

Engenharia de Prompt: Assistente professor de programação de microcontroladores © 2024
by Lucas dos Santos Venâncio, Louis Brommelstroet is licensed under [CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/)



Membros da equipe:

RA	Nome	Curso de cada membro (SISTEL, Eng. Mecatrônica, etc.)
2128004	Lucas dos Santos Venâncio	BSI
2127997	Louis Brommelstroet	BSI

Cliente externo (se houver cliente externo é necessário preencher uma ficha de autorização de imagem)

CPF	Nome
	Manuela de Almeida Gomes
Idade	Observações

Título do projeto:

Assistente professor de programação de microcontroladores

Descrição resumida do projeto (breve contexto, descrição do problema que se deseja resolver, objetivos, metodologia, resultados e conclusão – máximo 500 palavras)

<p>Descrição do problema: Identificamos uma dificuldade no aprendizado de linguagens de programação, especificamente no contexto de disciplinas de microcontroladores. Na disciplina cursada por Manuela, se utiliza a linguagem C, que acreditamos ser uma boa linguagem para tratar nos modelos de linguagem, pela vasta quantidade de recursos disponíveis sobre a linguagem e também por estar disponível no treinamento destes modelos. Assim, consideramos que estes modelos, auxiliados à engenharia de prompt, podem contribuir para um aprendizado interativo da linguagem C.</p> <p>Objetivos do projeto: Criar prompts que funcionam como assistentes professores que criam exemplos, exercícios para completar de programação em C, ou até interativamente guiar os estudantes possibilitando dicas e exemplos incompletos, agindo como um professor em vez de meramente uma ferramenta para completar ou criar código.</p> <p>Metodologia: Levando em conta este problema, iremos inicialmente considerar uma exploração inicial do conteúdo e criação de prompts básicas improvisadas. Então, iremos avaliar a saída desses prompts, e adicionar um contexto maior ao mesmo tempo que avaliamos esses</p>

prompts, utilizando as técnicas aprendidas na disciplina. Por fim, utilizaremos também prompts criados pelo próprio Gemini e avaliaremos esses prompts.

Prompt 1:

Utilizando os conhecimentos sobre microcontroladores programados na linguagem C, crie um passo a passo para os seguintes requisitos, sem exemplifica-los diretamente:

- a) Utilizando o operador de deslocamento binário e o comando FOR, faça uma lógica onde uma sequência de 8 LED's seja acesa do tipo "bola pingando". Monte o circuito no proteus e no protoboard utilizando o PORT que desejar
- b) Piscar 3 vezes os 8 LED's de forma a mostra que o microcontrolador e o circuito estão funcionando.
- c) Iniciar o acendimento dos LED's no modo "bola pingando", como se a bola fosse solta a uma certa distância e começasse a pingar, perdendo altura a cada novo quique.
- d) Antes do ciclo da bola pingando reiniciar, coloque um atraso maior no código para mostrar ao usuário que o sistema irá ser reiniciado. Você pode projetar outras métodos de reinício caso queira.
- e) O programa deve ficar em "loop" infinito.

Resposta:

Passo a Passo para Criar o Código "Bola Pingando" em 8 LEDs

1. Configuração do Ambiente:

- **Microcontrolador:** Escolha o microcontrolador que você irá utilizar (ex: PIC16F877A, Arduino Uno).
- **Compilador:** Baixe e configure o compilador C compatível com o microcontrolador escolhido (ex: MPLAB X para PIC, Arduino IDE).
- **Ambiente de Simulação (Opcional):** Utilize um software de simulação (ex: Proteus) para testar o código.

2. Criação do Projeto:

- **Compilador:** Crie um novo projeto e configure-o para o microcontrolador escolhido.
- **Proteus (Opcional):** Crie um novo projeto e adicione o microcontrolador e os LEDs ao circuito.

3. Definição de Pinos e Portas:

- **Código:** Defina a porta de saída (ex: PORTB) e os pinos individuais (ex: RB0, RB1, RB2, etc.) que controlarão os LEDs.
- **Proteus (Opcional):** Conecte os LEDs aos pinos do microcontrolador definidos no código.

