UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

CURSO DE ENGENHARIA DE COMPUTAÇÃO

CAUÃ BARNEZE ROCHA

SISTEMA DE CONTROLE SUPERVISÓRIO PARA VEÍCULOS DE CONDUÇÃO AUTOMÁTICA

Versão final da monografia apresentada à UTFPR como requisito parcial da disciplina de Trabalho de Conclusão de Curso do Curso de Engenharia de Computação, da Universidade Tecnológica Federal do Paraná.

CURITIBA

2017

CAUÃ BARNEZE ROCHA

SISTEMA DE CONTROLE SUPERVISÓRIO PARA VEÍCULOS DE CONDUÇÃO AUTOMÁTICA

Versão final da monografia apresentada à UTFPR como requisito parcial da disciplina de Trabalho de Conclusão de Curso do Curso de Engenharia de Computação, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Flávio Neves Junior.

CURITIBA

2017

RESUMO

BARNEZE ROCHA, Cauã. **Sistema de Controle Supervisório para Veículos de Condução Automática**. 2017. Versão final da monografia apresentada à UTFPR como requisito parcial da disciplina de Trabalho de Conclusão de Curso do Curso de Engenharia de Computação, da Universidade Tecnológica Federal do Paraná.

Com o rápido avanço da globalização e da tecnologia novos conceitos de mercado e de indústria estão se instaurando na sociedade moderna. Ao longo dos anos o ser humano vem melhorando os padrões de manufatura a fim de tornar o processo cada vez mais automatizado e ter menor interferência humana. Neste meio encontram-se os veículos de condução automática, da sigla em inglês AGV - *automated guided vehicle*, que podem auxiliar a manufatura de diversas formas, sendo a mais comum no transporte e deslocamento de objetos e componentes pela linha de produção. O objetivo deste trabalho é elaborar, através do modelo de desenvolvimento ágil de software, um sistema para controle e gestão de veículos guiados automaticamente. Para a programação deste sistema será utilizada a linguagem *C#* em conjunto com a biblioteca de comunicação *S7.Net Plus*, os controladores lógicos programáveis da marca *Siemens*, e os modelos dos veículos do tipo transportadores de *pallets* orientados por laser. Este sistema facilitará a gestão dos veículos presentes na linha de produção, aumentará a interação homem-máquina relacionada aos veículos e possibilitará uma melhora no desempenho da manufatura.

Palavras-chave: Veículo de condução automática, Controlador lógico programável, Interface homem-máquina, C#, Manufatura.

ABSTRACT

BARNEZE ROCHA, Cauã. **Supervisory Control System for Automatic Guided Vehicles**. 2017. Final version of the monograph presented to UTFPR as a partial requirement of the discipline of Course Conclusion Work of Computer Engineering course at the Federal University of Technology - Paraná.

With the rapid advances of globalization and technology, new concepts to industry and market are emerging in modern society. Over the years, manufacturing industries have testified the growing of new modern and dynamic practices in order to make industrial process more and more automated and run without human. In this context, *AGV* (automated guided vehicle) can assist the manufacture in several ways, especially in the transportation and placement of objects and components in the production line. The goal of this project is to develop, through an agile software model development, a system to control and manage AGVs in a manufacture line. The *C#* language is used in conjunction with the *S7.Net Plus* communication library to implement this system. The *Siemens* programmable logic controllers and the laser-guided pallet conveyor-type vehicle models are also used in the project. The developed system will be easy to manage and coordinate all vehicles in a production line, increasing the human-machine (*AGV*) interaction and will allow an improvement of manufacture performance.

Keywords: Automated guided vehicle, Programmable logic controller, Human-machine interface, C #, Manufacturing.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

AGV	Automated Guided Vehicle
CLP	Controlador Lógico Programável
GPS	Global Positioning System
SCADA	Supervisory Control and Data Acquisition
GC	Garbage Collector
CSV	Comma-separated Values
PLC	Programmable Logic Controller
DB	Data Block
FB	Function Block
IP	Internet Protocol

SUMÁRIO

1 INTRODUÇÃO

1.1 **MOTIVAÇÃO**

Na atualidade econômica cada vez mais competitiva, a automação pode significar a vantagem necessária para fazer a diferença entre sucesso e fracasso de um empreendimento. Tendo em vista que cada aperfeiçoamento na linha de produção de uma indústria que ajude a reduzir o tempo de manufatura, mesmo que para apenas um objeto produzido, é de grande significância quando se observa a fabricação por período de tempo. Pode-se concluir que ao se utilizar veículos guiados automaticamente está se corroborando com a melhoria nas linhas de produção e, portanto, gerando um aumento no lucro e produtividade. Uma pequena redução no tempo de produção ao longo de um grande período de tempo pode acarretar em altos ganhos, e é atrás de recursos como estes que empresas visam melhorar seus negócios.

Outro fator determinante para a realização deste projeto é melhorar a qualidade de vida do trabalhador fabril. A implantação deste sistema evita que o operador realize o deslocamento de peças pesadas ao longo da linha de produção, destinando o tempo para realizar tarefas mais específicas com relação à manufatura do produto, podendo gerar também desta forma um aumento na velocidade de produção e consequentemente aumento nos lucros.

Supervisionar e gerir grandes sistemas de manufaturas pode se tornar um problema quando não se tem o controle correto e efetivo do que acontece no chamado *chão de fábrica*. O sistema realizado neste trabalho visa auxiliar neste gerenciamento, fazendo com que o controle dos veículos de condução automática instalados na linha de produção se torne confiável e robusto, condizente com os requisitos necessários para uma empresa que visa melhorar a gestão da sua produção.

1.2 **RESULTADOS ESPERADOS**

Através do desenvolvimento desse sistema de gerenciamento de *AGVs* espera-se como resultado tecnológico viabilizar uma ferramenta para auxiliar os gestores de manufatura fabril a realizar um melhor controle e administração da sua linha de produção aplicada aos veículos de condução automática presentes na fábrica.

O resultado econômico obtido pelo desenvolvimento deste sistema é a obtenção de maiores lucros, através da melhoria da linha de produção pela diminuição do tempo de manufatura e da melhor utilização do tempo de serviço dos funcionários [1].

O resultado social se tem pela desobrigação das tarefas de manuseio de cargas pesadas e perigosas pelos funcionários das manufaturas, destinando-as aos veículos de condução automática, atingindo um maior bem estar e qualidade de vida aos empregados [1].

Através da redução de emissão de poluentes pela substituição da utilização intensiva de veículos movidos a gás, como empilhadeiras e transportadores, normalmente utilizados nas indústrias, por veículos autoguiados com motorização elétrica, se obtém o resultado ambiental atingido pelo desenvolvimento deste projeto.

A plataforma de solução estabelecida neste trabalho abre possibilidades de um estudo mais aprofundado, visando então a extensão deste trabalho através de projetos futuros e a continuidade para um mestrado, ocasionando desta forma resultados científicos.

1.3 **OBJETIVOS**

1.3.1 **Objetivos Gerais**

O objetivo deste projeto é desenvolver um sistema para controle e gestão de veículos guiados automaticamente através da tecnologia de controladores lógicos programáveis, aumentar a interação homem-máquina relacionada a estes veículos e possibilitar uma melhora no desempenho da manufatura.

1.3.2 **Objetivos Específicos**

- Estabelecer a comunicação entre o software e os veículos de condução automática presentes na linha de produção.
- Desenvolver a gestão de rotas para estes veículos, realizando o envio da decisão de qual rota será executada por qual veículo da linha de produção.
- Desenvolver uma interface com níveis de acesso, contendo um modo para desenvolvedor e um modo para usuário.
- Possibilitar a criação e edição de rotas, assim como importá-las e exportá-las.
- Permitir a visualização gráfica individual ou geral da posição dos veículos conectados ao sistema em tempo real.
- Gerar um relatório de utilização dos veículos em tempo real, permitindo o monitoramento do status de cada veículo conectado ao sistema, como por exemplo, posição, nível de bateria, velocidade, direção.
- Permitir a visualização gráfica individual ou geral das rotas cadastradas no sistema.

1.4 **DIAGRAMA**

Na figura 1 é apresentado um diagrama simplificado do projeto exemplificando sua disposição de acordo com o escopo, contendo o meio de comunicação, o modelo de veículo pelo qual o software a ser desenvolvido será utilizado e o computador no qual este será instalado.

Figura 1 - Figura Ilustrativa do Projeto

Fonte: Autoria própria

1.5 **METODOLOGIA**

Nesta etapa do projeto será apresentado o plano de desenvolvimento seguido durante os períodos de elaboração do sistema. Inicialmente foram avaliadas as expectativas decorrentes da elaboração deste software e concluiu-se que este sistema deverá ser tratado como um produto destinado ao meio fabril, portanto, deve seguir os requisitos normalmente utilizados na indústria. Para tanto, observou-se os diversos métodos de desenvolvimento de software presentes na atualidade e

verificou-se que dentre estes, o que mais se encaixa no modelo de desenvolvimento para os requisitos exigidos pela indústria seria o modelo de desenvolvimento ágil de software [5], por possuir características corriqueiras ao modelo econômico capitalista, onde se tenta otimizar a produção, reduzir custos e ter rápido desenvolvimento, se adequando assim mais facilmente as necessidades do mercado industrial. Defender o planejamento adaptativo, o desenvolvimento evolutivo, a entrega precoce e a melhoria contínua do produto, e o incentivo a resposta rápida e flexível à mudança são características vitais deste modelo de desenvolvimento que se pretende adotar para este projeto.

Em um primeiro momento avaliou-se projetos semelhantes para se instruir sobre possíveis problemas que poderiam ser encontrados durante o desenvolvimento do sistema proposto. Em seguida estudou-se a viabilidade do projeto, levando em consideração as adversidades encontradas destes projetos semelhantes e os requisitos necessários para o seu desenvolvimento. A partir deste momento a implementação do código se iniciou através do estudo da linguagem de programação *C#,* necessária para o seu desenvolvimento, e posteriormente com o domínio desta linguagem o estudo da biblioteca *S7.Net Plus*. Em cada uma destas fases de aprendizado, tanto a implementação de códigos de ensaio e de testes foram direcionados ao produto final, desta forma, sendo reutilizados posteriormente durante a fase definitiva de implementação. Após a obtenção do domínio desta linguagem e biblioteca, destinou-se o foco ao estudo das características do veículo modelo *palleteira*, escopo deste projeto, para personalização do sistema, para pesquisas sobre as necessidades da gestão de veículos de condução automática e para o estudo sobre interfaces de interação homem-máquina. Neste momento a implementação do código do sistema se iniciou em definitivo. Testes e correções de problemas foram efetuados repetitivamente durante esta fase, entretanto, com a conclusão do código, existiu uma fase destinada unicamente para testes e possíveis correções necessárias.

Como foi adotado o método de desenvolvimento ágil de software, nenhumas destas fases supracitadas foram necessariamente restritas ao que foi exposto e puderam inclusive se interseccionar entre si, objetivando sempre a ampliação da aprendizagem, a redução da perda de tempo, a obtenção de resultados o quanto mais rápido for possível e visando sempre a qualidade do produto final proposto pelo projeto.

2 GESTÃO DO PROJETO

2.1 **ESCOPO**

Com este projeto pretendeu-se realizar a gestão de veículos de condução automática na manufatura fabril através do sistema a ser desenvolvido. Para a elaboração deste sistema foi utilizado o ambiente de desenvolvimento *Microsoft Visual Studio* [2], e a linguagem de programação *C#* através do *.NET Framework* [16].

Utilizou-se a biblioteca de programação *S7.Net Plus* para estabelecer a comunicação entre o veículo e o sistema. Esta conexão se realiza através de rede sem fio *Wi-Fi* por protocolo *TCP/IP*, onde um computador industrial, que possua suporte para a linguagem *C#*, hospedará a rede que faz a conexão desses veículos.

No ambiente fabril se faz necessária a utilização de computadores industriais, uma vez que são mais robustos e possuem melhores desempenhos, qualidade e foram desenvolvidos especialmente visando suprir as condições deste meio. Entretanto, para a demonstração do funcionamento deste sistema em laboratório, um computador padrão do tipo *PC* que possua suporte para a linguagem *C#* é suficiente, uma vez que o escopo deste projeto é demonstrar o funcionamento do sistema e não a qualidade e desempenho do computador em que se está instalado.

Existem atualmente diversos tipos de veículos de condução automática, como por exemplo, rebocadores, carregadores, empilhadeiras, *palleteiras*. Para cada um deste tipo de veículo, o sistema deverá ser personalizado de acordo com os requisitos particulares do veículo, como por exemplo, para uma empilhadeira qual a altura de elevação, ou para um rebocador se o pino de reboque está levantado ou não. Devido a essas particularidades, a palleteira, utilizada para o transporte de pallets, foi tomada como objeto de desenvolvimento deste projeto e, portanto, todas as características peculiares a este tipo de veículo foram utilizadas para a elaboração do sistema. O controlador lógico programável utilizado pelo AGV adotado foi um *Siemens* modelo *ET-200*.

Dentre os diversos tipos de veículos, existem também os mais variados tipos de sistemas de navegação, e cada um deles se orienta de uma maneira específica. Fita magnética, óptico, cabo e laser são os mais comuns métodos de orientação de veículos de condução automática, e cada um deles, assim como cada modelo de veículo, possui suas características particulares, e o sistema a ser desenvolvido deverá ser ajustado para que corrobore com as especificações de direção necessárias. Tendo em vista estas particularidades, utilizou-se a navegação a laser para orientação, e o sistema foi personalizado para suprir as demandas necessárias deste método.

O sistema foi projetado para ter uma interface com dois níveis de acesso, contendo um modo para desenvolvedor e um modo para usuário. No modo de desenvolvedor, uma interface menos visual e mais objetiva, auxiliando na depuração de problemas e edição de configurações. No modo de usuário, uma interface homem-máquina com propósito de melhor visualização das rotas e das tomadas de decisões realizadas pelo software, bem como acompanhamento em tempo real das rotas sendo realizadas e das características momentâneas de cada veículo, como por exemplo, nível de bateria, velocidade e posição.

