Laboratórios de Informática 2014/2015
DETI - Universidade de Aveiro

**Docentes**
João Paulo Barraca <jpbarraca@ua.pt>
Diogo Gomes <dgomes@ua.pt>
João Manuel Rodrigues <jmr@ua.pt>
Mário Antunes <mario.antunes@ua.pt>

TEMA 22

# Micro-controladores e a plataforma MicroRato

**Objetivos:**
- Conhecer o modelo de Programação do MicroRato
- Conhecer as incertezas associadas a sensores analógicos
- Algoritmos para evitar obstáculos
- Algoritmos para seguir linhas

## 22.1 Introdução

O MicroRato é um sistema robótico desenvolvido na Universidade de Aveiro, tendo o mesmo nome da competição de robótica que o Departamento de Electrónica, Telecomunicações e Informática criou em 1995. O objetivo do concurso é o de atingir o centro de um labirinto, recorrendo apenas às capacidades sensoriais e de processamento presentes nos pequenos robots. A Figura 22.1 representa parte desse labirinto, exatamente no

As plataformas robóticas MicroRato são compostas por um conjunto de sensores e atuadores, podendo ou não ter um sensor de farol. Estes sensores são descritos nas secções seguintes.

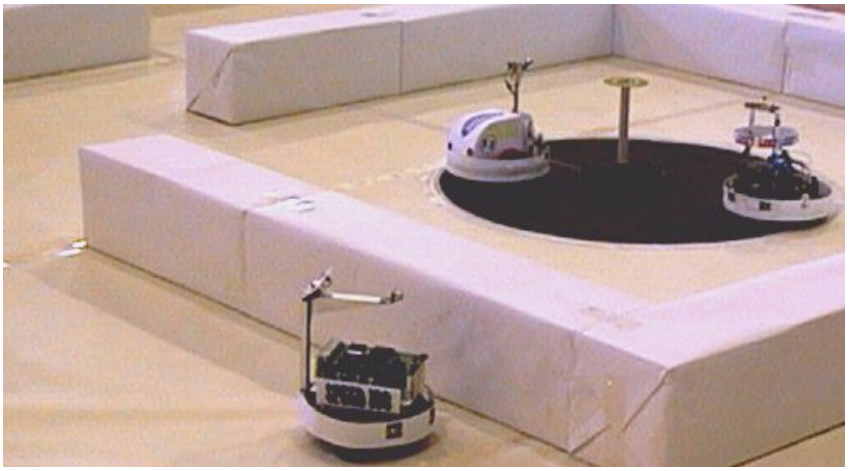

Figura 22.1: Final de uma prova do Micro Rato

Todo o processamento é realizado por um micro-controlador PIC32MX795F512H[1]. De notar que não se trata de um microprocessador de uso genérico como os encontrados nos computadores portáteis. Tanto os micro-controladores como os microprocessadores processam instruções que implementam um dado algoritmo que o programador criou e são muito semelhantes. No entanto existem diferenças muito importantes entre eles.

Os microprocessadores são compostos por um ou mais núcleos e eventualmente alguma memória *cache* interna ou um controlador de memória. Em casos específicos podem incluir um processador gráfico. Estes processadores não possuem qualquer Random Access Memory (RAM) ou Read Only Memory (ROM), mas são integrados em sistemas onde o designer tem a responsabilidade de desenhar uma placa mãe que permita acesso a estes recursos. O resultado é um computador comum como os dos laboratórios ou um computador portátil.

Um micro-controlador também possui um processador, mas possui igualmente RAM, ROM e mesmo outros sistemas eletrónicos, tudo no mesmo circuito integrado. Um exemplo de um sistemas eletrónicos frequentemente integrados nos micro-controladores são memórias flash ou Static Random Access Memory (SRAM), Analog to Digital Converter (ADC), interfaces Serial Peripheral Interface (SPI), Controller Area Network (CAN) ou Inter-Integrated Circuit ($I^2C$). Nada disto existe num microprocessador comum. Além disto, os micro-controladores são desenhados para tarefas mais simples, baseadas em entradas e saídas de dados. Por exemplo para processar dados de teclados, controlar máquinas de lavar ou outros eletrodomésticos. Estes sistemas necessitam de uma quantidade de recursos muito pequena e tudo pode ser incluído no mesmo circuito integrado. Um

