Yiqing Cai

Team G: The Excalibr

Teammates:
Huan-Yang Chang
Man-Ning Chen
Siddharth Raina
Sambuddha Sarka

ILR01
Oct. 14, 2016

Individual Progress

Overview

For the Sensors and Motors lab, I was primarily responsible for developing the Graphical User Interface (GUI) for the system. And because the GUI plays an important role in connecting the whole system and transferring data between the computer and the Arduino board, I also worked on the integration work of the sensors and motors. We decided a vague structure of the GUI and functionality of the whole system at the beginning, and made some slight changes and improvements during the entire work process.

The layout of the whole circuit is as follows (see Figure 1) :

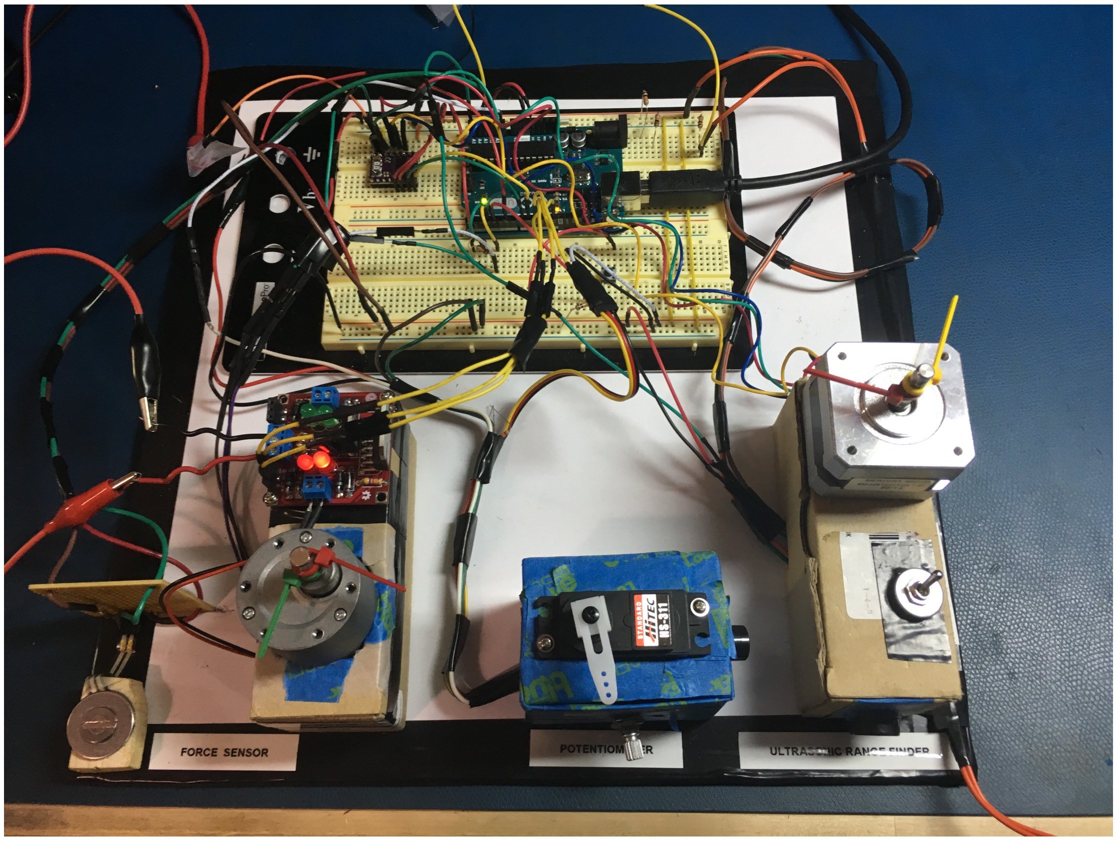


Figure 1. The final circuit for sensor and motor

In the end, our GUI is able to carried out specific functional requirements as follows.

1. Provide both text areas and sliders/knobs for setting control values for the motor.
2. Provide switches and buttons for state switches, working mode switches and independent control.
3. Display all sensor readings, motor speed, positions, and current status in real time.
4. Use colors and sliders/knobs to indicate the sensor input and motor information within a certain range.

Implementation

I chose to develop the GUI on processing, because it is easy to communicate with Arduino through serial port to get data, and there are libraries allowing direct control of Arduino without writing Arduino code. I chose to use the controlP5 library of processing, by which I can design the GUI layout and its functionalities entirely from java code. Previously I have experiences in designing GUI on Visual Studio MFC using C++ code and on Matlab GUI toolbox using Matlab code. Although java and Processing is entirely new to me, I think learning to use a new tool with a new platform is fun and I should be able to implement it with my previous knowledges in developing GUI.

The first step to development the GUI is to design the layout and all the functionality we need to perform and display. Sambuddha and I discussed to decide the string format used for data transmission, and all the flags for indicating states and all the variables we need to be controlled or displayed.

