**UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL**

**ESCOLA DE ENGENHARIA**
**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**RODRIGO GUERINI DA SILVA**

# ADAPTADOR USB PARA ROBÔ ABB MODELO IRB 1400

**Porto Alegre**

2015

**RODRIGO GUERINI DA SILVA**

# ADAPTADOR USB PARA ROBÔ ABB MODELO IRB 1400

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para Graduação em Engenharia Elétrica.

ORIENTADOR: Prof. Dr. Renato Ventura Bayan Henriques

Porto Alegre

2015

RODRIGO GUERINI DA SILVA

**ADAPTADOR USB PARA ROBÔ ABB MODELO IRB 1400**

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de "Projeto de Diplomação", do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

_____

Prof. Dr. Renato Ventura Bayan Henriques, UFRGS

Aprovado em: ___/___/2015

Banca Examinadora:

Prof. Dr. Flávio José Lorini, UFRGS _____

Doutor pela Politecnico Di Milano – Milão, Itália

Prof. Dr. Walter Fetter Lages, UFRGS _____

Doutor pelo Instituto Tecnológico de Aeronáutica, São José dos Campos, Brasil

Porto Alegre, julho de 2015.

Dedico este trabalho aos meus pais, Alceu e Irina, à minha irmã Carina, minha prima Cristina, minha *nona* Catharina, meu primo Rafael e aos demais familiares e amigos. Cada uma dessas pessoas foi importante em momentos diferentes da minha vida.

# AGRADECIMENTOS

Agradeço aos meus queridos pais por terem sempre me incentivado e acreditado em mim, me orientado da melhor forma possível em todas as fases da minha vida.

Ao Prof. Dr. Renato Ventura Bayan Henriques, orientador deste trabalho, pela compreensão e incentivo no desenvolvimento desta tarefa e também pela demonstração de interesse no aprendizado de seus alunos.

Aos colegas pelo seu auxílio nas tarefas desenvolvidas durante o curso, em especial aos amigos Argus, Clodoaldo Lambiase e Igor Gomes que foram sempre ótimos colegas e sempre me ajudaram em muitas das árduas tarefas demandadas pelas disciplinas.

Aos bons professores como Luis Fernando Pereira, Gladis Bordin, Luiz Tiarajú, Roberto Homrich, Alexandre Balbinot, Tristão dos Santos, Yeddo Blauth, Ály Flores Filho, Irene Strauch da matemática e Mario Baibich da física.

Aos meus colegas de trabalho, especialmente ao Fábio Fernandez Bezerra e Gustavo Luchine Ishihara, que me ajudaram de forma fraternal na conclusão deste curso e sempre foram os que mais sofreram dividindo comigo minhas tarefas do trabalho. Não posso esquecer-me de José Pedro, Luiza de Marilac, Mirian Cunha, Naiche Barcelos e Vanessa Cravo pelo incentivo, e João Bettoni pela colaboração, permitindo meus estudos.

À minha vó Catharina Guerini, que tomou conta de mim durante muitos anos, minha mãe Irina, que toma conta de mim até hoje, meu pai Alceu, que me ensina importantes lições até quando permanece em silêncio e à minha irmã que sempre me deu forças.

À Tatiana Taís Vieria por ter compartilhado comigo muitos bons e maus momentos durante este período.

Aos meus amigos que sempre me incentivaram e auxiliaram nessa jornada, André Santin, Cristiano Leite, Cristina Linck, Fabiano Chies, José Linck, Lucio Fabricio Antunes, Luis Carlos Catto, Márcio Coelho, Marcos Guerini, Rafael Atolini, Rafael de Figueiredo e Roberto Hoffmann.

.

# RESUMO

Este projeto tem como objetivo modernizar uma das entradas de dados do robô da marca ABB Robotics Products, modelo IRB 1400. A entrada substituída será o drive de disquete por conta da dificuldade de manter e trabalhar com tecnologia ultrapassada, isto melhorará a produtividade dos alunos de engenharia da UFRGS em sala de aula. Para concretização desta ideia, será criado um protótipo de uma placa capaz de realizar a comunicação entre uma porta USB (Universal Serial Data) e a controladora de disquete que faz parte do robô IRB 1400.

Palavras-chaves: ABB IRB 1400, Adaptador USB, Controladora de disquete.

# ABSTRACT

This project has the objective of modernizing one of the data inputs of a robot brand ABB Robotics Products, model IRB 1400. The entry that will be replaced is the floppy drive on account of the difficulty of maintaining and working with outdated technology, it will improve the productivity of engineering students of UFRGS in classroom. To implement this idea will be created o prototype of a board capable of performing the communication between a USB port (Universal Serial Date) and the floppy-disk controller that is part of the robot IRB 1400.



Keywords: ABB IRB 1400, USB Adapter, Floppy Disk Controller.

# SUMÁRIO

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

## LISTA DE ABREVIATURAS

ASCII: *American Standard Code for Information Interchange* (Código Padrão Americano para o Intercâmbio de Informação)

BT: *Buffer de Track* (Buffer de trilha)

CD: *Compact Disc* (disco compacto)

CODEC: Codificador e decodificador MFM

CRC: *Cyclic Redundancy Check* (verificação de redundância cíclica)

DC: *Direct Current* (corrente direta)

DELET: Departamento de Engenharia Elétrica

DMPIS: *Dhrystone Millions of Instructions Per Second* (Dhrystone Milhões de instruções por segundo)

EB: *Expansion Board* (placa de expansão)

FDC: *Floppy Disk Controller* (controladora da unidade de disquete)

FDD: *Floppy Disk Drive* (unidade de disquete)

ID: *Identification* (identificação)

IDE: *Integrated Development Environment* (Ambiente de desenvolvimento integrado)

LAN: *Local Area Network* (Rede de área local)

RAM: *Random Access Memory* (memória de acesso aleatório)

ROM: *Read Only Memory* (memória apenas para leitura)

OEM: *Original Equipment Manufacturer* (fabricante original de equipamento)

PC: *Personal Computer* (computador pessoal)

MFM: *Modified Frequency Modulation* (modulação em frequência modificada)

MSD: *Mass Storage Device* (dispositivo de armazenamento massivo)

PLL: *Phase-Locked Loop* (Malha de captura de fase)

RISC: *Reduced Instruction Set Computer* (computador com um conjunto reduzido de instruções)

SCSI: *Small Computer System Interface* (pequena interface de sistema de computador)

SD Card: *Secure Digital Card* (cartão digital seguro)

UFRGS: Universidade Federal do Rio Grande do Sul

USB: *Universal Serial Bus* (barramento universal serial)

WLAN: *Wireless Local Area Network* (Rede de área local sem fio)

# 1. INTRODUÇÃO

O adaptador USB será desenvolvido para ser implantado no braço robótico da ABB, modelo IRB 1400, disponível aos alunos da graduação no laboratório de robótica, em substituição ao sistema de disco magnético atual.

## 1.1. Objetivo

O objetivo da troca desse dispositivo é atualizar uma das tecnologias de comunicação do braço mecânico, fazendo assim com que seja ampliada a gama de dispositivos que poderão ser conectados a este braço e excluindo a necessidade do uso de disquetes, que já caíram em desuso e são difíceis de serem encontrados no comércio para troca daqueles desgastados que estão disponíveis aos alunos.

## 1.2. Motivação

O aprendizado a respeito do uso do robô industrial, do laboratório de robótica, é parte da formação acadêmica de alguns alunos da Engenharia Elétrica e também de outros cursos da UFRGS. A robótica industrial tem grande aplicação nas áreas de produção automatizada e os grandes fabricantes de robôs são todos estrangeiros. Há sem dúvida uma necessidade de profissionais especializados que possam trabalhar ou mesmo desenvolver essa tecnologia em solo nacional. Considerando que a UFRGS procura formar bons profissionais e dificilmente tem retorno direto, uma modernização em equipamentos de seus laboratórios, no caso a viabilização de um adaptador USB foi pensado não só para resolver o problema da dificuldade de manutenção do uso de disquetes, material obsoleto, mas também como forma de retribuição à Universidade.

### 1.3. Robótica

O campo da robótica tem sido um dos maiores impulsionadores dos avanços tecnológicos da humanidade já há muito tempo. A partir do ponto em que as tarefas repetitivas realizadas por humanos passaram a ser realizadas por robôs, muitas das falhas que ocorriam nos processos de produção fabril foram sendo excluídas das rotinas. Isso porque fatores como fadiga, desatenção, doenças e outros inerentes ao ser humano não estão presentes da mesma forma nos equipamentos eletromecânicos.

A palavra checa *robota* deu origem à palavra robô. A palavra original significa "trabalho forçado".

Desde os tempos primórdios o homem tem buscado a libertação de trabalhos mais penosos ou que exigem repetição. Primeiro com o emprego de animais para obter maior força. Desse ponto em diante vieram: máquinas hidráulicas, máquinas a vapor, máquinas com motores a combustão interna, e máquinas elétricas.

A robótica como ciência teve sua origem no século XX, mas obviamente que a automatização de tarefas já vem de longa data, de tempos distantes. Na antiga Alexandria já haviam sido produzido textos sobre pneumática e *automata*, que relatavam a existência de várias máquinas capazes de realizar tarefas repetitivas com movimentos automatizados. Claro que quando se fala em robótica a primeira imagem que nos vem à cabeça é de uma criatura humanoide construída a partir de metal e plásticos, capaz de reproduzir as tarefas humanas com maior velocidade, destreza, força e precisão.

A utilização de robôs para automatizar grandes linhas de produção é uma realidade em todo o mundo. Grandes fábricas, principalmente as indústrias ligadas ao setor de metalurgia e automobilística utilizam robôs para construção de seus produtos, o que traz muitas vantagens ao fabricante, dentre eles a economia de material e padronização do produto em razão da repetição de movimentos com baixo índice de erro.

Outra vantagem é na configuração destes equipamentos. Desde que instalados corretamente, a configuração de um robô pode ser replicada em outro que realiza o mesmo tipo de tarefa. Assim, em certa linha de produção, basta programar um destes equipamentos e transportar o arquivo de configuração para os outros. Os meios mais comuns utilizados para transferir arquivos entre robôs são dispositivos de armazenamento portáteis e redes de computadores.

Os tipos de robôs comumente encontrados na robótica industrial são os que realizam movimentos rotacionais ou lineares, podendo haver uma combinação dos dois. Cada movimento que um robô pode realizar é considerado um grau de liberdade. Em muitos casos eles necessitam de uma série de sensores que fornecem sinais elétricos interpretados pela programação do robô para que ele realize sua tarefa de forma adequada.

A classificação de robôs manipuladores pode ser realizada quanto a estrutura cinemática da seguinte maneira [19]: cartesianos, cilíndricos, esféricos, com articulação horizontal e com articulação vertical.

Robôs Cartesianos – com três das articulações principais do tipo deslizante, ou prismática. Por possuírem grande rigidez mecânica e controle simples são geralmente utilizados para transporte e armazenamento de cargas.

Robôs Cilíndricos – compostos por uma junta rotacional e duas prismáticas, possui rigidez mecânica menor quando comparado ao cartesiano, devido a variação do momento de inércia em diferentes pontos do movimento.

Robôs Esféricos – possuem duas juntas rotacionais, uma na base e outra no ombro, e a terceira linear, proporcionando uma área de trabalho esférica. A maior quantidade de juntas rotacionais implica em menor rigidez mecânica e controle mais complexo em relação ao robô cilíndrico.

Robôs com Articulação Horizontal – conhecidos como robôs SCARA, geralmente utilizados em tarefas de montagem, devido ao movimento linear da terceira junta, as outras duas são rotacionais. A área de trabalho é menor que a dos robôs esféricos.

Robôs com Articulação Vertical – esta é a configuração mais semelhante ao braço humano, composta pro três juntas de rotação. Proporciona maior área de trabalho que os outros robôs, porém com baixa rigidez mecânica e controles complexos.

**Figura 1 - A linha robotizada de montagem de carcaças de automóvel permitiu a democratização do automóvel [8].**

Os processos automatizados tem reflexo direto nos custos de fabricação porque contribuem para menor desperdício de materiais, maior agilidade na produção, melhor controle de qualidade e em geral menores custos operacionais quando comparados com os custos das mesmas tarefas realizadas por seres humanos. A Figura 1 ilustra uma linha de produção de veículos.

## 1.4. ABB – IRB 1400

### 1.4.1. Apresentação do equipamento

O robô da ABB Robotics, modelo IRB 1400 [21], consiste em um braço mecânico industrial de 225 kg, com capacidade de carga de 5 kg, alcance horizontal de 1440 mm, precisão de repetição de ±0,05 mm. São seis eixos de rotação, ou juntas, com velocidade máxima de rotação de 110º/s (1,92 rad/s) nos eixos 1, 2 e 3, e 280º/s (4,89 rad/s) nos eixos 4, 5 e 6. É um mecanismo de fina precisão e rápido, ideal para automação de linhas de produção.

**Figura 2 - Robô ABB IRB-1400 instalado no laboratório de robótica da UFRGS.**

No laboratório onde é ministrada parte da disciplina de Robótica da Escola de Engenharia da UFRGS, o equipamento está montado no chão e tem função didática. A Figura 2 apresenta o equipamento. As disciplinas ministradas têm como objetivo familiarizar os alunos com técnicas de programação e utilização desse tipo de equipamento, tanto para o desenvolvimento acadêmico quanto para pesquisa. O laboratório é de responsabilidade do Grupo de Projeto, Fabricação e Automação Industrial, do Departamento de Engenharia Mecânica, e está localizado no prédio da Escola de Engenharia, Avenida Osvaldo Aranha, 99, sala 109, Centro de Porto Alegre/RS e sua página eletrônica é: http://www.mecanica.ufrgs.br/gpfai/.

### 1.4.2. Funcionamento

O equipamento funciona com um sistema operacional dedicado chamado de BaseWare, e pode ser descrito em cinco propriedades. A primeira é a linguagem e ambiente RAPID que permitem uma boa combinação de simplicidade, flexibilidade e funcionalidade. A segunda é o tratamento das exceções, o que faz com que o equipamento tenha capacidade de se recuperar rapidamente de uma situação de erro, diminuindo assim o tempo de indisponibilidade. A terceira é o controle de movimento, muito amplo com vários modos de

operação que permitem que o IRB 1400 possa ter alta acuracidade nas trajetórias e velocidade, podendo ter velocidades independentes por trecho, forma flexível e intuitiva para especificar curvas ou dobras, otimização automática visando diminuir o tempo de ciclo, sem comprometer a acuracidade, um conceito de coordenadas que facilitam sua operação, ajuste e cópia entre robôs, uso de eixos externos com muita flexibilidade, pode passar por pontos de singularidade, onde dois eixos coincidem, supervisão constante dos movimentos, velocidade e posição, permitindo a rápida detecção de condições anormais, pode trabalhar com cargas que tenham grande inercia e também pode adotar um comportamento oscilatório. A quarta é a segurança, além dos recursos de segurança de hardware o software contempla controle de velocidade, limites de todas as partes do corpo do robô, de novo a supervisão de movimento, autorização de acesso a diferentes grupos de usuários limitam o acesso e garantem que os limites estabelecidos a cada um não sejam excedidos, inclusive limitando velocidade e rotação. A quinta é o sistema de entradas e saídas, um sistema robusto com muitos recursos diferentes, além da entrada de serial e acesso a memória RAM e disquete.

Esse robô possui basicamente duas formas de programação, que é o conhecimento explorado nas aulas do laboratório de robótica. Uma delas é através do *pendant* do equipamento, que permite operá-lo diretamente através de um *joystick* ou ainda entrar com uma rotina de operações. A outra forma é trabalhar em *software* compatível com o equipamento e desenvolver rotinas onde podem ser estabelecidas, trajetórias, posições, velocidades, coordenadas e tudo mais que o robô reconheça como instruções de sua programação. Depois de criar o programa, o desenvolvedor transfere o arquivo em *ASCII* para um disquete que será inserido no drive de disquete do robô, para que o programa seja transferido ao equipamento.

O ambiente de programação isolado do robô é o mais utilizado por vários motivos, tais como, conforto na digitação e visualização do programa, necessidade de desenvolvimento de

vários programas ao mesmo tempo pelos vários alunos das disciplinas que utilizam o equipamento, permite o desenvolvimento da programação à distância, dentre outros.

No caso específico do IRB 1400 instalado no laboratório de robótica da UFRGS, o desenvolvimento à distância tem tido um pequeno entrave porque a tecnologia de discos magnéticos flexíveis já está obsoleta e necessita ser atualizada para que não se perca uma das portas de entrada de dados do equipamento. Então, a substituição da unidade de disquetes por um dispositivo que disponibilize uma porta USB ao usuário pode permitir uma melhor utilização do equipamento, evitando que problemas de transferência de arquivos causem atrasos na realização das tarefas das disciplinas que o utilizam. Uma porta USB permite que além da utilização de dispositivos de memória, que é o objetivo deste trabalho, ainda poderão ser desenvolvidas ferramentas para utilização de tecnologias para conexão a redes de computadores, sejam LAN ou WLAN.

## 1.5. FDD – Floppy Disk Drive

### 1.5.1. Histórico

O primeiro disco magnético flexível foi desenvolvido no final dos anos sessenta e tinha oito polegadas (200 mm) de diâmetro, tornando-se comercializáveis no início dos anos setenta e traziam a nomenclatura "*floppy disk*", e apesar da IBM ter apresentado sua primeira mídia com o nome "Type 1 Diskette" poucos anos depois, a indústria continuou usando o nome "*floppy disk*".

Sabe-se então que o primeiro formato foi o disco de 8 polegadas, porém, em 1976 foi introduzido pela Shugart Associates o disco de 5¼ polegadas e substituiu seu antecessor na maioria das aplicações, mesmo ano em que surgiram discos dupla face. Logo depois surgiu a possibilidade de aumentar a capacidade de armazenamento com a dupla densidade [3]. Na década de oitenta as limitações do disco de 5¼ ficaram claras e então surgiram algumas propostas de substituição em tamanhos de 2, 2½, 3 e 3½ polegadas. No final dos anos oitenta

o discos de 5¼ começaram a ser substituídos por discos de 3½ polegadas, que incluíam entre outras melhorias, invólucro rígido com capa de metal deslizante sobre a área de leitura e escrita, que ajudavam a proteger o frágil disco magnético de poeira e eventuais danos físicos.

Nos anos oitenta e noventa esses discos foram largamente difundidos com o uso de computadores pessoais, servindo para distribuição de software, transferência de dados e criação de cópias de segurança. Antes dos discos rígidos estarem disponíveis no mercado, os *floppy disks* serviam para guardar o sistema operacional dos PC's. Em 1996 foram estimados mais de 5 bilhões de discos em uso no mundo. Nesse momento os distribuidores de grandes pacotes de software começaram gradualmente a utilizar CD-ROM. Surgiram algumas soluções com discos de alta capacidade, mas incompatíveis mecanicamente com a infraestrutura existente. Em 1998 a Apple lançou o iMac que trazia apenas o drive de CD-ROM, sem a unidade de disquetes. Então foram lançados os CD's graváveis, com capacidade maior que a dos novos discos magnéticos, e compatíveis com a infraestrutura do CD-ROM drive, tornando a tecnologia de discos magnéticos obsoleta. Ainda outras tecnologias que ajudaram a decretar o fim do uso de disquetes foram as redes e depois os dispositivos de memória *flash* através de portas USB e atualmente servidores de *on-line storage*. Depois de 2010 os fabricantes de placas mãe deixaram de disponibilizar controladoras de disquete em seus produtos.

