BOSCH
Invented for life

Bosch Engineering Latin America

DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Pesquisa e desenvolvimento de CLP para automação predial utilizando ECUs para veículos a gasolina.

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação na disciplina
DAS 5511: Projeto de Fim de Curso

Mauricio Reck

Curitiba, fevereiro de 2014

Pesquisa e desenvolvimento de CLP para automação predial utilizando ECUs para veículos a gasolina.

Mauricio Reck

Esta monografia foi julgada no contexto da disciplina
DAS5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Prof. *Marcelo Stemmer*

_____
Assinatura do Orientador

Banca Examinadora:

Eduardo Kamaroski Neto
Orientador na Empresa

Prof. Marcelo Stemmer
Orientador no Curso

Prof. <nome do professor avaliador>
Avaliador

<nome aluno 1>
<nome aluno 2>
Debatedores

Dedicado ao meu avô, Antônio Reck.

Agradecimentos

Aos meus pais, Mozarte e Maritê, que mais do que proporcionar uma ótima infância e vida acadêmica, formaram as principais características e valores que hoje carrego comigo. Obrigado por serem de todas as maneiras a minha referência e por estarem sempre presentes na minha vida. Agradeço ao meu irmão, Marcelo, que me ensina a cada dia como ser uma pessoa melhor e por ser meu amigo e companheiro a vida toda. Agradeço à minha família por me oferecerem a oportunidade de viajar a diversos lugares do mundo, ponto fundamental para minha formação.

À minha namorada, Lis, que é, e sempre foi, meu porto seguro, minha companheira de tantos anos e minha maior crítica. Agradeço por ser minha referência pela busca ao aprendizado, por sempre acreditar em mim e pela inquestionável vontade de fazer coisas novas e diferentes a cada dia. Obrigado por tudo que representa na minha vida.

Ao meu grande amigo Felipe Biru Finger, pelo companheirismo e parceria durante esses 5 anos, pelos ensinamentos, conversas e sessões de surf. Agradeço por todos os momentos que passamos juntos, pois desempenharam papel fundamental na minha vida, tanto profissional como pessoal. Um agradecimento especial ao meu amigo, Gustavo Fontes, por ser um exemplo de pessoa ética, de caráter e por tudo que nossa amizade representa para mim.

Aos meus colegas de trabalho da BEG pelos ensinamentos, orientação profissional e por estarem sempre disponíveis para qualquer dúvida. Em especial ao orientador, Eduardo Kamaroski, que nunca mediu esforços para me auxiliar, ensinar e guiar durante o tempo de projeto, e é um exemplo de profissional a ser seguido.

Agradeço a todos os amigos da UFSC que são muitos e sempre dividiram as angústias, dúvidas e felicidades. Desde os amigos de curso até os companheiros do voleibol, agradeço por fazerem parte de minha graduação, e por se tornarem os amigos que sempre levarei comigo.

Um agradecimento a todos os professores dos cursos aos quais frequentei na UFSC, em especial ao professor Marcelo Stemmer que dedicou parte de seu tempo a orientar-me neste trabalho.

Agradeço a UFSC por oferecer a possibilidade de vivenciar uma diversidade de conhecimento e de culturas que teve grande participação na formação de meu caráter e modo de pensar.

A todos o meu muito obrigado.

Resumo

O presente trabalho foi desenvolvido na empresa Robert Bosch em Curitiba, Paraná, na área de desenvolvimento de produto customizado chamada *Bosch Engineering Group* (BEG). A BEG é uma prestadora de serviços de engenharia de alta tecnologia, é 100% subsidiária do Grupo Robert Bosch e tem como objetivo oferecer adaptações customizadas para alguns sistemas, de acordo com as necessidades do cliente. Este projeto visa responder a uma demanda proveniente de uma divisão da Bosch na Colômbia (Security Systems), que tem a intenção de realizar a automação predial de edificações de médio porte. Atualmente, a Bosch possui somente soluções baseadas em CLPs para grandes aplicações como aeroportos, estádios e complexos prisionais. O objetivo do projeto é desenvolver um CLP para automação predial. Tal CLP deve utilizar como hardware uma unidade de controle eletrônico (ECU) para veículos movidos à gasolina, fabricadas em série pela Bosch e com baixo custo. Para tanto, foram realizados estudos sobre a solução utilizada atualmente; estudos para planejamento da solução proposta; bem como os trabalhos efetivos de retirada de partes automotivas da ECU, e desenvolvimento do Software de Configuração Inicial de CLPs (PICS). Os resultados obtidos nessa primeira etapa de validação do conceito foram satisfatórios e garantiram o futuro andamento do projeto.

Palavras-chave: Automação predial; controladores lógicos programáveis; unidade de controle eletrônico; software de configuração inicial de CLPs.

.

Abstract

The present work was developed at the company Robert Bosch in Curitiba, Paraná, in the area of customized product development called Bosch Engineering Group (BEG). BEG is a service provider of high technology engineering, is 100% subsidiary of Robert Bosch Group and aims to provide customized systems for some adjustments in accordance with customer needs. This project aims to respond to a demand from a division of Bosch in Colombia (Security Systems), which intends to carry out building automation of midrange buildings. Currently, Bosch has only applied solutions based on PLCs (Programmable Logic Controllers) for large applications such as airports, stadiums and prison complex. The project objective is to develop a PLC for building automation. The PLC needs to use a hardware from an electronic control unit (ECU) for gasoline powered vehicles manufactured in series by Bosch and with low cost. Therefore, it was performed studies on the solution currently used; planning studies for the proposed solution; as well as the work of removing the ECU automotive parts, and the development of the PLCs Initial Configuration Software (PICS). The results obtained in this first validation of the concept were satisfactory and ensured the future progress of the project.

Palavras-chave: Building Automation; Programmable Logic Controllers; Electronic Control Unit; PLCs Initial Configuration Software.

Sumário

Lista de Figuras

Lista de Tabelas

Simbologia

CLP – Controlador Lógico Programável

PLC – *Programmable Logic Controller*

SCADA – *Supervisory control and data acquisition* (Sistema de supervisão e Aquisição de Dados)

ECU – *Eletronic Control Unit* (Unidade de controle eletrônico)

IHM – Interface Homem Maquina

HMI – *Human Machine Interface* (Interface Homem Maquina)

PID – Controlador Proporcional-Integral-Derivativo

OPC – *OLE for Process Control*

API - *Application Programming Interfaces*

SED – Sistemas de eventos discretos

SP – Set-Point

ASIC – *Application Specific Integrated Circuit*

IC – *Integrated Circuit* (Circuito Integrado)

ODBC – *Open Database Connectivity*

DDE – *Dynamic Data Exchange*

TCP – *Transmission Control Protocol* (Protocolo de Controle de Transmissão)

IP - *Internet Protocol (*Protocolo de Internet)

DDC – *Direct Digital Control*

UPS – *Uninterruptible Power Supply*

RTU – *Remote Terminal Unit*

NEMA – *National Electrical Manufacturers Association*

Capítulo 1: Introdução

1.1: Problemática

As soluções oferecidas pela multinacional Robert Bosch na área de controladores lógicos programáveis (CLP, ou PLC, em inglês) envolvem uma divisão da empresa chamada Bosch Rexroth. Tais controladores oferecidos como produtos, pela divisão responsável, fornecem soluções para complexos sistemas de automação de edificações e possuem qualidade assegurada dentro do mercado com alto índice de satisfação. Tais controladores foram desenvolvidos com extensa pesquisa e com alta tecnologia, resultando em produtos direcionados a edificações de grande porte, como aeroportos, estádios e complexos prisionais. O trabalho em questão se enquadra no desenvolvimento de controladores lógicos programáveis que atuem no mercado de edificações de médio porte. Edificações as quais não são contempladas pelas soluções já oferecidas pela Bosch Rexroth por apresentarem alto custo de implementação devido ao uso de controladores muito mais complexos do que necessário.

A divisão de sistemas de segurança (Security Systems) da Robert Bosch na Colômbia utiliza os controladores lógicos programáveis fornecidos pela Bosch Rexroth, para implementar sistemas de segurança que são oferecidos pela divisão. Tais sistemas envolvem, entre outros, o controle de acesso de pessoas, controle de incêndio, controle das câmeras de segurança, controle de elevadores e diversas outras tarefas que estão ligadas ao bem estar e a segurança dos usuários. A área de Security Systems da Colômbia avaliou que atualmente o mercado colombiano que demanda a realização de serviços de automação de edificações está em pleno crescimento e prevê num futuro próximo a necessidade de preencher a lacuna existente no portfólio de CLPs que suportam a divisão.

O presente trabalho foi desenvolvido na empresa Robert Bosch em Curitiba, Paraná, na área de desenvolvimento de produto customizado chamada Bosch Engineering Group (BEG). A BEG é uma prestadora de serviços de engenharia de alta tecnologia, é 100% subsidiária do Grupo Robert Bosch e tem como objetivo

oferecer adaptações customizadas para alguns sistemas, de acordo com as necessidades do cliente.

A situação atual dos sistemas oferecidos pela Colômbia envolvem diversos itens que encarecem muito a solução. Dentre os principais itens que se fazem necessários para a implementação dos atuais sistemas estão o CLP (Bosch Rexroth) que é aplicado não utilizando todas as funções que oferece, um servidor com padrão OPC (Terceirizado), um software SCADA, e a interface usuário máquina chamada de Building Integration System (BIS) da Bosch.

1.2: Motivação

O estudo aprofundado das funções de cada item da solução fornecida pela Bosch Colômbia se faz necessário para avaliar a necessidade de atuação de cada item e então definir aonde serão realizados os efetivos trabalhos de desenvolvimento do projeto.

À primeira vista, ao analisar a problemática apresentada como escopo de trabalho dentro do Grupo de Engenharia da Bosch, a ideia de desenvolver um novo produto Bosch, um novo controlador, já foi bastante motivante. Além disso, o desafio de trabalhar com um hardware voltado completamente para a tecnologia automotiva (utilizado para injetar combustível em veículos movidos à gasolina) e o risco de sua aplicação não ser bem sucedida no projeto foi mais um fator de motivação.

Outro ponto importante de motivação foi a importância dada a cada membro do grupo na pesquisa das soluções para atender às expectativas do cliente. Desta maneira, todos participaram ativamente nos momentos de decisões fundamentais para o andamento do projeto. Pude compreender que um fator que dificultou o desenvolvimento da solução foi a falta de experiência da BEG-Curitiba em trabalhos com CLPs, uma vez que esta, e a própria Bosch de Curitiba, são do ramo automotivo. Ao mesmo tempo, inseriu-se uma nova área de atuação para a BEG.

1.3: Objetivo

O objetivo geral do projeto é realizar o desenvolvimento de uma solução para automação predial, utilizando uma central eletrônica, para veículos movidos à gasolina, como controlador principal. Para tanto se pretende modificar a estrutura tanto de hardware como de software da ECU e aplicá-la em edificações, fazendo com que essa central atue como os atuais controladores lógicos programáveis designados para estas funções.

1.4: Expectativa

A expectativa para o resultado final do projeto de conclusão de curso é a validação do conceito apresentado como proposta, mostrando que o estudo realizado e os desenvolvimentos efetuados provam que a realização do projeto pode ser garantida, e que futuramente o CLP pode ser completamente implementado utilizando uma unidade de controle eletrônico para veículos.

1.5: Contexto no Curso

Em relação com o curso de Engenharia de Controle e Automação, dentre as matérias que compõem a grade curricular, notoriamente podemos verificar o uso de conceitos de Informática Industrial I e II, Processos em Engenharia, Redes de Computadores para Automação Industrial e Sistemas Distribuídos para Automação Industrial.

1.6: Metodologia

Com o intuito de contextualizar a participação de outros neste trabalho, podem ser citados os participantes Thomas Junge (Gerente da BEG), Eduardo Kamaroski Neto (Orientador e Líder de projeto), Filipe Cordeiro (Engenheiro de Software) e Hector Ibanez (Engenheiro de Calibração de Sistemas), porém todo escopo do projeto que será apresentado neste documento foi desenvolvido pelo presente autor, e quando outro estiver envolvido será citado. A metodologia da empresa para a validação do conceito consiste em quatro etapas bem definidas.

A primeira etapa consiste no entendimento do problema e na especificação de requisitos juntamente com o cliente. As especificações devem guiar todo o estudo em cima da proposta que será oferecida, e o entendimento do problema deve fornecer dados suficientes para tomada de decisões e planejamento de desenvolvimento. Assim são feitas as reuniões necessárias com o cliente para que o projeto seja bem definido. Ao longo do desenvolvimento é mantida sempre uma linha direta com o cliente para que o projeto atenda a todas as necessidades.

A segunda etapa é a etapa de estudos, dividia em duas partes: a primeira que estuda a solução aplicada atualmente, e a segunda onde é feita a análise da proposta. É na fase de estudo da solução atual que a solução utilizada atualmente é desmembrada e estudada ponto a ponto, com intuito de ambientar a equipe com as ferramentas e padrões utilizados, e principalmente fornecer dados para direcionamento dos campos onde o projeto irá atuar. Assim, nesta fase entende-se o problema por completo, a partir de dados provenientes do cliente, de revisões bibliográficas, e dados internos da empresa. É nesta parte também que são definidas aonde serão efetuados os trabalhos da BEG. Já na parte de análise da proposta são levantadas diversas hipóteses que se encaixam nos requisitos do cliente, de acordo com os dados provenientes do estudo da solução atual. Todas as hipóteses plausíveis são apresentadas ao cliente, que então faz o alinhamento de qual o melhor caminho a ser tomado.

A terceira e mais duradoura etapa é o desenvolvimento. Nesta são realizados os trabalhos definidos, no planejamento, como necessários para validação do conceito. Neste projeto esta etapa é determinada pela limpeza de partes automotivas da ECU, e pelo desenvolvimento do Software de Configuração Inicial de CLPs (PICS, do inglês, *PLCs Inicial Configuration Software*).

A quarta e última etapa é a validação do conceito e apresentação ao cliente. O projeto é, então, apresentado e confirma a validade do conceito definido como proposta. Nesta etapa também se define se o projeto irá ter continuidade, a partir da aceitação do cliente. Por fim, toda a documentação do projeto é redigida e arquivada.

Para dar apoio ao desenvolvimento, durante o período de estágio a empresa ofereceu cursos de capacitação, aulas de ambientação com o INCA (Hector Ibanez), treinamento com ECUs (Filipe Cordeiro), entre outros. Diariamente o líder de projeto e orientador, Eduardo Kamaroski, fez o acompanhamento do projeto para sanar dúvidas, passar conhecimentos, direcionar e verificar o rumo do projeto.

1.7: Estrutura do trabalho

O trabalho será apresentado em seis capítulos. O primeiro é a introdução, já apresentado. No Capítulo 2: é apresentado o local de estágio (Bosch Engineering Group Latin America). No Capítulo 3: é abordado o estudo da solução utilizada atualmente, envolvendo os conceitos para entendimento e as devidas conclusões em cima dos tópicos que dizem respeito à solução. No Capítulo 4: são apresentados os conceitos necessários para entendimento da solução proposta, bem como uma análise de custos atual e uma apresentação das diretrizes. No Capítulo 5: é apresentado todo o desenvolvimento em cima do Software de Configuração Inicial de CLPs. Por fim, são apresentadas as conclusões e as perspectivas futuras.

Capítulo 2: Local de Estágio

A Robert Bosch GmbH (conhecida apenas por Bosch) é uma empresa alemã fundada em Stuttgart em 1886, por Robert Bosch (1861-1942). A empresa é, há mais de um século, sinônimo de tecnologia de ponta não só no setor automobilístico, onde é conhecida por suas velas de ignição e rádios automotivos, mas também nos setores de produtos presentes na vida cotidiana da população, como: refrigeradores, ferramentas elétricas, equipamentos domésticos, máquinas industriais, sistemas de segurança, entre outros. A intenção é oferecer soluções úteis e inovadoras para melhorar a qualidade de vida de milhões de pessoas ao redor do mundo. [1]

O Grupo BOSCH é uma das maiores corporações industriais privadas do mundo, empregando mais de 302.000 pessoas (das quais 118.000 somente na Alemanha) e vendas anuais acima de €51 bilhões (dados referentes ao ano de 2011), estando presente em 150 países ao redor do mundo, com 50% de suas vendas oriundas de fora da Alemanha. Como pode ser visto na Figura 1, a empresa atua nos setores de tecnologia automotiva, tecnologia industrial, bens de consumo e tecnologia de construção. O grupo é composto pela Robert Bosch GmbH e suas mais de 350 subsidiárias e empresas regionais em 60 países. Além disso, tem 185 fábricas de produção no mundo, das quais 142 estão localizadas fora da Alemanha, na Europa, América do Norte e América do Sul, África, Ásia e Austrália. A BOSCH tem participação em 37 empresas através de parcerias, e é a terceira maior produtora de eletrodomésticos do mundo. [1][2]

A empresa é uma das mais tecnológicas e inovadoras do planeta. São 42.800 pessoas trabalhando em pesquisa e desenvolvimento na Bosch, em 89 localidades no mundo todo, que resultam na média de 19 registros de patentes por dia de trabalho. No ranking mundial, a Bosch está entre as companhias que mais requerem o registro de patentes e está em primeiro lugar na Alemanha. [1]

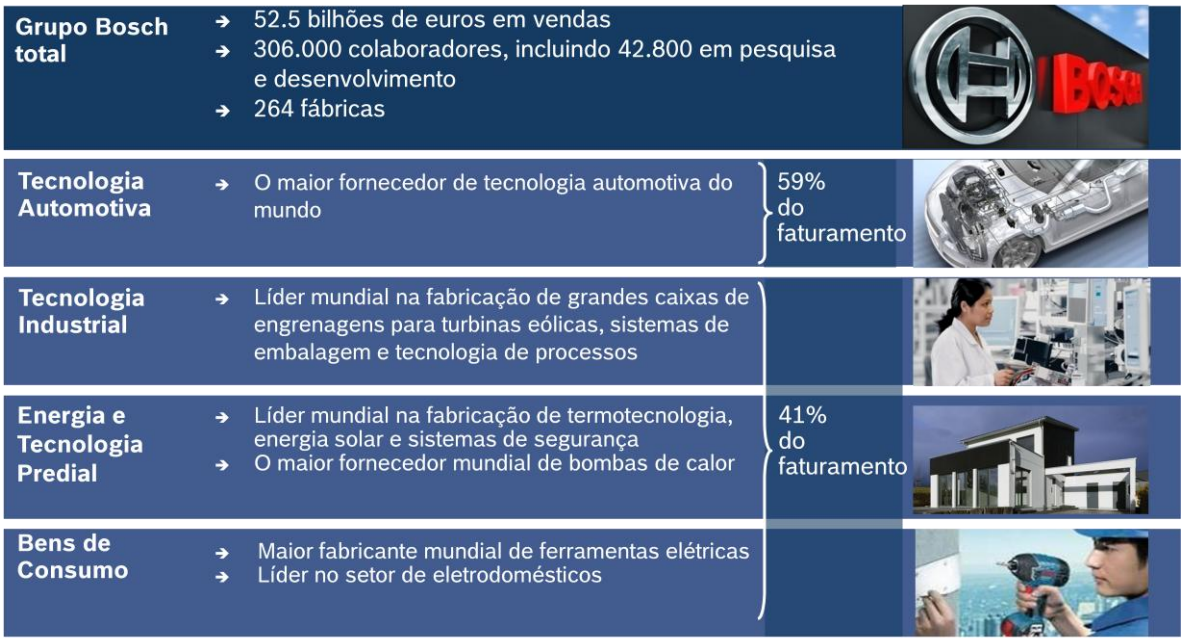


Figura 1 - Setores de atuação do Grupo Bosch. [1]

Nos últimos dez anos, os investimentos em pesquisa e desenvolvimento ultrapassam 35 bilhões de euros. O grupo BOSCH é detido em 92% pela Bosch Stiftung GmbH (Fundação Robert Bosch). [1]

Robert Bosch foi um dos pioneiros da indústria. Seu objetivo sempre foi aperfeiçoar a tecnologia de seus produtos, para satisfazer as necessidades diárias de seus clientes. Sua filosofia era: “Prefiro ganhar menos dinheiro a perder a confiança dos meus consumidores”. E é justamente esta filosofia de seu fundador que o Grupo BOSCH segue até os dias de hoje. [2]

2.1: Bosch Latin America (LA)

Na América Latina, a Bosch está presente em diversos países e cidades, principalmente no Brasil, aonde o número de funcionários chega a quase dez mil, e o faturamento equivale à aproximadamente 4,1 bilhões de reais. Os locais onde a Bosch se faz presente na America Latina pode ser visto na Figura 2, a seguir.

Figura 2 - Localidades e Atividades - Bosch Latin America. [1]

2.1.1: Bosch Engineering Group (BEG)

A Bosch Engineering Group, conhecida como BEG, é uma prestadora de serviços de engenharia de alta tecnologia e é 100% subsidiária do Grupo Robert Bosch. Surgiu em 1999, com o nome de Fundação ASSET, e ao longo de 11 anos teve sua evolução e crescimento até chegar ao Brasil em 2011. Atualmente, conta com 16 colaboradores na Planta de Curitiba e mais sete na Planta de Campinas, totalizando 23 colaboradores da nova divisão da América Latina. [3]

No domínio de sistemas eletrônicos, a BEG oferece adaptações sob medida de soluções padrão de acordo com requerimentos do cliente. Ela permite aos engenheiros tornar um produto único sem sacrificar as vantagens dos componentes de qualidade comprovada. A BEG oferece solução completa, desde o primeiro estudo até a liberação do produto para produção nas plantas Bosch. Entre os principais serviços prestados estão:

- ➔ **Adaptações personalizadas** baseadas na tecnologia Bosch;

➔ Soluções completas, **do estudo** e projetos de demonstração até **a liberação de produtos pelas plantas Bosch;**

➔ Especificações, desenvolvimento de funções, calibração e integração de software, incluindo todos os testes, avaliações e validações.

A integração da BEG no mundo Bosch pode ser observada na Figura 3 abaixo, a qual demonstra os campos de atuação, bem como a relação direta com o cliente.

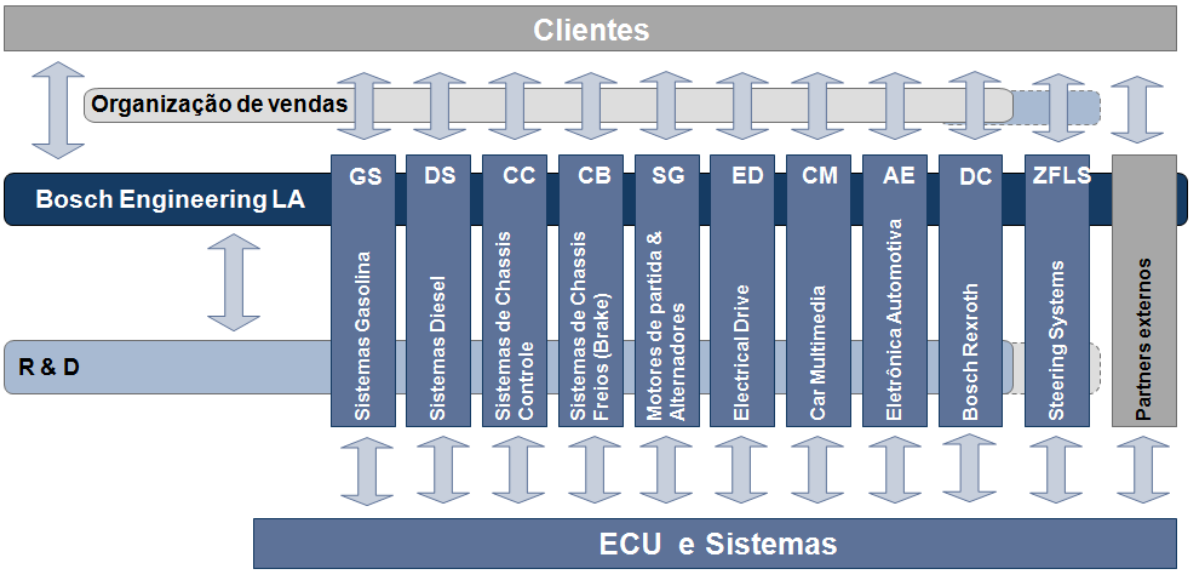


Figura 3 - Integração da BEG no mundo Bosch. [3]

Nota-se, então, que a BEG é uma empresa que realiza o estudo e desenvolvimento de projetos de tecnologia em escalas que não caracterizam produção em série. Uma vez que algum projeto teve seu conceito validado e possibilita a fabricação do mesmo em série, a BEG direciona o novo produto à produção nas plantas Bosch.

As próximas duas divisões que serão apresentadas fazem parte do escopo do projeto, sendo a divisão de **Security System** da Colômbia (definida como cliente) e a **Bosch Rexroth mundial** (beneficiária do novo produto, caso seja futuramente produzido em série).

2.1.2: Bosch Security Systems - Colômbia

A Bosch Security Systems está entre os maiores fabricantes mundiais de segurança eletrônica e sistemas de comunicação, oferecendo soluções de acordo com as necessidades de cada aplicação. Esta divisão da Bosch emprega 12.500 funcionários por todo o mundo e produziu mais de 1,4 bilhões de euros no ano fiscal de 2012.

O portfólio abrangente de produtos da Bosch inclui videovigilância, detecção de intrusão, detecção de incêndio e alarme por voz, bem como controle de acesso e sistemas de gestão de segurança e controle. Os sistemas de áudio profissional e de conferências para a transmissão de voz, som e música completam esta gama. Com o auxílio do grupo de produtos "*Engineered Solutions and Software* [Soluções de engenharia e software]", a Bosch ajuda e apoia a sua base de clientes internacionais através da concepção, elaboração de propostas e implementação de projetos extremamente complexos. Estes projetos são executados juntamente com os parceiros locais. [4]

2.1.3: Bosch Rexroth

A Bosch Rexroth é uma das principais especialistas no segmento de tecnologias de acionamento e comando. Desenvolve conceitos inovadores para construtores de máquinas e instalações industriais em todo o mundo. Com sua longa história e conhecimento único do setor, a empresa tornou-se sinônimo de soluções customizadas.

Bosch Rexroth é sinônimo de soluções produtivas, energeticamente eficientes e personalizadas, para todas as tecnologias de comando e acionamento, a partir de uma única fonte. A empresa tem uma história de mais de 200 anos. Seus 38.000 colaboradores, que trabalham na produção e customização de fábricas em 25 países, além de representantes de vendas em 80 países, conseguiram em 2011, um faturamento de mais de 6,4 bilhões de Euros. [5]

É a Bosch Rexroth que fornece os produtos relacionados ao controle e acionamento dos sistemas aplicados pela divisão de sistemas de segurança da

Colômbia. Nos casos mais usuais são utilizados os controladores lógicos programáveis "IndraLogic L”. [5]

Capítulo 3: Estudo da solução atual

3.1: Introdução

A necessidade de se desenvolver um novo controlador lógico programável utilizando como hardware uma unidade de controle eletrônico resulta na demanda por uma série de trabalhos a serem realizados, divididos em três macro-áreas:

- Estudo da solução utilizada atualmente pelo cliente (Security Systems - Bosch Colômbia)
- Estudo e definição da proposta de desenvolvimento
- Desenvolvimento efetivo dos primeiros passos da solução sugerida

Este capítulo irá discorrer sobre o estudo feito em cima da solução utilizada atualmente. Nos capítulos seguintes os outros dois tópicos serão abordados, tendo cada um uma preparação teórica que será apresentada de acordo com o tema do capítulo.

Vale ressaltar que tanto na etapa de estudo da solução atual como no planejamento da solução proposta, foi utilizado de um software, baseado em Java, chamado de *Freemind*. A partir de um diagrama de mapa da mente, sistematizado pelo inglês Tony Buzan, o software é voltado para pessoas que buscam a gestão de informações e de conhecimento, para a compreensão e solução de problemas, para memorização e aprendizado, e também é utilizado por profissionais que precisam de auxílio na gestão estratégica da empresa e como ferramenta de *brainstorming*. O *Freemind* busca aproximar o modo como o cérebro funciona, fazendo conceitos serem ligados da maneira mais natural possível.

Um exemplo de uma tela do Freemind pode ser vista no "APÊNDICE 1", o qual foi resultado de um *brainstorming* envolvendo os conceitos aplicados na solução atual. Nesse diagrama nota-se a presença de dúvidas, ideias e tópicos vislumbrados como tarefas futuras.

3.2: Ambientação com o problema

Em reuniões realizadas com o cliente, a equipe da Colômbia apresentou dados que confirmar a existência de uma demanda por monitoramento dos seguintes sistemas, como os concorrentes já oferecem:

1) Sistemas de força
2) Sistemas de Iluminação
3) Tratamento de águas
4) Plantas de Refrigeração
5) Boilers
6) Ar condicionado HVAC

Antes mesmo fazer contato com a BEG, a Bosch Colômbia testou diferentes tipos de soluções para implementação de controle e monitoramento de edificações. Tais soluções basicamente se resumem a substituir o CLP da Bosch Rexroth por um de outro concorrente, baixando assim o valor do principal componente do sistema, tendo em vista o alto valor de compra do CLP Rexroth. No Capítulo 4:, de análise da proposta, serão apresentados os valores gastos com uma solução Bosch para uma aplicação, bem como o que será feito para baixar o valor da solução como um todo.

