MyFOC: An Auditable Field-Oriented Control Library for Low-Cost Rehabilitation Robotics

by

Hoang Tuan Kiet Vu

Bachelor of Engineering in Electrical Engineering, University of Victoria, 2025

A Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

© Hoang Tuan Kiet Vu, 2026
University of Victoria

All rights reserved. This report may not be reproduced in whole or in part, by photocopying or other means, without the permission of the author.

We acknowledge and respect the Lək̓ʷəŋən (Songhees and Xʷsepsəm/Esquimalt) Peoples on whose territory the university stands, and the Lək̓ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

MyFOC: An Auditable Field-Oriented Control Library for Low-Cost Rehabilitation Robotics

by

Hoang Tuan Kiet Vu

Bachelor of Engineering in Electrical Engineering, University of Victoria, 2025

Supervisory Committee

Dr. Ilamparithi Thirumarai Chelvan, Supervisor
Department of Electrical and Computer Engineering

Dr. Ling Xing, Departmental Member
Department of Electrical and Computer Engineering

Abstract

Robotic devices that work in direct physical contact with people – rehabilitation trainers in particular – need smooth, back-drivable torque control at and near zero speed. Field-oriented control (FOC) is the standard way to obtain it from a brushless motor, but the firmware that implements it is normally either a closed vendor binary or a large general-purpose library, neither of which can be read end to end and defended line by line in a safety argument. This project presents MyFOC, a compact FOC library written from first principles for the Arduino UNO and the Texas Instruments DRV8302 gate driver, together with the upper-limb rehabilitation trainer built on top of it.

The library implements the complete control chain – quadrature encoder decoding, synchronized 31.4 kHz PWM, the Clarke and Park transformations, space-vector modulation, sensor alignment, cascaded current and velocity loops, automatic identification of the motor's electrical parameters, and an impedance controller – in roughly 21 KB of program memory, leaving ample headroom for application code on the 32 KB ATmega328P. It is deployed on an eleven-pole-pair gimbal motor with an incremental encoder and verified stage by stage on the bench, with the widely used SimpleFOC library as a documented reference point.

The application is a single-axis upper-limb rehabilitation trainer. Its impedance controller renders a programmable virtual spring–damper: the clinician sets the stiffness, the damping and the rest position; a software torque ceiling, backed by hardware over-current protection, bounds the commanded actuation; and an adaptive layer firms up the resistance when the patient moves vigorously and relaxes it when they move gently. By changing only these parameters at run time, without re-flashing the firmware, the same controller reproduces five clinically recognised exercise modes: isometric, isotonic, isokinetic, adaptive-resistance and transparent back-drive. All results were obtained in the library's voltage-mode commutation path, with the current loop implemented but not closed. A deadweight calibration shows that the rendered torque follows the programmed law faithfully in shape and repeatability but carries an absolute-scale error of about 1.6×, which closing the current loop would remove. Taken together, the results show that a complete and auditable torque-control stack – parameter identification, space-vector commutation, impedance rendering and layered safety limiting – fits within an 8-bit microcontroller and on apparatus costing about USD 135 per axis, roughly twenty times less than the cheapest clinical-grade configuration costed in Chapter 6. MyFOC is therefore offered as a transparent and reproducible reference platform for low-cost rehabilitation robotics, on which closing the current loop is the immediate next step towards absolute torque accuracy.

Table of Contents

Supervisory Committee ii
Abstract iii
List of Tables vii
List of Figures viii
Symbols x
Acronyms xi
Chapter 1 Introduction 1
1.1. Background and Motivation 1
1.2. Problem Statement 1
1.3. Related Work 3
1.3.1. Field-Oriented Control and Its Implementations 3
1.3.2. Rehabilitation Robotics and Interaction Control 4
1.3.3. Actuation in Deployed Rehabilitation Devices 4
1.3.4. Interaction Forces and Torques in Upper-Limb Therapy 5
1.3.5. Low-Cost and Open Hardware Platforms 7
1.4. Objectives and Contributions 7
1.5. Scope and Limitations 8
1.6. Organization of the Report 8
Chapter 2 Theoretical Background 10
2.1. The Three-Phase PMSM 10
2.2. Reference-Frame Transformations 11
2.2.1. Clarke Transformation ($\boldsymbol{abc \rightarrow \alpha\beta}$) 11
2.2.2. Park Transformation ($\boldsymbol{\alpha\beta \rightarrow dq}$) 12
2.2.3. PMSM Model in the dq Frame 12
2.3. Space-Vector Pulse-Width Modulation 13
2.3.1. Sinusoidal PWM 13
2.3.2. Space-Vector PWM with Min-Max Injection 13
2.4. Quadrature Encoder Decoding 13
2.5. Cascaded Control of a PMSM 14
2.5.1. Architecture 14
2.5.2. Cross-Axis Decoupling Feed-Forward 14
2.5.3. Internal-Model-Control (IMC) Tuning of the Current Loops 15
2.5.4. Anti-Windup and Saturation 15
2.6. Impedance Control 15

