UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CONCEITO "A"
DELET
EM 16/05/2002

Relatório do Estágio Supervisionado

Márcio Portal Longaray
2042/95-2

Porto Alegre, Maio de 2002.

FOLHA DE AVALIAÇÃO

Esta folha resume a avaliação do estágio feita pelo próprio aluno, pelo supervisor, pelo orientador e pela banca examinadora, que também atribuiu o conceito final do aluno nesta disciplina.

1. Preencher o campo comentário, quando achar necessário ou adequado;
2. Preencher o campo "Conceito" com A, B, C ou D, de acordo com sua avaliação: excelente, médio superior, médio ou insuficiente, respectivamente.

AUTO AVALIAÇÃO: Conceito: ( )

Assinatura do Aluno:_____ Data:____/____/____

Comentário:_____
_____
_____

AVALIAÇÃO DO SUPERVISOR: Conceito: ( )

Nome:_____
Assinatura:_____ Data____/____/____

Comentário:_____
_____
_____

AVALIAÇÃO DO ORIENTADOR: Conceito: ( )

Nome:_____
Assinatura:_____ Data____/____/____

Comentário:_____
_____
_____

AVALIAÇÃO DA BANCA EXAMINADORA

Nome do membro da banca	Assinatura	Conceito
Conceito da Banca		

Conceito final atribuído ao aluno ☐

Porto Alegre,_____de_____de_____.

SERVIÇO PÚBLICO FEDERAL

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ENG04 497 - ESTÁGIO SUPERVISIONADO III

RELATÓRIO DO ORIENTADOR

Nome do Aluno: MÁRCIO PONTES LONGARAY
Número da Matrícula: 204295-2
Endereço: ANITA GARIBALDI 1750/202
Cidade: POA UF: RS Fone: 33286182
Área de Atividade:
Data de Início: / / Data de Término: / /
Duração:
Supervisor: ROMEU REGINATTO
Orientador: WALTER FETTER LAGES

APRECIAÇÃO DA ENTIDADE	A	B	C	D
Interesse pelo Estagiário	X			
Nivel do Estágio oferecido		X		

APRECIAÇÃO DO ESTAGIÁRIO	A	B	C	D
Qualidade das tarefas realizadas	X			
Nivel das consultas feitas p/ Estagiário	X			
Aplicação de Conhecimentos adquiridos no curso		X		
Aproveitamento	X			

CONCEITO FINAL (A)

A=Excelente B=Médio Superior C=Médio D=Insuficiente

CONSIDERAÇÕES:

Porto Alegre, 30 de ABRIL de 2002

Assinatura do Orientador

SERVIÇO PÚBLICO FEDERAL

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ENG04 497 - ESTÁGIO SUPERVISIONADO III

RELATÓRIO DO SUPERVISOR

Nome do Aluno: MÁRCIO PORTAL LONGARAY
Número da Matrícula: 2042/95-2
Endereço: ANITA GARIBALDI 1750/202
Cidade: POA UF: RS Fone: 33286187
Área de Atividade:
Data de Início: / / Data de Término: / /
Duração:

Supervisor: ROMEU REGINATTO
Orientador: WALTER FETTER LAGES

	A	B	C	D
Assiduidade	X			
Interesse demonstrado	X			
Capacidade de adaptação	X			
Disciplina e responsabilidade	X			
Aproveitamento	X			

A = Excelente
B = Médio Superior
C = Médio
D = Insuficiente

CONSIDERAÇÕES:

Porto Alegre, 30 de ABRIL de 2002.

Assinatura do Supervisor

SERVIÇO PÚBLICO FEDERAL

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ENG04 497 - ESTÁGIO SUPERVISIONADO III

RELATÓRIO DO ESTAGIÁRIO

Nome do aluno: MÁRCIO PORTAL LONGARAY
Número da matrícula: 2042/95-2
Endereço: R. ANITA GARIBALDI 1750/202
Cidade: POA UF: RS Fone: 33286181
Área de atividade:
Data de início: / / Data de término: / /
Duração:

Supervisor: ROMEU REGINATTO

Orientador: WALTER PETER LAGES

	A	B	C	D
Assistência do Orientador	X			
Assistência do Supervisor	X			
Organização da Entidade	X			
Conhecimentos adquiridos em relação às exigências do Estágio	X			
Aproveitamento	X			

A = Excelente
B = Médio Superior
C = Médio
D = Insuficiente

CONSIDERAÇÕES:

Porto Alegre, 30 de ABRIL de 2002

Assinatura do Estagiário

Sumário

1. **LISTA DE FIGURAS** II
2. **RESUMO** III
3. **INTRODUÇÃO** 1
4. **O SISTEMA ATUAL** 2
5. A ARQUITETURA PROPOSTA 3
6. **O BARRAMENTO CAN** 4
   - 4.1. Um barramento serial 4
   - 4.2. O que o CAN apresenta 4
   - 4.3. Tipos de frames 7
   - 4.4. Componentes disponiveis 8
7. **A PLACA TINI** 9
   - 5.1. Introdução 9
   - 5.2. O hardware da TINI 10
   - **5.3.** O ambiente da TINI 11
6. **O PROJETO** 12
   - 6.1. TINI 12
   - 6.1. Alimentação 12
   - 6.2. Interface ethernet e serial 12
   - 6.3. Interface CAN 12
   - 6.4. Mapa de memória 13
   - 6.5. Decodificação dos periféricos 14
   - 6.6. Encoder 15
   - 6.7. PWM 16
   - 6.8. Ponte H 18
   - 6.9. PCB 19
   - 6.10. Firmware 19
8. **CONCLUSÃO** 21
9. **REFERÊNCIAS** 22

1. Lista de Figuras

Figura 1 O Janus é um robô antropomórfico
Figura 2 Os processadores das juntas se comunicam com o PC através de um barramento CAN
Figura 3 Principio de arbitragem destrutiva
Figura 4 Frame de mensagem no formato padrão
Figura 5 Placa Tini Simm-72
Figura 6 A Tini permite conversão de protocolos
Figura 7 O ambiente da TINI
Figura 8 Conexão Física do CAN de acordo com a norma ISSO 11898
Figura 9 A pinagem sugerida pelo padrão CiA DS-102
Figura 10 O mapa de memória da TINI
Figura 11 Estrutura da macrocélula de um PLD tipo GAL
Figura 12 Sinais nos canais do *encoder*
Figura 13 Um *encoder* incremental
Figura 14 Esquema do PWM
Figura 15 Esquema da Ponte H
Figura 16 Esquema básico de ligação do LT1162

2. Resumo

O presente relatório descreve as atividades empreendidas durante o período de estágio supervisionado realizado no laboratório de automação industrial do Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul.

Os trabalhos envolveram o planejamento da implementação de um barramento CAN para o controle do robô Janus, a escolha de uma arquitetura adequada para o caso, o estudo da placa TINI adquirida pelo laboratório para fazer o acionamento microcontrolado das juntas do robô, a análise das funcionalidades relevantes desta e o desenvolvimento de todo o circuito eletrônico periférico necessário. A documentação da placa TINI disponibilizada pelo fabricante (Dallas Semicondutores) [1] deixa muito a desejar; de forma que em muitos momentos fez-se necessário montar protótipos intermediários para averiguação, como também contatos diretos com o departamento técnico para esclarecimento de dúvidas. O ambiente de desenvolvimento (Java Sun), constituído dos *softwares* de comunicação e programação da TINI, também dificultou o bom andamento do projeto até que se conseguisse configurá-lo corretamente.

Uma placa de circuito impresso foi desenvolvida para prototipação e testes iniciais. Como a linguagem de programação Java não é própria para um sistema de tempo real como o controle de um robô, conforme será comentando no capítulo *firmware* do presente relatório, foi necessário escrever o firmware do sistema em linguagens mais adequadas como o Assembly e o C.

3. Introdução

O estágio foi realizado no laboratório de automação do DELET sob orientação do professor Dr. Walter Fetter Lages, pertencente ao grupo de controle, automação e robótica (GCAR) cujas principais atividades compreendem a pesquisa, orientação e formação de recursos humanos nas áreas de Automação Industrial, Controle de Sistemas Dinâmicos e Robótica. As atividades do GCAR são atualmente voltadas às seguintes linhas de pesquisa:

1. Arquiteturas Computacionais para Sistemas de Controle e Automação Industrial
2. Sistemas de Tempo-Real
3. Barramentos Industriais
4. Controle de Processos
5. Sistemas não Lineares
6. Controle sob Restrições
7. Sistemas Flexíveis de Manufatura
8. Robótica

Os trabalhos desenvolvidos durante o estágio são parte constituinte de um planejamento de atividades com mais longo prazo, pois terão continuidade durante o curso de pós-graduação. Esta primeira etapa empreendida durante o período de estagio supervisionado compreende a implementação do sistema de sensoriamento e acionamento do robô Janus e a camada física de um barramento industrial para comunicação destes sensores e atuadores. Durante o curso de pós-graduação serão implementadas então as demais camadas numa abordagem bottom-up.

4. O Sistema Atual

O Janus é uma das atividades de pesquisa mantidas pela área de robótica do GCAR, e consiste de um robô antropomórfico dotado de dois braços e um sistema de visão estéreo consistindo cada braço de 8 juntas e a cabeça de 2 juntas. [2]

A medição da posição é feita por *encoders* incrementais óticos cuja saída é composta por dois sinais em quadratura e um sinal de referência (não presente nas juntas 6, 7 e 8). Um sinal de ponto de calibração (sensor indutivo de fim de curso) é obtido em cada uma das juntas para ser usado no processo de determinação do ponto zero. Os atuadores são compostos por motores CC de imã permanente, recebendo os acionadores uma referência de ±10V que é então amplificada e aplicada nos motores.

