

## U.C. Sistemas e Automação

### Trabalho Prático 3 (TP3):

Projeto de sistema de controlo de  
Cancelas baseado em Arduino  
Com recurso a Máquinas de Estado

Armando Jorge Sousa – [asousa@fe.up.pt](mailto:asousa@fe.up.pt)

Luís Almeida – [lda@fe.up.pt](mailto:lda@fe.up.pt)

Paulo Costa – [paco@fe.up.pt](mailto:paco@fe.up.pt)

# 1. Apresentação do Trabalho Prático

Este trabalho continua o Trabalho Prático 1 (TP1) onde se utilizou uma linguagem próxima da linguagem C para projetar e implementar o sistema de controlo de garagens utilizando “Arduino”.

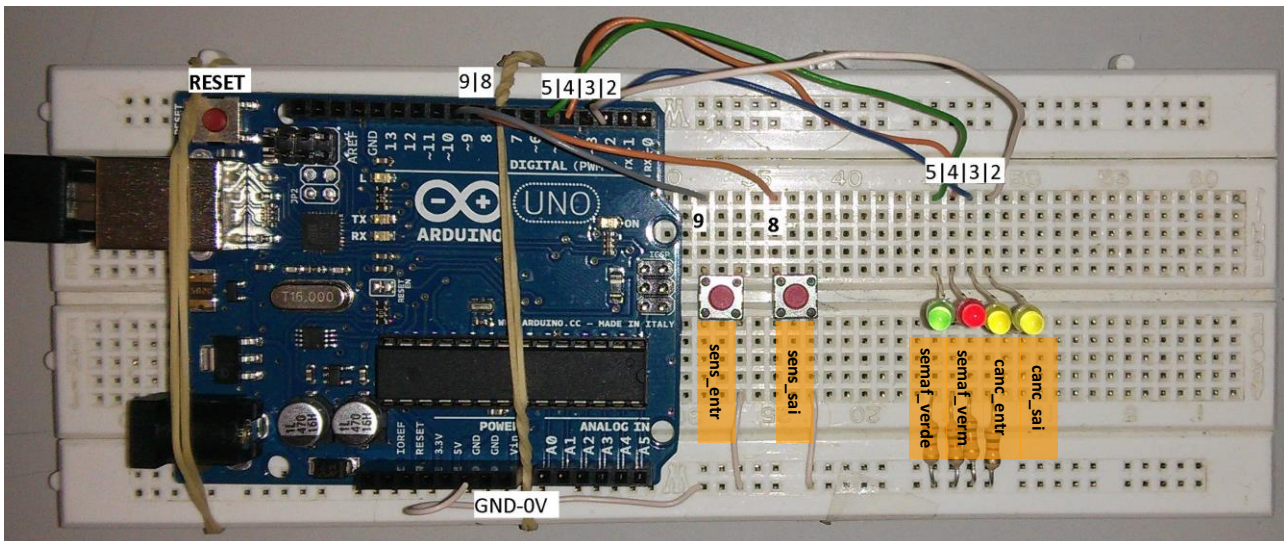


Figura 1a – Placa de montagem com Arduino

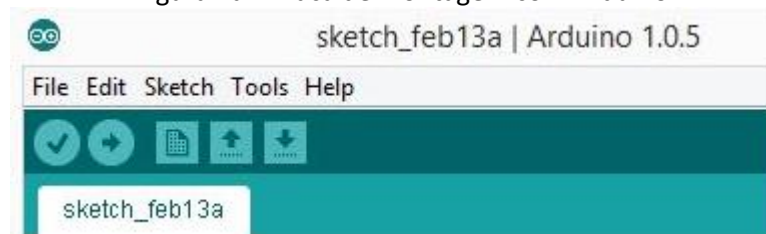


Figura 1b – IDE Arduino

## Objetivos do trabalho TP3:

- Responder ao caderno de encargos com recurso às Maquinas de Estado (ME)
- Programar uC da placa Arduino em linguagem similar à linguagem C
- Tomar contacto com dificuldades de problemas multitarefa e temporizações

## Atenção:

Leia todo o guião antes de iniciar o seu trabalho e no final da aula submeta os elementos produzidos.

# 2. Preparação da aula

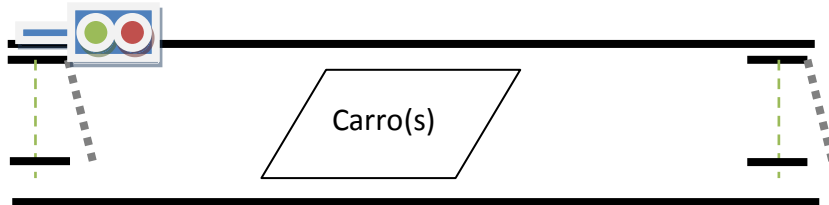
A preparação deste Trabalho Prático (TP3) inclui ler todo o guião **cuidadosamente** e desenhar os DTEs, levando-os para a aula em papel. É ainda necessário entender a forma de implementar a máquina de estados no ambiente apresentado, incluindo temporizadores e contadores. Opcionalmente os estudantes podem já implementar os cadernos de encargos aqui apresentados no ambiente do TP1, a IDE do Arduino que pode ser descarregado no seguinte endereço: <http://arduino.googlecode.com/files/arduino-1.0.5-r2-windows.zip>

### 3. Caderno de encargos

Considere o problema simplificado de um sistema que controla o acesso a um (ou mais) lugar(es) de estacionamento privilegiado(s) mas partilhado por diversas pessoas protegido por cancelas.