2.7. SimpleFOC as a Reference Implementation 16
2.8. BLDC and PMSM: The Distinction from a Firmware Perspective 16
2.9. Impedance Control Compared with Admittance Control 18
Chapter 3 System Design and Architecture 20
3.1. Hardware Overview 20
3.2. Software Architecture 23
3.3. Control-System Architecture 24
3.4. Safety Architecture 25
Chapter 4 Implementation 27
4.1. Mathematical Kernel (MyFOCMath) 27
4.1.1. Sin/Cos Lookup Table 27
4.1.2. Clarke, Park, Inverse Park, Inverse Clarke 27
4.1.3. Space-Vector PWM 27
4.2. PWM Driver (MyBLDCDriver3PWM) 28
4.3. Quadrature Encoder Decoder (MyEncoder) 28
4.4. Current Sensing (MyCurrentSense3PWM) 29
4.4.1. Zero calibration 29
4.4.2. Reconstruction of i_c 29
4.4.3. ADC clock acceleration 29
4.5. PID and Low-Pass Filter (MyPID, MyLPF) 30
4.6. Motor Object and FOC Pipeline (MyBLDCMotor) 30
4.7. Sensor Alignment (*alignSensor*) 31
4.8. Motor Parameter Identification 31
4.8.1. Resistance 31
4.8.2. Inductance 32
4.8.3. Flux Linkage 32
4.9. Impedance Controller 32
4.10. Impedance-Control Signal Path Through Firmware and Hardware 33
4.11. Data Logging and Monitoring 34
Chapter 5 Experimental Setup and Results 35
5.1. Bench Configuration 35
5.2. Test Procedures 38
5.2.1. Parameter Identification Procedure 38
5.2.2. FOC Validation Procedure 39
5.2.3. Clinical Exercise-Mode Procedure 39

5.2.4. Relation to Clinical Testing Practice 40
5.3. Data Logging and Plotting Pipeline 42
5.4. Compile-Time Verification 42
5.5. Hardware Bring-Up and Auto-Detected Parameters 43
5.6. Identified Motor Parameters 44
5.6.1. Improving the Inductance Measurement 45
5.7. Functional Verification of the FOC Pipeline 46
5.8. Application: Robotic Rehabilitation Devices 47
5.8.1. Clinical Exercise Modes: Five-Mode Validation 47
5.8.2. Back-Drivability in the As-Built System and Clinical Safeguards 53
Chapter 6 Conclusion and Future Work 56
6.1. Limitations 56
6.2. Future Work 57
6.3. Path to Clinical Deployment 58
6.4. Regulatory Pathway and Certification 60
6.5. Conclusion 62
References 64
Appendix A. Source-File Organisation 68
Appendix B. Mathematical Derivations 69
B.1 Clarke Transformation (Amplitude-Invariant Form) 69
B.2 Park Transformation 69
B.3 PMSM Torque Equation 70
B.4 IMC Tuning of the First-Order Current-Loop Plant 70
B.5 Impedance Dynamics for a Single-Degree-of-Freedom Load 70
Appendix C. Engineering Drawings 72

List of Tables

Table 1.1: Actuation in representative rehabilitation devices, compared with the bench of this report. 5
Table 1.2: Normative isometric torque of the healthy adult elbow, dominant arm, weighted across 19 studies and 1880 subjects [34]. Standard deviations in parentheses. 6
Table 2.1: BLDC and PMSM compared from the point of view of the control firmware. 17
Table 2.2: Impedance and admittance control compared for a rehabilitation application. 18
Table 3.1: Hardware configuration. 22
Table 3.2: Pin assignments. 22
Table 3.3: Module decomposition of the MyFOC library. 23
Table 4.1: Quadrature decoding logic. 28
Table 5.1: Diagnostic outputs from the as-built bench. 43
Table 5.2: Empirical motor parameters from the identification procedure. 44
Table 5.3: Options for improving the inductance identification, with the constraint each removes. 46
Table 5.4: Mechanisms for resisting back-driving, and their behavior under fault. 54
Table 6.1: Indicative per-axis parts cost (USD) for the bench presented here and two clinical-grade configurations. Excludes certification, software licences, and labour. 59
Table 6.2: Standards applicable to a powered upper-limb rehabilitation device. 61

