MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

SÉRGIO GABRIEL DOS SANTOS DIAS

ENGENHARIA REVERSA: MODELAGEM DE UMA PLATAFORMA DE MANUFATURA INTEGRADA POR COMPUTADOR

RIO DE JANEIRO
2021

SÉRGIO GABRIEL DOS SANTOS DIAS

ENGENHARIA REVERSA: MODELAGEM DE UMA PLATAFORMA DE MANUFATURA INTEGRADA POR COMPUTADOR

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia Eletrônica do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador(es): Antonio Eduardo Carrilho da Cunha, Dr. Eng

Rio de Janeiro

2021

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

Santos Dias, Sérgio Gabriel dos.
Engenharia Reversa: Modelagem de uma Plataforma de Manufatura Integrada por Computador / Sérgio Gabriel dos Santos Dias. – Rio de Janeiro, 2021.
28 f.

Orientador(es): Antonio Eduardo Carrilho da Cunha.

Projeto de Final de Curso (graduação) – Instituto Militar de Engenharia, Engenharia Eletrônica, 2021.

1. Mecatrônica. 2. Engenharia de Sistemas. 3. Engenharia Reversa. 4. FS-100. 5. ScorBase. i. da Cunha, Antonio Eduardo Carrilho (orient.) ii. Título

SÉRGIO GABRIEL DOS SANTOS DIAS

Engenharia Reversa: Modelagem de uma Plataforma de Manufatura Integrada por Computador

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia Eletrônica do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador(es): Antonio Eduardo Carrilho da Cunha.

Aprovado em Rio de Janeiro, 21 de Outubro de 2021, pela seguinte banca examinadora:

Cel. **Antonio Eduardo Carrilho da Cunha** - Dr. Eng

Maj. **Raquel Stella da Silva de Aguiar** - Dr. ISAE

Maj. **Erick Menezes Moreira** - Dr. IME

Sr. **Igor Cohen Calixto** - MsC IME

Rio de Janeiro

2021

Ao Instituto Militar de Engenharia, alicerce da nossa formação e conhecimento.

AGRADECIMENTOS

Os agradecimentos principais são direcionados à minha família e amigos, que juntamente com todos os meus professores me deram o suporte necessário durante esse projeto e todos os meus anos de estudos no IME.

Um agradecimento especial ao orientador Cel Antonio Eduardo Carrilho da Cunha, Maj Raquel Stella da Silva de Aguiar e Maj Erick Menezes Moreira e ao aluno de doutorado Igor Calixto Cohen, por todo o auxílio e disponibilidade.

"Todos podem ver as táticas de minhas conquistas, mas ninguém consegue discernir a estratégia que gerou as vitórias."

Sun Tzu

RESUMO

Com o intuito de modernizar o ensino do Instituto Militar de Engenharia, foi adquirida uma plataforma de Manufatura Integrada por Computador, do inglês CIM - *Computer Integrated Manufacture*, que simula uma linha de produção dentro dos princípios e tecnologias associados à indústria 4.0.

A plataforma MecatrIME representa um Sistema Ciberfísico, um do pilares da Indústria 4.0, no qual o sistema físico é controlado por meio de atuadores e sensores utilizando um algoritmo, com todos os componentes conectados por meio de uma rede de comunicação. No caso da plataforma MecatrIME os algoritmos de controle se encontram organizados em camadas de software, sendo de grande importância a verificação da validade do funcionamento dos algoritmos.

O presente projeto trata de uma proposta de extração automática de modelos de forma automática de autômatos a partir das rotinas de controle.

ABSTRACT

In order to modernize the teaching in IME, it was acquired a CIM - Computer Integrates Manufacture platform, which simulates a production line within the principles and technologies associated with industry 4.0.

The MecatrIME platform represents a Cyber-Physical System, one of the pillars of Industry 4.0, in which the physical system is controlled through actuators and sensors using an algorithm, with all components connected through a communication network. In the case of the MecatrIME platform, the control algorithms are organized in software layers, with the verification of the validity of the algorithms functioning being of great importance.

The present project deals with a proposal for the automatic extraction of models automatically from automata from control routines.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CIM	*Computer Integrated Manufacture*, ou Manufatura Integrada por Computador
bdd	*Block Definition Diagram*, ou Diagrama de Definição de Blocos
EAP	Estrutura Analítica do Projeto

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

O laboratório de mecatrônica do IME por meio da plataforma MecatrIME possui a capacidade de operação de uma planta industrial de pequeno porte dentro dos conceitos de Indústria 4.0, contendo todo o hardware e software correspondente a esse tipo de aplicação

A Quarta Revolução Industrial, também chamada de Indústria 4.0, engloba os conceitos de inteligência artificial, robótica, internet das coisas e computação em nuvem, com o objetivo de melhorar processos e aumentar a produtividade por meio da digitalização das atividades industriais. Essa capacidade é associada à utilização de atuadores e sensores em estações de trabalho, por meio de diversas camadas de software e interligado por uma rede de comunicação, configurando dessa forma um Sistema Ciberfísico.

A plataforma MecatrIME por possuir a capacidade de uma planta industrial de produção de pequena escala, conta com seis estações de trabalho composto por braços robóticos e algumas funções específicas, uma esteira para fazer a movimentação física entre as estações e uma estação de gerenciamento, como visto na Figura 1.

Figura 1 – Organização da plataforma MecatrIME a partir da estação de gerenciamento

A organização da plataforma MecatrIME encontra-se detalhada na Figura 2, identificando cada uma das 6 estações de trabalho. Da figura temos P1 como a estação de armazenamento, P2 identifica a estação de torneamento, P3 a estação de usinagem, P4 a estação de soldagem, P5 a estação de metrologia, P6 a estação de montagem e inspeção visual, P7 a instrução de gravação a laser. A estação P8 corresponde ao controle da esteira de transporte e o P9 corresponde ao *Manager* que faz o gerenciamento de todo o sistema.

