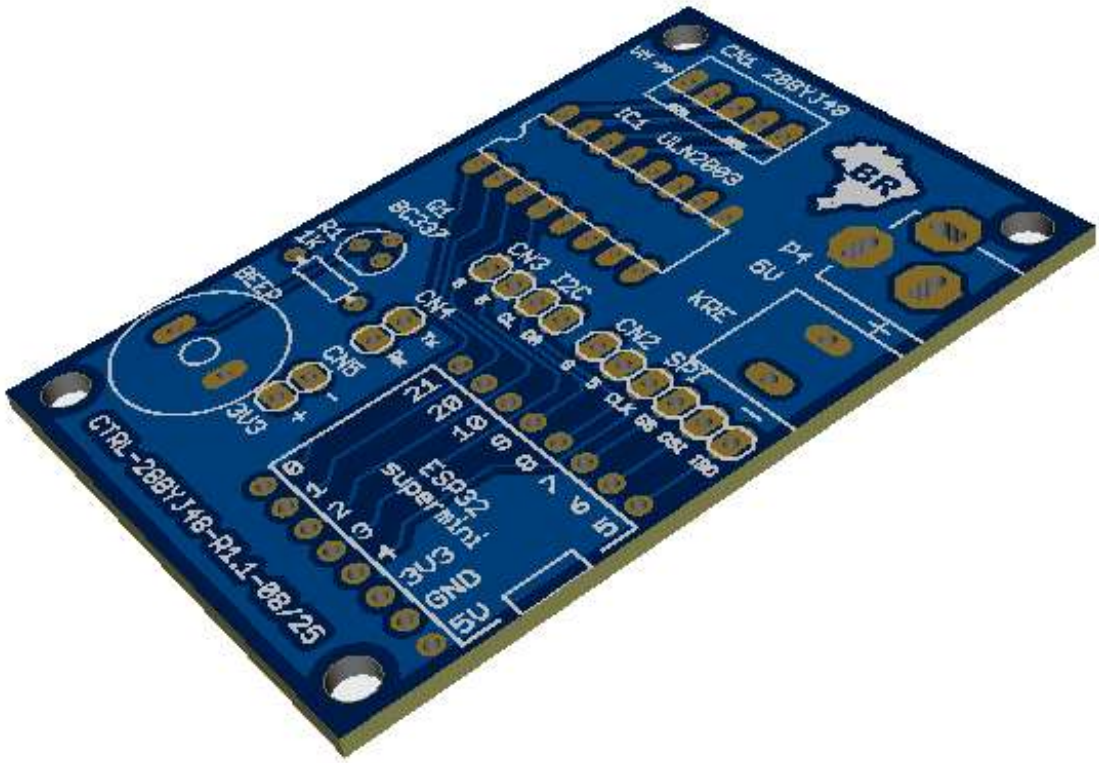
PCI
CTRL-28BYJ48



MANUAL DO USUÁRIO

Aplicável.as placas:

CTRL-28BYJ48 versão 1.1 e posteriores
com o módulo microcontrolador ESP32 C3 supermini de 16 pinos

Manual rev. 1.2 - 06/2025 **- Para atualizações, acessar:**https://github.com/izyino/X28BYJ48.h

INTRODUÇÃO:

O presente placa de circuito impresso intitulada CTRL-28BYJ48 foi concebida para facilitar a construção de dispositivos que necessitem de controle via WiFi e de um motor de passo do tipo 28BYJ48, contemplando em uma única placa de circuito impresso de apenas 4 x 6 cm, o moderno microcontrolador ESP32 C3 em sua versão supermini e o driver do motores de passo, substituindo assim os antigos, obsoletos e saudosos Arduinos e seus “shields” sobrepostos em forma de sanduíche.

A versão 1.1 da placa de circuito impresso CTRL-28BYJ48 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 modelo 28BYJ48 ou similar
- Utilização do driver ULN2003
- Beep sonoro passivo
- Conectores para acesso aos sinais I2C, SPI, Tx/Rx
- Completa biblioteca para perfeito controle do motor de passo e do beep sonoro

CONECTORES EXISTENTES NA PLACA CTRL-28BYJ48:

CN1 – 28BYJ48:

1 – Sem conexão (fio vermelho)
2 – L1 (fio laranja)
3 – L1 (fio amarelo)
4 – L2 (fio rosa)
5 – L2 (fio azul)

CN2 – SPI:

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

CN3 – I2C:

1 – Gnd
2 – Vcc (+5V)
3 – SCL
4 – SDA

CN4 – Tx Rx:

