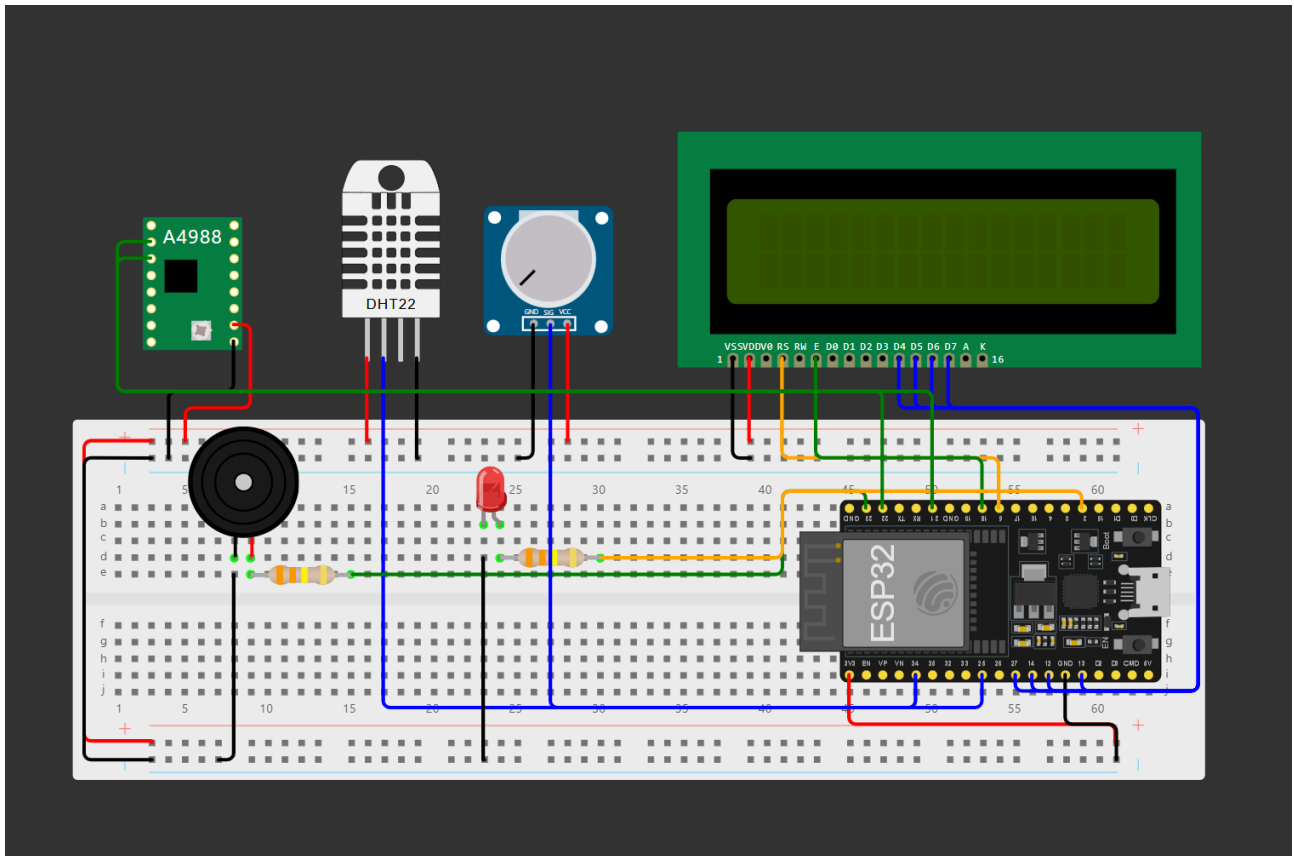


Esquema de Conexões



Link do simulador: <https://wokwi.com/projects/427981743622261761>

Atuadores, sensores e conexões no ESP32:

- **DHT11:** Conectado ao pino 25, VCC e GND
- **Potenciômetro (batimentos cardíacos):** Conectado ao pino 34 (analógico), VCC e GND
- **Sensor RFID:** SDA conectado ao pino 22, SCL conectado ao pino 21, VCC e GND
- **Buzzer:** Conectado ao pino 23 e GND
- **LED vermelho:** Conectado ao pino 2 e GND
- **Display LCD:** Conectado ao VCC e GND
 - RS conectado ao pino 5
 - E conectado ao pino 18
 - D4 conectado ao pino 13
 - D5 conectado ao pino 12
 - D6 conectado ao pino 14



