UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇAO

MARCOS ALVES LEITÃO

Implementação de um Servidor OPC UA em linguagem C# para comunicação com dispositivos através do protocolo Modbus/Ethernet em tempo real

Prof. Dr. Fernanda Gusmão Kastensmidt
Orientador

Porto Alegre, Dezembro de 2010

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
BMS	Sistemas de Automação Industrial
CLP	Controlador Lógico Programável
COM	Component Object Model
DCOM	Distributed Component Object Model
ERP	Enterprise Resource Planning
HDLC	High Level Data Link Control
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
IAS	Sistemas de Automação Industrial
IP	InternetProtocol
MES	Manufacturing Execution system
OLE	Object Linking and Embedding
OPC UA	OLE for Process Control Unified Architecture
PLC	Programador Lógico Programável
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SDK	Software Development Kit
TCP	Transmission Control Protocol
XML	Extensible Mark Up Language
FDT	Field Device Tool

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O setor de automação industrial possui níveis de aquisição e integração de dados que vão desde os equipamentos de chão de fábrica até os sistemas de gestão empresarial conhecidos como ERP (Entreprise Resource Planning). Para que os diversos sistemas existentes nesses níveis do setor industrial pudessem se comunicar, foram desenvolvidos diversos protocolos, os quais eram inerentemente dependentes dos fabricantes. A dificuldade de integração de dados originada por essa heterogeneidade de protocolos resultou em uma necessidade pelo desenvolvimento de um padrão de comunicação, conhecido como OPC UA (Ole for Process Control - Unified Architecture).

O protocolo OPC UA trata de um mecanismo de comunicação baseado em uma arquitetura orientada a serviços, onde clientes enviam requisições a servidores, e estes disponibilizam os dados através de um espaço de endereçamento bem definido. Além disso, esse padrão especifica modelos de informação e segurança e perfis aos quais clientes e servidores podem ter conformidade, trazendo flexibilidade de implementação para os diversos sistemas presentes no setor industrial.

O projeto deste Trabalho de Conclusão dedica-se ao desenvolvimento de um Servidor OPC UA em linguagem C# para comunicação com CLPs através do protocolo Modbus/Ethernet em tempo real e disponibilização dos dados através do protocolo OPC UA na Internet. Nesse contexto, aplicações cliente podem comunicar-se com o Servidor OPC UA desenvolvido e visualizar as variações de valores dos dados dos CLP's. Dessa forma, é possível utilizar o padrão OPC UA em sistemas legados - já em funcionamento na indústria - tornando muito mais acessível a aceitação desse novo padrão.

Embora seja aplicado apenas a dispositivos com comunicação Modbus/Ethernet, esse projeto pode ser expandido para interfacear com os outros protocolos de chão de fábrica, como o Profibus e o FDT.

Palavras-chave: OPC UA, Modbus, OPC Server, Driver Modbus, Automaçao Industrial.

ABSTRACT

The industrial automation sector has levels of acquisition and data integration ranging from plant floor devices to the business management systems known as ERP (Enterprise Resource Planning). In order to allow this systems to communicate with each other, several protocols were developed, which were inherently dependent on manufacturers. The communication issues from these heterogeneous protocols resulted in the developing of a communication standard, known as OPC UA (Ole for Process Control - Unified Architecture).

The OPC UA protocol is a communication mechanism based on a service-oriented architecture, in which clients send requests and servers provide the data through an well-defined address space. Additionally, this standard specifies information and security models and profiles to clients and servers, adding flexibility to implement the many systems present in the industrial sector.

This project aimed to develop an OPC UA Server with the C# language in order to communicate PLCs through Modbus/Ethernet in real time and to provide data through OPC UA protocol on Internet. In this context, client applications may communicate with this OPC UA Server and visualize data values changes. Additionally, it is possible to use the OPC UA standard in legacy systems - already working in the industry - making the transition to this new standard much easier.

Although the server developed in this project is applied only to devices with Modbus/Ethernet communication, it may be expanded to interface with other plant floor protocols, such as Profibus and FDT.

1 INTRODUÇÃO

Na área de automação industrial existem diversos sistemas componentes como SCADA (Supervisory Control and Data Acquisition), MES (Manufacturing Execution system) e HMI (Human Machine Interface). Cada um desses sistemas se comunica com um protocolo diferente e, muitas vezes, proprietário. Assim, no processo produtivo, há muitas partes desconexas pelo uso de diferentes protocolos, tornando o processo não tão eficiente quanto poderia ser. Além disso, esses protocolos de chão de fábrica não disponibilizam os dados de maneira adequada para os níveis acima, como gerenciamento e geração de relatórios.

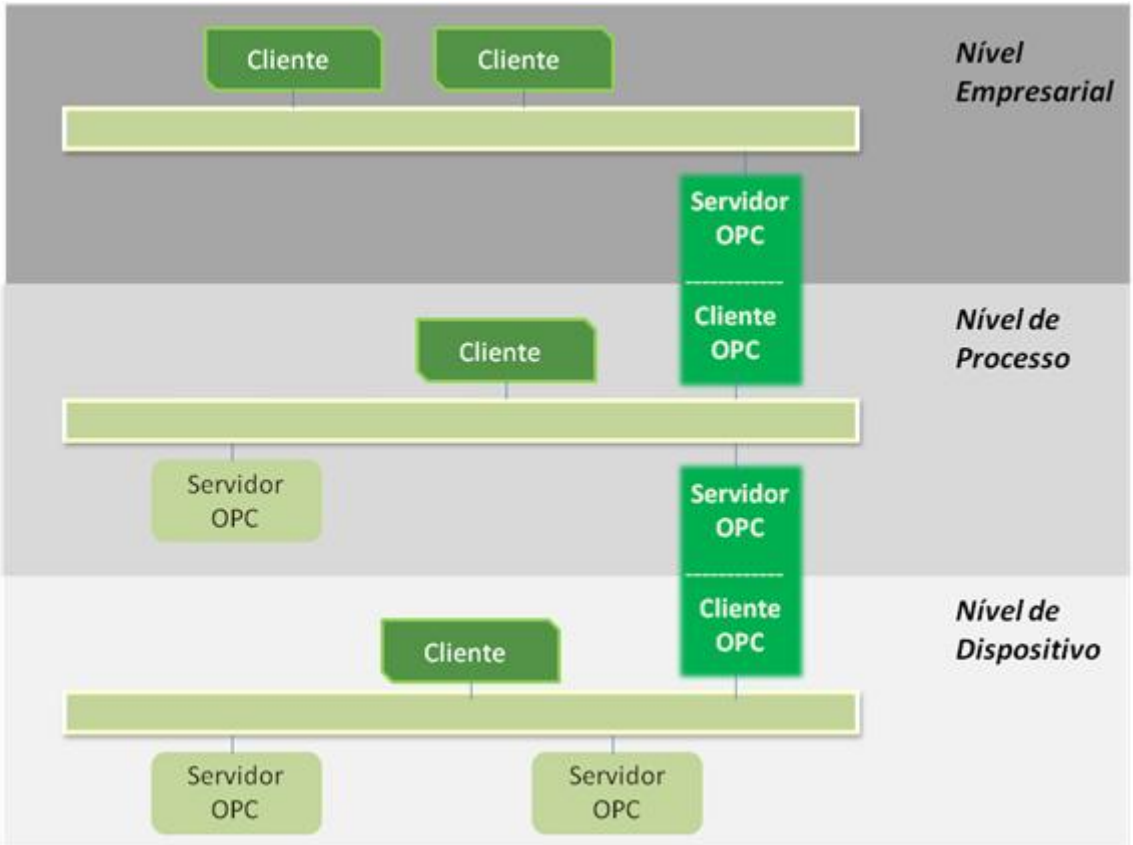

Figura 1.1: Camadas de comunicação entre Servidores e Clientes OPC UA

O padrão OPC UA (OLE for Process Control Unified Architecture) permite a integração dos dados de toda a empresa, desde o processo fabril, no chão de fábrica, aos

ERP (Enterprise Resource Planning) e setores corporativos. É uma tecnologia derivada do OPC COM/DCOM da Microsoft, mas é um padrão aberto, o OPC unifica a comunicação permitindo a fácil integração de diversos sistemas, desde instrumentos de campo até os sistemas de gerenciamento e de Gestão Corporativa.

O OPC UA é padronizado pela OPC Foundation, que é a responsável pela criação e divulgação das suas especificações aos seus membros. Esse padrão pode ser mapeado sobre uma variedade de protocolos de comunicação e os dados podem ser codificados de várias formas para permitir portabilidade e eficiência. Mantém assim a compatibilidade com os sistemas legados, como o próprio protocolo OPC COM/DCOM e outros protocolos muito utilizados, como o protocolo Modbus, que padroniza a comunicação no modelo mestre e escravo entre dispositivos em diferentes redes e barramentos, provendo uma comunicação Cliente/Servidor. Esse projeto se insere nesse ponto, criar compatibilidade de dispositivos apenas comunicação Modbus com esse novo padrão utilizando a linguagem C# e as suas bibliotecas.

1.1 Objetivo

Desenvolvimento de software em linguagem C# de um Servidor OPC UA para comunicação com diversos dispositivos CLP, com protocolo Modbus/Ethernet, em tempo real e visualização de variações dos valores obtidos pelos seus clientes.

