ME 375 Lab 6 Report
myRIO Sheepdog Project

Prepared by:
Emily Udell (eudell@purdue.edu)
Akash Mattupalli (amattupa@purdue.edu)

School of Mechanical Engineering,
Purdue University
585 Purdue Mall
West Lafayette, IN 47907

Date: 2023.12.3

1 Introduction

For the final project, we built and programmed a two-wheeled MyRIO 'sheep-herding' robot that performs two behaviors: line-following and herding. To meet the requirement of the robot fulfilling the task autonomously with closed-loop control, we implemented a *Control & Simulation* loop on LabVIEW for each behavior and used PI controllers to control the wheel velocities.

2 State Machine Design

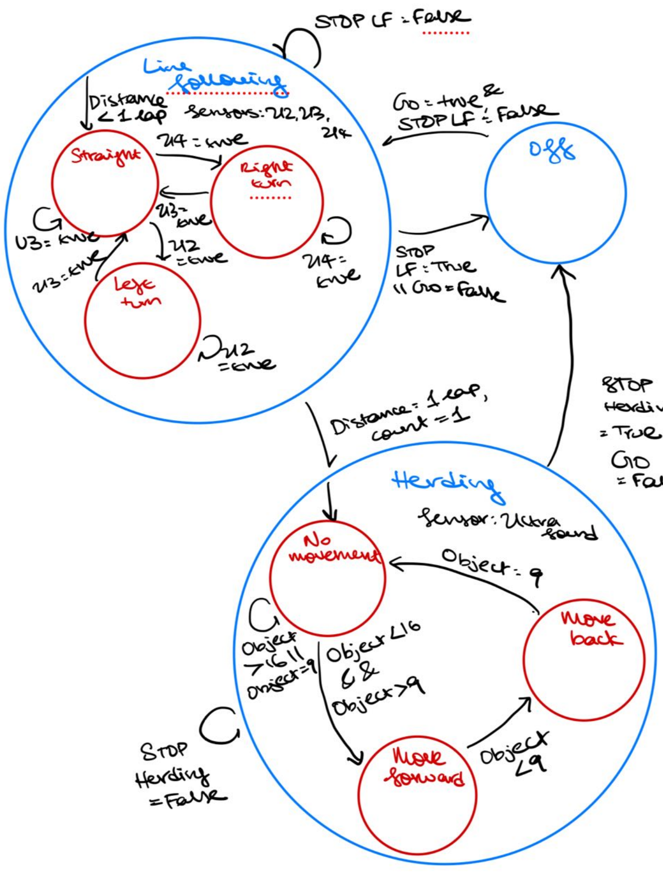


Figure 2.1: Finite State Machine

A nested finite state machine (FSM) was used to represent the robot's actions and behaviors. The transitions between the *off* state and the overarching *line following* and *herding* states are controlled by the *Go, STOP LineFollower,* and *STOP Herding* buttons as well as the *count* variable that tracks the distance traveled by the robot.

The transitions between the *straight, right turn,* and *left turn* states in *line following* are dictated by the line follower sensor readings U2, U3, and U4 that output 'True' if they sense a black line. If U3 is true, the robot will move straight. If U4 is true, the robot will turn right and if U2 is true, it will turn left.

Once the robot completes the lap, the count variable is set to 1, activating *herding.* The transitions between the *no movement, move back,* and *move forward* states in *herding* are controlled by the object distance sensed by the ultrasound sensor. The robot will not move if the object distance is 9" or more than 16”. If the object distance is less than 16” and more than 9”, the robot will move forward. If the object distance is less than 9”, the robot will move backward. The robot will transition to the *off* state if *Go* button is false or either of the *STOP* buttons is true.

3 Controller Design

For the overarching states of *line following* and *herding,* two PI controllers were used as a closed-loop feedback mechanism to control the wheels’ velocities. The measured/sensed motor positions and velocities were obtained from the MyRIO and incorporated into the Control & Simulation loops to compare with the desired velocities.