Na Figura 4 abaixo, pode ser observado como são implementados os sistemas de controle da Bosch Security Systems na Colômbia. De acordo com a figura podem ser levantados os seguintes itens que farão parte fundamental da solução utilizada pelo cliente, são eles:

1) Building Integration System (BIS)
2) Winstudio
3) PLC, ou CLP.
4) Comunicação em padrão OPC
5) Comunicação com protocolo Ethernet
6) Outros (Módulos I/Os, visualizadores, etc.).

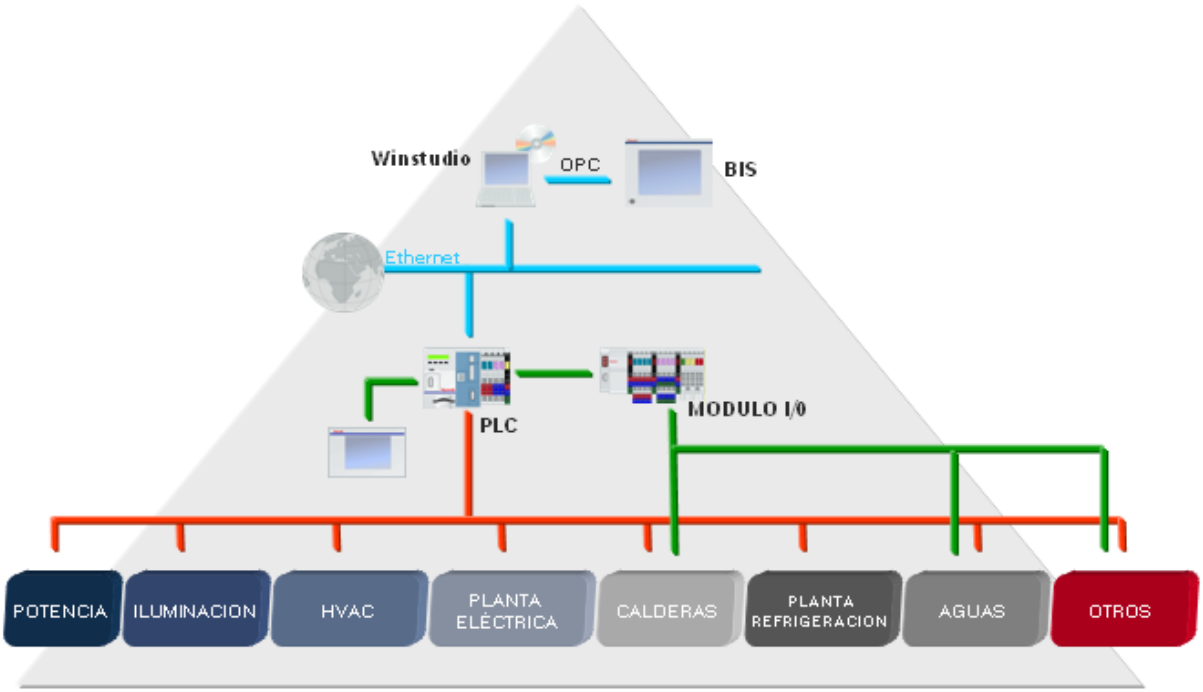


Figura 4 - Solução atual utilizada pela Bosch Security Sistems. [5]

Com intuito de facilitar o entendimento dos processos, um exemplo de caso de uso, seria o controle da iluminação de um determinado ambiente. Uma vez que o usuário deseja modificar a intensidade luminosa (Luminescência) deste ambiente, o mesmo insere o valor correspondente na interface humana-máquina (Software BIS), que, por sua vez, transmite a mensagem em padrão OPC para o servidor *Winstudio*. Tal servidor realiza o envelopamento da mensagem recebida em protocolo Ethernet e envia ao controlador lógico programável (CLP). Assim, possuindo os parâmetros desejados, o controlador pode atuar e alterar a luminescência atendendo o desejo do usuário.

Como mencionado acima, o software BIS atuará como uma interface humano máquina. Apesar do conceito variar um pouco, entende-se **HMI** (*Human Machine Interface*) ou **IHM** como um meio onde se traduz o que está acontecendo na máquina ou no sistema. Uma *Interface* é um dispositivo que está entre o homem e a máquina. A **IHM** tem vários formatos, desde um dispositivo isolado até um computador. No entanto, especialistas em instrumentação discordam desta última classificação, pois embora um PC com teclados ou telas de toque e monitor seja

uma interface, ele é mais comumente denominado de **estação de controle**, ou **console de operação**. Mais que cultural essas denominações decorrem da evolução da tecnologia. [7]

O capítulo a seguir irá apresentar os conceitos necessários para entendimento da solução, bem como as lacunas deixadas por cada item, de modo a levantar as possibilidades de proposta para a solução a ser desenvolvida neste projeto.

3.3: Conceitos para entendimento da solução atual

3.3.1: Controlador Lógico Programável (CLP)

Controladores Lógicos Programáveis ou Controladores Programáveis são frequentemente definidos como miniaturas de computadores industriais que contem um hardware e um software que são utilizados para realizar as funções de controle. Os **CLPs** leem dados de entrada, realizam lógica sobre uma linguagem específica e geram um sinal de saída atuando nos equipamentos. Geralmente as famílias de Controladores Lógicos Programáveis são definidas pela capacidade de processamento de um determinado numero de pontos de Entradas e/ou Saídas (E/S).

Segundo a NEMA (*National Electrical Manufacturers Association*), um CLP é um aparelho eletrônico digital que utiliza uma memória programável para armazenar internamente instruções e para implementar funções específicas, tais como lógica, sequenciamento, temporização, contagem e aritmética, controlando, por meio de módulos de entradas e saídas, vários tipos de máquinas ou processos. [8]

Um CLP é o controlador indicado para lidar com sistemas caracterizados por eventos discretos (SEDs), ou seja, com processos em que as variáveis assumem valores zero ou um (ou variáveis ditas digitais, ou seja, que só assumem valores dentro de um conjunto finito). Podem ainda lidar com variáveis analógicas definidas por intervalos de valores de corrente ou tensão elétrica. As entradas e/ou saídas

digitais são os elementos discretos, as entradas e/ou saídas analógicas são os elementos variáveis entre valores conhecidos de tensão ou corrente. [9]

3.3.1.1: Histórico

O Controlador Lógico Programável (CLP[1]) nasceu dentro da General Motors, em 1968, devido a grande dificuldade de mudar a lógica de controle dos painéis de comando a cada mudança na linha de montagem. Tais mudanças implicavam em altos gastos de tempo e dinheiro. Sob a liderança do engenheiro Richard Morley, foi preparada uma especificação que refletia as necessidades de muitos usuários de circuitos e relés, não só da indústria automobilística como também de toda a indústria manufatureira. Nascia assim um equipamento bastante versátil e de fácil utilização, que vem se aprimorando constantemente, diversificando cada vez mais os setores industriais e suas aplicações, o que justifica hoje um mercado mundial estimado em quatro bilhões de dólares anuais. [9]

As vantagens dos CLPs em relação a circuitos de comandos eletromagnéticos são: menor espaço físico, menor consumo de energia elétrica, são reutilizáveis, programáveis, apresentam maior confiabilidade, maior flexibilidade, promovem maior rapidez na elaboração dos projetos e oferecem interfaces de comunicação com outros CLPs e computadores.

3.3.1.2: Princípio de funcionamento

Um CLP pode ter seu funcionamento representado utilizando três partes, são elas: os módulos de entradas, a unidade de processamento (CPU) e os módulos de saída. O fluxo representativo é demonstrado na Figura 5.

[1] (***CLP é marca registrada da Rockwell Automation, no Brasil, portanto nas literaturas técnicas e nos manuais dos demais fabricantes costuma-se usar a abreviação do nome inglês, PLC***)

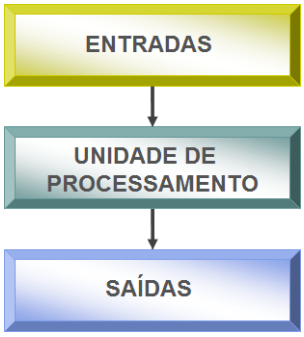


Figura 5 - Representação do funcionamento de um CLP. [25]

O ciclo de varredura de um CLP determina o conjunto de tarefas que serão necessárias para que o processador siga uma determinada sequencia de operações durante seu funcionamento. Quando o CLP é posto para funcionar um programa faz rotina de reconhecimento de diversos itens como reconhecimento dos módulos de entradas e saídas ligadas ao CLP e estado da memória (verifica se existe um programa de usuário instalado). Se todo o hardware (parte física do CLP) está em condições e se existe um algoritmo de controle gravado na memória, a rotina de inicialização busca o programa do usuário e, a partir daí, o ciclo de varredura começa a ser realizado. Este ciclo, repetitivo, consiste em verificar o estado das entradas e saídas, armazenar esta leitura na memória, fazer a comparação desta imagem com o programa do usuário e atualizar as saídas, caso a imagem esteja diferindo do programa. Veja o diagrama explicativo na Figura 6. [9]

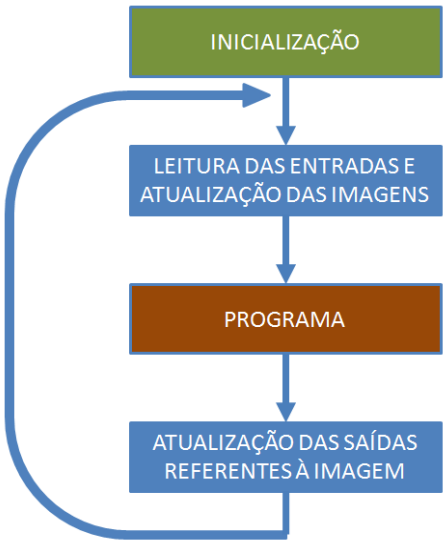


Figura 6 - Ciclo de varredura de um CLP.[25]

3.3.1.3: Entradas

As entradas são as interfaces que permitem ao CLP receber informações sobre o processo, é onde entram os sinais provenientes de botoeiras, contatos de relés, sensores, encoders, e todos os tipos de dispositivos usados para monitorar o processo e fornecer um retorno de informação ao CLP. As entradas podem ser digitais ou analógicas e ainda podem ser internas ou externas. As entradas externas são aquelas por onde entrará o sinal enviado por um sensor ao CLP e as internas são aquelas que recebem sinal de outro componente interno do CLP, como por exemplo, o contato de um temporizador utilizado para ligar um outro componente interno ou uma saída externa. [9]

- **Entradas digitais:** são aquelas que recebem sinais discretos, ou seja, sinais que só possuem dois valores que são denominados de nível alto, representado pelo algarismo 1, e nível baixo, representado pelo algarismo 0. Em outras palavras, um sinal discreto pode ser representado por um interruptor que só oferece as opções ligado (nível alto) ou desligado (nível baixo).

- **Entradas analógicas:** são aquelas que recebem sinais contínuos no tempo e que podem assumir qualquer valor entre o mínimo e o máximo valor de trabalho da entrada. Um sinal analógico pode ser, por exemplo, o sinal enviado por um tacogerador para controlar a rotação de um motor. A tensão aumenta continuamente à medida que aumenta a rotação do motor.

3.3.1.4: Saídas

As saídas são as interfaces através das quais o CLP pode alimentar uma carga. Assim como as entradas, as saídas também podem ser do tipo digitais ou do tipo analógicas. As saídas, assim como as entradas, podem ser externas ou internas. As saídas externas são aquelas por onde se comanda um motor, por exemplo. Isto é, o CLP irá enviar um sinal elétrico para um componente externo a ele, enquanto que uma saída interna pode ser a bobina de um FLAG (será visto adiante) ou ainda a bobina de um temporizador interno.

- **Saídas digitais:** são aquelas que só oferecem dois valores, nível alto ou nível baixo. Nestas saídas podem ser ligadas lâmpadas, solenoides de contatores, solenoides de eletroválvulas ou qualquer dispositivo que só precise ser alimentado com tensão nominal ou desligado.

- **Saídas analógicas:** são as interfaces através das quais o CLP pode variar continuamente no tempo a tensão ou a corrente sobre uma carga. Um bom exemplo de carga que pode ser ligada a uma saída analógica, através de uma placa de controle, é um motor CC que varia a rotação conforme varia a tensão sobre o induzido.

3.3.2: Building Integration System (BIS)

O Building Integration System (BIS) é uma plataforma flexível projetada para atuar no gerenciamento, controle e monitoramento de edificações, e que atenda a requisições feitas por um usuário específico e necessidades individuais.

Segundo o fabricante: "*O poderoso Building Integration System é um software que oferece níveis de integração e customização considerados como um dos melhores na indústria. Uma vez instalado, este software oferece ao seu cliente o melhor em termos de gestão de alarme, status dos dispositivos conectados, e o mais importante, a integração de todos os sistemas de segurança. O software é flexível o suficiente para trabalhar com os produtos Bosch Security, bem como equipamentos de outros fabricantes usando a interface padrão OPC.".* [10]

A ferramenta contém uma imensa gama de aplicações e fornece uma solução completa para gerenciamento de edificações, como pode ser visto na próxima página, na Figura 7.

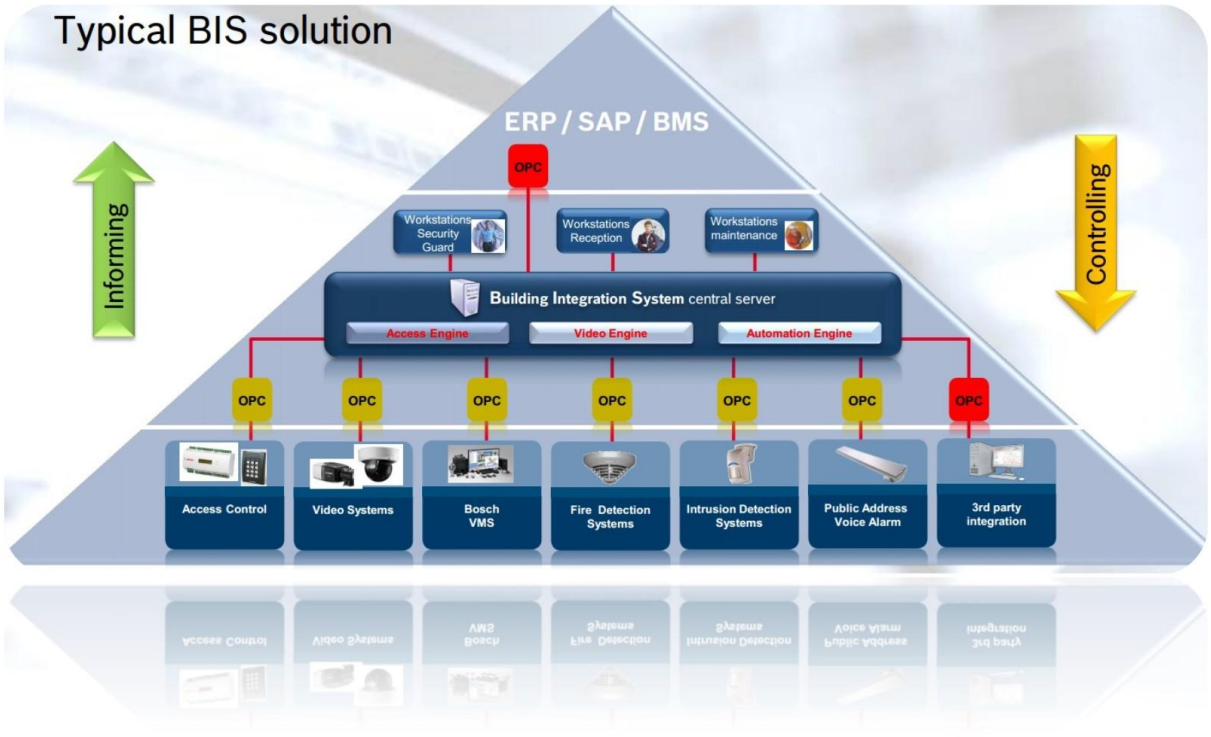


Figura 7 - Soluções típicas BIS. [10]

Hoje em dia o gerenciamento eficiente de edificações está se tornando não somente mais importante como muito mais complexo de se desenvolver. Atualmente organizações utilizam múltiplos sistemas para monitorar e controlar seus prédios, variando desde alarme de fogo e de intrusão até controle de acesso, supervisão por câmeras e sistemas automatizados. O Building Integration System é uma plataforma projetada para atuar em ambiente de controle e monitoramento, que exerce tarefas de segurança customizada e gerenciamento de fatores vinculados à automação de edificações. [11]

Dentre suas capacidades podem ser evidenciadas:

- Gerenciamento completo e eficiente, integração de edificações e controle de segurança em uma única solução;
- O uso de consistentes padrões abertos de TI para configurar, interfacear, e demonstrar;
- Provê instalação e uso amigáveis;
- Oferece integração entre Bosch e produtos terceirizados via padrão OPC;

- Oferece uma grande variedade de alarmes informativos que facilmente são ligados a configurações feitas pelo usuário, resultando em planos de ação pré-programados;
- Estrutura modular que facilita a configuração de uma solução de segurança e atende exatamente aos requerimentos do usuário.

O Building Integration System é essencialmente uma família de produtos feita por diversos outros produtos principais, também conhecidos como "*Engines*", os quais são baseados na mesma plataforma de software. Tais produtos principais, que tem hierarquia como mostrado na Figura 8, são os seguintes:

- Ferramenta de Automação (*Automation Engine*)
- Ferramenta de Acesso (*Access Engine*)
- Ferramenta de Vídeo (*Video Engine*)
- Ferramenta de Segurança (*Security Engine*)

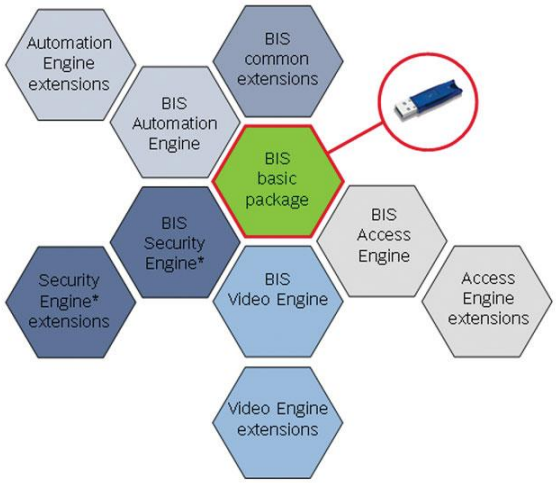


Figura 8 - "Engines" do BIS. [11]

Em princípio estas ferramentas podem ser combinadas em um ilimitado numero de formas. Isso permite ao usuário criar um sistema de gerenciamento que atinja seus requisitos específicos. Atualmente, a utilização do BIS é feita em edificações de grande porte, uma vez que tais ambientes possuem sistemas de alta complexidade. Um exemplo de aplicação pode ser observado na Figura 9.

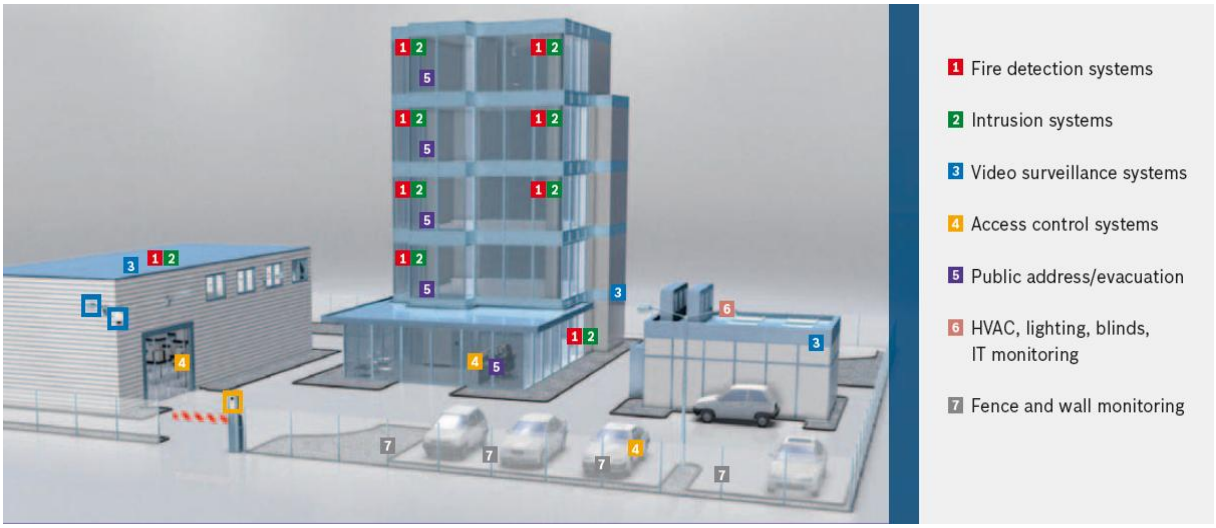


Figura 9 - Exemplo de implementações de "engines". [12]

O Brasil também possui uma divisão da Security Systems como a Bosch da Colômbia, a qual é situada em Campinas, no estado de São Paulo. Visando levantar o maior número de informações para o projeto, várias reuniões foram realizadas com os integrantes da divisão no Brasil, bem como uma semana de treinamento em Campinas com a ferramenta BIS. Durante o tempo de trabalho com a equipe de Campinas, puderam-se levantar pontos importantes sobre a utilização dessa ferramenta.

Analisando a necessidade de tal ferramenta para a solução do novo CLP, ficou claro que o BIS possui uma arquitetura bem fechada e de difícil modificação em termos de software. Além disso, o BIS possui todas as regulamentações propostas por lei para atuação, e qualidade assegurada por testes e simulações. Tais fatores, somados à requisição do cliente em permanecer com a plataforma por ser um carro-chefe de vendas e por possuírem experiência com a mesma, impossibilita a exclusão do software BIS da solução final do novo CLP.

Tendo em vista que o software que fará a interface entre usuário e máquina está definido e deve ser o BIS, restam as seguintes considerações:

- O BIS é um software baseado em internet - A plataforma deve rodar num ambiente de internet;

- Por motivos de incompatibilidade, o BIS não roda em máquinas que contenham o Peacy, o que gera um obstáculo para o desenvolvimento de qualquer solução a ser realizada em cima dessa ferramenta. Peacy é um sistema de controle de segurança da Bosch que roda em todos os computadores da empresa. O Peacy atua assegurando a confidencialidade de documentos, a proteção contra vírus, e outras tarefas.;
- Protocolo IP é o mais utilizado nas aplicações atuais;
- Não é tarefa simples criar uma interface de visualização: como o software é web-based, o desenvolvimento de uma interface de visualização faz necessário o uso de outra ferramenta de desenvolvimento (Dojo), que ainda não foi testada;
- Não é possível mandar informações pela interface - Atualmente o BIS exerce função somente de monitoramento nas aplicações feitas pela equipe de Campinas, não sendo usado para mandar nenhum tipo de informação, set-point (SP) ou alteração de valores lidos;
- Ferramenta de conexão para servidor OPC: Na última versão lançada do BIS é oferecida uma ferramenta de auxílio à conexão de servidores com padrão OPC. Anteriormente somente poderiam ser conectados os servidores que faziam parte da lista de servidores que eram suportados pelo BIS.

Portanto, o software de integração de edificações deve ser usado na solução final e participa como item principal na venda dos produtos relacionados à integração de soluções para a Bosch Security Systems. Dessa forma a ideia para o BIS é mantê-lo como interface humana-máquina, e trabalhar em cima das possibilidades oferecidas para desenvolver a melhor interface de visualização e de atuação, bem como fornecer uma solução de comunicação via padrão OPC para intercomunicar com o restante do sistema.

3.3.3: Padrão OPC

Em meados dos anos 90 algumas empresas se reuniram para unir forças com o objetivo de desenvolver um padrão para acesso à dados em tempo real dentro do sistema operacional Windows. Nesta tarefa alguns membros da Microsoft foram

envolvidos para oferecer suporte técnico à solução a ser adotada. Este grupo sem fins lucrativos é formado por diversas empresas, e é gerenciado por uma organização chamada OPC Foundation (www.opcfoundation.org). Basicamente, o padrão OPC estabelece algumas regras para que sistemas com interfaces padrões para comunicação sejam desenvolvidos, ou melhor, resume-se em conectividade aberta via padrões abertos. OPC é conectividade aberta na automação industrial e em sistemas de empresas que auxiliam a indústria. A interoperabilidade é assegurada através da criação e manutenção de especificações-padrão abertas. Existem, atualmente, sete especificações-padrão completas ou em desenvolvimento. [7] Para entendimento do padrão OPC, os seguintes conceitos devem ser apresentados:

- **COM** *(Component Object Model)* é um padrão criado pela Microsoft para que uma aplicação cliente possa acessar os serviços de uma aplicação servidora. O *server* implementa seus serviços como “objetos” que podem ser acessados pelos *clients*. COM especifica as *interfaces* para clientes e para a comunicação padronizada. A *interface* é uma coleção de métodos ou funções e procedimentos relacionados que implementam os serviços específicos providos por COM. [7]
- **DCOM** *(Descentralized Component Object Model)* provê os mesmos serviços do COM, porém utiliza como meio de acesso uma rede de comunicação de dados. [7]
- **OLE** *(Object linking and Embedding)*: *object* é uma unidade de informação que pode ser criada e manipulada pelos usuários. Em outras palavras, objetos são módulos de dados/softwares que podem ser incluídos nos pacotes de softwares. Podem ser *linked* ou *embedded*. Enquanto no primeiro os objetos são armazenados separadamente e “repartidos” por várias aplicações, no segundo são armazenados junto com as aplicações, tornando-se de uso exclusivo das mesmas. O computador “se entende” com os periféricos (impressoras e unidades de CD, por exemplo) através de troca de informações sob a forma de impulsos elétricos. Para interpretá-los, é necessário um programa. Esse programa (na verdade, um simples conjunto de rotinas) chama-se *driver* (ou "gerenciador") de dispositivo. Ele se integra

ao sistema operacional e funciona como "intérprete" entre o computador e o periférico. Cada periférico usa um *driver*. Como os *drivers* são específicos para o dispositivo (ou seja, cada modelo de dispositivo usa seu próprio *driver*), em geral eles são desenvolvidos pelo fabricante do dispositivo e são fornecidos com o próprio dispositivo (em disquete ou CD-ROM) ou repassados ao desenvolvedor do sistema operacional para serem distribuídos com o sistema. [7]

Uma regulamentação de padrão de comunicação nunca foi tão necessária como o padrão OPC. Hoje em dia, o acrômio para “OLE for Process Control”, se contextualiza dentro de um padrão que é certamente inevitável para qualquer fabricante de componentes de automação.

A importância do uso de software no campo da automação vem ganhando cada dia mais força, e isso faz com que se direcionem soluções de software baseadas em computadores. Tais soluções não devem ser desenvolvidas como um bloco de software monolítico, mas devem ser compostas de módulos de software individuais, de modo a facilitar o reuso de alguns blocos para formar novas soluções para outros problemas.

Devido à incompatibilidade de interface de comunicação entre blocos de software, tempo e dinheiro devem ser investidos sempre que se faz necessário adaptar uma interface de comunicação para cada nova combinação de módulos de software. Isso resulta num imenso número de *drivers* de software produzidos para aplicações específicas. O *OLE for Process Control* trouxe a cura para esse problema: o OPC habilita componentes de software como conectores de software. Esses conectores combinados permitem a intercomunicação sem adaptações especiais.

A introdução do padrão OPC fez com que a quantidade de *drivers* desenvolvidos fosse reduzida para a necessidade de apenas um: o *OPC Server* (Servidor OPC - responsável por disponibilizar os dados). Da mesma maneira, somente uma única interface de *driver* fosse requerida para o fabricante do software: a interface do *OPC Client* (Cliente OPC - responsável por requisitar os dados). Como resultado tanto fabricantes como clientes se beneficiam desta solução, isso porque

hoje em dia os usuários são capazes de combinar qualquer sistema de controle ou de visualização com qualquer tipo de hardware via OPC.

3.3.3.1: Software Winstudio

Uma ferramenta utilizada na solução utilizada pela Bosch Colômbia nas aplicações atuais é o *software* chamado de *Winstudio*. *Winstudio* é uma poderosa ferramenta para construção de aplicações SCADA (*Supervisory Control And Data Acquisition*) ou HMI para automação industrial que explora propriedades do *Microsoft® Windows® NT/2000/CE.* A aplicação consiste em uma interface animada (Figura 10), drivers (configuráveis para CLPs ou outros dispositivos), um banco de dados para diferentes aplicações e módulos opcionais como alarmes, gráficos, timers, e sistemas de segurança. O software comunica-se com sistemas de entradas e saídas industriais, e outras aplicações que envolvam a plataforma Windows em ambiente de tempo de execução utilizando ODBC, DDE, NetDDE, OPC ou protocolos TCP/IP.

