ROBÔ ASPIRADOR

JOSÉ PEDRO FARIA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA E TELECOMUNICAÇÕES

2021

RESUMO

Palavras-chaves: -------

ÍNDICE

ÍNDICE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ACRÓNIMOS E TERMOS

- **OPCODE** – Código de Operação
- **RISC** – Reduced Instruction Set Computer, Computador com um Conjunto de Instruções Reduzido
- **CISC** – Complex Instruction Set Computer, Computador com um Conjunto de Instruções Complexo

LISTA DE SÍMBOLOS

- **V** – Volts
- **A** – Ampere
- **mAh** – mili Ampere hora
- **Ω** - Ohm
- **C** -Capacidade (Fards)
- **Hz** – Hertz
- **W** – Watts
- **µs** – microssegundos
- **f** – Frequência

INTRODUÇÃO

Este capítulo apresenta, ---

ENQUADRAMENTO DO PROJETO

OBJETIVOS

DESCRIÇÃO DO TRABALHO

ORGANIZAÇÃO DO RELATÓRIO

1.SISTEMA DE LOCOMOÇÃO

Este capítulo explica o sistema de locomoção do robô. Toca pontos como: material e a sua função, circuito, lógica teórica, programação e testes práticos.

FORMA DE LOCOMOÇÃO

A forma mais eficiente, e barata, de locomoção é através de motores, sejam eles DC, Passo-a-Passo ou servos. E independentemente do tipo de motor escolhido, existem diversas maneiras de os utilizar, com um, dois, três, quatro ou mais unidades, com roda livres ou sem elas, com diversos posicionamentos das rodas, entre outras. O modelo escolhido foi o de 2 motores, cada um encarregado de uma roda e com controlo individual; e uma roda livre para ter o terceiro ponto de apoio do robô.

TIPO DE MOTOR ESCOLHIDO

Os servo motores foram descartados logo de início por não serem os mais eficientes para a função, pois não estão preparados para muita rotação e são mais complexos de controlar. Entre os motores DC e os passo-a-passo a escolha foi mais renhida, pois ambos cumpririam a função proposta com muita eficiência. As tabelas a seguir apresenta as vantagens e desvantagens de cada um:

TABELA 1 - VANTAGENS/DESVANTAGENS MOTOR DC

MOTOR DC	
VANTAGENS	**DESVANTAGENS**
Simples Configuração	Ruidoso
Baixo Custo	Necessita de uma caixa de redução grande para ter um bom torque
	Consumo energético elevado (aproximadamente 400/500mA)

TABELA 2- VANTAGENS/DESVANTAGENS MOTOR PASSO-A-PASSO

MOTOR PASSO-A-PASSO	
VANTAGENS	**DESVANTAGENS**
Silencioso	Elevado Custo
Bom torque com uma caixa de redução pequena	Configuração Complexa
Baixo consumo energético (aproximadamente 200mA)	

Analisando o que cada motor tinha para oferecer, optei por utilizar motores Passo-a-Passo, principalmente pelos factos do consumo energético, que faz com que a bateria dure mais tempo; e do ruído, o robô aspirador para ser eficiente não pode incomodar as pessoas enquanto as poupa de uma tarefa, é contra intuitivo.

CARATERÍSTICAS DOS MOTORES PASSO-A-PASSO

Tensão de funcionamento

Para assegurar que o robô ia ter força para se locomover, optei por utilizar motores de 12V, pois têm mais torque do que os de 5V e os de 24V já não eram viáveis devido ao seu preço e às implicações que traria para a bateria, que seria maior e mais cara.

Polaridades

Existem diversos tipos de polaridades para os motores passo-a-passo. As mais utilizadas são as unipolares e as bipolares, e mesmo dentro da categoria dos bipolares existem vários tipos.

Devido à oferta do mercado as duas opções disponíveis eram a unipolar e o tipo mais simples dos bipolares. Cada um dos tipos têm as seguintes caraterísticas:

UNIPOLAR

Possui 5 fios de ligação, 4 para comando e 1 para alimentação. O circuito é o seguinte (datasheet):

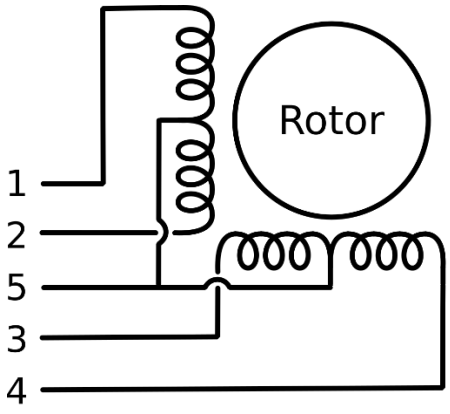


FIGURA 1- CIRCUITO MOTOR PASSO-A-PASSO UNIPOLAR

Tendo em conta este circuito, o fabricante também disponibiliza no datasheet a seguinte tabela de verdade (- negativo, + positivo, e vazio sem polaridade):

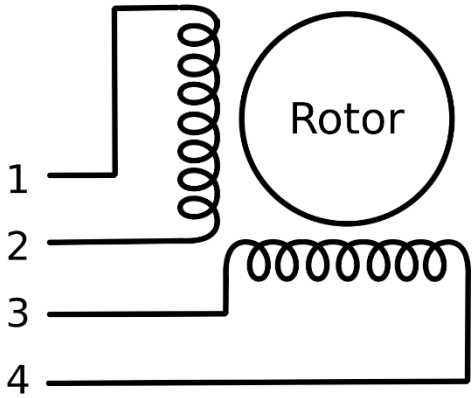
TABELA 3- TABELA DE VERDADE MOTOR PASSO-A-PASSO UNIPOLAR

	1	2	3	4	5
1º	-			-	+
2º	-		-		+
3º		-	-		+
4º		-		-	+

Se seguirmos a ordem 1º, 2º, 3º, 4º o motor roda no sentido horário. Se seguirmos a ordem 4º, 3º, 2º, 1º o motor roda no sentido anti-horário.

BIPOLAR

Possui 4 fios que fazem as funções de alimentação e comando simultaneamente. O circuito é o seguinte (datasheet):



O fabricante disponibiliza no datasheet a seguinte tabela de verdade (- negativo. + positivo):

TABELA 4- TABELA DE VERDADE MOTOR PASSO-A-PASSO BIPOLAR

	1	2	3	4
1º	-	+	+	-
2º	-	+	-	+
3º	+	-	-	+
4º	+	-	+	-

Se seguirmos a ordem 1º, 2º, 3º, 4º o motor roda no sentido horário. Se seguirmos a ordem 4º, 3º, 2º, 1º o motor roda no sentido anti-horário.

Motor Escolhido

O motor escolhido foi o motor passo-a-passo de 12V unipolar. A escolha do unipolar é por ser mais compacto e por facilitar o circuito de controlo.

CONTROLO DOS MOTORES

A forma de controlo dos motores foi inspirada num sistema criado pelo Eng. Wagner Rambo, do canal de Youtube WRKits. [1]

Como visto pela tabela de verdade, para controlar os motores precisamos de gerar sequências de bits (negativo = zero, positivo = um). Sendo assim ficamos com a seguinte sequência:

1 0 0 1

1 0 1 0

0 1 1 0

0 1 0 1

Contudo, se prestarmos atenção, os bits da primeira e segunda, e da terceira e quarta colunas são o inverso uma da outra. Sendo assim podemos aproveitar apenas o bit mais significativo (MSB) e o bit menos significativo (LSB) e inverte-los, aproveitando simultaneamente o bit e o seu oposto.

MSB			**LSB**
1	0	0	**1**
1	0	1	**0**
0	1	1	**0**
0	1	0	**1**

Para inverter os bits utilizamos um circuito inversor, o CD4069.

[1] https://www.youtube.com/c/canalwrkits

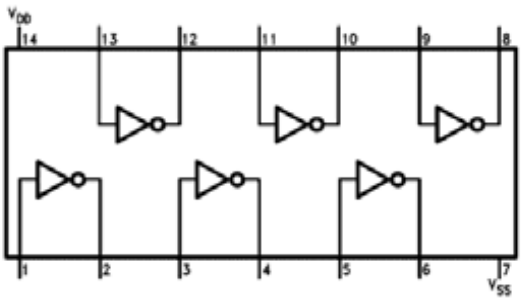


FIGURA 2- CD4069

Para gerar os bits utilizamos um microcontrolador de baixa capacidade de processamento, um PIC12F629 (um para cada motor). Esta opção deve-se ao facto de estarmos a trabalhar com um hardware “fraco” e termos de poupar pinos e ciclos no processador principal. O PIC12F629 recebe do PIC18F4520 (microcontrolador principal) apenas dois valores, um bit de direção (DIR), 0 roda para um lado e 1 roda para o lado oposto, e um sinal de clock, que é a frequência de rotação do motor, ou seja, a velocidade. Desta forma todos os ciclos de processamento necessários para a geração da sequência de bits de controlo do motor são gastos apenas pelo PIC12F629.

