INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
CAMPUS SÃO JOSÉ DOS CAMPOS

André Papi Cardoso
Jonas Fucitalo dos Reis
Karl Max Tavares de Lira

Sensores e Atuadores em Sistemas VANT

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo ï Campus São José dos Campos, como requisito para obtenção do Título de Técnico em Automação Industrial sob orientação do Professor Aguinaldo Cardozo da Costa Filho e Co-orientação do Professor Dr. Edgard José Faria Guimarães e da Professora Dra. Neusa Maria Franco de Oliveira (ITA).

São José dos Campos
2014

BANCA EXAMINADORA

Trabalho de Conclusão de Curso (TCC) defendido e aprovado em _____ de _____ de 2014, pela banca examinadora constituída pelos professores:

Prof. Aguinaldo Cardozo da Costa Filho

.....

Orientador(a)

Prof.ª Neusa Maria Franco de Oliveira

.....

Co-orientador(a)

Prof. Edgard José de Faria Guimarães

.....

Co-orientador(a)

Aos nossos queridos pais e professores.

Com muito carinho,

Dedicamos

Agradecimentos

Primeiramente a Deus por ter nos dado saúde e força para superar todas as dificuldades, não somente nesses anos de curso, mas em todas nossas vidas.

Aos nossos pais, grandes incentivadores, por toda preocupação, apoio e incentivo à persistência.

Ao IFSP-Instituto Federal de Educação Ciência e Tecnologia de São Paulo campus São José dos Campos/PETROBRAS, seu corpo docente, direção, setor administrativo, serviço sócio pedagógico, por nos ter proporcionado, entre as inúmeras coisas que poderiam ser citadas, o conhecimento.

Ao Prof. Dr. Aguinaldo Cardozo da Costa Filho, nosso orientador, por dedicar seu valioso tempo à correção, revisão e suporte ao nosso trabalho, sempre se propondo a tirar nossas dúvidas e nos auxiliar no projeto.

À Prof.ª Vania Battestin por acreditar em nossa capacidade e contribuir para a elaboração desta monografia

À Prof.ª Dra. Neusa Maria Franco de Oliveira, coordenadora do nosso projeto de iniciação científica pelo comprometimento e dedicação.

Ao Prof. Dr. Edgard José de Faria Guimarães, pela paciência e pelo empenho em nos fazer aprender.

Ao ITA- Instituto Tecnológico de Aeronáutica, por nos ceder o laboratório e os materiais de eletrônica para os dias de reunião e elaboração deste projeto.

Ao CNPq por ter nos dado a possibilidade de desenvolver este projeto e por financiar nossa pesquisa.

Por fim, a todos os que acreditaram em nossa capacidade e de alguma forma contribuíram para o nosso trabalho, o nosso muito obrigado!

ñSe enxerguei mais longe
foi porque me apoiei
nos ombros de gigantes.ò

Isaac Newton

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SÍMBOLOS

VANT	Veículo Autônomo Não Tripulado
DC	Direct current (Corrente contínua)
CC	Corrente contínua
PWM	Pulse Width Modulation (Modulação por largura de pulso)
I/O	Input/Output (Entrada/Saída)
RAM	Random Access Memory (Memória de acesso aleatório)
CPU	Central Processing Unit (Unidade central de processamento)
KB	Kilo Bytes
MHz	Mega Hertz
V	Volts
s	Segundo
>	Maior que
<	Menor que
RADAR	Radio Detection and Ranging (Detecção e localização por rádio)
GND	Ground (Terra)

RESUMO

Esta monografia apresenta um sistema de controle de um veículo autônomo. As condições mínimas para a movimentação do mesmo são: a delimitação do trajeto pelos sensores ópticos com a identificação de obstáculos dentro da área de alcance de seu radar ultrassônico. A utilização de um radar traz um detalhe relevante, que é o fato do mesmo identificar apenas obstáculos em um ângulo de aproximadamente 180° em relação à horizontal, e esses obstáculos devem estar em determinado ângulo limite da onda de ultrassom emitida pelo radar. Caso seja ultrapassado o limite, a identificação do obstáculo será imprecisa, uma vez que a onda será emitida, porém não retornará diretamente no sensor. O não retorno direto da onda ultrassônica causa imprecisão, pois a distância é medida a partir do tempo que essa onda demora para retornar ao sensor. O veículo funciona da seguinte forma: O radar emite uma onda ultrassônica, essa onda se depara com o obstáculo e retorna para o radar que envia o sinal ao microcontrolador, onde é programado o que o veículo irá fazer a partir dos dados obtidos. A partir do tempo que a onda leva para retornar ao veículo, o código em *Assembly* converte esses dados para distância e provoca o deslocamento, ou não, para determinado sentido. Se não há obstáculos o veículo se movimenta para frente, e apenas se um obstáculo for identificado e estiver próximo o bastante ele mudará sua trajetória. Já quando se utiliza os sensores ópticos é necessário um trajeto específico tomando como referência uma linha clara ou escura.

Palavras chave: microcontrolador, robô, controle, autônomo, *Assembly*.

ABSTRACT

This monograph shows a control system of autonomous vehicle. The minimum conditions for its movement are: pathôs delimitation by the optical sensors with the identification of obstacles inside the range area of a ultrasonic radar. The radar's usage has a relevant detail, which is the fact of it only identify obstacles in a range of 180° angle relative to the horizontal, and this obstacles must stay in a certain angle limit of the wave emitted by the radar. If this limit is exceeded, the obstacleôs identification will be imprecise, once the wave will be emitted but does not return directly to the radar. This causes imprecision because the distance is measured by the time of this wave takes to return directly to the radar. The operation of the vehicle is given by the following sequence: The radar emits an ultrasonic signal, that returns and goes to the microcontroller, whereôs programmed what the robot will do from the data obtained. From the time the signal takes to returns to the vehicle, the Assembly code converts these data for distance and moves or not. If there are no obstacles the vehicle will move forward and only if the identified obstacle be close enough it will change its trajectory. Already when using optical sensors is necessary a specific path, either to follow reference.

Key words: microcontroller, robot, control, autonomous, *Assembly.*

1. INTRODUÇÃO

Os Veículos Aéreos Não Tripulados (VANTs) vêm sendo cada vez mais aplicados no cenário da aviação moderna, seja pelo fato de evitar riscos desnecessários à vidas humanas, seja pelo custo reduzido de execução de uma missão. Tarefas que antigamente eram realizadas por pilotos humanos, como por exemplo, reconhecimentos de território, atualmente podem ser executadas por uma aeronave não tripulada dotada de equipamentos que permitam o envio de imagens instantaneamente para uma central em terra. Esse tipo de aeronave tem sido empregado em vários tipos de aplicações: pulverização agrícola, inspeção de dutos de petróleo, gás e água, vigilância de perímetros rurais, vigilância de regiões de fronteiras, entre outras (GUIMARÃES e OLIVEIRA, 2013).

Será possível corrigir a trajetória do veículo autônomo, utilizando um microcontrolador para tomar as decisões cabíveis a cada caso, como desviar de obstáculos ou seguir em frente, a partir das informações recebidas dos sensores?

Neste projeto serão apresentados dispositivos eletrônicos (sensores e atuadores), bem como o microcontrolador específico com o objetivo de realizar a leitura dos sensores e comandar os atuadores.

Para que o veículo autônomo execute as tarefas desejadas, será utilizada a linguagem *Assembly* utilizada para escrever os códigos com as instruções e gravá-las no microcontrolador.

2. OBJETIVO

O objetivo do projeto é promover o movimento controlado de um veículo autônomo não tripulado, que ao encontrar obstáculos dentro da área de alcance de seus sensores corrigirá sua trajetória.

3. REVISÃO BIBLIOGRÁFICA

Nesta revisão bibliográfica serão abordados os aspectos característicos dos componentes utilizados na elaboração do veículo, como o microcontrolador, a ponte H, o servomotor, os sensores ópticos e o radar de ultrassom.

3.1. Microcontrolador 8052

ñUm microcontrolador é um componente eletrônico de constituição interna arrojada, o qual possui vários outros componentes interligados de forma a receber instruções, trabalhá-las e responder o processoò(SILVA et al., 2013).

> Microcontrolador 8052 é um dispositivo bastante utilizado, facilmente encontrado e de baixo custo. Trata-se de um sistema simples de microcomputador com um núcleo central de processamento (microprocessador), memórias e periféricos. Entre as memórias, o 8052 possui uma memória de programa, não volátil, de modo que um programa possa ser armazenado nela para execução pelo microprocessador interno. O processo de gravação do programa na memória de programa exige circuitos e conexões adicionais. Possui também uma memória do tipo RAM, volátil, para armazenar dados temporários como variáveis e vetores. Entre os periféricos, o 8052 possui interfaces paralelas, interfaces seriais (síncrona e assíncrona), temporizadores, controladores de interrupções, etc (GUIMARÃES,2014).

As características do modelo 89S52, retiradas do Datasheet do microcontrolador (ATMEL,2008), são:

- Família/Série do microcontrolador: AT89S52
- Tamanho da memória de programa: 8KB
- Tamanho da RAM: 256Bytes
- Velocidade da CPU: 24MHz
- Número de pinos I/Oôs: 32
- Frequência de Clock: 33MHz
- Tamanho da memória Flash: 8KB
- Tipo de memória: Flash

- Número de Bits: 8bit
- Número de temporizadores(timers): 3
- Tamanho da memória de programa: 8KB
- ñRangeòda tensão de alimentação: 4V à 5.5V

A Figura 1 a seguir mostra a configuração de pinagem do modelo AT89S52:

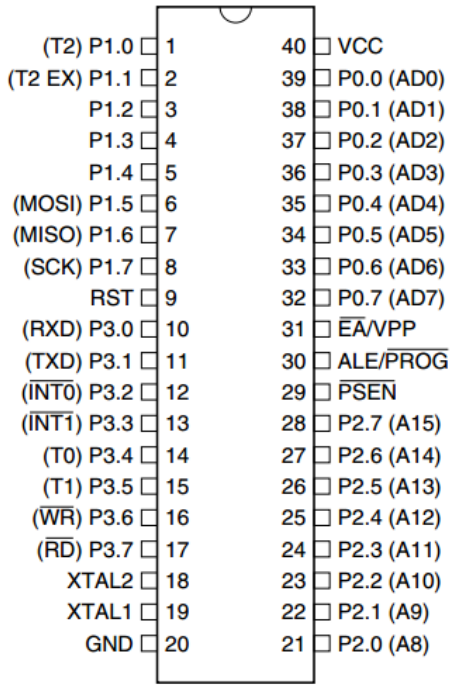

Figura 1: Configurações de pinagem do modelo AT 89S52.(Fonte: ATMEL, 2008)

3.2. Ponte H

De acordo com Guimarães (2014), a Ponte H é um circuito composto de 4 transistores de potência, operando no modo de chaveamento, de modo que possam ser ligados e desligados, e circuitos de controle que permitem a inversão de corrente em um dispositivo, que nesse projeto será um motor DC, e aplicação de um sinal PWM para controlar a potência desse motor.

