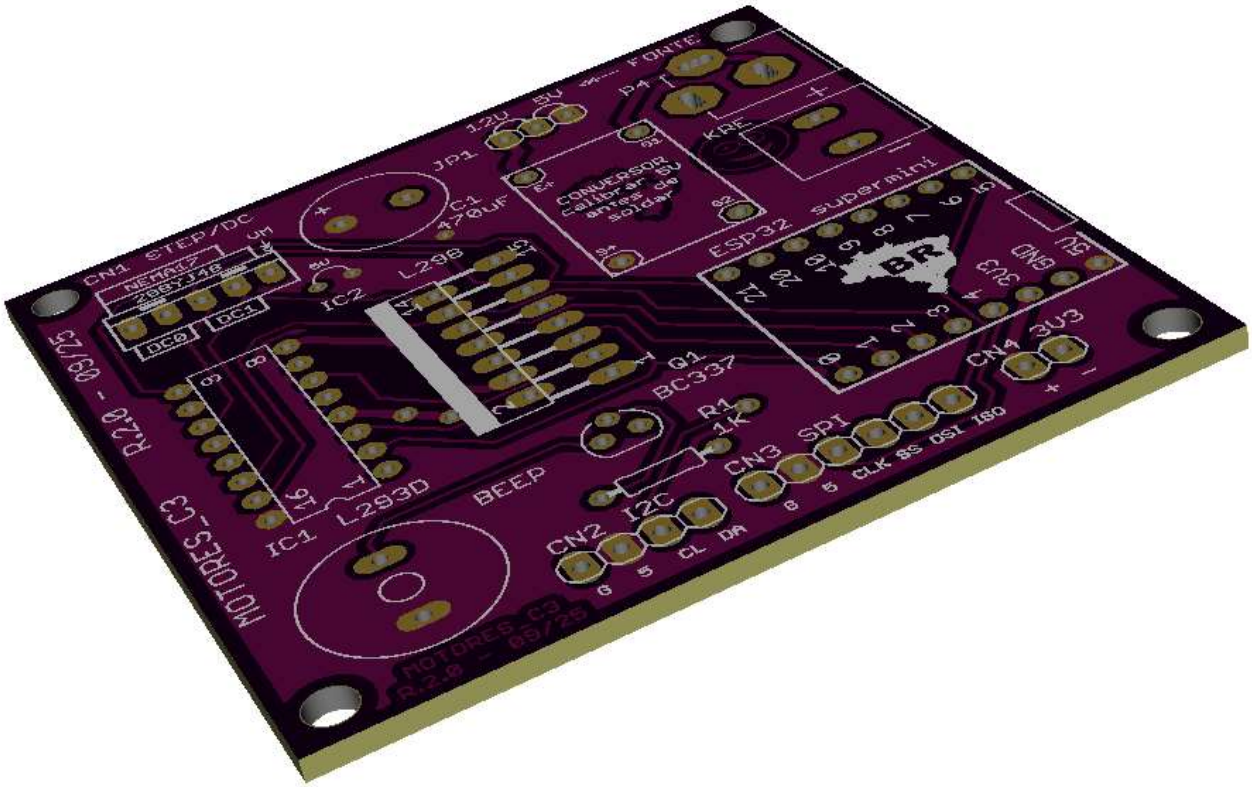
PCI
MOTORES_C3



MANUAL DO USUÁRIO

Aplicável.as placas:

MOTORES_C3 versão 2.0 e posteriores
com o módulo microcontrolador ESP32 C3 supermini de 16 pinos

Manual rev. 1.0 ─ 09/2025
Eventuais atualizações desse manual podem ser encontradas em: https://github.com/izyino/manuais

INTRODUÇÃO:

A presente placa de circuito impresso intitulada MOTORES_C3 foi concebida para facilitar a construção de dispositivos para controle de motores de passo ou até dois motores DC, via WiFi, contemplando em um reduzido espaço de apenas 6 x 6 cm, o moderno microcontrolador ESP32 C3 em sua versão supermini de 16 pinos, os drivers do motor de passo e dos motores DC, e todos os demais componentes, substituindo assim os antigos, obsoletos e saudosos Arduinos e seus “shields” sobrepostos em forma de sanduíche.