[1]Especificações em `http://goo.gl/4LkFps`

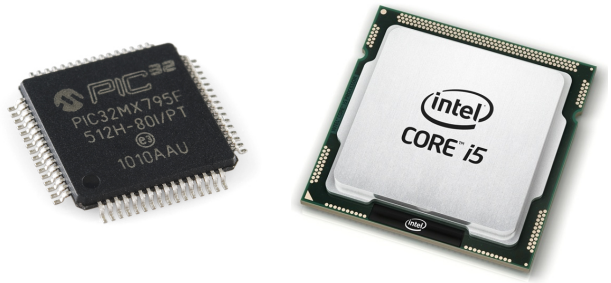


Figura 22.2: O Microchip PIC32MX795F512H e o Intel i5. Imagens pertencem aos seus fabricantes.

exemplo de um sistema muito popular construido com micro-controladores é o Arduino, que utiliza vários micro-controladores, sendo o mais famoso o ATMEGA328P a 8 MHz ou 16 MHz.

Este micro-controlador possui características específicas que estão relacionadas com os números de definem o seu modelo. Por isso o modelo é complexo, ao contrário dos microprocessadores comercializados como os Intel i5. A Figura 22.1 apresenta ambos. De notar que o Intel i5 mede 37.5 mm, enquanto o PIC32 mede 10 mm de lado, sendo portanto 7.5 vezes mais pequeno.

Neste caso as características relevantes deste micro-controlador são as seguintes:

· CPU 32bits MIPS M4K a 80Mhz

· Interface USB 2.0

· Interface Ethernet 10/100

· 2 Interfaces CAN

· 512KB de Flash

· 128KB de RAM

· ADC de 16 canais com 10 bits de resolução a 1MSps [2]

· RTC Interno

Uma diferença bastante grande é a velocidade do CPU (80 MHz), a quantidade de RAM (128 KiB) e a quantidade de memória flash (512 KiB). Num computador portátil

[2] *Mega Samples Per Second* ou Milhões de Leituras por Segundo.

o CPU funciona a vários GHz, existem vários GiB de RAM e temos discos rígidos com vários TiB de capacidade.

Por este motivo o modo de funcionamento e desenvolvimento para estes sistemas é bastante diferente do encontrado nos microprocessadores.

## 22.2 Linguagem *C*

No caso destes micro-controladores, a linguagem *Python* é deveras desadequada por o sistema não tem capacidade de armazenamento ou processamento compatíveis. Pode experimentar criar no seu computador um programa em *Python* muito simples (um `Hello World`), podendo ver que a quantidade de RAM necessária para a execução deste programa será bastante superior à disponível pelo micro-controlador.

Por estes motivos os micro-controladores são tipicamente programados usando linguagens de mais baixo nível como as linguagens Assembler e *C*. Nesta aula vamos utilizar *C* visto que a linguagem Assembler é deveras mais complexa para os problemas que tentaremos resolver.

A linguagem *C* não é interpretada como a *Python* e os programas resultantes não são executados dentro de uma máquina virtual, como a linguagem *Java*. Este guia não irá abordar os detalhes da linguagem, focando-se no mínimo necessário para implementar um programa para a plataforma MicroRato.

Do ponto de vista sintático a linguagem *C* é bastante semelhante à linguagem Java, sendo que os ciclos, condições e declarações de variáveis ou funções são bastante semelhantes. O código seguinte implementa o típico `Hello World`, podendo identificar-se comentários, declaração de uma função com argumentos, a inclusão de bibliotecas, assim como impressão de texto para a consola.

```c
/*
 * Ficheiro hello.c
 */
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char* argv[]){
    printf("Hello World!\n");

    return 0;
}
```

No exemplo o caráter `*` é um conceito novo, representando um ponteiro. Ou seja variáveis que em vez de possuírem um conteúdo, possuem o endereço desse conteúdo. No caso em questão a variável `argv` é um `Array` para ponteiros de `char`. Isto acontece porque na linguagem *C* não existe o tipo `String`, sendo que um texto é representado por um tipo `char *`. Portanto, `argv` contém um `Array` de `Strings` que são os argumentos passados ao programa.

