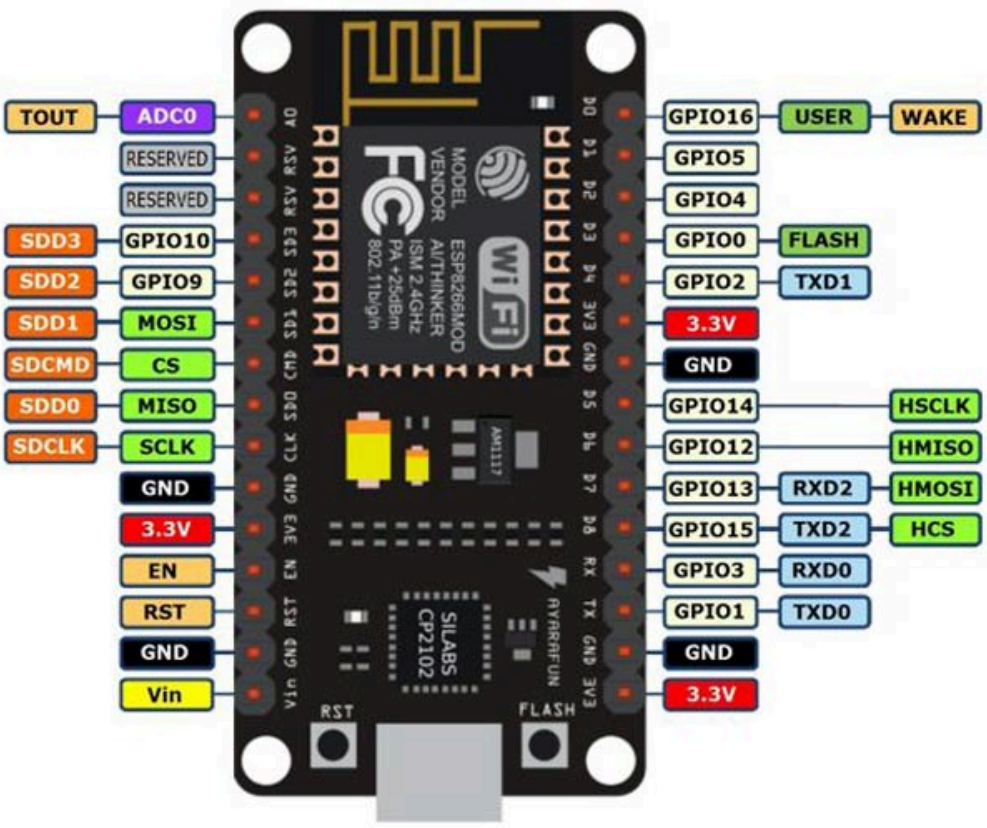
PLANO DE AULA

ESP8266

O **ESP8266** é um chip microcontrolador desenvolvido pela Expressif para facilitar a conectividade de placas e a criação de soluções dentro do ecossistema da Internet das Coisas – IoT. Ele traz incorporado os principais recursos necessários para comunicação Wi-Fi, além de portas GPIOs que permitem a conexão com placas e sensores. O chip possui uma CPU de 32 Bits rodando a 80 MHz e suporta internet nos padrões 802.11 b/g/n, além de vários protocolos de segurança como WEP, WPA, WPA2, etc. A seguir, destacamos os aspectos mais relevantes do microcontrolador, incluindo suas características de pinagem.



- **Pinos de Alimentação:**
  - **3.3V:** É o pino de entrada da fonte de alimentação, que opera com uma tensão de 3.3V. Ele fornece a energia necessária para o funcionamento interno do microcontrolador.
  - **GND:** É o pino de referência de terra (*ground*), essencial para completar o circuito elétrico.
  - **Vin:** Permite a alimentação do módulo com uma tensão de 5V.

- **Pinos de Entrada e Saída de Dados (GPIOs):**
  - O ESP8266 possui diversos pinos de propósito geral (GPIO - General-Purpose Input/Output) que podem ser configurados como entrada ou saída digital.
  - Esses pinos são identificados como **GPIO0**, **GPIO2**, **GPIO4**, **GPIO5**, **GPIO9**, **GPIO10**, **GPIO12**, **GPIO13**, **GPIO14**, **GPIO15** e **GPIO16**.
  - Alguns GPIOs têm funções adicionais, como os pinos **GPIO1** (TXD0) e **GPIO3** (RXD0) para comunicação serial.
  - Pinos como **GPIO16** têm funções especiais, como WAKE, que pode ser usada para “acordar” o microcontrolador de um modo de sono profundo.

- **Pinos de Comunicação:**
  - **SDA e SCL:** Pinos para comunicação I²C, utilizados para interagir com sensores e módulos.
  - **MISO, MOSI e SCLK:** Pinos para comunicação SPI, permitindo a comunicação de alta velocidade com outros dispositivos.

    - **RXD0/TXD0 e RXD2/TXD2:** Pinos de transmissão e recepção de dados seriais, usados para comunicação com computadores ou outros dispositivos.
    - **CS:** Chip Select, usado em comunicação SPI para selecionar o dispositivo.