A complexidade do sistema exige então o conhecimento de conceitos de Engenharia

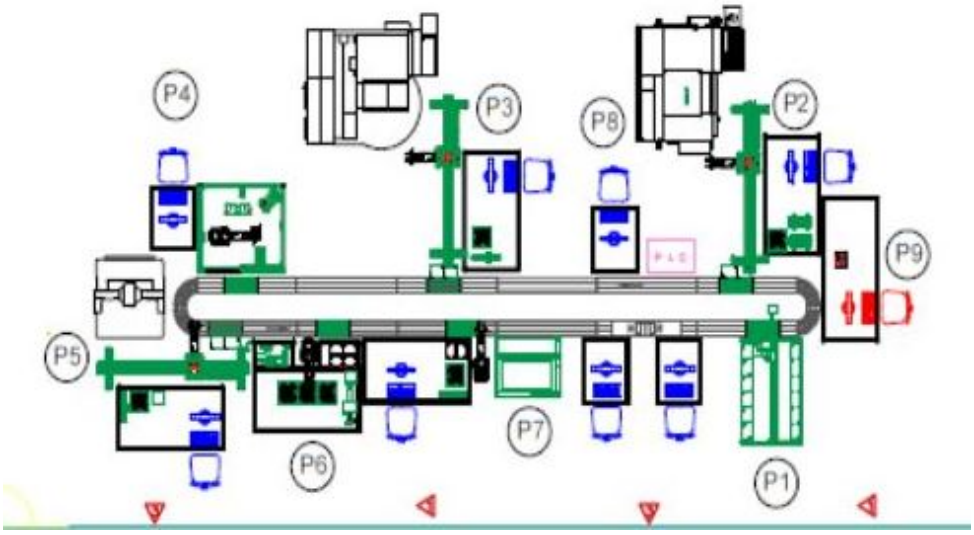

Figura 2 – Organização da planta MecatrIME identificada por estações

de Sistemas, que representam a organização e a relação entre os componentes físicos e programas computacionais, bem como a relação entre as estações e o gerente.

A relação entre os objetos físicos atuadores, como a esteira transportadora, os braços robóticos, o robô cartesiano e o torno por exemplo, e as estações virtuais com seus codigos de operação é descrita por meio de um diagrama de blocos conhecido como diagrama ciberfísico.

Cada uma das estações de trabalho está interligada para uma produção completa e integrada o *Manager* deve enviar comandos coordenados para cada uma das posições com suas devidas atividades. Esses comandos são subdivididos em camadas de código, sendo composto por rotinas de ScorBase e jobs do controlador FS-100. As rotinas de ScorBase contém os comandos necessários para a execução da atividade prevista na estação, utilizando-se dos jobs do FS-100 para os controles dos equipamentos envolvidos.

O conhecimento total do sistema requer o conhecimento das linguagens de programação com o intuito de alterar o código para a exploração completa das funcionalidades e alteração de parâmetros, conhecimento até então não dominado devido à falta de passagem completa do conhecimento por parte da empresa fornecedora.

Dessa forma, este trabalho apresenta uma proposta de extração automática de modelos na forma de autômatos a partir das rotinas de controle numa das camadas de software da plataforma MecatrIME.

1.2 Objetivo

Com base nos requisitos levantados de ferramenta de entrada e saída de autômatos, o escopo do projeto foi:

- Criação de arquivos no formato .dot do *GraphViz* das rotinas em ScorBase do torno;
- Criação dos autômatos dos programas;
- Desenvolvimento de uma ferramenta que automatize a extração de autômatos de forma automática das rotinas de controle em ScorBase.

1.3 Metodologia

Primeiramente foi definido o escopo do programa que seria utilizado como alvo de pesquisa do projeto. Para isso foi delimitada uma das estações de trabalho com foco na sua operação, sendo escolhida a estação de torno. Essa estação utiliza uma morça para posicionar uma bloco de metal e por meio de cortes conformá-la como a peça que deve ser utilizada nos pontos seguintes da produção. Como se trata de um Sistema Ciberfísico, cada parte da estação de trabalho é associada a um conjunto de instruções do ScorBase, sendo necessário conhecer o funcionamento desde os braços mecânicos até o acionamento da abertura e fechamento da porta.

Inicialmente, por meio de diagrama de blocos e de estados e utilizando os conceitos de Engenharia de Sistemas, foram modelados os componentes tanto de hardware quanto de software. Com a identificação dos elementos de software foram construídos os modelos dos autômatos, com o enfoque na ferramenta de torno, das rotinas de ScorBase.

Com a criação da ferramenta foi possível automatizar a criação dos autômatos no formato GraphViz convertidos a partir de arquivos-texto do ScorBase, testando-a então com outras rotinas existentes na estação do torno. Por meio de inspeção desses arquivos da saída foi possível validar o modelo e corrigí-lo para possíveis melhorias e erros encontrados.

1.4 Guia a Leitura

O presente texto está organizado seguindo a sequência: o Capítulo 1 é uma introdução com a contextualização do projeto, objetivos e metodologia; o Capítulo 2 apresenta uma visão geral do problema; O Capítulo 3 apresenta a base conceitual do projeto; o Capítulo 4 apresenta as ferramentas principais utilizadas no projeto; o Capítulo 5 apresenta a solução de tradução de ScorBase para máquina de estados; o Capítulo 6 apresenta os resultados dos ensaios com os scripts do escopo e por fim o Capítulo 7 aborda a conclusão do projeto, com as contribuições e possíveis próximos passos do projeto.

