



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Workshop CC1352R

Por Sara da Cunha Monteiro de Souza
Engenharia da Computação - IFF

O que vem por aí

- Interface de programação (IDE)
- Arquitetura CC1352R
- SDK
- TI RTOS
- Configurando RTOS e Debugando
- Usando Sensor Controller
- Usando SmartRF Studio
- Suporte
- Referências



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



INTERFACE PROGRAMAÇÃO (IDE)

- CCS, Uniflash, IAR, CCS Cloud, entre outros.
CCS: <http://www.ti.com/tool/ccstudio>
UNIFLASH: <http://www.ti.com/tool/UNIFLASH>
- **Recomendações:**
Windows com CCS.
Mantenha seu CCS atualizado. (Help -> Check for updates)
FAZER UMA CONTA NO SITE DA TEXAS
- **Sugestão para versionamento de código com o GitLab:**
GitBash:
<https://gitforwindows.org/>



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



CC1352R - Launchpad de Desenvolvimento

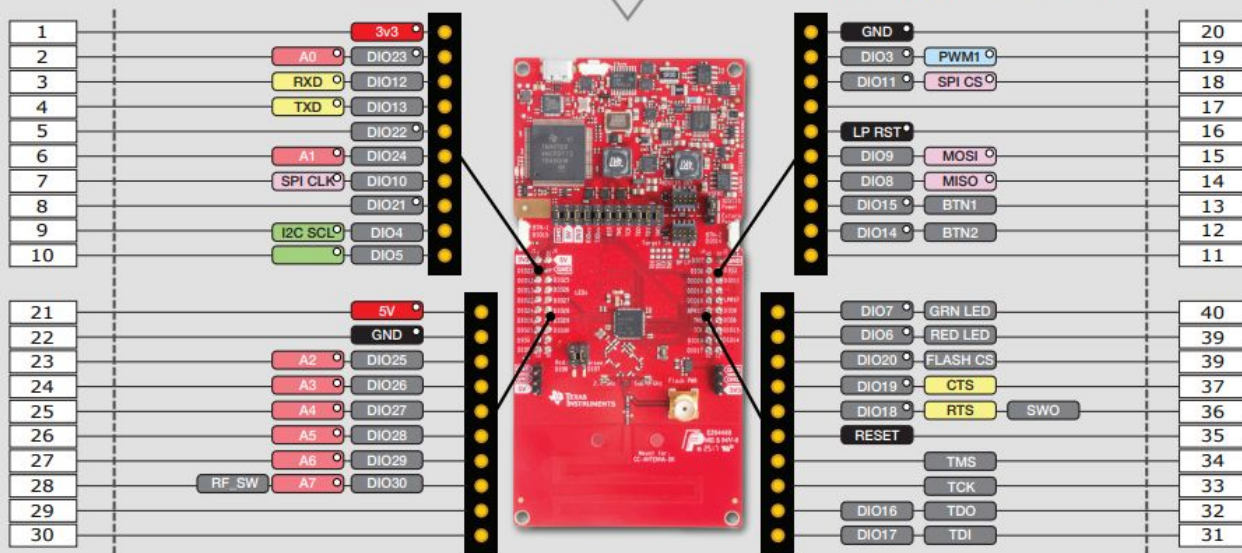


TI LaunchPad™ kit with CC1352 MCU

Microcontroller development kit for rapid prototyping
featuring the CC1352R microcontroller

PART NO: LAUNCHXL-CC1352R1

○ Pin aligns with LaunchPad pinout standard.



Explore all TI LaunchPad™ kits @ www.ti.com/launchpad

© 2017 Texas Instruments Incorporated. The platform bar, CC1312 and LaunchPad are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.



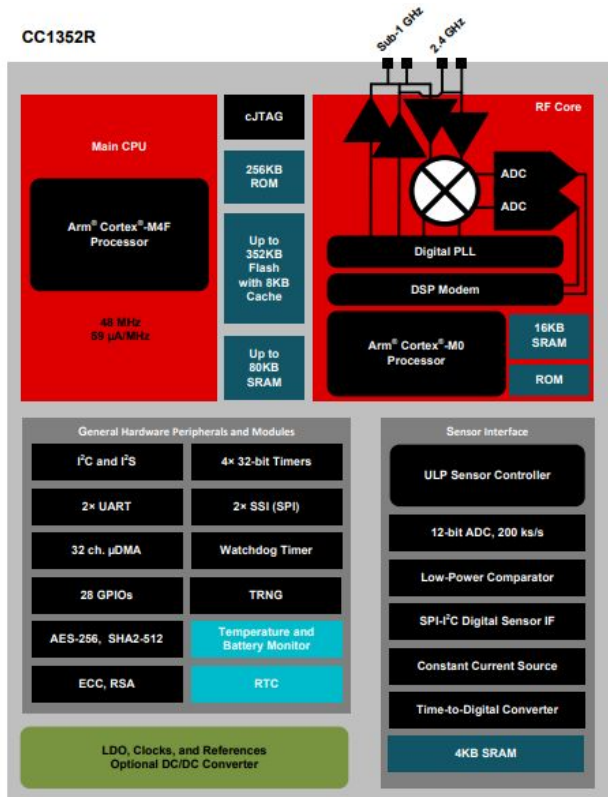
INSTITUTO
FEDERAL
Fluminense

Polo de Inovação

Campos dos Goytacazes



CC1352R - Arquitetura



Copyright © 2018, Texas Instruments Incorporated



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



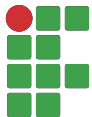
SDK - Simple Link Development Kit

API instalável via:

<http://www.ti.com/tool/SIMPLELINK-CC13X2-SDK>

O que o SDK proporciona?

- Portabilidade
- Abstração
- Integração
- Facilidade
- Documentação
- Exemplos



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



SDK - Simple Link Development Kit

Common SimpleLink™ Components

TI Drivers

(GPIO, I2C, UART, SPI, ADC,
PWM, ...)

Examples

POSIX

(IEEE Standard enabling code
portability between OSes)

Examples

Driver Lib

Examples

OS Kernel (optional)

TI-RTOS

FreeRTOS

Examples



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



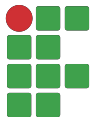
SDK - Simple Link Development Kit

TI DRIVERS: API fornecedora de drivers para explorar as funcionalidades dos periféricos para todos os dispositivos da linha SimpleLink.

DRIVER LIB: Camada de abstração de hardware (HAL), contém funções em C que abstraem a escrita aos registradores. TI Drivers e OS usam a camada para acessar o hardware. Podem fornecer melhor controle e otimização.

POSIX(Portable Operating System Interface): Padrão IEEE que garante compatibilidade entre diferentes Sistemas Operacionais. Faz a interface com o kernel. Quando uma thread é criada pelo POSIX, uma task é criada no RTOS. Possibilita o uso de códigos baseado em POSIX.

KERNEL: Fornece gerenciamento de tarefas, *scheduling*, *multitasking*, programação de *threads*, escalonamento, controle de prioridade, latência reduzida (RTOS), etc. (Open Source)



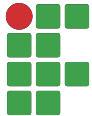
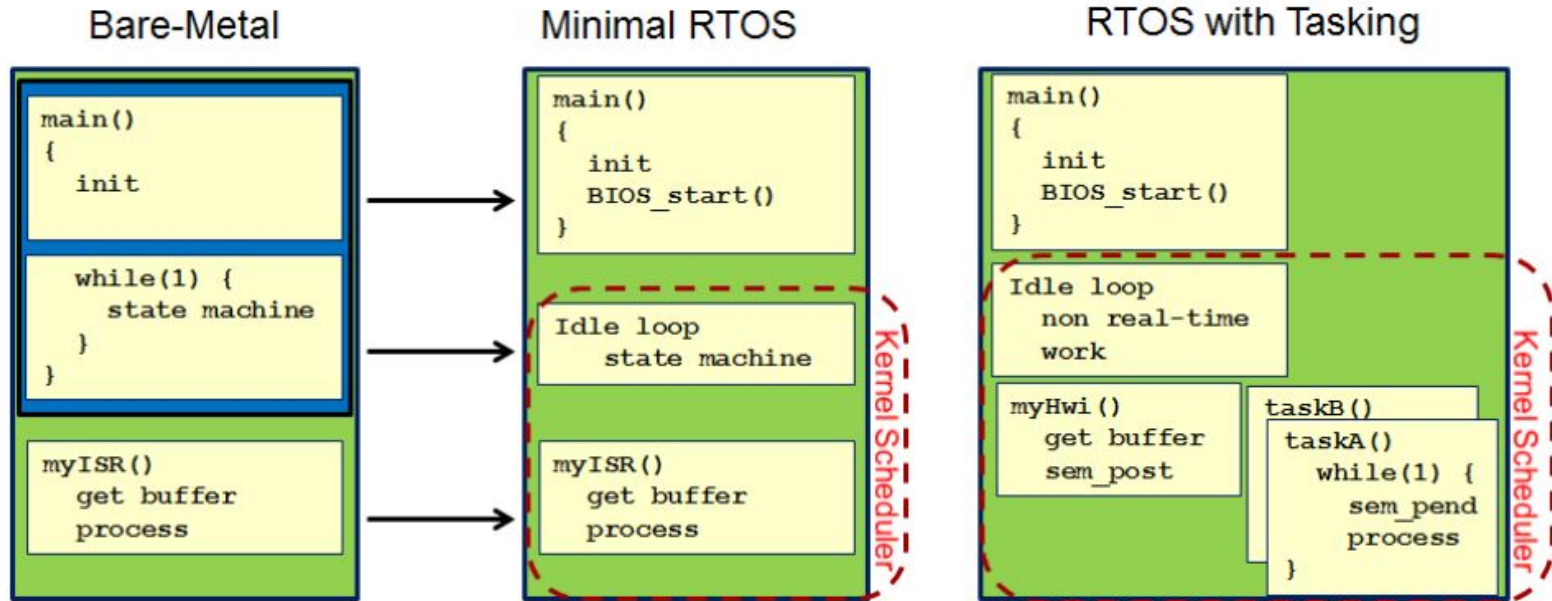
**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



TI - RTOS (Texas Instruments Real Time Operating System)



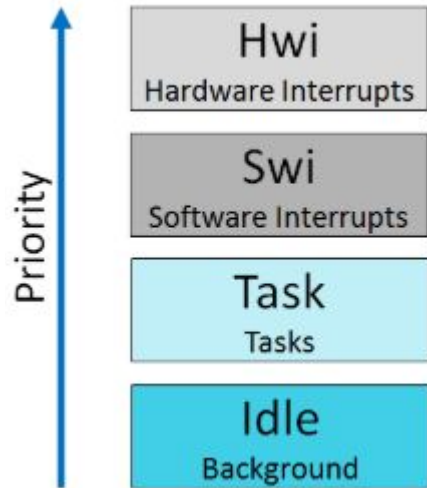
**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes

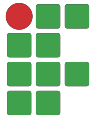


TI - RTOS (Texas Instruments Real Time Operating System)

- TI RTOS está incluído no SDK.



