UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CURSO DE ENGENHARIA ELÉTRICA/CONTROLE E AUTOMAÇÃO

Yuri Ramos Borges

Projeto Genius

RELATÓRIO DO PROJETO FINAL DA DISCIPLINA DE ELETRÔNICA DIGITAL

CURITIBA

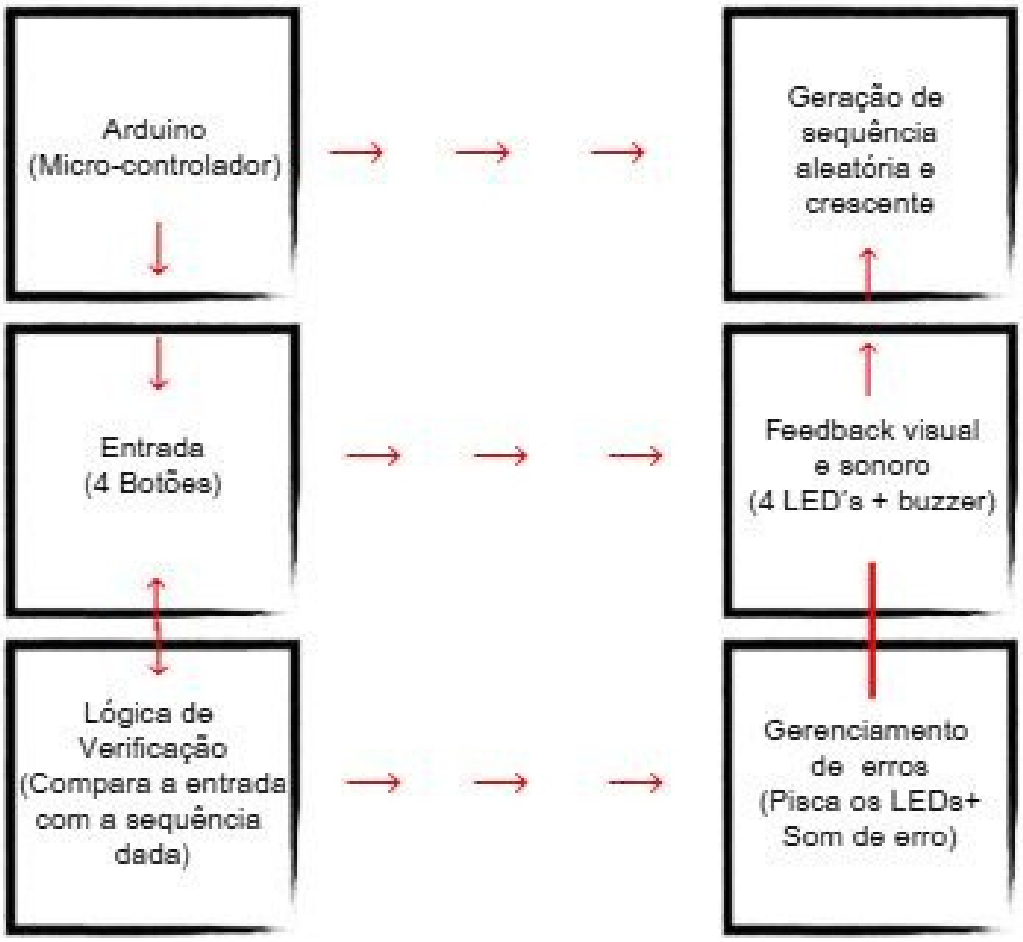
2025

INTRODUÇÃO

No cenário atual, a **interação com a eletrônica e a programação** pode parecer complexa e distante para iniciantes. Muitas ferramentas e conceitos são abstratos, dificultando a compreensão prática e o engajamento. Há uma lacuna entre a teoria e a aplicação real, que desmotiva o aprendizado e a exploração de projetos "maker". O problema a ser resolvido, portanto, é a **falta de uma abordagem prática e intuitiva para o aprendizado de eletrônica e programação básica**, que seja ao mesmo tempo divertida e desafiadora. A motivação para resolver esse problema reside na necessidade de **democratizar o acesso ao conhecimento tecnológico**, tornando-o mais acessível e estimulante para curiosos e futuros desenvolvedores, além de demonstrar a aplicabilidade de conceitos fundamentais de forma concreta.

