KARL'S ULTIMATE DIY DYNO

Functional Specification Document (FSD)

Firmware Version: 3.69
Build Date: 06 July 2026
Hardware Platform: ESP32-S3 Custom PCB
Document Status: Release Draft

Revisjons History

Revision	Date	Description	Author
3.69 Rev A	xx.xx.2026	Complete revision based on Firmware V3.69	Karl Bø / AI

---

1. Project Overview

1.1 Purpose

Karl's Ultimate DIY Dyno is a compact, WiFi-enabled engine dynamometer designed for motorcycles, scooters, mopeds, ATVs and other small combustion engines.

The system provides professional-grade horsepower and torque measurements using either a load-cell brake dynamometer or an inertia dynamometer. Unlike many hobby dynos, the system is completely self-contained and requires only an ESP32-S3 controller, the dedicated Karl's DIY Dyno PCB, and a standard web browser for operation.

All measurements, configuration, calibration and run management are performed through an integrated web interface hosted directly on the dyno. No dedicated PC software is required.

The primary design goals of the project are:

- High measurement repeatability
- Low hardware cost
- Easy installation
- Wireless operation
- Expandability
- Open architecture

- Professional graphical interface
- Permanent storage of recorded runs

1.2 Supported Operating Modes

Firmware Version 3.69 supports two independent dynamometer operating modes.

Brake Dynamometer Mode

Brake Mode calculates engine torque by measuring the reaction force generated on a load cell connected to the brake arm.

Measured values include:

- Engine RPM
- Brake Drum RPM
- Brake Torque
- Engine Torque
- Horsepower
- Exhaust Gas Temperature
- Air Density Correction
- Gear Ratio
- Peak Values

Brake Mode provides the highest measurement accuracy and is intended for steady-state tuning as well as acceleration testing.

Inertia Dynamometer Mode

Inertia Mode calculates engine output power by measuring the angular acceleration of the dyno drum.

The inertia algorithm determines:

- Drum Angular Velocity
- Angular Acceleration
- Drum Torque
- Engine Torque

- Horsepower
- Coast-down Drag Torque
- Automatic Gear Ratio

No load cell is required while operating in this mode.

1.3 Main Features

Firmware Version 3.69 provides the following functionality:

Measurement

- Engine RPM measurement using Hall sensor
- Drum RPM measurement using Hall sensor
- Load Cell force measurement
- Exhaust Gas Temperature (EGT)
- Ambient Temperature
- Atmospheric Pressure
- Air Density Calculation

Calculations

- Brake Torque
- Engine Torque
- Horsepower
- Automatic Gear Ratio
- Manual Gear Ratio
- Drivetrain Loss Compensation
- Air Density Correction
- Drag Torque Estimation

Recording

- Manual recording
- Automatic recording
- CSV storage

- Run comparison
- Peak HP
- Peak Torque
- Peak EGT
- Customer information
- Vehicle information
- Unit system storage

Connectivity

- WiFi Access Point
- WiFi Client
- Async Web Server
- WebSocket communication
- OTA firmware update
- OTA filesystem update

User Interface

- Live dashboard
- Digital gauges
- Historical run comparison
- Configuration pages
- Sensor status monitoring
- Calibration tools

1.4 Intended Applications

The Karl's Ultimate DIY Dyno has been designed primarily for:

- Motorcycles
- Mopeds
- Scooters
- Go-karts

- Snowmobiles
- ATVs
- Small industrial engines
- Two-stroke racing engines
- Four-stroke engines

The system may also be adapted for other rotating machinery where torque and rotational speed measurements are required.

1.5 System Philosophy

The overall philosophy behind Karl's Ultimate DIY Dyno is to provide professional measurement capability using readily available hardware and open software.

The firmware has been designed around the following principles:

- Real-time operation using FreeRTOS
- High-speed interrupt-driven signal acquisition
- Digital filtering to minimise measurement noise
- Wireless operation without dedicated software
- Persistent configuration storage
- Modular architecture for future expansion
- Automatic sensor detection
- Browser-based user interface

All signal processing and calculations are executed locally on the ESP32-S3 controller. The web interface functions solely as a presentation layer, while the embedded firmware performs all acquisition, filtering, calculations, storage, and system control. This architecture ensures deterministic timing, low communication latency, and consistent measurement performance independent of the client device used to access the dyno.

2. Hardware Architecture

2.1 Overview

The Karl's Ultimate DIY Dyno hardware has been designed as a dedicated embedded measurement platform centred around the ESP32-S3 microcontroller. The hardware architecture integrates engine speed measurement, brake force measurement, exhaust gas temperature monitoring, atmospheric correction and wireless communication into a single PCB.

The design philosophy emphasises:

- Low component count
- High measurement accuracy
- Noise immunity
- Easy installation
- Modular sensor interfaces
- Serviceability
- Future expandability

All external devices connect through dedicated PCB connectors, eliminating loose wiring and reducing installation errors.

2.2 Hardware Block Diagram

The hardware consists of five primary subsystems.

```
 ┌──────────────────────┐
 │   ESP32-S3   │
 │          │
 │ WiFi / Web Server │
 │ WebSocket    │
 │ Signal Processing │
 └──────────┬───────────┘
      │
─────────────┼─────────────
      │
┌──────────┬────────────┼────────────┬────────────┐
```

│ │ │ │ │

▼ ▼ ▼ ▼ ▼

Engine Drum RPM Load Cell EGT BMP280

Hall Hall NAU7802 MCP9600 Pressure/Temp

Sensor Sensor

The ESP32-S3 performs all timing measurements, sensor acquisition, calculations, storage and web communication.

2.3 Controller

The controller is based on an ESP32-S3 development module mounted directly on the custom PCB.

The ESP32-S3 provides:

- Dual-core 32-bit processor
- Integrated WiFi
- USB programming
- Hardware interrupts
- I^2C communication
- Flash storage
- FreeRTOS operating system

All measurement algorithms execute locally on the controller. No cloud services or external processing are required.

2.4 PCB Overview

The custom PCB provides all electrical interfaces required by the firmware.

Functions include:

- ESP32-S3 interface
- Engine RPM connector
- Drum RPM connector

- Trigger input
- Load Cell amplifier
- Thermocouple amplifier
- I^2C expansion
- Power distribution

The PCB has been designed specifically for Firmware Version 3.69 and should be considered the reference hardware platform for this document.

2.5 Connector Overview

Connector	Description
P1	Engine RPM Hall Sensor
P2	Drum RPM Hall Sensor
P3	External Run Trigger
U6	NAU7802 Load Cell Interface
U4	MCP9600 Thermocouple Interface
ESP32-S3	Main Controller

2.6 Engine RPM Sensor

The engine RPM input measures crankshaft or ignition speed using a Hall-effect sensor.

Function

The sensor produces a digital pulse for each detected trigger event.

Each pulse generates an interrupt inside the ESP32-S3 where the elapsed time since the previous pulse is recorded.

Measured interval data is subsequently filtered before RPM calculation.

