SAMIRA SOUIT

AMBIENTE DISTRIBUÍDO PARA MONITORAÇÃO E OPERAÇÃO REMOTA DE SISTEMAS PRODUTIVOS

São Paulo

2009

SAMIRA SOUIT

AMBIENTE DISTRIBUÍDO PARA MONITORAÇÃO E OPERAÇÃO REMOTA DE SISTEMAS PRODUTIVOS

Monografia apresentada à Escola Politécnica da Universidade de São Paulo referente à disciplina PMR 2550 – Projeto de Conclusão de Curso II

Curso de Graduação:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Paulo Eigi Miyagi

São Paulo
2009

FICHA CATALOGRÁFICA

Souit, Samira

Ambiente distribuído para monitoração e operação remota de sistemas produtivos/ Samira Souit. – São Paulo, 2009.

Monografia – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e Sistemas Mecânicos.

1. Monitoramento 2. Sistemas Distribuídos 3. Internet 4.Sistemas Colaborativos I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e Sistemas Mecânicos.

AGRADECIMENTOS

Ao Prof. Dr. Paulo Eigi Miyagi, por me orientar durante este trabalho de conclusão de curso e, também, por sua orientação em meu projeto de Iniciação Científica. Agradeço a sua dedicação, sempre mostrando novas visões sobre o ensino e a engenharia.

Ao colega José Isidro Garcia, pelo constante apoio e suporte ao desenvolvimento deste trabalho. Ao Prof. Dr. Fabrício Junqueira, por auxiliar-me nas diversas dificuldades encontradas. Ao colega Marcosiris Pessoa, pelo apoio técnico fornecido.

À CNPq, FAPESP e CAPES por financiarem este projeto.

RESUMO

Com os avanços da informática, comunicação e mecatrônica, os sistemas produtivos (SPs) têm evoluído uma estrutura distribuída e dispersa para assegurar a competitividade das empresas. Entretanto, existem ainda aspectos metodológicos que necessitam mais estudos, principalmente para a concepção de arquiteturas para estes sistemas produtivos. Assim, este trabalho contribui para este estudo através da implementação prática de programas computacionais para teleoperação e monitoração remota de SPs, dentro de uma arquitetura distribuída e colaborativa. O SP disperso considerado neste trabalho é emulado pelo sistema de manufatura flexível instalado na USP e o software de coordenação utiliza a tecnologia de *Web Services* (WS), para integrar as partes que compõe este SP segundo uma arquitetura distribuída baseada em serviços. Para a implementação destes programas foi utilizada a linguagem VB.NET e um banco de dados baseado num servidor SQL Server. Para a comunicação entre o controlador (CLP) de um dos subsistemas deste SP e o respectivo computador local (supervisório) utilizou-se os protocolos de comunicação OPC e Profibus. Para estruturação e especificação do projeto dos programas computacionais adotou-se a UML (*Unified Modeling Language*). Como resultado deste trabalho, através dos programas computacionais desenvolvidos, qualquer computador, via internet, pode acessar o servidor deste SP e assim pode teleoperar e monitorar remotamente os equipamentos do SP que estão num ambiente distribuído e disperso geograficamente.

Palavras Chaves: Monitoração Remota. Sistema Produtivo. Internet. Sistema Distribuído. Sistema Disperso. Sistema Colaborativo.

ABSTRACT

Information systems, communication and mechatronics have brought innovations to productive systems (PS) which evolved into a distributed and disperse structure to guarantee enterprise competitiveness. However, there are methodological aspects that require more studies, mainly for these productive systems architecture conception. Therefore, this work contributes for this study through the practical implementation of computational programs for teleoperation and remote monitoring of PSs in a distributed and collaborative architecture. The disperse PS considered on this work is emulated by the flexible manufacturing system installed in USP and the coordination software uses a web services (WS) technology to integrate the components of this PS according to a distributed architecture based on services. The VB.NET programming language and a database based on SQL Server were used for the implementation of these softwares. The OPC and Profibus communication protocols were adopted for communication between the controller (PLC) from one of the PS subsystems and the respective local computer (supervisory). For the specification and structuring of the computational programs project the Unified Modeling Language (UML) was employed. As result of this work, throughout the computational programs developed, any computer over the internet has the ability to access the PS server to remotely operate and monitor the PS equipments located in a geographically distributed and disperse environment.

Keywords: remote monitoring, productive system, internet, distributed system, disperse system, collaborative system.

LISTAS DE ILUSTRAÇÕES

Figura 1- Comportamento e interoperabilidade de um WS 16
Figura 2 - Troca de mensagens no protocolo SOAP 17
Figura 3 - Arquitetura do sistema SCTSP(Adaptado de Melo(2008)) 21
Figura 4 - Peças a serem montadas: corpo, pino, mola, tampa 22
Figura 5- Subsistema de Alimentação 23
Figura 6 - Arquitetura do ambiente distribuído proposto 27
Figura 7a – Interação entre os programas computacionais desenvolvidos 28
Figura 7 b - Configuração do controle local do subsistema de alimentação 29
Figura 8 - Casos de uso do Op.Alimentação 32
Figura 9 - Casos de uso do Cliente - SCTSP 33
Figura 10 - Casos de uso do Supervisório 33
Figura 11 - Arquitetura do software de coordenação 35
Figura 12 - Diagrama de classes do software de coordenação 36
Figura 13 - Diagrama de seqüência para o Registro de Teleoperador 38
Figura 14 - Diagrama de seqüencia para o acesso do Teleoperador 39
Figura 15 - Diagrama de seqüência para a escolha do modo de operação 40
Figura 16 - Diagrama de seqüência para o cadastro do cliente 41
Figura 17 - Diagrama de seqüência para a verificação de disponibilidade do subsistema de alimentação 42
Figura 18 - Diagrama de seqüência para a solicitação de pedido 43
Figura 19 - Diagrama de seqüência para execução do pedido: verificação do modo de operação 44
Figura 20 - Diagrama de seqüência da execução de pedido: Monitoração 45
Figura 21 - Diagrama de seqüência de execução de pedido: modo Teleoperação 46
Figura 22 - Diagrama de seqüência informação sobre o estado das atividades – Teleoperação 46
Figura 23 - Diagrama de seqüência para atualização de estado: Teleoperação 47
Figura 24 - Diagrama de seqüência para atualização de peças entregues 48
Figura 25 - Diagrama de seqüência para a consulta de peças entregues 48

Figura 26 - Diagrama de seqüência para o registro de informação sobre dispositivos do subsistema de alimentação 49
Figura 27 - Diagrama de seqüência para consulta de estado de dispositivos do subsistema de alimentação 50
Figura 28 - Diagrama de seqüência para parada de emergência 50
Figura 29 - Modelo entidade relação do BD utilizado 51
Figura 30 - CPU 412-2 PCI (Fonte: SIEMENS, 2002c) 53
Figura 31 - Módulo ET200M (SIEMENS, 2000d) 54
Figura 32 - Configuração do PG/PC Interface 55
Figura 33 - Configuração do Station Configurator Editor 56
Figura 34 - Configuração no STEP 7 57
Figura 35 - Configuração do OPCScout (Fonte: SIEMENS, 2008e) 58
Figura 36 - Seqüência de comandos para estabelecer a conexão *OPC Client* (Fonte: Siemens, 2008f) 60
Figura 37 - Fluxograma para o Método “AcordaSupervisorio” 61
Figura 38 - Página de acesso principal 62
Figura 39 - Página de *login* do Cliente 63
Figura 40 - Página de Operações do Cliente 64
Figura 41 - Página de *login* do Teleoperador 65
Figura 42 - Página de Operações do Teleoperador 66

LISTAS DE TABELAS

Tabela 1 - Dispositivos de atuação, detecção, comando e monitoração..... 24
Tabela 2 - Casos de uso do Op.Alimentação..... 30
Tabela 3 - Variáveis mapeadas para comunicação OPC..... 58

LISTAS DE ABREVIATURAS E SIGLAS

SP – Sistema Produtivo

WS – *Web Service*

SCTSP – Sistema Colaborativo de Teleoperação de Sistemas Produtivos

COS – Computação orientada a Serviços

CLP – Controlador Lógico Programável

CPU – *Central Processing Unit*

IO – *Input / Output*

MVC – *Model – View – Controller*

UML – *Unified Modeling Language*

SOAP – *Service Oriented Architecture Protocol*

XML – *eXtensible Markup Language*

DLL – *Dynamic Link Library*

LSA - Laboratório de Sistemas de Automação

SUMÁRIO

1. Introdução 13
2. Metodologia de Desenvolvimento e Ferramentas Consideradas 14
3. Conceitos Fundamentais 16
   3.1 Web Service (WS) 16
   3.2 Comunicação entre aplicativos 18
4. Definição do ambiente distribuído no SCTSP 20
   4.1 Descrição do Subsistema de alimentação 22
      4.1.1 Processo produtivo 23
   4.2 Arquitetura do ambiente distribuído do subsistema de Alimentação 27
      4.2.1 Requisitos do Subsistema de Alimentação 29
         Caso de Uso do Subsistema de Alimentação 30
      4.2.2 Software de Coordenação 34
         a) Diagrama de Classes 35
         b) Diagrama de Seqüência 37
            Cadastro de Teleoperador 37
            Acesso do Teleoperador 38
            Modo de Operação do Teleoperador 39
            Cadastro do cliente 40
            Verificação de Disponibilidade do Subsistema de Alimentação 41
            Solicitação de Pedido 42
            Execução de Pedido: Verificação do Modo de Operação 44
            Execução do Pedido: Modo Monitoração 44
            Execução do Pedido: Modo Teleoperação 45
            Informação sobre o estado – Teleoperação 46
            Atualização de estado - Teleoperação 46
            Registro de peças entregues 47
            Consulta de peças entregues 48
            Registro dos estados dos equipamentos 48
            Consulta dos estados dos equipamentos 49
            Parada de Emergência 50
      4.2.3 Banco de dados 51

4.3 Instalações Físicas 52
4.3.1 Programação do CLP 54
4.4 Software Supervisório 58
4.5 Interfaces dos Usuários 62
Acesso Principal 62
Página de *Login* do Cliente 63
Página de Operações do Cliente 63
Página de *Login* do Teleoperador 64
Página de Operações do Teleoperador 65
5. Conclusões 67
6. Referências Bibliográficas 69
ANEXO A – INSTALAÇÕES ELÉTRICAS 73
ANEXO B – PROGRAMA EM LADDER DO CONTROLADOR 74
ANEXO C – CÓDIGO DO SOFTWARE DE COORDENAÇÃO 77
ANEXO D – CÓDIGO DO SUPERVISÓRIO 94
ANEXO E – PÁGINAS WEB DE ACESSO 102

1. INTRODUÇÃO

Com a globalização e avanços nas tecnologias de informação, têm surgido novas estruturas organizacionais dos sistemas produtivos, onde não apenas os componentes são produzidos em diferentes plantas, mas em diferentes países, sendo as sub-montagens e montagens dos produtos finais também realizadas em diferentes locais de modo a assegurar o melhor preço, qualidade e disponibilidade para os clientes. Para enfrentar os desafios impostos por estas configurações dispersas, em um ambiente altamente competitivo, e reagir de modo ótimo e flexível às mudanças de condições, é necessário que gerentes operacionais e executivos tenham uma ampla visibilidade do estado de seu SP, em todas as camadas, desde o chão de fábrica até o nível de negócios (SOCRADES, 2009). Neste contexto, os sistemas colaborativos teleoperados para sistemas produtivos (SCTSP) têm surgido como uma alternativa interessante no alcance destes objetivos (MELO & JUNQUEIRA & MIYAGI, 2008). Este tipo de sistema é, fundamentalmente, o resultado da integração de componentes autônomos com funcionalidades especializadas, e que devem ter suas atividades coordenadas para executar um processo produtivo. Tal estrutura distribuída é comumente adotada, visando um aumento na qualidade com redução dos custos, o que é necessário para garantir a competitividade destes sistemas produtivos no mercado global. Neste sentido, e aproveitando os desenvolvimentos da internet em relação à capacidade de transmissão de dados, som, vídeo e criação de dispositivos virtuais além do gerenciamento de repositórios de informação (GORODIA & ELHAJJ, 2005), considera-se que é possível conceber sistemas colaborativos transnacionais interligados via algum tipo de rede de comunicação (JUNQUEIRA & MIYAGI, 2006) (HABIB, 2000). Por outro lado, considerando que a automatização de um sistema produtivo deve ser concebida de maneira balanceada de modo tão rígido quanto seja possível e tão flexível quanto seja necessário (ADLEMO & ANDREASSON, 2007), é imprescindível que os diferentes subsistemas que compõem um sistema colaborativo disponham de um ambiente de teleoperação e monitoração remota das diferentes atividades envolvidas no sistema produtivo, de maneira que se permita uma efetiva

interação de operadores humanos independentemente de sua localização física em situações tais como: gestão das atividades do serviço fornecido, arbitragem de conflitos, operação de interfaces, tratamento de falhas e situações anormais, entre outras.

Neste projeto o objetivo é desenvolver um ambiente distribuído para teleoperação e monitoração remota de sistemas produtivos, permitindo a interação via internet entre vários operadores e recursos deste sistema. Entende-se aqui por "ambiente" a composição e a configuração destes recursos como o banco de dados, o software Supervisório de controle local do SP e o software de coordenação; e a interação destes com os atores do SP, que são os "clientes" e os "teleoperadores", todos se relacionando de modo disperso e distribuído, por meio da internet.

No presente trabalho, um sistema flexível de montagem de peças será utilizado para emular um sistema produtivo disperso. Primeiramente, será implementado o controle de uma das partes deste sistema, isto é, um subsistema com um certo grau de autonomia, utilizando-se uma estrutura que envolve um computador local e controlador programável e, em seguida, será criada uma interface computacional tanto para a monitoração local como para monitoração e teleoperação remotas. Por fim, serão incorporadas subrotinas que integrarão este subsistema em um sistema de arquitetura distribuída, baseada em serviços, mais especificamente *Web Services* (WS) (MICROSOFT, 2009a), visando-se construir um sistema colaborativo de teleoperação produtivo (SCTSP).

2. METODOLOGIA DE DESENVOLVIMENTO E FERRAMENTAS CONSIDERADAS

Adotou-se neste trabalho uma metodologia de desenvolvimento de pesquisas muito adotada na área de rede de Petri proposta em Jensen (1992), que envolve três aspectos diferentes: teorias (por exemplo: técnicas de modelagem, controle colaborativo, programação orientada a objetos, computação distribuída), ferramentas (por exemplo: programas computacionais

para edição de modelos e simulação de sistemas) e aplicações (por exemplo: sistema produtivo disperso e sistema colaborativo de teleoperação), que foram abordados de forma cíclica e repetitiva.

Para o desenvolvimento dos programas computacionais, foi adotada a arquitetura *Model-View-Controller* ou MVC (Figura 1). Esta arquitetura MVC é a estruturação do software com camadas "bem" definidas de manipulação de dados, e permite que os dados sejam alterados em sua camada específica, sem a necessidade de alteração das demais, o que é adequado no desenvolvimento de softwares (ALMEIDA, 2009). Para a modelagem do software foram utilizados diagramas UML (*Unified Modeling Language*), tais como o diagrama de casos de uso (diagrama comportamental), diagrama de classes (diagrama estrutural) e diagrama de seqüência (diagrama de interação). Esta modelagem permite uma melhor visualização do software, auxiliando em seu projeto e em sua programação.

Adotou-se para desenvolver este projeto a linguagem de programação Visual Basic .NET, que é uma linguagem de programação orientada a objetos, que permite criar os WSs (MICROSOFT, 2009a) e implementar a arquitetura proposta. Aliada a essa linguagem de programação, foi considerado um banco de dados utilizando-se SQL Server (MICROSOFT, 2009c). Como se tratam de aplicações de um mesmo fabricante, elas apresentam uma fácil interconectividade (MICROSOFT, 2009a), além de serem utilizadas por várias indústrias, que são o foco principal deste projeto. Para o controle local, considerou-se um controlador lógico programável (CLP) da empresa SIEMENS e o software STEP 7 (SIEMENS, 2007a) para sua programação. Para os dados trocados entre o software supervisório local e o programa do CLP utilizou-se OPC XML-DA (SIEMENS,2004b). Para a comunicação entre a CPU do CLP e a placa de IO foi considerado o PROFIBUS (SIEMENS, 2007a). De fato, foi realizado um estudo preliminar sobre as várias opções de técnicas e ferramentas disponíveis e todas as escolhas foram baseadas na disponibilidade destes e informações para seu uso.

3. CONCEITOS FUNDAMENTAIS

A seguir, serão apresentados os conceitos fundamentais considerados neste projeto.

3.1 Web Service (WS)

O WS foi desenvolvido como um tipo de padrão a ser utilizado para implementar a prestação de um serviços via internet, a qual nos últimos anos tem favorecido o incremento das transações entre empresas (BUSINESS WIRE, 2006). Este padrão foi desenvolvido para aplicativos que utilizam comunicação distribuída, isto é, na comunicação remota com interação entre usuários simultaneamente em um mesmo aplicativo. Assim, um usuário pode acessar os serviços, ou seja, as funcionalidades que foram desenvolvidas e que são disponibilizadas por este aplicativo e que ficam assim disponíveis em outro computador, via Internet. Segundo Kreger (2001), esse padrão permite a criação de um aplicativo modular (ou seja, sua estrutura permite que ele componha/seja composto por outros aplicativos), auto-contido e auto-definido, que pode ser publicado, localizado e invocado via internet.

Baseado no WS tem-se um novo paradigma da computação chamado "computação orientada a serviços" (COS) (*Service-Oriented Computing*) o qual utiliza os serviços, disponibilizados por WS, como componentes fundamentais no desenvolvimento de aplicações (PAPAZOGLOU & GEORGAKOPOULOS, 2003).

Figura 1- Comportamento e interoperabilidade de um WS

A Figura 2 ilustra a interoperabilidade do WS criada pelo protocolo de comunicação que utiliza. Este protocolo, o SOAP (*Service Oriented Architecture Protocol* e *Service Object Acess Protocol*) é um protocolo que define uma gramática XML (*eXtensible Markup Language*) especializada, porém flexível, que padroniza o formato das estruturas das mensagens (RECKZIEGEL, 2006). As mensagens são o método fundamental de troca de informações entre os WSs e os seus usuários, como ilustra a Figura 2. Ao utilizar XML para codificar mensagens, o SOAP provê alguns benefícios como, por exemplo, a facilidade de leitura do XML, permitindo uma melhor analise e busca por erros; o XML *parsers* (analistas) e tecnologias correlatas são mundialmente disponíveis; o XML é um padrão aberto; e admite simplificação da especificação, diferente de outros protocolos binários como COM, DCOM e CORBA (RECKZIEGEL, 2006).

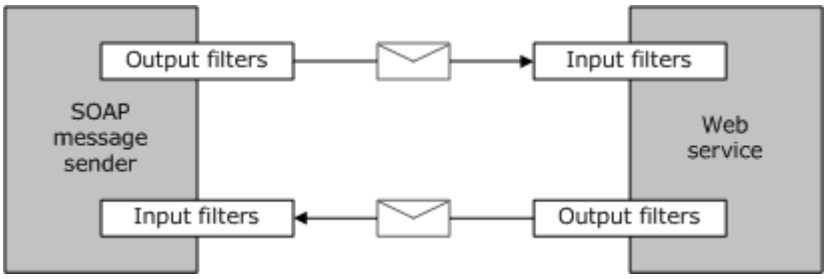


Figura 2 - Troca de mensagens no protocolo SOAP

Segundo Melo (2008), com a adoção deste paradigma, um SP pode ter melhores condições de se adaptar as mudanças dos mercados, considerando o ambiente heterogêneo próprio deste tipo de sistema. Assim, entre as principais características que promovem a adoção da COS no SCTSP encontra-se: a) flexibilidade na implementação de ambientes heterogêneos; b) flexibilidade do sistema frente a futuras alterações de especificações; c) reuso dos componentes fornecedores de serviços; d) melhores condições para assegurar a integração dos componentes modulares. Existem assim propostas com novos tipos de infra-estrutura que permitem a integração de empresas por meio de uma rede óptica de altíssima velocidade, constituindo um ambiente próprio de trabalho colaborativo à distância (KYATERA, 2003).

Como o WS é um fundamento para a implementação da computação orientada a serviços, vários trabalhos têm sido desenvolvidos com o intuito de

definir procedimentos de coordenação e integração dos WS aplicados à um ambiente fabril (SOCRADES, 2009).

3.2 Comunicação entre aplicativos

Neste projeto, o supervisório que é um aplicativo que gerencia as atividades dos controladores lógicos programáveis dos subsistemas do SP. É assim necessário estabelecer como é a comunicação deste equipamento com o aplicativo desenvolvido. Outro aspecto importante é sobre a comunicação entre os programas computacionais deste ambiente distribuído (supervisório, software de coordenação, programa do CLP) e que serão abordados posteriormente. Para isto, a seguir tem-se um breve histórico dos principais métodos de comunicação existentes.

Histórico

O padrão DDE (*Dynamic Data Exchange*) foi criado em 1987 com o lançamento da plataforma do Windows 2.0. Este padrão regulamentava a troca de dados entre diferentes aplicações dentro desta plataforma Windows. Na maioria das vezes uma aplicação (servidor) que possui o dado, fornece-o a outra aplicação (cliente)(TALTECH, 2009). Devido ao Windows ter uma arquitetura baseada em mensagens, a transferência de mensagens é o jeito mais prático de se trocar informações. Um dos usos mais comuns deste protocolo é para se transferir dados em tempo real, como por exemplo, dados de instrumentos científicos ou de controle de processos (MICROSOFT, 2009b).

A tecnologia OLE (*Object Linking and Embedding*) foi desenvolvida pela Microsoft em meados de 1990, para suprir a necessidade de se integrar diferentes aplicações dentro da plataforma Windows, de forma a solucionar os problemas de desempenho e confiabilidade do até então utilizado padrão DDE (FONSECA, 2009).

Como uma continuação da tecnologia OLE, o DCOM (*Distribuited Component Object Model*) surgiu junto com o sistema operacional Windows NT e foi logo aceito pela indústria. Basicamente, o DCOM é um conjunto de

definições para permitir a implementação de aplicações distribuídas (aplicações que estão em computadores remotos e que interagem entre si) em uma arquitetura cliente-servidor. Uma arquitetura cliente-servidor é uma arquitetura em que um aplicativo possui os dados (servidor) e outro aplicativo requisita esses dados (cliente). Desta forma, um cliente pode acessar diversos servidores ao mesmo tempo e um servidor pode disponibilizar suas funcionalidades para diferentes clientes ao mesmo tempo. Através da definição de interfaces, o DCOM permite que objetos sejam instanciados de forma distribuída e seus serviços e métodos (funções) sejam acessíveis por diferentes programas. Para isso é necessária a utilização de uma linguagem especial, a IDL *(Interface Definition Language*). Isto significa que cada cliente pode chamar os métodos de cada objeto DCOM em um determinado servidor, independentemente do ambiente de programação (linguagem, compilador, versão, etc) que os mesmos foram criados. Através de um identificador único, GUID (*Global Unique Identifier*), as interfaces são protegidas contra modificações após a sua publicação e a compatibilidade dos objetos DCOM é então garantida. (FONSECA, 2008)

Em seguida, surgiu a tecnologia ActiveX, em que os usuários ainda precisavam considerar funções de controle para implementar aproximadamente seis núcleos de interfaces. Como resposta a este problema, a Microsoft produziu macros, bibliotecas, extensões em C++, wizards, entre outras ferramentas a fim de se facilitar a implementação desses controles. Assim, *ActiveX controls*, que são pequenos programas criados em bloco, servem para criar aplicações distribuídas que funcionam na internet através de *web browsers* (MICROSOFT, 2009d).

Atualmente, essa tecnologia ActiveX está sendo substituída pela plataforma *.NET* da Microsoft, que visa uma plataforma única para desenvolvimento e execução de sistemas e aplicações. Com idéia semelhante na plataforma Java, o programador deixa de escrever código para um sistema ou dispositivo específico e passa a escrever programas computacionais para a plataforma .NET. Esta plataforma permite a execução, construção e desenvolvimento de WSs de forma integrada e unificada (MICROSOFT, 2009e). Por sua vez, WS configura-se como a mais nova solução de integração de sistemas e na comunicação de diferentes aplicações. Com essa tecnologia

é possível que novas aplicações interajam com as existentes e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Cada aplicação pode possuir sua própria linguagem, que é traduzida para um formato universal, o XML. Para as empresas, os WSs podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística (SOCRADES, 2009). Este projeto considerou assim este método de integração de aplicações para construir a arquitetura distribuída de controle e a monitoração remota dos subsistemas.

4. DEFINIÇÃO DO AMBIENTE DISTRIBUÍDO NO SCTSP

Este capítulo apresenta uma definição para o ambiente distribuído que está inserido no contexto de um SCTSP, considerado como base no estudo de caso utilizado neste trabalho. Para tal finalidade foi adotado o sistema automatizado de montagem industrial, instalado na Escola Politécnica da USP (LIRA, 2008).

A arquitetura do SCTSP (Figura 3) visa automatizar e coordenar as diferentes atividades dos vários subsistemas que compõem o SP, isto é, o subsistema de montagem, o subsistema de transporte, o subsistema de inspeção, o subsistema de alimentação de peças e o subsistema de vigilância, que são interligados através de uma rede de comunicações, fornecendo serviços específicos. Resumidamente, o subsistema de inspeção executa o serviço de controle de qualidade e identificação das características físicas do produto fornecido pelo subsistema de alimentação (peça “corpo”), de maneira que se garantam as especificações do produto requisitado pelo subsistema de montagem. O subsistema de transporte executa o serviço de transporte e movimentação dos pallets onde as peças são coletadas ou retiradas dos outros subsistemas. O subsistema de montagem, por sua vez, executa o serviço de montagem dos produtos finais ou “peças montadas” que podem ser de três tipos, dependendo das seguintes variáveis: “corpo” prata com “êmbolo” preto, “mola” e “tampa”; “corpo” rosa com “êmbolo” preto, “mola” e “tampa”; e “corpo”

preto com “êmbolo” prata, “mola” e “tampa”. A quantidade de produtos finais é requisitada por um usuário via internet. O subsistema de vigilância executa serviços de visualização dos equipamentos.

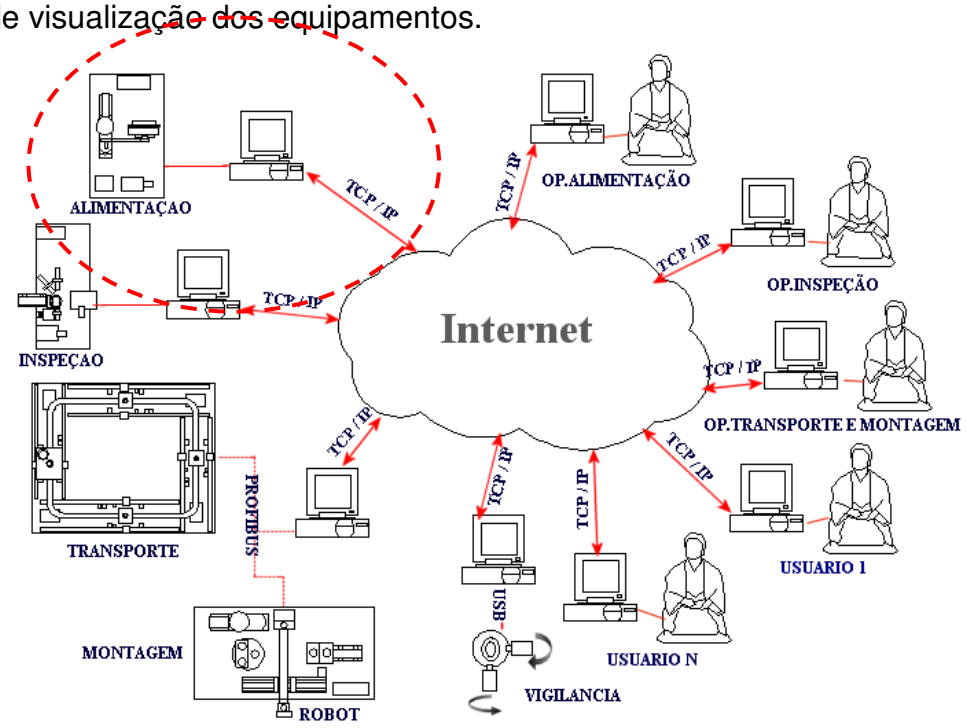


Figura 3 - Arquitetura do sistema SCTSP(Adaptado de Melo(2008))

As peças que compõem o produto são ilustradas na Figura 4. A idéia é que os diferentes subsistemas possam ser teleoperados e monitorados remotamente via internet. Desta maneira, os operadores de cada um dos subsistemas da arquitetura do SCTP (subsistema de alimentação, de montagem, de transporte, de vigilância e de inspeção), são chamados de op.alimentação, op.motagem, op.Transporte, op.Vigilância op.Inspeção respectivamente e, podem acessar o sistema em dois possíveis modos de operação: de monitoração remota e teleoperação. No modo de monitoração remota, uma série de informações é fornecida ao teleoperador, com o intuito de garantir o acompanhamento da execução remota do processo produtivo através do acompanhamento do serviço oferecido. No modo de teleoperação, uma série de comandos é disponibilizada ao operador para a intervenção na execução dos processos produtivos. Assim, através da troca de informações entre os diferentes serviços que integram o SCTP é estabelecido um trabalho colaborativo de vários operadores e subsistemas com o intuito de produzir um produto final.

Figura 4 - Peças a serem montadas: corpo, pino, mola, tampa

O foco deste do projeto é o desenvolvimento das interfaces para a monitoração e teleoperação do subsistema de alimentação (em destaque na Figura 3). A estratégia de desenvolvimento do sistema aqui adotada é focada na implementação de todos os serviços específicos de um único subsistema mas de modo que estes possam ser devidamente e facilmente replicados e adaptados a outros subsistemas do SP.

4.1 Descrição do Subsistema de alimentação

Para realizar a monitoração e teleoperação do subsistema de alimentação (Figura 5), é necessário conhecer quais elementos compõem este subsistema, identificando quais os tipos de dados tratados para a execução dos serviços, como estes estão fisicamente armazenados na memória dos controladores locais (no caso, no CLP), bem como as funções disponíveis. Este levantamento de dados foi feito, e organizado em tabelas, para facilitar a comunicação entre o subsistema e os programas computacionais das interfaces desenvolvidas.

