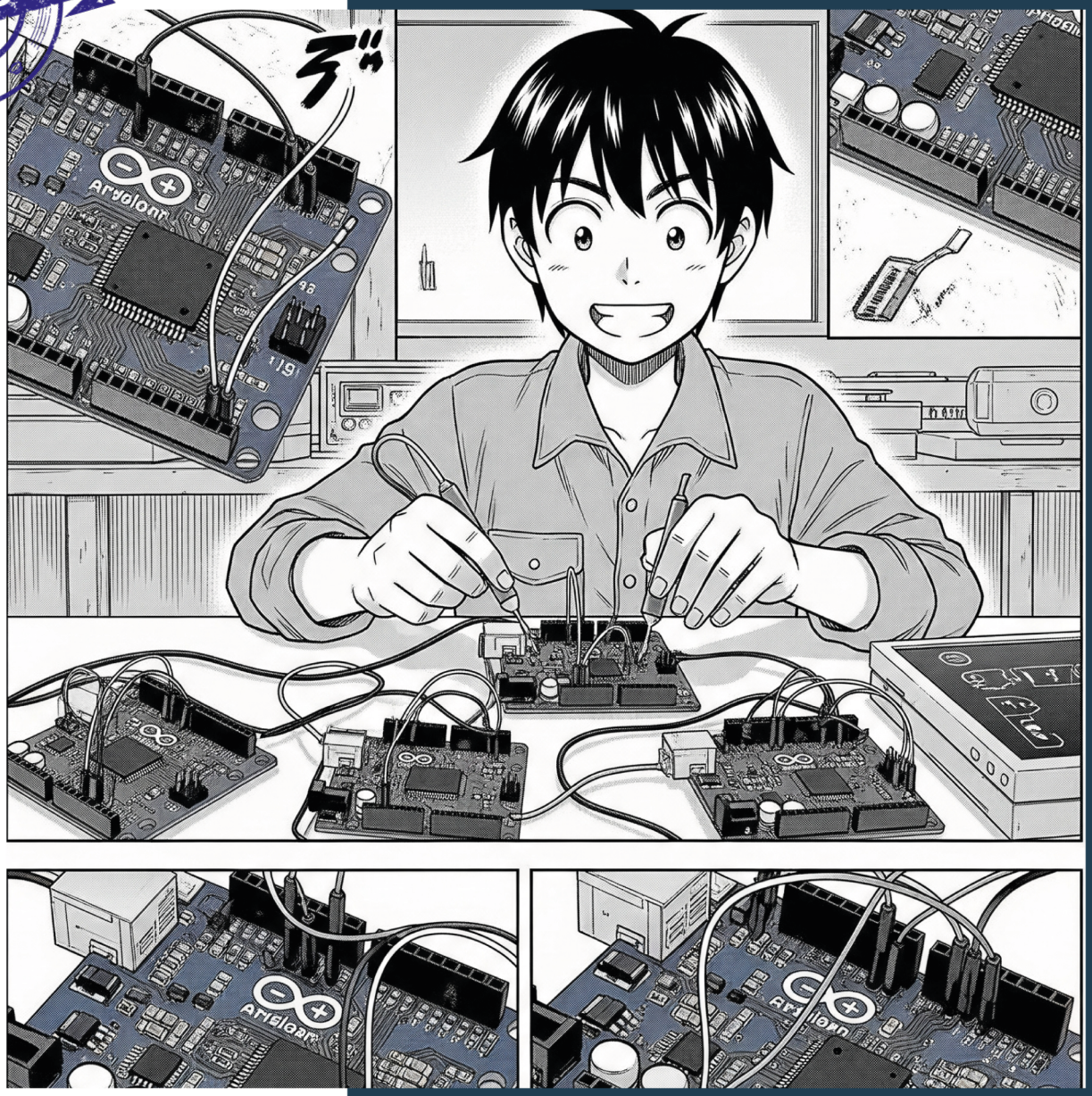
Robótica Educacional

Módulo 3

VERSÃO 2



Aula

34

Comunicação I2C - II

Diretoria de Tecnologia e Inovação

PARANÁ
GOVERNO DO ESTADO
SECRETARIA DA EDUCAÇÃO

GOVERNADOR DO ESTADO DO PARANÁ
Carlos Massa Ratinho Júnior

SECRETÁRIO DE ESTADO DA EDUCAÇÃO
Roni Miranda Vieira

DIRETOR DE TECNOLOGIA E INOVAÇÃO
Claudio Aparecido de Oliveira

COORDENADOR DE TECNOLOGIAS EDUCACIONAIS
Marcelo Gasparin

Produção de Conteúdo
Cleiton Rosa
Darice Alessandra Deckmann Zanardini

Validação de Conteúdo
Cleiton Rosa

Revisão Textual
Kellen Pricila dos Santos Cochinski

Projeto Gráfico e Diagramação
Edna do Rocio Becker

2025

Sumário

Aula 33 Comunicação I2C - II

Introdução

No mundo da Robótica e dos projetos com Arduino, frequentemente precisamos integrar diversos componentes – como sensores de temperatura, distância, acelerômetro e giroscópio, display OLED, módulo matriz de LED ou até mesmo outros microcontroladores. Conectar cada um desses componentes individualmente pode se tornar um problema em termos de organização, espaço e número de pinos disponíveis no Arduino.