Electrical Interface

Parameter	Value
Sensor Type	Hall Effect

Parameter	Value
GPIO	GPIO4
Trigger	Interrupt
Pull-up	Internal
Debounce	Software configurable
Signal Type	Digital

2.7 Drum RPM Sensor

The drum RPM sensor measures roller rotational speed.

Unlike engine RPM, the drum sensor may generate multiple pulses per revolution depending on the number of magnets installed.

The firmware supports configurable pulses per revolution.

Electrical Interface

Parameter	Value
Sensor Type	Hall Effect
GPIO	GPIO5
Trigger	Interrupt
Pull-up	Internal
Debounce	Software configurable
Pulses per Revolution	User configurable

2.8 External Trigger Input

An external momentary push button may be connected to the PCB.

The trigger allows hands-free operation without using the keyboard or web interface.

Depending on the current recording state, each button press performs one of the following actions:

1. Open Run Information dialog

2. Start Recording
3. Stop Recording

The firmware debounces the trigger in software to eliminate false activations.

Parameter	Value
Default GPIO	GPIO17
Trigger Type	Falling Edge
Pull-up	Internal
Debounce	500 ms
Configuration	User Selectable

2.9 Load Cell Measurement System

Brake mode utilises a strain-gauge load cell connected through an Adafruit NAU7802 24-bit analogue-to-digital converter.

The NAU7802 communicates with the ESP32-S3 using the I^2C bus.

The firmware continuously converts raw ADC values into force measurements after applying:

- Zero offset compensation
- Calibration factor
- Unit conversion
- Digital smoothing

The resulting force measurement is used to calculate brake torque.

Connector

Pin	Function
A+	Signal Positive
A−	Signal Negative
E+	Excitation Positive
E−	Excitation Negative

2.10 Exhaust Gas Temperature Measurement

Exhaust gas temperature is measured using a K-type thermocouple connected to an MCP9600 thermocouple amplifier.

The MCP9600 performs:

- Cold junction compensation
- Thermocouple linearisation
- Digital conversion

The ESP32-S3 retrieves temperature measurements over I^2C and applies additional averaging to improve display stability while preserving responsiveness during engine acceleration.

Connector

Pin	Function
+	Thermocouple Positive
–	Thermocouple Negative

2.11 Atmospheric Sensor

Atmospheric pressure and ambient temperature are measured using a BMP280 sensor.

These measurements allow the firmware to calculate current air density and compensate horsepower and torque values for environmental conditions.

Measured parameters include:

- Atmospheric Pressure
- Ambient Temperature
- Air Density

These values are continuously updated while the dyno is operating.

2.12 I^2C Communication Bus

The following devices share the common I^2C bus.

Device	Address	Function
NAU7802	0x2A	Load Cell ADC
MCP9600	0x67	Exhaust Gas Temperature
BMP280	0x76 / 0x77	Pressure & Temperature

The firmware automatically detects connected sensors during startup and reports sensor availability to the web interface.

2.13 GPIO Assignment

GPIO	Function
GPIO1	I^2C SDA
GPIO2	I^2C SCL
GPIO4	Engine Hall Sensor
GPIO5	Drum Hall Sensor
GPIO17	External Trigger (default)

Additional GPIOs may be assigned to the external trigger input provided they do not conflict with reserved hardware resources.

2.14 Power Requirements

The PCB is designed for low-power operation.

Primary power is supplied externally to the ESP32-S3 development module.

On-board regulators supply the logic voltages required by the peripheral sensors.

Power consumption varies depending on WiFi activity, sensor update rate and recording status but remains within the capability of a standard USB power source.

2.15 Hardware Design Considerations

The PCB layout has been optimised for measurement stability by separating high-frequency digital circuitry from low-level analogue sensor signals.

Particular attention has been given to:

- Short analogue signal paths
- Stable I^2C routing
- Reliable interrupt inputs
- Low-noise load cell interface
- Modular connector placement
- Easy field servicing

The hardware architecture has been designed to support future firmware expansion without requiring significant PCB redesign.

3. Firmware Architecture

3.1 Overview

Firmware Version 3.69 has been developed as a real-time embedded application for the ESP32-S3 platform. The firmware is responsible for acquiring sensor data, processing measurements, calculating engine performance, managing recorded runs, serving the web interface and maintaining communication with connected clients.

Unlike traditional desktop-based dynamometer software, all measurement algorithms execute locally on the controller. The web browser functions solely as a graphical user interface and does not participate in any calculations.

The firmware has been designed around the following objectives:

- Deterministic real-time operation
- High measurement accuracy
- Modular software architecture
- Efficient use of processor resources
- Low communication latency
- Robust fault tolerance
- Future feature expansion

3.2 Software Architecture

The firmware consists of several independent subsystems operating together.

```
┌─────────────────────────────────────┐
│        ESP32-S3         │
└─────────────────────────────────────┘

      Hardware Layer
          │
   Interrupt Service Routines (ISR)
          │
      Sensor Acquisition
          │
      Signal Processing
          │
     Calculation Engine
          │
 ┌────────────────┼──────────────────┐
 │       │       │
 ▼       ▼       ▼
Recording   WebSocket API   Dashboard
 │       │       │
 └───────────LittleFS────────────┘
```

Each subsystem has a clearly defined responsibility and communicates through shared data structures rather than direct hardware interaction.

This modular architecture simplifies maintenance and allows future firmware expansion without extensive modification of existing code.

3.3 Firmware Execution Model

The firmware operates continuously after power-up.

Once initialization has completed, execution becomes event-driven.

Primary execution sources are:

- Hardware interrupts
- FreeRTOS tasks
- WebSocket events
- Timer-based updates
- User interaction

No continuous polling loops are used for high-speed RPM measurements.

Instead, the firmware reacts immediately to incoming sensor events, ensuring accurate timing even at high engine speeds.

3.4 System Startup Sequence

During power-up the firmware performs a complete hardware initialization.

The startup sequence consists of:

1. Configure GPIO pins
2. Initialize serial communication
3. Initialize I^2C bus
4. Mount LittleFS
5. Load stored preferences
6. Detect connected sensors
7. Initialize WiFi
8. Configure Web Server
9. Configure WebSocket
10. Start FreeRTOS tasks
11. Enable hardware interrupts
12. Enter normal operating mode

Any unavailable sensors are reported to the user without preventing normal operation wherever possible.

3.5 Interrupt Service Routines

Real-time pulse acquisition is handled using hardware interrupts.

Three interrupt routines are implemented.

Interrupt	Purpose
Engine Hall ISR	Measures engine pulse intervals
Drum Hall ISR	Measures drum pulse intervals
GPIO Trigger ISR	External recording control

The interrupt routines are intentionally kept extremely short.

Each interrupt performs only the minimum operations required:

- Timestamp acquisition
- Debounce verification
- Interval calculation
- Circular buffer update

No mathematical calculations are performed inside an interrupt.

This minimises interrupt latency and ensures reliable operation at high RPM.

