UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
SISTEMAS MICROPROCESSADOS

Atividade Prática 01 - Introdução ao Ambiente de Desenvolvimento

Professor(a): Ricardo Jardel Nunes Silveira

Objetivos

- Familiarizar o aluno com as características elétricas e funcionais do kit de desenvolvimento;
- Apresentar uma visão geral da ferramenta STM32CUBEMX e suas funcionalidades;
- Compilar e executar um programa simples usando a IDE Atollic TrueSTUDIO.

Introdução

Nos últimos anos, os microcontroladores ARM Cortex tornaram-se bastante populares devido ao seu baixo custo, consumo reduzido e alta velocidade de operação. Os microcontroladores ARM são muito fáceis de usar quando comparados a processadores de 8 bits, porque possuem um mapa de memória simples e linear. Além disso, os fabricantes disponibilizam um vasto material de apoio aos projetistas.

Devido à grande variedade de recursos incorporados a este processador ele é usado em uma grande variedade de aplicações, tais como: indústria automotiva, computadores, impressoras, equipamentos de comunicações de dados, celulares, eletrodomésticos, enfim, ele é usado em todo tipo de dispositivo eletrônico.

Durante nosso curso usaremos um kit de desenvolvimento baseado no microcontrolador ARM CORTEX M4 que é um microcontrolador de 32 bits. Cabe ressaltar que este microcontrolador é uma versão do microcontrolador ARM-CORTEX-M3 aprimorada pela adição de um DSP e vários dados de instrução única (SIMD) e (opcionalmente) um hardware de ponto-flutuante unitário (FPU).

Atividade Prática

1. Medição de Sinais e Configurações iniciais (Alimentação, clock e inicialização)

1.1 Alimentação

O aluno deve antes de iniciar qualquer prática se certificar que o kit de desenvolvimento está operando normalmente e para isso deve começar medindo os sinais de alimentação da placa. Portanto, é necessário medir o regulador U800 de 5 Volts e o regulador U801 de 3,3 Volts. As figuras a seguir ilustram o circuito da fonte de alimentação do kit e o detalhe das medidas.

Figura 01 - Circuito da fonte de alimentação. Note que o circuito é divido em três partes. Na primeira temos o conector da fonte e um LED indicator enquanto nas outras duas temos reguladores de tensão que convertem uma tensão de 9V (ou similar) em 5V e 3,3V.

Medição na Fonte de 5 Volts

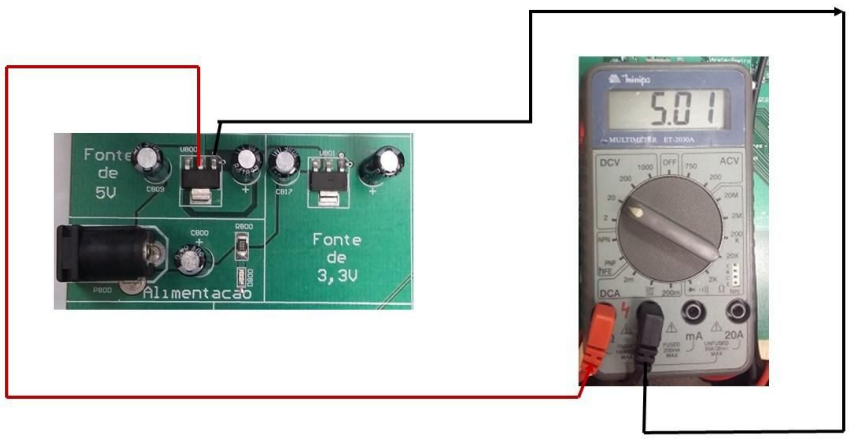


Figura 02 - Processo de medição da fonte de alimentação de 5V utilizando um multímetro.

Medição na Fonte de 3,3 Volts

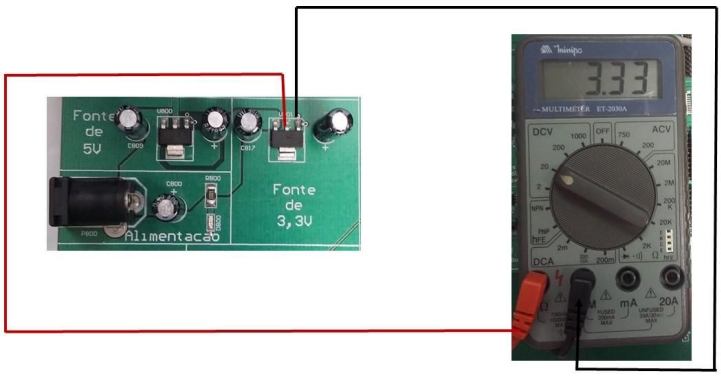


Figura 03 - Processo de medição da fonte de alimentação de 3,3V utilizando um multímetro.

1.2 Clock do Sistema

Após a execução das medidas na fonte de alimentação e diretamente no processador e tendo certificado o bom funcionamento da placa, agora iremos medir o sinal do clock e garantir que o mesmo está presente nos pinos correspondentes do processador.

O processador Arm cortex permite três formas diferentes de configuração do clock do sistema, mas para efeito de simplificação usaremos nesta prática apenas o modo high speed external clock signal (HSE) gerado por cristal externo, ficando a cargo do aluno a pesquisa e prática dos outros modos de configuração (Ver no manual de referência do microcontrolador na página 114 e seção 6.2). A figura a seguir ilustra o sinal de clock que deve ser visto pelo aluno nos pinos RCC_OSC_IN (PH0) e RCC_OSC_OUT (PH1) do processador.

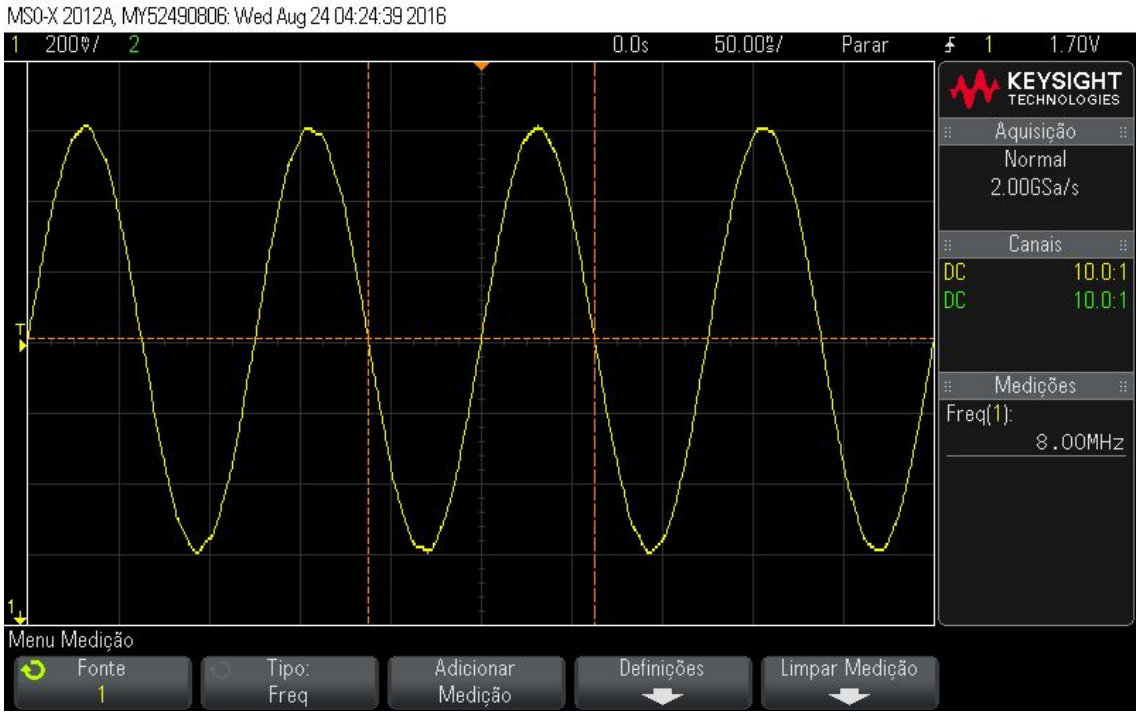


Figura 04 - Sinal de clock capturado utilizando um osciloscópio digital.

