

ESP12F_RELAY_X4

Traduzido por AthenasArch

Placa de desenvolvimento “ESP12F Relay x4” com entrada para Fonte de alimentação AC/DC utilizando o chip ESP8266-12, com módulo WIFI e conjunto de quatro relés.

VISÃO GERAL

A placa de desenvolvimento “**ESP12F Relay x4**” é equipada com módulo WiFi ESP-12F, todos os pinos de Entrada/Saída estão disponíveis para utilização de qualquer forma. A placa conta com uma fonte AC integrada, com capacidade de 90 a 250VAC / DC 7 a 30VDC / 5V e outros métodos de fonte de alimentação. É possível utilizar o ambiente de desenvolvimento Arduino, adequado para aprendizado e desenvolvimento com ESP8266, controle sem fio residencial inteligente e outras aplicações.

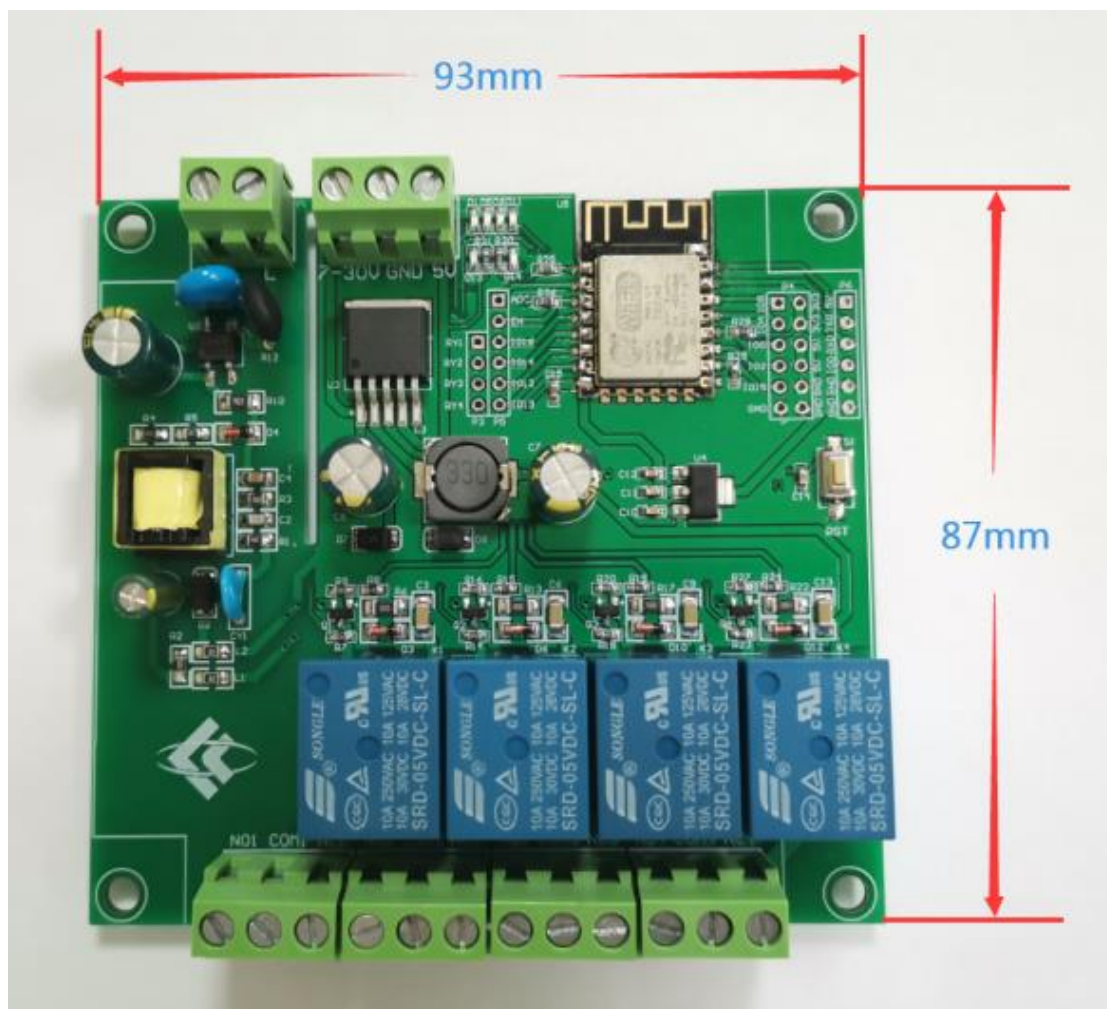
CARACTERÍSTICAS

- 1** - Módulo ESP-12F WiFi estável integrado, com memória Flash de 4M Byte;
- 2** - As portas I/O e a porta de download do programa (através da UART) estão com fácil acesso, para facilitar o desenvolvimento;
- 3** - Placa possui circuito para alimentação AC-DC integrada, é possível alimentar a placa através de:
 - a) Tensão alternada da rede (tomada) AC – de 90 a 250VAC;
 - b) Fonte de alimentação DC 7 a 30VDC;
 - c) Entrada por USB ou fonte de 5V;
- 4** - O botão de reinicialização RST do módulo WiFi integrado;
- 5** - ESP-12F oferece suporte a ferramentas de desenvolvimento como Eclipse / Arduino IDE e fornece programas de referência no ambiente de desenvolvimento Arduino;
- 6** - Existem **4 relés** de 5V na placa, que emitem sinais de contato seco NA/NF/COM (Aberto, fechado e comum), adequados para controlar cargas cuja tensão de trabalho está dentro de AC 250V – 10A / DC30V - 10A;
- 7** - Indicador de fonte integrado, 1 LED programável e indicador de relé;

