```
/*
 Código desenvolvido pelo Grupo 07 para o primeiro trabalho de introdução a robótica

 O robo deve ser programado para desenhar circulos e quadrados. Assim, construimos duas
funções
 básicas:
 -> Circulo, permite ao robô desenhar arcos de circunferência, recebendo como parâmetro o raio
 e o ângulo desejados.
 -> Reta, faz com que o robô desenhe uma linha reta cujo o comprimento é recebido como
parâmetro.

 Nosso robô também possui sensores movimento angular, e foi desenvolvida a função "emissor"
para
 monitorá-los. Como esta medida é contínua, esta função será executada concorrentemente ao
 programa principal, e depositará seus dados em uma variável global.
 */


 /****** Considerações *****
 1. Ao longo de todo o programa, quando usamos números para representar algo associado a um
lado
 ou outro do robô, adotaremos a seguinte convenção:
 0 => direita
 1 => esquerda
 */

 //Variáveis globais

//variáveis para a comunicação entre os processos de leitura do sensor e o programa principal.
float vel[2];

// valores de potência serão armazenados em variáveis globais, também para fins de comunicação
int pot_esq,pot_dir;

void emissor(int lado)
{/*
    Esta função monitora as medidas de um sensor de interrupção de feixe.
    A cada iteração mede-se o valor atual da contagem em um dado instante e o valor da
contagem
    meio segundo depois. O módulo da diferença entre estas medidas, multiplicado pelo raio
    fornece a velocidade da roda.
    É importante destacar que o esta função não diferencia o sentido de rotação. O
conhecimento
    sobre esta variável será totalmente derivado dos comandos enviados para acionar o motor.*/

    int count_atual = 0;
    int count_init = 0;

    while(1)
      {
        //Lê a contagem do encoder no início do intervalo
        count_init = read_encoder(lado);

        //gerando um atraso de meio segundo
        sleep(0.5);

        //Lê a contagem do encoder ao final do intervalo
        count_atual = read_encoder(lado);
```

```
        /*determinando da velocidade*/
        vel[lado] = sqrt((0.01*((float)(count_atual - count_init)))^2);
    }
}


void circulo(float raio, float angulo)
{
    /*
    Função para desenhar arcos de circunferência.
    A idéia é aplicar potências distintas em cada motor para gerar a circunferência. A
diferença
    entre as potências será determinada pelo raio desejado, enquanto o ângulo determinará a
    distância que o robô deverá percorrer.
    Para simplificar, vamos manter o motor esquerdo a uma potência constante e reduzir a do
    direito.
    Destacamos que esta abordagem implica em desenhar circulos sempre em um mesmo sentido,
que
    neste caso será horário. (consideramos que o parâmetro raio será sempre positivo. No
entanto
    não acrescentamos código para fazer esta verificação. O ângulo será recebido em graus e
    convertido a uma distância linear correspondente. A distância percorrida é a determinada
    pela velocidade do ponto central do eixo de tração multiplicada pelo raio das rodas
    */

    //cálculo da distância a percorrer a partir do ângulo
    float dist = 3.14159*raio*angulo/180.0;

    //variável que armazena a distância percorrida
    float distPerc = 0.0;

     //A potência do motor esquerdo será mantida em valor constante
    pot_esq = 50;

    /*A potência do motor direito é determinada a pelo raio desejado*/
    pot_dir = (int)(50.0*((raio-0.06)/(raio+0.06)));

    //acionando o terceiro motor para abaixar a caneta.
    motor(1, -55);

    /***** loop para desenhar o círculo ******/
    while(1)
      {
      //acionando dos motores
        motor(3,pot_dir);
        motor(0,pot_esq);

        /****** controle da velocidade ******/

        /*como a potência do motor esquerdo é "constante", o erro será avaliado pelo desvio
do
        motor direito em relação a este parâmetro*/

        //verificando se a velocidade do lado direito está muito alta
        if(vel[0] > vel[1]*(raio-0.06)/(raio+0.06))
        {// reduzindo a potência do motor direito, porém nunca abaixo de 20%
            if(pot_dir > 20)
```

```
            pot_dir--;

          //se a potência do motor direito já está muito baixa, aumenta a do lado esquerdo
          else
            pot_esq++;
      }

      //verificando se a velocidade do lado direito está muito baixa
      if(vel[0] < vel[1]*(raio-0.06)/(raio+0.06))
      {// aumentando a velocidade do lado direito
          ++pot_dir;

         /*
         note que agora não nos preocupamos em avaliar limites para a potência do motor
         direito, pois se ela se desviar demais, será corrigida pela verificação anterior
          */
      }
      sleep(0.001);

      //atualizando a distância percorrida
      distPerc += ((vel[1]+vel[0])/2.0)*0.001;

      //quando a distância percorrida superar o limite estabelecido, encerra o loop
      if(distPerc >= dist)
        break;
    }

    //levanta a caneta
    motor(1, 55);
    sleep(1.0);
    motor(1,0);
}

void reta(float distancia)
{
    /*
    função para desenhar uma linha reta de comprimento variável.
    Esta função se baseia em um controle de da velocidade das rodas. A cada iteração
avalia-se a
    velociade   de uma roda em relação a outra, e qualquer desvio gera provoca uma variação
de 1%
    na potência de ambos os motores. O valor de referência para a potência dos motores é 30%.