2 VISÃO GERAL DO PROBLEMA E ESTRUTURAÇÃO DO PROJETO

2.1 Visão geral do projeto

O projeto visa a obtenção automática de modelos formais a partir do código da camada de software relativa ao ScorBase da plataforma MecatrIME. Essa extração automática tem como objetivo, primeiramente, a verificação e validaçãodo software entregue pelo fabricante da plataforma por meio da técnica de *Model Checking* ou Checagem de Modelos, da Ciência da Computação(**??**). Na fase seguinte, por meio das ferramentas da Ciência da Automação, como a Teoria de Controle de Sistema Supervisório para Sistemas a Eventos Discretos buscou-se aumentar a capacidade operacional da plataforma por aperfeiçoamento das rotinas de controle lógico(**??**).

Para atingir esse objetivo foi idealizada uma ferramenta de extração de autômatos de forma automática. Dessa forma, utilizando-se dos conceitos de parser(1), fazendo a quebra do texto em palavras chave e identificando comandos a partir de um universo conhecido é possível realizar a tradução do código das rotinas de ScorBase para GraphViz.

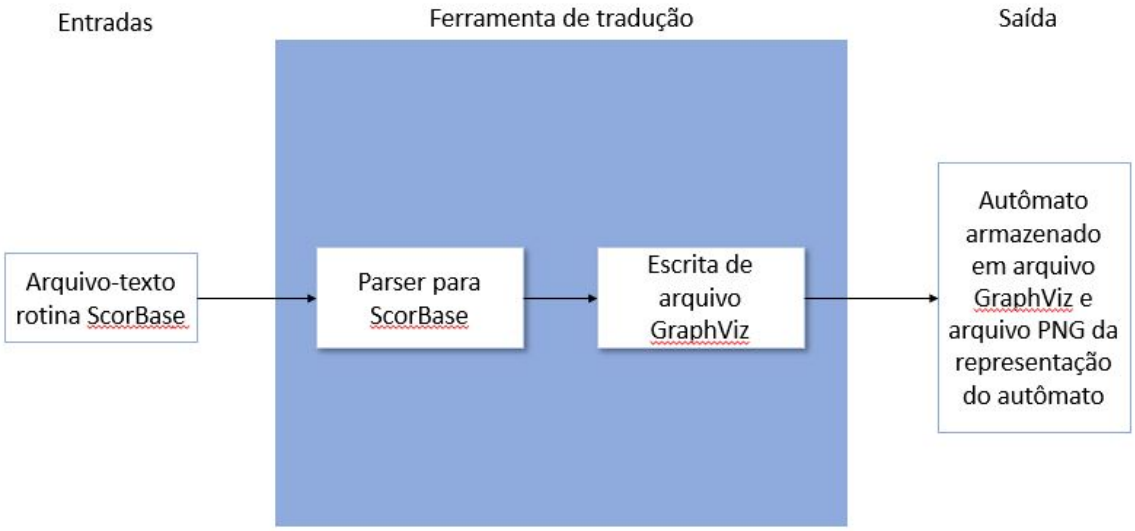

Figura 3 – Diagrama do funcionamento da ferramenta.

2.2 Estrutura do Projeto

Com base na demanda e nos prazos, o projeto foi estruturado conforme a Estrtura Analítica do Projeto (EAP)(2) da Figura 4.