É aí que o I2C pode entrar em ação!

As vantagens do I2C envolvem:

- Redução drástica do número de fios;
- Suporte a múltiplos dispositivos;
- Complexidades reduzidas;
- Comunicação bidirecional;
- Velocidade de comunicação adequada.

Nesta aula, vamos seguir, com novo projeto, pelo mundo da comunicação entre Arduinos.

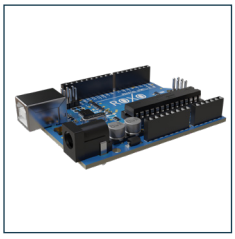
Aula 33 Comunicação I2C - II

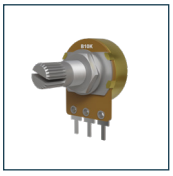
Objetivos desta aula

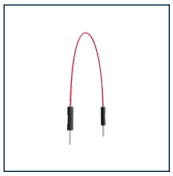
- Conhecer o protocolo de e comunicação I2C;
- Estabelecer a comunicação entre Arduinos.

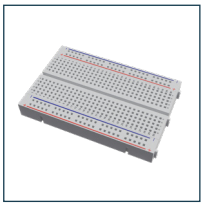
Lista de materiais

- 3 Arduinos Uno;
- 3 Protoboards;
- 1 Potenciômetro;
- 1 Display 7 segmentos com 4 dígitos TM1637;
- 1 Servomotor;
- Jumpers;
- Notebook ou computador;
- Cabo USB para carregamento dos códigos;
- Fonte de alimentação para cada Arduino.









Aula 33 Comunicação I2C - II

Roteiro da aula

1. Contextualização

No projeto da aula anterior, exploramos a comunicação I2C para o controle de Arduinos em um projeto o que compartilhou o mesmo endereço para executar a mesma ação: controle de LED.

Apesar de uma proposta simples, podemos compreender a sequência de eventos na comunicação I2C iniciada pelo Arduino mestre:

- **Start**: o mestre sinaliza o início da comunicação alterando o pino SDA de HIGH para LOW enquanto SCL está em HIGH.
- **Endereçamento**: o mestre envia o endereço do escravo desejado, seguido por um bit indicando se a operação será de leitura (HIGH) ou escrita (LOW).
- **ACK do escravo**: se o endereço corresponder, o escravo envia um bit de reconhecimento (ACK), colocando o pino SDA em LOW.
- **Transferência de dados**: o mestre envia os dados em partes (a cada 4 bits, espera ACK do escravo para enviar os bits restantes). O escravo sinaliza o recebimento correto mantendo SDA em LOW ou erro elevando SDA para HIGH.
- **Stop**: o mestre finaliza a comunicação alterando o pino SDA de LOW para HIGH enquanto SCL está em HIGH.

A base da comunicação é o mestre iniciar a conversa, chamando o escravo pelo endereço. O escravo confirma que está ali e os dados são trocados. Após, a comunicação é encerrada pelo mestre.

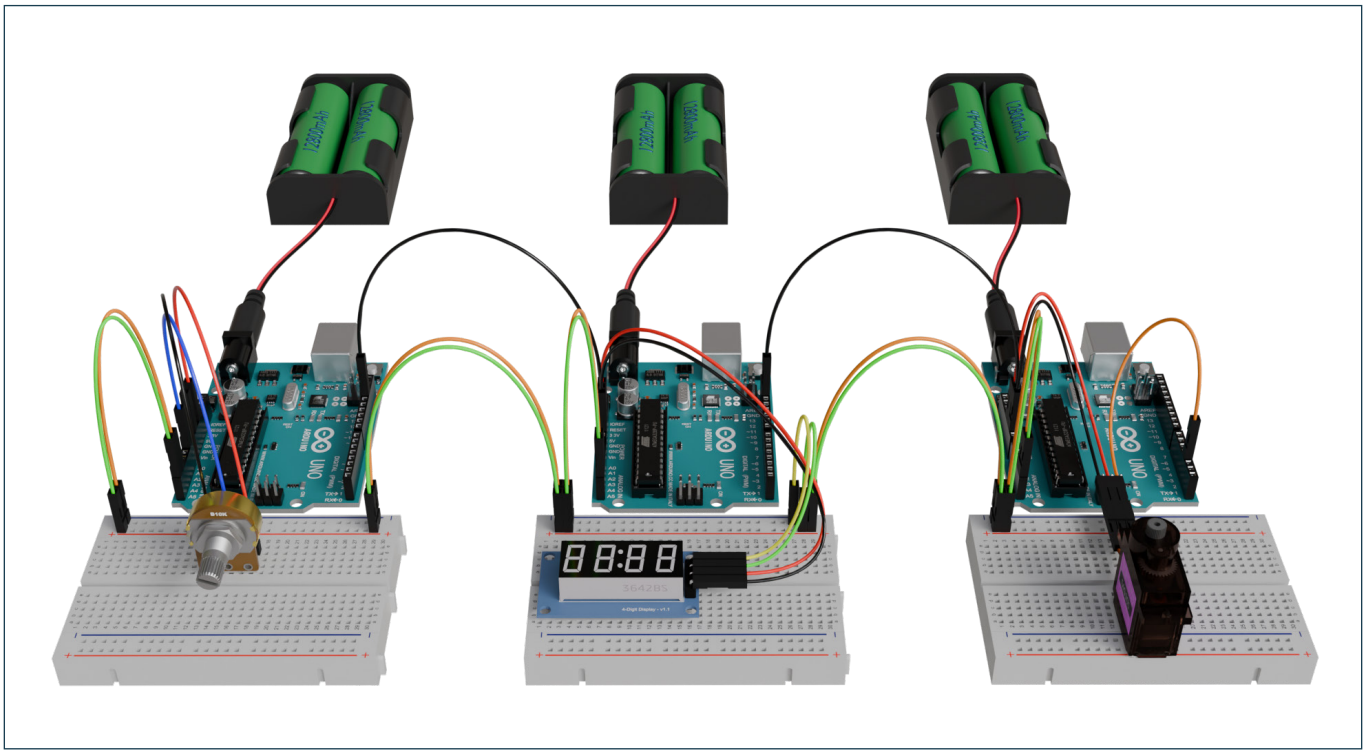
Vamos ao nosso **segundo projeto** para experimentarmos a comunicação I2C entre Aduinos? No primeiro projeto, realizado na **Aula 33 – Comunicação I2C [Parte I]**, utilizamos o mesmo endereço hexadecimal para os dois Arduinos escravos porque ambos possuíam a mesma montagem e executaram a mesma ação. Neste segundo projeto, aplicaremos endereços distintos aos Arduinos escravos para que o Arduino mestre faça o controle individual do display e do servomotor ligado a cada um deles.

