Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica

FABRÍCIO LUCAS DE ALMEIDA

Projeto e Implementação de uma rede AS-Interface

Uberlândia
2017

Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica

FABRÍCIO LUCAS DE ALMEIDA

Projeto e Implementação de uma rede AS-Interface

Trabalho apresentado como requisito parcial de avaliação na disciplina Trabalho de Conclusão de Curso do Curso de Engenharia Elétrica - Certificado de Estudos em Engenharia de Automação e Controle da Universidade Federal de Uberlândia.

Orientador: Dr. Marcelo Barros de Almeida

_____

Assinatura do Orientador

Uberlândia

2017

FABRÍCIO LUCAS DE ALMEIDA

Projeto e Implementação de uma rede AS-Interface

Trabalho apresentado como requisito parcial de avaliação na disciplina Trabalho de Conclusão de Curso do Curso de Engenharia Elétrica - Certificado de Estudos em Engenharia de Automação e Controle da Universidade Federal de Uberlândia.

Uberlândia, __ de _____ de ____

BANCA EXAMINADORA

_____

Prof. Dr. Marcelo Barros de Almeida
Universidade Federal de Uberlândia

_____

Prof. Me. Renato Santos Carrijo
Universidade Federal de Uberlândia

_____

Prof. Dr. Márcio José da Cunha
Universidade Federal de Uberlândia

AGRADECIMENTOS

Agradeço aos meus pais, Edson e Cleonice, e meu irmão Fábio, pelo cuidado dedicado a mim desde sempre e pelos valores morais compartilhados que foram base fundamental para a minha formação pessoal. Agradeço a minha namorada, Franciely, pelo companheirismo, paciência e constante motivação que foram de suma importância para concluir esta etapa da vida.

Aos meus professores que generosamente dividiram seu conhecimento, com estes aprendi a importância da educação na transformação de um aluno desde o início da faculdade até a formação, saindo como um engenheiro sedento por conhecimento.

Agradeço ao professor Dr. Marcelo Barros pela pré-disposição e auxílio na contribuição em todas as fases do projeto, auxiliando no planejamento, organização e desenvolvimento, além de sempre acreditar em minha capacidade, o que contribuiu bastante para a conclusão deste trabalho.

Agradeço também a todos os meus amigos e familiares que sempre acompanharam minha trajetória pessoal e profissional em busca do sucesso e satisfação em aprender e servir. A todos que torcem positivamente por mim, só tenho a agradecer pela força que me motiva a continuar sempre em frente.

RESUMO

O protocolo industrial AS-Interface é mundialmente conhecido e utilizado em certas aplicações devido a suas diversas vantagens como baixo custo, velocidade de comunicação e o fato de ser determinístico. Este trabalho apresenta uma proposta para construção de todo o firmware e eletrônica, implementando dispositivos (mestre e escravo), que se comuniquem através do protocolo AS-Interface. A metodologia proposta, baseia-se no projeto e desenvolvimento da PCB do escravo, no projeto e implementação de todo o firmware do mestre e no desenvolvimento da interface do usuário. É importante dar ênfase, que todo o projeto será implementado desde a origem, incluindo a definição de requisitos e escolha de componentes eletrônicos, até o final de todo o desenvolvimento, seguido de testes de comunicação na rede AS-Interface.

Palavras Chave: AS-Interface, protocolo, rede, mestre, escravo

ABSTRACT

The industrial protocol AS-interface is known and used worldwide in certain applications because of its many advantages as low cost, communication speed and the fact of being deterministic. In this paper is shown a development proposal of building firmware and electronic, implementing devices (master and slave) that communicate using the protocol AS-Interface. The methodology proposed is based in the project and development of the PCB of the slave, in the project and development of the entire firmware of the master and development of the user interface. It is important to emphasize, the entire project will be implemented from the beginning, including the definition of requirements and chose electrical components, until the end of the development, following communication tests in AS-Interface network.

Keywords: AS-Interface, protocol, network, master, slave

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

APM – Alternative Pulse Modulation

AS-Interface – Actuator Sensor Interface

BOM – Bill of Materials

CI – Circuito Integrado

CLP – Controlador Lógico Programável

CMSIS – Cortex® microcontroller software interface standard

CRC – Cyclic Redundancy Check

DC- Direct Current

DI – Digital Input

DO – Digital Output

EEPROM – Electrically Erasable Programmable Read-Only Memory

GND - Ground

GPIO – General Purpose Input / Output

HAL – Hardware Abstraction Layer

ID – Identification

IHM – Interface Homem Maquina

I/O – Input/Output

OSI - Open Systems Interconnection

PCB – Printed Circuit Board

PCI – Peripheral Component Interconnect

PTH – Pin Through Hole

SMD – Surface Mount Device

UART – Universal Asynchronous Receiver / Transmitter

USB – Universal Serial Bus

SUMÁRIO

1 INTRODUÇÃO

Quando se fala de protocolos industriais, a primeira coisa que é preciso entender bem são os níveis de automação industrial. De acordo com SMAR (2012), existem 5 níveis hierárquicos dentro da indústria. No nível 5 tem-se a parte de gerenciamento corporativo (gestão, finanças), mais abaixo, no nível 4, está o gerenciamento da planta onde são realizados cadastros de produção, agendamentos e logística. Já no nível 3 está a supervisão, incluindo supervisórios para acompanhamento do processo, banco de dados e/ IHM. O nível 2 é composto por todos os equipamentos que executam o controle automático centralizado da planta, representado pelo PLC e, finalmente o nível 1 de campo, composto por vários equipamentos incluindo sensores e atuadores da planta. É justamente entre o nível 1 e 2 que se aplica o protocolo AS-Interface.

O protocolo AS-Interface foi desenvolvido em conjunto com 11 empresas entre 1990 e 1994. Neste contexto já existiam diversos protocolos, mas nenhum deles forte o bastante para conexão de sensores e atuadores simples, dessa necessidade surgiu o novo protocolo de comunicação. O protocolo tem números expressivos ao redor do mundo com mais de 27 milhões de nós instalados, mais de 350 membros associados e número superior a 1800 produtos certificados pela norma (ZURAWSKI, 2015).

Dentre todas as tecnologias industriais existentes nos dias de hoje, os protocolos de comunicação, sem dúvida, foram os que tiveram maiores evoluções na última década pois, cada vez mais a rede porta dados mais robustos e mais informações acerca do meio (SCHNEIDER ELECTRIC, 2007).

O benefício da rede AS-Interface que mais se destaca é seu baixo custo. Segundo RTA (2014) tipicamente, os custos são cerca de 15 a 40% menores se comparados a outros sistemas tradicionais. O segundo ponto se destaca no que diz respeito a velocidade de comunicação do protocolo. Para uma rede com 31 escravos, o tempo máximo é de apenas 5 ms, sendo mais rápido que o ciclo da maior parte dos CLPs. Em terceiro e não menos importante, constitui um protocolo determinístico, onde a comunicação entre o mestre e os escravos são predefinidas em cada ciclo, resultando em uma grande precisão na comunicação. Existem diversos outros atrativos da rede AS-Interface como a facilidade de instalação e manutenção, incluindo a rápida configuração

da rede, topologia livre de rede, fornecimento da alimentação aos equipamentos e um enlace confiável para troca de dados em um mesmo cabo e tudo isso com proteção IP67.

O protocolo AS-Interface não pode ser comparado com outros como Profibus e DeviceNet, pois esses outros sistemas suportam uma comunicação mais genérica, tem custo mais elevado, são altamente configuráveis e possuem uma série de recursos para diagnósticos da rede, o que foge ao foco do protocolo AS-Interface.

1.1 OBJETIVO

Este trabalho objetiva projetar e desenvolver um produto capaz de comunicar através do protocolo industrial AS-Interface que, se destina ao controle, através do mestre, dos escravos da rede (sensores e/ou atuadores). A intenção é que esse sistema seja usado a princípio didaticamente, para ensino do protocolo em disciplinas de redes industriais, compondo assim, equipamentos com qualidade e segurança exigidas de um protocolo industrial.

Para que a comunicação do sistema seja possível é necessário projetar e implementar os seguintes itens:

- Interface entre o usuário e o mestre;
- Mestre AS-Interface;
- Escravo AS-Interface.

1.2 ESTRUTURA DO DOCUMENTO

Esse trabalho de Conclusão de Curso está estruturado em 6 capítulos, conforme descrição a seguir:

Capítulo 1 – Introdução: neste capítulo fez-se a apresentação do tema de pesquisa e o contexto no qual está inserido. Esta etapa teve a função de apresentar a importância e a relevância do tema que justifica a realização do trabalho, esclarecendo as vantagens e benefícios do protocolo AS-Interface.

Capítulo 2 – Revisão Bibliográfica: neste capítulo é realizado uma revisão da literatura sobre o protocolo AS-Interface, conceitos básicos e funcionamento em geral, além dos tópicos sobre o chip ASI4U e a placa de desenvolvimento

STM32F4DISCOVERY. Esta revisão da literatura estabelece uma base teórica para discutir os temas apresentados no discorrer do trabalho, fazendo uma síntese dos principais tópicos da rede.

Capítulo 3 – Projeto da rede AS-Interface: são definidas as informações necessárias para realizar a construção de uma rede AS-Interface, assim como também, a metodologia emprega para o desenvolvimento do projeto e softwares utilizados. É apresentado o projeto do mestre, como os métodos de comunicação, manchester, UART e outras definições, quanto ao escravo, são apresentados os requisitos construtivos e de funcionamento da PCB.

Capítulo 4 – Desenvolvimento da rede AS-Interface: neste capítulo é relatado toda a implementação da interface entre o usuário e o mestre em linguagem python, o desenvolvimento do projeto da PCB do escravo e seus aspectos construtivos e quanto ao mestre, foram explanados todos os assuntos quanto a lógica de funcionamento, fluxogramas, máquinas de estados, base de dados, tudo em prol do desenvolvimento do mestre em linguagem C.

Capítulo 5 – Testes e resultados da rede AS-Interface: são mostrados os resultados do funcionamento da rede, capturas através do osciloscópio dos frames de comunicação, comando e resposta, troca de dados e tempos de comunicação, englobando a camada física e o controle do protocolo. Também é discutido o funcionamento geral da rede (interface, mestre e escravo) com vários testes para troca de dados.

Capítulo 6 – Considerações finais: neste capítulo é verificado o atendimento aos requisitos e objetivos do projeto. Também são apresentadas as considerações sobre os resultados obtidos com a implementação da rede AS-Interface. E por fim são feitas algumas recomendações para trabalhos futuros no tema.

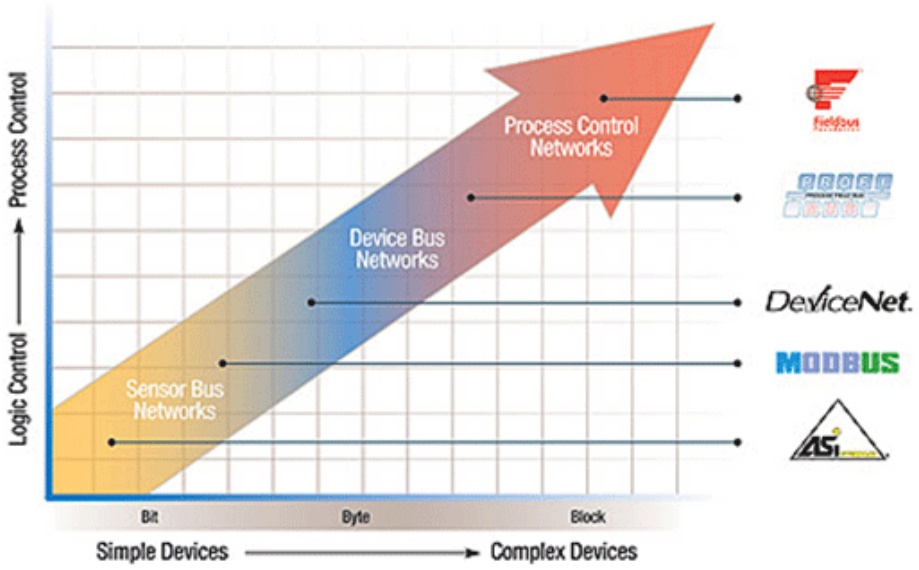
2 REVISÃO BIBLIOGRÁFICA

Este capítulo consiste na revisão literária acerca dos assuntos pertinentes ao projeto, englobando livros, normas, folha de dados e manuais, a fim de, fundamentar todos os aspectos abordados no projeto e implementação de uma rede AS-Interface.

2.1 PROTOCOLO AS-INTERFACE

O protocolo AS-Interface consiste numa interface de sensores e atuadores binários, destinado a dispositivos simples e de baixo nível, comunicando através de mensagens com poucos bits de dados (REYNDERS; MACKAY; WRIGHT, 2005). Tomando como base os níveis de automação industrial, tem-se a seguir, os protocolos entre o nível 1 e 2 mais usados, envolvendo dispositivos de campo e de controle, conforme se pode ver na figura 1.

Figura 1: Principais protocolos industriais e áreas de aplicação

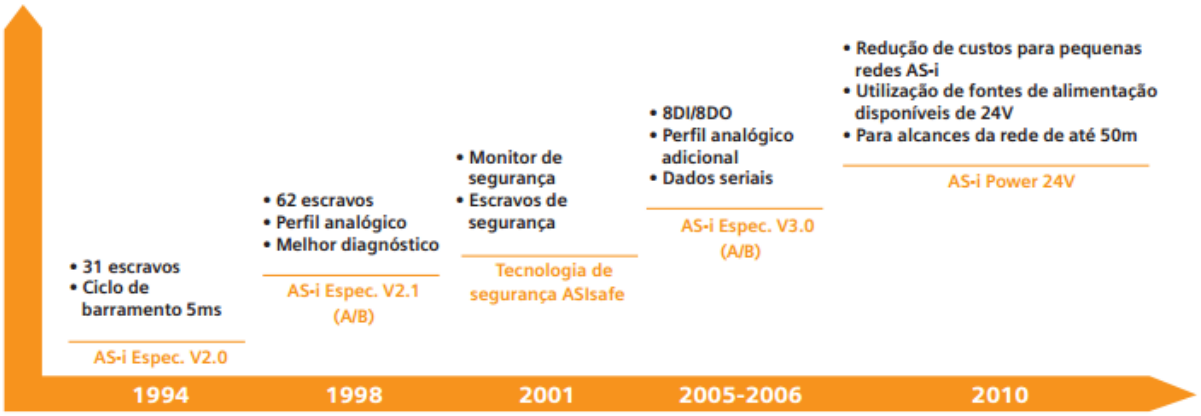


Fonte: ATAIDE, 2004.

Como já citado em SCHNEIDER ELECTRIC (2007), os protocolos de comunicação evoluíram e muito com o tempo. E com o AS-Interface não foi diferente pois, saindo de 1994 com apenas 31 dispositivos discretos ligados na rede para em 2006,

suportar 62 dispositivos conectados à rede, canais analógicos configuráveis e até 8DI/8DO. Na figura 2 se pode observar bem essa evolução.

Figura 2: Especificações AS-Interface



Fonte: SIEMENS, 2016.

Para exemplificar melhor essa evolução, observa-se na tabela 1 que, o número total de I/O passou de 248, desde a primeira especificação, até 992 na versão 3.0 (por mestre), onde cada escravo pode ter até 8 entradas e 8 saídas, porém o tempo de ciclo da rede também aumentou, saindo de 5 milissegundos para até 40 milissegundos.

Tabela 1: Características do protocolo para cada especificação

Versão de Especificação	Entradas	Saídas	Tempo de ciclo (ms)	N. de escravos digitais	N. de escravos analógicos	∑ I/O
2	4/4	4	5	31	31	248
2.1	4	3	10	62	31	434
3	4/8	4/8	20	62	62	992

Fonte: FESTO, 2016.

Após explicação da organização do frame do protocolo, posteriormente será explanado em detalhes como foi feita cada expansão, em relação ao número de entradas e saídas, para cada versão de especificação.

De acordo com AS-INTERNATIONAL ASSOCIATION (2002) o protocolo AS-Interface, pode ser categorizado com referência ao modelo OSI de sete camadas, porém

com a diferença que, o protocolo AS-Interface possui apenas três camadas implementadas que consistem em: aplicação, enlace e física.

Tabela 2: Referência entre o modelo OSI e o protocolo AS-Interface

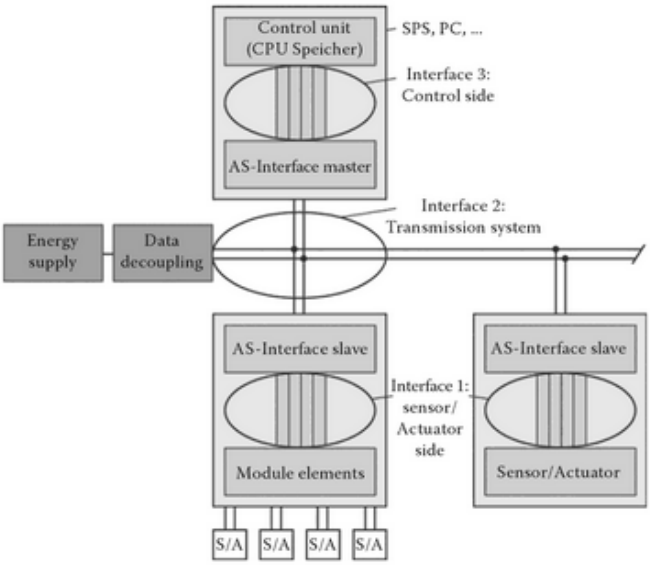
Camada ISO-OSI	Função	AS-Interface
7: Aplicação	Fornece o serviço de rede ao usuário	Mensagens, ciclo, perfis, endereçamento automático
6: Apresentação	Transformação de formatos de rede para usuário	
5: Sessão	Liga e desliga as conexões	
4: Transporte	Preparação do dado para a interface de transporte	
3: Rede	Endereços e rotas	
2: Enlace	Estrutura de dados, frame proteção e erros	Telegrama, bit de início, fim, paridade e erros
1: Física	Conexões elétricas para transmissão da informação	Cabo, alimentação, APM, desacoplamento de dados

Fonte: AS-INTERNATIONAL ASSOCIATION, 2002. TRADUÇÃO LIVRE.

2.1.1 Componentes e interfaces da rede

Antes de conhecer cada componente da rede, é preciso distinguir cada interface que ela possui. De acordo com a figura 3 existem 3 diferentes interfaces. A interface 1 é responsável por conectar o escravo com os sensores e atuadores no campo através de portas de entradas, saídas e parâmetros. Já a interface 2 consiste na conexão entre os pontos ASI+ e ASI-, responsável pela troca de informações entre o mestre e o escravo. A interface 3 é composta pelo acoplamento entre o controlador (CLP, PC) e o barramento de campo, através do mestre. Para essa finalidade, tipicamente o mestre é equipado com uma interface serial, que utiliza os padrões RS-232C/RS-485 (SVEDA e ZEZULKA, 2002).

Figura 3: Interfaces da rede AS-Interface



Fonte: ZURAWSKI 2015.

Segundo AS-INTERNATIONAL ASSOCIATION (2000) a rede AS-Interface é composta por 3 componentes principais, são eles:

- Escravo: qualquer unidade que possa ser acessada através do mestre para troca de dados, parâmetros ou monitoramento;
- Mestre: unidade que organiza e monitora a rede, realiza a troca de dados, parâmetros e comandos com os escravos;
- Fonte de Alimentação: unidade que fornece tensão DC para o sistema e possui um circuito de desacoplamento de dados. Faz parte da interface 2.

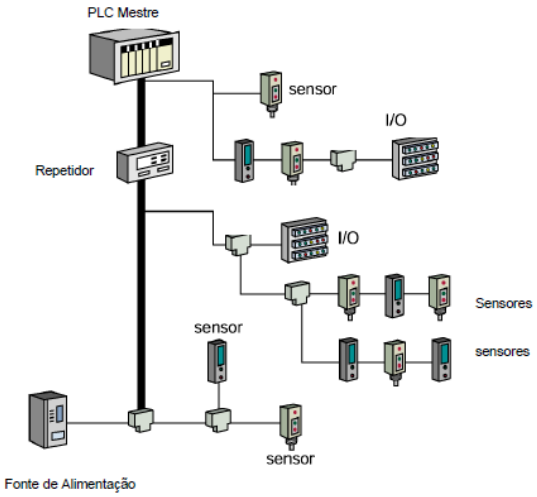
Existem outros componentes adicionais que podem vir a fazer parte da rede. Segundo SIEMENS (2006) é possível ter dispositivos adicionais na rede, como: repetidores, extensores, plugue extensor e outros acessórios.

2.1.2 Topologia

De acordo com AS- INTERNATIONAL ASSOCIATION (2000) a topologia empregada no protocolo é em árvore, mas permite várias outras configurações. O comprimento total da rede, incluindo a soma de todos as terminações de cabo, não deve exceder o limite de 100 metros de comprimento.

Se for necessário uma rede mais extensa, existem alternativas comerciais que permitem a rede ter até 600 metros. Na figura 4, observa-se o exemplo de uma rede AS-Interface em árvore, assim como os elementos que compõem a mesma.

Figura 4: Exemplo de topologia de uma rede AS-Interface



Fonte: SCHNEIDER ELECTRIC, 2007.

2.1.3 Endereçamento da rede

"Todos os escravos devem possuir um endereço (módulos, sensores, botoeiras e demais componentes com chip AS-i integrado), para que possam ser reconhecidos e colocados em funcionamento na rede AS-i" (SIEMENS, 2016, p. 8).

Diante disso, cada escravo deve ter um endereço único na rede, onde o mesmo pode ser atribuído através do próprio mestre ou com auxílio de um endereçador.

De acordo com AS- INTERNATIONAL ASSOCIATION (2000) os valores válidos de endereço a serem atribuídos a cada escravo estão em uma faixa de 1 à 31. A partir da versão 2.1 já existe o endereçamento estendido, onde os endereços válidos vão de 1A/1B até 31A/31B, contabilizando um total máximo de 62 escravos por enlace.

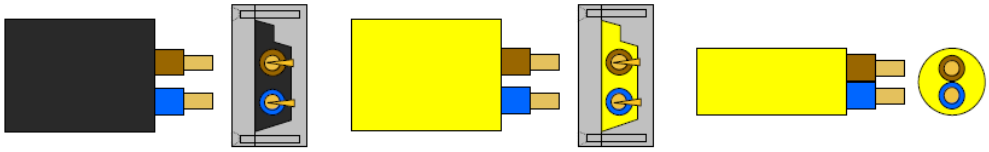
2.1.4 Características físicas

Primeiramente, vale ressaltar sobre o cabeamento utilizado na rede AS-Interface, que faz parte da interface 2 na camada física do protocolo. O cabo mais conhecido é

chamado de “*yellow flat cable*”, que possui 2 fios de 1.5 mm² (18 AWG) e um perfil especial, que elimina o risco de inversão de polaridade. Através desse cabo é possível levar tanto os dados quanto a alimentação simultaneamente para todos os dispositivos ligados à rede, o que economiza e muito os gastos no projeto com cabeamento. Porém em determinadas aplicações onde, os dispositivos necessitem de maior potência, é necessário levar separadamente um cabo de dados e um cabo para a alimentação, também chamado de “*black flat cable*”. Também existe o cabo no formato redondo, com as mesmas características elétricas do anterior. Na figura 5 é apresentado os aspectos construtivos e o perfil desses cabos.

A conexão do cabo geralmente é feita através de conectores do tipo vampiro que garantem uma boa conexão e, se porventura o conector for retirado, não é necessário realizar o isolamento do cabo pois, segundo SCHNEIDER ELECTRIC (2015) 58.6% da massa total do cabo é composta de elastômero, portanto o isolante tende a recuperar suas dimensões iniciais, assim se auto regenerando.

Figura 5: Diferentes tipos de cabos utilizados



Fonte: INTERLINKBT, 2002.

Outra característica do cabo AS-Interface é que em geral eles não possuem blindagem. De acordo com PEPPERL+FUCHS (2012), cabos blindados podem ser usados, não para melhorar a imunidade ao ruído, mas sim para simples proteção mecânica. Ressaltando ainda que, devido as características elétricas do protocolo, se utilizado um cabo blindado, pode-se reduzir a performance da rede em até 20%.

Abordando agora rapidamente sobre as características elétricas do cabo AS-Interface, segundo AS- INTERNATIONAL ASSOCIATION (2000), o cabo deve seguir as seguintes características elétricas (frequência de 167KHz):

R´: < 90 mΩ/m

C´: < 80 pF/m

Z: 70...140 Ω

G´: ≤ 5 µS/m

L’: 400...1300 nH/m

Mais um aspecto muito importante do cabo, no que diz respeito a suas características elétricas, é a curva de impedância por frequência. Para as faixas acima e abaixo da frequência usada no protocolo (167KHz) há uma forte atenuação, o que resulta num aspecto bastante positivo para a transmissão dos dados, visto que ruídos elétricos acima ou abaixo da frequência de operação são bastante atenuados.

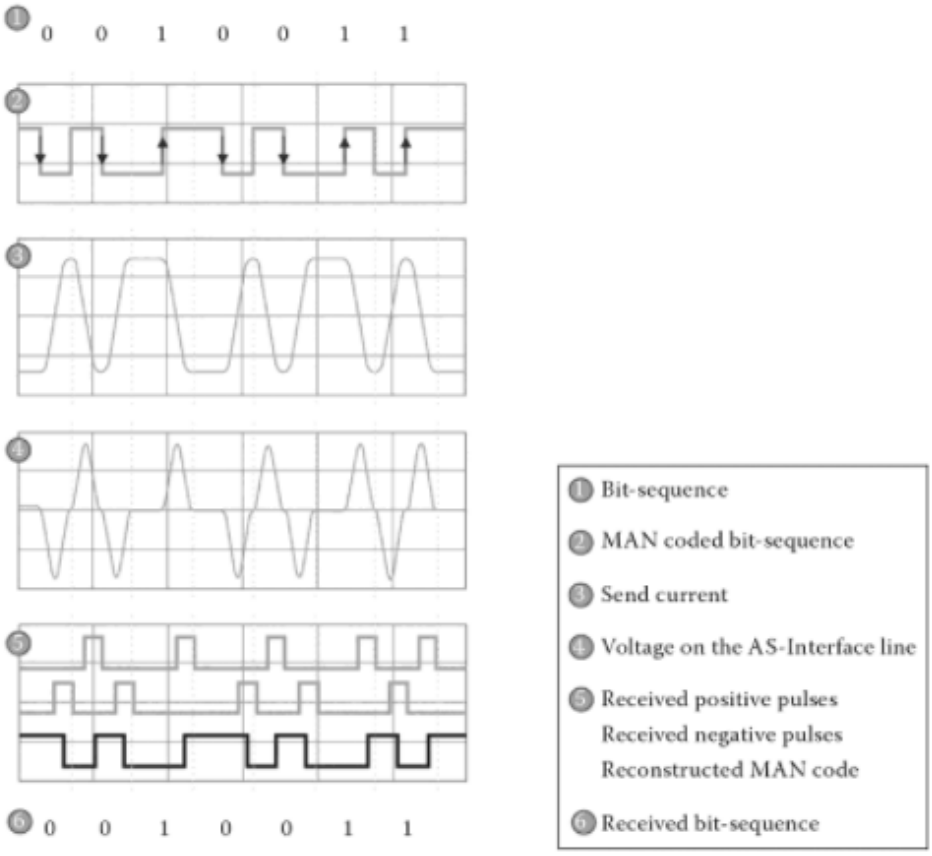
Agora que já foram vistas as principais características dos cabos utilizados na rede, se dará prosseguimento sobre a abordagem da camada física do protocolo, presente na interface 2. Será detalhado agora, como a informação trafega pela rede, ou seja, como os dados são codificados, modulados e transmitidos e recebidos para que a troca de mensagens enfim ocorra.

A modulação usada na rede AS-Interface é chamada de APM (*Alternative Pulse Modulation*). Esse tipo de modulação é utilizada pois o sinal das mensagens sobreposto a tensão de alimentação, deve ser livre da componente DC. Segundo ZURAWSKI (2015) esse tipo de modulação também traz várias vantagens como por exemplo, devido a forma de onda da modulação ser um sinal sen², as exigências quanto ao limite de baixa frequência e baixa emissão de ruído são atendidas simultaneamente. Outra característica desse tipo de modulação consiste no uso de tensão diferencial, portanto não existe um ponto de terra no barramento e, os níveis de tensão na linha variam entre ±2V.

Prosseguindo agora para o processo de transformação do sinal do dado binário até o sinal do barramento que, pode ser observado na figura 6. Primeiramente todas as mensagens binárias são codificadas no formato Manchester II, que será melhor explanado no capítulo 3.4.1. Essa codificação acarreta em uma mudança de fase no sinal que resulta no envio de corrente na linha (ZURAWSKI, 2015).

O próximo passo é a modulação através da técnica APM (Alternative Pulse Modulation) onde, cada nível de subida de corrente irá implicar em um pulso negativo de tensão e, para cada nível de descida de corrente, tem-se um pulso positivo de tensão na linha. A taxa de transferência de dados é de 166.67 Kbits/s, o que resulta num tempo de bit de 6µs.

Figura 6: Método de modulação e demodulação do sinal no barramento



Fonte: ZURAWSKI, 2015.

