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

Exercicio 5.1.3-1
Implementacao de um algoritmo de tres estados para seguir paredes usando um sensor
optico.
Usamos o metodo de curva suave (quando se deseja fazer uma curva, os dois motores sao
ativados na mesma direcao, com potencias diferentes), com o fator de aumento da potencia
calculado a partir do erro (referencia - valor medido).
Consideramos que a parede esta a esquerda do sensor. Assim, se a leitura exceder o
limite superior, o robo se afastou demais, e se exceder o limite inferior, o robo se
aproximou demais. Note que este mesmo algoritmo pode ser facilmente alterado para
funcionar com a parede a direita do sensor, bastando trocar a direcao das curvas.
*/

/* Mascaras para as portas de conexao usadas*/
#define motor_dir 0
#define motor_esq 3
#define sensor_wall 0

/*Mascaras para os limites de variacao de erro aceitaveis */
/*note que, se o robo se afasta muito, o erro e negativo. Se o robo se aproxima muito,
o erro e positivo. */
#define limite_pos ????
#define limite_neg ????

/* Variaveis globais de calibracao */
//referencia de distancia da parede
persistent int wall_ref;

//tempo de duracao da execucao (em decimos de segundo)
persistent int duration;

/* Variaveis globais para registro de dados capturados*/
persistent int data[1000];
persistent int data_index;

/*Ganho da funcao de executar curvas*/
#define ganho 5

void calibrar()
{
    /*
Funcao que permite definir um novo valor para a referencia.  */
    while(1)
      {
        int wall = analog(sensor_wall);
        //informa a referencia e distancia da parede atuais
        printf("ref = %d; dist = %d\n", wall_ref, wall)

          // se "start" e pressionado, a distancia atual se torna a nova referencia
          if(start_button())
          {
            wall_ref = wall;
        }
        //se "stop" e pressionado, exibe o valor atual da referencia e encerra a funcao
        if(stop_button())
```

```
            {
              printf("Referencia ajustada para %d\n", wall_ref);
              beep();
              //aguarda, para permitir que a mensagem seja lida
              sleep(0.5);
              break;
            }

          //aguarda, para permitir que a mensagem seja lida
          msleep(50L);
      }
}

void temporizar()
{
    /* funcao que permite definir por quanto tempo o algoritmo sera excutado. Este valor
pode variar de 100 a 355, o que corresponde a tempos entre 10 e 35 segundos
aproximadamente. O novo valor  ajustado atraves do "scroll da handyboard*/

    while(1)
      {
        //informa o valor atual do tempo de execucao
        printf("duracao = %d\n", duration);

        //se "start" e pressionado, recebe um novo valor para a duracao
        if(start_button())
          {
            while(!start_button())
              {
                printf("nova duracao = %d\n", knob() + 100);

                //aguarda, para permitir que a mensagem seja lida
                sleep(0.5);
              }
              //atribui o novo valor a variavel
              duration = knob()+100;
          }

        //se "stop" e pressiondo, exibe o valor atual da duracao e encerra
        if(stop_button())
          {
            printf("Duracao ajustada para %d\n", duration);
            beep();
            //aguarda, para permitir a leitura da mensagem
            sleep(0.5);
            break;
          }

        //aguarda, para permitir que a mensagem seja lida
        msleep(50L);
    }
}


}
void main()
{
```

```
    //chama a funcao de calibracao
    calibrar();

    //chama a funcao de temporizacao
    temporizar();

    //zera o indice do vetor de dados
    data_index = 0;

    //declara a variavel que armazena o tempo decorrido
    int tempo = 0;

    //loop principal, executa 10 vezes por segundo
    while(tempo < duration)
      {
        //le a atual distancia da parede
        int wall = analog(sensor_wall);

        //calcula o erro
        int erro = wall_ref - wall;

        //escreve o valor lido e o valor de referencia
        printf("ref = %d; dist = %d; erro =  %d\n", wall_ref, wall, erro);

        //testa se a distancia esta dentro da faixa de tolerancia
        if(erro < limite_pos && erro > limite_neg)
          avante(); //segue em linha reta
        else
        {
            /*executa a funcao para virar. O sinal do erro determina para qual lado a curva
            deve ser feita*/
            vira(erro);
        }

        //registra o valor de distancia medido no vetor de dados
        data[data_index++] = wall;

        //aguarda 100 milissegundos
        msleep(100L);

        //incrementa o contador
        ++tempo;
    }
}

// a seguir implementamos as tres funcoes usadas para mover o robo neste programa.

void avante()
{
    /*funcao para fazer o robo andar "em linha reta". Como nao monitoramos a velocidade
de cada roda, provavelmente haverao desvios em relacao a trajetoria desejada*/

    //colocamos ambos os motores a 60% de sua potencia maxima
    motor(motor_dir, 60);
    motor(motor_esq, 60);
}
```

```
/*Analisando a versao anterior do programa, concluimos que as duas funcoes de virar
executavam o mesmo codigo, portanto eram reduntantes. Da forma como estavam escritas, o
sinal do erro determinava o lado da curva em ambas, porem tinhamos chamadas distintas para
cada caso (erro positivo e negativo. Nesta implementacao, temos apenas uma funcao para
virar, e o sinal do erro gerado determina para qual lado sera a curva. */
void vira(int erro)
{
    motor(motor_esq, 50 + (erro * ganho));
    motor(motor_dir, 50 - (erro * ganho));
}

void dump_data(int erro)
{
    /*funcao para enviar a um computador os dados armazenados no vetor "data". Esta funcao
deve ser chamada pela linha de comandos do IC, e um emulador de terminal deve ser usado
para receber os dados.
Os parametros da transmissao sao: baud = 9600, data bits = 8, 1 stop bit e nenhum bit de
paridade (9600-N-8-1).
E necessario carregar os drivers "serialio".c e "printdec.c" para usar esta funcao.*/

    int i;

    disable_pcode_serial();

    printf("Pressione start para enviar...\n");
    start_press();

    for(i = 0; i < data_index; ++i)
      {
        printdec(dat[i]);
        serial_putchar(10);
        serial_putchar(13);
    }

    print("Dados enviados.\n");
    beep();
}
```