Universidade do Porto
Faculdade de Engenharia
FEUP

Mestrado Integrado em
Engenharia Eletrotécnica e de Computadores

UC Sistemas e Automação

Guião de trabalho prático

Introdução ao FEUPAutom & Parque 3D & Sistemas de Máquinas de Estados

Armando Jorge Sousa – asousa@fe.up.pt
José António Faria – jfaria@fe.up.pt

Versão: 26 fevereiro 2018
(Lab05 - Intro FeupAutom + 3D Parq v18.docx)

Apresentação

Leia todo o guião incluindo o anexo antes de iniciar o seu trabalho.

Este trabalho prático tem por objetivo apresentar o software **FEUPAutom**, desenvolvido por docentes da FEUP* com o objetivo de apoiar o desenvolvimento de sistemas automatizados guiados por eventos discretos.

Durante o decorrer do trabalho irá programar de forma Ad-Hoc e depois implementar um Sistema de Máquinas de Estados.

O modo de funcionamento do FEUAutom é muito semelhante aos dos autómatos programáveis que permitem controlar sistemas reais.

O trabalho prático ocupa uma única aula prática cujo plano é o seguinte:

- Na primeira parte, vai testar uma pequena aplicação de demonstração com o objetivo de se familiarizar com o princípio de funcionamento do FEUPAutom.

- Na segunda parte, vai executar um conjunto de exercícios que percorrem as principais funcionalidades oferecidas pelo FEUPAutom para apoiar o desenvolvimento de aplicações de controlo.

- Na terceira parte vai desenvolver o programa de controlo de outra versão mais completa do parque de estacionamento.

* Armando Sousa, asousa@fe.up.pt e Paulo Costa, paco@fe.up.pt

Preparação da Aula

A preparação obrigatória para a aula consiste:

1. No estudo atento de todo este guião (incluindo o anexo)
2. Realizar os exercícios E1 a E5
3. Levar os Diagramas de Transições de Estados (DTEs) relativos à parte 3 do guião

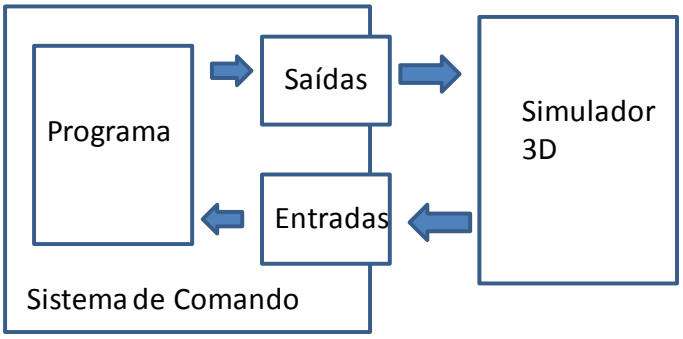
Obs: estude os elementos a submeter no final da aula

Parte 1: Demonstração do FEUPAutom

O software FEUPAutom será apresentado nas aulas teóricas e está disponível para "*download*" no moodle da UC. Descarregue o pacote com o FEUPAutom e os simuladores; atualize o FEUPAutom mantendo a estrutura de diretorias. Pode instalar estas aplicações no seu computador pessoal e ou utilizar um dos PC's do laboratório.

As aplicações contêm dois componentes principais:

- O **Sistema de Comando** através do qual são editados e executados os programas de controlo (e que é, sensivelmente, equivalente ao uComputador que utilizou no trabalho prático anterior)
- O **Simulador 3D** que simula os sistemas físicos controlados pelo Sistema de Comando de uma forma visual e realista.



No modo de funcionamento normal, o Sistema de Comando:

- Lê os valores das entradas (provenientes do simulador 3D, via entradas)
- Executa o programa de controlo e determina o novo valor das saídas
- Atualiza (escreve) o valor das saídas (a enviar para o Simulador 3D)

Por seu lado, o Simulador 3D:

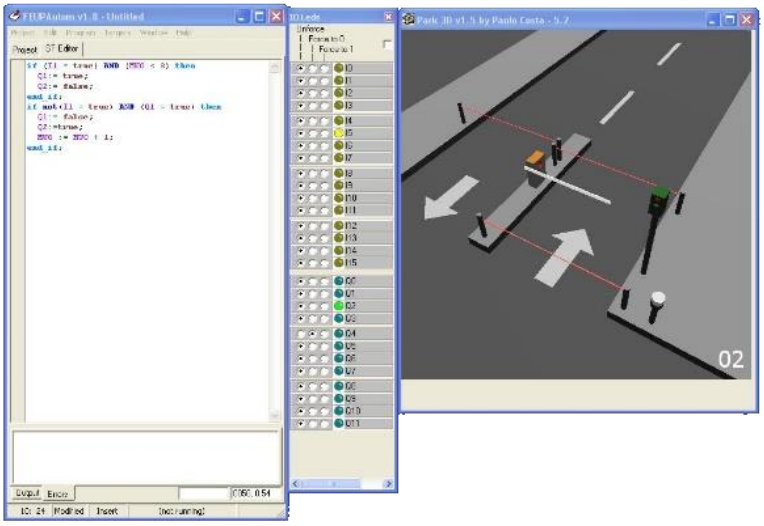
- Implementa os comandos provenientes do Sistema de Comando
- Simula a física envolvida (movimento, etc) e calcula os sensores simulados
- Envia para o Sistema de Comando

Para se familiarizar com o princípio de funcionamento do FEUPAutom, vai testar uma pequena aplicação de demonstração que controla a entrada de um parque de estacionamento idêntico ao do trabalho prático anterior.

1.1. Aplicação de demonstração

Para testar a aplicação de demonstração, proceda da seguinte forma:

- Efetue o *download* o ficheiro *FEUPAutom.zip* disponível na página da disciplina e descompacte-o para uma pasta à sua escolha dentro do **drive F**:
- Execute o programa *FEUPAutom.exe* presente na pasta *FEUPAutom*
- No menu *Targets* selecione, *Simple Park (Instant Barrier)*
- No menu *Window*, ative a janela *Inputs/Outputs*
- Redimensione e reposicione as janelas para ficar com uma interface próxima da da figura ao lado.



- Edite o seguinte programa na janela ST Editor (o mais simples será fazer copiar/colar):

```
if    (I1) then
      Q1 := true;
      Q2 := false;
end_if;

if    (not I1)  then
      Q1 := false;
      Q2 := true;
end_if;
```

- Compile o programa (menu Programs\Compile) e verifique que não ocorreram erros de compilação.
- Coloque o programa em execução (menu Programs\Run)
- Agora, na janela do simulador, clique sobre a seta junto à entrada para pedir um novo carro
- Na janela I/O Leds, observe e interprete a evolução dos seguintes linhas de entrada e saída:
  - I1: celula_entrada
  - Q1: abrir_cancela
  - Q2: fechar_cancela

1.2. Explicação do funcionamento

- Quando um novo carro se aproxima da entrada e atinge uma posição em que interrompe o feixe da célula fotoelétrica de entrada, o Simulador ativa a entrada do Sistema de Comando correspondente a essa célula (neste caso, a entrada I1)

- O programa deteta a alteração em I1, ativa a saída correspondente à abertura da cancela de entrada (Q1) e desativa a saída correspondente ao fecho da cancela (Q2).

```
if   (I1) then
     Q1 := true;
     Q2 := false;
end_if;
```

- O simulador recebe esta alteração das saídas do Sistema de Comando e atua sobre o sistema físico, isto é, movimenta a cancela no sentido da abertura.

- Estando a cancela aberta, o Simulador faz avançar o carro.

- Depois do carro entrar no parque, a célula de entrada deixa de estar atuada pelo que o Simulador desativa a entrada I1.

- O programa deteta esta alteração e desativa a saída Abrir cancela (Q1) e ativa a saída fechar cancela (Q2).

```
if   (I1=false)  then
     Q1 := false;
     Q2 := true;
end_if;
```

- Por seu lado, o simulador deteta esta nova alteração nas linhas de saída e movimenta a cancela no sentido do fecho.

- **Nota: Este programa muito simples deve ser melhorado ☺**

- Num programa real será conveniente utilizar nomes descritivos para cada variável em vez de I1 e Q1 e Q2.

Parte 2: ST & FEUPAutom

Nesta segunda parte do trabalho prático vai realizar um conjunto de pequenos exercícios práticos com o objetivo de se familiarizar com as funcionalidades oferecidas pelo FEUPAutom para apoiar o desenvolvimento de aplicações de controlo.

2.1. Linguagem de programação

A linguagem de programação utilizada no FEUPAutom é a linguagem ST – Structured Text.

