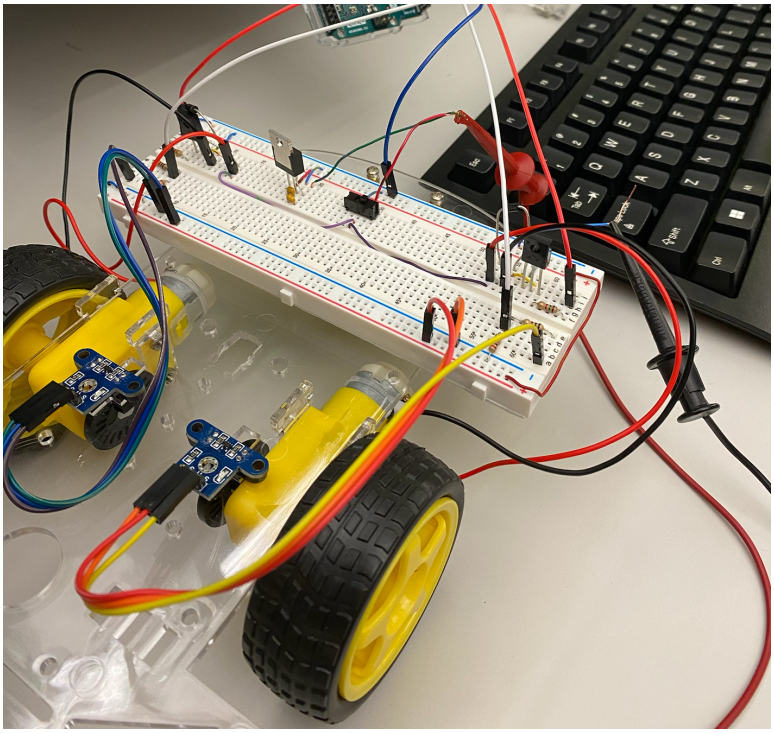
EECS16B Final Lab Report

Luis Cermeno Farro luis.cermeno@berkeley.edu
Nahom Ghebreselasie nahomsit7@berkeley.edu
Arduino ID : **74**
Resources: https://drive.google.com/file/d/1r4ekWYjXTwrhg6twjRF2NI7dZD8ZiSLP/view



1 Lab 6: System ID

Summary

In System ID Lab 6, first we fitted two encoder to measure our wheel velocity then, our main task involved establishing a relationship between the car's input and its output (changes in wheel velocity). To achieve this, we adopted a mechanical model defined by two parameters: θ, which measures the rate of velocity change per unit of voltage, and β, representing a consistent offset caused by any inherent system bias like friction or any natural bias. Initially, our selection of the PWM (Pulse Width Modulation) range for voltage input was wrong because our `data_coarse` was bunch of zero's which means we did not collect proper data, the PWM range of the collected data were not between 50 to 250, leading to unsuccessful data analysis. After collecting proper `data_coarse,` then we had a rough idea of our car's dynamics, then we collected `data_fine` . Finally, we performed least-squares regression on our data to determined θ and β values for both wheels.

Questions

1. What do θ and β represent physically, not mathematically? What are your values of θ and β , and do they reflect the car's performance while collecting data? Why/why not?

Physically, θ represents the measure of the wheel and motor's responsiveness or sensitivity to changes in the input duty cycle. It reflects how much the velocity of the wheel changes for a given change in the input signal . A higher θ value would indicate that a small change in the input causes a significant change in velocity, denoting high sensitivity. And β is a constant offset in the velocity of the wheel. This offset accounts for factors that cause the wheel to have a baseline velocity different from zero even when the input signal suggests otherwise. This could include aspects like static friction or other inherent biases in the system, which might cause the wheels to move differently than expected based on the input signal alone.

