Robótica Educacional

Módulo 3

Imagem gerada com I.A

Aula 11

Moteto - 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

Darice Alessandra Deckmann Zanardini

Validação de Conteúdo

Cleiton Rosa

Revisão Textual

Kellen Pricila dos Santos Cochinski

Projeto Gráfico e Diagramação

Edna do Rocio Becker

2024

Sumário

Moteto - II

Introdução

Conhecemos, na aula anterior, o gênero polifônico **moteto** e, inspirados pela instalação sonora "**Forty Part Motet**", de Janet Cardiff, iniciamos o projeto de recriar visualmente a obra, inicialmente em uma escala menor com 8 LEDs. A instalação é um exemplo de como a tecnologia pode ser usada para recriar e reinterpretar obras históricas e seculares, proporcionando uma nova maneira de integrar elementos com arte. "Forty Part Motet" é também um exemplo de como a arte transcende o tempo e as barreiras culturais, promovendo novas experiências e nos inspirando a projetos como o de controles diferenciados dos LEDs presentes no protótipo.

Aula 11 Moteto - II

Objetivos desta aula

- Programar protótipo inspirado na instalação "Forty Part Motet";
- Experimentar função millis() para controle simultâneo de LEDs, representando visualmente a proposta da obra.

Lista de materiais

- 8 LEDs (para ampliação do projeto, 40);
- 8 resistores 220 Ohms (ou outros valores, conforme ampliação do protótipo);
- 9 jumpers macho-macho (para ampliação do projeto, 25);
- protoboard;
- Arduino Uno;
- computador ou notebook.

Roteiro da aula

1. Contextualização

Desenvolver um projeto de Robótica para simular, visualmente, um moteto é a nossa proposta para combinar arte e tecnologia.

Dando sequência a projeto iniciado na **Aula 10 - Moteto [Parte I]**, pensaremos na Robótica como um recurso para o campo criativo, no qual se recria e reinterpreta, agora sob um viés visual, uma obra artística.

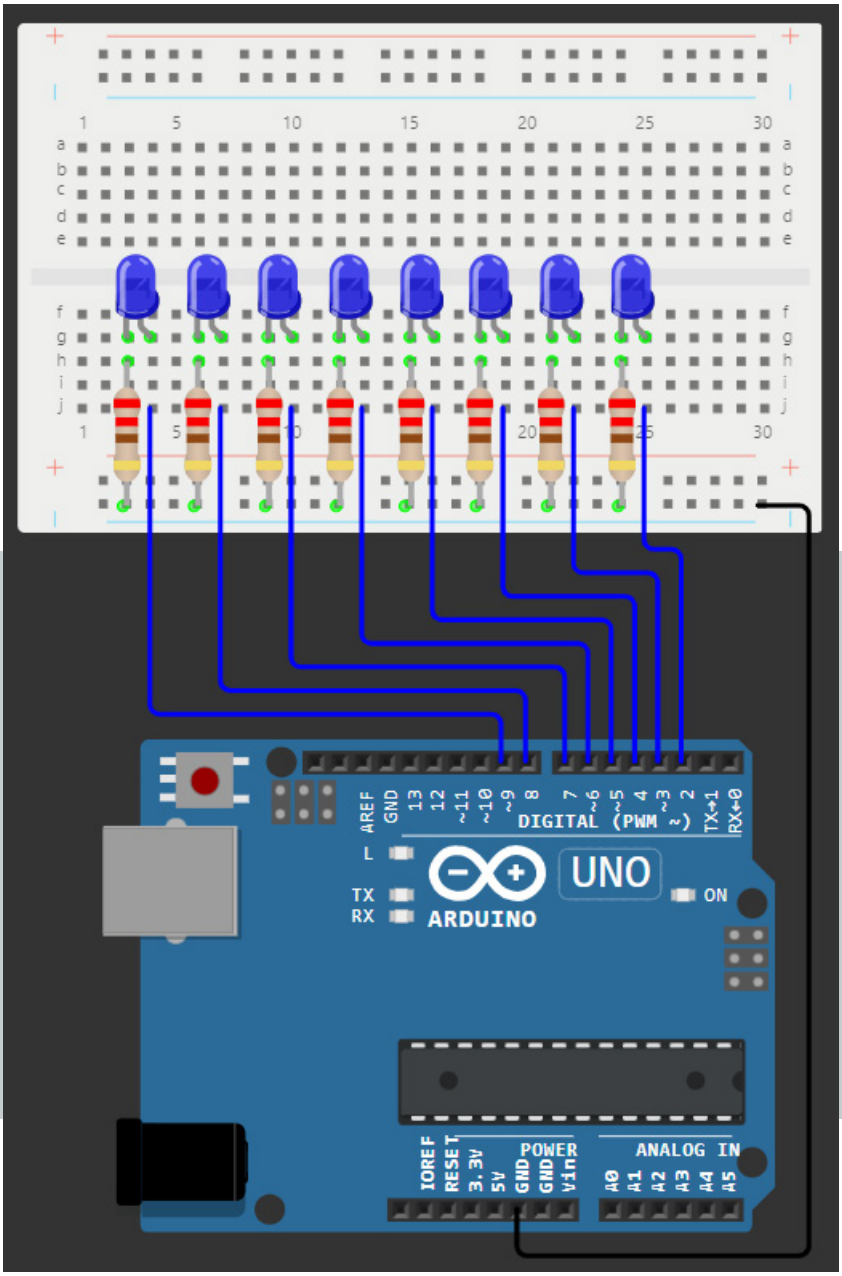
Neste projeto, o acionamento diferenciado de LEDs, controlados pela função **millis()**, representará de modo visual vozes distintas, tal como na instalação sonora "Forty Part Motet", nos dando uma dimensão acerca de, em uma mesma obra, se ter execuções variadas a cada movimento.

