A Design, Verification, and Testing (DVT) Protocol for a Detection System of Acoustic Signals

Sarah Ali
Clinical Assistant Professor, Department of Engineering
Loyola University Chicago
Chicago, Illinois, USA
sali29@luc.edu

Abstract

Detecting acoustic signals can help with daily life to a large extent. Depending on a set frequency, one can detect lifesaving emergencies such as intruders or leaks (Levine 1999). On top of this, once detection has been made, many different outputs can be created as a result of that detection. Once an intruder comes, the system can automatically contact authorities or if a leak is detected, the system can automatically notify the user that the leak has occurred. Detecting acoustic signals provides us with the comfort of being able to live more stress-free knowing that there is a system in place ready to detect an emergency at any time. The main question of our research was to come up with a system that detects acoustic signals and outputs a Peak frequency after performing a high pass filtering (Novick et al. 1999) of the frequency of detected sounds. In this work, we implemented a sound detection system that is fundamental to the functionality of devices such as alarm systems and medical devices. The implemented system is successful in detecting sounds. In addition, we tested the system using a Simulink testing and verification system which is the first verification system to be taught in undergraduate engineering programs. It involves a sound-based verification, in which, when the test passes, a particular sound of 400-490 Hz will ring. Loyola University Chicago is now the first university to have successfully completed DVT Design Verification in an Undergraduate Engineering program using Simulink. This testing and verification system has also proved to be useful in industries and companies as Simulink testing is becoming more commonly used because of its powerful model-based design and testing features.

Keywords

Sound, detection, LabVIEW, MyDAQ and Simulink.

1. Introduction

This system is designed in LabVIEW to detect any acoustic signal and process the signal through a high pass filter. After this, a peak frequency is outputted. This system is important because it can be used in many applications including but not limited to biomedical, cyber/ computer, environmental applications. We list some examples of the system's uses. The system could be used:

- as a fire alarm system after detecting sounds of fires.
- as a security system for the protection of homes, companies, industries belongings.
- to detect water leaks or hydraulic applications.
- to analyze acoustic signals and differentiates between sounds which could also open fields for interesting research projects that involves sound recognition.

The realistic constraints of the system include having a LabVIEW license, Ni-MyDAQ, and a sound sensor (KY038).

2. Methods

2.1 LabVIEW Sound Detection and MyDAQ

The security alarm system described requires only three components: the LabVIEW algorithm, the NI myDAQ, and the sound sensor KY038. Integration of hardware and software can be tedious without proper visualization of each subsystem, which is why LabVIEW serves as a visual programming language in which the user can customize and adjust hardware configurations as needed. The National Instruments myDAQ, a data acquisition device designed to be used with LabVIEW, can acquire digital and analog input signals from external sources or generate its own digital and analog output signals. LabVIEW interfaces with the myDAQ via an A-Male to B-Male USB cable. Finally, the sound sensor KY038 is connected to the myDAQ via three connections: signal, power, and ground.

Because this sensor detects microphone signals as voltage values, the sensor's output is analog. This analog output pin should be wired to any analog input terminal of the myDAQ. This sensor also needs to receive 3 volts in order to function properly; therefore the +V pin of the sensor must be connected to the 5V terminal of the myDAQ. The proper resistors should also be added to convert 5V to 3V for the sensor's correct power input (Lee 1996). Finally, the digital GND pin of the myDAQ should be wired to the GND input of the KY038 sound sensor. The hardware setup is crucial for the correct LabVIEW implementation.

The LabVIEW implementation involves only several steps in the software and connections drawn in between. To acquire the signals, the user should include have the hardware setup described previously and ready to go. To communicate with this hardware, the user should open LabVIEW, and create a blank VI. The user must use a series of blocks that can be used to acquire the sound signals. The use could do this, the user could simply use the DAQ Assistant Express VI. The user should right click on the block diagram to open the functions palette, then Measurement I/O, then NI DAQmx, then DAQ Assist. The user should then place this block in the block diagram. A dialog will appear from the DAQ Assistant for the user to configure the signal. The user should select Acquire Signals, then Analog Input, then Voltage. This Express VI will then prompt the user to select the physical channel 'ai0.' This is the myDAQ's physical, analog input pin that is wired to the signal output of the sound sensor. The user should then configure the parameters of this data acquisition by selecting 10 and -10 for the maximum and minimum input voltage, respectively. The acquisition mode should be set to continuous samples, samples to read set to 100, and the rate set to 1000 Hz. This will allow LabVIEW to acquire 100 samples of the sound sensor's input voltage at 1000 samples per second.

The voltage acquisition is now done in LabVIEW. However, noise filtering of the signal is vital for the system at hand. The alarm system detects all sounds surrounding the sensor, and some sounds may be unnecessary. The mains electricity frequency, produced by alternating current voltage of electricity to power homes, is about 60 Hz. This frequency must be filtered out of or end voltage signal to send the correct frequencies to the user. The user should right click on the block diagram, select signal processing, then waveform conditional, then filter. The user should then configure the filter type to High pass, and cutoff frequency to 100 Hz. This will allow only frequencies above 100 Hz to pass through the filter. The user should connect the data output terminal on the DAQ Assistant to the Signal input terminal of the Filter.

The user may observe the filtered signal with an indicator by right-clicking in the front panel of the VI, selecting 'Graph' in the controls, and the function, 'Waveform Graph.' In the block diagram, the user should connect the filtered signal output to the waveform graph block. However, the user of the security alarm may not want to see the entire signal, but the frequency associated with the highest tone. To do this, the user may search for the Express VI 'Tone Measurements' by right-clicking on the block diagram in LabVIEW. The user may configure this block to measure Frequency by checking the box 'Frequency' in the single tone measurements of the dialog box that appears after placing the Tone Measurements VI. Finally, the user should convert this frequency to a double precision floating-point number by searching for 'To Double Precision Float' function; the frequency output from the Tone Measurements function should be connected to the input of the To Double Precision Float function. This number must then be converted into a fattened string of binary values with the 'Flatten to String' LabVIEW function.

