Individual Lab Report
Sensors and Motors Lab

Brian Boyle
Team B: Team Space Jockey
Teammates: Nathaniel Chapman, Ardya Dipta Nandaviri, Songjie Zhong

ILR01
October 10, 2013

1. Individual Progress

My primary role within this lab was to develop a Graphical User Interface (GUI) for monitoring and controlling the sensors and motors. I'd developed GUI's for robotic simulations and web applications in the past, but this was the first time using the particular GUI toolkit we chose, and my first time interfacing with an Arduino board using an external client program.

I was able to design and implement the earlier iterations of the GUI independently of my team while they interfaced individual motors and sensors in parallel. The design process involved itemizing the desired functionalities of the lab, deriving relevant GUI elements for each, then arranging them in a clean, organized, layout. I then used the Tkinter library for python to build the GUI with dummy values for sensor readouts and empty function calls attached to interactive widgets.

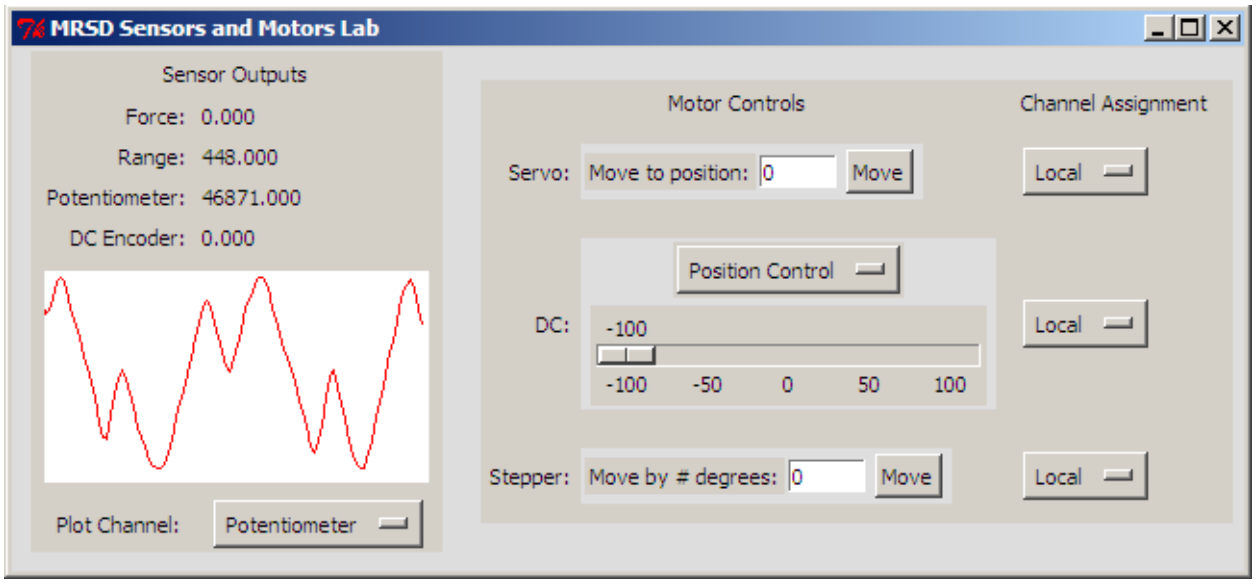

Figure 1: Graphical User Interface for Lab 1

The GUI is shown in Figure 1. The Sensor Output frame provides the most recent readout from each sensor. Below the input box is a plot canvas with a dropdown menu for selecting a sensor output to plot. The Motor frame includes a numeric input box for the stepper and servo motors and a button to send the corresponding command to the Arduino board. For the DC and Stepper motor, this command directs the motor to move by a given number of degrees. For the Servo motor this command send a position to which the motor must move. The DC motor controls also include a slider which allows the user to select any speed between full-reverse and full-forward velocity, and a corresponding command is sent to the Arduino each time the slider's position changes. Above this is a dropdown menu

which allows the user to select between position and velocity control. The slider below then commands the motor to move at speeds between full-forward and full-reverse. For position control, this slider moves the motor between its two extreme positions. Additionally, every motor has a “Channel Assignment” drop down menu, which directs the Arduino to move the motor based on GUI output (Local) or any of the four sensors.