- **Outros Pinos Importantes:**
    - **ADC0:** Pino de entrada analógica, útil para ler valores de sensores analógicos, como potenciômetros e sensores de luz.
    - **EN:** Chip Enable, usado para habilitar o módulo.
    - **RST:** Pino de reset, usado para reiniciar o microcontrolador.
    - **FLASH:** Pino que controla o modo de inicialização. Quando está em nível baixo, o microcontrolador entra no modo de gravação.

Principais Características do ESP8266

O ESP8266 é um microcontrolador compacto, mas poderoso, com as seguintes características principais:

- **Wi-Fi Integrado:** Permite que o módulo se conecte a redes Wi-Fi, servindo como ponto de acesso ou cliente.
- **Baixo Custo:** É uma solução econômica para projetos de IoT, tornando-o acessível a hobbistas e desenvolvedores.
- **Compatibilidade:** A programação pode ser feita via comandos AT, usando a linguagem LUA ou a IDE do Arduino.
- **Memória:** Possui memória flash para armazenamento de código, RAM para dados temporários e EEPROM para dados permanentes.

- **GPIOs Versáteis:** Oferece uma variedade de pinos para entrada e saída de dados, alguns com funções adicionais como PWM e interrupções.
- **Baixo Consumo de Energia:** Pode operar de forma eficiente em sistemas alimentados por bateria, graças aos seus modos de sono profundo.

Link datasheet:

- https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
- https://handsontec.com/dataspecs/module/esp8266-V13.pdf

EXERCÍCIOS

Loop For

Implemente um programa em C que imprima um vetor de n inteiros mostrando seu índice:

Requisitos:

- Utilize loops aninhados para preencher e exibir o vetor.
- Certifique-se de identificar corretamente como um vetor é inicializado.

Código:

```
#include <stdio.h>

int main() {
    int n;

    printf("Digite o tamanho do vetor:\n ");
    scanf("%d", n);

    int vetor[n];

    printf("Preencha o vetor:\n");

    for(int i = 1; i < n; i++) {
        printf("Digite o valor para a posição %d: ", i);
        scanf("%d", &vetor[i]);
    }

    printf("Vetor preenchido:\n");

    for(int i = 1; i < n; i++) {
        printf("vetor[%d] = %d\n", i, vetor[i]);


    return 0;
}
```

Contagem em C

Crie um programa em C que tenha uma entrada de usuário (s / n) e faça uma contagem regressiva utilizando um #define para o número máximo de iterações.

Requisitos:

- Use um #define para implementar um número máximo de iterações.
- Utilize de um loop While para fazer a contagem.
- O usuário deverá decidir se vai haver contagem, ou não.

Código:

```
#include <stdio.h>


#define TEMPO_INICIAL 10


int main{
    char resposta;
    int tempo = TEMPO_INICIAL


    printf("Deseja iniciar a contagem regressiva? (s/n): ");
    scanf(" %c", resposta);


    if (resposta == 's' || resposta == 'S') {
        while (tempo >= 0) {
            printf("Contagem: %d", tempo);
```

```
            tempo--;
        }
        printf("Finalizado!");
    } else {
        printf("Contagem cancelada.\n");
    }
    return 0;
}
```

Verificar palíndromo

Escreva um programa em C que verifica se uma palavra (fornecida pelo usuário) é um palíndromo. Um palíndromo é uma palavra que permanece a mesma quando lida de trás para frente (ex.: "arara", "radar").

Requisitos:

- Use um array de caracteres (char[]) para armazenar a palavra.
- Implemente uma função bool ehPalindromo(char palavra[]) que retorna true se a palavra for um palíndromo e false caso contrário.
- Ignore diferenças entre maiúsculas e minúsculas.

Código:

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
```

```
#include <ctype.h>

bool ehPalindromo(char palavra[]) {
    int inicio = 0;
    int fim = strlen(palavra) - 1;

    while (inicio < fim) {
        if (tolower(palavra[inicio]) != tolower(palavra[fim])) {
            return false;
        }
        inicio++;
        fim--;
    }
    return true;
}

int main() {
    char palavra[100];
    printf("Digite uma palavra: ");
    scanf("%s", palavra);

    if (ehPalindromo(palavra)) {
        printf("\"%s\" é um palíndromo!\n", palavra);
```

```
    } else {
        printf("\"%s\" não é um palíndromo!\n", palavra);
    }

    return 0;
}
```

Soma dos números pares de um vetor

Escreva um programa em C que:

- Solicita ao usuário o tamanho de um array (n).
- Lê n números inteiros e os armazena em um array.
- Calcula e exibe a soma de todos os números pares no array.

Requisitos:

- Implemente uma função int somaPares(int array[], int tamanho) que retorna a soma dos números pares.

Código:

```c
#include <stdio.h>

int somaPares(int array[], int tamanho) {
    int soma = 0;
    for (int i = 0; i < tamanho; i++) {
        if (array[i] % 2 == 0) {
            soma += array[i];
        }
    }
    return soma;
}

int main() {
    int n;
    printf("Digite o tamanho do array (máx 100): ");
    scanf("%d", &n);

    int array[100]; // Array estático com tamanho máximo 100

    printf("Digite os elementos do array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &array[i]);
    }
```

```
    int soma = somaPares(array, n);

    printf("A soma dos números pares é: %d\n", soma);


    return 0;

}
```

Giro suavizado com Servo Motor

Crie um código em .ino para controlar um servo motor que se mova suavemente entre 5° e 175°, fazendo um movimento de vai-e-volta (ida e retorno). O programa deve utilizar a biblioteca Servo.h e exibir no Monitor Serial a posição atual do servo a cada passo.

Requisitos:

- Incluir a biblioteca `Servo.h`
- Definir o pino de controle do servo com `#define`
- Criar uma variável `pos` para controlar a posição do servo
- Imprimir no Serial a posição atual usando `Serial.println()`
- Inserir `delay(20)` entre os passos para suavizar o movimento
- Adicionar `delay(1000)` após ida e após volta para criar pausa entre ciclos

código:

```
#include <Servo.h>

Servo meuServo;
#define SERVO_PIN 2
int pos = 0;
```

```
void setup() {
  Serial.begin(115200);
  meuServo.attach(SERVO_PIN, 500, 2400);
  Serial.println("Iniciando movimento do servo...");
}
void loop() {
  for (pos = 5; pos <= 175) {
    meuServo.write(pos);
    Serial.print("Posição atual: ");
    Serial.println(pos);
    delay(20);
  }
  delay(1000);
  for (pos = 175; pos >= 5) {
    meuServo.write(pos);
    Serial.print("Posição atual: ");
    Serial.println(pos);
    delay(20);
  }
  delay(1000);
}
```

Giro suavizado de Servo Motor Com Entrada.

Crie um código em `.ino` (Arduino) para controlar um servo motor que se mova de 0 até 180 graus de forma suavizada, utilizando a biblioteca `Servo.h`. A cada passo do movimento, o ângulo atual deve ser impresso no Monitor Serial.

Requisitos:

- Incluir a biblioteca `Servo.h`

- Definir um pino digital para controle PWM do servo
- Associar o pino ao servo com o método `attach()`
- Fazer o servo se mover de 0 até 180 graus em passos de 1 grau
- Inserir um pequeno `delay()` entre os passos para suavizar o movimento
- Imprimir o ângulo atual no Monitor Serial com `Serial.println()`
- Configurar `Serial.begin(115200)` no `setup()`

Código:

```
#include <Servo.h>


#define SERVO_PIN 2


Servo meuServo;
String inputString = "";


void setup() {
 Serial.begin(115200);
 meuServo.attach(SERVO_PIN, 1000, 2000);
 Serial.println("Digite o ângulo (0 a 180):");
}


void loop() {
 while (Serial.available()) {
   char c = Serial.read();
   if (c == '\n' || c == '\r') {
     if (inputString.length() > 0) {
       int angulo = inputString.toInt();
       if (angulo < 0) angulo = 0;
       if (angulo > 180) angulo = 180;


       Serial.print("Movendo para o ângulo: ");
       Serial.println(angulo);
```

```
      meuServo.write(angulo);
      inputString = "";
    }
  } else {
    inputString += c;
  }
 }
}
```

Primeiros Passos - ESP8266

O NodeMCU deve primeiramente ser conectado ao computador e deve ser feita a verificação se o sistema operacional identificou-o e atribuiu uma porta COM. Caso contrário será necessário instalar o driver do chip que estiver utilizando (CP2102 ou CH340).

Após a instalação do driver, é necessário instalar a biblioteca do ESP8266 na IDE Arduino, que por padrão não a possui. O link http://arduino.esp8266.com/stable/package_esp8266com_index.json, deve ser adicionado ao campo de “Additional boards manager URLs” por meio da guia “File” na opção “Preferences”.

Testando ESP8266

O “blink” é um programa simples com a finalidade de ligar e desligar, alternadamente, o LED que vem por padrão na placa. Este programa, mesmo possuindo a sua simplicidade possui uma papel importante quanto a comunicação com o microcontrolador, de forma a verificar se o dispositivo está realmente funcionando.

Código para o “blink”

```
#define LED 2
void setup() {
    pinMode(LED, OUTPUT);
}

void loop() {
    digitalWrite(LED, HIGH);
```

```
    delay(1000);
    digitalWrite(LED, LOW);
    delay(1000);
}
```

Circuito com LED

Garantindo a comunicação e funcionamento do ESP por meio do programa “blink”, o próximo passo será introduzir os circuitos desejados a sua aplicação. Neste exemplo faremos a configuração de um LED externo conectado à placa, para que acenda juntamente com LED interno do ESP8266. Para tal, será necessário uma Protoboard para a manipulação do circuito, juntamente com jumper e um resistor.