

Control Systems Analysis Laboratory:  
Lab #7  
*Machine Learning Control Systems*

Authors:

Nick Wilde,  
Nathan Phipps,  
Anthony Stehr,  
Erich Wanzek

# DHT11 Temperature and Humidity Sensor

Test Run by: Nick Wilde

## Summary of Previous Tests

In the previous lab, several tests were run on the DHT11 to quantify the characteristics of both the temperature and humidity components of the sensor. These tests were designed to determine the accuracy, stability, sensitivity, and efficacy of the DHT11 and were run in 6 different scenarios, high temp, medium temp, low temp, high humidity, medium humidity, and low humidity.

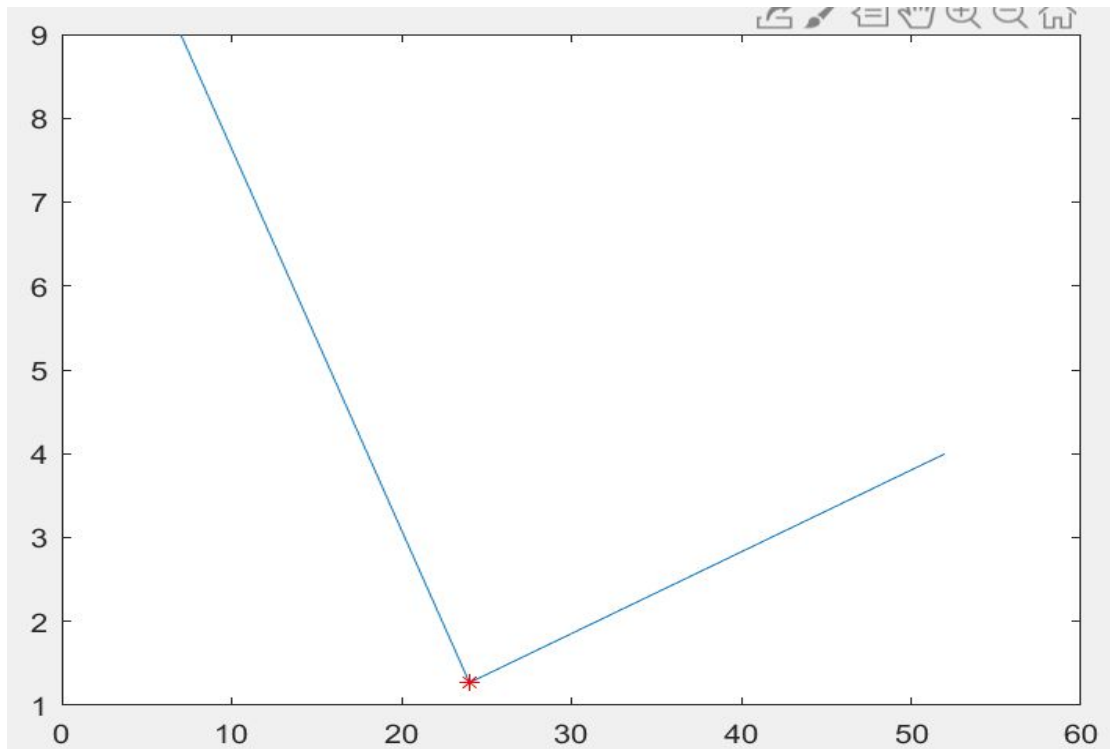
## Results of Previous Tests

The results of the previous lab yielded definitive results for the ideal conditions to operate the DHT11 with respect to temperature. However, humidity produced conflicting results that force the user to determine which aspect of the sensor is more pertinent to the design of the system.

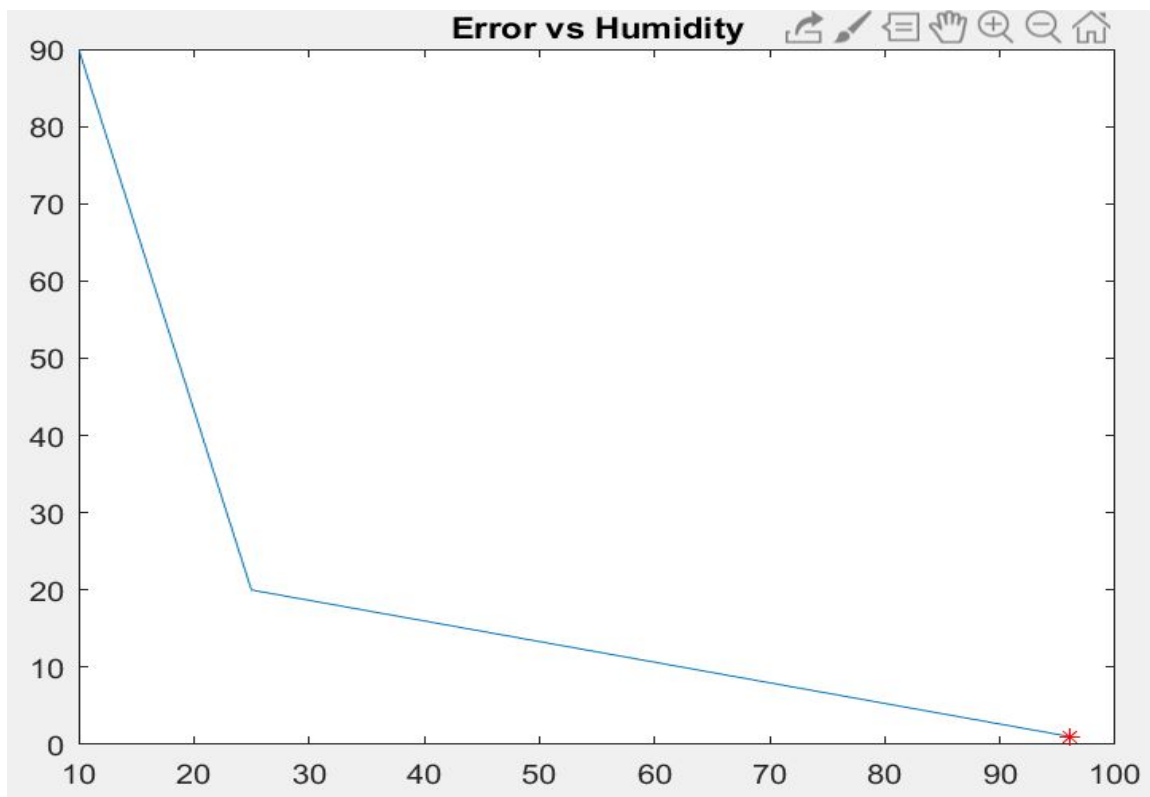
### Accuracy Test

	Temperature	Humidity
Ideal Conditions	24 C	95%
Error at Conditions	1.27%	1.04%

As shown above, the ideal operating conditions for accuracy is a warm environment with high humidity. At these conditions, the error for both temperature and humidity are at their minimum of around 1%. The previous lab also yielded that the increase in error as the temperature raised is lower than the increase when the temperature is decreased. This means that if the temperature needs to be varied, it should be increased instead of decreased. The error in the humidity increases dramatically as the humidity is decreased. Below are plots of error vs temp and humidity with the ideal operating conditions marked in red.



**Error vs Temperature**



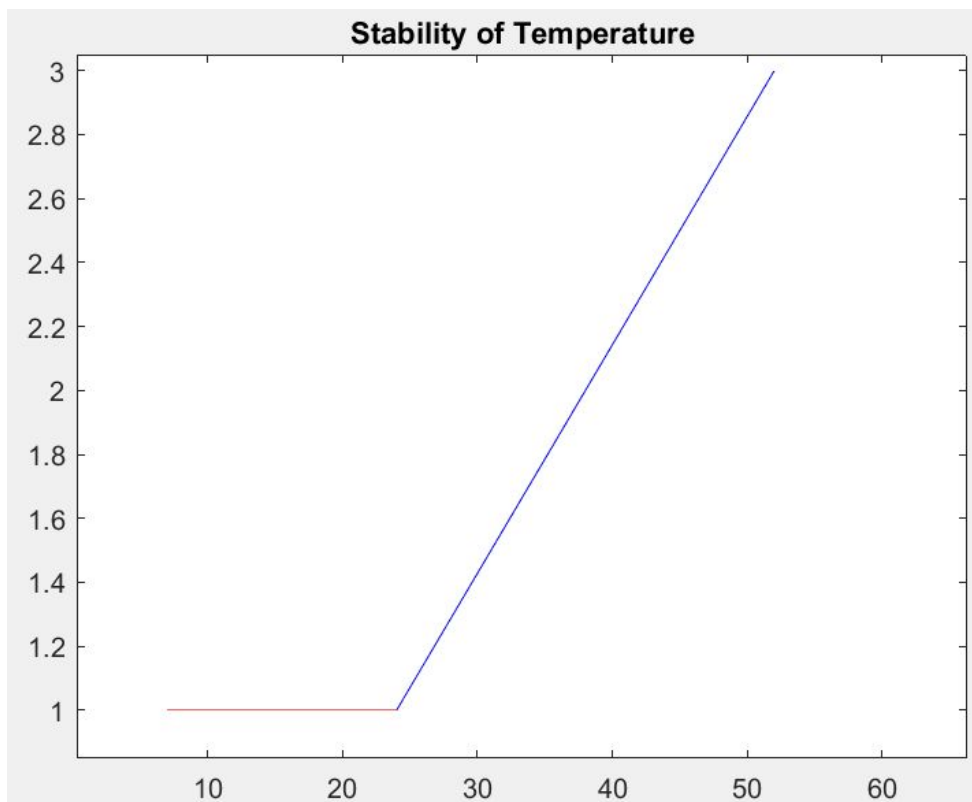
**Error vs Humidity**

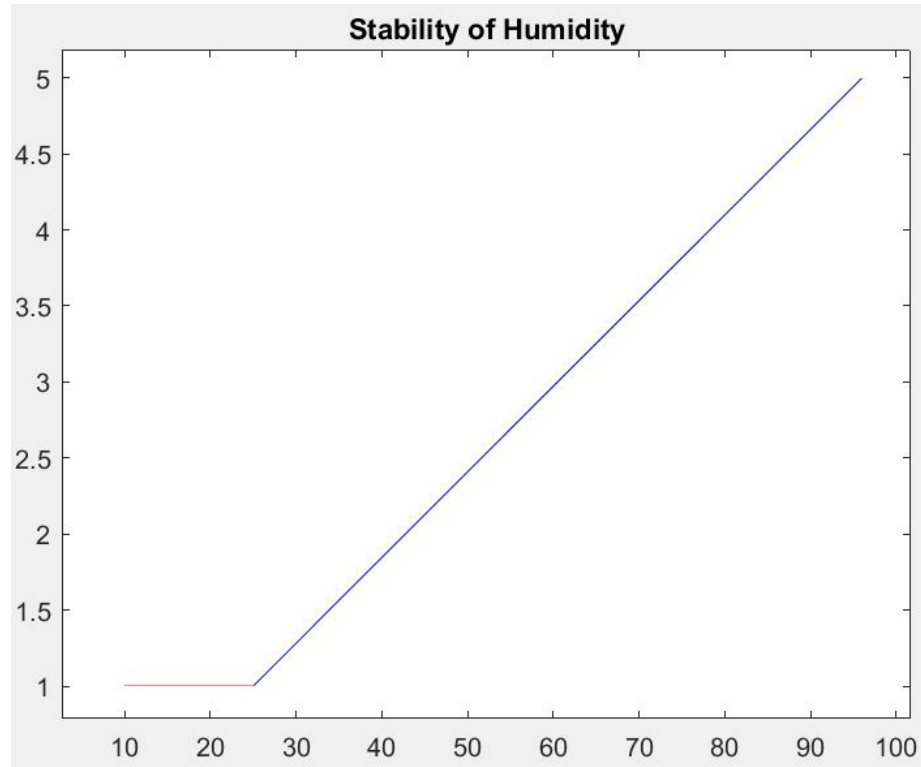
## Stability Test

As discussed in the previous lab, the reference thermometer used to determine the actual temperature and humidity of the room showed no variation in its steady state. This means that the error in stability between the DHT and the thermometer is always 100% for any value. Therefore, instead of an error measurement, the stability is measured by the variation in degrees or percent.

	Temperature	Humidity
Ideal Conditions	7 - 24 C	10 - 25%
Variation at Ideal	1 degree	1%

The stability of the temperature sensor was constant at 1 degree between 7 and 24 degrees celsius. The humidity stability was constant at 1% for the range of 10 - 25%. The plots for stability of temperature and humidity are below with the optimal region of operation highlighted in red. These results indicate that the sensors are most effective with respect to stability when the temperature and humidity are in the range of low to medium.



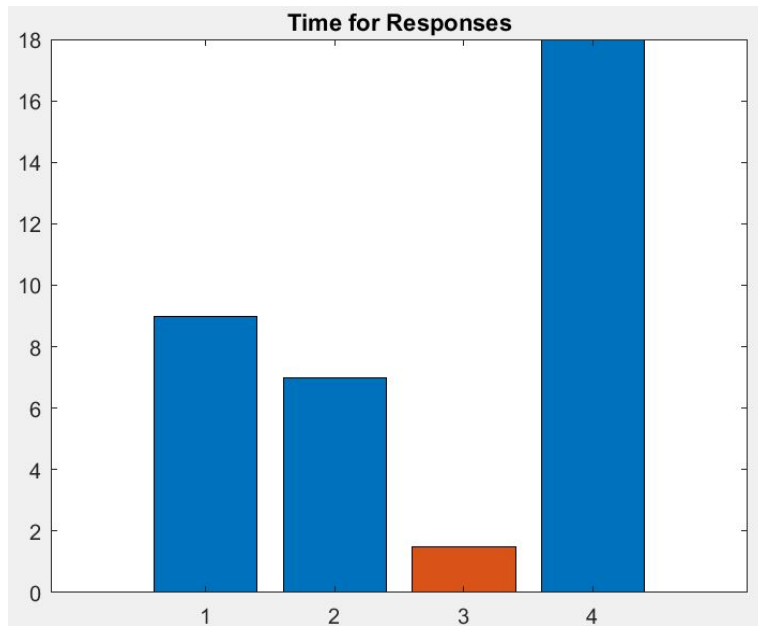


### Efficacy Test

The efficacy test examined the response of the DHT11 sensor to changes in temperature. The situations tested were medium to high, high to medium, medium to low, and low to medium situations for both temperature and humidity. The quickest response times are shown below.

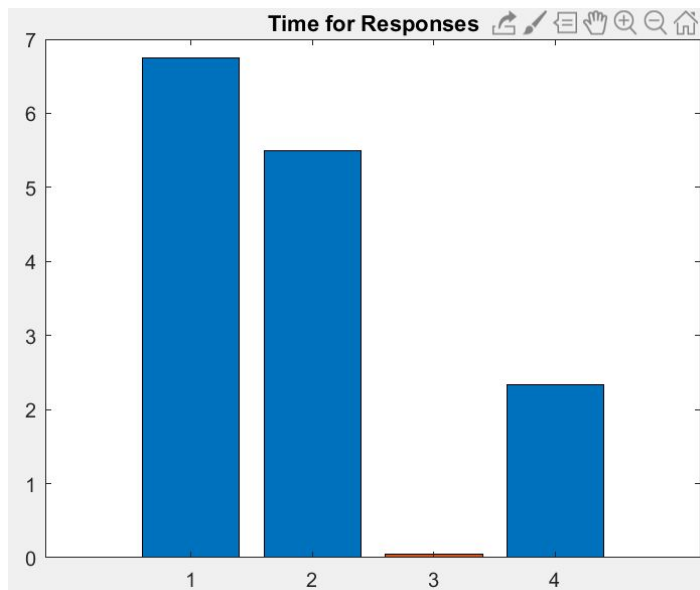
	Temperature	Humidity
Quickest Situation	24 to 52 increase	30% to 95% increase
Quickest Response	1m 30s	3s

The quickest responses seen from the sensor occur when increasing the temperature or humidity from medium situations (24 degrees or 30%) to high situations (52 degrees or 95%). The response times in minutes are shown below with the ideal situations in red. It can be seen that although the temperature response in bar 3 is the quickest, once the max temp has been achieved, the cool down to the normal temp is extensive. This indicates that the sensor is best utilized in situations when you need a quick response to increases in temperature but not decreases.



**Temperature**

1. 24 to 7 degrees
2. 7 to 24 degrees
3. 24 to 52 degrees
4. 52 to 24 degrees



**Humidity**

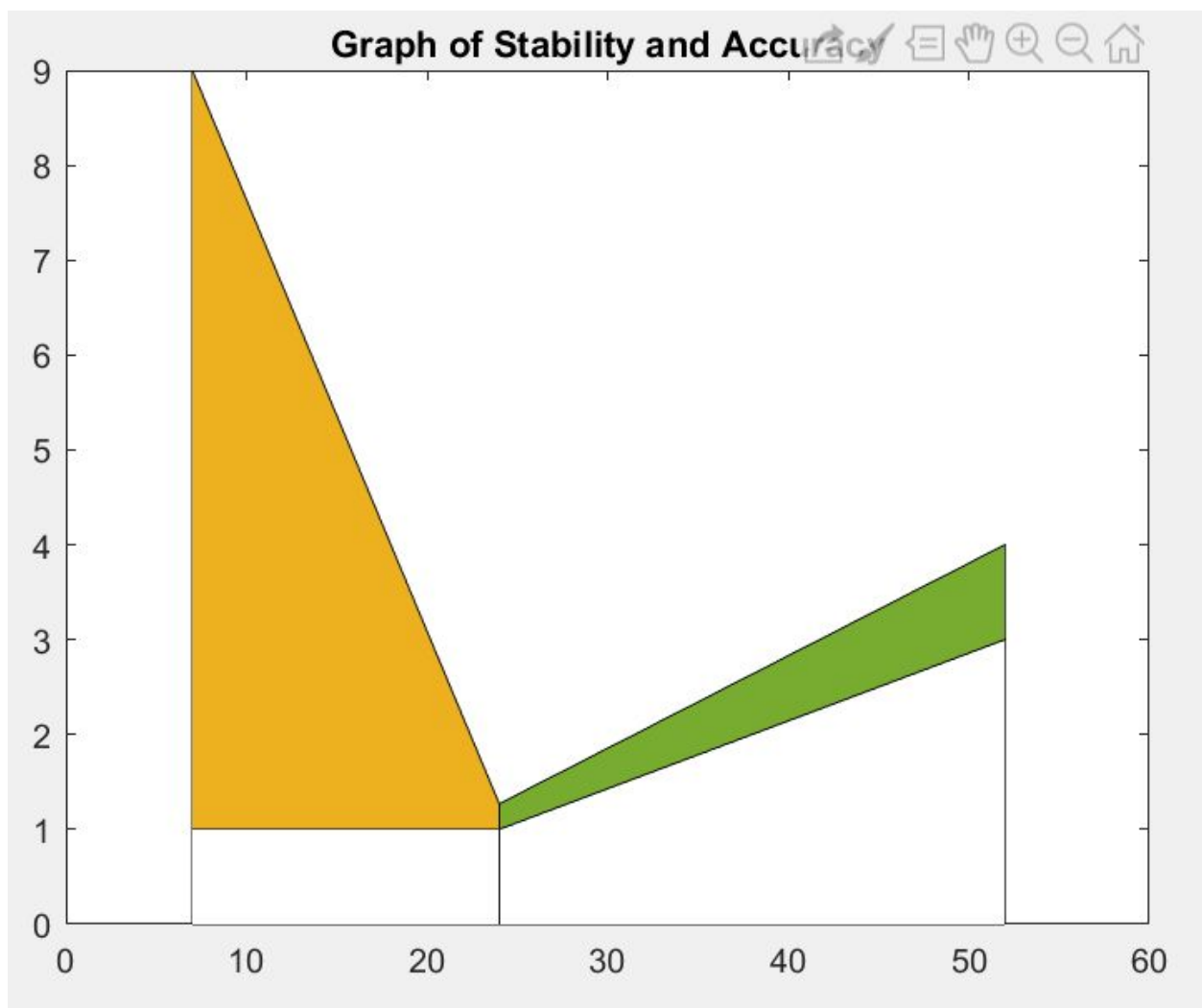
1. 10% to 30%
2. 30% to 10 %
3. 30% to 95%
4. 95% to 30%

Similar to the temperature response, the quickest situation occurs from 30% to 95%. However, the time it takes to return to the normal is much less for humidity and is the second quickest response for the sensor. This indicates that the sensor is best suited for uses where humidity increases from the normal value, and can also handle the reduction in humidity from the max.

## Ideal Operating Conditions for DHT11 Sensor - Temperature

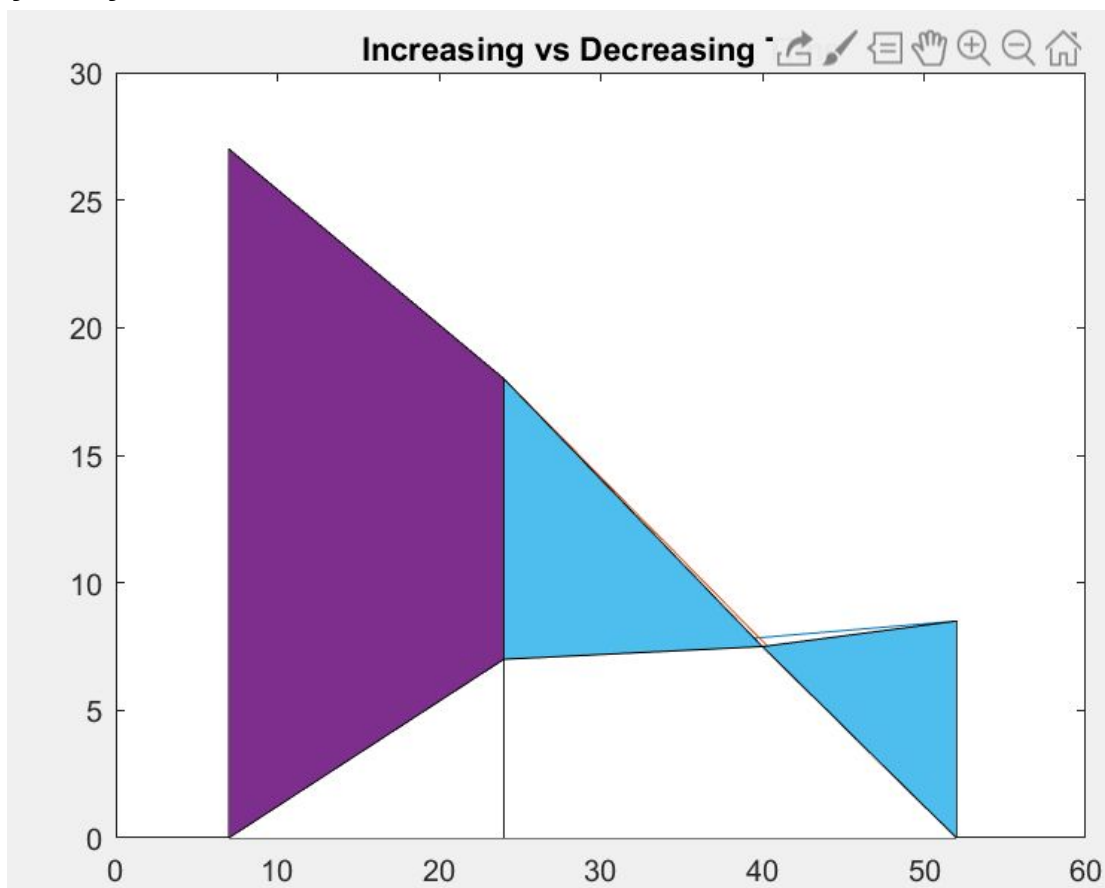
The results from the previous lab indicate areas of operation where the temperature and humidity sensors are best suited for use. To determine the best range of operation, all of these characteristics must be compared to each other to find the region that incorporates the best response. Since some of the results indicate that different regions are better suited for some characteristics, the operating conditions of the sensor should be determined by the order of importance.

### Accuracy vs Stability



Above is a graph of stability and accuracy vs temperature. The stability is the lower line and the accuracy is the higher line. The yellow area represents a temperature region where the sensor is outputting very stable results with approximately 1 degree variance. However, it can be seen that the accuracy of the results increases to 9% as the temperature decreases. This yellow region represents the area of operation where very low stability is valued more than accuracy. The green region is an area where the accuracy increases much slower, but the stability also increases. This is a region where the sensor can be operated if the accuracy and stability are equally valued due to the much smaller error at the expense of stability. The optimal region for both stability and accuracy is at room temperature at approximately 24 degrees celsius. However, since this point is at the boundary of the yellow and green regions, it is safer to operate the sensor at a slightly higher temperature to prevent variances from pushing the sensor into the yellow region.