Há circuitos integrados contendo praticamente todos os componentes de uma Ponte H. Um deles é o CI L298, que contém duas Pontes H independentes. A Figura 2 mostra a pinagem do CI L298 e a Figura 3, seu circuito interno.

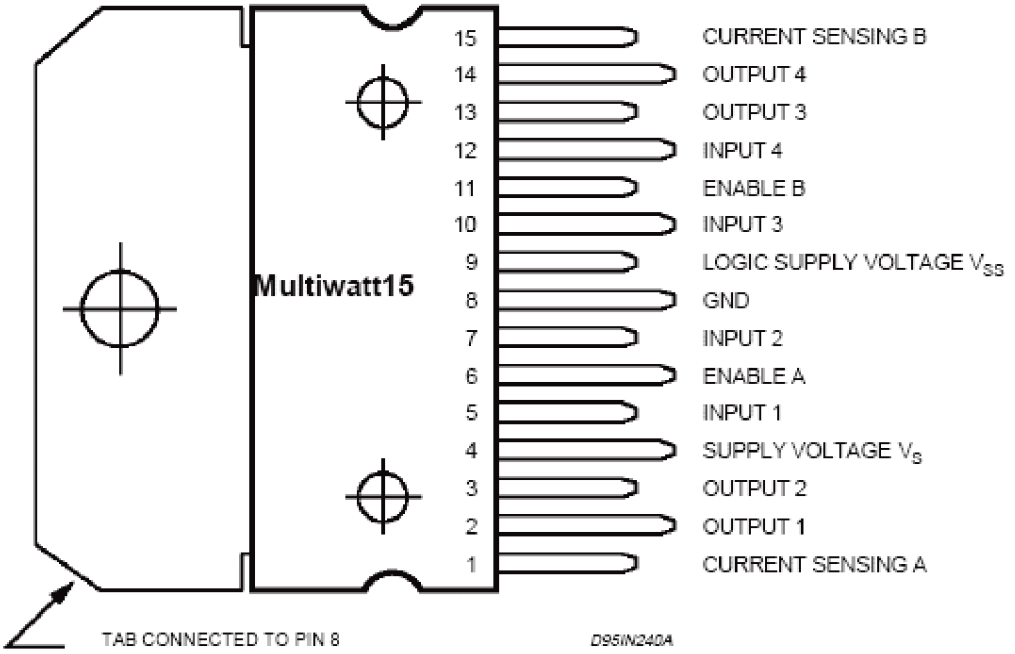

Figura 2: Pinagem do CI L298 (GUIMARÃES,2014)

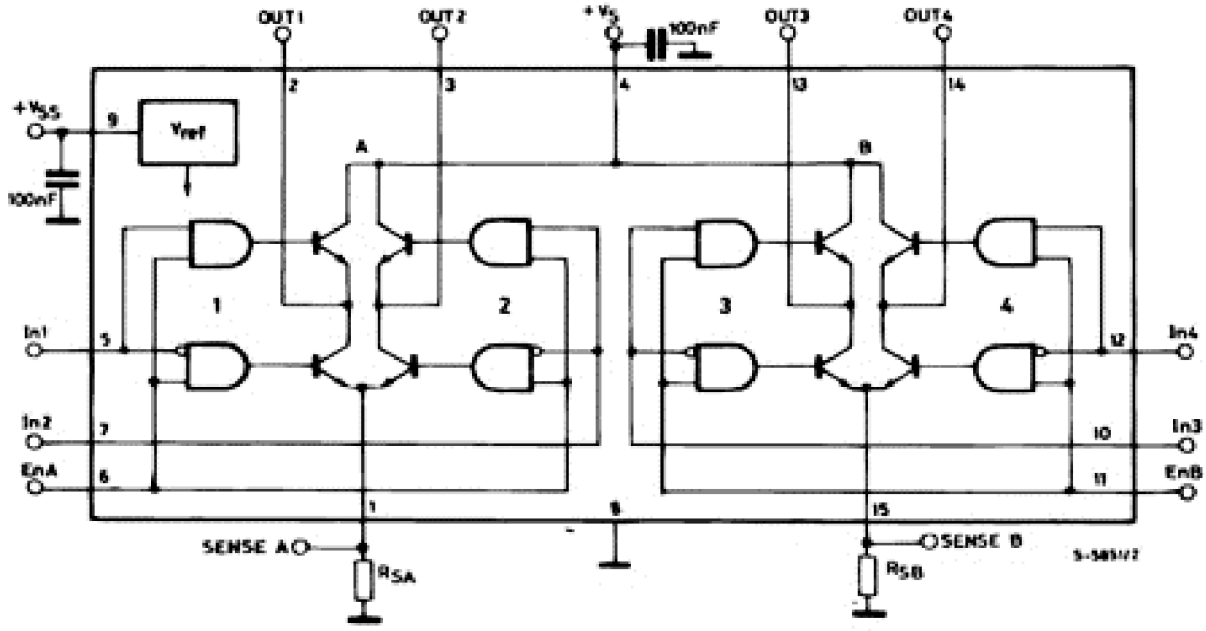

Figura 3: Circuito Interno do CI L298 (GUIMARÃES,2014)

Segundo Delai, Coelho e Mazzeo (2011) o objetivo da ponte H é controlar motores de corrente contínua CC, já que a fonte de energia é uma bateria de corrente contínua. A ponte H, cuja topologia está esquematizada na Figura 4, possibilita o motor girar em ambos os sentidos através do fechamento simultâneo das chaves Q1 e Q4 ou Q2 e Q3 e a velocidade do motor poderá ser controlada por

uma onda PWM aplicado à ponte. A largura do pulso controla a energia que é enviada e as inércias mecânicas mantém a rotação entre os pulsos, de forma que apenas o nível médio atue e a velocidade de rotação seja praticamente constante.

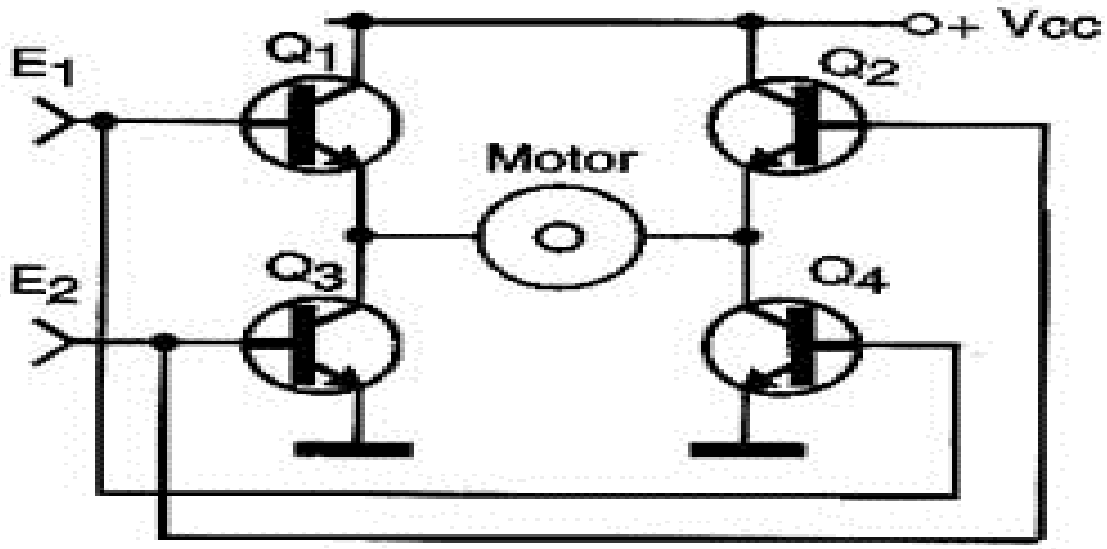

Figura 4: A ponte H básica (Site Newton C. Braga).

3.3. Servomotor

De acordo com Zucatelli e Oliveira (2007), sempre que se utiliza alguma forma de energia é necessário aplicar controle sobre a mesma. Os servomotores são equipamentos capazes de gerar movimento angular controlado utilizando energia elétrica. São muito utilizados na indústria em aplicações que necessitam de precisão, velocidade controlada assim como o torque e alto grau de repetitividade.

Neste projeto utilizou-se o servomotor para movimentar o radar de ultrassom em sua varredura de 180º.

3.4. Realimentação com Sensores Ópticos

Segundo Guimarães (2014), uma maneira simples para controlar o percurso do veículo, é fazê-lo seguir uma referência no chão. Para isso surge a necessidade de um conjunto de sensores ópticos que consigam identificar a referência a ser seguida.

Um sensor óptico que atende a essa finalidade é o GP452 da Sharp. É

composto de um diodo emissor de luz e um fototransistor, dispostos de modo que a luz emitida pelo diodo pode ser refletida em um anteparo e alcance o fototransistor. Se o anteparo for suficientemente claro (e com o devido circuito de polarização), o fototransistor atinge sua região de saturação. Caso contrário, o fototransistor fica na região de corte. A Figura 5 mostra esquema interno do sensor óptico e os circuitos de polarização do diodo e do transistor.