Agora estamos elevando o nível da comunicação I2C, utilizando um Arduino mestre para ler um sensor (potenciômetro) e distribuindo essa informação para dois Arduinos escravos distintos, cada um com uma função específica: visualização em display e controle de um servomotor.

Aula 33 Comunicação I2C - II

Assim como na nossa primeira experiência com I2C, utilizaremos três Arduinos para a troca de dados – um será o dispositivo controlador (mestre) e os outros os periféricos (escravos), conectados à linha SCL e SDA do controlador ou, no caso do Arduino Uno, às portas analógicas A5 e A4.

Figura 1 - Conexão I2C entre Arduino mestre com potenciômetro e Arduinos escravos com display e servomotor



Fonte: Seed/DTI/CTE.

Confira, na **montagem 3D do projeto**, que o **Arduino mestre** possui um potenciômetro conectado à porta analógica **A0** (14) e agora, diferente do projeto I2C da aula anterior (na qual ambos os escravos compartilhavam mesma montagem, endereço e programação), cada Arduino escravo possui uma montagem específica, lembrando que compartilharão com o Arduino mestre as conexões **SDA** (A4), **SCL** (A5) e **GND** pela protoboard:

- **Arduino escravo 1**: um módulo display sete segmentos com 4 dígitos TM1637 com pino CLK conectado à porta 2 e pino DIO à porta 3.
- **Arduino escravo 2**: um servomotor conectado à porta 3.

Aula 33 Comunicação I2C - II

A ideia do projeto I2C desta aula é, pelo sensor do Arduino mestre, girar o eixo do potenciômetro. O Arduino escravo 1 recebe via I2C os valores analógicos do mestre e os imprime no display, enquanto o Arduino escravo 2 movimenta o servomotor para direita ou esquerda, recebendo via porta I2C as informações sobre o ângulo conforme o giro do Arduino mestre – para isso, remapearemos os valores analógicos do potenciômetro para transformá-los em ângulos de 0 a 180°.

Como na **montagem** utilizamos três Arduinos, é preciso programar cada um deles e alimentá-los durante a execução do projeto. Vamos lá?

Programação - Arduino mestre com potenciômetro

Iniciaremos pelo preâmbulo com a inclusão da biblioteca **Wire.h**, essencial para habilitar a comunicação I2C no Arduino. Na sequência, definimos os **endereços** I2C para os dois Arduinos escravos. O **Arduino escravo 1** (display) terá o **endereço 10**, e o **Arduino escravo 2** (servo) terá o **endereço 20**. É crucial que esses endereços correspondam à configuração nos respectivos códigos dos Arduinos escravos. Por fim, o pino analógico **A** do Arduino mestre é definido para leitura do potenciômetro e duas variáveis inteiras são declaradas:

- **dadoAnalogico**, para armazenar a leitura analógica do potenciômetro (0 a 1023).
- **angulo**, para armazenar o valor mapeado para o ângulo do servo (0 a 180).

```
/* Arduino mestre */
#include <Wire.h>          /* Inclusão da biblioteca. */
#define endereco1 10       /* Endereço do escravo 1. */
#define endereco2 20       /* Endereço do escravo 2. */
int potPin = A0;           /* Pino do potenciômetro. */
int dadoAnalogico, angulo; /* Variáveis para armazenamento de dados do
potenciômetro e ângulo do servomotor. */
```

Aula 33 Comunicação I2C - II

No **void setup()** da programação do Arduino mestre, adicionamos a função **Wire.begin()** para inicializar o Arduino como o mestre no barramento I2C. Note que não há especificação de um endereço para o mestre, pois no modo mestre ele inicia a comunicação.

```
void setup() {
  Wire.begin();  /* Inicia como mestre. */
}
```

