Garmin Tone Generator User Manual

Garmin Tone Generator Team 1

December 10, 2023

Abstract

This document serves as a guide to utilizing the underwater tone testing device developed for the Garmin Dive team by the Garmin Tone Generator 1 Team in Duke's EGR101 class.

Contents

1 Overview

1.1 Useage Process

1. Connect the Arduino (via USB-A, or a USB-C to USB-A adapter can be used) and aux cable to computer.

2. Run Arduino code and upload to Arduino Uno R3. Ensure that the serial monitor is closed before proceeding.

3. Run the Python script. Enter the necessary file paths when prompted in the terminal.

4. The sound will play on the watch automatically. This is followed by the dial tone, which is the time that the user will twist the potentiometer dial to indicate their selection of four levels of feedback.

5. Feedback data will be outputted in the terminal and written to the text file that the engineer supplied.

6. The engineer has the option to continue testing. The sound engineer has the option to use the same feedback data file for subsequent runs without retyping the file path.

2 Software Systems

2.1 Arduino

In order to receive feedback data from the Arduino, one must be running the Arduino script through the Arduino IDE. This can be found at the following link: https://www.arduino.cc/en/software which offers distributions for macOS, Windows, and Linux. Run the script for reading potentiometer data within the IDE, being sure to select the Arduino UNO R3 board from the menu reading "Select Board." One can verify that this code is running properly by opening the serial monitor, which should display values roughly between 0 and 1023 at the extreme values of the potentiometer dial.

2.2 Python

In order to run the Python script, one must have access to a computer equipped with Python. Python can be downloaded from https://www.python.org/downloads/. The package dependencies can be downloaded using pip, Python's package installer. A guide for pip can be found at https://pip.pypa.io/en/stable/user_guide/. It is necessary to download:

1. serial
2. pyserial
3. playaudio

On Unix/macOS, this looks typing the following into the Bash terminal:

```
    python -m pip install playaudio
```

Listing 1: Unix/macOS Package installaion

On Windows, the same procedure looks like:

```
    py -m pip install playaudio
```

Listing 2: Windows Package installaion

The other packages should come pre-equipped with Python, including os and time. If not, they can be downloaded in the same process using pip. In order for the sound engineer to play a sound and receive feedback from the diver, running this code will begin by displaying prompts in the terminal. Sound engineers should have audio files for the test tones, a file to receive the feedback data, and a dial tone (which may be downloaded from the GarminTone1 Github). The code will attempt to identify the serial port for the Arduino automatically, however, the user will be prompted to enter the path to the port manually should this process fail.

There will then be subsequent prompts asking the engineer to enter the file path for the sound, a file path for a dial tone, and a file path for the text file to receive the output data. When entering the file paths, there is no need to enter the root user, the code autofills that portion for ease of use. Upon completing the prompts, the testing tone will play on the watch, followed by a dial tone that gives the time for the user to supply feedback. At the end of the dial tone, the feedback position of the potentiometer will be recorded and written to the output text file. Each line of the text file will contain the entered file path followed by the rating scored by the diver as one of "Worst","Bad", "Good", and "Best". A prompt will subsequently appear in the terminal allowing the engineer to continue testing new tones without reconnecting to the serial port and can optionally keep the same output file. Alternatively, one can quit the program by entering "q."

2.3 GitHub Repository

Software resources can be found in the GitHub repository: GitHub. One can download the necessary files and code from this repository, however will still need to locally set up Python with the necessary packages in order to run the code.

3 Electronics

3.1 Potentiometer

The potentiometer is made by Bourns, Inc. The link for the specific model of the variable resistor is here. The potentiometer is attached to the housing via velcro, enabling replacement of the potentiometer should the internal electronics fail. The potentiometer was tested individually for durability by placing it in fresh water and salt water for three weeks without a significant decline in performance, although rusting was observed. To change the potentiometer, remove the potentiometer from the velcro and unsolder the potentiometer connection, taking note of which wires are connected to the load, ground, and wiper respectively. The green wire should be plugged into the A0 port on the Arduino, while the oher two wires should be connected to 5 V and GND (ground). Testing by the EGR101 team revealed that the potentiometer should last for three weeks after its first exposure to water.