### 1.5.2. Padrões

O tamanho dos disquetes sempre foi dado em polegadas mesmo nos países em que se usa o sistema métrico de medidas. A capacidade de memória foi dada inicialmente em quilobytes (kB) e posteriormente em megabytes (MB).

Abaixo segue a Tabela 1 com as capacidades e formatos dos disquetes ao longo do tempo.

**Tabela 1 - Tabela histórica de lançamento de modelos de discos magnéticos.**

| Formato do Disco | Ano de lançamento | Capacidade de armazenamento formatada | Capacidade anunciada |
|---|---|---|---|
| 8-inch: IBM 23FD (read-only) | 1971 | 79.75 kB | Não anunciado comercialmente |
| 8-inch: Memorex 650 | 1972 | 175 kB | 1.5 megabit unformatted |
| 8-inch: SSSD IBM 33FD / Shugart 901 | 1973 | 237.25 kB | 3.1 megabit unformatted |
| 8-inch: DSSD IBM 43FD / Shugart 850 | 1976 | 500.5 kB | 6.2 megabit unformatted |
| 5¼-inch (35 track) Shugart SA 400 | 1976 | 87.5 kB | 110 kB |
| 8-inch DSDD IBM 53FD / Shugart 850 | 1977 | 980 kB (CP/M) - 1200 kB (MS-DOS FAT) | 1.2 MB |
| 5¼-inch DD | 1978 | 360 or 800 kB | 360 kB |
| 5¼-inch Apple Disk II (Pre-DOS 3.3) | 1978 | 113.75 kB (256 byte sectors, 13 sectors/track, 35 tracks) | 113 kB |
| 5¼-inch Atari DOS 2.0S | 1979 | 90 kB (128 byte sectors, 18 sectors/track, 40 tracks) | 90 kB |
| 5¼-inch Apple Disk II (DOS 3.3) | 1980 | 140 kB (256 byte sectors, 16 sectors/track, 35 tracks) | 140 kB |
| 3½-inch HP single sided | 1982 | 256×16×70 = 280 kB | 264 kB |
| 5¼-inch Atari DOS 3 | 1983 | 127 kB (128 byte sectors, 26 sectors/track, 40 tracks) | 130 kB |
| 3-inch | 1982 | 360 kB | 125 kB (SS/SD), 500 kB (DS/DD) [28] |
| 3½-inch SS (DD at release) | 1983 | 360 kB (400 on Macintosh) | 500 kB |
| 3½-inch DS DD | 1984 | 720 kB (800 on Macintosh, 880 on Amiga) | 1 MB |
| 5¼-inch QD | | 720 kB | 720 kB |
| 5¼-inch RX50 (SSQD) | Meados de 1982 | 400 kB | 400 kB |
| 5¼-inch HD | 1982 | 1155 kB | 1.2 MB |
| 3-inch DD | 1984 | 720 kB | Desconhecido |
| 3-inch Mitsumi Quick Disk | 1985 | 128 to 256 kB | Desconhecido |
| 2-inch | 1989 | 720 kB | Desconhecido |
| 2 ½-inch | 1986 | Desconhecido | Desconhecido |
| 5¼-inch Perpendicular | 1986 | 10 MB | Desconhecido |
| 3½-inch HD | 1987 | 1440 kB (1760 kB no Amiga) | 1.44 MB (2.0 MB unformatted) |
| 3½-inch ED | 1987 | 2880 kB | 2.88 MB |
| 3½-inch Floptical (LS) | 1991 | 20385 kB | 21 MB |
| 3½-inch Superdisk (LS-120) | 1996 | 120.375 MB | 120 MB |
| 3½-inch Superdisk (LS-240) | 1997 | 240.75 MB | 240 MB |
| 3½-inch HiFD | 1998/99 | 150/200 MB | 150/200 MB |

Abreviaturas: **SD** = Single Density; **DD** = Double Density; **QD** = Quad Density; **HD** = High Density; **ED** = Extended Density; **LS** = Laser Servo; **HiFD** = High capacity Floppy Disk; **SS** = Single Sided; **DS** = Double Sided

Capacidade de armazenamento formatada é o tamanho total de todos os setores do disco.
Para 5¼-polegadas e 3½-polegadas as capacidades cotadas são dos sistemas de vendas.
Capacidade anunciada é a capacidade, tipicamente não formatada, pelo vendedor OEM original do produto, ou no caso de produto da IBM, o primeiro OEM posterior. Outros formatos podem ter maior ou menor capacidade para os mesmos *drives* e discos.

As características físicas dos disquetes são apresentadas na Tabela 2.

**Tabela 2 - Características físicas de discos magnéticos.**

| Tamanho | Densidade | Trilhas | Trilhas Por Polegada | Bites Por Polegada | Coercividade | Capacidade Não-formatada Por lado |
|---|---|---|---|---|---|---|
| 3½-inch | single | 40 | 67.5 | | 600 Oe | 250 KB |
| | double | 80 | 135 | 8.717 | 600 Oe (300 Oe) | 500 KB |
| | high | 80 | 135 | 17.434 | 750 Oe (600 Oe) | 1000 KB |
| | extended | 80 | 135 | | 900 Oe | 2000 KB |
| 5¼-inch | single/double | 40 | 48 | 5.876 | 300 Oe | 250 KB |
| | quad | 80 | 96 | 5.876 | 300 Oe | 500 KB |
| | high | 80 | 96 | 8.646 | 600 Oe | 750 KB |
| 8-inch | single/double | 77 | 48 | | 300 Oe | 1000 KB |

**Figura 3 - Unidades de disquetes de 8, 5¼ e 3½polegadas.**


**Figura 4 – Disquetes de 8, 5¼ e 3½polegadas.**

### 1.5.3. Funcionamento

Uma unidade de disquete, Figura 3, possui um motor que gira o suporte de apoio ao disco magnético flexível a uma determinada velocidade enquanto um mecanismo operado por um motor de passo move as cabeças de gravação/leitura ao longo da superfície do disquete, Figura 4. Ambas as operações de leitura e escrita exigem que a mídia esteja em rotação e a cabeça em contato com o disco, essa ação é controlada por um solenoide de carga de disco. Para escrever os dados, uma corrente é enviada através de uma bobina na cabeça de escrita enquanto o disco está em rotação. A cabeça do campo magnético alinha as partículas magnéticas diretamente abaixo da cabeça em relação ao disco. Quando a corrente é invertida as partículas são alinhadas em sentido oposto para codificar os dados digitalmente. Para ler os dados as partículas magnéticas do disco induzem uma pequena corrente na bobina da cabeça quando passam por ela. Esta pequena corrente é convertida em tensão e amplificada, depois disso é enviada a controladora de disquete que converte a sequência de pulsos contida no disco em dados, verifica a existência de erros e envia esses dados ao computador hospedeiro. Então, basicamente, o que a unidade de disquete faz em um sentido é extrair dados de um disco magnético e comunicar-se com a controladora de forma que esses dados sejam enviados

ao processador central, em outro sentido a unidade de disquete comunica-se com a controladora para realizar a gravação de dados enviados pelo processador central em um disco magnético.

### 1.5.4. Sinalização

Para realização das operações de leitura e escrita de dados em disco magnético é necessário que a controladora da unidade de disquete (FDC) e a unidade de disquete (FDD) se comuniquem de maneira inteligível. Para tanto existe a padronização de uma série de sinais elétricos que interligam FDC e FDD através de um conector, mecanicamente padronizado, com 34 pinos. Os sinais disponíveis nesse conector são caracterizados pelo sentido da informação que carregam. Alguns desses sinais têm origem na FDC e servem para configurar os parâmetros da FDD, acionar dispositivos ou enviar a sequência de dados que será gravada no disco magnético. No caso contrário, quando a FDD envia sinais à FDC, são sinais que indicam sinalizações de disponibilidade ou a sequência de dados lidos do disco magnético. A Figura 5 ilustra a comunicação da FDC com a FDD [7].

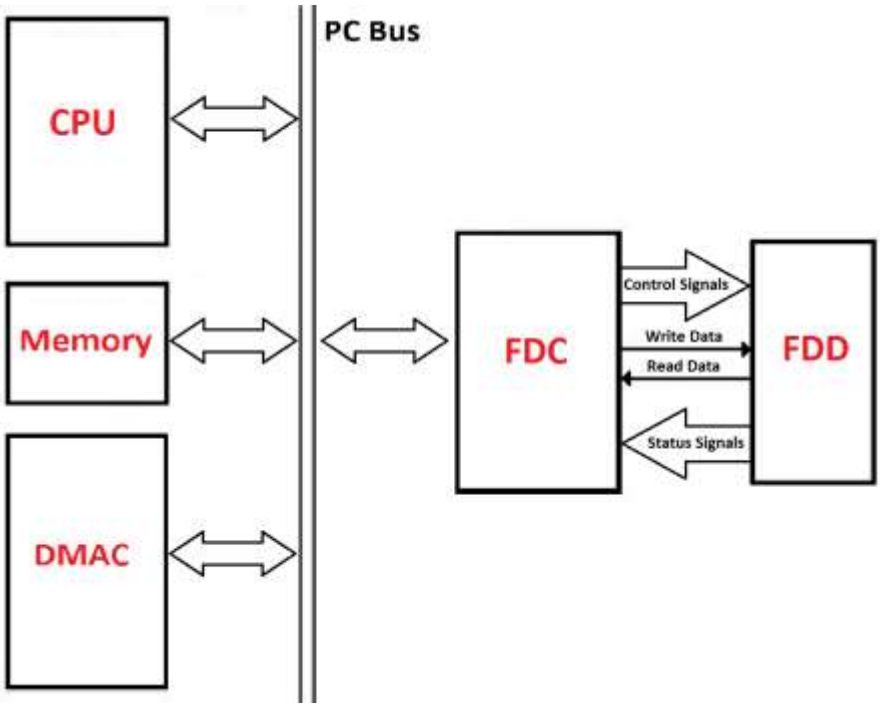


**Figura 5 - Diagrama de blocos que mostra a comunicação da FDC com a CPU e com a FDD.**

A distribuição dos sinais de comunicação que trafegam no conector padrão PC34 da FDC, visualizado na Figura 6, estão descritos na Tabela 3, onde também são apontados os sentidos do fluxo de comunicação.

**Tabela 3 - Tabela de sinais do conector padrão de 34 pinos da FDC.**

| | Sinal | | |
|---|---|---|---|
| **# Pino** | **Nome** | **Sentido FDC-FDD** | **Descrição** |
| 2 | $\overline{DENSEL}$ | → | Seleção de densidade |
| 4 | N/C | * | Não conectado |
| 6 | N/C | * | Não conectado |
| 8 | $\overline{INDEX}$ | ← | Índice |
| 10 | $\overline{MOTEA}$ | → | Ativar motor do drive A |
| 12 | $\overline{DRVSB}$ | → | Seleção do drive B |
| 14 | $\overline{DRVSA}$ | → | Seleção do drive A |
| 16 | $\overline{MOTEB}$ | → | Ativar motor do drive B |
| 18 | $\overline{DIR}$ | → | Direção |
| 20 | $\overline{STEP}$ | → | Passo da cabeça leitura/escrita |
| 22 | $\overline{WDATA}$ | → | Escrita de dados |
| 24 | $\overline{WGATE}$ | → | Ativação do modo de escrita |
| 26 | $\overline{TRK00}$ | ← | Trilha 0 |
| 28 | $\overline{WPT}$ | ← | Proteção de escrita |
| 30 | $\overline{RDATA}$ | ← | Leitura de dados |
| 32 | $\overline{SIDE}$ | → | Seleção de cabeça |
| 34 | $\overline{DSKCHG}$ | ← | Disco ejetado |
| Ímpares | GND | * | * |

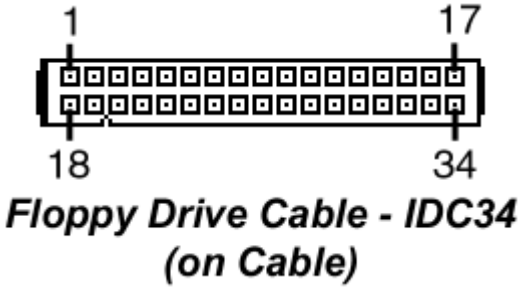


**Figura 6 - Conector de 34 pinos do Floppy Disk Drive [6].**

Para melhor entendimento de cada um desses sinais é necessário que se saiba que são sinais em nível TTL e que os sinais são considerados ativos quando o sinal elétrico é 0 V. Isso significa que o nível lógico ‘1’ representa um sinal elétrico de 0 V em corrente contínua e o

nível lógico ‘0’ representa um sinal elétrico de 5V também em corrente contínua. No conector padrão do FDD de 34 pinos, todos os pinos ímpares são ligados ao ponto de terra do circuito, por esse motivo, nenhum desses pinos aparece listado na tabela de sinais do conector.

A funcionalidade de cada um desses sinais, apresentados pelo nome, conforme Tabela 3 é:

$\overline{DENSEL}$ – realiza a seleção entre dupla densidade e alta densidade do disco magnético e ao mesmo tempo ajusta a corrente das bobinas da cabeça de escrita. Esse sinal já não é mais utilizado nas unidades de disquete mais modernas, que não dependem disso para reconhecer a taxa de dados do disco magnético.

$\overline{INDEX}$ – gerado pela detecção do marcador de índice. Este sinal inicia e termina a transferência do controle de dados para o FDD durante a formatação, gerado pelo FDD no início de cada trilha/segmento durante as operações de leitura e escrita.

$\overline{MOTEA}$ – liga o motor da unidade A. A unidade é definida por *jumper* contido na unidade de disquete. O barramento do PC34 suporta linhas de motores individuais, mas requer a inversão de fios no cabo para esse propósito e dessa forma define unidade A ou B.

$\overline{DRVSB}$ – seleciona a unidade de disquete B. Nas unidades de disquete de PC’s, não há jumpers para seleção do driver, eles são sempre reconhecidos como B. A seleção da unidade B é feita quando não há inversão dos cabos dos pinos 10 a 16 do conector.

$\overline{DRVSA}$ – seleciona a unidade de disquete A. Nas unidades de disquete de PC’s, não há jumpers para seleção do driver, eles são sempre reconhecidos como B. A seleção da unidade A é feita pela inversão dos cabos dos pinos 10 a 16 do conector.

$\overline{MOTEB}$ – liga o motor da unidade B. A unidade é definida por *jumper* contido na unidade de disquete. O barramento do PC34 suporta linhas de motores individuais, mas requer a inversão de fios no cabo para esse propósito e dessa forma define unidade A ou B.

$\overline{DIR}$ – em nível alto quando o movimento é para os cilindros externos (números inferiores) e nível baixo quando o movimento é para os cilindros internos (números superiores).

$\overline{STEP}$ – cada pulso desse sinal move a cabeça em uma unidade de passo.

$\overline{WDATA}$ – Sinal seria de dados, um pulso para cada fluxo de transição para gravação, a borda de descida indica a transição.

$\overline{WGATE}$ – em nível alto quando aos dados são lidos da unidade de disco e em nível baixo quando os dados são escritos para a unidade de disco.

$\overline{TRK00}$ – detecção da trilha zero do disco magnético.

$\overline{WPT}$ – indica que o disco magnético inserido na unidade está protegido contra gravação. O estado é enviado à controladora.

$\overline{RDATA}$ – Sinal seria de dados, um pulso para cada fluxo de transição, a borda de descida indica a transição.

$\overline{SIDE}$ – sinal que seleciona o cabeçote da unidade de disco. Em nível alto quando a cabeça 0 (superior) é usada e em nível baixo quando a cabeça 1 (inferior) é acionada.

$\overline{DSKCHG}$ – sinal de troca de disco. Será acionado quando não houver disco dentro da unidade. Permanecerá acionado até que a controladora desative. A desativação deste comando só acontece se houver um disco dentro da unidade, e ainda se o disco estiver válido, testando seu estado imediatamente. Geralmente a controladora desativa este comando enquanto busca por uma unidade de disco. Entretanto, em algumas unidades é suficiente a ativação ou desativação. Algumas unidades atualizam o nível lógico com certo atraso, ou seja, depois da seleção demoram um pouco para trocar para o devido estado. Em outras unidades é possível desativar este sinal procurando pela trilha 1 (sem ruído), ainda em outras é necessário procurar por uma trilha existente. Este sinal precisa ser explicitamente desativado para que seja possível ter certeza de que a unidade não perca uma troca de disco.

Nas unidades de disco de 3½ polegadas não há jumper para definição da identificação da unidade, se é A ou B, todas elas são reconhecidas como B. para que a seleção de unidade seja realizada, há uma inversão no cabo que conecta a controladora à unidade de disco, dos pinos 10 ao 16, de forma que a unidade possa ser reconhecida como a unidade A. Na Figura 7 é apresentada uma foto de um cabo capaz de conectar duas unidades de disquete a uma controladora e é possível ver a inversão do cabo para conexão da unidade A.

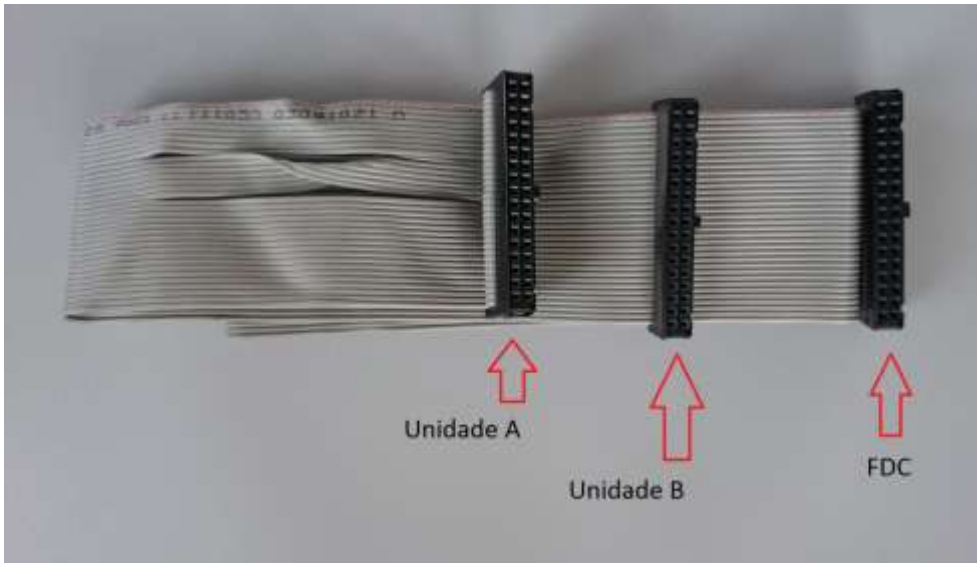


**Figura 7 - Cabo para unidades de disquete A e B, com inversão dos pinos 10 a 16.**

Os sinais, Motor Enable ($\overline{MOTEx}$) e Drive Select ($\overline{DRVSx}$) indicam à unidade de disque de algum tempo para que a rotação do disco estabilize nas 300 rotações por segundo, assim, a controladora pode acionar o sinal de motor antes e só ativar o drive assim que estiver também preparado para receber e enviar informação. A utilização dos dois sinais permite ainda a utilização de duas unidades de disquetes de forma simultânea em que os motores são mantidos ativos enquanto que o sinal de seleção de drive é alternado.

