Discussion

Setup:

For the analog configurations, the ADC was linked to pin PA3 while the DAC was linked to pin PA4. For the ADC and DAC to work as intended, direct memory access (DMA) one shot conversion is the DMA method used to assemble and access the array entirely without interfering with the main process since they work in parallel. The Microphone was attached to analog pin PA3 while the audio amplifier was connected to DAC pin PA4. The pin configuration for the push buttons and LEDs were connected to the pins of the H7 boards accordingly as detailed in Figure 3.

Pins	PD7	PD6	PD5	PD4	PF7	PF9	PG1
Name	BT_1	BT_6	BT_3	BT_4	LED_RED	LED_YELLOW	LED_GREEN

Figure 3. Table of Pin Configurations used for passive components. (Note: external interrupts for the GPIO buttons are configured with falling edge detection and 1KΩ pull-up resistors.)

Furthermore, the counter period of the timer is the incremental counter register for the board that had to be calculated to be compatible with the 75 MHz PLLCLK. Instead of tweaking the prescaler value which divides by the internal clock and becomes unreliable as the number of divisions increases, counter period is proven to be more reliable. In order to achieve 8 ksps or KHz in the ADC and DAC, the counter period must have a clock speed of **9374** as explained below:

75,000,000 (75 MHz) / counter period = 8,000 (8 KHz)
Counter period = 75,000,000 / 8,000 = 9,375
To offset the fact that the clock starts at 0, the counter period is subtracted by 1 which gives the value of 9,375-1 = 9,374.

Record_Audio

Using GPIO and lookup tables of the pins provided by STMicroelectronics for the H7 board, this function would program the board to light up each of the LED (red, yellow, and green chronologically) for one second each using writepin functions. Once the last (green) LED lights up, the ADC would take in inputs from the connected microphone. After collecting the 8000 samples through the microphone, the audio amplifier would subsequently playback the original audio input buffer.

Reverb

The Reverb function creates a new output audio by combining the original audio data with a delayed and diminished copy. Noting that 70ms under 8k Hz sampling rate equates to 560 samples, hence the output audio would be 8000 + 560 samples long. The expression can be written as:

For the first 8000 samples in output:

output audio = original audio + (⅔) original audio delayed by 70ms (i.e.560 samples)

For the last 560 samples in output:

output audio = (⅔) original audio

Hence, the first sample of the delayed signal is scaled and added to the 561th sample of the original audio, and so on.

The function creates a 8560-long array buffer that stores the output array. The array is then populated with the equation above.

SpeedUp

The function outputs a speed up audio of the original by a factor of ⅓. Hence, the output audio's speed is 4/3 of the original. Each 4 input samples is converted to 3 output samples, so the 8000-sample-long input becomes a 6000-sample-long output.

Programmatically, each point of the new array is computed by a weighted linear interpolation between the adjacent samples in the input audio array. We divided the **input array** into **4-sample segments**, which correspond to **3-sample segments in the output**:

1. the 1st output sample is equal to the 1st input sample;
2. the 2nd output sample = (⅔) 2nd input sample + (⅓) 3rd input sample
3. the 3rd output sample = (⅓) 3rd input sample + (⅔) 4th input sample

The idea is further shown by the figure below.

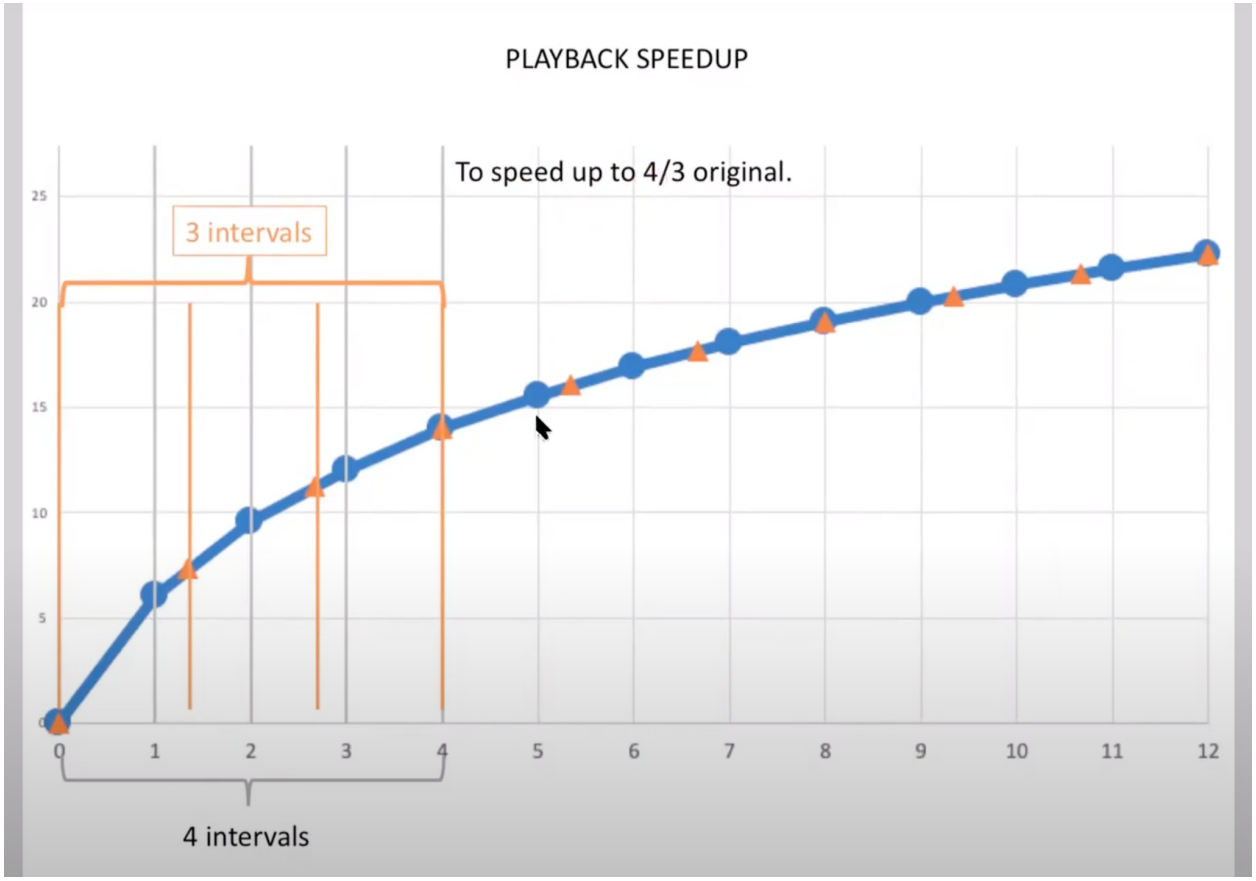

Figure 3.1 Sourced from Dr. Brigg's lecture demonstrating the weighted linear interpolation between data points (orange: output; blue: input)

SpeedDown

This function was straightforward in taking a linear interpolation of the given audio signal. No issue arose in this area due to the little room for arithmetic and logic error. The linear interpolation would ultimately double the initial audio signal sample size of 8000, and the extra points that were added were calculated as averages between each sample of the original audio buffer.