A leitura do *encoders* é feita numa placa padrão mezanino que desempenha as tarefas de controle digital de movimento e libera o processador *host* para outras tarefas. Enquanto as informações do *host* são aceitas assincronamente com relação as funções de controle, o comando do motor é computado com uma base de tempo programável. Existe ainda um outro modelo de placa mezanino que constitui um módulo de entradas e saídas digitais (+24V) utilizados para realizar sinalizações e intertravamentos da parte de potência e trabalhos de CLP em geral.

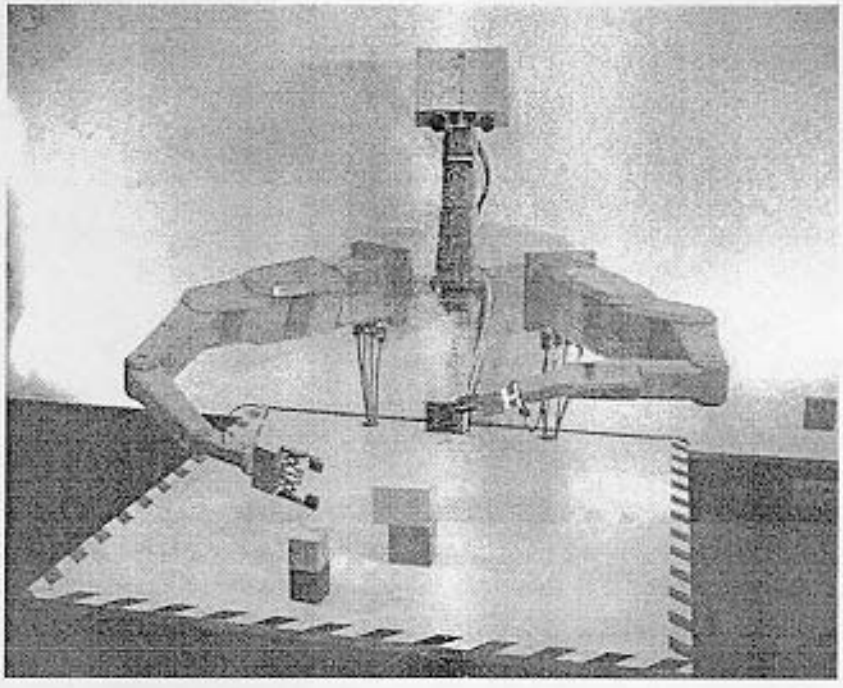


Figura 1: O Janus é um robô antropomórfico

4.1. A ARQUITETURA PROPOSTA

A moderna técnica de automação é caracterizada por uma crescente descentralização das funções de processamento por meio de comunicação serial. O emprego de sistemas de barramentos seriais no lugar das técnicas de cabeamento tradicionais permite uma maior flexibilidade dos sistemas com relação a alterações e ampliações. Cada uma destas juntas deve ter capacidade de efetuar um controle local e também comunicar com uma unidade de processamento de maior capacidade como, por exemplo, um PC. Desta forma a comunicação poderia suceder hipoteticamente de forma a cada nodo receber um valor de referência da unidade central e localmente implementar a malha de controle ou operar simplesmente como um processador de I/O do PC.

Foi idealizada uma arquitetura constituída de um PC conectado na rede do laboratório e internet, que fará, através de uma placa PCI, o elo com o barramento CAN onde estarão comunicando os sensores e atuadores conforme pode ser visto na figura 2.

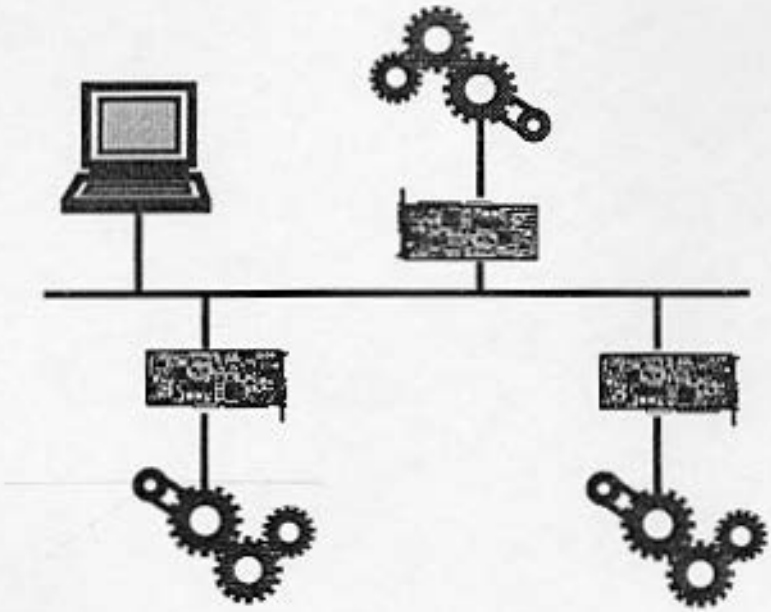


Figura 2: Os processadores das juntas se comunicam com o PC pelo barramento CAN.

5. O Barramento CAN

Com os complexos sistemas mecânicos como os dos carros, robôs, navios, aparelhos biomédicos, hoje em dia a ultima palavra em segurança só pode ser assegurada por controle eletrônico de todos os subsistemas. A necessidade de segurança e eficiência nos veiculos reforçou a importância da eletrônica embarcada. Hoje se tem eletrônica para controlar a injeção de combustível, emissão de gás, sistemas de freios ABS e muito mais. O CAN foi projetado para atender os requisitos de cada sistema de controle mecânico/eletrônico ou subsistema, e permite comunicação em alta-velocidade entre módulos eletrônicos conectados a um barramento serial bidirecional a dois fios para controlar cada subsistema. O CAN oferece flexibilidade, pois módulos de características especiais podem ser adicionados a um conjunto fixo de módulos básicos sem a necessidade de reconfiguração.

Desenvolvido para operar em ambientes de muito ruido, uma propriedade singular do protocolo CAN é a sua capacidade de tratamento automático de erro. Exaustivas simulações [6] revelaram que menos de um mal-funcionamento de comunicação não-detetado ocorreria em muitos milhares de carros durante sua vida útil (o CAN foi inicialmente projetado para carros e só mais tarde encontrou aplicações em outras áreas, sendo compreensível que a maior parte da literatura introdutória faça referência a estes primeiros testes realizados para uma única aplicação).

5.1. UM BARRAMENTO SERIAL

A idéia básica por detrás do barramento CAN é simples: cada dispositivo elétrico, sensor, atuador, ou combinação destes no subsistema está conectado a um pequeno módulo computadorizado. Isto representa uma total ruptura do conceito de controle centralizado, computadorizado conectado aos módulos via um complexo sistema de cabeamento. O protocolo CAN está baseado em simples interconexões de módulos via uma linha de dados serial, e desta forma oferece uma tremenda redução da fiação elétrica no carro ou robô (não é pouco usual para um carro luxuoso atual ter algo como 2 Km de fio a bordo representando um peso de mais de 100Kg) [3]. A implementação mais simples do CAN é uma em que todos os módulos estão interconectados pelo chassi do carro ('terra') e uma única linha de dados.

Cada módulo CAN tem em principio um único fio de alimentação, e é controlado localmente por um microprocessador (ou controlador CAN (do qual já estão disponiveis diversos tipos oferecidos por fabricantes como a Intel e a Phillips) [4]. Estes controladores rodam *software* dedicado que implementa o protocolo CAN. Originariamente definido pela Bosch em 1987, este protocolo tem sido reconhecido como extremamente confiável em ambientes de muito ruido [3], e atualmente é empregado em muitos sistemas de controle modernos [3].

5.2. O QUE O CAN APRESENTA

O barramento CAN prevê suporte para estruturas chamadas *multimaster* que podem ser familiares dos sistemas de comunicação em rede dos escritórios como, por exemplo, o Ethernet.

Cada usuário do barramento ('nodo', ou simplesmente 'unidade') tem permissão para começar a enviar mensagens tão logo o barramento esteja livre (CSMA: *carrier sense multiple access*) [4].
Em contraste com outros sistemas de barramento, como os com *Token Ring* [4], os usuários não precisam esperar por uma permissão para poderem começar a transmitir.

Um outro aspecto do CAN é a transferência de mensagens orientada a objetos. Muitos outros tipos de barramentos, como o I2C [3,5], são baseados no endereçamento de usuários. Nestes sistemas cada usuário tem seu próprio, único endereço. Se a unidade 'A' quer enviar uma mensagem para a unidade 'B', ela transmite um bloco de informações que contém o endereço de 'B', além do dado. A unidade 'B' por sua vez só aceita os dados quando reconhece seu próprio endereço no barramento.

O CAN Bus está baseado num principio diferente. As unidades do barramento não recebem endereços mas às mensagens são atribuídos identificadores de objeto. Num sistema de medidas, portanto, cada quantidade medida (temperatura, voltagem, velocidade da máquina) pode ter seu próprio identificador atribuído o qual é transmitido junto com a quantidade medida. Cada módulo conectado ao barramento pode receber e processar esta quantidade medida, até onde ela for relevante para seu funcionamento no sistema. O identificador de objetos é uma palavra de 11-bits, que permite o uso de até 2048 objetos diferentes. Na prática este número é reduzido a 2032 porque alguns identificadores estão reservados para funções especiais.

Como acontece com qualquer sistema de barramentos multimaster, deve-se prevenir que ocorram colisões no barramento quando duas ou mais unidades acham que o barramento está livre, e começam a transmitir. O protocolo CAN oferece um eficiente sistema de arbitragem do barramento orientado a prioridades, CA (Collision Avoidance), para prevenir corrupção dos dados devido a colisões no barramento.