### Exercício 22.1

Estas instruções podem ser convertidas num programa e depois executadas através dos comandos `gcc -o hello hello.c` e depois `./hello`.

Teste a compilação e execução deste pequeno programa.

A sintaxe dos ciclos `for`, `do` ou `while`, condições como `if` é exatamente igual à linguagem *Java*, tal como apresentado no pequeno programa seguinte. Este programa tem uma variável inteira $i$ que toma valores de 0 a 9, sendo imprimido o valor do fatorial para cada um dos valores. O objetivo é demonstrar algumas estruturas básicas da linguagem C.

```c
#include <stdlib.h>
#include <stdio.h>

int factorial(n){
    if(n == 0)
        return 1;

    return n*factorial(n-1);
}

int main(int argc, char* argv[]){
    int i;
    for(i=0; i < 10; i++){
        printf("Factorial %d = %d\n", i, factorial(i));
    }
}
```

### Exercício 22.2

Crie um ficheiro chamado `factorial.c`, compile-o com o compilador `gcc` e verifique a sua execução.

Os programas criados até agora executam no seu computador mas, usando princípios muito semelhantes, podem ser construídos programas que são executados pelo MicroRato. Neste caso será necessário utilizar um compilador que produza código com as instruções. Este compilador encontra-se na página da disciplina, devendo ser obtido e descompactado para o diretório `/opt/pic32mx` de forma a que existam diretórios `/opt/pic32mx/bin`, `/opt/pic32mx/lib`, `/opt/pic32mx/include` e por aí fora.

O processo de criação de um programa implica a compilação, conversão do programa para um formato capaz de ser transferido e depois a programação do micro-controlador. A compilação faz-se com o programa `pic32-gcc`, a conversão para o formato hexadecimal para transferência faz-se com o programa `pic32-bin2hex`. Finalmente a transferência faz-se com o programa `ldpic32`. Todos estes programas acompanham o compilador. Após a execução, o comando `pterm` criar um terminal de texto para interagir com o programa. A título de exemplo, realizando um `printf` irá apresentar esse resultado no terminal criado com o `pterm`.

**Exercício 22.3**

Verifique se o compilador existe no seu sistema e caso não exista obtenha-o da página da disciplina. Nesta mesma página existe um pacote contendo um programa base de exemplo muito simples. O programa é acompanhado de uma `Makefile` que permite compilar o exemplo com o compilador adequado ao MicroRato executando `make hello.hex`.

**Exercício 22.4**

Verifique que além de compilar, também consegue executar o programa criado no MicroRato. Para isto utilize o comando `make run` e de seguida ligue o MicroRato. Isto irá copiar o código desenvolvido para o micro-controlador. Se o processo falhar, volte a tentar.

O programa irá esperar que pressione o botão de inicio, correspondendo este ao botão preto da plataforma.

Se aparecer a frase `"Hello World from MicroRato"`, parabéns pois conseguiu desenvolver uma aplicação para um micro-controlador.

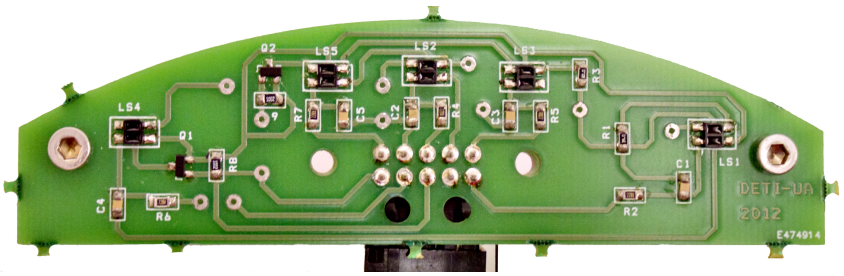


Figura 22.3: Sensores digitais de chão.

