10/16/2015

Sensor and Motor Control Lab

Individual Lab Report #1

Abhishek Bhatia

Team D: Team HARP (Human Assistive Robotic Picker)
Teammates: Alex Brinkman, Feroze Naina, Lekha Mohan, Rick Shanor

I. Individual Progress

For the Sensors and Motor Control Lab, I was responsible for controlling the stepper motor using the ultra-sonic range finder sensor. I used a MERCURY SM-42BYG011-25 bipolar stepper motor and a Pololu A4998 motor driver. Bipolar stepper motors have a single winding per phase. The current in the winding needs to be reversed in order to reverse a magnetic pole, hence an off-the-shelf motor driver is required. The stepper motor I received already had the wires soldered on the pins. I used wires and connectors to complete the circuit shown below.

Figure 1: Pololu A 4988 Stepper Motor driver (left) and MERCURY SM-42BYG011-25 Stepper Motor (right)

(Image Courtesy: Erin Hicks, ILR#1 (MRSD Team A, 2014) and Sparkfun.com)

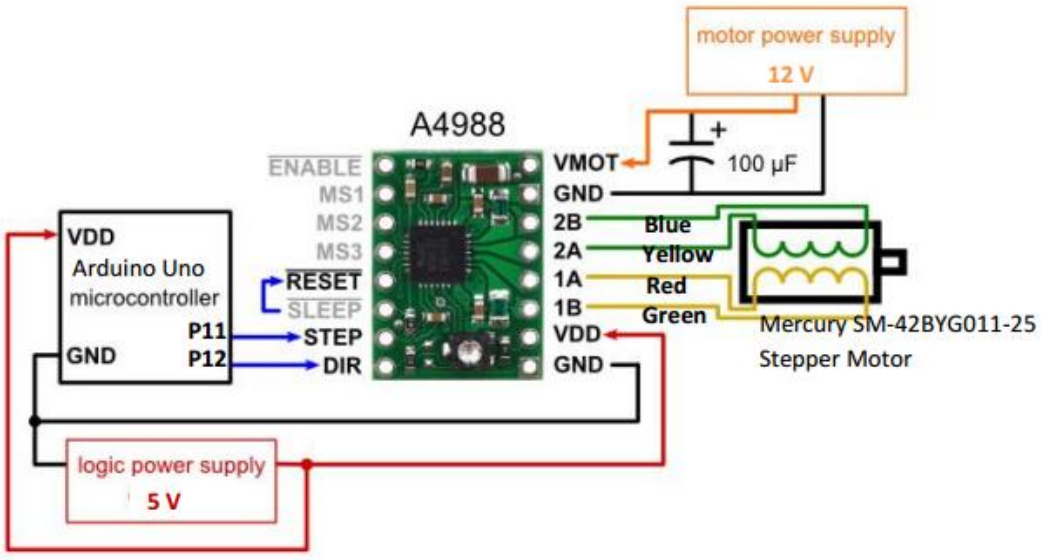

Figure 2: Circuit Diagram connecting Stepper motor and motor driver to Arduino Uno

(Image Courtesy: Erin Hicks, ILR#1 (MRSD Team A, 2014) and Sparkfun.com)

After setting up the circuit, I wrote the code to control the stepper output, number of steps and direction by setting the appropriate output at the step and direction pin. High output at the direction pin meant clockwise rotation of the stepper motor and low output meant anti-clockwise. Similarly, a toggle from high to low output at the step pin indicated stepper motor to rotate by 1 step in the direction indicated. 1 stepper motor step is equivalent to 1.8 degrees (datasheet). So a rotation of 360 degrees required 200 steps in the specified direction.

After testing stepper operation for various configurations. I set up the ultra-sonic range finder sensor circuit on the breadboard and then proceeded to write the code to acquire data from the sensor. The sensor works by sending out a burst of ultrasound and listening for the echo when it bounces off of an object. I used the trig pin to send out the pulse and echo pin to detect the same reflected pulse. I used the pulsein() arduino function to calculate the duration for ultrasound to travel back to the sensor. I used the simple calculation (Equation 1) mentioned below to convert the duration to distance in centimetres. The speed of sound is 29 microseconds per centimetre.

Equation 1:

$$\text{Distance (in cm)} = \frac{\text{Duration of the pulse (in microseconds)}}{29 * 2}$$

The pseudo code for both the stepper motor and sensor can be seen in Appendix A and B, respectively.

Figure 3: HC-SR04 Ultrasonic Range Finder Sensor
(Image Courtesy: Sparkfun.com)

Besides working on the stepper and ultrasonic range finder sensor, I worked on integrating the complete Arduino code for different motor-sensor combinations. Rick and Alex helped me set up the complete circuit including 3 motors and 4 sensors. I used a push button to switch states between different sensor/motor configurations. I designed the state machine such that in the initial configuration, state 0 controlled position of a DC motor

using IR range finder sensor, state 1 controlled speed of DC motor using pressure sensor, state 2 controlled RC servo motor using potentiometer and finally state 3 controlled stepper using ultrasonic range finder. This was the initial configuration of sensor and motors.

Finally I worked with Alex and Rick to verify and fine-tune the complete operation. We ended up changing the sensor motor combinations as to design the best possible stable system. In our final setup, we controlled position of a DC motor using an IR range finder sensor, speed of DC motor using pressure sensor, RC servo motor using potentiometer and finally stepper using ultrasonic range finder. We mapped the sensor outputs to appropriate values in centimetres (IR and ultrasonic range finder) and percentage (potentiometer and pressure sensor), similarly motor outputs to appropriate speed (DC speed) and degree/direction (DC position, servo and stepper).

