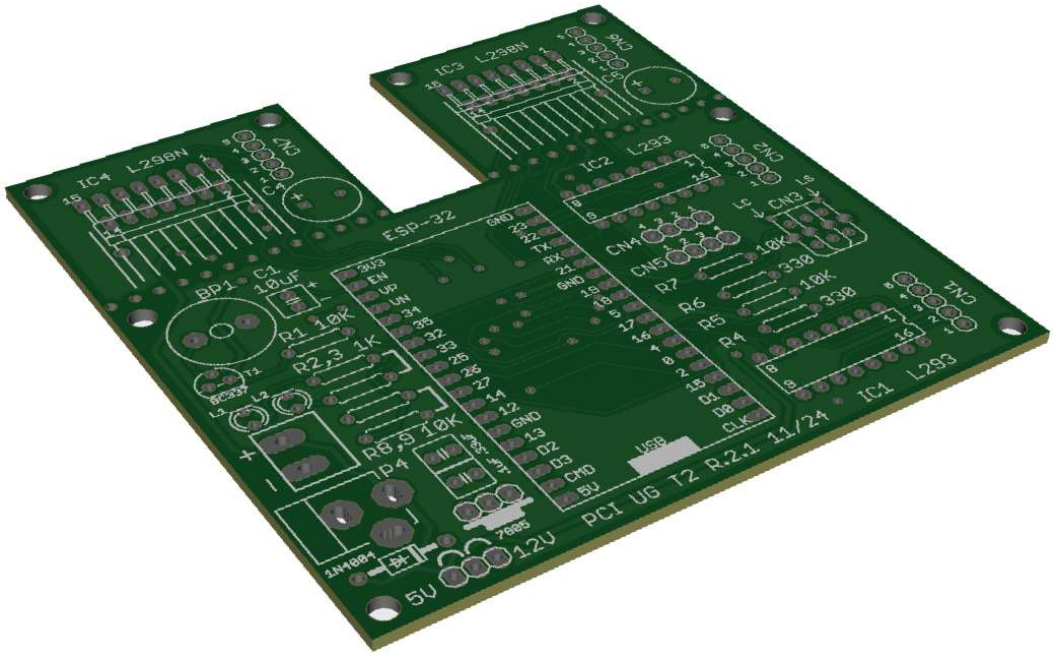
PCI DE USO GERAL

TIPO 2

Para controle de motores de passo
combinados com motores DC e outros dispositivos



DOCUMENTAÇÃO BÁSICA PRELIMINAR

Placas tipo PCI_UG_T2, versão 2.1 e posteriores

Manual rev. 2.1 – 11/2024
Eventuais atualizações desse manual podem ser encontradas em: https://github.com/izyino/manuais

1 - INTRODUÇÃO

A presente placa de circuito impresso PCI UG T2 foi concebida para permitir uma melhor organização na montagem de circuitos envolvendo motores de passo, motores DC, sensores reflexivos, sensores de distância e outros dispositivos, concentrando em uma placa de circuito impresso de dimensões reduzidas todo o hardware destinado a desempenhar as mencionadas funções, incluindo controle via Wi-Fi através de acesso a redes já existentes e como também gerenciamento de sua própria rede Wi-Fi, criada e administrada pelo poderoso microcontrolador ESP-32, tudo devidamente acomodado em uma placa de circuito impresso de apenas 90 x 90mm.

Dentre as aplicações para a placa PCI UG T2 Rev.2.1 destacam-se: alimentadores para animais domésticos, automação de portas, janelas, cortinas e persianas, robótica em geral, traçadores e registradores gráficos, ferremodelismo, furadeiras, frezadeiras, projetos educacionais em mecatrônica e uma infinidade de outras aplicações. As principais características da PCI UG T2 Rev.2.1 são as seguintes:

