Introdução

Esse é primeiro artigo da série "A Blue Pill", que não é aquela pílula azul, mas, também excita quem gosta de tecnologia. Os primeiros artigos conterão os fundamentos para desenvolvermos um tema por completo, e não apenas apresentar alguns efeitos especiais que impressionam, mas, não desenvolve pessoas. Na área de tecnologia vemos muitas coisas que enchem os olhos, mas, se quiser iniciar estudos autodidáticos sobre algo interessante, percebe que está num mato sem cachorro.

O que motivou eu trilhar por caminhos tortuosos em busca de conhecimento sobre a Blue Pill? Um dos meus hobbies era programar Arduíno UNO, apenas por diversão. Num determinado momento quis estudar outro microcontrolador, que tivesse melhor performance de forma geral e recursos para desenvolvimento. Então, comecei a pesquisar na internet. Queria um MCU com site de suporte, ferramentas para desenvolvimento e outros recursos que me auxiliassem. Encontrei uma empresa chamada STMicroelectronics (www.st.com), com muito daquilo que procurava. Com vasta gama de opções, o MCU STM32F103C8T6 foi minha escolha inicial, então, descobri que havia uma placa de desenvolvimento, denominada Blue Pill. Bingo!

No site da STMicroelectronics tem as ferramentas de desenvolvimento, como por exemplo, STM32Cube. Um problema pessoal que encontrei foi o nível de abstração dessas ferramentas. Como tenho um perfil bastante técnico, coisas com um nível muito alto de abstração me incomoda. Sei que essas novas ferramentas dão produtividade, mas, não estava querendo nada fast food. Gosto de ver as coisas por trás dos bastidores, então, ainda não estava totalmente no caminho que desejava. Pesquisa aqui e ali, então, encontrei um ebook chamado "Beginning STM32 – Developing with FreeRTOS, Libopencm3 and GCC- by Warren Gay". Ao olhar seu conteúdo com atenção, pude ver que realmente seria minha referência.

Na proposta do Warren Gay, houve uma decisão consciente de escolher um ambiente de desenvolvimento neutro à sua plataforma de desenvolvimento de desktop preferida. Havia várias IDE baseadas no Windows disponíveis, com licenças variadas, mas, essas IDEs mudam, as licenças mudam e as bibliotecas associadas mudam rapidamente com o tempo. Então, ele optou pelo uso de uma abordagem de código aberto, que tem a vantagem de você estar protegido contra todas essas mudanças repentinas.

Você pode usar todo o seu código e suas ferramentas de suporte, sabendo que todos poderão ser restaurados para a operação daqui a dez anos, se necessário. A restauração de software licenciado, por outro lado, deixa você vulnerável, as licenças expiradas ou sites online que já não funcionam mais. Nessa série eu sigo a linha do Warren Gay, então, desenvolveremos projetos com base nas seguintes ferramentas e bibliotecas de código aberto:

- ✔ Linux Ubuntu LTS 18.04 (Sistema Operacional: código aberto);
- ✔ gcc / g ++ (coleção de compiladores GNU: código aberto);
- ✔ Make (GNU binutils: código aberto);
- ✔ Libopencm3 (biblioteca: código aberto);
- ✔ FreeRTOS (biblioteca: código aberto e gratuito para uso comercial).

A série não é apenas uma tradução do livro do Warren Gay, mas, tem o livro "Beginning STM32" como referência, e com outros complementos encontrados nos manuais da ST, que são:

- ✔ Manual STM32F103x8 (DocID13587 Rev 17);
- ✔ Reference manual RM0008 (DocID13902 Rev 16).

Sistema Embarcados com Tecnologia ARM

Existe um interesse considerável na plataforma ARM Cortex porque os dispositivos ARM são encontrados em qualquer aplicação. As unidades que contêm dispositivos ARM variam desde pequenos microcontroladores de sistemas embarcados, telefones celulares e servidores maiores executando o Linux. Em breve, os dispositivos ARM também estarão presentes em maiores números nos data centers. Portanto, são boas razões para se familiarizar com a tecnologia ARM.

Com essa tecnologia sendo aplicada de microcontroladores a servidores completos, a questão que surge naturalmente é "Por que estudar a programação de dispositivos embarcados? Por que não focar nos sistemas de usuário final executando Linux, como o Raspberry Pi?".

A resposta simples é que os sistemas embarcados têm bom desempenho em cenários que são estranho para sistemas maiores. Eles são frequentemente usados para interagir com o mundo físico. Eles estão entre o mundo físico e um sistema de desktop, por exemplo, um simples teclado usa uma Unidade de Microcontrolador (MCU) dedicado para scanear as teclas do teclado e relatar os eventos de pressionamento de teclas. Isso não apenas reduz a quantidade de fiação, mas, também libera processamento da CPU principal, evitando processar tarefas específicas.

