

## Controlo de Motor Passo a Passo com PSoC CY8C27643 e ULN2803

### Objectivo:

Controlar um motor passo a passo, nos dois sentidos relógio e contra-relógio.

### Material Necessário:

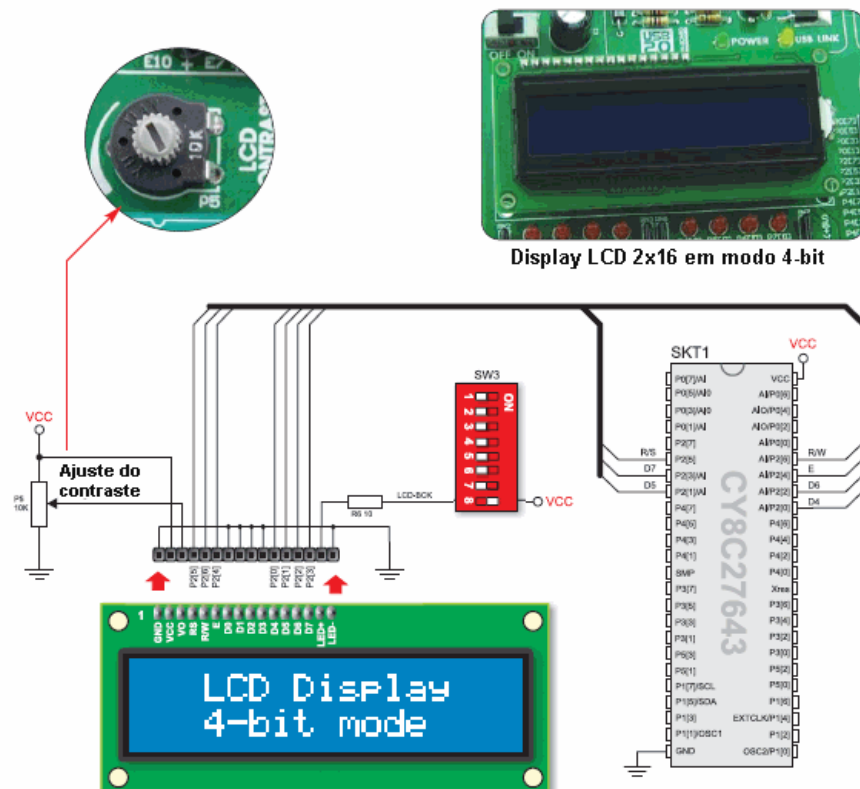
Placa de desenvolvimento EasyPSoC4 da Mikroelektronika, motor passo a passo (por exemplo, Parallax #27964) com CI ULN2803 e LCD 2x16 (este último dispositivo é incluído com a placa EasyPSoC4).