## 22.3 Sensores Analógicos e Digitais

A plataforma MicroRato está equipada com vários sensores, sendo que alguns são digitais, enquanto outros são analógicos. Neste contexto entende-se que os sensores digitais forneçam informação já processada com valores discretos, enquanto os sensores analógicos apenas produzem variações de tensão que têm de ser convertidas para valores concretos[3]. O processo de conversão implica adaptação dos valores a intervalos de tensão compatíveis com o micro-controlador, sendo que uma ADC irá depois converter os diferentes níveis de tensão em valores inteiros.

No caso da plataforma MicroRato existem sensores digitais que identificam se o utilizador pressionou um dos dois botões, assim como sensores que detetam se o chão é claro ou escuro, fornecendo um valor de 1 caso seja escuro e de 0 caso seja claro. Atenção que detetam claro e escuro e não branco e preto. O sensor emite pulsos de luz e tem um detetor de nível de luminosidade. Dependendo se o valor lido é superior ou inferior a um valor pré-configurado, ele irá considerar claro ou escuro. Estes sensores, que são 5, estão localizados à frente em baixo, tal como apresentado na Figura 22.3. É necessário especificar qual a amplificação a aplicar ao detetor, o que faz com o que o nível de decisão seja aumentado ou diminuído.

Todos os sensores, mesmo os digitais, possuem ruído. Isto é, além de 0 e 1 os sensores digitais podem oscilar rapidamente entre 0 e 1 devido a ruído. Um caso típico é o de pressionar um botão: até que o contacto seja realmente estável, existem momentos de alguns micro-segundos onde o valor irá oscilar entre 0 e 1. O exemplo seguinte irá imprimir para o terminal o tempo atual e o estado dos botões da plataforma, mas só quando for detetada uma diferença. Um programa deste tipo pode ser utilizado para detetar o ruído de pressionar um botão.

[3]Embora não seja aplicável a esta placa, também poderiam fornecer variações de corrente, frequência, capacidade, etc...

```c
#include "mr32.h"
#include <stdlib.h>

int main(void)
{
    initPIC32();
    closedLoopControl( false );
    setVel2(0, 0); //Stop Engines

    int start, oldStart = 2;
    resetCoreTimer();

    while(1)
    {
        start = startButton();
        if(start != oldStart)
                printf("%10d %d\n",readCoreTimer()/1000, start);
        oldStart = start;
    }
    return 0;
}
```

### Exercício 22.5

Implemente o exercicio anterior e verifique quantas alterações são apresentadas quando se pressiona uma vez o botão *start* (botão preto).

Os sensores de brilho do chão normalmente são tratados com as designações `CENTER`, `NEAR LEFT`, `NEAR RIGHT`, `FAR LEFT` e `FAR RIGHT`. Ao se realizar uma leitura, são lidos 5 bits, com valores de 0 ou 1 e que são interpretados de acordo com as designações anteriores. Para aceder aos sensores deve-se invocar a função `readLineSensors(int gain)`, indicando o ganho (amplificação) a aplicar aos sensores, sendo devolvido um inteiro em que cada bit corresponde ao estado de um sensor. O exemplo seguinte demonstra como se podem ser os sensores. Neste exemplo é chamada a função `waitTick40ms()` para introduzir uma cadência de 40 ms na execução de cada ciclo, evitando igualmente algum do ruído dos sensores.

```c
#include "mr32.h"
#include <stdlib.h>

#define GROUND_CENTER_BLACK(x) ( (x & 0b00000100) != 0)

int main(void)
{
    initPIC32();
```

```
    closedLoopControl( false );
    setVel2(0, 0); //Stop Engines

    while(1)
    {
        printf("Press start to continue\n");
        while(!startButton());
        do
        {
            waitTick40ms();
            int ground = readLineSensors(0);
            printf("%0x CENTER=%d\n", ground, GROUND_CENTER_BLACK(ground));
        } while(!stopButton());
    }
    return 0;
}
```

### Exercício 22.6

Crie um ficheiro chamado `ground.c` e altere o ficheiro `Makefile` para que compile esse ficheiro (veja a primeira linha). Programe a plataforma e verifique o resultado colocando o MicroRato em sítios onde o chão seja preto ou branco.

