

# Computação Embarcada - Tutorial

## *Desbravando o Atmel Studio*

Rafael Corsi  
rafael.corsi@insper.edu.br

20 de fevereiro de 2017



*Entregar na 6ª em formato PDF via github.*

1. Criar softwares para microcontroladores utilizando suas especificidades (periféricos/ low power);
2. Avaliar e melhorar soluções embarcadas integrando hardware/software levando em conta adequação a uma aplicação;
3. Integrar em um protótipo hardware, software básico, sistema operacional de tempo real e módulos de interfaceamento com usuários, de comunicação e de alimentação;
4. Compreender as limitações de microcontroladores e seus periféricos;
5. Buscar e analisar documentação (datasheet) e extrair informações

## 1 Atmel Studio

Atmel Studio <sup>1</sup> é uma IDE gratuita baseado no visual Microsoft Visual Studio, desenvolvida pela Atmel fornece todas as ferramentas necessárias para o desenvolvimento de projetos utilizando os microcontroladores fabricados por eles.

---

<sup>1</sup><http://www.atmel.com/tools/ATMELSTUDIO.aspx>

## 2 Exemplo 1 - Pisca Led

Iremos nesse exemplo executar um projeto "Pisca Led", o pisca led é normalmente o primeiro firmware que se embarca em um microcontrolador, serve basicamente para testar o fluxo de programação e as ferramentas de desenvolvimento, bastante similar ao "Hello World".

### 2.1 Carregando um exemplo

Conecte o kit de desenvolvimento via o computador na placa via o *debug USB*.

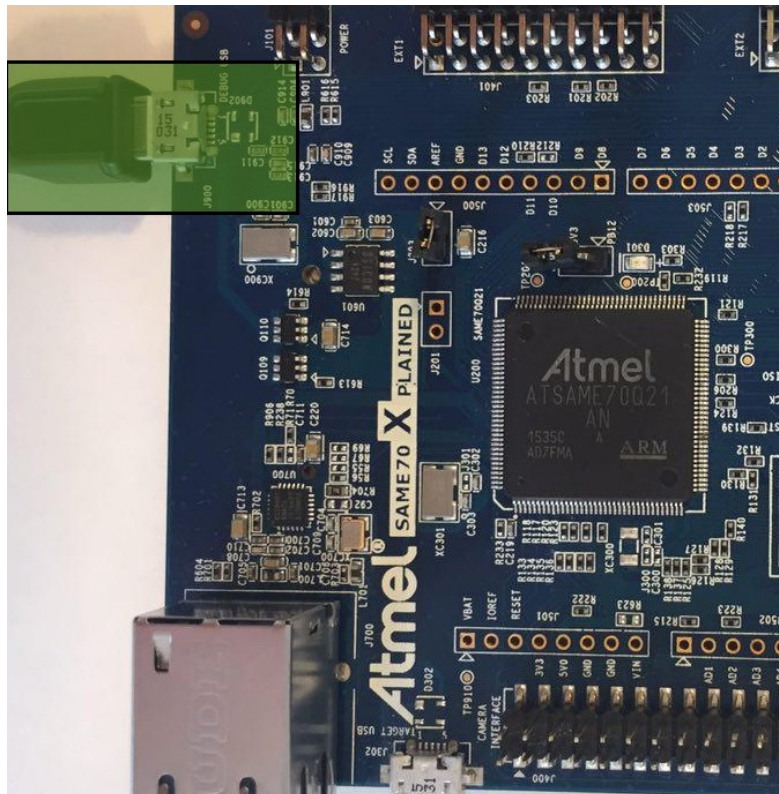


Figura 1: Detecção do kit de desenvolvimento

Após conectado abra o Atmel Studio. Nessa etapa o Software deve detectar qual kit de desenvolvimento está conectado ao computador, essa tela também exibe os periféricos que podem vir a estar plugados na placa (WIFI, LCD, BLT, ...).

