Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

BRUNA SALLES MOREIRA

ESTÁGIO INTEGRADO
SCHNEIDER ELECTRIC FRANCE

Campina Grande, Paraíba
Dezembro de 2017

BRUNA SALLES MOREIRA

ESTÁGIO INTEGRADO
SCHNEIDER ELECTRIC FRANCE

Relatório de Estágio Integrado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Instrumentação Eletrônica e Controle

Orientador:
Professor Jaidilson Jó da Silva, D. Sc.

Campina Grande, Paraíba
Dezembro de 2017

BRUNA SALLES MOREIRA

ESTÁGIO INTEGRADO
SCHNEIDER ELECTRIC FRANCE

Relatório de Estágio Integrado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica

Área de Concentração: Instrumentação Eletrônica e Controle

Aprovado em ____ / ____ / _____

_____

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

_____

Professor Jaidilson Jó da Silva, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho à minha mãe, Lívia Coelho Salles, à minha irmã, Isadora, à minha avó, Lêda, e a todos os colegas que de forma direta ou indireta me ajudaram nessa conquista. Obrigada!

AGRADECIMENTOS

À Deus, dedico o meu agradecimento maior, pela dádiva da vida e por iluminar minha caminhada.

Homenageio aqui a minha mãe, Lívia Salles, por sempre ter sido meu porto seguro nas tempestades mais severas, minha eterna companheira e melhor amiga. Ao meu pai, André Moreira, pelo amor e incentivo.

A minha avó Lêda Salles, na qual espelhei-me em sua espiritualidade e sabedoria.

Como também agradeço ao professor Jaidilson, pela orientação ao longo do curso, por toda a paciência, dedicação e ensinamentos.

A José Antônio Neto, amigo e companheiro, agradeço por toda a compreensão, incentivo e carinho. Aos amigos da graduação Clara, Sara e Rodolfo, que fizeram esta jornada muita mais alegre e divertida.

E a todos os meus familiares, mestres e demais amigos que participaram dessa trajetória, colaborando para concretização desta etapa em minha vida.

LISTA DE FIGURAS

SUMARIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Na estrutura curricular do curso de Engenharia Elétrica da UFCG, o estágio é uma disciplina obrigatória, sendo requisito necessário à conclusão do curso. Espera-se que a conciliação entre os conhecimentos teóricos e práticos seja uma contribuição para a formação profissional do futuro engenheiro.

Isto posto, este relatório tem como objetivo apresentar os conhecimentos adquiridos e relatar as principais atividades realizadas durante o estágio integrado realizado na empresa Schneider Electric France, no período compreendido entre 27 de fevereiro de 2017 e 18 de agosto de 2017.

As atividades desempenhadas pela estagiária se concentraram no estudo e na otimização dos testes automatizados realizados nos Controladores Lógicos Programáveis (CLP), da Schneider Electric, assim como na rede industrial Modbus e no *software* de programação SoMachine Basic.

1.2 OBJETIVOS

A avaliação é uma parte essencial no desenvolvimento e manutenção de produtos. O teste é um método para reduzir o risco de entregar um produto não utilizável para os clientes.

O laboratório OEM é especializado na verificação e na validação do *firmware* dos controladores da gama Machine Solutions, da Schneider Electric. A verificação certifica que o produto satisfaz as especificações e a validação certifica que o produto cumpre os requisitos pretendidos e respeita as normas em vigor, como descrito no *Offer Requirements* quando colocados em condições operacionais. A validação também deve considerar:

- Ergonomia e integração de sistemas,
- Documentação do cliente (manuais, folhas de instruções, rótulos de produtos, mensagens de aviso),
- Compatibilidade,

- Performance,
- Casos de utilização dos clientes,
- O nível de robustez do produto anterior, que será substituído.

O serviço de validação do Laboratório OEM é responsável pela qualidade do produto do ponto de vista do cliente, através da combinação dos produtos em teste a um sistema.

Os testes podem ser automáticos ou manuais. Os testes iniciados manualmente podem ser cansativos, pois os procedimentos são bastante repetitivos, e podem demorar um tempo relativamente longo, há também o risco de erro humano.

Testes automatizados são mais rápidos para executar do que os testes manuais, portanto, pode-se executá-los com mais frequência. A automação de testes permite executar a vontade os testes que asseguram a não-regressão, após a entrega de uma nova versão do *software* ou uma nova versão do *firmware*.

Desta forma, o estágio visa otimizar testes automatizados existentes em SoMachine Basic, para reduzir significativamente o tempo necessário para o teste de aceitação e assegurar a não-regressão sobre as diferentes versões do *firmware* e do *software*.

Em uma segunda parte, esse estágio visa realizar a integração de um sistema de visão industrial, que foi implementado em um estágio anterior, usando uma câmera para realizar o monitoramento automático dos LEDs indicadores nos controladores e M221ME16R e M221ME16T.

2 A EMPRESA

Este capítulo apresentará o grupo Schneider Electric e sua subsidiária em Carros, onde o estágio foi realizado durante 6 meses.

2.1 GRUPO SCHNEIDER ELECTRIC

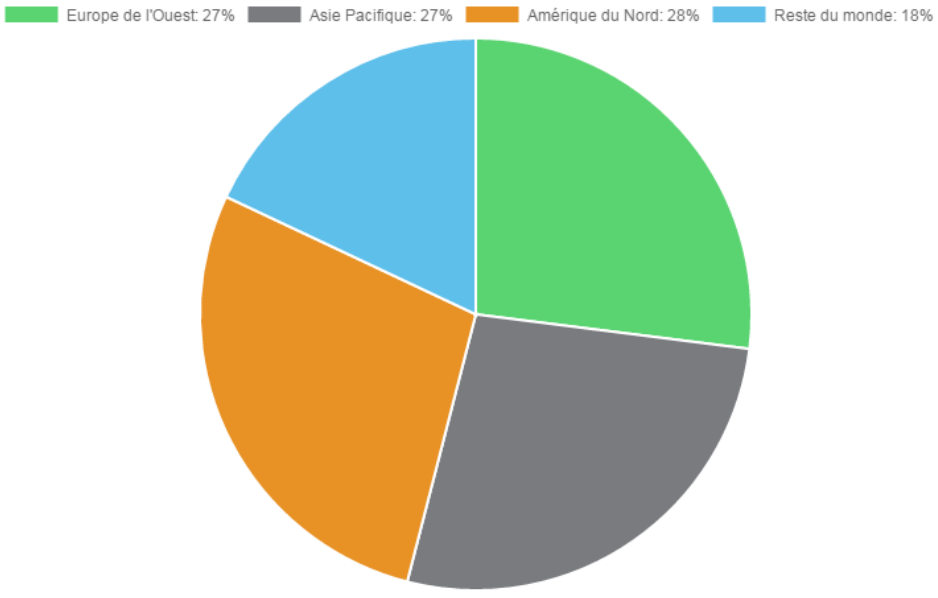
Schneider Electric é um grupo industrial francês de dimensão internacional, que é especialista mundial na gestão de energia e em automação.

Com um volume de negócios de 24,7 bilhões de dólares no ano de 2016, mais de 144.000 funcionários atendem clientes em 190 países, respondendo as expectativas de seus mercados com produtos adaptados às normas e práticas de cada país. A Figura 2.1 apresenta o volume de negócios por zona geográfica em 2016.

Schneider Electric desenvolve produtos, soluções e serviços para tornar o ecossistema de seus clientes seguro, confiável, eficiente e durável. O Grupo investe fortemente em tecnologia para apoiar a inovação, com um forte compromisso com o desenvolvimento sustentável [1].

A missão da Schneider Electric é desenvolver, o benefício de clientes, produtos e serviços inovadores que tornam a vida mais fácil para os seus utilizadores.

Figura 2.1: Volume de negócios por zona geográfica em 2016



2.2 HISTÓRIA

Em 180 anos de existência, a Schneider Electric passou da siderurgia, mecânica pesada e construção naval à gestão de eletricidade e automação [2].

Em 1836, os irmãos Schneider adquiriram minas e forjarias em Le Creusot, na França. Dois anos depois criaram a Schneider & Cie.

Após a II Guerra Mundial, a empresa passou por uma reestruturação profunda liderada por Charles Schneider.

Em 1975, O Grupo Schneider adquire uma participação na Merlin Gerin, uma das líderes em equipamentos de distribuição elétrica.

Entre 1981 e 1997, a empresa sai do setor de siderurgia e de fabricação de navios e foca principalmente em equipamentos para eletricidade por meio de aquisições estratégicas: Télémécanique em 1988, Square D em 1991 e, em 1995, Modicon torna-se uma marca Schneider.

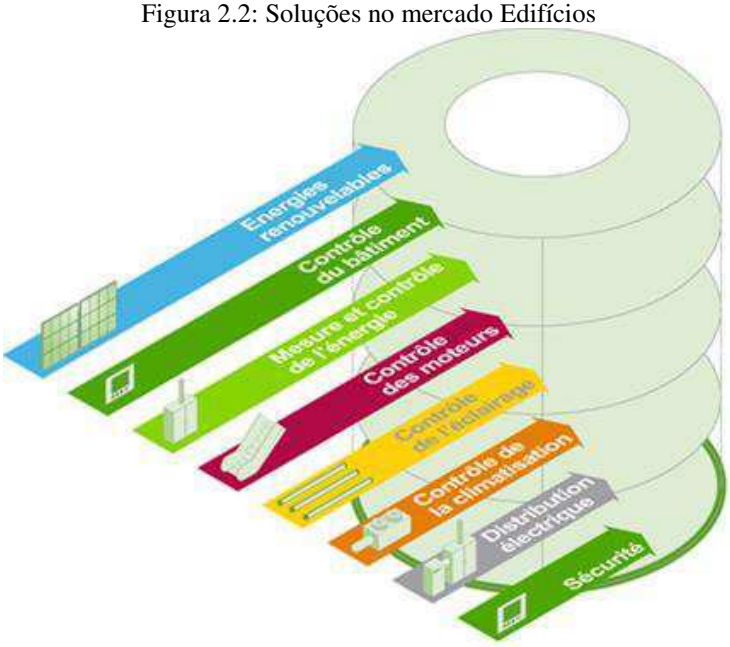
Desde 2000, a Schneider Electric continua sua política de aquisições para posicionar-se em novos segmentos de mercado: onduladores, no-breaks, automação predial e segurança para edificações. Hoje, a Schneider Electric tem mais de 170 marcas, o resultado de sua ativa estratégia de aquisição [2].

2.3 SOLUÇÕES, PRODUTOS E CLIENTES

Schneider Electric oferece soluções que cobrem as necessidades de eficiência energética e eficiência operacional dos clientes em 4 principais mercados: edifícios, centros de dados e redes, indústria, energia e infraestrutura.