- Placa de circuito impresso dupla face para suporte dos módulos utilizados, medindo apenas 90 x 90 mm (ou 60 x 90mm sem os apêndices destacáveis)
- Utilização do microcontrolador Tensilica Xtensa 32-bit LX6 dual-core ESP32 em sua versão WROOM com PCI de 38 pinos, com 448Kbytes de ROM, 520Kbytes de SRAM, 8+8Kbytes de SRAM, RTC, 1Kbit de eFuses, clock de 240MHz
- Suporte motores de passo do tipo 28BYJ-48, Nema17 e motores DC de até 4A, através de circuitos ponte H com a utilização dos CIs L293D ou L298N, em diversas combinações
- Suporte para até dois sensores reflexivos do tipo TCRT5000 ou quaisquer outros e suporte para sensor de distância do tipo VL53L0 ou quaisquer outros dispositivos com interface I2C
- Possibilidade de alimentação por 5 ou 12Vcc, selecionáveis por jumper, com regulador instalado na própria placa
- Possibilidade de Wi-Fi como ponto de acesso e/ou servidor, permitindo atualização de seu firmware via internet, automaticamente
- Beep e Led para sinalização sonora e visual. Led indicador de rede Wi-Fi conectada
- Placa de circuito impresso com possibilidade de se destacar ou seccionar um ou dois apêndices dos CIs L298N quando não utilizados
- Biblioteca auxiliar para controle de forma assíncrona, dos motores, beep e Led, disponível em https://github.com/izyino/motbepled
- Hardware flexível e aberto a uma infinidade de outras aplicações

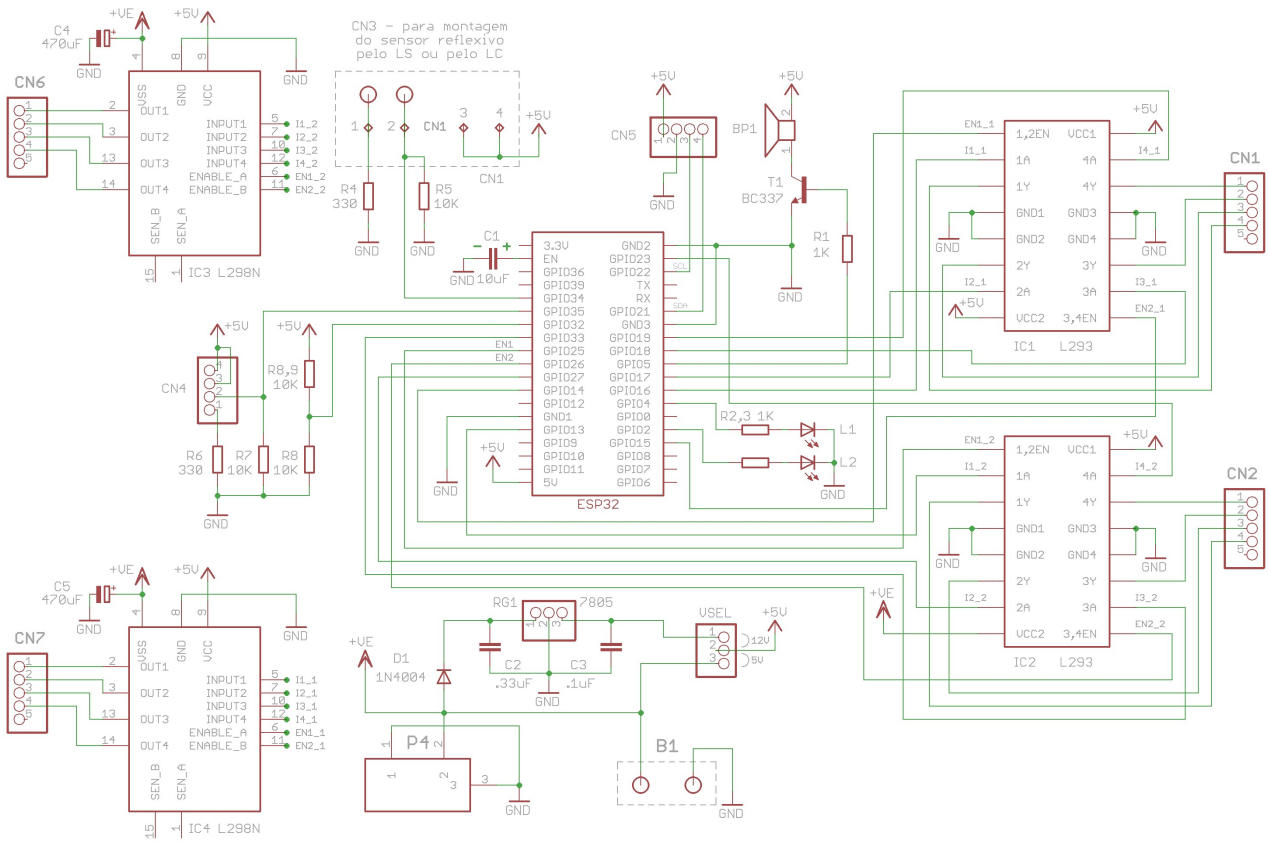
2 - COMPONENTES UTILIZADOS (nem todos ao mesmo tempo)

