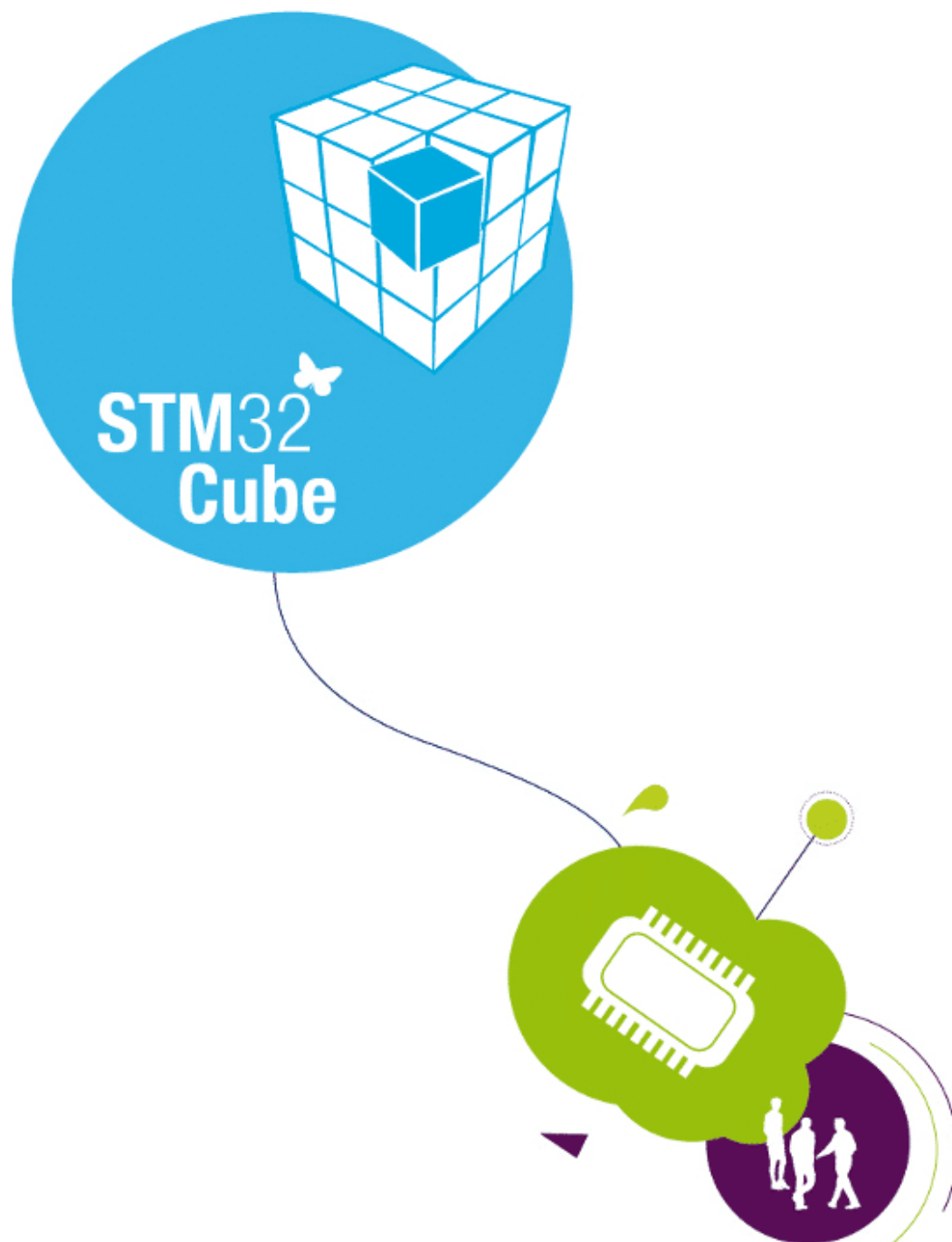


Crie projetos com o STM32Cube

Por **Eder Tadeu** - 02/12/2015



ÍNDICE DE CONTEÚDO [MOSTRAR]

Objetivo

O STM32Cube é uma ferramenta de desenvolvimento gratuita para microcontroladores ST, cujo objetivo é facilitar o ciclo de projeto, reduzindo esforços, tempo e custos. Como o próprio nome sugere, o STM32Cube foi idealizado pela empresa franco-italiana **STMicroelectronics** [↗](#), além disso, o STM32Cube cobre todo o portfólio STM32Fx (o “x” entende-se como 0 para ARM Cortex M0, 1 - Cortex M3, 3 e 4 - Cortex M4 e 7 - Cortex M7), incluindo a linha STM32Lx (L significa Low e diz respeito a linha Ultra Low Power).