3.6 Sensor Acquisition Layer

Sensor acquisition forms the foundation of the firmware.

Each sensor operates independently.

Supported sensors include:

- Engine Hall Sensor
- Drum Hall Sensor
- Load Cell
- MCP9600
- BMP280

Sensor update rates vary according to measurement requirements.

Fast-changing signals such as RPM are processed continuously.

Slow-changing signals such as atmospheric pressure are updated at longer intervals to reduce processor load.

3.7 Real-Time Sensor Task

A dedicated FreeRTOS task performs the majority of system processing.

The Sensor Task is responsible for:

- Reading load cell data
- Reading atmospheric data
- Processing RPM measurements
- Calculating gear ratio
- Performing horsepower calculations
- Updating live measurement values
- Broadcasting dashboard updates

The task executes continuously while maintaining deterministic timing.

Separating measurement processing from web communication improves system responsiveness and reduces the possibility of missed sensor events.

3.8 Calibration Task

Calibration is executed as an independent FreeRTOS task.

Separating calibration from the measurement task prevents long averaging operations from interrupting real-time data acquisition.

The calibration task performs:

- Stable sample collection
- Averaging
- Calibration factor calculation
- Non-volatile storage
- User notification

This architecture allows calibration to occur without degrading live measurement performance.

3.9 Data Flow Architecture

The firmware processes data through a structured pipeline.

Hall Sensor Pulse

|

▼

Interrupt Service Routine

|

▼

Interval Buffer

|

▼

Median Filter

|

▼

Exponential Filter

|

▼

Validated RPM

|

▼

Power Calculations

|

▼

LiveData Structure

|

▼

Dashboard

Recording

WebSocket

Every calculated value displayed by the dashboard follows this processing path.

This ensures consistent behaviour across all operating modes.

3.10 Shared Data Model

All calculated values are stored within a central data structure.

Rather than allowing individual modules to calculate values independently, each subsystem updates the shared LiveData structure.

This structure contains:

- Engine RPM
- Drum RPM
- Horsepower
- Engine Torque
- Drum Torque
- Peak Values
- Load Cell
- Gear Ratio
- Air Density
- Atmospheric Pressure
- Ambient Temperature
- Drag Torque

The dashboard, recorder and communication layer all reference this common dataset.

Maintaining a single source of truth prevents inconsistencies between displayed and recorded measurements.

3.11 Recording Engine

The recording subsystem operates independently of the calculation engine.

Recording may be initiated by:

- Manual start
- Automatic start
- GPIO trigger

During an active run, the recorder continuously stores processed measurement points in memory.

When recording is complete the firmware:

- Generates summary statistics
- Calculates peak values
- Stores metadata
- Writes CSV data
- Updates the run index
- Notifies connected clients

This design minimises file system activity during active measurements.

3.12 Configuration Management

User settings are maintained using the ESP32 Preferences library.

Configuration values include:

- Calibration factors
- Filter settings
- Drum dimensions
- Gear ratios
- WiFi credentials
- Sensor parameters
- Operating mode
- Dashboard limits

All configuration changes become immediately active without requiring firmware recompilation.

Only modified values are written to non-volatile memory, reducing flash wear.

3.13 Communication Layer

Communication between the firmware and the browser occurs using asynchronous WebSockets.

Unlike conventional HTTP polling, WebSockets provide full-duplex communication with minimal latency.

The communication layer performs:

- Live telemetry
- Configuration updates
- Calibration commands
- Run management
- Sensor status reporting
- Firmware information
- Recording control

The asynchronous implementation allows multiple browser sessions to remain connected simultaneously without interrupting measurement processing.

3.14 File System Management

Persistent storage is handled using LittleFS.

The file system stores:

- Recorded runs
- Run index
- Web interface files
- Firmware resources

Run files are stored in CSV format to simplify later analysis and export.

Metadata is maintained separately to allow rapid retrieval without reading every recorded file.

3.15 Fault Tolerance

The firmware has been designed to continue operating even if optional hardware is unavailable.

Examples include:

- Missing BMP280
- Missing thermocouple amplifier
- Missing load cell
- Empty run storage

- WiFi connection failure

Where possible, the firmware substitutes default values rather than terminating execution.

This behaviour allows continued operation with reduced functionality while informing the user of unavailable sensors.

3.16 Design Philosophy

The firmware architecture follows several guiding principles:

- Interrupts perform acquisition only.
- Tasks perform calculations.
- Shared structures distribute processed data.
- Communication never blocks measurements.
- Configuration remains persistent.
- Each subsystem has a single well-defined responsibility.

This layered approach improves readability, simplifies future development and provides a robust foundation for additional features in later firmware revisions.

4. Signal Acquisition and Processing

4.1 Overview

The primary objective of the signal processing system is to convert raw electrical sensor signals into stable, repeatable and accurate engineering measurements suitable for horsepower and torque calculations.

Unlike conventional hobby dynamometers that perform simple pulse counting, Firmware Version 3.69 employs a multi-stage processing pipeline consisting of interrupt-based acquisition, digital filtering, adaptive smoothing and validation before measurements are accepted by the calculation engine.

This approach significantly improves measurement stability while maintaining fast response during rapid engine acceleration.

4.2 Signal Processing Pipeline

Every measurement follows a defined processing sequence.

Sensor

↓

Electrical Signal

↓

Interrupt Capture

↓

Timestamp Buffer

↓

Interval Validation

↓

Median Filtering

↓

Exponential Filtering

↓

Adaptive Smoothing

↓

Engineering Units

↓

Calculation Engine

↓

Dashboard / Recording

Each processing stage removes a different class of measurement error while preserving transient engine behaviour.

4.3 Interrupt-Based Measurement

Both engine speed and drum speed are measured using hardware interrupts.

Each incoming Hall pulse immediately triggers an Interrupt Service Routine (ISR).

The ISR performs only four operations:

- Read the current microsecond timer
- Calculate elapsed time since previous pulse
- Verify debounce timing
- Store the interval in a circular buffer

No calculations are performed inside the interrupt.

This design minimises interrupt latency and ensures accurate timing even at high engine speeds.

4.4 Circular Interval Buffers

Instead of using only the most recent pulse, Firmware Version 3.69 stores multiple pulse intervals inside dedicated circular buffers.

Separate buffers are maintained for:

- Engine RPM
- Drum RPM

Using multiple samples provides significantly improved immunity against:

- Electrical noise
- False Hall triggers
- Missing pulses
- Mechanical vibration

The firmware continuously overwrites the oldest interval with each newly acquired measurement, ensuring the buffers always contain the most recent operating conditions.

4.5 Debounce Filtering

Hall sensors are susceptible to false triggering caused by vibration, electrical interference or signal ringing.

To eliminate invalid measurements, the firmware ignores pulses occurring within a configurable debounce period.

Separate debounce values are maintained for:

Signal	Default Behaviour
Engine RPM	Optimised for ignition pulse stability
Drum RPM	Optimised for high-frequency drum magnets

The debounce values may be adjusted by the user through the Settings page without recompiling the firmware.

