EA871

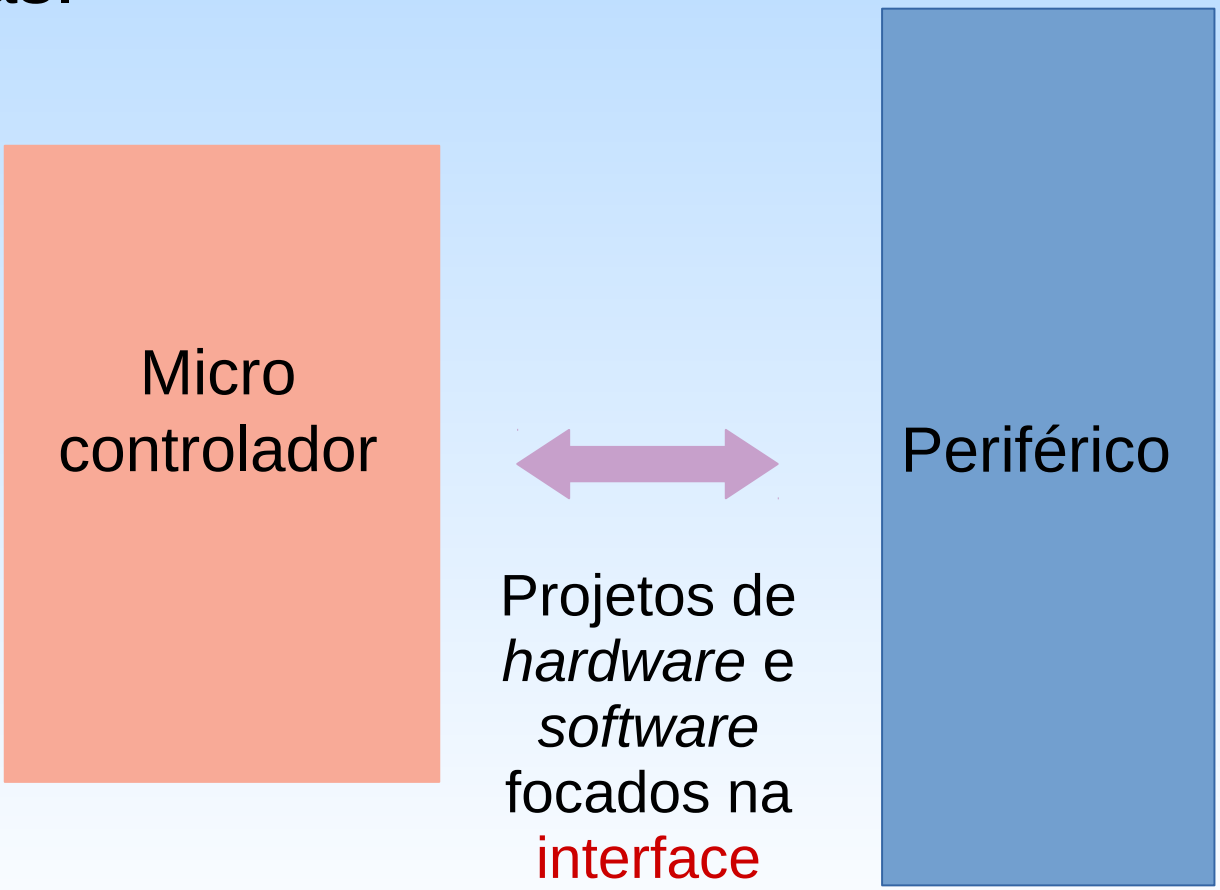
Chaves

Bouncing

Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

Microcontrolador

- Um “micro-computador” dedicado ao controle de dispositivos/periféricos com mínimas intervenções humanas.



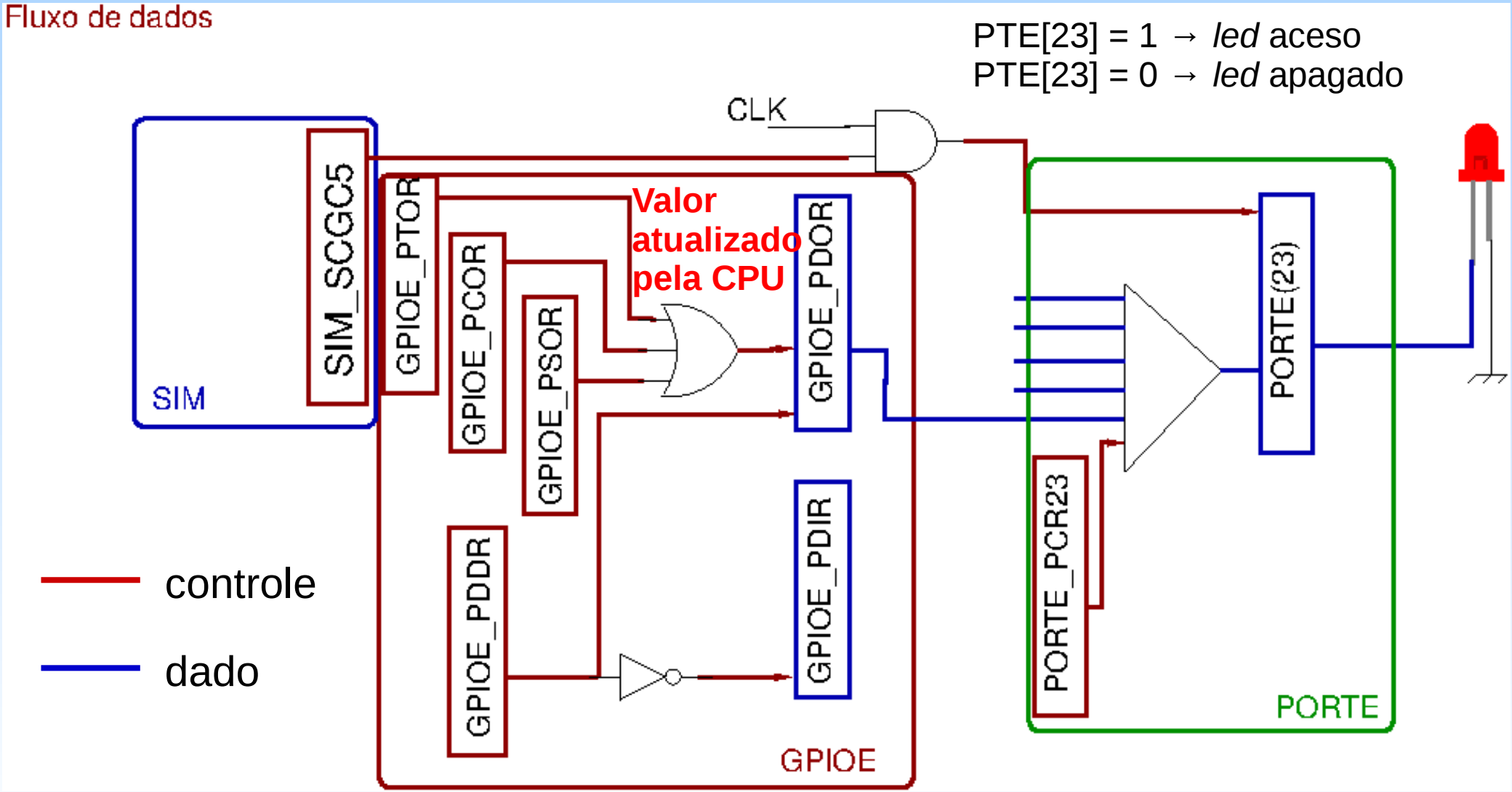
Compatibilidades

- **Funcional**
  - Funções de sinais compatíveis
- Elétrica
  - Níveis elétricos de sinais compatíveis
- **Temporal**
  - Temporização de sinais compatível
- Mecânica
  - Conexões de sinais compatíveis