## Efficacy Analysis

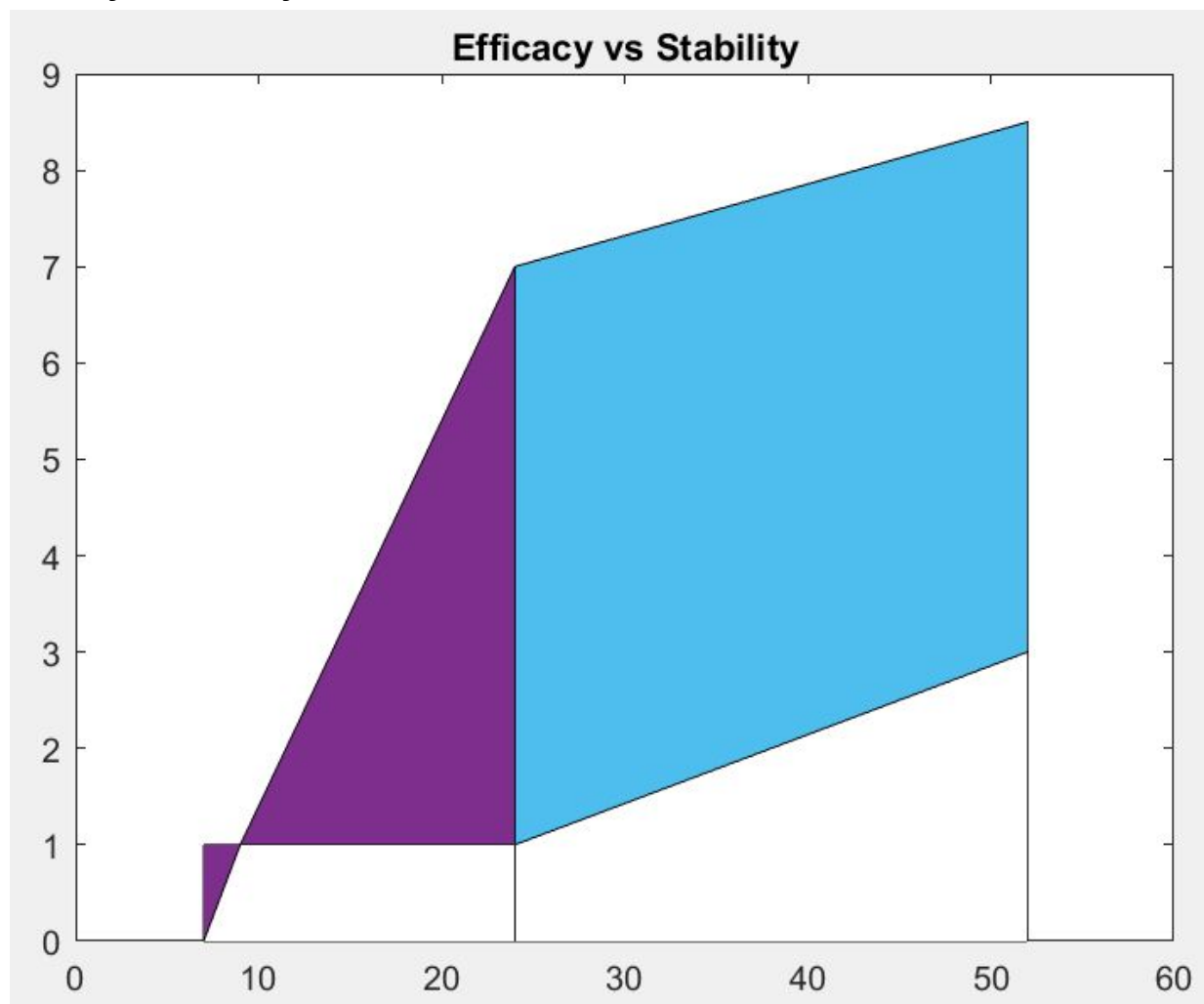


Since the DHT11's response to temperature change is different depending on if the change is increasing or decreasing, an analysis of the individual responses must be done. The increasing line is formed by taking the response as the temperature is being



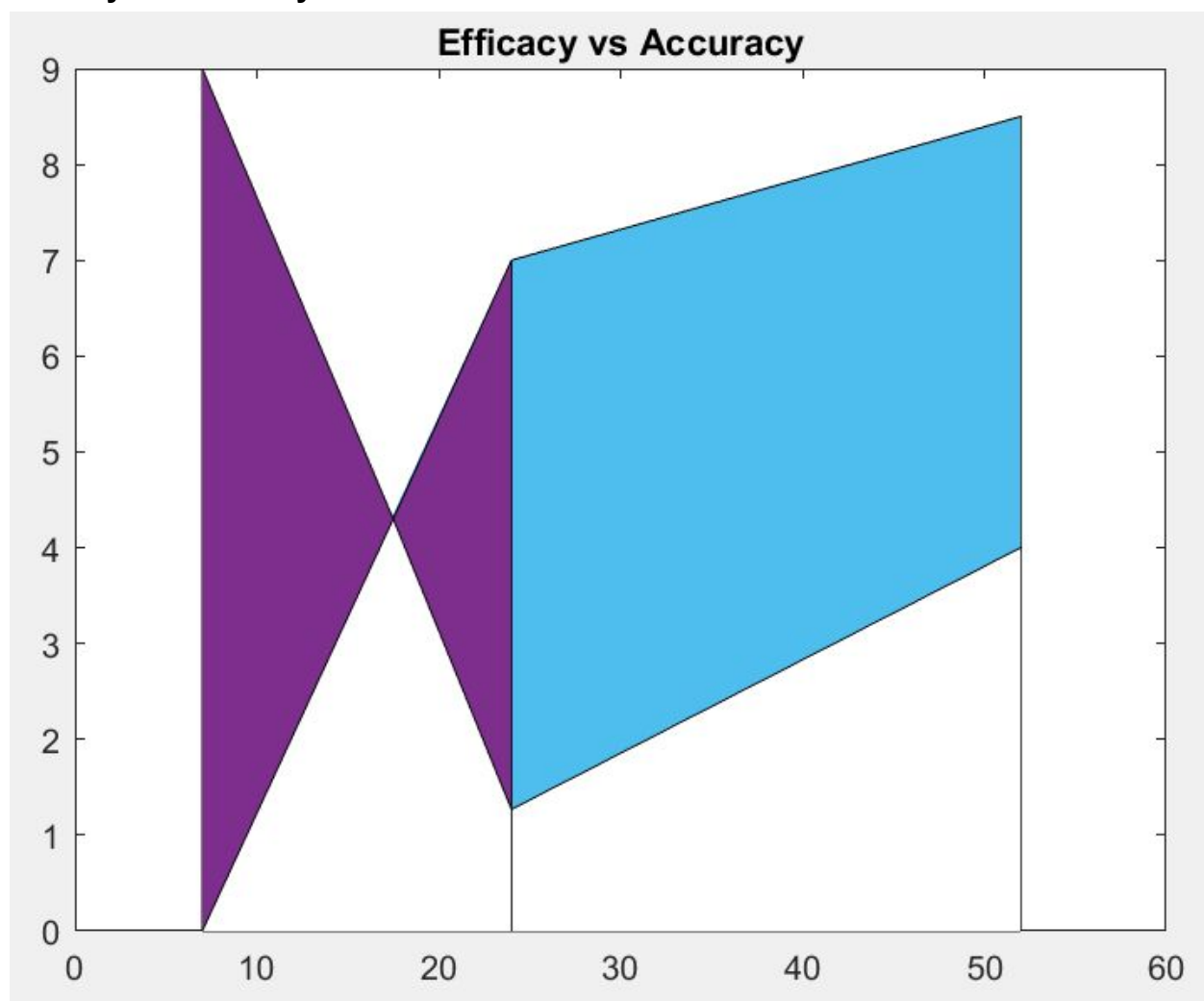
raised from 7 to 24 degrees and combining it with the response as the temperature is raised from 24 to 52 degrees. The opposite is done for the decreasing line. It must be added that the decreasing line on top must be read from right to left to mirror the temperature change occurring. This graph contains two different regions; the first region, purple, describes the area where the sensor goes from 24 to 7 and 7 to 24. It can be seen that both the increasing and the decreasing line have a large slope here. The second region, blue, shows the decreasing line still has a steep slope. However, the increasing line has a much smaller slope. This blue region is the ideal operating region for the sensor. This means that the sensor is best used in a situation when the temperature is increasing from room temp. It must be added that even though this is the ideal operating region, the decreasing line is still steep, indicating that the response to cool down will still be slow. This should be taken into account when designing the system. Since the increasing line defines the ideal operating region, future analysis will ignore the decreasing line.

### Efficacy vs Stability



In the graph, the lower line represents the stability of the system and the upper line represents the efficacy, with the slope being an indication of the response time. The above comparison of stability and efficacy has two regions. The purple region is a temperature area with very low stability and a slow response time. This region should be utilized if the stability of a sensor is much more important than the response time of the sensor. The second blue region exhibits a less stable system, but a quicker response time, making it a better choice for systems where both efficacy and stability are valued. The blue region is a more well rounded choice because it dramatically increases the response time of the sensor while only slightly decreasing the stability. As with the accuracy and stability test, the sensor should be placed a small ways from the blue and purple border to receive the maximum beneficial traits while also preventing small variances from pushing it into the purple region.

### Efficacy vs Accuracy



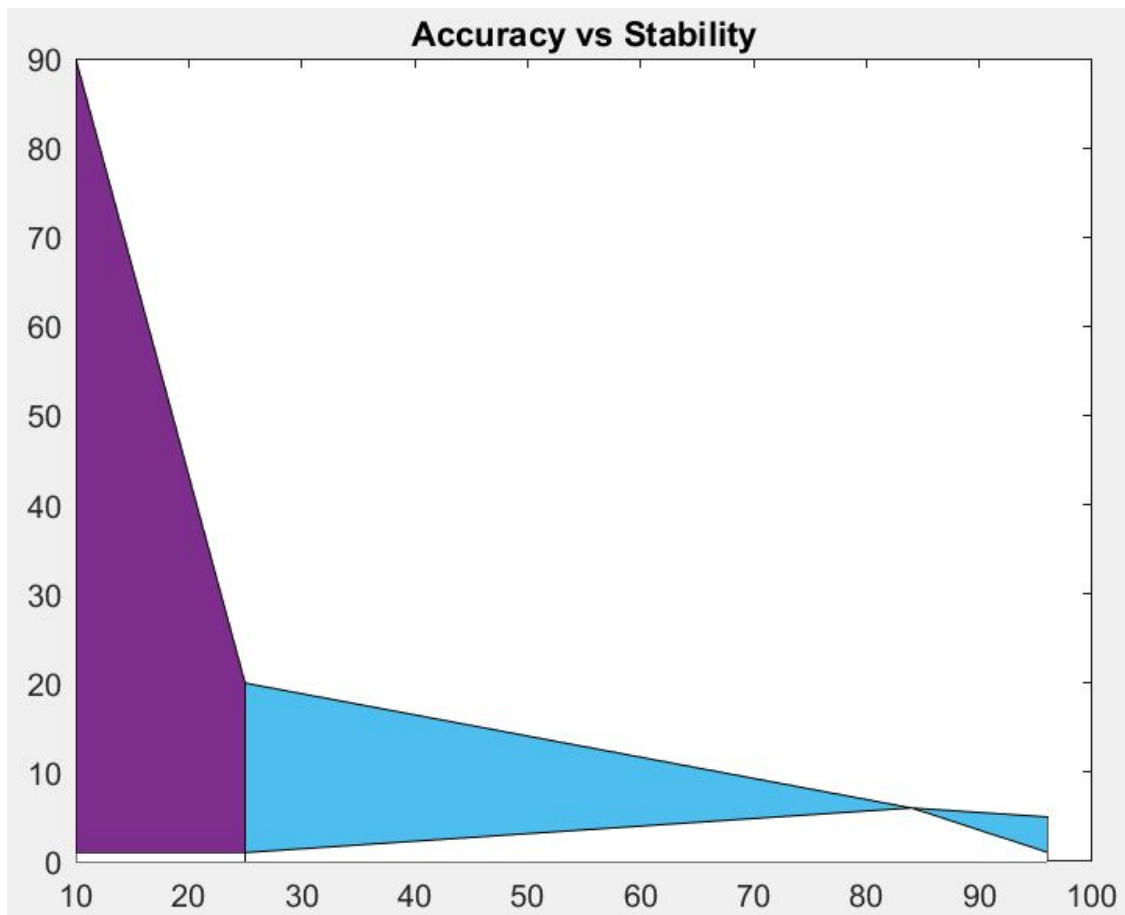
In the above graph, the upper line in the blue region is efficacy while the lower line is the error of the sensor readings. Once again, the sensor operates in two regions with the blue region representing the obvious choice for operation. In the purple region, the response time is high and the accuracy is low, making it undesirable. The blue region has both high accuracy and a quick response time.

### **Ideal Operating Conditions: Temperature**

This analysis shows that every characteristic except stability is optimized in the blue region. Since stability undergoes very minor decreases in optimization, the sensor's operating point should be slightly above 24 degrees celsius, or the border of the purple and blue region. This location allows for the best accuracy and response time while preventing variations in temperature to push the sensor into the purple region.

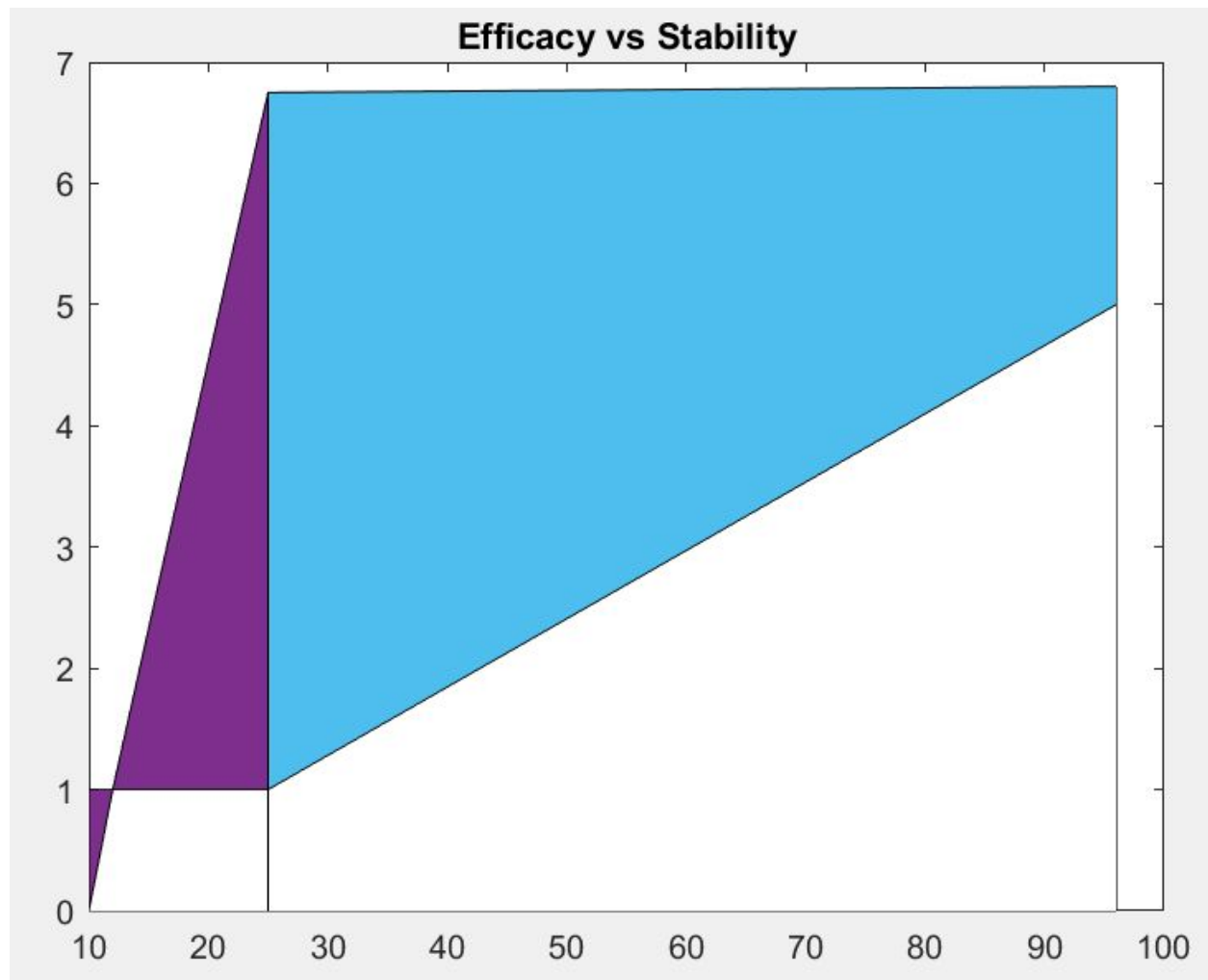
### **Ideal Operating Conditions for DHT11 Sensor - Humidity**

#### **Accuracy vs Stability**



The above graph depicts accuracy vs stability wrt percent humidity. The upper line is the error and the lower line is the percent variation at steady state. It can be seen that the error is very high in the purple region, making it undesirable. However, at the boundary of the purple and blue region, the error is still undesirable. As the humidity increases from there, the accuracy decreases and the stability decreases. This makes the blue region the ideal operating point. Unlike temperature, the sensor is better suited to be operated far into the blue region instead of at the boundary due to the significant decrease in error in the insignificant decrease in stability.

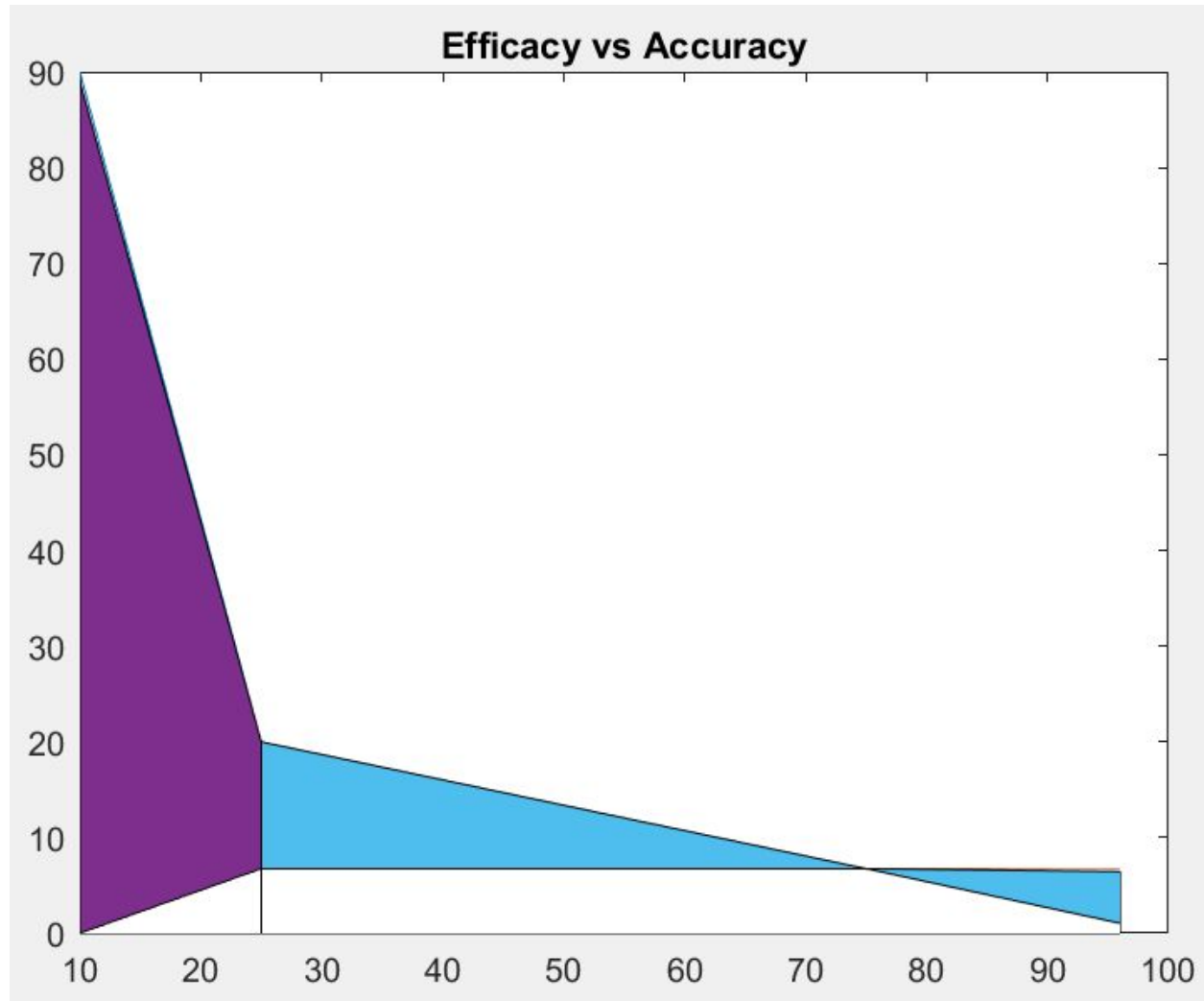
### Efficacy vs Stability



In the above graph of efficacy and stability, the slope of the upper line represents the response of the system while the lower line represents the percent variation during steady state. As seen above, in the purple region, the slope of the upper line is steep, meaning that the sensor has a very slow response time. However, in the blue region, the slope is approaching 0 indicating a very fast response time. The stability of the system appears to deteriorate as the humidity is increased, so the ideal operating point

to optimize the stability and efficacy of the sensor is around 24% humidity, or the boundary of the blue and purple region.

### Efficacy vs Accuracy



The graph above depicts accuracy as the upper line and efficacy as the lower line. It can be seen that in the purple region, the error of the sensor is very high and the response time is very slow, due to the steepness of the slope. However, once the blue region is reached, the response time drops dramatically along with the error of the system. Since the response time increases by less than 1 minute as it approaches the maximum humidity and the error continues to drop, the optimal operating region for accuracy and response time is at the end of the blue region around 96% humidity.

### **Ideal Operating Conditions: Humidity**

Similar to the temperature component of the sensor, the accuracy and response time are better suited for high humidity while the sensor is more stable in lower humidities. However, unlike the temperature sensor, the accuracy continues to increase as the humidity is increased. Since the stability of the sensor undergoes smaller deterioration than the improvement of the accuracy as the humidity is increased, the ideal operating point of the DHT11 humidity sensor is at higher humidities (around 90%).

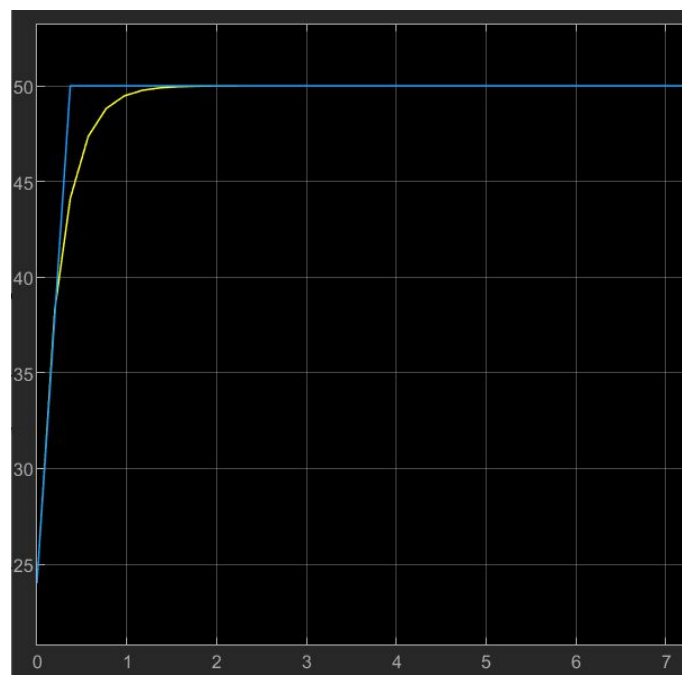
### **Effects of Scaling on Design**

By scaling the circuit, slight variations in the performance of the sensor can arise. As stated before, to account for these variations, the recommended operating conditions may not be the same as the ideal operating conditions. The recommended conditions tend to vary from the ideal because of the location of a drop off of performance. To avoid the possibility of entering this detrimental region of operation, the recommended conditions are pushed further into the acceptable region at the cost of mild reduction of performance.

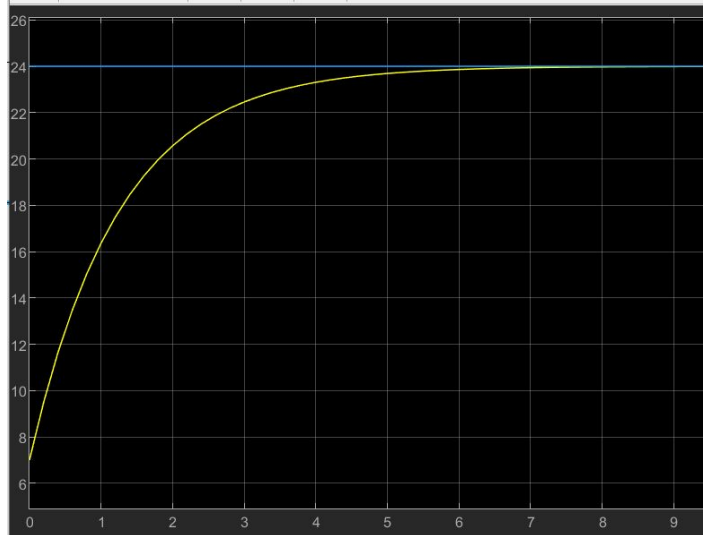
### **Simulink Model for Ideal Conditions**

The operating conditions above can be verified using a simulink model describing each of the characteristics of the sensor.