Finalizamos a programação do Arduino mestre com a função **void loop()**, atribuindo a leitura de dados analógicos pelo potenciômetro e outros três momentos importantes para nossa comunicação I2C:

- **Envio para o Arduino escravo 1 (display):**
  - **Wire.beginTransmission(endereco1)** inicia uma transmissão de dados para o Arduino escravo com o endereço **10**.
  - **Wire.write(dadoAnalogico / 256)** envia o byte mais significativo (parte alta) do valor **dadoAnalogico** – a divisão inteira por 256 desloca os 8 bits mais altos para a posição dos 8 bits mais baixos.
  - **Wire.write(dadoAnalogico % 256)** envia o byte menos significativo (parte baixa) do valor **dadoAnalogico**. O operador módulo (%) retorna o resto da divisão por 256, que corresponde aos 8 bits mais baixos.
  - **Wire.endTransmission()** finaliza a transmissão para o Arduino escravo 1. É importante notar que o Arduino escravo 1 precisará receber esses dois bytes e combiná-los para reconstruir o valor analógico original.

- **Mapeamento para o ângulo do servo:**
  - A função **map()** é usada para remapear o valor analógico lido (que varia de 0 a 1023) para uma faixa de 0 a 180 graus, que é a faixa típica de controle para um servomotor.

Aula 33 Comunicação I2C - II

- **Envio para o Arduino escravo 2 (servomotor)**:
  - **Wire.beginTransmission(endereco2)** inicia uma transmissão de dados para o Arduino escravo com o endereço 20.
  - **Wire.write(angulo)** envia o valor de **angulo** (que agora está na faixa de 0 a 180) para o Arduino escravo 2.
  - **Wire.endTransmission()** finaliza a transmissão para o Arduino escravo 2. O Arduino escravo 2 deverá receber este byte e usá-lo para definir a posição do servomotor.

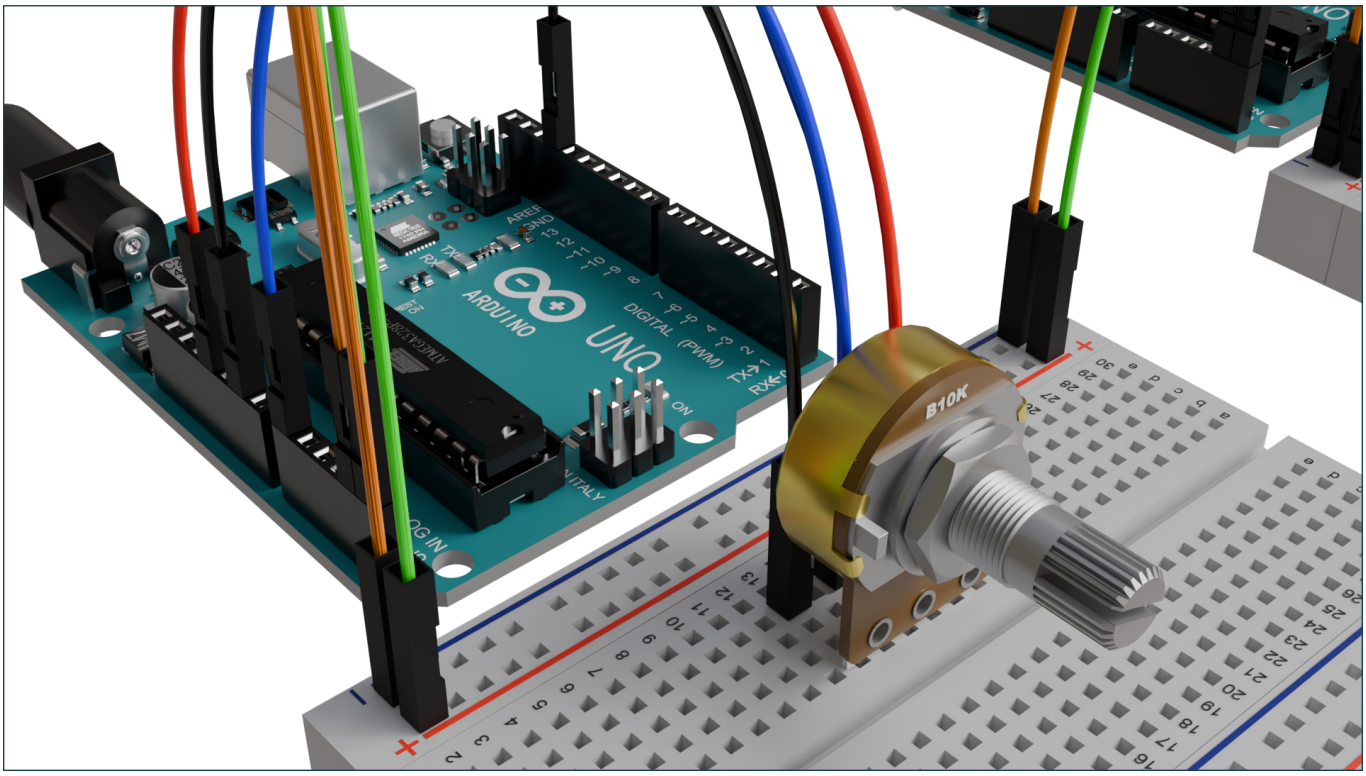
Finalizamos a **void loop()** com um pequeno delay de 50 milissegundos, prática que já adotamos para evitar sobrecarregar o barreamento I2C e o processador do Arduino.

```
void loop() {
  dadoAnalogico = analogRead(potPin);  /* Lê o valor do potenciômetro. */
  /* Enviar para o primeiro escravo. */
  Wire.beginTransmission(endereco1);
  Wire.write(dadoAnalogico / 256);  /* Parte alta. */
  Wire.write(dadoAnalogico % 256);  /* Parte baixa. */
  Wire.endTransmission();
  angulo = map(dadoAnalogico, 0, 1023, 0, 180);  /* Mapeia o valor para o
ângulo do servo (0 a 180). */
  /* Enviar para o segundo escravo. */
  Wire.beginTransmission(endereco2);
  Wire.write(angulo);  /* Envia o ângulo para o escravo. */
  Wire.endTransmission();
  delay(50);  /* Pequena pausa para não sobrecarregar. */
}
```

