ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos

PMR 3408 Instrumentação

Experiência sobre

7 bot, Processing, Kinect e Leap Motion

Objetivo

O objetivo dessa experiência é utilizar o braço robótico 7 bot para projetar com um laser um quadrado pré determinado na parede do laboratório, utilizando seguintes quatro fontes de entrada para os ângulos de rotação da base e inclinação do braço:

1) Software Processing e mouse;
2) Leap Motion;
3) Kinect;
4) Cinemática inversa e lista de ângulos com passo de 4 cm.

Nome_____ nºUSP_____
Nome_____ nºUSP_____
Nome_____ nºUSP_____
Nome_____ nºUSP_____
Nome_____ nºUSP_____

1. Cinemática direta e inversa

Conforme visto na matéria de mecanismos, a cinemática direta consiste em determinar a posição do efetuador de um mecanismo a partir dos ângulos/posições das articulações. De maneira análoga, a cinemática inversa tem por objetivo determinar os ângulos/posições de articulações a partir da posição do efetuador.
Entretanto, pode-se desejar calcular a projeção da direção de um braço robótico em um determinado plano, projetando por exemplo um laser. Esse será o caso de nossa experiência. A figura 1 detalha como pode ser determinada, a partir de trigonometria, a cinemática direta e inversa para que o braço robótico 7bot projete um laser no quadrado desenhado na cartolina, fixada na parede do laboratório.

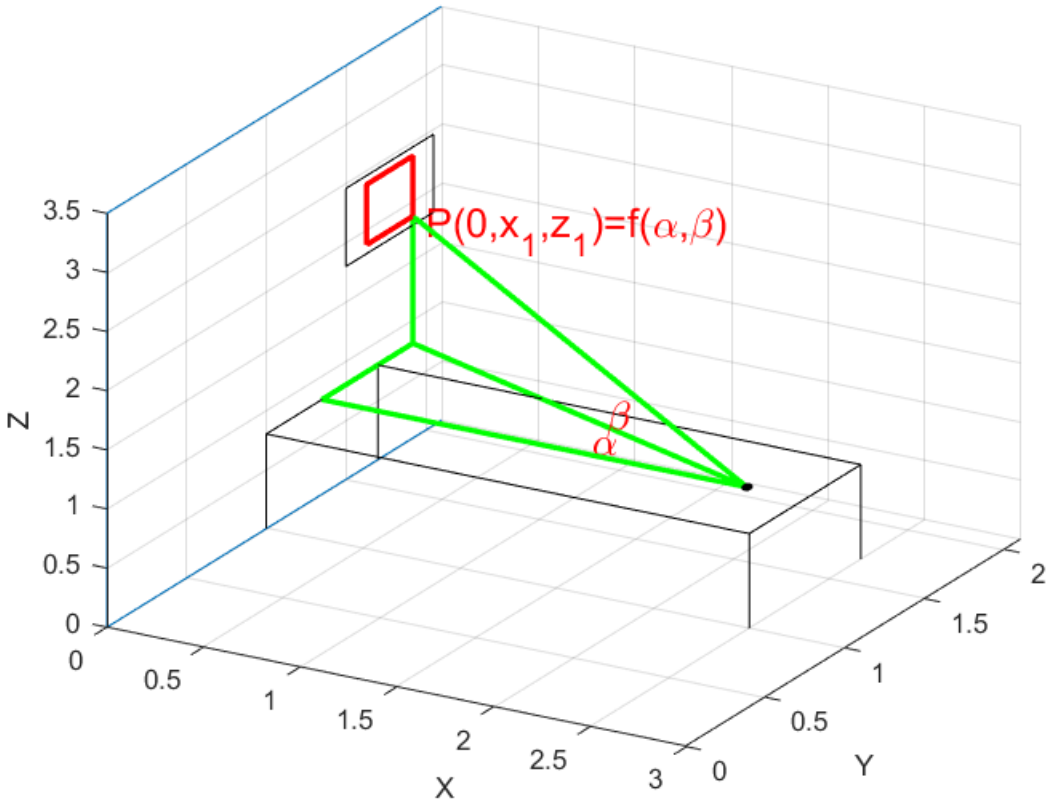

Figura 1 – Diagrama para cálculo da cinemática direta e inversa, **disponível no arquivo de Matlab na área de trabalho**.