A controladora de drive de disquetes (FDC) controla a cabeça de leitura do drive através de 3 sinais que servem para selecionar qual a faixa pretendida: *Head Step* ($\overline{STEP}$), *Head Direction* ($\overline{DIR}$) e *Head Select* ($\overline{SIDE}$). A localização da cabeça de leitura/escrita não é transmitida do drive de disquetes ao controlador a não ser quando está sobre a faixa 0. Neste caso o sinal Track0 ($\overline{TRK00}$) é ativado informando a controladora. A partir daqui, a

controladora deverá contar o número de impulsos que envia à unidade de disquetes de forma a selecionar a faixa pretendida. A unidade de disquetes move a cabeça de leitura ao longo do raio da superfície do disquete a cada impulso do sinal $\overline{STEP}$ na direção definida pelo sinal $\overline{DIR}$. Se $\overline{DIR}$ não estiver ativo a cabeça é movida na direção da borda do disco em direção à faixa 0. Caso contrário, sobe nas faixas até à faixa 79 no interior do disco. Com o sinal $\overline{DIR}$ inativo, o sinal de $\overline{STEP}$ é repetidamente impulsionado até que o sinal $\overline{TRK00}$ seja ativado.

Dependendo do sinal $\overline{SIDE}$, a cabeça do respetivo lado é ativada e as operações de leitura ou escrita são efetuadas com essa cabeça. Na verdade, a ação de leitura não necessita de qualquer comando por parte da controladora. A cabeça de leitura, ao encostar à superfície do disco, envia a informação que encontra de forma contínua. A cada inversão no campo magnético, é emitido um impulso no pino *Read Data* ($\overline{RDATA}$). É da controladora a tarefa de interpretar a informação quando esta precisar e estiver disponível. Devido à velocidade de rotação do disco, a informação presente nas faixas é repetida a cada 200 milissegundos. Para ajudar a controladora a sincronizar esta informação, um sinal Index ($\overline{INDEX}$) é ativado quando a cabeça de leitura passa sobre o preâmbulo da faixa que precede o primeiro sector.

Caso a controladora pretenda ler ou gravar dados, mas a unidade de disquetes não tenha um disquete inserido, a unidade sinaliza à controladora deste fato através da ativação do sinal *Disk Change* ($\overline{DSKCHG}$). Se a controladora tentar escrever dados num disquete que tenha a proteção contra escrita ativa, a unidade de disquetes também sinaliza o controlador através do sinal *Write Protect* ($\overline{WPT}$).

### 1.6. Estrutura da trilha e setores do disquete

Um disquete tem dentro de sua estrutura física, na superfície magnetizada, uma organização de endereçamento de dados através da separação de trilhas e dentro destas trilhas os setores de armazenamento. É necessário que seja possível ler ou escrever os dados no

disquete de maneira organizada dentro dos setores e com a garantia de interoperabilidade entre os disquetes.

Aqui neste documento trataremos apenas da organização do disquete com a modulação MFM.

Cada lado do disco possui então diversas trilhas (no formato em análise são 80) distribuídas concentricamente, e cada uma delas é composta por diversos setores onde são alocados os dados. Os setores são divididos em uma série de campos.

O primeiro campo é o preâmbulo, aos quais se seguem uma sequência de 18 setores compostos pelos respetivos campos de identificação (ID) e de Dados. A trilha é encerrada com um campo de preenchimento (Gap). O esquema de distribuição está representado na Tabela 4.

**Tabela 4 - Organização dos campos na trilha.**

| Preâmbulo | Setor 1 | | Setor 2 | | ... | Setor 17 | | Setor 18 | | Gap |
|---|---|---|---|---|---|---|---|---|---|---|
| | ID | Dados | ID | Dados | | ID | Dados | ID | Dados | |

Ao final do último setor (o 18º) segue-se novamente o preâmbulo. Devido ao formato circular do disco magnético e à sua velocidade de rotação, o preâmbulo e os dados são reenviados a cada 200 ms.

O sinal de relógio é inserido no próprio sinal, então é necessário detectar o sincronismo e extrair os dados do sinal. Nas controladoras de drives de disquete isto é realizado através da aplicação do PLL Data Separator (Separador de Dados PLL) que detecta e extrai os dados relevantes.

Para tornar certas operações de escrita e leitura menos complicadas, alguns disquetes podem apresentar a ordem dos setores alterada. Se, por exemplo, numa operação de leitura o computador hospedeiro solicita leitura dos setores 1 e 2, mas não tiver capacidade de processamento suficiente para processar rapidamente a informação recebida, pode pedir para

ler o setor 1 e só depois o setor 2. Por causa da rotação constante do disco magnético, o computador teria de esperar o tempo correspondente a uma rotação inteira para poder ler o setor seguinte. Com a ordem dos setores alterada para, por exemplo, 1 8 6 4 2 9 7 5 3, o tempo de espera entre os setores 1 e 2 é diminuído. Esta técnica é chamada de *interleaving* (entrelaçamento), mas caiu em desuso por conta do aumento da capacidade de processamento dos computadores.

A formatação dos campos que constituem uma trilha será ilustrada com auxílio de algumas tabelas. Como a maioria dos campos consiste em repetição de dados, as tabelas descrevem estas características. Na Tabela 5, o "Campo 1" é composto pela repetição do octeto "Byte" tantas vezes como especificado em "Comprimento". Em seguida, o "Campo 2", que é composto por outras sequências de bytes com determinadas repetições.

**Tabela 5 - Descrição das características das trilhas.**

| | | |
|---|---|---|
| Nome do Campo 1 | Comprimento<br>Byte | |
| Nome do Campo 2 | Comprimento | Comprimento |
| | Byte | Byte |

A descrição dos diversos campos que constituem o formato da trilha, começando pelo Preâmbulo será apresentada a seguir. Os bytes são apresentados na notação hexadecimal, sinalizada pelo prefixo "0x".

Preâmbulo: A unidade de disquete envia o conteúdo da trilha de forma cíclica e a controladora necessita processar essa informação para poder extrair os dados úteis. Isto é realizado através do reconhecimento de uma série de sinais pré-formatados que indicam o início da trilha. A controladora efetua uma pesquisa por estes sinais e, após obter a sincronização, poderá prosseguir com a extração e verificação.

O preâmbulo é o primeiro dos campos de sinais de controle e sincronismo que antecede o primeiro setor e marca o início da trilha, sendo constituído pelos sinais indicados na Tabela 6.

**Tabela 6 - Campo do Preâmbulo da trilha.**

| | | |
|---|---|---|
| GAP4a | 80 | |
| | 0x4E | |
| SYNC | 12 | |
| | 0x00 | |
| INDEX MARK | 3 | 1 |
| | 0xC2* | 0xFC |
| GAP | 50 | |
| | 0x4E | |

Para que a controladora encontre os dados que pretende, necessita sincronizar os diversos campos de controle e de dados. Para diferenciar a sequência de dados do preâmbulo dos dados úteis, o INDEX MARK possui um sinal que não respeita a codificação MFM usual, uma violação do código. Os sinais de relógio usados para gerar o sinal MFM são modificados, suprimindo um pulso, gerando uma sequência de impulsos que não pode ser encontrada durante o envio do campo de dados.

O octeto 0xC2 modulado normalmente no campo de dados está representado na Figura 8, com a comparação de quando é alterado pelo de relógio para ser utilizado como sinal de sincronismo. Verifica-se a falta do bit 5, violando assim a modulação MFM usual, garantindo que não será confundido com um octeto de dados. Nas tabelas esta alteração é indicada pela presença do símbolo "*" ao lado do octeto.

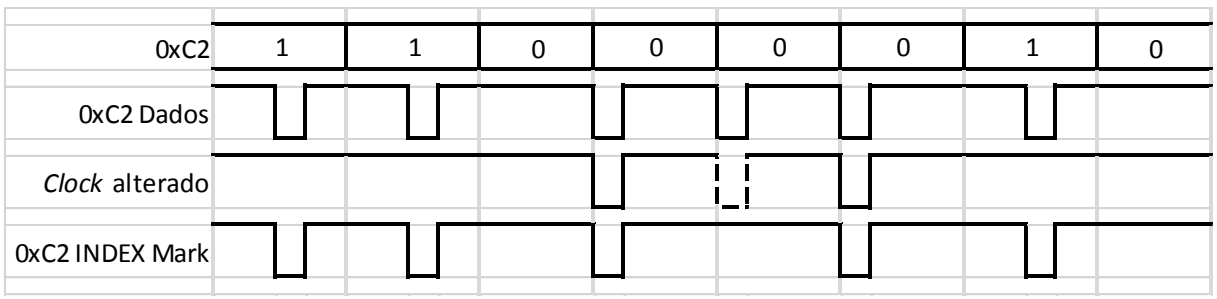


**Figura 8 - Comparação da codificação de 0xC2 e 0xC2*.**

Então, cada setor composto por dois campos: Campo de Identificação (ID) e Campo de Dados.

ID: as informações armazenadas no cabeçalho ID descrevem o setor. Este campo contém o número da trilha (ou cilindro), a cabeça e o número de identificação. A estrutura deste campo pode ser encontrada na Tabela 7. Existe também o comprimento do setor. Este valor é calculado através da expressão (1).

$$Comprimento\ do\ Setor = log_2(quantidade\ de\ bytes\ do\ setor) - 7 \quad (1)$$

Em um setor de 512 bytes o campo "Comprimento do Setor" é igual a 2.

**Tabela 7 - ID: Campo de identificação de setor.**

| | | |
|---|---|---|
| SYNC | 12 | |
| | 0x00 | |
| ID ADD Mark | 3 | 1 |
| | 0xA1* | 0xFE |
| Cilindro | 1 | |
| | 0xXX | |
| Cabeça | 1 | |
| | 0xXX | |
| Número do Setor | 1 | |
| | 0xXX | |
| Comprimento do Setor | 1 | |
| | 0xXX | |
| CRC | 2 | |
| | 0xXX | |
| GAP2 | 22 | |
| | 0x4E | |

Neste campo, no parâmetro ID *Address Mark*, também é utilizada a violação no sinal de relógio MFM de forma a que este possa ser distinguido dos octetos de dados.

Dados: é neste campo que são armazenados os dados úteis. É precedido de campo de sinalização que também contém um sinal que viola a codificação MFM e é finalizado por um campo de controle de erros (CRC) como pode ser consultado na Tabela 8.

**Tabela 8 - Campo de dados do setor.**

| | | |
|---|---|---|
| SYNC | 12 | |
| | 0x00 | |
| Data Mark | 3 | 1 |
| | 0xA1* | 0xFB |
| Data | 512 | |
| | 0xXX | |
| CRC | 2 | |
| | 0xXX | |
| GAP3 | 84 | |
| | 0x4E | |

GAP (Preenchimento): após os 18 setores, o espaço restante até ao final da trilha é ocupado por um campo de preenchimento com 510 octetos de comprimento representado na Tabela 9.

**Tabela 9 - GAP: campo de preenchimento.**

| | |
|---|---|
| GAP4b | 510 |
| | 0x4E |

Devido à propriedade cíclica do disco magnético, consequentemente, do envio dos dados, o campo GAP4a do preâmbulo é alocado após o GAP4b do campo de preenchimento.

No total, cada trilha, acolhe mais do que 18 campos de dados com 512 bytes cada. Com o preâmbulo e os sinais de controle dos setores, uma trilha totaliza 12500 bytes, o que pode ser verificado na Tabela 10. Como cada byte demora 16 µs (dezesseis microssegundos) para ser transmitido, o ciclo é repetido a cada 200 ms (duzentos milissegundos).

**Tabela 10 - Comprimento total de uma trilha.**

| | 1 dos 18 setores | | 17 setores | | |
|---|---|---|---|---|---|
| Preâmbulo | Campo de identificação | Campo de dados | ... | Preenchimento | Total |
| 146 | 44 | 614 | 17 x (44 + 614) | 510 | 12500 |

### 1.7. CRC

Para garantir a validade da informação contra erros de escrita ou leitura, são utilizados campos de verificação de erros CRC - *Cyclic Redundancy Check* (verificação de redundância cíclica). Estes campos estão presentes nos campos de identificação do setor e nos campos de dados. O algoritmo utilizado para cálculo de CRC é conhecido como CRC-16-CCITT. O polinômio gerador g(x) é apresentado na equação (2) e utiliza como semente a palavra 0xFFFF.

$$g(x) = 1 + x^5 + x^{12} + x^{16} \quad (2)$$

O código CRC segue imediatamente ‘após a sequência de dados a serem protegidos. Deste modo, o cálculo polinomial dos dados mais a sequência de CRC produzirá um resultado composto por zeros. Caso este resultado não seja zero, a conclusão é de que existiu um erro na transmissão os dados. Como exemplo, o código de verificação de um Campo de Identificação de um setor pode ter a seguinte sequência, representada na Tabela 11, constituída pelos campos de sincronismo, identificação e detecção de erros.

**Tabela 11 - Campo de Identificação de um setor.**

| ID Add Mark | | | | Cilindro | Cabeça | Setor | Comprimento | CRC | |
|---|---|---|---|---|---|---|---|---|---|
| 0xA1 | 0xA1 | 0xA1 | 0xFE | 0x00 | 0x00 | 0x01 | 0x02 | 0xCA | 0x6F |

### 1.8. Campo de Dados

Para que a informação seja útil e facilmente acessível ao usuário, ela precisa ser primeiramente entendida pelo sistema operacional do computador tem a unidade de disquete instalada. Para isso, um disquete é formatado com um sistema de arquivos de dados. O sistema de arquivos utilizados depende do ambiente em que os disquetes são utilizados. Os dados desta formatação são incluídos nas seções do campo de dados, fazendo com que a capacidade do disquete diminua. Devido ao tamanho diminuto dos disquetes, são também utilizados sistemas de arquivos que não ocupam um volume grande de dados. No caso dos

computadores pessoais com os sistemas operacionais DOS ou Windows, é utilizado o sistema de formatação FAT12. Este sistema de arquivos, apesar de ter um tamanho pequeno, ocupa alguns dos setores de dados reduzindo a capacidade efetiva para arquivos úteis. No total, dos 2 MB de dados escritos nos disquetes, apenas 1.44 MB estão disponíveis.

## 1.9. FAT12

### 1.9.1. Apresentação do sistema de organização

A FAT12 [11] é uma versão de FAT, um sistema de arquivos, que foi utilizada pela primeira vez no sistema operacional DOS. Hoje em dia empregado no Windows XP (formatação de dispositivos de armazenamento portáteis de câmeras digitais para placas de baixa capacidade). Por ser um sistema de arquivos simples, a FAT12 ainda é utilizada pelo Windows (mesmo com o sistema NTFS) para formatar disquetes. O sistema FAT12 possui um limite máximo para uma partição de 16 MB, com cluster de 512 bytes, 1K, 2K e 4K.

É um sistema que funciona através de uma tabela que mapeia os blocos (clusters) que armazenarão as informações dos arquivos. Quando um arquivo é gravado num disquete, por exemplo, a FAT12 o divide blocos do tamanho de um cluster. Assim, um arquivo pode ocupar vários blocos, mas eles não estarão necessariamente em sequência. Os blocos de determinado arquivo podem estar em várias posições diferentes, daí a razão da existência da tabela de mapeamento.

A FAT12 é um sistema de organização de arquivos que pode ser utilizado tanto em disquetes como em quaisquer meio de gravação de arquivos. O que caracteriza a FAT12 é a entrada de 12 bits na tabela de mapeamento. Além disso, o padrão FAT12 define estruturas para armazenamento dos nomes de arquivos e diretórios. Os nomes são constituídos por até 8 caracteres, um caráter separador constituído por um ponto (.) e uma extensão do nome com até 3 caracteres. A grande vantagem de volumes formatados com FAT é que são acessíveis pelos sistemas operacionais DOS, pelo Windows e pelo OS/2.

Com o surgimento de dispositivos de armazenamento com maior capacidade e cada vez mais sofisticados, o sistema FAT12 foi sofrendo alterações. Essas alterações foram necessárias porque a FAT12 era limitada à determinada capacidade de armazenamento, operando com tamanho máximo de 2 GB. Assim, num disco de 5 GB, seria necessário dividi-lo em 3 partições. Outro fato é que a FAT12 apresentava problemas com informações acima de 512 MB.

### 1.9.2. Funcionamento

Ao trabalharmos com disquetes ou discos rígidos é necessário prepará-los através de uma formatação física. Este processo divide os discos em trilhas (cilindros) e setores (subdivisões de cada trilha). Depois é necessária a formatação lógica, que nada mais é do que "instalar" o sistema de arquivos no dispositivo de armazenamento.

O sistema de arquivos FAT12 não trabalha diretamente com cada setor, mas sim com um grupo de setores. Esse grupo é chamado de cluster (ou unidade de alocação). Se por exemplo, um disco com setor de 512 bytes, tiver 5 KB de tamanho, ele terá 10 setores e 5 clusters, se cada cluster ocupar dois setores. Sendo assim, quando a FAT precisar acessar um determinado setor, primeiro ela descobre em qual cluster ele se encontra. Tanto a FAT12 quanto a FAT32 trabalham com este princípio.

### 1.9.3. Dimensão de um cluster

O sistema FAT12 exige que cada cluster do disco seja usado somente para um único arquivo, ou seja, não pode haver informações sobre mais de um arquivo num mesmo cluster. Isso pode parecer óbvio, mas gera um problema: desperdício. Na suposição que seja necessário guardar um arquivo de 5 KB num disquete, que este disquete tenha 8 KB de espaço e dois clusters de 4 KB, um cluster ocuparia 4 KB do arquivo, enquanto o outro cluster ocuparia apenas 1 KB. Como o cluster só pode trabalhar com um arquivo, haveria desperdício de 3 KB. Agora, em vez de clusters com 4 KB, sejam clusters com 2 KB. Dessa maneira, 3

clusters seriam usados e o desperdício seria de 1 KB. No entanto, sobraria um cluster com 2 KB, que poderia usado por outro arquivo.

