Ambiente Virtual para Treinamento de Controle Supervisório

Allan Yoshio Hasegawa[1]

[1]Universidade do Estado de Santa Catarina (UDESC)
Joinville – SC – Brasil

hasegawa.aran@gmail.com

1. Planta

Este trabalho automatizou uma planta virtual do software ITS PLC[1] da Real Games. ITS PLC é uma ferramenta didática que oferece cinco ambientes industriais emulados. Essas plantas virtuais simulam operações geralmente encontradas na indústria, e são executadas em tempo real usando gráficos 3D, som, simulações físicas e totalmente interativas. ITS PLC é um software privado e suporta, oficialmente, a automação por meio de CLP's.

A planta automatizada por este trabalho foi a *Palletizer*, que tem como objetivo principal o empilhamento de caixas em cima de um palete. A Figura 1 apresenta uma visão geral dos equipamentos dessa planta. As próximas subseções irão detalhar cada equipamento da planta.

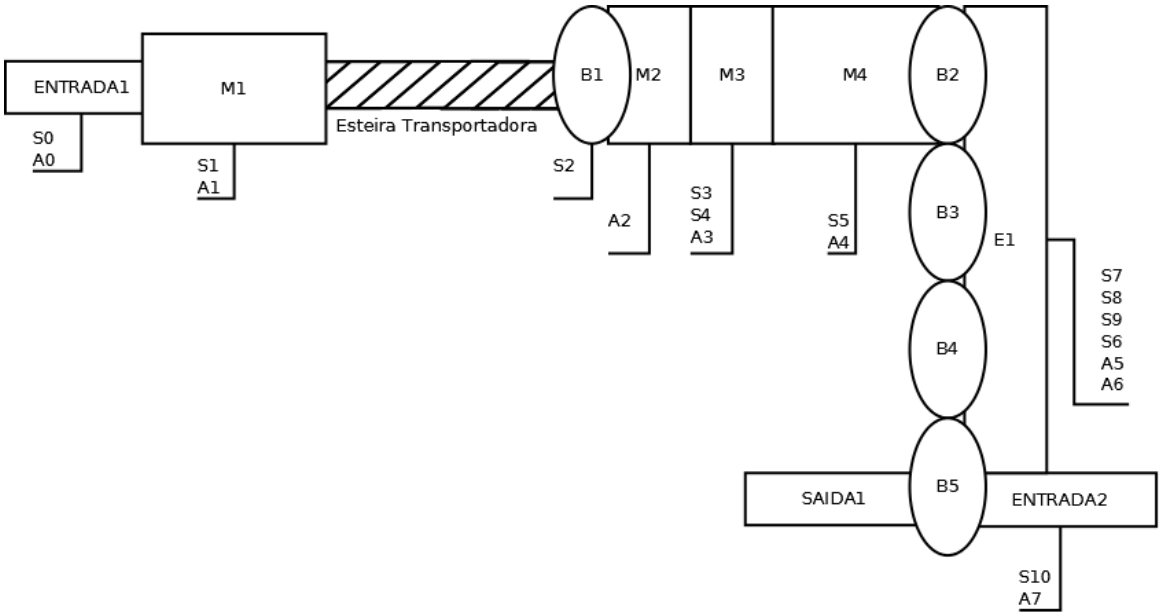

Figure 1. Visão geral da planta *Palletizer*.

1.1. ENTRADA1 e M1

Usando a Figura 1 como referência, os objetos **ENTRADA1** e **M1** têm como função:

- **ENTRADA1**: Entrada de caixas no sistema. Possui um atuador (**A0**) que liga e desliga um elevador que transporta as caixas. Quando uma caixa está em posição de retirada, o sensor (**S0**) é ativado.

[1]Disponível em: http://www.realgames.pt/index.php/en/products. Acesso (16/06/2013).

- **M1**: Transfere a caixa da **ENTRADA1** para a esteira transportadora. Ela possui um atuador (**A1**) que ao ligar empurra a caixa, acionando o sensor (**S1**), e ao desligar, volta a posição original.

Esses dois objetos são responsáveis pela entrada de caixas no sistema, e estão ilustrados na Figura 2 usando a planta virtual.

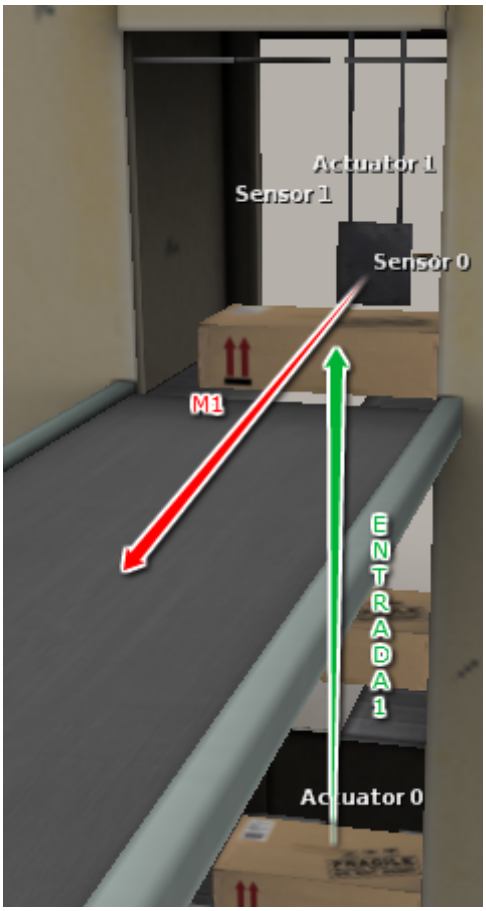

Figure 2. ENTRADA1 e M1 no ambiente virtual.

1.2. B1, M2, M3 e M4

A segunda parte da planta ajusta as caixas para serem empilhadas em cima de um palete. Essa parte é caracterizada pelos objetos:

- **B1**: Buffer com capacidade para duas caixas. Possui um sensor (**S2**) que indica a presença de uma caixa na sua entrada, ou seja, quando uma caixa entra no sistema, tal sensor é ativado e desativado. Ele permanecerá ligado apenas quando uma segunda caixa entrar no sistema (pois ela ficará na frente do sensor).
- **M2**: Máquina que bloqueia o avanço das caixas. Ela possui apenas um atuador (**A2**) que determina o bloqueio(ou desbloqueio) da caixa.
- **M3**: Chão retrátil. Tem como função transportar a caixa da **M2** até a **M4**. Possui um atuador (**A3**) que abre/fecha o chão, além de dois sensores (**S3** e **S4**) e indicam quando o chão está: totalmente aberto (**S3** ligado), totalmente fechado (**S4** ligado), ou em andamento (nenhum sensor ligado). **M3** precisa trabalhar em conjunto com

M2, pois caso **M2** libere uma caixa com **M3** aberto, essas irão cair na parte inferior da planta. Caso **M2** libere as caixas quando **M3** estiver fechado, essas caixas irão cair em uma região errada da **M3**, e problemas irão aparecer na próxima vez que **M3** se fechar, danificando as caixas. Essas duas máquinas precisam operar com um sincronismo correto.