O protocolo CAN define dois níveis de bit no barramento: o nível dominante e o recessivo. O nível dominante sobrescreve um nível recessivo. Estes níveis são mais facilmente implementáveis com ajuda de drivers coletor-aberto. Um nível recessivo do barramento é estabelecido quando todos os transistores estão cortados (entradas A, B, C e D em 0 V). Se um único transistor for ligado, o nível recessivo é sobrescrito por um nível dominante. Durante a fase de arbitragem o identificador de 11-bit é posto no barramento bit-a-bit. Concomitantemente o transmissor lê o estado do barramento e compara com o bit que foi enviado. Se os dois níveis forem diferentes a operação de transmissão é imediatamente interrompida. Desta forma colisões no barramento são prevenidas. Um exemplo: suponha-se que as unidades 'A' e 'B' comecem a transmitir simultaneamente. A unidade 'A' transmite um objeto com identificador 01100111001, e a unidade 'B' um objeto com identificador 01110111001.

Suponha-se ainda que o nível dominante do barramento seja '1'. As unidades 'A' e 'B' enviam cada seu primeiro, segundo e terceiro bit. Uma vez que os níveis dos bits são idênticos nenhuma unidade vai detectar qualquer erro quando ler o nível escrito no barramento. Entretanto quando o quarto bit é posto no barramento, a unidade 'A' detecta um erro, porque o bit recessivo transmitido por ela foi sobrescrito pelo bit dominante originado pela unidade 'B'. O resultado é que a unidade 'A' para de transmitir e a unidade 'B' pode continuar com a sua transmissão (note que os transistores invertem seus sinais). O sistema CA tem duas vantagens. Primeiramente o valor binário

do identificador permite que unidades tenham diferentes niveis de prioridade atribuidos. Quando acontece de duas unidades do barramento transmitirem simultaneamente, a mensagem da unidade com o menor identificador será transmitida primeiro. A segunda vantagem é que a mensagem não é perdida. Nos outros protocolos ambas unidades do barramento param suas transmissões quando uma colisão ocorre, e começam geradores randômicos de temporização antes de tentar enviar a mensagem de novo. Este sistema é conhecido como CD (collision detection).

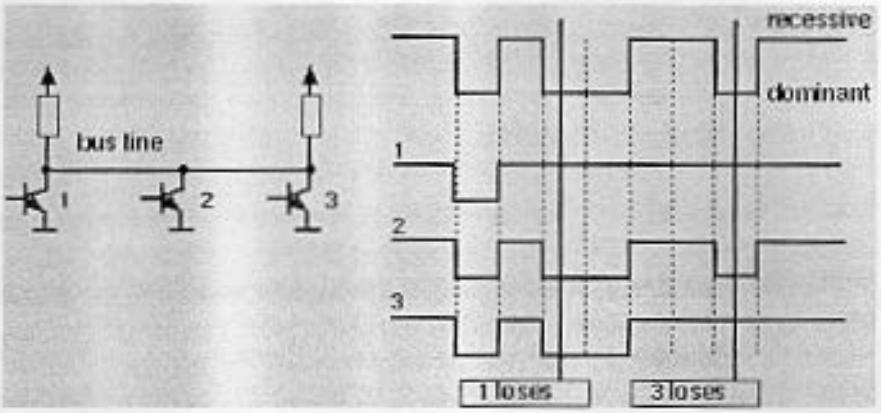


Figura 3: Princípio da arbitragem destrutiva

Dado o grande numero de funções de controle automotivo num veiculo, fica evidente que uma comunicação em tempo-real é necessária. Desta forma a velocidade do CAN deve ser tão alta quanto possível. Em princípio, a velocidade do CAN é limitada apenas pelo tempo de propagação na fiação do barramento e nos *drivers* do barramento. Supondo que um par trançado seja usado e que os *drivers* do barramento tenham um atraso de 100ns, uma taxa de 1Mbit/s pode ser atingida num barramento com um comprimento efetivo de 40m. Quando a taxa for diminuida, o comprimento máximo do barramento aumenta de acordo até 10.000m.

Um objeto tem espaço para blocos de dados de até 8 bytes cada. Isto talvez pareça um numero pequeno, mas tenha-se em mente que o CAN Bus não foi projetado para enviar grandes quantidades de dados. Normalmente 8 bytes são mais do que suficientes para enviar valores de medidas e estados do processo.

A taxa de dados efetiva é uma característica importante quando se trata de avaliar a velocidade "real" dos dados num barramento. A taxa de dados efetiva é basicamente a taxaentre o comprimento efetivo dos dados e o bloco de mensagens completo (cuja estrutura é discutida mais adiante). A razão funciona a um máximo de 57% para o CAN bus. Dado o curto comprimento dos dados 57% é um valor relativamente alto.

O protocolo CAN não contém especificações sobre o *hardware* de acoplamento ou fiação entre os módulos. Dependendo da aplicação pode-se usar um fio, dois fios ou elo de fibra ótica. Na maioria dos casos um par de fios trançados é utilizado com um *hardware* de acoplamento derivado do padrão RS-485 [3].

A integridade de dados é da maior importância em aplicações automotivas. Medidas poderosas para detecção de erro, sinalização e autodetecção estão implementadas em cada modulo CAN.

Medidas protetoras como um CRC (*cyclic redundancy check*) de 15 bits, *bit stuffing* e MSF (*message frame check*) estão implementadas no protocolo CAN, que atinge uma probabilidade de erro total residual total menor que 3.10E-5 para mensagens corrompidas não detectadas. Isto significa que um bit corrompido passa despercebido num total de 33.000bits corrompidos. O sistema é capaz de detectar confiavelmente até cinco bits corrompidos numa mensagem. Quando uma das unidades detecta um erro numa mensagem, ela envia um frame de erro, que sinaliza para todas as outras unidades que desprezam a mensagem que continha o erro. Isto é feito para assegurar que todas as unidades recebam a mesma mensagem isenta de qualquer erro. O frame de erro também serve para solicitar ao emissor da mensagem corrompida que a reenvie.

5.3. TIPOS DE FRAMES

O protocolo CAN especifica dois tipos de frames de mensagens para o intercambio de dados entre nós CAN. Os dados são enviados no frame de dados enquanto o frame remoto 'remote frame' serve para interrogar outras unidades do barramento. Uma unidade atuando como receptora de certos dados pode iniciar a transmissão do respectivo dado pelo seu nó-fonte pelo envio de um frame remoto. O inicio do frame (os tipos 'dado' ou 'remoto' são marcados pelo inicio do bit do frame). Todas as unidades do barramento estão sincronizadas na borda descendente (negativa) deste bit. A arbitragem subseqüente contém o identificador de 11 bits previamente discutido e o bit RTR (remote transmission request). RTR=0 num frame de dados e RTR=1 num frame remoto. O campo de controle serve para enviar o comprimento do dado efetivo em bytes (data length code DLC). Somente os quatro bits de mais baixa ordem são utilizados; os dois restantes são reservados para extensões. O campo de controle é ignorado num frame remoto uma vez que ele não contém nenhum dado efetivo neste caso. Os dados efetivos seguem o campo de controle (não há campo de dados num frame remoto), e podem ter oito bytes de comprimento. O campo CRC contém o CRC de 15 bits e um delimitador CRC que é um único bit recessivo que serve para dar ao receptor tempo de processar o CRC. O CRC é computado usando todos os bits prévios no frame da mensagem. O frame termina com o campo de confirmação (ACK) e o campo fim-da-mensagem. O campo ACK tem dois bits de comprimento e contem o slot ACK e p delimitador ACK. Durante o campo ACK a unidade transmissora envia a bits recessivos.

Todos os receptores que receberam mensagens válidas reportam isso ao transmissor enviando um bit dominante durante o slot ACK. O campo de fim do frame finalmente consiste de sete bits recessivos. Frames de dados e frames remotos são separados dos frames precedentes por um bit de campo chamado espaço entre frames (interframe space), que tem um comprimento mínimo de três bits. Em adição ao frame de dados e o frame remoto existem ainda outros tipos de frames que não estão diretamente relacionados a transmissão de dados. Um nó que requeira, mas tempo de processamento pode acusar um frame de sobrecarga (overload frame) que faz com que as outras unidades do barramento atrasem as transmissões dos próximos frames de dados ou frames remotos. A função do quarto tipo de frame, o frame de erro, foi discutido anteriormente.

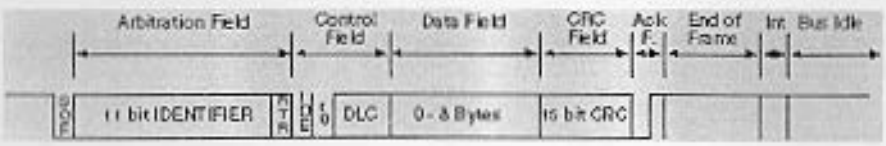


Figura 4: Frame de mensagem no formato padrão

5.4. COMPONENTES DISPONÍVEIS

Com a disseminação do protocolo CAN devido a sua vasta aceitação e também em grande parte devido ao investimento nele feito por grandes fabricantes de semicondutores como a Intel e a Phillips, existe hoje no mercado uma grande variedade de controladores implementando o protocolo. Alguns destes, como o SJA100 da Phillips, são integrados para serem usados em conjunto com um microcontrolador como o 8051 quando se deseja montar uma unidade local que controle determinado processo de forma compacta.

Todavia para estas alternativas estes mesmos fabricantes já oferecem microcontroladores que contém o controlador CAN internamente, como um periférico qualquer, da mesma forma como muitos já contém elementos digitais como PWM, ADC etc.

Por exemplo, o controlador 80C390 da Dallas Semicondutores, da família de controladores 8051, oferece o protocolo CAN implementado *on-chip* usando identificador padrão de 11 bits ou estendido com 29 bits, 15 centros de mensagens (*buffers*) por controlador e outro; por isso o 80C390 é um controlador bastante empregado. Seu preço também é muito atrativo comparado com os dos outros controladores do mesmo nível, com vantagem de ser bastante grande a disponibilidade de material para programação de compatíveis com a família 8051.

