Title: This project SUCKS (get it because it's a vacuum ;)

Topic of Interest: Robot Vacuum

Abstract: We will be studying the localization and path following algorithms of a Roomba. We chose this because many of the group members are interested in robotics and a roomba is a simple yet insightful example of robot control. We plan to investigate the Kalman filter, which is a feedback control system that uses sensor data to continuously update its believed position. We will also investigate the roomba's path following control, including both its forward velocity and heading angle controller. Lastly, we will take a look at the control algorithm behind the spinning dust collection system. We plan to study the state space model, block diagrams, transfer functions, inputs and outputs, and feedback control law of each of these smaller controllers that make up the roomba.

Students/Roles:

Student	**Task/Role**	**Portfolio**
Natalie	Model the localization algorithm of a Roomba using an Extended Kalman Filter. Studied the Kalman filter to better understand the EKF by creating a block diagram of the closed loop system. Simulated the EKF in MATLAB to visualize its accuracy.	https://cornell-mae-ug.github.io/spring-2025-portfolio-nataliekaplan/projects/2025-systems-roomba-analysis/
Alex	Investigate the speed control of the brushes on the roomba. Create a block diagram to describe the control system.	https://cornell-mae-ug.github.io/fa25-portfolio-akw76/projects/Roomba-Brush-Speed-Control-Analysis/
Jordan	Investigate the heading angle control of the roomba, creating a block diagram, transfer function, and MATLAB simulation. A main takeaway of this work is that the roomba's velocity and heading angle are deeply connected since both of them use the velocity of the wheels as inputs to the controller.	https://cornell-mae-ug.github.io/spring-2025-portfolio-jvogel457/projects/2025-%20Roomba%20Controls/
Javier	Explore the velocity control of the Roomba, including the state space models. Create a MATLAB script that simulates the step response of the motor to reach the target angular velocity.	https://javiermajumdar.github.io/portfolio-fa25/projects/Velocity_Control_(Roomba%20Analysis)/

List of MAE 3260 concepts or skills used in this group work

- TFs
- State space
- Block diagrams
- Open-loop system
- Active control
- Feedback control law
- Command following
- Disturbance rejection

STUDENT A- NATALIE - LOCALIZATION

In order to clean a house, a Roomba must maintain an idea of where it is. To do this, it uses its control inputs combined with sensor measurements and a dynamics model. However, there are many possible sources of error, including but not limited to wheel slippage, sensor noise, mechanical imperfections, and dynamic obstacles. With sources of error in the sensor, dynamics, and control, how does the Roomba know which to trust, particularly when they disagree?

This introduces the localization problem. Given an initial guess of the robot's pose (position and orientation in the global coordinate frame) bel(x_0), a model of robot dynamics p(x_t | x_{t-1}, u_t), a measurement function p(z_t | x_t, m), control inputs $u_{1...t}$, and measurements $z_{1...t}$ (where x=position, z=sensor measurements, m=map, u=control inputs), we must find the belief of the current pose bel(x_t). Since there is so much uncertainty, the robot's pose is modeled as a probability instead of just one value. By using the control inputs, sensor measurements, and dynamics model together, we can reduce the uncertainty in the Roomba's pose to much lower than it would have been if we had only used one source of data. [1]

Since iRobot does not publicly disclose which localization filter the Roomba uses, I will model it using the Extended Kalman filter. However, in order to study the Extended Kalman filter, we must first know how the Kalman filter works.

The Kalman filter introduces restrictive assumptions about the dynamics and measurement models to make the Bayes filter implementation plausible and give an exact solution to the robot's pose instead of an estimation. We assume that the dynamics and measurement models are linear with Gaussian zero-mean noise and that the initial belief is Gaussian. From these assumptions, we know that the belief for any time t must be Gaussian. [1]

The initial distribution is described below:

$bel(x_0) \sim N(\mu_0, \varepsilon_0)$ State $x_t \in R^n$ Measurement $z_t \in R^k$ Control $u_t \in R^m$

As mentioned above, the Kalman filter works under the assumption that the dynamics model is linear. The equation is shown below:

$$x_t = Ax_{t-1} + Bu_t + \varepsilon,$$

[1] where A is an nxn matrix and describes how the state evolves from t-1 to t without controls or noise, B is an nxm matrix and describes how the control u_t changes the state from t-1 to t, and $\varepsilon \sim N(0, R)$ is the process noise and is represented as a random variable. The process noise is assumed to be independent and normally distributed with covariance R [5]. For more details about the dynamics model, see the velocity control section of this report.