Imagem gerada com I.A

Moteto - II

Como vimos no vídeo Janet Cardiff and the Forty Part Motet | TateShots (02min56), a artista fala sobre o movimento das 40 harmonias presentes em sua instalação. Aqui, nós teremos o movimento promovido pelo tempo de acionamento de cada LED no nosso projeto inicial.

Figura 1 – Montagem do protótipo moteto com 8 LEDs no simulador Wokwi



Fonte: Wokwi

Moteto - II

2. Montagem e programação

Aprendemos, na **Aula 09 - Função millis()**, que o controle independente de componentes conectados ao Arduino pode ser feito de duas formas: trabalhando diretamente com a função **millis()** e aplicando os cálculos de controle temporal ou com o auxílio da biblioteca **Neotimer**, a qual facilita os cálculos e exige a definição de objetos de controle para cada temporizador. Na programação desta aula, utilizaremos diretamente os cálculos para a função **millis()** e, com o recurso **for** para loop e incremento, definiremos os pinos e controles do nosso protótipo de modo otimizado.

Para representar os conjuntos de caixas de som da obra "Forty Part Motet" com LEDs diferentes e em ritmos de acionamento específicos, a função **millis()** é uma escolha mais adequada para nossa programação do que o **delay()**. Isso porque **millis()** nos permite gerenciar o tempo sem bloquear o restante do código durante execução da programação no **void loop()**, o que é essencial para manter os LEDs piscando de forma independente, promovendo movimentos, e em sincronia, caso você se sinta desafiado a ampliar o projeto, a uma música para, como desafio, ser adicionada ao projeto.

Para cada controle dos LEDs que adicionaremos à programação, utilizaremos outro recurso. Neste código, cada voz representada tem seu próprio intervalo de tempo e sua verificação para determinar quando o LED será acionado. A função **digitalRead()** aplicada a cada pino lê o estado atual do LED e o operador **!** inverte esse estado lógico, fazendo com que o LED pisque conforme o tempo que definirmos. Com a função **millis()**, criamos um sistema de iluminação que simula as diferentes vozes de um moteto de forma independente e sincronizada, sem atrasar a execução do programa.

Vamos às etapas da nossa programação para você conferir também as utilizações do **for**? Como é uma estrutura de repetição na programação que permite executar um bloco de código várias vezes, na nossa programação ele vai iterar os **arrays** de pinos e tempos, ou seja, percorrer cada conjunto de elementos do mesmo tipo e em sequência, o que otimiza as linhas de programação. Logo compreenderemos melhor isso na análise das linhas do nosso código.

Neste código, cada LED é associado a um intervalo de piscar diferente, permitindo o acionamento em "frequências" distintas e um movimento de luzes! A função **millis()** é usada para verificar continuamente se chegou a hora de mudar o estado de cada LED, sem bloquear o restante do código, comparando o tempo atual com o tempo registrado da última vez que o LED piscou, mais o intervalo definido para cada LED.

Vamos à nossa programação?