### **Efficacy**



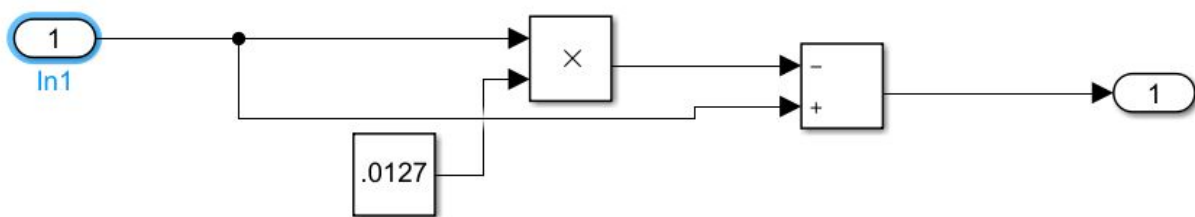
Increasing Temp from Ideal Conditions



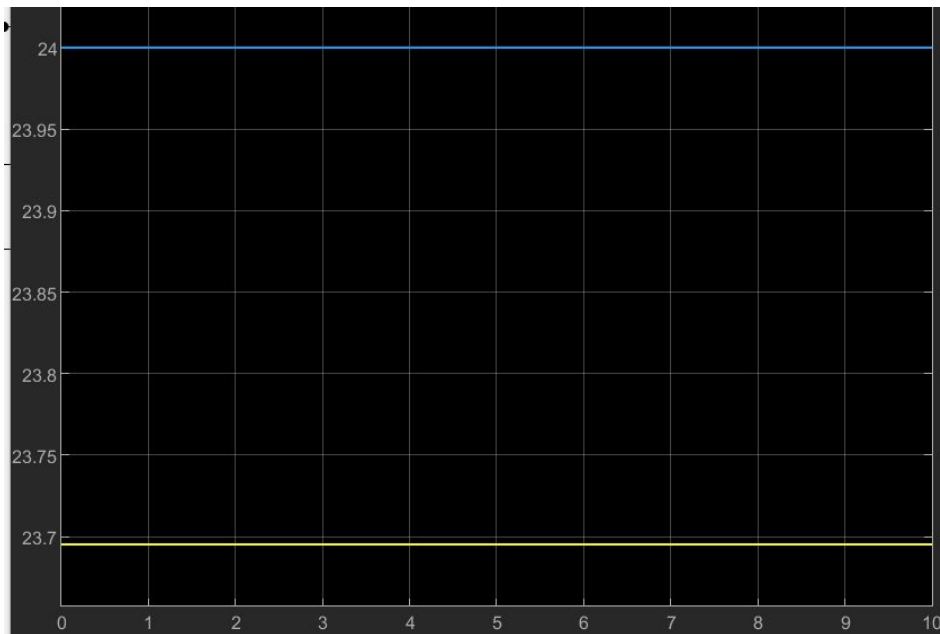
Increasing Temp from Non Ideal Conditions

The above model depicts the response due to a temperature change. The first response is when the sensor is operated at its ideal conditions around 25 degrees celsius; the time it takes to reach its final value is approximately 1 and a half minutes. The second response is taken from a location far from the ideal operating point, 7 degrees celsius. The response time from this is approximately 7 minutes. This much quicker response is indicative that the operating point chosen, slightly above 25 degrees, is ideal for response time. The simulink model used is the same as the one used in the previous lab.

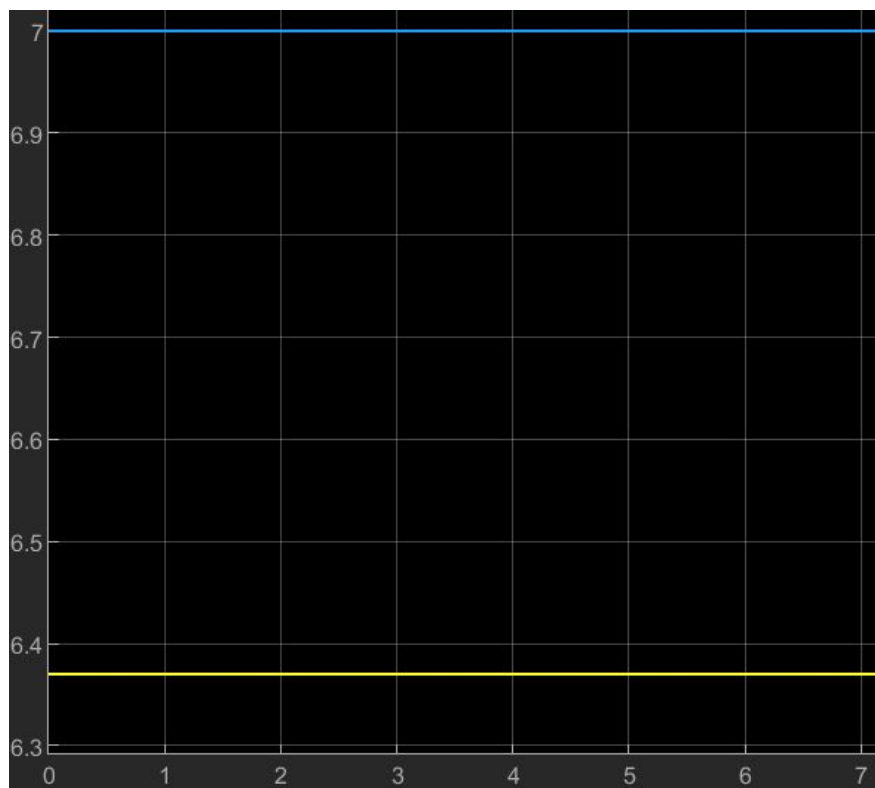
### Accuracy



Accuracy Model



Accuracy at Ideal Conditions

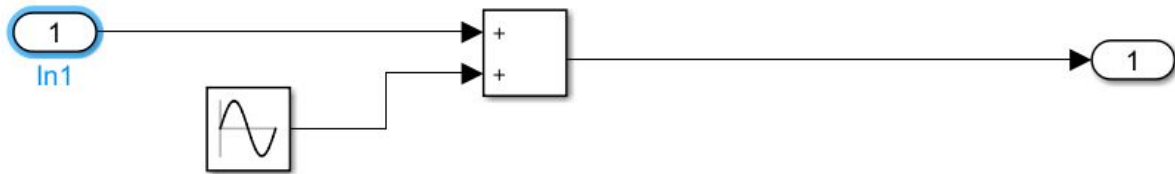


Accuracy at Non Ideal Conditions

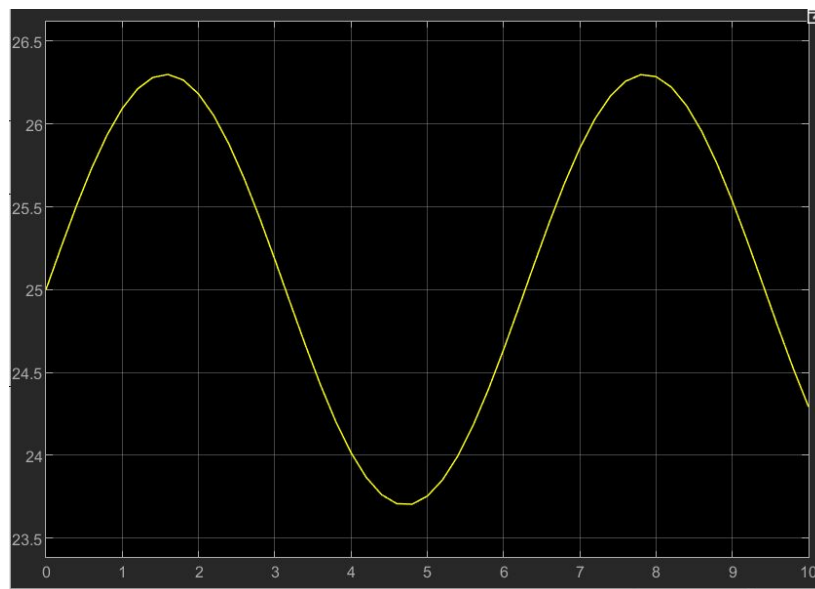


The accuracy model above depicts the error seen at ideal and non ideal conditions. The error at ideal conditions, slightly above 24 degrees, is approximately 1.3% and can be seen in the variation from the input temperature. The error at non ideal conditions, 7 degrees, is much higher at approximately 9%. These results verify the recommended operating conditions due to the smaller error seen.

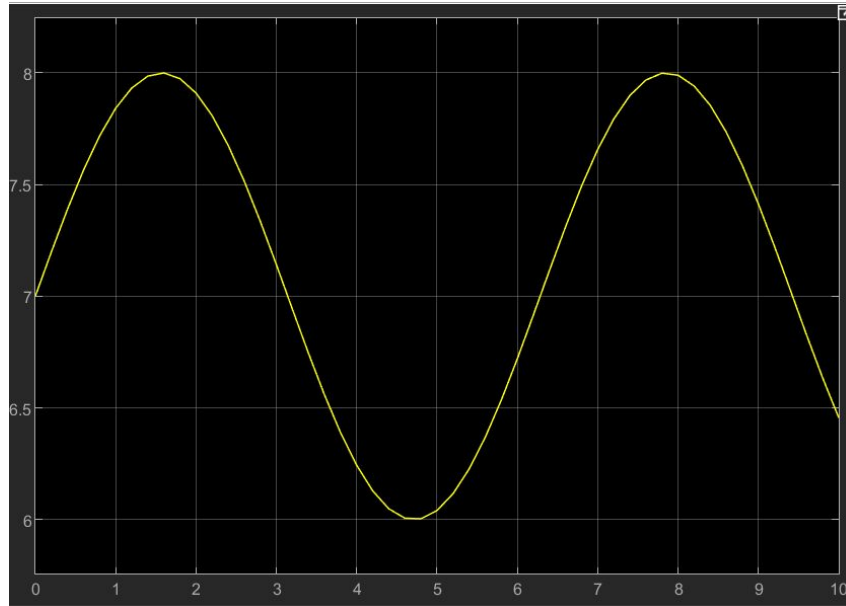
## Stability



Stability Model



Stability at Recommended Conditions

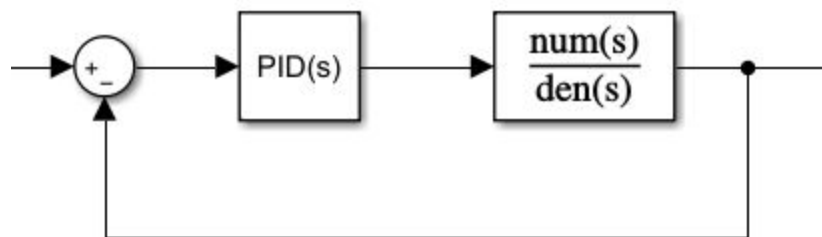


Stability at Ideal Conditions

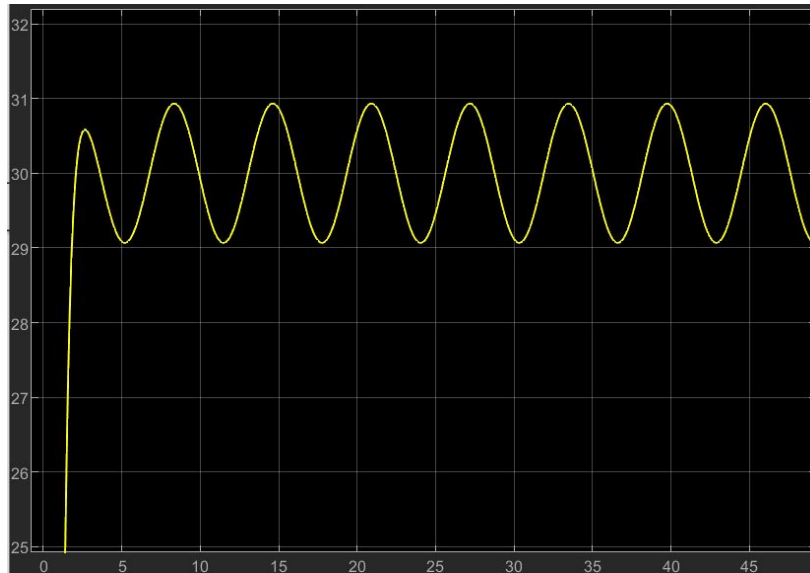
As previously stated, to avoid the possibility of entering a detrimental region of operation, the recommended operating point is placed in an area where the stability is not ideal. This can be seen in the model above, where the variation is 1 degree at the ideal conditions and slightly above 1 degree at recommended conditions.

### Implementation using PID Controller

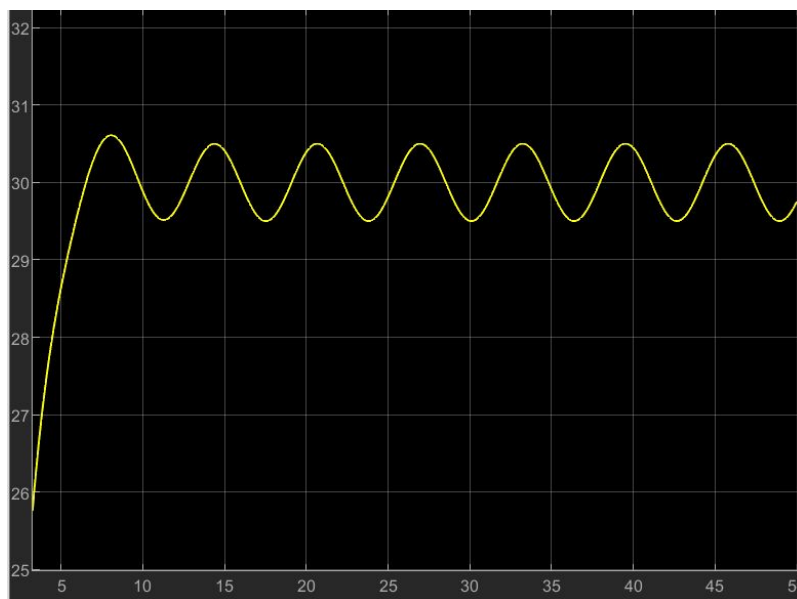
As previously mentioned, the only characteristic of the sensor that is not optimized at the recommended operating conditions is the stability. Because of this, the controller should be focused on eliminating the variation of the sensor at steady state conditions. Below are the results of the simulation with and without the PID controller. It can be seen that the controller is able to half the amplitude of oscillations in steady state.



PID Controller and Plant Subsystem



Original Output



Output with PID

# Voltage Sensor Module

## Results from Previous Lab

The results from the previous lab were able to give a description of the accuracy, stability, and response time of the sensor. However, the only characteristic that gave a reliable region where operation was most reliable is accuracy. The stability, classified by the amplitude of the sine wave seen at steady state, appeared to vary randomly for different input voltages. Also, the largest amplitude seen is 15 mV away from the lowest amplitude, bringing into question if optimizing the stability is even necessary. The response time of the sensor was too quick to accurately measure.

## Accuracy

Input Voltage	Error
1	8.1%
5	2.45%
10	.348%
15	.5%
20	.645%

The above table describes the error for various different input voltages. The minimum error is seen around 10V, making it the ideal operating point of the sensor. The maximum error occurs at 1V.

## Stability

Input Voltage	Amplitude of Noise (V)
1	.075
5	.06
10	.075
15	.065
20	.06

As stated above, the stability has no clear optimal operation point. It can be seen that the amplitude of noise varies from .075 to .06 V with two locations with minimum amplitude. These vague results combined with the low noise levels for all inputs make it unnecessary to optimize the operating point.

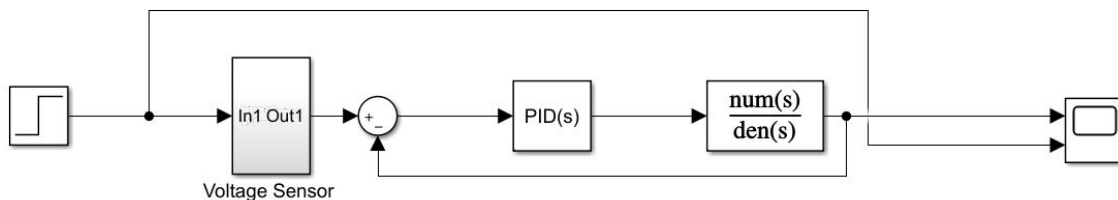
### Efficacy

The response time of the sensor was difficult to measure due to the quickness of the response. Because of this, the results taken were inaccurate and misleading due to the limitations of human response time. Therefore, the efficacy is considered to be equally fast at all inputs removing the need to find the optimal operation point.

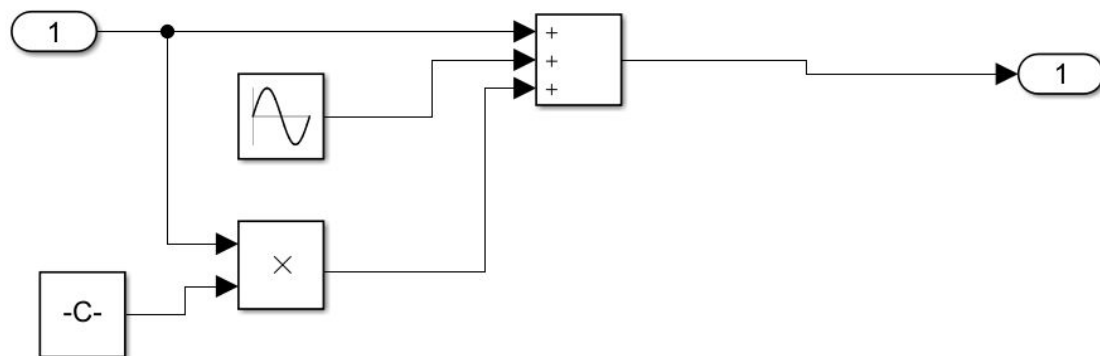
### Ideal Operating Point

The ideal operating point of the sensor is characterized solely by the accuracy for various inputs, as explained above. Because of this, the recommended operating point is around 10V, where the sensor had the minimum error of .348%.

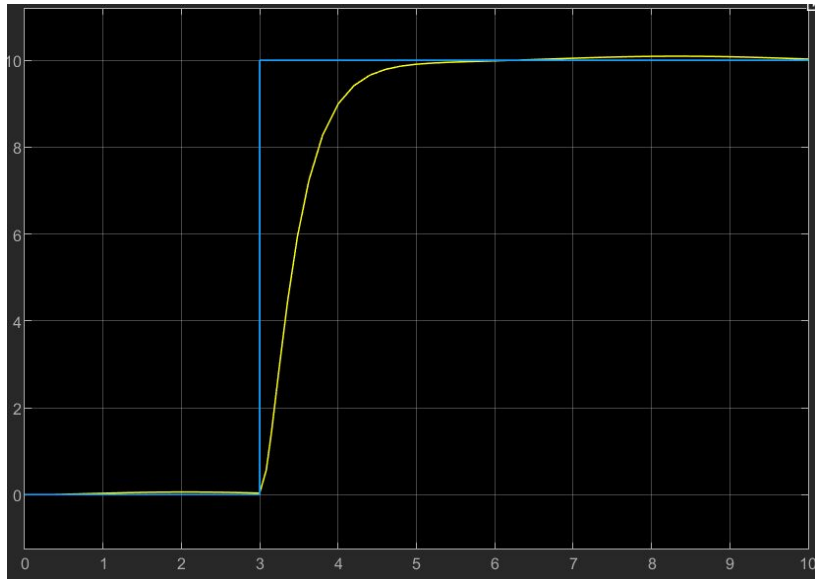
### Simulink Model with PID Controller



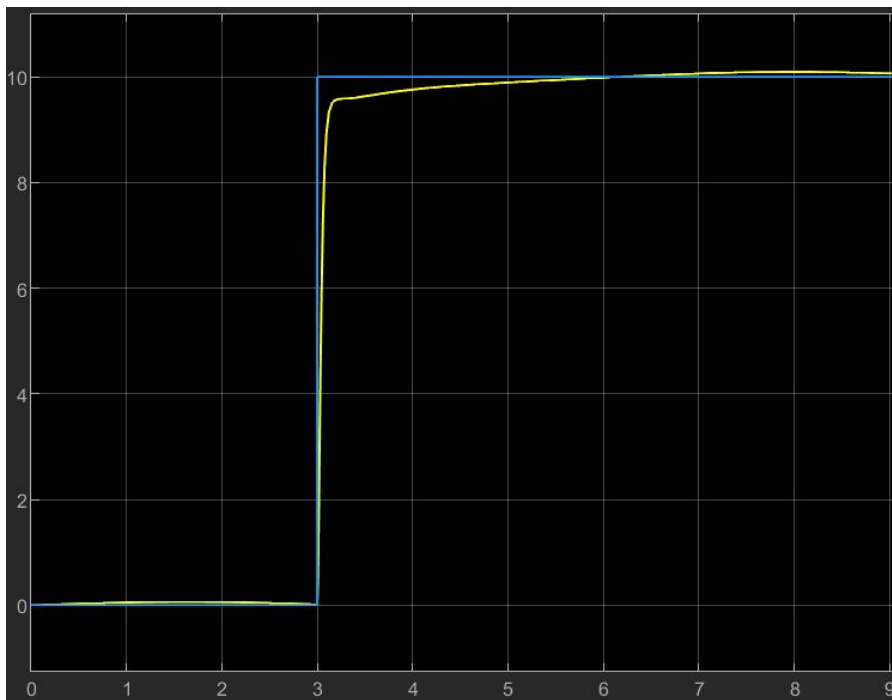
System with Sensor, PID, and Plant



Model of Voltage Sensor for Stability and Accuracy



Original Output without PID Controller



Output with PID Controller



# Sound Detection Sensor Module and DS18B20 Temperature Sensor:

Test Run by: Nathan Phipps

## Optimal Settings Sound Sensor:

**Note:** For full descriptions of each experiment carried out in Sensor Lab # 1 please consult my previous lab submission for sensor lab 1. For the sake of brevity, the in depth descriptions from sensor lab 1 have been omitted to avoid redundancy and to save time. My purpose is to use the data provided below to establish the “sweet spot set point,” or the optimal settings from my previous labs in order to adapt my lab 7 to be based upon the principle of control based upon data. Optimal Control based analysis are peppered into these previous data sections since the lab 7 guidelines require optimal settings analysis. Any new “control based on data,” sections within my data from previous labs will be highlighted. After these sections a section labeled “*Summarizing Control Based On Data, Finding a “Sweet Set Point” for Sound Detection Sensor and DS18B20 Temperature Sensor,*” will mark where the majority of new lab work begins for lab 7.

## Sine Wave Data:

Low Sine:



Figure: Low Sine

A low sine is played at note E2



Mid Sine:



Figure: Mid Sine

A mid sine is played at note E2

High Sine



Figure: High Sine

A high sine is played at note E2

**Reliability: Sound Detector**

Trials From Eight Cases (Low Volume = -14dB, High Volume = -10 dB, Note Played = E2):

	Low Sine	Mid Sine	High Sine
Experiment 1 (Low Volume)	0	1	0
Experiment 2 (Low Volume)	1	1	0
Experiment 2 (Low Volume)	0	0	0
Experiment 2 (Low Volume)	0	1	0
Experiment 5 (High Volume)	3	3	2
Experiment 6 (High Volume)	3	3	1
Experiment 7 (High Volume)	3	3	1
Experiment 8 (High Volume)	3	3	2

**Control Based On Data Result:** The Mid Sine wave performed was detected the best overall, whereby both low and high volume trials reflected the best stats when compared to the low and high sine waves. The Mid sine's best performance occurred at the high volume levels of -10 dB as opposed to the low volume trials at -14 dB.