The Kalman filter also assumes that the measurement model is linear. The equation is shown below:

$$z_t = Cx_t + \delta,$$

[1] where C is a kxn matrix that describes how to map the state x_t to an observation z_t and $\delta \sim N(0, Q)$ is the measurement noise and is represented as a random variable. The measurement noise is assumed to be independent and normally distributed with covariance Q [5].

The Kalman filter cycles between prediction and correction steps to continuously update the robot's pose. The prediction step, or the dynamics update, updates the pose using the

dynamics model. This step uses two equations: one to find the mean $\overline{\mu}_t$, and one to find the standard deviation $\overline{\Sigma}_t$ of the Gaussian describing the pose. The bars represent the prediction step [1]. The equations are shown below:

$$\overline{\mu}_t = A\mu_{t-1} + Bu_t \qquad \overline{\Sigma}_t = A\Sigma_{t-1}A^T + R$$

The correction step, or measurement update, uses the measurements from the sensors to adjust our prediction of the pose of the robot. The equations to find the mean and standard deviation are shown below:

$$\mu_t = \overline{\mu}_t + K_t(z_t - C\overline{\mu}_t) \qquad \Sigma_t = (I - K_t C)\,\overline{\Sigma}_t$$

The term $(z_t - C\overline{\mu}_t)$ is the discrepancy between the actual and predicted measurements and its weight is controlled by the Kalman gain K_t, which describes how much to trust the sensor measurements. As mentioned above, neither the dynamics model nor the measurement model is free of error, so the Kalman gain is used to balance them. If the sensor is very noisy, K_t will be close to 0 [1]. The equation is shown below:

$$K_t = \overline{\Sigma}_t C^T (C\overline{\Sigma}_t^T C^T + Q)^{-1}$$

A block diagram of the closed loop system for calculating the mean position is shown in Figure 1. This diagram illustrates the loop between the prediction and correction stages of the filter.

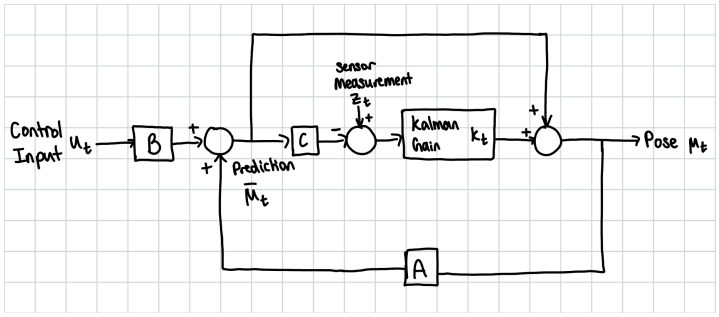


Figure 1. Block diagram of Kalman filter

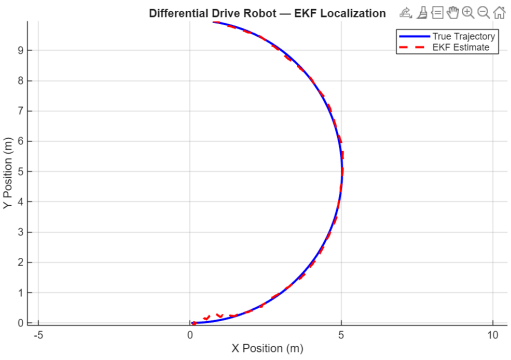


Figure 2. True trajectory vs. EKF estimate of robot pose

We cannot use the Kalman filter on this system since its dynamics model is not linear. However, we can use the Extended Kalman filter, which relaxes the linearity assumption of the Kalman filter by using Jacobians to linearize the dynamics model locally at each time step. Although we are moving from an exact solution to an approximation, the relaxation of this assumption makes the EKF applicable to many more systems, including this one.

