Engs 26 Final Project Report
Duck Car No. 11

Jake Markus & Jonathan Sumaili

Table of Contents

1. Introduction

The aim of this project is to design a controller for a "Duck car" that always stops at a predetermined distance away from an obstacle that is in front of its infrared sensor, with little to no oscillations. To achieve this goal, we used a modular approach in designing the system, which involved breaking it into multiple blocks and characterizing each block independently. We then analyzed the open-loop system using simulation tools, such as MATLAB's *sisotool*, in order to find a suitable controller that would make the closed-loop system meet our preset specifications. We implemented the theoretical compensator circuit, analyzed the system's response, and made the required changes whenever the system did not meet our design requirements. At the end, we compared and contrasted the predicted model versus the real-life system and outlined potential changes we could make to enhance our system's performance and robustness.

2. System Modeling

We developed the duck car system model through system identification. We split the system into three distinct blocks: the sensor, the plant (motor), and the controller. The system uses an infrared sensor to determine the physical distance to the obstacle and converts it into a voltage signal. In the closed-loop case, this sensor output is fed back into the system and used to recalculate the error from the desired output. The controller takes in the error voltage and outputs a voltage that drives the motor. As an actuator, the motor converts the controller output voltage into velocity that moves the car to the desired position. The block diagram of the system is shown below:

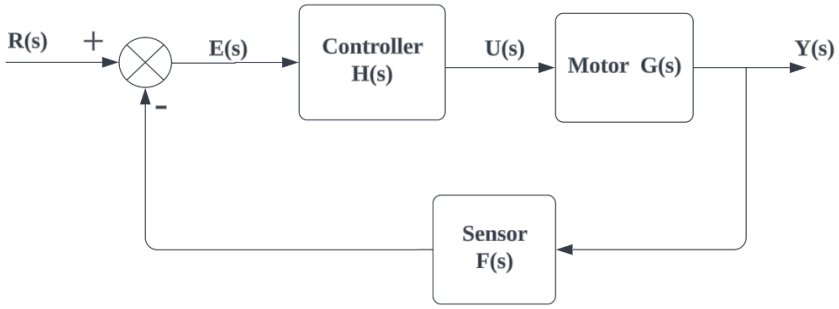


Fig 1: Block diagram of the system

1.1 Sensor Characterization

In order to find the transfer function for the sensor, we measured the voltage produced by the sensor with a foam block placed in front of it at distances ranging from 5 cm to 100 cm. We then plotted the two values which produced a curve with a decaying exponential trend. Selecting an operating point of 17 cm so that the car could be easily controlled, but still have a reasonable buffer range. We obtained the exponential curve of best fit from the measured values and linearized the system at 17 cm, which translated a reference voltage of $1.45\ V$. The slope of the line at this point, corresponding to the gain of the sensor, was $-\ 6.08\ V/m$ i.e.

$$K_S = -6.08\,V/m$$

The negative sign in the gain denotes the fact that smaller distances correspond to higher voltage produced by the sensor, and vice versa.