4. Definição de Variáveis e Constantes:

- **Código:** Declare uma variável para controlar a altura da bola, que irá variar de 8 a 1.
- **Código:** Declare uma variável para controlar o atraso, calculado com base na altura da bola.

- **Código:** Defina uma constante para o atraso base (em milissegundos), que será multiplicada pela altura para determinar o atraso total.

5. Criação da Função main:

- **Código:** Implemente a função main como o ponto de partida do programa.

6. Inicialização da Porta de Saída:

- **Código:** Configure a porta de saída definida como saída.

7. Animação de Inicialização:

- **Código:** Crie um loop para piscar todos os LEDs 3 vezes, com um atraso entre cada piscada (utilizando `__delay_ms` ou uma função de atraso similar).

8. Loop Infinito:

- **Código:** Utilize um `while(1)` para garantir que o programa execute continuamente.

9. Implementação da Animação "Bola Pingando":

- **Código:** Crie um loop externo para controlar a altura da bola (de 8 a 1).
- **Código:** Crie um loop interno para acender os LEDs da altura atual.
- **Código:** Utilize o operador de deslocamento binário (`<<`) para acender o LED correspondente à altura.
- **Código:** Calcule o atraso com base na altura da bola e a constante de atraso base. Utilize `__delay_ms` para implementar o atraso.
- **Código:** Apague todos os LEDs após acender os LEDs da altura atual.

10. Reinício do Ciclo:

- **Código:** Antes de reiniciar o loop da bola pingando, adicione um atraso maior para indicar ao usuário que o ciclo será reiniciado (ex: acenda todos os LEDs por 1 segundo, apague-os, aguarde 500ms).
- **Código:** Você também pode implementar um método de reinício diferente, como um botão que o usuário pressione para reiniciar o ciclo.

11. Compilação e Gravação do Código:

- **Compilador:** Compile o código para gerar o arquivo binário.
- **Compilador:** Grave o arquivo binário no microcontrolador.

12. Teste do Código:

- **Proteus (Opcional):** Utilize o simulador para verificar o funcionamento do código.
- **Protoboard:** Conecte o microcontrolador ao protoboard e teste se a animação da bola pingando funciona corretamente nos LEDs.

Lembre-se de adaptar este passo a passo para o seu microcontrolador específico e sua configuração de hardware.

Prompt 2:

Crie tópicos a serem estudados para que um aluno consiga entender melhor o uso da linguagem C na programação de microcontroladores. Não exemplifique códigos

Resultados:

Tópicos para Estudo de Linguagem C em Microcontroladores:

1. Fundamentos da Linguagem C:

- Sintaxe e estrutura básica da linguagem C
- Tipos de dados (int, char, float, etc.)
- Operadores (aritméticos, relacionais, lógicos, etc.)
- Estruturas de controle de fluxo (if, else, for, while, switch)
- Funções: definição, chamada e passagem de parâmetros

2. Conceitos de Microcontroladores:

- Arquitetura básica de um microcontrolador
- Tipos de microcontroladores (PIC, AVR, ARM)
- Recursos de um microcontrolador (memória, portas I/O, periféricos)
- Comunicação serial (UART, SPI, I2C)
- Interrupções

3. Programação de Microcontroladores em C:

- Incluindo arquivos de cabeçalho (header files) específicos do microcontrolador
- Configurando pinos de I/O como entrada ou saída
- Controlando LEDs, motores e outros atuadores
- Lendo dados de sensores
- Utilizando temporizadores e interrupções
- Comunicação serial com um computador ou outros dispositivos

4. Técnicas Avançadas de Programação:

- Ponteiros e alocação de memória
- Estruturas de dados (arrays, structs, etc.)
- Arquivos e manipulação de arquivos
- Multithreading e tarefas em tempo real

