Sensors and Motor Control Lab

Individual Lab Report #1

Hillel Hochsztein

Wholesome Robotics (Team E)

Teammates: Aman Agarwal, Dung Han Lee, John MacDonald, Aaditya Saraiya

February 14, 2019

Individual Progress

Sensors and Motor Lab

My primary role for the sensor and motor lab was to wire and code the servo motor and the Omron Slot Sensor. To program the servo, I used the Arduino native Servo.h library. This library allows the user to create a servo instance for a given pin on the Arduino, then the servo can be rotated to any angle using *servo.write(joint_angle)*. Since the Servo.h library is so complete, my main task for this implementation was to wrap the library into the format specified by our team integration plan. Doing so meant creating a struct that holds the pin number as well the servo handle,and implementing a constructor, *init_servo(struct servo_data *data)*, and a servo writing method, *handle_servo_position(int deg,struct servo_data *data)*.

The servo constructor method instantiates the servo on the chosen pin and stores that servo object's handle back into the struct. The servo writing method takes the servo handle and runs Servo.write() on it with the inputted angle.

Whereas the servo just has power, ground, and signal lines, the Omron Slot Sensor had a slightly more complicated wiring. Specifically, it required a resistor to limit the current throughput on the LED side, though the datasheet failed to mention a specific resistance value. Using an online calculator[1] I determined that for the suggested LED current and LED voltage drop, I needed a resistance of about 127Ω. I used 2 220Ω resistors for an effective resistance of about 110Ω (within the specified tolerances). The detector side of the sensor is wired with a source voltage, and a signal to a digital pin on the Arduino with a pulldown resistor to ground, to prevent floating values (see figure 1).

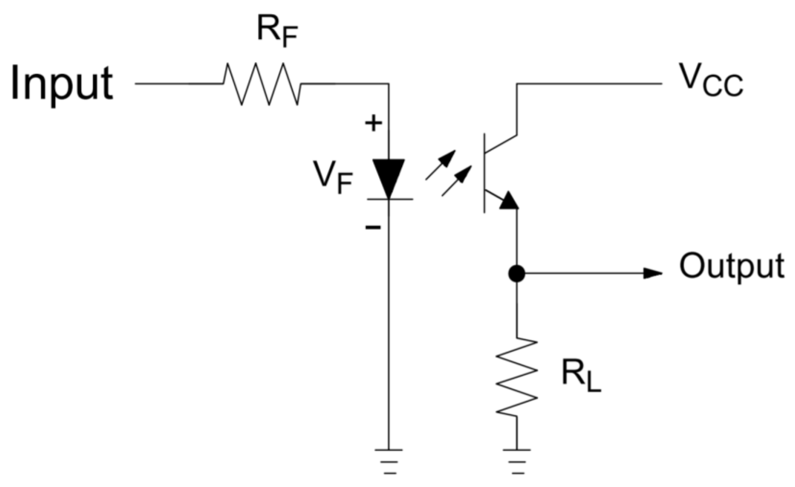

Figure 1 Wiring Diagram for Omron Slot Sensor. R_F is current limiting resistor, R_L is pulldown resistor[2]

[1] http://ledcalc.com/
[2] https://www.mouser.com/catalog/specsheets/sx1025.pdf

Coding the slot sensor with our predetermined integration scheme, I created a constructor, *init_slot(int pin_num),* that sets the pin associated with the sensor input to be a digital input pin. The function get_slot() returns a boolean: True if the slot is blocked, False if it is not.