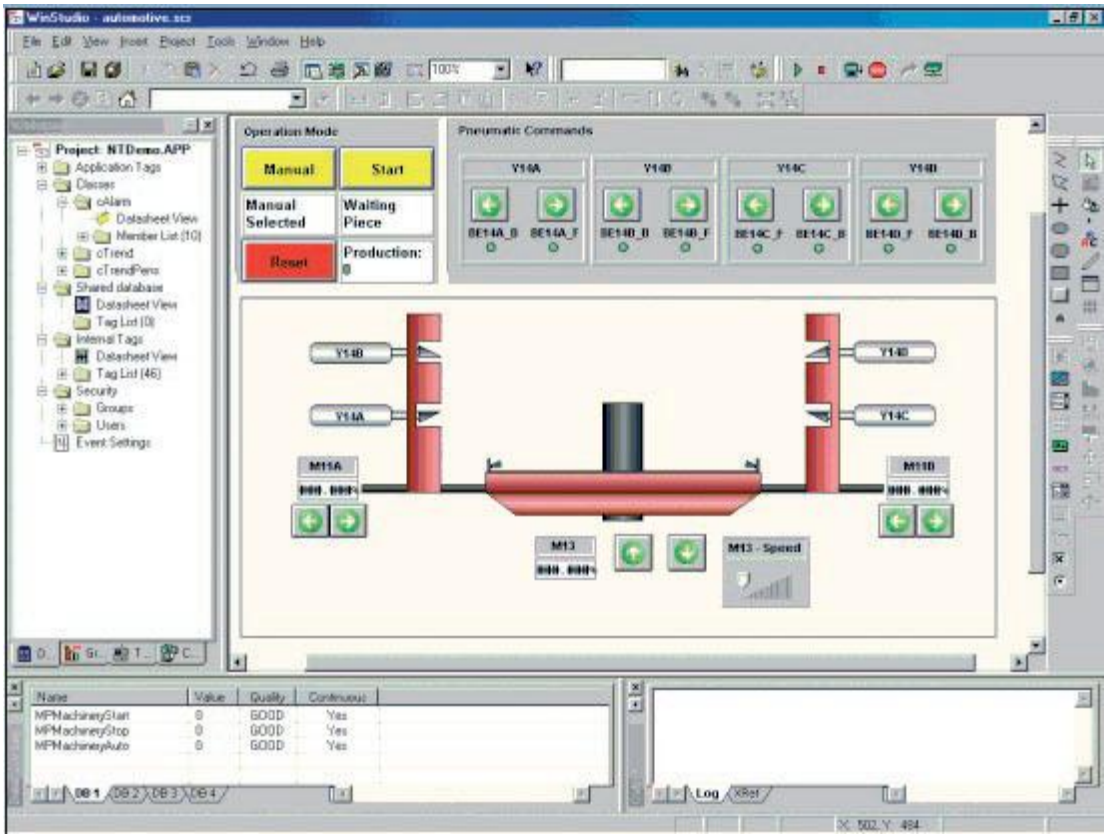


Figura 10 – Winstudio. [21]

Analisando os pontos levantados pelo estudo sobre o padrão de comunicação OPC e os requisitos demandados pelo cliente, deve ser levado em conta que no caso da solução atual utilizada pela Bosch Colômbia, o software *Winstudio* atua como um servidor OPC. Assim excluindo o fato de que o padrão OPC já é bastante consagrado e a utilização deste padrão no desenvolvimento do projeto se faz obrigatório, foi analisado que o software Winstudio possui um alto custo para as funções que o mesmo exerce. Na solução atual o SCADA não atua como tal, não inserindo nenhuma lógica de controle ou alguma interface humana-máquina, o Winstudio somente atua como um cliente-servidor, pois é desenvolvido para ambiente Microsoft, comunica-se via padrão OPC, e é um produto Bosch facilitando a interoperabilidade entre Winstudio e BIS. Diante de tal levantamento, uma primeira área de trabalho foi destacada, promovendo o posterior estudo de uma alternativa para a comunicação entre BIS e o novo CLP, via padrão OPC. O estudo conclusivo e as opções pesquisadas serão apresentados no próximo Capítulo 4:**,** que comenta sobre as decisões tomadas em direção à solução final.

3.4: Exemplo de solução para sistemas de integração

A seguir podem ser observadas algumas breves observações acerca de uma típica instalação de um sistema de integração e automação utilizando os serviços básicos do BIS. Nesta seção é possível perceber as dificuldades citadas anteriormente, bem como notar os diferentes tipos de comunicação que se empregam numa solução deste tipo.

Um exemplo de modelo geral centra-se em colocar controladores coletores de sinais ao longo de todo o edifício, ligados por uma rede Ethernet, comunicando-se com um sistema SCADA para visualização e gerenciamento de comandos. Tal modelo pode ser observado na Figura 11, na próxima página.

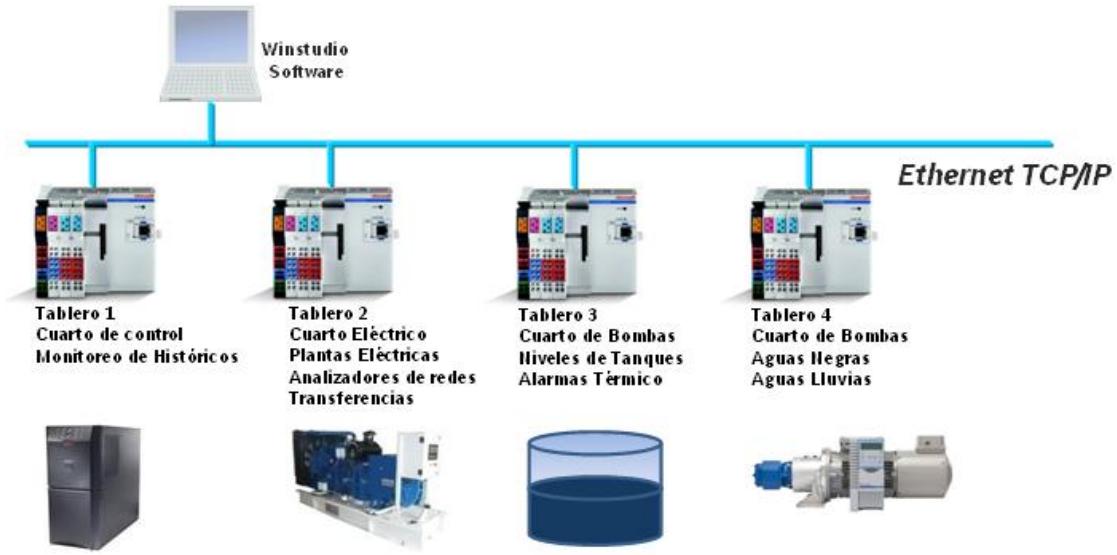


Figura 11 - Modelo geral típico de um sistema integrado.[18]

3.4.1: Painel de Controle 1

O Painel de Controle 1 faz a coleta de sinais dos distintos UPS conectados através de um controlador Maestro Modbus RTU (cabeado), com no máximo 32 escravos, como no exemplo da Figura 12.

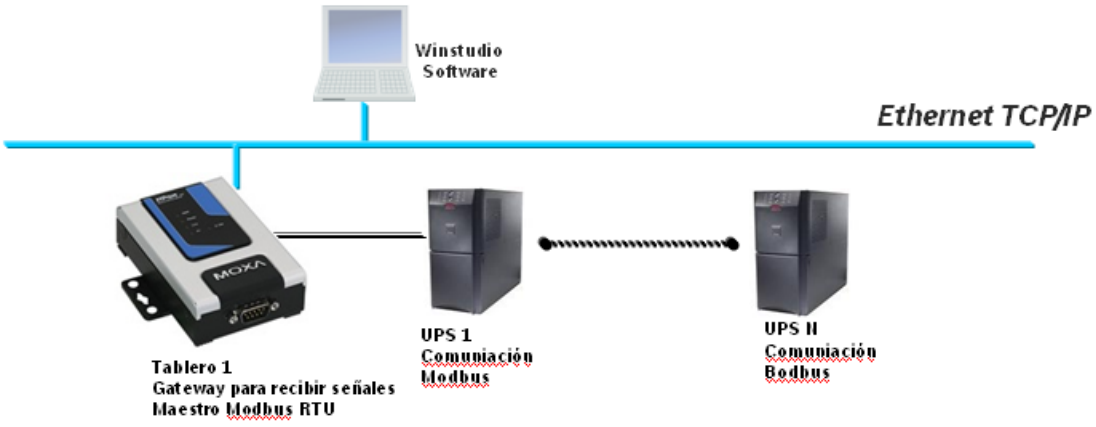


Figura 12 - Painel de controle 1. [18]

3.4.2: Painel de Controle 2

Na Figura 13, observa-se que o Painel de Controle 2 pode realizar a coleta de sinais de transferências, analisadores de rede, usina de energia e os níveis do tanque de combustível da usina.

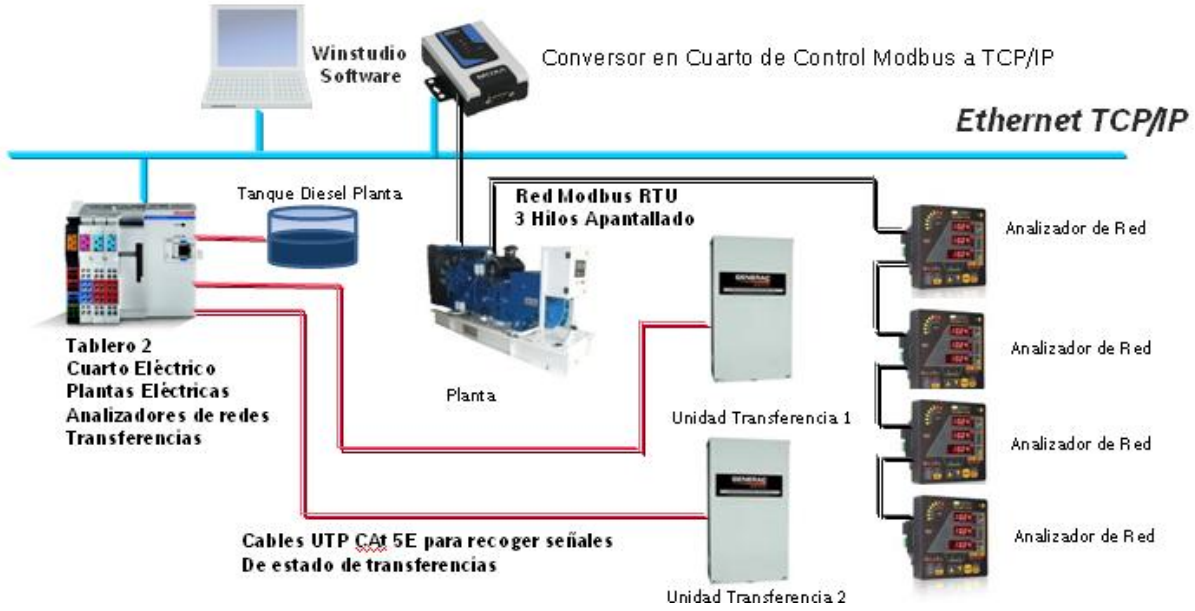


Figura 13 - Painel de controle 2. [18]

Para isso, deve-se colocar um conversor Modbus RTU para recorrer ao barramento que interliga os analisadores de rede e a planta eletrica. O módulo atua como mestre, e converte os sinais para o TCP / IP. Para monitorar o status da transferência através de um contato seco, verifica-se o interruptor. Para o alto e baixo níveis dos tanques é o mesmo caminho, porém para este tipo de contato seco é necessário colocar um CLP com entradas digitais, juntamente com o módulo conversor.

3.4.3: Painel de Controle 3

O terceiro painel de controle (Figura 14) lida com sinais de bombas de água potável, bombas de água da chuva, e os níveis de tanque (alto, baixo, e mínimo nível de água para a extinção). Bombas de água e bombas de água de chuva tem

controladores que, por meio de contatos secos, possibilitam monitorar sinais de incêndio (manual ou automático), sinais térmicos das bombas e, em alguns casos, o estado do inversor de frequência das bombas.

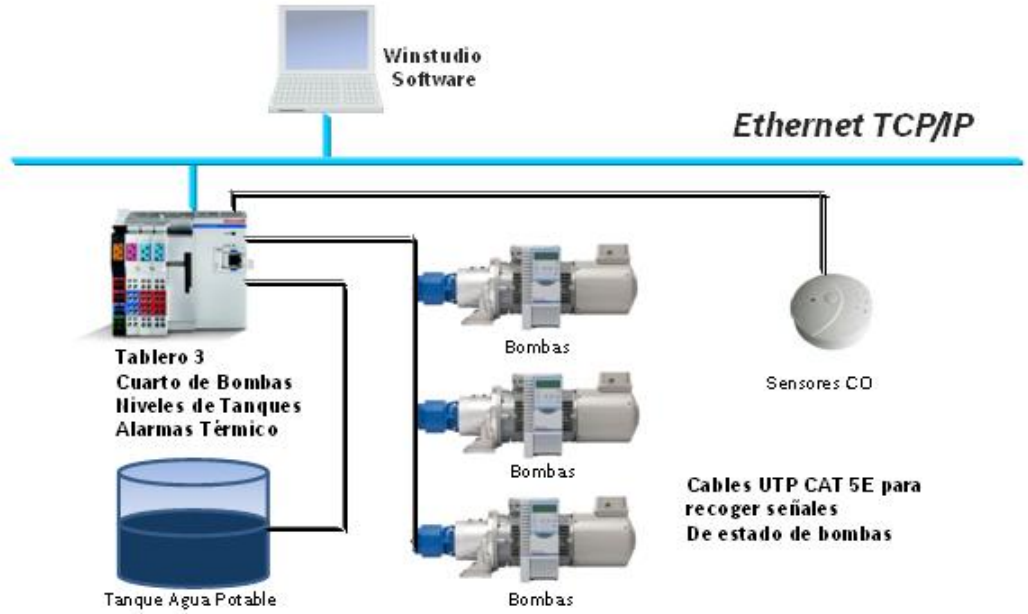


Figura 14 - Painel de controle 3. [18]

3.4.4: Painel de Controle 4

O Painel de Controle 4, mostrado na Figura 15, trata do controle de ar condicionado. Neste são necessários alguns cuidados, uma vez que, para ar condicionados, geralmente se utilizam controladores do tipo DDC (Direct Digital Control), e não CLPs. Somente devem ser usados CLPs quando não se deseja controlar grande quantidade de sinais analógicos, cujo custo é muito mais elevado.

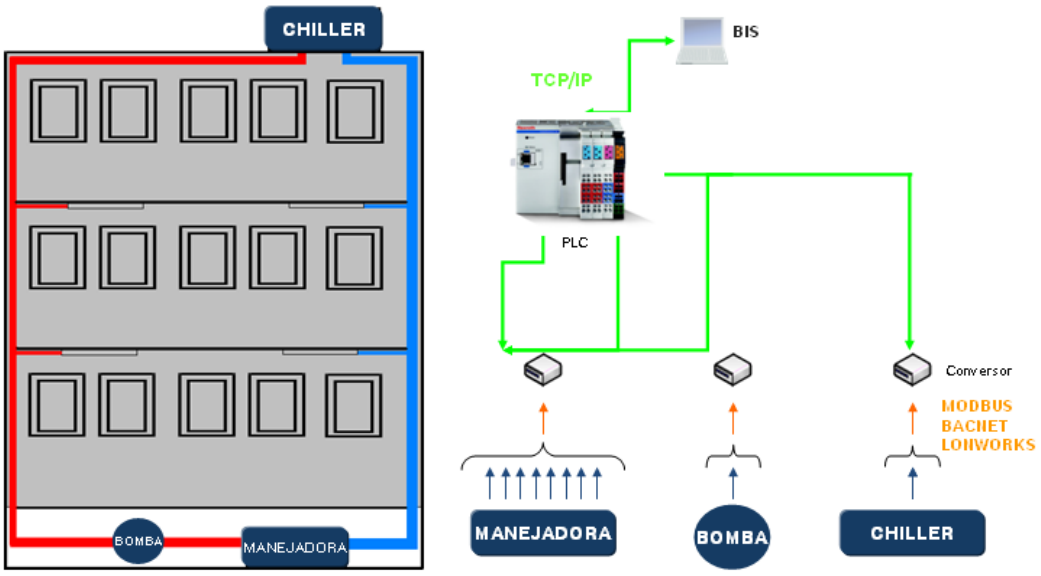


Figura 15 - Painel de controle 4. [18]

O Chiller e as demais unidades comunicam-se normalmente em três protocolos diferentes: BACnet, Modbus ou Lontalk. Isso significa que se deve usar uma conversor de protocolo para protocolo IP, para depois enviar o sinal ao controlador principal.

Para a parte de software, deve existir um software SCADA que permita gráficos dinâmicos, gerenciamento de processos e de horários. Este software se comunica com o BIS por padrão OCP, o que permitirá visualizar as associações e alarmes no CLP. A programação do software SCADA é feita por terceiros (não membros da Bosch Security Systems), uma vez que existem grandes diferenças em programar controladores de segurança e controladores de automação.

Os controladores de sistemas de automação levam cabeamentos mais exigentes do que quando comparado com sistemas de segurança.

Finalmente, a interoperabilidade entre CLP, SCADA e BIS é feita de acordo com a ilustração da Figura 16 a seguir:

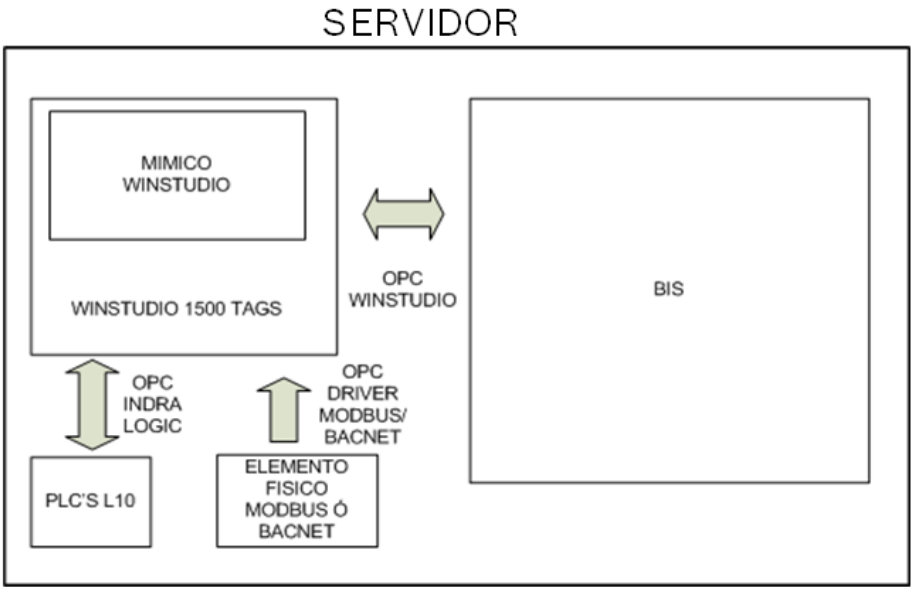


Figura 16 - Comunicações entre SCADA, BIS, e controladores. [18]

Abaixo, na Figura 17, pode-se observar uma tela gráfica do software SCADA, Winstudio.

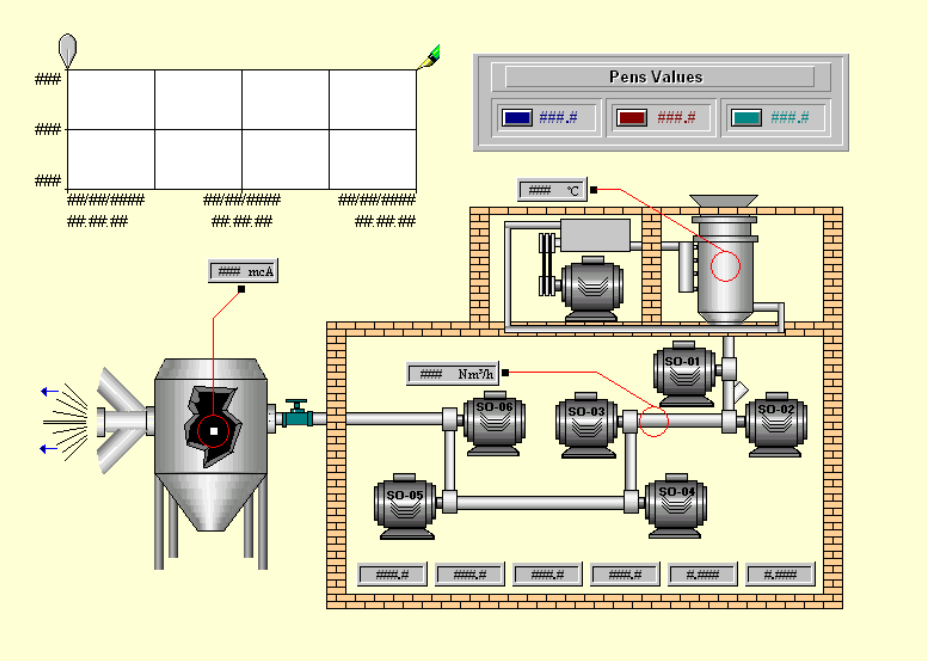


Figura 17 - Exemplo de tela do Winstudio. [18]

Capítulo 4: Estudo e proposta de solução

Este capítulo irá apresentar todo o estudo feito em cima da preparação teórica para entendimento da proposta sugerida, bem como os valores empregados atualmente, e as diretrizes para melhoramento da solução atual.

4.1: Conceitos para entendimento solução proposta/sugerida

4.1.1: ECU Value Motronic

No ramo da tecnologia automotiva, controlar as diversas tarefas que um automóvel precisa desempenhar é tarefa da unidade de controle eletrônico, que dentre outras tarefas, realiza:

- Controle da relação ar / combustível
- Controle de ponto de ignição
- Controle de marcha lenta
- Controle de comando de válvulas
- Válvula de controle eletrônico
- Ativação da ventoinha

Visando atender uma demanda para veículos de baixo custo, a Bosch desenvolveu uma central eletrônica que possui qualidade consistente em termos de eletrônica, manufatura, software, e calibração. A ECU Value Motronic (ECU VM) caracteriza a linha de baixo custo de centrais de controle eletrônico da Bosch, e pode ser vista na Figura 18.

Figura 18 - Eletronic Control Unit (ECU). [13]

A vantagem da ECU VM é ser produzida em série na planta da Bosch na cidade de Campinas-SP, resultando num custo de aquisição baixo, quando comparado com um CLP convencional como citado anteriormente. O valor de uma ECU VM é de aproximadamente US$ 100,00.

Sendo utilizada para veículos de até quatro cilindros essa central preenche os pré-requisitos de mercados emergentes oferecendo uma arquitetura de hardware com chipset de alto nível integrado, uma estrutura funcional de software, e calibração eficiente e otimizada para redução de custo. Todas as outras particularidades da VM não serão comentadas por não serem importantes para entendimento da solução ou por motivos de confidencialidade. [13]

Em termos de hardware (Figura 19) as principais da ECU são:

- Controlador moderno de 32bit com memória interna (Flash, RAM).
- Um ASIC (Application Specific Integrated Circuit) integrado. (Um ASIC combina funcionalidade de quatro ICs)
- Ponte de comunicação de alta velocidade entre o controlador e o ASIC.
- Conceitos de chip para motores a gasolina de até quatro cilindros.
- Baseado na família de software ME17

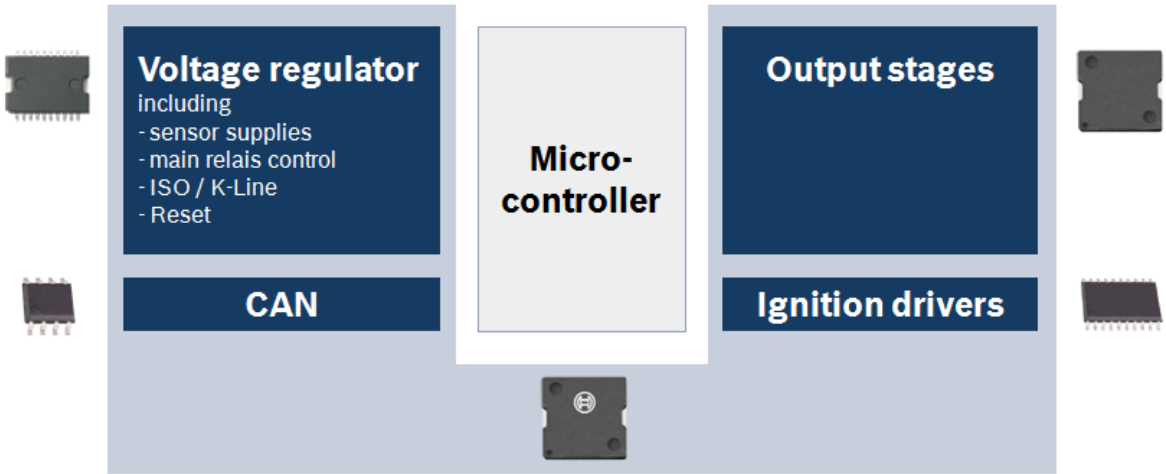


Figura 19 - Hardware da ECU. [13]

Em termos de software, a ECU Value Motronic oferece diversas funcionalidades para o uso em veículos e dentre as principais estão: o controle de

injeção de combustível e o controle de torque e controle de pressão, tudo voltado para otimização de desempenho. Uma ilustração dessas funcionalidades, bem como as regiões de atuação da VM, são observadas na Figura 20, abaixo:

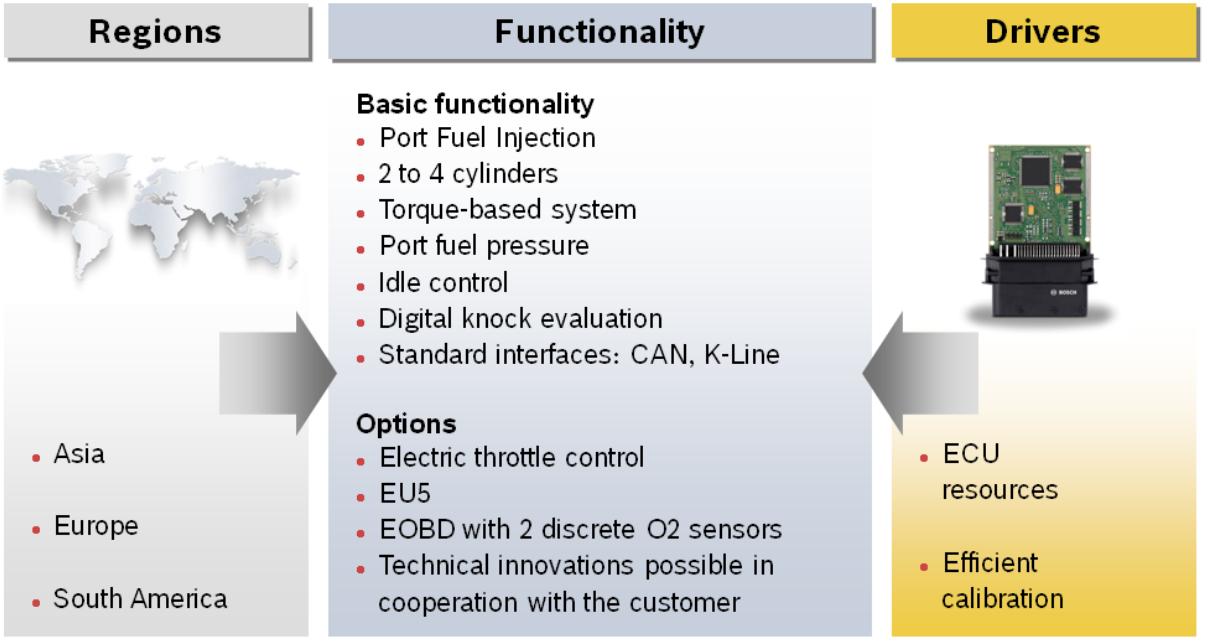


Figura 20 - Funcionalidades de uma ECU. [13]

A estrutura de software da ECU é constituída por três camadas, que para o desenvolvimento deste projeto serão reestruturadas, uma vez que o software é totalmente voltado para veículos, e a intenção é utilizar para controle de funções em edificações. As três camadas podem ser vistas na Figura 21 abaixo:

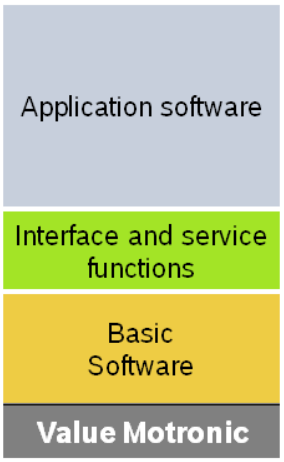


Figura 21 - Camadas de software na ECU. [13]

Por fim, a apresentação das propriedades de calibração, é feita levando em consideração os seguintes itens:

- Utilização protocolo de calibração CAN;
- Uso do processador com memória adicional para calibração;
- Modificação de dados de calibração em memória RAM não utilizada;
- Utilização da ferramenta de calibração INCA do fabricante ETAS.