Proper debounce selection is essential for measurement accuracy.

Values that are too short may allow false pulses, while excessively long values may reject valid pulses at high RPM.

4.6 RPM Interval Validation

Not every measured interval represents a valid engine speed.

Before RPM calculation the firmware validates each interval.

Intervals may be rejected if they indicate:

- Impossible engine speed
- Missing sensor activity
- Timeout conditions
- Excessively short pulse spacing

Invalid intervals are discarded before entering the digital filtering stage.

4.7 Median Filtering

Following interval validation, the firmware applies median filtering.

Unlike moving-average filters, a median filter effectively removes isolated spikes without significantly affecting genuine acceleration.

Advantages include:

- Excellent rejection of electrical noise
- Immunity to single erroneous pulses
- Stable low-speed operation
- Improved repeatability

Median filtering is especially beneficial during ignition systems that exhibit occasional pulse jitter.

4.8 Exponential Smoothing

The median-filtered RPM signal is further processed using exponential smoothing.

The exponential filter reduces high-frequency variation while preserving rapid engine acceleration.

The general form of the filter is:

Filtered Value = α × New Measurement + (1 − α) × Previous Filtered Value

where:

- α = smoothing coefficient

A larger α provides faster response.

A smaller α provides smoother display behaviour.

Firmware Version 3.69 dynamically adjusts filtering characteristics depending on operating conditions.

4.9 Adaptive RPM Processing

Engine acceleration characteristics vary significantly between idle operation and full-throttle acceleration.

To maintain optimum response, the firmware adjusts filtering behaviour according to measured operating conditions.

Higher RPM operation favours rapid response.

Lower RPM operation favours increased stability.

This adaptive approach provides smooth idle indication while maintaining responsive horsepower measurements during acceleration.

4.10 Brake Mode Processing

Brake Mode places priority on measurement stability.

The processing sequence is:

1. Capture Hall pulse
2. Validate interval
3. Median filtering
4. Exponential filtering
5. Torque calculation
6. Horsepower calculation

Brake Mode assumes relatively steady engine loading and therefore applies stronger smoothing than Inertia Mode.

4.11 Inertia Mode Processing

Inertia Mode presents significantly greater processing challenges because power calculations depend upon angular acceleration rather than steady-state force.

The firmware therefore performs additional calculations including:

- Drum acceleration
- Angular velocity
- Angular acceleration
- Drag estimation
- Coast-down analysis

These calculations require significantly faster response than Brake Mode while maintaining acceptable noise rejection.

4.12 Drum Acceleration Calculation

Drum acceleration is determined from successive drum RPM measurements.

The firmware continuously calculates the rate of change in angular velocity.

Positive acceleration represents engine power input.

Negative acceleration represents mechanical losses during coast-down.

The resulting acceleration is subsequently used for inertia-mode torque calculations.

4.13 Drag Torque Estimation

During coast-down the firmware estimates parasitic drivetrain losses.

These losses include:

- Bearing friction
- Tyre rolling resistance
- Air resistance
- Mechanical drag

Rather than treating drag as constant, the firmware continuously estimates drag torque during deceleration.

This value may subsequently be used for improved inertia calculations and future compensation algorithms.

4.14 Gear Ratio Tracking

The firmware continuously estimates the relationship between engine RPM and drum RPM.

Each new measurement contributes to an adaptive gear ratio model.

The tracking algorithm evaluates:

- Ratio stability
- Measurement variance
- Confidence level
- Historical behaviour

Only sufficiently stable ratios are accepted for power calculations.

When confidence decreases below the required threshold, the firmware automatically falls back to the configured manual ratio.

This prevents unstable ratio estimates from affecting calculated horsepower.

4.15 Load Cell Processing

Brake Mode utilises measurements from the NAU7802 analogue-to-digital converter.

The processing sequence consists of:

1. Raw ADC reading
2. Zero offset removal
3. Calibration scaling
4. Unit conversion
5. Force calculation

The calculated force is then multiplied by the configured brake arm length to determine brake torque.

4.16 Exhaust Gas Temperature Processing

The MCP9600 performs thermocouple conversion and cold-junction compensation internally.

Firmware processing subsequently performs:

- Sample averaging

- Peak temperature tracking
- Ambient reference storage
- Recording of peak EGT during runs

This provides stable displayed temperatures while preserving rapid response to engine load changes.

4.17 Atmospheric Correction

The BMP280 continuously measures:

- Atmospheric pressure
- Ambient temperature

The firmware calculates air density using the ideal gas equation.

Calculated air density is then compared with the standard atmospheric reference.

Horsepower and torque are corrected to reduce variations caused by environmental conditions.

This improves repeatability between tests conducted under different weather conditions.

4.18 Noise Rejection Strategy

Measurement stability is achieved through multiple independent filtering stages rather than relying on a single aggressive filter.

The processing chain includes:

- Debounce filtering
- Interval validation
- Circular buffering
- Median filtering
- Exponential smoothing
- Adaptive response
- Confidence evaluation

Each stage targets a different source of measurement error.

This layered approach allows the firmware to reject erroneous data while maintaining rapid response during transient engine conditions.

4.19 Data Integrity

Only validated engineering values are distributed to the remainder of the firmware.

Raw sensor data is never transmitted directly to:

- Dashboard
- Recording subsystem
- Power calculations
- WebSocket communication

Every subsystem operates exclusively on processed measurements.

This guarantees that displayed values, recorded runs and exported data originate from the same validated signal processing pipeline.

4.20 Summary

The signal acquisition system implemented in Firmware Version 3.69 combines high-speed interrupt acquisition with multiple stages of digital filtering to achieve stable and repeatable measurements.

By separating acquisition, validation, filtering and calculation into independent processing stages, the firmware provides accurate real-time measurements while remaining robust against electrical interference, mechanical vibration and transient operating conditions.

The resulting measurement architecture forms the foundation upon which all subsequent torque, horsepower and recording functions are built.

5. Calculation Engine

5.1 Overview

The Calculation Engine is the computational core of the Karl's Ultimate DIY Dyno firmware. Its responsibility is to transform validated sensor measurements into meaningful engineering values representing engine performance.

Unlike the Signal Acquisition subsystem, which focuses on measurement integrity, the Calculation Engine performs mathematical modelling of the complete dynamometer system.

Primary calculated values include:

- Engine Speed (RPM)
- Drum Speed (RPM)
- Brake Torque
- Engine Torque
- Horsepower
- Brake Horsepower
- Angular Velocity
- Angular Acceleration
- Gear Ratio
- Air Density Correction
- Peak Values
- Coast-down Drag

All calculations operate exclusively on validated data produced by the Signal Processing subsystem.

5.2 Calculation Sequence

The firmware performs calculations in a defined order.

Validated Sensor Data

│

▼

RPM Calculation

│

▼

Gear Ratio

│

▼

Brake / Inertia Selection

│

▼

Torque Calculation

│

▼

Horsepower Calculation

│

▼

Environmental Correction

│

▼

Peak Detection

│

▼

LiveData Structure

This sequence ensures that every calculation is based on stable, validated measurements.