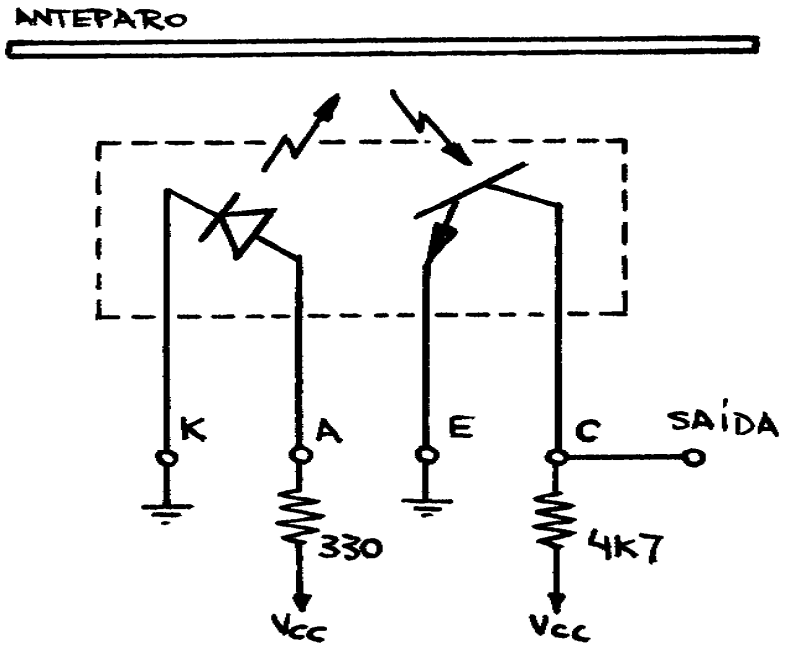

Figura 5: Esquema interno do sensor óptico.(GUIMARÃES,2014)

Segundo Guimarães (2014), com o circuito de polarização mostrado, quando o anteparo for bem claro, o transistor satura e a tensão no coletor é de 1 V. Quando o anteparo for mais escuro, o transistor conduz menos e a tensão no coletor é de 3,5 V. Esses valores são entendidos como nível lógico 0 (0) e 1 (1), respectivamente, pelo *port* de entrada do microcontrolador 89S52.

Com isso o veículo irá corrigir sua trajetória caso seja necessário ao se deslocar para fora da referência.

3.5. Radar ultrassônico HC-SR04

> O Radar com Ultrassom é um módulo que permite a medição de distância (entre alguns centímetros e alguns metros), usando ondas sonoras não audíveis na frequência de 40KHz. O módulo é composto de um transmissor e um receptor de ultrassom, mais circuito de controle. A partir de um pulso de início (TRIG), o módulo emite um som durante um intervalo curto de tempo e espera até que esse som reflita num obstáculo e volte, sendo captado pelo receptor. O tempo entre a transmissão e a recepção é fornecido por um terminal de saída (ECHO). Com base nesse tempo de ida e volta da onda sonora, e sabendo-se a velocidade do som no ar, é possível calcular a distância (GUIMARÃES,2014).

De acordo com BRAGA (2012) o princípio de operação desses sensores é exatamente o mesmo do sonar usado pelo morcego para detectar objetos e presas em seu voo cego.

O comprimento do sinal utilizado e sua frequência são muito importantes nesse tipo de sensor, pois determinam as dimensões mínimas do objeto que pode ser detectado.

O diagrama de tempo para o controle de módulo é dado a seguir, na Figura 6:

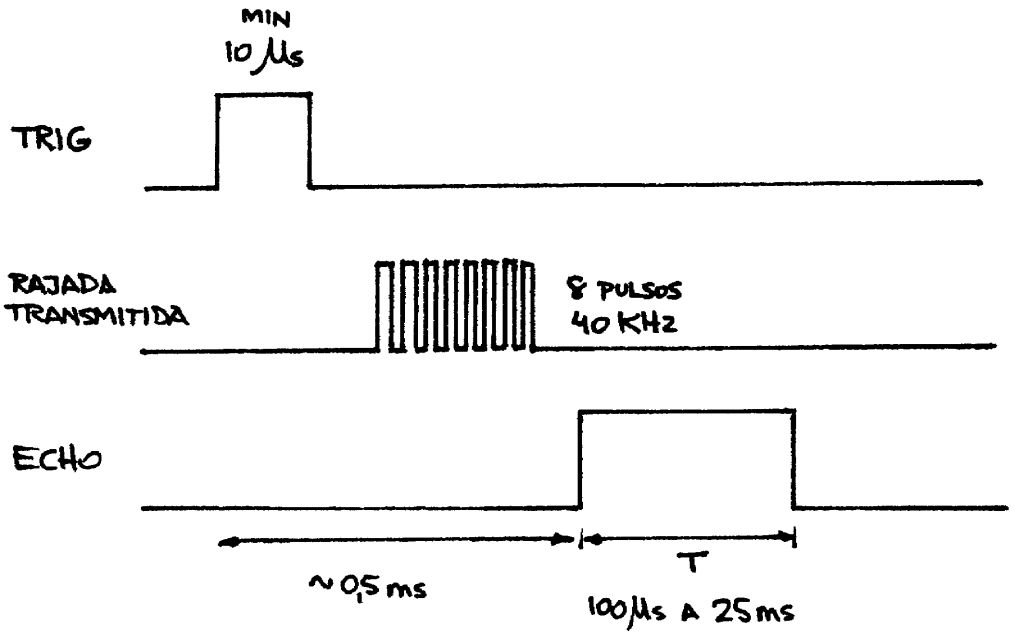

Figura 6: Diagrama de tempo para o controle do módulo. (GUIMARÃES 2014)

O dispositivo utilizado nesse trabalho é o Ultrassonic Sensor HC-SR04, cujas características fornecidas pelo fabricante são:

- Medida de distância entre 2 e 450cm.
- Ângulo máximo: 15º (entre as perpendiculares do módulo e do obstáculo).

- Precisão: 3mm.

4. MATERIAIS E MÉTODOS

4.1. Materiais

Os materiais utilizados no projeto do veículo são listados a seguir:

Para dar sustentação aos componentes do veículo, utilizou-se uma plataforma, feita de lâmina de acrílico com dimensões:11x26 (largura x comprimento), porcas e parafusos.

Como materiais para movimentar o veículo, utilizou-se duas rodas com eixo próprio para o motor, uma roda boba com eixo giratório, para seguir o movimento das rodas traseiras, dois motores DC, para controlar o movimento das rodas traseiras. A Figura 7 mostra o motor DC utilizado, a roda, um suporte de metal em formato L para fixar o motor na plataforma e os parafusos e porcas necessários para a fixação.

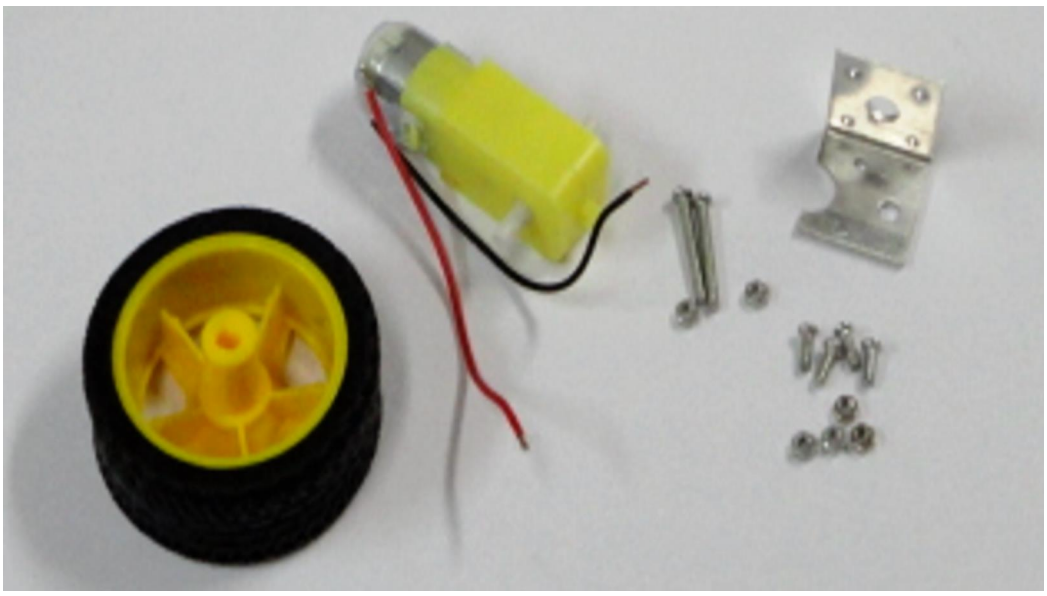

Figura 7: Materiais utilizados na plataforma. (GUIMARÃES,2014).

Para controlar a velocidade de rotação dos motores DC, utilizou-se duas pontes H, contidas no CI L298.

Os sensores utilizados para coletar os dados do mundo externo e enviá-los ao

microcontrolador foram: dois sensores ópticos GP452 Sharp, e um radar de ultrassom HC-SR04.

Na alimentação de todo o sistema utilizou-se duas baterias de 3,7V e cabos conectores.

Com o objetivo de simular e compilar os códigos utilizou-se o programa computacional Ísis Proteus.

4.2. Métodos

4.2.1. Familiarização com a Linguagem *Assembly e a plataforma de desenvolvimento Ísis PROTEUS*

Para Sousa (2005), a linguagem *Assembly* é o nível mais baixo em que se pode programar com alguma comodidade. A linguagem pode ser traduzida em código de máquina, à mão, mediante uma tabela de conversão, mas normalmente é feita recorrendo a um assemblador.

O Proteus é uma plataforma que auxilia o desenvolvimento de circuitos eletrônicos desde sua concepção, fazendo a simulação do circuito até a elaboração do layout da placa de circuito impresso.

A seguir, será mostrado, passo a passo, como se utilizou o Proteus para confeccionar o circuito utilizando o Microcontrolador 8052.

O objetivo desta etapa é entender a estrutura da linguagem de programação *Assembly,* bem como compilar o programa criado no bloco de notas, ou em qualquer outro editor de texto.

Primeiramente carregou-se o software Ísis Proteus, e adicionou-se os componentes no projeto. A Figura 8 mostra o exemplo de um circuito de controle de rotação de motor DC retirado do próprio Proteus.