Trata-se de uma linguagem standard para a programação de dispositivos automáticos (entre os quais os autómatos programáveis) que faz parte da norma IEC 61131 e que, atualmente, é suportada por praticamente todos os fabricantes de autómatos programáveis.

A linguagem ST será brevemente apresentada nas aulas teóricas e, no help do FEUPAutom, pode encontrar o respetivo manual de referência.

A linguagem ST foi inspirada do Pascal e, por isso:

- As atribuições são efetuadas com :=
- As comparações são efetuadas com =
- Não distingue maiúsculas e minúsculas

Na escrita dos programas, para melhorar a legibilidade do código:

- Escreva uma instrução por linha
- Utilize uma indentação correta
- Utilize nomes descritivos para todas as variáveis
- Utilize (* comentários *) ou // comentários

Relativamente ao editor:

- Não é possível alterar código durante a sua execução
- Utilize **ctrl space** para chamar o menu de "*completion*" – este menu lista variáveis e keywords frequentes da linguagem (não todas!).
- Tenha em atenção a colorização do código à medida que o vai introduzindo.

2.2. Traçados temporais

O FEUPAutom permite efetuar traçados da evolução temporal das variáveis de um programa, um pouco como um osciloscópio, o que pde ser muito útil na fase de teste dos programas.
Para ilustrar esta funcionalidade, proceda do seguinte modelo:

- Abra a janela dos traçados temporais selecionando *Log & Trace* no menu *Window*.
- De seguida, edite o seguinte programa:

```
Q0 := I0 and I1;
Q1 := I0 or I1;
```

- Execute o programa em ciclo infinito (menu Program\Run)
- Faça experiências com o conceito de forçagem de variáveis: o normal é a variável não estar forçada e uma entrada será proveniente da simulação e uma saída proveniente do script; outros estados são forçar a False ou a True. Modifique através de forçagens os valores das entradas I0 e de I1 (para isso, deve ativar a janela Inputs/Outputs), e confira os valores de Q0 e Q1.
- Páre a execução do programa (menu Program\Stop) e escolha as variáveis que pretende ver no gráfico. Neste caso, selecione I0,I1, Q0 e Q1

<u>Exercício E1</u>

Copie o seguinte programa de controlo do parque para o editor:

```
if    (I1) then
      Q1 := true;
      Q2 := false;
end_if;

if    (I1=false) and (I2=false)  then
      Q1 := false;
      Q2 := true;
end_if;
```

(I2 diz respeito ao sensor depois da cancela)

Coloque o programa em execução e simule a entrada de vários carros.

Obtenha o traçado temporal das variáveis relevantes e copie-o (com ALT+PRTSCR) para posterior inclusão no relatório do trabalho a enviar ao docente.

Em que situação é que este programa não funcionaria?

2.3. Nomes das variáveis

Os nomes atribuídos por defeito aos sinais de entrada e saída (I0, I1, .., Q0, Q1, ...) e às variáveis de memória (M0, M1, ..., MW0, MW1, ...) podem ser alterados.

Para isso, deve ativar a janela *Variables* (menu *Window*) e filtrar pelo tipo de variáveis que quer alterar (para já considere apenas *Inputs*, *Outputs* e *MemWords*).

De seguida pode alterar o nome da variável escrevendo o novo nome no campo *UserName*.

Exercício E2

Altere os nomes de:

- entrada I1 para CelulaEntrada
- entrada I2 para CelulaSaida
- saída Q1 para AbrirCancela
- saída Q2 para FecharCancela

De seguida, re-escreva o programa de controlo anterior agora com nomes descritivos para as variáveis.

Dica: utilize CTRL+SPACE para evitar teclar nomes compridos de variáveis

2.4. Forçagem do valor das entradas e saídas

Como se viu na primeira parte do trabalho, no modo de funcionamento normal do FEUPAutom:

- O programa de controlo lê as entradas e escreve nas saídas
- O simulador 3D lê as saídas e escreve nas entradas.

No entanto, durante a fase de desenvolvimento e teste dos programas, é muito útil poder forçar (i.e., impor) valores a linhas de saída, independentemente dos valores gerados pelo Sistema de Controlo, e forçar valor de entradas, independentemente dos valores gerados pelo Simulador.

A possibilidade de atuar diretamente sobre as entradas e as saídas permite ao Utilizar testar determinadas situações sob o seu controlo e, assim, diagnosticar mais facilmente eventuais erros de programação.