Values of θ and β

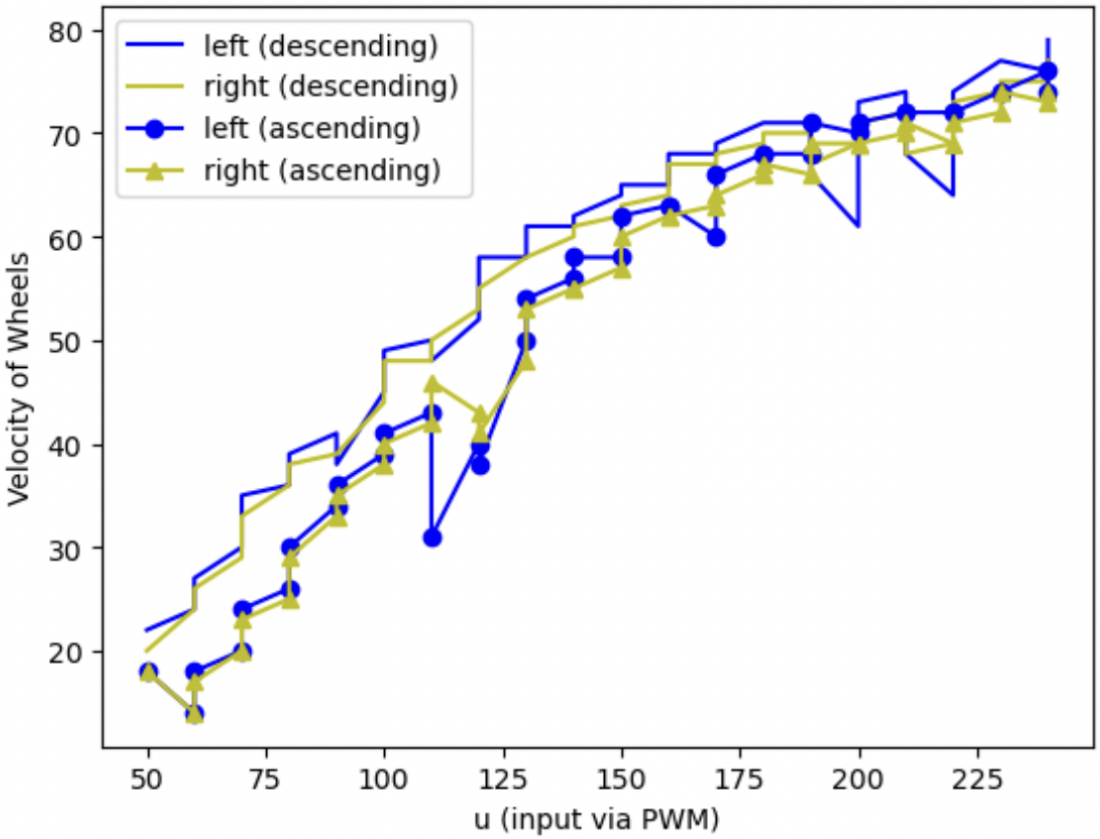
Float theta_left = 0.1618 ;
Float theta_right = 0.1726 ;
Float beta_left = -38.07 ;
Float beta_right = -35.98;

Values of θ and β don't reflect the car's performance while collecting data, because they may not effectively represent the car's performance due to its operation in an open-loop system. This system lacks feedback control, making the car's trajectory unpredictable and limiting the

usefulness of Theta and Beta in forecasting future performance based on current inputs and conditions.

2. How did you choose the PWM input range for fine data collection? If there were other data ranges that could have also conceivably been chosen, why did you choose this range over other ranges? Please include a graph of your car's coarse data to support your answers.

The PWM input values that we chose for the fine data collection ranged from 50 to 250. We selected this range because the data it produced was the most linear and optimal for obtaining comparable wheel velocities on our vehicle. We experimented and tested a range of values before deciding that these were ideal for our needs.



Coarse data Graph

3. To implement a higher order polynomial model, what would you need to change in the current lab flow to calculate the coefficients for this new model? Evaluate the benefits and drawbacks between using a higher order model vs. a linear model.

To implement a higher order polynomial model into the current lab setup, several adjustments would be necessary, especially in how we calculate the model parameters. The key change would involve adding more state variables. This addition would require expanding the matrix used in our analysis, introducing more columns to support the extra state variables. As a result, the column for Theta and Beta in the matrix would include additional terms to ensure a balance between the rows and columns as well.