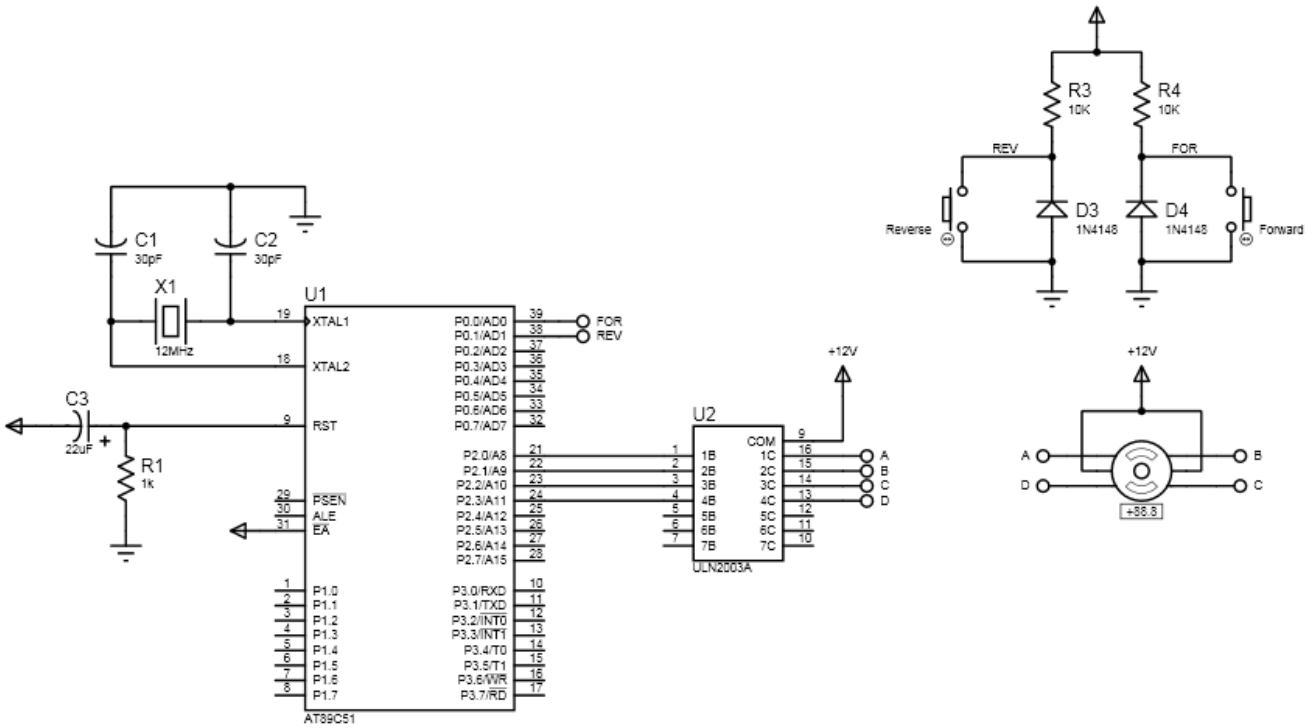

Figura 8: Circuito montado no software Ísis Proteus (GUIMARÃES, 2014)

Após a montagem do circuito, procedeu-se à criação do programa no bloco de notas. Neste exemplo o programa utilizado é fornecido como modelo no diretório SAMPLES do Ísis Proteus.

Para escrever o programa bastou clicar em *Source>Add/Remove Source files...>New* e deve se escolher o programa escrito no bloco de notas ou qualquer outro editor de texto com a extensão ñASMò Para compilar deve-se ir em *Source>Build All* e o programa antes em *Assembly* é carregado no microcontrolador agora em linguagem de máquina e extensão ñHEXò

O programa para controlar a rotação deste motor, encontra-se no ñANEXO 1ò

4.2.2. Sensores Ópticos e o programa AULA6.ASM

Inicialmente, o programa irá averiguar em que nível lógico se encontra o bit *port* 2.7, caso esteja em 0 (GND), executa o programa em que os 2 sensores ficam fora da faixa de referência, caso esteja em nível lógico 1 (+5V), executa o programa em que os dois sensores estão dentro da faixa.

A Figura 9 a seguir descreve o processo de averiguação do nível lógico do bit do *port* 2.7 e a execução dos códigos.

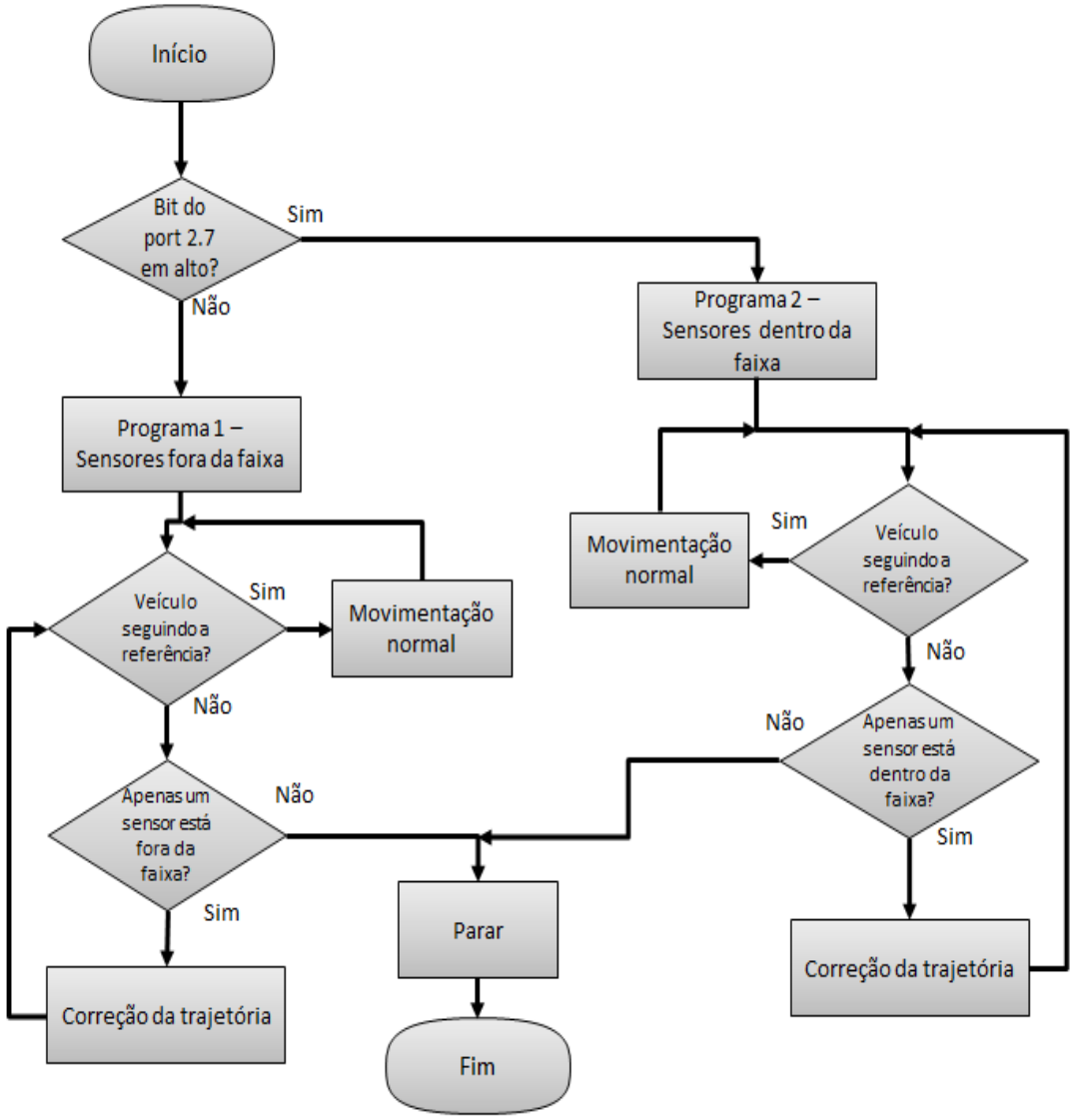

Figura 9: Fluxograma de execução dos códigos. (Elaborado pelos autores)

De acordo com Guimarães (2014), para o controle de trajetória é preciso no mínimo dois sensores ópticos para que se saiba se o veículo está saindo da linha a ser seguida, de um lado ou de outro. Assim, usando um pedaço de placa padrão, foi montado o circuito com dois sensores, separados de 2,5 cm. Este circuito será fixado na frente do veículo, com os sensores ópticos voltados para baixo, à uma distância 1 cm do chão, o que é suficiente para que os sensores identifiquem uma referência.

A referência tida como clara, a ser seguida será feita de cartolina branca com 6 cm de largura.

Para o código com os sensores dentro da faixa, a rotina é seguida: quando a

parte frontal do veículo estiver sobre a linha branca, os dois sensores estarão com seus transistores saturados (nível lógico "0" nos dois coletores). Se o veículo sair para a esquerda da referência, o sensor da esquerda terá seu transistor cortado (nível lógico "1" em seu coletor); se o veículo sair para a direita da referência, o sensor da direita terá seu transistor cortado.

Os coletores dos dois transistores foram ligados da seguinte maneira:

- coletor do sensor da direita (Saída D) no bit P0.0;
- coletor do sensor da esquerda (Saída E) no bit P0.1.

O programa principal tem 4 rotinas:

A primeira rotina, começando no *label* PRO_D1_E1x, verifica os níveis de P0.0 e P0.1. Se algum destes for nível "0", o programa vai para outra rotina. Se os dois forem nível "1", significa que o veículo saiu totalmente da referência, portanto, os motores devem ser desligados para que o mesmo pare.

A segunda rotina, começando no *label* PRO_D1_E0x, é chamada quando o sensor do lado direito for nível "1" e o do lado esquerdo for nível "0". Isso significa que o veículo está saindo da linha para o lado direito. Para que ele retorne para a referência, o motor do lado esquerdo é desligado e a tensão do motor do lado direito é aumentada.

A terceira rotina, começando no *label* PRO_D0_E1x, é chamada quando o sensor do lado direito for nível "0" e o do lado esquerdo for nível "1". Isso significa que o veículo está saindo da linha para o lado esquerdo. Para que ele retorne para a referência, o motor do lado direito é desligado e a tensão do motor do lado esquerdo é aumentada.

A última rotina, começando no *label* PRO_D0_E0x, é chamada quando os dois sensores forem nível "0". Isto significa que o veículo está sobre a referência e deve continuar seguindo em frente. Para isso, os motores dos dois lados são ligados para frente.

Para o código em que a faixa é interna aos sensores, o *label* PRO_D0_E0 é responsável por verificar se os dois sensores estão na referência tida como clara, caso estejam o motor da direita e da esquerda desligam.

O segundo *label*, PRO_D0_E1, traduz que o sensor da direita está na referência tida como clara, e o sensor da esquerda segue o que é entendido como escuro, portanto corrigirá sua trajetória desligando o motor da direita e ligando o da esquerda para frente.

O terceiro trecho, do *label* PRO_D1_E0, traduz que o sensor da direita segue o que é entendido como escuro, e o sensor da esquerda está na referência tida como clara, portanto corrigirá sua trajetória ligando o motor da direita para frente e desligando o da esquerda.

Por fim, o trecho com o *label* PRO_D1_E1 representa que os dois sensores estão na referência tida como escura, portanto o motor da direita e da esquerda serão ligados para frente.