1.3 Inicialização

Devido ao seu mapa de memória fixa, a área de código começa a partir do endereço 0x0000 0000 (acessado através dos barramentos ICODE / Dcode), enquanto a área de dados (SRAM) começa a partir do endereço 0x2000 0000 (acessado através do bus de sistema). O Cortex®-M4 com FPU CPU sempre busca o vetor de reset no barramento ICODE, o que implica ter a área de inicialização disponível apenas na área de código (normalmente, a memória Flash). microcontroladores STM32F446xx implementam um mecanismo especial para serem capazes de inicializar a partir de outras memórias (como a SRAM interna).

No STM32F446xx, três modos diferentes de inicialização podem ser selecionados através do BOOT [1: 0] pinos como mostrado na Tabela 1.

BOOT1	BOOT0	Modo de Inicialização
X	0	Memória do Flash Principal
1	0	Memória do Sistema
1	1	SRAM Incorporada

Tabela 1 - Modos de inicialização.

Verifique se o Kit de desenvolvimento utilizado por sua equipe está configurada como BOOT0 = 0 e BOOT1 = 0.

2. Criar um projeto usando a ferramenta STM32CubeMX

O nosso primeiro exemplo prático será o clássico “Hello, world” que exibiremos na tela do computador. Para realizar essa tarefa enviaremos os dados pela porta serial. Para facilitar o ciclo do projeto usaremos a ferramenta STM32CUBEMX. A partir de agora vamos configurar o microcontrolador e gerar um código fonte em linguagem C para funcionamento de nosso projeto a partir dos seguintes passos:

1. Abra a ferramenta através do ícone (**STM32CubeMX**) presente na área de trabalho e escolha a opção New Project. Então será aberta uma tela com duas abas e várias opções a serem configuradas;

2. Na aba **MCU Selector** em series escolha a opção STM32F4;

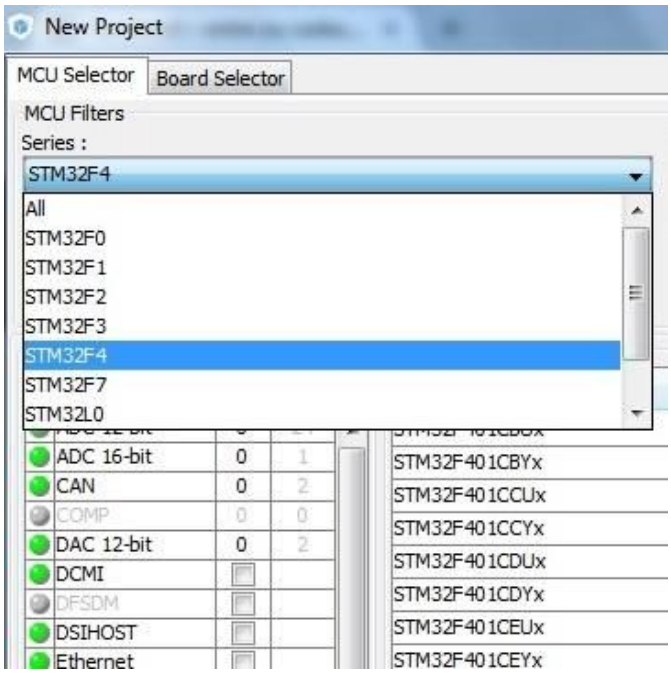


Figura 05 - Seleção da família F4 de microcontroladores.

3. Em seguida, na aba **MCU Selector,** e em Lines selecione STM32F446;

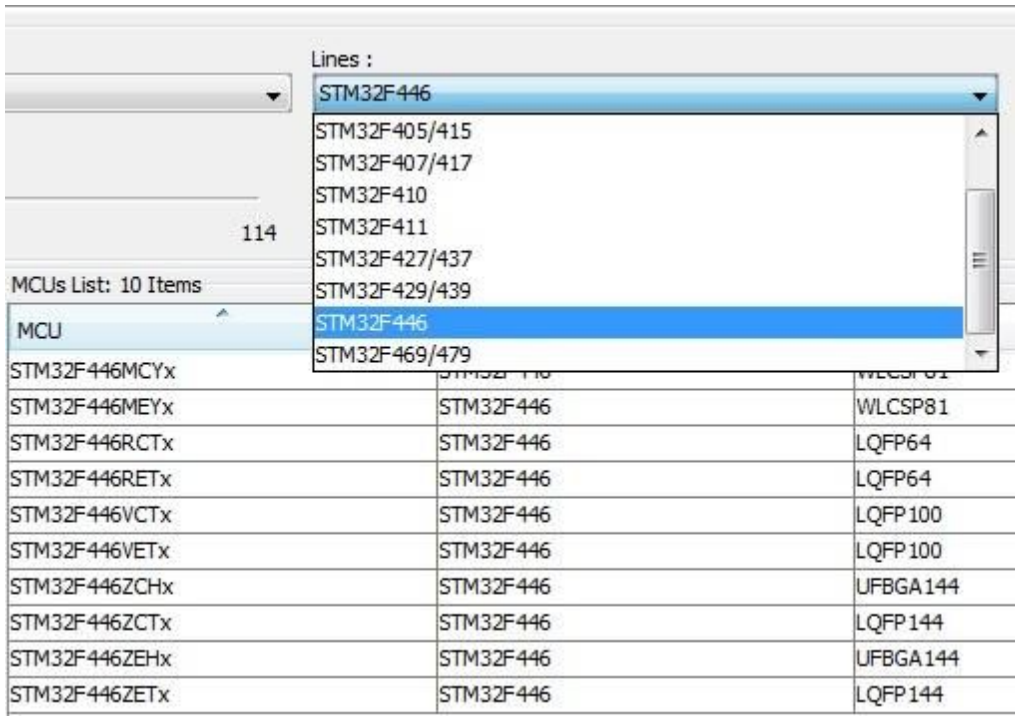


Figura 06 - Seleção do modelo F446.