Microcontrolador – *Led*

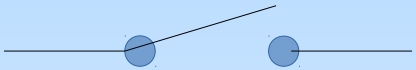
	Microcontrolador	*Led*
Funcional	1/0	Aceso/Apagado
Elétrico	3V3/0V	3V3/0V
Temporal	1 ciclo de instrução ~ 50ns	Resposta na ordem de ns
Mecânico	Pino de saída e terra	Dois pinos (catodo e anodo)

Programação do Microcontrolador

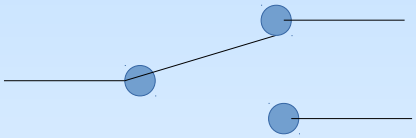


Chaves Simples

- Quanto ao número de posições
  - Um pólo e uma posição (*Single Pole Single Throw* – SPST)



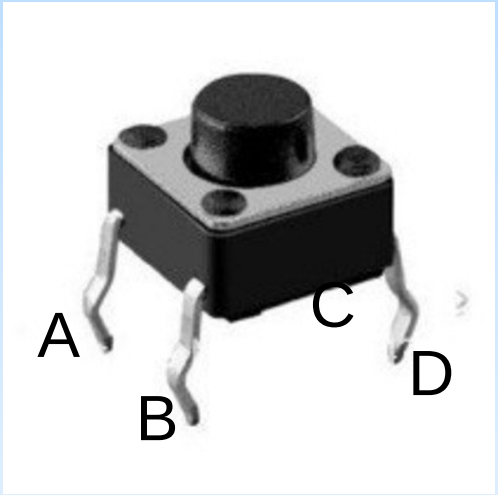
  - Um pólo e duas posições (*Single Pole Double Throw* – SPDT)

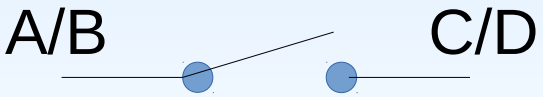


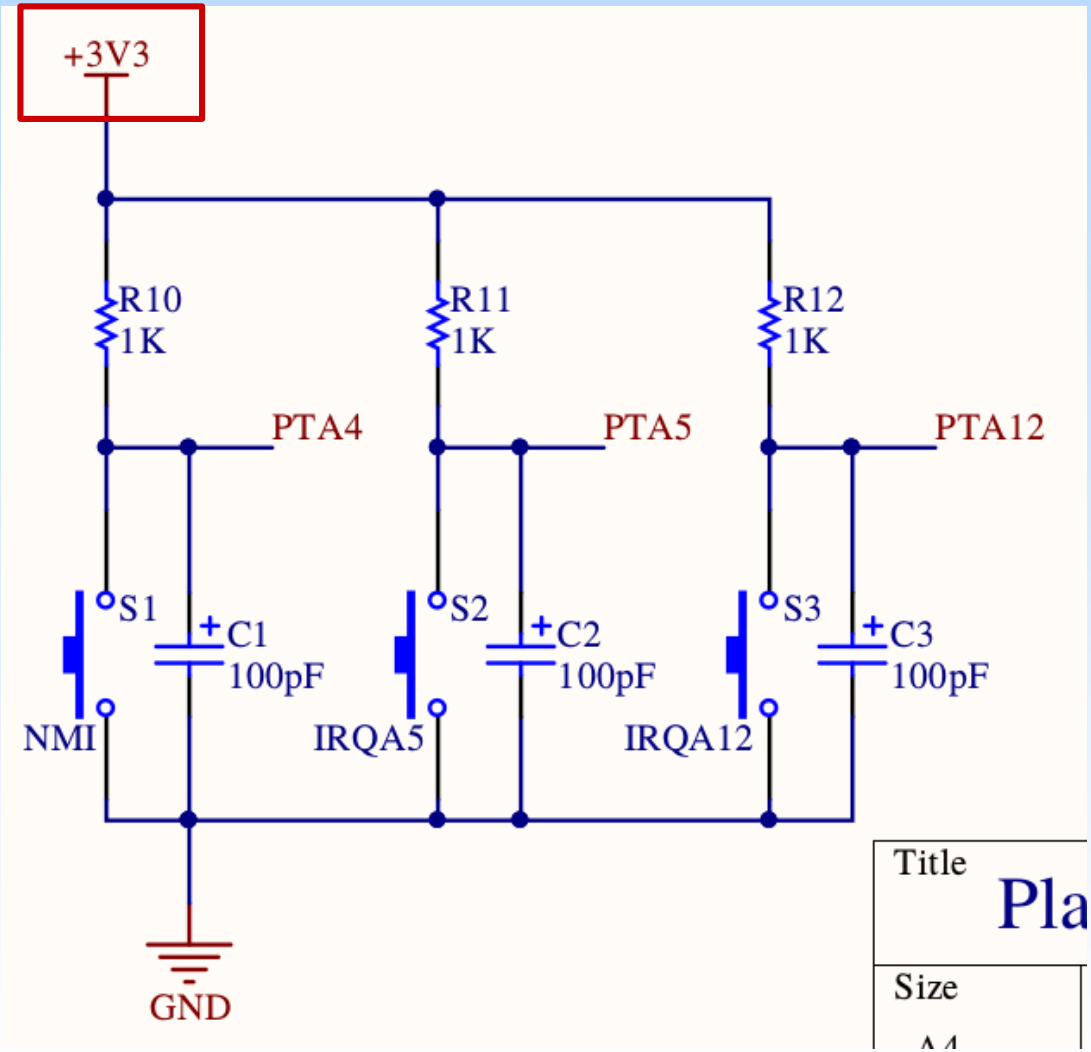
- Quanto à manutenção do estado
  - Permanente: dois estados bem definidos. Ex.: chaves liga-desliga
  - Momentânea: passa momentaneamente para outro estado quando se atua sobre ela. O seu estado normal pode ser
    - Aberto: *push-button* normalmente aberto
    - Fechado: *push-button* normalmente fechado

Chaves no *Shield* FEEC871

- Chaves SPST, momentâneas, normalmente abertas.





