ME 375 FINAL PROJECT REPORT

ROBOT CONTROL COMPETITION

Team Member: Gabriel Kurfman
Contribution: 50%
Summary: Contributed to the Introduction, Proposed Controller Structure and Design, and Potential Problems sections of the written report. Also worked on controller design and Simulink code.

Team Member: Michael Olson
Contribution: 50%
Summary: Contributed to the Subsystem Characterization and Full System Model section of the written report. Also worked on controller design and testing.

1 INTRODUCTION

1.1 The Competition

The ME 375 final project involves a robot competition where teams program an autonomous robot to complete two main tasks: line following and wall parking. The robot must autonomously follow a line around a track for two complete laps. Timing starts when the robot passes through a trigger mark for the first time and ends when it passes through for the third time. After completing two laps, the robot must autonomously switch to parking mode and position itself 8 inches away from a wall (measured from the frontmost point of the robot).

Two different tracks will be used on the competition day: Field A, an easy track known in advance, and Field C, a harder competition track only revealed on the day of the competition.

The scoring system awards points for successfully completing two laps (up to 50 points on Field C). There are bonuses for completing laps under time thresholds (up to 20 additional points for Field C in under 30 seconds), parking precision (up to 25 points for zone 1), and first attempts (5 points).

Extra credit is also available for teams with the fastest times overall, best first attempts, and for teams whose robots can follow the track while traveling backwards.

1.2 Hardware Specifications

The robot used in this project is a "Skitter Classroom Robot" kit from AndyMark, Inc. The competition requirements state that robots should be assembled as originally intended and use only official robot kit parts. A photo of the Skitter robot used is shown in **Fig 1.1**.

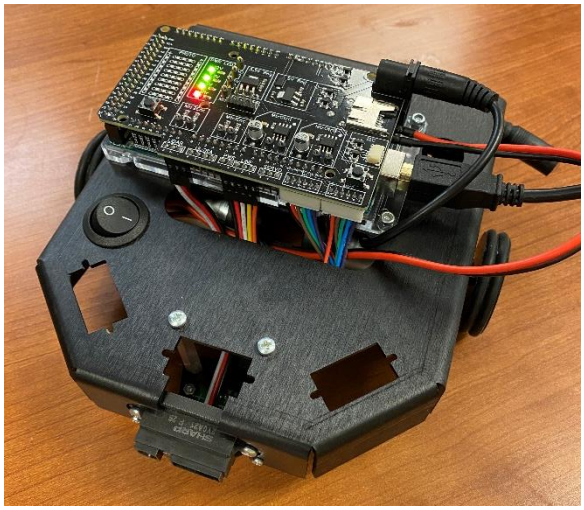

Fig 1.1. Skitter robot

The robot is comprised of several major electro-mechanical components that allow it to complete the tasks of the competition. These components are tabulated below.

Table 1.1. Robot Components

Component	**Subsystem**
Frame	Chassis
Ball Caster	Chassis

Wheel	Chassis
DC Gearmotor + encoder	Chassis
IR Distance Sensor	Input Sensors
Line Follower Sensor	Input Sensors
Arduino Mega	Controller
Skitter board shield	Controller
12V Li-Ion Battery	Controller

The first subsystem of the robot is its chassis, which consists of the aluminum frame, ball caster, 3D printed wheels, and motors. The chassis measures 6.5" (width) x 7" (length) x 3" (height). It is powered by two 12V gearmotors with a no load torque of 300 g-cm.

Attached to the aluminum frame are the two input sensors. The IR distance sensor has an effective range of 10 – 80 cm and outputs a single analog voltage 0.3 – 3.1 V. The line following sensor has an optimal sensing distance of 3mm and outputs 3 analog voltages 0 – 5 V.

Finally, the brains of the robot are the Arduino Mega microcontroller and skitter board shield. Both are powered via the 12V battery, and are connected to each of the sensors and motors. The Arduino Mega has an impressive 54 digital I/O and 16 analog I/O pins, and runs on a clock speed of 16 MHz. The skitter board shield fits on top of the Mega to regulate the power supplies and connect the JST ports to the I/O pins.