### Exercício 22.7

O programa anterior possui uma macro que devolve 1 ou 0 dependendo se o sensor de chão central está a ver preto ou branco. Implemente um conjunto de 4 outras macros para os restantes sensores de chão. Cada macro irá corresponder a um teste sobre os restantes 4 bits devolvidos pela função `readLineSensors()`.

Existem igualmente sensores analógicos que são utilizados para medir a distância entre a plataforma e obstáculos. O seu método de funcionamento consiste na emissão de um impulso de luz infra-vermelha, medindo-se o ângulo do sinal refletido. Estes sensores fornecem um valor que, diretamente não tem unidades, mas pode ser convertido para valores de distância através de cálculos simples. De realçar que os sensores analógicos raramente devolver valores precisos, existindo sempre um erro associado aos valores. Isto resulta que se a plataforma estiver parada, com um objeto a uma distância fixa, os valores irão variar naturalmente. Estes sensores também possuem um limite de distância, que corresponde a uma distância entre os 25 cm e os 30 cm. A Figura 22.3 apresenta estes sensores instalados na plataforma MicroRato.

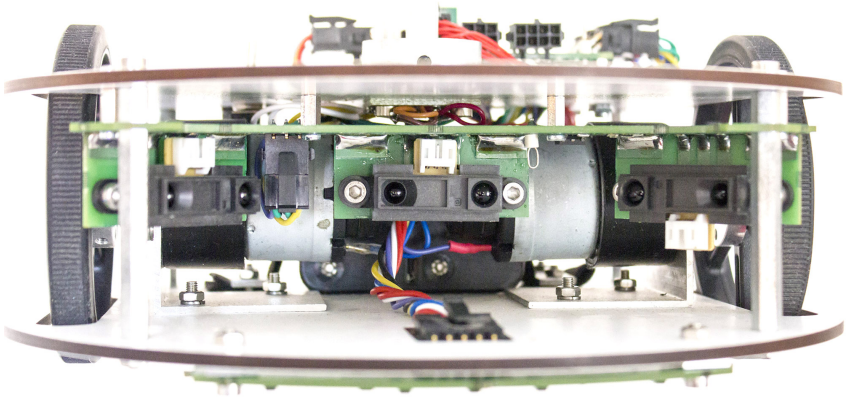

Figura 22.4: Sensores de distância por medição de ângulo da reflexão.

A leitura dos sensores analógicos é efetuada através da função `readAnalogSensors()`, sendo que uma estrutura chamada `analogSensors` é preenchida com os valores lidos. São lidos valores para a distância diretamente em frente, a 45º à direita e a 45º à esquerda. O programa seguinte imprime para o terminal o valor medido pelo sensor.

```
...
while(1)
{
    printf("Press start to continue\n");
    while(!startButton());
    enableObstSens()
    do
    {
        waitTick40ms();
        readAnalogSensors();
        printf("%d\n", analogSensors.obstSensFront);
    } while(!stopButton());
}
return 0;
...
```

### Exercício 22.8

Implemente o exercício anterior e programe o MicroRato com o programa resultante. Verifique que valores são devolvidos para várias distâncias e crie uma função `int sensorDistance(int x)` que transforme o valor medido numa distância em centímetros.

A estrutura `analogSensors` possui o seguinte formato:

```
struct
{
    int obstSensRight;
    int obstSensFront;
    int obstSensLeft;
    int an6;
    int an7;
    int batteryVoltage;
};
```

**Exercício 22.9**

Volte a implementar o exercício anterior de forma a reportar a distância detetada nos 3 sensores. Pode reutilizar a mesma função ou ter de implementar funções independentes (com fórmulas diferentes) para cada sensor de distância.

## 22.4 Atuadores

Além de ter capacidade de observar o meio à sua volta, a plataforma MicroRato também possui a capacidade de atuar, isto é, de gerar ações. Dois tipos de ações são possíveis: controlar os Light Emitting Diode (LED) que a plataforma possui no topo, ou controlar os motores que estão diretamente ligados a cada roda. As Figuras 22.4 demonstra a localização dos LED e dos motores da plataforma.

