```
//#include "testeEncoder1.c"
#use "encoders\sencdr2.icb"
#use "encoders\sencdr5.icb"

#define LEFT_MOTOR 3
#define RIGTH_MOTOR 2

#define ESQUERDA 1
#define FRENTE 0
#define DIREITA -1

#define LESTE 0
#define NORTE 90
#define OESTE 180
#define SUL 270

#define X_DIM 10
#define Y_DIM 10

#define MAX 255

/* Mapa contendo os obstaculos do ambiente */
int map[X_DIM][Y_DIM];

/* Pose = posicao (x y) + orientacao */
int pose_atual[3];    // posicao (x,y) atual + orientacao (N / S / L / O)
int pose_inicial[3];  // posicao inicial
int pose_desejada[3]; // posicao final desejada (ignorar a orientacao)

/* Variaveis extras*/
int cntFifo;
int fifo[100][2];
int indice = 0;
int indice2 = 0;

//void motor(int m, int p) {}
//void ao() {}
//void msleep(long time) {}

/* Esta funcao deve fazer o seu robo andar 1 unidade de distancia para
 * frente e tambem deve atualizar a variavel pose_atual, de acordo
 * com a orientacao atual
 */
void goAhead()
{
   int cntAtual;
   if (pose_atual[2] == LESTE)
     pose_atual[0]++;
   if (pose_atual[2] == NORTE)
     pose_atual[1]++;
   if (pose_atual[2] == OESTE)
     pose_atual[0]--;
   if (pose_atual[2] == SUL)
     pose_atual[1]--;

   cntAtual = encoder2_counts;
   motor(LEFT_MOTOR, 57);
   motor(RIGTH_MOTOR, 70);
   //msleep(2000L);
   while(encoder2_counts < cntAtual + 60);
```

```
    ao();
    sleep(0.5);
}

/* Esta funcao deve fazer o seu robo girar 90 graus para a esquerda e
 * tambem deve atualizar a variavel pose_atual
 */
void turnLeft() {
    int cntAtual;
    pose_atual[2] += 90;
    if (pose_atual[2] == 360)
      pose_atual[2]= 0;

    cntAtual = encoder2_counts;
    motor(LEFT_MOTOR,-55);
    motor(RIGTH_MOTOR,+55);

    while(encoder2_counts < cntAtual + 13);
    ao();
    sleep(0.5);
}

/* Esta funcao deve fazer o seu robo girar 90 graus para a direita e
 * tambem deve atualizar a varivael pose_atual
 */
void turnRight() {
    int cntAtual;
    pose_atual[2] -= 90;
    if (pose_atual[2] == -90)
      pose_atual[2]= 270;

    cntAtual = encoder2_counts;
    motor(LEFT_MOTOR,+55);
    motor(RIGTH_MOTOR,-55);

    while(encoder2_counts < cntAtual + 12);
    ao();
    sleep(0.5);
}

void push(int x, int y, int valor) {
    if (x >= 0 && x < X_DIM && y >=0 && y < Y_DIM && map[x][y] == 0) {
        //printf("fifo[%d] <= [%d %d]\n", indice, x, y);
        map[x][y] = valor;
        fifo[indice][0] = x;
        fifo[indice][1] = y;
        indice++;
        cntFifo++;
    }
}

void pop(int *x, int *y) {
    *x = fifo[indice2][0];
    *y = fifo[indice2][1];
    indice2++;
    cntFifo--;
    //printf("fifo[%d] => [%d %d]\n", indice2, *x, *y);
}
```

```
/* Esta funcao deve calcular o mapa de distancias usando o algoritmo
 * wavefront.