Neste momento vocês devem dividir seu grupo em dois subgrupos: A e B.
O grupo trabalho do grupo A irá valer 3 pontos e consiste em:
1) determinar as equações das cinemáticas direta e inversa, relacionando as coordenadas Y e Z do ponto projetado P a partir dos ângulos α e β,
2) escolher uma origem e uma posição para o 7bot, conforme a Figura 1, e medir as distâncias necessárias usando a trena e a régua disponíveis;
3) determinar 2 listas das coordenadas Y e Z que gerem a trajetória pré determinada do quadrado desenhado na cartolina, uma com distância entre os pontos de 20mm e outra com 200mm;
4) usar a cinemática inversa para obter listas dos ângulos de rotação da base e de inclinação para as duas listas de posições geradas;
5) Implementar as duas listas em vetores dentro do software Arduino;
6) Fixar o laser no 7bot utilizando fitas e/ou cinta plástica de fixação (ou popularmente conhecido por laça gato ou enfoca gato), implementar e rodar o programa, fazendo a trajetória desejada.

Enquanto o subgrupo A faz as tarefas 1 a 5, o subgrupo B irá continuar este tutorial. O grupo deve estar junto na execução da tarefa 6!

As deduções das cinemática direta e inversa, as medidas importantes aos cálculos e as listas de pontos e ângulos, referentes as etapas 1 a 4, devem ser passadas a limpo no quadro abaixo.

Valor: 3.0 pontos

2. Processing

O Processing foi criado em 2001 por Casey Reas e Bem Fry, no grupo de computação e estética do MIT, com o objetivo de ser usado como uma ferramenta para incentivar o aprendizado de linguagens de programação. O Processing foi desenvolvido para ensinar os fundamentos da programação ao se criar imagens, animações e interações entre o usuário e as aplicações gráficas.

Entretanto, o Processing evoluiu em uma ferramenta de uso profissional, adotada por uma grande quantidade de estudantes, artistas, designers, pesquisadores e hobbyistas.

Como é baseado em Java, pode rodar em qualquer Sistema Operacional que tenha a JVM instalada. Uma das grandes vantagens do Processing é o fato de possuir todas as bibliotecas abertas já desenvolvidas para Java, como OpenGL ou mesmo protocolos de comunicação com o Kinect ou Leap Motion.

O programa pode ser baixado do site http://processing.org/.

Vale ressaltar que um dos principais fatores para o sucesso do Arduino foi sua IDE (Integrated Development Envirolment), *inspirada* na IDE do Processing. Ambas podem ser vistas na Figura 2.