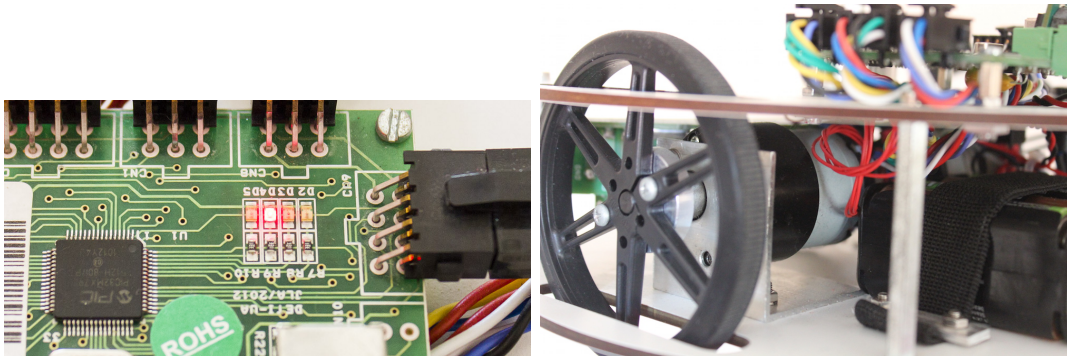

Figura 22.5: LEDs e motores

Os LED são frequentemente utilizados para indicar aos utilizadores da plataforma qual o estado interno de algumas variáveis ou processos. Por exemplo, podem ser utilizados para indicar visualmente se está a ser detetado algum objeto próximo, ou qual a distância ao objeto.

O controlo dos LED é feito através da função `led(int nr, int value)`, sendo que `nr` corresponde ao número do LED, possuindo valores entre 0 e 4. O argumento `value` corresponde ao valor que o LED deve ter, sendo admitidos dois valores: 0 (desligado) e 1 (ligado). O exemplo seguinte demonstra como se podem ativar sequencialmente todos os LED da plataforma.

```
 ...
while(!startButton());
int i = 0;
do
{
    waitTick40ms();
    led(i, 0);
    i = (i+1) % 4;
    led(i, 1);
    delay(10000);

} while(!stopButton());
...
```

**Exercício 22.10**

Complete o programa anterior e verifique que os LEDs são ativados de forma sequencial. Implemente depois uma variação que ativa os LEDs de acordo com a distância entre a plataforma e um objeto à sua frente. Uma forma simples é considerar que cada LED corresponde a um intervalo de 5 cm. Se todos os LEDs se encontrarem ativos deverá existir um objeto muito próximo. Se nenhum estiver ativo, o objeto está a mais de 20 cm.

Os motores são sem dúvida os principais atuadores da plataforma, permitindo que ela se mova no labirinto de uma prova. Existem várias versões da plataforma, sendo que a que usamos possui motores Direct Current (DC). Este tipo de motores não permite movimentos precisos pois baseia-se apenas no controlo da corrente útil que lhes é aplicada, através do método Pulse Width Modulation (PWM). Variações no fabrico dos motores ou rodas e peso da estrutura resultam em movimentos com menor previsão. Por outras palavras, não é possível dizer aos motores para moverem a plataforma durante 5 cm, apenas dizer que se movam com potência $X$, sendo que o programador terá de controlar

o tempo durante o qual esta ordem está aplicada. Outras versões da plataforma possuem sensores denominados de *encoders* que contam quantos graus as rodas realmente rodam. Para definir a velocidade dos motores (indiretamente indicando a potência), é utilizada a função `setVel2(int velLeft, int velRight)`. Os argumentos da função indicam qual a percentagem de potência a aplicar, podendo variar entre -100 e 100. Valores negativos indicam movimento para trás e valores positivos indicam movimento para a frente. De notar que só valores absolutos superiores a 40-60 implicam movimento, variando este valor de motor para motor. Mesmo na mesma plataforma, o mesmo valor aplicado aos dois motores pode resultar em velocidades diferentes para cada motor. O excerto seguinte roda o plataforma durante 1 segundo numa direção, invertendo depois a direção de rotação.

```
...
while(!startButton());
do
{
    setVel2(60, -60);
    delay(10000);
    setVel2(-60, 60);
    delay(10000);

} while(!stopButton());
...
```

**Exercício 22.11**

Simplifique o programa fornecido como exemplo mas varie o argumento da função `delay(int t)`. Esta função cria um período de espera em múltiplos de 100 µs.

