****

**UFPI - CCN - DC**

**CIÊNCIA DA COMPUTAÇÃO**

**SISTEMAS EMBARCADOS**

**Prof. Dr. Antonio Helson Mineiro Soares**

Relatório Técnico

**Tutorial de uso: NodeMCU ESP8266 ESP-12**

CLÁUDIO MÁRCIO DE OLIVEIRA LEAL FILHO

LUCAS HERLON DOS SANTOS MOREIRA CUNHA

**Teresina**

**2024**

1. **Introdução**

O presente trabalho visa detalhar as características e explicar o funcionamento da placa NodeMCU ESP-12, que contém o microcontrolador ESP-8266 da marca Espressif. Esta placa tem amplo uso em IoT (internet das coisas) e eletrônica embarcada, podendo ser utilizada para projetos de automação, monitoramento e robótica. Na Tabela 1, estão listados os itens que compõem o hardware da placa e suas respectivas descrições. Nela podemos ver que a placa possui conexão wi-fi, conexão micro-usb, entradas digitais e uma entrada analógica.

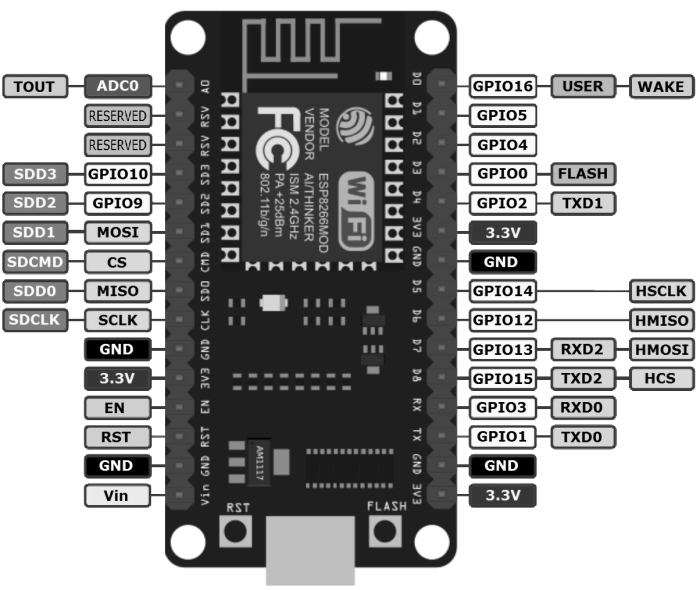
**Tabela 1 - Descrição do Hardware**

| **Conversor Serial** | **CI** | CP2102 |
| --- | --- | --- |
| **Conexão** | Micro-USB |
| **Regulador de Tensão** | **Tensão de Entrada** | 3,3V AMS1117 |
| **Conector Externo** | 5,0V |
| **Microcontrolador** | **Nome** | ESP12 |
| **Arquitetura** | Tensilica Xtensa LX106 |
| **Core** | Dual-core de 32 bits |
| **Clock** | 240Mhz |
| **Tensão de Operação** | 3,3 VDC |
| **Memória Flash** | 4 MB |
| **Memória SRAM** | 520 KB |
| **Memória EEPROM** | 448 KB |
| **Qt Portas Digitais** | 11 |
| **Qt Portas Analógicas** | 1 |
| **Qt Portas PWM** | 11 |
| **Qt Portas Interrupção** | 10 |
| **Led Builtin** | D4/GPIO2 |
| **Interfaces de Comunicação** | UART/SPI/I2C |
| **DAC/ADC** | 1 |
| **Wireless** | 16 |

1. **Descrição da Placa**

A placa ESP12 possuiu 11 entradas digitais, que vão da GPIO0 à GPIO16 (nem todos os pinos estão disponíveis) e uma entrada analógica (A0). É importante notar que o número da GPIO não corresponde necessariamente com o número exibido na placa (silk da placa) como mostrado na Figura 01, podemos ver por exemplo que a GPIO0 corresponde ao silk d3. Além disso, a placa apresenta entradas de alimentação como 3v3, GND e EN. Na Tabela 2 vemos a descrição mais detalhada do pinout do ESP12.

**Figura 01 - Placa Pinout**

****

**Tabela 2 - Descrição das GPIO (Silk da placa)**

| **Funcionalidades** | **GPIO** |
| --- | --- |
| PWM | 1-16 |
| DAC/ADC | ADC0 |
| Interrupção | RST |
| UART | 1, 3 |
| I2C | 4, 5 |
| SPI | 12-15 |

1. **Como programar**

Foi utilizada a Arduino IDE. Configuramos ela com os seguintes links:

| http://arduino.esp8266.com/stable/package\_esp8266com\_index.json  https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\_esp32\_index.json |
| --- |

Após isso, colocamos a Board como “Generic 8266 Module”, indo em Tools -> Boards Manager e selecionando ela.

