

Integrated Vibration Controller — Quick Start

To begin using the vibration controller, follow these steps:

1. Components

- Vibration control board
- DC supply, 12–48 V, ≥ 1 A, 2.1 mm x 5.5 mm barrel plug (center positive)
- High bandwidth actuator (ex: voice coil)
- Computer or phone

2. Connections

1. **Actuator** -> on the PCB 3-position terminal, connect A (+) & B (-)
2. **Power** -> barrel jack (center positive!)

3. Power

1. Plug in power. The LED turns white (self-test) then blue — idle and safe (actuator braked)
2. With a computer or phone, join the setup WiFi network `irma-setup` (no password); this can take a minute to show up.
3. On a web browser, go to `http://192.168.4.1`
4. *Optional:* in the WiFi row enter your network name and password and press **Connect**. The board joins your network at `http://irma.local` or its assigned IP address can be read on the dashboard.

4. Dashboard

- **Buttons** — **STOP**, **Run** (test), **Engage control**, **Clear fault**
- **Mode** — current control mode of the actuator
- **Time plot** — live 3-axis acceleration + control effort
- **PSD plot** — vibration spectrum, 1–1000 Hz
- **Status** — board temperature, driver state, loop rate, drive activity.
- **Config** — duty limit, overtemp and instability ceilings, accelerometer range (auto or fixed $\pm 10/\pm 20/\pm 40$ g), and the dashboard heartbeat
- **Controller gain** — scales the loaded controller live, so a design can be walked up or down without re-exporting. Negative flips the loop sign.
- **Degradation & Faults** — amber inset lists warnings; red banner lists latched faults.

5. Safety

- **STOP** button brakes the actuator from any state.
- Drive is clipped at the **duty limit** (Config, default 50 %). Lower it before the first closed-loop run if the actuator or structure cannot take full-supply drive.
- The **instability guard** cuts drive if the average command stays high for half a second, catching a misbehaving controller.
- Faults **latch**: drive stays off and the LED blinks red until *Clear fault*, which reboots through a self-test. The **Fault log** panel lists them with timestamps.

6. First test run

Warning: actuator will move!

1. In the **Test** panel choose plant (noise -> actuator), RMS 5 %, dur 2 s (limit 5 s)
2. Press **Run**. The LED goes amber and the actuator drives random noise for 2 s.
3. Press **Download CSV** — a 4 kHz record of all three axes plus the drive command and the injected excitation (t_s, ax_g, ay_g, az_g, drive, excite, after a metadata header line).

Expect the time plot to move during the run and a peak to appear in the PSD at your structure's resonance. Flat plots mean the actuator is not moving — check **DRV nFAULT** and the A/B wiring (§9) before going further.

record (no drive) captures ambient vibration without driving anything.

OLTF, CLTF, and **Engage control** need a controller uploaded first — §7.

7. Designing a controller

Needs a computer with MATLAB, or the project repo and Python.

Note: feedback controller assumes standard cascade series form with zero command and negative feedback. For plant $G(s)$ and controller $H(s)$, OLTf stability determined from $G(s)H(s)$ however CLTF *disturbance rejection* determined by $G(s)/(1+G(s)H(s)) = \text{feedback}(G,H)$.

MATLAB

Run test, download the CSV, and analyze the response. See example below:

```
% Read the CSV file containing the test run data
data = csvread('path_to_file.csv',2,0); % 2 header lines: metadata + column names

% parse CSV columns
t = data(:,1); % time [s]
x = data(:,2); % x accel [g]
y = data(:,3); % y accel [g]
z = data(:,4); % z accel [g]
d = data(:,5); % drive command, duty fraction -1..+1 (NOT volts)
e = data(:,6); % excitation injected this run, same units as drive
Fs = 1/mean(diff(t)); % sample frequency [Hz]

% Which column is the input depends on the test mode:
% plant -> use d: az/drive = P, the plant
% oltf -> use e: az/excite = P*C, the loop gain
% cltf -> use e: az/excite = closed loop, and drive/excite = sensitivity
% In oltf/cltf the `drive` column is the CONTROLLER OUTPUT, so az/drive
% returns the plant again and the controller cancels out of the estimate.

%% FRFs
nfft = 2^10;
window = hanning(nfft);
noverlap = nfft/2;

% Estimate the frequency response function (FRF)
[Z,F] = tfestimate(d,detrend(z),window,noverlap,nfft,Fs);

% create a frequency response data (FRD) object
G = frd(Z,2*pi*F);
bode(G);
```