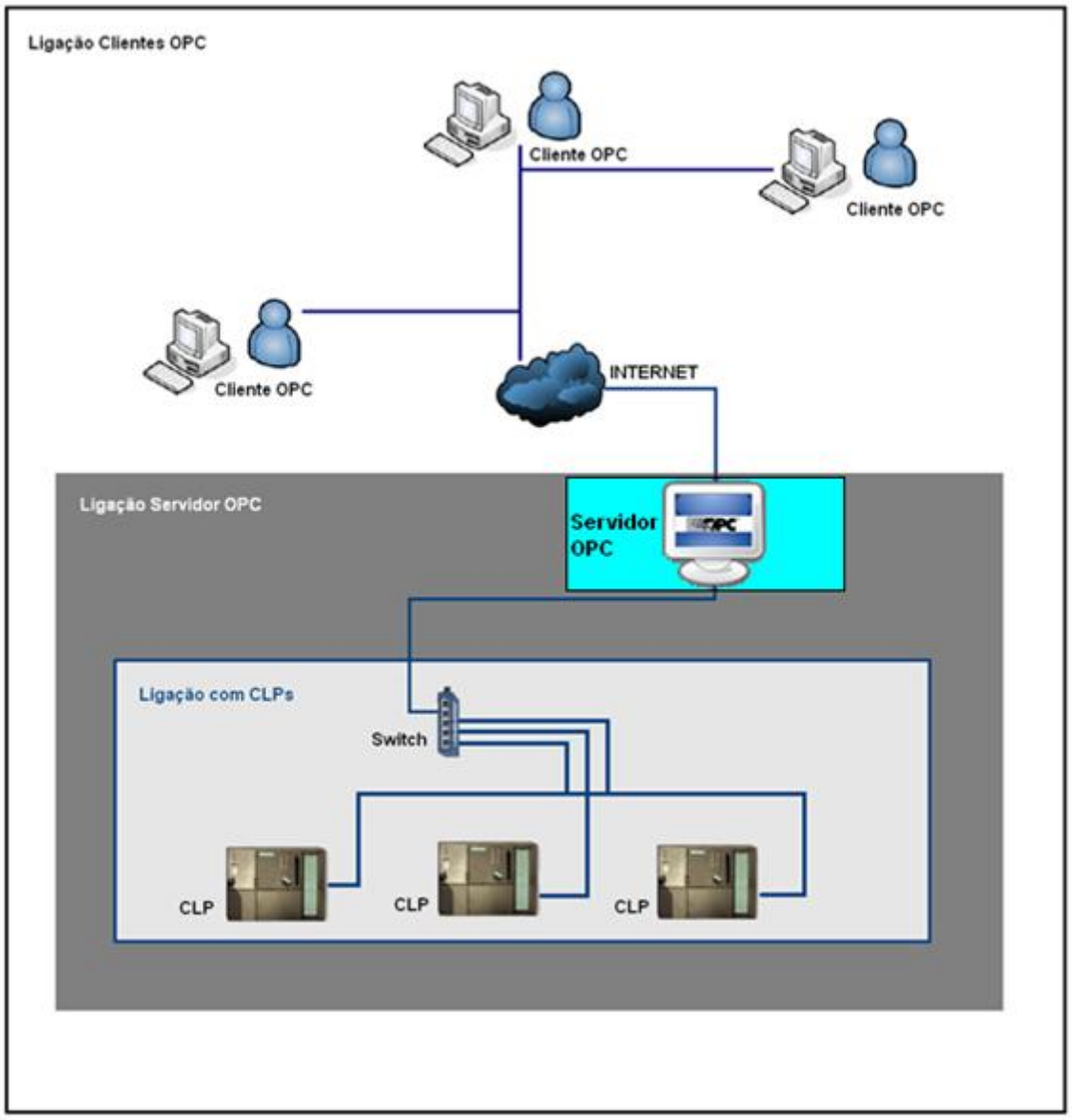

Figura 1.2: Comunicações Completa entre dispositivos CLP Modbus com conexão Ethernet utilizando tecnologia OPC UA para acesso de dados

2 OPC UA

2.1 Introdução

O OPC UA é um padrão independente de plataforma, pelo qual vários tipos de sistemas e dispositivos podem se comunicar pela troca de mensagens entre Clientes e Servidores sobre inúmeros tipos de redes. Suporta comunicação robusta e segura, que assegura a identidade dos Clientes e Servidores, resistindo a possíveis ataques. O OPC UA define conjuntos de serviços que Servidores podem fornecer e os Servidores individuais especificam o conjunto de serviços que suportam para seus Clientes. A informação é transmitida utilizando as definições de tipos de dados do OPC UA e de sistemas proprietários e os Servidores definem modelos de objetos que os Clientes podem descobrir dinamicamente. Os Servidores podem prover acesso a dados atuais e históricos, bem como alarmes e eventos para notificar Clientes de mudanças importantes (OPC FOUNDATION, 2010). O OPC UA pode ser mapeado sobre uma variedade de protocolos de comunicação e os dados podem ser codificados de várias formas para permitir portabilidade e eficiência.

Algumas vantagens da interface OPC são independência de fabricantes de hardware e software, simplicidade na configuração da informação a ser trocada, capacidade de rede, independência de protocolos e versões de controladores e ainda permite acesso simultâneo aos dados fornecidos pelo Servidor OPC.

O OPC UA define diversos modelos em sua especificação, sendo eles os seguintes:

I) Modelo de Segurança:

- Fornece o modelo para interações seguras entre clientes e servidores.

II) Modelo do Espaço de endereçamento:

- Descreve o conteúdo e a estrutura do espaço de endereçamento do Servidor.

III) Serviços:

- Especifica os serviços prestados pelo Servidor.

IV) Modelo de Informação:

- Especifica os tipos e seus relacionamentos definidos para os Servidores.

V) Mapeamentos:

- Especifica o mapeamento de transporte e codificação de dados suportados.

VI) Perfis:

- Especifica os perfis que estão disponíveis para Clientes e Servidores. Estes perfis fornecem grupos de serviços ou funcionalidades que podem ser utilizados para níveis de conformidade em certificações. Servidores e Clientes são testados de acordo com seus perfis.

2.2 Histórico

O protocolo OPC (OLE for Process Control) clássico foi desenvolvido em 1996 para solucionar problemas de interoperabilidade em sistemas de automação industrial, integrando dados entre os diversos níveis de suas redes. Essa primeira versão ligada a tecnologias proprietárias: OLE, COM e DCOM da Microsoft. Serve de interface para a troca de dados entre sistemas e foi especialmente desenvolvido para interfacear com máquinas, equipamentos e dispositivos, diretamente na camada de automação do chão de fábrica (OPC FOUNDATION, 2010).

A partir da primeira especificação foi desenvolvida a idéia de uma arquitetura unificada (Unified Architecture - UA), desvinculando o OPC de tecnologias proprietárias para que ele pudesse se tornar um padrão. Assim, em 2006 foi feito seu desenvolvimento em Java, .NET e C baseada em arquitetura orientada a serviço (SOA) e foram acrescentadas novas tecnologias como XML (Extensible Mark Up Language) (OPC FOUNDATION, 2010).

A especificação do OPC UA é divulgada em partes, e tendem a virar um padrão de fato no meio industrial.

2.3 Contexto

O OPC UA é aplicável no desenvolvimento de softwares em áreas de aplicação como dispositivos de campo, sistemas de controle, sistemas de gestão da produção e sistemas integrados de gestão. Estes sistemas destinam-se à troca de informações e no comando e controle de processos industriais. OPC UA não se limita a uma única hierarquia, assim, um Servidor OPC UA pode apresentar dados de diversas formas, adaptando-se a um determinado conjunto de Clientes normalmente costuma visualizar os dados. Esta flexibilidade, combinada com o suporte a definição de tipos, torna o OPC UA aplicável a uma ampla matriz de problemas de domínio. Como ilustrado abaixo, o OPC UA orientado para proporcionar maior interoperabilidade entre funções de baixo e alto nível.

O OPC UA é projetado para fornecer robustez dos dados publicados. Uma característica importante de todos os servidores OPC é a capacidade de publicarem dados e notificações de eventos. OPC UA prevê mecanismos para que os Clientes rapidamente possam detectar e recuperar falhas de comunicação associados a estas transferências sem necessidade de esperar por longos timeouts gerados por outros protocolos na rede.

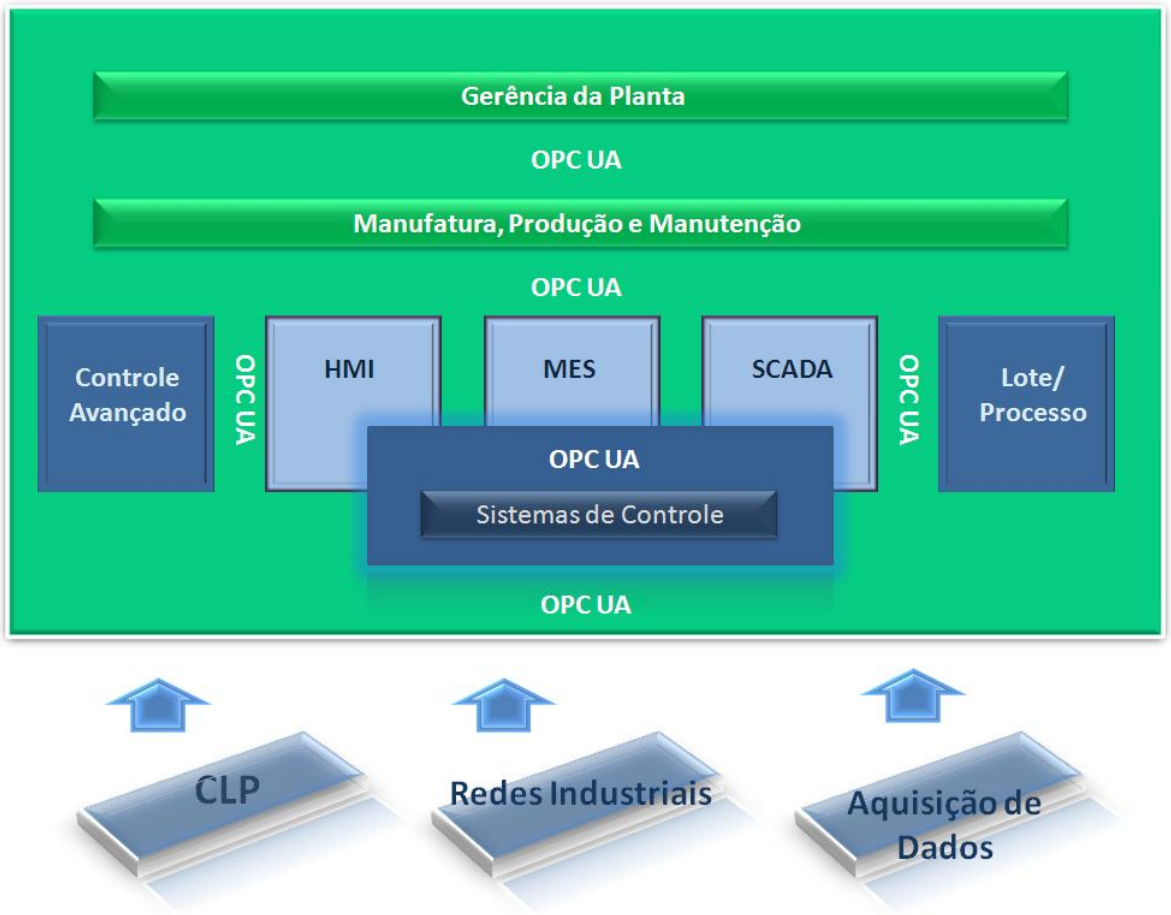

Figura 2.1: Aplicações OPC

OPC UA é projetado para suportar um grande número de Servidores, desde CLPs do chão de fábrica até Servidores corporativos. Estes Servidores são caracterizados por ampla capacidade, performance, plataforma de execução e funcionalidades. Sendo assim, o OPC UA define um conjunto amplo de recursos e os Servidores podem implementar um subconjunto destes recursos. Para promover a interoperabilidade o OPC UA define subconjuntos, designados como perfis, pelos quais os Servidores podem solicitar conformidade. Os Clientes podem então descobrir os perfis de cada Servidor e adequar suas interações com determinado Servidor de acordo com estes perfis.

2.4 Modelos Integrados e serviços fornecidos

Os requisitos mostrados a seguir, como Modelo de Segurança, Modelo do Espaço de Endereçamento, Modelo de Serviços e Sessões e que foram necessários no projeto, já estão implementados no SDK fornecido pela OPC Foundation. Sendo eles utilizados como ferramentas para criação do servidor.

2.4.1 Modelo de Segurança

A segurança do OPC UA é concebida para a autenticação de Clientes e Servidores, autenticação de usuários, integridade e confidencialidade de suas comunicações e para a verificação das alegações de funcionalidades (OPC FOUNDATION, 2010).

As medidas de segurança podem ser selecionadas e configuradas de acordo com as necessidades de uma determinada instalação. Este modelo de segurança inclui mecanismos de segurança e parâmetros. Em alguns casos, o mecanismo para troca de parâmetros de segurança é definido, mas a maneira na qual a aplicação utiliza estes parâmetros não. Este framework define, ainda, um conjunto mínimo de perfis de segurança que todos os Servidores UA suportam, mesmo que possam não ser utilizados em todas as instalações.