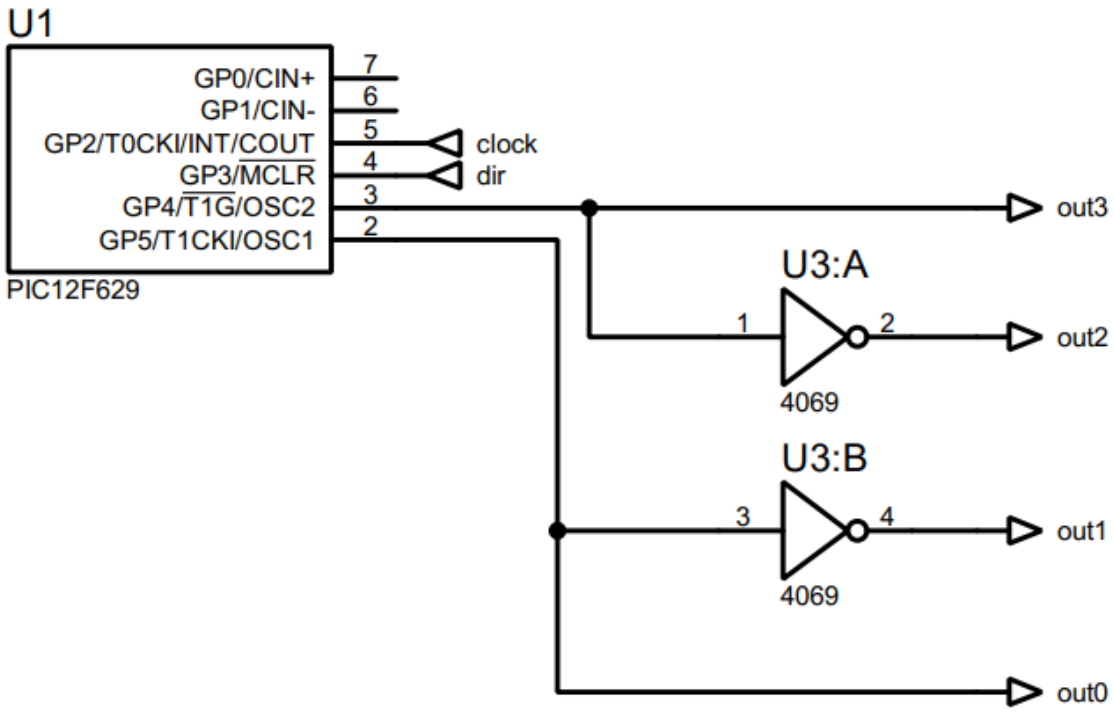


FIGURA 3- CIRCUITO PIC GERADOR DE BITS COM INVERSOR

Contudo os problemas ainda não estão todos resolvidos, pois as saídas do PIC fornecem apenas 5V e os motores funcionam a 12V. Para fazer o aumento da tensão podemos utilizar o ULN2003 (que é o drive utilizado pela maioria das placas de controlo dos motores passo-a-passo), ou, como temos 2 motores para controlar, utilizar um circuito que faça o trabalho de dois ULN2003, o ULN2804.

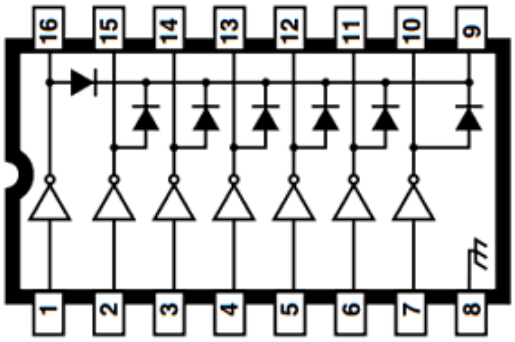


FIGURA 4- CIRCUITO ULN2003

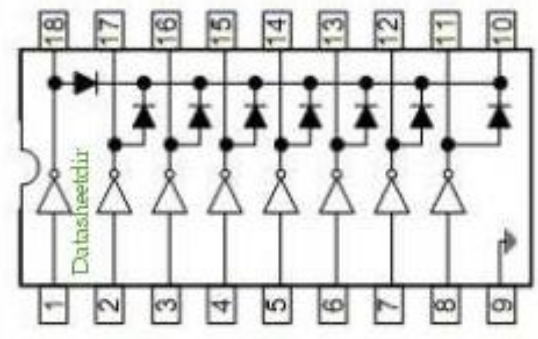


FIGURA 5- ULN2804

É preciso ter em atenção que tanto o ULN2003 e o ULN2804 invertem os bits, como tal é necessário ter atenção ou na geração dos bits ou apenas nas ligações do motor. Desta forma, adicionando o ULN2804, o circuito fica o seguinte:

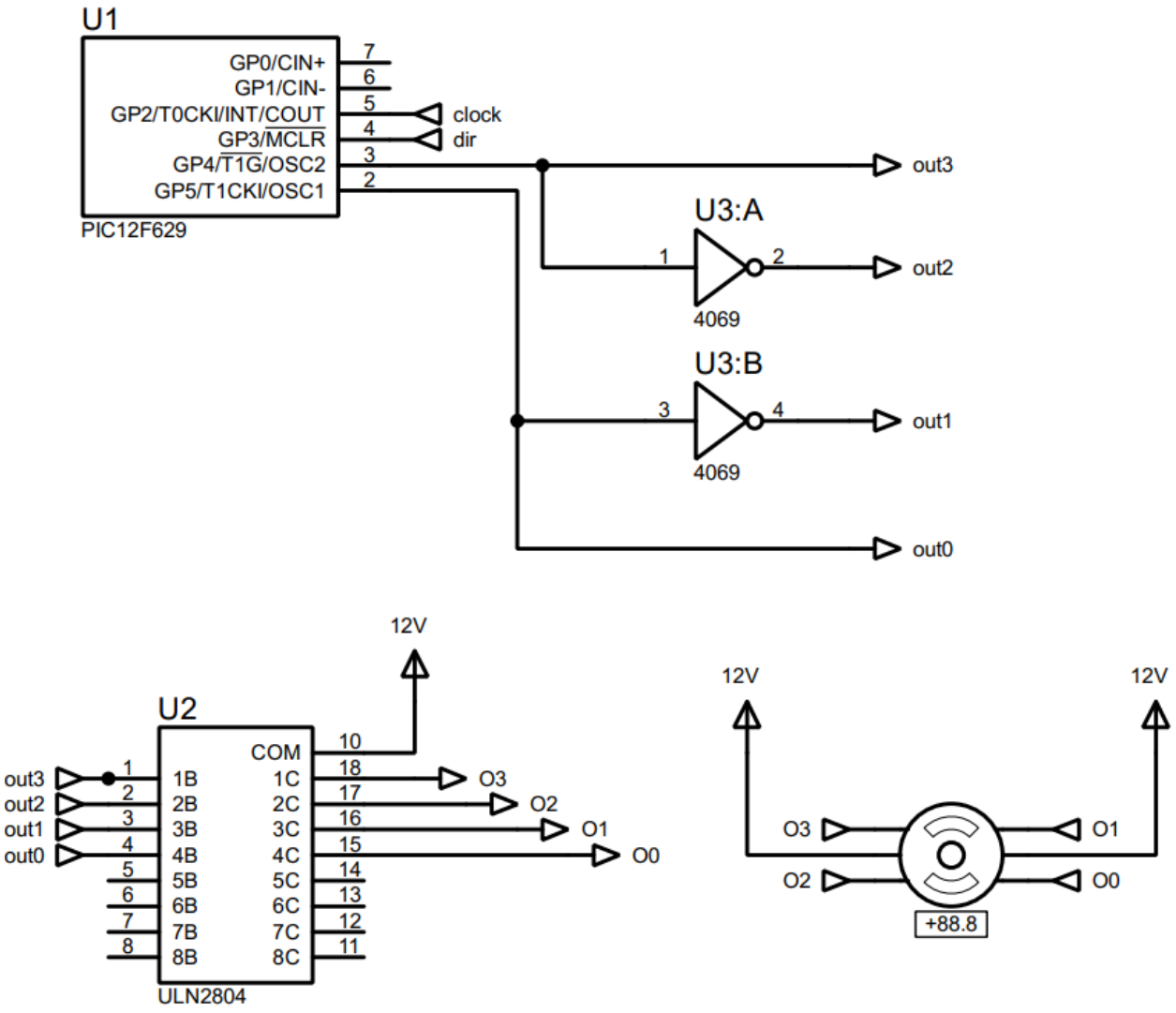


FIGURA 6- PIC GERADOR DE BITS COM INVERSORES E DRIVER

PROGRAMA

O programa foi desenvolvido em Assembly para otimizar os recursos do PIC12F629, que mesmo assim o consumo de memória ronda os 90%.