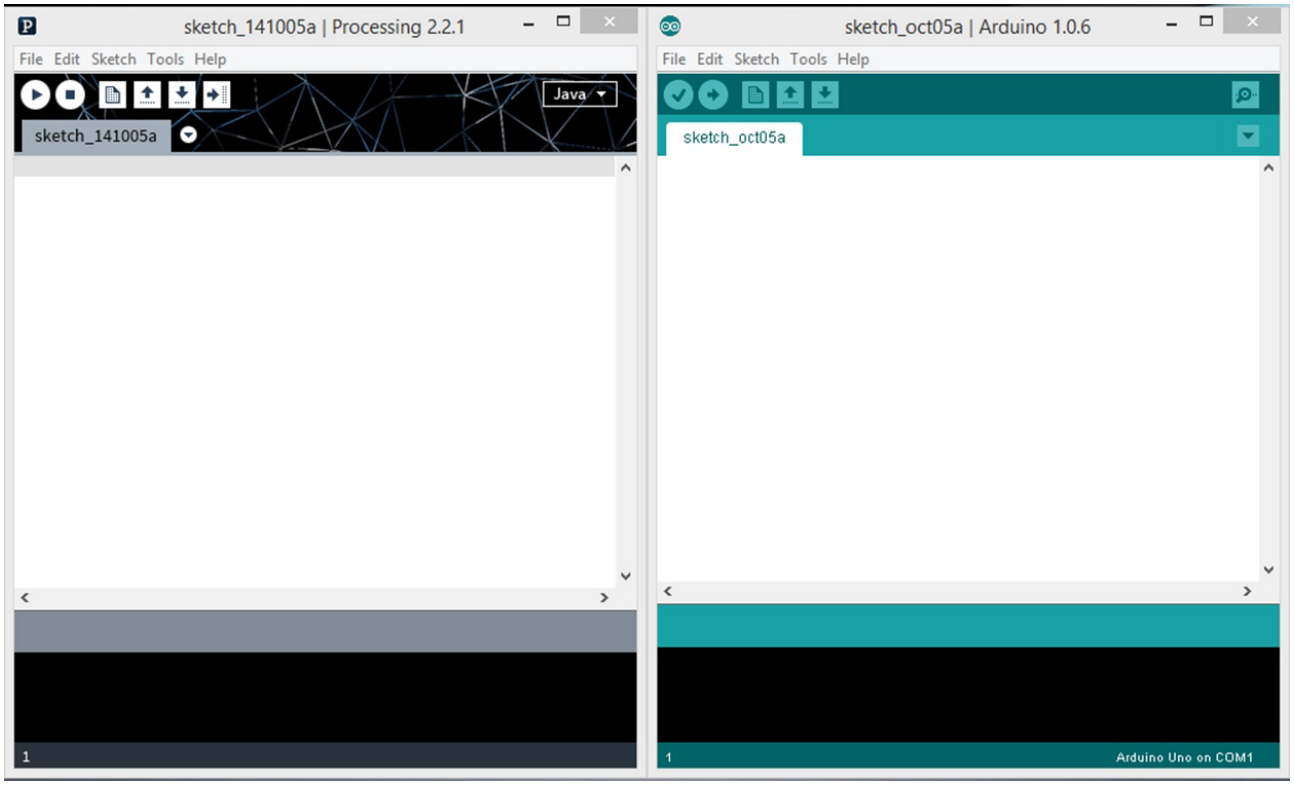

Figura 2 – IDE do Processing e do Arduino.

A IDE possui diversos elementos. O maior e mais visível é a área de editor de texto, parte branca, no qual o código é digitado. Logo abaixo, encontra-se o console, em preto, no qual são mostradas mensagens de erro de compilação e a saída do *println* durante a execução do programa.

Ainda há, no topo, os botões de comando e uma barra de controle de abas. As funções dos botões são, respectivamente, run,stop, new, open, save e export.

O código de um programa em Processing é chamado de sketch. A estrutura básica de um sketch é extremamente similar à do Arduino e conta com três partes: o escopo de variáveis globais, a função *setup()* e a função *draw()*.

O Escopo de variáveis é a parte na qual são declaradas as variáveis globais do sketch, acessíveis de qualquer função. Vale relembrar que existem oito tipos de dados em Java aceitos no Processing: byte, short, int, long, float, double, boolean e char.

A função *setup()* é chamada uma vez apenas no início da execução do programa. É nela que se epecifica, por exemplo, o tamanho da janela com a função *size()*.

Por fim, a função *draw()* será executada em loop até que o programa seja terminado, uma vez para cada *refresh* de tela. É nela que se inclui a função background() para determinar a cor do fundo da aplicação.