2 SUBSYSTEM CHARACTERIZATION AND FULL SYSTEM MODEL

2.1 Encoder Calibration

The primary purpose of calibrating the two wheel encoders is to determine how many encoder ticks is equivalent to the amount of centimeters traveled by the robot down the track. In order to determine such formula, the robot was rolled forward in a straight line a known distance for 10 trials. By using the Encoder Duo block from the ME 375 Robot Project library, the total number of encoder counts was found for each known distance. The distance vs encoder count plots for each motor are shown in **Fig 2.1.1** and **Fig 2.1.2** below.

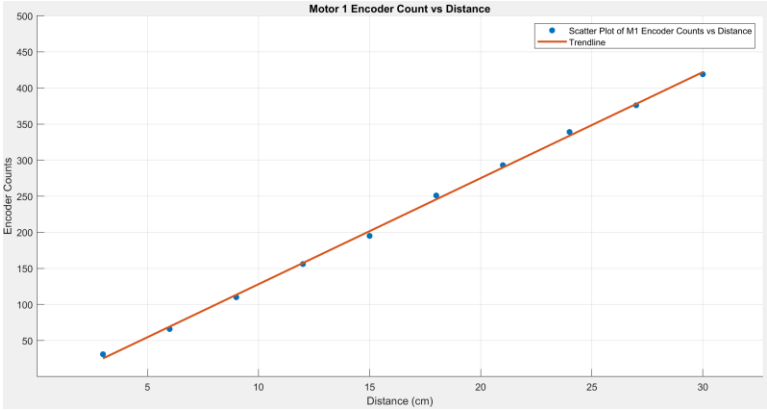

Fig 2.1.1 Motor 1 Encoder Counts vs Distance

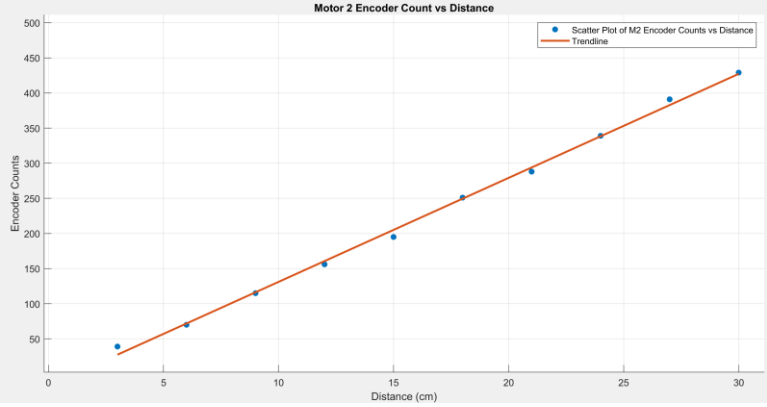

Fig 2.1.2 Motor 1 Encoder Counts vs Distance

The trendline equations for each figure were used to validate the calibration formula. To validate the calibration formula, the robot was rolled a known distance and the predicted encoder counts from the trendline equation was compared against the actual encoder counts measured by the robot. For reference, the trendline equation for motor 1 is $14.695x - 18.867$, and $14.806x - 17$ for motor 2. The values for each motor are shown below in **Table 2.1.1** and **Table 2.1.2**

Table 2.1.1 Validation of Motor 2 Encoder Calibration

Distance (cm)	Predicted Encoder Counts	Actual Encoder Counts
35	495.458	495
40	568.933	573
45	642.408	639
50	715.883	716
55	789.358	772

Table 2.1.2 Validation of Motor 2 Encoder Calibration

Distance (cm)	Predicted Encoder Counts	Actual Encoder Counts
35	501.21	506
40	575.24	584
45	649.27	644
50	723.3	702
55	797.33	800