- **M4**: Ajustador de caixas. Além de ajustar as caixas, essa máquina segura elas no lugar, evitando que elas caiam quando o chão retrátil (**M3**) abrir. Possui um atuador (**A4**) para ligar/desligar e um sensor (**S5**) que indica quando o ajuste está pronto.

A Figura 3 apresenta os objetos **B1**, **M2**, **M3** e **M4**.

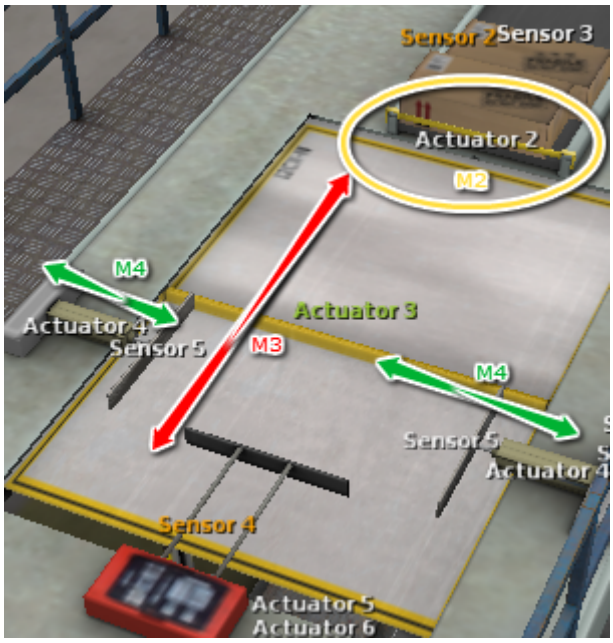

Figure 3. B1, M2, M3 e M4 no ambiente virtual.

1.3. ENTRADA2 e E1

A **ENTRADA2** possui apenas um atuador (**A7**) que liga a esteira que faz o transporte de novos paletes. O sensor (**S10**) indica quando um palete está na posição do elevador (**E1**). A Figura 4 mostra a **ENTRADA1** na planta virtual.

A planta possui um elevador **E1** que transporta caixas entre os buffers **B2**, **B3**, **B4** e **B5**. A disposição desses buffers pode ser visualizada na Figura 5. Usando o elevador em conjunto com os buffers, a planta consegue fazer o empilhamento de até três níveis de caixas. Inicialmente é colocado um nível de caixa no buffer **B2**. Esse nível é transferido para o buffer **B3** pelo elevador e um novo nível é adicionado no buffer **B2**. Esse processo é repetido para o próximo nível.

O elevador **E1** é controlado por meio dos atuadores (**A5**) e (**A6**) que comandam a subida e descida do elevador, respectivamente. Os sensores (**S7**), (**S8**), (**S9**) e (**S6**) informam as posições do elevador, essas sendo os buffers **B2**, **B3**, **B4** e **B5**, respectivamente.

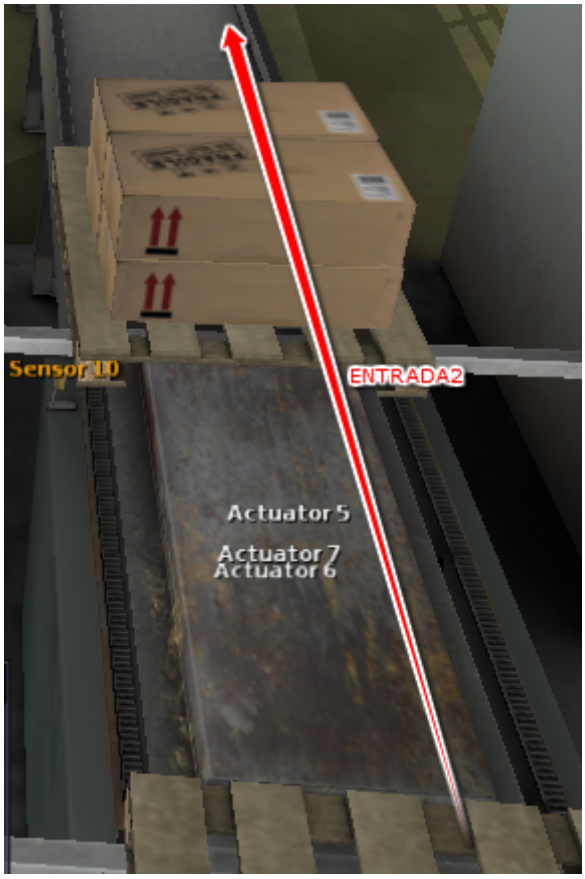

Figure 4. ENTRADA1 no ambiente virtual.

2. Controle Supervisório de Sistemas a Eventos Discretos

O controle supervisório deste trabalho fez uso de supervisores modulares locais da teoria de Sistemas a Eventos Discretos (SED). O primeiro passo na modelagem desses supervisores foi a abstração da planta para um modelo mais simples. Esse processo está descrito na Seção 2.1. O segundo passo foi o desenvolvimento das especificações, que está detalhado na Seção 2.2. Em seguida foi gerado um supervisor monolítico para fazer testes manuais de funcionalidade. Este está presente na Seção 2.3. Após testado o supervisor monolítico, o supervisor modular local foi gerado, e o teste de modularidade executado. Esses últimos passos da teoria de controle supervisório estão presente na Seção 2.4.

2.1. Abstração da Planta

A abstração da planta física apresenta dois objetivos. O primeiro é a própria modelagem da planta para a teoria de SED. O segundo objetivo é a simplificação que esses modelos proporcionam. A manutenção manual dos modelos é facilitada usando eventos mais intuitivos, como pode ser visto no primeiro modelo de planta **P_INPUT_BOX**, Figura 6. Informações que são desnecessárias para o modelo do supervisor são descartadas.

O segundo modelo de planta, **P_RETRACTABLE_FLOOR**, Figura 7, apresenta o funcionamento da **M2** e **M3** em conjunto. Como visto na Seção 1.2, essas máquinas precisam trabalhar em sincronismo. A teoria de SED não oferece recursos de temporização, e a criação de especificações que impõem uma certa sequência enquanto bloqueia todas

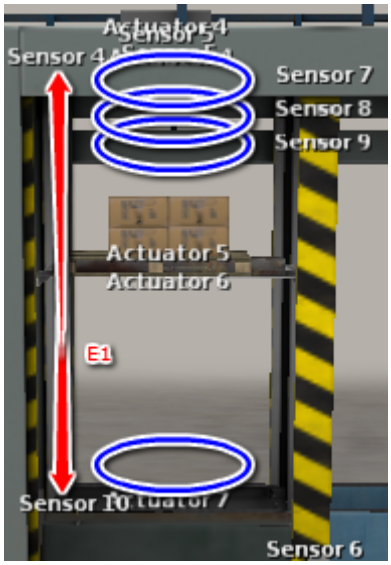

Figure 5. E1 e os buffers B2, B3, B4 e B5 no ambiente virtual.

as outras operações (assim evitando que **M2** seja acionada fora de tempo), impedem uma solução elegante para este problema. A solução adotada neste trabalho foi a abstração do funcionamento dessas duas máquinas com os mesmos eventos controláveis/não controláveis. O evento não controlável *end_mid_close_floor* foi inserido ao modelo após a realização de testes práticos. Essa novo evento permite que novas caixas possam entrar no sistema enquanto a porta estiver se fechando, otimizando o funcionamento da planta.

