```
/* Introducao a Robotica - Trabalho Pratico 3
Grupo 7: Corrida Maluca

Exercicio 5.2.3-4: Aplicando controle PD ao robo

este algoritmo permite ao robo receber um valor de distancia e percorrer
uma linha reta correspondente. Para tanto, cada roda deve ter um
shaft-encoder acoplado.
Consideramos que sao shaft-encoders simples, porntanto determinam apenas
a velocidade, mas nao a direcao. Sabemos em qual sentido o robo estara se
movendo pela funcao usada para ativar os motores. No entanto esta
implementacao garante que aquela distancia sera atingida, porem nada pode
fazer se esta for ultrapassada.

O controle sera implementado usando um controlador PD em cada roda. Como
nosso objetivo de controle depende de uma combinacao destes dois
sistemas, acrescentamos em cada controlador um terceiro termo, que e
proporcional a diferenca de posicao entre as rodas.

Por fim, este programa requer que os drivers "sencdr0.icb" e "sencdr1.icb"
(ou "fencdr0.icb" e "fencdr1.icb", caso opte pela versao "rapida" dos
drivers) sejam carregados na handyboard. Caso se deseje usar outra porta
para os encoders, alem de das mascaras, os drivers a carregar devem tambem
ser alterados.
*/

/* Mascaras para as portas utilizadas */
#define motor_dir 0
#define motor_esq 3

//consideramos que o encoder da roda direita esta na porta 0.
#define encdr_dir 0
#define encdr_esq 1

/* Mascaras para os ganhos de cada controlador */
#define ganho_p_dir ????
#define ganho_d_dir ????
#define ganho_p_esq ????
#define ganho_d_esq ????
#define ganho_diff ????

/* Variaveis globais associadas a parametros recebidos */
//distancia que deve ser percorrida, dada em pulsos
persistent int alvo = 0;

void distancia()
{
/*permite ajustar o valor da distancia a ser percorrida */

    /*multiplicadores para permitir ajustar individualmente cada casa
    decimal*/
    int i, mult[4];

    // inicializacao dos multiplicadores
    for(i = 0; i < 4; ++i)
    {
        mult[i] = 10^i;
```

```
    }

    //loop principal
    while(1)
    {
        printf("distancia atual = %d\n", alvo);
        sleep(0.5);
        printf("start -> recebe valor\n stop -> mantem atual");

        //testa qual botao foi pressionado
        if(start_button())
        {
            i = 0;
            while(1)
            {
                /*cada vez que star e pressionado, alterna qual casa decimal
                esta sendo ajustada. O scroll da handyboard permite escolher
                o valor a ser colocado em cada casa)*/
                printf("distancia = %d\n", (alvo - alvo%(i + 1) + (knob()%10)*mult[i]));
                sleep(0.5);
                if(start_button())
                {
                    alvo = (alvo - alvo%(i + 1)) + (knob()%10);
                    i = (++i)%4;
                }
                if(stop_button())
                {
                    alvo = (alvo - alvo%(i + 1)) + (knob()%10);
                    printf("nova distancia = %d\n", alvo);
                    sleep(0.5);
                    break;
                }
            }
        }
        if(stop_button())
        {
            printf("Distancia = %d armazenada\n", alvo);
            sleep(0.5);
            break;
        }

    }
}

void main()
{
    // Variavel que recebera a distancia percorrida pelo robo.
    int dist_total = 0;

    /*/variaveis que armazenam a potencia a ser aplicada em uma dada
    iteracao*/
    int pot_esq, pot_dir, pot_diff;


    //solicita ao usuario o valor do alvo
    distancia();
```

```
    //zera a contagem dos encoders
    encoder0_counts = 0;
    encoder1_counts = 0;



    while(dist_total < alvo)
    {
        /*Calculo do ganho associaod a diferenca de contagem entre as
        rodas. Note que este ganho e aplicado com sinal diferente de
        acordo com o lado em questao*/
        pot_diff = ganho_diff * (encoder0_counts - encoder1_counts);

        //calculo da potencia de cada motor
        pot_dir = ganho_p_dir * (alvo - encoder0_counts) - (ganho_d_dir * encoder0_velocity)
- pot_diff;
        pot_esq = ganho_p_esq * (alvo - encoder1_counts) - (ganho_d_esq * encoder1_velocity)
+ pot_diff;

        /* se a contagem do encoder direito e maior, a potencia esquerda aumenta e a direita
diminui. Se
        a contagem do encoder esquerdo for maior, pot_diff tera sinal negativo, entao a
potencia direita
        aumenta e a esquerda diminui.*/

        motor(motor_dir, pot_dir);
        motor(motor_esq, pot_esq);

        /*consideramos que a distancia percorrida pelo robo e a media
        entre as contagens dos encoders*/
        dist_total = (encoder0_counts + encoder1_counts)/2;
        printf("distancia percorrida = %d\n", dist_total);

        //Gera um intervalo entre execucoes continuas
        msleep(50L);
    }
}
```