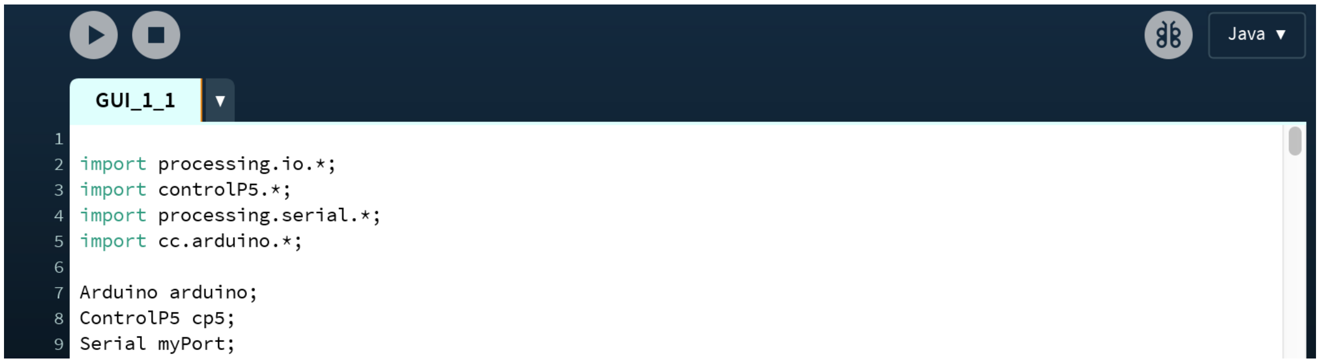
String format: (Sensor - a capital letter)(ReadingValue - an integer)(“ “ - indicate the end)
(Motor - a capital letter)(ReadingValue - an integer)(“ “ - indicate the end)

A capital letter stands for a certain type of sensor or motor with its unique parameter, and a number following it indicates the value of the reading, a space will indicate the end of the string. The data for read and the data for write through the serial port should be in the same format.

Besides, all the input from the GUI for motor control instructions would be an integer in a certain ranges, and all the values to displayed on the motor information and sensor input on the

GUI would be a float or double. I need to do a series of calculation in the processing before displaying on the GUI and sending back to the Arduino board.

To enable the serial port and Arduino control from the Processing, I installed several libraries and studied the format of calling corresponding functions (see Figure 2).



```

import processing.io.*;
import controlP5.*;
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
ControlP5 cp5;
Serial myPort;
```

Figure 2. Import the necessary library in processing

And then I need to define and set up all the layouts and define the functionality of each element on the GUI by defining functions which will be implemented when calling an event.

For indicating states, I need to design the functions for the corresponding event calling. For distinguishing from different control instructions, I set flag to be different values; and for states and modes changes, I used the combinations of toggles and buttons to switch on/off. Besides, the buttons can be used to change directions of DC motor and stepper motor.

In order to give intuitive display of the sensor input and motor information, I used colors and sliders/knobs to indicate that. The values can be set within a certain range corresponds to a specific sensor, and the values of color and slider/knob with be changed in realtime.

Also for the GUI motor control part, we can either give instructions by entering a value in the textfield or by changing the value of sliders/knobs. The toggle button is to control the direction of the motor, and the motor set button is an indication of giving slider/knob instructions.

The final GUI is displayed as follows (see Figure 3) :