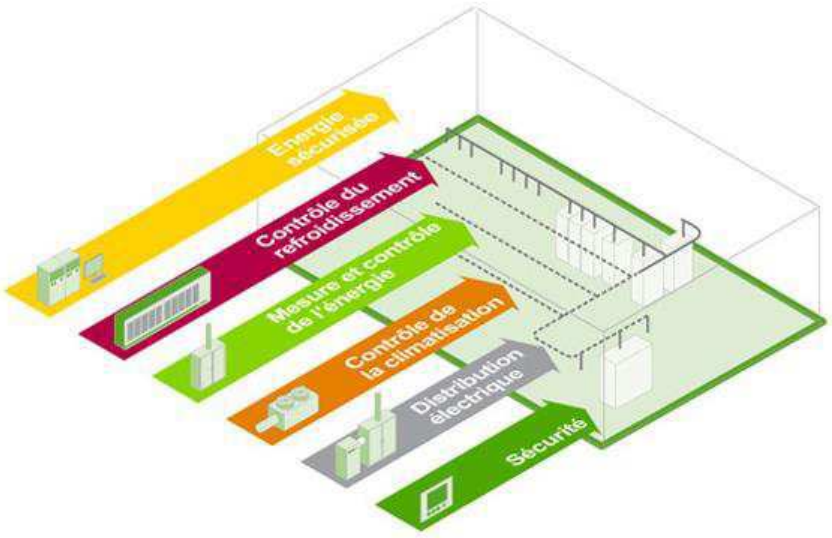
Para os edifícios, a Schneider Electric oferece uma série de soluções, como apresentado na Figura 2.2, por exemplo: o disjuntor Masterpact MTZ, um produto conectado que melhora o desempenho, a fiabilidade e eficiência energética. As soluções incluem a transformação e a distribuição de energia, gestão de utilidades, troca de dados, medição e controle do consumo de energia, segurança. Os principais clientes são os desenvolvedores, consultores, integradores de sistemas, instaladores e fabricantes de quadros, distribuidores elétricos, unidades operacionais e clientes finais [3].

Figura 2.2: Soluções no mercado Edifícios



Os centros de dados devem ser capazes de monitorar e controlar toda a infraestrutura física para melhorar o desempenho através da automatização e integração da gestão de centro de dados. Desta forma, a Schneider Electric oferece soluções, apresentado na Figura 2.3, como o inversor trifásico Galaxy VX UPS, um produto conectado para proteger a disponibilidade de centros de dados através de um ondulador evolutivo. As soluções incluem a distribuição de energia, gestão e controle de energia, a auditoria da instalação, o monitoramento e análise de dados on-line, treinamento, manutenção, monitoramento e segurança. Os principais clientes vão desde PME a multinacionais [3].

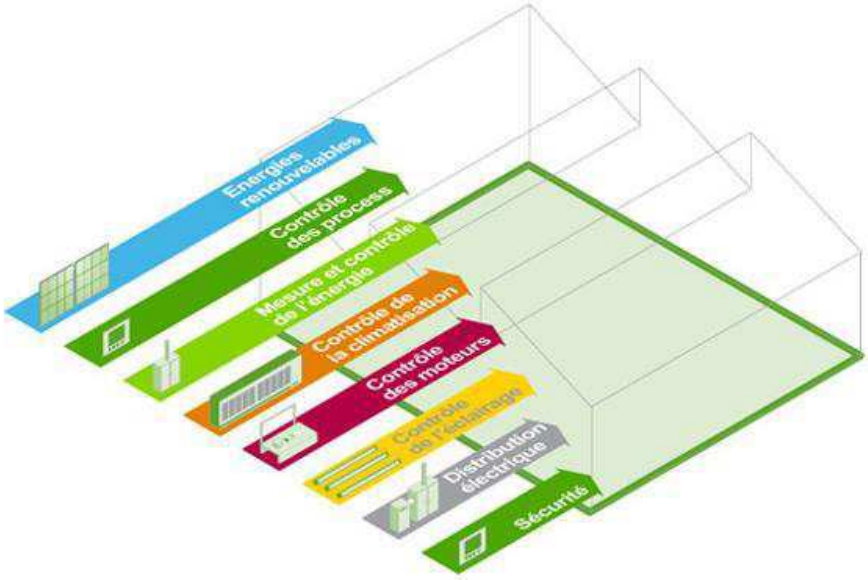
Figura 2.3: Soluções no mercado de centro de dados e redes



Em todos os setores da indústria, Schneider propõe soluções para distribuir energia, otimizar a sua utilização, melhorar a produtividade, garantir a segurança, qualidade e

rastreabilidade nas linhas de produção, como apresentado na Figura 2.4. Os principais clientes são fabricantes de máquinas, grandes indústrias, distribuidores de equipamentos elétricos e os clientes finais [3].

Figura 2.4: Soluções no mercado da indústria



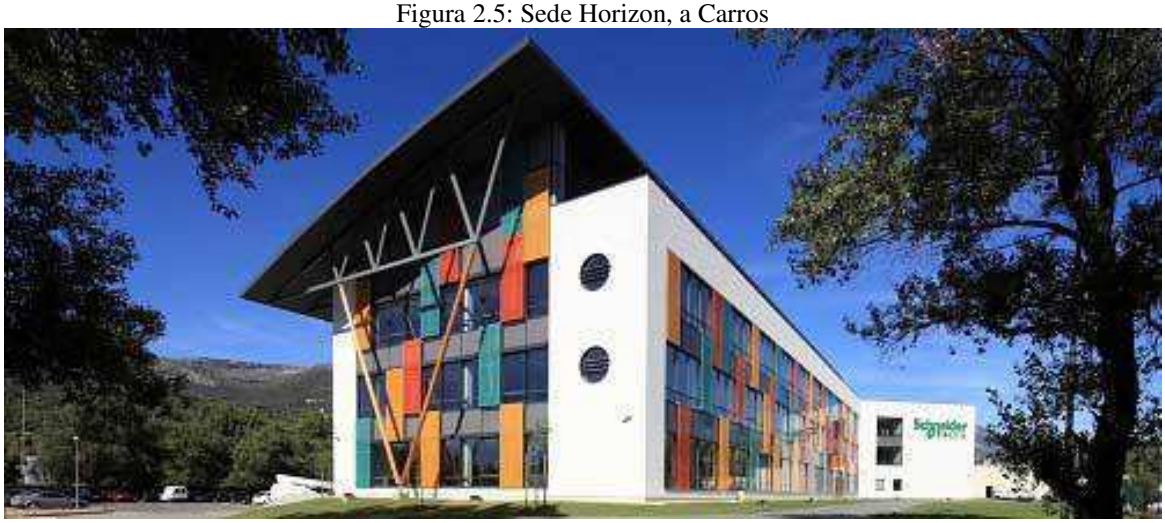
No mercado de energia e infraestrutura, Schneider Electric oferece soluções para garantir a qualidade da energia elétrica, transporte e distribuição confiável, garantindo a segurança e disponibilidade. Os produtos e serviços incluem: processamento e distribuição de energia, medição e controle do consumo e da qualidade, criação e gestão de redes inteligentes, gestão de serviços públicos, controle e supervisão do processo, a gestão descentralizada de um ou mais locais [3].

2.4 SCHNEIDER ELETRIC SEDE HORIZON

A sede social da empresa está localizada, em Carros, França, apresentada na Figura 2.5, e é especializada no setor de fabricação de componentes dos sistemas de controle e da montagem de componentes para placas eletrônicos e módulos CLP.

O centro Horizon reúne todas as atividades de automação, foi criado dando continuidade de uma política da empresa que tem como objetivo desenvolver, dentro das indústrias do Grupo, verdadeiras vitrines da oferta Schneider Electric.

Figura 2.5: Sede Horizon, a Carros

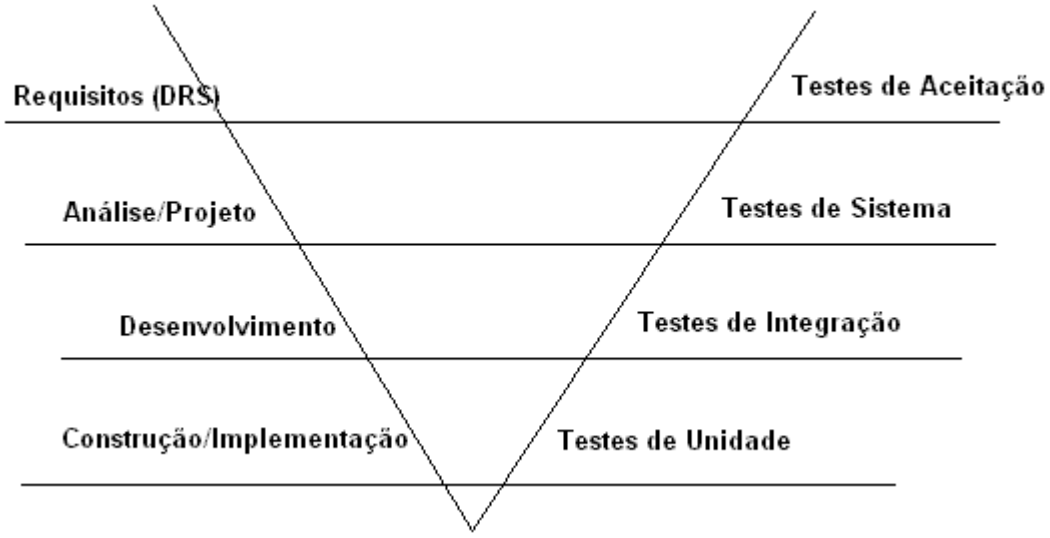


2.4.1 LABORATÓRIO OEM (ORIGINAL EQUIPMENT MANUFACTURER)

O laboratório OEM (agora "Controller Centric Lab & OI"), em Carros, tem como objetivo verificar e validar os produtos de controle do departamento Machine Solutions.

No desenvolvimento do ciclo em V de produtos, como apresentado na Figura 2.6, o laboratório OEM intervém no nível dos testes unitários, teste de integração e de validação. Ele executa as etapas de verificação e validação de novos produtos antes que eles sejam presentes no mercado.

Figura 2.6: Laboratório OEM no ciclo em V



3 FUNDAMENTAÇÃO TEÓRICA

3.1 REDES INDUSTRIAIS

A capacidade de comunicação entre dispositivos e o uso de mecanismos padronizados, abertos e transparentes são componentes indispensáveis do conceito de automação. A comunicação vem se expandindo rapidamente no sentido horizontal nos níveis inferiores, assim como no sentido vertical integrando todos os níveis hierárquicos.

As redes industriais são redes de comunicação dedicadas ao ambiente industrial, permitindo a troca de informação entre diferentes equipamentos. As vantagens alcançadas pelo uso das redes industriais como tecnologia são diversas, como: redução de instalações elétricas, redução do tamanho de painel elétrico, modularização de máquinas e equipamentos, diagnóstico local de falhas, diagnóstico real time em supervisório e flexibilidade na ampliação ou modificação [4].

3.1.1 MODBUS

Modbus é um protocolo de comunicação serial desenvolvido pela Modicon e publicado pela Modicon® em 1979 utilizado para comunicação entre dispositivos mestre-escravo/cliente-servidor.

Este protocolo define uma estrutura de mensagem que os controladores reconhecerão independentemente do tipo de rede sobre as quais eles se comunicam. Ele descreve o processo que um controlador usa para solicitar acesso a outro dispositivo, como ele responderá às solicitações dos outros dispositivos e como os erros serão detectados e relatados. Ele estabelece um formato comum para o formato e o conteúdo dos campos de mensagem [5].

Modbus é um dos protocolos mais utilizados em automação industrial, graças à sua simplicidade e facilidade de implementação, podendo ser utilizado em diversos padrões de meio físico, como: RS-232, RS-485 e Ethernet TCP/IP (Modbus TCP).

A velocidade de comunicação varia em cada um desses padrões, bem como o comprimento máximo da rede e o número máximo de dispositivos conectados.

3.1.1.1 MODELO DE COMUNICAÇÃO

O protocolo Modbus é baseado no modelo de comunicação mestre-escravo, onde apenas o único dispositivo mestre pode inicializar a comunicação e os demais dispositivos escravos respondem enviando os dados solicitados pelo mestre, ou realizam alguma ação solicitada. Os dados transmitidos podem ser digitais ou analógicos, ou seja, é possível enviar valores numéricos como temperatura e pressão ou enviar um *bit* para ligar e desligar um motor [5].

