Sensor and Motor Control Lab

Individual lab report #1 || Oct, 14, 2016

Xiaoyang Liu

Team F

Rescue Rangers

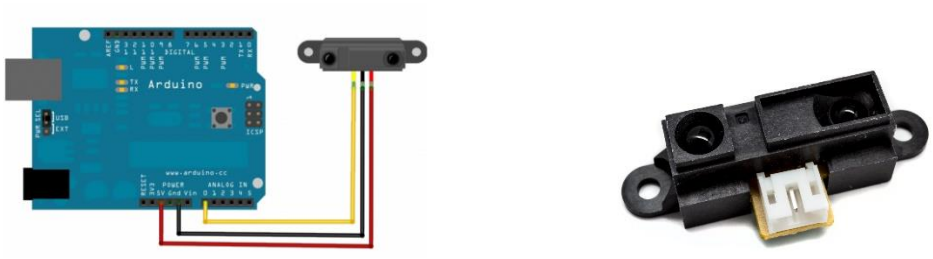
Juncheng Zhang (Henry)
Karthik Ramachandran
Sumit Saxena

Individual Progress

In the Sensors and Motor lab, I was responsible for the integration of the stepper motor and IR sensor as well as the hardware integration of all the components for the final assembly.

1) Integrating IR sensor and Stepper Motor

- Interfacing IR sensor with Arduino



(a) (b)

Figure 1. (a) Connection Diagram of IR Sensor (b) Infrared Proximity Sensor - Sharp GP2Y0A21YK

The Input data of IR sensor are analog signals, and thus the signals can fluctuate in a certain range even if there are no obvious distance changes between objects and the sensor, so I did some accumulated calculation to make the measurement more stable and effective. My final calculations ended up using 20 measurements and rejecting consecutive measurements that were not within +/- 5% of the previous measurement. I also ignored measurements that were outside the band of 20 cm to 80 cm. Sensor readings that were too close the device or too far away from the device were super noisy and thus had to be clipped.

- Stepper motor control

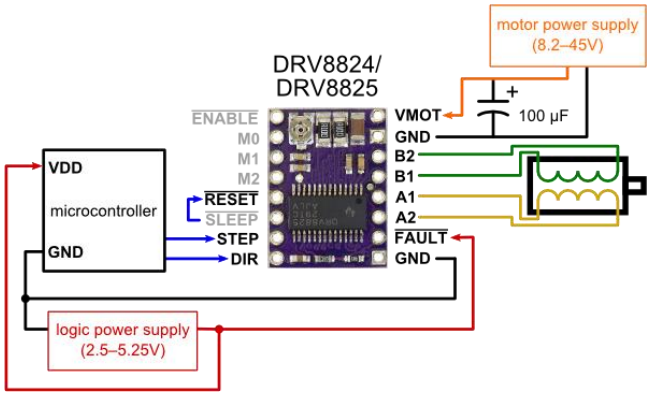


Figure 2. The circuit diagram of DRV8825

Setting current limit: we need to ensure that the driver chip supplies optimum current

to the motor and doesn't burn out itself due to high current. After reading the datasheet of the driver ICs, I learned that a possible way to set current limit on driver is tuning the potentiometer on the board. In order to set a desirable current limit, I put the driver into full-step mode and measured the current running through a single motor coil without clocking the STEP input. Since the measured current will be 0.7 times the current limit, I tuned the potentiometer letting the current limit to be lower than 1A.

Counting step-size: the step (and microstep) size of a stepper motors is crucial for it should be able to move the motor a user-selectable number of degrees in either direction. According to the datasheet, I set the resolution (step size) selector inputs M0, M1, M2 to be LOW, HIGH, LOW, so that the stepper motor has a quarter microstep revolution. Then once the user-selectable number of degrees is inputted, the conversion formula between steps and angles should be

```
int step = int(angle_2*4/1.8)
```

While integrating the sensor and motor, I used the processed IR data as the input of stepper motor, and then mapped the inputs to a range of the speed of stepper motor (20-200). Also because of the useless sensor data which is out of range, I made the stepper motor stop (0) when the measured distance is below 20cm or over 80cm.

In writing the code, the libraries in Arduino can be used to control both the stepper motor and IR sensor. I tried to make the code modular so as to easily integrate with the rest of the code base. Constants and pins were listed on top of the whole script, and I define all the input and output variants in a struct which can be used as global variants when calling each function. This allowed Karthik to easily plug it into the code that communicated with the GUI.