Embora ocorram alguns aspectos particulares no desenvolvimento deste sistema, como a escolha do tipo de veículo e navegação, a base pode ser utilizada em quaisquer outros tipos de veículo presente no chão de fábrica e citados anteriormente, uma vez que o sistema pode ser personalizado para qualquer um dos veículos citados e não desenvolvido exclusivamente para *palleteiras*.

2.2 **ANÁLISE DE RISCOS**

O projeto proposto envolveu alguns riscos pertinentes ao seu desenvolvimento, e estes estão apresentados a seguir no quadro 1.

Quadro 1 - Riscos do Projeto

Risco 1			
Importância:	Alto: ( )	Médio: ( )	Baixo: (X)
Descrição: Falta de experiência com a linguagem C#			
Efeitos no projeto: A linguagem C# é base para desenvolvimento desta aplicação.			
Probabilidade:	Alto: ( )	Médio: ( )	Baixo: (X)
Impacto	Alto: (X)	Médio: ( )	Baixo: ( )
Ação: Estudar a linguagem C#.			

Risco 2			
Importância:	Alto: ( )	Médio: ( )	Baixo: (X)
Descrição: Falta de experiência com a biblioteca *S7.Net Plus*			
Efeitos no projeto: O projeto é uma aplicação baseada na conexão entre o veículo controlado por um CLP e o sistema, portanto o domínio da biblioteca responsável por essa conexão é indispensável.			
Probabilidade:	Alto: ( )	Médio: ( )	Baixo: (X)
Impacto	Alto: (X)	Médio: ( )	Baixo: ( )
Ação: Estudar a biblioteca *S7.Net Plus*			

Risco 3			
Importância:	Alto: (X)	Médio: ( )	Baixo: ( )
Descrição: Danificação do sistema de orientação por laser.			
Efeitos no projeto: O desenvolvimento do sistema será personalizado para veículos que possuam orientação guiada por laser, sem este não há como realizar testes do sistema.			
Probabilidade:	Alto: ( )	Médio: (X)	Baixo: ( )
Impacto	Alto: (X)	Médio: ( )	Baixo: ( )
Ação: Conviver, manusear com cautela o aparelho.			

Risco 4			
Importância:	Alto: (X)	Médio: ( )	Baixo: ( )
Descrição: Danificação do veículo guiado automaticamente.			
Efeitos no projeto: Os testes do sistema baseiam-se todos num veículo guiado automaticamente modelo paleteira, portanto o seu correto funcionamento é essencial para realizar os testes deste sistema.			
Probabilidade:	Alto: ( )	Médio: (X)	Baixo: ( )
Impacto	Alto: (X)	Médio: ( )	Baixo: ( )
Ação: Conviver, manusear com cautela o veículo. Desenvolver um simulador.			

Risco 5			
Importância:	Alto: ( )	Médio: ( )	Baixo: (X)
Descrição: Danificação do computador utilizado para programação e perda do código			
Efeitos no projeto: Perda do progresso realizado no desenvolvimento do sistema.			
Probabilidade:	Alto: ( )	Médio: ( )	Baixo: (X)
Impacto	Alto: (X)	Médio: ( )	Baixo: ( )
Ação: Utilizar gerenciamento de controle de versão em nuvem como *github*.			

2.3 **CRONOGRAMA E CUSTOS**

O cronograma de execução do projeto será apresentado nos itens que seguem, bem como seus custos em horas, e foi baseado numa estimativa de planejamento de tempo médio necessário para a execução de cada tarefa.

- Pesquisar por projetos similares, viabilidade do projeto e requisitos necessários para a realização. Duração de um mês com 30 horas efetivas.
- Estudar a linguagem C#, a biblioteca necessária para a conexão do sistema com o veículo e as características do veículo a ser utilizado nesse projeto. Duração de um mês com 40 horas efetivas.
- Pesquisar sobre as necessidades ao se gerenciar veículos de condução automática e estudar as interfaces de sistemas de interação homem-máquina. Duração de duas semanas com 25 horas efetivas.
- Implementar o modo de desenvolvedor do sistema. Duração de dois meses com 150 horas efetivas.
- Implementar o modo de usuário do sistema. Duração de um mês e meio com 100 horas efetivas.

- Melhorar o sistema, corrigir bugs e testar em campo. Duração de 20 dias com 45 horas efetivas.
- Redigir a monografia. Duração de 4 meses com 160 horas efetivas.

Quadro 2 - Cronograma do Projeto

Atividade	**Período**	**Horas**
Pesquisar por projetos similares	01/10/2016 a 10/10/2016	10h
Estudar viabilidade do projeto	10/10/2016 a 31/10/2016	10h
Pesquisar os requisitos necessários para o desenvolvimento do projeto	10/10/2016 a 31/10/2016	10h
Estudar a linguagem C#	01/11/2016 a 15/11/2016	15h
Estudar a biblioteca *S7.Net Plus*	10/11/2016 a 25/11/2016	15h
Estudar as características do veículo modelo palleteira, escopo deste projeto, para personalização do sistema.	25/11/2016 a 30/11/2016	10h
Pesquisar as necessidades da gestão de veículos automaticamente guiados	01/12/2016 a 15/12/2016	15h
Estudar interfaces de interação homem-máquina	01/12/2016 a 15/12/2016	10h
Implementar *Software* do sistema: modo desenvolvedor	15/12/2016 a 15/02/2017	150h
Implementar *Software* do sistema: modo usuário	15/02/2017 a 30/03/2017	100h
Realizar melhorias e correção de bugs	01/04/2017 a 15/04/2017	20h
Testar em campo	15/04/2017 a 20/04/2017	25h
Redigir Monografia	15/02/2017 a 15/06/2017	160h
Total de horas	550h	

3 FUNDAMENTAÇÃO TEÓRICA

Nesta etapa do documento será exposta a fundamentação teórica para o embasamento deste projeto.

3.1 VEÍCULOS DE CONDUÇÃO AUTOMÁTICA

Entende-se por veículo de condução automática, da sigla em inglês *AGV – automated guided vehicle,* um robô móvel que segue marcadores ou fios no chão, ou usa visão, ímãs ou lasers para navegação, se deslocando de forma autônoma e independente. Eles são mais frequentemente utilizados em aplicações industriais para mover materiais ao longo da planta de uma fábrica ou armazém. Dentre estes veículos de condução automática existem diversos modelos diferentes, cada um com suas características particulares, que se adaptam ou se encaixam melhor de acordo com o modo a ser utilizado. Os diversos modelos serão apresentados a seguir bem como os seus respectivos sistemas de navegação.

Abaixo verificamos dois exemplos de utilização de veículos autoguiados. Na figura 2 uma empilhadeira auxiliando no transporte de pallets pelo meio fabril e na figura 3, o deslocamento pela linha de montagem da carcaça de um trator enquanto é realizada a sua montagem.

Figura 2 – AGV transportando *pallets*
Fonte: Wikimedia

Figura 3 – AGV na linha de produção
Fonte: ytimg

3.1.1 **Modelos de Veículos**

Existem diversos modelos diferentes de *AGVs* que visam suprir diferentes necessidades específicas, sendo assim necessário personalizar o software a ser desenvolvido para que este atenda aos requisitos exigidos por cada modelo de veículo autoguiado.

Na figura 4 são apresentados alguns modelos de veículos: *palleteira*, empilhadeira, rebocador, transportador. Escolheu-se para o escopo deste projeto o modelo *palleteira*.

Figura 4 - Modelos de *AGVs*
Fonte: Forklift, Manutenção e Suprimentos, Techtonagv

3.1.2 **Modelos de Orientação**

Além do modelo do *AGV*, é necessário também personalizar o sistema de acordo com o modelo de orientação utilizado pelo veículo para que os dados sejam adquiridos e tratados adequadamente. Dentre os diversos modelos para navegação

existentes, estão entre os principais os modelos por fio, fita, laser, inercial, visão e *GPS*.

O sistema de navegação escolhido a ser utilizado neste projeto foi o modelo a *laser*. Esta navegação é realizada através da instalação de fitas reflexivas ou espelhos espalhados na mesma altura do *scanner* a laser. O *AGV,* portanto, transporta um transmissor e receptor de laser em uma torre rotativa que realiza diversas varreduras por segundo. O ângulo e a distância para quaisquer refletores na linha de visão do *scanner* são automaticamente computados através de cálculos trigonométricos pertinentes, permitindo ao sistema de navegação triangular a posição atual do AGV, retornando para o software a posição métrica em coordenadas cartesianas e o ângulo em relação ao ponto inicial previamente definido pelo software do sistema de navegação a *laser.* O navegador a *laser* utilizado é da marca *Sick* modelo *NAV350* [12], exposto na figura 5 abaixo.

Figura 5 – Modelo do Scanner de Orientação a Laser
Fonte: Sick

É possível, portanto, concluir que para que este sistema funcione seja necessário que pelo menos três objetos reflexivos estejam visíveis para o sistema de navegação *laser*, permitindo a localização do *AGV* através de um algoritmo de

triangulação. Na figura 6, a ilustração do funcionamento deste modelo de navegação.

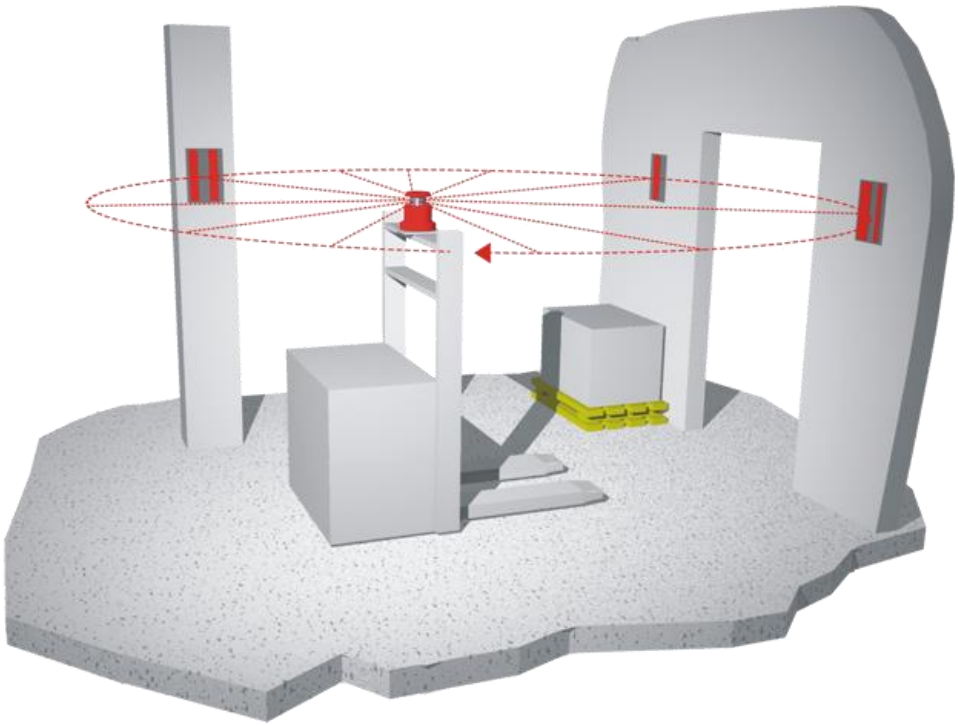


Figura 6 - Modelo de Navegação a *Laser*
Fonte: *Goetting-agv*

3.2 **SISTEMAS DE CONTROLE SUPERVISÓRIO**

Entende-se por controle supervisório um sistema capaz de capturar, armazenar informações sobre um processo produtivo ou instalação física e monitorar esses dados para a tomada de decisão sobre o que fazer a partir da análise das informações obtidas. Estas informações são provenientes da obtenção de dados de sensores presentes na planta industrial. Posteriormente estes dados são processados, manipulados e apresentados ao usuário. Estes sistemas também são conhecidos como *SCADA*, do inglês *Supervisory Control and Data Aquisition*, referindo-se à combinação de telemetria e aquisição de dados [3].

3.3 **CONTROLADOR LÓGICO PROGRAMÁVEL**

O controlador lógico programável, do inglês *PLC - programmable logic controller*, foi projetado para monitorar e administrar máquinas e processos industriais. Este, que executa o controle de máquinas através de software desenvolvido pelo utilizador, é um computador industrial baseado em microprocessador especializado em desempenhar funções pertinentes ao meio fabril. O *CLP* é robusto e foi desenvolvido para suportar as adversidades encontradas no ambiente industrial, assim como água, sujeira em excesso e temperaturas extremas. Utilizam-se de uma forma especial de microprocessador de controle capaz de executar instruções e implementar operações lógicas, sequenciais, de tempo, contador e aritméticas através de memorias programáveis para controlar máquinas e processos [4].

3.3.1 **CONTROLADOR SIEMENS ET200**

Dentre as diversas marcas de controladores, como por exemplo, ABB [8], B&R [10], Phoenix Contact [4], Allen-Bradley-Rockwell [9], foi adotada a marca Siemens [14] pela questão de confiabilidade, aceitabilidade de mercado e disponibilidade para testes. Dentro de cada uma destas marcas existem ainda famílias variadas de controladores, que tem por objetivo suprir de formas e com potenciais diferentes os requisitos de projeto. Para este projeto foi utilizado o controlador ET200 [13]. Na figura 7 encontra-se uma ilustração do modelo, contendo seus respectivos cartões de entradas e saídas digitais e analógicas.

.