The plot in Figure 2 shows the true trajectory compared to the EKF estimate of the pose of a differential drive robot driving with a forward velocity of 0.5 m/s and an angular velocity of 0.1 m/s using a noisy GPS-like sensor model. The plot shows that the EKF does a very good job of estimating the pose of the robot. The filter quickly hones in on the true pose and stays very accurate. The small errors are due to the linearization of a circular path. In reality, at every time step there may be a different control input and GPS cannot be used indoors.

STUDENT B- ALEX

A roomba's main function is to clean up messes, so arguably its most important component is the speed control of its vacuum brushes. How does it spin its brushes at the right speed to pick up messes? How does it adapt to different amounts of dirt it picks up in different locations? Actually, a Roomba's brush speed control system is open-loop, meaning there is no feedback control. It does not read the current brush speed and doesn't use any feedback controller.

Instead, it uses two sensors to set the brush speed depending on the conditions it is vacuuming in, according to [6]. A piezoelectric sensor on the front end of the bottom of the vacuum detects dirt levels. When bits of dirt, dust, or other debris hit the sensor, the Roomba receives tiny electric impulses and it knows to ramp up its speed. The other sensor in this system is an optical texture sensor. This sensor shines IR light onto the ground, and this helps it detect the surface type. If the Roomba is on carpet, it activates carpet boost to work harder to clean the mess.

When the microcontroller in the Roomba receives data from the sensors, it uses Pulse-Width Modulation (PWM) to vary the voltage input to the motor for the brushes. This is because microcontrollers can only send a fixed voltage or 0 volts to the motor, so it uses a square wave signal with different lengths of 0V to send different voltages [7]. The exact PWM wavelengths used are dependent on the duty cycle, of which there are three, corresponding to three different voltages and power levels. The highest duty cycle is the carpet boost, when the most voltage is being sent to the motor. The lower duty cycle is the eco cleanup mode, with the least amount of voltage being sent to the motor. Here is an equation for voltage sent by the MCU to the motor drive as a function of time. $V_{battery}$ is the fixed voltage that can be sent, and T is the period of the square wave signal. dT is a length of time that changes depending on the duty cycle being used. Figure 3 is a Desmos graph for this function.

$$V(t) = \{0 \leq t < dT: V_{battery}, dT \leq t < T: 0\}$$

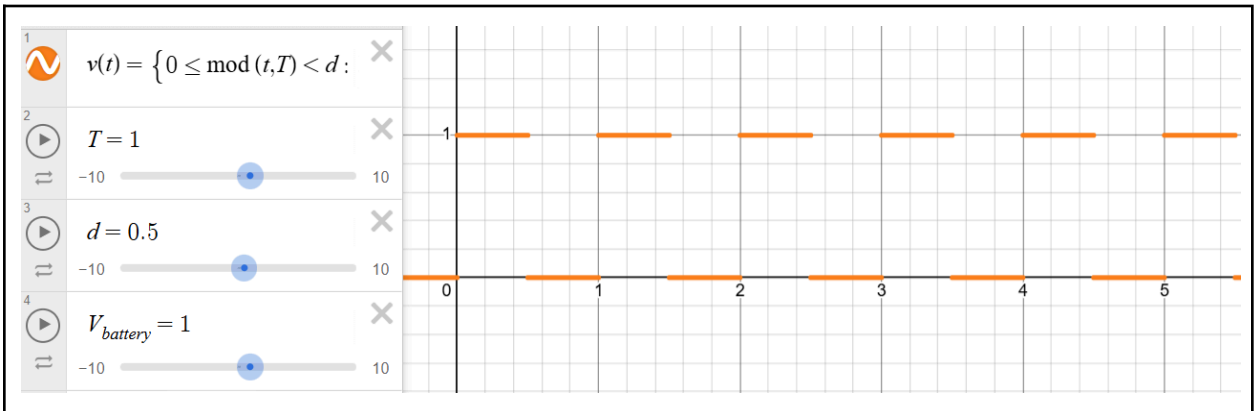


Figure 3: Desmos graph of the square wave function from the PWM to the motor drive.