4. Ainda na aba **MCU Selector** vamos definir o tipo de encapsulamento do microcontrolador**,** em **package** selecione a opção **LQFP100**;

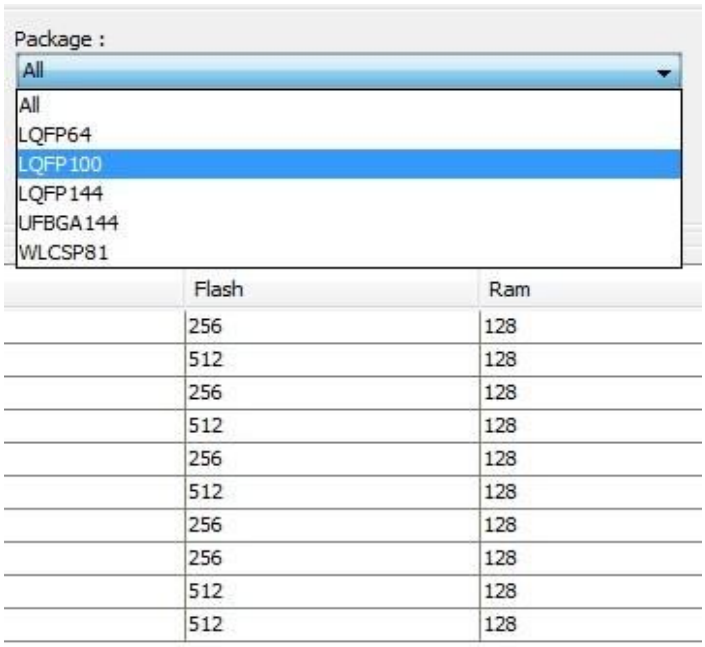


Figura 07 - Seleção do modelo encapsulamento LQFP100.

5. E finalmente após filtrarmos e configurarmos todos os itens da aba **MCU Selector,** identificamos no espaço reservado **MCU's List** apenas dois itens, então escolha a

opção referente ao microcontrolador usado em nosso Kit de Desenvolvimento, que é o **STM32F446VETx** e aperte **OK;**

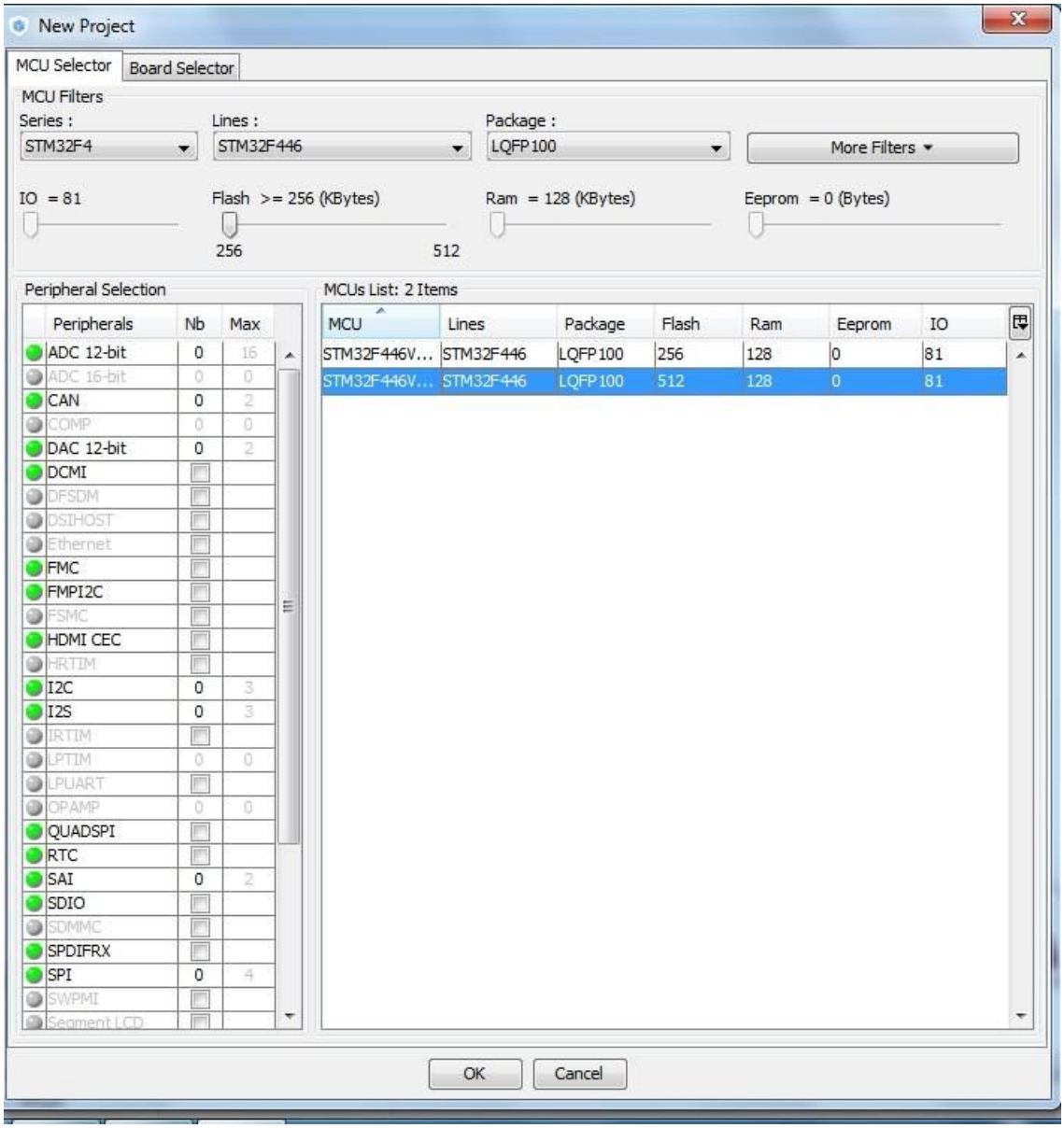
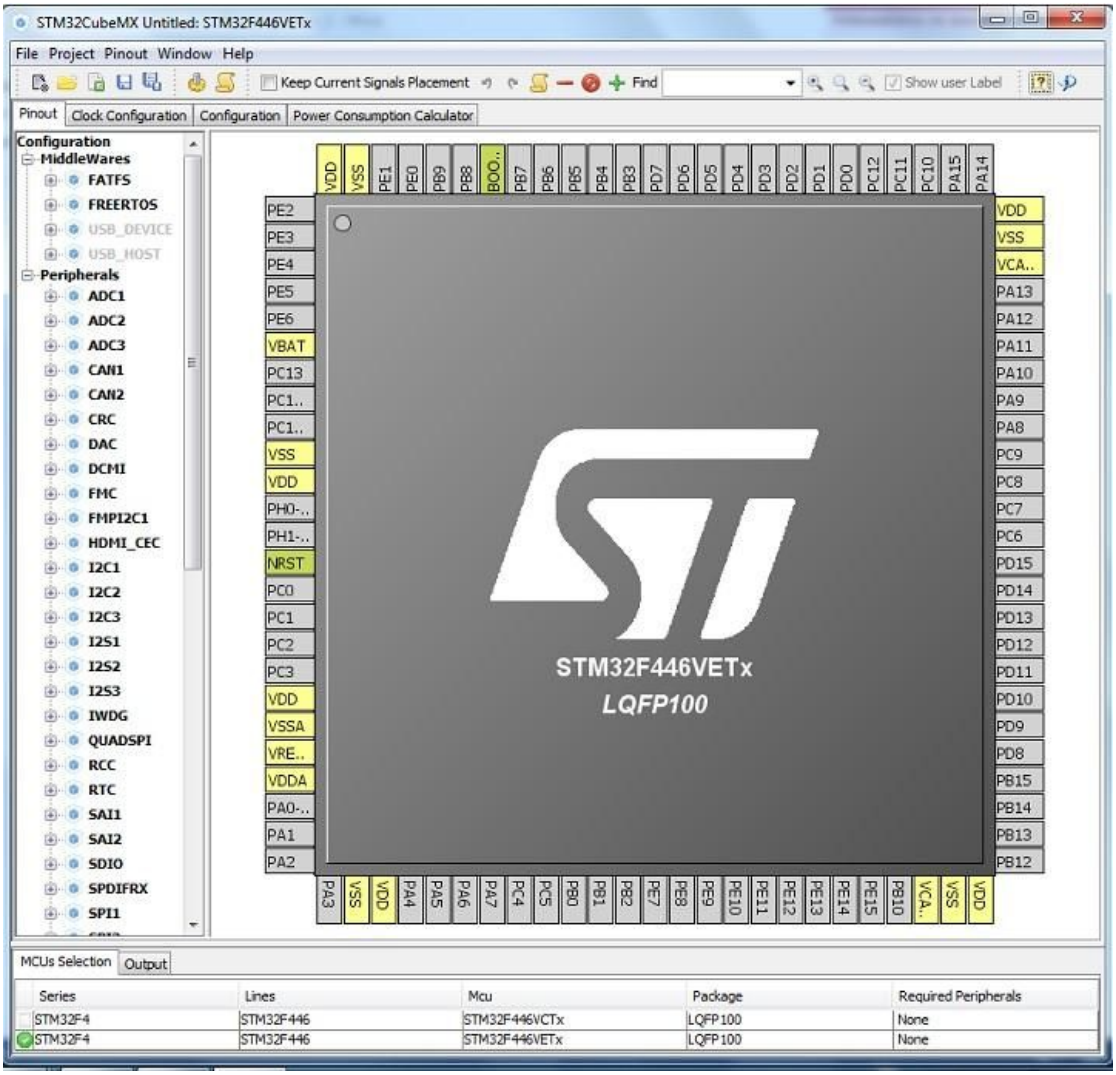


