**Trabalho LIP (tinkercard e físico)**

**Integrantes:  
Daniel Bueno Lacerda**

**Matheus Felipe Cavalcanti**

**Etapas do projeto:**

Jogo gênios (Arduino físico)

Implementação de um sistema de automação residencial que controla a iluminação e a climatização de um ambiente (tinkercard)

**Planejamento:**

Pensamos em fazer o jogo gênios no Arduino físico, tivemos como base o projeto deste mesmo jogo que fizemos no tinkercard.

A partir disso, pensamos em fazer um projeto novo no tinkercard de um sistema de automação residencial que controla a iluminação e a climatização de um ambiente (primeiro exemplo).

**Montagem:**

A montagem do jogo gênios foi bem rápida, ligamos os fios, leds, resistores, botões, já o código colocamos no aplicativo e assim conseguimos fazer o jogo funcionar.

Já o outro projeto foi mais trabalhoso, mas após pensarmos muito conseguimos elaborar todo o projeto.

**Código:**

**Descrição breve**: O gênios é um jogo de memoria no qual o participante deve decorar a sequencia de luzes que vão acender, no caso os leds**.**

**O código do jogo gênios:**

int sequencia[100] = {};

int rodada\_atual = 0;

int passo\_atual\_na\_sequencia = 0;

int pinosLeds[4] = { 2, 4, 6, 8 };

int pinosBotoes[4] = { 3, 5, 7, 9 };

int botao\_pressionado = 0;

int perdeu\_o\_jogo = false;

void setup() {

for (int i = 0; i <= 3; i++) {

pinMode(pinosLeds[i], OUTPUT);

}

for (int i = 0; i <= 3; i++) {

pinMode(pinosBotoes[i], INPUT\_PULLUP);

}

randomSeed(analogRead(0));

}

void loop() {

if (perdeu\_o\_jogo) {

int sequencia[100] = {};

rodada\_atual = 0;

passo\_atual\_na\_sequencia = 0;

perdeu\_o\_jogo = false;

}

proximaRodada();

reproduzirSequencia();

aguardarJogador();

delay(1000);

}

void proximaRodada() {

int numero\_sorteado = random(0, 4);

sequencia[rodada\_atual++] = numero\_sorteado;

}

void reproduzirSequencia() {

for (int i = 0; i < rodada\_atual; i++) {

digitalWrite(pinosLeds[sequencia[i]], HIGH);

delay(500);

digitalWrite(pinosLeds[sequencia[i]], LOW);

delay(100);

}

}

void aguardarJogador() {

for (int i = 0; i < rodada\_atual; i++) {

aguardarJogada();

if (sequencia[passo\_atual\_na\_sequencia] != botao\_pressionado) {

gameOver();

}

if (perdeu\_o\_jogo) {

break;

}

passo\_atual\_na\_sequencia++;

}

passo\_atual\_na\_sequencia = 0;

}

void aguardarJogada() {

boolean jogada\_efetuada = false;

while (!jogada\_efetuada) {

for (int i = 0; i <= 3; i++) {

if (!digitalRead(pinosBotoes[i])) {

botao\_pressionado = i;

digitalWrite(pinosLeds[i], HIGH);

delay(300);

digitalWrite(pinosLeds[i], LOW);

jogada\_efetuada = true;

}

}

delay(10);

}

}

void gameOver() {

// GAME OVER

for (int i = 0; i <= 3; i++) {;

digitalWrite(pinosLeds[i], HIGH);

delay(200);

digitalWrite(pinosLeds[i], LOW);

}

perdeu\_o\_jogo = true;

}

**Descrição breve**: Como vimos, o projeto a seguir tem como proposito calcular a temperatura de um determinado ambiente. Ele funciona da seguinte forma: ao clicar o botão para iniciar o projeto e colocar a determinada temperatura no potenciômetro deve analisar o seguinte: se o led verde quando ligado, indica que a temperatura está boa entre 20 até no máximo 30 graus, já quando o led amarelo acende, a temperatura está entre 30 até 35 graus, enfim se o led vermelho acende quer dizer que a temperatura está muito alta (acima de 35 graus) e assim é disparado um alarme (piezo).