A Figura 8 apresenta o modelo de planta **P_ELEVATOR**. Ele permite que o elevador **E1** se mova para qualquer um dos quatro buffers **B2-5**. O próprio modelo de planta impede que o elevador vá para a posição que ele já esta. Paradas em lugares que não sejam os buffers não são possíveis no modelo, mas sim na planta, porém sem utilidade.

A **M4** esta modelada no modelo de planta **P_BOX_FITTER**, Figura 9. Esse modelo já informa a sequencia serial de operações que ela pode efetuar. Por último, o modelo de planta *P_INPUT_PALLET* é apresentado, Figure 10.

2.2. Especificações

Segue a lista das especificações implementadas neste trabalho:

1. **E_IPALLET_ELEVATOR** (Figura 11)
   (a) Evitar a entrada de um palete com o elevador fora da posição **B5**;
   (b) Evitar movimentos do elevador sem um palete.
2. **E_FITTER_FLOOR** (Figura 12)
   (a) Evitar abrir a porta retrátil sem fixar a caixa;
   (b) Evitar fechar a porta com o fixador ativado;
   (c) Evitar acionar **M4** sem uma caixa.
3. **E_ELEVATOR_FITTER** (Figura 13)
   (a) Evitar que o elevador desça um andar sem que a caixa esteja ajustada em cima dele, no andar adequado.
4. **E_IBOX_FLOOR** (Figura 14)
   (a) Evitar abrir o chão sem uma caixa no buffer de entrada;
   (b) Evitar iniciar entrada de caixa com a porta aberta.

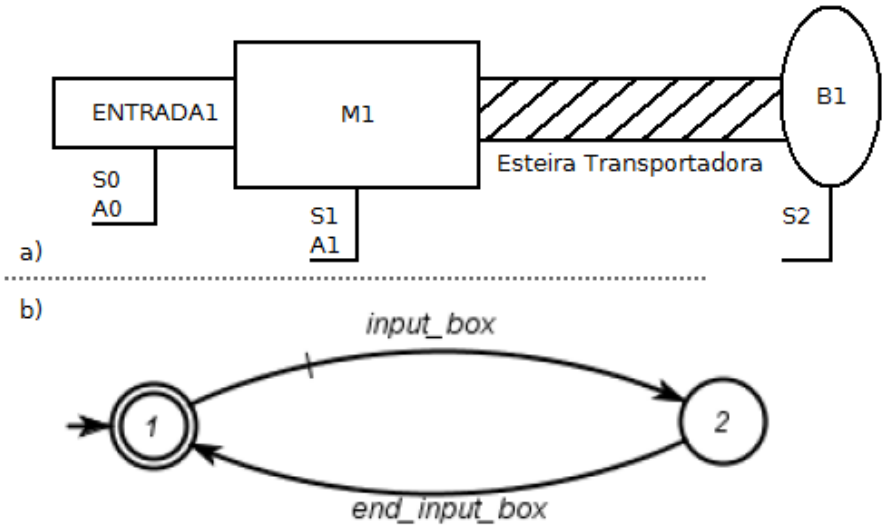

Figure 6. a) Parte original da planta. b) Modelo de planta P_INPUT_BOX.

2.3. Supervisor Monolítico

Etapas usadas para geração do supervisor monolítico:

1. $PlantTotal \leftarrow$ **P_INPUT_BOX** $\|$ **P_INPUT_PALLET** $\|$ **P_ELEVATOR** $\|$ **P_BOX_FITTER** $\|$ **P_RETRACTABLE_FLOOR**
2. $SpecsTotal \leftarrow$ **E_IPALLET_ELEVATOR** $\|$ **E_FITTER_FLOOR** $\|$ **E_ELEVATOR_FITTER** $\|$ **E_IBOX_FLOOR**
3. $K \leftarrow PlantTotal \| SpecsTotal$
4. Supervisor Monolítico $\leftarrow supcon(PlantTotal, K)$

A função *supcon* é implementada pelo software IDES[2]. A Tabela 1 apresenta os números de estados e transições dos modelos resultantes.

Table 1. Mono

Modelo	Número Estados	Número Transições
$PlantTotal$	720	4480
$SpecsTotal$	72	188
K	5920	20984
Supervisor Monolítico	339	753

2.4. Supervisor Modular Local

A primeira etapa para a modelagem dos supervisores modulares locais é a determinação dos sistemas produtos. A Tabela 2 apresenta os sistemas produtos gerados nesse trabalho.

Após criação dos sistemas produtos foi realizado os seguintes passos:

1. $K\_IPALLET\_ELEVATOR \leftarrow$ **S_IPALLET_ELEVATOR** $\|$ **E_IPALLET_ELEVATOR**
2. $K\_FITTER\_FLOOR \leftarrow$ **S_FITTER_FLOOR** $\|$ **E_FITTER_FLOOR**

[2]Disponível em: `https://qshare.queensu.ca/Users01/rudie/www/software.html`. Acesso (16/06/2013).