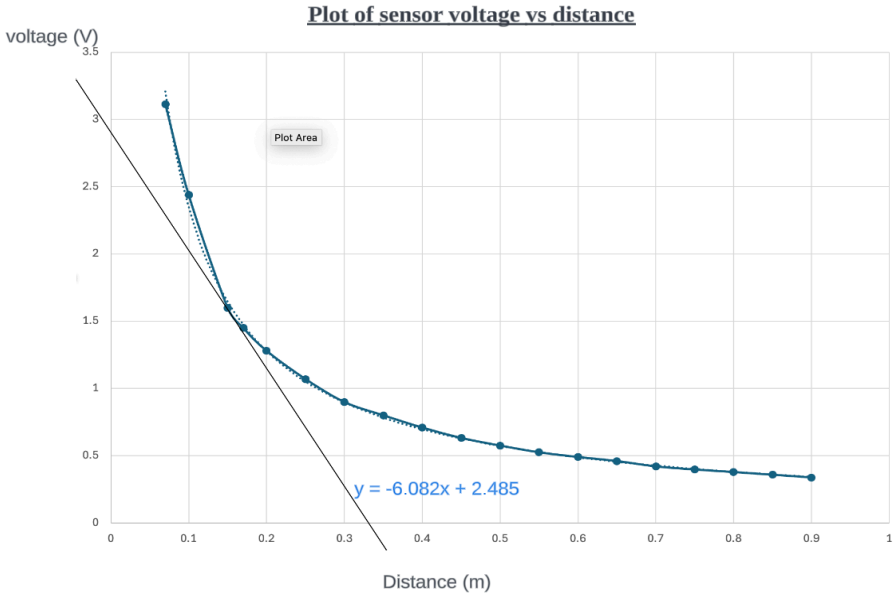


Fig 2: Linearization of sensor data and finding sensor gain value

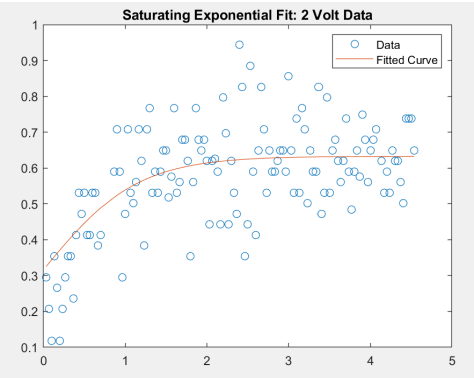
- Sensor modeling data: https://www.desmos.com/calculator/x04wbf0md9

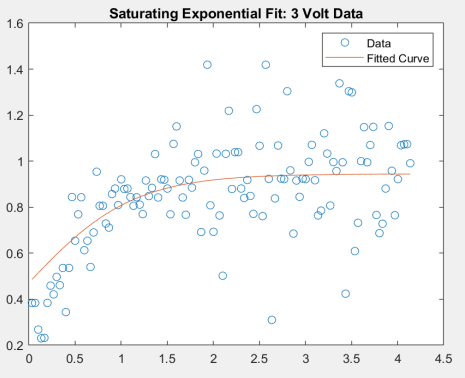
Physically, this means that the car should always stop 17 cm away from an obstacle placed in front of it, in ideal conditions. However, the reference voltage varies depending on the battery, so this can vary in practice.

1.2 Motor (*Plant*) Characterization

The motor converts electrical energy into mechanical energy that makes the duck car move. In order to characterize this relationship, we recorded the car's motion for 3 step-input voltage values ($2V$, $3V$ & $4V$) as an open-loop system and obtained the velocity data using Tracker. Due to the irregular distribution of the points, we analyzed the velocity vs time data collected using MATLAB and derived the best curve of fit for each step input as demonstrated in the plots below:

- Motor modeling data: https://www.desmos.com/calculator/jgfluocm8w





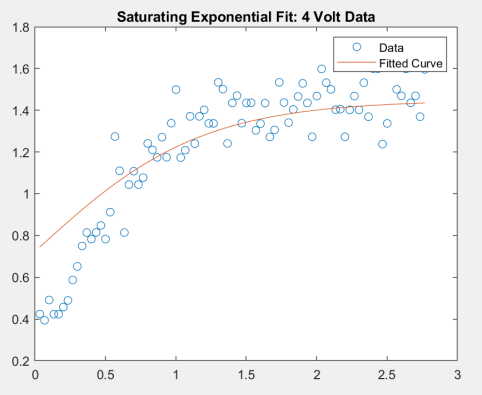


FIG 3: The velocity of the TS11 car as a response to three step inputs (2V, 3V, 4V)

The general equation of the exponential curve is:

$$v_x(t) = A(1 - e^{-Bt})$$

where $\frac{A}{V_{in}} = K_m$ is the average motor gain and $B = \frac{1}{\tau}$ gives the time constant.