O Nível de segurança da aplicação depende de um canal de comunicação seguro que continua ativo pelo período da sessão da aplicação e assegura a integridade de todas as mensagens que são trocadas. Isto significa que o usuário necessita ser autenticado apenas uma única vez, enquanto a sessão da aplicação está ativa. Quando uma sessão é estabelecida, a aplicação Cliente e o Servidor negociam um canal de comunicação seguro e trocam entre si certificados de software que identificam o Cliente e o Servidor e as capacidades que cada um suporta. Autorizações geradas por certificados de software indicam os perfis do OPC UA que tais aplicações implementam e o nível de certificação de cada perfil.

O Servidor ainda autentica o usuário e autoriza pedidos subseqüentes para acessar objetos no Servidor. Mecanismos de autorização, como listas de controle de acesso, são especificos de cada aplicação ou sistema. OPC UA inclui suporte para auditoria de segurança e rastreabilidade entre Clientes e Servidores através de logs. Se um problema relacionado a segurança é detectado pelo servidor, o log associado ao Cliente pode ser localizado e examinado. OPC UA fornece ainda a capacidade para Servidores gerarem notificações de eventos que reportam eventos auditáveis para Clientes capazes de processá-los e gerar log destes. O OPC UA define parâmetros seguros de auditoria, que podem ser inclusos nos logs e nos eventos de notificação.

OPC UA complementa a infra-estrutura de segurança fornecida pela maioria das plataformas de serviços Web. O nível de segurança de transporte pode ser utilizado para criptografar e assinar mensagens. A criptografia e assinatura protegem as mensagens de divulgação da informação e a integridade das mensagens. A criptografia é fornecida por tecnologias de comunicação subjacentes utilizadas para troca de mensagens entre aplicações OPC UA.

2.4.2 Modelo Integrado do Espaço de endereçamento

O conjunto de objetos e informações relacionadas que o Servidor OPC UA torna disponível aos Clientes é referenciada ao seu espaço de endereçamento. O espaço de endereçamento do OPC UA representa seu conteúdo com um conjunto de nós conectados por referências. Características primitivas dos nós são descritas por atributos do OPC. Atributos são os únicos elementos de um Servidor que possuem valores de dados (OPC FOUNDATION, 2010). Tipos de dados que definem um atributo podem ser simples ou complexos.

Nós, no espaço de endereçamento, são conceituados baseados no princípio de Orientação a Objetos. Classes de nós definem os metadados para o espaço de endereçamento do OPC UA. A classes de nós básica define atributos comuns a todos os nós, permitindo identificação, classificação, e nomes. Cada classe de nós herda estes atributos e pode adicionalmente definir seus próprios atributos. Para promover a interoperabilidade de Clientes e Servidores, o espaço de endereçamento do OPC UA é estruturado hierarquicamente, com os níveis superiores sendo o mesmo para todos os Servidores. Os Servidores OPC UA podem organizar o espaço de endereçamento em um subconjunto de visualizações para simplificar ou limitar o acesso de Clientes.

2.4.3 Serviços Integrados

A interface entre Clientes e Servidores OPC UA é definida como sendo um conjunto de serviços. Estes serviços estão organizados dentro de grupos lógicos chamados de conjuntos de serviços. Os serviços fornecem duas capacidades aos Clientes, podendo emitir pedidos para os Servidores e receber a resposta destes. Eles permitem também a subscrição de Clientes para os Servidores, para notificações. As notificações são utilizadas pelo Servidor para reportar ocorrências, como alarmes, mudança de valores de dados, eventos e resultados da execução de programas (OPC FOUNDATION, 2010). As mensagens do OPC UA podem ser codificadas em texto XML ou em formato binário, para propósitos de eficiência. Elas podem ser transferidas através de múltiplos protocolos subjacentes como, por exemplo, o TCP ou os serviços Web sobre HTTP.

2.4.4 Sessões

O OPC UA exige um modelo repleto de estados. O estado de uma informação é mantido dentro de uma sessão da aplicação. Exemplos do estado da informação são subscrições, credenciais do usuário e pontos de continuação em operações em que se atende a múltiplos pedidos (OPC FOUNDATION, 2010).

Sessões são definidas como conexões lógicas entre Servidores e Clientes. Os Servidores podem limitar o número de sessões concorrentes, baseado na disponibilidade de recursos, restrições de licenciamento ou outros limitadores. Cada sessão é independente dos protocolos de comunicação subjacentes, falhas nestes protocolos não causam o encerramento imediato da sessão, o fim da sessão é baseado em solicitação de Servidor ou Cliente, ou na inatividade do Cliente. O tempo de timeout é definido durante o estabelecimento da sessão.

2.4.5 Redundância

O design do OPC UA assegura que desenvolvedores possam criar Clientes e Servidores redundantes de uma maneira coerente e robusta. A redundância pode ser utilizada para alta disponibilidade, tolerância a falhas e balanceamento de cargas (OPC FOUNDATION, 2010).

2.5 Arquitetura

A arquitetura do sistema OPC UA modela Clientes e Servidores interagindo como parceiros. Cada sistema pode conter múltiplos Servidores e Clientes. Cada Cliente pode interagir concorrentemente com um ou mais Servidores e cada Servidor pode interagir concorrentemente com um ou mais Clientes. Uma aplicação pode combinar componentes de Servidor e Cliente para permitir a interação com outros Servidores e Clientes. A Figura 2.2 ilustra a arquitetura que inclui uma combinação de Servidores e Clientes (OPC FOUNDATION, 2010).

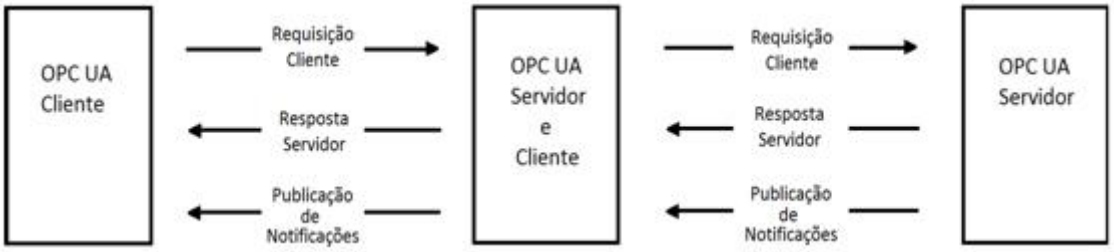

Figura 2.2: Arquitetura de Comunicação

2.5.1 Clientes OPC UA

A arquitetura do Cliente OPC UA modela o ponto final da interação entre Cliente/Servidor no lado do Cliente. A figura 2.3 ilustra os elementos principais de um típico Cliente OPC UA e como eles relacionam-se uns com os outros.

O aplicativo Cliente é o código que implementa a função de Cliente. Ele utiliza a API do Cliente OPC UA para enviar e receber requisições e respostas para o Servidor OPC UA. A API do Cliente OPC UA é uma interface interna que isola o código, da aplicação Cliente, da pilha de comunicação. A pilha de comunicação do OPC UA converte a API Cliente em mensagens e as envia, através da entidade de comunicação subjacente, para o Servidor na requisição da aplicação Cliente. A pilha de comunicação do OPC UA ainda recebe respostas e mensagens de notificação da entidade de comunicação subjacente e as entrega ao Cliente através da API Cliente.

2.5.2 Servidor OPC UA

A arquitetura do Servidor OPC UA modela o ponto final da interação entre Cliente/Servidor no lado do Servidor. A figura 2.4 ilustra os elementos principais de um típico Servidor OPC UA e como eles relacionam-se uns com os outros.

- Objetos Reais

  Objetos reais são objetos físicos ou objetos de software que são acessíveis pela aplicação Servidor ou que este mantém internamente. Exemplos incluem dispositivos

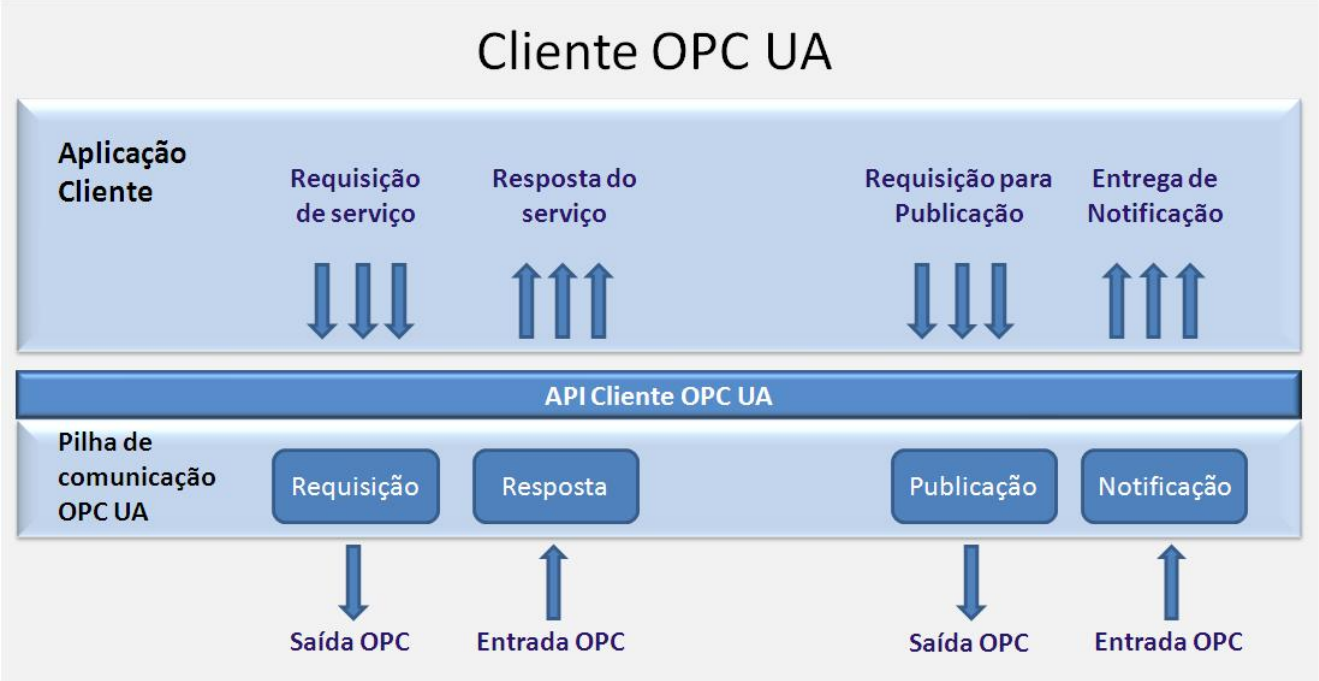

Figura 2.3: Arquitetura do Cliente OPC UA

físicos e contabilizadores de eventos.

- Aplicação Servidor OPC UA

  A aplicação Servidor OPC UA é o código que implementa a função de Servidor. Ele utiliza a API do Servidor OPC UA para enviar e receber mensagens de Clientes OPC UA. A API do Servidor OPC UA é uma interface interna que isola o código, da aplicação Servidor, da pilha de comunicação OPC UA.