2.2 IR Sensor Calibration

The primary purpose for calibrating the IR Sensor is to build a formula that converts the voltage measured by the IR sensor, to a distance in centimeters a given object is from the IR sensor. To build this formula, first a meter stick was placed perpendicular to the IR sensor, with the front end of the IR sensor aligned with the zero centimeter mark. A flat end of a cardboard box was used to trigger the IR sensor. The box was set at various distances away from the IR sensor, and the IR sensor's output voltage was recorded. The IR sensor distance vs output voltage is shown in **Fig 2.2.1** below:

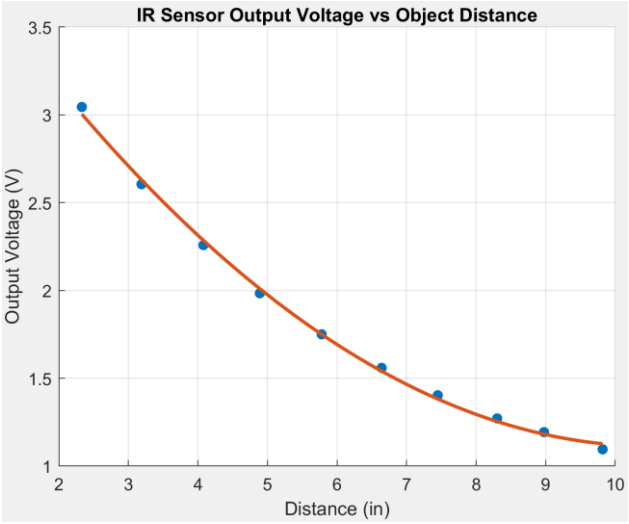

Fig 2.2.1 IR Sensor Output Voltage vs Object Distance from IR Sensor

The trendline equation from **Fig 2.2.1** was used to validate the calibration formula. To validate the calibration formula, an object was placed a known distance away from the IR sensor, and the predicted voltage output from the trendline equation was compared against the actual voltage output measured by the robot. For reference, the trendline equation from the IR sensor calibration is $3.9076e^{-0.136x}$

The are shown below in **Table 2.2.1**

Table 2.2.1 Validation of IR Sensor

Object Distance (in)	Predicted Voltage Output	Actual Voltage Output
13	0.667	0.6794
16	0.4435	0.4965
19	0.295	0.4307
22	0.196	0.2219
25	0.13	0.1586

2.3 Line Following Sensor Calibration

The goal of calibrating the Line Following sensor is to determine how far off the robot is from the center of the track line based on the binary outputs of the left, right, and center line following sensors, and the continuous position reading. To calibrate the sensor, a coordinate system was mapped based on the thickness of the track, with the center of the track being 0 inches. Left of the track indicated a negative value, while right of the track indicated a positive value. Binary sensor and continuous position values were recorded against robot position for the right and left lines, by increasing increments of 0.0625in. The edges and center of the spacing area region were also recorded. The values are shown below in **Tables 2.3.1, 2.3.2, and 2.3.3**.

Table 2.3.1 Binary Sensor and Continuous Position Outputs for Right Line

Distance from Center (in)	Binary 1	Binary 2	Binary 3	Continuous Position
0.035	1	0	0	1
0.0975	1	1	0	0.94
0.23	1	1	0	0.77
0.3275	1	1	0	0.74
0.4	1	1	0	0.44
0.5225	0	1	1	0.001

Table 2.3.2 Binary Sensor and Continuous Position Outputs for Right Line

Distance from Center (in)	Binary 1	Binary 2	Binary 3	Continuous Position
-0.035	0	0	1	-1
-0.0975	0	1	1	-0.93
-0.23	0	1	1	-0.86
-0.3275	0	1	1	-0.69
-0.4	0	1	1	-0.47
-0.5225	0	1	0	-0.01

Table 2.3.3 Binary Sensor and Continuous Position Outputs for Spacing Region

Distance from Center (in)	Binary 1	Binary 2	Binary 3	Continuous Position
-0.03125	0	0	0	-0.01
0.0	0	0	0	0.000268
0.03125	0	0	0	0.001

Given the thickness of the right and left lines and the spacing area, and based upon the results above, the flow chart, shown in **Fig 2.3.1**, maps the output of the three line following sensors and the continuous position reading to the position of the robot relative to the center of the line:

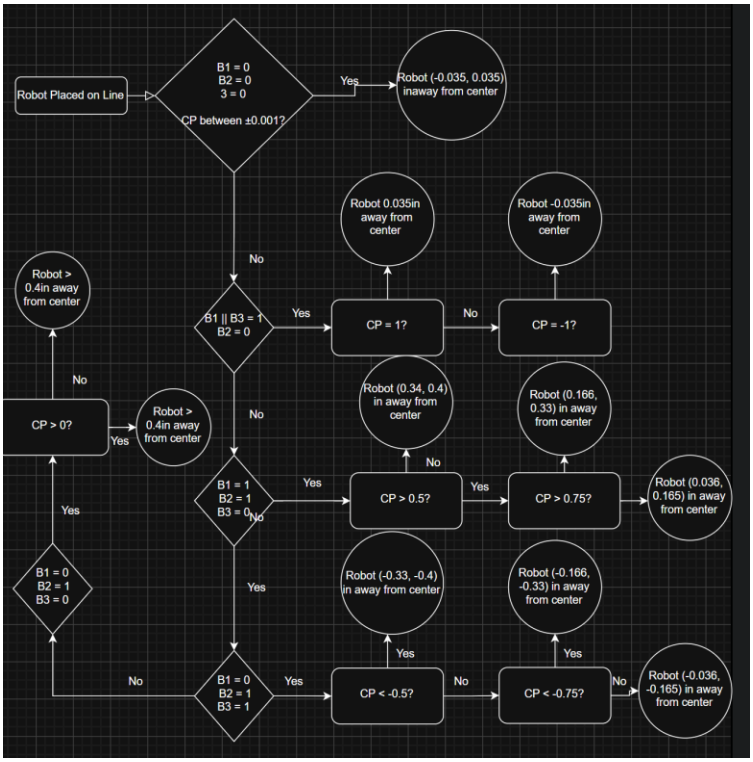

Fig 2.3.1 Binary Sensor Value to Robot Position From Center

To validate the line following sensor calibration, the robot was moved from the center of the line at known distances, and the **Fig 2.3.1** was followed to see if the position was predicted correctly. The validation is shown in **Table 2.3.4** below:

Table 2.3.4 Validation of Line Following Sensor Calibration

Actual Distance From Center (in)	**Predicted Distance From Center (in)**
-0.375	(-0.34, -0.4)
-0.25	(-0.166, -0.33)
-0.125	(-0.036, -0.165)
-0.015625	(-0.035125, 0.0351250)
0.015625	(-0.035125, 0.0351250)
0.125	(0.036, 0.165)
0.25	(0.166, 0.33)
0.375	(0.34, 0.4)

2.4 Motor-Gearbox-Wheel Subsystem Transfer Function

To determine the transfer function that models the motor-gearbox-wheel subsystem, the robot was assumed to behave like a first-order system of the form $\frac{K}{\tau s+1}$. By applying a square wave with input PWM signals, the resulting wheel speed was able to be observed. A square wave input with

an amplitude of 1, period of 1 second, pulse width of 50%, and phase delay of 0 seconds were applied to each motor individually using a Pulse Generator block in Simulink. The step response of the speeds for motor 1 and motor 2 are shown below in **Fig 2.4.1**.

