Trabalho Prático 2

Objetivo

1. O objetivo desse trabalho prático é familiarizar o grupo com a Handy Board e com as principais características do ambiente de programação Interactive C, por meio da utilização de sensores, avaliando montagem e processamento dos sinais medidos.

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

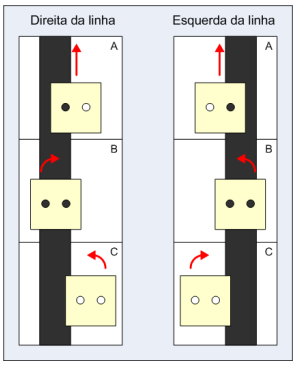
1. Encontramos diversos problemas com os sensores óticos fornecidos ao grupo. Eles não possuem quase nenhum valor semelhante nas medidas de cores e por diversas vezes retornavam valores muito diferentes em situações praticamente idênticas.
2. Dessa forma, as cores que os sensores indicam podem não corresponder muito bem com o bloco analizado naquele instante.
3. Foram encontrados alguns problemas também com os blocos fornecidos. As faces dos mesmos não são semelhantes e caso se aperte um pouco o bloco o sensor verifica outro valor completamente diferente. Sabemos que isso ocorre pois pode-se refletir de forma diferente dessa forma mas mesmo assim, valores um pouco absurdos surgiram neste instante.
4. Na questão do movimento do robô, tivemos dificuldade em colocar os sensores a uma distância boa para ele seguir a linha mas ao fim do trabalho conseguimos estabelecer uma boa distância.
5. Na parte relacionada ao código rodado na HandyBoard, tivemos dificuldade em identificar os estados que o robô pode estar durante seu movimento mas ao final todos os estados foram implementados, inclusive quando o robô se perde e deve fazer uma espiral para encontrar a linha.

Montagem

1. A montagem foi a mesma do Tp1, porém ao invés da caneta, colocamos os sensores óticos (dois) dentro de duas peças de lego e as colocamos abaixo da parte da frente do robô ao invés de abaixo do eixo.

Funcionamento

1. Seguir a linha preta: dois sensores ópticos foram colocados embaixo do robô para que ele fosse capaz de seguir a linha preta. O controle implementado busca sempre manter um dos sensores sobre a parte preta e outro na branca, de forma a seguir a borda da faixa. O robô guarda como um estado a informação de qual borda da linha ele está seguindo (i.e.: esquerda ou direita) e vira para o lado correto quando os sensores mudam de leitura, como na figura abaixo. Caso o robô fique por mais de 10 segundos sem passar pela linha preta, ele entra em movimento espiralado até encontrá-la novamente.

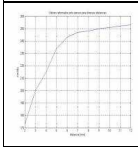
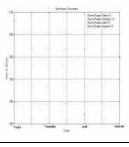
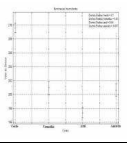
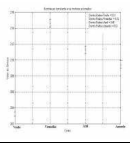


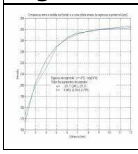
1. Calibrar e identificar as cores: a identificação e calibração de cores é feita baseada em um vetor com 6 posições que guarda a leitura de cada cor (branco, amarelo, vermelho, verde, azul e preto) no momento da calibração. Cada leitura é na verdade uma média entre três medidas subsequentes. Esse vetor é então ordenado, de forma que ele contenha uma escala gradativa das cores. Quando uma leitura é feita, pesquisa-se o vetor procurando a cor com leitura imediatamente acima desta. Posteriormente é calculado um limiar simples para identificar se a leitura feita corresponde a cor anterior ou a cor atual.
2. Procurar a linha preta: primeiro foi preciso estabelecer quando o robô se encontra perdido, ou seja, procurando a linha preta mas não a encontrando-o. Para isso, foi definido que se o robô ficasse girando por mais de 10 segundos ele estaria perdido. Quando perdido, ele começa a andar em espiral até encontrar a linha preta e sair do estado perdido. A função para andar em expiral segue abaixo:

```
int i = 40;
motor(2,100);
while (i<100) {
  motor(0,i);
  sleep(5.0);
  i = i + 40;
}
```

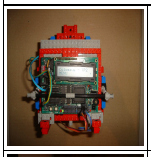
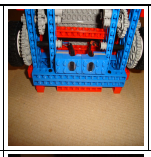
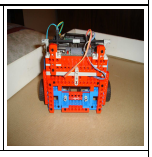
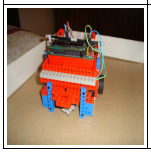
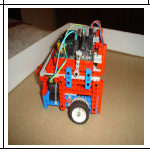
Medições

1. Foram realizadas 10 medidas para os sensores com iluminação constante e variável e depois mais 10 para distâncias diferentes, além das medidas com os motores ligados.
2. Após definida a potência, definimos que menores distâncias possuem melhores medidas e diferenciação de valores.
3. Além disso, foi realizada uma regressão exponencial para o gráfico Sinal x Distância.
4. Abaixo seguem os gráficos das medidas realizadas:

Gráficos			
Sinal x Distância	**Iluminação Constante**	**Iluminação Variável**	**Motor Ligado**
			

Regressão Exponencial


Demonstração

Fotos		
		
		

Exercícios

Normalize Exercise

1. O máximo valor de saída obtido com o sensor foi de 250. a mínima medida feita pelo sensor foi de 7. O código de normal.c fica então:

int normalize(int ligth) { int MAX_LIGHT = 7; int MIN_LIGHT = 250; int output = 100 – ((light – MAX_LIGHT) * 100 / (MIN_LIGHT – MAX_LIGHT); if (output < 0) output = 0; if (output > 100) output = 100; return output; }

2. Feito o experimento com a função normal.c os valores retornados por essa ficaram na feixa de 0 a 100, como era de se esperar.

3. Quando o valor lido pelo sensor é menor do que o valor de MAX_LIGHT obtido na calibração, normal.c retorna um valor maior que 100. Isso ocorre quando a superfície para a qual o sensor está apontado reflete mais luz do que a superfície usada na obtenção de MAX_LIGHT. Quando o valor lido pelo sensor é maior que o valor de MIN_LIGHT obtido na calibração normal.c retorna um valor menor que zero. Nesse situação a superfície para a qual o sensor está apontando reflete menos luz do que a superfície na obtenção de MIN_LIGHT.

Sensor History Exercises

1. Com uma freqüência de amostragem de 7.8125 Hz em 6.4s o buffer do sensor é preenchido.

2. Para fazer com que o robô siga uma parede pode-se fazer uso de 4 sensores de distância. Um em cada lado do robô. Um desses sensores faz com que o robô mantenha uma distancia fixa da parede, os outros 3 são usados para detectar a aproximação de paredes componentes de "quinas". Com esse arranjo o robô tem o comportamento esperado.

3. Stuck.c

int stuck_sensor{ int buffer[50], j; int index = 0;

while (1) {

```
buffer[index] = analog(0);
for ( j = 0; j < 50; j++ ) {
        if ( buffer[j] != buffer[0] ) break;
        if ( j ==49 ) return 1;
}
index++;
if (index == 50 ) index = 0;
```

}

```
No código acima a função stuck_sensor retorna 1 quando o buffer de histórico do sensor apresenta os mesmos \
```