### 1.10. MSD – Mass Storage Device

Um *Mass Storage Device*, ou dispositivo de armazenamento massivo, é qualquer dispositivo de armazenamento que torne capaz a possibilidade de armazenar e portar grandes quantidades de dados através de computadores, servidores e dentro do ambiente de TI. MSDs são dispositivos de mídia portáteis que proporcionam a interface de armazenamento de dados dos computadores, tanto internamente quanto externamente.

Um dispositivo de armazenamento massivo também pode ser descrito como um dispositivo de armazenamento auxiliar. O termo é comumente usado para descrever dispositivos de armazenamento massivo USB.

MSD é primariamente relacionado a dispositivos de armazenamento que possibilitam uma capacidade de armazenamento consistente e permanente. Um MSD é conectado a um computador ou servidor através de uma interface de transferência de dados, tais como SCSI, USB ou mesmo uma rede de computadores, como uma rede de armazenamento. Alguns dos dispositivos mais comuns incluem disquetes, discos óticos, discos rígidos magnéticos, fitas magnéticas, RAID ou dispositivos USB. Atualmente, os dispositivos MSD tipicamente possuem a capacidade entre alguns gigabytes até pentabytes de dados. Os MSDs internos geralmente não podem ser removidos, entretanto os externos podem ser facilmente removidos, transportados e conectados em outros computadores. A Figura 9 representa um dispositivo de armazenamento massivo para conexão em porta USB.


Figura 9 - Dispositivo de Armazenamento Massivo [9].

## 1.11. USB – Universal Serial Bus

Atualmente a USB é o tipo mais comum de porta utilizada por computadores. Essa porta pode ser usada para conectar teclados, *mouse*, controles de jogos, impressoras, scanners, câmeras digitais, dispositivos de armazenamento de dados portáteis, dentre outros.

Atualmente a porta USB é mais rápida, por questões de desenvolvimento tecnológico, que as portas disponíveis antigamente, tais como as portas seriais e paralelas. As especificações da USB 1.1 permitem que sejam suportadas taxas de transferência de dados de até 12 Mbps, já a versão 2.2 pode elevar o valor máximo da taxa de transferência de dados para até 480 Mbps. Desde que a USB foi introduzida em 1997, a tecnologia não teve uma difusão realmente significativa até a introdução do Apple iMac no final do ano de 1998, que disponibilizava exclusivamente portas USB para conexão de dispositivos periféricos. Na época, a utilização da nova porta pela Apple parecia certa ironia, considerando que a criação da USB foi projetada pela Intel, Compaq, Digital e IBM. Com o passar dos anos as portas USB se tornaram largamente utilizadas como uma interface cruzada entre ambos os sistemas de Macs e PCs.

## 1.12. Microcontroladores

Microcontroladores são componentes eletrônicos programáveis, capazes de simular inteligência para realizar o controle lógico de processos lógicos.

Diz-se que são dotados de inteligência programável porque dentro do componente possui uma Unidade Lógica Aritmética, onde todas as operações lógicas e matemáticas são eletronicamente executadas. Toda essa lógica é estruturada na forma de um programa e gravado dentro do componente, assim, sempre que o microcontrolador é ligado à fonte de alimentação o programa carregado na memória interna é executado.

Amplamente utilizado para controlar processos, entende-se como o controle de periféricos como LEDs, mostradores digitais, relés, sensores, etc..

As aplicações são inúmeras e variadas, muitas vezes são utilizados vários microcontroladores em conjunto, com diferentes configurações de *software*, agregando valor e viabilizando a criação de produtos com funcionalidades mais amplas, com maior eficiência, maior segurança operacional e mais adequado às necessidades da solução desejada.

Sistemas que operam com a utilização de microcontroladores estão presentes nas mais diversas áreas de desenvolvimento tecnológico e industrial, muito comum em processos de automação, sejam eles comerciais, prediais, comerciais, agrícola, fabricação dos mais variados tipos de produtos e tudo mais que possa ter uma rotina de execução de tarefas.

Os microcontroladores PIC apresentam uma estrutura interna (arquitetura) do tipo Harvard, enquanto a maior parte dos outros microcontroladores apresenta arquitetura do tipo *von Neumann*.


**Figura 10 - Microcontroladores PIC, fabricados por Microchip [10].**

Na arquitetura *von Neumann* há apenas um barramento interno, em geral de oito bits, por onde trafegam as instruções e dados de programas.

Já na arquitetura Harvard, mais recente do que a outra apresentada, surgiu da necessidade de aumentar a velocidade dos processadores sem que isso impactasse diretamente em um aumento de tamanho do componente ou da frequência do oscilador. Como solução, foram criadas duas áreas para alocação de memória. Uma das áreas é dedicada exclusivamente para armazenar o programa e a outra é exclusivamente para os dados. Conjuntamente com a ação de criar dois bancos de memória foram criados também barramentos de comunicação distintos de tal forma que, como nas memórias, um deles é dedicado exclusivamente ao tráfego de dados enquanto no outro trafega o endereçamento dos dados.

Existem dois blocos de memória na estrutura interna de um microcontrolador, a memória de programa e a memória de dados. A arquitetura Harvard permite que o acesso à memória de programa seja independente do acesso da memória de dados, logo, cada bloco tem seu próprio barramento e o acesso a cada bloco pode ocorrer simultaneamente durante o mesmo ciclo do oscilador (*clock*).

# 2. DESENVOLVIMENTO DO PROJETO

## 2.1. Introdução

O adaptador desenvolvido neste trabalho possui o objetivo de estabelecer conexão de dispositivos de armazenamento massivo USB através da controladora de disquetes, substituindo o drive de disquete. Embora, já existam equipamentos com a mesma funcionalidade disponíveis no mercado, o diferencial proposto é de criar um adaptador que seja capaz de acessar qualquer dispositivo de memória através da USB e transformar automaticamente os arquivos com a extensão de texto em formato que a controladora possa realizar a leitura como se estivesse acessando o drive de disquetes.

A instalação deste adaptador no robô IRB-1400 deve permitir a conexão de dispositivos de memória, comumente conhecidos como *pendrives*, ou ainda outros dispositivos de memória que permitam acesso aos arquivos previamente formatados. Esses arquivos são reconhecidos por uma interface que realizará uma conversão automática para o padrão reconhecido pela controladora da placa mãe do robô.

A rotina de reconhecimento dos dados de interesse é programada no adaptador, onde são definidos os parâmetros de busca estabelecidos no projeto.

Para viabilizar a leitura dos arquivos é necessário que eles estejam alocados dentro de pastas com nome específico e tamanho limitado. O nome escolhido para as pastas foi “disk”. Juntamente com o nome cada pasta tem um número de indexação, limitado pelo número máximo que pode ser gerenciado pelo adaptador. Neste projeto foram definidos dois bits de endereçamento, o que significa que é possível trabalhar com quatro pastas de arquivos dentro de um mesmo dispositivo de memória, listadas com os nomes “disk0”, “disk1”, “disk2” e “disk3”. A interface de leitura do dispositivo de memória, conectado ao adaptador, lista apenas as pastas que têm tamanho menor do que o tamanho máximo de um disquete. Então, a soma dos tamanhos dos arquivos contidos em uma pasta dever ser limitada a 1.44 MB para

que esta pasta possa ser disponibilizada à controladora de disquete pela interface do adaptador.

Uma vez reconhecida e escolhida a pasta de trabalho, a interface carrega o conteúdo do dispositivo de memória para uma memória RAM, interna ao adaptador. A partir daí os arquivos de trabalho passam a ser acessados diretamente da memória interna do adaptador, mas sem que o dispositivo de memória portátil tenha sido removido da USB. Os arquivos de trabalho podem ser alterados através do painel do robô, o que altera o conteúdo da RAM do adaptador, mas ainda não altera o conteúdo do dispositivo de memória portátil. No momento em que o comando de ejeção é acionado, tal qual fosse um disquete comum, o conteúdo da RAM é transferido para o dispositivo de memória portátil conectado, dentro da pasta selecionada através do endereçamento, e depois que os dados estão completamente transferidos o adaptador sinaliza indicando que o dispositivo conectado à USB pode ser removido.

A operação de criação de arquivos através do painel do robô também é permitida, nesse caso o arquivo é salvo na RAM interna do adaptador e depois seguem os mesmos passos de transferência para o dispositivo de memória portátil.

### 2.2. Metodologia

Por se tratar da substituição de um dispositivo de tecnologia ultrapassada, muitas informações adquiridas estão disponíveis através da internet. A extensão de informações é grande e muitas vezes algumas dessas informações são contraditórias.

A pesquisa a respeito dos sinais de controle utilizados pela controladora de disquete foi suficiente para entender quais eram os sinais importantes, como estes sinais eram interpretados, como se comportavam e quais eram suas características.

Depois da pesquisa foi necessário realizar o mapeamento dos sinais de controle e de dados, de forma que fosse possível planejar e criar uma simulação desses sinais, aplicando-os a uma controladora de disquetes e verificando se seriam corretamente interpretados.

Para realizar essa etapa do projeto foi escolhido um computador com características semelhantes à característica da placa mãe do robô. Para ser um pouco mais exato, não se trata de uma semelhança e sim de características técnicas iguais ou equivalentes entre as controladoras de disquete, assim foi criada a possiblidade de realizar experimentos que teoricamente seriam equivalentes àqueles que seriam realizados na placa do robô, mas sem o risco de danificar um equipamento de grande valor para a instituição de ensino.

A placa mãe escolhida foi uma placa mãe da marca Asus®, modelo P4S533, montada pela Itautec, com processador Pentium® 4, memória RAM de 512 MB e sistemas operacional Windows® XP, com controladora de disquetes embarcada no componente IT8711F [20], no mesmo padrão de funcionamento daquela contida na placa mãe do robô.

Utilizando um osciloscópio digital de quatro canais, foram analisados os sinais de sinalização e os sinais de dados para que fosse verificada a ordem em que eles ocorriam, quais eram os níveis de tensão, os intervalos de tempo e possivelmente o período de cada um.

As conclusões depois da análise de sinais foram extremamente importantes porque foi possível dirimir várias dúvidas que surgiram durante a etapa de pesquisa, onde havia informações conflitantes a respeito da sequência e mesmo das circunstâncias em que alguns sinais ocorriam. Essa análise foi realizada enquanto estava instalado o drive de disquete genérico, *floppy*. As situações em que os sinais foram verificados precisavam ser aquelas nas quais a controladora recebe ou envia alguma sinalização do drive, sendo assim, foi estabelecido que as situações de transição ocorriam nos seguintes momentos: da inserção do disco, da leitura, da escrita (gravação) e ejeção. Essas situações foram simuladas por inúmeras

vezes até que fosse possível estabelecer como ocorriam os sinais, suas sequências, níveis e períodos.

Com todos os testes realizados e com a confirmação de como o drive respondia aos comandos da controladora de disquete foi possível então seguir para o passo seguinte, substituir o drive por um sistema que obedecesse aos comandos da controladora e respondesse da mesma forma, considerando as melhores características técnicas que um drive de disquete padrão poderia fornecer.

Nesse ponto do projeto foi elaborada a maneira de tratamento, considerando os sinais de comando e de dados. Já que são dois tipos de sinais, era necessário criar dentro do adaptador pelo menos dois blocos, um para tratamento dos dados e outro para o controle. O bloco de controle recebe todos os sinais exceto aqueles de dados modulados, sejam da leitura ou da escrita, e é responsável pelo controle de fluxo, pelo reconhecimento de taxas de leitura, presença de disco, dentre outros. O bloco de dados recebe os sinais de dados já modulados diretamente da controladora e alguns sinais gerados pelo bloco de controle.

No bloco de dados foi necessário criar um demodulador e um modulador MFM, isso porque a controladora transmite e recebe dados modulados em MFM. O drive de disquete não trata esses dados, ou seja, a unidade de disco não corrige erro, não modula, não verifica paridade ou executa qualquer outro processo que possa alterar o conteúdo da comunicação. A unidade de disco apenas grava o que recebe pronto da controladora e só transmite à controladora aquilo que está gravado nas trilhas do disquete. Todo tratamento do pacote de dados é executado pela controladora. Isso gerou a necessidade de criar dentro do adaptador um bloco capaz de abrir o envelope de dados modulados (MFM), interpretá-los e verificar se existe erro, quando a controladora executa a função de gravação no disquete, e um bloco capaz de modular dados de acordo com o que a controladora é capaz de interpretar quando executa a função de leitura.

Tratando especificamente do processo de leitura de um arquivo através do adaptador objeto deste trabalho, quando a controladora sinaliza ao adaptador que ele deve transmitir os dados, a memória RAM do adaptador é acessada, ela contém os dados organizados da mesma forma que estariam organizados nas trilhas de um disquete, então, o conteúdo da RAM é enviado pelo canal de dados à controladora, trilha por trilha.

Quando a controladora executa a função de escrita, há a necessidade de realizar dentro do adaptador a gravação na RAM do sinal MFM recebido. A cada trilha recebida é verificada a existência de erros e se houver existe a necessidade de sinalizar para a controladora que há erro e a transmissão de dados deve ser executada novamente. Caso não sejam encontrados erros, os dados são interpretados por outra interface, que formata esses dados como se eles fossem parte de um de disquete, todos eles organizados em lados, trilhas e setores.

Nesse ponto tem-se a função básica do adaptador, que é interpretar os dados da controladora e gerar dados que sejam interpretados pela mesma controladora, ou seja, um canal de comunicação capaz de simular a presença de um drive de disquete, tornando possível a manutenção do uso de um equipamento robusto como a antigo robô da ABB, sem que ele perca nenhuma de suas funcionalidades.

Essa etapa do projeto é a mais significativa porque é aqui onde existe a efetiva conversão de tecnologia, qualquer outra etapa seguinte ficaria sem propósito algum sem essa funcionalidade. Isso permite a entrada e saída de dados pela antiga controladora de disquetes, que apesar das limitações de velocidade, ainda assim pode ser necessária em muitos casos.

O próximo passo, que não foi concluído neste projeto, é a interface que trabalha com os dados contidos na RAM, dados esses que simulam o conteúdo de um disquete com capacidade de 1.44 MB, com a função de importar ou exportar dados para outra parte do sistema do adaptador. Essa interface foi planejada para tornar possível a conversão de arquivos em *ASCII*, de maneira que possa ocorrer a exportação de dados para um dispositivo

de armazenamento conectado à USB do adaptador, ou a importação dos dados desse dispositivo para a memória RAM do adaptador.

A interface de conexão da memória RAM com a porta USB é responsável pelo reconhecimento das pastas previamente formatadas com tamanho máximo de 1.44 MB. Depois de reconhecidas, elas podem ser acessadas de acordo com o endereçamento selecionado. Este endereçamento é realizado através de parâmetro programável do adaptador e inicialmente foi limitado a apenas quatro pastas, mas essa quantidade pode ser alterada através de nova programação. O sistema operacional do robô pode ser carregado em três disquetes, o que ainda permitiria o uso de outra pasta além do sistema operacional em um mesmo dispositivo de memória portátil.

Além do reconhecimento das pastas disponíveis para leitura essa interface tem a função de converter o conteúdo da pasta escolhida e transcrever na memória RAM do adaptador de forma que possa ser interpretado pela interface da controladora, ou seja, copiar os dados da pasta escolhida para a RAM, com a forma de endereçamento (lado, trilha e setor) do disquete. Nessa etapa há a conversão de formatação de dados. Então, de maneira inversa, esta interface também exporta os dados gravados na memória RAM do adaptador para uma pasta de destino, escolhida pelo usuário, dentro do dispositivo de memória portátil conectado à USB do adaptador.

O grande volume de trabalho necessário à implementação desta etapa não tornou possível a sua realização, mas como o adaptador foi elaborado de forma modular, não há nenhum impedimento para que o projeto continue em desenvolvimento, inclusive com outras possibilidades de acesso através da USB, sendo possível até mesmo a utilização de tecnologias sem fio [1].

### 2.3. Definição do *hardware*

Logo no início do trabalho havia a preocupação a respeito da tecnologia de microcontroladores que poderia ser utilizada para implementar o adaptador. Logo depois das primeiras pesquisas foram realizados alguns testes, talvez até prematuros, mas que foram de grande relevância para o desenvolvimento geral do projeto.

Os primeiros testes foram realizados com uma placa de desenvolvimento do PIC18F4550, Figura 11, com a qual havia a expectativa de elaborar toda a operação. As características de desempenho e de acesso de dados pareciam adequadas às necessidades do projeto. Na fase inicial do trabalho já se percebia que as limitações desse microcontrolador tornariam o desenvolvimento mais complicado do que se esperava. A quantidade de portas disponíveis era suficiente para realizar o mesmo controle que uma unidade de disquete realiza. Apesar disso, a primeira conclusão é que ele não teria velocidade suficiente para gerenciar o controle e a transferência de dados, então, seriam necessários pelo menos dois deles, programados em *assembly* para melhorar o desempenho, isso porque o melhor compilador disponível não teve capacidade de converter o programa desenvolvido na linguagem C de forma otimizada, então a programação teve que ser feita diretamente em *assembly*. Outra dificuldade foi a comunicação com módulos de memória, com grande instabilidade e capacidade insuficiente aos propósitos do projeto inicial. Ainda depois dessas complicações foi constatado que a USB disponível na placa de desenvolvimento servia apenas para comunicar com o microcontrolador, e mesmo que se utilizasse uma placa com desenvolvimento próprio a USB do controlador não poderia ser usada para acessar os arquivos do dispositivo de armazenamento portátil.

Figura 11 - Placa de desenvolvimento utilizando microcontrolador PIC18F4550.

Na busca de alternativas, foram realizados testes com um Arduino Mega 2560, Figura 12, com a placa de acesso a cartão de memória acoplado. Com essa placa de desenvolvimento foi um pouco mais simples de acessar o conteúdo da memória do cartão SD, mas o objetivo do projeto era criar uma interface que pudesse ser desenvolvida, expandida, então, apesar de existir forma de acesso sem fio pela interface do cartão SD, talvez no futuro essa solução também pudesse trazer alguma outra limitação e com isso a necessidade de substituição do adaptador criado. Outras características observadas no outro microcontrolador já testado foram também analisadas nessa placa de desenvolvimento, e apesar de se mostrar um pouco mais rápido, ainda assim não seria suficiente utilizar apenas um desses. Até porque a quantidade de portas oferecidas nessa placa de desenvolvimento não era suficiente para tratamento de todos os sinais de controle necessários. Assim, a utilização deste equipamento foi descartada também.