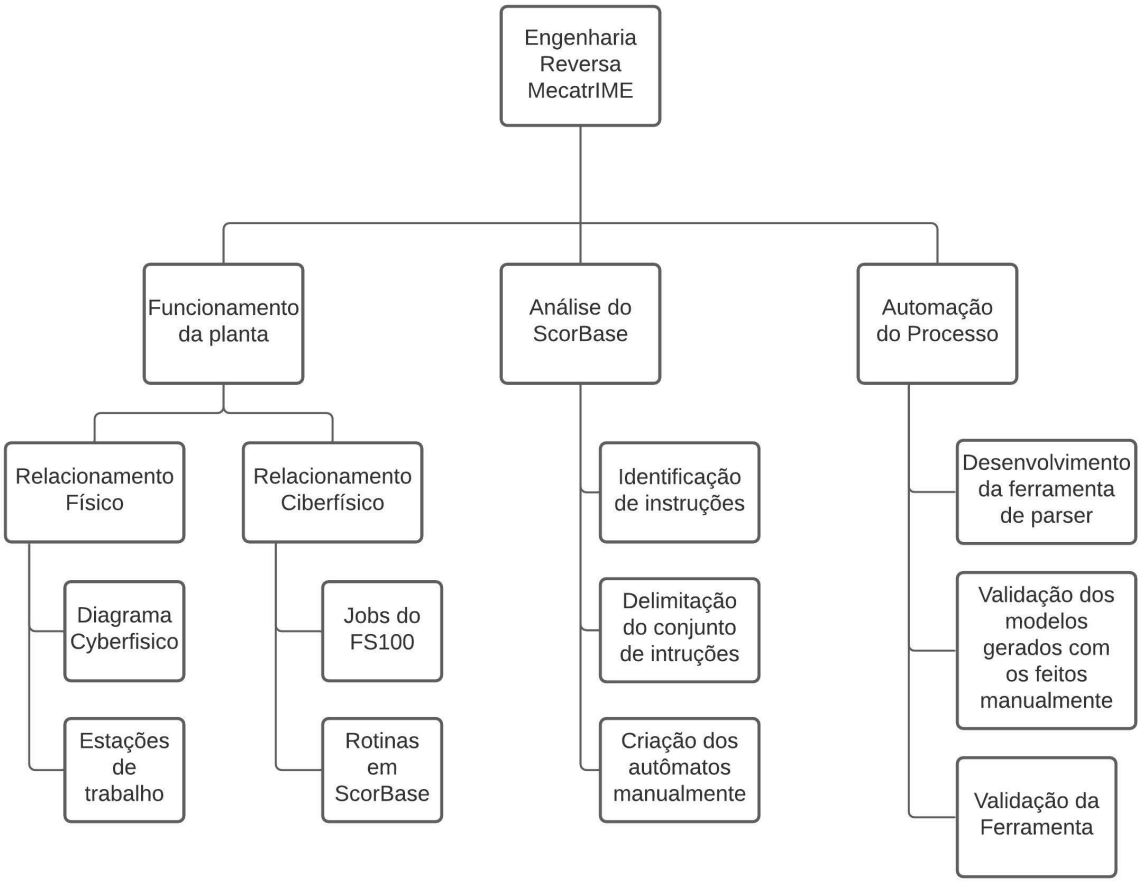

Figura 4 – Estrutura Analítica do Projeto.

Com a divisão de frentes do trabalho do projeto, foi possível organizar um cronograma conforme as atividades previstas. Na primeira fase do projeto o foco foi o entendimento do funcionamento do sistema, bem como as tecnologias envolvidas. A segundo etapa foi focada na análise dos arquivos-texto do ScorBase da estação de trabalho do torno e definição do escopo do código a ser trabalhado, juntamente com o desenvolvimento de autômatos de forma manual utilizando o *yEd*(3). Por fim, foi desenvolvida a ferramenta de tradução de código de ScorBase armazenando o autômato em um arquivo GraphViz e também uma representação gráfica do autômato para inspeção.

3 BASE CONCEITUAL

3.1 Organização da plataforma MecatrIME

A plataforma da MecatrIME, por ter a capacidade produtiva de uma planta industrial de pequeno porte apresenta uma grande complexidade e integração de partes físicas e mecânicas com níveis de software de controle, sendo assim um sistema ciberfísico. Com os conceitos de Engenharia de Sistemas, a plataforma pode ser representada por um diagrama de definição de blocos ou bdd, representando as conexões em níveis de hardware e software com suas interações.

Para a plataforma MecatrIME a organização em subsistemas de forma visual é organizada conforme a Figura 5. Essa estrutura é a base para o entendimento do relacionamento entre cada uma das partes, bem como a melhor forma de representação visual de cada uma dessas partes.

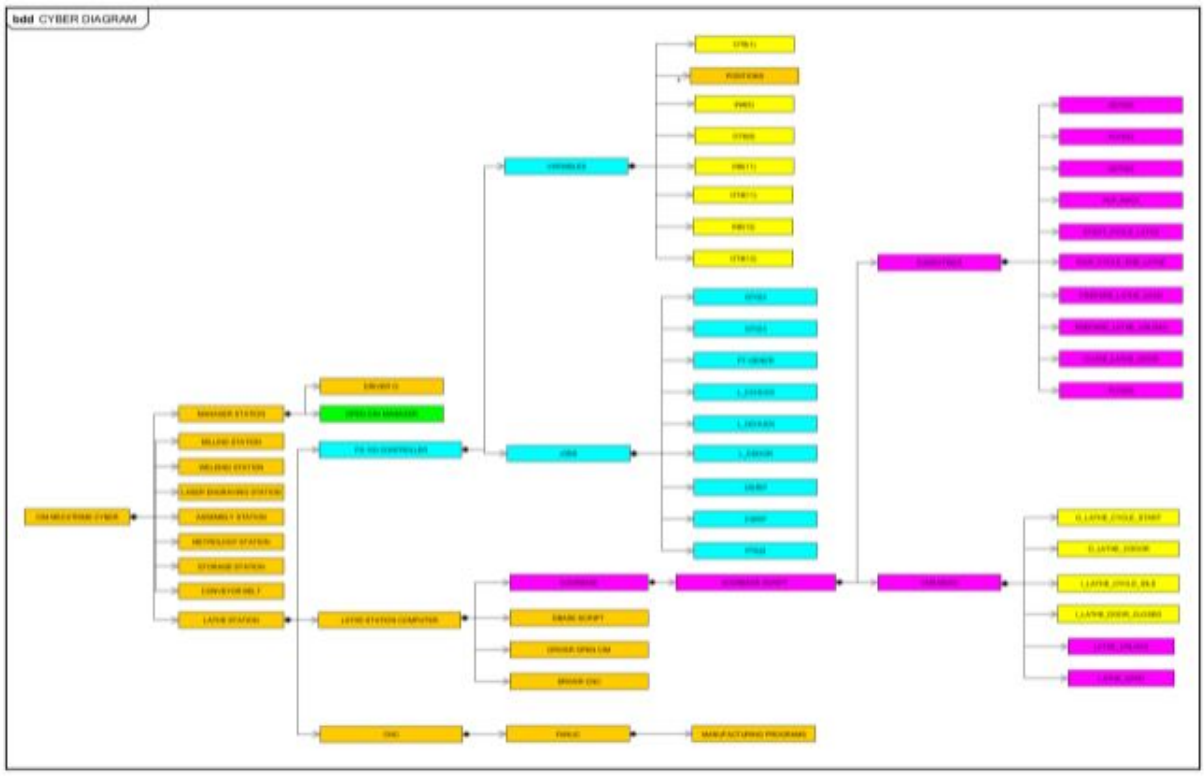

Figura 5 – Diagrama de Blocos cibernético da plataforma MecatrIME.

As conexões do sistema ciberfísico é a presente na Figura 6, no qual são identificadas quais estações de trabalho da plataforma são afetadas por rotinas da linguagem ScorBase e como essas estações interagem com o *Manager* e entre si. O esquema de cores identifica

as rotinas do ScorBase, em rosa, os jobs da FS100, em azul, e as partes mecânicas do sistema, em amarelo. Por meio do diagrama cyberfísico também fica evidente a relação entre o ScorBase e os jobs da FS100.