Ao utilizar o meio físico Ethernet, o protocolo Modbus opera com o mecanismo de controle de acesso CSMA-CD, que é próprio da rede Ethernet, com mensagens no modelo cliente-servidor.

3.1.1.2 MODOS DE TRANSMISSÃO

O protocolo Modbus pode ser configurado para trabalhar com um dos dois modos de transmissão disponíveis: American Code for Informastion Interchange (ASCII) ou Remote Terminal Unit (RTU), os quais definem como os dados serão empacotados na mensagem, estes modos são escolhidos durante a configuração dos parâmetros de comunicação, tais como: *baud rate*, paridade e *stop bits*.

Ao todo o protocolo Modbus possui 256 endereços onde:

- 0 - (zero) é o endereço de *broadcast*, quando o mestre envia uma mensagem para o endereço zero, todos os escravos recebem a mensagem;
- 1 até 247 endereços disponíveis para os escravos;
- 248 até 255 endereços reservados.

3.1.1.3 MODO RTU

No modo RTU, cada mensagem de 8 *bits* contém dois caracteres hexadecimais de 4 *bits*. A principal vantagem desse modo é que sua maior densidade de caracteres permite um melhor processamento de dados do que o modo ASCII para a mesma velocidade de comunicação. Cada mensagem deve ser transmitida em um fluxo contínuo de caracteres.

No modo RTU não existe um caractere específico que indique o início ou o fim de um telegrama. A indicação de quando uma nova mensagem começa ou quando ela termina é feita pela ausência de transmissão de dados na rede, por um tempo mínimo de 3,5 vezes o

tempo de transmissão de um *byte* de dados. Sendo assim, caso um telegrama tenha iniciado após a decorrência desse tempo mínimo, os elementos da rede irão assumir que o primeiro caractere recebido representa o início de um novo telegrama. E da mesma forma, os elementos da rede irão assumir que o telegrama chegou ao fim quando, recebidos os *bytes* do telegrama, este tempo decorra novamente.

Se durante a transmissão de um telegrama o tempo entre os *bytes* for maior que este tempo mínimo, o telegrama será considerado inválido, pois o controlador irá descartar os *bytes* já recebidos e montará um novo telegrama com os *bytes* que estiverem sendo transmitidos. O tempo para transmitir uma palavra varia de 573 μs para taxas de comunicação acima de 19200 bits/s e 9 ms para a taxa de 1200 bits/s.

O modo de mensagem RTU inclui um método de checagem de erro que é baseado no *Cyclical Redundancy Checking* (CRC). O campo de checagem de erro contém um valor de 16 *bits* com o resultado do cálculo de CRC sobre o conteúdo da mensagem.

3.1.1.4 MODO ASCII

No modo de transmissão ASCII, cada *byte* em uma mensagem é enviado como dois caracteres ASCII. Apesar de gerar mensagens legíveis pela tabela ASCII esse modo consome mais recursos da rede. A principal vantagem dessa modalidade é que permite intervalos de tempo de cerca de um segundo entre os caracteres sem causar erro, se ocorrer um intervalo maior, o dispositivo receptor assume que ocorreu um erro. O campo de checagem de erros é baseado no método *Longitudinal Redundancy Check* (LRC), em que a verificação ocorre ao final de cada bloco como se fosse transmitido um caractere extra ao final de cada bloco.

Os dispositivos monitoram constantemente a rede para o início de uma mensagem. Quando uma mensagem é iniciada pelo mestre, todos os dispositivos da rede decodificam o campo de endereço para determinar qual escravo deve receber a mensagem. O início de uma mensagem é reconhecido pelo caractere (:) "dois pontos".

3.1.1.5 DESCRIÇÃO DO PROTOCOLO MODBUS

O protocolo Modbus define uma Unidade de Dados de Protocolo simples (PDU) independente das camadas de comunicação subjacentes. O mapeamento do protocolo Modbus em barramentos específicos ou redes pode introduzir alguns campos adicionais na Unidade de

Dados de Aplicação (ADU) [6]. A estrutura da mensagem Modbus é apresentada na Figura 3.1.

A PDU e seu código formam a base da especificação do protocolo de aplicação Modbus. Essa especificação define o formato da PDU, os diversos conceitos de dados usados pelo protocolo, o uso de códigos de função para acessar esses dados e a implementação dos diversos códigos de função e as restrições específicas de cada um deles.

Figura 3.1: Estrutura da mensagem Modbus.

O campo de endereço contém dois caracteres ASCII ou oito *bits* RTU. Os dispositivos escravos individuais são atribuídos endereços no intervalo de 1 a 247.

O código de função de uma unidade de dados Modbus é codificado em um *byte*. Os códigos válidos estão no intervalo de 1 a 255 decimal (o intervalo de 128 a 255 é reservado e usado para respostas de exceção). O código de função "0" não é válido.

O campo de dados disponibiliza ao escravo alguma informação necessária pelo escravo para completar a ação específica pelo código da função. O dado é formado de *bytes* de caracteres múltiplos (um par de caracteres ASCII no modo ASCII) ou de dois dígitos hexadecimais no modo RTU, na faixa de 00h até FFh. Os dados tipicamente incluem registradores de endereços, contadores de valores e escrita de dados. Se nenhum erro é encontrado, o campo de dados da resposta do escravo retornará do pedido de dados. Se alguns erros ocorrem, o campo de dado retorna um código de exceção que a aplicação mestre pode usar para determinar a próxima ação a tomar.

Se nenhum erro é encontrado, o campo de dados da resposta contém os dados solicitados. Se ocorrer um erro relacionado com a função Modbus solicitada, o campo de dados contém um código de exceção que a aplicação do servidor pode usar para determinar a próxima ação a ser tomada.

3.1.1.6 PRINCIPAIS CÓDIGOS DE FUNÇÃO

Existem três categorias de códigos de funções Modbus [6]:

- Códigos de Função públicos

- Códigos de Função definidos pelo usuário
- Códigos de Função reservados

Os códigos de função públicos, apresentados na Figura 3.2, são códigos bem definidos, únicos, validados pela comunidade Modbus.org, documentados e possuem teste de conformidade disponível.

Figura 3.2: Códigos de Funções públicos

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**		**14**	6.14
			Write File record	**21**		**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Server ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19
			CANopen General Reference	**43**	**13**	**2B**	6.20

3.2 CONTROLADORES LÓGICOS PROGRAMÁVEIS

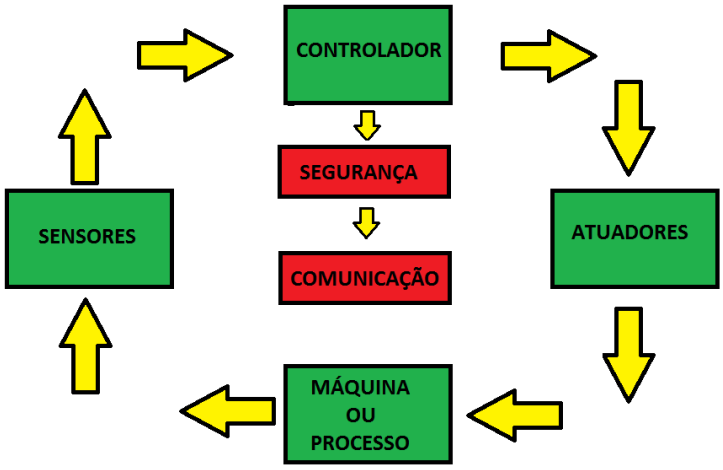
A National Electrical Manufacturers Association (NEMA) define um Controlador Lógico Programável como um aparelho eletrônico digital que utiliza uma memória programável para o armazenamento interno de instruções para implementações específicas, tais como lógica, sequenciamento, temporização, contagem e aritmética, para controlar, através de módulos de entradas e saídas, vários tipos de máquinas ou processos.

Os CLPs foram desenvolvidos na década de 60, com a finalidade de substituir painéis de relés que eram utilizados nas indústrias automobilísticas para executar controles baseados em lógicas combinacionais e sequenciais.

Atualmente, esses dispositivos são largamente utilizados na indústria no controle microprocessado de processos industriais, por serem facilmente programados via *software* por um operador, ao invés de utilizar relés e bobinas. As linguagens de programação permitem aos usuários se comunicarem com o CLP e definir as tarefas que o mesmo deverá executar. Pela normalização, esses controladores devem ter no mínimo três linguagens de programação: Ladder, Lista de Instruções e Diagrama de Funções.

A Figura 3.3 representa o ciclo de funcionamento de um sistema de automação genérico, onde podemos conferir as funções básicas do CLP.

Figura 3.3: Representação do ciclo de funcionamento de um sistema de automação



Esses controladores comunicam-se com diversos tipos de equipamentos e até mesmo com outros controladores. Essa versatilidade é importante, pois é possível a segmentação do controle de um processo entre diversos CLPs, e cada processo pode obter informação do outro através da comunicação entre os controladores.

Durante o estágio, o CLP M221 foi utilizado, um produto Schneider Electric, programado pelo *software*: SoMachine Basic. Ele se destina a substituir Twido Suite e Twido, antigos CLPs da Schneider Electric.

3.2.1 Estrutura de um CLP

Controladores lógicos programáveis podem ser do tipo compacto ou modular:

- Tipo compacto: integra o processador, fonte de alimentação, entradas e saídas. Dependendo dos modelos e fabricantes, pode executar algumas funções adicionais (contador rápido, E / S analógica ...) e receber extensões limitadas. Estes controladores de funcionamento simples são geralmente destinados a controlar pequenas aplicações.
- Tipo modular: o processador, a fonte de alimentação, e as interfaces E /S residem em unidades (módulos) e são fixados em um rack.

Um CLP geralmente é composto por: fonte, CPU, módulos de entrada, módulos de saídas, módulos de função e módulos de comunicação. Esses são separados e interconectados através de um chassi que fornece proteção, conexão elétrica e suporte mecânico. Desta forma, o CLP pode ser montado na medida exata dos requisitos do sistema a ser controlado.

A fonte permite fornecer a energia necessário para o funcionamento do CLP. De um 220V AC ou 24V DC fornecer fontes de tensão com as necessidades do controlador: 24V, 12V, 5V contínuo. Este módulo deve ter boas performances contra a rede de quedas e um transformador de isolamento para lutar contra distúrbios da mesma rede.

A CPU é o coração da unidade na unidade central, onde se realiza todas as funções lógicas, funções de temporizador, contagem e cálculo, a partir de um programa contido na memória baseado em microprocessador. É ligado a outros elementos (memória e Interface E/S) por ligações que transmitem informações em formato binário.

A área de memória é um elemento funcional que pode receber, armazenar e recuperar informações dos diversos setores do sistema.

Vários tipos de módulos E/S estão disponíveis:

- Digital: Estes módulos cobrem a informação que pode levar apenas dois estados (verdadeiro ou falso, 0 ou 1). Eles são a interface entre o PLC e os vários sensores e atuadores presentes.
- Analógico: Neste caso, o sinal processado é analógico e toma valores dentro de um determinado intervalo.
- Especializado: a informação processada está contida em palavras codificadas na forma binária ou hexadecimal. Este é o tipo de informações fornecidas por um computador ou um módulo inteligente.
- Módulos de comunicação: comunicação pode ser realizada com diferentes meios:
    - CANOpen,
    - Serial Line Modbus,

- USB port,
- Ethernet, etc…

3.2.2 AS LINGUAGENS DE PROGRAMAÇÃO

As linguagens de programação permitem aos usuários se comunicarem com o controlador e definirem as tarefas que deverão ser executadas. Existem 6 principais linguagens utilizadas para programar um CLP desenvolvido pela Schneider Electric:

- Diagrama Blocos Funcionais (Function Block Diagram – FBD)
- Texto Estruturado (Strutured Text – ST)
- Funções Gráficas de Sequenciamento (Sequential Function Chart - SFC)
- Funções Gráficas Contínuas (Continuous Function Chart - CFC),
- Diagrama Ladder (LD),
- Lista de Instruções (Instruction List – IL).