Outras aplicações incluem sistemas embarcados no chão de fábrica para monitorar temperatura, segurança e detecção de incêndio. Não faz sentido usar sistemas maiores e completos para este tipo de finalidade. Sistemas embarcados independentes economizam dinheiro e inicializam imediatamente. Finalmente, o tamanho pequeno de um MCU faz dele uma ótima opção em drones voadores, onde peso é um fator crítico.

O desenvolvimento de sistemas embarcados tradicionalmente exigia os recursos de duas disciplinas:

- Engenheiro de hardware;
- Desenvolvedor de software.

Frequentemente, uma pessoa recebe a tarefa de projetar o produto final. Os engenheiros de hardware são especializados no design dos circuitos eletrônicos envolvidos, mas, eventualmente o produto requer software. Isso pode ser um desafio, porque as pessoas de software geralmente não possuem o conhecimento em eletrônica, e os engenheiros geralmente não possuem a experiência em software. Devido a redução de orçamento e de prazos de entrega, o engenheiro eletrônico também se torna o engenheiro de software.

Não há desvantagem uma pessoa executar os dois aspectos do design, desde que tenha conhecimento e habilidades necessários. Seja você um engenheiro eletrônico, desenvolvedor de software, hobista ou desenvolvedor amador, não há nada melhor como uma prática pé no chão para ajudá-lo. É disso que trataremos nessa série.

Placa de Desenvolvimento STM32F103C8T6, Chamada de Blue Pill

O MCU STM32F103C8T6 em um projeto utilizando uma PCB azul é carinhosamente conhecido como "Blue Pill" inspirado no filme Matrix. Existem algumas PCBs mais antigas com cores vermelhas e foram referidos como "Red Pill". Ainda existem outros que são pretos e são conhecidos como "Black Pill". Assumiremos que temos o modelo Blue Pill.

O dispositivo ARM CORTEX-M3 escolhido é o STMicroelectronics STM32F103C8T6. A codificação é longa, então, vamos dividi-la:

- ✔ STM32 (plataforma STMicroelectronics);
- ✔ F1 (família de dispositivos);
- ✔ 03 (subdivisão da família de dispositivos);
- ✔ C8T6 (recursos que afetam a quantidade de SRAM, flash memória e assim por diante).

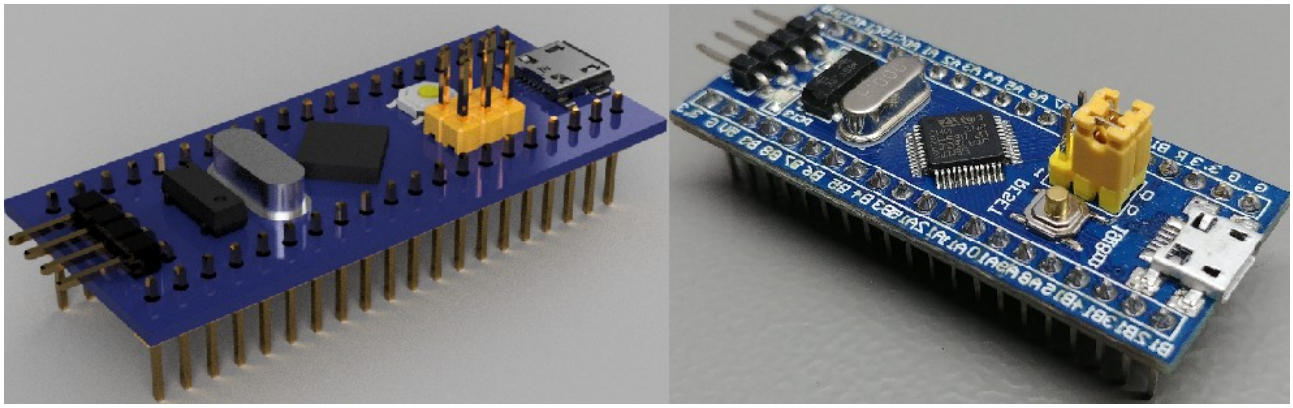

Figura 1 – STM32F103C8T6 Placa Blue Pill by www.grabcad.com

Pelo nome da plataforma sabemos que são dispositivos de 32 bits e são consideravelmente mais poderoso que os dispositivos de 8 bits. O F103 é um ramo (F1+03) da plataforma STM32. Esta subdivisão define a CPU e os recursos periféricos do dispositivo. Por último, o sufixo C8T6 define ainda mais os recursos do dispositivo, como a capacidade de memória e velocidade do clock do dispositivo.

A placa Blue Pill foi escolhida devido aos seguintes fatores:

- ✔ Custo baixo;
- ✔ Disponibilidade (encontrado facilmente);
- ✔ Capacidade avançada;
- ✔ Formato pequeno da PCB.