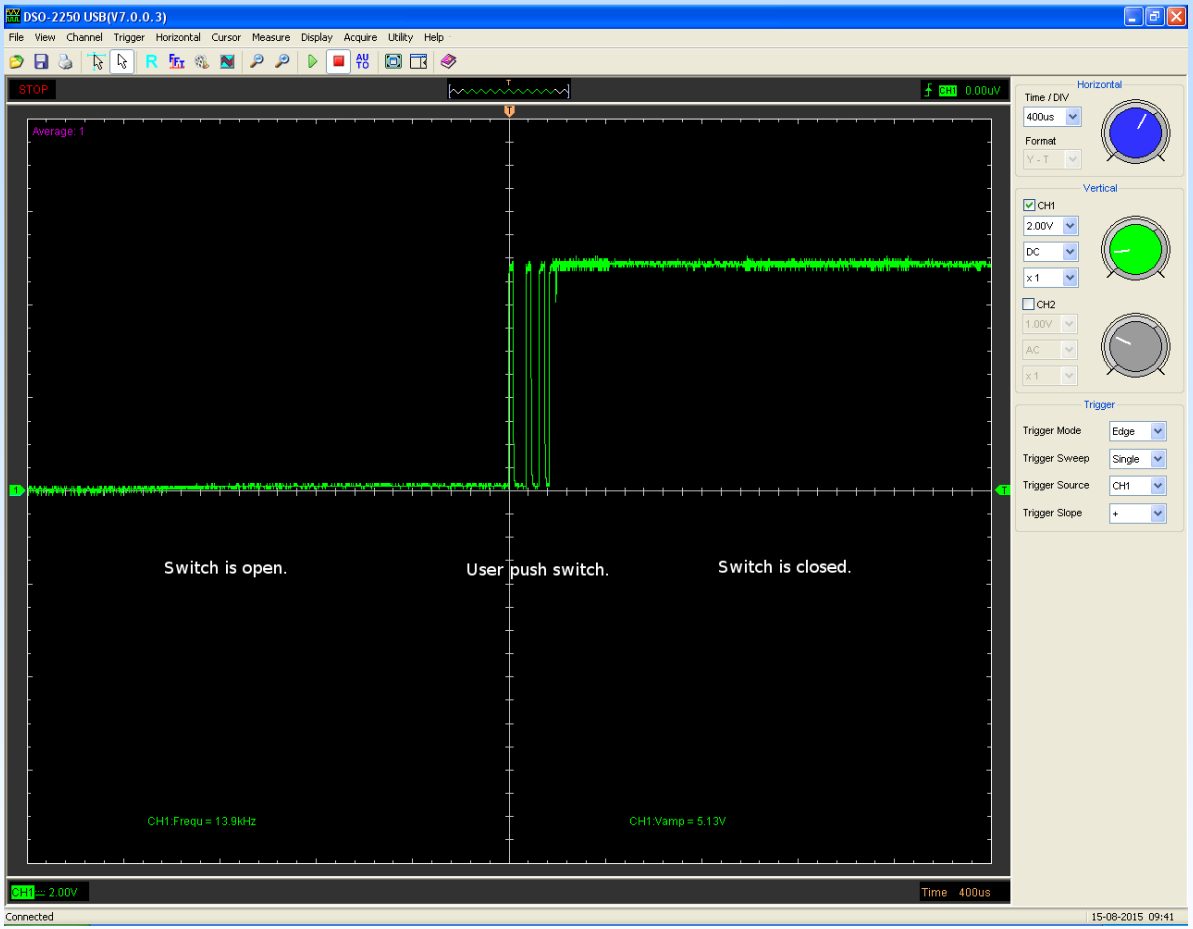


Microcontrolador – Chave Mecânica

	Microcontrolador	*Chave*
Funcional	1/0	Abre/Fecha
Elétrico	3V3/0V	3V3/0V (**até se estabilizar**)
Temporal	1 ciclo de instrução ~ 50ns	**ação assíncrona do usuário**
Mecânico	Pino de entrada e terra	Um pólo e uma posição.

Bounce em Chaves

- Consiste de uma sequência de abre—fecha de uma chave, tipicamente 10 a 100 vezes num intervalo de 1 ms, que faz um circuito digital interpretar 10 a 100 transições ao invés de uma única transição em 1 ms. E o período típico de *bounce* numa chave é 20ms.



Fonte: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/

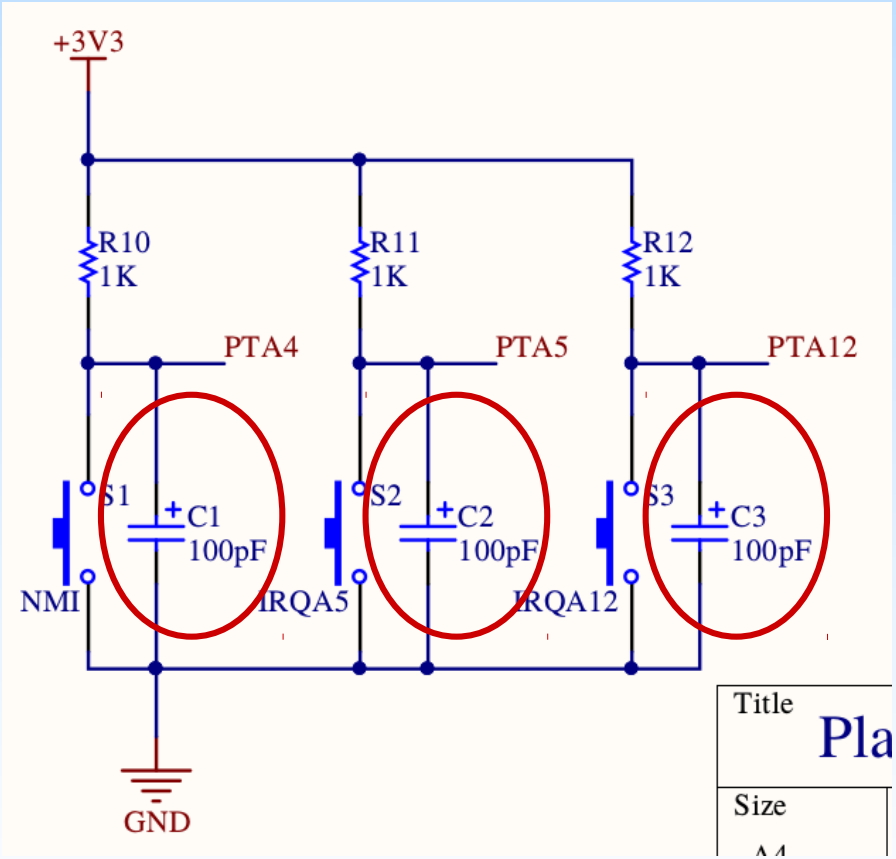
Um possível efeito intrigante

- Efeito visual esperado: a cada acionamento na chave, o *led* aceso “move” para o próximo *led* à esquerda.

