No quarto artigo da série “A Blue Pill”, daremos continuidade no estudo. A dica é acompanhar a série desde o primeiro artigo porque é um caminho estruturado, onde a sequência traz benefícios no aprendizado (linha de raciocínio). Nesse artigo veremos como instalar os softwares, a biblioteca, e os utilitários que utilizaremos nesse aprendizado.

Instalação de software

Antes de iniciar os projetos, precisamos de alguns softwares instalados. Há várias etapas envolvidas. Apesar disso, o processo deve prosseguir sem problemas.

Convenções de diretório usadas

No decorrer dessa série, diferentes subdiretórios do software serão referidos. Supõe-se que o nível superior do software instalado seja nomeado "~/stm32f103c8t6". Portanto, quando me refiro ao nome do caminho "~/stm32f103c8t6/libopencm3/README.md", assumo que começa no seu diretório home (~).

Software Operacional

Também assumiremos que você tem um ambiente POSIX (Linux/Unix) para executar comandos. Os ambientes Linux ou Raspberry Pi, usando o shell bash são talvez os mais naturais. Eu utilizo e recomendo o Linux Ubuntu LTS 18.04.

Se você usa o Windows, convém instalar o Cygwin, site https://www.cygwin.com. Alguns podem usar o MSYS. Após instalar o sistema Cygwin básico, certifique-se de instalar também o make e o git. Isso fornecerá um ambiente de linha de comando semelhante ao Linux para o desenvolvimento de software.

O pacote de Software

A estrutura de diretórios do pacote de software está disponível em github.com. Escolha um local adequado para criar um subdiretório. Aqui assumiremos o diretório home:

```
$ cd ~
```

Estando dentro do diretório home, use o seguinte comando git para baixar e criar um subdiretório:

```
~$ git clone https://github.com/ve3wwg/stm32f103c8t6.git
```

O comando anterior criará o diretório ~/stm32f103c8t6. No entanto, fique à vontade para renomeá-lo para algo mais fácil de digitar, como ~/stm32, mas, usarei ~/stm32f103c8t6.

A biblioteca ‘libopencm3’

Agora devemos fazer o download da biblioteca libopencm3 no local correto. Primeiro, mude para o subdiretório ~/stm32f103f8t6, e em seguida, use o seguinte comando git clone, conforme mostrado a seguir:

```
~$ cd stm32f103f8t6
```

```
~/stm32f103f8t6$ git clone https://github.com/libopencm3/libopencm3.git
```

Isso baixará do github o diretório ~/stm32f103c8t6/libopencm3 com arquivos e subdiretórios.

FreeRTOS

O próximo software importante é o FreeRTOS. Ele deve ser baixado e descompactado, como um arquivo zip.

- ✔ Acesse http://www.freertos.org;
- ✔ Localize "Download Source" à esquerda;
- ✔ Clique no link “Clique para baixar o último lançamento oficial do SourceForge.”.

Dependendo do seu navegador e sistema operacional, um arquivo zip deve ser baixado automaticamente. Terá um número de versão no nome do arquivo. Nesse momento, o nome do arquivo baixado é FreeRTOSv10.0.1.zip.

Mude para o subdiretório ~/stm32f103c8t6/rtos, antes de descompactar o arquivo zip. Como estou usando o Linux, o diretório de download é ~/Downloads, então, faça conforme mostrado a seguir:

```
~$ cd stm32f103c8t6/rtos
~/stm32f103c8t6/rtos$ unzip ~/Downloads/FreeRTOSv10.0.1.zip
```

Depois de concluído, deve haver vários arquivos e subdiretórios em ~/stm32f103c8t62/rtos/FreeRTOSv10.0.1.

O número da versão do FreeRTOS que você baixar está incluído no nome do subdiretório, conforme visto no tópico anterior, portanto, pode ser que seja necessário uma alteração no arquivo Project.mk, que está no diretório ~/stm32f103c8t62/rtos/. Edite o arquivo Project.mk com o seu editor favorito, e localize a seguinte linha na parte superior do arquivo:

```
FREERTOS? = FreeRTOSv10.0.1
```

Se sua versão do FreeRTOS for mais recente que essa, como por exemplo, o FreeRTOSv11.0.0, edite-a para corresponder à sua versão e salve o arquivo.

```
FREERTOS? = FreeRTOSv11.0.0
```

Isso permitirá que o arquivo de criação do Project.mk funcione corretamente mais tarde, quando você desejar criar um projeto RTOS.

Compilador ARM Cross

Se você ainda não possui um Compilador ARM Cross, ele precisará ser instalado. Se você estiver executando Linux, poderá usar apenas o comando apt-get para instalá-lo. Apesar disso, recomendo que você baixe e instale o pacote de ferramentas, conforme descrito a seguir, porque algumas ferramentas do Compilador ARM Cross não são bem organizadas e, às vezes, incompletas.