Para forçar o valor de uma entrada ou saída deve proceder da seguinte forma:

- Ativar a janela Inputs/Outputs no menu Window.
- Para forçar o valor 0 numa linha de entrada ou de saída, clicar sobre o botão correspondente à coluna “0” em frente a essa linha
- De forma idêntica, para forçar o valor 1, clicar sobre o botão da coluna “1”.

Exercício E3

Para testar esta funcionalidade, simule a chegada de vários carros à entrada do parque e comande a abertura da cancela e o semáforo atuando diretamente sobre as linhas de saída (Q1: abrirCancela, Q2: fecharCancela; Q3: semaforoVerde; Q4: semaforoVermelho).

Efetue o traçado temporal dos sinais I1, Q1, Q2, Q3 e Q4.

Copie a imagem do traçado para posterior inclusão no seu relatório do trabalho prático a enviar ao docente.

2.5. Variável Auxiliares

O FEUPAutom permite utilizar variáveis de memória do tipo bit e word:

- As variáveis do tipo bit (M0, M1, ...) armazenam um booleano [True (1) ou False (0)].
- As variáveis do tipo word (MW0, MW1, ...) armazenam um valor inteiro (16 bits sem sinal).

Exercício E4

Para testar a utilização de variáveis em memória:

- Introduza o seguinte programa:

```
m0  := not m0;
mw0 := mw0 + 1;
```

- Escolha repetidamente *Run Once* no menu *Program* (ou prima F8) para executar um ciclo de programa de cada vez.
- No final de cada ciclo, observe o valor de M0 e MW0 passando o cursor do rato por cima do nome da variável no editor (o valor é apresentado na barra de estado situada ao fundo da janela do editor).

2.6. Variáveis do sistema e Inicialização

O FEUPAutom tem um conjunto de variáveis especiais internas cujo valor é alterado pelo próprio FEUPAutom, mas que podem ser consultadas pelo programa.

A listagem de todas as variáveis internas do sistema pode ser encontrada no HELP.

Uma das mais importantes é a variável SW0, que conta o número de ciclos executados desde sempre no atual programa.

Esta variável pode ser utilizada para executar operações de inicialização do programa (i.e., operações que apenas são executadas uma vez, no primeiro ciclo do programa, como se verá no ponto seguinte).

Exercício E5

Para testar o comportamento da variável de sistema SW0, utilize as seguintes extensões FEUPAutom à linguagem ST:

- `WriteS` e `WriteSLn` imprimem uma 'string' (...Ln muda de linha)
- `WriteB` e `WriteBLn` imprimem um Booleano (...Ln muda de linha)
- `WriteI` e `WriteILn` imprimem um inteiro (...Ln muda de linha)

- Escreva as seguintes instruções no editor ST

```
WriteS('SW0='); writeILn(sw0);
```

  Prima CTRL+SHIFT+R * para re-inicializar todas as variáveis incluindo SW0.

- De seguida, prima F8 diversas vezes e depois F9 e depois F10 e depois F8.
- Como explica o resultado?

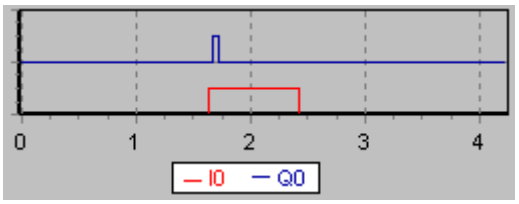
* CTRL+SHIFT+R é equivalente a menu *Program /Clear All* e re-inicializa todas as variáveis incluindo SW0.

2.7. Operadores de flanco (RE e FE)

A função especial RE (*Rising Edge*) fica activa durante um ciclo do autómato apenas se a variável em causa estava a False (0) no ciclo anterior e agora está a True (1).