- 1 x Módulo microcontrolador ESP-32 de 38 pinos
- 2 x CI ponte H, L293D
- 2 x CI ponte H, L298N com dissipador
- 2 x Soquete slim para CI de 16 pinos
- 2 x Sensor reflexivo TCRT5000
- 1 x Sensor de distência VL53L0
- 1 x Regulador 7805
- 1 x Beep TMB12A05, ϕ 12mm, alt 9,6mm, esp 7,8mm
- 1 x Transistor BC337
- 1 x Diodo 1N4004
- 2 x Led colorido 3mm
- 4 x Resistores de 10K, 1/8W
- 3 x Resistor de 1K, 1/8W
- 2 x Resistor 330, 1/8W
- 2 x Capacitor eletrolítico de 470uF, 25V ou mais
- 1 x Capacitor eletrolítico de 10uF, 25V ou mais
- 1 x Capacitor disco .33uF
- 1 x Capacitor disco .1uf
- 1 x Conector P4 fêmea, solda placa
- 1 x Borne KRE de 2 pinos
- 4 x Barra de 5 pinos (conectores CN1, CN2, CN6 e CN7)
- 2 x Barra de 4 pinos (CN4 e CN5)
- 1 x Barra de 3 pinos (conector para seleção de voltagem 5Vcc ou 12Vcc)
- 1 x Placa de circuito impresso PCI UG T2 Versão 2.1 ou posteriores

3 – PINOS E CONEXÕES

GPIO 16, 17, 18, 19 – Motor de passo n.0 / motor DC n.0 e n.1
GPIO 13, 27, 33, 23 – Motor de passo n.1 / motor DC n.2 e n.3
GPIO 14 – Enable do motor de passo n.0 / PWM motor DC n.0
GPIO 15 – Enable do motor de passo n.0 / PWM motor DC n.1
GPIO 25 – Enable do motor de passo n.1 / PWM motor DC n.2
GPIO 26 – Enable do motor de passo n.1 / PWM motor DC n.3
GPIO 5 – Beep de uso geral
GPIO 4 – Led de uso geral
GPIO 2 – Led indicador de rede Wi-Fi conectada
GPIO 32 – Divisor de tensão para leitura do nível do Vcc
GPIO 34 – Sensor reflexivo n.1
GPIO 35 – Sensor reflexivo n.2
GPIO 21, 22 – Sensor de distância I2C - SDA/SCL

5 - DIAGRAMA ESQUEMÁTICO



CONSIDERAÇÕES SOBRE O CIRCUITO:

O CI4 (L298N) e o CI1 (L293D), associados aos conectores CN7 e CN1 possuem os sinais de controle paralelos, ou seja, os comandos enviados ao(s) motor(es) conectado(s) ao CN7 são enviados também ao(s) motor(es) conectado(s) ao CN1. O mesmo ocorre em relação ao CI3 (L298N) e o CI2 (L293D), associados aos conectores CN6 e CN2. Assim, recomenda-se que seja usado CN7 ou CN1 bem como CN6 ou CN2.

