UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

ROBERTO FERMINO MEDEIROS

PROTÓTIPO DE UM ROBÕ AUTÔNOMO PARA AUTOMAÇÃO DE TRANSPORTE DE MATERIAIS EM AMBIENTE CONTROLADO

CRICIÚMA

2013

ROBERTO FERMINO MEDEIROS

PROTÓTIPO DE UM ROBÕ AUTÔNOMO PARA MOVIMENTAÇÃO E ORIENTAÇÃO EM AMBIENTE CONTROLADO

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Sergio Coral

CRICIÚMA

2013

ROBERTO FERMINO MEDEIROS

PROTÓTIPO DE UM ROBÔ AUTÔNOMO PARA MOVIMENTAÇÃO E ORIENTAÇÃO EM AMBIENTE CONTROLADO

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Sistemas Operacionais.

Criciúma, 24 de Junho de 2013.

BANCA EXAMINADORA

Prof. Sérgio Coral - Especialista - (UNESC) - Orientador

Prof. Christine Vieira Scarpato - Mestre - (UNESC)

Prof. Gustavo Bisognin - Mestre - (UNESC)

AGRADECIMENTOS

Primeiramente agradeço a Deus por ter me dado força, motivação e a oportunidade de realizar esse trabalho.

A minha mãe Terezinha por estar sempre do meu lado, acreditando nos meus sonhos, pelos valores ensinados, compreensão e amor eterno.

Ao professor Especialista Sergio Coral por ter aceitado me orientar, pelos ensinamentos, pelos conselhos valiosos sempre que precisei.

E especialmente a minha esposa Caroline por ter acreditado em min quando eu não acreditava, por ter chamado atenção quando estava distraído, por ter acreditado nas minhas idéias, por estar sempre ao meu lado, sempre me dando carinho e amor que precisava.

Enfim, todos que direta ou indiretamente contribuíram para a realização desse trabalho.

OBRIGADO!

“O único lugar onde o sucesso vem antes do trabalho é no dicionário.”

Albert Einstein

RESUMO

A robótica já existe há muito tempo, tentando ajudar o homem em atividades que ofereça riscos ou muito desgastantes, robôs já são muito utilizados na linha de montagem de automóveis para diminuir os custos e aumentar a produção, assim como a área militar também já fez muitos avanços na área da robótica. Diante a este cenário, pretende-se desenvolver um protótipo de um robô autônomo utilizando a tecnologia arduino, para utilização na automação industrial.

Palavras-chave: Robótica. Protótipo Autônomo. Sistemas Embarcados. Arduino.

ABSTRACT

Robotics has been around for a long time, trying to help the man in activities that offer risk or very stressful, robots are already widely used in automobile assembly line to reduce costs and increase production as well as the military has also made many advances in robotics. Given this backdrop, we intend to develop a prototype of an autonomous robot using arduino technology for use in industrial automation.

Palavras-chave: Robotics. Standalone prototype. Embedded Systems. Arduino.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Hoje a computação está em todos os lugares, está em empresas, nas casas, nos carros, em uma infinidade de computadores e produtos. Uma área que se destacou muito nos últimos tempos foi a robótica, hoje existem muitas formas de robôs, que auxiliam na indústria, segurança e pesquisas aeroespaciais. Pesando isso imaginou para esse trabalho a construção um protótipo de um robô para auxilio de transporte de carga dentro da área de deposito de uma indústria.

Esse protótipo traria mais segurança e eficiência para a empresa e para os colaboradores envolvidos na operação. Para isso será necessário entender os componentes básicos da arquitetura dos computadores e sistemas embarcados.

A tecnologia Arduino é muito utilizada na área da robótica, pois une desempenho e baixo consumo energético e baixo custo. O Arduino conta com seu próprio processador e memória com um computador convencional, será analisada em detalhes mais adiante.

Pensa-se que a robótica é uma tecnologia nova, que surgiu junto com o computador, mas essa tecnologia surgiu há muito tempo atrás com os gregos, passando pelos árabes até chegar nós dias atuais. As invenções criadas naquela época não se assemelhando com os robôs criados na atualidade, mas que possuíam utilidades, mesmo sem a tecnologia que dispomos hoje.

O desejo do ser humano em criar seres mecânicos para auxiliar em tarefas difíceis, com foi mencionado, já vem de muito tempo atrás e continua nos dias atuais, pois desenvolve-se robôs até mesmo para explorar outros planetas, como é o caso dos robôs que exploram o planeta Marte. Como enviar um ser humano tão longe do planeta seria muito perigoso e caro, foi escolhido enviar um robô. A oportunidade de substituir um ser humano por um robô em situações de alta periculosidade já está sendo utilizada a muito tempo, como nas empresas automobilísticas, para soldar as pesas em um linha de montagem, operando sem a presença humana.

1.1 OBJETIVO GERAL

Este trabalho tem como objetivo geral desenvolver um protótipo de um robô para uso na área industrial, sem a interação humana, de forma, contribuindo para segurança e produtividade.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos, que serem abordados nesse trabalho, os principais são:

a) compreender as principais dificuldades no desenvolvimento de um protótipo;

b) estudar a tecnologia arduino e suas limitações;

c) estudar a comunicação entre diferentes tecnologias;

d) desenvolver uma aplicação que recebe e envia dados para o Arduino;

e) Desenvolver um protótipo de um robô autônomo.

1.3 JUSTIFICATIVA

O homem já busca construir robôs há muito tempo, por exemplo, em 1495 Leonardo da Vinci projetou um leão mecânico que caminhava e apresentava flores para homenagear o rei da França.

A construção de robôs é um sonho muito antigo da humanidade, embora somente nos dias atuais a área da robótica tinha ganhada a atenção da sociedade.

A robótica usa conceitos de diversas outras áreas como a engenharia elétrica, física, biologia, neurologia, entre outras (Gonçalves, 2005), para construir robôs melhores e mais adaptados as suas tarefas, para qual foi construído.

Uma área muito importante que a robótica também abrange é a ciência da computação, pois com o surgimento do computador possibilitou um avanço muito grande na área da robótica.

Hoje há robôs em muitas áreas diferentes como: automação industrial, em pesquisas espaciais, área militar, entre outros. Robôs são muito utilizados na área industrial, na linha de montagem de diversas linhas de produção, garantindo segurança e produtividade.

Essa tecnologia ainda tem um custo muito levado, impossibilitando que muitas empresas tenham acesso. Com a utilização da tecnologia Arduino irá baixar os custos de fabricação.

Diante disso, justifica-se a importância desse trabalho devido a necessidade da construção de robôs de baixo custo, para que pequenas e medias empresas possam ter acesso a esse nível de automação.

1.4 ESTRUTURA DO TRABALHO

Este trabalho é composto de 9 capítulos, sendo que o primeiro apresenta o tema, o objetivo geral, objetivos específicos, justificativa e estrutura do trabalho.

No Capítulo 2 há uma introdução sobre o a arquitetura interna de um computador, conceitos sobre o processador, memória primária e secundária.

O Capítulo 3 apresenta uma introdução sobre as duas arquiteturas básicas de processadores, RISC e CISC.

As famílias dos microcontroladores PIC serão apresentadas no Capítulo 4, essa tecnologia tem como principal característica o baixo custo e pouco consumo de energia, a base para a tecnologia Arduino e a tecnologia PIC.

O Capítulo 5 apresenta a tecnologia ARM e o conceito de sistemas embarcados que também contribuíram para a tecnologia arduino.

No Capítulo 6 será abordado a história da robótica, desde quando os gregos começaram a construir os primeiros experimentos, passando pelos árabes e até chegar na nossa era.

O Capítulo 7 apresenta os trabalhos correlatos, dentre os quais destaca-se um mini robô móvel, seguidor de pistas guiado por visão local. Foi um trabalho desenvolvido no Centro Universitário da FEI em São Bernardo do Campo, na cidade de São Paulo.

Todas as etapas de desenvolvimento do protótipo estão descritas no capítulo 8, incluído a metodologia e os resultados obtidos.

No Capítulo 9 será apresentado a conclusão do trabalho e os trabalhos futuros.

2 ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

Antes de tudo precisamos definir o que é arquitetura e que organização de computador. Podemos definir arquitetura como atributos de um sistema visíveis ao programador que possuem impacto direto na lógica de um programa (TANENBAUM, 2007), como mecanismos de Entrada e Saída (E/S) de dados, número de bits usados para representar tipos de dados (por exemplo, números, caracteres) e técnicas de endereçamento de memória (STALLINGS, 2010). A organização de computador seriam os atributos não visíveis ao programador, como sinais de controle, interfaces entre computadores e tecnologia de memória utilizada (TANENBAUM, 2007).

2.1 ORGANIZAÇÃO DE COMPUTADORES

Segundo Tanenbaum (2007) um computador digital consiste em um sistema interconectado de processadores, memórias e dispositivos de E/S. Serão visto os três componentes com mais detalhes.

2.2.1 PROCESSADORES

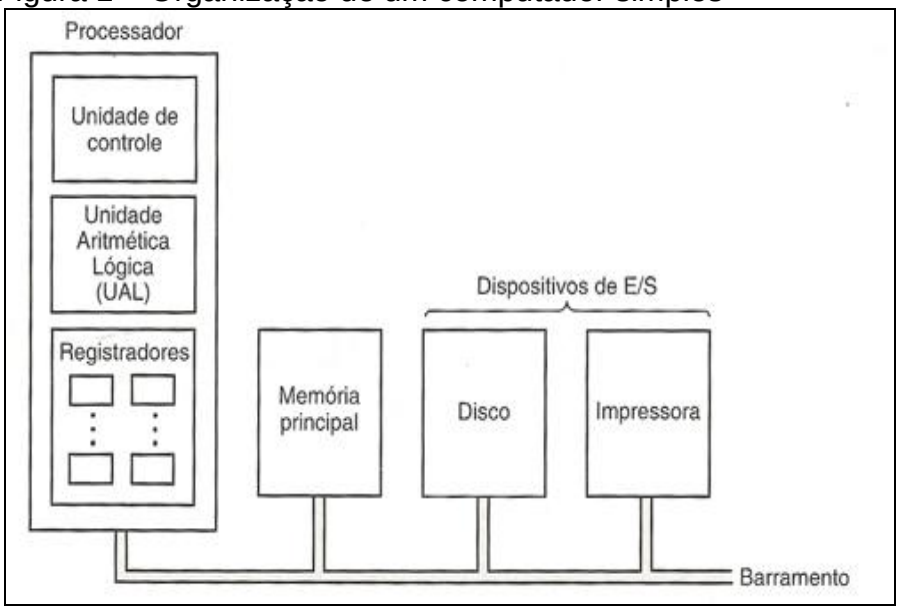
A função de um processador é executar programas armazenados na memória principal, buscar suas instruções necessárias, examinar e depois executar uma de cada vez. (TANENBAUM, 2007) Todos os componentes são conectados por barramentos, que é um conjunto de caminhos compartilhados responsáveis por levar sinais, dados e endereços. (MURDOCA, 2000)

Os barramentos são compartilhados por muito dispositivos, porém se dois dispositivos utilizarem o mesmo barramento, os sinais serão sobrepostos e ficarão distorcidos. Dessa forma os dados não chegaram ao destino de forma limpa e clara (STALLINGS, 2010).

O processador ainda conta com os registradores, que são pequenas memórias que armazenam resultados temporariamente (MURDOCA, 2000). Cada registrador possui uma função especifica e normalmente tem o mesmo tamanho. Por mais que sejam pequenos, podem ser lidos e escritos em alta velocidade (TANENBAUM, 2007).

A Unidade Aritmética e Lógica (UAL) é responsável por realizar todos os cálculos lógicos e aritméticos sobre os dados no computador (STALLINGS, 2010). Todos os outros componentes do computador existem para trazer dados para UAL e depois levar os resultados ao seu destino. A figura 1 mostra a organização de um computador simples.

Figura 1 – Organização de um computador simples



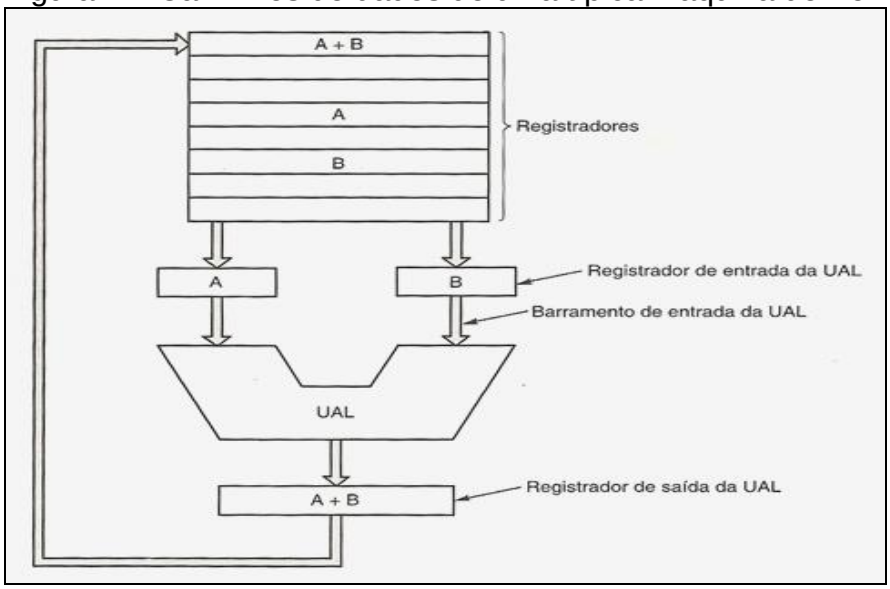
Fonte: Tanenbaum (2007)

A UAL está localizada dentro do processador, conectada com o resto dos componentes do mesmo (MURDOCA, 2000). Os dados são levados a UAL através dos registrados e seus resultados são gravados temporariamente em registradores diferentes, que posteriormente será enviado para a memória, se desejado.

A UAL executa basicamente operações de adição, subtração e outras operações simples sobre suas entradas de dados (TANENBAUM, 2007). A entrada de dados na UAL resulta em uma saída que é armazenada em outros registradores e posteriormente gravada na memória.

Na figura 2 temos um exemplo de uma operação de adição de dois operandos (A e B) nos registradores. O processo de trazer dois operando para UAL, fazer alguma operação com eles e depois armazenar nos registradores é denominada ciclo de caminho de dados e é o coração da maioria dos processadores (Tanenbaum, 2010). Quanto mais rápido for o ciclo de dados, mais rápido a máquina vai funcionar.

Figura 2 – Caminhos de dados de uma típica máquina de Von Neumann



Fonte: Tanenbaum (2007)

2.2.1 Máquina de Von Neumann

Na figura 2 vemos uma exemplo típico de uma máquina idealizada por Von Neumann, que foi um dos consultores do projeto ENIAC, o primeiro computador de uso geral (STALLINGS, 2010).