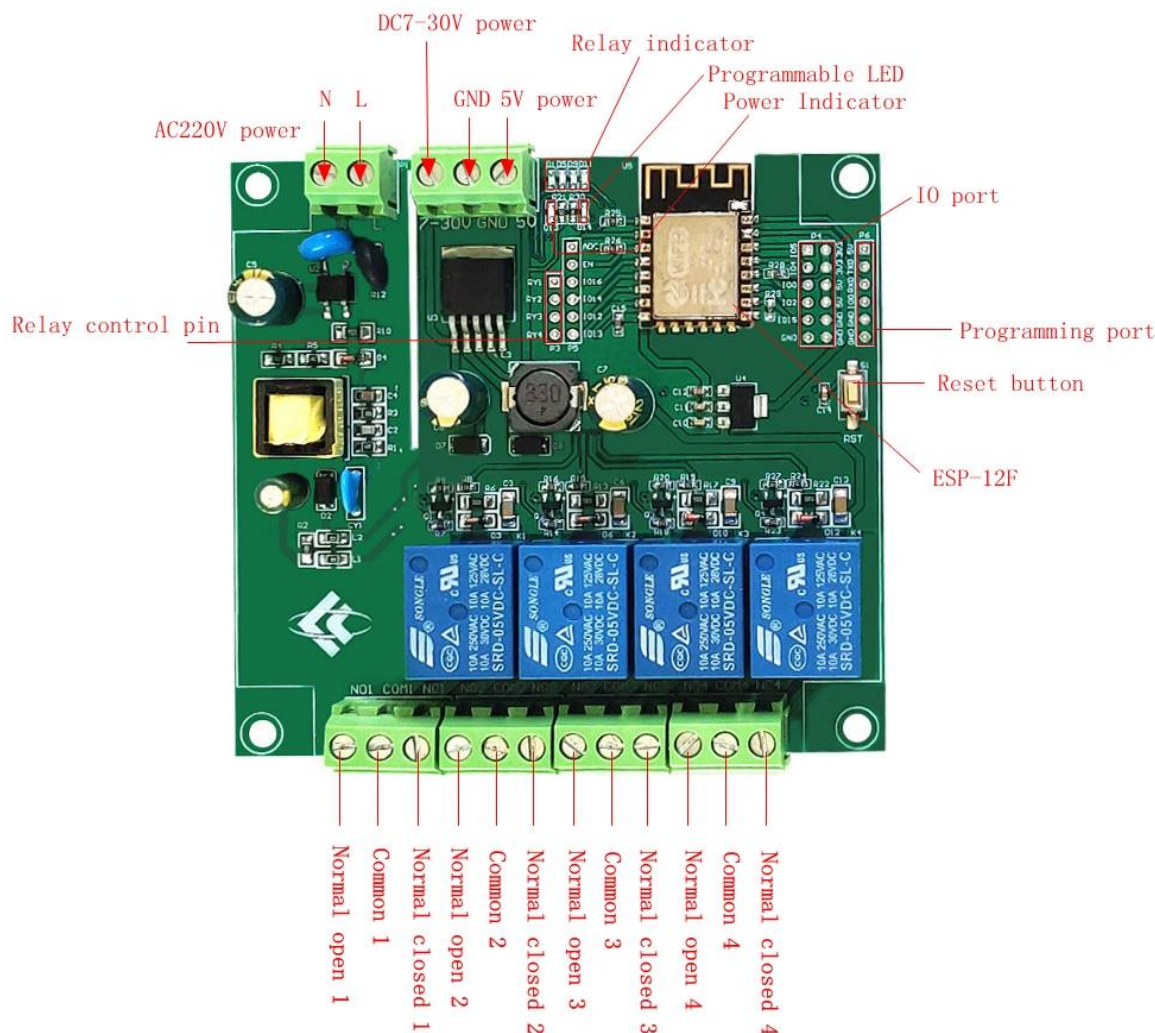
INTRODUÇÃO E DESCRIÇÃO DO HARDWARE

- Dimensões da placa: 93 x 87mm.

- Peso: 90g.



INTERFACE DE OPERAÇÃO



PROGRAMAÇÃO DA PLACA

Pinos de programação no conector **P6**: GND, GND, RX, TX e 5V do ESP8266

IO0 precisa ser conectado a GND durante o download e, em seguida, desconecte a conexão entre IO0 e GND após a conclusão do download;

SAÍDA DE RELÉ

NO: Normalmente aberto (NA), este pino está aberto em relação ao comum até o relé ser acionado;

COM: Comum (COM), este pino pode ser usado com o NO ou com o NC;

NC: Normalmente fechado (NC), este pino está em curto com o comum até o relé ser acionado;

2. Introduction to GPIO port

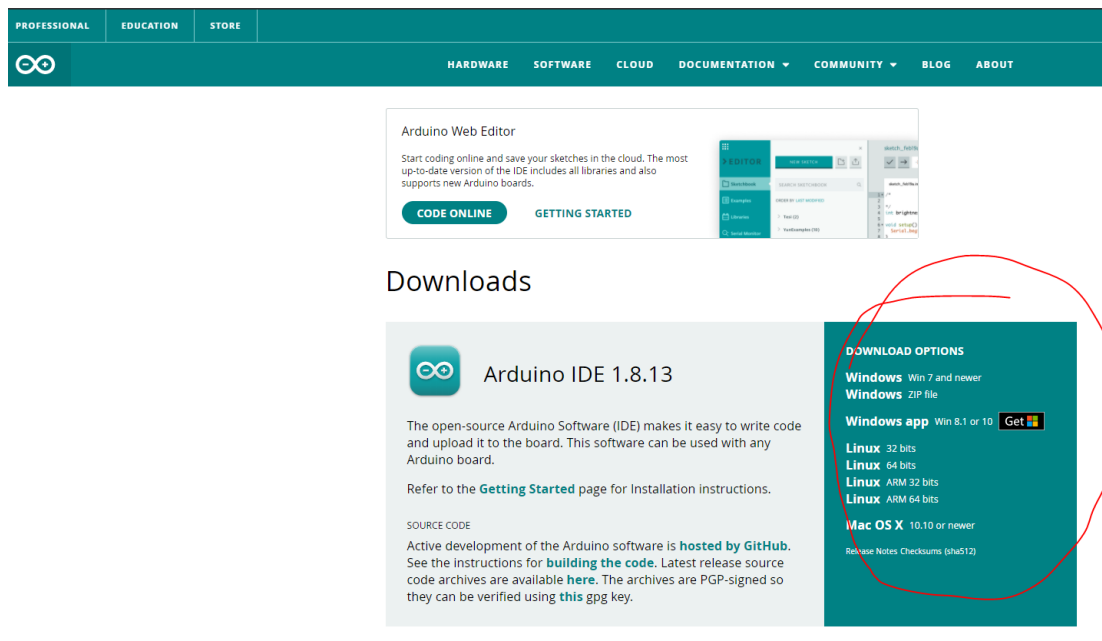
Num.	Descrição	Função	Num.	Descrição	Função
1	ADC	A/D conversor analógico digital. Entrada de 0~1V, range do valor: 0~1024	11	IO15	GPIO15; MTDO; HSPI_CS; UART0_RTS
2	EN	Pino EM, utiliza Pull UP como padrão	12	TXD	UART0_TXD; GPIO1
3	IO16	GPIO16	13	RXD	UART0_RXD; GPIO3
4	IO14	GPIO14; HSPI_CLK	14	GND	Power ground
5	IO12	GPIO12; HSPI_MISO	15	5V	5V power supply
6	IO13	GPIO13; HSPI_MOSI; UART0_CTS	16	3.3V	3.3V power supply
7	IO5	GPIO5	17	RY1	Pino utilizado para o I/O 16. Para utilizar o rele será necessário adicionar o Jumper
8	IO4	GPIO4	18	RY2	Pino utilizado para o I/O 14. Para utilizar o rele será necessário adicionar o Jumper
9	IO0	GPIO0	19	RY3	Pino utilizado para o I/O 12. Para utilizar o rele será necessário adicionar o Jumper
10	IO2	GPIO2; UART1_TXD	20	RY4	Pino utilizado para o I/O 13. Para utilizar o rele será necessário adicionar o Jumper.