The frequency must be stored as a string to be safely sent to a remote UDP socket. The 'UDP Write' LabVIEW function takes the frequency in the form of a string and writes it to the user's port of choice. The data string containing the frequency should be connected to the 'data in' port of the UDP Write function. The user should place a while loop, located in the Structures folder of the Functions pallet, around the DAQ Assistant, Filter, Tone Measurements, DBL, Flatten to String, and UDP Write functions. The structure in the while loop will always execute once before executing again and will continue to send the sound frequencies to the user until the specified stop condition.

The stop condition may depend on whether the user would like to stop the data acquisition or if an error occurs after the UDP Write function executes. If either of these conditions is true, the while loop should stop executing and the data acquisition should end. To insert a stop button, the user may choose 'Stop Button' located in the Boolean file of the Controls palette in the front panel. In the block diagram, the user should select the OR gate in the Boolean folder, place it in the while loop next to the stop icon, and connect its output to the input of the stop icon. The user should then connect the output of the stop button to one input terminal of the OR gate. The other input terminal of the OR gate should be connected to the error output from the UDP Write function.

After the while loop executes, the UDP socket must close. The user should select the 'UDP Close' function from the function palette and place it to the right of the while loop. The connection ID and error outputs of the UDP write function should be connected to the connection ID and error inputs of the UDP Close function, respectively. The Simple Error Handler function in the function palette should be placed in the block diagram with its input connected to the error output of the UDP Close function. The error handler indicates whether an error occurred in the functions wired to this error cluster.

The UDP Write function requires a connection ID, address, and port or service name to send the frequency to the user. The connection ID comes from the local port with which the user creates a UDP socket. This UDP socket can be created via the UDP Open Function in the functions palette of the block diagram. This function should be placed outside the while loop and requires the local port input, a numeric constant or string. The output connection ID from the UDP Open function should be connected to the connection ID input of the UDP Write function.

The address input of the UDP Write function comes from the address of the computer to which the user would like to send the frequency (LabVIEW Tools 2021). If the user would like to send this datagram to his or her own computer, he or she should select the String to IP function and place it in the block diagram outside the while loop. The user should then create a string constant, and type "localhost" as the value of the string constant. This constant value should be connected to the String to IP function input, whose output should be connected to the address input of the UDP Write function inside the while loop.

The port or service name identifies the remote port to be sent the signal's frequency and can be a numeric or string input. This numeric or string can be a constant value connected to the input of the UDP Write function. The LabVIEW code of the security alarm system is complete and may be started via the Run button in the top left corner of LabVIEW.

2.2 Communication between LabVIEW and Simulink
Linking the sound detection system built in LabView to Simulink is essential to carrying out verification and testing. Simulink is a powerful modeling tool with various features that aid in system building, verification, and validation. Generally, communication between software's can be achieved in the following ways: through Transmission Control Protocol (TCP) like chatting applications, Simple Mail Transfer Protocol (SMTP) like email communication, or User Datagram Protocol (UDP). This system will utilize UDP to enable communication between LabVIEW and Simulink (figure 1).

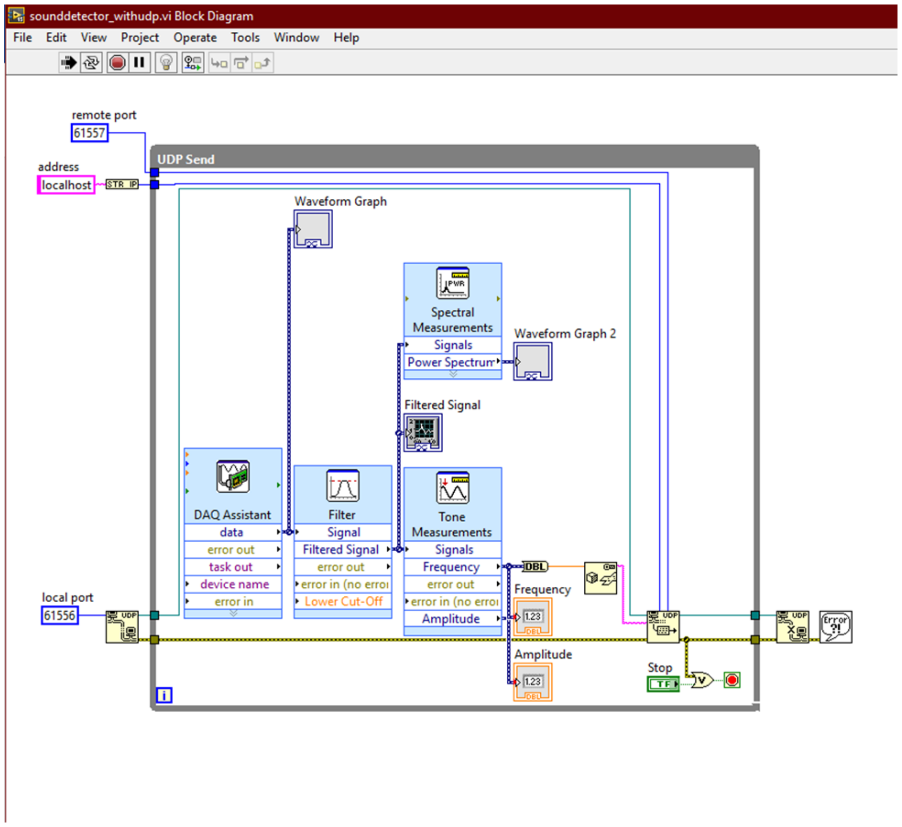


Figure 1. LabVIEW model that shows the complete sound detecting system and how it communicates to Simulink through UDP.

On the left side of the diagram, outside of the iteration loop, three things should be specified:
1. A remote port (61557)
2. An address (localhost)
3. Local port (61556) linked to an UDP read block

3. Data Collection
3.1 Simulink Model
In order to construct the model, a toolbox (Simulink Support Package for Arduino) should be installed first. Then the passive buzzer KY-006 which can produce a range of sound tones depending on the input frequency.

