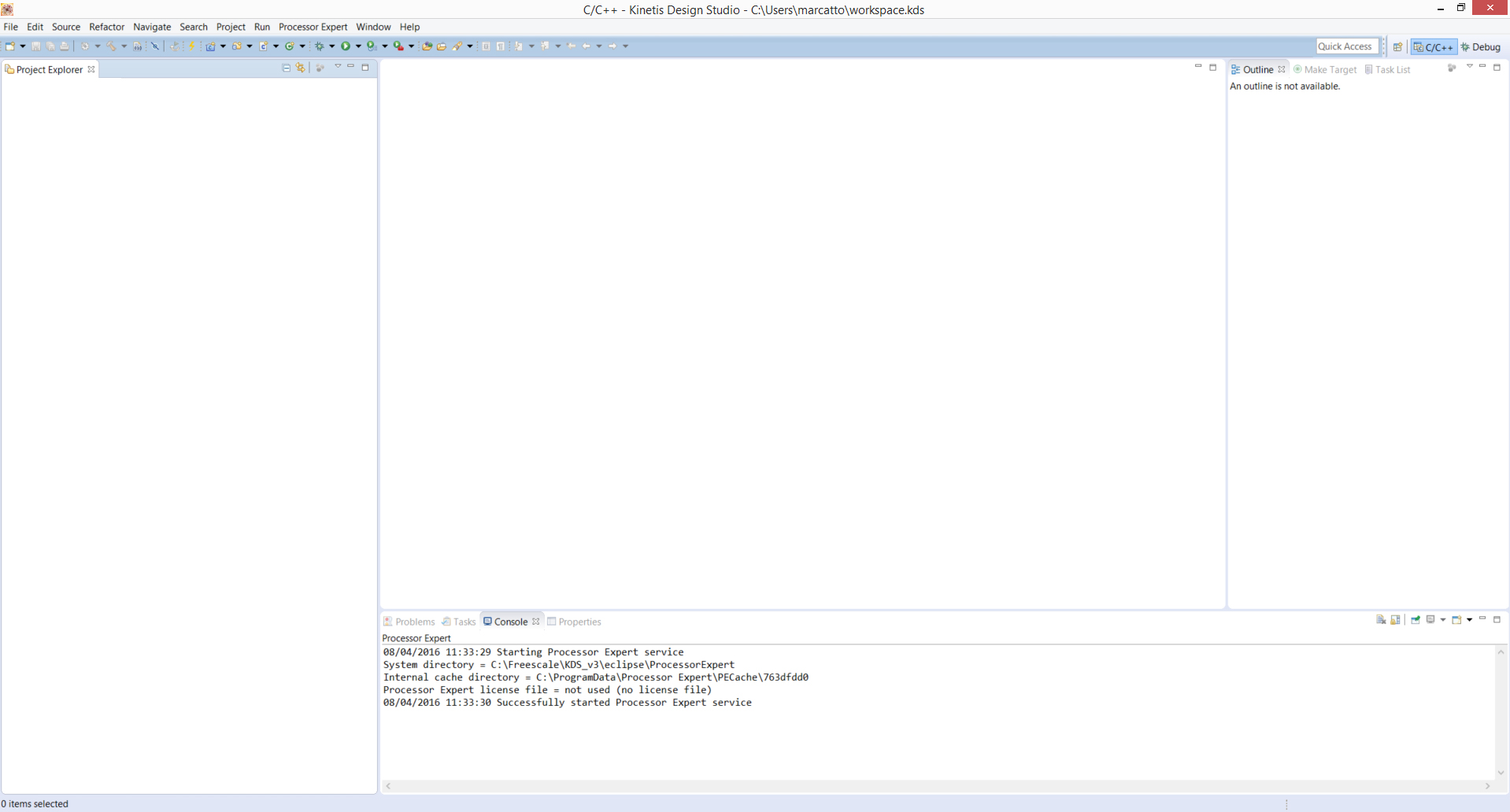
**Autores : Gabriel Silva Marcatto e Rodrigo Maximiano Antunes de Almeida**