Seguem alguns exemplos básicos:

2.1 Círculo se movimentando

Este primeiro sketch tem por finalidade demonstrar as funções mais básicas de um programa. Para isso, uma janela em branco será criada, na qual um círculo irá se movimentar horizontalmente (Figura 3).

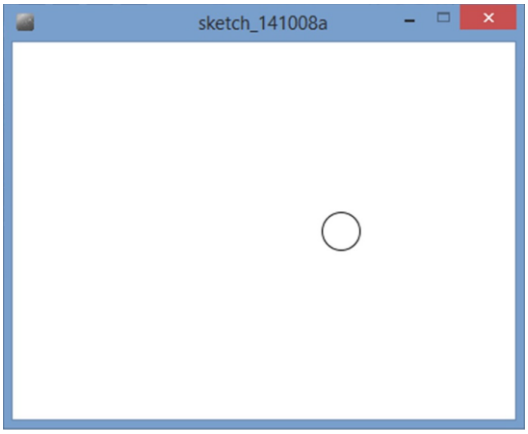

Figura 3 – Círculo se movimentando.

O código consiste em:

```
int timer;
void setup(){
size(400,300);
}
void draw(){
background(255);
ellipse(timer,height/2, 30,30);
timer = timer + 1;
}
```

No início está declarada a variável timer. Esta irá aumentar a cada vez que a função *draw()* for executada, servindo de posição do centro do nosso círculo e sendo então responsável pela translação do mesmo.

A função *size(width, height)* serve para criar a janela, com seus argumentos sendo a largura *width* e a altura *height*, em pixels.

A função *background()* serve para definir a cor de fundo da janela. Esta deve ser determinada sempre na função *draw()*.

Por fim a função *ellipse* serve para desenhar nosso círculo na tela. Seus argumentos são compostos por dois pares: seu centro e seus dois diâmetros principais.

Como sugestões,

a) utilizando a função *rect(x,y,width,height)* substitua nosso círculo por um quadrado;

b)modifique a velocidade;

c) modifique novamente o código para que o quadrado, ao sair totalmente pela direita, apareça na esquerda, estando então SEMPRE presente na tela, em um movimento contínuo.

2.2 Gráfico em tempo real

O código consiste em:

```
ArrayList<Points> curva = new ArrayList();
void setup() {
   size(500, 200);
}
void draw() {
   background(-1);
   noFill();
   stroke(0);
   beginShape();
   for (int i=0;i< curva.size();i++) {
      Points P = (Points) curva.get(i);
      vertex(P.x, P.y);
      if (P.x<10) curva.remove(i);
      P.x--;
   }
endShape();
}
void keyPressed() {
   float t = random(20, height-20);
   Points P = new Points(width-10, t );
   curva.add(P);
}
class Points {
   float x, y;
   Points(float x, float y) {
   this.x = x;
   this.y = y;
   }
}
```

O comando da primeira linha do código cria uma nova lista chamada curva. Toda vez que uma tecla é pressionada, o evento acontece e a função *keyPressed()* é chamada. Nela, são definidas as coordenadas x e y de um ponto P: x será a largura da janela menos 10; e valor y será encontrado randomicamente, entre 20 e a altura da nossa janela menos 20. Esse ponto é então adicionado à lista curva, através do comando *curva.add(P)*.

Na função *draw()* a curva á sempre desenhada ponto a ponto, dentro do escopo definido por *beginShape()* e *endShape()*. Por fim, o valor d e x de todos os pontos é reduzido em um pixel, dando a sensação de movimento a curva. Se este tem valor negativo, é retirado da lista curva através de *curva.remove(i)*.

2.3 Interação com o arduino

Agora você possui o conhecimento necessário para compreender a interação entre Processing e Arduino do modo que o politécnico mais gosta: através de exemplos.

Desta forma, utilize o Arduino Due disponível na bancada para rodar e compreender os códigos dos exemplos 1 a 3, dentro da pasta *Processing Sketches* localizada no Desktop.