Durante o estágio, as linguagens LD, IL e SFC foram utilizadas no SoMachine Basic V1.5, e FBD e ST no SoMachine.

Na Schneider Electric, a programação de um CLP é muitas vezes feita com o SoMachine, Unity Pro ou SoMachine Basic. Quando uma aplicação é programada, ela é carregada para o controlador principalmente via USB ou Ethernet. Estes cabos são utilizados para a programação, mas também para a comunicação entre os controladores.

3.3 SOMACHINE BASIC

O SoMachine Basic é uma ferramenta de programação gráfica criada para facilitar a configuração, o desenvolvimento e a colocação em funcionamento de programas para controladores lógicos Modicon M221 [7].

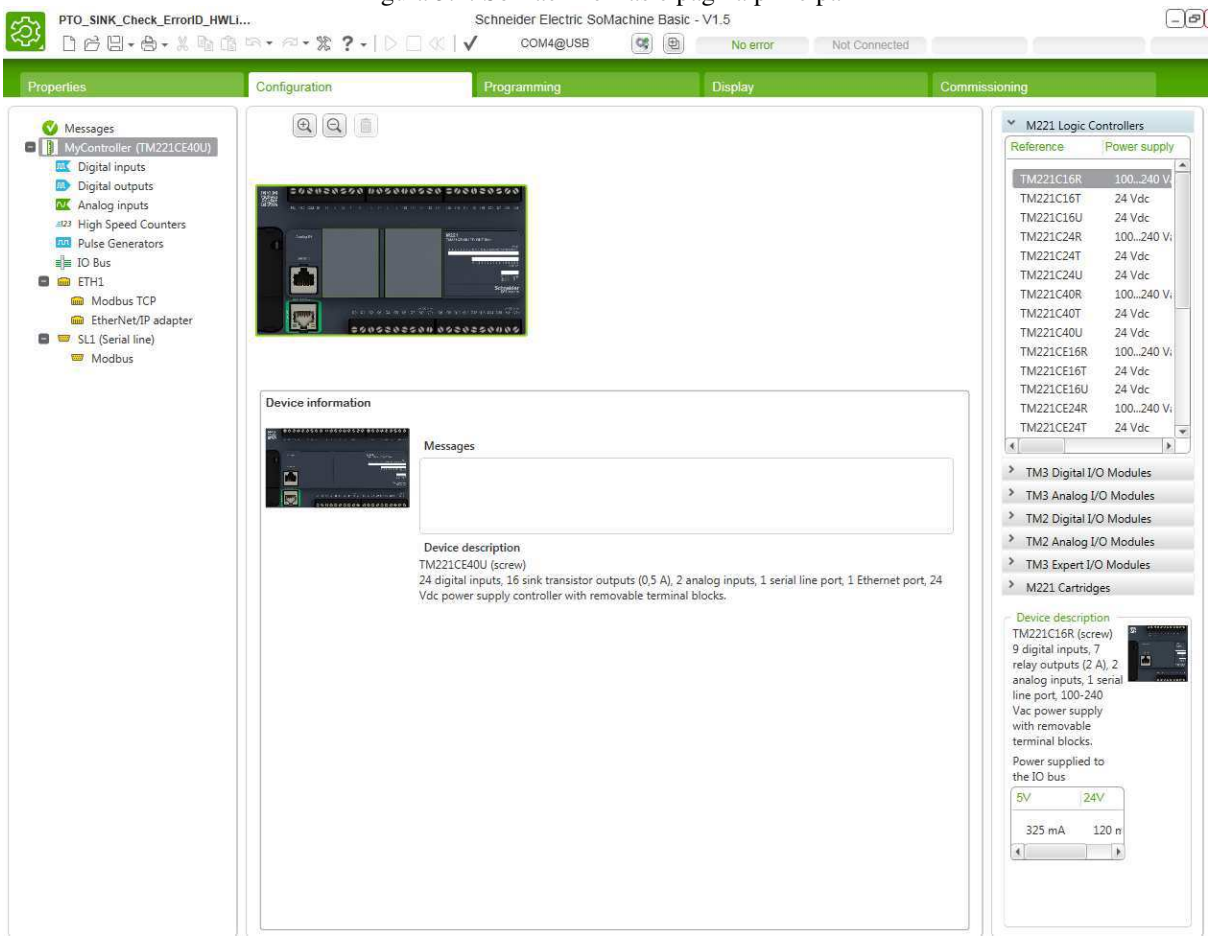
A janela principal está dividida em vários módulos. Cada módulo controla um estágio diferente do ciclo de desenvolvimento e está acessível clicando em uma guia na parte superior da área de módulos: Propriedades, Configuração, Programação, Exibição e Colocar em funcionamento, como apresentado na Figura 3.4.

Cada tela é dividida em 3 zonas: uma árvore de seleção, uma área de edição e um catálogo de referência de produtos organizado por gama.

O programa é organizado em Unidades Organizacionais do Programa (POUs) e seções. POUs são objetos reutilizáveis que contém uma declaração de variável e um conjunto de instruções utilizadas em um programa. Já as seções consistem em redes (*rungs*) para simplificar a leitura e a navegação dentro do programa.

O SoMachine Basic suporta as seguintes linguagens de programação da IEC-61131-3: Linguagem do Diagrama Ladder, Linguagem da lista de instruções, Grafcet (Lista) e Grafcet (SFC).

Figura 3.4: SoMachine Basic página principal



3.3.1 MODOS DE OPERAÇÃO

Os modos operacionais proporcionam controle para desenvolver, depurar, monitorar e modificar a aplicação quando o controlador está ou não conectado ao SoMachine Basic. O SoMachine Basic pode operar nos seguintes modos: *off-line*, *on-line* e simulador [7].

Em modo *off-line*, pode-se configurar SoMachine Basic para que corresponda aos componentes de *hardware* desejados e depois desenvolver a aplicação.

No modo *on-line*, é possível proceder para o *download* da aplicação para o CLP. SoMachine Basic sincroniza a aplicação na memória do PC com a versão armazenada no controlador lógico, permitindo depurar, monitorar e modificar a aplicação.

Em modo de simulador, não é estabelecida nenhuma conexão física com um controlador lógico. Em vez disso, SoMachine Basic simula uma conexão a um controlador lógico e os módulos de expansão para executar e testar o programa.

3.3.2 TAREFAS E MODOS DE VERIFICAÇÃO

SoMachine Basic possui os seguintes modos de busca para a tarefa mestre [7].

- Modo normal (modo autônomo): uma nova busca começa imediatamente após a busca anterior ter sido concluída.
- Modo periódico (periódico cíclico): uma nova busca começa somente após ter decorrido o tempo de busca configurado da busca anterior. Assim sendo, cada busca tem a mesma duração.

Os POUs estão associados a várias tarefas da aplicação: mestre, periódica e eventos.

- Tarefa mestre: tarefa principal do aplicativo. A tarefa mestre é controlada por uma verificação cíclica contínua (no modo de busca normal) ou especificando o período de busca de 1 a 150 ms (padrão 100 ms) no modo de busca periódica.
- Tarefa periódica: uma sub-rotina de curta duração processada periodicamente. As tarefas periódicas são configuradas especificando-se o período de busca de 1 a 255 ms (padrão 255 ms).
- Tarefa de evento: uma sub-rotina de duração muito curta para reduzir o tempo de resposta do aplicativo.

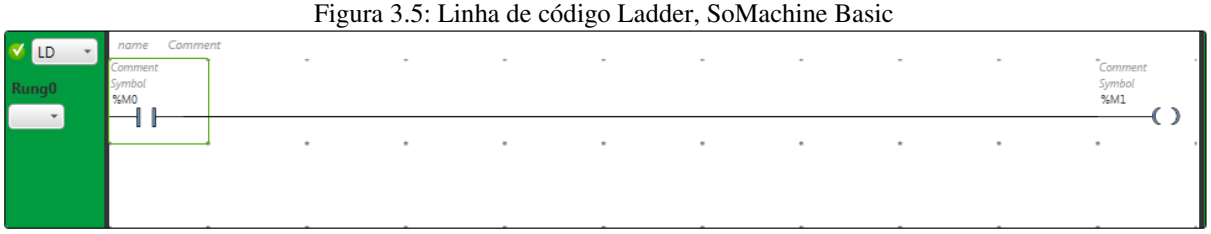
3.3.3 PROGRAMAÇÃO EM LINGUAGEM LADDER

A linguagem Ladder, também conhecida como lógica de diagrama de contatos, é o sistema de representação que mais se assemelha à tradicional notação de diagramas elétricos. O nome Ladder deve-se à representação da linguagem se parecer com uma escada, do inglês

ladder, na qual duas barras verticais paralelas são interligadas pela lógica de controle, formando os degraus, do inglês *rungs*, da escada. Os degraus são executados sequencialmente pelo controlador lógico. Portanto, a cada lógica de controle existente no programa da aplicação dá-se o nome de *rung*, a qual é composta por colunas e linhas, conforme apresentado na Figura 3.5.

As instruções em diagramas Ladder são inseridas arrastando e soltando elementos gráficos da barra de ferramentas que aparece acima do espaço de trabalho de programação em uma célula de grade.

Figura 3.5: Linha de código Ladder, SoMachine Basic



3.3.4 PROGRAMAÇÃO GRAFCET (SFC)

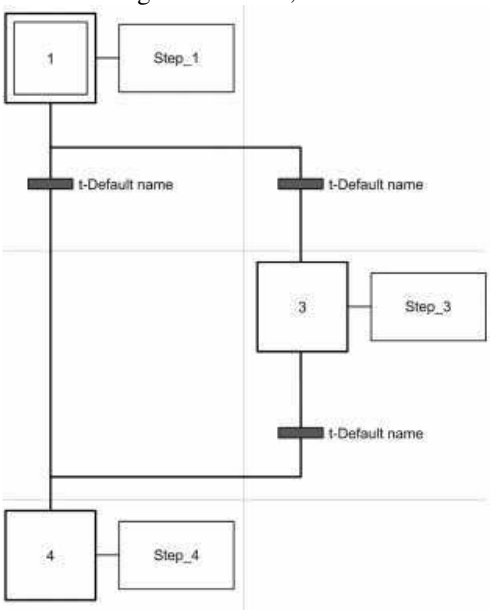
Grafcet (SFC) é uma linguagem de programação gráfica que descreve uma ordem cronológica de execução de tarefas discretas, conhecidas como etapas. A ordem pela qual as etapas são executadas é determinada pelas transições que conectam as etapas [7].

Um Grafcet (SFC) POU possui os seguintes componentes:

- Etapa: executa um conjunto de ações definidas em um ou vários *rungs*. As etapas podem ser:
    - Etapa inicial: executada no início do programa ou após um reinício do controlador. É representada por uma célula com uma borda dupla.
    - Etapa regular: etapas executadas condicionalmente após a conclusão da execução da etapa inicial.
- Transição: uma expressão booleana avaliada entre etapas. É a conexão entre duas ou mais etapas. A expressão booleana é definida em um único *rung* escrito em Ladder ou IL.