O ENIAC foi construído entre 1943 e 1946, na Universidade da Pensilvânia. Uma das principais motivações para realização desse projeto, foi a necessidade de construir tabelas automáticas para o exercito americano (WEBER, 2004).

O conceito idealizado por Vou Neumann ficou conhecido como conceito de programa armazenado. A máquina em questão deveria ter os seguintes componentes:

a) uma memória principal, que armazena os dados e instruções;
b) uma UAL capaz de operar sobre dados binários;
c) uma unidade de controle, que interpreta as instruções na memória e faz com que sejam executadas;
d) equipamento de E/S operado pela unidade de controle.

Em 1946 foi iniciada a construção de outro computador de programa armazenado, o IAS feito pelo próprio Von Neumann (WEBER, 2004).

O IAS possuía 1 000 locais de armazenagem em sua memória, chamadas de palavras, de 40 dígitos binários. Em geral uma palavra é um conjunto ordenado

de bytes e bits, que é a unidade normal na qual a informação pode ser armazenada, transmitida ou operada (STALLINGS, 2010).

Tanto dados e instruções eram armazenados em palavras, um número era representado com um bit de sinal e um valor de 39 bits. Ainda uma palavra podia conter duas instruções de 20 bits, cada instrução possuía 8 bits para o código da operação a ser realizada (opcode) e 12 bits de endereço, onde era referenciada uma das palavras na memória(0 a 999) (STALLINGS, 2010).

Hoje com raras exceções, todos os computadores utilizam essa estrutura, claro que as máquinas atuais estão muito mais poderosas que o IAS, mas seguem a mesma estrutura e função geral, por isso são chamadas de máquinas de Von Neumann.

2.2.2 Execução de Instruções

O processador ou Central Processing Unit (CPU) executa cada instrução seguindo algumas etapas:

a) trazer a próxima instrução da memória para os registradores.
b) alterar o contador de programa para indicar a próxima instrução.
c) determinar o tipo de instrução trazida.
d) se a instrução usar uma palavra na memória, determinar onde essa palavra está.
e) trazer a palavra para dentro de um registrador da CPU, se necessário.
f) executar a instrução.
g) voltar a etapa 1 para iniciar a execução da instrução seguinte.

Cada programa deve aguardar sua vez para ser executado no processador, enquanto não é carregado para o processador fica aguardando na memória primária (MURDOCA, 2000).

Essa seqüência de etapas de execução é denominada de “buscar-decodificar-executar” e é fundamental para a operação dos computadores (TANENBAUM, 2007).

Ainda há programas que simulam esse processo realizado pela CPU, esses programas são chamados de “interpretadores”. Com um interpretador um

programa pode deixar que outro programa busque, examine e execute suas instruções, esse é o caso na linguagem Java, que conta com um interpretador para executar os programas escritos na linguagem objeto (TANENBAUM, 2007).

Os interpretadores são muito importantes na organização dos computadores e em novos projetos de sistemas de computadores, pois responde mais rápido as modificações no código fonte em relação com as linguagens compiladas (WEBER, 2004). Um interpretador subdivide as instruções de máquina a serem executadas em pequenas etapas, dessa forma a máquina onde o interpretador irá rodar pode ser muito mais simples e barata que um processador construído para essa mesma máquina (TANENBAUM, 2007).

Uma das principais vantagens dos interpretadores é a retirada da complexidade dos processadores, tornando os processadores mais simples e com custo baixo, deixando toda a complexidade para o interpretador. Dessa forma o projeto complexo de *hardware* se tornou um projeto complexo de *solftware* (TANENBAUM, 2007).

Um exemplo de sucesso do uso de interpretadores foi o Motorola 6800, que possuía uma grande quantidade de instruções interpretadas (TANENBAUM, 2007).

Computadores simples, com instruções interpretadas, possuem mais vantagens. As mais importantes são:

a) a capacidade de corrigir instruções em tempo de execução instruções implementadas incorretamente ou até compensar deficiências de projeto no hardware básico.
b) a oportunidade de acrescentar novas instruções a custo mínimo.
c) projeto estruturado que permitia desenvolvimento, teste e documentação eficiente de instruções complexas.

2.3 MEMÓRIA

A memória é a parte do computador que são armazenados os programas e dados. Sem a memória o processador não teria como ler e gravar informações, sem a memória não haveria computadores digitais com programas armazenados (TANENBAUM, 2007).

A memória de um computador está divida em dois tipos, a memória principal, onde são armazenados os dados e instruções que o processador irá utilizar e a memória secundária onde são armazenados os programas e os dados do usuário.

2.3.1 Bits

A unidade básica de memória é o digito binário, denominado bit (Tanenbaum, 2007). Um bit pode assumir somente dois valores, 0 ou 1.

Como um digito binário necessita apenas diferenciar entre dois valores, é o método mais simples e eficiente de codificar informações digitais. Pois quantos mais valores a memória precisar diferenciar menos eficientes serão e suscetíveis a mais erros (CARTER, 2003).

2.3.2 Memória Principal

Como já foi mencionada, a memória principal é o dispositivo eletrônico onde é armazenado os programas, instruções e dados que o processador irá precisar para realizar sua função.

A memória principal é uma memória de acesso aleatório (RAM), esse tipo de memória é caracterizado por possibilitar a leitura e escrita de dados (STALLINGS, 2010). Memórias RAM são dividas em dois subgrupos denominadas de memórias RAM Dinâmica (DRAM) e memória RAM estática (SRAM).

2.3.2.1 DRAM

Memórias RAM Dinâmicas (DRAM) possuem células individuais que armazenam apenas 1 bit, as células armazenam dados como cargas em capacitores. A presença de carga é representada como 1 e a ausência é representada como 0 (STALLINGS, 2010).

Para a operação de escrita um sinal de tensão é aplicada na linha bit, uma tensão alta representa 1 e um tensão baixa representa 0. Em seguida é aplicado um sinal tensão na linha de endereço permitindo a transferência de tensão para os capacitores. Como os capacitores possuem a tendência natural de perder

carga, é necessário que se tenha uma realimentação constante dos capacitores, esse processo é chamado de “refresh” de memória (STALLINGS, 2010).

Esse tipo de memória é volátil, enquanto existir fornecimento de energia para o dispositivo, os dados serão preservados, a partir do momento que o fornecimento é interrompido os dados são perdidos.

2.3.2.1 SRAM

São memórias RAM estáticas (SRAM), ou seja, enquanto houver fornecimento de energia os dados não serão apagados (STALLINGS, 2010). Como nas memórias DRAM as memórias SRAM usam a linha de endereço para gravar ou ler os dados. A linha de endereço controla dois transistores, quando uma tensão é aplicada na linha os transistores são ligado, permitindo assim a leitura e gravação dos dados.

Assim como a memória DRAM, as memórias SRAM necessitam de energia continua para manter os bits armazenados nas células (STALLINGS, 2010).

2.3.2.1 Endereços de memória

A memória possui uma quantidade de células também chamadas de endereço, cada célula (endereço) pode conter uma informação. Cada célula possui um número, denominado seu endereço, para que programas possam se referir a ela (TANENBAUM, 2007, p. 40).

Se uma memória tiver n células, os endereços irão de 0 até n – 1, o numero de bits no endereço determina a quantidade de células de uma memória diretamente endereçáveis de uma memória e é independente do número de bits que uma célula pode armazenar, ou seja, uma memória com 2^{12} de 8 bits cada e outra com também 2^{12} de 64 bits necessitam de um endereço de 12 bits. Dessa forma se o endereço contiver **m** bits, a quantidade máxima de células endereçáveis será 2^{M} .

2.3.3 Memória Secundária

Não importa o qual seja o tamanho da memória principal, sempre será menor que a memória secundária (TANENBAUM, 2007). A memória principal é

encarregada de armazenar os programas e dados em tempo de execução, quando é necessário armazenar um arquivo de texto, por exemplo, é usada a memória secundária. Essa memória é muito mais lenta que a memória principal, porem pode gravar os dados por muito tempo.

2.3.3.1 Discos magnéticos

Segundo Tanenbaum (2007) um disco magnético é composto de um ou mais pratos de alumínio com um revestimento magnetizável. No começo esses pratos eram de 50cm de diâmetro e não tinham muita capacidade de armazenamento, hoje esse pratos tem de 3cm a 12cm e sua capacidade de armazenamento aumentou significantemente.

O cabeçote de disco é encarregado de fazer a leitura e gravação no prato, o cabeçote flutua sobre uma superfície do disco, apoiado sobre um colchão de ar. Quando uma corrente positiva ou negativa passa pelo cabeçote, magnetiza a superfície do disco, alinhando as partículas magnéticas para esquerda ou para direita, dependendo da corrente (TANENBAUM, 2007). Dessa forma os dados são gravados no prato possibilitando a leitura dos dados mais tarde.

O cabeçote grava e lê os dados sobre o prato que gira por baixo, isso significa que os dados são organizados no prato por um conjunto concêntrico de anéis, chamados de trilhas (STALLIGNS, 2010). Isso quer dizer que o cabeçote fica parado enquanto o prato se move por baixo dele, formando assim anéis com dados gravados neles.

Cada trilha é dividida em setores onde são armazenados os dados, existem centenas de setores por trilha. Os tamanhos dos setores podem ser fixos ou variados, mas em sistemas modernos estão sendo usados 512 bytes (STALLIGNS, 2010).

Os dados gravados na parte mais perto do centro do disco são lidos mais lentamente que os mais afastados do centro. Para correção desse problema foi aumentado o espaço entre os bits nos segmentos do disco, dessa forma o disco pode girar em uma velocidade fixa, conhecida por Velocidade Angular Constante (CAV) (STALLINGS, 2010).

A densidade, em bits por polegada linear, aumenta na passagem da trilha mais externa para a mais interna, a capacidade de armazenamento em um sistema

CAV é limitada pela densidade de gravação máxima. Para aumentar à densidade e consequentemente a capacidade de armazenamento os discos modernos utilizam uma técnica chamada de gravação em múltiplas zonas, que consiste em dividir toda a superfície do disco em várias zonas, normalmente o número de zonas é 16 (STALLINGS, 2010).

O número de bits em uma zona é constante, mas as zonas mais externas do disco possuem mais bits que as zonas mais internas. Essa técnica aumenta a capacidade, mas possui um custo. Como o número de bits pode mudar de uma zona e outra, o tempo de leitura e gravação muda também (STALLINGS, 2010).

Para determinar o inicio e o fim das trilhas e dos setores é usado um dado de controle gravado no disco, esses dados são acessados pela unidade de disco e não são visíveis pelos usuários.

Cada disco é composto de vários pratos, e cada prato conta com um cabeçote e braço individual. Os braços são agrupados de modo que possam se movimentar para diferentes posições radiais ao mesmo tempo (TANENBAUM, 2007).

Segundo Tanembaum (2007) o conjunto de trilhas em uma dada posição radial é denominado de cilindro. Os discos usados hoje em PCs normalmente têm 12 pratos por drive, o que resulta em 12 a 24 superfície de gravação.

Para ler ou escrever um setor, o braço precisa se posicionar na posição radial correta. Essa ação é denominada de busca, o tempo médio entre trilhas aleatórias está entre 5 a 10ms, esse tempo baixa muito quanto ocorre em trilhas consecutivas que é de 1ms. Quando o braço está posicionado radialmente, ocorre um atraso que é denominado de latência rotacional, até que o setor desejado gire por baixo do cabeçote. A maioria dos discos gira em 5.400 RPM ou 10.800 RPM, dessa forma o atraso médio é de 3 a 6 ms. Com o tempo de transferência de dados típicas de 20 a MB/s, um setor de 512 bytes demora entre 13 e 26 ms. Dessa forma o tempo de busca e latência rotacional domina o tempo de transferência (TANENBAUM, 2007).

Hoje os drives possuem os cilindros divididos em zonas (10 a 30 por drive) e o número de setores por trilha aumenta de zona em zona, partindo da trilha mais interna até a mais externa. Essa técnica dificulta o rastreamento das informações mais aumenta a capacidade de armazenamento do drive, que é considerado mais importante (TANENBAUM, 2007).

Junto com cada drive está o controlador de disco, um chip que controla o drive. Algumas tarefas que o controlador realiza é aceitar comandos de Softwares, como comandos de leitura e gravação de dados, controlar o movimento dos braços, detectar e corrigir erros.

2.3.3.2 Disquete

Com a chegada dos computadores pessoais surgiu a necessidade da distribuição de software, até então não existia a internet como nós conhecemos hoje para conectar os computadores. A solução encontrada foi o disquete, ou disco flexível, era um meio removível, pequeno e de baixo custo (TANENBAUM, 2007).

A IBM criou o disco flexível para armazenar informações de manutenção de seus mainframes para o seu pessoal de serviço, mas os fabricantes de computadores logo viram o potencial do disco flexível para a venda de software. O disco flexível ganhou essa denominação pelo fato de ser fisicamente flexível (TANENBAUN, 2007).

2.3.3.3 Discos IDE

Os discos modernos dos computadores pessoais evoluíram do disco usado no IBM PC XT, era um disco Seagate de 10 MB controlado por um controlador de disco Xebec em um cartão de encaixe (TANENBAUM, 2007). O sistema operacional lia e escrevia no disco colocando parâmetros em registradores da CPU e logo em seguida chamava a Basic Input Out System – sistema básico de entrada e saída (BIOS) localizada na memória somente leitura da CPU, então as BIOS realizavam a gravação no disco (TANENBAUM, 2007).

Como a evolução da tecnologia o controlador passou de um placa separada do drive para uma placa integrada, assim em 1980 surgiu o drive Integrated Drive Eletronics – eletrônica de drives integrados (IDE). Porem as convenções da chamada do BIOS não foram alteradas por causa da compatibilidade (TANENBAUM, 2007).

Antes o máximo que os drives podiam chegar era 504 MB, com o tempo essa marca foi ultrapassada, mas com a geometria errada. O sistema operacional não conseguia endereçar o drive. Para corrigir esse o problema os controladores de

disco começaram a remapear a geometria virtual para o real, dessa forma parecia que a geometria estava dentro da geometria da BIOS (TANENBAUM, 2007).

Então surgiram os drives EIDE que também possuíam suporte a um segundo esquema de endereçamento chamado de Logical Block Addressing – endereçamento de blocos lógicos (LBA). Esse esquema requer que o controlador converta endereços de LBA para endereços de cabeçote, cilindro e setores, ultrapassando o limite 504 MB. Em 1994 quando adotado esse padrão não se imaginava disco muito maiores que 128 GB, esse era o limite para os drives EIDE (TANENBAUM, 2007).

Os sucessores do padrão EIDE foi denominado ATA-3, depois chegaram o ATAPI-4 e ATAPI-5. O limite de 128 GB imposto pelo LBA de 28 bits estava cada vez mais próximo, então com o ATAPI-6 aumentou o tamanho do endereço LBA para 48 bits, assim o limite para os drives passou de 128 GB para 128 PB, esse limite deve durar até 2035 (TANENBAUM, 2007).

