Tempo e execução de múltiplas tarefas

Prof. Dr. Roberto Kenji Hiramatsu
Prof. Dr. João Henrique Correia Pimentel

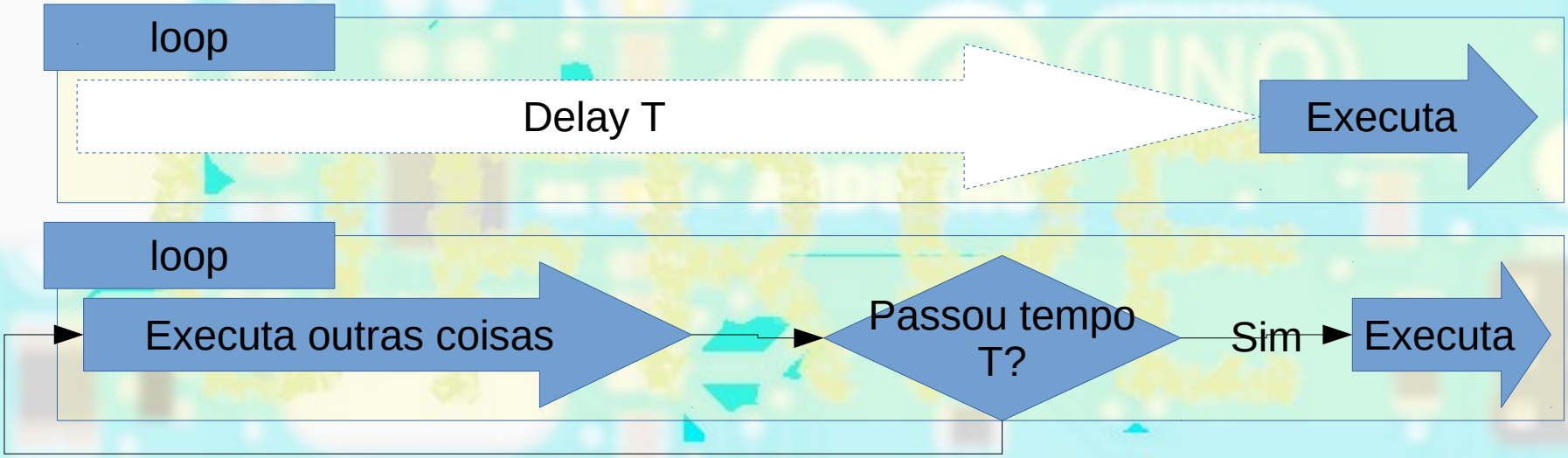
Agenda

Tratamento de tempo na plataforma
Usar medição de tempo ao invés de delay
Vamos piscar o led da placa medindo o tempo
Eventos de entrada com pushbutton
Implementado tratamento anti trepidação
Geração de evento
Multitarefa
Exercício proposto

Tratamento de tempo na plataforma

- Esperando um tempo
  - delay(<tempo em ms>)
  - delayMicrosecounds(<tempo em microssegundos>)
- Quando usa-se o delay:
  - Não permite executar nenhuma outra tarefa
  - A exceção são os eventos tratados por interrupção
- Medindo o tempo:
  - millis() - medição do tempo com limite para medir até 50 dias. Retorna o tempo em milissegundos
  - Micros() - medição do tempo com limite de até 70 minutos.

Usar medição de tempo ao invés de delay



- Observe que para medir um tempo é necessário ter uma variável que mantenha o valor
  - Uso de variável global ou uso de modificador static

Vamos piscar o led da placa medindo o tempo

Exemplo_02_pisca_led_medindo | piscar.cpp | piscar.h

```
1 /*Autor: Roberto Kenji Hiramatsu
2 Implementando */
3 void iniciaPiscar(long tempo,int pino);
4 void piscaLed();
```