Apresentação

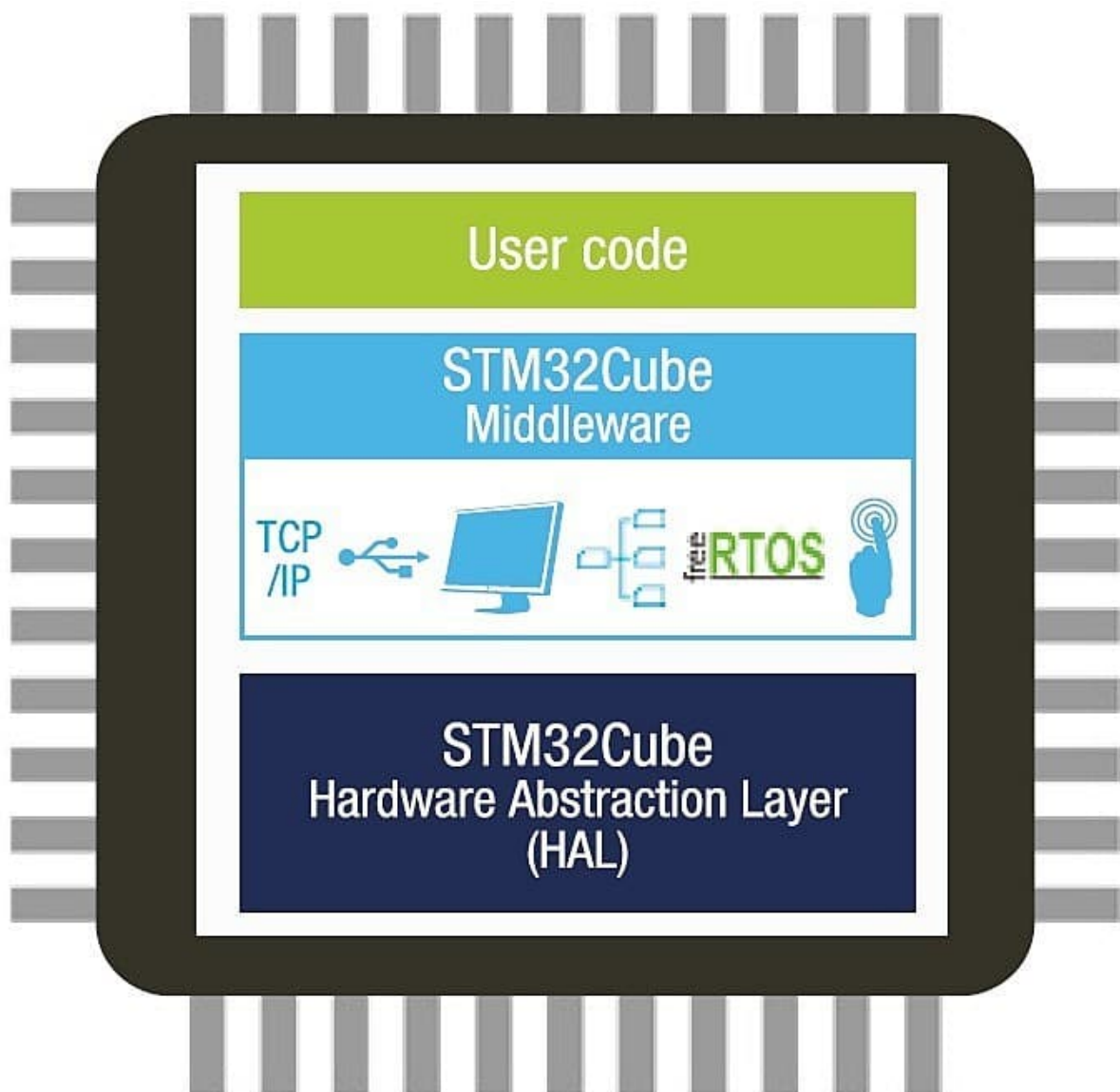


Figura 1: Diagrama de blocos da solução com STM32Cube.

O STM32Cube é composto pelo STM32CubeMX, ferramenta geradora de código fonte em linguagem “C”, mediante as configurações e opções escolhidas pelo usuário através dos **wizards** [🔗](#). Estas opções dizem respeito em habilitar Timers, ADC (Analog to Digital Converter), DAC (Digital do Analog Converter), USART (Universal Synchronous Asynchronous Receiver Transmitter), etc, observar toda a árvore de clock e escolher as opções pré-determinadas. Além disso, qual fonte de alimentação do clock. Dispõe também de funções dos GPIOs (General Propose Input Output), ou seja, input, output ou Alternate Function, tudo isso feito visualmente (figura 2). Além do código fonte gerado, também é disponibilizado o projeto pronto para uso de acordo a com escolha do usuário, ou seja, pode ser um projeto para as IDEs (Integrated Development Environment): EWARM, MDK-ARM v4, MDK-ARM v5, TrueSTUDIO e SW4STM32, e também exemplos, tudo alocado em um diretório informado pelo usuário antes da geração de códigos.

Também é embutido o STM32Cube HAL (HAL - Hardware Abstraction Layer), uma biblioteca software embarcado de abstração de camada do STM32, assegurando a máxima portabilidade através de todo o portfólio STM32 e, por fim, o conjunto de pacotes middleware, incluindo RTOS, USB, TCP/IP e gráficos.

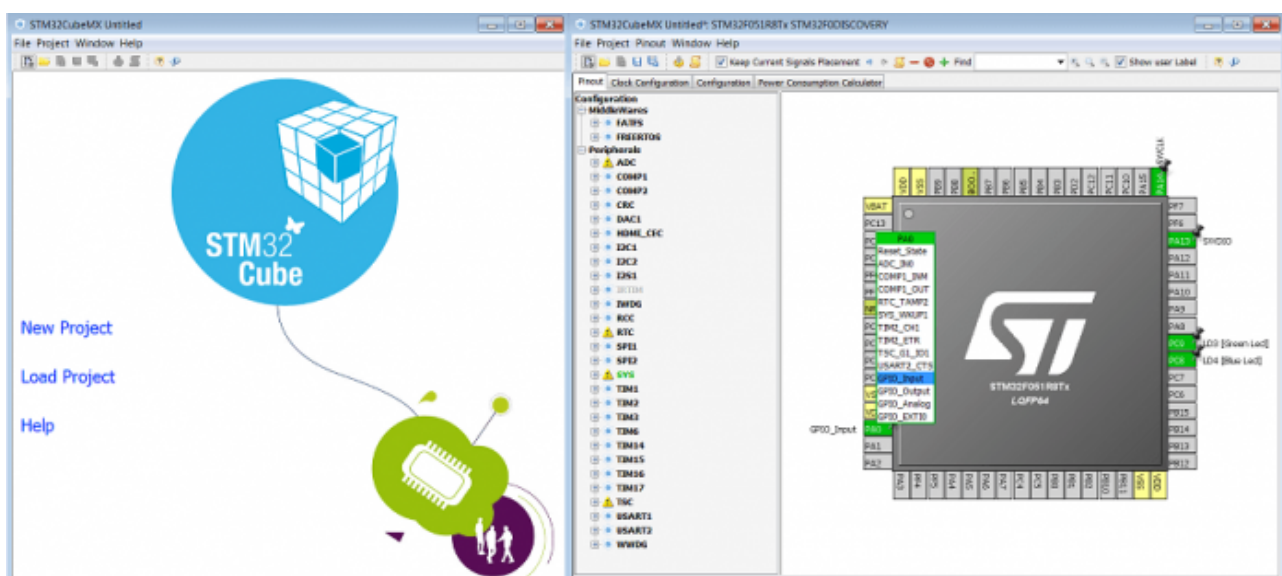


Figura 2: À esq. a tela inicial do STM32Cube e à dir. a tela de configurações.