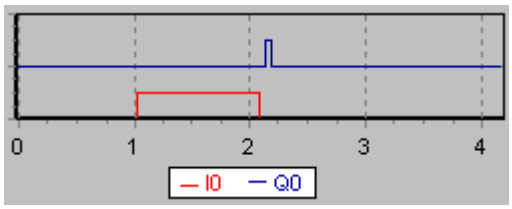
Teste a seguinte instrução visualizando o respetivo traçado temporal:

```
q0 := RE i0;
```



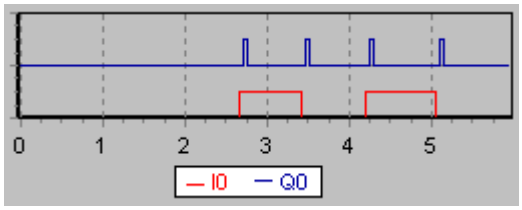
A função especial FE (*Falling Edge*) fica activa durante um ciclo do autómato apenas se a variável em causa estava a True (1) e está a False (0). Teste a seguinte instrução visualizando o respetivo traçado temporal:

```
q0 := FE i0;
```



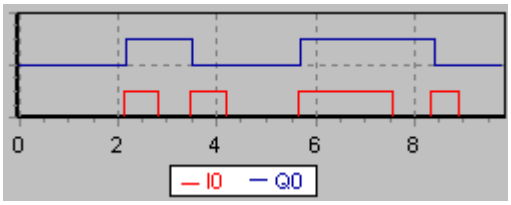
Introduza e analise o funcionamento da seguinte instrução:

```
q0 := (FE i0) or (RE i0);
```



Relembre que a função booleana XOR permite obter a funcionalidade de comutação (var_bool XOR 1 resulta em var_bool negada). Introduza e analise o funcionamento da seguinte instrução:

```
q0 := (RE i0) xor q0;
```



Resumo dos operadores Booleanos e sob flancos de variáveis

- Operadores Booleanos: **NOT**, **AND**, **OR**, **XOR**
- *Operadores de Flanco*: **RE** (*rising edge*) e **FE** (*falling edge*)

2.8. Temporizadores

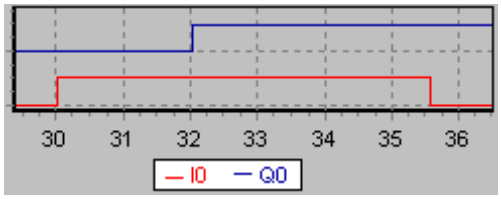
A contagem de tempo é um aspeto essencial nos programas de controlo.
Cada temporizador TMxx do FEUPAutom está associado à seguinte estrutura de dados: (i) o modo de operação `TMxx.mode`; (ii) o tempo pré-programado "preset" `TMxx.P` em décimos de segundo e (iii) o bit de saída `TMxx.Q` que indica se o tempo já passou ou não.

Os temporizadores do FEUPAutom nunca param (arrancam com o arranque do sistema) e o comando START marca o instante do início da contagem de tempo.
Por exemplo, para um temporizador `T0` em modo `TOn` e com `T0.P=20`, após o `start` iniciar a contagem de tempo, o bit `T0.Q` inicia-se a 0 e depois toma o valor 1 após ter decorrido o tempo pré-programado `T0.Q` (neste caso 20 x 0,1 seg = 2 seg).

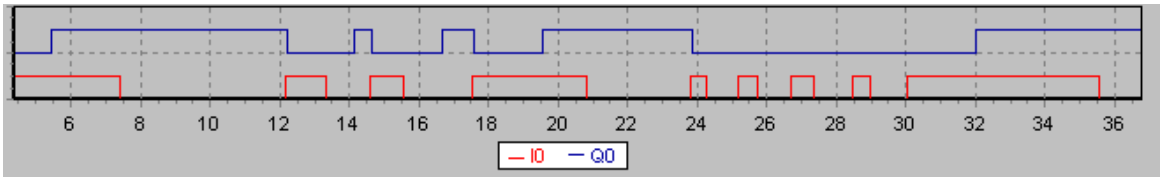
```
if RE i0 then           // inicializações
    t0.mode := TOn;     // modo de atraso à ativação
    t0.P := 20;         // tempo ativação em décimos de segundo
    start t0;           //  início da contagem do tempo
end_if;
q0 := t0.q;
```

Exercício E6

Teste o programa anterior numa situação simples, tal como exemplificado na figura seguinte.
O programa conta 2 segundos desde o flanco ascendente de I0. Passado esse tempo, a saída T0.Q toma o valor 1.



O mesmo programa numa situação mais complicada dá origem ao seguinte traçado temporal.