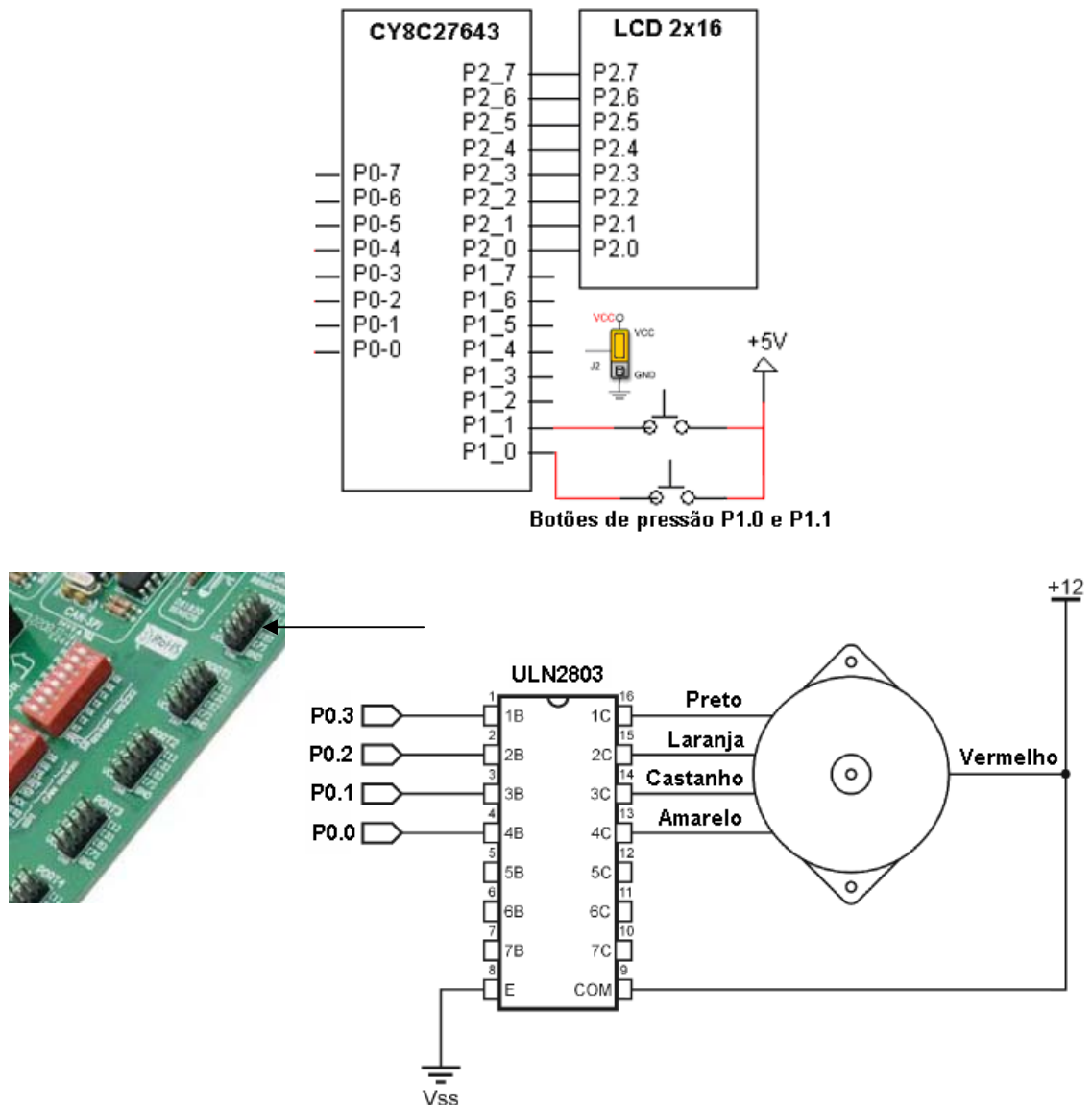
### Pré-Requisitos:

É necessário saber programar em linguagem "C" e ter noções básicas sobre PSoCs, nomeadamente sobre o integrado CY8C27643 da Cypress. O software gratuito "PSoC Designer" da Cypress deverá estar instalado no PC, com a opção do software de programação em "C" da Imagecraft, escolhido como padrão (que pode ser utilizado gratuitamente por um período de 45 dias).

### Montagem do Circuito:

Na placa EasyPSoC4 (que inclui como padrão o PSoC DIL CY8C27643 da Cypress), montar o display LCD 2x16 e o motor servo de rotação contínua. Assegurar que o display GLCD não esteja montado na placa EasyPSoC4, por este causar conflitos, quando montado em simultâneo com o LCD 2x16.





**Fig 2:** Diagramas dos circuitos com ligações do display LCD 2x16, dos botões de pressão, do CI ULN2803 e do motor de passo a passo (com indicação dos fios do motor e de como ligá-los ao ULN2803).

**NOTA:** A porta P0 encontra-se nos terminais no extremo direito da placa EasyPSoC4 indicado pela seta. O chip ULN2803 deverá ser alimentado externamente por uma fonte de 12V@1A.

## Introdução:

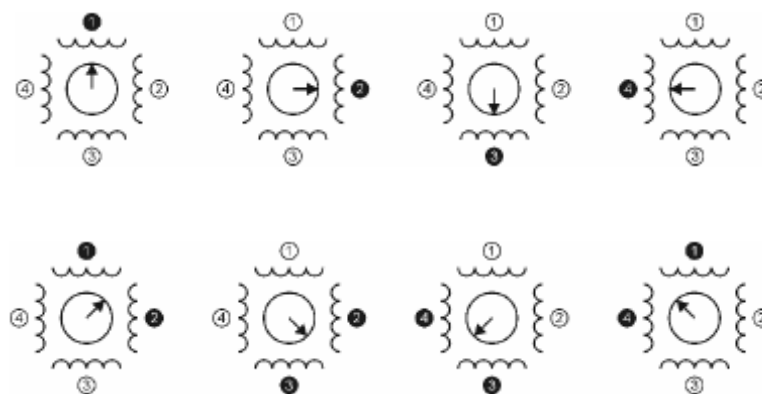
Um motor de passo a passo é um tipo de motor eléctrico, usado quando algo tem que ser posicionado com precisão, ou rodado a um ângulo exacto.

