departamento de eletrónica, telecomunicações e informática

Curso	8204 - Mestrado Integrado em Engenharia Eletrónica e Telecomunicações
Disciplina	47243- Sistemas e Controlo I
Ano letivo	2017/18

Relatório

RT060 – Sistema de Controlo de Posição

Autores:

79965	Bruno Miguel Silva Santos
79970	Rui Filipe Santos Carapinha
Turma	P6

Data	4/11/2017
Docente	Telmo Reis Cunha

Introdução

O sistema de controlo de posição denominado RT060 oferece um sistema de controlo onde é possível compreender os fundamentos da engenharia de controlo. Esta consiste num cursor que move horizontalmente sobre uma calha.

Tem como principio básico um transdutor (motor DC) que converte energia elétrica em energia mecânica de rotação, que vai atuar sobre um eixo que se desloca fazendo variar a posição do ponteiro sobre a régua. O controlo da posição é feito com recurso a um potenciómetro, que em função da posição do seu ponteiro vai ter diferentes valores de resistência e assim é possível saber a posição do ponteiro ao longo da calha.

O controlo deste sistema é realizado através de ligação USB e com recurso a funções dedicadas de MATLAB. Uma das funções oferecidas permite especificar o valor de tensão que é aplicado ao motor DC (que faz varia numa gama de -5V a 5V) enquanto a outra permite ler a posição atual do cursor.

Descrição do Problema e Objetivos

O objetivo principal deste trabalho era desenvolver funções base para a máquina RT060. Posto isto, foram realizadas duas funções (RT060_GotoPosition(ReferencePos) e RT060_GotoPosition_V2(ReferencePos)). As duas funções têm como objetivo levar o cursor à posição indicada por *ReferencePos*. A única diferença entre as duas funções é que a segunda função devolve o vetor de medidas de posição e o respetivo vetor de instantes de tempo.

Na 2ª parte deste trabalho o objetivo era a obtenção de um modelo do sistema de segunda ordem, por resposta em frequência. Neste ponto, foram dados valores à tensão a aplicar ao motor DC em frequência (Utilizou-se para este efeito 9 frequências, de 0.1Hz a 0.9Hz) e foram retiradas as devidas medições e feita a devida análise para cada frequência. Com recurso ao MATLAB, foi possível obter a amplitude, a fase, a média do valor, o ganho, o gráfico da tensão ao longo do tempo e o gráfico da posição ao longo do tempo. Com isto, foi possível estimar os parâmetros da função transferência do modelo que representa o sistema.

Procedimento

1.Parte

Numa primeira abordagem tentámos achar uma relação entre a tensão que oferecíamos ao motor e o erro de medição, rapidamente verificámos que este erro só descia para valores aceitáveis quando usávamos tensões baixas. Por essa razão, decidimos fazer com que quando o cursor estiver perto do valor pretendido (cerca de 30mm da *ReferencePos*) diminuir a tensão dada ao motor DC (para um valor baixo) de modo a que este pare no valor pretendido. Neste passo, tivemos de por um intervalo de erro, senão o cursor nunca pararia no sitio pretendido (andava sempre a tentar corrigir a posição, até chegar ao valor exato de *ReferencePos*).

Na segunda parte era necessário desenvolver um programa que retirasse medições de posição e de tempo, para esse efeito foi usada uma das função dedicadas à máquina (RT060_GetPosition()) e dos comandos do MATLAB denominados TIC e TOC. De modo a facilitar a análise dos resultados obtidos, registamos os valores obtidos em vetores posição, tempo e tensão no motor.

Por fim, desenvolveu-se um programa que demonstrasse a posição em função do tempo quando o cursor começava em 10mm, evoluía para os 100mm, 50mm, 200mm e 150mm. De modo a termos medições de tempo o mais coincidentes possível com as medições de posição decidimos concatenar os vários vetores que a rotina RT060_GotoPosition_V2() ofereceu.

2. Parte

Nesta segunda parte, decidimos *a priori* que a melhor abordagem a ter seria não considerar o primeiro período (pois este podia ser afetado pelo regime transitório). Posto isto decidimos então realizar três períodos e considerar apenas o último para cada medição/cálculo efetuado.