O projeto é a criação de uma versão interativa do clássico **jogo Genius (também conhecido como Simon)**, utilizando a plataforma Arduino. Este jogo é ideal para demonstrar princípios básicos de eletrônica e programação de forma engajadora. O **objetivo principal** é desenvolver um protótipo funcional que recrie a experiência do Genius, onde o jogador precisa memorizar e reproduzir sequências crescentes de luzes e sons. Através da montagem do circuito com LEDs, botões e um buzzer, e da programação do Arduino para gerenciar a lógica do jogo (geração de sequências aleatórias, feedback visual/sonoro, detecção de acertos e erros, e reinício), o projeto visa proporcionar uma **experiência prática e divertida** de aprendizado. Isso inclui entender o papel de cada componente eletrônico, a interatividade entre hardware e software, e a lógica por trás de um sistema responsivo, incentivando o raciocínio lógico e a solução de problemas.

DIAGRAMA EM BLOCOS DO PROJETO



DESCRIÇÃO DA SOLUÇÃO

O sistema a ser implementado é um **jogo interativo** baseado no conceito do popular "Genius" (ou "Simon"), construído sobre a plataforma Arduino. Ele foi projetado para ser um protótipo didático que demonstra a interação entre componentes eletrônicos básicos e a lógica de programação. O objetivo do jogo é testar a memória do usuário através de sequências crescentes de estímulos visuais (luzes LED) e auditivos (sons do buzzer), que o jogador deve replicar usando botões.

Para entender o funcionamento do sistema, podemos dividi-lo em blocos com funções específicas:

Arduino (Micro-controlador Central)

- **Função:** Atua como o **cérebro** de todo o sistema. Ele armazena e executa o código que contém a lógica do jogo. Suas principais responsabilidades incluem gerar as sequências aleatórias de cores, controlar o acendimento dos LEDs e a emissão dos sons, ler o estado dos botões pressionados pelo jogador, verificar a correção das entradas e gerenciar os diferentes estados do jogo (início, mostrando sequência, esperando entrada, erro).

Módulo de Entrada (Botões)

- **Função:** É o meio pelo qual o **jogador interage** com o jogo. Os quatro botões (um para cada cor) servem como chaves de entrada. Quando um botão é pressionado, ele altera o estado elétrico de um pino digital

Bloco de Gerenciamento de Erros e Reinício

- **Função:** Lida com a situação em que o jogador comete um erro na sequência. Este bloco ativa um feedback visual específico (todos os LEDs piscam) e um feedback sonoro de erro. Após uma breve pausa, ele redefine o estado do jogo para o início, permitindo que uma nova partida seja iniciada ao pressionar qualquer botão.

Interface entre os Blocos do Sistema

A interface entre os blocos do sistema é primordialmente **elétrica e digital**, gerenciada pelo Arduino:

Arduino e Módulo de Entrada (Botões):
Tipo de Interface: Digital.
Funcionamento: Os botões estão conectados a pinos digitais de entrada do Arduino. O Arduino "lê" o estado desses pinos. Quando um botão é pressionado, ele fecha o circuito para o GND, e o pino digital muda seu estado de HIGH (pela configuração INPUT_PULLUP interna) para LOW. Essa mudança de estado é interpretada pelo Arduino como um toque no botão correspondente.

Arduino e Módulo de Saída (LEDs e Buzzer):
Tipo de Interface: Digital (para LEDs), Digital/PWM (para Buzzer, embora tone() abstraia isso).
Funcionamento:
LEDs: Os LEDs estão conectados a pinos digitais de saída do Arduino. O Arduino envia um sinal de ALTO (5V) para acendê-los e um sinal de BAIXO (0V) para apagá-los. Resistores são

inseridos em série com os LEDs para controlar a corrente e protegê-los.
Buzzer: O buzzer está conectado a um pino digital de saída do Arduino. O Arduino gera ondas quadradas de diferentes frequências (usando a função tone()) através deste pino para produzir os diversos tons musicais e o som de erro.