5. Bibliotecas e Frameworks:

- Bibliotecas específicas do microcontrolador (ex: Arduino, HAL)
- Bibliotecas para periféricos (ex: LCDs, sensores)
- Frameworks para facilitar o desenvolvimento (ex: FreeRTOS)

6. Ferramentas de Desenvolvimento:

- Compiladores C (ex: MPLAB X, Arduino IDE)
- Simuladores (ex: Proteus)
- Gravadoras de firmware (ex: PICkit3)

7. Boas Práticas de Programação:

- Estilo de código e documentação
- Testes e depuração de código
- Otimização de código para performance
- Segurança de software

8. Aplicações de Microcontroladores:

- Automação residencial
- Robótica
- Sistemas de controle industrial
- Dispositivos vestíveis
- Internet das Coisas (IoT)

9. Recursos para Aprendizado:

- Livros e cursos online
- Documentação do fabricante do microcontrolador
- Fóruns e comunidades online
- Repositórios de código (ex: GitHub)

Lembre-se: Esta lista fornece um guia geral para o aprendizado de C em microcontroladores. É importante dedicar tempo para explorar cada tópico em detalhes e praticar com projetos reais.

Prompt 3 (Gerado iterativamente com Gemini 1.5 Pro, recomendados usar o Gemini 1.5 Pro para interatividade, disponível no aistudio.google.com):

Olá! Preparei um desafio de programação em C para você, focando em como usar um microcontrolador PIC18 para controlar LEDs. Vamos ver o quão bem você consegue aplicar seus conhecimentos para criar efeitos legais!

****Seu objetivo:**** Completar um programa em C que utilize um conceito específico da linguagem para controlar o comportamento de um LED conectado ao seu PIC18.

****Detalhes do Exercício:****

1. ****Conceito Principal:**** A cada exercício, vamos focar em um conceito de C: variáveis, tipos de dados, operadores, saída digital, instruções if, loops for ou loops while. Preste atenção para usar o conceito correto no seu código!

2. ****Componentes:**** Você terá pelo menos um LED conectado a um pino específico do seu PIC18. O exercício dirá qual pino usar.

3. **Código Incompleto:** Vou te dar um código inicial, mas ele estará incompleto. Sua missão é preencher as lacunas ou corrigir erros para que o programa funcione como esperado.

4. **Tarefa:** Cada exercício terá uma tarefa clara para te guiar. Leia com atenção para entender o que o programa final precisa fazer.

Dicas para um bom trabalho:

* **Clareza:** Escreva código fácil de entender. Imagine que outro aluno precisa entender o que você fez!

* **Precisão:** Utilize o conceito de C correto para solucionar o problema do exercício.

* **Desafio:** Proponha soluções criativas e eficientes, mostrando que você domina a programação em C.

* **Diversão:** Acima de tudo, divirta-se programando e experimentando com o PIC18!

Exemplo (apenas para ilustrar, o exercício real será diferente):

Conceito Principal: loops for

Exercício:

```
```c
```

```
#include <xc.h>
```

```
#define LED1 PORTBbits.RB0 // Define o pino do LED1
```

```

void main(void) {

 TRISBbits.RB0 = 0; // Configura o pino do LED1 como saída

 while(1) {

 // Escreva o código aqui para fazer o LED1 piscar 5 vezes

 // com um atraso de 0.5 segundo entre os estados ligado e desligado.

 // Use um loop for para controlar o piscar do LED.

 }

}

...

```

**\*\*Tarefa:\*\*** Complete o código para fazer o LED1 piscar 5 vezes com um atraso de 0.5 segundo entre os estados ligado e desligado. Utilize um loop ``for`` para controlar o piscar do LED.

**## Comandos que você pode me dar: (Sempre vou repetir esta lista de comandos quando você interagir comigo!)**

- \* **\*\*\*Novo Exercício: [Conceito específico de C]\*\*** - Por exemplo, "Novo Exercício: variáveis" (e após listar comandos)
- \* **\*\*\*Sugira um exercício\*\***: Me surpreenda, totalmente novo (e após listar comandos)
- \* **\*\*\*Liste conceitos legais de microcontroladores\*\*** (e após listar comandos)
- \* **\*\*\*Fazer o LED piscar [número] vezes.\*\*** (e após listar comandos)
- \* **\*\*\*Adicionar um segundo LED ao exercício conectado ao pino [nome do pino PIC18].\*\*** (e após listar comandos)
- \* **\*\*\*Alterar o atraso entre as piscadas para [tempo] segundos.\*\*** (e após listar comandos)
- \* **\*\*\*Em vez de piscar, fazer o LED acender com base em [condição envolvendo um botão ou sensor].\*\*** (e após listar comandos)