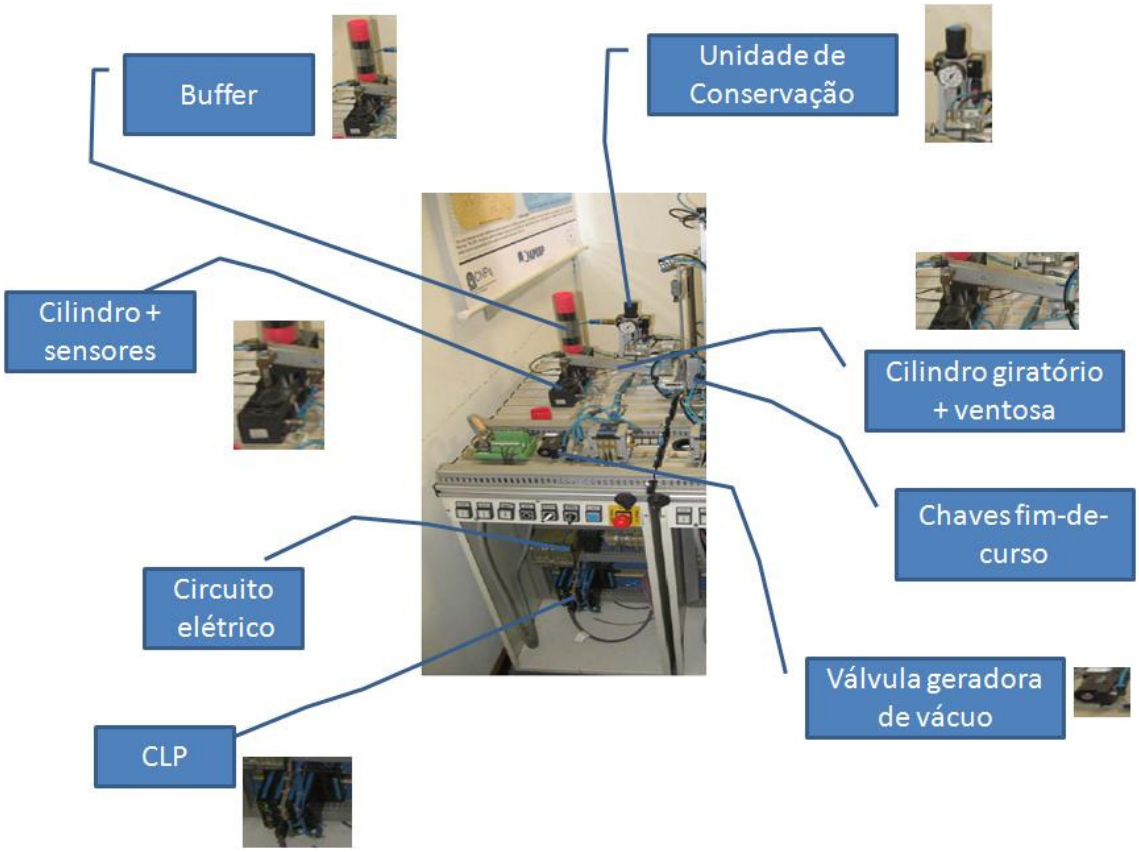


Figura 5- Subsistema de Alimentação

4.1.1 Processo produtivo

O subsistema de alimentação possui um *buffer* com capacidade de até 10 peças "corpo", que podem ser nas cores preta, prata ou rosa e ser de plástico ou metálica. Para extrair tais peças do buffer (Figura 5) é utilizado um pistão linear de simples ação, 1A, que posiciona a peça "corpo" para ser transferida por um braço basculante, 3A, ao subsistema de inspeção. O cilindro é atuado por uma válvula de comando elétrico, 1Y1. Quando o pistão está recuado, a peça se mantém no buffer, e quando o pistão está estendido, a peça é retirada do buffer. As posições finais do pistão são detectadas utilizando sensores magnéticos de proximidade, 1B1 e 1B2. Além disso, o *buffer* conta com um sensor óptico de presença que está situado na sua base, 1B4. O transporte da peça "corpo" a outro sistema produtivo é feito pela ação de um braço basculante acionado por um cilindro pneumático rotatório ou giratório de acordo com o nome do fabricante, 3A, e que pode ser ajustado para vários

ângulos de deslocamento entre 0° e 180°. As posições finais são determinadas por micro-sensores fim-de-curso, 3S1 e 3S2. O cilindro é atuado por válvulas de comandos elétrico, 3Y1 e 3Y2. Na extremidade livre do braço basculante tem-se uma garra com uma ventosa de aspiração para segurar e soltar uma peça “corpo”. A aspiração é feita por um gerador de vácuo, 2A. O controle da aspiração é detectado por um vacuostato eletrônico ajustável, 2B1, que possui um pressostato que regula o ponto de comutação. Para o ajuste e configuração inicial do dos atuadores do subsistema de alimentação é disponibilizado um conjunto de dispositivos de comandos e de monitoração.

Na Tabela 1, tem-se uma lista dos dispositivos de atuação, detecção, comando e monitoração deste subsistema:

Tabela 1 - Dispositivos de atuação, detecção, comando e monitoração

Mapeamento de IOs - Subsistema de alimentação					
Código	**Função**	**Tipo de Dispositivo**	**Tipo de Dado**	**Endereço CLP**	
1A	Retira a peça “corpo” do depósito	Cilindro pneumático de simples ação	--		Dispositivo de atuação
2A	Gerador de vácuo e ventosa	Gerador de vácuo e ventosa	--	--	Dispositivo de atuação
3A	Transporta a peça “corpo” ao subsistema seguinte	Cilindro pneumático giratório de dupla ação (braço basculante)	--	--	Dispositivo de atuação
1Y1	Altera o estado de válvula de atuação do cilindro 1A	Solenóide	Booleano	A0.0	Dispositivo de atuação
3Y1	Altera o estado de válvula de atuação do cilindro 3A na posição de alimentação	Solenóide	Booleano	A0.1	Dispositivo de atuação
3Y2	Altera o estado de válvula de atuação do cilindro 3A na posição de inspeção	Solenóide	Booleano	A0.2	Dispositivo de atuação

2Y1	Altera estado de válvula+ atuação para posição de desligar o gerador de vácuo	Solenóide	Booleano	A0.3	Dispositivo de atuação
2Y2	Altera estado de válvula atuação para posição de ligar gerador de vácuo	Solenóide	Booleano	A0.4	Dispositivo de atuação
H1	Lâmpada botão de start	Lâmpada	Booleano	A1.0	Dispositivo de Monitoração
H2	Lâmpada botão de reset	Lâmpada	Booleano	A1.1	Dispositivo de Monitoração
H3	Lâmpada botão magazine *empty*	Lâmpada	Booleano	A1.2	Dispositivo de Monitoração
H4	Indica o estado do canal de comunicação	Lâmpada	Booleano	A1.3	Dispositivo de Monitoração
1B2	Detecta se o pistão está recuado (carrega peça “corpo” do depósito)	Sensor magnético de proximidade	Booleano	E0.0	Dispositivo de Detecção
1B1	Detecta se pistão está estendido (retirou e retém peça “corpo”)	Sensor magnético de proximidade	Booleano	E0.1	Dispositivo de Detecção
3S1	Detecta se braço basculante está posicionado no sistema de alimentação	Sensor de contato mecânico	Booleano	E0.2	Dispositivo de Detecção
3S2	Detecta se o braço basculante está posicionado no sistema de	Sensor de contato mecânico	Booleano	E0.3	Dispositivo de Detecção

	inspeção				
2B1	Detecta se a garra ventosa pegou peça “corpo”	Vacuostato	Booleano	E0.5	Dispositivo de Detecção
1B4	Detecta a presença de peças “corpo” no sistema de alimentação (High - não tem peça/ Low - tem peça)	Sensor óptico	Booleano	E0.6	Dispositivo de Detecção
S1	Comando de início do processo de alimentação (Botão start on)	Chave de contato	Booleano	E1.0	Dispositivo de Comando
S2	Comando de reset do sistema de alimentação	Chave de contato	Booleano	E1.1	Dispositivo de Comando
S3	Comando de magazine empty (botão magazine empty)	Chave de contato	Booleano	E1.2	Dispositivo de Comando
S4	Comando de controle automático/manual do sistema	Chave de contato	Booleano	E1.3	Dispositivo de Comando
S5	Comando de parada do sistema (Botão STOP)	Chave de contato	Booleano	E1.4	Dispositivo de Comando
NOTAUS	Comando de parada de emergência/ Botão de QUIT	Chave de contato	Booleano	E1.5	Dispositivo de Comando
S6	comando de comunicação remota	Chave de contato	Booleano	E1.6	Dispositivo de Comando

4.2 Arquitetura do ambiente distribuído do subsistema de Alimentação

Neste projeto foram desenvolvidos os programas computacionais para a interação entre os usuários e operadores locais e remotos do subsistema de alimentação de maneira flexível e distribuída. Neste ambiente um usuário tipo cliente pode fazer uma ordem de pedido de maneira interativa e que esta ordem de pedido é executada automaticamente por um teleoperador disponível (aqui chamado de op.Alimentacao), sem a necessidade de sua presença física nas instalações do subsistema de alimentação. Para assegurar a execução dos serviços devidos, é necessário implementar acordos de conexão não pré-fixados (ou seja, sem a prévia definição de "quando"/ "onde" ou "como" será o acesso) e de escolhas flexíveis (isto é, sem a imposição ou pré-definição) de colaboradores (LEAL&BAX, 2001). Neste contexto, como dito anteriormente, WSs se configuram como a solução adequada a este problema, já que permitam uma interação transparente entre serviços, pois colaboradores distintos, mesmo usando tecnologias e plataformas diferentes, podem se conectar de maneira padrão (MICROSOFT, 2009a). As rotinas computacionais responsáveis por essa integração de diversos agentes será denominada neste trabalho por software de coordenação.

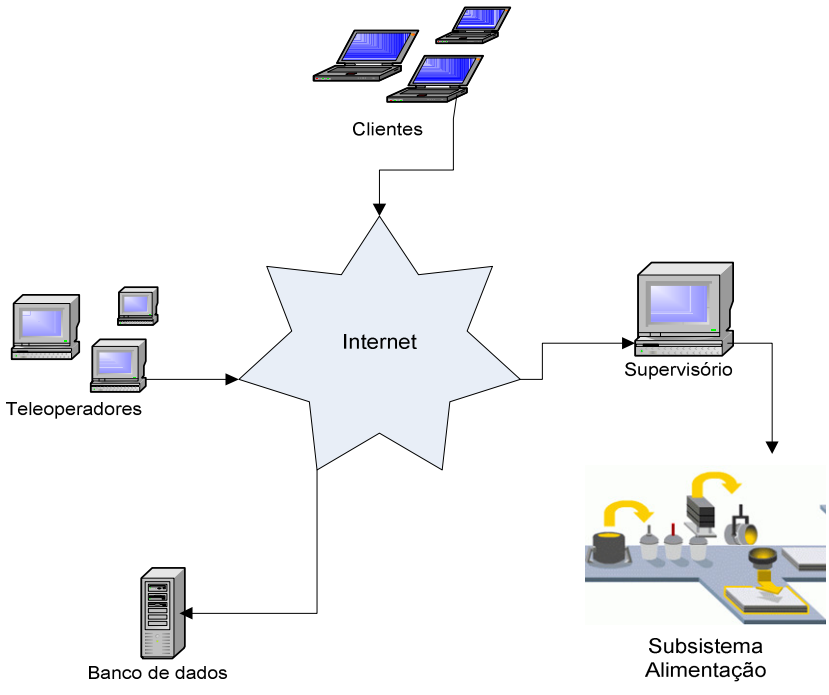


Figura 6 - Arquitetura do ambiente distribuído proposto

A Figura 6 ilustra o ambiente distribuído considerado neste projeto. Este ambiente é gerenciado por um SCTSP. O SCTSP é entendido assim como um tipo de usuário do subsistema de alimentação. Como neste trabalho o escopo não é mostrar como o SCTSP é implementado, será utilizada a denominação "cliente" para a atuação deste tipo de usuário. O "cliente" é assim o usuário que solicita uma ordem de pedido de peça ao software de coordenação.

Por sua vez, entende-se por "supervisório" um tipo de usuário responsável pela interação entre os usuários "cliente e teleoperadores com o subsistema de alimentação. O software supervisório comunica-se com o software de coordenação e com o CLP do subsistema de alimentação, de acordo com a Figura 7. O CLP utilizado neste projeto possui uma CPU (unidade de processamento) instalada numa placa que está no barramento principal de um computador local. A placa de IO (sinais de entradas e saídas) do CLP está num gabinete próximo ao equipamento/dispositivos do subsistema de alimentação e a comunicação entre a CPU e a placa de IO é feita via PROFIBUS (SIEMENS, 2007a). A comunicação entre o programa de controle do subsistema que está na CPU do CLP e o software supervisório é feita via OPC.

Figura 7a – Interação entre os programas computacionais desenvolvidos

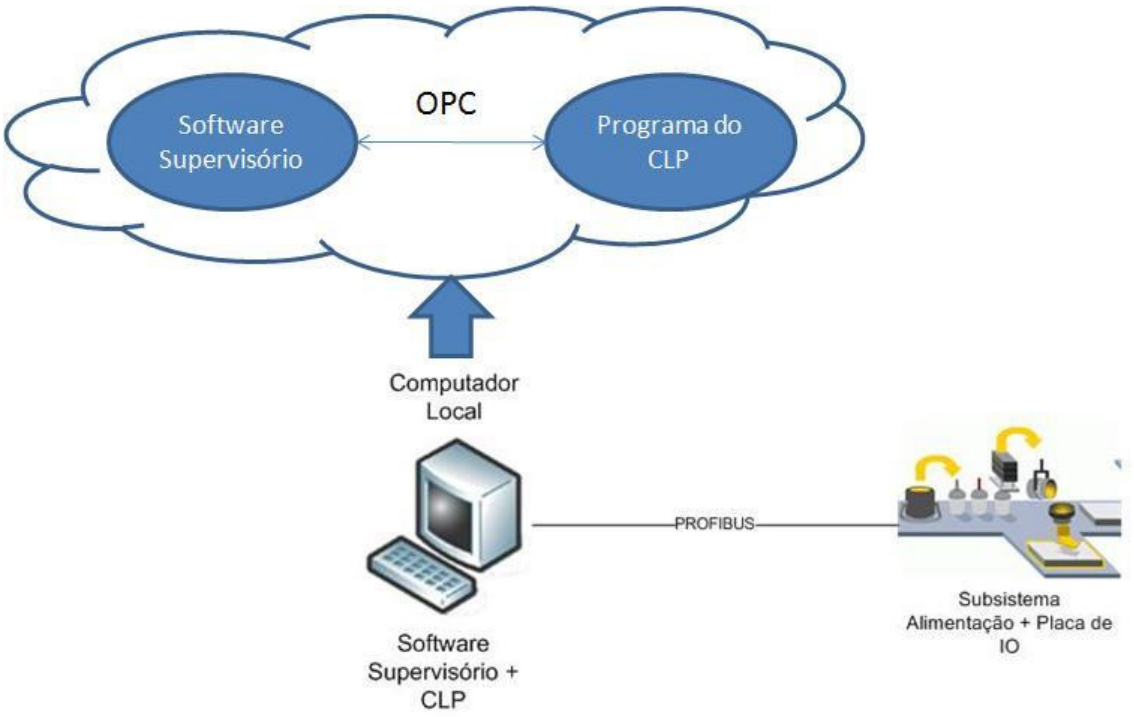


Figura 7 b - Configuração do controle local do subsistema de alimentação

O usuário tipo teleoperador indicado na Figura 6, é o op.Alimentação. Este usuário representa os operadores do subsistema, que podem ou não estarem fisicamente próximos ao equipamento/dispositivos do subsistema de alimentação. O Op.Alimentação tem dois modos de atuação sobre o subsistema: monitoração e teleoperação. A seguir, são apresentados os "requisitos de atuação" para este usuário, bem como os "casos de usos" considerados.

4.2.1 Requisitos do Subsistema de Alimentação

Ao conceber um sistema, deve-se ter um consenso sobre o que o sistema deve fazer, porém é importante ressaltar que a compreensão sobre os requisitos necessários, em geral, evolui à medida que se implementa e se interage com o determinado sistema. É essencial para o gerenciamento do projeto expressar tais requisitos como casos de uso e diagramas de casos de uso da UML (BOOCH; RUMBAUGH; JACOBSON, 2000).

Caso de Uso do Subsistema de Alimentação

Para o subsistema de alimentação, foram definidos especificamente, os casos de uso a seguir, que são fundamentais para a elaboração das interfaces (programa computacional para que os teleoperadores interajam com o subsistema de alimentação) e desenvolvimento da arquitetura do ambiente distribuído.

Tabela 2 - Casos de uso do Op.Alimentação

Atores:	op.alimentação: Quem aceita ou recusa o atendimento dos pedidos de peça de inspeção. Adicionalmente, executa as tarefas de monitoração ou teleoperação do sistema de alimentação.
Descrição:	Processa as tarefas de alimentação, monitoração e teleoperação do sistema de alimentação de peças solicitadas pelo cliente (SCTSP).
Pré-condições:	O subsistema de alimentação tem que validar as condições iniciais, ou seja, garantir que está apto a atender o(s) pedido(s) do cliente (SCTSP).
Pós-condições:	O subsistema de alimentação informa ao cliente (SCTSP).o atendimento completo do pedido. Por outro lado, caso o pedido proveniente do subsistema de inspeção não possa ser atendido, é enviado a este um relatório com o motivo do não atendimento.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Subsistema:**
1-Ao op.alimentação é apresentada uma mensagem de solicitação proveniente do cliente (SCTSP).	
	1.1- O subsistema de alimentação confirma ao cliente (SCTSP) a aceitação do pedido.

2- Ao op.alimentação é apresentada uma opção do modo de operação: monitoração ou teleoperação.	
	2.1.a- Se a opção de monitoração é selecionada, o subsistema apresenta ao op.alimentação uma interface própria para a monitoração das atividades de produção do subsistema de alimentação.
	2.1.b- Se a opção de teleoperação é selecionada, o subsistema apresenta ao op.alimentação uma interface própria para a teleoperação das atividades de produção do sistema de alimentação.
2.2.b. O op.alimentação executa os comandos para as atividades de alimentação.	
Fluxo Alternativo de Eventos	
Ações do Ator:	**Ações do Subsistema:**
	3- Após o passo 1.1. o subsistema de alimentação notifica ao cliente (SCTSP) que sua demanda não poderá ser atendida.

Depois de estabelecidos os casos de usos para o op.Alimentação (Tabela 2), são definidos os requisitos necessários para as interfaces deste, bem como do supervisório e do cliente (SCTSP) que são modelados a seguir, utilizando os diagramas de casos de uso (Figura 8) . Também, em seguida são modelados os diagramas de casos de uso para o usuário tipo “supervisório” (Figura 10) e tipo “cliente” (Figura 9).

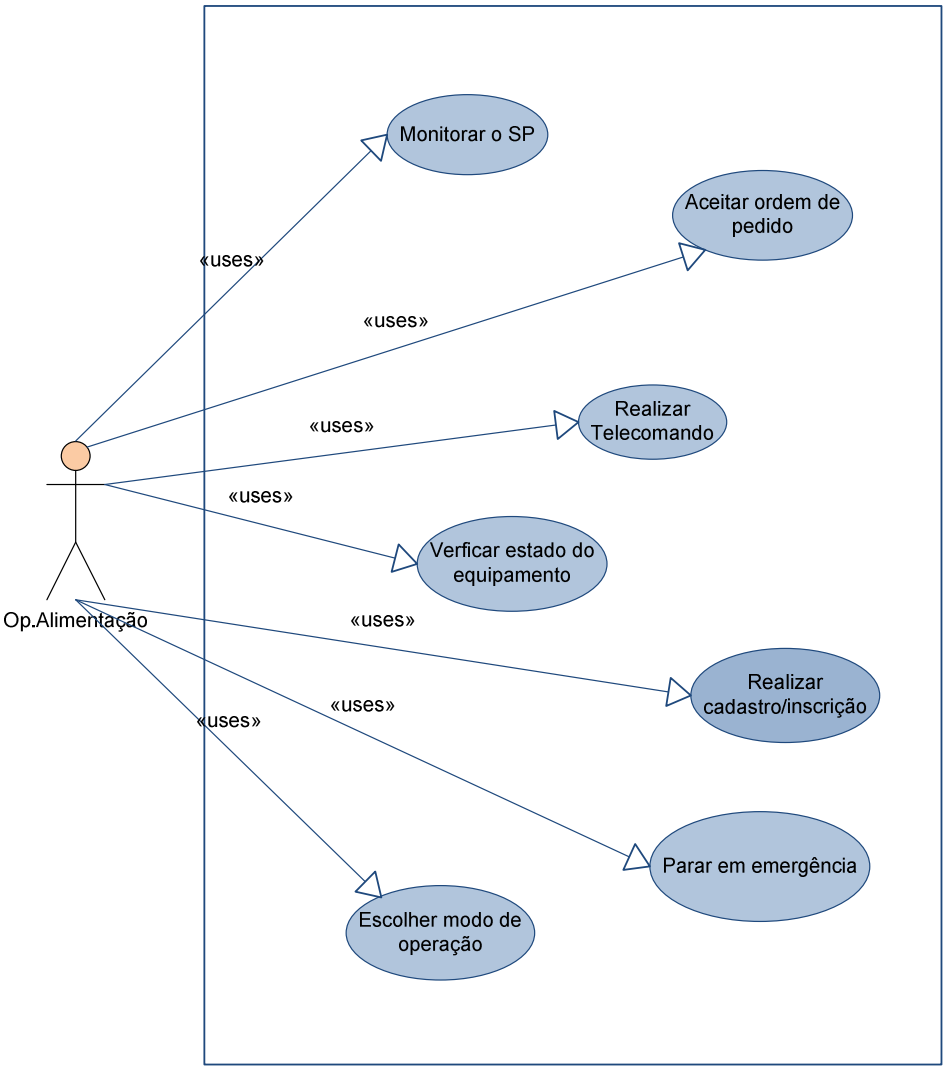


Figura 8 - Casos de uso do Op.Alimentação

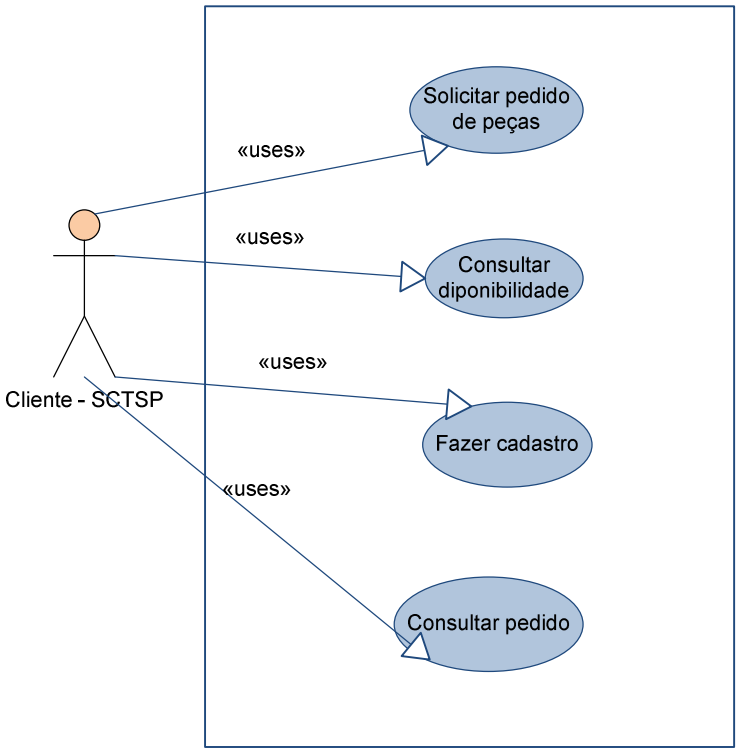


Figura 9 - Casos de uso do Cliente - SCTSP

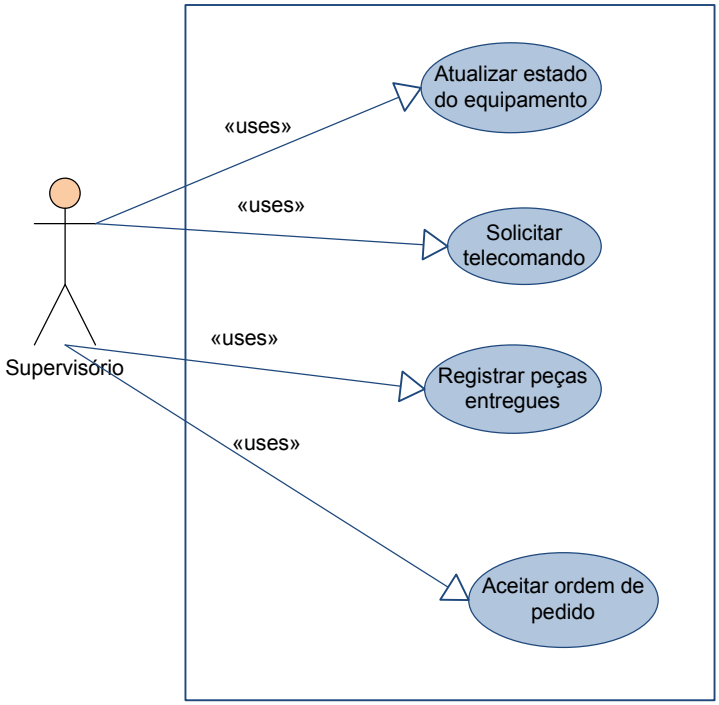


Figura 10 - Casos de uso do Supervisório

4.2.2 Software de Coordenação

Foi adotada a tecnologia de WS para o desenvolvimento do software de coordenação. Este software faz a coordenação de serviços do subsistema de alimentação. Este software também possui a arquitetura MVC (como explicado na secção 2), de maneira adaptada a esta aplicação. Basicamente, existem 3 camadas: a "camada de WS", a "camada de lógica de negócio" e a "camada de acesso a dados".

A "camada de WS' é uma camada de interface (*view*) com os usuários externos. Isto não significa que exista uma "tela" de visualização. O WS permite uma interação padrão entre serviços, que são publicados em XML, permitindo o acesso de qualquer interface que compreenda esses serviços. Dessa maneira, os usuários "cliente", "Op.Alimentação" e "supervisório" poderão acessar o software de coordenação por paginas de internet (HTTP, JSP, ASP, etc) e aplicativos em qualquer linguagem, celulares, *palms*, entre outros dispositivos que possuam acesso a internet.

As "camadas de lógica de negócio" (*controller*) e a "camada de acesso a dados" (*model*) serão acessadas pela "camada de WS". Estas serão DLLs (*dynamic-link library*), isto é, bibliotecas que serão consultadas quando solicitado pelo WS, já que estas DLL são módulos que contém funções e dados que podem ser consultados por outros módulos (aplicativos ou DLL) (MICROSOFT, 2009b) . Como se pode concluir pelo nome, a "camada de lógica de negócio" contem as ações do software interrelacionando os métodos do WS com a "camada de acesso a dados". Por sua vez, a camada de acesso a dados é responsável pelo acesso e armazenagem de dados no banco de dados deste ambiente distribuído.

A seguir, na Figura 11, tem-se uma ilustração das camadas descritas anteriormente. Estas camadas foram subdivididas em categorias de atuação. WSTeleoperação, se refere a métodos relacionados a atividades relacionadas ao Teleoperador (Op.Alimentação) e ao Telecomando (monitoração e teleoperação realizada pelo Op.Alimentação). WSAlimentação é a "camada de WS" relacionada ao subsistema de alimentação, sendo responsável pela interação com o "cliente" e "supervisório", com a execução de atividades relacionadas ao pedido, cadastro de cliente, estado de equipamento, etc.

lnTeleoperador, bdTeleoperador são as “camadas de lógica de negócio” e “acesso a dados” responsáveis pelas atividades relacionadas ao teleoperador. Por sua vez, as camadas lnTelecomando e bdTelecomando estão relacionadas ao Telecomando e as lnAlimentacao e bdAlimentacao estão relacionadas a atividades referentes ao subsistema de alimentação, assim como o WSAlimentação.

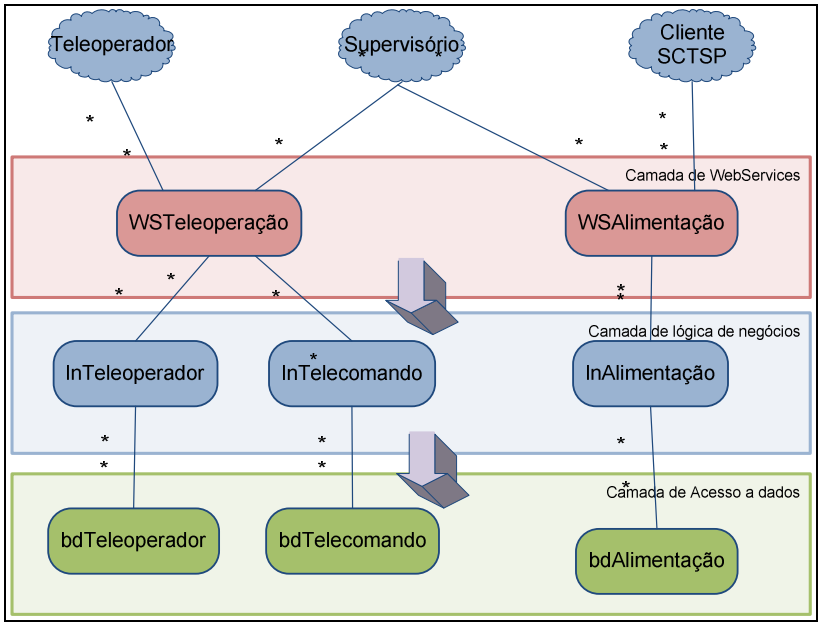


Figura 11 - Arquitetura do software de coordenação

a) Diagrama de Classes

Dada a arquitetura proposta anteriormente e ilustrada na Figura 12, desenvolveu-se um diagrama de classes para este software de coordenação de modo que, além de atender aos requisitos descritos nos “casos de uso” para o Op.Alimentação (Teleoperador), cliente (SCTSP) e supervisório, ainda respeite a estrutura requerida. Dada a quantidade de métodos existentes e o número de interrelações entre as classes, optou-se por criar também um diagrama de sequencia para cada caso de uso do software de coordenação, possibilitando assim uma melhor visualização da lógica do software. A seguir, também é feita uma descrição de cada um dos métodos existentes.

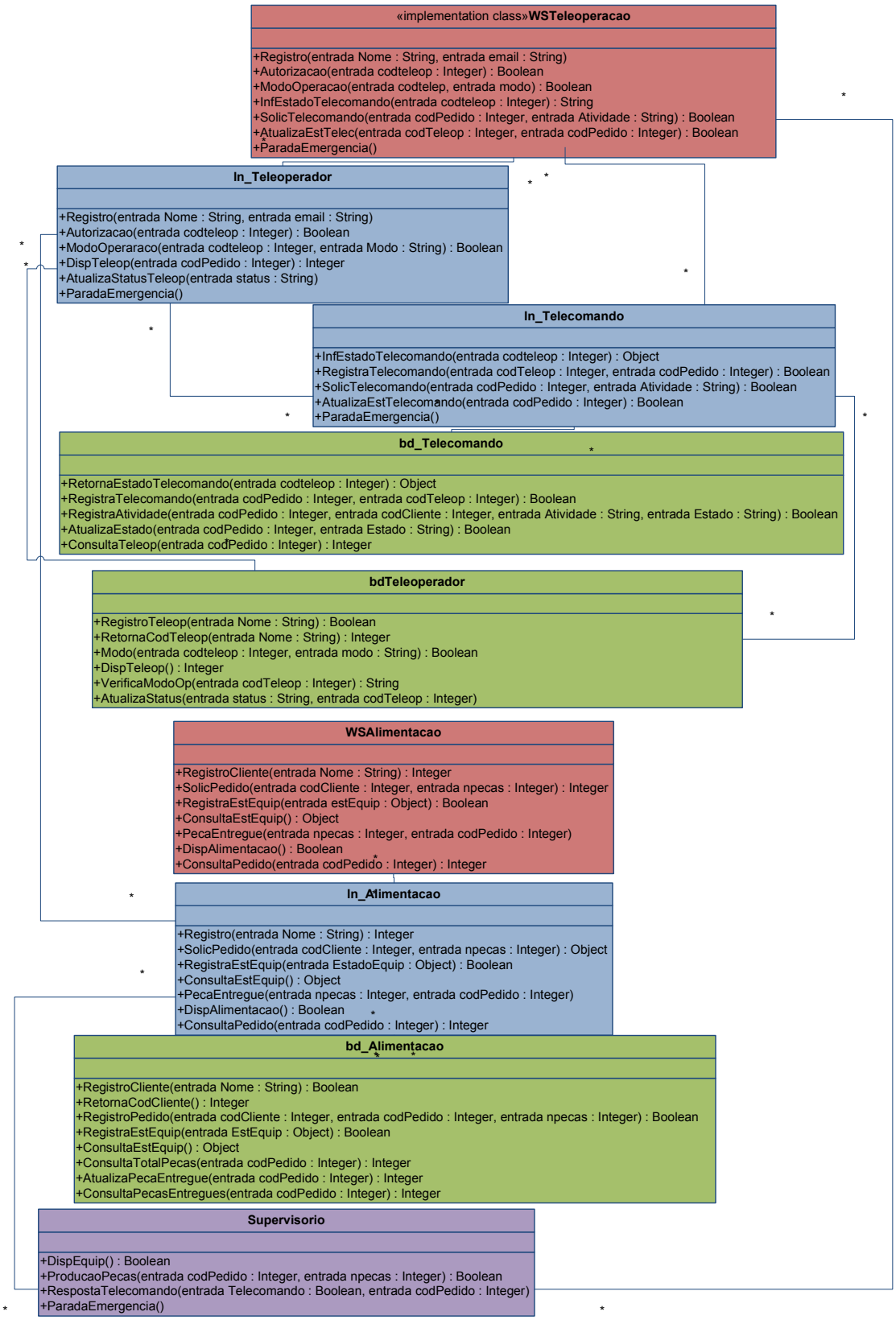


Figura 12 - Diagrama de classes do software de coordenação

Como pode ser observada no diagrama de classes da Figura 12, a classe “Supervisório” é um WS, possuindo métodos são publicados no padrão SOAP para que o software de coordenação tenha acesso. O WSAlimentação e WSTeleoperação também são WS e disponibilizam serviços que são acessados pelos usuários “Op.Alimentação” (teleoperador) e “cliente” (SCTSP). As demais classes são acessadas e ativadas somente pelas WS da “camada de WS”. O acesso ao banco de dados será feito somente pelas classes (bdAlimentação, bdTelecomando e bdTeleoperador) da “camada de acesso a dados”. Este banco de dados será melhor descrito no item 6.2.3.

b) Diagrama de Seqüência

Devido ao grande número de métodos torna-se difícil determinar a seqüência global do comportamento do software. Com isto o diagrama de seqüência foi aqui utilizado para representar os aspectos estruturais e comportamentais do software de coordenação (LI & LILUS, 2000). O diagrama de seqüência é uma das ferramentas UML usadas para representar interações entre objetos de um cenário, realizadas através de operações ou métodos (procedimentos ou funções). Este diagrama enfatiza a ordenação temporal em que as mensagens são trocadas entre os objetos de um sistema. Entende-se aqui por mensagens os serviços solicitados de um objeto a outro, e as respostas desenvolvidas para as solicitações (BOOCH; RUMBAUGH; JACOBSON, 2000).