Começamos por criar um sinal em MATLAB sinusoidal V=2*sin(2*π*f_K) e aplicamos esse sinal ao motor, registando os valores de posição do ponteiro e tempo. Calculamos a fase calculando a diferença de tempos entre picos, sabendo que um período corresponde a 360º, então $\varphi = \frac{\Delta t}{2\pi}$ e o ganho $\frac{\Delta x}{V_p}$.

Repetimos este procedimento para diferentes frequências.

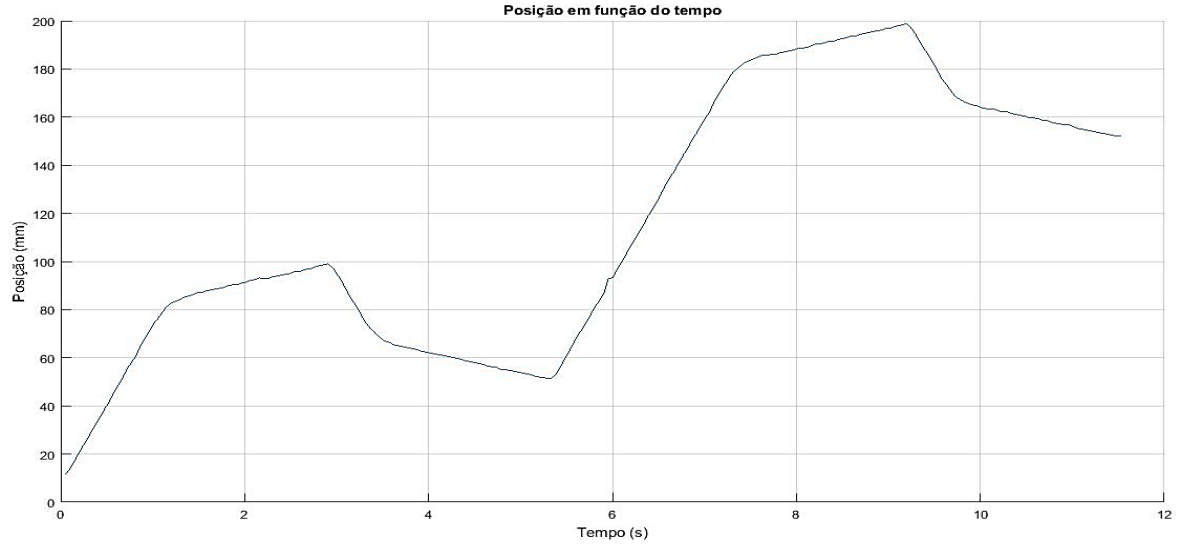
De modo a testar a função transferência que representava o modelo do sistema, aplicamos um sinal DC durante um tempo determinado de forma a simular um degrau. Registamos a posição do ponteiro ao longo da régua.

Depois simulamos o sistema, através da ferramenta do MATLAB, aplicando degraus unitários à função transferência e analisando a saída.