Moteto - II

```
/*
Aulas 10 e 11 - Projeto Moteto com LEDs.

Blinks diferentes para controle independente de
cada LED representado visualmente os diferentes
grupos da obra “Forty Part Motet”, de Janete Cardiff.
*/

// Define os pinos dos LEDs.
const int pinoLEDs[] = { 2, 3, 4, 5, 6, 7, 8, 9 };
/* Define os intervalos de piscar para cada LED (em milisse-
gundos).*/
const long intervaloBlink[] = { 900, 800, 700, 600, 500,
400, 300, 200 };
// Armazena a última vez que cada LED piscou.
unsigned long tempoAnterior[] = { 0, 0, 0, 0, 0, 0, 0, 0 };

void setup() {
  // Configura todos os pinos como saída.
  for (int i = 0; i < 8; i++) {
    pinMode(pinoLEDs[i], OUTPUT);
  }
}
```

Moteto - II

```
void loop() {
  // Obtém o tempo atual.
  unsigned long tempoAtual = millis();

  // Verifica cada LED.
  for (int i = 0; i < 8; i++) {
    /* Se o tempo atual for maior que o último piscar mais o
intervalo.*/
    if (tempoAtual - tempoAnterior[i] >= intervaloBlink[i])
{
      // Atualiza o último tempo de piscar.
      tempoAnterior[i] = tempoAtual;
      // Muda o estado do LED.
      digitalWrite(pinoLEDs[i], !digitalRead(pinoLEDs[i]));
    }
  }
}
```

Como você pode perceber, temos três etapas em nossa programação – **preâmbulo**, **void setup()** e **void loop()** – e utilizamos diferentes tipos de variáveis para armazenar e gerenciar as informações do nosso projeto:

- **const int**, um valor constante inteiro atribuído à variável para os pinos dos LEDs.
- **const long**, também um valor constante, porém armazenando números maiores relacionados ao intervalo para cada "piscada" dos LEDs.
- **unsigned long**, variável que armazena apenas números inteiros positivos, ou seja, sem sinal negativo, para o registro de tempo.

Esses tipos são escolhidos com base no tamanho e na natureza dos dados que precisam ser armazenados. A função que utilizamos no nosso projeto, **millis()**, retorna um **unsigned long** porque o número de milissegundos que o Arduino chega, depois de iniciado, pode ser muito grande e não será negativo. Portanto, conforme os objetivos do projeto e programação escolhida, é importante escolher o tipo de variável adequada, garantindo assim que o programa funcione corretamente e de forma eficiente.

Outro detalhe que você pode ter percebido é que nossa programação está utilizando **arrays** para as definições das variáveis **pinoLEDs[]**, **intervaloBlink[]** e tempo **Anterior[]**. Nas **Aulas 06** e **07** – **Instax OLED [Partes I e II]**, utilizamos **arrays** para as matrizes de bytes das imagens a serem impressas. Aqui, utilizamos essas estruturas de dados para manter o código organizado, agrupando os elementos relacionados em uma única variável e aplicando o **for** para realizar as operações repetitivas nos elementos do **array**.

- **pinoLEDs[]**: Armazena os números dos pinos aos quais os LEDs estão conectados e o **for**, presente no **void setup()** com sua variável local de controle **i**, itera sobre o **array**, percorrendo cada pino do LED para configurá-los como saída. No caso da iteração no **void loop()**, o **for** presente neste trecho aciona os LEDs e atualiza o tempo.
- **intervaloBlink[]**: Armazena os intervalos de tempo para o piscar de cada LED.

Como o foco do nosso projeto é o controle dos LEDs, a array **tempoAnterior[]** tem a função de armazenar o momento exato em que cada LED foi acionado pela última vez e, quando o programa inicia, os valores indicados em **{0, 0, 0, 0, 0, 0, 0, 0}** são atualizados para refletir o tempo do sistema que obtemos pela função **millis()**. Isso é essencial para controlar o piscar de cada LED em intervalos diferentes com o Arduino "sabendo" quanto tempo passou desde que o estado lógico do LED foi alterado pelo recurso ! aplicado à leitura das portas digitais conforme o intervalo (em milisegundos) definido para cada LED pela array **intervaloBlink[] = { 900, 800, 700, 600, 500, 400, 300, 200 }**.