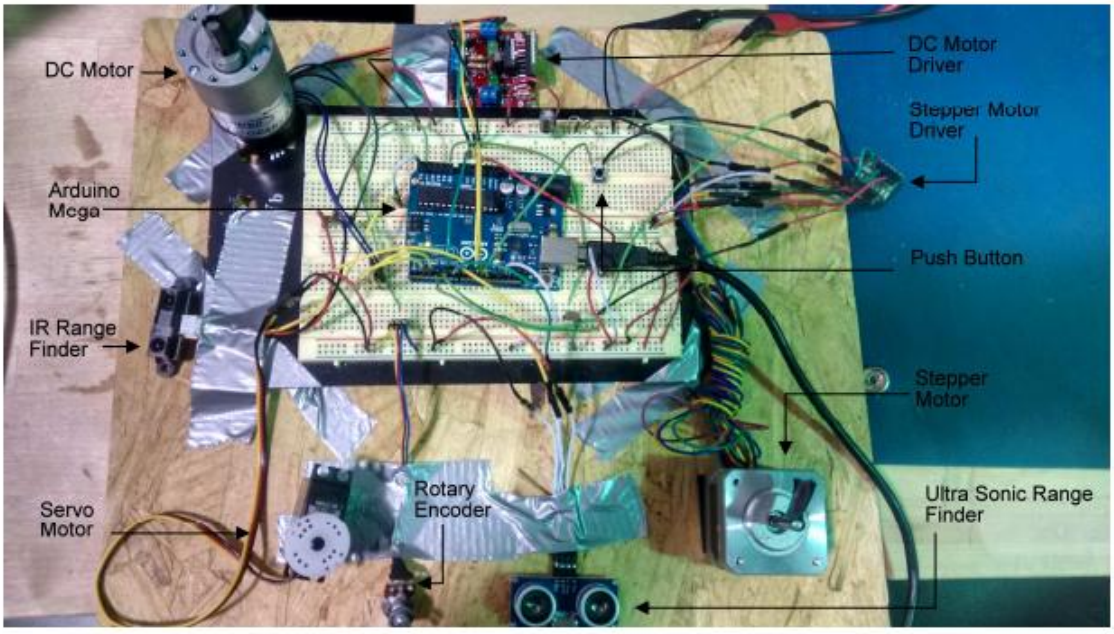

Figure 4: Final Circuit Diagram of the complete system
(Image Courtesy: Lekha Mohan)

II. Challenges

The first challenge I faced was with the Pololu A4998 stepper motor driver. The first two motor drivers I used were not working. I spent couple of hours trying to understand why the first driver was not working, debugging the circuit, ensuring all connections were proper. Finally, when I was confident that the circuit was correct, I tried two more drivers and the stepper operation worked with the third driver on the same circuit. This also made sure that the 2 drivers were in fact faulty.

Other minor challenge I faced was during integrating the complete code. Since we all did not follow structured coding standards, the complete code, once in place was messy and hard to understand. I was getting an error because one of my variable was getting over-written. The same global variable was being used at 2 different places in the integrated code. Once I got the whole operation to work, I refined the code by adding comments and removing anything unused code.

III. Teamwork

As a team we followed a very structured approach to divide the task within ourselves. We created a github repository to keep the code organized and decided the pin layout for Arduino with respect to each sensor and motor before starting with individual tasks. This really helped me integrate the complete code without any major issues. We then worked on our individual tasks:

Alex: Started with DC motor speed control using pressure sensor. Later took over the GUI development from Feroze and Lekha. Finally demonstrated the working of the system as a whole.
Feroze: Started with RC servo control using a potentiometer and later took care of the initial QT GUI development.
Lekha: Worked completely on GUI development, initially with Feroze and finally with Alex.
Rick: Started with DC motor position control using IR range finder sensor. Later worked with me and Alex to set up the complete circuit and helped me integrate the complete Arduino code.
Abhishek: I initially started with stepper motor control Ultrasonic range finder sensor. Later I worked to integrate the complete code and finally worked with Alex to test the system as a whole.

Although we had designated individual tasks to everybody, but we always worked as a team and helped each other. This way, we were able to accomplish the complete task successfully.

IV. Future Plans

For the Progress Review 1, my target areas are to work with Alex and Feroze to finalize the ROS skeleton code (base code) for our project. Besides, I will be working on doing extensive research on previous years Amazon Picking Challenge teams, understanding their game strategy, and pros and cons of their approach. This will help us in developing the best system for the required approach. On a whole, for the remaining part of this semester I will be targeting on two major things:

1. Electronic Design of the gripper sus-system.
2. Navigation Control of our system.

I will break down these tasks into smaller weekly goals and try to successfully accomplish these.

Appendix A

Pseudo code for stepper operation:

```
//Stepper Operation
// scale to use it with the stepper (value between 0 and 200,
stepper rotates by 1.8 degree per step)
val = map(motor_control_value, -100, 100, 0, 200);
stepper_val = val;

if (val>val_old) {
   digitalWrite(4,HIGH); // Set Dir high
}
else {
   digitalWrite(4,LOW); // Set Dir low
}
// Number of steps to rotate is based on the change in the pot
value
int steps;
if (val > val_old)
   steps = val-val_old;
else
   steps = val_old-val;

for(int x = 0; x < steps; x++) // Loop step times
{
   digitalWrite(5,HIGH); // Output high
   delayMicroseconds(1000); // Wait
   digitalWrite(5,LOW); // Output low
   delayMicroseconds(1000); // Wait
}
```

Appendix B

Pseudo code to read Ultrasonic range finder sensor value:

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);

//Calculate the distance (in cm) based on the speed of sound.
distance = (duration/29.1)/2;
```