Fonte: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/

Debounce

- Por *hardware*
  - *Circuito RC (carga e descarga de capacitor)*

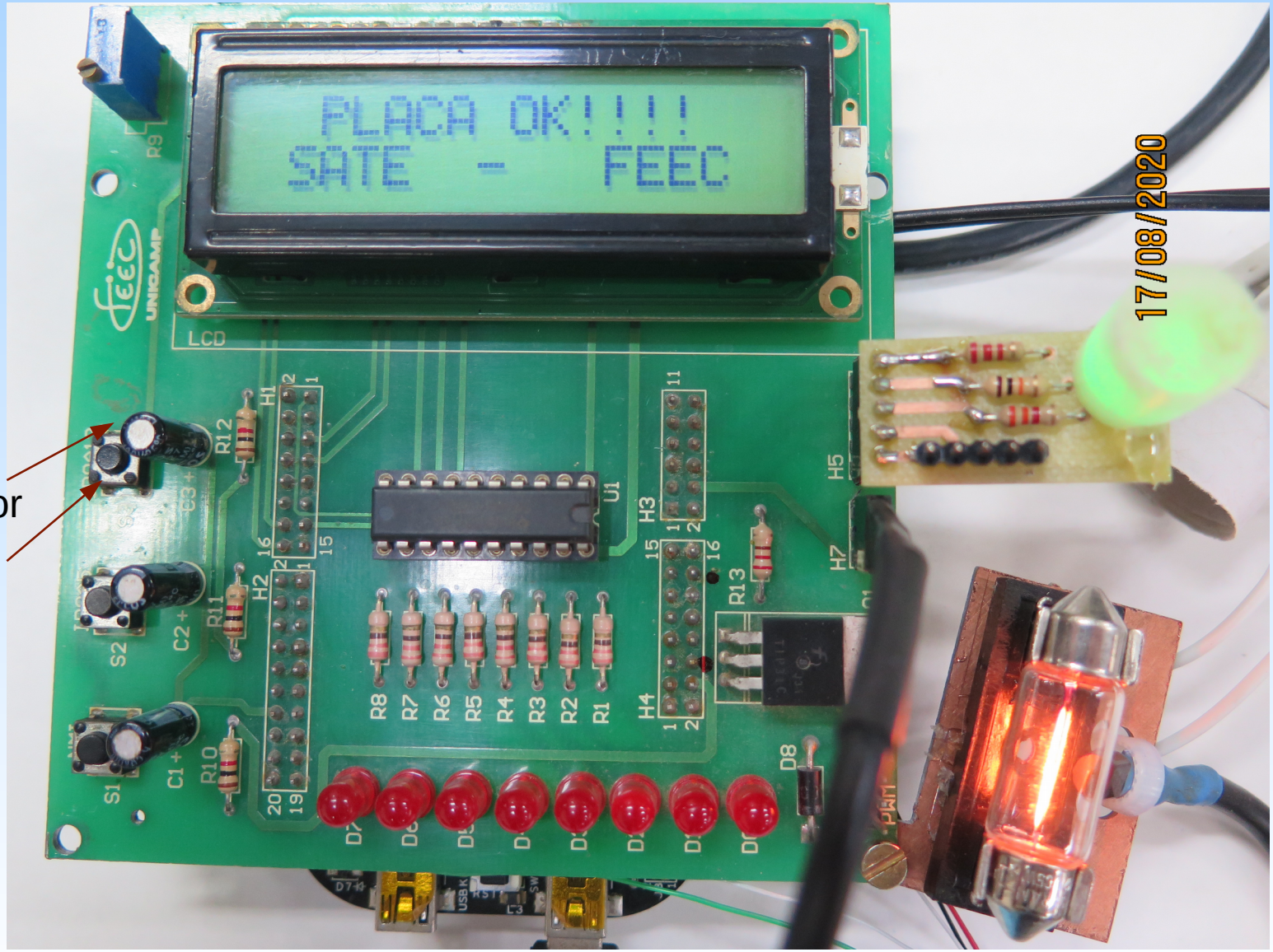


- Por *software*
  - Inserir um atraso de 10 a 50ms antes de interpretar o estado da chave.
  - Aplicar o mecanismo de interrupção na inserção de atrasos.