A Figura 3.6 apresenta um exemplo de um POU desenvolvido em linguagem Grafcet, no SoMachine Basic.

Figura 3.6: Código em Grafcet, no SoMachine Basic



3.3.5 COLOCAR EM EXECUÇÃO

Para colocar em funcionamento a aplicação criada, faz-se necessário realizar os seguintes passos:

1. Conectar-se ao CLP,
2. Baixar a aplicação no CLP (PC para Controlador),
3. Iniciar controlador.

A Figura 3.7 apresenta a janela Colocar em funcionamento, do SoMachine Basic.

Figura 3.7: Janela Colocar em funcionamento, no SoMachine Basic

3.4 AUTOIT

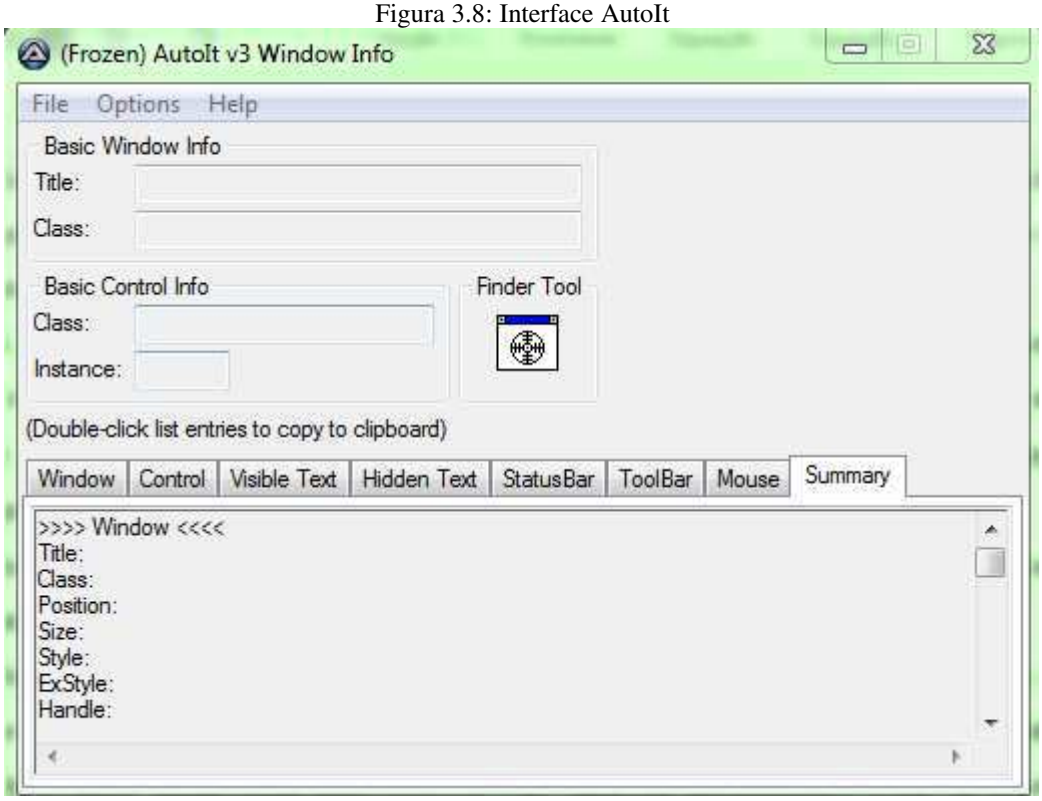
Esse *software* é a linguagem de *script freeware* que automatiza muitas funções sob o sistema operacional Microsoft Windows. Ele é capaz de agir automaticamente sobre os movimentos do *mouse*, teclado ou janelas do Windows para simular ações do usuário.

A princípio essa linguagem foi destinada a criar *scripts* de automação para tarefas altamente repetitivas, como a implantação e instalação de um grande número de PCs em uma rede. Com versões posteriores, AutoIt tem crescido para incluir melhorias tanto na concepção da linguagem de programação quanto nas funcionalidades gerais [8].

AutoIt pode compilar um *script* para criar um arquivo .exe utilizável sem o *software*. Os programas AutoIt tem extensão .au3 que pode ser editado com o IDE SciTE4AutoIt3.

Neste estágio, AutoIt é usado para implementar os testes 100% automatizados em SoMachine Basic. Desta forma, os *scripts* foram criados para abrir, fechar, baixar o aplicativo no SoMachine Basic, assim como para escrever e ler variáveis da memória (%M), do sistema (%S), entradas e saídas etc. A Figura 3.8 apresenta a interface que o AutoIt utiliza para o reconhecimento de variáveis no Microsoft Windows.

Figura 3.8: Interface AutoIt



3.5 TMS

TMS é um sistema de gerenciamento de teste que fornece a capacidade de criar um plano de teste abrangente, sendo eles testes automáticos ou manuais. É uma ferramenta interna Schneider Electric para estruturar os testes que são concebidos para atender às necessidades da empresa e seu processo de gerenciamento de projetos.

TMS pode associar um atelier a diferentes configurações e versões, o que é necessário para o processo de verificação/validação. Ele pode gerar automaticamente um relatório de resultado do teste, com cada versão e cada configuração, e também pode gerar o relatório para todos os testes com a mesma versão / configuração. O documento pode ser ligado a cada nível (Atelier / Categoria / Cenário / Caso de Teste) do plano de teste. Ele é usado para [9]:

- Definir bibliotecas para as etapas de testes reutilizáveis,
- Definir o plano de teste para testes manuais e automatizados para diferentes versões,
- Criar cenários de teste para os requisitos definidos no *Requisite Pro*,
- Definir e documentar as várias configurações de teste e versões de testes,
- Executar e documentar os Workshops dos testes,
- Criar e visualizar o estado dos IPR em *Clear Quest*.

TMS é utilizado para gerar o plano de teste, pois:

- TMS gera o relatório de teste automaticamente, com a mesma forma.
- TMS pode ter várias contas com direitos de acesso diferentes para diferentes planos de teste (workshop), o conteúdo é protegido contra as alterações.
- O progresso do projeto é muito claro para o gerente.
- A geração do relatório de ensaio é automática. Pode-se facilmente criar um relatório de todos os testes com a mesma versão ou um relatório de um workshop com todas as versões, que não é tão óbvio para conseguir com arquivos do Excel.
- Pode-se associar um documento a cada nível do Plano de Teste sem prejudicar a visão do plano.

4 OS TESTES AUTOMÁTICOS DE SOMACHINE BASIC

A principal razão para a automação dos testes é o tempo. O profissional que realizará os testes precisa de mais e mais tempo para verificar os novos componentes da aplicação e quanto mais a aplicação cresce, mais ele precisa de tempo para testar novamente as funcionalidades anteriores, o que pode tornar-se cansativo e o risco da ocorrência de erros humanos cresce.

Assim, o principal objetivo da automação dos testes é de reduzir o tempo das zonas de testes que já foram testados para o mesmo nível de qualidade. Evitando a perda de tempo, por exemplo, com lançamento de testes durante a noite para obter os resultados na manhã seguinte.

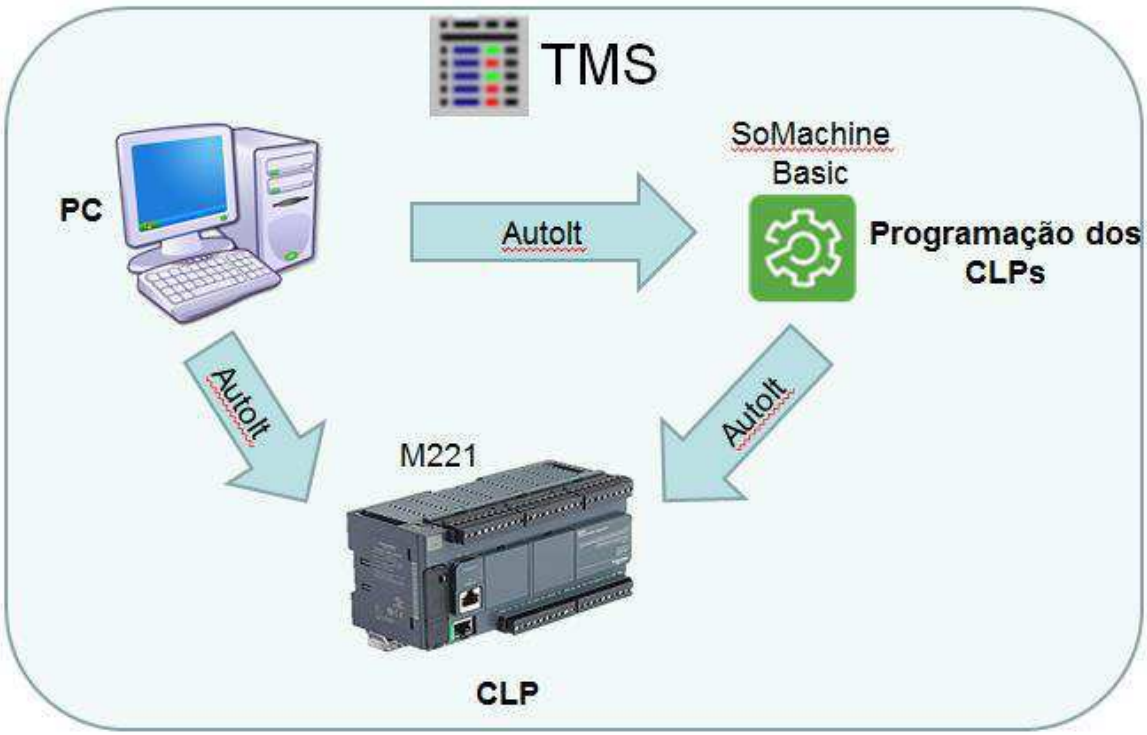
Mais uma razão que reforça a importância dos testes automatizados são os testes de regressão: cada vez que o *software* é modificado, é preciso se certificar de que o que funcionava antes ainda funciona.

Outra razão favorável para testes automatizados, é a possibilidade de executar antes de começar o teste manual e se a qualidade não é boa, rejeita-se a versão diretamente.

Desta forma, a cada versão de SoMachine Basic entregue durante a verificação e validação, existem cerca de 2.400 testes, dos quais 2.200 já são automatizadas.

Neste estágio, TMS foi utilizado para estabelecer os testes automatizados usando os *scripts* gerados em AutoIt. Todos os *scripts* implementados no AutoIt são encontrados na biblioteca do TMS. Assim, os testes automátizados podem ser construídos de acordo com as necessidades da aplicação. Em resumo, a Figura 4.1 apresenta ametodologia utilizada para a criação e colocação em serviço dos testes automatizados com os *softwares*: TMS, AutoIt, SoMachine Basic e o CLP M221.

Figura 4.1: Metodologia para realização dos testes.



5 ATIVIDADES DESENVOLVIDAS

A primeira etapa do estágio consistiu em compreender as aplicações dos testes existentes e, em seguida, identificar os possíveis pontos de melhoria.

As otimizações foram implementadas especialmente em aplicações SoMachine Basic e em alguns *scripts* TMS.

Nas aplicações SoMachine Basic, as principais alterações foram:

- Junção de pequenas aplicações. Desta forma, sempre que houver *scripts* para abrir, fazer o *download* e iniciar o controlador, ganha-se tempo, tendo em conta que há menos aplicativos que serão executados.
- Às vezes o teste no TMS é muito longo e é de difícil compreensão, então modificou-se a aplicação para ser mais complexa, deixando TMS mais simples.
- Melhorar os comentários para que as pessoas possam entender melhor o teste.
- Criação de testes que ainda não foram implementados.
- Adaptação de algumas aplicações que precisam ser atualizados devido a mudanças na nova versão do *software*, tais como o número de variáveis persistentes.