Devido à grande importância na interoperabilidade da solução que será proposta, um novo conceito deve ser inserido: o protocolo de comunicação CAN.

4.1.1.1: Protocolo CAN

O protocolo CAN foi desenvolvido pela empresa Robert Bosch em 1986 para utilização em tecnologia automotiva, com o objetivo de simplificar os complexos sistemas de fiação em veículos que são compostos por múltiplos microcontroladores. As principais especificações são a elevada taxa de transmissão, grande imunidade a interferências elétricas e a capacidade de detecção de erros.

O CAN é um protocolo de comunicações que permite controle distribuído em tempo real, com elevado nível de segurança. É um sistema em barramento com capacidades multi-mestre, isto é, vários nós podem pedir acesso ao meio de transmissão em simultâneo. Este protocolo comporta também o conceito de *multicast*, isto é, permite que uma mensagem seja transmitida a um conjunto de receptores simultaneamente.

A informação transmitida possui tamanho curto. Assim, cada mensagem CAN pode conter um máximo de oito *bytes* de informação útil, sendo, no entanto possível transmitir blocos maiores de dados recorrendo à segmentação. A taxa máxima de transmissão especificada é de 1 Mbit/s, correspondendo este valor a sistemas com comprimento de barramento até 40 metros. Para distâncias superiores à taxa de transmissão recomendada, o valor diminui. Alguns dos valores recomendados são: 500 Kbit/s para distâncias até 1 km e 125 Kbit/s para distâncias até 500 m. [14]

Assim o protocolo CAN, devido à sua fiabilidade e baixo custo, tem tido bastante aceitação, existindo atualmente vários dispositivos acessíveis no mercado de circuitos integrados. Uma das vantagens do protocolo CAN é a capacidade de suportar o erro. Esta capacidade consiste em retransmitir a mensagem que não aparece corretamente no barramento CAN.

Pode-se concluir que a aceitação do protocolo CAN por parte da indústria de automação se deve a vários fatores, designadamente: [14].

- Baixo custo;
- Capacidade de funcionar em ambientes com condições elétricas adversas;
- Elevado grau de capacidades de tempo real e controle distribuído;
- Fácil utilização;
- Disponibilidade de componentes e controladores com o protocolo CAN;
- Existência de padronização.

Atendendo ao requisito de se usar uma ECU VM para atuar como o controlador lógico programável em edificações, deve-se levar em consideração o fato de a ECU ser comunicável via protocolo CAN, enquanto o software BIS comunica-se via padrão OPC, então a interação entre BIS e ECU deverá traduzir informações de CAN para OPC e vice versa.

Outro ponto de grande importância é a utilização da central eletrônica, que é totalmente voltada para gerenciamento de dispositivos e processos em veículos, situação a qual diverge totalmente do objetivo principal deste projeto. O estudo sobre a ECU levantou pontos que serão discutidos na análise da proposta da solução, apresentando os passos necessários para desenvolvimento dos trabalhos na ECU como modificações no software, e utilização de outros dispositivos externos que auxiliem a ECU a funcionar como um CLP.

Uma unidade de controle eletrônico é responsável por diversas funções dentro de um veículo, sendo algumas de grande risco, ao usuário ou ao próprio veículo, quando realizadas indevidamente. Para assegurar-se contra mau funcionamento a Bosch faz uso de diversos recursos de controle de segurança e de qualidade que promovem à ECU um padrão de confiabilidade conhecido e

renomado mundialmente. Tais controles envolvem desde metodologia de desenvolvimento de softwares, controles de versão, profissionais que trabalham somente para fiscalizar erros, até avançados métodos de simulação e ambiente de testes. Para que uma ECU não tenha sua estrutura de software modificada indevidamente, o acesso às páginas de memória somente se faz possível, mediante uso de um software produzido pela ETAS, empresa do grupo Bosch. O software, chamado de INCA, carrega consigo todos os protocolos necessários para que uma ECU seja gravada, calibrada ou apagada. Assim é de grande importância para o projeto, uma vez que é imprescindível para configurar uma ECU.

4.1.2: INCA

O software INCA, produzido pela ETAS, oferece ferramentas flexíveis para calibração, diagnóstico e validação de sistemas eletrônicos automotivos. O INCA é um software utilizado para desenvolvimento e testes com ECUs, bem como para a validação e calibração de sistemas controlados eletronicamente no veículo, na bancada de ensaios, ou em um ambiente virtual no PC. A ferramenta é usada em todo o mundo em mais de 20.000 instalações em projetos de desenvolvimento e produção. O produto base INCA oferece medição abrangente e funcionalidade de calibração, que podem ser citados como: gestão dos parâmetros de calibração, análise de dados de medição, e programação de memória flash de ECU. [15]

Este capítulo abordará a ferramenta INCA de forma geral, apresentando somente conceitos necessários para entendimento do projeto em questão. A seguir algumas janelas são explicadas com intuito de ambientar o leitor à ferramenta INCA. Os demais conceitos serão apresentados quando necessário e de forma breve, junto com a apresentação do desenvolvimento do software de configuração inicial dos CLP, no Capítulo 5:**.** As janelas mais utilizadas do INCA, para desenvolvimento do projeto, são: a Janela Principal, o Editor de configuração de Hardware, o Ambiente de Experimentos, e a janela de Gerenciamento de Páginas de Memória.

4.1.2.1: Janela principal

A janela principal do INCA, ilustrada na Figura 22, visualiza a estrutura de banco de dados e permite o acesso a todos os itens armazenados nele. Além disso, oferece opção de configurar, modificar e criar os dados de exportação e importação.

Na janela principal, pode-se acessar o Editor de Configuração de Hardware e o Ambiente de Experimentos.

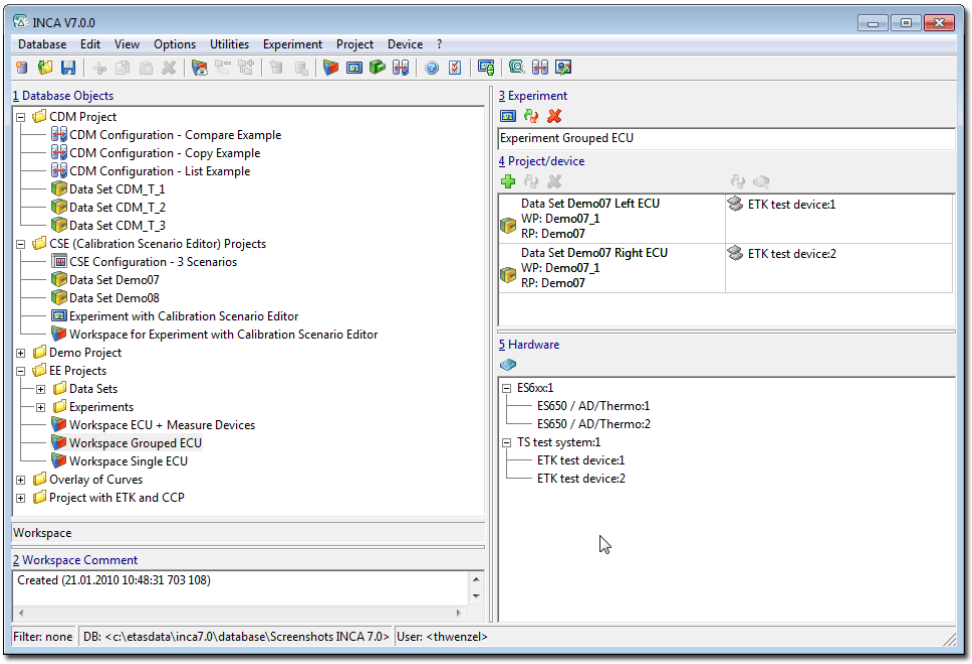


Figura 22 - Janela Principal do INCA. [15]

4.1.2.2: Editor de configuração de hardware

O Editor de configuração de hardware oferece as opções de hardware disponíveis e ligadas a ferramenta INCA, de modo que as comunicações que serão feitas entre dispositivos sejam declaradas neste ambiente. Uma vez selecionados, os hardwares estarão disponíveis para, por exemplo, uma medição ou tarefa de calibração.

A estrutura da árvore visualiza as interconexões entre vários módulos, dispositivos e protocolos. Várias caixas de diálogo proporcionam fácil acesso para dispositivos e módulos de parametrização. Esta janela oferece diversas opções no

que diz respeito à interoperabilidade com hardwares da empresa ETAS (O hardware utilizado no projeto foi o ES592).Um exemplo destas configurações é ilustrado na Figura 23.

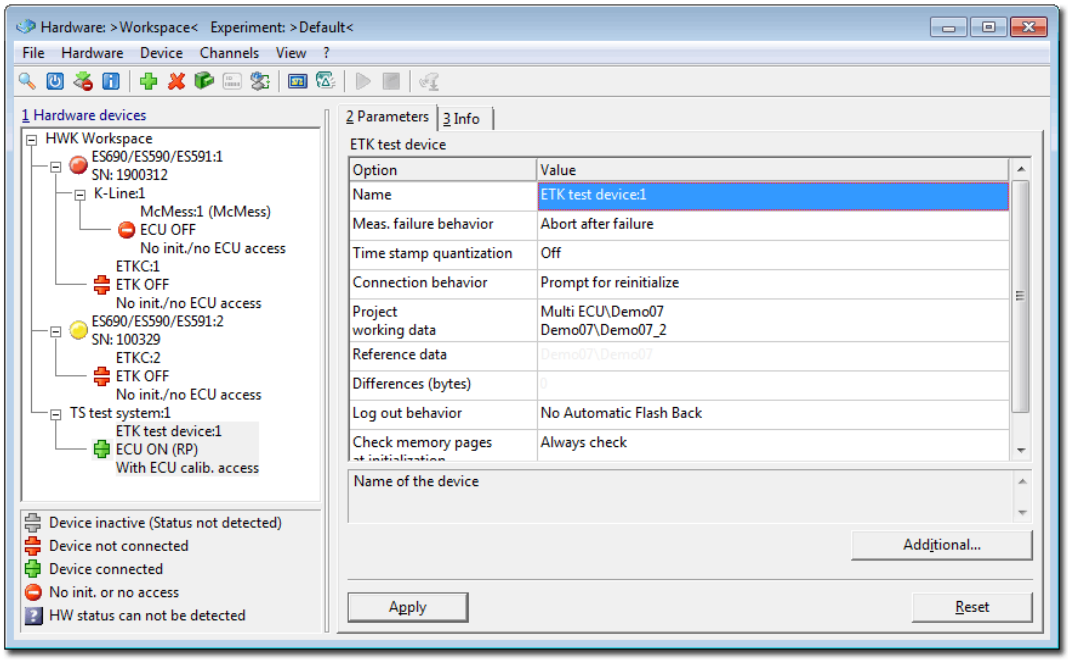


Figura 23 - Editor de configurações de hardware. [15]

4.1.2.3: Ambiente de experimentos

O ambiente de experimentos (Figura 24) fornece a interface do usuário para atividades de medição e calibração. A interface de Experimento pode ser adequadamente configurada para acomodar tarefas individuais. Ela facilita a manipulação simultânea de tarefas de medição e calibração, e também suporta o acesso simultâneo a vários dispositivos.

Este ambiente fornece uma gama de elementos de visualização para melhor aproveitamento dos dados, por exemplo: osciloscópio, display numérico, editor numérico, editor gráfico, editor de cenários de calibração e, além disso, várias camadas permitem fácil gerenciamento de ambientes complexos.

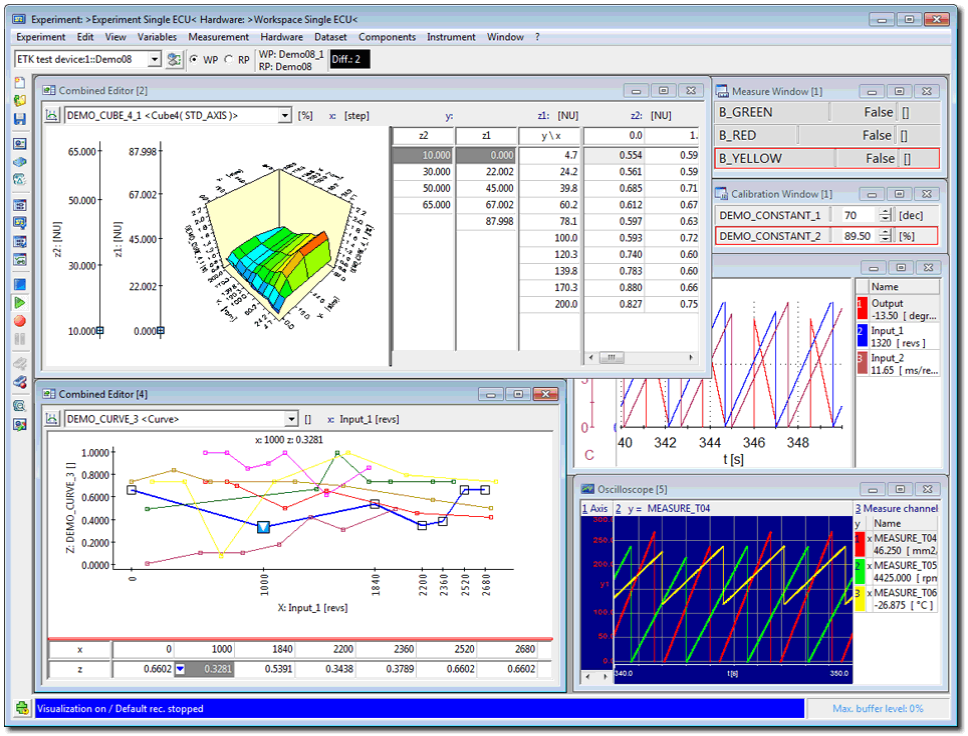


Figura 24 - Ambiente de Experimentos do INCA. [15]

4.1.2.4: Gerenciamento de paginas de memória

Dados de calibração e códigos de programa são armazenados a tanto na ECU, como no INCA, na forma de páginas de memória. A transferência de dados entre essas páginas, por exemplo, do PC para a ECU, é tratado pela janela de Gerenciamento de páginas de memória (Figura 25). Algumas ECUs fornecem opções de gravação de memória podem via protocolo ETK, que é uma interface de transmissão de dados para ECU baseadas em Ethernet, e desenvolvido pela ETAS.

Na janela de gerenciamento as páginas são representadas graficamente na janela, e podem sofrer as seguintes ações:

- Transferência de dados para ECU;
- Carregar dados a partir de ECU;
- Cópia de página de trabalho para a página de referência (serão explicadas futuramente);
- Programação da memória flash na ECU ou no emulador de memória;

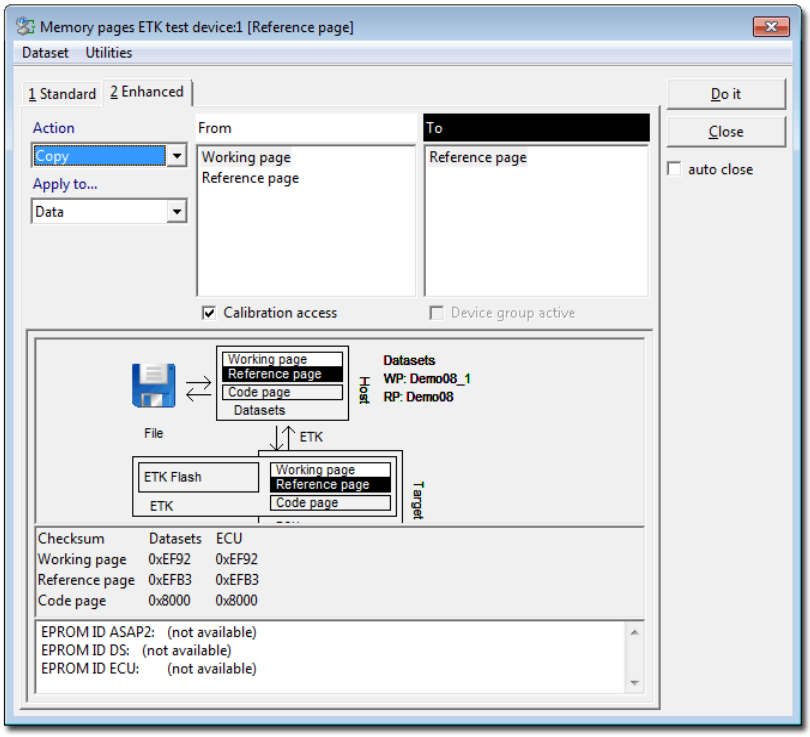


Figura 25 - Janela de gerenciamento de páginas de memória. [15]

É de extrema importância resaltar que o uso do INCA foi feito utilizando *Application Programming Interfaces* (API), ou Interface de Programação de Aplicativos, que nada mais é do um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem se envolver em detalhes da implementação do software, mas apenas usar seus serviços. De modo geral, a API é composta por uma série de funções acessíveis somente por programação e que permitem utilizar características do software menos evidentes ao utilizador tradicional. [16]

4.1.2.5: COM-API

Uma API é usada para invocar funções de um programa a partir de outro aplicativo, e pode também ser usada para a transferência de dados entre diferentes aplicações. Um exemplo é quando se abre uma tabela do Excel em um arquivo do Word para processá-lo. Neste processo, as funções do Excel no Word são executadas através da chamada COM-API (COM, de *Component Object Model*). A COM-API é uma interface padrão fornecida pelos sistemas operacionais Windows. A

COM-API permite ao usuário utilizar as funções de vários aplicativos do Windows de dentro de um aplicativo de destino para cumprir uma tarefa específica. O INCA oferece interfaces específicas para permitir o acesso através da COM-API. [17]

4.2: Análise de custos, diretrizes e proposta

Tendo conhecimento dos tópicos apresentados nos capítulos anteriores, o próximo passo é apresentar a análise dos valores empregados na solução oferecida pela Bosch Security Systems, as hipóteses de solução levantadas e as diretrizes para desenvolvimento e validação do conceito.

De acordo com a problemática apresentada, percebe-se que o principal ponto negativo da solução atual é o alto custo dos componentes. Segundo os dados provenientes da divisão de Security Systems da Colômbia a aplicação completa oferecida pela Bosch apresenta os custos, mostrados na Tabela 1.

Bosch (Principais custos e preço total)	
Custo médio para uma única caixa elétrica	2.700 USD$
Software Winstudio com 2 web clients	1.600 USD$
Custo médio para programação inicial do Winstudio	2.000 USD$
Outros custos	800 USD$
Solução de integração OPC por caixa elétrica	**7.100 USD$**

Tabela 1 - Custo médio de uma solução Bosch. [18]

Deve-se levar em consideração que uma aplicação típica necessita de, no mínimo, quatro caixas elétricas. Para grandes empreendimentos, como complexos prisionais, aeroportos e estádios, o custo pode não ser considerado tão elevado, porém para edificações de médio porte este valor implica à Bosch no encarecimento excessivo dos componentes, uma vez que tais componentes não serão utilizados aproveitando todas as propriedades oferecidas. Isso faz com que a empresa não consiga oferecer uma "solução Bosch" para estes casos, resultando em perda de clientes e não atuação neste mercado. Com isso, a equipe colombiana teve de se adequar ao mercado e continuar vendendo suas soluções, porém baseadas em CLPs de concorrentes.

Com a intenção de reduzir os custos que se traduzem em preços oferecidos aos clientes, três concorrentes foram testados e exibem os seguintes custos:

→ **Moellher** é um fabricante de CLPs de baixo custo e funciona bem com o software BIS.

Moellher	
Custo médio para um único driver	1.100 USD$
Não é necessário comissionamento com programação	-
Custo total da solução por caixa	**1.100 USD$**

Tabela 2 - Custo médio de solução da empresa Moellher. [18]

→ **National Instruments**

National Instruments	
Custo médio para uma única caixa elétrica	6.000 USD$
Custo médio para programação inicial (Labview)	2.000 USD$
Custo total da solução por caixa	**8.000 USD$**

Tabela 3 - Custo médio de solução da National Instruments. [18]

→ **Kepware** é uma solução de software para drivers OPC, e tem bastante representatividade ao redor do mundo.

Kepware	
Custo médio para uma única caixa driver	2.500 USD$
Custo médio para programação inicial do driver	2.000 USD$
Custo total da solução por caixa	**4.500 USD$**

Tabela 4 - Custo médio de uma solução de servidor OPC da Kepware. [18]

A Figura 26 ilustra a situação atual e indica onde serão realizadas as principais mudanças do sistema. A ação de retirar o software Winstudio da solução proposta resulta em uma necessidade de se preencher essa lacuna com outro componente que realize tarefa similar. A retirada do CLP convencional e utilização da ECU implica em reestruturar completamente o método como o controlador será programado, uma vez que a ECU é voltada totalmente para o campo automotivo, e tem restrições de acesso às páginas de memória.

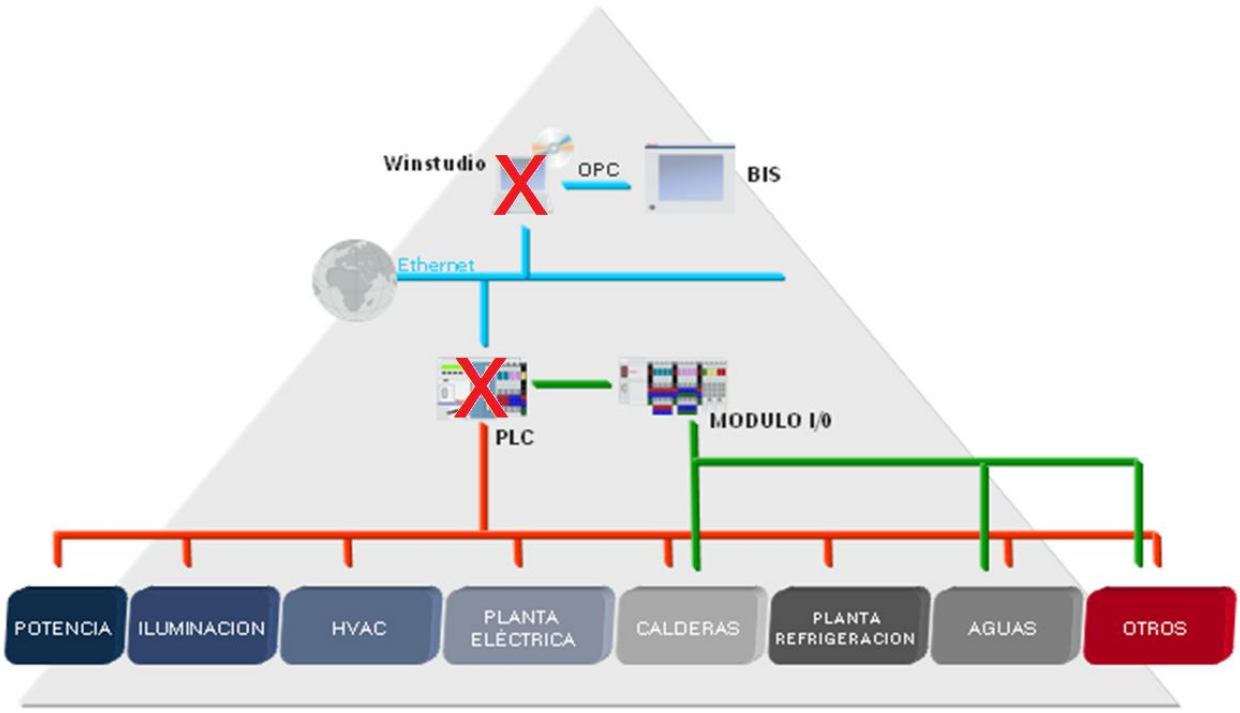


Figura 26 - Solução atual com itens que serão retirados em destaque.[25]

Com base na solução atual oferecida pela Bosch Security Systems foram evidenciados alguns pontos que devem ser trabalhados de modo a desenvolver a solução que irá validar o conceito do protótipo. Pontos os quais foram encarados como as primeiras diretrizes de projeto e podem ser listados como segue:

- Utilizar o BIS como IHM;
    - Criar Interface de visualização no BIS;
    - Criar interface de envio de parâmetros no BIS;
- Comunicação via padrão OPC entre BIS e CLP
    - Server
    - Conversor
- Utilizar como CLP o hardware de uma ECU.
- Desenvolver Software de Configuração Inicial da ECU;

A seguir, na Figura 27, é possível visualizar um esboço da estrutura de desenvolvimento da proposta.

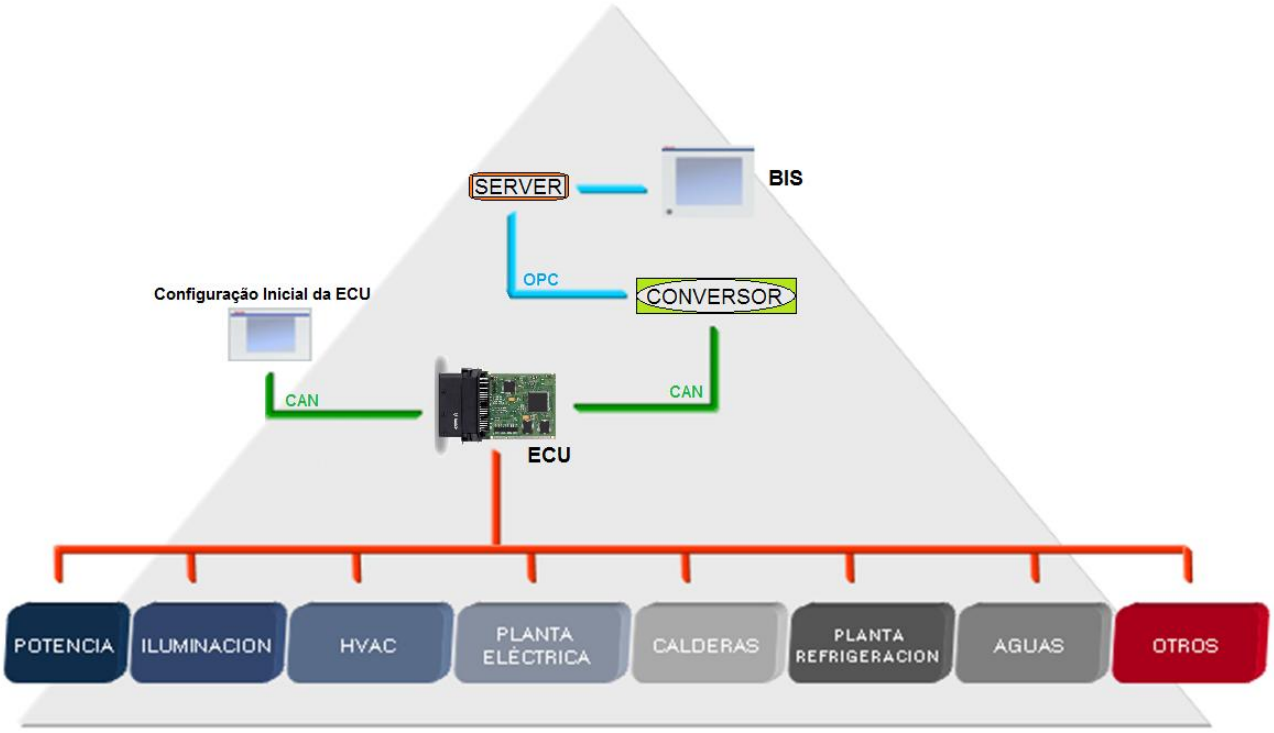


Figura 27 - Estrutura da solução proposta. [25]

4.2.1.1: Diretrizes para o BIS

Atualmente, o software que atua como interface com o usuário final do CLP é o BIS, que atua realizando a supervisão e o monitoramento dos sistemas de controle de edificações. Tais sistemas se dão geralmente por controles de incêndio, de acesso, entre outros. O nível de atuação do usuário se dá pela observação e monitoramento do ambiente controlado, restando poucos recursos de atuação efetiva no controle, uma vez que os parâmetros são pré-programados na instalação e não são modificados constantemente. Na Figura 28, podem ser observadas duas interfaces de controle do BIS, uma mostrando a planta de uma edificação e a outra a ferramenta de câmeras de segurança e controle de acesso.

Figura 28 - Exemplo de interfaces BIS. [22]

O BIS é um software Web que quando utilizado é aberto em navegadores de internet. Na sua ultima versão suporta algumas aplicações de terceiros, desde que sejam desenvolvidas em ambiente web (Dojo, DotNet, Frontpage e etc.). A solução encontrada para solucionar o obstáculo do controle mais dinâmico foi desenvolver duas interfaces diferentes, uma que será somente para visualização gráfica, e outra para o envio dos parâmetros ao controlador, ambas em algum ambiente desenvolvedor Web, provavelmente o Dojo que já tem uma arquitetura aceita pelo BIS.