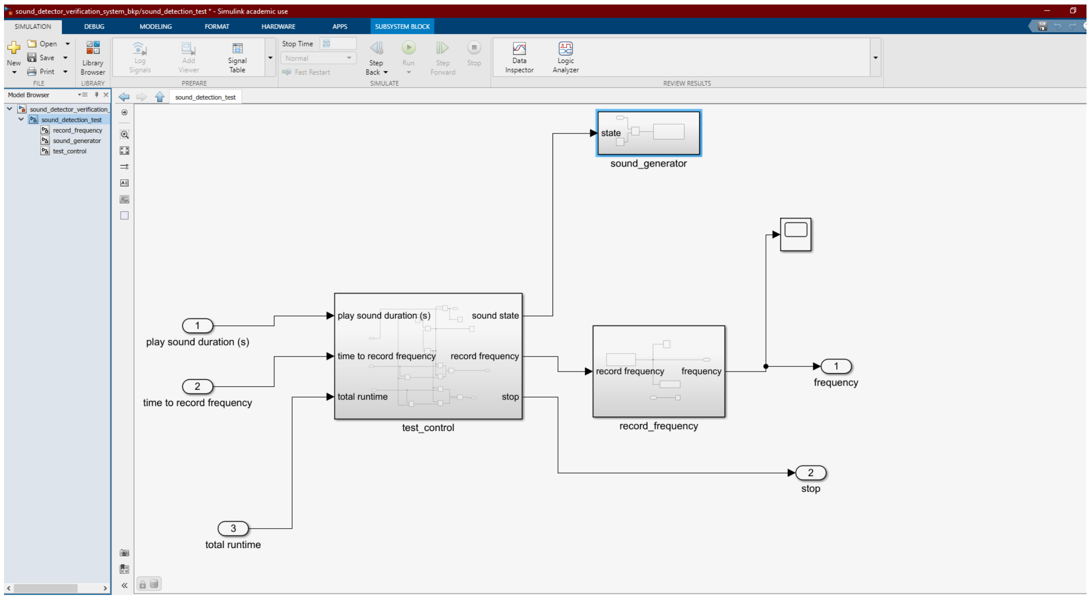


Figure 2. The whole sound detection testing Simulink model.

As shown in Figure 2, the three subsystems of this model are:
1. The sound generator
2. The test control
3. The record frequency

3.2 Sound Generator

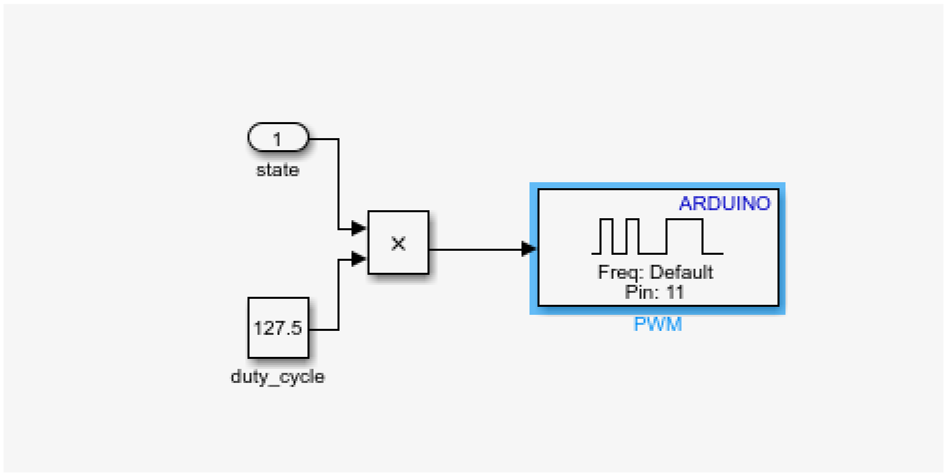


Figure 3. The sound generator subsystem

In order to find the PWM ARDUINO blo4k just simply open the library browser and select the block in common under Simulink Support Package for Arduino (see figure 3). The PWM (Pulse Width Modulation) block generates a square waveform (sound) on the specified output pin. Then configure the block by entering 11 as the pin number. The constant 127.5 in figure 2, is the duty cycle 50% of the waveform which should shows the same constant gaps for the sound frequency.

3.3 Test Control

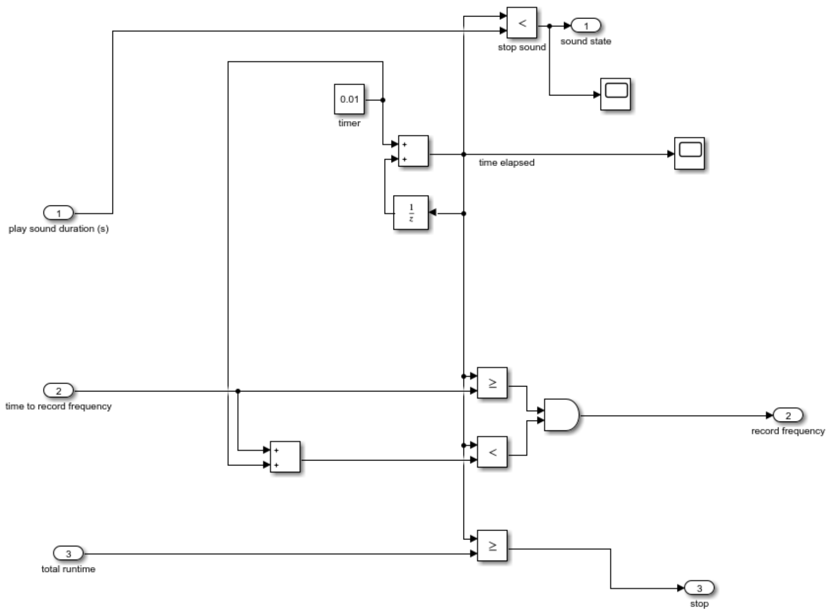


Figure 4. Test control subsystem.

As you can see in figure 4, the Test control subsystem is divided into 3 parts: the upper part controls how long to play the sound, the middle part records the frequency, and the lower part gives the total runtime.