Design your compensator, discretize, and export. See example below:

```

% design controller (example)
s = tf('s') % Laplace operator
H = tf(1); % initialize
H = H*s/(2*pi*10); % AC-couple for bias rejection
w=2*pi*10;z=0.7;H = H/((s/w)^2 + 2*z/w*s + 1); % rolloff to integrator
H = H*0.5; % final gain adjustment

% plotting
opts = bodeoptions;
opts.Grid = 'on';
opts.FreqUnits = 'Hz';
opts.PhaseWrapping = 'on';
opts.Xlim = {[1 4000]};
bode(G,H,G*H,feedback(G,H),opts);
legend('plant','control','OLTF','CLTF','location','SW')

% discrtime
H_d = c2d(H,1/Fs,'tustin');

% convert
H_ss = ss(H_d);

% check stability
rho = max(abs(eig(H_ss.A)));
if rho >= 1
    error('controller unstable: spectral radius %.4f >= 1', rho);
end

% export
control_filename = 'controller.json';
txt = sprintf(['{"name":"%s","fs_hz":%d,"A":%s,"B":%s,"C":%s,"D":%s,' ...
    '"notes":"exported %s, rho=%.4f"}'], ...
    control_filename, Fs, mat2json(H_ss.A), mat2json(H_ss.B), mat2json(H_ss.C), ...
    mat2json(H_ss.D), datestr(now, 'yyyy-mm-dd'), rho);

fid = fopen(control_filename, 'w');
fwrite(fid, txt);
fclose(fid);

% helper
function s = mat2json(M)
    rows = cell(size(M, 1), 1);
    for i = 1:size(M, 1)
        rows{i} = [' ' strjoin(compose('%17g', M(i, :)), ',') ' '];
    end
    s = [' ' strjoin(rows, ',') ' '];
end

```

8. Engaging Control

1. **Upload controller JSON** -> choose the .json. The board rejects anything malformed, missampled, or unstable.
2. Run a low-amplitude OLTF test and inspect the response for stability margins — want gain margin > 6 dB, phase margin > 30°. Short of that, lower **Controller gain** and re-run; no re-export needed.
3. Run CLTF and compare its spectrum against a plant run **at the same amplitude** to measure real attenuation.
4. Then **Engage control**. Control ramps up over 1 s to limit sudden actuator movement. The loop runs until STOP or a fault. The board runs headless; you can close the browser.

Backing out: press **STOP**, or walk **Controller gain** down toward 0. If the structure starts to sing, the instability guard cuts drive by itself within 0.5 s.

9. Troubleshooting

Symptom	Meaning / fix
No <code>irma-setup</code> network	No power or still booting — check supply, wait 10 s
Download CSV greyed out	Nothing captured yet — it enables once a run has recorded samples. The capture card states why. The buffer resets on every reboot
DRV nFAULT card shows FAULT	Motor supply absent or too low (normal when running from USB alone). Tests that drive the motor are refused until it clears
Run completes but plots stay flat	Actuator not connected, or wired to the wrong terminal positions — use A and B
Controller upload rejected	Not exactly 4 kHz, > 16 states, more than one output, or spectral radius ≥ 1 . Check all four before uploading
LED blinking red	Latched fault — read the banner, fix the cause, press <i>Clear fault</i>
Warning: "DRDY inoperative"	Expected on this unit. The sensor's data-ready output is unusable, so the loop runs on its own 4 kHz timer and samples asynchronously. Measurement and control are valid; it adds a little timing jitter on top of the fixed 1 ms delay
Warning: "CAPTURE DISABLED" or "PSRAM absent"	Capture buffer failed to allocate or fell back to a short one. Tests still run but recording is disabled or truncated. Power-cycle; if it persists, reseal the MCU module
Recorded run shorter than requested	Duration exceeded the buffer (5 s with PSRAM, 1 s without). The dashboard warns when a capture truncates
Plots frozen / "disconnected"	WiFi dropped — rejoin <code>irma-setup</code> or check your LAN. The board keeps running
Can't reach <code>irma.local</code>	Some networks block mDNS — use the board's IP from your router

10. Good to know

- The board never drives the actuator on its own initiative. Power-cycles land in idle; only dashboard commands or a controller you engaged produce motion.
- USB on the MCU module is for firmware and serial logs only — normal operation is fully wireless.
- Firmware updates arrive over-the-air (OTA), i.e. WiFi, and are applied automatically. The board reboots into a self-test after an update.