Cadastro de Teleoperador

O caso de uso de cadastro de teleoperador está ilustrado a seguir no diagrama de seqüência da Figura 13. Primeiramente, a interface do Op.Alimentação ativa o serviço de registro do WSTeleoperação, que por sua vez, consulta o método correspondente da classe lnTeleoperador. Esta classe, responsável pela lógica de negócio, encaminha a informação referente ao teleoperador (nome e e-mail) a classe bdTeleoperador, e se esta efetuar o registro com sucesso no banco de dados (BD) e retornar o codTeleop, a classe

lnTeleoperador enviará um e-mail ao endereço fornecido contendo este código (codTeleop). O codTeleop é necessário para a autenticação do Op.Alimentação no sistema. Como o escopo deste trabalho não é administração de acesso e segurança do ambiente, não foi aqui desenvolvido nenhuma política de senhas.

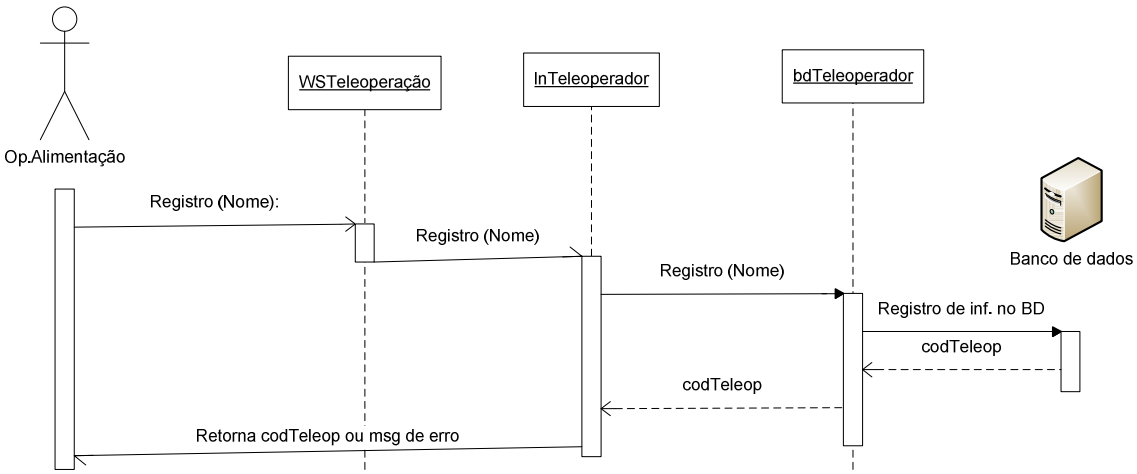


Figura 13 - Diagrama de seqüência para o Registro de Teleoperador

Acesso do Teleoperador

O acesso do Op.Alimentação ao software de coordenação está ilustrado no diagrama de seqüência da Figura 14. Primeiramente, o Op.Alimentação informa o codTeleop ao WSTeleoperação, que por sua vez, informa este código ao lnTeleoperador através do método autorização. Esta classe então, solicita a alteração do "status" deste teleoperador para "online" à classe bdTeleoperador, através do método "conexão". Se a classe bdTeleoperador conseguir alterar o campo "status" do BD através do índice fornecido pelo "codTeleop", será retornado um booleano com valor de TRUE a classe lnTeleoperador. Por sua vez será informado ao WSTeleoperação que, por fim, informa ao Op.Alimentação que a conexão foi realizada com sucesso.

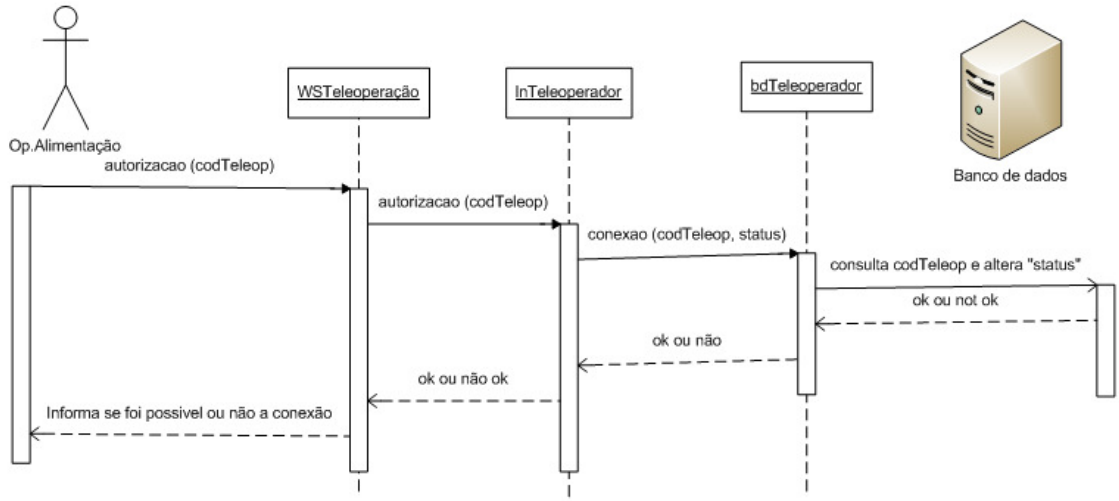


Figura 14 - Diagrama de seqüencia para o acesso do Teleoperador

Modo de Operação do Teleoperador

A escolha do modo de operação que o Op.Alimentação irá trabalhar é modelada no diagrama de seqüência da Figura 15. Basicamente, o Op.Alimentação deve selecionar um modo de operação (Teleoperação ou Monitoração), sendo este modo e o codTeleop enviado ao WSTeleoperação. O WS envia ao InTeleoperador estes dados que, por sua vez, os informa ao bdTeleoperador juntamente com a data e hora atual (campo “DataHora”). Dessa maneira, o bdTeleoperador os registra no BD e, se isto for executado com sucesso, um booleano é enviado a todas as camadas.

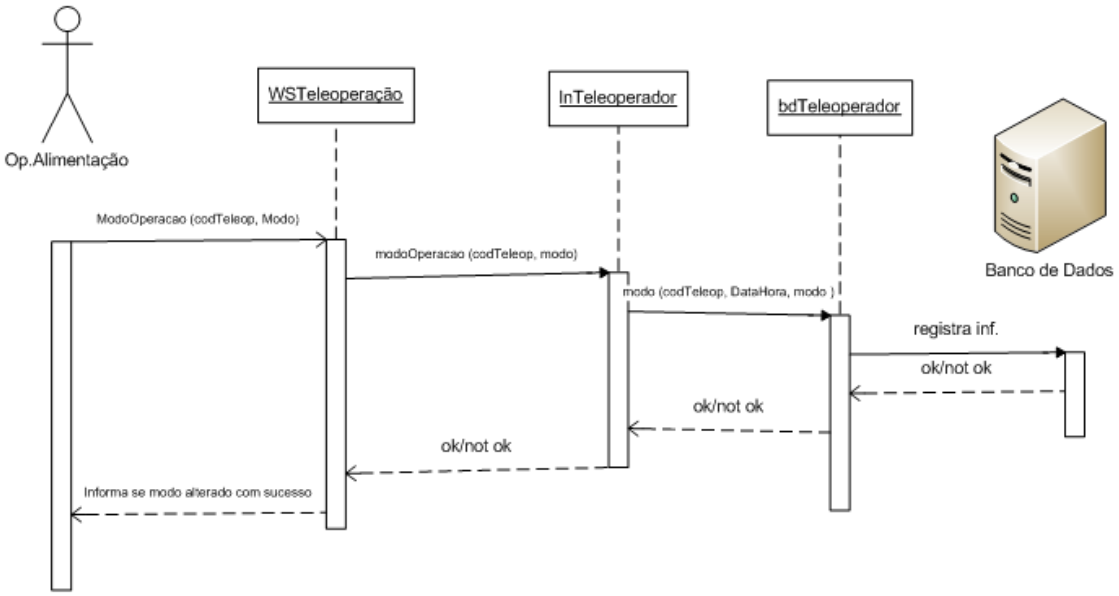


Figura 15 - Diagrama de seqüência para a escolha do modo de operação

Cadastro do cliente

De maneira muito semelhante ao cadastro do teleoperador é o cadastro do cliente (SCTSP) e seu diagrama de seqüência pode ser observado na Figura 16. Primeiramente, a interface do cliente ativa o serviço de RegistroCliente do WSAlimentação, que por sua vez, consulta o método correspondente da classe InAlimentação. Esta classe, responsável pela lógica de negócio, encaminha a informação referente ao cliente (nomeCliente) a classe bdAlimentação, e se esta efetuar o registro com sucesso no banco de dados (BD) e retornar o codCliente, a classe InAlimentação retorna o codCliente ao WSAlimentação que por sua vez, o fornece ao cliente. O codCliente é necessário para a autenticação do cliente no sistema. Como o escopo deste trabalho não é administração de acesso e segurança do ambiente, não foi aqui desenvolvida nenhuma política de senhas.

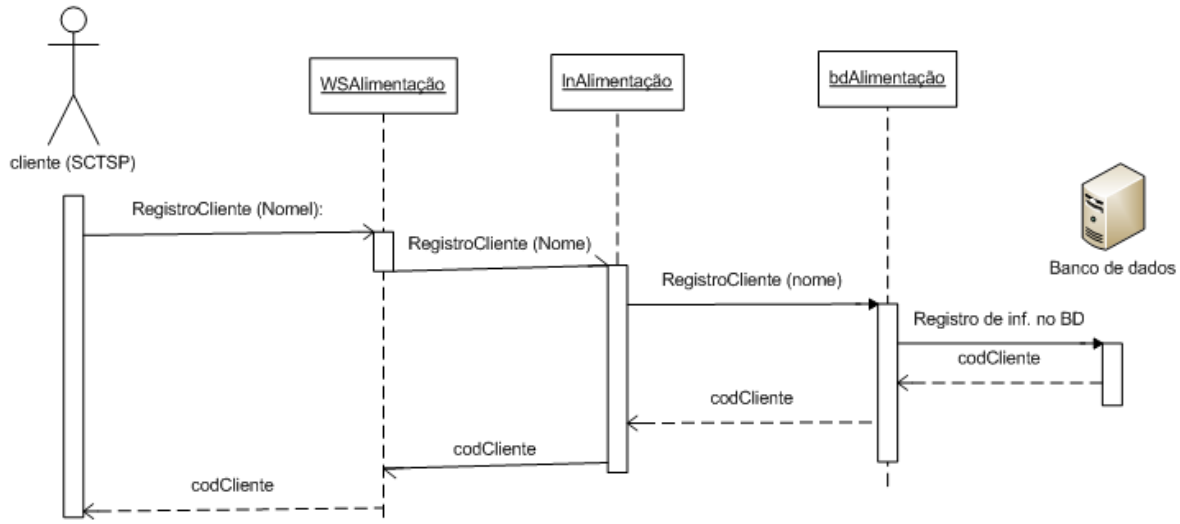


Figura 16 - Diagrama de seqüência para o cadastro do cliente

Verificação de Disponibilidade do Subsistema de Alimentação

Um dos casos de uso do ator "cliente" é a consulta de disponibilidade do subsistema de alimentação. A verificação desta disponibilidade é necessária, pois o "cliente", ou SCTSP, necessita gerenciar não somente o subsistema de alimentação, mas o conjunto formado por todos os subsistemas, e para isso é necessário saber a disponibilidade de todos os subsistemas para orquestrar a execução global dos pedidos.

Resumidamente, o cliente verifica a disponibilidade com o WSAlimentação, que encaminha o pedido a classe InAlimentação. Esta classe, por sua vez, verifica a disponibilidade de equipamento com o agente "supervisório" e a disponibilidade de Teleoperador com o InTeleoperador.

Ao analisar este trabalho de maneira independente a ação de orquestração do STCSP, isto é, na situação em que o ator "cliente" atua somente como um cliente do subsistema de alimentação, tem-se que ao fazer uma solicitação de pedido, é necessária novamente a verificação de disponibilidade de teleoperador e do equipamento. Esta situação é analisada no diagrama de seqüência referente ao caso de uso de solicitação de pedido (Figura 17).

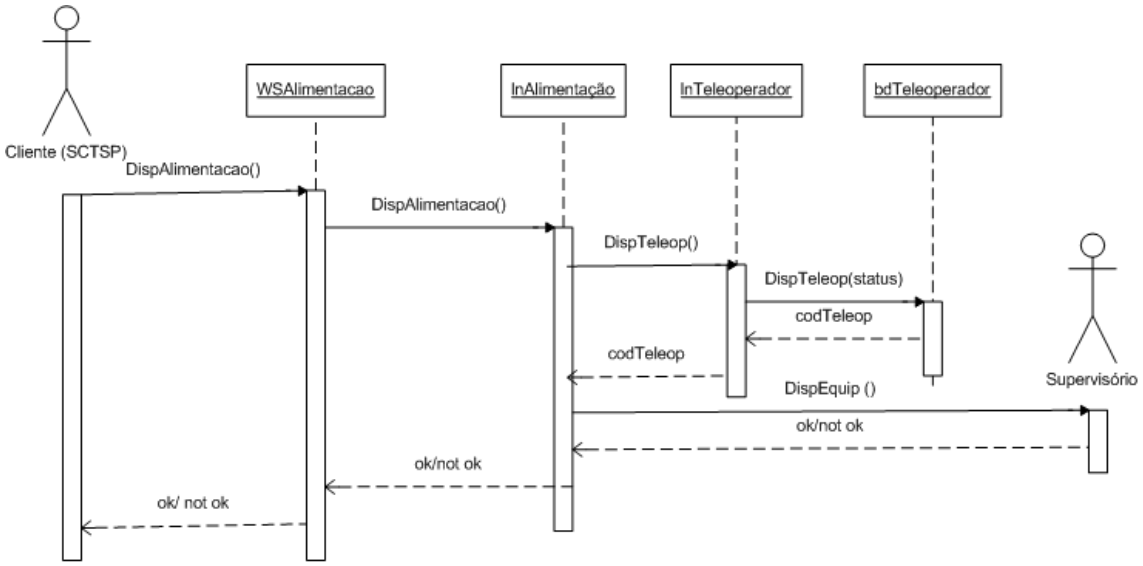


Figura 17 - Diagrama de seqüência para a verificação de disponibilidade do subsistema de alimentação

Solicitação de Pedido

A seguir tem-se o diagrama de seqüência para caso de uso de solicitação de pedido (Figura 18). Este diagrama de seqüência é um pouco mais elaborado que os demais, pois a camada de "lógica de negócio", lnAlimentação, verifica a disponibilidade do equipamento com o supervisório e também a disponibilidade de um teleoperador para que seja possível a concretização do pedido e só assim armazená-lo no BD através do bdAlimentação.

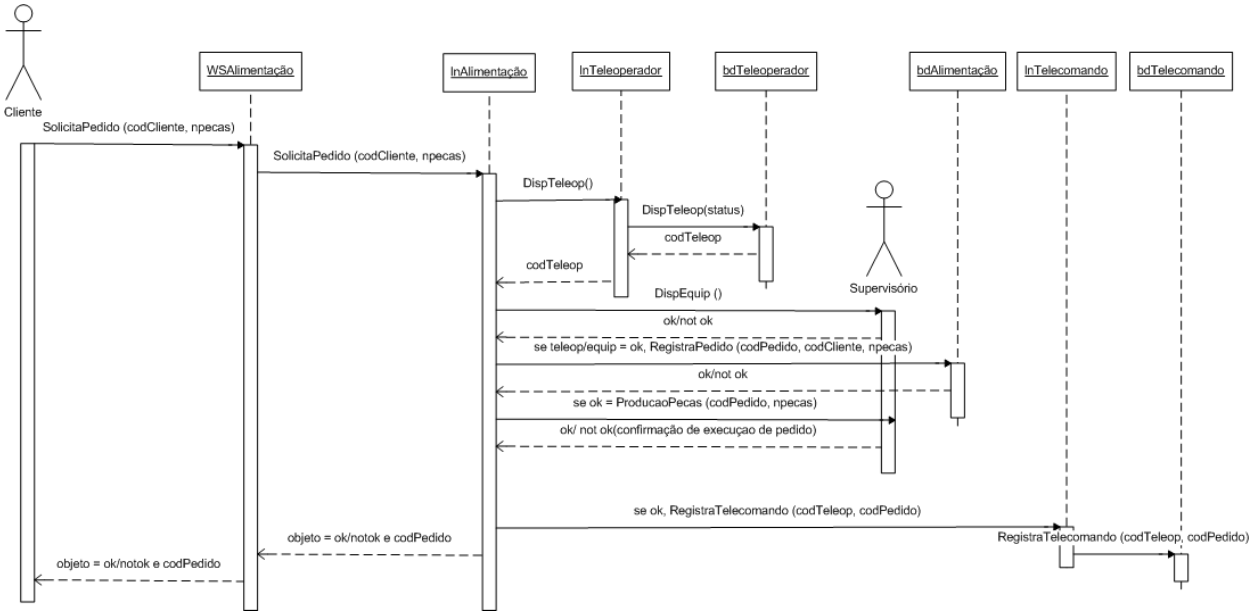


Figura 18 - Diagrama de seqüência para a solicitação de pedido

Como se pode observar na Figura 18, primeiramente o cliente faz a solicitação do pedido, informando ao WSAlimentação o número de peças desejadas e o seu codCliente. Em seguida, o pedido é encaminhado a camada de "lógica de negócios" (lnAlimentação), que verifica a disponibilidade do subsistema de alimentação com o supervisório, através do método "DispEquip()". Ela verifica também a disponibilidade de teleoperador através do método "DispTeleop" que pertence a classe lnTeleoperador. Esse método faz a classe bdTeleoperador verificar no banco de dados se existe algum teleoperador cujo "status" está "online". Caso existe, é retornado a classe lnAlimentação o seu código codTeleop e seu "status" é alterado para "trabalhando".

Se a classe lnAlimentação confirmar a disponibilidade do equipamento e de teleoperador, o pedido pode ser registrado. Assim, através do método RegistraPedido (codPedido, codCliente, npecas) da classe bdAlimentação, os dados do pedido são registrados no BD. Em seguida, lnAlimentação solicita a execução do pedido ao supervisório, através do método ProducaoPecas(codPedido, npecas). Por fim, é retornado ao WSAlimentação e ao cliente se o pedido pode ser cadastrado e qual o codPedido caso ele tenha sido feito com sucesso.

Infelizmente, devido à restrição de espaço para a representação do diagrama da Figura 18, não foi descrito o registro das informações no BD,

porém subentende-se que este está sendo feito pela “camada de acesso a dados”, assim como ocorre nos demais casos.

Execução de Pedido: Verificação do Modo de Operação

Para assegurar a execução de cada ação referente à produção de uma peça, o supervisório deverá solicitar o telecomando ao WSTeleoperação. Caso o modo de operação do teleoperador deste pedido seja de “monitoração”, não será necessário aguardar a autorização do Teleoperador para a execução da atividade. Caso o modo esteja em “teleoperação”, o supervisório deve aguardar a autorização do op.Alimentação para prosseguir. O diagrama de seqüência (Figura 19) a seguir ilustra a verificação do modo de operação durante a execução do pedido.

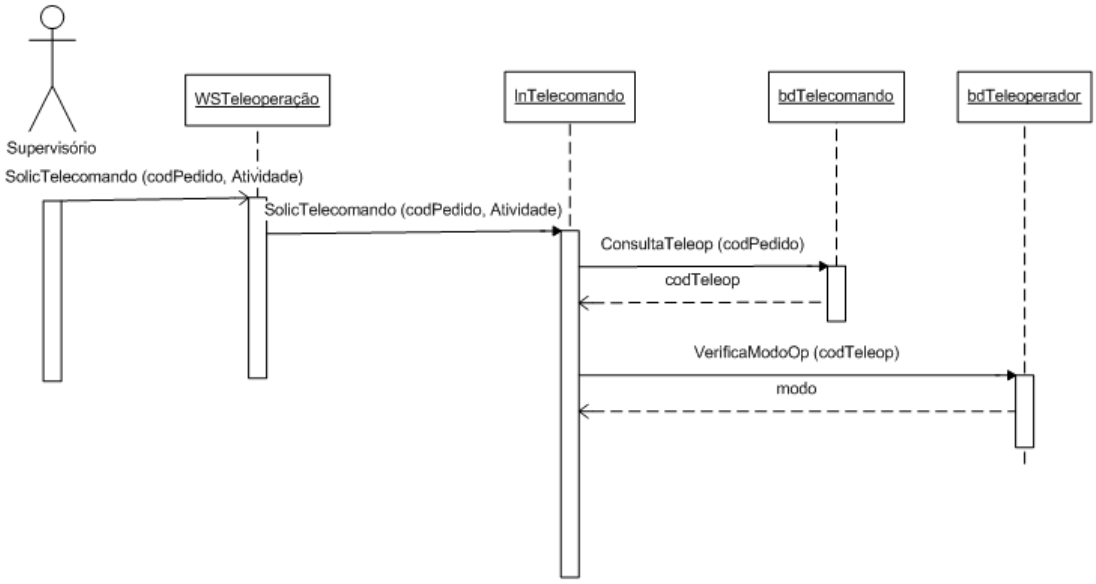


Figura 19 - Diagrama de seqüência para execução do pedido: verificação do modo de operação

Execução do Pedido: Modo Monitoração

Dando continuidade ao diagrama de seqüência anterior, se o modo de operação for de monitoração, ter-se-á o diagrama de seqüência da Figura 20. Quando o modo for de monitoração, simplesmente será atualizado no BD,

através da classe bdTelecomando, a atividade que é executada no subsistema de alimentação. Após esse registro, a classe lnTelecomando indica ao supervisório que ele pode prosseguir com a atividade (método: RespostaTelecomando()).

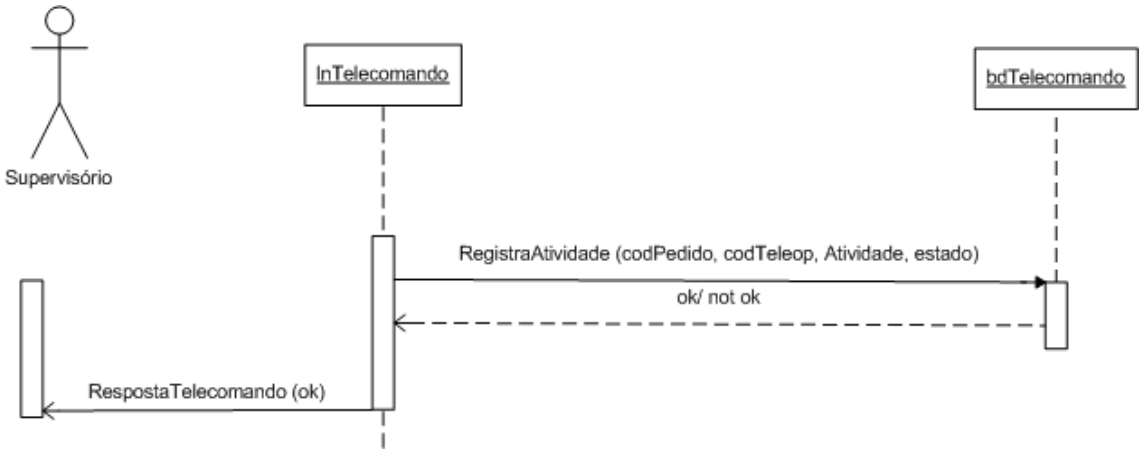


Figura 20 - Diagrama de seqüência da execução de pedido: Monitoração

Execução do Pedido: Modo Teleoperação

A diferença básica entre o modo de monitoração e o modo de teleoperação está no método RegistraAtividade (codPedido, codTeleop, Atividade, estado). Isto porque, quando o supervisório solicita um telecomando, é registrada no BD a atividade atual que será executada no subsistema de alimentação. Caso o modo for de monitoração, a variável "estado" receberá o valor "concluído" e a classe lnTelecomando envia a resposta do telecomando para o supervisório. Se o modo é de teleoperação, a variável "estado" é registrada com o valor "pendente", para que o Op.Alimentação possa consultar as atividades pendentes e autorizá-las para que a classe lnTelecomando possa enviar uma resposta ao supervisório. A Figura 21 ilustra esse processo de execução de pedido no modo de teleoperação.

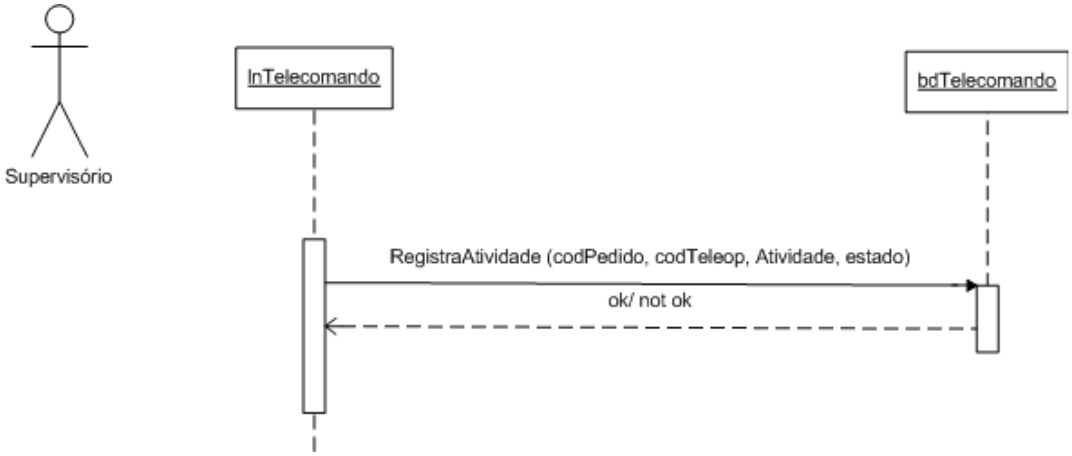


Figura 21 - Diagrama de seqüência de execução de pedido: modo Teleoperação

Informação sobre o estado – Teleoperação

O Op.Alimentação quando está em modo de teleoperação precisa consultar o software de coordenação para saber se existe alguma atividade pendente no subsistema de alimentação e assim autorizar sua execução. A verificação se o estado da atividade está pendente é feita pela classe lnTelecomando. No diagrama de seqüência da Figura 21 é mostrado como esta consulta é feita. Em seguida, no item "atualização de estado" é mostrada a autorização de execução desta atividade.

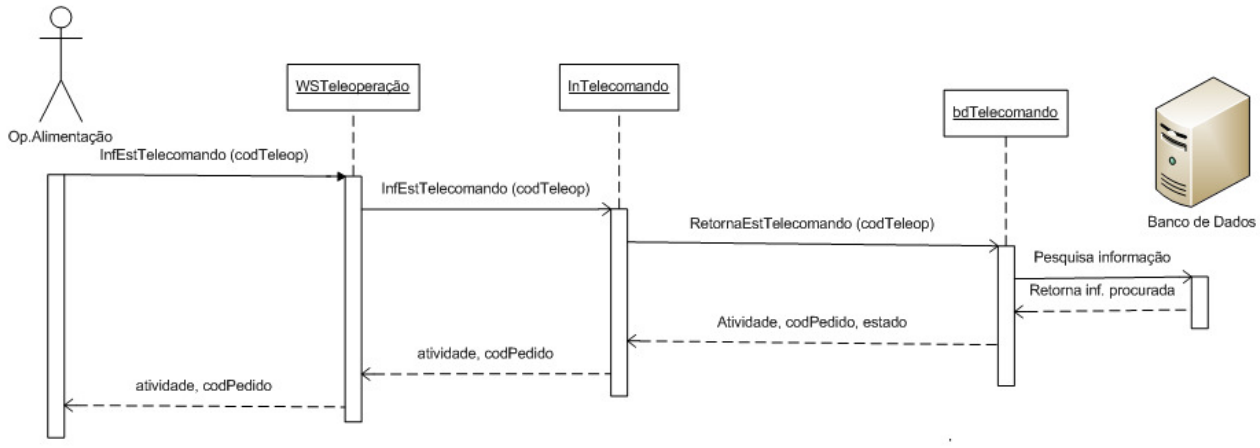


Figura 22 - Diagrama de seqüência informação sobre o estado das atividades – Teleoperação

Atualização de estado - Teleoperação

Quando em modo de teleoperação, o Op.Alimentação deve consultar periodicamente se existe alguma atividade que está pendente em algum pedido do subsistema de alimentação e autorizá-la ou não. Caso a autorize, seu "estado" passa a ser "concluído" e a lógica de negócio (lnTelecomando), envia uma resposta ao subsistema indicando que a atividade pode ser executada (RespostaTelecomando (ok)), de acordo com o ilustrado no diagrama da Figura 23.

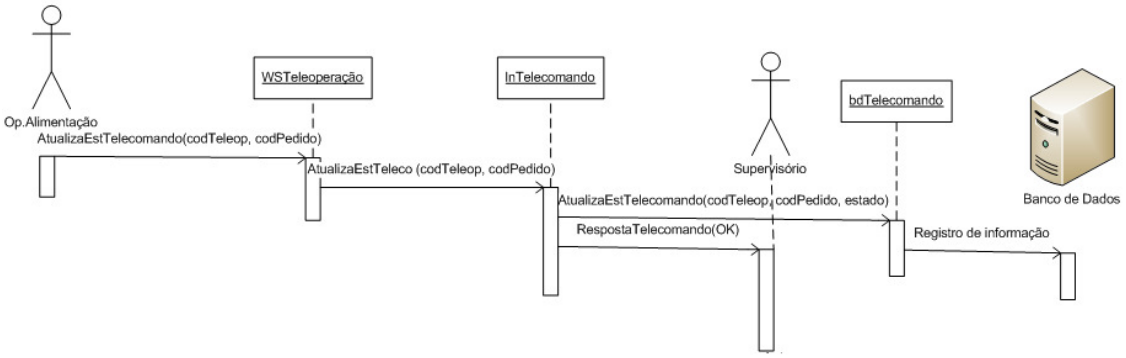


Figura 23 - Diagrama de seqüência para atualização de estado: Teleoperação

Registro de peças entregues

Para que o software de coordenação tenha um registro ao longo do tempo da quantidade de peças que já foram entregues, desenvolveu-se um método que atualiza periodicamente o número de peças entregues. O "supervisório" assim que finaliza uma peça, aciona o método do WSAlimentação informando o número total de peças entregues. Esta seqüência pode ser observada no diagrama de seqüência da Figura 24.

Esta atualização é importante para que a classe lnAlimentação possa acompanhar a execução do pedido e quando as peças entregues for igual a quantidade de peças solicitadas pelo cliente, esta classe deverá finalizar a ordem de pedido e liberar o teleoperador para que execute outro atendimento.

