

U.C. Sistemas e Automação

Lab03

Projeto de sistema de controlo de cancelas baseado em Arduino utilizando Sistemas de Máquinas de Estado

Armando Jorge Sousa – asousa@fe.up.pt

Luís Almeida – lda@fe.up.pt

Paulo Costa – paco@fe.up.pt

1. Apresentação do Trabalho Prático

Este trabalho contruirá um controlador para um parque de estacionamento simplificado utilizando um Sistemas de Máquinas de Estado (de automação, com tempos e contagens).

Preparação

Antes da aula, individualmente ou em grupo, é necessário:

- Estudar todo o guião;
- Saber implementar uma máquina de estado e um sistema de máquinas de estado;
- Levar os DTEs a implementar (em papel).

Software

- O ambiente de programação Arduino está disponível gratuitamente na web <https://www.arduino.cc/en/Main/Software>
- Opcionalmente, ao critério do estudante é possível utilizar o software gratuito <https://www.tinkercad.com/circuits> para ajudar no desenvolvimento

Hardware

O kit laboratorial é o mesmo do TP1, Arduino Uno, botões e LEDs

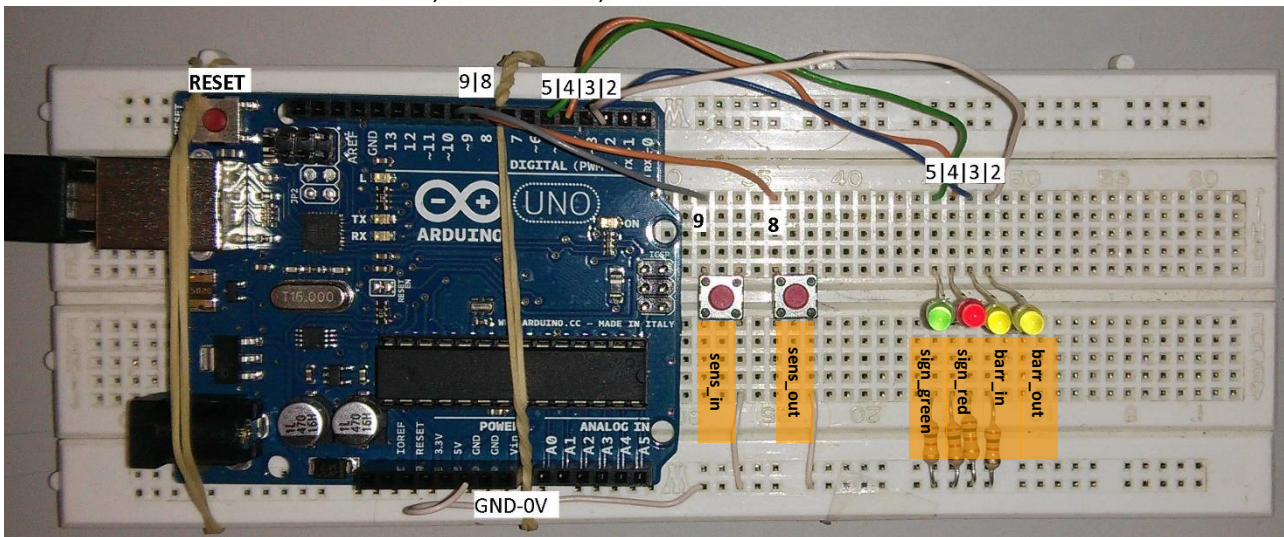


Figura 1 – Placa de montagem com Arduino

Objetivos

- Implementação em Arduino_C de um controlador modelizado como um Sistema de Máquinas de Estados (com temporizadas e com contadores)
- Tomar contacto com dificuldades de problemas envolvendo paralelismo
- Necessidade/vantagens de operações na transição entre estados

2. 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 e protegido por cancelas.

O acesso é conseguido através da passagem numa cancela automática de entrada; também a saída está também protegida por uma cancela 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.

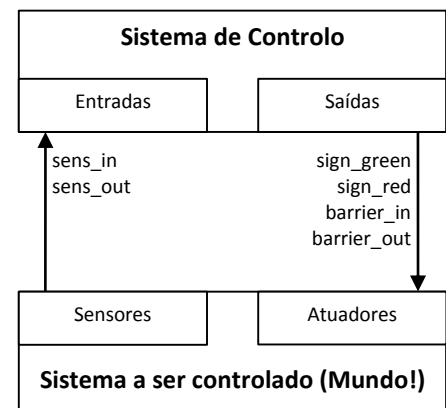
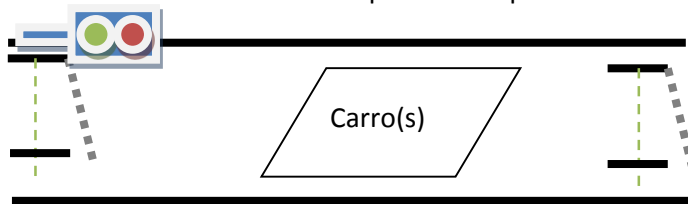
Adicionalmente há um “semáforo” com luzes verde e vermelha.