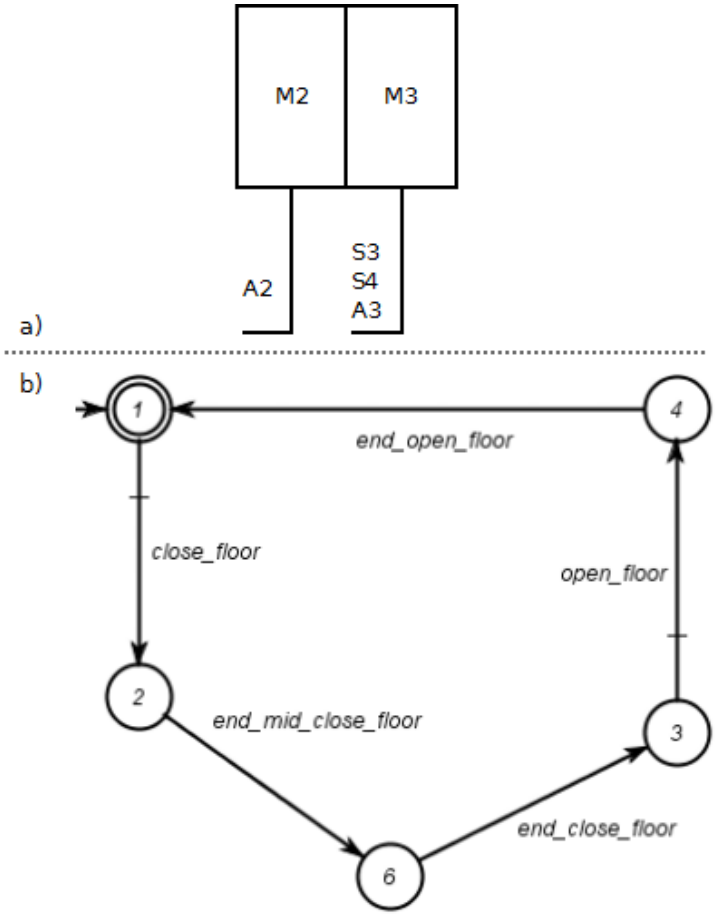

Figure 7. a) Parte original da planta. b) Modelo de planta P_RETRACTABLE_FLOOR.

3. $K\_ELEVATOR\_FITTER \leftarrow$ **S_ELEVATOR_FITTER** $\|$ **E_ELEVATOR_FITTER**
4. $K\_IBOX\_FLOOR \leftarrow$ **S_IBOX_FLOOR** $\|$ **E_IBOX_FLOOR**
5. $SUP\_IPALLET\_ELEVATOR \leftarrow supcon($**S_IPALLET_ELEVATOR**$, K\_IPALLET\_ELEVATOR)$
6. $SUP\_FITTER\_FLOOR \leftarrow supcon($**S_FITTER_FLOOR**$, K\_FITTER\_FLOOR)$
7. $SUP\_ELEVATOR\_FITTER \leftarrow supcon($**S_ELEVATOR_FITTER**$, K\_ELEVATOR\_FITTER)$
8. $SUP\_IBOX\_FLOOR \leftarrow supcon($**S_IBOX_FLOOR**$, K\_IBOX\_FLOOR)$

As operações acima geram os quatro supervisores modulares. O teste de modularidade foi feito sincronizando todos os supervisores, o que resultou no mesmo modelo do supervisor monolítico. A Tabela 3 apresenta o tamanho dos modelos obtidos. O número total de estados, somando todos supervisores, foi de 75, que é em torno de 4,5 vezes menos estados que o monolítico. Já em números de transições, o total foi de 110 no modular local, sendo em torno de 7 vezes mais eficiente que o monolítico.

3. Implementação

Diversos desafios que apareceram durante o desenvolvimento deste trabalho fizeram com que quatro sistemas fossem desenvolvidos. O primeiro desafio é a comunicação com o software ITS PLC, que é proprietário e só se comunica com CLP's. A solução para este problema está na Seção 3.1. O segundo desafio foi a virtualização de um microcontrolador PIC e a comunicação com outros softwares. Esse está discutido na Seção 3.2. Uma análise

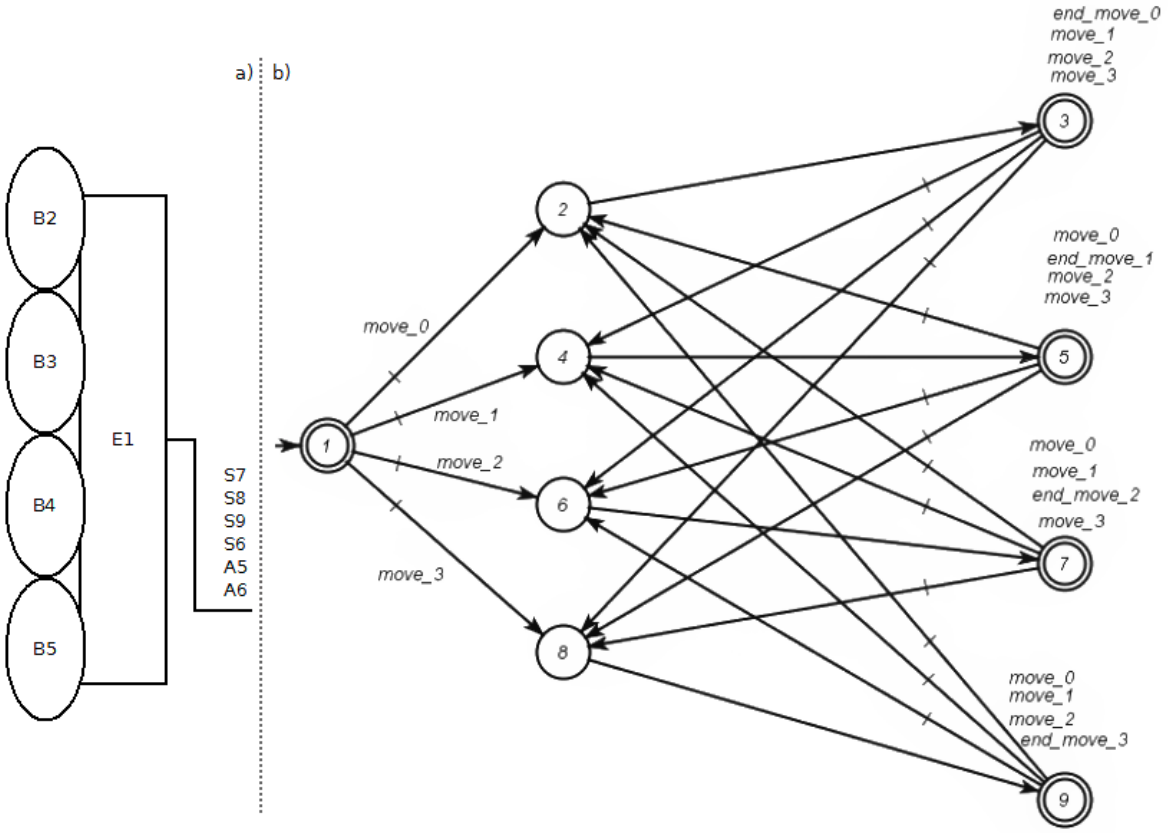

Figure 8. a) Parte original da planta. b) Modelo de planta P_ELEVATOR.

Table 2. Tabela de Sistema Produto.

Plantas	Especificações			
	IPALLET_ELEVATOR	**FITTER_FLOOR**	**ELEVATOR_FITTER**	**IBOX_FLOOR**
INPUT_BOX				X
INPUT_PALLET	X			
ELEVATOR	X		X	
BOX_FITTER		X	X	
RETRACTABLE_FLOOR		X		X
Sistema Produto	**S_IPALLET_ELEVATOR**	**S_FITTER_FLOOR**	**S_ELEVATOR_FITTER**	**S_IBOX_FLOOR**

do programa embarcado no PIC é feita na Seção 3.3. Para facilitar a geração do programa PIC para os supervisores modulares locais, e assim evitar erros humanos, um compilador foi desenvolvido para converter modelos no formato GRAIL+ para instruções C. Esse compilador é detalhado na Seção 3.4.