5.3 Engine Speed Calculation

Engine speed is determined from the elapsed time between consecutive Hall sensor pulses.

The firmware measures pulse intervals with microsecond resolution and converts the measured interval into rotational speed.

The calculation accounts for:

- Pulses per revolution
- Missing pulses
- Debounce validation

- Timeout detection

The resulting RPM value becomes the primary reference for all subsequent calculations.

5.4 Drum Speed Calculation

Drum rotational speed is calculated using the same methodology as engine RPM but allows multiple pulses per revolution.

Because different dyno drums may contain different numbers of trigger magnets, the pulses-per-revolution value is fully configurable.

This flexibility allows the firmware to support a wide variety of drum configurations without modification.

5.5 Brake Torque Calculation

When operating in Brake Mode, torque is calculated directly from the measured reaction force acting on the brake arm.

The relationship is:

Torque = Force × Lever Arm Length

Where:

- Force is obtained from the calibrated load cell.
- Lever arm length is user configurable.

The calculated brake torque represents the torque applied directly to the brake assembly.

This value forms the basis for engine torque and horsepower calculations.

5.6 Engine Torque

Engine torque is derived from measured brake torque together with the current transmission ratio.

Depending upon the selected operating mode, the firmware uses either:

- Automatic Gear Ratio Tracking
- User-defined Manual Gear Ratio

This conversion allows brake torque measured at the dyno roller to be represented as estimated crankshaft torque.

5.7 Horsepower Calculation

Horsepower is calculated from engine torque and engine speed.

The firmware continuously updates horsepower using the current measured engine RPM and calculated engine torque.

The calculation follows the standard mechanical relationship between torque and rotational speed.

Horsepower is recalculated each processing cycle, allowing smooth real-time updates throughout an acceleration run.

5.8 Brake Horsepower

In Brake Mode the calculated horsepower represents the power delivered to the brake mechanism.

Additional correction factors may be applied to estimate engine output depending on drivetrain efficiency and operating configuration.

This distinction allows future firmware revisions to support both wheel horsepower and estimated crankshaft horsepower.

5.9 Inertia Mode Calculations

When operating without a load cell, engine output is determined from the rotational acceleration of the dyno drum.

The firmware continuously calculates:

- Angular velocity
- Angular acceleration
- Effective drum inertia
- Mechanical losses

Torque is then calculated from Newton's Second Law for rotational systems.

Unlike Brake Mode, Inertia Mode derives torque indirectly from acceleration rather than measured force.

5.10 Angular Velocity

Drum RPM is converted into angular velocity before inertia calculations begin.

Angular velocity provides the mathematical representation required for rotational dynamics.

This intermediate value is used internally and is generally not displayed to the operator.

5.11 Angular Acceleration

Angular acceleration is determined from the rate of change of angular velocity.

Positive acceleration indicates engine power input.

Negative acceleration indicates deceleration or coast-down.

The firmware continuously evaluates angular acceleration during active recording.

5.12 Coast-Down Drag Estimation

Mechanical systems exhibit losses due to friction and aerodynamic drag.

Firmware Version 3.69 estimates these losses by observing drum deceleration when engine torque is removed.

Estimated drag includes:

- Bearing friction
- Roller friction
- Tyre deformation
- Windage
- Mechanical drivetrain losses

This estimate improves the accuracy of inertia-based power calculations and provides a foundation for future automatic correction methods.

5.13 Automatic Gear Ratio Tracking

Automatic Gear Ratio Tracking continuously estimates the relationship between engine speed and drum speed.

Rather than accepting every instantaneous measurement, the firmware evaluates ratio stability over time.

Parameters considered include:

- Short-term variation
- Historical consistency
- Confidence level
- Measurement stability

Only ratios meeting the required confidence criteria are accepted for power calculations.

If confidence falls below the required threshold, the firmware automatically reverts to the configured manual ratio until stability has been restored.

This prevents transient disturbances such as wheel slip, clutch operation or rapid gear changes from introducing calculation errors.

5.14 Air Density Compensation

Engine performance varies with atmospheric conditions.

Firmware Version 3.69 continuously measures:

- Atmospheric pressure
- Ambient temperature

These measurements are used to calculate the current air density.

The calculated density may then be used to compensate measured horsepower and torque, improving repeatability between tests conducted under different environmental conditions.

5.15 Peak Detection

Throughout every recorded run the firmware continuously evaluates calculated values for new maximums.

Peak values recorded include:

- Maximum Horsepower
- Maximum Torque

- Maximum Engine RPM
- Maximum Drum RPM
- Maximum EGT

Peak values are stored together with the recorded run and displayed in the dashboard and comparison views.

5.16 Live Data Generation

Following completion of each calculation cycle, all engineering values are written to the shared LiveData structure.

This structure serves as the authoritative source for:

- Dashboard displays
- WebSocket telemetry
- CSV recording
- Run comparison
- Export functions

Because every subsystem references the same dataset, displayed values and recorded values remain fully synchronized.

5.17 Calculation Accuracy

The Calculation Engine has been designed to prioritise repeatability over excessive numerical responsiveness.

Measurement stability is achieved through:

- Validated sensor inputs
- Multi-stage digital filtering
- Adaptive processing
- Confidence-based gear tracking
- Continuous environmental monitoring

The objective is not merely to calculate horsepower, but to provide results that are consistent across repeated tests performed under comparable operating conditions.

5.18 Design Philosophy

The Calculation Engine separates measurement from mathematics.

Sensor acquisition is responsible only for obtaining accurate physical measurements.

The Calculation Engine is responsible for interpreting those measurements according to established engineering principles.

This separation simplifies maintenance, improves software modularity and allows future enhancements to the mathematical models without modifying the underlying sensor acquisition system.

6. Operating Modes

6.1 Overview

Firmware Version 3.69 supports multiple operating modes, allowing the Karl's Ultimate DIY Dyno to accommodate different dynamometer configurations and testing methodologies.

Each operating mode is optimised for a specific measurement principle while maintaining a consistent user interface and data recording format.

The firmware automatically adapts its calculation engine, filtering characteristics and recording behaviour according to the selected operating mode.

Supported operating modes include:

- Brake Dynamometer Mode
- Inertia Dynamometer Mode
- Manual Gear Ratio Mode
- Automatic Gear Ratio Mode

These modes may be combined where appropriate to suit the specific test configuration.

6.2 Brake Dynamometer Mode

Brake Mode is the primary operating mode for steady-state engine testing and controlled acceleration measurements.

In this mode, engine output is measured directly through the braking system using a calibrated load cell.

The firmware continuously measures the reaction force acting on the brake arm and converts this force into brake torque.

Because the measurement is based on direct force rather than calculated acceleration, Brake Mode provides the highest measurement accuracy and repeatability.