- Espaço de Endereçamento OPC UA

  – Nós
  O espaço de endereçamento é modelado como um conjunto de nós acessíveis pelos Clientes usando os serviços do OPC (interfaces e métodos). Os nós no espaço de endereçamento são usados para representar objetos reais, suas definições e suas referências uns com os outros. A figura 2.5 mostra como os nós podem ser relacionados através de referências.

  – Organização do Espaço de endereçamento
  As especificações do OPC UA contém os detalhes do modelo de metadados para se construir blocos utilizados para criar um espaço de endereçamento fora dos nós interconectados de uma maneira consistente. Os Servidores possuem liberdade para organizar seus nós dentro deste espaço de endereçamento da forma que lhes for conveniente. O uso de referências entre nós permite aos Servidores organizar os espaços de endereços em hierarquias, uma rede completa de nós, ou outra possibilidade de mixagem. Nas especificações existem

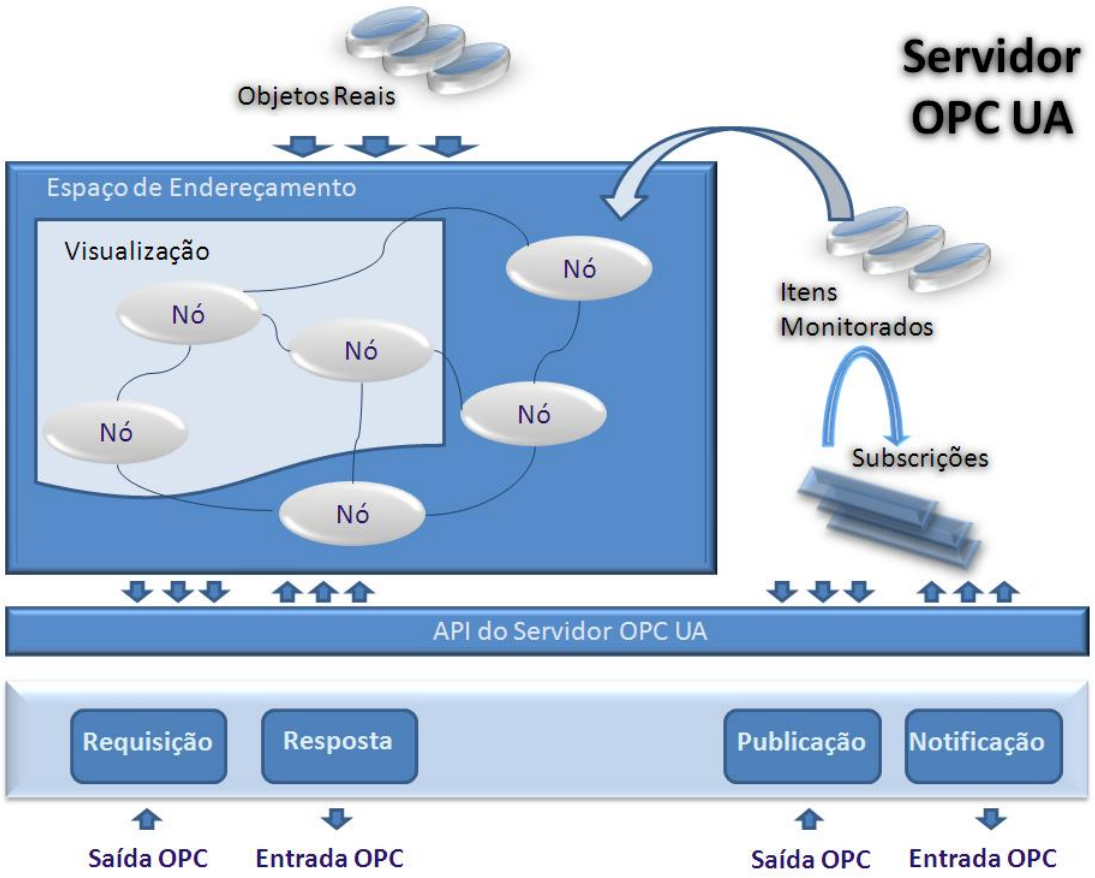

Figura 2.4: Arquitetura do Servidor OPC UA

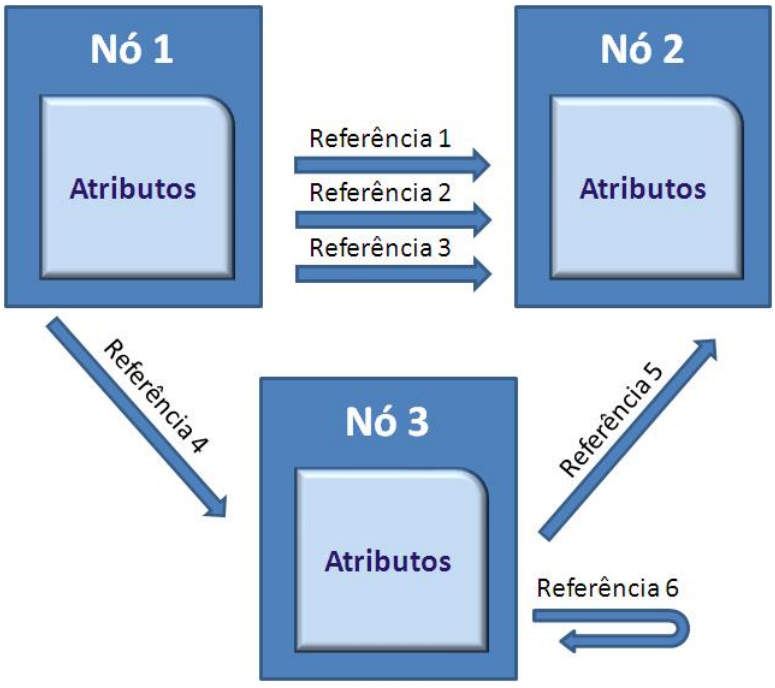

Figura 2.5: Exemplo de nós e de referências entre os nós

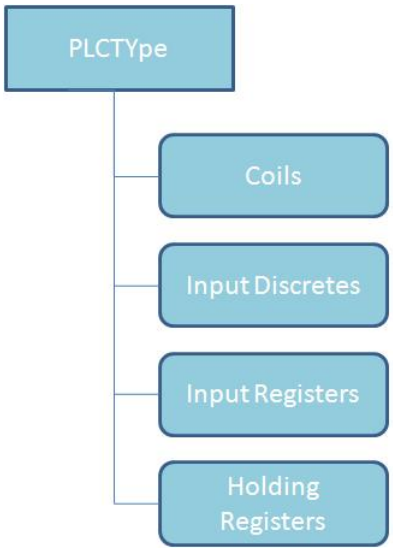

Figura 2.6: Exemplo de uma organização Espaço de Endereçamento de um Modelo de Informação

definições de nós e referências e suas organizações esperadas dentro do espaço de endereçamento.

– Visualizações do espaço de endereçamento

  Uma visualização é um subconjunto do espaço de endereçamento. As visualizações são utilizadas para restringir os nós que o Servidor torna visível ao Cliente, deste modo, restringindo o tamanho do espaço de endereçamento para as solicitações de serviços do Cliente. A visualização default é todo o espaço de endereçamento, podendo os Servidores, opcionalmente, definir outras visualizações. Visualizações escondem alguns nós ou referências no espaço de endereçamento. As visualizações são visíveis através do espaço de endereçamento e os Clientes estão habilitados a navegar por elas para determinar suas estruturas. Visualizações são freqüentemente hierarquias, que para os Clientes são fáceis de percorrer e representar como uma árvore.

– Suporte a modelos de informação

  O espaço de endereçamento do OPC UA suporta modelos de informação, sendo estes fornecidos através de:

  I) Referências aos nós, que permitem que objetos no espaço de endereçamento se relacionem uns com os outros.

  II) Nós com tipos de objetos, que fornecem informação semântica para objetos reais (definição de tipos).

  III) Definições do tipo de dados, mostrados no espaço de endereçamento para permitir que tipos de dados industriais específicos possam ser utilizados.

IV) Padrões complementares do OPC UA, que permitem grupos industriais específicos definirem como seus próprios modelos de informação devem ser representados no espaço de endereçamento dos Servidores OPC UA.

- Entidade editores/assinantes

  – Itens monitorados

    Itens monitorados são entidades no Servidor criadas pelo Cliente, que monitoram o espaço de endereçamento dos nós e suas contrapartes no mundo real. Quando estes detectam a ocorrência de uma mudança ou alarme/evento, eles geram uma notificação que é transferida para o cliente por uma subscrição.

  – Subscrições

    Uma subscrição é o ponto final no Servidor, que publica notificações para Clientes. Os Clientes controlam a freqüência destas publicações pelo envio de mensagem de solicitação de publicações.

    A figura 2.7 mostra a relação entre itens monitorados com subscrições:

- Interfaces de serviços OPC UA

  – Serviços de solicitação/resposta

    Serviços de solicitação e resposta são serviços chamados pelo Cliente OPC UA através da interface de serviços OPC para realizar uma tarefa específica, em um ou mais nós no espaço de endereçamento e que necessitam de uma resposta.

  – Serviços de notificação

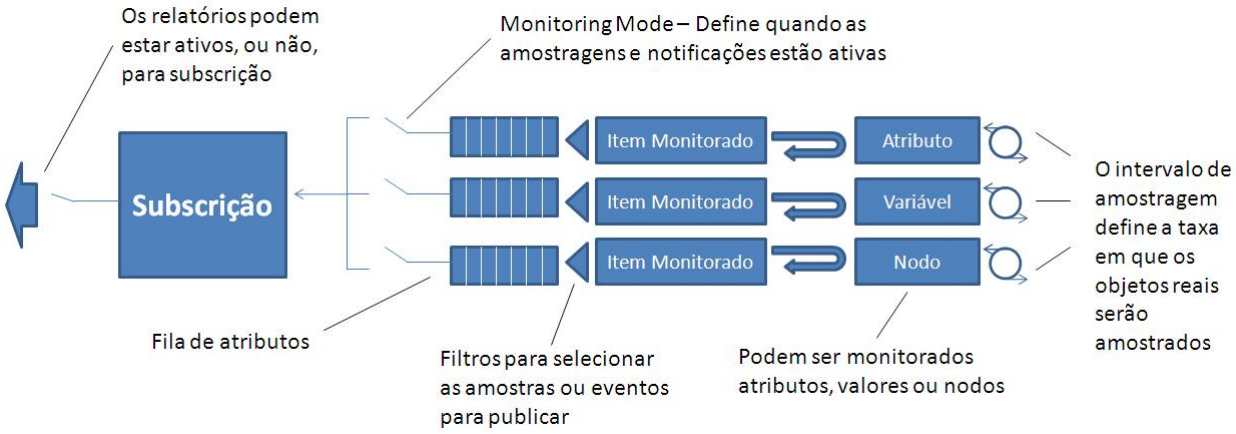

Figura 2.7: Subscrição

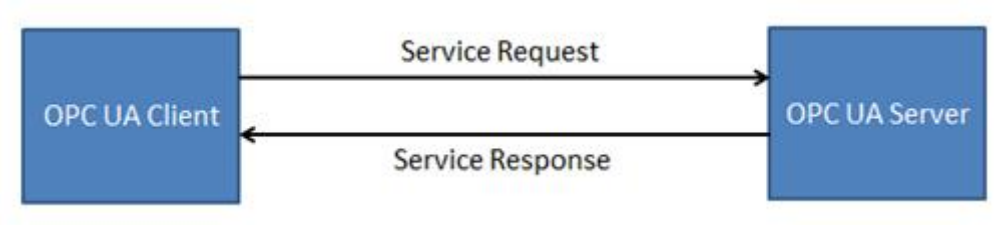

Figura 2.8: Troca de mensagens entre Cliente e Servidor

Serviços de notificação são serviços chamados através da interface OPC UA com o propósito de periodicamente enviar notificações aos Clientes. Notificações incluem eventos, alarmes, mudanças nos dados e saídas de programas.

