Assignment 2: Interrupts for real-time control implementation, and the identification of Coulomb and Static friction

Aaron John Sabu: 170070050

September 13, 2019

1 Overview of the assignment

- To learn programming philosophy of periodic interrupt timer (PIT) interface and why it is necessary
- Read through datasheet of XEP 100 and find out registers to be set and values in them to program interrupt to be generated every sampling time
- Learn to use this interrupt to run a subroutine called interrupt service routine.
- Learn why it is important to use this interface rather than looping continuously without bothering about exact sampling time.
- Open loop program to identify Coulomb and static levels of friction in motor and attached gearbox combined as seen on motor side.

2 Setup and Results

2.1 Part 1

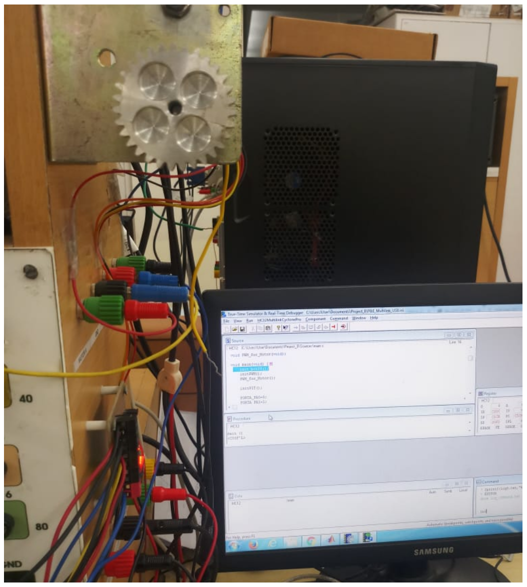

Figure 1: Motor Setup

The above setup was made for the first part of the assignment, wherein the DC micromotor provided in the lab was interfaced with the XEP100 micro-controller and was run at varying duty cycles, hence determining the start and stop duty cycles from which the static friction and dynamic friction of the motor were calculated respectively.
The motor starts rotating when the duty cycle reaches 5 from 0.

$$V_{DC} = 12V \implies V_{sup} = \frac{5}{256} \times 12 = 0.234V$$

Resistance given in datasheet, $R = 5.78\Omega$ Torque constant, $T_C = 34.6mN \cdot mA^{-1}$ Hence, the torque required due to static friction is,

$$T_S = \frac{T_C \times V_{sup}}{R} = \frac{34.6 \times 10^{-3} \times 0.234}{5.78} = 1.403N \cdot m$$

The motor stops rotating when the duty cycle reaches 4 from 7.

$$V_{DC} = 12V \implies V_{sup} = \frac{4}{256} \times 12 = 0.1875V$$

Hence, the torque due to dynamic friction is,

$$T_D = \frac{T_C \times V_{sup}}{R} = \frac{34.6 \times 10^{-3} \times 0.1875}{5.78} = 1.122 N \cdot m$$

2.2 Part 2

The second part used Matlab for solving the three-variable second-order differential equation system which was derived from the Lagrangian using the Euler-Lagrange equation. This ODE system was solved using the state space variable system wherein the ode45 algorithm-based function was utilized. A figure representing the initial state of the pendulum as well as figures representing a few other states are portrayed below:

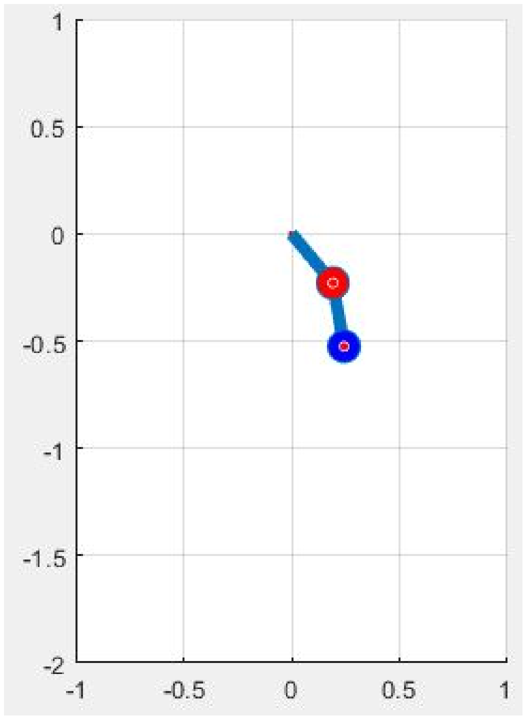

Figure 2: Initial Condition

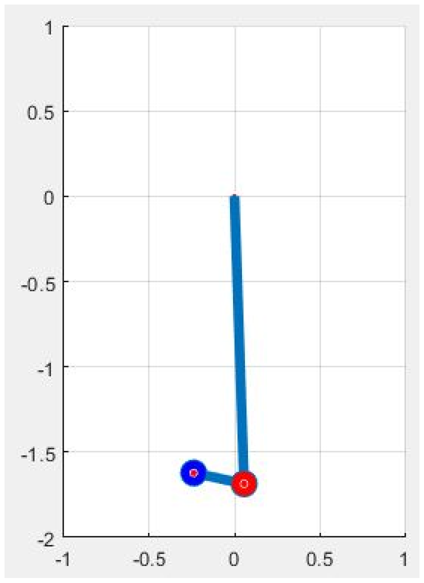

Figure 3: Snapshot 1

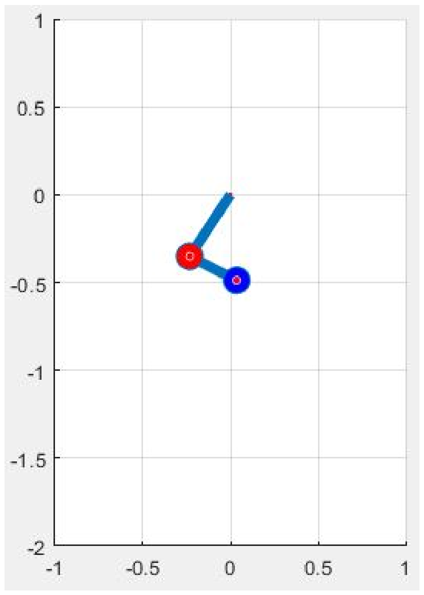

Figure 4: Snapshot 2

3 Codes

3.1 Part 1

MotorCode.txt has been used as the basic code for running the motor:

3.1.1 MotorCode.txt

```
#include <hidef.h>         /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */

int U_DATA = 0;
int L_DATA = 0;
int newDuty = 0;
int timeinsec = 0;
int count = 0;
int En_Read = 0;
float value = 0.00;

void initPIT(void);
void init_bitIO(void);
void initPWM(void);
void PWM_for_Motor(void);

void main(void) {
    init_bitIO();
    initPWM();
    PWM_for_Motor();

    initPIT();

    PORTA_PA0=0;
    PORTA_PA1=1;

    EnableInterrupts;

    for(;;){
        _FEED_COP();
    }
}

void initPIT(void){
    PITCFLMT_PITE = 1;
    PITMTLD0 = 199;
    PITLD0 = 9;
    PITMUX_PMUX0 = 0;
    PITINTE_PINTE0 = 1;
    PITCE_PCE0 = 1;
}

void initPWM(void){
    PWMCLK_PCLK0 = 0;
    PWME_PWME0 = 1;
    PWMPOL_PPOL0 = 1;
    PWMPRCLK = 0x00;
    PWMDTY0 = 0x04;
    PWMPER0 = 0x08;
}

void PWM_for_Motor(){
    PWMCLK_PCLK5 = 0;
    PWME_PWME5 = 1;
    PWMPOL_PPOL5 = 1;
    PWMPRCLK = 0x01;
    PWMDTY5 = 0;
    PWMPER5 = 0xFF;
}

void init_bitIO(void){
    DDRA = 0b11111111;
    DDRB = 0;
}

void interrupt 66 f_pit(void){
    PITTF_PTF0 = 1;
    PORTA_PA2 = 0;
    PORTA_PA3 = 0; // Enable = 0, Select Pin = 0 {SEL = PA2, OE = PA3}
    U_DATA = PORTB;
    PORTA_PA2 = 1;
    PORTA_PA3 = 0; // Enable = 0, Select Pin = 1 {SEL = PA2, OE = PA3}
    L_DATA = PORTB;

    // Combining lower byte and Upper Byte into a 16-bit binary number
    En_Read = (U_DATA<<8)|L_DATA;

    PORTA_PA2 = 0;
    PORTA_PA3 = 1; // Inhibit Logic Reset: Enable = 1, Select Pin = X

    count++;

    value += 0.001;
    timeinsec = ((int)value)%16;
    if(timeinsec < 8) {
        newDuty = timeinsec;
    }
    else{
        newDuty = 15 - timeinsec;
    }
    PWMDTY5 = newDuty;
}
```