- Hardware Interrupts (Hwi):** Executa até o fim, não bloqueantes. Pode ser “preemptadas” por outra Hwi de maior prioridade. Todas Hwi compartilham a mesma pilha (System Stack)..
- Software Interrupts (Swi):** Similar a Hwi, mas são disparadas por software e compartilham a mesma pilha..
- Tasks:** Uma tarefa é uma thread do SO. Cada uma possui sua própria pilha (Que mantém o seu estado).
- Idle:** Trata-se de uma tarefa especial. Executa na prioridade mínima (0). Dispositivos LP podem ser colocados em modo de economia.



**INSTITUTO
FEDERAL**
Fluminense

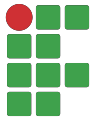
Polo de Inovação
Campos dos Goytacazes



TI - RTOS (Texas Instruments Real Time Operating System)

Conceitos importantes de RTOS:

- **Scheduler:** Preemptive scheduler that guarantees the highest priority thread is running.
- **Communication Mechanism:** Semaphores, Message Queues, Queues, etc.
- **Critical Region Mechanisms:** Mutexes, Gates, Locks, etc.
- **Timing Services:** Clocks, Timers, etc.
- **Power Management:** For low power devices, power management is generally part of the RTOS since it knows the state of the device.
- **Memory Management:** Variable-size heaps, fixed-size heaps, etc.
- **Peripheral Drivers:** UART, SPI, I2C, etc.
- **Protocol stacks:** BLE, WiFi, etc.
- **File System:** FatFs, etc.
- **Device Management:** Exception Handling, Boot, etc.



**INSTITUTO
FEDERAL**
Fluminense

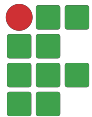
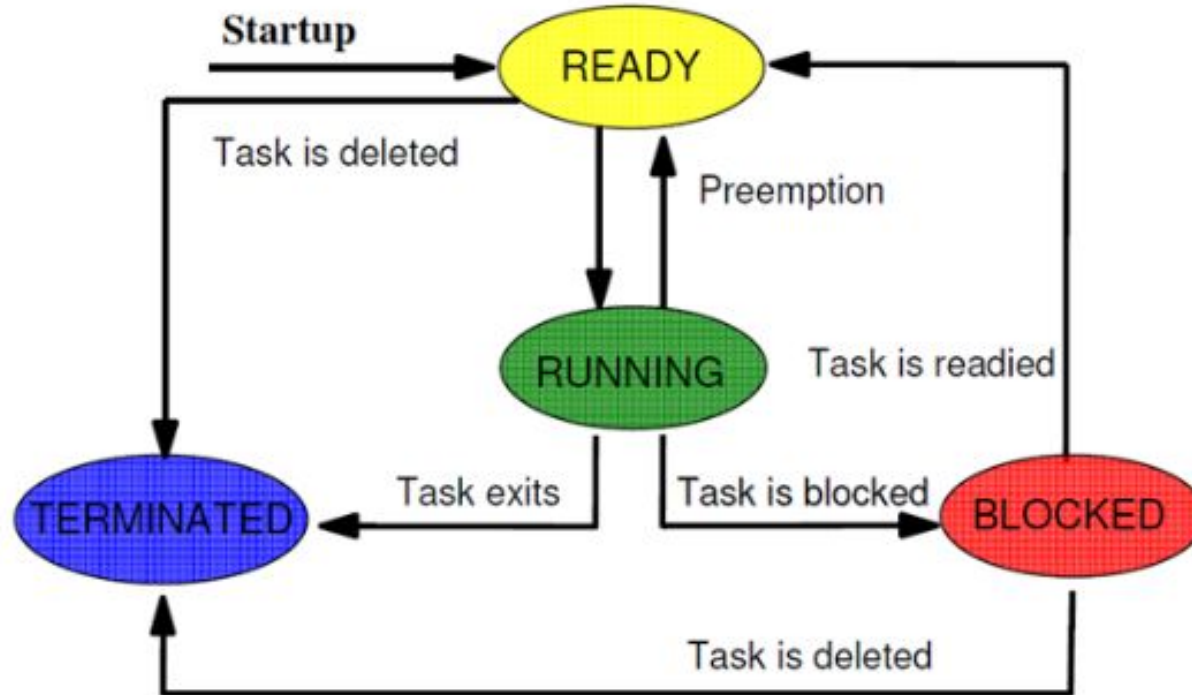
Polo de Inovação

Campos dos Goytacazes



TI - RTOS (Texas Instruments Real Time Operating System)

- Escalonador Preemptivo

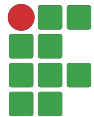
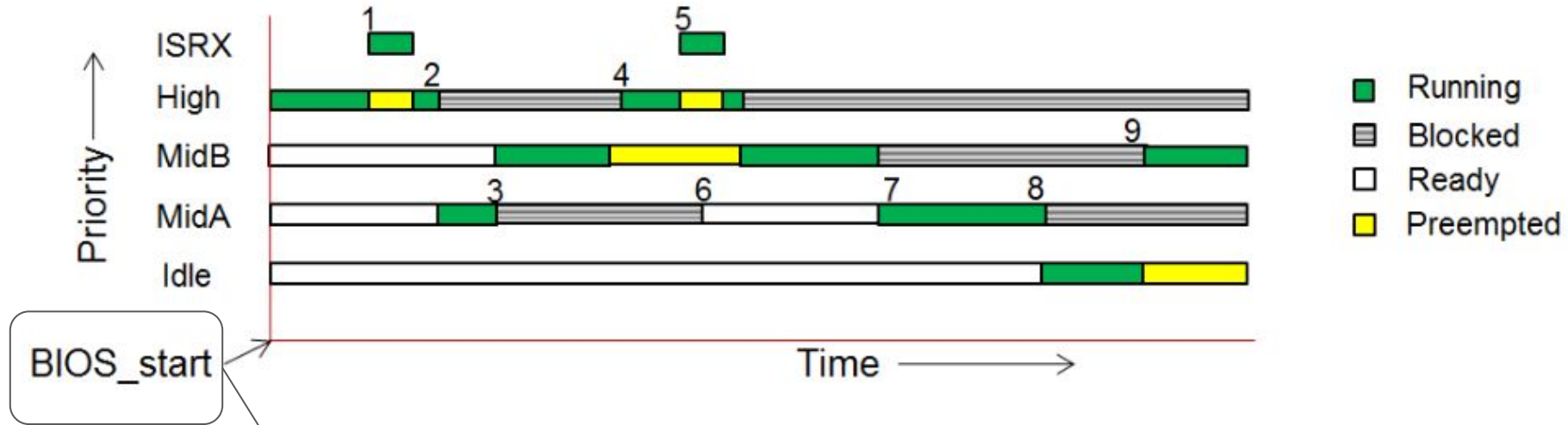


**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



TI - RTOS (Texas Instruments Real Time Operating System)



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes

Inicializa o
Escalonador:
função
chamada no
main().



TI - RTOS (Texas Instruments Real Time Operating System)

- Semáforos, Message Queues e Mutex

```
typedef struct MyMsg {
    Queue_Elem elem;
    int cmd;
} MyMsg;

Void senderTask(UArg arg0, UArg arg1)
{
    int i;
    MyMsg *msg;

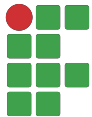
    for (i = 0; i < 5; i++) {
        msg = (MyMsg *)malloc(sizeof(MyMsg));
        msg->cmd = i;
        Display_printf(display, 0, 0, "Sending cmd = %d", msg->cmd);
        Queue_put(queueHandle, (Queue_Elem *)msg);
        Semaphore_post(semHandle);
    }
}

Void receiverTask(UArg arg0, UArg arg1)
{
    MyMsg *msg;

    for (;;) {
        Semaphore_pend(semHandle, BIOS_WAIT_FOREVER);
        msg = (MyMsg *)Queue_get(queueHandle);
        Display_printf(display, 0, 0, "Received cmd = %d", msg->cmd);
        free(msg);
    }
}
```

```
/* senderTask has priority 1 */
/* readerTask has priority 2 */
Sending cmd = 0
Received cmd = 0
Sending cmd = 1
Received cmd = 1
Sending cmd = 2
Received cmd = 2
Sending cmd = 3
Received cmd = 3
Sending cmd = 4
Received cmd = 4
```

```
/* senderTask has priority 2 */
/* readerTask has priority 1 */
Sending cmd = 0
Sending cmd = 1
Sending cmd = 2
Sending cmd = 3
Sending cmd = 4
Received cmd = 0
Received cmd = 1
Received cmd = 2
Received cmd = 3
Received cmd = 4
```



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



TI - RTOS (Texas Instruments Real Time Operating System)

- Semáforos, Message Queues e Mutex

Sincronização

```
typedef struct MyMsg {
    Queue_Elem elem;
    int cmd;
} MyMsg;

Void senderTask(UArg arg0, UArg arg1)
{
    int i;
    MyMsg *msg;

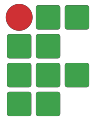
    for (i = 0; i < 5; i++) {
        msg = (MyMsg *)malloc(sizeof(MyMsg));
        msg->cmd = i;
        Display_printf(display, 0, 0, "Sending cmd = %d", msg->cmd);
        Queue_put(queueHandle, (Queue_Elem *)msg);
        Semaphore_post(semHandle);
    }
}

Void receiverTask(UArg arg0, UArg arg1)
{
    MyMsg *msg;

    for (;;) {
        Semaphore_pend(semHandle, BIOS_WAIT_FOREVER);
        msg = (MyMsg *)Queue_get(queueHandle);
        Display_printf(display, 0, 0, "Received cmd = %d", msg->cmd);
        free(msg);
    }
}
```

```
/* senderTask has priority 1 */
/* readerTask has priority 2 */
Sending cmd = 0
Received cmd = 0
Sending cmd = 1
Received cmd = 1
Sending cmd = 2
Received cmd = 2
Sending cmd = 3
Received cmd = 3
Sending cmd = 4
Received cmd = 4
```

```
/* senderTask has priority 2 */
/* readerTask has priority 1 */
Sending cmd = 0
Sending cmd = 1
Sending cmd = 2
Sending cmd = 3
Sending cmd = 4
Received cmd = 0
Received cmd = 1
Received cmd = 2
Received cmd = 3
Received cmd = 4
```



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes

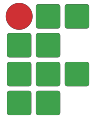


TI - RTOS (Texas Instruments Real Time Operating System)

- Semáforos, Message Queues e Mutex

Mutex

<u>writer1 (lower priority)</u>	<u>writer2 (higher priority)</u>	<u>reader</u>
<code>myGlobal.cmd = WRITE_BUFFER;</code> <code>myGlobal.srcBuffer = bufferA;</code>	<code>// blocked</code>	<code>//blocked sleeping</code>
 <code>// timer expires and unblocks writer2</code>		
	<code>myGlobal.cmd = READ_BUFFER;</code> <code>myGlobal.srcBuffer = bufferC;</code> <code>myGlobal.dstBuffer = bufferD;</code> <code>// later blocks on a semaphore</code>	
 <code>// starts to run again...</code> <code>myGlobal.dstBuffer = BufferB;</code>		
		 <code>//processes myGlobal but</code> <code>//it's garbage</code>



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



TI - RTOS (Texas Instruments Real Time Operating System)

- Semáforos, Message Queues e Mutex

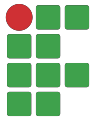
Mutex

```
Void writer1(UArg arg0, UArg arg1)
{
    ...

    /* Get access to resource */
    Semaphore_pend(semHandle, BIOS_WAIT_FOREVER);

    myGlobal.cmd      = WRITE_BUFFER;
    myGlobal.srcBuffer = bufferInFlash;
    myGlobal.dstBuffer = bufferInRAM;

    /* Release access to resource */
    Semaphore_post(semHandle);
}
```



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



TI - RTOS (Como usar?)