Nos *scripts* TMS, as principais alterações foram:

- Correção das variáveis que estão sendo testadas.
- Adaptação dos scripts *wait*, para esperar apenas o tempo necessário.
- Reescrita de alguns *scripts* muito longos.

Na segunda parte, ao contrário de outros recursos que já são automatizados, realizou-se a integração da validação do comportamento dos LEDs em um programa de testes automatizados para monitorar seu desempenho. Esta ferramenta foi desenvolvida durante um estágio anterior, mas ainda não era adequado para os CLPs do tipo M221ME16R e M221ME16T.

5.1 CONFIGURAÇÃO MATERIAL

6 modelos de CLP são utilizados na plataforma de testes automatizados do M221:

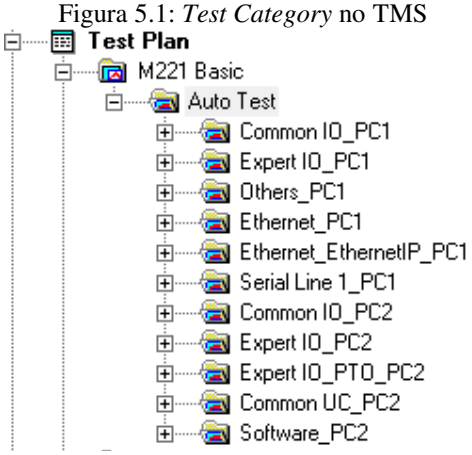
- TM221CE40U, TM221CE24U, TM221CE16U: para testar o gerador de frequência e 4 canais PTO (Pulse Train Output) (TM221CE40U é usado para a geração de impulsos e TM221CE24U + TM221CE16U são utilizados para contar os impulsos e medir a frequência com HSC (High Speed Counter)).
- TM221CE40T TM3 com 5 módulos e 1 cartucho: para realizar testes em módulos TM3, os eventos desencadeados pelo HSC e pelas entradas, blocos de funções de pulso e HSC.
- TM221ME16T / L com 14 módulos TM3 e 4 módulos TM2: para testar a HSC (todos os tipos: monofásico, bifásico, de frequência), PWM (Pulse-Width Modulation), a comunicação série (os modos de operação, a taxa de transmissão, consultas e blocos de função), os módulos TM3 e TM2, o bloco de função pulso, funções de comparação, as funções complexas, variáveis persistentes, as transições de estado do controlador, as funções básicas, palavras de sistema e bits, os blocos de função, modos de operação e Grafcet.
- TM221CE16T: para testar os 4 canais PTO.

5.2 OS TESTES EXISTENTES

Para executar um teste com TMS, deve-se selecionar a aba *Test Results* que está relacionada a um Plano de Teste. Dentro do Plano de Teste, tem-se a *Test Area* neste caso: M221 Basic. Depois o *Workshop*: *AutoTest* e, finalmente, os *Test Category*.

Os testes criados para SoMachine Basic versão 1.5 estão distribuídos em 11 *Test Categories,* como apresentado na Figura 5.1. Os testes desse Workshop são totalmente automatizados.

Figura 5.1: *Test Category* no TMS



5.3 ORGANIZAÇÃO DE UM TESTE EM TMS

Em cada teste, ao menos três instruções são utilizadas, como apresentado na Figura 5.3:

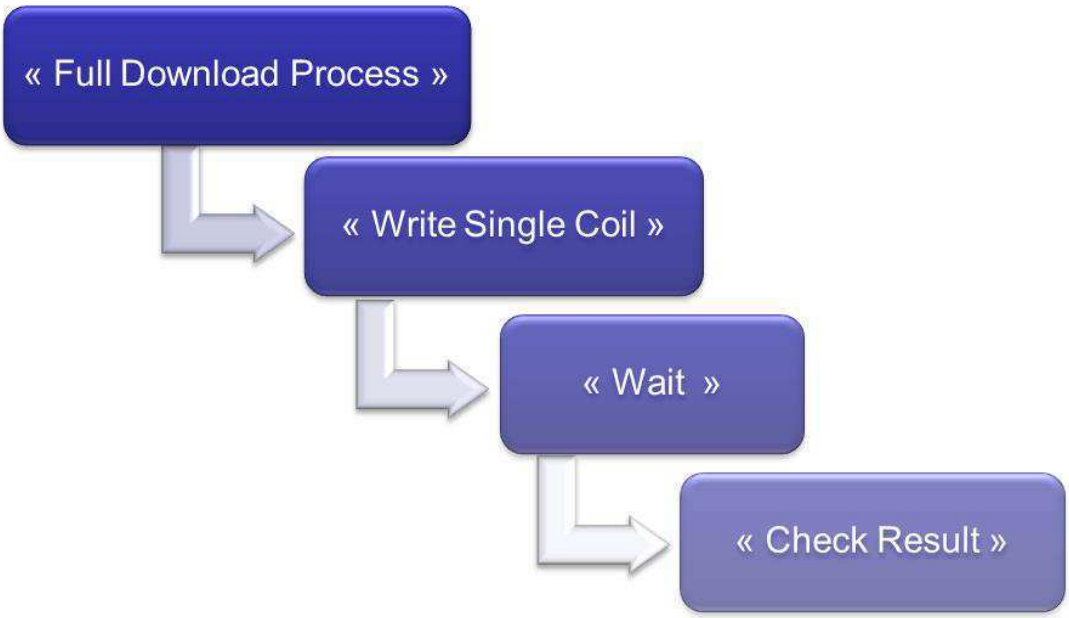
- *Full Download Process* para fazer o download da aplicação. Este *script* executa essencialmente: abertura o *software* SoMachine Basic, abertura do projeto, conexão ao CLP e download da aplicação.
- *Write Single Coil* ou *Multiple Coils* para iniciar o teste. Este *script* escreve em uma variável da memória (% M) um valor (1 ou 0). Às vezes, os testes podem começar ao escrever um valor na palavra de memória (% MW), isso depende de como a aplicação foi desenvolvida. Por exemplo, a Figura 5.2 mostra a configuração de uma instrução de escrita em uma única bobina. Neste caso, devemos colocar o endereço IP do controlador, o número da variável que será alterada, o valor enviado para a bobina e um comentário para explicar o propósito desta instrução.
- *Check Result* para verificar se o teste foi em sucedido. Este *script* lê os valores armazenados em alguma variável do controlador.

Figura 5.2: Parâmetros da instrução Write Single Coil, no TMS

Parameters

Name	Data Type	Value	Comment
PLC IP address	STRING	85.221.15.20	IP address of the equipement to be tested(e.g. 85.17.60.52).If your device have an Unid Id different than 255, par exemple Unid ID 121 use 85.17.60.52:121
Coil to Write	WORD	5	Coil that will be written in the PLC, the value is in decimal format (e.g. 2)
Value to send	BINARY	1	0= OFF, 1=ON (e.g. 1)
Step Name	STRING	Start PLS0 at 1hz	Text that will appear when this step is executed

Figura 5.3: Principais instruções utilizadas em um teste.



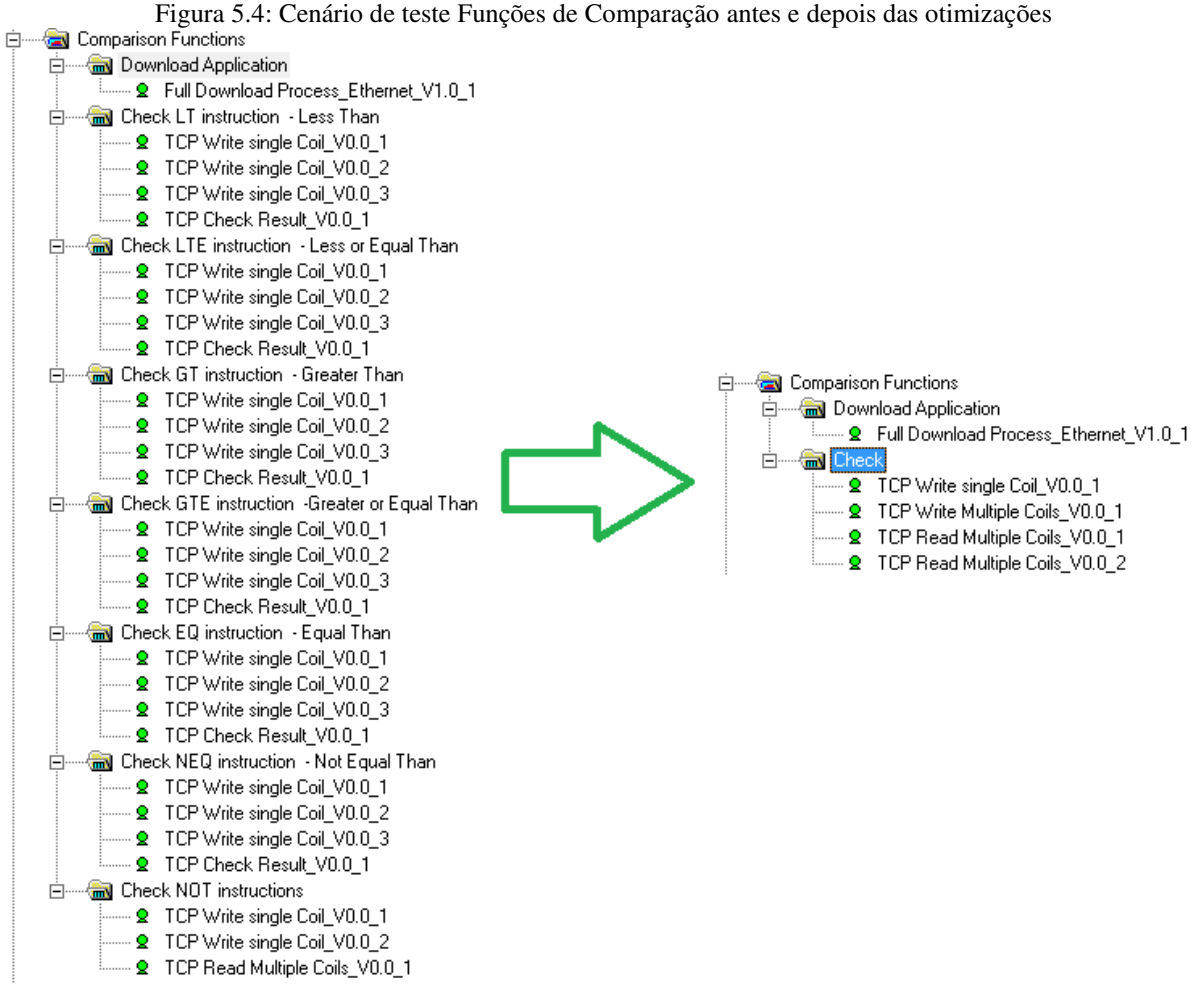
5.4 OTIMIZAÇÕES EFETUADAS

Esta seção apresentará o detalhamento das otimizações que foram realizadas em cada cenário de teste.

5.4.1 FUNÇÕES DE COMPARAÇÃO

Neste cenário, modificou-se a aplicação para permitir um código mais conciso no TMS, como apresentado na Figura 5.4. Os testes podem ser executados ao mesmo tempo, e pode-se conferir os resultados de vários testes utilizando um único *script*. Desta forma, é mais simples a compreensão e o tempo de execução do teste é reduzido.