Para criarmos um projeto a partir de um exemplo, clique em :

*File-> New -> Example Project*

A nova tela que irá aparecer contém todos os exemplos de código disponibilizados pela Atmel, para diversas placas e microcontroladores, estamos interessados para exemplos do SAME-70. Digite na busca *Search for example projects* o nome do kit: *same-70*

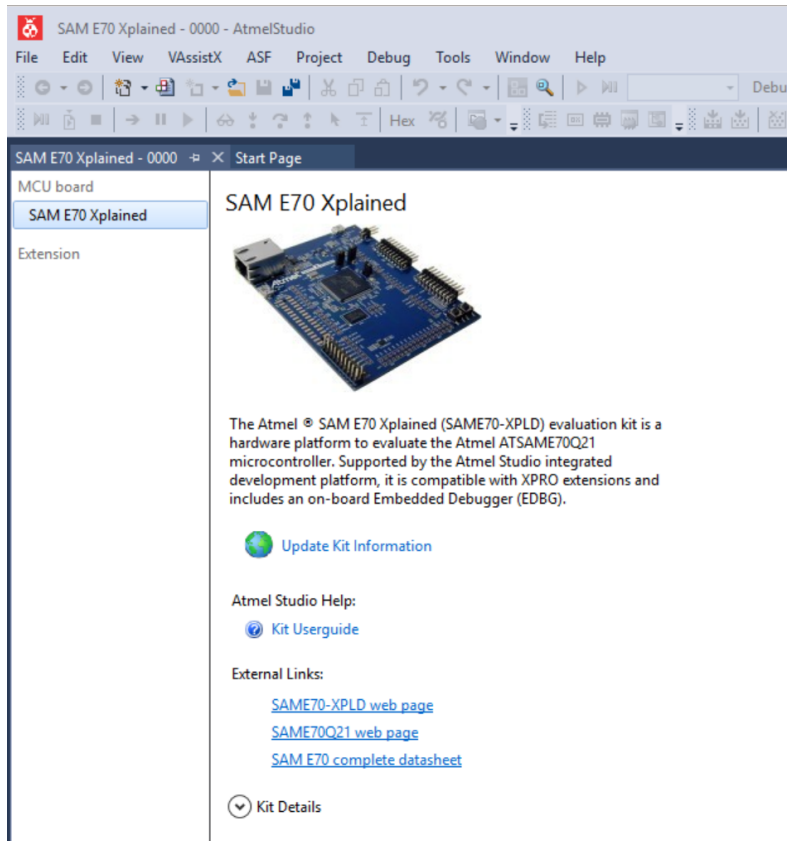


Figura 2: Detecção do kit de desenvolvimento

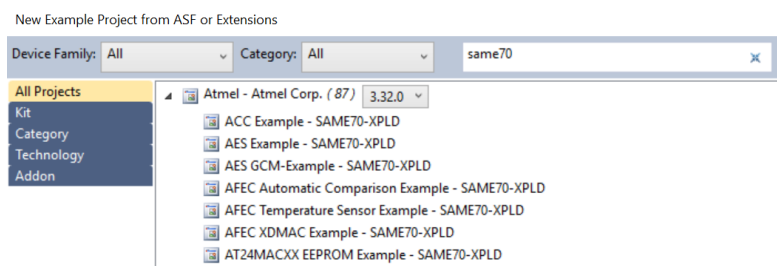


Figura 3: Exemplos

## 2.1: Exemplos

De uma olhada nos exemplos fornecidos, algum interessante ?

A lista de exemplos é vasta, de uma olhada no que existe de exemplos. Procure pelo

exemplo intitulado de : *Getting-Started Application on SAM - SAME70-XPLD*. Esse exemplo possui o descritivo a seguir :

*This demonstration program makes two LEDs on the board blinking at a fixed rate. [Getting-Started Application on SAM - SAME70-XPLD - ATSAME70Q21]*

---

Com o exemplo selecionado, nomeie para Aula5-Pisca-Led e salve-o em uma pasta Aula5 dentro do seu repositório :

- Project Name : Aula5-Pisca-Led
- Location : repositorio/Aula5/

! Com o exemplo criado e sem nenhuma alteração, faça um commit no repositório.

### 2.1.1 Barra superior

As barras superior do AtmelStudio possuem atalhos para a gravação e depuração do código.

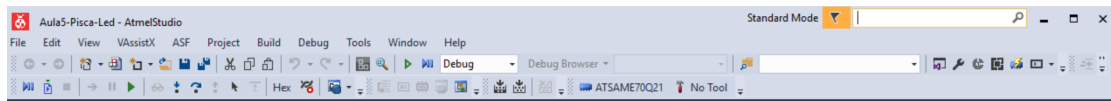
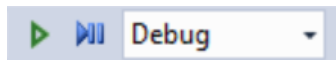


Figura 4: Barra Superior

Podemos programar o código de duas maneiras: com ou sem debug. Sem debug o código irá executar sem que possamos ver seu estado interno ou bloquearmos a execução. Já no modo de debug, temos acesso os valores interno.