The resulting transfer function from the motor's input (volage) to its output (velocity) is

$$G(s) = \frac{K_m}{\tau s+1}$$

The table below shows how we determined K_m and τ values:

V_{in} (Volts)	**A** (m/s)	**K_m= A/V_{in}** (m/s/V)	**B** (1/sec)	**Tau = 1/B** (sec)
2	0.6329	0.3164	1.7406	0.5745
3	0.9445	0.3149	1.7753	0.5633
4	1.4479	0.3618	1.6969	0.5893
Avg. K_m		0.3311	**Avg. Tau**	0.5757

FIG 4: Averaging trials to obtain a first order model of the plant

Since we were aiming at controlling the car's distance and not velocity, we obtained the distance output of the motor by integrating its velocity. This results in the transfer function:

$$G(s) = \frac{K_m}{s(\tau s+1)} = \frac{0.3311}{s(0.5757s+1)}$$

1.3 Block Diagram

From the transfer functions obtained above, we can model the uncompensated duck car system with the following block diagram:

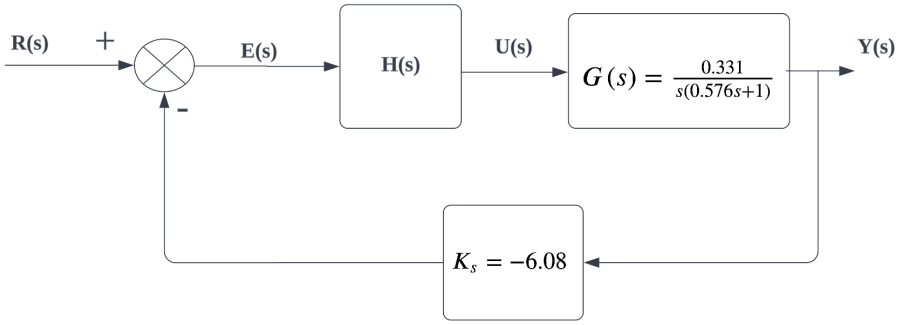


Fig 5: Block Diagram of the with motor and sensor transfer functions

3. Stability Analysis of Uncompensated System

To gauge the system's stability, we used MATLAB's *sisotool* and analyzed the root locus, step response and frequency response as shown in the figure 6.

Analyzing the root locus plot, we deduced that the system is stable without a controller (or with a unit gain proportional controller because $H(s) = 1$. We confirmed this experimentally by connecting the motor input to the output of the summing junction. This inherent stability means that the system is theoretically stable no matter the gain: an infinite gain margin! The Phase margin of 48.9 degrees is also a safe threshold to survive sensor delays and other real world impracticalities.