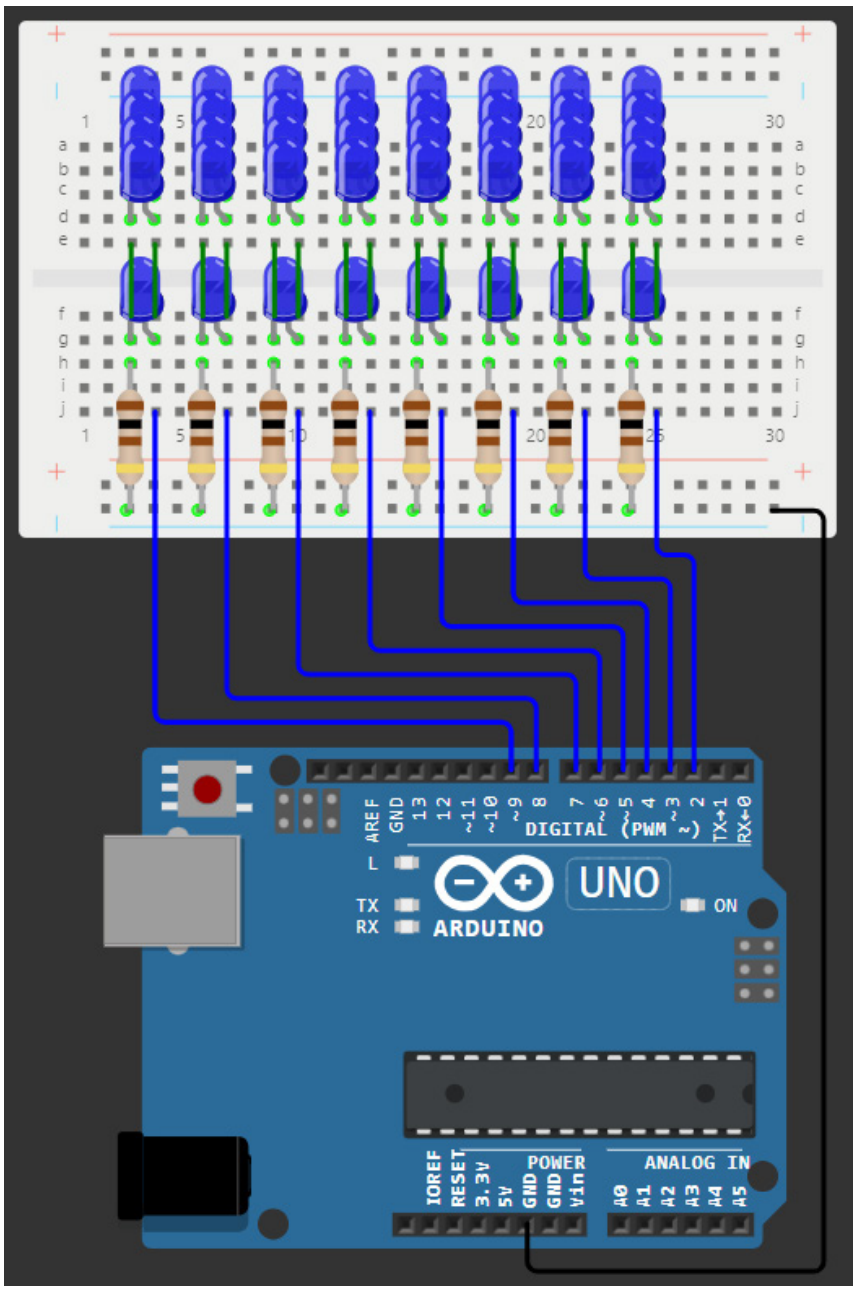
Como podemos analisar pela execução do nosso projeto no simulador, essa técnica permite, de modo otimizado, que o programa controle múltiplos LEDs de forma independente e assíncrona, criando um efeito visual dinâmico a cada LED sem interrupção às demais execuções.

Moteto - II

Modifique o código e a configuração dos componentes para verificar como essas alterações podem afetar a simulação. Lembre-se de que, no Wokwi, você pode simular esse código e ver o resultado, entendendo como a função **millis()** pode ser usada para criar efeitos de iluminação mais complexos e sincronizados e provocar reações como uma pequena instalação artística.

Para transformar seu projeto em uma instalação, como você pode montar fisicamente o projeto que simulamos? Com seus colegas, pense nas possibilidades e na ampliação do projeto, como no modelo em que associamos em série mais LEDs a cada LED já programado, agora com resistores de 100 Ohms nas colunas de LEDs azuis para preservar o circuito e não ultrapassar os 50mA suportado pelo Arduino, criando assim uma coluna de luzes como ampliação do movimento.