2.1.5 Comunicação

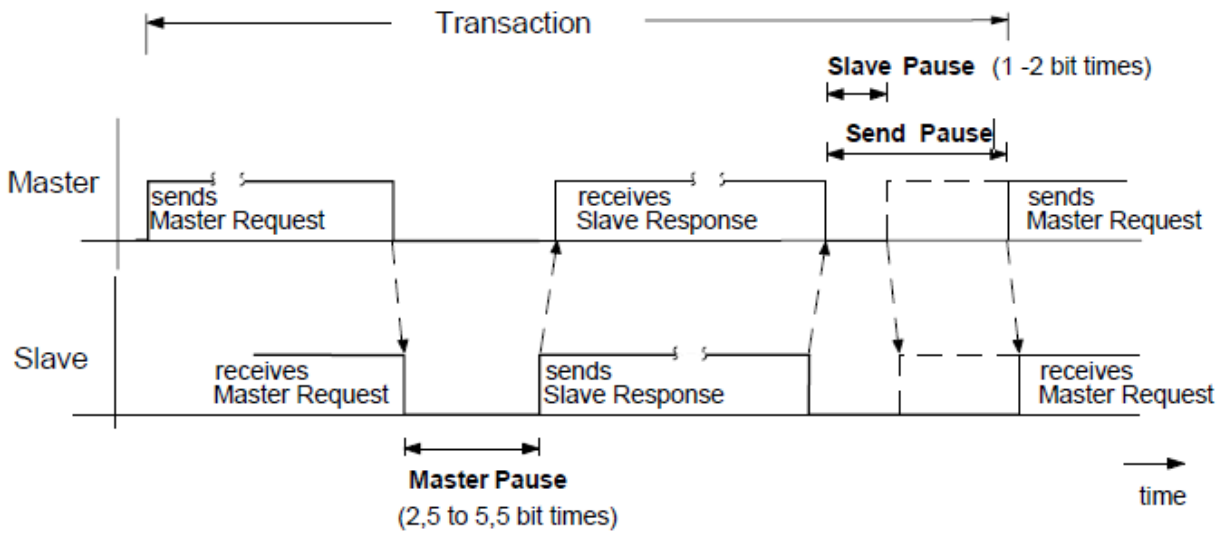
A comunicação da rede AS-Interface é realizada através de um sistema de acesso do tipo mestre-escravo, onde um único mestre pode controlar diversos escravos. O mestre dita para os escravos quando devem transmitir ou receber uma mensagem, evitando assim, conflitos de dados no barramento. Esse tipo de comunicação utiliza a técnica de *polling* cíclico em todos os escravos (LOUIS, 2016).

A comunicação se inicia com um comando do mestre e, logo após, ele aguarda pelo frame de resposta do escravo. Nesse intervalo, há uma pausa, geralmente entre 15 µs à 33 µs (2.5 à 5.5 bits) antes que haja efetivamente a resposta do escravo. Porém, o mestre é capaz de aceitar uma resposta válida em até 60 µs (10 bits) e, caso isso não ocorra, o mestre interpreta como uma resposta negativa e tenta retransmitir o frame apenas mais uma vez, em caso de transações múltiplas. Havendo uma resposta válida

do escravo, inicia outro tempo de pausa no barramento que tem duração de 6 µs à 12 µs (1 à 2 bits). Todo esse processo pode ser visto na Figura 7: Tempos de pausa do mestre e do escravofigura 7.

Segundo ALMEIDA et al. (2007) essa pausa antes do envio da próxima requisição pode atingir até 500 µs, desde que, não exceda o tempo total de ciclo da rede[1], isso só é possível se houver 30 escravos ou menos no enlace. Dessa forma, disponibilizando ao mestre maior tempo para o processamento interno das suas funções de controle.

Figura 7: Tempos de pausa do mestre e do escravo



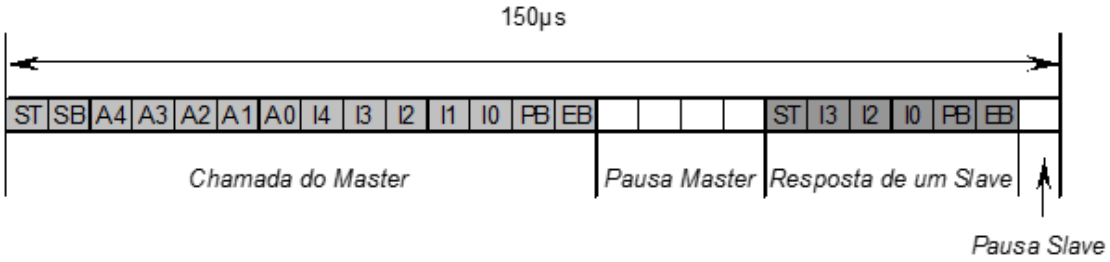
Fonte: AS-INTERNATIONAL ASSOCIATION, 2000.

A estrutura da mensagem, também conhecida como frame[2] é composta por diversos elementos que caracterizam a comunicação do protocolo AS-Interface. Os frames enviados pelo mestre e pelos escravos são diferentes tanto em tamanho quanto em conteúdo. O frame da requisição do mestre é composto de 14 bits, já o frame de resposta do escravo possui apenas 7 bits, conforme visto na figura 8.

[1] Para 31 escravos o ciclo da rede é de 5ms. Para 62 escravos o ciclo é de 10ms.

[2] Frame: Unidade de dados presente na camada de enlace e formada por uma sequência de bits.

Figura 8: Estruturação de um frame AS-Interface



Fonte: ALMEIDA et al. (2007)

O conteúdo do frame de requisição do mestre, na especificação 2.0, pode ser visto na tabela 3.

Tabela 3: Conteúdo do frame de requisição do mestre

Bit	Semântica	Descrição
ST:	Bit de início	Identifica o início do frame. ST = 0 sempre
CB:	Bit de controle	Identifica o tipo de requisição dos dados 0 = dado, parâmetro e endereço 1 = comandos
A4..A0:	Endereço	Endereço do escravo 0x00 = endereço zero 0x01 .. 0x1F = escravos de 1 à 31
I4..I0:	Informação	Os 5 bits de informação a serem transferidos para o escravo
PB	Bit de paridade	A soma de todos os bits "1", excluindo o ST e EB deve ser par
EB	Bit de fim	Identifica o fim do frame. EB = 1 sempre

Fonte: ALMEIDA et al. (2007).

Na tabela 4 pode se observar o frame de resposta do escravo.

Tabela 4: Conteúdo do frame de resposta do escravo

Bit	Semântica	Descrição
ST:	Bit de início	Identifica o início do frame. ST = 0 sempre
I3..I0:	Informação	Os 4 bits de informação a serem transferidos para o mestre
PB	Bit de paridade	A soma de todos os bits "1", excluindo o ST e EB deve ser par
EB	Bit de fim	Identifica o fim do frame. EB = 1 sempre

Fonte: ALMEIDA et al. (2007).

Como se pode observar na tabela 1, o protocolo AS-Interface passou por 3 versões de especificação, aumentando cada vez mais seus pontos de entrada e saída. Na especificação 2.1, para realizar o endereçamento estendido de até 62 escravos, o bit de informação I3 passou a carregar o bit de seleção A ou B (1 ou 0), permitindo assim endereços de 1A até 31A e 1B até 31B, ao custo de 1 bit de dados no frame. Por essa razão os dispositivos dessa especificação possuem o número de I/O como 4/3 (4 entradas e 3 saídas).

Posteriormente, na versão 3.0, foi incluído um novo "*profile*" 7.A.x.E, que permite o uso de 4 entradas e 4 saídas mesmo com o endereçamento estendido. Isso é possível pois os dados de saída foram divididos em 2 *nibble*[3] onde, o bit I3 continua sendo o bit de seleção para o endereçamento estendido, o bit I2 passa a ser o bit de seleção da transmissão do *nibble* desejado e, apenas os bits I1 e I0 contém os dados. Assim, para realizar a atualização das 4 saídas é necessário enviar 2 frames de requisição do mestre, por isso o tempo de ciclo da rede passa de 10 ms para 20 ms.

Existem 12 tipos diferentes de mensagem de requisições do mestre no protocolo AS-Interface. Na tabela 5 estão listados todos os frames segundo AS-INTERNATIONAL ASSOCIATION (2000) que são utilizados no protocolo.

Tabela 5: Frames do mestre em modo de endereçamento padrão

Instrução	ST	CB	Endereço					Informação					PB	EB
Troca de Dados	0	0	A4	A3	A2	A1	A0	0	D3	D2	D1	D0	PB	1
Escrita de Parâmetros	0	0	A4	A3	A2	A1	A0	1	P3	P2	P1	P0	PB	1
Atribuição de endereço	0	0	0	0	0	0	0	A4	A3	A2	A1	A0	PB	1
Escrita ID-Code_1	0	1	0	0	0	0	0	0	ID3	ID2	ID1	ID0	PB	1
Apagar endereço	0	1	A4	A3	A2	A1	A0	0	0	0	0	0	PB	1
Reiniciar escravo	0	1	A4	A3	A2	A1	A0	1	1	1	0	0	PB	1
Leitura configuração IO	0	1	A4	A3	A2	A1	A0	1	0	0	0	0	PB	1
Leitura ID-Code	0	1	A4	A3	A2	A1	A0	1	0	0	0	1	PB	1
Leitura ID-Code_1/2	0	1	A4	A3	A2	A1	A0	1	0	0	1	0/1	PB	1
Leitura de Status	0	1	A4	A3	A2	A1	A0	1	1	1	1	0	PB	1
R1	0	1	A4	A3	A2	A1	A0	1	1	1	1	1	PB	1
Broadcast (Reiniciar)	0	1	1	1	1	1	1	1	0	1	0	1	PB	1

Fonte: AS-INTERNATIONAL ASSOCIATION, 2000. TRADUÇÃO LIVRE.

[3] *Nibble*: Sucessão de 4 bits (meio byte).

- Troca de Dados: Mensagem utilizada pelo mestre para transferir 4 bits de saída para o escravo e obter os 4 bits de entrada através do frame de resposta;
- Escrita de Parâmetros: Usado pelo mestre para transferir 4 bits de parâmetros para o escravo e ler 4 bits de parâmetros de entrada como resposta;
- Atribuição de Endereço: Atribui o endereço (não volátil) ao escravo;
- Escrita ID-Code_1: Usado para atribuir o *nibble* do ID-Code ao escravo que possui endereço zero;
- Apagar Endereço: Utilizado para deletar o endereço de um escravo específico;
- Reiniciar escravo: Comando utilizado para reiniciar um escravo específico;
- Leitura Configuração I/O: Comando para leitura da configuração I/O de um escravo;
- Leitura ID-Code: Comando para leitura do código de identificação de um escravo;
- Leitura ID-Code_1/2: Comandos para leitura do código de identificação 1 ou 2 de um escravo específico;
- Leitura de Status: Comando para leitura do registro de status do escravo;
- R1: Comando de requisição reserva;
- Broadcast: Comando utilizado pelo mestre para reiniciar todos os escravos da rede.

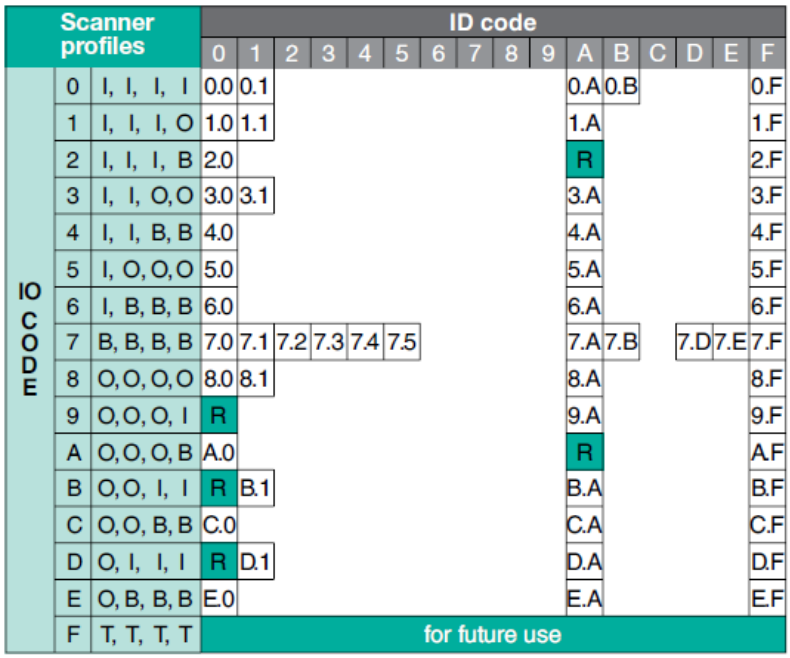
2.1.6 Perfis

De acordo com AS-INTERNATIONAL ASSOCIATION (2002) os perfis dos dispositivos englobam a representação dos bits de dados e de parâmetros se houver, ou seja, especificam a semântica dos dados. Outro aspecto dos perfis é que, independem das características físicas da rede. Na rede AS-Interface existem dois tipos de perfis: os do mestre, que será discutido posteriormente e dos escravos, que consistem em uma combinação única entre a configuração I/O, o código ID e os códigos estendido ID1 e ID2.

A configuração I/O, segundo PEPPERL+FUCHS (2012) define as entradas e saídas utilizadas em cada escravo. O código ID é responsável por definir o tipo de escravo como sensor, atuador, escravo padrão ou escravo A/B. Já os códigos ID1 e ID2

são opcionais, mas vários dispositivos AS-Interface possuem e, caracterizam os sub perfis alterando algumas funcionalidades do módulo. Na figura 9 pode ser observado as possíveis combinações entre os códigos I/O e ID, onde: (I = entrada, O = saída, B = bidirecional, T = *tristate*, V = virgem, R = reservado).

Figura 9: Perfis dos escravos



Fonte: PEPPERL+FUCHS, 2012.

A tabela 6 apresenta o significado para cada código ID do escravo. Já o código ID1 tem por padrão o valor 0x0F para escravos com endereçamento padrão e 0x07 para escravos A/B. O código ID2 é mais complexo, usado em escravos com funções especiais, como analógicos.

Tabela 6: Valores e significados do código ID

Código ID	Descrição
0	Conexões 4 I/O para sensores e atuadores binários com um sinal cada
1	Duas conexões 2 I/O para sensores e atuadores binários com dois sinais cada
A	Escravos em modo de endereçamento extendido (A/B)
B	Escravo de segurança AS-i
F	Dispositivo especificado pelo fabricante

Fonte: IFM, 2012b. TRADUÇÃO LIVRE.

Conforme a AS-INTERNATIONAL ASSOCIATION (2000) existem diversos perfis diferentes de escravos, cada um dedicado a um tipo de aplicação. Será citado aqui, apenas os principais grupos e, alguns perfis serão discorridos com mais detalhes no capítulo 3.5.3.

- Perfis livres (S-X.F): I/O = X, ID = F, onde X = 0 … E;
- Portas I/O remota (S-X.0): I/O = X, ID = 0, onde X = 0 … E, menos 9, B e D;
- Reservado para escravos A/B (S-X.A): I/O = X, ID = A, onde X = 0 … E, menos 2 e A.

Equipamentos de acordo com a especificação V2.11 suportam apenas simples modos de operação e acesso a variáveis analógicas. Já os dispositivos mais novos de acordo com a especificação V3.0 suportam vários perfis mais complexos, como por exemplo: S-7.A.7 (4DI/4DO), S-7.A.A (8I/8O), S-7.A.9 (2 canais analógicos) e S-6.0 (canal analógico rápido) (SIEMENS, 2006).

Os perfis do mestre, identificados na tabela 7, segundo AS-INTERNATIONAL ASSOCIATION (2000), distinguem no número de escravos suportados, ou seja, se suportam escravos padrão ou com endereçamento estendido (máximo de 31 e 62 respectivamente), em certas funcionalidades e suporte a determinados perfis específicos de escravos.

Tabela 7: Perfis do mestre de acordo com a especificação

Versão da Especificação	V2.0	V2.11	V3.0
Ano	1994	1998	2004
Mestre	M0, M1, M2	M3	M4
Escravos	Todos os outros perfis de escravos permitidos	S-*.A.*.*, S-7.3.*.*, S-7.4.*.*	S-6.0.*.*, S-7.5.*.5, S-7.A.*.5, S-B.A.*.5, S-7.A.*.7, S-7.A.*.8, S-7.A.*.9, S-7.A.*.A

Fonte: AS- INTERNATIONAL ASSOCIATION: AS-Interface Academy. TRADUÇÂO LIVRE.

Mestres do tipo M0 possuem o mínimo de funcionalidades, suportando apenas endereçamento padrão de no máximo 31 dispositivos e suportam apenas a simples comunicação de dados. O perfil M2 e M1 também suportam apenas 31 dispositivos mas diferenciam no suporte as funcionalidades, enquanto o M2 utiliza apenas alguns

parâmetros, o M1 tem suporte completo a especificação V2.0. Já mestres do tipo M3 possuem endereçamento estendido de até 62 escravos, troca de dados 4I/3O e também suporta os perfis analógicos S-7.1 e S-7.3. Finalmente o perfil M4 que possui as funcionalidades do M3, porém segue a especificação V3.0 onde, onde há o suporte de 4I/4O para troca de dados, suporte a analógicas com endereçamento estendido e comunicação serial bidirecional (PEPPERL+FUCHS, 2012).

2.2 CHIP ASI4U

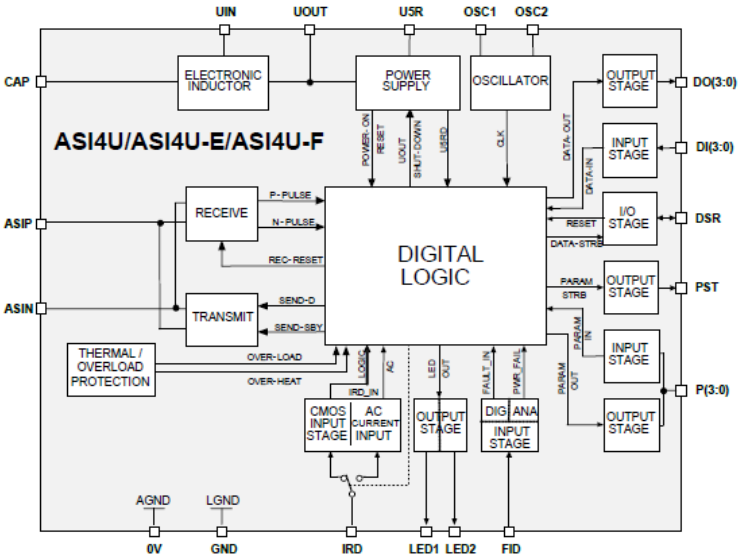
A maior parte da informação que será apresentada neste capítulo foi extraída de IDT (2016a) e algumas documentações técnicas do próprio fabricante.

O chip ASI4U consiste em um circuito integrado para redes AS-Interface totalmente compatível com a especificação V3.0. Suas aplicações consistem em: escravos, mestres, repetidores e analisadores de rede.

2.2.1 Funcionalidades do chip

Na figura 10 é possível observar o diagrama de blocos contendo as funcionalidades do circuito integrado.

Figura 10: Diagrama de blocos chip ASI4U



Fonte: IDT, 2016a.

Os blocos de recepção e transmissão são responsáveis pela conversão entre o sinal da rede AS-Interface e o sinal digital codificado para os dois sentidos da comunicação. Esse processo de modulação de sinal foi visto anteriormente na figura 6.

O bloco de Lógica Digital é responsável pelo processamento de toda a informação e pela comunicação com os outros blocos. Ele contém a decodificação do sinal UART, a máquina de estados do dispositivo, a memória EEPROM e outras lógicas de controle do chip.

O oscilador suportado por esse chip pode ser tanto de 8MHz quanto de 16MHz e, sua detecção é realizada automaticamente após aproximadamente 200µs. Também possui um relógio *watchdog* de 20µs caso não seja detectado nenhum oscilador.

O CI também possui módulos de entrada com *Schmitt Trigger* e resistores de *pull-up*. Já os estágios de saída suportam altas tensões, são do tipo dreno aberto e permitem um consumo de corrente de até 10mA.

As requisições do mestre AS-Interface são de acordo com a AS-INTERNATIONAL ASSOCIATION (2000), vista na tabela 5, porém, no chip ASI4U não existe a instrução R1 e existe uma instrução a mais, chamada de “entrar em modo de programa”, que será melhor explicada no próximo capítulo.

Assim como o mestre tem sua lista de frames a serem enviados aos escravos da rede, também há uma lista de frames de resposta para cada frame correspondente do mestre, enviado por cada escravo. Os comandos *broadcast* e modo de programa não retornam nenhuma resposta do escravo, como pode ser observado na tabela 8.

Tabela 8: Frames de resposta do escravo

Instrução	ST	I3	I2	I1	I0	PB	EB
Troca de Dados	0	D3 E3	D2 E2	D1 E1	D0 E0	PB	1
Escrita de Parâmetros	0	P3 I3	P2 I2	P1 I1	P0 I0	PB	1
Atribuição de endereço	0	0	1	1	0	0	1
Escrita ID-Code_1	0	0	0	0	0	0	1
Apagar endereço	0	0	0	0	0	0	1
Reiniciar escravo	0	0	1	1	0	0	1
Leitura configuração IO	0	IO3	IO2	IO1	IO0	PB	1
Leitura ID-Code	0	ID3	ID2	ID1	ID0	PB	1
Leitura ID-Code_1	0	ID3	ID2	ID1	ID0	PB	1

Leitura ID-Code_2	0	ID3	ID2	ID1	ID0	PB	1
Leitura de Status	0	S3	S2	S1	S0	PB	1
Broadcast (Reiniciar)			sem resposta				
Modo de programa			sem resposta				

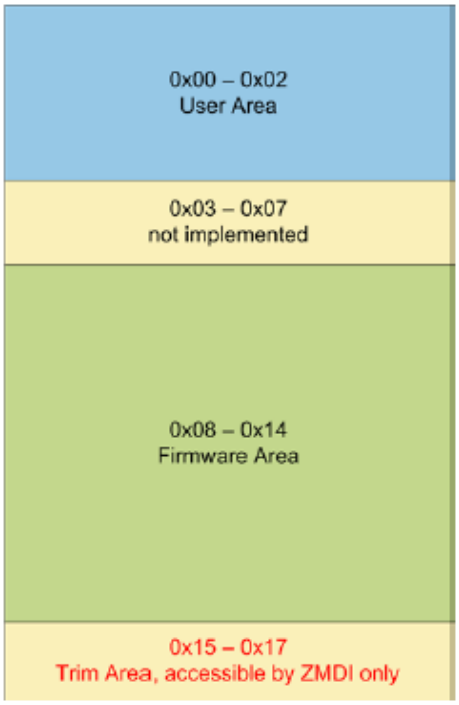
Fonte: IDT, 2016b. TRADUÇÂO LIVRE.

2.2.2 Memória EEPROM

O ASI4U contém uma memória do tipo EEPROM e, segundo CHEN (2004) esse tipo de memória pode ser apagada e reprogramada repetidas vezes através de pulsos elétricos, possuindo um número limitado de ciclos de gravação. Elas são usadas quando se necessita de fácil e rápida reprogramação e característica não voláteis, ou seja, não perdem o conteúdo mesmo após o desligamento do dispositivo.

A memória EEPROM do chip ASI4U possui tempo de escrita de 12,5ms e tempo de leitura de 110ns. Possui uma estruturação em duas áreas independentes: área de usuário e área de *firmware*. Também existem uma zona de memória não implementada e outra área acessível apenas pelo fabricante do chip.

Figura 11: Setores da memória EEPROM



Fonte: IDT, 2016b.

Conforme visto na figura 11, a EEPROM possui endereços hexadecimais de 0x00 até 0x17 com um total de 24 endereços, onde cada endereço da memória possui 4 bits.

A área de memória do usuário (0x00 - 0x02) contém apenas dados relacionadas a aplicação final e, não é necessário nenhuma configuração especial para escrever nesses endereços, bastando apenas utilizar os comandos AS-Interface de atribuição de endereço e escrita ID-Code-1. Na tabela 9, pode-se observar os detalhes dessa área de memória.

Tabela 9: Mapa da memória EEPROM - Área do usuário

ASI4U Endereço EEPROM [hex]	Posição do Bit	Parâmetro da EEPROM	Conteúdo do Registro da EEPROM
0	0 a 3	A0 a A3	Endereço do escravo *nibble* baixo
1	0	A4	Endereço do escravo *nibble* alto
2	0 a 2	ID1_Bit0 a ID1_Bit2	Código estendido ID1
2	3	ID1_Bit3	Código estendido ID1, seleção A/B escravo no modo de endereço estendido

Fonte: IDT, 2016b. TRADUÇÂO LIVRE.

Já a área de *firmware* (0x08 - 0x14) contém todos os dados de configuração relacionados a fabricação e, possui proteção contra leitura e escrita através da configuração de uma *flag* da própria memória, presente no endereço 0x0C, bit 1, visto na tabela 10. Portanto, para acessar essa área de memória é necessário o comando de uma função especial, chamada de “entrar em modo de programa”.

Tabela 10: Mapa da memória EEPROM – Área de firmware

ASI4U Endereço EEPROM [hex]	Posição do Bit	Parâmetro da EEPROM	Conteúdo do Registro da EEPROM
8	0 a 3	*ID_Bit0 to ID_Bit3*	Código ID
9	0 a 3	*ID2_Bit0 to ID2_Bit3*	Código estendido ID2
A	0 a 3	*IO_Bit0 to IO_Bit3*	Código IO
B	0	*Multiplex_Data*	Porta dados multiplex bidirecional
	1	*Multiplex_Parameter*	Porta parâmetros multiplex bidirecional
	2	*P0_Watchdog_Activation*	*Watchdog* ligado/desligado pelo pino P0
	3	*Watchdog_Active*	*Watchdog* continuamente ativado
C	0	*Master_Mode*	Se 1, sem acesso a área de *Firmware*
	1	*Program_Mode_Disable*	Se 1, área de *Firmware* protegida
	2	*Repeater_Mode*	Se 1, sem acesso a área de *Firmware*

	3	*Invert_Data_In*	Todas as portas de entrada são invertidas
D	0 a 3	*DI_Invert_Configuration*	Inversão portas de entrada selecionadas
E	0 a 3	*DI_Filter_Configuration*	Filtro *anti-boucing* portas de entrada selecionadas
F	0 a 2	*DI_Filter_Time_Constant*	Constante de tempo filtro de entrada
	3	*P1_Filter_Activation*	Se 1, o pino P1 ativa filtro de entrada
10	0 a 3	*Data_Out_Configuration*	Define se os pinos de saída são controlados pelo registro de saída de dados ou pelo *Data_Out_Value*
11	0 a 3	*Data_Out_Value*	Se Data_Out_Configuration é 1 armazena os dados da porta de saída
12	0	*Enhanced_Status_Indication*	Se 1 modo de indicação status avançado
	1	*Dual_LED_Mode*	
	2	*FID_Invert*	Inversão do valor de entrada FID
	3	*Safety_Mode*	Se 1, modo de segurança ativado
13	0	*Synchronous_Data_IO*	Ativa sincronização de dados I/O
	1	*P2_Sync_Data_IO_Activation*	Se 1, o pino P2 ativa sincronização de dados I/O
	2	*Ext_Addr_4I/4O_Mode*	Ativa suporte para modo 4I/4O
	3	*ID_Code1_Protect*	Se 1, ID1 protegido contra acesso usuário
14	0 a 3	*ID1_Bit0 to ID1_Bit3*	Se ID_Code1_Protect é 1, a requisição Read_ID_code_1 responderá com os valores armazenados nesse registro

Fonte: IDT, 2016b. TRADUÇÂO LIVRE.

2.2.3 Modos de operação

O chip ASI4U possui dois modos principais de operação que são o modo escravo e o modo mestre. Existe também dois sub modos, o modo repetidor e o modo analisador. A seleção entre eles é feita por dois bits da memória EEPROM (endereço 0x0C, bit 0 e bit 2), conforme a tabela 11.

Tabela 11: Modos de operação do chip ASI4U

Modo de operação	*Flag* Modo Mestre	*Flag* Modo Repetidor
Modo escravo	0	0
Modo mestre	1	0
Modo repetidor	1	1
Modo analisador	0	1

Fonte: IDT, 2016b. TRADUÇÂO LIVRE.

No modo escravo, o CI funciona com todas as características previstas na especificação V3.0, sendo esse, o modo de operação mais complexo do chip.

Já no modo mestre, o CI simplesmente tem a tarefa de converter o sinal AS-Interface em sinal digital manchester e vice versa. Outra característica do chip em modo mestre é que toda resposta recebida do escravo é verificada quanto a sua consistência e gerado um sinal de recepção no pino P2 (Parâmetro 2), já no pino P1 o mestre detecta se houve alguma falha de alimentação da rede.

2.2.4 Modo de programação

Conforme visto no capítulo 2.2.2, existe uma área de *firmware* na EEPROM que só tem permissão de escrita mediante a requisição de modo de programa do mestre. Outro detalhe importante é que este modo só está disponível quando o chip opera em modo escravo. O CI ASI4U também deve estar programado com o endereço zero.

Após o recebimento desse frame de requisição, o endereço 0x0C, bit 2 (*flag* modo de programa) é colocado em nível lógico baixo, permitindo a escrita na área de *firmware* da memória.