Programa

```
; ==================================================================
; --- Listagens e Inclusões de Arquivos
        list            p=12f629                ; Microcontrolador utilizado no projeto
        #include        <p12f629.inc>           ; inclui arquivo com registradores do PIC12F629

; ==================================================================
; --- FUSE Bits ---
; -> Oscilador Interno 4MHz sem clock externo;
; -> Sem WatchDog Timer;
; -> Power Up Timer Habilitado;
; -> Master Clear Desabilitado;
; -> Sem Brown Out;
; -> Sem proteções.
        __config        _INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_ON & _MCLRE_OFF & _BOREN_OFF & _CP_OFF & _CPD_OFF
; ==================================================================
; --- Paginação de Memória ---

        #define bank0   bcf     STATUS,RP0      ; cria mnemônico para seleção do banco 0 de memória
        #define bank1   bsf     STATUS,RP0      ; cria mnemônico para seleção do banco 1 de memória
; ==================================================================
; --- Definicação de Entradas e Saídas ---

        #define out3    GPIO,GP4                ; saída 3 (mais significativa)
        #define out0    GPIO,GP5                ; saída 0 (menos significativa)
        #define ctrl    GPIO,GP3                ; entrada de controle de direção do motor
; ==================================================================
; --- Registradores de Uso Geral ---

        cblock  H'20'                           ; início da memória disponível para o usuário
        W_TEMP                                  ; armazena valor temporário de W
        STATUS_TEMP                             ; armazena valor temporário de STATUS
        STEP                                    ; armazena valor do passo atual do motor
        endc                                    ; final da memória disponível para o usuário
; ==================================================================
; --- Vetor de RESET ---
        org             H'0000'                 ; origem no endereço 00h de memória
        goto    inicio                          ; desvia do vetor de interrupção

; ==================================================================
; --- Vetor de Interrupção ---
        org             H'0004'                 ; todas interrupções apontam para este endereço na memória de programa
; -- Guarda Contexto --
        movwf   W_TEMP                          ; guarda conteúdo de Work no registrador W_TEMP
        swapf   STATUS,W                        ; carrega conteúdo de STATUS no registrador Work com nibbles invertidos
        bank0                                   ; seleciona o banco 0 de memória
        movwf   STATUS_TEMP                     ; guarda conteúdo de STATUS no registrador STATUS_TEMP
```

```
; -- Teste das flags --
        btfsc   INTCON,INTF             ; ocorreu interrupção externa?
        goto    trata_INTE              ; sim, desvia para tratar interrupção Externa
        goto    exit_ISR                ; não, desvia para saída de interrupção
; -- Trata Interrupção Externa --
trata_INTE:
        bcf             INTCON,INTF     ; limpa flag
        call    process                 ; chama função de controle do motor
; -- Recupera Contexto (Saída de Interrupção ) --
exit_ISR:
        swapf   STATUS_TEMP,W           ; carrega conteúdo de STATUS_TEMP no registrador
Work
        movwf   STATUS                  ; recupera STATUS pré ISR
        swapf   W_TEMP,F                ; inverte nibbles do W_TEMP e armazena em W_TEMP
        SWAPF           W_TEMP,W        ; inverte novamente nibbles de W_TEMP armazenando
em Work (Recupera Work pré ISR)
        retfie                          ; retorna da interrupção
; ================================================================
; --- Configurações Iniciais ---
inicio:
        bank1                           ; seleciona banco1 de memória
        movlw           H'0F'           ; carrega literal 00001111b para o registrador Work
        movwf           TRISIO          ; configura GP4 e GP5 como saída (TRISIO = 0x0F)
        movlw           H'C0'           ; carrega literal 11000000b para o registrador Work
        movwf           OPTION_REG      ; (OPTION_REG = 0xC0)
                                        ; - desabilita Pull-Ups internos
                                        ; - configura interrupção externa por borda de subida
        bank0                           ; seleciona banco0 de memória
        movlw           H'07'           ; carrega literal 00000111b para o registrador Work
        movwf           CMCON           ; desabilita comparadores internos (CMCON = 0x07)
        movlw           H'90'           ; carrega literal 10010000b para o registrador Work
        movwf           INTCON          ; (INTCON = 0x90)
                                        ; - habilita interrupção global
                                        ; - habilita interrupção externa
        bcf                     out3    ; out3 inicia em low
        bcf                     out0    ; out0 inicia em low
        movlw           H'04'           ; carrega literal 00000100b para o registrador Work
        movwf           STEP            ; inicializa STEP
        goto            $               ; LOOP INFINITO, aguarda interrupção
; ================================================================
; --- Função para controle do Motor ---
process:
        rrf             STEP,F          ; shift right em STEP
        movf    STEP,W                  ; carrega conteúdo de STEP no registrador Work
        andlw   H'01'                   ; W = W and 00000001b
        btfsc   STATUS,Z                ; resultado da operação foi zero?
        goto    dir                     ; sim, desvia para steps
        movlw   H'10'                   ; não, move 00010000b para work
        movwf   STEP                    ; reinicia STEP
dir:
        btfss   ctrl                    ; entrada ctrl em high?
        goto    steps2                  ; não, desvia para steps2
steps1:
        movf    STEP,W                  ; carrega o conteúdo de STEP no registrador Work
        andlw   H'02'                   ; W = W and 00000010b
        btfss   STATUS,Z                ; resultado da operação foi zero?
        goto    step_1                  ; não, desvia para o step_1
        movf    STEP,W                  ; sim, carrega o conteúdo de STEP no registrador Work
        andlw   H'04'                   ; W = W and 00000100b
        btfss   STATUS,Z                ; resultado da operação foi zero?
        goto    step_2                  ; não, desvia para o step_2
        movf    STEP,W                  ; sim, carrega o conteúdo de STEP no registrador Work
```

```
        andlw   H'08'                   ; W = W and 00000100b
        btfss   STATUS,Z                ; resultado da operação foi zero?
        goto    step_3                  ; não, desvia para o step_3
        movf    STEP,W                  ; sim, carrega o conteúdo de STEP no registrador Work
        andlw   H'10'                   ; W = W and 00010000b
        btfss   STATUS,Z                ; resultado da operação foi zero?
        goto    step_4                  ; não, desvia para o step_4
        return                          ; retorna
steps2:
        movf    STEP,W                  ; carrega o conteúdo de STEP no registrador Work
        andlw   H'02'                   ; W = W and 00000010b
        btfss   STATUS,Z                ; resultado da operação foi zero?
        goto    step_4                  ; não, desvia para o step_4
        movf    STEP,W                  ; sim, carrega o conteúdo de STEP no registrador Work
        andlw   H'04'                   ; W = W and 00000100b
        btfss   STATUS,Z                ; resultado da operação foi zero?
        goto    step_3                  ; não, desvia para o step_3
        movf    STEP,W                  ; sim, carrega o conteúdo de STEP no registrador Work
        andlw   H'08'                   ; W = W and 00000100b
        btfss   STATUS,Z                ; resultado da operação foi zero?
        goto    step_2                  ; não, desvia para o step_2
        movf    STEP,W                  ; sim, carrega o conteúdo de STEP no registrador Work
        andlw   H'10'                   ; W = W e 00010000b
        btfss   STATUS,Z                ; resultado da operação foi zero?
        goto    step_1                  ; não, desvia para o step_1
        return                          ; retorna
step_1:
        bsf             out3            ; out3 em high
        bsf             out0            ; out0 em high
        return                          ; retorna

step_2:
        bsf             out3            ; out3 em high
        bcf             out0            ; out0 em low
        return                          ; retorna
step_3:
        bcf             out3            ; out3 em low
        bcf             out0            ; out0 em low
        return                          ; retorna
step_4:
        bcf             out3            ; out3 em low
        bsf             out0            ; out0 em high
        return                          ; retorna
; ==============================================================================
; --- Final ---
        end                             ; final do programa
```

TESTES PRÁTICOS

PRIMEIRO TESTE

No primeiro teste o objetivo foi confirmar a correta geração de bits, necessários para a rotação do motor. Para isso construí todo o circuito desenvolvido, com a exceção de na saída do ULN2804 não estarem ligados os motores nem 12V, mas sim 8 leds e uma tensão de 5V. Para ter uma tensão de saída de 5V basta alimentar o ULN2804 com 5V e não com os 12V anteriormente definidos.

Para simular o clock proveniente do microcontrolador principal construi um pequeno circuito oscilador, que gera um clock com aproximadamente 1Hz. É uma frequência extremamente baixa, contudo é o ideal para observar a mudança de estado dos leds e confirmar o funcionamento.

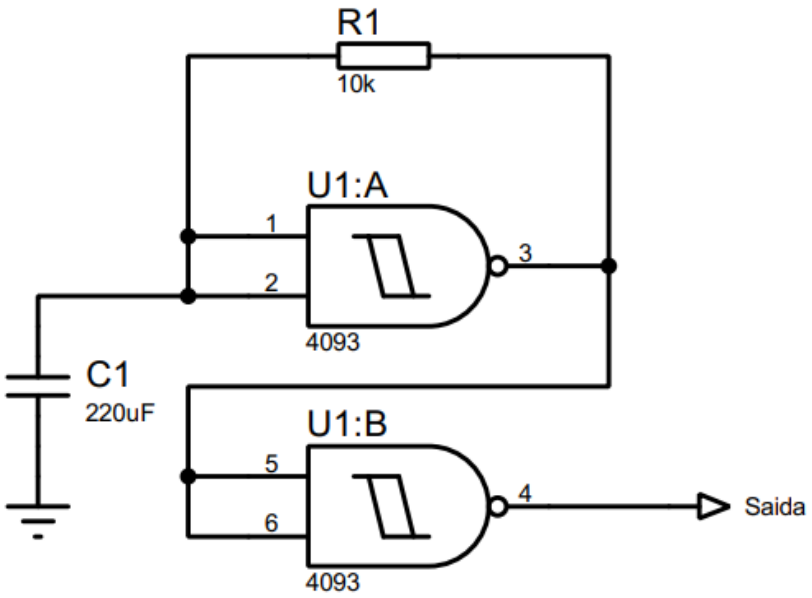


FIGURA 7- CIRCUITO OSCILADOR 1HZ

Juntando ambos os circuitos, oscilador e locomoção, ficamos com o resultado apresentado na figura a seguinte:

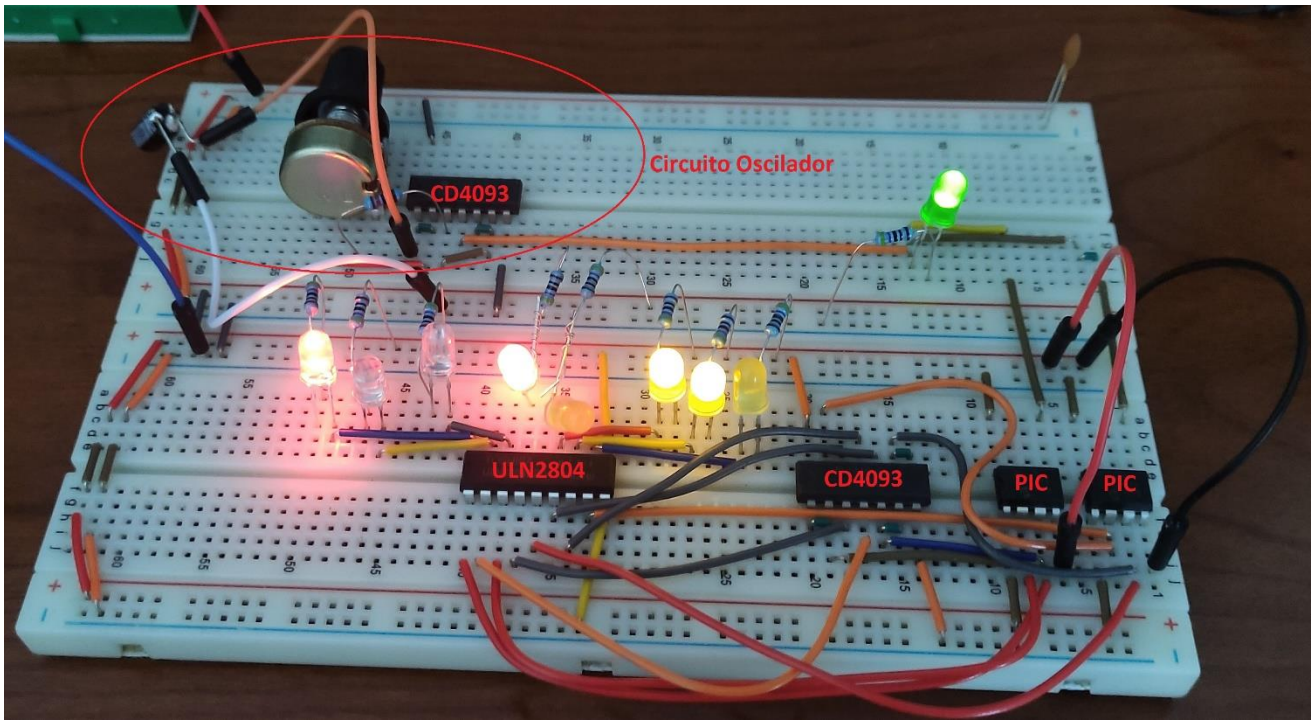


FIGURA 8- PRIMEIRO TESTE PRÁTICO SISTEMA DE LOCOMOÇÃO

No circuito acima, na parte superior tem o circuito oscilador, e na parte inferior o circuito do sistema de locomoção. Na saída do ULN2804 estão ligados 8 leds, os vermelhos representam um motor e os amarelos o outro.