Typical applications include:

- Engine tuning
- Carburettor adjustment
- EFI mapping
- Ignition timing optimisation
- Continuous load testing
- Thermal testing
- Endurance testing

6.3 Brake Mode Data Flow

Engine

↓

Engine RPM

↓

Brake Drum

↓

Load Cell Force

↓

Brake Torque

↓

Engine Torque

↓

Horsepower

↓

Dashboard

Recording

The measurement chain remains active throughout the entire recording period.

6.4 Brake Mode Characteristics

Advantages include:

- High repeatability
- Stable measurements
- Direct torque measurement
- Suitable for long-duration testing
- Excellent low RPM performance
- Minimal mathematical assumptions

Limitations include:

- Requires calibrated load cell
- Requires braking mechanism
- Mechanical installation is more complex

6.5 Inertia Dynamometer Mode

Inertia Mode determines engine output by measuring the acceleration of a rotating mass.

Instead of measuring force directly, the firmware calculates torque from the known rotational inertia of the dyno drum.

This method eliminates the need for a brake assembly while still providing reliable horsepower measurements during acceleration runs.

Typical applications include:

- Performance comparison
- Quick tuning sessions
- Portable dyno installations
- Lightweight test benches
- Educational use

6.6 Inertia Mode Data Flow

Engine

↓

Engine RPM

↓

Drum RPM

↓

Angular Velocity

↓

Angular Acceleration

↓

Drum Torque

↓

Engine Torque

↓

Horsepower

Unlike Brake Mode, every calculated value depends upon accurate acceleration measurements.

6.7 Inertia Mode Characteristics

Advantages include:

- No load cell required
- Simple mechanical construction
- Lower hardware cost
- Minimal maintenance

- Fast acceleration testing

Limitations include:

- Sensitive to noise
- Dependent on accurate drum inertia
- Greater mathematical complexity
- Less suitable for steady-state tuning

6.8 Automatic Gear Ratio Mode

Automatic Gear Ratio Mode continuously estimates the transmission ratio between engine speed and drum speed.

The firmware evaluates the stability of the measured ratio over time before accepting it as valid.

The algorithm monitors:

- Ratio consistency
- Measurement variance
- Historical stability
- Confidence level

Once sufficient confidence has been established, the calculated ratio becomes the active transmission ratio for engine torque calculations.

Automatic Gear Ratio Mode significantly reduces operator workload and eliminates manual ratio entry for most applications.

6.9 Manual Gear Ratio Mode

Manual Gear Ratio Mode allows the operator to specify a fixed transmission ratio.

The configured value remains constant throughout the recording session.

Manual mode is recommended when:

- Testing fixed-ratio transmissions
- Verifying automatic ratio calculations
- Operating without reliable drum measurements

- Diagnosing drivetrain behaviour

The manually entered ratio overrides automatic estimation whenever selected by the user.

6.10 Recording Modes

The firmware supports multiple recording methods.

Manual Recording

Recording begins and ends under direct operator control.

This mode is suitable for workshop tuning where the operator wishes to decide precisely when data collection starts and stops.

Automatic Recording

Automatic recording begins when operating conditions satisfy predefined criteria.

Typical trigger conditions include:

- Minimum engine speed
- Minimum drum speed
- Stable sensor operation

Recording automatically terminates when engine activity falls below the configured threshold.

This mode simplifies repeated testing by reducing operator interaction.

External Trigger Recording

An external push-button connected to the trigger input may also control recording.

Depending upon the current system state, the trigger performs:

- Open Run Information
- Start Recording
- Stop Recording

This allows convenient operation while standing beside the motorcycle.

6.11 Dashboard Operating Mode

The dashboard remains active regardless of the selected operating mode.

It continuously displays:

- Live Engine RPM
- Drum RPM
- Torque
- Horsepower
- EGT
- Atmospheric Conditions
- Gear Ratio
- Sensor Status

Display behaviour automatically adapts to the active measurement mode.

For example, Brake Mode presents measured brake force while Inertia Mode emphasises drum acceleration and calculated inertia values.

6.12 Sensor Operating States

The firmware continuously monitors sensor availability.

Each sensor may exist in one of three operational states:

State	Description
Active	Sensor operating normally
Unavailable	Sensor not detected during startup
Fault	Sensor communication lost during operation

Unavailable sensors do not necessarily prevent dyno operation.

The firmware enables all functions that remain possible with the currently available hardware.

6.13 Measurement State Machine

During operation the firmware transitions through several internal operating states.

Power Up

↓

Initialization

↓

Idle

↓

Monitoring

↓

Ready

↓

Recording

↓

Run Complete

↓

Save Results

↓

Idle

Each state has clearly defined entry and exit conditions.

Separating operational states simplifies firmware maintenance while ensuring predictable system behaviour.

6.14 Mode Selection

The active operating mode is selected through the Settings interface.

Changing operating mode updates:

- Calculation engine
- Recording logic
- Dashboard presentation
- Data validation
- Configuration parameters

Some configuration values are retained independently for each operating mode, allowing rapid switching between different dyno configurations.

6.15 Operational Safety

Changing operating mode during an active recording session is prohibited.

The firmware requires recording to be stopped before:

- Switching between Brake and Inertia modes
- Changing gear ratio mode
- Performing calibration
- Modifying hardware-related parameters

This prevents inconsistent datasets and protects measurement integrity.

6.16 Future Expansion

The operating mode architecture has been designed to support additional measurement methods in future firmware revisions.

Potential future additions include:

- Eddy-current brake control
- Automatic load scheduling
- Closed-loop RPM control
- Constant torque testing
- Constant horsepower testing
- Multi-stage acceleration testing
- Automatic test sequencing

Because operating modes are implemented as independent functional layers, these features may be integrated with minimal modification to the existing firmware architecture.

6.17 Summary

The operating mode architecture implemented in Firmware Version 3.69 provides a flexible framework capable of supporting a wide variety of dynamometer configurations.

By separating measurement methodology from the underlying acquisition and calculation systems, the firmware achieves high modularity while maintaining a consistent user experience across all supported operating modes.

This design allows the Karl's Ultimate DIY Dyno to function equally well as a precision brake dynamometer, a portable inertia dynamometer, or a hybrid system capable of future expansion.

7. Data Management & Communications

7.1 Overview

Firmware Version 3.69 incorporates an integrated communication and data management subsystem that provides real-time interaction between the embedded controller and one or more connected client devices.

Unlike traditional dynamometer systems that rely on external software for configuration and recording, the Karl's Ultimate DIY Dyno hosts its own web application directly on the ESP32-S3. All configuration, calibration, monitoring, and run management are performed through a standard web browser.

The communication subsystem is responsible for:

- Wireless networking
- Web server operation
- Live telemetry
- User configuration
- File management
- Run recording
- Firmware updates
- Persistent configuration storage

Together these functions create a completely self-contained measurement platform.