Figura 08 - Seleção do *part number* correto.

6. Se todos os passos foram seguidos corretamente o **STM32CubeMX** apresentará uma janela com a imagem do microcontrolador pertencente ao kit de desenvolvimento em estudo, onde poderemos configurar todos pinos de nosso projeto:



Q’

Figura 09 - Tela inicial do STM32CubeMX após a seleção do microcontrolador desejado.

7. Agora temos o microcontrolador a nossa disposição e podemos configurá-lo para enviar dados via **UART**. Consulte o esquemático do Kit de Desenvolvimento e veja que os pinos que correspondem a **UART e** ao clock externo são:

   a) PA9 – **UART1_TX;**
   b) PA10 - **UART1_RX;**
   c) PH0 - **OSC_IN**;
   d) PH1 - **OSC_OUT.**

Perceba que os pinos do microcontrolador são multiplexados, portanto ao selecionar um pino será exibida uma lista de opções. Então, ao clicar em PA9 escolha a opção **UART1_TX** e para os demais pinos consulte o esquemático ou faça como descrito acima. Veja na figura a seguir.

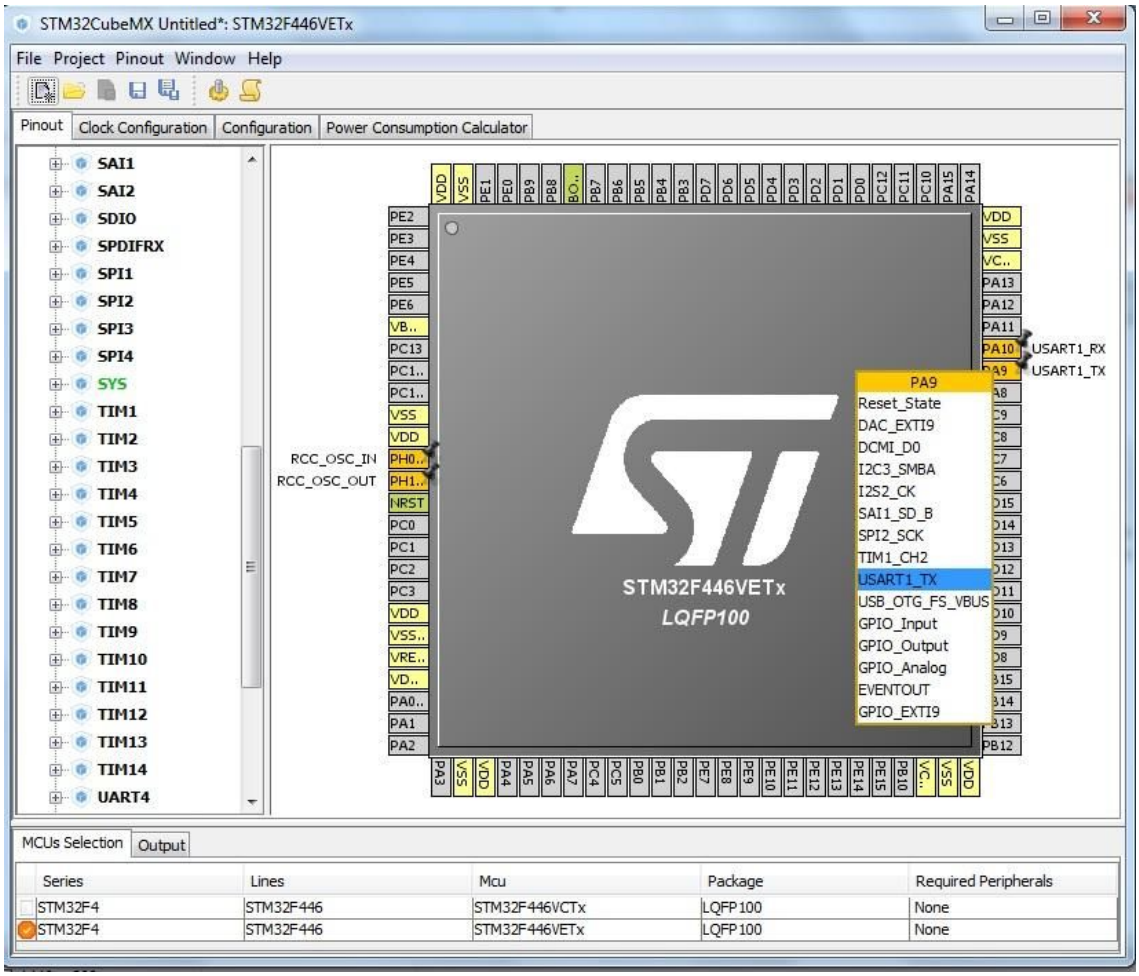


Figura 10 - Seleção das funções de cada um dos pinos que serão utilizados no projeto.