Feito esse procedimento, os comandos troca de dados e escrita de parâmetros passam a ter outro significado, onde o primeiro é usado para ler um endereço específico da EEPROM e, o segundo para escrever em algum endereço na memória. No frame do mestre, os bits de endereço A0 à A3 são usados para endereçar um dos doze endereços disponíveis na área de *firmware*. O bit A4 não é utilizado. Já os bits de informação I0 a I3 carregam o valor a ser armazenado ou lido na EEPROM. Uma vez dentro do modo de programa, para sair basta reiniciar o escravo.

2.2.5 Canais de comunicação

Em modo escravo, o chip ASI4U pode se comunicar através de dois canais diferentes: o canal AS-Interface e o canal IRD. O canal AS-Interface é conectado diretamente ao barramento da rede através dos pinos ASIP e ASIN do CI.

O canal IRD, também chamado de canal de endereçamento, utiliza dois pinos para a comunicação. O pino IRD recebe o sinal digital codificado no formato Manchester-II, enquanto o pino LED1 retorna todas as respostas do escravo também no mesmo formato.

Para que esse canal seja ativado é necessário enviar ao pino IRD uma sequência de frames que, consiste em quatro telegramas AS-Interface consecutivos, codificados no formato Machester-II com uma janela de tempo de 8ms. O escravo não retornará nenhuma resposta a essa requisição, apenas verificará internamente se a sintaxe do frame está correta (número de bits, bit de início, bit de fim e paridade). Uma vez ativado o canal de comunicação IRD, ele só será desativado com a reinicialização do chip. Com a ativação do canal IRD o canal AS-Interface é desativado, sendo este último, o canal padrão do chip.

2.3 PLACA STM32F4DISCOVERY

A placa STM32F4DISCOVERY é utilizada para o desenvolvimento de diversas aplicações e possui um microcontrolador STM32F407VGT6 com ARM® Cortex® - M4 e núcleo de 32 bits. O microcontrolador também conta com 1 Mbyte de memória flash e 192 Kbytes de RAM. Também está inclusa uma ferramenta embarcada de depuração, acelerômetros, diodos emissores de luz, botões, microfone e conector USB (ST, 2016a).

Essa placa possui um cristal de 8 MHz, mas de acordo com ST (2015) internamente ela pode ser configurada com uma frequência de até 168MHz. Existem 2 barramentos (APB1 e APB2) onde estão conectados os periféricos como: temporizadores, portas seriais, entradas e saídas analógicas e outros pinos de comunicação. O barramento APB1 suporta uma frequência de 42 MHz e o APB2 de 84 MHz. O diagrama de blocos do microcontrolador com todas essas funcionalidades pode ser visto em maiores detalhes no anexo A (ST, 2015).

Está incluso também uma ferramenta embarcada de programação e depuração, a ST-LINK/V2. Com ela é possível programar a placa e depurar todo o programa através do uso de *breakpoints*. Outra opção permitida consiste na utilização da ST-LINK/V2 com uma placa externa através do conector SWD (ST, 2016a).

2.3.1 Periféricos

O microcontrolador STM32F407VGT6 possui inúmeros periféricos para utilização do usuário e, segundo HITEX (2009) são divididos em dois grupos, os periféricos de propósito geral e os periféricos de comunicação. Foge do escopo, uma abordagem profunda de todos os recursos do microcontrolador, portanto será explanado apenas os periféricos utilizados no projeto do mestre.

2.3.1.1 Interrupção externa

Segundo HITEX (2009) o microcontrolador STM32 possui 16 linhas dedicadas de interrupção externa nas portas de GPIO. Uma vez criada, basta selecionar uma requisição de interrupção ou evento por borda de subida, borda de decida ou ambas. O único detalhe é que, cada linha de interrupção só pode ser usada em um pino de qualquer porta. Outro detalhe importante sobre as interrupções externas é que elas só são detectadas se ocorrerem por um prazo maior que o período do barramento APB2, que é de aproximadamente 11.9 ns (ST, 2015).

2.3.1.2 Temporizador

O primeiro temporizador é o *systick*, *timer* este, de uso geral do sistema onde é permitido a configuração de sua base de tempo (1 µs, 1 ms, etc.) para uso principalmente em atrasos de tempo para o programa.

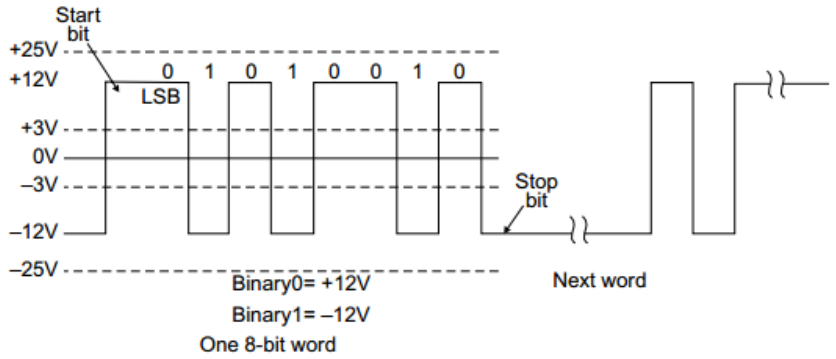
Quanto aos outros temporizadores, existem *timers* de propósito geral, básicos e avançados, diferenciando entre eles as funções que cada um suporta, como por exemplo PWM, DMA, *encoder,* entre outras. Foi usado no programa apenas o temporizador básico (TIM6 e TIM7). Os dois *timers* possuem 16 bits de resolução, ou seja, contam até 65535, onde seu período depende do valor de *prescaler* definido. Também é possível configurar uma interrupção para que aconteça ciclicamente, em um período pré-estabelecido, ou então apenas utilizar o temporizador para contagem de tempo.

2.3.1.3 Comunicação serial

As interfaces de comunicação com periféricos podem ser realizadas através da UART, lendo e escrevendo bits assincronamente com velocidades de transmissão típicas de 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 ou 115200 bps. A principal característica da comunicação assíncrona, consiste no fato de poder transmitir o dado apenas quando for desejado. Isso acarreta o uso de bits adicionais, além dos 8 bits de dados, para delimitação do frame, os chamados bits de início e bit de fim e, em alguns casos também possui o bit de paridade para verificação de erros no frame (LOUIS, 2016; CHEN, 2004).

Utilizando a interface de comunicação da UART, LOUIS (2016) diz que é possível ter diversos protocolos, como por exemplo: RS-232, RS-422 e RS-485. Esses protocolos diferem quanto ao nível de tensão do sinal, conectores, taxa de sinalização, capacitâncias e características de conexão da rede. Por exemplo o protocolo RS-232 permite conexões ponto a ponto e com nível de tensão para o bit 0 entre +3 e +25V e o bit 1 com -3V a -25V. Tipicamente os dados transmitidos são codificados em caracteres através da tabela ASCII. Na figura 12 é mostrado um exemplo de dado transmitido via UART com protocolo RS-232 onde, o dado 01001010, 74 em decimal, pela tabela ASCII representa o caractere ‘J’.

Figura 12: Transmissão via UART em RS-232



Fonte: LOUIS, 2016.

O microcontrolador STM32F407 possui duas interfaces UART, a UART4 e UART5, estando estas, conectadas ao barramento APB1 (42 MHz). Dependendo da taxa de

amostragem, a frequência máxima de comunicação onde, para taxas de 16 amostras por bit, é de 2.62 Mbit/s. Já com 8 amostras por bit a frequência máxima é de 5.25 Mbit/s (ST, 2015).

2.4 APLICAÇÕES INDUSTRIAIS

As características mais importantes da rede AS-Interface para as diversas aplicações industriais são: baixo custo, performance robusta e a eficiência em conectar todos os dispositivos usando apenas um par de fios, transmitindo alimentação e comunicação digital simultaneamente (SENSE, 2004).

2.4.1 Equipamentos e fabricantes

Este tópico tem como objetivo mostrar alguns equipamentos industriais comerciais que, fazem parte de uma rede AS-Interface. Entre os principais fabricantes estão: Siemens, FMI, Festo, Pepperl+Fuchs e Bihl+Wiedemann.

Para aplicações típicas industriais tem-se, por exemplo, o módulo mestre CM AS-Interface da Siemens, usado em comunicação com o CLP SIMATIC ET 200SP. É permitido conectar diversos mestres em um único CLP, porém, o limite de módulos depende de dois fatores: o máximo espaço de endereçamento utilizado no módulo de interface do CLP e a versão de *firmware* do mestre onde, a V1.0 ocupa 32 bytes de endereçamento, enquanto a V1.1 suporta até 288 bytes (SIEMENS, 2015).

Figura 13: Módulo CM Mestre AS-i

Fonte: SIEMENS, 2015.

Uma alternativa ao uso de cartões mestre acoplados ao CLP são as placas PCI mestres do fabricante Bihl+Wiedemann. Essas placas possuem 2 mestres AS-Interface de acordo com a especificação 3.0, possuindo perfil M4, embutidos em uma única placa. São do tipo *plug and play* onde, bastam ser conectadas a entrada PCI do PC (BIHL+WIEDEMANN, 2016).

Figura 14: Placa PCI Mestre AS-i

Fonte: BIHL+WIEDEMANN, 2016.

Gateways são dispositivos que interconectam a rede AS-Interface a redes de nível superior como: PROFIBUS, DeviceNet e Ethernet (PEPPERL+FUCHS, 2012).

Figura 15: Gateway entre o protocolo AS-Interface e RS-232

Fonte: PEPPERL+FUCHS, 2012.

Um acessório bastante útil é o endereçador da rede AS-Interface. Com ele é possível realizar a leitura dos códigos: ID, ID1, ID2 e IO, escrever parâmetros, ler e escrever dados e o mais importante, atribuir endereço ao escravo. Todas essas

funcionalidades podem ser feitas conectando o escravo diretamente ao dispositivo ou conectando ao cabo AS-Interface provido de alimentação e o mestre desligado (IFM, 2012a).

Figura 16: Endereçador de escravos

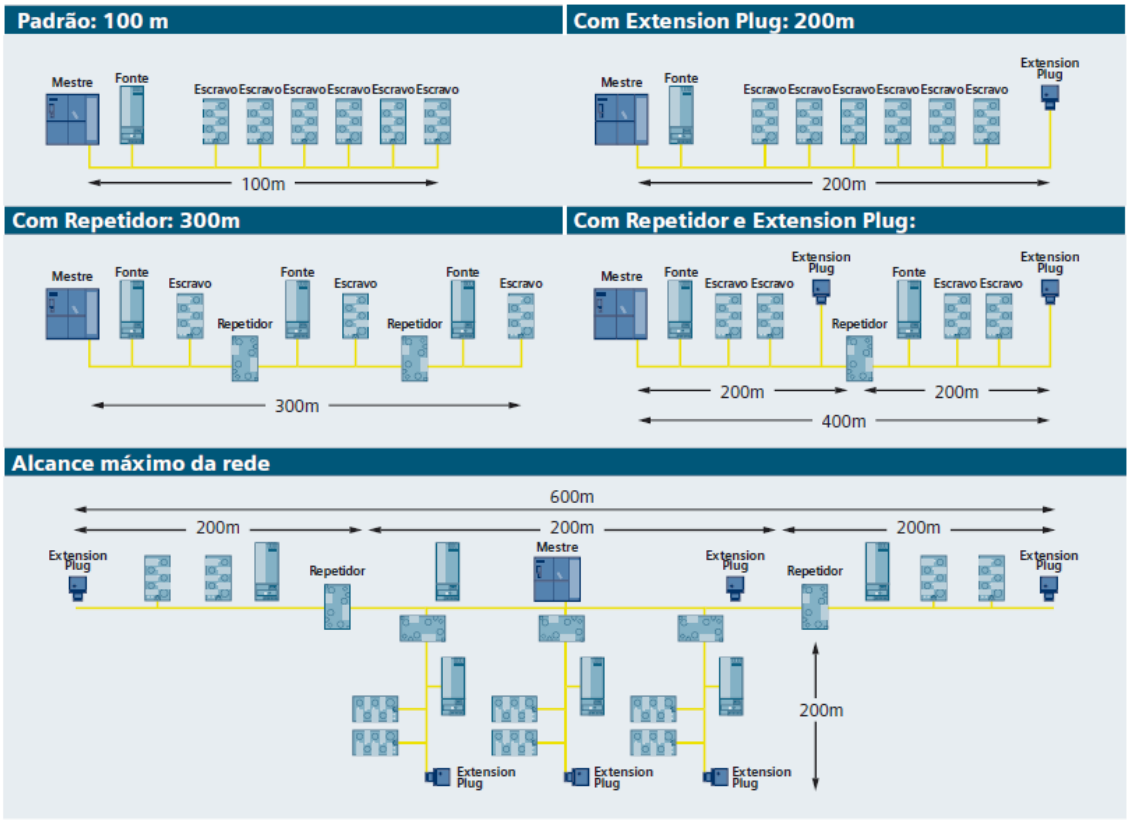
Fonte: IFM, 2012a.

2.4.2 Recomendações gerais de instalação

Segundo SENSE (2004) existem diversos requisitos, do ponto de vista da camada física, que são de suma importância para garantir a comunicação eficiente entre os dispositivos. Um deles é a tensão nominal de alimentação da rede AS-Interface, entre 29.5V e 31.6V, permitindo uma queda de tensão ao longo do cabo de até 3V. A fonte de alimentação pode ser instalada em qualquer lugar da linha e deve possuir um conjunto de indutores para desacoplamento dos dados.

Além da flexibilidade das topologias de rede, é possível expandir a distância da linha em até 600 metros com o uso de certos dispositivos. Segundo SIEMENS (2016) uma distância de até 100 metros não necessita de nenhum componente adicional. Utilizando um plugue extensor, pode-se atingir a marca de 200 metros. Existe também os repetidores de sinal onde, cada trecho da linha entre os repetidores deve possuir uma fonte de alimentação própria, o que resulta numa distância de até 300 metros. Chegando a até 600 metros de comprimento. Combinando vários destes elementos, pode-se atender as mais diversas aplicações industriais, conforme visto na figura 17.

Figura 17: Estrutura e alcance da rede AS-Interface



Fonte: SIEMENS, 2016.

3 PROJETO DA REDE AS-INTERFACE

Através deste capítulo será exposto a definição do projeto de uma rede AS-Interface, a metodologia utilizada para atingir os objetivos definidos no escopo, através do estudo das tecnologias presentes no mercado e atendimento dos requisitos, seguindo a especificação AS-Interface.

3.1 METODOLOGIA

Foram realizadas as seguintes etapas para o desenvolvimento do projeto:

1) Estudo da literatura de protocolos industriais

Para realizar a implementação da rede AS-Interface foi necessário estudar e analisar sobre os aspectos dos protocolos industriais em geral e aplicações onde cada um se destina.

2) Estudo da norma do protocolo AS-Interface

Foi de imprescindível importância o estudo aprofundado acerca do protocolo AS-Interface, como suas características elétricas, tempos de transmissão, frames, modulação, enfim tudo sobre o funcionamento completo da rede.

3) Testes em equipamentos industriais

Essa etapa foi fundamental para analisar o funcionamento da rede e poder ver todas as suas funcionalidades e configurações necessárias. Foi utilizado um CLP SIMATIC ET 200SP em conjunto com um módulo AS-Interface mestre, controlado pelo PC através do software TIA Portal da SIEMENS, um escravo da ThinkTop para válvulas da Alfa Laval e uma fonte de alimentação AS-Interface.

4) Definição dos componentes da rede

Após ter obtido o máximo de informações sobre o protocolo AS-Interface e adquirido as primeiras experiências práticas da rede, foi definido todos os módulos que iriam compor a rede, incluindo a interface de usuário, o mestre, os escravos e a fonte de alimentação.

5) Estudo acerca dos melhores componentes e placas eletrônicas

Foi realizado um levantamento diante das soluções comerciais disponíveis, para a escolha de um microcontrolador que atendia os requisitos de comunicação do mestre AS-Interface. Também foi levantado soluções para o escravo, desde seu núcleo, composto pelo chip AS-Interface, até todos os componentes eletrônicos necessários para o seu correto funcionamento.

6) Estudo detalhado sobre o chip ASI4U

Escolheu-se o chip ASI4U do fabricante IDT, pois há poucas alternativas disponíveis no mercado. Nessa etapa foi estudado detalhadamente todo o manual e documentações técnicas do CI, placas de exemplo, programação, enfim tudo que pudesse ter alguma informação relevante a ser extraída e agregada ao projeto.

7) Estudo detalhado sobre a placa STM32F4DISCOVERY

Durante essa fase foi estudado em detalhes o funcionamento da placa STM32F4DISCOVERY, incluindo todos os seus recursos de hardware para atender os

requisitos básicos da rede e, também a plataforma a ser utilizada para a programação da placa, para que o mestre pudesse ser desenvolvido.

8) Definição da comunicação entre os módulos

Em posse de todos os módulos que compõem a rede, dos detalhes de funcionamento do CI ASI4U e da placa STM32F4DISCOVERY, foi necessário definir como seria feita a comunicação em duas frentes diferentes: entre a interface de usuário e o mestre, e entre o mestre e o escravo. Para a primeira escolheu-se utilizar a interface de comunicação serial UART em conjunto com o padrão RS-232 suportado tanto pelo computador quanto pela DISCOVERY. Já para o segundo meio de comunicação, utilizando o cabo AS-Interface, foi realizada no barramento apenas a modulação do sinal no formato Manchester-II.

9) Desenvolvimento da PCB do escravo

Após a definição de todas as diretrizes foi desenvolvido e implementado a placa de circuito impresso do escravo, envolvendo o chip ASI4U, toda a eletrônica necessária para o seu correto funcionamento e conexões para os cabos da rede. Optou-se por elaborar duas versões de PCB, a primeira para testes, construída com componentes PTH como primeiro protótipo. Após várias alterações, foi construida a segunda versão, 100% funcional, mais completa e com componentes SMD.

10)Desenvolvimento em linguagem C do *firmware* do mestre

O mestre, representado pela placa STM32F4DISCOVERY foi programado em linguagem C, desenvolvido do zero, abordando um mestre com perfil M2, implementando toda a camada de usuário, todos os fluxogramas que compõem a camada de controle de execução e a camada de controle de transmissão, de acordo com AS-INTERNATIONAL ASSOCIATION (2000), especificação V2.11.

11)Interligação e testes da rede

Finalizada as etapas de desenvolvimento do mestre e do escravo, foi feita a conexão elétrica entre o mestre e os escravos na rede e com o computador, estabelecendo a conexão com a interface de usuário, utilizando apenas fios comuns. Após toda a rede conectada, realizou-se diversos testes para correções de erros e para aferir a eficiência da rede projetada.

3.2 PROGRAMAS UTILIZADOS

Para o desenvolvimento do projeto foram utilizados os seguintes softwares: Eclipse, STM32Cube, SourceTree e KiCad.

- Eclipse

O Eclipse consiste numa plataforma de desenvolvimento voltada a várias linguagens diferentes como: Java, C/C++, JavaScript e PHP. Nesse projeto foi utilizada a plataforma em linguagem C, que possui suporte para ferramentas de depuração e compilador GCC específico para microcontroladores ARM. Com essa plataforma é possível programar todo o código, compilar, fazer download na placa e depurar linha a linha todo o programa.

- STM32Cube

O STM32Cube é um software embarcado, que compreende uma ferramenta voltada a microcontroladores STM, reduzindo substancialmente o tempo e o custo de desenvolvimento. Através dele se escolher funcionalidades a serem utilizadas na placa STM32DISCOVERY, como por exemplo o uso de GPIO's, configuração de temporizadores, portas de comunicação serial via UART, entre outras. Após a seleção dos recursos pode-se gerar um código fonte em linguagem C, contendo todas as definições especificadas na aplicação, preocupando-se assim, apenas com a lógica do programa.

- SourceTree

O SourceTree consiste em uma poderosa ferramenta gráfica para controle de versão para clientes Git ou Mercurial. Através dela pode-se realizar no repositório todas as funcionalidades do controle de versão, visualizando e gerenciando facilmente o desenvolvimento da aplicação.

- KiCad

O KiCad consiste em uma plataforma gratuita e completa para desenhos de placas eletrônicas (PCB), integrando desde o desenho do esquemático do circuito, o *layout* da placa, visualização em 3D até a geração de arquivos Gerber para manufatura. Outra

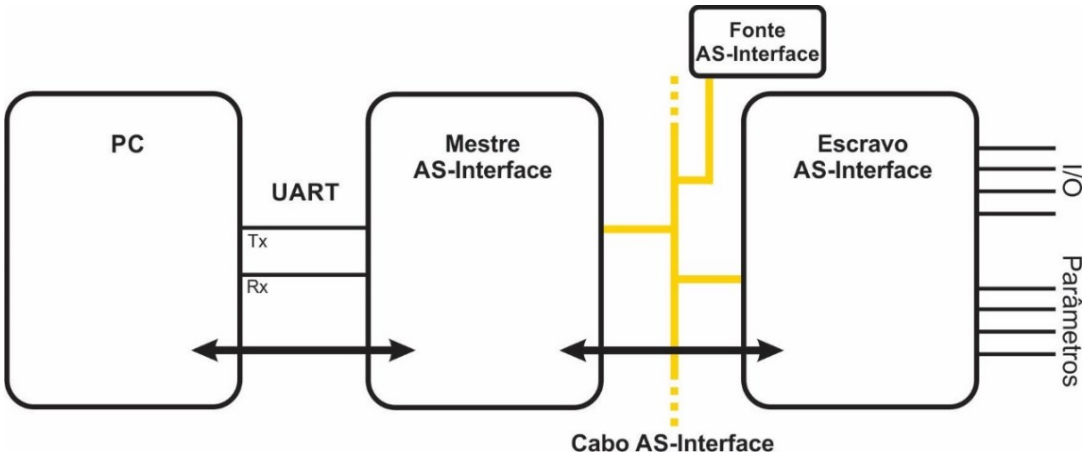
característica bastante interessante consiste no programa ser multiplataforma, disponível para Linux, Windows e Mac OS X.

3.3 DEFINIÇÃO DOS COMPONENTES DO PROJETO

De acordo com AS-INTERNATIONAL ASSOCIATION (2000), existem 3 componentes principais na rede: mestre, escravo e a fonte de alimentação, já vistos em detalhes no capítulo 2.1.1. O mestre é o coração da rede, sem ele os escravos ficam basicamente obsoletos, sem escravos o mestre não tem a quem comandar e trocar mensagens e sem a fonte de alimentação, não há energia para prover a rede. Portanto, sem qualquer um desses componentes, não é possível ter uma rede funcional, pois são de suma importância para o projeto do sistema como um todo.

O único item que foi adicionado foi a interface do usuário, mediante a necessidade de se possuir uma interface do usuário com a rede AS-Interface, ou seja, uma interface entre o computador e o mestre. Ficando claro que, sem esse componente a rede ainda funcionaria de forma autônoma, o mestre realiza o escaneamento da rede e atualiza suas listas de dados, porém não seria possível ler e escrever valores nas portas de entrada, saída e parâmetros de determinado escravo, configurar a memória EEPROM do chip ASI4U do escravo, alterando suas características, entre outras diversas funcionalidades que serão explicadas em detalhes posteriormente.

Figura 18: Componentes no projeto da rede AS-Interface



Fonte: Elaboração própria, 2017.

3.4 COMUNICAÇÃO DA REDE AS-INTERFACE

Na figura 18, pode-se observar que a rede é formada por duas interfaces independentes de comunicação, a comunicação entre o computador e o mestre e entre o mestre e os escravos. A primeira nada mais é que um conversor de sinais USB em sinais UART com padrão RS-232, para que o computador possa comunicar com o mestre da rede, enviando comandos e diagnosticando a rede através de mensagens de erro, status e respostas.

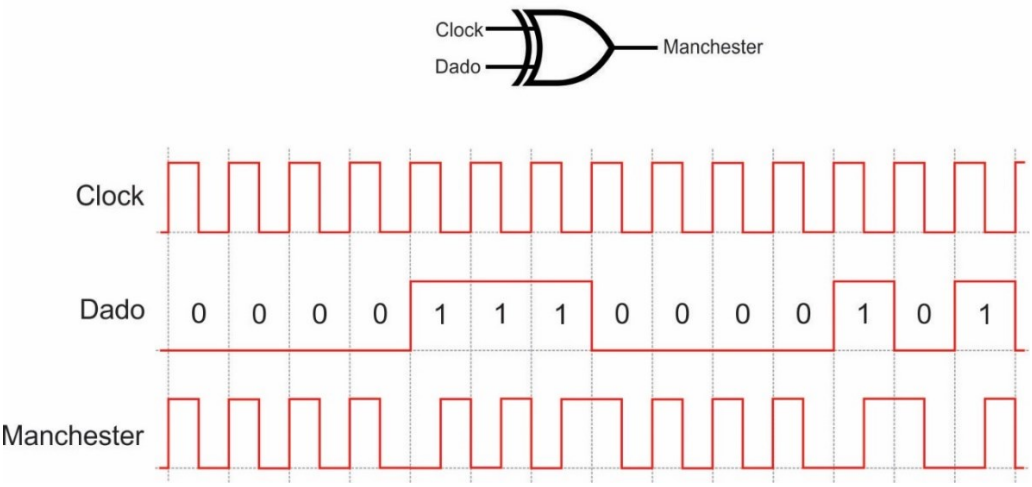
A segunda interface de comunicação está entrelaçada a limitações do chip ASI4U, como foi discutido no capítulo 2.2.5. Existem apenas duas alternativas, ou a comunicação é feita segundo AS-INTERNATIONAL ASSOCIATION (2000), seguindo a especificação com modulação APM, ou então comunicando apenas com sinal codificado no formato Manchester-II. A escolha baseou-se em dois aspectos primordiais: custos e o dimensionamento. A rede, comunicando através da modulação APM seguiria fielmente a especificação, porém seria necessário dois CI ASI4U, um para o mestre e outro para o escravo, acarretando em um aumento substancial de custos ao projeto. Já para a comunicação com o sinal Manchester-II seria necessário apenas um CI para o escravo. Como a rede que será construída terá dois escravos, suficiente para testar o protocolo, problemas como interferência eletromagnética, ruídos em geral e impedância do cabo podem ser desprezados. Assim a modulação APM do sinal no cabo não é um item vital no projeto, portanto não será necessária.

3.4.1 Codificação manchester

A transmissão de dados requer uma certa sincronização entre o transmissor e o receptor para que os dados possam ser transmitidos e detectados corretamente. Pode-se utilizar a transmissão assíncrona, enviando os dados e o *clock* separadamente, porém apenas para curtas distancias. Outro método é a transmissão síncrona, que é feito adicionando-se o *clock* aos dados, podendo atingir grandes distancias. Um desses métodos síncronos consiste na codificação manchester. A codificação do bit de dados 1 é representada pela transição de 0 para 1 durante o tempo de bit. Já a codificação do bit 0 é representada pela transição de 1 para 0 (ATMEL, 2015; KHORWAT; NAAS, 2010).

A codificação manchester pode ser verificada utilizando uma porta lógica ou exclusiva, cujas entradas são os dados e o *clock*, este último com o dobro da frequência dos dados. A saída da porta será o sinal codificado no formato manchester-II. Um exemplo disso está na figura 19 onde há a codificação manchester com um frame de requisição do mestre para troca de dados com o escravo de endereço 7 e o dado 1, conforme a mostrado na tabela 5.

Figura 19: Codificação Manchester



Fonte: Elaboração própria, 2017.

3.5 DIRETIVAS DA REDE AS-INTERFACE

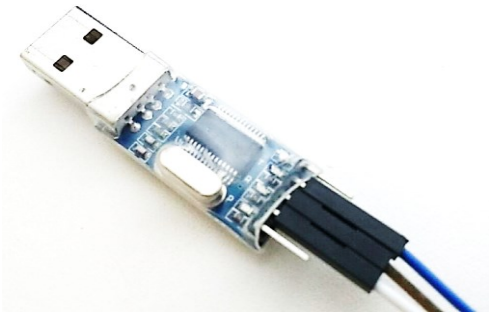
3.5.1 Interface do usuário

A interface de usuário será desenvolvida em linguagem Python versão 2.7 pois, através dela se tem mais recursos ao usuário onde, a comunicação será realizada através da porta COM do computador, sendo possível conectar a placa STM32F4DISCOVERY através de um conversor USB para serial RS-232.

O componente da interface do usuário nada mais é que uma interface entre o computador e o mestre na qual se dará os comandos por parte do usuário. Para essa finalidade será utilizado um conversor USB para serial que utiliza o chip da Profilic PL2303, visto na figura 20, para conversão de sinais USB em sinais UART com padrão RS-232, suportado pela placa STM32F4DISCOVERY. Assim sendo, é possível enviar

comandos indiretamente para a rede AS-Interface, através do teclado do computador e também receber comandos e mensagens de volta, para que o usuário possa operar e diagnosticar claramente a rede de comunicação.

Figura 20: Conversor USB para Serial com chip PL2303



Fonte: Elaboração própria, 2017.

A configuração da porta serial ficou a seguinte: velocidade de 115200 bps, dados de 8 bits, apenas 1 bit de parada, sem paridade e sem controle de fluxo.

