ILR #1: Sensors and Motor Control Lab

Zihao (Theo) Zhang- Team A

October 14, 2016

Teammates: Amit Agarwal, Harry Golash, Yihao Qian, Menghan Zhang

Individual Progress

For my team's sensors and motor control lab, I was responsible for developing the software program to control the Mercury stepper motor (SM-42BYG011-25) and actually implementing it on the hardware. The control signal for the stepper motor was chosen to be the ambient lighting condition around the motor. The lighting condition was detected by a TEMT6000 ambient light sensor installed close to the motor. An Arduino Uno microcontroller board was used to collect signals from the light sensor in real time, generate control signals, and transmit them to the stepper motor.

The TEMT6000 ambient lighting sensor is essentially a phototransistor with higher precision than lighting sensors made of photoresistors. Its detection range is about the same as the human eye's. The measurement on ambient lighting condition from the sensor is expressed in terms of the analog voltage ranging from 0 to 5 volts. This is because the power supply to the sensor is typically 5 volts. The measurement data can be accessed after connecting the lighting sensor to our microcontroller (as shown in figure 1 below). However, the readings from the sensor in serial terminal will be converted into numbers from 0 to 1023 instead of value of analog voltage. Here, 1023 represents full brightness in our visible spectrum while 0 represents a completely dark environment around the sensor.

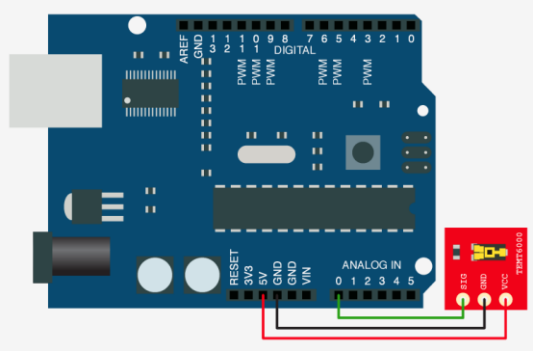

Figure. 1 Connection between ambient light sensor and microcontroller

A stepper motor requires a motor driver to energize each 'step' or phase in order to make it move forward or backward. Its motion is counted by 'steps'. This is also the reason that position of a stepper motor can be controlled more accurately than a DC or servo motor.

In our case, the TI DRV8825 bipolar stepper motor driver was selected. The male headers were first soldered in so that the driver could be attached to the breadboard. The driver was then connected to both the Arduino microcontroller and the stepper motor, as shown in figure 2 below. It took control commands generated by the microcontroller as the input and sent the signal output to directly control the motor. Note that Arduino board, with only the 3.3v and 5v power supply options, was not able to provide sufficient voltage to the stepper motor driver.

Therefore, the DC power generator in the lab was used to provide 9 volts to the motor driver. In addition, a 100 micro-farad capacitor was needed across the power supply in order to stabilize the voltage as current changed.

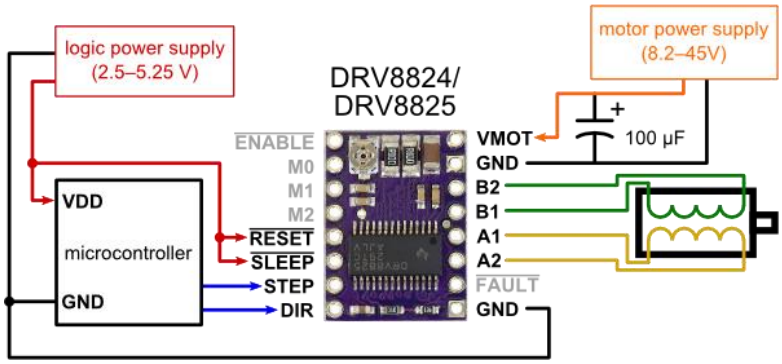

Figure. 2 Connection of the motor driver to the stepper motor and microcontroller

In terms of motor control, two different types of control on the stepper motor were developed using the same sensor input, namely the speed control and the position control. Under both cases, the Stepper.h library in Arduino were used to simplify the design process.

For the speed control, brightness value from 0 to 1023 was obtained using the analogRead() function and was mapped to rotational speed of the stepper motor from 0 to 60 RPM (1 rotation per second). In other words, the brighter the surrounding environment was, the faster the motor would rotate; a completely dark environment would just keep the stepper motor stationary. The setSpeed() function was called to set the stepper motor speed. The motor speed, directly related by the brightness value, was updated every 4 steps using the step() function in a loop. Note that a positive value in the step() function stood for clockwise rotation and vice versa.