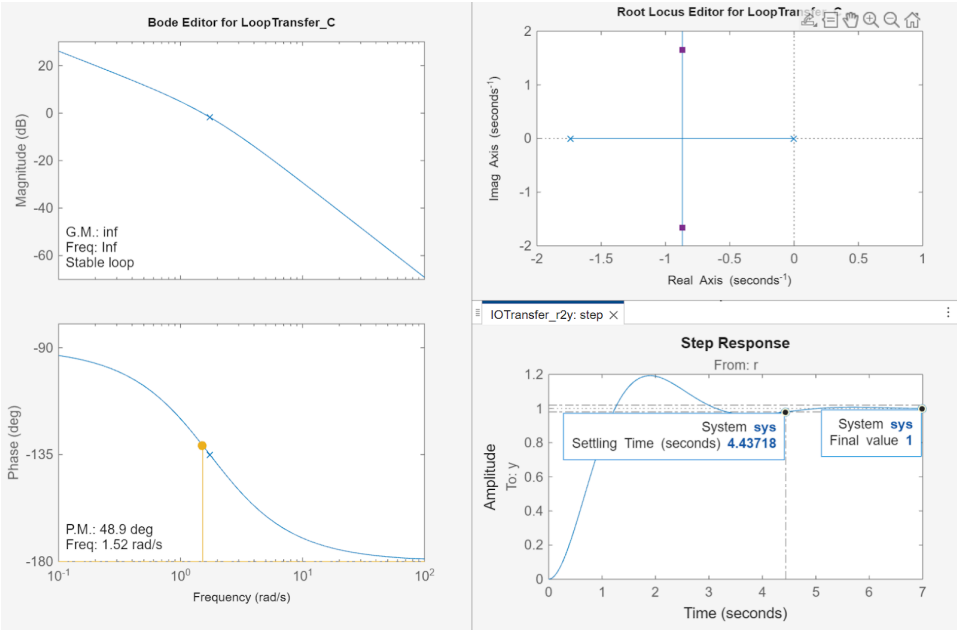


Fig 6: Bode plot, root locus diagram, and step response for our modeled uncompensated system

However, this setup's performance has several areas to be improved. Firstly, the step response is well over 4 $seconds$ which would translate to a slow response to a target. Secondly, the system has an undesirable overshoot of about 20% which means that the car will always go beyond (or below, depending on the direction) the 17 cm mark before settling. This also means that if the car approaches a target too fast it might crash into it before pulling back.

Moreover, a controller proportional gain of one is not sufficient to start the car from rest even at maximum error. Due to motor stiction and rolling friction, the car moves very slowly in practice, even when nudged. Simply implementing a proportional controller would make the car much more practical but would not meet performance requirements such as quicker response time, less overshoot and reduced oscillations. There's a better alternative. We can greatly improve on the system's settling time and maximum overshoot specifications using a Proportional Derivative controller, which will make the car respond promptly to a target, and stop quickly and smoothly as it approaches a distance of 17cm from the object with little to no overshoot.

4. Compensator Design and Simulation

Based on concerns expressed in the previous section, we decided on the following design specifications for a more robust system and better performance:

- A 2% settling time of less than 1 sec, $T_s < 1\ sec.$
- A steady state error specification of $e_{ss}(x) < 15\%$ to highlight consistency of the car's performance on a given surface.

- An overshoot value of less than 10%, $M_p < 0.1$, which would require a damping ratio of about $\zeta > 0.6$. This would translate to reduced oscillations as the car is stopping.
  We calculated the damping ratio as follows:

$$M_p = e^{-\pi\left(\frac{\zeta}{\sqrt{1-\zeta^2}}\right)} < 0.1$$

$$\zeta > \frac{-\left(\frac{ln|0.1|}{\pi}\right)}{\sqrt{\left(1+\left(\frac{ln|0.1|}{\pi}\right)^2\right)}}$$

$$\zeta > 0.6$$

  This specification will be difficult to measure and analyze in our physical system without the help of modeling tools. We mostly focused on response time (prompt/smooth action) and steady state error of the duck car. Modeling tools would not have represented imperfection in the real world discussed later in this report.

As noted before, the open-loop step response is already stable with a 20% overshoot and a 2% settling time of $4.44\ seconds$. To bring these values down to the specifications we set above, we settled for a PD controller and modeled the compensated system's response.

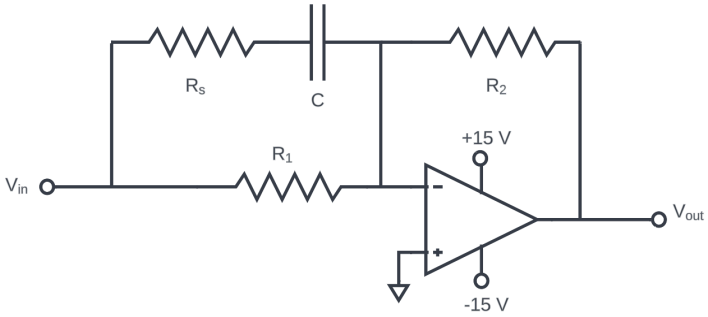


Fig 7: PD controller circuit diagram

The general circuit of a PD controller implemented using an operational amplifier (figure 7) is related to a transfer function from V_{in} to V_{out} with one zero, one pole and a proportional gain:

$$\frac{V_{OUT}}{V_{IN}} = \frac{R_2}{R_1} \cdot \frac{\left(s + \frac{1}{R_1 \cdot C}\right)}{\left(s + \frac{1}{R_s \cdot C}\right)} = K_c \cdot \frac{(s + \omega_z)}{(s + \omega_p)}$$

where ω_z and ω_p are the zero and pole values respectively.

This means that, with an well chosen capacitor value, the remaining component values can be found using the formulae:

$$R_1 = \frac{1}{\omega_z \cdot C}$$

$$R_2 = K_p \cdot R_1$$

$$R_s = \frac{1}{\omega_p \cdot C}$$

Initially, we started with a small-valued pole at $s = -15$, a zero at $s = -2.05$ and a compensator gain $K_c = 3$. This produced the following controller transfer function:

$$\frac{V_{OUT}}{V_{IN}} = 3\frac{(s+2.05)}{(s+15)}$$

and component values:

$$C = 10\mu F,\ R_1 = 40\ k\Omega, R_2 = 100\ k\Omega,\ \text{and}\ R_s = 10k\Omega.$$

However, the PD controller implemented with these component values did not work. After further deliberation and extensive trial-and-error simulation, we changed the pole to $s = -290$, the zero to $s = -2.1$, and the gain to 2.8 resulting in the transfer function:

$$\frac{V_{OUT}}{V_{IN}} = 2.8\frac{(s+2.1)}{(s+290)}$$

This produced and a 2% overshoot and a settling time of $0.62\ seconds$ as shown in figure 7.

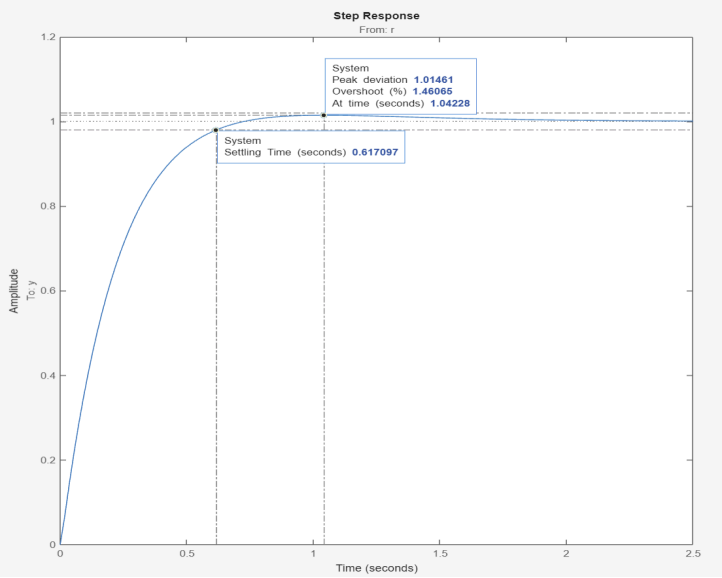


FIG 8: Time domain response of controlled system in simulation

Using the relationship between the transfer function and circuit elements, we estimated the component values that should produce this kind of step response:

$$C = 3.3\ \mu F,\ R_1 = 144.3\ k\Omega,\ R_2 = 404\ k\Omega,\ and\ R_S = 1\ k\Omega.$$

Figure 9 shows the theoretical PD circuit.