**Exercício 22.12**

Utilizando valores de velocidade apenas suficientes para a plataforma se movimentar, determine quais os valores a aplicar a cada motor de forma a garantir que o MicroRato anda em frente a direito.

## 22.5 Tarefas

Através da leitura dos sensores analógicos, dos sensores digitais e da ativação dos motores é possível realizar tarefas bastante interessantes. Através de um detetor que indique a

direção, o que não se encontra presente nas plataformas que usamos, é possível percorrer labirintos, determinando a direção correta e atingindo o centro. Seguem-se diversas tarefas simples que permitem utilizar os sensores e atuadores da plataforma MicroRato de forma combinada.

### 22.5.1 Evitar obstáculos

Um dos objetivos de qualquer plataforma móvel é o de evitar obstáculos. Isto consegue-se através da conjugação de leituras contínuas dos sensores de distância, com a ativação dos motores de forma coordenada para evitar qualquer colisão. O algoritmo base considera que o MicroRato deve andar em frente, caso detete um objeto à esquerda, deve rodar para a direita. Se for detetado um objeto à direita, deve rodar para a esquerda. Caso o objeto se encontre imediatamente em frente podem ser tomadas várias ações. O MicroRato pode rodar sempre numa direção, pode rodar numa direção aleatória, pode rodar na direção inversa da última ordem. Estas ações podem ser isoladas ou combinadas com uma deslocação para trás. O excerto seguinte, após detetar um objeto a menos de 5 cm, roda para a esquerda durante aproximadamente 1 segundo.

```
 ...
while(!startButton());
do
{
    waitTick40ms();
    readAnalogSensors();
    if(distance(analogSensors.obstSensFront) < 5){
        for(int i = 0;i<25; i++) {
            waitTick40ms()
            setVel(-50, 50)
        }
    }

} while(!stopButton());
...
```

**Exercício 22.13**

Adapte o programa apresentado de forma que o, quando for encontrado um obstáculo em frente, o MicroRato rode aproximadamente 90°.

**Exercício 22.14**

Implemente um programa que permita ao MicroRato seguir objetos. Isto é, manter-se próximo de objetos detetados.

**Exercício 22.15**

Adapte o programa apresentado de forma que o, quando for encontrado um obstáculo em frente, o MicroRato reduza a sua velocidade até que estabeleça contacto. Depois deverá empurrar lentamente o objeto.

**Exercício 22.16**

Implemente um programa que permita ao MicroRato evitar obstáculos à direita, esquerda ou à frente. Compare a efetividade das diferentes estratégias possíveis para evitar colisões.

### 22.5.2 Seguir Linhas

Uma das funções permitidas pelos sensores de brilho é o de orientar o MicroRato com base no brilho do chão. Por exemplo, de forma a evitar zonas que estejam rodeadas por uma linha preta como uma pista, ou seguir linhas. Este último caso é muito utilizado em sistemas de automação industrial onde autómatos seguem linhas pelas instalações fabris para transportar cargas através de um percurso. Neste caso o sensor apenas distingue chão claro e escuro, enquanto sensores mais evoluídos podem distinguir várias cores, permitindo seguir apenas linhas de uma determinada cor.

O processo de seguir ou evitar uma cor é bastante simples, baseando-se na leitura contínua do brilho do chão. Se o objetivo foi seguir uma linha, deve-se manter o sensor central sempre sobre a linha preta. Se isto deixar de acontecer ou se os sensores laterais detetarem a linha, é necessário realizar uma alteração na rota, rodando a plataforma. Se

o objetivo for o de evitar linhas pretas, caso um dos sensores detete a linha, a rota tem de ser corrigida, rodando a plataforma na direção oposta.