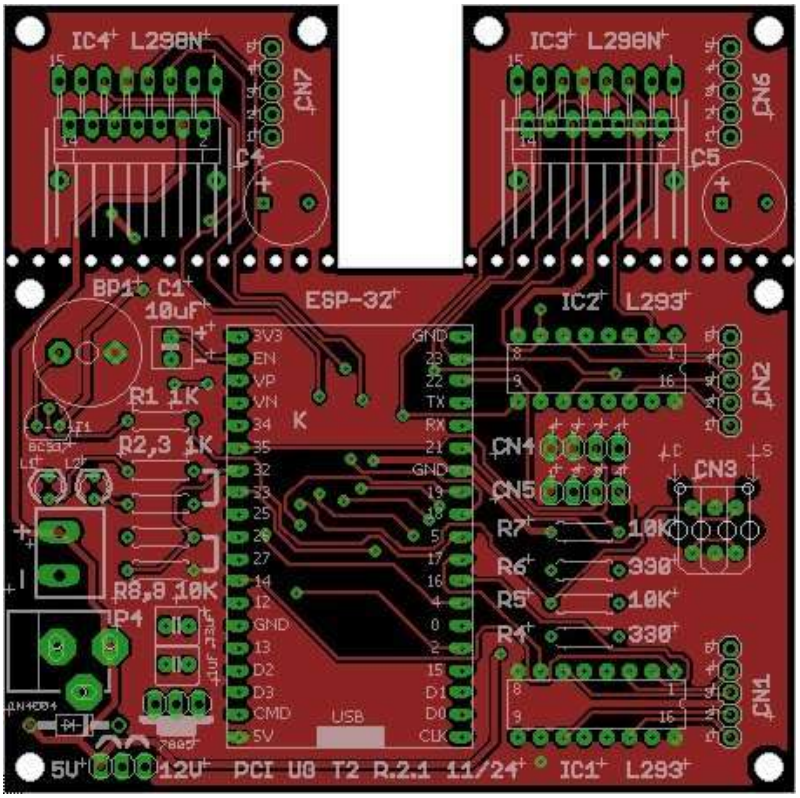
Deve-se utilizar o CI4 (L298N) para motores Nema17 ou motores DC de até 4A conectados ao CN7. Nestes casos não montar o CI1 (L293D), seu soquete e nem seu conector CN1.

Para motores 28BYJ-48 ou motores DC de até 1.5A, deve-se utilizar o CI1 (L293D) associado ao conector CN1. Nestes casos não montar o CI4 (L298N), o capacitor C4 e nem o seu conector CN7.

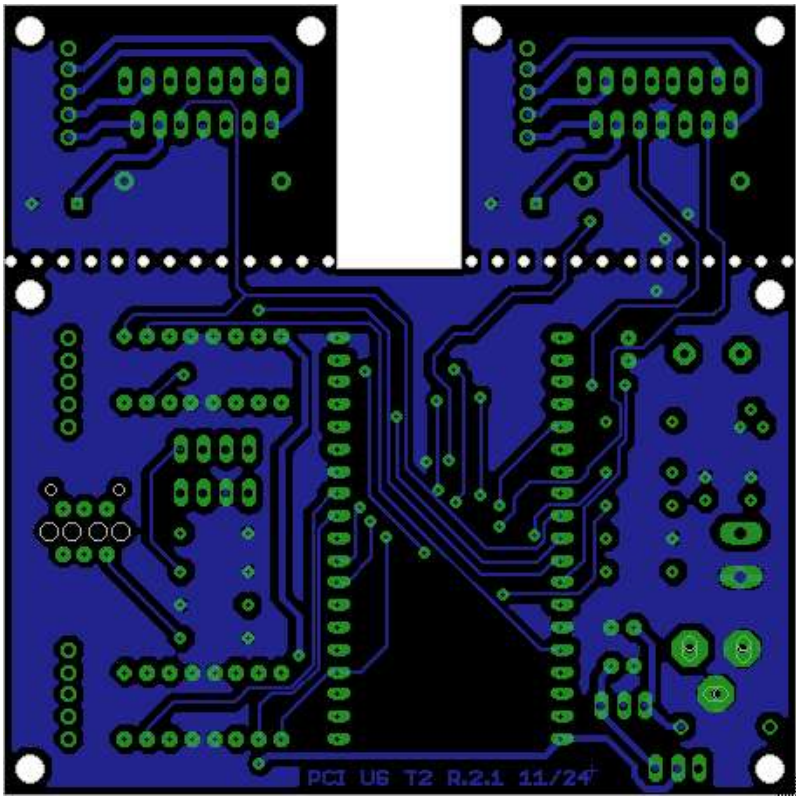
Deve-se utilizar o CI3 (L298N) para motores Nema17 ou motores DC de até 4A conectados ao CN6. Nestes casos não montar o CI2 (L293D), seu soquete e nem seu conector CN2.

Para motores 28BYJ-48 ou motores DC de até 1.5A, deve-se utilizar o CI2 (L293D) associado ao conector CN2. Nestes casos não montar o CI3 (L298N), o capacitor C5 e nem o seu conector CN6.