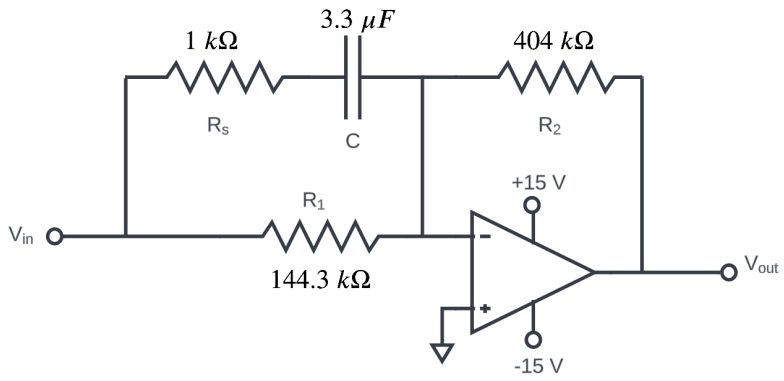


Fig. 9: Theoretical PD controller from the simulation

5. Experimental Compensator Adjustment

Although it provided a good starting point, this predicted controller did not work very well in practice. To account for the motor stiction, friction, non-linearities, and other real world imprecisions, we relied on engineering intuition and the system's system to adjust the controller.

With the calculated component values, the car's response was too aggressive and had a huge overshoot followed by oscillations. Additionally, we had to replace our fuses a couple times meaning that the PD circuit was drawing too much current. The cause for this was either the abnormally large ratio of R_1 to R_s or a huge gain leading to saturation.

First, we decreased R_1 to about $40\ k\Omega$ and consequently R_2 to about $110\ k\Omega$, ensuring that the gain remained constant. After doing this, the car's response became slower with reduced oscillations. Although it was a better response, it meant that we had to increase the overall gain of the system, because the car's response was always slow even while cruising. We changed the gain by decreasing R_1, since the D.C. gain of this inverting amplifier is $\frac{R_2}{R_1}$. This greatly improved the car's performance, however its response wasn't "snappy" enough.

Because the pole and zero were not far enough apart, the controller was mostly acting like a proportional controller with a single gain. Therefore, since $\omega_z = \frac{1}{R_1 \cdot C}$ and $\omega_p = \frac{1}{R_s \cdot C}$, we decreased R_s, and adjusted the absolute value of the DC gain ratio. This spread the pole and zero frequencies apart, allowing for more of the Bode plot to have a slope of 20 dB/decade. With the following changes implemented, the car responded much better. Further experimentation culminated in the PD circuit shown below:

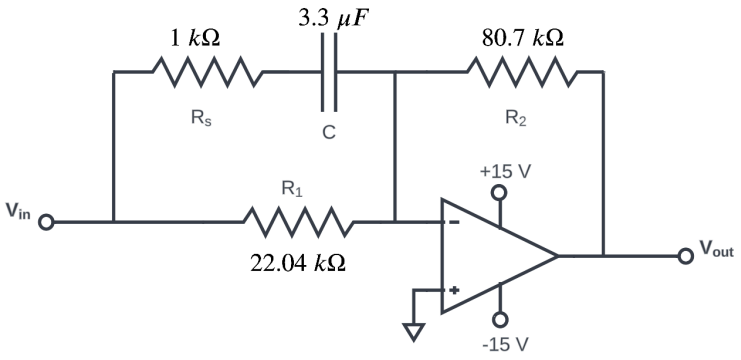
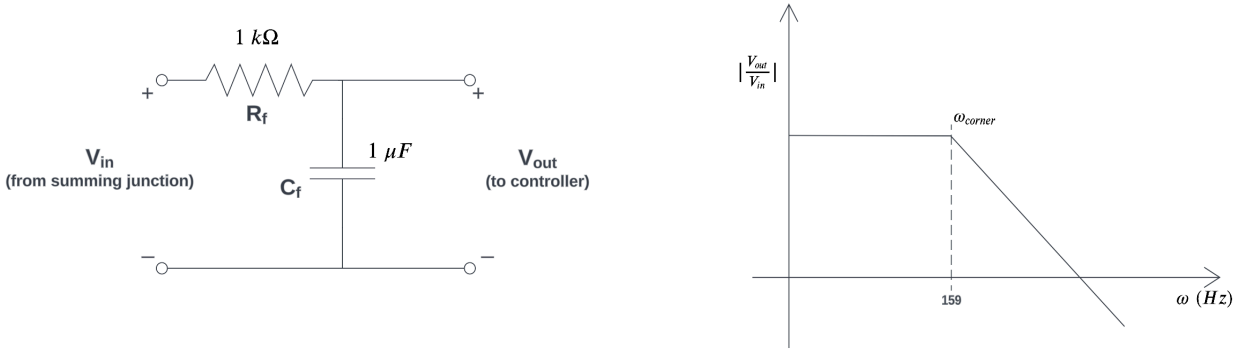