Exemplo_02_pisca_led_medindo | piscar.cpp | piscar.h

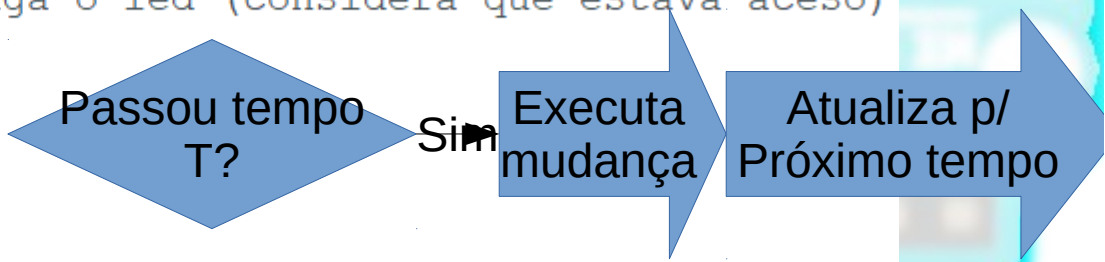
```
1 /*Autor: Roberto Kenji Hiramatsu
2 Exemplo para medir e fazer um led da placa piscar */
3 #include "piscar.h"
4 void setup() {
5   iniciaPiscar(1000,13);
6 }
7 void loop() {
8   piscaLed();
9 }
```

Exemplo_02_pisca_led_medindo | piscar.cpp | piscar.h

```
1 /*Autor: Roberto Kenji Hiramatsu
2 Exemplo para medir e fazer um led da placa piscar */
3 #include "piscar.h"
4 #include <arduino.h>
5 void iniciaPiscar(long tempo,int pino){
6
7 }
8 void piscaLed(){
9
10 }
```

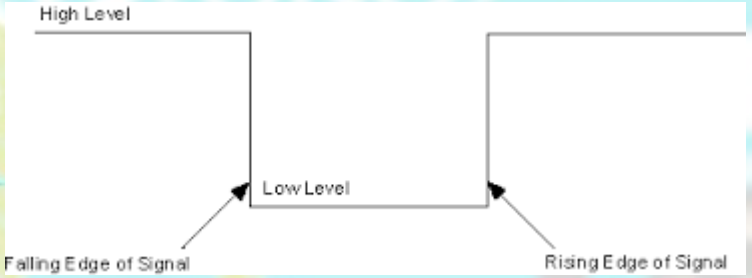
Implementando piscar.cpp

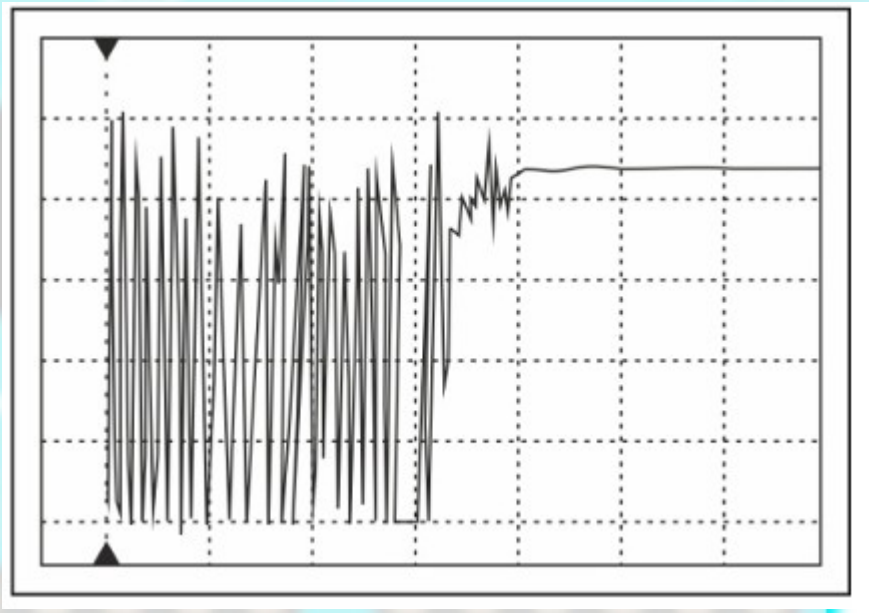
```
3  #include "piscar.h"
4  #include <arduino.h>
5  long esperaT;           // cria para indicar tempo de espera
6  long tempoAnterior;     // marcacao de tempo anterior
7  int pinoLed;            // variavel para lembrar numero do pino
8  bool aceso=true;       // saber se esta acesso ou apagado
9  void iniciaPiscar(long tempo,int pino){
10   esperaT=tempo;              // guarda tempo que deve esperar
11   tempoAnterior=millis();     // mede o tempo inicial
12   pinoLed=pino;               // guarda o pino a  ser acionado
13   pinMode(pinoLed,OUTPUT);   // Pino em modo saida
14   digitalWrite(pinoLed,HIGH);
15 }
16 void piscaLed(){
17   if((millis()-tempoAnterior)>esperaT){ // se a diferenca de tempo for
18                       // maior que o tempo de espera executa o bloco se
19     if(aceso){
20       aceso=false;    // se estiver acesso entao deve apagar
21       digitalWrite(pinoLed,LOW);
22     }
23     else{             // senao apaga o led (considera que estava aceso)
24       aceso=true;
25       digitalWrite(pinoLed,HIGH);
26     }
27     tempoAnterior=millis();
28   }
29 }
```