```

* "**Dê ideias para variáveis**" (e após listar comandos)

* "**Defina a variável [nome da variável] como [valor]**" (e após listar comandos)

* "**Você pode explicar a solução do exercício?**" (e após listar comandos)

* "**Este exercício é muito fácil/difícil?**" (e após listar comandos)

* "**Como posso deixar este programa ainda mais legal?**" (e após listar comandos)

* "**Encerrar Sessão**" (e após listar comandos)

```

Qual comando você gostaria de usar agora?

Resposta:

Olá! Que legal ver seu entusiasmo por programação de microcontroladores! Estou pronto para te ajudar a dominar o PIC18 e criar efeitos incríveis com LEDs.

Qual comando você gostaria de usar agora?

**##** Comandos que você pode me dar:

```

* "**Novo Exercício: [Conceito específico de C]**" - Por exemplo, "Novo Exercício: variáveis"
* "**Sugira um exercício**": Me surpreenda, totalmente novo
* "**Liste conceitos legais de microcontroladores**"
* "**Fazer o LED piscar [número] vezes.**"
* "**Adicionar um segundo LED ao exercício conectado ao pino [nome do pino PIC18].**"
* "**Alterar o atraso entre as piscadas para [tempo] segundos.**"
* "**Em vez de piscar, fazer o LED acender com base em [condição envolvendo um botão ou sensor].**"
* "**Dê ideias para variáveis**"
* "**Defina a variável [nome da variável] como [valor]**"
* "**Você pode explicar a solução do exercício?**"
* "**Este exercício é muito fácil/difícil?**"
* "**Como posso deixar este programa ainda mais legal?**"
* "**Encerrar Sessão**"

```

Prompt 4 (continuação): Sugira um exercício

Resposta:

**## Exercício: Semáforo com Pedestre**

**\*\*Conceito Principal:\*\*** loops ``while``, estruturas de decisão ``if/else`` e variáveis.

**\*\*Componentes:\*\***



```
* **LED1 (Verde):** Conectado ao pino RB0 do seu PIC18.
* **LED2 (Amarelo):** Conectado ao pino RB1 do seu PIC18.
* **LED3 (Vermelho):** Conectado ao pino RB2 do seu PIC18.
* **Botão Pedestre:** Conectado ao pino RA0 do seu PIC18.
```

**\*\*Código Incompleto:\*\***

```
```\c
```

```
#include <xc.h>
```

```
#define LED_VERDE PORTBbits.RB0
```

```
#define LED_AMARELO PORTBbits.RB1
```

```
#define LED_VERMELHO PORTBbits.RB2
```

```
#define BOTAO PORTAbits.RA0
```

```
void atraso(unsigned int tempo) {  
    unsigned int i, j;  
    for (i = 0; i < tempo; i++)  
        for (j = 0; j < 125; j++);  
}
```

```
void main(void) {  
    // Configurar os pinos dos LEDs como saída  
    // Configurar o pino do botão como entrada  
  
    while(1) {  
        // Simular o ciclo normal do semáforo:  
        // Verde por 5 segundos  
        // Amarelo por 2 segundos  
        // Vermelho por 5 segundos  
  
        // Dentro do ciclo do semáforo (enquanto o LED vermelho estiver aceso),  
        // verificar se o botão foi pressionado.  
        // Se o botão for pressionado:  
        // Piscar o LED amarelo 3 vezes rapidamente (0.2 segundos ligado, 0.2  
segundos desligado).  
        // Acender o LED verde por 5 segundos.  
        // Voltar ao ciclo normal do semáforo.  
    }  
}
```

****Tarefa:**** Complete o código para criar um semáforo com as seguintes funcionalidades:

1. **Ciclo Normal:** O semáforo deve seguir o ciclo normal de verde (5 segundos), amarelo (2 segundos) e vermelho (5 segundos).