A conclusão retirada deste teste prático foi que está tudo a funcionar como pretendido. Assim sendo estamos prontos para aumentar a tensão de saída e ligar os motores.

SEGUNDO TESTE

Para este teste ligamos os motores na saída do ULN2804. Como desta vez precisamos de dois valores de tensão diferentes, 5 e 12V, foi utilizada uma tensão base de 12V fornecida pela bateria e um conversor DC-DC para baixar a tensão para 5V sem perder corrente. Como forma de tentar compactar o circuito, e substituir o conversor DC-DC, foi testado um divisor de tensão utilizando resistências, como o a figura seguinte:

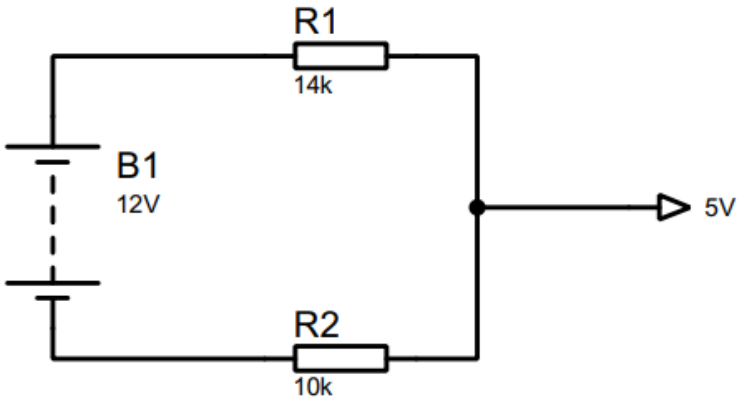


FIGURA 9- DIVISOR DE TENSÃO

O circuito funcionou como esperado, na saída tinha 5V, contudo a corrente era extremamente baixa. Resumindo, tive de continuar a utilizar o conversor DC-DC.

Para este teste continuamos com o mesmo oscilador de 1Hz, logo os motores rodam muito devagar, a frequência de funcionamento projetada (gerada pelo microcontrolador principal) é de 150/200Hz.

O circuito atualizado ficou então da seguinte forma:

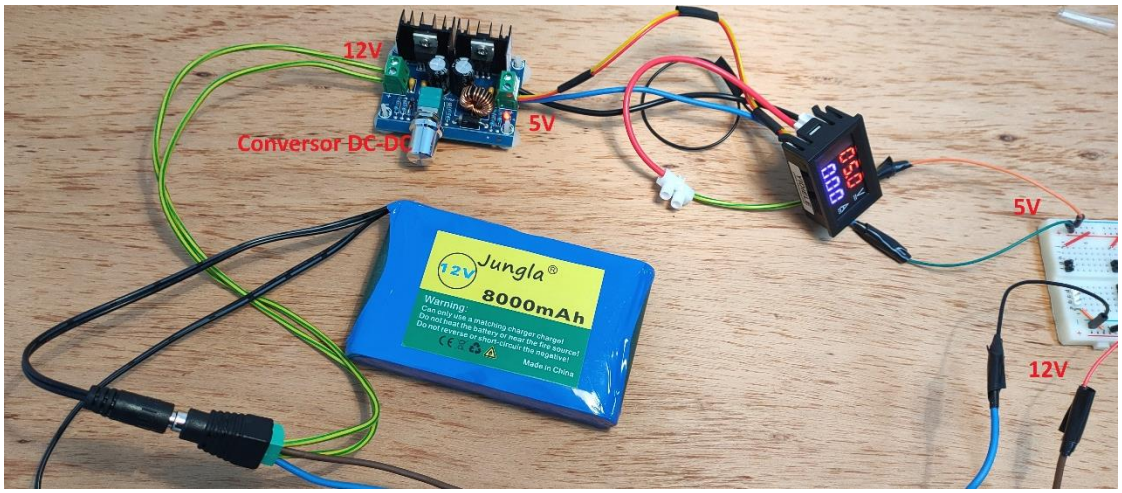


FIGURA 10 - CIRCUITO ALIMENTAÇÃO MOTORES PASSO-A-PASSO

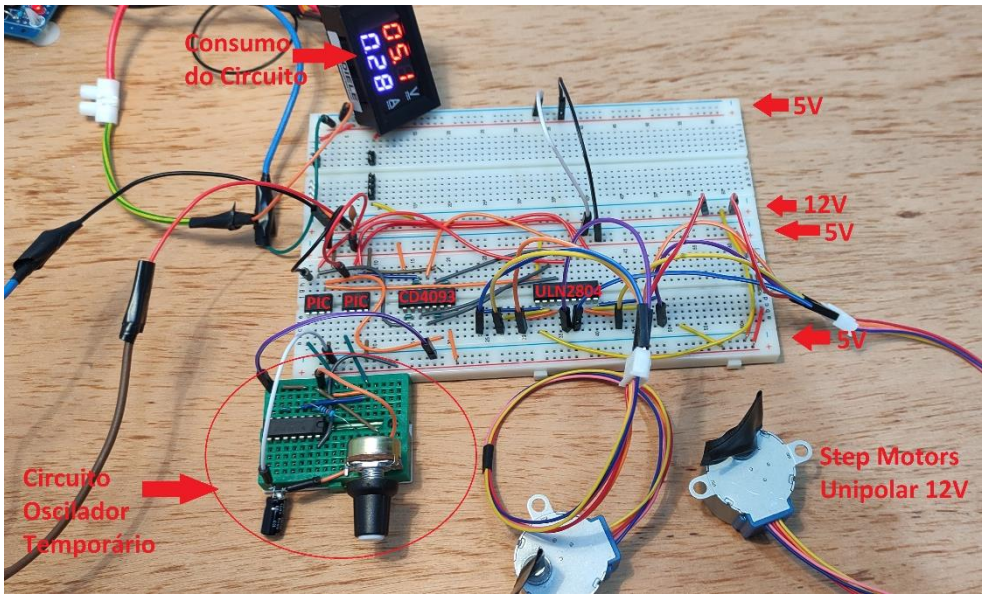


FIGURA 11 - CIRCUITO MOTORES PASSO-A-PASSO

TERCEIRO TESTE

Para este teste utilizamos o mesmo circuito do teste anterior, mas aumentamos o clock de funcionamento dos motores. Para isso, substituímos o circuito oscilador temporário de 1Hz, por uma frequência gerada pelo microcontrolador PIC18F4520, o nosso microncontrolador principal. Sendo assim ficamos com o seguinte circuito:

(AINDA EM DESENVOLVIMENTO)

2.SISTEMA DE DETEÇÃO DE OBSTÁCULOS

Este capítulo apresenta o sistema de deteção de obstáculos, o material utilizado, função do mesmo, circuito, lógica teórica, programação e testes práticos.

SENSORES DE DETEÇÃO DE OBSTÁCULOS

Para detetar obstáculos foram utilizados sensores de ultrassom, comuns em projetos de robótica, os HC-SR04.

FIGURA 12- HC-SR04 (SENSOR ULTRASSÓNICO)

FUNCIONAMENTO DOS SENSORES

O funcionamento dos sensores é bastante simples, a parte difícil de trabalhar com eles é o processamento dos resultados emitidos. Analisando o datasheet podemos ver que existem 3 fases:

1. Gerar um pulso de 10µs no pino TRIGGER;
2. O sensor gera 8 pulsos a 40kHz;
3. Lê o eco dos 8 pulsos (gerados no passo 2) e gera um sinal com tamanho proporcional à distância a que um ou mais objetos de encontram.

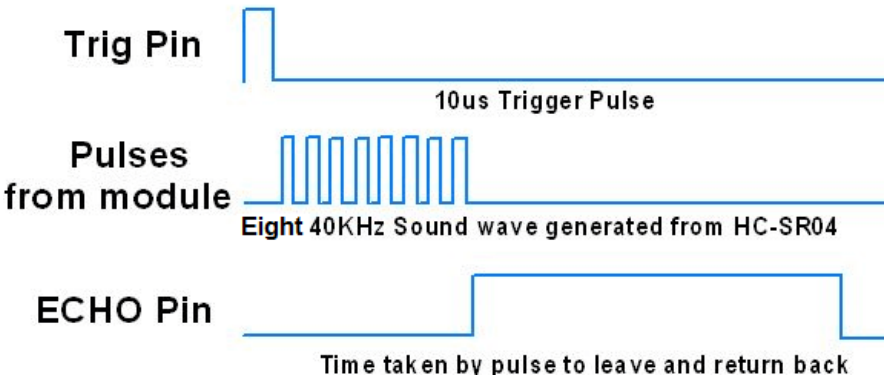


FIGURA 13 - PASSOS FUNCIONAMENTO HC-SR04

O ciclo de geração de leituras é repetido a cada 60ms, ou seja, aplicamos um sinal no TRIGGER a cada 60ms.

Para traduzir o valor do sinal gerado no passo 3 de µs para centímetros, temos duas possibilidades, uma com mais precisão que a outra. A primeira possibilidade, que é a mais precisa, utiliza a velocidade do som e tem a seguinte fórmula:

$$distância = \frac{HIGH\ Level\ \times velocidade\ 340\ m/s}{2}$$

A segunda possibilidade utiliza uma fórmula mais simplificada, e menos precisa, contudo funcional, que é a seguinte:

$$distância = \frac{largura\ do\ pulso\ (\mu s)}{58}$$

LEITURA E TRATAMENTO DOS SINAIS

Existem várias formas e dispositivos para ler e tratar os sinais dos sensores ultrassónicos. Se utilizarmos um Arduíno, por exemplo, podemos utilizar bibliotecas que nos ajudam com esse processo. Contudo, quando utilizamos microcontroladores PIC, não temos tanta facilidade, pois temos "manualmente" de gerar uma interrupção para gerar um pulso, utilizar o modo de captura do PIC, utilizando talvez um PWM que podíamos utilizar para outra função, ou utilizar ainda uma interrupção externa. É um processo que demanda muitos recursos e processamento apenas para a leitura de um sensor.

Para contornar o problema de gestão de recursos e processamento, vamos utilizar um microcontrolador PIC dedicado para gerar o pulso inicial no TRIGGER, ler o eco do sensor, interpretá-lo e entregar o resultado já processado ao microcontrolador principal, que a única coisa que terá de fazer é ser o valor em bits.

O modelo do PIC utilizado para esta função é o mesmo do utilizado para o controlo dos motores, o PIC12F629. Esta escolha deve-se ao facto de ser um microcontrolador simples, barato e pequeno. Teremos de utilizar um para cada sensor.

