Universidade do Porto
Faculdade de Engenharia
FEUP

**Mestrado Integrado em**
**Engenharia Electrotécnica e de Computadores**

# UC Sistemas e Automação

## Guião de trabalho prático

# Introdução ao software FEUPAutom

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

Versão: 14 março 2014
(TP3 Cancelas Arduino ME+Timers v7 .doc)

# Apresentação

Este trabalho prático tem por objectivo apresentar o software **FEUPAutom**, desenvolvido por docentes da FEUP* com o objectivo de apoiar o desenvolvimento de sistemas de controlo automático a eventos discretos.

Durante o decorrer do trabalho irá programar de forma Ad-Hoc e depois utilizando código obtido a partir de um modelo de uma Máquina de Estados (Diagrama de Transição de estados - DTE).

O modo de funcionamento do FEUAutom é muito semelhante aos dos autómatos programáveis pelo que, conforme irá verificar mais tarde na disciplina, é possível executar quase directamente nos autómatos programáveis programas que foram desenvolvidos no FEUPAutom.

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 objectivo 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 um parque de estacionamento (idêntico ao considerado da aula anterior).

- Na quarta parte (a realizar extra aula), vai desenvolver o programa de controlo de uma 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:

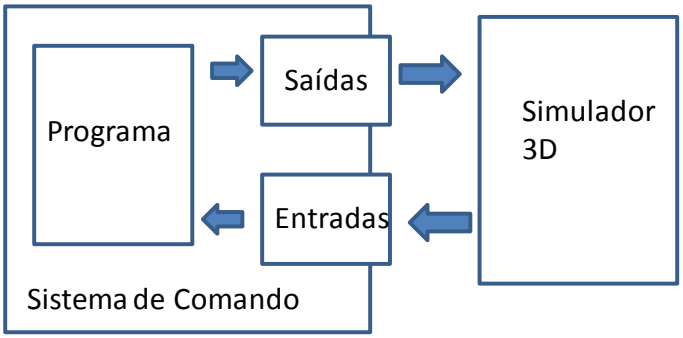
1. No estudo atento de todo este guião
2. Levar os DTEs relativos à parte 3 do guião

# Parte 1: Demonstração do FEUPAutom

O software FEUPAutom será apresentado nas aulas teóricas e está disponível para "*download*" na página da disciplina no SiFEUP. Para o executar, podera instalá-lo no seu computador pessoal ou utilizar um dos PC's do laboratório I006.

O software contém 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
- Executa o programa de controlo e determina o novo valor das saídas
- Actualizar (escreve) o valor das saídas

Por seu lado, o Simulador 3D:

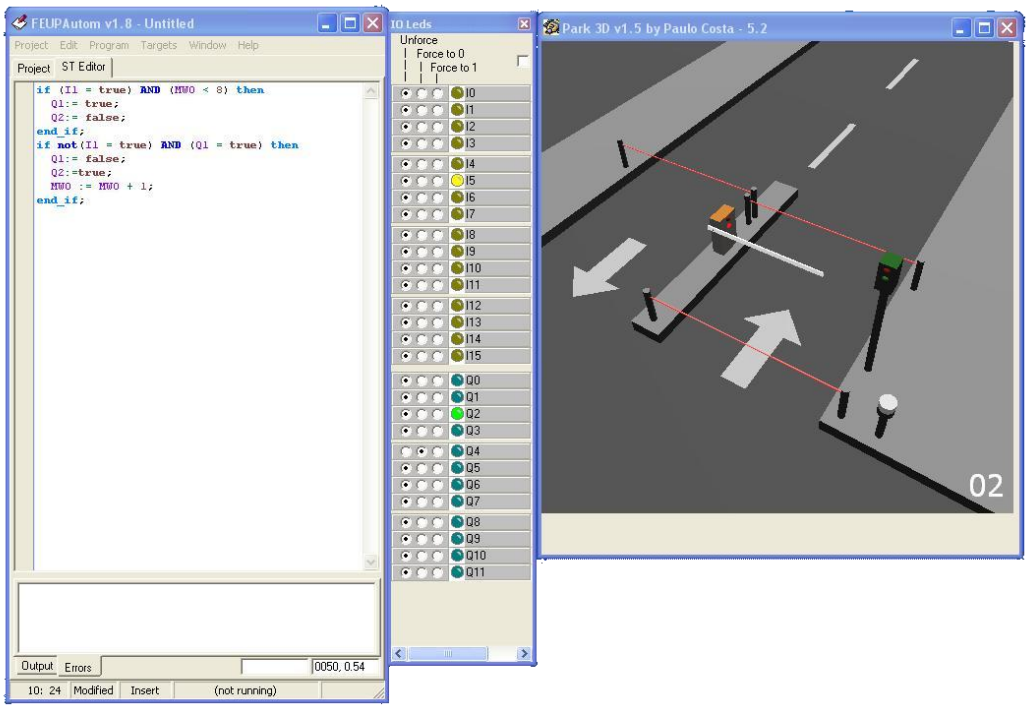
- Executa os comandos provenientes do Sistema de Comando através das linhas de saída
- Gera o valor das entradas em função da situação actual do sistema físico