Fig 10: Final Implemented Compensator Circuit with measured component values

The transfer function of the circuit is:

$$H(s) := 3.66 \cdot \frac{(s+13.15)}{(s+303)}$$

With this implemented on the car, we noticed a hum from the motor, indicating a very high frequency controller output. Through oscilloscope measurements and Bob Barry's wisdom, we determined that the digital infrared sensor was updating the distance measurements discreetly, generating spikes in the differentiator at each step. To filter out this high frequency update, we implemented the following low pass filter to isolate true readings from the output of the summing junction:



(a) (b)

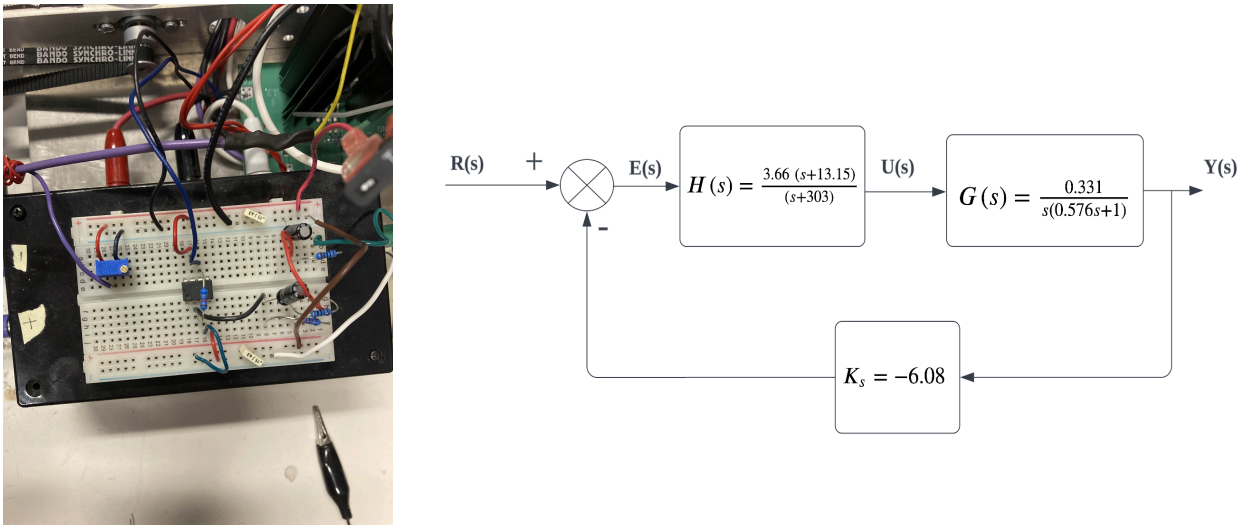
Fig. 11: Low pass filter on compensator input (a)-circuit diagram, (b)-magnitude vs frequency response showing cut-off (corner) frequency

The corner frequency was calculated using the formula:

$$\omega_{corner}(Hz) = \frac{1}{2\pi R_f C_f} = 159.2\ Hz$$

This addition to the circuit made the car operate silently, and presumably saved battery power as well. Because of the small time constant and negligible resistance, we established that this didn't affect performance negatively. With the low-pass filter in series with the PD controller, we were satisfied with

the precision, response, and motion of the car. We implemented our full circuit on the TS11's breadboard as shown in **figure 12** below. The reference voltage was adjusted by a potentiometer acting as a voltage divider.