For the position control, on the other hand, brightness value from 0 to 1023 was mapped to steps of the motor from 0 to 200, as one revolution of the motor is comprised of 200 discrete steps. Therefore, a change in ambient lighting condition from complete darkness to full brightness would make the motor turn 360 degrees, assuming it started at 0 degree. No change, or minimal change below a defined threshold, in surrounding brightness would result in no motion for the motor. The brightness value was updated every 100 milliseconds. In every single loop, the current/new step value, mapped from the updated brightness value, was compared with the last step value. If the difference between the current and the last step value was above our defined threshold, the motor would be moved by this difference in steps to its new position.

Both methods were successfully implemented on the hardware as the final result. However, only the position control on the stepper motor was implemented during the lab demonstration, because it was the only explicitly stated requirement in the instruction.

Challenges

In general, the main challenge I encountered during this lab came from insufficient past experience on microcontroller programming. My lack of knowledge on working principles of sensors and the stepper motor also added the difficulty on top of the challenge. Nevertheless, spending more time on online tutorials and investigating datasheets of the components certainly alleviated the difficulty and helped increase my working pace to a great extent.

In terms of specific technical issues, one good example is setting proper parameters for position control on the stepper motor. These parameters include the time delay between each single loop, the upper bound step value corresponding to 1023 (full brightness) in the map() function, and the threshold on difference between the current and the last step value. For better illustration purpose, a screenshot of the code is attached as the following:

```
void loop() {

  int brightness = analogRead(light_pin); // 0-1023
  Serial.println(brightness);

  brightness = analogRead(light_pin);
  currStep = map(brightness, 0, 1023, 0, stepsPerRevolution);
  if (currStep - lastStep >= 2){
    myStepper.step(currStep - lastStep);
    lastStep = currStep;
  }
  delay(100);


}
```

Since we would like to make the control system robust enough and sensitive to input change from the light sensor, the time delay should be kept relatively small. At the same time, because of the high sensitivity of the ambient light sensor, any subtle change on the brightness could result small degrees of motion in the motor. This continuous subtle change in real time would cause the motor shaft to fluctuate all the time about a certain position, greatly increasing the heating rate of the motor. Decreasing the upper bound step value in the map() function was one way to solve the problem. However, this would result in smaller degree of motion in general. Alternatively, a threshold was defined and added in the loop to better solve this problem. For example, when the range between 0 and 1023 was be mapped to the range between 0 and 200, change in every 5 units of brightness would result in one step in the motor. After the threshold value of 2 was added as shown in the picture above, any subtle change in brightness below 10 units would be filtered out. This way, the position control with better stability and less power consumption by the motor was achieved.

Teamwork

Major job duties of my teammates:

Amit Agarwal	DC motor control (position and velocity) using IR sensor
Harry Golash	GUI, buzzer control using IR sensor, and code integration
Yihao Qian	Servo motor control using the pressure sensor with switch
Menghan Zhang	Servo motor control using the potentiometer with switch

Yihao and Menghan first combined their circuits on a single breadboard and together designed a physical switch to shift between the control from the pressure sensor and from the potentiometer on the servo motor.

After that, I combined the circuit for the ambient light sensor, the stepper motor, and its driver with the integrated one on the same breadboard.

Lastly, Amit combined the circuit for the IR sensor, the DC motor, and its driver with the integrated one on the same breadboard.

I, Yihao, and Menghan also conducted the experiment on determining the transfer function plot for the SHARP 2Y0A02 IR sensor that we used for controlling the DC motor.

In addition, I made the display stand with Amit to hold our motors, sensors, and the circuit board together for lab demonstration.

Plans

Before the next ILR/ progress review, our team would like to achieve the following goals on our project of developing the perception system using Stereo Vision and radar:

1. Finalize the design of mounting rack as well as the method of fixing the cameras and radar on the mounting rack
2. Make the cameras functional (i.e., be able to provide images to the computer)
3. Get started on synchronization of the two cameras for the stereo camera system that will be built later

I will be mainly responsible for designing a holder for the cameras and radar that can be attached rigidly to the mounting rack. I would also like to help Harry design a waterproof housing for our sensors and assist other teammates on setting up the cameras.