Average Ability to Detect Sound Based on Sensor Light Output by the Sound Detector:

	Low Sine	Mid Sine	High Sine
Average Low Volume	0.25	0.75	0
Average High Volume	3	3	1.5

Plot, Low Volume Trials: Average Ability to Detect Sound Based on Sensor Light Output by the Sound Detector:

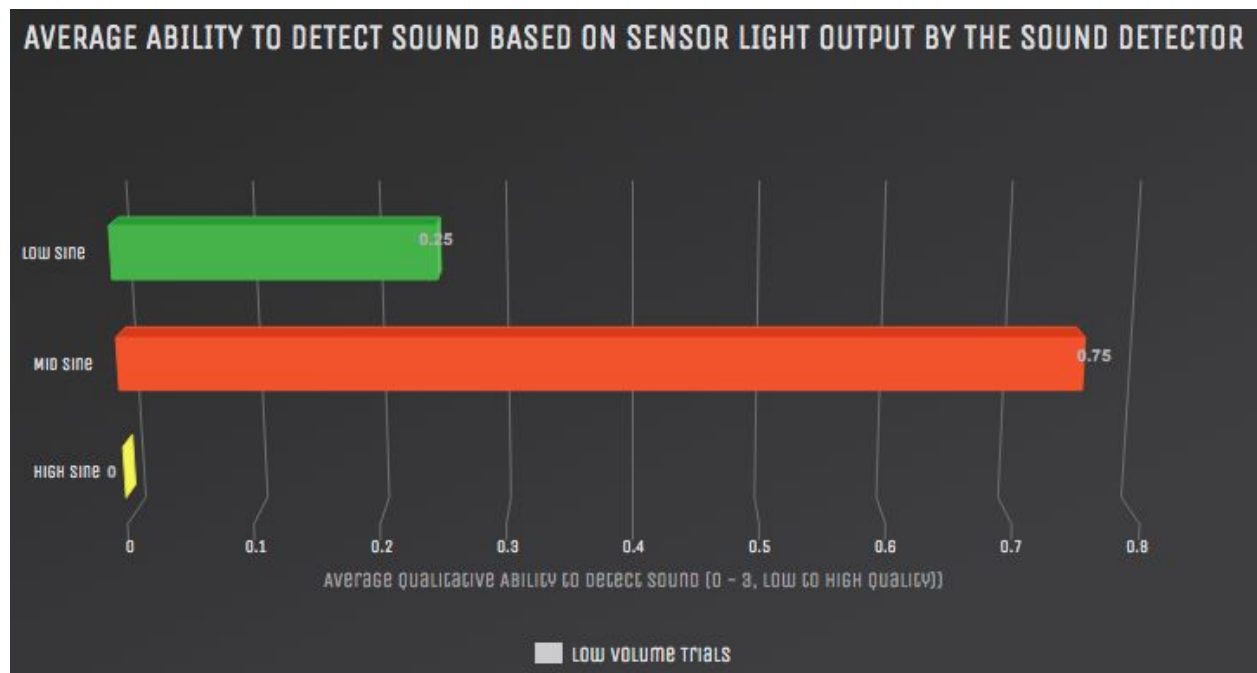


Figure: Graph of Low Volume Trial Averages for Ability to Detect Sound:

Plot, High Volume Trials: Average Ability to Detect Sound Based on Sensor Light Output by the Sound Detector:

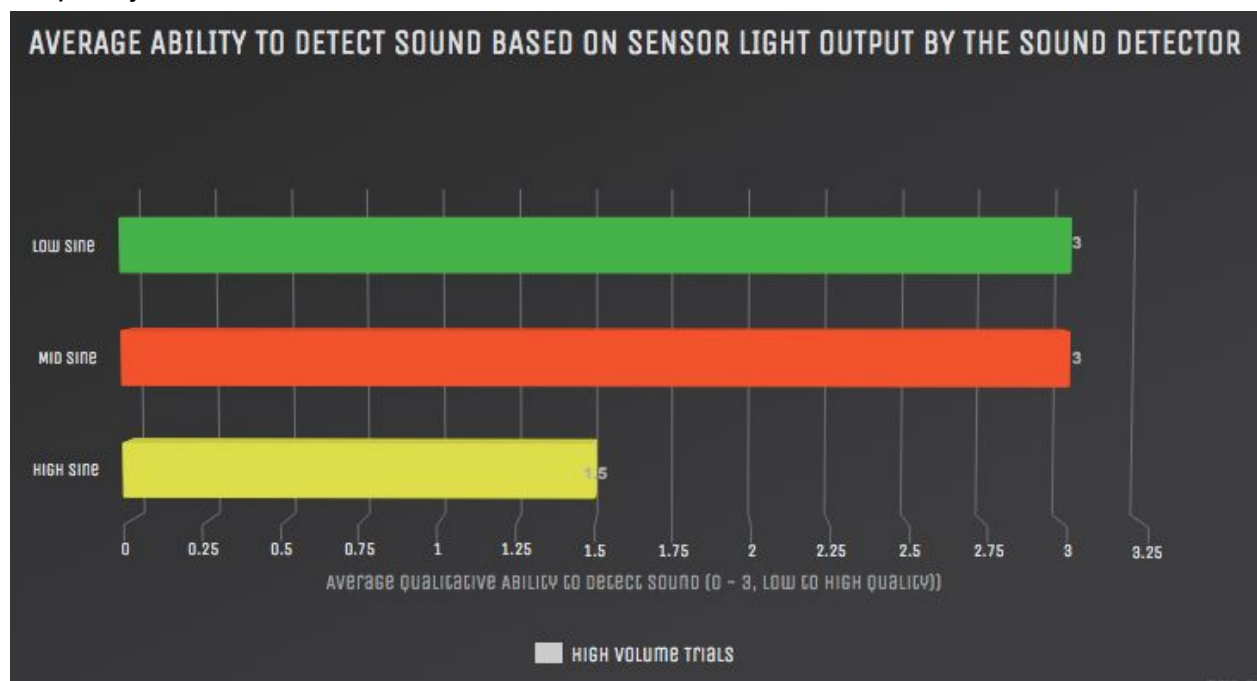


Figure: Graph of High Volume Trial Averages for Ability to Detect Sound:

**Accuracy and Precision:**

	Low Sine	Mid Sine	High Sine
% Error, Low Volume	91.67%	75%	100%
% Error, High Volume	0%	0%	50%

*Figure: Accuracy of Sound Detection Sensor based on Average Values Data:***Stability:**

	Low Sine	Mid Sine	High Sine
Stability Level	0	0	0

*Figure: Stability of Sound Detection Sensor During No Environment Noise:***Effectiveness and Efficiency:**

Efficacy of Sound Detection Over a Longer Sustained Note (2 Bars):

	Low Sine Wave	Mid Sine Wave	High Sine Wave
Experiment 1	2 (Several Blips)	3 (Sustains perfectly)	1 (Single Initial Blip)
Experiment 2	2 (Several Blips)	3 (Sustains perfectly)	1 (Single Blip)

*Figure: Efficacy of Sound Detection Over a Longer Sustained Note:*

**Sound Detection Sensor, Control Based Upon Data Result:** Here the Mid Sine Wave again displays it's dominance in being detected, whereby a 3 is the best rating and a 0 is the lowest rating.

**Sound Detection Sensor, Overall Result of Control Based On Data:** The mid sine wave performed at a high volume was best detected by the sound detection sensor for both eighth notes and 2 bar sustained notes played at the key of E2. Thus the mid sine wave allows for the most reliable output if reliability in positive and correct sound detection is considered to be the control law being examined.

## Data for Optimal Settings DS18B20 Temperature Sensor:

**Note:** For full descriptions of each experiment carried out in Sensor Lab # 2 please consult my previous lab submission for sensor lab 2. For the sake of Brevity, the in depth descriptions from sensor lab 2 have been omitted to avoid redundancy and to save time. My purpose is to use the data provided below to establish the “sweet spot set point,” or the optimal settings from my previous labs in order to adapt my lab 7 to be based upon the principle of control based upon data. Any new “control based on data,” sections within my data from previous labs will be highlighted. After these sections a section labeled “*Summarizing Control Based On Data, Finding a “Sweet Set Point” for Sound Detection Sensor and DS18B20 Temperature Sensor,*” will mark where the majority of new lab work begins for lab 7.

## Reliability and Consistency of Results:

	Trial 1 (°C)	Trial 2 (°C)	Trial 3 (°C)	Trial 4 (°C)	Average (°C)
Living Room	19.37	19.31	19.37	19.37	19.36
Breathe	23.56	24.00	23.62	23.43	23.65
Heat Fan (High)	21.75	21.75	21.93	21.81	21.81
Heat Fan (Low)	21.12	21.25	21.18	21.31	21.21
Thermostat	18.89	18.89	18.89	18.89	18.89

Figure: Temperature Sensor Output in Degrees C, Versus Thermostat Output:

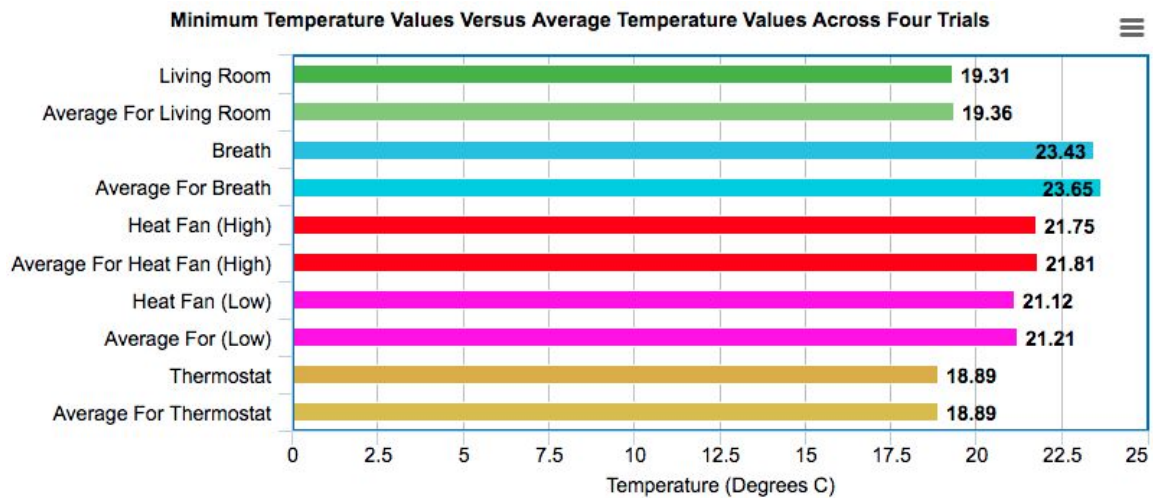
## Accuracy and Precision:

	Temperature (Fahrenheit)	Temperature (Celsius)
Average Living Room Temperature	66.83	19.36
Thermostat Temperature	66	18.89
Percent Error	-2.43%	-2.43%

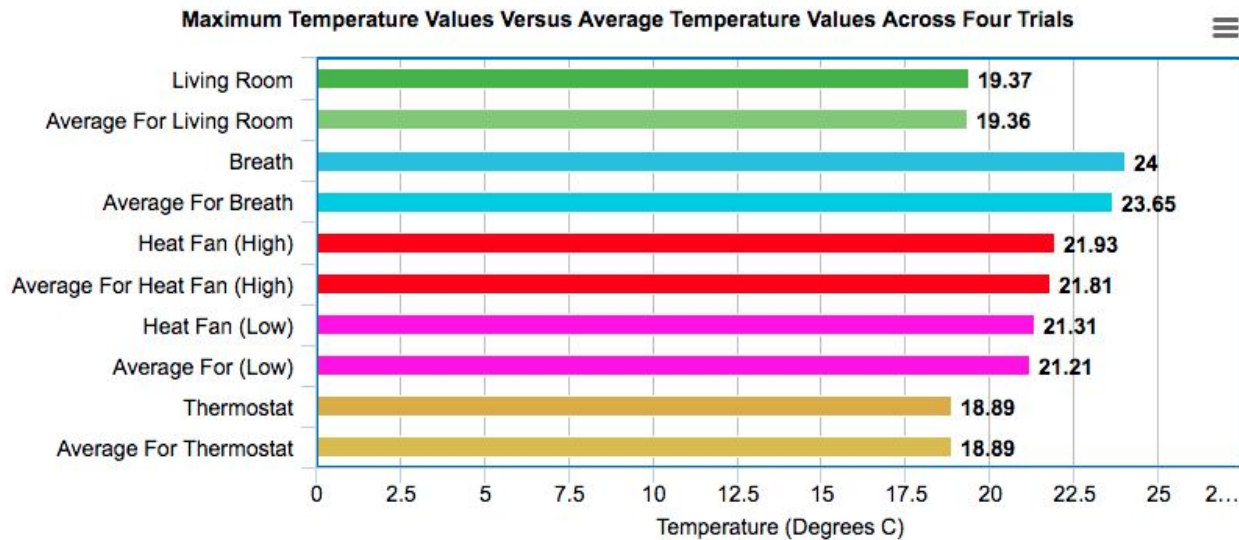
Figure: Percent Error Between Living Room Temperature Sensor Trial and Thermostat:

	Minimum (°C)	Maximum (°C)	Average (°C)	% Error Between Minimum and Average (%)	% Error Between Maximum and Average (%)
Living Room	19.31	19.37	19.36	0.26	-0.052
Breath	23.43	24.00	23.65	2.43	1.48
Heat Fan (High)	21.75	21.93	21.81	0.0028	-0.55
Heat Fan (Low)	21.12	21.31	21.21	0.43	-0.47
Thermostat	18.89	18.89	18.89	0	0

*Figure: Percent Error Between Maximum and Minimum Trial Values and Averages:*



*Figure: Minimum Temperature Values Versus Average Temperature Values:*



*Figure: Maximum Temperature Values Versus Average Temperature Values:*

#### **Temperature Sensor DS18B20, Control Based On Data Result:**

	% Error Between Minimum and Average (%)	% Error Between Maximum and Average (%)	Average of the Percent Error Using the Absolute Value of Two Extremes (Min and Max) of Each Data Group:
Living Room	0.26	-0.052	$(0.26 +  (-0.052) ) / 2 = 0.156$
Breath	2.43	1.48	$(2.43 + 1.48) / 2 = 1.955$
Heat Fan (High)	0.0028	-0.55	$(0.0028 +  (-0.55) ) / 2 = 0.5528$
Heat Fan (Low)	0.43	-0.47	$(0.43 +  (-0.47) ) / 2 = 0.45$

*Figure: Percent Error Between Maximum and Minimum Trial Values and their Averages, and Average Percent Error Using the Absolute Value of Percent Error for Both Extremes for Each Environment:*

**Temperature Sensor DS18B20, Overall Control Based On Data Result:** Here the living room's temperature data reflected the lowest amount of variance in data among its extremes as denoted by taking the average of the percent errors between the maximum and minimum extreme values in the experiments. This means that the Living room provided the most consistent data across each of its trials as compared to the other rooms. Thus the Living room appears to be the most optimal condition to achieve consistency in data.

## **Summarizing Control Based On Data, Finding a “Sweet Set Point” for Sound Detection Sensor and DS18B20 Temperature Sensor:**

(1) The Sound detection sensor performs optimally with a 14 dB (high volume) setting, using a mid sine wave when performing the note E2 at the speed of a 160 bpm eighth note or during a 2 bar sustained note.

(2) The living room temperature conditions worked most optimally in terms of least variance among multiple experiments and the extreme values (minimum and maximum) observed among the four trials for the multiple temperature environments, when compared to the average temperature of the four trials for each environment. Furthermore the living room temperature trials provide value in their ability to compare with the supposed exact temperature environment as determined by the thermostat settings.

### **Analysing Earlier Labs and How These Labs Can Help in Designing an Effective Combination of Simulink Models:**

My initial analysis relied on consistency in pulses being the same exact signal every time for the sound sensor. However, to improve my design and combine my sound temperature design with a temperature sensor, I could use methods of numerical analysis. There are numerous ways to improve my initial analysis and experiments to create a system that incorporates a temperature sensing simulink and sound detection system; thus I will discuss a few of these ways. Since in this experiment I intend to emulate a very basic sound detection sensor to intake a sound or vocal command in order to receive data back from my sound sensor through a monitor reading.

The idea is to say something like “hey Alexa, what is the temperature,” or “hey Siri, what is the temperature,” since Alexa and Siri are popular vocally commanded sound bars. With these parameters as my intention for experimentation one would need to consider what exactly occurs in a system that performs these functions. First a signal is entered as a vocal command, next that signal is compared to a list of possible commands and if that vocal or sound resembles that command enough then that command is outputted. So in terms of my temperature reading experiment via a vocal command, a good way to utilize methods of numerical analysis and machine learning might be to have a predetermined, or preprogrammed command signal, which allows for temperature output, then compare that signal to a user signal.

Since my sound detection sensor doesn’t read volume, dynamics, pitch tone etc, it seems viable to compare pulse waves; given the output is either a one or a zero based on input into the sensor. So, specifically one could integrate the command signal and compare the area of the command signal to the integrated area of the user input signal. Then if this signal falls within a particular percent error tolerance, such as 5% to



10% difference or less, of the original signal's area one could output the data. The issue with this method involves the timing of each pulse when compared with total read time from the sound detection sensor.

To elaborate, if the original signal is 5 seconds long and your incoming signal is read for 10 seconds, yet the whole ten seconds of integration is read and falls within an integration area tolerance of what should normally be a 5 second long command then maybe the user didn't actually intend to activate this command. Other issues with this methodology include an earlier discussed fact, that is to say this sensor doesn't actually detect volumes, pitches and so forth; thus many sounds in general will likely match the pulse settings of a particular command and be incorrect.

A way to alleviate some of the aforementioned issues is to use case switching which could mean that depending on a particular input time being different, maybe other commands are compared to first. Another way to resolve some issues could be to perform a series of different pulse integrations over chunks of time in order to deal with how a person might space out their words and silences, and the final series could be added up. Additionally, when comparing the signals one could perform root means squares of signal to help detect differences between those signals.

Thus, as mentioned earlier, in an ideal sound detection system that receives an input command, compares it to a signal command and outputs a temperature command on a monitor, one could incorporate multiple case systems for comparison and output. Overall, since not everybody says a vocal command the same way, at the same speed, with the same intonation every time, case switching between accepted preprogrammed temperature output commands would likely allow for the most ideal system design, in order to capture the most amount of correct input circumstances.

**1. Analyze all the data you have taken in #4 and #5 and for each parameter and determine a “sweet spot set point”. That means you will be designing “control based upon data” as opposed to “setting control based upon a pre-established set-point”.**

See Previous data sections, which provide descriptions of optimizations and optimal settings for each experiment. Also See analysis initial question which discusses how to improve previous experiments.

**2. Argue the efficiency and effectiveness of your final design**

**a. Use a Simulink model to help you prove your final design**

My final design appeared to be effective, as it properly displayed the sound signal percent error tolerance levels when above the threshold. Additionally it, for the most part, provided the correct output for signals not in the tolerance threshold, with accuracy increasing the higher the percent error for the difference in signals; thus operating as intended.

**b. Use statistics to help as well**

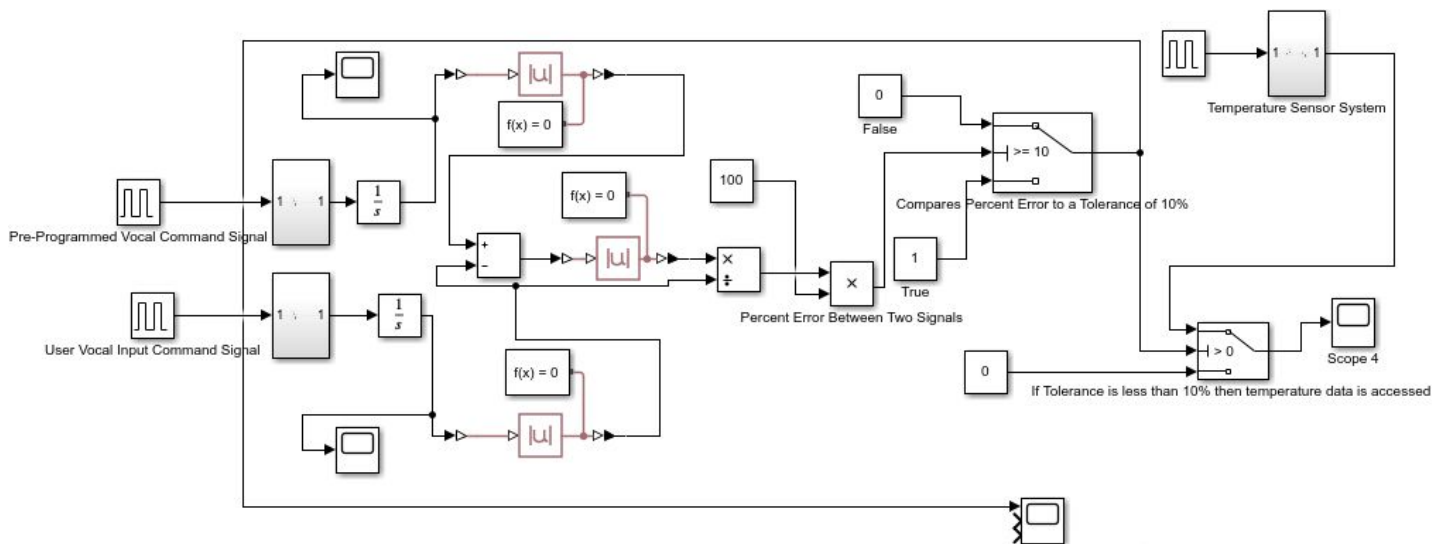
Percent error and integration of signals allowed me to compare two waveforms and determine if they were similar, while setting a threshold to allow for a certain similarity eventually resulting in temperature data requests from a switch.

**3. This means that you will take #4 and #5 labs and make a final combination lab #7**

**Description of Final Combined Simulink Model:**

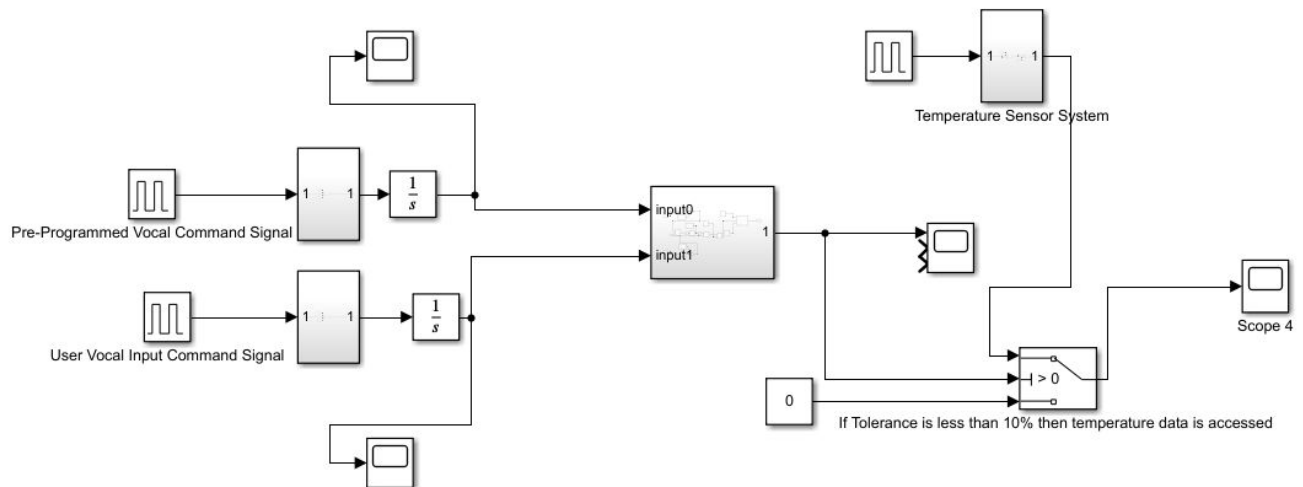
From my previous analysis section I decided to implement 2 sound detection sensors (as opposed to just 1 in previous experiments). Next I compared the signals via an integration over a 10 second period (however the integrator models used were continuous so a static total of area under each signal is not used). I compared the two signal areas (provided via signal integrations) using a percent error algorithm. Next a switch is used to determine if the signals are within 10 percent of each other's percent error. If the two signals are under 10 percent error from each other then a 1 (representing truth) is output, otherwise a zero (representing false) is output. Next based on the truth statement switch output a very basic temperature sensor module is either activated or not based on the truth statement. Here the temperature sensor detects temperature between three different values and returns the correct value. Here the values consist of the original signal, or a 1, or a zero. The overall idea behind this model is that a user says a command such as "hey siri, what is the temperature," and the temperature is returned. Here the original signal is compared to a preprogrammed command signal and determines if the user's voice resembles the command and then allows for a temperature reading command to either be output or not based on similarity from integrating the signals and comparing them.