Uma vantagem de utilizar um microcontrolador dedicado para a leitura e tratamento dos sinais do sensor, é que podemos acrescentar ainda um algoritmo de suavização de leitura, evitando assim que movimentos bruscos e rápidos se reflitam nos movimentos do robô. O algoritmo será tratado num tópico posterior.

CIRCUITO

O circuito é extremamente simples de perceber, o sensor HC-SR04 tem 4 pinos (Vcc, GND, Echo e Trigger), o Vcc e o GND são para a alimentação, o trigger é onde emitimos um pulso (10µs) para iniciar o ciclo de funcionamento do sensor, o echo é por onde recebemos o pulso final correspondente à distância do obstáculo. O echo tem de estar ligado no pino 5 do PIC porque é uma interrupção externa, e o pino 5 é o único capaz de utilizar interrupções externas.

O PIC tem 3 pinos de saídas, que são os responsáveis por entregar os valores da distância ao PIC principal, dist1, dist2 e dist3.

E para terminar temos um jumper ligado no pino 4, denominado OPTION, que nos vai permitir selecionar o modo de funcionamento do sensor. A diferença no modo de funcionamento é a forma como os bits de saída são apresentados e o número de possibilidades de medida que cada modo consegue ler. Um dos modos consome menos que o outro, isto faz com que consigamos utilizar mais sensores ultrassónicos, ou utilizar o mesmo número, mas consumindo menos recursos. O funcionamento de casa modo está explicado posteriormente na seção dedicado ao programa.

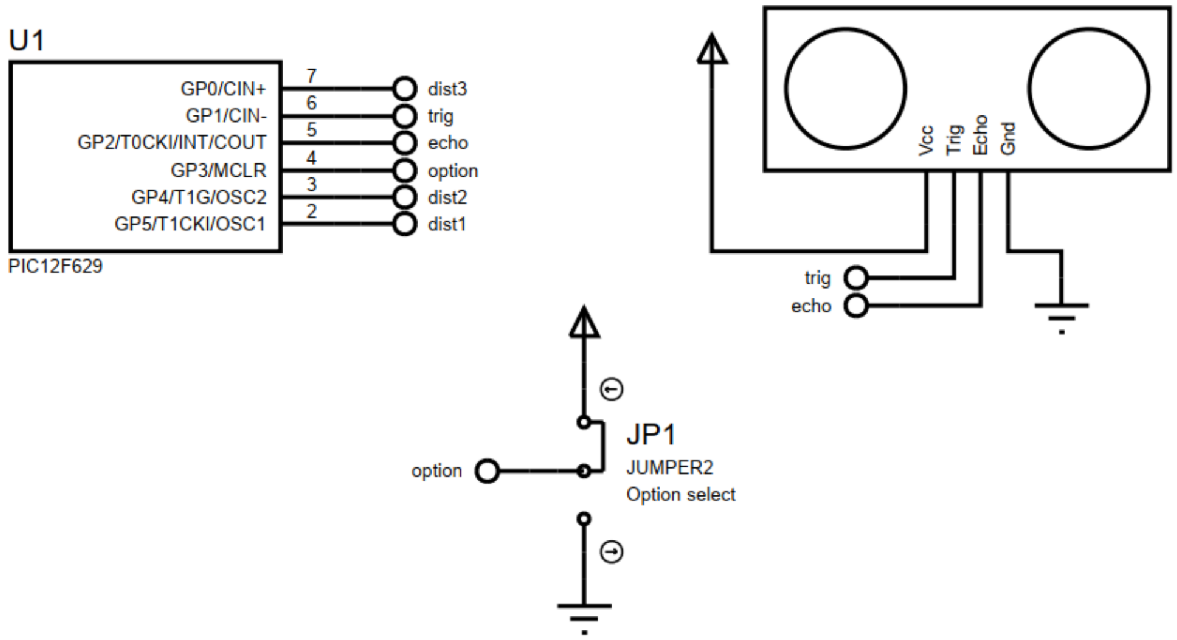


FIGURA 14- CIRCUITO DEDICADO SENSOR ULTRASSOM

PROGRAMA

O programa tem de satisfazer todas as restrições apresentadas anteriormente:

- Gerar pulso de 10µs no trigger a cada 60ms;
- Ler o pulso de echo, tratar e filtrar;
- 3 pinos de saídas binárias;
- 2 modos de configuração

As duas opções de funcionamento foram denominadas por A e B respetivamente. Na opção A (bit OPTION LOW), temos a seguinte configuração de saídas (os valores das distâncias podem ser alterados na programação):

	Dist1	**Dis2**	**Dist3**
>20cm	0	0	0
<20cm	1	0	0
<15cm	1	1	0
<10cm	1	1	1

Desta forma conseguimos com apenas 1 bit fazer o controlo de uma distância, por exemplo, sabemos que o bit dist3 só vai ser 1 quando a distância for menor que 10cm.

Na opção B (bit OPTION HIGH), não conseguimos fazer o controlo com apenas um bit, contudo temos um leque maior de distâncias. Temos as seguintes configurações de saídas (os valores das distâncias podem ser alterados na programação):

	Dist1	**Dist2**	**Dist3**
>=40cm	0	0	0
<40cm	0	0	1
<35cm	0	1	0
<30cm	0	1	1
<25cm	1	0	0
<20cm	1	0	1
<15cm	1	1	0
<10cm	1	1	1

Programa

```
// ===================================================================
// --- Mapeamento de Hardware ---
#define   trig        GPIO,1                  //saída para pulso de trigger
#define   dist1       GP5_bit                 //saída de indicação de distância 1
#define   dist2       GP4_bit                 //saída de indicação de distância 2
#define   dist3       GP0_bit                 //saída de indicação de distância 3
#define   option      GP3_bit                 //entrada para seleção de opções de funcionamento

// ===================================================================
// --- Macros e Constantes Auxiliares ---
#define   trig_pulse   asm bsf trig;     \
                       asm nop; asm nop; \
                       asm nop;          \
                       asm nop; asm nop; \
                       asm nop; asm nop; \
                       asm nop; asm nop; \
                       asm bcf trig;               //trig_pulse: gera pulso com duração
                                                   //de 10µs. Atraso criado em Assembly
#define   N           10                          //número de pontos da média móvel

 // ===================================================================
// --- Protótipo das Funções ---
void optionA(unsigned char cm1, unsigned char cm2, unsigned char cm3);  //opção de funcionamento A
void optionB();                                    //opção de funcionamento B
long moving_average(unsigned pulseIn);             //calcula a média móvel

// ===================================================================
// --- Variáveis Globais ---
unsigned      pulseL    = 0x00,                  //armazena a largura do pulso filtrada em µs
              pulseEcho = 0x00,                  //armazena a largura do pulso de echo em µs
              values[N];                         //vetor para média móvel

// ===================================================================
// --- Interrupções ---
void interrupt() {
  // +++ Interrupção Externa +++
  // Controla leitura em high do pulso de echo do sensor
  if(INTF_bit) {                                 //houve interrupção externa?
    INTF_bit = 0x00;                             //sim, limpa a flag
    if(!TMR1ON_bit) {                            //timer1 desligado?
      TMR1ON_bit = 0x01;                         //sim, liga timer1
      INTEDG_bit = 0x00;                         //configura int externa para borda de descida
    } //end if TMR1ON_bit
    Else {                                       //senão, timer1 ligado
      TMR1ON_bit = 0x00;                         //desliga timer1
      pulseEcho = (TMR1H << 8) + TMR1L;          //salva largura do pulso em µs
      TMR1H  = 0x00;                             //reinicia TMR1H
      TMR1L  = 0x00;                             //reinicia TMR1L
      INTEDG_bit = 0x01;                         //configura int externa para borda de subida
    } //end else TMR1ON_bit
  } //end INTF_bit

  // +++ Interrupção do Timer0 +++
  // Controla base de tempo para disparo do sensor
  // T0OVF = CM x PS x (256-TMR0) = 1E-6 x 256 x 235 = 60.16ms
  if(T0IF_bit) {                                 //houve interrupção do timer0?
    T0IF_bit = 0x00;                             //sim, limpa a flag
    TMR0     =   21;                             //reinicia timer0
   // -- base de tempo de aprox. 60ms --
    trig_pulse;                                  //gera pulso de trigger
```

```
  } //end T0IF_bit
} //end interrupt

// ============================================================================
// --- Função Principal ---
void main() {
  CMCON      = 0x07;                  //desabilita comparadores internos
  OPTION_REG = 0xC7;                  //interrupção externa por borda de subida
                                      //timer0 com prescaler 1:256
  INTCON     = 0xB0;                  //habilita interrupção global, do timer0 e externa
  TMR0       =   21;                  //inicializa Timer0 em 21
  TRISIO     = 0x0C;                  //configura I/Os
  GPIO       = 0x0C;                  //inicializa I/Os
  T1CON      = 0x00;                  //timer1 inicia desligado com prescaler 1:1

  while(1) {                          //loop infinito
    pulseL = moving_average(pulseEcho);//filtra pulso de echo e armazena em pulseL
    if(option) optionA(20,15,10);     //se option em HIGH, opção de funcionamento A
    else       optionB();             //se option em  LOW, opção de funcionamento B
  } //end while
} //end main

// ============================================================================
// --- Desenvolvimento das Funções ---
void optionA(unsigned char cm1, unsigned char cm2, unsigned char cm3) {
  if(pulseL < (cm1*58) && pulseL >= (cm2*58)) {  //distância menor que cm1 e maior igual a cm2?
    dist1 = 0x01;                     //sim, dist1 em HIGH
    dist2 = 0x00;                     //dist2 em  LOW
    dist3 = 0x00;                     //dist3 em  LOW
  } //end pulseL 1160

  else if(pulseL < (cm2*58) && pulseL >= (cm3*58)) {  //distância menor que cm2 e maior igual a cm2=3?
    dist1 = 0x01;                     //sim, dist1 em HIGH
    dist2 = 0x01;                     //dist2 em HIGH
    dist3 = 0x00;                     //dist3 em  LOW
  } //end pulseL 870

  else if(pulseL < (cm3*58)) {        //distância menor que cm3?
    dist1 = 0x01;                     //sim, dist1 em HIGH
    dist2 = 0x01;                     //dist2 em HIGH
    dist3 = 0x01;                     //dist3 em HIGH
  } //end pulseL 580

  else {                              //senão, distância superior a cm1
    dist1 = 0x00;                     //dist1 em  LOW
    dist2 = 0x00;                     //dist2 em  LOW
    dist3 = 0x00;                     //dist3 em  LOW
  } //end else
} //end optionA

void optionB() {
  if(pulseL < 2320 && pulseL >= 2030)  {  //distância menor que 40cm? (2320/58)
    dist1 = 0x00;                     //sim, dist1 em  LOW
    dist2 = 0x00;                     //dist2 em  LOW
    dist3 = 0x01;                     //dist3 em HIGH
  } //end pulseL 2320

  else if(pulseL < 2030 && pulseL >= 1740) {  //distância menor que 35cm? (2030/58)
    dist1 = 0x00;                     //sim, dist1 em  LOW
    dist2 = 0x01;                     //dist2 em HIGH
    dist3 = 0x00;                     //dist3 em  LOW
  } //end pulseL 2030
```

```
  else if(pulseL < 1740 && pulseL >= 1450) {          //distância menor que 30cm? (1740/58)
    dist1 = 0x00;                                     //sim, dist1 em  LOW
    dist2 = 0x01;                                     //dist2 em HIGH
    dist3 = 0x01;                                     //dist3 em HIGH
  } //end pulseL 1740

  else if(pulseL < 1450 && pulseL >= 1160) {          //distância menor que 25cm? (1450/58)
    dist1 = 0x01;                                     //sim, dist1 em HIGH
    dist2 = 0x00;                                     //dist2 em  LOW
    dist3 = 0x00;                                     //dist3 em  LOW
  } //end pulseL 1450

  else if(pulseL < 1160 && pulseL >= 870) {           //distância menor que 20cm? (1160/58)
    dist1 = 0x01;                                     //sim, dist1 em HIGH
    dist2 = 0x00;                                     //dist2 em  LOW
    dist3 = 0x01;                                     //dist3 em HIGH
  } //end pulseL 1160

  else if(pulseL < 870 && pulseL >= 580) {            //distância menor que 15cm? (870/58)
    dist1 = 0x01;                                     //sim, dist1 em HIGH
    dist2 = 0x01;                                     //dist2 em HIGH
    dist3 = 0x00;                                     //dist3 em  LOW
  } //end pulseL 870

  else if(pulseL < 580)  {                            //distância menor que 10cm? (580/58)
    dist1 = 0x01;                                     //sim, dist1 em HIGH
    dist2 = 0x01;                                     //dist2 em HIGH
    dist3 = 0x01;                                     //dist3 em HIGH
  } //end pulseL 580

  else  {                                             //senão, distância superior a 40cm
    dist1 = 0x00;                                     //dist1 em  LOW
    dist2 = 0x00;                                     //dist2 em  LOW
    dist3 = 0x00;                                     //dist3 em  LOW
  } //end else
} //end optionB

long moving_average(unsigned pulseIn) {       //retorna a média móvel de acordo com a resolução designada
  int i;                                              //variável para iterações
  long adder = 0;                                     //variável para somatório

  for(i = N; i > 0; i--)                      //desloca todo vetor descartando o elemento mais antigo
    values[i] = values[i-1];
    values[0] = pulseIn;                              //o primeiro elemento do vetor recebe o valor do pulso

  for(i = 0; i < N; i++)                              //faz a somatória
    adder = adder + values[i];
    return adder / N;                                 //retorna a média
} //end moving_average
// ================================================================================
// --- Final do Programa ---
```

