Che-Yen Lu

Team E: PLAID

Teammates: Michael Beck, Akshay Bhagat, Matt Lauer, Jin Zhu

October 14, 2016

1. Individual progress

For sensor and motor lab, I was responsible for developing Graphical User Interface (GUI) interface to control the motor and show sensor readings on GUI. My second role is to integrate interface between Arduino and GUI application. The requirements for GUI are listed below

1. Plot all sensor readings in different GUI windows
2. Update all sensor readings in real time
3. Provide input field to control motor velocity or position.
4. Connect to Arduino via comport
5. Make sensor and motor pair to control motor by specific sensor

There are three suggestions for developing the GUI interface, which are processing, QT, and ROS. Because QT could be run on different OS platforms and we need to be familiar with C++ to develop ROS application, I chose QT to implement the GUI.

QT is easy to design UI with the help of QT designer. First, I used the layout functionality in designer to sketch the GUI layout. There are four types of layout, which are vertical, horizontal, grid and form. Grid layout is my top choice since it is flexible. Second, I create the connection between signals and slots. Signal and slot are like event and callback function in other framework. When user click a button or choose different radio box, the callback function will be called. The callback function allows us to customize behavior based on the UI component user interact with. One combobox and one push button are created for comport selection. Four push buttons and 4 text fields are added to send control command to Arduino. Eight radio buttons and one push button are implemented for making sensor and motor pair. Also, one push button is used for showing sensor reading graph.

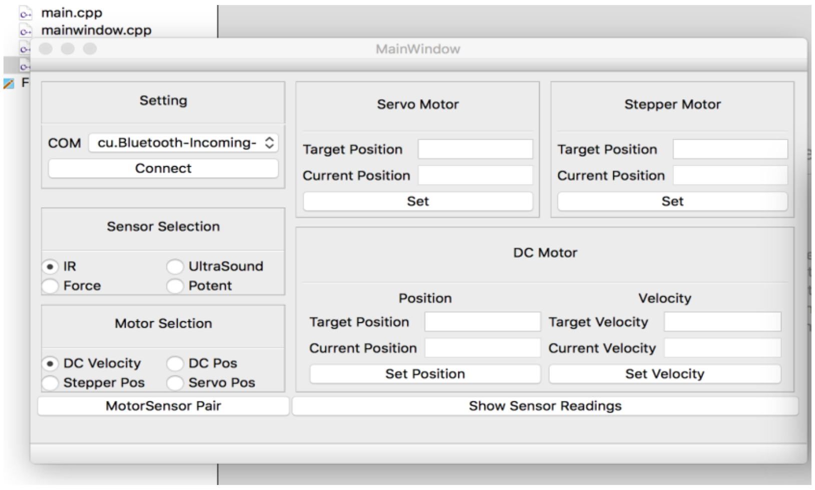

Figure 1. GUI interface layout

As for communication between GUI and Arduino, I define a two characters' protocol to ease the effort of implementation. The first character stands for the input to control the motor, and the input could be either GUI interface or sensor reading. The second character is the motor abbreviation for the motor selection. If the user wants to control motor via GUI interface, the first character will always be 'G', and the second character will still be the abbreviation of the motors. However, a number from 0 to 100 is appended after two characters if the first character is 'G'. For example, 'GV100' means user wants DC motor to rotate at maximum speed. 'US' means user wants stepper motor to rotate based on the reading of ultra sound sensor.

Motor, Sensor or GUI	Abbreviation
GUI interface	G
DC Motor Position	D
DC Motor Velocity	V
Stepper Motor	S
Servo Motor	O
IR Sensor	I
Ultra Sound Sensor	U
Force Sensor	F
Potentimeter	P

Table 1. Two Character's Protocol Table

For visualizing sensor reading, I use the QT plotting widget, Qcustomplot, as our GUI plugin. Qcustomplot has no other dependencies and is well documented. It uses OpenGL as 2D/3D plotting engine, and it can use graphic card to accelerate the computing.

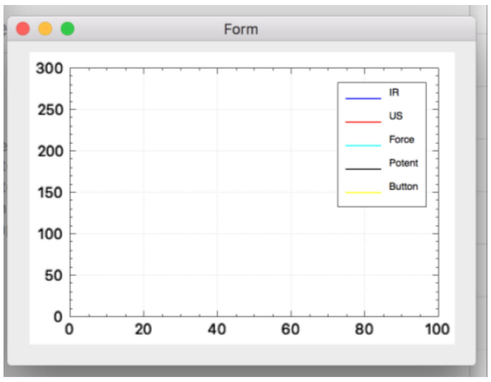

Figure 2. Sensor Reading Window

2. Challenges

The primary challenge was the integration between Arduino and QT. Although serial port is a standard communication protocol, I still spend tons of energy to make sure the commands or sensor readings are received correctly. Sometimes serial.available in Arduino will return false even if there are pending string need to be received. We try to add delay in the reading loop and it works perfectly. Also, the baud rate should be as low as possible since the communication is unreliable at higher speed rate.

Moreover, we got into troubles when we try to integrate whole sensor and motor's code. The coding style and interface for every function is different and incoherent, so we integrate the software till last moment.

3. Teamwork

This project need us to split task into five sub-tasks, and we break down the task as follows:

- Michael Beck – Develop driver for force sensor and help Matt with DC motor.
- Akshay Bhagat – Develop driver for IR sensor and Stepper motor. Integrate all sensor and motor Arduino software and hardware.
- Matt Lauer – Develop driver for Potentimeter and DC motor.
- Che-Yen Lu – Design GUI interface and integrate communication between Arduino and GUI interface.
- Jin Zu – Develop driver for Ultra sound sensor and Servo motor.

We learned a lot of lessons from this project. We should integrate all motor and sensor earlier, and we should reserve at least two days for testing and performance tuning. Also, the coding style and interface for each function should be defined in the beginning.

4. Future plans

From now on, I will focus on designing software architecture. New software architecture should be modular and easy to test. I will read the code from last year and stand on their shoulder to avoid pitfalls. Also, I will implement code skeleton in order to help teammates to start with coding. Good principle of coding will definitely help us to identify issue root causes quickly.

Since I am the software architecture in team, I need to understand each domain to sketch the draft. I will help Akshay and Jin to integrate the perception pipeline and camera, and work with Matt to deploy planner on UR5.

Note: The code is in private repo, but we can grant permission if it's necessary
https://github.com/wixter/SensorMotor