O excerto seguinte pode ser utilizado para movimentar o robot até que seja encontrado uma linha, parando de seguida.

```
 ...
while(!startButton());
do
{
    waitTick40ms();
    int ground = readLineSensors(0);
    if(ground != 0)
        setVel2(0, 0);
    else
        setVel2(50, 50);
} while(!stopButton());
...
```

**Exercício 22.17**

Implemente um programa com o exemplo anterior de forma a que a plataforma MicroRato pare quando encontrar uma linha preta na mesa.

**Exercício 22.18**

Implemente um programa de forma a que a plataforma evite linhas pretas.

**Exercício 22.19**

Implemente um programa de forma a que a plataforma siga linhas pretas. Quando encontrar o final da linha, deverá dar meia volta até voltar a detetar a linha.

**Exercício 22.20**

Implemente um programa que faça o MicroRato capaz de lidar com cruzamentos. Quando atinge um cruzamento, deve sempre virar à direita. Se chegar ao final da linha deve dar meia volta. Um cruzamento deteta-se facilmente pois deverá ser uma das únicas situações onde tanto o sensor central como os laterais estarão sobre uma linha preta.

### 22.5.3 Evitar obstáculos na linha

Os sensores de brilho e de distância podem ser combinados de forma a implementar algoritmos mais complexos, por exemplo o de seguir uma linha onde existem obstáculos. O algoritmo base é o de seguir uma linha. Quando for encontrado um objeto, o MicroRato deverá contornar o objeto até que a linha volte a ser encontrada. Quando isto acontece, retoma-se o algoritmo de seguir linha.

**Exercício 22.21**

Implemente um algoritmo que permite ao MicroRato contornar objetos na linha. Utilize uma carteira, caderno ou outro objeto.

### 22.5.4 Música com o MicroRato

Os sensores de distância também podem ser utilizados para outros fins, tal como o de gerar música, imitando de forma grosseira parte do funcionamento de um instrumento chamado Teremim[4]. Para isto considere que pode gerar frequências diferentes dependendo da distância a que coloca as suas mãos. O excerto seguinte demonstra a parte principal de um programa *Python* que gera notas com duração de 200 ms e com uma frequência lida da porta de série.

```
import serial
import math
import pyaudio

RATE = 44100
BASE_DURATION = 0.2
```

[4] http://pt.wikipedia.org/wiki/Teremim

```
stream = player.open(format = player.get_format_from_width(2),
    channels = 1,
    rate = RATE,
    output = True)

sp = serial.Serial('/dev/ttyUSB0', 115200, timeout=1)

while True:
        freq = int(sp.readline())
        data = []
        for i in xrange(0, int(RATE*BASE_DURATION)):
                data.append(amplitude * sin(math.pi*i*freq/RATE))

        wvData = ""
        for v in data:
                wvData += pack('h',v)

        stream.write(data)
...
```

**Exercício 22.22**

Implemente um programa para executar na plataforma MicroRato de forma devolver uma frequência com base na distância da mão à plataforma. Complete o *script* de forma a tornar o sistema funcional.

**Exercício 22.23**

Melhore o sistema anterior de forma a eliminar ruído das leituras lendo vários valores e calculando a média. Implemente igualmente suporte para duas mão, uma controlando a amplitude e outra controlando a frequência. Terá de fazer uso dos vários sensores de distância da plataforma, ou em alternativa, dos sensores de brilho.

# Glossário

**ADC** Analog to Digital Converter

**CAN** Controller Area Network

**CPU** Central Processing Unit

| | |
|---|---|
| **DC** | Direct Current |
| **I²C** | Inter-Integrated Circuit |
| **LED** | Light Emitting Diode |
| **PWM** | Pulse Width Modulation |
| **RAM** | Random Access Memory |
| **ROM** | Read Only Memory |
| **RTC** | Real Time Clock |
| **SPI** | Serial Peripheral Interface |
| **SRAM** | Static Random Access Memory |
| **USB** | Universal Serial Bus |