## Simulink Combination Expanded Model:



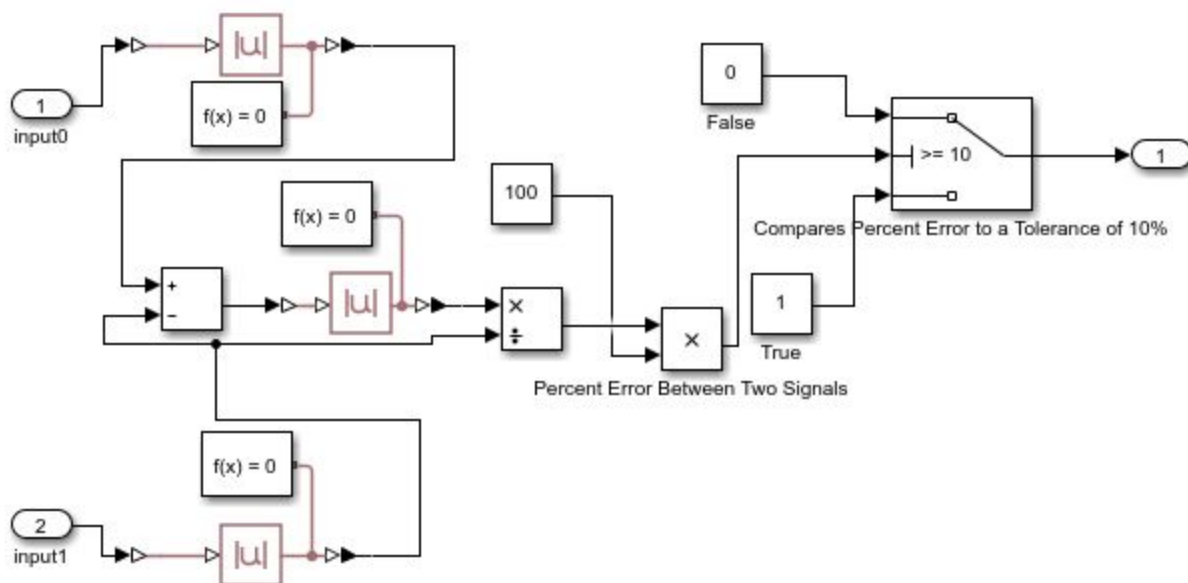
**Description:** The intention was for this model to take an input signal, and compare that input signal to a preprogrammed accepted command signal. In performing this comparison the intention was to integrate and find the entire area under each signal with amplitudes 0 to 1. The idea is that a set of pulses replicates the human voice saying a string such as “hey Siri, what’s the temperature,” then that set of pulses is compared, using integration of the signals, to a preprogrammed set of pulses, whereby if the percent error difference between the two signals during the time period is less than 10% then the temperature program is activated. The temperature program compares the original signal for temperature to establish an output of 1 of 3 values, the original signal, a lower signal based on a switching comparison statement, or a larger signal based on a switching comparison statement. In real applications this comparison of the input to preprogrammed serial readings, in order to achieve an output, would likely need a larger number of comparisons to occur than just 3 comparisons. Thus if the temperature limits, based on the datasheet for the sensor, went from -55.00 degrees C to 125.00 degrees C, there might need to be at least 18000 comparisons if there were 180 total values with 2 additional decimal place significant figures for sensing temperature.

## Compressed Version of Aforementioned Simulink Model:



**Description:** Here the Simulink model has a compressed substem to represent the percent error algorithm inside of it. The percent error algorithm receives the continuous integrals over 10 seconds as inputs from the integral blocks. Next the percent error formula converts those integrals into absolute values, just in case the integrals for some reason result in a negative value due to any negative amplitudes (although, I did not feed any negative amplitudes into my pulse generators). Next after the subtraction in the percent error equation, the output value is converted into an absolute value

## Subsystem Model For Percent Error Between Two Signals, Utilizing a 10% Tolerance Switch Based upon Percent Error for the Two Signals:



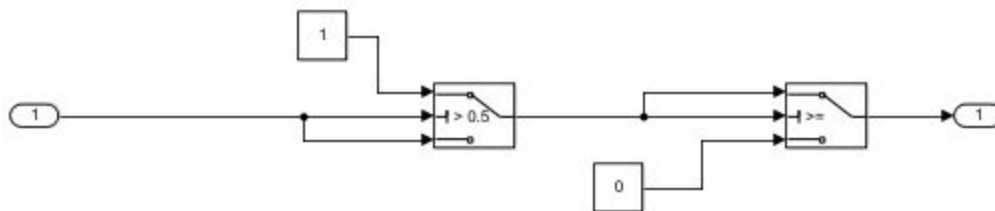
**Description:** Handles the Percent Error function and the true or false percent error 10% tolerance switch.

### Percent Error Algorithm Displayed In Part of the Percent Error and Tolerance Subsystem:

$$\text{Percent Error} = \frac{|(\text{Approximate} - \text{Exact})|}{\text{Exact}} * (100)$$

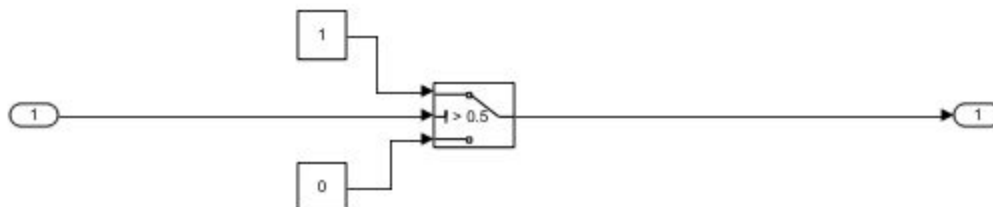
**Description:** The algorithm above displays percent error. However, it is important to note, this algorithm doesn't include the additional switching mechanism listed in the Percent Error and Tolerance Subsystem above. This only shows a portion of the subsystem above displayed in numerical terms.

### Temperature Sensor Subsystem:



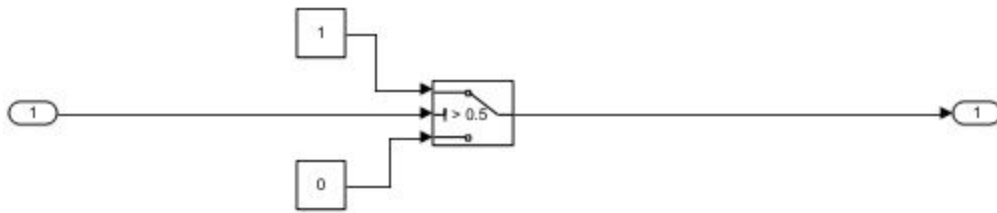
**Description:** This compares a pulse signal of 50% for 10 seconds with a particular amplitude (in this case 0.6) to a threshold of 0.5 to either output a constant of 1 or the original signal (if threshold is equal to the signal or less). Next the second switch compares the value to 0.5 again to decide if the original input signal is 0, 1, or whatever the original signal was (thus denoting the “=” branch). This means the final output is either 0, 1, or the original signal.

### Sound Detection Subsystem 1 (Pre programmed sound signal):



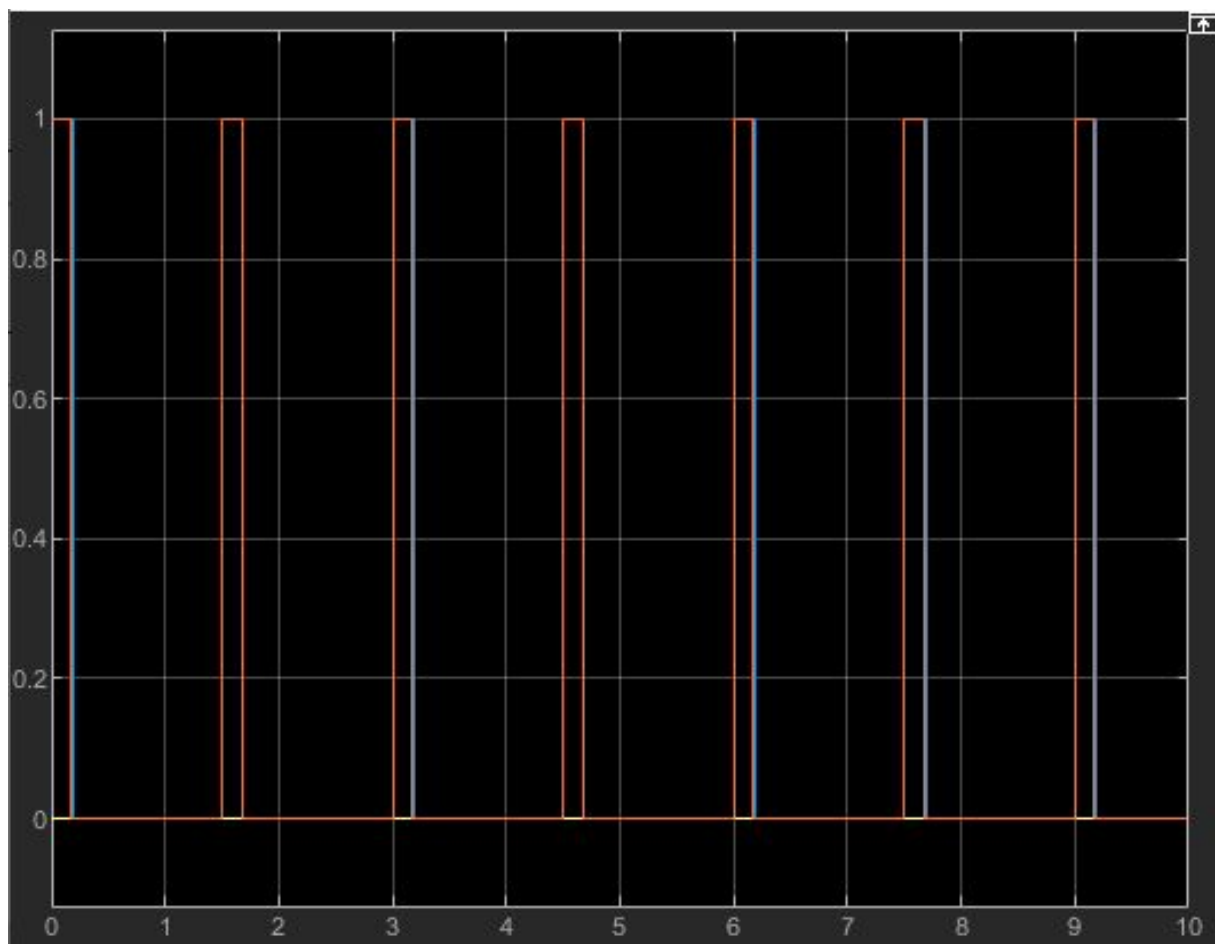
**Description:** This is a basic sound detection model that outputs a signal if a sound is detected. Here if the input amplitude is greater than 0.5 then the result is a constant one fed out; otherwise a zero is fed out.

### Sound Detection Subsystem 2 (User Input sound signal):



**Description:** This is a basic sound detection model that outputs a signal if a sound is detected. Here if the input amplitude is greater than 0.5 then the result is a constant one fed out; otherwise a zero is fed out.

**Input Pulses Going Into The Sound Detection Subsystem (Pre-program Pulse = 12.5%, User Input = 11.4%):**



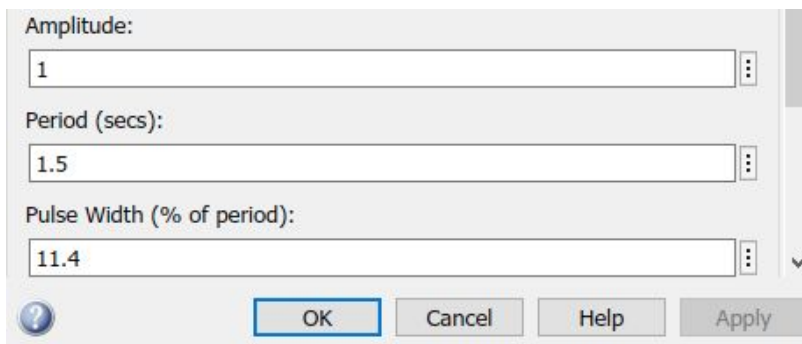
**Description:** Here the two input pulses are within 10% tolerance of each other. The preprogrammed command pulse occurs with 12.5% width, while the user input occurs with 11.4 % width, resulting in a percent error of 9.649%.

### Output From 10% Tolerance True or False Switch Statement For Input Pulse 11.4%:



**Description:** The pre-programmed exact input pulse, which results in the voice command being accepted is compared to over the course of 10 seconds. Here the user input vocal pulse is at 11.4% in the input pulse settings. For perspective the preprogrammed input pulse settings are 12.5% in the menu. This means the percent error here was 0.965% for the overall area.

### User Input Pulse Width Settings Image for Context:



Amplitude:  
1

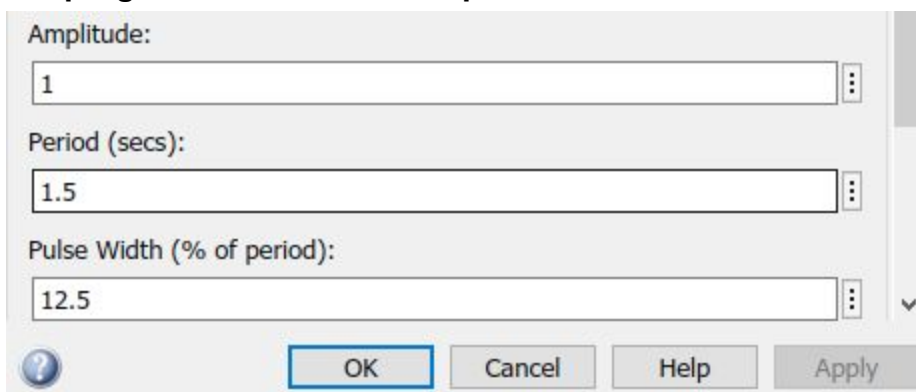
Period (secs):  
1.5

Pulse Width (% of period):  
11.4

? OK Cancel Help Apply

**Description:** Here are the user input pulse width settings for a single test of the system. These are the pulse width settings from the user input for the sound signal section.

### Pre-programmed Command Input For Pulse Width for Context:



Amplitude:  
1

Period (secs):  
1.5

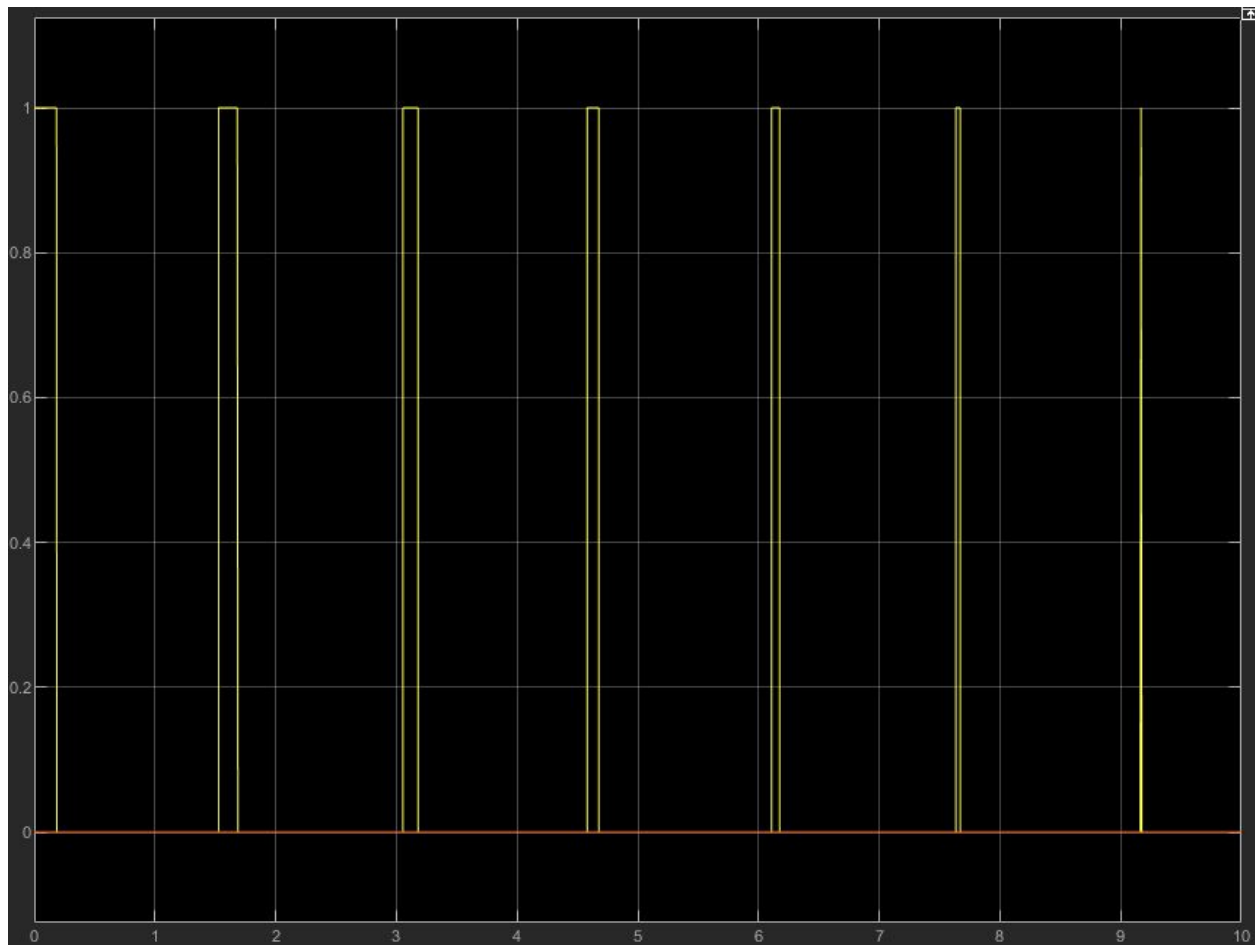
Pulse Width (% of period):  
12.5

? OK Cancel Help Apply

**Description:** Here are the pre - programmed command signal input pulse width settings for all system tests. These are the pulse width settings from the pre programmed command signal input for the sound signal section.



**Output From 10% Tolerance True or False Switch Statement For User Pulse Width Input of 11.2 % (Percent error is 11.607%):**



**Description:** Ideally none of these other pulses would occur after the percent error tolerance switch in the simulink model, however, since my model integrates total area, I believe these spikes are showing brief changes in the percent error, as time runs, rather than as a discrete, constant or static total. Here, for the most part the desired outcome was achieved, however, further algorithms and experiments could perhaps improve these results so that a 0 is reflected across the full 10 seconds instead of spikes and a trend of 0 for the majority of the 10 seconds. Additionally, for the sake of brevity and to avoid redundancy I will avoid posting every pulse width menu setting picture.

**Output From 10% Tolerance True or False Switch Statement For Pulse Width at 10% (Percent error 150%):**



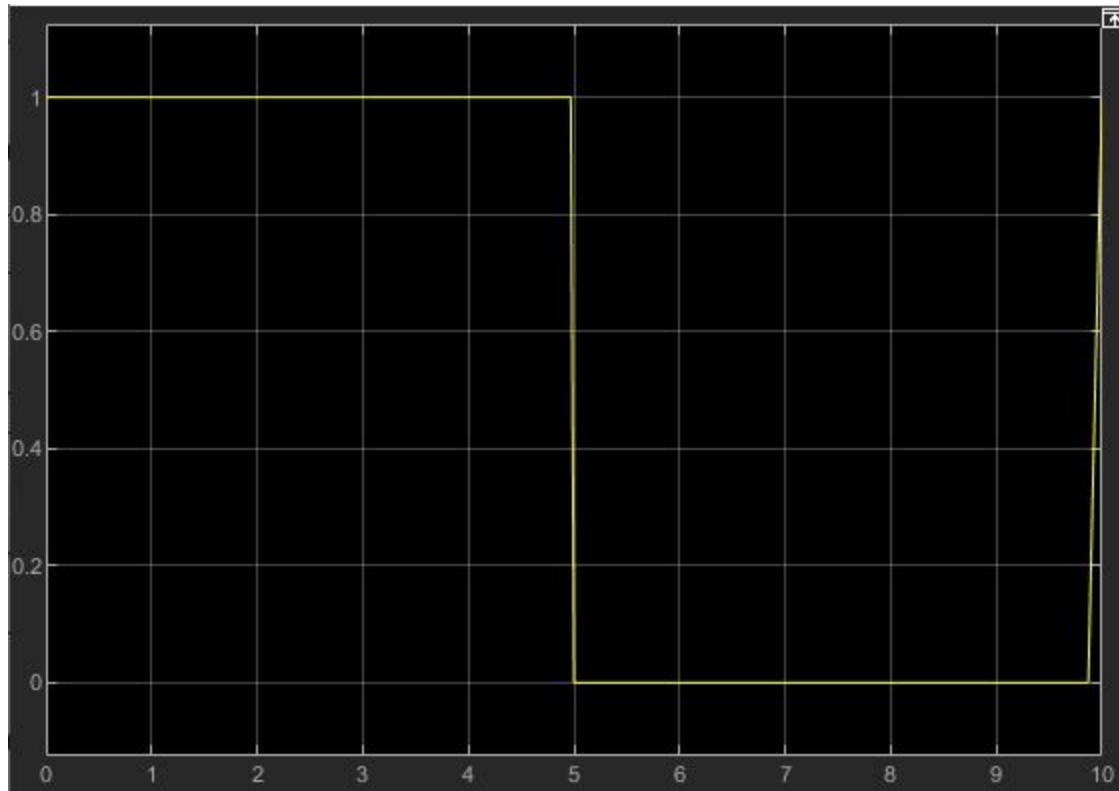
**Description:** Output across the 10% tolerance switch when the input pulse is 10% width. Here the percent error is 25% when compared to the 12.5% pre-programmed pulse. This output is very optimal, where the tolerance switch is very briefly bypassed by a short initial signal from a percent error calculation within tolerance very briefly.

**Output From 10% Tolerance True or False Switch Statement For Pulse Width at 1% (Percent Error 1150%):**



**Description:** Here, at a 1% pulse width we experience very minimal output of signal across the 10% tolerance switch, which is optimal. There is a slight transient for a very brief amount of time, as denoted by the slight yellow line.

### Temperature Sensor Output Across Final Switch From Pulse Width at 11.4% or Greater:



**Description:** This wave is output exactly as desired. Here the temperature sensor receives a high input pulse of 50% for 10 seconds. Thus this output matches the temperature sensor when the 10% tolerance switch is met from the percent error calculations from the sound detection signal integrations and comparisons. This means that the sound detection signal comparison model is behaving correctly when an input pulse of 11.4% is compared to a pulse of 12.5%, thus resulting in less than 10% for the percent error switch threshold, which means a true signal is sent, allowing the sound sensor signal to pass through a switch and to start processing a comparison to display a basic set of 1 of 3 different temperatures based on the signal sent into the sensor.

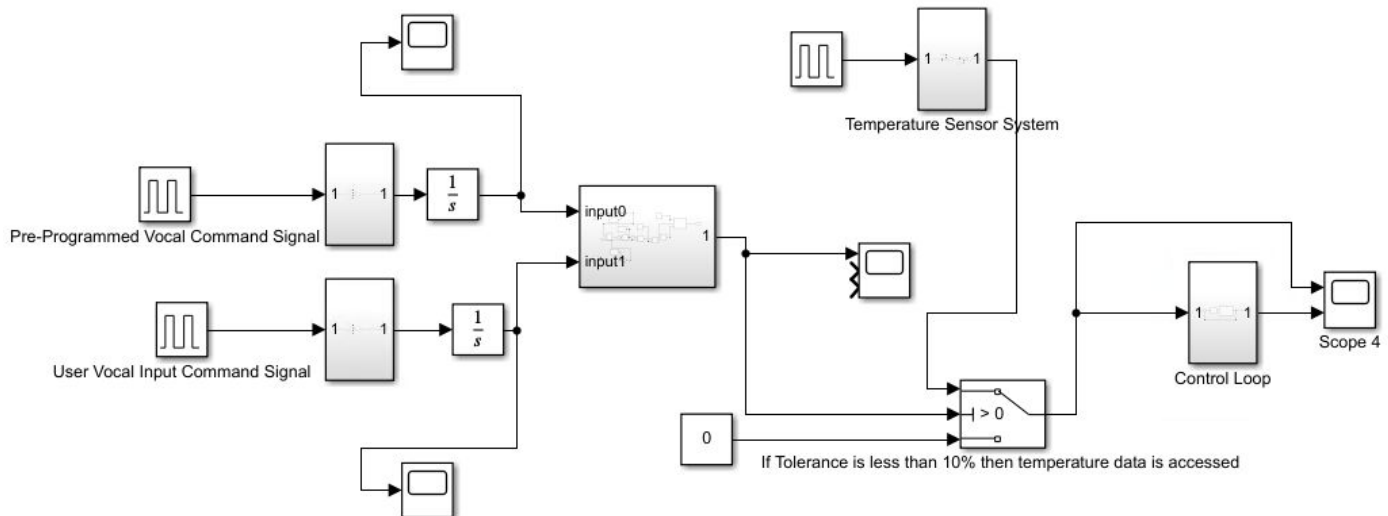
### Designing Based on Analysis of the Data:

The most optimal outcomes for my simulink model occurred when the user input was at 10% pulse width, or lower for non-operating commands, or a signal from the user that did not fit a 10% error tolerance and outputted a reasonably long and accurate value of 0, while values of 11.4% pulse width, or higher resulted in a seemingly perfect steady signal of 1; thus resembling an accurate pickup of the preprogrammed voice command. Therefore in performing future controller tests values within these extremes will be utilized to measure data.

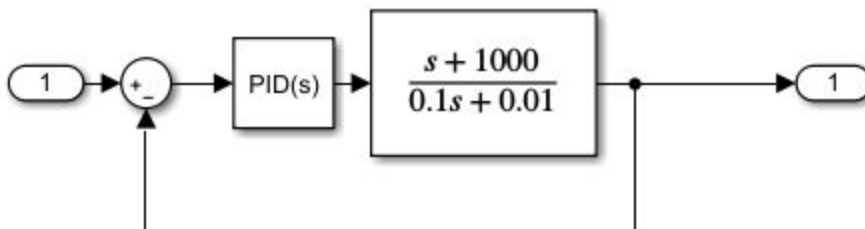
## Simulink With a PID and Plant Function Introduced:

Based on the previous data I ran my simulink model with a PID controller and plant function while using the two extremes of my optimal data.