A versão 2.0 da placa de circuito impresso MOTORES_C3 possui as seguintes características básicas:

- Utilização do microcontrolador ESP-32 C3 na versão supermini de 16 pinos, com WiFi e Bluetooth, CPU de 32-bits RISC-V Single-core 160MHz, WiFi: 802.11b/g/n 2.4GhHz, Bluetooth 5.0, Consumo ultra baixo de apenas 43uA, 400KB SRAM, 384KB ROM, 4Mb flash

- Controle de dispositivos com base em motor de passo modelos Nema-17 ou 28BYJ48 ou até dois motores DC de baixa ou média corrente, de forma bidirecional, com controle de velocidade PWM

- Utilização do driver ponte H dupla, L293D, dip 16 ou L298

- Beep sonoro passivo

- Conectores para acesso aos sinais I2C, SPI e 3V3

- Completa biblioteca para controle dos motores de passo, dos motores DC, do beep sonoro e do Led azul, bem como um controle de tempo, não blocante, em alternativa às funções millis() e delay(), disponível em https://github.com/izyino/motbepled.h

CONECTORES EXISTENTES NA PLACA MOTORES_C3:

CN1 – motor STEP ou motores DC:

1 – motor DC 0 ou Step L1 (fio azul, se 28BYJ48)
2 – motor DC 0 ou Step L1 (fio rosa, se 28BYJ48)
3 – motor DC 1 ou Step L2 (fio amarelo, se 28BYJ48)
4 – motor DC 1 ou Step L2 (fio laranja, se 28BYJ48)
5 – sem conexão - Step comum (fio vermelho, se 28BYJ48)

CN2 – I2C:

1 – Gnd
2 – Vcc (+5V)
3 – SCL
4 – SDA

CN3 – SPI:

1 – Gnd
2 – Vcc (+5V)
3 – CLK
4 – SS
5 – MOSI
6 – MISO

CN4 – 3V3:

1 – Gnd
2 – +3V3

RELAÇÃO DE COMPONENTES:

Placa de circuito impresso Rev.1.1
Módulo ESP32 C3 supermini, 16 pinos
CI driver L293D, dip 16 ou L298 (ou um ou outro, nunca os dois)
Dissipador de calor para o driver L298 (se necessário)
Conector molex 5 pinos com polarizador (para os motores em CN1)
Conector P4 fêmea, 90 graus, solda placa
Conector 3 pinos macho (JP1 – seleção fonte 12V ou 5V) e um jumper plástico
Módulo conversor 12V–>5V MP2307 (montar se a fonte de alimentação for de 12V)
Borne KRE de 2 pinos (montagem opcional)
Beep passivo (montagem opcional)
Transistor BC337 (montagem opcional)
Resistor de 1K (montagem opcional)
Conector KRE de 2 pinos (montagem opcional)
Conector 4 pinos macho (CN2 – I2C, montagem opcional)
Conector 6 pinos macho (CN3 – SPI, montagem opcional)
Conector 2 pinos macho (CN4 – 3V3, montagem opcional)

POSSIVEIS CONFIGURAÇÕES:

1) Para aplicações que utilizem até dois motores DC de baixa potência ou um motor de passo modelo 28BYJ48 ou similar, deve-se montar a PCI com o driver L293D apenas (o L298 e conversor não montados), com um jumper entre o pino central e o pino mais a direita (5V) de JP1. Nessa configuração deve-se utilizar uma fonte de 5Vcc em P4 ou no borne KRE. Observação: esta aplicação poderia também utilizar o driver L298, que apesar de ser mais caro que o L293, o substitui perfeitamente

2) Para aplicações que utilizem até dois motores DC de média potência ou um motor de passo modelo Nema17 ou similar, deve-se montar a PCI apenas com o driver L298 e o módulo conversor, com um jumper entre o pino central e o pino mais a esquerda (12V) de JP1. Nessa configuração deve-se utilizar uma fonte de 12Vcc em P4 ou no borne KRE.