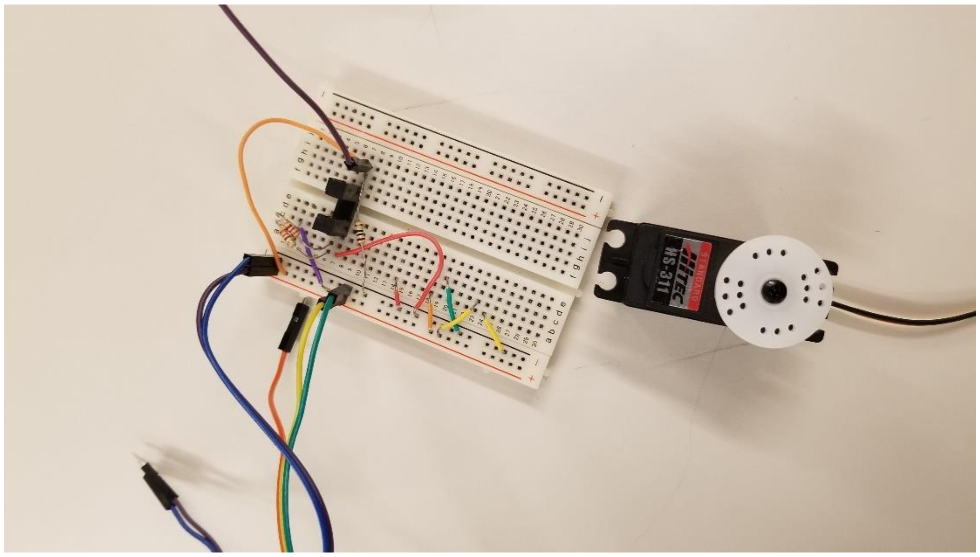

Figure 2 Wiring of the Servo and Omron Slot Sensor

Wholesome Robotics

In the past few weeks, I have been working on mechanical assembly of the Robotanist platform as the basis of our robot. We were given a CAD model and other documentation of the existing Robotanist and told that many of the components needed to build the robot were in the Newell Simon Highbay. Our first task was to inventory our supplies, during which we discovered that we have all the major electrical components, except for a main computer, an IMU, and some peripherals (RAM, cables, etc.). We do not have any of the structural parts, but with the CAD model we are in the midst of updating these parts and plan on sending them out for manufacturing shortly.

We were given, among the documentation provided, a high-level schematic of the robot's wiring. This was written on paper and outdated relative to our plans, so I was tasked with recreating this schematic in a digital format that can be easily updated as changes are made (see figure 3). In doing so, I created a color-coding plan for cataloging both the types of wiring used and the status of individual connections.

Next, I worked on the actual wiring for the drive motors. The robotanist uses Hebi Robotics' driver board to control BLVM620K motors made by Oriental Motor. The motor is a three phase DC Brushless motor, and has an incorporated hall sensor for feedback. The Motors do not come with compatible connectors for the Hebi board, so I spent a number of hours tracking down the correct types of connectors and rewiring the motors.

I have also been in communication with the Phipps conservatory, and have secured a small area to begin testing our robot. They have graciously agreed to plant three rows of kale so that we have a local test site (the robot can drive to the test site and does not need special transportation arrangements).

Finally, I have been working on designing the end effector that will be used to cultivate weeds. I have designed a cad assembly (see figure 4), with focuses on ease of design and cost (this will be a prototype for an iterative process). The assembly is made from #10-32 threaded rods that are ground down on one side and screwed into a plate on the other side. They will be Zinc coated for corrosion resistance, as they will be placed directly into the ground. The aluminum plate is attached via set screw to a motor axis. The total cost of parts will be around $10 per end effector and the machining operations are very trivial and have high tolerances.