### Simulink Model With Control Loop Subsystem, PID Controller, and Plant Function:

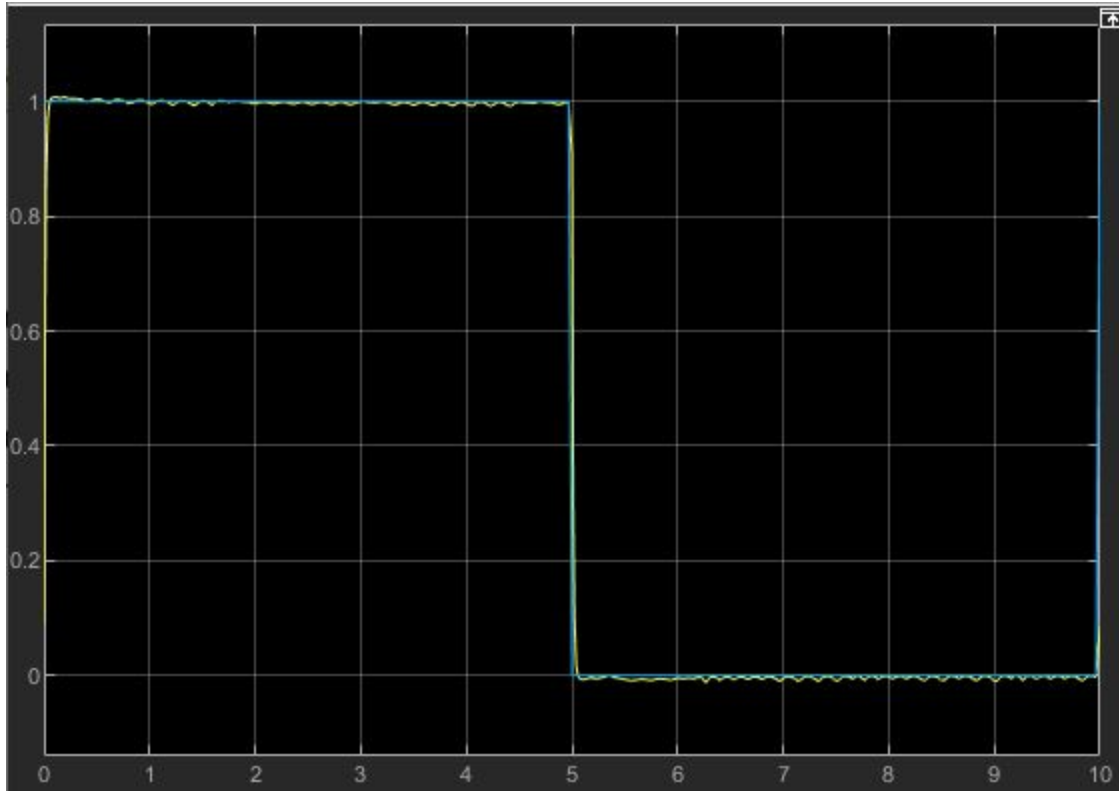


### PID Control Loop Subsystem Simulink Model:




**Description:** Here the Temperature Simscape system is fed through the switch when the sound detection switch receives under 10% error tolerance. I attached a PID controller and a plant function while emulating the original signal passing into the sound signal output. The Sound signal output also receives the original sound signal before passing through the PID controller and plant function, for comparison to the output signal.

**Output of the Control Loop, PID Controller With a Plant Function in Comparison to the Original Temperature Signal:**



**Description:** Output plot demonstrating the optimal settings to emulate original signal entering the control loop. Here a bit of noise can be seen in the output as compared to the original signal. This displays 2 outputs, (1) the original signal, and (2) the signal utilizing optimal PID Controller and Plant Function settings in order to emulate the original signal.

## Transfer Function Settings for Control Loop:

 Block Parameters: Transfer Fcn ✕

**Transfer Fcn**

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.

**Parameters**


Numerator coefficients:

Denominator coefficients:

Absolute tolerance:

State Name: (e.g., 'position')

## PID Controller Settings for Control Loop:

 Block Parameters: PID Controller >

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID Form: Parallel

Time domain:  
☒ Continuous-time  
☐ Discrete-time

Discrete-time settings  
Sample time (-1 for inherited):

▼ Compensator formula

$$P + I \frac{1}{s} + D \frac{N}{1 + N \frac{1}{s}}$$

Main Initialization Output Saturation Data Types State Attributes

Controller parameters

Source: internal

Proportional (P):

Integral (I):

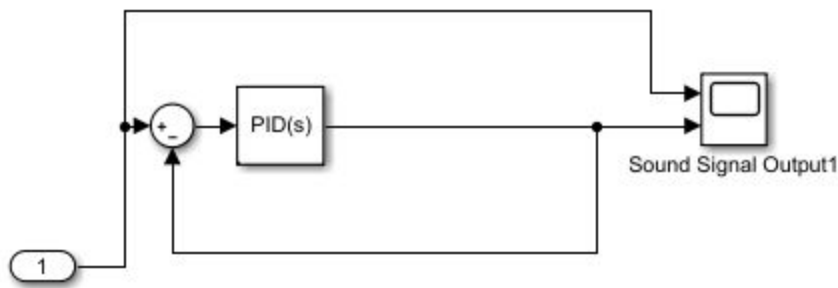
Derivative (D):

☒ Use filtered derivative

Filter coefficient (N):

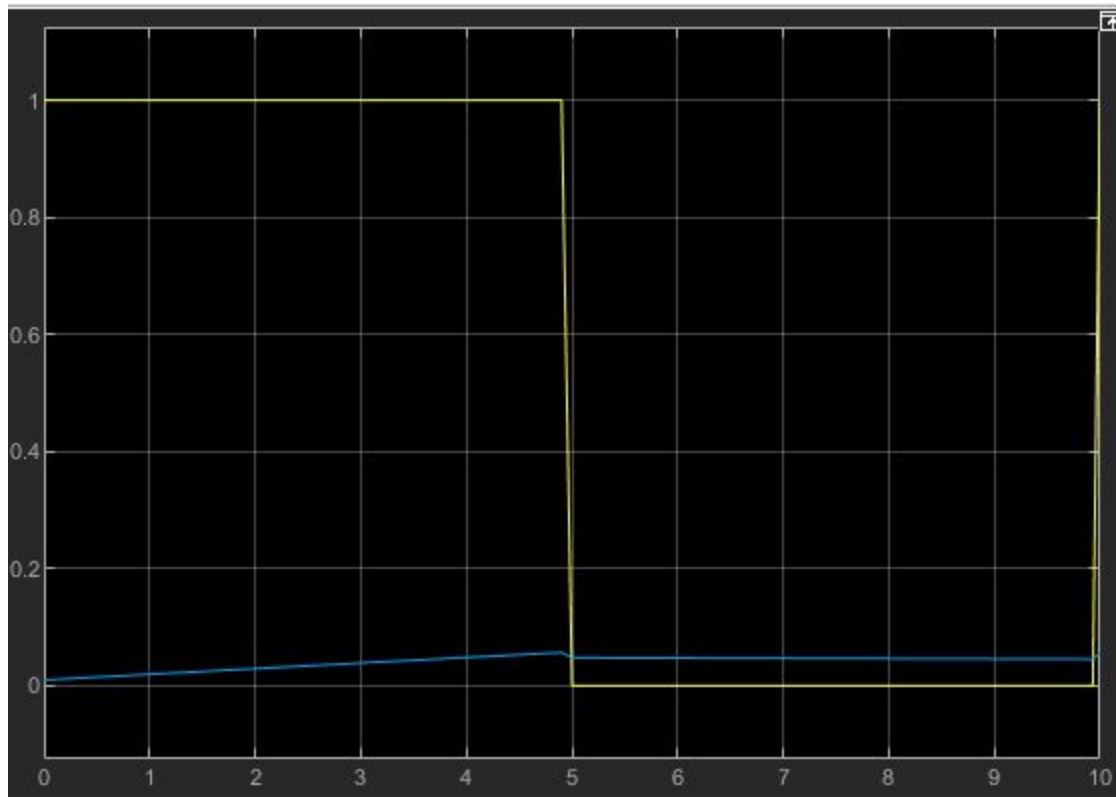
**Description:** Above are the PID controller settings and plant function settings used to closely emulate the initial input into the control loop system.

### Simulink Subsystem Model without Plant Function in Control Loop Subsystem:



**Description:** Here I show a new subsystem replacing the old control loop subsystem. This new subsystem forgoes the use of a plant function in the control loop.

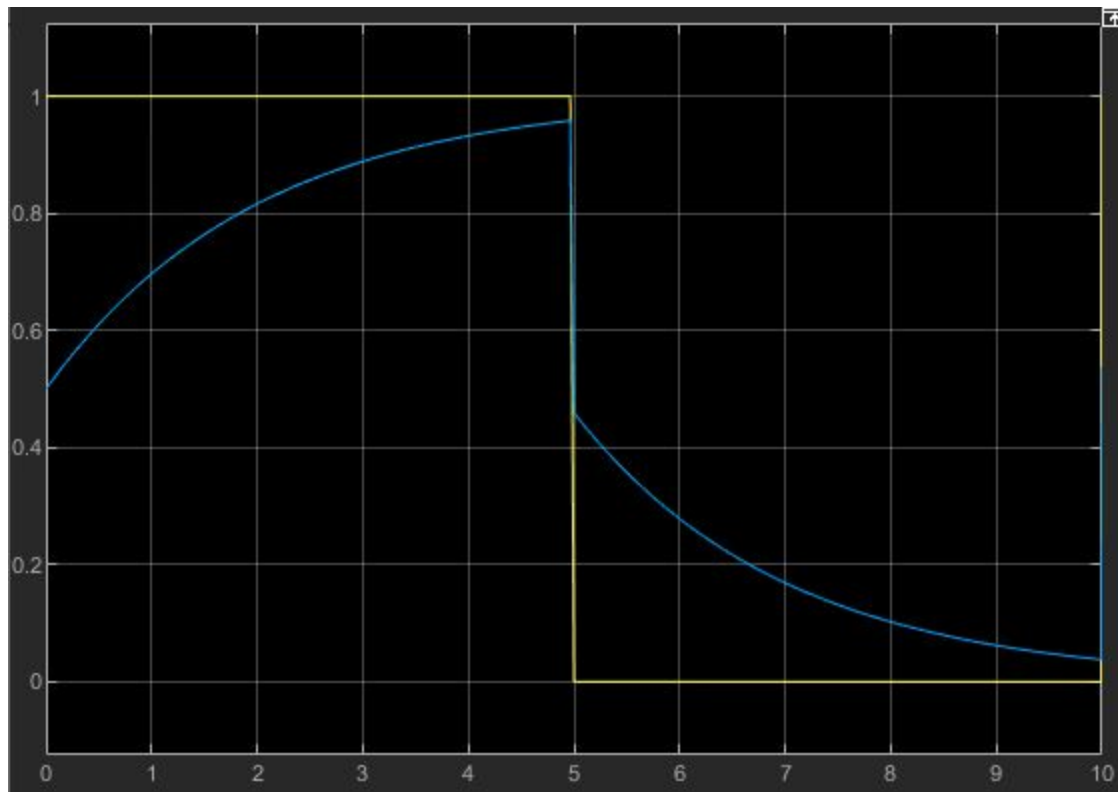
### Plots of Control Loop With PID Controller and Without Plant Function (at PID Settings $P = 0.01$ , $I = 0.01$ , $D = 0$ ):



**Description:** Here, the settings for my PID controller remain the same as when I had the plant, however the output signal varies because there is no plant in the control loop in this plot. There is a dramatic difference in the resulting blue waveform as compared to the control loop with the plant function.



**Plots of Control Loop With PID Controller and Without Plant Function at Initial Settings (at PID Settings  $P = 1$ ,  $I = 1$ ,  $D = 0$ ):**



**Description:** Here is the result of my simulink model without a plant function in the control loop. Additionally the PID controller settings have been reset to stock values in order to display the original output on initialized settings.

**Comparing the “Control Loop With a Plant Function” to the “Control Loop Without the Plant Function:” (Optimal Control Loop Settings for Achieving an Output Similar to the Input Signal of the Control Loop):** The plant function allowed for me to better emulate the original signal with less error. When using a PID controller without the designated plant function the output wasn’t nearly as similar to the original input signal in both the case of refined PID settings and in the case of stock, or, “initialized,” settings. The Plant function performed significantly better in emulating the original control loop input signal.

## **At the end of your design, answer these questions:**

### **1. Will my results be changed by scaling?**

My overall results would change due to scaling. In combining the two sensors my initial sound detection outputs stayed the same based on my input signals which were chosen and optimized based on data. However, my original sound detection experiment was slightly changed to reflect a way to optimize sound detection to activate and receive a reading from a basic temperature sensor model. The overall output from my sound detection model was a percent error that resulted from two signals, which eventually led into a true or false statement used to activate the temperature sensor. This means that scaling made the output from the sound detection systems model very different than it was in previous labs.

The sound detection system however remained very similar or the same as the previous experiment, while the sound detection model became much more complicated. However, at the end of the output of the temperature system, the sound signal had a switch to temper when the temperature sensor would occur, which did ultimately affect the end output of the system and the temperature sensing comparison system. Here, at the end of the whole system a PID controller and plant function were added. The settings were optimized for this control loop to achieve an output signal consistent with the pulse wave input plots provided by the temperature system being triggered by the sound system.

### **2. How would I change or improve my initial analyses to help me make a good design?**

To improve my initial analysis, which analyzed lab 4 and 5 and gave context to some of my ideas for creating a system with multiple sensors, I might improve my system so that it resets summation of a signal if it is invalid to the 10% error threshold. The idea here would be that if a signal doesn't demonstrate the pulses representing the vocals "hey siri, what temperature is it," quick enough, it resets so that at any moment the user can then once again begin having a new signal integrated, that doesn't involve old useless pulses that don't contribute to achieving the pre programmed command signal.

### **3. Can I defend my design?**

#### **a) Demonstrate your design statistically using your data.**

From earlier contributions to this paper, the integration process appeared to work correctly allowing for very little temperature reading signal to be processed. When an 11.4% pulse width was compared to a 12.5% command pulse width, this value was about 9.65% error, which allowed it to fully express a temperature signal for 10 seconds, because the percent error was in the 10% tolerance range of the switch. Additionally since not many pulses made it out at 10% or greater of sound comparison

tolerance, and since the majority of signal allowed became 0 for the temperature signal, it seems like this design was particularly effective for a basic design involving sound detection and temperature sensing output.

**b) Demonstrate your design using Simulink.**

See the simulink sections above for further detail.

**c) Other**

Overall, the design seemed very effective in detecting pulses based on a second run time and a 12.5% command controlling pulse for comparison to other pulse widths.

**4. Explain how the understanding of numerical methods and machine learning can help me design better control?**

The more I understand about numerical methods and machine learning the better equipped I am to understand the algorithmic and functional modeling required to represent whatever system, control law, or function I'm modeling. From what I understand many functions used to represent populations, genetics and so forth can be broken down into multiple and intricate algorithms, sometimes updating in real time, adapting to certain measures, functions or algorithms as well, all with the intent of modeling a system to the best abilities of the control analyst. So, as discussed in the Initial Analysis Refinement Section of this paper, a better understanding of machine learning and numerical methods could allowing for multiple algorithms to model the numerous ways a person could input a sound signal command of "hey Alex, what is the temperature," or, "hey Siri, what's the temperature," while allowing for the command to be recognized as a preprogrammed command and outputting the correct functionality of displaying the temperature for the user.

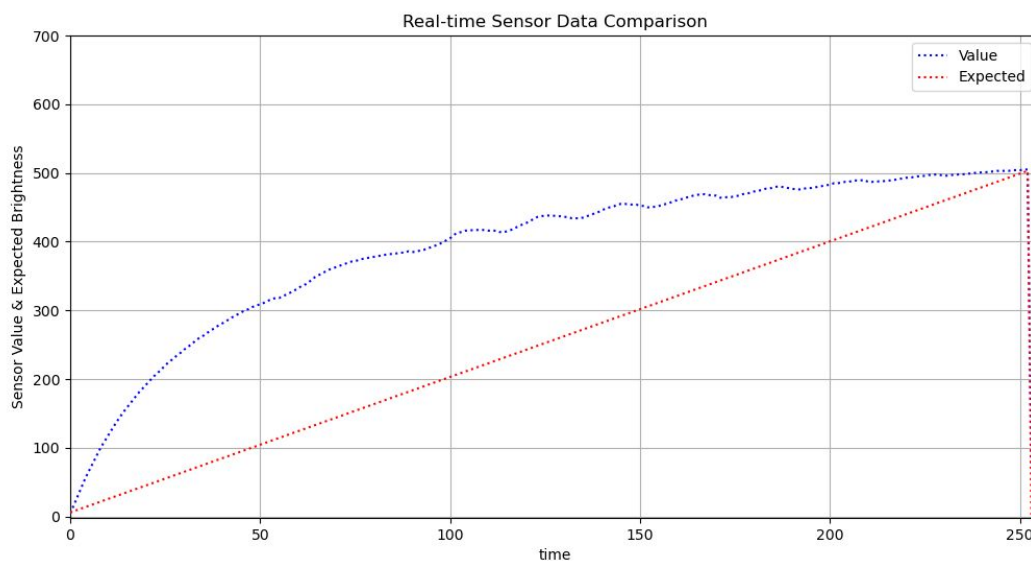
# Dual Ultrasonic Sensor

## Test Run by: Anthony Stehr

### Overview Lab 4:

In this experiment, I placed the Photoresistor sensor in a dark box along with an LED. This was accomplished by first taking a measurement with the sensor when the LED was at full brightness and 0 brightness to scale the output, then slowly increasing the voltage to the led from analog values 0-255 and measuring the output from the sensor. An expected result was then created to show what the data would look like if it was perfectly linear as the LED brightness increased linearly. Using the max and min values to scale the read data and expected data, then plotting both data sets in Python, a visual comparison was created to show the linearity of the response of the data from the sensor. Below it can be seen that the sensor has a logarithmic response instead of a linear one, being more sensitive at low values and less sensitive as the brightness increases.

### Measured Value vs Expected Value



Sweet Spot: Here the sweet spots appear to be at the minimum and maximum values due to the logarithmic response of the sensor.

## Accuracy and Precision

```
average percision = 0.00001230
```

```
average error = -133.58
```

Sweet Spot: This shows that the measurements are precise, but average error(accuracy) is not desirable because of the logarithmic response of the sensor confirming that the sweet spots are at the max and min values.

## Transient Response(sensitivity & stability)

### Full brightness to 0 brightness(state change)

```
00ms : 501
10ms : 357
20ms : 271
30ms : 214
40ms : 173
50ms : 144
60ms : 122
70ms : 105
80ms : 91
90ms : 80
100ms : 71
110ms : 63
120ms : 57
130ms : 51
140ms : 46
150ms : 42
160ms : 38
170ms : 35
180ms : 32
190ms : 29
200ms : 26
210ms : 24
220ms : 22
230ms : 20
240ms : 18
250ms : 17
260ms : 16
270ms : 14
280ms : 13
290ms : 12
300ms : 11
310ms : 9
320ms : 8
330ms : 8
340ms : 6
350ms : 5
360ms : 5
370ms : 4
380ms : 3
390ms : 3
400ms : 2
410ms : 1
420ms : 1
430ms : 1
440ms : 0
450ms : 0
460ms : -1
```

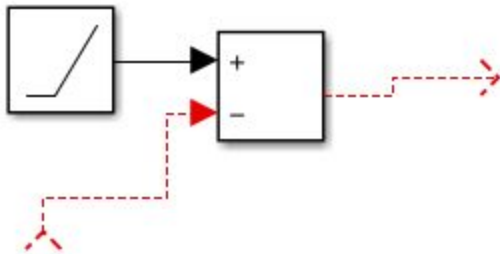
### 0 brightness to full brightness(steady state)

```
00ms : -3
10ms : 374
20ms : 448
30ms : 471
40ms : 482
50ms : 488
60ms : 491
70ms : 493
80ms : 495
90ms : 496
100ms : 497
110ms : 498
120ms : 498
130ms : 499
140ms : 499
150ms : 499
160ms : 499
170ms : 500
180ms : 500
190ms : 500
200ms : 500
210ms : 500
220ms : 500
230ms : 500
240ms : 501
250ms : 501
260ms : 501
270ms : 501
280ms : 501
290ms : 501
```

Sweet Spot: From this test, it showed that there is a 450 ms period until a state change is fully recognized and a 100-170ms period to reach a steady state. This means that the sweet spot for a sample frequency needs to be < 225 ms to satisfy the nyquist frequency principle. The sweet spot for measurement without the effect of a transient is then 100-170ms

### Simulink Model of Photoresistor Sensor

The idea behind this simulink model was to closely show how the Arduino code showed how far the light sensor strayed from a linear response. A ramp function was created with a slope equal to  $((\text{max sensor output} - \text{min sensor output})/255)$ . Using a difference operator between the input of the sensor and the ramp function, a variance from perfect linearity could be found. Putting all of this into a subsystem model allows for a simple input and output for the device.



## Overview Lab 5:

The purpose of this experiment was to test the blindspots of the Ultrasonic Sensor. Since this sensor works by emitting frequencies from a directional source, it would make sense that the sensor can only be accurate to a certain outward radial angle. Preparing for this test, radial distances from the sensor were plotted at 3,6, and 12 inches at angles 0,  $\pm 15^\circ$ ,  $\pm 30^\circ$ , and  $\pm 45^\circ$  with  $0^\circ$  drawn perpendicular to the sensor's face, and the positive angles rotating clockwise from 0 and negative angles rotating counterclockwise from 0. Using a solid object with an inch width was used to test the distance reading from each location.

### Distance test at 0 degrees

3.12 inches measured	6.02 inches measured	11.91 inches measured
3.00 inches expected at 0 degrees	6.00 inches expected at 0 degrees	12.00 inches expected at 0 degrees
0.12 distance error	0.02 distance error	-0.09 distance error

### Distance test at +15 degrees

3.05 inches measured	6.08 inches measured	11.98 inches measured
3.00 inches expected at 15 degrees	6.00 inches expected at 15 degrees	12.00 inches expected at 15 degrees
0.05 distance error	0.08 distance error	-0.02 distance error

### Distance test at +30 degrees

47.27 inches measured	6.58 inches measured	12.40 inches measured
3.00 inches expected at 30 degrees	6.00 inches expected at 30 degrees	12.00 inches expected at 30 degrees
44.27 distance error	0.58 distance error	0.40 distance error

### Distance test at +45 degrees

44.01 inches measured	44.43 inches measured	32.70 inches measured
6.00 inches expected at 45 degrees	3.00 inches expected at 45 degrees	12.00 inches expected at 45 degrees
38.01 distance error	41.43 distance error	20.70 distance error

### Distance test at -15 degrees

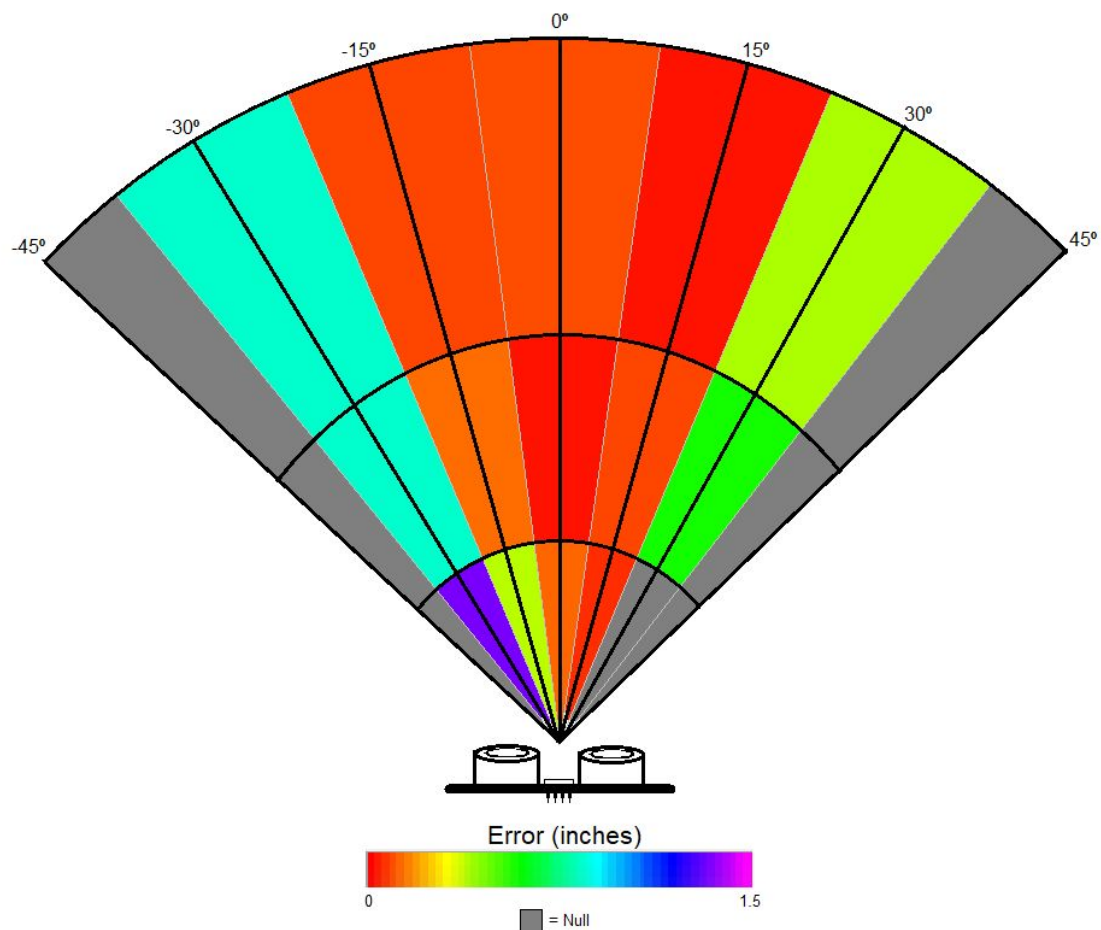
3.38 inches measured	6.13 inches measured	12.08 inches measured
3.00 inches expected at -15 degrees	6.00 inches expected at -15 degrees	12.00 inches expected at -15 degrees
0.38 distance error	0.13 distance error	0.08 distance error

### Distance test at -30 degrees

4.34 inches measured	6.84 inches measured	12.83 inches measured
3.00 inches expected at -30 degrees	6.00 inches expected at -30 degrees	12.00 inches expected at -30 degrees
1.34 distance error	0.84 distance error	0.83 distance error

### Distance test at -45 degrees

32.49 inches measured	41.71 inches measured	45.17 inches measured
3.00 inches expected at -45 degrees	6.00 inches expected at -45 degrees	12.00 inches expected at -45 degrees
29.49 distance error	35.71 distance error	33.17 distance error



**Sweet Spot:** The sweet spots in this test are shown to be at 0° between 3 and 6 inches from the sensor and 15° between 6 and 12 inches from the sensor. This shows that the best data will come at these locations.

