DSP – FILTRO DIGITAL

Autor: Lucas Daudt Franck
GitHub: github.com/LDFranck/DSP-with-CMSIS

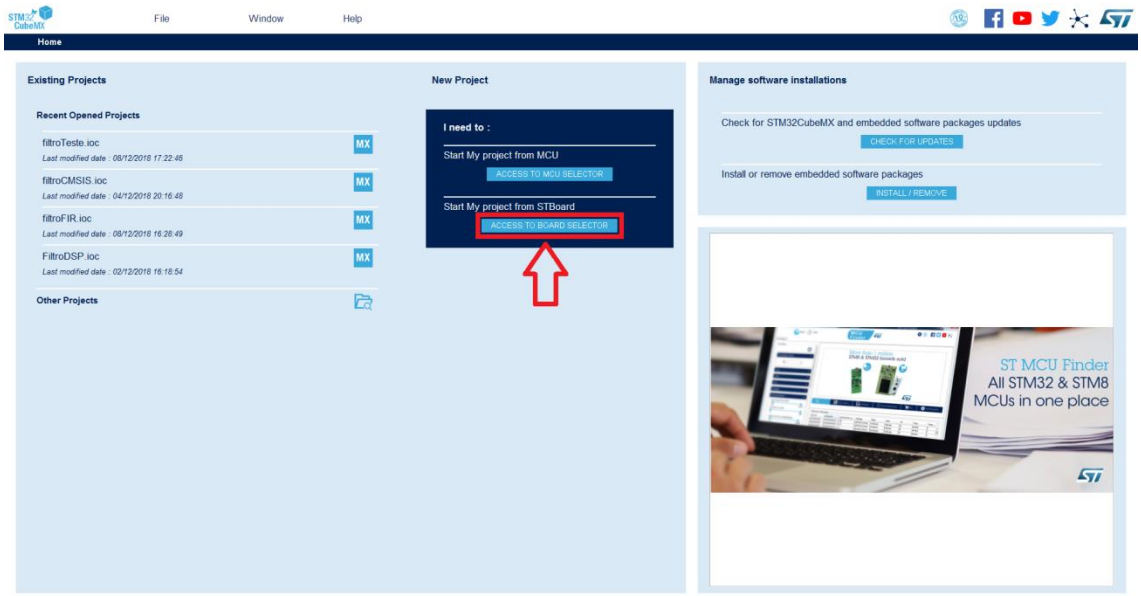
INTRODUÇÃO:

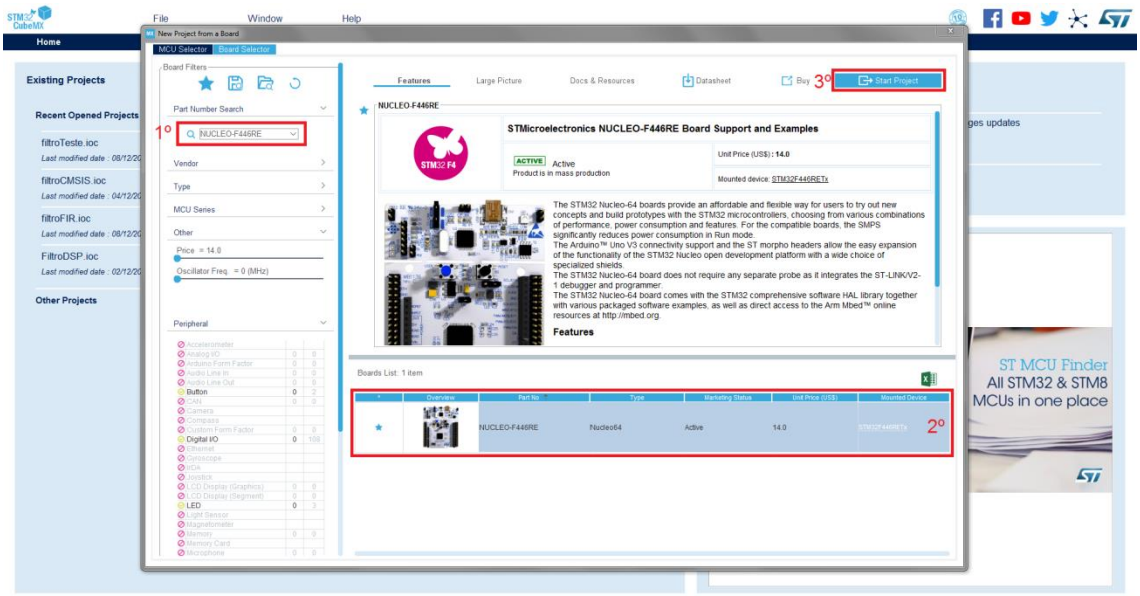
Este documento é um breve tutorial a respeito da implementação de filtros digitais na placa de desenvolvimento STM32F446RE utilizando a CMSIS. O projeto dos filtros pode ser realizado no site www.micromodeler.com/dsp/ ou em CADs matemáticos como o MATLAB e o OCTAVE. Um código exemplo está disponibilizado no GitHub do autor.

INICIANDO UM PROJETO NOVO NO STM32CUBEMX:

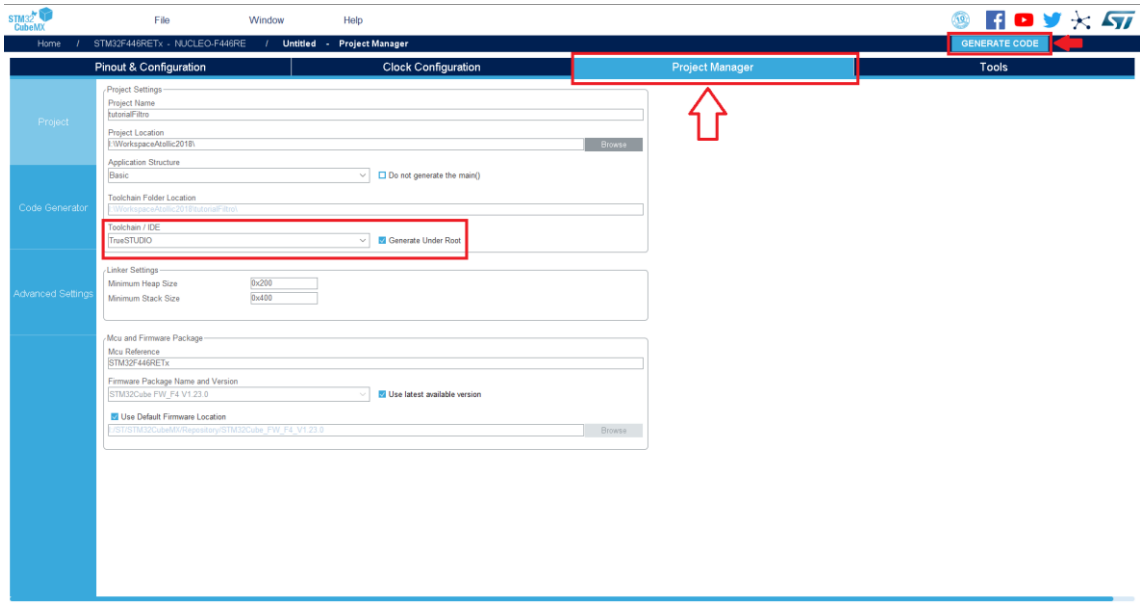
Antes de realizar o projeto dos filtros, é necessário preparar o projeto para que seja possível a utilização das funções da CMSIS. Neste tutorial será utilizado a IDE TrueSTUDIO – Atollic e a versão 5.0.0 do STM32CubeMX.

A primeira etapa consiste em gerar os arquivos base do projeto através do STM32CubeMX. Para isto, inicializar o programa e criar um novo projeto. Para facilitar a utilização do filtro, é interessante que a placa de desenvolvimento utilizada tenha um conversor digital-analógico. Neste tutorial será utilizada a placa de desenvolvimento STM32F446RE.