Eventos de entrada com pushbutton

- Podemos mapear as seguintes condições em uma entrada:
  - Nível baixo;
  - Nível alto;
  - Borda de subida : rising
  - Borda de descida: falling
  - Mudança de nível.
- Em um acionamento de botão pode ocorrer trepidação (bounce)
- Como resolver?
  - Medindo tempo e garantir que ficou tempo suficiente no estado.





Implementado tratamento anti trepidação

- Os cabeçalhos

Exemplo_02_debounce | botao.cpp § | botao.h §

```
4  #ifndef __BOTAO__
5  #define __BOTAO__
6  /*Define os eventos a serem emitidos*/
7  #define EVENTO_SUBIDA 1
8  #define EVENTO_DESCIDA 2
9  extern int switchState;   // variavel de estado do botao
10 void iniciaTratadorBotao(int pino,long tempoDebon);
11 int processaBotao();
12 #endif
```

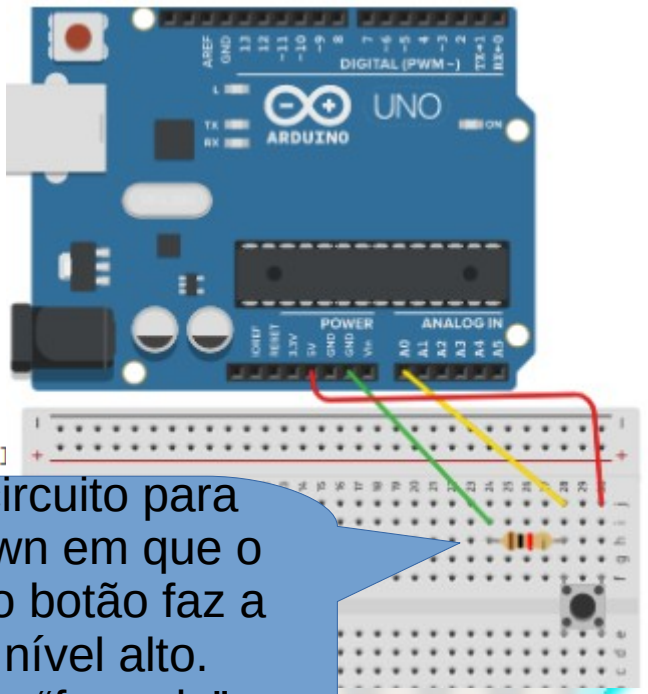
Implementado tratamento anti trepidação – setup e loop

```
2  #include "botao.h"
3  const int ledPin = 13; // pino de saida no led
4  void setup() {
5    Serial.begin(9600);
6    pinMode(ledPin,OUTPUT); // pino do led como saida
7    iniciaTratadorBotao(3,50);
8  }
9  void loop() {
10   int event=processaBotao();
11   if(switchState==HIGH) { // teste a chave esta acionada ou nao
12     digitalWrite(ledPin,HIGH); // chave acionada
13   } else {
14     digitalWrite(ledPin,LOW); // chave nao acionada
15   }
16   if(event==EVENTO_SUBIDA){
17     Serial.println("Botao pressionado");
18   }else if(event==EVENTO_DESCIDA){
19     Serial.println("Botao liberado");
20   }
21 }
```