Quando um carro se apresenta para entrar (sens_in ativo) a “cancela de entrada” deve passar à posição de aberta (barr_in ativo) até ao carro passar (quando sens_in ficar inativo, barr_in deve ser desligado).

O mesmo para a saída: quando um carro se apresenta para sair (sens_out ativo) a “cancela de entrada” passa à posição de aberta (barr_out ativo) até ao carro passar (quando sens_out ficar inativo, barr_out deve ser desligado).

Admitindo que o parque não está cheio, o verde (sign_green) deve estar ativo. Nessa situação podem entrar e sair carros em simultâneo, a qualquer ritmo.

Quando o parque ficar cheio, o semáforo deve ficar vermelho (sign_red) e o controlador deve impedir a entrada de novos carros. Os carros podem sempre sair.



3. Conceitos para acesso ao Hardware

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.

Relativamente a sensores / entradas: Utilize as variáveis de lógica positiva sens_in e sens_out.


Relativamente a atuadores / saídas: Utilize as variáveis sign_red, sign_green, barr_in e barr_out.

4. Instalação e teste inicial

1.1 Vá ao Moodle da UC e faça *download* do *.zip do lab03


1.2 Crie uma diretoria com o seu nome em F: e desempacote aí o ficheiro retirado do Moodle (mantendo as subdiretorias)

1.3 Abra o IDE do Arduino e faça open do ficheiro lab03

1.4 No IDE clique no ‘certo’  ou prima CTRL+R em cima a esquerda (compila o programa) – confirme que o programa não tem erros; se ocorrer o erro relativo à falta do ficheiro TimerOne, instale essa biblioteca em falta:
Sketch -> Include Library -> Manage Libraries -> Search "timerone" -> more info -> install

1.5 De seguida clique na ‘seta’  ou prima CTRL+U (descarregar o programa para o uC).

1.6 Verifique que todos os Leds acendem durante 1 segundo

1.7 No 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)

1.8 Prima RESET e verifique que todos os LEDs acendem

5. Sequência de passos para o trabalho

Atenção:

- Guarde separadamente as respostas de cada alínea copiando o lab03.ino para o nome certo
- Não se esqueça de comentar e ou apagar o código de teste
- No final da aula guarde os ficheiros para si
- Apague todo o drive F:

Passos para a solução completa:

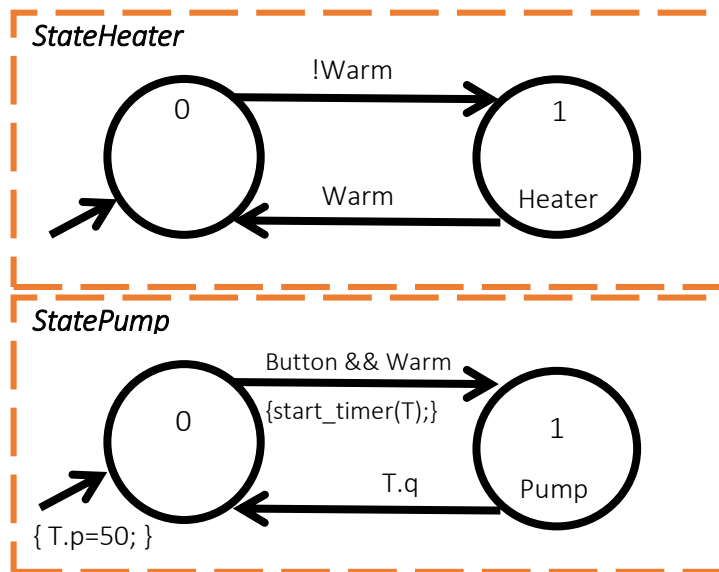
1. Confirme as ligações elétricas do seu kit de hardware
2. Estude os anexos deste guião: é obrigatório a implementação respeitar as normas de produção sistemática de código das máquinas de estado
3. Identifique a função loop_10ms() onde deverá escrever o seu código
4. Projete e implemente o sistema de máquinas de estados para os pontos que se seguem:
5. **Sistema_Máquina_Estados_A**
 - Coloque a capacidade do parque de estacionamento numa constante do seu programa;
 - Controle o parque de estacionamento com a capacidade de 4 lugares (utilizando a constante definida acima);
 - Teste diversas entradas e saídas de carros em sequência e em simultâneo; confirme também o funcionamento do semáforo e o comportamento com o parque cheio.
 - No sistema operativo, copie este ficheiro **lab03.ino** para **LAB03A_TXX_GYY_PrimNomeUltNomeAAA_PrimNomeUltNomeBBB.ino**
6. **Sistema_Máquina_Estados_B**
 - Altere a solução anterior para que ambas as cancelas tenham um tempo mínimo de abertura de 5 segundos; utilize os temporizadores fornecidos;
 - No sistema operativo, copie este ficheiro **lab03.ino** para **LAB03B_TXX_GYY_PrimNomeUltNomeAAA_PrimNomeUltNomeBBB.ino**

6. Final de aula – submissão e questionário

Valide a sua solução e submeta os 2 ficheiros *.ino no moodle até 24 horas depois da aula.