A Figura 15 mostra uma visão geral do sistema, e como as várias componentes se comunicam.

3.1. ipgm

O software ITS PLC não suporta comunicação com microcontroladores PIC. Por causa disso, esse trabalho desenvolveu uma biblioteca capaz de se comunicar com o software ITS PLC e oferecer uma interface genérica para qualquer aplicação, assim funcionando como um *middleware*. Essa característica da biblioteca dá o nome a ela: ITS PLC Generic Middleware, ou, ipgm. O código fonte da ipgm está disponível no endereço `https://github.com/AranHase/ipgm`.

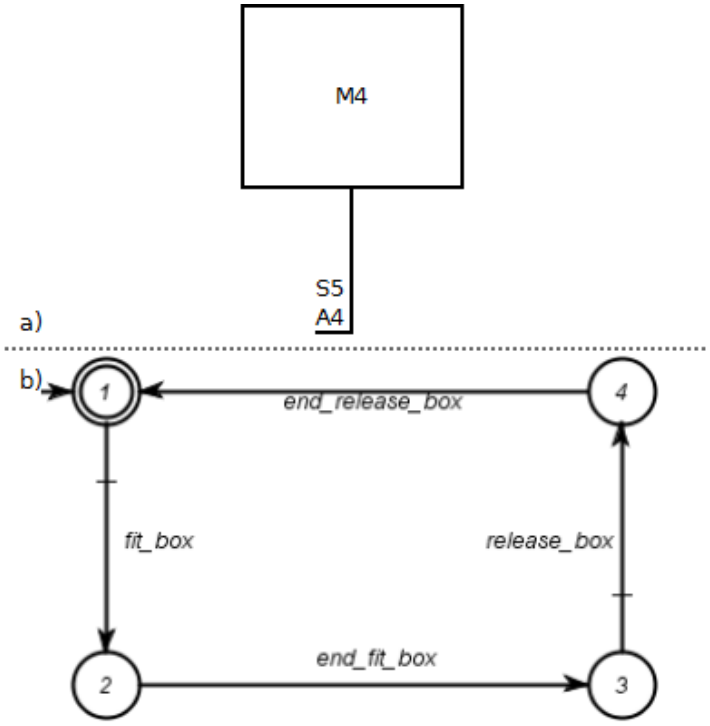

Figure 9. a) Parte original da planta. b) Modelo de planta P_BOX_FITTER.

Table 3. Resultado Supervisores Modular Local.

Modelo	Número Estados	Número Transições
$SUP\_IPALLET\_ELEVATOR$	42	65
SUP_FITTER_FLOOR	9	9
$SUP\_ELEVATOR\_FITTER$	13	19
SUP_IBOX_FLOOR	11	17

A ipgm usa técnicas de processamento de imagem para detectar o estado atual da planta virtual, assim capturando informações como sensores e atuadores ativos. Além de coletar informações em tempo real, a biblioteca também é capaz de manipular a planta (ativando/desativando atuadores) simulando o uso de um teclado. Todas essas operações são feitas sem alterar qualquer parte do software ITS PLC. Todas elas foram feitas sem interferir com o processo do software ITS PLC.

A Figure 16 apresenta o diagrama de classe da arquitetura usada para o desenvolvimento da ipgm. Destaque para as funções de *callback*, pois uma aplicação precisa apenas implementá-las para se comunicar com o ITS PLC.

3.2. itspic e PICsim

PICsim[3] foi o simulador de microcontroladores PIC usado neste trabalho. Ele oferece kits de desenvolvimento/testes completos, incluindo funcionalidades como LEDs, teclados, botões, LCD e principalmente porta serial. A comunicação serial foi o meio escolhido para enviar e receber informações do PIC. Assim, a aplicação itspic é criada, onde ela tem

[3]Disponível em: `http://sourceforge.net/projects/picsim/`. Acesso (16/06/2013).

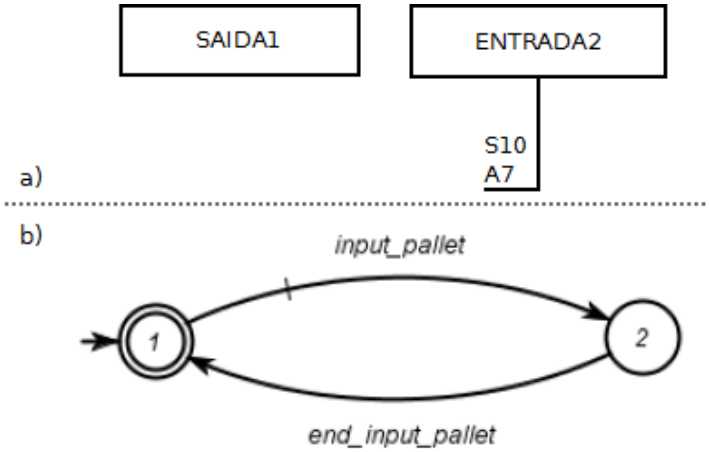

Figure 10. a) Parte original da planta. b) Modelo de planta P_INPUT_PALLET.

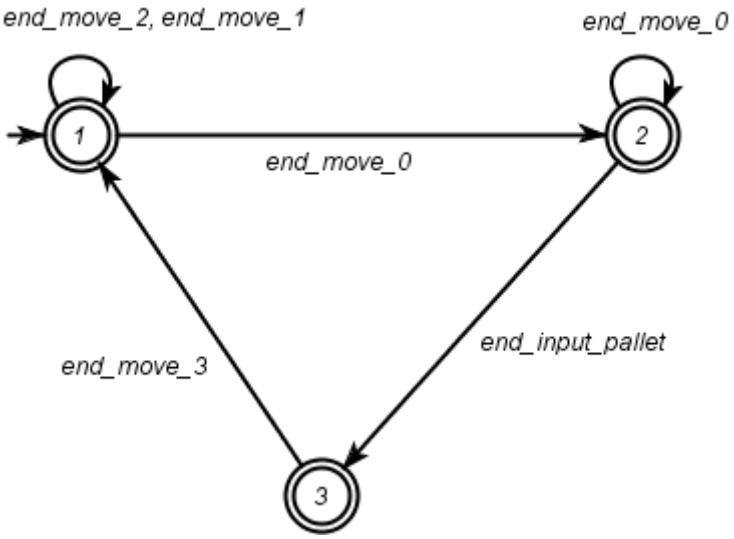

Figure 11. Especificação E_IPALLET_ELEVATOR.

como principal função captar os *callbacks* da ipgm e enviar as informações para o microcontrolador PIC, além de monitorar a porta serial e ativar atuadores quando requisitado.