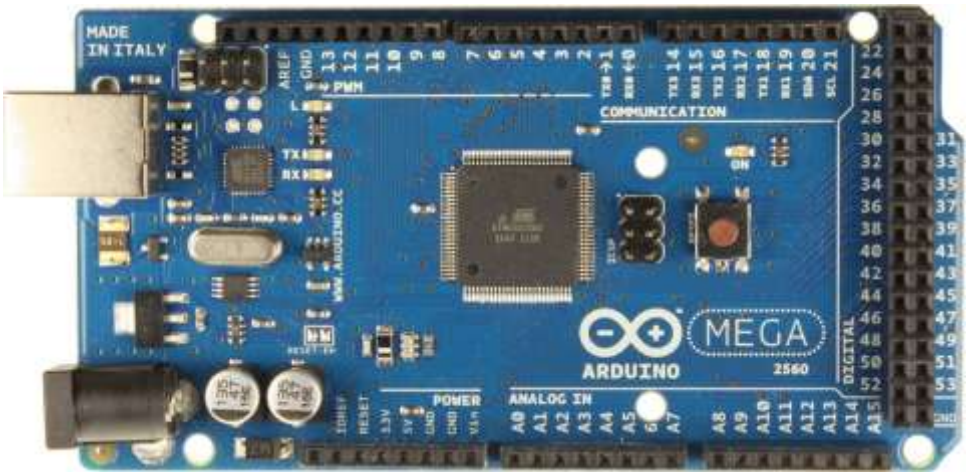


**Figura 12 - Arduino Mega 2560, com microcontorlador ATMEGA 2560.**

Outra tentativa, um pouco mais ousada, foi utilizar um Raspberry Pi 2, Figura 13, uma placa de desenvolvimento poderosa, com características técnicas superiores às outras já testadas, processador de 4 núcleos rodando a 700 MHz, mas a escrita do código também não poderia ser em linguagem C por questões de desempenho, mesmo utilizando instruções ditas mais rápidas. Assim se fazia necessário o aprendizado de outra linguagem de programação, Pyhton, que é a linguagem nativa dessa plataforma de desenvolvimento. Mesmo assim foram testadas algumas funções básicas. A plataforma não respondeu muito bem, porque como a programação é de alto nível, a resposta nas saídas não acontecia no tempo desejado, ou seja, a plataforma embora superior às outras, não tinha velocidade suficiente para trabalhar com sinais da ordem de centésimos de microssegundos. Além disso, seria uma solução cara, trabalhando muitos recursos inutilizados. Apesar disso ainda haviam algumas portas listadas como disponíveis no manual, mas que de fato não podiam ser usadas para as funções específicas do projeto do adaptador.

Figura 13 - Raspberry Pi 2, com processador ARM1176JZF-S.

Finalmente se chegou numa plataforma de desenvolvimento compatível com o custo adequado, com funcionalidades e capacidade de processamento desejado. Para isso foi necessária a utilização de uma placa construída com microcontrolador da família PIC32. Esse processador possui uma das características desejadas ao projeto, diria até mesmo que é a característica essencial para que o projeto exista, USB-OTG (*universal serial bus on the go*), que permite a comunicação do microcontrolador com dispositivos USB para transferência de dados, tornando viável todo o desenvolvimento da interface entre o adaptador e o dispositivo de armazenamento portátil.

Assim, a plataforma de desenvolvimento escolhida foi a placa da Microchip construída com o microcontrolador PIC32MX470F512L, modelo DM-320003-3, apresentada na Figura 15, com frequência de operação de 96 MHz, cinco *pipelines*, 1.56 DMIPS/MHz e memória de 512 kB *Flash*. A arquitetura interna do microcontrolador é ilustrada na Figura 14.

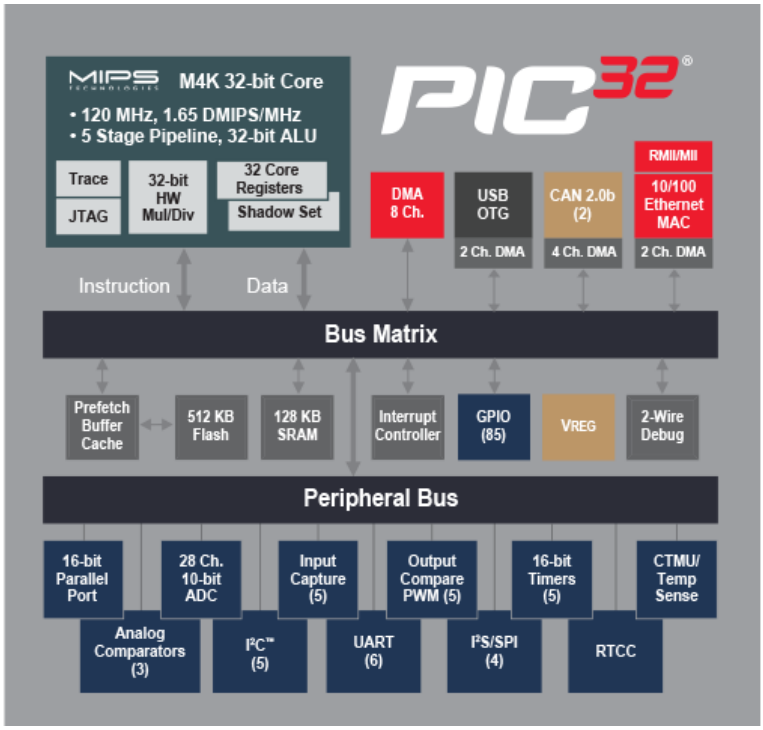


Figura 14 - Diagrama de arquitetura interna dos PIC32 [17].

As características completas do microcontrolador estão disponíveis em documento de especificações técnicas, produzido pela empresa Microchip Technology Inc. e acessível na página em http://www.microchip.com/.


Figura 15 - Placa de desenvolvimento escolhida para o trabalho, modelo DM320003-3 [18]. Microcontrolador PIC32MX470F512L.

Juntamente com kit de desenvolvimento escolhido para o projeto foi necessária a aquisição de uma placa de expansão, de forma que as portas de comunicação do microcontrolador pudessem ser acessadas fisicamente, permitindo a conexão fácil de fios, cabos e ferramentas de medida. A Figura 16 ilustra a placa de expansão.

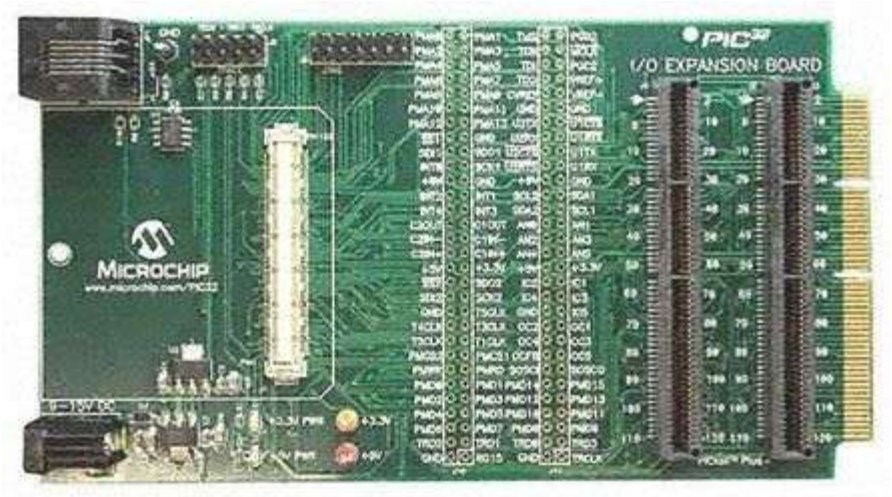


**Figura 16 - Placa de expansão modelo DM320002 [12].**

O próximo passo então era verificar a compatibilidade do sistema com o qual se estava trabalhando, saber se seria possível realiza a conexão do adaptador desenvolvido sem causar nenhum prejuízo à parte elétrica do robô. Como as condições do protótipo ainda não estavam adequadas para transporte, foi testado um equipamento similar, com características construtivas aproximadas. O primeiro teste realizado foi substituir a unidade de disquete do computador de trabalho por um emulador de disquete comercial e verificar se as características elétricas eram compatíveis. Com o resultado positivo deste teste foi testada a conexão do mesmo emulador com a placa mãe do robô da ABB, onde foi constatada a capacidade de realizar a escrita e a leitura de arquivos dentro de um dispositivo de memória portátil sem que houvesse nenhum tipo de incompatibilidade.

### 2.4. Definição do ambiente de desenvolvimento

O ambiente de trabalho está diretamente ligado ao dispositivo de *hardware* escolhido. A empresa Microchip, produtora dos microcontroladores PIC, também é responsável pelo

desenvolvimento dos aplicativos MPLAB® X IDE e MPLAB® XC, disponibilizando-os através da rede mundial de computadores em versões de avaliação com todas as funcionalidades e possibilidades de compilação, por um período de dois meses. Transcorrido o período, o aplicativo de desenvolvimento da programação de PIC restringe sua capacidade de desenvolvimento e passa a disponibilizar apenas um quarto de seus recursos. Para um desenvolvimento de aplicações em baixo nível, onde se faz necessário operar com velocidade de processamento, essa limitação na compilação da programação, ocasiona uma série de transtornos porque nem sempre as funções escritas em linguagem C através da IDE (*Integrated Development Environment*) são convertidas da maneira correta para linguagem de montagem (*assembly*). A Figura 17 apresenta o ambiente de desenvolvimento utilizado no projeto.

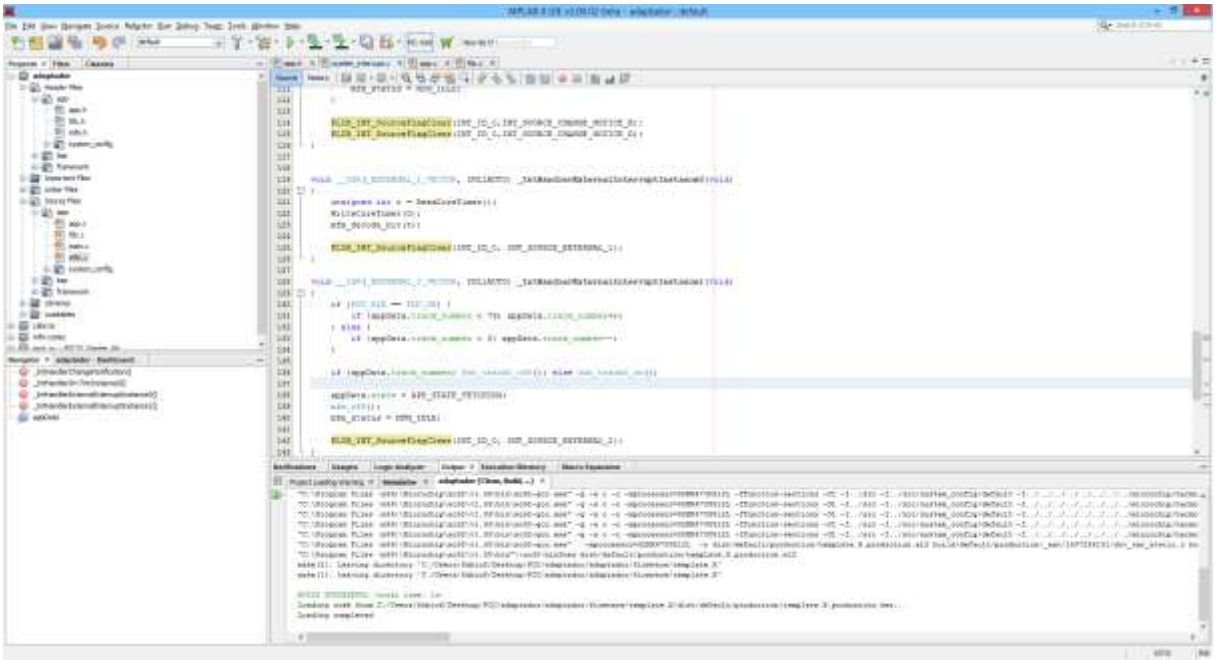

Figura 17 - Ambiente programação da placa de desenvolvimento

## 2.5. Adaptador USB

Como exposto anteriormente, o adaptador desenvolvido neste trabalho não será um mero substituto da unidade disquete, ele servirá como um dispositivo de utilização da

controladora de disquetes, provendo a possibilidade de leitura de dispositivos de memória assim como expansão para outros tipos de tecnologia baseados em conexão USB.

Durante a pesquisa para elaboração do projeto, a primeira ideia era utilizar tecnologia muito parecida com a tecnologia encontrada nos dispositivos da época da controladora. Para esse propósito, a utilização de um componente eletrônico dedicado ao controle da porta seria ideal para o equipamento, porque trabalha com os mesmos sinais, níveis e temporizadores, enfim, todos os padrões da FDC instalada no robô da ABB.


**Figura 18 – Componente dedicado para controladora de disquetes fabricada pela NEC na década de 80, modelo µPD765, com capacidade para controlar até quatro unidades de disquete de dupla densidade (MFM) [14].**

Entretanto, como a principal ideia no meio acadêmico é buscar o desenvolvimento continuamente, a melhor forma de construir o adaptador seria aplicar componentes novos, deixando como alternativa a possibilidade de expansão e assim, fazendo com que o projeto não fosse um equipamento obsoleto como a controladora. O uso de um componente eletrônico dedicado que está fora de linha poderia trazer os problemas de manutenção tão comuns a equipamentos antigos.

Com essa decisão tomada, o projeto foi desenhado para que tivesse um formato modular, escrevendo em software os componentes necessários para realizar a comunicação

com a controladora de disquetes da placa do robô. A estrutura básica está ilustrada na Figura 19.

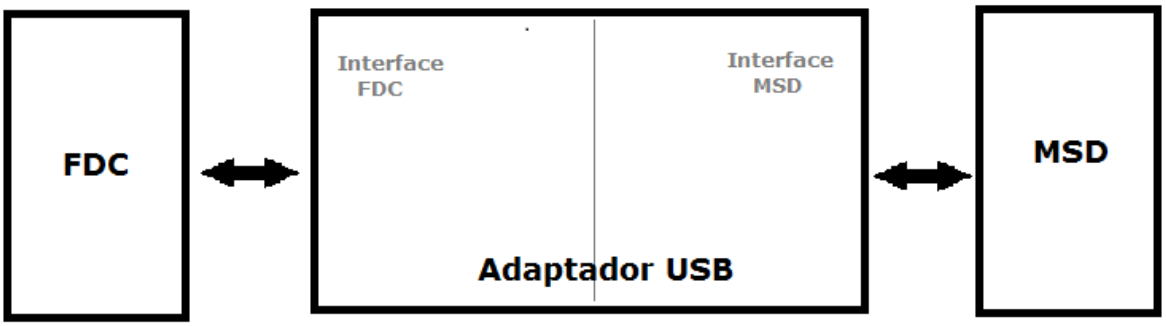


Figura 19 - Configuração básica do adaptador USB.

No adaptador USB há uma divisão, correspondente a programação específica para cada função. Essa configuração de construção permite que sejam incluídas novas funcionalidades sem que algum tipo de alteração física do equipamento.

Por exemplo, conforme esquema da Figura 20, é possível viabilizar a utilização de tecnologias sem fio para estabelecer comunicação com a controladora de disquetes do robô, todas elas através da mesma porta USB.

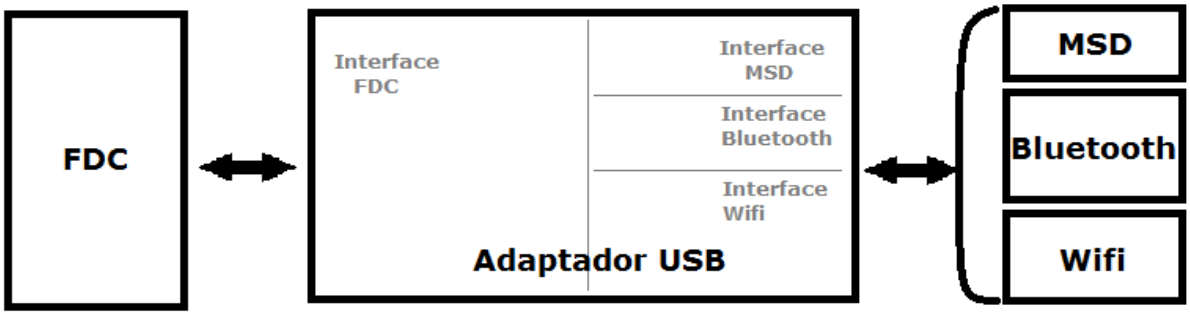


Figura 20 – Configuração do adaptador USB com expansão para tecnologias sem fio.

Dentre as configurações possíveis, foi escolhida a configuração que conta com o auxílio de uma memória RAM, com capacidade para armazenar o conteúdo inteiro de um disquete. O esquema da Figura 21 ilustra como o adaptador trabalharia idealmente.

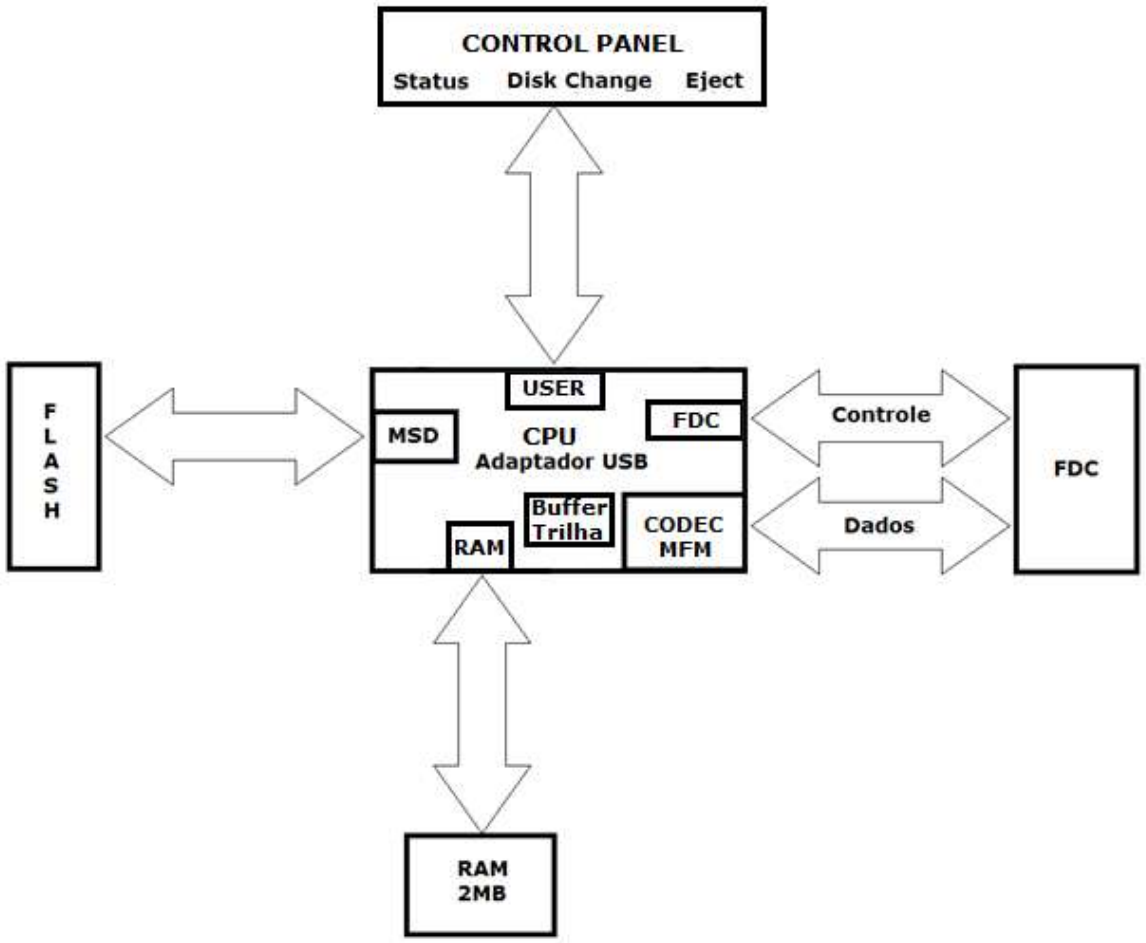


Figura 21 - Adaptador conectado ao dispositivo de memória portátil, RAM e FDC.