O Building Integration System é um programa pesado que tem uma estrutura de software nos padrões Bosch, sendo assim existem diversos controles de segurança que dificultam a modificação para desenvolvedores externos. As configurações necessárias para instalação do BIS são as seguintes:

Required hardware for Server/Client	**Required software for Server/Client**
– CPU 3GHz Pentium 4 or Intel Core 2 Duo 2.66 GHz or higher – 4 GB RAM (server and client) – 80 (Server) / 20 (Client) GB available space – VGA with 1280x1024, 32k color, 256 MB – 100 Mbit/s or higher Ethernet card (PCI) – DVD drive (Server only) – 1x USB or parallel port for dongle (Server only)	Operating system: – Windows XP Professional SP3 (32 bit)*** **** – Windows 7 SP1 (32/64 bit, but for the BIS Server not Home or Starter editions) – Windows 2003 Server SP2 (32 bit)*** **** – Windows 2008 Server R2 SP1 (64 bit)*****

Tabela 5 - Requisições para instalação do BIS. [11]

A necessidade de se ambientar com o BIS e testar opções de servidores OPC terceirizados fez com que fosse preciso instalar o software em um computador da

Bosch Curitiba, porém um problema foi encontrado. Tal problema se dá pela incompatibilidade do controle de segurança, que a Bosch utiliza dentro da empresa, com o BIS. A Bosch utiliza um programa chamado Peacy que realiza diversas tarefas como configuração, atualização, reparos e controles de acessos dos computadores dos colaboradores. Esse programa não permite que o BIS rode perfeitamente, sendo assim foi necessário adquirir uma máquina que rodasse com sistema operacional sem Peacy, uma liberação da empresa para trabalhar com tal máquina, e uma liberação para utilização de mídias removíveis para transferência de arquivos e programas, uma vez que o novo computador não utiliza o software de controle de segurança e não pode se conectar a rede Bosch.

Com intuito de ter o conceito do protótipo validado dentro do prazo estipulado, os trabalhos com o BIS se resumiram a ambientação, instalação, e estudo sobre as possibilidades de desenvolvimento das interfaces e conexão com servidores. As principais tarefas realizadas para validação do conceito foram feitas em cima do estudo das possibilidades de comunicação entre BIS e ECU, desenvolvimento do software da ECU e da configuração inicial.

4.2.1.2: Comunicação CAN - OPC

A comunicação entre o protocolo CAN e o padrão OPC é, provavelmente, a questão que será mais problemática do ponto de vista da confiabilidade das informações. Por esse motivo, nessa etapa do projeto, produtos como servidor e conversor serão adquiridos de terceiros. Isso se dá pela consistência na qualidade de troca de informações oferecida por empresas especializadas.

A interoperabilidade entre CAN e OPC era prevista de se resolver utilizando algum tipo de conversor stand-alone, ou algum hardware que realizasse a tradução entre ambos os protocolos. Porém após alguns estudos pode-se perceber que as diferenças entre os dois padrões são grandes e a principal diferença é que o protocolo OPC foi desenvolvido para trabalhar em sistemas operacionais Windows e funciona em tempo real, enquanto que o CAN é uma comunicação de mais baixo nível.

Após alguns estudos e discussões sobre essa questão, a ideia será converter o protocolo CAN para OPC utilizando um servidor CAN-OPC que rodará na mesma máquina do BIS, e um conversor para o meio físico a ser ligado no computador. A necessidade de se utilizar este conversor existe, pois o meio físico que o CAN atua não pode ser muito extenso, com pena de perder informações no meio do caminho, como citado no embasamento conceitual da ECU. Futuramente no projeto, quando for produzido em série, o protocolo de comunicação será o IP, uma vez que atualmente existe uma grande demanda por esse tipo de comunicação. Isso poderia ser feito nessa etapa de projeto, mas para validação do conceito o conversor CAN-OPC será suficiente e exigirá menos esforço por parte do time. O motivo de citar essa opção é por ela ser uma segunda opção caso a primeira ideia não seja implementável.

Visando selecionar a melhor proposta para o projeto algumas empresas de servidores OPC e conversores foram pesquisadas e suas avaliações são descritas a seguir, bem como as combinações de servidores e conversores que melhor se encaixam aos pré-requisitos (Figura 29).

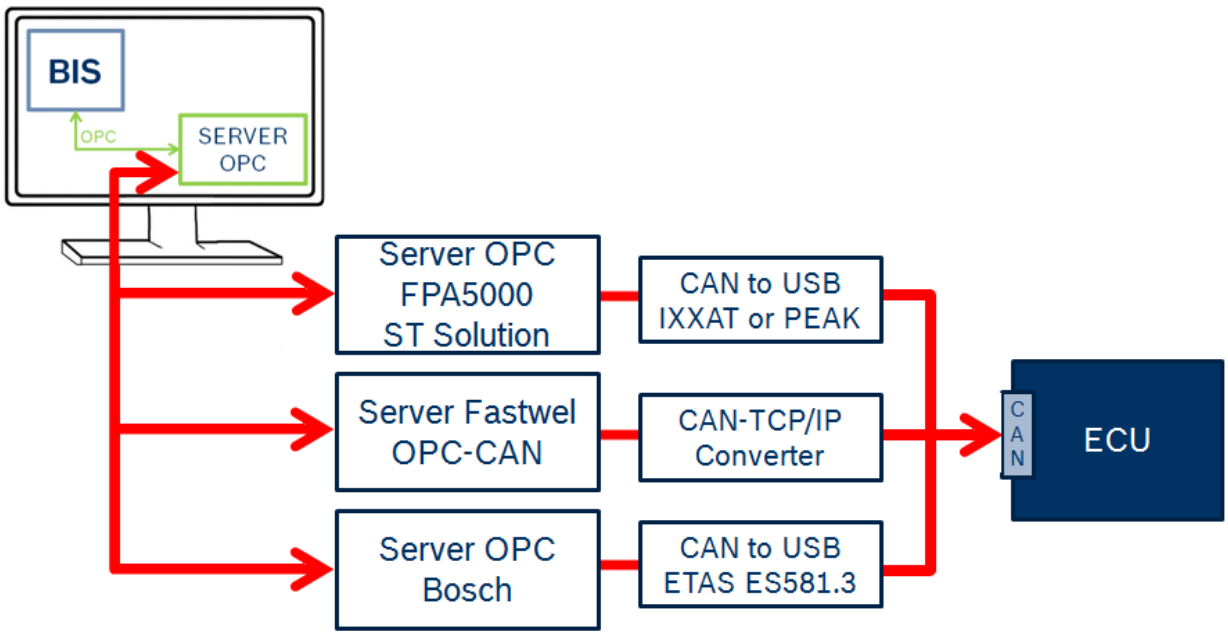


Figura 29 - Opções de servidores e conversores.[25]

- ***Server IP-OPC utilizado na FPA5000*+ Conversor CAN-IP**

Esse servidor já é utilizado pela Bosch Security Systems de Campinas para controle de centrais de incêndio (FPA5000). Com isso, a experiência necessária para instalar e configurar o servidor já é de domínio da Bosch e o hardware (FPA5000) atua de maneira semelhante no que diz respeito ao recebimento de mensagens no BIS.

Segundo Mauricio Santos, responsável pela área de Campinas, esse servidor utiliza o protocolo TCP/IP, que é bastante desejado pelos usuários dos sistemas vendidos por ele. Porém como estamos tratando de uma ECU se faz necessário um conversor CAN-IP.

Quando se utiliza os sistemas de controle de incêndio FPA5000, o hardware deste dispositivo já tem acoplado um conversor na própria placa, o que no caso de uma aplicação deste tipo, a compra de um conversor não se faria necessário. Em todos os outros casos a compra de um conversor CAN-IP é oferecida por diversas empresas e custa entre US$200,00 a US$300,00.

- ***Server CAN-OPC Bosch para o conversor ES581.3 da empresa ETAS.***

Utilizar o conversor seria muito simples, uma vez que transporta a informação da CAN para o PC via USB, e já é bastante utilizado pela BEG Latin America.

O servidor CAN-OPC Bosch ainda não foi desenvolvido, e esta opção implica na necessidade de se ter mão de obra especializada, e maior esforço do time, o que não foi planejado pelo responsável.

- ***Server CAN-OPC da Fastwel* + Conversor CAN-USB da IXAAT**

A empresa Fastwell fornece soluções para servidores CAN-OPC, e indica um conversor CAN-USB compatível, que é produzido pela empresa IXAAT. A escolha por esse servidor não implica no sucesso da comunicação, porém em contato com o fornecedor, a troca de informações direciona para o

sucesso da operação. O ponto negativo é justamente a falta de experiência com a ferramenta, porém a empresa responsável oferece suporte em todos os aspectos referentes ao produto.

Para esta etapa do projeto, onde a validação do conceito é o principal objetivo, será usado um conversor CAN-USB que oferece maior facilidade para realização de testes, uma vez que o protótipo será montado em uma bancada, onde todos os componentes estarão próximos uns dos outros. Porém quando tal protótipo se transformar em produto essa solução deverá ser repensada, uma vez que a distância entre a IHM e a ECU será bastante acrescida, devido ao fato do controlador se localizar na caixa de controle da edificação e a IHM em um local específico, distantes um do outro.

4.2.1.3: Desenvolvimento de Software de Controle na ECU

O software a ser gravado na unidade de controle eletrônico será realizado seguindo algumas funções pré-programadas de controle que serão disponibilizadas em blocos dentro da ECU. Esses blocos quando forem interligados podem gerar estruturas de controle diversas, atendendo assim a demanda do cliente para cada aplicação que for desejada.

A Figura 30, abaixo, exemplifica a ideia de controle da ECU, onde os set-points serão provenientes da interface humana-máquina. Vale lembrar que na ilustração não é definido o tamanho da estrutura de controle que será utilizada no final, podendo variar de 3 até milhares de blocos, uma vez que ainda não foram feitos os estudos para definir a máxima abrangência de atuação do controlador, tarefa que não tem urgência nesta etapa de validação do conceito.

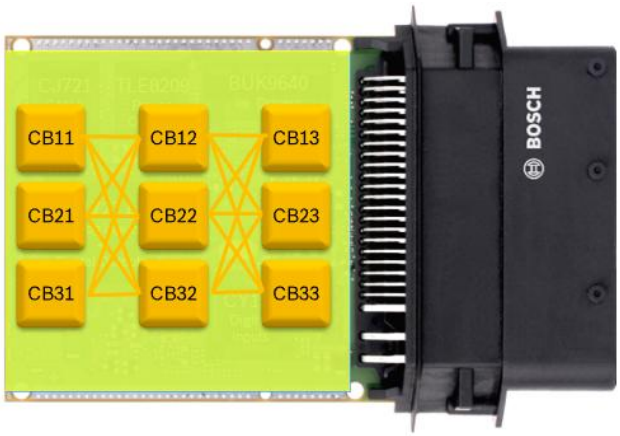


Figura 30- Estrutura de Controle da ECU

A ECU vai receber um programa com a estrutura de software mostrada na figura acima, portando diversos tipos de controle que podem ser combinados de diferentes maneiras dependendo da configuração feita inicialmente durante a instalação da aplicação. O software base será desenvolvido em cima da estrutura dos 3x3 blocos e poderá ser configurado via interface de usuário. Cada bloco da matriz pode ser configurado independentemente e poderá conter uma diferente estrutura de controle, como por exemplo: PI, PID,PD, OR, NOR,XOR, e muitas outras funções que podem chegar até a 40 diferentes possibilidades.Os blocos CBx1 podem atuar como entradas de acordo com a configuração do usuário, bem como os blocos CBx3 podem atuas como saídas.

A configuração inicial, definida de acordo com a necessidade do cliente, será gravada na memória EEPROM da ECU. Para realização dessa tarefa faz-se uso do software de gravação de memória de ECU chamado INCA, apresentado anteriormente. Esse software é o único capaz de realizar a tarefa de gravar na memória da ECU, devido às liberações de controle de segurança exigidos para escrita e leitura de dados de uma unidade de controle eletrônico. Isso traz um ônus ao projeto uma vez que a licença para tal software deve ser adquirida para realização da configuração inicial dos CLPs

Pelo ponto de vista do usuário, deseja-se que este possa ser capaz de realizar a configuração de maneira simples e eficaz. Sabendo que o software INCA foi desenvolvido calibradores de ECUs, sua interface é não é muito amigável para

quem não a domina, além de que ela é geralmente desenvolvida para calibração de unidades utilizadas em automóveis, que não é o presente caso.

4.2.1.4: Desenvolvimentos de Software

Baseando-se nos conceitos, estudos e decisões de cada especificação do sistema apresentados até então, o desenvolvimento efetivo dos trabalhos do projeto pode ser dividido em dois campos: Monitoramento/Controle, e Configuração.

A Figura 31, abaixo, ilustra o campo de monitoramento e controle de uma aplicação do novo CLP, onde o software BIS atua como interface humana-máquina, enviando e recebendo os parâmetros através do servidor OPC da empresa Fastwell. Este, por sua vez, transmite as mensagens através de um conversor de protocolos até a porta CAN da ECU Value Motronic.

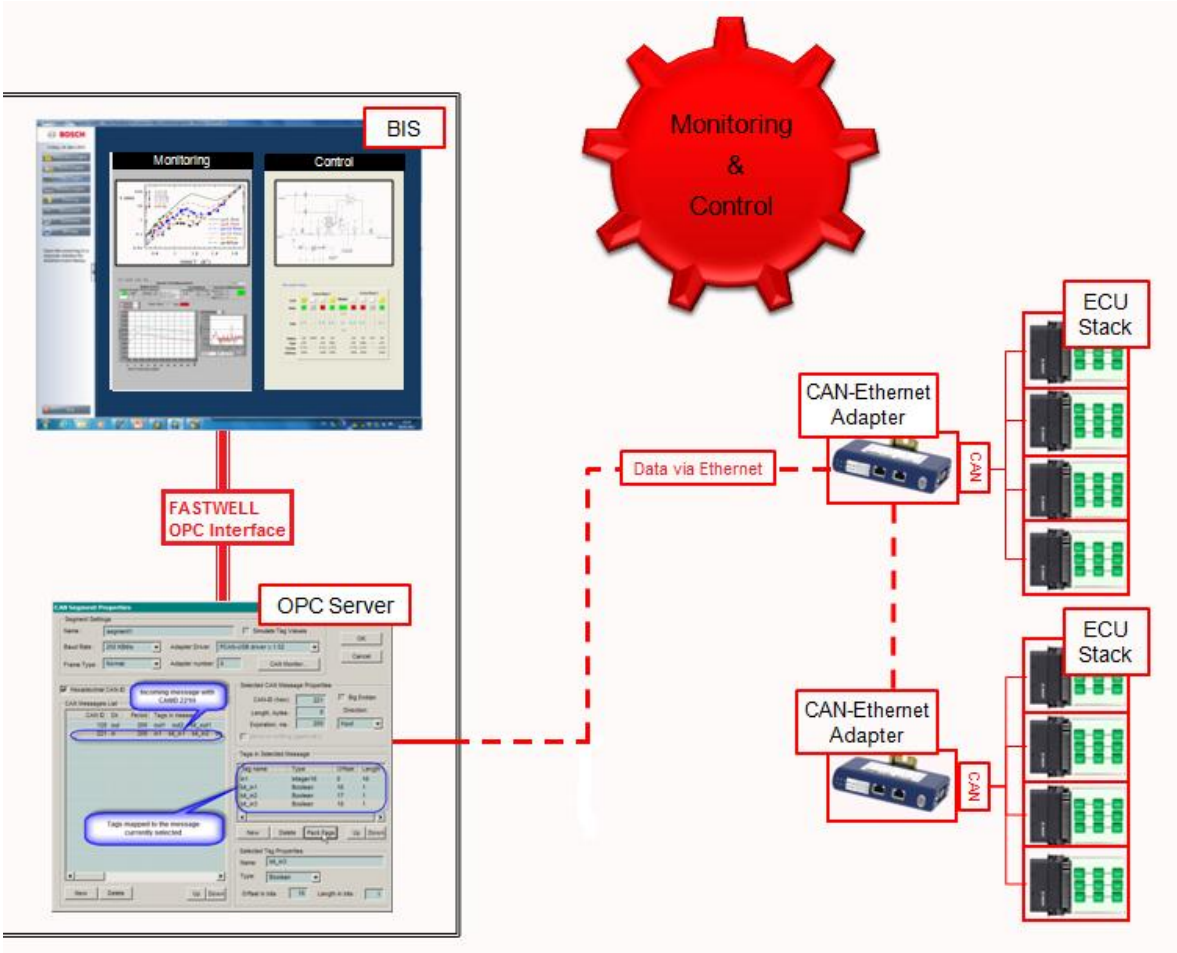


Figura 31 - Estrutura do monitoramento e controle. [25]

Uma particularidade que ainda não foi citada é o fato da Value Motronic já possuir uma estrutura de comunicação CAN-CAN, ou seja, caso seja necessário ligar mais de uma ECU ao BIS, essa ação já tem seu caminho facilitado. O que, no caso deste projeto, se faz extremamente necessário, devido à ECU não possuir tantas entradas e saídas como um CLP normal.

A Figura 32 abaixo ilustra os componentes eletrônicos e as opções de entradas e saídas de dados e informações, onde devem ser destacadas as 12 entradas digitais, 9 entradas analógicas e as 5 saídas digitais. Lembrando que as 9 entradas analógicas podem ser transformadas em digitais via software, caso seja necessário.

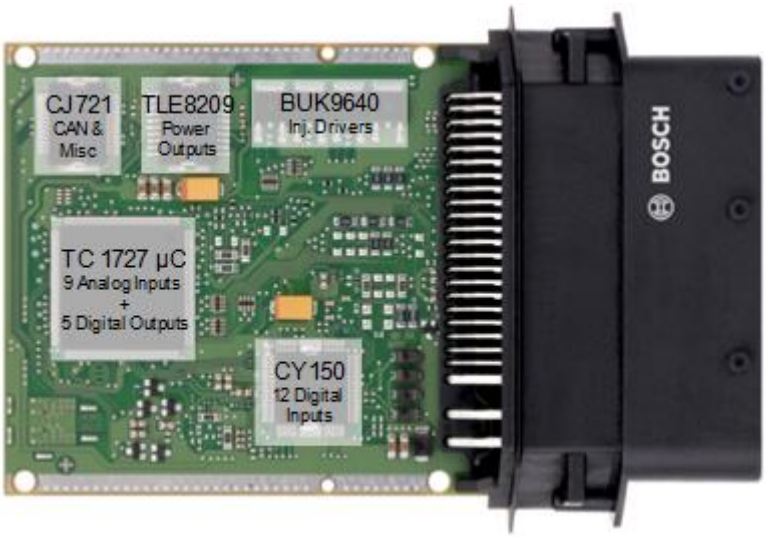


Figura 32 - Componentes eletrônicos da EC. [25]

O trabalho efetivo em cima da ECU foi feita com auxílio do Engenheiro de Software, Filipe Cordeiro. As tarefas relacionadas à central eletrônica foram: a retirada das partes automotivas, a limpeza de funções que não serão necessárias, bem como a declaração dos vetores, de modo a receber os dados provenientes da configuração inicial a ser realizada pelo aplicador.

Diante de tais dados, a diretiva de se desenvolver um software que pudesse realizar a configuração inicial do novo CLP foi colocada em ação. A Figura 33 ilustra o campo de Configuração, e a forma como irá funcionar.

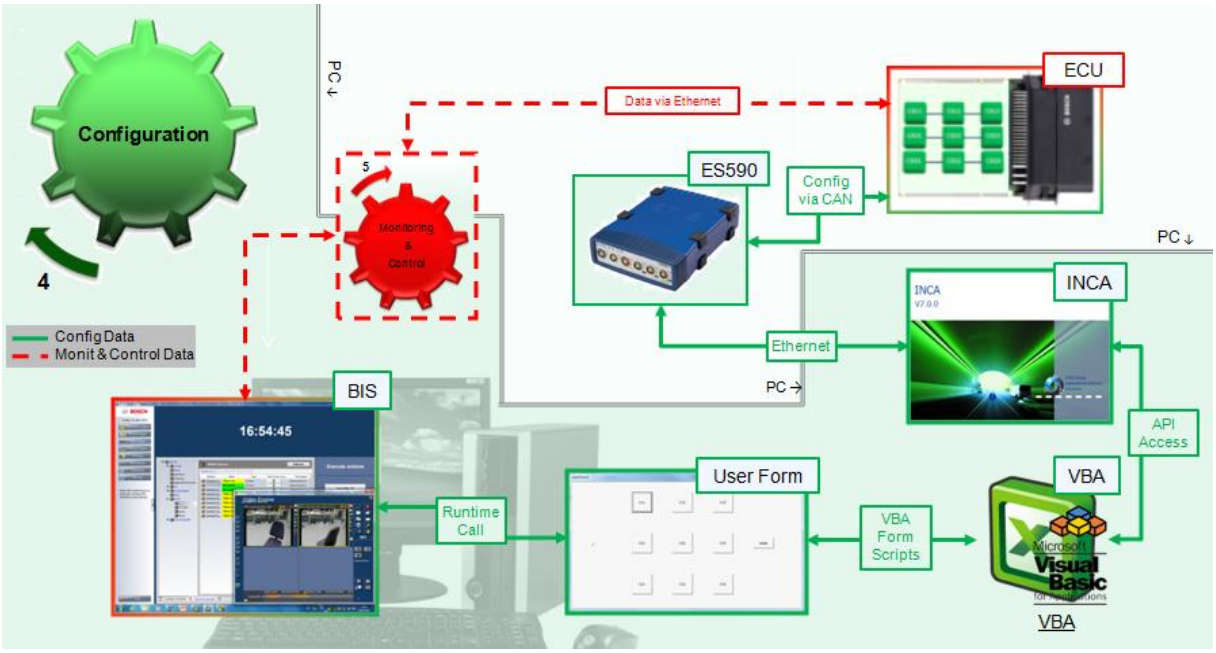


Figura 33 - Estrutura de configuração inicial

Na ilustração podem ser vistos dois caminhos, o do monitoramento/controle, e o da configuração. A configuração inicial do CLP será realizada utilizando as APIs do software INCA, que como citando anteriormente, é o único capaz de inserir a malha de controle desejada dentro da memória da ECU. Para tanto, com objetivo de diminuir custos relativos ao protótipo, a escolha do ambiente de programação foi feita com base na facilidade e no custo de tal software. Alguns programas já possuem implementações utilizando os APIs do INCA, dois deles são o *Matlab* e o *Microsoft Excel*. No caso do *Matlab*, a licença para utilização tem grande custo e foi cortada da proposta inicial de validação do conceito, não sendo descartada como uma opção futura. Então, o programa Excel, da Microsoft, foi escolhido por ser facilmente encontrado em diversos computadores, e ter licença menos custosa que a do *Matlab*. O próximo capitulo irá descrever todo desenvolvimento realizado em cima do Software de Configuração Inicial (PICS, do inglês "*PLCs Initial Configuration Software*").

Capítulo 5: Software de Configuração Inicial de CLPs(PICS)

Sabendo que o software INCA foi desenvolvido para pessoas capacitadas e com embasamento em calibração de ECUs, a interface do mesmo é bem complicada aos olhos de quem utiliza pela primeira vez. Para utilizar o software o usuário precisa ter treinamento e possuir conhecimento relacionado ao ramo automotivo, uma vez que o uso é geralmente direcionado a calibração de unidades eletrônicas utilizadas em veículos. Assim, foi desenvolvido um software em VBA (Visual Basic for Applications), dentro da ferramenta Microsoft Office Excel, que tem como objetivo realizar duas funções básicas, nessa ordem:

1) Realizar a configuração desejada de forma gráfica;
2) Gravar a configuração na memória EEPROM da ECU.

Para realizar s segunda função, algumas APIs do INCA foram utilizadas e serão explicadas posteriormente, mas para oferecer um panorama, elas resumidamente realizam de forma automática funções como:

- Abrir o INCA;
- Selecionar a database;
- Checar se a database está aberta;
- Abrir o experimento;
- Abrir variável de calibração;
- Ler variável de calibração;
- Escrever na variável de calibração;
- Reset do Device;
- Fechar ambiente de experimento;
- Reset dos parâmetros;
- Criar o arquivo”.hex" de flash
- Gravar o ".hex" na EEPROM
- Selecionar Working Page;
- Selecionar Reference Page;

Já para o entendimento da primeira função, esta pode se desmembrar em diversos tópicos importantes e terá a explicação dividida de acordo com os formulários desenvolvidos. Na ilustração hierárquica da Figura 34 observa-se que a janela (formulário) principal do PICS chama as demais janelas quando necessário. Assim a explicação será dividida por formulário, e dentro de cada formulário serão

apresentados tópicos como mostrados na Tabela 6. O diagrama de caso de uso modelado para o software pode ser visto no “APÊNDICE 2”, no final deste documento.

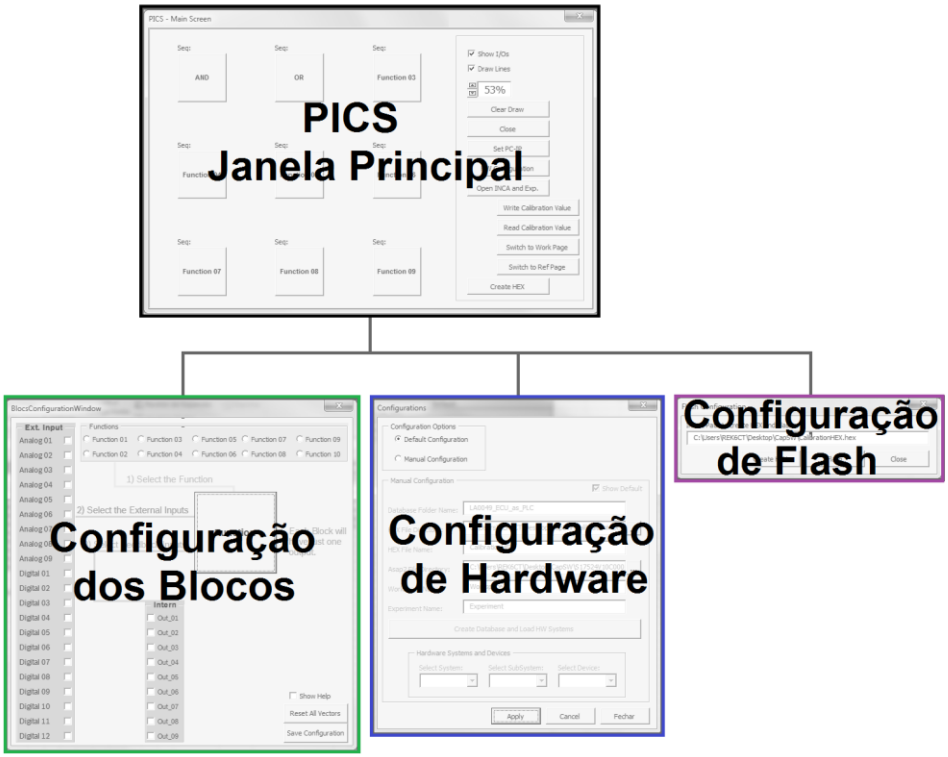


Figura 34 - Estrutura das janelas do software PICS. [25]

FORMULÁRIO	TÓPICOS
TELA PRINCIPAL	• Parte Gráfica Principal • Explicação da Biblioteca • Desenho das linhas • Desenho das IOs • Sequência de Seleção • Demais Botões • Configuração dos protocolos de comunicação
CONFIGURAÇÃO DOS BLOCOS	• Explicação do formulário • Desenho das IOs • Opções de ajuda • Parâmetros de Calibração • Demais Botões
CONFIGURAÇÃO DE FLASH	• Explicação do formulário • APIs mais importantes • Control Flow
CONFIGURAÇÃO DE HARDWARE	• Apresentação de conceitos do INCA • Panorama geral de uma configuração do INCA • Como criar e modificar uma configuração

Tabela 6 - Tópicos de cada formulário. [25]

5.1: Formulário Principal

5.1.1: Parte gráfica

A tela inicial do formulário em VBA visa principalmente demonstrar graficamente os caminhos que serão feitos pelos valores de entradas externas, saídas internas e saídas externas.

A intenção dessa tela é ter aparência similar à estrutura de software que será inserido na ECU (Figura 30), padronizando a linha de pensamento tanto para desenvolvimento do projeto, como para o entendimento do aplicador em campo. Sendo assim, a parte gráfica principal tem formato de matriz e possui nove blocos dispostos em linhas e colunas de três blocos cada. É nesta tela que o usuário irá definir a malha de controle que será utilizada na aplicação desejada. Para tanto o usuário deve clicar em cada bloco, o qual irá encaminhá-lo para um novo formulário. Este, por sua vez, oferecerá opções para realização da configuração individual de cada bloco e terá sua explicação apresentada no formulário de Configuração dos Blocos. A Figura 35 exemplifica uma aplicação com realimentação, que será considerada para explicação do desenvolvimento efetuado.