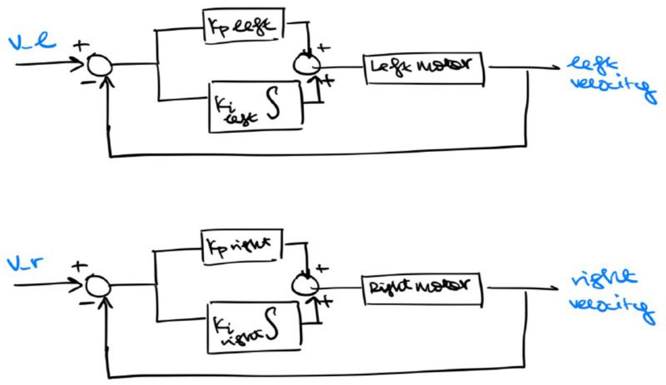


Figure 3.1: PI controllers of both motors

The values of the controller gains, while being based on calculated values from previous labs, were changed on a trial-and-error basis. We noticed that the robot exhibited ‘jittery’ behavior if the gains were high, and did not respond well at the turns if the gains were low. The final gains for the *line following* were:

$$K_{p,left} = 1.2 \quad K_{p,right} = 0.9$$
$$K_{i,left} = 0.4 \quad K_{i,right} = 0.3$$

Our gains for the left motor were higher than the gains for the right motor since at equal gains the robot would turn right, due to the left motor rotating faster than the right. By increasing the gains, we were able to quicken the speed of the response for the left motor and ensure that the robot would follow the line.

A similar process was conducted for the *herding* state, with the following gains:

$$K_{p,left} = 0.6 \quad K_{p,right} = 0.5$$
$$K_{i,left} = 0.4 \quad K_{i,right} = 0.3$$

The gains are different from the *line following* state, and we theorized that this could be due to the different sensors being used. We initially set the gains to be the same, but lowered it since the robot was jittery, a sign that the controllers were too fast for the ultrasound sensor.

4 Performance

Once the controller was designed, a lot of the performance came down to optimizing the velocity on the straight lines, as well as the turning velocities. This allowed us to choose the fastest speed while still being able to consistently complete the course. Trials were performed at system line velocity values of 9, 12, 15, and 20 counts/sec. The average lap time of each and the number of successful laps out of three can be seen in Table 4.1. Using this data, the conclusion was made that a line velocity of 12 counts/sec would be used during the trial. Interesting things to note are that the controllers weren't quite fast enough to be consistent at velocities of 15 and 20, but at lower velocities, like 9, the travel also became inaccurate. This is why the velocity ended up being optimized at around 12 counts/sec.

Velocity (counts/sec)	Average Lap Time (s)	Successful Laps
9	32.31	2/3
12	21.59	3/3
15	17.12	2/3
20	13.41	2/3

Table 4.1: Average Lap Times and Success Rates at different velocities

In order to properly turn, through testing, it was found that the motor on the side of the turn the robot was taking should be operated at 1.6*velocity. The other motor should operate at 0.5*velocity. These multipliers were directly embedded in the code.

The response graph of the wheel velocities over time followed this general form shown in Figure 4.1. This form shows the desired left wheel velocity in light blue, the actual left wheel velocity in dark blue, the desired right wheel velocity in black, and the actual right wheel velocity in red. This graph illustrates the response of the wheel velocities to the desired controller and shows a quick response to try to meet the desired velocity while eliminating error. The alternating pattern is due to the consecutive turns on the ends of the track.