Obs1: Neste momento o FEUPAutom só trabalha com temporizadores TOn
Obs2: As inicializações só necessitam de ser feitas uma vez, no início do programa. Por vezes utiliza-se um teste para a variável SW0=0 para fazer as inicializações, por exemplo:

```
if sw0 = 0 then         // inicializações
    t0.mode := TOn;     // opcional, é o único "modo"
    t0.P := 20;         // 20 décimas de segundo
end_if;
```

2.9. Exercícios Livres, opcionais

Se ainda se sente pouco seguro quanto à utilização do FEUPAutom, deverá resolver os livres exercícios seguintes, antes de passar à terceira parte do guião, código livre (Ad-Hoc), isto é, sem utilizar máquinas de estados.

Exercício L91

Crie um programa que liga a saída Q0 sempre e só quando todas as entradas I0, I1 e I2 estiverem ligadas.

Exercício L92

Acrescente ao enunciado anterior a funcionalidade de ligar a saída Q1 quando pelo menos uma das saídas I0 ou I1 ou I2 estiverem ativas.

Exercício L93

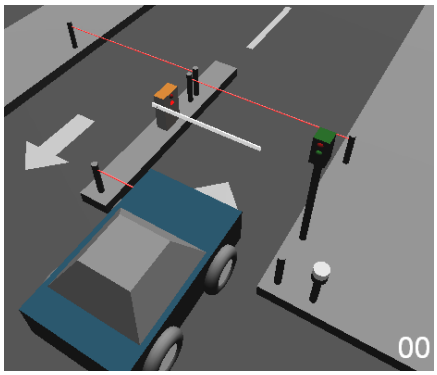
Crie um programa que no início ativa Q0 e a cada flanco descendente da variável I0 ativa a saída seguinte à que está atualmente ligada, criando a sequência: Q0, Q1, Q2, Q0, Q1…

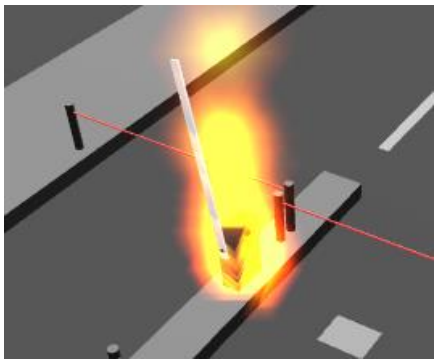
Exercício L94

Crie um programa que ligue Q0 quando I0 estiver ativo um tempo superior a 1 segundo (quando I0 ficar inativo, Q0 também deve ficar inativo).

Parte 3: Controlo de Parque Estacionamento

Considere um parque de estacionamento similar ao considerado no trabalho anterior, mas, agora, no posto de entrada do parque existem duas células fotoelétricas, uma localizada antes da cancela e a outra depois, conforme representado na figura ao lado. A saída é livre. Admita que o parque tem capacidade para **4 carros**.
Utilize sistemas de máquinas de estados com contadores e temporizadores (se entender necessário)





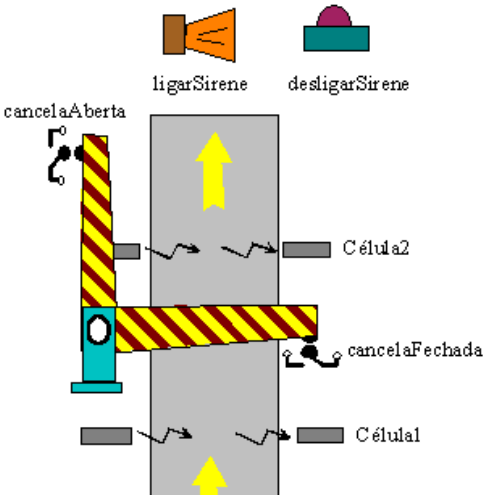
Interface:
As operações sobre o parque disponíveis no grafismo da aplicação são as seguintes:

- Clicar sobre a seta de entrada / saída para fazer entrar /sair carro, respetivamente.
- Clicar sobre a sirene para ativar o sensor ("Siren Off")

Nota: estas opções estão também disponíveis no menu de contexto do rato – premir o botão do lado direito do rato sobre a imagem

Se forem enviados comando incoerentes ou se a barreira bater no carro, é possível "estragar" a barreira. Esta situação de avaria é representada através de chamas na caixa da barreira – para a simulação voltar a funcionar, é necessário escolher "*Reset World*" ou premir sobre a chama.

