Functional Specification Document (FSD)
Karl's Ultimate DIY Dyno – Brake & Inertia & EGT
Date: February 17, 20261.

Project Overview

1.1 **PurposeThe Karl's Ultimate DIY Dyno is a low-cost**, ESP32-based dynamometer system for measuring engine performance (RPM, horsepower, torque) on motorcycles, scooters, small engines, or similar applications. It supports two measurement modes:

- Brake mode — uses load cell force on a lever arm + drum RPM to calculate torque and power.
- Inertia mode — uses drum acceleration (inertia of the flywheel/roller) to calculate power. The system is fully self-contained (access point or client WiFi), runs a web dashboard for real-time monitoring, calibration, and run export (PDF).

1.2 Key Hardware• ESP32 (WiFi + processing)

- Adafruit NAU7802 breakout + load cell (force measurement)
- MCP9600 thermocouple amplifier (EGT measurement)
- 2 × Hall sensors (engine RPM + drum RPM)
- Lever arm (adjustable, default 25 cm)
- Drum/roller (inertia mass/diameter configurable)

1.3 Key Software Features

- Real-time web dashboard with gauges (RPM, HP, Torque) + live loadcell kg display
- Ambient temperature display upper right corner
- Left fixed column for EGT display
- Manual tare & calibration (raw ADC, skips settling ramp)
- Auto-run (start/stop at RPM thresholds, customer/unit/comments)
- Brake & inertia calculation modes
- PDF export of run data (peak values, curves)
- Settings page for all parameters (arm length, pulses/rev, filter, WiFi, etc.)
- Safety timeout (45 s max run)

1.4 Hardware Connections / Pinout

The system uses the following ESP32 GPIO pins for peripherals. All connections assume a standard ESP32 DevKit (30-pin or 38-pin version). Use 3.3 V logic levels (NAU7802, Hall sensors, and MCP9600 are 3.3 V compatible).

1.4.1 NAU7802 Load Cell Amplifier

• SDA → ESP32 GPIO 21 (shared I²C bus with MCP9600)
• SCL → ESP32 GPIO 22
• VCC → ESP32 3.3 V (recommended; breakout supports 3–5 V)
• GND → ESP32 GND
• Load cell excitation (E+/E– or red/black) → breakout board's E+ / E– pins
• Load cell signal (A+/A– or white/green) → breakout board's A+ / A– pins Notes:

- Fixed I²C address: 0x2A (no conflict with MCP9600 at 0x60/0x67)
- If readings are inverted, swap A+/A– on the load cell
- Use short wires; I²C is more noise-sensitive — keep bus clean, add 4.7 kΩ pull-ups to 3.3 V if breakout lacks them
- Typical settings: Gain 128 or 64, Channel A, sample rate 10–80 SPS (balance speed vs noise)

1.4.2 Hall Sensor – Engine RPM (Ignition / Crank Trigger)

• Signal → ESP32 GPIO 13 (INPUT_PULLUP enabled)
• VCC → ESP32 3.3 V (or 5 V)
• GND → ESP32 GND1.4.3 Hall Sensor – Drum RPM• Signal → ESP32 GPIO 12 (INPUT_PULLUP enabled)
• VCC → ESP32 3.3 V (or 5 V)
• GND → ESP32 GND1.4.4 Power & Wiring Notes• Stable 5 V supply for ESP32 (USB or regulator)
• NAU7802 breakout prefers 3.3 V for logic, but many support 5 V excitation
• Short, shielded wires for Hall sensors (ignition noise)
• Twisted-pair cable for load cell1.4.5 MCP9600 – EGT Sensor module• SCL → GPIO 22
• SDA → GPIO 21
• VCC → ESP32 3.3 V
• GND → ESP32 GND1.5 Required Arduino Libraries

Library Name	Version (recommended)	Purpose / Author	Installation Method	Required?	Notes
ESPAsyncWebServer	1.2.3 or later	Async web server + WebSocket	Library Manager: "ESPAsyncWebServer"	Yes	Core for dashboard
AsyncTCP	1.1.1 or later	Dependency for ESPAsyncWebServer	Library Manager: "AsyncTCP"	Yes	Required
Adafruit_NAU7802	Latest (1.0.x+)	NAU7802 24-bit ADC driver	Library Manager: "Adafruit NAU7802"	Yes	I²C-based; gain, channel, offset handling; averaging implemented manually
DNSServer	Built-in	Captive portal (AP mode)	ESP32 core	Yes	No install
Preferences	Built-in	NVS settings storage	ESP32 core	Yes	No install
WiFi	Built-in	WiFi AP + STA mode	ESP32 core	Yes	No install
ESPmDNS	Built-in	mDNS (karlsdyno.local)	ESP32 core	Yes	No install
LittleFS	Built-in	Filesystem for static HTML/CSS/JS/favicon	ESP32 core	Yes	Required (split from PROGMEM)