7.2 Communication Architecture

The communication subsystem is divided into several independent layers.

```
   Web Browser
     │
     ▼
  WiFi Network
     │
     ▼
 Async Web Server
     │
┌────────┴────────┐
│        │
▼        ▼
Static Web UI   WebSocket API
```

```
      │              │
      └──────────┬──────────┘
                 ▼
          Firmware Core
                 │
      ┌──────────┼──────────┐
      ▼          ▼          ▼
 Preferences  LittleFS   LiveData
```

Each layer has a clearly defined responsibility and communicates through well-defined interfaces.

7.3 WiFi Operation

The ESP32-S3 provides integrated IEEE 802.11 wireless networking.

Firmware Version 3.69 supports both:

- Access Point (AP) Mode
- Station (STA) Mode

Access Point Mode

Access Point Mode allows direct connection to the dynamometer without requiring existing network infrastructure.

Typical applications include:

- Portable dyno installations
- Race tracks
- Workshops
- Field service

In this configuration the ESP32-S3 operates as the network host.

Station Mode

In Station Mode the dyno joins an existing wireless network.

Advantages include:

- Internet access
- Multiple client devices
- Firmware downloads
- Remote access
- Integration into workshop networks

Stored credentials are retained in non-volatile memory and automatically reused during startup.

7.4 Embedded Web Server

The firmware hosts an asynchronous web server that serves the complete graphical user interface directly from internal flash storage.

The web server is responsible for delivering:

- HTML pages
- JavaScript
- CSS
- Images
- Icons
- Configuration pages

Because all resources are served locally, no Internet connection is required for normal operation.

The browser functions solely as a presentation layer.

All calculations remain within the firmware.

7.5 WebSocket Communication

Real-time communication between the browser and firmware is implemented using asynchronous WebSockets.

This architecture provides several advantages over conventional HTTP polling:

- Low latency
- Bidirectional communication

- Reduced network traffic
- Immediate updates
- Efficient processor utilisation

The firmware broadcasts live measurement data while simultaneously accepting commands from connected clients.

Typical WebSocket traffic includes:

- Live RPM
- Horsepower
- Torque
- EGT
- Atmospheric conditions
- Gear ratio
- Recording status
- Sensor health
- Configuration updates

7.6 JSON Message Format

Communication between firmware and browser uses structured JSON messages.

Typical message categories include:

Live Telemetry

Continuously updated engineering values.

Examples:

- Engine RPM
- Drum RPM
- Horsepower
- Torque
- EGT
- Air Density

Configuration Messages

Used for:

- Saving settings
- Loading settings
- Updating calibration
- Operating mode selection

Command Messages

Examples include:

- Start Recording
- Stop Recording
- Delete Run
- Save Run
- Calibrate Load Cell
- Zero Load Cell

Status Messages

Provide feedback regarding:

- Sensor availability
- WiFi connection
- Recording progress
- Storage usage
- Firmware version

7.7 Live Dashboard

The dashboard represents the primary user interface.

Rather than polling for values, the dashboard receives continuous updates through the WebSocket connection.

Displayed information includes:

- Live Engine RPM
- Drum RPM
- Horsepower
- Torque
- Brake Force
- EGT
- Ambient Temperature
- Atmospheric Pressure
- Air Density
- Gear Ratio
- Peak Values
- Recording Status

Display refresh occurs automatically whenever new processed measurements become available.

7.8 Persistent Configuration

User configuration is stored using the ESP32 Preferences library.

This allows settings to survive:

- Power loss
- Firmware restart
- Browser refresh
- WiFi reconnection

Typical stored parameters include:

- Calibration factor
- Load cell offset
- Drum diameter

- Brake arm length
- Filter settings
- Gear ratio
- Trigger configuration
- WiFi credentials
- User preferences

Only modified values are written to flash memory, reducing unnecessary write cycles.

7.9 File System

Firmware Version 3.69 uses LittleFS as its primary file system.

LittleFS provides a lightweight and robust flash-based storage solution suitable for embedded systems.

The file system stores:

- Recorded runs
- Dashboard resources
- HTML files
- JavaScript
- CSS
- Icons
- Images

The firmware automatically mounts the file system during startup.

If mounting fails, the firmware reports the error and continues operating with reduced functionality where possible.

7.10 Run Recording

During an active recording session the firmware collects processed engineering values and stores them in memory.

Upon completion of the run:

- Peak values are calculated

- Metadata is generated
- Customer information is attached
- Vehicle information is attached
- Environmental data is recorded
- The run is written to LittleFS

Each recorded run becomes available for later review and comparison through the dashboard.

7.11 CSV Data Format

Recorded measurements are stored using the Comma Separated Value (CSV) format.

Each row represents a single measurement point recorded during the run.

Typical recorded parameters include:

- Timestamp
- Engine RPM
- Drum RPM
- Horsepower
- Engine Torque
- Brake Torque
- Brake Force
- EGT
- Air Density
- Atmospheric Pressure

The CSV format allows recorded runs to be imported directly into spreadsheet software or external analysis tools.

7.12 Run Index

In addition to individual run files, the firmware maintains an index containing summary information for all stored recordings.

The run index includes:

- Run identifier
- Customer name
- Vehicle description
- Date and time
- Peak Horsepower
- Peak Torque
- Peak EGT
- Operating mode

Maintaining a separate index allows the dashboard to present available runs rapidly without reading every CSV file individually.

7.13 Run Management

The firmware provides integrated management functions for recorded runs.

Supported operations include:

- Create
- Save
- Rename
- Delete
- Compare
- Export
- Reload

These functions are accessible through the browser interface without requiring direct access to the internal file system.

7.14 Firmware Updates

Firmware Version 3.69 supports Over-The-Air (OTA) updates.

Separate update mechanisms exist for:

- Firmware
- Web interface resources

- File system content

OTA updates allow maintenance without physically connecting a programming cable.

Future versions may extend this mechanism to include automated version verification and update notifications.

7.15 Multiple Client Support

The asynchronous communication architecture allows multiple browser sessions to remain connected simultaneously.

Examples include:

- Workshop monitor
- Laptop
- Tablet
- Mobile phone

Each client receives identical live measurement data while maintaining independent browser sessions.

This architecture is particularly useful during demonstrations, tuning sessions and educational environments.

7.16 Fault Handling

The communication subsystem continuously monitors connection status.

Typical events include:

- Client connected
- Client disconnected
- WiFi lost
- WiFi restored
- File system unavailable
- Storage full

Whenever possible, communication failures do not interrupt measurement processing.

The dyno continues acquiring and calculating data independently of browser connectivity.

This separation ensures measurement integrity even if wireless communication is temporarily interrupted.

7.17 Data Integrity

To maintain consistency throughout the system, all transmitted and recorded values originate from the same validated LiveData structure.

The firmware never transmits partially processed sensor values.

Consequently:

- Dashboard values
- Recorded runs
- WebSocket telemetry
- Exported CSV files

all contain identical engineering data.

This design guarantees that every displayed measurement can be traced directly to the corresponding recorded dataset.