O protocolo de comunicação implementado faz uso de apenas 1 byte por mensagem. Isso é possível por causa do número limitado de informações (sensores/atuadores). Um byte consegue armazenar números inteiros de 0 a 255. A planta oferece 11 sensores. Cada sensor apresenta dois estados, ligado/desligado, assim, cada sensor recebe dois números únicos entre 0 e 255. Esse número então é transmitido para indicar se o sensor foi apagado ou ligado. O mesmo é feito para os 8 atuadores. Outros valores foram usados para mensagens diversas, como reportagem de erro no sistema, por exemplo.

3.3. Programa PIC

O fluxo de operação do programa PIC implementado neste trabalho pode ser visto na Figura 17. Na parte da direita da imagem estão três atividades executando de modo assíncrono, essas são: Serial RX (recebimento de mensagens pela porta serial), Serial TX (envio de mensagens pela porta serial) e TMR0 (*timer*).

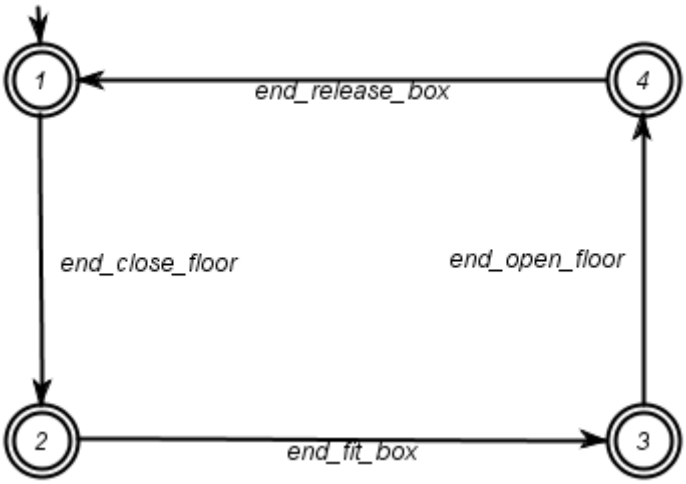

Figure 12. Especificação E_FITTER_FLOOR.

As operações no meio da Figura 17 representam o *loop* principal do programa. Segue uma lista descritiva dessas funções:

- *sup_response_step*: Executa todas as transições de estados, de todos os supervisores, que fazem uso de eventos não controláveis.
- *sup_des_step*: Executa todas as desabilitações de eventos controláveis de todos os supervisores.
- *sup_decisions_step*: Esse passo decide qual o próximo evento controlável a ser executado. A decisão implementada neste trabalho é de escolha aleatória de eventos.
- *sup_advance_step*: Executa todas as transições de estados, de todos os supervisores, de acordo com a decisão tomada no passo anterior.

O último passo do *loop* antes de sua repetição é executar as sequencias operacionais. As sequencias operacionais são implementações em baixo nível de todos os eventos controláveis. Por exemplo, o evento *input_box*, da Figura 2, é implementado usando como base o objeto *EventoControlavel* da Figura 17. Para toda transição de estado com o evento controlável *input_box*, o método *start()* desse objeto será chamado, e enquanto ele estar ativo, o método *update()* irá ser executado. O método *update()* tem como função executar as operações necessárias para realização do evento controlável, e ao concluir, criar um evento não controlável indicando o seu fim, nesse exemplo sendo *end_input_box*.

3.4. grail_itspic_compiler

Um compilador, chamado grail_itspic_compiler, foi desenvolvido para converter os modelos dos supervisores em instruções C, assim automatizando parcialmente o programa que é embarcado no PIC. O compilador recebe como entrada diversos supervisores modulares locais, em formato GRAIL+, além de arquivos implementados manualmente. A saída deste compilador é um arquivo *main.c* que é então compilado pelo XC8[4], assim gerando um arquivo *.hex* que é então carregado pelo PIC. Esse fluxo de trabalho é visto na Figura 18.

[4]Disponível em: `http://www.microchip.com/pagehandler/en_us/devtools/mplabxc/`. Acesso (16/06/2013).

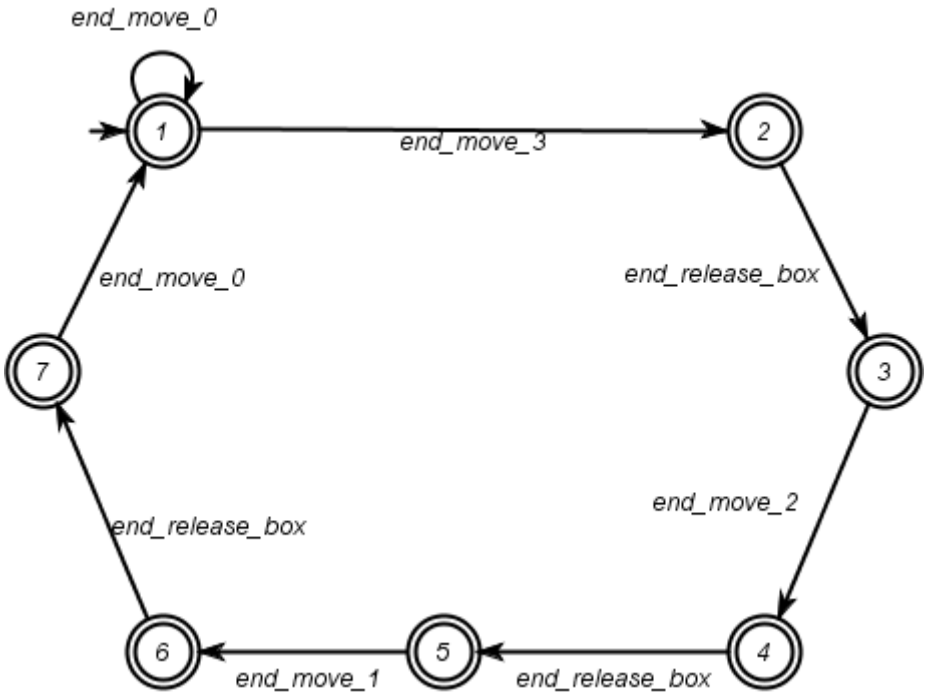

Figure 13. Especificação E_ELEVATOR_FITTER.

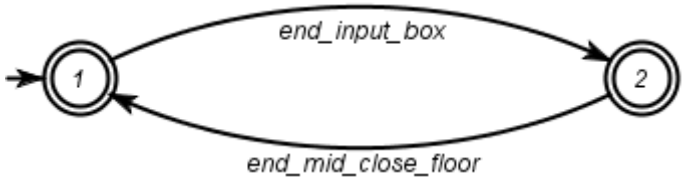

Figure 14. Especificação E_IBOX_FLOOR.

As funções *sup_response_step*, *sup_des_step*, *sup_decisions_step* e *sup_advance_step* da Figura 17 são geradas automaticamente, enquanto o resto, como sequencias operacionais, são geradas manualmente. Isso dá a flexibilidade para alterações no modelo dos supervisores de forma eficiente, portanto que novos eventos controláveis não sejam introduzidos no sistema. Outra vantagem no uso deste compilador é a redução de falhas humanas, pois o processo de implementação dos supervisores pode ser extenso e tedioso.