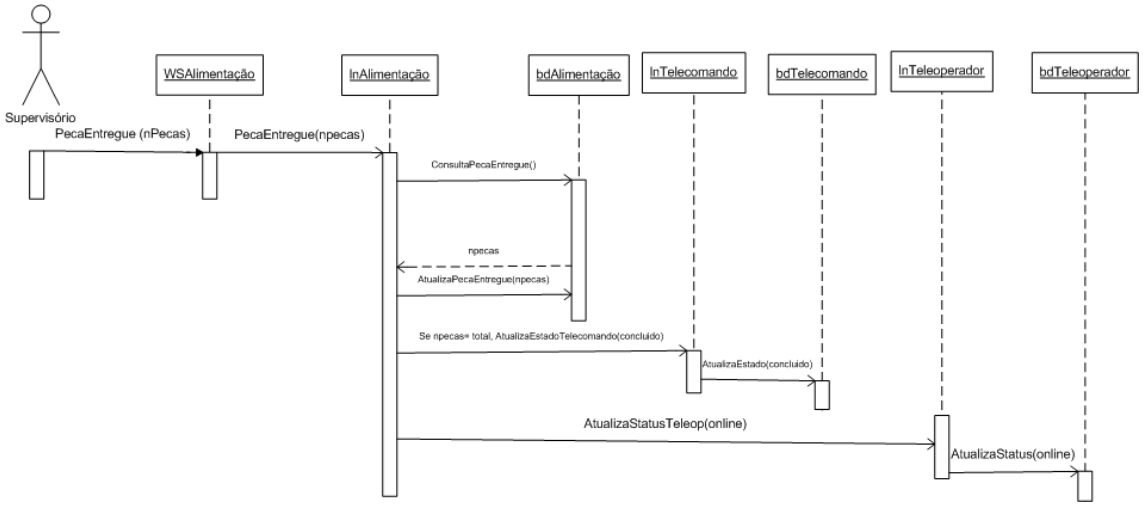


Figura 24 - Diagrama de seqüência para atualização de peças entregues

Consulta de peças entregues

Também, um dos casos de uso do "cliente" é consultar o pedido, verificando quantas peças já foram entregues. Para isso, o WSAlimentação disponibiliza um método que permite realizar esta consulta, de acordo com o diagrama de seqüência da Figura 25. De acordo com o codPedido, é realizada uma consulta no BD e retornado o número de peças (npeca) já produzidas.

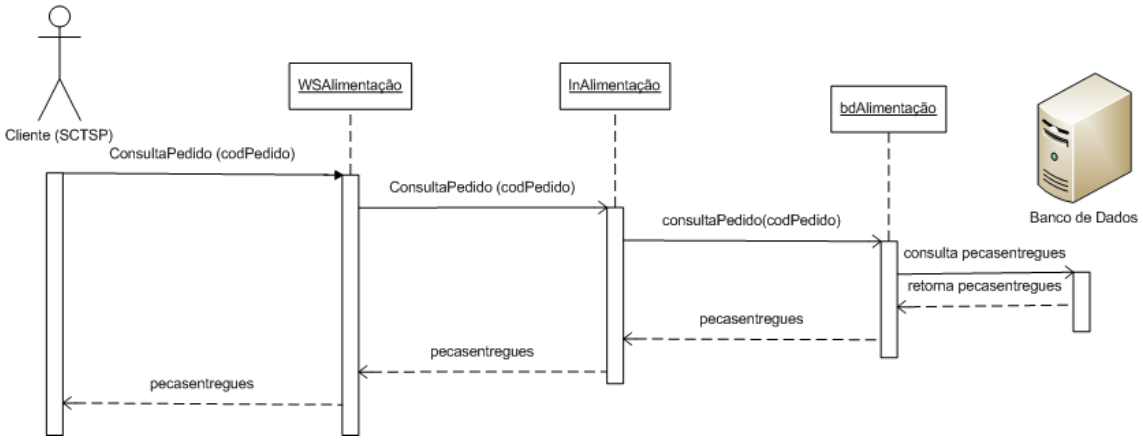


Figura 25 - Diagrama de seqüência para a consulta de peças entregues

Registro dos estados dos equipamentos

É importante armazenar periodicamente os estados dos sensores, atuadores e demais dispositivos presentes no subsistema de alimentação. Isto

permite que o Op.Alimentação tenha acesso a informação atualizada sobre o estado do equipamento/subsistema. Este armazenamento é descrito no diagrama de seqüência da Figura 26.

Primeiramente, o supervisório acessa o WSAlimentação e através do método RegistraEstadoEquipamento, envia um objeto da *structure* "Equipamento" (criada na linguagem VB.NET e que contém propriedades referentes a cada dispositivo) com todos os estados dos dispositivos. Este objeto é enviado a classe lnAlimentação e em seguida ao bdAlimentação, onde é registrada no BD.

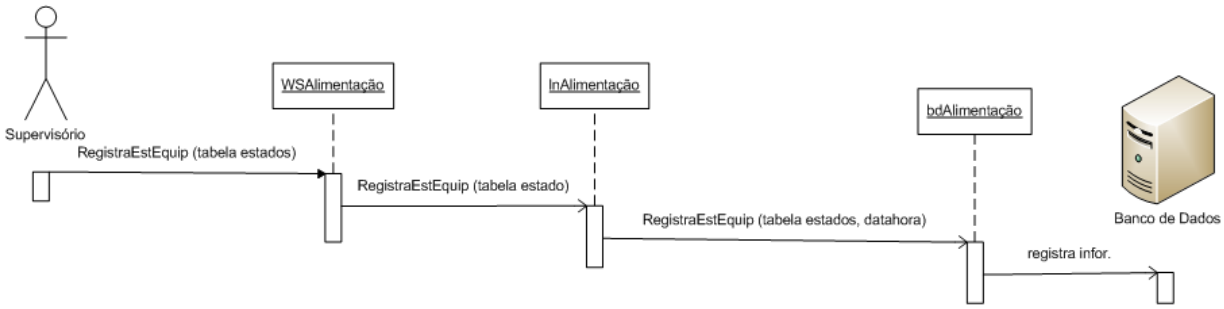


Figura 26 - Diagrama de seqüência para o registro de informação sobre dispositivos do subsistema de alimentação

Consulta dos estados dos equipamentos

Para que o Op.Alimentação possa monitorar e teleoperar o subsistema de alimentação, é necessário conhecer os estados atuais dos dispositivos. O diagrama de seqüência da Figura 27 mostra como a consulta destes estados é feita. A classe lnAlimentação irá consultar o dado mais recente e através do bdAlimentação será disponibilizada uma tabela com os estados dos dispositivos (sensores, atuadores, etc).

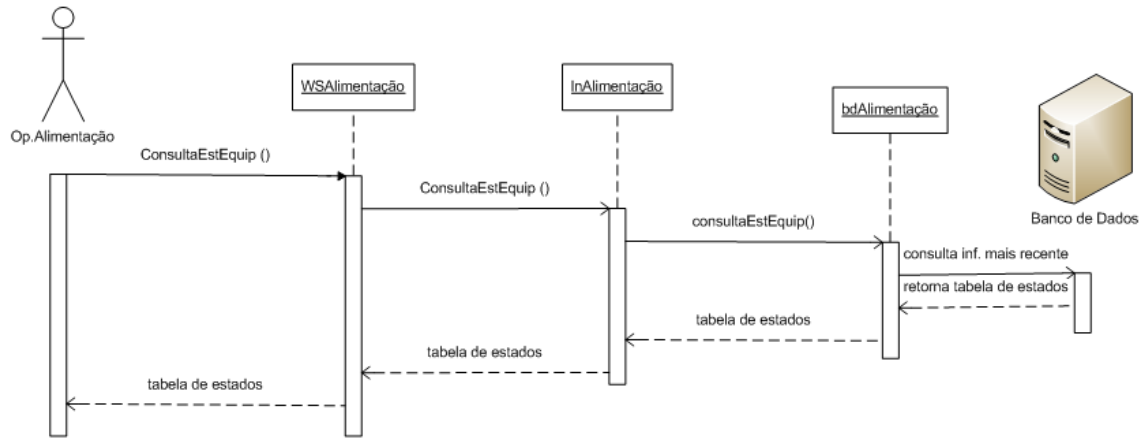


Figura 27 - Diagrama de seqüência para consulta de estado de dispositivos do subsistema de alimentação

Parada de Emergência

Como o Op.Alimentação pode acompanhar o estado dos dispositivos, monitorando e teleoperando este subsistema produtivo, tem-se que ele deverá ter acesso a parada de emergência deste SP, caso detecte alguma situação anormal ou de perigo. O diagrama de seqüência deste caso de uso está na Figura 28. Basicamente, o Op.Alimentação solicita a "Parada de Emergência" ao WSTeleoperação. Este por sua vez, encaminha o pedido a classe InTelecomando, que informa a classe InAlimentação e por fim, realiza a parada de emergência no agente "supervisório".

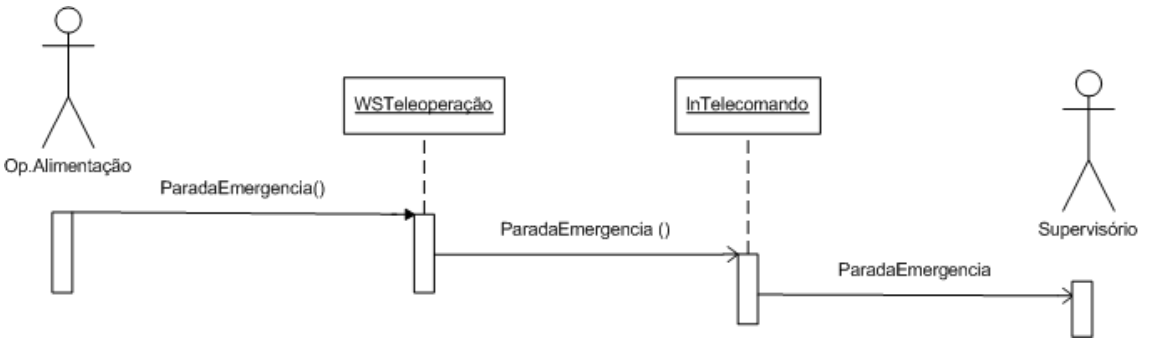


Figura 28 - Diagrama de seqüência para parada de emergência

4.2.3 Banco de dados

Dados as classes e métodos modelados anteriormente para o software de coordenação desenvolveu-se um banco de dados para atender as necessidades destas rotinas computacionais. Na Figura 29, tem-se o modelo-entidade deste BD.

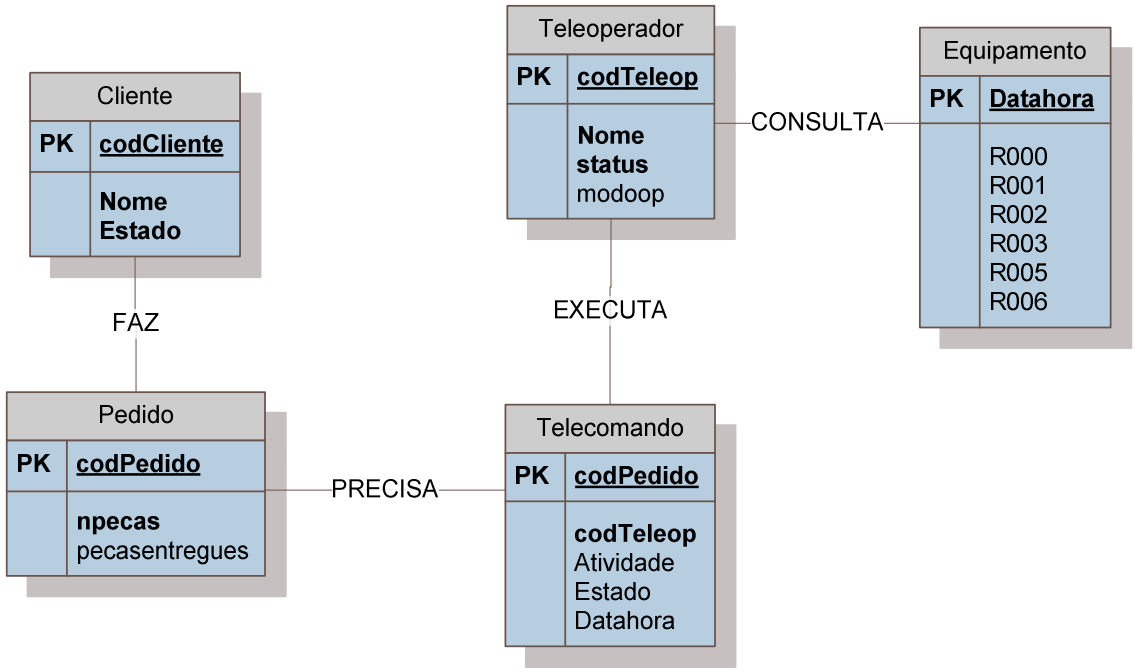


Figura 29 - Modelo entidade relação do BD utilizado

No modelo da Figura 29 pode-SE observar que as tabelas são indexadas pelos campos que possuem PK (*primary-key*). A seguir, tem-se uma descrição de cada tabela:

- **Cliente:** Esta tabela contém as informações referentes ao cliente. Cada cliente tem um nome (*string*) e um codCliente (*Integer*) para indexação desta tabela. O campo estado (*string*) poderá receber os valores “ativo” e “inativo”;
- **Pedido:** Esta tabela armazena as informações dos pedidos realizados pelo subsistema. Cada pedido conterá um codPedido (*integer*) para sua indexação. São também armazenadas as quantidades de peças solicitadas no campo npecas (*integer*), e a quantidade de peças entregues (*integer*);

- **Telecomando:** Esta tabela contem as informações referentes ao telecomando que está sendo executado no momento atual. Esta tabela é indexada pelo codPedido a que se refere o telecomando. O campo “atividade” é uma string, que contém a descrição da atividade que está sendo realizada. O campo “estado” é uma *string* e contem a informação sobre a execução atual da atividade. Existem somente duas opções para o campo “estado”: “pendente”, se a atividade ainda não foi executada; e “concluída”, se a atividade já foi concluída. O campo “datahora” contem a informação sobre a hora em que a atividade foi concluída para se manter um histórico de finalização do pedido.
- **Teleoperador:** Esta contém as informações sobre os Op.Alimentação existentes. Todos os campos desta tabela são *strings*. O campo “*status*” se refere a situação do teleoperador, isto é, se ele está “*online*” (disponível para execução de pedido), “ocupado” (se já estiver fazendo um pedido) ou “*offline*” (se não estiver logado no sistema). O campo “modoop” , se refere ao modo de operação selecionado pelo teleoperador e pode possuir os valores de “teleoperação” e “monitoração”.
- **Equipamento:** Esta tabela é indexada pelo campo “datahora”(datetime), e contém a situação dos dispositivos de detecção e de atuação.

4.3 Instalações Físicas

O subsistema automatizado de montagem que estava instalado no Laboratório de Sistemas de Automação da Escola Politécnica era um produto da empresa FESTO baseado numa arquitetura tipo stand-alone, isto é, sem comunicação com o mundo externo, e do ponto de vista interno tinha uma estrutura centralizada das funções de supervisão e controle. Para emular um sistema de manufatura disperso e distribuído foi assim concebido e implementado um tipo de *retrofitting* desse sistema de modo que cada estação

de trabalho tenha autonomia para programar e executar seus serviços e que a comunicação entre estas estações seja via internet atendendo assim uma estrutura aberta e distribuída. Desse modo, cada estação de trabalho é agora vista como um subsistema produtivo com seu próprio controle local. O hardware de controle local de cada subsistema foi implementado através de CLPs. O CLP adquirido para o subsistema de alimentação foi o do fabricante SIEMENS AG, com a seguinte configuração:

- **CPU 412-2 PCI:** Esta CPU (*Central Processing Unit*) é instalada no barramento PCI do computador local (Figura 30). Possui dois modos de comunicação, PROFIBUS e MPI, utilizados para se comunicar com o módulo de IOs distribuído ETM200.

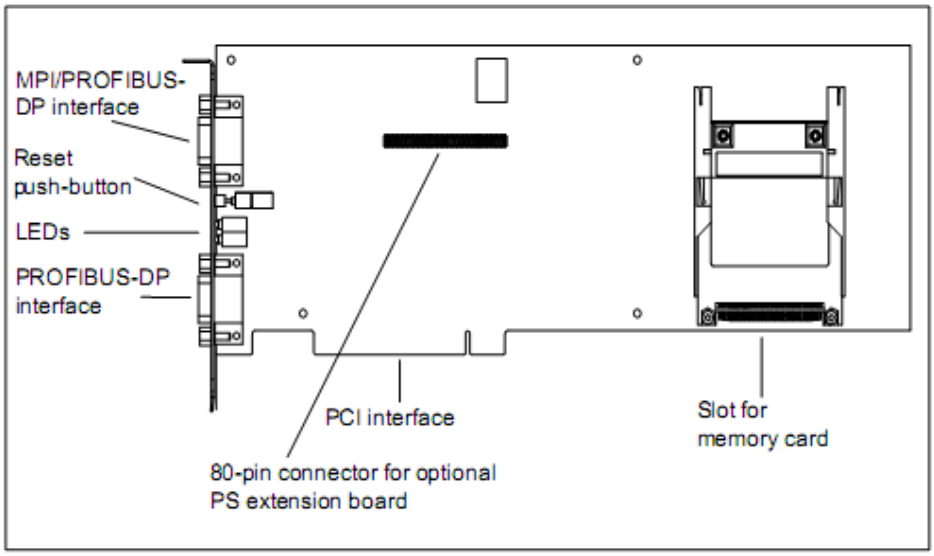


Figura 30 - CPU 412-2 PCI (Fonte: SIEMENS, 2002c)

- **ET200M:** Trata-se de um módulo de IOs distribuído, que foi instalado no gabinete que fica fisicamente próximo aos dispositivos de controle do subsistema de alimentação. Este módulo possui um trilho DIN, onde são conectados o módulo de comunicação PROFIBUS IM153-3, os dois módulos de IO com 8 entradas digitais e 8 saídas digitais cada um, e a fonte PS307. Esta fonte converte a tensão de rede (110/220 V) para a tensão DC que é utilizada na alimentação dos circuitos do módulo ET200M e dos dispositivos de atuação e monitoração do subsistema. O módulo de comunicação IM153-3 é responsável

pela comunicação entre a CPU 412-2 e o módulo ET200M, sendo esta comunicação estabelecida via PROFIBUS. A Figura 31 ilustra a configuração do módulo ET200M utilizado.

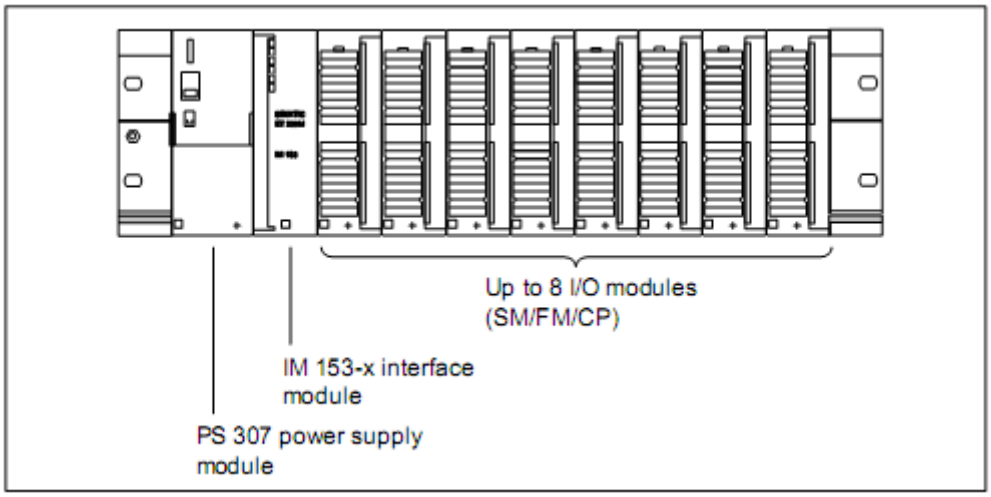


Figura 31 - Módulo ET200M (SIEMENS, 2000d)

As fiações e conexões elétricas foram refeitas e utilizou-se anilhas para identificar o cabeamento e facilitar a realização de reparos. O diagrama elétrico esquemático da instalação está no ANEXO A.

4.3.1 Programação do CLP

O programa do CLP foi desenvolvido em linguagem *Ladder* e está no ANEXO B. Ele foi construído de modo a facilitar a interação com o software supervisório sobre ele. A idéia principal é que cada passo de operação seja ativado através de uma *flag,* isto é, ao mudar o valor de uma variável na memória do CLP, será iniciada a execução da operação em questão.

Para o subsistema de alimentação existem cinco operações principais:

- **mEstendeCilindro** - Responsável por avançar o cilindro que posiciona a peça (1A+);
- **mRecuarCilindro** – Responsável por recuar o cilindro que posiciona a peça (1A-);
- **mBracoHorario** – Responsável por movimentar o braço giratório para o subsistema de inspeção (3A+);

- **mBracoAntiHorario** – Responsável por movimentar o braço giratório para o subsistema de alimentação (3A-);
- **mDesligaVentosa** – Responsável por desligar o vácuo que segura a peça durante o transporte;
- **mLigaVentosa** – Responsável por ligar o vácuo que segura a peça durante o transporte;

Como já dito, para que o software supervisório altere o valor destas variáveis acima foi utilizada a comunicação OPC. Para isso, foram estabelecidas configurações no programa do CLP para que este atue associado a um OPC Server no computador em que está instalada a placa da CPU. Este OPC Server é responsável pela troca de dados entre a CPU e o software supervisorio (*OPC Client*).

Primeiramente, no software "Set PC/PG Interface" do pacote SIMATIC NET fornecido pela SIEMENS para configuração da CPU, selecionou-se a opção "PC Internal (local)", configurando assim o modo que será utilizado para realizar esta troca de dados (Figura 32).

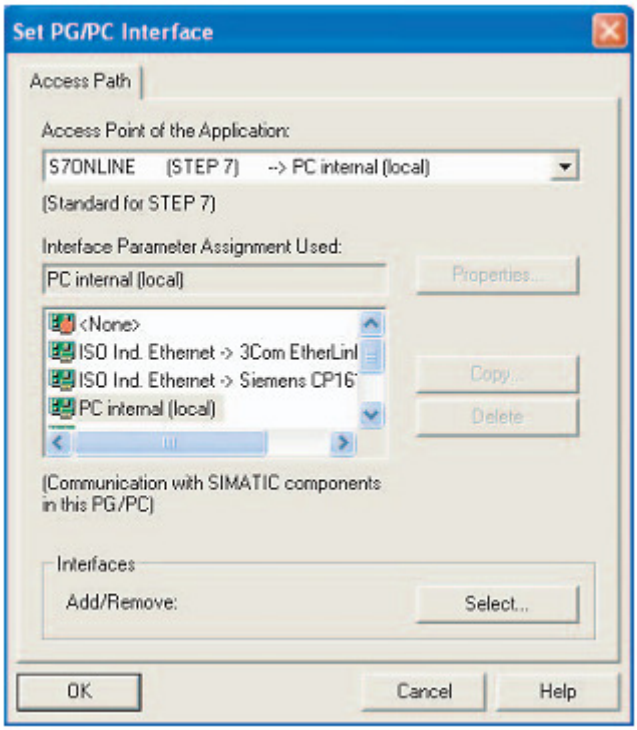


Figura 32 - Configuração do PG/PC Interface

Em seguida, no software “Station Configurator Editor”, também do pacote SIMATIC NET, selecionou-se, no primeiro índice, o OPC Server, como indicado na Figura 33. Isto configura a CPU 412-2 PCI como OPC Server para o computador em que está instalada (Figura 33). Como se trata de uma CPU 412-2, no segundo índice é que esta foi configurada.

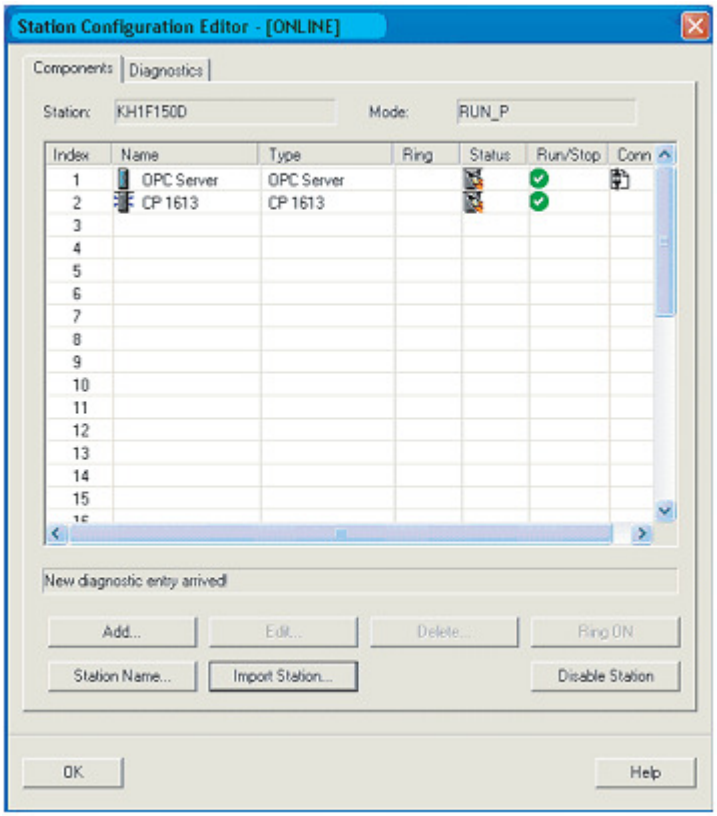


Figura 33 - Configuração do Station Configurator Editor

A programação do CLP e a configuração de Hardware (*Hardware Configuration*) existente foram realizadas no software STEP S7, também presente no pacote SIMATIC NET. Deve-se também acrescentar o OPC Server neste item, como ilustra a Figura 34. A configuração utilizada para o subsistema de alimentação encontra-se no ANEXO B.

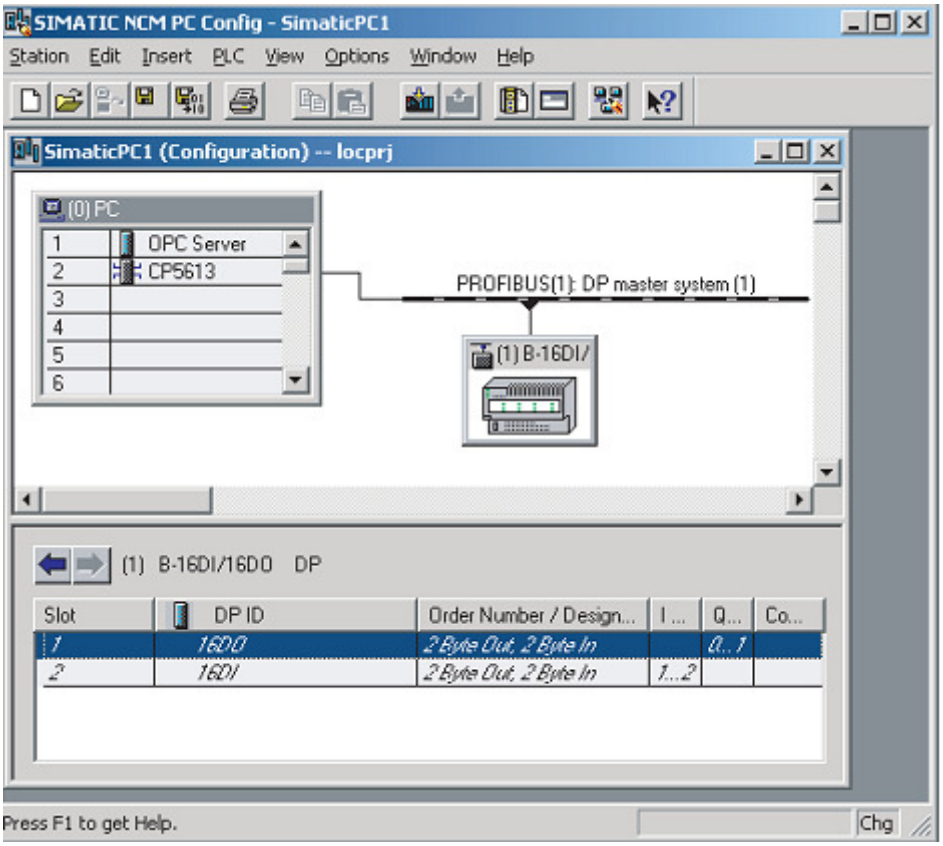


Figura 34 - Configuração no STEP 7

Por fim, configurou-se o OPC Scout (Figura 35), que assim como os outros, é um software do pacote SIMATIC NET que foi instalado junto com a CPU 412-2 PCI. A seguir, são apresentadas as variáveis do programa do CLP que são monitoradas e alteradas pelo software supervisório. Tais variáveis são mapeadas no OPC Scout para que a troca de informações seja realizada com sucesso. Detalhes de toda esta configuração podem ser encontrados no manual "*Industrial Communication Commissioning: Manual and Quick Start"* da Siemens (SIEMENS, 2008e).

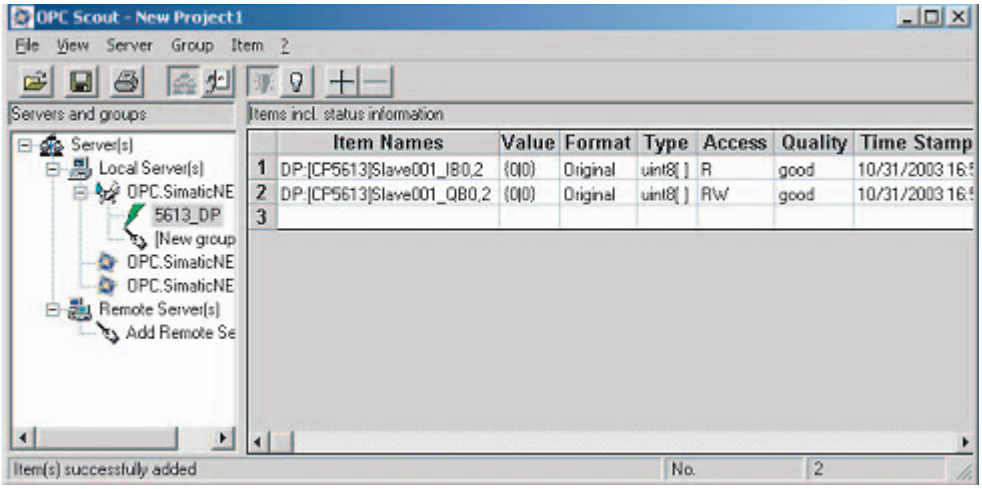


Figura 35 - Configuração do OPCScout (Fonte: SIEMENS, 2008e)

Na Tabela 3, seguem as variáveis que foram configuradas para o subsistema de alimentação:

Tabela 3 - Variáveis mapeadas para comunicação OPC

Nome do Item	Identificador	Descrição
S7:[S7Alimentacao]iCilindroRecuado	R000	Lê o sensor que identifica se o cilindro foi recuado
S7:[S7Alimentacao]iCilindroEstendido	R001	Lê o sensor que identifica se o cilindro foi estendido
S7:[S7Alimentacao]iBracoAlimentacao	R002	Lê o sensor que informa se o braço está sobre a alimentação
S7:[S7Alimentacao]iBracoInspecao	R003	Lê o sensor que informa se o braço esta sobre a inspeção
S7:[S7Alimentacao]iPecaVentosa	R005	Lê o sensor que identifica se a peca foi presa na ventosa
S7:[S7Alimentacao]iSensorPecasBuffer	R006	Lê o sensor que detecta a presença de pecas no buffer
S7:[S7Alimentacao]mEstendeCilindro	W000	Estende o cilindro que tira a peça do buffer
S7:[S7Alimentacao]mRecuarCilindro	W001	Recua o cilindro que retirou a peça do buffer
S7:[S7Alimentacao]mBracoHorario	W002	Movimenta o braco sentido horário
S7:[S7Alimentacao]mBracoAntiHorario	W003	Movimenta o braco no sentido anti-horário
S7:[S7Alimentacao]mLigaVentosa	W004	Liga a ventosa
S7:[S7Alimentacao]mDesligaVentosa	W005	Desliga a ventosa

4.4 Software Supervisório

A listagem/codificação do software supervisório que foi desenvolvido está no ANEXO D. Para simplificar a implementação deste software, utilizou-se a mesma tecnologia que a usada para o desenvolvimento do software de Coordenação, a tecnologia de WS.