A linguagem utilizada nos códigos do presente trabalho foi o C++.

**3.1 - Exemplo de blink de led on board**

O “Blink” é o equivalente do “Hello World” dos Sistemas Embarcados, e corresponde ao fato de fazer com que um LED, seja onboard ou separado, piscar. Nesse exemplo, realizamos o Blink no ESP12.

**Código 1 - Exemplo de blink**

| 1. void setup() { 2. //código do setup 3. pinMode(LED\_BUILTIN, OUTPUT); 4. } 5. void loop() { 6. //código do loop 7. digitalWrite(LED\_BUILTIN, HIGH); //faz o LED\_BUILTIN receber 5v 8. delay(1000); // sleep 9. digitalWrite(LED\_BUILTIN, LOW); //faz o LED\_BUILTIN receber 0v 10. delay(1000); // sleep 11. } |
| --- |

**3.2 - Exemplo de blink de led on board sem Delay**

Dessa vez, utilizaremos um código um pouco diferente, que simulará o uso do comando delay() através de um mecanismo que mede o tempo no relógio do ESP12, com uma grande diferença, pois o ESP12 não ficará imune a sensibilidade do ambiente enquanto ele encontra-se neste modo.

De acordo com o que foi dito em sala, temos dois relógios diferentes, um para o LED vermelho e um para o LED amarelo. Eles alternam o seu estado atual cada vez que seu timer chega.

Na implementação técnica realizada em sala de aula, o Led amarelo corresponde à entrada D3 (e por isso possui valor GPIO0, segundo o Pinout) e o Led vermelho corresponde à entrada D2 (valor GPIO4).

| 1. #define LEDV 4 2. #define LEDA 0 3. //inicializei as variáveis aqui pra que elas possam ser acessadas nas funções, 4. //entretanto elas possuem valor nulo 5. //no C++, valor nulo é acessível? (isto é, o valor começa como 0, no caso do Int) 6. //ou é considerado nulo e dá erro de null pointer? 7. //teste feito: nenhum dos dois. a variável possui valor não determinístico. 8. int piscaA; //variável responsável por dizer se atualmente, no ciclo de blink, 9. //está em HIGH ou LOW 10. int piscaV; 11. int tempoAnteriorV;//tempo da ultima mudança de borda 12. int tempoAnteriorA; 13. int duracaoBlinkA; 14. int duracaoBlinkV; 15. void setup() { 16. pinMode(LEDA, OUTPUT); 17. pinMode(LEDV, OUTPUT); 18. piscaV = LOW;//começa-se apagado 19. piscaA = LOW; 20. tempoAnteriorA = 0; 21. tempoAnteriorV = 0; 22. duracaoBlinkA = 500; 23. duracaoBlinkV = 800; 24. } 25. void loop() { 26. int tempoAtual = millis(); 27. if(tempoAtual - tempoAnteriorA >= duracaoBlinkA){ 28. //se o tempo entre a última mudança de borda for maior que a duração 29. //do blink, é hora de mudar novamente 30. tempoAnteriorA = tempoAtual; 31. revert(LEDA); 32. } 33. if(tempoAtual - tempoAnteriorV >= duracaoBlinkV){ 34. //se o tempo entre a última mudança de borda for maior que a duração 35. //do blink, é hora de mudar novamente 36. tempoAnteriorV = tempoAtual; 37. revert(LEDV); 38. } 39. } 40. void revert(int WTR){//essa função faz um digitalWrite de valor contrário 41. //aquele atualmente presente naquele pino. 42. if(WTR == 4){//vermelho 43. //WTR -> which to revert, nesse caso o 4 44. if(piscaV == LOW){ 45. piscaV = HIGH; 46. }else{ 47. piscaV = LOW; 48. } 49. digitalWrite(WTR, piscaV); 50. } 51. else{ 52. if(piscaA == LOW){ 53. piscaA = HIGH; 54. }else{ 55. piscaA = LOW; 56. } 57. digitalWrite(WTR, piscaA); 58. } 59. } |
| --- |

**4. Entrada/saída**

Nesta seção mostramos como ocorrem as operações de entrada e saída no ESP12 considerando que estas podem ocorrer de modo digital, analógico e serial. Aqui iremos acender um LED recebendo o comando de um botão pela entrada digital, acenderemos outro LED lendo o sinal de um potenciômetro pela entrada analógica e convertendo-a em sinal digital e utilizaremos também o monitor serial para acender o segundo LED.