4. Resultados

Este trabalho conseguiu fazer a automação da planta virtual *Palletizer* da ITS PLC, porém, apenas quando a planta se comporta bem. Um dos pontos interessantes do software ITS PLC é que ele executa simulações físicas, assim reduzindo a diferente entre realidade e virtual. Essa característica na simulação implica que ao entrar uma caixa no sistema, por exemplo, ela pode vir em uma posição errada, ou cair de forma errada na esteira transportadora, com isso, gerando uma infinidade de estados da planta.

Um vídeo demonstrando o funcionamento deste conjunto de software, em uma planta bem comportada, pode ser visto no endereço `https://www.youtube.com/watch?v=Hh3ZN3VIYo4`. A introdução de eventos inesperados na planta leva o sis-

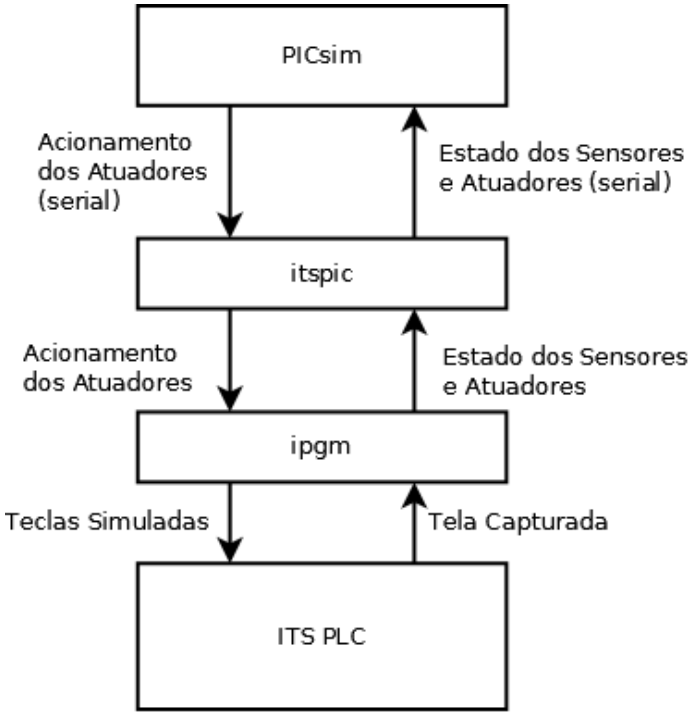

Figure 15. Visão geral do sistema.

tema a funcionar de forma indefinida.

5. Conclusões

Este trabalho fez a implementação da integração de um ambiente virtual para práticas de técnicas de controle supervisório, usando apenas software gratuitos e acessíveis a estudantes. O sistema possibilita a comunicação entre qualquer planta virtual do software ITS PLC com softwares externos usando a linguagem C++.

A planta *Palletizer* do ITS PLC foi automatizada usando a teoria de controle supervisório de SED. Modelos de supervisores modulares locais foram desenvolvidos, e uma implementação em um microcontrolador foi feita. Este trabalho conseguiu automatizar com sucesso a planta virtual, porém apenas quando ela é bem comportada. A introdução de elementos externos, ou instabilidade na planta, geram resultados indefinidos.

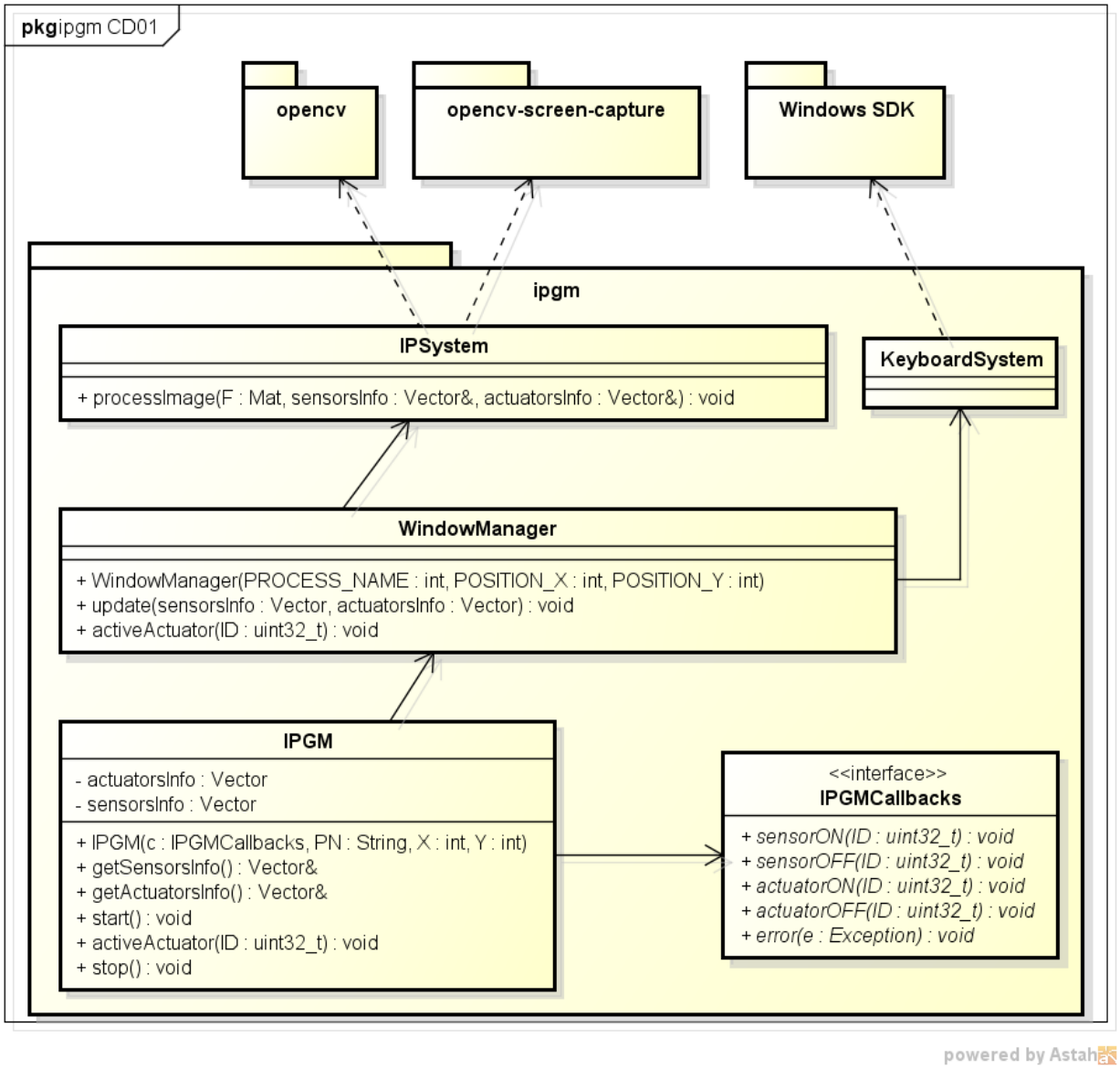

Figure 16. Arquitetura da biblioteca ipgm.