Neste tipo de motor, a rotação é controlada por uma série de campos electromagnéticos que são activados, e desactivados, eletronicamente.

Os motores de passo a passo não usam escovas, ou comutadores, e possuem um número fixo de pólos magnéticos, que determinam o número de passos por revolução. Os motores de passo mais comuns possuem de 3 a 72 passos/revolução, significando isso que o motor leva de 3 a 72 passos a completar uma revolução. No mercado existem controladores avançados de motores de passo a passo, que utilizam Modulação por Largura de Impulso, ou PWM, para realizarem micropassos, obtendo-se assim uma maior resolução de posição, e um funcionamento mais macio, em detrimento de outras características.

Os motores de passo a passo são classificados pelo binário que produzem. Para atingir todo o seu binário, as bobinas do motor devem receber toda a corrente típica durante cada passo. Os controladores devem possuir circuitos reguladores de corrente, para poderem fazer isso. O efeito da tensão (se houver) é praticamente sem utilidade.

O motor de passo a passo é controlado, aplicando uma sequência específica de passo; a velocidade rotacional é controlada pela temporização dos passos aplicados. Os diagramas seguintes mostram o efeito de sequenciamento de fase no movimento rotacional.



A figura mostra dois diferentes tipos de controlos. A primeira imagem de 4 sequências mostra controlo de passo completo, que resulta num binário fraco. A segunda imagem de 4 sequências mostra controlo de passo completo, que resulto num binário forte por factor de 1,4 em relação ao primeiro método.

**Fig. 3.** Sequência para controlo de passos

## Experiência:

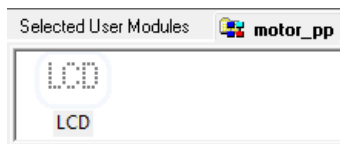
### Projecto de controlo de motor passo a passo, passo completo binário forte


#### Programação

**a)** Iniciar novo projecto no PSoC Designer, dando-lhe um nome, como por exemplo "motor\_pp". O software perguntará se se quer criar um novo directório, e a resposta deverá ser SIM premindo no botão respectivo. Na janela seguinte escolhe-se "C", premindo-se depois o botão Concluir.

**b)** No PSoC Designer escolhe-se LCD que se encontra dentro de Misc Digital do módulo User Module Selection, premindo duas vezes sobre o ícone do LCD. Renomear o default LCD\_1 para LCD, premindo com o botão direito do rato sobre o ícone no User Modules.

**c)** Selected User Modules deverá ficar com os ícones, como se pode vêr a seguir:



**d)** Premir a seguir o botão  (Interconnect View) para se poderem configurar as ligações no PSoC.


Não se mexe nos recursos globais.


Para Parâmetros de Módulos de Utilizador fazem-se as seguintes configurações no LCD:

User Module Parameters	Value
LCDPort	Port_2
BarGraph	Disable

**Fig. 4** Configurações do LCD

#### Estamos agora aptos para continuar:

**f)** A seguir, premir no botão  (Generate Application). O PSoC Designer gerará as aplicações.

**g)** Agora, premir no botão  (Application Editor), e dentro de pasta motor\_pp files e a sub-pasta Source files, abrir o ficheiro main.c e escrever o seguinte código, substituindo o croquit que é gerado automaticamente:

```
//-----
// Linhas principais em "C"
// Controlo de Motor Passo a Passo
// Autor: Tayeb Habib - Aliatron email: tayeb.habib@aliatron.pt
// Biblioteca de delay adaptado de código de PSoC Development System
// da Empresa mikroElektronika
// 18 de Agosto de 2009
//-----

#include <m8c.h> //Especificação de constantes e macros
#include "PSoCAPI.h" //Definições de API PSoC para todos
#include "delays.h" //os Módulos
#include "LCD.h"

//-----
// Linhas main do programa em C
//-----
void main()

{
    LCD_Start(); //Inicialização do LCD
    do {

        LCD_Position( 0, 0 ); // Linha 0, coluna 0
        LCD_PrCString( "P1.0 Relogio" ); // Mostrar menú
        LCD_Position( 1, 0 ); // Linha 1, coluna 0
        LCD_PrCString( "P1.1 Anti-Relog." ); // Mostrar menú

        if (PRT1DR & 0x01) { // Prefixo em hexadecimal
            //se botão P1.0 for premido motor roda no sentido relógio

            LCD_Position( 0, 0 ); // Linha 0, coluna 0
            LCD_PrCString( "P1.0 Sentido" ); // Botão P1.0 premido
            LCD_Position( 1, 0 ); // Linha 1, coluna 0
            LCD_PrCString( "Relogio " ); // Rotação sentido relógio

            PRT0DR = 0x0C; // On/off de Bits na Porta 0
            Delay_Nms( 1 ); // Atraso 0.01 segundos

            PRT0DR = 0x06; // On/off de Bits na Porta 0
            Delay_Nms( 1 ); // Atraso 0.01 segundos

            PRT0DR = 0x03; // On/off de Bits na Porta 0
            Delay_Nms( 1 ); // Atraso 0.01 segundos

            PRT0DR = 0x09; // On/off de Bits na Porta 0
            Delay_Nms( 1 ); // Atraso 0.01 segundos

        }

        if (PRT1DR & 0x02) {
            //se botão P1.1 for premido motor no sentido anti-relogio

            LCD_Position( 0, 0 ); // Linha 0, coluna 0
```

```

LCD_PrCString( "P1.1 Sentido " ); // Botão P1.1 premido
LCD_Position( 1, 0 ); // Linha 1, coluna 0
LCD_PrCString( "Anti-Relogio " ); // Rotação sentido anti-relógio

PRTODR = 0x09; // On/off de Bits na Porta 0
Delay_Nms( 1 ); // Atraso 0.01 segundos

PRTODR = 0x03; // On/off de Bits na Porta 0
Delay_Nms( 1 ); // Atraso 0.01 segundos

PRTODR = 0x06; // On/off de Bits na Porta 0
Delay_Nms( 1 ); // Atraso 0.01 segundos

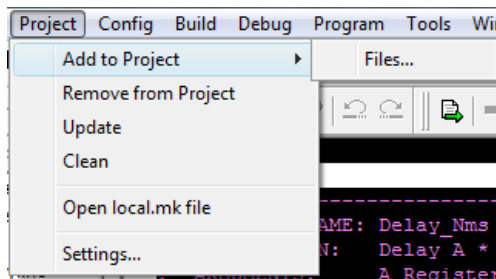
PRTODR = 0x0C; // On/off de Bits na Porta 0
Delay_Nms( 1 ); // Atraso 0.01 segundos
    }

}
while(1);
}
    
```

**h)** A seguir vamos adicionar as bibliotecas que geram o atraso entre cada passo. Colocar numa pasta os ficheiros que podem ser descarregados de:

[http://www.aliatron.pt/download/motor\\_pp\\_delays.rar](http://www.aliatron.pt/download/motor_pp_delays.rar)

Passo a passo, acrescentar todos os ficheiros usando o menú Project -> Add to Project -> Files, conforme se vê na figura seguinte:



Os ficheiros \*.asm serão acrescentados a Library Source e os \*.h ficarão na Library Header.


**NOTA:** Se não houver lugar a um atraso adequado o programa executar-se-á demasiado rápido para o motor passo a passo reagir.

