Olin College of Engineering

Digital Signal Processing

DIY AirDrums?

Build Your Own Air-Drum Set at Home with a few simple steps!

by Zhi Hong on 6 May 2025

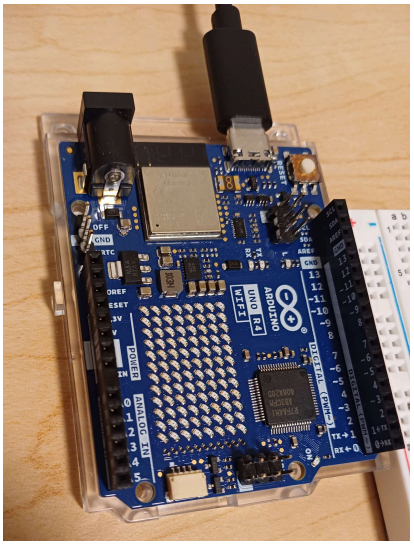
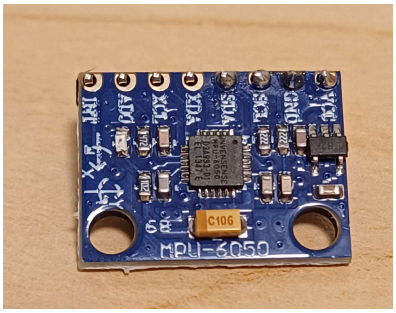
Ever wanted to slam out epic drum beats — Everywhere and anywhere? In this guide, we'll show you how to turn an ordinary drumstick into a wireless AirDrum, using stuff you probably have lying around and some slick Digital signal processing wizardry.

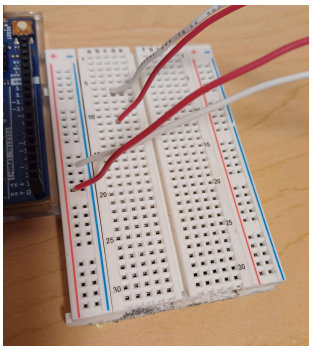
1. Why AirDrums?

- **Budget-Friendly:** No need to spend thousands on electronic drums.
- **Portable:** Play anywhere and everywhere- living room, park, or in the shower.

The Challenge: How do we reliably detect drum hits mid-air and switch between toms, snares, and cymbals — all with a tiny IMU on your stick?

2. Gear up

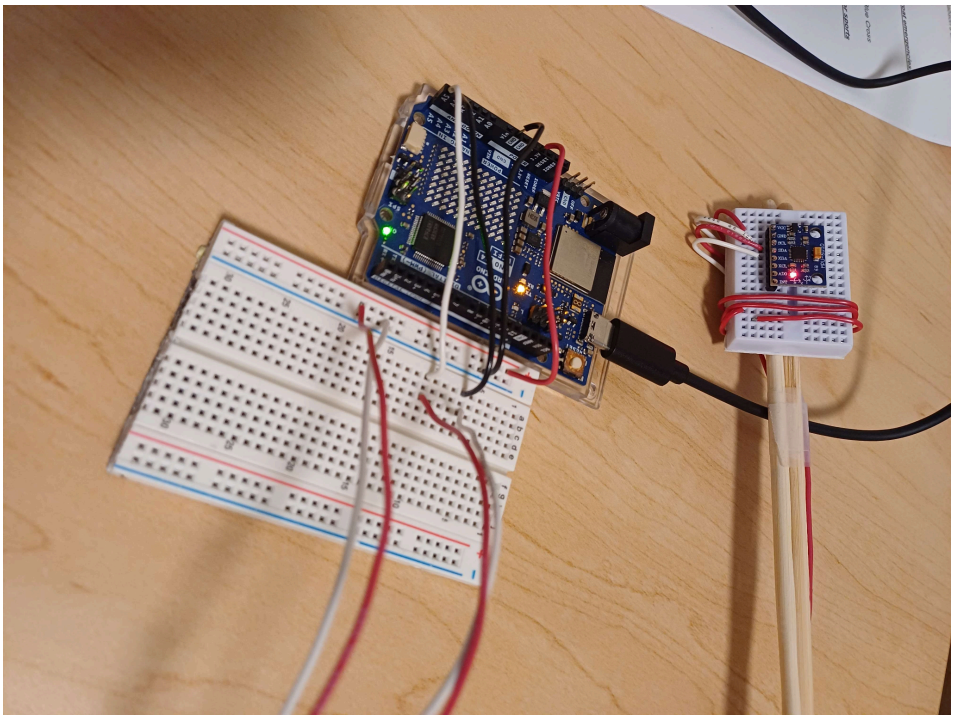
What You Need	Description	Image
1 x Arduino Uno	For DSP and mini PC	
2 x MPU-6050	Acceleration & rotation measuring tool	
Any old drumstick	Something that mimic drumsticks	Chopsticks are fine too!
Zip-ties or tapes	Put everything together	Keep it all snug