It's also important to note that while there are two brushes in the cleanup system, they are both controlled by the same motor. The first brush loosens, lifts, and pulls debris under the Roomba, while the second brush accelerates the debris into the vacuum's suction channel. The two brushes rotate at the same RPM, but in opposite directions. The speed the motor spins at is slowed by a gearbox to appropriate speeds for the brushes. Unfortunately, data telling us the exact RPM of the motor and brushes during each duty cycle is not publicly available online for us to find.

One other factor in motor speeds is if there is a jam. If, for some reason, the brushes become obstructed and can no longer rotate, back-emf from the motor stop and this causes a current impulse. If that impulse is detected it tells the microcontroller unit to cut PWM to 0% and stop the cleaning cycle. An error message plays out loud, and a human must clean the brushes before the Roomba can resume cleaning.

Figure 4 is a block diagram that summarizes the brush speed control system that has been discussed up to this point.

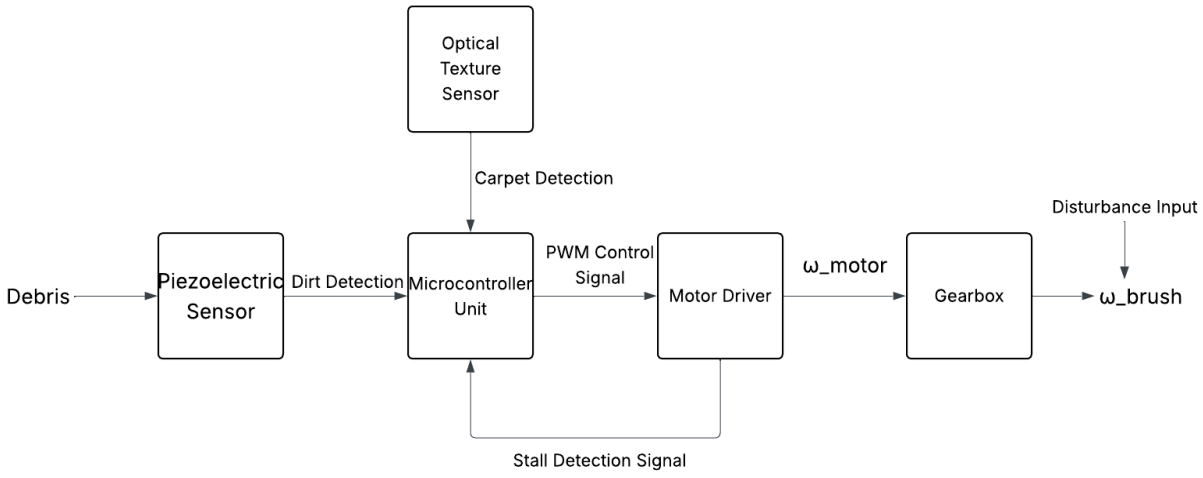


Figure 4: Block Diagram of Roomba Brush Speed Control System

STUDENT C- JORDAN- HEADING ANGLE CONTROL

The “Roomba” style robot is very interesting since it is extremely mobile. While cars and bicycles can only move along paths defined by the steering radius, a roomba has the ability to rotate in place, giving it much more freedom in its path planning. Thinking about the use case, this makes a lot of sense, since roombas are typically deployed in small home environments, where there is not sufficient room to turn with such a large radius.