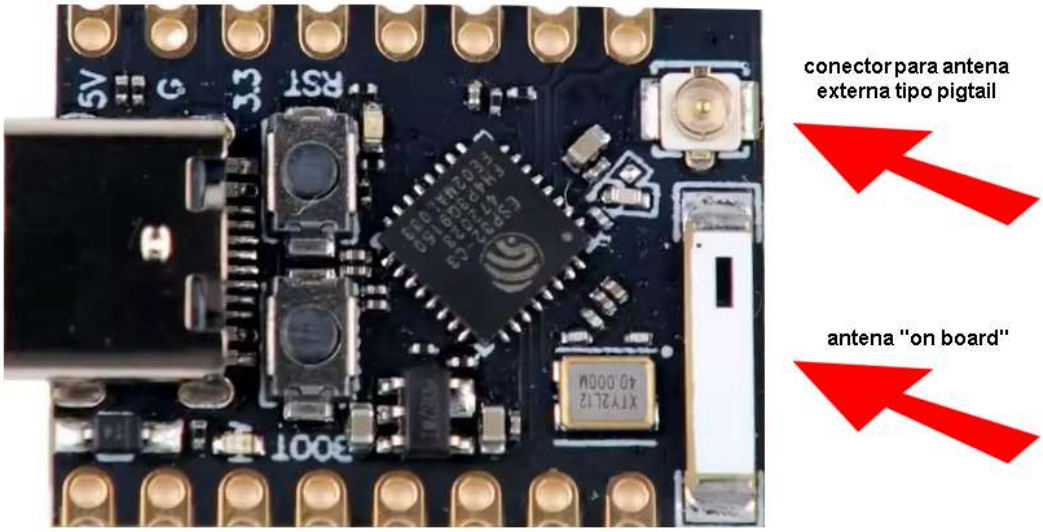


Fig. 1 – Modelo de ESP32 C3 recomendado

Existe diversos modelos diferentes de ESP32 C3 disponíveis no mercado. O modelo que mostrou mais facilidade de conexão e melhor desempenho, especialmente no tocante ao alcance WiFi é o mostrado na figura 1 acima.

O modelo mais comum, mostrado na figura 2 abaixo, **deve ser evitado** pois apresenta sérios problemas na conexão com redes WiFi além de possuir um alcance muito pequeno. Sua antena “on board” é visivelmente menos elaborada, sem contar com a ausência do conector para antena externa.

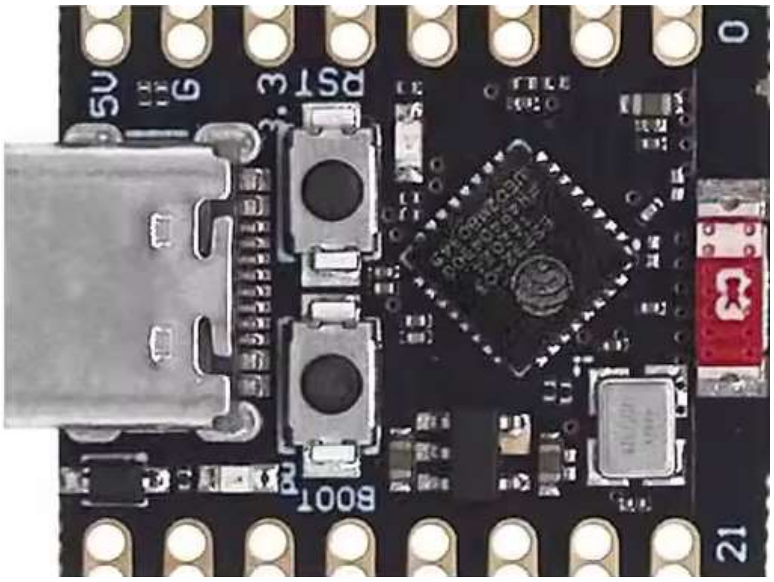


Fig. 2 – Modelo de ESP32 C3 **-NÃO-** recomendado

DIAGRAMA ELÉTRICO:

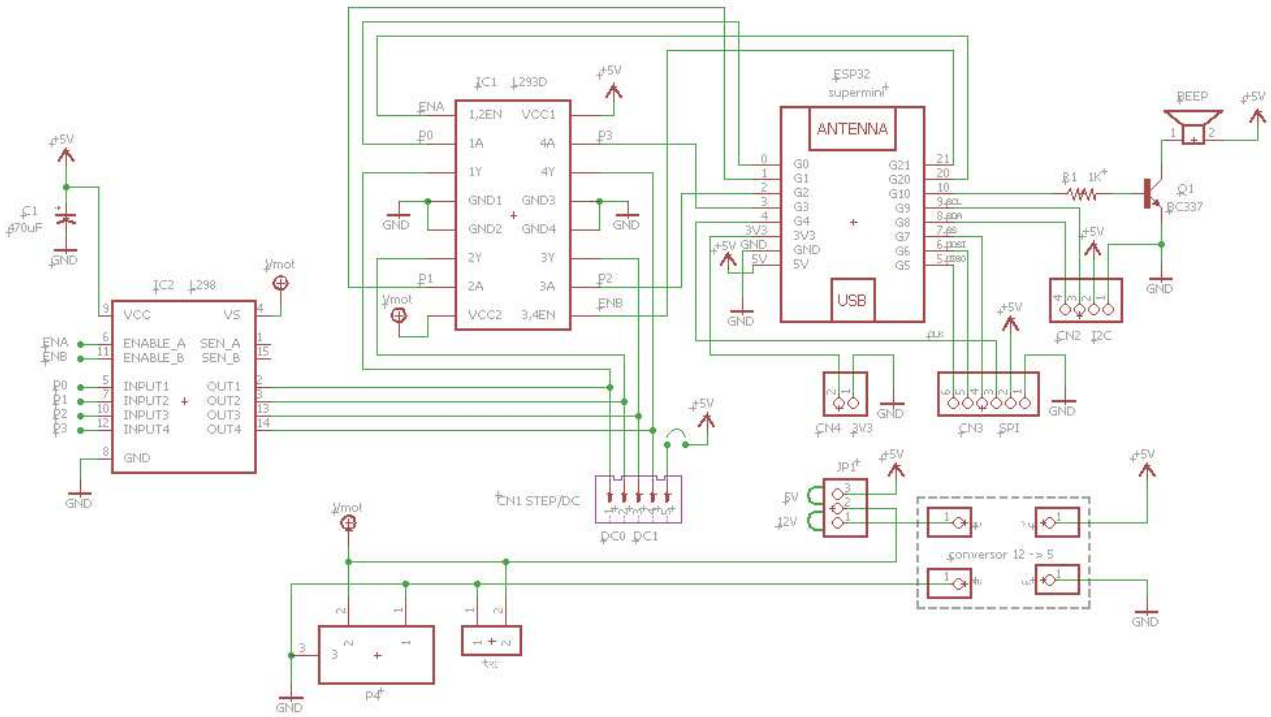


Fig. 1 – Diagrama elétrico da placa MOTORES_C3 Rev. 2.0

ASPÉCTO DA PLACA MOTORES_C3:

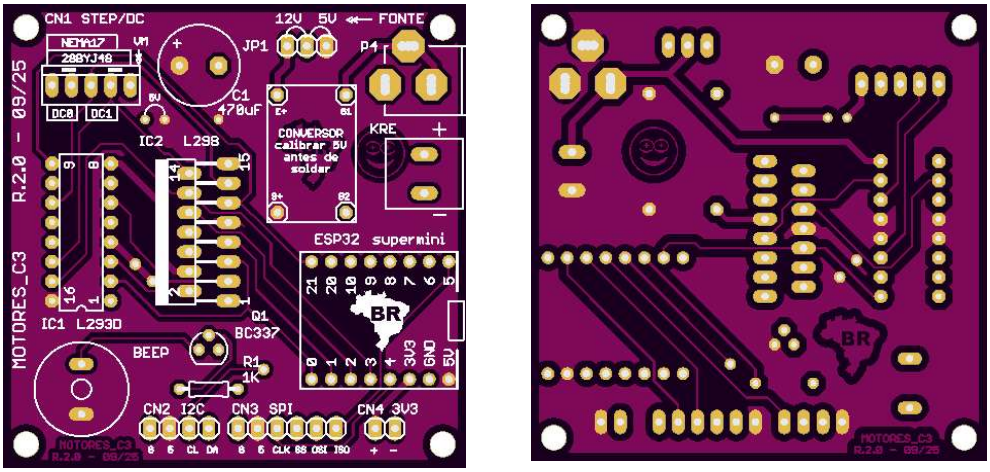


Fig. 2 – Placa MOTORES_C3 rev. 2.0 (aproximadamente em tamanho real)