8. Observe que logo abaixo da aba pinout existe uma árvore que disponibiliza todos os periféricos do microcontrolador. Vamos chamá-la de árvore de configuração de periféricos. Procure nesta árvore a opção que habilita o clock (**Rcc**) e click em High Speed Clock (**HSE**) e escolha Cristal Ceramic Ressonator para habilitá-lo.

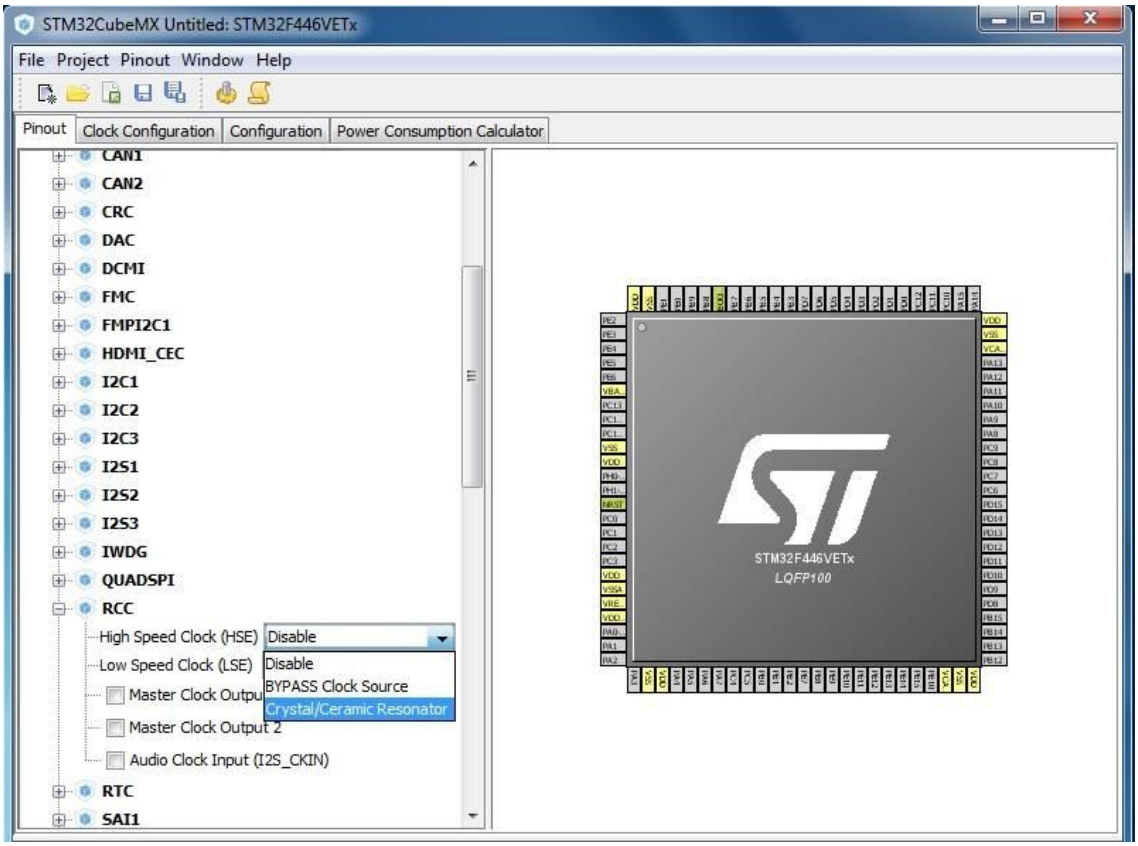


Figura 11 - Habilitação do cristal externo como fonte de clock de alta velocidade (HSE).

9. Procure na árvore de configuração de periféricos a opção **USART1** e escolha Mode *Asynchronous*;

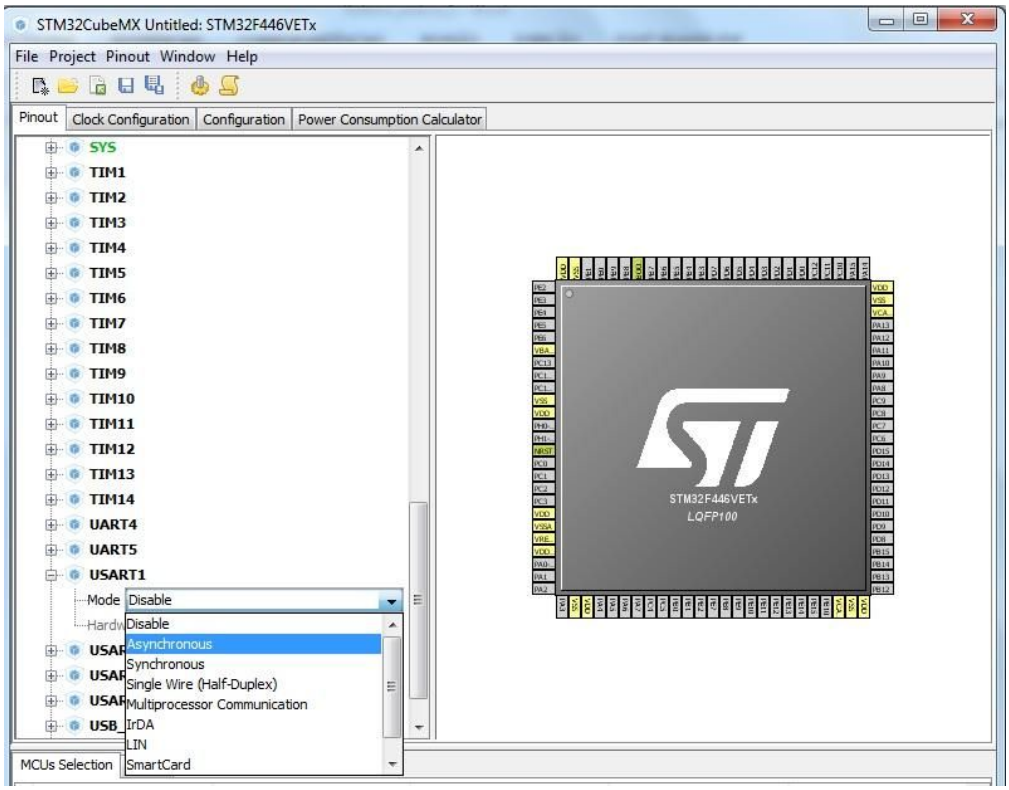


Figura 12 - Habilitação do modo assíncrono para o periférico USART1.

10. Na inicialização um oscilador RC interno de 16 MHz é selecionado como o clock da CPU padrão. A aplicação pode então selecionar como clock do sistema, quer o oscilador RC ou uma fonte de clock de 4-26 MHz externo. O nosso kit de Desenvolvimento utiliza um clock externo de 8 MHz. Para configurá-lo acesse a aba Clock Configuration, observe no circuito de clock o dispositivo **PLL Source Mux** e selecione a entrada **HSE** e o dispositivo **System Clock Mux** mude também a entrada para **HSE**. Finalmente, Na caixa Input frequency que por default é 25 MHz mude para 8 MHz. Os pontos de mudança estão realçados em vermelho na figura a seguir.