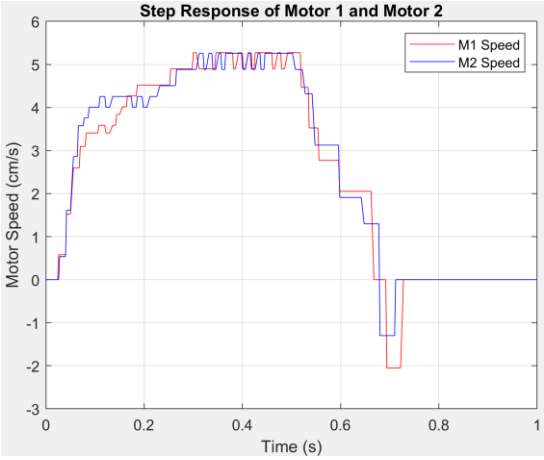

Fig 2.4.1 Step Response Data for Motor 1 and Motor 2

From these responses, the static gain K and time constant tau were estimated using each motor's output speed due to the step response from the pulse generator. The static gain found using the average steady state speed reached by each motor. For motor 1 and motor 2 those values were 5.08 and 5.07 in/s per unit PWM, respectively. The time constant for motors 1 and 2 were found to be 0.057 and 0.039 seconds, respectively. The resulting transfer functions for each motor are as follows. Motor 1: $\frac{5.08}{0.057s+1}$ and Motor 2: $\frac{5.07}{0.039s+1}$. To validate each transfer function, the Laplace transform of the transfer function was taken to solve for velocity and then integrated to get the position of the robot as a function of time. The actual time it took the robot to travel a known distance was compared the predicted time it took from the transfer function. The results are shown in **Table 2.4.1** and **Table 2.4.2** for both motors.

Table 2.4.1 Validation of Motor 1 Transfer Function

PWM Value	**Distance Robot Traveled (in)**	**Predicted Duration (s)**	**Actual Duration (s)**
0.5	10	1.296	1.04
0.5	20	2.535	2.09
0.5	30	3.774	3.13

Table 2.4.2 Validation of Motor 1 Transfer Function

PWM Value	**Distance Robot Traveled (in)**	**Predicted Duration (s)**	**Actual Duration (s)**
0.5	10	1.280	1.04
0.5	20	2.520	2.09
0.5	30	3.761	3.13

2.5 Friction Identification

While calculating the motor-gearbox-wheel transfer function, two nonlinearities were observed, both in the form of a deadband, where for a certain range of input values the motor speed's output remained the same. Motor one had zero output speed for PWM duty cycles below 0.27, and motor two below 0.25. This can most likely be explained due to static and dynamic friction experienced by the robot's wheels, which prevent motion until the motors are run at enough power to overcome the frictional forces. Additionally, saturation was observed at PWM values above 0.95, where no additional increase in amplitude resulted in an increase in output speed.

2.6 Full System Model

The full system model was created by utilizing the motor-gearbox-wheel transfer function models for the left and right motors of the robot, with an input of the PWM Duty Cycle, and then relating the velocity of each wheel to the kinematics of the robot. The PWM input passes through a function block acting as a deadband, where the output is set to zero if the PWM input is below the deadband threshold. The deadband function is shown in **Fig 2.6.2.** The resulting PWM input is then bounded between 1 and -1 with a Saturation block, before passing through the motor-gearbox-wheel transfer functions of each motor. The transfer functions output the wheel speed for both wheels, which were used to find the robot's position and orientation. To start, the robots angular speed can be modeled as the difference between wheel speeds 1 and 2, multiplied by the inverse of the width of the robot. The robot's angle was found by integrating the angular speed. The robot's x-position was found by adding both wheel speeds together and the multiplying that product by half of the cosine of the robots angle. Similarly, the robot's y-position was found by adding both wheel speeds together and the multiplying that product by half of the sine of the robots angle. The resulting Full System Simulink Model can be seen in **Fig 2.6.1**.

