Trabalho Final - Tópicos Especiais em Sistemas Embarcados

Daniel Porto

4 de julho de 2025

Introdução

O presente trabalho tem como objetivo o desenvolvimento de um sistema embarcado capaz de controlar três saídas digitais em tempo real por meio de uma interface web e de um sensor de toque capacitivo. A aplicação envolve conceitos de redes Wi-Fi, programação embarcada e comunicação assíncrona com o uso de AJAX.

Placa de Microcontrole

A placa utilizada é a NodeMCU, que possui o microcontrolador ESP8266. Esta placa conta com suporte a Wi-Fi integrado, diversos pinos digitais e um pino analógico. A NodeMCU é compatível com a IDE do Arduino, embora o mapeamento dos pinos do ESP8266 difira dos números marcados na placa física.

Especificações básicas do ESP8266

- Processador: Tensilica L106 32-bit
- Frequência: 80 MHz (pode ser overclockada para 160 MHz)
- Memória Flash: até 4 MB (dependendo do modelo)
- Conectividade: Wi-Fi 802.11 b/g/n
- Tensão de operação: 3.3V

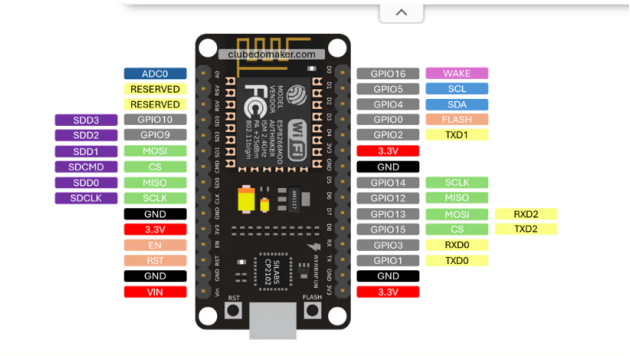
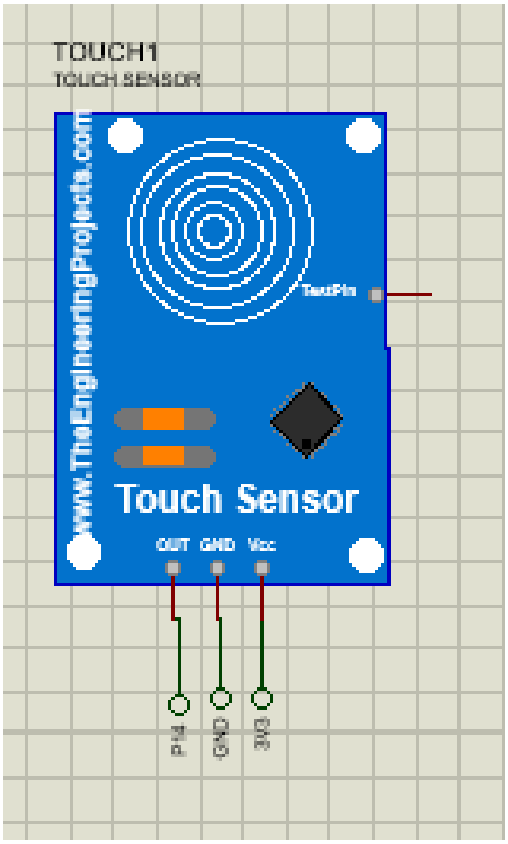


Diagrama de pinos do ESP8266 (Fonte: Clube do Maker)

Sensor de Toque

Foi utilizado um sensor capacitivo de toque compatível com Arduino, composto por três pinos: sinal (data), alimentação (VCC) e terra (GND). Sua função é detectar o toque humano de forma sensível e rápida.



Sensor capacitivo de toque (Fonte: elaborada pelo autor)