As sinalizações do adaptador para o usuário são: *Status*, *Disk Change* e *Eject*. A função destas sinalizações é fornecer ao usuário informações básicas a respeito da operação da unidade, para permitir que os procedimentos sejam seguidos de forma adequada, dessa maneira é possível evitar que ocorra algum tipo de dano, principalmente à integridade dos dados que estão sendo trabalhados.

O adaptador realiza a leitura da pasta seleciona na porta USB, identifica os arquivos e carrega a memória RAM com o conteúdo da pasta. Na memória RAM os dados são mapeados como num disquete. Com essa capacidade de memória é possível alocar o equivalente a 80 trilhas de disquete, ou seja, um disquete de 1.44 MB.

Obviamente que este codificador mantém todos os padrões que um modulador dedicado apresenta como características. Uma unidade de disquete tem uma série de

constantes de tempo devido a utilização de peças mecânicas. Quando a cabeça se move um *step*, ou seja, de uma trilha para outra, o tempo típico de uma unidade de disquete de 3½ polegadas é de no máximo 3 ms. Esse tempo tem influência direta no ruído sonoro produzido pelos motores. Um tempo muito curto ou muito longo causa ruído e pode desgastar as partes mecânicas, mas isso não importa aqui, já que o adaptador não tem partes móveis. Depois que a cabeça é movida de uma trilha para outra é necessário aguardar aproximadamente 15 ms (quinze milissegundos) enquanto a cabeça estabiliza e para de vibrar. Também tem o tempo que o motor necessita para atingir a velocidade nominal depois da partida, tipicamente 1 s (um segundo), mas em algumas unidades de disquete esse tempo é reduzido pela metade, já que a correção pode ser feita pela controladora, solicitando uma nova tentativa de leitura. Existe ainda o tempo que a cabeça necessita para se mover na superfície do disquete, mas não é importante porque é de cerca de 4 ms, muito menor que o tempo de partida do motor.

### 2.5.1. O CODEC – codificador/decodificador MFM

Dentro do CODEC estão implementadas duas etapas fundamentais ao bom funcionamento deste projeto, um codificador e um decodificador MFM.

O codificador desenvolvido em *software* é capaz de realizar a conversão de dados com tamanho de oito bits em uma sequência de sinais conhecida como modulação MFM. A codificação de cada um dos bits segue a seguinte regra. O bit “1” é sempre convertido como um espaço seguido de um pulso. O bit “0” tem duas formas de conversão. Quando um “0” aparece logo depois de um “1” ele é convertido como dois espaços e quando ele aparece logo de outro “0” ele é convertido como um pulso seguido de um espaço.

A duração da codificação de um bit inteiro é de 2 µs. Um espaço corresponde a um tempo de 1 µs, também como o espaçamento do pulso inteiro, mas o pulso ainda tem uma característica própria. Cada pulso começa com nível alto e permanece nesse estado por 0,4 µs e depois cai para nível baixo durante 0,6 µs.

Assim, quando se codifica um “1”, temos um sinal elétrico que começa em nível baixo, permanece nesse estado por 1 µs, depois altera o nível para alto e permanece por 0,4 µs e volta para nível baixo por mais durante 0,6 µs, finalizando o tempo do bit de 2 µs. Ilustração apresentada na Figura 22.

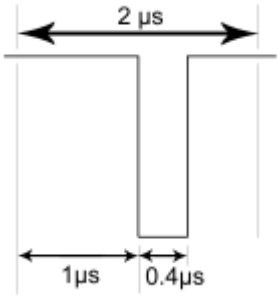


**Figura 22 - Bit "1" em MFM (TTL tem nível lógico inverso).**

Um “0” depois de um “1” é simplesmente um sinal de nível baixo durante o tempo total do bit, ou seja, durante 2 µs (dois microssegundos). Ilustração apresentada na Figura 23.

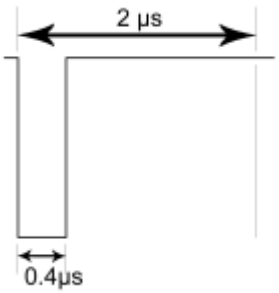


**Figura 23 - Bit "0" depois de "1" em MFM (TTL tem nível lógico inverso).**

Um “0” depois de outro “0” é um sinal que inicia em nível alto e permanece durante 0,4 µs (quatrocentos nanossegundos), depois retorna para nível baixo por 1,6 µs (um microssegundo e seiscentos nanossegundos), totalizando 2 µs (dois microssegundos). Ilustração na Figura 24.

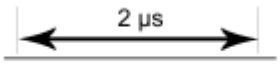


Figura 24 - Bit "0" depois de "0" em MFM (TTL tem nível lógico inverso).

A codificação MFM é realizada a cada conjunto de 8 bits contidos em arquivos proveniente do dispositivo de memória portátil.

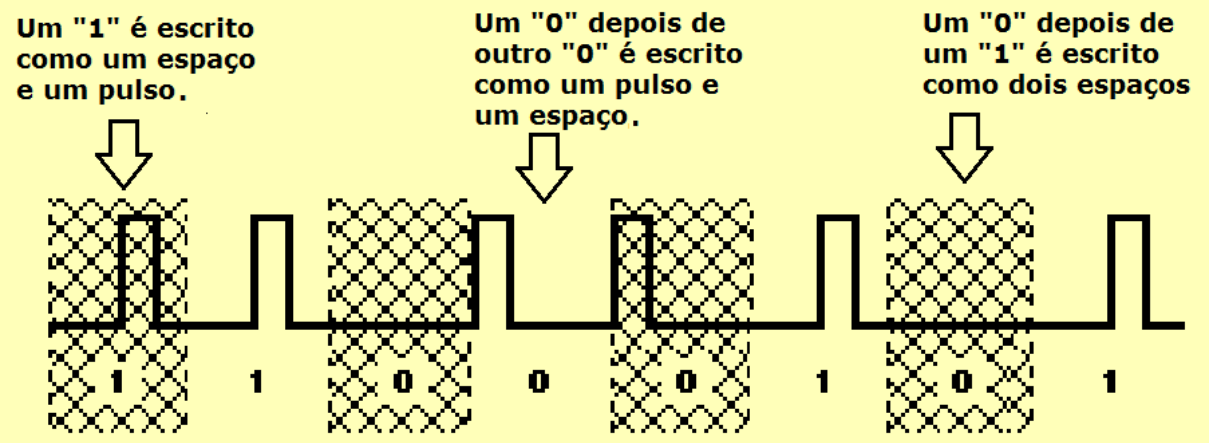


Figura 25 - A palavra digital 11000101 convertida em sinal MFM.

Os sinais enviados pela controladora de disquete e recebidos pelo codificador são: os dados paralelos de oito bits das palavras binárias (D0-D7) e o comando para realizar a modulação. Os sinais que o codificador envia para a controladora de disquete são: dados serializados modulados em MFM e a sinalização de INDEX.

Para realizar a leitura dos sinais que são transmitidos pela controladora à unidade de disquete foi criado um decodificador. Os tempos associados ao mecanismo da unidade de disquete também são respeitados nesta etapa. Com função contrária ao codificador, essa etapa do aplicativo recebe os sinais modulados em MFM e interpreta, paralelizando novamente os dados em uma palavra digital de oito bits. Os sinais recebidos pelo decodificador são: sinal MFM e comando de leitura. O sinal gerado é uma série de dados paralelos, de D0 a D7 (8 bits).

O CODEC MFM também estabelece comunicação com o *buffer* de trilha e com este módulo realiza a gravação ou leitura da trilha de trabalho, dependendo da requisição da controladora.

### 2.5.1. O Buffer de Trilha

Como já mencionado no item anterior, o *buffer* de trilha realiza comunicação com o módulo CODEC MFM, com o objetivo de realizar a leitura ou a gravação de uma trilha do disquete.

Este módulo opera a memória interna da CPU, sendo responsável por manter disponíveis os dados da trilha que a controladora requisita, isto se deve ao fato de que a controladora opera com apenas uma trilha por vez, com tamanho de 12500 bytes. Depois que os dados lidos na interface USB são transferidos para a memória RAM, a trilha “0” é automaticamente carregada para o *buffer* de trilha, e fica disponível à controladora.

### 2.5.2. A Interface FDC

A interface FDC é responsável pela comunicação de todos os dados de controle entre a CPU do adaptador e a controladora de disquetes. Cada uma das sinalizações recebidas ou enviadas por uma unidade de disquetes comum é tratada por este módulo, exceto os sinais de seleção de densidade, ativação do motor A e proteção de escrita. A proteção de escrita pode ser ativada via software para proteger o dispositivo de memória portátil e os outros sinais são irrelevantes.

### 2.5.3. A Interface USER

A interface USER serve para realizar a interação com o painel de acesso do usuário. Ele sinaliza quando funções da controladora estão prontas ou ainda permitem que o usuário acione comandos. Através dessa interface é escolhida a pasta de trabalho, que corresponde ao disquete, também pode ser acionado o comando de ejeção que tem a função de salvar os

dados nas pastas. A sinalização “Status” indica com uma luz verde quando o adaptador está operando ou quando o dispositivo de memória portátil pode ser removido sem que haja perda dedados. A sinalização “Disk Change” informa ao usuário que a memória do dispositivo de memória não pode gravar os dados contidos no adaptador, e é necessário substituí-lo. O botão “Eject” permite que o usuário informe ao adaptador que o dispositivo de memória portátil será removido e assim, os dados contidos no adaptador são transferidos ao dispositivo de memória antes que ele seja desconectado. Este módulo não foi implementado, já que até o momento não foi possível realizar interação com o usuário.

### 2.5.4. A Interface MSD

Esta interface tem a função de importar e exportar para a RAM do adaptador todos os dados da pasta selecionada do dispositivo de memória portátil. Cada vez que um dispositivo for conectado à USB do adaptador esta interface identifica a pasta de trabalho é carrega o conteúdo da pasta na memória RAM e também, quando for acionado o comando de ejeção, o conteúdo da memória RAM é escrito no dispositivo de memória portátil.

### 2.5.5. Interação da CPU com o módulo de memória RAM

A pasta de arquivos selecionada pelo usuário e contida no dispositivo de memória portátil conectado à porta USB tem seu conteúdo completo transferido ao módulo de RAM. Depois da transferência os dados de interesse passam a ser acessados somente através do módulo de RAM. Para esta operação a CPU do adaptador coordena as ações entre a interface MSD e o módulo de memória.

Para especificar a quantidade necessária de memória basta lembrar que o disquete comum tem capacidade de armazenar 2 MB em sua totalidade. Para a tecnologia escolhida, a capacidade máxima disponível em um só componente é de 128 KB. Então, para construir o módulo de memória RAM são necessários 16 componentes agrupados adequadamente para fornecer a capacidade de memória requisitada pelo projeto.

A CPU do adaptador organiza os dados da pasta lida na porta USB dentro da memória RAM obedecendo a limitação de capacidade de armazenamento de uma trilha, que é de 12500 bytes. Essa organização segue ilustrada na Figura 26.

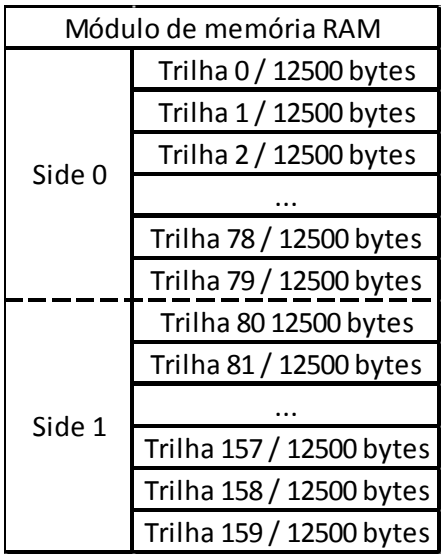


**Figura 26 - Organização dos dados dentro do módulo de memória RAM.**

Além de carregar os dados escolhidos pelo usuário para o módulo de memória RAM, a CPU do adaptador também carrega em sua memória interna, no buffer de trilha, inicialmente a trilha zero e posteriormente a trilha requisitada pela FDC. A Figura 27 ilustra como fica o buffer de trilha depois dessa operação.

| Buffer de trilha | | | | | | |
|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte 3 | ... | Byte 12497 | Byte 12498 | Byte 12499 |
| 0xXX | 0xXX | 0xXX | ... | 0xXX | 0xXX | 0xXX |

**Figura 27 - Distribuição dos bytes dentro do buffer de trilha.**

Quando a CPU do adaptador inicia a comunicação com a controladora de disquetes, as requisições de trilha começam a ser apresentadas para realizar a montagem do arquivo selecionado, para isso há um processo de transferência do conteúdo da trilha armazenada na RAM para o buffer de trilha, que está na memória interna da CPU.

Para cada trilha solicitada pela controladora, a CPU envia o conteúdo do buffer de trilha, descarrega a trilha enviada, recarrega a próxima trilha solicitada pela FDC e repete este

ciclo até que todas as trilhas solicitadas tenham sido enviadas de forma correta à controladora. Para esta operação foi criado um ponteiro que indica o endereço da memória RAM onde a trilha de interesse está armazenada, representada na Figura 28.

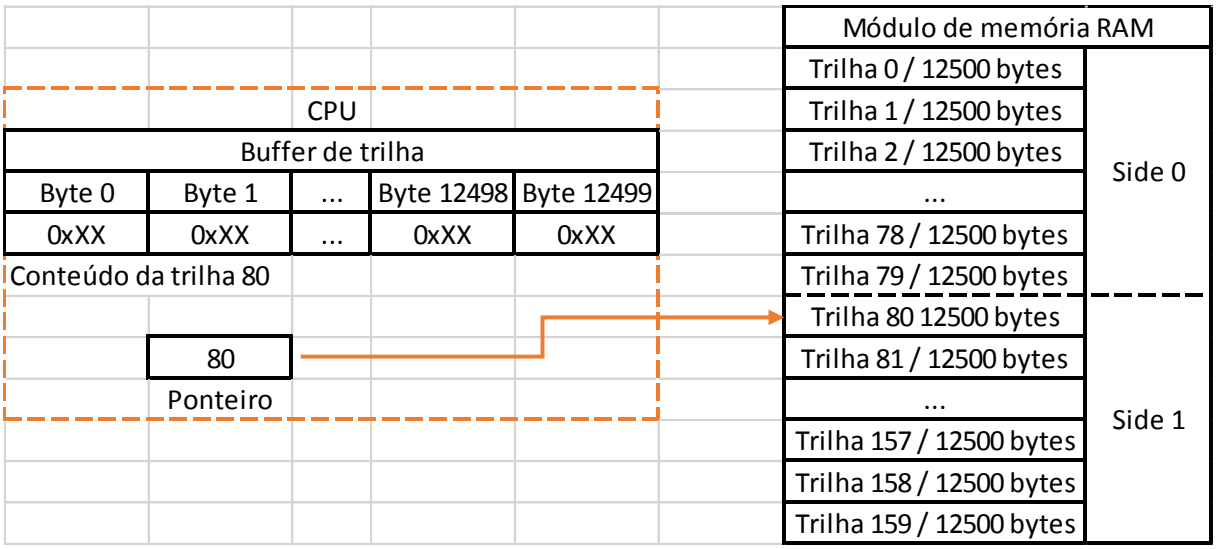


**Figura 28 - Carregamento do buffer de trilha com conteúdo da RAM.**

O conteúdo do buffer de trilha é alterado pelo CODEC MFM antes de ser enviado à controladora.

### 2.5.6. Interação da CPU com o CODEC MFM

Com os dados carregados no buffer de trilha o CODEC MFM tem a função de codificar os dados da trilha selecionada e enviar à controladora de forma inteligível. De forma inversa, quando a controladora envia uma trilha para o adaptador, acontece a decodificação do conteúdo MFM que é carregado no buffer de trilha. Depois disso a CPU faz com que aconteça a interação do buffer com o módulo de memória RAM.

### 2.5.7. Interação da CPU com a interface FDC

Através desta interface é que a controladora recebe as informações da CPU do adaptador e vice-versa. Os sinais utilizados para selecionar a trilha como, $\overline{SIDE}$, $\overline{DIR}$ e $\overline{STEP}$ são interpretados pela CPU para criar o ponteiro que indicará o conteúdo do módulo de RAM de interesse da controladora. Dependendo do tipo de operação o conteúdo da trilha contido na

RAM é carregado no buffer de trilha, ou, o conteúdo do buffer que foi recebido da FDC é carregado na RAM.

No caso em que a controladora sinaliza com $\overline{WDATA}$ a CPU do adaptador entende que deve receber a trilha da FDC, carregá-la no buffer de trilha, que enviará à RAM, e aguardar a sinalização para realizar a troca de trilha se for necessário.

No caso em que a controladora sinaliza com $\overline{RDATA}$ então a CPU do adaptador entende que deve enviar o conteúdo contido no buffer de trilha à FDC, aguarda a sinalização para realizar a troca de trilha. Se receber a solicitação de nova trilha, será necessário recarregar o buffer com o conteúdo solicitado e envia-lo à FDC.

### 2.5.8. Sequência de operação interna do adaptador

Aos usuários do sistema talvez a informação de como o adaptador opera internamente não seja relevante, mas é importante do ponto de vista do desenvolvimento. Para descrever as atividades gerais do adaptador será usado o diagrama de máquina de estado, sem o rigor normalmente requerido para tal representação, ilustrada na Figura 29.