A placa TINI desenvolvida pela Dallas Semicondutores contém o controlador 80C390, controlador Ethernet, Real Time clock, e outros atrativos sendo sem comparação mais barata que a aquisição dos componentes discretos.

6. A Placa TINI

6.1. INTRODUÇÃO

Tiny InterNet Interface (TINI) é uma plataforma desenvolvida pela Dallas Semicondutores para dotar os projetistas de sistemas e desenvolvedores de *software* de um meio simples, flexível e econômico de projetar uma grande variedade de dispositivos de *hardware* capazes de se conectar diretamente a redes corporativas e residenciais.

Figura 5: Placa Tini Simm-72

A plataforma é uma combinação de um chip pequeno, mas poderoso, e um ambiente de execução de JAVA oferecendo capacidades de controle e comunicação e *networking*. As características do *hardware* são apresentadas ao desenvolvedor do *software* através de umconjunto de interfaces de programação de aplicações JAVA.

A capacidade de conexão da TINI expande a conectividade de qualquer dispositivo anexado ao permitir interação com sistemas remotos e usuários através de aplicações de rede padrão como, por exemplo, *web browsers*, e ao permitir conversão de protocolos. Desta forma podemos utiliza-la, por exemplo, no nosso projeto, para ler equipamentos industriais que comuniquem em padrão RS-232, levar suas informações através do CAN para o PC e deste para a Internet. Ou, se quisermos, levar pontos de rede a cada uma das placas e desta forma fazer o acesso utilizando o TCP/IP.

O objetivo primário da plataforma TINI é dar voz na rede a tudo desde pequenos sensores e atuadores a equipamentos de automação industrial e *legacy hardware*. Isto permite que os dispositivos sejam monitorados, controlador e gerenciados remotamente. A combinação de uma grande capacidade de I/O, o TCP/IP e um poderoso ambiente de programação orientada a objeto

possibilitam que programadores possam criar rapidamente aplicações que permitam não apenas controle local, mas também a capacidade de gerenciamento remoto de dispositivos baseados na TINI.

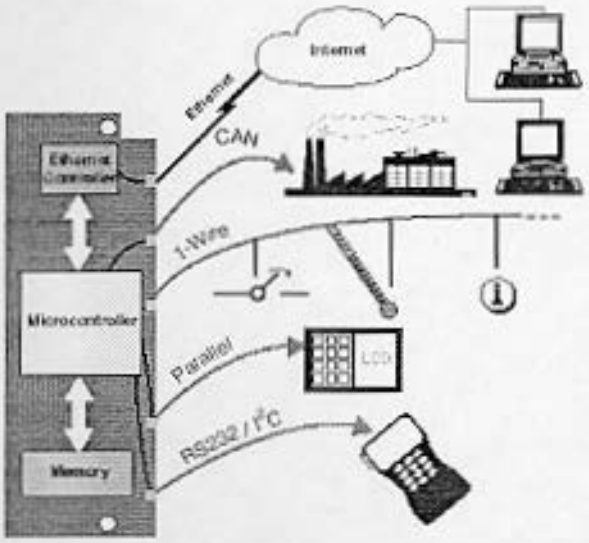


Figura 6: A Tini permite conversão de protocolos

6.2. O HARDWARE DA TINI

No mínimo cada implementação de *hardware* TINI contém os seguintes chips de larga escala de integração:

- Microcontrolador
- ROM Flash
- RAM Estática

O coração do *chipset* é o microcontrolador DS80C390. O DS80C390 integra suporte para diversas formas distintas de I/O incluindo serial, 1-Wire® e barramento Controller Area Network (CAN). Ele também oferece diversas portas de uso geral que podem ser usadas para realizar tarefas simples como acionar relays e LEDs. Entre algumas características do 80C390:

- Dois controladores CAN 2.0B completamente funcionais
- Identificador padrão de 11 bits ou formato estendido de 29 bits
- 15 centros de mensagens por controlador
- Filtro de mensagens suporta DeviceNet, SDS, e outros protocolos CAN estendidos
- Arquitetura de memória aprimorada:
  Endereça até 4MB de memória externa de programa e 4 MB memória externa de dados
  Endereçamento de programas/dados de 24 bits habilitado pelo usuário
  Modos de endereçamento paginados/ contíguos de 16 bits/ 24 bits
  Interface de memória multiplexada ou não-multiplexada selecionável pelo usuário
  Apontador de pilha de 10 bits opcional
- Coprocessador aritmético de alta velocidade
  Operações matemáticas de 16/32 bits

Multiplica, divide, desloca e normaliza
Contém um acumulador de 40 bits 4kB SRAM interna

- Arquitetura de alta velocidade:
  4 ciclos de relógio por ciclo de máquina (8051 = 12)
  Freqüência do clock vai de DC a 40MHz
  Multiplicador de freqüências reduz EMI e facilita a seleção de cristal
- Duas portas seriais full-duplex
- Integração em alto nível:
  Reset em caso de falha na alimentação
  Temporizador watchdog programável
  6 fontes externas de interrupção

6.3. O AMBIENTE DA TINI

O ambiente de execução da TINI está armazenado na memória flash, o que permite que o sistema mantenha o código na ausência de alimentação. Isto também permite que o ambiente de execução seja atualizado quando necessário. A RAM é empregada para dados do sistema bem como armazenagem de arquivos. Um não-volatilizador de SRAM pode ser adicionado opcionalmente ao chipset para permitir que os conteúdos da SRAM persistam na ausência da alimentação principal.

Com o I/O altamente integrado da TINI e a facilidade de programação, o ciclo de desenvolvimento de produtos pode ser encurtado dramaticamente.

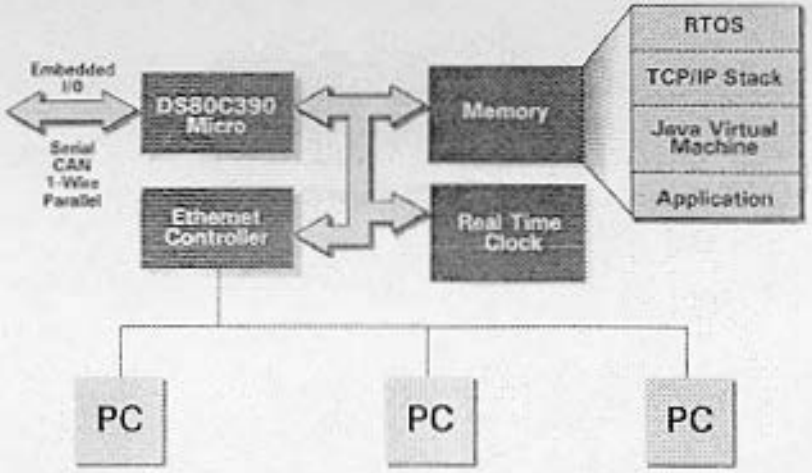


Figura 7: O ambiente da TINI

7. O Projeto

Um robô é um sistema mecânico que tem de ser controlado para realizar uma determinada tarefa. Esta tarefa envolve o movimento do braço manipulador, e portanto a função primária do sistema de controle do robô é posicionar e orientar o braço com determinada velocidade e precisão.

7.1. PLACA TINI

Após a introdução geral sobre a plataforma TINI, a atenção volta-se agora a detalhes mais especificos do *hardware* e sua implementação.

7.2. ALIMENTACAO

Na folha 1 do conjunto de esquemáticos encontramos a TINI, representada pelo conector SIMM-72, e algumas conexões importantes. A alimentação foi feita utilizando-se reguladores da série 78XX, componentes de emprego comum, para adequar a tensão de alimentação estabelecida como 24V devido a ser o valor de alimentação dos motores DC utilizados para tracionamento das juntas de robôs.

7.3. INTERFACES ETHERNET E SERIAL

A TINI já apresenta em sua placa um controlador Ethernet ( ) sendo então necessário penas que se faça a ligação devida dos terminais da placa aos de um conector padrão RJ-45. Da mesma forma, a placa também faz a adequação dos valores de tensão necessários ao protocolo RS232, bastando então levar os pinos corretos ao conector DB9 padrão deste tipo de comunicação.

7.4. INTERFACE CAN

Comum a todos os controladores CAN é que todos eles precisam ser conectados ao meio fisico através de uma adequação. Em se fazendo certas exigências quanto a proteção contra interferência, contra curto-circuito e outros, bem como comprimento do cabeamento ou número de nodos, será necessário que se utilize comutação externa adicional. Para evitar a montagem de acoplamentos discretos, desde cedo se utilizou os *transceivers* já disponiveis.

Por este motivo encontram-se em muitas aplicações CAN por exemplo *transceivers* utilizados em transmissão serial de dados segundo o padrão RS-485. Tais componentes são muito econômicos devido a sua vasta utilização e oferecem propriedades que são diferenciais numa transmissão de dados segura. Infelizmente estes são apropriados para utilização em sistemas de barramento CAN em alguns poucos casos.

Esta dificuldade foi identificada pelos fabricantes de semicondutores e hoje existem alguns *transceivers* que seguem a especificação ISO/DIS 11898, como por exemplo:

- Philips PCA 82C250
- Siliconix Si9200
- Bosch CF15

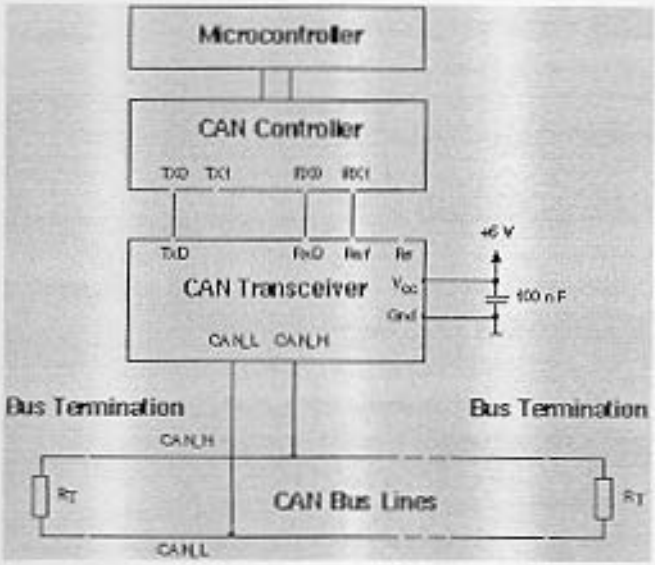


Figura 8: Conexão fisica do CAN de acordo com a norma ISO 11898

Na folha 2 do conjunto de esquemáticos encontraremos o PCA 82C250 ligado a um conector DB 9 pinos seguindo o padrão CiA DS-102 conforme figura 8. Foram empregados optoacopladores para isolação galvânica do circuito já que a alimentação do barramento será por fonte externa.