Não saia da sala sem responder ao questionário (questione o professor).

7. Anexo – Implementação exemplo de Sistema de Máquinas de Estados



```
// Assume both States are 0 after reset
// Assume T is a timer variable and, at setup, T.p=50
```

```
if ((StateHeater==0) && !Warm) then
    StateHeater = 1;
elseif ((StateHeater==1) && Warm) then
    StateHeater = 0;
end_if;

Heater = (StateHeater==1);

if ((StatePump==0) && Button && Warm) then
    StatePump = 1;
    start_timer(T);
elseif ((StatePump==1) && (T.q)) then
    StatePump = 0;
end_if;

Pump = (StatePump==1);
```

8. Anexo – código do projeto base

```
////////////////////////////////////
// Sistemas e Automação //
// (C) FEUP paco@fe.up.pt and asousa@fe.up.pt //
////////////////////////////////////

// Define 8 global timers
const int total_timers = 8;
timer_t timer[total_timers];

// To be updated by read_inputs at start of cycle
byte sens_in;
byte sens_out;

// To be updated by write_outputs at end of cycle
byte sign_red;
byte sign_green;
byte barr_in;
byte barr_out;

// Input PINs =====> Rectify with your own pins if needed
const int pin_sens_in = 9;
const int pin_sens_out = 8;

// OUTPUT PINs =====> Rectify with your own pins if needed
const int pin_green = 5;
const int pin_red = 4;
const int pin_barr_in = 3;
const int pin_barr_out = 2;

// Run Once After Reset or PowerUp (to setup hardware, etc)
void setup()
{
    // Serial Port Speed
    Serial.begin(115200);

    // Init message
    Serial.println("(C)FEUP S&A - start init");

    // ...

    // Setup Timers
    timer[0].p = 20; // 2 seconds
    start_timer(timer[0]);
    timer[1].p = 5; // 0.5 seconds
    start_timer(timer[1]);

    // ...
} // End of Setup

// Update timers
void refresh_timers(void)
{
    byte i;
    for(i = 0; i < total_timers; i++)
        refresh_timer(timer[i]);
} // End of refresh_timers()

// Read from pins to positive "Image_Variables"
void read_inputs(void)
{
    if (digitalRead(sens_in_pin) == LOW) sens_in = 1; else sens_in = 0;
    if (digitalRead(sens_out_pin) == LOW) sens_out = 1; else sens_out = 0;
} // End of read_inputs()

// Write to pins from "Image_Variables"
void write_outputs(void)
{
    digitalWrite(pin_green, sign_green);
    digitalWrite(pin_red, sign_red);
    digitalWrite(pin_barr_in, barr_in);
    digitalWrite(pin_barr_out, barr_out);
} // End of write_inputs

// Global Variables for S&A stuff - do not touch
long previousMicros = 0;

// Arduino Loop - do not touch
void loop()
{
    unsigned long currentMicros = micros();
    if(currentMicros - previousMicros > 10000) { // 10 ms

        previousMicros = currentMicros;

        refresh_timers();
        read_inputs();

        loop_10ms();

        write_outputs();
    }
} // End of Arduino loop

////////////////////////////////////
////////// Write Your Code below this line //////////
////////////////////////////////////

// Global Variables, define additional global vars below
byte state1 = 0;
byte state2 = 0;

// Loop where you should write your state machine code, etc
void loop_10ms(void)
{
    // Don't forget to comment or delete example code
    // Don't read nor write directly hardware pins

    // Example System of State machines with 2 independent
    state machines

    // Example StateMachine1 - StateVar => state1
    if ((state1 == 0) && (sens_out)) {
        state1 = 1;
        start_timer(timer[0]);
    } else if ((state1 == 1) && timer[0].q) {
        state1 = 0;
    } // end of calculation of next state for state1

    // Example StateMachine2 - StateVar => state2
    if ((state2 == 0) && (sens_in)) {
        state2 = 1;
        start_timer(timer[1]);
    } else if ((state2 == 1) && timer[1].q) {
        state2 = 0;
    } // end of calculation of next state for state2

    // Calculate Outputs
    sign_red = (state1 == 0);
    sign_green = (state1 == 1);
    barr_in = (state2 == 0);
    barr_out = (state2 == 1);

    // Debug - print states and variables, change as you wish!
    Serial.println(
        "St1:" + ((String) state1) +
        "St2:" + ((String) state2) +
        "Sens:" + ((String) sens_in ) + ((String) sens_out ) +
        "Sign:" + ((String) sign_green) + ((String) sign_red) +
        "Barr:" + ((String) barr_in) + ((String) barr_out));
} // End of loop_10ms
```

- Fim do Guião TP3 -