WORKS COMPLETED:

For this week, several objects were accomplished. The first of which was the camera and its associated components were finally ordered. The camera and components were ordered from a company called Mightex. The camera is a buffered 5 Megapixel 12-bit monochrome CMOS camera, enclosed, with GPIOs, C-mount, and CS-mount. This is a USB 2.0 powered camera, thus no power supply is needed. Components of the camera included a 2/3 inch 25 mm megapixel lense and an extension tube kit. The tube kit allows a lens to be focused on object that is nearer to the camera than the normal lens focus movement provides. The total cost of the camera was $924, not including tax, shipping and handling. This was below the projected cost for the camera and the team was extremely pleased.

Aside from ordering the camera, I’ve worked on updating the MS Project Timeline for the team. Together, the team listed tasks that had been done and tasks that need to be done and the time that is required to accomplish these tasks. I’ve inputted these into the MS Project Timeline and posted them on the website.

In addition, I’ve also worked on testing the motor with the carriage positioning module program and began to write the codes for the testing module. I connected the motor to the linear actuator and the stepper drive. I first configured the motor in MAX, and then used the carriage positioning module to control the motion of the motor. The carriage positioning module was able to control to movement of the motor, which in turn controlled the movement of the linear actuator, very smoothly and precisely. I recorded a video of the movement of the actuator using the program.

One problem I encountered was when I connected the motor to the actuator, the actuator was not completely aligned with the motor, and thus it was not balanced. One thing I did was to glue some rubber materials to the bottom of the actuator. This was able to balance the motor and the actuator, causing the movement of the actuator to be smoother. Figure 1 shows the rubber materials glued to the bottom of the actuator.

For the testing module program, I began by determining what are the inputs and outputs. I decided that the inputs will be the number of cycle the tissue will be loaded/unloaded, the speed of the motor, and the displacement of the motor. Basically, the testing module will be running using a displacement-controlled looping condition. This means that the displacement inputted by the user will control when how far the motor will move and when it will retracts. Because this is a stepper motor, the position and velocity of the motor are in units of steps and steps per second, respectively. Thus, I needed a method to convert this into real world displacement in either millimeter or micrometer. In order to do this, first, I determined that the maximum displacement of the linear actuator is 25 mm.
Then, I moved the motor from one end of the actuator to the other end using an arbitrary velocity. The program, MAX, allowed me to view how far the motor has moved in unit of steps. I was able to figure out that it took the motor approximately 46,000 steps to move from one end of the actuator to the other. Therefore, 46,000 steps corresponded to 25 mm of displacement. Through a small mathematical manipulation, I figured out that it took the motor 1840 steps to displace 1 mm on the actuator. From this information, the following equations were derived:

\[
\text{Speed: } \frac{X \text{ \mu m}}{s} \left(\frac{1840 \text{ steps}}{1000 \mu m}\right) = \frac{A \text{ steps}}{s}
\]

\[
\text{Displacement: } Y \text{ mm} \left(\frac{1840 \text{ steps}}{1 \text{ mm}}\right) = B \text{ steps}
\]

These equations will be used to control the motion of the motor.

In order for the testing module to be part of the whole flexural testing program, it has to have a case structure. The case structure will allow to program to run ONLY when the user presses the start button. Thus, the testing module coding will begin with a START Boolean button connected to a case structure. When the button is pressed, the Boolean will passes a true command to the case structure, and the case structure will run the TRUE case. When the button is not pressed, the case structure will run the false case which is doing nothing. Furthermore, in order to control the number of cycle, there has to be a For loop. The number of cycle will be connected to the terminal on the for loop and this will control the amount of times the For loop will run. In addition, within the For loop will be a sequence structure of three sequences. This sequence structure controls the sequence of events occurring within the For loop. There will be a total of three sequences. The first sequence controls the movement of the motor towards the tissue, bending the tissue. The second sequence controls the movement of the motor backward, unbending the tissue. And finally, the third sequence controls the writing of the data into a text file. When sequence one and two are running, the camera is also running and simultaneously collecting the data. Sequence three will combined all these data and write them into a text file. The followings outputs will be written into the text file: 1) time, 2) markers positions, 3) applied loads, and 4) positions of the posts. Figure 2 shows a working version of the testing module program.
FUTURE WORKS:

In the near future, I will continue to work on the testing module program. Several things needed to be added to the program such as a sequence for controlling the movement of the motor back to its original positioning, and a sequence to write data to a text file. Moreover, codes for the camera need to be written and incorporated into the testing module. However, any codes for the camera will have to wait until the camera arrives. This is because the camera needs to be tested to check for image quality. I need to know the image quality and how to process this image using Vision Assistant before writing the actual codes for the camera.

In addition, I will need to discuss with Eric on how to incorporate the calculations as a subroutine into the main flexural testing program. Eric is currently working on the necessary calculations. We need to decide whether these calculations will be as part of the main program (that is, a subroutine) or a separate program by itself (such as a post-processing program). If possible, however, we would like the calculations to be a part of the main program, so that the user will be able to view the results right away. If that is a case, then we will need to create another module containing those calculations within the main program.