Aula 33 Comunicação I2C - II

Confira como fica a **programação do Arduino mestre** e carregue-a para o Arduino correspondente.

Figura 2 - Arduino mestre com potenciômetro



Fonte: Seed/DTI/CTE.

O bacana desta **programação do Arduino mestre** é que avançamos na comunicação I2C com Arduinos escravos distintos, cada um recebendo dados específicos para sua função, com o Arduino mestre se concentrando na leitura do sensor e na distribuição dos dados processados para os Arduinos escravos.

Para o sistema funcionar corretamente, os códigos dos Arduinos escravos (para o display e o servomotor) devem estar configurados para receber e interpretar os dados da maneira esperada (reconstruindo o valor analógico no Arduino escravo 1 e utilizando o ângulo diretamente no Arduino escravo 2), além de possuírem os endereços I2C correspondentes (**10** e **20**).

Seguiremos programando os Arduinos escravos e, ao contrário do projeto da aula anterior, no qual ambos Arduinos escravos compartilhavam mesmo endereço, montagem e objetivos, no projeto atual cada Arduino escravo possuirá sua própria programação devido às montagens e endereços específicos. Vamos lá?

Aula 33 Comunicação I2C - II

Programação - Arduino escravo 1 com display

Iniciamos o preâmbulo com a inclusão das duas bibliotecas necessárias: **Wire.h** para a comunicação I2C e **TM1637Display.h** para controlar o display de 7 segmentos com 4 dígitos. Na sequência, definimos os pinos digitais 2 e 3 do Arduino escravo 1 para a conexão com os pinos CLK e DIO do dispay. O endereço I2C deste Arduino escravo 1 é definido como 10 e esse endereço deve corresponder ao endereço para o qual o Arduino mestre envia os dados destinados a este display.

Precisamos criar também o objeto de controle **display** para interagir com a biblioteca TM1637Display, utilizando os pinos CLK e DIO definidos. E, por fim, declaramos a variável **dadoAnalogico** para armazenar o valor analógico recebido do Arduino mestre.

```
/* Escravo 1. */
#include <Wire.h>          /* Biblioteca de controle I2C. */
#include <TM1637Display.h> /* Biblioteca de controle do display. */
/* Define os pinos de conexão do TM1637. */
#define CLK 2  /* Pino de clock do TM1637. */
#define DIO 3  /* Pino de dados do TM1637. */
#define endereco1 10 /* Definição do endereço do escravo 1 (1 a 255). */
TM1637Display display(CLK, DIO); /* Cria o objeto de controle do dis-
play. */
int dadoAnalogico;  /* Variável que receberá os dados analógicos via
I2C. */
```

Para o **void setup()** do **Arduino escravo 1**, adicionamos três funções:

- **Wire.begin(endereco1)** inicializa o Arduino como um escravo no barramento I2C com o endereço 10, deixando-o pronto para receber dados enviados para esse endereço.
- **Wire.onReceive(executar)** registra a função **executar** como a função de *callback* que será chamada automaticamente quando o Arduino escravo 1 receber dados do Arduino mestre.
- **display.setBrightness(0)** define o brilho do display TM1637.

Aula 33 Comunicação I2C - II

```
void setup() {
  Wire.begin(endereco1);        /* Inicia como escravo no endereço defi-
nido. */
  Wire.onReceive(executar);     /* Define função de recepção. */
  display.setBrightness(0);     /* Define brilho máximo do display (0 a
15). */
      }
```

A função **void loop()** é executada continuamente e, a cada iteração, pega o valor armazenado na variável **dadoAnalogico** e o exibe no display como um número decimal pela função **display.showNumberDec()**.

```
void loop() {
  /* Exibe o valor lido do potenciômetro no display de 4 dígitos. */
    display.showNumberDec(dadoAnalogico);
      }
```