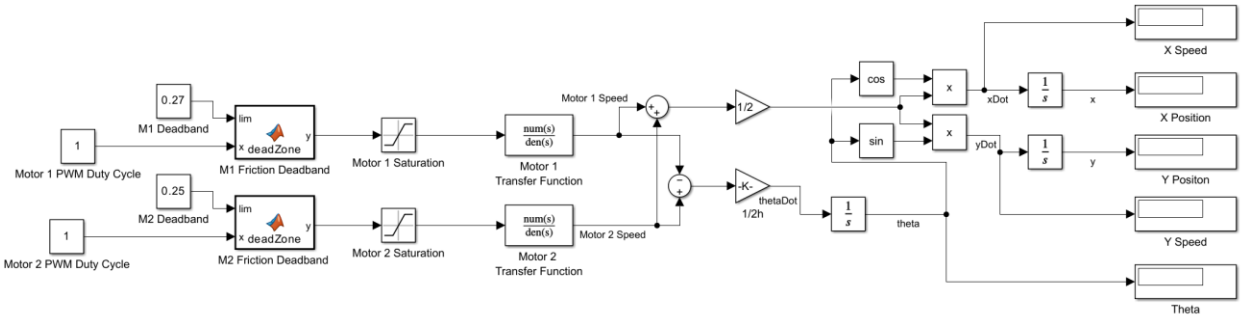

Fig 2.6.1 Full System Model

FullProgram1 ▸ M1 Friction Deadband

```
function y = deadZone(lim, x)
    % Set output to zero if input is outside [-d, d]
    if abs(x) > lim
        y = x;
    else
        y = 0;
    end
end
```

Fig 2.6.2 Deadband Function Block Used for Both Motors

3 CONTROLLER STRUCTURE AND DESIGN

3.1 State Machine

At the highest level, we propose for the robot to operate as a simple state machine with three distinct modes: line following, parking, and complete. The transitional conditions between each of these states aligns with the requirements of the competition: completing two laps around a track before aligning a distance from a wall and halting. The high level state transition diagram is shown below in **Fig 3.1**.

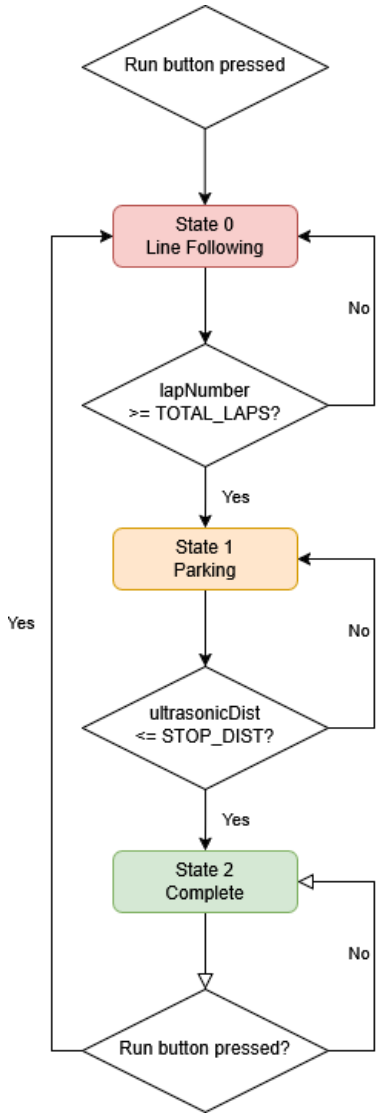

Fig 3.1. State Transition Diagram

When the robot is first powered on, all output is paused until the run button is pressed, as illustrated at the top of the state transition diagram. After the button press is detected, the lap number is initialized to zero and the robot enters State 0: Line Following. While state 0 is active, a feedback loop goes into effect to achieve line following, which will be further discussed below.

Additionally in State 0, a continuous loop is monitoring sensor data to count the number of trigger marks (laps) that the robot has completed. This is the conditional that will allow the machine to move to State 1: parking.

In State 1, line following continues but with an additional speed restriction feedback loop. Based on readings from the IR sensor, the robot continues moving forward until it reaches a predetermined distance from the wall. This condition move the machine to state 2: complete.

In state 2, all motor movement is stopped and the competition is considered complete. Despite this, the state is not terminal and can pass back to state 0 via another button press to reset the robot. This is beneficial for testing and allows multiple runs without re-running the Simulink code.

3.2 Control Structure Schematic

To dive deeper into the details how we plan to structure our robot's controller, we will present a schematic and explain the key features of its implementation. It is convenient to show the primary flow of the control structure as a Simulink block diagram, as illustrated in **Fig. 3.2**. In this schematic, you can see that the controller is divided into four main components: sensor input, state machine, line-following controller, and motor output (which contains both the encoder blocks and motor blocks).