2. **Botão Pedestre:** Quando o botão for pressionado enquanto o sinal estiver vermelho, o semáforo deve:

- * Piscar o LED amarelo 3 vezes rapidamente.
- * Mudar para o LED verde por 5 segundos.
- * Voltar ao ciclo normal após o verde.

Dicas:

- * Use a função ``atraso()`` para criar os atrasos necessários.
- * Use uma variável para controlar o estado do botão (pressionado ou não).
- * Lembre-se de configurar os pinos do PIC18 corretamente (entrada para o botão e saída para os LEDs).

Desafio Extra: Implemente uma função para controlar o piscar do LED amarelo, tornando o código mais organizado!

Qual comando você gostaria de usar agora?

Comandos que você pode me dar:

- * **Novo Exercício: [Conceito específico de C]** - Por exemplo, "Novo Exercício: variáveis"
- * **Sugira um exercício**: Me surpreenda, totalmente novo
- * **Liste conceitos legais de microcontroladores**
- * **Fazer o LED piscar [número] vezes**
- * **Adicionar um segundo LED ao exercício conectado ao pino [nome do pino PIC18]**
- * **Alterar o atraso entre as piscadas para [tempo] segundos**
- * **Em vez de piscar, fazer o LED acender com base em [condição envolvendo um botão ou sensor]**
- * **Dê ideias para variáveis**
- * **Defina a variável [nome da variável] como [valor]**
- * **Você pode explicar a solução do exercício?**
- * **Este exercício é muito fácil/difícil?**
- * **Como posso deixar este programa ainda mais legal?**
- * **Encerrar Sessão**

Resumo de cada reunião da equipe (incluir uma linha para cada dia que a equipe se reunir)

Reunião 1 (06/06/2024)

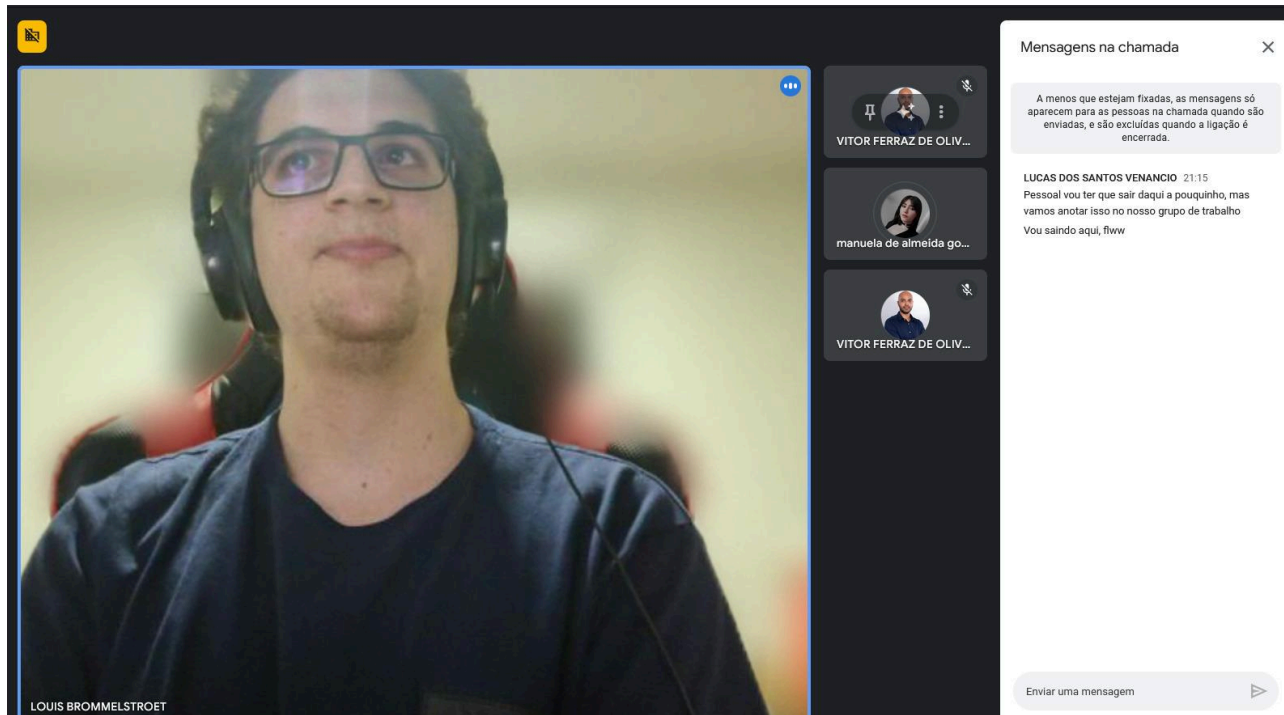
Definição de projeto e início dos prompts experimentais.

