Sistema de Alarme Inteligente

Professor: Sales

Dupla: Henrique Nunes; Antonio Victor Gonçalves

IFCE – CEDRO

Componentes Utilizados:

Nome	Quantidade	Componente
D1	1	/Vermelho LED
R1 R2	2	/220 OHMS Resistor
Rpot1	1	/10 kΩ Potenciômetro
U1	1	/LCD 16 x 2
U2	1	/Arduino Uno R3
PIEZO1	1	/Piezo
PIR1	1	/Sensor PIR

Explicação do Código, e suas bibliotecas

Inclusão de Bibliotecas

```
#include <avr/io.h>
#include <util/delay.h>
```

★ <avr/io.h>: Define os registradores de I/O do microcontrolador (ex.: PORTD, DDRD).
★ <util/delay.h>: Fornece funções de atraso como _delay_ms().

Definições dos Pinos

```
#define LCD_RS 13
#define LCD_EN 12
#define LCD_D4 6
#define LCD_D5 5
#define LCD_D6 3
#define LCD_D7 2
#define LED_PIN 7
#define PIR_PIN 4
#define BUZZER_PIN 8
```

★ Define os pinos usados para:
  ★ **LCD**: RS (registro de seleção), EN (enable), D4-D7 (dados).
  ★ **LED**: Pino 7.
  ★ **Sensor PIR (movimento)**: Pino 4 (entrada).
  ★ **Buzzer**: Pino 8.

Protótipos de Função

```
void lcd_command(uint8_t command);
void lcd_write(uint8_t data);
```

```
void lcd_init(void);
void lcd_print(const char *str);
void lcd_set_cursor(uint8_t col, uint8_t row);
void lcd_clear(void);
```

★ Declara as funções para controlar o LCD (implementadas mais tarde).

Função Principal (main**)**

```
int main(void) {
```

★ Início do programa.

Configuração dos Pinos

```
DDRD |= (1 << LED_PIN) | (1 << BUZZER_PIN); // LED e Buzzer como saída
DDRD &= ~(1 << PIR_PIN); // PIR como entrada
```

★ DDRD: Define a direção dos pinos do PORTD.
- LED_PIN (7) e BUZZER_PIN (8) como saída (1).
- PIR_PIN (4) como entrada (0).

Inicialização do LCD

```
lcd_init();
```

★ Chama a função que configura o LCD.

Loop Infinito (while(1)**)**

```
while(1) {
```

★ O programa roda indefinidamente.

Leitura do Sensor PIR

```
uint8_t pir_status = PIND & (1 << PIR_PIN);
```

★ Lê o pino do sensor PIR (PIND é o registrador de entrada do PORTD).
★ Se houver movimento, pir_status será diferente de zero.

Se Movimento Detectado (if (pir_status)**)**

```
if (pir_status) {
```

★ Se o PIR detectar movimento:

Limpa o LCD

```
lcd_clear();
```

Liga o LED e o Buzzer

```
PORTD |= (1 << LED_PIN); // Liga LED
PORTD |= (1 << BUZZER_PIN); // Liga Buzzer
```

★ Escreve 1 nos pinos do LED e do buzzer.

Gera um Beep (simulação de tone())

```
for (int i = 0; i < 30; i++) {
  PORTD ^= (1 << BUZZER_PIN); // Alterna o estado do buzzer
  _delay_ms(1);
}
```

★ Alterna o buzzer rapidamente (30 vezes) para criar um som.

Exibe "ALERTA MOVIMENTO" no LCD

```
lcd_set_cursor(0, 1); // Segunda linha, primeira coluna
lcd_print("ALERTA MOVIMENTO");
_delay_ms(3000); // Espera 3 segundos
lcd_clear(); // Limpa o LCD
```

Se Nada for Detectado (else**)**

```
} else {
```

★ Caso não haja movimento:

Exibe "NADA DETECTADO"

```
lcd_set_cursor(0, 0); // Primeira linha, primeira coluna
lcd_print("NADA DETECTADO");
```

Desliga LED e Buzzer

```
PORTD &= ~((1 << LED_PIN) | (1 << BUZZER_PIN)); // Desliga ambos
```

Atraso de 1 Segundo

```
_delay_ms(1000);
```

★ Evita leitura muito rápida do sensor.