1 – Tx
2 – Rx

CN5 – 3V3:

1 – Gnd
2 – +3V3

RELAÇÃO DE COMPONENTES:

Placa de circuito impresso Rev.1.1
Módulo ESP32 C3 supermini, 16 pinos
Driver ULN2003, dip 16
Conector molex 5 pinos com polarizador (CN 1, para o motor)
Conector P4 fêmea, 90 graus, solda placa
Beep passivo (opcional)
Transistor BC337 (opcional)
Resistor de 1K (opcional)
Conector KRE de 2 pinos (opcional)
Conector 6 pinos macho (CN2 – SPI, opcional)
Conector 4 pinos macho (CN3 – I2C, opcional)
Conector 2 pinos macho (CN4 – TxRx, opcional)
Conector 2 pinos macho (CN5 – 3V3, opcional)

DIAGRAMA ELÉTRICO:

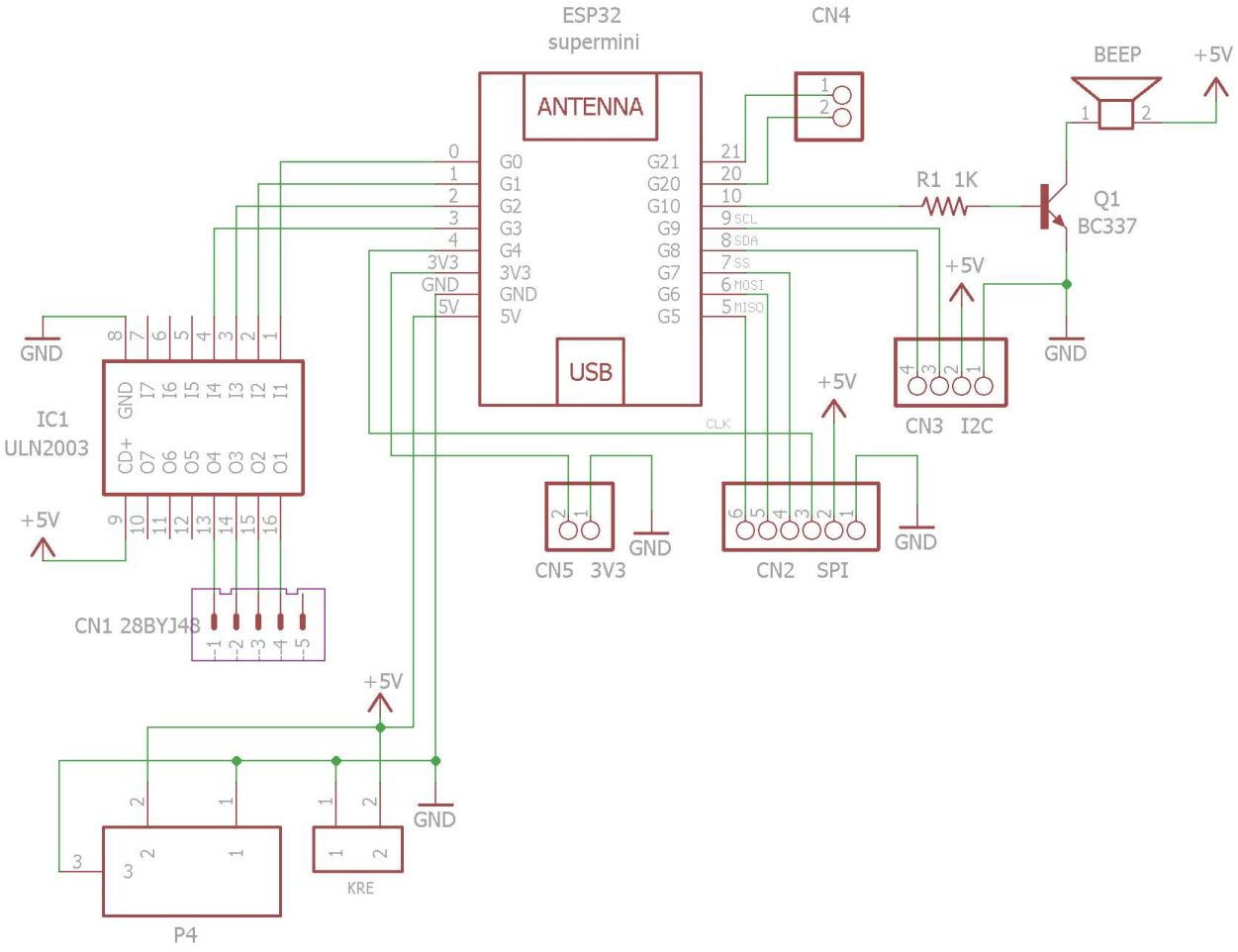


Fig. 1 – Diagrama elétrico da placa CTRL-28BYJ48 Rev. 1.1a

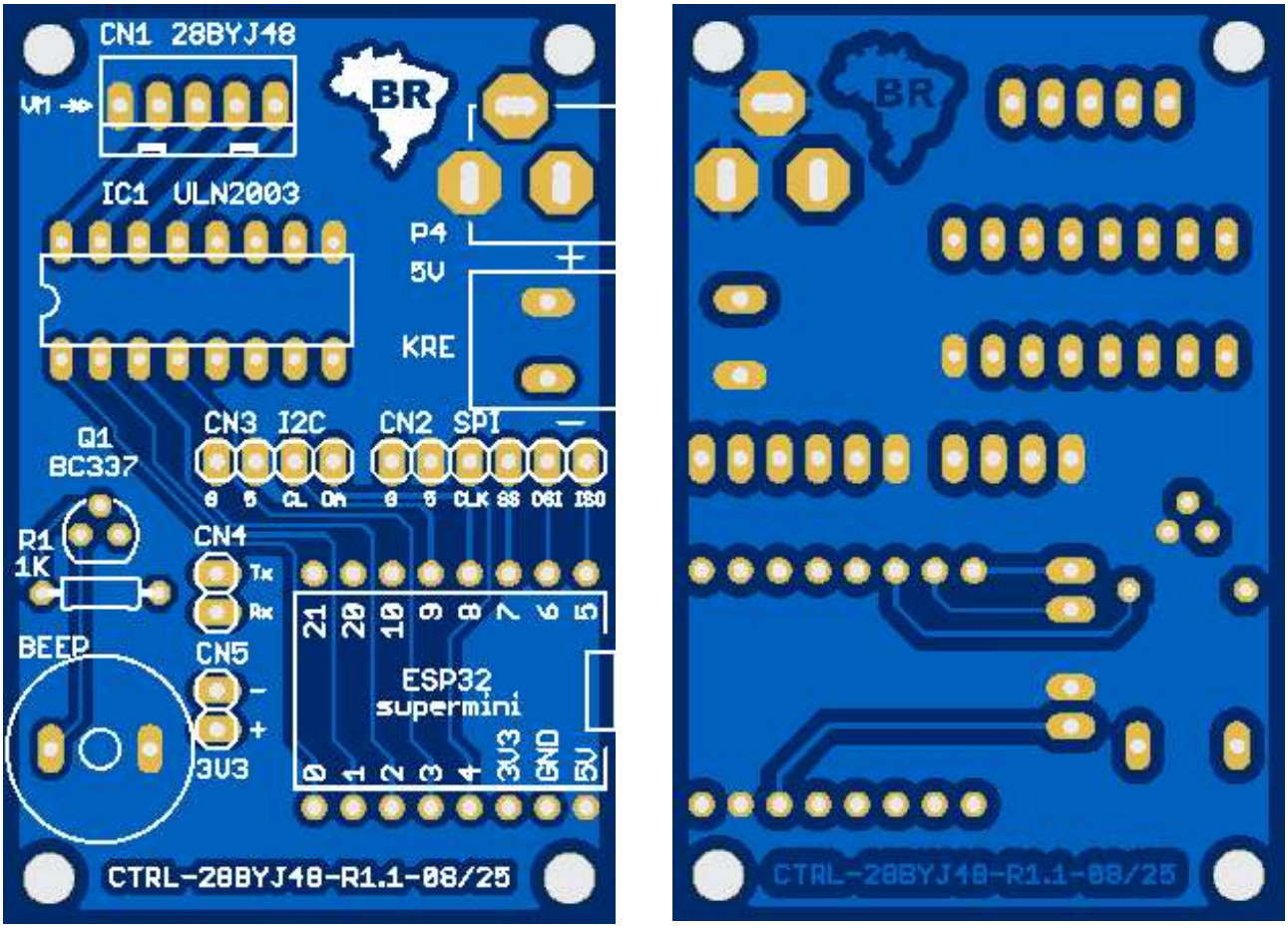


Fig. 2 – Placa CTRL-28BYJ48 rev. 1.1 (escala 2:1)

BIBLIOTECA:

Visando facilitar o desenvolvimento de aplicações a serem hospedadas nas placas CTRL-28BYJ48, foi criada uma biblioteca de nome X28BYJ48.h, disponível em https://github.com/izyino/X28BYJ48.h a qual dispõe das seguintes funções:

#include < X28BYJ48.h>

para incluir a biblioteca ao programa. Dependendo de onde a biblioteca estiver gravada, pode-se usar alternativamente o formato #include "**X28BYJ48**"

X28BYJ48 x(t);

comando construtor que deve ser informado logo após o include, sendo t uma variável do tipo uint8_t que define o tipo de motor conectado a CN1, sendo possível os seguintes valores:

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

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(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.where();

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.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();

esta função interrompe o movimento do motor de passo

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

Como alternativa para algumas das funções e comandos acima, pode-se acessar diretamente algumas variáveis internas da biblioteca. Entretanto, tais acessos se feitos de forma indevida, podem provocar erros e paradas inesperadas. Recomenda-se portanto que as variáveis relacionadas sejam acessadas diretamente apenas em último caso e apenas por programadores experientes:

variáveis de controle dos motores de passo (índice n abaixo entre 0 e 1)

x.xtipostep	tipo do motor de passo (pode ser1,2 ou 3)
x.xsteps	quantidade de passos a mover
x.xvelstep	velocidade (RPM)
x.xvelnow	velocidade no momento (RPM, vide .cpp)
x.xfase	fase atual do ciclo do motor (vide .cpp)
x.xcwstep	sentido (1=true=horário, 0=false=anti-horário)

variáveis de controle dos beeps sonoros

x.bdur	duração dos beeps (ms) * 10
x.binter	pausa interbeeps(ms) * 10
x.bfreq	frequência dos beeps (Hz)
x.bnum	quantidade de beep+pausa a emitir

variáveis de controle das piscadas do Led

x.ldur duração do Led aceso (ms)*10
x.linter duração do Led apagado (ms)*10
x.lnum quantidade de piscadas

variável de controle temporal

x.xms	quantidade desejada de milisegundos * 10

Exemplos de utilização da biblioteca

No início do programa:
```
#include < X28BYJ48.h>
X28BYJ48 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:

//via chamada convencional:
```
x.runStep(2048, 3, true);
```

//via acesso direto as variáveis da biblioteca:
```
x.xtipostep=2; x.xvelstep=3; x.xcwstep=1; x.xsteps=2048;
```
//o motor começa a se movimentar imediatamente após a variável x.xsteps ser inicializada

//para saber se o motor de passo já chegou ao destino, fazer
//via chamada convencional:
```
if (x.where()>0) {ainda não chegou ao destino. Está em movimento...};
```

//via acesso direto as variáveis da biblioteca:
```
if (x.xsteps>0) {ainda não chegou ao destino. Está em movimento...};
```

//a qualquer momento o movimento do motor de passo nº0 pode ser interrompido
//via chamada convencional:
```
x.stopStep();
```

//via acesso direto as variáveis da biblioteca:
```
x.xsteps=0;
```

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

//via chamada convencional:
```
x.beep(10, 500, 2000, 250);
```

//via acesso direto as variáveis da biblioteca:
x.bdur=5000(*); x.binter=2500(*); x.bfreq=2000; x.bnum=10;
//os beeps começam a ser emitidos imediatamente após a variável x.bnum ser inicializada

//a qualquer momento a emissão dos beeps sonoros pode ser interrompida
//via chamada convencional:
x.stopBeep();

//via acesso direto as variáveis da biblioteca:
x.bnum=0;

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

//via chamada convencional:
x.led(50, 250, 100);

//via acesso direto as variáveis da biblioteca:
x.ldur=2500(); x.linter=1000(); x.lnum=50;
//o Led começa a piscar imediatamente após a variável x.lnum ser inicializada

//a qualquer momento as piscadas do Led podem ser interrompidas
//via chamada convencional:
x.stopLed();

//via acesso direto as variáveis da biblioteca:
x.lnum=0;

//contagem de 4 segundos, de forma assíncrona:

//via chamada convencional:
x.setms(4000);while (x.getms()>0){enquanto espera 4s, pode fazer coisas…}

//via acesso direto as variáveis da biblioteca:
x.xms=40000(*); while (x.xms>0){enquanto espera 4s, pode fazer coisas…}
//a variável x.xms começa a ser decrementada a cada um milisegundo imediatamente após ter sido inicializada

---

(*) - Esses valores são sempre multiplicados por 10 quando se tratar de acesso direto as variáveis da biblioteca

---