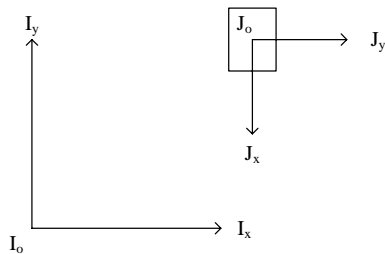
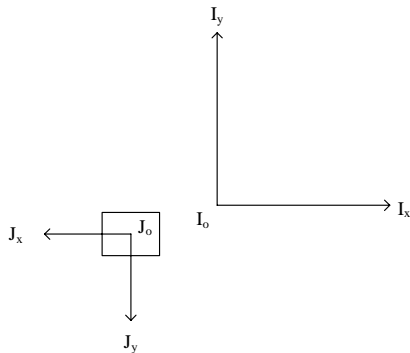


GE 423 Mechatronics Homework Assignment #4
Spring 2009, Due In Lecture April 1st. The Microcontroller Demonstration Check-Off for Questions 5 is due by 3PM Tuesday
March 31st.
Most answers should be typed. Graphs, etc. can be hand drawn if you wish.

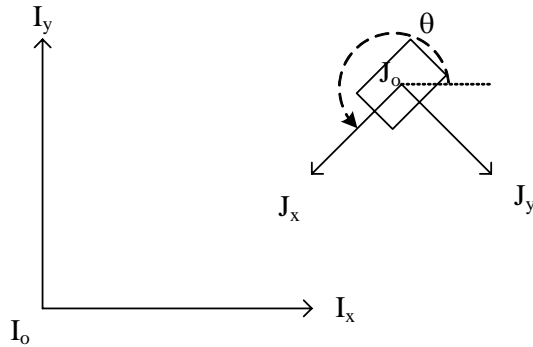
1. Read Chapter 10 in “Teach Yourself C”. Read Wikipedia and other websites explaining “HSV Color Space”. Read Chapter 9 in “Essentials of Mechatronics”.
2. In Lab 8 we are going to be working with a color camera. Each pixel of the camera picture is described by a red, green and blue intensity value. This is called the RGB color space. Each component is stored in an 8-bit char variable with 0 representing no contribution and 255 full contribution. For the robot to find other colors besides pure red, green, and blue it will need to look at a combination of these colors. An elegant way to do this is to convert the picture from the RGB color space to the HSV color space (Hue, Saturation, Value). In this color space, all the colors are arranged on a color wheel with a different angle indicating a different color. Do a web search for HSV color space to find more information about this color space and the equations to convert RGB to HSV. Write a function with the prototype `void RGBtoHSV(unsigned char r, unsigned char g, unsigned char b, unsigned char *h, unsigned char *s, unsigned char *v);`. This function should take the values passed in r, g and b, and convert them to the corresponding h, s and v values. h, s, and v are passed by reference so that their values can be changed by the function. Scale h so that its value from 0-360 is scaled to 0-255 (unsigned char). Scale the s and v values that range from 0 to 1 to a 0 to 255 value. Inside the function you will need to use floats to perform the conversion but return unsigned chars in the three variables passed by reference. *This code does not have to run on the MSP430F2012 as it probably would not fit.*
3. Obtain the rotation matrix, which converts base J's coordinates into base I's coordinates (R_J^I), for the following cases:
 - a.



b.



c.



4. Given the following translation vectors:

- a. $\overrightarrow{I_o J_o} = (7, 9, 0)$
- b. $\overrightarrow{I_o J_o} = (-8, 2, 0)$
- c. $\overrightarrow{I_o J_o} = (11, 21, 0)$

and using the results on the previous a, b, and c questions respectively, obtain the homogeneous transformation matrix, which converts frame J coordinates to frame I coordinates.

Using the homogeneous transformation matrix found in part c. above, solve the following two questions:

- i. If a golf ball's coordinates are (3, -2, 0) in the robot's (J) frame, what are the golf ball's coordinates in the world (I) frame as a function of θ ?
- ii. If a golf ball's coordinates are (3, 7, 0) in the world (I) frame, what are the golf ball's coordinates in the robot (J)

frame as a function of θ ? Note that the inverse of a homogeneous transformation matrix $H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}$ is

$$H^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0 & 1 \end{bmatrix}.$$

5. For this exercise you are going to take advantage of the hardware interrupt capability of the MSP430F2012's Digital I/O lines. You will wire two pushbuttons to pins P2.6 and P2.7 so that when pressed, individual hardware interrupts will be generated in the F2012. Read again chapter 8 of the MSP430 users guide:

http://coecsl.ece.uiuc.edu/ge423/datasheets/MSP430Ref_Guides/MSP430x2xx_usersguide.pdf.