3. Leap Motion

O Leap Motion, figura 4, é um dispositivo capaz de localizar a posição de suas mãos e dedos no espaço. O dispositivo usa três LEDs infravermelhos e duas câmeras, localizando os pontos de cada articulação de sua mão através de estereoscopia com uma precisão de até 0.7mm no espaço.

Atualmente o Leap Motion vem sendo utilizado anexado em vários visores de realidade virtual, com o intuito de reproduzir fielmente a posição e movimento das mãos do usuário.

Figura 4 – Dispositivo Leap Motion e exemplo de aplicação.

Apesar de existir uma SDK disponível para o uso do Leap Motion, iremos utilizar uma biblioteca do Processing.

Desta forma, execute os exemplos de 4 a 7, disponíveis dentro da pasta *LeapMotion Processing* e implemente, **valendo um ponto**¸ um programa similar aos sensores de ré de carro, no qual um beep, emitido por um buzzer, vai tocando intermitentemente com a frequência dos beeps aumentando enquanto sua mão se aproxima do dispositivo. Salve seu código no desktop e explique seus códigos para Arduino e Processing no quadro abaixo.

Valor: 1.0 pontos
Descrição do dispositivo de proximidade baseado em Leap Motion e *buzzer*

4. Kinect

O Kinect, figura 5, é um dispositivo capaz de localizar a posição de suas articulações corporais no espaço, especialmente conhecido devido ao uso no videogame Xbox. O dispositivo utiliza um projetor de pontos infravermelhos, uma câmera comum e uma câmera de profundidade. Este foi originalmente desenvolvido para a Microsoft pela companhia israelense PrimeSense, adquirida pela Apple em 2013.

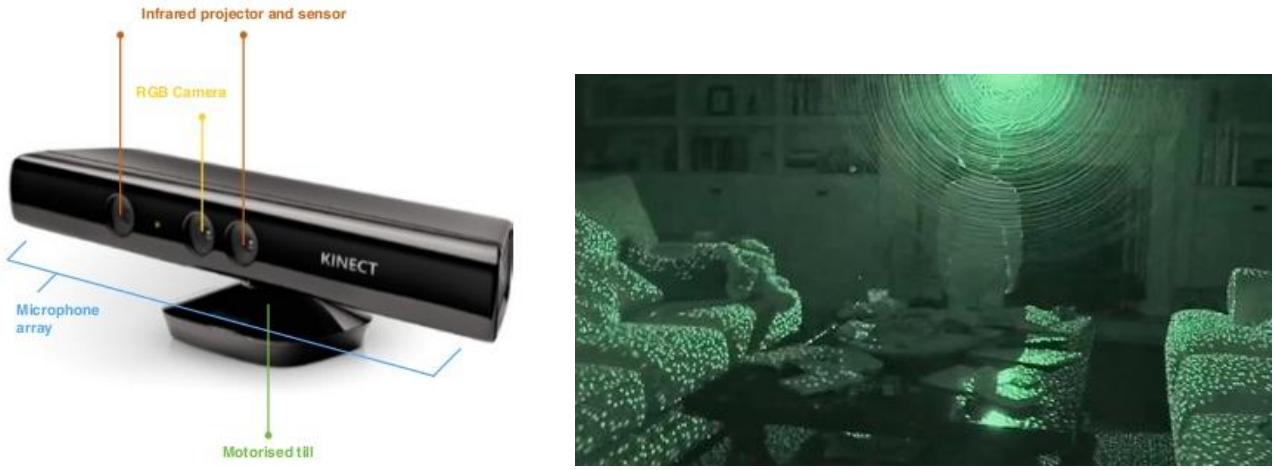

Figura 5 – Dispositivo Kinect e padrão de projeção dos pontos infravermelhos.

Apesar de existir uma SDK disponível para o uso do Kinect, iremos utilizar uma biblioteca do Processing.