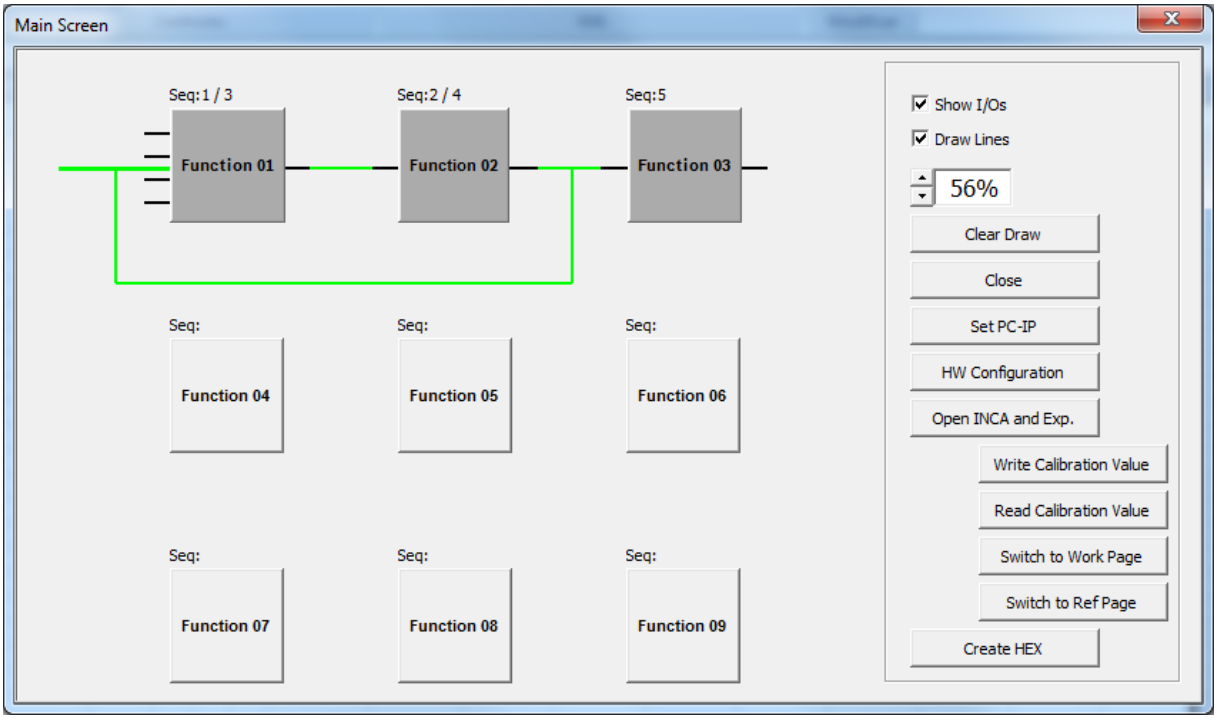


Figura 35 - Parte gráfica principal. [25]

5.1.2: Biblioteca de desenho

A biblioteca de desenho utilizada é um módulo de classe que tem algumas funções gráficas pré-programadas, e facilitam a sua utilização. Dentro das principais tarefas dessa biblioteca está a definição do formulário onde o gráfico será desenhado, a inicialização e finalização da classe, e as funções pré-programadas que desenham linhas, caixas, efeitos de texto, curvas, formas livres, elipses e círculos.

5.1.3: Desenho das linhas entre os blocos

Para traçar as linhas que fazem as conexões entre os blocos, utilizamos as funções da biblioteca de desenho citada anteriormente, sendo uma em particular:

Linha(Posição_Inicial_X, Posição_Inicial_Y, Posição_Final_X, Posição_Final_Y)[2]

A função “Linha” é utilizada para desenhar o caminho entre os blocos. A premissa para o tratamento da rotina de desenho das linhas é de que qualquer bloco pode ser ligado a qualquer outro bloco, inclusive nele mesmo. Outra especificação é que o desenho deve diferenciar entradas e saídas dos blocos. Para tanto percebeu-se que o caminho entre a saída de um bloco e a entrada de outro poderá ser feito no melhor caso, utilizando um movimento, e no pior caso 5 movimentos.

Explicando melhor, na Figura 36 nota-se que para fazer a conexão entre a saída do bloco 1 com a entrada do bloco 2, precisa-se de somente um movimento. Já na Figura 37, três movimento foram necessários para ligar blocos em colunas adjacentes, um movimento para sair do bloco 1, um movimento para atingir a altura do bloco 5, e um último movimento para entrar no bloco 5.

[2] *Pseudo-código*

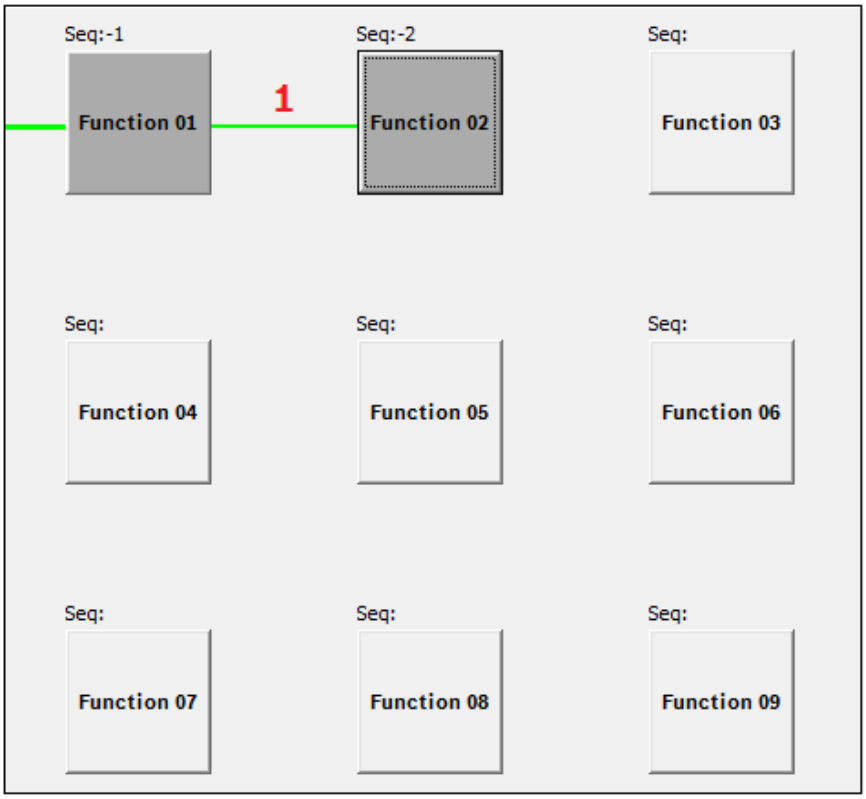


Figura 36 – Conexão com 1 movimento. [25]

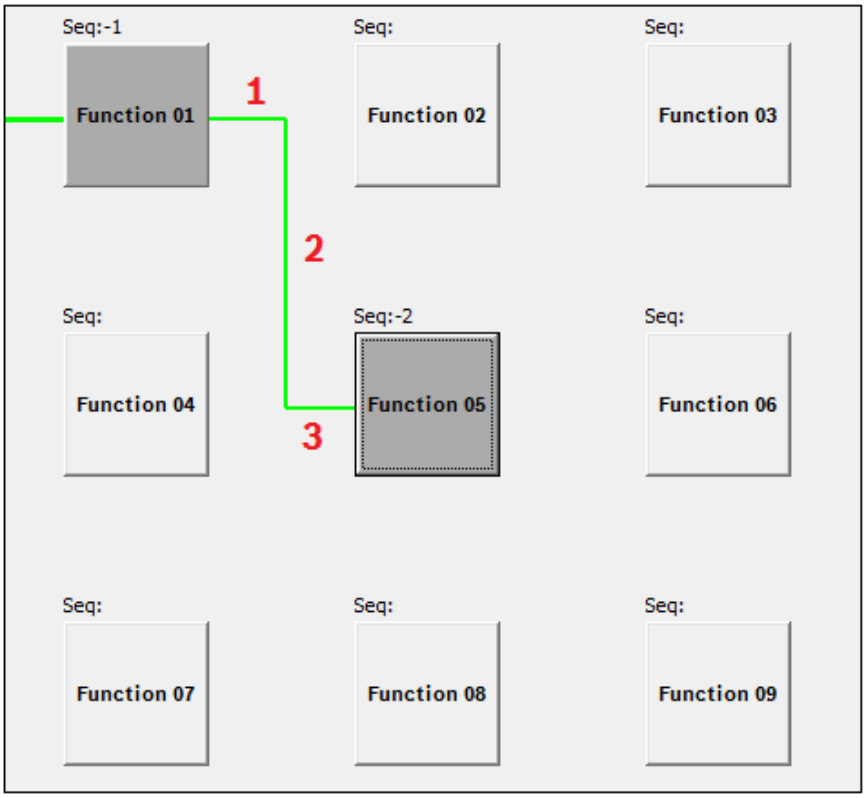


Figura 37 – Conexão com 3 movimentos. [25]

No caso da Figura 38, foram necessários cinco movimentos para ligar blocos em colunas distantes, são eles:

- Um movimento para sair do bloco 1;
- Um movimento para atingir a altura superior (definido como padrão) do bloco 9;
- Um movimento para atingir a distância horizontal até o bloco 9;
- Um movimento para atingir a altura do bloco 9;
- Um último movimento para entrar no bloco 9.

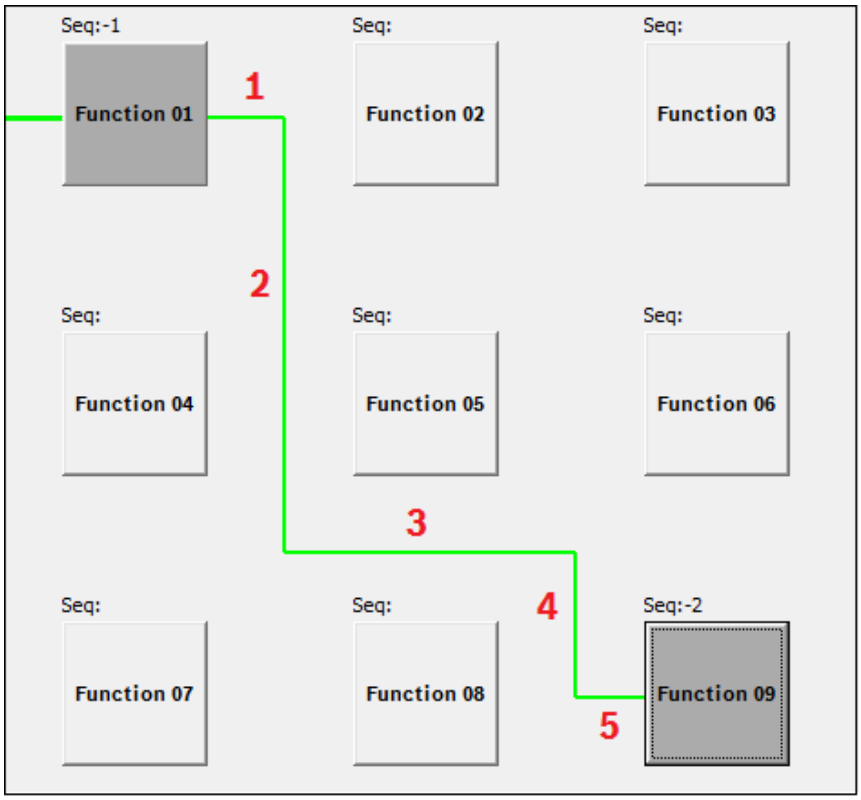


Figura 38 - Conexão com 5 movimentos. [25]

Da mesma forma que para o caso anterior, no caso de uma realimentação também serão necessários 5 movimentos, como pode ser observado na Figura 39, a seguir:

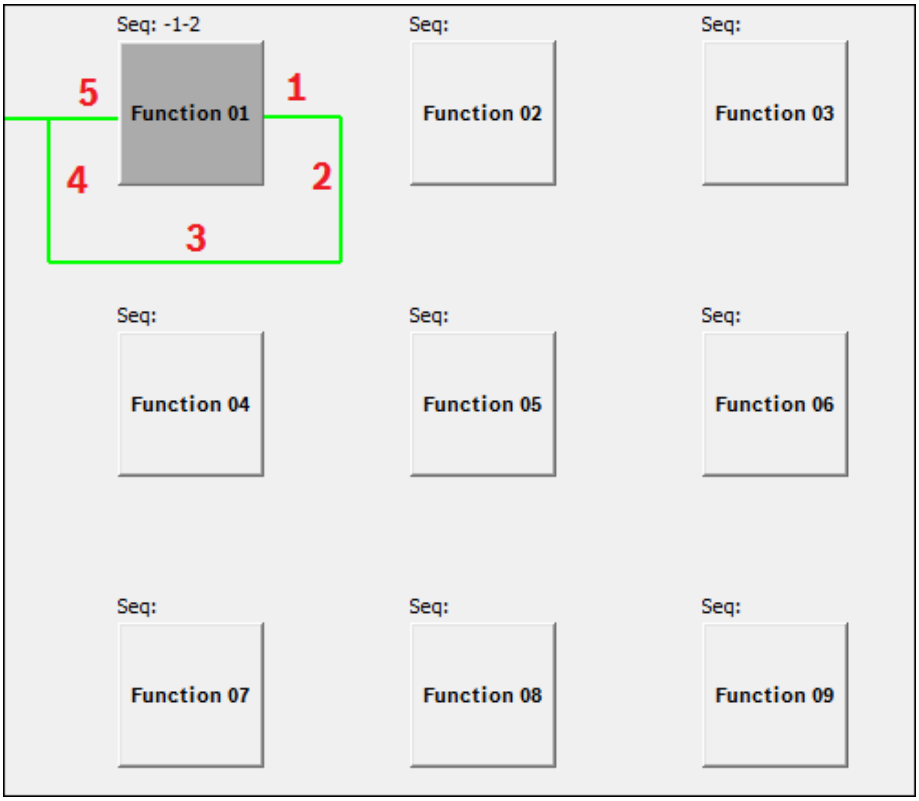


Figura 39 - Conexão no mesmo bloco (Realimentação). [25]

A realização do desenho da parte gráfica sofre com alguns obstáculos que são impostos pela biblioteca de desenho do Excel. Como será visto adiante, na Janela de Configuração dos Blocos também há a utilização de gráficos da mesma biblioteca, assim, a cada troca de janelas, a biblioteca de desenho define o formulário atual como o plano de fundo do desenho, fazendo assim com que não exista uma classe que seja um *Holder* do desenho do formulário anterior. Assim, a cada vez que se configura um bloco, abre-se uma segunda tela que fica com o foco da classe de desenho, e então ao voltar à tela gráfica principal, as conexões entre os blocos precisam ser apagadas antes de serem desenhadas novamente, de modo a redefinir o plano de fundo da janela principal como o ambiente de desenho da biblioteca. Isso gera uma demora na rotina de desenho que será apresentada a seguir.

A rotina de desenho leva em consideração a otimização das linhas de código programadas. Por isso, definiu-se como padrão que toda conexão entre blocos atenderia ao pior caso dos movimentos, no caso, 5 ações.

Para que a conexão entre os blocos atenda sempre à rotina de cinco movimentos, foram definidos três parâmetros que servirão de base para a movimentação entre blocos. A Figura 40 exemplifica um caso de 5 movimentos, com os parâmetros “a”, “b” e “o”.

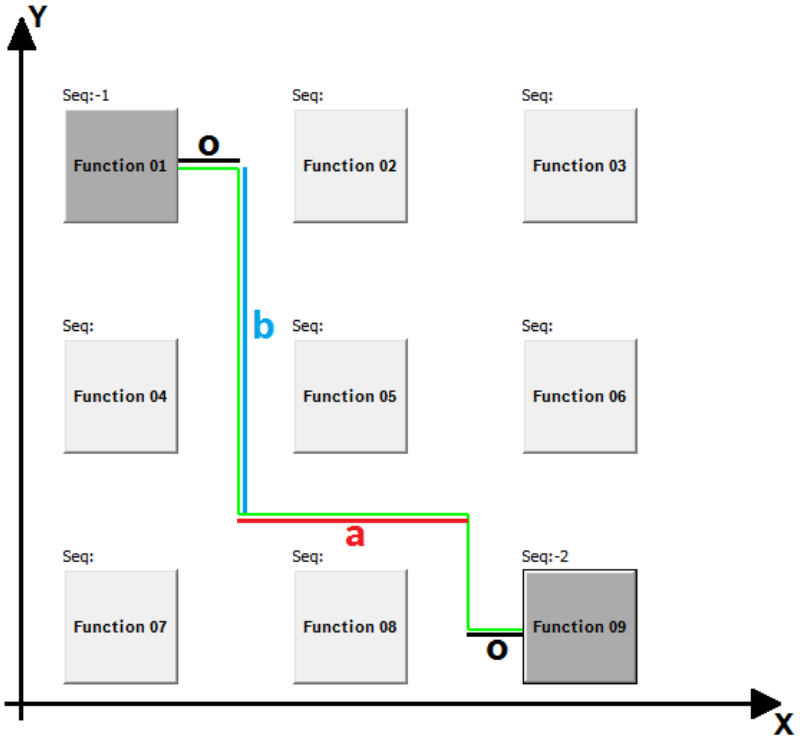


Figura 40- Parâmetros de ajuste para rotina de conexão entre blocos. [25]

Sempre o primeiro e último movimento estão relacionados com o parâmetro “o”, que levam a linha do centro do bloco ao ponto entre dois blocos, que é o local onde se iniciam os movimentos relacionados aos parâmetros "a" e "b". O mesmo acontece para o último movimento, onde a linha é desenhada a partir do local onde distância para o bloco final é o próprio parâmetro “o”, e vai até o centro do bloco final.

Os três movimentos intermediários são os mais importantes, os quais envolvem os outros dois parâmetros. O parâmetro “a” está relacionado com o eixo X, onde o valor do parâmetro é a distância horizontal entre o primeiro bloco e o último bloco menos duas vezes o parâmetro "o", como exemplifica a figura 41.

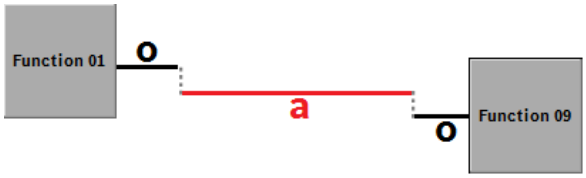


Figura 41 - Parâmetro "a". [25]

Já o parâmetro “b” está relacionado ao eixo Y e tem seu valor atribuído dependendo da distância vertical entre os blocos. No caso dos blocos estarem em colunas adjacentes, o valor do parâmetro “b” é referente à distância entre o centro do bloco inicial até o centro do bloco final. No caso em que se deseja ligar blocos que estão em colunas espaçadas, o valor do parâmetro “b” será a distância entre o centro do primeiro bloco e a altura do bordo superior do bloco final mais o parâmetro “o”.

Para os demais casos, os parâmetros podem assumir valores negativos, como é o caso das realimentações, ou até valores nulos, no caso das conexões que não precisam de cinco movimentos (somente um ou três). Os valores dessas variáveis são definidos em linha de código, relacionando colunas e linhas das posições dos blocos, como pode ser visto na tabela abaixo:

```
‘Cf = Coluna do bloco final
‘Ci = Coluna do bloco inicial
‘Lf = Linha do bloco final
‘Li = Linha do bloco inicial
'Valores padrão(maioria dos casos)
a = 2 * (Cf - Ci - 1)
b = -2 * (Li - Lf - 0.5)
‘Teste de início
If (Lf - Li = 1) Then
    b = -2 * (Li - Lf + 0.5)
End If
If (Lf - Li = 2) Then
    b = -2 * (Li - Lf + 0.5)
End If
 ‘Teste de fim
If (Cf - Ci = 1) Then
    b = -2 * (Li - Lf)
    If (Lf - Li = -2) Then
        b = 2
    End If
End If
If (Lf - Li = -2) Then
    b = -1
End If
```

Tabela 7 - Código para definição dos parâmetros de desenho. [25]

Por fim, a rotina de desenho dos cinco movimentos trabalha sempre atuando no pior dos casos cinco vezes. Quando o valor de algum parâmetro é nulo, laços condicionais fazem com que o passo não seja dado, assim não se perde tempo por processamento desnecessário e não faz com que as funções trabalhem de forma obsoleta. A conexão entre os bloco se dá da seguinte forma:

Legenda:
Bi_x = Posição X do bloco Inicial
Bi_y = Posição Y do bloco Inicial
Bf_x = Posição X do bloco Final
Bf_y = Posição Y do bloco Final
O = Parâmetro “o”
a = Parâmetro “a”
b = Parâmetro “b”

	Função	**Pos.Inicial X**	**Pos. Inicial Y**	**Pos. Final X**	**Pos.Final Y**
1°	Linha	Bi_x	Bi_y	Bi_x+O	Bi_y
2°		Bi_x+O	Bi_y	Bi_x+O	Bi_y+b*O
3°		Bi_x+O	Bi_y+b*O	Bi_x+a*O	Bi_y+b*O
4°		Bi_x+a*O	Bi_y+b*O	Bf_x-O	Bf_y
5°		Bf_x-O	Bf_y	Bf_x	Bf_y

Tabela 8 - Parâmetros dos cinco movimentos. [25]

- 1°Movimento - Linha(Bi_x, Bi_y, Bi_x+O, Bi_y)
- 2°Movimento - Linha(Bi_x+O, Bi_y, Bi_x+O, Bi_y+b*O)
- 3°Movimento - Linha(Bi_x+O, Bi_y+b*O, Bi_x+a*O, Bi_y+b*O)
- 4°Movimento - Linha(Bi_x+a*O, Bi_y+b*O, Bf_x-O, Bf_y)
- 5°Movimento - Linha(Bf_x-O, Bf_y, Bf_x, Bf_y)

5.1.4: Desenho das entradas e saídas nos blocos

Para desenhar as entradas e saídas escolhidas, o usuário primeiramente precisa configurar cada bloco na janela de configuração dos blocos, que será explicada posteriormente. Porém, para entendimento deste tópico, essa janela, basicamente, oferece ao usuário a possibilidade de escolher quais entradas e saídas ele deseja utilizar do controlador. Uma vez escolhidas, essas entradas e saídas formam vetores de dados que são escritos em uma planilha do Excel. Tais vetores serão utilizados tanto para enviar os dados de calibração à ECU, como para desenhar as configurações de entradas e saídas nos blocos.

Exemplificando, imagina-se que o usuário decidiu configurar o bloco 1, e então a Janela de Configuração dos blocos se abre. O mesmo decidiu que o bloco 1 terá 3 entradas externas (que serão definidas por ele) e uma saída. Assim, o vetor de calibração dos parâmetros de entrada e saída é escrito com as devidas escolhas.

Novamente na janela gráfica principal, quando se deseja visualizar o número de entradas e saídas que foram selecionadas em cada bloco, deve-se selecionar a opção de visualizar as entradas e saídas ("Show I/Os", pode ser visto na Figura 35), e então elas são desenhadas na janela principal. O modo como isso é feito leva em consideração uma varredura nos vetores de calibração, de modo a contabilizar o número de entradas e saídas que cada bloco possui, podendo assim usar a mesma função “linha”, utilizada para fazer a conexão entre os blocos, para realizar o desenho gráfico.

A rotina de tal função para desenhar as entradas e saídas usufrui dos valores provenientes da varredura dos vetores para processar um laço condicional que gera as linhas de entrada nos blocos. As linhas são posicionadas de maneira uniforme seguindo uma divisão da altura do bloco pelo número de entradas. Podemos observar na Figura 42 que o bloco 1 possui 3 entradas, sendo 2 delas entradas externas, e uma entrada interna (saída do bloco 2), já no bloco 2 e 3 cada um possui somente uma entrada e uma saída.

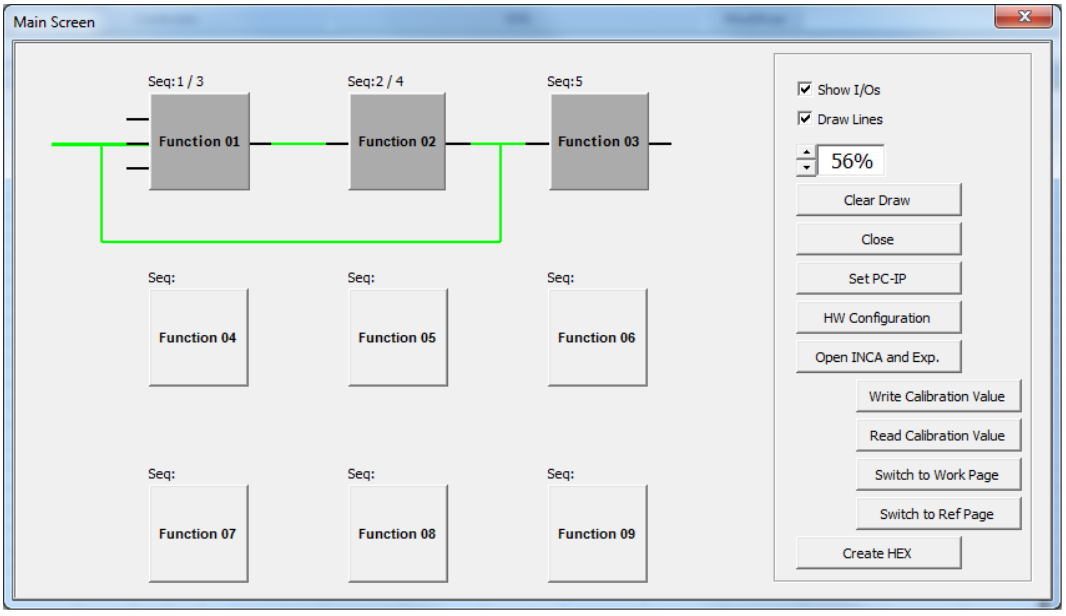


Figura 42 - Exemplo de configuração. [25]

5.1.5: Sequencia de desenho

Como dito, sempre que se alterna de janela onde se tem tratamentos gráficos, o foco da classe de desenho precisa ser alterado. Uma vez alterado o objeto de desenho, todo trabalho gráfico realizado anteriormente é sobreposto pelo novo desenho. Fazendo com que seja sempre necessário atualizar o desenho por completo, redesenhando as linhas e conexões que já existiam.

Essa tarefa exige um tratamento de sequência, onde a parte gráfica se baseará para redesenhar conexões já feitas. Essa sequência é escrita pela Janela de Configuração dos Blocos e toda vez que um bloco é selecionado, a sequência é incrementada com o tratador do bloco. Dessa forma, a rotina de desenho da janela principal lê os blocos que fazem parte da sequência e desenha todos sempre que necessário.

A leitura da sequência, em linha de código, procura o primeiro bloco da sequência e o último, assim o desenho entre eles se faz gradativamente sempre usando a função dos cinco movimentos para cada dupla de blocos. Exemplificando para o caso da Figura 35, a sequência escrita seria:

B1 – B2 – B1 – B2 – B3

A rotina então iria identificar o primeiro bloco e o último, para então realizar as seguintes conexões entre blocos:

1) B1 – B2
2) B2 – B1
3) B1 – B2
4) B2 – B3

Além de ser útil para auxiliar na parte gráfica, a sequência também é utilizada para informar ao usuário qual é a ordem que os blocos irão seguir. Esse informativo fica na parte superior de cada bloco ao lado da palavra “Seq:”.

5.1.6: Outras funcionalidades

Além da parte gráfica, a janela principal oferece um quadro de botões e configurações. Neste quadro, visto na Figura 43**Erro! Fonte de referência não encontrada.**, nas duas primeiras caixas de marcação, estão as opções de visualizar, ou não, tanto as entradas e saídas selecionadas, como as próprias linhas que ilustram a sequência da malha de controle. Logo abaixo o usuário pode diminuir ou aumentar o tamanho dos blocos.

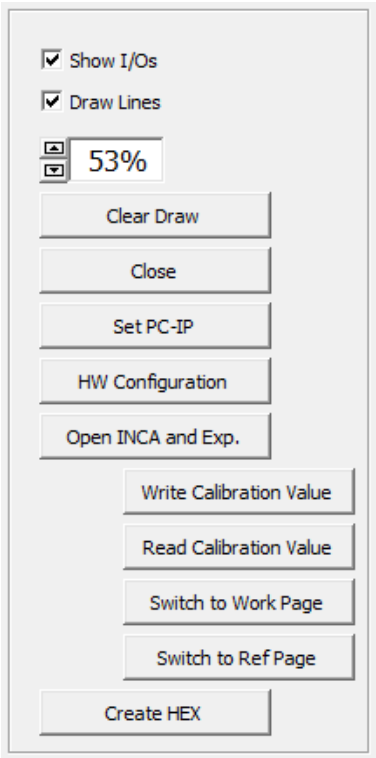


Figura 43 - Quadro dos demais botões do formulário principal. [25]

Os demais botões realizam nessa ordem funções que limpam a tela de desenho, fecham a aplicação, configuram os protocolos de comunicação, abrem a janela de Configuração de Hardware, abrem o INCA, realizam algumas tarefas dentro do INCA e abrem a janela de Configuração do Flash. Este quadro de botões deverá no futuro ser retirado da solução, de modo que o usuário precise se preocupar somente com a malha de controle e, uma vez definida, todas as tarefas do plano de fundo devem acontecer sem a intervenção do mesmo. A razão disso é que o usuário não precise ser habituado com o INCA, que em planos futuros pode ser retirado da solução.