Benefits :

- A higher order model often provides a more accurate representation for real-world systems.
- higher order polynomial models offer detailed accuracy and are suited for complex systems.

Drawbacks:

- Higher order models are significantly more complex than linear models, making them more challenging to calculate, analyze, and manage computationally.

4. As the batteries run low, assume that the corresponding velocities for each PWM input go down linearly. In other words, the velocity vs. PWM input curves for both wheels shift down by some value vo , where vo is non-negative and increases as the battery power gets lower. If we still aim for the same operating point velocity and use the same linear model, how will the car's performance change as the battery level decreases?

As battery power drops, it becomes more difficult to sustain the desired performance with the existing linear model. If this happens, we need to modify the PWM inputs and potentially tweak the model to accommodate the altered behavior of the system. We faced a similar situation in Lab 9 Integration. Our left battery was at 7 volts and the right at 8 volts. After replacing the batteries with new batteries, the car didn't drive straight like it used to, indicating a need to adjust our model to align with the new battery setup. And it took us a whole lab to figure that out.

2 Lab 7: Controls

Summary

In Controls Lab 7, our overall goal was to implement a control system that help SIXTEEN drive straight and turn. We used the model we obtained from SystemID to implement an open loop control, a closed loop control, and finally, a turning mechanism.

For the closed loop control, we first came up with the equations which we obtained via simple isolation of the input to our model (u).

We noticed that this control was inaccurate and did not account for disturbances of the environment. Naturally, the second step was implementing a feedback mechanism: a closed loop control.

For this, we came up with two equations for each wheel in which the feedback term was proportional to the difference in distance travelled by each wheel (delta) and a feedback factor (f). We learned that, the f factors (f_left and f_right) were related to the eigenvalue of our system, and we needed to chose it value wisely so that our system became stable. That is, converged to a value, which we learned had to be a constant for our delta so both wheels drive at the same velocity.

To find the accurate f_left and f_right factors, we coded `turning.ino` to test different f values. After we found the best f_left and f_right, we proceeded to implement turning, which was a simple functionality leverage of our closed loop control system.

At the end of the lab, our car was able to go straight far, straight short, turn left, and turn right.

1. What are the open loop equations of SIXT33N's control scheme for our PWM input, $u[i]$? What are some advantages and disadvantages of open loop control?

The open loop equations are:
u_l = (v* + beta_l) / theta_l
u_r = (v* + beta_r) / theta_r

Advantages of closed loop are:
- Easy to implement
- The system responds fast as there is no feedback as input.
- Cheaper to implement, as there is no code overhead to implement feedback.

Disadvantages are:
- Not very accurate because it is very sensitive to disturbances.
- It does not correct for errors (e.g a slight change of floor friction)

2. What are our equations for closed loop control? Why might we want to incorporate closed loop control, rather than open loop, in SIXT33N?

The closed loop equations are:
u_l = (v* + beta_l) / theta_l - (f_l / theta_l) * delta[i]
u_r = (v* + beta_r) / theta_r + (f_r / theta_r) * delta[i]

We incorporate closed loop so that both wheels converge to the same velocity after applying feedback. This helps the car to correct for disturbance and eventually makes it drive straight.