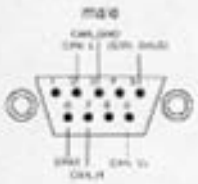


Figura 9: A pinagem sugerida pelo padrão CiA DS-102

7.5. MAPA DE MEMÓRIA

Na folha 3 do conjunto de esquemáticos encontramos os periféricos que estão mapeados como memória. Um mapa de memória especifica onde a memória e outros dispositivos periféricos são decodificados no espaço de endereçamento do microcontrolador. O mapa da TINI, mostrado na figura 7 abaixo, consiste dos seguintes segmentos distintos:

- Código
- Dados
- Periféricos

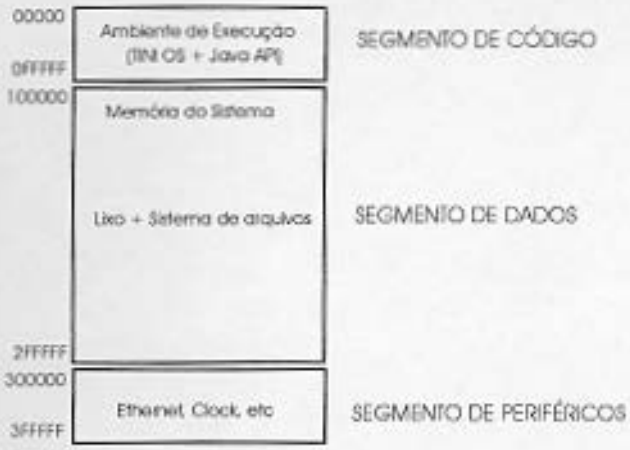


Figura 10: O mapa de memória da TINI

Os segmentos de código e de dados são ocupados por chips de memória, e o segmento periférico é ocupado por outros tipos de componentes de *hardware* como o controlador Ethernet e o Real-Time Clock. Existe ainda uma área periférica chamada espaço PCE (Peripheral Chip Enable) que pode ser usado para interfacear com memórias grandes (até 4x1Mb) ou outros dispositivos de *hardware* diretamente nos barramentos de endereços e dados. A opção foi por este espaço, porque desta forma não se faz necessário usar lógica extra para evitar colisão com osendereços do controlador ethernet e o real-time clock.

Para decodificar as posições de memória onde serão mapeados os periféricos a serem empregados foram utilizados os bits mais significativos do barramento de endereços, além do pino de habilitação PCE0.

O equacionamento das temporizações envolvendo a placa TINI e os periféricos é facilitado pela versatilidade do controlador 80C390 ao oferecer um SFR responsável pela inserção de *wait states* nos ciclos de leitura e escrita de modo a alongá-los, permitindo interfacear com dispositivos lentos.

7.6. DECODIFICAÇÃO DOS PERIFÉRICOS

Toda a lógica de decodificação de periféricos e demais lógicas necessárias no circuito foram implementadas utilizando-se lógica programável com um chip GAL 22v10. A introdução de dispositivos de lógica programável (PLDs) foi um grande boom no campo do projeto de *hardware* digital, sendo a segunda geração de PLDs, os GAL (Generic Array Logic) particularmente apropriada ao projetista de *hardwares* em pequena escala. As GALs oferecem as seguintes benesses ao projetista:

- Flexibilidade - as GALs são dispositivos flexíveis que podem implementar tanto lógica combinacional (lógica AND, OR, NAND) como funções lógicas registradas (contadores, registradores de deslocamento, etc) no mesmo chip.
- Reposição - as GAL 16v8 e 20v8 podem repor diretamente mais de 20 das PALs (Programmable Array Logic - a primeira geração de PLD) cada.
- Economia de espaço - casa GAL repõe entre 2 e 4 chips TTL, economizando bastante espaço em placa.

- Velocidade - as GALs são dispositivos rápidos podendo ter atrasos de propagação de aproximadamente 7ns. As típicas tem delay de aproximadamente 15ns - mais rápidas que a série 7400 e 74LS.
- Reprogramabilidade - não só as GALs são reprogramáveis, possibilitando corrigir erros de projeto mais facilmente e um roteamento de placa mais efetivo, como também podem ser reprogramadas até 100 vezes. Apagamento e gravação levam apenas alguns segundos.
- Custo - além da economia propiciada já na PCB, as GALs custam apenas uns poucos dólares mesmo em muito pequenas quantidades.

Todos os pinos apresentam características combinatoriais exceto CSLATCH. O sensor de fim de curso (REFPOS) seta um bit o freio (BREAK) é ativado por outro. Como estes valores precisam ser registrados - e não se desejava adição de periféricos extra - foi então atribuído a cada um destes sinais um bit do barramento de dados (D0 para o sensor e D1 para o freio) e utilizado o flip-flop da célula de programação para registrar D1.

Desta forma a lógica decodifica um endereço de memória onde o bit menos significativo contém o valor do sensor e o bit seguinte o valor para o freio, sendo que este endereço ativa o pino CSLATCH, e este leva então o sinal ao pino de CLK (estão em conexão física como se pode inferior no esquemático do projeto) ocasionando a atualização do valor no flip-flop.

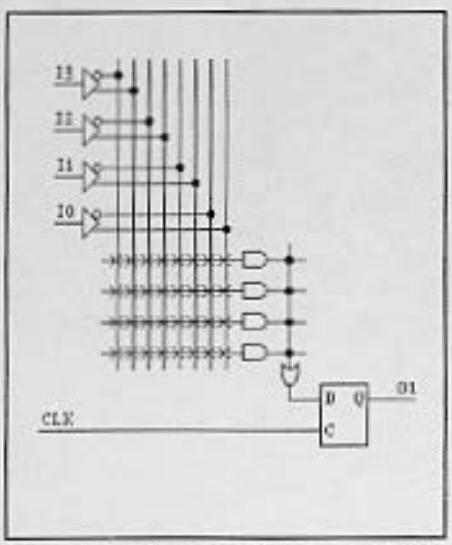


Figura 11: Macrocélula de um PLD

7.7. ENCODER

No Janus o sensoriamento é feito através de *encoders* incrementais, que fornecem uma informação relativa. Conforme vai ocorrendo uma rotação ou translação linear, o *encoder* incremental envia um pulso para cada intervalo de distancia percorrida. Estes pulsos podem ser contados para determinar a posição linear ou rotatória relativa a outra posição. O movimento é quantificado por um certo numero de pulsos. Usualmente o *encoder* incremental vem com três canais nomeados A, B e Z. Tipicamente o sinal de saída é como mostrado a seguir:

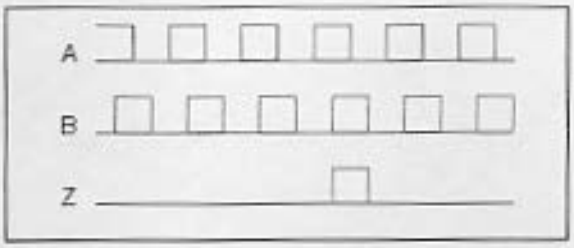


Figura 12: sinais nos canais do *encoder*

A e B estão posicionados em defasamento de 90 graus. Com estes dois canais, o processador determina a distância percorrida pelo número de passos, e a direção pela forma de onda. O terceiro canal é a referência. Usualmente o canal Z terá apenas um único pulso por rotação ou por comprimento do *encoder*, de forma a poder ser utilizado para determinar a localização atual do que apenas um numero incremental. Tais *encoders* podem ser magnéticos, ópticos, de contato ou capacitivos. Podemos encontrá-los em qualquer mouse de computador. A desvantagem do *encoder* incremental é que ele é incapaz de determinar sua localização logo depois do *start-up*, mas este problema pode ser solucionado dando tempo para fazer uma ida ao início do curso ou uma seqüência de pulso de referencia, e então movendo a quantidade de passos desejada a partir de então.

Figura 13: Um *encoder* incremental

Observe-se que os dois canais defasados de 90 graus entre si permitem ainda uma resolução 4 vezes maior, pois podemos ter AB igual a 00, 01, 10 e 11.

Existem integrados prontos para esta função, que não sobrecarregam a CPU com o processamento (crítico em aplicações de tempo real). A Hewlett-Packard oferece a linha HCTL (HCTL-2000, HCTL-2016 e HCTL-2020 e a LSI os 7083, 7084, 7166 e 7266 [7].

7.8. PWM

Temporização e contagem são dois requisitos básicos em sistemas de controle, e existem muitas soluções diferentes embora tipicamente estejam entre *software* ou *hardware*.

Na eletrônica moderna o rendimento com pequenas perdas e a ausência de grandes dissipadores que ocupem espaço é fundamental, principalmente quando circuitos de alta potência estão sendo controlados. Desta forma este tipo de controle de potência não é conveniente, sendo

requisitadas outras configurações de maior rendimento como as que fazem uso da tecnologia PWM.