A escolha entre um ou outro modo se da por esses botões :



### 2.1.2 Solution Explorer

A janela na lateral direita exibe todo material de código carregado no projeto, a Atmel optou por sempre realizar uma cópia das bibliotecas para dentro do projeto, tornando assim o projeto mais fácil de ser compartilhado. Outros fabricantes optam por deixar as bibliotecas em uma pasta global, onde todos os demais projetos podem acessar.

Uma solução pode ter mais de um projeto.

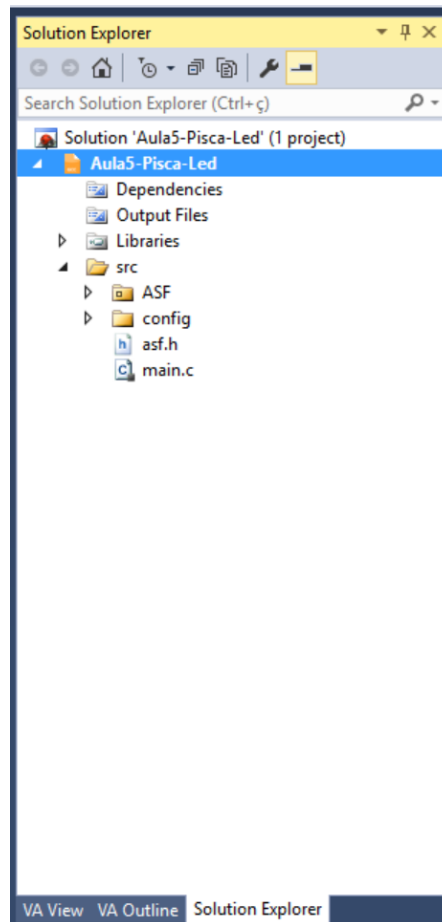


Figura 5: Solution Explorer

## 2.2: Códigos

Explore os códigos contidos nesse projeto.

### 2.1.3 ASF Wizard

ASF é um acrónimo para Atmel Software Framework, uma biblioteca de códigos mantido pela Atmel para a rápida prototipação de projetos. Nessa janela selecionamos as bibliotecas que serão copiadas e utilizadas dentro do projeto.

## 2.3: ASF

Quais são as bibliotecas carregadas no pisca led ?