O ATAPI-7 rompeu com o passado, em vez de aumentar o tamanho do conector do drive, esse padrão usa o modelo ATA serial, que consiste passar um 1 bits por vez por um cabo de 7 pinos. Substituir o cabo tradicional de 80 fios por um cabo menor tem outra vantagem, o ATA serial usa apenas 0,5 volts para sinalização enquanto os cabos tradicionais usam 5 volts (TANENBAUM, 2007).

Com o tempo espera-se que todos os computadores usem drives com ATA serial, principalmente os notebooks pela vantagem do consumo de energia.

2.3.3.4 CD-ROM

Os discos ópticos foram criados pela Philips para gravar programas de televisão, mas são muito utilizados para distribuição de software, livros, filmes e dados de diversos tipos, sendo muito utilizado em *backup* de disco rígido (TANENBAUM, 2007).

Em 1980 a Philips junto com a Sony desenvolveu o Compact Disc (CD), substituindo o disco de vinil para gravações de musicas. As dimensões dos CDs são 120 mm de diâmetro e 1,2 mm de espessura, essa especificação é mantida até hoje.

Um CD é preparado utilizando um disco mestre revestido com vidro, um laser de alta potencia realiza pequenos furos no disco mestre. Com base no disco mestre é feito um molde, com pequenas saliências onde estavam os furos nos disco

mestre, depois é aplicado policarbonato sobre o molde para formar o CD. Por fim o CD ganha uma camada de alumínio e uma etiqueta (TANENBAUM, 2007).

As marcas na superfície do policarbonato são denominadas depressões (pits) e as áreas planas são denominados planos (lands). Para realizar a leitura do disco é emitido um laser de baixa potencia, quando é reconhecida a passagem de depressão/plano ou plano/depressão é reconhecido como 1 e sua ausência é reconhecida como 0 (TANENBAUM, 2007).

O CD-R ou CD graváveis, hoje muito utilizados por usuários normais para gravar dados de diversos tipos, tem a sua gravação e leitura parecidas, mas com algumas diferenças.

Além da camada de policarbonato e de alumínio, os CDs graváveis ganham uma camadas de corante entre essa camadas. Quando um laser é aplicado no CD para gravar, rompe a ligação química do corante, deixando uma mancha no local. Então quando for realizada a leitura essas manchas serão interpretadas como depressões e a ausência da mancha será interpretada como planos (TANENBAUM, 2007).

Hoje os CDs são utilizados com uma memória secundária de armazenamento, armazenando os dados enquanto o CD não for danificado, dessa forma é muito utilizada para gravar dados importantes como fotos, documentos, cópia de segurança de disco rígido, entre outros.

2.3.4 Memória cache

A memória cache é a memória mais perto do processado, em relação com a memória principal e a memória secundária. Os registradores do processador são memórias muito rápidas, mas possuem pouca capacidade de armazenamento e seu custo de fabricação é alto (STALLINGS, 2010).

O principal motivo do uso da memória cache é obter velocidade semelhante às memórias mais rápidas (registradores), e ao mesmo tempo, fornecer uma memória de grande capacidade e com baixo custo (STALLINGS, 2010).

A CPU sempre foi mais rápida que a memória principal, isso é um problema, pois a CPU terá que aguardar vários ciclos de CPU até a palavra chegar (TANEMBAUM, 2007). Para resolver esse problema foi criado a memória cache, palavras que são referenciadas na memória principal que é mais lenta ficam

armazenadas na memória cache, quando a CPU requisitá-las novamente não irá precisar ir até a memória principal, simplesmente pegará da memória cache que é mais rápida que a memória principal (STALLINGS, 2010).

A memória cache possui vários níveis, sendo os principais L1, L2 e L3, a capacidade e velocidade variam entre os níveis. O nível L1 é mais rápido e menor que o L2 e L3 (STALLINGS, 2010).

Enquanto a memória principal é divida em blocos, a memória cache é dividia em linhas de 4 a 64 bytes, essas linhas são numeradas começando de 0 (TANENBAUM, 2007). Quando a memória é referenciada, o circuito que controla o cache verifica se a palavra está na cache, se não estiver procura nos níveis inferiores de cache, se encontrar inclui na cache, senão vai até a memória principal busca a palavra.

A memória cache é cada vez mais necessária em computadores de alto nível, quanto maior a memória cache, maior será o desempenho da memória e também maior será o seu custo (TANENBAUM, 2007).

2.3.5 Entrada/Saída

Todo computador possui três componentes principais: a CPU, as memórias (primária e secundária) e os equipamentos de E/S, que são impressoras, scanners, modens, entre outros (TANENBAUM, 2007).

2.3.5.1 Módulo de E/S

O principal meio de comunicação com o computador é o conjunto teclado/mouse, que são dispositivos de entrada dados, o usuário realiza a entrada de dados pelo teclado ou pelo mouse, que pode ser visualizada no monitor, esse é um dispositivo de saída, essa interação é realizada pelo módulo de E/S (STALLINGS, 2010).

As principais funções do módulo de E/S:

a) controle e temporização.
b) comunicação com o processador.
c) comunicação com o dispositivo
d) armazenamento temporário de dados.

e) detecção de erro.

Segundo Stallings (2010) durante qualquer período, o processador pode se comunicar com um ou mais dispositivos externos em padrões imprevisíveis, dependendo da necessidade de E/S do programa.

Memória principal e o barramento do sistema são compartilhados entre várias atividades, entre elas está a E/S de dados. Por isso o módulo de E/S necessita do requisito de controle e temporização, essa função é requisitada para o módulo poder controlar o tráfego entre os recursos internos e os dispositivos externos (STALLINGS, 2010).

O controle de transferência de dados entre um dispositivo externo para o processador pode envolver as seguintes etapas (STALLINGS, 2010):

a) o processador interroga o módulo de E/S para verificar o estado do dispositivo conectado.
b) o módulo de E/S retorna o estado do dispositivo.
c) se o dispositivo estiver operacional e pronto para transmitir, o processador solicita a transferência de dados ao módulo de E/S.
d) o módulo de E/S obtém uma unidade de dados do dispositivo externo (por exemplo, 8 ou 16 bits).
e) os dados são transferidos do módulo de E/S para o processador.

Uma tarefa muito importante do módulo de E/S é o *buffering* de dados. Geralmente a taxa de transferência da memória e dos dispositivos externos pode ser diferente, se os dados são enviados da memória principal para o módulo de E/S de maneira muito mais rápida que o dispositivo externo. Os dados são armazenados em um buffer no módulo de E/S e depois transferidos para o dispositivo externo na sua própria taxa de transferência, esse procedimento é realizado para não ocupar a memória ocupada em uma operação muito lenta (STALLINGS, 2010).

Do mesmo modo, se a taxa de transferência de dados do dispositivo externo for mais rápido que a taxa de acesso a memória principal, o módulo realiza o *buffering* necessário para igualar a taxa de transferência do dispositivo com o a taxa de acesso a memória (STALLINGS, 2010).

E para finalizar o modulo de E/S é responsável pela detecção de erro, e relatar ao processador quando e o tipo de erro como: papel emperrado, trilha de disco com defeito, pendrive não está mais respondendo, entre outros (STALLINGS, 2010).

3 TECNOLOGIA RISC E CISC

Existem duas arquiteturas de processadores principais, cada uma tem características diferentes e possui aplicações diferentes também.

3.1 RISC

Segundo Stallings (2010) a arquitetura RISC é um grande avanço na tendência na arquitetura de processadores. Embora a arquitetura RISC tenha sido definida e projetada de maneiras diferentes e por grupos diferentes, abaixo estão alguns elementos principais da maioria dos projetos:

a) um grande número de registradores de propósito geral e/ou o uso de tecnologia de compiladores para otimizar uso de registradores.
b) um conjunto de instruções simples e limitada.
c) uma ênfase na otimização do pipeline de instruções.

3.1.1 Características da arquitetura RISC

Como foi mencionado, várias abordagens foram implementadas para a arquitetura RISC, mas algumas características são comuns.

a) uma instrução por ciclo.
b) operações registrador-para-registrador.
c) modos de endereçamento simples.
d) formato de instruções simples.

Um ciclo de máquina é o tempo em obter dois operandos dos registrados, executar uma operação na UAL e depois armazenar o resultado nos registradores. Máquinas RISC possuem instruções simples e de ciclo único, dessa forma as instruções de máquina podem ser embutidas diretamente no hardware, essas instruções deveriam ser executadas mais rapidamente do que em computadores semelhantes, pois não é necessário acessar um microprograma de controle durante a execução da instrução (STALLINGS, 2010).

A segunda característica da arquitetura RISC é que as operações devem ser de registrador-para-registrador, com operações simples de acesso a memória.

Isso simplifica o conjunto de instruções e assim simplifica a unidade de controle também (STALLINGS, 2010).

Outra característica muito importante é o uso de modos de endereçamento simples. Quase todas as instruções RISC usam endereço de registradores simples, essa característica também ajuda a tornar as instruções e a unidade de controle mais simples. Modos de endereçamentos podem ser sintetizados a partir dos mais simples (STALLINGS, 2010).

A última característica é o uso de formatos de instruções simples, são usados um ou alguns formatos. O tamanho da instrução é fixo e ajustado dentro do limite da palavra, a posição dos campos dentro da instrução também é fixa. Esse recurso tem muitos benefícios, com campos fixos a decodificação e o acesso aos registradores podem ocorrer ao mesmo tempo. Formatos simples de instruções assim como as outras características também ajudam a simplificar a unidade de controle (STALLINGS, 2010).

3.2 CISC

Segundo Prado (2011) a arquitetura CISC tem como base a utilização de instruções mais complexas, objetivando assim a diminuição do número de instruções que o programa necessita para ser implementado.

Diferente da arquitetura RISC que tem como principal características, instruções de máquinas mais simples, a arquitetura CISC possui instruções mais complexas para simplificar os compiladores de programas (STALLINGS, 2010).

3.2.1 Características da arquitetura CISC

As principais características da arquitetura são:

a) tamanho variável de instruções de acordo com o endereçamento;
b) instruções que requerem múltiplos ciclos;
c) instrução operando direto na memória;
d) possui um pequeno numero de registradores “gerais”, e muitos registradores específicos.

Dois principais motivos para o desenvolvimento da arquitetura CISC são: simplificar os compiladores e melhorar o desenvolvimento da máquina. Programadores estão migrando para linguagens de alto nível, dessa forma os projetistas desejavam construir máquinas que melhor oferece suporte a essas novas linguagens (STALLINGS, 2010).

Os programadores de compiladores têm a tarefa de gerar seqüência de instruções de máquina para cada instrução da linguagem de alto nível. Então se houver instruções de máquina semelhante às instruções de alto nível, essa tarefa será muito mais simples de ser realizada. Contudo instruções complexas são muito difíceis de serem exploradas porque o compilador precisa achar casos que as instruções de máquina se encaixem perfeitamente (STALLINGS, 2010).

Outra motivação para a arquitetura CISC era que máquinas com essa arquitetura teriam um desempenho melhor. Um dos motivos para essa idéia era que programa CISC seriam menores, então dessa forma ocupariam menos espaço na memória e como as instruções seriam menores, assim menos bytes de instruções para serem obtidos (STALLINGS, 2010).

Isso tudo tinha um problema, um programa CISC em linguagem de máquina era menor que um programa RISC, porém o número de bits em memória não seria muito menor. Além disso todos os componentes internos da CPU devem ser mais complexos, para suportar as instruções mais ricas que as instruções RISC (STALLINGS, 2010).

Segundo Stallings (2010) não está nem um pouco claro que uma tendência pelo aumento da complexidade dos conjuntos de instruções seja apropriada. Isso levou vários grupos a seguirem o caminho oposto.

4 MICROCONTROLADOR

Segundo Souza (2003, p. 21) “Em poucas palavras, podemos definir o microcontrolador como um “pequeno” componente eletrônico, dotado de uma “inteligência” programável, utilizado no controle de processo lógico”.

Microcontroladores diferem de microprocessadores pelo fato de incorporar vários componentes em uma única pastilha que chamamos de “chip”, garantindo baixo custo e alto rendimento.

4.1 APLICAÇÃO

São muitas as aplicações para os microcontroladores, e estão presentes em vários seguimentos do mercado e da vida de todos nós. Segundo Corteletti (2006) no final do século XX, e início do século XXI, muitos equipamentos sofreram evoluções bastante radicais, grande parte graças aos microcontroladores.

Por seu baixo custo e alta performace, os microcontroladores está abrindo bastante espaço no mercado, não só no mundo, mais também no Brasil (TAURION, 2005). Os microcontroladores estão presentes em várias áreas como automação industrial, automação comercial, automação predial, área automobilística, agrícola, produtos manufaturados, eletrodomésticos, telecomunicações, entre muitas outras (CORTELETTI, 2006).

4.2 MICROCONTROLADOR NA INDÚSTRIA AUTOMOBILÍSTICA

Quando entramos em um carro não vemos, mas estão no freio ABS, direção eletrônica, injeção eletrônica de combustível, controle de tração, controle de suspensão e travas elétricas, entre muitos outros (CORTELETTI, 2006). Os microcontroladores estão em muitos lugares para nos trazer eficiência, segurança e conforto.

Estima-se que são produzidos cerca de 63 milhões de carros anualmente no mundo, e cada veículo possui em média 30 microcontroladores. As aplicações então no conforto na segurança e eficiência do veículo (CORTELETTI, 2006).

4.3 MICROCONTROLADOR PIC

Um dos tipos de microcontroladores mais usados é os do tipo PIC, que são conhecidos pela sua capacidade de processamento, custo e flexibilidade de aplicações (CORTELETTI, 2006).

4.3.1 Características PIC

Uma das principais características dos microcontroladores PIC está na simplicidade de aplicação, assim facilitando a implementação e diminuindo os custos de desenvolvimento, permitindo a aplicação em projetos de pequeno porte (CORTELETTI, 2006).

Os microcontroladores PIC são divididos em famílias, cada uma com características relacionada a sua performance e funcionalidade. Temos a família PIC10, de menos recursos, que são muito aplicadas em funções de on-off mais simples e de menos porte, já as famílias PIC30 e PIC33 possuem mais recurso e são aplicadas em projetos mais complexos, essas famílias são muito utilizadas em telecomunicações, rede sem fios de alta performance e controles de tempo real de alta velocidade (CORTELETTI, 2006).

4.4 FAMÍLIAS PIC

Agora vamos ver algumas famílias mais comuns dos microcontroladores PIC, e também algumas características de cada uma.