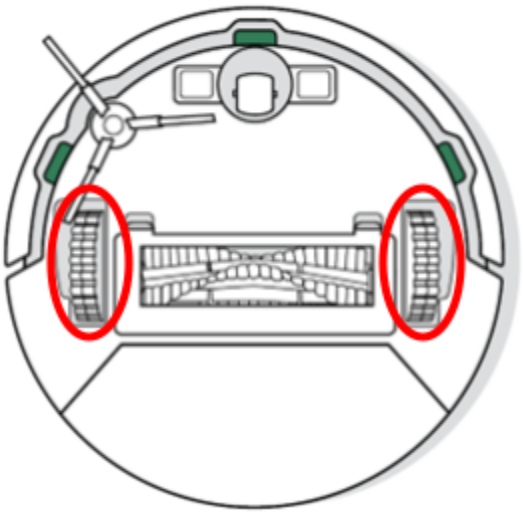


Figure 5
Source: Adapted from [4]

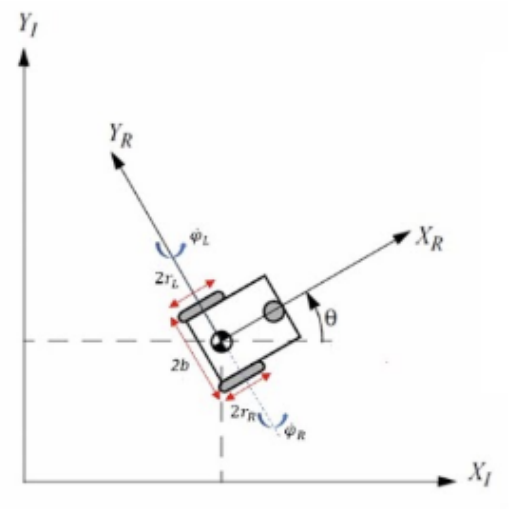


Figure 6
Source: Adapted from [3]

The magic behind this spin-in-place mechanism is the differential drive system. As seen in Figure 5, the robot has 2 driven wheels. By rotating them in opposite directions, the roomba is able to spin in place, adjusting its heading angle directly. By rotating the wheels in the same direction at different speeds, the robot can robot and translate simultaneously, opening up the door for smooth paths that are more optimal. We can model the roomba with the following coordinate frames and variables shown in Figure 6.

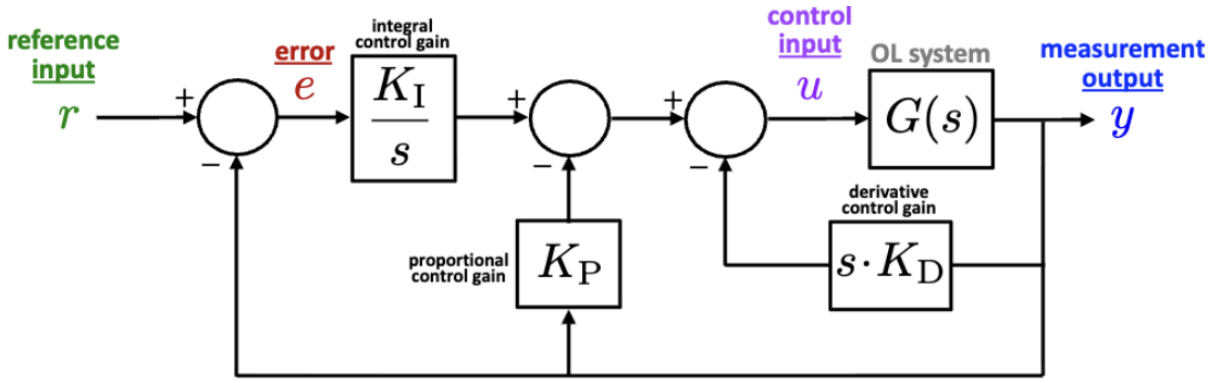
For our analysis of the heading angle controller, the most important equation is

$$\frac{d\theta}{dt} = \frac{r}{2b}(\omega_R - \omega_L)$$

where r is the radius of the wheels, 2b is the base diameter of the roomba, and ω_R and ω_L are the angular velocities of the right and left wheels respectively. For convenience, we will create a variable u(t)= $\omega_R - \omega_L$ for the control input. Taking the Laplace transform of this equation and solving for the transfer function we get

$$G(s) = \frac{\Theta}{U} = \frac{r}{2b} \cdot \frac{1}{s}$$

, which fully encapsulates the dynamics of this system. Next, we need to address the controller. The roomba uses a PID controller to decrease the response time of the system and ensure there is zero steady state error. The block diagram might look something like this.