Figura 7 – Controlador Siemens ET200s
Fonte: Aotewell

Para esta família de controlador está disponível uma variedade de tipos de dados para realizar o armazenamento de informações. O quadro 3 exibe os tipos elementares de dados a serem utilizados pelo software desenvolvido, o seu tamanho em bits, o formato de utilização e a notação e escala, que se fazem essenciais para a realização da troca de dados com o *CLP*. É necessário ter conhecimento destes dados para efetuar uma correta utilização da biblioteca *S7.Net Plus* [15] ao se ler e escrever dados no controlador.

Quadro 3 - Tipos Elementares de Dados para Controlador *ET200*
Fonte: *PLCDEV*

Tipo e Descrição	**Tamanho em bits**	**Opções de Formatação**	**Notação e escala**
BOOL (Bit)	1	Booleano	TRUE/FALSE
BYTE (Byte)	8	Número hexadecimal	B#16#0 até B#16#FF
WORD (Word)	16	Número binário	2#0 até 2#1111_1111_1111_1111
		Número hexadecimal	W#16#0 até W#16#FFFF

		BCD	C#0 até C#999
		Número decimal sem sinal	B#(0,0) até B#(255,255)
DWORD (Double word)	32	Número binário	2#0 até 2#1111_1111_1111_1111_ 1111_1111_1111_1111
		Número hexadecimal	W#16#0000_0000 até W#16#FFFF_FFFF
		Número decimal sem sinal	B#(0,0,0,0) até B#(255,255,255,255)
INT (Integer)	16	Número decimal com sinal	-32768 até 32767
DINT (Double integer)	32	Número decimal com sinal	L#-2147483648 até L#2147483647
REAL (Floating-point number)	32	Número de ponto flutuante IEEE	Limite superior +/-3.402823e+38 Limite inferior +/-1.175495e-38
S5TIME (SIMATIC time)	16	Tempo S7 em espaços de 10ms	S5T#0H_0M_0S_10MS até S5T#2H_46M_30S_0MS e S5T#0H_0M_0S_0MS
TIME (IEC time)	32	Tempo em espaços de 1ms, inteiro com sinal	T#24D_20H_31M_23S_648MS até T#24D_20H_31M_23S_647MS
DATE (IEC date)	16	Data IEC em espaços de 1 dia	D#1990-1-1 até D#2168-12-31
TIME _OF_DAY (Time)	32	Tempo em espaços de 1ms	TOD#0:0:0.0 até TOD#23:59:59.999
CHAR (Character)	8	Caracteres ASCII	'A', 'B' etc.

3.4 **PATENTES E PROPRIEDADES**

Nas pesquisas realizadas, não foram encontrados sistemas ou produtos que se assemelhem ao escopo do software proposto neste documento e que pudessem entrar no estudo pretendido por este tópico. As pesquisas foram realizadas em bases de dados do Brasil, dos Estados Unidos e Europa.

4 DESENVOLVIMENTO

4.1 **CARACTERÍSTICAS GERAIS**

Este sistema foi personalizado para uma vertente no seguimento de veículos de condução automática, e embora ocorram alguns aspectos específicos no desenvolvimento deste sistema, a base pode ser utilizada em quaisquer outros tipos destas particularidades, uma vez que o sistema é personalizado para estas finalidades e não desenvolvido exclusivamente para elas.

4.2 **REQUISITOS**

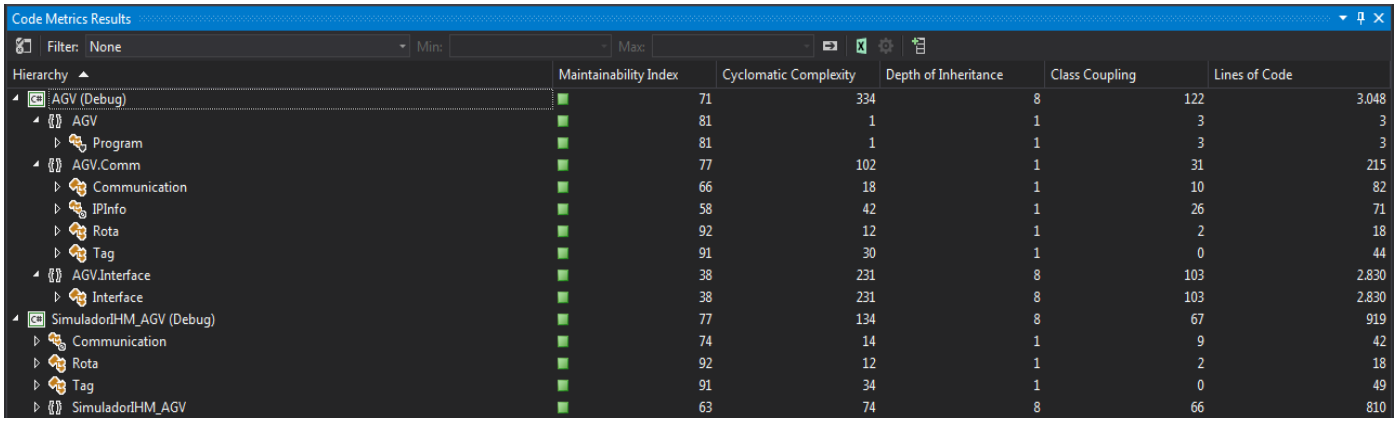
Para a instalação deste sistema no ambiente fabril, se faz necessário um computador industrial, uma vez que este está preparado para enfrentar as adversidades encontradas neste meio, assim como água, sujeira em excesso e calor extremo. O computador a ser utilizado deve possuir suporte para a linguagem *C#,* contendo a componente do pacote *.NET Framework* 3.5 [16] ou superior da *Microsoft*, e ter conectividade com a rede na qual os veículos estão ligados.

4.3 **ESTRUTURA**

Nesta etapa será descrito a hierarquização do software projetado e apresentado através de imagens a sua estrutura de classes, bem como uma análise quantitativa de linhas de códigos escritas.

A figura 8 apresenta a divisão dos pacotes e as classes presentes no software juntamente da quantidade de linhas de código atribuídas a cada uma dessas classes.

A classe *Program* é responsável pela inicialização do código. A classe *Communication* pelo tratamento, recebimento e envio de dados entre o computador hospedeiro do software e os *AGVs* conectados. A classe *IPInfo* é responsável pela aquisição automática dos veículos conectados à rede. A classe *Rota* é constituída dos diversos pontos a serem percorridos por cada *AGV*. Estes pontos, por sua vez, são descritos pela classe *Tag*. A classe *Interface* é responsável por implementar a interface do software, contendo ambos modos desenvolvedor e usuário.



Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
AGV (Debug)	71	334	8	122	3.048
AGV	81	1	1	3	3
Program	81	1	1	3	3
AGV.Comm	77	102	1	31	215
Communication	66	18	1	10	82
IPInfo	58	42	1	26	71
Rota	92	12	1	2	18
Tag	91	30	1	0	44
AGV.Interface	38	231	8	103	2.830
Interface	38	231	8	103	2.830
SimuladorIHM_AGV (Debug)	77	134	8	67	919
Communication	74	14	1	9	42
Rota	92	12	1	2	18
Tag	91	34	1	0	49
SimuladorIHM_AGV	63	74	8	66	810

Figura 8 - Divisão de Classes

Fonte: Autoria própria

Na figura 9 é possível verificar a hierarquização das classes presentes no software.

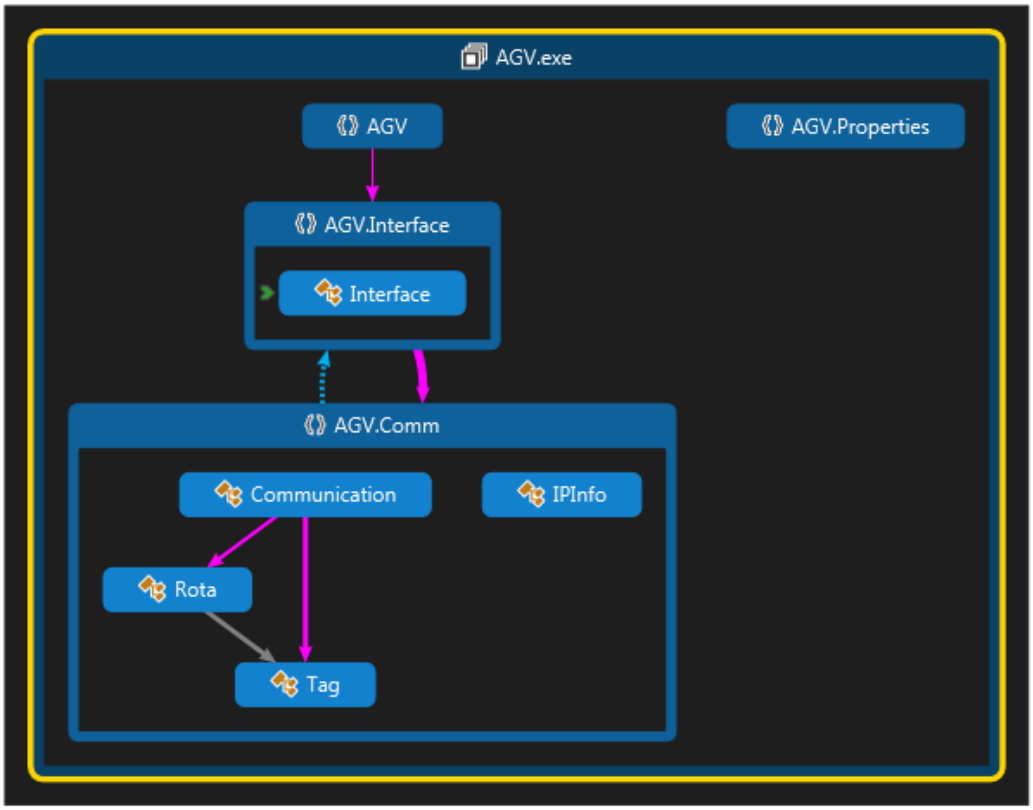


Figura 9 - Hierarquização das Classes
Fonte: Autoria própria

Na figura 10 são mostradas as funções internas da classe *Communication* relativas à comunicação com o *AGV* no quesito responsável pelo tratamento, envio e recebimento de dados.

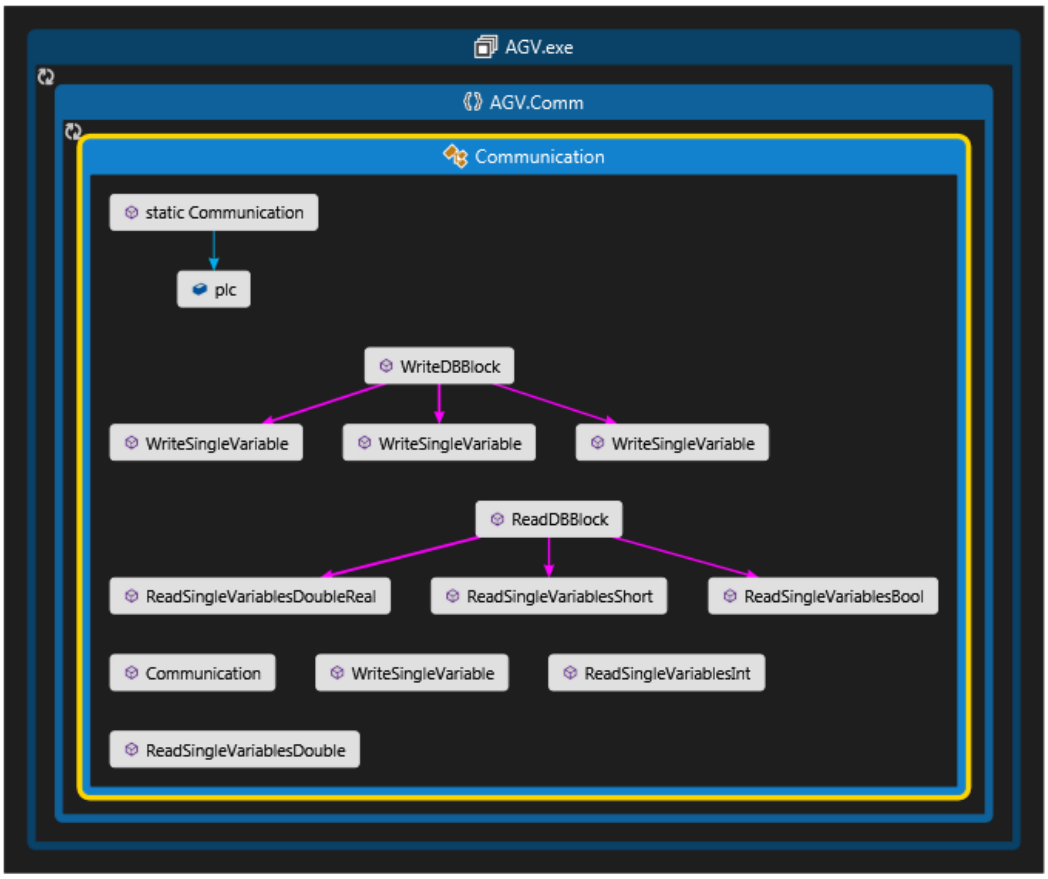


Figura 10 - Detalhamento da Classe de Comunicação
Fonte: Autoria própria

A descrição do conteúdo de cada objeto *Rota* contido neste programa se encontra na figura 11. Cada rota que o *AGV* deve percorrer contém diversos pontos, que neste escopo são chamados de *Tags* e identificados cada um como um objeto da classe *Tag*.