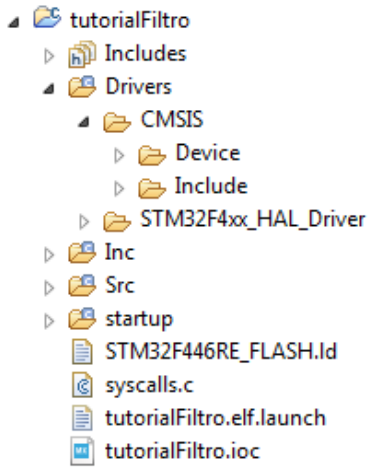




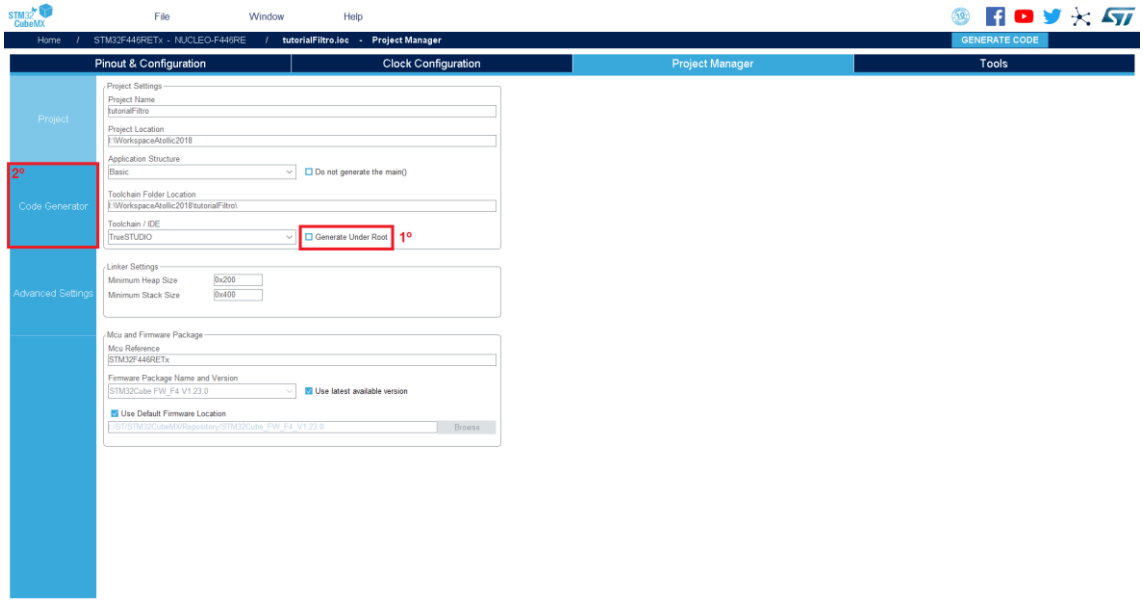
Após criar o projeto e configurar os periféricos, é necessário gerar os arquivos do projeto. Para isso, clicar na aba “Project Manager” e preencher os campos conforme a figura abaixo. É necessário escolher corretamente a Toolchain / IDE a ser utilizada. Para a utilização do TrueSTUDIO – Atollic, escolha “TrueSTUDIO” e confirme que a opção “Generate Under Root” esteja marcada. Com isso feito, gerar o código clicando na opção “Generate Code” no canto superior direito.



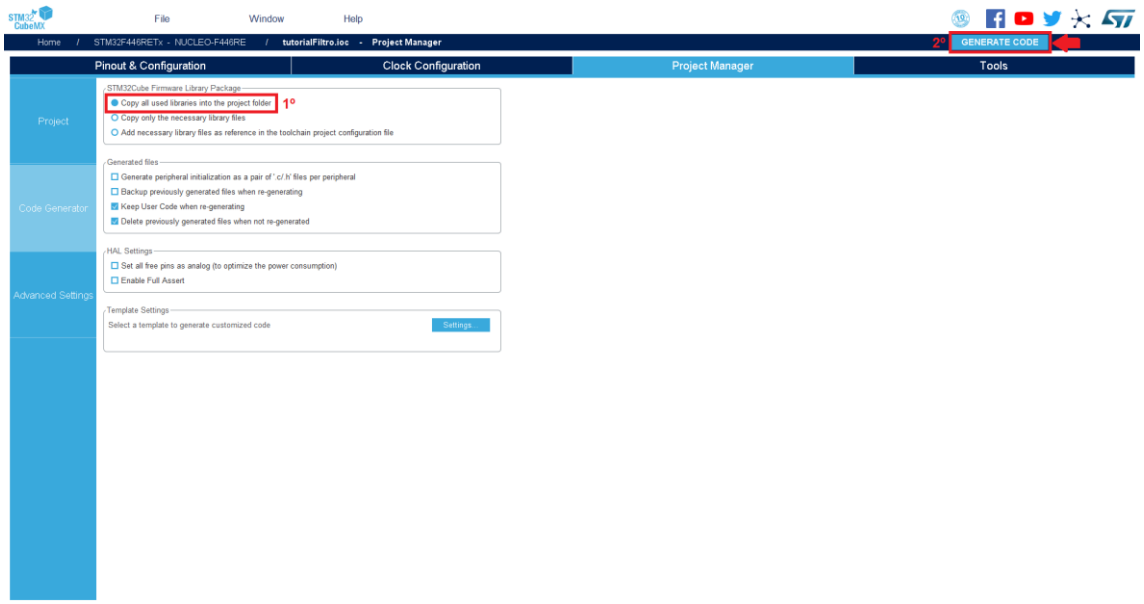
Após esta etapa, importar o projeto gerado para o TrueSTUDIO – Atollic. É interessante notar que os arquivos “.c” da CMSIS não foram gerados. Você deve ter um projeto semelhante à figura abaixo.



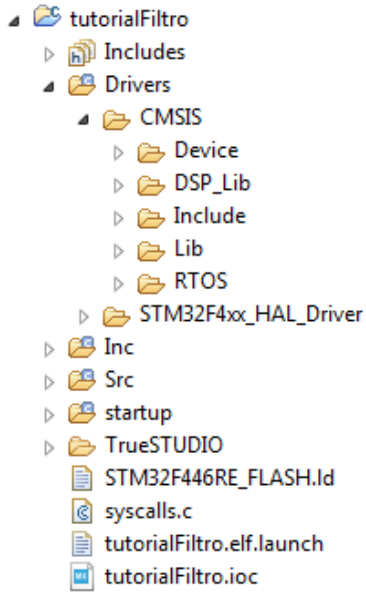
Para gerar os arquivos “.c” da CMSIS é necessário retornar ao STM32CubeMX do projeto e desmarcar a opção “Generate Under Root”. Após isto feito, clicar na aba “Code Generator” localizada na esquerda.