### Precision Test

0.06 Percision Error over 10 data samples

0.04 Percision Error over 100 data samples

0.03 Percision Error over 1000 data samples

**Sweet Spot:** The sweet spot for this test shows the higher the amount of samples taken, the more precise the data due to the linear correlation between a rising sample size and a decreasing error between measurements.

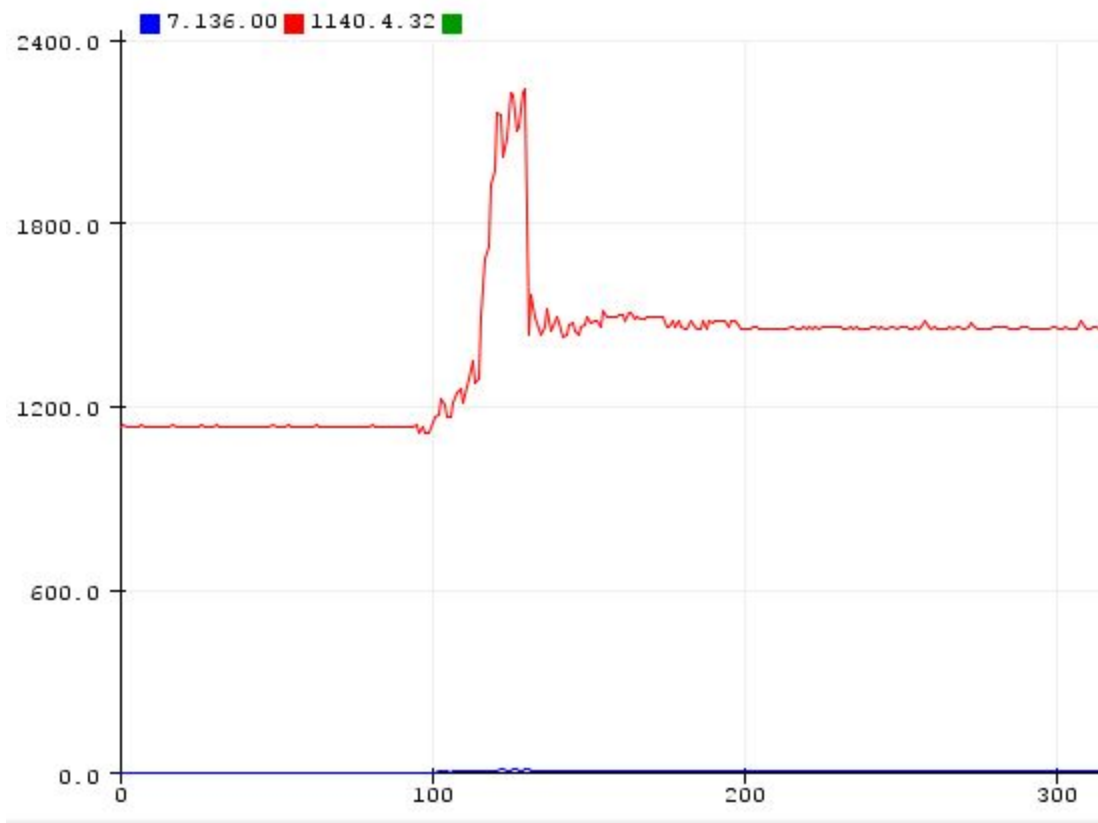


## Stability Test

.97,084.00  
3.98,1084.00  
3.97,1084.00  
3.93,1084.00  
3.93,1084.00  
3.93,1084.00

Sweet Spot: This test shows that the sweet spot time for measuring data is after 1.168ms when the data is within 2% of its steady state. This will help eliminate any transient data that may throw off measurements.

## Sensitivity Test



Sweet Spot: This test shows that the sweet spot for sensitivity is when the object is not in motion. The measurements stay constant while the object is in motion, but experiences a red-shift effect in the time taken to measure position data when an object is moved away from the sensor(as discussed in detail in lab 5).

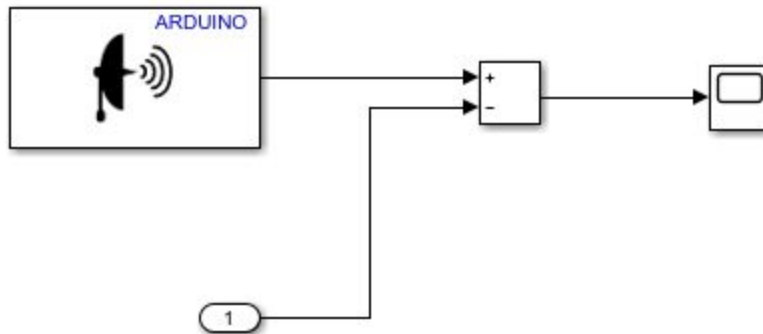
## Efficacy Test

0.02 average error at 11.14 millisecond response time

Sweet Spot: The sweet spot for a sample frequency is 11.14ms. Any faster than this and the sensor cannot gather data and process it within a single sample, any slower and the experiment will not be able to take as many readings making it less reliable.

## Simulink Model of Ultrasonic Sensor

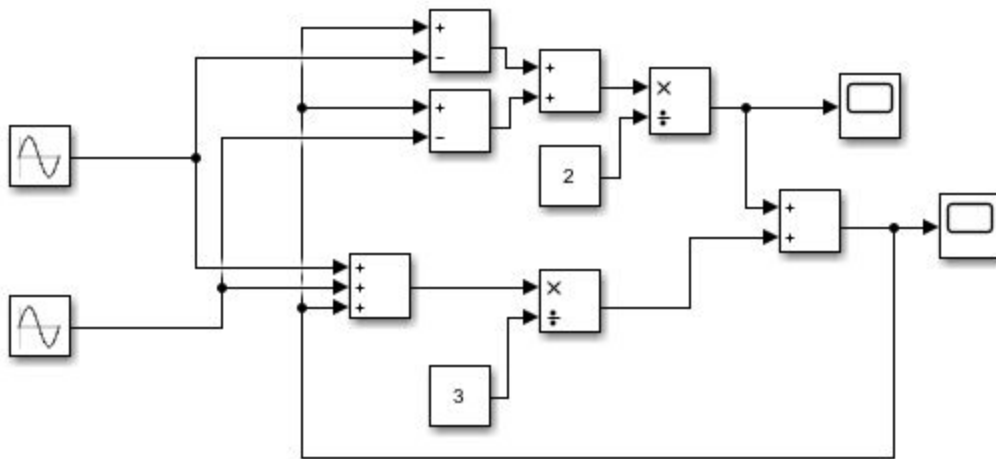
This simulink model shows how the Ultrasonic sensor is compared to an input distance value and outputs the error between them. This simple setup is useful to know at which relative positions the sensor is most accurate and where read data will be known to below a needed threshold of accuracy.



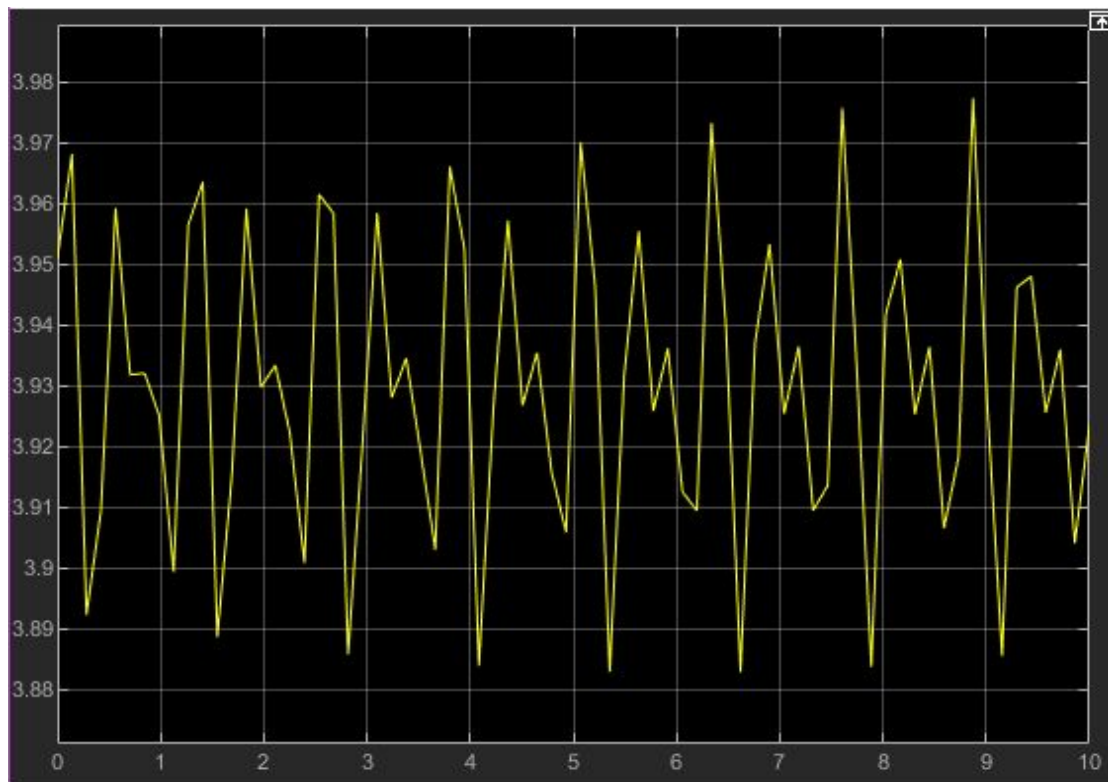
### Overview Lab 7:

This test was built to ensure the most accurate and dependable measurement from ultrasonic sensor measurements. To do this, two ultrasonic sensors were used as well as a PI system. This was accomplished by creating an instantaneous average of the two input signals with the feedback of the previous signal that was then summed with the error calculated between the input waveforms and the feedback signal. In the simulink model below, the two ultrasonic sensor inputs are estimated by two sinusoidal sources that oscillate between the values found in the stability test in Lab 6. These two signals were given slightly different phases and frequencies to simulate slight differences in the sensor hardware.

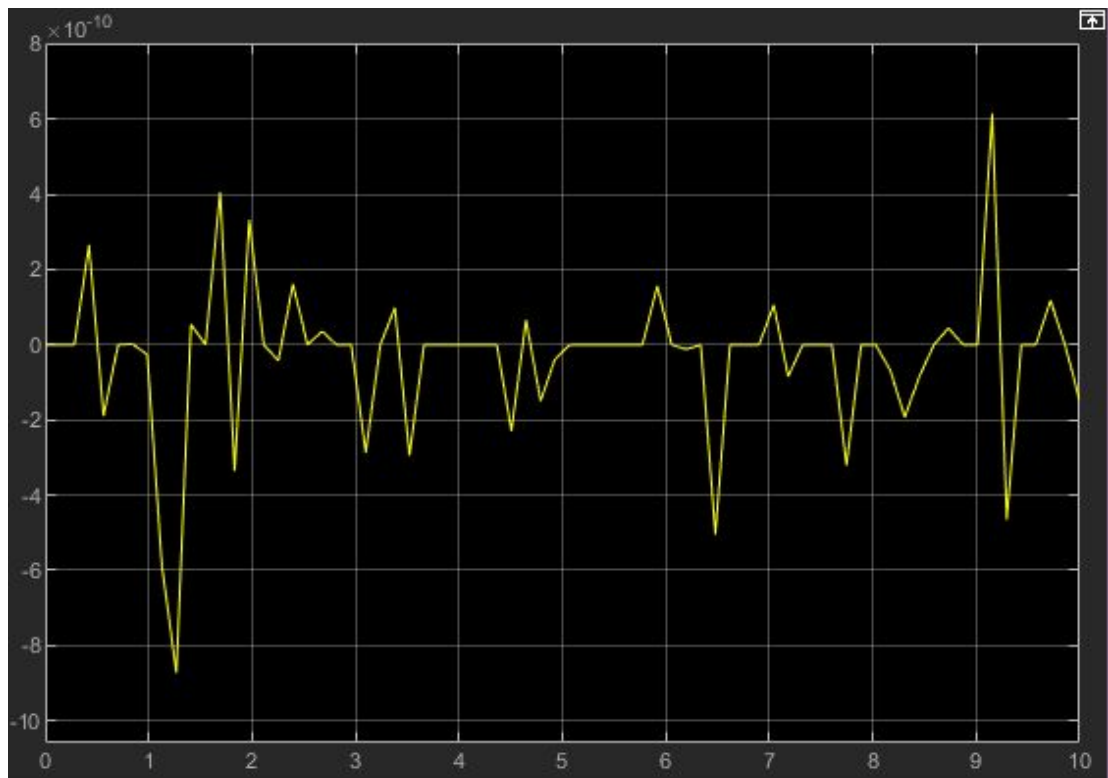
### Simulink Model



Final Signal Output



Error Signal



## 1st Input Signal

Sine Wave

Output a sine wave:

$$O(t) = \text{Amp} * \sin(\text{Freq} * t + \text{Phase}) + \text{Bias}$$

Sine type determines the computational technique used. The parameters in the two types are related through:

Samples per period =  $2 * \pi / (\text{Frequency} * \text{Sample time})$

Number of offset samples =  $\text{Phase} * \text{Samples per period} / (2 * \pi)$

Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.

Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: .05

Bias: 3.93

Frequency (rad/sec): 15

Phase (rad): 0

Sample time: 0

☒ Interpret vector parameters as 1-D

## 2nd Input Signal

Sine Wave

Output a sine wave:

$$O(t) = \text{Amp} * \sin(\text{Freq} * t + \text{Phase}) + \text{Bias}$$

Sine type determines the computational technique used. The parameters in the two types are related through:

Samples per period =  $2 * \pi / (\text{Frequency} * \text{Sample time})$

Number of offset samples =  $\text{Phase} * \text{Samples per period} / (2 * \pi)$

Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.

Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: .05

Bias: 3.93

Frequency (rad/sec): 10

Phase (rad): 1

Sample time: 0

☒ Interpret vector parameters as 1-D

## Statistical Backing

By observing the plot of the error found in the system, the maximum signal amplitude was only  $8 \times 10^{-10}$ , which shows that even with a variance of amplitude and phase between the two signal inputs, the error is successfully diminished allowing for the most accurate distance measurement possible.

## Lab 7 Conclusion

This experiment set out to attain precise distance data by using two ultrasonic sensors that are linked via a control system. From lab 6, we were able to notice that the distance values would float around a semi correct value. To achieve a more dependable reading, data from two sensors can compare their measurements to themselves and past data. The setup seen in the simulink model shows this behavior that is similar to a PI system. To test the system, 2 noise oscillators were used as inputs to simulate the error seen by the ultrasonic sensors when tested in lab 6. The min and max values were generated from those previous tests. To simulate any latency variance between the sensors and sure the two oscillators were not outputting identical data, a phase shift was introduced as well. From these tests, we can observe a less noisy and more constant output. This is accomplished by creating an error signal composed of the difference of the two oscillators with the previous data, then separately averaging the two oscillators with the previous data, which is then combined with the error signal.

## Numerical Methods and Machine Learning

With the understanding of numerical methods, we can examine and predict system behavior. This is incredibly useful for error correcting, especially when using a PI or PID system. Being able to understand the response will help not only determine the most accurate  $k_p$  and  $k_i$  values, but will allow for the ideal output. Generally with error correction, a critically damped response is desired. The error correction method could be completely correct, but if not adjusted for the system, will give poor results, such as an oscillatory response. This type of response will make inaccurate corrections to a system that can make output data worse than it was before the implementation. Ways numerical methods could make for a better design with this lab would be implementing a PID system and testing for  $k_i$  and  $k_p$  values that fit the system the best. This approach can give the desired output without working through large quantities of data first. This is ideal for systems that need to work best immediately when a system is more delicate, especially when attached to mechanical systems.

Machine learning lightens the load of numerical methods done by the engineer. Machine learning can look for patterns in a system over thousands of iterations to hone in on a solution by trial and error. This process can be incredibly helpful when there is a large bank of data to work with and when not directly attached to a sensitive system. An example of this is when using machine learning to develop complex facial recognition software with banks of millions of photos. The software will not work accurately at first, but will only get more precise when able to go through more test data. This could help in this lab by measuring the stability of data coming in from the two sensors over a long period of time and adjusting the internal system to provide the most steady output.

# **Integrated Temperature and Ultrasonic Sensor System for accurate and precision distance measurements using the TC74 Temperature Sensor and HC-SR04 Ultrasonic Distance Measuring Sensor**

**Section by Erich Wanzek**

## **A. Introduction:**

In the previous two labs, I explored the properties of the HC-SR04 ultrasonic sensor and the TC74 temperature sensor. The response and accuracy of each sensor was characterized. In this project here I designed a combined control system utilizing the two different sensors in an integrated sensor system to perform accurate distance measurements. The ultrasonic sensor relies on the time difference measurement of the time of propagation of an ultrasonic pulse. This requires the knowledge of the value of the speed of sound to perform the distance calculation. The speed of sound is the distance travelled per unit of time by a sound wave as it propagates through an elastic medium. Sound is physically understood as the compression and expansion of an elastic medium at the molecular/physical. Temperature has direct effects on a molecular level and thus the speed of sound depends strongly on temperature as well as the medium through which a sound wave is propagating. For an Ideal Gas, the speed of sound depends mainly on its temperature and composition. There is a weak dependence on frequency and pressure in ordinary air, but this does not have as big a consequence for the speed of sound. The speed of sound is determined by the collision of air molecules, and thus the temperature of the atmosphere alters the speed of sound. At 20 degrees celsius the speed of sound is 343m/s. It can be seen that due to the large dependence of the speed of sound on temperature, it would be of interest to take temperature measurements to update the current speed of sound value for the use in real time calculations of distance with ultrasonic sensors. The method proposed here to accomplish this, is to use a temperature sensor operating in conjunction with the ultrasonic sensor as an integrated sensor unit. The temperature sensor will sample the air temperature at a prescribed sampling rate and continuously update a calculated speed of sound by computation on a microcontroller. The continuously updating value of the speed of sound will be used for the calculation of distance by the ultrasonic sensors. This kind of integrated sensor unit could prove to be quite effective for robotic systems that operate in temperature varying environments that depend on precise distance measurements using ultrasonic measurements.

Before diving into the simulink model and simulation analysis of the integrated sensor unit, characters of the temperature and ultrasound sensor determined in the last two labs will be reviewed here briefly.

## **B. Review of the Characteristics of TC74 and HC-SR04:**

### **A. Review of Discussion and Analysis of the Experimental results of the TC74**

The temperature data that was collected for the TC74 showed the exponential nature of the cooling of the sensor and was in correspondence with what was expected to be observed from Newton's law of cooling equation. An exponential equation resembling Newton's law of cooling equation was fitted to the temperature data in matlab using the curvefit tool provided by matlab.

The fitted equation had the form of:

$$T = 256 + 42.16e^{-\frac{t}{387.6}}$$

The exponential time constant has a value of 387.6 and characterizes how fast the sensors thermal temperature reacts to temperature changes. The time constant is equivalent to the mass of the sensor multiplied by its effective heat capacity divided by the surface area of the sensor multiplied by the heat transfer conductance coefficient. This is all expressed as:

$$\tau = mc/(hA)$$

Thus the larger the time constant of the system the larger the heat capacity of the temperature sensor and/or the smaller the heat transfer capability of the sensor. The TC74 had an effective thermal time coefficient of 387.6 which means that for every 387.6 seconds the temperature of the sensor will have decayed to 63.2% of its initial temperature in a temperature changing environment. This can be seen from the calculator below:

$$1 - 1/e \approx 63.2\%$$

#### **i. Simulink Model of the TC74**

A simple simulink model of the sensor was devised. The sensor itself consists of just one transfer function which is a 1st order transfer function that gives an exponential and logarithmic response to an input. To test the simulink model the temperature was represented by a baseline 273 kelvin temperature environment being modulated by an additional 50 degree kelvin pulse function that had a period of 8000 seconds. The transfer function had a time constant pole value of 0.002578 which is the inverse of the experimentally determined thermal time constant and thus a corresponding gain factor 0.002578. Thus the transfer function is represented by:

$$\frac{0.002578}{s+0.002578}$$

This transfer function here would be used in analysis of the integrated temperature ultrasonic distance measuring system.

### **A. Review of Discussion and Analysis of the Experimental results of the TC74**



This table here tabulates the sensor characteristics of the HC-SR04 ultrasonic distance sensor. The chart here shows the results of medium length range scale which was subjectively quantified to be one 2 decades of range spanning from 0.1 m to 10m.

Medium Length Scales (decimeters) for 10 second continuous data acquisition					
Actual Distance Measured by ruler/tape	0.1m	0.2m	0.3m	0.5m	1m
Distance Measured By Ultrasonic sensor	0.101m	0.203m	0.2975m	0.483	0.976
Amplitude of Distance Fluctuations	+/-0.0002m	+/-0.003m	+/-0.0025m	+/-0.003	+/-0.004m
Accuracy	1%	1.5%	0.83%	3.4%	2.4%
Precision	0.001m	0.001m	0.001m	0.001m	0.001m

#### **i. Discussion and Analysis of the results of the DOE**

##### **1. Sensor Reliability**

The ultrasound had decent reliability over 10 second spans with no major fluctuations. From the accuracy results it can be concluded that the sensor is more reliable when it comes to measuring distances greater than 10 centimeters. Sensor reliability is good up to about 10-20 meters at which sound detection becomes more difficult with the low power level of the received signal becoming weak to detect thus limiting the sensor's accuracy at longer distances.

##### **2. Sensor Accuracy and Precision**

The ultrasound sensor has a precision of about 0.0001m to 0.001m and extremely poor accuracy at distances less than 3 cm , poor accuracy for distances less than 10 cm and decent accuracy for distances greater than 10 cm and less than 20 meters.

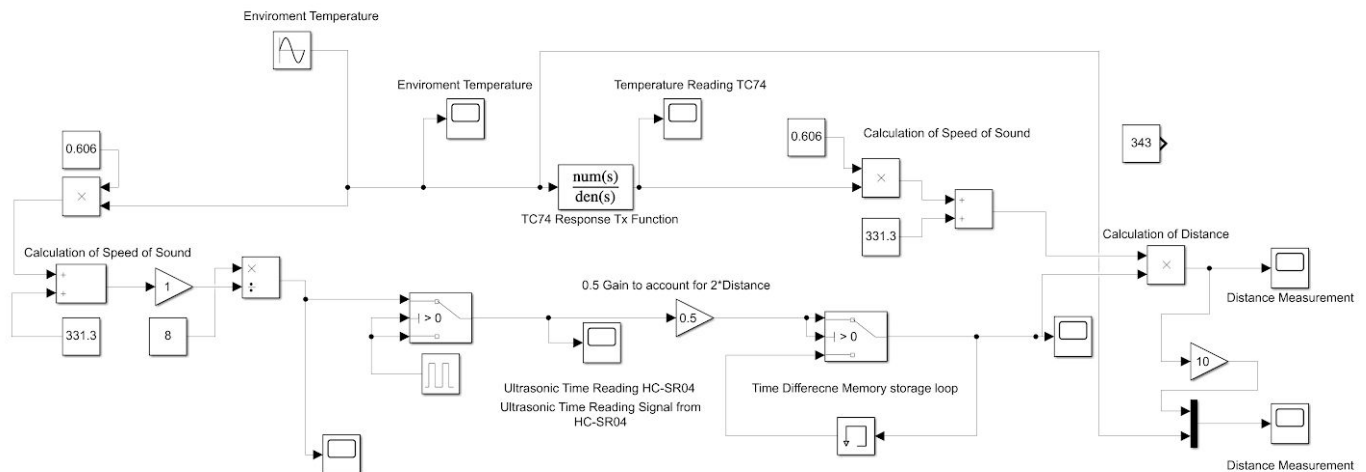
##### **3. Sensor Stability**

The sensor has a long time scale stability, but poor small time scale stability as there are fluctuations on the scale of 0.0001 meter at a period of about 0.001 seconds.