**4.1 Digital (sem delay com debounce)**

Para acender o LED1 sem delay com debounce, implementamos um sistema que apenas altera o estado do LED1 se uma certa quantia de tempo tiver passado com o botão sendo pressionado. Isso faz com que o bounce seja evitado.

**4.2 Analogica/PWM**

A entrada analógica do ESP12 se encontra na porta A0. Chamamos ela no código de potenciometroPIN, e ligamos ela com o pino central do potenciômetro. Aos pinos laterais do potenciômetro ligamos o ground do ESP12 e o pino 3.3v. Fazemos então a leitura da entrada do potenciômetro por meio da função analogRead() e, uma vez de posse desse valor, passamos ele para o pino do LED2 (conectado a um pino PWM) por meio da função analogWrite().

**4.3 Serial**

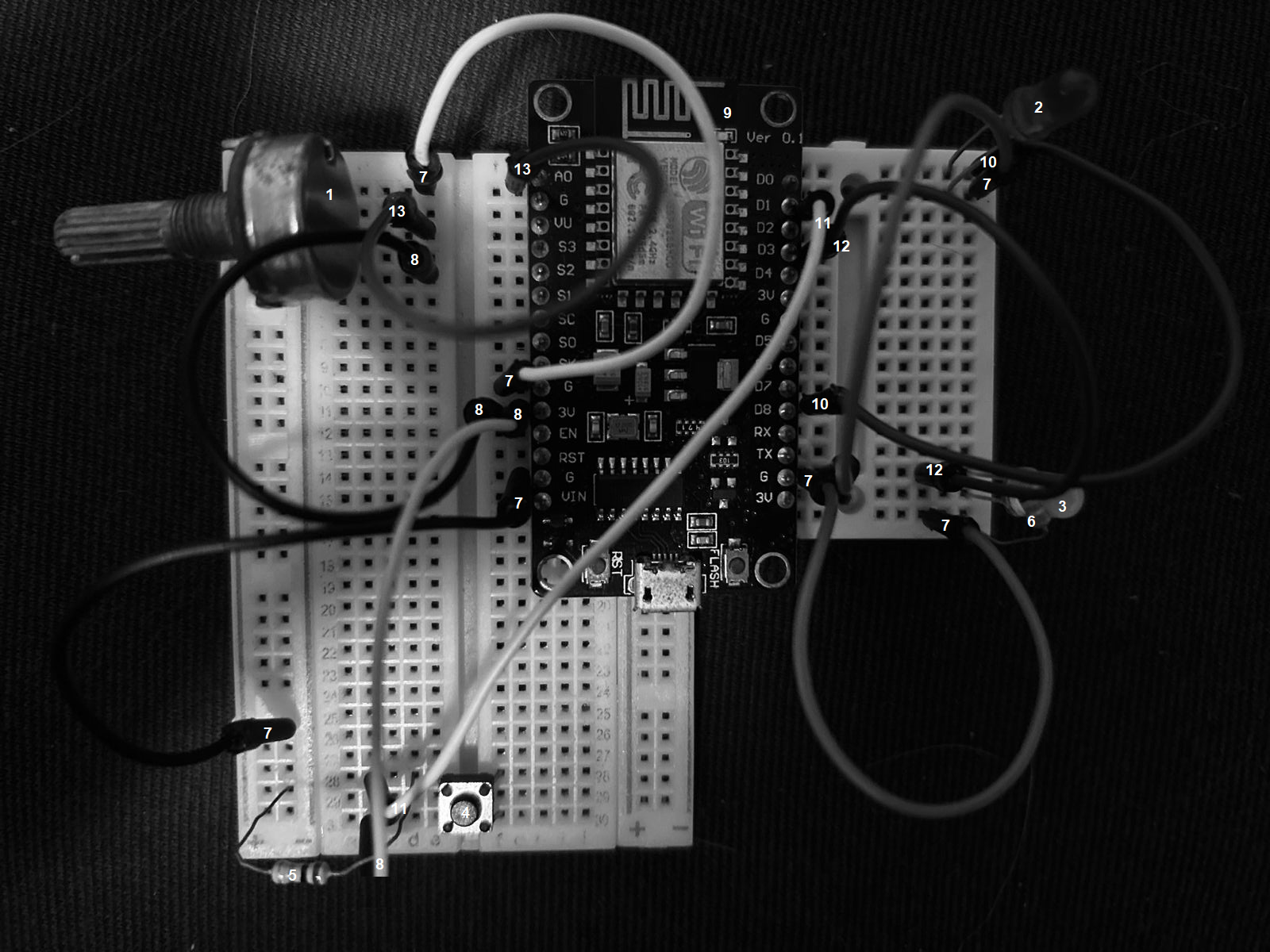
O programa se comunica numa frequência de 9600 baud (data rate in bits per second), configurada na função setup(). Isso faz com que possamos utilizar entrada e saída. Caso a placa receba “ON” ou “OFF” no Serial Monitor, ela irá colocar esse valor no pino LED1. Caso o pino LED1 esteja no modo ON (isto é, o último comando digitado foi ON e não houve nenhum comando OFF após isso), apertar o botão não desligará o pino. Além disso, caso a placa receba um número em vez dessas duas palavras, ela irá passar esse valor para o LED2.