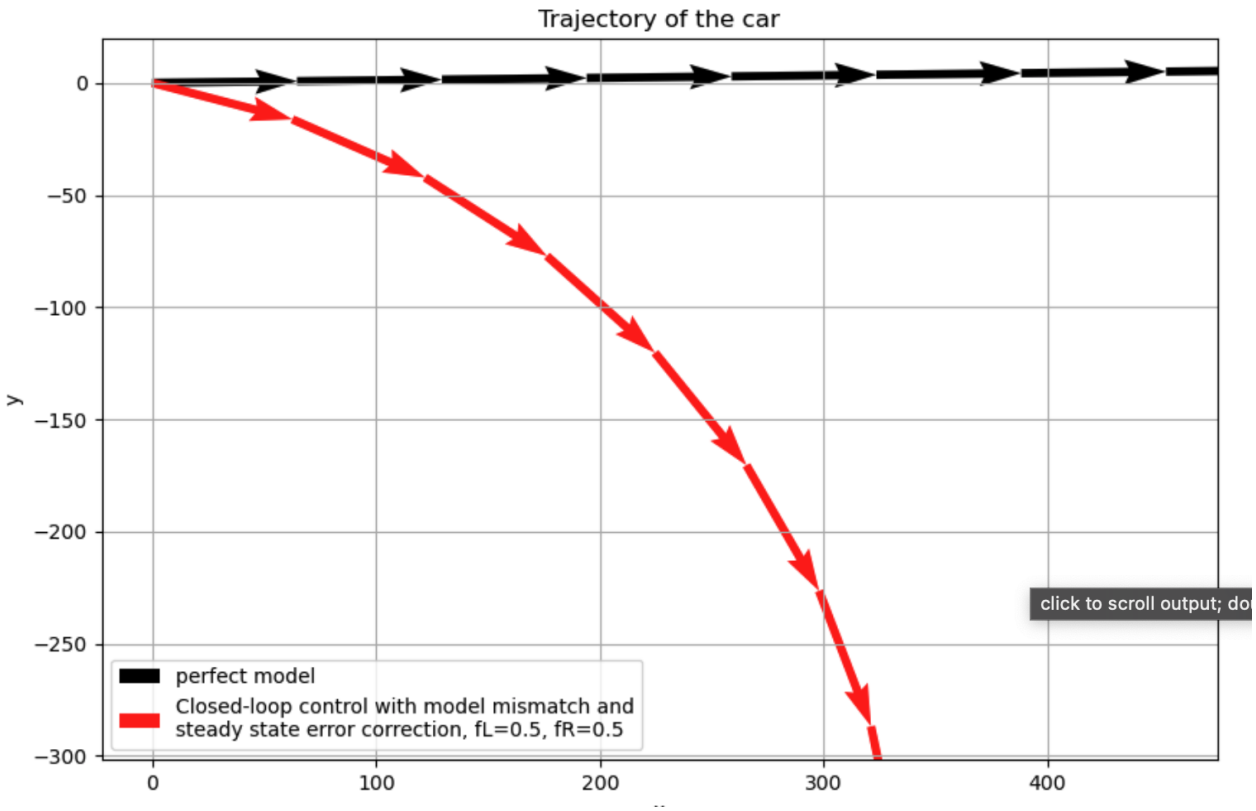
3. What is our system eigenvalue? What are the conditions on fl and fr such that our system eigenvalue is internally stable?

Our eigenvalue is e = 1 - f_l - f_r. For our system to be stable, the absolute value of this expression must be less than 1.That gives us a condition in f_l and f_r:

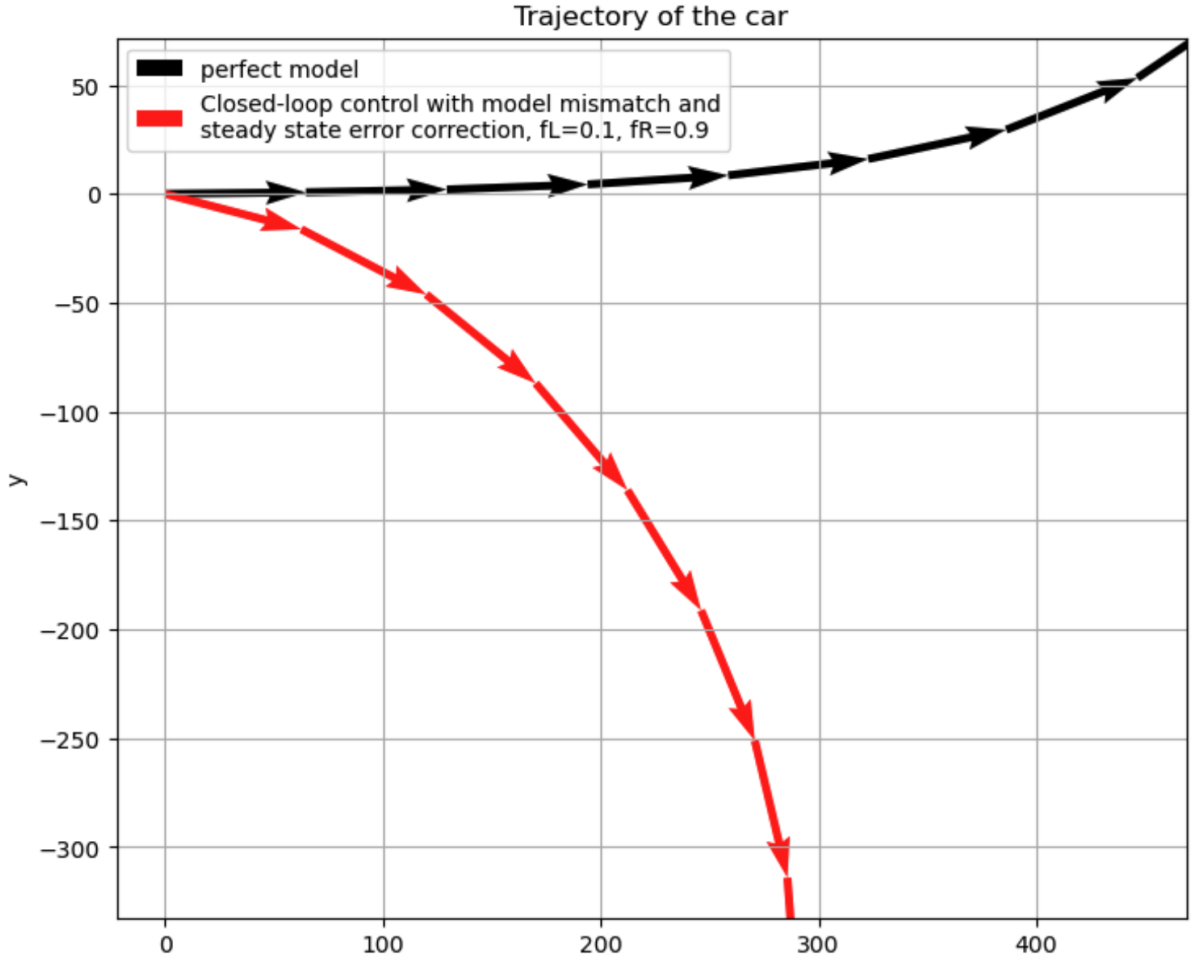
| 1 - f_l - f_r | < 1

4. Draw the trajectory of SIXT33N with f-values of:

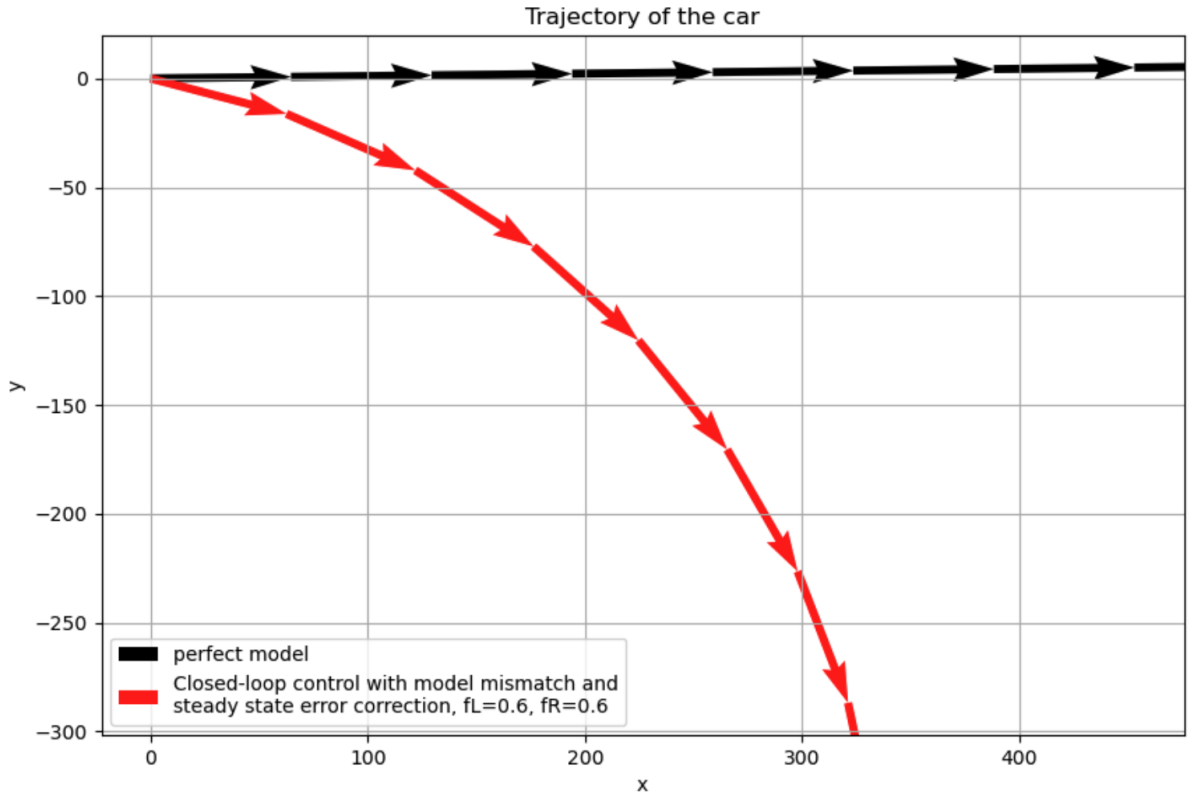
a. fl=0.5 fr=0.5



b. fl=0.1 fr=0.9


Trajectory of the car
perfect model
Closed-loop control with model mismatch and steady state error correction, fL=0.1, fR=0.9
y
50
0
−50
−100
−150
−200
−250
−300
0
100
200
300
400

c. fl=0.6 fr=0.6



Assume your car has a nonzero, positive delta_ss

5. What is the effect of setting both fl=0 and fr=0 ?

No feedback will be given to either wheel, thus, the model reduces to open loop.

6. What is the purpose of the jolts? Why might we have different jolts for left and right wheels?

The jolts are an initial spike in voltage given to each wheel so that they can overcome the initial static friction before they start moving. We might have different jolts for each wheel due to mechanical differences in our car structure (e.g distribution of weight).

7. Why can't we use negative f-values for both wheels? If we wanted to use negative f-values for both wheels, how should we change our closed-loop model equations such that our car goes straight and corrects any errors in its trajectory?

If we were to use negative f values for both wheels, the feedback each wheel would receive would not be consistent with our definition of delta = d_l - d_r. For example, if our delta > 0, that

means, the left wheel is ahead of the right wheel, hence we should give the right wheel a positive value of feedback, so it catches up. However, if we were to use negative f values, the right wheel would instead effectively get a negative value of feedback, which would increase the delta further rather than converging it to a constant. Again, this is inconsistent with our definition of delta. If we would want to use this, we should change the definition of delta to be d_r - d_l .

8. What does a zero delta_ss value tell you about your car's trajectory? What about a non-zero delta_ss value? What kind of error is it supposed to correct when we add it to our control scheme? (Hint: Think about the difference between the trajectories for a zero versus a non-zero delta_ss value.)

A zero delta_ss means that, in the steady state, our feedback loop had effectively converged delta to zero which means the cars is driving straight and in the same direction that is started. However, this rarely happens due to mismatch between the physical system and the model. In practice, our feedback loop converges delta to a non zero value, that means the car eventually drives straight but in a another direction that it started. To account for this error, we add the delta_ss value.

9. How did you change the closed-loop model equations to allow the car to turn? Write the equations below and explain how they change for turning left, turning right, and going straight.

u_l = (v* + beta_l) / theta_l - (f_l / theta_l) * (delta[i] + delta_ref)
u_r = (v* + beta_r) / theta_r + (f_r / theta_r) * (delta[i] + delta_ref)

where delta_ref = CAR_WIDTH*(v*i)/TURN_RADIUS

To support turning, we basically trick the closed loop system into thinking there is an external discrepancy in the delta. We do this so that the feedback loop tries to correct it, effectively producing turning. To do this, we add an extra delta_ref term to our original delta. This term is a function of the turn radious, car width, and time i.