The only small difference is that this block diagram is completely isolated from the localization step discussed earlier in the report. It assumes that the measurement output is completely accurate, which in reality, it is not. In a more accurate version of this block diagram, the measured output would lead into the localization block diagram. The output of that block diagram, a more correct estimate of the pose of the robot, would then be fed back into this diagram to compute the error.

To better understand this system, take a look at this matlab animation.

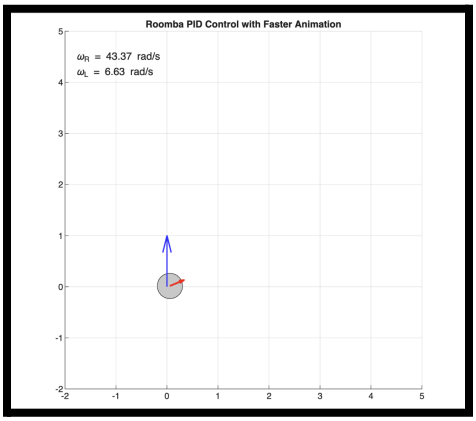


Figure 7

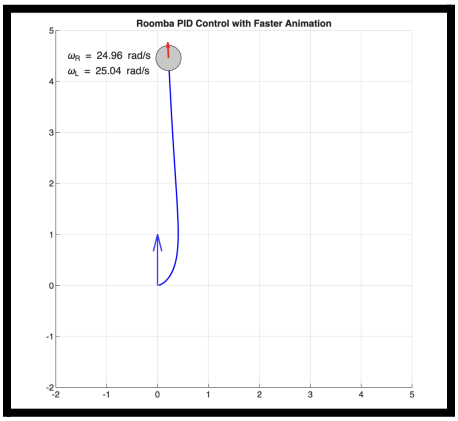


Figure 8

In Figure 7, the roomba is very far away from its target angle. As a result, the right wheel velocity is much higher, which will cause the robot to rotate to the left. In Figure 8, the roomba is pretty much aligned with the desired heading angle, so the velocities are very similar. This is the proportional part of the controller in action! Additionally, take note of how the robot first overshoots the target angle and then comes back– that proves that this is a higher order system. The I and D parts of the controller each add a pole. Lastly, all of this happens while the roomba is in motion, so no time is wasted. So cool!

STUDENT D- JAVIER

Velocity control is central to how a Roomba moves and responds to its surroundings. The Roomba adjusts its wheel speeds based on sensor inputs, which may include dirt detection sensors, obstacle sensors, cliff sensors, and surface hardness detection sensors [6]. These adjustments are controlled through a PID controller that manages the motor speed in RPM. Understanding the velocity control system provides insight into how the Roomba converts sensor inputs to motion through the wheels.