The upper part of the subsystem simply indicates that the sound will play for a chosen number of times, for instance 5 seconds, and then it stops after this time. Note that the block holds the data for one second before releasing it. The data is added to the number given by the timer to compute the elapsed time.

The middle part of Figure 4 indicates that at the frequencies are recorded at the end of the test i.e., after we hear the sound. The lower part of the test control subsystem in figure simply indicates that if the total runtime is greater or equal than the elapsed time then the system should stop.

4. Results and Discussion

Through testing, we have determined that the system functions as expected. The system successfully detects a generated sound signal. The filter was then properly applied, and the peak frequency identified. After a frequency was identified, an email was sent to the owner of the system detailing that a sound was detected.
This system will be further tested in another work using Simulink Verification Testing. Simulink testing will provide validation that the system works properly by utilizing Simulink Requirements, Test, and Coverage. Design Verification testing ensures that the system not only meets specified requirements but also the clinical needs of the user. A Simulink Test report will show if the requirements are met within the system.

4.1 Numerical Results

To show numerical results we must build a subsystem that records the frequency.

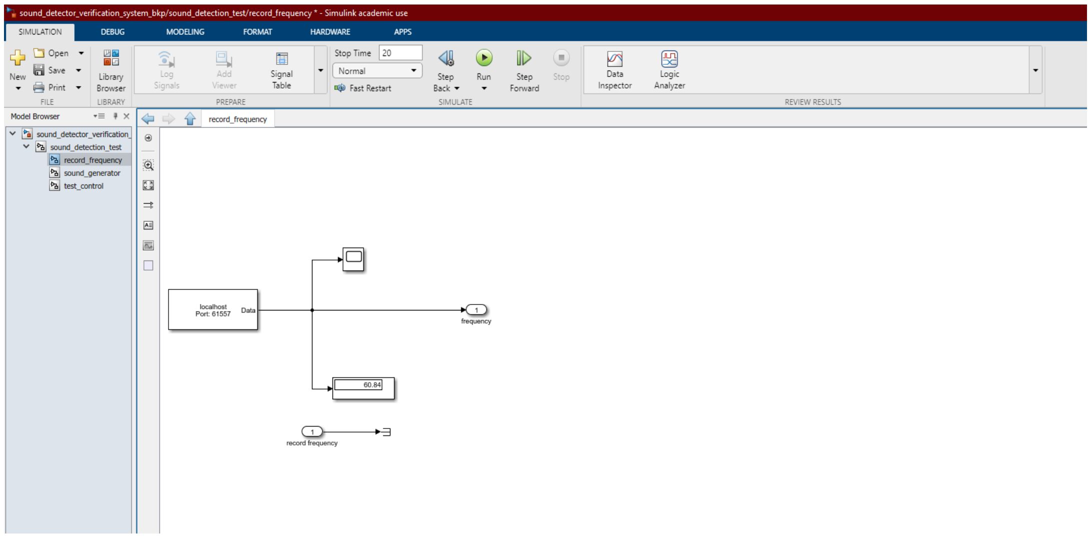


Figure 5. The subsystem that records frequency. In this example the recorded frequency is 60.84 Hz.

The record frequency gets the data from LabVIEW and sends it to the output frequency port which is the frequency. The record frequency subsystem is shown in figure 5.

4.2 Graphical Results

After running the whole simulation, which includes the verification of the sound detection, we can hear a constant beeping sound for five seconds, which in the first test, is called the Alarm test of sound frequency between 400 and 490 Hz. This is the normal range of a sound of a beep generated from a PWM Arduino processor of a duty cycle 50%. Next the beeping sound stops which is the second test called the false alarm test (i.e. when no beeping sound is detected). The result of the scope of the model is shown in figure 6. This scope shows the plot of sound frequencies of both tests. Figure 6 demonstrates that between 0 and 5 seconds, the frequencies recorded were between 400 and 450 Hz for the background noise. However, after 5 seconds the frequencies were lower than that range because other types of sounds were detected.