Implementado tratamento anti trepidação – inicialização do módulo

Exemplo_02_debounce | botao.cpp § | botao.h §

```
1  #include "botao.h"
2  #include <arduino.h>
3  int switchPin; // pino de entrada do botao
4  int switchState;   // variavel de estado do botao
5  int lastSwitchState = LOW;
6  // variaveis para tratamento de trepidacao - debouncing
7  long lastDebounceTime = 0;
8  long debounceDelay;
9  void iniciaTratadorBotao(int pino,long tempoDebon){
10   switchPin=pino;
11   debounceDelay = tempoDebon;
12   switchState = 0;
13   lastSwitchState = LOW;
14   lastDebounceTime=millis();
15   pinMode(switchPin,INPUT); // pino de chave como entr
16 }
```

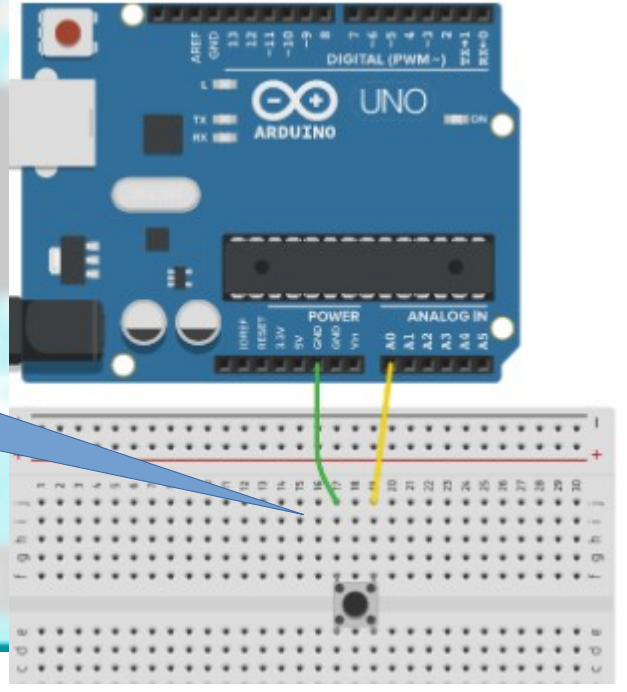


O esquema de circuito para código e o pulldown em que o pressionamento do botão faz a entrada ir para nível alto.
No entanto deve ser “forçado” ao terra quando não pressionado

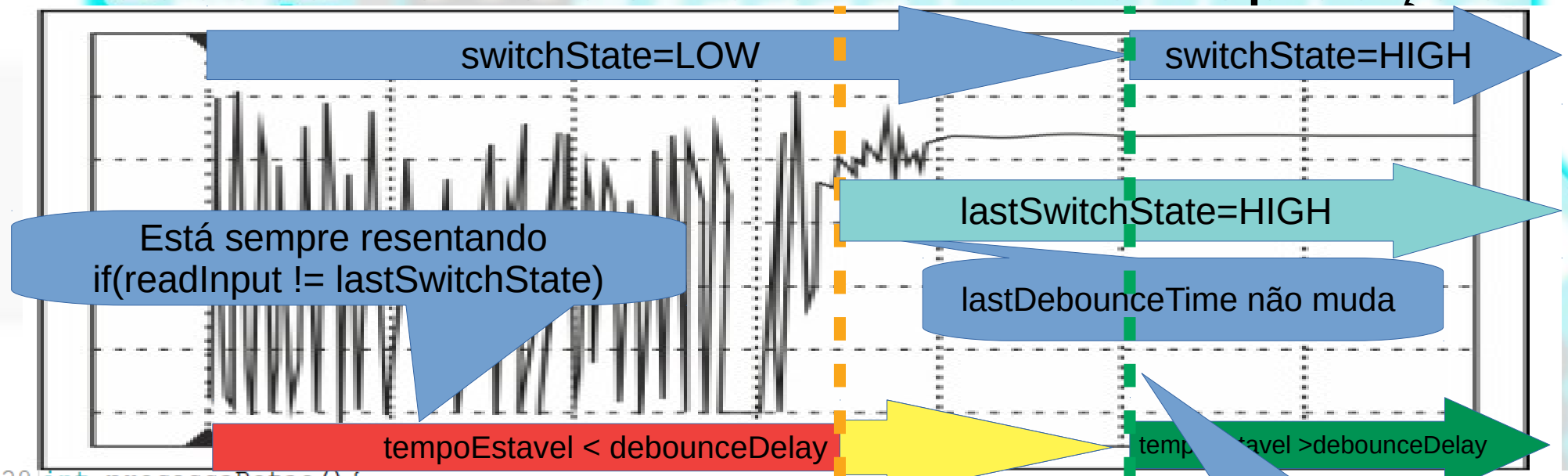
Função de inicialização do módulo alternativo

```
 9 void iniciaTratadorBotao(int pino,long tempoDebon){
10   switchPin=pino;
11   debounceDelay = tempoDebon;
12   switchState = 0;
13   lastSwitchState = LOW;
14   lastDebounceTime=millis();
15   pinMode(switchPin,INPUT); // pino de chave como como entrada e ligacao
16                             // interna para vcc por resistor
17 }
```