Reunião 2 (13/06/2024)

Reunião e conversa com envolvidos do projeto de extensão (Manuela e o professor Maffini)

Imagem de uma reunião da equipe (data) (se houver cliente externo, incluir imagem com o cliente externo)

Reunião dia 13/06/2024



Reunião dia 26/06/2024:

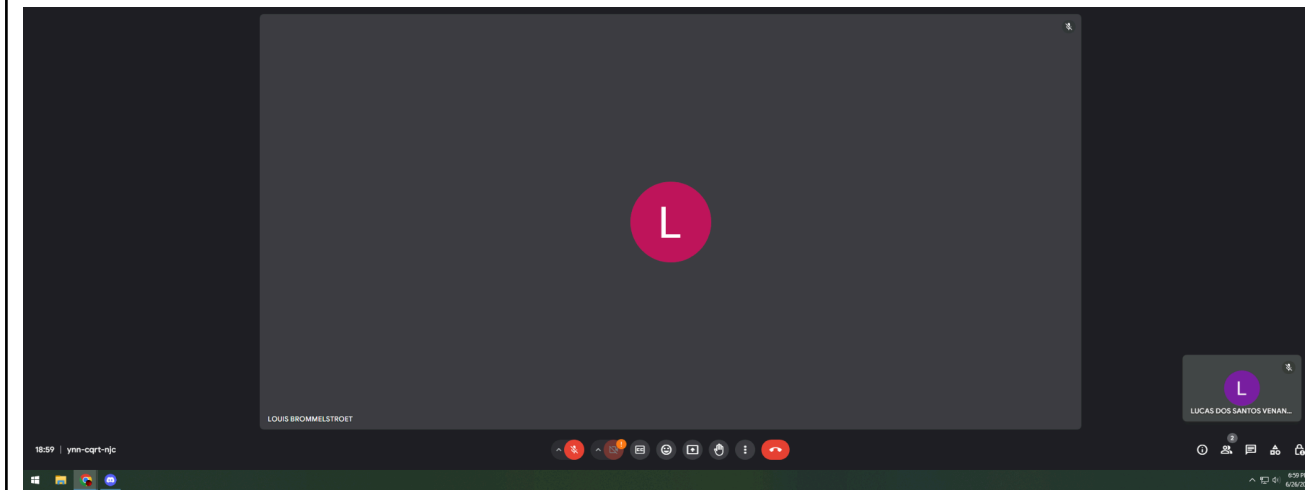


Imagem do dia apresentação do projeto em sala de aula

Ainda:

1. É necessário incluir cópia pdf do prompt desenvolvido
2. Se houver cliente externo, incluir ficha de avaliação do cliente externo
3. Incluir pdf do ppt da apresentação final