Figura 2 – Modelo do protótipo moteto com 40 LEDs



Fonte: Wokwi

Confira, no modelo Moteto com 40 LEDs, a simulação de funcionamento da proposta e inspire-se às suas criações!

Moteto - II

Desafios:

Para ampliar o projeto proposto – e até desenvolver outros, você e seus colegas podem traçar um esboço, de modo semelhante ao que propomos na **Aula 02 - Projetando ideias**:

- **Pesquisa e planejamento:** Estude mais sobre as características dos motetos, ou outros gêneros que você queira representar, e defina os parâmetros a serem simulados.
- **Design do projeto:** Que tal ampliar o projeto e criar pequenos robôs ou outros elementos para representarem as vozes do moteto?
- **Construção do protótipo:** Quais novos materiais poderão ser incorporados?
- **Programação:** Quais ajustes são necessários? Será possível, no nosso projeto de representação visual do moteto, associar um software ou arquivo de áudio sincronizado ao acionamento dos LEDs que corresponde aos coros?
- **Testes e ajustes:** Chegou o momento de analisar o desempenho do protótipo e os sincronismos propostos e ajustar o sistema para melhorias.
- **Demonstração:** Que tal apresentar seu projeto em uma feira ou evento? Aproveite o momento para explicar sobre o gênero musical e como a Robótica pode ser utilizada para proporcionar momentos artísticos e culturais.

E se...

O projeto não funcionar?

- Verifique as variáveis criadas para armazenamento do tempo do LED e do tempo decorrido do Arduino, considerando sua sintaxe e atribuições pelo recurso **for**.
- Confira a criação dos objetos de controle destinados a cada componente que terá um controle temporal próprio.
- Revise as conexões do protótipo e todas as portas declaradas no sketch da programação para corresponderem à montagem que você realizou.

Moteto - II

3. Feedback e finalização

Como é a experiência de se inspirar em projetos variados para desenvolver seus próprios, atribuindo novos significados? E nesse processo, como é também perceber que a própria programação de projetos pode ser desenvolvida de formas diferenciadas? Neste Módulo 3, nossa proposta é perceber isso, outras possibilidades!

Continue sua jornada para se aprimorar cada vez mais! Bons estudos!

REFERÊNCIAS

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

BACILDO, Walter Costa; CIRILLO, Aparecido José. O ambiente expositivo no processo criativo de Janet Cardiff em Forty Part Motet. **Revista Farol**, ***[S. l.],*** v. 14, n. 19A, p. 151–159, 2018. DOI: 10.47456/rf.v1i19A.20484. Disponível em: https://periodicos.ufes.br/farol/article/view/20484. Acesso em: 15 abr. 2024.

CARDIFF & MILLER. **The Forty Part Motet**. Disponível em: https://cardiffmiller.com/installations/the-forty-part-motet/. Acesso em: 12 abr. 2024.

KQED ARTS. **One Collective Breath: Janet Cardiff's 'The Forty Part Motet' | KQED Arts**. YouTube, 4 de dez. de 2015. Disponível em: https://youtu.be/rZXBia5kuqY. Acesso em: 29 abr. 2024.

NATALIA AUREA. **Aula 20 História da Música - Philippe de Vitry e Moteto**. YouTube, 23 de set. de 2021. Disponível em: https://youtu.be/YbWbDzYmNCQ. Acesso em: 29 abr. 2024.

TATE. **Janet Cardiff and the Forty Part Motet | TateShots**. YouTube, 7 de jul. de 2017. Disponível em: https://youtu.be/38ORiaia9r8. Acesso em: 29 abr. 2024.

DIRETORIA DE TECNOLOGIAS E INOVAÇÃO (DTI)

COORDENAÇÃO DE TECNOLOGIAS EDUCACIONAIS (CTE)

EQUIPE ROBÓTICA PARANÁ

- Ailton Lopes
- Andrea da Silva Castagini Padilha
- Cleiton Rosa
- Darice Alessandra Deckmann Zanardini
- Edna do Rocio Becker
- Kellen Pricila dos Santos Cochinski
- Marcelo Gasparin
- Michele Serpe Fernandes
- Michelle dos Santos
- Roberto Carlos Rodrigues
- Sandra Aguera Alcova Silva

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.

Este trabalho está licenciado com uma Licença
Creative Commons – CC BY-NC-SA
Atribuição - NãoComercial - CompartilhaIgual 4.0