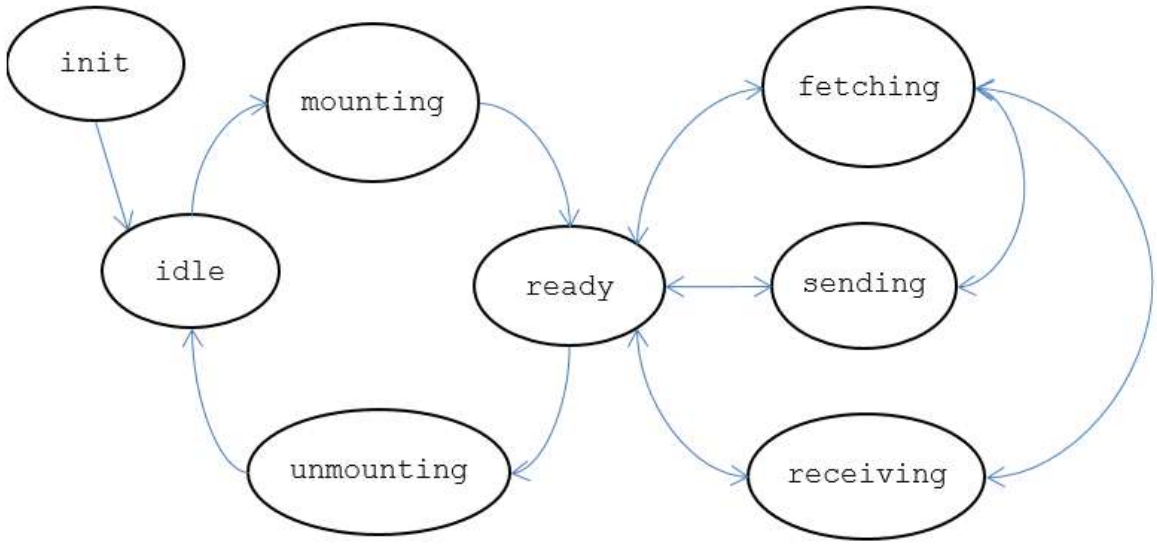


**Figura 29 - Máquina de estados - lógica do adaptador.**

O estado definido como *init* representa o momento em que o adaptador é ligado. Depois da verificação de que todos os módulos estão funcionando a CPU passa para o

próximo estado, *idle*, onde o adaptador aguarda a conexão de um dispositivo de memória portátil na porta USB. Depois que ocorre a conexão de um dispositivo adequado na porta USB o adaptador passa para o estado *mounting*, monta o volume e verifica se há alguma pasta disponível com o nome requerido para utilização do robô, se houver, o conteúdo da pasta é carregado para o módulo de memória RAM, e assume o estado *ready*, onde aguardará novos comandos.

Quando o adaptador está no estado *ready* pode receber o comando de ejeção através da interface USER e os comandos de leitura ou escrita da interface FDC. Quando recebe o comando de ejeção passa para o estado de *unmounting* e executa as funções de transferência de conteúdo entre o módulo de RAM e o dispositivo de memória portátil. Quando recebe comandos através da interface FDC o adaptador pode assumir três estados distintos. No estado de *fetching* troca informações com a controladora com o objetivo de localizar o conteúdo requisitado dentro da memória RAM e torná-lo disponível através do buffer de trilha. Dependendo obviamente dos estados das variáveis, esse estado tem três saídas possíveis, volta ao estado *ready*, vai para o estado *sending* ou passa para o estado *receiving*.

No estado de *sending* o adaptador envia dados para a controladora, mas como em geral um arquivo ocupa mais de uma trilha, o retorno para o estado de *fetching* pode ser necessário, ou quando concluído o envio retorna para a posição de *ready*.

No estado de *receiving* o adaptador recebe dados da controladora, mas como em geral um arquivo ocupa mais de uma trilha, o retorno para o estado de *fetching* pode ser necessário, ou quando a recepção é concluída retorna para o estado *ready*.

Tanto nos estados de *sending* e *receiving* pode não ser necessário realizar a busca da trilha novamente por dois motivos. Uma razão pela qual não é necessário retornar ao estado de *fetching* ocorre quando a controladora realiza comunicação informando que há erro e a

mesma trilha é reenviada ou recebida. Outa razão é quando o arquivo é suficientemente pequeno e não há alteração da trilha de trabalho.

É possível notar que a que a nomenclatura dos estados da máquina, no idioma inglês, possui verbos no gerúndio, verbo no infinitivo e outras são adjetivos. Nos estados estacionários, onde há necessidade de ação externa para que aconteça alguma alteração, são usados os adjetivos e o verbo no infinitivo. Nos estados em que o adaptador segue realizando ações que independem de comandos do usuário são utilizados os verbos no gerúndio.

Para esta representação é importante mencionar que no estado *idle*, com a configuração ideal apresentada, o adaptador espera a ação da interface MSD que está ligada à porta USB, entretanto essa interação poderia ser realizada através de tecnologias sem fio, bastando apenas desenvolver as interfaces Wifi e Bluetooth.

### 2.5.9. Etapas implementadas do protótipo do adaptador

#### 2.5.9.1. Delimitação do escopo

Devido ao volume de informações, a carência de equipamentos e componentes eletrônicos, foi possível implementar apenas uma parte do protótipo. Nesta etapa é capaz de realizar a comunicação com a controladora FDC, fazendo com que ela reconheça a presença de um disquete, tornando possível a escrita e leitura de arquivos com o protótipo adaptador conectado à placa mãe de um computador pessoal. Como existe compatibilidade do circuito eletrônico do computador e do robô, verificada em experimento que utilizou dispositivo similar ao protótipo, tudo aquilo que é possível fazer num PC também é possível no módulo de controle do robô.

A Figura 30 ilustra a estrutura montada do que está sendo descrita.

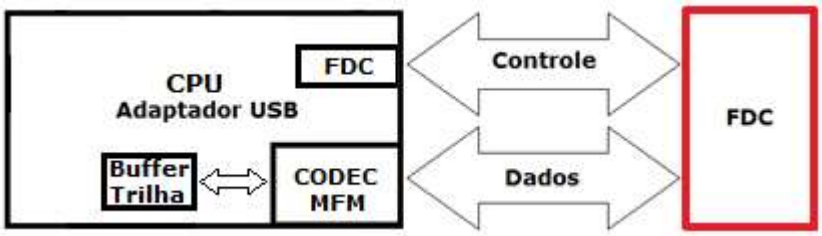


**Figura 30 - Etapas implementadas do protótipo do adaptador USB.**

Como nem todas as etapas do adaptador foram implementadas, foi criada uma estratégia para simular a operação e realizar os testes necessários a respeito do funcionamento lógico.

Na memória interna do microcontrolador foram armazenadas duas trilhas, como na organização do disquete. Uma das trilhas com as informações da FAT, para realizar o endereçamento dos dados, e outra trilha com o conteúdo de dados de interesse, um arquivo com extensão de texto, com tamanho inferior à capacidade da trilha.

Em relação ao diagrama da máquina de estados, onde foram representadas as etapas de comunicação do adaptador, o experimento inicia no estado *idle*, não realiza *mounting* nem *umounting* a passa ao estado *ready*, e depois do processo retorna neste mesmo estado, isso porque não é possível transferir o conteúdo do buffer de trilha para a memória RAM.

Foram realizados os testes de leitura e escrita e o resultado obtido foi satisfatório, comprovando a capacidade de comunicação do adaptador USB com uma controladora de disquetes. Na Figura 31 é apresentada a imagem do circuito montado de maneira rudimentar.

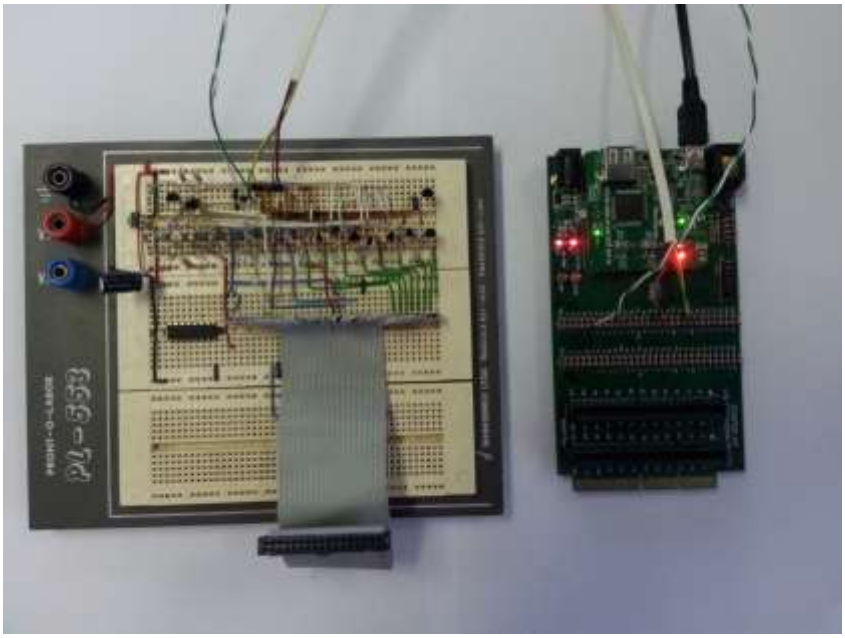

Figura 31 - Circuito montado Interface FDC e Buffer de Trilha operacionais.

O mapeamento dos pinos da placa de desenvolvimento em relação aos pinos da controladora FDC são apresentados na Tabela 12.

Tabela 12 - Mapa de pinos do protótipo.

| Pino Cabo | Sinais | Sentido ADT ⟶ FDC | Descrição | Porta PIC | Pino PIC | Pino EB |
|---|---|---|---|---|---|---|
| 2 | $\overline{DENSEL}$ | ← | Seleção Densidade | - | - | - |
| 8 | $\overline{INDEX}$ | → | Índice | D5 | 82 | 25J10.13 |
| 10 | $\overline{MOTEA}$ | ← | Ativar Motor Drive A | - | - | - |
| 12 | $\overline{DRVSB}$ | ← | Selecionar Drive B | B0 | 25 | 72J11.34 |
| 14 | $\overline{DRVSA}$ | ← | Selecionar Drive A | - | - | - |
| 16 | $\overline{MOTEB}$ | ← | Ativar Motor Drive B | B1 | 24 | 70J11.33 |
| 18 | $\overline{DIR}$ | ← | Selecionar direção | B3 | 22 | 66J11.31 |
| 20 | $\overline{STEP}$ | ← | Passo cabeça leitura/escrita −↓_ | B8 | 32 | 71J10.33 |
| 22 | $\overline{WDATA}$ | ← | Escrita de Dados −↓_ | B2 | 23 | 68J11.32 |
| 24 | $\overline{WGATE}$ | ← | Ativação de modo escrita | B5 | 20 | 62J11.29 |
| 26 | $\overline{TRK00}$ | → | Faixa 0 | D8 | 68 | 58J11.25 |
| 28 | $\overline{WPT}$ | → | Proteção Escrita | - | - | - |
| 30 | $\overline{RDATA}$ | → | Leitura de Dados | D4 | 81 | 28J10.14 |
| 32 | $\overline{SIDE}$ | ← | Seleção de cabeça | B4 | 21 | 64J11.30 |
| 34 | $\overline{DSKCHG}$ | → | Disco ejetado | D9 | 69 | 54J11.26 |

As informações da tabela apresentada contêm, nesta ordem, o pino do conector padrão PC 34, o sinal contido que é transportado pela conexão física, o sentido da informação (vem ou vai para FDC), descrição do sinal, porta do microcontrolador que receberá a informação, o pino do microcontrolador e o pino da placa de expansão (*expansion board* – EB). O código contido na coluna “Pino EB” foi elaborado da seguinte forma: os primeiros dois dígitos indicam o pino da placa de desenvolvimento, os próximos três caracteres identificam o conector da placa de expansão, o ponto é usado para separação e finalmente os últimos dois dígitos identificam o pino do conector da placa de expansão.

A Figura 32 apresenta o diagrama esquemático do circuito conversor de nível, montado de forma experimental, utilizado para conectar os pinos da placa de extensão do microcontrolador ao cabo que conecta à controladora de disquetes.

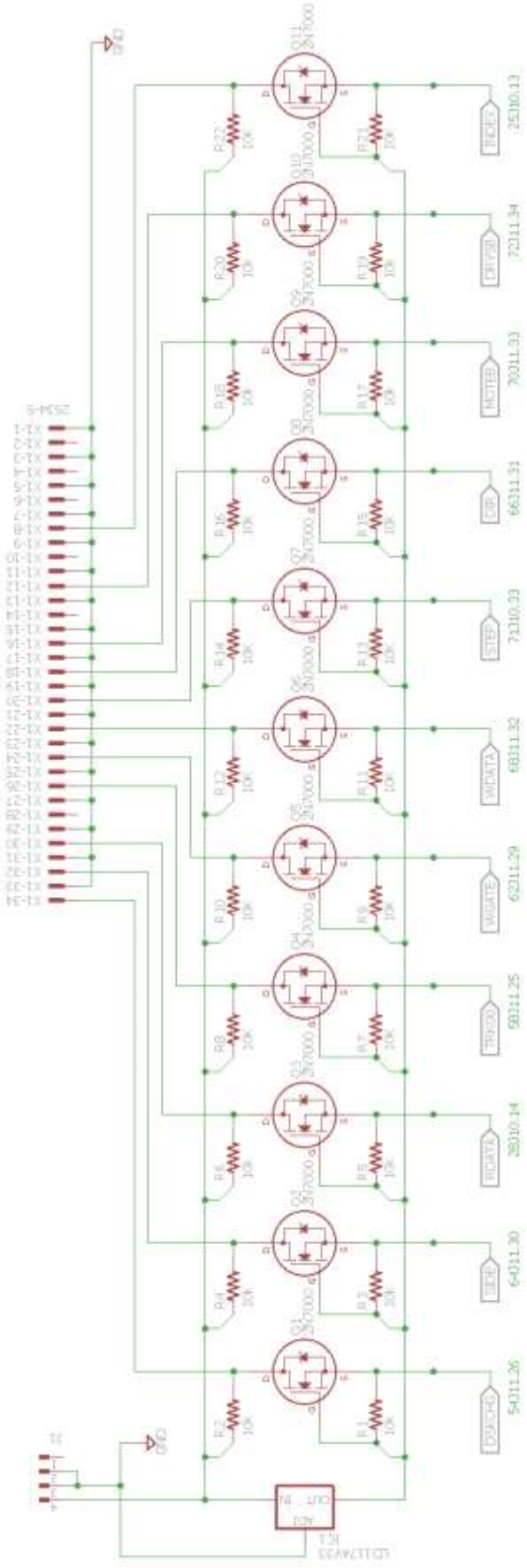

**Figura 32 - Diagrama esquemático do conversor de nível DC.**

#### 2.5.9.2. Apresentação dos resultados

Para realização das medições dos sinais de interesse foi utilizado um osciloscópio da Hewllet Packard, modelo 54540C, com quatro canais. Nas figuras geradas pelo equipamento o sinal de cor amarela representa $\overline{TRK00}$, o sinal de cor verde representa $\overline{STEP}$, o de cor azul $\overline{DRVSA}$ e o de cor vermelha $\overline{DIR}$.

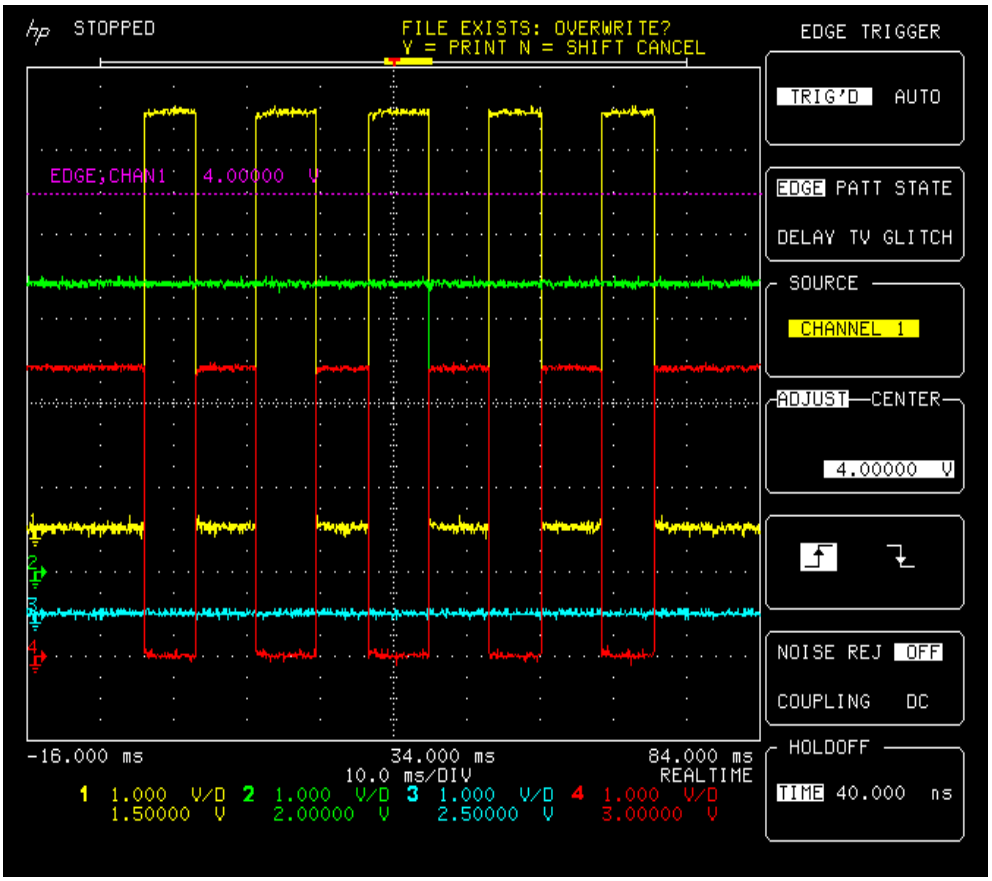


**Figura 33 - Teste da FDC.**

A Figura 33 ilustra um teste realizado para entender o funcionamento da controladora de disquetes. Nota-se que o sinal $\overline{DRVSA}$ está ativo e que os sinais de $\overline{DIR}$ e $\overline{TRK00}$ alternam-se. Isso acontece porque a controladora está realizando comandos de $\overline{STEP}$, que não são visíveis na escala de tempo ajustada no osciloscópio, mas a cada comando percebe-se que a unidade de disquetes responde através de $\overline{TRK00}$ que saiu da trilha 0 de acordo com a direção selecionada e logo em seguida, quando $\overline{DIR}$ está em nível alto a situação se inverte e quando ocorre $\overline{STEP}$ a FDD sinaliza que voltou para trilha 0.

Na Figura 34 verifica-se o funcionamento do adaptador conectado à controladora de disquetes da mesma maneira que a FDD, nas mesmas condições de operação. O adaptador inicia sua operação sempre com o sinal de $\overline{TRK00}$ em nível baixo. Verifica-se que quando o sinal de $\overline{DIR}$ está em nível baixo e ocorre o sinal de $\overline{STEP}$ o adaptador responde sinalizando que saiu da trilha 0.