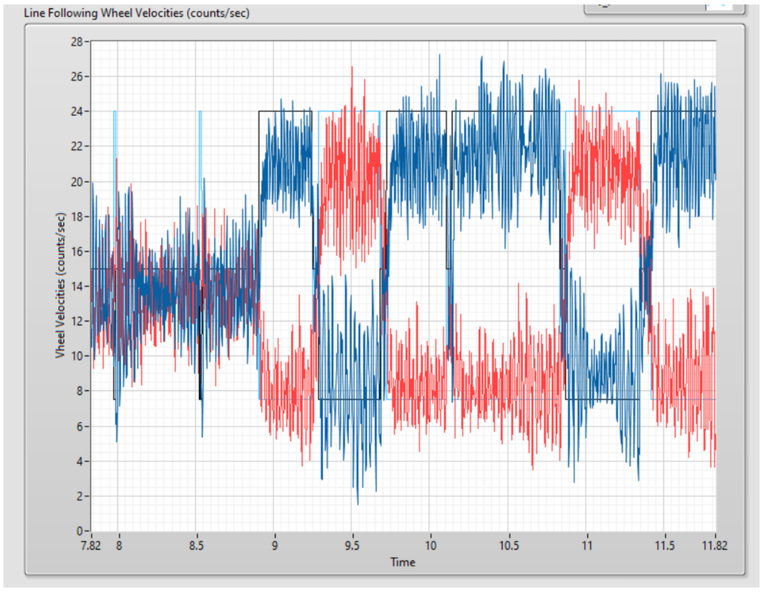
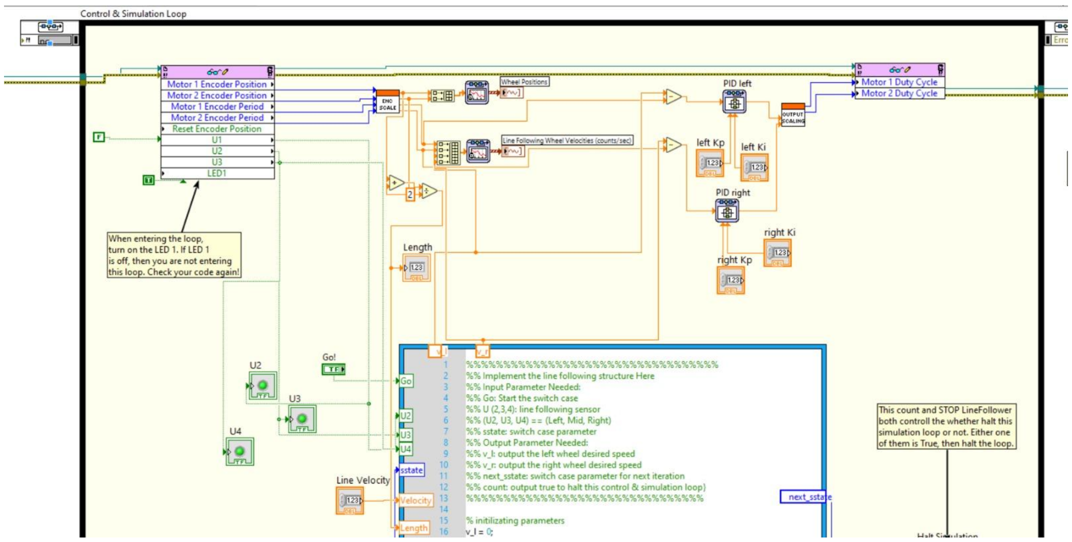


Figure 4.1: Response graph at Velocity = 15

One additional performance aspect was the relative accuracy of each motor. It was noticed during testing that the left motor had more power than the other. Because of this, a multiplier of 1.1 was applied to the general velocity of the right motor when completing the herding, so that the robot could herd in a straight line.

5 Code Description

The LabVIEW project code used to complete the project objectives utilized one main VI to control both the line following and the herding. This VI takes in readings from both the line following sensor, and the ultrasonic sensor, applies the designed PI controller, and outputs assigned wheel velocities based on these. This VI can be seen in Figure 5.1 and 5.2. One sub-VI was used to transmit readings from the ultrasonic sensor to be used in the herding logic. The expanded sub-VI is shown in Figure 5.3. Two separate control and simulation loops were used for the line following and herding. As shown in the state machine design section, the code uses a true/false count powered by a distance reading to begin the herding code after one full lap is achieved on the track.



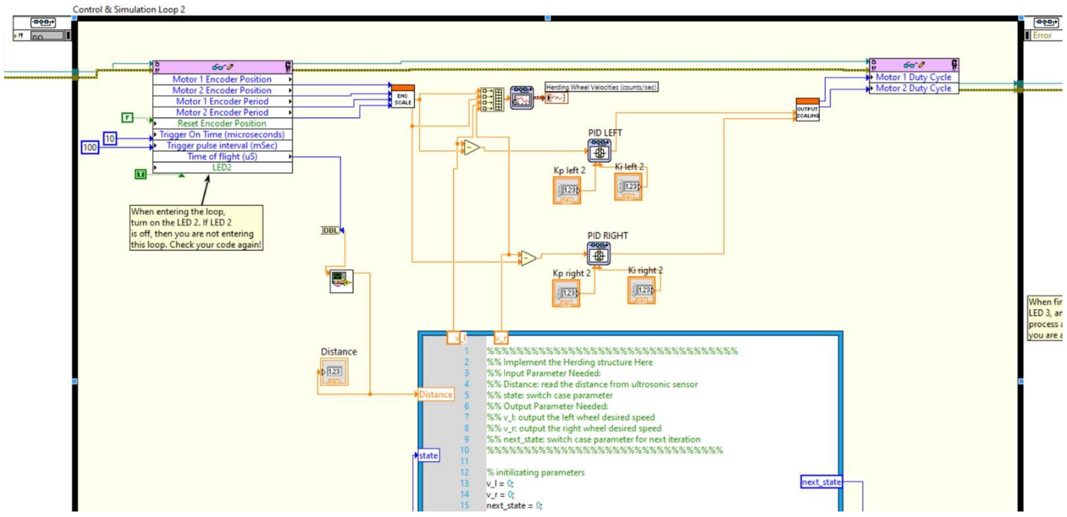
When entering the loop, turn on the LED 1. If LED 1 is off, then you are not entering this loop. Check your code again!