- Interações de Servidor para Servidor
  Interações entre Servidores ocorrem quando um Servidor estabelece comunicação com o Cliente de outro Servidor. Estas interações permitem o desenvolvimento de Servidores que:

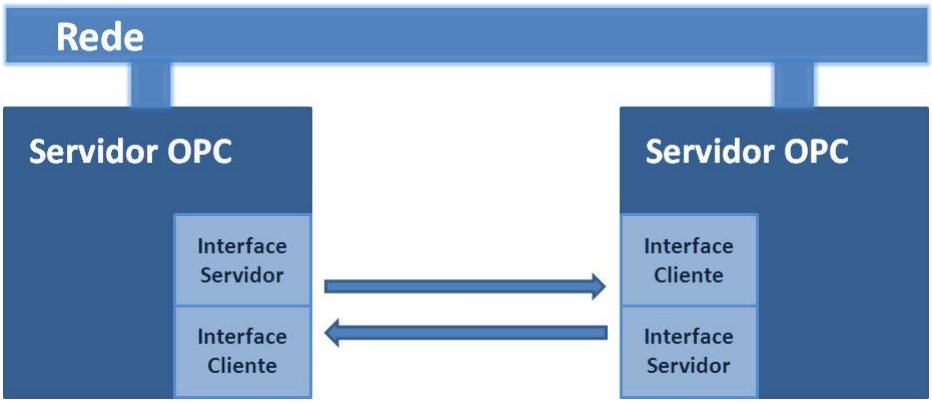

Figura 2.9: Interações peer-to-peer entre Servidores

I) MTrocam informações uns com os outros em uma conexão peer-to-peer, isto poderia incluir redundância ou Servidores remotos que são usados para manutenção ampla do sistema de definição de tipos.

II) São encadeados em uma arquitetura de Servidores por camadas para:

- Agregar dados de Servidores de camadas inferiores;
- Fornecer construtores para Clientes de camadas superiores;
- Fornecer interfaces centrais para Clientes como pontos únicos de acesso a múltiplos Servidores subjacentes.

2.6 Mapeamento da tecnologia OPC UA em outros protocolos

O OPC UA implementado diretamente em dispositivos de chão de fábrica, mas possui, também, a possibilidade de ser mapeado sobre outros protocolos já consolidados na indústria. Isso permite a compatibilidade com sistemas já em funcionamento, que a adoção ao protocolo OPC UA não cause prejuízo aos sistemas atuais.

Um dos protocolos mais utilizados pela indústria é o Modbus, que padroniza a comunicação no modelo mestre e escravo entre dispositivos em diferentes redes e barramentos. Ele é mostrado com detalhes no próximo capítulo, pois esse projeto, cria um Servidor OPC UA, que utiliza a comunicação Modbus para comunicação com os CLPs.

3 PROTOCOLO MODBUS

3.1 Introdução

É um protocolo da camada de aplicação - nível sete do sistema OSI -, que faz a comunicação com dispositivos de diversos barramentos e redes. Todo tipo de dispositivo pode usar o protocolo Modbus como, por exemplo, CLP (controlador Lógico Programável), HMI (Interface Homem-Máquina) e dispositivos de Entrada/Saída. É basicamente utilizado em chão de fábrica para comunicar unidades de terminal remoto (RTU) a Sistemas de Aquisição de Dados e Controle de Supervisão (SCADA) (MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b, 2006).

Modbus possui uma comunicação baseada na troca de mensagens entre dispositivos Cliente e Servidor, não guardando nenhum estado a respeito da conexão. Apenas o Cliente pode iniciar uma comunicação e o Servidor recebe as requisições que lhe foram feitas e as responde. Ele pode rodar sobre diversos meios físicos, o mais comum é utilizando interface serial, mas existem extensões para que o protocolo rode sobre redes Ethernet. O protocolo é de fácil implementação, fácil de ser usado e muito confiável, por isso é um dos protocolos de automação mais populares. Suas principais utilizações ocorrem nos próprios IAS (Sistemas de Automação Industrial) e também em BMS (Sistemas de Gerenciamento de Edifícios).

3.2 Histórico

Utilizado por milhares de dispositivos, Modbus foi desenvolvido para comunicar controladores da Modicon em 1979. Originalmente criado apenas para comunicação entre CLPs e dispositivos no chão de fábrica, o protocolo evoluiu e permite hoje comunicação através de uma rede Ethernet, a qual é a versão utilizada para o desenvolvimento desse projeto (MODICON MODBUS PROTOCOL REFERENCE GUIDE, 1996).

Modbus Padrão, primeira versão do protocolo, foi desenvolvido para fazer a comunicação mestre e escravo entre os CLPs e os dispositivos de entrada e saída, instrumentos eletrônicos e atuadores de válvula, tendo como comunicação física, por exemplo, a interface serial RS-232 ou RS-485. Ele foi aprimorado para o Modbus Plus, podendo ser utilizado para comunicar CLPs não só com dispositivos de entrada e saída, mas também com HMI (interfaces homem máquina). Nesse caso, o meio de comunicação física é a serial RS-485 com controle de acesso ao meio por HDLC (High Level Data Link Control), permitindo uma comunicação mais rápida entre distâncias maiores. Por outro lado,

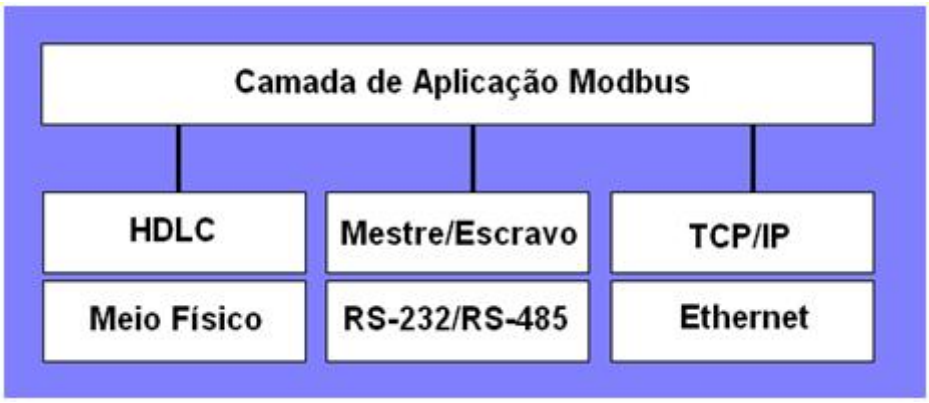

Figura 3.1: Esquemático com tipos de protocolos Modbus citados em relação a camada de aplicação Modbus

o Modbus TCP/IP é utilizado para comunicar CLPs aos sistemas de supervisão, sendo esse protocolo encapsulado pelo TCP/IP - transmitido pela interface Ethernet - e tendo uma porta reservada para sua comunicação: a porta 502.

3.3 Contexto

Todos dispositivos de chão de fábrica podem utilizar o protocolo Modbus - sensores, HMI, motores e CLPs - para comunicação entre eles e operações remotas. Além disso, como foi observado, tais dispositivos podem utilizar tanto linhas seriais como a rede Ethernet. O protocolo faz a troca de mensagens de maneira rápida e eficiente entre esses dispositivos, permitindo uma grande integração e aumentando muito a eficiência da produção. A figura 3.2 mostra um esquemático de comunicação entre diferentes dispositivos com diferentes interfaces físicas, mas todos utilizando o protocolo Modbus para fazer as suas trocas de mensagens.

3.4 Funcionamento

Baseado na troca de mensagens entre Cliente e Servidor, o protocolo define uma PDU (Protocol Data Unit), que é a formada pelo código da função e por uma área de dados, que vem a ser encapsulada em uma ADU (Application Data Unit).

3.4.1 Troca de Mensagens

O Cliente é o iniciador da conversa, sendo ele quem forma a PDU para enviá-la ao servidor. O código da função indica qual o tipo de ação está sendo solicitada pelo Cliente, e essa ação pode requisitar algum tipo de dado extra como, por exemplo, o endereço da bobina a ser lida no CLP (MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE V1.0b, 2006). Essa informação complementar, mas necessária, é transmitida na área de dados da requisição, podendo ser nula, caso a ação não necessite de nenhuma informação além do código.

São reservados 8 bits para representar o código, portanto são permitidos números entre

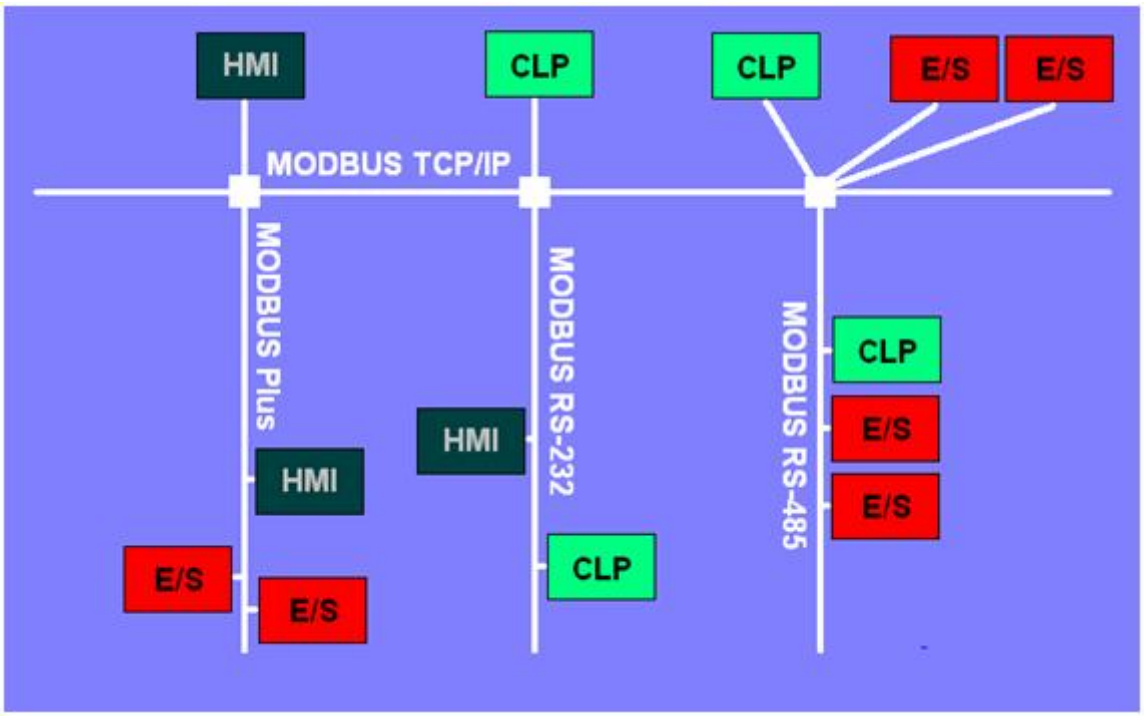

Figura 3.2: Dispositivos conectados utilizando o protocolo Modbus

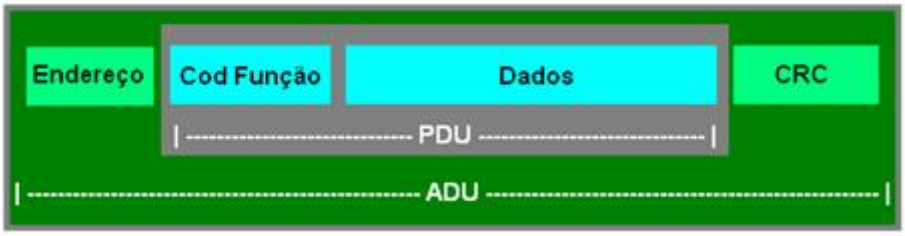

Figura 3.3: ADU e PDU Modbus

1 e 255 no campo de código. Desses, apenas de 1 a 127 são para solicitações (o bit mais significativo sempre em 0). O código "0"não é permitido e os outros valores são reservados resposta do Servidor para indicação de erro (exception responses), isto é, quando o bit mais significativo está ligado, é sinal de que houve algum erro no processamento da requisição.