O programa AULA6.ASM, que foi implementado para o projeto encontra-se no ñANEXO 2à

4.2.3. Radar de ultrassom acoplado ao servomotor

O servomotor foi comandado para fazer uma varredura de 180º no ambiente frontal do veículo, durante um tempo de aproximadamente 1s. Por motivos de limitações do servomotor, seu controle só é possível com incrementos de ângulos de aproximadamente 20º. Portanto, o radar acoplado ao servomotor efetuou cerca de 9 medições, a partir do lado direito(0º) até o lado esquerdo(180º).

Na prática, o servomotor só consegue realizar o movimentos menores que 180º. Portanto, considerou-se a posição mais à direita como 180º e à posição mais à esquerda como 170º, com divisões a cada 20º, resultando em 9 posições de um extremo a outro. A Figura 10 mostra o radar de ultrassom acoplado ao servomotor.

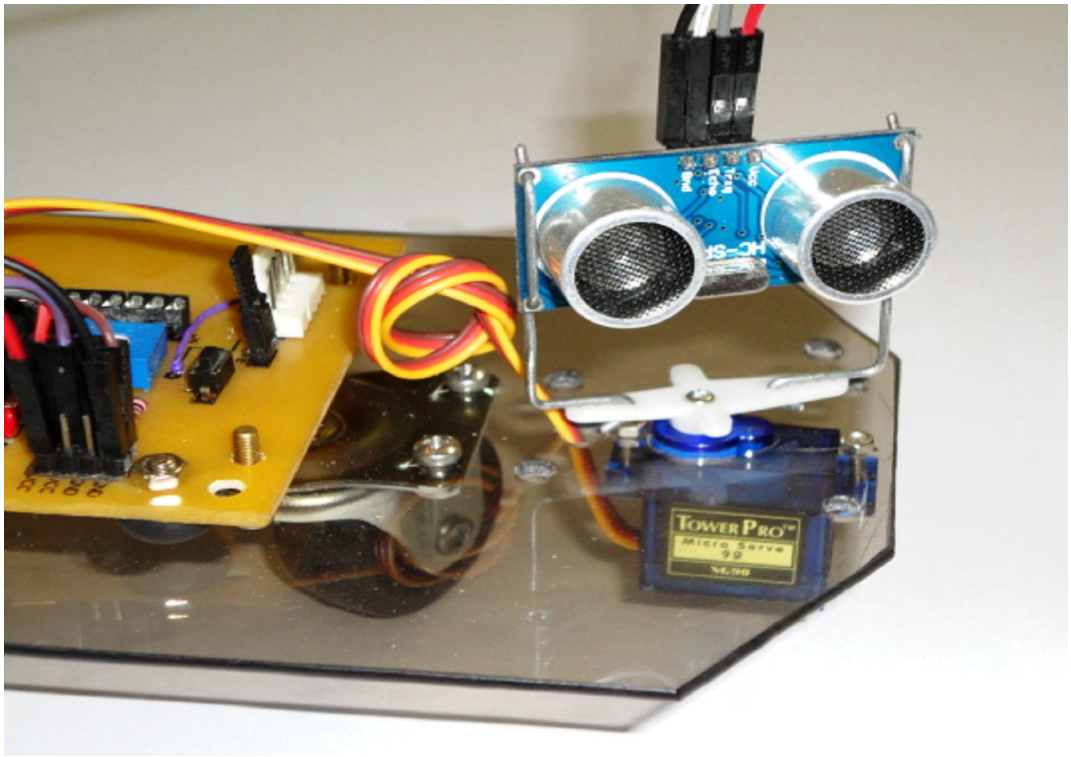

Figura 10: Radar de ultrassom acoplado ao servomotor. (GUIMARÃES,2014).

4.2.4. O programa CARRO09.ASM

Utilizando o programa em Assembly CARRO09.ASM, que foi implementado, carregou-se o software Ísis Proteus para simular e compilar, e escreveu-se o programa CARRO9.ASM no componente do software (microcontrolador). Em seguida mandou-se compilar o programa, para que o código em linguagem *Assembly* se convertesse em linguagem de máquina, pois as linguagens de programação são apenas maneiras mais confortáveis de se escrever as sub-rotinas,

A sub-rotina LER_RADAR_A faz a varredura do radar no sentido anti-horário, começando na posição 10º (ANGULO = 3), espera um tempo de 0,1s para que o servomotor chegue à posição desejada, aciona o radar (sub-rotina CALC_DIST), e armazena o valor lido da distância na variável DIST3. Em seguida, define a posição 30º (ANGULO = 4), espera 0,1s para que o servomotor chegue à posição desejada, aciona o radar e armazena o novo valor na variável DIST4, e assim por diante para as posições 50º (ANGULO= 5), 70º (ANGULO = 6), 90º (ANGULO = 7), 110º (ANGULO= 8), 130º (ANGULO=9), 150º (ANGULO = 10) e 170º (ANGULO = 12), com as medidas armazenadas respectivamente nas variáveis DIST5, DIST6, DIST7, DIST8, DIST9, DIST10 e DIST12.

Com os nove valores lidos e armazenados, a sub-rotina MEDIA calcula as médias aritméticas dos três valores mais à direita, dos três valores mais frontais e dos três valores mais à esquerda, guardando esses valores, respectivamente nas variáveis DIST_D, DIST_F, e DIST_E. O motivo de tirar as médias aritméticas é que o radar só consegue boa medida quando o obstáculo está bem perpendicular ao mesmo. Quando isso não acontece o erro na medida será grande.

A lógica de tomadas de decisões a ser seguida, já com os nomes das sub-rotinas a serem chamadas para a execução de cada movimento é descrita na Tabela 1.

Tabela 1: Lógica a ser seguida.

DIST_E	DIST_F	DIST_D	Sub-rotina	Decisão
<CONST1	<CONST1	<CONST1	PARAR	Parar
<CONST1	< CONST1	> CONST1	GIRAR_H90	Girar 90º no sentido horário
< CONST1	> CONST1	< CONST1	PARAR	Parar
< CONST1	> CONST1	> CONST1	GIRAR_H45	Girar 45º no sentido horário
> CONST1	< CONST1	< CONST1	GIRAR_AH90	Girar 90º no sentido anti-horário
> CONST1	< CONST1	> CONST1	GIRAR_AH90 ou GIRAR_H90	Girar 90º no sentido anti-horário ou girar 90º no sentido horário
> CONST1	> CONST1	< CONST1	GIRAR_AH45	Girar 45º no sentido anti-horário
> CONST1	> CONST1	> CONST1	FRENTE	Seguir em frente

O *label* CONST1 é inicialmente definido como 15 (que representa 15 cm), ou seja, o valor das variáveis DIST_E, DIST_E e DIST_F são comparados com essa constante e as decisões são tomadas de acordo com casa caso.

A segunda metade do código, faz a mesma coisa que a primeira metade, apenas chamando a sub-rotina LER_RADAR_H, que faz a varredura no sentido

horário, ou seja de DIST12 à DIST3.

O código do programa CARRO09.ASM encontra-se no ñANEXO 3ò

5. RESULTADOS E DISCUSSÕES

Neste trabalho abordamos as formas de controle autônomo do veículo e esperava-se que ao final do projeto, o veículo fosse capaz de corrigir seu percurso ao se deparar com obstáculos.

Durante o desenvolvimento do projeto, foram testadas algumas formas de controle autônomo do veículo, ou seja, recebia-se as informações do mundo externo, tais como distâncias entre os obstáculos e o veículo, por intermédio dos sensores, e o microcontrolador tomava as decisões encaminhando o sinal para os atuadores.

Ao longo do trabalho analisamos e criamos códigos para realizar determinadas tarefas e posteriormente as testamos, verificando que o veículo realizou o que se esperava de forma satisfatória, porém com algumas limitações de construção, como no caso do servomotor, que não consegue realizar a rotação de 180° de forma precisa.

Ao colocar o veículo em um ambiente com obstáculos, o mesmo realizou o que se esperava, corrigindo sua trajetória quando necessário. Com isso, pode-se concluir que a utilização de sensores, e atuadores trabalhando junto com um microcontrolador com determinadas tarefas programadas, é capaz de fazer com que o veículo se desloque de forma autônoma, seguindo uma referência e guiando-se pelas informações recebidas pelos sensores.

6. SUGESTÕES PARA TRABALHOS FUTUROS

Parar dar continuidade ao nosso projeto, sugerimos que se converta os programas utilizados nesse trabalho (linguagem *Assembly*), para a linguagem de programação C, que é a utilizada no curso técnico em Automação Industrial no IFSP.

7. REFERÊNCIAS BIBLIOGRÁFICAS

DELAI, R.L. ; COELHO, A.D.; MAZZEO, P.T.C., **Desenvolvimento de veículo autônomo - Inteligência periférica, sensoriamento de sistemas de emergência.** São Caetano do Sul. Instituto Mauá de Tecnologia. Seminário Mauá de iniciação científica, p.2, 2011. Disponível em: <http://www.maua.br/arquivos/index/h/81d3fc8a63b6daf60bf0a3805f447ef2?> . Acessado em: 17 mar. 2014.

SILVA, F. S. L. da; SILVA, T. S. L. da; SILVA, A. V. da; HORTA, M.M.B., **Conversor de frequência CC-CA.** Belo Horizonte. e-xacta. Editora UniBH, p.78, 2013. Disponível em: <revistas.unibh.br/index.php/dcet/article/download/883/559>. Acessado em: 17 mar. 2014.

ZUCATELLI, F. H. G.; OLIVEIRA, M. A. V. de., **Controle de servomotores CC.** São Bernardo do Campo. Faculdade de Tecnologia Termomecânica. Artigo Científico, p.1, 2007. Disponível em: <pt.scribd.com/doc/16300774/Controle-de-Servomotores-CC>. Acessado em: 17 mar. 2014.

BRAGA, N. C., **Como funcionam os sensores ultrassônicos (ART691).** Instituto Newton C. Braga, 2012. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1111/j.1745-4549.1996.tb00761.x/abstract> . Acessado em: 17 mar. 2014.

GUIMARÃES, E., **Curso prático de aquisição de informações e atuação em sistemas VANT: Controle com Radar.** Instituto Tecnológico de Aeronáutica- Departamento de Eletrônica Aplicada. 2014.

ATMEL**. *AT89S52-24PU - ATMEL - IC, 8BIT MCU FLASH, 89S52, DIP40 | Farnell UK*.** 06 de 08 de 2008. Disponível em: <http://uk.farnell.com/atmel/at89s52-24pu/ic-8bit-mcu-flash-89s52-dip40/dp/1095743> . Acessado em: 17 mar. 2014.