List of Figures

Figure 2.1: Complete field-oriented control signal flow as implemented in MyFOC. The dashed path is the voltage-mode fallback used for the experiments reported here. 11
Figure 4.1: Impedance-control signal path. The encoder supplies shaft angle and velocity to the impedance law and the electrical angle to the transforms; the commanded torque is clamped to the ceiling, converted to a current reference through the torque constant, and passed to the field-oriented core. The dashed marker shows where the deployed build departs from this path and writes a q-axis voltage directly. 33
Figure 5.1: Electrical diagram of the setup. 35
Figure 5.2: The 3D-printed motor mount, as built. 36
Figure 5.3: The 3D-printed pendulum arm, as built. 36
Figure 5.4: The assembled experimental apparatus. 37
Figure 5.5: Close-up of the gimbal motor with the coupled AMT103 encoder. 37
Figure 5.6: Bench power supply and switched-mode supply used for the 12 V DC bus. 38
Figure 5.7: Isotonic loading. (a) Clinical practice [57]: a cable and pulley keep the applied force constant and its moment arm approximately constant, so the resisting torque does not vary with joint angle. (b) The arrangement used here: a mass hung directly from the lever produces a resisting torque that varies as sinφ, so the lever settles at the angle where the commanded torque balances it. 40
Figure 5.8: Isokinetic testing. (a) Clinical practice: the servo imposes the angular velocity, and a calibrated in-line torque transducer records the torque the patient develops against it. (b) The arrangement used here: the velocity set-point is held by the velocity loop, but with no transducer and no resisting load the run verifies velocity regulation rather than developed torque. 42
Figure 5.9: Setup for the isometric test (Test 1). The controller holds the lever level with a stiff virtual spring while calibrated weights are hung; the holding voltage and the known weight together give a torque measurement that is independent of the commanded control law. 48
Figure 5.10: Isometric test (Test 1). (a) Holding q-axis voltage steps up as weight is added. (b) Deadweight calibration: the true gravitational torque (50–150 g) follows a 0.032 N·m/V line, about 1.6× below the voltage-mode reconstruction K_t/R = 0.0511 N·m/V; the 200 g point is saturated (about 40° droop) and excluded. 49
Figure 5.11: Setup for the isotonic test (Test 2). With the position spring disabled a constant torque command is applied and the lever rotates until the gravitational torque of the fixed 200 g load balances it; the settle angle is recorded against the commanded torque. 49
Figure 5.12: Isotonic test (Test 2). (a) Reconstructed torque commanded in flat steps. (b) The constant torque command lifts a fixed 200 g load to a progressively larger settle angle (about 28° to 62°) as the command is increased from 0.10 to 0.30 N·m. 50
Figure 5.13: Setup for the isokinetic test (Test 3). A velocity set-point of ±2 rad/s drives the arm at constant speed through continuous rotation. 50
Figure 5.14: Isokinetic test (Test 3). Measured shaft velocity (raw and 0.4 s moving average) against the ±2 rad/s set-point; the mean tracks the command (+2.20 / −2.02 rad/s) while the instantaneous estimate is quantization-noisy. 51
Figure 5.15: Setup for the adaptive test (Test 4). The operator pulls the arm slowly and then quickly; the effective stiffness and ceiling scale with the measured effort (low-pass-filtered speed while pushing outward). 52