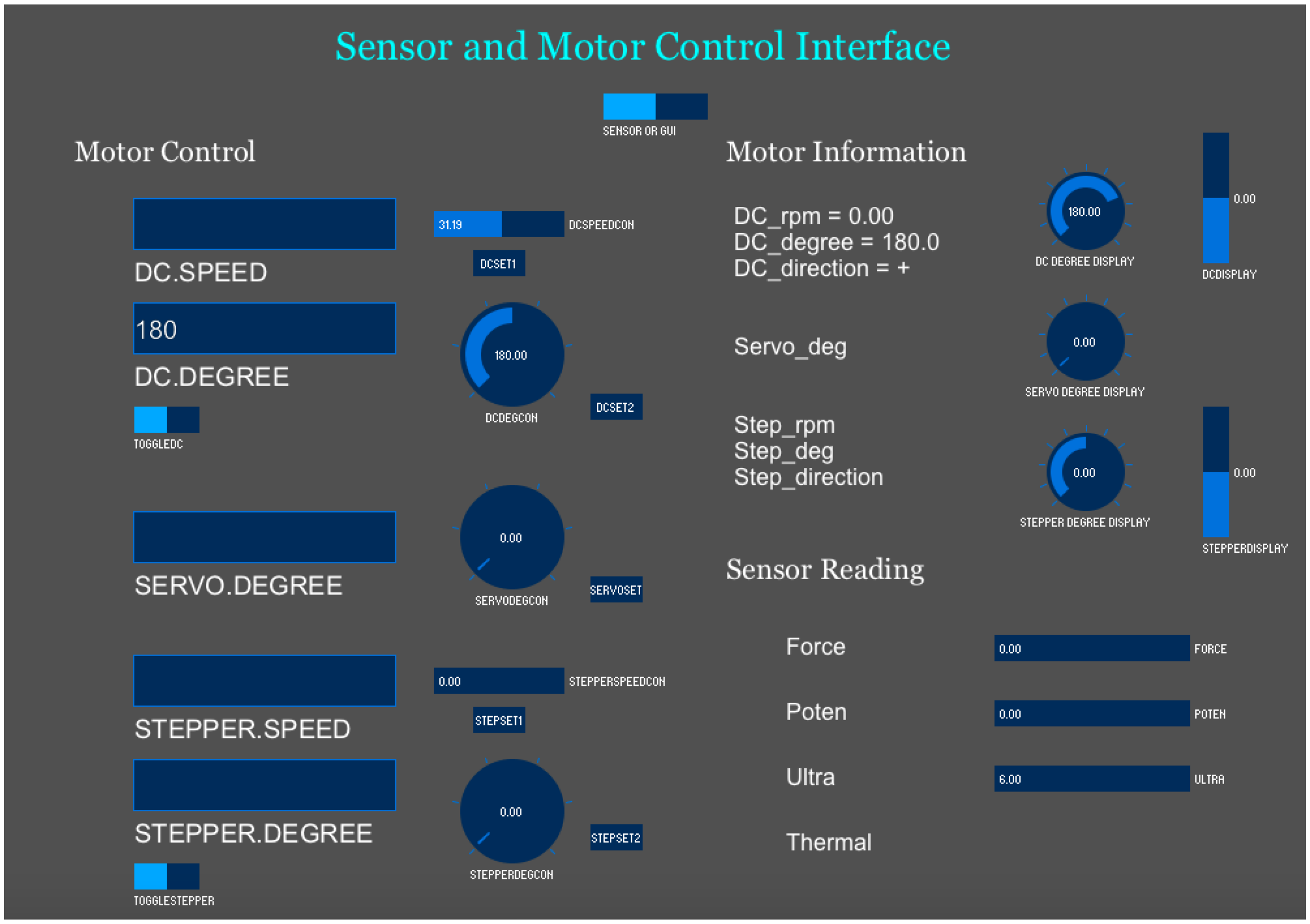


Figure 3. The final GUI for sensor and motor

The left section is for motor control, which acts as inputs and the right section is for sensor and motor reading, which acts as outputs. The sliders and knobs on the left are used for input control and the sliders and knobs on the right are just for parameter displaying. Toggles are used for direction control and mode switches.

Challenges

The main challenges I faced during this task were:

1. Integrating code and all the functionalities. As each group member defined their input and output and designed their own functionality, it is hard to combine all of them onto one board due to limit of ports and space. When we tried to combining the circuit and the code, we spent a lot of time deciding the final functionality of our system and what to display on the GUI. And because all the data were transferred through the serial port, we need to set many flags to indicate the correspondence between inputs and outputs, and also state machines to indicate which functionality is to carried out, in order to make the GUI more interactive.

2. Hardware I/O library of Processing can only work on the Raspberry Pi and other Linux-based computers, but not on our Arduino board. So we cannot write all the code on Processing and directly control the Arduino board as we wished because then we wouldn't be able to use Interrupt for PID control for the DC motor. Instead, we finally changed plans and wrote all the functional code on Arduino and all the layout of GUI on processing, and relied on the serial port for communication between Arduino and Processing.

Teamwork

Our group use four different sensors to control three different motors, and we develop a GUI based on Processing. Work undertaken by each team member is as follows (see Table 1):

Member	Tasks
Huan-Yang Chang	Ultrasonic Sensor, DC Motor Control and System Integration
Man-Ning Chen	Potentiometer, Thermometer and Circuit Integration
Yiqing Cai	GUI Development and System Integration
Sambuddha Sardar	Stepper Motor, Servo Motor Control and System Integration
Siddharth Raina	Force Sensor and Circuit Integration

Table 1. Team co-work

The team worked with great coordination during execution of the entire task. Each one of us completed their own part of designing and testing and then integrated all the parts. We communicated during the entire task and solved problems together when different parts weren't not compatible and when the Arduino functionality didn't perform as we wish on the processing. Finally, Sid and Mandy worked on the circuit integration and Me and Peter and Sam worked on the code integration. We faced many difficulties but we worked them out eventually as a group.

Future Plans

From now on, our team will work on two main areas of our project:

Calibration Process:
As we have finished setting up the robotic arm and the multi-camera capturing system, we will begin to collect image data and positional information of the robotic arm with the calibration target fixed on it. We need to firstly implement the Fixed Pattern Noise (FPN) calibration, which should precede all the other calibration procedures. Then we will design the method to compensate for vignette patterns which is caused by photometric staff and lens artifacts. After that, we can calibrate camera responses (photometric) and correct lens distortion (geometric), and estimate the extrinsic parameters between all cameras.

Aerotech Control and Motion planning:
We need to program to control the robotic arm to move in a certain pattern, the PSO output, and to trigger multi cameras to capture images when the robotic arm give a pause.