6 - ASPÉCTO DA PLACA

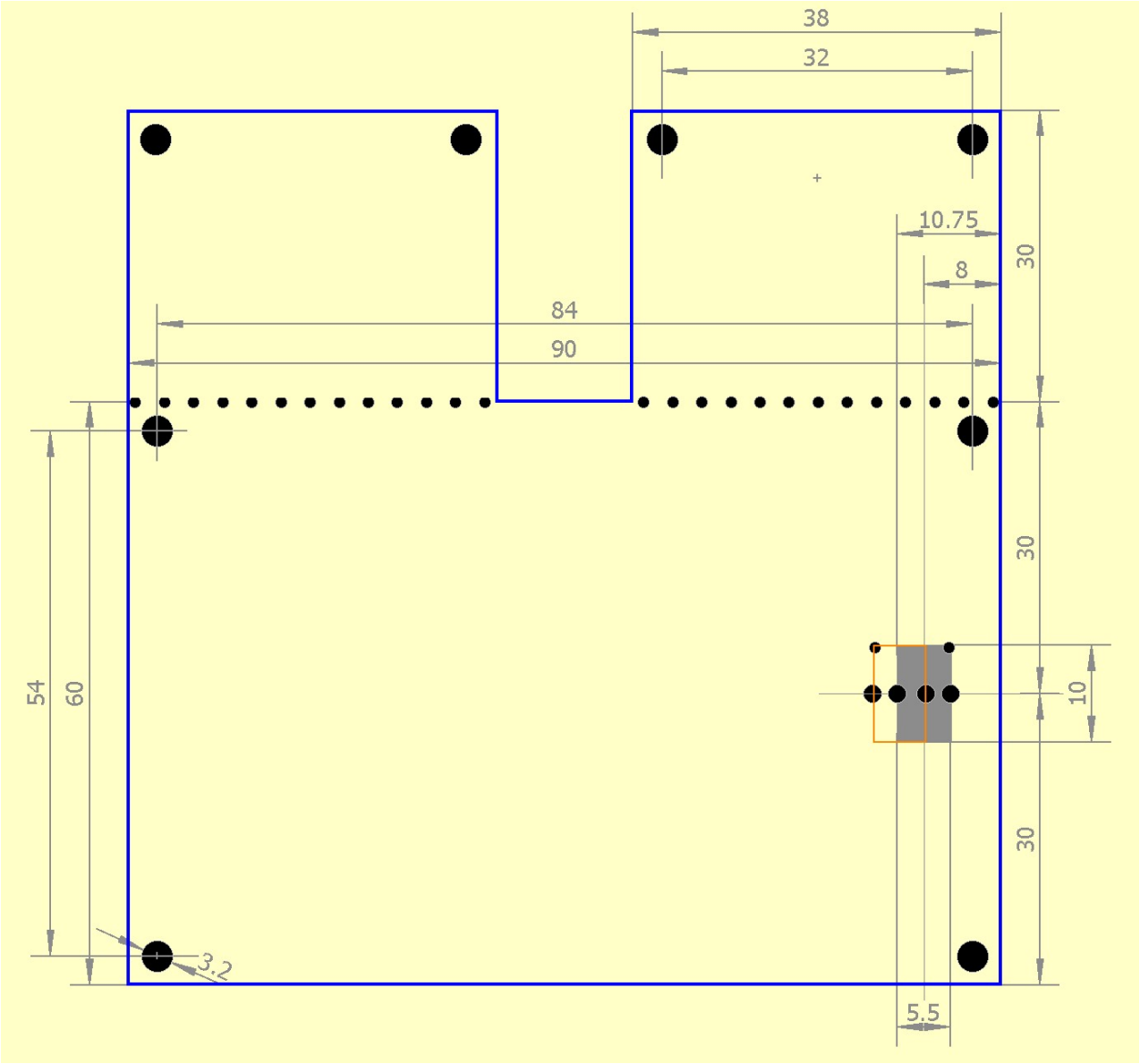


Lado dos componentes



Lado da solda

7 - DIMENSÕES E MECÂNICA DA PLACA



Observações:

1) Caso não se utilize motores Nema17 ou quaisquer outros motores de corrente até 4A que necessitem dos CIs L298N, a placa poderá ser seccionada na linha composta pela série de pequenos furos, que definem a posição do possível “picote”, eliminando assim um ou dois apêndices complementares da placa