A resposta do Servidor, caso não haja nenhum tipo de erro durante o processo, deve conter o mesmo código da requisição feita pelo Cliente. Caso ocorra algum erro, o Servidor deve retornar o código somado de 128, isto é, o mesmo código da requisição, mas com o bit mais significativo em "1", informado que houve erro. A figura 3.4 mostra um exemplo, no qual o cliente faz requisição de leitura das entradas (código 2) e ocorre um erro, sendo assim, o servidor responde com o código 130 (código + 128).

3.4.2 Estrutura dos dados

As operações básicas são de escrita e leitura de registradores ou bobinas, portando os dados são: Bobinas (Coils), Entradas Discretas, Registradores de Entrada e Registradores

Figura 3.4: Resposta com erro de uma requisiçao

Intermediários. Seus tipos e representações são descritos na tabela 3.1:

Dados	Representação	Tipo	Observações
Entradas Discretas (Discrete Inputs)	Um bit	Somente-Leitura	Pertencem as entradas e saídas do sistema
Bobinas (Coils)	Um bit	Leitura-Escrita	Podem ser alteradas por programa
Registradores de Entrada (Input Registers)	Palavra de 16 bits	Somente-Leitura	Pertencem as entradas e saídas do sistema
Registradores de Espera (Holding Registers)	Palavra de 16 bits	Leitura-Escrita	Podem ser alteradas por programa

Tabela 3.1: E/S e Registradores Disponíveis no Protocolo Modbus

Todos esses dados estão mapeados na memória dos dispositivos e a quantidade máxima endereçável das variáveis é de 65536, mas cada CLP tem uma quantidade específica menor que essa.

3.4.3 Funções

São divididas em três categorias: públicas (são funções bem-conhecidas), definidas pelo usuário (não são definidas pela especificação) e reservadas (para fins de compatibilidade com outras tecnologias) (MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE V1.0b, 2006).

O protocolo define várias funções públicas, sendo várias delas podendo ser utilizadas apenas por dispositivos com comunicação serial, não sendo relevantes para esse trabalho,

pois a troca de informações será feita apenas pela interface Ethernet.

Na área de dados da PDU da requisição pode ser informado, dependendo da requisição, os seguintes campos:

Campo	Função
Start Address (Endereço de Inicio)	Indicar a partir de qual endereço deve ser feita a leitura do bloco.
Quantity (Quantidade)	Número de variáveis a serem lidas, no caso do Read Coils, por exemplo, o número de bobinas.

Tabela 3.2: Principais Campos das funções do Protocolo Modbus

4 DESENVOLVIMENTO DO SERVIDOR OPC UA COM COMUNICAÇÃO MODBUS

4.1 Introdução

O desenvolvimento do Servidor OPC UA com comunicação com dispositivos Modbus TCP/IP pode ser divido em três módulos: de um driver de comunicação com o protocolo Modbus, a criação do Espaço de Endereçamento próprio, a comunicação com Clientes OPC UA, figura 4.1.

Figura 4.1: Módulos do Servidor OPC UA

4.2 Desenvolvimento da comunicação com Modbus TCP/IP

Para realizar a comunicação com o protocolo Modbus, foi utilizada a biblioteca NModbus, em C#. Ela fornece classes que fazem a conexão com dispositivos em uma porta definida pelo usuário, além dos métodos de leitura e escrita, definidos pelo protocolo. Um diagrama de classes mostra os principais métodos e propriedades utilizados da Classe ModbusIPMaster, Figura 4.2.

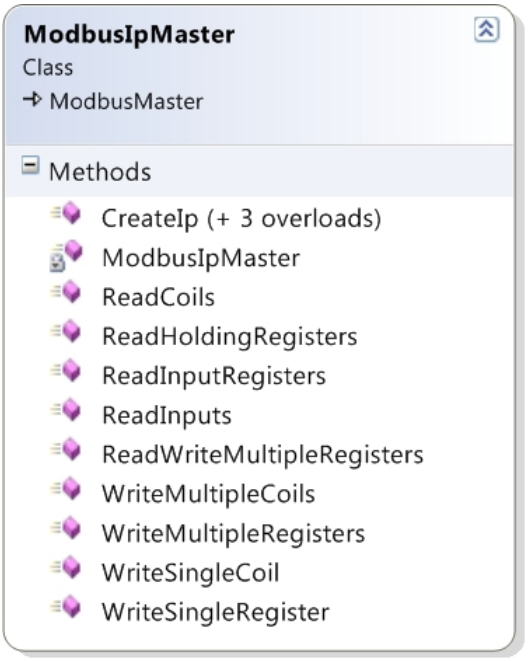

Figura 4.2: Diagrama de Classes da ModbusIpMaster

Dessa forma foi necessária uma adaptação para o projeto, alguns métodos que não seriam utilizados, figura 4.2, sendo, então, criada a classe ModbusChannel. Essa classe criada é responsável pela criação de um canal de comunicação com um dispositivo CLP através do protocolo Modbus. Ela permite a realização de leituras e escritas em um CLP.

Os métodos criados na classe possuem como parâmetros básicos o endereço de início de leitura e o número de variáveis a serem lidas, conforme é especificado no protocolo Modbus.

4.2.1 Operações de Leitura

ReadCoils (startAddress, numCoils) Faz a leitura dos valores de um bloco bobinas (Coils), o bloco é definido pelo endereço de início de leitura, startAddress, e o número de bobinas a serem lidas, numCoils.

ReadDiscreteInputs (startAddress, numInputs) Faz a leitura de um bloco de entradas do CLP, o bloco é definido pelo endereço de início de leitura, startAddress, e o número de inputs a serem lidos, numInputs.

ReadHoldingRegisters (startAddress, numberOfPoints) Faz a leitura de um bloco Holding Registers, o bloco é definido pelo endereço de início de leitura, startAddress, e o número de Holding Registers a serem lidos, numberOfPoints.

ReadInputRegisters (startAddress, numberOfPoints) Faz a leitura de um bloco de registradores, o bloco é definido pelo endereço de início de leitura, startAddress, e o número de registradores a serem lidos, numberOfPoints.

4.2.2 Operações de Escrita

WriteSingleCoil (coilAddress, value) Escreve valor em uma única bobina (Coil), o endereço da bobina é definido pelo coilAddress e o valor a ser escrito, pelo value.

WriteSingleRegister (registerAddress, value) Escreve o valor em um único Holding Register, o endereço da bobina é definido pelo registerAddress e o valor a ser escrito, pelo value.

WriteMultipleCoil (coilAddress, data) Escreve valores em um bloco de bobinas, o bloco é definido pelo endereço de início de leitura, coilAddress, e os valores a serem escritos no data.

WriteMultipleRegisters (registerAddress, data Escreve valores em um bloco contínuo de registradores, o bloco é definido pelo endereço de início de leitura, coilAddress, e os valores a serem escritos no data.

4.2.3 Operações de Leitura/Escrita em uma única operação

ReadWriteMultipleRegisters Combinação de leitura e escrita em uma única operação Modbus, a função de escrita é feita antes da função de leitura, o bloco de leitura é definido pelo endereço de início de leitura, numberOfPointsToRead, e a quantidade de registradores a serem lidos pelo numberOfPointsToRead, o startWriteAddress, define início do bloco de escrita e os dados a serem escritos no writeData.

4.3 Desenvolvimento de Servidor OPC UA

O Servidor OPC UA é formado por três módulos básicos: o Espaço de Endereçamento, a comunicação com Clientes OPC UA e a comunicação com dispositivos Modbus. Para criar um Servidor OPC UA básico foi utilizada uma ferramenta da OPC Foundation, a qual disponibiliza aos seus membros um SDK para auxiliar o desenvolvimento de produtos com tecnologia OPC UA. Esse kit fornece bibliotecas em C#, juntamente com alguns exemplos de funcionalidades básicas para conexão entre Clientes e Servidores em formato executável, permitindo assim o teste do que for produzido, além de exemplos para acesso a dados histórico, eventos e alarmes - funcionalidades não incorporadas a esse projeto. Essas bibliotecas permitem comunicação tanto com softwares trabalhando com a antiga arquitetura COM/DCOM quanto com a nova, OPC UA.

O primeiro objetivo aqui é criar um Servidor OPC UA básico utilizando as ferramentas e exemplos fornecidos pelo SDK e acoplar os módulos citados acima. Além disso, a OPC Foundation indica uma ferramenta, a "OPC UA Address Space Model Designer"para criação de um Espaço de Endereçamento próprio. Esta é uma ferramenta visual que gera automaticamente as classes para serem adicionadas ao projeto, bastando ser referenciada. Assim, com o SDK e a ferramenta de modelagem de Espaço de Endereçamento, foi criado um Servidor OPC UA básico apenas para comunicação com outro Cliente OPC UA, ainda sem a comunicação com dispositivos Modbus TCP/IP.

4.3.1 Criação de um Servidor Básico

Foi criado um Servidor sem informações, apenas com a estrutura e as funcionalidades básicas de conexão OPC UA, necessárias para que fosse possível comunicação de um Cliente com o Servidor utilizando o protocolo OPC UA.

4.3.2 Criação de um Espaço de Endereçamento Próprio

Para que os dados de um CLP sejam lidos corretamente no servidor, é necessário que as suas estruturas de Entrada e Saída sejam mapeadas no Espaço de Endereçamento do servidor, figura 4.3. Assim, foi criado - utilizando a ferramenta "OPC UA Address Space Model Designer- um Espaço de Endereçamento próprio com os dados relativos a um CLP: Bobinas (Coils), Entradas Discretas (Digital Inputs) e Registradores de Espera (Holding Registers), isto é, a criação de um tipo CLP genérico, com todas as variáveis de um CLP real.