4.4.1 PIC10F200

São microcontroladores de baixo custo, aproximadamente US$ 0,40, esse microcontrolador possui 4 pinos de E/S, sendo 3 pinos configuráveis como entradas ou saídas, e o 4º pino é somente saída. O processador do microcontrolador é um RISC de 22 instruções de máquina, 12 bits, 2 níveis de pilha, operando em até 4MHZ, com capacidade para 1 milhão de instruções por segundo (CORTELETTI, 2006).

Esse microcontrolador ainda conta com 16 bytes de memória RAM, 12 bits de memória ROM e um contador/temporizador de 8 bits programável.

Segundo Corteletti (2006, p. 8) estas características tornam este microcontrolador bastante aceitável em aplicações simples, como em brinquedos, eletrodomésticos e tratamento de sinal digital de sensores e periféricos.

4.4.2 PIC16F84

A principal característica dessa família é a memória ROM do tipo flash, que facilita a gravação e modificação de programas no microcontrolador, dispensando as antigas gravações eletrônicas que demoravam muitos minutos (CORTELETTI, 2006).

Os microcontroladores PIC16F84 foram os primeiros a utilizar esse recurso, aliado o seu baixo custo, recursos e flexibilidade ajudou a disseminar os microcontroladores (CORTELETTI, 2006).

As outras características dessa família são: processador RISC de 35 instruções, com velocidade de 20 MHZ, 68 bytes de memória RAM e 64 bytes de memória EEPROM.

Segundo Corteletti (2006, p. 9) “hoje, não é recomendado o uso deste microcontrolador, uma vez que sua fabricação tende a ser descontinuada, (atualmente é fabricado como PIC16F84A)”.

4.4.3 PIC16F628

Um microcontrolador relativamente barato e com muitos recursos incorporados, seu custo é de aproximadamente US$ 3,00, o microcontrolador possui as seguintes características: processador RISC de 35 instruções, 16 pinos de E/S, dois comparadores analógicos, comunicação serial, memória RAM de 224 bytes e EEPROM de 128 bytes (CORTELETTI, 2006).

4.4.4 PIC16F877A

Esse é o melhor microcontrolador da família 16, oferecendo suporte para aplicações de controle. Seu custo por unidade está em US$ 4,00, porém esse

microcontrolador oferece alguns recursos que podem eliminar a implementação de outros periféricos (CORTELETTI, 2006).

As mais importantes características são: processador RISC de 35 instruções chegando a 5 milhões de instruções por segundo, 368 bytes de memória RAM, 256 bytes de EEPROM, comunicação de serial e 33 pinos de E/S (CORTELETTI, 2006).

5 TECNOLOGIA ARM

Segundo Stallings (2010, p. 35) “a arquitetura ARM refere-se a uma arquitetura de processadores que evoluiu dos princípios do projeto RISC e é usada em sistemas embarcados”.

5.1 SISTEMAS EMBARCADOS

Podemos definir sistemas embarcados como uma combinação de hardware e software projetada para uma função especifica. Por exemplo, na área médica temos vários dispositivos embarcados, como: bombas de infusão, máquinas de diálise, dispositivos protéticos, monitores cardíacos (STTALLINGS, 2010).

Existente uma variedade muito maior de sistemas embarcados do que sistemas de computadores de uso geral, como laptop ou desktop. Além da área médica, sistemas embarcados existem em diversas outras áreas como: automotivo, eletrônica, controle industrial e automação de escritório (STTALLINGS, 2010).

5.2 EVOLUÇÃO DA ARQUITETURA ARM

A arquitetura ARM é uma família de microprocessadores e microcontroledores baseado em RISC, projetada pela ARM Inc., Cambridge, Inglaterra. A empresa não fabrica processadores, mas sim projeta e licencia os fabricantes (STALLINGS, 2010).

Os chips ARM são processadores pequenos de baixo custo, alta velocidade e baixo consumo enérgico. Eles são muito utilizados em dispositivos portáteis, como telefones, celulares, tablets, muitos outros produtos de consumo (STALLINGS, 2010).

A arquitetura ARM foi criada pela Acorn Computers. Em 1980 a Arcon ganhou o contrato da British Broacasting Corporation (BBC) para desenvolvimento de uma arquitetura de microcomputador para BBC Computer Literacy Project (Prado, 2011). Com esse contrato a Arcon conseguiu desenvolver o primeiro processador RISC comercial, o Arcon RISC Machine (ARM) (STALLINGS, 2010).

A primeira versão foi o ARM1 e começou a operar em 1985, e foi usado como coprocessador na máquina da BBC. Ainda em 1985 a Arcon lançou o ARM2

como maior velocidade em um mesmo espaço físico. Já em 1989 foi lança o ARM3, com inclusão de mais funcionalidades e melhorias (STALLINGS, 2010).

Depois do ARM3 o desenvolvimento, além disso, ficou fora do escopo da Arcon. Então uma nova empresa foi organizada, com Arcon, VLSI e Apple Computer como parceiros fundadores, conhecida como ARM Ltd. O primeiro produto desenvolvido foi o ARM6, uma melhoria sobre o ARM3 (PRADO, 2011).

Os processadores ARM são projetados para atender as seguintes necessidades de sistemas (STALLINGS, 2010):

a) sistemas embarcados de tempo real: sistemas para aplicações de armazenamento, automotivas, industriais e de redes;
b) plataformas de aplicação: dispositivos executando sistemas operacionais, como Linux, Palm OS, Symbian OS e Windows CE em aplicações sem fio, entretenimento e imagens digitais;
c) aplicações seguras: smart cards, placas SIM e terminais de pagamento.

Hoje a tecnologia ARM está presente nas mais variada áreas e está evoluindo cada vez mais, tornando a tecnologia acessível a todos.

5.3 ARDUINO

O objetivo do projeto Arduino é promover uma plataforma fácil para prototipação de projetos interativos. Basicamente as placas Arduino são microconroladores, ele faz parte da computação física, área da computação onde o software interage diretamente com o hardware (JUSTEM, 2010).

5.3.1 Shields

A placa Arduino já vem com vários recursos de fábrica, porém dependendo do projeto a ser implementado é necessário incorporar outros recursos que não vem com o Arduino. As Shields são placas que podem ser conectadas no Arduino que podem adicionar recursos extras. Existem várias Shields diferentes, cada uma com recursos adicionais diferentes ao Arduino (JUSTEM, 2010).

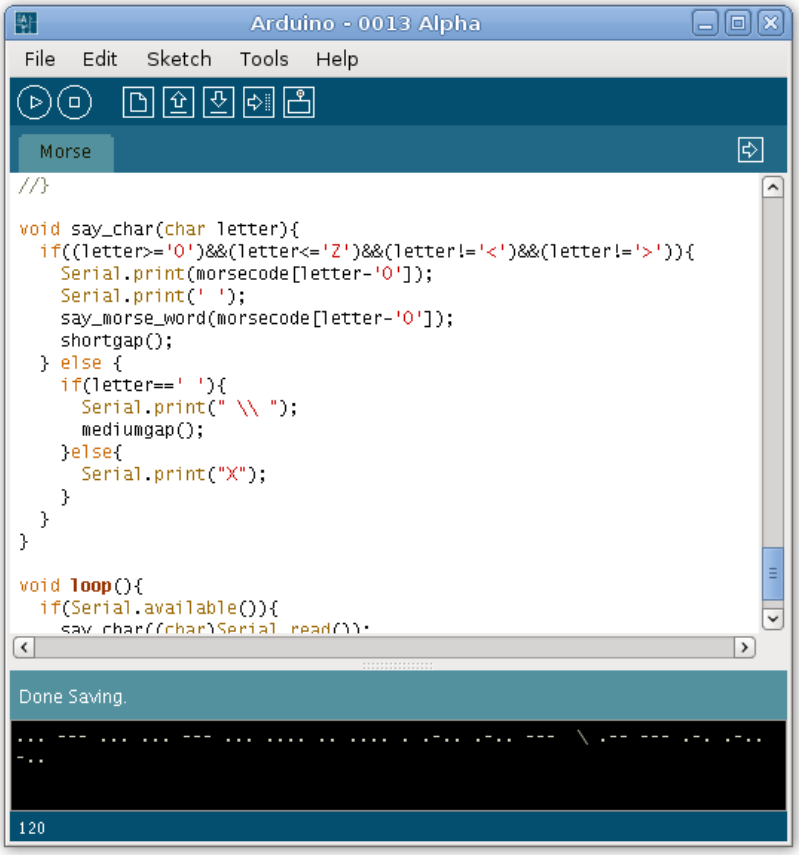
Temos Shields que pode adicionar recursos como: comunicação via wifi, envio de SMS, reconhecimento de voz, comunicação via ethernet, entre outros.

5.3.2 IDE Arduino

O projeto Arduino ainda contato com sua própria IDE de desenvolvimento que utiliza a linguagem C/C++. O programa é compilado em um computador normal pela IDE e depois enviado a placa Arduino conectada no computador. A partir desse momento o computador normal não é mais necessário, o próprio Arduino irá executar o programa compilado (JUSTEM, 2010).

Um programa para ser executado no Arduino é necessário utilizar à definição de duas funções. A primeira função é a “setup”, ela irá ser executada quando a placa inicializar, a segunda função é “loop”, essa função ficará sendo executada enquanto o Arduino estiver ligado (JUSTEM, 2010). A figura 3 mostra im exemplo de um programa na IDE do Arduino.

Figura 3 – Exemplo de programa na IDE Arduino



Fonte: Neto (2009)

6 ROBÓTICA

O termo Robô vem do Checo robota que significa trabalho, foi utilizado pela primeira vez 1921 por Karel Capek no romance “Rossum’s Universal Robots”. Os robôs de Capek eram máquinas de trabalho incansáveis, de aspecto humano (PIRES, 2002).

6.1 HISTÓRIA DA ROBÓTICA

Temos uma visão muito fantástica da robótica, vemos os robôs como seres metálicos semelhantes ao ser humano. Como veremos a diante os robôs são muitos simples, mas muito úteis em diversas áreas.

A robótica é muito mais antiga do conhecemos, o engenheiro grego Ctesibius fez experimentos nessa área a 270 A.C. Mas esses inventos eram mais estéticos do que práticos (PIRES, 2002).

Também os árabes baseados no conhecimento que os gregos desenvolveram para mera estética e contemplativa e colocaram algo em pratica, criando dispositivos para higiene pessoal.

Os árabes tinham bastante preocupação com a utilidade das suas criações (PIRES, 2002). Suas criações eram basicamente lavatório de mãos, para auxiliar a higiene do ser humano, tudo de forma mecânica, utilizando bóias para trazer o sabão e depois a tolha a medida que a água ia saindo.

6.2 ROBÓTICA HOJE

Hoje a robótica nos tem auxiliado em quase todas as áreas conhecidas, auxiliando desde a polícia com robôs antibombas até na exploração de planetas com é o caso de Marte.

Temos robôs operando na construção de carros, na indústria de cerâmica, na área militar, entre muitas outras áreas. Na atualidade o que mais chama a atenção da robótica é a exploração espacial, onde está ajudando o homem a chegar a lugares que nunca chegou.

Temos como exemplo mais recente o robô Curiosity que foi mandado a

Marte para tentar localizar vestígios de vida no solo de Marte. Mais robôs já auxiliaram o ser humano na exploração de marte, que é o caso de Oportunuty e Sprit que diferente do Curiosity que é movido a energia nuclear, esse eram movidos a energia solar (Robô Curiosity começa sua "longa caminhada" por Marte, 2012).

7 TRABALHOS CORRELATOS

Desenvolvimento de um robô autônomo detector de obstáculos foi um trabalho desenvolvido no Instituto de Estudos Superiores da Amazônia, os autores são William Penante dos Santos, Ronaldo Bentes Batista Junior, Raphael Diego Comesanha e Silva. O trabalho teve como objetivo o desenvolvimento de um robô que desvia de obstáculos usando a tecnologia Arduino. Como esse trabalho chegou-se ao resultado que a tecnologia Arduino é muito indicada para esse tipo de implementação, e a utilização de sensores de distancia por sonar garante muita precisão.

Um mini robô móvel seguidor de pistas guiado por visão local foi um trabalho desenvolvido no Centro Universitário da FEI em São Bernardo do Campo, na cidade de São Paulo. O trabalho foi desenvolvido por Eduardo R. Costa, Marcel L. Gomes e Reinaldo A. C. Bianchi em 2003. Foi desenvolvido um robô que se orienta através de uma pista, abaixo do robô existe um sistema visual composto de uma câmera colorida de 330 linhas coloridas para identificar a pista. Os resultados obtidos foi que o robô apresenta uma aceleração e uma desaceleração dependendo dos dados enviados pela câmera.

Planejamento de trajetória para o robô omni utilizando o algoritmo mapa de rotas probabilístico foi um trabalho desenvolvido no Departamento de Engenharia Elétrica (ENE) - Universidade de Brasília (UnB) em Brasília, no ano de 2005. Os autores foram Bruno Vilhena Adôrno, Geovany Araújo Borges e Carla Silva Rocha Aguiar, o trabalho consistiu no desenvolvimento de algoritmos para interpretação de mapas de rota para o robô Omni. Os resultados obtidos para geração do mapa foi realizada em menos de 1 segundo, foram necessárias mais de 85 configurações aleatórias para que espaços livres pudessem ser satisfatórios.

8 DESENVOLVIMENTO DO PROTÓTIPO

Foi desenvolvido um protótipo de um robô autônomo para transporte de cargas dentro do pátio de uma indústria, o robô é encarregado de transportar a matéria prima das docas até o ponto de processamento, depois deve transportar o produto final pronto até as docas de expedição.

8.1 METODOLOGIA

O desenvolvimento do protótipo foi realizado em três partes diferentes. De inicio foi realizado testes básicos com o hardware, configuração da IDE de programação, da porta serial no Windows e pesquisa de comunicação entre Java e Arduino. Nessa etapa foram desenvolvidos a aplicação básica de monitoramento do protótipo.

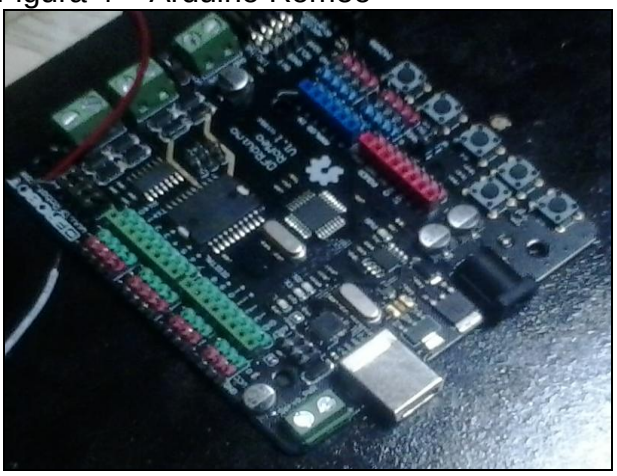
Na segunda parte teve início os testes com o sensor e motores. Nessa etapa foi desenvolvida a aplicação básica de monitoramento do protótipo.