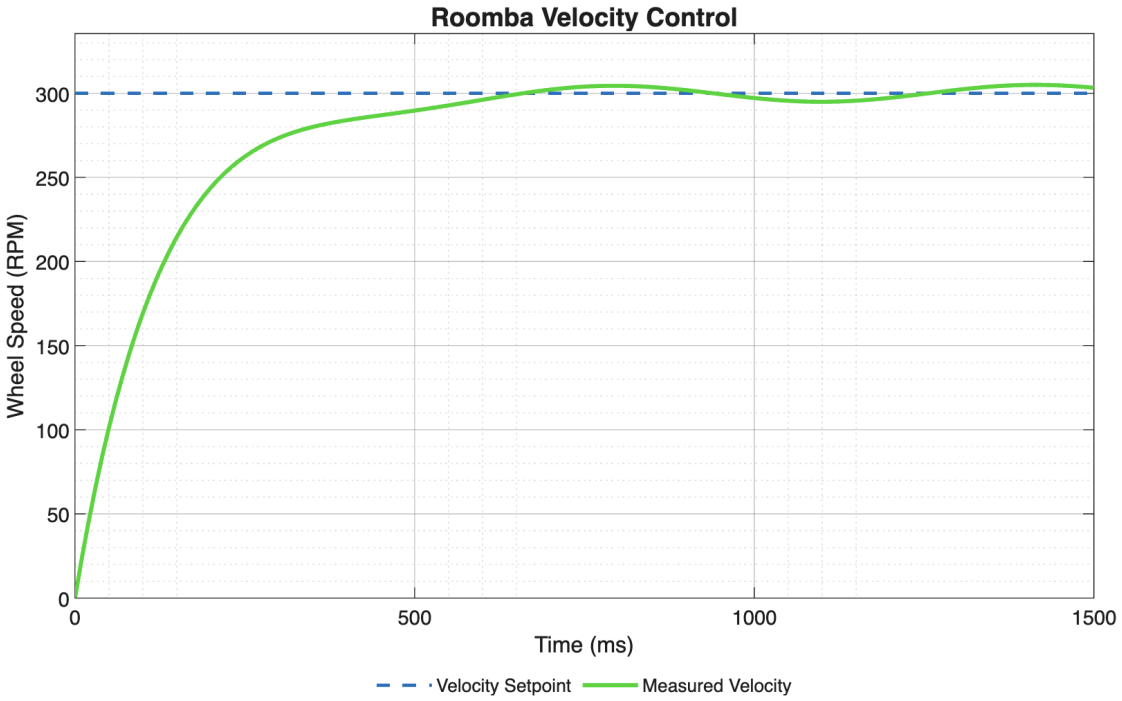


Figure 9

To visualize the behavior of a closed-loop velocity controller, I created a MATLAB script that simulates the velocity response of a motor similar to that of a Roomba. *Figure 9* shows the step response to a velocity setpoint of 300 RPM. The model I used to get the measured velocity curve was made using a simplified model of DC motor dynamics. The equation I used to model this was:

$$\omega(t) = 300(1 - e^{-0.008t}) + 5\,sin(0.01t)$$

The term with the exponential represents the first-order increase in speed due to the dynamics and inertia of the system, and the sinusoidal term represents the effects of closed-loop behavior due to disturbances, friction, or controller tuning.

With the line of the setpoint plotted next to the system response, performance metrics may be evaluated, including the rise time, settling time, overshoot, and steady state error. These metrics are important for tuning the velocity controllers for the Roomba.