Componentes Utilizados

- 1 buzzer

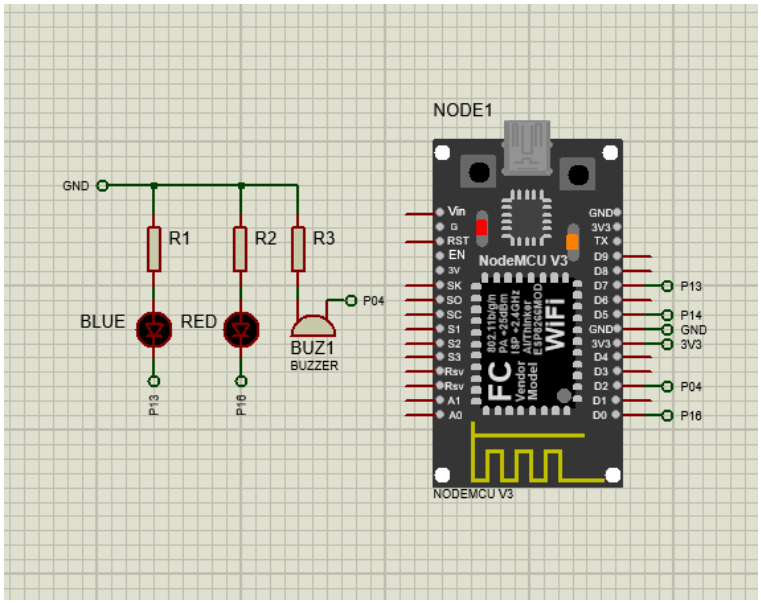
- 1 LED azul
- 1 LED vermelho

Descrição do Projeto

A interface web permite controlar individualmente cada uma das saídas digitais — LEDs e buzzer — além de mostrar em tempo real o estado do sensor de toque. A lógica principal implementa uma condição de segurança: o LED azul só pisca quando o sensor é tocado e o LED vermelho está ligado. Caso contrário, o buzzer é acionado, funcionando como um alarme.

Esquemático e Construção

O esquemático do projeto foi elaborado no Proteus.



Esquemático do projeto (Fonte: elaborada pelo autor)

Foram utilizados 4 pinos digitais: 3 como saída e 1 como entrada:

```
#define pinSensor 14     // Sensor no D5
#define FreeSensor 16    // LED vermelho (condição) no D0
#define pinBuzzer 4      // Buzzer no D2
#define pinLed 13        // LED azul no D7

byte pinos_io[QNT_PINOS] = {pinLed, pinSensor, pinBuzzer, FreeSensor};
byte setup_pinos[QNT_PINOS] = {OUTPUT, INPUT, OUTPUT, OUTPUT};
```

Bibliotecas Utilizadas

```
#include <ESP8266WiFi.h>
#include <Arduino.h>
```

A biblioteca `ESP8266WiFi.h` é responsável pela comunicação via rede, enquanto a `Arduino.h` oferece funções básicas de controle de hardware.

Fluxo de Execução do Projeto

Setup e Conexão Wi-Fi

```
void conexao_wifi() {
  Serial.begin(115200);
  Serial.println();
  Serial.print("Conectando a ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("WiFi conectado!");
  server.begin();
  Serial.println("Server iniciado");
  Serial.println(WiFi.localIP());
}
```

O `localIP()` representa o endereço IP atribuído ao ESP8266 na rede local, que deve ser usado para acessar a página web. Esse IP pode ser visualizado no monitor serial, nas configurações do roteador (se for hotspot) ou pelo comando `ipconfig` no terminal do computador conectado à mesma rede.

Loop Principal e Página Web

A função `carrega_pagina` é responsável por lidar com as requisições HTTP feitas pelo navegador. Ela atende a duas situações principais: o carregamento da página HTML e as requisições AJAX para atualização de estado dos pinos.

Pseudo-código da função carrega_pagina

```
Função carrega_pag:
    Espera conexão de cliente HTTP
```

```
Se cliente conectado:
    Enquanto conexão ativa:
        Se há dados disponíveis:
            Lê requisição caractere a caractere
            Monta string de requisição

            Se fim do cabeçalho:
                Se página principal requisitada:
                    Envia HTML com JavaScript
                Se requisição AJAX:
                    Envia estado das portas
                Senão:
                    Responde com cabeçalho vazio

                Limpa requisição e encerra

    Aguarda 1ms e encerra a conexão
```

Função processaPorta

Responsável por configurar saídas conforme os dados da página:

```
void processaPorta(byte porta, byte posicao, WiFiClient cl) {
  String cHTML = "P" + String(porta) + "=" + String(porta);

  if (modoPinos[posicao] == OUTPUT) {
    LED_status = (URLValue.indexOf(cHTML) > -1) ? HIGH : LOW;
    digitalWrite(porta, LED_status);
  } else {
    LED_status = digitalRead(porta);
  }

  // Envia checkbox com estado da porta
}
```

Função lePortaDigital

Responsável por enviar atualizações via AJAX:

```
void lePortaDigital(byte porta, byte posicao, WiFiClient cl) {
  if (modoPinos[posicao] == INPUT) {
    cl.print("PD" + String(porta) + "#" +
             String(digitalRead(porta)) + "|");
  }
}
```

Controle do LED e Buzzer

```
void controla_led(int pinBuzzer, int FreeSensor, int pinSensor, int pinLed) {
  int sensorValue = digitalRead(pinSensor);

  if (sensorValue == HIGH && libera_sensor(FreeSensor)) {
    blinka_led(pinLed);
  } else if (sensorValue == HIGH) {
    digitalWrite(pinBuzzer, HIGH);
  } else {
    digitalWrite(pinBuzzer, LOW);
  }
}
```

Conclusão

O projeto permitiu explorar conceitos de microcontroladores utilizando ESP8266, sensores e interface web. A integração entre hardware e software embarcado foi feita com sucesso, proporcionando controle e monitoramento dinâmico dos dispositivos conectados.

Referências

- https://clubedomaker.com/esp8266-pinout
- https://portal.vidadesilicio.com.br/sensor-touch-capacitivo-arduino/
- https://www.alldatasheet.com/view.jsp?Searchword=Esp8266%20datasheet