Resultados e Análise

1.Parte

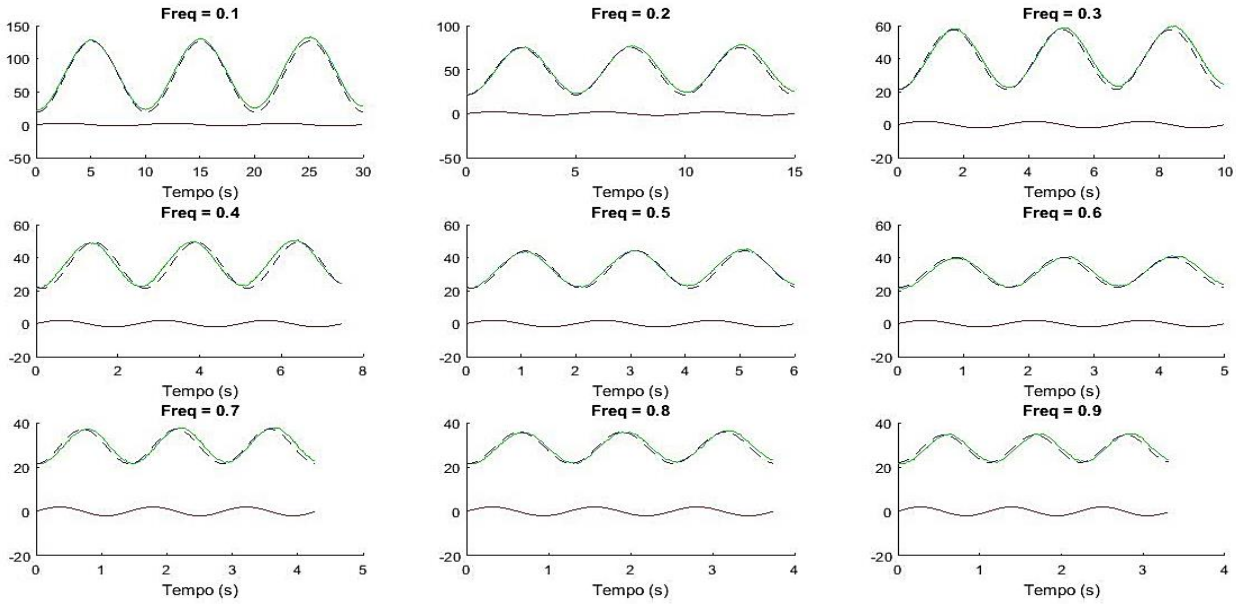
Nesta primeira parte, a evolução no tempo do cursor ao longo da calha obtida foi a seguinte:



A evolução que obtivemos coincide com a evolução esperada, podemos verificar que o cursor mudou de posição logo que atingiu os 100mm, 50mm, 200mm e os 150mm. De salientar, que o decréscimo no declive quando o cursor está próximo do valor pedido acontece devido à implementação que decidimos dar ao programa de modo a este ser mais preciso.

2.Parte

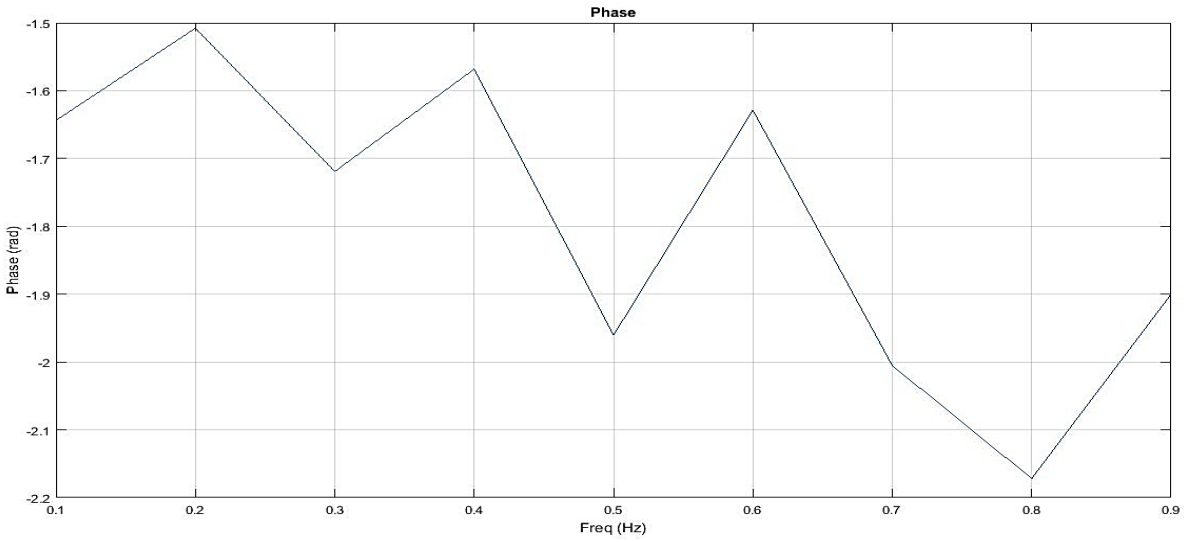
2.1. Quando analisamos os gráficos posição e tensão no motor em função do tempo, foi possível observar que a posição do ponteiro é uma sinusoide com a mesma frequência, mas com uma diferença de fase φ e um ganho associado.