**A IDE KDS**

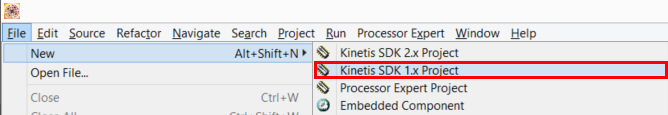
O [Kinetis Design Studio ( KDS )](http://www.nxp.com/products/software-and-tools/run-time-software/kinetis-software-and-tools/ides-for-kinetis-mcus/kinetis-design-studio-integrated-development-environment-ide:KDS_IDE" \t "_blank) é um ambiente de desenvolvimento integrado para microcontroladores Kinetis, que permite uma edição robusta para compilar e depurar seus projetos. Com base no [Eclipse IDE](https://eclipse.org/" \t "_blank), [GNU Compiler Collection (GCC)](https://gcc.gnu.org/) e [GNU Debugger (GDB)](https://www.gnu.org/software/gdb/), o Kinetis Design Studio oferece uma ferramenta simples de desenvolvimento que não possui limitação no tamanho do código, fazendo uso de algumas das melhores ferramentes open source disponívels.

A interface do Kinetis Studio é simples e fácil de utilizar, quem possui experiência com o eclipse IDE não enfrentará problemas ao migrar de programa.

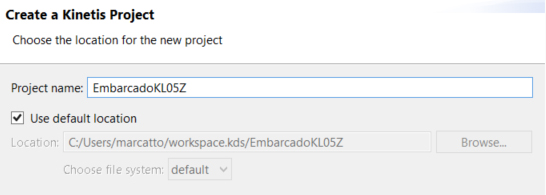


**Criando um Projeto**

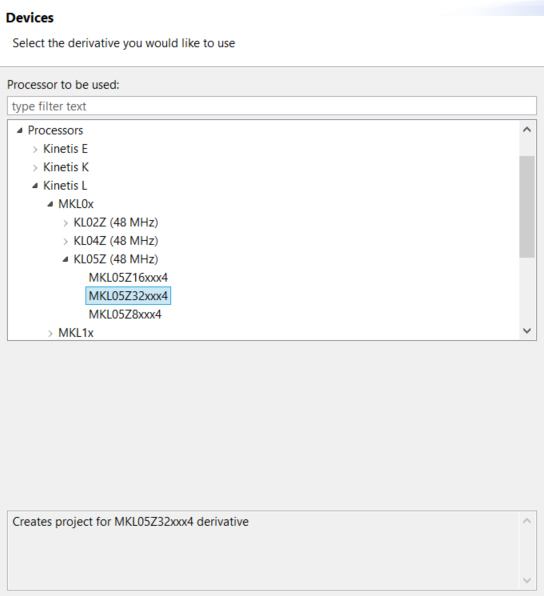
Para se criar um novo projeto, clque em **File > New > Kinetis SDK 1.x Project**. A NXP permite a customização dos SDKs para algumas famílias de microcontroladores, através do [NXP SDK Builder](http://kex.freescale.com/en/welcome" \t "_blank).



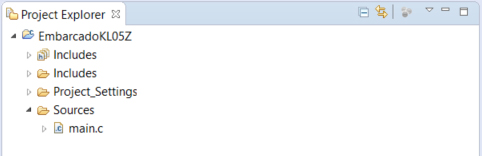
Abrirá uma janela para a escolha do nome do projeto e pasta destino.



Após escolher o nome, uma janela com os processadores e placas ***suportados*** pelo programa irá aparecer. Escolha o processador MKL05Z32xxx4.



