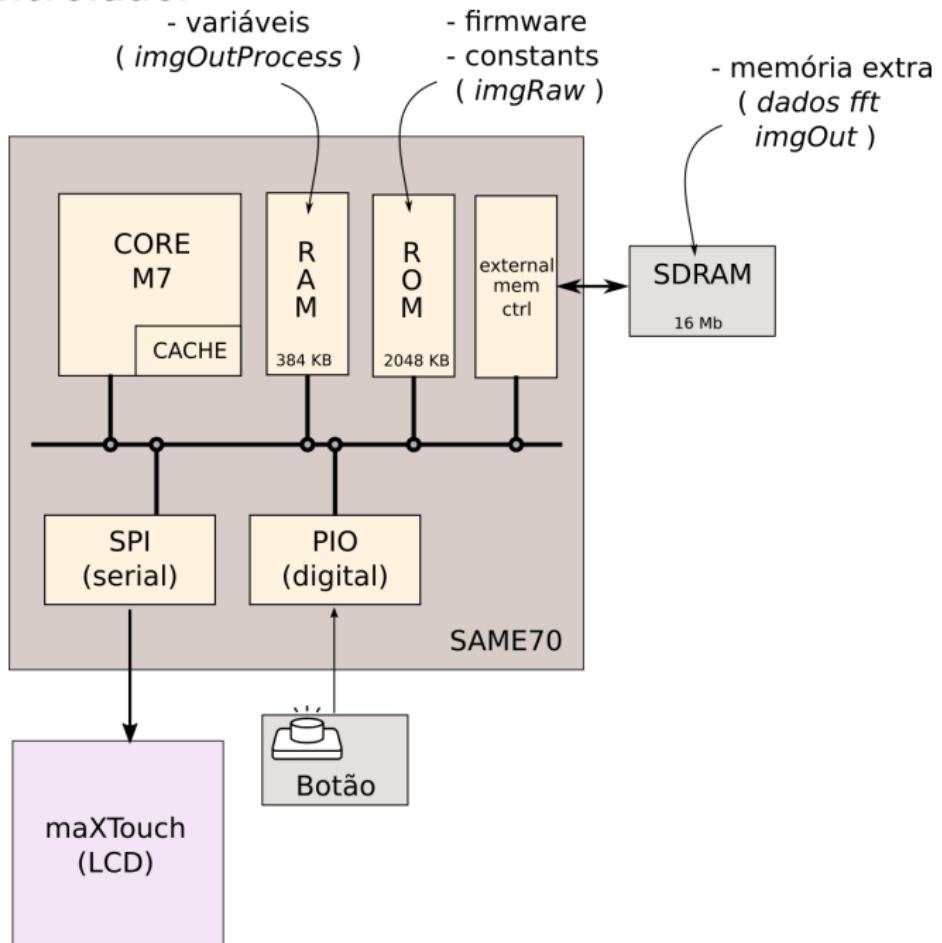


Figure 9: Diagrama detalhado

Microcontrolador



Firmware

main

```
int main(){

    uint32_t time;
    uint8_t  imageSelect = 1;

    // inicializa placa e seus perifericos
    initBoardMutirao();

    // exibe imagem original, tempo de processamento suprimido
    imgShow(imgRaw, 0);

    ...
}
```

imgShow

A função *imgShow* possui dois parâmetros : - a imagem *image[320][320]* a ser exibida - o tempo de processamento para ser exibido :

```
void imgShow(ili9488_color_t image[320] [320] ,  
             uint32_t time);
```

SuperLoop

Aplicações embarcadas não devem nunca retornar do main.

```
while (1) {  
  
    // se buttonFlag = 1 existe alteracao  
    // no estado do botao  
    if(buttonFlag){
```

Clock

Energia em sistemas MOSFET

É natural pensarmos que quanto maior a frequência do clock maior será o gasto energético de um sistema, lembre de camada física onde vimos que o gasto energético em sistemas baseados em MOSFET é :

$$P = \alpha C V_{DD}^2 f$$

- ▶ alpha : fator de chaveamento (influenciado pelo código)
- ▶ C : capacitância
- ▶ V : tensão de operação
- ▶ f : frequência de chaveamento

Portanto quanto maior a frequência de chaveamento maior será o gasto energético do sistema, mas um detalhe deve ser levado em consideração : quanto maior a frequência do clock mais rápido uma tarefa é realizada e mais rapidamente um sistema pode entrar em modo de baixo consumo energético *sleep mode, suspensão,*

Frequência de operação 4.5Mhz

src/config/conf_clock.h

```
// ===== Processor Clock (HCLK) Prescaler Options
//(Fhclk = Fsys / (SYSCLK_PRES))
#define CONFIG_SYSCLK_PRES           SYSCLK_PRES_64
...
...
```

- ▶ Onde Fsys é equivalente a 300Mhz, na configuração inicial a frequência do processador é $300\text{Mhz}/64 = 4.5 \text{ MHz}$, altere esse trecho para a forma a seguir, selecionando a frequência do principal do uC para 300Mhz.

Alterando para 300 Mhz

src/config/conf_clock.h

```
// ===== Processor Clock (HCLK) Prescaler Options      (Fhclk
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3
```

Programe e veja a diferença

Aplicações

Similar ?



Áreas em destaque



Automotive

Keeping you safe at 70mph. Higher performance system platforms for connectivity, advanced recognition and decision intelligence will enable coming generations of assisted and automated driving.

[Read more](#)

Healthcare

Healthcare is becoming personal. Pervasive mobile computing, highly-efficient embedded intelligence and secure technology are driving new interaction capabilities and advancing diagnostic and treatment options.