ALGORITMO DE SUAVIZAÇÃO DE LEITURA

TESTES PRÁTICOS

---

3. SISTEMA DE DETEÇÃO DE COLISÕES

Este capítulo apresenta o sistema desenvolvido para detetar colisões, o sistema que entrará em funcionamento caso o de deteção de obstáculos falhe.

PRIMEIRO TÓPICO SISTEMA DE DETEÇÃO DE COLISÕES

---

4. SISTEMA DE CARREGAMENTO

---

5. SISTEMA DE ASPIRAÇÃO

---

6. SISTEMA DE MAPEAMENTO

7.INTEGRAÇÃO DE SISTEMAS

CONCLUSÃO

APÊNDICE A – MATERIAL E RECURSOS

BREADBOARD

A breadboard, ou protoboard, ou placa de testes, é uma placa que como o próprio nome diz, é para fazer testes de circuitos. Evita que se soldem componentes desnecessariamente, pois tem vários furos onde podemos facilmente encaixar e desencaixar os componentes. Os furos têm uma forma caraterística de condução de energia elétrica, pois no seu interior possuem várias pistas condutoras, organizadas como na representação a seguir:

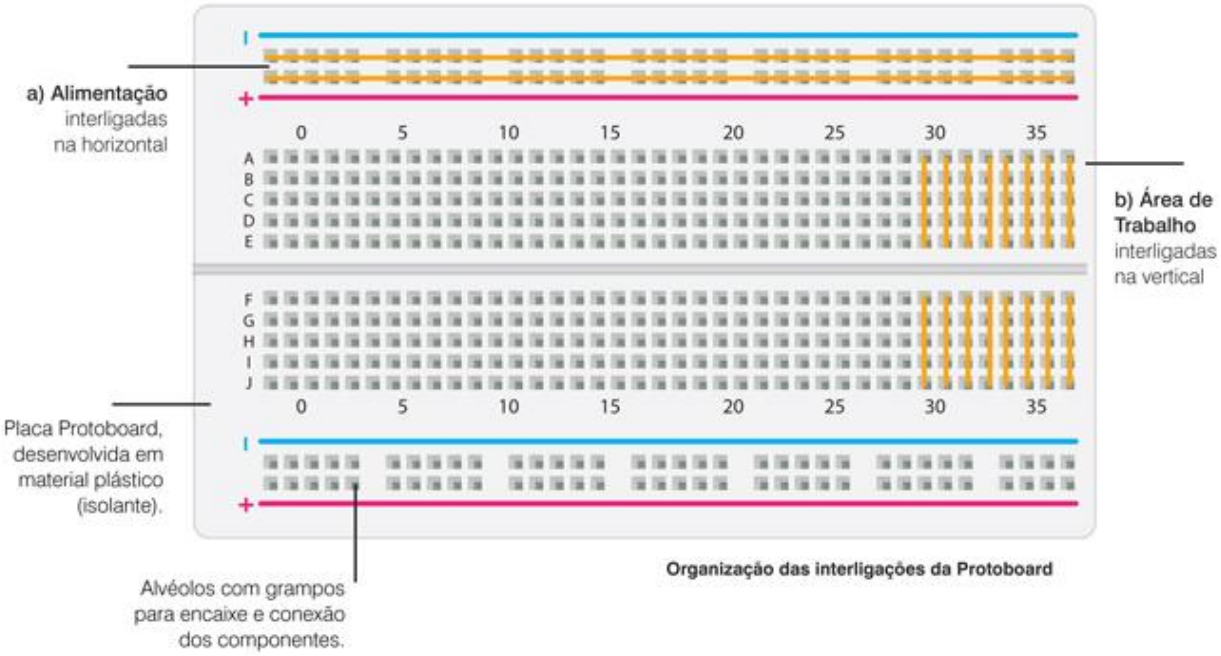


FIGURA 15- BREADBOARD, ESQUEMA DE CONDUÇÃO[2]

Na figura as linhas laranjas sinalizam a forma de condução. Este padrão repete-se para a maioria das breadboards.

[2] https://www.dreaminc.com.br/sala_de_aula/protoboard/

PINOS CONETORES MACHO

Pinos condutores que facilitam a realização de medições em circuitos, ou ligações de componentes periféricos, permitindo um contacto mais fácil e eficaz.

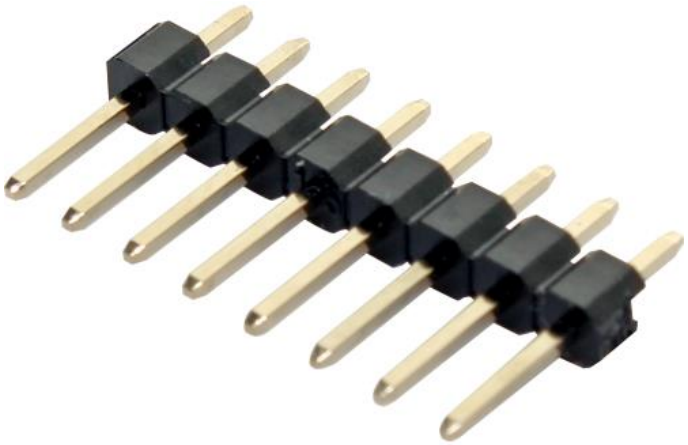


FIGURA 16- PINOS CONETORES MACHO

CONETOR DE CONTACTO MAGNÉTICO

Conetor de 3 pinos com contacto magnético. A sua função é ser a ficha de carregamento do robô aspirador, uma parte do conetor estará no robô e a outra parte na estação de carregamento, facilitando assim a ligação robô-base.

FIGURA 17- CONETOR DE CONTACTO MAGNÉTICO

CONDENSADORES

Os condensadores são componentes eletrónicos formados por duas placas condutoras, separadas por um material isolador (dielétrico). As duas placas condutoras, tendo um material isolador entre elas, manifestam a propriedade de armazenar cargas elétricas, e com isso também energia elétrica. Neste projeto os condensadores foram utilizados tanto para armazenas picos elétricos, como para filtrar altas frequências (ruídos).[3]

FIGURA 18- CONDENSADOR ELETROLÍTICO

FIGURA 19- CONDENSADOR CERÂMICO

[3] Eletrônica Básica, Newton C. Braga

RESISTÊNCIAS

As resistências reduzem, de maneira controlada, a intensidade da corrente, oferecendo-lhe uma oposição, ou então fazem cair a tensão num circuito para valores mais convenientes para uma determinada aplicação. As resistências possuem 3 especificações importante: resistência, tolerância e dissipação.[4]

O valor de cada resistência pode ser calculado com um ohmímetro ou através do código de cores, seguindo a lógica da tabela que se segue:

TABELA 5- ESQUEMA DE CORES RESISTÊNCIAS

Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	X1Ω	
Castanho	1	1	1	X10Ω	+/- 1%
Vermelho	2	2	2	X 100Ω	+/- 2%
Laranja	3	3	3	X 1KΩ	
Amarelo	4	4	4	X 10KΩ	
Verde	5	5	5	X 100KΩ	+/- 0.5%
Azul	6	6	6	X 1MΩ	+/- 25%
Violeta	7	7	7	X 10MΩ	+/- 0.1%
Cinza	8	8	8		+/- 0.05%
Branco	9	9	9		
Dourado				X 0.1Ω	+/- 5%
Prateado				X 0.01Ω	+/- 10%

FIGURA 20- RESISTÊNCIA

[4] Eletrônica Básica, Newton C. Braga

FIGURA 21- RESISTÊNCIA DE PRECISÃO

POTENCIÓMETROS

Os potenciómetros são resistências de valor variável. Foram utilizados vários tipos de potenciómetros, pois em algumas aplicações foi necessário um elevado nível de precisão, utilizando, portanto, trimpots, que não se conseguiria obter com os potenciómetros "convencionais" (mais comuns).