For example, if we wanted to turn left, we give the system an extra positive delta (making the feedback loop think that the left wheel is ahead of the right wheel), so that a positive feedback is produced for the right wheel and the car starts turning right.

In summary:

delta_ref > 0 → delta > 0 → turn right

delta_ref < 0 → delta < 0 → turn left

10. Describe how the trajectory of the car would look if we had a constant δref , rather than our δref , which changes as a function of the timestep.

If delta_ref is constant, the car will make an aggressive sharp turn rather than a gradual nice curve turn.

3 Lab 8: SVD/PCA

Summary

In Lab 8,our objective was to identify four command words that are distinct enough for S1XT33N to easily differentiate, focusing on aspects like syllable count and intonation. To achieve this, we developed an SVD classifier enabling S1XT33N to distinguish between these commands. We then recorded 45 audio samples for each of six words to determine which are most effectively classified by PCA, and we selected 4 distinct words representing a different drive mode (straight far, left, straight close, right) for the Arduino. We made sure to keep track of who recorded each word and how it was pronounced, using phone recordings for reference, which was helpful for later to reproduce these pronunciations during live classification. We then used PCA to compress data for efficient storage within the Arduino's limited memory. Then we visually identified clusters and evaluate how well our word choices worked. Then we monitored the classification accuracy using the Arduino IDE's serial monitor and used an oscilloscope to confirm that our voices were correctly captured. A big challenge we faced was dealing with the differences in how the words sounded each time we recorded them.We had to be very careful to say the words the same way during live tests as we did when we recorded them. This was important to make sure the Arduino could recognize the words accurately.

1. What 4 words did you choose for classification? What characteristics of this set make your words good for classification? Provide at least two features. Compared to other sets of words with similar ideally good characteristics, why is this set preferable to others?

We chose:

- Smile:
  - 2 syllables.
  - Hard ending
  - Emphasis in ‘-mile’
- Apple
  - 2 syllabes
  - Hard beggining
  - Emphasis in ‘A-’
- Back
  - 1 syllable
  - Hard beggining
- Watermelon

- 4 syllable
- Emphasis in ‘Wa-’ and ‘-me’.

This set is preferable to others because it has a variety of variable syllable length words with different emphasis on syllables at different timesteps of the sample. This produces waveforms that are considerably distinguishable to each other which forms clear separate clusters when performing PCA.

2. Why is taking the envelope your the voice signals a good choice for classification, especially considering that this classifier is implemented on an Arduino?

Enveloping is the process of capturing the distinctive shape or magnitude of a speech signal and is particularly important given the limited storage capacity of the Arduino. Each word or sound in a speech signal has a unique pattern, and enveloping helps in visually representing these patterns. Our objective is to filter the audio signal to obtain its envelope, which is crucial for the calculation of the SVD. Once we apply this filtering, we noticed a clearer alignment in the samples. By examining the envelope, we were able identify different words. However, if the envelopes of any words are too similar, it might pose the PCA in differentiating them. In such cases, we might need to consider selecting or recording more distinct words to ensure better recognition.

3. Why do we need to use SVD/PCA to represent our data set?

SVD and PCA are used together to compress data in a way that fits into the Arduino's small memory, while still keeping the most important information. This is important because the Arduino doesn't have a lot of storage space. These techniques can also help us filter out noise or irrelevant information from the data.

4. If we were to use the transpose of our data matrix for SVD, which rows/columns of which matrix correspond to the principal component vectors representing the recorded words? Why?
If we were to use the transpose of our data matrix, that means each word will be a column of our data matrix. That means that we would need the PCA vectors for the columns of our input data matrix when performing SVD. That is, we will need the **columns of U**.

5. How many basis vectors are you using, and how did you choose this number? What are the benefits vs. tradeoffs of increasing or decreasing the number of basis vectors by a small amount? What about for increasing the number of basis vectors by a large amount?