Circuito com *Debounce*

Fonte: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/

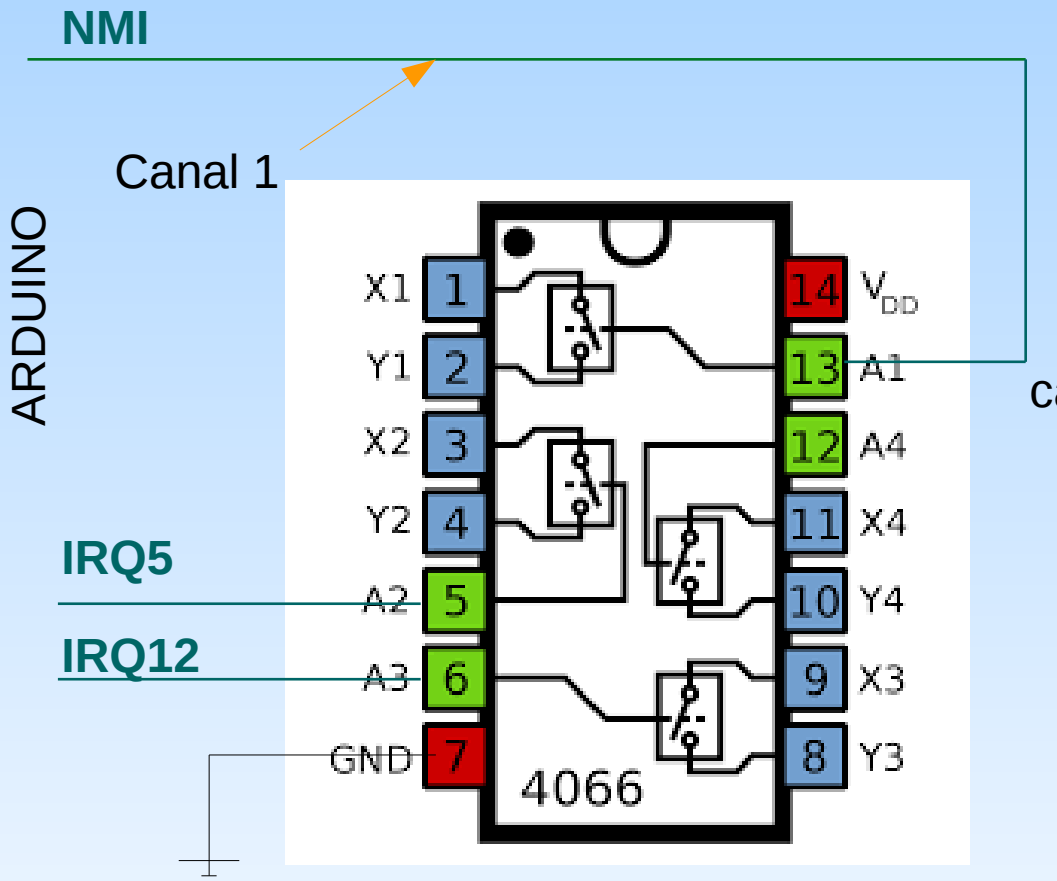
Debounce

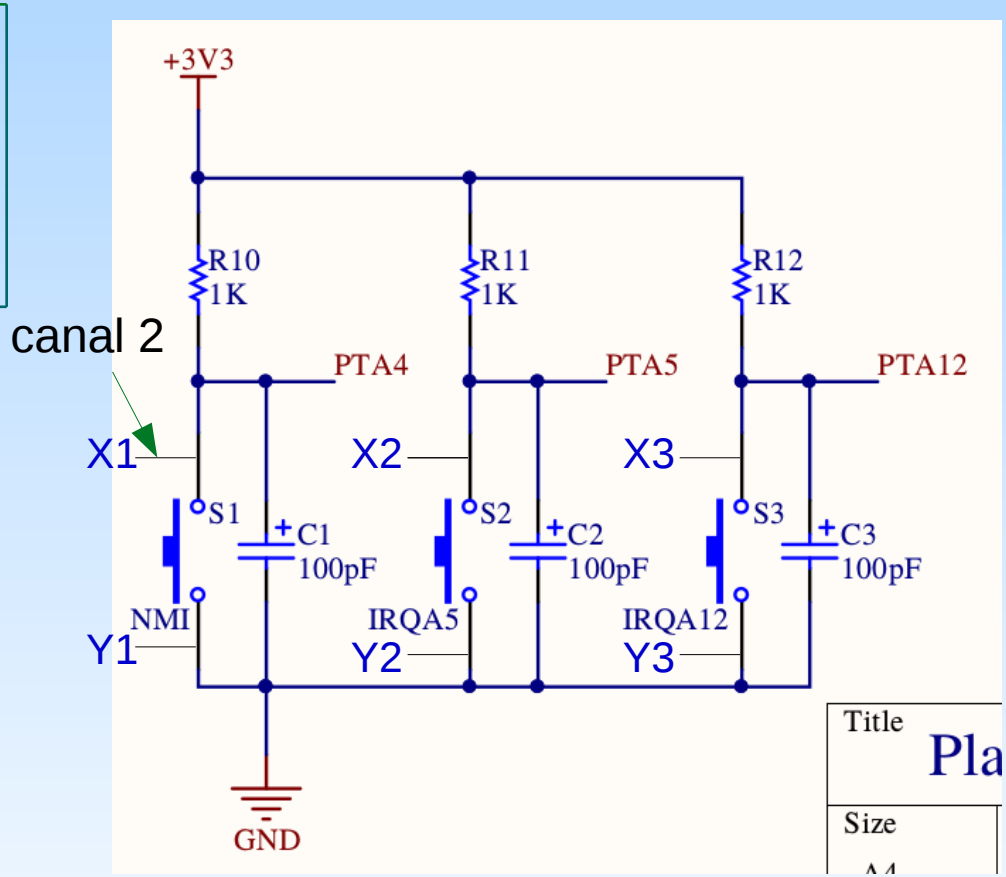


Microcontrolador – Chave Mecânica

	Microcontrolador	*Chave*
Funcional	3V3/0V	Aberta/Fechada
Elétrico	3V3/0V	3V3/0V (~~**até se estabilizar**~~)
Temporal	1 ciclo de instrução ~ 50ns	**ação assíncrona do usuário**
Mecânico	Pino de saída e terra	Um pólo e uma posição.