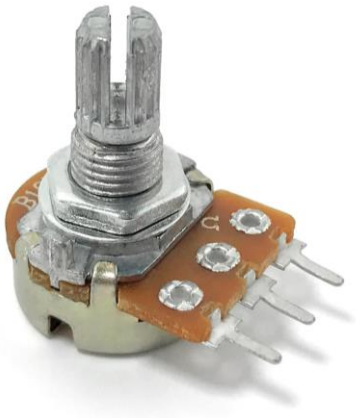


FIGURA 22- POTENCIÓMETRO "CONVENCIONAL"

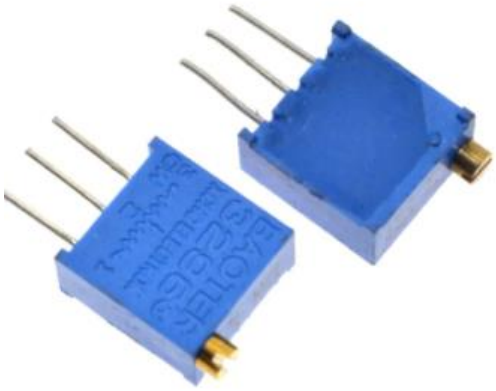


FIGURA 23- POTENCIÓMETRO TRIMPOT VERTICAL

FIGURA 24- POTENCIÓMETRO TRIMPOT LINEAR

DIODOS EMISSORES DE LUZ - LEDS

Os Leds no contexto deste projeto não são essenciais, no entanto facilitam muito o trabalho de um desenvolvedor para sinalizar bits a 0 ou a 1 num determinado circuito de teste, por exemplo.

FIGURA 25- DIODOS EMISSORES DE LUZ - LEDS

MOTORES PASSO-A-PASSO

É um motor com torque elevado, compacto e silencioso, no entanto necessita de um sistema relativamente complexo para o controlar. No projeto utilizamos os motores unipolares por uma questão de facilidade no circuito e nos seus componentes. Estes motores são de 12V para assegurar torque suficiente.

FIGURA 26- MOTOR PASSO-A-PASSO BIPOLAR 12V

ULN2804

O ULN2804 é uma matriz Darlington (transístores) de alta tensão e alta corrente, composta por 8 pares NPN Darlington. Possui saídas de coletor aberto com díodos de supressão para cargas indutivas e é ideal para interface entre circuitos lógicos de baixo nível e cargas de alta potência. Pode ser utilizado em relés, motores DC, lâmpadas de filamento, buffers de alta potência, e entre muitas outras aplicações, vamos utilizá-lo para os motores passo-a-passo, transformando os sinais de 5V do microcontrolador em 12V, que é a tensão de funcionamento dos motores.[5]

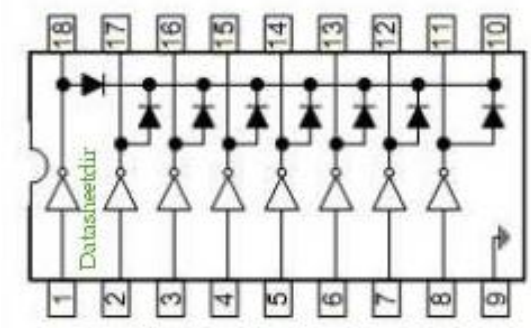


FIGURA 27- CIRCUITO ULN2904

[5] http://www.unisonic.com.tw/datasheet/ULN2804.pdf

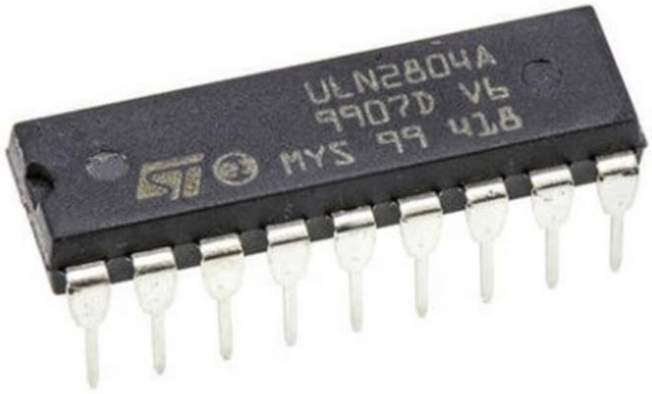


FIGURA 28- ULN2804

SENSORES ULTRASÓNICOS

Os sensores ultrassónicos HC-SR04 são os responsáveis pela deteção de obstáculos. Analisando o datasheet [6] percebemos o seu método de operação. Geramos um pulso de 10µs no pino de trigger, o sensor gera 8 pulsos a 40KHz, lê o eco desses pulsos e gera um sinal com um tamanho específico (quantidade de tempo em µs), quanto maior o sinal gerado, maior a distância a que se encontra o objeto. O ideal é repetir o ciclo anterior de 60 em 60ms.

Para converter o sinal de saída numa distância que consigamos perceber, existem dois métodos, um que utiliza a velocidade do som e dá valores mais certos, e um outro método que facilita as contas e dá valores aproximados muito próximos da realidade. Para este projeto, como não precisamos de uma precisão muito elevada e temos recursos extremamente limitados, vamos utilizar o método de conversão aproximado:

$$\frac{largura\ do\ pulso\ em\ \mu s}{58} = distância\ em\ centímetros$$

[6] https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

FIGURA 29- SENSOR ULTRASÓNICO - HC-SR04

MICROCONTROLADORES

Podemos definir um microcontrolador como um “pequeno” componente eletrónico, dotado de uma “inteligência” programável, utilizado no controlo de processos lógicos.

Os microcontroladores são o cérebro do Robô Aspirador ou de qualquer outro projeto que necessite de processamento. Neste projeto os microcontroladores escolhidos foram os microcontroladores PIC da Microchip. A escolha foi feita devido a uma paixão por esta família de microcontroladores. Foram utilizados dois modelos, ambos de 8 bits, que trabalham em equipa de forma a dividir o processamento, poupar recursos (nomeadamente pinos do microcontrolador principal) e maior precisão em algumas medidas recolhidas de sensores.

Os microcontroladores PIC têm uma arquitetura interna do tipo Harvard, diferente da maioria dos microcontroladores tradicionais que utilizam a de Von-Neumann. A diferença está na forma como os dados e o programa são processados. Na arquitetura de Von-Neumann existe apenas um barramento interno, geralmente de 8 bits, por onde passam as instruções e os dados. Na arquitetura de Harvard existem dois barramentos internos, sendo um de dados e outro de instruções. No caso dos microcontroladores PIC, o barramento de dados é sempre de 8 bits e o de instruções pode ser de 12, 14 ou 16 bits. Este tipo de arquitetura permite que, enquanto uma instrução é executada, outra seja procurada na memória, o que torna o processamento mais rápido.

Como o barramento de instruções é maior do que 8 bits, o OPCODE da instrução já inclui o dado e o local onde ela vai operar (quando necessário), o que significa que apenas uma posição de memória é utilizada por instrução, economizando assim muita memória de programa. Dentro da palavra do OPCODE, que pode ser de 12, 14 ou 16 bits, não sobra muito espaço para o código da instrução propriamente dito. Por isso, os PICs utilizam uma tecnologia chamada RISC, Reduced Instruction Set Computer (computador com set de instruções reduzido), que possui cerca de 35 instruções (o número varia de acordo com o microcontrolador), muito menos que os microcontroladores CISC (Complex Instruction Set Computer) que chegam a possuir mais de 100 instruções. Esta caraterística torna a aprendizagem dos PIC mais fácil, mas por outro lado, implica que tenham de ser criadas muitas funções, pois não possuem uma instrução direta.[7]

PIC12F629

O PIC12F629 é o que eu gosto de ganhar "mágico". Com recursos extremamente limitados, como os que podem ser vistos na tabela abaixo, ele é capaz de realizar tarefas muito eficientemente.

TABELA 6- CARATERÍSTICAS PIC12F629[8]

Memória de Programa	Memória de Dados		I/O	10-bit A/D	Comparadores	Timers 8/16-bit
Flash (words)	SRAM (bytes)	EEPROM (bytes)				
1024	64	128	6	-	1	1/1

Foi utilizado para controlar os motores com orientações do microcontrolador de processamento central, e para ler, interpretar e entregar os dados processados dos sensores ultrassónicos também ao microcontrolador de processamento central.

[7] Desbravando o PIC 16F628A, David José de Souza
[8] https://ww1.microchip.com/downloads/en/devicedoc/41190c.pdf

FIGURA 30 - PIC12F629

PIC18F4520

O PIC18F4520 é o primeiro PIC da linha 18F que tenho oportunidade de trabalhar, possui 40 pinos e muitos mais recursos do que o PIC12F629, como se pode ver na tabela abaixo. Como tal é o PIC selecionado para realizar o processamento principal do Robô Aspirador.

TABELA 7- CARATERÍSTICAS PIC18F4520[9]

Memória de Programa		Memória de Dados		I/O	10-bit A/D	CCP/ ECCP (PWM)	MSSP		EUSART	Comp	Timers 8/16-bits
Flash (bytes)	Instruções Single-Word	SRAM (bytes)	EEPROM (bytes)				SPI	Master I2C			
32K	16384	1536	256	36	13	1/1	Y	Y	1	2	1/3

FIGURA 31- PIC18F4520

[9] https://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf

CRISTAL OSCILADOR 20MHZ

O cristal oscilador é um componente que utiliza a ressonância de um cristal em vibração de um material piezoelétrico, para criar um sinal elétrico com uma frequência bastante precisa. No projeto, o cristal oscilador vai ser utilizado como oscilador externo do microcontrolador PIC18F4520.