Dentre todos esses botões o que merece ser detalhado é o botão "Set PC-IP", que tem a função de configurar os protocolos de comunicação do computador de acordo com o utilizado pelo hardware do INCA.

5.1.6.1: Configuração dos protocolos de comunicação

Para que seja possível a comunicação entre o PC e a unidade de controle eletrônico, algumas configurações de protocolos precisam ser realizadas. Lembrando que a intenção do projeto é fazer com que o usuário não precise ter conhecimento no software de calibração INCA, tais configurações precisam ser feitas automaticamente. Portanto, uma função foi criada para chamar um arquivo do tipo “*Batch*” que irá automaticamente modificar o IP do computador.

O arquivo Batch ou arquivo de lote (também conhecidos por .bat) é um arquivo de computador utilizado para automatizar tarefas. Esse arquivo deve ser executado sobre direito de administrador da máquina, portanto a chamada da função dentro do Excel precisa passar primeiramente pelo Processador de Comandos do Windows (conhecido como CMD). O CMD irá requerer a senha de administrador, para então chamar o arquivo batch e configurar o IP de acordo com o roteiro do arquivo. [20]

Por fim a configuração do IP precisa ser também modificada no INCA, uma vez que é através dele que o Excel se comunicará com o hardware. Esta tarefa é realizada nas configurações de hardware automaticamente, utilizando as APIs do INCA.

5.2: Formulário de Configuração dos Blocos

Ao selecionar um dos blocos na janela principal, o usuário será encaminhado a uma nova tela, que proporcionará, ao mesmo, várias opções de configuração, no que se trata das entradas e saídas desejadas, bem como a função que será exercida pelo bloco.

Dentre as principais tarefas dessa nova janela estão: a demonstração gráfica das entradas escolhidas, a definição da função desejada e a escrita dos vetores de

calibração. São os vetores de calibração que carregarão os dados de informação a serem inseridos na ECU e gerarão a lógica de controle no hardware.

5.2.1: Desenho das entradas e saídas

Da mesma forma como foi explicado para o caso da janela principal, as entradas e saídas selecionadas na janela de Configuração dos blocos são desenhadas no formulário atual de modo a indicar graficamente as escolhas feitas pelo usuário.

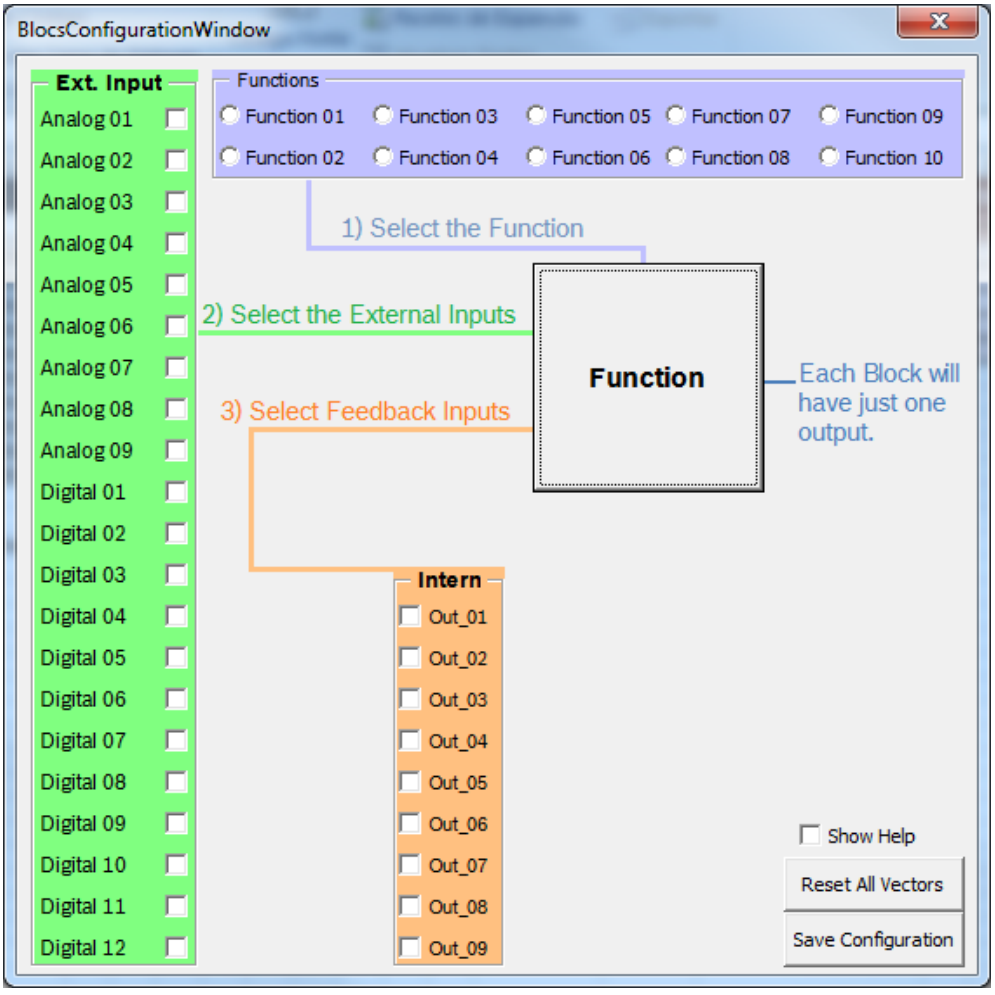


Figura 44 - Janela de Configuração dos Blocos. [25]

Primeiramente, ao ser iniciada, a tela mostra opções de ajuda para melhor entendimento do usuário. Tal ajuda pode ser mostrada a qualquer momento ao selecionar o campo de um *checkbox*. Como pode ser observado na Figura 44, o quadro em verde representa as entradas externas, provenientes do hardware da

ECU, sendo, dessas, nove entradas analógicas e doze entradas digitais, podendo, no caso de necessidade, via software transformar as entradas analógicas em entradas digitais. O quadro laranja apresenta as saídas dos outros nove blocos, que podem ser usadas como entradas de qualquer bloco, existindo a possibilidade de representar, então, realimentações ou a união de estruturas de diferentes funções em série. Já o quadro em roxo mostras as possíveis funções que serão desenvolvidas em C e serão armazenadas no hardware, podendo assim, o usuário, selecionar uma, e somente uma, função para cada bloco.

Quando se configura algum bloco da última coluna, a janela ainda oferece a opção de escolher qual saída de hardware será utilizada.

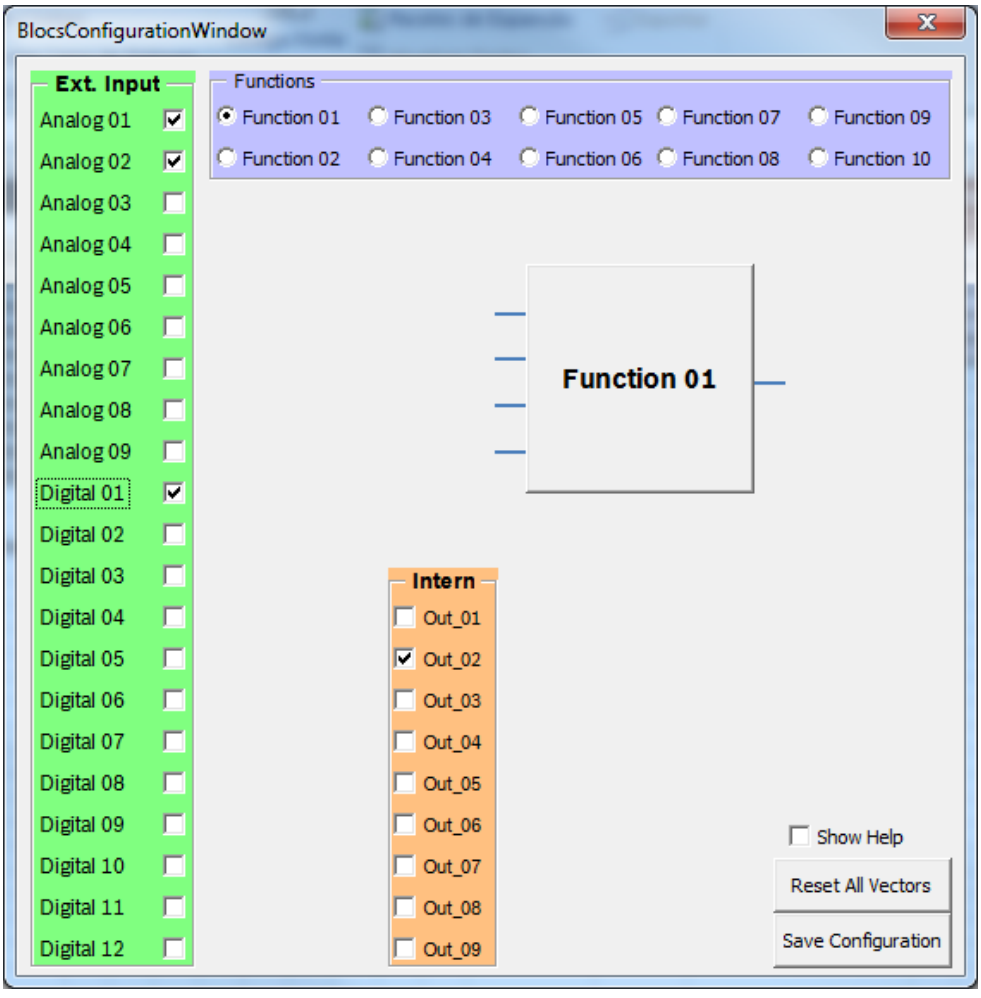


Figura 45 - Exemplo de configuração de um bloco. [25]

A figura acima exemplifica o caso de um bloco que irá exercer a função "*Function 01*", e possui como entradas externas as portas analógicas "*Analog 01*" e

"*Analog 02*" e uma porta digital "*Digital 01*", bem como uma entrada interna proveniente da saída do bloco 2, chamada de "*Out_02*".

5.2.2: Parâmetros de calibração

Ao ser configurado, um bloco deve gravar nos vetores de calibração os respectivos valores das entradas e saídas utilizadas, as funções selecionadas e, caso existam, os parâmetros da função utilizada. Tais vetores serão utilizados para serem gravados no software da ECU, de modo que ao recebê-los, o hardware identifique-os por meio de uma multiplicação de matrizes. Para tanto uma planilha do Excel é utilizada para armazenar os valores das configurações, que posteriormente serão enviadas como parâmetros de calibração.

Para gravação dos vetores na planilha, os dados seguem uma ordem fixa de gravação, onde cada entrada tem um valor de casa associado a ela na planilha. Assim foram criados três tipos de vetores:

Pe – Vetor que identifica as entradas, sejam externas ou internas.

Pf – Vetor que identifica a função associada ao bloco.

Pc – Vetor que carrega os valores dos parâmetros, caso existam, da função selecionada.

Cada tipo de vetor, possui nove vetores de calibração, que representam cada um dos nove blocos que fazem parte da estrutura do sistema, assim cada bloco, quando configurado vai gravar na planilha o seu respectivo vetor de entradas (Pe), de funções (Pf) e de parâmetros da função (Pc). Dessa forma a nomenclatura dos vetores segue a lógica:

Pe → Pei = [Pe1,Pe2,Pe3...,Pe9]

Pf → Pfi = [Pf1,Pf2,Pf3...,Pf9.]

Pc → Pci = [Pc1,Pc2,Pc3...,Pc9]

Uma vez que os parâmetros estão configurados, a ideia para a leitura e tradução dos dados por parte da ECU será feita, baseando-se nos parâmetros calibrados. A estrutura de nove blocos é exatamente o que será levado como

estrutura para as funções e rotinas de cálculos dentro da unidade de controle eletrônico. Assim a ideia é criar uma função genérica que defina a rotina de cálculo baseada nos parâmetros de calibração. Parâmetros que irão informar à rotina quais entradas externas e saídas internas que estão sendo utilizadas, a função desejada, os parâmetros da função, e a noção temporal que será calculada (relativo à sequência). Na Figura 46 observa-se uma ideia do que acontece dentro do software da ECU, em termos de multiplicação de matrizes, onde a entrada “e(k-t)” define o vetor de entrada com todas as entradas e saídas possíveis, que quando multiplicadas pelo vetor "Pe" traduzem exatamente as variáveis que serão utilizadas (após limpeza de termos nulos). O mesmo acontece com os parâmetros Pf e Pc que definem para a função como ela deve agir e quais os seus parâmetros, caso existam. Algumas operações para eliminar entradas nulas são realizadas para não existirem multiplicações por zeros.

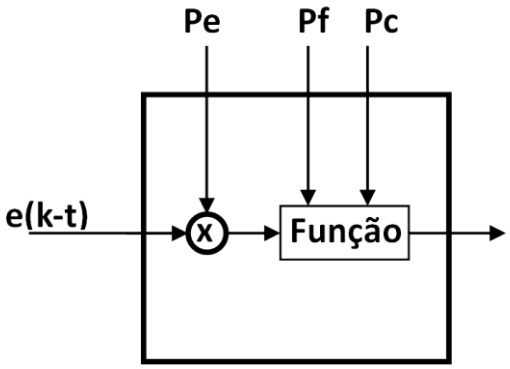


Figura 46 - Tradução dos parâmetros de calibração. [25]

Tomando como base o exemplo da Figura 45, onde o bloco que está sendo configurado é o bloco 1, e o mesmo possui 4 entradas (“*Analog 01”, “Analog 02”, “Digital 01”, “Out_01*”) e exerce a função “*Function 01*”, pode-se observar que os vetores de calibração resultam nos seguintes arranjos:

	Pe1		Pf1		Pc1
Analog 01	**1**	*Function 01*	**1**	*Parameter 01*	**0**
Analog 02	**1**	*Function 02*	**0**	*Parameter 02*	**0**
Analog 03	**0**	*Function 03*	**0**	*Parameter 03*	**0**
Analog 04	**0**	*Function 04*	**0**	*Parameter 04*	**0**
Analog 05	**0**	*Function 05*	**0**	*Parameter 05*	**0**
Analog 06	**0**	*Function 06*	**0**		
Analog 07	**0**	*Function 07*	**0**		
Analog 08	**0**	*Function 08*	**0**		
Analog 09	**0**	*Function 09*	**0**		
Digital 01	**1**	*Function 10*	**0**		
Digital 02	**0**				
Digital 03	**0**				
Digital 04	**0**				
Digital 05	**0**				
Digital 06	**0**				
Digital 07	**0**				
Digital 08	**0**				
Digital 09	**0**				
Digital 10	**0**				
Digital 11	**0**				
Digital 12	**0**				
Out_01	**0**				
Out_02	**1**				
Out_03	**0**				
Out_04	**0**				
Out_05	**0**				
Out_06	**0**				
Out_07	**0**				
Out_08	**0**				
Out_09	**0**				

Tabela 9 - Parâmetros de calibração do exemplo [25]

5.3: Formulário de envio de parâmetros para o INCA (Flash)

Como dito anteriormente, as tarefas principais do software de configuração inicial, o PICS, são de realizar a configuração da estrutura de controle (gráfico) e então efetuar a programação Flash da ECU com os valores dos vetores de calibração. Tendo uma estrutura de controle definida, com todos os blocos configurados de maneira satisfatória, a segunda tarefa utiliza as APIs do INCA de modo a calibrar a ECU, e enviar os parâmetros que foram configurados. Fazendo assim com que a ECU trabalhe de acordo com o que o usuário deseja.

Quando se deseja gravar a memória da ECU, o formulário de Configuração de Flash irá aparecer, como mostrado na Figura 47. Nesta janela é oferecido um campo para que o usuário possa fornecer o diretório o qual será utilizado para armazenar o novo arquivo de calibração que posteriormente será gravado na ECU.

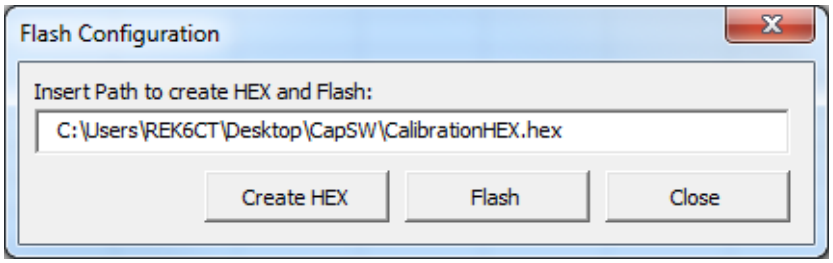


Figura 47- Janela de criação e gravação do arquivo de calibração. [25]

Para realizar essa tarefa, foram utilizadas algumas APIs do INCA que, como citadas anteriormente, realizam algumas funções automaticamente. Dentre essas funções as mais importantes estão:

Open INCA

Abre o software com direitos de administrador. E carrega o banco de dados atual.

Open Experiment

Abre um experimento pré-setado no banco de dados atual do programa. A ideia para o cliente final será de entregar o banco de dados pronto, de modo que dentro deste estejam todos os itens necessários para que o PICS possa configurar, calibrar e gravar a ECU, sem que o usuário deva intervir manualmente no software INCA.

Write Calibration Value

Essa função escreve valores nas variáveis de calibração do software da ECU, porém a função fornecida pelo fabricante teve de ser modificada, pois a mesma faz a escrita de valores escalares e não de vetores ou *arrays*. Assim a nova função realiza em seus passos a declaração de algumas variáveis de auxílio, a atribuição do elemento de calibração a partir do nome da variável a ser calibrada, a escrita dos valores configurados e escritos na planilha para o vetor de calibração desejado, bem como algumas rotinas de tratamento de erros.

Create Hex File

Cria o arquivo com extensão .hex que será gravado na ECU, esse arquivo é escrito baseando-se nas configurações desejadas e nos vetores de calibração definidos. Ao fornecer o diretório através da caixa de texto o usuário deve clicar no botão “Create HEX”, chamando a função:

CreateHexFileForWorkPageAndCode (PathName)

Flash HEX

Essa API é a que mais envolve cuidados, pois ao gravar na memória flash de uma ECU, alguns acontecimentos podem fazer com que a ECU seja perdida, ou que ocorra erros na gravação, ou que a função não atenda os requisitos de acesso. A função requisita dois parâmetros do tipo String: um que é o diretório do arquivo ".hex" criado e outro é o “Control Flow” que nada mais é que o modo como a ECU será gravada.

5.3.1: Control Flow

O Control Flow é geralmente criado pelo PROF toolbox, aplicativo responsável pela segurança de acesso aos dispositivos das unidades de controle eletrônico e depende do dispositivo a ser gravado, no caso deste projeto, uma ECU Value Motronic. Uma dificuldade que foi encontrada nesta etapa foi obter o Control Flow para essa ECU, que nunca havia sido usada pela divisão da Bosch em Curitiba e que normalmente é disponível (quando é disponível) para colaboradores Bosch em outro formato de arquivo compatível com o PROF toolbox.

Atualmente, o projeto já tem posse do Control Flow desejado, e por motivos de segurança e confidencialidade não pode ser informado.

Os vetores de calibração, já explicados anteriormente, seguem um padrão que pode ser visto na Tabela 10 - Vetor de Calibração a ser escrito no INCA. [25]

Tal vetor não será enviado ao INCA exatamente no mesmo formato como é escrito no ambiente do Excel. Com o objetivo de facilitar a leitura e escrita dos vetores, uma unificação dos vetores de calibração foi feita, com intuito de facilitar o desenvolvimento do software da ECU, e diminuiria o tempo de processamento da

informação no hardware. Assim, a mensagem que será transmitida em formato de *array*, segue o seguinte padrão:

Posição	**Range**
Analog 01	***0 / 1***
…	
Analog 09	
Digital 01	
…	
Digital 12	
Out_01	
…	
Out_09	
Function	***[1...10]***
Flag	***0 / 1***

Tabela 10 - Vetor de Calibração a ser escrito no INCA. [25]

De acordo com a tabela acima, podemos ver que os valores podem ser zero ou um para todas as opções de entrada, de 1 a 10 para cada função, que terá seu valor indicativo respectivo e a última casa irá indicar se a função selecionada irá precisar de parâmetros extras. Um exemplo de uma função que precisaria de parâmetros é a de um controlador PID em que além da função propriamente dita, o processador precisa de algumas variáveis como tempo de subida, tempo de resposta, lugar das raízes, ou outros que poderão servir como elementos de cálculo. Caso sejam necessários os parâmetros para alguma função, um novo *array* (Pc) é enviado pela ECU.

5.4: Formulário de configuração de hardware

Após o arquivo de calibração ser criado e gravado na ECU, utilizando o formulário apresentado no tópico anterior (Formulário de envio de parâmetros para o INCA (Flash)), alguns ajustes devem ser feitos no INCA. Como o usuário não terá acesso ao mesmo, o Formulário de Configuração de Hardware precisou ser desenvolvido visando atender esse requisito não planejado anteriormente. Para apresentação deste formulário deve-se ter em mente alguns conhecimentos do INCA, apresentados a seguir. [23]

- **Páginas de memória (Memory Pages) –** A memória de uma unidade de controle eletrônico contendo uma calibração de hardware normalmente consiste de duas páginas (“Reference” e “Working Page”), cada uma contendo o programa inteiro da unidade de controle, incluindo os dados. O usuário tem a opção de trocar entre essas duas páginas. Normalmente, a “*Reference Page*” é somente de leitura e contém a versão de referência já salva. A “*Working Page*” pode ser editada ou modificada pelo usuário, por exemplo, um calibrador.

- **Página de referência (*Reference Page*) -** página de memória da ECU, contendo uma versão de dados específica. Os dados apresentados são usados como uma referência e não podem ser modificados (*read-only*).

- **Página de Trabalho (*Working Page*) -** página de memória na ECU, contendo uma versão de dados específica. Os dados incluídos podem ser modificados e são, portanto, chamados dados de trabalho ou calibração.

- **Projeto –** Um item do banco de dados que geralmente contém o arquivo de descrição do projeto (*A2L), e um arquivo de calibração (por exemplo, * hex). O arquivo do calibração pode conter o código e os dados. O projeto é criado através da leitura do arquivo de descrição do projeto (arquivo A2L) e guardado como um item de banco de dados no banco de dados. Ele contém a descrição de todas as informações relevantes para a gestão da calibração (endereços, métodos de depósito, etc.) de um programa da unidade de controle.

- **Arquivo A2L -** formato padrão de troca em formato ASCII usado para projetos .

- **Arquivo HEX -** formato de troca de uma versão do programa, o arquivo hex (*hex ; * s19 ; Intel hex ou formato Motorola) contém o programa de unidade de controle que consiste no código e dados. O conteúdo deste arquivo pode ser carregado diretamente para a unidade de controle e executado pelo respectivo processador.

- **Espaço de trabalho (*Workspace*) -** Um espaço de trabalho é um item do banco de dados no qual todas as informações necessárias para uma medição específica ou tarefa de calibragem são armazenados e consolidados.

- **Parâmetro –** É o objeto de calibração. Parâmetros tem um valor que pode ser calibrado e, dentre as opções, existem vários tipos de variáveis, como escalar, curva, mapa, matriz, cadeia ascii, e etc.

- **Experimento -** Dataset que contém as configurações para uma experiência específica de calibração (canais, tempo de amostragem, representação, etc.)

Uma configuração padrão do INCA envolve os conceitos acima e tem uma hierarquia como pode ser vista na Figura 48. A pasta do projeto ("*Project_Folder*") contém três itens dos explicados conceitualmente anteriormente, são eles: o experimento ("*Experiment*"), o projeto ("*Project*"), e o espaço de trabalho ("*Workspace*").

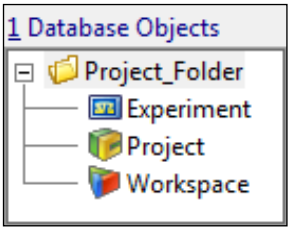


Figura 48 - Hierarquia da Configuração do INCA. [25]

Para um projeto ser inserido no banco de dados, ele precisa ser vinculado a um arquivo com extensão ".a2l", e então a um arquivo de calibração do tipo ".hex". Quando se cria o item, já é necessário fornecer o diretório do arquivo de programa (".a2l"), e então se vincula o arquivo de calibração(".hex"), como pode ser visto na Figura 49.

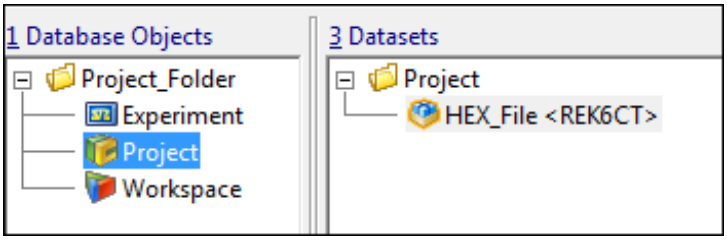


Figura 49 - Projeto e o Arquivo de calibração (".hex") vinculado. [25]

A configuração do espaço de trabalho envolve algumas ações como vincular o arquivo de calibração, e definir qual vai ser o hardware utilizado, bem como suas opções de protocolos de comunicação, e o dispositivo. Na Figura 50, percebe-se que o espaço de trabalho está vinculado com o arquivo de calibração (“HEX_File”) e se comunicará com o Hardware “ES592” através do subsistema de protocolo “CAN”, pelo dispositivo “CCP”.

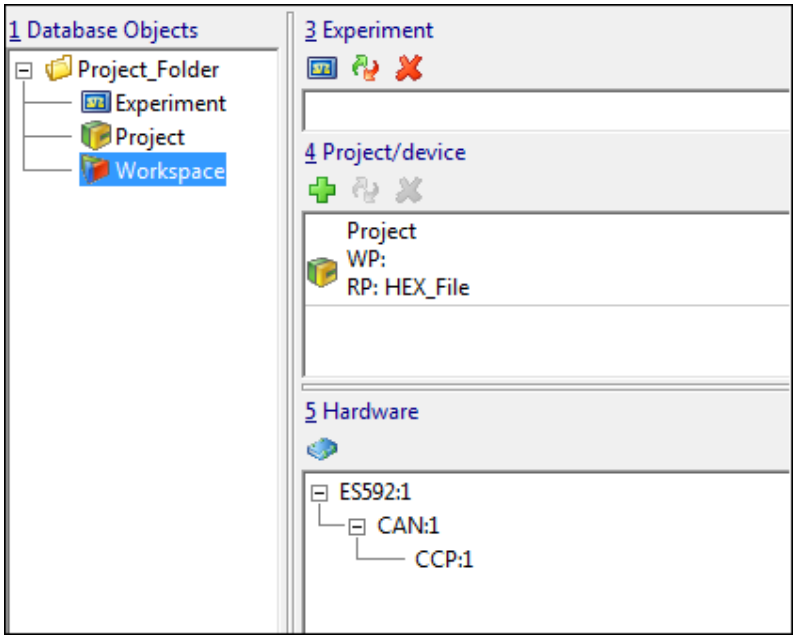


Figura 50 - Workspace com o projeto e o hardware associados. [25]

Resumidamente, toda configuração possui:

- Um **banco de dados**, contendo toda a configuração;
- Um **projeto** contendo os arquivos com extensão “.hex” e “.a2l”;
- Um **espaço de trabalho** ("Workspace”), onde são vinculados o projeto e o hardware que será utilizado para fazer a ponte entre PC e ECU, no caso o ES592.1 do fabricante ETAS;
- Um **experimento** (“Experiment”), onde são calibradas as variáveis (Parâmetros de calibração).

A intenção como produto final do projeto era entregar o PICS, o INCA e a configuração padrão a ser instalada, de modo que o mesmo pudesse instalar e rodar o INCA sem precisar tomar decisões em cima dessa configuração. Assim o aplicador

faria a calibração em cima do software base, com o INCA rodando em segundo plano e então gravaria na ECU.

Porém, o gerenciamento das páginas de memória da unidade de controle eletrônico faz com que a configuração padrão deva ser atualizada toda vez que uma das páginas de memória é modificada, criando assim um problema, uma vez que se deseja que o usuário não tenha acesso a modificações na configuração padrão do INCA.

Em outras palavras, toda vez que uma ECU é calibrada a *Working Page*, no INCA, é modificada. Como visto acima o INCA oferece duas páginas de memória, a *Reference Page* e a *Working Page*. A primeira contém os dados do software base e não pode ser modificada, sendo útil então para atuar como um arquivo de segurança. Já a *Working Page* é uma cópia da *Reference Page* que pode ser modificada para calibração. Uma vez que a calibração é satisfatória, essa *Working Page* modificada pode ser gravada na ECU se tornando a *Reference Page* da ECU.