Breadboard and wires	For wiring and extension	
USB cable	Power + data to your computer	USB-C cable

3. Assembly 101

1. **Mount the IMU** near the tip with zip-ties or a tiny clamp. Make sure each axis points the right way.
2. **Wire it up** on your breadboard: VCC->5V, GND->GND, SDA->A4, SCL->A5.
3. **Upload the arduino sketch** from our GitHub (see Appendix) to stream ax,ay, az, gx, gy, gz at 200 Hz.
4. **Plug in** via USB and open MATLAB or the serial monitor at 115200 baud
5. **Test swings**: Give it a few practice taps and swings to ensure clean data.

Side note: Want two-handed drumming? Duplicate this setup on a second stick!



4. Software Setup and Modes

Getting your AirDrum up and running is just a few clicks away:

1. **Get the Code**
   Head to my GitHub page at https://github.com/zhi41/DSP-final-project and download or clone the repository to your computer.
2. **Install the softwares**
   - In the Arduino IDE, install the Adafruit_MPU6050 library.
   - In MATLAB, ensure the Signal Processing Toolbox needed is available.
3. **Configure Your Environment**
   - Choose your play mode by running the different modes:
     - **airdrum** — optimized for floating, in-air strokes (uses band-pass).
     - **tapdrum** — optimized for precise pad hits (uses Daubechies wavelet).
4. **Calibrate & Play**
   - Strike your stick in the air or tap a surface to see the Z-axis waveform peak above the red threshold line.
   - Adjust the zHigh, zLow, and gyroThr parameters in real time to match your playing style.

5. Digital signal Processing

Building an AirDrum relies on a series of signal processing stages that extract musical hits from raw IMU data.

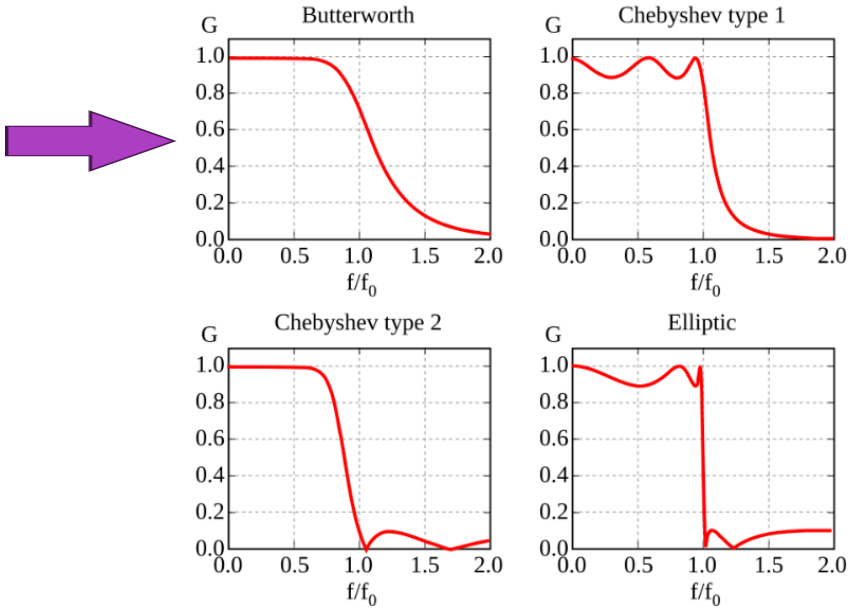
5.1 Sampling at 200 Hz

We set the Arduino to read the MPU-6050 every 5 ms, yielding a 200 Hz sample rate. This rate captures the fast impact transients of a drum hit without flooding the PC with unnecessary data.