- O Kernel é gerado durante a compilação da aplicação a partir de um arquivo .cgf (JavaScript), pode ser editado em formato de texto ou graficamente através do CCS.
- O SDK fornece o Kernel nos diretórios kernel/tirtos/builds/BOARD/release and debug*. Assim o Projeto Kernel pode ser importado pelo CCS: **Project** → **Import CCS Projects...**
- O Kernel é automaticamente importado quando um exemplo TI-RTOS é importado. Uma cópia do projeto Kernel é feito no Workspace. A aplicação faz uma referência ao projeto Kernel. (É possível incluir em uma só aplicação, mas é **melhor** referenciar. Isso permite o uso no free RTOS e TI RTOS.



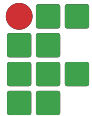
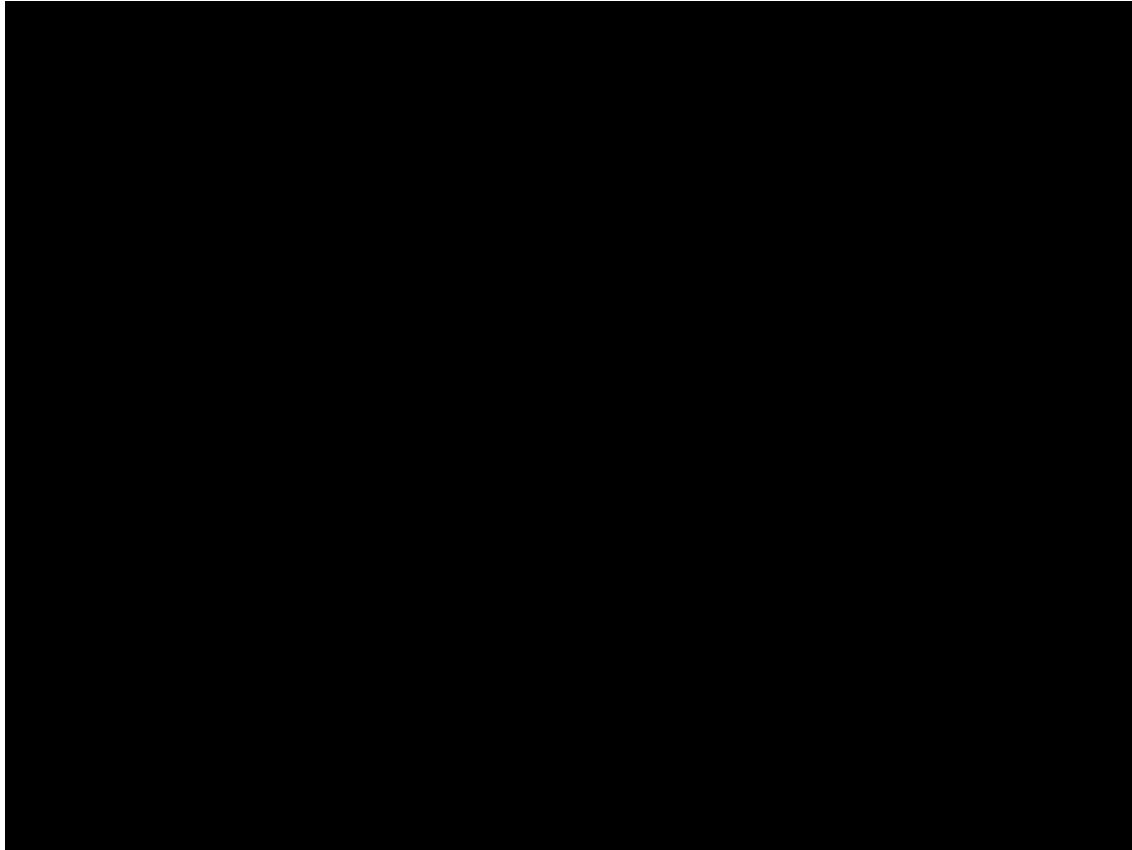
**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes

*Habilita algumas funcionalidades de Debug.



TI - RTOS (Importando o projeto Kernel)

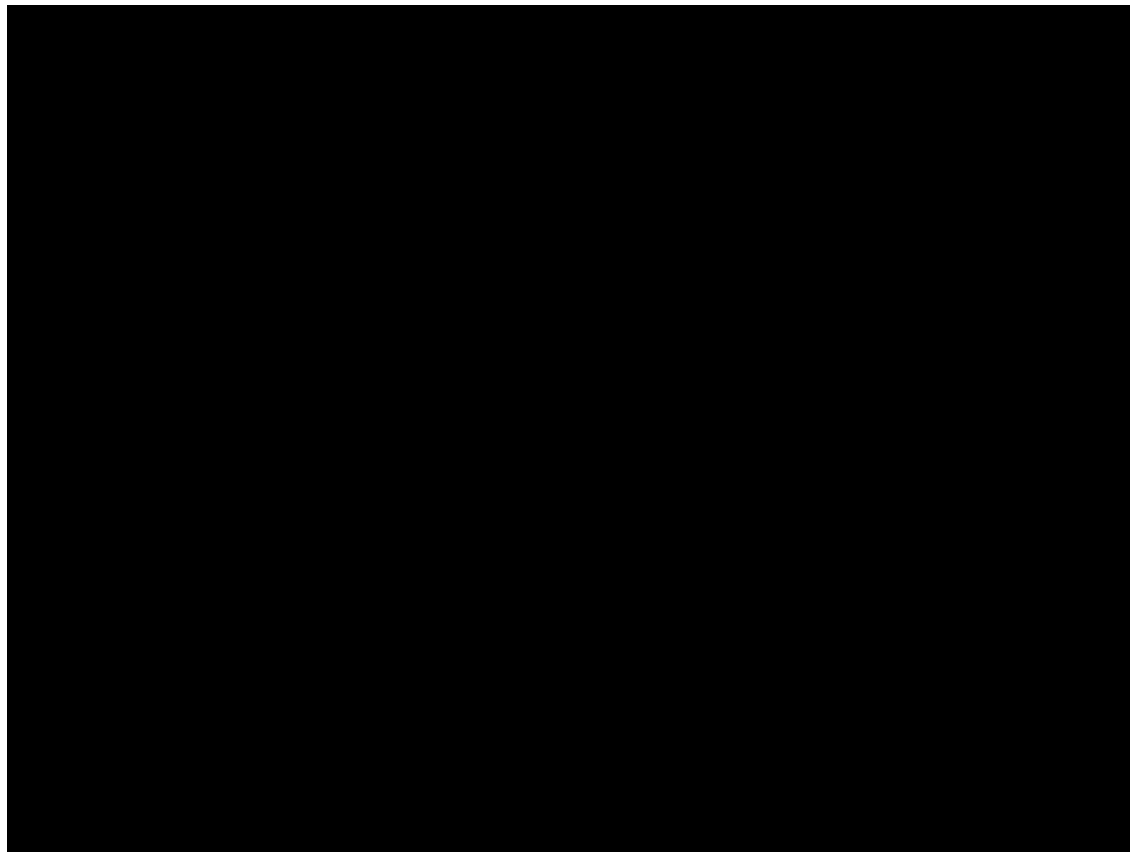


**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



TI - RTOS (Importando o projeto “empty” e criando dependência com o Kernel)



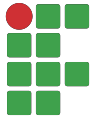
**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

- Processador RISC 16-bit
- O SCS pode acessar periféricos de seu domínio (AUX).
- CSC - Link de instalação:
<http://www.ti.com/tool/SENSOR-CONTROLLER-STUDIO>
- Dedicado a sensoriamento e monitoramento



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes

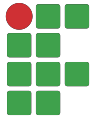


Sensor Controller Studio (SCS)

Laboratório 1:

O que vamos fazer nesse laboratório?

Vamos fazer a leitura de um pino analógico (usando um ADC) periodicamente através do SCS e acender ou apagar LEDs dependendo da saída do ADC e de thresholds definidos pela aplicação central.



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes

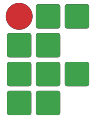


Sensor Controller Studio (SCS)

Laboratório 1:

Tarefa 1: Usar o SCS separadamente.

Tarefa 2: Integrar com uma aplicação central.