Neste trabalho, para acessar os WSs desenvolvidos, tanto no software de coordenação quanto neste supervisório, foram utilizadas páginas desenvolvidas em *HTML (Hyper Text Markup Language)* e *Javascript*. HTML é uma linguagem de marcação, onde as "marcas" são utilizadas para se desenvolver páginas da web acessadas por qualquer navegador (W3CSCHOOLS, 2009a). *Javascript,* por sua vez, é uma linguagem de *script* desenvolvida para adicionar interatividade às páginas em HTML (W3CSCHOOLS, 2009b). Trata-se de uma linguagem de programação relativamente simples, interpretada, que não necessita de licença para ser utilizada (W3CSCHOOLS, 2009b). O código Javascript é interpretado por qualquer *browser* e pode ser inserido dentro de um código em HTML (W3CSCHOOLS, 2009b).

O software supervisório é um WS, por isso, tem um tempo de vida pré-definido pelo servidor em que está instalado (MELO, 2008). Sendo assim, para que o supervisório se mantenha em funcionamento, controlando o subsistema de alimentação, utilizou-se a função Javascript "setTimeOut". Esta função possui dois parâmetros: o primeiro é o nome do método criado que deverá ser chamado, e o segundo parâmetro é o tempo em milisegundos de chamada. Por exemplo, a função setTimeOut("AcordaSupervisorio", 2000), irá chamar o método "AcordaSupervisório", desenvolvido no código Javascript da página, a cada 2 segundos.

Este método, "setTimeOut" é uma das ferramentas AJAX (*Asynchronous Javascript and XML*) disponibilizadas no ambiente de desenvolvimento Visual Studio 2008, utilizado neste trabalho. AJAX é um modo de programação que permite a execução de chamadas assíncronas de clientes (KHOSRAVI, 2006).

O software supervisório, além dos métodos descritos no diagrama de classes da Figura 12, que são essenciais para o correto funcionamento do software de coordenação, tem outros dois métodos: "ConectaOPC" e o "AcordaSupervisorio".

O método "ConectaOPC" é responsável por configurar o software supervisório como *OPCClient* para o servidor OPC configurado na CPU 412-2 PCI do CLP. Ele utiliza uma DLL desenvolvida no Laboratório de Sistemas de Automação (LSA da Escola Politécnica da USP), a "opcconn.dll". Esta DLL

utiliza a seguinte seqüência de comandos (Figura 36), disponibilizados no componente COM “Siemens OPC DA Automation 2.0”, para realizar a conexão.

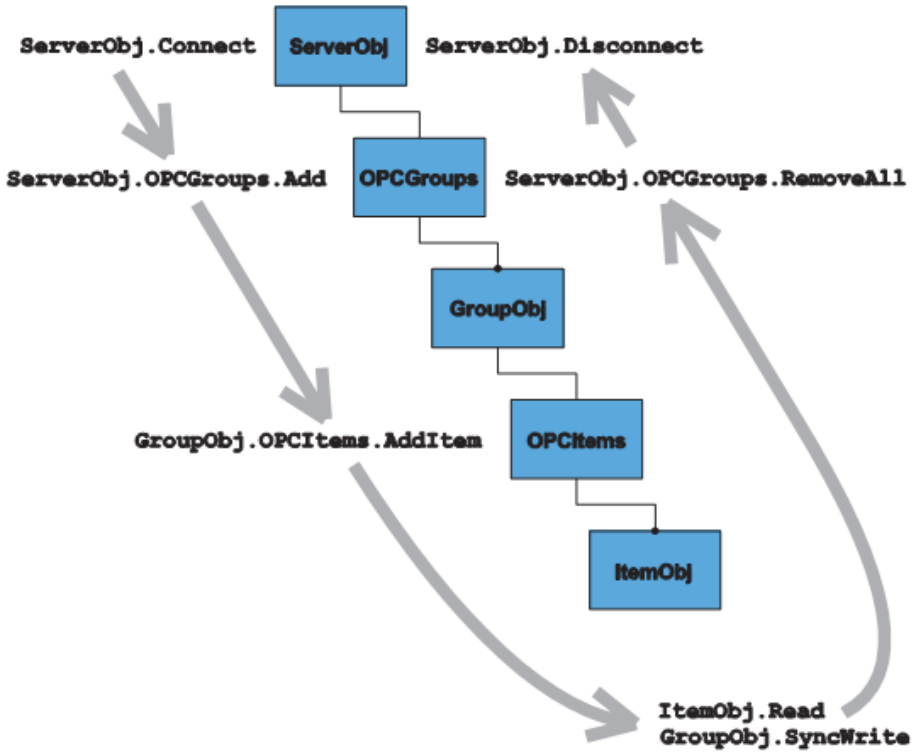


Figura 36 - Seqüência de comandos para estabelecer a conexão *OPC Client* (Fonte: Siemens, 2008f)

O método “AcordaSupervisorio”, por sua vez, é responsável por executar as operações do programa do CLP. Seu fluxo lógico é dado de acordo com a Figura 37. Basicamente, este método verifica se existe algum pedido em andamento, e caso exista, solicita autorização ao WSTeleoperação para a próxima atividade ser executada. O WSTeleoperação, responde através do método “RespostaTelecomando” do software do supervisório, que em seguida, realiza a próxima atividade pendente. Para o correto funcionamento do subsistema de alimentação, as seqüências de operações a serem realizadas são, respectivamente: estende pistão 1A, movimenta braço giratório 3A para o subsistema de alimentação, liga geração de vácuo, movimenta braço giratório 3A para o subsistema de inspeção, desliga geração de vácuo, recua pistão 1A.

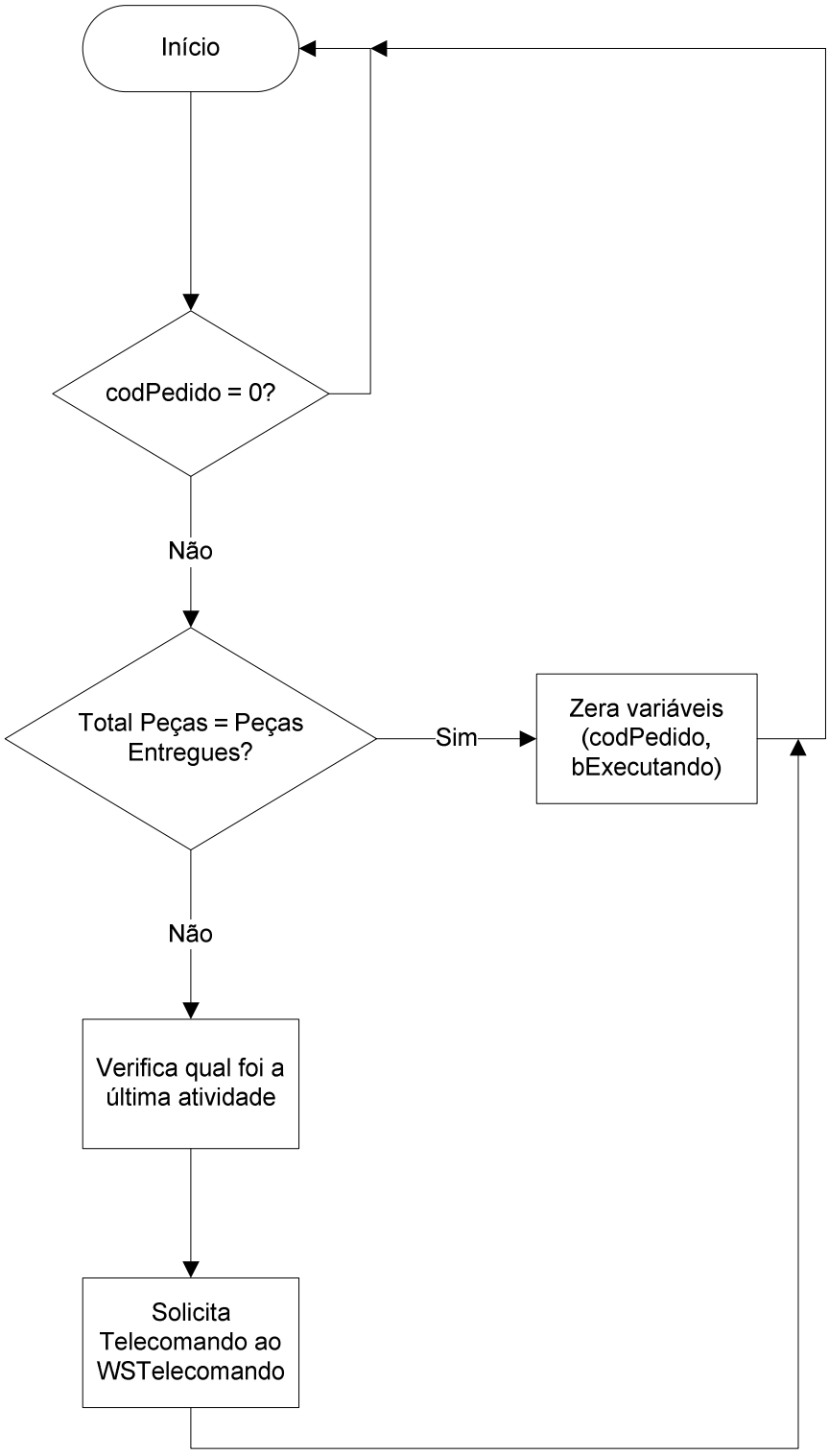


Figura 37 - Fluxograma para o Método "AcordaSupervisorio"

4.5 Interfaces dos Usuários

Para verificar e validar o processo de supervisão e controle de um sistema produtivo disperso construiu-se as páginas da Web mostradas a seguir. Estas páginas foram elaboradas utilizando a linguagem de marcação HTML e Javascript, estando seus códigos no ANEXO E. É importante frisar que as páginas foram desenvolvidas de maneira a atender os requisitos especificados na seção 6.3.1.

Acesso Principal

A página de acesso principal (default.aspx), Figura 38, possui uma breve apresentação sobre o subsistema de alimentação e duas opções de escolha “Cliente” e “Teleoperador”.

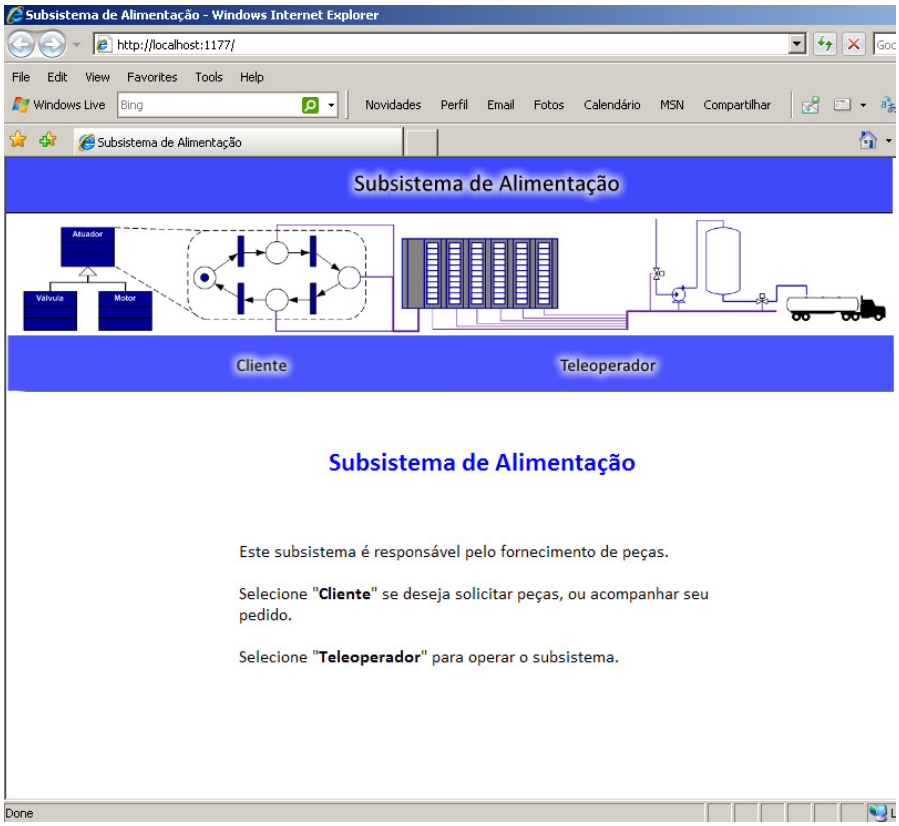


Figura 38 - Página de acesso principal

Página de *Login* do Cliente

A página de *login* do cliente (Figura 39, cliente.aspx) possui duas partes, uma para cadastro, caso seja um novo usuário e não possui ainda um "Código de Cliente", ou a opção de *Login*, caso já seja um usuário cadastrado e possua *login*. É importante frisar que só é possível acessar a página de operações do cliente, se o *login* for efetuado com sucesso.

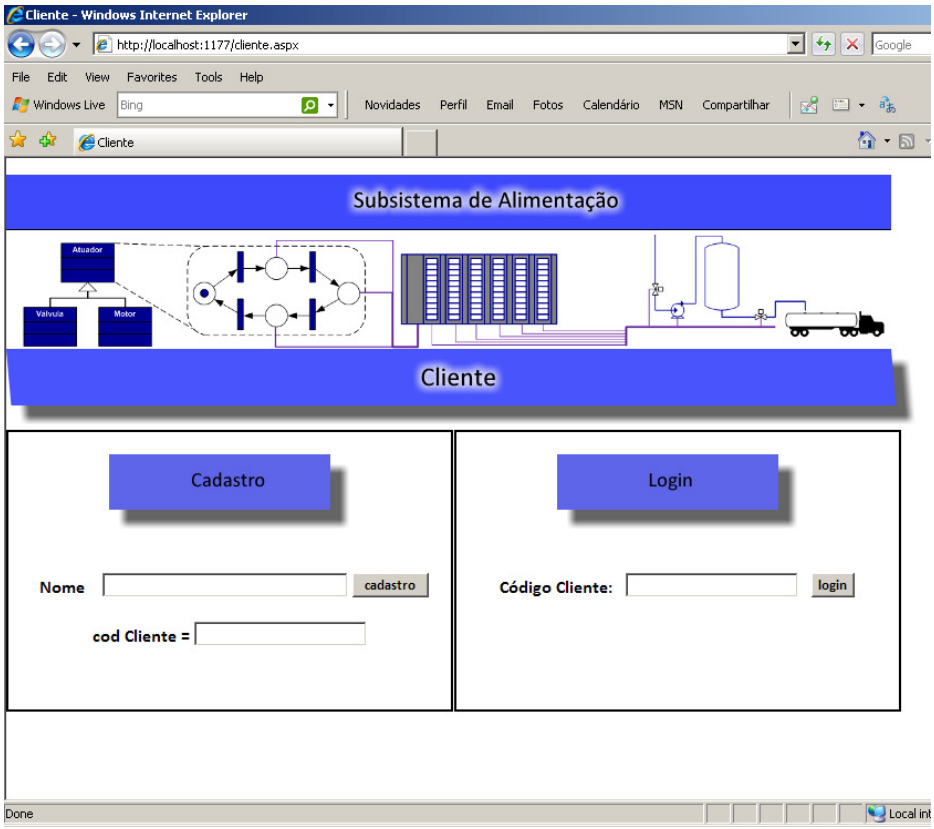


Figura 39 - Página de *login* do Cliente

Página de Operações do Cliente

A página de operações do cliente (Figura 40, opCliente.aspx) permite que ele solicite um pedido, verifique a disponibilidade do subsistema de alimentação, e confira a execução do pedido já realizado.

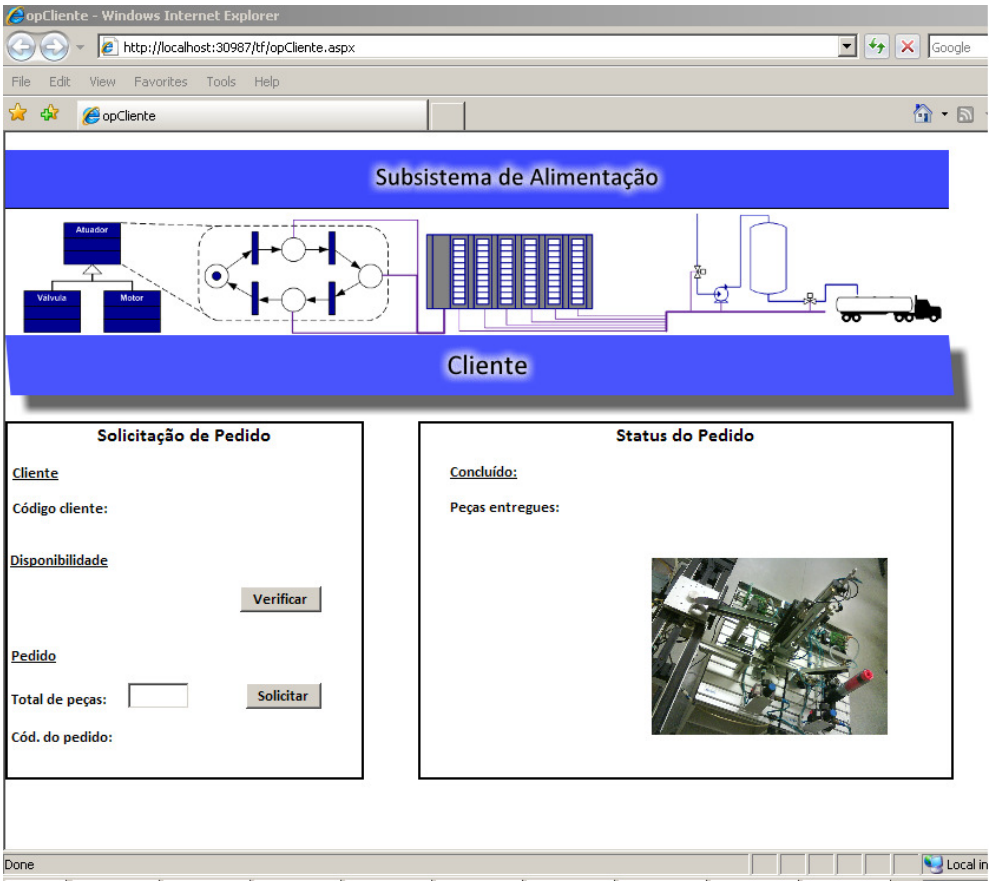


Figura 40 - Página de Operações do Cliente

Página de *Login* do Teleoperador

A página de *login* do Teleoperador (Figura 41, teleop.aspx) possui duas partes, uma para cadastro, caso seja um novo usuário e não possui ainda um "Código de Teleoperador", ou a opção de *Login*, caso já seja um usuário cadastrado e possua um código de acesso. É importante frisar que só é possível acessar a página de operações do teleoperador, se o *login* for efetuado com sucesso.

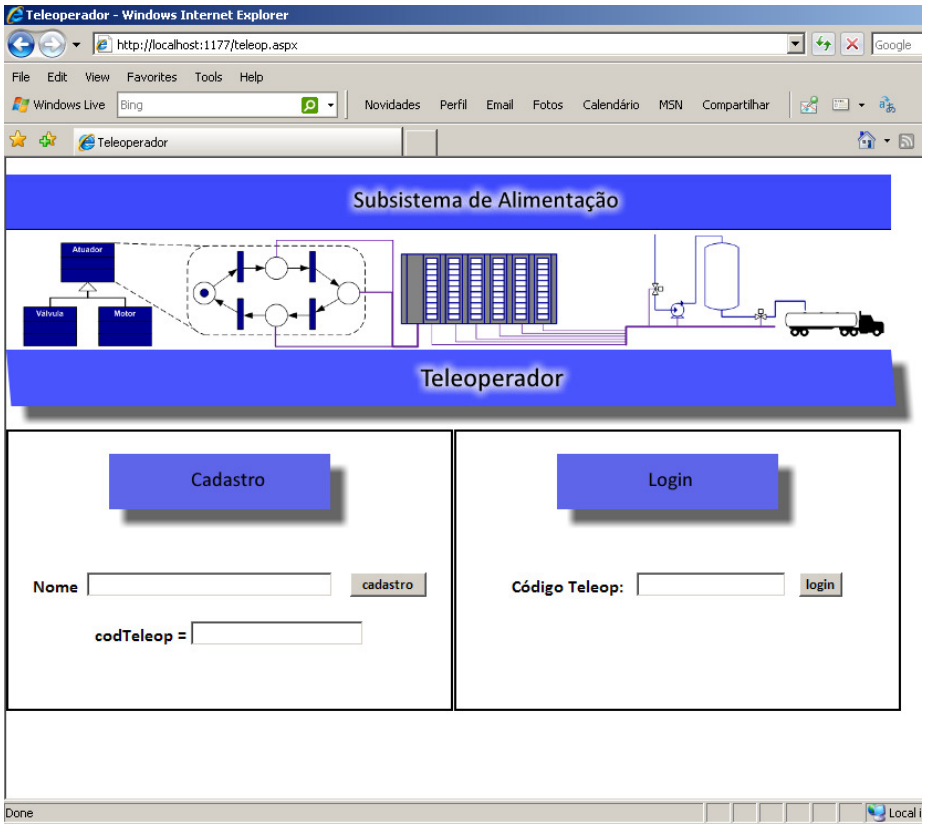


Figura 41 - Página de *login* do Teleoperador

Página de Operações do Teleoperador

A página de operações do teleoperador (Figura 42, opTeleop.aspx) permite que o teleoperador escolha o modo de operação (monitoração ou teleoperação), verifique o código do pedido que está executando, visualize o subsistema de alimentação, autorize ou não a execução da atividade pendente, efetue parada de emergência e, por fim, verifique o estado dos equipamentos do SP.

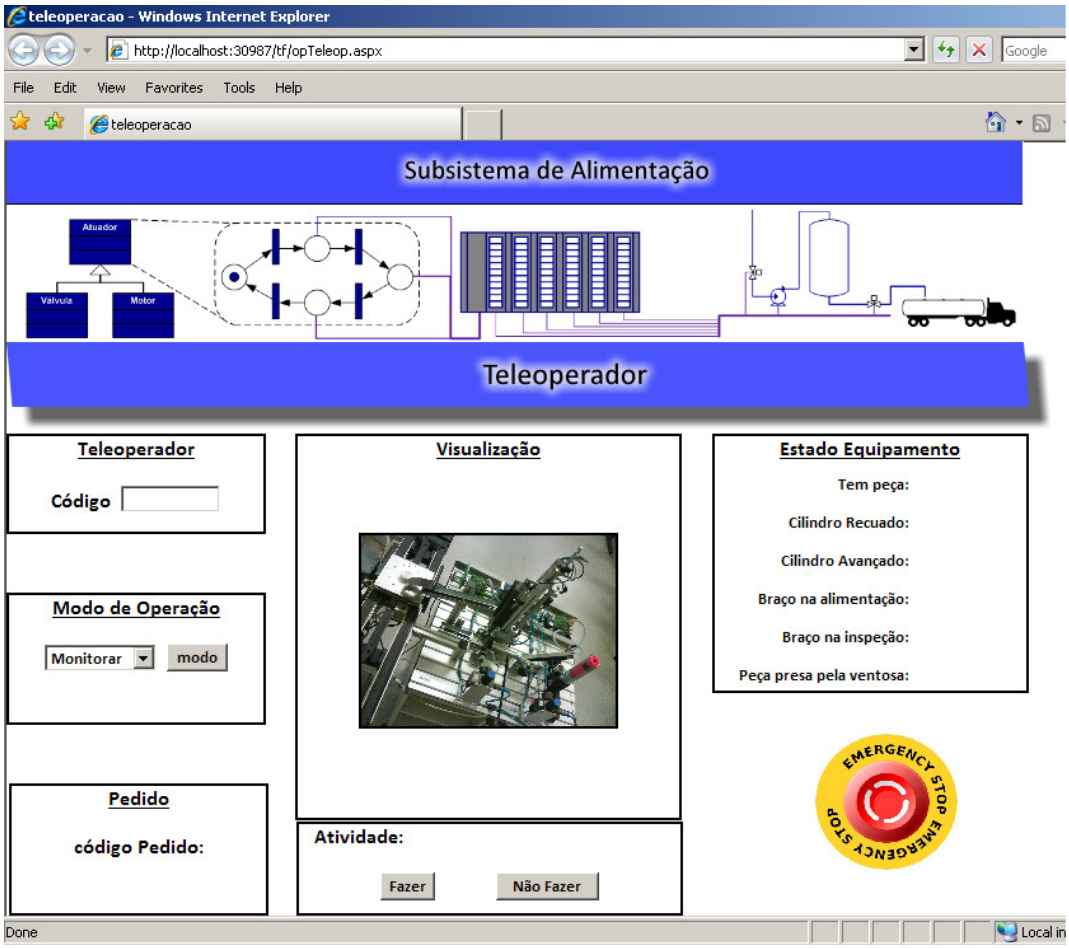


Figura 42 - Página de Operações do Teleoperador

5. CONCLUSÕES

A primeira etapa de elaboração do projeto consistiu em uma pesquisa geral sobre os fundamentos necessários para a implementação deste ambiente distribuído proposto, assim como uma definição dos requisitos e casos de usos do subsistema de alimentação, que é o caso de estudo deste projeto. Em seguida, foi elaborado o projeto e modelagem da arquitetura deste ambiente distribuído propriamente dito. Para isto, foram utilizadas as ferramentas UML para modelagem e a arquitetura MVC para o desenvolvimento do software de coordenação, procurando aproximar este projeto ao máximo possível da realidade industrial em que se aplica. Esta estruturação também permitiu a organização dos softwares e, juntamente com a modelagem em UML, foi possível uma melhor a análise do comportamento do software e a interação global de cada serviço e método criado.

Constatou-se que para atingir o objetivo principal deste projeto, que é possibilitar que um subsistema de um sistema produtivo disperso seja controlado e manipulado de maneira distribuída, é necessário assegurar que todas as interações das diversas partes do sistema sejam devidamente especificadas e que mecanismos de controle sejam implementados por razões da confiabilidade e segurança. Não é possível permitir que atores emitam ordens distintas e que produzam um resultado inesperado no comportamento do sistema. Por isso, foi importante manter a estruturação e a documentação do software desenvolvida sempre atualizada.

O software supervisório foi desenvolvido utilizando-se WS, porém este poderia ser elaborado como um aplicativo do *Windows*, que seria executado desde o *start-up* do sistema operacional. Esta solução seria mais segura que a solução adotada neste trabalho, uma vez que o software supervisório estaria sempre em funcionamento, atuando sobre o subsistema. No entanto, se esta solução fosse adotada, a comunicação do software supervisório com o software de coordenação deveria ser estabelecida via TCP/IP, o que poderia ser mais uma fonte de problema, caso esta comunicação não fosse estabelecida com sucesso.

É importante frisar que as páginas da Web desenvolvidas foram criadas apenas para testar e avaliar o funcionamento do ambiente distribuído, sendo que outras páginas ou aplicativos podem ser desenvolvidos, para se comunicar com os WSs do software de coordenação, explorando melhor o conceito de usabilidade, portabilidade, etc.

Por fim, o uso de WS e o desenvolvimento de uma arquitetura orientada a serviços possibilitam que este ambiente produtivo seja: **colaborativo**, no sentido que permite a atuação de diversos atores simultaneamente; e **distribuído**, uma vez que os WSs podem ser acessados por qualquer computador conectado a internet, em qualquer local. Em um sentido mais amplo, o software de coordenação desenvolvido neste trabalho pode ser expandido para envolver a outros subsistemas do sistema produtivo, uma vez que a especificação funcional para o subsistema está restrita apenas ao software supervisório, onde ocorre a comunicação com o CLP local.

6. REFERÊNCIAS BIBLIOGRÁFICAS

Almeida, R. R. *Model-View-Controller*. Disponível em http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/arqu/mvc/mvc.htm Acessado em Junho de 2009.

Adlemo A., Andreasson *S. Balanced automation in flexible manufacturing*, disponível em systems http://pierre.ici.ro/ici/revista/sic1996_2/art8.html, acessado em Junho de 2008.

Booch G. , Rumbaugh J., Jacobson I., *UML : Guia do Usuário.* 14ª Reimpressão. Rio de Janeiro : Elsevier,2000.

Business Wire, 2006, *SOA Software Products Drive More Than 10 Billion Web Service Transactions; Company's Products Manage and Secure over 425 Million Transactions Per Month in Production at the World's Largest Companies,* Disponível em: <http://findarticles.com/p/articles/mi_m0EIN/is_2006_Sept_18/ai_n16728778 />. Acessado em Junho de 2009.

Fonseca, M. O.. *OPC, OLE for process control*. http://www.pims.com.br/pt/divsist/auto/opc/D453319.swf acessado em Março de 2009

Goldberg K, Song D*. Unsupervised scoring for scalable Internet - based collaborative teleoperation.* In: Proceedings of IEEE Int. conf. on Robotics & Automation, New Orleans, USA, 2004

Gorodia A.; Elhajj I. *Internet based robots: applications, impacts challenges and future direction*. In: IEEE Workshop on Advanced Robotics and its Social Impacts, 2005.

Habib M.K. *Telecooperation: Concept, applications and the need from the Internet.* In: Symposium Proceedings of the Japan Institute of Electronics Information and Communication Engineers, Kyushu, Japan, 2000.

Jensen, K.; *Coloured Petri nets: basic concepts, analysis methods and practical use*. SpringerVerlag, Berlin, 1992.

Junqueira F, Miyagi P.E. *A new method for the hierarchical modeling of productive systems*. In: 7th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services, Niagara Falls, Ontario, Canada , 2006.

Khorasvi, S. *Professional Asp.Net 2.0 Server Control and Component Development.* John Wiley & Sons, 2006.

Kreger, H., *Web Services Conceptual Architecture (WSCA 1.0), IBM Software Group,* 2001. www-4.ibm.com/software/solutions/webservices/resources.htm acessado em janeiro de 2008.

Kyatera*, Plataforma óptica fiber-to-the-lab*, 2003. http://kyatera.incubadora.fapesp.br/portal/projeto-kyatera/rede-kyatera/ acessado em março de 2008.

Leal, G. J.; Bax, M. *Serviços web e evolução de serviços em TI.* Datagrama zero. Revista de Ciência da Informação. Vol.2, No.2. Abril, 2001.

Li, X.; Lilus, J.; *Checking compositions of UML sequence diagrams for timing inconsistency*. In: Software Engineering Conference, 2000. APSEC 2000.

Lira, D.N.; Melo, J. I. G.; Junqueira, F.; Morales, R.; Miyagi, P.E. *Fault detection in flexible assembly systems using Petri net*. In: IEEE Latin America Transactions. vol.6, No. 7, 2008.

Melo, J. I. G.; Junqueira, F.; Morales, R.; Miyagi, P.E. *A procedure for modeling and analysis of service-oriented and distributed productive system.* In: CASE, IEEE, Washington, 2008

Microsoft. *Web Services.* http://msdn.microsoft.com/en-us/library/ms978748.aspx acessado em março de 2009. (a)

Microsoft*. Dynamic-link library*. http://msdn.microsoft.com/en-us/library/ms682589.aspx acessado em março de 2009. (b)

Microsoft. *Visão geral de recursos e ferramentas dos SQL Server 2008.* http://msdn.microsoft.com/pt-br/library/bb500397.aspx acessado em maio de 2009. (c)

Microsoft. *Introduction to ActiveX Controls.* http://msdn.microsoft.com/en-us/library/aa751972(VS.85).aspx acessado em novembro de 2009 (d).

Microsoft. *WSE Architecture*. http://msdn.microsoft.com/en-us/library/aa529139.aspx acessado em novembro de 2009(e).

Papazoglou, M.P.; Georgakopoulos, D., *Service oriented computing*, Communications of the ACM, Vol. 46, No. 10, pp. 25–28, 2003.