After the GUI was mostly completed, I was able to work with Nathaniel to implement serial communication between the Arduino and the client program run from a laptop. Nathaniel had more experience than me in this domain so most of my work here involved learning the packet protocols and general functionality, and implementing as much as I understood on the client program.

The plotting functionality was added as an afterthought; it currently only plots the relative motion of sensor data. I developed this functionality from scratch after I encountered significant difficulty integrating existing plotting packages with Tkinter. Regardless of its shortcomings (no unis or axes labels), it was still very useful as a basic feedback source for integrating the sensors. This could be developed further in the future but I’d rather find an existing package that works either with Tkinter or another toolkit.

2. Challenges/Issues

The first challenge in developing the GUI was familiarizing myself with a GUI toolkit that was compatible with python. I found example implementations of the Tkinter package at Nathaniel’s suggestion, and found it to be very simple to integrate into a python program and to have all of the necessary interactive widgets for the scope of the lab. I spent some time experimenting with these widgets and the methods for arranging them in a workspace. I found the most flexible option to be a hybrid of frame- and grid-based layouts. Frames appeared most useful for broadly partitioning the GUI window into sections, while grids were most useful for arranging elements so that they cleanly aligned with each other in an organized and visually pleasing way.

The next challenge was to design a GUI that presented both sensor/encoder data and the desired interactive widgets in a way that was intuitive and compact. The basic design was first sketched, then coded, and then iterated over the course of the lab by presenting the GUI to teammates and modifying it based on their feedback. I began by partitioning the GUI window into two frames, left and right, which would present sensor readouts and control widgets, respectively. The right half was then further partitioned into frames for each of the three motors. Within each frame I used a grid layout to place labels, sensor readouts, and motor controls in individual cells.

The greatest challenge of this lab was integration. Each of the teammates interfacing with sensors did so on independent boards that had to later be integrated together. At the same time, drivers for each sensor had to be integrated into a single program for the Arduino. Finally, that program had to integrate with the GUI. All three of these processes took place at once, which meant that an error in one process often held up the other two or had a cascading effect throughout. In the future, I'd like to conduct integration more incrementally as individual sensor and motor drivers are prepared. This way, it might also be more feasible for all the team members to be familiar with the system as a whole as their individual work is integrated and the system complexity grows iteratively, instead of all at once.