**4.4 Exemplo de entradas e saídas**

Nesse exemplo, executamos tudo o que foi pedido nas outras seções ao mesmo tempo.

**Código 2 - Exemplo da seção 4**

1. #define LEDOB 2 //onboard, porta D4
2. #define LED1 15 //VERMELHO, porta D8
3. #define LED2 0 //AMARELO, porta D3
4. #define BOTAO 5
5. #define duracaoBlinkV 800
6. #define delayDebounce 50
7. #define potenciometroPIN A0
8. #define debug false
9. const int TAMANHO\_FILTRO = 10; // Tamanho do buffer para a média móvel
10. int bufferFiltro[TAMANHO\_FILTRO] = {0}; // Buffer circular para as leituras
11. int indiceFiltro = 0; // Índice atual no buffer
12. int somaFiltro = 0; // Soma acumulada do buffer
13. int ultimoEstadoBOTAO = LOW;
14. int tempoDsdUltimoEstado = 0;
15. int estadoBotao;
16. int LedBotao = HIGH;
17. int tempoAnteriorV = 0;
18. int mudanca = LOW; // flag que indica que ocorreu mudança significativa no estado do potenciômetro
19. int menuativado = true;
20. // Novas variáveis para controle do LED
21. int valorSerialLED = -1; //armazena o último valor recebido via serial (-1 indica que nenhum valor foi recebido)
22. int ultimoValorPot = 0; //armazena o último valor do potenciômetro
23. const int LIMIAR\_MUDANCA = 20; //diferença mínima no potenciômetro para sobrescrever o valor serial
24. void setup() {
25. Serial.begin(9600);
26. pinMode(LED1, OUTPUT);
27. pinMode(LEDV, OUTPUT);
28. pinMode(LED2, OUTPUT);
29. pinMode(BOTAO, OUTPUT);
30. digitalWrite(LED1, HIGH);
31. }
32. void loop() {
33. // BOTÃO ACENDE O LED AMARELO
34. int leitura = digitalRead(BOTAO);
35. if (leitura != ultimoEstadoBOTAO) {
36. tempoDsdUltimoEstado = millis();
37. }
38. if ((millis() - tempoDsdUltimoEstado) > delayDebounce) {
39. if (leitura != estadoBotao) {
40. estadoBotao = leitura;
41. }
42. if (estadoBotao == HIGH) {
43. LedBotao = !LedBotao;
44. }
45. }
46. digitalWrite(BOTAO, LedBotao);
47. ultimoEstadoBOTAO = leitura;
48. // BLINK ONBOARD
49. int tempoAtual = millis();
50. if (tempoAtual - tempoAnteriorV >= duracaoBlinkV) {
51. tempoAnteriorV = tempoAtual;
52. revert(LEDV);
53. mudanca = HIGH;
54. }
55. //LEITURA DO POTENCIÔMETRO COM FILTRO DE MÉDIA MÓVEL
56. int leituraPot = analogRead(potenciometroPIN);
57. //atualiza o buffer do filtro de média móvel
58. somaFiltro -= bufferFiltro[indiceFiltro]; //subtrai o valor mais antigo
59. bufferFiltro[indiceFiltro] = leituraPot; //adiciona a nova leitura
60. somaFiltro += leituraPot; //atualiza a soma
61. indiceFiltro = (indiceFiltro + 1) % TAMANHO\_FILTRO; //avança no buffer circular
62. // Calcula o valor médio filtrado
63. int valorFiltrado = somaFiltro / TAMANHO\_FILTRO;
64. int valorPWM = map(valorFiltrado, 0, 1023, 0, 255);
65. // Verifica se houve mudança significativa no potenciômetro
66. if (abs(valorPWM - ultimoValorPot) > LIMIAR\_MUDANCA) {
67. valorSerialLED = -1; // Reset do valor serial
68. analogWrite(LED2, valorPWM);
69. mudanca = HIGH;
70. ultimoValorPot = valorPWM;
71. }
72. int input = -1;
73. int number = -1;
74. // LOOP DE ENTRADA NÚMERO
75. if (Serial.available() > 0) {
76. String leitura = Serial.readStringUntil('\n');
77. if (leitura == "ON") {
78. input = 0;
79. } else if (leitura == "OFF") {
80. input = 1;
81. } else if(leitura == "MENUON"){
82. input = 2;
83. } else if(leitura == "MENUOFF"){
84. input = 3;
85. } else if (leitura.toInt() >= 0 && leitura.toInt() <= 255) {
86. input = 4;
87. number = leitura.toInt();
88. } else {
89. input = -2;
90. }
91. }
92. switch(input){
93. case 0:
94. if(debug) Serial.println("Usuário escreveu ON.");
95. digitalWrite(LED1, HIGH);
96. if(debug) Serial.println("LED1 ajustado para HIGH.");
97. break;
98. case 1:
99. if(debug) Serial.println("Usuário escreveu OFF.");
100. digitalWrite(LED1, LOW);
101. if (debug) Serial.println("LED1 ajustado para LOW.");
102. break;
103. case 2:
104. menuativado = true;
105. break;
106. case 3:
107. menuativado = false;
108. break;
110. case 4:
111. if (debug) Serial.print("Serial: ");
112. valorSerialLED = number; // Armazena o valor recebido via serial
113. analogWrite(LED2, valorSerialLED);
114. if (debug) Serial.println("LED2 ajustado para " + String(valorSerialLED));
115. break;
116. case -1:
117. break;
118. default:
119. Serial.println("Comando inválido!");
120. break;
121. }
122. // Se tiver um valor serial definido e não houver mudança significativa no potenciômetro,
123. // mantém o LED no valor definido via serial
124. if (valorSerialLED != -1) {
125. analogWrite(LED2, valorSerialLED);
126. }
127. if(menuativado) menu();
128. delay(42);
129. }
130. void revert(int WTR) {
131. int estadoatual = digitalRead(WTR);
132. if (estadoatual == HIGH) {
133. digitalWrite(WTR, LOW);
134. } else {
135. digitalWrite(WTR, HIGH);
136. }
137. }
138. void menu() {
139. if (mudanca) {
140. for(int i = 0; i < 5; i++){
141. Serial.println("\n\n\n\n");
142. }
143. Serial.println("Sinal do LED on-board: " + String(digitalRead(LEDV)));
144. Serial.println("Sinal do LED1: " + String(digitalRead(LED1)));
145. Serial.println("Sinal do LED2: " + String(analogRead(LED2)));
146. Serial.println("Status do botão: " + String(digitalRead(digitalRead(BOTAO))));
147. debug ? Serial.println("Informações debug ativadas") : Serial.println("Informações debug desativadas");
148. }
149. mudanca = LOW;
150. }