Bloco de Lógica do Jogo e Demais Blocos:

- **Tipo de Interface:** Software/Controle de Fluxo.
- **Funcionamento:** A lógica implementada no código do Arduino atua como o maestro, coordenando as interações. Ela recebe os dados do Módulo de Entrada (botões), processa-os, e envia comandos para o Módulo de Saída (LEDs e Buzzer). Além disso, ela comunica o estado atual do jogo e as transições entre os blocos (por exemplo, passar de "mostrando sequência" para "esperando entrada" ou para "erro").

Em essência, o Arduino é o mediador central, traduzindo as ações físicas do jogador em comandos digitais e convertendo a lógica do jogo em respostas visuais e sonoras no mundo físico.

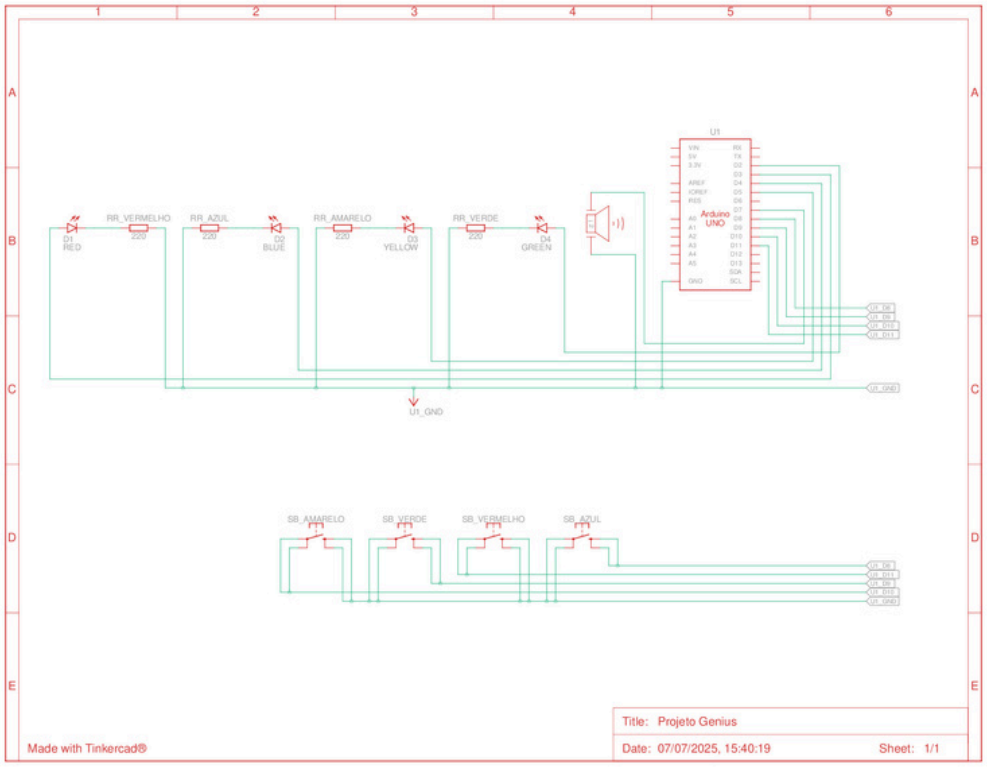
LISTA DE MATERIAIS

Tabela 1 – Lista de materiais.

Item	Quantidade	Fonte
Arduino UNO	1	24 de maio
LED’s de 5mm (cores distintas)	4	24 de maio
Botão Tátil	4	24 de maio
Buzzer Ativo	1	24 de maio
Resistor de 220 Ohms	4	24 de maio
Placa Protoboard	1	24 de maio
Jumpers Macho-Macho	30	24 de maio

Fonte: Autoria própria.

CIRCUITO ESQUEMÁTICO DO PROJETO



MANUAL DE OPERAÇÃO

Este mini manual descreve como operar o seu jogo Genius interativo com Arduino.