A Blue Pill provavelmente continuará sendo, por um bom tempo, o menor custo para estudantes e entusiastas que gostam de explorar a plataforma ARM Cortex-M3. O dispositivo está prontamente disponível e é extremamente capaz. Finalmente, seu formato pequeno dentro de uma placa de circuito impresso (PCB) que permite que terminais sejam soldados em suas bordas e conectados a uma Protoboard. Usando Protoboard é a maneira mais conveniente de realizar uma grande variedade de experimentos. O baixo custo tem outra vantagem, permite que você possua vários dispositivos para projetos envolvendo comunicações CAN, por exemplo.

Os periféricos do STM32F103 são simplesmente incríveis quando você considera seu preço. Os periféricos incluídos consistem em:

- ✔ 4 portas GPIO de 16 bits (a maioria é tolerante a 5 Volts);
- ✔ 3 x USART (Receptor/Transmissor Síncrono e Universal Assíncrono);
- ✔ 2 x controladores I2C;
- ✔ 2 x controladores SPI;
- ✔ 2 x ADC (conversor analógico para digital);
- ✔ 2 x DMA (controladores de acesso direto à memória);
- ✔ 4 x temporizadores;
- ✔ Temporizadores Watchdog;
- ✔ 1 x controlador USB;
- ✔ 1 x controlador CAN;
- ✔ 1 x gerador CRC;
- ✔ RAM estática de 20K;
- ✔ Memória Flash 64K (ou 128K);
- ✔ CPU ARM Cortex M3, clock máximo de 72 MHz.

No entanto, existem algumas restrições. Por exemplo, os controladores USB e CAN não pode operar ao mesmo tempo. Outros periféricos podem entrar em conflito com os pinos de E/S usados. A maioria dos conflitos de pinos é gerenciada através da configuração do AFIO (Função Alternativa de Entrada/Saída), permitindo que diferentes pinos sejam usados para a função de um periférico.

Na configuração do periférico, vários clocks separados podem ser habilitados individualmente para adaptar o uso de energia. A capacidade avançada deste MCU o torna adequado para estudo. O que você aprende sobre a família STM32F103 é aproveitado posteriormente em um MCU mais avançado como o STM32F407. A memória flash do STM32F103 é oficialmente 64K bytes, mas você pode chegar a suportar 128K.

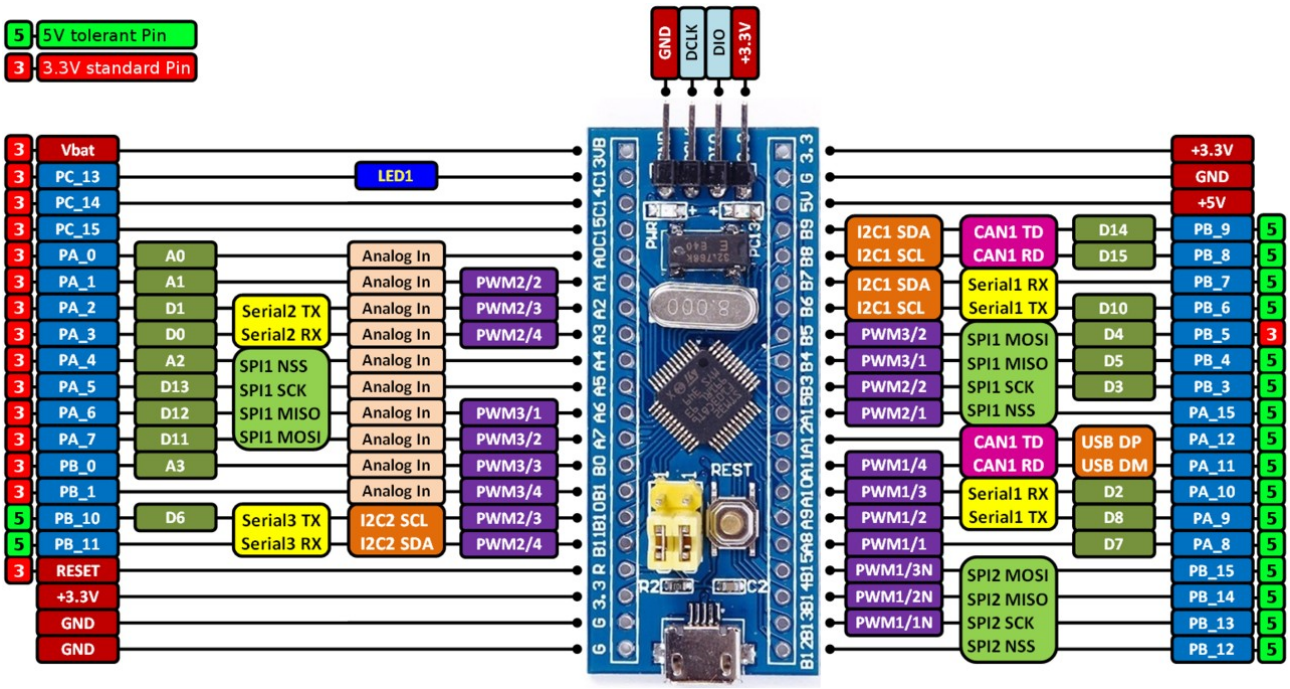

Figura 2 – Os pinos da Blue Pill by https://os.mbed.com

Com base no projeto original eu refiz o esquemático da Blue Pill usando o KiCad 5.1.4, conforme ilustrado na figura a seguir.

