UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Ciência da Computação

Trabalho Prático 2

Controle de trajetória e programação multi-tarefas

GRUPO 6

Alunos: Douglas de Almeida Ferreira
Gustavo Mendes d'Angelis
João Paulo Arruda Amaral

Professor: Mario Fernando Montenegro Campos

Data: 10 de setembro de 2009

1. Introdução

Nesse trabalho foi proposto desenvolver um robô que tivesse a capacidade de realizar algumas tarefas simultaneamente. Em computação, Multitarefa é a característica dos sistemas operativos que permite repartir a utilização do processador entre várias tarefas simultaneamente, um exemplo casual é uma pessoa assistir televisão, ao mesmo tempo em que está almoçando (nesse caso ela está realizando duas tarefas simultaneamente, podemos pensar assim que o cérebro é multitarefa).

Também foi proposto que o robô fosse capaz de realizar rotações para determinados ângulos, que acendesse alguns leds de maneira aleatória, e realizasse deslocamentos de determinadas distâncias. Algumas dessas tarefas seriam realizadas em paralelo.

A incerteza e os erros de atuação será algo presente nesse trabalho, e merecerá uma análise especial. Essa análise será feita através de gráficos e de determinação do desvio padrão.

2. Objetivos

Esse trabalho tem por objetivo:

- Avaliar erros de atuação experimentalmente;
- Desenvolver um controle simples para o robô
- Programar o robô com tarefas concorrentes

3. Tarefa

Foram propostas as seguintes tarefas:

1. Estrutura e Controle: Desenvolver um robô apenas com peças do kit Lego (incluindo as rodas) e dois motores. Esse robô deverá ser controlado apenas pela HandyBoard.
2. Menu: Todas as tarefas deverão ser acessadas facilmente por meio de um menu, do potenciômetro e dos botões.
3. Erros de translação e rotação: Deve ser criado um dispositivo que levante e abaixe uma marcador (pincel atômico, marcador de quadro branco, caneta hidrográfica, etc.) sobre uma cartolina ou papel, no qual o robô trafegará. Dessa forma as trajetórias ficarão claramente marcadas e poderão ser medidas e verificadas com maior facilidade e precisão.
4. Calibração: Realizar uma “calibração” prévia. Para isso faça, pelo menos, 10 testes para cada valor de potência P. Deve-se apresentar uma tabela onde para cada tempo t haja, pelo menos, 10 colunas com valores das distâncias (ângulos) percorridas. Tem-se que plotar um gráfico t x d (t-tempo; d-distância percorrida pelo robô ou theta - ângulo de rotação do robô). (P e t devem ser selecionados no menu). O

gráfico deve mostrar o valor médio e a barra de erro correspondente de um desvio padrão em torno da média.

4. Desenvolvimento

4.1. Parte mecânica

O robô possui dois motores que movimentam cada roda independentemente. Um Roll-on (retirado de um desodorante) serve como roda de apoio (foi utilizado pelo pouco atrito que causou, nesse caso, diminuindo os erros nas medições).

Em relação às reduções, utilizamos uma rosca sem fim ligada diretamente ao eixo do motor que causa uma grande redução na velocidade e aumento considerável do torque, além dessa redução utilizamos mais duas reduções antes de rodar a roda,

Utilizamos muitas reduções nesse trabalho, pois essa foi umas das principais dificuldades encontradas no trabalho prático 1. Apesar da velocidade de deslocamento do robô ter ficado bastante lenta, ganhamos um controle maior sobre o robô, tendo notado que ele é pouco afetado pelo atrito ou deslizamentos que podem ocorrer.

Outra modificação clara em relação ao robô do primeiro trabalho foi a diminuição da altura do centro de gravidade. Nesse robô, colocamos a Handyboard (objeto mais pesado), mais perto do chão, e o robô é claramente mais comprido, com a intenção de deixá-lo mais o mais equilibrado possível. Esse equilíbrio foi obtido com perfeição.

4.2. Programação

O funcionamento do programa é bem simples e objetivo. Ao iniciá-lo o botão START deve ser pressionado para que entre no Menu. Neste temos várias opções como modo de teste: onde as calibragens foram feitas, podendo ser passadas à handyboard os valores desejados da potência do motor e do tempo de funcionamento; modo de translação: cumprindo a especificação do trabalho prático, onde o robô possui 3 medidas para deslocamento retilíneo, 10 cm, 20 cm e 30 cm; modo de rotação: semelhante ao anterior, existe 3 opções de angulos, 30, 60 e 90 graus; modo de leds: demonstração dos 4 leds em funcionamento; modo multi-tarefa: nesta etapa duas threads são criadas e uma fica responsável pelos leds enquanto outra é encarregada de fazer o robô andar em uma trajetória livre, durante exaatos 30 segundos; e o último mas não menos importante, o modo da tarefa final, onde o robô realiza um trajeto formando 3 quadrados concêntricos (aproximadamente) à sua esquerda e depois 3 circunferências também aproximadamente concêntricas à sua esquerda.