3.2 Speaker

The speaker is manufactured by Dayton Audio. The link for purchase can be found here. The manufacturer cites the speaker as waterproof, and the EGR101 team has not observed any issues regarding utilizing the speaker underwater. The speaker is superglued to the housing. Replacement of the speaker would be conducted by removing it from the housing and unsoldering the speaker from the extension wire (connection occurs at location of red electrical tape).

This speaker is rated to cover the range of 20 Hz to 20 kHz at or above 70 dB. The EGR101 team has tested the speaker to generate a frequency response curve so that the Garmin sound engineers can engineer the amplitude of the tones in order for the speaker to output 70 dB. This chart was generated by measuring the sound pressure level in air at a distance of 1.22 m away. Frequency was measured every 100 Hz from 200 Hz to 1000 Hz, every 250 Hz from 1000 Hz to 4000 Hz, every 500 Hz from 4000 Hz to 6000 Hz, and every 1000 Hz from 6000 Hz to 20000 Hz. The plot can be found in Figure 1.

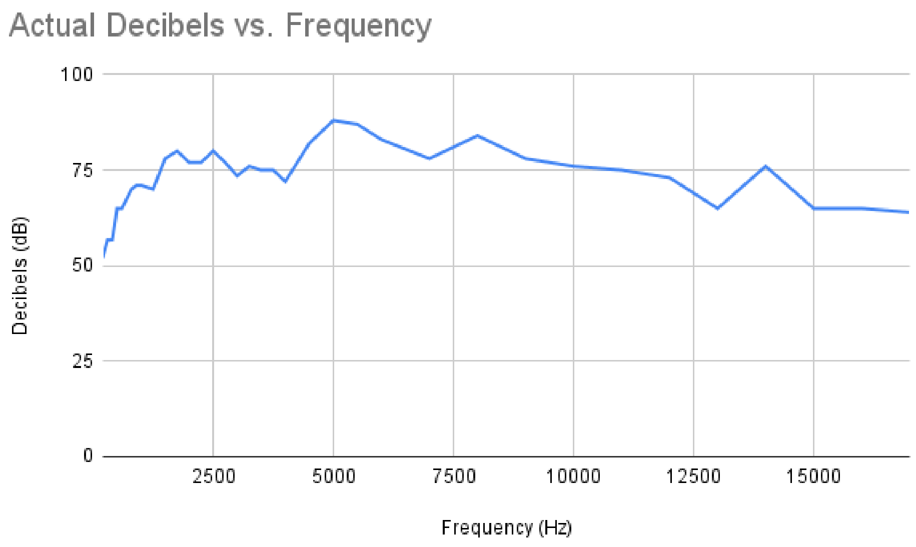

Figure 1: Measured Decibel vs. Frequency Plot of Speaker from 1.22 m Away

Additionally, we have included the frequency response curve provided from the manufacturer in Figure 2.

