Lab 3: A Line Following Robot

Julia Benton & Richard Gao

October 15, 2019

1 Description

In this lab, we created a line-following robot using a given robot chassis, an Arduino, a motor shield, and two infared reflectance sensors. In completing the previous labs, we learned how to output digital signals, read analog inputs, and utilize these two skills to measure and act on the physical world. For this lab, we integrated our ability to sense the physical world through sensors connected to a microcontroller with our ability to interact with the physical world through actuators.

We integrated sensing and control with a two-wheeled, mechanical robot and wrote a closed-loop controller that runs on the Arduino that enables our robot to follow a track consisting of electrical tape laid out on the floor. We tuned our controller without recompiling and reloading the code to enable our robot to complete a lap around the course as quickly as possible.

We used the IR sensors to detect the presence or absence of the tape line on the ground, and our controller logic used that information to compute the signals that must be sent to the two independently-moving DC motors to enable the robot to follow the line. The IR sensor consists of an infrared light emitting diode and a phototransistor. The more IR light that is reflected back to the phototransistor, the closer Vout will be to zero. We used the reflectance reported by our sensors as signals around which to design a feedback loop that ensured our motors would keep our robot following the tape line.

2 Process

We started by creating a wiring harness going from the screw terminals on our barrel jack adapter and the screw terminals on our motor shield to a 4 pin Molex connector. We received a wiring harness that attaches a barrel jack to screw terminal adapter on our Arduino and the screw terminals on our motor shield. At the other end was a four pin, wire-to-wire Molex connector shown below. We also made sure the power jumper was removed from our motor shield to prevent our shield from trying to take power from the Arduino's USB connection.

With our power wires fully assembled and in place, we designed a circuit for integrating our two IR tape sensors, seen in Figure 1. To ensure that we were able to reliably detect the difference between the tape and the floor, we calibrated and tested our IR sensors with multiple resistance values for R_sense. Optimizing for the most notable difference of readings between the sensor on the tape and on the floor, we landed upon the 30kΩ resistor. Using this resistor, we were able to

accurately and repeatedly read values above 700 on the analog pins when the sensor is above the tape, and values below 700 when off the tape.

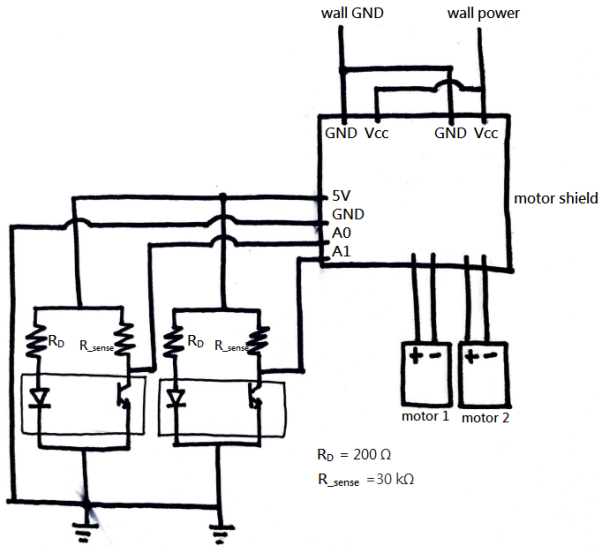

Figure 1: Our circuit which connected the IR reflectance sensors, and the *R_sense* value that we optimized.

We then affixed our electronics and sensors to the robot chassis with duct tape and screws and connected the two DC motors to the motor shield arriving at the system seen in Figure 2 and Figure 3.

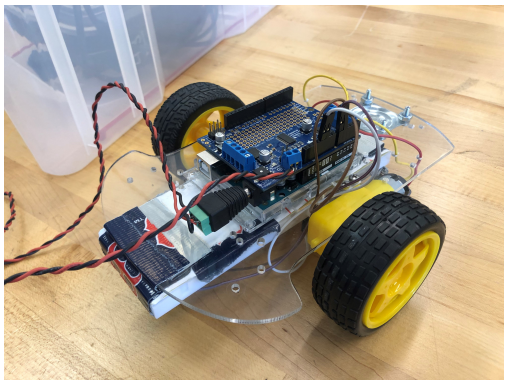

Figure 2: Our fully-assembled line follower showing our mechanical integration with the chassis. The Arduino sits on top of the chassis connected to the breadboard taped below which holds the two IR sensors.

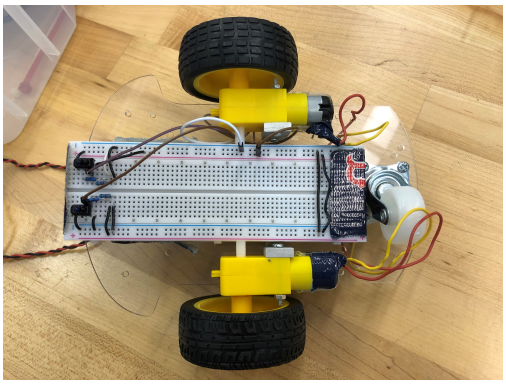

Figure 3: Our robot viewed from underneath to showcase our circuit built on the breadboard. The distance we put between our IR sensors is clear in this image.