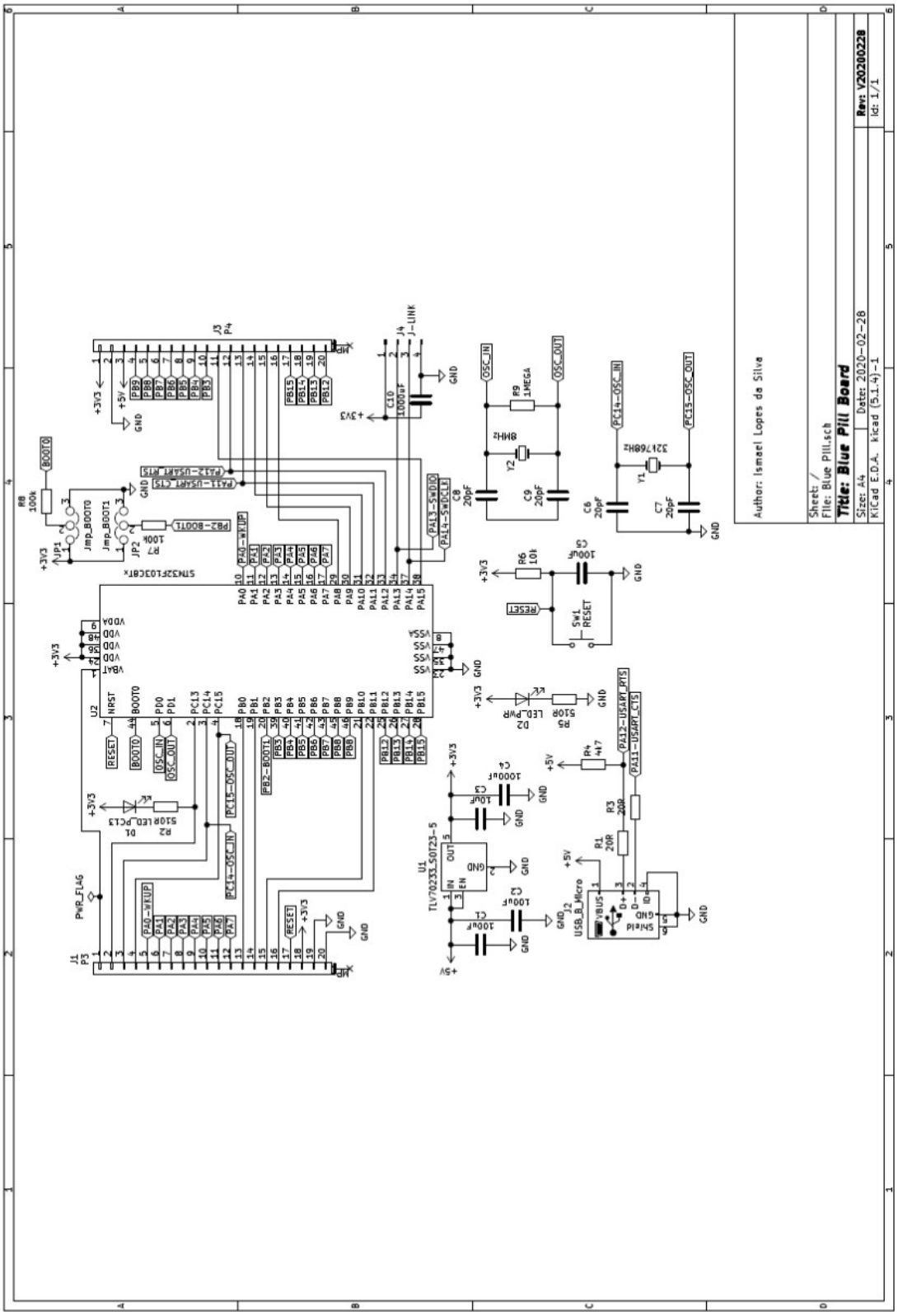

Figura 3 – O esquemático da Blue Pill usando KiCad 5.1.4

Conforme a proposta inicial vamos trabalhando nas bases para fundamentar o aprendizado, portanto, aqui concluo o primeiro artigo da série.