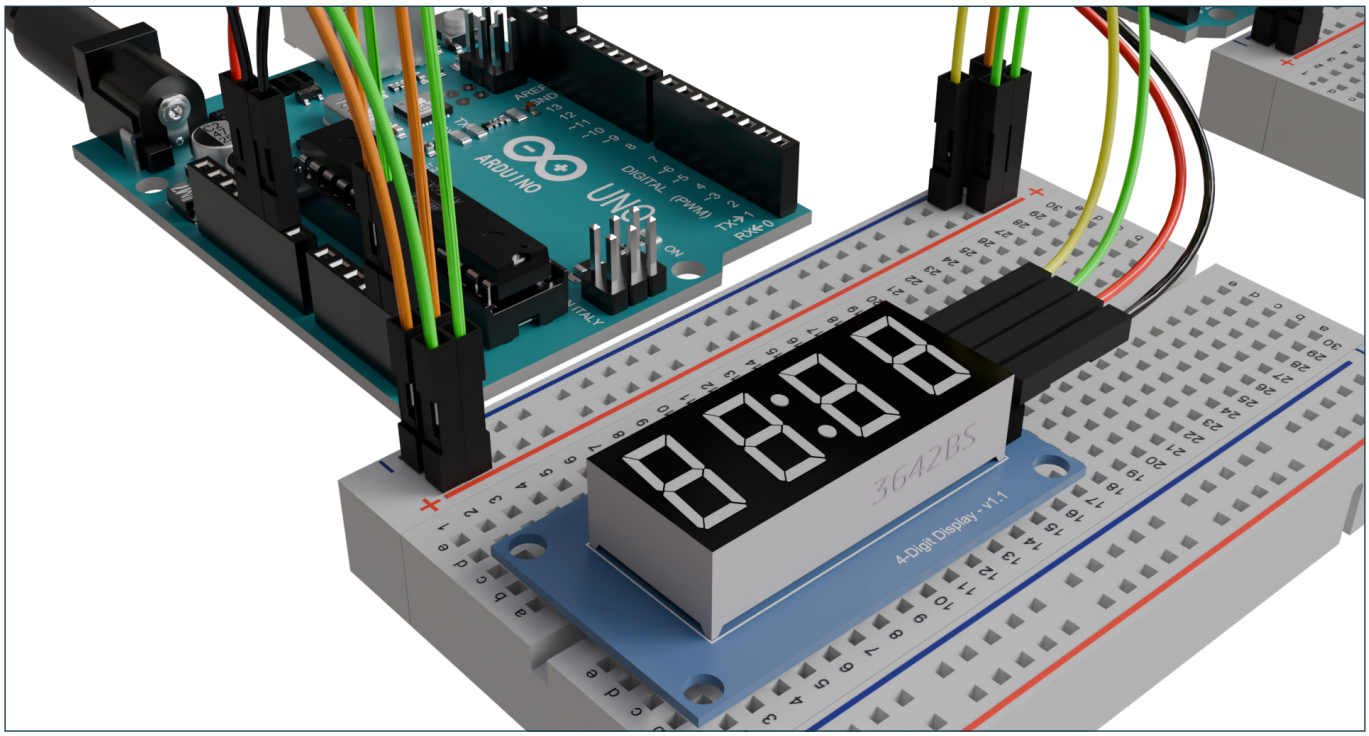
Finalizamos a programação do Arduino escravo 1 com a criação da função **void executar(int bytes)**. Essa função é chamada automaticamente quando o Arduino escravo 1 recebe dados do Arduino mestre e possui a seguinte estrutura:

- **if (Wire.available())** verifica se há bytes disponíveis para leitura no buffer I2C.
- **int parte1 = Wire.read()** lê o primeiro byte recebido (que o Arduino mestre enviou como a parte alta do **dadoAnalogico)** e o armazena na variável **parte1**.
- **int parte2 = Wire.read()** lê o segundo byte recebido (a parte baixa do **dadoAnalogico)** e o armazena na variável **parte2**.
- **dadoAnalogico = parte1 * 256 + parte2** reconstrói o valor analógico original combinando os dois bytes. O byte mais significativo (**parte1**) é multiplicado por 256 e somado ao byte menos significativo (**parte2**).

```
/* Função chamada ao receber dados via I2C. */
void executar(int bytes) {
  if (Wire.available()) {        /* Se receber dados via I2C, faça... */
    int parte1 = Wire.read();    /* Armazena na variável o byte mais sig-
nificativo. */
    int parte2 = Wire.read();    /* Armazena na variável o byte menos
significativo. */
    dadoAnalogico = parte1 * 256 + parte2; /* Junta os bytes em um valor
inteiro. */
  }
}
```

Confira como fica a **programação do Arduino escravo 1** e carregue-a para o Arduino correspondente.

Figura 3 - Arduino escravo 1 com display



Fonte: Seed/DTI/CTE.

Aula 33 Comunicação I2C - II

A **programação do Arduino escravo 1** inicializa a comunicação I2C no endereço **10** e configura o display conectado aos pinos 2 e 3. Ao receber dados do Arduino mestre através da função de *callback* **executar**, o código lê dois bytes sequenciais, combina-os para reconstruir um valor inteiro representando a leitura analógica do potenciômetro, e então exibe esse valor numericamente no display de 4 dígitos dentro do loop principal.

Vamos agora à última programação para que o servomotor se movimente e confirme esses valores.