Todas as opções do menu e da interface gráfica da handyboard são facilmente acessadas utilizando os botões START, STOP e o potenciômetro. Também é possível realizar vários modos em sequência, já que após terminar um, o menu é acionado e pode-se escolher outro.

4.3. Reuniões

24 de Setembro de 2009

Pontapé inicial do trabalho. Após todos os integrantes lerem a especificação, foi discutido opções para os protótipos. Ficou decidido que o robô terá 3 rodas sendo que 2 serão tracionadas por motores. Foi decidido que a roda dianteira serviria apenas como um apoio, para tal foi pensado em um apoio que deixaria a roda livre pra se alinhar com o movimento do robô.

Para as reduções, inicialmente foi utilizado a rosca sem fim e mais 2 reduções em série, o movimento ficou lento, entretanto, manteremos essa configuração já que não existe limitação e tempo.

27 de Setembro de 2009

Nessa reunião terminamos a estrutura do robô e começamos a programação. Nesse momento não tínhamos a definição da terceira "roda", pois primeiramente colocamos uma roda que não agüentava o esforço e saia, depois colocamos um apoio que gerava muito atrito. Por indicação do Grupo 5, decidimos pegar um Roll-on no laboratório. Como o robô "lembra" um carro de fórmula 1, nomeamos nosso robô: Nelsinho PIC-E, o piloto da moda juntamente com um famoso controlador PIC, que criamos uma nova versão.

30 de Setembro de 2009

Reunião de ajustes finais, testes no laboratório, sobre a cartolina. Todos os testes demoraram bastante. O problema com o eixo do motor atrasou ainda mais, mas por fim conseguimos todos os dados necessários.

4.4. Dificuldades encontradas

A principal dificuldade encontrada em todo trabalho foi a determinação do tempo que o motor deve manter-se ligado em relação aos ângulos e distancias que devem ser percorridos. Além disso, encontramos bastante dificuldade na diferença de potência que os motores devem ter, para realizar circunferências de diferentes círculos.

Essas dificuldades somente foram sanadas com a realização de muitos testes.

Percebemos também que com o passar do tempo e do número de testes, o robô acaba ficando descalibrado, justamente pela carga da bateria ir diminuindo após todos esses testes.

Outra dificuldade foi o apoio para o eixo do motor. Tivemos problemas pois alguns tentativas estavam atrapalhando no movimento do eixo, mas por fim encontramos uma maneira de diminuir o atrito e a influencia no movimento do eixo.

Um dos motores foi outro problema, ele estava com sérios problemas de mal contato, e mesmo com ajustes e soldas realizadas ele continuava dando problemas, mas por fim ele simplesmente voltou a funcionar inexplicavelmente.

5. Resultados e conclusões

Com esse trabalho vimos a dificuldade que é realizar um boa calibração em um robô, existem muitas variáveis que influenciam diretamente nessa calibragem. Algumas delas que podemos citar são:

- Carga da bateria da HandyBoard;
- Atrito;
- Diferenças entre motor;
- etc.

Para diminuir o efeito dessas variáveis, tivemos que realizar muitos testes até encontrar um valor de tempo e potência que tivesse o melhor resultado.

Tivemos uma boa idéia das dificuldades que serão encontradas nos próximos trabalhos, mas em contrapartida temos vários métodos e ferramentas aprendidas nesse trabalho que podem ser usados para nos ajudar a enfrentar todos os problemas.