2) Se o sensor reflexivo for montado pelo lado da solda, o centro da sua projeção deve estar a 8mm da borda, conforme indicado acima pela área em cinza Se for montado pelo lado dos componentes, deslocar o centro da sua projeção para 10.75mm da borda, conforme indicado acima na cor laranja. Em qualquer caso, a projeção mede 5.5 x 10mm

8 - DETALHES DE MONTAGEM

COMPONENTES ESSENCIAIS EM QUALQUER VERSÃO:

- Módulo ESP-32 WROOM de 38 pinos
- Capacitor C1 (10uF/16V)
- Jumper Vsel (3 pinos)
- Conector P4 e/ou o borne KRE B1

COMPONENTES MONTADOS OPCIONALMENTE, DEPENDENDO DA VERSÃO:

- Com beep de uso geral: montar R1 (10K), T1 (BC337) e o BP1 (TMB12A05)
- Com led de uso geral: montar R2 (1K) e o L1 (led 3mm)
- Com led monitor de conexão em rede Wi-Fi: montar R3 (1K) e o L2 (led 3mm)
- Com sensor reflexivo n.1: montar R4 (330), R5(10K) e o sensor TCRT5000 em CN3
- Com sensor reflexivo n.2: montar R6 (330), R7(10K), CN2 (4 pinos) e o sensor TCRT5000 em CN4
- Com sensor de distância: montar CN5 (4 pinos) e o sensor VL53L0 em CN5
- Com medidor da tensão do Vcc: montar R8 (10K) e o R9 (10K)
- Com motores em CN1: Não montar, IC4 (L298N), o capacitor C5 e o conector CN7 (5 pinos). Montar soquete de 16 pinos, IC1 (L293D) e o conector CN1 (5 pinos). Motores possíveis: 1 x 28BYJ-48 ou até 2 x DC de até 1.5A conectados aos pinos 3,4 e 1,2 do conector CN1. Alimentação dos motores: **Sempre 5V**
- Com motores em CN2: Não montar, IC3 (L298N), o capacitor C4 e o conector CN6 (5 pinos). Montar soquete de 16 pinos, IC2 (L293D) e o conector CN2 (5 pinos). Motores possíveis: 1 x 28BYJ-48 ou até 2 x DC de até 1.5A conectados aos pinos 3,4 e 1,2 do conector CN2. Alimentação dos motores: 5V ou 12V, dependendo da posição do jumper Vsel
- Com motores em CN6: montar, IC3 (L298N), o capacitor C4 e o conector CN6 (5 pinos). Não montar soquete de 16 pinos, IC2 (L293D) e o conector CN2 (5 pinos). Motores possíveis: Se fonte de 12V: 1 x Nema17 ou até 2 x DC de até 4A conectados aos pinos 3,4 e 1,2 do conector CN6. Se fonte de 5V: 1 x 28BYJ-48 ou até 2 x DC de até 1.5A conectados aos pinos 3,4 e 1,2 do conector CN6
- Com motores em CN7: montar, IC4 (L298N), o capacitor C5 e o conector CN7 (5 pinos). Não montar soquete de 16 pinos, IC1 (L293D) e o conector CN1 (5 pinos). Motores possíveis: Se fonte de 12V: 1 x Nema17 ou até 2 x DC de até 4A conectados aos pinos 3,4 e 1,2 do conector CN7. Se fonte de 5V: 1 x 28BYJ-48 ou até 2 x DC de até 1.5A conectados aos pinos 3,4 e 1,2 do conector CN7

- Se for usado fonte externa de 5V, colocar um jumper entre o pino central e o pino da esquerda do conector Vsel.
- Se for usado fonte externa de 12V, colocar um jumper entre o pino central e o pino da direita do conector Vsel

OBSERVAÇÕES:

1) Para utilização de motores de passo, os pinos GPIO 25, 26 e GPIO 14, 15 devem ser mantidos em nível alto (enable das duas sessões dos CIs L293D ou dos CIs L298N).

2) Para utilização de motor(es) DC de até 1.5A, utilizar o conector CN1, sendo os pinos GPIO 14 e 15 usados como controle PWM de velocidade para os motores DC n.0 e DC n.1 respectivamente. Nesse caso, o motor DC n.0 deverá estar conectado aos pinos 3 e 4 do conector CN1 e o motor DC n.1 (se houver) deverá estar conectado aos pinos 1 e 2 do conector CN1.