Considering the Arduino's limited memory capacity, we chose to use only three basis vectors. Increasing the number of basis vectors could be counterproductive, as it might lead to capturing more unnecessary noise and also we need to keep in mind that our Arduino's limited memory,

making it challenging to use more basis vectors. On the other hand, reducing the number of basis vectors aligns better with the Arduino's memory limitations However, a potential downside of using fewer basis vectors is the risk of losing important data , which might result in a less accurate representation of the dataset.

6. What are length, prelength, and threshold for our data pre-processing? How does changing them affect your alignment? Include both the definitions and the values you chose. What kinds of words are better suited for our pre-processing method with live classification?

These parameters are used to identify a word in the recording. As the code scans each timesteps it compares the amplitude to the **threshold** times the maximum value in the recording. If it is greater or equal, it starts looking for a word by going back a **pre-length** number of samples, considers that to be the start of the word, and counts
the next **length** number of samples to be the entire word. If we change these, for each word, we might see waveforms that appear shifted/trimmed, and sometimes, some waveforms might be dropped/added if we modify the threshold. The **words** that are better suited are those that have different numbers of syllables, and that differ in soft/hard beginnings/endings. We do this so that the waveforms produced have considerably different shapes for each word.

7. Why can we simply take the dot product when projecting our recorded data vector onto the principal component vectors?

The goal of PCA is to ultimately compress each word (array of 80 timestep columns) to 3 coordinates in 3-vector basis (array of 3 coordinate columns). We do this by taking the dot product between the word array and each of the cols of the basis, to form a 3-element array.

When we take the dot product of our recorded data onto the principal components vector, matrix multiplication achieves the goal described above, for each row of our recorded data, producing a projected data matrix.

8. What is EUCLIDEAN_THRESHOLD? What is LOUDNESS_THRESHOLD? Include both the definitions and the values you chose. During live classification, which threshold was more difficult to satisfy, and how do you know?

Loudness threshold - The maximum value of the sample must be greater than this threshold to be considered a word. If the recorded data is too soft, we do not classify it as it is probably noise. The value we chose is: 150
Euclidean threshold - The distance of the sample to the nearest centroid must be less than this threshold to be considered a valid word. If the L2 norm (distance) is larger than the threshold, the classification algorithm simply ignores it and waits for the next sample. The value we chose is: 0.1

Eucledian threshold was more difficult to classify, as many of the mispronounced words did not classify and we had to repeat them.

9. How different was live classification in practice from what you found in the SVD/PCA lab? Why do you think that is?

Live classification didn't always work as expected, similar to what we found in the SVD/PCA lab. Our Sixteen car was supposed to recognize words by comparing them with patterns we had stored previously. However, we found it challenging to pronounce words exactly the same way each time, and as a result, our car, often misclassified our pronunciation. Background noise also contributed to these misclassifications. To improve this, we moved to a quieter area, but still faced some issues. Finally, we managed to fix these problems by closely examining our enveloped graph, which helped us understand how Sixteen was interpreting our pronunciation and adjust accordingly. This approach made it easier for Sixteen to correctly classify the words despite the variations in pronunciation and background noise.

4 Feedback

1. Extra Credit: To receive extra credit, please provide 1-2 sentences for each of the following parts:

a) How well do you feel that this assignment evaluated your understanding of labs this semester?

This assignment did a great job in showing how much we learned from our labs this semester. At first, we had trouble understanding how the things we talked about in lectures were used in the real world. But the labs, where we got to work with things like SVD/PCA, made it all click for us. So, this assignment was a really good way to see how well we could use what we learned in class in actual lab work.

b) How much time do you think this assignment took you to complete? (Just a number is fine for this part.)

It took us 6 hours.

c) What changes to lab reports would improve your experience?

To improve our experience, I believe shifting from a midterm and final report format to writing individual reports for each lab would be more beneficial.

2. Additionally, please feel free to provide any feedback you have about the 16B lab or anything we can do to better support you.
Regarding the 16B lab, I'd like to mention that our TAs did a great job, but they often seemed overworked. There were instances where only one TA was available to assist the entire Lab, which made it challenging to receive help.