    Note que a função aceita também valores negativos do comprimento, indicando neste caso
que o
    robô andará "de ré"
    */

    /*dist é uma variável auxiliar, para armazenar o módulo do comprimento a ser percorrido*/
    float distPerc, dist;

    //inicializa a distância já percorrida
    distPerc = 0.0;

    /****** determinando dos parâmetros em função do sentido do movimento ******/
    if(distancia >= 0.0)
      {
        pot_esq = 30;
```

```
        pot_dir = 30;
        dist = distancia;
    }
    else
      {
        pot_esq = -30;
        pot_dir = -30;
        dist = -distancia;
    }

    /****** loop para desenhar a reta ******/
    while(1)
      {
        //acionando os motores com o valor calculado para a potência de cada lado.
        motor(3,pot_dir);
        motor(0,pot_esq);

        if(vel[1] > vel[0])
          {/*
            se a velocidade da roda direita é maior que a da esquerda, verifica o desvio em
relação
            ao valor de referência (30%)*/
            if(sqrt((float)pot_dir^2.0) >= 30.0)
              {/*
               se o módulo de potência é maior do que 30%, aplica a correção adequada
baseado no
               sentido do movimento */
                if(distancia >=0.0)
                {
                    pot_dir--;
                }
                else
                {
                    pot_dir++;
                }
            }
            else
            {/*
              se o módulo de potência é menor do que 30%, aplica a correção adequada baseado
no
              sentido do movimento */
                if(distancia >=0.0)
                {
                    pot_dir++;
                }
                else
                {
                    pot_dir--;
                }
            }
        }
        if(vel[0] > vel[1])
          {/*
            se a velocidade da roda esquerda é maior que a da direita, verifica o desvio em
relação
            ao valor de referência (30%) */
            if(sqrt((float)pot_esq^2.0) >= 30.0)
              {/*
```

```
                se o módulo de potência é maior do que 30%, aplica a correção adequada
baseado no
                sentido do movimento */
                 if(distancia>=0.0)
                 {
                     pot_esq--;
                 }
                 else
                 {
                     pot_esq++;
                 }
             }
             else
             {/*
              se o módulo de potência é menor do que 30%, aplica a correção adequada baseado
no
              sentido do movimento */
                 if(distancia>=0.0)
                 {
                     pot_esq++;
                 }
                 else
                 {
                     pot_esq--;
                 }
             }
         }
         sleep(0.1);

         //atualiza o valor da distância já percorrida
         distPerc += vel[1]*0.1;

         printf("Reta: %f\n",distPerc);

         //quando a distância percorrida superar o limite estabelecido, encerra o loop
         if(distPerc >= dist)
           break;
     }

     //desligando os motores
     motor(0,0);
     motor(3,0);
}

void retangulo(float a, float b)
{
     /*
     Esta função desenha um retângulo, recebendo como parâmetros o comprimento de dois de
seus lados.
     Cada lado é desenhado fazendo uma chamada à função "reta". Após desenhar um lado, é
ordenado ao
     robô executar um récuo equivalente à distância entre o centro do eixo de tração e o
ponto de
     contato da caneta, um giro de noventa graus (usando a função "circulo", implementada
neste mesmo
     arquivo), e um avanço de mesmo comprimento do récuo (que também é o comprimento do raio
do arco).
     */
```

```
/****** Desenhando o lado 1 ******/

//encosta a caneta no papel
motor(1,-55);
reta(a);

//levanta a caneta
motor(1, 55);
sleep(1.0);
motor(1,0);

//Alinhando com o próximo lado
reta(-0.16);
circulo(0.16, 90)
reta(0.16);

/****** Desenhando o lado 2 ******/

//encosta a caneta no papel
motor(1,-55);
reta(b);

//levanta a caneta
motor(1, 55);
sleep(1.0);
motor(1,0);

//Alinhando com o próximo lado
reta(-0.16);
circulo(0.16, 90)
reta(0.16);

/******Desenhando o lado 3 ******/

//encosta a caneta no papel
motor(1,-55);
reta(a);

//levanta a caneta
motor(1, 55);
sleep(1.0);
motor(1,0);

//Alinhando com o próximo lado
reta(-0.16);
circulo(0.16, 90)
reta(0.16);

/****** Desenhando o lado 4 ******/

//encosta a caneta no papel
motor(1,-55);
reta(b);

//levanta a caneta
motor(1, 55);
sleep(1.0);
```

```
    motor(1,0);
}

//função principal
void main()
{
    /*
    Para fim de registro, definimos que nosso robô desenharia uma linha reta, um quadrado e
um
    circulo, nesta ordem. Para que o robô pudesse ser deslocado entre cada uma destas etapas
(para
    evitar a superposição de desenhos, por exemplo), ele espera que o botão "start" da
handyboard
    seja pressionado antes de começar a desenhar.
    Note que os parâmetros dos desenhos (comprimento da reta, dimensões do quadrado e do
círculo)
    são definidos em no código, portanto sua alteração exige que se refaça o download do
código para
    a handyboard.
    */

    //habilita a porta onde conectamos os sensores para o uso da função "read_encoder()"
    enable_encoder(1);
    enable_encoder(0);

    //cria os processos para leitura continua de cada sensor ótico
    start_process(emissor(0));
    start_process(emissor(1));

    //aguarda que o botão "start" da handyboard seja pressionado
    while(!choose_button()){}
    while(choose_button()){}

    /****** desenhando uma linha reta *****/

    //encosta a caneta no papel
    motor(1,-55);

    //desenha uma reta de 30 cm
    reta(0.30);

    //levanta a caneta
    motor(1,55);
    sleep(1.0);
    motor(1,0);

    //aguarda que o botão "start" da handyboard seja pressionado
    while(!choose_button()){}
    while(choose_button()){}

    /****** desenhando um quadrado com 10 cm de lado ******/
    retangulo(0.10,0.10);

    //aguarda que o botão "start" da handyboard seja pressionado
    while(!choose_button()){}
    while(choose_button()){}

    /***** desenha um círculo de raio 30 cm ******/
```

```
    circulo(0.3,360.0);
}
```