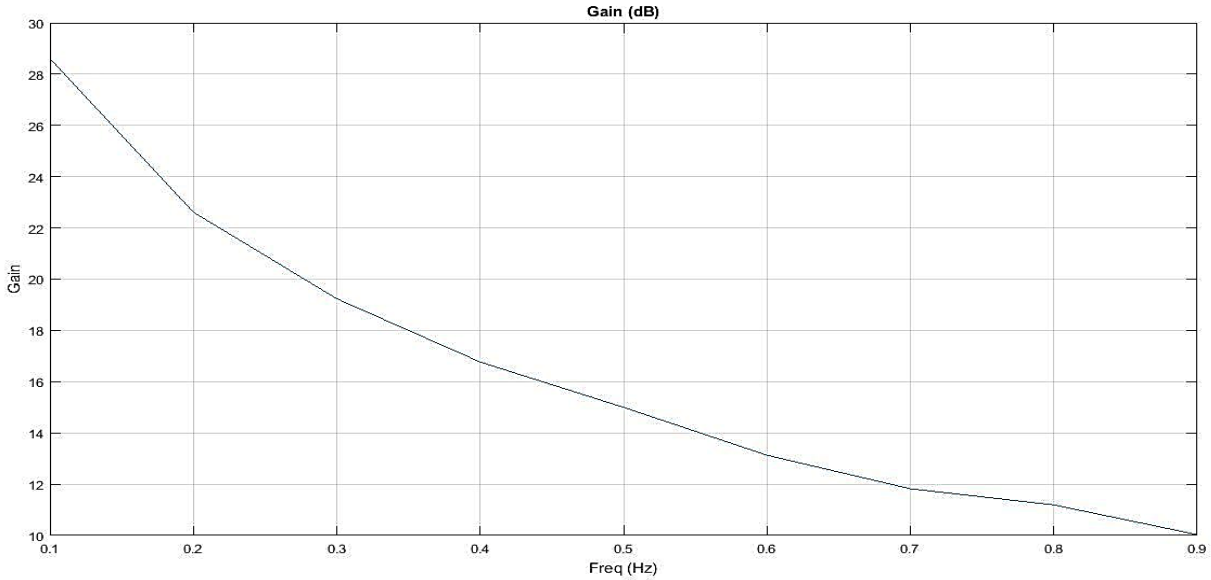
Nestes gráficos, podemos verificar a posição obtida (representada a verde), a posição teórica (representada a azul) e a tensão aplicada (representada a vermelho) para cada frequência utilizada.

Esta é uma das características dos sistemas de segunda ordem, a resposta a um sinal sinusoidal é um outro sinal sinusoidal com uma diferença de fase associada.

O gráfico obtido da fase foi o seguinte:



O gráfico obtido do ganho foi o seguinte:

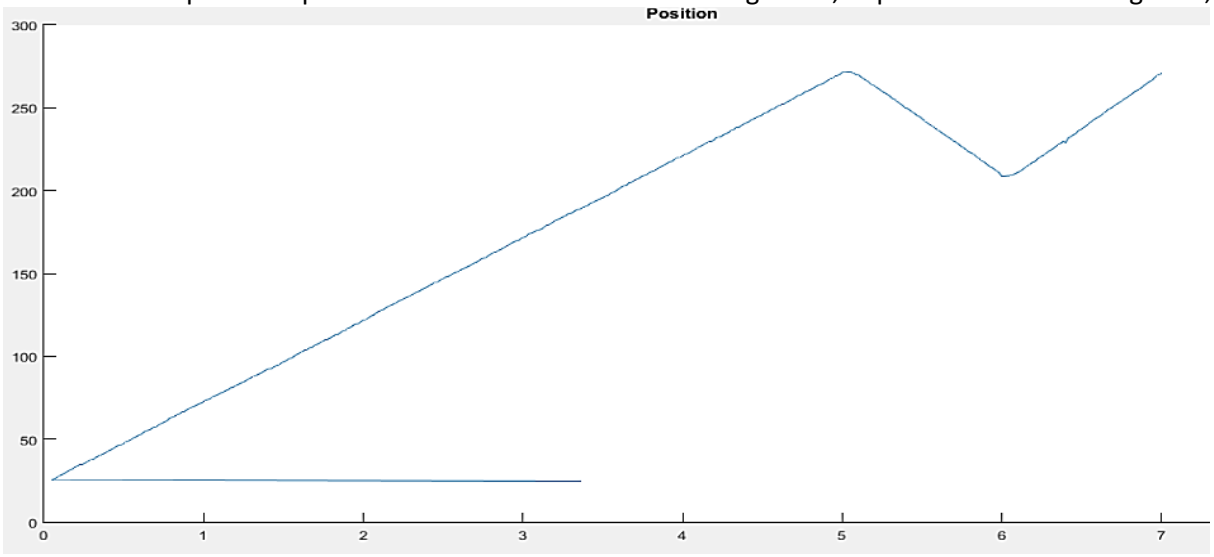


Pela análise do gráfico, facilmente, se verifica que quando menor a frequência utilizada maior será o nosso ganho.

2.2. De forma a modelar este sistema por uma função transferência de segunda ordem, temos de analisar o gráfico do ganho em função da frequência, pois os polos da função transferência são quando o ganho do sistema baixa 20dB/década.

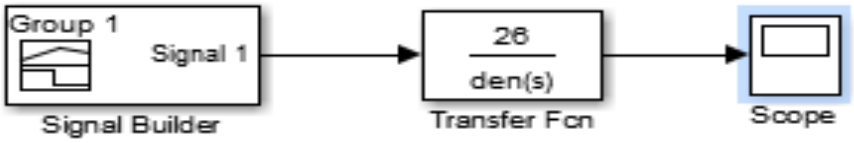
Se calcularmos o declive do ganho em função de ω, em escala logarítmica, obtemos um declive $m= \frac{y2-y1}{\log\left(\frac{w2}{w1}\right)} = \frac{18}{0.95} \approx 20$ dB/dec. Significa que vai ter um polo em $1+\frac{s}{2\pi 0.1}$. A partir da frequência 0.8 verificamos que o ganho diminuía ainda mais do que tinha vindo a diminuir, por isso assumimos que fosse um polo. Portanto, a nossa função transferência ficava desta forma: $G(s)= \frac{k}{(1+\frac{s}{2\pi 0.1})(1+\frac{s}{2\pi 0.8})}$ em que o K era o limite quando o $s \to 0$, ou seja, $20\log(k) = 28.3 \leftrightarrow k = 26$

2.3. Nesta fase, o objetivo foi testar a evolução do sistema e do modelo, quando este partia dos 10mm e depois era aplicado um 3V ao motor durante 5 segundos, depois -5V durante 1 segundo,