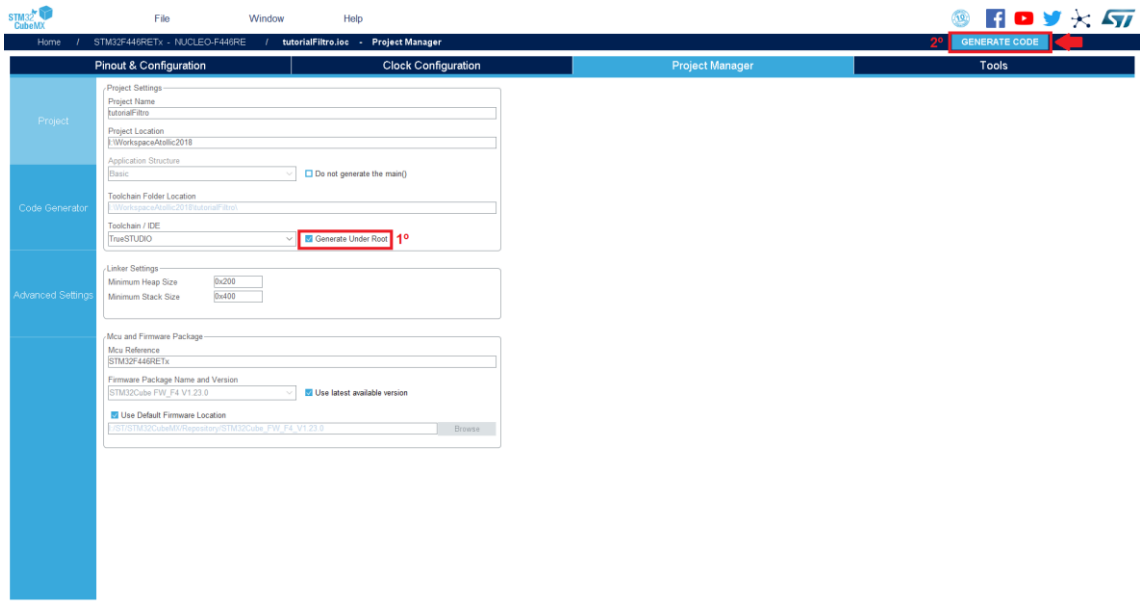
Na aba “Code Generator”, marcar a opção “Copy all used libraries into the project folder” e gerar novamente o projeto clicando no botão “Generate Code” no canto superior direito.



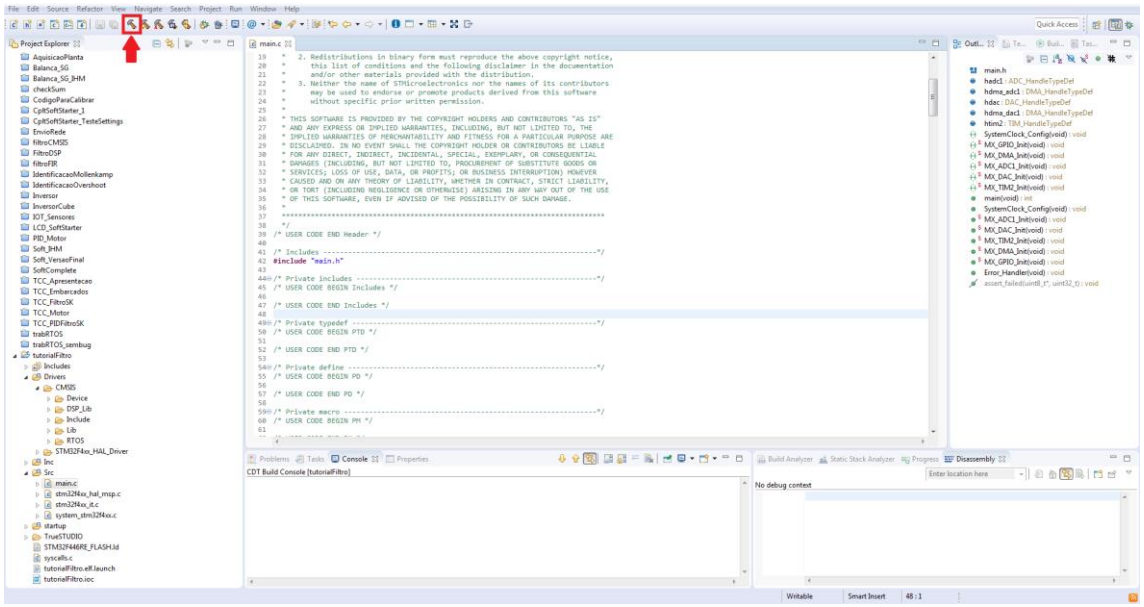
Agora o seu projeto deve estar semelhante à figura abaixo.



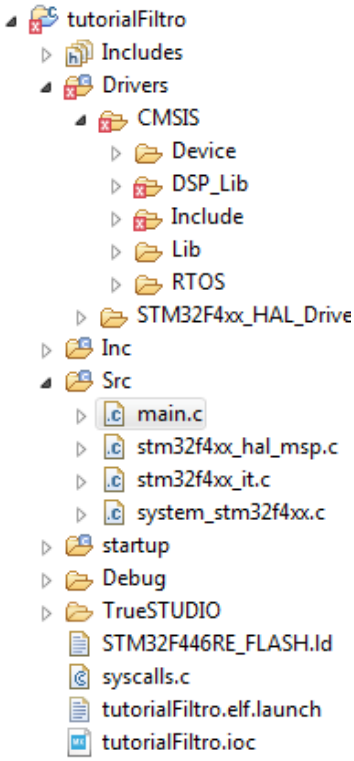
Agora será necessário retornar ao STM32CubeMX do projeto e gerar novamente os arquivos do projeto com a opção “Generate Under Root” marcada. É interessante notar que os arquivos da CMSIS não serão apagados do seu projeto.



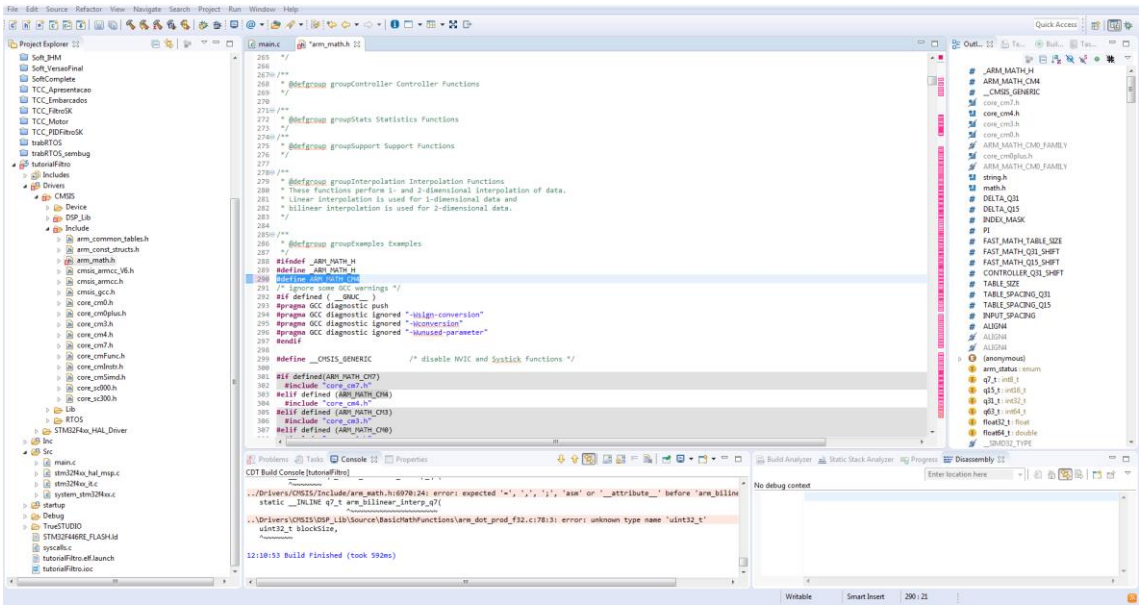
A próxima etapa consiste em dar um “build” no projeto. Para isso, entrar no arquivo “main.c” e clicar sobre o ícone do martelo localizado na barra de ferramentas superior.



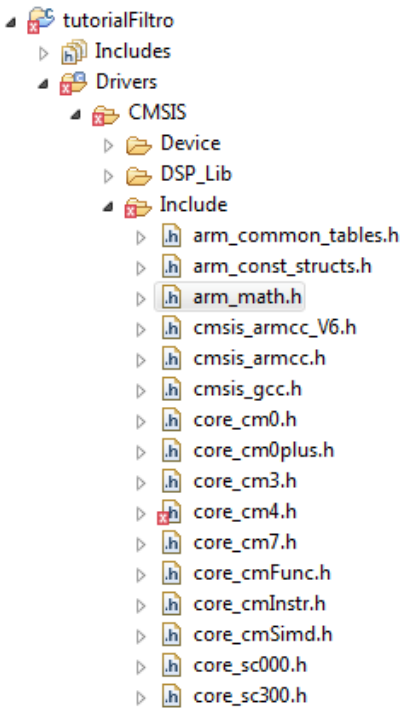
Com isto feito, o seu projeto deve apresentar erros em algumas pastas como na figura abaixo.