Na última parte foi realizado toda a programação do Arduino e testes de movimentação do robô.

8.1.1 Primeira Etapa

Nessa primeira etapa foram realizados os primeiros testes com o Arduino, primeiramente foi configurado a IDE para comunicar com Arduino via USB. Esse processo é muito importante, pois é única forma de gravar os programas no Arduino. A figura 4 mostra o Arduino modelo Romeo utilizado nesse trabalho.

Figura 4 – Arduino Romeo



Fonte: Do autor

Primeiramente foi conectado o Aduino ao notebook utilizado, em seguida o Windows reconheceu que foi conectado um dispositivo na porta USB Windows não possui o driver do Aduino, a própria IDE do Aduino possui esse driver, somente foi necessário informar ao Windows que aquele dispositivo era serial e colocar o driver correto do Aduino, de acordo com o modelo. O tutorial com todas as instruções está no site oficial do Aduino.

Com o Aduino configurado foi criado alguns programas para dominar as características básica da linguagem. Primeiramente foram criados programas que retornavam caracteres via serial, para testar a comunicação entre o Arduino e o Windows. A própria IDE possui um monitor serial para enviar e receber dados pela porta serial onde o Arduino está conectado. Esse monitor facilita em muito os testes dos programas, pois dispensa uma aplicação separada somente para monitorar a porta serial.

Em seguida teve inicio o desenvolvimento da aplicação em Java, para testar a comunicação entre o Windows e o Arduino sem a passar pela IDE. A comunicação foi realizada via cabo serial, onde o Arduino deveria enviar um sinal a cada 10 segundos.

8.1.2 Segunda Etapa

Na segunda etapa teve inicio testes com o sensor infravermelho, onde o sensor após conectado ao Arduino retornava um número inteiro de acordo com a

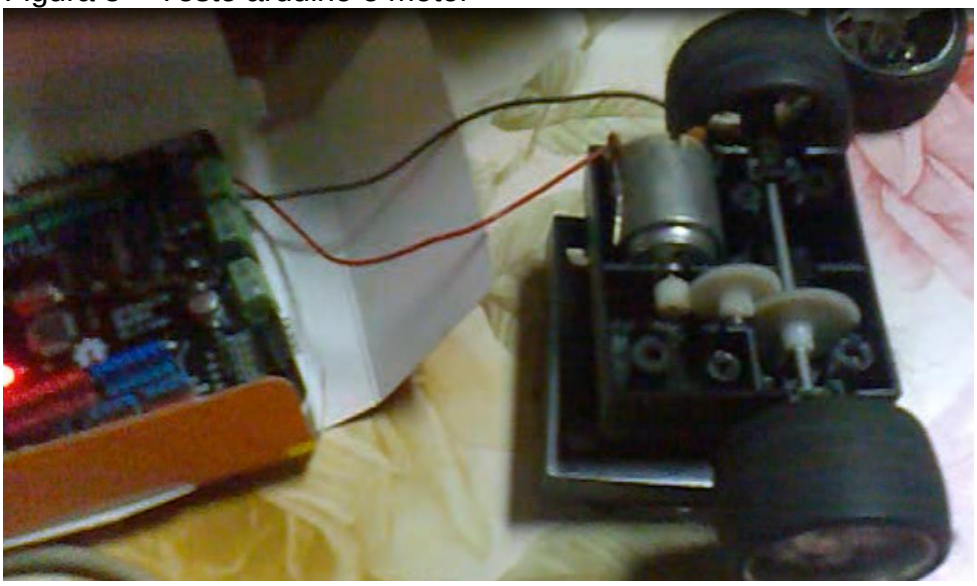
distancia do objeto a frente. Com esse teste foi possível determinar a distancia máxima de alcance e também calibrar a sensibilidade do sensor.

Para realizar um teste mais específicos do sensor, foi criado uma aplicação para fazer o arquivo enviar um sinal para outra aplicação em Java no Windows, caso um objeto se aproximar a 10 centímetros do sensor. Em seguida foi realizado outros testes enviando palavras caso o sensor fosse ativado.

Foram realizados testes fazendo o caminho inverso, a aplicação Java enviar dois números inteiros e o Arduino retornar a soma desses números. Também foram realizados testes de envio de caracteres para o Arduino.

Seguindo com os testes foi conectado um motor DC no Arduino, colocado as instruções para controle do motor no Arduino, onde o motor deveria funcionar por 5 segundos e parar por mais 10 antes de voltar a girar conforme figura 5. Com esses testes foi detectado que o motor não tinha energia suficiente para tracionar com a alimentação do Arduino, seria necessária uma alimentação externa.

Figura 5 – Teste arduino e motor



Fonte: Do Autor

Seguindo a especificação do modelo do Arduino, existe a possibilidade de conectar uma fonte de alimentação externa somente para os motores. Usando uma bateria de celular foi possível fornecer a energia suficiente para os motores.

Com o sensor infravermelho e o motor conectado ao Arduino, foi criado um programa para o Arduino para utilizar o sensor junto com o motor. Quando o sensor identifica-se um objeto a sua frente deveria parar o motor para, inverte a rotação, aguardar por 5 segundos e depois ativar o motor novamente.

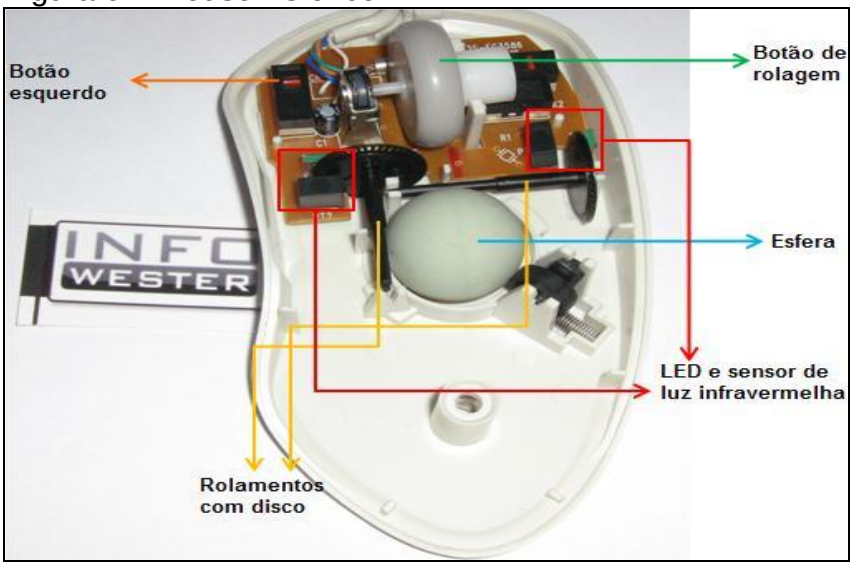
8.1.2.1 Encoder

Um *encoder* é um sensor que realiza a contagem de voltas que uma roda ou eixo realiza, dessa forma pode-se calcular a distância percorrida até o momento.

Dentro de mouse esférico existem dois encoder para detectar os movimentos da esféra. Foi utilizado dois tipos de mouse diferentes, um esférico e outro óptico, que detecta os movimentos através da emissão de uma luz vermelha.

O mouse esférico utiliza dois sensores de luz infravermelha, dentro ainda existem duas engrenagens para detectar os movimentos da esfera, uma para vertical e outra para horizontal.

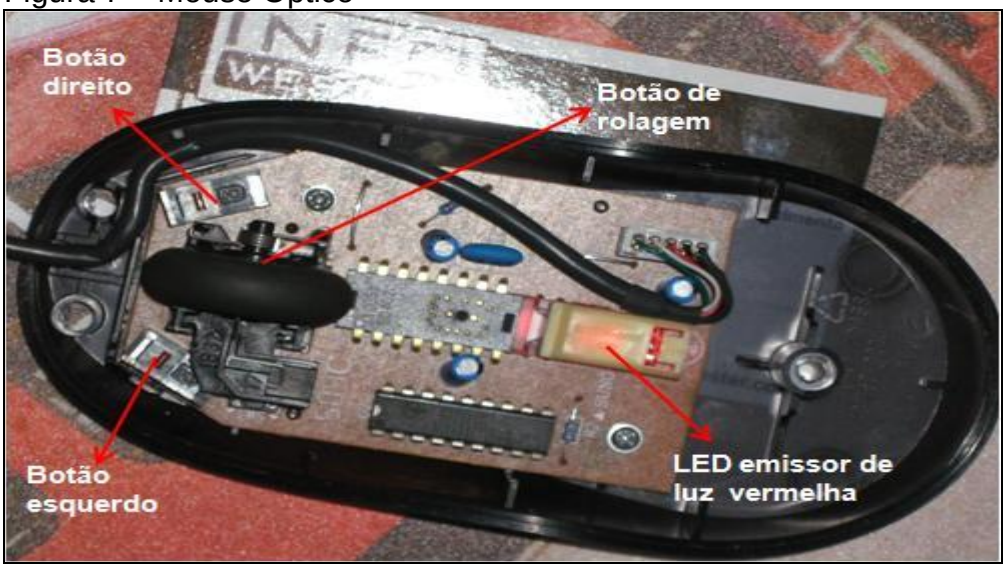
Figura 6 – Mouse Esférico



Fonte: Alecrim (2008)

Os mouses ópticos são diferentes, pois detectam o movimento através da emissão de luz vermelha, com isso dispensa o uso de partes mecânicas como mouse esférico conforme mostrado na figura 7.

Figura 7 – Mouse Óptico



Fonte: Alecrim (2008)

Para esse teste esperava-se que depois de conectado no Arduino, os mouses enviassem dados quando movimentados. Dessa forma os sensores seriam colocados no eixo do robô, para contar as rotações da roda.

Com isso seria possível criar um sistema de orientação para o robô, contado as rotações do eixo para determinar a distancia do destino final, e também a posição do robô em relação ao terreno.

9.1.3 Terceira Etapa

Para orientação do robô foi desenvolvido o algoritmo de dijkstra, dentro do Arduino. Assim foi possível calcular a distancia e o melhor caminho para alcançar o destino.

Para o robô reconhecer o caminho que deverá percorrer, foi criado um par de linhas com reflexão de luz, onde o sensor infravermelho matem o protótipo dentro do par de linhas.

Utilizando a teoria dos grafos criou-se um grafo para servir de rota para o protótipo. Onde os nós dos grafos foram simulados com uma série de barras reflexivas, para informar ao robô que havia chegado a um nó e assim recalcular a trajetória, para ter a confirmação que ainda está no caminho certo.

Outro teste que foi realizado foi somente com uma linha reflexiva, onde o robô deveria caminhar seguindo a linha até o destino final, mas não havia era

necessário utilizar o algoritmo de dijkstra, pois o robô só possuía um caminho para seguir.

8.2 RESULTADOS OBTIDOS

Neste capitulo será apresentado os resultados obtidos com o desenvolvimento desse trabalho. Durante o desenvolvimento do protótipo foram identificado várias dificuldades que foram contornadas, e outras que não foram possíveis de serem contornadas.

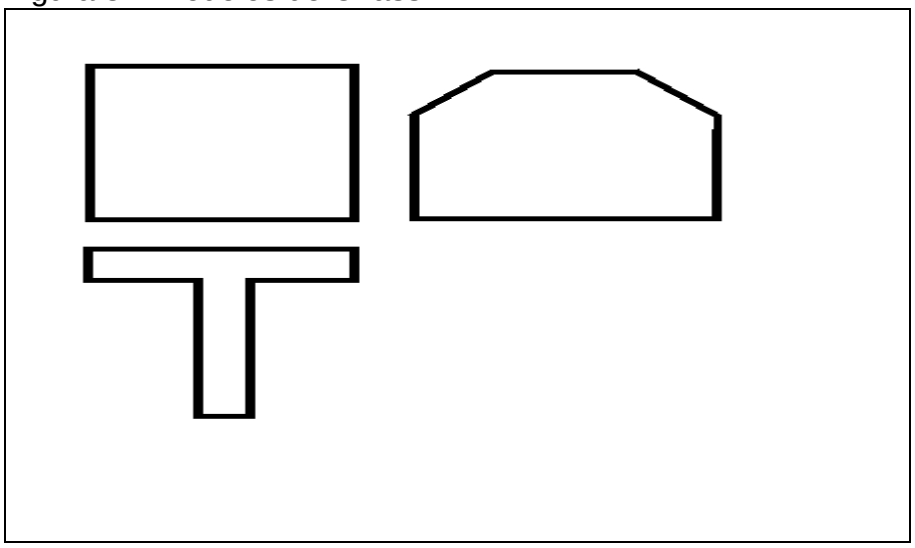
8.2.1 MONATAGEM DO PROTIPO E DESENVOLVIMENTO DA APLICAÇÃO JAVA

Durante a pesquisa desse trabalho viu-se a necessidade de uma fonte de energia secundária para auxiliar os motores, pois o Arduino não conseguiu alimentar os mesmos com a energia necessária.

Não foi preciso qualquer tipo desenvolvimento de um mecanismo de alimentação a parte, o modelo do Arduino utilizado nessa pesquisa possuiu uma entrada para conexão de uma alimentação externa para os motores, somente foi retirado *jumper* da placa e conectado a fonte externa.

As opções de chassi para realizar a implementação de um protótipo ainda são caros no mercado, por isso foi optado a construção do próprio chassi, dessa forma levou-se um tempo até localizar o material ideal, para ser resistente e ao mesmo tempo não muito pesado. Foram idealizados vários formatos diferentes, conforme ilustra a figura 8, porem identificou-se que o ideal seria em forma de trapézio, pois foi o modelo que mais forneceu distribuição do peso nas rodas, sem a necessidade de um contrapeso na roda traseira.

Figura 8 – Modelos de Chassi



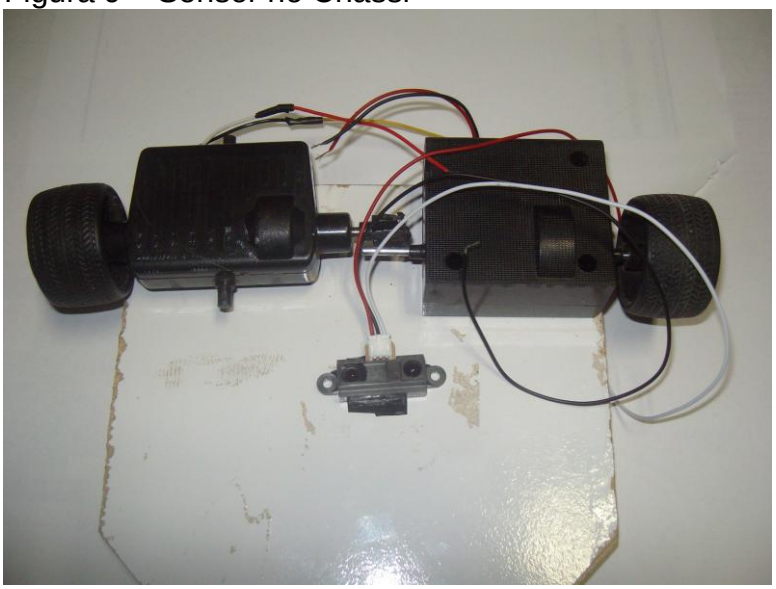
Fonte: Do Autor

Com esse modelo de chassi o protótipo ficou mais estável nas curvas e nas arrancadas, em relação ao modelo no formato retangular, que apresentava derrapagens nas arrancadas. O modelo retangular era mais pesado em relação aos outros modelos, pois era maior em relação aos outros modelos. Quando o protótipo começava o movimentar-se, apresentava derrapagens até começar realmente o movimento.