Figure 5.16: Adaptive test (Test 4) at the deployed baseline (K_base = 0.5 N·m/rad, L_base = 0.05 N·m, A = 2.0). (a) Effort and K_eff against time: with adaptation off (A = 0) effort spikes but K_eff stays at baseline; with A = 2.0 it scales with effort. (b) K_eff versus effort - the adaptive points match the law K_base·(1 + A·effort/v_ref) to an RMS of 3.3×10^{-4} N·m/rad (n = 1362). 52
Figure 5.17: Setup for the back-drive test (Test 5). The motor is de-energised (v_q = 0); the free arm is released to ring down and small weights are added to find the breakaway torque. 53
Figure 5.18: Back-drive / transparency test (Test 5). (a) The de-energised arm swings freely with light damping ($\zeta \approx 0.02$–0.05). (b) Static breakaway torque for the 15 / 20 / 32 g weights, 0.018–0.038 N·m. 53
Figure C.1: Motor mount, dimensioned drawing. The Ø47 register seats the GM4108-120T stator, the four M3 clearance holes on a Ø38.2 bolt circle retain it, and the Ø12 post clamps to the bench. 72
Figure C.2: Pendulum arm, dimensioned drawing. The arm bolts to the motor rotor through four M3 holes on a Ø30 bolt circle; the Ø4.9 hole at a radius of 120 mm carries the calibration weights used in Section 5.6. 73
Figure C.3: Actuator-module assembly, side and front elevations. The stack from the mounting plate to the outer face of the arm measures 43 mm, and the 120 mm lever radius defines the torque arm used throughout Chapter 5. 73

Symbols

Symbol	Meaning
θ_m, ω_m	Mechanical angle (rad), mechanical angular velocity (rad/s)
θ_e, ω_e	Electrical angle (rad), electrical angular velocity (rad/s); $\omega_e = pp \cdot \omega_m$
pp	Number of pole pairs
v_a, v_b, v_c	Three-phase stator voltages (V)
i_a, i_b, i_c	Three-phase stator currents (A)
v_α, v_β	Stator voltages in the stationary (Clarke) reference frame
i_α, i_β	Stator currents in the stationary reference frame
v_d, v_q	Direct- and quadrature-axis voltages (rotor frame, V)
i_d, i_q	Direct- and quadrature-axis currents (rotor frame, A)
R, R_{phase}	Stator phase resistance (Ω)
L_d, L_q	Direct- and quadrature-axis inductance (H)
λ_{PM}	Permanent-magnet flux linkage (Wb = V·s/rad)
$K_t = \left(\frac{3}{2}\right) \cdot pp \cdot \lambda_{PM}$	Torque constant (N·m/A)
τ	Electromagnetic torque (N·m)
V_{DC}, V_{bus}	DC bus voltage of the inverter (V)
K, B, θ_{eq}	Virtual stiffness, damping and equilibrium of the impedance controller

Acronyms

Acronym	Expansion
ADC	Analogue-to-Digital Converter
AVR	Advanced Virtual RISC (Atmel/Microchip 8-bit MCU family)
BLDC	Brushless Direct Current (motor)
CPR	Counts Per Revolution (encoder, post-quadrature)
DRV	Driver IC (Texas Instruments DRV8302 used here)
EMF	Electro-Motive Force (back-EMF)
FOC	Field-Oriented Control
IMC	Internal Model Control (PI tuning method)
ISR	Interrupt Service Routine
LPF	Low-Pass Filter
MCU	Microcontroller Unit
MMF	Magneto-Motive Force
PI / PID	Proportional-Integral / Proportional-Integral-Derivative controller
PMSM	Permanent-Magnet Synchronous Machine
PPR	Pulses Per Revolution (encoder, pre-quadrature)
PWM	Pulse-Width Modulation
SPWM	Sinusoidal Pulse-Width Modulation
SVPWM	Space-Vector Pulse-Width Modulation

Chapter 1
Introduction

1.1. Background and Motivation

Robotic systems intended for direct physical interaction with human beings - collaborative manipulators, exoskeletons, surgical robots, and rehabilitation devices - share a common requirement: they must produce smooth, accurately controlled torque at low speeds, often with large dynamic range and sometimes with the actuator effectively back driven by the operator. Three-phase permanent-magnet synchronous machines, when driven by FOC, are well suited for these applications because the rotor-frame transformation eliminates the commutation-induced torque ripple that plagues trapezoidal-driven brushless DC motors and brings the inner-current dynamics into a form (a stable first-order DC plant) for which classical linear control is sufficient.

Field-Oriented Control has been studied since the late 1960s and matured into a textbook subject by the 1990s [1], [2], but its software realization on inexpensive hardware became commonplace only with the rise of capable 32-bit ARM Cortex-M microcontrollers in the 2010s. Open-source motor-control firmware accelerated this trend significantly: in particular, the SimpleFOC project [3] provides a polished, well-documented Arduino-compatible library that runs on hardware ranging from the AVR-based UNO to STM32 and ESP32 platforms, and has become a common reference in undergraduate motor-control courses and small-scale research.