****

1. **Potenciômetro, ligado à A0 por (13), a Ground por (7) e a 3V por (8).**
2. **LED1 (vermelho)**
3. **LED2 (amarelo)**
4. **Botão**
5. **Resistor entre Botão e Ground.**
6. **Resistor entre LED e Ground.**
7. **Múltiplos fios Ground.**
8. **Múltiplos fios 3V.**
9. **LEDOB (on-board), correspondente à GPIO2.**
10. **Cabo unindo LED1 (2) à porta D8, que possui valor GPIO15.**
11. **Cabo unindo o botão (4) à porta D1, que possui valor GPIO5.**
12. **Cabo unindo a porta D3, com valor GPIO0, ao resistor (6) que está ligado ao botão LED2 (3).**
13. **Cabo que une A0 ao potenciômetro (1).**
14. **Comunicação Serial**

**Tipos de comunicação serial: CAN, RS232, …**

**5.1 UART**

<código fonte comentado e numerado>

**5.2 SPI**

<código fonte comentado e numerado>

**5.3 I2C**

<código fonte comentado e numerado>

1. **Wireless**

**Nativo ou por módulo, descrever um ou outro**

**6.1 Bluetooth<com foto da placa>**

<código fonte comentado e numerado>

**6.2 Wi-fi Local web service <com foto da placa>**

<código fonte comentado e numerado>

**6.3 Wi-fi Global usando Blynk<prints>**

<código fonte comentado e numerado>

1. **Considerações finais**

Baseado nos exemplos a placa tem seu uso mais voltado para: uso geral; uso em IoT; num serve pra nada? ….

1. **Referências bibliográficas**