2) Hardware Integration

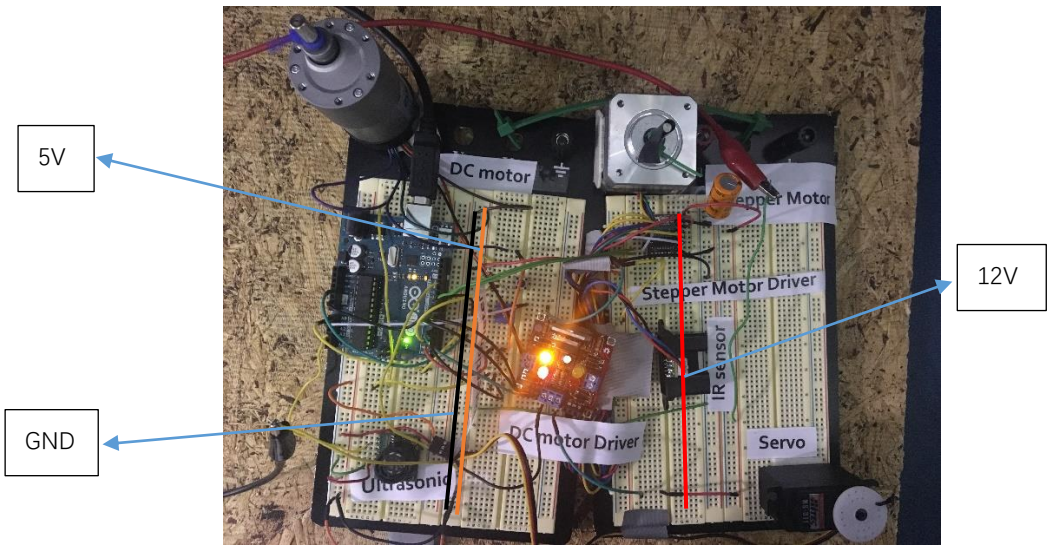


Figure 3. Image of final integrated system

The second mission for me is to integrate all the hardware. There are 3 subsystems each contains one motor and one sensor in our system: DC motor and potentiometer, stepper motor and IR sensor, servo and ultrasonic sensor. Plus, there are 2 power system (5V for powering servo and sensors, and 12V for powering stepper motor and DC motor), and we should make sure that they have the common ground(GND). I put 5V and 12V power supply in different bread board in order to isolate strong and weak current.

After ensuring that all the sensors and motors were functioning as they should as separate units, I built the hardware layout of the system, ensuring minimal wiring and reasonable spacing between components.

Challenges

1. At first it's hard to control the stepper driver, because by using the stepper motor driver, there are only 2 control pins (direction and step). However, the library in Arduino for stepper motor is used for 4 channel-control stepper drivers. I spent a lot of time debugging this issue and finally realize that pin9&10 are used for direction control, while pin8&9 are for step controlling. Also, the stepper motor is very strict with current, so I need to set the current limit before putting it into the subsystem and add capacitor between power and ground making sure that
2. Another issue was the noise due in the IR sensor. Due to a variety of features of objects, the output voltage will change despite distance does not. Therefore, I need to make the output of IR sensor more stable and effective. The method that finally worked in eliminating false readings was averaging over 20 readings and discarding values which were way out of the expected range.

Team Work

The project was split among the four teammates. Juncheng(Henry) handled the DC motor with the potentiometer. Karthik worked on the servo and the ultrasound sensor as well as the integration of all the Arduino code. Sumit worked on the GUI. In terms of working as a group, we each had to make sure our individually developed code would work with the larger system. The included making sure the ports and pins used on our individual boards wouldn't conflict in the larger system and code base. Karthik plugged all our code into one cpp file and made sure that we were all able to output sensor and motor variable for easy interfacing from the GUI.

Plans

Over the course of next a few weeks, our team will focus on

- **Localized Pattern Navigation**
  - Design strategy for localized navigation pattern.
  - Software for planning localized navigation pattern with acceptable sensor coverage (as per FVR).
  - Software for planning localized navigation pattern with high quality sensor coverage (as per SVR).
  - Software for planning localized navigation pattern for rescue operation to drop packet accurately (as per SVR).
- **Rescue assembly system**
  - Evaluate and finalize the design and actuators for rescue assembly mechanism.
  - Fabrication of mechanical structure and test functionality independently.
- **Global waypoint generation**
  - Software to generate optimal navigation path based on simplified coordinate input in a constrained environment (as per FVR).
  - Software to translate region information from a Global Map to specific GPS coordinates for a large environment (as per SVR).
- **Signature detection and analysis**
  - Finalize set of signatures to be used for detection.
  - Collect data and evaluate techniques for signature detection.
  - Build a rudimentary signature detection module (as per FVR).
  - Software for integration of the module with data obtained from payload sensors.
  - Collect large amount of data to train more accurate models.
  - Build advanced signature detection module (as per SVR)
  - Performance optimizations/scaling to operate within acceptable time (as per SVR).

REFERENCES:

Figure 1. (a): http://communityofrobots.com/tutorial/kawal/how-use-sharp-ir-sensor-arduino
Figure 1. (b): https://www.sparkfun.com/products/242
Figure 2: https://www.pololu.com/product/2133