O comando enviado através do terminal do computador deve possuir no máximo 38 bytes, no caso do comando de escrita de saídas ODI e no mínimo 7 bytes no caso de todos os comandos que não possuem argumentos. Já o comando de resposta, provindo do mestre possui no mínimo 5 bytes para os comandos que possuem apenas o *status* da requisição como resposta e no máximo 36 bytes para os comandos de leitura da lista IDI e leitura da lista LDS. A estrutura da codificação dos frames da interface de usuário pode ser vista em maiores detalhes no apêndice A.

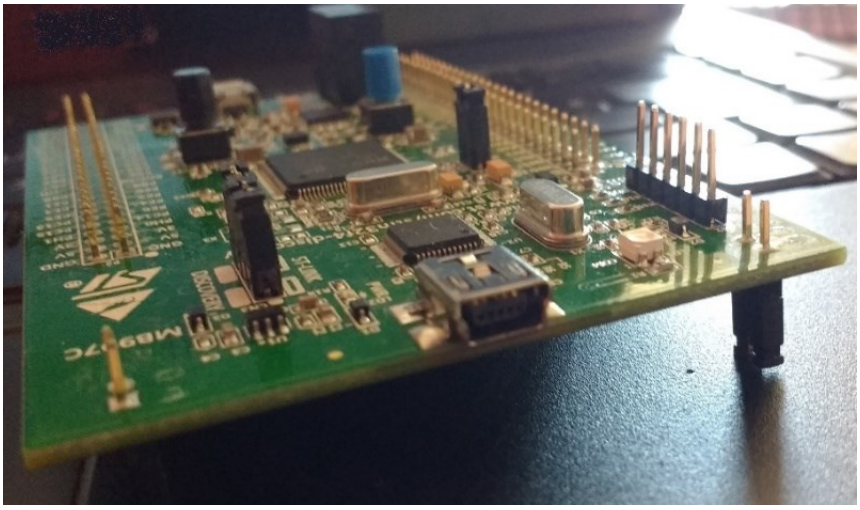
Todas as requisições são feitas de maneira assíncrona, ou seja, após o comando enviado não existe um *timeout* para a resposta, para isso se utiliza um byte dos frames sempre com o ID do comando. *Será* utilizado a técnica *byte stuffing* onde serão utilizados dois caracteres especiais, o primeiro deles usado para delimitação do frame (início e fim, representado pelo caractere 0x7E) e o segundo usado para escape (caractere 0x7D).

Após o recebimento de um comando, a interface verifica a priori a sintaxe correta do comando, se houver algum erro no comando ou nos argumentos, o comando é considerado invalido e é retornado ao usuário uma mensagem de comando inválido. Caso o comando esteja com a sintaxe correta ele é processado pela aplicação em python e enviado ao mestre via comunicação serial.

3.5.2 Mestre

O hardware do mestre será a placa STM32F4DISCOVERY, visto na figura 21, e o *firmware* que será embarcado sobre essa paca utilizará linguagem C, desenvolvido através do programa eclipse.

Figura 21: Placa STM32F4DISCOVERY



Fonte: Elaboração própria, 2017.

O maior desafio deste projeto consiste no desenvolvimento do *firmware*, que atenda as funcionalidades básicas para a comunicação de uma rede AS-Interface. Segundo GANSSLE, 2008 para desenvolver um bom *firmware* embarcado existem uma série de passos a serem realizados antes mesmo de começar a programar. De início é necessário definir pastas, controle de versão, software utilizado, depurador, compilador, *linker*, enfim uma série de passos que podem ser vistas brevemente no capítulo 3.6. Em seguida é aconselhado identificar todos os periféricos que serão utilizados no projeto, sejam eles: memórias, GPIO, conversor analógico para digital, displays, temporizadores, DMA, UART, I2C, SPI, USB e outros canais de comunicação. Para o projeto do mestre AS-Interface os periféricos a serem utilizados são: GPIO, UART (canal serial), interrupção externa e temporizadores. Com a ajuda do software STM32Cube são definidos todos os parâmetros desses periféricos onde pode ser gerado o código em C para o microcontrolador ARM da placa.

A seguir é preciso definir de acordo com nossa necessidade se haverá interrupções no programa e se sim, quais serão. A princípio será utilizado três interrupções no programa. A primeira consiste na interrupção do temporizador systick, que ocorrerá sempre em uma base de tempo de 1 ms, utilizado para diversas funcionalidades. Tem-se também o timer 6, configurado com um *prescaler* de maneira a formar um período base de 250 ns. Esse timer é habilitado como interrupção com valores diferentes de tempo para cada situação. Seu primeiro uso é no processo de transmissão, onde é definido com um período de 3 µs para codificação manchester. Logo após é configurado para 60 µs, onde, se houver a interrupção, a camada de transporte considera como *timeout* da requisição. E finalmente, caso não haja *timeout*, o timer é alterado para 8 µs durante a recepção. O valor desse timer também será utilizado para mensurar o tempo entre cada borda do frame manchester recebido do escravo e a interrupção será utilizada para detecção de erros na resposta, pois se após o início do recebimento do frame, se passar mais de 8 µs sem alguma borda de subida ou descida, isso caracteriza um erro no recebimento do frame. Já o timer 7 não possui interrupção, e tem um período de 3 µs (tempo de meio bit AS-Interface), usado para os tempos de comunicação vistos na figura 7 (pausa do mestre e pausa de envio).

A última, é a interrupção externa do pino de entrada da comunicação manchester, para identificar o início de uma resposta do escravo. Definido todos esses detalhes do projeto pode-se enfim criar um programa teste a fim de, assegurar que todos os periféricos e interrupções estão funcionando corretamente e atendendo as necessidades do projeto.

Um item crucial para o desenvolvimento do mestre está em como será implementado a codificação e decodificação manchester. Para esse problema existem duas possíveis soluções. A primeira, uma implementação em hardware, que pode ser vista em detalhes em KHORWAT et al. (2010), utilizando apenas componentes eletrônicos. A codificação seria feita através de uma porta lógica XOR e a decodificação teria 3 etapas, o primeiro estágio seria um circuito RC, o segundo seriam dois comparadores e o último um *flip flop* do tipo JK. Com esse circuito em hardware seria possível codificar e decodificar sinais em manchester, porém esse projeto teria um limite de frequência de dados de 90 kbps, sendo que o protocolo AS-Interface utiliza uma

frequência de 166.67 kbps. Devido às dificuldades em relação ao limite de frequência de operação dos componentes eletrônicos e os custos, optou-se pela segunda opção, que consiste na implementação da codificação e decodificação manchester via *firmware*, embarcada na própria placa STM32F4DISCOVERY.

Existem dois tipos diferentes de mestre no protocolo AS-Interface, o padrão que suportam até 31 escravos e, os de endereçamento estendido, que suportam até 62 escravos. Será implementado nesse projeto o mestre com suporte a endereçamento padrão. Todos os perfis de mestres foram vistos na tabela 7 e de acordo com ela, os perfis com suporte a 31 escravos apenas são M0, M1 e M2. Quanto ao suporte aos perfis analógicos, optou-se nesse projeto por não englobar esse item, pois deixaria o projeto ainda mais complexo, portanto o mestre não terá suporte nem aos perfis analógicos mais básicos S-7.1 e S-7.2. Com todas essas características, o mestre aqui implementado, terá o perfil M2 porém, com algumas pequenas diferenças, explicadas posteriormente.

A seguir, é apresentado a lista completa de todos os perfis de escravos que o mestre M2, desenvolvido nesse projeto irá suportar.

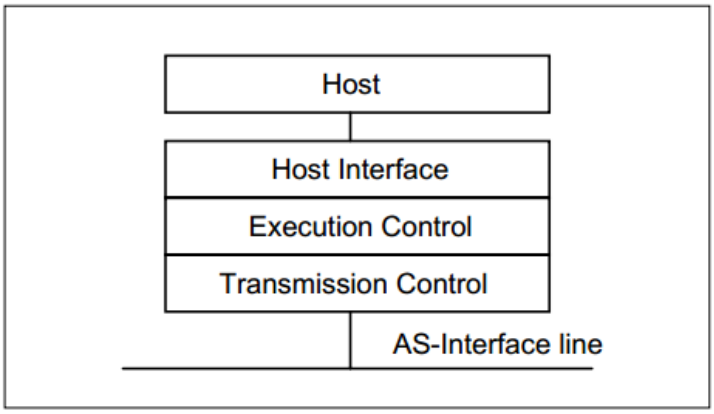
Tabela 12: Lista dos perfis de escravo suportados

Código IO	Código ID	Código ID2	Descrição
0 à E menos 9, B, D	0	x	Conexões IO binárias para sensores e atuadores
0, 3, 8	1	x	Dispositivos binários com sinal duplo
1	1	x	Sensor simples remoto com 3 entradas e 1 saida
7	0	E	4 entradas binárias + 4 saidas binárias
8	1	x	4 saidas binárias para dois atuadores duplo
B	1	x	Atuador duplo sinal com *feedback* (2 entradas + 2 saidas)
D	1	x	Atuador simples com monitoramento (1 saida + 3 entradas)

Fonte: Elaboração própria, 2017.

O mestre provê a troca de informações entre o controlador e os escravos da rede AS-Interface, através de várias camadas, o que pode ser observado na figura 22.

Figura 22: Modelo do mestre AS-Interface



Fonte: AS-INTERNATIONAL ASSOCIATION, 2000.

Existem então 3 etapas muito importantes na essência de um mestre AS-Interface. A primeira, chamada de interface *host*, representa a interface entre as informações recebidas do usuário (controlador) e a próxima camada. Neste ponto a maneira como a informação é modelada e enviada ao mestre deve obedecer um padrão, de forma a facilitar a comunicação.

Ainda na etapa de interface, é necessário definir todos os comandos válidos aceitos pelo mestre, de acordo com seu perfil e suas características de implementação. O projeto consiste no desenvolvimento de um mestre com perfil M2 com endereçamento padrão (31 escravos), contendo todas as funções obrigatórias desse perfil e mais alguns comandos opcionais, podendo ser visualizada na tabela a seguir.

Tabela 13: Lista de comandos implementados

Comando	Transferência	Opção
Leitura lista de entradas IDI	Mestre -> usuário	Obrigatório
Escrita lista de saídas ODI	Usuário -> mestre	Obrigatório
Armazenar parâmetros atuais	Mestre -> mestre	Obrigatório
Armazenar configurações atuais	Mestre -> mestre	Obrigatório
Leitura de *flags*	Mestre -> usuário	Obrigatório
Definição de modo de operação	Usuário -> mestre	Obrigatório
Leitura da lista LDS	Mestre -> usuário	Opcional
Escrita de parâmetros	Usuário -> escravo	Obrigatório
Atribuição de endereço	Usuário -> escravo	Opcional
Escrita ID-Code_1	Usuário -> escravo 0	Opcional

Apagar endereço	Usuário -> escravo	Opcional
Reiniciar escravo	Usuário -> escravo	Opcional
Leitura configuração IO	Usuário -> escravo	Opcional
Leitura ID-Code	Usuário -> escravo	Opcional
Leitura ID-Code_1	Usuário -> escravo	Opcional
Leitura ID-Code_2	Usuário -> escravo	Opcional
Leitura de status	Usuário -> escravo	Opcional
Broadcast (Resetar)	Usuário -> escravo	Opcional
Modo de programa	Usuário -> escravo 0	Opcional
Ativar canal IRD	Usuário -> escravo	Opcional

Fonte: Elaboração própria, 2017.

O próximo passo consiste no controle de execução, responsável por iniciar o mestre e o modo normal de operação, além de administrar todas as funções do sistema. Também é aqui que ocorre o escaneamento constante da rede para encontrar novos escravos ou diagnosticar erros. Para isso são necessários um conjunto de listas de dados a serem armazenados e atualizadas constantemente pelo sistema, são elas: imagem de dados de entrada (IDI), imagem de dados de saída (ODI), imagem de dados de configuração (CDI), dados permanentes de configuração (PCD), imagem de parâmetros (PI), parâmetros permanentes (PP), lista de escravos detectados (LDS), lista de escravos ativos (LAS), lista de escravos projetados (LPS) e lista de falhas de periféricos (LPF). Por aqui também passam os frames vindos da etapa anterior onde, alguns deles em específico, irão apenas ler determinada lista do mestre e retornar à interface de usuário, enquanto os frames direcionados exclusivamente aos escravos passarão para a próxima etapa de transmissão.

Certas listas precisam ser armazenadas em uma memória não volátil, as chamadas memórias permanentes. Para isso, segundo ST (2011) existem duas alternativas. A primeira é utilizar os 4 Kbytes de SRAM mantida através de uma bateria para backup quando a alimentação da placa fosse desligada, obtendo assim, alta velocidade de acesso a memória. A segunda seria criar um algoritmo para utilizar uma seção dos 1024 Kbytes de memória flash. Existe ainda uma terceira alternativa que, consiste na aquisição de uma memória EEPROM a ser conectada externamente a placa STM32F4DISCOVERY podendo comunicar através da interface I2C, porém isso demandaria maiores custos e tempo de implementação. A opção de se usar a própria

memória flash da placa é a mais atrativa dentre as outras, onde sua principal vantagem é a facilidade de escrita utilizando a camada de abstração de hardware (HAL) do fabricante ST. Porém existem duas desvantagens que para esse projeto não são tão cruciais quanto aos requisitos. A primeira consiste no tempo relativamente elevado para escrita na flash, da ordem de milissegundos até segundos, porém, como as listas armazenadas nessa memória não serão atualizadas com frequência, esse tempo não se torna um problema. A segunda, como a flash tem 1024 Kbytes e, é dividida em 11 setores diferentes com um tamanho específico, sempre que houver a reescrita de no mínimo um byte na memória, é necessário apagar todo o setor de dados já previamente armazenado, ou seja, para que funcione corretamente, é de suma importância ter variáveis temporárias alocadas na memória RAM que, seriam usadas para backup do setor da flash, e posteriormente escrita novamente na flash, o que também não consiste em um empecilho pois são apenas três listas curtas, totalizando um total de 372 bytes.

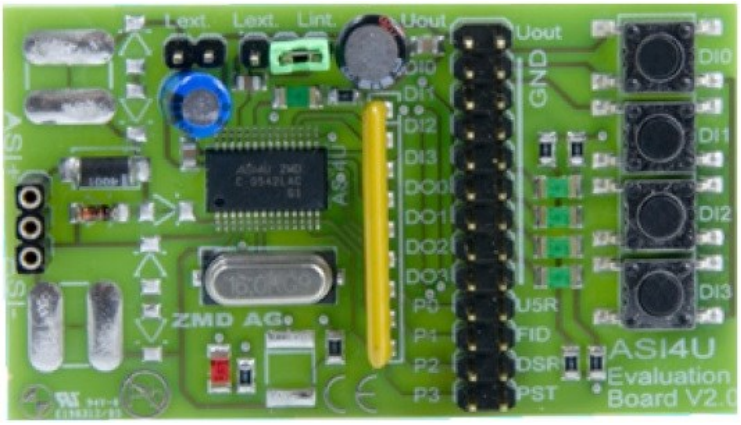
A etapa de transmissão é responsável por processar e transmitir o frame à rede AS-Interface para um determinado escravo e, receber a resposta. Podem haver dois tipos de erros nessa fase: falta de resposta do escravo, determinada por um período máximo de *timeout* de 60 µs, e falha no frame de resposta. Para isso existem dois tipos diferentes de transações: simples e múltipla. Na simples, o controle de transmissão enviará o frame ao escravo e em caso de erro não há nenhuma retransmissão da requisição do mestre, retornando imediatamente ao controle de execução. Na transação múltipla, em caso de erro na transmissão, há a retransmissão apenas uma única vez da requisição do mestre. Uma transmissão é finalizada com sucesso se houver uma resposta válida do escravo após o tempo de pausa do mestre (15 µs) e antes do *timeout* (60 µs). Também é nessa etapa que há a implementação da codificação e decodificação do sinal manchester e posterior envio e recepção do sinal elétrico.

3.5.3 Escravo

Todo o projeto e desenvolvimento do escravo se baseiam em torno do chip ASI4U que, realiza todas as operações de um escravo da rede AS-Interface. Para que o chip funcione corretamente, é de suma importância uma PCB com toda a eletrônica

necessária. Baseou-se na placa do próprio fabricante, segundo (ZMD, 2006), que pode ser vista na figura 23.

Figura 23: Placa de avaliação ASI4U V2.0



Fonte: ZMD, 2006.

Essa placa possui três modos diferentes de operação com o chip ASI4U e pode ser alterada adicionando novos componentes e alterando algumas trilhas. A primeira é o modo de aplicação padrão, que utiliza o indutor interno do próprio chip e possui limitações de corrente máxima do escravo, devido as características do indutor, de 55 mA, incluindo o consumo de corrente do chip, que é de aproximadamente 6 mA. Se o indutor for sobrecarregado com mais corrente que o especificado por mais de dois segundos, resultara no desligamento imediato de todo o CI. A segunda aplicação padrão, utiliza apenas um indutor externo, suportando correntes na placa maiores que 55 mA, onde o valor máximo será limitado pelas características elétricas do indutor utilizado. Para isso é necessário desabilitar o indutor interno, bastando apenas conectar o pino CAP do CI ao GND. O terceiro e último é o modo de alta simetria, utilizando dois indutores externos, suporta operações flutuantes entre o receptor e o transmissor em relação ao GND e também é indicado para correntes maiores que 55 mA.

A opção tomada em projeto foi de desenvolver uma PCB para o escravo utilizando o próprio indutor interno do CI, limitando assim a corrente de trabalho em 55 mA. Tal escolha foi feita visando a simplicidade do circuito que, mesmo assim atenderia as necessidades de se fazer um escravo para testes de funcionalidades da rede.

Como há o limite de consumo de corrente pelas portas de entrada e saída de 49 mA, é necessário dimensionar corretamente todos os resistores para que a soma total da corrente consumida pela placa não seja superior a esse limite. Recorrendo a IDT (2016b), para cada saída deve drenar uma corrente máxima de 10 mA. Como será visto no capítulo 3.5.4 Fonte de alimentação, a fonte de alimentação utilizada será de 30V. No chip ASI4U existem 3 níveis de alimentação diferentes. O primeiro, o U_{IN}, representa a tensão de entrada da placa, no caso 30V menos 0,7V da queda de tensão devido ao diodo, dando um total de 29,3V. O segundo é o pino U5R, onde o chip provê uma alimentação de 5V caso seja necessário, utilizado para adequar o sinal do mestre de 3,3V. E por último o U_{OUT}, que vale o valor de U_{IN} - U_{DROP} (5,5 – 6,7), no projeto foi medido e o valor de U_{DROP} apresentado foi de 5,7V, portanto U_{OUT} terá um nível de tensão de 24,3V, que será utilizado para alimentar todos os LEDs da placa e para os resistores de pull-up das entradas digitais.

De posse dos níveis de tensão, prosseguindo agora para o dimensionamento de todos os resistores da placa. Iniciando pelas saídas digitais onde, existem 3 LEDs de 3mm de cores diferentes. A queda de tensão no vermelho é de 1,7V e no verde de 2,1V. Já a queda de tensão nos amarelos é de 2V. Através de testes, foi constatado que um valor de corrente entre 4mA e 5mA seria mais que suficiente para acender os LEDs. A seguir, pode-se ver o cálculo de dimensionamento do resistor para os LEDs amarelos, assumindo uma corrente de 4,8mA.

$$U_{OUT} = I * R + 2V$$

$$R = \frac{24.3 - 2}{4.8 * 10^{-3}}$$

$$R \approx 4{,}7\ K\Omega$$

Como a queda de tensão dos LEDs é quase a mesma, será utilizado resistores de 4,7K para todos. No caso do LED vermelho e verde a corrente também é de aproximadamente 4,8mA.

Já as entradas digitais são bem simples, onde o dimensionamento se reduz apenas ao valor do resistor de pull-up, onde serão utilizados resistores de 22K, portanto a corrente em nível lógico baixo seria de aproximadamente 1 mA e em nível lógico alta menor que 10 µA.

Somando assim todas as entradas e saídas da placa, incluindo os 6 LEDs com consumo de 4,8 mA cada, e no pior caso 1 mA para cada pino de entrada, compondo um total de 8, tem-se então:

$$I_{max} = 6 * 4.8 * 10^{-3} + 8 * 10^{-3}$$
$$I_{max} = 36{,}8mA$$

Segundo ZMD (2006) existem diversas recomendações para a confecção da PCB do escravo, são elas:

- A conexão entre o cristal e o CI deve ser a menor possível para que não haja capacitâncias parasitas;
- O pino 0V e o GND devem ser conectados diretamente. Os capacitores de desacoplamento entre o pino U5R e o GND/0V devem estar o mais próximo possível dos pinos U5R e 0V;
- Capacitores de desacoplamento entre o pino Uout e GND/0V devem estar o mais perto dos pinos Uout e 0V;
- A conexão entre os pinos ASIN e 0V deve ser realizada diretamente, e não através do GND;
- Capacitor entre os pinos CAP e 0V de 47 nF;
- Capacitor entre os pinos U5R e 0V de 1 µF em paralelo com 100 nF;
- Capacitor entre os pinos Uout e 0V de 10 µF em paralelo com 100 nF.

O desenvolvimento da placa do escravo foi dividida em 4 etapas, utilizando o software Kicad. A primeira etapa foi a criação das bibliotecas de componentes necessárias, neste caso, apenas para o chip ASI4U, tanto o desenho para o esquemático, como o *footprint* e o desenho em 3D. A segunda etapa consiste na criação do esquemático da placa, contemplando todos os componentes eletrônicos, conectores e suas conexões elétricas.

A terceira fase é a mais importante pois, é onde há mais variáveis a serem analisadas e decisões a serem tomadas quanto ao projeto da placa. Ela consiste na criação do *layout* da PCB, contendo todas as trilhas, vias e furos. Foi escolhido uma placa de fibra de vidro de duas camadas, com 1 oz/ft^2 e com expessura da fibra de 1,6 mm, com dimensões de 60 mm x 50 mm. Todas as trilhas da placa possuem uma espessura

de 10 mils[4], mais do que suficiente para suportar a corrente, que será no máximo de 55 mA, esse valor de espessura levou em conta as limitações quanto a fabricação da placa. A distância mínima entre as trilhas foi definida em 10 mils. Quanto a largura das vias e diâmetro dos furos, foi adotado 36 mils e 18 mils respectivamente para trilhas de sinais e 50 mils e 25 mils para trilha de alimentação. Ainda no projeto da placa, adotou-se a criação de malhas de terra, nas duas camadas, para melhor dissipação de calor e menor interferência eletromagnética.

Será criada duas versões da PCB, a primeira onde adota-se o uso de componentes eletrônicos do tipo *through hole* pela fácil disponibilidade em lojas locais e para confecção da placa protótipo. Já a segunda versão utilizará componentes SMD, caracterizando uma placa comercial.

Finalmente a última fase consiste na criação dos arquivos de manufatura da placa, o arquivo *gerber* e de furação, onde estarão definidos num padrão próprio, todas as trilhas, vias e furos da PCB. Em posse então desses arquivos de manufatura a placa pode enfim ser confeccionada.

Como visto no capítulo 3.4, a comunicação a ser utilizada será a Manchester. O chip ASI4U possui o canal IRD, utilizando dois pinos para a comunicação, o pino IRD (receptor) e o pino LED1 (transmissor). Portanto no projeto da placa, estará disponível esses dois pinos para o uso, além é claro, de diversos outros pinos que poderão ser utilizados na placa.

Para o projeto da placa escravo da rede AS-Interface foi feita a elaboração da lista de materiais (BOM) da PCB, vista na tabela 14.

Tabela 14: Lista de matérias (BOM) da PCB do escravo

Part Number	Qtd.	Descrição	Identificador
ASI4UE-G1-ST	1	CI Interface para sensor AS-Interface	U1
VO2611-X007T	2	Acoplador óptico alta velocidade 10Mbd	U2, U3
ABM8AIG-16.000MHZ	1	Cristal 16MHz	X1
A6S-4104-H	1	Chave DIP SWITCH 4 Pos	P4
CMD15-21VGC/TR8	1	Led Verde SMD 1206	D6
CMD15-21SRC/TR8	1	Led Vermelho SMD 1206	D5

[4] Mil: Unidade de medida de comprimento bastante utilizada na confecção de PCB, onde 1 mil equivale a 0.001 polegadas.

CMD15-21VYC/TR8	4	Led Amarelo SMD 1206	D1, D2, D3, D4
CRCW08054K70FKEA	6	Resistor 0805 SMD 4.7K 1/8W 1%	R1, R2, R3, R4, R5, R6
CRCW080522K0FKEA	12	Resistor 0805 SMD 22K 1/8W 1%	R7, R8, R9, R10, R12, R13, R14, R15, R16, R17, R18, R19,
CRT0805-FZ-1002ELF	1	Resistor 0805 SMD 10K 1/8W 1%	R11
CRCW0805360RFKEA	2	Resistor 0805 SMD 360R 1/8W 1%	R20, R23
CR0805-FX-3300ELF	1	Resistor 0805 SMD 330R 1/8W 1%	R21
CRCW0805560RFKEA	1	Resistor 0805 SMD 560R 1/8W 1%	R22
08055C104KAT2A	4	Capacitor 0805 SMD 100nF 50V 10%	C3, C5, C6, C7
08055C473KAT2A	1	Capacitor 0805 SMD 47nF 50V 10%	C1
EEE-1HA010SR	1	Capacitor Eletrolítico SMD 1uF 50V 20%	C2
EEE-1HA100WR	1	Capacitor Eletrolítico SMD 10uF 50V 20%	C4
BZT52B39-E3-08	1	Diodo Zener SMD 39V 2%	D8
CGRA4001-G	1	Diodo Retificador 1A 50V	D7
BPSC-11	2	Pin HEADER 2.54mm Pitch 11 pinos	P1, P2
BPSC-2	1	Pin HEADER 2.54mm Pitch 2 pinos	P3

Fonte: Elaboração própria, 2017.

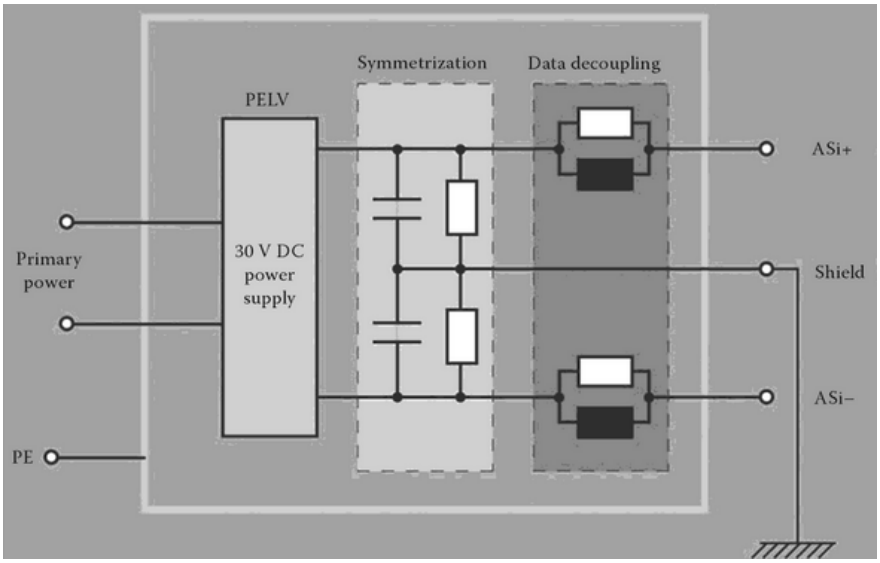
Existem dois recursos adicionais no chip ASI4U, além de todas as funcionalidades da rede AS-Interface. A primeira delas é a existência de um frame para entrar em modo de programação, visto em mais detalhes no capítulo 2.2.4. Neste modo é possível alterar o conteúdo de vários endereços da memória EEPROM do chip, configurando-o assim de acordo com nossas necessidades de projeto. O segundo recurso adicional desse chip, diz respeito aos canais de comunicação, também já visto no capítulo 2.2.5 onde, é permitido através de uma sequência de frames, ativar a comunicação do chip diretamente para a codificação manchester, que será utilizada na rede definida neste projeto. Para o funcionamento dessas funcionalidades, será criado no mestre a lógica de programação para reconhecimento e envio da sequência mágica e o controle de fluxo do modo de programação, permitindo assim, uma alta flexibilidade do escravo AS-Interface.

3.5.4 Fonte de alimentação

Em uma rede AS-Interface convencional, em que a comunicação tenha modulação APM, tem-se um cabo com dois pares condutores, que interconectam o mestre aos escravos e a fonte de alimentação da rede AS-Interface. A fonte tem um nível de sinal DC entre 29.5 V e 31.6 V e existem opções no mercado que suportam um consumo de

até 8 A. Nesse tipo de fonte, existem dois módulos adicionais para garantir a segurança e a imunidade da rede. Segundo ZURAWSKI (2015), a primeira etapa, chamada de balanceamento, é responsável por deixar a operação da rede simétrica onde a conexão de blindagem deve ser devidamente aterrada ao GND, visto que seus dois condutores de sinal trabalham com diferença de tensão e, nenhum é conectado diretamente ao GND. Já a etapa de desacoplamento de dados consiste em dois indutores de 50 mH em paralelo com resistores de 39 Ω, responsável por prevenir curto circuitos pela transmissão de dados e converter os pulsos de corrente gerados pelos transmissores. O esquemático da fonte completa pode ser visto na figura 24.