Para se familizarizar 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:

- Efectue o *download* o ficheiro *FEUPAutom.zip* disponível na página da disciplina e descompacte-o para uma pasta à sua escolha
- Execute o programa *FEUPAutom.exe* presente na pasta *FEUPAutom*
- No meu *Targets* seleccione, *Simple Park (Instant Barrier)*
- No menu *Window*, active a janela *Inputs/Outputs*
- Redimensione e reposicione as janelas para ficar com uma interface próxima da seguinte:




- 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    (I1=false)  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 fotoeléctrica de entrada, o Simulador activa a entrada do Sistema de Comando correspondente a essa célula (neste caso, a entrada I1)

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

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

- O simulador detecta esta alteração nas linhas de saída e actua 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 actuada pelo que o Simulador desactiva a entrada I1.

- O programa detecta esta alteração e desactiva a saída Abrir cancela (Q1) e activa a saída fechar cancela (Q2).

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

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

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

# Parte 2: ST & FEUPAutom

Nesta segunda parte do trabalho prático vai realizar um conjunto de pequenos exercícios práticos com o objectivo 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 liguagem ST – Structured Text.

Trata-se de um linguagem standard para a programação de dispositivos automáticos (entre os quais os autómatos programáveis) que faz parte da norma IEC 61131 e que, actualmente, 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 respectivo manual de referência.

A linguagem ST descende do Pascal e, por isso:

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

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

- Escreva uma instrução por linha
- Utilize uma indentação correcta
- 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 efectuar traçados temporais 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 seleccionando *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)
- Modifique através de forçagens os valores das entradas I0 e de I1 (para isso, deve activar 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, seleccione I0,I1, Q0 e Q1


**<u>Exercício 2.1.</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 às linhas de entrada e saída (I0, I1, .., Q0, Q1, ...) e às variáveis de memória (M0, M1, ..., MW0, MW1, ...) podem ser alterados.

Para isso, deve activar 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 2.2.**

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., impôr) 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 actuar directamente 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:

- Activar 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 2.3.**

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

Efectue 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 com sinal).

**Exercício 2.4.**

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 actual programa.

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

**Exercício 2.5.**

Para testar o comportamento da variável de sistema SW0:

- Escreva a seguinte instrução no editor ST

```
println (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?

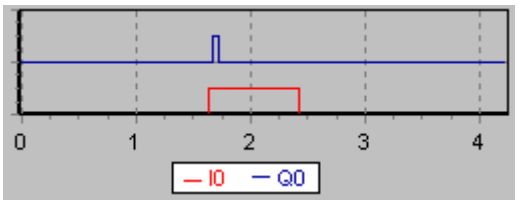
* CTRL+SHIFT+R é equivalente a menu *Program* *Clear All* e re-inicializa todas as variáveis incluindo SW0.

## 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).

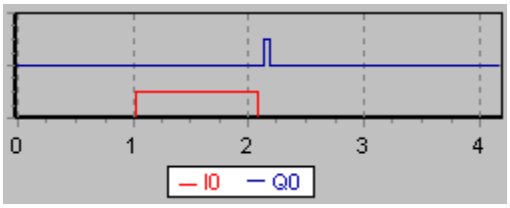
Teste a seguinte instrução visualizando o respectivo traçado temporal:

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



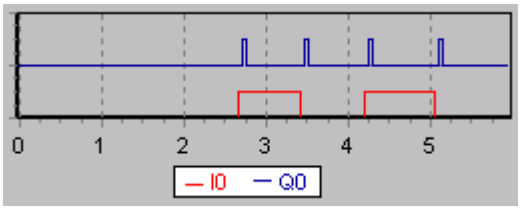

A função especial FE (*Falling Edge*) fica activa durante um ciclo do autómato apenas se a variável em causa estava a True (1) e está a False (0). Teste a seguinte instrução visualizando o respectivo traçado temporal:

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



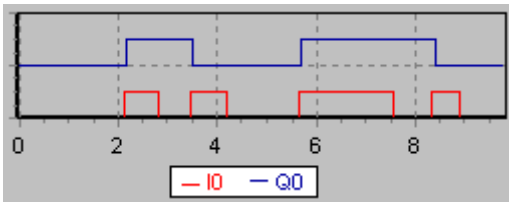

Introduza e analise o funcionamento da seguinte instrução:

```
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

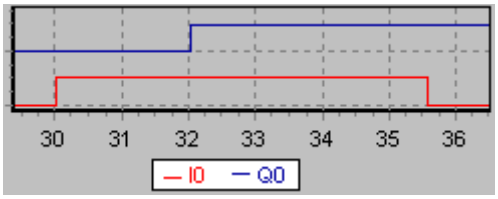
A contagem de tempo é um aspecto essencial nos programas de controlo.
Cada temporizador Txx do FEUPAutom está associado à seguinte estrutura de dados: (i) o modo de operação `Txx.mode`; (ii) o tempo pré-programado "preset" `Txx.P` em décimos de segundo e (iii) o bit de saída `Txx.Q` que indica se o tempo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

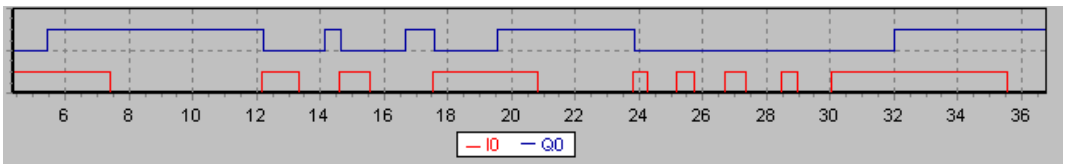
```
if RE i0 then           // inicializações
    t0.mode := TOn;     // modo de atraso à activação
    t0.P := 20;         // tempo activação em décimos de segundo
    start t0;           //  início da contagem do tempo
end_if;
q0 := t0.q;
```

**Exercício 2.6.**

Teste o programa anterior numa situação simples, tal como o 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 modo default
    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 2.91.

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

### Exercício 2.92.

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

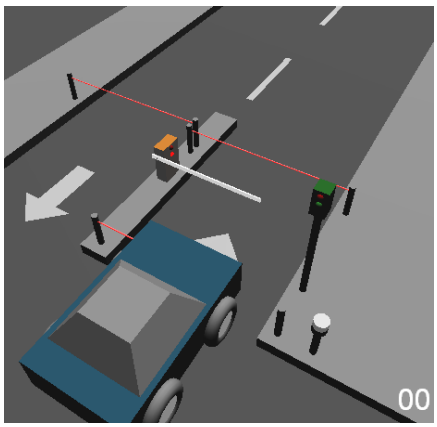
### Exercício 2.93.