3.2 Part 2

The second part of the assignment utilized a number of Matlab scripts for execution. It utilized the **ODEFunction.m** function for solving the ordinary differential equation which was further called in **Animation.m**. The equations pertaining to the system are given below:

$$(m_0 + m_1)\ddot{x} + m_1 l_1(\ddot{\theta}cos(\theta) - \dot{\theta}^2 sin(\theta)) + kx - \frac{kl_0 x}{\sqrt{x^2 + y^2}} = 0$$

$$(m_0 + m_1)\ddot{y} + m_1 l_1(-\ddot{\theta}sin(\theta) + \dot{\theta}^2 cos(\theta)) + ky - \frac{kl_0 y}{\sqrt{x^2 + y^2}} + (m_0 + m_1)g = 0$$

$$l_1\ddot{\theta} + \ddot{x}cos(\theta) + \ddot{y}sin(\theta) + 2\dot{\theta}\dot{y}cos(\theta) + gsin(\theta) = 0$$

3.2.1 ODEFunction.m

```
function [dO] = A2_2_ODEFunction(t,O)

% Developed by Aaron John Sabu for ME6102 under Prof Prasanna Gandhi
% ODEFunction: Solve the ODEs related to a double pendulum
% Fixed pendulum (flexible massless rod) + Mobile pendulum (rigid massless rod)

l0  = 0.5;
l1  = 0.3;
m0  = 0.02;
m1  = 0.03;
k   = 1;
g   = 9.81;
e00 = 40.0*pi/180.0;
e10 = 10.0*pi/180.0;

dO    = zeros(6,1);

dO(1) = O(2);
dO(2) = -(1/(m0+m1))*( (m1*l1*( (dO(6)*cos(O(5)))-(dO(5)*dO(5)*sin(O(5))) ))
    + k*O(1) - k*l0*O(1)/(((O(1)*O(1))+(O(3)*O(3)))^0.5) );
dO(3) = O(4);
dO(4) = -(1/(m0+m1))*( (m1*l1*( (dO(5)*dO(5)*cos(O(5)))-(dO(6)*sin(O(5))) ))
    + k*O(3) - k*l0*O(3)/(((O(1)*O(1))+(O(3)*O(3)))^0.5) + (m0+m1)*g );
dO(5) = O(6);
dO(6) = -(1/(l1))*( dO(5)*(cos(O(5))*dO(3)-sin(O(5))*dO(1) )
    +(dO(2)*cos(O(5))+dO(4)*sin(O(5))) + dO(5)*(dO(1)*sin(O(5))-dO(3)*cos(O(5)))
    + g*sin(O(5)) );

end
```

3.2.2 Animation.m

```
%% Developed by Aaron John Sabu for ME6102 under Prof Prasanna Gandhi
```

```
clc;
clear all;

l0  = 0.5;
l1  = 0.3;
m0  = 0.02;
m1  = 0.03;
k   = 1;
g   = 9.81;
e00 = (-90.0+40.0)*pi/180.0;
e10 = (-90.0+10.0)*pi/180.0;

time  = linspace(0,100,5000)';
IC    = [l1*cos(e00); 0; l1*sin(e00); 0; e10; 0];
[t,y] = ode45('A2_2_ODEFunction',time,IC);

%% Plot of generalised coordinates w.r.t. time

plot(t,y(:,1),'r')
hold on
plot(t,y(:,3),'b')
title('Pendulum with Non-Linear Spring');
xlabel('Time t');
ylabel('Solution y');
legend('theta','r')

%% Animation

figure
O    = [0 0];
axis(gca,'equal');
axis([-1.0 1.0 -2.0 1.0]);
grid on;
origincircle = viscircles(O,0.001);

for j = 1:5
    for i = 1:length(t)
        P1           = [y(i,1), y(i,3)];
        P2          = [y(i,1), y(i,3)]+[l1*cos(y(i,5)), l1*sin(y(i,5))];
        pendulum1   = line([O(1)  P1(1)],[O(2)  P1(2)], 'LineWidth',5);
        pendulum2   = line([P1(1) P2(1)],[P1(2) P2(2)], 'LineWidth',5);
        bob1        = line([P1(1) P1(1)],[P1(2) P1(2)], 'Marker','o',
            'MarkerSize',13,'MarkerFaceColor','r');
        bob2        = line([P2(1) P2(1)],[P2(2) P2(2)], 'Marker','o',
            'MarkerSize',13,'MarkerFaceColor','b');
        ball1       = viscircles(P1, 0.007);
        ball2       = viscircles(P2, 0.007);
        pause(0.0001);
        if i<length(t)
            delete(pendulum1);
            delete(bob1);
            delete(ball1);
            delete(pendulum2);
            delete(bob2);
            delete(ball2);
        end
    end
break;
end
```