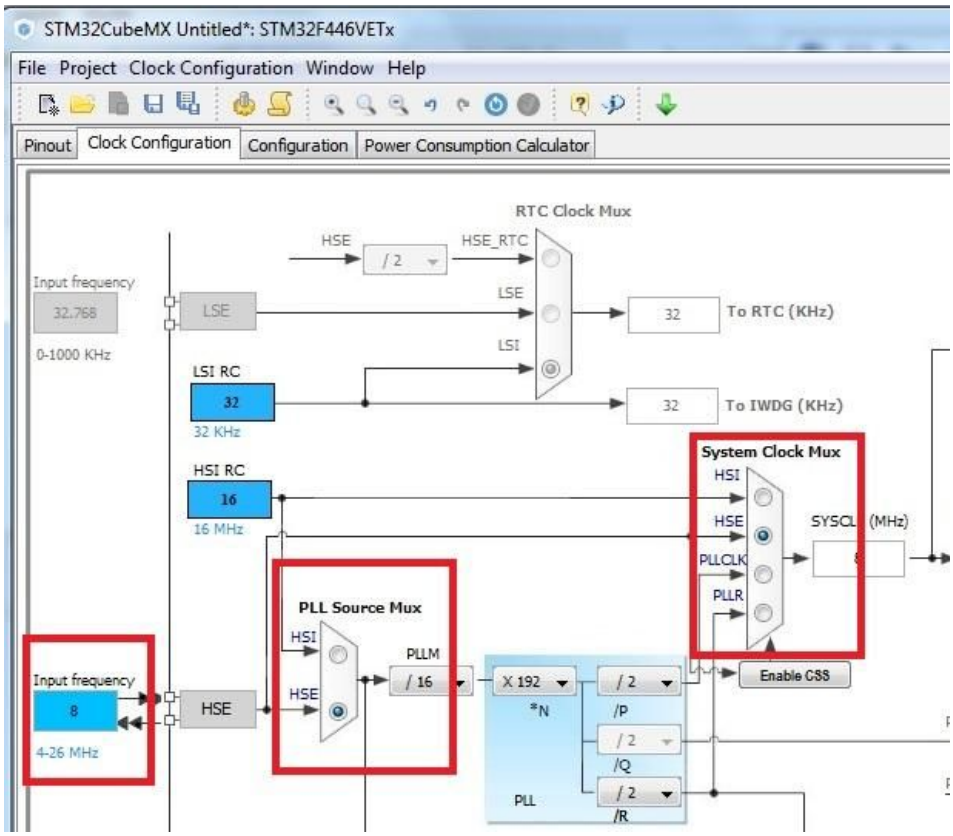


Figura 13 - Configuração da frequência de clock de acordo com o cristal presente na placa. Note que o clock a ser utilizado pelo microcrontrolador passa a ser o HSE e não mais o HSI (System Clock Mux).

11. Na aba **Configuration** click no botão USART1 definir os parâmetros de configuração para o modo assíncrono:
    a) Baud Rate
    b) Word length
    c) Stop bit
    d) Parity

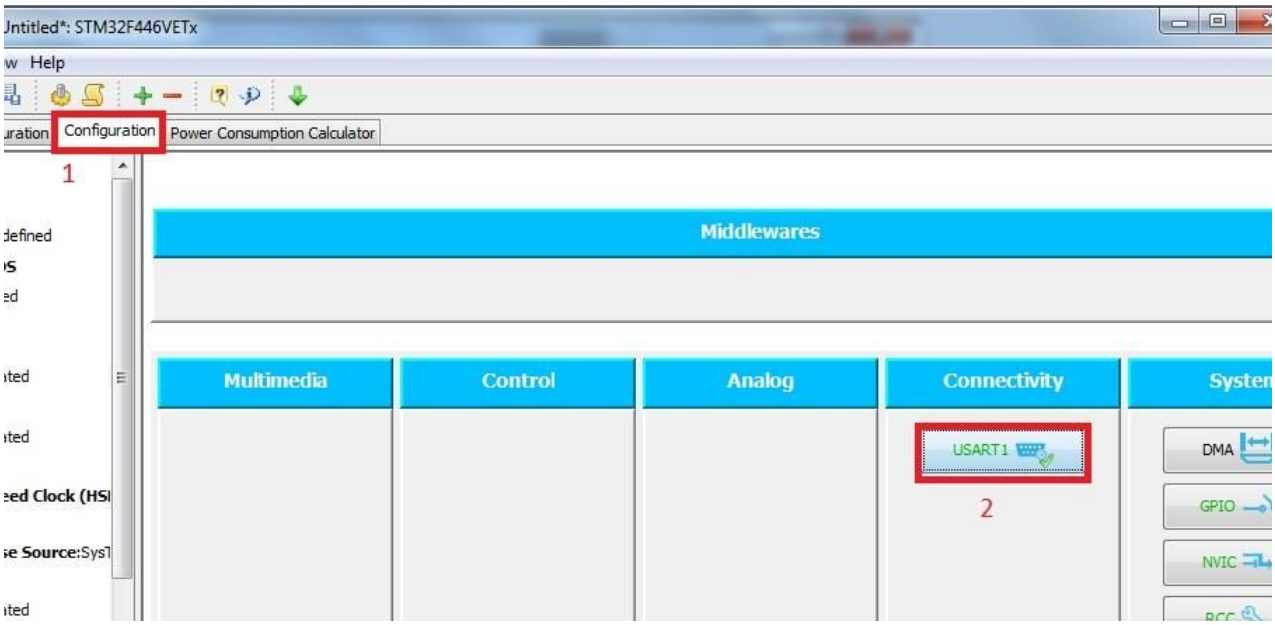


Figura 14 - Botão de configuração da USART1.

12. Seguindo os passos anteriores será exibida a janela de configuração onde o baudrate, tamanho do stop bit e a paridade poderão ser alterados conforme a necessidade de cada projeto.