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes

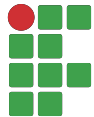
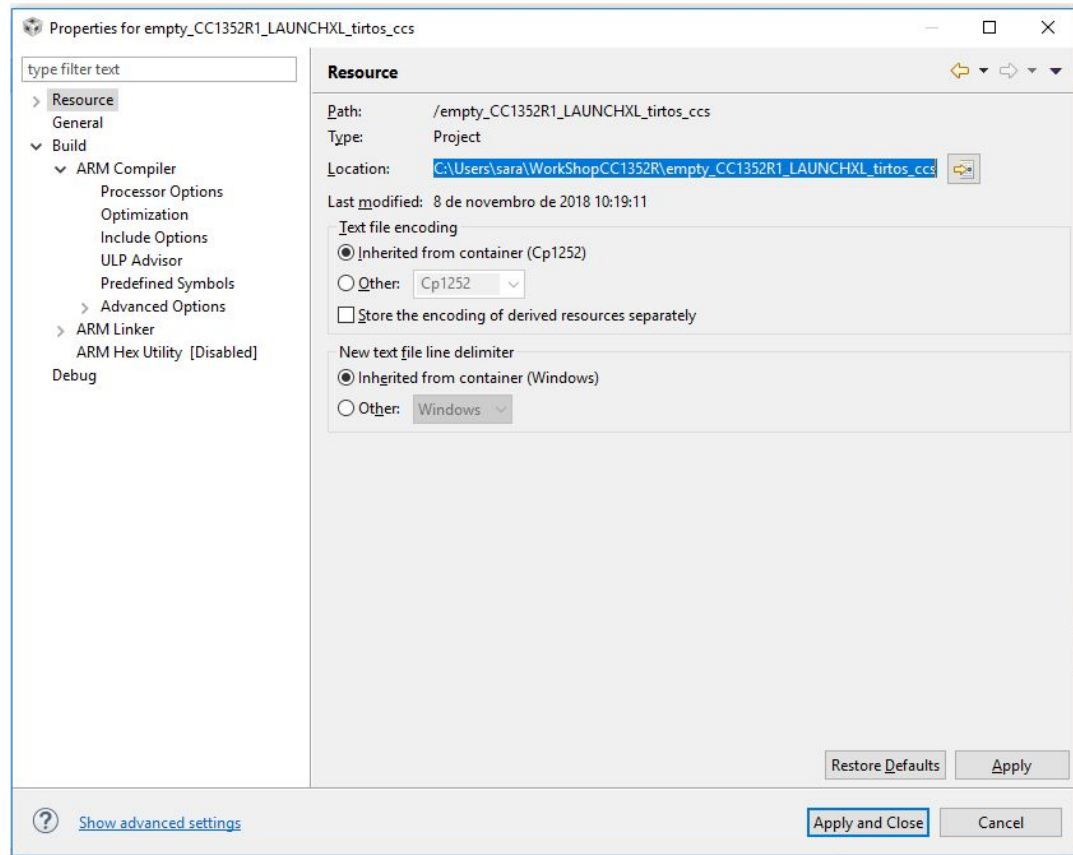


Sensor Controller Studio (SCS)

TAREFA #1

Encontre o caminho do seu projeto e copie.

1. Selecione o projeto
2. Clique no botão direito.
3. Properties



**INSTITUTO
FEDERAL**
Fluminense

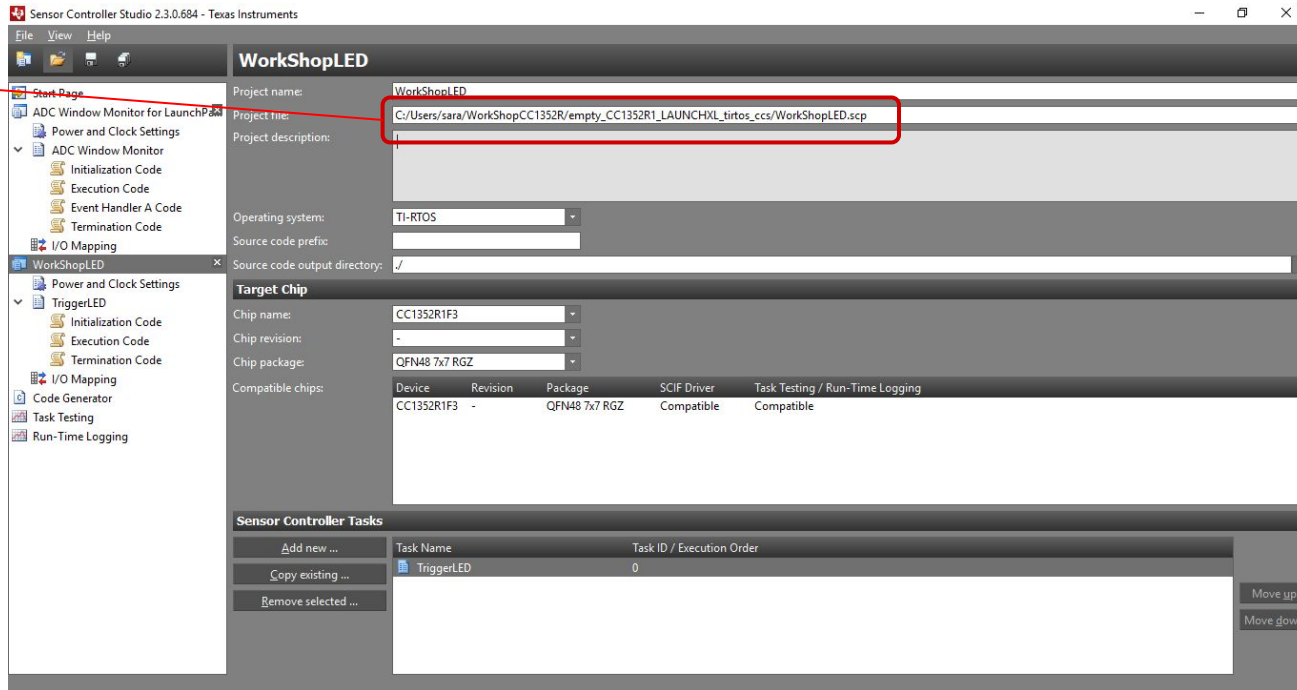
Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Crie um novo projeto do SCS, dê nome ao projeto, configure como na imagem abaixo e crie uma nova task.

Endereço do Projeto “empty” criado



- 1.File
- 2.Save as
- 3.Coloque o endereço.
- 4.Preencha com o nome do projeto e salve.



**INSTITUTO
FEDERAL**
Fluminense

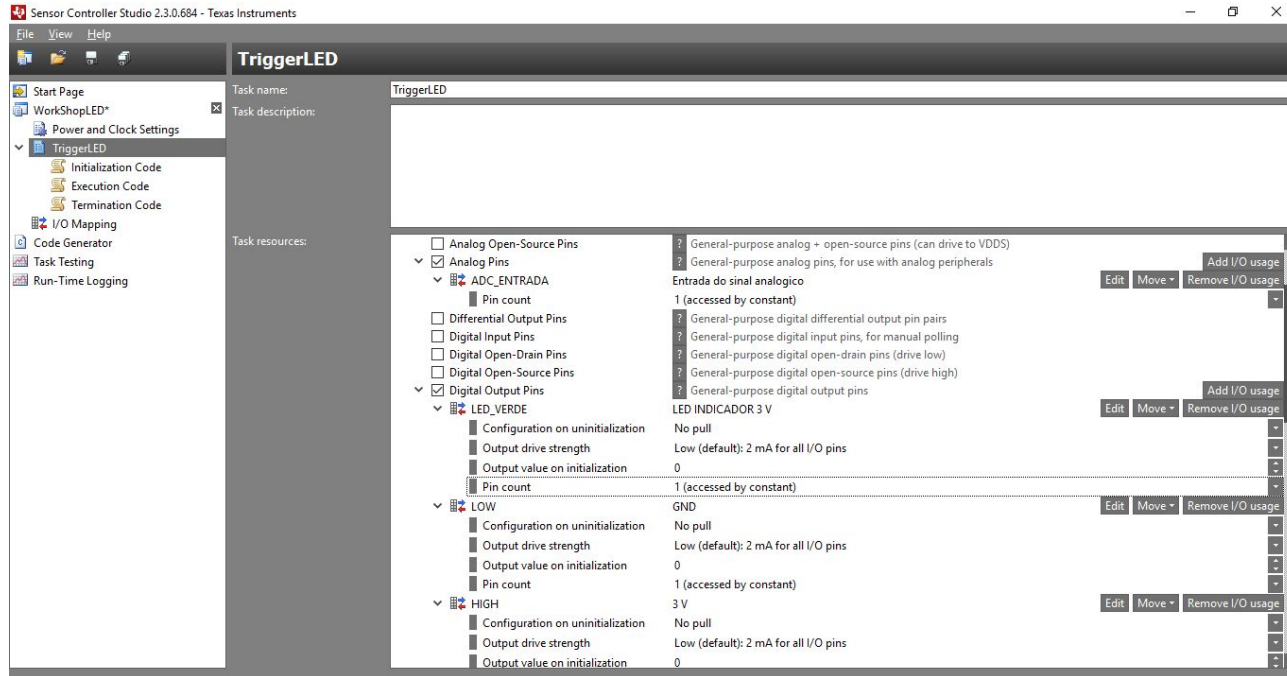
Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Selecione os recursos que deseja utilizar:

- 1 AI
- 3 DO
- System CPU Alert
- RTC Based Execution Scheduling
- ADC



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Realize o
mapeamento no
painel
I/O Mapping

Sensor Controller Studio 2.3.0.684 - Texas Instruments

File View I/O Mapping Help

I/O Mapping

Select board: None

Start Page
WorkShopLED*
Power and Clock Settings
TriggerLED
Initialization Code
Execution Code
Termination Code
I/O Mapping
Code Generator
Task Testing
Run-Time Logging

TriggerLED

A: Entrada do sinal analogico

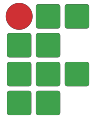
O: 3 V

O: GND

O: LED INDICADOR 3 V

CC1352R1F3, Revision -, Package QFN48 7x7 RGG

I/O Name	AUX Mapping	Pin Number	Board Mapping	Currently Mapped I/O Function(s)
DIO3	15	8	-	-
DIO4	14	9	-	-
DIO5	13	10	-	-
DIO6	12	11	-	-
DIO7	11	12	-	O: LED INDICADOR 3 V
DIO8	10	14	-	-
DIO9	9	15	-	-
DIO10	8	16	-	-
DIO11	7	17	-	-
DIO12	6	18	-	-



INSTITUTO
FEDERAL
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Configure o período RTC para a tarefa ser executada. O período RTC determina o intervalo para tarefa ser executada.

Tarefa -> File > Preferences

Task testing

Log and display initial data structure values	Yes
Maximum task code execution time before triggering debug mode	1 seconds
Minimum task iteration interval	250 milliseconds

OBS: Essa configuração é para testar no SCS.

Isso **não** configura o período do RTC quando o driver é integrado na aplicação principal.

Isso é feito no código da aplicação com as chamadas:

`scifStartRtcTicks()` e `scifStartRtcTicksNow()`.

`fwScheduleTask(N)` agenda a próxima tarefa para N RTC ciclos.

Pergunta:

Se A tarefa do SC é agendada com `fwScheduleTask(3)`; e o intervalo na qual a tarefa está sendo executada é de 1500 ms, qual é a frequência do RTC?

2 Hz

3 Hz

5 Hz

6 Hz



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



Sensor Controller Studio (SCS)

Dados

Estrutura de Dados (4 tipos, são armazenadas na RAM do SC - 16 bits):

Data structure	Intended use
cfg	Configuration of SC Task
input	Input data for SC Task
output	Output data from SC Task
state	Internal state of SC Task

Área de constantes

Estruturas de dados e valores iniciais

Procedimentos Disponíveis:
Clique duplo no procedimento para saber como usá-lo.