 * A variavel map deve conter as distancias entre o ponto final e inicial
 * ao final da sua execucao
 */
void calculaWaveFront() {
   int x,y;
   int i,j;
   int stop = 0;
   int flagErro = 0;

   for (i = 0; i < X_DIM; i++) {
      map[i][0] = MAX;
      map[i][Y_DIM - 1] = MAX;
   }
   for (j = 0; j < Y_DIM; j++) {
      map[0][j] = MAX;
      map[X_DIM- 1][j] = MAX;
   }

   push(pose_desejada[0], pose_desejada[1], 1);
   x = pose_desejada[0], y = pose_desejada[1];
   cntFifo = 0;

   while (!stop) {
      pop(&x , &y);
      if (y - 1 >= 0) push(x, y - 1, map[x][y] + 1);
      if (y + 1 < 10) push(x, y + 1, map[x][y] + 1);
      if (x + 1 < 10) push(x + 1, y, map[x][y] + 1);
      if (x - 1 >= 0) push(x - 1, y, map[x][y] + 1);
      if (x == pose_inicial[0] && y == pose_inicial[1])
        stop = 1;

      if (cntFifo == 0)
        {
          flagErro = 1;
          break;
        }
   }

   if (flagErro)
     {
       printf("alvo nao pode ser alcancado\n");
       while (1);
   }

   for (j = 0; j < Y_DIM; j++)
     for (i = 0; i < X_DIM; i++)
       if (map[i][j] == 0) map[i][j] = MAX;
}

/* Esta funcao deve retornar a direcao que o robo deve seguir para sair
 * do quadrante atual para chegar ate o proximo quadrante, obedecendo
 * o mapa criado pelo algoritmo wavefront
 */
int calculaDirecao() {
   int x, y, xa, ya, xl, yl, xr, yr;

   x = pose_atual[0];
```

```
    y = pose_atual[1];
    xa = x;    xl = x;
    xr = x;    ya = y;
    yl = y;    yr = y;

    if (pose_atual[2] == LESTE) {
        xa = x + 1;
        yl = y + 1;
        yr = y - 1;
    }
    if (pose_atual[2] == NORTE) {
        ya = y + 1;
        xl = x - 1;
        xr = x + 1;
    }
    if (pose_atual[2] == OESTE) {
        xa = x - 1;
        yl = y - 1;
        yr = y + 1;
    }
    if (pose_atual[2] == SUL) {
        ya = y - 1;
        xl = x + 1;
        xr = x - 1;
    }
    if (map[xa][ya] < map[x][y])
      return FRENTE;
    if (map[xl][yl] < map[x][y])
      return ESQUERDA;
    if (map[xr][yr] < map[x][y])
      return DIREITA;
}

/* Imprime o mapa na tela. util para realizar a depuracao (so funciona no PC).
 */
void printMap() {
    int i, j;
    for (j = 0; j < Y_DIM; j++) {
        for (i = 0; i < X_DIM; i++) {
            printf("%3d ", map[i][j]);
        }
        printf("\n");
    }
}

/* Le um valor do knob variando de 0 at intervalo*/
int getKnob(char texto[], int intervalo) {
    int valor;
    while (!start_button()) {
        valor = (intervalo * knob()) / 255;
        printf("%s %d\n", texto, valor);
        msleep(100L);
    }
    while (start_button());
    return valor;
}

void escolhePosicao() {
    int posicao;
    pose_inicial[0] = getKnob("X inicial: ", X_DIM);
```

```
    pose_inicial[1] = getKnob("Y inicial: ", Y_DIM);

    pose_atual[0] = pose_inicial[0];
    pose_atual[1] = pose_inicial[1];

    pose_desejada[0] = getKnob("X desejado: ",X_DIM);
    pose_desejada[1] = getKnob("Y desejado: ",Y_DIM);
}

void escolheMapa() {
    int x, y;
    int n, i;
    n = getKnob("Numero de obstaculos: ", 20);
    for (i = 0; i < n; i++) {
        x = getKnob("X obstaculo: ",X_DIM);
        y = getKnob("Y obstaculo: ",Y_DIM);

        /*if (x == pose_desejada[0] || x == pose_inicial[0] || y == pose_desejada[1] || y == pose_inicial[1]
            || x == 0 || y == 0 || x == X_DIM -1 || y == Y_DIM -1)
          {
            printf("parede na posicao inicial/final\n");
            while (1);
        }*/
        map[x][y] = MAX;
    }
}

int main() {
    int direcao;
    int stop = 0;

    printf("Wavefront\n");
    msleep(1000L);

    escolhePosicao();
    escolheMapa();

    printf("Comecando em [%d %d], alvo: [%d %d]\n",
          pose_inicial[0], pose_inicial[1],
          pose_desejada[0], pose_desejada[1]);

    calculaWaveFront();
    //printMap();
    printf("chegei aqui\n");

    while (!stop) {
        direcao = calculaDirecao();
        if (direcao == ESQUERDA)
          turnLeft();
        if (direcao == DIREITA)
          turnRight();
        goAhead();

        printf("Estou em [%d %d]\n", pose_atual[0], pose_atual[1]);

        /* Verifica se ja chegou no alvo*/
        if (pose_atual[0] == pose_desejada[0] &&
            pose_atual[1] == pose_desejada[1])
          stop = 1;
    }
```

```
    return 0;
}
```