(a) (b)

Fig. 12: Implemented circuit, (a)-breadboard connections, (b)- block diagram of the full system

6. Experimental Results and Evaluation

To test the steady state error of our car in practice, we held it at 2 meters and then released it towards a wall. By our design, our reference input is constant at 17 cm. The results were as follows:

Trial	Steady State distance (cm)	Error (cm)	Error (%)
1	18.5	1.5	8.82
2	18.2	1.2	7.06
3	18.6	1.6	9.41
4	18.8	1.8	10.59
5	18.5	1.5	8.82

Therefore, on average, we measured a steady state error of about 8.94% for this test, which is well within the specification of 10% error. Although our model predicted a steady state error of zero, it was an approximation that disregarded friction, motor stiction, and real world non-linearities. These factors also damped out the smaller oscillations in the response, meaning that the car only ever rolls back once, and

then stops immediately. Furthermore, our car performed well on a variety of floor surfaces, and was very easy to control with a foam block. It could be made to slightly hit the wall at full cruising speed, but partially this seems to be an inherent limitation on stopping time.

7. Conclusion

This project was a great opportunity to apply the abstract concepts we learned this quarter to a tangible product. We learned a great deal about system modeling . In addition the design process highlighted the value and limitations of system characterization. While our model of the uncontrolled system provided a valuable starting point to begin experimentation, it did not match closely enough with observed results to predict the exact component values of an ideal PD controller. Still, with a little engineering intuition and fiddling in the lab, we were able to produce a rewarding product.

8. Appendix

A. Sensor Characterization Data

x(m)	V(volt)
0.07	3.115
0.10	2.44
0.15	1.7
0.20	1.28
0.25	1.07
0.30	0.899
0.35	0.8
0.40	0.709
0.45	0.632
0.50	0.575
0.55	0.525
0.60	0.49
0.65	0.46
0.70	0.42
0.75	0.397
0.80	0.378
0.85	0.358
0.90	0.338

B. Motor Characterization Code (MATLAB)

```
%% LOADING 2 VOLT DATA
data = readtable('velocity_data_2volts_take2.txt');
x = data.Var1;  % time
y = data.Var2;  % velocity

%% LOADING 3 VOLT DATA
data = readtable('velocity_data_3volts_take2.txt');
x = data.Var1;
y = data.Var2;

%% Loading 4 VOLT DATA
data = readtable('velocity_data_4volts.txt');  % Or replace with your actual
filename
x = data.Var1;
y = data.Var2;
```

```
%% Finding the exponential curve of best fit
format short;
model_func = @(b, x) b(1) ./ (1 + exp(-b(2)*x));  % b(1) = A, b(2) = B
% Initial guess for parameters [A, B]
initial_guess = [max(y), 1, mean(x)];
% using lsqcurvefit
options = optimset('Display', 'off');
params = lsqcurvefit(model_func, initial_guess, x, y, [], [], options);
y_fit = model_func(params, x);

%% Plotting the original data and the fitted curve
figure;
plot(x, y, 'o', 'DisplayName', 'Data');  % Original data points
hold on;
plot(x, y_fit, '-', 'DisplayName', 'Fitted Curve');  % Fitted curve
hold off;
legend;
title('Saturating Exponential Fit using lsqcurvefit');
A = params(1);
B = params(2);
tau = 1/B;
```

C. References and Appreciation

Thank you to Bob Barry, Professor Phan, and TA Anthony for all the help and know-how!