Se você estiver executando Windows (Cygwin), definitivamente use o seguinte procedimento. Este procedimento também é recomendado para Linux se você tiver problemas com os pacotes instalados:

- ✔ Acesse o site https://developer.arm.com;
- ✔ Clique no link "Linux/Código aberto";
- ✔ Role para baixo e clique em "ARM GNU Embedded Toolchain";
- ✔ Role para baixo e clique no botão grande chamado "Downloads";
- ✔ Role para baixo até encontrar o download requerido da plataforma. Windows de 32 bits, Linux de 64 bits, Mac OS X de 64 bits, etc. Clique na escolha apropriada para o download da sua plataforma;
- ✔ Crie um diretório do sistema /opt, caso você ainda não tiver um:

  ```
  ~$ sudo -i
  # mkdir /opt
  ```

- ✔ Altere para o diretório /opt:

  ```
  # cd /opt
  ```

- ✔ A partir deste ponto, você descompactará o download do compilador. Certifique-se de ser específico sobre o seu diretório pessoal:

  ```
  # tar xjf ~myuserid/Downloads/gcc-arm-none-eabi-6-2017-q2-update-mac.tar.bz2
  ```

  Use a opção tar "j" se o final do arquivo for .bz2. Use "z" quando o final for .gz.

- ✔ Uma vez extraído o arquivo tar, ele pode produzir um grande diretório nomeado como: gcc-arm-none-eabi-6-2017-q2-update. Agora é um bom momento para encurtar isso:

  ```
  # mv gcc-arm-none-eabi-6-2017-q2-update gcc-arm
  ```

  Isso renomeará o diretório /opt/gcc-arm-none-eabi-6-2017-q2-update para um nome mais gerenciável /opt/gcc-arm.

- ✔ Agora, saia do diretório atual e retorne ao diretório de desenvolvedor. Nesse diretório, adicione o seguinte caminho ao seu PATH:

  ```
  ~$ export PATH="/opt/gcc-arm/bin:$PATH"
  ```

- ✔ Neste ponto, você pode testar seu Compilador ARM Cross:

  ```
  ~$ arm-none-eabi-gcc --version
  ```

  ```
  arm-none-eabi-gcc (GNU Tools for ARM Embedded Processors
  6-2017-q2-update) 6.3.1 20170620 (release) [ARM/embedded-6-branch revision 249437]
  Copyright (C) 2016 Free Software Foundation, Inc.
  This is free software; see the source for copying conditions. There is NO
  warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
  ```

Se o compilador não iniciar e enviar uma mensagem como esta:

```
-bash: arm-none-eabi-gcc: command not found
```

Sua variável PATH não está configurada corretamente, ou não foi exportada, ou as ferramentas instaladas estão usando um prefixo diferente. Se necessário, execute o seguinte (a saída desse comando foi abreviada):

```
~$ ls -l /opt/gcc-arm/bin

total 75128
-rwxr-xr-x@ 1 root  wheel  1016776 21 Jun 16:11 arm-none-eabi-addr2line
-rwxr-xr-x@ 2 root  wheel  1055248 21 Jun 16:11 arm-none-eabi-ar
-rwxr-xr-x@ 2 root  wheel  1749280 21 Jun 16:11 arm-none-eabi-as
. . .
-rwxr-xr-x@ 2 root  wheel  1206868 21 Jun 16:11 arm-none-eabi-g++
-rwxr-xr-x@ 2 root  wheel  1202596 21 Jun 16:11 arm-none-eabi-gcc
. . .
```

Se você obteve seu Compilador ARM Cross de uma fonte diferente da indicada, talvez não tenha os nomes dos prefixos. Se você ver o nome do arquivo gcc em vez de arm-none-eabi-gcc, precisará invocá-lo como simplesmente gcc. Mas tenha cuidado neste caso, porque seu Compilador ARM Cross pode confundir com o compilador da plataforma. O prefixo arm-none-eabi- impede essa confusão. Quando você usar seu gcc cross plataforma, verifique se o compilador correto está sendo usado quando digitar o comando type:

```
~$ type gcc

arm-none-eabi-gcc is hashed (/opt/gcc-arm/bin/gcc)
```

Se seu bash estiver localizando o gcc em um diretório diferente daquele que você instalou, seu PATH não foi definido corretamente. Se você precisar alterar o prefixo das ferramentas, o ~/stm32f103c8t6/Makefile.incl de nível superior deve ser editado:

Modifique a seguinte linha para adequá-la e salvá-la novamente:

```
PREFIX      ?= arm-none-eabi
```

Em uma situação normal onde o prefixo da cross-platform é usada, você também deve poder fazer esta verificação:

```
~$ type arm-none-eabi-gcc

arm-none-eabi-gcc possui hash (/opt/gcc-arm/bin/arm-none-eabi-gcc)
```

Isso confirma que o compilador está sendo executado no diretório instalado /opt/gcc-arm.