Reckziegel, M. *Protocolo de Transporte Padrão SOAP.* http://imasters.uol.com.br/artigo/4379/webservices/protocolo_de_transporte_padrao_-_soap/ acessado em Novembro de 2009.

Siemens. *S7-300 Automation Module Data Manual.* Automation and Drives. Nurmberg. Germany. Abril, 2007(a).

Siemens. *Aplication for Human Machine Interfaces.* Automation and Drives. Nurmberg. Germany. Julho, 2004(b).

Siemens. *WinAC Controlling with CPU 412-2 PCI/ CPU 416-2 PCI Setting-up CPU Version 3.3*. Siemens AG. Nurmberg. Germany. 2001-2002. (c)

Siemens. *ET 200M Distributed IO Device*. Siemens AG. Nurmberg. Germany. 2000. (d)

Siemens. *Industrial Communication Commissioning: Manual and Quick Start*. Siemens AG. Nurmberg. Germany. 2008. (e)

Siemens. *Industrial Communication with PG/PC Volume 2 - Interfaces*. Siemens AG. Nurmberg. Germany. 2008. (f)

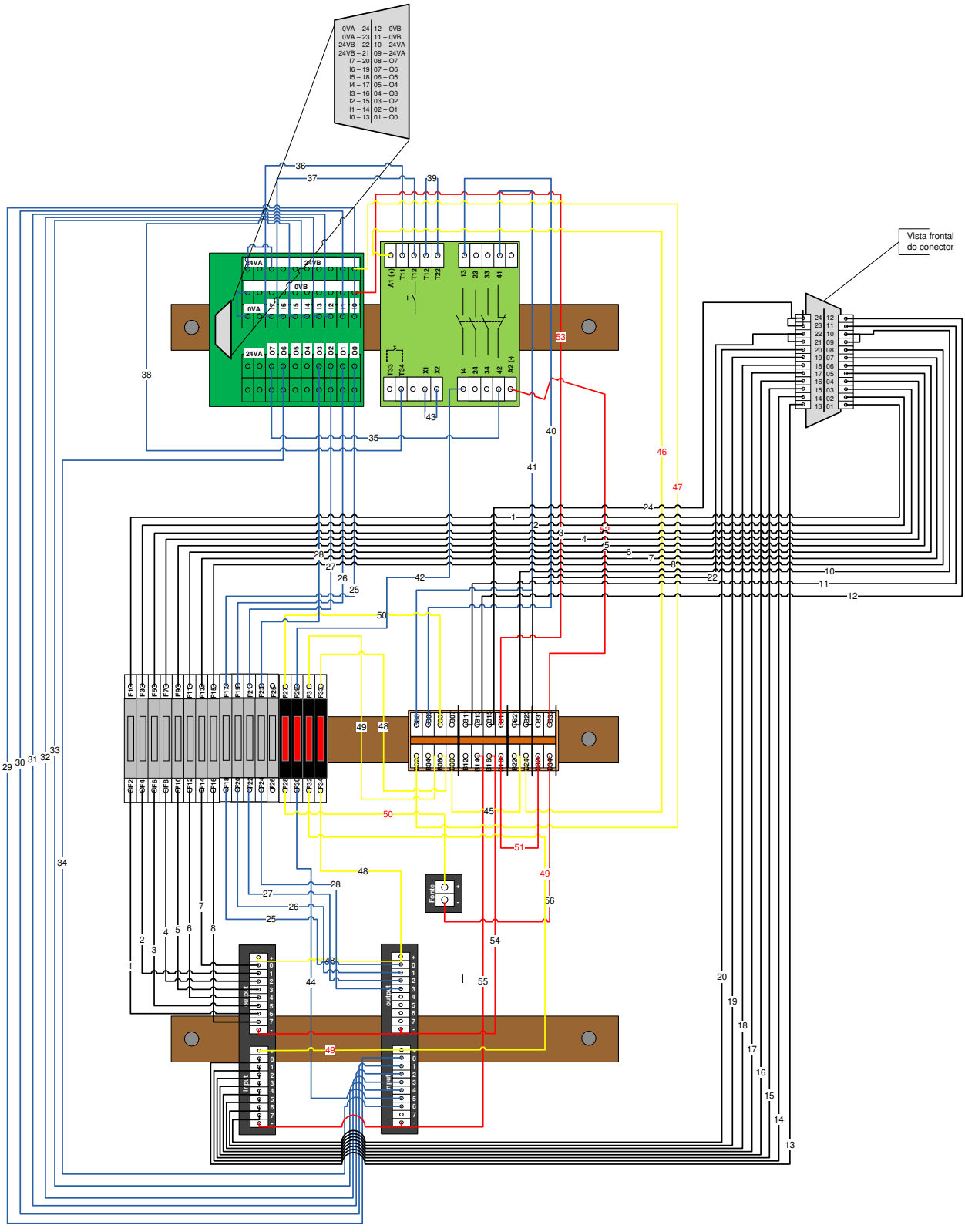
Socrades. website http://www.socrades.eu Acessado em Novembro de 2009.

Taltech. *Understanding Dynamic Data Exchange*. http://www.taltech.com/TALtech_web/support/dde_sw/ddeunder.htm. Acessado em Janeiro de 2009.

W3School. *Introduction to HTML.* Disponível em http://www.w3schools.com/html/html_intro.asp Acessado em Novembro de 2009.

ANEXO A – INSTALAÇÕES ELÉTRICAS

O diagrama elétrico a seguir representa as instalações realizadas no subsistema de alimentação:

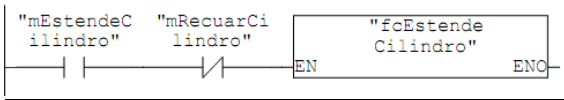


ANEXO B – PROGRAMA EM LADDER DO CONTROLADOR

Programa principal OB1

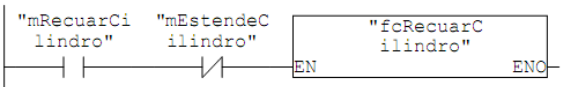
Network: 1 EstendeCilindro

Funcao de estender o cilindro, para tirar a peca corpo do buffer



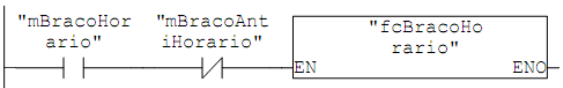
Network: 2 RecuarCilindro

Funcao de recuar cilindro



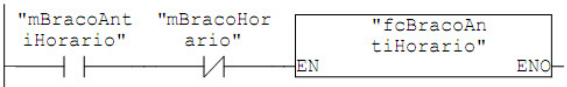
Network: 3 BracoHorario

funcao de movimentar o braco sentido horario (inspecao)



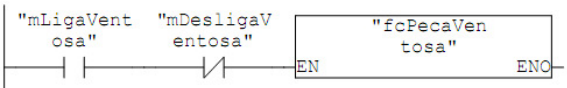
Network: 4 BracoAntiHorario

funcao de movimentar o braco sentido antihorario (alimentacao)



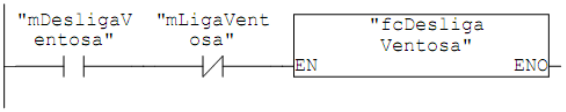
Network: 5 LigaVentosa

Liga a ventosa



Network: 6 Desliga ventosa

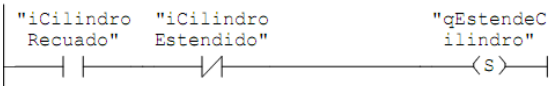
Desliga ventosa



Bloco FC10

Block: FC10

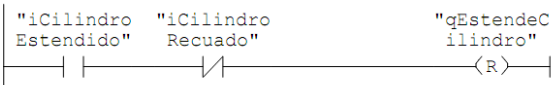
Network: 1 Estende o cilindro de posicionamento de peças



Bloco FC20

Block: FC20

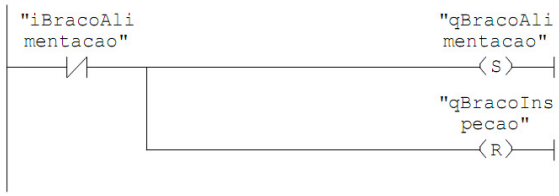
Network: 1 Recua o cilindro de posicionamento de peças



Bloco FC30

Block: FC30

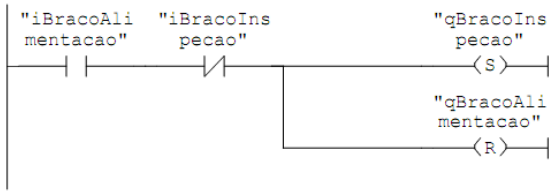
Network: 1 Aciona braço para a posição de alimentação



Bloco FC40

Block: FC40

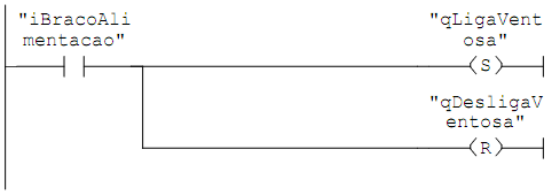
Network: 1 Aciona braço para a posição de inspeção



Bloco FC50

Block: FC50

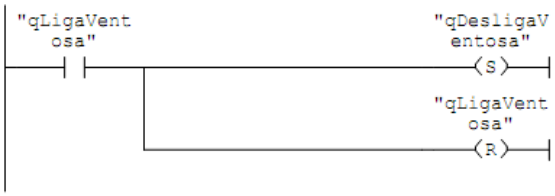
Network: 1 Liga o atuador de vacuo



Bloco FC60

Block: FC60

Network: 1 Desliga atuador de vácuo



ANEXO C – CÓDIGO DO SOFTWARE DE COORDENAÇÃO

Código do WSAlimentacao.asmx

```
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.ComponentModel
Imports System.Web.Script.Services
Imports System.Web
Imports System.Web.Script
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports System.Web.UI.WebControls.WebParts


' To allow this Web Service to be called from script, using ASP.NET
AJAX, uncomment the following line.
<System.Web.Script.Services.ScriptService()> _
<System.Web.Services.WebService(Namespace:="http://tempuri.org/")> _
<System.Web.Services.WebServiceBinding(ConformsTo:=WsiProfiles.BasicPr
ofile1_1)> _
<ToolboxItem(False)> _
Public Class WSAlimentacao
    Inherits System.Web.Services.WebService


    <WebMethod()> _
        Public Function RegistroCliente(ByVal Nome As String) As
Integer
        Dim codCliente As Integer
        codCliente = lnAlimentacao.RegistroCliente(Nome)
        Return codCliente
    End Function

    <WebMethod()> _
        Public Function SolicPedido(ByVal codCliente As Integer, ByVal
npecas As Integer) As Integer
        Dim codPedido As Integer = 0
        codPedido = lnAlimentacao.SolicPedido(codCliente, npecas)
        If codPedido <> 0 Then
            Application.Contents("codPedido") = codPedido
            Application.Contents("codTeleop") =
lnAlimentacao.RetornaTeleop
        End If
        Return codPedido
    End Function

    <WebMethod()> _
    Public Function RetornaCodPedidoTeleop(ByVal codTeleop As Integer)
As Integer
        Dim codPedido
        If codTeleop = Application.Contents("codTeleop") Then
            codPedido = Application.Contents("codPedido")
        Else
            codPedido = 0
            'TELEOPERADOR TEM QUE SABER QUE SE CODPEDIDO = 0 NÃO TEM O
QUE FAZER
        End If
        Return codPedido
    End Function
```

```
    <WebMethod()> _
        Public Function RegistraEstEquip(ByVal estEquip As
Equipamento) As Boolean
        Dim Ok As Boolean
        Ok = lnAlimentacao.RegistraEstEquip(estEquip)
        Return Ok
    End Function

    <WebMethod()> _
        Public Function ConsultaEstEquip() As Object
        Dim Equip As Equipamento
        Equip = lnAlimentacao.ConsultaEstEquip()
        Return Equip
    End Function

    <WebMethod()> _
        Public Function PecaEntregue(ByVal nPecasEntregues As Integer,
ByVal codPedido As Integer) As Boolean
        Dim Ok As Boolean
        Ok = lnAlimentacao.PecaEntregue(nPecasEntregues, codPedido)
        Return Ok
    End Function

    <WebMethod()> _
        Public Function DispAlimentacao() As Boolean
        Dim Ok As Boolean
        Ok = lnAlimentacao.DispAlimentacao()
        Return Ok
    End Function

    <WebMethod()> _
    Public Function ConsultaPedido(ByVal codPedido As Integer) As
Integer
        Dim nPecasEntregues As Integer
        nPecasEntregues = lnAlimentacao.ConsultaPedido(codPedido)
        Return nPecasEntregues
    End Function

    <WebMethod()> _
    Public Function LoginCliente(ByVal codCliente As Integer) As
Boolean
        Dim Ok As Boolean
        Application.Contents("codTeleop") = 0
        Ok = lnAlimentacao.LoginCliente(codCliente)
        If Ok Then
            Application.Contents("codCliente") = codCliente
        End If
        Return Ok
    End Function

    <WebMethod()> _
 Public Function RetornaLoginCliente() As Integer
        Dim codCliente As Integer
        codCliente = Application.Contents("codCliente")
        Return codCliente
    End Function
```

```
End Class
```

Código do WSTeleoperacao.asmx

```
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.ComponentModel
Imports System.Web.Script.Services

' To allow this Web Service to be called from script, using ASP.NET
AJAX, uncomment the following line.
<System.Web.Script.Services.ScriptService()> _
<System.Web.Services.WebService(Namespace:="http://tempuri.org/")> _
<System.Web.Services.WebServiceBinding(ConformsTo:=WsiProfiles.BasicPr
ofile1_1)> _
<ToolboxItem(False)> _
Public Class WSTeleoperacao
    Inherits System.Web.Services.WebService

    <WebMethod()> _
    Public Function RegistroTeleop(ByVal Nome As String) As Integer
        Dim codTeleop As Integer
        codTeleop = lnTeleoperador.RegistroTeleop(Nome)
        Return codTeleop
    End Function

    <WebMethod()> _
    Public Function RetornaLoginTeleop() As Integer
        Dim codTeleop As Integer
        codTeleop = Application.Contents("codTeleop")
        Return codTeleop
    End Function

    <WebMethod()> _
    Public Function Autorizacao(ByVal codTeleop As Integer) As Boolean
        Dim Ok As Boolean
        Ok = lnTeleoperador.Autorizacao(codTeleop)
        Application.Contents("codTeleop") = codTeleop
        Return Ok
    End Function

    <WebMethod()> _
  Public Function ModoOperacao(ByVal codTeleop As Integer, ByVal modo
As String) As Boolean
        Dim ok As Boolean
        ok = lnTeleoperador.modoOperacao(codTeleop, modo)
        Return ok
    End Function

    <WebMethod()> _
Public Function InfEstTelecomando(ByVal codTeleop As Integer, ByVal
codPedido As Integer) As String
        Dim atividade As String
        atividade = lnTelecomando.InfEstadoTelecomando(codTeleop,
codPedido)
        Return atividade
    End Function
```

```
    <WebMethod()> _
Public Function SolicitaTelecomando(ByVal codPedido As Integer, ByVal
Atividade As String) As Boolean
        Dim Ok As Boolean
        Ok = lnTelecomando.SolicTelecomando(codPedido, Atividade)
        Return Ok
    End Function

    <WebMethod()> _
Public Function AtualizaEstTelecomando(ByVal codPedido As Integer,
ByVal codTeleop As Integer) As Boolean
        Dim Ok As Boolean
        Ok = lnTelecomando.AtualizaEstadoTelecomando(codPedido)
        Return Ok
    End Function

    <WebMethod()> _
Public Sub ParadaEmergencia()
        lnTelecomando.ParadaEmergencia()
    End Sub

    <WebMethod()> _
Public Function HelloWorld() As String
        Return "Hello World"
    End Function
End Class
```

Código do lnAlimentacao.vb

```
Imports System.Xml

Public Class lnAlimentacao

    Shared CodigoTeleoperador As Integer = 0
    Shared CodigoPedido As Integer = 0



    Public Shared Function RegistroCliente(ByVal Nome As String) As
Integer
        Dim codCliente As Integer
        Dim bNovoCliente As Boolean
        bNovoCliente = bdAlimentacao.ConsultaCliente(Nome)
        If bNovoCliente = True Then
            bNovoCliente = bdAlimentacao.ResgistraCliente(Nome) 'Não
utilizo este booleano
        End If
        codCliente = bdAlimentacao.RetornaCodCliente(Nome)
        Return codCliente
    End Function

    Public Shared Function LoginCliente(ByVal codCliente As Integer)
As Boolean
        Dim bOk As Boolean
        bOk = bdAlimentacao.ConsultaClientecod(codCliente)
        Return bOk
    End Function

    Public Shared Function DispAlimentacao() As Boolean
```

```
        Dim Ok As Boolean
        ' Dim Resp_XML As XmlDocument
        Dim codTeleop As Integer
        Dim Ok_Equip As Boolean

        codTeleop = lnTeleoperador.DispTeleop()
        Ok_Equip = lnSupervisorio.DispEquip()
        If codTeleop = 0 Then
            Ok = False 'Não tem teleoperador disponível
        Else
            If Ok_Equip = True Then
                Ok = True 'Tem equipamento e teleop ok!
            Else
                Ok = False 'Tem teleop mas não tem equipamento
            End If
        End If
        Return Ok
    End Function


    Public Shared Function SolicPedido(ByVal codCliente As Integer,
ByVal npecas As Integer) As Integer
        'Verifica disponibilidade de Teleoperador
        Dim codTeleop As Integer
        Dim codPedido As Integer
        Dim Ok_Equip As Boolean
        Dim Ok As Boolean

        'O correto pensado inicialmente é este, pensando em um
ambiente distribuido, porém, para teste só terá um teleop
        codTeleop = lnTeleoperador.DispTeleop()
        If codTeleop = 0 Then
            codPedido = 0 'o cliente deve entender que codPedido = 0
significa que
            ' solicitação não foi executada com sucesso, neste caso,
pq não tinha teleoperador
            Return codPedido
        End If

        Ok_Equip = lnSupervisorio.DispEquip
        If Ok_Equip = False Then
            'Não tem equipamento mais disponível, portanto codPedido =
0
            codPedido = 0
            Return codPedido
        End If
        codPedido = bdAlimentacao.ConsultaUltimoPedido()
        codPedido = codPedido + 1       'Atualiza o novo valor para o
codPedido
        Ok = bdAlimentacao.RegistroPedido(codCliente, codPedido,
npecas)
        If Ok = False Then
            codPedido = 0 'Não conseguiu fazer o registro do pedido,
então o cliente precisa saber que deu erro de novo
            Return codPedido
        End If
        Ok_Equip = lnSupervisorio.ProducaoPecas(codPedido, npecas)
        If Ok_Equip = False Then
            'Supervisorio não quis fazer o pedido
            codPedido = 0
            Return codPedido
```

```
        End If
        Ok = lnTelecomando.RegistraTelecomando(codPedido, codTeleop)
        'Se chegou até aqui, é pq o pedido e o tele-operador já estão
confirmados.
        CodigoTeleoperador = codTeleop
        CodigoPedido = codPedido
        Return codPedido
    End Function


    Public Shared Function RetornaTeleop() As Integer
        Dim codTeleop As Integer
        codTeleop = CodigoTeleoperador
        Return codTeleop
    End Function

    Public Shared Function RegistraEstEquip(ByVal EstEqui As
Equipamento) As Boolean
        Dim Ok As Boolean
        Dim DataHora As DateTime
        DataHora = DateTime.Now
        Ok = bdAlimentacao.RegistraEstEquip(EstEqui, DataHora)
        Return Ok
    End Function


    Public Shared Function ConsultaEstEquip() As Object
        Dim Equip As Equipamento
        Equip = bdAlimentacao.ConsultaEstEquip
        Return Equip
    End Function


    Public Shared Function PecaEntregue(ByVal nPecasEntregues As
Integer, ByVal codPedido As Integer) As Boolean
        Dim Ok As Boolean = True
        Dim nPecasTotais As Integer
        Dim codTeleop As Integer
        codTeleop = bdTelecomando.ConsultaTeleop(codPedido)
        nPecasTotais = bdAlimentacao.ConsultaTotalPecas(codPedido)
        bdAlimentacao.AtualizaPecaEntregue(nPecasEntregues, codPedido)
        If nPecasTotais = nPecasEntregues Then
            'Acabou a execução do Pedido!
            Ok = lnTelecomando.AtualizaEstadoTelecomando(codPedido)
            lnTeleoperador.AtualizaStatusTeleop(codTeleop, "online")
        End If
        Return Ok
    End Function

    Public Shared Function ConsultaPedido(ByVal codPedido As Integer)
As Integer
        Dim nPecasEntregues As Integer
        nPecasEntregues =
bdAlimentacao.ConsultaPecasEntregues(codPedido)
        Return nPecasEntregues
    End Function
```

```
End Class
```

Código do lnTeleoperador.vb

```
Public Class lnTeleoperador

    Shared CodigoTeleoperador As Integer = 0

    Public Shared Function RegistroTeleop(ByVal Nome As String) As
Integer
        Dim codTeleop As Integer = 0
        Dim RegistroOk As Boolean = False

        'codTeleop = bdTeleoperador.RetornaCodTeleop(Nome)

        'verifica se esse teleoperador já foi inscrito, se não foi,
codTeleop = 0
        'If codTeleop = 0 Then
        'RegistroOk = bdTeleoperador.RegistroTeleop(Nome)
        'If RegistroOk Then      'se o registro foi feito com sucesso,
o codTeleop agora é dif. de zero
        'codTeleop = bdTeleoperador.RetornaCodTeleop(Nome)
        'End If
        'End If
        RegistroOk = bdTeleoperador.RegistroTeleop(Nome)
        If RegistroOk Then
            codTeleop = bdTeleoperador.RetornaCodTeleop(Nome)
        End If
        Return codTeleop
    End Function

    Public Shared Function Autorizacao(ByVal codTeleop As Integer) As
Boolean
        Dim Ok As Boolean
        Dim status As String
        status = "online"
        Ok = bdTeleoperador.AtualizaStatus(codTeleop, status)
        If Ok = True Then
            CodigoTeleoperador = codTeleop
        End If
        Return Ok
    End Function


    Public Shared Function modoOperacao(ByVal codTeleop As Integer,
ByVal modo As String) As Boolean
        Dim Ok As Boolean
        Ok = bdTeleoperador.Modo(codTeleop, modo)
        Return Ok
    End Function


    Public Shared Function DispTeleop() As Integer
        Dim codTeleop As Integer
        'Correto para Ambiente Distribuido, porém para teste TF só vai
utilizar o teleoperador que estiver logado
        'Dim status As String
        'status = "online"
        'codTeleop = bdTeleoperador.DispTeleop(status)
        codTeleop = CodigoTeleoperador
        Return codTeleop
```

```
    End Function


    Public Shared Function AtualizaStatusTeleop(ByVal codTeleop As
Integer, ByVal status As String) As Boolean
        Dim Ok As Boolean
        Ok = bdTeleoperador.AtualizaStatus(codTeleop, status)
        Return Ok
    End Function


End Class
```

Código do lnTelecomando.vb

```
Public Class lnTelecomando


    Public Shared Function InfEstadoTelecomando(ByVal codTeleop As
Integer, ByVal codPedido As Integer) As String
        Dim telec As Telecomando
        Dim ativ As String
        telec = bdTelecomando.RetornaEstadoTelecomando(codTeleop,
codPedido)
        If telec.Estado = "pendente  " Then
            ativ = telec.Atividade
        Else
            ativ = "atividades concluidas"
        End If
        Return ativ

    End Function


    Public Shared Function AtualizaEstadoTelecomando(ByVal codPedido
As Integer) As Boolean
        Dim Ok As Boolean
        Dim estado As String = "concluido"
        Ok = bdTelecomando.AtualizaEstadoTelecomando(codPedido,
estado)
        lnSupervisorio.RespostaTelecomando(True, codPedido) 'true =
pode fazer atividade
        Return Ok
    End Function

    Public Shared Function SolicTelecomando(ByVal codPedido As
Integer, ByVal Atividade As String) As Boolean
        Dim Ok As Boolean
        Dim codTeleop As Integer
        Dim modo As String
        Dim Estado As String
        codTeleop = bdTelecomando.ConsultaTeleop(codPedido)
        modo = bdTeleoperador.VerificaModoOp(codTeleop)
        If modo.TrimEnd = "monit" Then
            'Simplesmente atualiza a atividade (com estado =
"concluido") e já responde ao Supervisório
            Estado = "concluido"
            Ok = bdTelecomando.RegistraAtividade(codPedido, Atividade,
Estado)
            lnSupervisorio.RespostaTelecomando(Ok, codPedido)
        Else
```

```
            'Se modo é teleoperacao, atividade será registrada com
estado pendente, para que depois o teleoperador possa autoriza-la ou
não
            Estado = "pendente"
            Ok = bdTelecomando.RegistraAtividade(codPedido, Atividade,
Estado)
        End If
        Return Ok
    End Function

    Public Shared Function RegistraTelecomando(ByVal codPedido As
Integer, ByVal codTeleop As Integer) As Boolean
        Dim Ok As Boolean
        Ok = bdTelecomando.RegistraTelecomando(codTeleop, codPedido)
        Return Ok
    End Function

    Public Shared Sub ParadaEmergencia()
        lnSupervisorio.ParadaEmergencia()
    End Sub

End Class
```

Código do bdAlimentacao.vb

```
Imports System.Data
Imports System.Data.SqlClient

Public Class bdAlimentacao

    Public Shared conn As SqlConnection
    Public Shared SQLCmd As New SqlCommand() 'The SQL Command
    Public Shared SQLStr As String


    Public Shared Sub ConectaBD()
        Try
            conn = New SqlConnection("Data
Source=192.168.0.197;Initial Catalog=TF;User ID=tf;Password=samira;")
            conn.Open()
            SQLCmd.Connection = conn
        Catch ex As Exception
            'MsgBox("Erro ao realizar a conexao com Banco de Dados")
        End Try
    End Sub


    Public Shared Sub DesconectaBD()
        conn.Close()
    End Sub

    Public Shared Function ConsultaCliente(ByVal Nome As String) As
Boolean
        Dim Ok As Boolean
        Dim codCliente As Integer
        ConectaBD()
        'http://msdn.microsoft.com/pt-
br/library/system.data.sqlclient.sqlcommand.aspx

        SQLStr = "SELECT codCliente FROM Cliente WHERE Nome ='" & Nome
& "'"
```

```
        SQLCmd.CommandText = SQLStr
        codCliente = CType(SQLCmd.ExecuteScalar, Integer)
        If codCliente = 0 Then
            Ok = True 'Cliente ainda não registrado
        Else : Ok = False   'Cliente registrado
        End If
        DesconectaBD()
        Return Ok
    End Function

    Public Shared Function ConsultaClientecod(ByVal cod As Integer) As
Boolean
        Dim Ok As Boolean = False
        Dim Nome As String = Nothing
        ConectaBD()
        'http://msdn.microsoft.com/pt-
br/library/system.data.sqlclient.sqlcommand.aspx

        SQLStr = "SELECT Nome FROM Cliente WHERE codCliente ='" & cod
& "'"
        SQLCmd.CommandText = SQLStr
        Nome = CType(SQLCmd.ExecuteScalar, String)
        If Nome = "" Then
            Ok = False 'Cliente ainda não registrado
        Else : Ok = True  'Cliente registrado
        End If
        DesconectaBD()
        Return Ok
    End Function

    Public Shared Function ResgistraCliente(ByVal Nome As String) As
Boolean
        Dim Ok As Boolean
        ConectaBD()
        SQLStr = "INSERT INTO Cliente (Nome) VALUES ('" & Nome & "')"
        SQLCmd.CommandText = SQLStr
        Try
            SQLCmd.ExecuteNonQuery()
            Ok = True
        Catch
            Ok = False
        End Try
        DesconectaBD()
        Return Ok
    End Function

    Public Shared Function RetornaCodCliente(ByVal Nome As String) As
Integer
        Dim codCliente As Integer
        ConectaBD()
        SQLStr = "SELECT codCliente FROM Cliente WHERE Nome ='" & Nome
& "'"
        SQLCmd.CommandText = SQLStr
        codCliente = CType(SQLCmd.ExecuteScalar, Integer)
        DesconectaBD()
        Return codCliente
    End Function

    Public Shared Function RegistroPedido(ByVal codCliente As Integer,
ByVal codPedido As Integer, ByVal npecas As Integer) As Boolean
        Dim Ok As Boolean
```

```
        Try
            ConectaBD()
            SQLStr = "INSERT INTO Pedido (codPedido, npecas,
codCliente) VALUES ( '" & codPedido & "','" & npecas & "','" &
codCliente & "')"
            SQLCmd.CommandText = SQLStr
            SQLCmd.ExecuteNonQuery()
            DesconectaBD()
            Ok = True
        Catch ex As Exception
            Ok = False
        End Try
        Return Ok
    End Function

    Public Shared Function RegistraEstEquip(ByVal EstEquip As
Equipamento, ByVal DataHora As DateTime) As Boolean
        Dim Ok As Boolean
        ConectaBD()
        SQLStr = "INSERT INTO Equipamento (DataHora, R000, R001, R002,
R003, R005, R006) VALUES ('" & DataHora & "','" & EstEquip.R000 &
"','" & EstEquip.R001 & "','" & EstEquip.R002 & "','" & EstEquip.R003
& "','" & EstEquip.R005 & "','" & EstEquip.R006 & "')"
        SQLCmd.CommandText = SQLStr
        Try
            SQLCmd.ExecuteNonQuery()
            Ok = True
        Catch ex As Exception
            Ok = False
        End Try
        DesconectaBD()
        Return Ok
    End Function

    Public Shared Function ConsultaEstEquip() As Object
        Dim EstEquip As Equipamento
        Dim reader As SqlDataReader
        Dim dataho As DateTime
        Dim Indice As Integer
        Dim r0 As Integer
        Dim r1 As Integer
        Dim r2 As Integer
        Dim r3 As Integer
        Dim r5 As Integer
        Dim r6 As Integer
        ConectaBD()
        SQLStr = "SELECT MAX(Indice)FROM Equipamento"
        SQLCmd.CommandText = SQLStr
        Indice = CType(SQLCmd.ExecuteScalar, Integer)
        SQLStr = "SELECT DataHora, R000, R001, R002, R003, R005, R006
FROM Equipamento WHERE Indice =" & Indice
        SQLCmd.CommandText = SQLStr
        reader = SQLCmd.ExecuteReader
        While reader.Read()
            dataho = reader.GetDateTime(0)
            r0 = CType(reader.GetValue(1), Integer)
            r1 = CType(reader.GetValue(2), Integer)
            r2 = CType(reader.GetValue(3), Integer)
            r3 = CType(reader.GetValue(4), Integer)
            r5 = CType(reader.GetValue(5), Integer)
            r6 = CType(reader.GetValue(6), Integer)
```

```
        End While
        EstEquip.DataHora = dataho
        EstEquip.R000 = r0
        EstEquip.R001 = r1
        EstEquip.R002 = r2
        EstEquip.R003 = r3
        EstEquip.R005 = r5
        EstEquip.R006 = r6
        DesconectaBD()
        Return EstEquip
    End Function

    Public Shared Function ConsultaTotalPecas(ByVal codPedido As
Integer) As Integer
        Dim nPecas As Integer
        ConectaBD()
        SQLStr = "SELECT npecas FROM Pedido WHERE codPedido ='" &
codPedido & "'"
        SQLCmd.CommandText = SQLStr
        nPecas = CType(SQLCmd.ExecuteScalar, Integer)
        DesconectaBD()
        Return nPecas
    End Function

    Public Shared Function ConsultaPecasEntregues(ByVal codPedido As
Integer) As Integer
        Dim nPecasEntregues As Integer
        ConectaBD()
        SQLStr = "SELECT npecasentregues FROM Pedido WHERE
codPedido='" & codPedido & "'"
        SQLCmd.CommandText = SQLStr
        nPecasEntregues = CType(SQLCmd.ExecuteScalar, Integer)
        DesconectaBD()
        Return nPecasEntregues
    End Function

    Public Shared Sub AtualizaPecaEntregue(ByVal nPecasEntregues As
Integer, ByVal codPedido As Integer)
        ConectaBD()
        SQLStr = "UPDATE Pedido SET npecasentregues =" &
nPecasEntregues & " WHERE codPedido=" & codPedido
        SQLCmd.CommandText = SQLStr
        SQLCmd.ExecuteNonQuery()
        DesconectaBD()
    End Sub

    Public Shared Function ConsultaUltimoPedido() As Integer
        Dim codPedido As Integer
        ConectaBD()
        SQLStr = "SELECT MAX(codPedido) FROM Pedido"
        SQLCmd.CommandText = SQLStr
        codPedido = CType(SQLCmd.ExecuteScalar, Integer)
        DesconectaBD()
        Return codPedido
    End Function


End Class
```