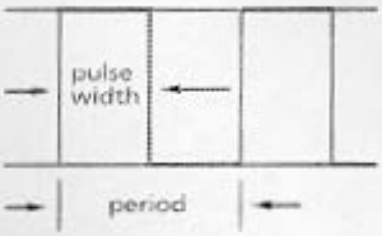


Figura 14: Esquema do PWM

PWM é abreviação de Pulse Width Modulation, ou modulação por largura de pulsos, e consiste em se variar o valor médio da tensão - e por conseguinte da potência - através do tempo de duração de cada pulso, já que pela definição o valor médio é a integral do valor instantâneo da função no intervalo de tempo correspondente.

Os dispositivos usados no controle não são capazes de abrir e fechar o circuito num tempo pequeno Eles precisam de um tempo para mudar de estado e, neste intervalo de tempo, sua resistência sobe de um valor muito pequeno até infinito e vice-versa, de forma que neste intervalo a queda de tensão e a corrente através do dispositivo não são nulas, e uma boa quantidade de calor poderá ser gerada de acordo com a carga controlada.

Mais uma vez procurando distribuir funções de forma a não sobrecarregar a CPU, a opção foi pela utilização de um integrado dedicado a esta função, mapeado como memória como os demais periféricos.

O 8254 foi projetado originalmente pela Intel como uma solução one-chip para muitos problemas de temporização e contagem. Tem contadores de 16-bits com seis modos programáveis, os incrementa on-chip usando pulsos, podendo ser estes quaisquer pulsos (transição 0 para +5V). Desta forma, além de contador ele também pode ser utilizado como timer, porque se aplicando um sinal de incremento com freqüência (período) fixo e determinado, a cada instante podemos verificar o número de contagens e multiplicando este valor pelo período do sinal de incremento, obter o tempo decorrido. Finalmente cada contador tem um pino de porta (*gate*) que permite o início ou término instantâneo de cada contagem, permitindo assim a geração de sinais PWM com qualquer ciclo de condução que se queira.

O 82C54 foi então utilizado para gerar o sinal PWM para acionamento dos motores e também como relógio do sistema, gerando uma interrupção na CPU após um intervalo de tempo programado, já que o real-time clock onboard na TINI apresenta como menor intervalo de tempo o segundo, que é tempo muito grande para o PWM de acionamento do motor.

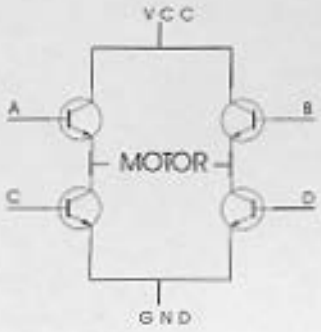
Na folha quatro do conjunto de esquemáticos encontramos a etapa de acionamento dos motores, composta da ponte H de MOSFETs, do driver encarregado deles, a etapa de frenagem do motor, e do divisor resistivo de tensão para o sensor de fim de curso. Este divisor foi calculado de modo a fazer uma relação de modo a dividir os 24V em apenas 3V, suficientes para indicar nível

lógico alto e mesmo assim tendo ainda uma margem para caso a tensão vinda do sensor ultrapassar a prevista e continuar dentro dos 5V toleráveis pelos circuitos digitais.

7.9. PONTE H

Uma ponte H é um circuito eletrônico que alimenta um motor DC em sentido direto, reverso, parada, e ainda permite modulação por largura de pulsos (PWM) de forma que o motor possa ser desacelerado pela variação do valor médio da tensão aplicada como função da largura do pulso.

Para acionar um motor, bastaria um único transistor; mas em se querendo rotação direta e reversa, faz-se necessária uma ponte H:



A	B	C	D	Função
1	0	0	1	Sentido direto
0	1	1	0	Sentido reverso
1	1	0	0	Freio
1	0	1	0	Estado proibido
0	1	0	1	Estado proibido

Figura 15: Esquema da Ponte H

Diodos podem ser inseridos entre emissor e coletor para proteger contra a tensão gerada pela bobina dos motores quando a alimentação é ligada e desligada, podendo esta tensão ser muitas vezes maior que a alimentação do circuito.

MOSFETs são preferíveis, pois podem trabalhar com correntes bem maiores e não aquecer tanto, além de já terem os diodos de proteção encapsulados.

O LT1162 é um integrado que vai servir de *driver* para os MOSFETS, recebendo o sinal PWM num de seus terminais e comutando os transistores de forma adequada, o que inclui oferecer toda a etapa de proteção contra inversões no acionamento configurando a ponte de forma a queimá-la. Isso facilita bastante o projeto do sistema de controle do motor.

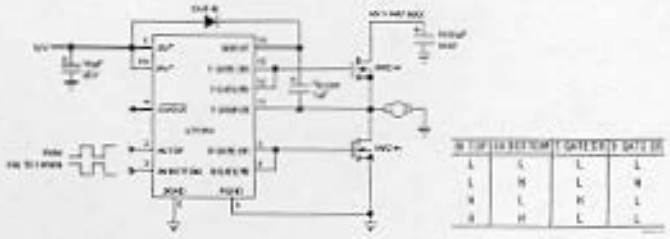


Figura 16: Esquema básico de ligação do LT1162

7.10. A PCB

A placa de circuito impresso foi desenvolvida utilizando a ferramenta Orcad, que constitui um pacote muito interessante de *softwares* independentes porém integrados, como editor de esquemáticos, programador de lógicas configuráveis, editor de *layout* de PCI, com o qual foi feita toda a parte de documentação do projeto elétrico, desde os esquemáticos até o roteamento da placa e geração dos arquivos de confecção desta (Gerbers).

Para se encaixar a placa no gabinete existente foi utilizado o modelo de placa tamanho Eurocard 160x120mm2. Os conectores foram dispostos apenas de modo a facilitar os testes do primeiro protótipo, pois se almeja levar a maioria dos conectores para um *blackplane* interfaceado nos 3x33 pinos do conector da placa eurocard.

No anexo II podemos ver os *layouts* das camadas da placa desenvolvida. As trilhas foram dimensionadas de forma a assegurar a corrente necessária sem risco de fusão ou descolagem destas, sendo atribuído às da etapa de potência uma largura de 50 mils e às demais trilhas em média 12,5 mils.

Após orçamentos em algumas empresas, enviou-se a placa para confecção mas esta não ficou pronta a tempo de inclui-la no material do presente relatório. A empresa não dispunha de software que abra o formato Orcad ou o industrial padrão gerber e desta forma foi enviada a documentação para geração dos fotolitos em formato AutoCAD.

Erros de escala por parte da empresa fizeram com que as primeiras placas não servissem e foram reenviadas para confecção no fechamento do presente relatório.

7.11. O FIRMWARE

A linguagem Java tem encontrado enorme difusão atualmente em sistemas em rede pela sua independência de plataforma. Na linguagem Java são criados *Byte Codes* imaginando-se uma máquina virtual universal, e então algoritmos específicos para cada ambiente adequarão as instruções codificadas nestes *byte codes* em instruções condizentes com o seu sistema operacional local. Desta forma, o programa compilado em *byte codes* numa máquina Linux poderá rodar numa máquina com sistema operacional Windows desde que se tenha instalado o software que faça a correta adequação dos *byte codes* no Windows.

Porém isso não é próprio para sistemas críticos com os de tempo real, fazendo com que se necessite deixar de lado todo o ambiente oferecido pela Sun Microsystems e pela Dallas para o desenvolvimento de aplicações na TINI e se parta para uma implementação do firmware usando linguagens mais otimizadas como Assembly e C.

O controlador 80C390 é um derivado da família 8051 cujo conjunto de instruções básico é idêntico ao dois demais controladores pertencentes à mesma família. Ele oferece inúmeras funcionalidades por acesso a seus registradores a um preço muito vantajoso, tornando-o desta forma economicamente bastante atrativo.

Em contrapartida são poucos os ambientes de desenvolvimento para este controlador, e seus custos são bastante elevados. O ambiente mais atraente é o MicroVision desenvolvido pela Keil Software. Este é um dos mais difundidos softwares para programação de microcontroladores 8051, havendo grande disponibilidade de referencias bibliográficas (sobretudo em páginas da internet), podendo-se obter até mesmo uma cópia de experimentação perfeitamente funcional diretamente na página da Keil Software (www.keil.com) com limitação do tamanho do código em 2Kb.

Todavia as funcionalidades do 80C390 - através das bibliotecas referentes a ele - não estão oferecidas, como por exemplo programação dos registradores CAN, programação dos registradores de alargamento da temporização dos sinais de *strobe* e comando e outras. Para se ter estas bibliotecas faz-se necessário obter uma licença do software mais o adicional das bibliotecas de suporte ao 80C390.

Após extensa pesquisa na internet foi encontrada uma versão aberta e em desenvolvimento do compilador SDCC que é *freeware* (GPL General Public License) - e que por hora oferece suporte aos controladores Intel 8051 e Zilog Z80 e compatíveis, estando em andamento projeto de estender o suporte à família Atmel AVR e Microchip PIC. O SDCC é oferecido para DOS, Windows e Linus PCC e 86 e permite programação *inline* em assembler ou seja permite chamadas a rotinas assembler que podem ser desenvolvidas diratamente dentro do programa em C.

Este foi o software empregado para programar o firmware da TINI.

8. Conclusão

O mercado de eletrônica embarcada (embedded), que compreende muitos dispositivos diferentes - de simples termostatos a sistemas de aviação super sofisticados - cresce muito rapidamente. A pressão pela disponibilização de sistemas para atender necessidades de mercado no prazo mais rápido possível deixa pouco espaço para erros no processo de manufatura e diferentemente das aplicações para PCs, as aplicações embarcadas não podem de forma alguma falhar, pois poderiam tais sistemas estão na maioria das vezes relacionados a controles de sistemas perigosos ou críticos.