##### **4. Sensor Sensitivity**

An experiment was conducted to analyze the sensitivity of the distance detection dynamically by measuring the distance versus time measurement of a moving surface. The sensor exhibited reasonable sensitivity, but suffered from some fluctuations in the accuracy of the reading as the object was moved. Accuracy became less acceptable for objects that were rapidly accelerating.

### **C. Simulink Model of integrated TC74 HC-SR04 distance measuring system:**



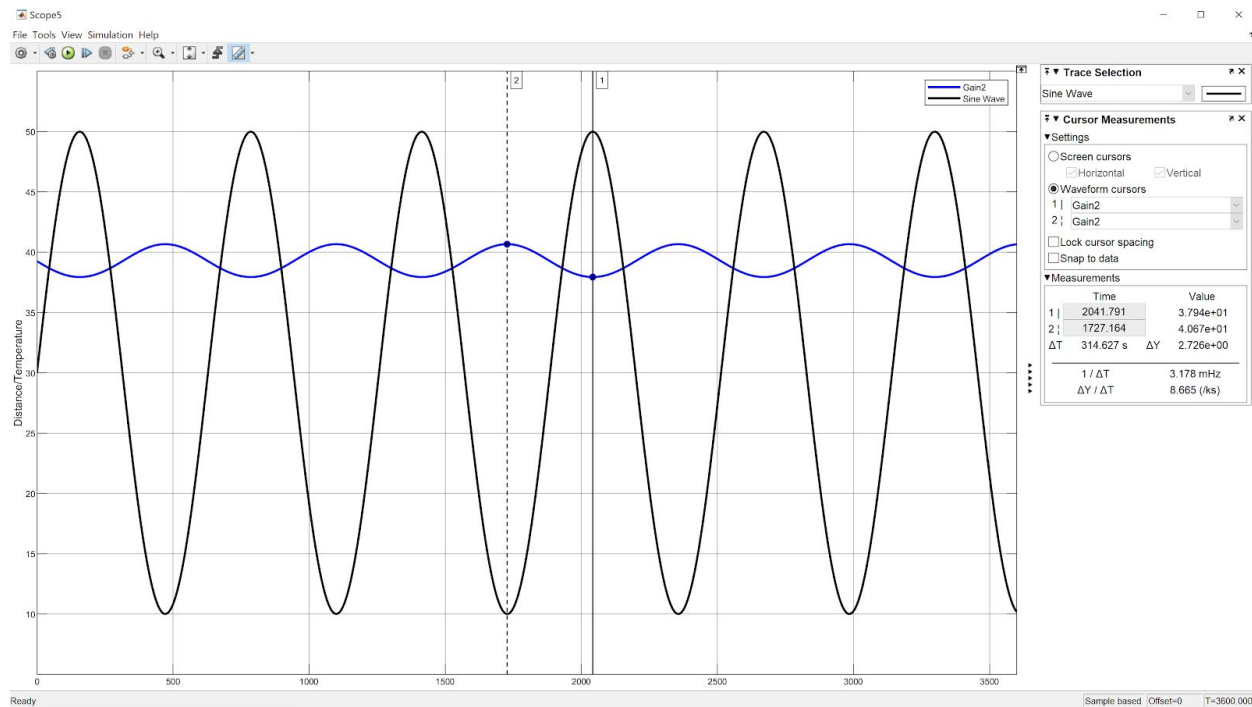
**Description:** The above image shows the integrated temperature/ultrasonic distance measuring system modeled in simulink. A sinusoidal signal simulating a low frequency temperature fluctuation is the top of the simulink schematic. The left side represents a side group to calculate the physical speed of sound at given air temperature to calculate the time of travel of hypothetical ultrasonic pulse in such a thermal environment to a distance of 4 meters and back (hence why 8 meters appear as an input constant there). This hypothetical time of travel of the hypothetical ultrasonic pulse is fed into a sampling loop that models sampling of the HC-SR04 ultrasonic sensor. The input has to be scaled by a 0.5 gain to account for the double counting in the time of travel of the ultrasonic pulse as it travels to and back from the distance to be measured. Simultaneously, as can be seen at the top of the schematic, the hypothetical fluctuating environmental temperature is continuously sampled by the TC74 which is represented by the TC74 transfer function modeling the thermal response time of the TC74 temperature sensor. The temperature reading is then used to calculate the speed of sound by the SOS temperature dependence estimation equation. The value of the time pulse and the calculated speed of sound are finally multiplied together to calculate the measured distance.

### **D. Validation of the Integrated Sensor System by Simulink Simulation:**

The first simulation of the Simulink sensor system was done with a low frequency sinusoid representing an environment experiencing a low frequency temperature

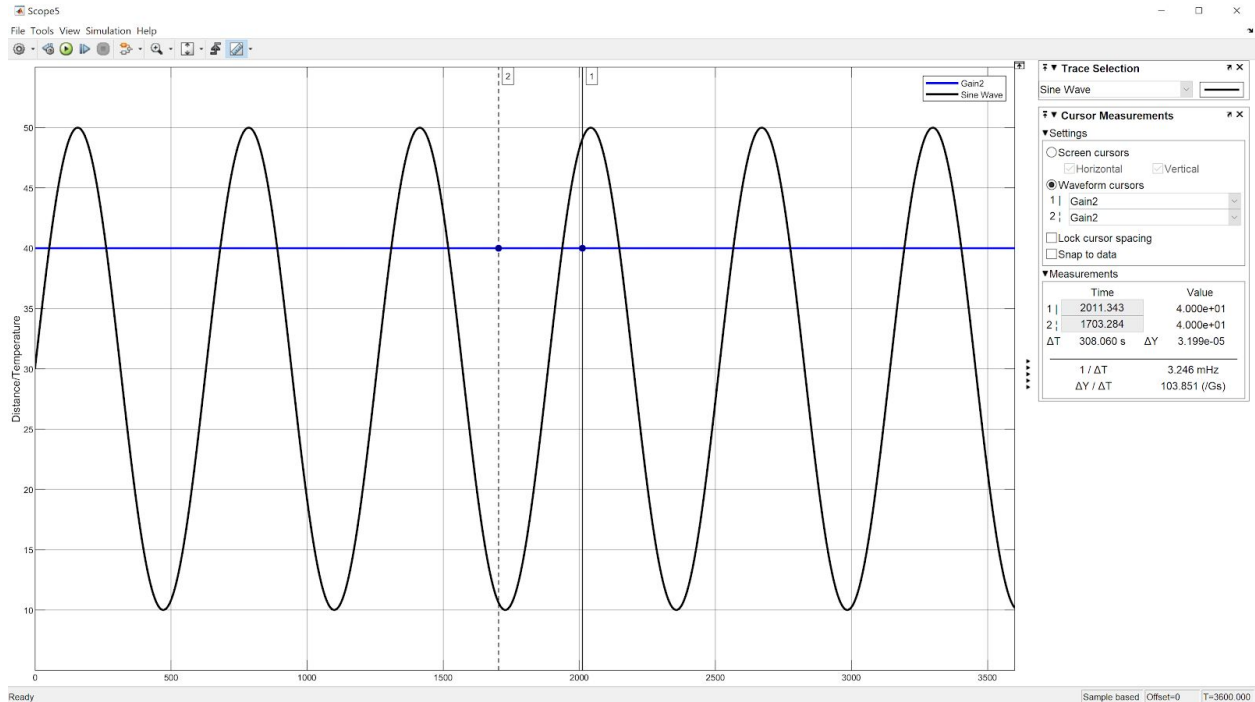
variation from 10 to 50 degrees celsius at a frequency of (0.01 rad/s) for a time duration of one hour.

### **No Corrective Calculation of Speed of Sound and No Thermal Transfer Function:**



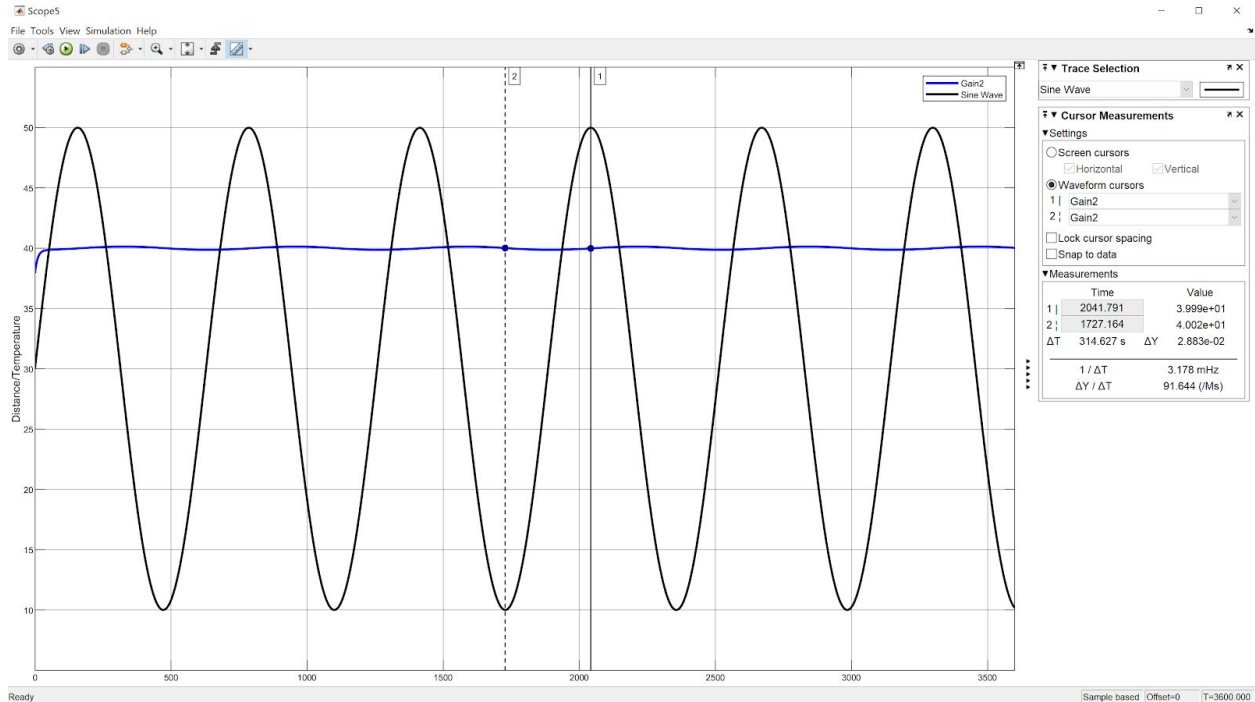
**Description:** Shown here is the simulink output. The black sinusoid represents the temperature fluctuation and the blue sinuous represents the measured distance. Both are functions of time. In the case shown here, there is no updating of the speed of sound calculation as the temperature changes, thus there is a sinusoidal variation in the speed of sound centered about the actual physical distance shown as 40 decimeters (4 meters). There is no phase lag between the output and the input because the transfer function is deactivated and thus there is no delay due to the time constant of the thermal properties of the temperature sensor.

### **Corrective Calculation of Speed of Sound and no Thermal Transfer Function:**



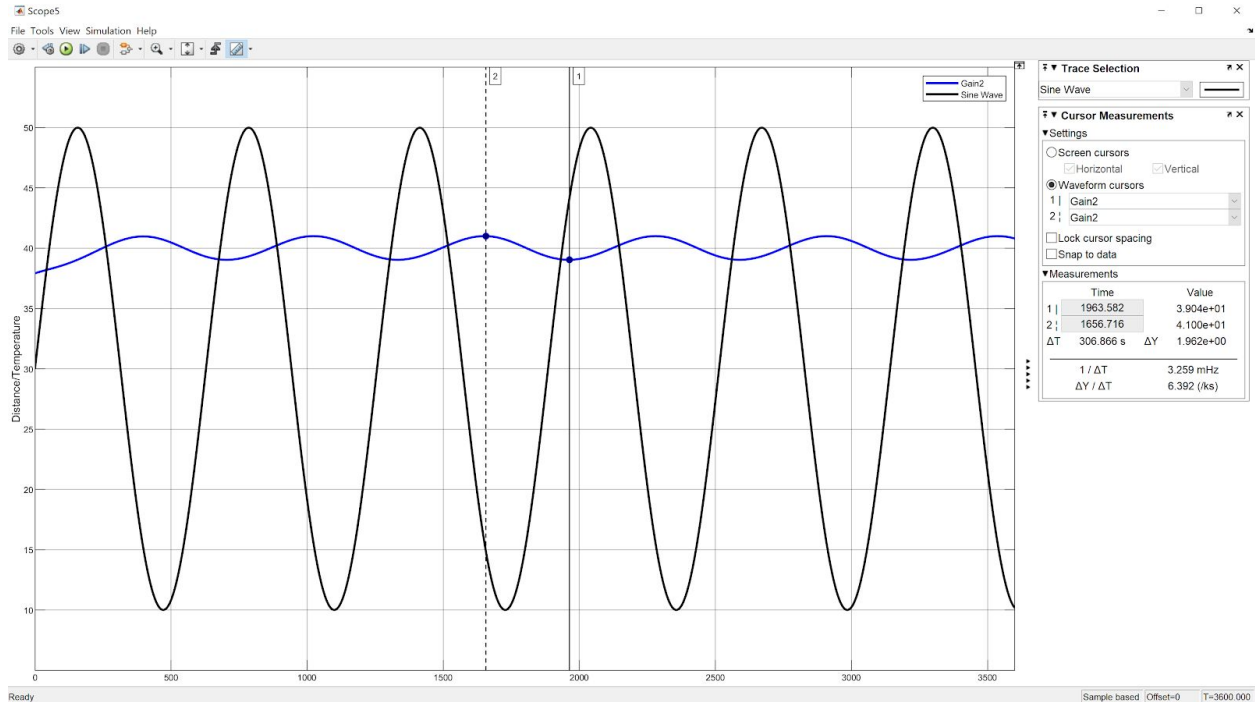
**Description:** Shown here is the simulink output. The black sinusoid represents the temperature fluctuation and the blue sinusoid represents the measured distance. Both are functions of time. In the case shown here, there is active updating of the speed of sound calculation as the temperature changes, thus there is no sinusoidal variation in the speed of sound centered about the actual physical distance shown as 40 decimeters (4 meters). The distance measured was always 4 meters despite the temperature fluctuations. There is no phase lag between the output and the input because the transfer function is deactivated and thus there is no delay due to the time constant of the thermal properties of the temperature sensor, this represents instantaneous measurement of the current temperature.

**(TimeConstant=0.1) Thermal Transfer Function Activated and Corrective Calculation of Speed of Sound:**



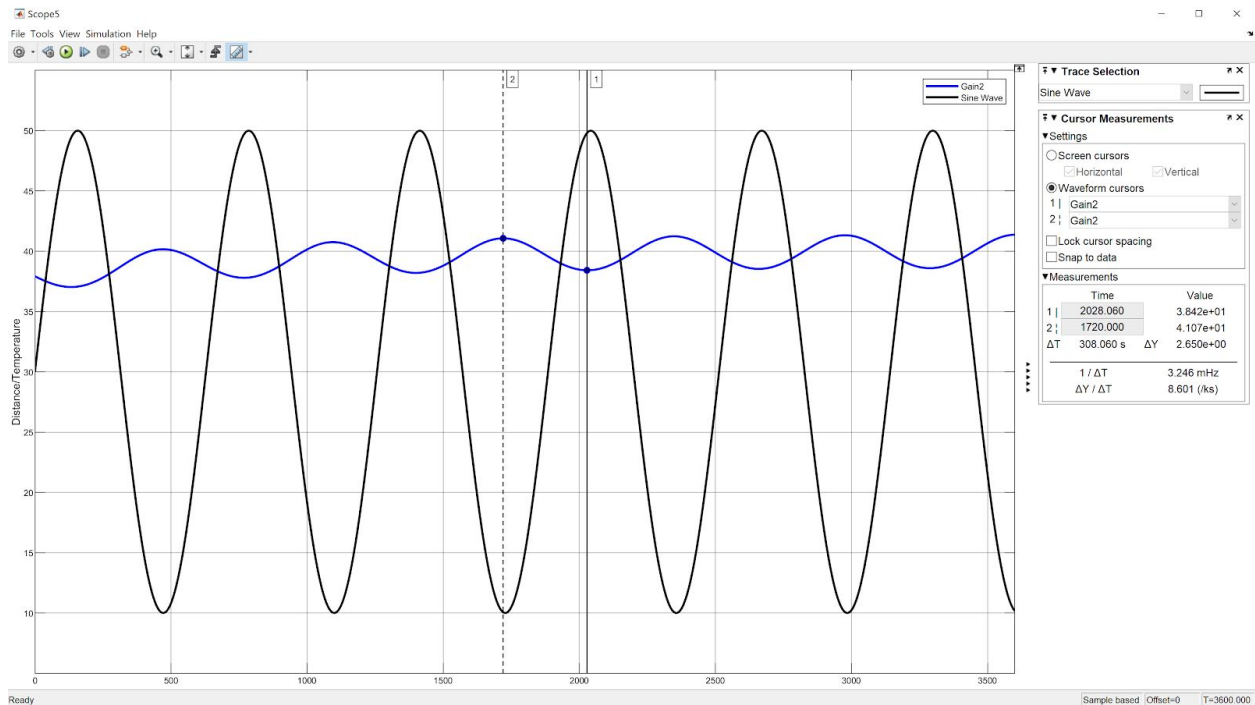
**Description:** Shown here is the simulink output. The black sinusoid represents the temperature fluctuation and the blue sinusoid represents the measured distance. Both are functions of time. In the case shown here, there is active updating of the speed of sound calculation as the temperature changes, thus there is small variation in the speed of sound centered about the actual physical distance shown as 40 decimeters (4 meters). However there is 314.6 seconds of phase lag between the output and the input because the transfer function is now activated and now the delay due to the thermal time constant of the thermal properties of the temperature sensor, is being taken into account. The thermal time constant of the sensors here is higher than the TC704 at 0.1 representing fast response measurement time of the temperature.

**(TimeConstant=0.01) Thermal Transfer Function Activated and Corrective Calculation of Speed of Sound:**



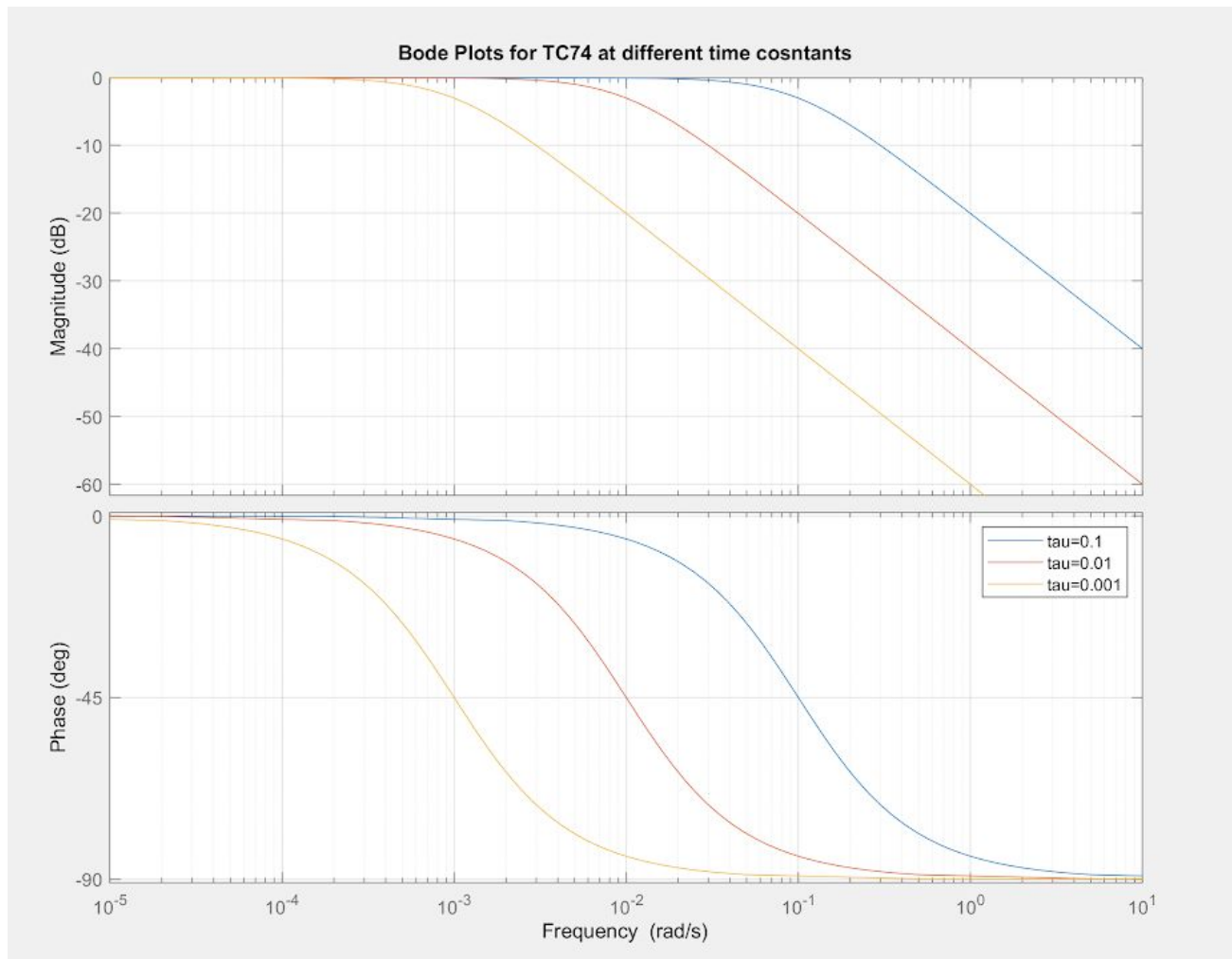
**Description:** Shown here is the simulink output. The black sinusoid represents the temperature fluctuation and the blue sinusoid represents the measured distance. Both are functions of time. In the case shown here, there is active updating of the speed of sound calculation as the temperature changes, thus there is small variation in the speed of sound centered about the actual physical distance shown as 40 decimeters (4 meters). However, there is approximately 500 seconds of phase lag between the output and the input because the transfer function is now activated and now there is delay due to the thermal time constant of the thermal properties of the temperature sensor being taken into account. The thermal time constant of the sensor here is similar to that of the TC704 at 0.01 representing realistic response measurement time of the temperature.

**(TimeConstant=0.001) Thermal Transfer Function Activated and Corrective Calculation of Speed of Sound:**



**Description:** Shown here is the simulink output. The black sinusoid represents the temperature fluctuation and the blue sinusoid represents the measured distance. Both are functions of time. In the case shown here, there is active updating of the speed of sound calculation as the temperature changes, thus there is small variation in the speed of sound centered about the actual physical distance shown as 40 decimeters (4 meters). However, there is phase lag between the output and the input because the transfer function is now activated and now there is delay due to the thermal time constant of the thermal properties of the temperature sensor being taken into account. The thermal time constant of the sensor here is smaller than that of the TC704 at 0.001 representing a slow response measurement time of the temperature.

### **Code Plots of Thermal Transfer Functions for the Different Time Constants Used in the Above Simulink Simulations**



**Description:** Shown here are bode plots of the Thermal Transfer Functions for the Different Time Constants Used in the Above Simulink Simulations. It can be seen from the bode plots that the smaller the time constant of the Thermal Transfer function (the greater the pole value) the faster the system responds and this allows higher frequency signals to be transmitted without change in magnitude or phase (time lag). In the case of the integrated temperature system this means that the faster the response time of the sensor, the more accurate the ultrasonic distance measurements will be in an environment with rapidly fluctuating temperature.



## **E. Conclusion of the Integrated Temperature/Ultrasonic Distance Measuring Sensor System.**

The integrated sensor module proved to function as desired after analyzing the simulink simulations. For environments with very rapid temperature variations, a fast reaction temperature (small time constant) is required in order for accurate results to be calculated. If a slow reacting temperature sensor is used in such a case, phase and magnitude change due to the thermal transfer function of the temperature sensor will cause unacceptable errors causing inaccurate errors in the measured distance. The results would not necessarily change due to scaling as the thermal time constant of the temperature is constant for any size of environment, however when measuring longer distance thermal gradients in the environment could cause variation in the speed of the sound along the travel path of the ultrasonic pulse. Thus this needs to be kept in mind when considering long distance measurements up to the distance measuring limits of the HC-SR04 ultrasonic sensor. Initial analysis of the design could be improved by using more accurate speed of sound equations and more accurate measurement of the time constant to the temperature sensor being used in order to assure good response time. With the good accuracy and precision of the individual sensors established beforehand, the sensors in conjunction with precision speed of sound calculations and fast thermal time constant would allow for a very accurate and precise distance measuring device using ultrasound to be constructed.