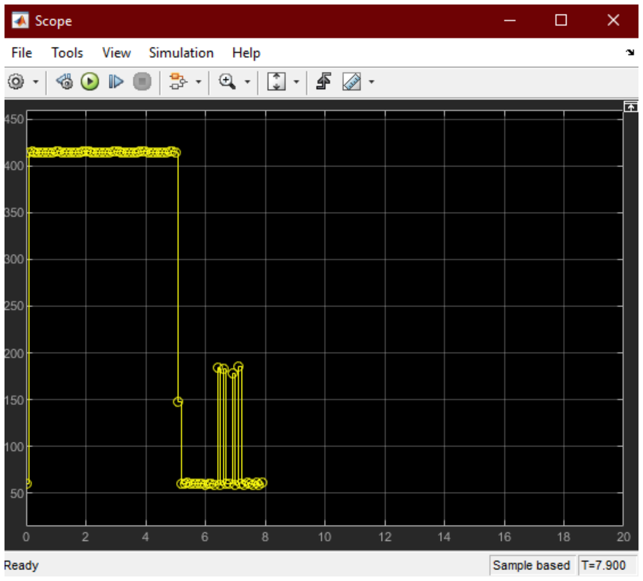


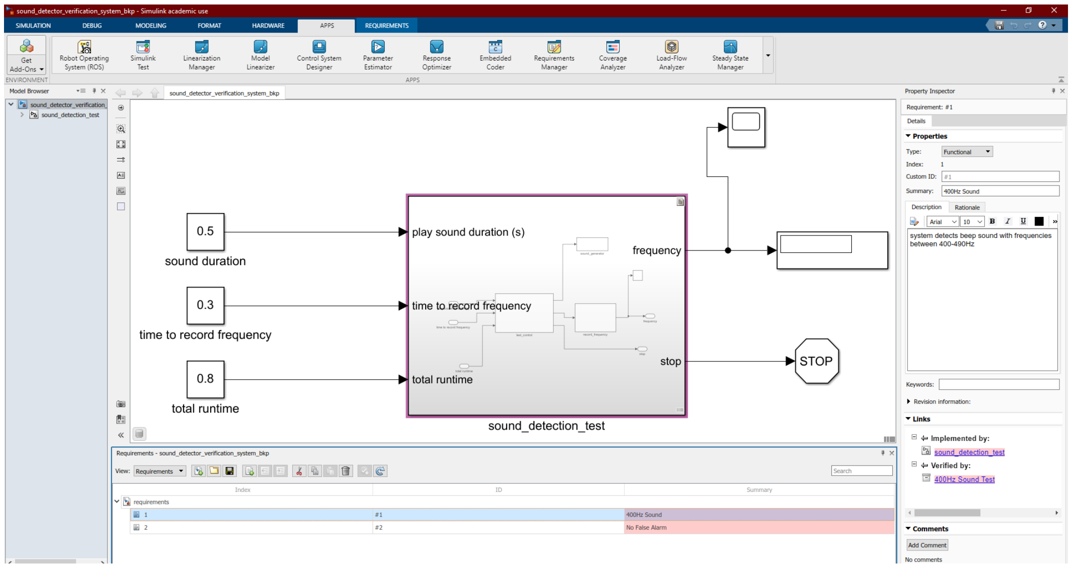
Figure 6. The output of the frequency of the sound detection test.

4.3 Proposed Improvements

In order to improve the system, several updates could be made. Including more filters could make sound detection more specific (Jespersen 1991). The system could identify specific frequencies to mean different events have occurred, and the notification to the use could then also be more specific. The system could also then be made to ignore certain frequencies that are naturally in an environment, for instance a dog walking around an empty house (Hackman and Sullivan 1996). The system could also be used for voice recognition to arm or disarm the system. Another feature that could be added is a timer. Sounds could be detected, and the system could measure the duration of time the sound was detected. Certain frequencies or lengths of time in which a sound is detected could trigger more serious communication such as directly informing emergency services. If this were a security system set up in someone's home or in another building, a hardware setup with dials or buttons could allow the user to determine what frequencies they wish to detect or ignore before they arm the system and leave the building. A timer or clock could also allow the system to be armed for certain amounts of time or at specific times during the day. The system could function automatically at certain times of the day even if a human is not present.

5. Validation

To validate our results, we built a testing and verification system using Simulink. First, we open the Requirements Editor (Simulink Requirements 2021). In the Apps tab, we click Requirements Manager. In the Requirements tab, we click Requirements Editor.
Using the + button the first requirement is added, which is the Alarm test (of frequencies between 400 and 480 Hz). Then we complete the fields of this requirement see the bottom right of figure 7. Next, we add the second requirement No false alarm and complete its fields. To link those two requirements, we click on the first requirement and then RC on the model and select requirements -> link to selection to Requirement Browser. Then, the same steps are repeated to create the second requirement.



Requirements - sound_detector_verification_system_bkp

View: Requirements

Index	ID	Summary
requirements		
1	#1	400Hz Sound
2	#2	No False Alarm

Requirement: #1

Properties

Type: Functional

Index: 1

Custom ID: #1

Summary: 400Hz Sound

Description: system detects beep sound with frequencies between 400-490Hz

Keywords:

Links

Implemented by: sound_detection_test

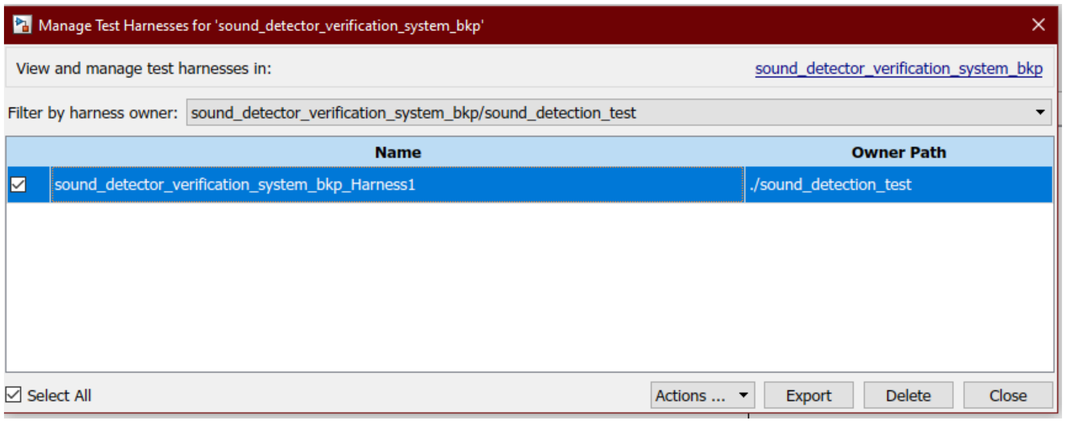
Verified by: 400Hz Sound Test

Comments

No comments

Figure 7. Creating the requirements.

To create the test harness, we open the requirement editor and RC on the model and select test harness-> Manage test harness then we double click on sound_detector_system_bkp_Harness1 (see figure 8).


Manage Test Harnesses for 'sound_detector_verification_system_bkp'