GUIMARÃES, E., **Primeira Experiência Microcontrolador 8052 e Plataforma de Desenvolvimento Proteus.** Instituto Tecnológico de Aeronáutica- Departamento de Eletrônica Aplicada IEEA-EEA-27 ï Microcontroladores e Sistemas Embarcados- 2014.

GUIMARÃES, E., **Curso Prático de Aquisição de Informações e Atuação em Sistemas VANT:RADAR COM ULTRASSOM**. Instituto Tecnológico de Aeronáutica- Departamento de Eletrônica Aplicada IEEA-EEA-27 ï Microcontroladores e Sistemas Embarcados- 2014.

GUIMARÃES, E., **Quarta experiência: Construção e Teste de uma Plataforma Móvel**. Instituto Tecnológico de Aeronáutica- Departamento de Eletrônica Aplicada IEEA-EEA-27 ï Microcontroladores e Sistemas Embarcados- 2014.

GUIMARÃES, E.; OLIVEIRA, Neusa. **Curso Prático de Aquisição de Informações e Atuação em Sistemas VANT: Familiarização com Sistema Autônomo**. Instituto Tecnológico de Aeronáutica- Departamento de Eletrônica Aplicada IEEA-EEA-27 ï Microcontroladores e Sistemas Embarcados- 2013.

SOUSA, J.P. **Primeiros Passos na Programação em Linguagem Assembly**. U.Porto Faculdade de Engenharia - Departamento de Engenharia Eletrotécnica e de Computadores. 4 Ed. Portugal, 2005. Disponível em: <http://paginas.fe.up.pt/~jmf/mp0506/dwnlds/mp1-0506-print.pdf> Acessado em: 12 mai. 2014.

GUIMARÃES, E., **Sexta Experiência:Controle da Plataforma Móvel Usando Realimentação com Sensor Ótico**. Instituto Tecnológico de Aeronáutica- Departamento de Eletrônica Aplicada IEEA-EEA-27 ï Microcontroladores e Sistemas Embarcados- 2014.

ANEXO 1- Programa em linguagem Assembly, para controlar um motor.

```
ORG      00H
START:
      MOV   DPTR,#TAB1
      MOV   R0,#3
      MOV   R4,#0
      MOV   P2,R0
WAIT: MOV   P0,#0FFH
      JNB   P0.0,POS    ; Wait for a key pressed
      JNB   P0.1,NEG
      MOV   P2,#00H
      SJMP  WAIT
POS:  MOV   R4,#1
      MOV   A,R4        ; Forward direction
      MOVC  A,@A+DPTR
      MOV   P2,A
      ACALL DELAY
      AJMP  KEY
NEG:  MOV   R4,#7       ; Reverse direction
      MOV   A,R4
      MOVC  A,@A+DPTR
      MOV   P2,A
      ACALL DELAY
       AJMP  KEY
```

```
KEY:  MOV   P0,#03H
      JB    P0.0,NR1
      INC   R4
      CJNE  R4,#9,LOOPP
      MOV   R4,#1
LOOPP:
      MOV   A,R4
      MOVC  A,@A+DPTR
      MOV   P2,A
      ACALL DELAY
      AJMP  KEY
NR1:
      JB    P0.1,START
      DEC   R4
      CJNE  R4,#0,LOOPN
      MOV   R4,#8
LOOPN:
      MOV   A,R4
      MOVC  A,@A+DPTR
      MOV   P2,A
      ACALL DELAY
      AJMP  KEY
DELAY:
      MOV   R6,#1
DD1:  MOV   R5,#80H
```

```
DD2:  MOV   R7,#0

DD3:  DJNZ  R7,DD3

      DJNZ  R5,DD2

      DJNZ  R6,DD1

      RET

      ; Table of Stepping Sequences

TAB1: DB    00H,02H,06H,04H

      DB    0CH,08H,09H,01H,03H

      END
```

ANEXO 2- Código do programa AULA6.asm

```
;*******************************************************************************
; 5° acréscimo - Realimentação com sensor Ótico

; Arquivo AULA6.ASM
;*******************************************************************************

REGO            EQU                     20H
REG1            EQU                     21H
MILHAR          EQU                     22H
CENTENA         EQU                     23H
DEZENA          EQU                     24H
UNIDADE         EQU                     25H
NUMERO          EQU                     26H
CONT            EQU                     27H
V_D             EQU                     28H
y_E             EQU                     29H
ANGULO          EQU                     2AH

ORG             0000H
JMP INI

ORG             000BH
JMP ROT_T0 ;rotina de interrupção do Timer 0

ORG             100H


INI:            MOV SP, #6FH
                CALL INICIAL ;inicialização do display
                CALL ESC_MENS1 ;escreve a Mensagem 1
                CALL ESC_MENS_2 ;escreve a Mensagem 2
                MOV MILHAR, #20H
                MOV P2, #OFFH
```

```
                MOV CONT, #0
                MOV V_D, #0
                MOV y_E, #0
                MOV ANGULO, #0

                MOV TMOD, #02H ;Timer 0 no Modo 2
                MOV TCON, #0
                MOV TL0, #56 ;tempo de 200us
                MOV TH0, #56
                MOV IE, #10000010B ;interrupção Timer 0
                SETB TCON.4 ;liga o Timer 0

                MOV CENTENA, #20H
                MOV DEZENA, #20H
                MOV UNIDADE, #20H

;*********************************************************************
; Programa principal para Realimentação com Sensor Ótico
;*********************************************************************

                P2.7, PRO_D1_E1x ;vai para o 2° programa

;1° Programa - a faixa branca fica interna aos sensores

PRO_D0_E0:      JB P0.0, PRO_D1
                JB P0.1, PRO_D0_E1
                CALL DES_D
                CALL DES_E
                MOV UNIDADE, #30H
                CALL ESC_VALOR_1
                CALL ESC_VALOR_2
                JMP PRO_D0_E0

PRO_D0_E1:      MOV V_D, #0
                MOV V_E, #100
                CALL DES_D
                CALL LIG_E_F
                MOV UNIDADE, #30H
                CALL ESC_VALOR_1
                MOV UNIDADE, #31H
                CALL ESC_VALOR_2
                JMP PRO_D0_E0

PRO_D1:         JB P0.1, PRO_D1_E1
PRO_D1_E0:      MOV V_D, #100
                MOV V_E, #0
                CALL LIG_E_F
                CALL DES_E
                MOV UNIDADE, #31H
                CALL ESC_VALOR_1
                MOV UNIDADE, #30H
                CALL ESC_VALOR_2
                JMP PRO_D0_E0

PRO_D1_E1:      MOV V_D, #50
                MOV V_E, #50
                CALL LIG_D_F
                CALL LIG_E_F
                MOV UNIDADE, #31H
                CALL ESC_VALOR1
```

```
                CALL ESC_VALOR_2
                JMP PRO_DO_EO

;2° Programa - os sensores ficam internos à faixa branca

PRO_Dl_Elx:     JNB P0.0, PRO_D0x
                JNB P0.1, PRO_Dl_E0x
                CALL DES_D
                CALL DES_E
                MOV UNIDADE, #31H
                CALL ESC_VALOR_1
                CALL ESC_VALOR_2
                JMP PRO_D1_Elx

PRO_D1_E0x:     MOV V_D, #100
                MOV V_E, #100
                CALL LIG_D_F
                CALL LIG_E_T
                MOV UNIDADE, #31H
                CALL E5C_VALOR_1
                MOV UNIDADE, #30H
                CALL ESC_VALOR_2
                JMP PRO_D1_Elx


PRO_D0x:        JNB P0.1, PRO_D0_E0x

PRO_D0_Elx:     MOV V_D, #100
                MOV V_E, #100
                CALL LIG_D_T
                CALL LIG_E_F
                MOV UNIDADE, #30H
                CALL ESC_VALOR_1
                MOV UNIDADE, #31H
                CALL ESC_VALOR_2
                JMP PRO_D1_Elx

PRO_D0_E0x:     MOV V_D, #50
                MOV y_E, #SO
                CALL LIG_D_F
                CALL LIG_E_F
                MOV UNIDADE, #30H
                CALL ESCVALOR_1
                CALL ESC_VALOR_2
                JMP PRO_D1_E1x

;*****************************************************************************
; SUB-ROTINAS

ESC_DADO:       MOV P1, A ;Escreve Dado
                SETB P3.7
                CLR P3.6
                SETB P3.5
                CLR P3.5
                CALL DELAY5Ou
                RET

ESC_COM:        MOV P1, A ;Escreve Comando
                CLR P3.7
                CLR P3.6
                SETB P3.5
```

```
                CLR P3.5
                CALL DELAY50u
                RET

INICIAL:        CALL DELAY5m ;espera 15 milissegundos
                CALL DELAY5m
                CALL DELAY5m
                MOV A, #38H ;Function Set
                CALL ESC_COM ;(8 bits, 2 linhas, 5x7 pontos)
                CALL DELAY5m
                MOV A, #38H ;Function set novamente
                CALL ESC_COM
                CALL DELAY5m ;espera 5 milisseguridos
                MOV A, #06H ;Entry Mode Set
                CALL ESC_COM ;(incrementa, shift cursor)
                MOV A, #0EH ;Display Control
                CALL ESC_COM ;(display on, cursor on)
                MOV A, #01H ;clear Display
                CALL ESC_COM
                CALL DELAY5m ;espera 5 milissegundo
                RET

CONVERTE:       MOV A, NUMERO ;obtém códigos ASCII da Centena,
                MOV B, #100 ;Dezena e unidade do NUMERO
                DIV AB
                ADD A, #30H
                MOV CENTENA, A
                MOV A,B
                MOV B, #10
                DIV AB
                ADD A, #30H
                MOV DEZENA, A
                MOV A, B
                ADD A, #30H
                MOV UNIDADE, A
                RET

MOSTRA_NUM:     MOV A, MILHAR ;Mostra Milhar, Centena,
                CALL ESC_DADO ;Dezena e unidade
                MOV A, CENTENA
                CALL ESC_DADO
                MOV A, DEZENA
                CALL ESC_DADO
                MOV A, UNIDADE
                CALL ESC_DADO
                RET

MOSTRA_MENS:    CLR A ;mostra Mensagem apontada por DPTR
                MOVC A, @A+DPTR ;caracter nulo terminador da mensagem
                JZ MOSTRA_FIM
                CALL ESC_DADO
                INC DPTR
                JMP MOSTRA_MENS

MOSTRA_FIM: RET

ESC_MENS_1:     MOV A, #80H ;1ª posição da 1ª linha do display
                CALL ESC_COM
                MOV DPTR, #MENS_1 ;aponta para a 1ª mensagem fixa
                CALL MOSTRA_MENS
                RET
```

```
ESC_MENS_2          MOV A, #0C0H ;1ª posição da 2ª linha do display
                    CALL ESC_COM
                    MOV DPTR, #MENS_2 ;aponta para a 2ª mensagem fixa
                    CALL MOSTRA_MENS
                    RET

ESC_VALOR_1:        MOV A, #8CH ;13ª posição da 1ª linha do display
                    CALL ESC_COM
                    CALL MOSTRA_NUM ;mostra VALOR 1
                    RET

ESC_VALOR_2:        MOV A, #0CCH ;13ª posição da 2ª linha do display
                    CALL ESC_COM
                    CALL MOSTRA_NUM ;mostra VALOR 2
                    RET

DELAY5u:            NOP ;atraso de 5 microssegundos
RET

DELAY5Ou:           MOV REG0, #22 ;atraso de 50 microssegundos
DELAY5Ou1:          DJNZ REG0, DELAY50u
                    RET

DELAY1m:            MOV REG0, #248 ;atraso de 1 milissegundo
DELAY1m1:           NOP
                    NOP
                    DJNZ REG0, DELAY1m1
                    NOP
                    NOP
                    RET

DELAY5m:            MOV REG1, #96 ;atraso de 5 milissegundos

DELAY5m1:           CALL DELAY50U
                    DJNZ REG1, DELAY5m1
                    NOP
                    NOP
                    RET

DELAY100m:          MOV REG1, #95 ;atraso de 100 milissegundoS
DELAy100m1:         CALL DELAY1m
                    CALL DELAY50u
                    DJNZ REG1, DELAy100m1
                    CALL DELAY50u
                    NOP
                    NOP
                    NOP
                    NOP
                    RET

DELAY05s:           MOV REG1, #249 ;atraso de 0,5 segundo
DELAYO5s1:          CALL DELAY1m
                    CALL DELAY1m
                    CALL DELAY5u
                    NOP
                    DJNZ REG1, DELAY05s1
                    NOP
                    NOP
                    RET

DELAY1s:            CALL DELAY05s ;atraso de 1 segundo
```

```
                    CALL DELAY05s
                    RET

DELAY2S:            CALL DELAY1s ;atraso de 2 segundos
                    CALL DELAY1s
                    RET

DES_D:              SETB P2.1 ;desliga motor direita
                    SETB P2.2
                    RET

DES_E:              SETB P2.3 ;desliga motor esquerda
                    SETB P2.4
                    RET

LIG_D_F:            SETB P2.1 ;liga motor direita p/ frente
                    CLR p2.2
                    RET

LIG_D_T:            CLR P2.1 ;liga motor direita p/ trás
                    SETB P2.2
                    RET

LIG_E_F:            CLR P2.3 ;liga motor esquerda p/ frente
                    SETB P2.4
                    RET

LIG_E_T:            SETB P2.3 ;liga motor esquerda p/ trás
                    CLR P2.4
                    RET

ROT_T0:             PUSH PSW     ;rotina de interrupção do Timer 0
                    PUSH ACC     ;gera 3 ondas PWM em P2.0, P2.5 e P2.6
                    MOV A, CONT  ;as larguras das ondas dependem de
                    CLR C       ;V_D, V_E e ANGULO
                    SUBB A, V_D  ;para V_D e V_E, cada incremento
                    JC ROT_T0A   ;aumenta 1% na largura da onda PWM
                    CLR P2.0     ;para ANGULO, cada incremento
                    JMP ROT_T0B  ;(somente entre os valores 3 e 12)

ROT_T0A:            SETB P2.0      ;representa aumento de 200 de rotação
ROT_TOB:            MOV A, CONT    ;no eixo do servomotor
                    CLR C          ;o período de cada onda PWM é 2Oms
                    SUBB A, V_E    ;o que significa frequência de 50Hz
                    JC ROT_T0C
                    CLR P2.5
                    JMP ROT_T0D

ROT_T0C:            SETB P2. 5
ROT_T0D:            MOV A, CONT
                    CLR C
                    SUBB A, ANGULO
                    JC ROT_T0E
                    CLR P2.6
                    JMP ROT_T0F

ROT_T0E:            SETB P2.6
ROT_T0F:            INC CONT
                    MOV A, CONT
                    CJNE A, #100, ROT_T0G
                    MOV CONT, #0
```

```
ROT_T0G:                POP ACC
                        POP PSW
                        RETI

MENS_1:                 DB "VALOR DIR:", 0
MENS_2:                 DR "VALOR ESQ:", 0

END
```