Programação - Arduino escravo 2 com servomotor

No preambulo da programação do Arduino escravo 2, responsável pelo controle do servomotor neste projeto I2C, incluímos as bibliotecas **Wire.h** para a comunicação IC e **Servo.h** para controlar o servomotor. Na sequência, o pino digital **3** do **Arduino escravo 2** é definido para conexão com o pino de sinal do servomotor e o endereço I2C deste Arduino escravo 2 é definido como **20**, lembrando que esse endereço deve corresponder ao endereço para o qual o Arduino mestre envia os dados de controle do servomotor.

Finalizamos o preâmbulo com um objeto chamado **meuServo**, criado para interagir com a biblioteca **Servo** e permitir o controle do servomotor, e a variável **angulo**, declarada para armazenar o valor do ângulo recebido do Arduino mestre.

```
/* Escravo 2. */
#include <Wire.h>  /* Biblioteca de controle I2C. */
#include <Servo.h> /* Biblioteca de controle de servomotores. */
#define servoPin 3   /* Pino onde o servo está conectado. */
#define endereco2 20 /* Definição do endereço do escravo 2 (1 a 255). */
Servo meuServo;      /* Cria o objeto servo. */
int angulo;          /* Variável que receberá os ângulos via I2C. */
```

Aula 33 Comunicação I2C - II

No **void setup()**, adicionamos três funções:

- **Wire.begin(endereco2**) inicializa o Arduino como um escravo no barramento I2C com o endereço **20**, deixando-o pronto para receber dados enviados para esse endereço.
- **Wire.onReceive(executar)** registra a função **executar** como a função de *callback* que será chamada automaticamente quando o Arduino escravo 2 receber dados do Arduino mestre.
- **meuServo.attach(servoPin)** associa o objeto **meuServo** ao pino digital **3**, configurando esse pino para controlar o servomotor.

```
void setup() {
  Wire.begin(endereco2);          /* Inicia como escravo no endereço defini-
do. */
  Wire.onReceive(executar);       /* Define função de recepção. */
  meuServo.attach(servoPin);      /* Conecta o servo ao pino. */
}
```

O **void loop()** fica vazio. Isso indica que toda a lógica de controle do servomotor é acionada pela recepção de dados I2C na função **executar**, última etapa da programação do Arduino escravo 2.

```
void loop() {

}
```

Aula 33 Comunicação I2C - II

Precisamos criar a função **void executar()**, chamada automaticamente quando o Arduino escravo 2 receber dados do Mestre. Sua constituição envolve três etapas:

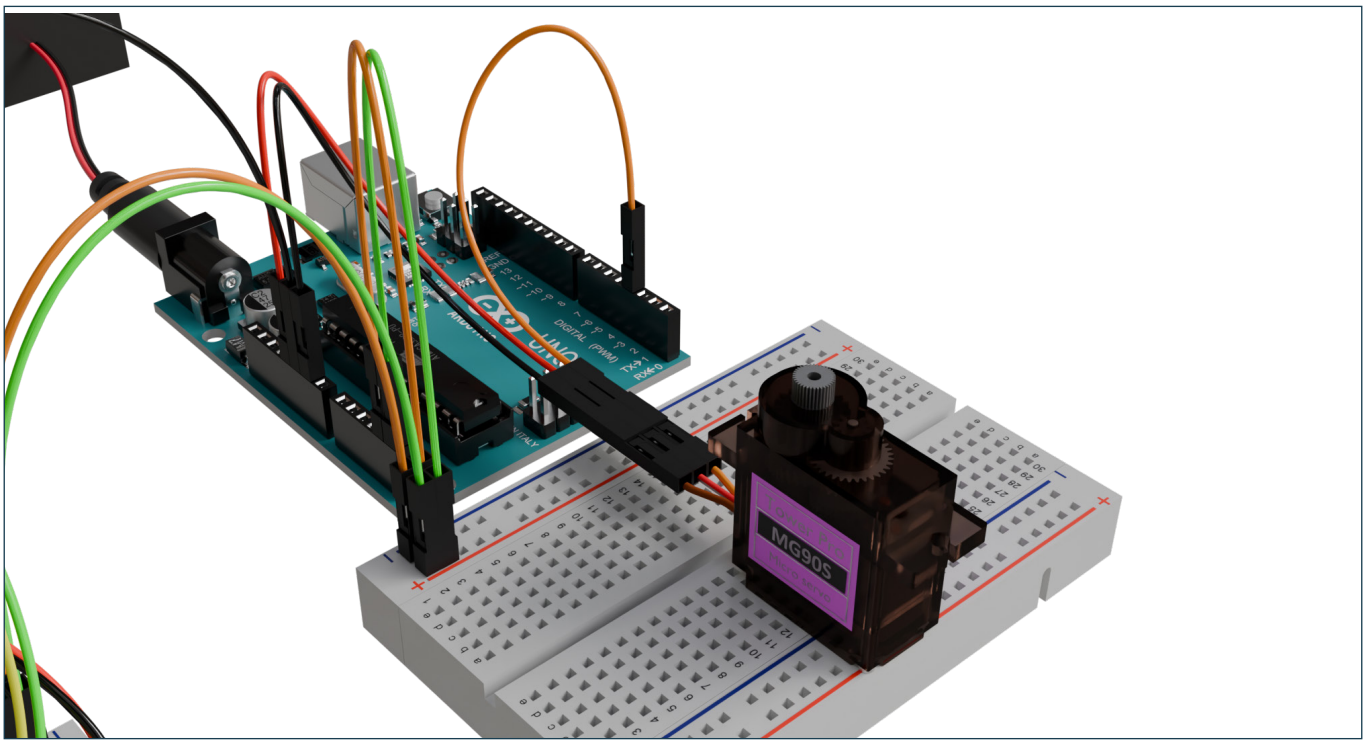
- **if (Wire.available())** verifica se há bytes disponíveis para leitura no buffer I2C.
- **angulo = Wire.read()** lê o primeiro (e nesse caso, único) byte recebido do Arduino mestre, que representa o ângulo mapeado para o servomotor, e o armazena na variável **angulo**.
- **meuServo.write(angulo)** utiliza o valor recebido na variável **angulo** para definir a posição do servomotor.

```
/* Função chamada ao receber dados via I2C. */

void executar(int bytes) {

  if (Wire.available()) {  /* Se receber dados via I2C, faça... */

        angulo = Wire.read();  /* Recebe os dados enviados pelo Mestre e
armazena na variável angulo. */

        meuServo.write(angulo); /* Move o servo para o ângulo correspon-
dente. */

  }

}
```