BIBLIOTECA:

Para facilitar o desenvolvimento de aplicações a serem hospedadas em placas baseadas nos microcontroladores ESP32, foi criada uma biblioteca de nome motbepled.h, disponível em https://github.com/izyino/motbepled.h a qual deve ser utilizada conforme as instruções seguintes.

As funções descritas a seguir referem-se apenas àquelas relevantes para a placa MOTORES_C3. Registre-se que a biblioteca motbedled.h possui muitas outras funções que fazem sentido apenas para placas mais complexas que a MOTORES_C3.

#include < motbepled.h>

para incluir a biblioteca ao programa. Dependendo de onde a biblioteca estiver gravada, pode-se usar alternativamente o formato #include “**motbepled.h**”

motbepled x(t);

comando construtor que deve ser informado logo após o include, sendo t uma variável do tipo int8_t que define o tipo e o modo de operação do motor conectado a CN1, sendo possível os seguintes valores:

0 – Para motor DC
1 – Para motor 28byj-48, 2048 passos por volta, baixo torque, baixo consumo
2 – Para motor 28byj-48, 2048 passos por volta, alto torque, alto consumo
3 – Para motor 28byj-48, 4096 passos por volta, médio torque, médio consumo
4 – Para motor Nema17, 200 passos por volta, modo único

x.pinsStep0(0, 1, 2, 3, 20, 21);

comando que informa os pinos do microcontrolador ESP associados ao motor de passo, sendo os quatro primeiros das bobinas principais e os dois últimos dos sinais enable. O valor -1 significa que os sinais correspondentes não se aplicam no presente caso. Para uso da biblioteca motbepled.h com a placa MOTORES_C3 deve-se informar o comando pinsStep0 exatamente como mostrado acima, ou seja: x.pinsStep0 (0, 1, 2, 3, 20, 21); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.pinBeep(10);

comando que informa o pino do microcontrolador ESP associado ao beep. Para uso da biblioteca motbepled.h com a placa CTRL_H28BYJ48 deve-se informar o comando pinBeep exatamente como mostrado acima, ou seja: x.pinBeep (10); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.pinLed(8, 0);

comando que informa o pino do microcontrolador ESP associado ao Led e o nível lógico para o led aceso. Para uso da biblioteca motbepled.h com a placa CTRL_H28BYJ48 deve-se informar o comando pinLed exatamente como mostrado acima, ou seja: x.pinLed (8, 0); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.begin();

inicializa as diversas funções da biblioteca. Deve ser colocado na sessão de setup de todos os programas que se utilizem da biblioteca

x.runStep(0, steps, velstep, cwstep);

comando que ativa o motor de passo, de forma automática e assíncrona, conforme as seguintes variáveis:

steps – variável uint32_t contendo o número de passos a movimentar

velstep – variável unint8_t que define a velocidade da movimentação em RPM (rotações por minuto). Este valor pode variar entre 1 e 16, dependendo do motor utilizado de da corrente disponível na fonte de alimentação

cwstep – variável booleana que define o sentido da movimentação, sendo “true” para sentido horário e “false” para sentido anti-horário

x.stepstogo(0);

esta função retorna no formato uint32_t o número de passos ainda restantes para que o motor chegue ao seu destino. Zero significa que o motor já chegou ao seu último destino e já encontra-se parado. Antes de comandar qualquer movimentação deve-se consultar esta função para ter certeza que o motor encontra-se parado

x.runDC(n, time, veldc, cwdc);

comando que ativa o motor DC n.0, de forma automática e assíncrona, conforme as seguintes variáveis:

n – variável uint8_t com número do motor DC que será movimentado (0 ou 1):

time – variável uint32_t contendo o tempo em milisegundos que o motor DC ficará ativado

velDC – variável unint8_t que define a velocidade da movimentação, em termos de porcentagem entre 0 e 100. Sendo 0=0% motor parado, 100=100% motor com velocidade máxima.