The screenshot displays the SCS interface with three main panels. The 'Constants' panel lists various constants with their values, such as ADC_TRIGGER_MANUAL (0x003f) and AUXIO_A_ADC_ENTRADA (20). The 'Data structures' panel shows the initial values for the 'cfg', 'output', and 'state' structures. The 'Available procedures' panel lists various ADC-related functions like adcDisable, adcEnableAsync, and adcGetFifoStatus.

Constants	Value
ADC_TRIGGER_MANUAL	0x003f
AUXIO_A_ADC_ENTRADA	20
AUXIO_O_HIGH	21
AUXIO_O_LED_VERDE	11
AUXIO_O_LOW	19
BV_ADC_FIFO_ALMOST_FULL	0x0002
BV_ADC_FIFO_EMPTY	0x0001
BV_ADC_FIFO_FULL	0x0004
BV_ADC_FIFO_OVERFLOW	0x0010
BV_ADC_FIFO_UNDERFLOW	0x0008

Data structures	Initial
cfg	
threshold	0
output	
ValorADC	0
state	
FlagHIGH	0

Available procedures
adcDisable
adcDisableInputScaling
adcEnableAsync
adcEnableSync
adcFlushFifo
adcGenManualTrigger
adcGetFifoStatus



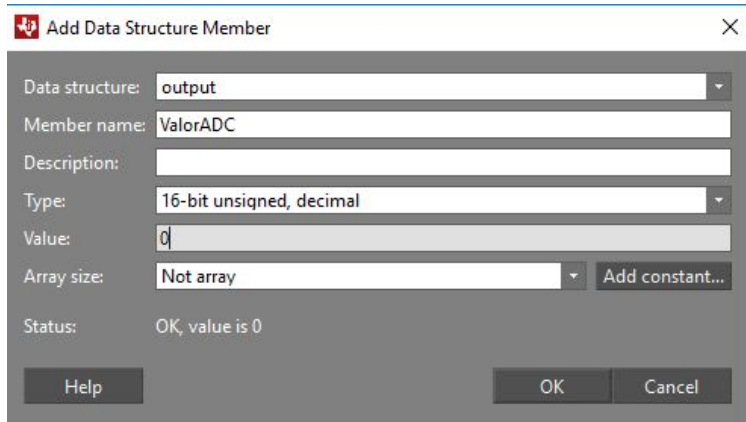
**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

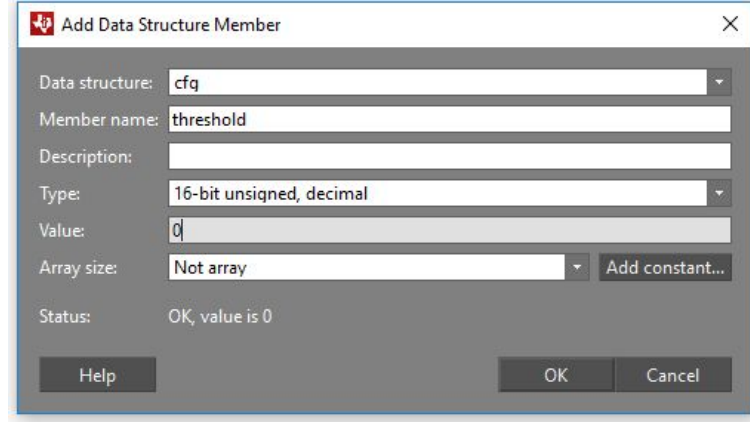
Adicione as estruturas de dados: A estrutura é a mesma para os 3 códigos: Inicialização, Execução e Finalização. Ou seja é visualizável dentro dos 3 escopos.



The dialog box 'Add Data Structure Member' is shown with the following fields:

- Data structure: output
- Member name: ValorADC
- Description: (empty)
- Type: 16-bit unsigned, decimal
- Value: 0
- Array size: Not array
- Status: OK, value is 0

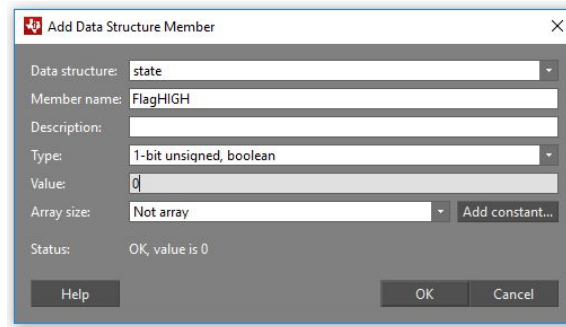
Buttons: Help, OK, Cancel



The dialog box 'Add Data Structure Member' is shown with the following fields:

- Data structure: cfq
- Member name: threshold
- Description: (empty)
- Type: 16-bit unsigned, decimal
- Value: 0
- Array size: Not array
- Status: OK, value is 0

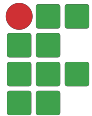
Buttons: Help, OK, Cancel



The dialog box 'Add Data Structure Member' is shown with the following fields:

- Data structure: state
- Member name: FlagHIGH
- Description: (empty)
- Type: 1-bit unsigned, boolean
- Value: 0
- Array size: Not array
- Status: OK, value is 0

Buttons: Help, OK, Cancel



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



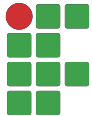
Sensor Controller Studio (SCS)

Faça o código da Inicialização:

Os Códigos apresentados aqui podem ser copiados do meu repositório no GitHub:
<https://github.com/saramonteiro/WorkShopC1352R>

TriggerLED - Initialization Code

```
1 // Pino DI028 - AUXIO_O_HIGH com Vcc
2 gpioSetOutput(AUXIO_O_HIGH);
3
4 // Pino DI030 - AUXIO_O_LOW com GND
5 gpioClearOutput(AUXIO_O_LOW);
6
7 // Selecione a entrada na qual o ADC vai atuar para converter
8 adcSelectGpioInput(AUXIO_A_ADC_ENTRADA);
9
10 // Agende a primeira execucao
11 fwScheduleTask(1);|
```



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Faça o código de execução:

TriggerLED - Execution Code

```
1 // Ative o ADC
2 adcEnableSync(ADC_REF_FIXED, ADC_SAMPLE_TIME_2P7_US, ADC_TRIGGER_MANUAL);
3
4 // Gera um trigger manual para obter a leitura e lê da memória que o ADC armazenou
5 adcGenManualTrigger();
6 adcReadFifo(output.ValorADC);
7
8 // Desabilite o ADC
9 adcDisable();
10
11 //Inicializa o estado anterior
12 U16 EstadoAnterior = state.FlagHIGH;
13
14 //Se o valor do ADC for maior que o threshold
15 if (output.ValorADC > cfg.threshold){
16     state.FlagHIGH = 1; //Atualiza a flag indicadora
17     gpioClearOutput(AUXIO_O_LED_VERDE); //Apaga o LED verde
18 } else{
19     state.FlagHIGH = 0; //Atualiza a flag indicadora |
20     gpioSetOutput(AUXIO_O_LED_VERDE); // Acende o LED
21 }
22
23 //Se o estado anterior for diferente do estado atual
24 if (EstadoAnterior != state.FlagHIGH){
25     //Sinalize ao processador da aplicação central através de uma interrupção
26     fwGenAlertInterrupt();
27 }
28
29 //Agende a próxima execução para o período de 1 "tick" de RTC
30 fwScheduleTask(1);
```



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes

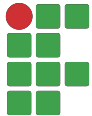


Sensor Controller Studio (SCS)

Código de Finalização:

O código de finalização é executado uma única vez:

- quando a tarefa do SC é finalizada pela aplicação principal ou quando não há o agendamento (*shcheduling*) (Linha 30 do código de execução).



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes

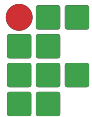


Sensor Controller Studio (SCS)

Dicas importantes para programar:

1. Utilize o “*Auto-Complete*” nos argumentos das funções com Ctrl + Espaço.
2. Clique Duplo nos procedimentos para saber como usá-los.
3. Muita atenção à sintaxe. Pressione F2 para documentação da linguagem.
4. Cheque se o código está ok no painel “Task Testing”. Lá indicará erros de sintaxe.

Event	Time / Line	Description
■ Selecting project	17:42:26.074	ADC Level Trigger
■ Validating project	17:42:26.074	ADC Level Trigger
■ Reading source template files	17:42:26.074	ADC Level Trigger
■ Validating task	17:42:26.104	adc level trigger
■ Processing task	17:42:26.104	adc level trigger
■ Compiling task code	17:42:26.104	Initialization Code
■ Compiling task code	17:42:26.114	Execution Code
■ Execution Code	12	Invalid syntax in expression



**INSTITUTO
FEDERAL**
Fluminense

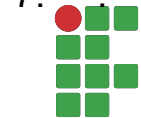
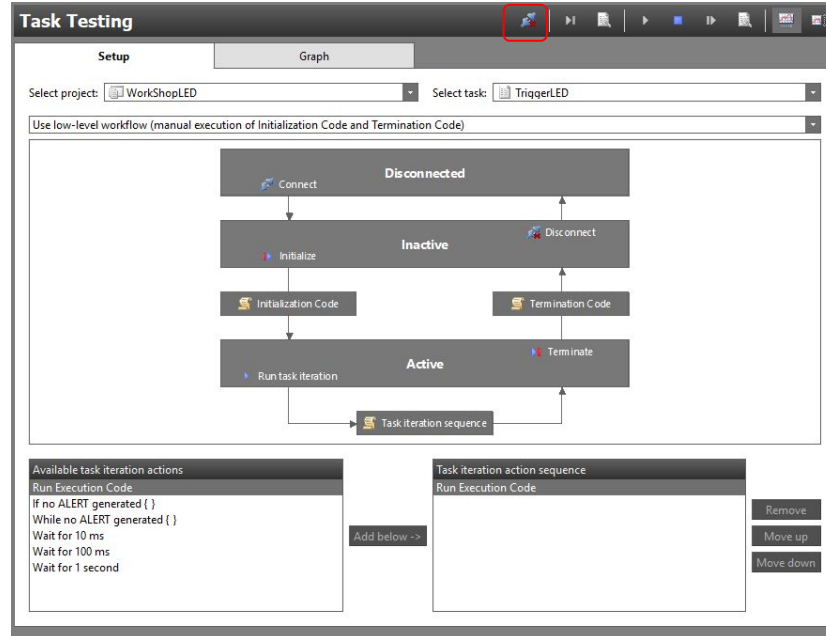
Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Execute a tarefa no SCS:

1. Vá no painel “Tak Testing”, se não houver erro nenhum no código você deve ir para essa janela:
2. Configure como na imagem e adicione “Run Execution Code” a “Task Iteration Sequence”.
3. Conecte a launchpad.
4. Clique em Conectar. (F12)
5. Clique em Run Initialization Code (F6)
6. Na aba Graph, no lado direito mude o valor de “threshold” para “400”.
7. Clique em Run Execution Code Continuously (F5).