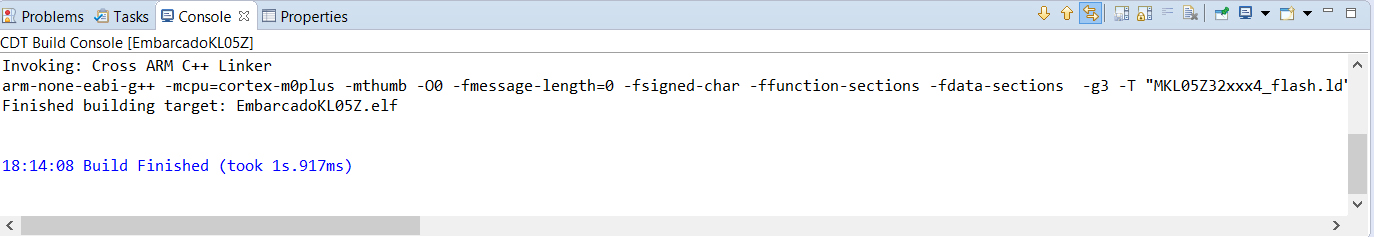
Criado o projeto, ele se encontrará no *Project Explorer*. O programa cria automaticamente o arquivo **main.c** com um código simples e pré determinado.



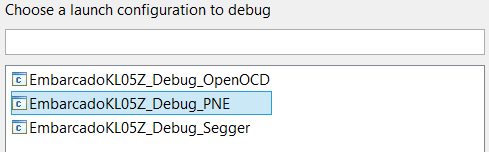
Para testar a placa podemos utilizar um programa simples que faz o acionamento do LED RGB. O código apresentado abaixo faz o uso das configurações mínimas para permitir que a placa seja iniciada corretamente e pisque o led azul, que está conectado no PIN 10 da porta B. Os detalhes do software, bem como as configurações iniciais, serão abordadas em outros artigos.

|  |  |
| --- | --- |
|  | //Autor: Gabriel Silva Marcatto  //       Rodrigo Almeida      #include "MKL05Z4.h"    #define bitSet(arg,bit) ((arg) |= (1<<bit))  #define bitClr(arg,bit) ((arg) &= ~(1<<bit))    static int i = 0;    void main(void){  //-------------------------Config para boot da placa -------------------------------------------------------      SIM\_BASE\_PTR ->SCGC5 |= (SIM\_SCGC5\_PORTA\_MASK | SIM\_SCGC5\_PORTB\_MASK); //init clock das portas      MCG\_BASE\_PTR ->C4 |= 0x80;           //configura para usar clock interno em 24MHz      PORTB\_PCR(5) = (PORT\_PCR\_MUX(1) | PORT\_PCR\_DSE\_MASK);   //portb 5, remover o NMI  //-----------------------------------------------------------------------------------------------------------      PORTB\_BASE\_PTR ->PCR[10] = (PORT\_PCR\_MUX(1) | PORT\_PCR\_DSE\_MASK);      bitSet(PTB\_BASE\_PTR ->PDDR, 10);      for(;;){          bitSet(PTB\_BASE\_PTR ->PDOR, 10);          for(float i = 0; i<100000; i++);          bitClr(PTB\_BASE\_PTR ->PDOR, 10);          for(float i = 0; i<100000; i++);      }  } |

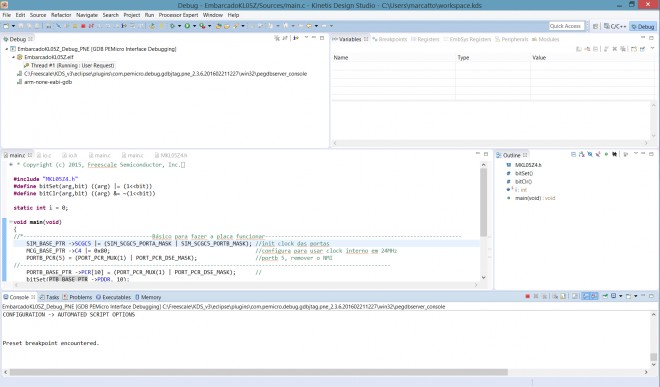
Para compilar o código, clique em **BUILD** (build). Não havendo nenhum erro, deverá aparecer no console: **Build Finished** (tempo de gravação).



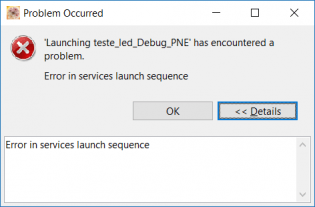
Após terminado o build, clique em **Debug** (debug). Uma janela irá aparecer para a escolha do modo de depuração (tipo de protocolo/debugger a ser utilizado). Para a Freedom KL05z selecione o **\_Debug\_PNE**:



Após a gravação da placa a IDE irá mudar um pouco a interface exibindo o modo de debug: uma mensagem de gravação completa, o código fonte com indicativo da linha atual, informações sobre a stack de chamada de funções, os valores das variáveis locais/globais e uma barra de ferramentas com controle para execução do código.