**i)** Premir no botão  (Compile/Assemble). O resultado poderá ser visto na janela Build da parte inferior de PSoC Designer, e que será:

```

Compiling...
creating project.mk
./main.c

main.o - 0 error(s) 0 warning(s) 17:00:44
    
```

i) Finalmente premir no botão  (Build). O resultado poderá ser visto na janela Build da parte inferior de PSoC Designer, e que será:

```
Starting MAKE...
creating project.mk
lib/delays_12m.asm
lib/delays_1m5.asm
lib/delays_24m.asm
lib/delays_3m.asm
lib/delays_6m.asm
lib/delays_750k.asm
lib/lcd.asm
lib/psocconfig.asm
lib/psocconfigtbl.asm
./boot.asm
./main.c

Linking..
ROM 6% full. 1127 bytes used (does not include absolute areas).
RAM 2% full. 7 bytes used (does not include stack usage).

motor_pp - 0 error(s) 0 warning(s) 17:01:36
```

j) O ficheiro motor\_pp.hex será gerado, e que ficará dentro da pasta Output do directório criado no início da programação, o qual pode agora ser utilizado para programar o PSoC da placa EasyPSoC4. Quando o PSoC estiver programado com o hex, o display LCD 2x16 mostrará na primeira linha "P1.0 Relógio" e na segunda linha "P1.0 Anti-Relógio.". Premindo o botão P1.0, o motor passo a passo será accionado, e rodará no sentido de relógio. Premindo o botão P1.1, o motor girará no sentido anti-relógio de Reset do EasyPSoC4, e depois premindo o botão de pressão P0.4. O display indicará também o sentido de rotação. Se alterarmos o atraso no programa para 100, os passos serão de 1 segundo de cada vez. Assim, o atraso por cada passo determinará a velocidade de rotação do motor passo a passo.

## Conclusões:

Os PSoCs oferecem ciclos rápidos de desenvolvimento, traduzindo-se em completos sistemas embebidos, com componentes activos analógicos e digitais incluídos neles, sendo assim necessários poucos componentes externos. Estes dispositivos fabricados pela Cypress, têm-se provado ser extremamente extensíveis, sendo usados, por exemplo, em coisas tão simples como escovas de dentes da marca "Sonicare" e sapatilhas de alta competição da marca "Adidas". Uma característica de solução chamada "CapSense", da Cypress, controla no ecrã do Apple iPod, o scroll sensível ao toque.

Neste caso específico de controlo de motor passo a passo existe uma outra maneira de controlar usando Registos de Deslocamento (SR, isto é Shift Registers), emulados num componente existente dentro do PSoC. A

vantagem de usar SRs é que o processador do PSoC poderá executar outras tarefas uma vez inicializados os SRs (vêr bibliografia).

## **Bibliografia:**

Produtos PSoC na Aliatron:

[http://www.aliatron.com/loja/catalog/advanced\\_search\\_result.php?keyword=s=pSOC&search\\_in\\_description=1](http://www.aliatron.com/loja/catalog/advanced_search_result.php?keyword=s=pSOC&search_in_description=1)

Livro gratuito online "Architecture and Programming of PSoC Microcontrollers:

<http://www.easypsoc.com/book/>

Folheto técnico do PSoC CY9C27643:

<http://www.cypress.com/products/index.jsp?fid=24&rpn=CY8C27643>

PSoC – Arquitectura:

[www.engricardofranco.kit.net/scapitulo-1.pdf](http://www.engricardofranco.kit.net/scapitulo-1.pdf)

PSoC – Teoria e Aplicação em Linguagem C:

[www.engricardofranco.kit.net/scapitulo-2.pdf](http://www.engricardofranco.kit.net/scapitulo-2.pdf)

PSoC – Programação C

[www.engricardofranco.kit.net/scapitulo-3.pdf](http://www.engricardofranco.kit.net/scapitulo-3.pdf)

PSoC – Display LCD:

[http://www.engricardofranco.kit.net/4a-projeto-1\(lcd\).pdf](http://www.engricardofranco.kit.net/4a-projeto-1(lcd).pdf)

Motor Passo a Passo na Wikipédia:

[http://pt.wikipedia.org/wiki/Motor\\_de\\_passo](http://pt.wikipedia.org/wiki/Motor_de_passo)

Folheto Técnico do Motor Passo a Passo:

[https://dscl.lcsr.jhu.edu/wiki/images/9/98/Stepper\\_Motor\\_27964.pdf](https://dscl.lcsr.jhu.edu/wiki/images/9/98/Stepper_Motor_27964.pdf)

Folheto Técnico do CI ULN2803:

<http://www.rentron.com/Files/uln2803.pdf>

Método Alternativo de Controlo de Motor Passo a Passo com o PSoC:

<http://www.cypress.com/?docID=311>