Código do bdTeleoperador.vb

```
Imports System.Data
Imports System.Data.SqlClient

Public Class bdTelecomando

    Public Shared conn As SqlConnection
    Public Shared SQLCmd As New SqlCommand() 'The SQL Command
    Public Shared SQLStr As String


    Public Shared Sub ConectaBD()
        Try
            conn = New SqlConnection("Data
Source=192.168.0.197;Initial Catalog=TF;User ID=tf;Password=samira;")
            conn.Open()
            SQLCmd.Connection = conn
        Catch ex As Exception
            '  MsgBox("Erro ao realizar a conexao com Banco de Dados")
        End Try
    End Sub

    Public Shared Sub DesconectaBD()
        conn.Close()
    End Sub

    Public Shared Function RetornaEstadoTelecomando(ByVal codTeleop As
Integer, ByVal codPedido As Integer) As Object
        'http://msdn.microsoft.com/pt-br/library/9kcbe65k.aspx
        Dim telec As Telecomando
        Dim ativ As String = ""
        Dim est As String = ""
        Dim datahora As DateTime
        Dim reader As SqlDataReader

        ConectaBD()
        SQLStr = "SELECT Atividade, Estado FROM Telecomando WHERE
codPedido ='" & codPedido & "'"
        SQLCmd.CommandText = SQLStr
        reader = SQLCmd.ExecuteReader
        While reader.Read()
            ativ = reader.GetString(0)
            est = reader.GetString(1)
            'datahora = reader.GetDateTime(2)
        End While
        'telec = New Telecomando(codTeleop, codPedido, ativ, est,
datahora)
        telec.Atividade = ativ
        telec.codPedido = codPedido
        telec.codTeleop = codTeleop
        telec.DataHora = datahora
        telec.Estado = est
        DesconectaBD()
        Return telec
    End Function

    Public Shared Function RegistraTelecomando(ByVal codTeleop As
Integer, ByVal codPedido As Integer) As Boolean
        Dim Ok As Boolean
        ConectaBD()
        SQLStr = "INSERT INTO Telecomando (codTeleop, codPedido)
VALUES ('" & codTeleop & "','" & codPedido & "')"
```

```
        SQLCmd.CommandText = SQLStr
        Try
            SQLCmd.ExecuteNonQuery()
            Ok = True
        Catch
            Ok = False
        End Try
        DesconectaBD()
        Return Ok
    End Function


    Public Shared Function RegistraAtividade(ByVal codPedido As
Integer, ByVal Atividade As String, ByVal Estado As String) As Boolean
        Dim Ok As Boolean
        ConectaBD()
        SQLStr = "UPDATE Telecomando SET Atividade='" & Atividade &
"', Estado='" & Estado & "' WHERE codPedido ='" & codPedido & "'"
        SQLCmd.CommandText = SQLStr
        Try
            SQLCmd.ExecuteNonQuery()
            Ok = True
        Catch
            Ok = False
        End Try
        DesconectaBD()
        Return Ok
    End Function

    Public Shared Function AtualizaEstadoTelecomando(ByVal codPedido
As Integer, ByVal Estado As String) As Boolean
        Dim Ok As Boolean

        ConectaBD()
        SQLStr = "UPDATE Telecomando SET Estado ='" & Estado & "'
WHERE codPedido ='" & codPedido & "'"
        SQLCmd.CommandText = SQLStr
        Try
            SQLCmd.ExecuteNonQuery()
            Ok = True
        Catch
            Ok = False
        End Try
        DesconectaBD()
        Return Ok
    End Function


    Public Shared Function ConsultaTeleop(ByVal codPedido As Integer)
As Integer
        Dim codTeleop As Integer
        ConectaBD()
        SQLStr = "SELECT codTeleop FROM Telecomando WHERE codPedido
='" & codPedido & "'"
        SQLCmd.CommandText = SQLStr
        codTeleop = CType(SQLCmd.ExecuteScalar, Integer)
        DesconectaBD()
        Return codTeleop
    End Function
```

```
End Class
```

Código do bdTelecomando.vb

```
Imports System.Data
Imports System.Data.SqlClient

Public Class bdTeleoperador

    Public Shared conn As SqlConnection
    Public Shared SQLCmd As New SqlCommand() 'The SQL Command
    Public Shared SQLStr As String


    Public Shared Sub ConectaBD()
        Try
            conn = New SqlConnection("Data
Source=192.168.0.197;Initial Catalog=TF;User ID=tf;Password=samira;")
            conn.Open()
            SQLCmd.Connection = conn
        Catch ex As Exception
            'MsgBox("Erro ao realizar a conexao com Banco de Dados")
        End Try
    End Sub

    Public Shared Sub DesconectaBD()
        conn.Close()
    End Sub


    Public Shared Function RegistroTeleop(ByVal Nome As String) As
Boolean
        Dim Ok As Boolean
        ConectaBD()
        SQLStr = "INSERT INTO Teleoperador (Nome) VALUES ('" & Nome &
"')"
        SQLCmd.CommandText = SQLStr
        Try
            SQLCmd.ExecuteNonQuery()
            Ok = True
        Catch
            Ok = False
        End Try
        DesconectaBD()
        Return Ok
    End Function

    Public Shared Function RetornaCodTeleop(ByVal Nome As String) As
Integer
        Dim codTeleop As Integer
        ConectaBD()
        SQLStr = "SELECT codTeleop FROM Teleoperador WHERE Nome='" &
Nome & "'"
        SQLCmd.CommandText = SQLStr
        codTeleop = CType(SQLCmd.ExecuteScalar, Integer)
        DesconectaBD()
        Return codTeleop
    End Function

    Public Shared Function AtualizaStatus(ByVal codTeleop As Integer,
ByVal status As String) As Boolean
```

```
        Dim Ok As Boolean
        ConectaBD()
        SQLStr = "UPDATE Teleoperador SET status='" & status & "'
WHERE codTeleop='" & codTeleop & "'"
        SQLCmd.CommandText = SQLStr
        Try
            SQLCmd.ExecuteNonQuery()
            Ok = True
            DesconectaBD()
        Catch
            Ok = False
        End Try
        DesconectaBD()
        Return Ok
    End Function


    Public Shared Function Modo(ByVal codTeleop As Integer, ByVal
modoTeleop As String) As Boolean
        Dim Ok As Boolean = True
        ConectaBD()
        SQLStr = "UPDATE Teleoperador SET modoop='" & modoTeleop & "'
WHERE codTeleop='" & codTeleop & "'"
        SQLCmd.CommandText = SQLStr
        Try
            SQLCmd.ExecuteNonQuery()
            Ok = True
        Catch
            Ok = False
        End Try
        DesconectaBD()
        Return Ok
    End Function



    Public Shared Function DispTeleop(ByVal status As String) As
Integer
        Dim codTeleop As Integer
        ConectaBD()
        SQLStr = "SELECT codTeleop FROM Teleoperador WHERE status='" &
status & "'"
        SQLCmd.CommandText = SQLStr
        codTeleop = CType(SQLCmd.ExecuteScalar(), Integer)
        DesconectaBD()
        Return codTeleop
    End Function


    Public Shared Function VerificaModoOp(ByVal codTeleop As Integer)
As String
        Dim modoop As String
        ConectaBD()
        SQLStr = "SELECT modoop FROM Teleoperador WHERE codTeleop='" &
codTeleop & "'"
        SQLCmd.CommandText = SQLStr
        modoop = CType(SQLCmd.ExecuteScalar(), String)
        DesconectaBD()
        Return modoop
    End Function
```

```
End Class
```

Código do Equipamento.vb

```
Public Structure Equipamento
    Dim R000 As Integer
    Dim R001 As Integer
    Dim R002 As Integer
    Dim R003 As Integer
    Dim R005 As Integer
    Dim R006 As Integer
    Dim DataHora As DateTime
End Structure
```

Código do Telecomando.vb

```
Public Structure Telecomando
    Dim codTeleop As Integer
    Dim codPedido As Integer
    Dim Atividade As String
    Dim Estado As String
    Dim DataHora As DateTime
End Structure
```

ANEXO D – CÓDIGO DO SUPERVISÓRIO

Código do Sup.asmx

```
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.ComponentModel
Imports System.Web.Script.Services
Imports System.Web
Imports System.Web.Script
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports System.Web.UI.WebControls.WebParts
Imports System.Xml
Imports opcconn

' To allow this Web Service to be called from script, using ASP.NET
AJAX, uncomment the following line.
<System.Web.Script.Services.ScriptService()> _
<System.Web.Services.WebService(Namespace:="http://tempuri.org/")> _
<System.Web.Services.WebServiceBinding(ConformsTo:=WsiProfiles.BasicPr
ofile1_1)> _
<ToolboxItem(False)> _
Public  Class Sup
    Inherits System.Web.Services.WebService

    <WebMethod()> _
Public Function ConectaOPC() As Boolean
        Dim ok As Boolean
        ok = lnSupervisorio.ConectaOPC()
        Return ok
    End Function

    <WebMethod()> _
Public Sub CarregaXml()
        lnSupervisorio.CarregaXml()
    End Sub

    <WebMethod()> _
    Public Function DispEquip() As Boolean
        Dim Ok_Equip As Boolean = True
        lnSupervisorio.DispEquip()
        Return Ok_Equip
    End Function

    <WebMethod()> _
Public Function ProducaoPecas(ByVal codPed As Integer, ByVal npecas As
Integer) As Boolean
        Dim Ok As Boolean = True
        lnSupervisorio.ProducaoPecas(codPed, npecas)
        Return Ok
    End Function

    <WebMethod()> _
Public Sub RespostaTelecomando(ByVal PodeFazer As Boolean, ByVal
codPed As Integer)
        'PodeFazer = true - pode executar atividade
        'PodeFazer = false - não pode executar atividade
        lnSupervisorio.RespostaTelecomando(PodeFazer, codPed)
    End Sub
```

```
    <WebMethod()> _
Public Sub ParadaEmergencia()
        lnSupervisorio.ParadaEmergencia()
    End Sub

    <WebMethod()> _
Public Sub Acorda()
        lnSupervisorio.Acorda()
    End Sub

    'FAZER MÉTODOS PARA TESTES BÁSICOS


    <WebMethod()> _
    Public Sub Setup()
        lnSupervisorio.Setup()
    End Sub

    <WebMethod()> _
    Private Sub Execucao()
        lnSupervisorio.Execucao()
    End Sub

    'Métodos utilizados apenas para testes

    <WebMethod()> _
Public Sub EstendeCilindro()
        lnSupervisorio.EstendeCilindro()
    End Sub

    <WebMethod()> _
Public Sub BracoAlimentacao()
        lnSupervisorio.BracoAlimentacao()
    End Sub

    <WebMethod()> _
    Public Sub LigaVentosa()
        lnSupervisorio.LigaVentosa()
    End Sub

    <WebMethod()> _
Public Sub BracoInspecao()
        lnSupervisorio.BracoInspecao()
    End Sub

    <WebMethod()> _
Public Sub DesligaVentosa()
        lnSupervisorio.DesligaVentosa()
    End Sub

    <WebMethod()> _
Public Sub RecuaCilindro()
        lnSupervisorio.RecuaCilindro()
    End Sub

    <WebMethod()> _
Public Function lerPorta(ByVal porta As String) As String
        Dim estado_port As String
        estado_port = lnSupervisorio.lerPorta(porta)
```

```
        Return estado_port
    End Function

End Class
```

Código do lnSupervisório.vb

```
Imports System.ComponentModel
Imports System.Xml
Imports opcconn

Public Class lnSupervisorio

    Shared xmlDoc As XmlDocument = New XmlDocument
    Shared opc As opcClient
    Shared group As opcClientGroup
    Shared codPedido As Integer = 0
    Shared bEmProducao As Boolean = False
    Shared nTotalPecas As Integer = 0
    Shared nPecasEntregues As Integer = 0
    Shared Atividade As String = ""



    Public Shared Function ConectaOPC() As Boolean
        Dim Ok As Boolean = True
        Dim xmlItem As XmlNode
        Dim xmlServer As XmlNode
        Dim xmlGroup As XmlNode
        xmlServer = xmlDoc.GetElementsByTagName("server").ItemOf(0)
        'Cria um novo objeto opcclient
        opc = New
opcClient(xmlServer.Attributes.GetNamedItem("name").Value)
        If opc.ErrorMessage.Length > 0 Then
            'Se mensagem de erro tem tamanho maior que zero é porque
ocorreu erro
            opc.Dispose()
            'Não foi possível criar o objeto OPC
            Ok = False
        Else
            'cria grupo
            For Each xmlGroup In xmlServer.SelectNodes("group")

opc.AddGroup(xmlGroup.Attributes.GetNamedItem("name").Value)
                If opc.ErrorMessage.Length = 0 Then
                    'cria item para cada grupo
                    For Each xmlItem In xmlGroup.SelectNodes("item")

opc.GetGroupByPosition(0).AddItem(xmlItem.Attributes.GetNamedItem("id"
).Value, xmlItem.Attributes.GetNamedItem("memory").Value)
                        If
opc.GetGroupByPosition(0).ErrorMessage.Length > 0 Then
                            'não foi possível criar ao menos um grupo
                            Ok = False
                        End If
                    Next
                Else
                    'não foi possível criar grupo
```

```
                    Ok = False
                End If
            Next
            '*************
            'MsgBox("Conexão OPC Montada")
        End If
        'Só após a conexão OPC ter sido efetuada que são inicializadas
as variaveis
        'Inicializa as variaveis
        Atividade = ""
        bEmProducao = False
        codPedido = 0
        nTotalPecas = 0
        nPecasEntregues = 0
        Return Ok
    End Function

    Public Shared Sub CarregaXml()
        ' MsgBox("Carregando XML...")

xmlDoc.Load("http://192.168.0.197/alimentacao/xml/opcconfig.xml")
        ' MsgBox("Xml Carregado!")

    End Sub

    Public Shared Function DispEquip() As Boolean
        Dim Ok_Equip As Boolean = True

        If bEmProducao = True Then
            Ok_Equip = False
        Else
            If lerPorta("R006") = "1" Then
                'Não tem peça no Buffer
                Ok_Equip = False
            Else
                'Situação em que tem peças no buffer, e não está em
produçao
                'Refaz o setup novamente (coloca sistema no estado
inicial)
                Setup()
            End If
        End If

        Return Ok_Equip
    End Function

    Public Shared Function ProducaoPecas(ByVal codPed As Integer,
ByVal npecas As Integer) As Boolean
        Dim Ok As Boolean = False

        If bEmProducao = False Then
            bEmProducao = True
            codPedido = codPed
            nTotalPecas = npecas
            nPecasEntregues = 0
            Setup()
            Ok = True
        Else
            Ok = False
        End If
```

```
        Return Ok
    End Function

    Public Shared Sub RespostaTelecomando(ByVal PodeFazer As Boolean,
ByVal codPed As Integer)
        'PodeFazer = true - pode executar atividade
        'PodeFazer = false - não pode executar atividade

        If codPedido = codPed Then
            'Telecomando Correto
            If PodeFazer = True Then
                'Se telecomando está correto e pode executar a
atividade
                Execucao()
            End If
        End If

    End Sub

    Public Shared Sub ParadaEmergencia()

        group = opc.GetGroupByPosition(0)
        ' Desativa todos os acionamentos
        ' Verificar se está correto
        group.GetItemById("W004").Write("0")
        group.GetItemById("W005").Write("0")
        group.GetItemById("W000").Write("0")
        group.GetItemById("W001").Write("0")
        group.GetItemById("W002").Write("0")
        group.GetItemById("W003").Write("0")

    End Sub

    Public Shared Sub Acorda()
        'Este é o Main!! que será chamado de tempos em tempos pela
tela
        Dim ok As Boolean

        If codPedido <> 0 Then
            'Isto é, tem pedido
            If nPecasEntregues = nTotalPecas Then
                Setup()
                codPedido = 0
                nTotalPecas = 0
                nPecasEntregues = 0
                bEmProducao = False
                'Atividade = 0
                Atividade = "braco_inspecao_0"
            Else
                'Ainda precisa executar pedido
                ok = lnTelecomando.SolicTelecomando(codPedido,
Atividade)
            End If
        Else
            'Se as variaveis globais não funcionarem, eu uso o
Application! e transformo todas em applications!

            Setup()
            codPedido = 0
            nTotalPecas = 0
            nPecasEntregues = 0
```

```
            bEmProducao = False
            'Atividade = 0
            Atividade = "braco_inspecao_0"

        End If

    End Sub

    'FAZER MÉTODOS PARA TESTES BÁSICOS

    Public Shared Function lerPorta(ByVal var As String) As String
        Dim Variavel As String
        group = opc.GetGroupByPosition(0)
        Variavel = group.GetItemById(var).Read()
        Return Variavel
    End Function

    Public Shared Sub Setup()
        'If Application.Contents("ConexaoOPC") = Nothing Then
        '    MsgBox("Conexão com o OPC Server não encontrada!")
        'Else
        group = opc.GetGroupByPosition(0)
        ' Desliga ventosa
        group.GetItemById("W004").Write("0")
        group.GetItemById("W005").Write("1")
        'Recua cilindro do buffer
        group.GetItemById("W000").Write("0")
        group.GetItemById("W001").Write("1")
        'Gira braço para ficar no sub-sistema de alimentacao
        group.GetItemById("W002").Write("0")
        group.GetItemById("W003").Write("1")
        'End If
    End Sub

    Public Shared Sub Execucao()
        'variável utilizada para que a rotina de execução seja
finalizada antes que ocorra
        'a atualizaçao das variaveis pelo método Acorda()
        Dim npecasentregadas As Integer
        group = opc.GetGroupByPosition(0)
        'Select Case Atividade
        Select Case Atividade

            Case "braco_inspecao_0"
                'Leva o braço para posição de inspeção
                group.GetItemById("W002").Write("1")
                group.GetItemById("W003").Write("0")
                System.Threading.Thread.Sleep(5000)
                Atividade = "estende_cilindro"

            Case "estende_cilindro"
                group.GetItemById("W000").Write("1")
                group.GetItemById("W001").Write("0")
                System.Threading.Thread.Sleep(2000)
                'Atualiza qual é a próxima atividade que deve ser
feita
                'Atividade = "braco_alimentacao"
                Atividade = "braco_alimentacao"

            Case "braco_alimentacao"
                group.GetItemById("W002").Write("0")
```

```
                group.GetItemById("W003").Write("1")
                System.Threading.Thread.Sleep(2000)
                'Atualiza qual é a próxima atividade que deve ser
feita
                'Atividade = "liga_ventosa"
                Atividade = "liga_ventosa"

            Case "liga_ventosa"
                group.GetItemById("W004").Write("1")
                group.GetItemById("W005").Write("0")
                System.Threading.Thread.Sleep(2000)
                If lerPorta("R005") = 0 Then
                    'Não pegou a peça
                    'MsgBox("Peça não foi presa!")
                End If
                System.Threading.Thread.Sleep(2000)
                'Atualiza qual é a próxima atividade que deve ser
feita
                'Atividade = "recua_cilindro"
                Atividade = "recua_cilindro"

            Case "recua_cilindro"
                group.GetItemById("W000").Write("0")
                group.GetItemById("W001").Write("1")
                System.Threading.Thread.Sleep(2000)
                'Atualiza qual é a próxima atividade que deve ser
feita
                'Atividade = "braco_inspecao"
                Atividade = "braco_inspecao"

            Case "braco_inspecao"
                'Leva o braço para posição de inspeção
                group.GetItemById("W002").Write("1")
                group.GetItemById("W003").Write("0")
                System.Threading.Thread.Sleep(5000)
                'atualiza o numero de peças entregues
                npecasentregadas = nPecasEntregues + 1
                ' Desliga ventosa
                group.GetItemById("W004").Write("0")
                group.GetItemById("W005").Write("1")
                'Seta condições iniciais do subsistema de alimentacao
                Setup()
                'Atualiza qual é a próxima atividade que deve ser
feita
                'Atividade = "estende_cilindro"
                If npecasentregadas = nTotalPecas Then
                    Atividade = "Pedido Finalizado!"
                Else
                    Atividade = "braco_inspecao_0"
                    nPecasEntregues = npecasentregadas
                End If
                'Atualiza quantas peças foram entregues no Banco de
Dados
                lnAlimentacao.PecaEntregue(npecasentregadas,
codPedido)
        End Select
        'End If
    End Sub

    'Métodos utilizados apenas para testes
```

```
    Public Shared Sub EstendeCilindro()
        group = opc.GetGroupByPosition(0)
        group.GetItemById("W000").Write("1")
        group.GetItemById("W001").Write("0")
        System.Threading.Thread.Sleep(1000)
    End Sub

    Public Shared Sub BracoAlimentacao()
        group = opc.GetGroupByPosition(0)
        group.GetItemById("W002").Write("0")
        group.GetItemById("W003").Write("1")
        System.Threading.Thread.Sleep(1000)
    End Sub

    Public Shared Sub LigaVentosa()
        group = opc.GetGroupByPosition(0)
        group.GetItemById("W004").Write("1")
        group.GetItemById("W005").Write("0")
        System.Threading.Thread.Sleep(1000)

    End Sub

    Public Shared Sub BracoInspecao()
        group = opc.GetGroupByPosition(0)
        group.GetItemById("W002").Write("1")
        group.GetItemById("W003").Write("0")
        System.Threading.Thread.Sleep(1000)
    End Sub

    Public Shared Sub DesligaVentosa()
        group = opc.GetGroupByPosition(0)
        group.GetItemById("W004").Write("0")
        group.GetItemById("W005").Write("1")
        System.Threading.Thread.Sleep(1000)
    End Sub

    Public Shared Sub RecuaCilindro()

        group = opc.GetGroupByPosition(0)
        group.GetItemById("W000").Write("0")
        group.GetItemById("W001").Write("1")
        System.Threading.Thread.Sleep(1000)
    End Sub


End Class
```

ANEXO E – PÁGINAS WEB DE ACESSO

Código da página de acesso principal (Default.aspx)

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="Default.aspx.vb" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head  runat="server">
<meta content="pt-br" http-equiv="Content-Language" />
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>Subsistema de Alimentação</title>

<style type="text/css">
h1 {
      font-family: Calibri;
      font-size: x-large;
      color: #0000FF;
      font-variant: normal;
      text-transform: none;
}
textarea {
      font-family: Calibri;
      font-size: medium;
}
.texto {
      font-family: calibri;
      font-size: medium;
}
.style1 {
      text-align: center;
}
.div {
      margin-right: auto;
      margin-left: auto;
}
.style2 {
      border-width: 0px;
}
</style>
<meta content="monitoração, teleoperação, sistema distribuído e
colaborativo" name="keywords" />
<meta content="Acesso ao software de Coordenação do subsistema de
alimentação" name="description" />
</head>

<body style="margin-left: 0; margin-top: 0; background-color:
#FFFFFF">

<div id="titulo" style="position: absolute; width: 800px; height:
50px; z-index: 3">
      <img alt="titulo" class="div" height="50"
src="images/titulo.png" width="800" /></div>
<div id="cliente" style="position: absolute; width: 400px; height:
51px; z-index: 1; left: 0px; top: 160px">
      <a href="cliente.aspx">
```

```
      <img alt="cliente" class="style2" height="51"
src="images/Cliente.png" width="400" /></a></div>
<div id="image" style="position: absolute; width: 513px; height:
100px; z-index: 2; left: 0px; top: 55px">
      <img alt="PMRLSA" class="div" height="105"
src="images/CabecalhoPmrLsa.png" width="800" /></div>
<div id="texto" class="div" style="position: absolute; width: 440px;
height: 273px; z-index: 4; left: 210px; top: 259px">
      <h1 class="style1">Subsistema de Alimentação</h1>
      <br />
      <br />
      <span class="texto">Este subsistema é responsável pelo
fornecimento de
      peças.<br />
      <br />
      Selecione &quot;<strong>Cliente</strong>&quot; se deseja
solicitar peças, ou
      acompanhar seu pedido.<br />
      <br />
      Selecione &quot;<strong>Teleoperador</strong>&quot; para operar
o subsistema. </span>
</div>
<div id="teleop" style="position: absolute; width: 401px; height:
51px; z-index: 5; left: 400px; top: 160px">
      <a href="teleop.aspx">
      <img alt="teleop" class="style2" height="51"
src="images/Teleop.png" width="401" /></a></div>

</body>

</html>
```

Código da página de login do cliente (cliente.aspx)

```
<%@ Page Language="vb" AutoEventWireup="true"
CodeBehind="cliente.aspx.vb" Inherits="Coordenador.cliente" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml">


<head runat="server">
<title>Cliente</title>

<script language="javascript" type="text/javascript" src="cliente.js"
>
</script>

<style type="text/css">
.div {
      margin-right: auto;
      margin-left: auto;
}
.centro_img {
      text-align: center;
      border-style: solid;
      border-width: thin;
}
```

```
.centralimg {
      margin-top: 20px;
}
.botao {
      font-family: Calibri;
      font-style: normal;
      font-weight: bold;
}
</style>

</head>

<body>

 <div>
     <form  id="form1" runat="server"  style="background-repeat:no-
repeat; width:900px; height:700px;" >
         <asp:ScriptManager ID="ScriptManager1" runat="server" >
              <Scripts>
                  <asp:ScriptReference Path="cliente.js" />
              </Scripts>
              <Services>
                  <asp:ServiceReference Path="WSAlimentacao.asmx"
InlineScript="false" />
              </Services>
         </asp:ScriptManager>

        <div id="titulo" style="position: absolute; width: 800px;
height: 50px; z-index: 3; left: 0">
              <img alt="titulo" class="div" height="50"
src="images/titulo.png" width="800" /></div>

        <div id="image" style="position: absolute; width: 513px;
height: 100px; z-index: 1; left: 0px; top: 69px">
              <img alt="PMRLSA" class="div" height="105"
src="images/CabecalhoPmrLsa.png" width="800" /></div>

        <div id="layer1" style="position: absolute; width: 822px;
height: 68px; z-index: 4; left: 0; top: 172px">
              <img alt="tit_cliente" height="68"
src="images/tit_cliente.png" width="822" /></div>

        <div id="cadastro" class="centro_img" style="position:
absolute; width: 400px; height: 250px; z-index: 5; left: 0px; top:
245px">
              <label id="Label1"></label>
              <img alt="cadastro" class="centralimg" height="67"
src="images/cadastro.png" width="217" /><br />
              <br />
              <strong> </strong>
              <strong>
                 <span class="botao">Nome  </span>
                  <span class="centralimg"> </span>
                </strong>
                    <input class="centralimg" id="txtNome"
name="txtNome" style="width: 216px"
                 type="text" />

                    <input class="botao" name="btnCadastro"
type="button" value="cadastro"
                 onclick="return cadastro()" id="btnCadastro"/>
```

```
                <br />
                    <span class="botao">cod Cliente = </span>
                    <input class="centralimg" name="txtCodigo"
style="width: 150px"
                type="text" id="txtCodigo" /></div>

        <div id="login" class="centro_img" style="position: absolute;
width: 400px; height: 250px; z-index: 6; left: 405px; top: 245px">
            <img alt="login" class="centralimg" height="67"
src="images/login.png" width="217" /><br />
            <br />
                <strong><span class="botao">Código Cliente:</span>
                   <span class="centralimg"> </span>
                  </strong>

                  <input class="centralimg" name="txtCodCliente"
style="width: 150px"
                type="text" id="txtCodCliente" />
                    <span class="centralimg"> </span>

                    <input class="botao" name="btnLogin" type="button"
value="login"
                id="btnLogin" onclick="return login()" />
        </div>
    </form>
</div>
</body>

</html>
```

Código do javascript utilizado para página de login do cliente (cliente.js)

```
function cadastro() {
    var wsa = new Coordenador.WSAlimentacao();
    var nome
    nome = document.getElementById("txtNome").value
    window.alert("vai tentar conectar o webservice");
    wsa.RegistroCliente(nome, cadastroCliente);
}

function login() {
    var wsa = new Coordenador.WSAlimentacao();
    var codCliente
    codCliente = document.getElementById("txtCodCliente").value
    window.alert("Verificando...");
    wsa.LoginCliente(codCliente, loginCliente);
}


function cadastroCliente(codCliente) {
    document.getElementById("txtCodigo").value = codCliente;
}

function loginCliente(Ok) {
    if (Ok == true) {
        window.alert("Bem-vindo!");
        window.open("opCliente.aspx", "", "", "");
    }
    if (Ok == false) {
```

```
        window.alert("Código inválido! Por favor, tente novamente ou
realize cadastro.");
    }

}
```

Código da página de login do teleoperador (teleop.aspx)

```
<%@ Page Language="vb" AutoEventWireup="true"
CodeBehind="teleop.aspx.vb" Inherits="Coordenador.teleop" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat = "server">
<meta content="pt-br" http-equiv="Content-Language" />
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>Teleoperador</title>
<style type="text/css">
.div {
      margin-right: auto;
      margin-left: auto;
}
.centro_img {
      text-align: center;
      border-style: solid;
      border-width: thin;
}
.centralimg {
      margin-top: 20px;
}
.style1 {
      margin-top: 0px;
}
.botao {
      font-family: Calibri;
      font-style: normal;
      font-weight: bold;
}
</style>
<meta content="Página destinada ao teleoperador. Monitoração e
Operação do Sistema produtivo" name="description" />
</head>

<body>

<form id="form1" runat="server">
         <asp:ScriptManager ID="ScriptManager1" runat="server" >
                        <Scripts>
                            <asp:ScriptReference Path="teleop.js" />
                        </Scripts>
                        <Services>
                            <asp:ServiceReference
Path="WSTeleoperacao.asmx" InlineScript="false" />
                        </Services>
        </asp:ScriptManager>

        <div id="titulo" style="position: absolute; width: 800px;
height: 50px; z-index: 3; left: 0">
```

```
            <img alt="titulo" class="div" height="50"
src="images/titulo.png" width="800" /></div>

        <div id="image" style="position: absolute; width: 513px;
height: 100px; z-index: 1; left: 0px; top: 69px">
            <img alt="PMRLSA" class="div" height="105"
src="images/CabecalhoPmrLsa.png" width="800" /></div>

        <div id="layer1" style="position: absolute; width: 822px;
height: 68px; z-index: 4; left: 0px; top: 173px">
            <img alt="tit_teleop" height="68"
src="images/tit_teleop.png" width="822" /></div>

        <div id="cadastro" class="centro_img" style="position:
absolute; width: 400px; height: 250px; z-index: 5; left: 0px; top:
245px">
            <img alt="cadastro" class="centralimg" height="67"
src="images/cadastro.png" width="217" /><br />
      <br />
            <span class="botao">Nome  </span>

              <input class="centralimg" id="txtNomeTeleop"
name="txtNomeTeleop" style="width: 216px" type="text" /><span
class="centralimg">   
              </span>

              <input class="botao" name="btnCadastro" type="button"
value="cadastro"
              onclick="return cadastro_Teleop()" id="btnCadastro"/>

              <span class="botao">codTeleop = </span>

              <input class="centralimg" name="txtCodigo"
style="width: 150px" type="text" id="txtCodigo" />

        </div>
        <div id="login" class="centro_img" style="position: absolute;
width: 400px; height: 250px; z-index: 6; left: 405px; top: 245px">
            <img alt="login" class="centralimg" height="67"
src="images/login.png" width="217" /><br />

            <br />
            <strong><span class="botao">Código Teleop:</span>

             <span class="centralimg"> </span>

              </strong><input class="centralimg" id="txtCodTeleop"
name="txtCodTeleop"
              style="width: 129px" type="text" />
              <span class="centralimg"> </span>