Para a ECU ser gravada com um arquivo de calibração, ou seja, utilizando uma *Working Page* modificada, a ECU precisa ter uma *Reference Page* igual a *Reference Page* do INCA. Ou seja, caso o usuário tenha feito uma calibração e tenha gravado a ECU com a "*Working_Page01*", na próxima vez que essa ECU for calibrada, o INCA precisa ter alterado a *Reference Page* antiga, modificando o projeto e colocando no lugar a ”*Working_Page01*". Caso isso não seja feito, um erro ocorre, pois os parâmetros de calibração são modificados baseando-se nos parâmetros atuais gravados na memória da ECU. Assim, a referência de valores estará incorreta para calibração.

Partindo dessa problemática, algumas novas funções tiveram que ser criadas possibilitando que a configuração do INCA possa se atualizar sempre que se fizer necessário. Para tanto, foi criado um novo formulário que permite ao usuário escolher, em caso de algum erro, a configuração ideal.

O formulário de configuração do INCA tem interface que pode ser vista na

Figura 51. Essa janela escreve e lê os diretórios e nomes de variáveis utilizando uma planilha do próprio *Excel* para servir de banco de dados. Essa planilha tem o seguinte formato:

	CONFIGURAÇÃO
Database Folder	Nome da pasta
Asap2 file	Diretório do arquivo .a2l
HEX file	Diretório do arquivo .hex
Hex File Project Name	Nome do arquivo .hex a ser criado
System Name	Nome do Hardware
SubSytem Name	Nome do Hardware_Protocolo
Device Name	Nome do dispositivo
WorkspaceName	Nome do espaço de trabalho
ExperimentName	Nome do experimento

Tabela 11 - Dados para configuração do INCA. [25]

A primeira decisão a ser tomada quando se deseja acessar a janela de configuração de hardware é definir se o usuário deseja criar a configuração padrão, ou se deseja gerir manualmente uma nova configuração. Caso a intenção seja configurar de forma padrão, o usuário precisa somente aplicar (“Apply”) as modificações e esperar a operação finalizar. Caso deseje realizar uma configuração manual, como pode ser observado na Figura 51, o segundo quadro de opções se faz visível para preenchimento dos campos de texto e de direcionamento de pastas. Após preencher todos os dados, o usuário deve pressionar o botão “Create Database and Load HW Systems”, o qual chama funções de criação de configuração (“SetHWConfiguration”), e de listagem de todos os sistemas possíveis de fazer a ponte entre PC e ECU (“ListAllSystems”).

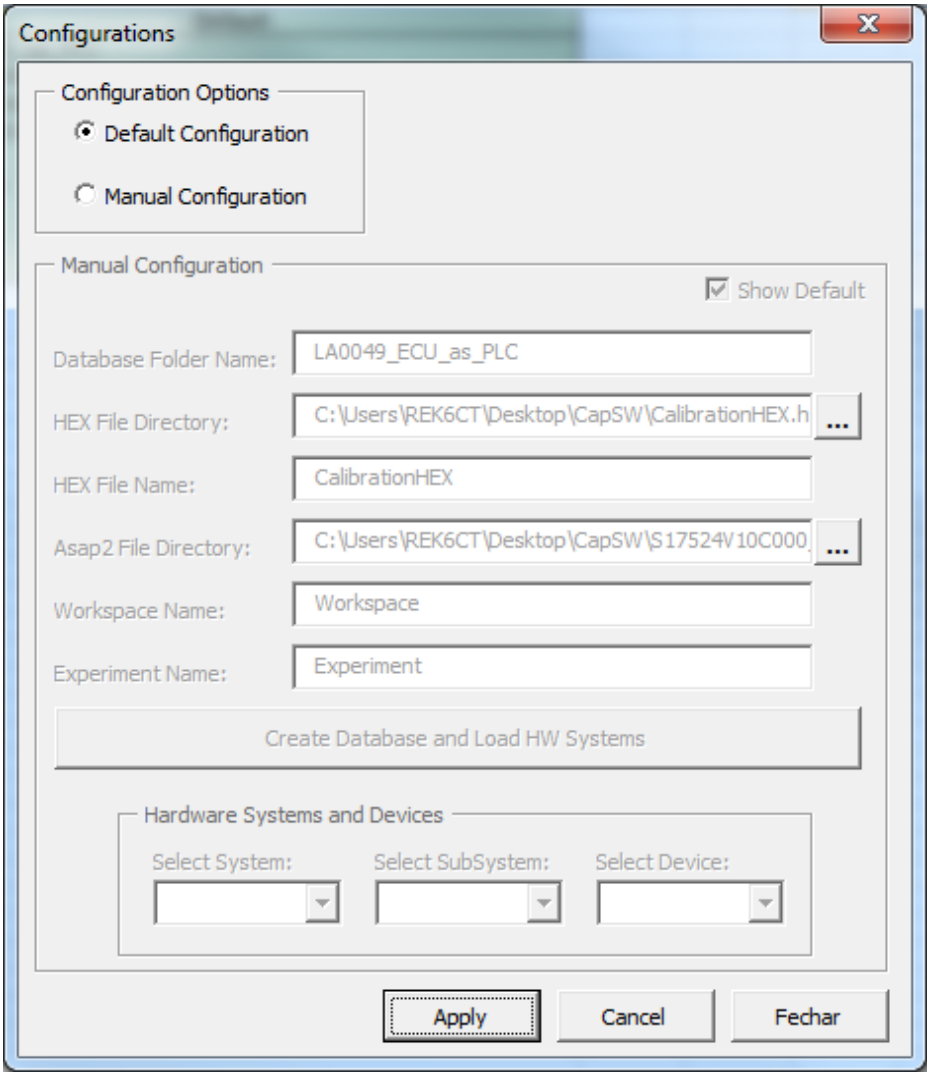


Figura 51 - Formulário para configuração do INCA. [25]

Uma vez que o banco de dados foi criado e os itens foram inseridos, resta configurar o hardware do espaço de trabalho. As opções são colocadas nas caixas de listagem dentro do terceiro quadro, e dependem do usuário para escolher as configurações e então aplicar e finalizar a operação.

Esse novo formulário foi criado para responder a problemática explicada anteriormente, onde o gerenciamento das páginas de memória se faz fundamental. Dessa maneira todas as funções criadas para realizar a configuração são genéricas, e estão disponíveis para serem chamadas após uma gravação da ECU, sem que o usuário precise se preocupar em realizar esses passos, caso não deseje.

5.5: Códigos, funções e esquemas

Todo o desenvolvimento realizado em cima do software resultou em milhares de linhas de código, que não serão apresentadas por motivos de confidencialidade e de proteção de dados da empresa. Porém algumas das relações entre os módulos e os formulários, que foram citadas neste texto podem ser compreendidas de acordo com a ilustração abaixo. Onde as setas pretas indicam quais formulários são alimentados por quais módulos, as linhas tracejadas vermelhas mostram a relação de acesso entre formulários, e os pontos roxos definem quem tem acesso aos dados da planilha.

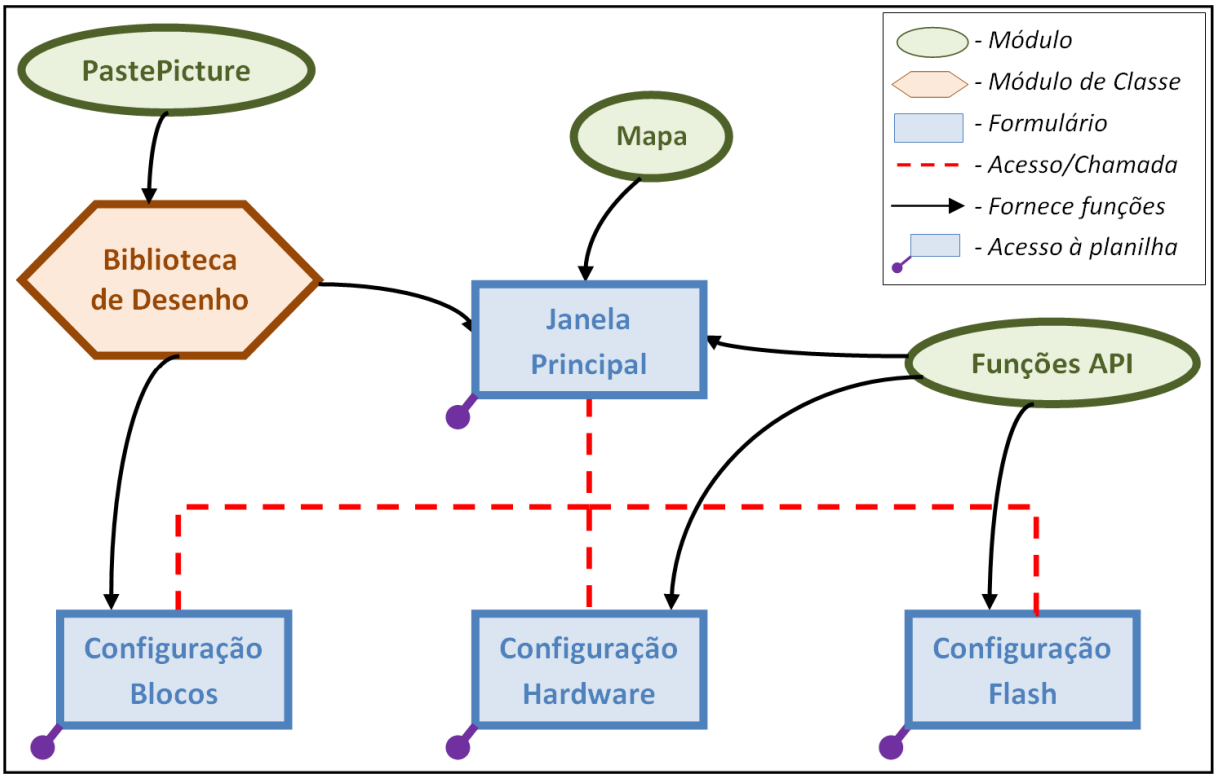


Figura 52 - Relações entre módulos e formulários. [25]

As principais funções utilizadas no Software de Configuração Inicial de CLPs podem ser conferidas na lista abaixo, dividida por módulos e classes, e colorida de acordo com o diagrama acima. Uma breve descrição de cada função também é apresentada na relação abaixo.

	Função	Descrição
Funções API	ConnectToInca	*Conecta ao INCA*
	OpenINCA	*Abre o INCA*
	GetDatabase	*Busca o banco de dados usado atualmente*
	IsDatabaseOpen	*Verifica se banco de dados está aberto*
	GetDevice	*Busca o dispositivo e verifica se está conectado*
	ResetDevice	*Reinicia o dispositivo em caso de erro*
	OpenExperiment	*Abre um ambiente de experimentos*
	OpenCalibrationValue	*Abre uma variável de calibração*
	WriteCalibrationValue	*Escreve o valor de uma variável de calibração (escalar)*
	CloseExperimentView	*Fecha o ambiente de experimentos*
	ResetParameters	*Reinicia os valores dos parâmetros*
	RefreshVariables	*Atualiza as variáveis*
	WriteCalibrationArray	*Escreve o valor de uma variável de calibração (array)*
	VectorColumn	*Determina a coluna fixa que será gravado o vetor*
	SwitchToReferencePage	*Muda para página de referência no INCA*
	SwitchToWorkingPage	*Muda para página de trabalho no INCA*
	ReadCalibrationValue	*Lê o valor de uma variável de calibração*
	AddIncaFolder	*Adiciona uma pasta ao banco de dados do INCA*
	GetIncaFolder	*Busca uma pasta dentro do banco de dados do INCA*
	SetProjectAndDataset	*Vincula ao projeto um arquivo .a2l e um .hex*
	ReadASAP2FileAndHexFile	*Lê os arquivos .a2l e .hex*
	AddExpEnvironment	*Adiciona um experimento ao projeto*
	AddHWConfiguration	*Adiciona uma configuração de hardware*
	GetSystemDescription	*Retorna a descrição do sistema*
	GetAllSystemDescriptions	*Retorna a descrição de todos os sistemas*
	GetAllDeviceDescriptions	*Retorna a descrição de todos os dispositivos de hardware*
	GetDeviceDescription	*Retorna a descrição de dispositivos de hardware*
	AddDevice	*Adiciona um dispositivo de hardware*
PastePicture	IsClipboardAvailable	*Verifica se o clipboard contém um bitmap/metafile*
	OpenClipboard	*Abre o clipboard para leitura*
	GetClipboardData	*Busca o dado do clipboard*
	CloseClipboard	*Fecha o clipboard*
	OleCreatePictureIndirect	*Converte o handle em uma interface de figura OLE*
	CopyEnhMetaFile	*Copia o metafile para proteção contra atualizações*
	CopyImage	*Copia o bitmap para proteção contra atualizações*
	PastePicture	*Cola a figura que está no clipboard*
	CreatePicture	*Converte uma imagem em um Picture Object*
	fnOLEError	*Retorna mensagens para erros padrão OLE*
Map	MapCalibration	*Determina o formato da estrutura e grava na planilha*

	DrawSequence	*Calcula os parâmetros para a rotina de desenho*
	CleanAllVectorValues	*Limpa o valor de todos os vetores*
	CleanPeVectorValues	*Limpa os valores do vetor de entradas*
	CleanPfVectorValues	*Limpa os valores do vetor de funções*
	CleanPcVectorValues	*Limpa os valores do vetor de parâmetros de calibração*
	CleanPoVectorValues	*Limpa os valores do vetor de saídas*
Biblioteca de Desenho (AJPiUFDraw) *Módulo de Classe*	Box	*Adiciona uma caixa ao desenho*
	Curve	*Adiciona uma curva ao desenho*
	FreeForm	*Adiciona uma forma livre ao desenho*
	Oval	*Adiciona um círculo ao desenho (com raio em x e y)*
	Round	*Adiciona um círculo ao desenho (somente com um raio)*
	line	*Adiciona uma linha ao desenho*
	CanvasGeneric	*Ponteiro para objeto genérico e dimensiona a figura*
	CanvasUserform	*Ponteiro para um Formulário, e dimensiona a figura*
	WipeCanvas	*Limpa o canvas*
	m_ClearHolder	*Remove todos os desenhos da planilha holder*
	m_CreateFrame	*Cria frame no topo esquerdo para referência de posição*
	Paint	*Une todos os desenhos e copia para uma figura*
	Class_Initialize	*Inicializa a classe*
	Class_Terminate	*Finaliza a classe*
Tela Principal *Formulário*	WriteBlocName	*Escreve nome no bloco, e atualiza indicador de sequência*
	Draw5Moves	*Rotina de desenho dos cinco movimentos*
	ChangeColorLine	*Função para modificar a cor das linhas*
	DrawRoutine	*Determina início e fim da rotina de desenho*
	DrawFirstInput	*Desenha entrada nos blocos da primeira coluna*
	DrawLastOutput	*Desenha saída nos blocos da última coluna*
	ShowConfOnBlocks	*Mostra configurações dos blocos já configurados*
	DrawInsOutsOnBlocks	*Desenha entradas e saídas configuradas no bloco*
	ClearSequence	*Limpa sequência de desenho e dos indicadores.*
	ResizeBlocks	*Ajusta tamanho dos blocos*
	ClearDraw	*Limpa o canvas, atualiza blocos, e variável de desenho*
Configuração dos Blocos *Formulário*	CleanChecks	*Seta falso para todos os checkboxes*
	Assign01ValuesToVector	*Escreve vetores de (checkbuttons)*
	Assign01ValuesToVector2	*Escreve vetores (optionbuttons)*
	Assign01ValuesToVector3	*Escreve vetores (optionbuttons de saídas)*
	LoadVectorValues	*Carrega os valores provenientes de prévia configuração*
	SelectFunction	*Seleciona a função desejada*
	CheckActualBlock	*Verifica qual é o bloco selecionado para configuração*

	InicializeVectorValues	*Inicializa os vetores com valores nulos*
Configuração de Hardware *Formulário*	RefreshNamesHWConfig	*Atualiza os dados de configuração de hardware*
	SetHWConfiguration	*Atualiza dados, conecta ao INCA e cria configuração*
	DefaultHWConfig	*Configura a definição padrão*
	AssignAsap2Project	*Vincula ao projeto um arquivo .a2l*
	CreateHWConfiguration	*Cria configuração de Hardware*
	ListAllSystems	*Lista todos os sistemas disponíveis no INCA*
	ListAllSubSystems	*Lista todos os subsistemas disponíveis no INCA*
	ListAllDevices	*Lista todos os dispositivos disponíveis no INCA*
	SelectHWSystem	*Seleciona um sistema*
	SelectHWSubSystem	*Seleciona um subsistema*
	SelectHWDevice	*Seleciona um dispositivo externo*
	SliptNames	*Separa string de acordo com símbolo*
Flash *Formulário*	FlashHexFileWithControlFlow	*Grava .hex na memória flash da ECU com Control Flow*
	CreateHexForWorkPageAndCode	*Gera .hex com o código da página de trabalho*

Tabela 12 - Principais funções utilizadas no PICS. [25]

Capítulo 6: Conclusões e Perspectivas

O presente trabalho de fim de curso, desenvolvido pelo acadêmico nas dependências da Bosch Engineering Group, é constituído do estudo e do desenvolvimento de um CLP a partir de uma ECU para veículos a gasolina.

O principal objetivo do projeto era validar o conceito provando que uma central de controle eletrônico teria capacidade de ser programada de modo a atuar como um controlador lógico programável. Todos os testes realizados com a gravação de parâmetros calibrados na ECU funcionaram de maneira perfeitamente correta, provando que a ECU pode ser configurada como um CLP. Os testes foram realizados com o próprio INCA no ambiente de experimentos e de gerenciamento de memória. O acesso à ECU pelo software Excel utilizando as APIs do INCA resultaram num novo software capaz de ajudar não somente os programadores dos CLPs, mas também pode ser futuramente desenvolvido para ajudar na calibração das ECUs que são utilizadas em veículos.

Todos os prazos predefinidos para cada parte do projeto foram respeitados sem necessidade de adiamentos nas datas. Os custos relacionados ao tempo de pesquisa e desenvolvimento do projeto foram custeados pela própria BEG, até o momento da aprovação do cliente. Este ponto trouxe uma fragilidade nas relações comerciais entre a BEG e o cliente, fazendo com que não fosse possível alocar mais recursos para desenvolvimento do protótipo.

O projeto proporcionou uma experiência de trabalho em equipe, com engenheiros e com projetistas com grande experiência na área de desenvolvimento de produto, o que transformou o trabalho em um grande aprendizado sobre as diversas áreas de atuação da BEG.

As responsabilidades incumbidas neste trabalho também contribuirão para um grande crescimento profissional. Além disso, a convivência diária em uma seção com diversos projetos de diferentes tipos, e a convivência com mentes pensantes das mais diferentes áreas trouxeram um aumento da percepção de novas soluções de projetos, e pensamentos não triviais para solução de problemas.

Após conclusão do desenvolvimento apresentado como proposta, o projeto foi apresentado ao cliente na Colômbia, que ficou bastante satisfeito com o resultado. O projeto então foi aceito e tem seus próximos passos garantidos. Além do cliente a revista corporativa da Bosch, premiada como uma das três melhores revistas corporativas do mundo, e com público de 300.000 pessoas, tem intenção de realizar uma publicação de inovação tecnológica quando existir um protótipo físico.

Para os passos futuros, já em 2014, reuniões com partes envolvidas, trouxeram grande expectativa para o andamento dos mesmos. A empresa ETAS, responsável pelo software INCA, tem intenção de participar do projeto no desenvolvimento de um software específico, mais leve e com licença menos custosa. A divisão da Bosch responsável pelo BIS irá disponibilizar profissionais capacitados no desenvolvimento da ferramenta para auxiliar e participar na ampliação das funcionalidades do BIS. A BEG irá receber estudantes de mestrado e de graduação para trabalhar no projeto como um programa de pesquisa e desenvolvimento tecnológico.

De forma conclusiva o projeto teve seu conceito validado e com isso atraiu o interesse de diversas empresas e pessoas. Nos planos futuros a efetivação do projeto como um produto de série da Bosch irá resultar em um grande e positivo impacto financeiro para o Bosch Engineering Group, e as partes envolvidas, além de completar o portfólio de produtos de CLPs, e oferecer às demais divisões soluções Bosch completas.

Bibliografia:

[1] Robert Bosch. **Apresentação Institucional Robert Bosch.** Curitiba: Robert Bosch, 2013. 43 slides, color.

[2] DIAS, Kadu. **Mundo das Marcas:** Bosch - Invented for Life. 2006. Disponível em: <http://mundodasmarcas.blogspot.com.br/2006/06/bosch-invented-for-life.html>. Acesso em: 20 jan. 2014.

[3] Bosch Engineering Group (Brasil). **Bosch Engineering Latin America.** Curitiba: Beg, 2011. 10 slides, color.

[4] Bosch Security Systems. **Sobre a Bosch Security Systems:** Sistemas de Segurança América Latina. Disponível em: <http://la.boschsecurity.com/pt/latam_product/06_about_us_10/06_01_company_profile_10/bosch-sicherheitssysteme-unternehmensueberblick?p_l_id=4530461>. Acesso em: 05 fev. 2014.

[5] REXROTH, Bosch. **Sobre a Bosch Rexroth:** Números e Fatos. Disponível em: <http://www.boschrexroth.com.br/country_units/south_america/brasil/pt/company/about_us/20_facts_figures/index.jsp>. Acesso em: 05 fev. 2014.

[6] Robert Bosch ST/SCO (Colombia). **Solución para integrar sistemas de Servicios Básicos con BIS:** Señores: Integradores y Distribuidores Seguridad. Bogota: ., 2011.

[7] CÂNDIDO, Ronei Vilas BÔas. **PADRÃO OPC:** Uma Alternativa de Substituição dos Drivers Proprietários para Acessar Dados de PLCs. 2004. 51 f. Monografia (Especialização) - Curso de Engenharia de Ciência da Computação, Universidade Fumec, Belo Horizonte, 2004.

[8] Wikipédia. **Controlador lógico programável.** Disponível em: <http://pt.wikipedia.org/wiki/Controlador_lógico_programável>. Acesso em: 31 jan. 2014.

[9] SILVA, Gladimir Pinto da. **Controlador lógico programável.** Porto Alegre: Cefet Rs, 2004. 31 p.

[10] Bosch Security Systems. **Building Integration System:** Brochure. Germany: Ms-ot-en-01_f01u517821_04, 2008. 12p.

[11] Bosch Security Systems. **Building Integration System:** Feature overview. St-ist/prm1: Bosch, 2008. 30 p.

[12] Bosch Security Systems B.V.. **Access Control and Integrated Systems:** Databook. Germany, 2010. 164 p. Disponível em: <www.boschsecurity.com>. Acesso em: 16 dez. 2013.

[13] BÄRO, Dr. Stephan; LANG, Eberhard; KALLERHOFF, Tobias. **ValueMotronic:** Abstatt: Robert Bosch, 2007. 16 slides, color.

[14] PROTOCOLO DE COMUNICAÇÕES CAN: Características do CAN. In: Faculdade De Engenharia Da Universidade Do Porto (Porto - Portugal). **Projecto de desenvolvimento de software de aplicação para uma rede CAN e sua interligação com uma rede Ethernet.** Porto. 2002. p. 37-61.

[15] ETAS GMBH (Stuttgart). **INCA Software Products.** 2014. Disponível em: <http://www.etas.com/en/products/inca_software_products.php>. Acesso em: 10 jan. 2014.

[16] API. 2014. Disponível em: <http://pt.wikipedia.org/wiki/API>. Acesso em: 14 jan. 2014.

[17] ETAS GMBH (Germany). **INCA Tool API:** COM - API. Stuttgart: Etas, 2009.

[18] Bosch Sistemas De Seguridad (Colombia). **Symposium Value Motronic:** Bogotá: St/mkc, 2012. 12 slides, color.

[19] Bosch Engineering Group (Brasil). **LA0049 ST Value Motronic as PLC:** Curitiba, 2012. 11 slides, color.

[20] BATCH: Arquivo Batch. Disponível em: <http://pt.wikipedia.org/wiki/Batch>. Acesso em: 15 jan. 2014.

[21] BOSCH REXROTH AG (Germany). **Rexroth WinStudio:** Getting Started Guide. Lohr: Bosch Rexroth Ag, 2004.

[22] DRAZEN, Baic (Deutschland). **Anwendungsgebiete und Konfiguration des CAN in der Steuergeräte Generation "EDC/ME(D)17".** 0.3 Abstatt: Bosch Engineering Gmbh, 2009. 78 p.

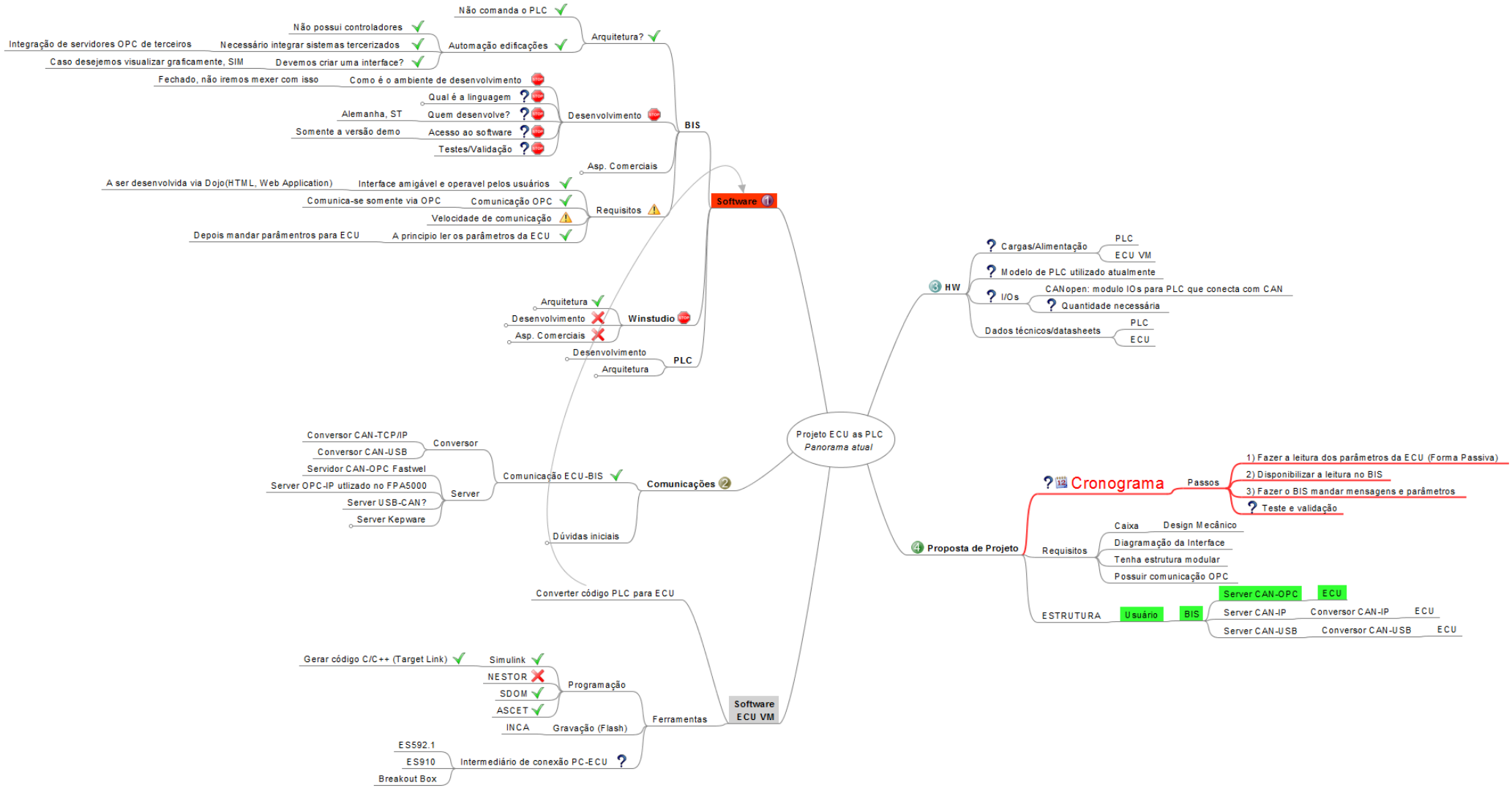
[23] ETAS ENTWICKLUNGS- UND APPLIKATIONSWERKZEUGE FÜR ELEKTRONISCHE SYSTEME GMBH & CO.KG. **ProF GSAUDK.** 1.42 Stuttgart: Etas Gmbh, 1998. 17 p.

[24] Wikipédia. **Controlador lógico programável.** Disponível em: <http://pt.wikipedia.org/wiki/Controlador_lógico_programável>. Acesso em: 31 jan. 2014.

[25] Elaborado pelo autor. Mauricio Reck.

APÊNDICES

APÊNDICE 1: Exemplo de uso do software Freemind.



APÊNDICE 2:Diagrama de Casos de Uso