Figura 24: Esquema elétrico da fonte AS-Interface



Fonte: ZURAWSKI, 2015.

Como neste projeto, utiliza-se o sinal diretamente com a modulação manchester, não é possível haver uma fonte ligada diretamente ao enlace. Portanto não há um único cabo para levar os dados e a alimentação, e se torna necessário alimentar diretamente o mestre e também cada escravo da rede AS-Interface separadamente. O mestre, como consiste na placa STM32F4DISCOVERY, sua alimentação será fornecida por qualquer equipamento com uma porta USB, no caso um notebook. Já para os escravos da rede será necessário uma fonte de no mínimo 16V e no máximo 33V e uma corrente de pelo menos 100 mA, mais que suficiente para fornecer a potência necessária aos escravos. Não é necessário nenhum circuito de desacoplamento de dados, pois eles iram trafegar

no formato manchester entre os pinos Rx e Tx do mestre e IRD e LED1 do escravo, onde haverá apenas sinal digital. A fonte escolhida, por questões comerciais, foi uma fonte chaveada de 30V e corrente de 1A.

3.6 DIRETIVAS DE DESENVOLVIMENTO DO *FIRMWARE*

A linguagem de programação C vem crescendo bastante, principalmente quanto ao uso em sistemas embarcados em tempo real. Devido suas características de flexibilidade, suporte e potencial de portabilidade, muitos microcontroladores utilizam dessa linguagem para o seu desenvolvimento. Outra característica de suma importância para esses tipos de aplicação embarcada, consiste na essência da linguagem C ser de baixo nível, permitindo um bom suporte para entradas e saídas em alta velocidade e um tratamento eficiente quando se deseja trabalhar com bits (MIRA, 2004).

Por esses motivos e pelo amplo uso da linguagem C em sistemas embarcados e o suporte pela placa STM32F4DISCOVERY foi escolhido a linguagem C para o desenvolvimento do *firmware* do mestre AS-Interface.

Escolhida a linguagem é preciso agora de um sistema de controle de versão para o programa. O sistema escolhido foi o Git, responsável por fazer esse controle de arquivos, permitindo alterações simultâneas, criação de *branches,* cópias de segurança, vistos através da linha do tempo do repositório em todos os *commits* realizados. Tudo isso traz mais segurança, praticidade e agilidade no desenvolvimento de um programa. O Bitbucket é um serviço web que oferece diversas funcionalidades aplicadas ao git, hospedando seu código na nuvem e, também permite a criação de um repositório local na própria máquina, através do software SourceTree. Essas ferramentas foram utilizadas durante todo o desenvolvimento do *firmware*.

O compilador bastante utilizado para linguagem C é o GCC, neste caso, específico para microcontroladores ARM, o GNU ARM *Embedded Toolchain*. Em conjunto com a ferramenta GDB *debugger* é possível através do eclipse programar o código, compilar, fazer download na placa e depurar todo o programa.

Para facilitar a programação, não precisando acessar e trabalhar diretamente com os registradores do microcontrolador ARM é possível utilizar a interface CMSIS, contendo diversas definições desenvolvida pela própria ARM, que permitem facilmente o acesso

do processador aos periféricos e a sistemas operacionais em tempo real. Entretanto existe outra especificação para programar a placa STM32F4DISCOVERY que se torna bastante útil no desenvolvimento, essa por sua vez, é fornecida pelo próprio fabricante, a ST, responsável pela criação da interface HAL, que fornece um conjunto simples e genérico de interface de programação de aplicação que interage com a camada superior (aplicação, bibliotecas e pilhas). Munidos desses dois recursos, é possível agora programar todas as funcionalidades do escravo AS-Interface, utilizando todos os periféricos necessários na placa.

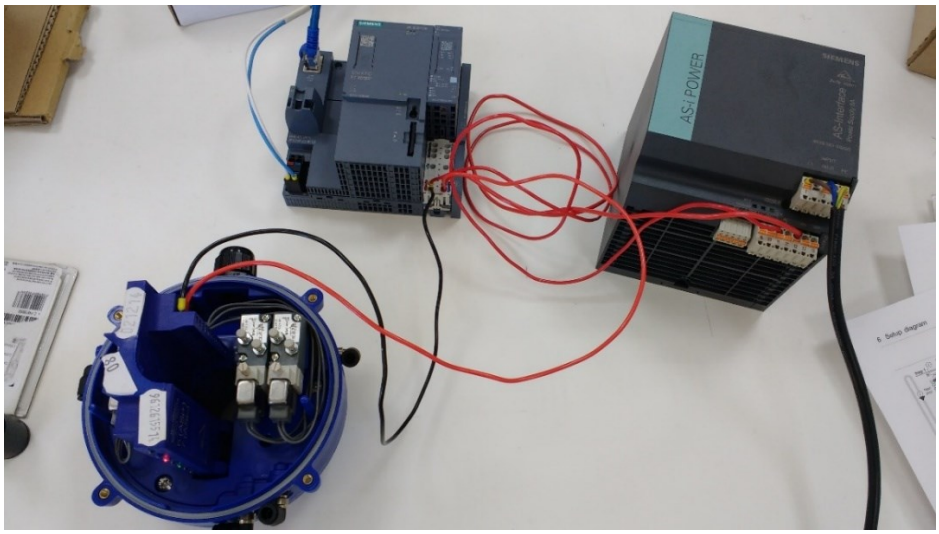
4 DESENVOLVIMENTO DA REDE AS-INTERFACE

4.1 TESTES EM EQUIPAMENTOS INDUSTRIAIS

A primeira etapa do projeto foi testar uma rede AS-Interface utilizando os componentes da Siemens e AlfaLaval. O primeiro dispositivo é o escravo AS-Interface da ThinkTop do fabricante AlfaLaval que é utilizado em conjunto com uma válvula para processos alimentícios. O mestre da rede é o módulo CM AS-Interface, acoplado diretamente ao CLP, SIMATIC ET 200SP com CPU S7-1500, que por sua vez é o controlador da rede. Existe também uma interface de usuário entre o CLP e o computador, através do software TIA Portal, capaz de diagnosticar, alterar valores de saída do escravo, leituras de entrada, etc. Para finalizar, a fonte de alimentação utilizada foi a da Siemens, modelo AS-Interface 30 V que suporta correntes de até 8 A e, que possui o módulo de desacoplamento de dados embutido.

Com esse teste utilizando equipamentos industriais, foi possível aprender um pouco mais sobre o funcionamento da rede AS-Interface, verificar diversas funcionalidades como por exemplo, a escrita dos códigos de perfis do escravo através do software, endereçamento automático dos escravos e a conexão em geral da rede.

Figura 25: Exemplo de rede AS-Interface industrial



Fonte: Elaboração própria, 2017.

4.2 INTERFACE DO USUÁRIO - APLICAÇÃO

A primeira interface, entre o computador e o mestre, permite ao usuário dar comandos na rede, direcionados aos escravos (telegramas AS-Interface padrão), e receber mensagens de status e erro, incluindo também os comandos para ativação do canal IRD e para programação da memória EEPROM do chip ASI4U. Também existem comandos com funções especiais, para troca de informações apenas entre o mestre e o usuário, como por exemplo para ler determinada lista dos escravos conectados à rede.

Na figura 26 podemos visualizar todos os comandos que podem ser processados pela aplicação do usuário e enviados ao mestre pela serial, através da interface utilizando o *prompt* de comando do windows.

Figura 26: Lista de comandos da interface de usuário

```
Documented commands (type help <topic>):
========================================
active_ird      quit        read_idi       set_operation_mode   write_param
address_assign  read_flags  read_io        store_actual_config
broadcast       read_id     read_lds_list  store_actual_param
delete_addr     read_id_1   read_status    write_id_code1
program_mode    read_id_2   reset_slave    write_odi
```

Fonte: Elaboração própria, 2017.

A título de demonstração, na figura 27, temos dois testes na interface, o primeiro de um comando de leitura de configuração IO para o escravo com endereço 12 da rede onde, nesse caso retornou o valor 7. O segundo comando é de leitura das *flags*, onde se pode observar a seguir o valor delas.

Figura 27: Exemplo de comandos na interface do usuário

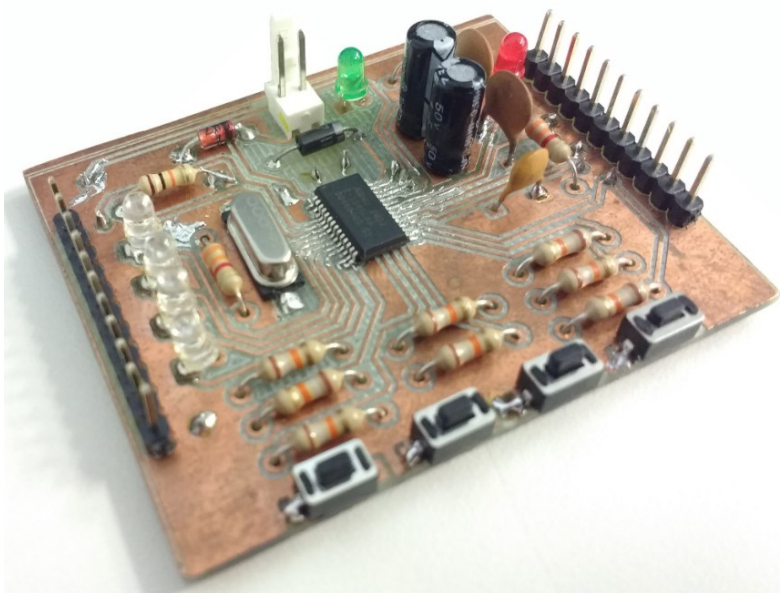
```
>> read_io 12 0
>> [CMD_READ_IO_CONFIG]  070001070241
STATUS: OK
IO CODE: 07

>> read_flags
>> [CMD_READ_FLAGS]  13000901000000001000000BA7D
STATUS: OK
Config_OK: 01
LDS.0: 00
Auto_addr_assign: 00
Auto_addr_available: 00
Configuration_active: 00
Normal_operation_active: 01
APF/not APO: 00
Offline_ready: 00
Periphery_OK: 00
```

Fonte: Elaboração própria, 2017.

4.3 ESCRAVO

Como já visto o software para o desenvolvimento da placa escravo utilizando o chip ASI4U foi o Kicad. É importante ressaltar que para o projeto da placa do escravo AS-Interface foram implementados duas versões, a V1.0 e a V1.1. A primeira versão de protótipo foi construída com componentes discretos e simples do tipo PTH. Já a segunda versão foi desenvolvida com todas as correções da placa anterior de protótipo e visando custo de produção e características comerciais da PCB, consistindo assim, em um produto com qualidade comerciável, utilizando componentes SMD. Na figura 28, pode-se observar a PCB V1.0 para protótipo e, a seguir, será descrito, passo a passo, todo o desenvolvimento da versão 1.1 da placa do escravo.

Figura 28: Placa escravo AS-Interface V1.0



Fonte: Elaboração própria, 2017.

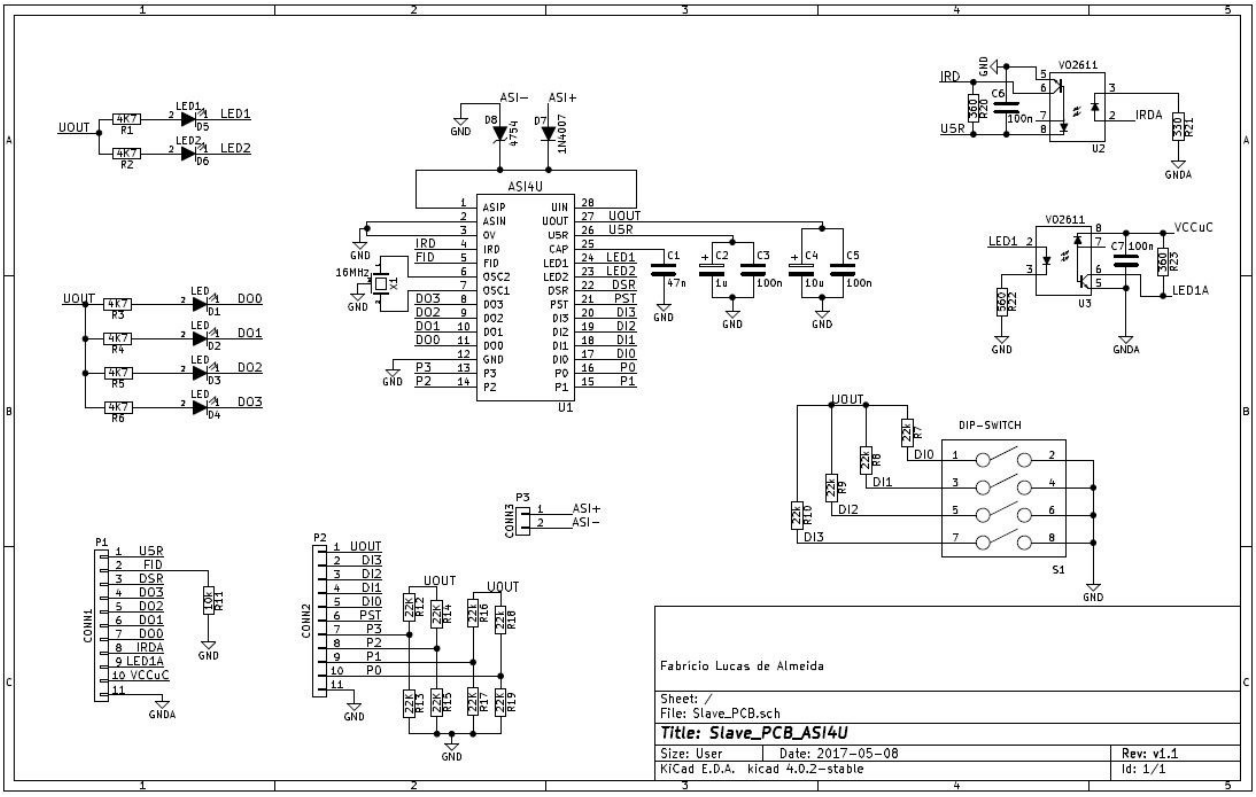
A primeira etapa do desenvolvimento de qualquer PCB é a escolha de todos os componentes a serem utilizados, definindo assim o *package* de cada um. Foi adotado o uso de componentes SMD devido a seu tamanho reduzido, facilidade de montagem e, se comparado aos componentes PTH, muitos já possuem preço reduzido. Todos os resistores e capacitores da PCB foram adotados com tamanho 0805, já os LEDs são 1206 para melhor visualização e, os outros componentes, cada um com seus aspectos construitivos.

O segundo passo é a criação de todas as bibliotecas dos componentes eletrônicos usados na PCB, conforme a tabela 14, como por exemplo, a criação do desenho esquemático e *footprint* do CI ASI4U, SSOP com 28 pinos, seguindo IDT (2016b). Outros componentes como a chave DIP e os opto acopladores também necessitavam da criação de suas respectivas bibliotecas. Outros componentes mais comuns como resistores, capacitores e LEDs, já possuem suas respectivas bibliotecas padrões no próprio software Kicad.

O próximo passo foi a criação do esquemático da placa. Para os sensores e atuadores da placa, foram utilizados 4 LEDs de baixa corrente e uma chave do tipo *DIP-Switch*, para assim, ser facilmente capaz de testar as entradas e saídas do escravo.

Quanto aos pinos disponível na placa, tem-se todos os 4 pinos de entrada, os 4 de saída e 4 de parâmetros, além das tensões provenientes do chip, GND, pinos de sinalização e os pinos para a comunicação manchester, todos estes, disponíveis na barra de pinos. Os pinos de parâmetros possuem dois resistores cada, um de *pull-up* e outro de *pull-down*, pois a semântica dos pinos de parâmetros varia entre os perfis de escravos e em alguns casos os valores são invertidos, por esse motivo o projeto contempla as duas possibilidades para atender todos os perfis de escravo, mas vale ressaltar que, para cada par, por exemplo, R12 e R13, apenas um deles deve ser montado na placa. O restante dos componentes eletrônicos e conexões seguiram as recomendações de ZMD (2006). Após a ligação elétrica de todos os componentes da placa, é necessário então, selecionar componente por componente, de acordo com os itens comerciais, definindo cada *footprint* que será importado durante o processo de projeto do layout da PCB. O esquemático completo pode ser visto na figura 29.

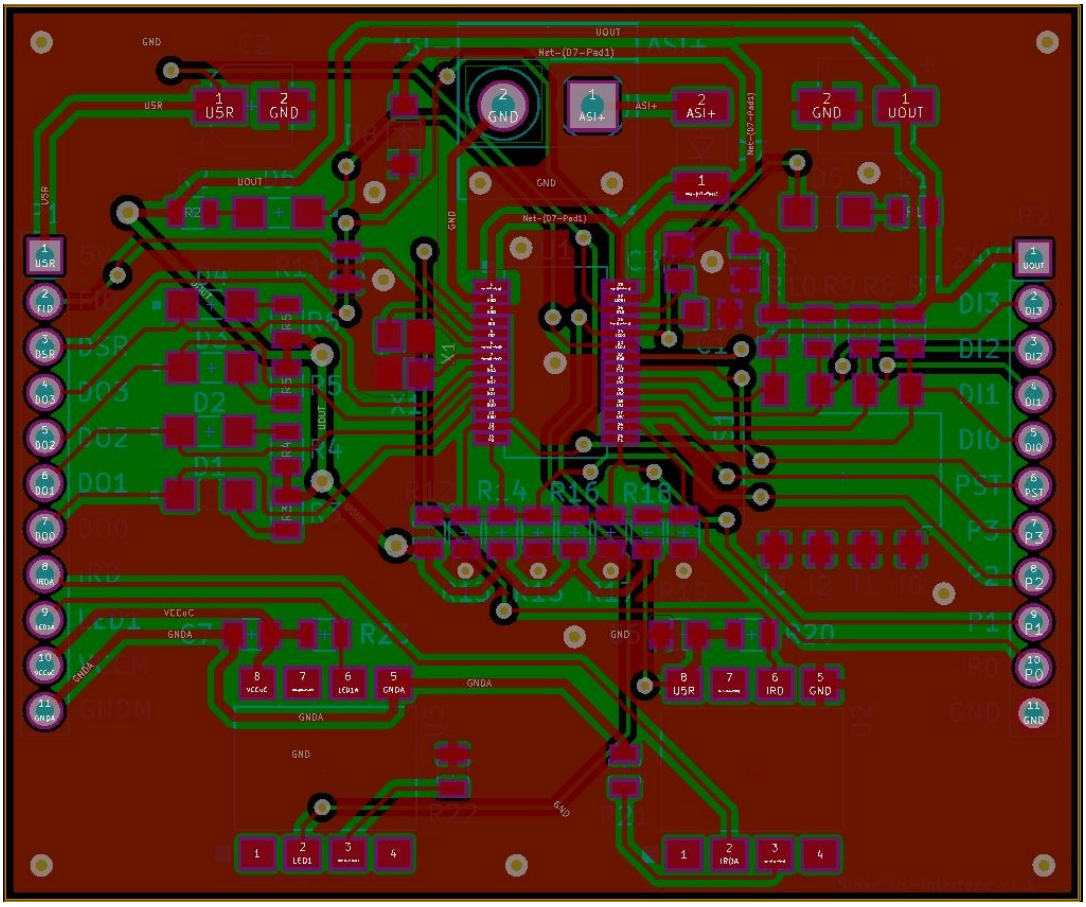
Figura 29: Esquemático da placa escravo AS-Interface



Fonte: Elaboração própria, 2017.

Partindo agora para o desenvolvimento do *layout* da PCB, o primeiro item é a definição dos parâmetros de projeto. Foi utilizado uma placa com duas camadas, 1.6mm de espessura, 1 oz, FR-4 e componentes apenas na camada superior. A dimensão da placa é de 60 mm x 50 mm e todas as trilhas foram definidas com uma espessura de 15 mils, suficiente para a passagem de corrente da placa. Após várias tentativas e correções, foi realizado uma boa disposição dos componentes ao longo da placa, de forma que houvesse o menor número possível de vias, os componentes mais críticos, como o cristal e os capacitores de desacoplamento estivessem o mais próximo possível do chip ASI4U. Fez-se o roteamento completo da placa com base nas conexões criadas no esquemático, utilizando vias de 36 mils e furos de 18 mils para os sinais e vias de 50 mils por 25 mils para a alimentação da placa. Também foi feita a malha de terra nas duas camadas de cobre da placa. Todo o *layout* da PCB pode ser visto na figura 30.

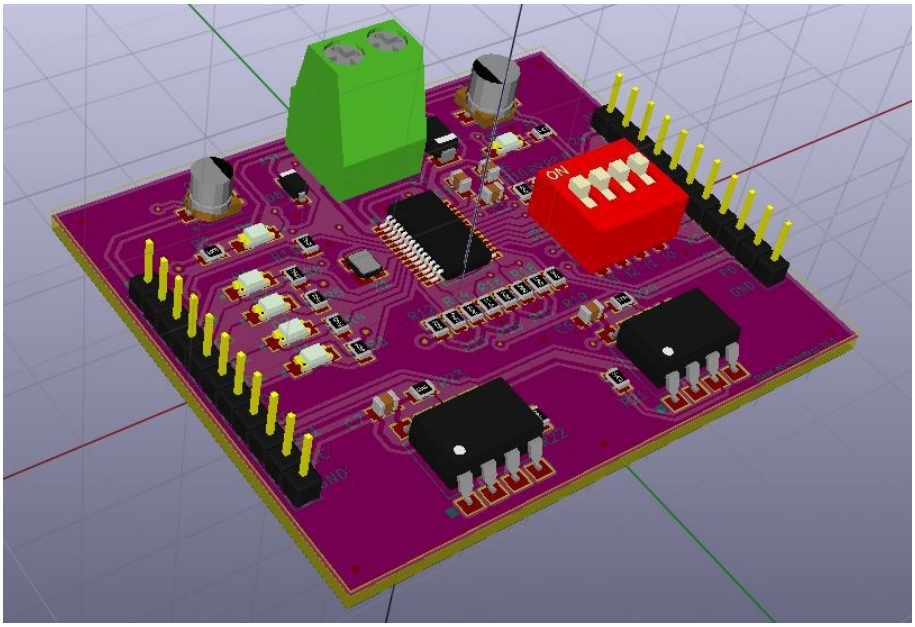
Figura 30: Layout da placa escravo AS-Interface



Fonte: Elaboração própria, 2017.

Depois de todo o projeto da placa houve a criação de alguns componentes em 3D através do software wings 3D e, aquisição de outros disponíveis no meio eletrônico. Após reunir todos os componentes em 3D e ajustar suas dimensões, é possível visualizar através do kicad o desenho em 3D da placa escravo AS-Interface, vista na figura 31.

Figura 31: Placa 3D escravo AS-Interface



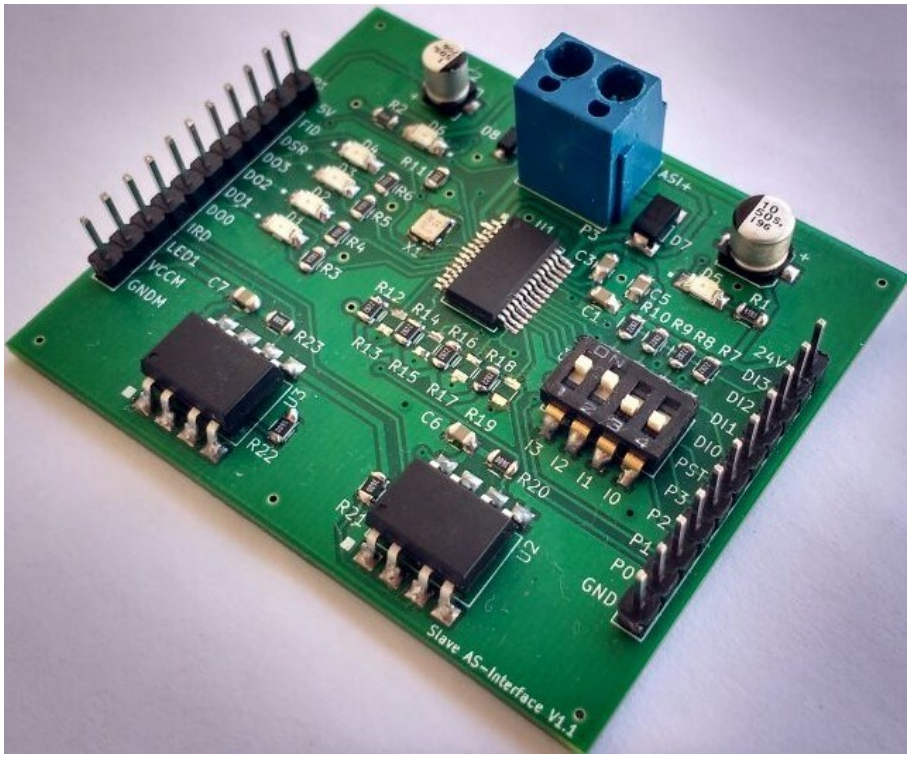
Fonte: Elaboração própria, 2017

A próxima etapa é a criação dos arquivos de manufatura da placa, o arquivo gerber e de furação, onde estarão definidos num padrão próprio de coordenadas, todas as trilhas, vias e furos da placa. De posse desses arquivos pode-se enfim confeccionar a PCB.

Para confecção da PCB existem diversas opções, ela pode ser feita através da impressão do *layout* em uma folha e transferido termicamente para a placa de cobre, ou então transferir através de tinta foto sensível e soluções específicas. Posteriormente pode-se corroer o cobre, apenas onde não há tinta, com solução de percloreto de ferro ou ácido clorídrico junto a água oxigenada. Outra alternativa a esses métodos é de usar máquinas fresadores, de alta precisão, para que possam fresar toda a parte da placa que, de acordo com o *layout* não possui cobre, além de perfazer toda a furação da PCB. Para este projeto a confecção da PCB foi feita comercialmente através de um fabricante na China.

Após a confecção da placa, resta agora soldar todos os componentes eletrônicos e conectores, listados na tabela 14, diretamente na face superior da placa. Enfim finalizada a PCB do escravo AS-Interface com o chip ASI4U pode ser vista na figura 32

Figura 32: Placa escravo AS-Interface V1.1



Fonte: Elaboração própria, 2017.

Para efeitos comerciais, será realizado um comparativo breve dos custos entre a placa de testes do fabricante IDT, vista na figura 23 e a placa de testes desenvolvida neste projeto, figura 32. A Placa de teste ASI4UEVBV2P1 pode ser encontrada no fornecedor Avnet por $31,18. Para a PCB escravo desenvolvida, considerando o custo em dólares dos componentes do fornecedor mouser e o custo de confecção da PCB no fornecedor seedStudio, tem-se um valor total por placa de aproximadamente $25,00 dólares, apenas de material. Vale salientar que todos esses custos, para as duas placas, não consideram taxas de importação e fretes para o país de destino.

4.4 MESTRE

Como já referenciado, o mestre será desenvolvido encima da placa STM32F4DISCOVERY utilizando a linguagem C como ferramenta de construção do firmware. O programa foi desenvolvido com estruturação em módulos, um deles é a camada de abstração da ST, a HAL, onde, por exemplo no caso da utilização da UART, todos os detalhes da arquitetura do microcontrolador como registros, endereços ficam suprimidos da camada de abstração e o usuário tem acesso as funções prontas de leitura e escrita da serial. Outros módulos foram desenvolvidos para processar uma série de funções específicas, onde os principais módulos do *firmware* podem ser vistos a seguir:

- *Mon*: Nesse módulo está toda a implementação que trata a serial UART, tanto envio quanto resposta, onde os frames provindos da interface de usuário são armazenados em uma fila;
- *Util_cbuf*: Módulo responsável por tratar a fila do tipo FIFO (*first in first out*), onde são armazenados os comandos vindos da interface de usuário;
- *RW_Flash*: Módulo responsável pela inicialização e escrita no setor 1 da memória flash, armazenando algumas listas de variáveis;
- Crc16: Módulo responsável pelo cálculo do CRC de 16 bits, usado para checar o erro nos comandos da interface de usuário e variáveis da memória flash;
- *Asi_hal*: Aqui são definidos todas as funções que dependem da camada do HAL, como ativação, desativação e chamadas de interrupções externa e de temporizadores;
- *Asi*: Esse módulo compõe toda a parte do tratamento do frame e sinal elétrico dos comandos AS-Interface, incluindo montagem do frame, codificação manchester, recebimento da resposta, decodificação, tratamento dos temporizadores, estouro do *timeout*, validação do frame recebido e envio do status da transação da camada de transporte;
- *Asi_sm*: Módulo que envolve toda a máquina de estados do mestre, principalmente o controle de execução, com todas suas etapas, tratamento dos frames, incluindo as interfaces para o usuário e para o controle de transmissão, controle do ciclo da rede, etc.

A placa STM32F4DISCOVERY possui quatro LEDs embutidos, LEDs estes utilizados no mestre para indicação de estados e falhas, conforme pode ser observado na figura 33.

Figura 33: LEDs de status do mestre

Fonte: Elaboração própria, 2017.

O LED laranja sinaliza que o controle de execução está nas operações de inicialização (fases *off-line*, detecção e ativação). Já o LED azul indica que o mestre está processando as operações normais (fases troca de dados, gerenciamento e inclusão). O LED verde indica em qual modo de operação o mestre está trabalhando (modo de configuração – LIGADO, modo protegido - DESLIGADO). E por fim, o LED vermelho sinaliza se houve alguma falha elétrica na rede AS-Interface, como por exemplo, tensão muito baixa ou algum curto circuito.