**O código do projeto do tinkercard:**

const int potPin = A0;

const int ldrPin = A1;

const int greenLedPin = 2;

const int yellowLedPin = 3;

const int redLedPin = 4;

const int buzzerPin = 5;

const int buttonPin = 6; // Pino do botão

const int whiteLedPin = 7; // Pino da luz branca

const int segA = 8; // Pinos do display de sete segmentos

const int segB = 9;

const int segC = 10;

const int segD = 11;

const int segE = 12;

const int segF = 13;

const int segG = A2;

bool systemState = false;

bool lastButtonState = HIGH;

unsigned long debounceDelay = 50;

void setup() {

pinMode(greenLedPin, OUTPUT);

pinMode(yellowLedPin, OUTPUT);

pinMode(redLedPin, OUTPUT);

pinMode(buzzerPin, OUTPUT);

pinMode(buttonPin, INPUT\_PULLUP);

pinMode(whiteLedPin, OUTPUT);

pinMode(segA, OUTPUT);

pinMode(segB, OUTPUT);

pinMode(segC, OUTPUT);

pinMode(segD, OUTPUT);

pinMode(segE, OUTPUT);

pinMode(segF, OUTPUT);

pinMode(segG, OUTPUT);

Serial.begin(9600);

}

void displayNumber(int num) {

switch (num) {

case 1: // Verde

digitalWrite(segA, LOW);

digitalWrite(segB, HIGH);

digitalWrite(segC, HIGH);

digitalWrite(segD, LOW);

digitalWrite(segE, LOW);

digitalWrite(segF, LOW);

digitalWrite(segG, LOW);

break;

case 2: // Amarelo

digitalWrite(segA, HIGH);

digitalWrite(segB, HIGH);

digitalWrite(segC, LOW);

digitalWrite(segD, HIGH);

digitalWrite(segE, HIGH);

digitalWrite(segF, LOW);

digitalWrite(segG, HIGH);

break;

case 3: // Vermelho

digitalWrite(segA, HIGH);

digitalWrite(segB, HIGH);

digitalWrite(segC, HIGH);

digitalWrite(segD, HIGH);

digitalWrite(segE, LOW);

digitalWrite(segF, LOW);

digitalWrite(segG, HIGH);

break;

default:

digitalWrite(segA, HIGH);

digitalWrite(segB, HIGH);

digitalWrite(segC, HIGH);

digitalWrite(segD, HIGH);

digitalWrite(segE, HIGH);

digitalWrite(segF, HIGH);

digitalWrite(segG, HIGH);

break;

}

}

void loop() {

bool reading = digitalRead(buttonPin);

// Verifica se o estado do botão mudou

if (reading != lastButtonState) {

delay(debounceDelay); // Delay para debounce

reading = digitalRead(buttonPin); // Lê novamente após o debounce

// Se ainda estiver diferente, indica que foi uma mudança válida

if (reading != lastButtonState) {

lastButtonState = reading; // Atualiza o último estado do botão

// Se o botão foi pressionado (mudou de HIGH para LOW)

if (reading == LOW) {

systemState = !systemState; // Alterna o estado do sistema

Serial.print("Botão pressionado, sistema ");

Serial.println(systemState ? "ligado" : "desligado");

}

}

}

// Atualiza os LEDs e outros componentes com base no estado do sistema

if (systemState) {

digitalWrite(whiteLedPin, HIGH); // Liga a luz branca

int potValue = analogRead(potPin);

int ldrValue = analogRead(ldrPin);

float temperature = map(potValue, 0, 1023, 20, 40);

int heatingPower = map(ldrValue, 0, 1023, 0, 255);

temperature += heatingPower \* 0.025;

if (temperature >= 20 && temperature <= 30) {

digitalWrite(greenLedPin, HIGH);

digitalWrite(yellowLedPin, LOW);

digitalWrite(redLedPin, LOW);

displayNumber(1); // Mostra o número 1 no display de sete segmentos

} else if (temperature >= 31 && temperature <= 35) {

digitalWrite(greenLedPin, LOW);

digitalWrite(yellowLedPin, HIGH);

digitalWrite(redLedPin, LOW);

displayNumber(2); // Mostra o número 2 no display de sete segmentos

} else if (temperature > 35) {

digitalWrite(greenLedPin, LOW);

digitalWrite(yellowLedPin, LOW);

digitalWrite(redLedPin, HIGH);

displayNumber(3); // Mostra o número 3 no display de sete segmentos

tone(buzzerPin, 1000, 500);

}

Serial.print("Temperatura da piscina: ");

Serial.print(temperature);

Serial.println(" °C");

Serial.print("Potencia de aquecimento: ");

Serial.print(heatingPower);

Serial.println(" (0-255)");

} else {

digitalWrite(whiteLedPin, LOW); // Desliga a luz branca

digitalWrite(greenLedPin, LOW);

digitalWrite(yellowLedPin, LOW);

digitalWrite(redLedPin, LOW);

noTone(buzzerPin);

displayNumber(0); // Desliga o display de sete segmentos quando o sistema está desligado

Serial.println("Todos os componentes desligados"); // Mensagem de debug

}

delay(100); // Pequeno delay para estabilidade do loop

}

**Resultados obtidos:**

Após termos feito todos estes passos, conseguimos entender mais a lógica do Arduino. Conseguimos fazer tudo da maneira como planejamos.