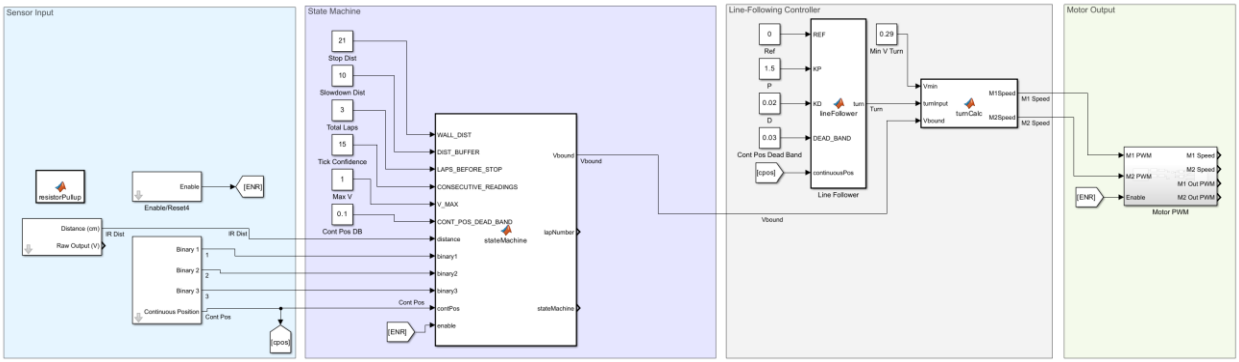

Fig 3.2. Control Structure Schematic

The sensor input area in light blue illustrates the data collection from our two primary sensors: IR distance and line following light sensor. This raw data is immediately fed into the second area, the state machine.

The state machine's logic has already been discussed previously, but now it is apparent how the logic fits into the overall control flow, shown in the purple area. The stateMachine function takes all sensor data (plus a multitude of configurable constants) and makes decisions on what the robot should do next. It then outputs a single value: a velocity boundary for the line-follower.

This upper boundary for velocity is then fed into the third area in grey: the line-following controller. The details of this feedback system will be discussed below, but the important fact to note is that the component converts the velocity input from the state machine to raw motor output values.

This is then fed to the fourth and final component of the control structure, motor output. The motor output from the line-following controller is clamped between 1 and -1 by saturation blocks, and dead-banded before being sent to the motors as PWM. A detailed view of the "Motor PWM" subsystem is presented below in **Fig 3.3**, showing both the motor and encoder blocks.

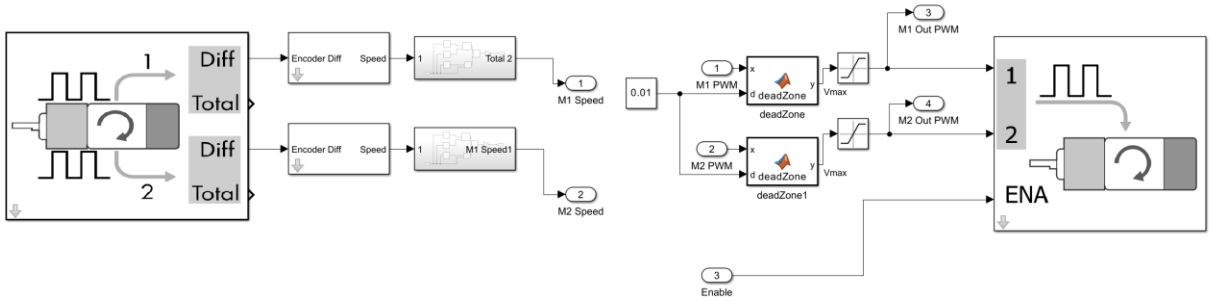

Fig 3.3. Motor PWM block internals

Please note that the control structure presented above is idealized and does not include a variety of output scopes, debug displays, and additional user inputs for enabling each motor separately and testing turn radius. The full Simulink controller for the robot will be submitted separately for your consideration.