INSTALAÇÃO E UTILIZAÇÃO DO AMBIENTE DE DESENVOLVIMENTO ARDUINO

ESP8266 oferece suporte a ferramentas de desenvolvimento como Eclipse (Nativo e em linguagem C) / Arduino IDE. É relativamente simples usar o Arduino. Aqui está como construir o ambiente de desenvolvimento Arduino:

1. Instale o Arduino IDE 1.8.9 ou a versão mais recente;

<https://www.arduino.cc/en/software>



Faça o download da IDE para seu sistema operacional.

2. Abra o IDE do Arduino, clique em Arquivo-Preferências na barra de menu e clique em Adicionar URL em "URL do gerenciador de placa de desenvolvimento adicional" após inserir as preferências: http://arduino.esp8266.com/stable/package_esp8266com_index.json
3. Clique em Ferramentas-Placa de desenvolvimento-Gerenciador de placa de desenvolvimento na barra de menus e, em seguida, pesquise "ESP8266" para instalar o pacote de suporte do Arduino 2.5.2 ou a versão mais recente do ESP8266

DOWNLOAD DO PROGRAMA PARA A PLACA

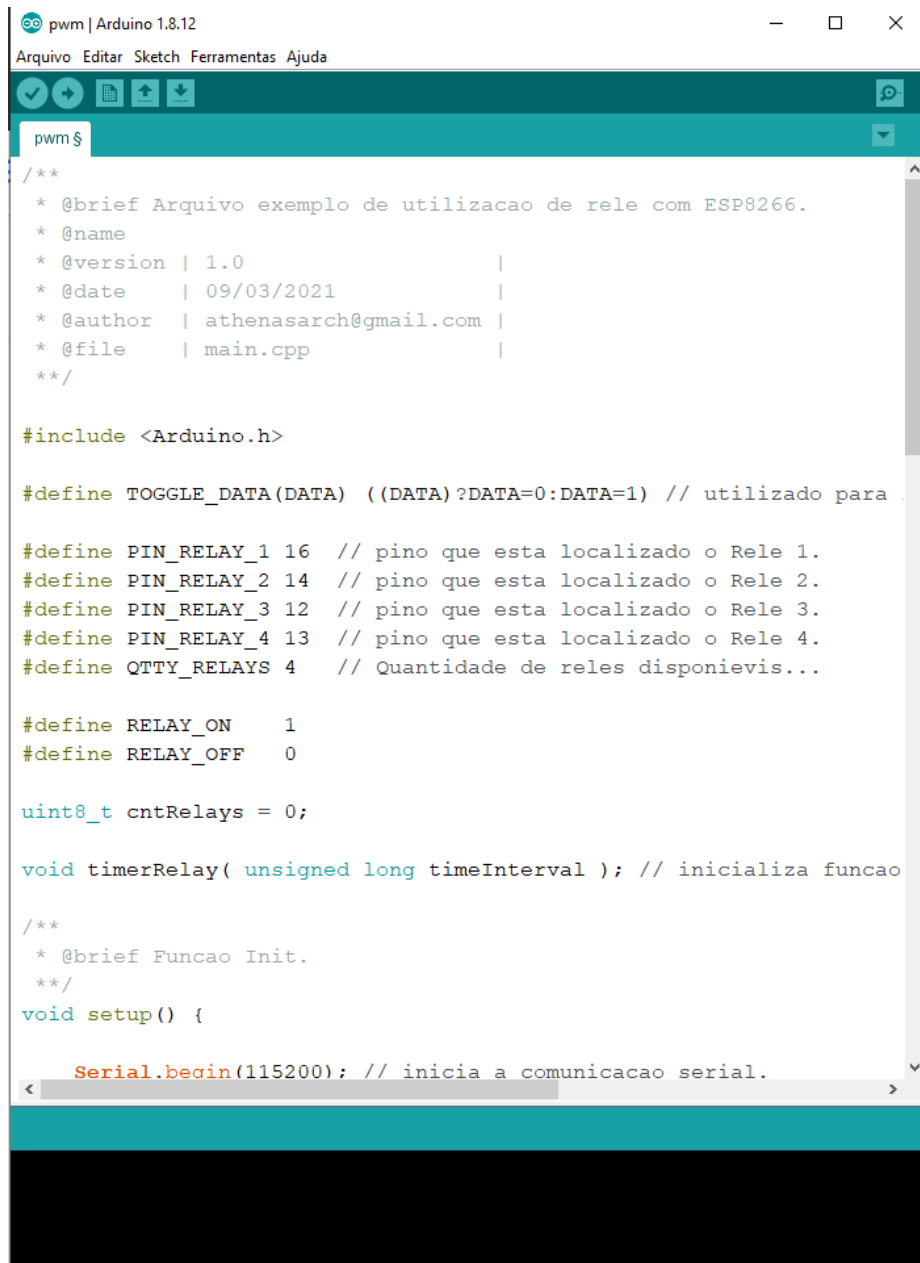
1. use uma conexão de jumper para conectar os pinos **IO0 e GND**, prepare um módulo de porta serial TTL (por exemplo: FT232) para conectar ao USB do computador, o módulo de porta serial e a placa de desenvolvimento são conectados da seguinte forma:

TTL serial port module	ESP8266 development board
GND	GND
TX	RX