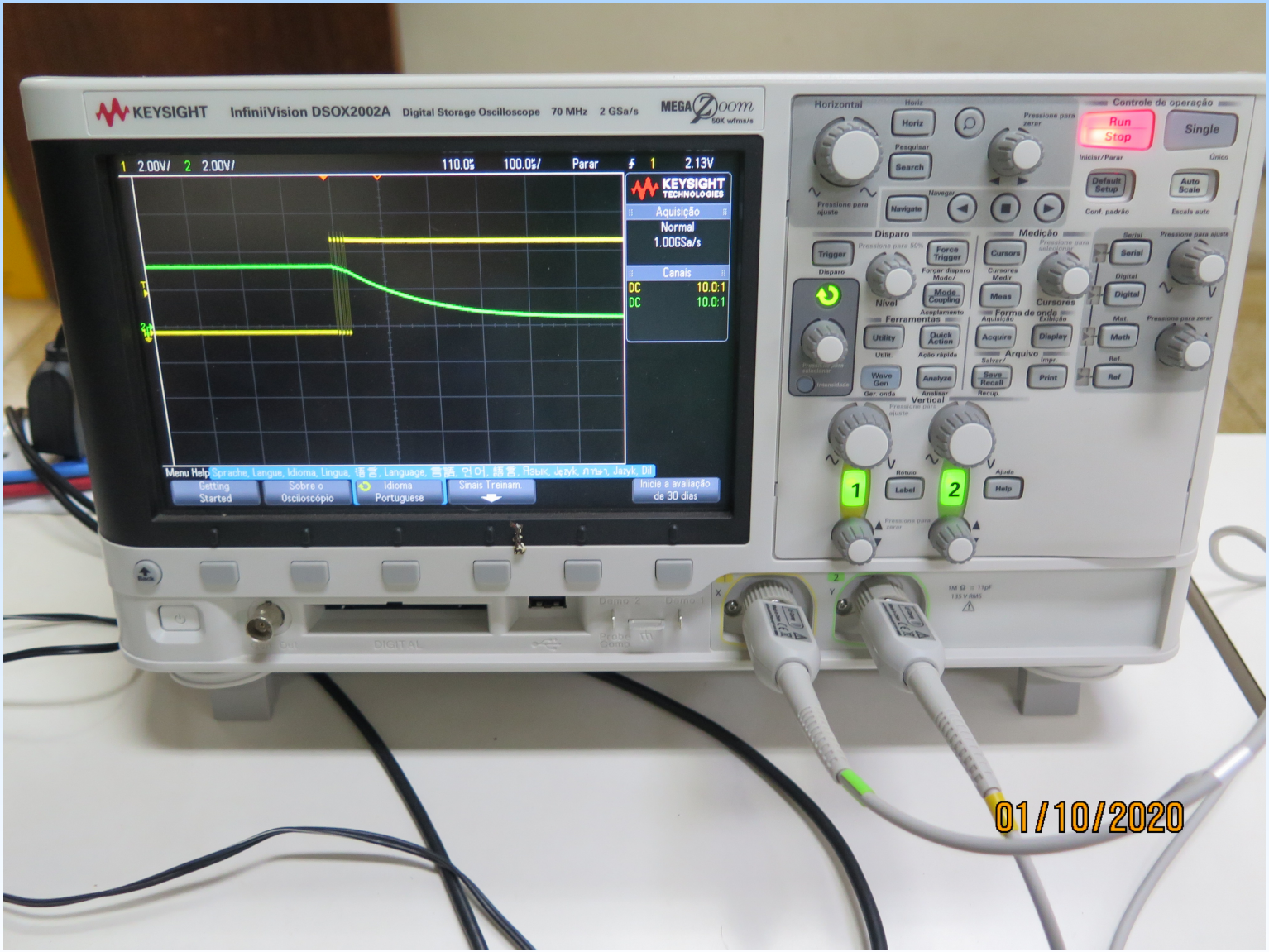
Versão Remota das Chaves



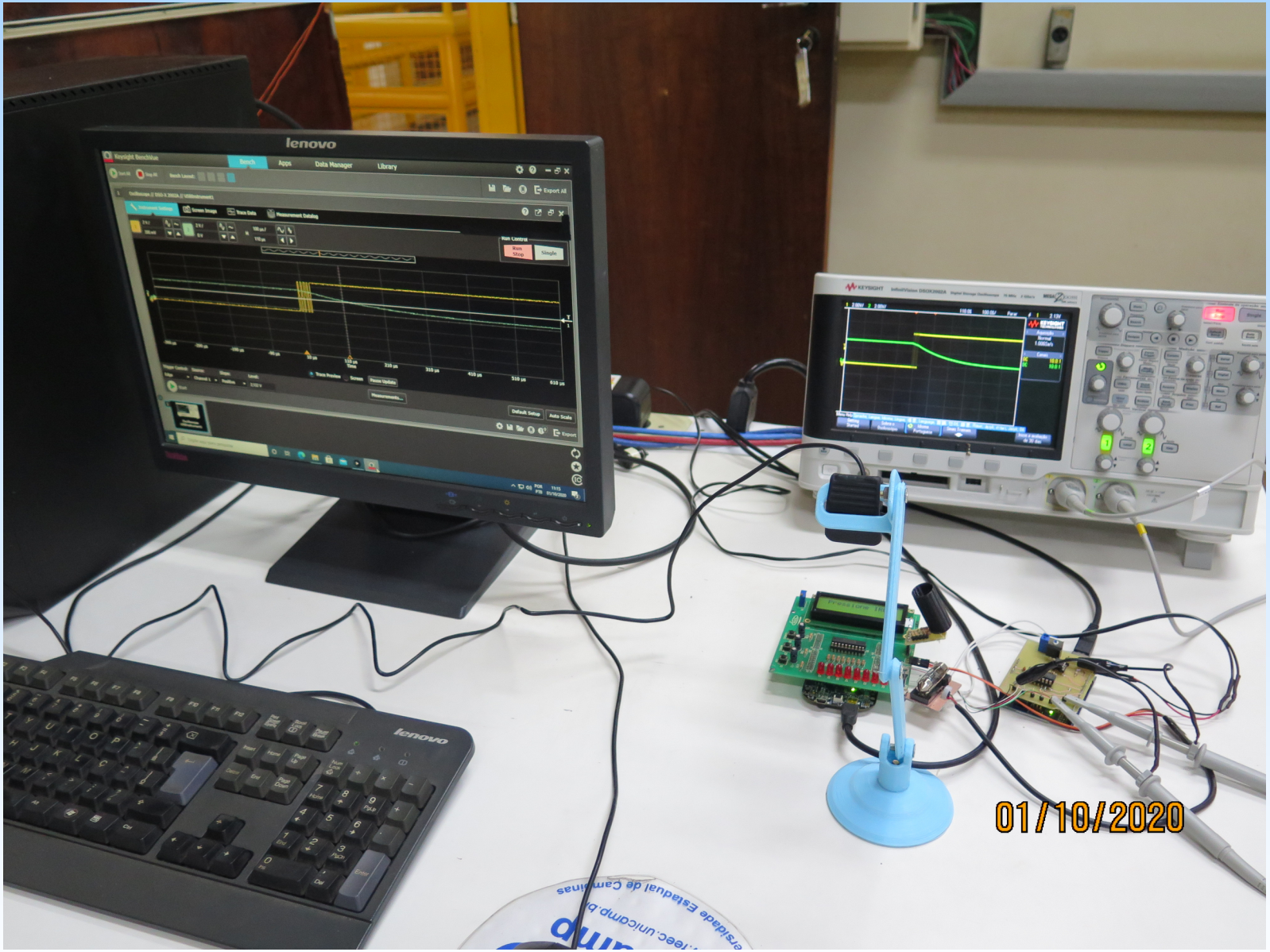


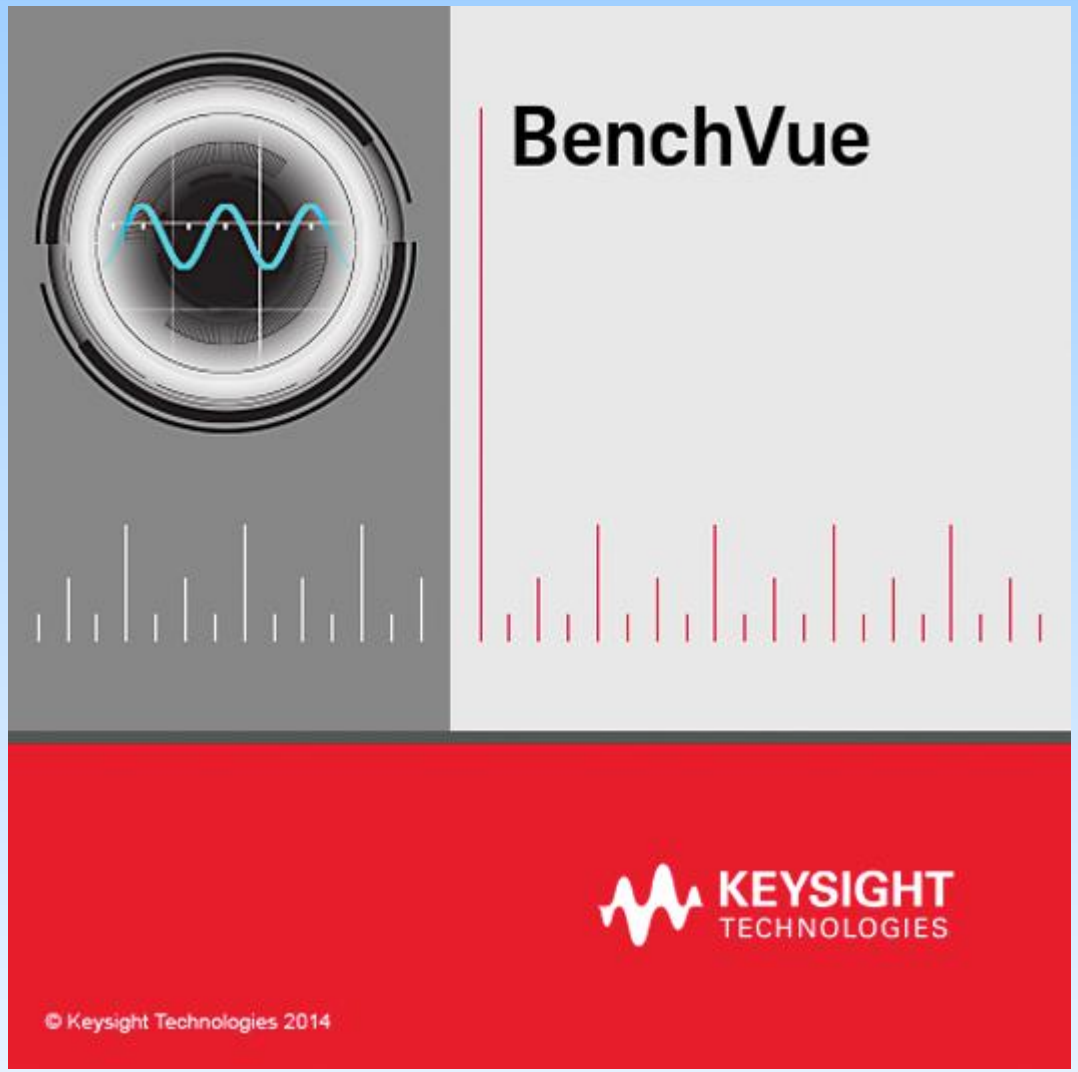
Emulação de *bounces* em Arduino

Debounced Signal



Acesso Remoto via BenchVue





BenchVue 2017 Software

Informações Adicionais

- Switch Basics

  https://learn.sparkfun.com/tutorials/switch-basics/all

- Chaves

  http://newtoncbraga.com.br/index.php/almanaque-tecnologico/192-c/1367-alm217

- Switch Bounce and How to Deal with It

  https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/

- Designing an RC debounce circuit

  https://mayaposch.wordpress.com/2018/06/26/designing-an-rc-debounce-circuit/

- Keysight Technologies – BenchVue 2017 Software: Demo Guide

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/manuais/BenchVue_Keysight_2017.pdf