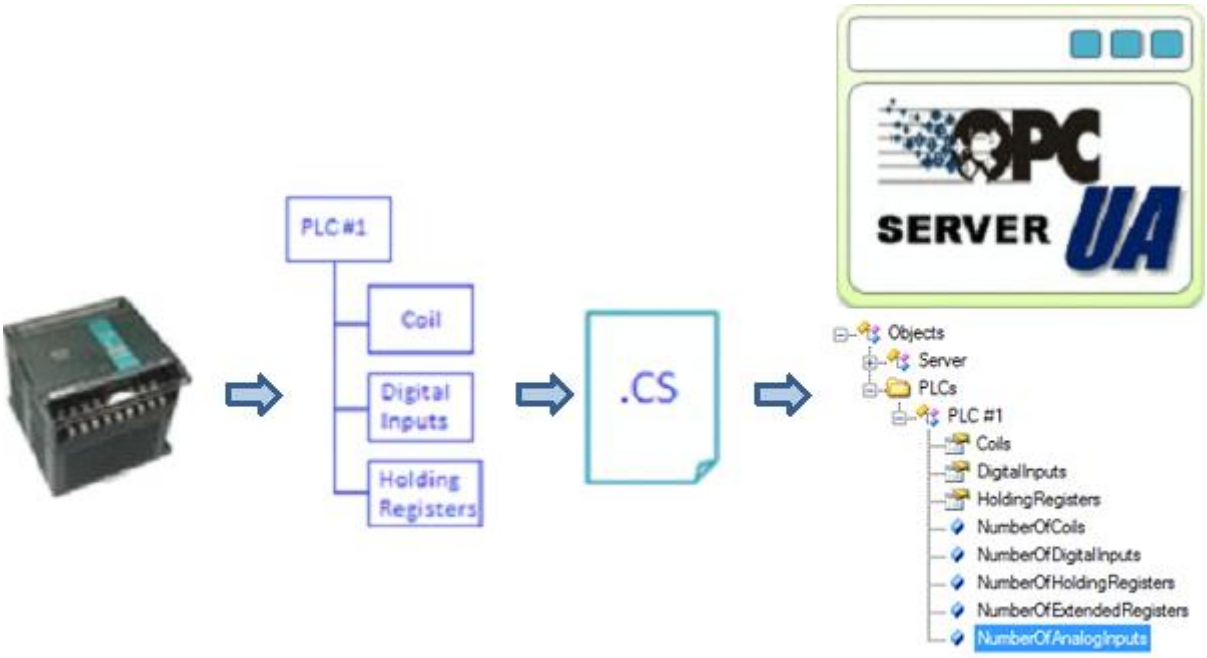

Figura 4.3: Processo de mapeamento do CLP para um espaço de endereçamento, usando OPC UA Address Space Model Designer

4.4 Conexão Servidor OPC-ModBus

4.4.1 Incorporação da classe ModbusChannel

Com a estrutura de um servidor OPC UA pronta e tendo comunicação com clientes OPC UA, é necessário conectá-lo diretamente a um dispositivo CLP Modbus TCP/IP. Para isso, é feita a incorporação da classe ModbusChannel, mostrada anteriormente no servidor OPC UA básico.

Foi criada uma classe com a estrutura de um CLP, PLCItem, com as suas características, como: Nome, IP, Porta e Intervalo de Amostragem.Como a quantidade de dispositivos Modbus conectados ao servidor pode ser variável, foi criada uma estrutura de dicionário (Dictionary) para os CLPs.

4.5 Conexão do Servidor ModbusToOPCUA com Cliente OPC UA

4.5.1 Estrutura de comunicação completa - ModbusToOPCUA

Projeto
Para os primeiros testes foi utilizado um simulador Modbus TCP/IP.

Cliente OPC UA
Para fazer a comunicação com o servidor criado, foi utilizado um servidor para testes, fornecido pela OPC Foundation.

Utilização de Simulador
Foi utilizado um simulador Modbus TCP/IP para os primeiros testes, figura 4.4, nele é possível configurar a porta que ele irá trabalhar e o IP é igual ao da máquina que ele está rodando. Também é possível alterar valores individuais de cada bobina ou registrador.

Todos os tipos de E/S podem ser manipulados no simulador, além disso, há opção de uma variação constante dos valores de entrada e saída, para uma verificação em tempo real do funcionamento do servidor.

4.5.2 Configurando CLP no Servidor OPC UA:

- Servidor em funcionamento
  - A tela inicial do servidor possui duas abas, figura 4.7: a de configuração e a de gerenciamento de dispositivos. Na de configurações existem opções para adicionar e remover CLPs do servidor.
  - Caso seja selecionada a opção para adicionar um CLP, é necessário fornecer as configurações, figura 4.5, dele como: intervalo de consulta dos valores de E/S (taxa de amostragem), número de bobinas, de entradas analógicas e de registradores. Além disso, é necessário colocar as informações de conexão com o CLP, como o seu endereço IP, a porta de comunicação em que ele envia e recebe os dados e o nome que ele receberá no Servidor. Feito isso, deve ser feita o teste da conexão com o aparelho, figura 4.6.

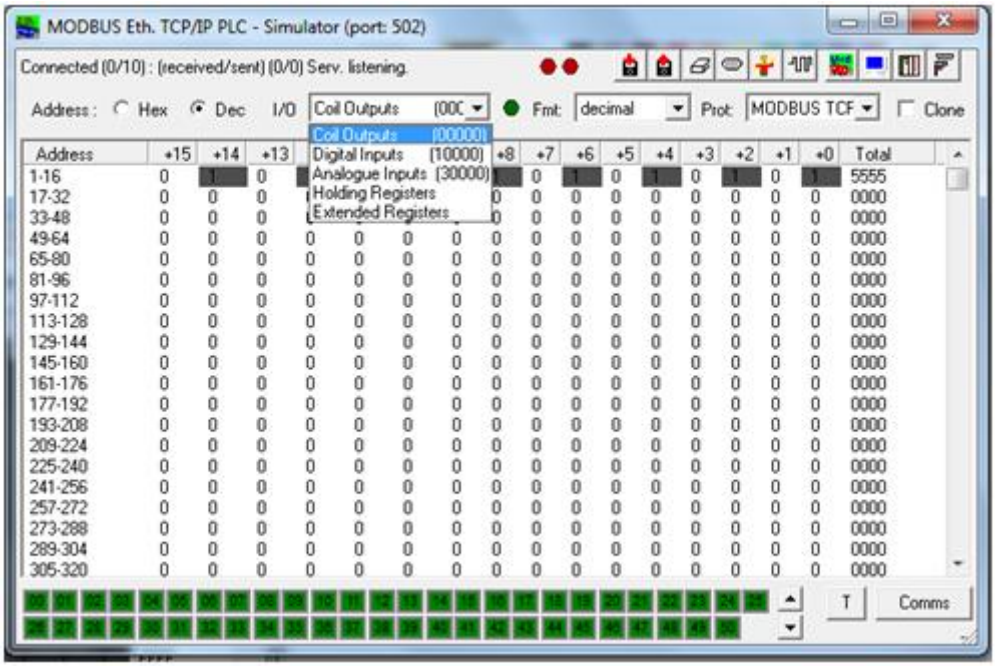

Figura 4.4: Seleção do tipo de E/S que será simulada

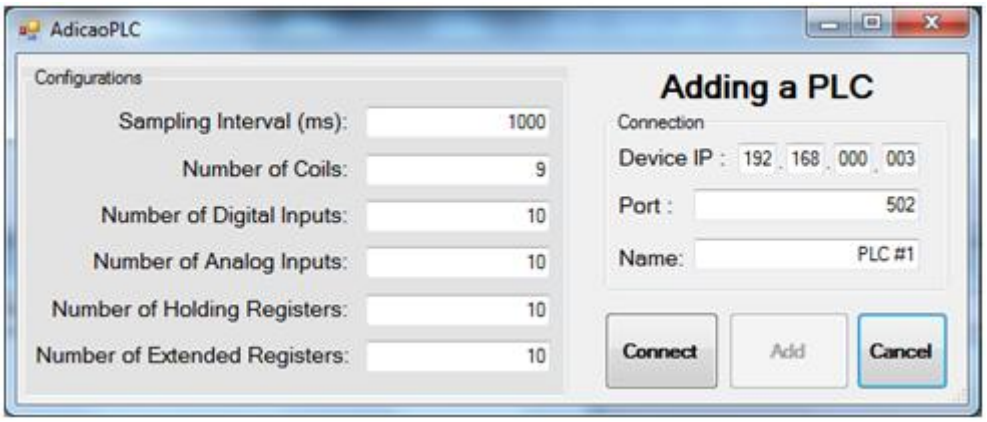

Figura 4.5: Teste de conexão com CLP

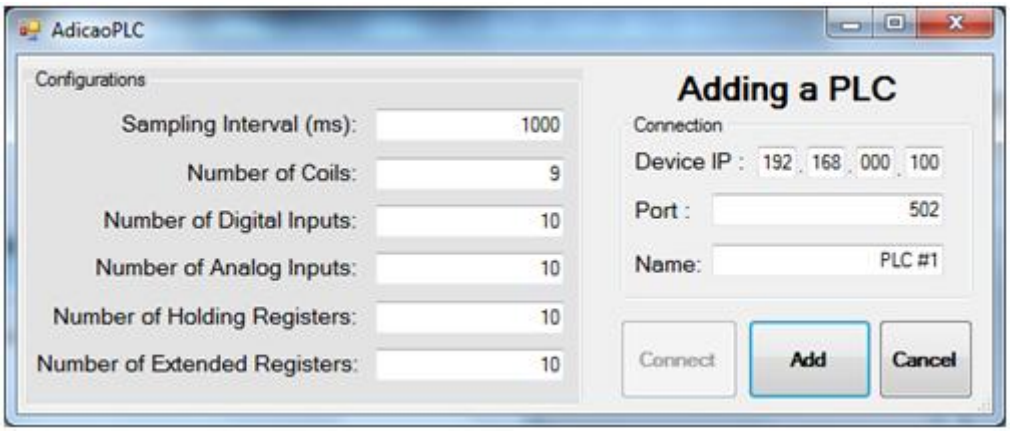

Figura 4.6: Tela após a confirmação de conexão, só então é possível adicionar o CLP ao Servidor.