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

### Exercício 2.94.

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

# Parte 3: Controlo de Parque Estacionamento – Versão simples




Considere um parque de estacionamento similar ao considerado no trabalho anterior mas, agora, no posto de entrada do parque existem duas células fotoeléctricas, uma localizada antes da cancela e a outra depois, conforme representado na figura abaixo.
Admita que o parque tem capacidade para 5 carros.

**Atenção**:

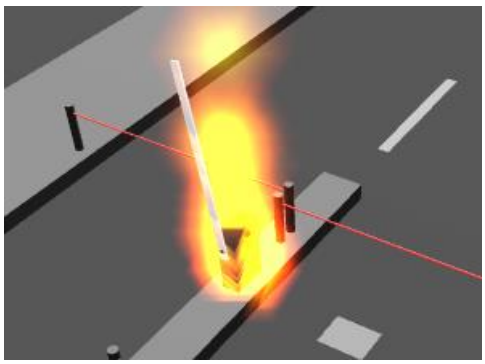
- Utilize máquinas de estados
- Utilize contadores (se entender necessário)
- Utilize temporizadores (se entender necessário)
- Pode utilizar diversas máquinas de estados (se entender necessário)
- Não pode haver código que não esteja representado no DTE da Máquina Estados

O simulador *Simple Park* reproduz o funcionamento de um parque de estacionamento com uma entrada com cancela e uma saída livre.

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

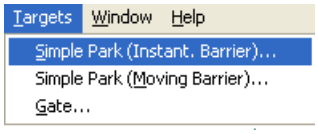
- Clicar sobre a seta de entrada para fazer entrar um carro
- Clicar sobre a seta de saída para fazer sair um carro
- Clicar sobre a sirene para activar o sensor ("Siren Off")

**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 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

## Exercício 3

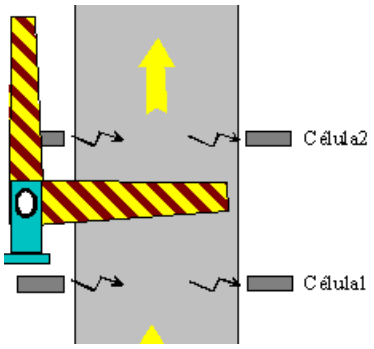

Desenvolva o programa de controlo do parque de forma que, quando um novo carro chega à entrada do parque, e no caso de haver lugares disponíveis:

- A cancela abre para o carro entrar.
- A cancela fecha depois do carro deixar de ser detectado pela célula 2 (e não após um tempo fixo como, simplificadamente, se admitia no trabalho anterior).

Para isso:

- Confirme as configurações do projeto, ver figura ao lado
- Selecione a versão *Instant Barrier* do simulador *Simple Park* (menu *Targets*)
    - Se o simulador não abrir, executar no Sistema Operativo o 3dPark.exe
- 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 |
| **Saídas** | |
| abrirCancela | Q1 |
| fecharCancela | Q2 |
| semaforoVerde | Q3 |
| semaforoVermelho | Q4 |




- Edite, compile e teste o programa de controlo do parque recorrendo às funcionalidades do FEUPAutom apresentadas na segunda parte do guião.

## Parte 4: Controlo de Parque Estacionamento – Versão completa

Nesta quarta parte do trabalho, **que poderá realizar como trabalho extra aula**, vai desenvolver uma versão um pouco mais completa do sistema de controlo do parque de estacionamento.

Relativamente à situação anterior, considere as seguintes especificações adicionais:

- Existem dois novos fim-de-curso designados por *cancelaAberta* e *cancelaFechada* que estão activos quando a cancela está completamente aberta ou fechada, respectivamente.

- Assim, a ordem de abertura da cancela deve passar a estar activa apenas enquanto o fim-de-curso *cancelaAberta* não estiver activo (caso contrário, o motor forçaria a cancela para além da sua posição limite)

- Para o fecho da cancela deve considerar uma especificação semelhante, isto é, a ordem de fecho só deve estar activa até o fim-de-curso *cancelaFechada* indicar que a cancela já está completamente fechada.

- Como anteriormente, a cancela fecha quando o carro que entrou deixar de ser detectado pela *célula2*.