EA871

Chaves

Polling

Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

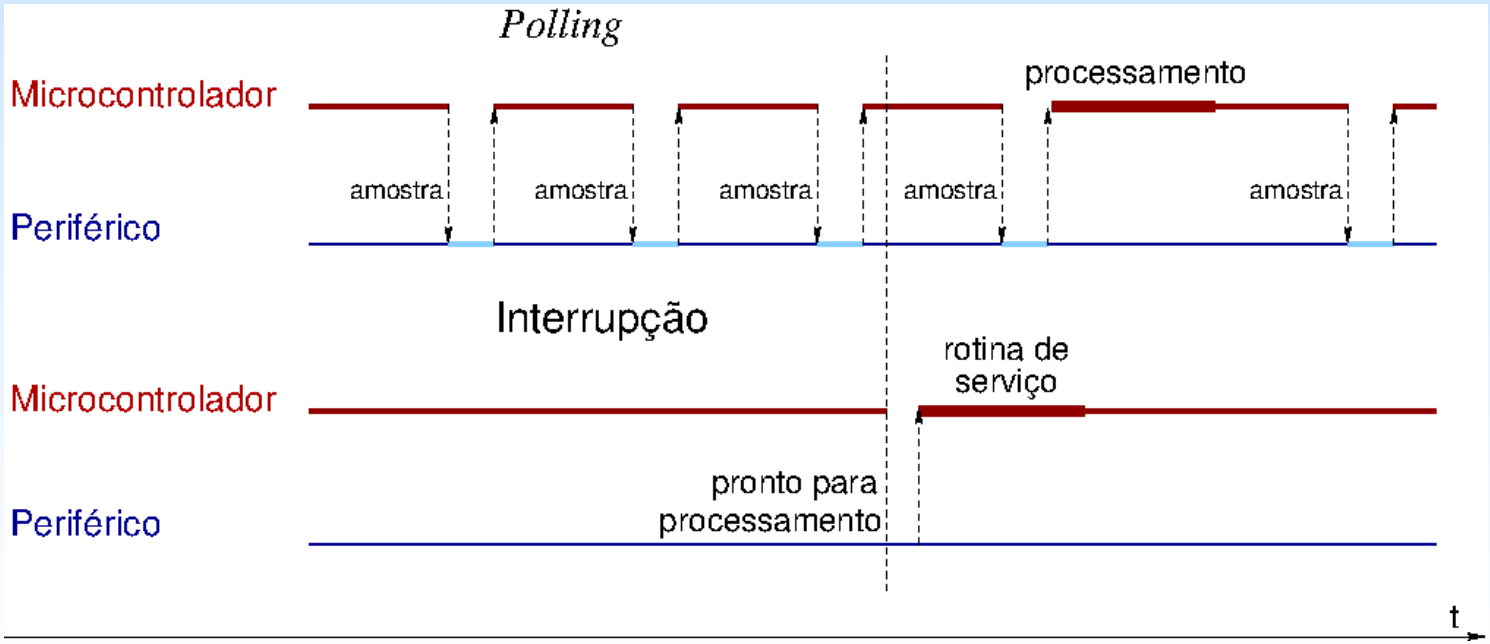
Microcontrolador – Chave Mecânica

	Microcontrolador	*Chave*
Funcional	3V3/0V	Abre/Fecha
Elétrico	3V3/0V	3V3/0V (~~**até se estabilizar**~~)
Temporal	1 ciclo de instrução ~ 50ns	**ação assíncrona do usuário**
Mecânico	Pino de saída e terra	Um pólo e uma posição.

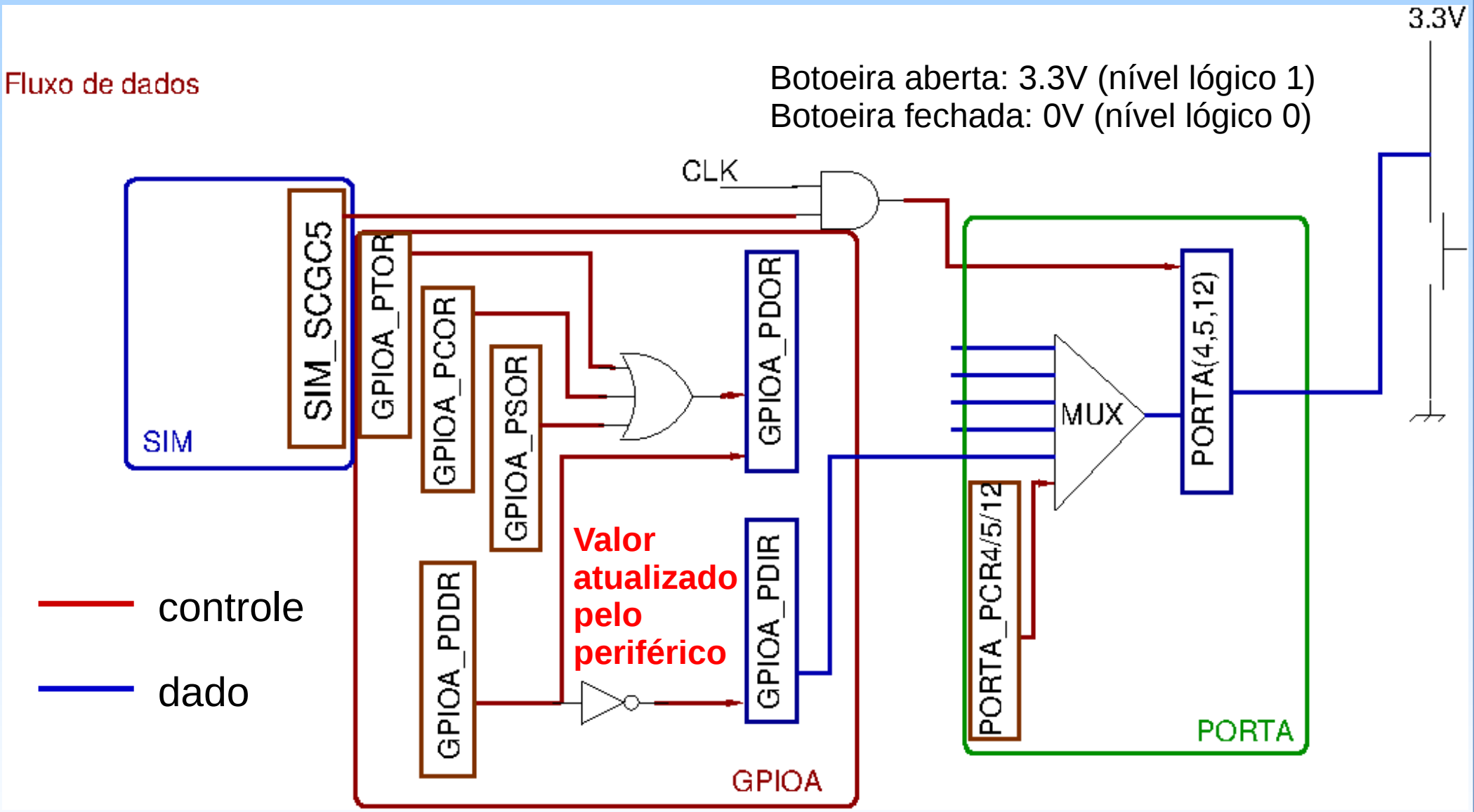
Como um microcontrolador consegue perceber que uma chave mudou de estado?