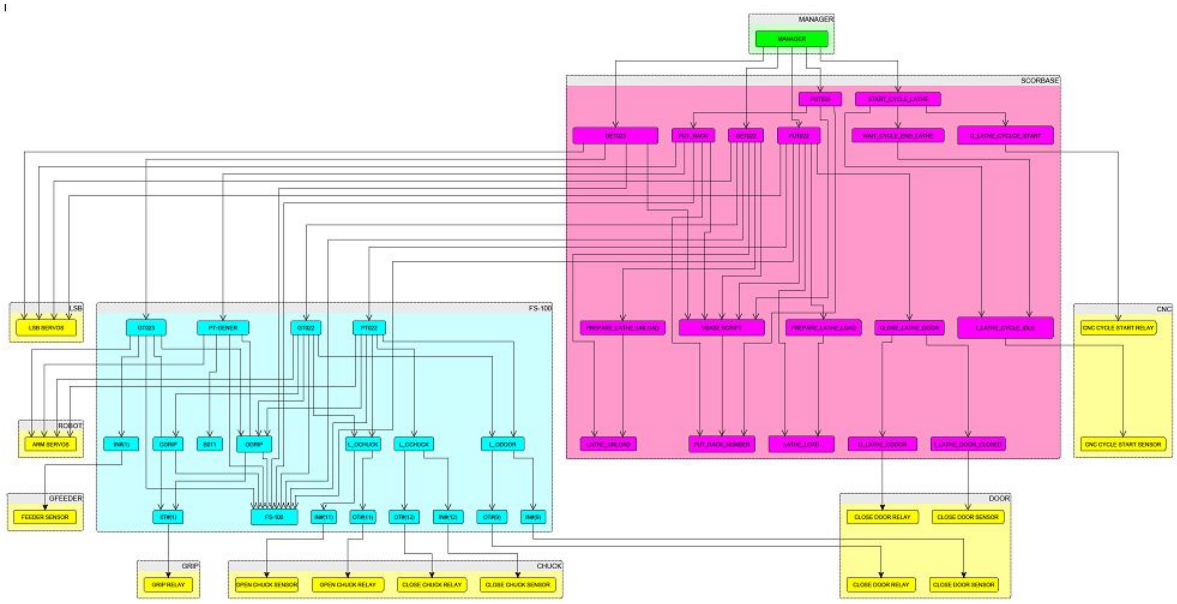

Figura 6 – Conexões no sistema ciberfísico.

3.2 ScorBase e FS100

Dentro da parte computacional o sistema possui um controlador FS100(4) aplicadas no robô Motomas MH5 F da fabricante Yaskawa, que possui dentro de suas vantagens um software aberto, permitindo novas implementações e alterações. O controlador representa os jobs em linguagem de mais baixo nível.

A linguagem utilizada em cada uma das estações, em nível mais alto, é a ScorBase da Intelitek(5). Essa linguagem é amplamente utilizada em aplicações de robótica. Juntamente com o FS100 compõem a parte computacional do sistema.

3.3 Autômatos

Na teoria da computação, autômatos são modos abstratos de máquinas que realizam ações baseadas em uma entrada seguindo uma série de estados configurados anteriormente. Em cada estado uma função do código decide qual estado seguinte deve ser tomado, para que no fim o diagrama completo de estados e transições represente a função ou código implementado7.

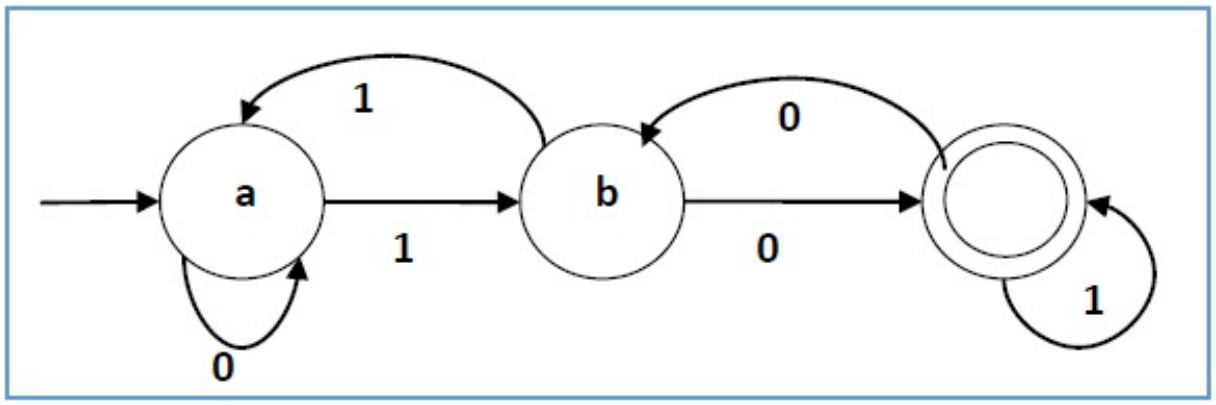

Figura 7 – Exemplo de um autômato finito determinístico

3.4 GraphViz

Para a automatização da criação dos autômatos foi escolhido utilizar o GraphViz, uma ferramenta de construção de gráficos de codigo aberto, e utilizando a linguagem DOT(6). Essa linguagem utiliza-se de uma gramática abstrata para criação de diagramas de acordo com sua sintaxe.

3.5 Parser

Parser é uma parte principal dos compiladores, sendo responsável por fazer a quebra do texto em elementos que possam ser interpretados. Com base em tabelas da linguagem para qual o parser foi construído é possível identificar se a sintaxe está correta e apontar erros caso encontrado. Por meio da identificação da ordem da escrita com as instruções, sendo palavras reservadas e escritas nas tabelas da linguagem que são consultadas pelo parser, e os parâmetros associados à instrução é possível identificar os casos de erro de utilização de funções ou instruções.