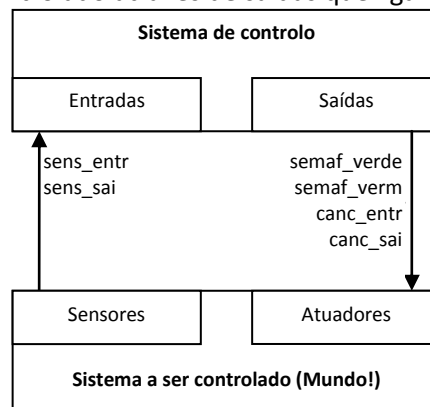
O acesso é conseguido através da passagem numa porta automática de entrada e a saída está também protegida por uma porta automática. Há um sensor de entrada dos carros que indica que um carro pretende entrar e o mesmo para outro sensor de saída, indicando que um carro pretende sair.

Adicionalmente há um semáforo apenas com uma luz verde e outra vermelha. A luz verde indica estacionamento livre e a vermelha estacionamento ocupado.



### 4. Conceitos e problema sob análise

Relembre que qualquer sistema de controlo/comando visa comandar um determinado sistema real (sistema a ser controlado). O sistema de comando recebe informação através de entradas onde ligam sensores e altera o sistema a ser controlado através de saídas que ligam a atuadores.






Para resolver o problema proposto temos:

**Atenção:** utiliza-se lógica negada em todos as entradas

- Sensor de Carro à Entrada => Pino\_9 => variável "sens\_entr"
  - É "0" quando o carro está presente (um carro quer entrar) = premir botão
  - É "1" quando não está nenhum carro presente = botão no estado de repouso
- Sensor de Carro à Saída => Pino\_8 => variável "sens\_sai"
  - É "0" quando o carro está presente (um carro quer sair)
  - É "1" quando não se deteta presença de carro no sensor
- Atuador da Cancela de Entrada => Pino\_3 => variável "canc\_entr"
  - Quando este atuador é posto a "1"="HIGH" o LED liga e a cancela de entrada levanta
  - Quando é posto a "0"="Low" o LED desliga e a cancela não permite entrada de carros
  - Depois de a cancela levantar, deve-se esperar até o carro "desaparecer" do sensor de entrada para descer
- Um Atuador da Cancela de Saída => Pino\_2 => variável "canc\_sai"
  - Quando este atuador é ligado, isto é, posto a "1", a cancela de saída levanta
  - Quando está a "0", a cancela desce e não permite a saída de carros
  - Depois de a cancela levantar, deve esperar até ao carro "desaparecer" do sensor de entrada para descer

Adicionalmente, estude o funcionamento do semáforo que tem luz vermelha (Pino\_4) e luz verde (Pino\_5).

## 5. Instalação e teste inicial

1. Vá ao Moodle da UC e faça *download* do \*.zip do TP3
2. Crie uma diretoria com o seu nome para o seu trabalho e desempacote aí o ficheiro retirado do Moodle (mantendo as subdiretorias).
3. Abra a IDE do Arduino e faça open do ficheiro TP3
4. Na IDE clique no 'certo'  ou prima CTRL+R em cima a esquerda (compila o programa) – confirme que o programa não tem erros
5. De seguida clique na 'seta'  ou prima CTRL+U (descarregar o programa para o uC).
6. Verifique que todos os LEDs acendem durante 1 segundo
7. Na IDE clique na 'lupa'  ou prima CTRL+Shift+M para abrir o “monitor” (consola) que lhe permite ver o que o Arduino escreveu; configure o Baud Rate para 115200 (canto inferior direito)
8. Verifique o bom funcionamento do programa de teste

## 5. Sequência de passos para o trabalho

Para cada um dos seguintes pontos, implemente e teste. Não espere pelo professor, prossiga o seu trabalho, aproveite bem o tempo de aula.

**Atenção:** Guarde separadamente as respostas de cada alínea. Guarde os ficheiros .ino recorrendo ao IDE na opção “Save As” no menu “File” para assim evitar problemas nos ficheiros à posteriori.

