# Projeto: Jogo da Memória com PIC18F4520

Plataforma: MPLAB X IDE + XC8

Simulador: PicSimLab - Placa McLab2

Microcontrolador: PIC18F4520

# 1. Objetivo do Projeto

Desenvolver um jogo da memória interativo utilizando o PIC18F4520, que:

- Mostre sequências com LEDs coloridos: verde, vermelho, azul e amarelo;
- Receba entradas do jogador através dos botões RB0, RB1, RB2 e RB3, correspondentes a cada LED;
- Exiba a pontuação (nível atual) no display de 7 segmentos (valores de 0 a 9);
- Toque músicas no buzzer, com volume controlado por um potenciômetro analógico;
- Inicie o jogo detectando pouca luz via sensor LDR (gatilho de início).

# 2. Funcionamento do Jogo

#### 2.1 Início

- O sistema fica em modo espera, monitorando o sensor LDR;
- Quando o sensor detectar pouca luz (cobertura do LDR), o jogo inicia no nível 1.

#### 2.2 Rodada do Jogo

- O microcontrolador gera uma sequência aleatória de LEDs (começando por 1 LED no nível 1);
- Cada LED da sequência acende, um por vez, com um pequeno intervalo;
   O jogador deve repetir a sequência pressionando os botões correspondentes (RB0 a RB3);
- O sistema verifica a entrada do jogador a cada botão pressionado.

# 2.3 Validação

- Se a sequência for repetida corretamente:
  - Incrementa o nível (pontuação) em 1;
  - Atualiza o display de 7 segmentos com a nova pontuação;
  - Gera uma nova sequência (aumentando a quantidade de LEDs);
- Se o jogador errar algum botão:

- o Toca uma música de erro no buzzer;
- Reseta o jogo para o nível 0 (espera pelo sensor LDR novamente).

## 3. Final do Jogo

- O jogo termina com o jogador alcançando o nível 9 (sequência de 9 LEDs);
- Opcionalmente, pode tocar uma música de vitória e reiniciar o jogo.

### 4. Componentes Utilizados

- PIC18F4520;
- LEDs: verde, vermelho, azul e amarelo;
- Botões: RB0, RB1, RB2, RB3;
- Display de 7 segmentos para mostrar pontuação (0 a 9);
- Buzzer para sons/músicas;
- Potenciômetro para controle de volume do buzzer;
- Sensor LDR para gatilho de início do jogo.

### 5. Observações

- Sequência aleatória gerada usando um algoritmo simples de RNG interno;
- Tempo de resposta para pressionar botões será limitado para tornar o jogo desafiador;
- Volume do buzzer ajustado de acordo com a leitura do potenciômetro analógico;
- O display de 7 segmentos suporta apenas números de 0 a 9 para facilitar a visualização da pontuação.