5.2 Band-Pass Filter (7–12 Hz)

To visualize and clean up the raw accelerometer data, we apply a **2nd-order Butterworth band-pass** between 7 Hz and 12 Hz. This filter isolates the frequency band where air-drum strokes exhibit most energy (stick swing + reverberation) while rejecting both low-frequency drift (gravity at 0 Hz) and high-frequency sensor noise.

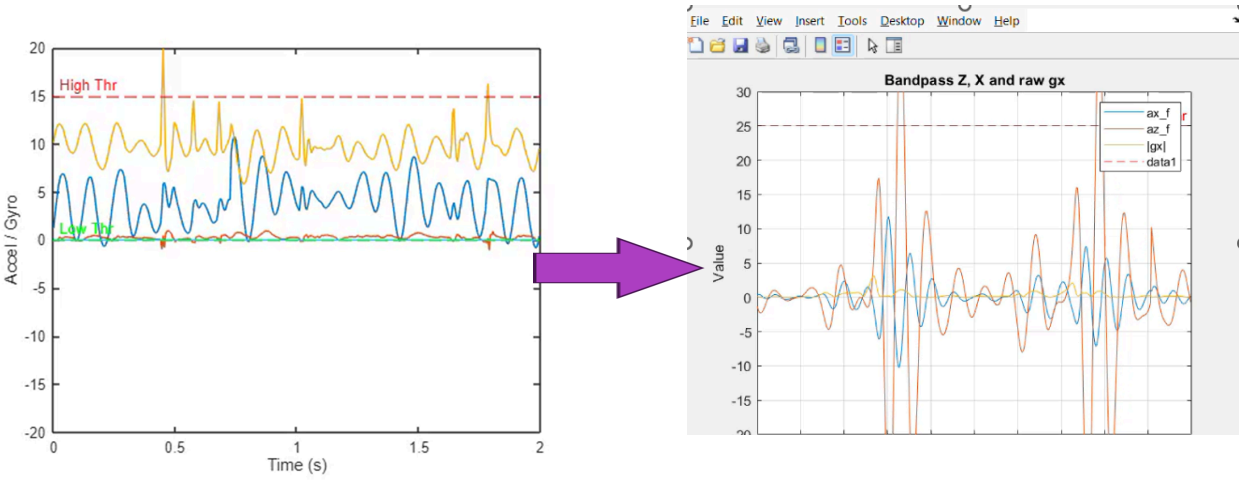
Here is an image showing the gain of a discrete-time Butterworth filter next to other common filter types. All of these filters are fifth-order.



The Butterworth filter rolls off more slowly around the cutoff frequency than the Chebyshev filter or the Elliptic filter, but without ripple.

Image from Wiki

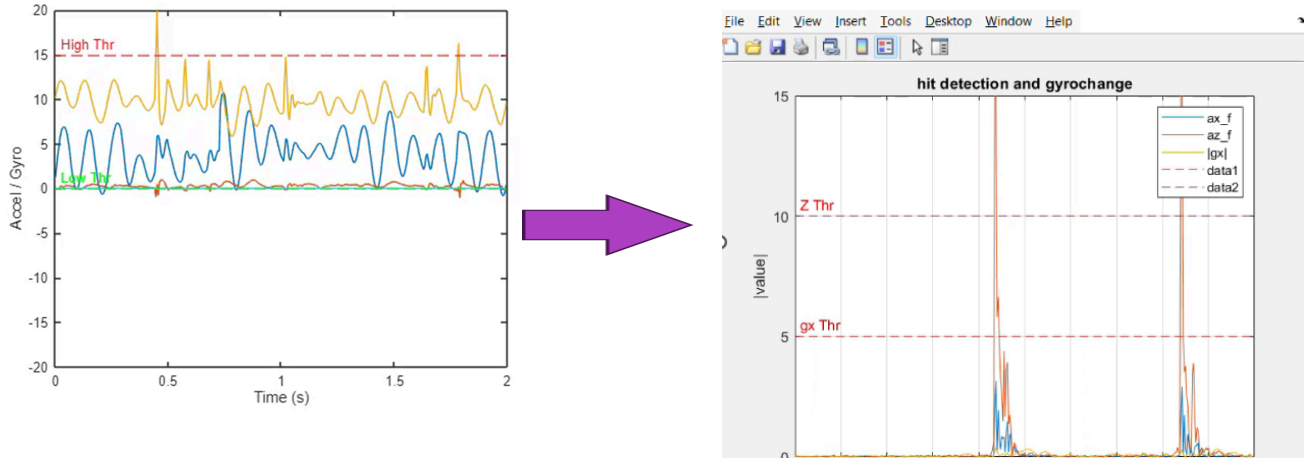
- **Why 7–12 Hz?** That's my Air-drum strokes frequency, DO CHANGE TO YOUR LIKING
- **Benefit:** Removes gravity bias (0 Hz) and high-frequency sensor noise
- **Butterworth filters** maximize flatness in the passband. A 2nd-order already gives a good response from 7 Hz to 12 Hz while removing a good portion of "ripples".
- **In MATLAB** each order adds multiplications and memory for delays. Thus maintaining low order is faster.