RX	TX
5V	5V

2. Clique em “Ferramentas” > “Placa de Desenvolvimento” na barra de menu, selecione a placa de desenvolvimento como “ESPino” (módulo ESP-12).

3. Abra o programa que deseja baixar, clique em “Ferramentas” > “Porta” na barra de menu para selecionar o número da porta correta;

4. Após clicar em "Upload", o programa será compilado automaticamente e baixado para a placa de desenvolvimento, da seguinte forma:



```
pwm | Arduino 1.8.12
Arquivo Editar Sketch Ferramentas Ajuda

pwm $

/**
 * @brief Arquivo exemplo de utilizacao de rele com ESP8266.
 * @name
 * @version | 1.0
 * @date | 09/03/2021
 * @author | athenasarch@gmail.com
 * @file | main.cpp
 */

#include <Arduino.h>

#define TOGGLE_DATA(DATA) ((DATA)?DATA=0:DATA=1) // utilizado para

#define PIN_RELAY_1 16 // pino que esta localizado o Rele 1.
#define PIN_RELAY_2 14 // pino que esta localizado o Rele 2.
#define PIN_RELAY_3 12 // pino que esta localizado o Rele 3.
#define PIN_RELAY_4 13 // pino que esta localizado o Rele 4.
#define QTTY_RELAYS 4 // Quantidade de reles disponievis...

#define RELAY_ON 1
#define RELAY_OFF 0

uint8_t cntRelays = 0;

void timerRelay( unsigned long timeInterval ); // inicializa funcao

/**
 * @brief Funcao Init.
 */
void setup() {

    Serial.begin(115200); // inicia a comunicacao serial.
```

