Trabalho Prático 3

Objetivo

1. Familiarizar o grupo com sensores ópticos e técnicas básicas de controle.

Decisão

1. Foi utilizada a mesma montagem do Tp1, na qual o robô se apóia sobre três rodas, sendo uma totalmente livre, sem nenhum tipo de acionamento, e outras duas acionadas, independentemente, pelos motores.

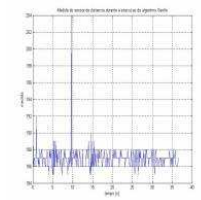
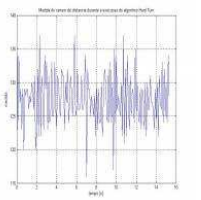
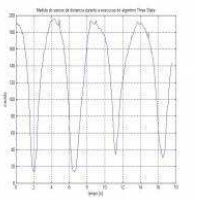
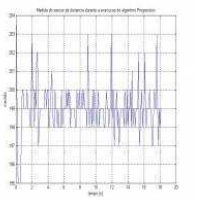
Problemas Encontrados

1. Tivemos muitos problemas com a qualidade dos motores fornecidos ao grupo, um deles estava bom, mas o outro se mostrou completamente ineficiente, funcionando apenas a uma potência maior que 50 em quase todos os casos.
2. Foi difícil implementar o controle PD para a movimentação do robô pois as equações disponíveis no livro não se comportaram muito bem com nosso robô, nos obrigando a fazer algumas alterações para um correto funcionamento.

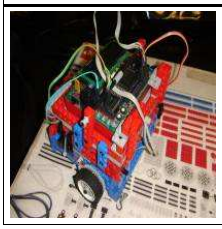
Montagem

1. A montagem foi a mesma do Tp1, mas além disso, instalamos os shaft encoders em um eixo intermediário ao eixo do motor e o eixo que a roda gira, pegando as rotações de uma peça do lego que já é furada, com 6 furos. Fizemos isso para que os shaft encoders não ficassem em um eixo que girasse muito rápido ou que girasse muito lento.

Gráficos

Gentle-Turn	Hard-Turn	Three-State	Progressive
			

Demonstração

Fotos		
		

Vídeo
Vídeo YouTube

Exercícios

Seção 3.6.5

1. Após realizar alguns testes, determinamos que o melhor valor de resistência para o resistor ligado em paralelo com o resistor de 47 Kohms da HandyBoard é de 47 Kohms. Este foi o melhor valor pois quanto maior a resistência do potenciômetro maior será a sensitividade do sensor.
2. Letras:
   1. a) V = 5 - 2 = 3 ⇒ V = RI ⇒ 3 = 330 * I ⇒ I = 9 mA
   2. b) 25 mA / 9 mA =~ 3 LED's

Seção 5.1.3

1 - Three-State:

```
while(1) {
    sensor_l = analog(5);
    if (sensor_l < limiar_perto) {
        motor(0,100);
        motor(2,50);
    }
    else if (sensor_l > limiar_longe) {
        motor(0,70);
        motor(2,70);
    }
    else {
        motor(0,100);
        motor(2,70);
    }
}
```

2 - Hard-Turn X Gentle-Turn:

```
Hard-Turn:
while(1) {
    sensor_l = analog(5);
    if (sensor_l < limiar) {
        motor(0,100);
        motor(2,0);
    }
    else {
        motor(0,0);
        motor(2,100);
    }
}
```

```
Gentle:
while(1) {
    sensor_l = analog(5);
    if (sensor_l < limiar) {
        motor(0,80);
        motor(2,0);
    }
    else {
        motor(0,0);
        motor(2,50);
    }
}
```

3 - Three-State X Gentle-Turn X Hard-Turn

```
Descobrimos, pelos testes, que o método Three-State é o melhor dos três. Com os dois limiares ele conseguiu
```

4 - Progressive

```
while(1) {
    sensor_l = analog(5);
    if (sensor_l < limiar) {
        int pow_l = pow_l + (int)((limiar - sensor_l)*p_gain);
        if (pow_l > 100) pow_l = 100;
        if (pow_l < 50) pow_l = 50;
        motor(0,pow_l);
        motor(2,40);
    }
    else {
        motor(0,70);
        pow_r = pow_r + (int)((sensor_l - limiar)*p_gain);
        if (pow_r > 100) pow_r = 100;
        if (pow_r < 20) pow_r = 20;
        motor(2, pow_r);
    }
}
```

Seção 5.2.3

1 - Utilizamos dois Pgain em nosso PD, um para a roda esquerda e outro para a direita. Pgain_left = 5.0 e Pgain_right = 1.0. Tivemos de fazer isso pois nosso motor esquerdo é terrivelmente inferior ao nosso motor direito. Já o Dgain foi de 0.05 para ambos. Para chegar nestes valores apenas testamos o robô em linha errata e jogamos valores diferentes para então descobrir os melhores nos experimentos.

4 -

```
while (1) {
    v_left = analog(2);
    v_right = analog(3);

    current_time = mseconds();

    if (count_l >= count_goal) {
        motor(0,0);
    }
    if (count_r >= count_goal) {
        motor(2,0);
    }
    if (count_r >= count_goal && count_l >= count_goal){
        kill_process(pid);
        beep();
        break;
    }

    if (v_left_old){
        if (v_left < threshold - hysteresis){
            count_l = count_l + 1;
            current_time = mseconds();
            speed_l = 1000.0 /(float)(current_time - msec_l);
            msec_l = current_time;
            pow_l = (int)(p_gain_l*(float)(count_goal - count_l) - d_gain*speed_l);
            if (pow_l > 100) pow_l = 100;
            if (pow_l < 65) pow_l = 65;
            if (turn == -1) motor(0, -pow_l);
            else motor(0, pow_l);
            v_left_old = 0;
        }
    }
    else {
        if (v_left > threshold + hysteresis){
            v_left_old = 1;
        }
    }

    if (v_right_old){
        if (v_right < threshold - hysteresis){
            count_r = count_r + 1;
            current_time = mseconds();
            speed_r = 1000.0 /(float)(current_time - msec_r);
            msec_r = current_time;
            pow_r = pow_r + (int)((speed_l - speed_r)*p_gain_r);
            if (pow_r > 100) pow_r = 100;
            if (pow_r < 15) pow_r = 15;
            if (turn == 1) motor(2, -pow_r);
            else motor(2, pow_r);
            v_right_old = 0;
        }
    }
    else {
        if (v_right > threshold + hysteresis){
            v_right_old = 1;
        }
    }
}
```