Ligando o Jogo

Conecte o seu Arduino ao computador usando o cabo USB.
Os **LEDs piscarão** algumas vezes e o buzzer emitirá um som inicial, indicando que o jogo está pronto para começar.

Início de uma Nova Rodada

O jogo mostrará automaticamente a primeira sequência. Um dos **LEDs acenderá e o buzzer tocará uma nota**.
Aguarde a sequência ser concluída.

Repetindo a Sequência

Após a sequência ser mostrada, será a sua vez de jogar.
Pressione e solte o botão correspondente à cor do LED que acendeu na sequência.
Para cada botão pressionado corretamente, o **LED referente acenderá e um som será emitido**.

Avançando no Jogo

Se você **acertar toda a sequência** da rodada, o jogo adicionará uma nova cor (e som), aumentando a sequência e o desafio.
Aguarde a nova sequência ser mostrada e tente repeti-la.

Errando a Sequência (Game Over)

Se você **pressionar o botão errado** ou não seguir a ordem correta da sequência:
Todos os **LEDs piscarão três vezes** juntos.
Um **som de erro** será emitido pelo buzzer.
O jogo fará uma **pausa de 5 segundos**.
Após a pausa, o jogo aguardará. Para **recomeçar**, basta **pressionar qualquer um dos botões**.

Dicas para Jogar:

Aperte e solte: Não segure os botões. Um toque rápido é suficiente.
Observe e escute: Preste muita atenção tanto nas luzes quanto nos sons para memorizar a sequência.
Divirta-se jogando o seu próprio Genius!

LINK PARA O VÍDEO DEMONSTRATIVO NO YOUTUBE

https://youtube.com/shorts/f1R4iqAajYU?si=yrzc6AZ9ckMW_j0d

PRINCIPAIS DIFICULDADES ENCONTRADAS E SOLUÇÕES UTILIZADAS

O desenvolvimento do projeto Genius, embora recompensador, apresentou desafios comuns à prototipagem eletrônica. Identificar e solucionar esses problemas foi uma parte crucial do processo de aprendizado.

1. LEDs Queimando e Brilho Inadequado

- **Problema:** Inicialmente, os **LEDs queimavam** rapidamente ao serem conectados, ou acendiam com um brilho excessivo/insuficiente. Isso era um indicativo de corrente inadequada.
- **Causa:** A fiação inicial utilizava **resistores de 100 Ohms**, valor que se mostrou muito baixo para proteger os LEDs adequadamente quando alimentados pelos 5V do Arduino. Além disso, houve um problema de **polaridade invertida** em alguns LEDs, o que os impedia de acender ou contribuía para sua queima.
- **Solução:** Substituímos os resistores por outros de **220 ou 330 Ohms**, que são valores seguros para LEDs comuns de 5mm com alimentação de 5V. A polaridade dos LEDs foi corrigida (ânodo no pino digital, catodo via resistor para o GND), garantindo o fluxo correto da corrente.

2. LEDs Acessos Constantemente

Problema: Mesmo após corrigir a questão da queima, os **LEDs permaneciam acesos o tempo todo**, sem serem controlados pelo código.

Causa: A fiação havia conectado os LEDs diretamente ao pino **5V da protoboard/Arduino**, em vez de conectá-los aos pinos digitais de saída (2, 3, 4, 5). Dessa forma, os LEDs recebiam energia constante e não podiam ser ligados e desligados pelo programa.
Solução: Refizemos a fiação, garantindo que o ânodo de cada LED estivesse conectado ao seu respectivo **pino digital de saída do Arduino**, permitindo que o microcontrolador controlasse seu estado (ligado/desligado) através do código.

3. Botões Não Respondendo

Problema: Os **botões não registravam os pressionamentos**, impossibilitando a interação do jogador com o jogo.
Causa: As causas mais comuns para isso foram **cabos frouxos** na protoboard ou nos pinos do Arduino, ou uma **conexão incorreta** dos botões ao GND. Além disso, a presença de um **resistor na linha do botão** (o que é desnecessário com INPUT_PULLUP) poderia interferir na leitura.
Solução: Realizamos uma **verificação minuciosa de todos os cabos e conexões dos botões**, garantindo que um terminal fosse para o pino digital e o outro diretamente para o GND, sem resistores. Testes individuais com Serial.print foram cruciais para confirmar que o Arduino estava lendo os estados dos botões corretamente. A persistência em refazer e testar essas conexões foi essencial.