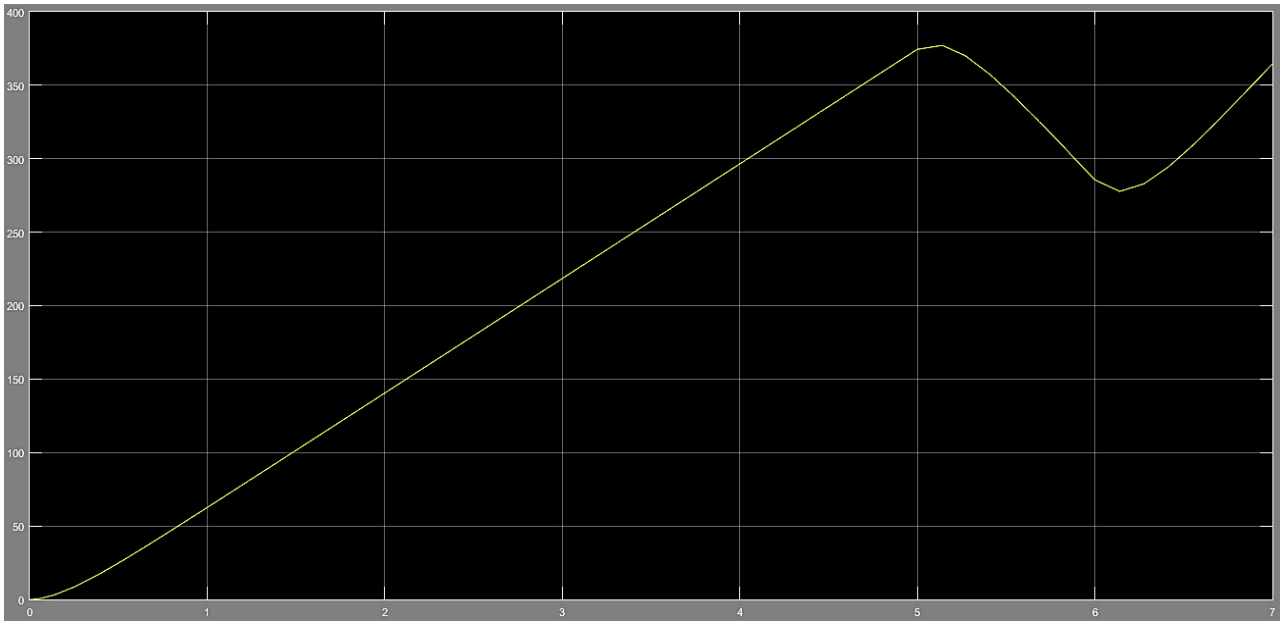


depois 5V durante 1 segundo e, por fim, 0V. O resultado obtido foi este:

Com isto realizado o próximo passo era testar o nosso modelo, para isto decidimos usar a ferramenta SIMULINK disponível no MATLAB. Foi feito o seguinte modelo:



E com base nisto obtivemos o seguinte resultado:



O resultado obtido do modelo em termos de forma de onda assemelha-se bastante ao resultado obtido do sistema, contudo, os valores no eixo yy do modelo e do sistema (que corresponde à posição do cursor) são um pouco dispares.

Conclusão

Com o término deste trabalho foi possível verificar a diferença entre os modelos que são muito utilizados na teoria e os sistemas reais. Os comportamentos obtidos coincidiam com o esperado, ao contrario dos valores absolutos, que por sua vez, tinham uma diferença significativa. Não conseguimos compreender o porquê da fase, a cada vez que corríamos o programa, tinha comportamentos e valores diferentes, mesmo sob as mesmas condições.

Neste trabalho, a maior dificuldade que encontrámos foi encontrar os polos da função transferência do modelo que representa o sistema, pois não obtivemos valores para uma maior gama de frequências, o que limitou a precisão da função transferência.

Anexos

RT060_GotoPosition():

```
function [] = RT060_GotoPosition(ReferencePos)
    RT060_SetMotorVoltage(0);
    while (1)
        if abs(ReferencePos - RT060_GetPosition()) < 30
            Speed = 0.5*sign(ReferencePos - RT060_GetPosition());
        else
            Speed = 5*sign(ReferencePos - RT060_GetPosition());
        end

        RT060_SetMotorVoltage(Speed);
        if abs(ReferencePos - RT060_GetPosition()) < 2
            RT060_SetMotorVoltage(0);
            break
        end
    end
end
```

RT060_GotoPosition_V2():

```
function [Measure,Time] = RT060_GotoPosition_V2(ReferencePos)
    Measure = zeros(1000,1);
    Time = zeros(1000,1);
    index = 0;
    timerVal = tic;
    while (1)
        index = index + 1;

        if abs(ReferencePos - RT060_GetPosition()) < 30
            Vel = 0.5*sign(ReferencePos - RT060_GetPosition());
        else
            Vel = 5*sign(ReferencePos - RT060_GetPosition());
        end

        RT060_SetMotorVoltage(Vel);
        Measure(index) = RT060_GetPosition();
        Time(index) = toc(timerVal);

        if abs(ReferencePos - RT060_GetPosition()) < 2
            RT060_SetMotorVoltage(0);
            Measure(index) = RT060_GetPosition();
            Time(index) = toc(timerVal);
            break
        end
    end
    Measure(Measure == 0) = [];
    Time(Time == 0) = [];
end
```