Installation steps:

1. Arduino IDE → Boards Manager → install "esp32" by Espressif (v2.0.17+)
2. Library Manager → install:
    - ESPAsyncWebServer (me-no-dev)
    - AsyncTCP (me-no-dev)
    - Adafruit NAU7802 (by Adafruit)
    - (optional: Adafruit BusIO if not pulled in automatically)
3. LittleFS: Install ESP32 LittleFS Uploader tool (Random Nerd Tutorials guide)

Board settings:

• Board: ESP32 Dev Module
• Partition scheme: "Default 4MB with spiffs" or "Minimal LittleFS (1.9MB APP / 190KB SPIFFS)" or larger
• PSRAM: disabled (unless needed)1.6 Project Folder StructureCurrent layout (after splitting HTML from PROGMEM to LittleFS):

karls-dyno-project/

```
├── karls-dyno.ino          # Main sketch (only .ino file on ESP32)
├── platformio.ini          # (optional) if using PlatformIO
├── data/                   # LittleFS filesystem contents (uploaded to ESP32)
│   ├── index.html          # Main dashboard page
│   ├── settings.html       # Settings page
│   ├── favicon.ico         # Favicon (optional, 16×16 or 32×32 ICO/PNG)
│   └── (future files)      # e.g. style.css, script.js, images
├── src/                    # (optional) if using PlatformIO
│   └── main.cpp            # (symlink or copy of karls-dyno.ino)
├── lib/                    # (optional) local libraries
├── FSD.md                  # This document
└── README.md               # (optional) quick start, wiring photos, etc.
```

How to upload LittleFS files:

1. Install ESP32 Sketch Data Upload tool (or use PlatformIO)
2. Put files in data/ folder
3. Tools → ESP32 Sketch Data Upload (uploads data/ to LittleFS on ESP32)

Note: The .ino file uses LittleFS.begin() and server.serveStatic("/", LittleFS, "/") to serve files from LittleFS.

2. Functional Requirements

2.1 User Roles & Access

• Single user (owner/operator)
• No login — open AP or client WiFi access2.2 Main Dashboard• Live gauges: RPM, HP (hk), Torque (Nm)
• Live text under Torque: Loadcell reading KG: X.XX (green, calculated from torque + lever arm)
• Peak indicators
• Trend charts (last ~400 points)
• Recording modal
• Buttons: TARE, Auto Run (F8), Settings2.3 Calibration Flow• Tare first (no weight)
• Place known weight
• Settings → enter kg → CALIBRATE
• System averages raw readings (e.g. 20–100 samples), computes offset (zero) and scale factor
• Scale factor = known_kg / (avg_raw_with_weight - offset)
• Saved to Preferences (offset and scale as floats), broadcasted via WebSocket

2.4 Auto-Run Flow

• Modal: start/end RPM, customer/unit/comments
• Starts at start + 180 RPM hysteresis
• Stops at end – 180 or 45 s timeout
• Live modal graph (HP & Torque vs RPM)
• Complete: peaks + SAVE PDF2.5 Brake Mode Calculation• Force (N) = kg × 9.81
• Torque (Nm) = force × lever_arm_m
• HP = torque × RPM / 71212.6 Inertia Mode Calculation• Torque = I × α
• HP = torque × RPM / 7121
• I from drum mass/diameter/type2.7 Settings Page• Calibration (tare + known weight + CALIBRATE)
• Current offset & scale factor (displayed)
• Gauge maxes, lever arm (cm), mode, RPM source, pulses/rev, filter, drum params, WiFi2.8 Data & Communication (WebSocket)• data: live values +

recording + peaks
• settings: config values
• info: toast messages
• run_complete: peaks for modal2.9 Non-Functional Requirements• Update rate: ~10 Hz
• Calibration: 5 s
• Timeout: 45 s
• No internet required
• Accuracy: ±1–4% after calibration (potentially better than HX711 due to 24-bit resolution and lower noise)

2.10 Assumptions & Constraints

• Load cell ~1 kg range
• NAU7802 gain/channel set to match load cell (typically gain 128 or 64, channel A)
• No temp compensation
• Dashboard freezes during settings (calibration exclusive)

2.11.1 Next Steps / Open Items

• Real dyno testing
• Fine-tune averaging count and sample rate for stability/noise
• Optional: peak load kg in modal/PDF
• Validate I^2C bus noise with shielded cables in real engine environment