# 6. Códigos

#### //main.c

```
#include <xc.h>
#include "game.h"
#include <stdint.h>
#define _XTAL_FREQ 8000000

// CONFIG
#pragma config OSC = HS  // Oscillator Selection bits (HS oscillator)
#pragma config WDT = OFF  // Watchdog Timer (WDT disabled)
```

```
#pragma config LVP = OFF  // Low Voltage ICSP (Disabled)
#pragma config PBADEN = OFF // PORTB<4:0> digital on reset
#pragma config BOREN = OFF // Brown-out Reset disabled
#pragma config MCLRE = ON // MCLR pin enabled
#define _XTAL_FREQ 8000000 // Frequência do oscilador
void main(void) {
  game_init();
  while (1) {
    game_loop();
  }
}
//game.c
#include "game.h"
#include "leds.h"
#include "buttons.h"
#include "display.h"
#include "buzzer.h"
#include "sensors.h"
#include <xc.h>
#include <stdint.h>
#define XTAL FREQ 8000000
#define MAX_LEVEL 9
static uint8 t sequence[MAX LEVEL];
static uint8_t current_level;
void generate_sequence(void) {
  for (uint8_t i = 0; i < MAX_LEVEL; i++) {
     sequence[i] = rand() % 4;
  }
}
void game_init(void) {
  leds_init();
  buttons_init();
  display_init();
  buzzer init();
  sensors_init();
  game_reset();
}
void game_reset(void) {
```

```
current level = 1;
  display_show_digit(0);
  generate_sequence();
}
void game_loop(void) {
  if (!ldr_dark_detected()) return;
  __delay_ms(500); // Debounce do LDR
  for (uint8_t i = 0; i < current_level; i++) {
     leds_flash(sequence[i]);
     __delay_ms(300);
  }
  for (uint8_t i = 0; i < current_level; i++) {
     int8_t btn = -1;
     while (btn == -1) {
       btn = buttons_read();
     }
     leds_flash(btn);
     if (btn != sequence[i]) {
       buzzer_play_error();
       game_reset();
       return;
     }
  }
  buzzer_play_success();
  display_show_digit(current_level);
  if (current_level < MAX_LEVEL) {</pre>
     current_level++;
     __delay_ms(800);
  } else {
     // Vitória: reiniciar
     __delay_ms(1000);
     game_reset();
  }
}
//game.h
#include <stdint.h>
#ifndef GAME_H
#define GAME_H
void game_init(void);
void game_loop(void);
```

```
void game_reset(void);
#endif
//leds.c
#include "leds.h"
#include <xc.h>
#include <stdint.h>
#define _XTAL_FREQ 8000000
void leds_init(void) {
  TRISD &= 0xF0; // RD0-RD3 como saída
  LATD &= 0xF0; // Apaga LEDs
}
void leds_flash(uint8_t index) {
  LATD = (1 << index);
   __delay_ms(300);
  LATD = 0;
}
void leds_show_sequence(uint8_t* seq, uint8_t len) {
  for (uint8_t i = 0; i < len; i++) {
     leds_flash(seq[i]);
     __delay_ms(200);
  }
}
//leds.h
#include <stdint.h>
#ifndef LEDS_H
#define LEDS_H
#include <stdint.h>
void leds_init(void);
void leds_flash(uint8_t index);
void leds_show_sequence(uint8_t* seq, uint8_t len);
#endif
//buttons.c
#include "buttons.h"
#include <xc.h>
#include <stdint.h>
#define _XTAL_FREQ 8000000
```

```
void buttons_init(void) {
  TRISB |= 0x0F; // RB0-RB3 como entrada
}
int8_t buttons_read(void) {
  if (!PORTBbits.RB0) return 0;
  if (!PORTBbits.RB1) return 1;
  if (!PORTBbits.RB2) return 2;
  if (!PORTBbits.RB3) return 3;
  return -1;
}
//buttons.h
#include <stdint.h>
#ifndef BUTTONS_H
#define BUTTONS_H
void buttons_init(void);
int8_t buttons_read(void);
#endif
//display.c
#include "display.h"
#include <xc.h>
#include <stdint.h>
#define _XTAL_FREQ 8000000
const uint8_t digit_map[10] = {
  0x3F, 0x06, 0x5B, 0x4F,
  0x66, 0x6D, 0x7D, 0x07,
  0x7F, 0x6F
};
void display_init(void) {
  TRISC = 0x00;
  LATC = 0x00;
}
void display_show_digit(uint8_t digit) {
  LATC = digit_map[digit % 10];
}
//display.h
#include <stdint.h>
```

```
#ifndef DISPLAY_H
#define DISPLAY_H
#include <stdint.h>
void display_init(void);
void display_show_digit(uint8_t digit);
#endif
//buzzer.c
#include "buzzer.h"
#include "sensors.h"
#include <xc.h>
#include <stdint.h>
#define _XTAL_FREQ 8000000
void buzzer_init(void) {
  TRISEbits.TRISE0 = 0;
  LATEbits.LATE0 = 0;
}
void buzzer_play_success(void) {
  LATEbits.LATE0 = 1;
   _delay_ms(100);
  LATEbits.LATE0 = 0;
}
void buzzer_play_error(void) {
  for (int i = 0; i < 3; i++) {
     LATEbits.LATE0 = 1;
       _delay_ms(100);
    LATEbits.LATE0 = 0;
     __delay_ms(100);
  }
}
//buzzer.h
#include <stdint.h>
#ifndef BUZZER H
#define BUZZER_H
void buzzer_init(void);
void buzzer_play_success(void);
void buzzer_play_error(void);
void buzzer_set_volume(uint8_t level);
```

#endif

```
//sensors.c
#include "sensors.h"
#include <xc.h>
#include <stdint.h>
#define _XTAL_FREQ 8000000
void sensors init(void) {
  TRISAbits.TRISA0 = 1; // Potenciômetro
  TRISAbits.TRISA1 = 1; // LDR
  ADCON1 = 0x0E;
                       // AN0 e AN1 analógicos
}
uint16_t adc_read(uint8_t channel) {
  ADCON0 = (channel << 2) | 0x01; // Canal e liga ADC
    delay us(10);
  GO_nDONE = 1;
  while (GO_nDONE);
  return ((ADRESH << 8) + ADRESL);
}
uint8_t ldr_dark_detected(void) {
  return adc_read(1) < 128;
}
uint8 t read potentiometer(void) {
  return adc_read(0) >> 2; // Converte para 8 bits
}
//sensors.h
#include <stdint.h>
#ifndef SENSORS_H
#define SENSORS_H
#include <stdint.h>
void sensors_init(void);
uint8_t ldr_dark_detected(void);
uint8 t read potentiometer(void);
```

# 7. Configuração dos elementos

## **Botões:**

RB0- Botão para led vermelho

RB1- Botão para led verde

RB2- Botão para led azul

RB3- Botão para led amarelo

#### Leds:

RD0- led vermelho

RD1- led verde

RD2- led azul

RD3- led amarelo

# Display de 7 segmentos:

RC0- Segmento A

RC1- Segmento B

RC2- Segmento C

RC3- Segmento D

RC4- Segmento E

RC5- Segmento F

RC6- Segmento G

RC7- Point

#### Potenciômetro:

RA1- POT 2 (Entrada analógica)

#### LDR:

RA0- 3 AO (Entrada analógica)

#### **Buzzer:**

**RE0 - PIN 1** 

8. Conclusão

O desenvolvimento do jogo da memória utilizando o microcontrolador PIC18F4520 permitiu aplicar na prática diversos conceitos de eletrônica digital e programação embarcada. A integração de periféricos como LEDs, botões, buzzer, display de 7 segmentos, sensor LDR e potenciômetro demonstrou a versatilidade do microcontrolador no controle de dispositivos externos e na interação com o usuário.

Além de reforçar habilidades técnicas, o projeto proporcionou uma experiência lúdica e interativa, estimulando o raciocínio lógico e a memória. Com isso, conclui-se que o projeto foi bem-sucedido tanto em seu funcionamento quanto em seu valor didático.