Para aqueles que já utilizam essas ferramentas de configuração há bastante tempo, o antecessor do STM32CubeMX é o MicroXplorer. Atualmente a STMicroelectronics não mais dispõe para downloads, apenas o **STM32CubeMX** [↗](#) cuja versão atual pode ser vista na figura 3.



Figura 3: STM32CubeMX

Iniciando um projeto com o STM32Cube

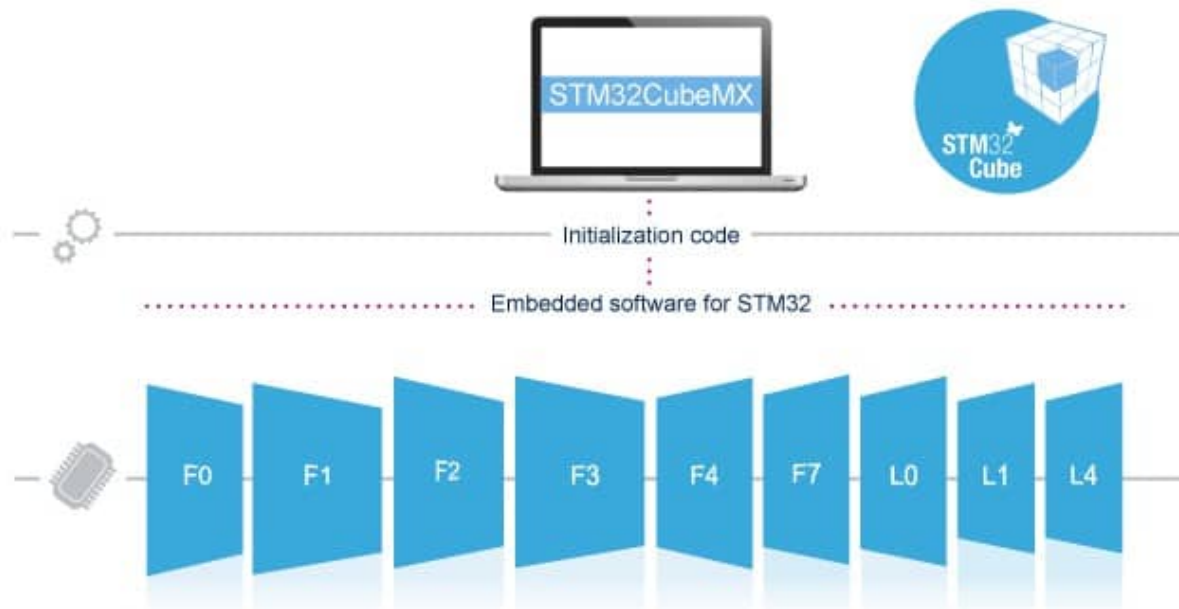


Figura 4: STM32CubeMX

Para iniciar um projeto no STM32Cube, deve-se primeiramente, fazer o download da ferramenta através do site da STMicroelectronics (vide link mencionado acima) e instalá-lo em seu computador. Feito isto, abra o software através do ícone criado na sua área de trabalho (STM32CubeMX) e escolha a opção New Project. Assim, será visto a tela com as opções disponíveis para os microcontroladores, situado na aba "MCU Selector", bem como, as opções para os kits de desenvolvimento suportados, situados na aba "Board Selector" (figura 5).