Uma das maiores conclusões ao longo deste período de estágio foi a da importância de um ambiente de desenvolvimento no processo de projeto de um sistema. Durante o curso de graduação somos apresentados a diversas ferramentas de software no auxilio do projeto para algumas disciplinas, mas estas são uma única classe: simuladores para confirmação do comportamento elétrico do projeto.

Todavia não somos apresentados a softwares de documentação eletrônica e ambientes completos de desenvolvimento como os disponíveis no mercado e adotados largamente em empresas e instituições que desenvolvem equipamentos e dispositivos, os quais permitem além desta etapa de simulação e conferência de erros, também a documentação dos esquemáticos, o desenvolvimento de placas de circuito impresso, a programação de dispositivos de lógica configurável e com uma visão gerencial do processo, permitindo por exemplo que determinados integrantes do grupo de desenvolvimento possam trabalhar em rede e ter liberdade total ou parcial programável de fazer modificações no circuito.

O projeto desenvolvido durante o período de estágio foi enormemente ajudado pelas facilidades de contar com um software como o referido, e seu andamento ficou a contento, tendo sido limitado apenas por eventos externos como serviços terceirizados na confecção das placas e demora na entrega das placas TINI por parte do fabricante.

Os trabalhos deverão ter continuidade durante o curso de pós-graduação em engenharia elétrica atuando no laboratório de controle, automação e robótica.

9. Referências

1. Site da Dallas Semicondutores. Introducing TINI: Tini internet interface. Http://www.ibutton.com/TINI/index.html

2. The Janus Project. GMD. Http://www.ais.gmd.de/as/janus/pages/janus.html

3. Elektor Electronics

4. Konrad Etschberger. "CAN – Controller Area Network. Grundlagen, Protokolle, Anwendungen, Bausteine"

5. Phillips Application Notes. Keil GmbH.

6. "CAN – Das Netzwerk für die Elektronik im Kraftfahrzeug". BOSCH. www.bosch.de/k

7. Quadrature Decoder/Counter Interface ICs. HCTL 2000. Technical Data. Agilent Technologies

8. Loomis, Don; "The TINI Specification and Developer's Guide". http://www.ibutton.com/TINI/tinispec.pdf

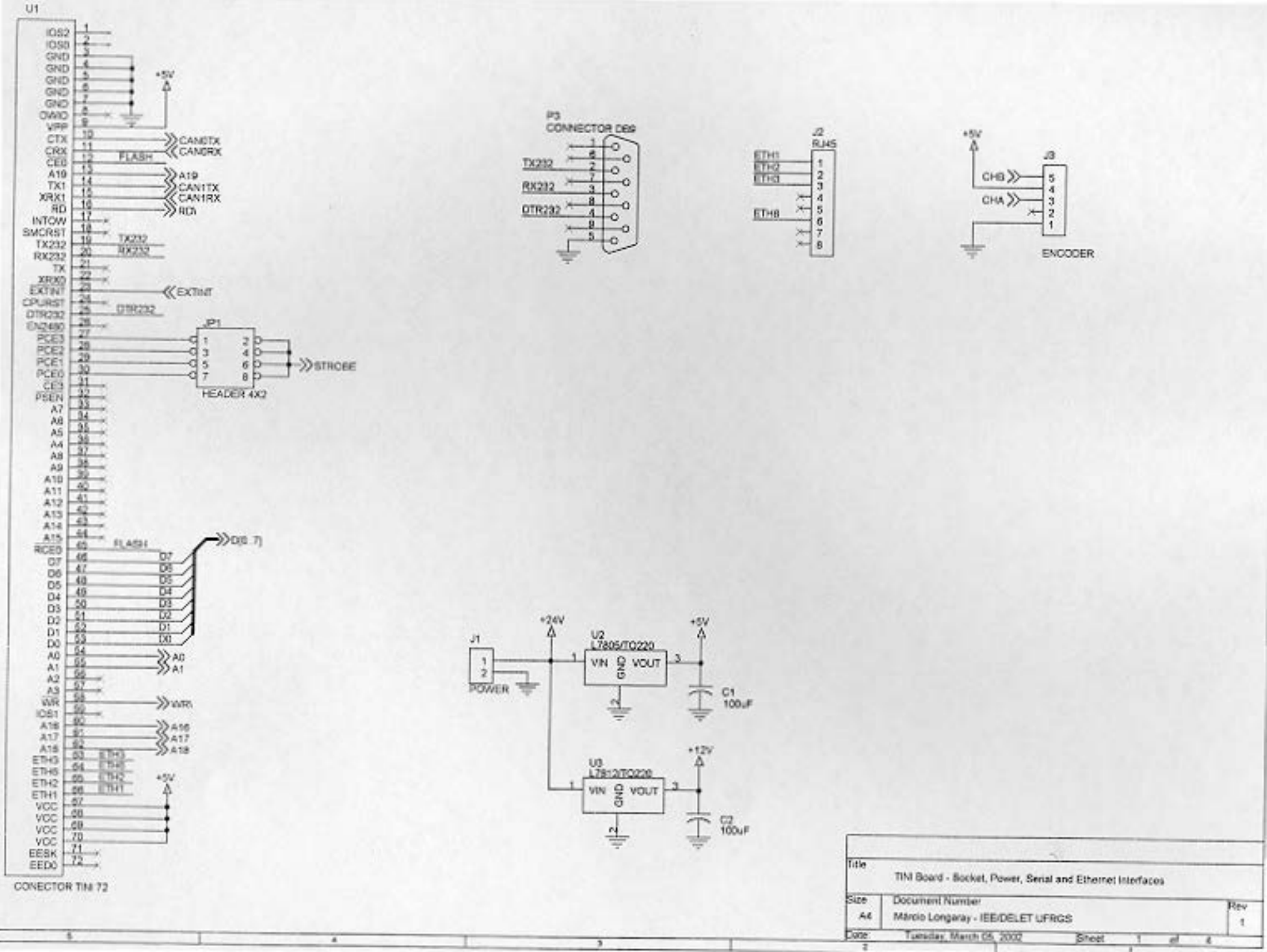
9. Kosow, Irving L.; Máquinas Elétricas e transformadores. 13 Edição. 1998. Editora Globo. ISBN 85-250-0230-5

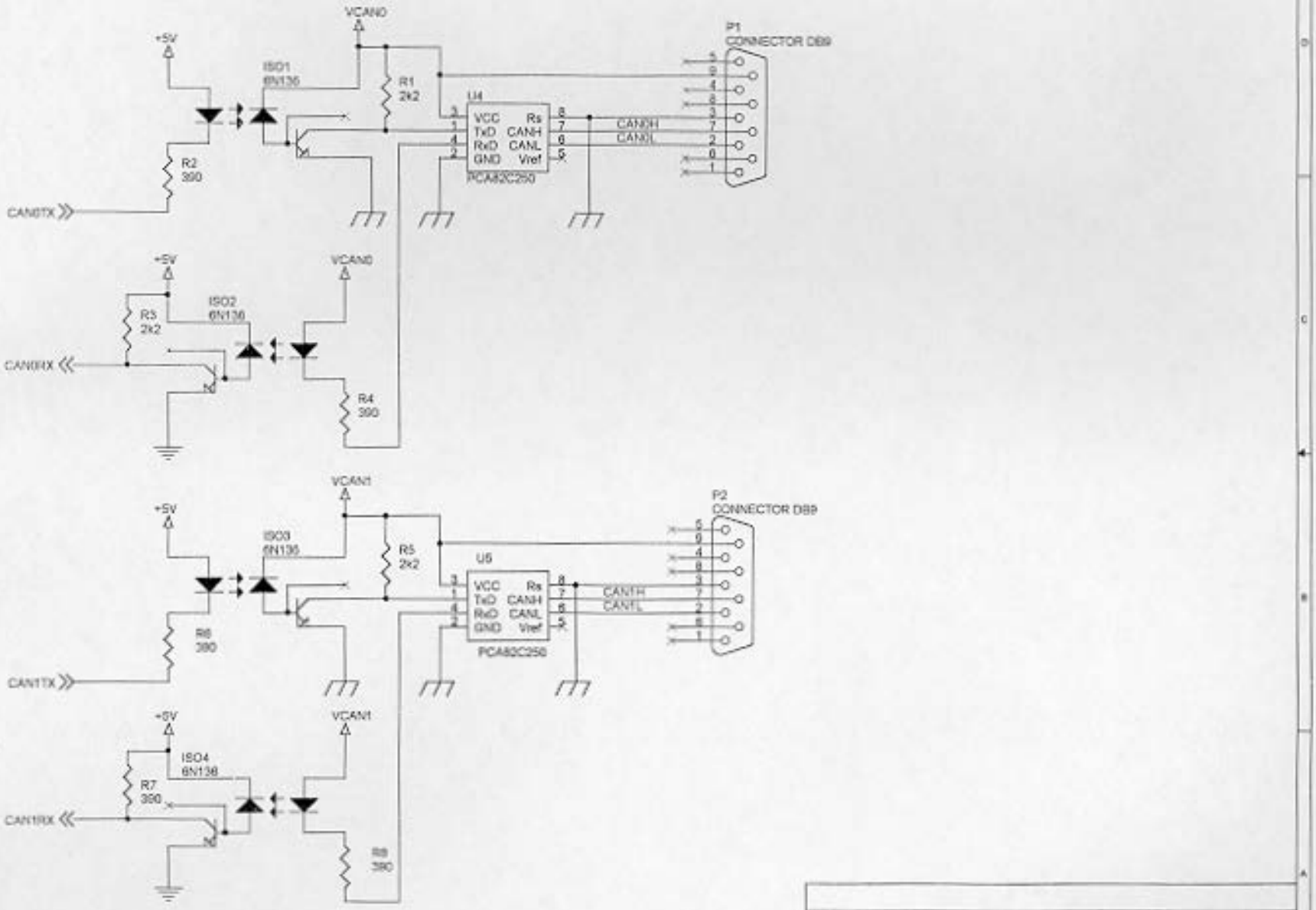
10. Cady, Fredrick M.; "Microcontrollers and Microcomputers - Principles of Software and Hardware Engineering". 1 Edição. 1997. Oxford University Press. ISBN 0 -19-511008-0