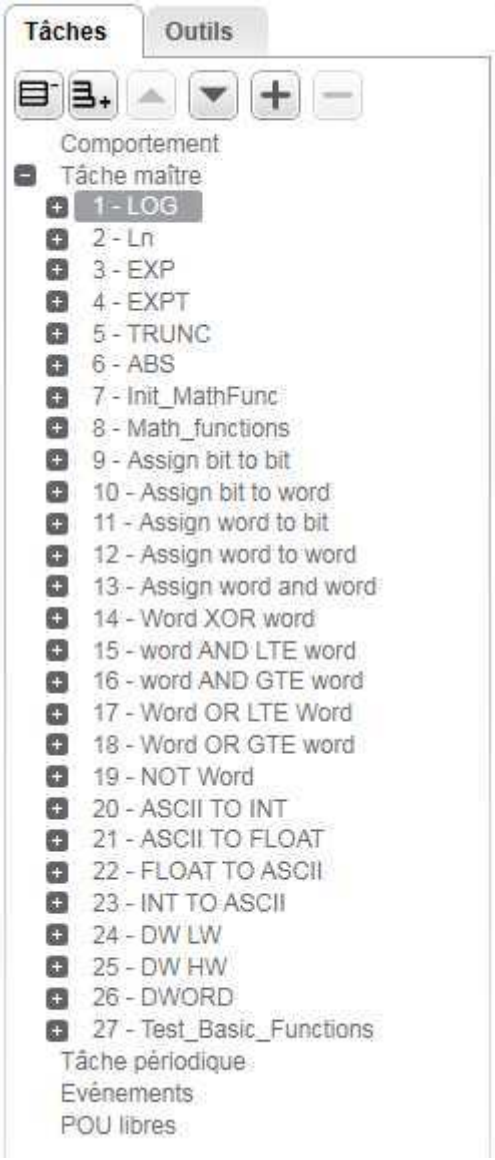
Figura 5.4: Cenário de teste Funções de Comparação antes e depois das otimizações



5.4.2 FUNÇÕES COMPLEXAS

Neste cenário, mudanças significativas foram realizadas, pois antes haviam 27 aplicações. Então, criaram-se três novas aplicações que reúnem as 27 existentes anteriormente. Por exemplo, a Figura 5.5 apresenta os diversos POUs existentes em uma aplicação. Portanto, economiza-se tempo, tendo em conta o fato de que há menos aplicações a serem baixadas para o CLP.

Figura 5.5: Aplicação em SoMachine Basic, do cenário de teste Funções Complexas



5.4.3 SYSTEM BIT E SYSTEM WORD

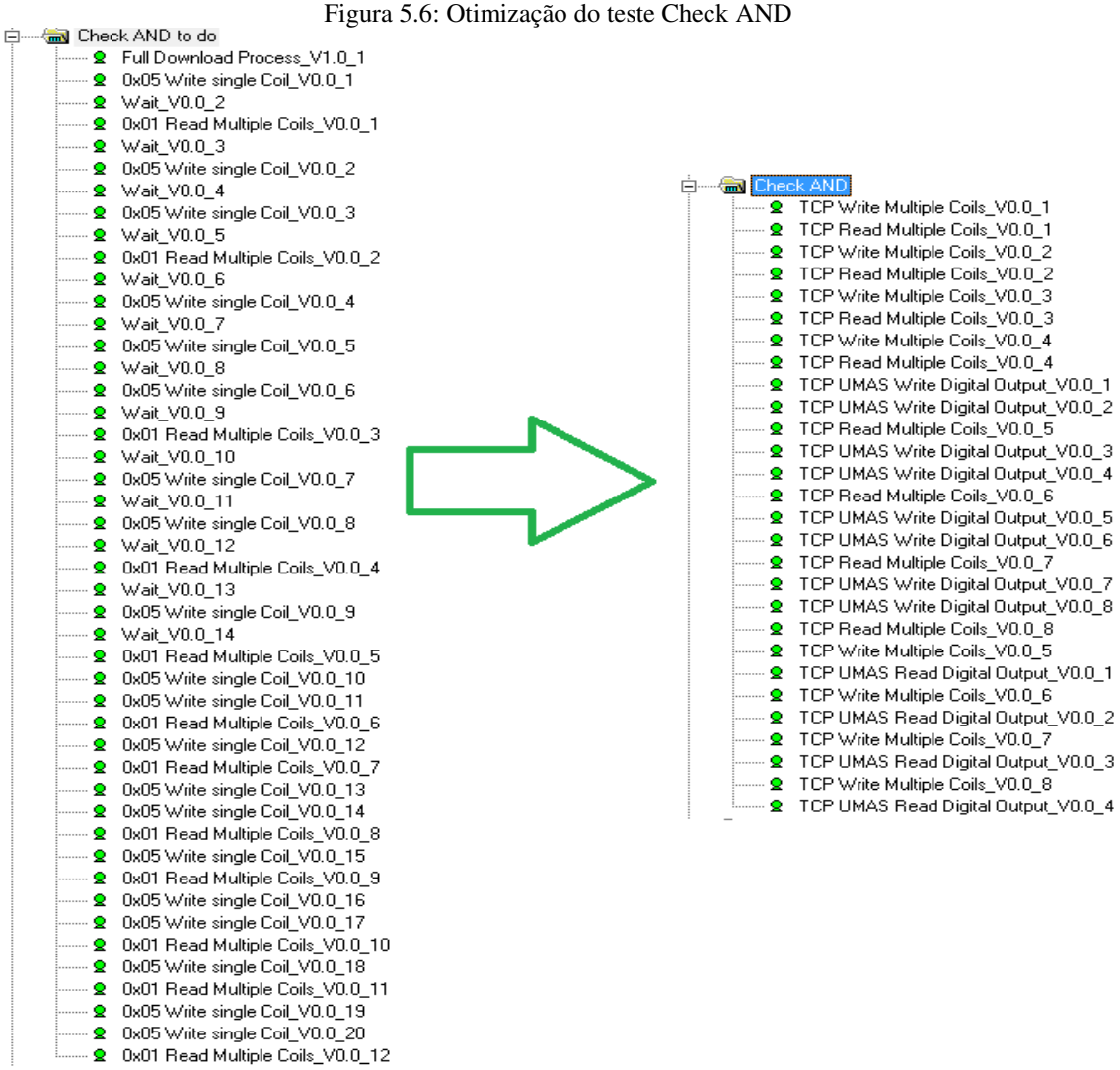
No cenário de teste System bit, criou-se uma aplicação que reúne 10 outras aplicações existentes. Como também, testes para verificar o funcionamento do *bit* do sistema %S75 que indica o *status* da bateria.

No cenário System Word criou-se uma aplicação que reúne 9 outras aplicações existentes.

5.4.4 FUNÇÕES BÁSICAS

Neste cenário, criaram-se duas aplicações que reúnem 15 aplicações existentes. Também foi aprimorado os scripts em que TMS era demasiado longo e de difícil compreensão, como apresentado na Figura 5.6.

Figura 5.6: Otimização do teste Check AND



5.4.5 BLOCOS DE FUNÇÃO

Neste cenário, criaram-se duas aplicações que reúnem 19 outras aplicações existentes. Também testes foram implementados para verificar o funcionamento do bloco de função *Step Counter*.

5.4.6 START IN RUN, START IN STOP, START IN PREVIOUS STATE E START IN UNCODITIONAL START

Alguns desses cenários de testes foram inseridos no teste de variáveis persistentes. Assim, estas quatro categorias foram apagadas e uma nova categoria foi criada: *PLC in Halt*, para testar todos os modos quando o PLC está em estado de "parada".

5.4.7 SIMULADOR

Este cenário foi otimizado com todas as novas melhorias feitas na "Categoria Teste UC_PC2 Comum" descritas acima. E com a finalidade de ser mais organizado, este cenário está agora dividido nas seguintes partes: "Test Simulator_General", "Sistema Bit_Word", "Blocos de Função" e "Funções Complexas".

5.4.8 CHECK_FC_BRICK, MAX CONF E TEST WITH LEXIUM

No cenário Check_FC_Brick, foram criadas duas aplicações que reúnem outras 4 aplicações existentes. Como também, a validação dos testes foi aprimorada.

No cenário MAX Conf, criou-se uma aplicação para testar todas as entradas e saídas dos 14 módulos de expansão.

No cenário Test com Lexium, criaram-se três aplicações que reúnem 8 outras aplicações existentes.

5.4.9 SERIAL LINE

Neste cenário, foram criadas três aplicações que reúnem 5 outras aplicações existentes. Também foi modificada a aplicação do M258 CLP (mestre) para executar menos ciclos do que antes, porque não há necessidade de repetir o ciclo 200 vezes.

5.4.10 Novos Testes

Novos testes automatizados foram desenvolvidos, pois percebeu-se que alguns blocos de funções, modos de operação e bits do sistema não estavam senso testados. Dessa forma implementou-se os seguintes teste, como apresentado na Figura 5.7: Step Counter, Max Conf 14, System Bit 75, PLC in Halt e um para testar o comportamento dos LEDs, que será apresentado na próxima seção deste relatório.

Figura 5.7: Novos testes implementados.

5.5 Reconhecimento do Comportamento dos LEDs

Na fase de validação dos CLPs, a performance dos LED indicadores é também um critério importante a se testar, mas esta etapa de validação, solicita a vigilância constante de um testador.

No quadro do projeto, um sistema de visão industrial desenvolvido em um estágio anterior, com a finalidade de verificar o comportamento dos LEDs, foi utilizado.

5.5.1 Interface Vision Launcher

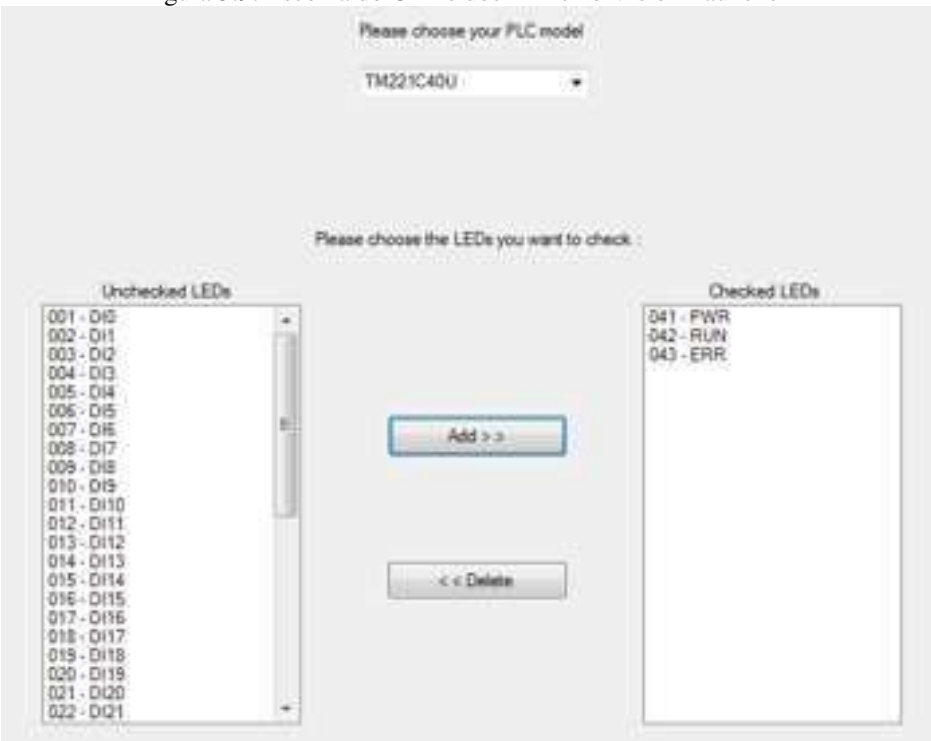
Antes de executar os testes, é preciso configurar a interface "Vision Launcher", desenvolvida no estágio anterior, como segue:

- Selecione a região de interesse, como apresentado na Figura 5.8.

