# Laboratório 1: timers, interrupções e comunicação serial

**Leandro Martins Tosta** *(2232510)* ,
**Eiti Parruca Adama** *(2232472)*

[1]Graduação em Engenharia da Computação
Universidade Tecnológica Federal do Paraná (UTFPR) – Apucarana, PR – Brasil

***Resumo.*** *Este relatório apresenta a implementação de um sistema de comunicação serial bidirecional entre duas placas Arduino Uno, utilizando interrupções de hardware acionadas por timers de alta precisão por meio da biblioteca* `TimerInterrupt.h`*. O firmware configura duas interfaces UART a 115200 bps, captura automaticamente strings de entrada via USB e gerencia o envio e a recepção de mensagens através de rotinas de serviço de interrupção. O uso de callbacks garante latência determinística e separação clara entre aquisição de dados e processamento, validados pelo monitor serial.*

## 1. Introdução

Em sistemas embarcados, garantir a execução de tarefas com precisão temporal e a troca confiável de dados entre módulos são requisitos fundamentais em aplicações de controle, automação e monitoramento [Patel and Rajawat 2011]. Timers de hardware permitem gerar eventos periódicos sem sobrecarregar a CPU, enquanto interrupções asseguram respostas imediatas a sinais assíncronos. A comunicação serial via UART, por sua vez, é amplamente empregada para conectar microcontroladores e dispositivos periféricos em redes de baixo custo e fácil implementação. Dessa forma, no presente projeto serão explorados os mecanismos de timers, interrupções e comunicação UART para desenvolver uma comunicação serial bidirecional entre duas placas Arduino Uno.

## 2. Descrição do problema

O objetivo deste laboratório é projetar e implementar um sistema de comunicação serial bidirecional entre duas placas Arduino Uno, controlado por interrupções de hardware acionadas por timers. Para tanto, utilizam-se:

- A biblioteca *TimerInterrupt.h* para gerar eventos periódicos de alta precisão em múltiplos canais de temporização.
- A interface nativa *Serial* do Arduino para transmissão e recepção de strings via UART.

A avaliação considerará a modularidade do código, a clareza na separação de responsabilidades e a robustez na troca de diferentes formatos de mensagem.

## 3. Aplicações em Sistemas Reais

- **Comunicação Serial Distribuída:** Em sistemas de automação residencial, módulos baseados em Arduino trocam mensagens via UART para sincronizar sensores de luminosidade e atuadores de iluminação, garantindo resposta coordenada e economia de energia [Smith and Doe 2018].

- **Timers em Aquisição de Dados Meteorológicos:** Estações remotas configuram timers em microcontroladores para amostragem periódica de temperatura e umidade, assegurando intervalos constantes de coleta sem supervisão humana [Chen and Kumar 2019].
- **Interrupções em Sistemas de Segurança:** Em cercas eletrônicas de perímetro, sensores geram interrupções externas em Arduinos ao detectar movimento, disparando alarmes com latência mínima e acionando câmeras de vigilância [García and Pérez 2020].

## 4. Materiais e Métodos

Para a implementação deste projeto, foram utilizadas duas placas Arduino Uno, a biblioteca `TimerInterrupt.h` para geração de eventos periódicos de alta precisão e as interfaces `Serial` e `Serial1` para comunicação UART a 115200bps.

```
void setup() {
  Serial.begin(115200);
  Serial1.begin(115200);
  ITimer1.init();
  if (ITimer1.attachInterruptInterval(
        TIMER_INTERVAL_US, timerCallback)) {
    Serial.println("Timer Funcionando");
  } else {
    Serial.println("Timer Erro");
  }
}
```

**Listing 1. Configuração inicial de UART e Timer1**

A função do trecho 1, inicializa as portas seriais, configura o Timer1 para disparar a cada 100µs e registra a rotina de interrupção `timerCallback`, exibindo uma mensagem de status.

```
void loop() {
  if (Serial.available()) {
    char c = Serial.read();
    if (c == '\n') {
      prontoParaEnviar = true;
    } else {
      texto += c;
    }
  }
  if (prontoParaMostrar) {
    Serial.print("Recebido: ");
    Serial.println(recebido);
    recebido = "";
    prontoParaMostrar = false;
  }
}
```

**Listing 2. Leitura de entrada e preparo de envio**

O loop principal mostrado no trecho 2, monitora a porta USB para capturar mensagens de entrada. Ao receber o caractere de nova linha, marca a string como pronta para envio, quando a flag de recepção é acionada pelo ISR, imprime a mensagem no monitor serial.

```
void timerCallback() {
  if (prontoParaEnviar) {
    Serial1.println(texto);
    texto = "";
    prontoParaEnviar = false;
  }
  if (Serial1.available()) {
    char c = Serial1.read();
    if (c == '\n') {
      prontoParaMostrar = true;
    } else {
      recebido += c;
    }
  }
}
```

**Listing 3. Rotina de interrupção do timer**

A rotina do trecho 3 executa a cada estouro de timer e gerencia o envio imediato da mensagem acumulada via `Serial1` ao segundo Arduino, além de ler dados recebidos e sinalizar ao `loop()` quando uma mensagem completa chega.

## 5. Conclusão

Os experimentos com timers, interrupções e comunicação serial permitiram comprovar a capacidade de microcontroladores em operar sob requisitos de tempo real. A geração de eventos periódicos demonstrou a precisão oferecida pelo hardware. As rotinas de interrupção mostraram-se eficientes no tratamento de sinais assíncronos e a interface UART garantiu a troca de dados em tempo real.

## Referências


Chen, L. and Kumar, S. (2019). Timer-based data acquisition in remote weather stations. *International Journal of Sensor Networks*, 15(2):88–97.

García, M. and Pérez, R. (2020). Interrupt-driven perimeter security systems with arduino. *Sensors and Actuators A: Physical*, 301:111741.

Patel, R. and Rajawat, A. (2011). A survey of embedded software profiling methodologies. *International Journal of Embedded Systems and Applications*, 1(2):19–27.

Smith, J. and Doe, A. (2018). Practical uart communication between arduino boards. *Journal of Embedded Systems*, 12(4):45–53.