Neste caso a ligação é do tipo pullup em que um resistor interno ao uC é ligado ao 5V e o pressionamento do botão faz a tensão ir para baixo

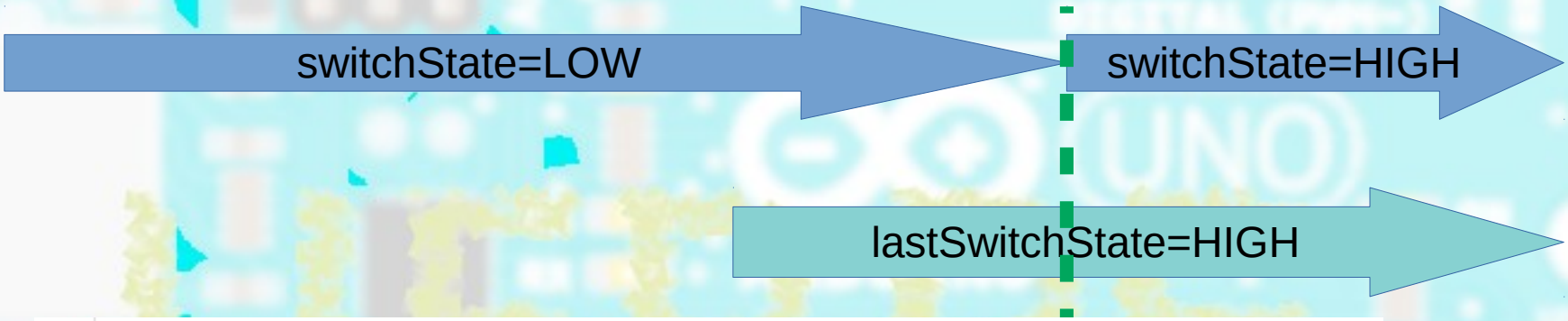


Processamento de tratamento de trepidação




```
28 int processaBotao(){
29   int evento=0;
30   int readInput = digitalRead(switchPin); // leitura do pino de botao
31   // se a leitura atual for diferente do que o ultimo valor de debounc
32   if(readInput != lastSwitchState){ // se a leitura atual for diferente
33     lastDebounceTime = millis();    // reseta o contador de debounce contando o tempo corrente
34   }
35   long tempoEstavel=millis()-lastDebounceTime;
36   if(tempoEstavel > debounceDelay){
37     evento=processaEvento(switchState,lastSwitchState);
38     switchState = readInput; // armazena o valor se o tempo de seguro para eliminacao do debounce
39   }
40   lastSwitchState = readInput; // armazena o ultimo estado lido para a proxima passagem no loop p
41   return evento;
42 }
```