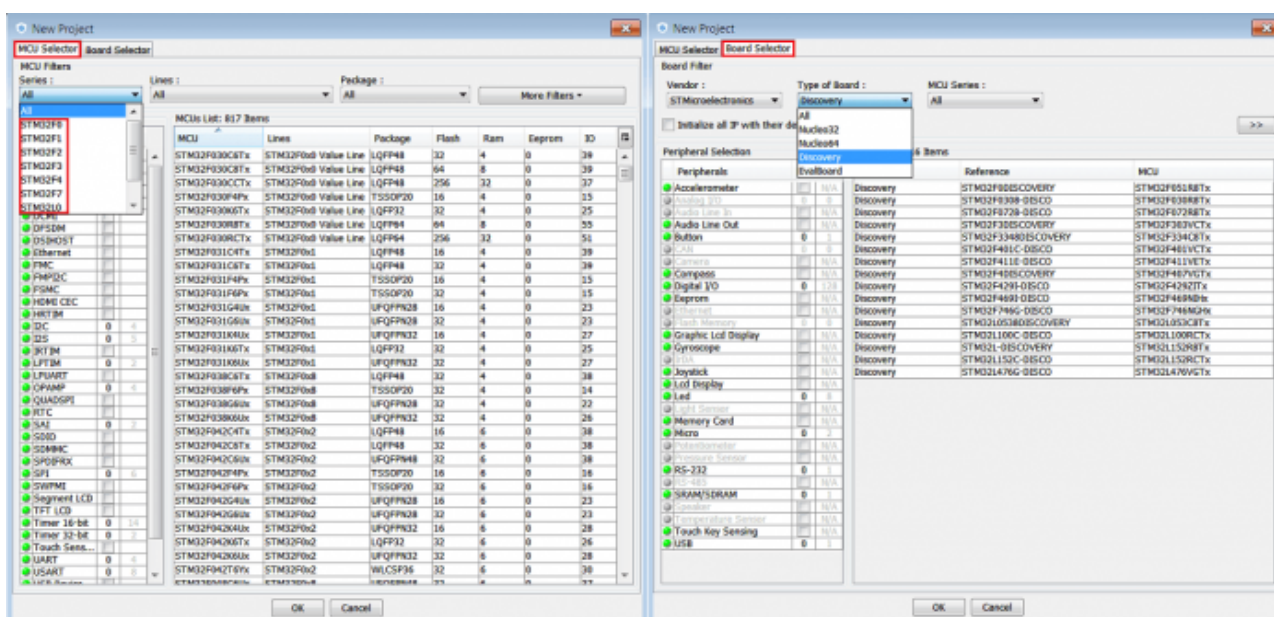



Figura 5: À esq. MCU Selector e à dir. Board Selector.

Para este projeto, iremos aproveitar o que foi desenvolvido no artigo “[Primeiros passos com a Placa STM32F0 Discovery](#)”, ou seja, a aplicação que foi desenvolvida, mas antes faremos a nossa configuração do Kit Discovery. Assim, escolha a primeira opção disponível no “Board List”, pois esta escolha é exatamente o kit em questão que utiliza o [STM32F051R8T6](#) , vide figura 6.

| Boards List: 16 Items | | |
|-----------------------|---------------------|---------------|
| Type | Reference | MCU |
| Discovery | STM32F0DISCOVERY | STM32F051R8Tx |
| Discovery | STM32F0308-DISCO | STM32F030R8Tx |
| Discovery | STM32F072B-DISCO | STM32F072R8Tx |
| Discovery | STM32F3DISCOVERY | STM32F303VCTx |
| Discovery | STM32F3348DISCOVERY | STM32F334C8Tx |
| Discovery | STM32F401C-DISCO | STM32F401VCTx |
| Discovery | STM32F411E-DISCO | STM32F411VETx |
| Discovery | STM32F4DISCOVERY | STM32F407VGTx |
| Discovery | STM32F429I-DISCO | STM32F429ZITx |
| Discovery | STM32F469I-DISCO | STM32F469NIHx |
| Discovery | STM32F746G-DISCO | STM32F746NGHx |
| Discovery | STM32L0538DISCOVERY | STM32L053C8Tx |
| Discovery | STM32L100C-DISCO | STM32L100RCTx |
| Discovery | STM32L-DISCOVERY | STM32L152R8Tx |
| Discovery | STM32L152C-DISCO | STM32L152RCTx |
| Discovery | STM32L476G-DISCO | STM32L476VGTx |

Figura 6: Placas suportadas.

Logo, estará disponível a imagem do microcontrolador pertencente ao Kit Discovery, além da árvore de periféricos que o componente possui para as configurações desejadas (à esquerda do aplicativo). De imediato, pode-se verificar que os pinos destinados aos LEDs (Light Emissor Diode) e ao Push Button (Chave Tact Switch azul) já estão configurados (figura 7).

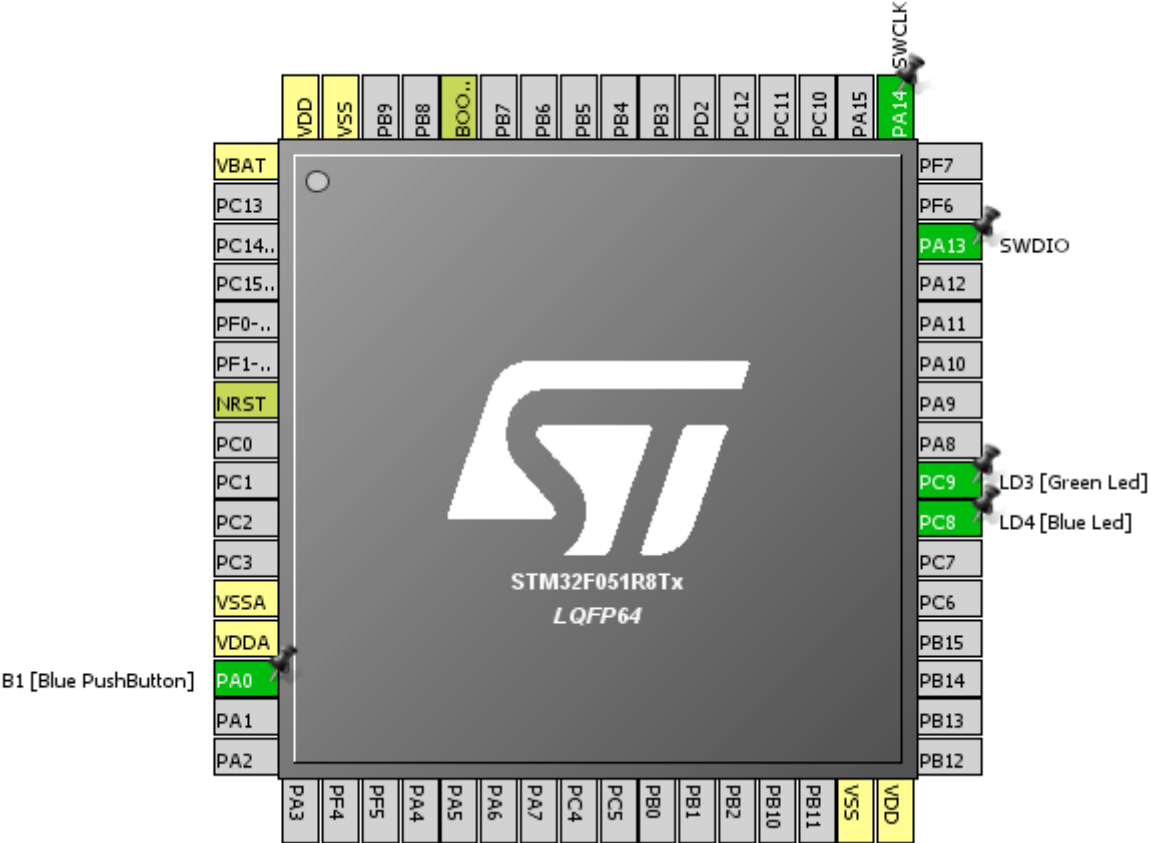


Figura 7: Periféricos suportados.