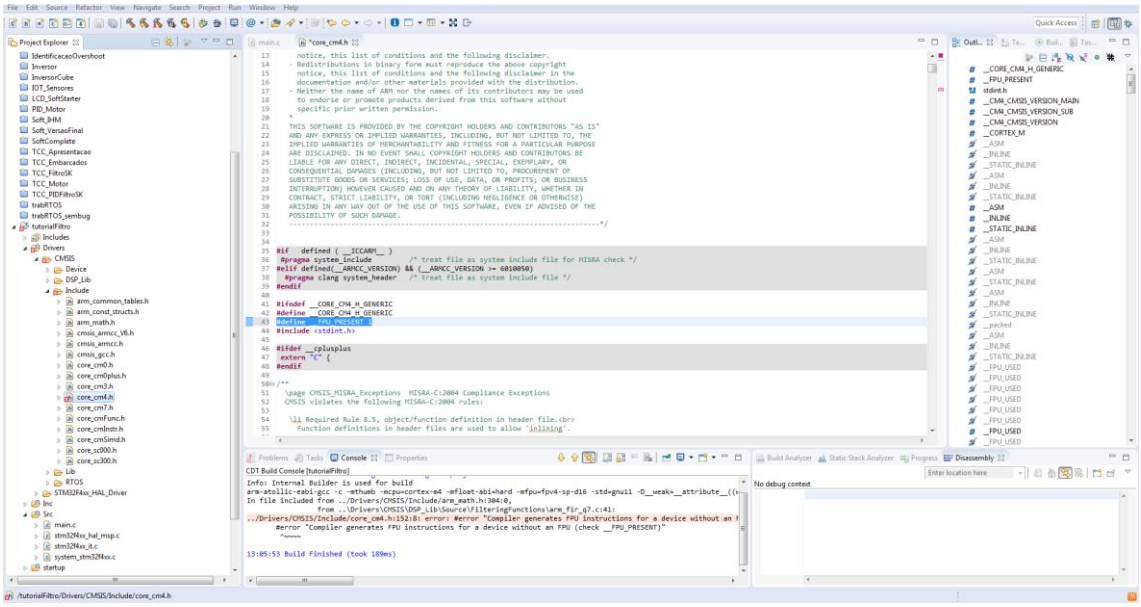
O primeiro erro a ser corrigido encontra-se no arquivo “arm_math.h” na pasta “Include”. Para corrigir este erro, é necessário incluir na linha 290 do arquivo “arm_math.h” a seguinte instrução: “#define ARM_MATH_CM4”.



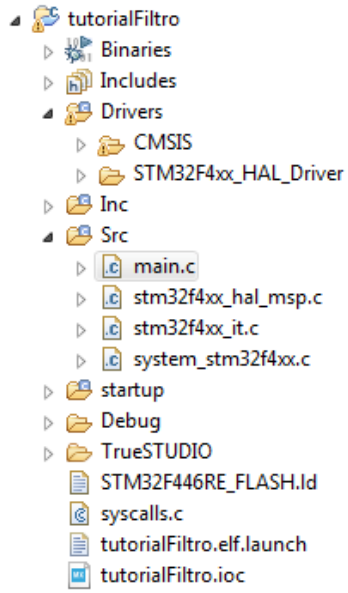
Com isto feito, salvar o arquivo “arm_math.h” e dar um build no projeto. O erro no arquivo “arm_math.h” deve desaparecer e um novo erro no arquivo “core_cm4.h” na pasta “Include” deve aparecer.



Para solucionar este erro, é necessário incluir na linha 43 do arquivo “core_cm4.h” a seguinte instrução: “#define __FPU_PRESENT 1”.



Após completar a alteração, salvar o arquivo “core_cm4.h” e dar um build no projeto. Os erros do projeto devem desaparecer. O projeto ainda irá apresentar “warnings”, mas estes permitem fazer o debug do código.

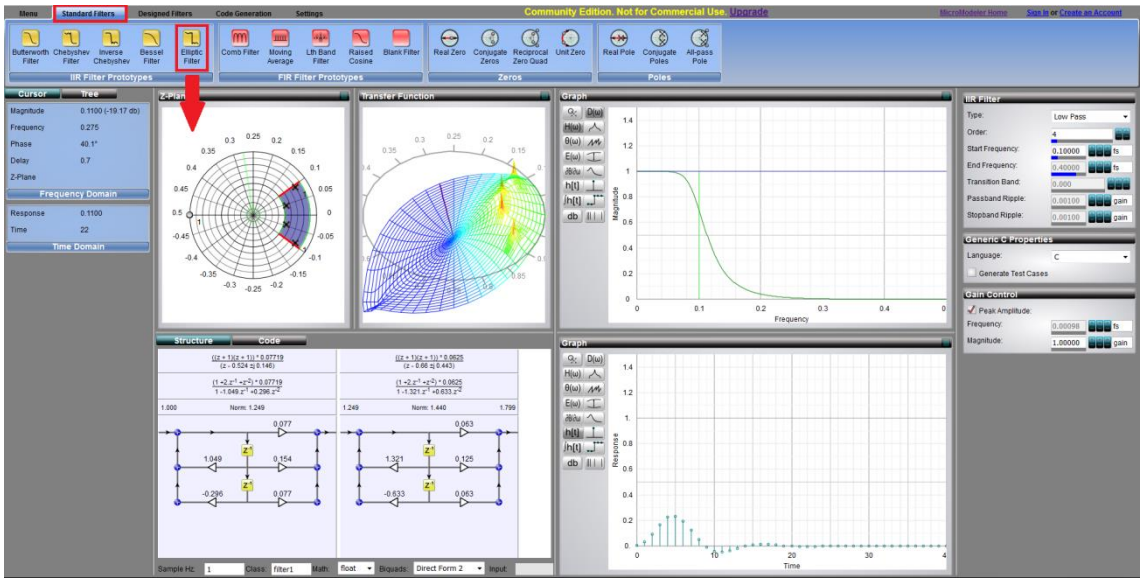


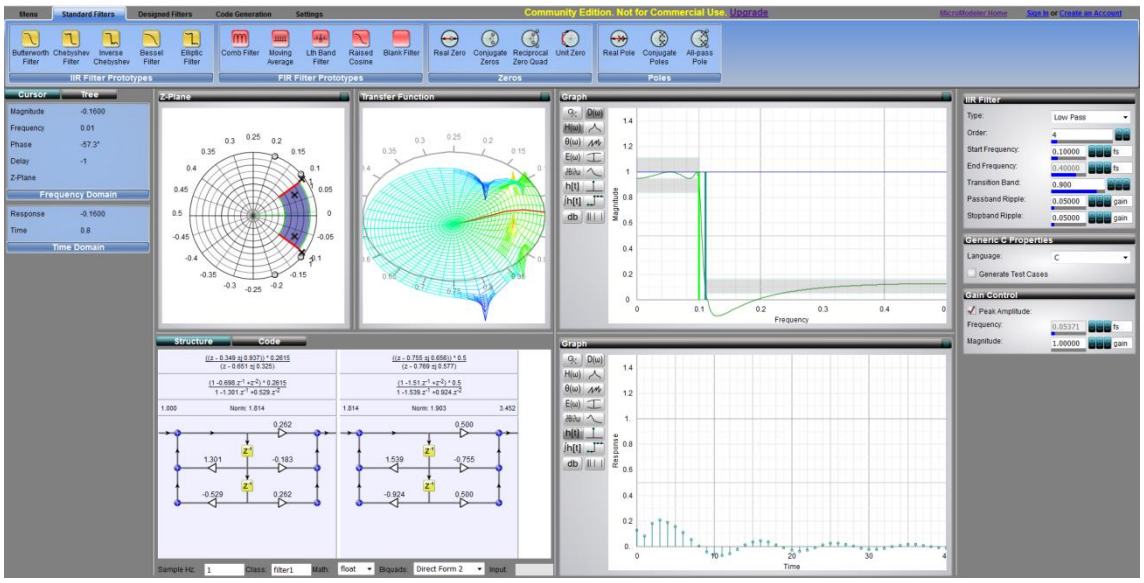
Agora o projeto se encontra pronto para a utilização das funções da CMSIS. O código exemplo já está pronto para o uso, não havendo necessidade de realizar todos estes procedimentos.