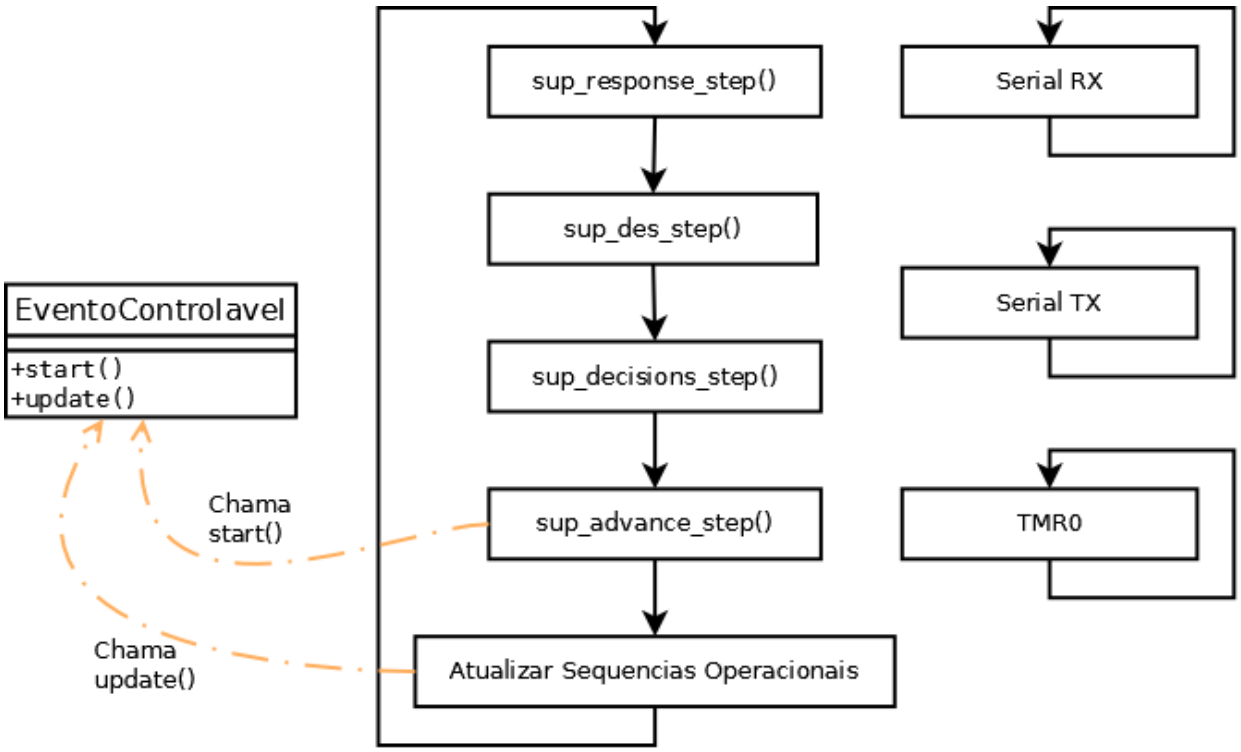

Figure 17. Fluxo do Programa PIC.

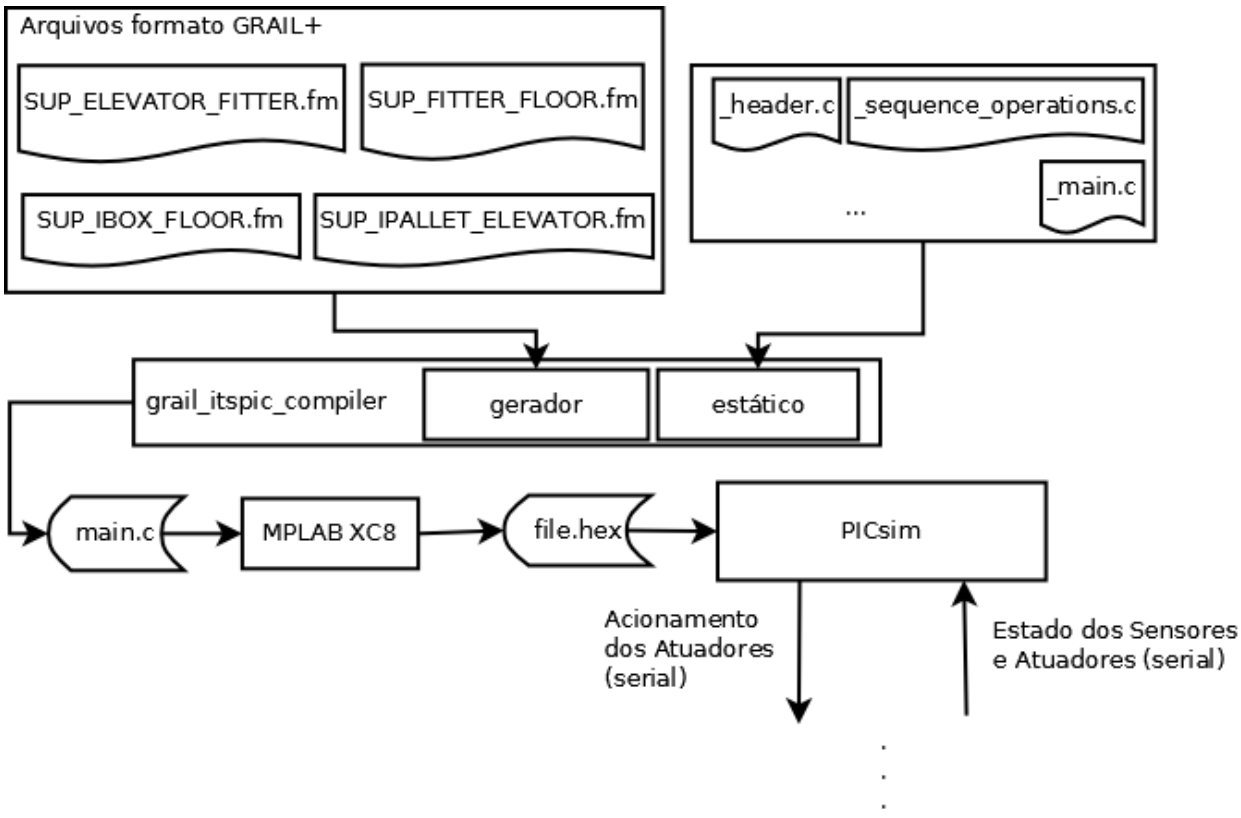

Figure 18. Fluxo de trabalho da compilação.