3) Para utilização de motor(es) DC de até 2.5A, utilizar o conector CN7, sendo os pinos GPIO 14 e 15 usados como controle PWM de velocidade para os motores DC n.0 e DC n.1 respectivamente. Nesse caso, o motor DC n.0 deverá estar conectado aos pinos 3 e 4 do conector CN7 e o motor DC n.1 (se houver) deverá estar conectado aos pinos 1 e 2 do conector CN7.

4) Para utilização de motor(es) DC de até 1.5A, utilizar o conector CN2, sendo os pinos GPIO 25 e 26 usados como controle PWM de velocidade para os motores DC n.2 e DC n.3 respectivamente. Nesse caso, o motor DC n.2 deverá estar conectado aos pinos 3 e 4 do conector CN2 e o motor DC n.3 (se houver) deverá estar conectado aos pinos 1 e 2 do conector CN2.

5) Para utilização de motor(es) DC de até 2.5A, utilizar o conector CN6, sendo os pinos GPIO 25 e 26 usados como controle PWM de velocidade para os motores DC n.2 e DC n.3 respectivamente. Nesse caso, o motor DC n.2 deverá estar conectado aos pinos 3 e 4 do conector CN6 e o motor DC n.3 (se houver) deverá estar conectado aos pinos 1 e 2 do conector CN6.

6) **ATENÇÃO:** O CI1 L293D associado ao conector CN1 é **sempre alimentado com 5Vcc** proveniente da fonte externa ou do regulador 7805. Já o CI2 (L293D) e os CI3 e CI4 (L298N) (se existirem) serão alimentados sempre diretamente da fonte externa. Isso possibilita o uso de motores de 5V (no conector CN1) simultaneamente com motores 12V (nos conectores CN2, CN6 e CN7).

7) O sensor reflexivo n.1, quando existir, poderá ser soldado pelo lado dos componentes ou do lado da solda, sempre em CN3. Observar atentamente o correto encaixe das saliências de plástico do sensor nos furos correspondentes da placa

9 - 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 a qual deve ser utilizada conforme as instruções seguintes.

#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(t0, t1);

comando construtor que deve ser informado logo após o include, sendo t0, t1 variáveis do tipo uint8_t que definem o tipo de motor conectado a CN1 e CN2 respectivamente, 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(16, 17, 18, 19, 15, 14);

comando que informa os pinos do microcontrolador ESP associados ao motor de passo n.0, **caso exista**, 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 PCI_UG_T1 deve-se informar o comando pinsStep0 exatamente como mostrado acima, ou seja: x.pinsStep0 (16, 17, 18, 19, 15, 14); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.pinsStep1(13, 27, 33, 23, 25, 26);

comando que informa os pinos do microcontrolador ESP associados ao motor de passo n.1, **caso exista**, 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 PCI_UG_T1 deve-se informar o comando pinsStep0 exatamente como mostrado

acima, ou seja: x.pinsStep0 (13, 27, 33, 23, 25, 26); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.pinsDC0(16, 17, 15);

comando que informa os pinos do microcontrolador ESP associados ao motor DC n.0, **caso exista**, sendo os dois primeiros do motor e o terceiro da velocidade PWM. O valor -1 significa que os sinais correspondentes não se aplicam no presente caso. Para uso da biblioteca motbepled.h com a placa PCI_UG_T1 deve-se informar o comando pinsDC0 exatamente como mostrado acima, ou seja: x.pinsDC0 (16, 17, 15); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.pinsDC1(18, 19, 14);

comando que informa os pinos do microcontrolador ESP associados ao motor DC n.1, **caso exista**, sendo os dois primeiros do motor e o terceiro da velocidade PWM. O valor -1 significa que os sinais correspondentes não se aplicam no presente caso. Para uso da biblioteca motbepled.h com a placa PCI_UG_T1 deve-se informar o comando pinsDC0 exatamente como mostrado acima, ou seja: x.pinsDC1 (18, 19, 14); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.pinsDC2(13, 27, 25);

comando que informa os pinos do microcontrolador ESP associados ao motor DC n.2, **caso exista**, sendo os dois primeiros do motor e o terceiro da velocidade PWM. O valor -1 significa que os sinais correspondentes não se aplicam no presente caso. Para uso da biblioteca motbepled.h com a placa PCI_UG_T1 deve-se informar o comando pinsDC0 exatamente como mostrado acima, ou seja: x.pinsDC2 (13, 27, 25); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.pinsDC3(33, 23, 26);