Passos para a solução completa:

- a. Entrada do carro: Inicialmente considere apenas a entrada de um único carro no estacionamento; a cancela de entrada abre quando o carro é detetado na entrada; depois do carro entrar, o semáforo vermelho liga; teste e confirme a solução;  
Saída do Carro: Considere a posterior saída do carro no estacionamento; o semáforo verde liga depois do carro ter deixado de ser detetado na saída; admita que não entram e saem carros ao mesmo tempo; garanta que quando o lugar está ocupado, a cancela não abre;  
Na IDE, guarde o ficheiro referente a esta alínea com o seguinte nome:  
**TP03A\_GXX\_PrimNomeUltNomeAAA+PrimNomeUltNomeBBB.ino**
- b. Altere a solução anterior para que ambas as cancelas tenham um tempo mínimo de abertura de 5 segundos; **Atenção:** Utilize os timers  
Na IDE, guarde o ficheiro referente a esta alínea com o seguinte nome:  
**TP03B\_GXX\_PrimNomeUltNomeAAA+PrimNomeUltNomeBBB.ino**
- c. Admita agora que o estacionamento tem capacidade para 5 lugares; Desenhe uma nova solução para este problema; Na IDE, guarde o ficheiro referente a esta alínea com o seguinte nome:  
**TP03C\_GXX\_PrimNomeUltNomeAAA+PrimNomeUltNomeBBB.ino**

**Notas:** Substituir XX pelo número da bancada; Substituir PrimNomeUltNomeAAA pelo primeiro e último nome do primeiro autor por ordem alfabética do 1º nome; substituir PrimNomeUltNomeBBB de forma equivalente para o segundo autor

## 6. Final de aula – submissão e questionário

No final da aula submeta os 3 ficheiros \*.ino no moodle (ou então todos os ficheiros até então conseguidos).

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

## 7. Código base

Para preparação da aula, analise o seguinte código e todos os conceitos associados, explicados nos comentários ao código.

```
#include "sa.h"

// Definir pinos atribuídos a cada variável
const int pino_sens_entr    = 9; // Entrada
const int pino_sens_sai     = 8; // Entrada
const int pino_semaf_verde  = 5; // Saída
const int pino_semaf_vermelho = 4; // Saída
const int pino_canc_entr    = 3; // Saída
const int pino_canc_sai     = 2; // Saída

const int total_timers = 8;
timer_t timer[total_timers];

// Estas variáveis são atualizadas automaticamente no início do ciclo
byte sens_entr;
byte sens_sai;

// Estas variáveis são escritas efetivamente para as saídas no final do ciclo
byte semaf_verde;
byte semaf_vermelho;
byte canc_entr;
byte canc_sai;

// Atualizar os timer para refletirem o seu valor atual durante o ciclo de controlo
void refresh_timers(void)
{
    byte i;
    for(i = 0; i < total_timers; i++)
        refresh_timer(timer[i]);
}

// Ler e atualizar as variáveis correspondentes as entradas
void read_inputs(void)
{
    sens_entr = digitalRead(pino_sens_entr);
    sens_sai  = digitalRead(pino_sens_sai);
}

// Ler e atualizar as variáveis correspondentes as saídas
void write_outputs(void)
{
    digitalWrite(pino_semaf_verde, semaf_verde);
    digitalWrite(pino_semaf_vermelho, semaf_vermelho);
    digitalWrite(pino_canc_entr, canc_entr);
    digitalWrite(pino_canc_sai, canc_sai);
}

long previousMillis = 0;

// Sequencia de código que corre em ciclo, NÃO ALTERAR
void loop()
{
    unsigned long currentMillis = millis();
    if(currentMillis - previousMillis > 10) { // 10 ms
        refresh_timers();
        read_inputs();
        loop_10ms();
        write_outputs();
    }
}
```

```

// Corre aquado do reset (uma única vez)
void setup()
{
    Serial.begin(115200); // Configura a velocidade das comunicações porta serie

    // Definição de pinos como Entradas/Saídas.
    pinMode(pino_semaf_verde, OUTPUT);
    pinMode(pino_semaf_vermelho, OUTPUT);
    pinMode(pino_canc_entr, OUTPUT);
    pinMode(pino_canc_sai, OUTPUT);
    pinMode(pino_sens_entr, INPUT_PULLUP);
    pinMode(pino_sens_sai, INPUT_PULLUP);

    digitalWrite(pino_semaf_verde, HIGH);
    digitalWrite(pino_semaf_vermelho, HIGH);
    digitalWrite(pino_canc_entr, HIGH);
    digitalWrite(pino_canc_sai, HIGH);
    delay(500); //Espera 500ms (0.5s)

    digitalWrite(pino_semaf_verde, LOW);
    digitalWrite(pino_semaf_vermelho, LOW);
    digitalWrite(pino_canc_entr, LOW);
    digitalWrite(pino_canc_sai, LOW);
    delay(500); //Espera 500ms

    timer[0].p = 10; // 1 segundo
}

////////////////////////////////////

byte estado = 0; // Variável Global

// A função seguinte é chamada a cada 10ms

void loop_10ms(void) // Alterar apenas o código desta função
{
    // Calcular o próximo estado
    if ((estado == 0) && (sens_entr == 0)) {
        estado = 1;
        start_timer(timer[0]);
    } else if ((estado == 1) && (timer[0].q == 1)) {
        estado = 0;
    }

    // Calcular as saídas
    semaf_verde = (estado == 0);
    semaf_vermelho = (estado == 1);

    // Comunicar dados relevantes
    Serial.print (estado);
    Serial.print (" ");
    Serial.print (sens_entr);
    Serial.println(sens_sai);
}

```

- Fim do Guião TP1 -