              <input class="botao" name="btnLogin" type="button"
value="login"
              id="btnLogin" onclick="return login_Teleop()" />
        </div>
</form>
</body>

</html>
```

Código do javascript utilizado para página de login do teleoperador (teleop.js)

```
function cadastro_Teleop() {
    var wsa = new Coordenador.WSTeleoperacao();
    var Nome
    Nome = document.getElementById("txtNomeTeleop").value
    wsa.RegistroTeleop(Nome, cadastroTeleop);
}

function login_Teleop() {
    var wsa = new Coordenador.WSTeleoperacao();
    var codTeleop
    codTeleop = document.getElementById("txtCodTeleop").value
    window.alert("Verificando...");
    wsa.Autorizacao(codTeleop, loginTeleop);
}


function cadastroTeleop(codTeleop) {
    document.getElementById("txtCodigo").value = codTeleop;
}

function loginTeleop(Ok) {
    if (Ok == true) {
        window.alert("Bem-vindo!");
        window.open("opTeleop.aspx", "", "", "");
    }
    if (Ok == false) {
        window.alert("Código inválido! Por favor, tente novamente ou
realize cadastro.");
    }

}
```

Código da página de operações do cliente (opcliente.aspx)

```
<%@ Page Language="vb" AutoEventWireup="false"
CodeBehind="opCliente.aspx.vb" Inherits="Coordenador.opCliente" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>opCliente</title>
<style type="text/css">
    .div {
          margin-right: auto;
          margin-left: auto;
    }
    .centro_img {
          text-align: center;
```

```
      border-style: solid;
      border-width: thin;
}
.centralimg {
      margin-top: 20px;
}
.botao {
      font-family: Calibri;
      font-style: normal;
      font-weight: bold;
      top: 20;
}
h1 {
      font-family: calibri;
}
.h1 {
      font-family: calibri;
      font-size: medium;
      font-weight: bold;
      text-align: center;
}
h1 {
      font-family: calibri;
      font-size: medium;
      font-weight: bold;
      text-align: center;
}
.emerg {
      border-style: solid;
      border-color: #FF0000;
      font-family: calibri;
}
.emerg {
      font-family: calibri;
      font-size: large;
      font-weight: bold;
      font-variant: small-caps;
      text-transform: none;
      color: #FF0000;
      border-style: solid;
      border-color: #FF0000;
      text-align: center;
}
h2 {
      font-family: calibri;
      font-size: small;
      font-weight: bold;
      text-align: left;
}
h2 {
      font-family: calibri;
      font-size: small;
      font-weight: bold;
      text-align: left;
}
h2 {
}
.style3 {
      text-decoration: underline;
}
    .style4 {
```

```
        text-align: left;
          border-style: solid;
          border-width: thin;
          font-family: calibri;
          font-size: medium;
          font-weight: bold;
    }
    #webcam1
    {
        height: 199px;
        width: 305px;
    }
</style>
<meta content="Janela de operações do cliente" name="description" />
</head>

<body>

<form id="form1" runat="server">

         <asp:ScriptManager ID="ScriptManager1" runat="server" >
              <Scripts>
                  <asp:ScriptReference Path="opCliente.js" />
              </Scripts>
              <Services>
                  <asp:ServiceReference Path="WSAlimentacao.asmx"
InlineScript="false" />
                  <asp:ServiceReference Path="Sup.asmx" />
              </Services>
         </asp:ScriptManager>

    <div id="image" style="position: absolute; width: 513px; height:
100px; z-index: 1; left: 0px; top: 69px">
          <img alt="PMRLSA" class="div" height="105"
src="images/CabecalhoPmrLsa.png" width="800" /></div>

    <div id="titulo" style="position: absolute; width: 800px; height:
50px; z-index: 3; left: 0">
          <img alt="titulo" class="div" height="50"
src="images/titulo.png" width="800" /></div>

    <div id="layer1" style="position: absolute; width: 822px; height:
68px; z-index: 4; left: 0; top: 172px">
          <img alt="tit_cliente" height="68"
src="images/tit_cliente.png" width="822" /></div>

          <div id="solicitacao" class="centro_img"
             style="position: absolute; width: 317px; height: 300px;
z-index: 5; left: 0px; top: 245px">
                    <span class="h1">Solicitação de Pedido<br />
                    </span>
                    <div id="disponibilidade0" class="style4"
                     style="position: absolute; width: 308px; height:
53px; z-index: 1; left: 4px; top: 34px">
                          <strong><span
class="style3">Cliente</span><br class="style3" />
                          Código
cliente:       
                          </strong> 
```

```
                        <input name="txtCodCliente" style="width:
46px" type="text"
                    id="txtCodCliente" />

                    </div>
                    <div id="disponibilidade" class="style4"
                    style="position: absolute; width: 311px; height:
67px; z-index: 1; left: 2px; top: 108px">
                        <strong><span
class="style3">Disponibilidade</span><br />

       </strong>

                        <input name="txtDisp" style="width: 102px"
type="text"
                    id="txtDisp"
/>           &n
bsp;   

                        <input class="botao" name="btnDisp"
type="button" value="Verificar"
                       id="btnDisp" onclick="return
disponibilidade()"/>        
           &nbs
p;      
                </div>
                    <div id="layer2" class="style4"

                    style="position: absolute; width: 308px; height:
107px; z-index: 2; left: 3px; top: 189px">
                        <strong><span class="style3">Pedido</span><br
/>
                        Total de peças:    </strong>

                        <input name="txtTotalPecas" style="width:
46px" type="text"
                    id="txtTotalPecas"
/>           &n
bsp;          
           &nbs
p;           &n
bsp; 

                        <input class="botao" name="btnSolic"
type="button" value="Solicitar"
                       id="btnSolic" onclick="return
solicitar()"/><strong><br />
                    Cód. do pedido:   </strong> <input
name="txtCodPedido" style="width: 46px" type="text"
                    id="txtCodPedido" />
                      </div>
          </div>
    <div id="layer3" class="centro_img" style="position: absolute;
width: 450px; height: 300px; z-index: 6; left: 350px; top: 245px">
          <span class="h1">Status do Pedido<br />
          <br />
          </span>
          <div id="entregue" class="style4"

            style="position: absolute; width: 418px; height: 254px; z-
index: 1; left: 25px; top: 33px">
```

```
                <strong><span class="style3">Concluído:</span><br
class="style3" />
                Peças
entregues:       
                </strong>
                <input name="txtPecasEntregues" style="width: 46px"
type="text"
                    id="txtPecasEntregues" /><br />
           &nbs
p;   
<img src="http://192.168.0.195:8080/loading.jpg" class="webcam"
id="webcam1"
                onmousedown="PTZMouseDown1(event)" alt="Live Stream"
/>    
<script type="text/javascript">
<!--
currentCamera1= 1;
errorimg1= 0;
document.images.webcam1.onload = DoIt1;
document.images.webcam1.onerror = ErrorImage1;
function LoadImage1()
{
        uniq1 = Math.random();
        document.images.webcam1.src = "http://192.168.0.195:8080/cam_"
+ currentCamera1 + ".jpg?uniq="+uniq1;
        document.images.webcam1.onload = DoIt1;
}
function PTZMouseDown1(e)
{
        var IE = document.all?true:false;
        var x,y;
        var myx,myy;
        var myifr = document.getElementById("_iframe-ptz");
        tp = getElPos1();
        myx = tp[0];
        myy = tp[1];
        if(IE){
        var scrollX = document.documentElement.scrollLeft ?
document.documentElement.scrollLeft : document.body.scrollLeft;
        var scrollY = document.documentElement.scrollTop ?
document.documentElement.scrollTop : document.body.scrollTop;
        x = event.clientX - myx + scrollX;
        y = event.clientY - myy + scrollY;
        } else {
        x = e.pageX - myx;
        y = e.pageY - myy;
        }
        if ((width_array[currentCamera1] != null) &&
(width_array[currentCamera1] > 0)) x = Math.round((x * 400) /
width_array[currentCamera1]);
        if ((height_array[currentCamera1] != null) &&
(height_array[currentCamera1] > 0)) y = Math.round((y * 300) /
height_array[currentCamera1]);
        if (x > 400) x = 400;
        if (y > 300) y = 300;
        if (myifr != null) myifr.src =
"http://192.168.0.195:8080/ptz?src=" + currentCamera1 + "&moveto_x=" +
x + "&moveto_y=" + y +"";
        return true;
}
function getElPos1()
```

```
{
            el = document.images.webcam1;
            x = el.offsetLeft;
            y = el.offsetTop;
            elp = el.offsetParent;
            while(elp!=null)
              { x+=elp.offsetLeft;
                y+=elp.offsetTop;
                elp=elp.offsetParent;
              }
            return new Array(x,y);
}
function ErrorImage1()
{
        erroimg1++;
        if (erroimg1>3){
              document.images.webcam1.onload = "";
              document.images.webcam1.onerror = "";
              document.images.webcam1.src = "offline.jpg";
              }else{
                uniq1 = Math.random();
            document.images.webcam1.src =
"http://192.168.0.195:8080/cam_" + currentCamera1 +
".jpg?uniq="+uniq1;
              }
}
function DoIt1()
{
        erroimg1=0;
        window.setTimeout("LoadImage1();", 40);
}
//-->
</script>



        </div>
    </div>
</form>
</body>

</html>
```

Código da javascript da página de operações do cliente (opcliente.js)

```
var disponivel;
var intervalo = window.setInterval(AtualizaPedido, 10000);

window.onload = function() {
    var wsSupervisorio = new Coordenador.Sup();
    var wsa = new Coordenador.WSAlimentacao();
    wsa.RetornaLoginCliente(descobrecodcliente);
    wsSupervisorio.CarregaXml();
    //window.alert("XmlCarregado!");
    wsSupervisorio.ConectaOPC(conecta);
};

function descobrecodcliente(codCliente) {
    document.getElementById("txtCodCliente").value = codCliente;
```

```
}


function conecta(ok) {
    if (ok == true) {
        window.alert("OPC conectado");
    }
    if (ok == false) {
        window.alert("OPC não conectado!");
    }
}

function disponibilidade() {
    var wsa = new Coordenador.WSAlimentacao();
    window.alert("Verificando disponibilidade...");
    wsa.DispAlimentacao(Disp);
}

function solicitar() {
    var wsa = new Coordenador.WSAlimentacao();
    var codCliente;
    var npecas;
    if (disponivel == true) {
        codCliente = document.getElementById("txtCodCliente").value;
        npecas = document.getElementById("txtTotalPecas").value;
        window.alert("Solicitando pedido");
        wsa.SolicPedido(codCliente, npecas, SolicitaPedido);
    }
    if (disponivel == false) {
        window.alert("Por favor, aguarde a disponibilidade!
Obrigada!");
    }
}


function Disp(Ok) {
    if (Ok == true) {
        disponivel = Ok;
        document.getElementById("txtDisp").value = "DISPONÍVEL";
    }
    if (Ok == false) {
        disponivel = Ok;
        document.getElementById("txtDisp").value = "INDISPONÍVEL";
    }
}

function SolicitaPedido(codPedido) {
    document.getElementById("txtCodPedido").value = codPedido;
}

function AtualizaPedido() {
    var wsa = new Coordenador.WSAlimentacao();
    var codPedido;
    codPedido = document.getElementById("txtCodPedido").value;
    wsa.ConsultaPedido(codPedido, estadopedido);
}

function estadopedido(nPecasEntregues) {
    var pecastotais;
    pecastotais = document.getElementById("txtTotalPecas");
```

```
    document.getElementById("txtPecasEntregues").value =
nPecasEntregues;
    if (pecastotais <= nPecasEntregues) {
        window.alert("Pedido Entregue!");
        document.getElementById("txtPecasEntregues").value = "";
        document.getElementById("txtTotalPecas").value = "";
        document.getElementById("txtDisp").value = "";
        document.getElementById("txtCodPedido").value = "";
    }

}
```

Código da página de operações do teleoperador (opteleop.aspx)

```
<%@ Page Language="vb" AutoEventWireup="false"
CodeBehind="opTeleop.aspx.vb" Inherits="Coordenador.opTeleop" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta content="pt-br" http-equiv="Content-Language" />
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>Teleoperacao</title>
<style type="text/css">
    .div {
          margin-right: auto;
          margin-left: auto;
    }
    .centro_img {
          text-align: center;
          border-style: solid;
          border-width: thin;
    }
    .centralimg {
          margin-top: 20px;
    }
    .botao {
          font-family: Calibri;
          font-style: normal;
          font-weight: bold;
          top: 20;
        height: 22px;
        margin-bottom: 0px;
    }
    h1 {
          font-family: calibri;
    }
    .h1 {
          font-family: calibri;
          font-size: medium;
          font-weight: bold;
          text-align: center;
    }
    h1 {
          font-family: calibri;
          font-size: medium;
          font-weight: bold;
```

```
      text-align: center;
}
.style1 {
      text-decoration: underline;
}
.style2 {
      font-family: calibri;
      font-size: medium;
      font-weight: bold;
      text-align: center;
      text-decoration: underline;
}
.style3 {
      text-align: center;
      border-style: solid;
      border-width: thin;
      font-family: calibri;
      font-size: medium;
      font-weight: bold;
}
.style4 {
      text-align: center;
      border-style: solid;
      border-width: thin;
      font-family: calibri;
      font-size: medium;
      font-weight: bold;
      text-decoration: underline;
}
.style5 {
      text-align: left;
      border-style: solid;
      border-width: thin;
      font-family: calibri;
      font-size: medium;
      font-weight: bold;
}
.style6 {
      text-align: right;
      border-style: solid;
      border-width: thin;
      font-family: calibri;
      font-size: medium;
      font-weight: bold;
}
.style7 {
      text-align: right;
}
.style8 {
      font-family: Calibri;
      font-style: normal;
      top: 20;
      font-size: small;
}
.emerg {
      border-style: solid;
      border-color: #FF0000;
      font-family: calibri;
}
.emerg {
      font-family: calibri;
```

```
            font-size: large;
            font-weight: bold;
            font-variant: small-caps;
            text-transform: none;
            color: #FF0000;
            border-style: solid;
            border-color: #FF0000;
            text-align: center;
        }
        .style9 {
              text-align: left;
              margin-left: 80px;
        }
        #webcam1
        {
            height: 217px;
            width: 267px;
        }
</style>
<meta content="Operacao de tele-operação." name="description" />
</head>

<body>

<form id="form1" runat="server">

       <asp:ScriptManager ID="ScriptManager1" runat="server" >
              <Scripts>
                  <asp:ScriptReference Path="opTeleop.js" />
              </Scripts>
              <Services>
                  <asp:ServiceReference Path="Sup.asmx" />
                  <asp:ServiceReference Path="WSTeleoperacao.asmx"
InlineScript="false" />
                  <asp:ServiceReference Path="WSAlimentacao.asmx" />
              </Services>
         </asp:ScriptManager>

    <div id="image" style="position: absolute; width: 513px; height:
100px; z-index: 1; left: 0px; top: 69px">
          <img alt="PMRLSA" class="div" height="105"
src="images/CabecalhoPmrLsa.png" width="800" /></div>

    <div id="layer1" style="position: absolute; width: 822px; height:
68px; z-index: 4; left: 0px; top: 173px">
          <img alt="tit_teleop" height="68"
src="images/tit_teleop.png" width="822" /></div>

    <div id="titulo" style="position: absolute; width: 800px; height:
50px; z-index: 3; left: 0">
          <img alt="titulo" class="div" height="50"
src="images/titulo.png" width="800" /></div>

    <div id="modo" style="position: absolute; width: 200px; height:
100px; z-index: 5; left: 0px; top: 370px" class="centro_img">
          <span class="style2">Modo de Operação<br /><br /></span>

          <select class="botao" id="sel_modo" name="sel_modo" size="1"
style="width: 88px">
              <option value="monit">Monitorar</option>
              <option value="teleop">Teleoperar</option>
```

```
          </select> 

          <input class="botao" name="btnModo" id="btnModo"
type="button" value="modo" onclick="return modo()" />
      </div>

      <div id="Pedido" class="centro_img" style="position: absolute;
width: 200px; height: 100px; z-index: 6; left: 2px; top: 520px">
                <span class="h1"><span class="style1">Pedido<br />
            <br />
            </span>código Pedido: </span>
                <input name="txtCodPedido" style="width: 46px"
type="text"
                    id="txtCodPedido" />
            <br />
            <br />
            </div>
      <div id="Teleoperador" class="style3" style="position: absolute;
width: 200px; height: 75px; z-index: 7; left: 0px; top: 245px">
                <span class="style1">Tele-operador</span><br />
                <br />
                Código 
                <input class="botao" name="txtCodTeleop"
id="txtCodTeleop" style="width: 72px" type="text" />
      </div>



      <div id="video" class="style4" style="position: absolute; width:
300px; height: 300px; z-index: 8; left: 227px; top: 245px">
                Visualização<br />
            <br />

     <script type="text/javascript">
  var height_array = new Array();
  var width_array = new Array();
  width_array[1] = 352;
  height_array[1] = 288;
                        </script>
                        <img
src="http://192.168.0.195:8080/loading.jpg" class="webcam"
id="webcam1"

onmousedown="PTZMouseDown1(event)" alt="Live Stream" />
                        <script type="text/javascript">
                        <!--
                        currentCamera1= 1;
                        errorimg1= 0;
                        document.images.webcam1.onload = DoIt1;
                        document.images.webcam1.onerror = ErrorImage1;
                        function LoadImage1()
                        {
                                uniq1 = Math.random();
                                document.images.webcam1.src =
"http://192.168.0.195:8080/cam_" + currentCamera1 +
".jpg?uniq="+uniq1;
                                document.images.webcam1.onload =
DoIt1;
                        }
                        function PTZMouseDown1(e)
```

```
                        {
                                var IE = document.all?true:false;
                                var x,y;
                                var myx,myy;
                                var myifr =
document.getElementById("_iframe-ptz");
                                tp = getElPos1();
                                myx = tp[0];
                                myy = tp[1];
                                if(IE){
                                var scrollX =
document.documentElement.scrollLeft ?
document.documentElement.scrollLeft : document.body.scrollLeft;
                                var scrollY =
document.documentElement.scrollTop ?
document.documentElement.scrollTop : document.body.scrollTop;
                                x = event.clientX - myx + scrollX;
                                y = event.clientY - myy + scrollY;
                                } else {
                                x = e.pageX - myx;
                                y = e.pageY - myy;
                                }
                                if ((width_array[currentCamera1] !=
null) && (width_array[currentCamera1] > 0)) x = Math.round((x * 400) /
width_array[currentCamera1]);
                                if ((height_array[currentCamera1] !=
null) && (height_array[currentCamera1] > 0)) y = Math.round((y * 300)
/ height_array[currentCamera1]);
                                if (x > 400) x = 400;
                                if (y > 300) y = 300;
                                if (myifr != null) myifr.src =
"http://192.168.0.195:8080/ptz?src=" + currentCamera1 + "&moveto_x=" +
x + "&moveto_y=" + y +"";
                                return true;
                        }
                        function getElPos1()
                        {
                                    el = document.images.webcam1;
                                    x = el.offsetLeft;
                                    y = el.offsetTop;
                                    elp = el.offsetParent;
                                    while(elp!=null)
                                      { x+=elp.offsetLeft;
                                        y+=elp.offsetTop;
                                        elp=elp.offsetParent;
                                      }
                                    return new Array(x,y);
                        }
                        function ErrorImage1()
                        {
                                erroimg1++;
                                if (erroimg1>3){
                                      document.images.webcam1.onload =
"";
                                      document.images.webcam1.onerror
= "";
                                      document.images.webcam1.src =
"offline.jpg";
                                      }else{
                                        uniq1 = Math.random();
```

```
                                        document.images.webcam1.src =
"http://192.168.0.195:8080/cam_" + currentCamera1 +
".jpg?uniq="+uniq1;
                                            }
                        }
                        function DoIt1()
                        {
                                errorimg1=0;
                                window.setTimeout("LoadImage1();",
40);
                        }
                        //-->
                        </script>


        </div>


      <div id="atividade" class="style5" style="position: absolute;
width: 300px; height: 70px; z-index: 9; left: 228px; top: 550px">
           Atividade:

        <input class="botao" name="txtAtividade" id="txtAtividade"
style="width: 174px"
            type="text" />

           <br />
                   <br />
                 
   <input class="botao" name="btnFazer" id="btnFazer"
type="button" value="Fazer" onclick="return fazer()"/>

             <input class="botao" name="btnNaoFazer" id="btnNaoFazer"
type="button" value="Não Fazer" onclick="return nfazer()" />
      </div>
      <div id="estado" class="style6"
           style="position: absolute; width: 245px; height: 218px; z-
index: 10; left: 555px; top: 245px">
                <span class="style2">Estado Equipamento</span><span
class="style8">      
                     <br
/></span>
                        <div id="layer2" class="style7"
style="position: absolute; width: 150px; height: 170px; z-index: 1;
left: 3px; top: 40px">
                            <span class="style8">
                            <span class="botao">Tem peça:<br />
                              <br />
                    </span>Cilindro Recuado:<br />
                            <br />
                            Cilindro Avançado:<br />
                            <br />
                            Braço na alimentação:<br />
                            <br />
                            Braço na inspeção:<br />
                            <br />
                            Peça presa pela ventosa:</span>
```

```
                        <span class="botao"><br />
                    </span>
                    </div>
                    <div id="layer6" class="style7"style="position:
absolute; width: 48px; height: 20px; z-index: 1; left: 159px; top:
92px">
                        <asp:Image ID="imgOk_R001" runat="server"
ImageUrl="~/images/ok.png"
                            Width="16px" />
                     <asp:Image ID="imgNOK_R001" runat="server"
ImageUrl="~/images/nok.png"
                            Width="19px" />
                </div>
                    <div id="layer5" class="style7" style="position:
absolute; width: 48px; height: 20px; z-index: 1; left: 159px; top:
64px">
                        <asp:Image ID="imgOk_R000" runat="server"
ImageUrl="~/images/ok.png"
                            Width="16px" />
                     <asp:Image ID="imgNOK_R000" runat="server"
ImageUrl="~/images/nok.png"
                        Width="19px" />
                    </div>
                      <div id="layer4" class="style7" style="position:
absolute;width: 48px; height: 20px; z-index: 1; left: 160px; top:
38px">
                        <asp:Image ID="imgOk_R006" runat="server"
ImageUrl="~/images/ok.png"
                            Width="16px" />
                            </div>
                      <div id="layer9" class="style7"

                style="position: absolute; width: 48px; height: 20px;
z-index: 1; left: 161px; top: 185px">
                        <asp:Image ID="imgOk_R005" runat="server"
ImageUrl="~/images/ok.png"
                            Width="16px" />
                     <asp:Image ID="imgNOK_R005" runat="server"
ImageUrl="~/images/nok.png"
                            Width="19px" />

          
                    <br />

           &nbs
p;
                    <br />

           &nbs
p;
                    <br />

           &nbs
p;
                    <br />
                    <br />

           
                    <br />
```

```
           &nbs
p;           
                    </div>
                      <div id="layer8" class="style7"

                style="position: absolute; width: 48px; height: 20px;
z-index: 1; left: 161px; top: 153px">
                            <asp:Image ID="imgOk_R003" runat="server"
ImageUrl="~/images/ok.png"
                        Width="16px" />
                     <asp:Image ID="imgNOK_R003" runat="server"
ImageUrl="~/images/nok.png"
                        Width="19px" />

          
                    <br />

           &nbs
p;
                    <br />

           &nbs
p;
                    <br />

           &nbs
p;
                    <br />
                    <br />

          
                    <br />

           &nbs
p;           
                    </div>
                      <div id="layer7" class="style7"

                style="position: absolute; width: 48px; height: 20px;
z-index: 1; left: 161px; top: 122px">
                            <asp:Image ID="imgOk_R002" runat="server"
ImageUrl="~/images/ok.png"
                        Width="16px" />
                     <asp:Image ID="imgNOK_R002" runat="server"
ImageUrl="~/images/nok.png"
                        Width="19px" />

          
                    <br />

           &nbs
p;
                    <br />

           &nbs
p;
                    <br />

           &nbs
p;
```

```
                    <br />
                    <br />

          
                    <br />

            &nbs
p;           
                    </div>
      </div>

      <div id="emergencia" style="position: absolute; width: 245px;
height: 140px; z-index: 11; left: 555px; top: 480px">
          <p class="style9">
              <asp:ImageButton ID="btnEmergencia" runat="server"
Height="116px"
                  ImageUrl="~/images/emergency.PNG" Width="115px"
OnClientClick="emergencia(); return false;"   />
          </p>
      </div>
</form>
</body>

</html>
```

Código da javascript da página de operações do teleoperador (opteleop.js)

```
var codTeleoperador;
var modoop;
var codigoPedido;
var intervalo = window.setInterval(AtualizaSensores, 5000);
var intervalo2 = window.setInterval(acordaSup, 7000);
var intervalo3 = window.setInterval(Telecomando, 10000);
var time = window.setTimeout(inicio, 1000);


window.onload = function() {
//inicializa imagens da tela
    document.getElementById("imgNOK_R000").style.visibility =
"hidden";
    document.getElementById("imgOk_R000").style.visibility = "hidden";
    document.getElementById("imgNOK_R001").style.visibility =
"hidden";
    document.getElementById("imgOk_R001").style.visibility = "hidden";
    document.getElementById("imgNOK_R002").style.visibility =
"hidden";
    document.getElementById("imgOk_R002").style.visibility = "hidden";
    document.getElementById("imgNOK_R003").style.visibility =
"hidden";
    document.getElementById("imgOk_R003").style.visibility = "hidden";
    document.getElementById("imgNOK_R005").style.visibility =
"hidden";
```

```
    document.getElementById("imgOk_R005").style.visibility = "hidden";
    document.getElementById("imgNOK_R006").style.visibility =
"hidden";
    document.getElementById("imgOk_R006").style.visibility = "hidden";
    //conecta opc e loga tele-operador
};

function inicio() {
    var wsSup = new Coordenador.Sup();
    var wsa = new Coordenador.WSTeleoperacao();
    //window.alert("abriu o load");
    wsa.RetornaLoginTeleop(desccod);
    wsSup.CarregaXml();
    //window.alert("XmlCarregado!");
    wsSup.ConectaOPC(conecta);
}

function desccod(codTeleop) {
   // window.alert(codTeleop);
    document.getElementById("txtCodTeleop").value = codTeleop;
    codTeleoperador = codTeleop;
}

function conecta(ok) {
    if (ok == true) {
        window.alert("OPC conectado");
    }
    if (ok == false) {
        window.alert("OPC não conectado!");
    }
}

function modo() {
    var wsa = new Coordenador.WSTeleoperacao();
    modoop = document.getElementById("sel_modo").value;
    wsa.ModoOperacao(codTeleoperador, modoop, alteramodo);
}

function emergencia() {
    var wsSup = new Coordenador.Sup();
    wsSup.ParadaEmergencia();
}

function alteramodo(ok) {
    if (ok == true) {
        if (modoop == "monit") {
            window.alert("Modo de Monitoração");
            document.getElementById("btnFazer").disabled=true;
            document.getElementById("btnNaoFazer").disabled=true;
        }
        if (modoop == "teleop") {
            window.alert("Modo de Tele-operação. Observe as atividades
pendentes!");
            document.getElementById("btnFazer").disabled = false;
            document.getElementById("btnNaoFazer").disabled = false;
        }
    }
    if (ok == false) {
        window.alert("Por favor, selecione o modo novamente.
Obrigada");
    }
```

```
}

function AtualizaSensores() {
    var wsSup = new Coordenador.Sup();
    //wsSup.Acorda();
    wsSup.lerPorta("R000", atualizaEstado_R000);
    wsSup.lerPorta("R001", atualizaEstado_R001);
    wsSup.lerPorta("R002", atualizaEstado_R002);
    wsSup.lerPorta("R003", atualizaEstado_R003);
    wsSup.lerPorta("R005", atualizaEstado_R005);
    wsSup.lerPorta("R006", atualizaEstado_R006);
}

function atualizaEstado_R000(estado_port) {
    if (estado_port == "0") {
        document.getElementById("imgNOK_R000").style.visibility =
"visible";
        document.getElementById("imgOk_R000").style.visibility =
"hidden";
    }
    if (estado_port == "-1") {
        document.getElementById("imgNOK_R000").style.visibility =
"hidden";
        document.getElementById("imgOk_R000").style.visibility =
"visible";
    }
}

function atualizaEstado_R001(estado_port) {
    if (estado_port == "0") {
        document.getElementById("imgNOK_R001").style.visibility =
"visible";
        document.getElementById("imgOk_R001").style.visibility =
"hidden";
    }
    if (estado_port == "-1") {
        document.getElementById("imgNOK_R001").style.visibility =
"hidden";
        document.getElementById("imgOk_R001").style.visibility =
"visible";
    }
}

function atualizaEstado_R002(estado_port) {
    if (estado_port == "0") {
        document.getElementById("imgNOK_R002").style.visibility =
"visible";
        document.getElementById("imgOk_R002").style.visibility =
"hidden";
    }
    if (estado_port == "-1") {
        document.getElementById("imgNOK_R002").style.visibility =
"hidden";
        document.getElementById("imgOk_R002").style.visibility =
"visible";
    }
}

function atualizaEstado_R003(estado_port) {
    if (estado_port == "0") {
```

```
        document.getElementById("imgNOK_R003").style.visibility =
"visible";
        document.getElementById("imgOk_R003").style.visibility =
"hidden";
    }
    if (estado_port == "-1") {
        document.getElementById("imgNOK_R003").style.visibility =
"hidden";
        document.getElementById("imgOk_R003").style.visibility =
"visible";
    }
}
function atualizaEstado_R005(estado_port) {
    if (estado_port == "0") {
        document.getElementById("imgNOK_R005").style.visibility =
"visible";
        document.getElementById("imgOk_R005").style.visibility =
"hidden";
    }
    if (estado_port == "-1") {
        document.getElementById("imgNOK_R005").style.visibility =
"hidden";
        document.getElementById("imgOk_R005").style.visibility =
"visible";
    }
}
function atualizaEstado_R006(estado_port) {
    if (estado_port == "0") {
        document.getElementById("imgNOK_R006").style.visibility =
"visible";
        document.getElementById("imgOk_R006").style.visibility =
"hidden";
    }
    if (estado_port == "-1") {
        document.getElementById("imgNOK_R006").style.visibility =
"hidden";
        document.getElementById("imgOk_R006").style.visibility =
"visible";
    }
}
// ************************************

function acordaSup(){
    var wsSup = new Coordenador.Sup();
    wsSup.Acorda();
}

function Telecomando(){
    var wsTele = new Coordenador.WSTeleoperacao();
    var wsa = new Coordenador.WSAlimentacao();
    var codTeleop;
    var ModoOperacao;
    codigoPedido = 0;
    ModoOperacao = document.getElementById("sel_modo").value;
    codTeleop = document.getElementById("txtCodTeleop").value;
  //  window.alert(codTeleop);
    wsa.RetornaCodPedidoTeleop(codTeleop, descobrecodpedido);
}

function descobrecodpedido(codPedido) {
    var wsTele = new Coordenador.WSTeleoperacao();
```

```
    codigoPedido = codPedido;
  //  window.alert(codigoPedido);
    document.getElementById("txtCodPedido").value = codigoPedido;
    if (codigoPedido == 0) {

    }
    else {
        if (modoop == "teleop") {
            wsTele.InfEstTelecomando(codTeleoperador, codPedido,
atualizaatividade);
        }
    }

}


function atualizaatividade(atividade) {
   // window.alert(atividade);
    document.getElementById("txtAtividade").value = atividade;
}

function fazer() {
    var wsTele = new Coordenador.WSTeleoperacao();
    var wsa = new Coordenador.WSAlimentacao();
    var codPedido = 0;
    var codTeleop;
    codTeleop = document.getElementById("txtCodTeleop").value;
    codPedido = document.getElementById("txtCodPedido").value;
    wsTele.AtualizaEstTelecomando(codPedido, codTeleop, fazerok);
}


function fazerok(Ok) {
    if (Ok == false) {
        window.alert("Por favor, decida novamente sobre esta
operação");
    }
    if (Ok == true) {
        window.alert("Operação em processo!");
    }
}

function nfazer() {
    window.alert("Ok. Aguardando autorização para executar esse
passo");
}
```