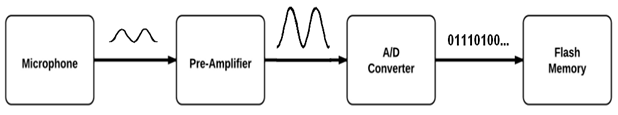
**Background**

The Wave-Motion Arduino Shield was created by Steven Diemer and Jordan Visser, both juniors enrolled in the Computer Engineering program at Grand Valley State University. As part of a semester long project for EGR-326 (Embedded System Design), an Arduino Shield was to be designed and marketed to the public. The basic premise of the design is that the user can attach the Wave-Motion Shield to an Arduino Uno, record a message/sound and play the audio back faster, slower, in reverse, or forwards based on the movement of the shield (of which is captured by an accelerometer).

The shield was designed as a novelty item with the intention of making the software easy to hack to allow for additional functionality. For example, the software could be modified to recognize movements like the beat of a drum and then play back varying drum sounds based on accelerometer readings. This is one example amongst many and by making the software user friendly, hopefully the Arduino Community will use the shield for a variety of sound and motion based creative endeavors.

**Technical Information**

As the shield required audio recording capability as well as audio playback based on accelerometer readings, a variety of hardware was interfaced. To record audio, a signal from a small condenser microphone is first offset to oscillate around 2.5VDC before being passed through a pre-amplifier. The pre-amplifier is tuned to amplify the microphone signal to 2.52.5VDC. This amplified signal is then output to the Analog to Digital Converter (ADC) on the Arduino. The ADC on the Arduino then reads the analog voltage and converts the voltage to an 8-bit value that can be written to on-board flash memory. The microphone signal is continuously amplified, converted by the ADC and written to memory as long as the user holds the record button (or until they run out of memory). The flow of this record operation is shown graphically by Figure 1.

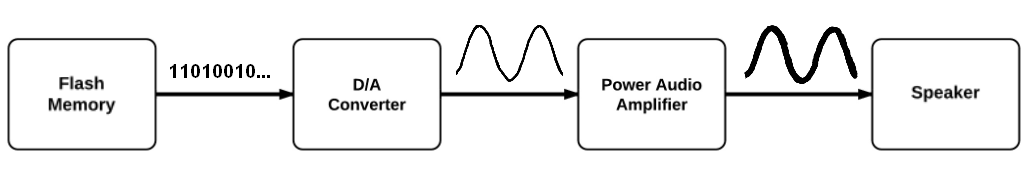


**Figure 1: Audio Recording Flow Chart**

The quality of the audio recording is a function of two variables, the bit-depth of the ADC and sampling rate. The bit-depth of the recording is the number of bits used to characterize a single audio sample. The sampling rate is also very characteristic of the audio quality as it is the frequency at which the audio signal is sampled. In the world of microcontrollers though, these two parameters are fundamentally related to one another. That is, by increasing the sampling rate, bit depth is compromised, and vice versa. To maintain audio integrity at slow playback speeds (2x slower than original recording), it was determined that a 32kHz sampling rate at 8 bit-depth was needed. This sampling rate would ensure that the shield would play back audio at MP3 quality (8-bit/16kHz), even when the playback is reduced to half the native frequency. By using clock prescaler values necessary to obtain 8-bit ADC depth, the maximum sampling rate was determined to be 41kHz, thus exceeding the 32kHz requirement and consequently allowing for playback speeds slower than half of the native frequency. Sampling rate derivations are shown below.

*Sampling Rate:*

To play the recorded audio, 8-bit values are first read from flash memory before being written to the digital to analog converter (DAC). The purpose of the DAC is to take digital values and convert them to an analog voltage that can then be amplified and output to a speaker. This is essentially how the Wave-Motion play-back functionality works. Byte values are first read from flash memory and are then written to the DAC where they are converted to an analog voltage. The DAC however is only capable of driving low current signals. As such, the signal from the DAC is current amplified (effectively increasing the signal’s power) by an on-board audio amplifier. The output of the audio amplifier then drives the on-board speaker. The flow of this playback operation is shown graphically by Figure 2.



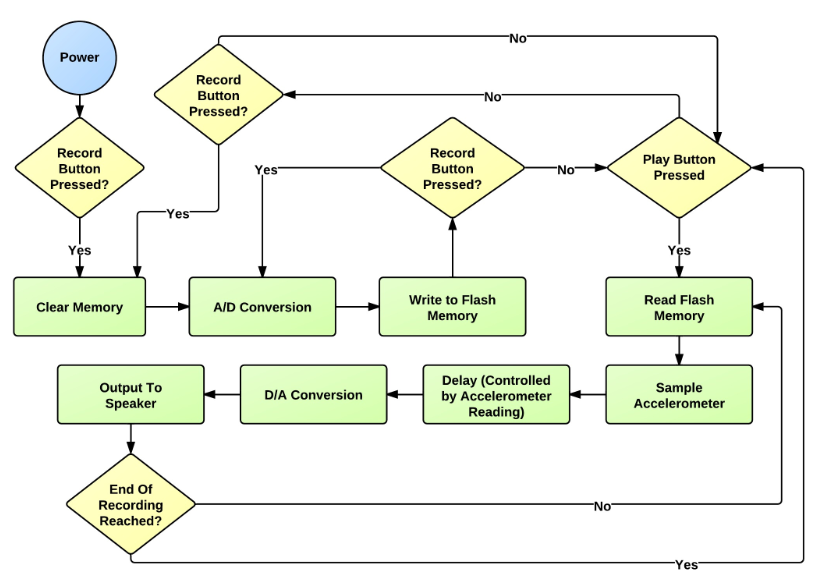
**Figure 2: Audio Playback Flow Chart**

To ensure the audio could be played back at 2x the native recording speed, the DAC conversion and memory read time needed to be twice the native recording speed. Given the selected components, the playback speed of the shield is 87kHz. This allows for playback of the original recording (41kHz) at over twice its speed. Derivations for playback rate are shown below.

*Playback Rate:*

As suggested by the preceding text, the Wave-Motion Shield adjusts the playback speed by increasing or decreasing the rate at which data is read from memory and output to the DAC. This stretches or condenses the audio signal and effectively decreases or increases the frequency of the recording. By periodically sampling the accelerometer while playing back audio, the accelerometer readings are processed and the playback speed is adjusted appropriately. The Wave-Motion features a 3-Axis ±8g which allows for motion processing in the x, y and z-axis. This allows for 3-dimensional motion-based playback control.

The device operation is summarized graphically by Figure 3.



**Figure 3: Flow Chart of Wave-Motion Operation**

**Using the Wave-Motion**

The Wave-Motion Shield is a plug-and-play device. That is, the device can be used by first powering the Arduino through USB or any other 5VDC source and docking the Wave-Motion on top of it. The device comes pre-loaded with software that allows one to record and play back messages while interfacing with the accelerometer. To record, one simply holds the button on the device labeled “RCRD” until they are done recording or until the device beeps (indicating end of memory). To play back the recording, simply press the “PLAY” button once and the message will begin to play back. To increase or decrease the playback speed, one simply needs to tilt the device up or down the in x-y plane.

If one wishes to hack or add functionality to the device, the standard pre-loaded code is made readily available on the Wave-Motion WordPress site along with sample code and documentation that demonstrates how to incorporate pattern recognition into the device. This however requires that the user download and become familiar with the Arduino IDE.