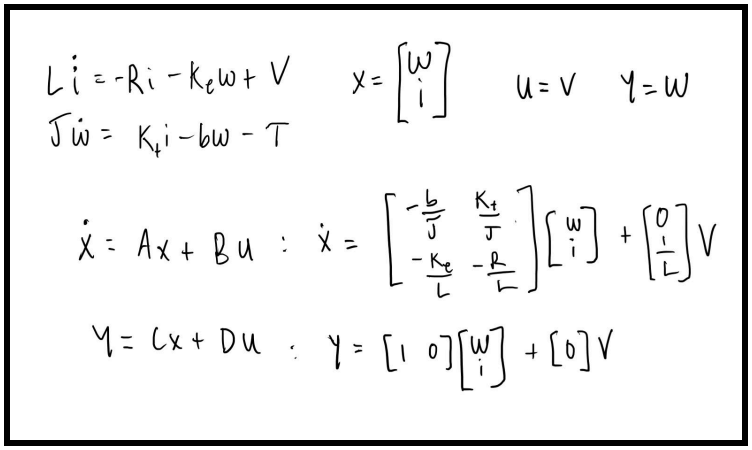


Figure 10: Motor Dynamics State Space Model

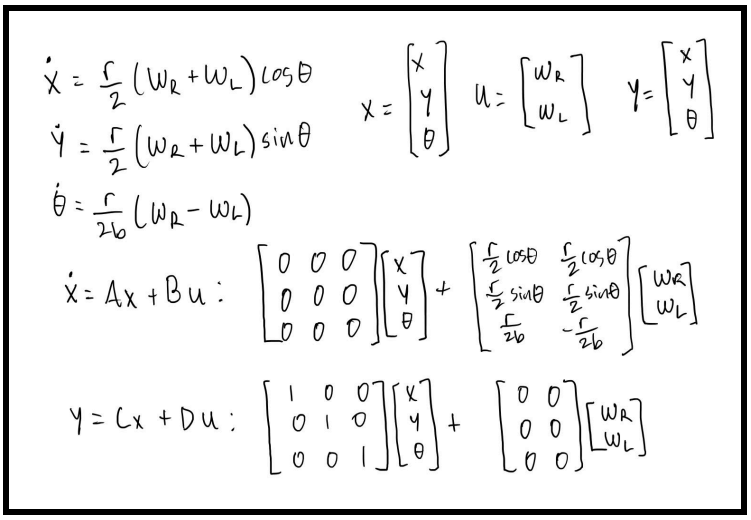


Figure 11: Kinematic State Space Model

These figures illustrate the different state space models used to describe the velocity-controlled motion of the Roomba.

Figure 10 shows the state space model for the motor dynamics of the Roomba. This model describes how the motor voltage delivers the wheel velocity. The state variables include the motor angular velocity and the motor current. The input of this model is the applied motor voltage, while the output is measured by the wheel angular velocity. The state space model was formed using both electrical and mechanical dynamics of the system using the torque-current relationship, the back-emf relationship, and other friction/resistance terms.

Figure 11 shows the kinematic state space model that describes how the wheel velocities control the Roomba's motion. The state variables include the x and y position on the floor and theta, the angle at which it turns. Inputs include the wheel angular velocities for both the left and right wheels coming from the motor velocity controllers. The kinematic equations used to describe this system include the forward velocity, angular velocity, and position, which are used to form the state space model as shown above.

Both of the state space models demonstrate the complete motion of the velocity control in the Roomba. The motor voltage creates angular velocity in the wheels, and the kinematic model shows how the wheel speeds translate into motion. The models are separated since the motor dynamics occur much faster than the motion of the Roomba. Furthermore, it allows the velocity controller to regulate the wheel speed independently of the position.

References

[1] A. Bizyaeva. MAE 4180. Class Lecture, Topic: “Kalman Filter.” College of Engineering, Cornell University, Ithaca, NY, Feb. 11, 2025

[2] A. Bizyaeva. MAE 4180. Class Lecture, Topic: “Extended Kalman Filter.” College of Engineering, Cornell University, Ithaca, NY, Feb. 20, 2025

[3] Engineering Educator Academy, **“Kinematics of Differential Drive Robots and Odometry,”** *YouTube*, Jul. 31, 2021. [Online]. Available: https://www.youtube.com/watch?v=RZIZcDxQ8P4. [Accessed: Dec. 5, 2025].

[4] iRobot, “Article 525 — Find Answers,” *iRobot Home Support*, n.d. [Online]. Available: https://homesupport.irobot.com/s/article/525. [Accessed: Dec. 5, 2025].

[5] T. Bhattacharjee, P. Culbertson. MAE 4760. Class Lecture, Topic: “Kalman Filters.” College of Engineering, Cornell University, Ithaca, NY, Oct. 16, 2025

[6] The Zebra, “How Roomba Works: Sensors, Navigation, and Cleaning Technology Explained,” The Zebra, n.d. [Online]. Available: https://www.thezebra.com/resources/home/how-roomba-works/#:~:text=While%20we%20use%20our%20eyes,as%20stairs%20or%20a%20balcon.[Accessed: Dec. 9, 2025].

[7] Rohde and Schwarz, “Understanding Pulse Width Modulation,” YouTube, Feb 26, 2024. [Online]. Available: https://www.youtube.com/watch?v=nXFoVSN3u-E. [Accessed: Dec 10, 2025]