PROJETO A PARTIR DO SITE MICROMODELER:

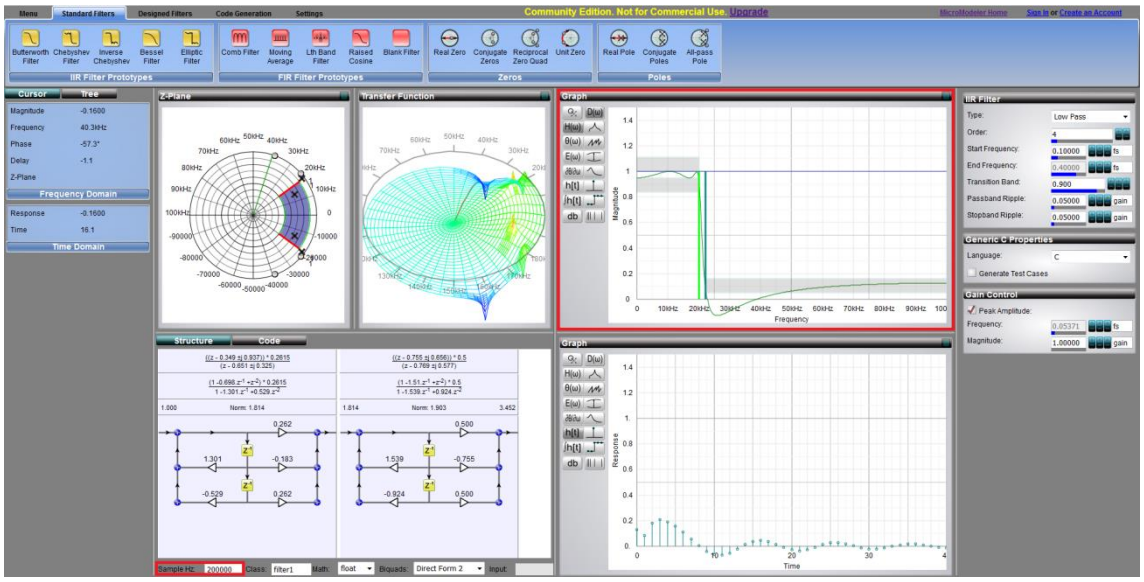
O filtro digital pode ser facilmente projetado com o auxilio do site www.micromodeler.com/dsp/. A primeira vista o site pode parecer complexo, mas sua utilização é relativamente simples. Esta ferramenta é capaz de projetar filtros IIR (*Infinite Impulse Response*) e filtros FIR (*Finite Impulse Response*). Em sua versão gratuita, o site possibilita o usuário criar filtros IIR de até 4ª ordem e filtros FIR de até 21ª ordem. Como exemplo, será projetado um filtro passa-baixas Elíptico de 4º ordem com uma frequência de corte de 1kHz. A frequência de amostragem adotada será de 200kHz (frequência de amostragem do código exemplo – configurada no STM32CubeMX).

O primeiro passo é definir o tipo do filtro. Para isto, é necessário ir à aba “Standard Filters” e selecionar a opção “Elliptic Filter” e arrastá-la para a área de trabalho do site.

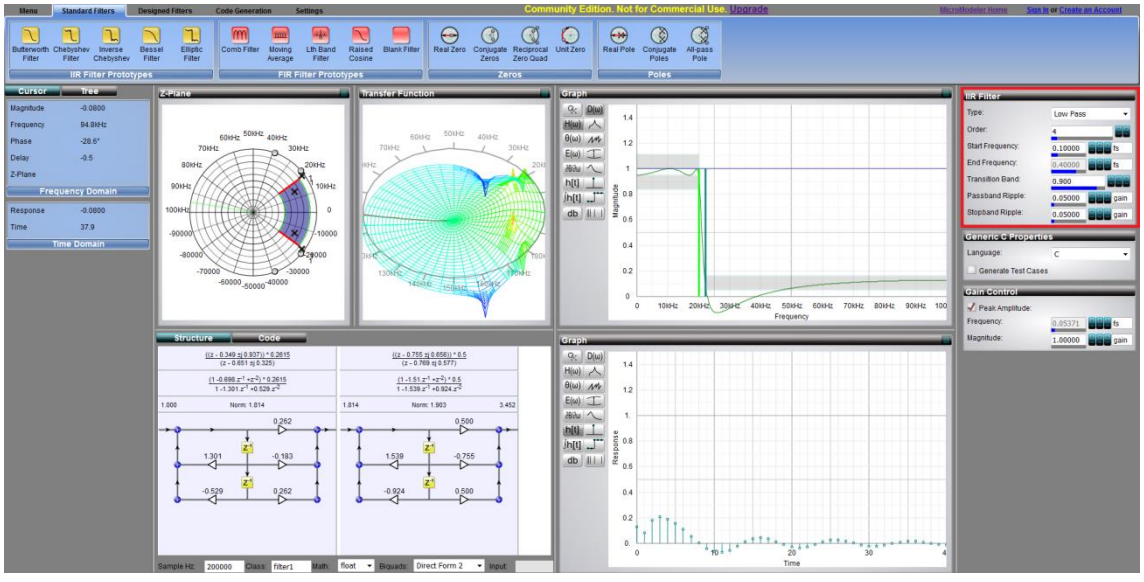




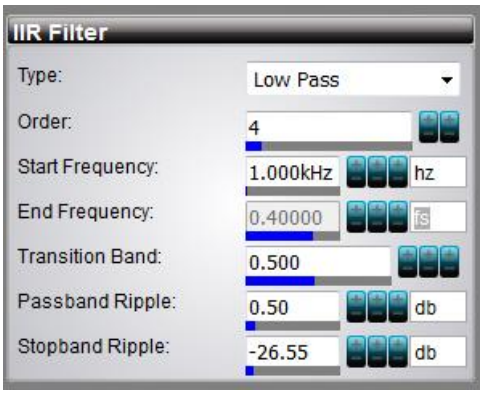
A próxima etapa consiste em alterar a frequência de amostragem para 200kHz. Para isto, alterar o valor “1” contido na opção “Sample HZ” para “200000”. Com esta alteração, as frequências do gráfico se alteram, facilitando o projeto do filtro.