Desta forma, execute o exemplos 8, disponível dentro da pasta *Kinect Processing*, entenda o program e, **valendo um ponto**¸ responda abaixo.

Valor: 1.0 pontos
Responda as seguintes questões sobre o exemplo 8:
1) Qual o nome da variável que possui o valor de x da posição do punho direito? E da posição y? Porque há uma multiplicação pela largura e pela altura?
2) Porque o valor de saída é mapeado entre 0 e 255? Comente o mapeamento, fazendo as alterações necessárias e comente o que acontece com o código. Porque isso ocorre?

5. 7 bot

O braço robótico 7Bot foi desenvolvido e vendido na plataforma *Kisckstarter*. O 7bot possui 6 graus de liberdade, estrutura de alumínio e seis servomotores digitais de alto torque (42kgf.cm@7.4V) e alta velocidade (0.17 seg/60º). Cada motor possui controle PID ajustado individualmente, aumentando sua velocidade e estabilidade e um cabo de feedback, permitindo gravar as posições do braço robótico para posterior reprodução. A figura 6 detalha o ID de cada motor do 7 Bot.

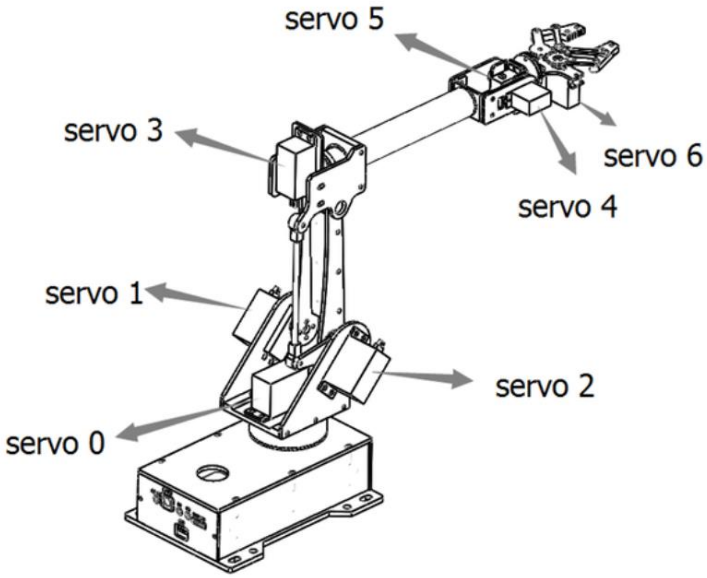

Figura 6 – Identificação dos graus de liberdade do braço robótico 7Bot.

Execute, entenda e estude os exemplos 9 e 10, disponíveis na pasta *7 Bot processing*. O exemplo 10 controla o 7 Bot utilizando o *Leap Motion*. Valendo um ponto, responda as questões abaixo.

Valor: 1.0 pontos
Responda as seguintes questões sobre o exemplo10:
1) Há alguma limitação no uso do Leap Motion dentro do Processing para o controle do 7 Bot?
2) Com base no manual do 7 Bot (*Getting Started with 7Bot*), página 17, você consegue descrever se é possível utilizar uma das funções descritas para melhora o desempenho desta interface 7 Bot – *Leap Motion Processing*?

6. Tudo Junto

Chame seus colegas do subgrupo A.
Implementem juntos, valendo mais 3 pontos, o programa em Arduino que faz o 7 Bot projetar o quadrado na parede, sobre o desenhado na cartolina, utilizando primeiro a lista de ângulos calculados pelo subgrupo A e depois através do Kinect.
Não esqueça de mostrar ao professor para que este contabilize os pontos.

7. Finalização

Valendo o último ponto, faça um resumo de tudo que foi aprendido e deixe opiniões, comentários e sugestões para que o professor possa melhorar esta experiência para os próximos anos.

Valor: 1.0 pontos
Resumo do aprendizado com a experiência e sugestões.