Funções do LCD (Implementação Simplificada)

lcd_command **(Envia Comando ao LCD)**

```
void lcd_command(uint8_t command) {
  // (Código específico para controlar o LCD)
}
```

★ Envia comandos como limpar tela, mover cursor, etc.

lcd_write **(Envia Dados ao LCD)**

```
void lcd_write(uint8_t data) {
  // Envia caracteres para o LCD
}
```

★ Usado para imprimir texto.

lcd_init **(Inicializa o LCD)**

```
void lcd_init(void) {
  _delay_ms(50); // Espera inicialização
  // Configuração do modo 4 bits, liga display, etc.
}
```

★ Configura o LCD para operação.

lcd_print **(Imprime String)**

```
void lcd_print(const char *str) {
  while (*str) {
    lcd_write(*str++); // Escreve cada caractere
  }
}
```

★ Imprime uma string no LCD.

lcd_set_cursor **(Posiciona Cursor)**

```
void lcd_set_cursor(uint8_t col, uint8_t row) {
  uint8_t address = (row == 0) ? 0x80 : 0xC0; // Linha 0 ou 1
  address += col; // Adiciona coluna
  lcd_command(address); // Move cursor
}
```

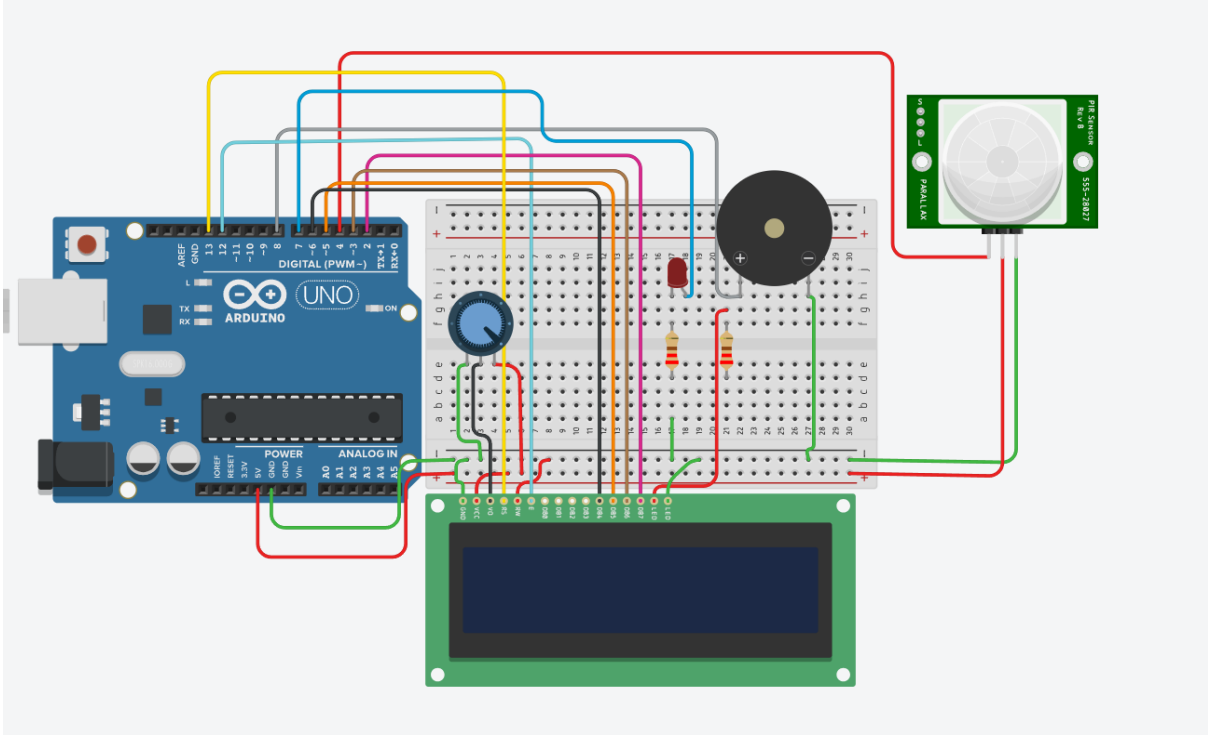
★ Posiciona o cursor em (col, row).