ANEXO 3- Código do programa CARRO09.asm

```
;*********************************************************************
;                  Programa com todas as implementações do CARRO
;                        7° acréscimo – Controle com Radar
;                  Arquivo CARRO09.ASM
;*********************************************************************

CONST1      EQU   15
CONST2      EQU   30
REG0        EQU   20H
REG1        EQU   21H
MILHAR      EQU   22H
CENTENA     EQU   23H
DEZENA      EQU   24H
UNIDADE     EQU   25H
NUMERO      EQU   26H
CONT        EQU   27H
V_D         EQU   28H
V_E         EQU   29H
ANGULO      EQU   2AH
DISTH       EQU   2BH
DISTL       EQU   2CH
DIST        EQU   2DH
DISTM       EQU   2EH
DIST3       EQU   2FH
DIST4       EQU   30H
DIST5       EQU   31H
DIST6       EQU   32H
DIST7       EQU   33H
DIST8       EQU   34H
DIST9       EQU   35H
DIST10      EQU   36H
DIST12      EQU   37H
DIST_F      EQU   38H
DIST_D      EQU   39H
DIST_E      EQU   3AH


ORG         0000H
            JMP   INI

ORG         000BH
            JMP   ROT_T0                ; rotina de interrupção do Timer
      0

ORG         100H
```

```
INI:        MOV   SP,#6FH
            CALL  INICIAL                 ; inicialização do display
            CALL  ESC_MENS_1              ; escreve a Mensagem 1
            CALL  ESC_MENS_2              ; escreve a Mensagem 2
            MOV   MILHAR, #20H
            MOV   P2, #0FFH

            MOV   V_D, #0
            MOV   V_E, #0
            MOV   CONT, #0

            MOV   TMOD, #02H              ;Timer 0 no modo 2
            MOV   TCON, #0
            MOV   TL0, #56                ;Tempo de 200us
            MOV   TH0, #56
            MOV    IE, #10000010B         ;Interrupção timer 0
            SETB  TCON.4                  ;Liga o timer 0

            MOV   CENTENA, #20H
            MOV   DEZENA, #20H
            MOV   UNIDADE, #20H

            ORL   TMOD, #10H              ;Timer 1 no modo 1
            CLR   TCON.6
            CLR   TCON.7

;-------------------------------------------------------------------------
;           Programa principal para controle com radar

            MOV   ANGULO, #3
            CALL  DELAY2s
VOLTA:      CALL  LER_RADAR_A
            CALL  MEDIA
            CALL  ESC_RADAR
            CALL  PROCESSAR
            CALL  LER_RADAR_H
            CALL  MEDIA
            CALL  ESC_RADAR
            CALL  PROCESSAR
            JMP   VOLTA


;*************************************************************************
;           SUB-ROTINAS
;*************************************************************************

ESC_DADO:   MOV   P1, A                   ;Escreve dado
            SETB  P3.7
            CLR   P3.6
            SETB  P3.5
            CLR   P3.5
            CALL  DELAY50u
            RET

ESC_COM:    MOV   P1, A                   ;Escreve comando
            CLR   P3.7
            CLR   P3.6
            SETB  P3.5
            CLR   P3.5
            CALL  DELAY50u
```

```
            RET

INICIAL:    CALL  DELAY5m               ;Espera     15 milissegundos
            CALL  DELAY5m
            CALL  DELAY5m
            MOV   A, #38H               ;Function set
            CALL  ESC_COM               ;(8 bits, 2 linhas, 5x7 pontos)
            CALL  DELAY5m
            MOV   A, #38H               ;Function set novamente
            CALL  ESC_COM
            CALL  DELAY5m               ;Espera 5 milissegundos
            MOV   A, #06H               ;Entry Mode Set
            CALL  ESC_COM               ;(incrementa, shift cursor)
            MOV   A, #0EH               ;Display control
            CALL  ESC_COM               ;(display on, cursor on)
            MOV   A, #01H               ;clear display
            CALL  ESC_COM
            CALL  DELAY5m               ;Espera 5 milissegundos
            RET

CONVERTE:   MOV   A, NUMERO             ;Obtém códigos ASCII da Centena,
            MOV   B, #100                     ;Dezena e Unidade do número
            DIV   AB
            ADD   A, #30H
            MOV   CENTENA, A
            MOV   A, B
            MOV   B, #10
            DIV   AB
            ADD   A, #30H
            MOV   DEZENA, A
            MOV   A, B
            ADD   A, #30H
            MOV   UNIDADE, A
            RET

MOSTRA_NUM: MOV   A, MILHAR         ;Mostra milhar, centena
            CALL  ESC_DADO          ;Dezena e unidade
            MOV   A, CENTENA
            CALL  ESC_DADO
            MOV   A, DEZENA
            CALL  ESC_DADO
            MOV   A, UNIDADE
            CALL  ESC_DADO
            RET

MOSTRA_MENS:      CLR   A           ;Mostra mensagem apontada por DPTR
                  MOVC  A, @A+DPTR  ;Caractere nulo terminador da mensagem
                  JZ    MOSTRA_FIM
                  CALL  ESC_DADO
                  INC   DPTR
            JMP   MOSTRA_MENS
MOSTRA_FIM:       RET

ESC_MENS_1:       MOV   A, #80H     ;1ª posição da 1ª linha dodisplay
                  CALL  ESC_COM
                  MOV   DPTR, #MENS_1     ;Aponta para a 1ª mensagem fixa
                  CALL  MOSTRA_MENS
                  RET
ESC_MENS_2:       MOV   A, #0C0H    ;1ª posição da 2ª linha do display
                  CALL  ESC_COM
                  MOV   DPTR, #MENS_2     ;Aponta para a 2ª mensagem fixa
```

```
                CALL   MOSTRA_MENS
                RET

ESC_VALOR_1:    MOV    A, #8CH     ;13ª posição da 1ª linha do display
                CALL   ESC_COM
                CALL   MOSTRA_NUM  ;Mostra valor 1
                RET

ESC_VALOR_2:    MOV    A, #0CCH    ;13ª posição da 2ª linha do display
                CALL   ESC_COM
                CALL   MOSTRA_NUM  ;Mostra valor 2
                RET

DELAY5u:        NOP                ;Atraso de 5 microssegundos
                RET

DELAY50u:       MOV    REG0, #22   ;Atraso de 50 microssegundos
DELAY50u1:      DJNZ   REG0, DELAY50u1
                RET

DELAY1m:        MOV    REG0, #248  ;Atraso de 1 milissegundo
DELAY1m1:       NOP
                NOP
                DJNZ   REG0, DELAY1m1
                NOP
                NOP
                RET

DELAY5m:        MOV    REG1, #96   ;Atraso de 5 milissegundos
DELAY5m1:       CALL   DELAY50u
                DJNZ   REG1, DELAY5m1
                NOP
                NOP
                RET

DELAY01s:       MOV    REG1, #95   ;Atraso de 0,1 segundo
DELAY01s1:      CALL   DELAY1m
                CALL   DELAY50u
                DJNZ   REG1, DELAY01s1
                CALL   DELAY50u
                NOP
                NOP
                NOP
                NOP
                RET

DELAY05s:       MOV    REG1, #249  ;Atraso de 0,5 segundo
DELAY05s1:      CALL   DELAY1m
                CALL   DELAY1m
                CALL   DELAY5u
                NOP
                DJNZ   REG1, DELAY05s1
                NOP
                NOP
                RET

DELAY1s:        CALL   DELAY05s    ;Atraso de 1 segundo
                CALL   DELAY05s
                RET
```

```
DELAY2s:        CALL  DELAY1s           ;Atraso de 2 segundos
                CALL  DELAY1s
                RET

DES_D:          SETB  P2.1              ;Desliga motor da direita
                SETB  P2.2
                RET

DES_E:          SETB  P2.3              ;Desliga motor da esquerda
                SETB  P2.4
                RET

LIG_D_F:        SETB  P2.1              ;Liga motor direita p/ frente
                CLR   P2.2
                RET

LIG_D_T:        CLR   P2.1              ;Liga motor direita p/ trás
                SETB  P2.2
                RET

LIG_E_F:        CLR   P2.3              ;Liga motor esquerda p/ frente
                SETB  P2.4
                RET

LIG_E_T:        SETB  P2.3              ;Liga motor esquerda p/ trás
                CLR   P2.4
                RET

ROT_T0:         PUSH  PSW               ;Rotina de interrupção do timer 0
                PUSH  ACC               ;Gera 3 ondas PWM em P2.0, P2.5 e
                P2.6
                MOV   A, CONT           ;As larguras das ondas dependem
                de
                CLR   C                 ;V_D, V_E e ANGULO
                SUBB  A, V_D                  ;Para V_D e V_E, cada