cwDC – variável booleana que define o sentido da movimentação, sendo “true” para sentido horário e “false” para sentido anti-horário

x.timetogo(n);

esta função retorna no formato uint32_t, em milisegundos, o tempo ainda restante para que o motor DC n (n=0 ou 1) complete o último comando runDC. Se retornar zero significa que o motor DC n já está parado. Antes de comandar qualquer movimentação do motor DC n deve-se consultar esta função para ter certeza que o mesmo se encontra parado. A variável n é do tipo uint8_t

x.beep(bnum, bdur, bfreq, binter);

comando que ativa a emissão de beeps sonoros, de forma automática e assíncrona, conforme as seguintes variáveis:

bnum – variável inteira que especifica o número de beeps a serem emitidos

bdur – variável inteira que especifica a duração de cada beep, em milisegundos

bfreq – variável inteira que especifica a freqüência dos beeps, em Hertz (Hz). Os beeps passivos comuns respondem bem freqüências entre 200Hz e 5000Hz

binter – variável inteira que especifica a duração da pausa entre os beeps, em milisegundos

x.led(lnum, ldur, linter);

comando que ativa piscadas do Led (conectado ao pino 8 do módulo ESP32) , de forma automática e assíncrona, conforme as seguintes variáveis:

lnum – variável inteira que especifica o número de piscadas a serem emitidas

ldur – variável inteira que especifica a duração do Led acesso em cada piscada, em milisegundos

linter – variável inteira que especifica a duração do Led apagado em cada piscada, em milisegundos

x.setms(yms);

comando para inicializar o contador de milisegundos com o valor informado pela variável yms do tipo uint32_t. Imediatamente após inicializado, o contador começa ser subtraído de 1 a cada milisegundo

x.getms();

esta função retorna no formato uint32_t o estado atual do contador de milisegundos previamente inicializado pelo comando x.setms. Serve como alternativa para a função delay(), de forma assíncrona

x.stopStep(0);

esta função interrompe o movimento do motor de passo

x.stopDC(n);

esta função interrompe o movimento do motor DC n (n=0 ou 1). A variável n é do tipo uint8_t

x.stopLed();

esta função interrompe as piscadas do Led eventualmente em andamento

x.stopBeep();

esta função interrompe a emissão de beeps sonoros eventualmente em andamento

Exemplos de utilização da biblioteca

No início do programa:

```
#include < motbepled.h>
motbepled x(2);
```

na sessão do setup:

```
x.begin();
```

//movimenta o motor de passo (conectado em CN1), tipo 28BYJ-48,
//velocidade 3 RPM, sentido horário, 2048 passos:

```
//função principal:
x.runStep(0, 2048, 3, true);
//o motor começa a se movimentar imediatamente após a função runStep ser chamada
//para saber se o motor de passo já chegou ao destino, fazer
if (x.stepstogo(0)>0) {ainda não chegou ao destino. Está em movimento...};
//a qualquer momento o movimento do motor de passo pode ser interrompido
x.stopStep(0);
```

//movimenta o motor DC n.1,
//velocidade 75%, sentido anti-horário, durante 15segundos:

```
//função principal:
x.runDC(1, 15000, 75, false);
//o motor começa a se movimentar imediatamente após a função runDC ser executada
//para saber se o motor DC nº1 ainda está girando ou já esta parado, fazer
if (x.timetogo(1)>0) {ainda não terminou o último comando runDC. Está em movimento...};
//a qualquer momento o movimento do motor DC n.1 pode ser interrompido
x.stopDC(1);
```

//emite 10 beeps de 2KHz de 0,5s com pausa interbeeps de 0,25s:

```
//função principal:
x.beep(10, 500, 2000, 250);
//os beeps começam a ser emitidos imediatamente após a função beep ser chamada
//a qualquer momento a emissão dos beeps sonoros pode ser interrompida
x.stopBeep();
```