4 ENSAIOS

Na primeira fase de execução do projeto foram criadas as premissas do trabalho, optando pelo foco nas rotinas do ScorBase. Cada linha de uma rotina ScorBase foi identificada com uma possível operação, sendo dividida em casos especiais, como se segue:

- Chamada: Instruções de *Call* e *Run* não necessariamente resolvem sua instrução no estágio seguinte, necessitando ter um tratamento diferente;
- Comentários: Linhas do código que devem ser ignoradas na criação do autômato;
- Casos: Instrução de *If* fará com que o código tenha 2 estados possíveis dependendo da validação da condição;
- Espera: Instrução de *Wait* dos Jobs da FS100 podem resultar em *timeouts* encerrando a execução da subrotina.

As instruções foram, então, divididas em comandos de *START* e *FINISH*, e utilizadas como os labels das transições de estados. Dessa forma, o código de uma subrotina do ScorBase *GET023* abaixo:

```
Set Subroutine GET023
Print to Screen: GET FROM Gravity Feeder
Call Subroutine SCRIPT.GET_FROM_GFDR1
Print to Screen: P1,P2, PB1:  'SCRIPT.P1', 'SCRIPT.P2' , 'SCRIPT.PB1'
Print to Screen: P3,P4, P5:  'SCRIPT.P3', 'SCRIPT.P4' , 'SCRIPT.P5'
Copy FS100 Position SCRIPT.P1 to Position 1001
Copy FS100 Position SCRIPT.P2 to Position 1002
Copy FS100 Position SCRIPT.P3 to Position 1003
Copy FS100 Position SCRIPT.P4 to Position 1004
Go to Position SCRIPT.PB1 Speed 50 (%)
FS100 Start Job GT023
Wait FS100_DELAY_TIME (10ths of seconds)
FS100 Job Wait 60 (seconds)
Send Message $Start to MANAGER  ID=TASK_ID
Return from Subroutine
```

Quando traduzida de forma manual para a linguagem DOT do GraphViz, com as transições de estado e considerando os casos especiais temos:

```
digraph GET023 {

A -> B [label = "START Subroutine GET023"];
B -> C [label = "START Call Subroutine SCRIPT.GET_FROM_GFDR1"];
C -> D [label = "FINISH Call Subroutine SCRIPT.GET_FROM_GFDR1"];
D -> E [label = "START Copy FS100 Position SCRIPT.P1 to Position 1001"];
E -> F [label = "FINISH Copy FS100 Position SCRIPT.P1 to Position 1001"];
F -> G [label = "START Copy FS100 Position SCRIPT.P1 to Position 1002"];
G -> H [label = "FINISH Copy FS100 Position SCRIPT.P1 to Position 1002"];
H -> I [label = "START Copy FS100 Position SCRIPT.P1 to Position 1003"];
I -> J [label = "FINISH Copy FS100 Position SCRIPT.P1 to Position 1003"];
J -> K [label = "START Copy FS100 Position SCRIPT.P1 to Position 1004"];
K -> L [label = "FINISH Copy FS100 Position SCRIPT.P1 to Position 1004"];
L -> M [label = "START Go to Position SCRIPT.PB1 Speed 50(%)"];
M -> N [label = "FINISH Go to Position SCRIPT.PB1 Speed 50(%)"];
N -> O [label = "START FS100 Job GET023"];
O -> P [label = "START Wait FS100_DELAY_TIME (10ths of seconds)"];
P -> Q [label = "FINISH Wait FS100_DELAY_TIME (10ths of seconds)"];
Q -> R [label = "TIMEOUT Job GET023"];
Q -> S [label = "FINISH Job Get023"];
S -> T [label = "Send Message $Start to MANAGER ID = TASK_ID"];
T -> U [label = "FINISH Subroutine GET023"];
}
```

E a partir de cada um desses códigos foi gerado um autômato como o da Figura 8. Essa mesma conversão foi feita para cada uma das subrotinas do ScorBase, obtendo um autômato para cada de forma manual, presentes no Anexo A.

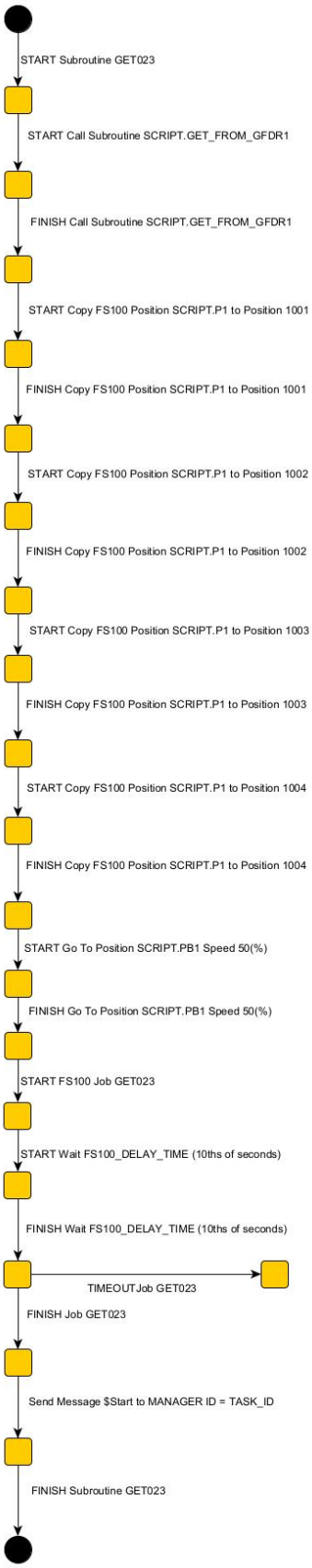

Figura 8 – Autômato da subrotina GET023 do ScorBase.