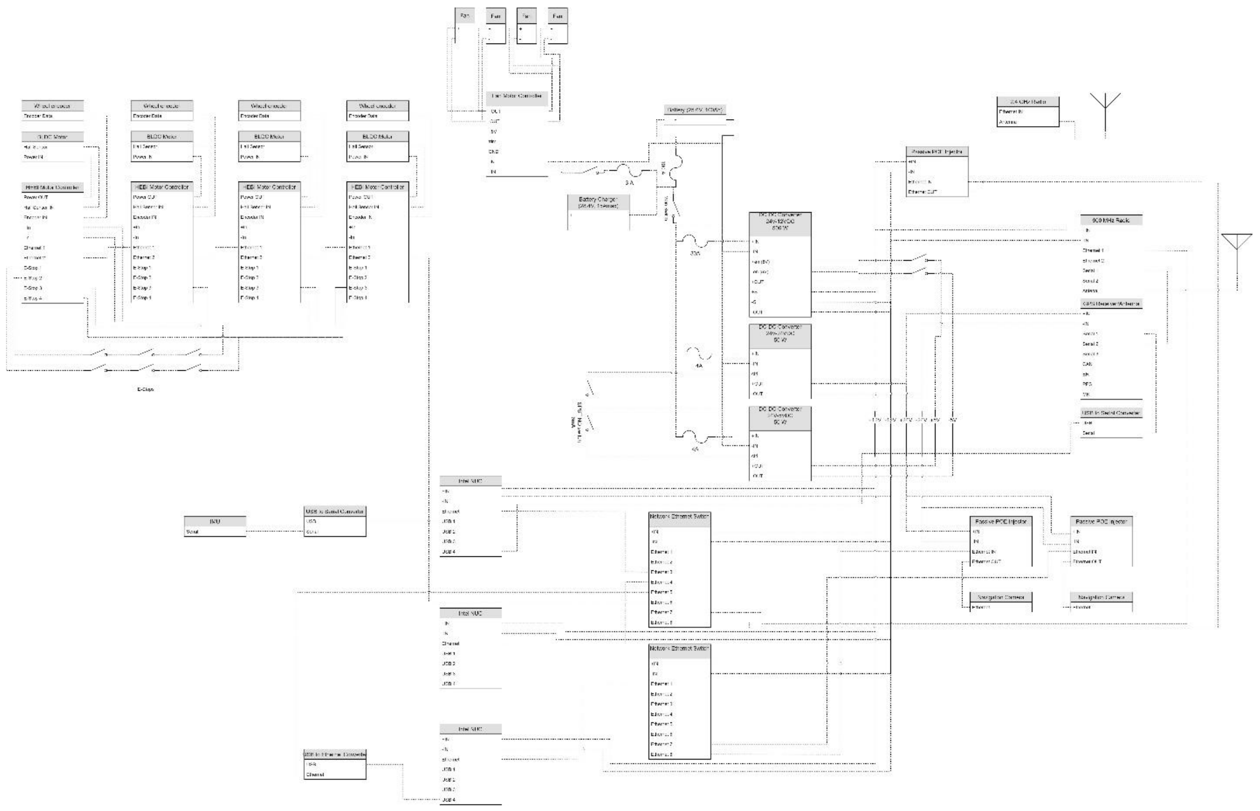

Figure 3 Electrical Wiring Schematic

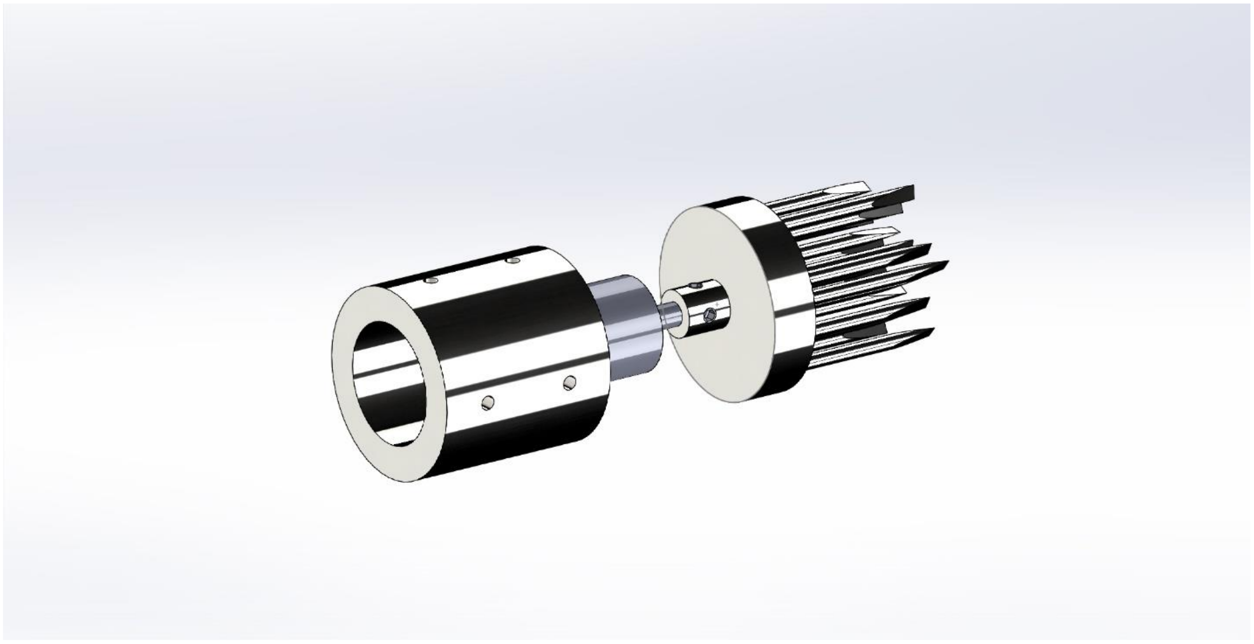

Figure 4 Assembly of first end effector design

Challenges

Motor and Sensors Lab

My main challenge for the Motor and Sensors Lab was C++. I have experience working with Arduino, but only in very DIY settings. As a team we made a plan for function naming and data types so that integration would be easier. However, doing so meant that my responsibilities (the servo and slot sensors) became more complicated as I had to generalize their usability into our integration plan. Specifically, instead of using just the native Servo.h library, I had to wrap its functions into our schematic. Doing so included creating a struct to hold the servo's handle and pin number. Having no prior C++ experience, manipulating the struct, particularly regarding addresses, was particularly difficult for me, and required a bit of online searching to properly implement.