O Modelo em forma de “T” é um formato também muito eficiente, distribui bem o peso, mais leve e compacto. Porém necessita de um contrapeso na roda traseira para manter o equilíbrio, senão pode haver capotagens.

Para realizar a leitura o sensor infravermelho foi colocado embaixo do chassi, na linha do eixo, como é mostrado na figura 9.

Figura 9 – Sensor no Chassi

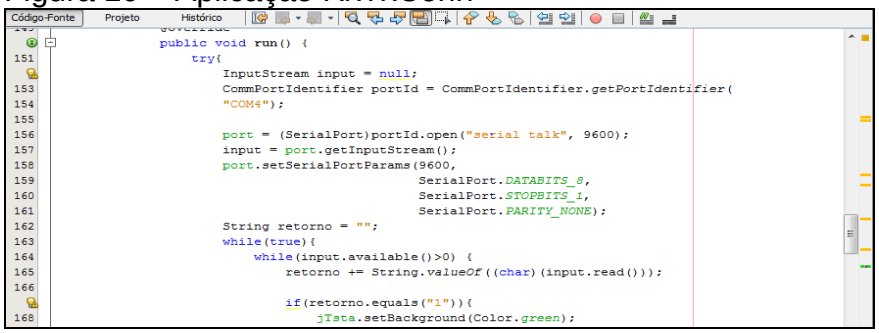


Fonte: Do Autor

8.2.1.1 Aplicação de Gerenciamento

O desenvolvimento da aplicação de comunicação com o Arduino, viu-se a necessidade de utilizar uma biblioteca Java RxTxConn para estabelecer a comunicação entre o Arduino e o Windows. Segue o trecho do código na figura 9.

Figura 10 – Aplicação RxTxConn



```
Código-Fonte   Projeto   Histórico
150         public void run() {
151             try{
153                 InputStream input = null;
154                 CommPortIdentifier portId = CommPortIdentifier.getPortIdentifier(
155                 "COM4");
156
157                 port = (SerialPort)portId.open("serial talk", 9600);
158                 input = port.getInputStream();
159                 port.setSerialPortParams(9600,
160                                          SerialPort.DATABITS_8,
161                                          SerialPort.STOPBITS_1,
162                                          SerialPort.PARITY_NONE);
163                 String retorno = "";
164                 while(true){
165                     while(input.available()>0) {
166                         retorno += String.valueOf((char)(input.read()));
168                         if(retorno.equals("1")){
                            jTsta.setBackground(Color.green);
```

Fonte: Do Autor

A comunicação com deu-se via serial conectado no Arduino e no micro onde se encontrava a aplicação desenvolvida em java.

Uma grande dificuldade foi encontrar a velocidade da porta serial, para estabelecer a comunicação com o protótipo. Sem a velocidade correta a aplicação não consegui enviar dados e receber.

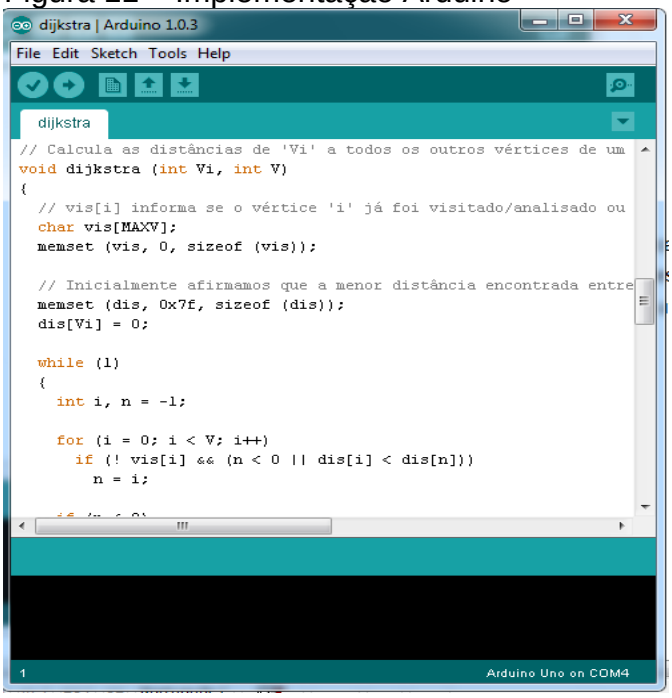
Esse parâmetro teve que ser configurado no método de envio e no de receber. Não existe muitos exemplos de código fonte para comunicação com o Arduino, os que existem fazem uso de outras bibliotecas java que toda o desenvolvimento mais difícil. O métodos mais fácil foi com a biblioteca RxTxConn.

O Arduino conta ainda com a função para receber dados na porta serial conforme a figura 9, essa função é ativada somente quando os dados são enviados para a serial. Isso simplifica o trabalho do desenvolvedor, pois não há necessidade de desenvolver uma lógica somente para ler a porta serial.

8.2.2 DESENVOLVIMENTO DO ALGORITMO DE DIJKSTRA

Durante o desenvolvimento do algoritmo de dijkstra verificou-se uma limitação da memória do Arduino. Para armazenar o caminho ser percorrido utilizou-se uma matriz, onde a maior matriz que foi criada era de 20 por 20 elementos, ou seja, uma matriz quadrada de 40 elementos. Segue amostra da implementação do algoritmo no arduino na figura 11.

Figura 11 – Implementação Arduino

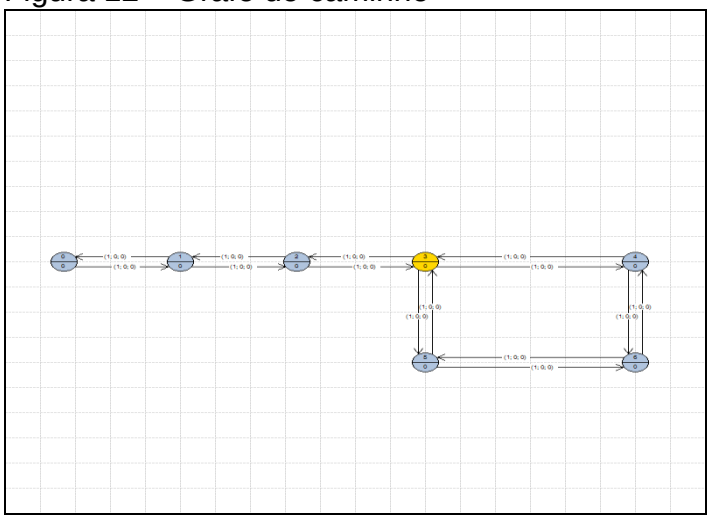


Fonte: Do Autor

Esse problema não foi possível contornar, pois se tratava de uma limitação do hardware utilizado no protótipo.

Então foi desenvolvido um grafo de 6 nó para representar o caminho que seria percorrido pelo protótipo. O grafo está representado na figura 12.

Figura 12 – Grafo do caminho



Fonte: Do Autor

Abaixo segue o algoritmo de dijsktra na IDE do Arduino:

```
// Calcula as distâncias de 'Vi' a todos os outros vértices de um grafo com 'V' vértices e armazena-as em dis[]
void dijkstra (int Vi, int V)
{
  // vis[i] informa se o vértice 'i' já foi visitado/analisado ou não (inicialmente nenhum vértice foi)
  char vis[MAXV];
  memset (vis, 0, sizeof (vis));

  // Inicialmente afirmamos que a menor distância encontrada entre Vi e qualquer outro vértice (exceto o próprio Vi) é infinita
  memset (dis, 0x7f, sizeof (dis));
  dis[Vi] = 0;

  while (1)
  {
    int i, n = -1;
```

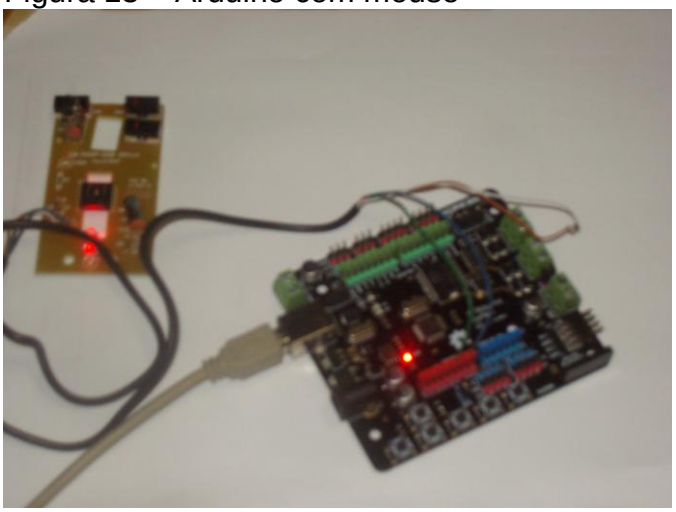
```
    for (i = 0; i < V; i++)
      if (! vis[i] && (n < 0 || dis[i] < dis[n]))
        n = i;

    if (n < 0)
      break;
    vis[n] = 1;

    Serial.println(n);
    for (i = 0; i < V; i++)
      if (MAdj[n][i] && dis[i] > dis[n] + MAdj[n][i])
        dis[i] = dis[n] + MAdj[n][i];
  }
}
```

A primeira tentativa de desenvolver um sensor de localização para o robô, deu-se utilizando um mouse esférico para contar as rotações do eixo, porém o arduino não conseguiu receber os dados. Em outra tentativa utilizou-se um mouse óptico, também sem sucesso, conforme figura 13.

Figura 13 – Arduino com mouse

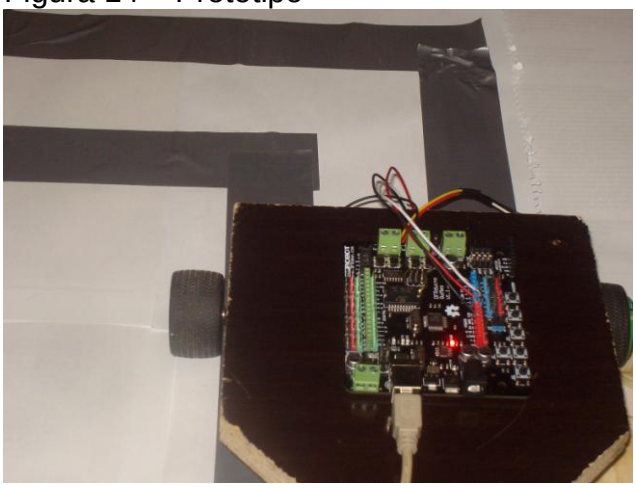


Fonte: Do Autor

Dessa forma optou-se em utilizar o sensor infravermelho para identificar o reflexo sobre uma superfície reflexiva, e assim o protótipo seguir a rota determinada pra o mesmo.

O protótipo caminhou conforme o esperado, porem o sensor não conseguia identificar os nós que eram fundamentais para o algoritmo de dijsktra, conforme na figura 14.

Figura 14 – Protótipo



Fonte: Do Autor

Foitentando colocar uma linha reflexiva larga no meio do caminho, dessa forma quando o sensor encontra uma linha em sua frente deveria reconhecer que encontrou um nó do caminho. Esse método apresentou problemas, pois apresentava conflito ao reconhecer as linhas de delimitavam as paredes do caminho.

O sensor reconhecia a linha dos nós como uma linha que delimitava o caminho a ser seguido ou identificava as paredes do caminho como nós, dessa forma o protótipo apresentava muitos erros ao tentar seguir o caminho ou ao identificar os nós do caminho. Outro método utilizado foi uma série de linhas reflexivas para simular um código de barras. Esses testes foram realizados na tentativa de informar ao robô a existência de nós para o algoritmo de dijsktra.

9 CONSIDERAÇÕES FINAIS

A tecnologia Arduino mostrou-se muito eficiente na construção de protótipos de robô. As opções de personalização, adicionando novos recursos para satisfazer as necessidades do projeto.

O Arduino suporta uma linguagem de alto nível como C, e aliado a uma IDE própria torna o domino da programação muito rápida e simples. A gravação dos programas dentro do arduino é muito mais simples, e a própria IDE se encarrega de gravar o programa.

Com esse projeto não foi possível desenvolver um protótipo totalmente autônomo, o arduino ainda não possui sensor com a capacidade e sensibilidade suficiente.

Para construir um robô totalmente autônomo seria necessária a confecção de um sensor para essa função, porém esse não era o objetivo desse trabalho, pensou-se em construir um protótipo utilizando os recursos que já estão disponíveis na atualidade.

Outro ponto observado é que esse hardware ainda é importado, não é produzido no Brasil, todos os sites visitados faziam importação do exterior, principalmente da China e da Itália.

Durante o desenvolvimento foi identificado que o modelo possui uma limitação de memória, a única opção seria trocar o modelo do arduino por outro com mais recursos, o arduino Mega seria uma opção para substituir o Romeo utilizado nesse trabalho.

Ainda é necessário mais pesquisas para criar um protótipo autônomo, mas com esse trabalho é possível criar um protótipo controlado remotamente, com aplicação em diversas áreas.

Com base nos conhecimentos adquiridos durante a pesquisa, a seguir serão descritas algumas sugestões para trabalhos, a fim de continuar com o desenvolvimento da tecnologia.

a) estudar sobre o desenvolvimento de sensor RGB para o Arduino;
b) estudar a performance dos outros modelos do Arduino e suas aplicações;

c) desenvolver protótipo robótico controlado remotamente para explorações subterrâneas;
d) estudar aplicação do Arduino em sistemas de monitoramento climático.

REFERÊNCIAS

Adôrno, Bruno Vilhena; Borges, Geovany Araújo; Aguiar, Carla Silva Rocha. **PLANEJAMENTO DE TRAJETÓRIA PARA O ROBÔ OMNI UTILIZANDO O ALGORITMO MAPA DE ROTAS PROBABILÍSTICO**. Disponivel em: <http://fei.edu.br/~rbianchi/publications/sbai2003.pdf> Acessado em: 25 nov. 2012.

CORTELETTI, Daniel**. Introdução à programação de microcontroladores Microchip PIC**. Disponível em: <http://www.adororobotica.com/sbrt-dossie51.pdf> Acessado em: 15 nov. 2012.