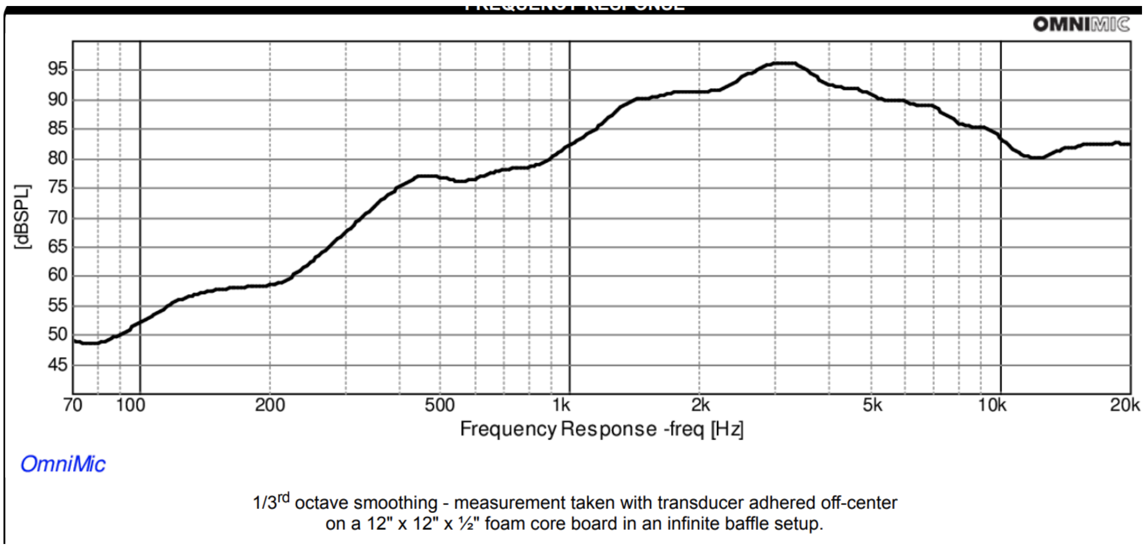

Figure 2: Manufactureer Decibel vs. Frequency Plot

3.3 Amplifier

The utilized speaker is an Adafruit Class D 20 W audio amplifier. The link for purchase can be found here. We use an aux cable to send sound information from the computer to the amplifier. A 12 V power supply is connected to the amplifier across the connection labelled 12 V. The speaker is subsequently connected to the audio amplifier at the “Left” output.

4 Mechanical Systems

4.1 3D Printed Base

The base, dubbed the “Millenium Falcon,” is a 3D print containing slits to hold the watch bands, a surface for the speaker, and an elevated tower that fits the potentiometer. The potentiometer is connected via velcro, a system that has been tested underwater and has proved viable. Additionally, the velcro allows easy removal should the potentiometer require replacement.

4.2 Watch Band

We used a soft silicone watch that follows the standard for the Apple Watch for ease of replacement. The particular band that we puchased can be found here. This band is affixed to the slits in the housing using superglue.

4.3 Potentiometer Knob

The knob of the potentiometer was 3D printed to fit snugly on the potentiometer. The indicator is a piece of electrical tape affixed to the knob.

5 Troubleshooting Q&A

Q: How do I know the root directory?
A: When the code is run, it will print the current root directory. If you are having issues accessing files using the file paths, it is helpful to read the error message produced. Often, this will show if something is mistyped or if there is an erroneous backslash.
Q: How do I know if the Arduino is reading values?
A: One can open the serial monitor in the Arduino IDE to ensure that the Arduino is reading values from the potentiometer. If these values are fluctuating, it is necessary to replace the potentiometer in order to get an accurate

reading. Ensure that you close the serial monitor in the Arduino IDE before running the Python script so that Python can access the serial port.
Q: What do I do if the code will not upload to the Arduino or the Arduino is not reading the potentiometer?
A: Ensure that all of the connections are secure. Check that the proper board is selected in the Arduino IDE and that it is reading from the proper port. If these steps fail, try restarting your computer to mitigate the issue.
Q: What should I do if the ratings from the potentiometer are reversed?
A: The easiest fix is to swap the 5 V and GND cables. Alternatively, one can reverse the code so that certain values map to "Worst" instead of "Best" and "Bad" instead of "Good" and vice versa.
Q: What should I do if the speaker is not outputting any sound?
A: Ensure that the dial on the amplifier is twisted clockwise maximally for the maximum volume. If this does not fix the issue, ensure that you entered the correct file path and that the sound information is in a .wav, .mp3, or .mp4 file format.
Q: How to we prevent problems before testing underwater?
A: Be sure to calibrate the device on land by playing sounds and making sure the output values are correct before testing underwater. When the diver dives underwater, ensure that they are given plenty of slack on the wire.
Q: How can I easily reuse presets for the output file and sound file?
A: The program will remember the file paths most recently used. Thus one can simply hit enter in order to fill in these values should the engineer want to output data to the same file or play the same sound. The dial tone and Arduino port will remain constant for all trials once the program is started.

6 Acknowledgements

We would like to acknowledge Mr. Wes Trenholme-Pihl for presenting us with this project and providing feedback throughout the entire process. Additionally, we would like to thank Dr. Martin Brooke, Dr. Ali Roghanizad, and Kayla Fericy for their technical expertise and support throughout this project. We would also like to thank the EGR101 program for allowing us the opportunity to work on this interesting and enriching project.