Quanto aos pinos PC8 e PC9, estes estarão configurados como GPIO_output. Caso queira confirmar, clique em cima do pino em verde e o que deve ser mostrado está na figura a seguir:

| PC9 |
|-------------|
| Reset_State |
| DAC1_EXTI9 |
| TIM3_CH4 |
| GPIO_Input |
| GPIO_Output |
| GPIO_Analog |
| GPIO_EXTI9 |

Figura 8:
Configuração
do pino PC9.

Em relação ao pino PA0, ele está configurado como GPIO_EXTI0, isto significa que ele será configurado como uma interrupção externa habilitado por este pino, contudo, na

nossa aplicação não o utilizaremos, somente os pinos dos LEDs. Mas caso queira, deixe-o como input para futuras aplicações (figura 9).

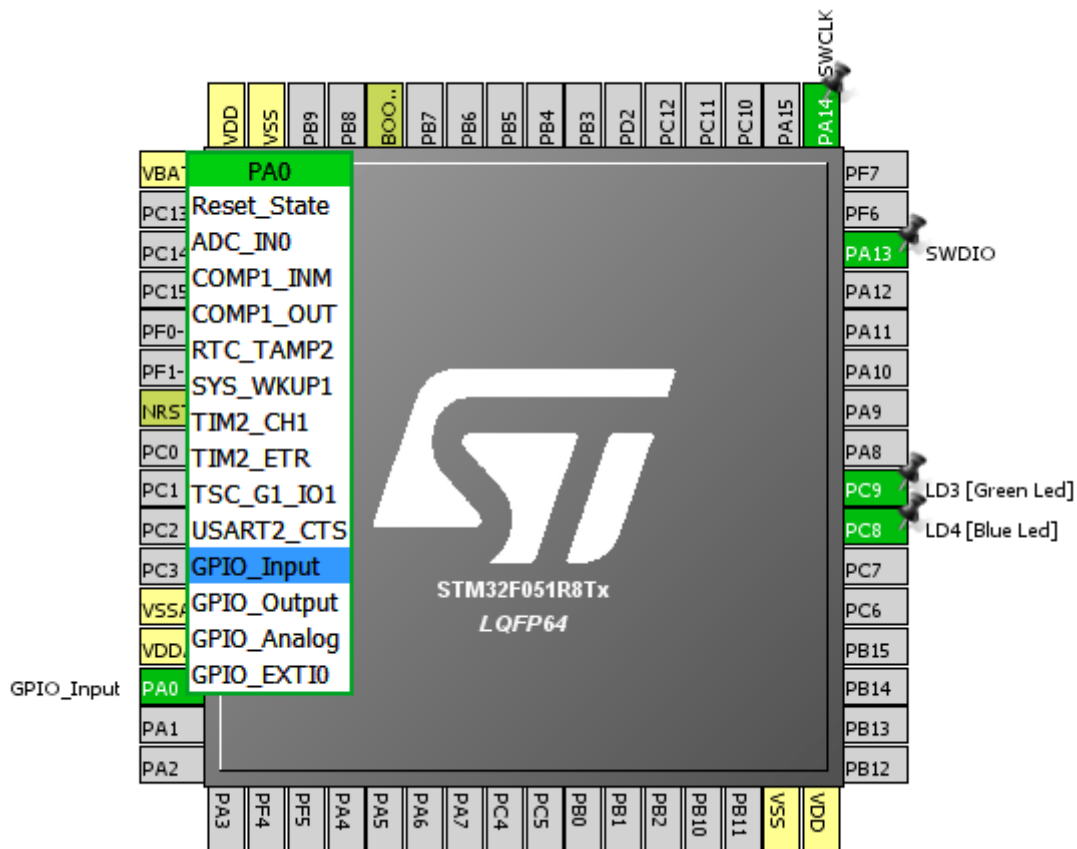


Figura 9: Configuração do pino PA0 como input.

Seguindo, clique na aba "Clock Configuration" e veja todo o sistema de clock e as opções possíveis. Neste projeto será utilizado o HSI (High Speed Internal - 8MHz) além do PLL (Phase Locked Loop), este que alimentará core do microcontrolador em 48MHz (figura 10).

