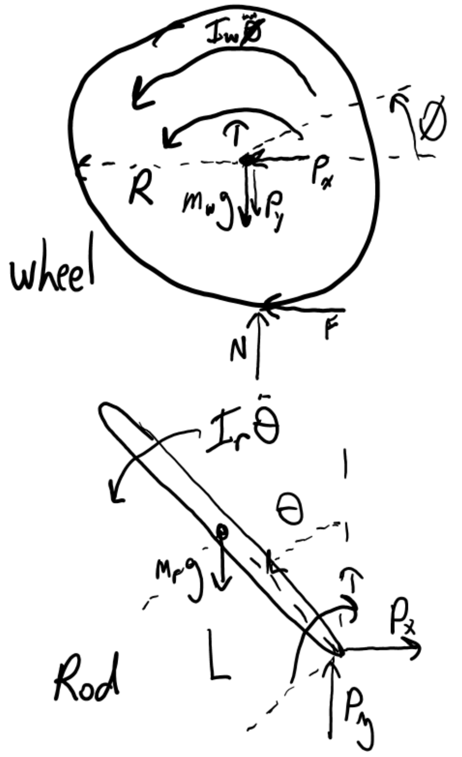
**SE 423 Mechatronics Homework Assignment #5, Spring 2020**
**Due by 5PM Friday April 17th. Submit a PDF copy of your homework by email and create a short video of your code working for problem 4.**

1. Read the computer vision chapter posted at the web site: http://coecsl.ece.uiuc.edu/ge423/datasheets/vision.pdf
2. (25 points) For this homework assignment you are going to explain much of the source code given to you in the file ColorVision.c. For the two parts of this question the source code has been divided by line number. Perform the tasks specified for each section of code.
   a. In `userProcessColorImageFunc_laser` lines 300 to 429, perform a "paper run" to understand what this code is doing. This paper run also includes the functions you will explain in part b below. Hand in your "paper run" crib sheet. The crib sheet is attached to this homework assignment. You can print out extra copies of the crib sheet at the web site. **Make sure to read the notes at the bottom of the crib sheet.**
   b. `int Fix_Equivalency(int num_equivalencies_used)`. Lines 638 to 708. Explain what this function does. The explanation should include the overall picture of what the connected parts algorithm in the function `userProcessColorImageFunc_laser` is accomplishing.

3. (15 points) This homework assignment is going to get you started thinking about your final project's self-balancing robot or Segbot. Actually I am hoping this problem helps me out. I have been building different versions of Segbot since the Segway was released around 2004. Over that time I have taken different approaches to derive the equations of motion for the Segway and identify the parameters like moment of inertia, mass, center of mass, torque constant, etc. Each time I have not come up with a model that I like. How I determine that I do not like the model is that if I use the model to design a control law in Matlab and use those control gains on the actual Segbot many times the designed controller works but somewhat poorly. Then if I hand tune the gains to get a well performing controller and try those gains on the simulated model it most of the time does not work in simulation and is an unstable controller. So what I would like you to do for this assignment is perform an online search for papers or websites that go into the derivation of the equations of motion for a planer balancing robot. I am sure there are many sources out there. All I am asking for is that you find one reference. If you have trouble ask me, but I am hoping that if I ask all of you to perform this search you may find a source I have not found myself. Below are the equations I use for the planer Segbot. Let's see if you find something different or with more detail.



$$(m_r R L\cos(\theta))\ddot{\phi} + (I_r + m_r L^2)\ddot{\theta} = m_r g L\sin\theta - \tau$$

$$(I_w + (m_r + m_w)R^2)\ddot{\phi} + (m_r R L\cos\theta)\ddot{\theta} = m_r R L\dot{\theta}^2\sin\theta + \tau$$

4. Using this C file http://coecsl.ece.illinois.edu/ge423/hw/user_test_SEQ2.c as a starter, setup the Data Transfer Controller (DTC) of the MSP430's ADC10 peripheral to sample channels A6, A3 and A0. Display all three of these readings in units of millivolts to Tera Term (Remember, since floating point calculations run slow on the MSP430 processor, do not to use floating numbers, only int or long). The given C file has much of the code you will need for this exercise but it was written to sample A3 and A0. Questions: Why then, in this sample code, does the array "ADC" have 4 elements and ADC10DTC1 equal 4? What will you need your array size to be and what will you need to set ADC10DTC1 to?

   Do not copy the entire given C code over the top of your project's starter code. Instead study this given code and only copy the needed code that helps you setup the ADC10 peripheral. Lecture will also go over how to setup the ADC10 using the DTC sequencer. Your code for this homework problem should sample, read and scale to millivolts the three channels, A6, A3 and A0, every one millisecond. In addition, every 250 milliseconds these three millivolt values should be printed to Tera Term. As a final step, for good practice, make one of your LEDs blink on and off every 250ms. Note: The given code samples A3 and A0 only every 500ms. Make sure your code samples every 1ms, even though you are printing only every 250ms.

   A0 should already be connected to a potentiometer on your board and A3 should already be connected to the photo-resistor on your board. For the pin A6 (P1.6) I would like you to wire it to the second potentiometer on your breakout board. This potentiometer was used to create the reference of 1.65 volts for the TLV5606 DAC chip. We will use it to vary the voltage coming into pin A6. An easy way to make this connection at home without soldering is to find a thin enough paper clip. The paper clip should be able to fit in one of the holes attached to A6 and then a hole attached to the middle pin of the second potentiometer. You will be able to just hold this wire (paper clip) in place when you are running the code. For you demo video, show me your breakout board and one of you LEDs blinking and then video your computer screen showing me the three voltages changing when you adjust the potentiometers and change the light intensity on the photo-resistor.