Several application domains have emerged that share a need for smooth, programmable, low-friction torque control. Robotic rehabilitation devices for stroke, spinal cord injury and orthopedic recovery have become an active clinical and research area [4], [5], [6]; the dominant low-level control paradigm here is impedance control, introduced by Hogan in 1985 [7], in which the actuator is commanded to render a programmable virtual mechanical impedance (typically a spring–damper field) rather than to track a position or velocity reference. Impedance control preserves passive compliance (the patient can always overpower the device), is robust to model uncertainty, and allows clinically motivated rehabilitation modes - isometric strength training, isotonic resistance, isokinetic constant-velocity motion - to be expressed cleanly as different choices of stiffness K and damping B [8].

The work reported in this project sits at the intersection of these threads: a from-scratch FOC firmware that is pedagogically transparent and benchmarkable, a low-cost hardware platform (Arduino UNO + DRV8302) accessible to undergraduate laboratories, and a rehabilitation application that exercises the impedance-control layer of the stack.

1.2. Problem Statement

Neurological injury is a leading cause of long-term motor disability in adults, and the recovery of upper-limb function depends strongly on the quantity of task-specific, actively initiated practice a patient performs [9]. Delivering that dose by hand is labor-intensive, which is why robot-assisted therapy has been pursued for three decades: a machine can supply hundreds of graded repetitions per session with a consistency no therapist can sustain

[6]. The largest multi-center trial of upper-limb robot therapy to date confirmed that intensive robot-assisted training improves motor outcomes relative to usual care, although its margin over an equally intensive conventional program was small [4]. Critically, the property that makes such machines therapeutically useful is not positioning accuracy but regulated mechanical impedance: the device must assist, resist, or yield at the human-machine interface according to therapeutic intent [7], [8].

The barrier to wider deployment is economic rather than scientific. Surveys of the field consistently observe that the devices which have demonstrated clinical benefit are complex, heavily instrumented and expensive, and that their cost confines them to well-resourced research hospitals [5]. Commercial end-effector and exoskeleton systems are priced in tens to hundreds of thousands of dollars, and even a single research-grade torque-controlled axis assembled from catalogue components carries a four-figure cost, as quantified in Chapter 6. The population that would benefit most from high-dose robot-mediated practice, namely patients in community clinics, rural hospitals and home settings, therefore has the least access to it.

Field-oriented control (FOC) is the enabling technology for the smooth, zero-speed, bidirectional torque production that impedance rendering demands, and it is, in industrial terms, a solved problem: every major semiconductor vendor supplies a tuned FOC stack [10], [11], and mature open-source libraries serve the hobby and research communities [3]. What is not readily available is a field-oriented controller that a graduate engineer can read in full, derive from first principles, modify at the level of an individual transformation, and defend line by line in a safety argument. Vendor stacks are optimized binaries documented at the block-diagram level; general-purpose open-source libraries are large, heavily templated, and abstract the control path behind convenience interfaces. In an application where an incorrect current loop gain or an unhandled angle wrap is transmitted directly into a patient's arm, an unreadable controller is a liability, and the ability to inspect every element of the loop has value independent of the numerical performance achieved.

This report therefore addresses the following problem. Can a field-oriented control library, written from scratch and small enough to run on an 8-bit ATmega328P, deliver torque control of sufficient quality to render the impedance behaviors used in upper-limb rehabilitation? And can that claim be substantiated experimentally, including where it fails, on apparatus costing on the order of one hundred dollars? The problem decomposes itself into four sub-problems:

- **1. Implementation.** Derive the complete FOC signal chain, comprising the reference-frame transformations, space-vector modulation, encoder-based electrical-angle estimation and the cascaded current, velocity and position loops, and implement it within the 32 KB flash and 16 MHz budget of an ATmega328P without recourse to vendor middleware (Chapters 2 to 4).
- **2. Identification.** Determine the electrical and electromechanical parameters of an inexpensive gimbal motor using bench instrumentation only and establish how far the resulting torque predictions may be trusted (Chapters 4 and 5).
- **3. Clinical expressiveness.** Establish that a single impedance-control kernel, suitably parameterized, reproduces the qualitatively distinct exercise modes used in

rehabilitation practice, rather than requiring a separate controller for each (Chapters 4 and 5).