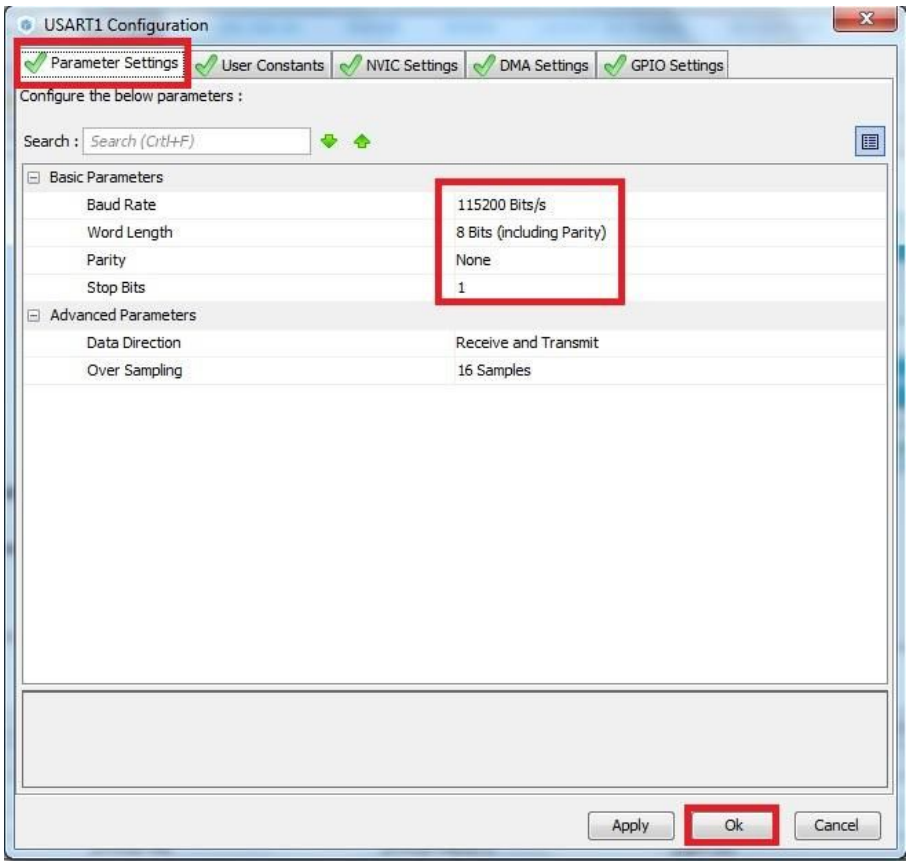


Figura 15 - Configuração dos parâmetros de comunicação da USART1.

13. Vamos definir parâmetros para a geração do projeto:

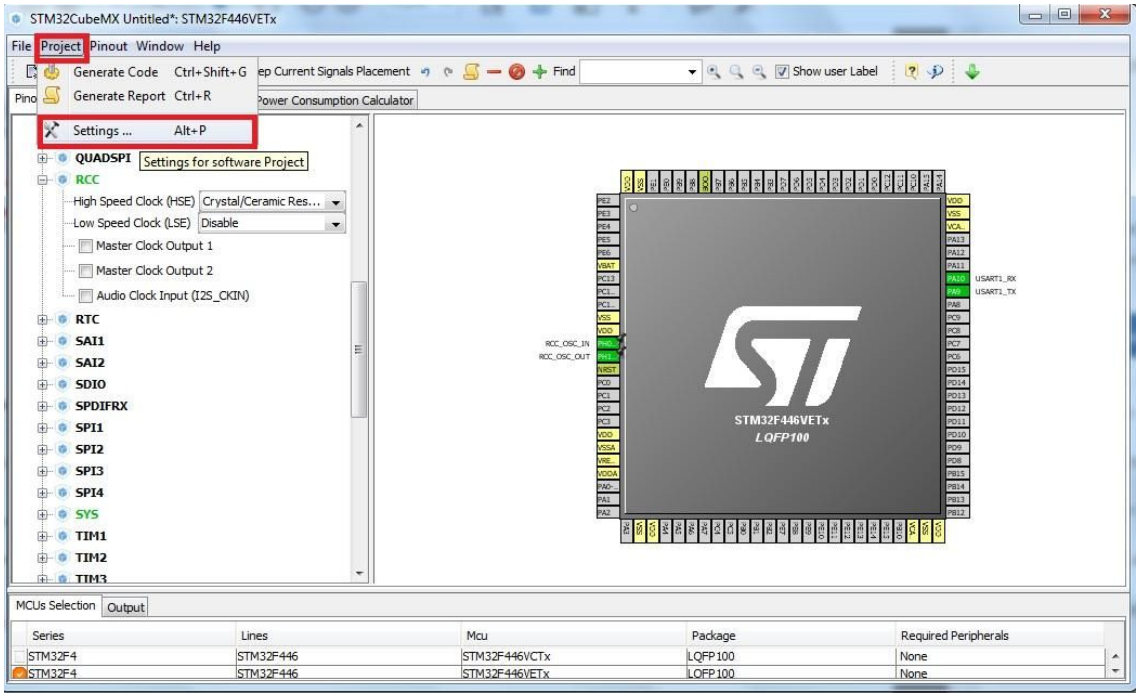


Figura 16 - Botão de configuração da USART1.

14. Defina o nome do projeto, local e compilador, conforme a figura a seguir:

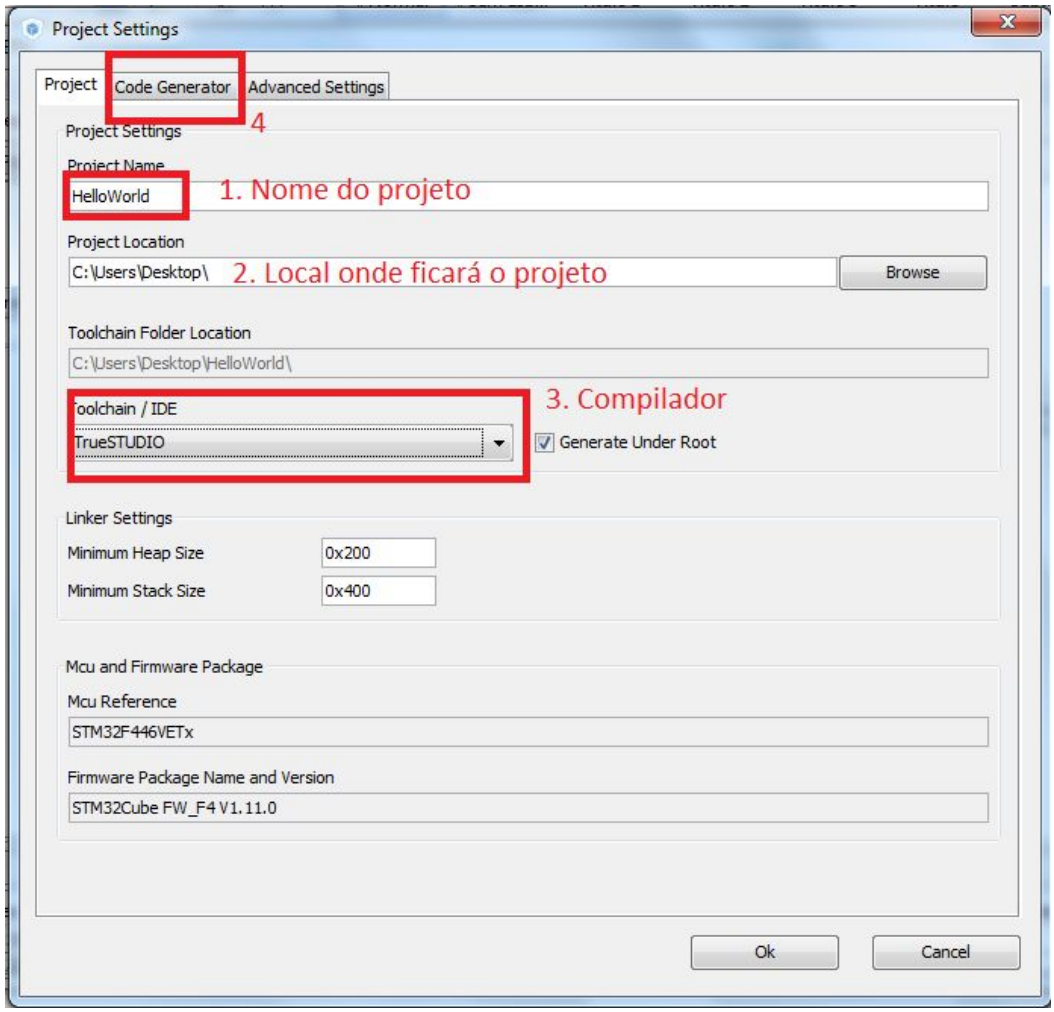


Figura 17 - Parâmetros de configuração do projeto.