Aula 33 Comunicação I2C - II

Confira como fica a **programação do Arduino escravo 2** e carregue-a para o Arduino correspondente, posicionando a pazinha do servo.

Figura 4 - Arduino escravo 2 com servomotor



Fonte: Seed/DTI/CTE.

A **programação do Arduino escravo 2** configura o dispositivo para escutar no endereço I2C **20** e, ao receber um byte de dado do Arduino mestre, interpreta esse byte como um ângulo para um servomotor conectado ao pino 3. Através da função de *callback* **executar**, o valor recebido é imediatamente utilizado para posicionar o servo, sem nenhuma ação adicional sendo realizada no loop principal do Arduino.

Finalizamos o segundo projeto I2C entre Arduinos! Como destacamos na programação de comunicação I2C, verifique que os endereços definidos aos Arduinos escravos devem corresponder a cada endereço para os quais o Arduino mestre envia os dados do display e do servomotor. Assim, cada Arduino escravo consegue receber a comunicação e executar sua ação de acordo com as programações específicas.

Revise como ficou cada programação – código de **programação Arduino mestre**, código de **programação Arduino escravo 1** e código de **programação Arduino escravo 2** – e experimente seu protótipo!

Desafios:

Que tal ampliar a programação com outra função da biblioteca Wire para verificar se a transmissão foi bem-sucedida? Retorne à Aula 33 - Comunicação I2C [Parte I] para relembrar as demais funções da biblioteca.

E se...

O protótipo não funcionar?

- Confira as portas SDA (A4) e SCL (A5) utilizadas para a comunicação I2C no Arduino Uno.
- Verifique se não há inversão entre as conexões SDA e SCL entre as placas e que o GND das três está interligado.
- Confira os endereços atribuídos aos dispositivos periféricos (escravos).
- Verifique a programação e alimentação de cada um dos Arduinos.

3. Feedback e finalização

Como foi usar um potenciômetro no Arduino mestre para controlar tanto um display quanto um servomotor em dois Arduinos escravos diferentes via I2C? Este projeto é um pouco mais complexo que o anterior, de controle de LEDs, e é extremamente recompensador ao ver os três Arduinos trabalhando em conjunto, demonstrando a flexibilidade e poder desse protocolo!

REFERÊNCIAS

ARDUINO. **Documentação de Referência da Linguagem Arduino**. Disponível em: https://www.arduino.cc/reference/pt/. Acesso em: 27 mai. 2024.

ARDUINO FACTORY. **Arduino I2C.** Disponível em: https://arduinofactory.com/arduino-i2c/. Acesso em: 14 abr. 2025.

AUTOCORE ROBÓTICA. **Conhecendo o protocolo I2C com Arduino**. Disponível em: https://autocorerobotica.blog.br/conhecendo-o-protocolo-i2c-com-arduino/. Acesso em: 15 ago. 2023.

DIRETORIA DE TECNOLOGIAS E INOVAÇÃO (DTI)

COORDENAÇÃO DE TECNOLOGIAS EDUCACIONAIS (CTE)

EQUIPE ROBÓTICA PARANÁ

- Adilson Carlos Batista
- Ailton Lopes
- Andrea da Silva Castagini Padilha
- Cleiton Rosa
- Darice Alessandra Deckmann Zanardini
- Edna do Rocio Becker
- Enzo Enrico Giacomini Piolla
- Kellen Pricila dos Santos Cochinski
- Marcelo Gasparin
- Michele Serpe Fernandes
- Michelle dos Santos
- Roberto Carlos Rodrigues
- Sandra Aguera Alcova Silva
- Viviane Dziubate Pittner

Os materiais, aulas e projetos da "Robótica Paraná", foram produzidos pela Coordenação de Tecnologias Educacionais (CTE), da Diretoria de Tecnologia e Inovação (DTI), da Secretaria de Estado da Educação do Paraná (SEED), com o objetivo de subsidiar as práticas docentes com os estudantes por meio da Robótica.
Este material foi produzido para uso didático-pedagógico exclusivo em sala de aula.