NOTA: A variável PATH precisará ser modificada a cada nova sessão do terminal para usar as ferramentas do cross-compilador. Convém criar um script para automatizar essa tarefa.

Construindo (build) o software

Neste ponto, você instalou o framework para o MCU stm32f103c8t6, a biblioteca libopencm3, o FreeRTOS, o compilador ARM Cross e suas ferramentas, e configurou as variáveis de PATH, então, agora você deve ser capaz de mudar para o diretório stm32f103c8t6 e digitar make (alguns usuários podem precisar para usar o gmake), conforme mostrado a seguir.

```
~$ cd stm32f103c8t6
~/stm32f103c8t6$ make
```

O primeiro comando é apenas uma mudança de diretório. O segundo criará o subdiretório ~/stm32f103c8t6/libopencm3, seguido por todos os outros subdiretórios. Sempre existe a possibilidade de que uma nova versão do libopencm3 possa criar problemas de 'build'. É difícil prever isso, mas, aqui estão algumas possibilidades e soluções:

1. Algo no libopencm3 é sinalizado como um erro pelo compilador ARM Cross, onde anteriormente era aceitável. Você pode:

- ✔ Corrigir ou contornar o problema nas fontes da biblioteca libopencm3;
- ✔ Tente uma versão anterior do compilador, se não tiver sucesso tente uma versão mais recente. As ferramentas mais recentes geralmente corrigem o problema. Para referência, a versão usada foi "GNU Tools for ARM Embedded Processors 6-2017-q2-update) 6.3.1 20170620.";
- ✔ Instale uma versão mais antiga da biblioteca libopencm3.

2. Algo no framework do MCU está quebrado. Verifique por atualizações no repositório do git. À medida que os problemas se tornam conhecidos, as correções serão aplicadas e liberadas no repositório. Verifique também o arquivo README.md de nível superior.

Ferramenta ST-Link

Há finalmente um software que precisa ser instalado. Se você ainda não instalou o ST-Link usando o gerenciador de pacotes do seu sistema, você precisará instalá-lo agora. Mesmo que você tenha instalado, pode estar desatualizado. Vamos testar para ver:

```
~$ st-flash
```

Provavelmente apareça uma mensagem de comando invalido, seguido de várias mensagens de help. Procure pela seguinte mensagem de help:

```
./st-flash [--debug] [--reset] [--serial <serial>] [--format <format>] \
  [--flash=<fsize>] {read|write} <path> <addr> <size>
```

Se você não encontrar a opção --flash = <fsize> mencionada, convém baixar uma versão mais recente no github e faça o 'build' partir do código fonte. Isso é necessário apenas se você deseja usar mais de 64K de memória flash. Em nenhum projeto atingiremos esse limite.

As pessoas relataram que muitas das unidades STM32F103C8T6 suportam 128K de flash memória, mesmo que o dispositivo relate que possui apenas 64K. O seguinte comando analisa um dispositivo que estiver conectado. Se não conectou o seu programador ST-Link no Desktop/Notebook, então, conecte antes de realizar esse teste:

```
~$ st-info –probe

Found 1 stlink programmers
serial: 493f6f06483f53564554133f
openocd: "\x49\x3f\x6f\x06\x48\x3f\x53\x56\x45\x54\x13\x3f"
flash: 65536 (pagesize: 1024)
sram: 20480
chipid: 0x0410
descr: F1 Medium-density device
```

As informações relatadas indicam que o dispositivo suporta apenas 65536 bytes (64 KB) de flash. No entanto, eu sei que posso usar até 128K de Flash. O uso do programador ST-Link V2 será abordado nessa série.

Se você não possui esses utilitários instalados, faça-o agora usando apt-get ou qualquer que seja o seu gerenciador de pacotes. Na falta de instalação do pacote, você pode fazer o download das últimas código fonte do github aqui:

```
$ cd ~
~$ git clone https://github.com/texane/stlink.git
~$ cd ./stlink
~/stlink$ make
~/stlink$ cd build/Release
~/stlink$ sudo make install
```

Se você tiver problemas com isso, consulte os seguintes recursos online:

- ✔ O arquivo README.md em https://github.com/texane/stlink;
- ✔ https://github.com/texane/stlink/blob/master/doc/compiling.md;
- ✔ Verifique se o libusb está instalado;
- ✔ Algumas distribuições Linux podem exigir que você também execute sudo ldconfig após a instalação.

Com todos os softwares instalados, podemos finalmente abordar o hardware e fazer algo com isso. Conforme a proposta inicial vamos trabalhando nas bases para fundamentar o aprendizado, portanto, aqui concluo o quarto artigo da série.