15. Configurar a geração do código do projeto de acordo com os passos da figura a seguir:

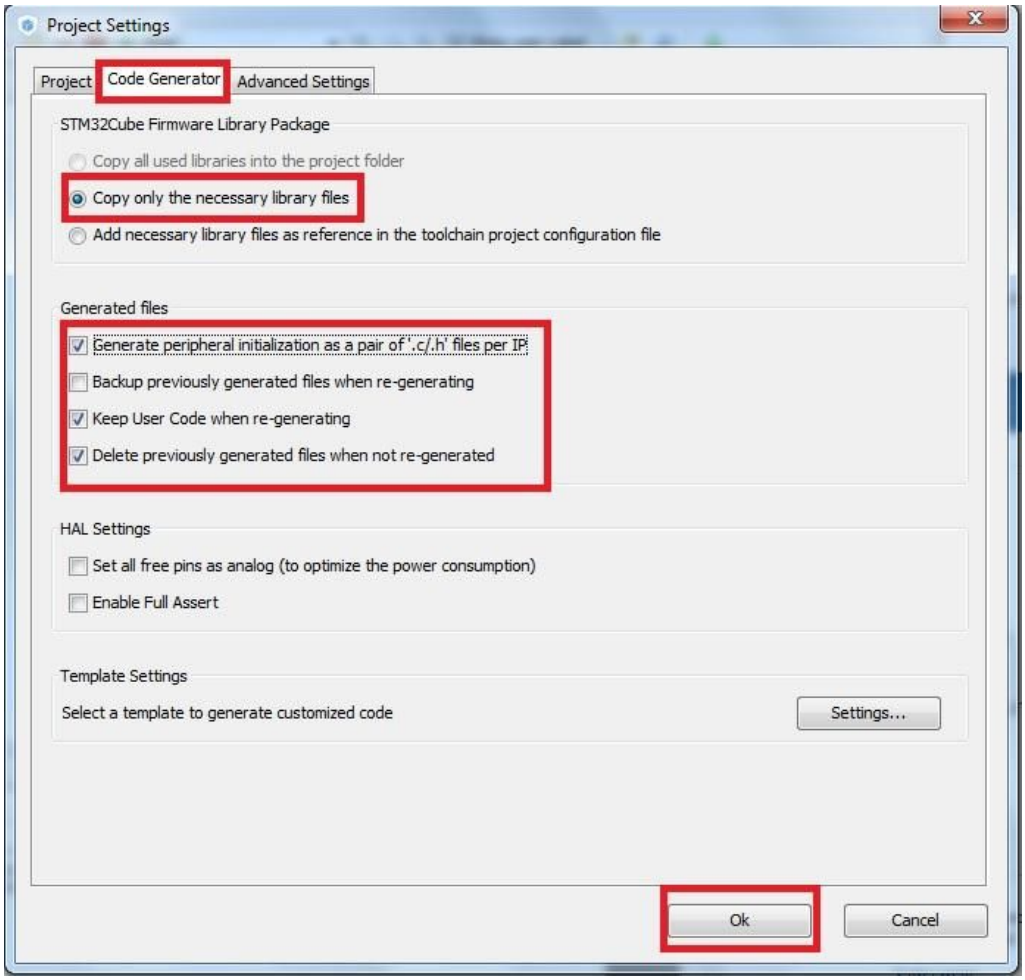


Figura 18 - Parâmetros de geração de código.

16. Após você ter configurado a geração do código para IDE Atollic TrueSTUDIO e clicado em Ok aparecerá a seguinte janela

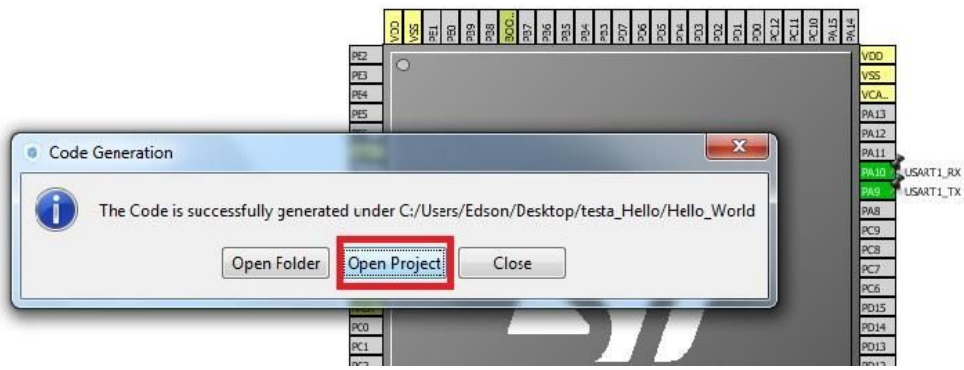


Figura 19 - Caixa de diálogo após a criação do projeto. Após clicar em "Open Project" a IDE TrueStudio deverá ser executada.

17. Então selecione Open Project para abrir a IDE

18. Após a inicialização da IDE iremos procurar o arquivo "main.c" que conterá um loop `while(1)` infinito. Este loop se faz necessário em aplicações de sistemas embarcados pois não queremos que a aplicação termine após a execução da função main().

19. Adicione o seguinte código ao loop infinito para que a porta serial envie a string "Hello World":

```
HAL_UART_Transmit(&huart1, "Hello world!/r/n", 14, 1000);
```

O código acima deverá ser inserido entre os comentários `/* USER CODE BEGIN3 */` e `/* USER CODE END 3 */` dentro do loop infinito pois o STM32CubeMX não removerá o código caso ele precise gerar novamente os drivers do projeto.

20. Compile o código utilizando o Build ou o atalho CTRL+B. Em caso de sucesso será criado um arquivo .ELF que deverá ser convertido para .HEX.

21. Exiba a aba de Terminal que já vem junto com o TrueStudio através do menu **Window > Show View > Other… > Terminal**. Configure o terminal na porta COM que surgiu quando a placa foi conectada. Utilize os mesmos parâmetros de configuração que você utilizou dentro do STM32CubeMX para a USART1. Conecte com a porta COM.

22. Converta e grave o novo arquivo .HEX através do procedimento abaixo.

Convertendo .ELF para .HEX e Gravando o Firmware

Passo 1 - Mudar os *jumpers* de boot para **BOOT0 = 1** e **BOOT1 = 0**;

Passo 2 - Conversão do arquivo .ELF em .HEX utilizando o **Ronetix ARM Toolset.** Na linha de comando executar:

```
arm-elf-objcopy -O ihex <arquivo>.elf <arquivo>.hex
```

Passo 3 - Utilização do **STM Flash Loader** na porta serial criada ao se conectar a placa. As configuração são 8 bits de dados e paridade par. Gravar o .HEX;

Passo 4 - Desligar a placa e retorna os *jumpers* para **BOOT0 = 0** e **BOOT1 = 0;**

Passo 5 - Religar a placa e testar o novo firmware.