//pisca o Led 50 vezes com 0,25s aceso seguido de 0,10s apagado:

```
//função principal:
x.led(50, 250, 100);
//o led começa a piscarimediatamente após a função led ser chamada
//a qualquer momento as piscadas do Led podem ser interrompidas
x.stopLed();
```

//contagem de 4 segundos, de forma assíncrona:

```
//função principal:
x.setms(4000);
while (x.getms()>0){enquanto espera 4s, pode fazer coisas…}
//a variável x.xms começa a ser decrementada imediatamente após ter sido inicializada
```

Programas contidos no diretório “examples” da biblioteca

Exemplo n.1:

utilização das funções principais presentes na biblioteca motbepled.h, para controle de um motor de passo, beep, led e timer nas placas de circuito impresso baseadas no microcontrolador ESP32 com driver ULN2003 ou com a ponte H-H L293D. Pressupõe que um motor de passo tipo 28byj48 esteja conectado. O programa faz movimentos repetitivos do motor, dando n voltas a cada repetição, alternando o sentido e emitindo a cada ciclo dois beeps e vinte piscadas rápidas do led azul.

Exemplo n.2:

semalhante ao Exemplo n.1, com acréscimo de dois motores DC, os quais giram, um durante 4 segundos a 100% da velocidade e o outro durante 12 segundos a 45% da velocidade. Este exemplo pressupõe uma placa baseada em ESP32 com dois drivers ponte H-H tipo L293D. O programa faz ainda movimentos repetitivos do motor de passo, dando n voltas a cada repetição, alternando o sentido e emitindo a cada ciclo dois beeps e vinte piscadas rápidas do led azul.

Exemplo n.3:

exemplo de utilização da placa de circuito impresso baseadas no microcontrolador ESP32 com driver ULN2003 ou com a ponte H-H L293D. O programa atua como servidor web na modalidade “access point”. Pressupõe que um motor de passo tipo 28byj48 esteja conectado. Ver instruções nos comentários no próprio Exemplo n. 2.

Exemplo n.4:

utilização das funções principais presentes na biblioteca motbepled.h, associadas a comunicação WiFi, incluindo um servidor web, em placas de circuito impresso baseadas em ESP32. Pressupõe que um motor de passo tipo 28byj48 esteja conectado.

Na primeira execução deve-se conectar via access point na rede motbepled com a senha 00000000. Acessar então o IP 192.168.4.1. Em resposta, o programa exibe uma tela contendo a lista de redes WiFi ao alcance, para que o usuário informe qual rede será utilizada, juntamente com a senha correspondente. Dependendo do número de redes ao alcance, pode demorar um certo tempo até que a lista seja exibida.

Informar então o número da rede escolhida e a sua senha, seguido por um clique no botão "submit". Feito isso, basta apontar o browser para o IP fixo normalmente igual a 192.168.1.99 ou outro, conforme informado pelo monitor serial.

Exemplo n.5:

utilização das funções principais presentes na biblioteca motbepled.h, associadas a comunicação WiFi, incluindo um servidor web, em placas de circuito impresso baseadas em ESP32. Pressupõe que um motor de passo tipo 28byj48 esteja conectado.

Esse exemplo mostra o esboço de um programa para controle de alimentação de animais via WiFi. Pressupõe que um motor de passo tipo 28byj48 esteja conectado., supostamente usado para despejar uma dose de ração a cada 45 graus girados.

Acessar via browser o IP informado no monitor serial. Fazer então a programação da alimentação, de até 4 vezes ao dia, informando para cada refeição: hora, minuto e a quantidade de doses (uma dose=45graus).

OBSERVAÇÃO: Deve-se rodar antes desse exemplo n.5, pelo menos uma vez, o programa exemplo n.4 para que o nome da rede WiFi a ser utilizada, juntamente com a sua senha fiquem armazenadas na memória flash do ESP32

IMPORTANTE: Antes da execução de qualquer um dos exemplos citados, deve-se conferir e alterar quando necessário a pinagem dos motores, beep e Led atribuída pelos comandos pinsStep, pinsDC, pinBeep e pinLed, para que correspondam fielmente ao hardware utilizado