Costa, Eduardo R.; Gomes, Marcel L.; Bianchi, Reinaldo A. C. **DESENVOLVIMENTO DE UM ROBÔ AUTÔNOMO DETECTOR DE OBSTÁCULOS**. Disponível em: <http://fei.edu.br/~rbianchi/publications/sbai2003.pdf> Acessado em: 25 nov. 2012.

GOMES, P. H.; LEITE, T. S.; CAETANO, U. I. **A Arquitetura ARM**. Disponível em <http://www.ic.unicamp.br/~rodolfo/Cursos/mc722/2s2005/Trabalho/g20-arm.pdf> Acessado em nov. 2012.

Gonçalves, Luiz Marcos Garcia. **Robótica, principais tendências e direções**. Disponível em <http://www.comciencia.br/reportagens/2005/10/09.shtml> Acessado em 25 nov. 2012.

JUSTEM, Álvaro. **Conheça o Arduino.** Disponível em: <http://dl.dropbox.com/u/96987/arduino.pdf> Acessado em: 25 nov. 2012.

NETO, Silveira. Morse **Code Translator with Arduino.** Disponivel em: <http://silveiraneto.net/2009/02/28/morse-code-translator-with-arduino/> Acessado em: 25 nov. 2012.

PRADO, Sergio. A **onipresente arquitetura ARM.** Disponível em: <http://www.sergioprado.org/2011/02/24/a-onipresente-arquitetura-arm/> Acessado em: 5 nov. 2012.

PIRES, J. Noberto. Das **Máquinas Gregas à Moderna Robótica Industrial**. Disponível em: <http://robotics.dem.uc.pt/norberto/nova/pdfs/gregosxxi.pdf> Acessado em: 16 set. 2012.

SÁ, Mauricio Cardoso de. Programação **C para microcontroladores 8051.** 1. ed São Paulo: Érica, 2005. 334 p.

SOUZA, David José de. **Desbravando o PIC: ampliado e atualizado para PIC16F628A.** 10. ed São Paulo: Érica, 2003. 268 p.

Santos, William Penante dos; Batista, Ronaldo Bentes; Silva, Raphael Diego Comesanha e. **DESENVOLVIMENTO DE UM ROBÔ AUTÔNOMO DETECTOR DE OBSTÁCULOS**. Disponivel em: <http://www3.iesam-pa.edu.br/ojs/index.php/CONTROLE/article/view/652/536> Acessado em: 25 nov. 2012.

STALLINGS, William. **Arquitetura e organização de computadores.** 8. ed. São Paulo: Prentice Hall, 2010. 624 p.

TANENBAUM, Andrew S. **Organização estruturada de computadores.** 5. ed. Rio de Janeiro: Pearson Prentice Hall, 2007. 449 p.

TAURION, Cezar. Software embarcado: a nova onda da informática chips e softwares em todos objetos. Rio de Janeiro: Brasport, 2005. 178 p.

ZELENOVSKY, Ricardo; MEDONÇA, Alexandre. **Arquitetura de Microcontroladores Modernos.** Disponível em: <http://www.mzeditora.com.br/artigos/mic_modernos.htm> Acessado em: 20 nov. 2012.

APÊNDICE(S)

Protótipo de um Robô Autônomo para Movimentação e Orientação em Ambiente Controlado

Roberto F. Medeiros[1], Sérgio Coral[1]

[1]Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – (UnaCET) – Universidade do Extremo Sul Catarinense (UNESC)
Av. Universitária, 1105 – Bairro Universitário – Criciúma – SC – Brasil
roberto_s.m@hotmail.com, sergiocoral@unesc.net

Abstract. *Robotics has been around for a long time, trying to help the man in activities that offer risk or very stressful, robots are already widely used in automobile assembly line to reduce costs and increase production as well as the military has also made many advances in robotics. Given this backdrop, we intend to develop a prototype of an autonomous robot using arduino technology for use in industrial automation.*

Resumo. *A robótica já existe há muito tempo, tentando ajudar o homem em atividades que ofereça riscos ou muito desgastantes, robôs já são muito utilizados na linha de montagem de automóveis para diminuir os custos e aumentar a produção, assim como a área militar também já fez muitos avanços na área da robótica. Diante a este cenário, pretende-se desenvolver um protótipo de um robô autônomo utilizando a tecnologia arduino, para utilização na automação industrial.*

1. INTRODUÇÃO

Hoje a computação está em todos os lugares, está em empresas, nas casas, nos carros, em uma infinidade de computadores e produtos. Uma área que se destacou muito nos últimos tempos foi a robótica, hoje existem muitas formas de robôs, que auxiliam na indústria, segurança e pesquisas aeroespaciais. Pesando isso imaginou para esse trabalho a construção um protótipo de um robô para auxilio de transporte de carga dentro da área de deposito de uma indústria.

Esse protótipo traria mais segurança e eficiência para a empresa e para os colaboradores envolvidos na operação. Para isso será necessário entender os componentes básicos da arquitetura dos computadores e sistemas embarcados.

A tecnologia Arduino é muito utilizada na área da robótica, pois une desempenho e baixo consumo energético e baixo custo. O Arduino conta com seu próprio processador e memória com um computador convencional, será analisada em detalhes mais adiante.

Pensa-se que a robótica é uma tecnologia nova, que surgiu junto com o computador, mas essa tecnologia surgiu há muito tempo atrás com os gregos, passando pelos árabes até chegar nós dias atuais. As invenções criadas naquela época não se assemelhando com os robôs criados na atualidade, mas que possuíam utilidades, mesmo sem a tecnologia que dispomos hoje.

O desejo do ser humano em criar seres mecânicos para auxiliar em tarefas difíceis, com foi mencionado, já vem de muito tempo atrás e continua nos dias atuais, pois desenvolve-se robôs até mesmo para explorar outros planetas, como é o caso dos robôs que exploram o planeta Marte. Como enviar um ser humano tão longe do planeta seria muito perigoso e caro,

foi escolhido enviar um robô. A oportunidade de substituir um ser humano por um robô em situações de alta periculosidade já está sendo utilizada a muito tempo, como nas empresas automobilísticas, para soldar as pesas em um linha de montagem, operando sem a presença humana.

2 TECNOLOGIA ARM

Podemos definir sistemas embarcados como uma combinação de hardware e software projetada para uma função especifica. Por exemplo, na área médica temos vários dispositivos embarcados, como: bombas de infusão, máquinas de diálise, dispositivos protéticos, monitores cardíacos.

Existente uma variedade muito maior de sistemas embarcados do que sistemas de computadores de uso geral, como laptop ou desktop. Além da área médica, sistemas embarcados existem em diversas outras áreas como: automotivo, eletrônica, controle industrial e automação de escritório.

A arquitetura ARM é uma família de microprocessadores e microcontroledores baseado em RISC, projetada pela ARM Inc., Cambridge, Inglaterra. A empresa não fabrica processadores, mas sim projeta e licencia os fabricantes.

Os chips ARM são processadores pequenos de baixo custo, alta velocidade e baixo consumo enérgico. Eles são muito utilizados em dispositivos portáteis, como telefones, celulares, tablets, muitos outros produtos de consumo.

A arquitetura ARM foi criada pela Acorn Computers. Em 1980 a Arcon ganhou o contrato da British Broacasting Corporation (BBC) para desenvolvimento de uma arquitetura de microcomputador para BBC Computer Literacy Project. Com esse contrato a Arcon conseguiu desenvolver o primeiro processador RISC comercial, o Arcon RISC Machine (ARM).

A primeira versão foi o ARM1 e começou a operar em 1985, e foi usado como coprocessador na máquina da BBC. Ainda em 1985 a Arcon lançou o ARM2 como maior velocidade em um mesmo espaço físico. Já em 1989 foi lança o ARM3, com inclusão de mais funcionalidades e melhorias.

Depois do ARM3 o desenvolvimento, além disso, ficou fora do escopo da Arcon. Então uma nova empresa foi organizada, com Arcon, VLSI e Apple Computer como parceiros fundadores, conhecida como ARM Ltd. O primeiro produto desenvolvido foi o ARM6, uma melhoria sobre o ARM3.

Os processadores ARM são projetados para atender as seguintes necessidades de sistemas:

a) sistemas embarcados de tempo real: sistemas para aplicações de armazenamento, automotivas, industriais e de redes;

b) plataformas de aplicação: dispositivos executando sistemas operacionais, como Linux, Palm OS, Symbian OS e Windows CE em aplicações sem fio, entretenimento e imagens digitais;

c) aplicações seguras: smart cards, placas SIM e terminais de pagamento.

d)

Hoje a tecnologia ARM está presente nas mais variada áreas e está evoluindo cada vez mais, tornando a tecnologia acessível a todos.

3 MICROCONTROLADOR

Em poucas palavras, podemos definir o microcontrolador como um “pequeno” componente eletrônico, dotado de uma “inteligência” programável, utilizado no controle de processo lógico.

Microcontroladores diferem de microprocessadores pelo fato de incorporar vários componentes em uma única pastilha que chamamos de “chip”, garantindo baixo custo e alto rendimento.

São muitas as aplicações para os microcontroladores, e estão presentes em vários seguimentos do mercado e da vida de todos nós. Segundo Corteletti (2006) no final do século XX, e início do século XXI, muitos equipamentos sofreram evoluções bastante radicais, grande parte graças aos microcontroladores.

Por seu baixo custo e alta performace, os microcontroladores está abrindo bastante espaço no mercado, não só no mundo, mais também no Brasil (TAURION, 2005). Os microcontroladores estão presentes em várias áreas como automação industrial, automação comercial, automação predial, área automobilística, agrícola, produtos manufaturados, eletrodomésticos, telecomunicações, entre muitas outras.

Quando entramos em um carro não vemos, mas estão no freio ABS, direção eletrônica, injeção eletrônica de combustível, controle de tração, controle de suspensão e travas elétricas, entre muitos outros. Os microcontroladores estão em muitos lugares para nos trazer eficiência, segurança e conforto.

Estima-se que são produzidos cerca de 63 milhões de carros anualmente no mundo, e cada veículo possui em média 30 microcontroladores. As aplicações então no conforto na segurança e eficiência do veículo.

Uma das principais características dos microcontroladores PIC está na simplicidade de aplicação, assim facilitando a implementação e diminuindo os custos de desenvolvimento, permitindo a aplicação em projetos de pequeno porte.

Os microcontroladores PIC são divididos em famílias, cada uma com características relacionada a sua performance e funcionalidade. Temos a família PIC10, de menos recursos, que são muito aplicadas em funções de on-off mais simples e de menos porte, já as famílias PIC30 e PIC33 possuem mais recurso e são aplicadas em projetos mais complexos, essas famílias são muito utilizadas em telecomunicações, rede sem fios de alta performance e controles de tempo real de alta velocidade.

4 ARDUINO

O objetivo do projeto Arduino é promover uma plataforma fácil para prototipação de projetos interativos. Basicamente as placas Arduino são microconroladores, ele faz parte da computação física, área da computação onde o software interage diretamente com o hardware.

A placa Arduino já vem com vários recursos de fábrica, porém dependendo do projeto a ser implementado é necessário incorporar outros recursos que não vem com o Arduino. As Shields são placas que podem ser conectadas no Arduino que podem adicionar recursos extras. Existem várias Shields diferentes, cada uma com recursos adicionais diferentes ao Arduino.

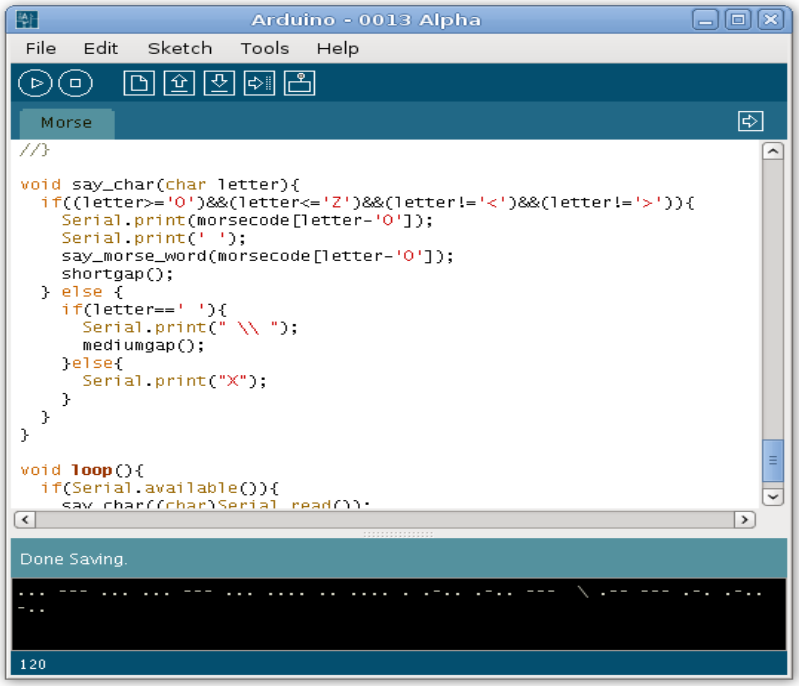
Temos Shields que pode adicionar recursos como: comunicação via wifi, envio de SMS, reconhecimento de voz, comunicação via ethernet, entre outros.

O projeto Arduino ainda contato com sua própria IDE de desenvolvimento que utiliza a linguagem C/C++. O programa é compilado em um computador normal pela IDE e depois enviado a placa Arduino conectada no computador. A partir desse momento o computador normal não é mais necessário, o próprio Arduino irá executar o programa compilado.

Um programa para ser executado no Arduino é necessário utilizar à definição de duas funções. A primeira função é a “setup”, ela irá ser executada quando a placa inicializar, a

segunda função é “loop”, essa função ficará sendo executada enquanto o Arduino estiver ligado. A figura 1 mostra im exemplo de um programa na IDE do Arduino.

Figura 1 – IDE Arduino



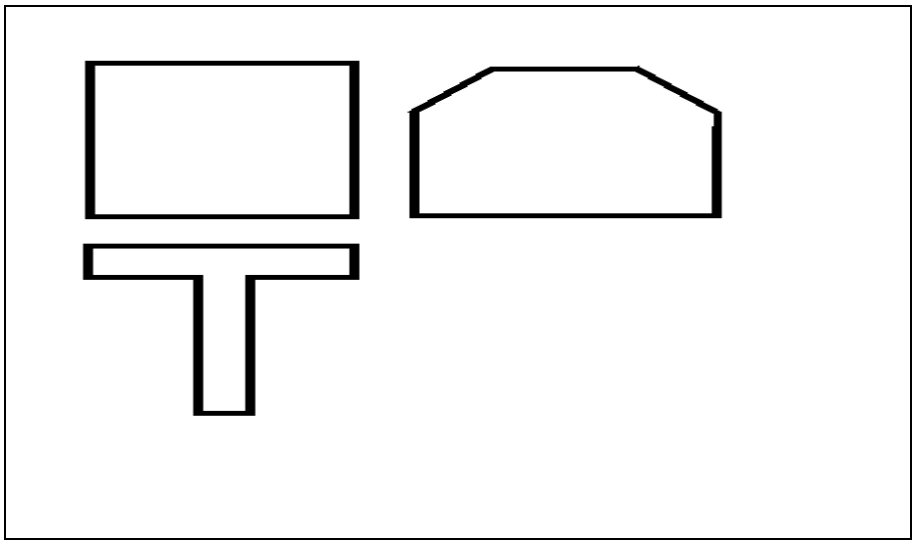
5 RESULTADOS OBTIDOS

Neste capitulo será apresentado os resultados obtidos com o desenvolvimento desse trabalho. Durante o desenvolvimento do protótipo foram identificado várias dificuldades que foram contornadas, e outras que não foram possíveis de serem contornadas.