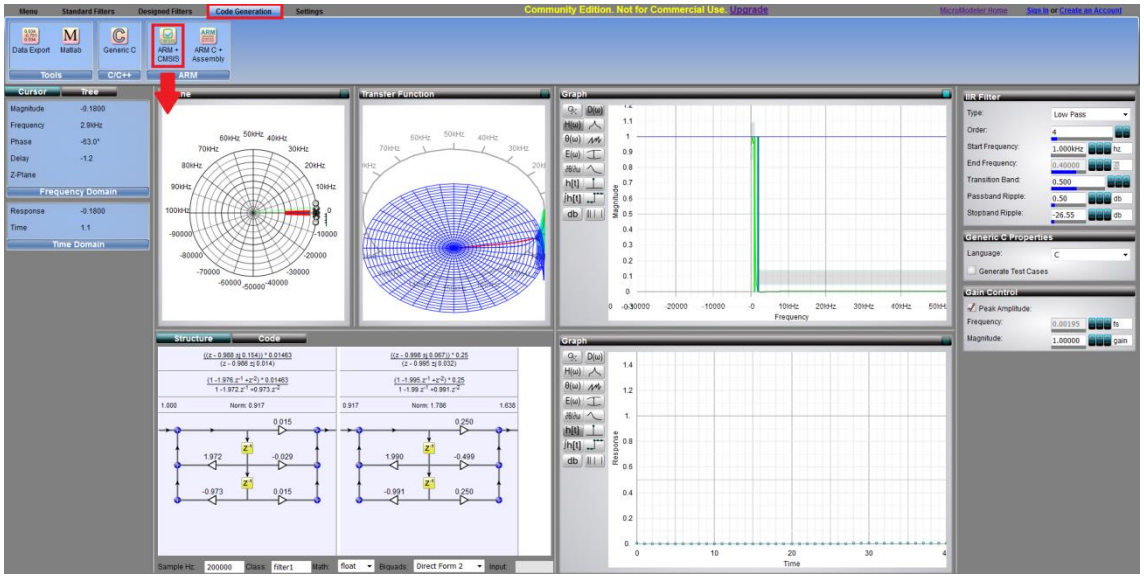
Uma vez com as frequências do filtro na escala correta, chega o momento de configurar o tipo do filtro, a frequência de corte e o ripple na banda passante. Todos esses parâmetros podem ser alterados na caixa denominada “IIR Filter”, localizada a direita da tela. Em muitos casos é conveniente mudar o tipo da unidade mostrada (ex.: ao invés de mostrar em “gain”, mostrar em “dB”). Para isto, basta clicar com o mouse sobre a caixa que indica a unidade.



Para contemplar as especificações do filtro necessárias, as seguintes alterações foram feitas nos parâmetros do filtro.

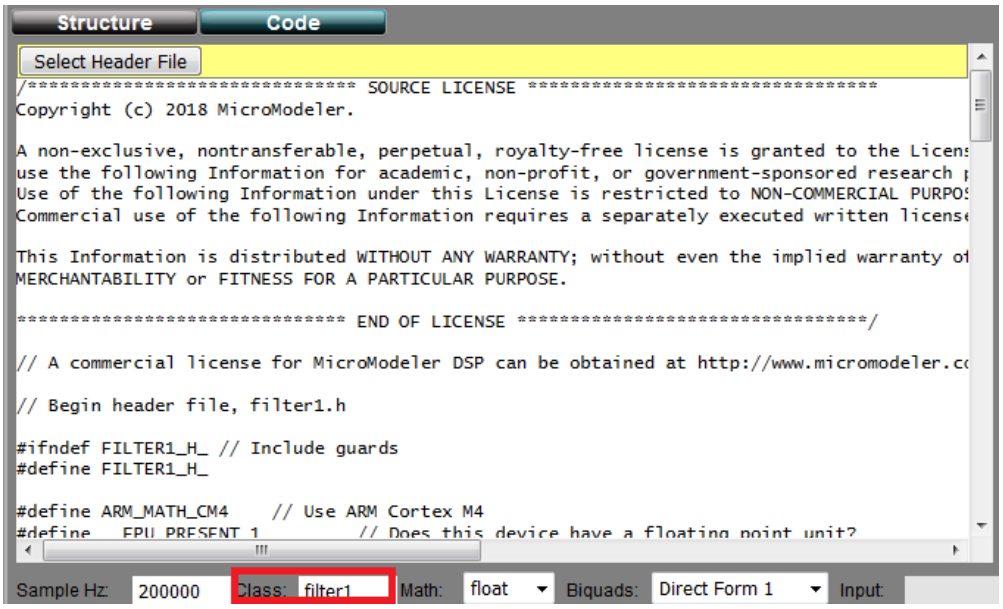


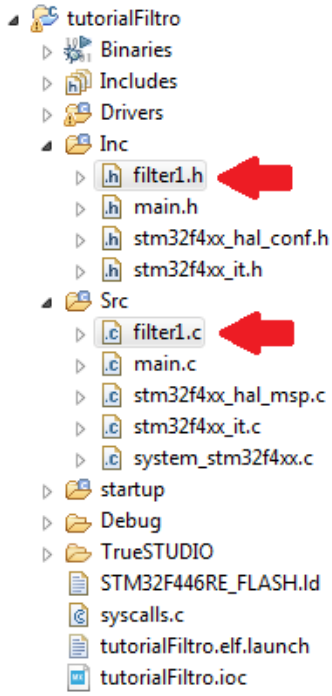
Uma vez que o filtro foi projetado corretamente, chega o momento de gerar o código para o ARM. Para isto, ir à aba “Code Generation” e selecionar a opção “ARM + CMSIS” e arrastá-la para a área de trabalho do site.



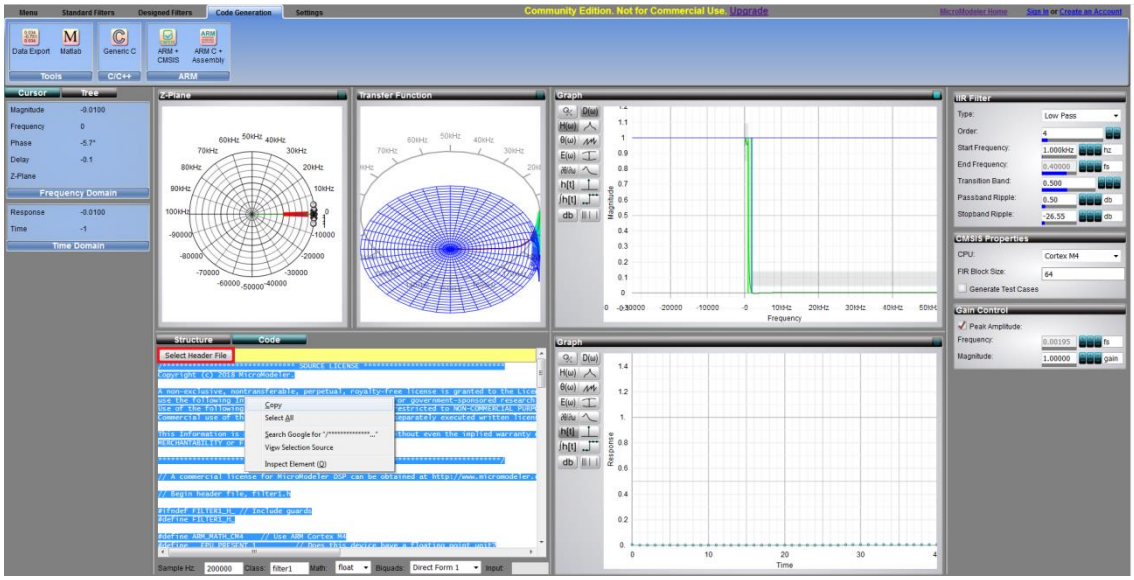
Com isto feito, o site se encarrega de gerar o código para ser utilizado no microcontrolador. Para visualizar o código gerado, basta ir à aba “Code” e mudar a opção “Biquads” de “Direct Form 2” para “Direct Form 1”. O site gera um arquivo “.h” e um arquivo “.c” com as funções para a utilização do filtro.