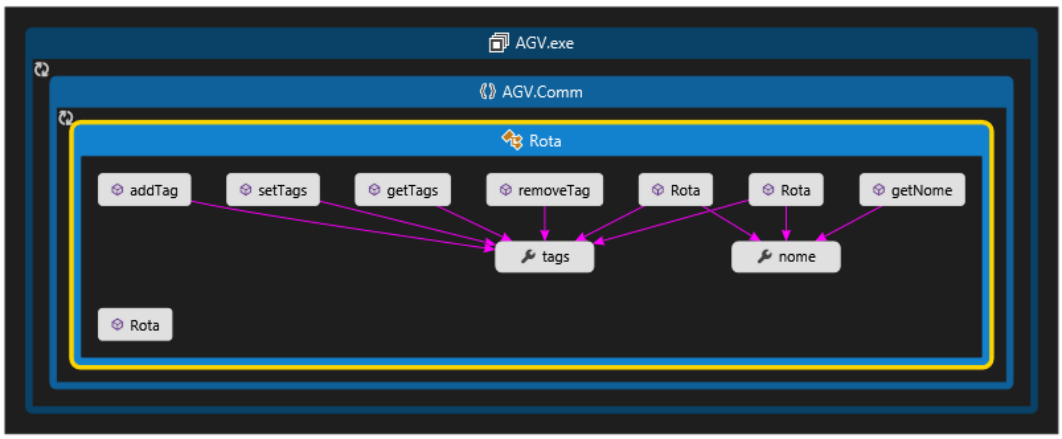


Figura 11 - Detalhamento da Classe *Rota*
Fonte: Autoria própria

Na figura 12 verifica-se o conteúdo de cada ponto na rota a ser percorrida pelo *AGV*, detalhados pela classe *Tag*.

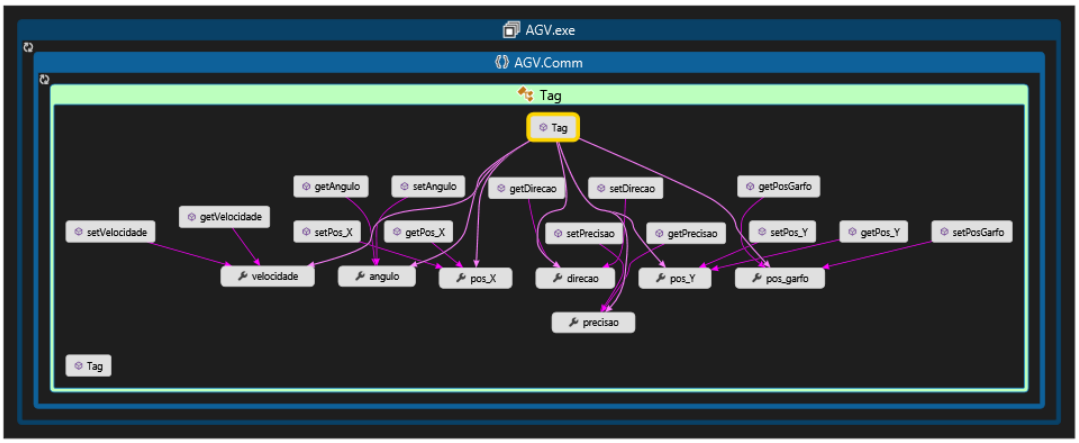


Figura 12 - Detalhamento da Classe *Tag*
Fonte: Autoria própria

A biblioteca *S7.Net Plus* utilizada para o estabelecimento da conexão pode ser verificada hierarquicamente dentro do código através da figura 13.

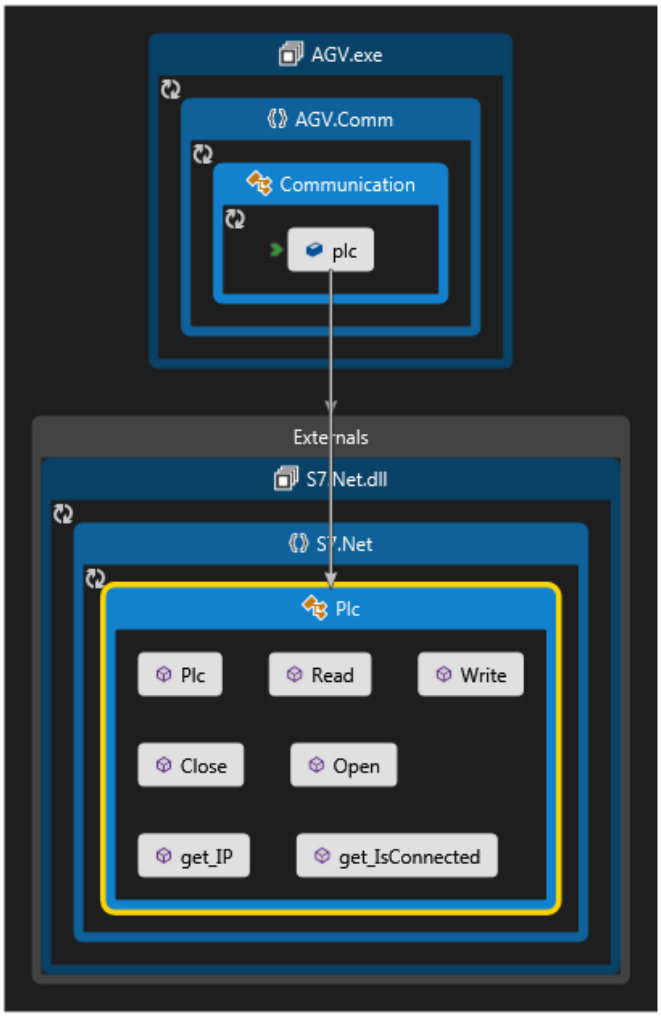


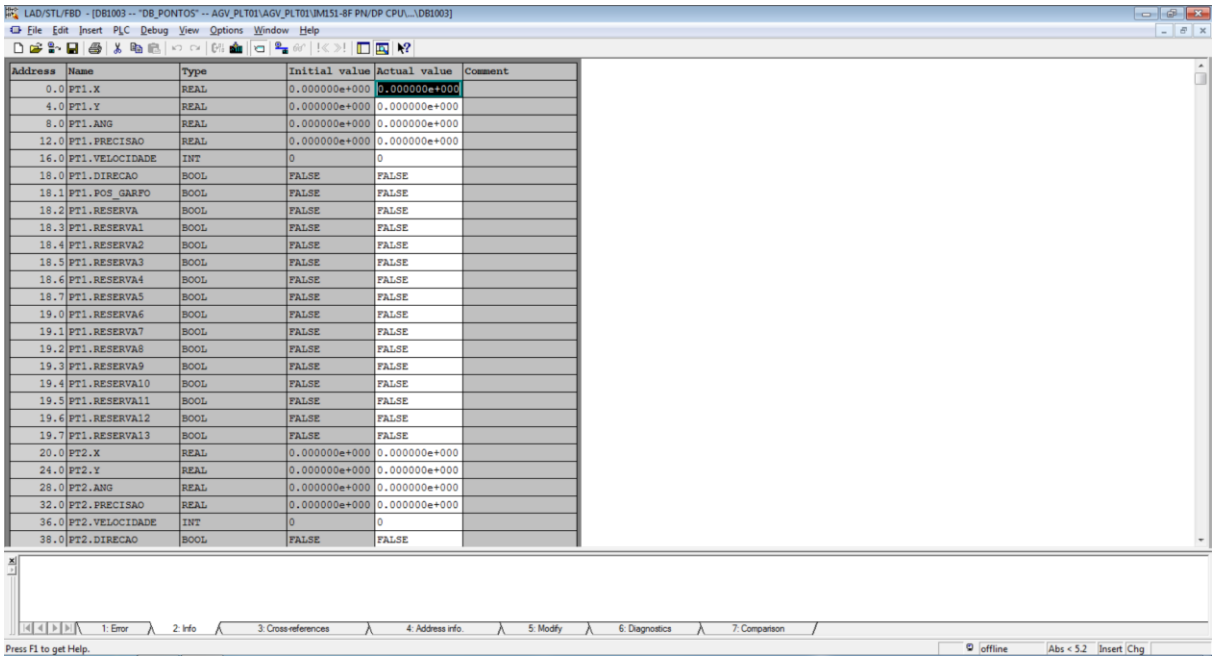
Figura 13 - Detalhamento da Biblioteca de Comunicação *S7.Net Plus*
Fonte: Autoria própria

4.4 **ARMAZENAMENTO DE DADOS NO CLP**

O controlador Siemens ET200 utiliza uma estrutura de dados chamada de *DB* para realizar a armazenagem de informações pertinentes ao programa, além de uma área de memória simbolizada por *M* reservada para acesso direto independente. Estas *DBs* podem ser geradas de forma independente, para utilização na armazenagem de dados destinadas ao programa como um todo, ou então serem utilizadas como uma memória instanciada de um bloco de função, chamado de *FB*, para guardar dados exclusivos da função executada pelo bloco. Essas memórias podem ou não ser retentivas, ou seja, após o desligamento do CLP as informações podem permanecer ou serem apagadas da memória.

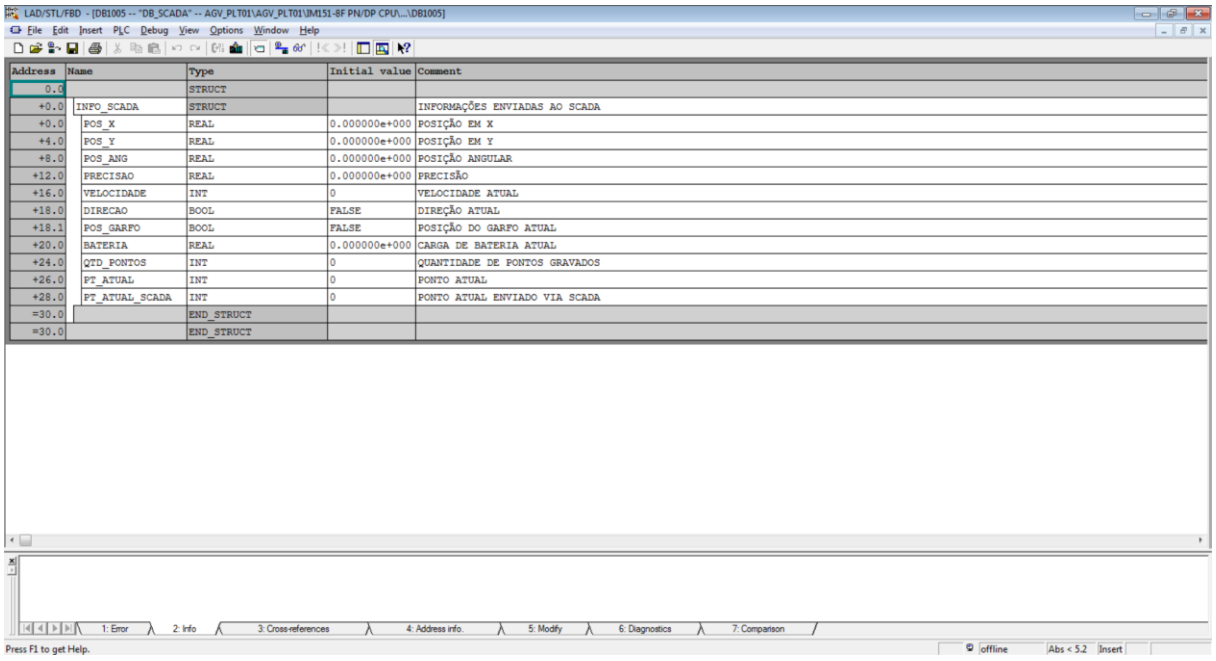
Para o escopo deste projeto serão necessárias duas *DB*s, uma será retentiva e conterá as inúmeras coordenadas cartesianas métricas e angulação para realizar o deslocamento do veículo pela planta da fábrica e as respectivas características de cada ponto a serem executadas pelo *AGV,* como por exemplo, velocidade, direção e posição do garfo, e a outra, não retentiva, servirá de meio de troca de dados entre o CLP e o software desenvolvido, para mostrar a situação em tempo real do AGV.

Na figura 14 encontra-se a *DB* com o seu conteúdo explicitado pelos pontos de cada uma das coordenadas e na figura 15 a estrutura de informação para as trocas de dados em tempo real com o AGV.



Address	Name	Type	Initial value	Actual value	Comment
0.0	PT1.X	REAL	0.000000e+000	0.000000e+000	
4.0	PT1.Y	REAL	0.000000e+000	0.000000e+000	
8.0	PT1.ANG	REAL	0.000000e+000	0.000000e+000	
12.0	PT1.PRECISAO	REAL	0.000000e+000	0.000000e+000	
16.0	PT1.VELOCIDADE	INT	0	0	
18.0	PT1.DIRECAO	BOOL	FALSE	FALSE	
18.1	PT1.POS_GARFO	BOOL	FALSE	FALSE	
18.2	PT1.RESERVA	BOOL	FALSE	FALSE	
18.3	PT1.RESERVA1	BOOL	FALSE	FALSE	
18.4	PT1.RESERVA2	BOOL	FALSE	FALSE	
18.5	PT1.RESERVA3	BOOL	FALSE	FALSE	
18.6	PT1.RESERVA4	BOOL	FALSE	FALSE	
18.7	PT1.RESERVA5	BOOL	FALSE	FALSE	
19.0	PT1.RESERVA6	BOOL	FALSE	FALSE	
19.1	PT1.RESERVA7	BOOL	FALSE	FALSE	
19.2	PT1.RESERVA8	BOOL	FALSE	FALSE	
19.3	PT1.RESERVA9	BOOL	FALSE	FALSE	
19.4	PT1.RESERVA10	BOOL	FALSE	FALSE	
19.5	PT1.RESERVA11	BOOL	FALSE	FALSE	
19.6	PT1.RESERVA12	BOOL	FALSE	FALSE	
19.7	PT1.RESERVA13	BOOL	FALSE	FALSE	
20.0	PT2.X	REAL	0.000000e+000	0.000000e+000	
24.0	PT2.Y	REAL	0.000000e+000	0.000000e+000	
28.0	PT2.ANG	REAL	0.000000e+000	0.000000e+000	
32.0	PT2.PRECISAO	REAL	0.000000e+000	0.000000e+000	
36.0	PT2.VELOCIDADE	INT	0	0	
38.0	PT2.DIRECAO	BOOL	FALSE	FALSE	