- **D7** conectado ao pino 27

Codificação

Código do sistema atualizado:

```
#include <Wire.h>
#include <Adafruit_PN532.h>
#include <LiquidCrystal.h>
#include <DHT.h>

//Definição dos pinos do display LCD
#define RS 5
#define EN 18
#define D4 13
#define D5 12
#define D6 14
#define D7 27

//Definição dos atuadores
#define BUZZER 23
#define LED_VERMELHO 2

// Definição dos pinos dos sensores DHT11 e BPM (potenciômetro)
#define DHTPIN 25
#define DHTTYPE DHT11
#define POT 34

// Definição dos pinos do PN532 (NFC)
#define SDA_PIN 22
#define SCL_PIN 21

bool alarmeAtivo = false;

// Inicialização do módulo PN532 (I2C padrão)
Adafruit_PN532 nfc(SDA_PIN, SCL_PIN);

// Inicializa a biblioteca do LCD
LiquidCrystal lcd(RS, EN, D4, D5, D6, D7);
```



```
// Inicializa o sensor DHT11
DHT dht(DHTPIN, DHTTYPE);

const String cartoesAutorizados[] = {"f3 fa aa d"};
const int numCartoes = sizeof(cartoesAutorizados) / sizeof(cartoesAutorizados[0]);

unsigned long ultimaLeituraRFID = 0;
const unsigned long intervaloLeituraRFID = 2000; // Intervalo de 2 segundos para ler RFID

void setup() {
    Serial.begin(115200);

    pinMode(LED_VERMELHO, OUTPUT);
    pinMode(BUZZER, OUTPUT);
    pinMode(POT, INPUT);
    dht.begin();
    lcd.begin(20, 4);

    nfc.begin();
    uint32_t versiondata = nfc.getFirmwareVersion();
    if (!versiondata) {
        Serial.println("Falha ao encontrar PN532");
        while (1);
    }
    nfc.SAMConfig();
}

// Função para verificar se um cartão está autorizado
bool verificarCartao(String cardID) {
    for (int i = 0; i < numCartoes; i++) {
        if (cartoesAutorizados[i] == cardID) {
            return true;
        }
    }
    return false;
}
```



```
void lerRFID() {

    uint8_t uid[] = {0, 0, 0, 0, 0, 0, 0};
    uint8_t uidLength;

    if (nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength)) {
        String cardID = "";
        for (uint8_t i = 0; i < uidLength; i++) {
            cardID += String(uid[i], HEX);
            if (i < uidLength - 1) cardID += " ";
        }

        Serial.print("ID do Cartao RFID: ");
        Serial.println(cardID);

        if (verificarCartao(cardID)) {
            Serial.println("Cartão autorizado! Desligando alarmes.");
            lcd.clear();
            lcd.print("Acesso Liberado!");

            // Desligar buzzer e LED
            digitalWrite(BUZZER, LOW);
            digitalWrite(LED_VERMELHO, LOW);
            alarmeAtivo = false;
        } else {
            Serial.println("Cartão não autorizado.");
            lcd.clear();
            lcd.print("Acesso Negado!");
        }
    }
}

void loop() {
    float temperatura = dht.readTemperature();
    int leituraPot = analogRead(POT);
    float bpm = map(leituraPot, 0, 4095, 0, 4000); // Mapeia para uma faixa de 60 a 100
```



```
if (isnan(temperatura) && (isnan(leituraPot))) { // Verifica se a leitura falhou
    lcd.setCursor(0, 1);
    lcd.print("Erro no sensor!");
} else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Monitorando...");
    lcd.setCursor(0, 1);
    lcd.print("Temp: ");
    lcd.print(temperatura);
    lcd.print(" C");
    lcd.setCursor(0, 2);
    lcd.print("Bat.Card.: ");
    lcd.print(leituraPot);
    lcd.print(" bpm");
}

// Condições para ativar o alarme
if ((temperatura > 27 || temperatura < 20 || leituraPot > 2000) && !alarmeAtivo) {
    Serial.println("Alerta! Condições críticas detectadas.");
    digitalWrite(BUZZER, HIGH);
    digitalWrite(LED_VERMELHO, HIGH);
    alarmeAtivo = true;
}

if((temperatura < 27 && temperatura > 20 && leituraPot < 2000) && alarmeAtivo) {
    Serial.println("Condições normalizadas. Desligando alarme.");
    digitalWrite(BUZZER, LOW);
    digitalWrite(LED_VERMELHO, LOW);
    alarmeAtivo = false;
}

if (alarmeAtivo && millis() - ultimaLeituraRFID >= intervaloLeituraRFID) {
    lerRFID();
    ultimaLeituraRFID = millis();
}
```



```
// Exibe temperatura e batimentos cardíacos
```

```
Serial.print("Temp: ");
```

```
Serial.print(temperatura);
```

```
Serial.println(" C");
```

```
Serial.print("Batimento cardíaco: ");
```

```
Serial.print(leituraPot);
```

```
Serial.println(" bpm");
```

```
// Ler RFID para possível desativação do alarme
```

```
delay(1000);
```

```
}
```

```
/*Parâmetros para ativar/desativar buzzer e LED
```

```
Serial.print("Temperatura: ");
```

```
Serial.println(temperatura);
```

```
if (temperatura < 20 || temperatura > 25) {
```

```
    digitalWrite(LED_VERMELHO, HIGH); // Liga o LED
```

```
    digitalWrite(BUZZER, HIGH);      // Ativa o buzzer
```

```
} else {
```

```
    digitalWrite(LED_VERMELHO, LOW); // Desliga o LED
```

```
    digitalWrite(BUZZER, LOW);      // Desativa o buzzer
```

```
}
```

```
delay(2000);
```

```
Serial.print("BPM: ");
```

```
Serial.println(leituraPot);
```

```
if (leituraPot < 60 || leituraPot > 100) {
```

```
    digitalWrite(LED_VERMELHO, HIGH); // Liga o LED
```

```
    digitalWrite(BUZZER, HIGH);      // Ativa o buzzer
```

```
} else {
```

```
    digitalWrite(LED_VERMELHO, LOW); // Desliga o LED
```

```
    digitalWrite(BUZZER, LOW);      // Desativa o buzzer
```

```
}
```

```
delay(2000);*/
```

```
/*Mensagens exibidas no Serial Monitor
```



```
Serial.println("Aguardando RFID...");
if (nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength)) {
    Serial.print("Cartao detectado: ");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ID: "); //Exibe no LCD, o ID do cartão detectado
    lcd.setCursor(0, 1);
    for (uint8_t i = 0; i < uidLength; i++) {
        Serial.print(uid[i], HEX);
        Serial.print(" ");
        lcd.print(uid[i], HEX);
        lcd.print(" ");
    }
    delay(2000);
    setup();
}*/
```