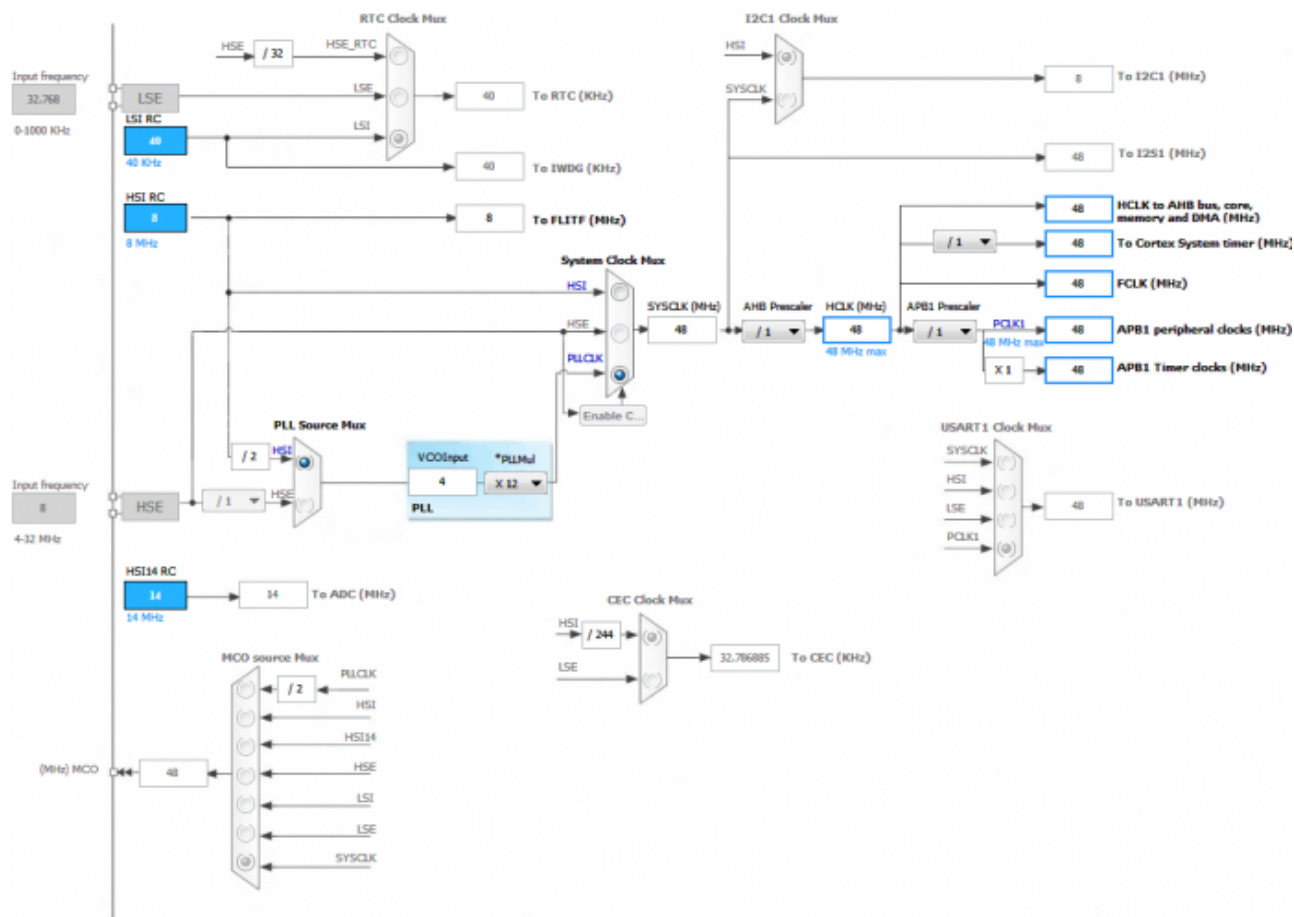


Figura 10: Sistema de clock.

Na aba seguinte, ou seja, Configuration, observa-se as opções de configurações: Middlewares, Multimedia, Control, Analog, Connectivity e Systems, neste último, pode ser visto outras opções como DMA (Direct Memory Access), GPIO, NVIC (Nested Vectored Interrupt Controller) e RCC (Reset and Clock Controller). Clique na opção GPIO conforme a figura 11.



Figura 11: Configuração.

Assim sendo, poderemos visualizar opções que dizem respeito ao pino que foi escolhido anteriormente, ou seja: push pull ou open drain, velocidade (low, medium ou high speed) e também deixá-lo como pull-up ou pull-down (figura 12).