**INSTITUTO
FEDERAL**
Fluminense

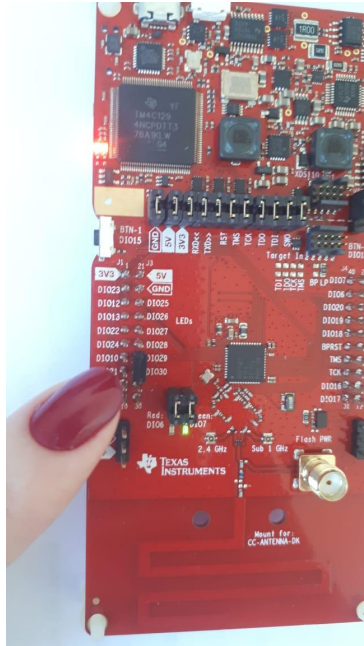
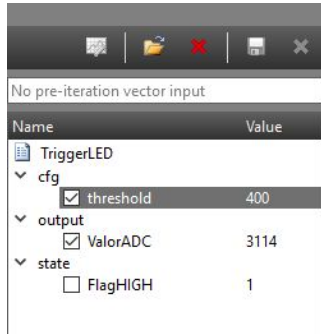
Polo de Inovação
Campos dos Goytacazes



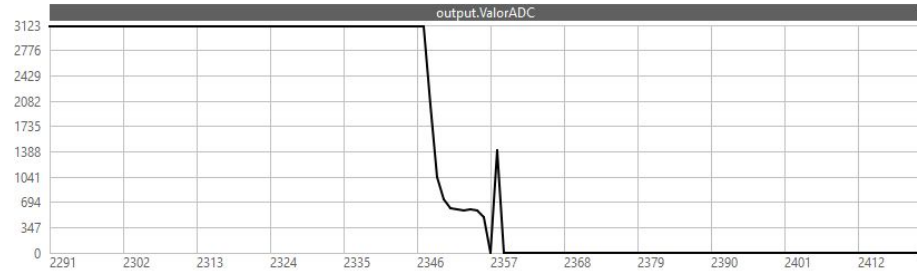
Sensor Controller Studio (SCS)

Observe a execução.

1. Selecione as variáveis que deseja observar no gráfico.



2. Com o Jump, alterne DIO29 para DIO28 e DIO30. Observe o LED verde (DIO7), o gráfico e a variável do estado.



DIO30	GND	LED ACESO
DIO28	VCC	LED APAGADO



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes

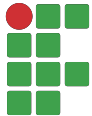
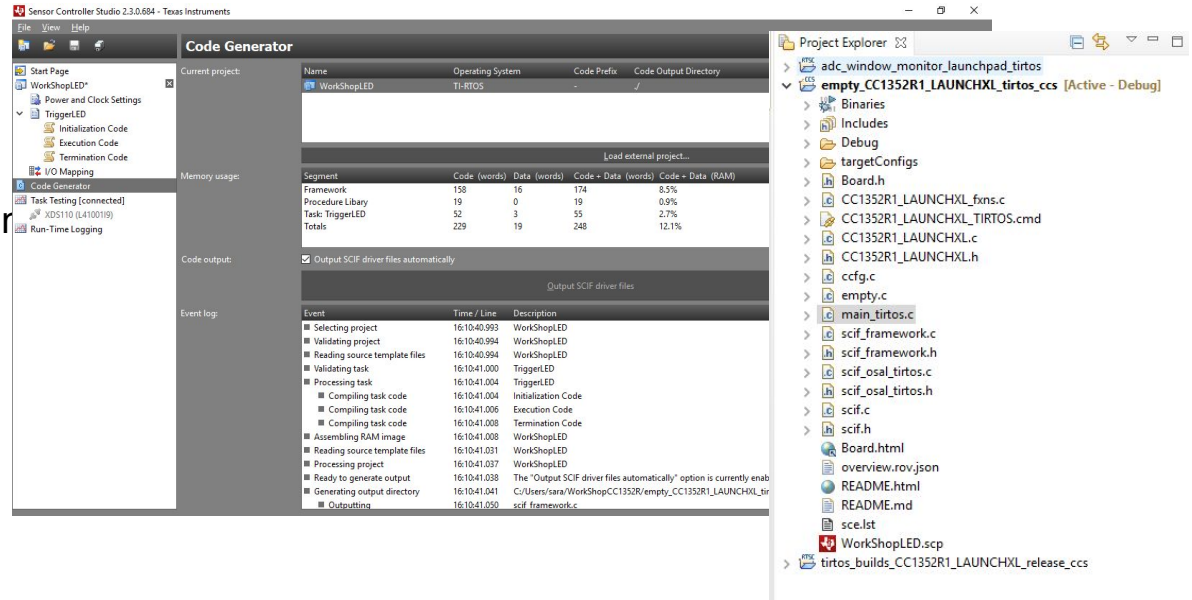


Sensor Controller Studio (SCS)

TAREFA #2

Integre com a aplicação principal.

1. Certifique-se que o Code Composer está aberto.
2. Vá no painel “Code Generator”,
3. Clique para gerar automaticamente ou clique no botão para gerar.
4. Cheque no projeto do CCS os drivers e framework.
5. Compile.



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Faça alterações no Projeto:

1. Modifique o nome da thread atual de “mainThread” para ThresholdScThread nos arquivos *main_tirtos* e no arquivo *empty.c*.
Dica: Dê um search.
2. Apague tudo que está no while(1) de *empty.c*.
3. Como o LED **VERDE** é controlado pela SC, o driver GPIO não deve inicializar nem configurá-lo. Para evitar que isso ocorra, acesse o arquivo *CC1352R1_LAUNCHXL.c/.h* e comente as seguintes linhas:

```
504 */
505 GPIO_PinConfig gpioPinConfigs[] = {
506     /* Input pins */
507     GPIOCC26XX_DIO_15 | GPIO_DO_NOT_CONFIG, /* Button 0 */
508     GPIOCC26XX_DIO_14 | GPIO_DO_NOT_CONFIG, /* Button 1 */
509
510     GPIOCC26XX_DIO_15 | GPIO_DO_NOT_CONFIG, /* CC1352R1_LAUNCHXL_SP
511     GPIOCC26XX_DIO_21 | GPIO_DO_NOT_CONFIG, /* CC1352R1_LAUNCHXL_SP
512
513     /* Output pins */
514     //GPIOCC26XX_DIO_07 | GPIO_DO_NOT_CONFIG, /* Green LED */
515     GPIOCC26XX_DIO_06 | GPIO_DO_NOT_CONFIG, /* Red LED */
516
517     /* SPI Flash CSN */
518     GPIOCC26XX_DIO_20 | GPIO_DO_NOT_CONFIG,
519
520     /* SD CS */
521     GPIOCC26XX_DIO_21 | GPIO_DO_NOT_CONFIG,
522
523     /* Sharp Display - GPIO configurations will be done in the Displ
524     GPIOCC26XX_DIO_24 | GPIO_DO_NOT_CONFIG, /* SPI chip select */
525     GPIOCC26XX_DIO_22 | GPIO_DO_NOT_CONFIG, /* LCD power control */
```

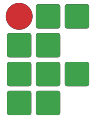
```
729 const PIN_Config BoardGpioInitTable[] = {
730
731     CC1352R1_LAUNCHXL_PIN_RLED | PIN_GPIO_O
732     //CC1352R1_LAUNCHXL_PIN_GLED | PIN_GPIO
733     CC1352R1_LAUNCHXL_PIN_BTN1 | PIN_INPUT_
734     CC1352R1_LAUNCHXL_PIN_BTN2 | PIN_INPUT_
735     CC1352R1_LAUNCHXL_PIN_FLASH_CS | PIN_GP
736
737     typedef enum CC1352R1_LAUNCHXL_GPIOName {
738         CC1352R1_LAUNCHXL_GPIO_S1 = 0,
739         CC1352R1_LAUNCHXL_GPIO_S2,
740         CC1352R1_LAUNCHXL_SPI_MASTER_READY,
741         CC1352R1_LAUNCHXL_SPI_SLAVE_READY,
742         //CC1352R1_LAUNCHXL_GPIO_LED_GREEN,
743         CC1352R1_LAUNCHXL_GPIO_LED_RED
```

Sensor Controller Studio (SCS)

O SCS interage com a aplicação principal através de duas interrupções de *callback*:

- Control READY
- Task ALERT

Basicamente o sinal de interrupção da CR é gerado por funções de controle da tarefa do SC: Ex: `scifStartTasksNbl()`, `scifStopTasksNbl()`. enquanto a TA é sinalizada quando o SC chama funções como `fwGenAlertInterrupt()` indicando algo como um evento.



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



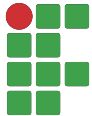
Sensor Controller Studio (SCS)

Como é feito o acesso às estruturas de dados do SC:

scifTaskData. + Nome da Task do SC. + tipo da estrutura. + nome do membro

Exemplo:

scifTaskData.TriggerLED.output.ValorADC



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



Sensor Controller Studio (scs)

Inicialize e configure o Driver SC.

1. No arquivo *empty.c*, adicione ao início:

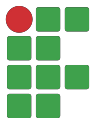
```
50 /* Drivers and macros from SC*/
51 #include "scif.h"
52 #define BV(x)    (1 << (x))
```

2. Crie as funções de callback. O código de inicialização do driver SC usa as duas funções de callback para tratar os dois sinais de interrupção gerados.

```
57 /* Callback Functions */
58 void scCtrlReadyCallback(void)
59 {
60
61 } // scCtrlReadyCallback
62
63 void scTaskAlertCallback(void)
64 {
65
66 } // scTaskAlertCallback
67
68 /*
```

3. Inicialize o driver finalmente e registre as funções de callback..

```
71 void *ThresholdScThread(void *arg0)
72 {
73     // Inicializa o Sensor Controller
74     // Inicializa a OSAL (Operating System Abstraction Layer) do framework scif
75     scifOsInit();
76     // Registra as duas funções de callback para os sinais de Interrupção Control READY e Task ALERT do SC
77     scifOsRegisterCtrlReadyCallback(scCtrlReadyCallback);
78     scifOsRegisterTaskAlertCallback(scTaskAlertCallback);
79     // Inicializa o SC com o driver gerado a partir do SC
80     scifInit(&scifDriverSetup);
81
82     // Configura o intervalo para a tarefa do Sensor Controller para 1 segundo
83     // Bits 31:16 = segundos
84     // Bits 15:0 = 1/65536 de um segundo
85     uint32_t rtc_Hz = 1; // 1Hz RTC
86     scifStartRtcTicksNow(0x00010000 / rtc_Hz);
87
88     // Configura a tarefa do Sensor Controller: Inicializa um valor de threshold (800)
89     scifTaskData.triggerLed.cfg.threshold = 600;
90
91     // Inicializa a tarefa do Sensor Controller
92     // As funções scif lidam com IDs de tarefas do SC e acessam um vetor de bit como argumento
93     // O ID está definido em scif.h
94     scifStartTasksNbl(BV(SCIF_TRIGGER_LED_TASK_ID));
95
96     ...
97 }
```



**INSTITUTO
FEDERAL**
Fluminense

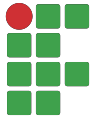
Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Vamos agora desenvolver o código que processa uma interrupção do Sensor Controller:

1. Vamos aguardar um sinal de alerta de interrupção (task ALERT).
2. Limpar flag de interrupção.
3. Processar a tarefa SC.
4. Confirmar o evento ao scif framework.