Técnicas

- ***Polling***
  - Processador amostra continua/periodicamente o estado do recurso para certificar se ele está pronto para executar uma operação.
- **Interrupção**
  - Processador executa uma operação com uso de um recurso somente quando o recurso estiver pronto.



Polling do Registrador de Entrada



Configuração dos Registradores de Controle

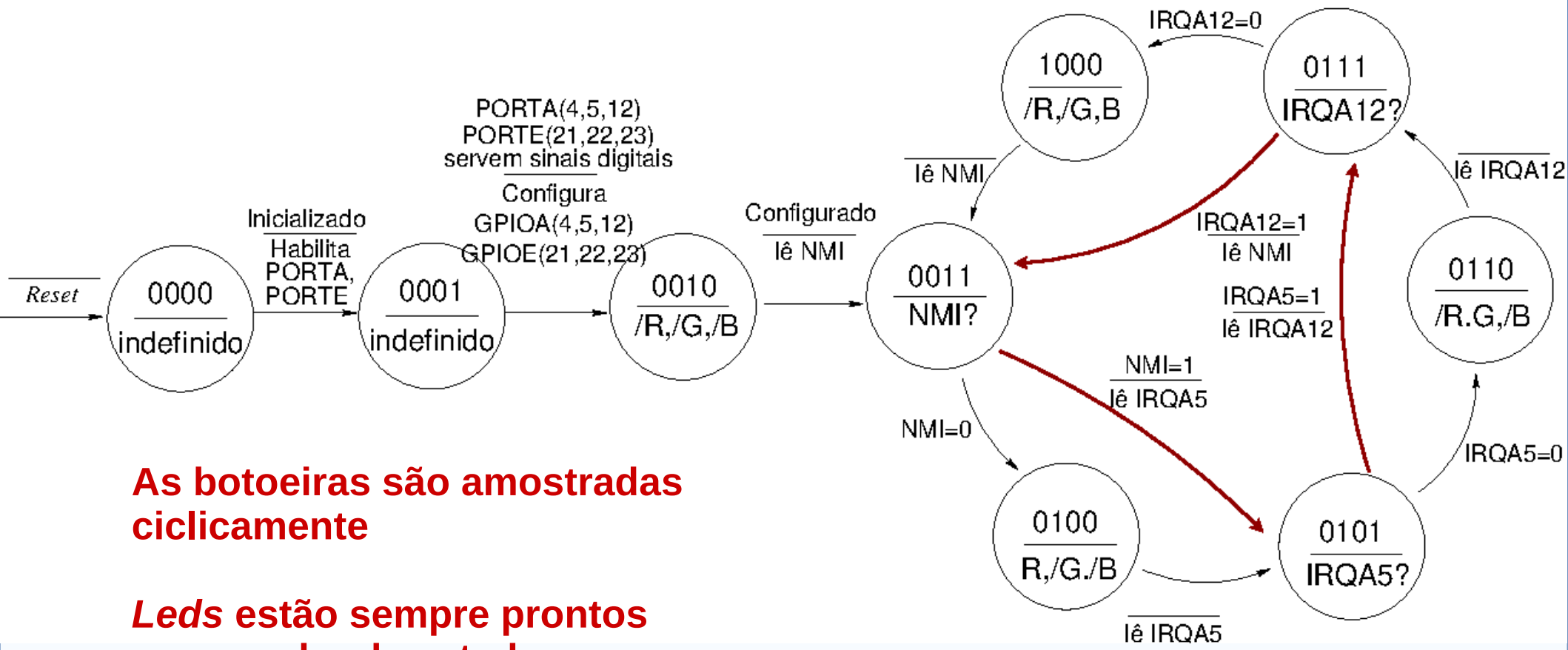
- SIM_SCGC5:
  - Habilitar o *clock* da porta A
- PORTA_PCRn:
  - Setar função dos pinos 4, 5 e 12 em “digital” (MUX = 0b001)
  - Atenção como os valores assumidos neste campo para cada registrador num *Reset*!
  - *Por default*, o pino 4 da porta A é um pino ***Non-Maskable Interrupt*** (NMI), cujo campo MUX do registrador PORTA_PCR4 é 0b111 no *reset* → Elaborar operações lógicas apropriadas.
- GPIOA_PDDR:
  - Setar que o sentido de fluxo nos pinos 4, 5 e 12 seja de entrada

Projeto-exemplo

- Acender os *leds* R, G e B usando, respectivamente, as botoeiras NMI, IRQ5 e IRQ12.
  - Periféricos de saída (digitais): *leds* R, G e B
    - Nível lógico 1 (3.3V): aceso
    - Nível lógico 0 (0V): apagado
  - Periféricos de entrada (digitais): botoeiras remotas
    - Nível lógico 1 (3.3V) de controle : botão em estado normal.
    - Nível lógico 0 (0V) de controle: botão acionado

Controle dos *Leds* pelas Botoeiras

Máquina de Estados



As botoeiras são amostradas ciclicamente

***Leds* estão sempre prontos para mudar de estado**

Técnicas de Programação

- Declaração de tipos de dados mais intuitivos

```
typedef enum estado_tag {
    DESABILITA,     /**< falso/apaga/desativa/liga */
    HABILITA,       /**< verdadeiro/acende/ativa/fecha */
} estado_type;

typedef struct RGB_bool_tag {
    estado_type vermelho;
    estado_type verde;
    estado_type azul
} rgb_bool_type;
```

Técnicas de Programação

- Declaração de tipos de dados mais intuitivos

```
rgb_bool_type cor[3]=
 { {HABILITA,DESABILITA,DESABILITA},
    {DESABILITA,HABILITA,DESABILITA},
    {DESABILITA,DESABILITA,HABILITA} };


enum cor_tag {
    VERMELHO,
    VERDE,
    AZUL
};
```

Técnicas de Programação

- Modularização de códigos
  - Inicialização de controle de chaves: GPIO_initSwitches()
  - Inicialização de controle de *leds* : GPIO_initLedRGB()
  - Amostragem do estado de cada chave:

    GPIO_sampleSwNMI(), GPIO_sampleSwIRQ5(), GPIO_sampleSwIRQ12().
  - Atribuição de valores a cada *led*:

    GPIO_ledRGB(RGB_bool_type)

Pseudocódigo

- GPIO_initLedRGB(); inicializa os registradores de controle dos *leds*
- GPIO_initSwitches(): inicializa os registradores de controle das chaves
- Laço:
  - Se (GPIO_sampleSwNMI() == HABILITA) então GPIO_LedRGB(cor[VERMELHO]);
  - Se (GPIO_sampleSwIRQ5() == HABILITA) então GPIO_LedRGB(cor[VERDE]);
  - Se (GPIO_sampleSwIRQ12() == HABILITA) então GPIO_LedRGB(cor[AZUL]);

CodeWarrior IDE Development Suite

Informações Adicionais

- KL25 Sub-Family Reference Manual

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf

  - Mapa de memória: Seção 4.2 (página 105)
  - SIM: Capítulo 12 (página 192)
  - PORT: Capítulo 11 (página 175)
  - GPIO: Capítulo 41 (página 771)