Figura 14 – DB com coordenadas e características de cada ponto
Fonte: Autoria própria



Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	INFO_SCADA	STRUCT		INFORMAÇÕES ENVIADAS AO SCADA
+0.0	POS_X	REAL	0.000000e+000	POSIÇÃO EM X
+4.0	POS_Y	REAL	0.000000e+000	POSIÇÃO EM Y
+8.0	POS_ANG	REAL	0.000000e+000	POSIÇÃO ANGULAR
+12.0	PRECISAO	REAL	0.000000e+000	PRECISÃO
+16.0	VELOCIDADE	INT	0	VELOCIDADE ATUAL
+18.0	DIRECAO	BOOL	FALSE	DIREÇÃO ATUAL
+18.1	POS_GARFO	BOOL	FALSE	POSIÇÃO DO GARFO ATUAL
+20.0	BATERIA	REAL	0.000000e+000	CARGA DE BATERIA ATUAL
+24.0	QTD_PONTOS	INT	0	QUANTIDADE DE PONTOS GRAVADOS
+26.0	PT_ATUAL	INT	0	PONTO ATUAL
+28.0	PT_ATUAL_SCADA	INT	0	PONTO ATUAL ENVIADO VIA SCADA
=30.0		END_STRUCT		
=30.0		END_STRUCT		

Figura 15 – DB para a troca de dados entre AGV e software
Fonte: Autoria própria

4.5 **BIBLIOTECA S7.NET PLUS**

Uma parte fundamental para o desenvolvimento deste projeto foi a utilização de uma biblioteca de comunicação entre o software e o controlador para realizar a leitura e escrita dos dados nas memórias de armazenamento descritas na sessão de *Armazenamento de Dados no CLP* deste documento. A linguagem de programação escolhida para a implementação da solução foi o *C#*, e para isto foi utilizada a biblioteca *S7.Net Plus*, disponível em [15]. Utilizando-se o *Microsoft Visual Studio,* disponível em [2], é possível inserir esta biblioteca através do gestor de pacotes da solução de desenvolvimento pelo *NuGet*, buscando por *S7netplus* e instalando a sua ultima versão disponível online. A figura 16 exibe como o projeto deverá ficar após a instalação da biblioteca na solução.

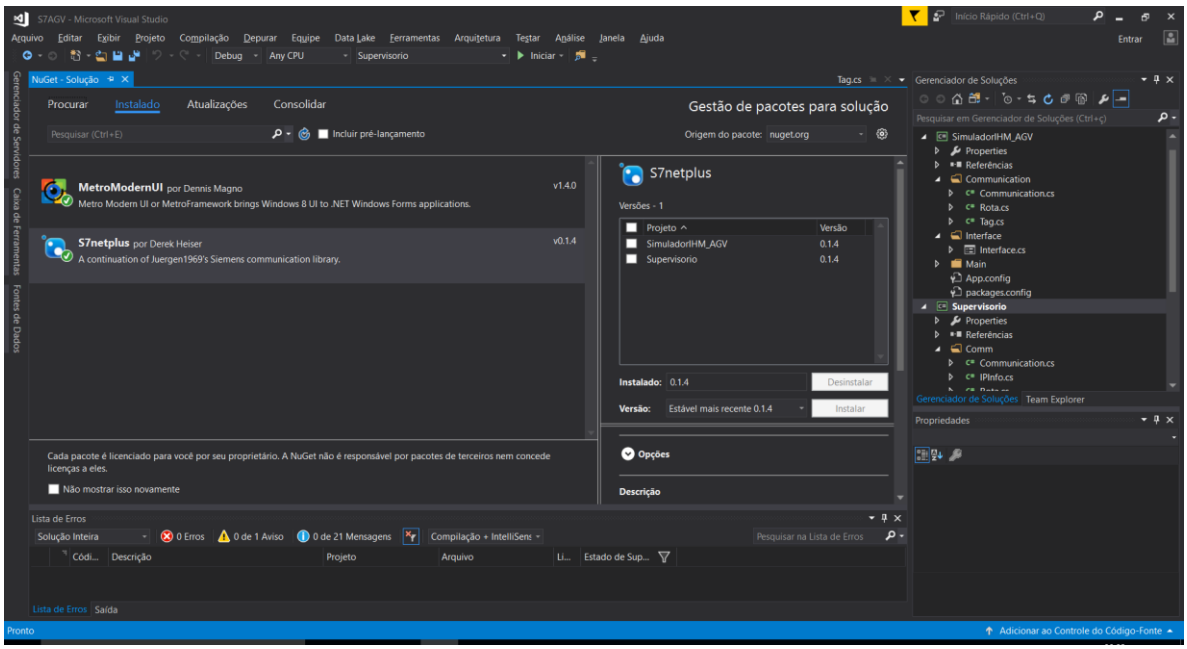


Figura 16 – Instalando a Biblioteca *S7.Net Plus*
Fonte: Autoria própria

Os controladores que possuem suporte para esta biblioteca são os da família S7-200, S7-300, S7-400, S7-1200 e S7-1500. O controlador escolhido no escopo deste projeto está incluído na família S7-200, modelo *ET200s*. Para a utilização desta biblioteca é necessário possuir instalado no computador hospedeiro

do software o *Framework .NET 3.5* ou superior, como citado no capítulo sobre *Requisitos* deste documento.

Para o estabelecimento da conexão entre controlador e software com esta biblioteca é necessário, no código, criar uma instância do controlador a se conectar, através do construtor, como mostrado na figura 17.

```
public Plc(CpuType cpu, string ip, Int16 rack, Int16 slot)
```

Figura 17 – Construtor da Instância do Controlador no Software
Fonte: *S7.Net Plus*

O parâmetro *CpuType* especifica a família do controlador a se conectar, e deve seguir a numeração contida na figura 18.

```
//     Types of S7 cpu supported by the library
public enum CpuType
{
    //
    // Resumo:
    //     S7 200 cpu type
    S7200 = 0,
    //
    // Resumo:
    //     S7 300 cpu type
    S7300 = 10,
    //
    // Resumo:
    //     S7 400 cpu type
    S7400 = 20,
    //
    // Resumo:
    //     S7 1200 cpu type
    S71200 = 30,
    //
    // Resumo:
    //     S7 1500 cpu type
    S71500 = 40
}
```

Figura 18 – Controladores Suportados pela Biblioteca *S7.Net Plus*
Fonte: Autoria própria

O parâmetro *ip* especifica a qual IP deverá ser conectado, referente ao identificador do veículo. Os parâmetros *rack* e *slot* devem ser configurados de acordo com o que foi projetado e programado na interface de desenvolvimento da *Siemens Simatic Step7* para controladores da marca.

Cria-se um objeto do veículo pelo procedimento mostrado na figura 19 através do construtor da classe.

```
// Definição de uma comunicação com o clp (tipo, endereço IP, rack, slot)
Plc plc = new Plc(CpuType.S7200, ip, 0, 2);
```

Figura 19 – Criando um Objeto *PLC* da Biblioteca *S7.Net Plus*
Fonte: Autoria própria

O próximo passo é abrir a conexão entre o software e o CLP, como apresentado na figura 20. Esta função retorna um booleano como resultado do status da conexão, sendo possível então com este resultado tratar erros referentes a abertura desta conexão, como por exemplo, exibir para o usuário se a conexão foi bem sucedida.

```
// Execução do código de abertura da conexão com o clp
var result = plc.Open();
```

Figura 20 – Abrindo a Conexão
Fonte: Autoria própria

Realiza-se o mesmo procedimento supracitado para encerrar a conexão, através da função exibida na figura 21.

```
plc.Close();
```

Figura 21 - Encerrando a Conexão
Fonte: Autoria própria

Para verificar se o controlador do IP que se deseja conectar está disponível, utiliza-se a propriedade do objeto exibida na figura 22.

```
bool disponivel = plc.IsAvailable;
```

Figura 22 – Disponibilidade do CLP
Fonte: Autoria própria

Após a conexão, para verificar se o controlador do IP que se está conectado ainda permanece ativo, utiliza-se a propriedade do objeto exibida na figura 23.

```
bool conectado = plc.IsConnected;
```

Figura 23 – Conexão Ativa com o CLP
Fonte: Autoria própria

Faz-se necessário então, após estabelecer a conexão e criar os objetos referentes a cada veículo a ser controlado pelo software, realizar os procedimentos de leitura e escrita nos blocos de armazenamento de dados do controlador das informações pertinentes a cada rotina de execução.

Para realizar a leitura dos dados a partir das informações armazenadas no controlador é importante observar a necessidade das devidas conversões entre os formatos de dados da linguagem *C#* e S7-200. Na figura 24 observa-se a leitura de um booleano a partir do endereço estabelecido na variável *db,* escrito, por exemplo, no formato *“DB1003.DBX18.0”*.

```
// leitura de uma variavel boolean na conexão plc na db endereçada
4 referências
public static bool ReadSingleVariablesBool(Plc plc, string db)
{
    // return (bool) plc.Read(db);
    return Convert.ToBoolean(plc.Read(db));
}
```

Figura 24 – Leitura de um booleano com a Biblioteca *S7.Net Plus*
Fonte: Autoria própria

Na figura 25 observa-se a leitura de uma variável do tipo *Int* ou *Word* no controlador, fazendo-se necessário o *cast* com o tipo *ushort* da linguagem *C#*, a partir do endereço estabelecido na variável *db,* escrito, por exemplo, no formato *"DB1003.DBW16"*.

```
// leitura de uma variavel short (Int, Word) na conexão plc na db endereçada
4 referências
public static string ReadSingleVariablesShort(Plc plc, string db)
{
    return (((ushort)plc.Read(db)).ConvertToShort()).ToString();
}
```

Figura 25 - Leitura de um *Int* ou *Word* com a Biblioteca *S7.Net Plus*
Fonte: Autoria própria

Na figura 26 observa-se a leitura de uma variável do tipo *Real* no controlador, fazendo-se necessário o *cast* com o tipo *uint* e a utilização da função *ConvertToDouble()* da linguagem *C#*, a partir do endereço estabelecido na variável *db,* escrito, por exemplo, no formato *"DB1003.DBD12"*.

```
// leitura de uma variavel double (Real) na conexão plc na db endereçada
0 referências
public static string ReadSingleVariablesDouble(Plc plc, string db)
{
    //return ((uint)(plc.Read(db))).ConvertToDouble().ToString("#.#####");
    return ((uint)(plc.Read(db))).ConvertToDouble().ToString();
}
```

Figura 26 - Leitura de um *Real* com a Biblioteca *S7.Net Plus*
Fonte: Autoria própria

Na figura 27 observa-se a leitura de uma variável do tipo *DInt* ou *DWord* no controlador, fazendo-se necessário o *cast* com o tipo *uint* e a utilização da função *ConvertToInt()* da linguagem *C#*, a partir do endereço estabelecido na variável *db,* escrito, por exemplo, no formato *"DB1003.DBD40"*.

```
// leitura de uma variavel int (DInt, DWord) na conexão plc na db endereçada
0 referências
public static string ReadSingleVariablesInt(Plc plc, string db)
{
    return (((uint)plc.Read(db)).ConvertToInt()).ToString();
}
```

Figura 27 - Leitura de um *DInt* ou *DWord* com a Biblioteca *S7.Net Plus*
Fonte: Autoria própria

Para realizar a escrita dos dados a partir das informações armazenadas no software é importante observar a necessidade das devidas conversões entre os formatos de dados da linguagem *C#* e S7-200. Na figura 28 observa-se a escrita de um booleano a partir do endereço estabelecido na variável *db,* escrito, por exemplo, no formato *"DB1003.DBX18.0"*.

```
// escreve uma variavel boolean na conexão plc na db endereçada
2 referências
public static void WriteSingleVariable(Plc plc, bool boolean, string db)
{
    plc.Write(db, boolean);
}
```

Figura 28 - Escrita de um booleano com a Biblioteca *S7.Net Plus*
Fonte: Autoria própria

Na figura 29 observa-se a escrita de uma variável do tipo *Int* ou *Word* no controlador, fazendo-se necessário a utilização da conversão pelo método *ConvertToUshort()* da linguagem *C#*, no endereço estabelecido na variável *db,* escrito, por exemplo, no formato *"DB1003.DBW16"*.

```
// escreve uma variavel short (Int, Word) na conexão plc na db endereçada
0 referências
public static void WriteSingleVariable(Plc plc, short valor, string db)
{
    plc.Write(db, valor.ConvertToUshort());
}
```

Figura 29 - Escrita de um *Int* ou *Word* com a Biblioteca *S7.Net Plus*

Fonte: Autoria própria

Na figura 30 observa-se a escrita de uma variável do tipo *Real* no controlador, fazendo-se necessário a utilização da conversão pelo método *ConvertToUint()* da linguagem *C#,* a partir do endereço estabelecido na variável *db,* escrito, por exemplo, no formato *"DB1003.DBD12"*.

```
// escreve uma variavel double (Real) na conexão plc na db endereçada
4 referências
public static void WriteSingleVariable(Plc plc, double real, string db)
{
    plc.Write(db, real.ConvertToUInt());
}
```

Figura 30 - Escrita de um *Real* com a Biblioteca *S7.Net Plus*
Fonte: Autoria própria

Na figura 31 observa-se a escrita de uma variável do tipo *DInt* ou *DWord* no controlador, fazendo-se necessário a utilização da conversão pelo método *ConvertToUint()* da linguagem *C#,* a partir do endereço estabelecido na variável *db,* escrito, por exemplo, no formato *"DB1003.DBD40"*.

```
// escreve uma variavel int (DInt, DWord) na conexão plc na db endereçada
3 referências
public static void WriteSingleVariable(Plc plc, int valor, string db)
{
    plc.Write(db, valor.ConvertToUInt());
}
```

Figura 31 - Escrita de um *DInt* ou *DWord* com a Biblioteca *S7.Net Plus*
Fonte: Autoria própria

O quadro 4 indica a relação entre os diferentes tipos de dados existem nas linguagens *C#* e *S7-200* utilizados no desenvolvimento deste projeto.