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



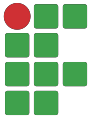
Sensor Controller Studio (SCS)

Crie a tarefa de processamento dos dados vindo do Sensor Controller:

1. Vamos criar uma função para fazer o processamento:

```
68 /* Funcoes para processar o sinal de alerta do SC */
69 void processTaskAlert(void)
70 {
71     // Limpa a flag de interrupcao fonte da interrupcao
72     scifClearAlertIntSource();
73
74     // Atribui o valor de FlagHIGH do SC para a variavel high
75     uint8_t high = scifTaskData.triggerLed.state.FlagHIGH;
76     // Atualiza o estado do led vermelho de acordo com a flag
77     GPIO_write(Board_GPIO_RLED, high);
78
79     // Confirma o evento ao framework
80     scifAckAlertEvents();
81 }
```

Mas e agora como “ligar” esse processamento a interrupção gerada pelo SC?



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Solução 1 (HWI):

1. Simplesmente faça a chamada da função de processamento dentro da função de callback. O SC gera o sinal de alerta, a função de callback do hardware é chamada.

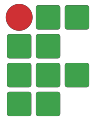
```
63 void scTaskAlertCallback(void)
64 {
65     processTaskAlert();
66 } // scTaskAlertCallback
```

2. Compile, faça o upload do código e execute. Teste novamente o jumper.

Solução 2 (SWI):

1. Inclua no início de empty.c

```
57 #include <ti/sysbios/knl/Swi.h>
58
59 // SWI Task Alert
60 Swi_Struct swiTaskAlert;
61 Swi_Handle hSwiTaskAlert;
62
63 // Function prototype //Serve para nao ter que colocar a funcao em cima
64 void processTaskAlert(void);
65
66 void swiTaskAlertFxn(UArg a0, UArg a1)
67 {
68     // Chame a funcao de processamento
69     processTaskAlert();
70 } // swiTaskAlertFxn
71
```



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

Solução 2 (SWI):

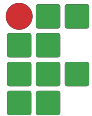
2. Inicialize o processo interrupção por software na thread.

4. Compile, faça o upload do código e execute. Teste novamente o jumper.

```
102 void *ThresholdScThread(void *arg0)
103 {
104     // Inicializacao da interrupcao de software
105     // Declara os parametros
106     Swi_Params swiParams;
107     // Inicializa com os parametros padroes
108     Swi_Params_init(&swiParams);
109     // Define Prioridade
110     swiParams.priority = 3;
111     // Constroi enviando os enderecios das estruturas e associando a funcao que deve ser chamada
112     Swi_construct(&swiTaskAlert, swiTaskAlertFxn, &swiParams, NULL);
113     hSwiTaskAlert = Swi_handle(&swiTaskAlert);
114 }
```

3. Sinalize a SWI dentro da função de callback.

```
93 void scTaskAlertCallback(void)
94 {
95     // Sinaliza um processo para uma interrupcao de software
96     Swi_post(hSwiTaskAlert);
97 } // scTaskAlertCallback
98
```



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



Sensor Controller Studio (SCS)

Solução 3 (Semáforo e Thread):

Nessa solução uma thread aguarda uma sinalização do hardware e utiliza semáforo para sincronizar.

1. Declare as variáveis utilizadas no início de *empty.c*.
2. Inicialize o Semáforo na thread.
3. Configure para haver a “postagem” no semáforo sempre que houver o alerta.

```
57 #include <ti/sysbios/kl/Semaphore.h>
58 #include <ti/sysbios/BIOS.h>
59
60 // Variaveis do Semaforo - struct e handler
61 Semaphore_Struct semMainLoop;
62 Semaphore_Handle hSemMainLoop;
```

```
85 void scTaskAlertCallback(void)
86 {
87     //Posta no Semaforo do main sempre que houver um alerta
88     Semaphore_post(hSemMainLoop);
89 } // scTaskAlertCallback
```

2. Inicialize o Semáforo na thread.

```
93 void *ThresholdScThread(void *arg0)
94 {
95     // Inicializacao e configuracao do Semaforo
96     //Inicializacao dos parametros padroes
97     Semaphore_Params semParams;
98     Semaphore_Params_init(&semParams);
99     // Construção da estrutura do semaforo com os parametros
100     Semaphore_construct(&semMainLoop, 0, &semParams);
101     // Armazena a estrutura no "tratador"
102     hSemMainLoop = Semaphore_handle(&semMainLoop)
```



INSTITUTO
FEDERAL
Fluminense

Polo de Inovação

Campos dos Goytacazes



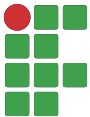
Sensor Controller Studio (SCS)

Solução 3 (Semáforo e Thread):

4. Faça o semáforo aguardar até que algo seja postado nele e libere a execução da tarefa de processamento.

```
144
145     while (1)
146     {
147         // Aguarda indefinidamente até que haja uma postagem no semáforo
148         Semaphore_pend(hSemMainLoop, BIOS_WAIT_FOREVER);
149
150         // Chame a função para executar o processamento
151         processTaskAlert();
152     }
153 }
154
```

5. Compile, faça o upload do código e execute. Teste novamente o jumper.



**INSTITUTO
FEDERAL**
Fluminense

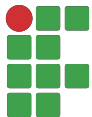
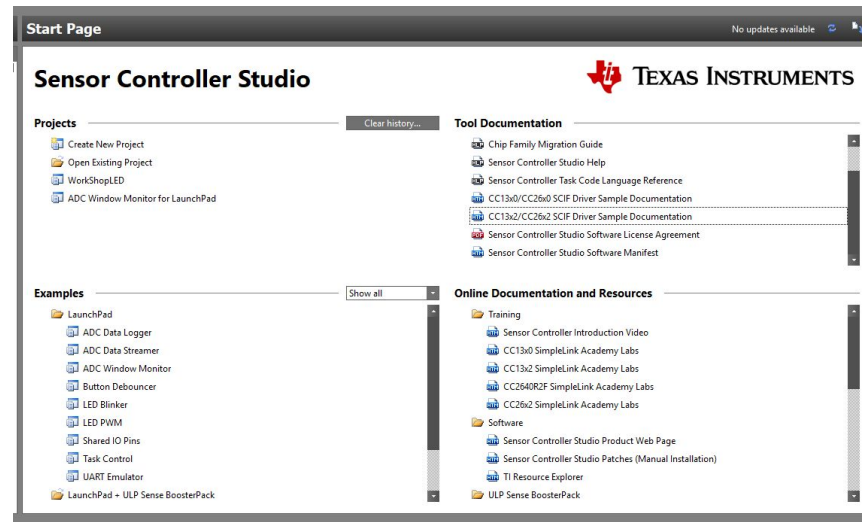
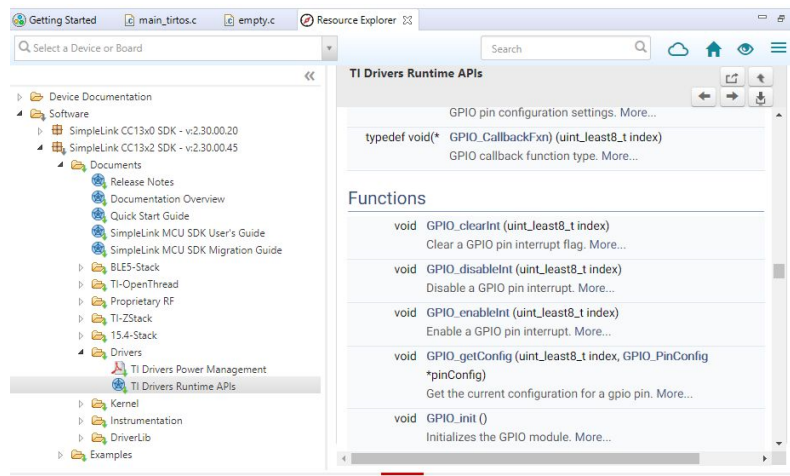
Polo de Inovação

Campos dos Goytacazes



Sensor Controller Studio (SCS)

Dicas Importantes para programar



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Bluetooth Low Energy (BLE)

- **Conexão**

Advertiser
CC1352R

Scanner
Mobile

- **Protocolos**

ATT (Attribute Protocol)
GATT (Generic Attribute Profile)
Serviços e atributos

- **Ferramentas**

BTool (vem no SDK) com Projeto Host_test (Scanner Side)

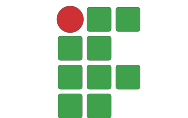
TI BLE5-Stack

Projeto ProjectZero (Comece modificando)

Gerador de Profile:

Bluetooth Developer Studio (Descontinuado)

http://dev.ti.com/tirex/content/simplelink_academy_cc13x2sdk_2_20_03_05/modules/ble5stack/ble_01_custom_profile/ble_01_custom_profile.html#gatt-server-callbacks



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



Bluetooth Low Energy (SCS)

- Gere seu profile
 - Adicione Serviço
 - Adicione Características ao Serviço
- Adicione o Serviço
- Inicialize a característica
- Personalize a função de callback



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



SmartRF Studio

Laboratório 2:

Tarefa 1: Usar o SmartRF Studio separadamente para testar comunicação.

Tarefa 2: Usar firmware como TX e SmartRF Studio como RX.