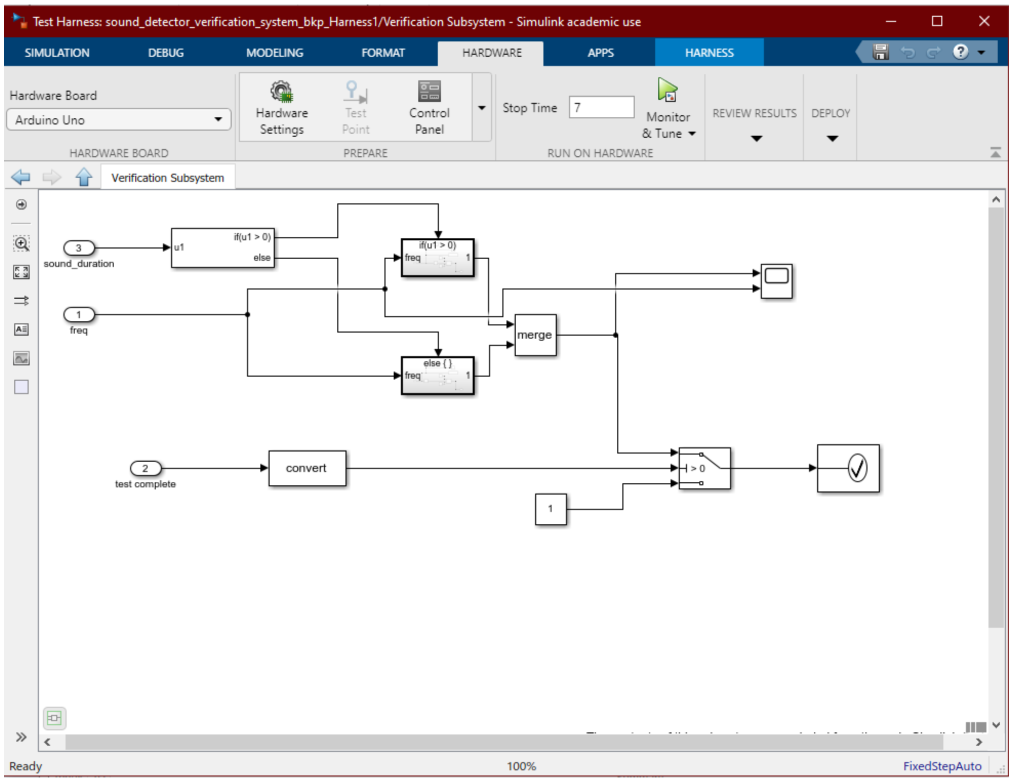
View and manage test harnesses in: sound_detector_verification_system_bkp

Filter by harness owner: sound_detector_verification_system_bkp/sound_detection_test

	Name	Owner Path
☑	sound_detector_verification_system_bkp_Harness1	./sound_detection_test

Figure 8. Managing test harness

In the test harness (Simulink Check 2021) we select Harness -> test cases-> Simulink test Manager (see figure 9).


Test Harness: sound_detector_verification_system_bkp_Harness1/Verification Subsystem - Simulink academic use

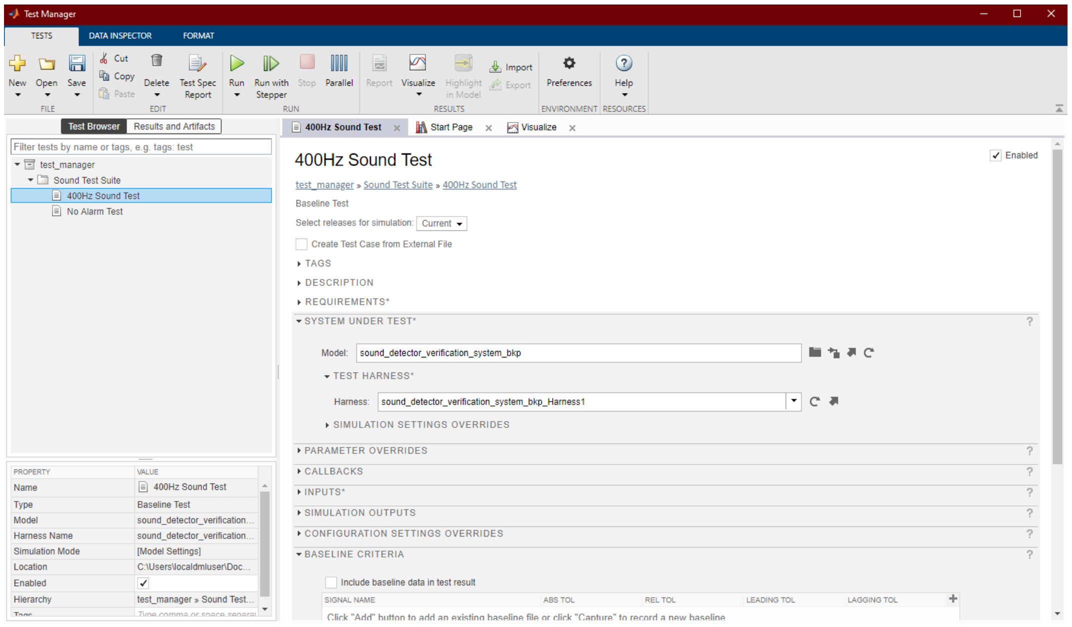
Hardware Board: Arduino Uno

Stop Time: 7

Figure 9. Simulink test Manager.

Figure 10 shows us how the verification is done. There are two tests. First if the duration is >0 then there is a beep sound and it will basically test to make sure there is a frequency between between 400-490 Hz using the interval test. Second, if the duration is zero then there is no sound and the test makes sure that not the sound frequencies are not within constant range.

In the test Manager, we click on Requirements tab and add using the + button the link to the selected requirement.



Filter tests by name or tags, e.g. tags: test
test_manager
Sound Test Suite
400Hz Sound Test
No Alarm Test

PROPERTY	VALUE
Name	400Hz Sound Test
Type	Baseline Test
Model	sound_detector_verification...
Harness Name	sound_detector_verification...
Simulation Mode	[Model Settings]
Location	C:\Users\localdmluser\Doc...
Enabled	✓
Hierarchy	test_manager » Sound Test...

400Hz Sound Test

✓ Enabled

test_manager » Sound Test Suite » 400Hz Sound Test

Baseline Test

Select releases for simulation: Current

Create Test Case from External File

TAGS
DESCRIPTION
REQUIREMENTS*
SYSTEM UNDER TEST*

Model: sound_detector_verification_system_bkp

TEST HARNESS*

Harness: sound_detector_verification_system_bkp_Harness1

SIMULATION SETTINGS OVERRIDES
PARAMETER OVERRIDES
CALLBACKS
INPUTS*
SIMULATION OUTPUTS
CONFIGURATION SETTINGS OVERRIDES
BASELINE CRITERIA