3.3 Feedback Loops

The proposed controller consists of two feedback loops. The primary loop is a line-following algorithm active during both States 0 and 1. This was selected to be a PD controller due to the importance of maintaining the centerline over long straightaways (Proportional) and tight curves with high rates of error (Derivative). The controller will take an input from the continuous position line sensor in the range [-1, 1] where 0 indicates the robot is perfectly centered. It will output another unitless constant in the range [-1, 1] which is referred to as "turn". This turn value will then be transformed into two PWM outputs, one for each motor. Based on estimates from the sensor calibration and system identification, we expect to have a K_p of approximately 1.5 and a K_d of approximately 0.02.

Additionally, during State 1 a second feedback loop comes into effect. This will be a proportional controller that scales the robot's maximum velocity as it approaches the wall. The P controller will take an input from the IR sensor in units of cm, and output a maximum velocity in the range [0, 1]. When the robot is the desired distance from the wall, the maximum velocity output will be zero. While the choice of K_p for this controller is fairly arbitrary and only determines how smoothly the robot will brake, a starting value of 0.1 has been chosen.

It is also important to note that there is no feedback loop for motor speed in this design. For several of the previous labs, a controller was developed to precisely tune the linear velocity of each motor via feedback from the encoders. However, this controller was not perfect and introduced slight delays and oscillations into the system. For line following, where fast response is prioritized over accurate speeds, it was decided to bypass the speed controller and directly drive the motors via PWM.

3.4 Configurable Parameters

One last feature of note in the controller design is the large quantity of configurable constants being input into the state machine and line-following controller. These determine the system's

behavior and can be dynamically modified to tune its performance. Table 1 below shows each of these parameters and their purpose.

Table 3.1. Configurable Parameter List

Parameter	Description	Nominal Value
Stop Dist	Distance from the wall at which the robot will stop completely after reaching the final lap	21 cm
Slowdown Dist	Distance added to Stop Dist at which the robot will begin proportionally slowing as it reaches the wall	10 cm.
Total Laps	Number of trigger marks to count before changing from State 0 to State 1	3 (1 passing start line, then 2 laps)
Tick Confidence	Number of consecutive trigger mark readings that must be met before incrementing the lap count	15
Max V	Unitless number [0, 1] that defines the maximum PWM the controller will send to the motors	1
Cont Pos DB	Part of the trigger detection condition that prevents a mark from being detected if the line sensor's continuous position reading is above this value	0.1
Ref	Desired continuous position value, an offset from center of the line [-1, 1]	0
P	Line following proportional gain	1.5
D	Line following derivative gain	0.02
Cont Pos Dead Band	Continuous position region where the controller will not attempt to make jittery corrections	0.03
Min V Turn	Unitless number [0, 1] that defines the minimum PWM the controller will send to the motors. This determines the minimum possible turn radius.	0.3

4 Potential Problems

While much consideration has been given to the proper functionality of the robot on the competition track, there are several problems that may arise during implementation and testing. One such issue is the calibration of event detections. The constants for trigger mark recording explained above are fairly arbitrarily chosen and may not perform perfectly in the real world. The same is true regarding the K_p for wall detection slowdown. It is expected that these values will need to be tweaked through trial and error to consistently detect the events as desired without false positives.

Along a similar vein, the idealized full system model explained above is likely imperfect and does not match the precise real-world dynamics. While it is anticipated that the estimated values for K_p and K_d for the line follower will work, they will likely need fine-tuning for optimal track navigation at speed. There are a number of considerations, like the tradeoff between more precise slow speeds and sketchier high speeds, that are not easily modeled and must be experimentally determined. We aim to achieve the points for lap completion under 38 seconds, which will require fiddling with the values for optimal performance.

A final concern is the differences between test tracks and competition tracks. There are a number of variables that will affect the performance of the robot, including light level (needed for reliable line detection), track cleanliness (dirt and particles affect line detection and wheel traction), and the uncertain layout of Field C, which is not revealed until the day of the competition. To overcome these problems, we will test the robot in as many varying conditions as possible, and utilize the practice tracks both forward and reverse to provide as much variety of experimental data as possible.