http://coecsl.ece.uiuc.edu/ge423/datasheets/MSP430Ref_Guides/Cexamples/c/msp430x20x3_P1_04.c is also a good example to look at to get started on this exercise. Begin by unwiring the two switches connected to P2.6 and P2.7. Next solder two pushbuttons to your EZ430 breakout board. One end of the pushbuttons should be connected to GND and the other end should be connected to P2.6, P2.7. In your source, just as with the switches, setup P2.6 and P2.7 as inputs, enable the resistor and make it a pull-up resistor. This will cause the state of the button to be logic high when the button is not pressed and logic low when pressed. Starting with a new project created in project creator and using the example given above as a guide, modify the source code to enable P2.6 and P2.7 as hardware interrupts. When a hardware interrupt is generated on line P2.6, increment an integer (say int1) by one. When an interrupt is generated on P2.7 increment a second integer (say int2). In the code that sends "HELLO" to HyperTerminal, change it to now send the values of these two integers. *Note: P2.6 and P2.7 both cause the same interrupt "PORT2_VECTOR" to be called. Inside the interrupt function you will have to decide which interrupt triggered the function call by checking the respective bit in the P2IFG register. Also don't forget to clear the P2IFG bit before exiting the function.* Compile and run your program. What happens? You should probably see that when you press a button you receive multiple interrupts with each press. This is because the processor is seeing multiple jumps from high to low at the point of switch contact.

This is termed “bouncing”. We need to “debounce” this input line. There are a number of ways to accomplish this and we will talk about a number of these in lecture. Here we will solve the problem by adding a delay. When an interrupt from a pin occurs, change your ISR code to still increment the integer and clear the P2IFG flag but now also disable that interrupt source (P2IE register). Set a flag variable indicating that P2.6 or P2.7 has been disabled. Then inside the Timer_A (void) interrupt function wait for 100 ticks (tick = 1/9600 seconds) to go by and then re-enable the P2.6 or P2.7’s interrupt. Now run your code and see if the extra interrupts have disappeared. If not increase the 100 tick delay until you find good results. Hand in your code and demonstrate it working to your TA.

6. This homework assignment is up to you. Use your creativity to build something using the two RC servos that are in your microcontroller kit. Of course you will also need to use the F2012 microcontroller to control the RC servos. Make something that you will be proud of and put on your desk at home or something that will scare your friends when they try to raid your refrigerator. Anything goes but keep in mind that you also have a final project with the DSP/Robot to complete by the end of the semester. So in other words, I am not expecting it to be an elaborate and finely polished design.

Items that you **CAN** use that are in the Mechatronics Lab: *(This is not a complete list so ask if there is a part that you need).*

- a. Any of the parts (resistors, capacitors, sensors, etc) used in this and previous homework assignments.
- b. Anything (hardware, sensor, actuator or integrated circuit) that you purchase.
- c. Your second microcontroller (F2013) that came with the kit you purchased. (This is an easy way to get points for your design).
- d. The lab has a speaker that you can use/have.
- e. The lab has a microphone you can use/have. It is a cheap microphone so don’t expect too much from it. It can pick up loud noises.
- f. Raw plastic and aluminum, “Super Velcro”, nuts and bolts. Cheap items that can be purchased from McMaster-Carr. I will be the judge of what is cheap.
- g. You can turn your RC servos into continuous turn servos. Sadly this is an irreversible process for the RC servo. Ask your TA to be shown the procedure. With this modification, you could create a small mobile robot car.

Items that you **CANNOT** take from the lab.

- a. Pretty much any of the pre-made parts for the RC servos. Unfortunately these items are relatively expensive and I can’t give them to you.
- b. Sensors used by the Robot.
- c. Gears, pulleys, belts.
- d. Other items? Ask before you plan on using them.

This assignment spans both HW #4 and HW #5. Your finished product should be complete and checked off by the due date of HW #5 (April 21st).

What needs to be turned in for HW #4? (20 points).

- a. A description of what you are planning to build. What will it do? What parts are you going to need?
- b. Mechanical drawings of your device. Pencil and paper is fine as long as it is neat.
- c. A wiring schematic of all the electronic parts of your project. Pencil and paper is fine as long as it is neat.
- d. Any source code you have developed to this point. This source code must be commented well!