This count and STOP LineFollower both controll the whether halt this simulation loop or not. Either one of them is True, then halt the loop.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Implement the line following structure Here
%% Input Parameter Needed:
%% Go: Start the switch case
%% U (2,3,4): line following sensor
%% (U2, U3, U4) == (Left, Mid, Right)
%% sstate: switch case parameter
%% Output Parameter Needed:
%% v_l: output the left wheel desired speed
%% v_r: output the right wheel desired speed
%% next_sstate: switch case parameter for next iteration
%% count: output true to halt this control & simulation loop)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% initilizating parameters
v_l = 0;
```

Figure 5.1 Line Following Simulation Loop



When entering the loop,
turn on the LED 2. If LED 2
is off, then you are not entering
this loop. Check your code again!

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Implement the Herding structure Here
%% Input Parameter Needed:
%% Distance: read the distance from ultrosonic sensor
%% state: switch case parameter
%% Output Parameter Needed:
%% v_l: output the left wheel desired speed
%% v_r: output the right wheel desired speed
%% next_state: switch case parameter for next iteration
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% initilizating parameters
v_l = 0;
v_r = 0;
next_state = 0;
```

Figure 5.2 Herding Simulation Loop

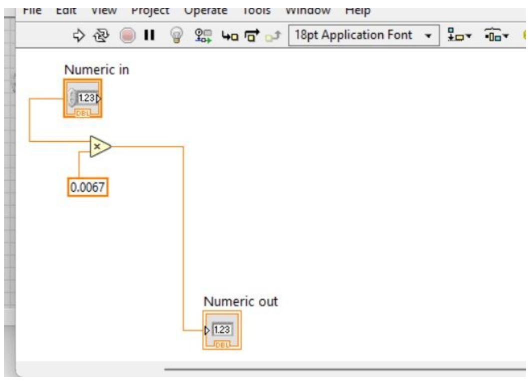


Figure 5.3 Ultrasonic Sub VI

The FPGA I/O layer plays a crucial role in communicating the software component inputs to the hardware component outputs. The main data providing inputs come from both the line sensor and ultrasonic sensor. The line sensor reads for a black line in three locations at the front of the robot, one on the left, middle, and right. This data can be used to see if the robot is on the line, and if it's off, what way it needs to turn. In addition to this, the ultrasonic sensor provides distance measurements. This data is used in the herding portion of the code to make the robot move forwards or backwards based on the distance of the item from the robot. No unit conversions were used outside of the ones embedded into the loops.

The feedback control loop follows the structure of a proportional-integral controller. This controller looks at the difference between the desired velocity and actual velocity of each wheel based on the code, and then adjusts the control signals sent to the motor to minimize any error that may occur. In general, the proportional part should help reduce any steady state error, while the integral component has the ability to adjust the control output further to reduce any remaining error after the proportional controller attempts to. The system uses input velocity, or desired velocity, as the reference. This setpoint is what the controller will attempt to adjust the motor outputs to.

In order to meet all objectives autonomously, several key items were implemented into the program. These items can be seen in the state machine design section. After the system's line following is started by the user, the distance count allows the system to automatically switch over to herding. The rest of the details can be seen in the diagram in Figure 2.1.

6 Discussion

Overall, the controller performed extremely well. The robot was able to follow the line and herd with precision and ran quite smoothly. See the performance section for additional information on lap speed and success. There were a few problems encountered during the project that the team managed to mitigate. The first was that the robot was led astray by scuff marks on the track. This ended up being solved by moving the line sensor closer to the ground with the addition of washers to allow for a more concentrated reading of the correct line. This system could be improved by having a faster response system. This could be done by creating a faster controller. This might allow the team to run at higher velocities more consistently without failure during line following.