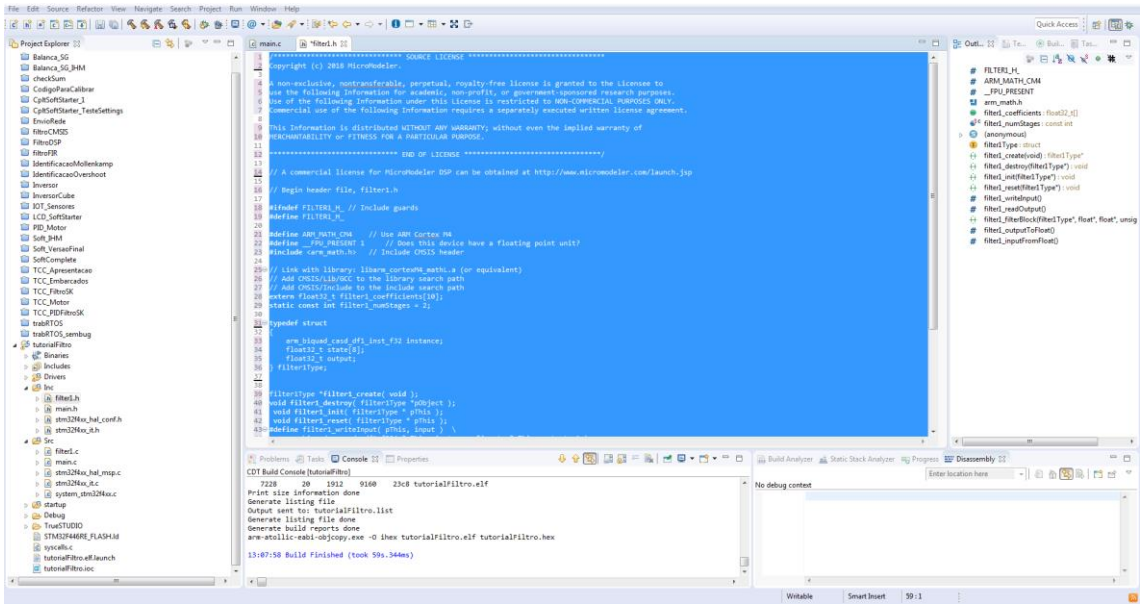
O próximo passo consiste em importar o código gerado pelo site para o projeto no TrueSTUDIO – Atollic. Para isto, é necessário criar um arquivo “.h” e um “.c” com o nome da classe indicado pelo site (pode ser alterado), no caso deste exemplo é “filter1”.



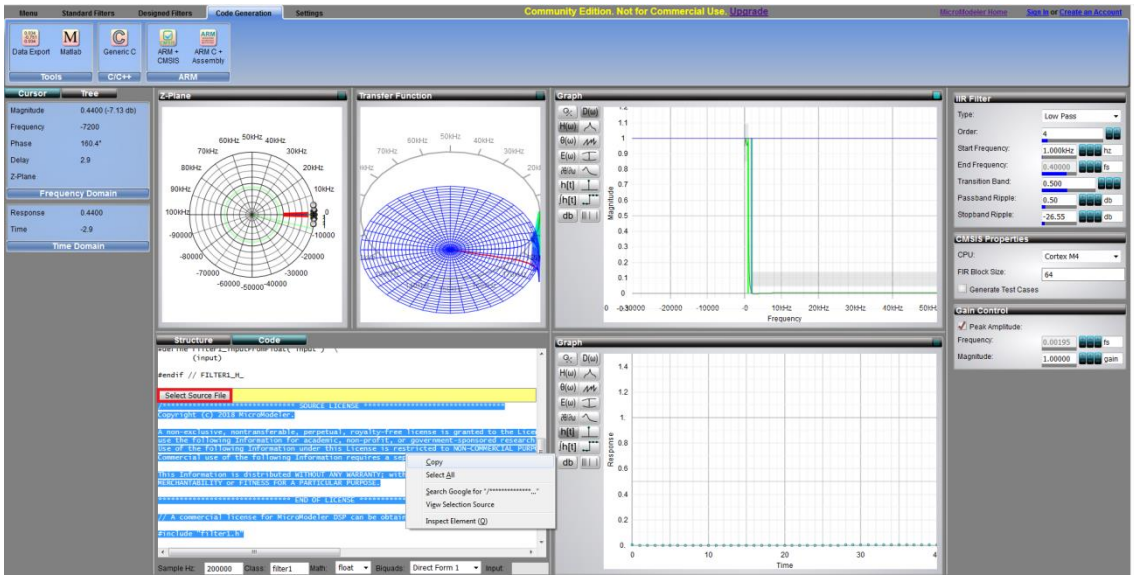


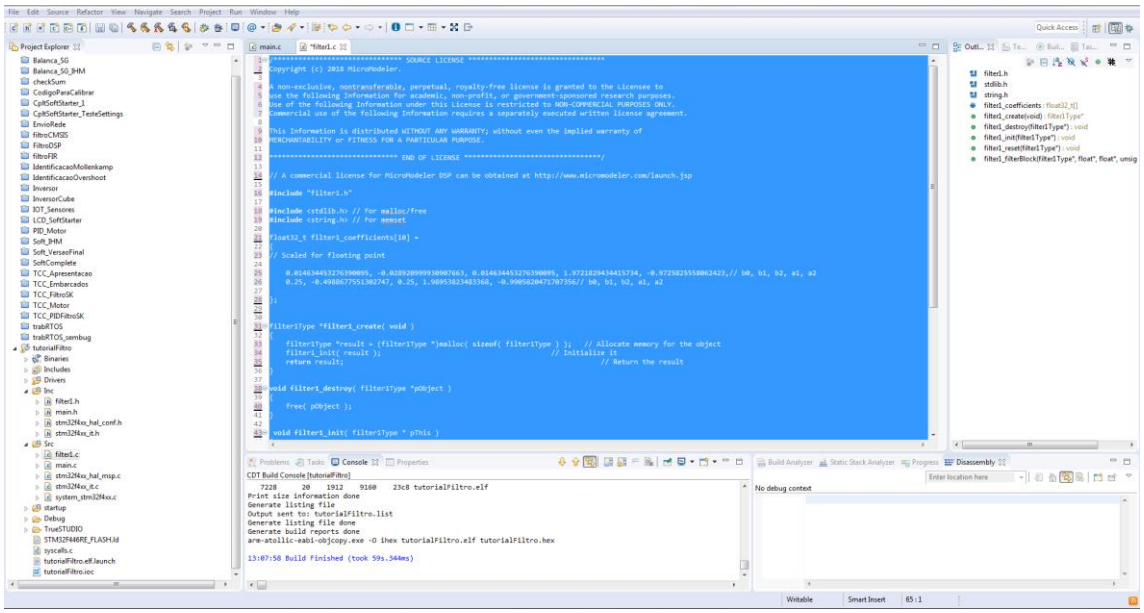
Agora basta copiar o código do “.h” gerado pelo site e colar dentro do arquivo “filter1.h” na pasta do projeto. Para selecionar o código pertinente ao “.h”, clicar sobre “Select Header File” e então copiar o código utilizando o botão direito do mouse (não funciona utilizando as teclas de atalho).