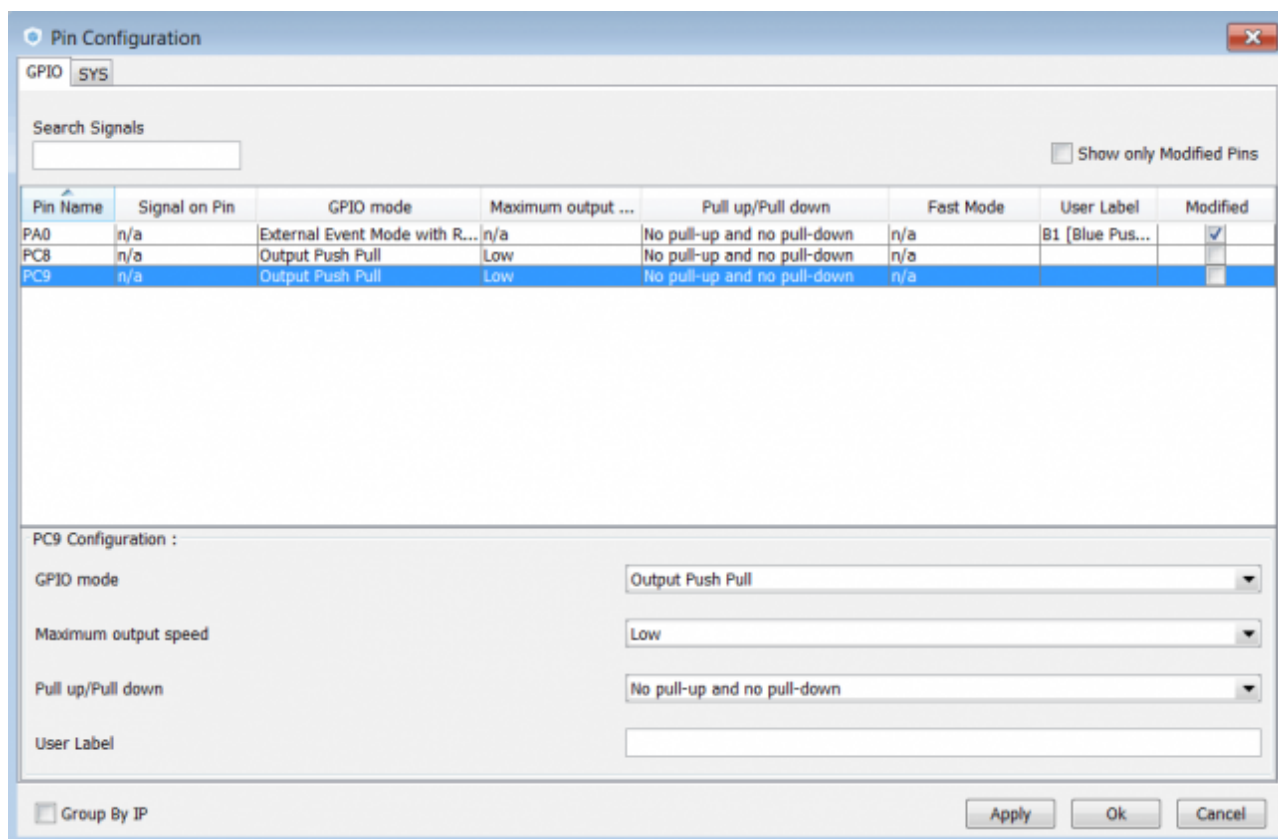


Figura 12: Configuração do pino.

Diante disto, podemos dizer que as configurações necessárias para “rodar” a aplicação já é suficiente, pois necessita apenas das configurações dos pinos e do clock do sistema. Assim, podemos gerar o código, escolher o tipo de projeto para a IDE e incluir o código da aplicação do “pisca-pisca” do artigo citado. Assim clique no ícone conforme a figura 13.

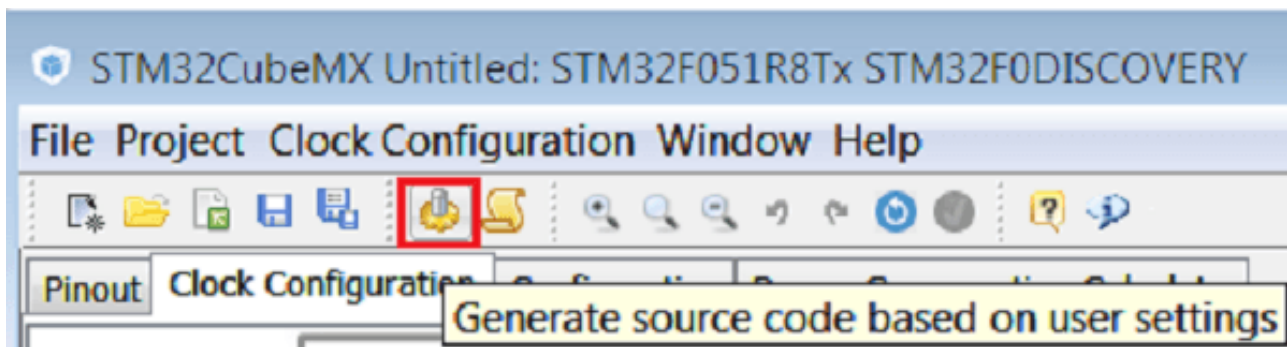


Figura 13: Ícone para geração do código fonte em C.

Uma vez clicado no ícone para gerar o código fonte, a tela seguinte poderá ser vista na figura 14. Nesta janela o usuário deverá informar o nome do projeto, a localidade em que o diretório com todos os arquivos ficará e também qual IDE será utilizada para abrir o projeto. Neste caso, escolheremos o MDK-ARM v4, depois clique em “OK”.