4. Diferença de Comportamento entre Simulação (Tinkercad) e Hardware Físico

Problema: O código funcionava perfeitamente na simulação do Tinkercad, mas o comportamento era inconsistente ou incorreto no **Arduino físico**.
Causa: O Tinkercad simula um ambiente ideal, sem as nuances do mundo real como cabos frouxos, componentes defeituosos, ruído elétrico ou pequenas variações de timing. No hardware, esses fatores podem causar falhas inesperadas.
Solução: Adotamos uma abordagem de **depuração sistemática no hardware**. Isso envolveu isolar os componentes (testar LEDs, botões e buzzer separadamente), refazer a fiação do zero com máxima atenção e usar o Monitor Serial do Arduino para verificar o fluxo do programa e as leituras dos sensores em tempo real. A experiência reiterada em montar, testar e corrigir as ligações físicas foi fundamental para alinhar o comportamento do protótipo com o esperado na simulação.

A superação desses obstáculos foi um testemunho da importância da paciência, da atenção aos detalhes na montagem e do uso de técnicas de depuração, transformando cada problema em uma oportunidade de aprendizado prático em eletrônica.

CONCLUSÕES

O desenvolvimento do projeto do jogo Genius com Arduino foi uma jornada de aprendizado extremamente **prática e enriquecedora**. Conseguimos criar um protótipo funcional que replica a experiência básica do jogo original, cumprindo os objetivos iniciais de interação visual e sonora. Um dos pontos mais importantes do projeto foi a **compreensão aprofundada da interação entre hardware e software**. Cada componente, desde os **LEDs** e **resistores** até os **botões** e o **buzzer**, exigiu uma fiação precisa e um entendimento claro de sua função elétrica. As etapas de depuração, especialmente quando o comportamento do circuito físico divergia da simulação no Tinkercad, foram cruciais. Elas reforçaram a importância de **testar cada componente isoladamente** e de realizar uma **verificação meticulosa da fiação**, o que é uma habilidade fundamental em qualquer projeto de eletrônica.
O uso do **Arduino** como micro-controlador central mostrou sua versatilidade e facilidade de uso para prototipagem rápida. A implementação da **lógica de máquina de estados** no código (INICIO, MOSTRANDO_SEQUENCIA, ESPERANDO_ENTRADA, ERRO) foi essencial para organizar o fluxo do jogo e lidar com os diferentes momentos de forma eficiente. Em suma, este projeto não apenas resultou em um jogo interativo divertido, mas também proporcionou um **aprendizado sólido sobre os fundamentos da eletrônica e da programação embarcada**, destacando a importância da paciência, da depuração sistemática e da persistência na resolução de desafios técnicos.

TRABALHOS FUTUROS

Para aprimorar o projeto, futuras implementações poderiam incluir:

- Contador de Pontos/Rodadas: Adicionar um display (LCD, 7 segmentos) para mostrar a pontuação atual (rodada alcançada).
- Níveis de Dificuldade: Implementar diferentes modos de jogo, como variar a velocidade da sequência ou o tempo de resposta.
- Sons Mais Complexos: Utilizar uma biblioteca de som mais avançada ou um módulo de áudio para reproduzir melodias mais elaboradas.
  Design da Interface: Desenvolver uma caixa ou invólucro para o jogo, com botões e LEDs bem integrados, tornando-o mais robusto e esteticamente agradável