Quadro 4 – Relação de Tipos de Dados entre C# e S7-200
Fonte: Autoria própria

C#	***S7-200***
Bool	*Bool*
short	*Int, Word*
Int	*DInt, Dword*
Double	*Real*

4.6 **SOFTWARE**

Para o desenvolvimento desse software foi realizado um estudo de campo, situando-se no meio fabril e analisando as necessidades dos modelos de veículos do tipo *palleteira*. Após o estudo, conseguiu-se um exemplar do veículo para o desenvolvimento do software e realização dos testes. Posteriormente ao desenvolvimento, cada uma das tarefas desempenhadas pelo software foi testada no meio fabril e executada de maneira exaustiva para assegurar a eficiência e qualidade do projeto. As imagens que seguem do software mostram uma planta baixa real de um meio fabril onde foram realizados esses testes.

O software foi dividido em dois módulos, para representar os dois níveis de acesso necessários a operação do sistema. Estes níveis foram implementados a fim de distinguir as posições de operador e supervisor do sistema, o que corrobora com os princípios de um sistema de controle supervisório [7]. No modo de desenvolvedor ou supervisor, foi implementado características pertinentes à configuração do sistema e conectividade com os veículos, no qual apenas pessoal autorizado e devidamente treinado teria acesso. No modo de usuário ou operador, uma interface que represente devidamente a interação homem-máquina, com a intenção de promover a visualização geral dos *AGVs* no chão de fábrica, das rotas cadastradas no sistema, das informações detalhadas sobre cada veículo e das rotas sendo realizadas por cada um destes.

A seguir pode-se visualizar figuras referentes a cada um destes módulos com as respectivas funcionalidades mencionadas. Primeiramente serão expostas as telas referentes ao modo de desenvolvedor.

Na figura 32 observa-se a janela de conectividade com os *AGVs*, sendo realizada através do *IP* na rede de cada veículo. Nesta tela é possível efetuar a conexão com os AGVs através de uma rede sem fio ou por cabo, ou selecionar o modo de desenvolvedor e executar o simulador de AGVs para realizar testes e simulações no programa. Ao clicar em *Listar*, todos os *IPs* dos *AGVs* conectados a rede serão listados na tabela da esquerda. Existe a opção de se conectar a todos da lista ao se clicar em *Conectar* ou efetuar dois cliques em cima de cada *IP* desejado. Os veículos conectados aparecerão na tabela localizada ao lado direito da tela. Para

desconectar, basta realizar o mesmo processo, clicando agora ou em *Desconectar* ou selecionando o *IP* desejado para encerrar a conexão na tabela da direita. Existe também a possibilidade de se restringir a faixa de *IPs* desejados para listagem, em caso de a rede ser compartilhada com mais máquinas ou usuários, portanto, basta inserir nas caixas de textos *Min IP* e *Max IP* referentes a faixa mínima e máxima pretendida. Faz-se necessário também, nesta tela, configurar a posição inicial de partida do veículo em metros, com referência ao canto inferior direito da planta da fábrica a ser exibida na interface do programa, através das caixas de texto *Pos Xo* e *Pos Yo.*

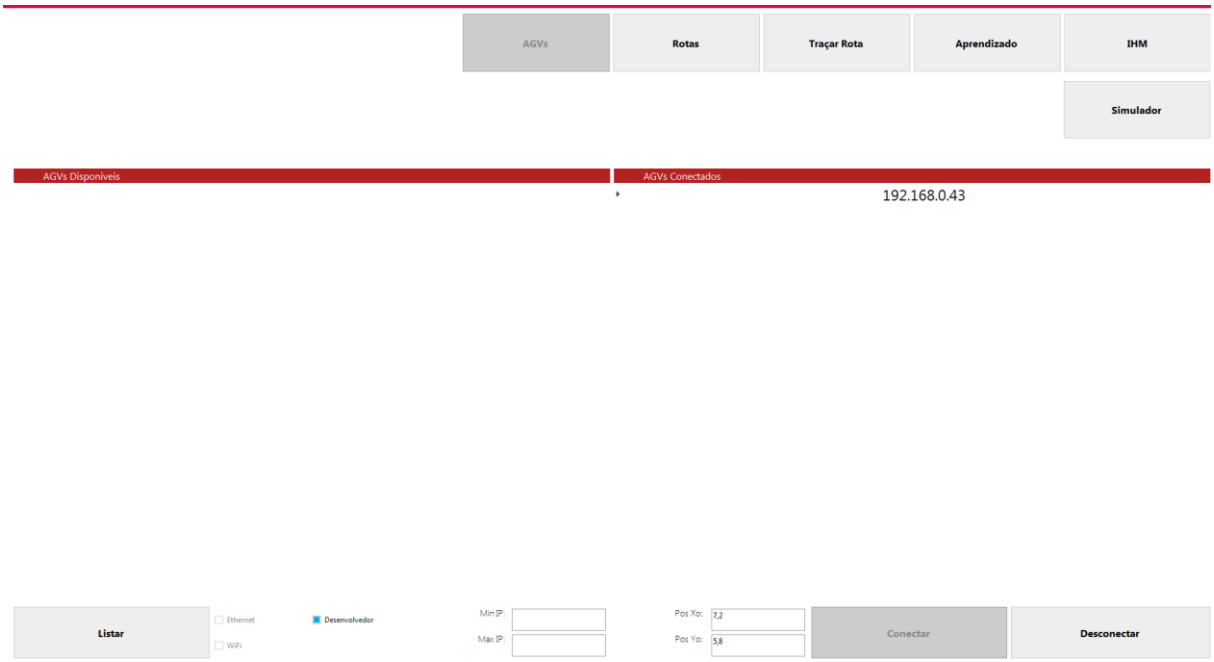


Figura 32 - Interface de Conectividade do Software
Fonte: Autoria própria

É possível também executar um simulador de um veículo conectado ao sistema. Para isto, faz-se necessário criar um projeto no software de programação de controladores Siemens chamado *Simatic Step 7* [14], contendo um *CLP* modelo ET200 no projeto e as duas *DBs* mostradas no capítulo deste documento sobre *Armazenamento de Dados no CLP.* Utiliza-se, para a correta execução do simulador, na *DB* de características dos pontos a numeração *DB1003* e na *DB* de troca de dados a numeração *DB1005.* Feito isto, execute o simulador de *CLP* do

programa *Simatic Step 7* e em seguida o simulador de *AGVs* desenvolvido. É preciso também utilizar um programa auxiliar chamado *NetToPLCSim*, disponível em [7] e monstrado na figura 34, para manter uma conexão entre o software de simulação de PLC da Siemens *PLCSIM* com o simulador de *AGVs* por uma interface de rede TCP/IP, configurando de acordo com os *IPs* do simulador Siemens e do computador hospedeiro. No simulador de *AGVs* basta carregar uma rota obtida através do programa desenvolvido e iniciar a simulação.

Na figura 33 apresenta-se a janela do simulador de *CLP Siemens Simatic Step 7* e na figura 35 a tela do simulador de *AGVs* desenvolvido neste projeto.

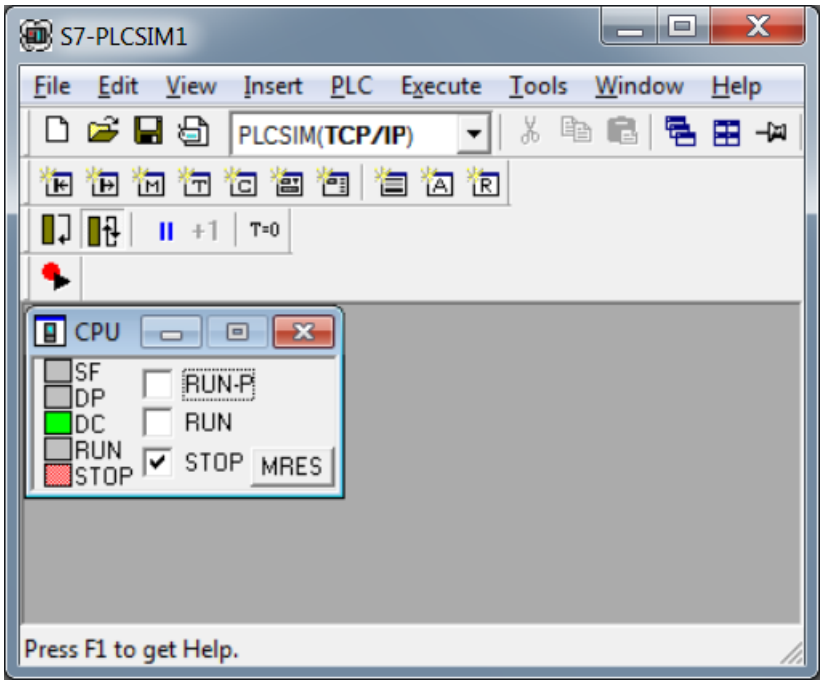


Figura 33 – Simulador de *CLP* Siemens Simatic Step 7
Fonte: Autoria própria

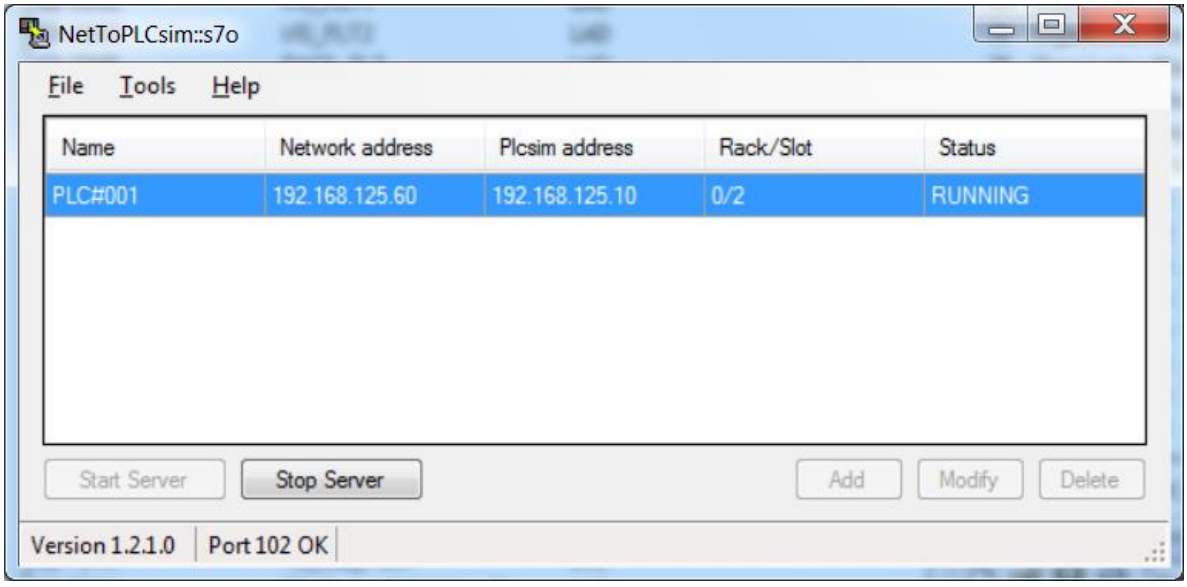


Figura 34 – *NetToPLCSim* Software de Interface entre PLCSIM e Simulador de AGVs
Fonte: Autoria própria

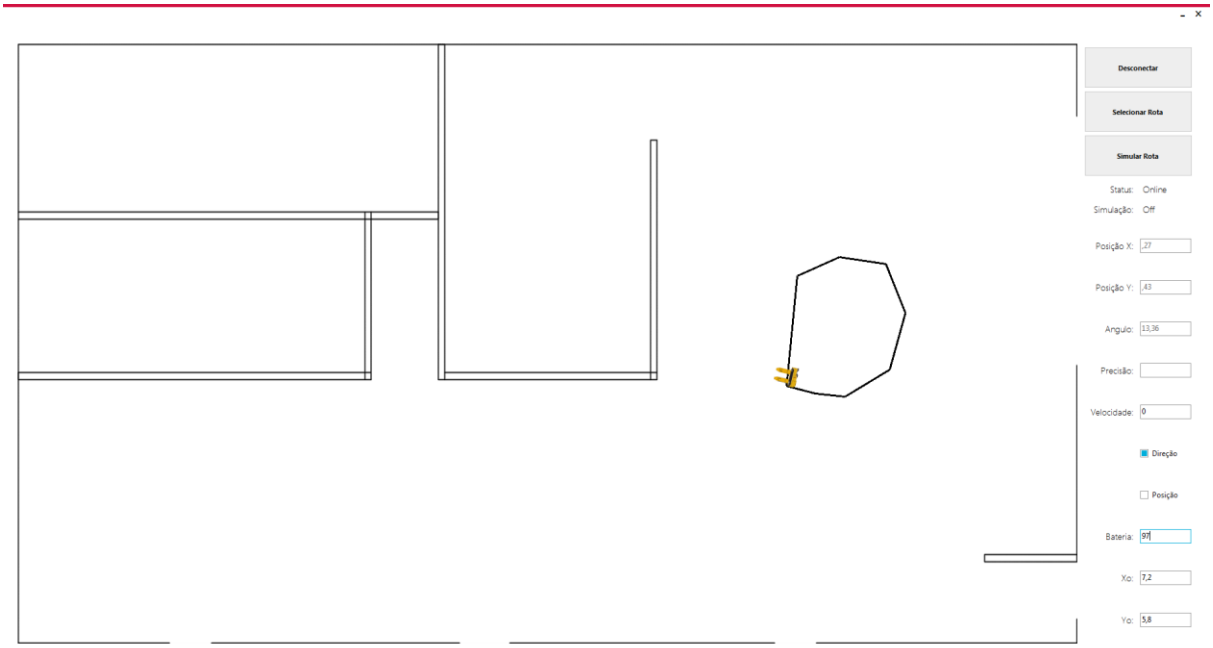


Figura 35 – Simulador de *AGVs*
Fonte: Autoria própria

Na figura 36 apresenta-se a interface de edição e envio de rotas cadastradas para o veículo, bem como a possibilidade de exportá-las e importá-las de um arquivo com extensão *csv.* Essas rotas ao serem enviadas para o *AGV* são gravadas na *DB1003* de coordenadas e características de cada ponto, mostrada no capítulo sobre *Armazenamento de Dados no CLP.*

O software propõe gerir as rotas disponíveis e enviá-las aos veículos, entretanto, a execução do percurso e as tomadas de decisões pelo chão de fábrica é encarregado ao software desenvolvido no *CLP* do veículo. Portanto, para que o veículo circule de fato pela rota enviada, é necessário que haja uma programação do controlador presente no veículo. Essa programação pode ser realizada através do software *Simatic Step 7* da *Siemens* nas linguagens de programação *ladder, scl, stl,* e foge ao escopo desse projeto.