4.3.1 Interface do usuário – mestre

Como citado anteriormente, o módulo *"mon"* do programa é responsável por gerenciar os comandos provenientes da aplicação python, decodificar esses frames e repassar para o controle de execução o comando a ser executado na fase de gerenciamento. Posteriormente, ele também é responsável pelo sentido inverso onde, o controle de execução após processado o frame retorna a reposta para a interface do usuário através da serial. Esse módulo também é responsável por checar o CRC do frame recebido assim como calcular e gerar o CRC do frame de resposta afim de, conferir a integridade do frame.

4.3.2 Funcionalidades para o chip ASI4U

Existem algumas funções para uso exclusivo do chip ASI4U e não necessariamente ao protocolo AS-Interface. A primeira delas e mais simples consiste na ativação do canal IRD para comunicação manchester entre o mestre e os escravos, conforme visto no capítulo 2.2.5. Para implementação dessa função é necessário enviar 4 frames AS-Interface quaisquer, foi escolhido um frame para leitura de status (RDST), com endereço 1. Assim basta ter uma estrutura de 4 repetições onde o telegrama é enviado, espera-se o fim do envio e o programa aguarda por 1 ms para enviar novamente o próximo frame, conforme observado na figura 34, garantindo assim, a ativação do canal IRD.

Figura 34: Código em linguagem C para ativação do canal IRD.

```
void asi_magic_sequence(asi_cmd_t *cmd)
{
    cmd->id = ASI_CMD_MAGC;
    cmd->address = 1;
    asi_tx_process_frame(cmd);

    for(uint8_t i = 0; i < 4; i++)
    {
        asi_tx_start();
        __WAIT_COMMUNICATION(asi_tx_transmitted);
        HAL_Delay(1);
    }
}
```

Fonte: Elaboração própria, 2017.

A segunda função especial consiste no modo de programa, onde o usuário pode configurar de acordo com sua aplicação os valores da memória EEPROM do chip ASI4U. Alguns detalhes importantes estão citados em ZMD, 2006, onde para acessar o modo de programa, o chip deve estar configurado como escravo, possuir endereço zero e ser o único chip conectado à rede. Todos os endereços de memória e respectivos conteúdos estão explicitados na tabela 10. Para escrever algum valor na EEPROM, depois de estar em modo de programa, basta utilizar o comando padrão: escrita de parâmetros, onde o campo endereço representa o endereço da memória EEPROM, utilizando apenas 4 bits, e o campo dado, com o valor a ser escrito. Se for necessário fazer a leitura de algum endereço da memória, pode ser feito através do comando de troca de dados. O mestre

só sai do modo de programa e volta a executar suas funções normais, assim como o escravo, quando o usuário enviar o comando “reiniciar escravo”, ou se houver reenergização do sistema.

4.3.3 Listas locais e *flags*

Para o correto funcionamento do controle de execução do mestre é necessário diversas listas, como já citados no capítulo 3.5.2. A seguir é possível visualizar em detalhes a estrutura de listas na memória RAM, criadas no código C e, assim como já citado, é necessário ter variáveis temporárias para backup das listas que estão na flash, pois toda vez que houver a reescrita de uma determinada posição da flash, todo o setor de memória deve ser apagado e reescrito novamente.

Figura 35: Estrutura de dados das listas

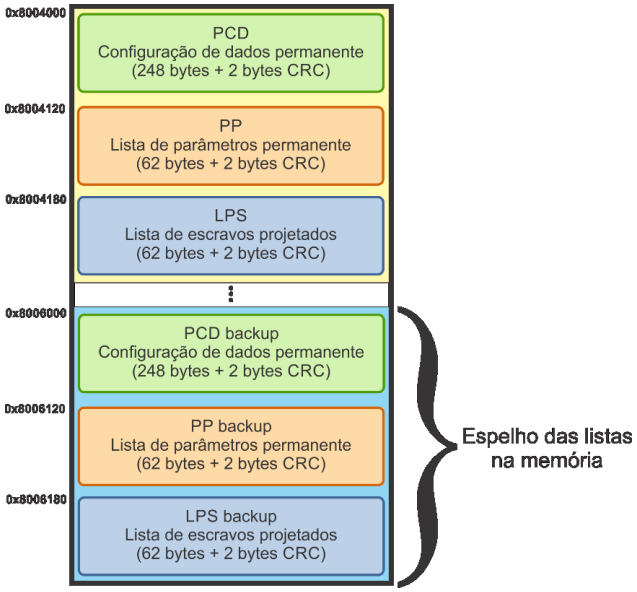
```
typedef struct config_list
{
    uint8_t IOcode;
    uint8_t IDcode;
    uint8_t ID1code;
    uint8_t ID2code;
} config_list_t;

typedef struct local_lists_s
{
    uint8_t ODI[63];
    uint8_t IDI[63];
    uint8_t PI[63];
    uint8_t PP_temp[63];
    config_list_t CDI[63];
    config_list_t PCD_temp[63];
    uint8_t LDS[63];
    uint8_t LAS[63];
    uint8_t LPF[63];
    uint8_t LPS_temp[63];
} local_lists_t;
```

Fonte: Elaboração própria, 2017.

Segundo AS-INTERNATIONAL ASSOCIATION (2002), existem três listas que devem ser não voláteis, são elas: parâmetros permanente (PP), configuração de dados permanente (PCD) e lista de escravos projetados (LPS). Na placa STM32F4DISCOVERY elas serão armazenadas no setor 1 da memória flash que possui tamanho de 16 Kbytes e se inicia no endereço 0x8004000, conforme a figura 36. Foi adotado por questões de

segurança, caso alguma variável esteja corrompida, o cálculo do CRC daquela lista que ficará também armazenado na flash. As três listas somam um total de 372 bytes na memória mais 372 bytes do espelho dessas listas, ou seja, caso a lista principal esteja com algum valor corrompido, checado através do cálculo do CRC o programa passa a usar a lista de backup, caso esta esteja integra, senão aquela lista é apagada da memória. Para novos mestres os valores padrão da lista PP e PCD devem ser todos os bits 1, já para a lista LPS todos os bits devem ser 0.

Figura 36: Arquitetura das listas na memória flash



Fonte: Elaboração própria, 2017.

Outro dado de suma importância para o mestre são as *flags* que, sinalizam diversos estados durante a execução, permitindo também ao usuário, controlar o mestre de uma melhor forma. A seguir é possível visualizar todas as *flags* presentes no *firmware* do mestre AS-Interface.

Figura 37: Estrutura de dados das *flags*

```
typedef enum asi_state_e
{
    ASI_PROTECTED_MODE = 0,
    ASI_CONFIGURATION_MODE,
} asi_master_mode_t;

typedef struct flags_s
{
    bool config_OK;
    bool LDS_0;
    bool auto_address_assign;
    bool auto_address_avaiable;
    bool configuration_active;
    bool normal_operation_active;
    bool power_fail;
    bool offline_ready;
    bool periphery_OK;
    bool data_exchange_active;
    bool offline;
    bool status_frame_ok;
    asi_master_mode_t asi_mode;
} flags_t;
```

Fonte: Elaboração própria, 2017.

4.3.4 Controle de execução

Antes de aprofundar no controle de execução do mestre, é preciso antes explanar acerca dos modos de operação do mestre AS-Interface. Existem dois modos, o modo de configuração e o modo protegido. O modo de configuração pode ser usado para o comissionamento de uma rede AS-Interface, onde nesse caso, ainda não foi realizada a configuração total da rede. E o segundo, o modo protegido onde o mestre apenas troca informações com escravos AS-Interface configurados, caso a configuração de um escravo seja alterada é necessário utilizar o comando “armazenar configurações atuais”, para que haja o armazenamento da nova configuração (SIEMENS, 2015).

Vale ressaltar que, durante a inicialização do mestre, automaticamente inicia-se em modo protegido. Caso seja necessário alterar o modo, basta usar o comando “definição de modo de operação”.

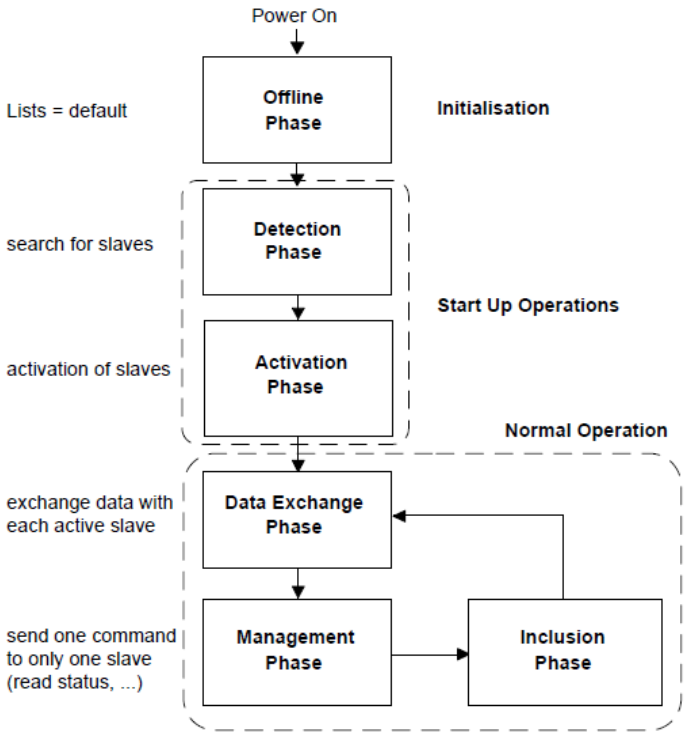
O controle de execução é composto de dois processos:

- Procedimentos de inicialização e operação normal para comunicação com os escravos;
- Funções de administração que são responsáveis por administrar o sistema AS-Interface fazendo a ponte entre a interface de usuário e o controle de execução.

Durante as fases de transmissão de dados o controle de execução envia requisições para o controle de transmissão que, envia fisicamente o frame aos escravos.

Logo após a inicialização do mestre ele inicializa todas as listas de variáveis na fase off-line. Depois, na fase de detecção faz-se a varredura em todos os endereços válidos (1 a 31) para pesquisar novos escravos conectados à rede. Então a fase de ativação ativa os escravos conectados dependendo do modo de operação definido (modo protegido ou modo de configuração). No caso do modo protegido os escravos detectados e pré configurados (códigos de configuração e identificação) serão ativados pelo mestre. Já no modo de configuração todos os escravos detectados serão ativados. Agora sim, se inicia a operação normal onde, há de início a troca de dados com todos os escravos ativados e no final de cada ciclo é enviado um comando, ditado pela função de administração, para um escravo em específico e depois é enviado um comando para um endereço livre em busca de posteriores escravos conectados à rede. Caso não haja resposta a máquina de estados retorna para a fase de troca de dados, porém caso haja uma resposta naquele endereço o escravo será ativado dependendo do modo de operação.

Figura 38: Fases da máquina de estados do mestre

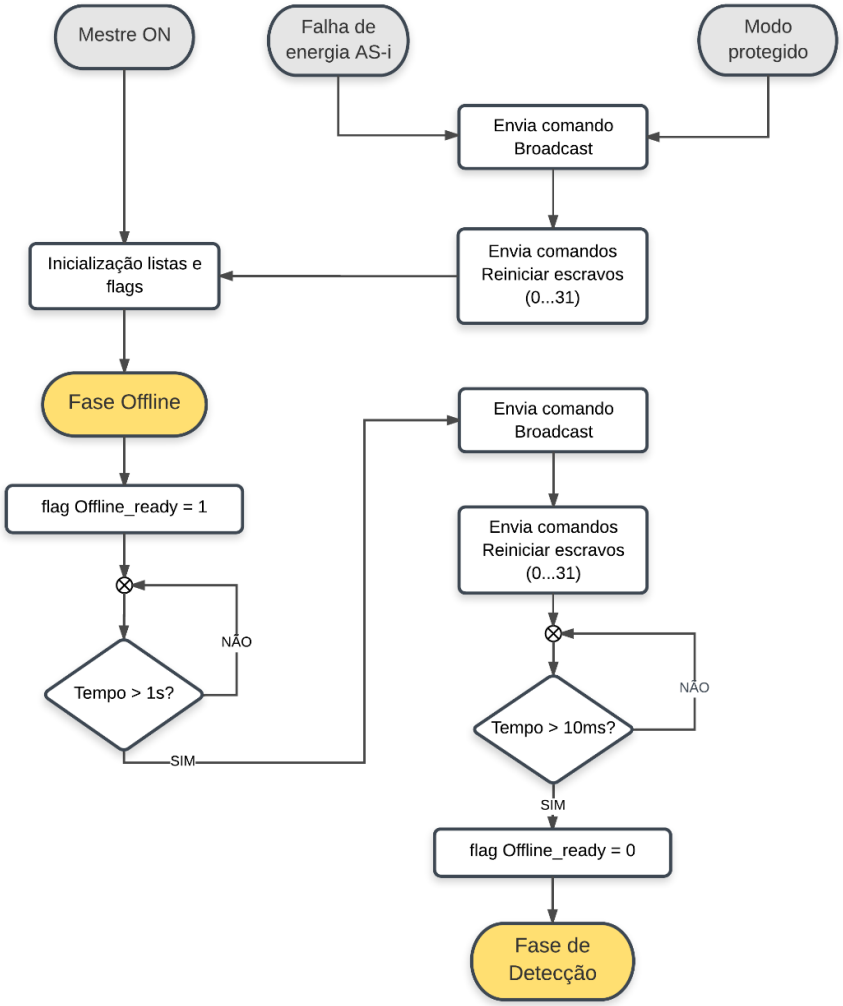


Fonte: AS-INTERNATIONAL ASSOCIATION, 2000.

Para todo esse processo são necessários um conjunto de listas e *flags* de dados a serem armazenados e atualizadas constantemente pelo sistema, conforme já citadas no capítulo 4.3.3.

A primeira fase da máquina de estados é a *off-line*, ela é chamada sempre que há a energização do sistema, se houver alguma falha na energia ou se for realizado a troca do modo de operação, passando do modo de configuração para o protegido. Nessa fase ocorre a inicialização das listas e *flags*, um atraso de 1 segundo para que todos os escravos se estabilizem e posteriormente o envio do comando "*broadcast*" e do comando "reiniciar escravos" para todos os endereços válidos, a fim de reiniciar todos os escravos da rede. Essa fase pode ser vista em detalhes na figura 39.

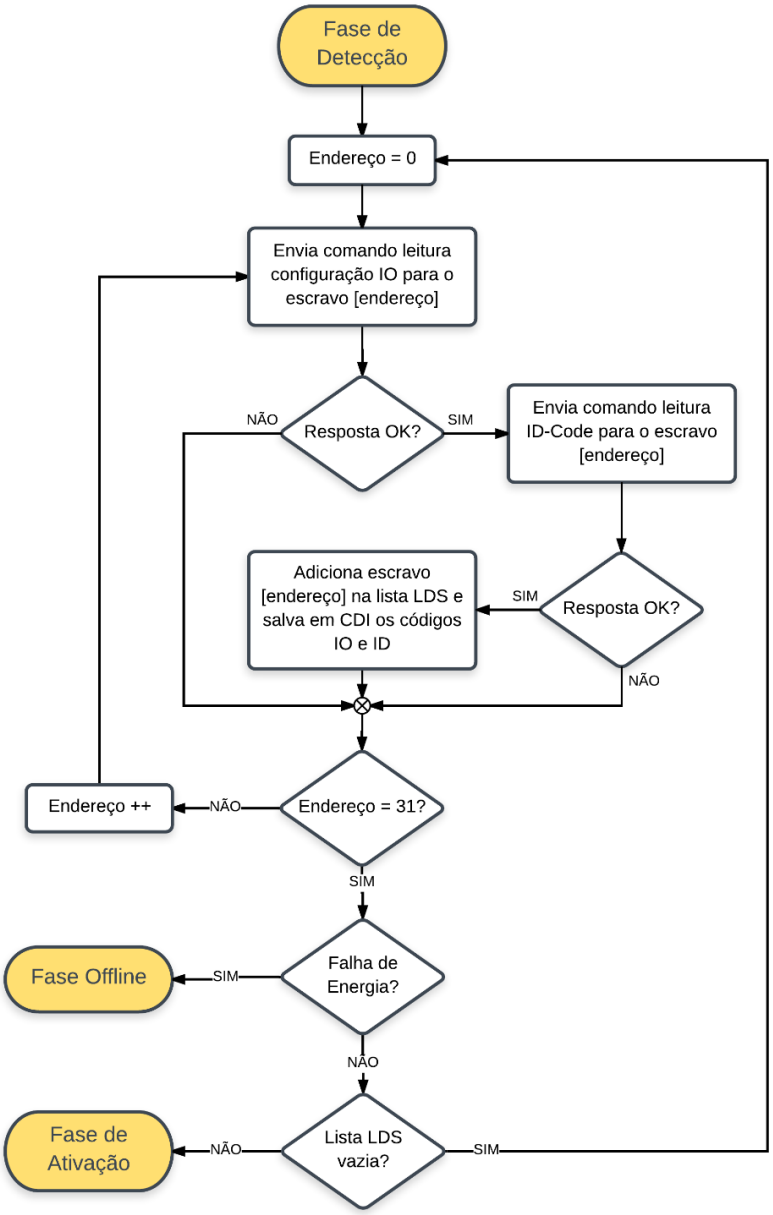
Figura 39: Fluxograma da fase offline



Fonte: Elaboração própria, 2017.

A segunda fase da máquina de estados é a fase de detecção, onde há o envio dos comandos “leitura configuração IO” e “leitura ID-code” para todos os endereços válidos de escravos na rede. Caso haja resposta naquele endereço, o mestre interpreta como detectado aquele escravo, o adicionando na lista LDS e CDI. Um detalhe importante é que se nenhum escravo for detectado na rede, o mestre fica executando infinitamente esta fase até que haja no mínimo um escravo ligado à rede AS-Interface.

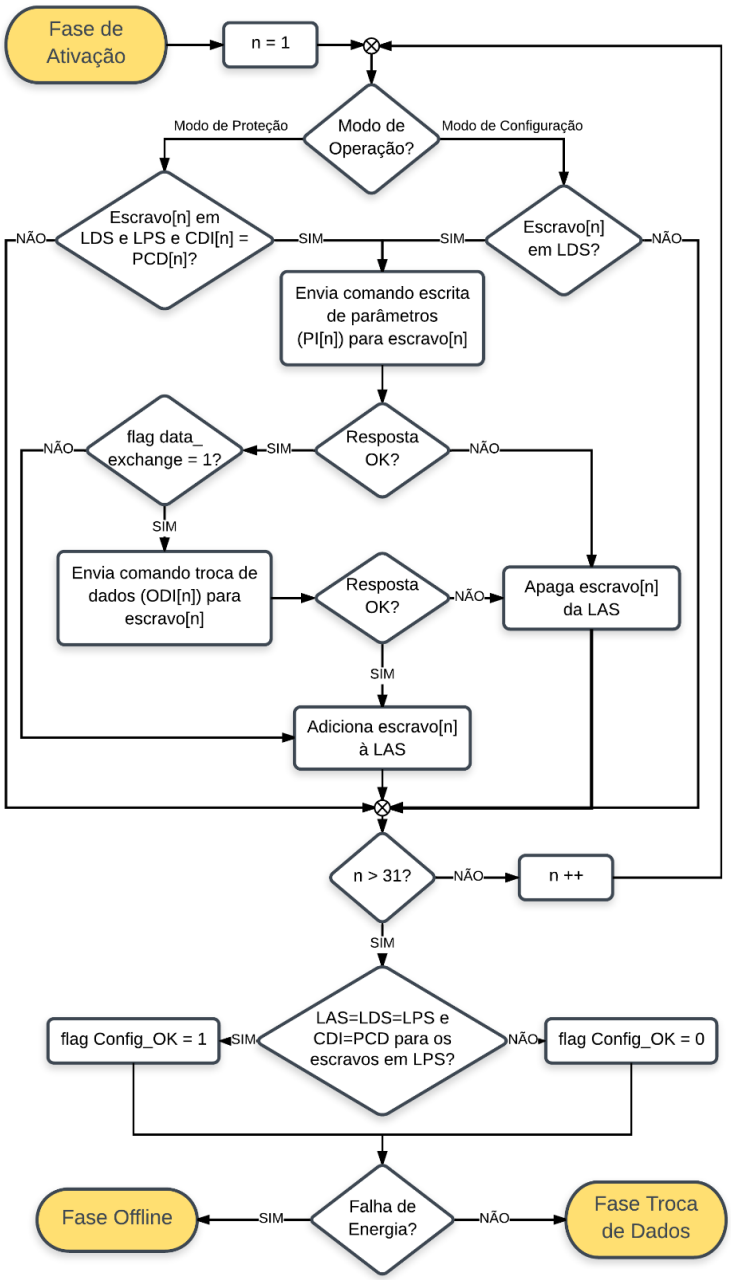
Figura 40: Fluxograma da fase de detecção



Fonte: Elaboração própria, 2017.

Como terceira fase, tem-se a etapa de ativação onde, através dos escravos detectados na última fase e dependendo do modo de operação do mestre há o envio do comando “escrita de parâmetro” e também podendo ser enviado o comando “troca de dados”, posteriormente esse escravo é adicionado na lista LAS. Também é nessa fase que é conferida as listas de escravos e de configuração para atribuição de *config_OK*.

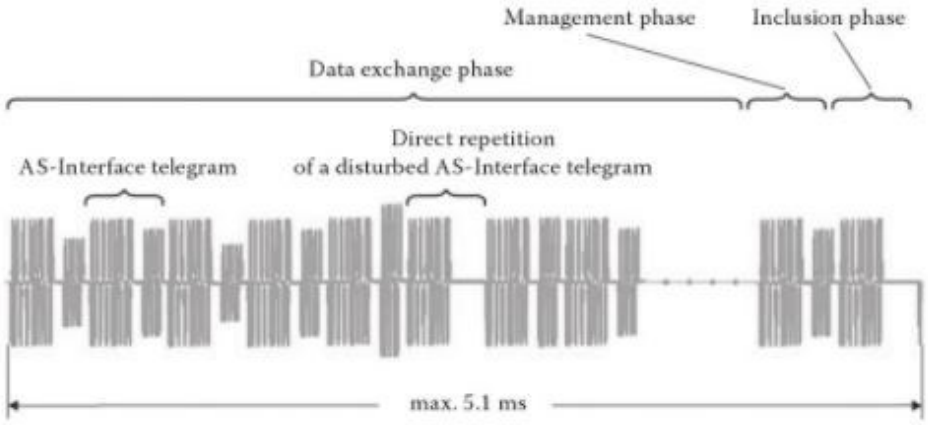
Figura 41: Fluxograma da fase de ativação



Fonte: Elaboração própria, 2017.

O ciclo AS-Interface é representado pelas fases de operação normal, compondo a troca de dados, gerenciamento e inclusão e, deve ter 5 ms de duração. Esse ciclo pode ser visto na figura 42, onde a troca de dados pode conter até 31 escravos.

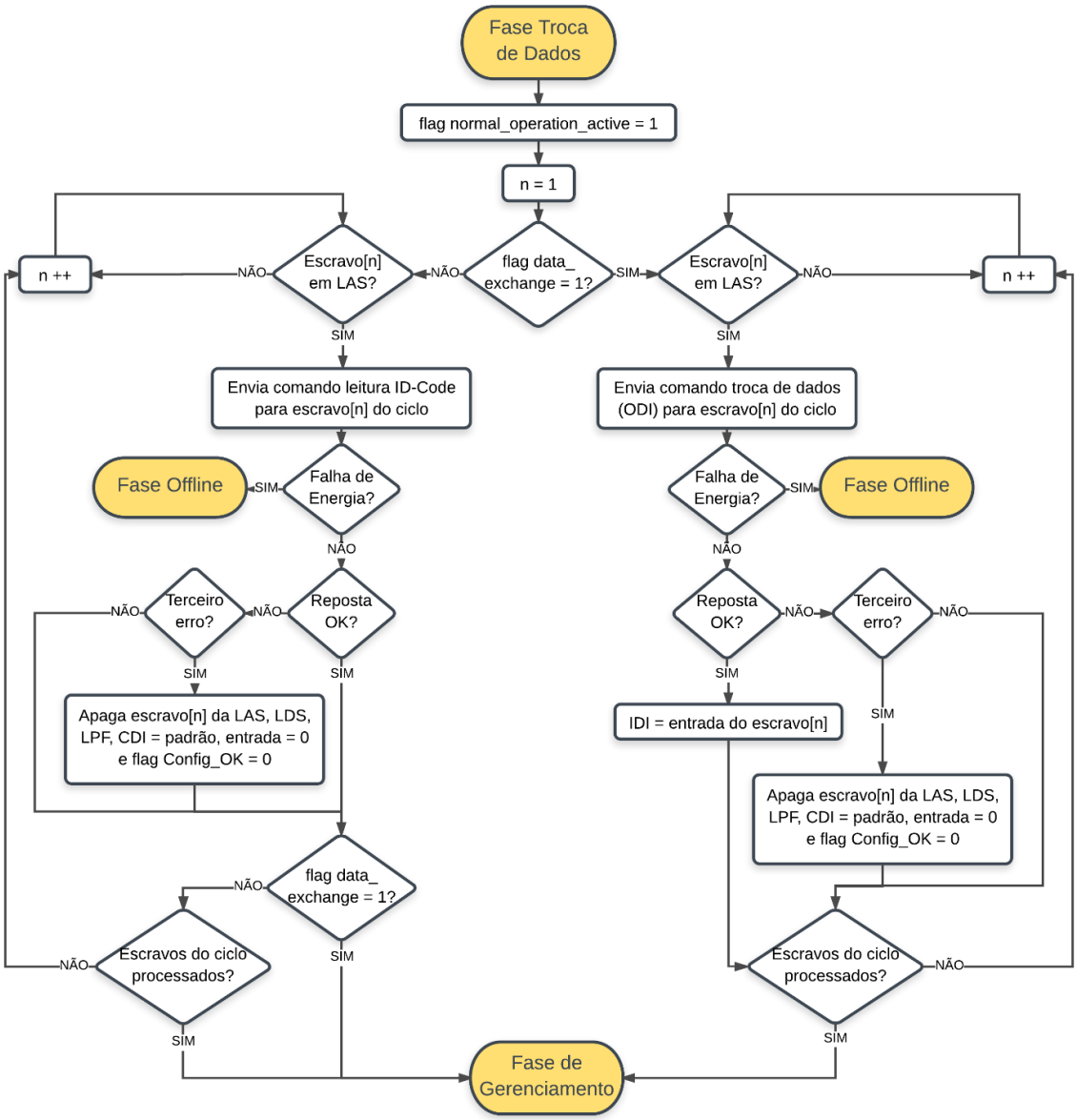
Figura 42: Ciclo da rede AS-Interface



Fonte: ZURAWSKI, 2015.

Na fase de troca de dados basicamente o mestre envia um comando de requisição de troca de dados um a um, para cada escravo, onde é enviado os valores dos 4 bits de dados. Como resposta o escravo retorna ao mestre um frame contendo os 4 bits de valores da entrada do escravo. No final dessa fase toda a lista ODI foi escrita nos escravos e a lista IDI atualizada com as respostas dos escravos. Esse processo pode ser visto em detalhes na figura 43.

Figura 43: Fluxograma da fase de troca de dados



Fonte: Elaboração própria, 2017.

Durante a fase de gerenciamento, todos os comandos são controlados pela função de administração, responsável por fazer a ponte entre a interface com o usuário e o controle de execução. Existem uma série de comandos que o usuário pode requerer à rede, conforme tabela 13. Os seguintes comandos são entre o mestre e o escravo com resposta: escrita de parâmetros, atribuição de endereço, escrita código ID1, apagar endereço, reiniciar escravo, leitura da configuração IO, leitura código ID, leitura código

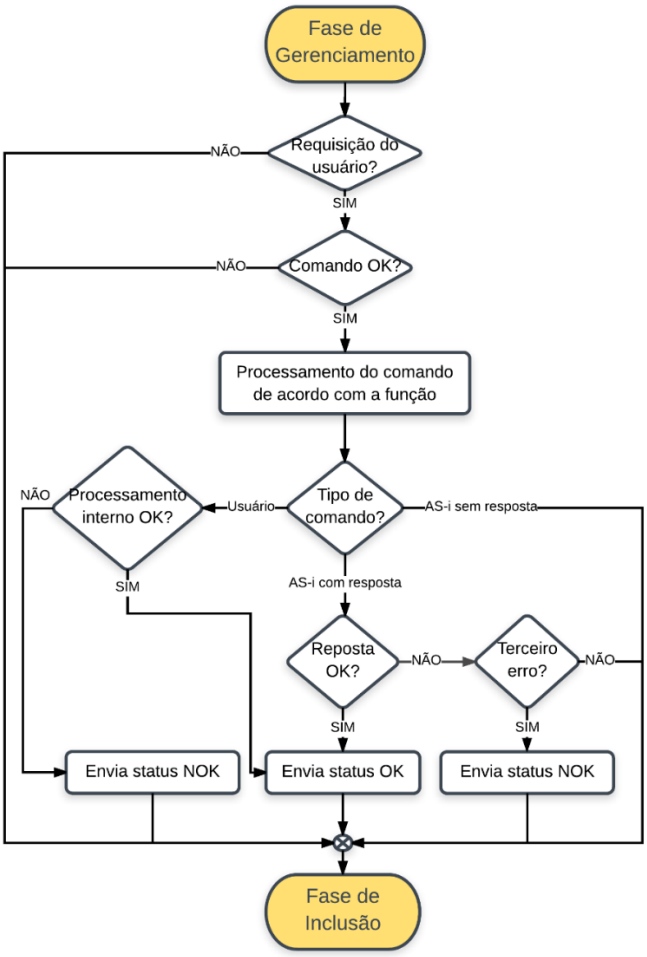
ID1, leitura código ID2 e leitura de status. Tem-se também, comandos também entre o mestre e o escravo, porém sem resposta por parte do escravo, são eles: *broadcast*, modo de programa e ativar canal IRD.