Geração de evento

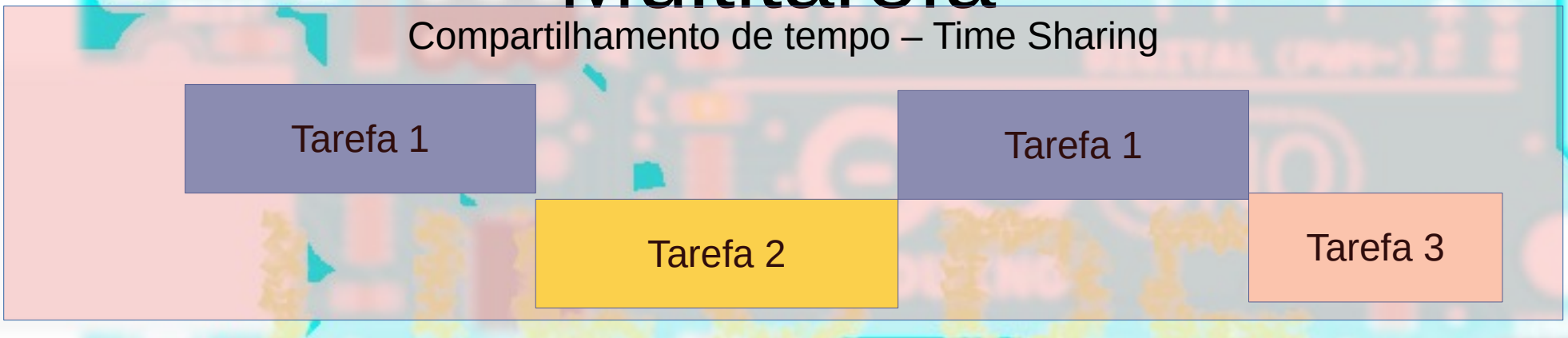



```
35   long tempoEstavel=millis()-lastDebounceTime;
36   if(tempoEstavel > debounceDelay){
37     evento=processaEvento(switchState,lastSwitchState);
38     switchState = readInput; // armazena o valor se o tempo
39   }
40   lastSwitchState = readInput; // armazena o ultimo estado
```

```
17 int processaEvento(int switchState,int lastSwitchState){
18   int evento=0;
19     if(switchState==HIGH && lastSwitchState==LOW){
20       evento = EVENTO_DESCIDA;
21     }
22     else if(switchState==LOW && lastSwitchState==HIGH){
23       evento = EVENTO_SUBIDA;
24     }
25     return evento;
26 }
```

Multitarefa

Compartilhamento de tempo – Time Sharing



- Implementação usando interrupções em temporizadores acionadas pelo uC.
  - Esta abordagem exige um cuidado adicional, pois dependendo da implementação e da biblioteca que usa. Ela afeta as funções como millis(), micros() e saídas de PWM. Além de dependerem do uC usado.
  - Apresenta uma vantagem por operarem sem atraso, pois executa exatamente no momento que foram especificadas.
  - A atualização do contador do millis() é realizada usando esta abordagem

Exemplo da biblioteca Timer para duas tarefas

Pisca led no pino 13 a cada 200ms

Executa a função takeReading a cada 1 segundo

read_A0_flashLED §

```
1  #include "Timer.h"
2  Timer t;
3  int pin = 13;
4  void setup(){
5    Serial.begin(9600);
6    pinMode(pin, OUTPUT);
7    t.oscillate(pin, 100, LOW);
8    t.every(1000, takeReading);
9  }
10 void loop(){
11   t.update();
12 }
13 void takeReading(){
14   Serial.println(analogRead(0));
15 }
```

```
112 void Timer::update(void)
113 {
114         unsigned long now = millis();
115         update(now);
116 }
```

Exercício proposto

- Construa usando dois botões (A e B) e 5 leds um circuito em que os leds acendam uma de cada vez da direita para esquerda inicialmente. Quando pressiona e solta o botão A faça com que os leds operem da esquerda para direita. Quando botão A é pressionada novamente faz com que volte a operar da direita para esquerda. O botão B deve fazer com que o tempo de transição de acendimento seja dobrada a cada pressionamento. O tempo de transição inicial e de 125 ms aceso e 125ms apagado e a transição mais lenta seja 2 segundos aceso e 2 segundos apagado. Construa 2 módulos em que uma trabalha os leds e outra trabalha os botões.

roberto.hiramatsu@ufrpe.br