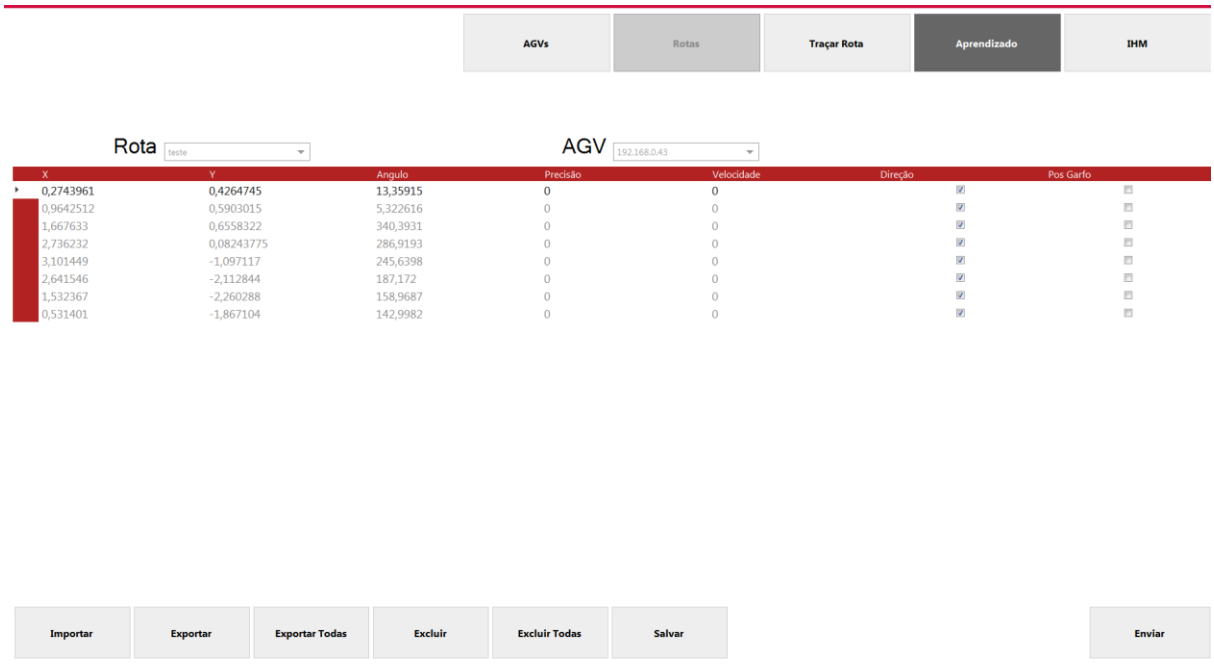


Figura 36 - Interface de Rotas
Fonte: Autoria própria

Na figura 37 expõe-se a possibilidade de traçar as rotas a serem enviadas para o veículo de forma manual, se desenhando diretamente na planta da fábrica. O traçado da rota se faz clicando com o botão marcando o ponto e arrastando e liberando o clique ao obter o ângulo desejado. Junto de cada ponto é possível verificar as coordenadas cartesianas em metros e o ângulo obtido.

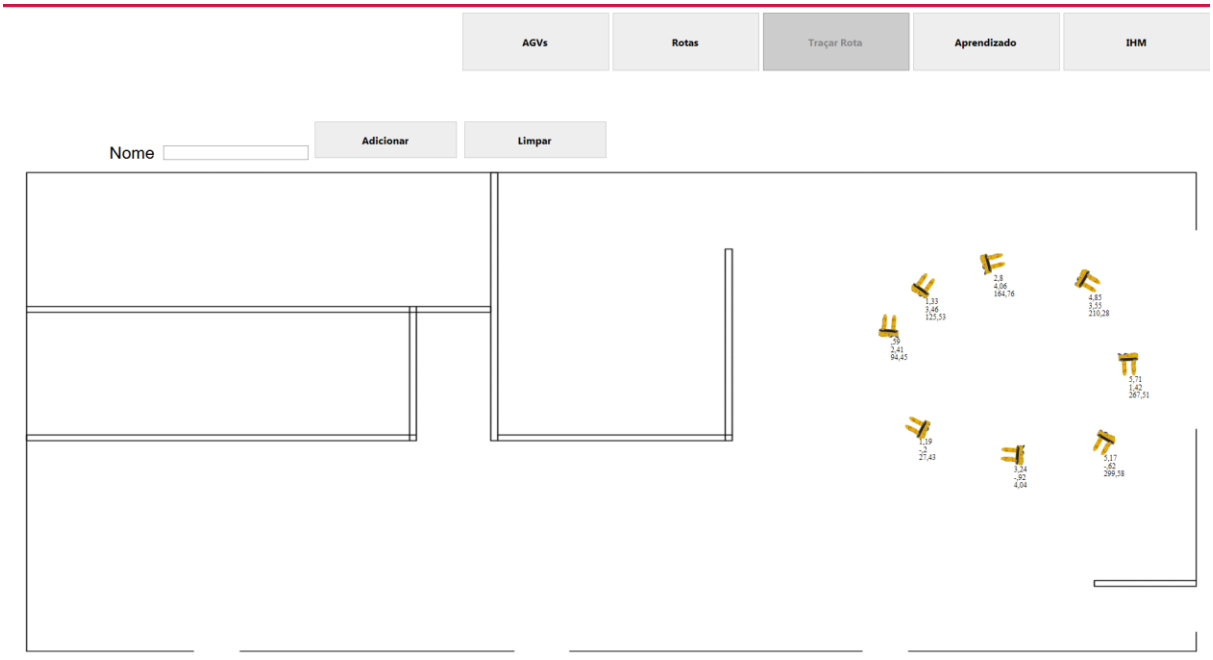


Figura 37 - Interface Traçar Rota
Fonte: Autoria própria

A figura 38 exemplifica o modo de aprendizado do sistema, onde ao invés de se desenhar manualmente os pontos na planta fabril pela interface supracitada, os pontos são gravados diretamente com o *AGV* no chão de fábrica e posteriormente lidos pelo software a partir da *DB1003* das características de cada ponto, mostrada no capítulo sobre *Armazenamento de Dados no CLP* deste documento.

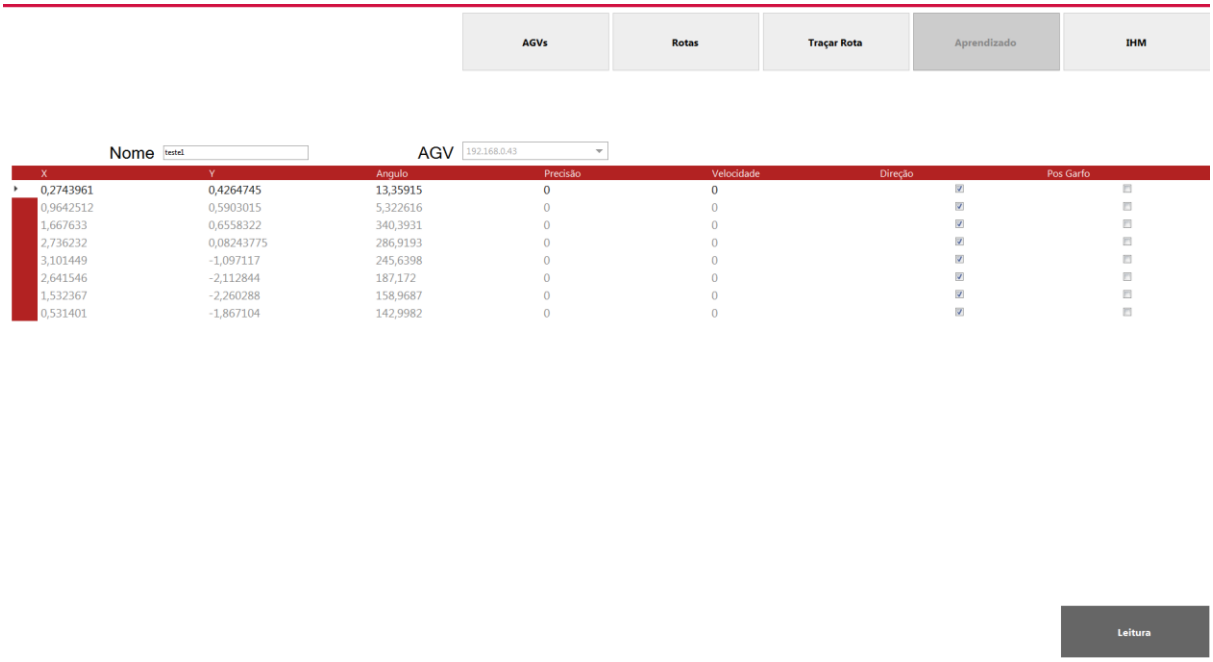


Figura 38 - Interface de Aprendizado
Fonte: Autoria própria

As imagens que seguem são relativas ao modo de usuário do software. Na figura 39 observa-se o monitoramento em tempo real no chão de fábrica da posição dos veículos conectados ao sistema.

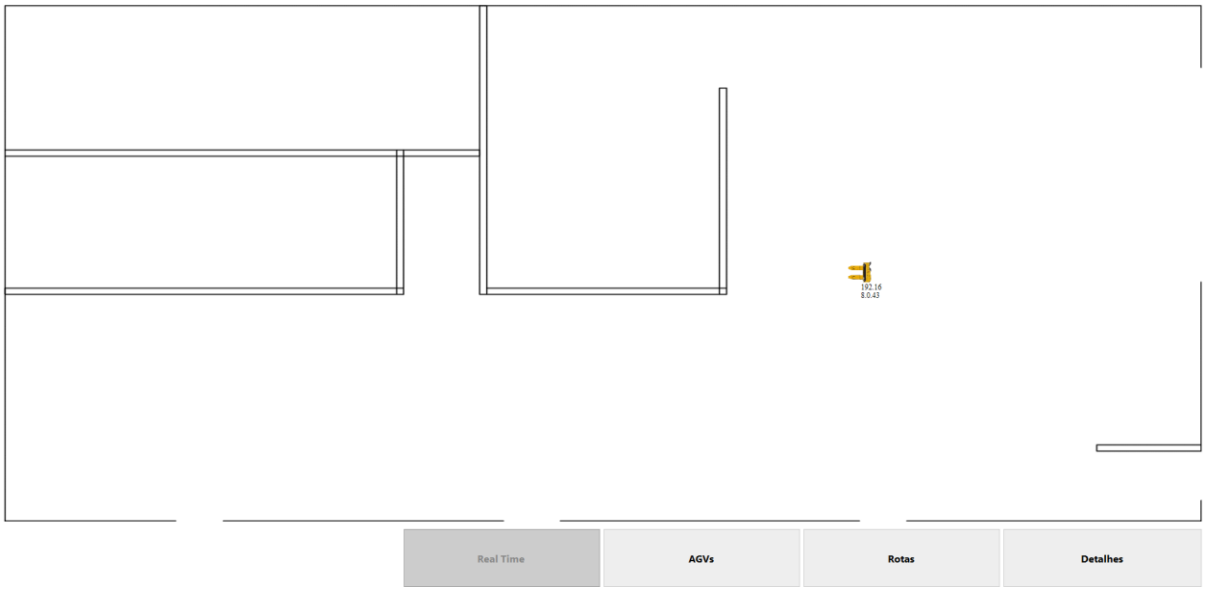


Figura 39 - Interface de Monitoramento de Tempo Real
Fonte: Autoria própria

A figura 40 exemplifica o monitoramento em tempo real dos veículos conectados ao sistema mostrando juntamente as rotas que cada um dos veículos deve seguir. É possível, nesta interface, visualizar um único veículo desempenhando a sua rota programada ou visualizar ao mesmo tempo todos os veículos cadastrados no sistema realizando suas respectivas rotas clicando-se no cabeçalho da tabela contendo o *IP* dos veículos disponíveis. Os veículos e as rotas serão exibidos todos ao mesmo tempo na planta da fábrica, cada um com uma cor de rota diferente.