6. Anexo

6.1. Código do Programa

```
int test(){
  /* test mode variables */
  int power=0;
  int time=0;

  /*setting the engine power*/
  printf("Press START to change the power value\n");
  while(!start_button());
  while(start_button());
  while(!stop_button()){
    printf("Power = %d \n",power);
    power=knob()-100;
    sleep(0.25);
  }

  /*setting the functioning time*/
  printf("Press START to change the time value\n");
  while(!start_button());
  while(start_button());
  while(!stop_button()){
    printf("Time = %d sec\n",time);
    time=knob();
    sleep(0.25);
  }

  /*motor functions*/
  motor(0,power);
  motor(1,power);
  sleep((float)time);
  alloff();

  /*end of function*/
  return 0;
}

int translation(){
  /*translaction mode variable*/
  int distance=0;

  /*setting the distance track*/
  printf("Press START to change the distance value\n");
  while(!start_button());
  while(start_button());
  while(!stop_button()){
    printf("Distance = %d cm\n",distance);
    sleep(0.25);
    if(knob()<85)
      distance=10;
    else if(knob()<170)
      distance=20;
    else
    distance=30;
  }

  /*function*/
  translate(distance);

  /*end of function*/
  return 0;
```

```
}

int translate(int value){
   if(value == 10){
      sleep(2.0);
      motor(0,95);
      motor(1,85);          /*calibrar valores*/
      sleep(2.3);
      alloff();
   }else if(value == 20){
      sleep(2.0);
      motor(0,95);
      motor(1,85);
      sleep(4.6);
      alloff();
   }else{
      sleep(2.0);
      motor(0,95);
      motor(1,85);
      sleep(7.0);
      alloff();
   }
   return 0;
}

int rotation(){
   /*rotation mode variables*/
   int angle=0;

   /*setting the angle rotation*/
   printf("Press START to change the angle
value\n");
   while(!start_button());
   while(start_button());
   while(!stop_button()){
      printf("Angle = %d grads\n",angle);
      sleep(0.25);
      if(knob()<85)
        angle=30;
      else if(knob()<170)
        angle=60;
      else
      angle=90;
   }

   /*function*/
   rotate(angle);

   /*end of function*/
   return 0;
}

int rotate(int value){
   if(value == 30){
      sleep(2.0);
      motor(0,90);       /* calibrar valores */
      motor(1,-90);
      sleep(1.3);
      alloff();
   }else if(value == 60){
      sleep(2.0);
      motor(0,90);       /* calibrar valores */
      motor(1,-90);
      sleep(2.6);
      alloff();
   }else{
      sleep(2.0);
      motor(0,90);
      motor(1,-90);       /* calibrar valores */
      sleep(3.9);
      alloff();
   }
   return 0;
}

int leds(){

   poke(0x1009,0x3c);              /*map spin digital
output*/
   while(!stop_button()){
      printf("Aperte STOP para abortar.\n");
      sleep(1.0);
      poke(0x1008,4<<random(4));
      sleep(1.0);
      poke(0x1008,4<<random(4));
      sleep(1.0);
      poke(0x1008,4<<random(4));
      sleep(1.0);
      poke(0x1008,4<<random(4));
      sleep(2.5);
      poke(0x1008,0);      /*turn off leds*/
   }

   return 0;
}

int walk(){

   /*turn on motors*/
   while(1){
      motor (0,85);
      motor(1,85);
      sleep(6.0);
      motor(1,-85);
      sleep(4.0);
```

```
   }
   return 0;
}

int mtask(){
   /*processes variables*/
   int pidLED=0;
   int pidWALK=0;

   /*creating processes*/
   pidLED=start_process(leds());
   pidWALK=start_process(walk());

   /*time of execution*/
   sleep(30.0);

   /*kill processes*/
   kill_process(pidLED);
   kill_process(pidWALK);
   alloff();
   /*end of function*/
   return 0;
}

int finaltask(){

   /*squares*/
   int sides=0;
   while(sides<12){
      motor(0,95);
      motor(1,85);
      sleep(7.0);          /*ajustar tempo para andar
30 cm*/
      motor(1,-85);
      sleep(3.75);/*ajustar tempo para virar 90
graus*/
      sides++;
   }

   alloff();
   sleep(3.0);

   /*circles*/
   motor(0,17);
   motor(1,100);
   sleep(141.0);

   alloff();

   return 0;

}
```

```
int main(){
   /* knob value */
   int value = 0;

   printf("Press START to go Menu! \n");
   while(!start_button());
   printf("START = Confirm\nKNOB = Change
Option");
   /* time for visualization */
   sleep(5.0);
   while(start_button());

   /*choise of modes*/
   while(1){
      while(!start_button()){
         sleep(0.3);
         if(knob()<45){
            value=0;
            printf("Test Mode.\n");
         }
         else if(knob()<90){
            value=1;
            printf("Translaction Mode.\n");
         }
         else if(knob()<135){
            value=2;
            printf("Rotation Mode.\n");
         }
         else if(knob()<180){
            value=3;
            printf("Leds Mode.\n");
         }
         else if(knob()<225){
            value=4;
            printf("Multitask.\n");
         }
         else{
            value=5;
            printf("FinalTask Mode.\n");
         }
      }
      while(start_button());

      if(value==0)
        test();
      if(value==1)
        translation();;
      if(value==2)
        rotation();
      if(value==3)
        leds();
      if(value==4)
        mtask();
```

```
    if(value==5)                                    return 0;
      finaltask();                                }
}
```

6.1. Fotos do Robô