3. Team Work

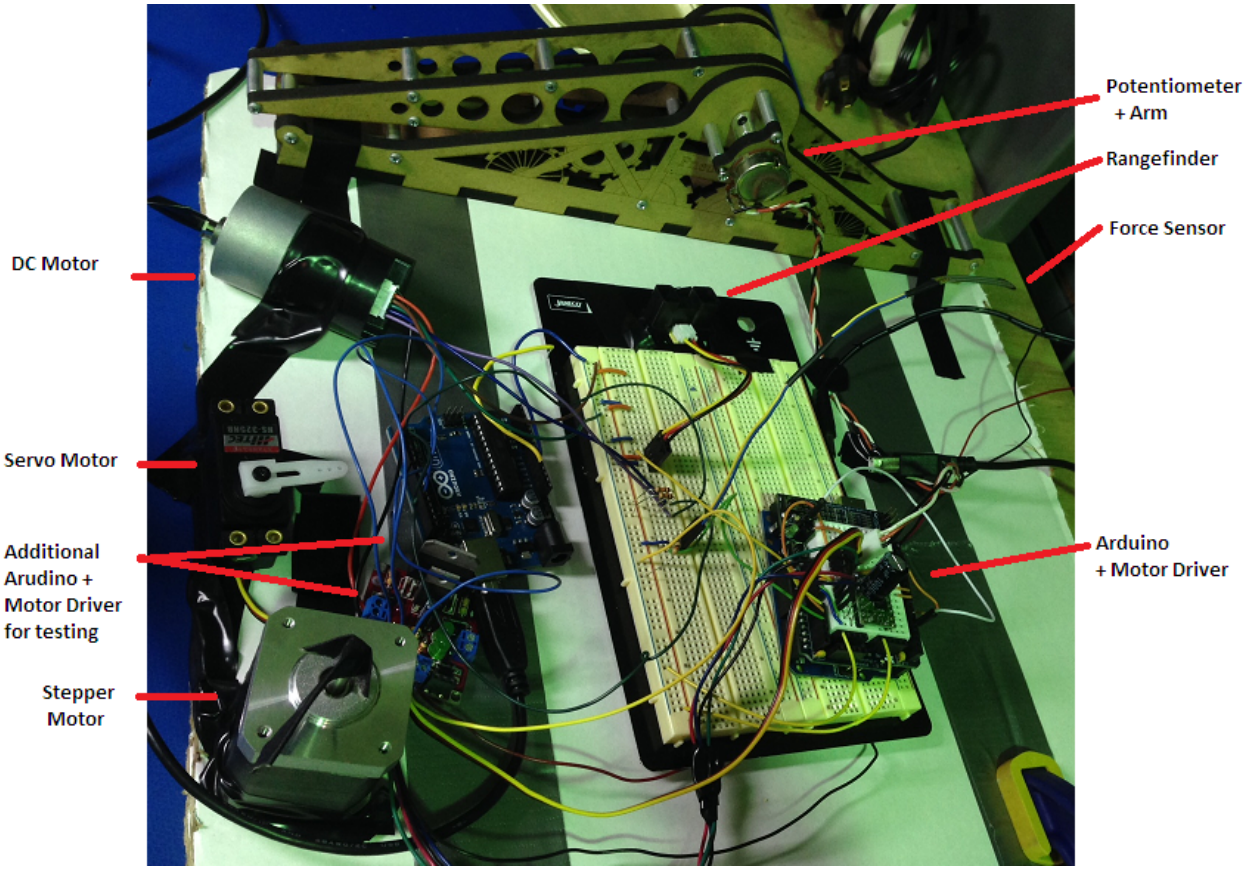

Figure 2: Integrated Bench-Squid (Photo by Dipta)

Sensors and motor interfaces were developed independently at first by my teammates with

their own arduino boards then integrated into one bench-squid (See Figure 2). Songjie developed the initial drivers for the Force Sensor and collaborated with Dipta for the DC motor. In addition to DC motor work. Dipta also interfaced with the Infrared Rangefinder and Servo motor. Nate was responsible for the Potentiometer and Stepper motor and took lead on system integration.

As each sensor and motor was successfully interfaced with, Nathaniel collected and abstracted the associated code into a single set of drivers. All code was stored in a repository where team members could review it. This was useful, as I was able to understand exactly how to interface with each sensor and motor even though my official task was GUI development.

Integrating the GUI with the Arduino code and making them communicate involved collaboration between Nathaniel and I to connect his codebase of drivers to my GUI client. After we had established serial communication and a packet protocol we worked to coordinate communication between the drivers and GUI elements.

4. Future Work

While I learned a lot in this lab about sensors, motors, tkinter, and the nuances of serial communication, I feel far too much time was spent setting up very basic functionalities that are already encapsulated and easier to access in toolkits like Processing and ROS. Using such a toolkit I expect we'd have been able to significantly reduce the total man-hours that went into reinventing the wheel. In the future, I plan to investigate these toolkits and try to persuade my teammates to settle on one for our project. A trade study would certainly be helpful for this.

For next week's Progress Update, I plan to create a mockup of a GUI for controlling the Space Jockey robot. This GUI will include at least two distinct views, one for monitoring the kinematic state of the robot and issuing joint commands, and another for mission control that will include mapping and waypoint functionalities. Additionally, my team plans to have ordered some hobby motors and other electrical components to form the beginnings of a project bench squid, design a leg for fabrication to begin prototyping, and conduct trade studies on motors and batteries.