Funcionamento do Parque:



- Desenvolva o programa de controlo do parque de forma que, quando um novo carro chega à entrada do parque (célula 1), e no caso de haver lugares disponíveis a cancela abra para o carro entrar; a cancela fecha depois do carro deixar de ser detetado pela célula 2 da entrada; para a abertura **abrirCancela** até **cancelaAberta** e depois para o fecho **fecharCancela** até **cancelaFechada** (ver variáveis, abaixo)
- Se, antes da cancela estar completamente fechada, a célula2 detetar a presença de um novo carro então, por uma questão de segurança, a cancela deve sempre abrir imediatamente.
- No caso de o parque estar cheio, esta situação corresponde à entrada ilegítima de um carro (acima da lotação do parque). Nesta situação deve ser acionada uma sirene, que permanecerá ligada até o botão desligarSirene ser premido (ver variáveis, abaixo).

Variáveis:

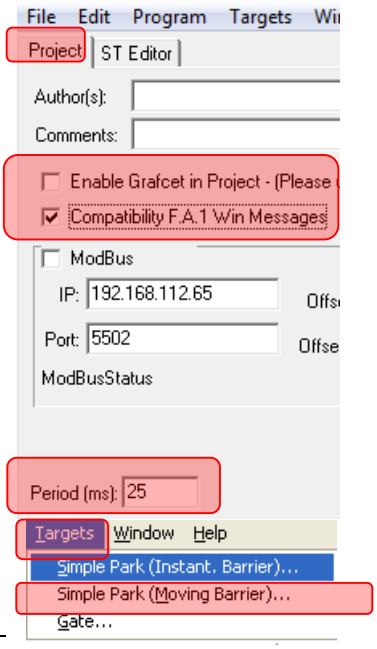
- Abra a janela *Variables* (menu *Window*) e defina o nome das variáveis de acordo com a tabela de atribuições abaixo.

Entradas	
celula1	I1
celula2	I2
celulaSaida	I3
cancelaAberta	I4
cancelaFechada	I5
desligarSirene	I6

Saídas	
abrirCancela	Q1
fecharCancela	Q2
semaforoVerde	Q3
semaforoVermelho	Q4
Sirene	Q5

Configurações:

- Confirme as configurações do projeto tal como mostrado nas figuras ao lado:
  - Tab Project: compatibilidade F.A.1
  - Tab Project: período 25 ms
  - Menu Targets: Simple Park / Moving Barrier
- Se o simulador não abrir, executar no Sistema Operativo o 3dPark.exe
- Projete, insira, compile e teste o programa de controlo do parque recorrendo às funcionalidades anteriormente apresentadas no FEUPAutom; **utilize o drive F: para gravar os projetos**



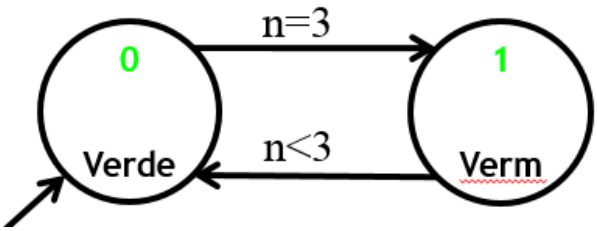
5. Final de aula, relatório, submissão e Pós-Teste

Crie um documento com o relatório de execução (minimalista) que deve conter:

1. Logo no início a identificação dos autores, nome completo turma, bancada e login
2. Relativamente à Parte 2, o número do exercício e o código da solução e o traçado temporal respetivo (utilizar ALT+PrintScreen e colar no documento) - inclua os exercícios E4 e E5
3. A digitalização do Sistema de ME do controlador do parque
4. O "copy and paste" do código gerado
5. Até 24h depois do final da aula, submeta no Moodle os ficheiros:
   - Relatório minimalista formato PDF, DOCX ou outro: **`LAB05_TXX_GYY_PrimNomeUltNomeAAA+PrimNomeUltNomeBBB.*`**
   - Ficheiro do projeto FEUPAutom do controlador do Parque: **`TP05_GXX_PrimNomeUltNomeAAA+PrimNomeUltNomeBBB.xml.FA5`**
6. Copie para si todos os ficheiros relevantes
7. Apague todo o drive F:
8. Não saia da sala sem responder ao questionário pós aula.

Bom Trabalho ☺ !

Anexo – Trecho de código – linguagem ST




```
IF    ((StateVarSmf=0) AND (n=3)) THEN
  StateVarSmf:=1;
ELSIF ((StateVarSmf=1) AND (n<3) THEN
  StateVarSmf:=0;
END_IF;

Verde := (SateSmf=0);
Verm  := (SateSmf=1);
```