5. Finalmente, desconecte **IO0** e **GND**, ligue a placa de desenvolvimento novamente ou pressione o botão de reinicialização para executar o programa.

```

/**
 * @brief Arquivo exemplo ESP12F_RELAY_X4.
 * @name
 * @version | 1.0 |
 * @date | 09/03/2021 |
 * @author | athenasarch@gmail.com |
 * @file | main.cpp |
 */

#include <Arduino.h>

#define TOGGLE_DATA(DATA) ((DATA)?DATA=0:DATA=1) // utilizado
para inverter estado de variavel.

#define PIN_RELAY_1 16 // pino que esta localizado o Rele 1.
#define PIN_RELAY_2 14 // pino que esta localizado o Rele 2.
#define PIN_RELAY_3 12 // pino que esta localizado o Rele 3.
#define PIN_RELAY_4 13 // pino que esta localizado o Rele 4.
#define QTTY_RELAYS 4 // Quantidade de reles disponievis...

#define RELAY_ON 1
#define RELAY_OFF 0

uint8_t cntRelays = 0;

void timerRelay( unsigned long timeInterval ); // inicializa f
uncao de intervalo do rele.

/**
 * @brief Funcao Init.
 */
void setup() {

    Serial.begin(115200); // inicia a comunicacao serial.
    pinMode(PIN_RELAY_1, OUTPUT); // seta pino do rele como sa
ida.
    pinMode(PIN_RELAY_2, OUTPUT); // seta pino do rele como sa
ida.
    pinMode(PIN_RELAY_3, OUTPUT); // seta pino do rele como sa
ida.
    pinMode(PIN_RELAY_4, OUTPUT); // seta pino do rele como sa
ida.
}

```



```

        Serial.println("\r\r\n\nESP8266 - Serial INIT\r\r\n\n"); /
/ imprime algo na porta serial.
}

/**
 * @brief Funcao Loop infinito.
 */
void loop() {

    timerRelay(1000);
}

/**
 * @brief Ativa e desativa o rele a cada intervalo de tempo.
 *
 * @param
 *
 * @return
 */
void timerRelay( unsigned long timeInterval ){

    static unsigned long timerRelay = 0;
    static uint8_t toggleRelay=0;

    // funcao executada a cada 1000ms
    if(millis() - timerRelay > timeInterval){

        timerRelay = millis();

        // Serial.print("CNT: "); Serial.println( toggleRelay
/ timeInterval); // printa o contador de tempo.

        TOGGLE_DATA(toggleRelay);

        if(cntRelays>4){
            Serial.print("OFF - Desliga todos os Reles.");
            cntRelays=0;
        } else {
            cntRelays++;
            Serial.print("ON - Rele ");
            Serial.print(cntRelays);

```

```
        Serial.println("Ativo.");
    }

    switch(cntRelays){
        case 0:
            digitalWrite(PIN_RELAY_1, RELAY_OFF);
            digitalWrite(PIN_RELAY_2, RELAY_OFF);
            digitalWrite(PIN_RELAY_3, RELAY_OFF);
            digitalWrite(PIN_RELAY_4, RELAY_OFF);
            break;
        case 1:
            digitalWrite(PIN_RELAY_1, RELAY_ON);
            break;
        case 2:
            digitalWrite(PIN_RELAY_2, RELAY_ON);
            break;
        case 3:
            digitalWrite(PIN_RELAY_3, RELAY_ON);
            break;
        case 4:
            digitalWrite(PIN_RELAY_4, RELAY_ON);
            break;
    }
}
}
```