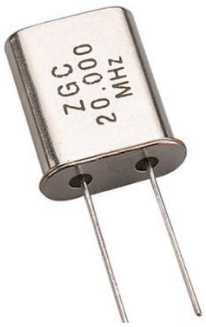


FIGURA 32- CRISTAL OSCILADOR 20MHZ

SOCKETS IC

Os sockets IC são encaixes universais para circuitos integrados. Não são componentes de extrema importância para o circuito, contudo ajudam a proteger o circuito integrado que estejamos a utilizar, principalmente no momento de soldadura, e permite uma substituição mais fácil do mesmo. No projeto todos os circuitos integrados, incluindo os microcontroladores PIC, estão encaixados num destes sockets.

FIGURA 33- SOCKETS IC

BATERIA 12V 8000MAH

Bateria de alimentação do circuito. É de 12V porque é o valor de tensão mais elevado exigido pelos componentes do circuito, por exemplo os motores passo-a-passo.

FIGURA 34- BATERIA LITIO 12V MAH

PLACA CARREGAMENTO/PROTEÇÃO DA BATERIA

É a placa responsável por receber a tensão de entrada no circuito do robô, alimentar a bateria e o circuito. Como extra protege ainda o circuito com isolamento galvânico.

FIGURA 35- PLACA CARREGAMENTO/PROTEÇÃO DA BATERIA

CONVERSOR DC-DC

O conversor DC-DC é necessário pois o projeto tem componentes com várias tensões de funcionamento, por exemplo os motores passo-a-passo, 12V, e os microcontroladores PIC, 5V. Como tal, a bateria é de 12V para fornecer a tensão original diretamente aos componentes de 12V e a quando passa pelo conversor fica a 5V para alimentar os restantes equipamentos.

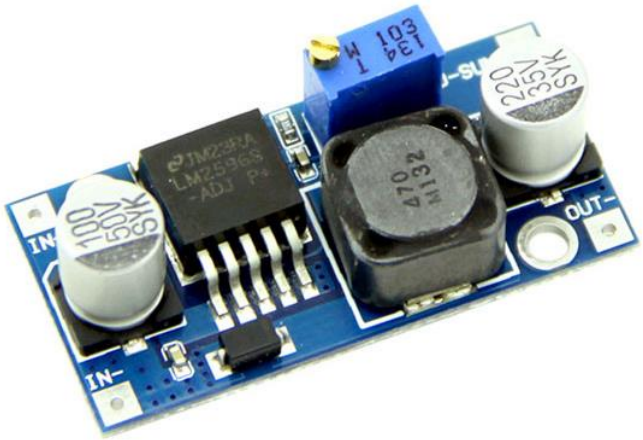


FIGURA 36- CONVERSOR DC-DC

FIOS JUMPER

Fios de ligações do circuito na breadboard.

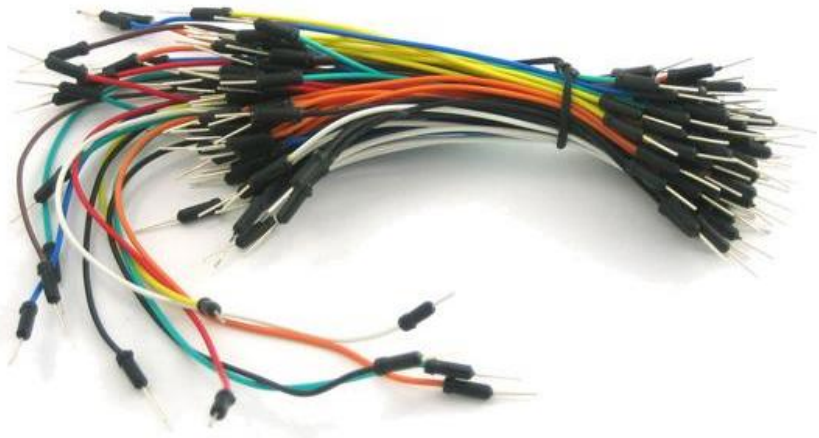


FIGURA 37- FIOS JUMPER

FIOS JUMPER COM GARRAS

Fios de ligações do circuito. A função principal é auxiliar equipamentos de medida.

FIGURA 38- FIOS JUMPER COM GARRAS

CD4069

Circuito integrado inversor. Contém no seu interior 6 inversores. Os inversores invertem a lógica dos bits, se o sinal entra a 1 sai a 0 e vice-versa. Na prática foi usado para obter o inverso de bits dos motores passo-a-passo, como explicado no capítulo dedicado aos motores; e foi também utilizado no circuito do gerador de sinais desenvolvido em casa.

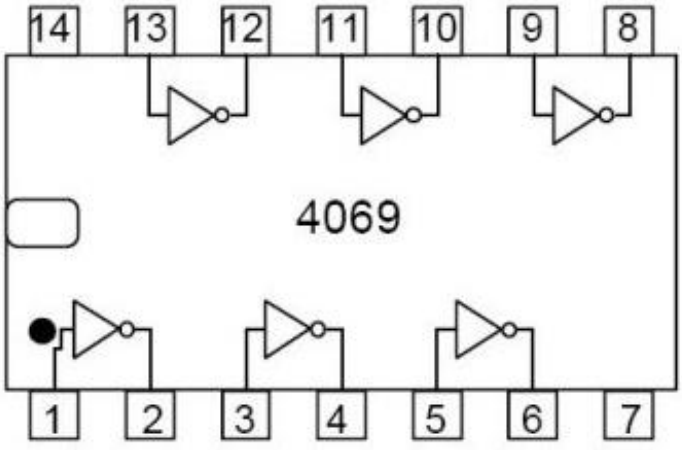


FIGURA 39- CIRCUITO CD4069

FIGURA 40- CD4069

ADAPTADOR TOMADA DE ALIMENTAÇÃO

Adaptadores para carregamento de bateria. São necessários, porque permitem uma extensão do cabo da bateria, facilitando assim o esquema de design do circuito.

FIGURA 41- ADAPTATOR TOMADA DE ALIMENTAÇÃO

FERRO/ESTAÇÃO DE SOLDA

A solda é de extrema importância, principalmente para oferecer mais robustez e melhor contacto ao circuito. Por vezes as breadboards (placas de teste) não são muito fiáveis, pois como são feitas para colocar e retirar componentes de maneira prática, ou seja, para fazer testes. O problema é resolvido soldando os componentes numa placa PCB.

FIGURA 42- FERRO/ESTAÇÃO DE SOLDA

MULTÍMETRO

Como em qualquer projeto de eletrónica, e este não é exceção, o multímetro é de extrema importância, principalmente nos modos: voltímetro, amperímetro, ohmímetro e condutividade. Uma vez que este projeto foi desenvolvido tanto na universidade como em casa, tive a necessidade de utilizar dois modelos diferentes, um para cada local de desenvolvimento. Os modelos em questão são as fotos apresentadas a seguir.

FIGURA 43- MULTIMETRO DE BANCADA - UNI-T UT803

FIGURA 44- MULTIMETRO WEIDMULLER 1037

OSCILOSCÓPIO

O osciloscópio analisa sinais elétricos no tempo e, como eu gosto de dizer, os osciloscópios são os olhos da eletrónica.

Para este projeto foram utilizados dois modelos, um Metrix OX 520B, disponibilizado no laboratório de eletrónica da ESTGL, e um modelo “caseiro” desenvolvido com Arduíno, pois com os vários confinamentos foi necessário trabalhar em casa e um equipamento novo deste género é muito dispendioso.

FIGURA 45- OSCILOSCÓPIO METRIX OX 520B

GERADOR DE SINAIS

O gerador de sinais, ou gerador de funções, é um aparelho que gera sinais elétricos de formas de onda, frequências e amplitudes variáveis. No contexto do projeto serviu para testes de componentes e/ou circuitos, como por exemplo no teste de controlo dos motores passo-a-passo, simulando o clock do PIC18F4520 e vendo o limite de frequência suportado pelos motores.

Assim como com o osciloscópio, utilizei dois modelos, um existente no laboratório de eletrónica da ESTGL e outro "caseiro", o qual fiz com orientação de um vídeo demonstrativo do canal de Youtube WRKits.[10] Este modelo de gerador de sinais "caseiro" possui transístores de Darlington na saída para amplificar os sinais.

FIGURA 46- GERADOR DE SINAIS TOPWARD 8110

[10] https://www.youtube.com/watch?v=cQZ-yPajBXg&t=685s&ab_channel=WRKits

PICKIT3

Gravador de microcontroladores PIC's.

FIGURA 47- PICKIT3

MPLAB IDE

O Mplab IDE (v.8.92) foi o IDE utilizado para programação em Assembly dos microcontroladores PIC.

FIGURA 48- MPLAB IDE

MIKROC PRO FOR PIC

O MikroC Pro for PIC é uma ferramenta de desenvolvimento para microcontroladores PIC. Foi o IDE utilizado para os microcontroladores PIC programados em C.

FIGURA 49- MIKROC PRO FOR PIC

PROTEUS 8 PROFESSIONAL

O Proteus é um software de criação e simulação de circuitos, foi nele que a grande maioria dos circuitos foram criados e testados.

FIGURA 50- LOGO PROTEUS 8

WEB/BIBLIOGRAFIA

WEBGRAFIA

- https://www.microchip.com/design-centers/8-bit/pic-mcus/device-selection
- https://pt.wikipedia.org/wiki/Microcontrolador_PIC
- https://ww1.microchip.com/downloads/en/devicedoc/41190c.pdf
- https://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf
- https://blog.eletrogate.com/wp-content/uploads/2018/07/Motor28BYJ48Kiatronics.pdf
- http://www.unisonic.com.tw/datasheet/ULN2804.pdf
- https://www.youtube.com/user/canalwrkits
- https://www.youtube.com/user/AllEletronicsGR
- https://pt.wikipedia.org/wiki/C%C3%B3digo_de_opera%C3%A7%C3%A3o
- https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
- https://pt.wikipedia.org/wiki/Oscilador_de_cristal
- https://pt.wikipedia.org/wiki/Arduino
- https://pt.wikipedia.org/wiki/Diodo_semicondutor

BIBLIOGRAFIA

- Desbravando o PIC 16F628A, David José de Souza
- Microcontroladores PIC 16F e 18F Teoria e Prática, Vidal Pereira
- Factfulness, Hans Rosling
- Eletrônica Básica, Newton C. Braga
- Eletrônica Analógica, Newton C. Braga
- Eletrônica Digital Vol1, Newton C. Braga
- Eletrônica Digital Vol2, Newton C. Braga