Primeira Parte:

```
clear
clc
figure
RT060_GotoPosition(10);

[M100,T100] = RT060_GotoPosition_V2(100);
```

```
[M50,T50] = RT060_GotoPosition_V2(50);
[M200,T200] = RT060_GotoPosition_V2(200);
[M150,T150] = RT060_GotoPosition_V2(150);

Measure = vertcat(M100,M50,M200,M150);
T50 = T50+T100(end);
T200 = T200+T50(end);
T150 = T150 + T200(end);
Time = vertcat(T100,T50,T200,T150);
hold on
plot(Time,Measure)
title('Posição em função do tempo')
grid on
xlabel('Tempo (s)')
ylabel('Posição (mm)')
```

Segunda Parte:

```
%2
clear
close all
clc
index = 0;
i = 1;
figure
Gain = zeros(9,1);
Freq = zeros(9,1);
Amp = zeros(9,1);
Phase = zeros(9,1);
Xmean = zeros(9,1);
RT060_GotoPosition(20);
for f = 0.1:0.1:0.9
    data = zeros(1000,3);
    RT060_GotoPosition(20);
    t = 0;
    tic;
    while t < 3/f
        index = index + 1;
        v = 2.*sin(2.*pi.*f.*t);
        RT060_SetMotorVoltage(v);
        data(index,1) = RT060_GetPosition();
        data(index,2) = t;
        data(index,3) = v;
        t = toc;
    end
    subplot(3,3,i)
    hold on
    idx_1 = find(data(:,1));
    idx_3 = find(data(:,3));
    plot(data(idx_1,2),data(idx_1,1),'g')
    plot(data(idx_3,2),data(idx_3,3),'r')
    title(['Freq = ',num2str(f)]);
    xlabel('Tempo (s)')
    Amp(i) = (max(data(round(0.6*length(idx_1)):length(idx_1),1))-
min(data(round(0.6*length(idx_1)):length(idx_1),1)))/2;
    Xmean(i) = mean(data(round(0.6*length(idx_1)):length(idx_1),1));
    Gain(i) = Amp(i)/2;
    Freq(i) = f;
    VoltageIndex = find(data(round(0.6*length(idx_3)):length(idx_3),3) ==
max(data(round(0.6*length(idx_3)):length(idx_3),3)));
    PositionIndex = find(data(round(0.6*length(idx_1)):length(idx_1),1) ==
max(data(round(0.6*length(idx_1)):length(idx_1),1)));
```

```
    Phase(i) = -(data(PositionIndex(1),2)-data(VoltageIndex,2))*2*f*pi;
    PosT = Xmean(i)+Amp(i)*sin(2*f*pi*data(idx_1,2) + Phase(i));
    plot(data(idx_1,2),PosT,'b--')
    index = 0;
    data = [];
    i = i + 1;
end
RT060_SetMotorVoltage(0);
figure
plot(Freq,Phase)
title('Phase')
xlabel('Freq (Hz)')
ylabel('Phase (rad)')
grid on
figure
Gain = 20.*log10(Gain(:));
plot(Freq,Gain)
title('Gain (dB)')
xlabel('Freq (Hz)')
ylabel('Gain')
grid on

%2.3
clear
clc
RT060_GotoPosition(20);
timerVal = tic;
RT060_SetMotorVoltage(3);
i = 1;
t(i) = toc(timerVal);
while t(i) <= 5
    Pos(i) = RT060_GetPosition();
    i = i + 1;
    t(i) = toc(timerVal);
end

RT060_SetMotorVoltage(-5);
while t(i) <= 6
    Pos(i) = RT060_GetPosition();
    i = i + 1;
    t(i) = toc(timerVal);
end

RT060_SetMotorVoltage(5);
while t(i) <= 7
    Pos(i) = RT060_GetPosition();
    i = i + 1;
    t(i) = toc(timerVal);
end

Pos(i) = RT060_GetPosition();
RT060_SetMotorVoltage(0);
plot(t,Pos);
title('Position')
```