Outro caso importante presente no código foi a identificação das variáveis de memória do ScorBase, que apresentam uma lógica de autômatos diferente da lógica de uma subrotina. Para as variáveis devem ser previstos quais estados ela pode ser inicializada e quais as alterações podem ser executadas em seu valor. Um exemplo de código DOT do *GraphViz* está abaixo, seguido pelo seu autômato na Figura 9

```
digraph LATHE_LOAD{

A -> B
B -> C
B -> D [label = "READ LATHE_LOAD"];
D -> B [label = "LATHE_LOAD = 0"];
B -> E [label = "SET VARIABLE LATHE_LOAD = 1"];
E -> B [label = "SET VARIABLE LATHE_LOAD = 0"];
E -> F [label = "READ LATHE_LOAD"];
F -> E [label = "LATHE_LOAD = 1"];
E -> G
H -> E
}
```

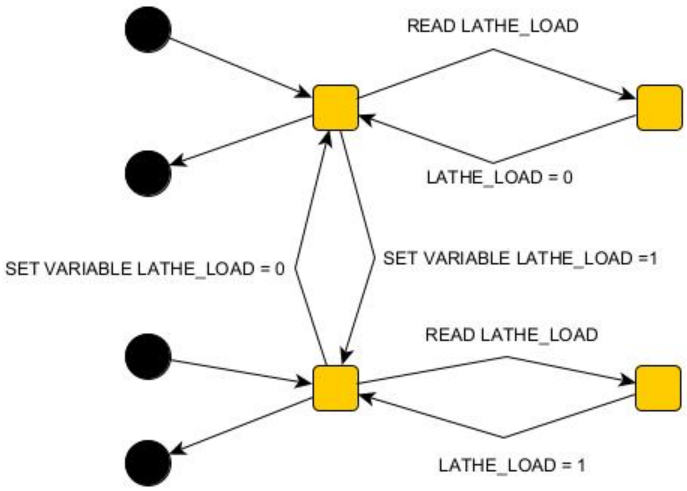

Figura 9 – Autômato da variável de memória LATHE_LOAD.

O próximo passo do projeto foi a construção da ferramenta de automação da criação de códigos DOT e os autômatos em png a partir do script de ScorBase. A linguagem escolhida para isso foi o Python, presente no Anexo B e também no endereço do *GitHub* <https://github.com/S-Santos17069/PFC-2021.git>, resultando em códigos como o que segum, novamente da subrotina GET023:

```
digraph GET023 {
0 -> 1 [label = "START Set Subroutine GET023"];
1 -> 2 [label = "START Print to Screen: GET FROM Gravity Feeder"];
2 -> 3 [label = "FINISH Print to Screen: GET FROM Gravity Feeder"];
3 -> 4 [label = "START Call Subroutine SCRIPT.GET_FROM_GFDR1"];
4 -> 5 [label = "FINISH Call Subroutine SCRIPT.GET_FROM_GFDR1"];
5 -> 6 [label = "START Print to Screen: P1,P2, PB1:  'SCRIPT.P1',
'SCRIPT.P2' , 'SCRIPT.PB1'"];
6 -> 7 [label = "FINISH Print to Screen: P1,P2, PB1:  'SCRIPT.P1',
'SCRIPT.P2' , 'SCRIPT.PB1'"];
7 -> 8 [label = "START Print to Screen: P3,P4, P5:  'SCRIPT.P3',
'SCRIPT.P4' , 'SCRIPT.P5'"];
8 -> 9 [label = "FINISH Print to Screen: P3,P4, P5:  'SCRIPT.P3',
'SCRIPT.P4' , 'SCRIPT.P5'"];
9 -> 10 [label = "START Copy FS100 Position SCRIPT.P1 to Position 1001"];
10 -> 11 [label = "FINISH Copy FS100 Position SCRIPT.P1 to Position 1001"];
11 -> 12 [label = "START Copy FS100 Position SCRIPT.P2 to Position 1002"];
12 -> 13 [label = "FINISH Copy FS100 Position SCRIPT.P2 to Position 1002"];
13 -> 14 [label = "START Copy FS100 Position SCRIPT.P3 to Position 1003"];
14 -> 15 [label = "FINISH Copy FS100 Position SCRIPT.P3 to Position 1003"];
15 -> 16 [label = "START Copy FS100 Position SCRIPT.P4 to Position 1004"];
16 -> 17 [label = "FINISH Copy FS100 Position SCRIPT.P4 to Position 1004"];
17 -> 18 [label = "START Go to Position SCRIPT.PB1 Speed 50 (%)"];
18 -> 19 [label = "FINISH Go to Position SCRIPT.PB1 Speed 50 (%)"];
19 -> 20 [label = "START FS100 Start Job GT023"];
20 -> 21 [label = "START Wait FS100_DELAY_TIME (10ths of seconds)"];
21 -> 22 [label = "FINISH Wait FS100_DELAY_TIME (10ths of seconds)"];
22 -> TIMEOUT [label = "TIMEOUT Job GT023"];
22 -> 23 [label = "FINISH Job GT023"];
23 -> 24 [label = "Send Message $Start to MANAGER  ID=TASK_ID"];
24 -> 25 [label = "FINISH Subroutine GET023
"];
}
```

E o autômato gerado foi o da Figura 10. Todos os códigos DOT e autômatos gerados pelo software estão presentes no Anexo C.