Wholesome Robotics

A key challenge for us is that we are building our robot using the backup parts from the existing robotanist. This is very helpful because we have a model to work from, however, things are not all organized intuitively. For example, I spent multiple hours searching for the specific connector (C-grid III) needed to connect the motor's hall sensor to the Hebi board. This is unfortunately a challenge that will likely continue to plague us, though, as time goes on and we begin finding the parts that we need, we will be able to organize our supplies better.

Teamwork

Motor and Sensors Lab

Aman coded the PID controller for the DC motor and the temperature sensor.

Aaditya coded the IR sensor and the PID controller for the DC motor.

Dung Han coded the stepper motor (including soldering the driver board) and the ultrasonic sensor.

John designed the GUI and serial communications.

Wholesome Robotics

Aman has been working on finalizing the structural parts of the robot that we will be sending out shortly.

Aaditya has been working on the mapping algorithms.

Dung Han has been working on the perception algorithm for recognizing holes in the plants' leaves.

John has been working on the perception for navigation algorithms.

Future Plans

Wholesome Robotics

In the coming weeks my plan is to begin some manufacturing and assembly. Specifically, I will be ordering parts to begin work on the end effector (I will be using a cordless drill as the actuator for testing different cultivators). I will also be setting up test areas with weeds to test the end effector. I will also begin to wire electrical parts of the robot and assemble structural parts of the robot as they arrive.

Quiz

1. Reading a datasheet
   a. The sensor's range is ±3.6g
   b. The sensors dynamic range is 7.2g
   c. The C_{DC} capacitor is used to decouple any noise from the power supply. A 0.1µF capacitor should remove all noise other than that at 50kHz (or any harmonic of that), which is the clock frequency of the accelerometer.
   d. $V = 1.5 + 0.300 * a$ where V is the output voltage and a is the input acceleration
   e. The maximum linearity error is 0.3% of full scale or 0.0216g
   f. 750µg
   g. You can determine this experimentally by dropping the accelerometer (onto something soft) during the fall this will give a constant acceleration of 1g (i.e. 0 Hz) any variation from 1g can be taken as noise (compute root mean square of this noise).
2. Signal conditioning
   a. Filtering
      i. Moving averages are problematic for any sharp changes: a single large value will increase the value of the surrounding points.
      ii. Median filters remove sharp changes, but are consequently slow on the uptake for any quick changes to the actual signal.
   b. OpAmps

i. We want a function $V_{out} = 2V_{in} + 3$ which means that V_1 is the reference and should be at -3V, V_2 is the input, and R_f/R_i=1.

ii. We want a function $V_{out} = V_{in} + 2.5$ which is impossible, because in order to have any offset in this configuration there must be a gain ≠1.

3. Controls
   a. An encoder can record both position and velocity of a dc motor. The position term is compared to the desired position for the proportional signal, and consequently, the velocity term can be used for the derivative signal. The integral signal is created by adding up all the proportional term errors since the beginning of operation (the integral gain should be very low, and the designer should consider making a reset for the integral signal).
   b. If the motor is sluggish, you may want to increase the derivative gain, or lower the proportional gain – both may lead to overshoot.
   c. The integral term is best for eliminating steady state error, as it basically increases its effect on the control signal the longer the value remains away from the target value.
   d. Adjustment to the derivative gain does not effect steady state error and can be adjusted (increasing will fix overshoot) after fixing steady state error.

Motor and Sensors Lab Code

```
#include <Servo.h>

struct servo_data{
  Servo handle;
  int pin_num = 9;
};

void init_servo(struct servo_data *data){
  Servo s = data->handle;
  s.attach(data->pin_num);

}

void handle_servo_position(int deg,struct servo_data *data){
  Servo s = data->handle;
  s.write(deg);
}

void init_slot(int pin_num){
  pinMode(pin_num,INPUT);
}

boolean get_slot(int pin_num){
  if (digitalRead(pin_num)==HIGH){
    return true;
  }
  else{
    return false;
  }
}
```