John Greth

Team N: Automators Anonymous

Teammates: Zhiwei Zhang, Tianyu Zhao, Xuesu Xiao

ILR02

Feb. 13, 2014

A. Individual Progress

I worked exclusively on the motor lab this week. I wrote most of the code, figured out the gui, and did almost all of the wiring.

The code is essentially a big loop that checks the current state of the system, calculates the desired state of the system, and makes adjustments to lower the discrepancy between the two. First, the loop checks the state of the button to see if it has changed. If the button was just pressed, the control state changes. There are two control states: Sensor control and computer control.

Next, the code loop updates the sensor values. The value is calculated as a rolling average of the last 50 data points. For each update, the a counter is decremented by the oldest sensor input value and decremented by the newest sensor input value. Then the result is the counter value divided by the number of datapoints. This process keeps the sensor value from fluctuating too much and requires many data points to be stored but only a small amount of computation on each iteration.

In the sensor state, the target position of each motor is determined by one of the sensor values. In the computer state, the gui (see figure 1) controls the position for each motor. The desired servo value is simply written on a pwm pin. We used a 'step' function to control the stepper motor. A call to 'step' moves the stepper the desired distance and updates a stepper state variable so that the controller can keep track of the actual current position. 'Step' is bounded to 20 steps to bound the loop time so that the proportional control of the DC motor can update often enough.

B. Challenges

The gui was the biggest software problem I encountered. I couldn't find any way to generate a gui from Processing. QT seemed perfect until I realized that there was a storm of dependencies needed to get serial communication working on my mac. After spending several hours attempting to install/use QT and Processing, I found the Guino library which seamlessly integrates with the Arduino environment and produced a fairly nice looking gui with minimal effort.

The stepper motor we had was broken. I was able to get it to step by modifying the voltage, stepper controller current limit, and step frequency, but the direction was only correct ~75% of the time. Since I was able to get it to work in some capacity, I assumed that the code wasn't controlling it correctly. After several hours of banging my head against the wall, I tried another stepper which worked on the first try.

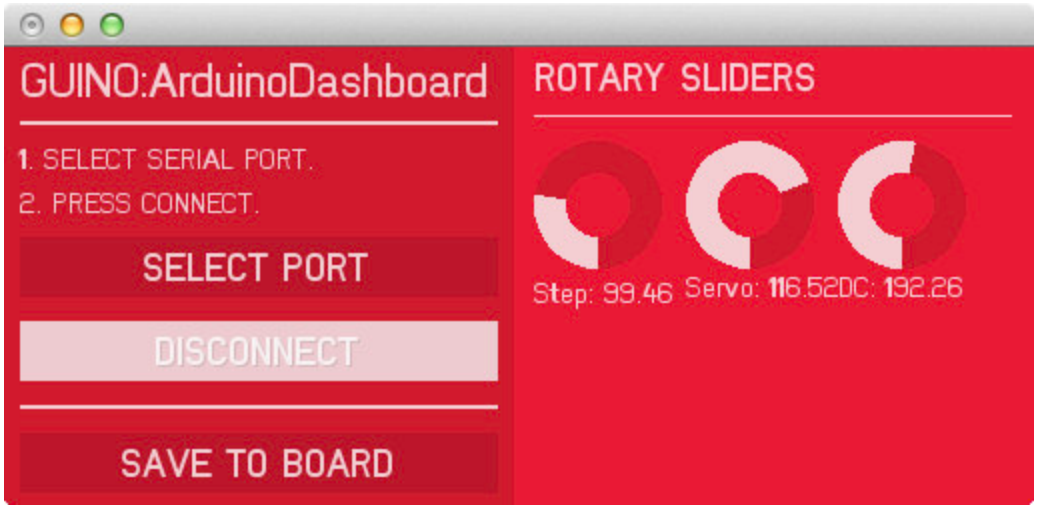
The last challenge was stopping the motors from jittering. The problem was that our control loop took too long so it would essentially alternate between moving the servo and the stepper motor

C. Fellow Team Members

We were all primarily focused on the motor lab. Jack wrote the PID code (we eventually

set the parameters to zero making it a proportional control). Xuesu used simulink to determine the optimal parameters for the PID control. Zhiwei did a little bit of everything, including creating he debouncing circuit, helping with the PID efforts, and writing/testing some of the code.

D. Figures



The three dials are used to control the absolute position of each motor individual motor.

Figure 1. The gui, powered by Guino.

E. Plans and Goals

My objectives for this week are to have a have a working sensor setup to do edge, defect, and ground detection. Based on our experience with the sensors so far, I think I will need to set up some type of shield to block ambient light and get more consistent values.