Durante a pesquisa desse trabalho viu-se a necessidade de uma fonte de energia secundária para auxiliar os motores, pois o Arduino não conseguiu alimentar os mesmos com a energia necessária. Não foi preciso qualquer tipo desenvolvimento de um mecanismo de alimentação a parte, o modelo do Arduino utilizado nessa pesquisa possuiu uma entrada para conexão de uma alimentação externa para os motores, somente foi retirado *jumper* da placa e conectado a fonte externa.

As opções de chassi para realizar a implementação de um protótipo ainda são caros no mercado, por isso foi optado a construção do próprio chassi, dessa forma levou-se um tempo até localizar o material ideal, para ser resistente e ao mesmo tempo não muito pesado. Foram idealizados vários formatos diferentes, conforme ilustra a figura 2, porem identificou-se que o ideal seria em forma de trapézio, pois foi o modelo que mais forneceu distribuição do peso nas rodas, sem a necessidade de um contrapeso na roda traseira.

Figura 2 – Modelos de Chassi

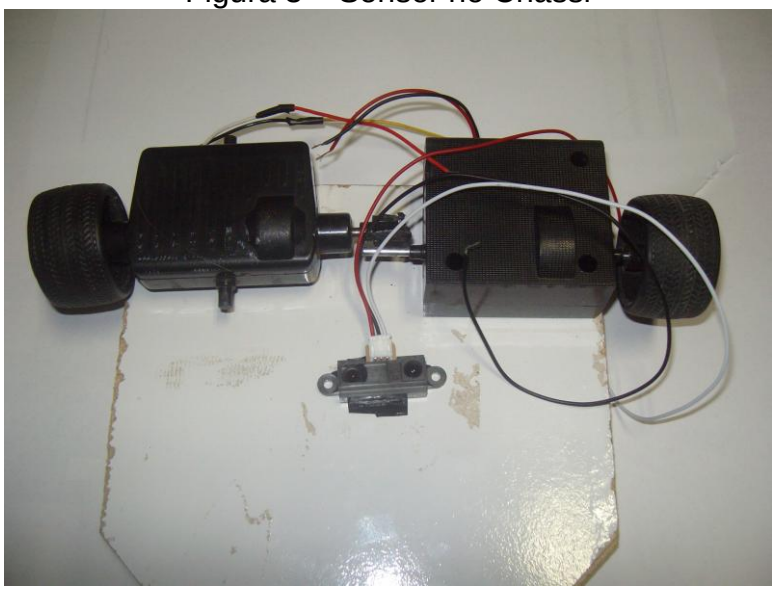


Com esse modelo de chassi o protótipo ficou mais estável nas curvas e nas arrancadas, em relação ao modelo no formato retangular, que apresentava derrapagens nas arrancadas. O modelo retangular era mais pesado em relação aos outros modelos, pois era maior em relação aos outros modelos. Quando o protótipo começava o movimentar-se, apresentava derrapagens até começar realmente o movimento.

O Modelo em forma de "T" é um formato também muito eficiente, distribui bem o peso, mais leve e compacto. Porém necessita de um contrapeso na roda traseira para manter o equilíbrio, senão pode haver capotagens.

Para realizar a leitura o sensor infravermelho foi colocado embaixo do chassi, na linha do eixo, como é mostrado na figura 3.

Figura 3 – Sensor no Chassi



O desenvolvimento da aplicação de comunicação com o Arduino, viu-se a necessidade de utilizar uma biblioteca Java RxTxConn para estabelecer a comunicação entre o Arduino e o Windows. Segue o trecho do código na figura 4.

Figura 4 – Aplicação RxTxConn

```
Código-Fonte  Projeto  Histórico
150        public void run() {
151            try{
152                InputStream input = null;
153                CommPortIdentifier portId = CommPortIdentifier.getPortIdentifier(
154                "COM4");
155
156                port = (SerialPort)portId.open("serial talk", 9600);
157                input = port.getInputStream();
158                port.setSerialPortParams(9600,
159                                         SerialPort.DATABITS_8,
160                                         SerialPort.STOPBITS_1,
161                                         SerialPort.PARITY_NONE);
162                String retorno = "";
163                while(true){
164                    while(input.available()>0) {
165                        retorno += String.valueOf((char)(input.read()));
166
167                        if(retorno.equals("1")){
168                            jTsta.setBackground(Color.green);
```

A comunicação com deu-se via serial conectado no Arduino e no micro onde se encontrava a aplicação desenvolvida em java.

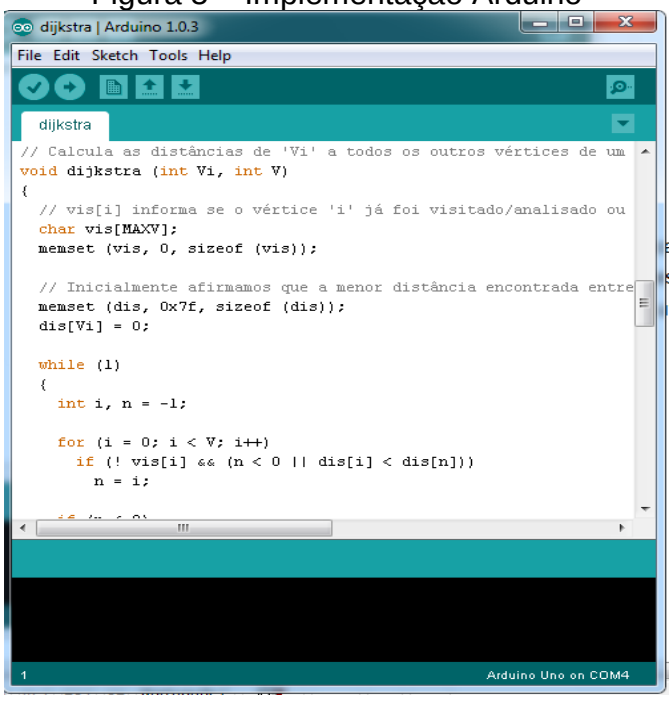
Uma grande dificuldade foi encontrar a velocidade da porta serial, para estabelecer a comunicação com o protótipo. Sem a velocidade correta a aplicação não consegui enviar dados e receber.

Esse parâmetro teve que ser configurado no método de envio e no de receber. Não existe muitos exemplos de código fonte para comunicação com o Arduino, os que existem fazem uso de outras bibliotecas java que toda o desenvolvimento mais difícil. O métodos mais fácil foi com a biblioteca RxTxConn.

O Arduino conta ainda com a função para receber dados na porta serial conforme a figura 9, essa função é ativada somente quando os dados são enviados para a serial. Isso simplifica o trabalho do desenvolvedor, pois não há necessidade de desenvolver uma lógica somente para ler a porta serial.

Durante o desenvolvimento do algoritmo de dijkstra verificou-se uma limitação da memória do Arduino. Para armazenar o caminho ser percorrido utilizou-se uma matriz, onde a maior matriz que foi criada era de 20 por 20 elementos, ou seja, uma matriz quadrada de 40 elementos. Segue amostra da implementação do algoritmo no arduino na figura 5.

Figura 5 – Implementação Arduino



Esse problema não foi possível contornar, pois se tratava de uma limitação do hardware utilizado no protótipo. A primeira tentativa de desenvolver um sensor de localização

para o robô, deu-se utilizando um mouse esférico para contar as rotações do eixo, porém o arduino não conseguiu receber os dados.

Dessa forma optou-se em utilizar o sensor infravermelho para identificar o reflexo sobre uma superfície reflexiva, e assim o protótipo seguir a rota determinada pra o mesmo. O protótipo caminhou conforme o esperado, porem o sensor não conseguia identificar os nós que eram fundamentais para o algoritmo de dijsktra.

Foi tentando colocar uma linha reflexiva larga no meio do caminho, dessa forma quando o sensor encontra uma linha em sua frente deveria reconhecer que encontrou um nó do caminho. Esse método apresentou problemas, pois apresentava conflito ao reconhecer as linhas de delimitavam as paredes do caminho.

O sensor reconhecia a linha dos nós como uma linha que delimitava o caminho a ser seguido ou identificava as paredes do caminho como nós, dessa forma o protótipo apresentava muitos erros ao tentar seguir o caminho ou ao identificar os nós do caminho. Outro método utilizado foi uma série de linhas reflexivas para simular um código de barras. Esses testes foram realizados na tentativa de informar ao robô a existência de nós para o algoritmo de dijsktra.

6 CONSIDERAÇÕES FINAIS

A tecnologia Arduino mostrou-se muito eficiente na construção de protótipos de robô. As opções de personalização, adicionando novos recursos para satisfazer as necessidades do projeto.

O Arduino suporta uma linguagem de alto nível como C, e aliado a uma IDE própria torna o domino da programação muito rápida e simples. A gravação dos programas dentro do arduino é muito mais simples, e a própria IDE se encarrega de gravar o programa. Com esse projeto não foi possível desenvolver um protótipo totalmente autônomo, o arduino ainda não possui sensor com a capacidade e sensibilidade suficiente.

Para construir um robô totalmente autônomo seria necessária a confecção de um sensor para essa função, porém esse não era o objetivo desse trabalho, pensou-se em construir um protótipo utilizando os recursos que já estão disponíveis na atualidade.

Outro ponto observado é que esse hardware ainda é importado, não é produzido no Brasil, todos os sites visitados faziam importação do exterior, principalmente da China e da Itália.

Durante o desenvolvimento foi identificado que o modelo possui uma limitação de memória, a única opção seria trocar o modelo do arduino por outro com mais recursos, o arduino Mega seria uma opção para substituir o Romeo utilizado nesse trabalho.

Ainda é necessário mais pesquisas para criar um protótipo autônomo, mas com esse trabalho é possível criar um protótipo controlado remotamente, com aplicação em diversas áreas. Com base nos conhecimentos adquiridos durante a pesquisa, a seguir serão descritas algumas sugestões para trabalhos, a fim de continuar com o desenvolvimento da tecnologia.

a) estudar sobre o desenvolvimento de sensor RGB para o Arduino;
b) estudar a performance dos outros modelos do Arduino e suas aplicações;
c) desenvolver protótipo robótico controlado remotamente para explorações subterrâneas;
d) estudar aplicação do Arduino em sistemas de monitoramento climático.

REFERÊNCIA

Adôrno, Bruno Vilhena; Borges, Geovany Araújo; Aguiar, Carla Silva Rocha.

PLANEJAMENTO DE TRAJETÓRIA PARA O ROBÔ OMNI UTILIZANDO O ALGORITMO MAPA DE ROTAS PROBABILÍSTICO. Disponivel em:

<http://fei.edu.br/~rbianchi/publications/sbai2003.pdf> Acessado

em: 25 nov. 2012.

CORTELETTI, Daniel**. Introdução à programação de microcontroladores**

Microchip PIC. Disponível em: <http://www.adororobotica.com/sbrt-dossie51.pdf> Acessado em: 15 nov. 2012.

Costa, Eduardo R.; Gomes, Marcel L.; Bianchi, Reinaldo A. C.

DESENVOLVIMENTO DE UM ROBÔ AUTÔNOMO DETECTOR DE OBSTÁCULOS. Disponível em: <http://fei.edu.br/~rbianchi/publications/sbai2003.pdf>

Acessado em: 25 nov. 2012.

GOMES, P. H.; LEITE, T. S.; CAETANO, U. I. **A Arquitetura ARM**. Disponível em <http://www.ic.unicamp.br/~rodolfo/Cursos/mc722/2s2005/Trabalho/g20-arm.pdf> Acessado em nov. 2012.

Gonçalves, Luiz Marcos Garcia. **Robótica, principais tendências e direções**. Disponível em

<http://www.comciencia.br/reportagens/2005/10/09.shtml> Acessado em 25 nov. 2012.

JUSTEM, Álvaro. **Conheça o Arduino.** Disponível em:

<http://dl.dropbox.com/u/96987/arduino.pdf> Acessado em: 25 nov. 2012.

NETO, Silveira. Morse **Code Translator with Arduino.** Disponivel em: <http://silveiraneto.net/2009/02/28/morse-code-translator-with-arduino/> Acessado em: 25 nov. 2012.

PRADO, Sergio. A **onipresente arquitetura ARM.** Disponível em: <http://www.sergioprado.org/2011/02/24/a-onipresente-arquitetura-arm/> Acessado em: 5 nov. 2012.

PIRES, J. Noberto. Das **Máquinas Gregas à Moderna Robótica Industrial**. Disponível em: <http://robotics.dem.uc.pt/norberto/nova/pdfs/gregosxxi.pdf> Acessado em: 16 set. 2012.

SÁ, Mauricio Cardoso de. Programação **C para microcontroladores 8051.** 1. ed São Paulo: Érica, 2005. 334 p.

SOUZA, David José de. **Desbravando o PIC: ampliado e atualizado para PIC16F628A.** 10. ed São Paulo: Érica, 2003. 268 p.

Santos, William Penante dos; Batista, Ronaldo Bentes; Silva, Raphael Diego

Comesanha e. **DESENVOLVIMENTO DE UM ROBÔ AUTÔNOMO DETECTOR DE OBSTÁCULOS**. Disponivel em:

<http://www3.iesam-pa.edu.br/ojs/index.php/CONTROLE/article/view/652/536>
Acessado em: 25 nov. 2012.

STALLINGS, William. **Arquitetura e organização de computadores.** 8. ed. São Paulo: Prentice Hall, 2010. 624 p.

TANENBAUM, Andrew S. **Organização estruturada de computadores.** 5. ed. Rio de Janeiro: Pearson Prentice Hall, 2007. 449 p.

TAURION, Cezar. Software embarcado: a nova onda da informática chips e softwares em todos objetos. Rio de Janeiro: Brasport, 2005. 178 p.

ZELENOVSKY, Ricardo; MEDONÇA, Alexandre. **Arquitetura de Microcontroladores Modernos.** Disponível em: <http://www.mzeditora.com.br/artigos/mic_modernos.htm> Acessado em: 20 nov. 2012.

ANEXO(S)