CÓDIGO DE PROGRAMAÇÃO DO PROJETO DO ARDUINO

Código de programação desenvolvido em linguagem C neste projeto final.

```
const int LEDs[] = {2, 3, 4, 5};
const int BOTOES[] = {8, 9, 10, 11};
const int BUZZER = 7;
const int NOTAS[] = {349, 330, 294, 262};
const int NOTA_ERRO = 100;
int sequencia[100];
int rodadaAtual = 0;
int entradaUsuario = 0;
enum EstadoJogo {
INICIO,
MOSTRANDO_SEQUENCIA,
ESPERANDO_ENTRADA,
ERRO
};
EstadoJogo estadoAtual = INICIO;

void setup() {
for (int i = 0; i < 4; i++) {
    pinMode(LEDs[i], OUTPUT);
    pinMode(BOTOES[i], INPUT_PULLUP);
  }    pinMode(BUZZER,  OUTPUT);
randomSeed(analogRead(A0));     }
void loop() {  switch (estadoAtual) {
    case INICIO:
      iniciarJogo();
      break;
    case MOSTRANDO_SEQUENCIA:
      mostrarSequencia();
      break;
    case ESPERANDO_ENTRADA:
      coletarEntradaUsuario();
      break;
    case ERRO:
      piscarErro();
      break;
   }
  }
```

```
void iniciarJogo() {
rodadaAtual = 0;
entradaUsuario = 0;
for (int i = 0; i < 100; i++) {
    sequencia[i] = 0;
 }
piscarTodos(100, 2);
delay(500);
proximaRodada();
}

void proximaRodada() {
rodadaAtual++;
entradaUsuario = 0;
sequencia[rodadaAtual - 1] = random(0, 4);
estadoAtual = MOSTRANDO_SEQUENCIA;
}

void mostrarSequencia() {
delay(1000);
for (int i = 0; i < rodadaAtual; i++) {
    acionarCor(sequencia[i], 400);
    delay(200);
 }  estadoAtual = ESPERANDO_ENTRADA; }

void coletarEntradaUsuario() {
int botaoPressionado =
lerBotoes();  if
(botaoPressionado != -1) {

acionarCor(botaoPressionado, 200);
if (botaoPressionado == sequencia[entradaUsuario]) {
       entradaUsuario++;
       if (entradaUsuario == rodadaAtual) {
           delay(500);
           proximaRodada();

      }
} else {
    }
     estadoAtual = ERRO;
    }
```

```
  }

   void piscarErro() {
   tone(BUZZER, NOTA_ERRO, 500);
   delay(500);
   noTone(BUZZER);

  for (int i = 0; i < 3; i++) {
    piscarTodos(300, 1);

    }
     delay(5000);
   while (lerBotoes() == -1) {
  }
  estadoAtual = INICIO;
  }

 void acionarCor(int corIndex, int duracao) {
 digitalWrite(LEDs[corIndex], HIGH);
 tone(BUZZER, NOTAS[corIndex], duracao);
 delay(duracao);
 digitalWrite(LEDs[corIndex], LOW);
 noTone(BUZZER); }

 int lerBotoes() {
 for (int i = 0; i < 4; i++) {
 if (digitalRead(BOTOES[i]) == LOW) {
       delay(50);
       while (digitalRead(BOTOES[i]) == LOW);
       return i;
     }
  }  return -1; }

void piscarTodos(int duracao, int vezes) {
for (int j = 0; j < vezes; j++) {

  for (int i = 0; i < 4; i++) {
    digitalWrite(LEDs[i], HIGH);

} delay(duracao); for (int i = 0; i < 4; i++) {

      digitalWrite(LEDs[i], LOW);
      } delay(duracao);

  }
}
```

REFERÊNCIAS

Documentação oficial d'Arduino: https://www.arduino.cc/reference/en/

TinkerCAD: https://www.tinkercad.com/

Conhecimentos de Eletrônica Básica: Princípios de funcionamento de LEDs, resistores, botões e buzzers, incluindo conceitos como polaridade, limitação de corrente e debounce.

Lógica de Programação: Estruturas de controle (if/else, switch, for, while), variáveis, arrays e máquinas de estado finitas.

Conceito do Jogo Simon/Genius: Regras e mecânicas básicas do jogo de memória

Relatório do projeto final da disciplina de Sistemas Digitais da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção da aprovação na disciplina no curso de Engenharia Elétrica/Controle e Automação.