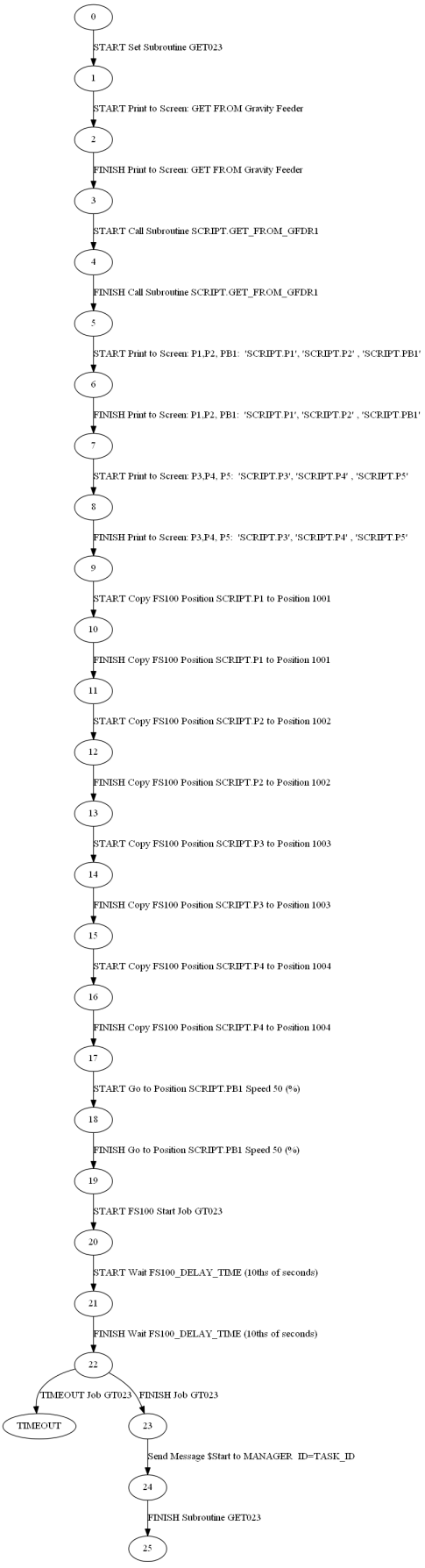

Figura 10 – Autômato da subrotina GET023 do ScorBase gerado pelo software.

Como é perceptível, os autômatos e o código gerado pelo programa automatizado tiveram uma grande confiabilidade quando comparados com os resultados de controle, gerados manualmente. Por isso a ferramenta pode ser considerada confiável dentro do escopo definido, fornecendo a capacidade de testes do código com alterações, devido a velocidade de testes e geração dos autômatos com a utilização da ferramenta, com uma análise posterior para validação dos autômatos gerados como plausíveis e para encontrar possíveis estados mortos, onde o autômato estaria encerrado, terminando o código.

A necessidade de conseguir avaliar códigos alterados antes de executar na plataforma é a possibilidade de encontrar os casos de erros ou de estados mortos, que poderiam danificar o maquinário causando um dano e um custo elevado para manutenção e conserto.

5 CONCLUSÃO

5.1 Contribuições

A ferramenta representou o início do desenvolvimento da capacidade operacional do laboratório de mecatrônica do IME. Com ela é possível validar os modelos criados em pesquisa de forma mais dinâmica e rápida devido a automatização do processo, tornando-o mais rápido. Com a ferramenta é possível visualizar os autômatos por meio de figuras permitindo a inspeção visual do funcionamento das rotinas alteradas.

A ferramenta também oferece um aumento na vida útil do sistema, pois a validação do código antes de utilizar na prática mitiga possíveis danos aos componetes mecânicos, diminuindo a necessidade de manutenção ou troca de equipamentos por falhas da parte de software.

5.2 Pontos não abordados

Alguns pontos observados como oportunidade de continuidade do projeto e que permitiriam uma abordagem completa da planta incluem:

- Inclusão de toda a gramática ScorBase, englobando as rotinas relativas às outras estações de trabalho;
- Inclusão das variáveis de memória do ScorBase e do controlador FS-100;
- Expansão da ferramenta para operar também com os Jobs do controlador FS-100;
- Inclusão dos diagramas ciberfísicos da planta.

5.3 Perspectivas de trabalho futuro

Dito isso, é de grande valia a continuidade desse trabalho para alcançar o completo domínio da plataforma aumento as capacidades da ferramenta. Ao incluir todas as rotinas do ScorBase e o controlador FS-100 é possivel criar um dicionário inclusivo das linguagens, tornando a ferramenta mais abrangente.

REFERÊNCIAS

1 PARSER. 2021. Site GNU Bison. Disponível em: <https://www.gnu.org/software/bison/manual/bison.html>. Acesso em: 13 setembro 2021.

2 INSTITUTE, P. M. Gerenciamento de escopo. In: _____. *Guia do conhecimento em gerenciamento de projetos - Guia PMBOK*. 14 Campus Boulevard Newtown Square, Pensilvânia, Estados Unidos: Project Management Institute, 2017. p. 129–171.

3 YED. 2021. Site com manual do yEd. Disponível em: <https://yed.yworks.com/support/manual/index.html>. Acesso em: 13 setembro 2021.

4 FS100 Controller. 2021. Site oficial da Motoman. Disponível em: <https://www.motoman.com/en-us/products/controllers/fs100>. Acesso em: 13 setembro 2021.

5 SCORBASE. 2021. Site oficial da Intelitek. Disponível em: <https://downloads.intelitek.com/Manuals/Robotics/ER-4u/Scorbase_USB_I.pdf>. Acesso em: 13 setembro 2021.

6 DOCUMENTATION for DOT in GraphViz. 2021. Site oficial do GraphViz. Disponível em: <https://graphviz.org/doc/info/lang.html>. Acesso em: 13 setembro 2021.