comando que informa os pinos do microcontrolador ESP associados ao motor DC n.3, **caso exista**, sendo os dois primeiros do motor e o terceiro da velocidade PWM. O valor -1 significa que os sinais correspondentes não se aplicam no presente caso. Para uso da biblioteca motbepled.h com a placa PCI_UG_T1 deve-se informar o comando pinsDC0 exatamente como mostrado acima, ou seja: x.pinsDC3 (33, 23, 26); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.pinBeep(5);

comando que informa o pino do microcontrolador ESP associado ao beep. Para uso da biblioteca motbepled.h com a placa PCI_UG_T1 deve-se informar o comando pinBeep exatamente como mostrado acima, ou seja: x.pinBeep (5); e deve ser informado na sessão de setup de todos os programas, sempre antes do x.begin()

x.pinLed(4, 1);

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 PCI_UG_T1 deve-se informar o comando pinLed exatamente como mostrado acima, ou seja: x.pinLed (4, 1); 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(n, steps, velstep, cwstep);

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

n – variável uint8_t contendo o número do motor que será movimentado (0 ou 1). Se n=0, o motor de passo deverá estar conectado ao CN1. Se n=1, o motor de passo deverá estar conectado ao CN2

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). Para motores 28BYJ-48 esse valor pode variar entre 1 e 16

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

x.stepstogo(n);

esta função retorna no formato uint32_t o número de passos ainda restantes para que o motor n (n=0 ou 1) chegue ao seu destino. Zero significa que o motor n já chegou ao seu último destino e já se encontra parado. Antes de comandar

qualquer movimentação deve-se consultar esta função para ter certeza que o motor n se encontra parado. A variável n é do tipo uint8_t

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

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

n – variável uint8_t com número do motor DC que será movimentado (0, 1, 2 ou 3):
Se n=0, o motor DC deverá estar conectado aos pinos 1 e 2 do CN1
Se n=1, o motor DC deverá estar conectado aos pinos 3 e 4 do CN1
Se n=2, o motor DC deverá estar conectado aos pinos 1 e 2 do CN2
Se n=3, o motor DC deverá estar conectado aos pinos 3 e 4 do CN2

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 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

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)

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, 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(n);

esta função interrompe o movimento do motor de passo n (n=0 ou 1)

x.stopDC(n);

esta função interrompe o movimento do motor DC n (n=0, 1, 2 ou 3)

x.stopBeep();

esta função interrompe a emissão de beeps sonoros

x.stopLed();

esta função interrompe as piscadas do Led

Exemplos de utilização da biblioteca

No início do programa:
```
#include <motbepled.h>
motbepled x(2, 0);
```

na sessão do setup:
```
x.begin();
```

//movimenta o motor de passo nº0 (conectado em CN1), tipo 28BYJ-48,
//velocidade 3, sentido horário, 2048 passos:

```
//função principal:
x.runStep(0, 2048, 3, true);
//o motor começa a se movimentar imediatamente após a função runStep ser executada
//para saber se o motor de passo n.0 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 nº0 pode ser interrompido
x.stopStep(0);
```

//movimenta o motor DC nº3 (conectado aos pinos 1 e 2 do CN2),
//velocidade 75%, sentido anti-horário, durante 15segundos:

```
// função principal:
x.runDC(3, 15000, 75, false);
//o motor começa a se movimentar imediatamente após a função runDC ser executada
//para saber se o motor DC nº3 ainda está girando ou já esta parado, fazer
if (x.timetogo(3)>0) {ainda não terminou o último comando runDC. Está em movimento...};
//a qualquer momento o movimento do motor DC nº3 pode ser interrompido
x.stopDC(3);
```

//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 executada
//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 piscar imediatamente após a função led ser executada
//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 a cada milisegundo imediatamente após a função setms ser executada
```

O diretório “examples” da biblioteca motbepled contém diversos exemplos de programas, das mais variadas aplicações, como: movimentação de motores, emissão de beeps, acesso a redes WiFi, servidor web, e muitas outras.

IMPORTANTE: Antes da execução de qualquer um dos exemplos, 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

---