11. Lawrenz, Wolfhard. "CAN System Engineering: From Theory to practical applications". 1997. Springer-Verlag New York, Inc. ISBN 0-387-94939-9

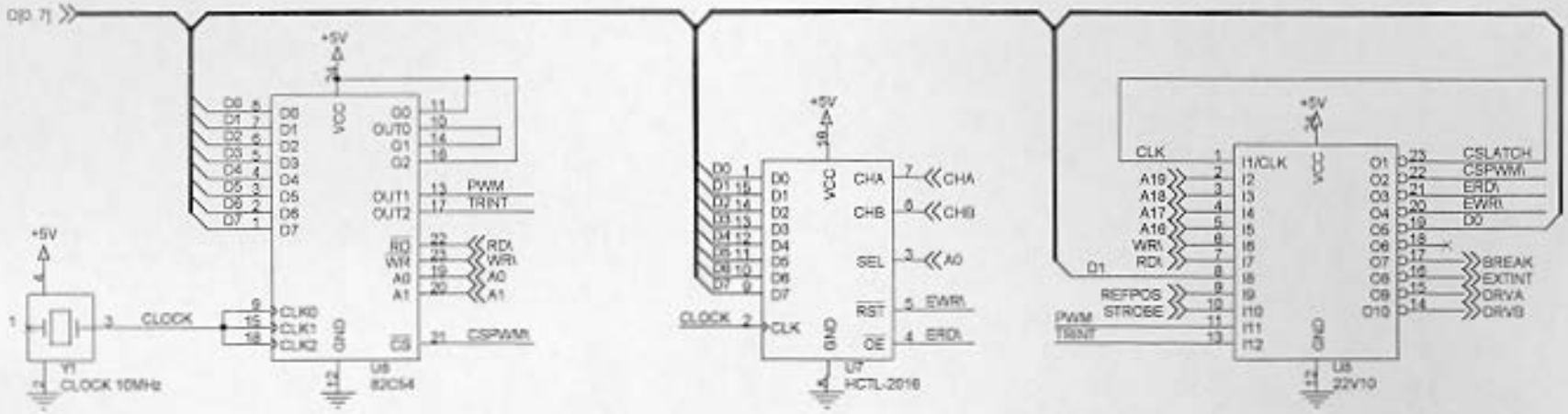
12. Etschberger, Konrad; "CAN- Controller Area Network. Grundladen, Protokolle, Anwendungen, Bausteine". Carl Hanser Verlag. ISBN3-446-17596-2

Anexo I - esquemáticos

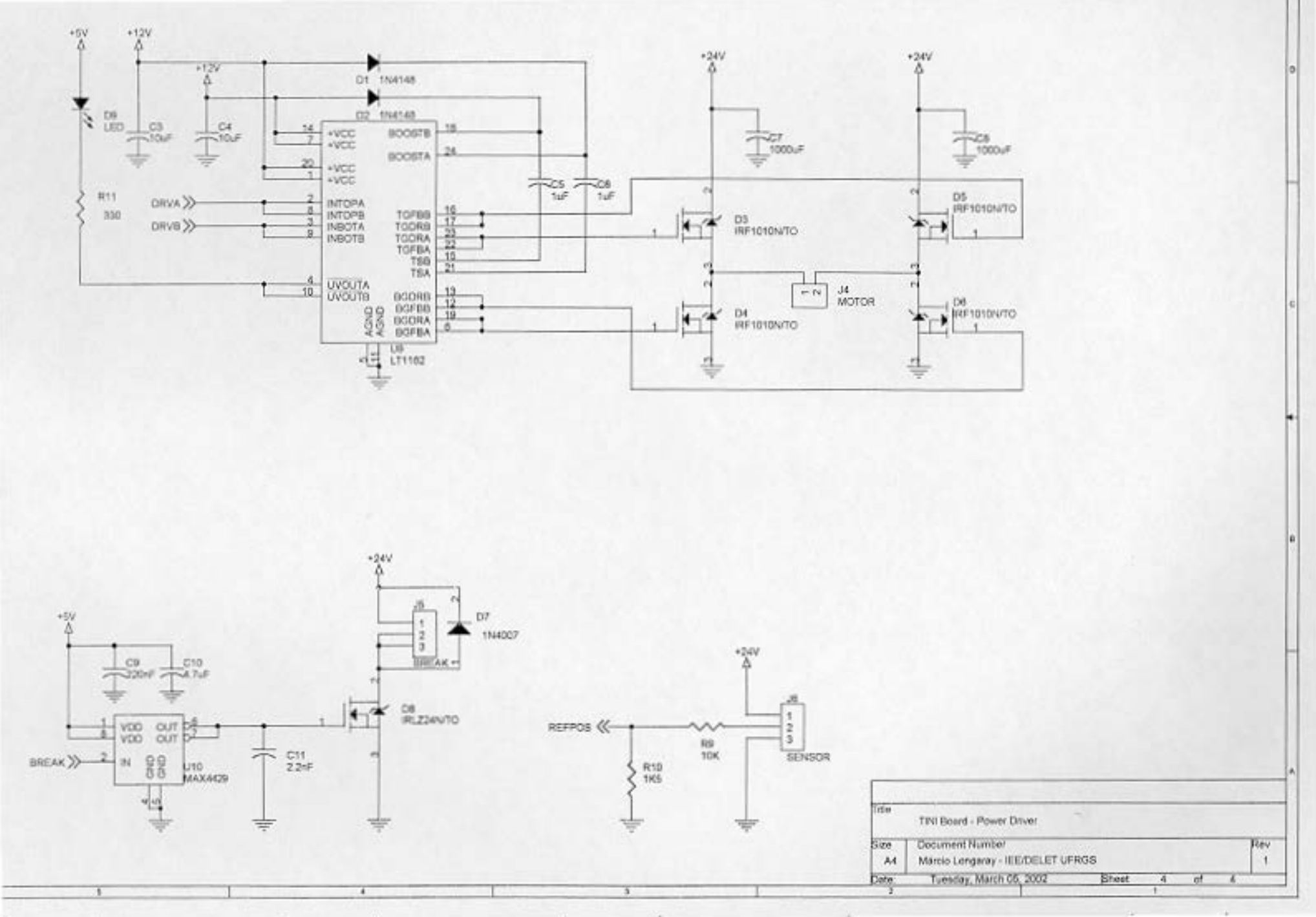

U1
IOS2
IOS0
GND
GND
GND
GND
GND
OWIO
VPP
CTX
CRX
CE0
A19
TX1
XRX1
RD
INTOW
SMCRST
TX232
RX232
TX
XRX0
EXTINT
CPURST
DTR232
ENZ480
PCE3
PCE2
PCE1
PCE0
CE3
PSEN
A7
A6
A5
A4
A8
A9
A10
A11
A12
A13
A14
A15
RCE0
D7
D6
D5
D4
D3
D2
D1
D0
A0
A1
A2
A3
WR
IOS1
A16
A17
A18
ETH3
ETH2
ETH1
VCC
VCC
VCC
VCC
EESK
EEDO
+5V
CAN0TX
CAN0RX
FLASH
A19
CAN1TX
CAN1RX
RD\
TX232
RX232
EXTint
DTR232
JP1
HEADER 4X2
STROBE
FLASH
D[0..7]
A0
A1
WR\
A16
A17
A18
ETH3
ETH2
ETH1
+5V
CONECTOR TINI 72
P3
CONNECTOR DB9
TX232
RX232
DTR232
J2
RJ45
ETH1
ETH2
ETH3
ETH8
+5V
J3
CHB
CHA
ENCODER
J1
POWER
+24V
U2
L7805/TO220
VIN
GND
VOUT
+5V
C1
100uF
U3
L7812/TO220
VIN
GND
VOUT
+12V
C2
100uF
Title
TINI Board - Socket, Power, Serial and Ethernet Interfaces
Size
A4
Document Number
Márcio Longaray - IEE/DELET UFRGS
Rev
1
Date
Tuesday, March 05, 2002
Sheet
1
of
4



Title	TINI Board - CAN Interface	
Size: A4	Document Number: Márcio Longaray - IEE/DELET UFRGS	Rev: 1
Date: Tuesday, March 05, 2002	Sheet 2 of 4	



Title	TN Board - Encoder, PWM and Glue Logic	
Size: A4	Document Number: Márcio Longaray - IEE/CELET UFRGS	Rev: 1
Date: Tuesday, March 05, 2002	Sheet: 3 of 4	


+5V
+12V
D1 1N4148
D2 1N4148
D9 LED
C3 10uF
C4 10uF
R11 330
DRVA
DRVB
+VCC
INTOPA
INTOPB
INBOTA
INBOTB
BOOSTB
BOOSTA
TGFBB
TGDRB
TGDRA
TGFBA
TSB
TSA
UVOUTA
UVOUTB
BGDRB
BGFBB
BGDRA
BGFBA
AGND
U9 LT1162
C5 1uF
C6 1uF
+24V
C7 1000uF
C8 1000uF
D3 IRF1010N/TO
D4 IRF1010N/TO
D5 IRF1010N/TO
D6 IRF1010N/TO
J4 MOTOR
D7 1N4007
J5 BREAK
D8 IRLZ24N/TO
C9 220nF
C10 4.7uF
C11 2.2nF
U10 MAX4429
VDD
OUT
IN
GND
BREAK
REFPOS
R9 10K
R10 1K5
J6 SENSOR
Title
TINI Board - Power Driver
Size A4
Document Number
Márcio Lengaray - IEE/DELET UFRGS
Rev 1
Date Tuesday, March 05, 2002
Sheet 4 of 4

Anexo II - layout da PCB