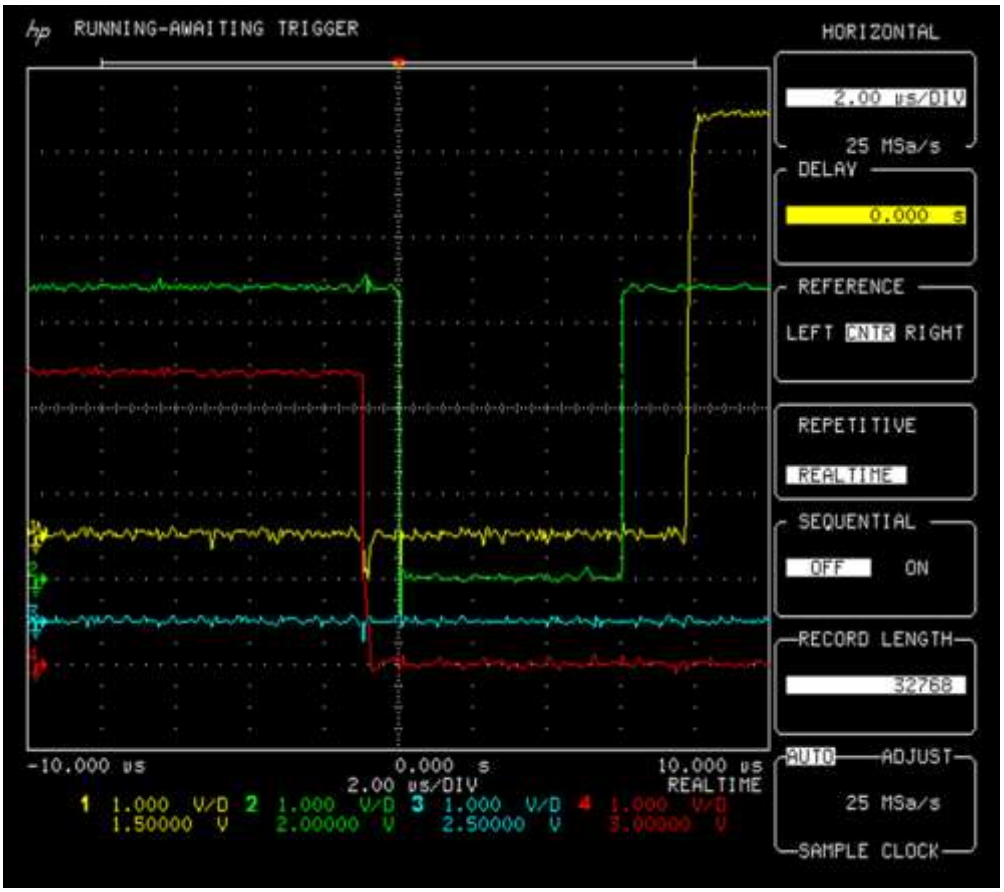


**Figura 34 - Adaptador sinaliza que saiu da trilha 0.**

Na Figura 35 é apresentado o resultado com a operação inversa, quando o adaptador recebe o sinal de $\overline{STEP}$ enquanto $\overline{DIR}$ está em nível alto, retorna à trilha 0, porque anteriormente estava na trilha 1.

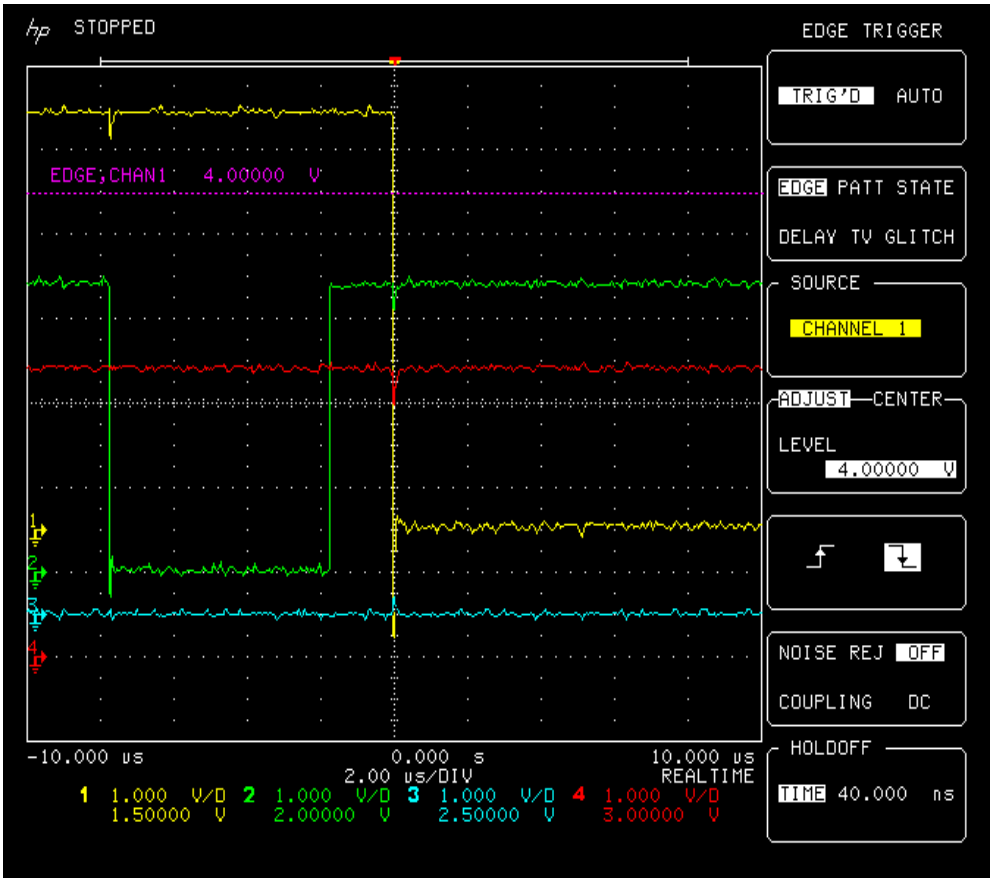


**Figura 35 - Adaptador sinaliza que está na trilha 0.**

Quando $\overline{DRVSA}$ é desabilitado todos os outros sinais são desabilitado também, como pode ser verificado na Figura 36.

As medidas para verificação do sinal MFM foram realizadas depois de transcorrido um período em que o circuito ficou exposto a variações de temperatura e humidade. Na Figura 37 pode-se verificar que a resposta em frequência do circuito foi comprometida e nestas condições a controladora não é capaz de interpretar os dados recebidos do adaptador, mas é possível perceber que o CODEC do adaptador respeita os tempos impostos pela modulação MFM.

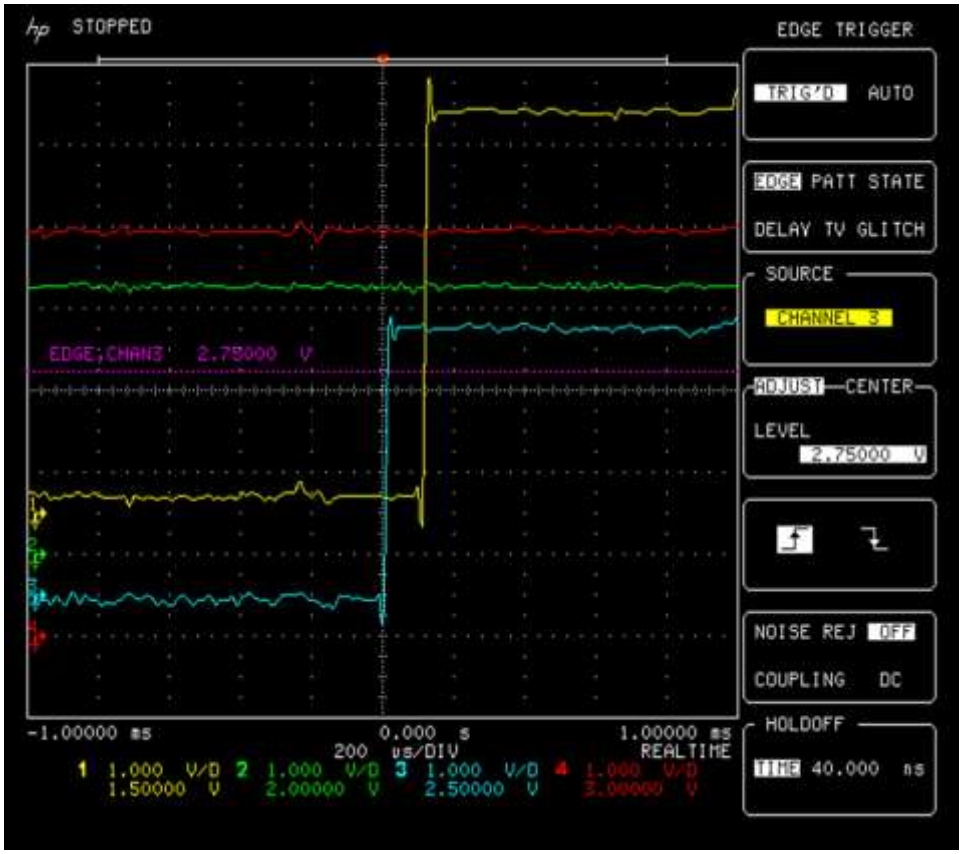


**Figura 36 - Seleção de *drive* é desabilitada.**

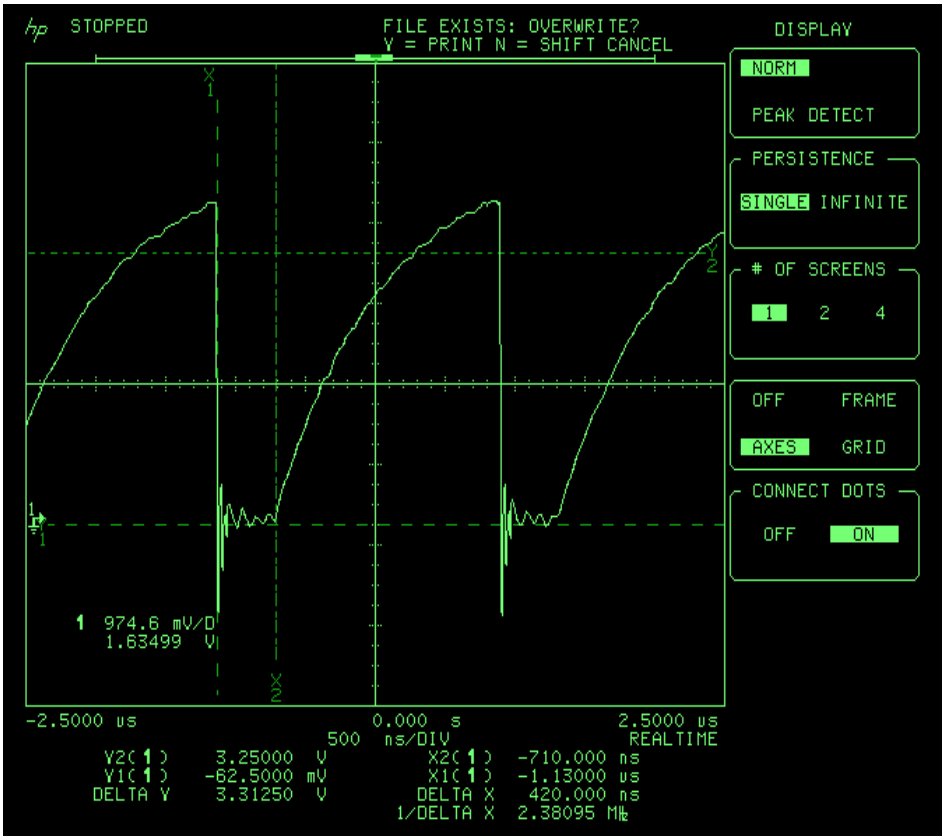


**Figura 37 - Sinal do codificado MFM do adaptador com problemas.**

## 3. CONCLUSÃO

A elaboração do projeto partiu de uma necessidade simples e acabou se transformando em um dispositivo interessante. A elaboração das possibilidades de acesso criadas a partir da alteração técnica do robô, com a instalação do adaptador USB, foi a etapa que mais trouxe incentivos para seu desenvolvimento.

Obviamente que existe mais de uma solução possível para o problema apresentado, substituição da unidade de disquete, mas o desafio proposto impulsionou o projeto para um nível mais alto justamente porque a solução óbvia poderia trazer tantos transtornos quanto manter a utilização da tecnologia antiga.

Embora nem todas as etapas do projeto proposto tenham sido realizadas de forma que fosse possível deixar um dispositivo operando no módulo de controle do robô, já é possível ter uma previsão dessa etapa. Em poucos meses é possível montar o adaptador e instalá-lo para permitir o uso da tecnologia USB nas aplicações do equipamento da ABB.

Os resultados obtidos dentro da limitação apresentada foram satisfatórios, considerando que a parte principal do projeto funcionou bem, comprovando a capacidade que o adaptador tem de estabelecer comunicação com a porta de controle da unidade de disquete do robô IRB-1400, da ABB.

As melhorias deste projeto estão já elencadas em seu desenvolvimento. Atualmente ele opera limitado a interpretação dos dados da controladora, ou seja, somente a interface que realiza a comunicação com o ambiente interno está operacional. O próximo passo é implementar a interface que realiza a comunicação com dispositivos de armazenamento portáteis. Em seguida vem a implantação da memória RAM, para permitir ao dispositivo que trabalhe com maior velocidade de acesso e independente da conexão de um dispositivo de memória portátil. Poderão ser criadas outras interfaces que utilizem protocolos para comunicação com redes de computadores.

A continuidade deste projeto é de grande valia para o meio onde ele está inserido. Ainda pode proporcionar a qualquer aluno da instituição de ensino, de outras áreas de tecnologia, um desenvolvimento multidisciplinar, impulsionando novos projetos do mesmo tipo, já prevendo a utilização de novas tecnologias de acesso que já estão em uso.

A quantidade de informações necessárias para o desenvolvimento do projeto é grande e o mais importante é não interromper o desenvolvimento. Foram realizados muitos testes e simulações para verificar uma série de informações desencontradas. Alguns testes inclusive substituíam os métodos de acesso e tornariam o projeto mais simples e fácil de construir, porém, o objetivo sempre foi trazer facilidade aos usuários do robô, e esse tipo de alteração não seria convergente com a ideia inicial, por isso, mesmo com as dificuldades e possibilidade de não entregar o projeto completo, o conceito inicial foi mantido.

O envolvimento com o projeto foi estimulante e proveitoso. Infelizmente em certo ponto da vida algumas atividades nos tomam muito tempo e prejudicam o andamento de outras. Espero ter oportunidade de continuar desenvolvendo o adaptador USB e auxiliar de alguma forma no crescimento acadêmico dos colegas de curso e futuros colegas de profissão.

# REFERÊNCIAS

[1] SÁ, José Pedro Patrício Gonçalves de. **Emulador em Hardware de Floppy Disk Drive com acesso sem fios**. 2011. 93 p. Dissertação (Mestrado em Engenharia)– Faculdade de Engenharia da Universidade do Porto, Porto, 2011.

[2] PROGRAMMING Floppy Disk Controllers. Disponível em <http://www.isdaman.com/alsos/hardware/fdc/floppy.htm > Acesso em: 05 agosto 2014.

[3] SINGLE/Double Density Floppy Disk Controller. Disponível em <http://retro.icequake.net/dob/files/bleuge/00info/8272sp.htm> Acesso em: 05 agosto 2014.

[4] JOHNSON, Herb. **Tech information on floppy disks drives and media**. Disponível em <http://retrotechnology.com/herbs_stuff/drive.html#info> Acesso em: 07 setembro 2014.

[5] CTL122: FIRMWARE Walkthrough (Pt. 2 USB Configuration). Disponível em <http://stephanos.io > Acesso em: 07 setembro 2014.

[6] BOUNDLESS Virgin WebPlayer 200MHZ Internet Appliance Floppy Drive Connector. Disponível em < http://www.webplayer.0catch.com/html/Floppy_Connector.htm> Acesso em: 07 setembro 2014.

[7] INTEL CORPORATION**. Interface Between 82077AA/SL and the Floppy Drive**. Disponível em <http://www.intel.com/design/archives/periphrl/docs/7282.htm > Acesso em: 07 setembro 2014.

[8] ECONÓMICO-FINACEIRO. **O Robocop 2014**. Disponível em <http://economicofinanceiro.blogspot.com.br/2014/02/o-robocop-2014.html> Acesso em: 26 maio 2015.

[9] MEMORY device,USB Storage Device,Purple USB vector. Disponível em <http://www.dailyfreepsd.com/vector/misc/memory-deviceusb-storage-devicepurple-usb-vector.html> Acesso em 27 maio 2015.

[10] ARTIMAR**. Portfolio Completo de Microcontroladores USB de 8, 16 e 32-bit**. Disponível em <http://www.artimar.com.br/news.asp?id=57&tipo=a1> Acesso em: 27 maio 2015.

[11] CHIDANANDAN, Archana. **Fat12description**. 2004. 9p. Rose-Hulman Institute of Technology – Terre Haute. Disponível em <https://www.coursehero.com/file/6918624/FAT12Description/> Acesso em: 30 maio 2015.

[12] RS COMPONENTS LTD. **PIC32 I/O Expansion Board DM320002**. Disponível em <http://uk.rs-online.com/web/p/products/549394/?utm_campaign=applegate.co.uk&utm_medium=referral&utm_source=applegate.co.uk> Acesso em: 02 junho 2015.

[13] TEAC CORPORATION. **TEAC FD-235HF-C829: Micro Floppy Disk Drive**. Specitication. Rev. A. 31 p.

[14] NEC ELETRONICS U.S.A. INC.. Single/Double Density Floppy disk Controller. In:_____. **µPD765A**. 1984. Cap. 9, p 459. Disponível em: <http://www.classiccmp.org/dunfield/r/765.pdf > Acesso em: 05 agosto 2014.

[15] NATIONAL SEMICONDUCTOR CORPORATION. **PC87338/PC97998: ACPI 1.0 and PC98/99 Compliant Super I/O**. 221 p. Santa Clara. 1998.

[16] INTEL COPRPORATION. **SBC 202: Double density Diskette Controller Hardware Reference Manual**. 86 p. Santa Clara. 1977.

[17] MICROCHIP TECHNOLOGY INC.. **PIC32MX Family Architecture Overview** <disponível em <http://www.microchip.com/pagehandler/en-us/family/32bit/architecture-pic32mxfamily.html> Acesso em: 26 maio 2015.

[18] MICROCHIP TECHNOLOGY INC. **PIC32 USB Starter Kit III**. <http://www.microchip.com/DevelopmentTools/ProductDetails.aspx?PartNO=dm320003-3> Acesso em: 11 novembro 2014.

[19] FRANCHIN, Marcelo N. **Elementos de Robótica**. Departamento de Engenharia Elétrica, Faculdade de Engenharia de Bauru, Universidade Estadual Paulista, 2005. Apostila.

[20] INTEGRATED TECHONOLOGY EXPRESS, INC.. Low Pin Count Input / Output (LPC I/O). **IT8711F Programming Guide V0.1**. 32 p. Hsin-Chu. 2001. Disponível em <http://www.datasheetarchive.com/dl/Datasheet-028/DSA00495365.pdf> Acesso em: 07 setembro 2014.

[21] ABB ROBOTICS PRODUCTS. **Product On-line Manual IRB 1400**. 1998. 406p. Disponível em <www.abb.com> Acesso em: 07 setembro 2014.