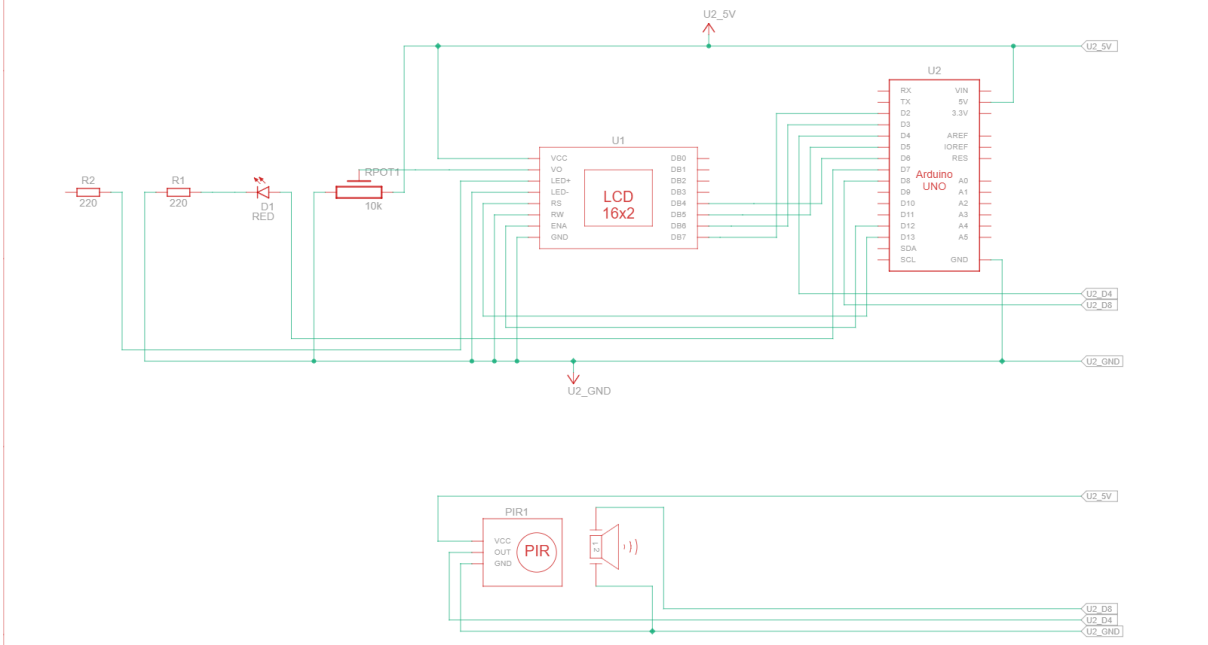
lcd_clear **(Limpa o LCD)**

```
void lcd_clear(void) {
  lcd_command(0x01); // Comando para limpar
```

```
    _delay_ms(2);                    //                Espera                execução
}
```

★ Apaga todo o conteúdo do LCD.





Resumo do Funcionamento

1. Configura os pinos do LED, buzzer e PIR.
2. Inicializa o LCD.
3. Em loop:
    a. Lê o sensor PIR.
    b. Se detectar movimento:
        i. Liga LED e buzzer.
        ii. Mostra "ALERTA MOVIMENTO".
    c. Se não detectar:
        i. Desliga LED/buzzer.
        ii. Mostra "NADA DETECTADO".
    d. Repete a cada 1 segundo.