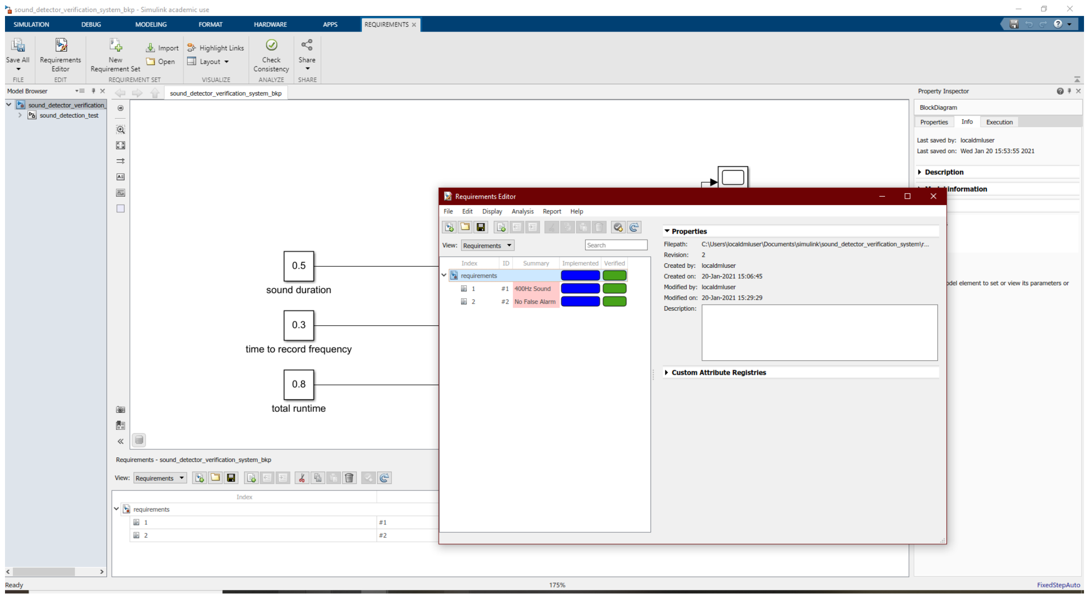
Include baseline data in test result

SIGNAL NAME	ABS TOL	REL TOL	LEADING TOL	LAGGING TOL

Click "Add" button to add an existing baseline file or click "Capture" to record a new baseline

Figure 10. Verification of the system.

Finally, we run the test to complete the verification. Figure 11 shows that requirements were implements and verified.



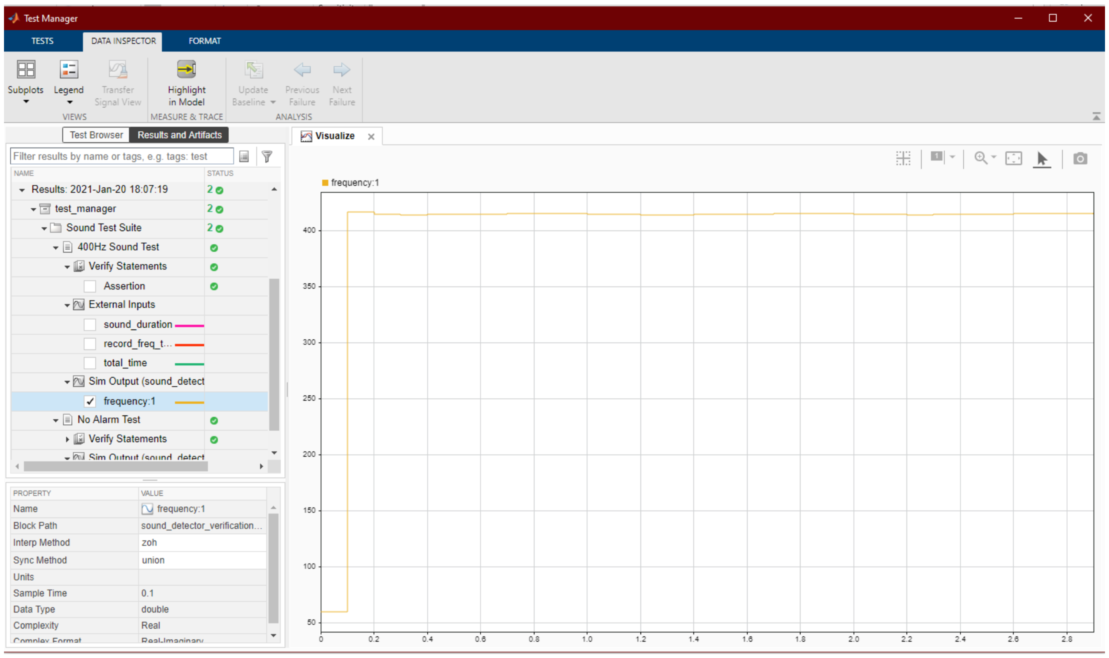
Index	ID	Summary	Implemented	Verified
requirements			🟦	🟩
1	#1	400Hz Sound	🟦	🟩
2	#2	No False Alarm	🟦	🟩

Properties
Filepath: C:\Users\localdmluser\Documents\simulink\sound_detector_verification_system\r...
Revision: 2
Created by: localdmluser
Created on: 20-Jan-2021 15:06:45
Modified by: localdmluser
Modified on: 20-Jan-2021 15:29:29
Description:
Custom Attribute Registries