Reading the motor shield documentation and experimenting, we came up with our controller

code to allow the robot to follow a line. By default, the robot moves forward in a straight line. Using the values reported by the IR sensors, the robot detects when a sensor passes above the black line (when analogIn is above 700 the sensor is above the line, when below 700 it is not). When this happens, the robot must react to continue moving along the line as being above the line means that the robot is intersecting the tape and not following parallel to it. Therefore, we program the robot to turn in response, realigning the robot to a position that when moving forward, drives with the line. So, when the left IR sensor reports a value greater than 700, the robot turns right to readjust, and when the right IR sensor gives a similar reading, the robot turns left.

```
// From LineFollower.ino

leftValue = analogRead(leftSensor);
rightValue = analogRead(rightSensor);

// Detect when to left turn
if (!rightTurn && !leftTurn && rightValue > threshold) {
    oldLSpeed = lSpeed;
    lSpeed += turnSpeed;
    leftTurn = true;
} else if (leftTurn && rightValue < threshold) { // End left turn
    lSpeed = oldLSpeed;
    leftTurn = false;
}

// Detect when to right turn
if (!leftTurn && !rightTurn && leftValue > threshold) {
    oldRSpeed = rSpeed;
    rSpeed += turnSpeed;
    rightTurn = true;
} else if (rightTurn && leftValue < threshold) { // End right turn
    rSpeed = oldRSpeed;
    rightTurn = false;
}
```

Using our controller code, we set our robot down on the track and successfully completed a loop following the line. You can check out our robot in action here.

We also gathered the IR sensor data and wheel motor speed and created the plot shown in Figure 4 for our robot along a snippet of the track. This superimposed plot shows how when the IR sensor reports a voltage above a threshold, the robot decides to turn, modulating the motor speeds. Because we decided for the robot to make sharp turns, we turned by restricting the motor in the direction of the turn, and this can be seen as the motor wheel speed slams down when above the threshold.

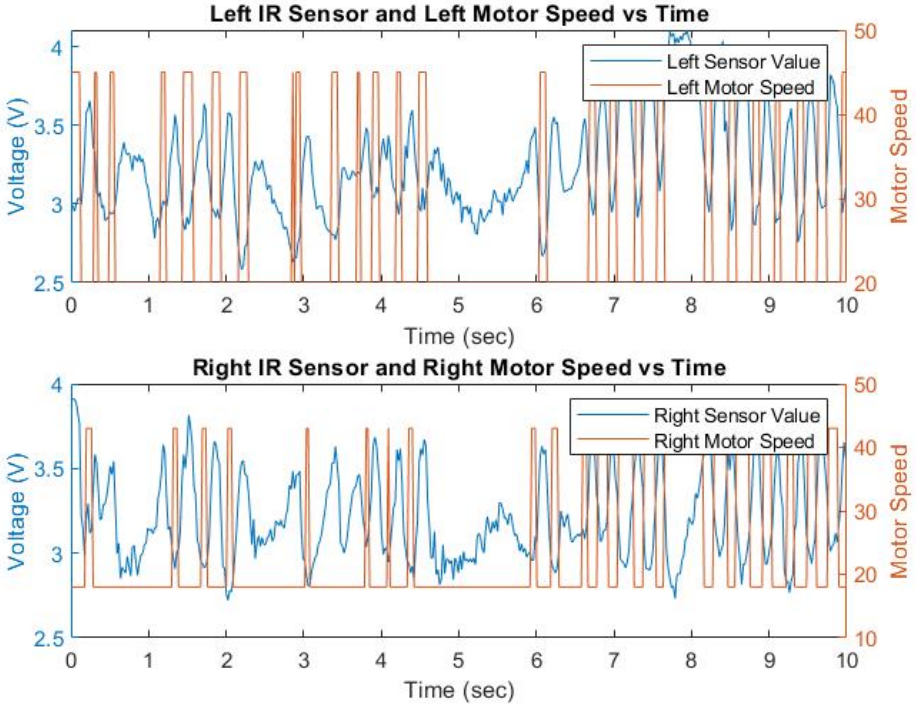

Figure 4: Yyplot superimposing sensor data from both IR sensors and commanded motor speeds for a short run.

3 Reflection

In this lab, Julia learned about several new electrical engineering concepts and got to try her hand at building new things, such as a wiring harness and crimping. This lab was a great learning opportunity since Julia has not done electrical engineering since ISIM first semester at Olin. She realized the importance of plugging wires into the proper positive or negative terminals in the wiring harness. She also learned that on the other hand, if a motor spins backwards when connected, it is simply a matter of switching wires. She learned about how an IR reflectance sensor works with them containing both an infared emitter and a phototransistor. Julia also gained more experienced in creating schematic diagrams (thank you to HK Rho for the basic schematic drawing on which we added our values).

Richard learned how to interface with the motor shield and write firmware code to control two DC motors independently. He also realized the importance of designing a feedback loop that is most efficient to the task at hand. This is because he initially designed the feedback loop where the sensors were placed close together and reported when any sensor moved off of the black taped line. But because the tape was relatively thin, this high fidelity detection was not possible or accurate using the IR sensors. The more optimal solution and the solution we landed on was to have a sensor detect when it "sees" a black line and turn the car to position it away from that line. The code for this process turned out to be far more robust and efficient.

For the future, we can try building our own chassis for a more solid design. This way instead

of duct-taping the Arduino and breadboard to the frame, we can have a custom mount so that the components are firmly fixed to the robot. We can also do away with using a breadboard by soldering the IR sensors to a PCB so that the sensors are closer to the ground to give more accurate readings.