É importante lembrar que devemos parar a depuração antes de recompilar e regravar a placa. Se a placa não for pausada poderá acontecer um conflito e ocorrer o seguinte erro:



Para resolver esse problema basta desconectar/reconectar a placar ou reseta-la. A modificação e recompilação do código pode ocorrer tanto no modo de depuração quanto no modo C/C++ de exibição da IDE. Para alterar entre ambos vá ao canto superior direito e selecione o modo desejado: IDE_C_Debug .

**SoftPWM para exibição de cores**

Para testar todas as cores de um LED RGB podemos utilizar o código abaixo. Ele vai simular um soft PWM para cada um dos leds e ao longo do tempo vai alterar o valor de R, G e B, percorrendo grande parte das cores disponíveis. Este código pode ser facilmente adaptado para testar leds em outras placas.

|  |  |
| --- | --- |
|  | //Autor : Gabriel Silva Marcatto  // e Rodrigo Almeida    #include "MKL05Z4.h"    #define LigaR()    (PTB\_BASE\_PTR->PDDR |= 1 << 8)  #define DesligaR() (PTB\_BASE\_PTR->PDDR &= ~(1 << 8))  #define LigaG()    (PTB\_BASE\_PTR->PDDR |= 1 << 9)  #define DesligaG() (PTB\_BASE\_PTR->PDDR &= ~(1 << 9))  #define LigaB()    (PTB\_BASE\_PTR->PDDR |= 1 << 10)  #define DesligaB() (PTB\_BASE\_PTR->PDDR &= ~(1 << 10))    void main(void) {  //-------------------------Básico para fazer a placa funcionar------------------------------------------------------------------------  SIM\_BASE\_PTR->SCGC5 |= (SIM\_SCGC5\_PORTA\_MASK | SIM\_SCGC5\_PORTB\_MASK); //init clock das portas  MCG\_BASE\_PTR->C4 |= 0x80;   //configura para usar clock interno em 24MHz  PORTB\_PCR(5) = (PORT\_PCR\_MUX(1) | PORT\_PCR\_DSE\_MASK);  //portb 5, remover o NMI  //--------------------------------------------------------------------------------------------------------------------------------------  //config R8 G9 B10  PORTB\_BASE\_PTR->PCR[8] = (PORT\_PCR\_MUX(1) | PORT\_PCR\_DSE\_MASK); &nbsp;&nbsp; //configura PortB 8 como Output  PORTB\_BASE\_PTR->PCR[9] = (PORT\_PCR\_MUX(1) | PORT\_PCR\_DSE\_MASK); &nbsp;&nbsp; //configura PortB 9 como Output  PORTB\_BASE\_PTR->PCR[10] = (PORT\_PCR\_MUX(1) | PORT\_PCR\_DSE\_MASK); &nbsp; //configura PortB 10 como Output    int valorR = 0;  int valorG = 0;  int valorB = 0;    int tempo = 0;  int valor= 0;  int cor=1;  for (;;){  LigaR();  LigaG();  LigaB();    //liga g e b  for (int i = 0; i < 256; i++){  if (i > valorR){  DesligaR();  }  if (i > valorG){  DesligaG();  }  if (i > valorB){  DesligaB();  }  }  tempo++;    if (tempo >= 10){  //criando a rampa  valor++;  if (valor == 255){  valor = 0;  //rampa no maximo, troca a cor  cor++;  if (cor>0b111){  cor = 0b1;  }  }    //qual cor  acompanha a rampa;  if (cor & 0b001){  valorR = valor;  }else{  valorR = 0;  }  if (cor & 0b010){  valorG = valor;  }else{  valorG = 0;  }  if (cor & 0b100){  valorB = valor;  }else{  valorB = 0;  }  tempo = 0;  }  }  } |