Last saved by: localdmluser
Last saved on: Wed Jan 20 15:53:55 2021

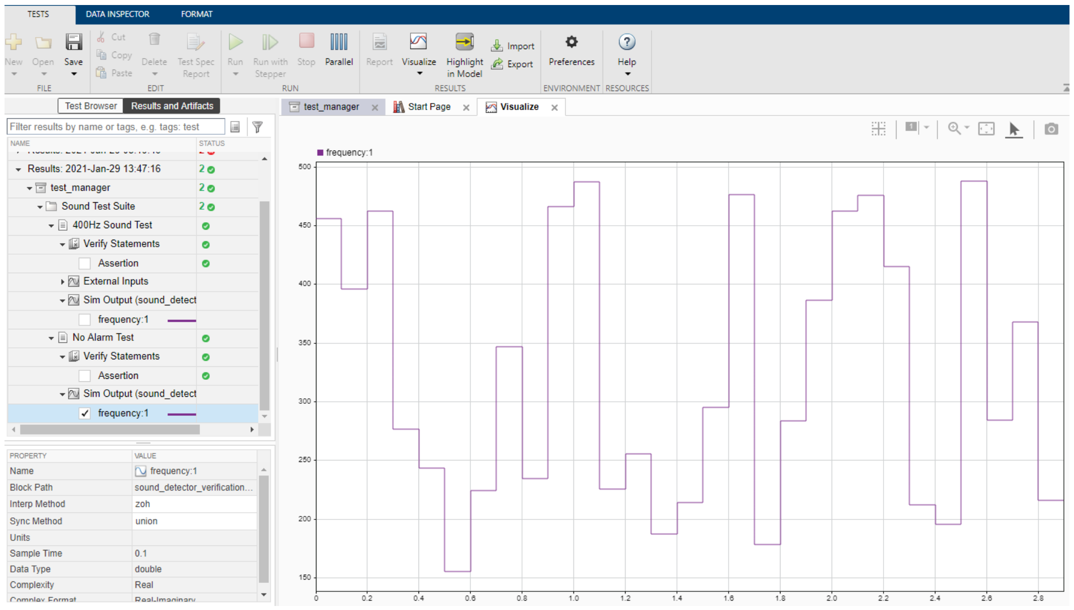
Figure 11. The verification of requirements is completed.

Using the Data Inspector Simulink feature, the details of the verification can be checked (Simulink Coverage 2021). For example, for the first verified test or the Alarm test, the plot of frequencies in figure 12 verifies that the range of frequencies is between 400 and 490 Hz. Figure 13 verifies that with no sound beep the frequencies are not within a constant range because they simply output to the background noise.



PROPERTY	VALUE
Name	frequency:1
Block Path	sound_detector_verification...
Interp Method	zoh
Sync Method	union
Units	
Sample Time	0.1
Data Type	double
Complexity	Real
Complex Format	Real-Imaginary

Figure 12. Plot of frequency of the second test.



PROPERTY	VALUE
Name	frequency:1
Block Path	sound_detector_verification...
Interp Method	zoh
Sync Method	union
Units	
Sample Time	0.1
Data Type	double
Complexity	Real
Complex Format	Real-Imaginary

Figure 13. Plot of frequency of the second test.

6. Conclusion

The simplicity of the design allows for a portable and low-risk system that can be used almost anywhere. The system could be used to detect intruders in someone's home, water leaks from pipes in a building, or the functionality of any appliances that are supposed to make noise or that are not supposed to make noise (Cook and Ali 2012).

Overall, the system is functional and extremely versatile. This sound detection system could improve functionality of devices as well as security of businesses and human lives (Sullivan et al. 1990.)

This testing system is the first verification system to be taught in undergraduate engineering programs. It involves a sound-based verification, in which, when the test passes, a particular sound of 400-490 Hz will ring.

Loyola University Chicago is now the first university to have successfully completed DVT Design Verification in an Undergraduate Engineering program. This has proved to be useful in students' other coursework, such as in Capstone Design Projects, and will prove to be useful in industry as Simulink testing is becoming more commonly used. There is growing interest by companies in industry to utilize Simulink for its powerful model-based design and testing features (Masud and Whitman 2010). Additionally, students implemented the testing system while working in teams. This is important as teamwork is critical in having a successful outcome. Students successfully completed verification of the system and learned about Simulink verification testing.

References

Hackman, C. and Sullivan, D. B., Eds., Time and Frequency Measurement, *American Association of Physics Teachers*, pp. 110-125, College Park MA, USA, November 12-15, 1996.

Levine, J. Introduction to time and frequency metrology. *Review of scientific instruments*, pp.2567-2596, Fort Collins CO, USA, September 11-13, 1999.

Novick, A. N et al. High-performance multi-channel time interval counter with an integrated GPS receiver, *Proceedings of the 31st Annual Precise Time and Time Interval (PTTI) Meeting*, Dana Point CA, USA, p. 561, 1999.

Sullivan, D. B et al. Characterization of Clocks and Oscillators, *NIST Technical Note 1337, U.S. Government Printing Office,* Washington, DC, October 23-27, 1990.

Jespersen, J., Introduction to the time domain characterization of frequency standards, *Proceedings of the 23rd Annual Precise Time and Time Interval (PTTI) Meeting*, pp. 83-91, Pasadena CA, October 11-15, 1991.

LabVIEW Tools for DAQ Configuration, Available: https://www.zone.ni.com, Accessed on May 21, 2021.

Simulink Requirements, MATLAB & Simulink, Available: https://www.mathworks.com/products/simulink-requirements.html, Accessed on May 21, 2021.

Simulink Check, Available: https://www.mathworks.com/products/simulink-check.html, Accessed on May 21, 2021.

Simulink Coverage, Available: https://www.mathworks.com/products/simulink-coverage.html, Accessed on May 21, 2021.

Lee, J., Measurement of machine performance degradation using a neural network model, *International Journal of Modelling and Simulation*, vol.16, no. 4, pp. 192-199, 1996.

Masud, A.S.M. and Whitman, L.E., Educating future engineers: An example, *Proceedings of the First International Conference on Industrial Engineering and Operations Management*, pp. 175-179, Dhaka, Bangladesh, January 9 – 10, 2010.

Cook, V. and Ali, A., End-of-line inspection for annoying noises in automobiles: trends and perspectives, *Applied Acoustic*, vol. 73, no. 3, pp. 265-275, 2012.

Biography

Sarah Ali, Ph.D. is a clinical assistant professor in Engineering with a specialization in the field of biomedical engineering. Dr. Ali graduated in 2015 with a Ph.D. in electrical engineering at Laval University in Quebec City, Canada. Prior to joining Loyola University Chicago, Dr. Ali worked at Size Stream as a Scientist and software developer. Her focus was to develop algorithms for the 3D modeling of human bodies. She also had the opportunity to work at GE Healthcare where she developed software related to healthcare applications. Designing software for medical device is very crucial in the field of healthcare. Dr. Ali is interested in investigating the recent technologies of processing medical images and designing software for medical devices. Her other interests are in computer vision, image processing and machine learning.