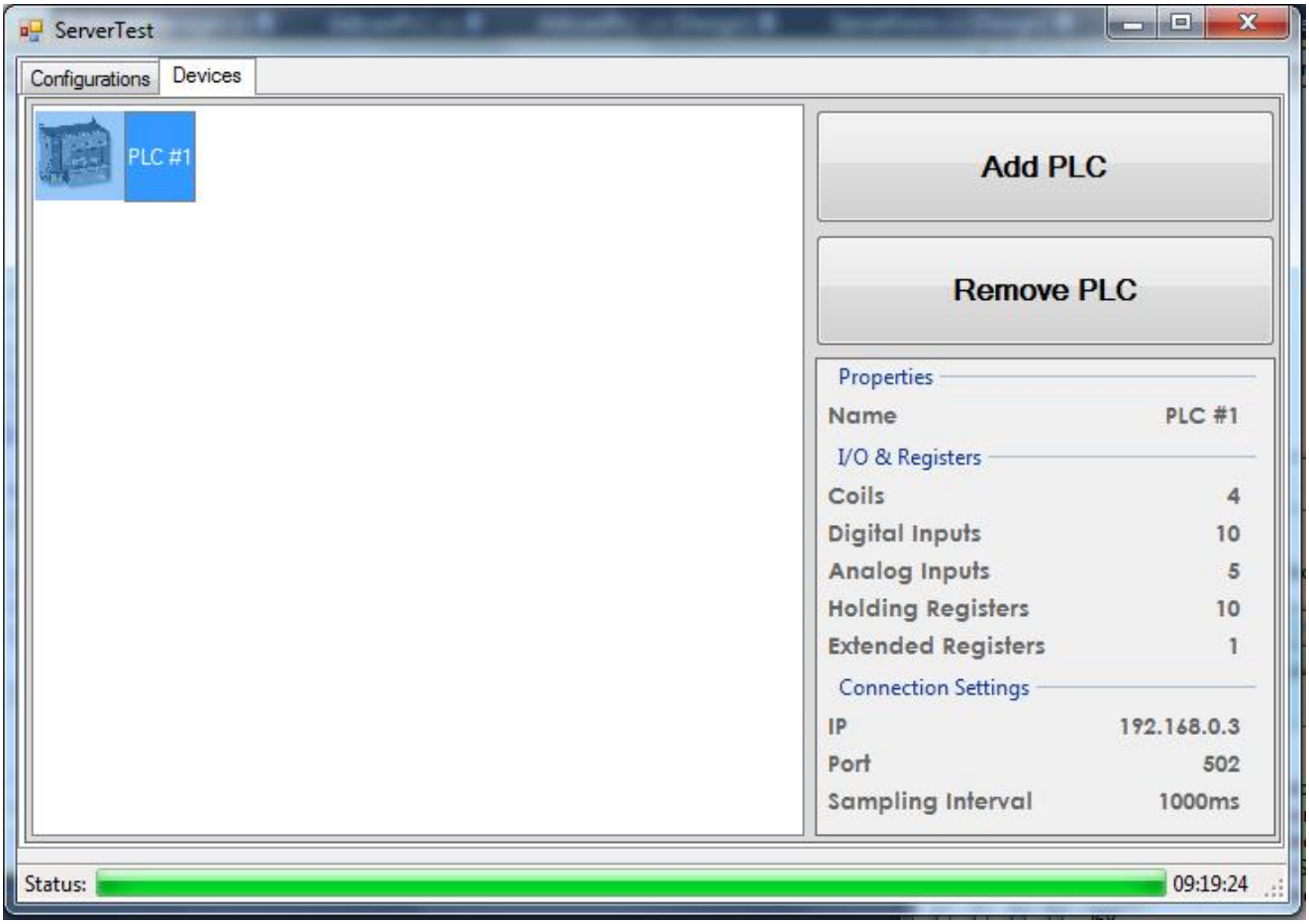

Figura 4.7: Tela após ser adicionado o CLP, ele aparece como um ícone e as suas propriedades podem ser visualizadas

- Visualização do espaço de endereçamento criado
  - É parte final da comunicação;
  - Com a comunicação entre dispositivo Modbus TCP/IP e o servidor sendo completa, foi utilizado o Cliente OPC UA fornecido pelo SDK, para visualizar o Espaço de Endereçamento do CLP criado no servidor;

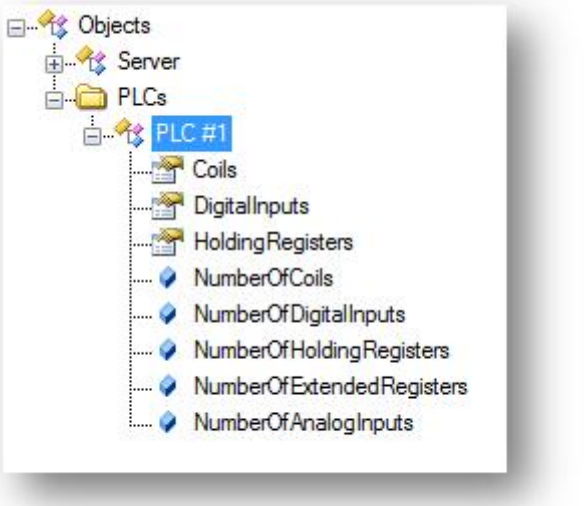

Figura 4.8: Espaço de Endereçamento do exemplo, com um CLP chamado de PLC #1, visto pelo Cliente

5 RESULTADOS

5.1 Utilização do Servidor OPC UA com um CLP real

Com a implementação do Servidor OPC UA finalizada, foram feitos testes com simulador Modbus TCP/IP. Após, foi utilizado um CLP FBS- 24MATJ, fabricado pela empresa chinesa Fatek Corp. e distribuído pela Altus Sistemas de Informática. O CLP possui 10 bobinas e 14 entradas discretas.

Com os testes, foi possível tanto escrever as bobinas quanto ler as bobinas e as entradas. Sendo assim, constatado o correto funcionamento do projeto do servidor, que quando ligado a um CLP com comunicação Modbus TCP/IP, disponibilizou os dados no formato OPC UA para clientes através da internet em tempo real. Foi feita ainda uma análise de tempo de resposta, apresentada na seção abaixo.

5.2 Análise de tempo de resposta

Para realizar uma análise temporal, foi criado um código em C#, figura 5.2, testando a comunicação com o CLP através da rede. Esse código faz uma seqüência de amostragens e mostra o tempo gasto para realizar tais operações, figura 5.1.

Os testes foram realizados com seqüências de 1000 amostras consecutivas. Embora outros fatores possam influenciar o tempo de resposta do sistema, como tempo de processamento e o gargalo da comunicação com próprio CLP, os resultados mostrados foram relativamente bons.

Número de leituras consecutivas	Tempo de resposta (segundos)
1	00.0690039
10	00.6430368
100	06.5913770
1000	66.2167873

Tabela 5.1: Tabela com os tempos de resposta em relaçao ao número de leituras consecutivas no PLC

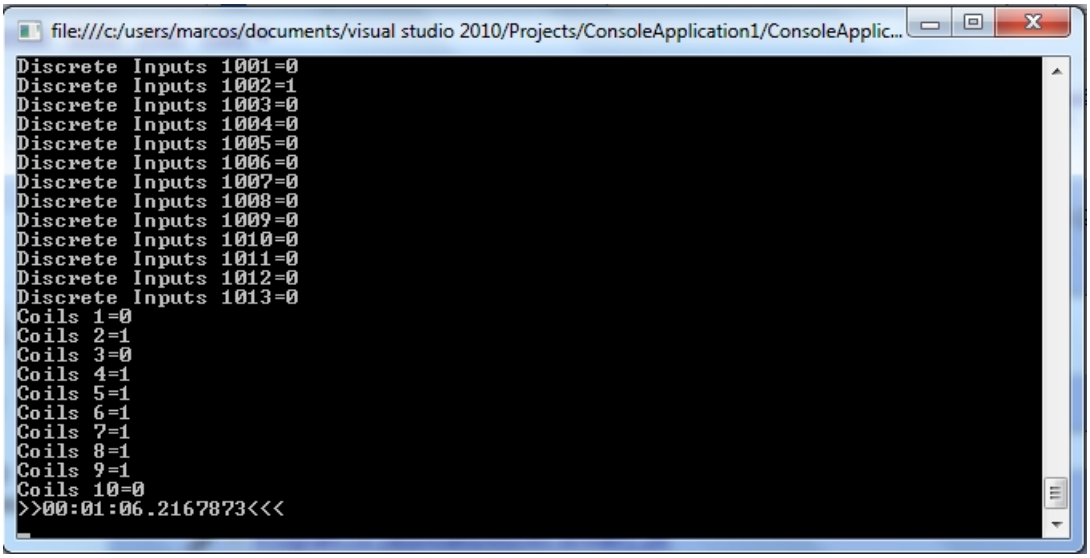

Figura 5.1: Resultado da simulação de tempo para 1000 amostras consecutivas de cada E/S do CLP FBS-25MATJ

```
int amostras = 1000;
ushort startAddressDI = 1000;
ushort numInputsDI = 14;
ushort startAddressC = 1;
ushort numInputsC = 10;
String deltaTempo;
DateTime tempo_2;
DateTime tempo_1 = DateTime.Now;


for (int j = 0; j < amostras; j++)
{
    bool[] inputsDI = PLC.ReadCoils(startAddressDI, numInputsDI);
    bool[] inputsC = PLC.ReadCoils(startAddressC, numInputsC);

    for (int i = 0; i < numInputsDI; i++)
        Console.WriteLine("Discrete Inputs {0}={1}", startAddressDI + i, inputsDI[i] ? 1 : 0);

    for (int i = 0; i < numInputsC; i++)
        Console.WriteLine("Coils {0}={1}", startAddressC + i, inputsC[i] ? 1 : 0);
}

tempo_2 = DateTime.Now;
TimeSpan timeDiff = tempo_2.Subtract(tempo_1);
deltaTempo = timeDiff.ToString();

Console.WriteLine(">>" + deltaTempo + "<<<");

Thread.Sleep(250000);
```

Figura 5.2: Código utilizado para realizar análise de tempo da comunicação com CLP real

6 CONCLUSÕES

Este trabalho apresentou o processo de desenvolvimento de um software Servidor OPC UA em linguagem C# com comunicação com o protocolo Modbus TCP/IP. Para tal, foi especificada a tecnologia OPC UA e seu histórico, assim como a estrutura do protocolo Modbus- como foco na comunicação via Ethernet TCP/IP-, os passos para sua criação e um exemplo ilustrativo.

Com o projeto finalizado, foi possível ter uma integração completa dos dados de CLPs com Clientes OPC UA pela internet. Isso possibilitou, não apenas a visualização dos dados, mas também o envio de informações para esses CLPs. Isto é, como as informações estão sendo disponibilizadas no formato OPC UA, podem-se conectar vários clientes, e esses têm a possibilidade de fazer controle de acesso histórico, geração de relatórios, controle de alarmes de qualquer dispositivo com comunicação Modbus Ethernet.

Esse projeto ainda pode ser expandido para servidores OPC UA com comunicação com outros tipos de protocolo de chão de fábrica e os seus dados, podendo ser disponibilizadas em todos os níveis de produção.

REFERÊNCIAS

OPC FOUNDATION. **OPC UA SPECIFICATION PART 1 to 11**. Disponível em: <http://www.opcfoundation.org/>. Acesso em: Setembro 2010.

OPC FOUNDATION. **OPC UA SDK 1.01**. Disponível em: <http://www.opcfoundation.org/>. Acesso em: Setembro 2010.

MODBUS.ORG **MODBUS APPLICATION PROTOCOL SPECIFICATION**. Disponível em: <http://www.modbus.org/docs/Modbus Application Protocol V1 1b.pdf/>. Acesso em: Setembro 2010.

MODBUS.ORG **MODICON MODBUS PROTOCOL REFERENCE GUIDE**. Disponível em: <http://www.modbus.org/docs/PI MBUS 300.pdf>. Acesso em: Setembro 2010.

NMODBUS.COM **LIBRARY NMODBUS IN .NET C#**. Disponível em: <http://nmodbus.com/>. Acesso em: Setembro 2010.

MODBUS.ORG **MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE**. Disponível em: <http://www.modbus.org/docs/Modbus Messaging Implementation Guide V1 0b.pdf>. Acesso em: Setembro 2010.

ALTUS.COM.BR **INFORMAÇOES CLP FBS-24MATJ**. Disponível em: <http://altus.com.br>. Acesso em: Novembro 2010.