A função de administração também recebe outros comandos além dos citados anteriormente. A diferença é que, estes não trocam informações entre o mestre e os escravos, mas sim entre o usuário e o mestre. São eles; escrita da lista ODI, leitura da lista IDI, armazenar parâmetros atuais, armazenar configuração atual, leitura de *flags,* definição de modo de operação e leitura da lista LDS. É possível perceber que todos esses comandos tem relação direta com as listas e *flags* de dados.

Caso algum comando do mestre receba um frame inválido ou não tenha resposta por parte do escravo, é requerido a retransmissão de até três vezes durante as próximas fases de gerenciamento consecutivas.

Um importante detalhe a ser ressaltado é que, caso algum comando seja enviado pelo usuário ao mestre durante o ciclo AS-Interface atual, esse comando será processado apenas no próximo ciclo, a fim de garantir o ciclo completo em 5 ms, conforme será discorrido em detalhes posteriormente.

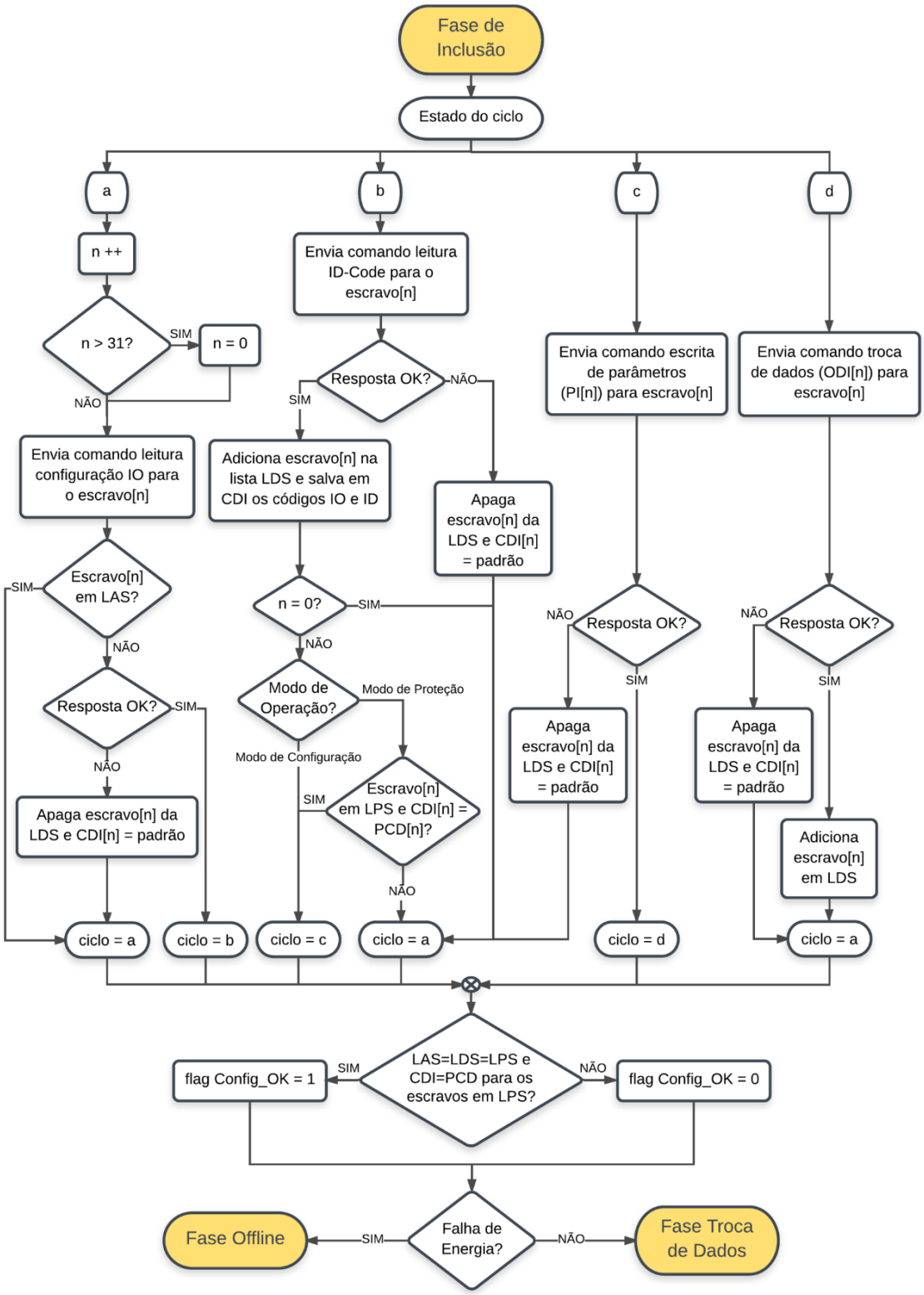
Figura 44: Fluxograma da fase de gerenciamento



Fonte: Elaboração própria, 2017.

Na fase de inclusão, figura 45, o protocolo AS-Interface provê uma maneira fácil e rápida para conexão de novos escravos na rede durante o período de operação. Durante cada fase de inclusão o controle de execução passa para a transmissão um comando de acordo com toda a faixa de escravos e o estado do ciclo atual. Essa fase é dividida em 4 estados, e sempre a cada execução da fase apenas um estado é executado por vez. Começando pelo ciclo “a”, este é responsável por enviar o comando “leitura configuração IO”, caso haja uma resposta positiva é passado para o ciclo “b” que envia o comando “leitura ID-code”, caso haja uma resposta positiva, este escravo é adicionado às listas LDS e CDI. Já os estados “c” e “d” enviam comandos de “escrita de parâmetros” e “troca de dados” a fim de ativar o escravo daquele endereço.

Figura 45: Fluxograma da fase de inclusão



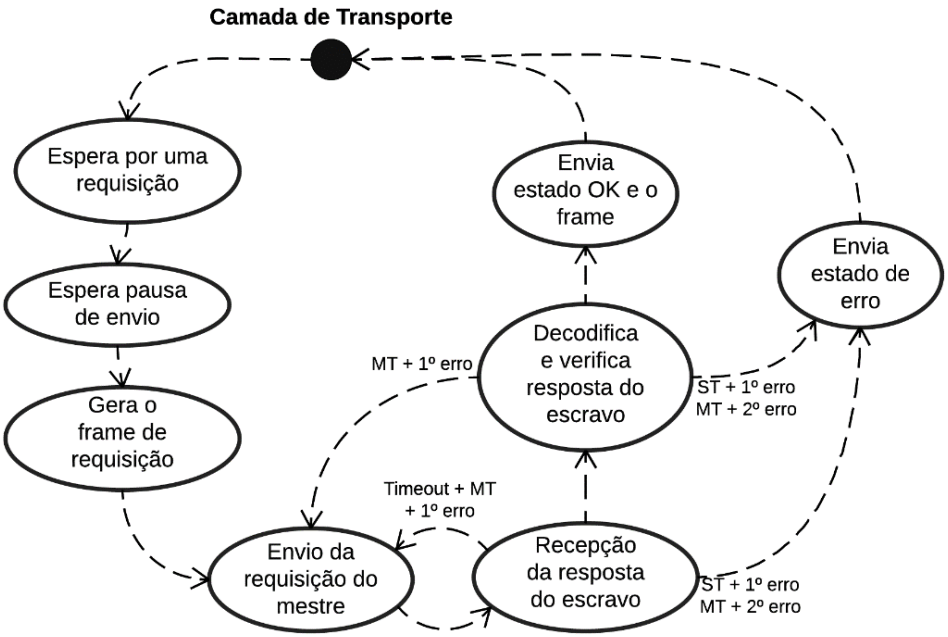
Fonte: Elaboração própria, 2017.

4.3.5 Controle de transmissão

O controle de transmissão é responsável pela troca de frames individuais com os escravos. Definido as características da modulação do sinal, conforme o capítulo 3.4.4, pode-se agora, partir para a implementação via *firmware* da modulação (frame de comando) e a demodulação (frame de resposta).

Antes de mais nada, é necessário explanar acerca dos dois tipos de transmissão, a transmissão simples (ST), onde o mestre envia e espera a resposta apenas uma única vez, modo esse utilizado durante a fase de inclusão e em alguns casos no processamento dos comandos “escrita ID-code_1” e “troca de endereço” pela fase de gerenciamento. Já o segundo método, a transmissão múltipla (MT), o mestre em caso de erro (*timeout* ou resposta inválida) tenta retransmitir novamente o frame apenas mais uma vez, caso essa segunda tentativa seja inválida um estado de erro é repassado para a camada de controle. Na figura 46 é possível visualizar em detalhes cada etapa de processamento da camada de transporte.

Figura 46: Máquina de estados da camada de transporte



Fonte: Elaboração própria, 2017.

Conforme visto na figura 7 todo o controle de transmissão tem regras acerca da comunicação, tempos a serem respeitados entre cada troca de dados. Por exemplo, existem duas pausas, a primeira entre o envio da requisição do mestre e a resposta do escravo, que possui um tempo médio de 18 µs, e caso não haja nenhuma resposta após 60 µs é tratado como *timeout*. A segunda pausa, de envio, é a pausa entre o fim da resposta do escravo e o envio da próxima requisição do mestre, esse valor é dinâmico e depende do número de transações presentes no ciclo da rede AS-Interface.

A seguir, pode-se observar um exemplo de cálculo do tempo do ciclo da rede para 10 escravos ativos na rede e com comando AS-Interface do usuário a ser processado no ciclo, resultando num total de 12 transações por ciclo (10 troca de dados + 1 gerenciamento + 1 inclusão).

$$Requisição\ do\ mestre \rightarrow n * 14bits\ (84\mu s) = 1008\ \mu s$$

$$Pausa\ do\ mestre \rightarrow n * 3bits\ (18\mu s) = 216\ \mu s$$

$$Resposta\ do\ escravo \rightarrow n * 7bits\ (42\mu s) = 504\ \mu s$$

$$Pausa\ de\ envio \rightarrow n * (273\mu s) = 3276\ \mu s$$

Tem-se assim, neste caso, uma duração total do ciclo normal AS-Interface de 5,004 ms. Vale ressaltar que nesse caso, foi assumido o valor da pausa do mestre de 18µs, valor esse que pode variar por inúmeros fatores. Para ser possível calcular qual será o valor da pausa de envio dependendo do número de transações no ciclo atual, basta seguir a seguinte equação:

$$Pausa\ de\ envio = \frac{5000 - \left(144 * n_{transações}\right)}{n_{transações}}\ [\mu s]$$

Na tabela 15 é apresentado todos os tempos já calculados das pausas de envio e do ciclo total da rede dependendo de cada número de escravos instalados na rede, levando em conta também mais uma transação por parte da fase de gerenciamento. Vale lembrar apenas os limites de cada pausa, sendo no mínimo 12 µs e no máximo 500 µs.

Tabela 15: Tempos de pausa de envio da rede

Escravos	Pausa de envio	Ciclo
1	500 µs	1932 µs
2	500 µs	2576 µs
3	500 µs	3220 µs
4	500 µs	3864 µs
5	500 µs	4508 µs
6	481 µs	5000 µs
7	412 µs	5004 µs
8	356 µs	5000 µs
9	311 µs	5005 µs
10	273 µs	5004 µs
11	241 µs	5005 µs
12	214 µs	5012 µs
13	190 µs	5010 µs
14	169 µs	5008 µs
15	151 µs	5015 µs
16	134 µs	5004 µs
17	120 µs	5016 µs
18	106 µs	5000 µs
19	95 µs	5019 µs
20	84 µs	5016 µs
21	74 µs	5014 µs
22	65 µs	5016 µs
23	56 µs	5000 µs
24	49 µs	5018 µs
25	42 µs	5022 µs
26	35 µs	5012 µs
27	29 µs	5017 µs
28	23 µs	5010 µs
29	18 µs	5022 µs
30	13 µs	5024 µs
31	12 µs	5148 µs

Fonte: Elaboração própria, 2017.

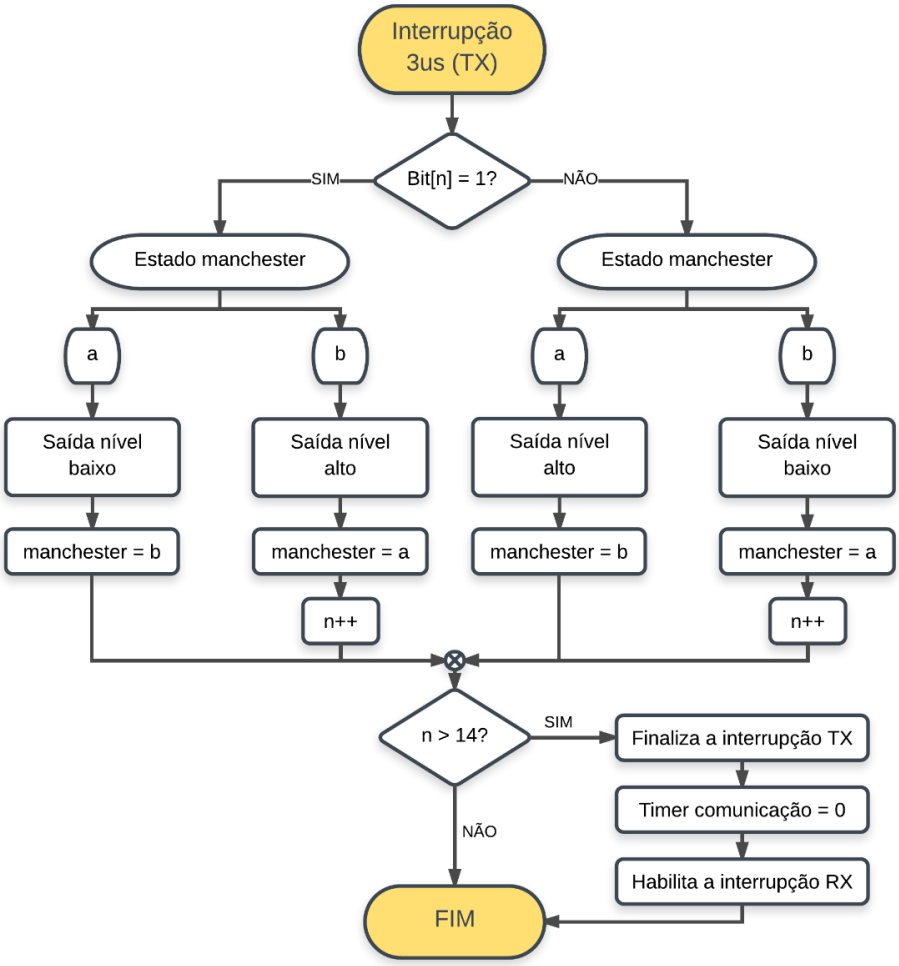
Nesse ponto, para a transmissão e recepção do frame, existem diversas estratégias de implementação, a mais comum é a estratégia de super *loop*, mais simples de ser implementada, porém traz diversas desvantagens como a falta de determinismo e o tempo ocioso de CPU, onde por exemplo na etapa de transmissão do frame a CPU estaria ociosa durante o intervalo de todo o envio entre cada transição de bits, tempo

esse que poderia ser aproveitado para processamento de outras funções, gerando assim um atraso na CPU no que diz respeito a velocidade de processamento do mestre que, dependendo do caso, iria contra os padrões da norma do protocolo, que diz que para 31 escravos digitais, o tempo máximo de um ciclo completo deve ser de 5ms.

Assim, será adotada outra estratégia, a de criar uma máquina de estados alimentada por interrupções (de timer e externa) fazendo assim, com que a CPU otimize ao máximo todo seu processamento. Uma máquina de estado se fundamenta em direcionar o funcionamento de um *firmware* em um número finito e pré-determinado de estados, sendo cada um desses estados uma situação relevante do sistema, onde é possível avançar, recuar ou permanecer no mesmo estado.

Todo e qualquer frame de requisição do mestre possui 14 bits, sendo eles: bit de início e fim, paridade, seleção, endereço e dados. Esse frame é passado para transmissão pelo controle de execução, onde haverá a modulação manchester e envio do sinal elétrico para a rede AS-Interface. Todo esse processo representa o estado de envio da requisição do mestre, visto na figura 46. Para esse fim utiliza-se uma interrupção de 3us (tempo de meio bit do protocolo) onde, a cada interrupção a CPU para o processamento e envia o respectivo bit codificado em manchester, onde apenas um bit manchester é enviado por interrupção até o fim do frame. Um importante ponto a ressaltar é que, uma vez ocupada a camada de transporte para envio de um frame, um novo frame, mesmo que solicitado pela camada de controle, só será processado após o tempo de espera da rede posterior a resposta do escravo, ou seja, não existe a chance de choque entre dados. O fluxograma de processamento da interrupção da transmissão pode ser observado na figura 47.

Figura 47: Fluxograma do envio da requisição

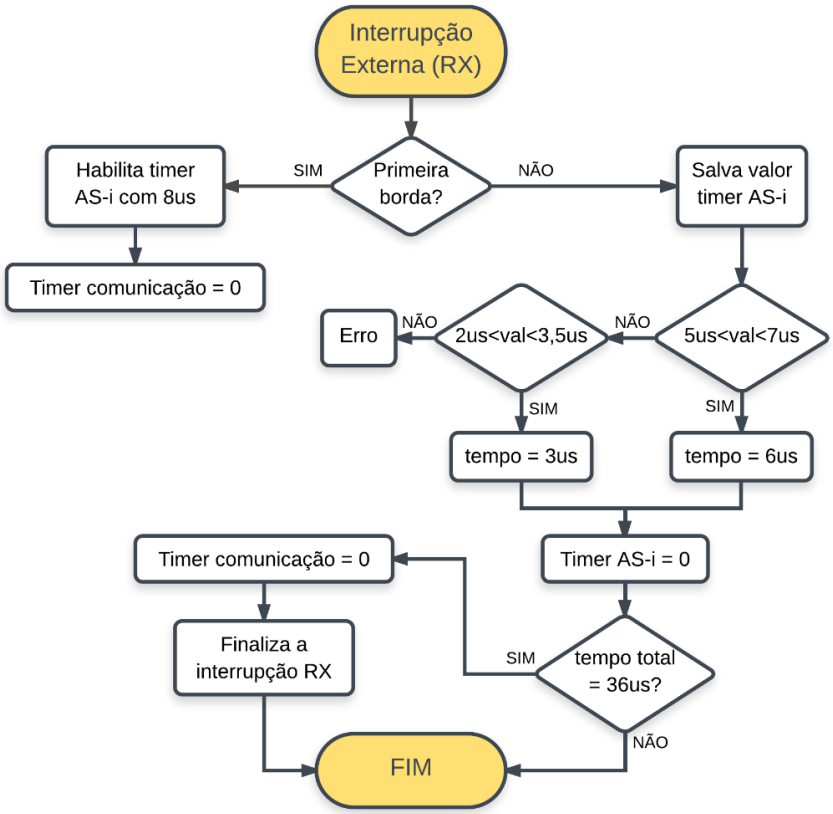


Fonte: Elaboração própria, 2017.

Para o recebimento da resposta do escravo, tem-se agora um frame de apenas 7 bits, incluindo bit de início, fim, paridade e dados. Para a aquisição dos bits, tem-se 2 interrupções, a primeira, consiste na interrupção externa do pino de entrada por borda de subida e descida, para detecção das transições de nível dos bits, já a segunda é uma interrupção de temporizador, utilizado como base de tempo para medir o tempo de duração entre cada borda recebida e, caso ultrapasse o valor de 8us, a camada de transporte interpretará como *timeout* do frame de resposta. O processo para recepção é simples, basicamente mensura-se o tempo de duração de cada transição de bit, que deve ser de 3us ou 6us, e logo após é realizado a decodificação desse sinal para enfim chegar

ao frame de resposta do escravo. Foi escolhida essa estratégia para a aquisição de dados por motivos de performance, pois ela é capaz de realizar o procedimento de uma função de *input capture*, apenas recebendo as bordas e marcando os tempos entre cada transição, permitindo assim com que outros microcontroladores mais simples possam usar essa mesma estratégia. O fluxograma de processamento da interrupção externa da recepção pode ser observado na figura 48.

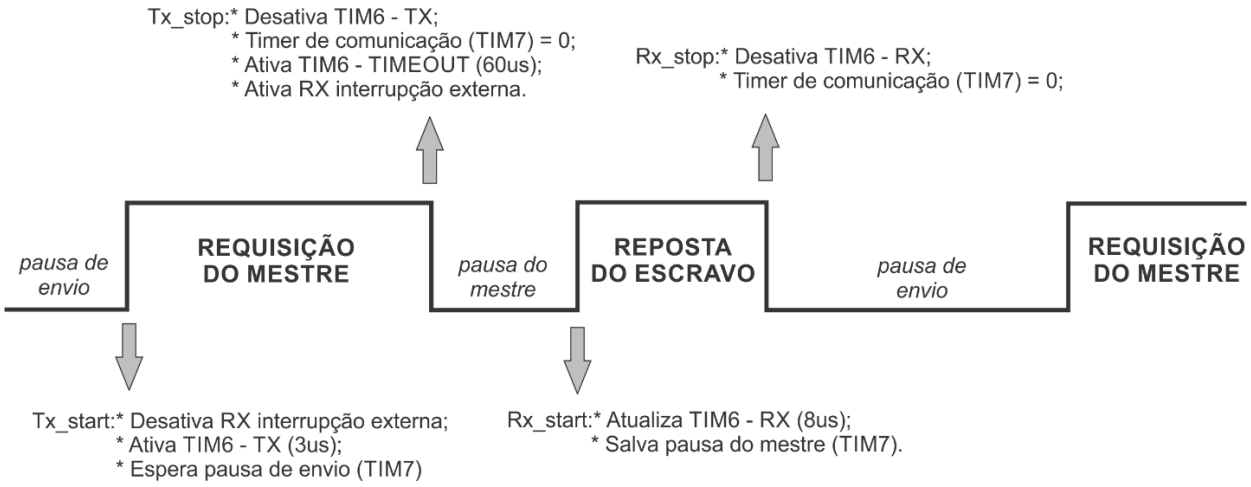
Figura 48: Fluxograma da recepção da resposta



Fonte: Elaboração própria, 2017.

Para uma melhor visualização do funcionamento das interrupções no controle de execução, a figura 49 mostra toda a comunicação, incluindo a requisição do mestre, resposta do escravo e tempos de pausa onde, para cada etapa, traz suas respectivas ações em relação as interrupções do programa.

Figura 49: Interrupções na comunicação da rede

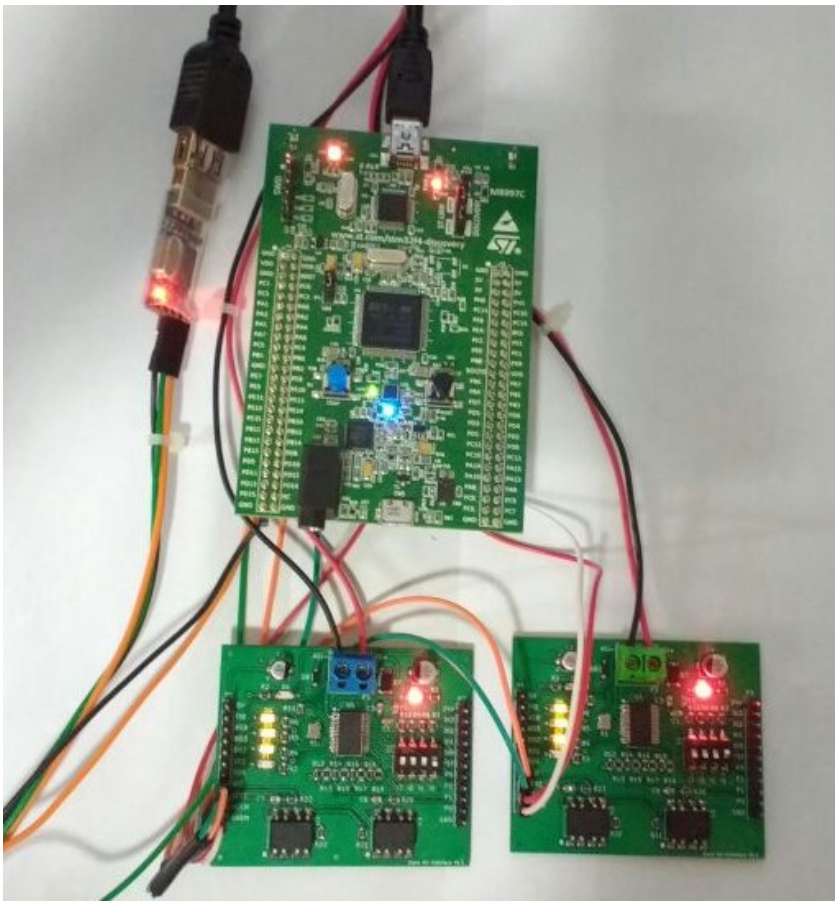


Fonte: Elaboração própria, 2017.

5 TESTES E RESULTADOS DA REDE AS-INTERFACE

Com todas as camadas do mestre implementadas, a interface e o escravo já desenvolvidos, pode-se enfim, montar a rede AS-Interface incluindo o mestre, escravo e a fonte de alimentação. Com a rede montada com apenas dois escravos e utilizando o próprio canal IRD de comunicação com codificação manchester, o usuário pode então prover diversos comandos ao mestre, sejam eles de processamento interno do mestre ou para a rede AS-Interface. Pode-se observar na figura 50 todos os componentes montados e interconectados (escravo, mestre, conversor USB e fonte de alimentação).

Figura 50: Rede AS-Interface montada



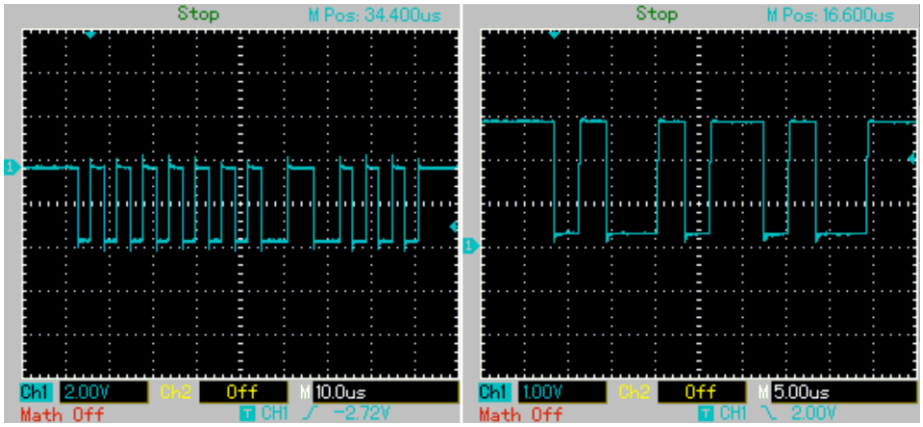
Fonte: Elaboração própria, 2017.

Através do modo de programação do chip ASI4U, explanado no capítulo 2.2.4, pela interface de usuário é possível definir toda a configuração do perfil dos escravos e outros aspectos do chip. O perfil que será utilizado para testes nos escravos é o S7.0.0-0 com 4 entradas e 4 saídas de sinal, para essa configuração, dentro do modo de programa, foi utilizado o comando de escrita de parâmetro para o endereço 0x08 (código ID) com valor zero, endereço 0x09 (código ID2) com zero e endereço 0x0A (código IO) com valor 7. Também foi atribuído o valor 1 ao bit 0 do endereço 0x12 (indicação de status) de forma a alterar a indicação de status dos LEDs vermelho e verde do escravo AS-Interface.

Para demonstrar o envio e a recepção dos frames AS-Interface, na figura 51, tem-se um comando de atribuição de endereço, com valor de endereçamento 11, atribuído a um escravo com endereço zero (padrão), e também o retorno da resposta processada

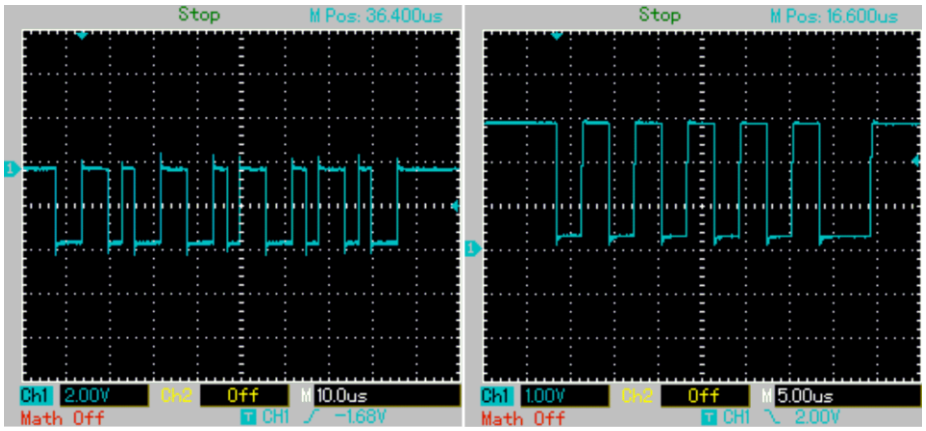
pelo controle de transmissão. Já na figura 52, temos um frame de leitura de status, enviado para o escravo com endereço 5 e a resposta onde, todos os bits de status são zero.