**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



SmartRF Studio

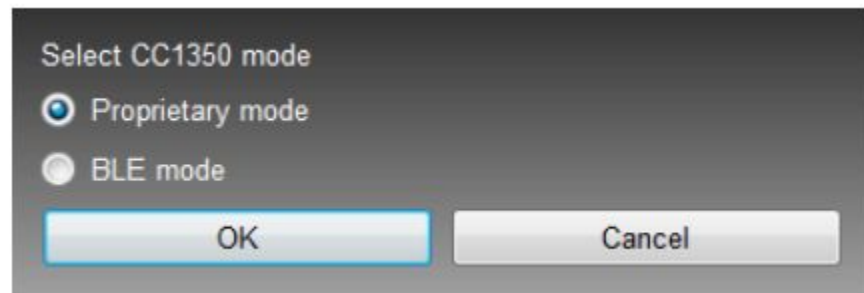
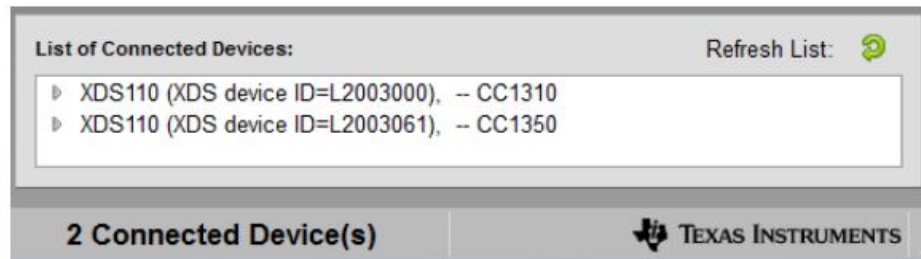
TAREFA #1

1. Conecte as placas e abra o programa.
2. Clique duplo em um dispositivo
3. Selecione o modo proprietário.
4. Abra a aba de “Packet TX” para configurá-lo como um transmissor.
5. Retorne ao menu e faça o mesmo para o outro dispositivo.
6. Abra a aba de “Packet RX” para configurá-lo como um receptor.
7. Selecione uma das configurações para ambos os dispositivos.
8. Inicie a recepção.
9. Inicie a transmissão.
10.  Observe.

**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

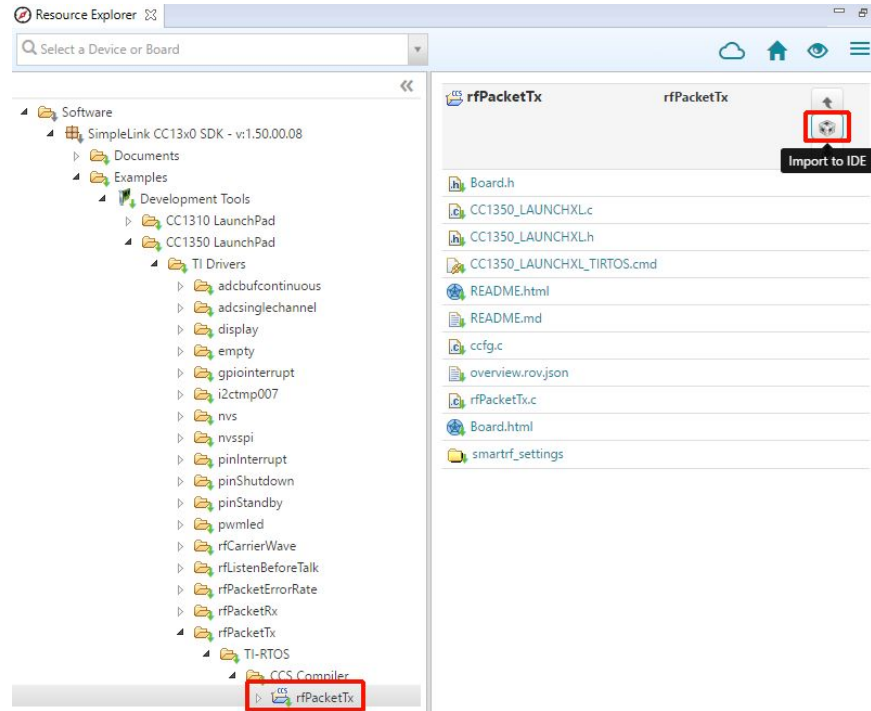
Campos dos Goytacazes



Sensor Controller Studio (SCS)

TAREFA #2

1. Feche o SmartRF Studio, retire uma das placas
2. Importe o Projeto TX no CCS
3. Compile.
4. Execute
5. Observe se o LED verde está piscando (indicando a transmissão).
6. Aqui temos um transmissor em firmware.



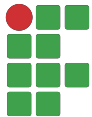
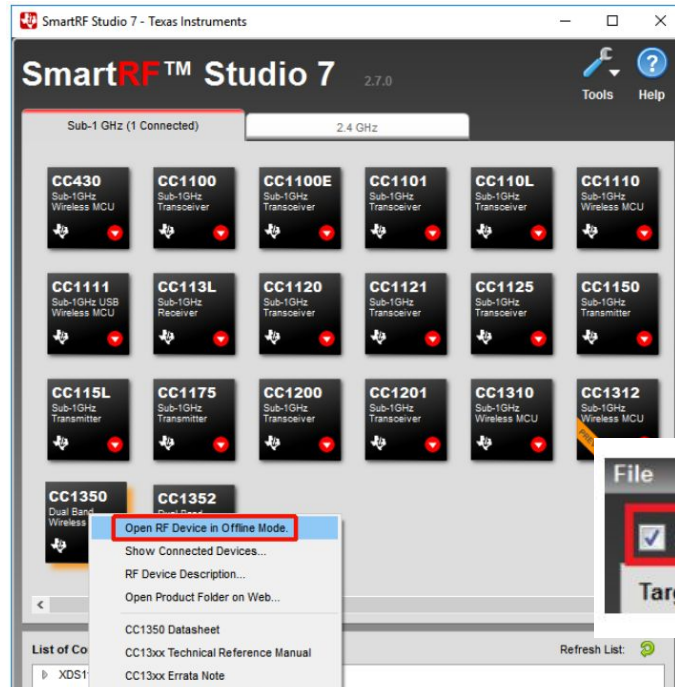
**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

1. Abra o SmartRF Studio para gerarmos drivers com outras configurações de rádio.
2. Abra um painel offline no modo proprietário para a família do seu dispositivo.
3. Habilite o “Command view”.
4. Primeiro clique na primeira configuração. Depois mude os parâmetros:



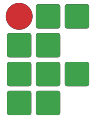
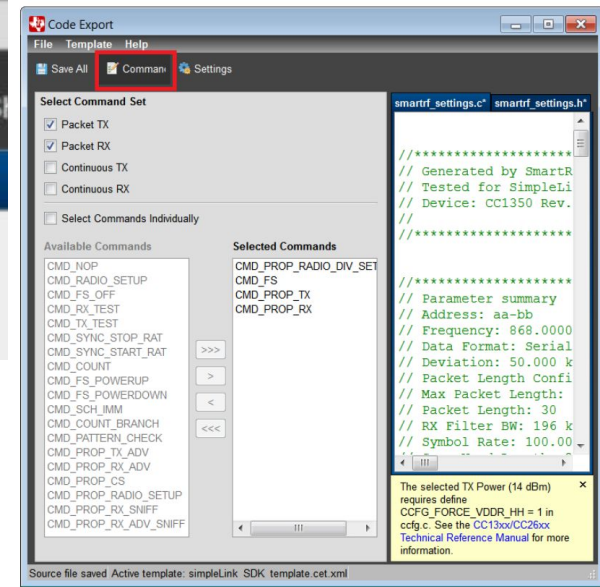
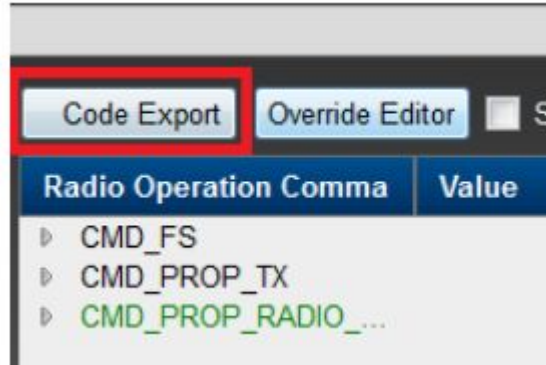
**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



Sensor Controller Studio (SCS)

1. Em Command View clique em Code Export.
2. Clique no botão comandos.
3. Observe os arquivos gerados.
4. Vá no seu projeto no CCS e veja o caminho dele em Properties.
5. Apague os arquivos de configuração “smartfsettings.h/.c” e salve os gerados no mesmo diretório.
6. Compile e execute.



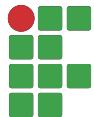
**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação
Campos dos Goytacazes



SmartRF Studio

1. **Feche os painéis abertos no SmartRF Studio.**
2. **Conecte o outro dispositivo.**
3. **Abra o outro dispositivo.**
4. **Configure conforme o primeiro.**
5. **Configure para receber infinitamente.**
6. **Formato em hexa.**
7. **Inicialize a recepção.**



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



SUORTE

1º. Fazer uma conta no site da Texas Instruments.

Link: www.ti.com (Login / Register)

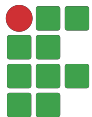
Dê preferência ao email institucional/Empresarial ou profissional

Alguns recursos e programas da TI só são disponíveis através do cadastro. Inclusive, para programar usando o CCS Cloud é necessário uma conta.

2º. Utilize o Resource Explorer.

Link: <http://dev.ti.com/tirex/#/All>

Trata-se de um acervo de documentação extensa, códigos exemplos e conteúdos de estudo. A SimpleLink Academy está contida no Resource Explorer e contém tarefas explicadas passo a passo para programar os MCUs. O Resource Explorer também é acessível pelo CCS na aba view -> Resource Explorer. Utilizando o RE pela web é possível importar os códigos e compilar na hora no CCS Cloud, assim como na IDE.



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes



SUPOORTE

3º. Utilize a comunidade.

Link: <http://e2e.ti.com/>

A TI oferece uma comunidade consolidada com programadores e funcionários e possui um tempo de resposta muito curto e conta com diversos fóruns. Lá você pode perguntar, responder e consultar.

4º. Utilize a documentação do SDK.

Ao instalar o SDK, ele cria um diretório no seu computador. Nesse diretório contém a documentação dos drivers. Tornando-o muito útil para consultar a API e saber como usá-la. (Parâmetros, tipos, retornos, interdependência de funções, etc). Ao instalar o SDK, é possível acessar os exemplos e conteúdo através do Resource Explorer dentro do CCS.



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes

5º. Utilize a SimpleLink Academy.

Também através do SDK, você pode participar do “treinamento” através da da SimpleLink Academy.

6º. Canal do YouTube Texas Instruments.

Conheça aplicações, curiosidades, novidades, testes e muito mais.



Referências:

RTOS (Conceitos Gerais de RTOS):

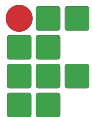
http://dev.ti.com/tirex/content/simplelink_academy_cc13x2sdk_2_20_03_05/modules/rtos/rtos_concepts/rtos_concepts.html

TI-RTOS (Conceitos, Debugger e Laboratório):

http://dev.ti.com/tirex/content/simplelink_academy_cc13x2sdk_2_20_03_05/modules/rtos/tirtos_basics/tirtos_basics.html

POSIX (Conceitos e Laboratório):

http://dev.ti.com/tirex/content/simplelink_academy_cc13x2sdk_2_20_03_05/modules/rtos/posix_project_zero/posix_project_zero.html



**INSTITUTO
FEDERAL**
Fluminense

Polo de Inovação

Campos dos Goytacazes