5.3 Daubechies-2 Detail for Hit Detection

For precise tap detection, we use a level-1 detail filter from a Daubechies-2 wavelet (4 taps):

- **Why Daubechies-2?** Its short, high-pass impulse response sharply isolates sudden changes (strike of the stick on the table) without long ringing.
- **Short support (4 taps):** Only $(4 - 1)/2 = 1.5$ samples of group delay (~7.5 ms at 200 Hz), so hits come through with minimal lag.
- **Good time localization:** The simple alternating-sign coefficients produce a sharp impulse response that reacts strongly to the sudden onset of a hit but rings very little afterward.
- **No parameter fitting:** Unlike windowed band-pass filters, the db2 wavelet is fixed and orthogonal, so you get consistent behavior across sticks and players.
- **Process:** We feed the Z-axis accel into the detail filter, then take the absolute value of the output. Spikes in this envelope correspond exactly to strike instants.



5.4 Thresholding

Real hits have rebound and mechanical ringing. To avoid double triggers, I implement two thresholds:

1. **High threshold** to detect the initial spike.
2. **Low threshold** to reset (re-arm) once the envelope decays.

This gate ensures one hit produces one sound, even if the stick bounces.

5.5 Gesture-Based Instrument Switching

We assign three drum voices (tom, ride, snare) to left/centre/right positions of the raw gyro-X signal:

- **Flick right** → next instrument ; **flick left** → previous Instrument
- **No extra buttons:** pure angular-velocity mapping for an intuitive performance.

5.6 Audio Pool for Overlapping Playback

MATLAB's audioplayer can play multiple samples simultaneously.

6. Insights

Our DIY AirDrum shines in cost and creativity, but let's zero in on the **delays** and what they mean for playability:

Latency Breakdown

- **IMU sample interval**: 5 ms between readings introduces a base latency of half the period (~2.5 ms on average).
- **Serial transfer**: 115 200 baud sends ~48 bytes (~4 ms).
- **MATLAB processing**: IIR filtering + wavelet detection loops add ~10 ms
- **Audio playback**: The audioplayer startup latency

Possible Optimizations

1. **On-board DSP**: Use a FPGA for faster processing and move the matlab processing tools onto the FPGA to remove the serial transfer lag/
2. **Low-level audio engine**: Replace audioplayer with PortAudio, WebAudio AudioWorklet, or a compiled C++ playback engine to reduce startup delay.
3. **Reduce GUI load**: Update the live plot less frequently (e.g., every 5th sample) to smooth out MATLAB overhead.

Broader Reflections

- **Strengths**: Incredibly low cost, highly customizable, no black-box hardware.
- **Limitations**: Real-time constraints of MATLAB and USB serial, single-axis hit detection misses angled strokes, manual threshold tuning per performer.
- **FPGA:** Faster processing speed and remove serial transfer lag
- **Dynamic Velocity**: Scale sample volume to the peak amplitude of the wavelet envelope
- **Drum sounds:** do drum sound simulation to include the echo and reverberation of the sound according to the striking location

- **Camera:** instead of using gyro to change the instrument, use camera and AI to detect the location of the strike

---

Appendix:

All scripts, sketches, and data live in GitHub repo:
https://github.com/zhi41/DSP-final-project

Special thanks:

Aaron Codrington and Mateo Macias for modelling the front page

Drum sounds taken from BeatLabX is an open-source drum kit website

---