Figura 5.8: Escolha da Região de Interesse

- Escolha o modelo CLP agora com opções M221ME16R e M221ME16T, e os LEDs que serão monitorados, como apresentado na Figura 5.9.

Figura 5.9: Escolha do CLP e dos LEDs no Vision Launcher



- Inicie a detecção de posição dos LED para identificar todos os LEDs, como apresentado na Figura 5.10.

Figura 5.10: Detecção dos LEDs no Vision Launcher

- Inicie o teste.

Se o aparelho for instalado em um ambiente que é muito escuro para se detectar todas as posições dos LEDs, o programa irá terminar a execução e apresentará o erro.

5.5.2 APLICAÇÃO SOMACHINE BASIC

Para testar os possíveis estados dos LEDs em um controlador, foi desenvolvida uma aplicação, no SoMachine Basic.

No primeira POU dessa aplicação, apresentado na Figura 5.11, através da variável %M0, o controlador ficará em estado *halt* e através de %M1, todas as saídas serão ligadas. No segundo POU, apresentado na Figura 5.12, testa-se a comunicação serial, através de um bloco funcional "READ_VAR", a fim de observar o LED SL (Serial Line).

Figura 5.11: Primeiro POU, aplicação LED

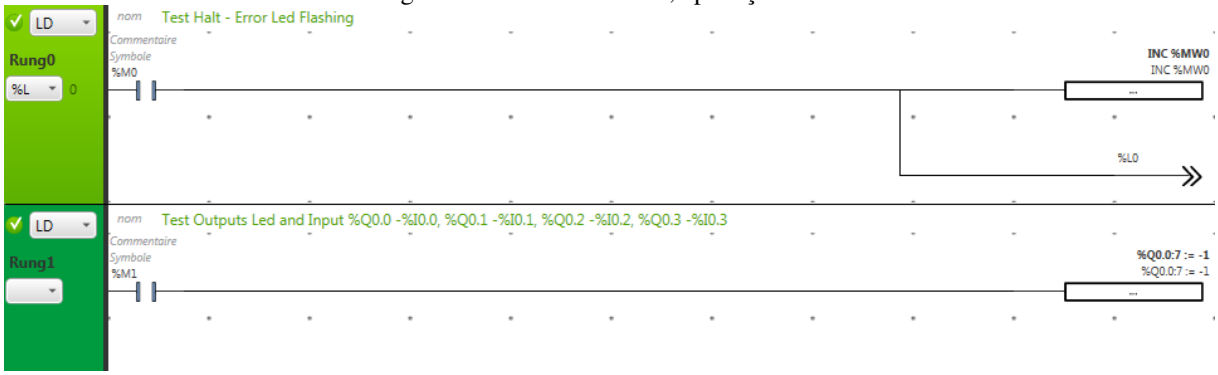


Figura 5.12: Segundo POU, aplicação LED

5.5.3 TESTE LED EM TMS

Em TMS, foi estabelecido o teste para verificar o comportamento dos LEDs nos modelos M221ME16T e M221ME16R.

A primeira instrução *Full Download Process* é a aplicação usada pelo escravo Modbus.

O primeiro diagnóstico é após a instrução *Full Download Process* 2, neste caso, os LEDs de *Power* e *Run* são verificados e devem estar ligados (ON).

O segundo diagnóstico é após a declaração "Write Single Coil 1". Então, como explicado na seção 6.5.2 do presente relatório, os LEDs de todas as saídas devem estar ligados. Depois, um comando *Stop* é executado e o LED *Power* deve ser ligado, mas o *LED Run* pisca.

Em seguida, o controlador retorna ao estado *Run* para executar a próxima instrução que serve para verificar a comunicação serial. Assim, o LED SL está ligado.

Após a variável %M0 ser acionada, o controlador está em estado *Halt*. Neste caso, o ERR LED (error) é ON, o LED *Run* está piscando.

Finalmente, uma inicialização é executada no controlador e um diagnóstico final é feito: o LED *Power* está ligado e o LED *Run* está piscando. A Figura 6.8 apresenta a sequência de *scripts* executado no TMS afim de realizar o teste dos LEDs.

6 RESULTADOS

Durante o estágio, foram realizados as otimizações nos seguintes cenários de teste:

Cenários de Teste	N App Antes	N App Depois	N App Criadas	Tempo antes Otimização	Tempo depois Otimização
Complex Functions	27	3	3	26m46s	4m22s
System Bit	14	7	1	22m11s	14m31s
System Word	21	12	1	34m0s	24m20s
Basic Functions	15	2	2	33m22s	10m49s
Function Blocks	21	5	2	51m37s	36m20s
Software	63	18	18	1h4m5s	57m47s
Check_FC_Brick_PLC20	4	2	2	7m0s	3m46s
Start in Run	2	0	0	4m23s	-
Start in Stop	2	0	0	4m20s	-
Start in Previous State	2	0	0	4m19s	-
Start in Unconditional Run	2	0	0	4m38s	-
PLC in Halt	-	8	8	-	11m48s
Max Conf	1	1	1	1m11s	2m19s
Source_Test with Lexium	11	6	3	36m0s	29m39s
Serial Line FB	14	10	1	1h24m21s	1h21m41s
Serial Line 1_PC1	32	32	-	2h57m51s	2h33m17s
Total	231	106	42	9h16m4s	7h8m37s

Deste modo, depois das otimizações realizadas, o tempo dos testes foi reduzido para 7 horas, 125 aplicações foram reduzidas e 42 novas aplicações foram desenvolvidos. Assim, otimizou-se 22% do tempo de teste total.

Todas as otimizações foram revistas pelo tutor do estágio da Schneider Electric, na qual se confirmou a cobertura completa dos testes após a otimização e ainda verificou-se a expansão dos casos de testes com o incremento dos seguintes testes: Step Counter, Max Conf 14, System Bit 75, PLC in Halt e um para testar o comportamento dos LEDs.

7 CONCLUSÃO

Durante os 6 meses de estágio na Schneider Electric, pode-se descobrir o campo de trabalho de automação, principalmente sobre os testes de validação e verificação.

Além disso, compreendeu-se que o teste automatizado é uma ferramenta poderosa para garantir a não regressão do *firmware* e testar a compatibilidade com versões mais antigas do programa. Embora ele não possa testar todas as funções, muitos elementos essenciais, especialmente do *firmware* são verificados – reduzindo uma grande parte da carga de trabalho dos testadores.

De um ponto de vista técnico, trabalhou-se na otimização dos testes automatizados, incluindo competências sobre o funcionamento de CLPs (gama Machine Solutions), a programação de *software* (SoMachine Basic) e ferramentas de automação (AutoIt e TMS) e linguagens de programação, em que pode-se aplicar os conhecimentos adquiridos durante a formação na UFCG e no INSA Lyon.

Finalmente, um dos aspectos mais importante, foi a percepção mais concreta da profissão de engenheiro do setor de automação.

8 REFERÊNCIAS

[1] SCHNEIDER ELECTRIC. **Our company**. Disponível em: < https://sdreport.schneider-electric.com/en/company-at-issue/>. Acesso em: 01 de junho 2017.

[2] SCHNEIDER ELECTRIC. **Notre histoire**. Disponível em: <http://www.schneider-electric.fr/fr/about-us/company-profile/history/>. Acesso em: 01 de junho 2017.

[3]SCHNEIDER ELECTRIC. **Solutions Overview**. Disponível em: <http://sdreport.schneider-electric.com/en/solutions-customers-overview/>. Acesso em: 01 de junho 2017.

[4] SANTOS, L., 2016. **As vantagens clássicas para Redes Industriais.** Disponível em: <http://blog.murrelektronik.com.br/redes-industriais-revendo-vantagens/>. Acesso em: 13 de junho de 2017.

[5] National Instruments. 2014. **Étude approfondie du protocole Modbus**. Disponível em: <http://www.ni.com/white-paper/52134/fr/>. Acesso em: 18 de junho de 2017.

[6]**Modbus Application Protocol**. Disponível em: <http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf>. Acesso em 15 de abril de 2017.

[7] Schneider Electric. 2016. **SoMachine Basic Guia de Instruções**. Disponível em: <www.schneider-electric.com> Acesso em: 15 de junho de 2017.

[8] **AutoIt Online help.** Disponível em: <http://www.autoitscript.com/autoit3/docs/>. Acesso em 15 de abril de 2017.

[9] **TMS Online help**. Acesso em 16 de maio de 2017.

9 ANEXO 1: BIMONTHLY REPORT

Bimonthly report n°2

Automated Tests
20/03/17~31/03/17

Internship summary:

Subject: Optimization of the SoMachine Basic – M221 automated tests

Description: The main goal of this internship is to optimize the existing automatic tests on SoMachine Basic, and to integrate a tool used to diagnose LEDs behavior which was developed in a previous internship if timing permits. Once finished, the time needed to run the acceptance tests will be significantly reduced and the tests will be improved (and also to ensure the non-regression). In the process of my project, I'm going to understand the existing tests applications, identify potential improvements and offer a documentation for the completed work.

Project organization: (For a totality of 25 weeks)

Tasks	Planning& Status
Study of SoMachine Basic and controller M221	1/3 weeks, 33% done
Study of M221 automated tests and its associated tools/scripts	1/2 weeks, 50% done
Analysis and optimization of existing tests	3/(13 to 15) weeks
Integration of LEDS behavior tests to M221 auto tests	0/ (2 to 4) weeks
Documentation	0/2 weeks

Detailed missions:

- Learn to use SoMachine Basic (SoMB) and M221
- The organization and process of verification/validation of Machine Solution Products
- Get familiar with the automated tests and tools TMS (Test Manager System)
- Communication protocol : Modbus Serial/TCP
- Understand the existing tests applications
- Identify potential improvements in the applications
- Integrate LEDs behavior diagnostic tool to existing automated tests

Fourth Week: SoMachine Basic Applications and TMS

During the fourth week, I started to analyze SoMachine Basic applications that are made for testing the Function Blocks. I created a new application that brings together others 14 applications (Scheduler / TM TON/ Timer TP / Timer TOFF TB 10ms Periodic / Drum periodic). Then, I also wrote a new application that assembled the tests of the following function blocks: FIFO, LIFO, Check FB DRUM, Check FB DRUM periodic, Check SBR, Counter Up Down, Step Counter.

After, I began to understand the System Bit (%S) and System Word (%SW) applications. I created a new application that brought together others 10 for the system bit and another one that brought together 9 applications for System Word. In this folder, there are some codes that contain Infinite Jumps, so there were some complications when I tried to put together because the PLC goes to the state Halted.

Fifth Week: Electrical Habilitation and section Software at TMS

This week, I did the course of electrical habilitation for three days. The course was about safety prescriptions for low voltage electrical order operations.

Then, I analyzed the section M221 Basic>Auto Test> Software_PC2> Simulator_General Test. In order to organize this section, I created the following Tests Scenarios: System Bit_Word, Function Blocks and Complex Functions. For these tests, I adapted the applications that were optimized at Common UC_PC2 section.

Planning for the following weeks:

- Begin the tests optimization on:

Operating Mode (starting mode), and Persistent variables	2 weeks
Common IO (embedded, TM2/TM3 modules)	2 weeks