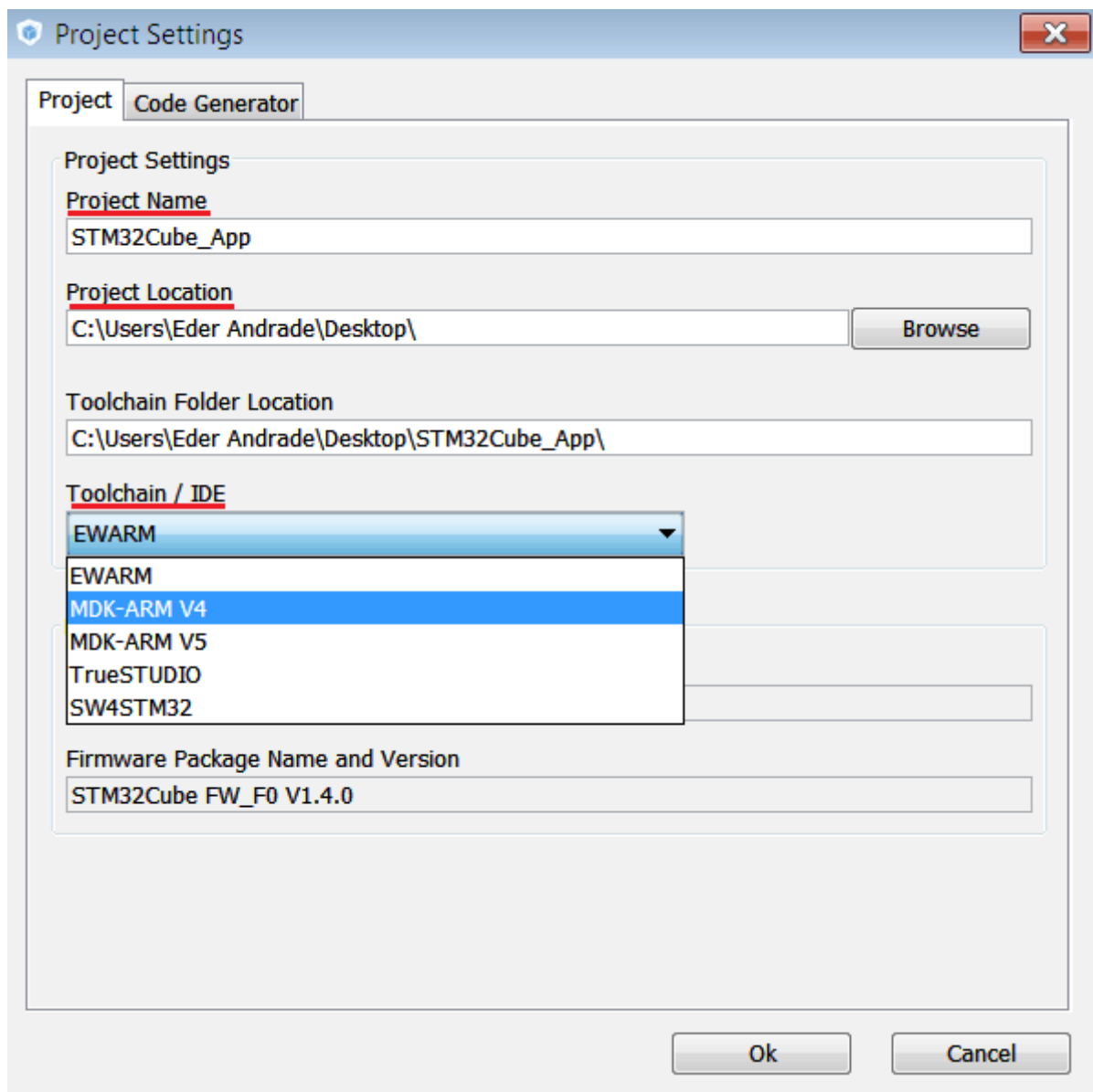


Figura 14: Novo projeto.

Feito isto, o projeto será criado, assim verifique se o diretório foi criado no local indicado. Este por sua vez, terá o nome do projeto informado, neste caso: "STM32Cube_App" (figura 15).

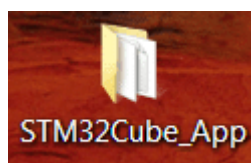


Figura 15: Projeto criado.

Dentro deste diretório, procure pelo subdiretório “MDK-ARM” e encontre o arquivo “.uVision4 Project”.

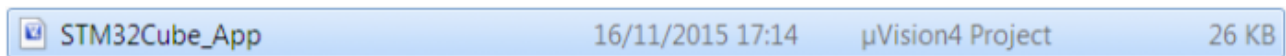


Figura 16: Arquivo uVision4 Project.

Abra-o, já na IDE é possível verificar a árvore do projeto à esquerda e também os arquivos incluídos (figura 17).

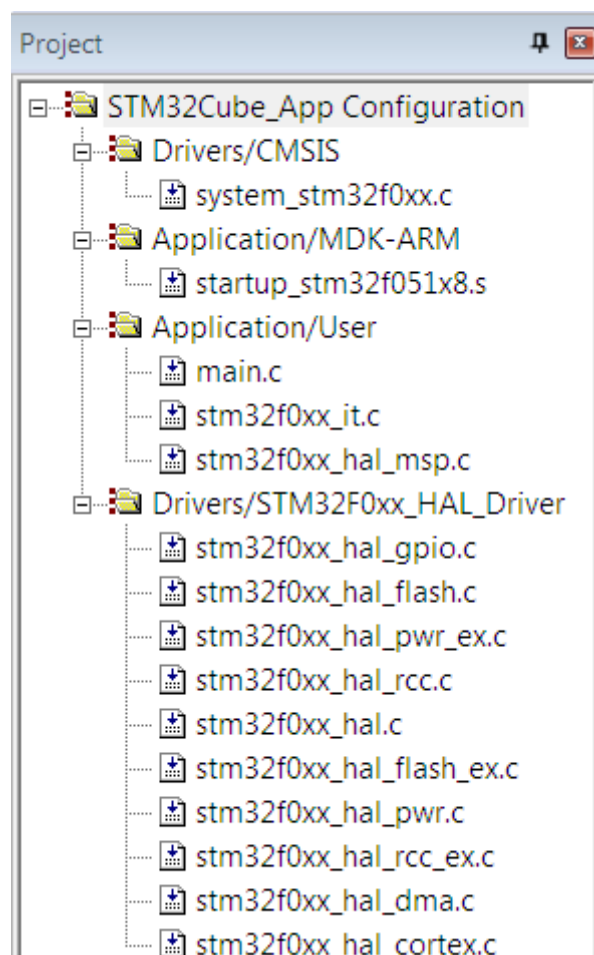


Figura 17: Árvore do projeto.

Abra o arquivo “main.c” e inclua as rotinas “vDelay()”, “vLED_Liga()”, “vLED_Desliga()”, as variáveis e também os “defines”. No “while(1)” inclua o “pisca-pisca” conforme a aplicação do “Primeiros passos com a Placa STM32F0 Discovery”.

O resultado esperado pode ser visto no vídeo a seguir:

No arquivo 'main.c', encontra-se a rotina "MX_GPIO_Init()", esta é responsável por configurar os três pinos do kit Discovery gerado a partir do STM32Cube. Assim gostaria que o leitor fizesse uma comparação com a rotina "vLEDs_Init()", desconsiderando o pino PA0, pois este não foi incluído no pisca-pisca, mas visualize a configuração dos pinos PC8 e PC9.

Conclusão

Podemos concluir, diante desta experiência, que o STM32Cube traz bastante benefícios, principalmente, no que diz respeito a tempo (consequentemente custo), pois o usuário que adquire ou já tem experiência com esta ferramenta, consegue configurar seu projeto em poucos minutos, e isto não é exagero. Além disso, pudemos importar a aplicação já feita do "pisca-pisca" sem se preocupar com as configurações do hardware, pois este é o principal motivo da ferramenta. Com isso, mostramos a agilidade em que se pode ganhar com um cenário deste, em que já se tem algo desenvolvido e apenas esperando a configuração do hardware.

Sugere-se que o usuário leia os códigos gerados, principalmente para configuração dos GPIOs e Clock, afim de comparação e também aprendizado.

Obrigado pela leitura!