7.18 Design Philosophy

The communication subsystem has been designed with the same philosophy as the remainder of the firmware:

- Calculations remain local.
- Browsers display information.
- Storage is persistent.
- Communication is asynchronous.
- No subsystem blocks another.
- Measurement integrity always takes priority over user interface responsiveness.

By isolating communications from acquisition and calculations, the firmware achieves excellent scalability while maintaining deterministic measurement performance.

7.19 Summary

The integrated communication and data management subsystem transforms the Karl's Ultimate DIY Dyno from a standalone measurement device into a complete self-contained dynamometer platform.

Through the combination of wireless networking, asynchronous communication, persistent storage and browser-based operation, Firmware Version 3.69 provides a modern user experience without compromising the deterministic behaviour required for precision engine testing.

Appendix A

Sensor Installation Guide

A.1 Overview

This appendix describes the physical installation and electrical connection of all sensors supported by Firmware Version 3.69.

Correct installation is essential for obtaining repeatable and accurate measurements. Incorrect sensor placement, poor grounding or improper cable routing may significantly degrade measurement quality.

The following sensors are supported:

- Engine RPM Hall Sensor
- Drum RPM Hall Sensor
- Brake Load Cell
- Exhaust Gas Temperature (EGT)
- Atmospheric Pressure & Temperature (BMP280)
- External Run Trigger

A.2 Engine RPM Hall Sensor

Purpose

The Engine RPM Hall Sensor measures crankshaft rotational speed by detecting a ferromagnetic trigger or permanent magnet mounted to the engine.

The sensor provides the primary timing reference used for:

- Engine RPM
- Horsepower

- Engine Torque
- Gear Ratio

Recommended Installation

The Hall sensor should be mounted securely to avoid vibration.

Recommended air gap:

0.5–2.0 mm

The trigger should pass centrally across the sensor face.

Avoid mounting near:

- Ignition coils
- Spark plug leads
- Starter motor wiring

These components may introduce electrical interference.

PCB Connection

PCB Connector P1

Pin 1	Signal
Pin 2	Ground

Recommended cable:

- Twisted pair
- Shielded cable preferred
- Maximum recommended length: 2 metres

A.3 Drum RPM Hall Sensor

Purpose

Measures roller rotational speed.

The firmware supports any number of magnets by configuring:

Pulses Per Revolution

inside the Settings page.

Recommended Installation

Mount magnets evenly around the drum.

Example:

Magnets	Pulses per Revolution
1	1
2	2
4	4
8	8
10	10

Maintain equal spacing.

Sensor gap:

0.5–2 mm.

PCB Connection

PCB Connector P2	
Pin 1	Signal
Pin 2	Ground

A.4 Brake Load Cell

Purpose

Measures brake reaction force.

The firmware converts force into brake torque using the configured lever arm length.

Mechanical Installation

The load cell should only experience:

Pure Tension

or

Pure Compression

Avoid:

- Side loading
- Twisting
- Bending

These forces reduce accuracy and may permanently damage the sensor.

Wiring

Connect directly to the NAU7802 interface.

PCB Terminal	Load Cell
E+	Excitation +
E-	Excitation -
A+	Signal +
A-	Signal -

Typical wiring colours (may vary by manufacturer):

Colour	Function
Red	Excitation +
Black	Excitation -
Green	Signal +
White	Signal -

Always verify using the load cell manufacturer's documentation.

A.5 Exhaust Gas Temperature

Purpose

Measures exhaust gas temperature for tuning and engine protection.

The firmware records:

- Live EGT
- Peak EGT
- Run EGT

Sensor Type

K-Type Thermocouple

Probe Position

Recommended installation:

100–150 mm from the exhaust port.

Install in the upper half of the header pipe.

Avoid installing:

- Immediately after the silencer
- At sharp bends

Wiring

Connect directly to the MCP9600 interface.

PCB Connector	Function
+	Thermocouple +
-	Thermocouple -

Do not extend thermocouple wiring using ordinary copper wire.

Use proper thermocouple extension cable.

A.6 BMP280

The BMP280 is mounted directly on the PCB.

No external wiring is required.

Avoid enclosing the PCB inside airtight boxes.

Ambient air must be able to reach the sensor.

A.7 External Trigger

A momentary push button may be connected.

Recommended wiring:

Normally Open Push Button.

Maximum cable length:

Approximately 3 metres.

Longer cables should be shielded.

A.8 Cable Routing

Signal wiring should be routed separately from:

- Ignition leads
- Starter cables
- Alternator wiring

Where possible:

- Use twisted pair wiring
- Ground shields at one end only
- Avoid ground loops

A.9 Final Inspection

Before first startup verify:

☐ Hall sensor gaps

☐ Load cell securely mounted

☐ Thermocouple polarity

☐ Cable strain relief

☐ No loose wiring

☐ All connectors fully inserted

Appendix B

PCB Connector Reference

B.1 PCB Overview

The Karl's DIY Dyno PCB has been designed to minimise installation complexity by providing dedicated connectors for each supported sensor.

Every connector is keyed by function.

No connector shares signal assignments with another.

B.2 Connector Summary

Connector	Function
P1	Engine RPM Hall
P2	Drum RPM Hall
P3	External Trigger
U6	Load Cell
U4	EGT
ESP32	Main Controller

B.3 P1 Engine RPM

Pin	Function
1	Hall Signal
2	Ground

Purpose:

Measures engine rotational speed.

B.4 P2 Drum RPM

Pin	Function
1	Hall Signal
2	Ground

Purpose:

Measures drum rotational speed.

B.5 P3 External Trigger

Pin	Function
1	Trigger
2	Ground

Purpose:

External Start/Stop Recording.

B.6 Load Cell Connector

Terminal	Description
E+	Excitation Positive
E-	Excitation Negative
A+	Signal Positive
A-	Signal Negative

B.7 Thermocouple Connector

Terminal	Description
+	Thermocouple Positive
-	Thermocouple Negative

B.8 Recommended Cable Types

Signal	Cable Recommendation
Hall Sensors	Shielded Twisted Pair
Load Cell	Shielded Instrument Cable
Thermocouple	K-Type Extension Cable
Trigger	Twisted Pair
USB	Standard USB-C

B.9 Grounding Recommendations

For optimum measurement stability:

- Connect all sensor grounds only through the PCB.
- Avoid multiple chassis grounding points.
- Route Hall sensor cables separately from ignition wiring.
- Do not share load cell wiring with high-current conductors.
- Use ferrite suppressors where high electromagnetic interference is present.

Proper grounding significantly improves measurement repeatability and reduces susceptibility to electrical noise.

B.10 Installation Checklist

Before powering the dyno:

☐ Engine Hall sensor connected to **P1**

☐ Drum Hall sensor connected to **P2**

☐ Trigger connected to **P3** (if used)

☐ Load cell connected to **U6**

☐ Thermocouple connected to **U4**

☐ ESP32 firmly seated in sockets

☐ All connectors checked for correct polarity

☐ Sensor cables secured against vibration

☐ No exposed conductors

☐ Brake arm moves freely

☐ Load cell unloaded before calibration