- Se, antes da cancela estar completamente fechada, a *célula2* detectar a presença de um novo carro então, por uma questão de segurança, a cancela deve abrir imediatamente.

- Como esta situação corresponde à entrada ilegítima de um novo carro, deve ser accionada uma *sirene*, que permanecerá ligada até o botão *desligarSirene* ser premido.

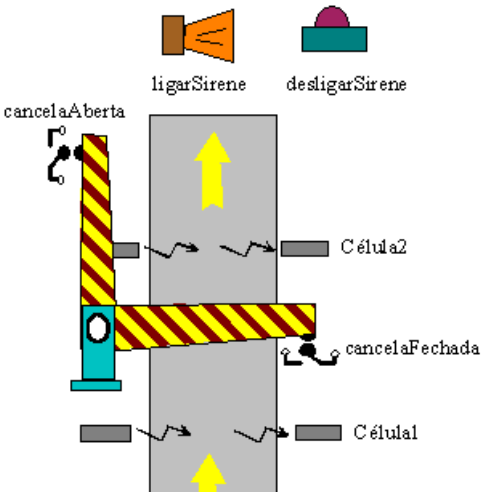
**Exercício 4**

Comece por testar o comportamento do simulador nesta nova situação actuando directamente sobre as saídas (i.e., forçando o seu valor) e observando a evolução das entradas.

De seguida, desenvolva o programa de controlo para esta nova especificação.

Para isso, utilize a versão *Moving Barrier* do simulador *Simple Park* e considere as atribuições seguintes:

| Entradas | |
|---|---|
| celula1 | I1 |
| celula2 | I2 |
| celulaSaida | I3 |
| cancelaAberta | I4 |
| cancelaFechada | I5 |
| desligarSirene | I6 |
| Saídas | |
| abrirCancela | Q1 |
| fecharCancela | Q2 |
| semaforoVerde | Q3 |
| semaforoVermelho | Q4 |
| ligarSirene | Q5 |




# 5. Final de aula, relatório, submissões e Pós-Teste

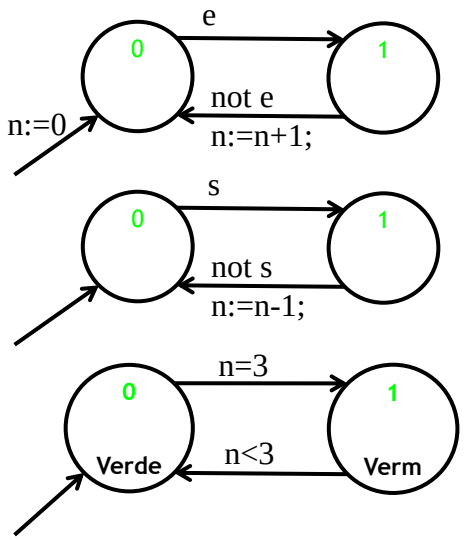
Crie um documento com o relatório de execução (minimalista) que deve conter:

1. Identificação dos autores, nome completo 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)
3. Repita para o código da Parte 3
4. Grave e submeta durante a aula o documento com o nome **`TP04A_GXX_PrimNomeUltNomeAAA+PrimNomeUltNomeBBB.PDF`**
5. Até à aula seguinte, submeta um novo "relatório" só com o exercício 4 num ficheiro **`TP04b_GXX_PrimNomeUltNomeAAA+PrimNomeUltNomeBBB.PDF`**

Não saia da sala sem responder ao questionário do moodle (questione o professor acerca da password).

Bom Trabalho ☺ !

# Anexo – exemplo Múltiplos DTEs com contadores




```
// Neste caso não é necessária nenhuma inicialização

 IF    ((StateE=0) AND E) THEN
   StateE:=1;
 ELSIF ((StateE=1) AND (not E)) THEN
   StateE:=0;
   n:=n+1;
 END_IF;


 IF    ((StateS=0) AND S) THEN
   StateS:=1;
 ELSIF ((StateS=1) AND (not S)) THEN
   StateS:=0;
   n:=n-1;
 END_IF;


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

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