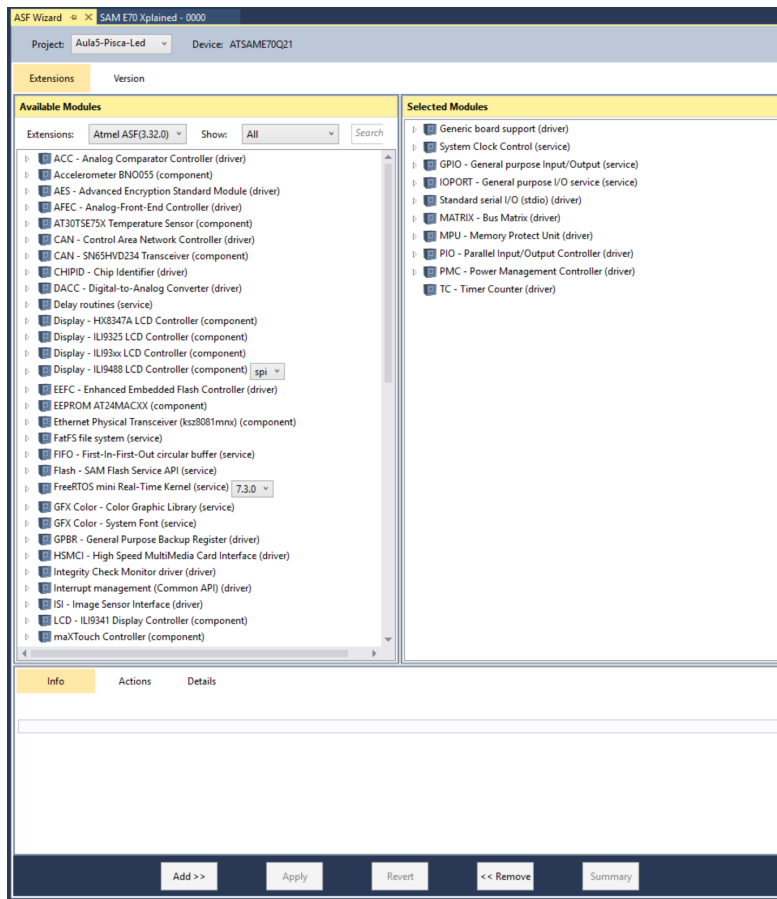

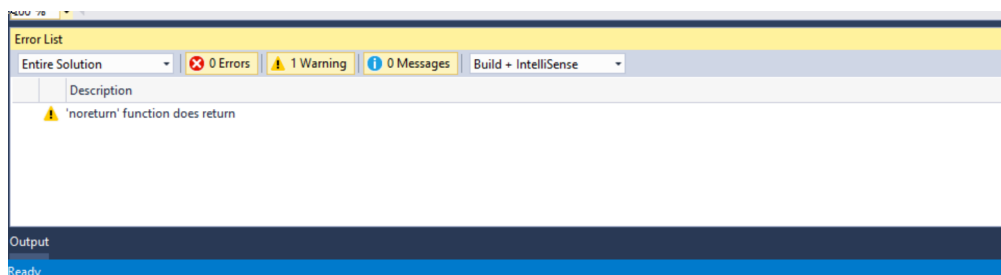


Figura 6: ASF


## 2.2 Embarcando

Para compilarmos e embarcamos o projeto no kit de desenvolvimento, basta clicar no ícone  que a ferramenta será responsável por compilar o código, e transferir o mesmo para o kit de desenvolvimento. Qualquer erro de compilação seria exibido na parte inferior da tela, intitulada de *output*:




### 3 Main.c

Abra o arquivo main.c, esse é um código feito pela atmel e que funciona em diversos kits de desenvolvimento diferente, por isso é muito denso para um simples pisca LED. Para termos uma visão mais geral do que compõem esse projeto, podemos compactar o código pressionando as teclas : CTR+M ,CRT+L.

Com isso podemos verificar mais facilmente as funções que compõem o firmware. Uma outra maneira de entendermos o que está sendo executado é realizarmos um step-by-step do código, ou seja, embarcamos dessa vez no modo de debug e verificamos quais partes do código estão sendo executadas. Para isso clique agora no debug .

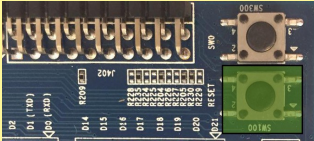
Launch Failed

 Timed out waiting for device to reset, would you like to continue waiting for the reset event ?

Yes

No


! O programador possui um erro quando executado via máquina virtual, caso a janela apareça clique em YES e logo em seguida aperta uma vez o botão da placa referente ao reset.



Quando programado, o código será interrompido logo em sua primeira linha :

```
...
// [main]
int main(void)
{
    ///! [main_step_sys_init]
    /* Initialize the SAM system */
    sysclk_init();
    board_init();
    ///! [main_step_sys_init]
```

Figura 7: Output do compilador

Para controlar os passos da depuração, utilize os comandos da barra superior : . Execute o código passo a passo utilizando o step-over e step-into.

#### 3.1 Entendendo o código

As perguntas a seguir irão ajudar entender o código.

### 3.1: Qual a frequência de operação do uC ?

Utilize a função `sysclk_get_cpu_hz()` para obter qual a frequência de operação do uC.

### 3.2: Frequência LED - parte 1

Qual a frequência com que o LED pisca ?

### 3.3: Frequência LED - parte 2

Altere a frequência do LED para o dobro da atual.

### 3.4: Frequência LED - parte 3

Qual parte do código é responsável por gerar a frequência com que o LED pisca ? Como isso funciona ?

### 3.5: Push button SW0 - parte 1

Executando e analisando o código, informe qual o uso do botão SW0 ?

### 3.6: Push button SW0 - parte 2

Como é feita a detecção da mudança de estado do botão ?  
Que função é responsável por lidar com isso?

### 3.7: Push button SW0 - parte 3

Faça com que cada vez que a frequência do led mude toda vez que o botão for pressionado

### 3.8: I/Os

Identifique os pinos responsáveis por acionar o LED e por ler o botão, classifique os pinos como entrada ou saída e indique qual função é responsável por cada um.

Item	Pino	PIO	Entrada/Saída	Função (handler)
Botão				
LED				



### 3.9: Diagrama de blocos

Esboce um diagrama de blocos do projeto Pisca-LED.

### 3.10: Fluxograma

Esboce um fluxograma do firmware.

### 3.11: Organizando o código

Remova os trechos de código não utilizados, limpe também o excesso de comentários, adicione comentários melhores.