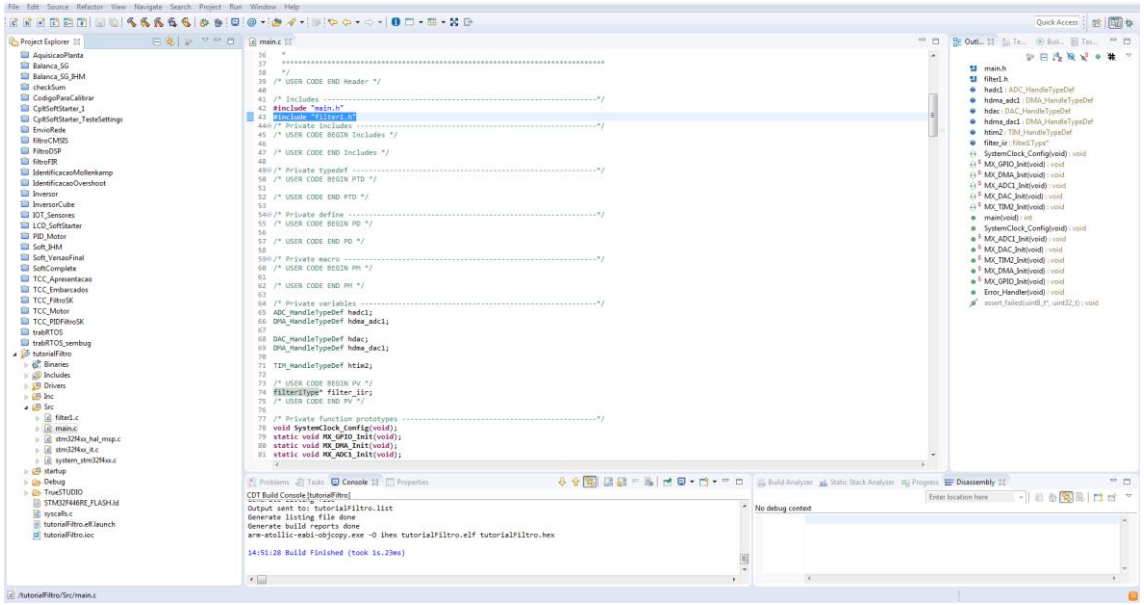


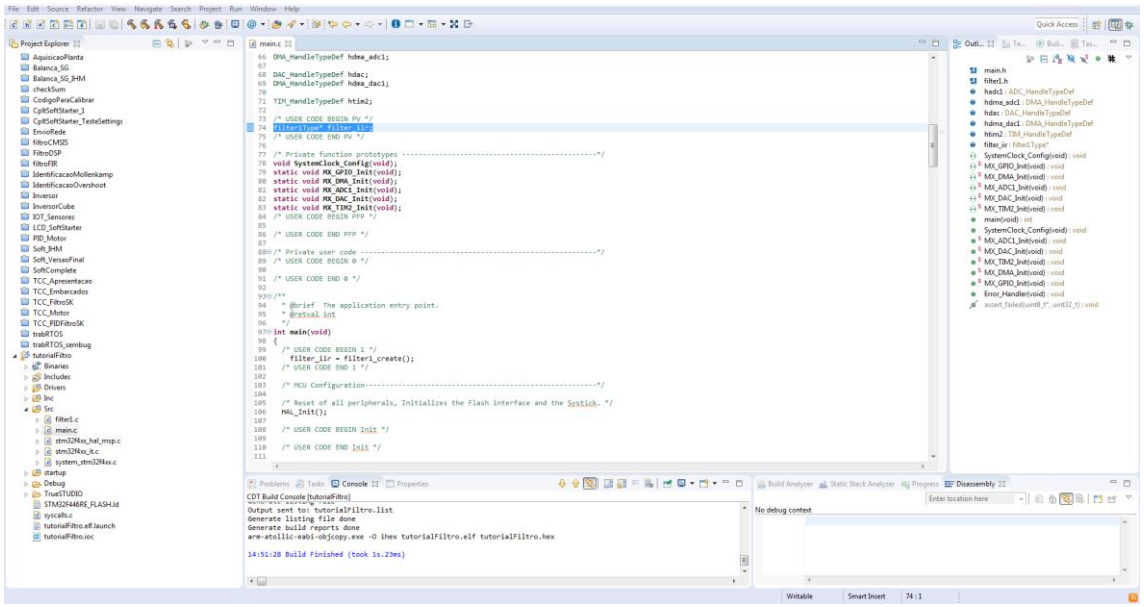
O mesmo procedimento deve ser adotado para transferir o código pertinente ao “.c” do site para a pasta do projeto. Desta vez clicar na opção “Select Source File”.

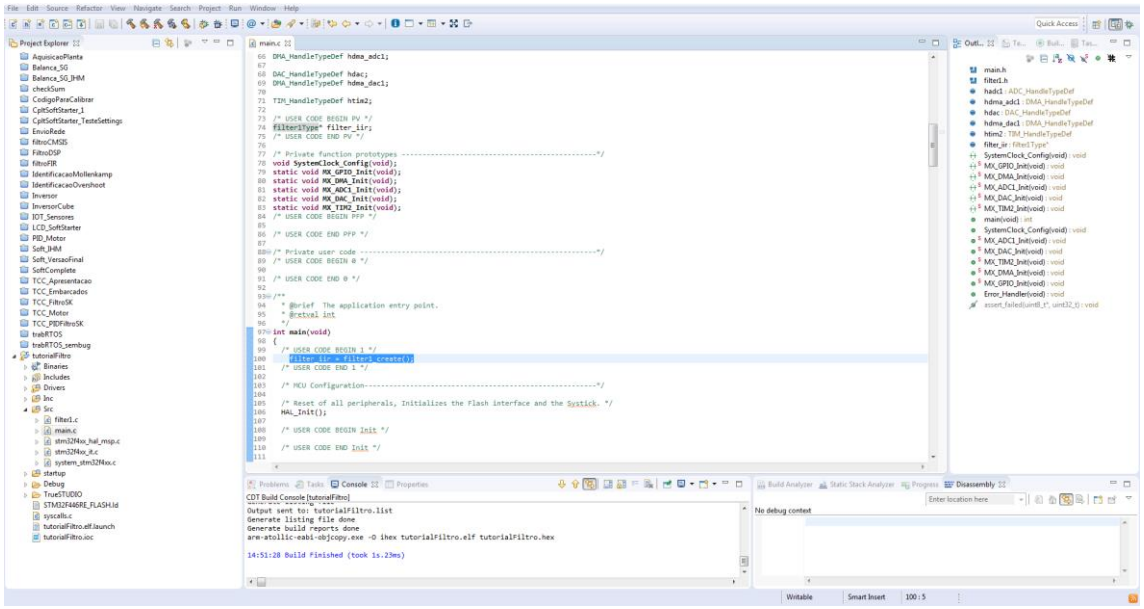




Após esta última etapa, o filtro está quase pronto para o uso. Basta incluir o arquivo “filter1.h” no arquivo “main.c” e criar uma instância do mesmo como no exemplo abaixo.







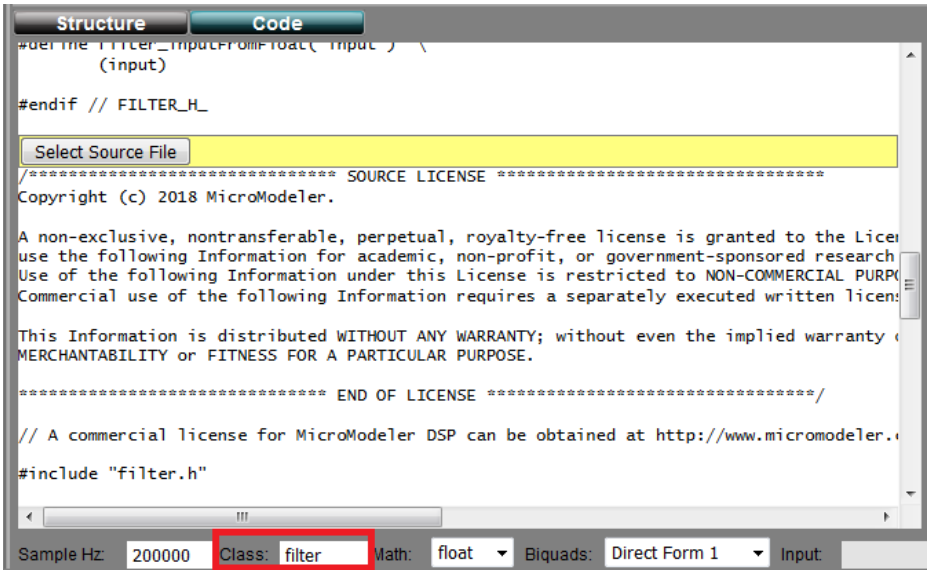
A seguinte função deve ser chamada para realizar a filtragem dos dados.

```
int filter1_filterBlock( filter1Type * pThis, float * pInput, float * pOutput, unsigned int count )
{
    arm_biquad_cascade_df1_f32( &pThis->instance, pInput, pOutput, count );
    return count;

}
```

É recomendado a filtragem de blocos de medidas para otimizar o processamento do microcontrolador. O código exemplo, disponibilizado no GitHub do autor, pode ser facilmente adaptado a outros tipos de filtros apenas

substituindo os arquivos “filter.h” e “filter.c” com o código gerado pelo site. É importante lembrar a necessidade de mudar o nome da classe do site para “filter” ao invés de “filter1” para que o código exemplo funcione corretamente após substituir os arquivos “.c” e “.h”.



PROJETO A PARTIR DO MATLAB / OCTAVE

< Em construção >