                incremento
                JC    ROT_T0A           ;Aumenta 1% na largura da onda
                PWM
                CLR   P2.0              ;Para ANGULO, cada incremento
                JMP   ROT_T0B           ;(somente entre os valores 3 e
                12)
ROT_T0A:        SETB  P2.0        ;Representa aumento de 20° de rotação
ROT_T0B:        MOV   A, CONT     ;No eixo do servomotor
                CLR   C           ;O período de cada onda PWM é 20ms
                SUBB  A, V_E      ;O que significa frequência de 50Hz
                JC    ROT_T0C
                CLR   P2.5
                JMP   ROT_T0D
ROT_T0C:        SETB  P2.5
ROT_T0D:        MOV   A, CONST
                CLR   C
                SUBB  A, ANGULO
                JC    ROT_T0E
                CLR   P2.6
                JMP   ROT_T0F
ROT_T0E:        SETB  P2.6
ROT_T0F:        INC   CONT
                MOV   A, CONT
                CJNE  A, #100, ROT_T0G
                MOV   CONT, #0
ROT_T0G:        POP   ACC
```

```
                POP   PSW
                RETI

CALC_DIST:      MOV   TH1, #0
                MOV   TL1, #0
                SETB  P2.7              ;Pulso de trigger
                CALL  DELAY5u
                CALL  DELAY5u
                CALL  DELAY5u
                CLR   P2.7
CALC1:          JNB   P0.2, CALC1 ;Espera Echo ir para alto
                SETB  TCON.6      ;Liga o timer 1
CALC2:          JB    P0.2, CALC2 ;Espera Echo voltar para baixo
                CLR   TCON.6      ;Desliga o timer 1
                CLR   TCON.7
                MOV   DISTH, TH1
                MOV   DISTL, TL1
                MOV   A, TL1
                RLC   A
                MOV   R0, A
                MOV   A, TH1
                RLC   A
                MOV   DIST, A           ;Contém distância em 1 byte
                MOV   R1, A             ;195 corresponde a 4,25m
                MOV   A, R0             ;Multiplicar 195 por 2,2
                RLC   A                 ;Para mostrar valor em cm
                MOV   A, R1             ;255 corresponde a 255cm
                RLC   A
                MOV   R0, A
                MOV   A, R1
                MOV   B, #5
                DIV   AB
                ADD   A, R0
                MOV   DISTM, A
                RET

LER_RADAR_A:    CALL  PARAR
                MOV   ANGULO, #3
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST3, DISTM
                MOV   ANGULO, #4
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST4, DISTM
                MOV   ANGULO, #5
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST5, DISTM
                MOV   ANGULO, #6
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST6, DISTM
                MOV   ANGULO, #7
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST7, DISTM
                MOV   ANGULO, #8
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST8, DISTM
```

```
                MOV   ANGULO, #9
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST9, DISTM
                MOV   ANGULO, #10
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST10, DISTM
                MOV   ANGULO, #12
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST12, DISTM
                RET
LER_RADAR_H:    CALL  PARAR
                MOV   ANGULO, #12
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST12, DISTM
                MOV   ANGULO, #10
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST10, DISTM
                MOV   ANGULO, #9
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST9, DISTM
                MOV   ANGULO, #8
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST8, DISTM
                MOV   ANGULO, #7
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST7, DISTM
                MOV   ANGULO, #6
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST6, DISTM
                MOV   ANGULO, #5
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST5, DISTM
                MOV   ANGULO, #4
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST4, DISTM
                MOV   ANGULO, #3
                CALL  DELAY01s
                CALL  CALC_DIST
                MOV   DIST3, DISTM
                RET

ESC_RADAR:       MOV   MILHAR, #20H
                MOV   A, #0C0H
                CALL  ESC_COM
                MOV   NUMERO, DIST_E
                CALL  CONVERTE
                CALL  MOSTRA_NUM
                MOV   A, #0C6H
                CALL  ESC_COM
                MOV   NUMERO, DIST_F
```

```
                CALL   CONVERTE
                CALL   MOSTRA_NUM
                MOV    A, #0CCH
                CALL   ESC_COM
                MOV    NUMERO, DIST_D
                CALL   CONVERTE
                CALL   MOSTRA_NUM
                RET

MEDIA           MOV    A, DIST6
                ADD    A, DIST8
                RRC    A
                ADD    A, DIST7
                RRC    A
                MOV    DIST_F, A
                MOV    A, DIST3
                ADD    A, DIST5
                RRC    A
                ADD    A, DIST4
                RRC    A
                MOV    DIST_D, A
                MOV    A, DIST9
                ADD    A, DIST12
                RRC    A
                ADD    A, DIST10
                RRC    A
                MOV    DIST_E, A
                RET

PROCESSAR:      MOV    A, DIST_E
                CLR    C
                SUBB   A,#CONST1
                JC     Eme
Ema:            MOV    A, DIST_F
                CLR    C
                SUBB   A,#CONST1
                JC     EmaFme
EmaFma:         MOV    A, DIST_D
                CLR    C
                SUBB   A, #CONST1
                JC     EmaFmaDme
EmaFmaDma:      CALL   FRENTE
                JMP    FIM_PROC
EmaFmaDme:      CALL   GIRAR_AH45
                JMP    FIM_PROC
EmaFme:         MOV    A, DIST_D
                CLR    C
                SUBB   A, #CONST1
                JC     EmaFmeDme
EmaFmeDma:      MOV    A, DIST_E
                CLR    C
                SUBB   A, DIST_D
                JC     EmaFmeDma1
                CALL   GIRAR_AH90
                JMP    FIM_PROC
EmaFmeDma1:     CALL   GIRAR_H90
                JMP    FIM_PROC
EmaFmeDme:      CALL   GIRAR_AH90
                JMP    FIM_PROC
Eme:            MOV    A, DIST_F
                CLR    C
```

```
                SUBB   A, #CONST1
                JC     EmeFme
EmeFma:         MOV    A, DIST_D
                CLR    C
                SUBB   A, #CONST1
                JC     EmeFmaDme

EmeFmaDma:      CALL   GIRAR_H4 5
                JMP    FIM_PROC

EmeFmaDme:      CALL   PARAR
                JMP    FIM_PROC

EmeFme:         MOV    A, DIST_D
                CLR    C
                SUBB   A, #CONST1
                JC     EmeFmeDme

EmeFmeDma:      CALL   GIRAR_H90
                JMP    FIM_PROC

EmeFmeDme:      CALL   PARAR

FIM_PROC:       RET

PARAR:          CALL   DES_D
                CALL   DES_E
                RET

FRENTE:         MOV    V_D, #100
                MOV    V_E, #100
                CALL   LIG_D_F
                CALL   LIG_E_F
                MOV    A, DIST_F
                CLR    C
                SUBB   A, #CONST2
                JC     FRENTE1
                CALL   DELAY01s

FRENTE1:        CALL   DELAY01s
                 RET

GIRAR_H45:      MOV    V_D, #100
                MOV    V_E, #100
                CALL   LIG_D_T
                CALL   LIG_E_F
                CALL   DELAY01s
                RET

GIRAR_H90:      MOV    V_D, #100
                MOV    V_E, #100
                CALL   LIG_D_T
                CALL   LIG_E_F
                CALL   DELAY01s
                CALL   DELAY01s
                RET

GIRAR_AH45:     MOV    VD, #100
                MOV    VE, #100
                CALL   LIG_D_F
                CALL   LIG_E_T
```

```
                CALL   DELAY01s
                RET

GIRAR_AH90:     MOV    V_D, #100
                MOV    V_E, #100
                CALL   LIG_D_F
                CALL   LIG_E_T
                CALL   DELAY01s
                CALL   DELAY01s
                RET

MENS_1:          DB    "ESQ.   FRENTE   DIR.", 0
MENS_2:          DB    "   ", 0

END
```