[Read more](#)

Infrastructure

Leading compute density solutions are transforming infrastructure; Building an end-to-end enterprise application platform with increased intelligence for the delivery of fast, agile, service-based 5G networks, and delivering a paradigm shift in data center workload efficiency.

[Read more](#)

Internet of Things

Enabling the foundations of IoT. Scalable embedded intelligence and software provides an agile, secure and flexible end-to-end platform for sensing, controlling and delivering business insights.

[Read more](#)

Mobile Computing

ARM low-power technology is the foundation of the mobile computing era, providing central processing capabilities as well as connectivity, and the industry's roadmap for evolving enhanced contextual, interactive and visual user experiences.

[Read more](#)

Smart Homes

The combination of an integrated mobile experience and increasing intelligence and connectivity across devices in the home is creating opportunities to transform how we interact with our living spaces. More intuitive and interoperable systems save energy, increase security and safety, and reduce costs.

[Read more](#)

Wearables

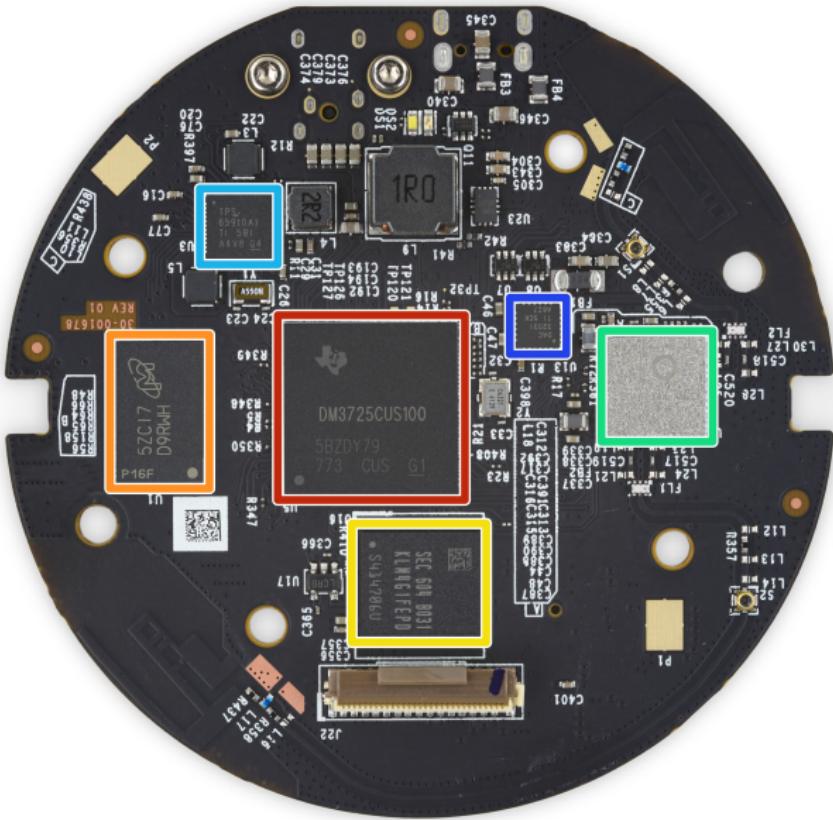
Redefining personal compute. Realizing new forms of interactions with our physical and digital worlds with highly-efficient processing for even the tiniest of devices.

[Read more](#)

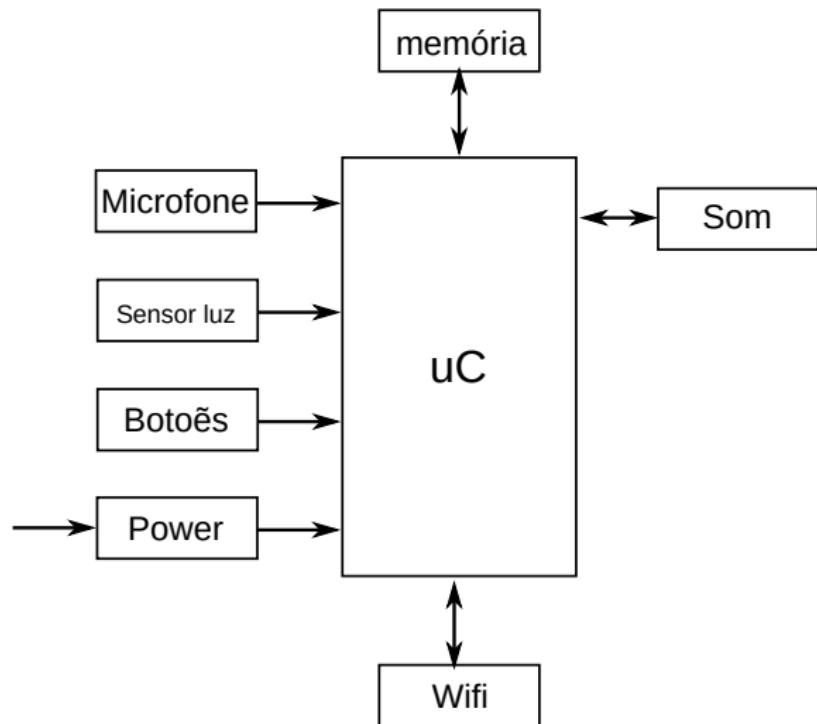
Exemplo 1. Amazon Echo Dot



Echo Dot



Echo - Diagrama



Echo - Diagrama detalhado