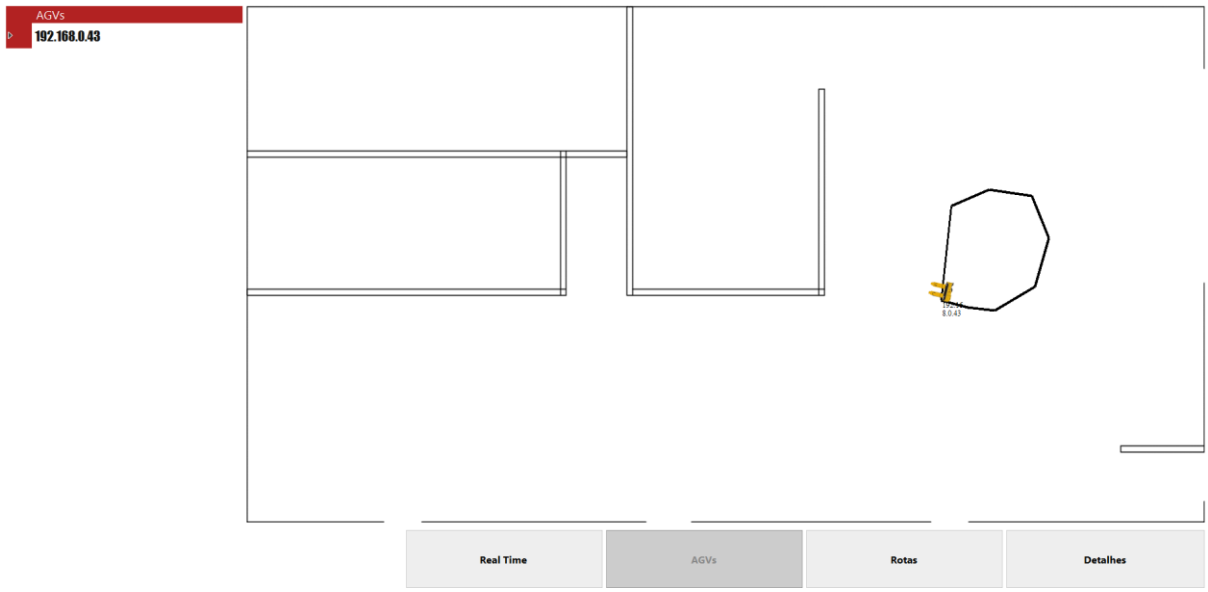


Figura 40 - Interface *AGVs* e Suas Rotas
Fonte: Autoria própria

Na figura 41 se encontra a interface expondo as rotas cadastradas no sistema. É possível, nesta interface, visualizar uma única rota selecionando a de referência ao nome desejado ou ao mesmo tempo todas as rotas cadastradas no sistema clicando-se no cabeçalho da tabela contendo o nome das rotas disponíveis. As rotas serão exibidas todas ao mesmo tempo na planta da fábrica com cores diferentes.

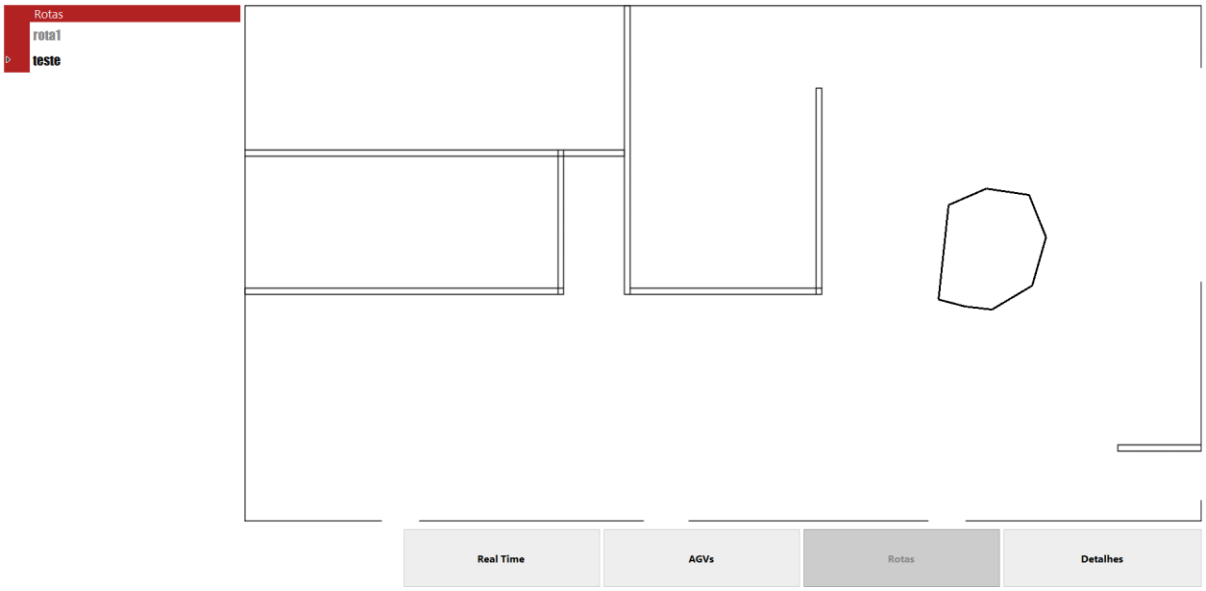


Figura 41 - Interface de Rotas Cadastradas
Fonte: Autoria própria

A figura 42 expõe os dados de maneira mais objetiva e textual das informações correntes de cada veículo conectado ao sistema, obtendo as informações do veículo a partir da *DB1005* mostrada no capítulo sobre *Armazenamento de Dados no CLP* deste documento.

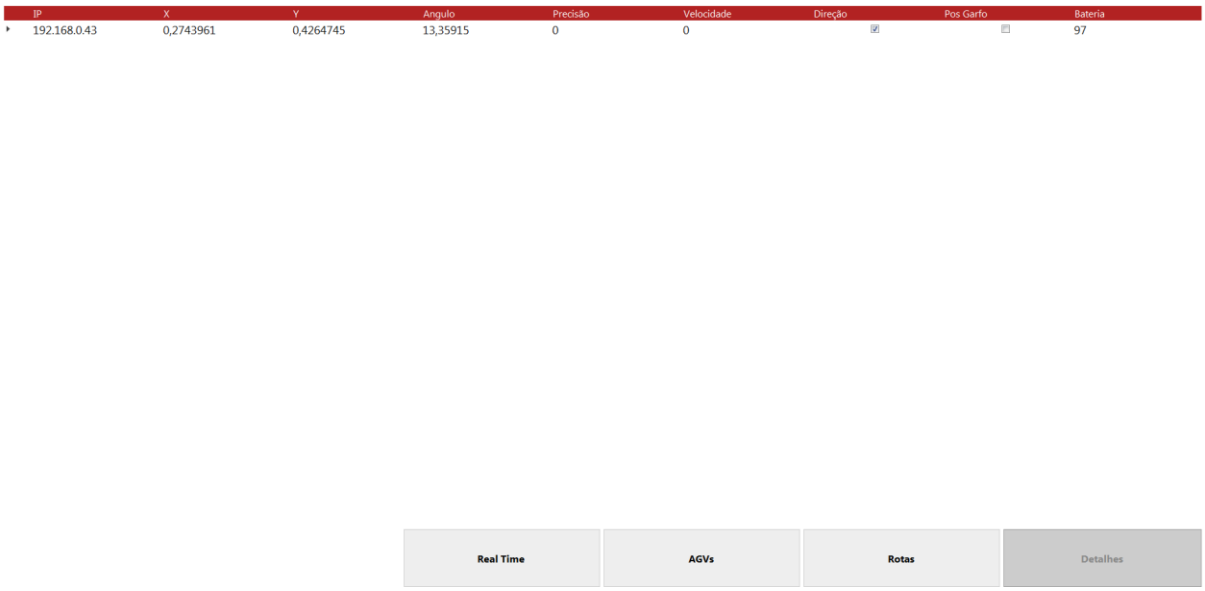


Figura 42 - Interface de Detalhes dos Veículos
Fonte: Autoria própria

5 RESULTADOS E DISCUSSÕES

5.1 **RESULTADOS**

Os resultados atingidos com o desenvolvimento deste software foram condizentes com os objetivos estipulados no escopo deste projeto. Aprendeu-se muito com a linguagem de desenvolvimento de software C#, sendo assim possível aplicar a biblioteca de comunicação de forma clara e efetiva.

O desenvolvimento deste software possibilitou realizar uma maior interação homem-máquina relacionado a veículos autoguiados, permitindo a visualização gráfica e textual em tempo real dos veículos realizando seus serviços no meio fabril, criação, edição, importação, exportação e envio de rotas para os veículos. O sistema possui ainda dois níveis de acesso, um para diagnóstico rápido das características dos veículos pelos operadores que trabalham no chão de fábrica, outro para a realização de alterações nas configurações do AGV que o sistema possibilita.

5.2 **DIFICULDADES ENCONTRADAS**

As principais dificuldades encontradas foram relacionadas à conectividade com os veículos, uma vez que se utiliza de biblioteca pronta e assume-se a confiabilidade desta. Existe a necessidade do monitoramento constante dos veículos conectados ao sistema, confirmação esta obtida através de uma função na biblioteca de conectividade utilizada que retorna a situação da conexão com o *AGV* solicitado. Este número de requisições poderia gerar uma grande demanda de processamento e troca de dados pela rede que seria capaz de acarretar uma falha se um alto volume de veículos fosse conectado simultaneamente.

Outra dificuldade ocorreu em decorrência do gerenciamento de memória. Devido ao grande número de requisições feitas solicitando constantemente as posições dos veículos e a sua situação, e o grande número de atualizações referentes a imagem exibida na tela da interface, ocorreu uma grande demanda por

memória *ram* por parte do programa. Chegou-se a necessitar até 1,5GB de memória para alocar todas as demandas necessárias. Foi possível solucionar este problema posteriormente através de uma classe do *C#* chamada *GC*, *garbage collector*, responsável por verificar os objetos não utilizados mais pelo programa e limpá-los da memória. Isto é feito automaticamente de tempos em tempos pela própria máquina do *Framework*, entretanto, devido ao grande número de requisições pela rede e atualizações de imagem na interface da planta, ocorreu essa demanda por limpar previamente o que não estava mais sendo usado. Com isto, possibilitou-se uma redução para 30MB de utilização de memória *ram* em média.

Devido a indisponibilidade de vários veículos para testes, houve a necessidade do desenvolvimento de um simulador de AGV para que pudesse ser realizada a validação do software, mesmo quando o veículo não estivesse disponível fisicamente no meio fabril.

6 CONCLUSÃO

Este documento apresentou uma visão geral do software proposto para desenvolvimento de um sistema de gerenciamento de AGVS, como requisito parcial da disciplina de trabalho de conclusão de curso do curso de Engenharia de Computação.

Objetivou-se, através da metodologia de desenvolvimento ágil de software, a rápida obtenção dos conhecimentos necessários para a execução completa deste projeto e prontamente colocar em prática através do desenvolvimento do código em si. Observou-se não somente o cumprimento do calendário, como também, o adiantamento de algumas tarefas e a sobreposição de outras. A entrega prévia e a melhoria contínua do produto podem ser verificadas na etapa de exposição do software e estrutura deste documento, onde etapas previamente planejadas no calendário do projeto foram adiantadas mediantes ao avanço obtido durante o seu desenvolvimento. A resposta rápida e flexível à mudança são características vitais, onde se observou por diversas vezes a necessidade de agir rapidamente a cada nova demanda obtida, como por exemplo, nos casos supracitados na etapa de dificuldades encontradas onde se necessitou reduzir o consumo de memória *ram* utilizada pelo software, e o desenvolvimento do software de simulação de AGV no caso de o veículo não estar disponível para testes e apresentação no meio fabril.

Mediante os fatos expostos neste documento e aos relatos evidenciados, o sistema foi devidamente implementado de acordo com suas projeções e expectativas, e alcançou os objetivos requisitados no escopo deste projeto.

7 REFERÊNCIAS

[1] M. A. Ribeiro, Automação industrial, Salvador-BA: Tek Treinamento e Consultoria LTDA, 1999.

[2] Microsoft, “Visual Studio,” Microsoft, [Online]. Available: visualstudio.com. [Acesso em 04 06 2017].

[3] A. Daneels e S. Wayne, “What is SCADA,” *International Conference on Accelerator and Large Experimental Physics Control Systems,* 1999.

[4] P. Contact, “Controladores,” [Online]. Available: https://www.phoenixcontact.com/online/portal/br?1dmy&urile=wcm%3apath%3a/brpt/web/main/products/subcategory_pages/Controllers_P-21/cb588613-fd60-41e7-a3e4-61c93d94155e. [Acesso em 05 06 2017].

[5] D. Cohen, M. Lindvall e P. Costa, “Agile Software Development”.*DACS State-of-the-Art.*

[6] W. Bolton, Programmable logic controllers, Newnes, 2015.

[7] D. Bailey e E. Wright, Practical SCADA for industry, Newnes, 2003.

[8] ABB, “PLC Automation,” [Online]. Available: http://new.abb.com/plc. [Acesso em 05 06 2017].

[9] Allen-Bradley, “Rockwell,” [Online]. Available: http://ab.rockwellautomation.com/allenbradley_pt/index.page. [Acesso em 05 06 2017].

[10] B&R, “PCs industriais,” [Online]. Available: https://www.br-automation.com/pt-br/products/pcs-industriais/. [Acesso em 05 06 2017].

[11] A Network Interface to PLCSim, “NetToPLCSim,” [Online]. Available: nettoplcsim.sourceforge.net. [Acesso em 04 06 2017].

[12] SICK, “Laser Scanner NAV350,” [Online]. Available: https://www.sick.com/de/en/product-portfolio/detection-and-ranging-solutions/2d-lidar-sensors/nav3xx/c/g91916. [Acesso em 05 06 2017].

[13] Siemens, “ET200,” [Online]. Available: http://w3.siemens.com.br/automation/br/pt/automacao-e-controle/controladores-simatic/et200/et200s/pages/et200s.aspx. [Acesso em 05 06 2017].

[14] Siemens, “CLP,” [Online]. Available: http://w3.siemens.com.br/automation/br/pt/automacao-e-controle/automacao-industrial/simatic-plc/s7-cm/pages/default.aspx. [Acesso em 05 06 2017].

[15] s7.Net Plus, “A .NET Library for Siemens S7 Connectivity,” [Online]. Available: github.com/killnine/s7netplus.

[16] Microsoft, “.NET Framework,” [Online]. Available: https://www.microsoft.com/net/download/framework.