Figura 51: Frame manchester do comando de atribuição de endereço



Fonte: Elaboração própria, 2017.

Figura 52: Frame manchester do comando de leitura de status



Fonte: Elaboração própria, 2017.

A fim de demonstrar todo o potencial da rede foram realizados diversos testes em cenários diferentes, com o intuito de validar o desenvolvimento do projeto e documentar as características de funcionamento da rede AS-Interface. Antes de demonstrar o ciclo normal da rede será apresentado alguns detalhes sobre as operações de inicialização do mestre. Na fase offline, segundo a figura 39, temos o envio do comando broadcast procedido de comandos de reinicialização dos escravos, desde o endereço 0 até o 31, vale ressaltar que, na figura 53 houve resposta apenas nos endereços 12 e 17 onde

haviam escravos na rede, para todos os outros casos foi realizada a tentativa de uma retransmissão do comando.

Figura 53: Teste da fase offline

Fonte: Elaboração própria, 2017.

Na figura 54, pode-se visualizar em detalhes toda a fase de detecção, onde temos o envio do comando de leitura de configuração IO para todos os endereços (0 a 31) e para os que tiverem resposta positiva (endereço 12 e 17) é enviado o comando de leitura do ID_Code, neste caso também existe as retransmissões caso não haja resposta.

Figura 54: Teste da fase de detecção

Fonte: Elaboração própria, 2017.

A fase de ativação, demonstrada na figura 41, mostra o envio, para os escravos detectados na rede, dos comandos de escrita de parâmetro e troca de dados, como existem apenas dois escravos na rede, tem-se um total de quatro comandos neste caso. Os frames podem ser vistos na figura 55.

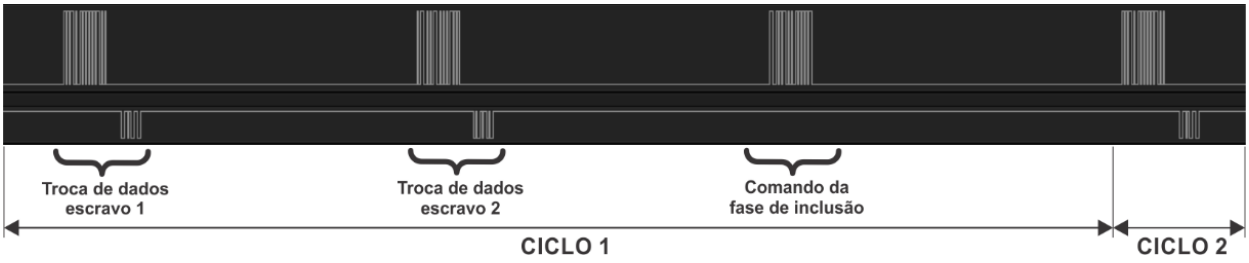
Figura 55: Teste da fase de ativação

Fonte: Elaboração própria, 2017.

O primeiro teste das operações normais do ciclo AS-Interface realizado é a presença de dois escravos na rede junto ao mestre e sem comandos por parte do usuário.

Na Figura 56: Teste do ciclo da rede com 2 escravos, podemos ver na parte superior o sinal provindo do mestre e na parte inferior o sinal de resposta dos escravos. Pode-se perceber que o ciclo da rede está correto e de acordo com a figura 42.

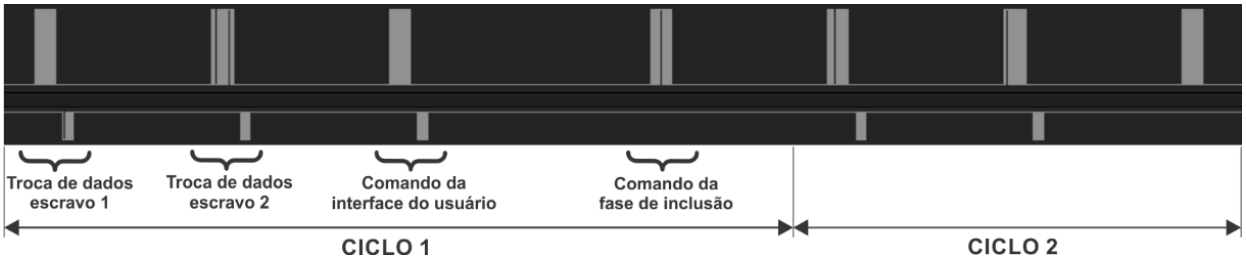
Figura 56: Teste do ciclo da rede com 2 escravos



Fonte: Elaboração própria, 2017.

O segundo teste consiste no cenário com dois escravo na rede e, neste caso, teremos um comando provindo da interface de usuário que deve ser processado juntamente no ciclo. Vale ressaltar que, o comando enviado a partir da interface era o comando de leitura de status para um dos escravos presente na rede.

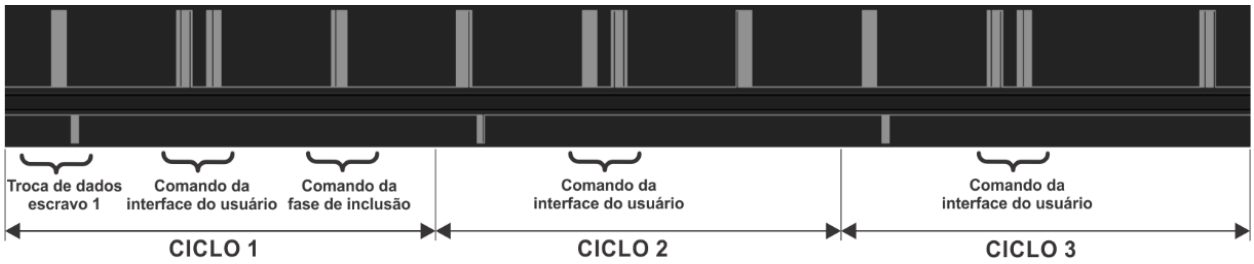
Figura 57: Teste do ciclo da rede com 2 escravos e comando do usuário



Fonte: Elaboração própria, 2017.

O terceiro teste consiste no mesmo cenário do teste anterior porém, agora com um escravo apenas e o comando enviado a partir da interface era o comando de leitura de status para um endereço diferente do escravo presente na rede.

Figura 58: Teste do ciclo da rede com 1 escravo e comando do usuário

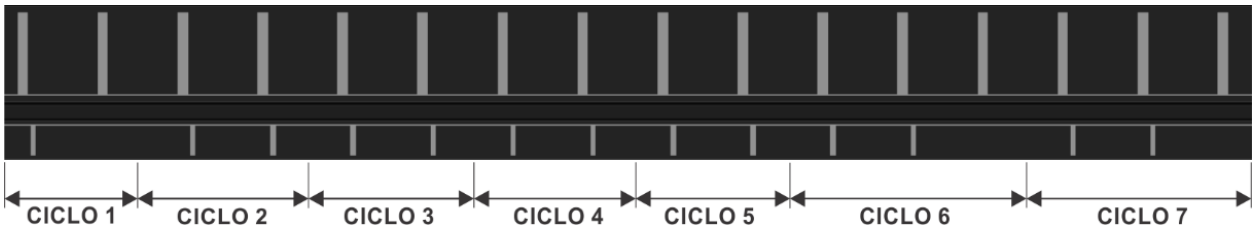


Fonte: Elaboração própria, 2017.

Na figura 58, podemos observar todos os casos de retransmissão em caso de *timeout* da rede. Na situação presente, foi enviado um comando para um endereço que não existia na rede, não havendo assim resposta. A primeira retransmissão, consiste na do controle de transmissão onde, no caso como os comandos advindos da interface de usuário são do tipo transmissão múltipla (MT), caso não haja resposta do escravo o mestre retransmite o comando imediatamente após a pausa. Caso ainda sim, não haja resposta para o comando, a camada de controle de execução tenta retransmitir mais duas vezes nos dois próximos ciclos consecutivos e, caso novamente não haja resposta positiva retorna ao usuário o status NOK, finalizando a tentativa de envio do comando em questão.

No último cenário retratado, tem-se apenas um escravo na rede e o segundo escravo é adicionado ao enlace durante o processamento do ciclo normal do mestre. A fase de inclusão, retratada na figura 45, é responsável pela troca de comandos com o novo escravo e se tudo ocorrer bem esse escravo entrará na lista de escravos ativos, passando assim a fazer parte na próxima fase de troca de dados. Na figura 59 temos essa troca de dados onde, no ciclo 1 tem-se o ciclo com apenas 1 escravo e a partir do ciclo 2 a fase de inclusão entra em ação, enviando o primeiro comando ao novo escravo, após os ciclos 3, 4 e 5 finaliza-se o envio das quatro requisições para adição do escravo na LAS. A partir do ciclo 6, a fase de troca de dados já possui dois escravos ativos na rede.

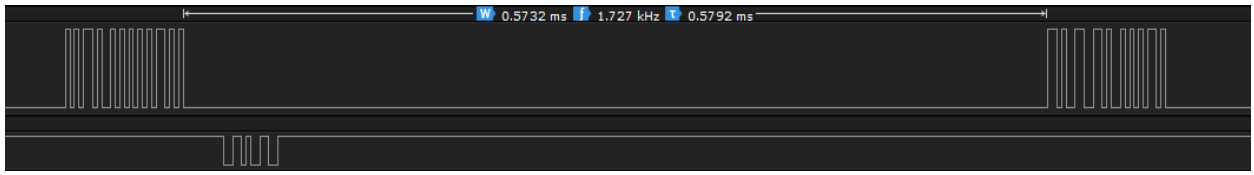
Figura 59: Teste do ciclo da rede com adição de um escravo



Fonte: Elaboração própria, 2017.

Como visto no capítulo 4.3.5, temos logo após a transmissão a pausa de envio até a próxima transmissão. No caso de 1 escravo na rede essa pausa deve ser de 500 µs. Todavia, como se pode observar na figura 60, essa pausa está com aproximadamente 573 µs, mais de 70 µs a mais que o esperado, em outros casos para o tempo de pausa menor, a pausa real é sempre 70 µs a mais aproximadamente. Isso se deve ao fato de que na máquina de estados do controle de execução, existem vários lugares que logo após o envio do frame é necessário decodificar e verificar a resposta do usuário, isso foi feito através de um *loop* de espera até o final da recepção, ou seja, o processamento fica congelado até o final da recepção da resposta, só ai inicia-se a contagem do tempo de pausa de envio. Esse atraso de aproximadamente 70 µs é composto pela pausa do mestre (~28us) mais o tempo do frame de resposta (42 µs).

Figura 60: Teste da pausa de envio



Fonte: Elaboração própria, 2017.

6 CONSIDERAÇÕES FINAIS

6.1 CONCLUSÕES

O protocolo AS-Interface tem suas inúmeras vantagens no campo de aplicação e o desenvolvimento de qualidade de um bom *firmware* para o mestre é de suma importância para a rede, principalmente no que diz respeito a seguir toda a norma do protocolo, garantindo a qualidade do sinal e funcionamento correto da rede e respeitando todos os tempos de transmissão, de ciclo, retransmissão e tratamento de erros.

Neste projeto, após realizar um revisão da literatura acerca de protocolos industriais foi possível o entendimento dos conceitos necessários para os processos de comunicação e controle da rede AS-Interface. Em seguida com todos os conceitos perfeitamente esclarecidos foram definidos os requisitos do projeto, incluindo a escolha dos componentes eletrônicos, definições do mestre, do escravo e da interface.

Já foi visto que, o mestre é composto de 3 camadas principais, a interface de usuário, que liga o usuário diretamente ao mestre através de comunicação UART, o controle de execução, que controla através de máquinas de estado, todas as fases e todo o ciclo da rede, recebe também os comandos vindos da interface do usuário para processamento interno. Já o controle de transmissão é responsável pela codificação do sinal em formato manchester-II a partir do frame, provindo do controle de execução e enviado ao enlace da rede AS-Interface onde, também é processado o recebimento do sinal de resposta do escravo, permitindo assim a respectiva comunicação entre os dispositivos da rede.

Com todo o projeto bem definido e todos os requisitos listados, foi realizado o desenvolvimento dos 3 elementos que compõem a rede: a interface, o escravo e o mestre. A interface consiste na aplicação em software com a linguagem python, o escravo consiste no desenvolvimento do hardware e para o mestre foi desenvolvido todo o código em linguagem C do sistema embarcado na placa da ST.

Ao final desse projeto pode-se concluir através dos testes validados e resultados obtidos que, o objetivo geral de desenvolver todos os dispositivos do sistema para que possam comunicar através do protocolo AS-Interface foi alcançado com sucesso.

6.2 SUJESTÕES PARA TRABALHOS FUTUROS

Ainda há muito o que ser feito em relação ao sistema como um todo. Quanto ao mestre, como visto no projeto possui perfil M2, pode-se implementar um mestre de perfil M4 de acordo com a especificação 3.0 da norma, suportando até 62 escravos na rede com endereçamento estendido, perfis de escravos analógicos, entre outros. Para os tempos da pausa de envio deve-se reestruturar o programa trocando os *loops* de espera por interrupções em tempo real. Também é aconselhado o cálculo do tempo de envio a cada ciclo da rede onde, dependerá dos valores da média de pausas do mestre, que variam de acordo com cada escravo, distância do mestre, etc.

Quanto ao desenvolvimento dos escravos, pode-se desenvolver escravos que utilizam indutores externos ao chip ASI4U, no caso para escravos que consomem mais de 55 mA de corrente. Também é possível construir mais escravos, a fim de ter uma rede maior para diferentes testes e situações adversas do sistema de comunicação.

No que diz respeito a interface do usuário pode ser desenvolvido uma interface gráfica para facilitar a operação e diagnóstico da rede pelo usuário e incluir mais comandos de acordo com o perfil do mestre. Também pode-se ter mais recursos, como dados estatísticos de erros da rede, por exemplo, para cada escravo é possível ter uma lista de erros da camada de transporte junto a contadores, tudo isso com o objetivo de otimizar o diagnóstico da rede.

REFERÊNCIAS

ALMEIDA M. B.; ATAÍDE F. H.; SANTOS M. M. D. **Protocolo AS-i: agregando inteligência a sensores e atuadores**. In: Profinews Edição 15, Abril, 2007. Disponível em: <http://www.profibus.org.br/news/abril2007/news.php?dentro=2>. Acesso em: 03/09/16.

AS-INTERNATIONAL ASSOCIATION. **AS-Interface: The Automation Solution**. Germany, 2002.

AS-INTERNATIONAL ASSOCIATION. **Complete Specification: Version 2.11**. Germany, 2000.

AS-INTERNATIONAL ASSOCIATION. **AS-Interface Academy**. Disponível em: <http://www.as-interface.net/media/academy/content/sys/start/start_asi.en.html>.Acesso em: 10/10/16.

ATAIDE, F.H. **Estudo Técnico EST-DE-0007-04 - AS-Interface**. SMAR Equipamentos Industriais Ltda, 2004.

ATMEL. **Aplication Note: Manchester Coding Basics**. San Jose, CA: [s.n], 2015.

BIHL+WIEDEMANN. **AS-I 3.0 Master PCI Board / AS-I 3.0 Master Compact PCI Board**. Mannheim: [s.n], 2016.

CHEN, W. K. **The Electrical Engineering Handbook**. London: Elsevier. p. 326-331. 2004.

FESTO. **AS-interface® components**. [S.l.: s.n], 2016.

GANSSLE J. G. **The Art of Designing Embedded Systems**. 2 ed. [S.l.]: Newnes, 2008.

HITEX. **The Insider's Guide to the STM32 ARM® Based Microcontroller: An Engineer's Introduction to the STM32 Series**. 2 ed. Coventry: [s.n], 2009.

IDT. **ASI4U Application Note: EEPROM Programming**. San Jose, CA: [s.n], 2016a.

IDT. **ASI4U / ASI4U-E / ASI4U-F Datasheet**. San Jose, CA: [s.n], 2016b.

IFM. **AS-Interface Manual: Tips and Tricks for users**. 2.2 ed. [S.l.: s.n], 2012a.

IFM. **Basic device manual AS-I controller**. [S.l.: s.n], 2012b.

INTERLINKBT. **AS-interface® Tutorial**. Plymouth, MN: [s.n], 2002.

KHORWAT, I.A.; NAAS, N. **A New Hardware Implementation of Manchester Line Decoder**. In: International Journal of Electrical Computer, Energetic, Electronic and Communication Engineering, Vol. 4, No.9, 2010.

LOUIS, E. F. **Handbook of Serial Communications Interfaces: A Comprehensive Compendium of Serial Digital Input/Output (I/O) Standards**. Waltham, MA: Elsevier. 2016.

MIRA. **MISRA-C-2004: Guidelines for the use of the C language in critical systems.** Misra: Nuneaton**,** 2004.

PEPPERL+FUCHS. **Reference & Buyer's Guide: AS-Interface Products**. 6 ed. USA : [s.n], 2012.

REYNDERS, D.; MACKAY, S.;WRIGHT, E. **Practical Industrial Data Communications: Best Practice Techniques**. Burlington, MA: Elsevier, p.271-278, 2005.

RTA. **AS-Interface: I/O Solution Protocol**. 2014. Disponível em: <http://www.rtaautomation.com/technologies/as-interface>. Acesso em 09/08/17.

SCHNEIDER ELECTRIC. **Redes de Comunicação Industrial: Documento Técnico nº 2**. [S.l.: s.n], 2007.

SCHNEIDER ELECTRIC. **Product Environmental Profile: AS-Interface Cable**. [S.l.]: Pep-Ecopassport, 2015.

SENSE. **Manual de Instruções Rede AS-Interface: Recomendações de Instalação**. São Paulo: [s.n], 2004.

SIEMENS. **Catálogo Técnico: AS-Interface**. São Paulo: FSC, p. 8-11, 2016.

SIEMENS. **Manual AS-Interface – Introduction and Basics**. Nürnberg: [s.n], 2006.

SIEMENS. **Manual AS-Interface Master CM AS-i Master ST**. Nürnberg: [s.n], 2015.

SMAR. **Industrial Networks: Part 1**. Março, 2012. Disponível em: <http://www.smar.com/en/technical-article/industrial-networks-part-1>. Acesso em 09/08/16.

SMAR. **Tutorial sobre a Tecnologia AS-i.** Disponível em: <http://www.smar.com/brasil/asi>. Acesso em: 11/08/16.

ST. **Application note: EEPROM emulation in STM32F40x / STM32F41x microcontrollers**. DocID022108, Rev. 1, 2011.

ST. **Datasheet: STM32F407xx**. DocID022152, Rev. 6, 2015.

ST. **User Manual: Discovery kit with STM32F407VG MCU**. DocID022256, Rev. 5, 2016a.

ST. **Programming Manual: STM32F3, STM32F4 and STM32L4 Series Cortex® - M4 programming manual**. DocID022708, Rev. 5, 2016b.

ST. **Reference Manual: STM32F407xx advanced ARM® - based 32-bit MCUs**. DocID018909, Rev 15, 2017.

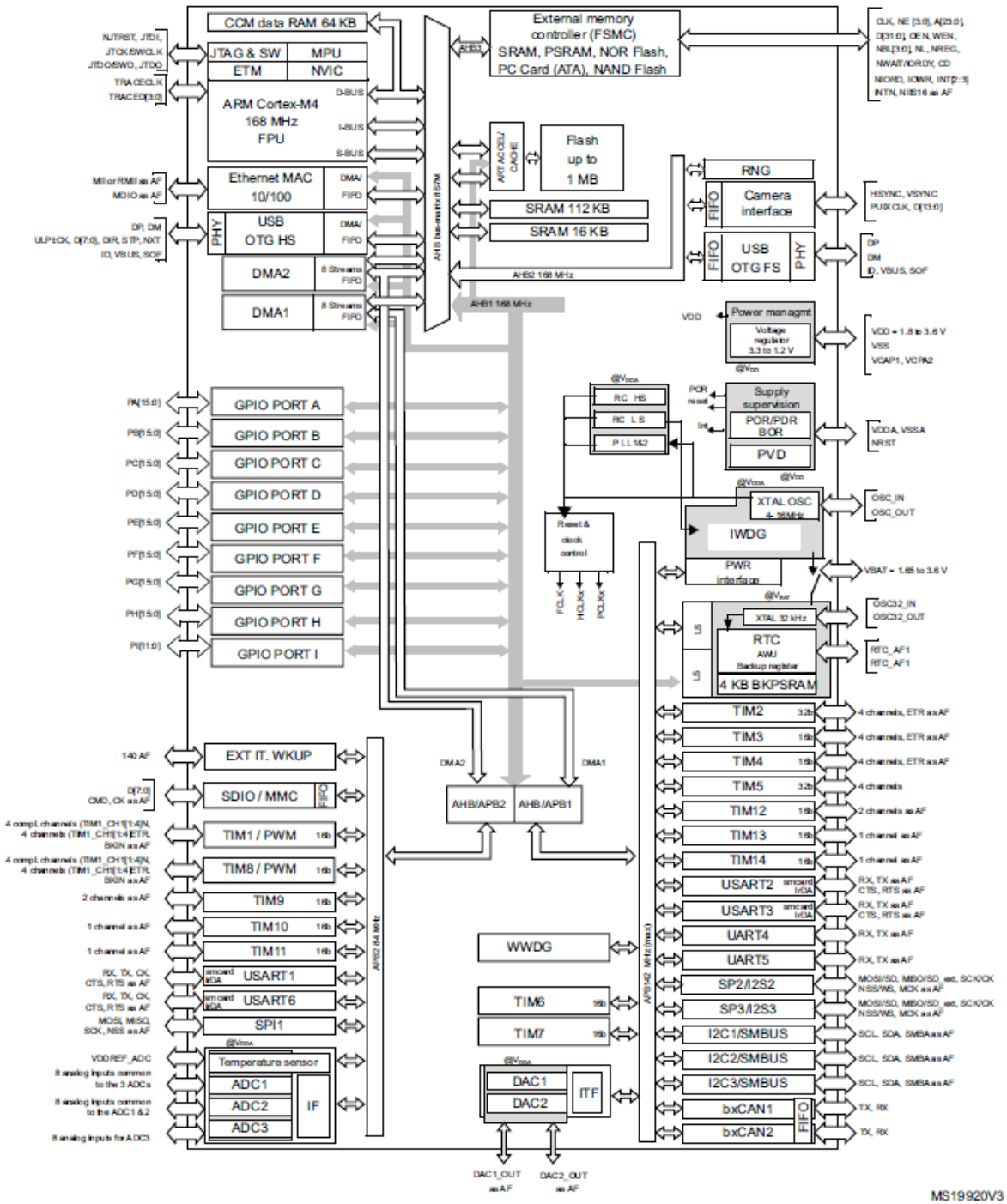
SVEDA M. ; ZEZULKA F. **Interconnecting Low-Level Fieldbusses**. In: EUROMICRO 97. New Frontiers of Information Technology, 1997. IEEE, 2002.

SVEDA M. ; VRBA M. ; ZEZULKA F. **Coupling architectures for low-level fieldbusses**. In: Engineering of Computer Based Systems, 2000. (ECBS 2000), 2000. IEEE, 2002.

ZMD. **ASI4U Evaluation Board**. Rev 1. 2006.

ZURAWSKI R. **Industrial Communication Technology Handbook**. 2 ed. San Francisco: CRC Press, p. 6-11 à 6-55. 2015.

ANEXO A – Diagrama de blocos STM32F40xxx



APÊNDICE A – Estrutura dos comandos da interface do usuário

Escrita de Parâmetros

Valor	Tamanho	Descrição
DST	B	Endereço do escravo
0x02	B	Comando ID
TIPO	B	Tipo de endereço
DADO	B	Valor do parâmetro
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x02	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X01	B	Tamanho do payload
PARAM	B	Valor de Parametro ecoado
CRC	2B	CRC

Atribuição de endereço

Valor	Tamanho	Descrição
DST	B	Endereço do escravo
0x03	B	Comando ID
TIPO	B	Tipo de endereço
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x03	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X00	B	Tamanho do payload
CRC	2B	CRC

Escrita ID_Code_1

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x04	B	Comando ID
X	B	NÃO USADO
DADO	B	Valor ID_Code_1
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x04	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X00	B	Tamanho do payload
CRC	2B	CRC

Apagar Endereço

Valor	Tamanho	Descrição
DST	B	Endereço do escravo
0x05	B	Comando ID
TIPO	B	Tipo de endereço
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x05	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X00	B	Tamanho do payload
CRC	2B	CRC

Reiniciar Escravo

Valor	Tamanho	Descrição
DST	B	Endereço do escravo
0x06	B	Comando ID
TIPO	B	Tipo de endereço
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x06	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X00	B	Tamanho do payload
CRC	2B	CRC

Leitura Configuração IO

Valor	Tamanho	Descrição
DST	B	Endereço do escravo
0x07	B	Comando ID
TIPO	B	Tipo de endereço
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x07	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X01	B	Tamanho do payload
IO CODE	B	Código IO do escravo
CRC	2B	CRC

Leitura ID_Code

Valor	Tamanho	Descrição
DST	B	Endereço do escravo
0x08	B	Comando ID
TIPO	B	Tipo de endereço
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x08	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X01	B	Tamanho do payload
ID CODE	B	Código ID do escravo
CRC	2B	CRC

Leitura ID_Code_1

Valor	Tamanho	Descrição
DST	B	Endereço do escravo
0x09	B	Comando ID
TIPO	B	Tipo de endereço
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x09	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X01	B	Tamanho do payload
ID CODE 1	B	Código ID 1 do escravo
CRC	2B	CRC

Leitura ID_Code_2

Valor	Tamanho	Descrição
DST	B	Endereço do escravo
0x0A	B	Comando ID
TIPO	B	Tipo de endereço
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x0A	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X01	B	Tamanho do payload
ID CODE 2	B	Código ID 2 do escravo
CRC	2B	CRC

Leitura de Status

Valor	Tamanho	Descrição
DST	B	Endereço do escravo
0x0B	B	Comando ID
TIPO	B	Tipo de endereço
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x0B	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X0B	B	Tamanho do payload
STATUS 0	B	Acessibilidade da EEPROM
STATUS 1	B	Falha de periféricos detectada
STATUS 2	B	Zero
STATUS 3	B	Consistência da EEPROM
CRC	2B	CRC

Broadcast (Reiniciar)

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x0C	B	Comando ID
X	B	NÃO USADO
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

SEM RESPOSTA

Modo de programa

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x0D	B	Comando ID
X	B	NÃO USADO
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

SEM RESPOSTA

Ativar canal IRD

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x0E	B	Comando ID
X	B	NÃO USADO
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

SEM RESPOSTA

Leitura lista de entradas IDI

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x0F	B	Comando ID
X	B	NÃO USADO
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x0F	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X1F	B	Tamanho do payload
INPUT 1	B	input data 1
...	...	...
INPUT 31	B	input data 31
CRC	2B	CRC

Escrita lista de saídas ODI

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x10	B	Comando ID
X	B	NÃO USADO
X	B	NÃO USADO
0X1F	B	Tamanho do payload
OUTPUT	B	output data 1
...	...	...
OUTPUT	B	output data 31
CRC	2B	CRC

Valor	Tamanho	Descrição
0x10	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X00	B	Tamanho do payload
CRC	2B	CRC

Armazenar Parametros Atuais

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x11	B	Comando ID
X	B	NÃO USADO
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x11	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X00	B	Tamanho do payload
CRC	2B	CRC

Armazenar Configurações Atuais

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x12	B	Comando ID
X	B	NÃO USADO
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x12	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X00	B	Tamanho do payload
CRC	2B	CRC

Leitura de *flags*

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x13	B	Comando ID
X	B	NÃO USADO
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x13	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X09	B	Tamanho do payload
FLAG 1	B	Flag Config OK
...	...	...
FLAG 9	B	Flag Periphery_OK
CRC	2B	CRC

Definição do Modo de Operação

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x14	B	Comando ID
X	B	NÃO USADO
DADO	B	Modo de operação (0 ou 1)
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x14	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X00	B	Tamanho do payload
CRC	2B	CRC

Leitura da lista de escravos detectados LDS

Valor	Tamanho	Descrição
X	B	NÃO USADO
0x15	B	Comando ID
X	B	NÃO USADO
X	B	NÃO USADO
0X00	B	Tamanho do payload
CRC	2B	CRC

Valor	Tamanho	Descrição
0x15	B	Comando ID
STATUS	B	Status da requisição (OK, NOK)
0X1F	B	Tamanho do payload
SLAVE 1	B	Escravo detectado 1 (1, 0)
...	...	...
SLAVE 31	B	Escravo detectado 31 (1, 0)
CRC	2B	CRC