- **4. Validation.** Validate the library on a physical single-axis testbed and report the discrepancies between predicted and measured behavior as carefully as the agreements (Chapters 5 and 6).

The objectives stated in Section 1.4 address these sub-problems in order, and the design decisions of Chapter 3 are traced back to them explicitly.

1.3. Related Work

The work reported here draws on two literatures that rarely appear together: the machine-drives literature, from which field-oriented control originates, and the rehabilitation-robotics literature, which supplies the interaction-control requirements. This section surveys both, together with the smaller body of work on deliberately low-cost torque-controlled hardware and identifies the gap this project occupies.

1.3.1. Field-Oriented Control and Its Implementations

Field-oriented control rests on the reference-frame transformations introduced by Clarke [12] and Park [13], and on the insight, formulated independently by Hasse [14] and Blaschke [15], that orienting the control axes to the rotor flux makes an AC machine behave, in the rotating frame, like a separately excited DC machine, with one current component commanding flux and the other commanding torque. Modern treatments of the resulting cascade are given by Krishnan [1], Leonhard [2], Sul [16] and Vas [17]. Holmes and Lipo [18] provide the definitive analysis of the modulation stage, and Holtz [19] the standard account of the pulse-width-modulation and angle-estimation issues that dominate practical implementations. These sources form the analytical basis of Chapter 2 and Appendix B.

Implementations are divided into three groups. Vendor stacks, such as Texas Instruments' InstaSPIN-FOC [10] and STMicroelectronics' X-CUBE-MCSDK [11], deliver production-grade performance with automatic parameter identification and sensorless observers, but ship as pre-compiled libraries or generated code whose internals are exposed only as block diagrams and tuning parameters. Open-source motion firmware, including VESC [20], ODrive [21] and moteus [22], is fully readable and has driven the recent proliferation of quasi-direct-drive robots, but targets 32-bit floating-point microcontrollers and couples the control code tightly to specific hardware. Between these lies SimpleFOC [3], a portable Arduino-compatible library and the reference against which the present implementation was checked; it is deliberately general, and that generality, in the form of templated sensor and driver abstractions and extensive run-time configuration, is precisely what makes its control path easier to use than to read. Katz et al. [23] show what becomes possible when the actuator, the drive electronics and the control law are designed together, and the control code is left open for inspection. Their Mini Cheetah estimates joint torque from motor current alone, with no torque sensor, using commodity motors and drives. The present work adopts the same principle on a smaller scale: an inexpensive motor, with control quality obtained from readable software rather than from costly hardware.

1.3.2. Rehabilitation Robotics and Interaction Control

The theoretical foundation of physical human-robot interaction is Hogan's impedance-control formulation [7], which replaces trajectory tracking with the imposition of a desired dynamic relationship between motion and force at the interface. Hogan and Buerger [8] later analyzed the passivity and stability conditions under which such a controller may safely be coupled to an unknown human limb. MIT-MANUS [6] carried that formulation into clinical trials and established the therapeutic vocabulary of assistive, resistive and passive exercise still in use today; the same platform underlies the multi-center trial of Lo et al. [4]. ARMin [24] extended the approach to a multi-degree-of-freedom exoskeleton with patient-cooperative control, and the Lokomat [25] applied it to gait training.

Two actuation strategies dominate the resulting safety literature. Series elastic actuation [26] interposes a compliant element between motor and load and measures its deflection to obtain force, trading bandwidth for intrinsic safety and force fidelity. Quasi-direct drive [23] removes the gearbox instead, relying on low-reduction, high-torque motors and a fast current loop to obtain both transparency and torque control. The present work follows the second strategy, for the reasons set out in Chapter 3.

Reviews by Maciejasz et al. [5] and Proietti et al. [27] catalogue the resulting device space, while Marchal-Crespo and Reinkensmeyer [28] survey the control strategies themselves. Their central finding shapes the design of Chapter 3: assistance must be graded to the patient's residual ability rather than imposed, because a device that performs the movement on the patient's behalf suppresses the voluntary effort on which recovery depends. Wolbrecht et al. [29] realized this principle with an adaptive controller that withdraws assistance as performance improves and is the direct antecedent of the adaptive-assistance mode evaluated in Chapter 5. The isokinetic mode implemented here derives instead from clinical dynamometry practice, in which the limb is moved at a constant angular velocity while the resisting torque is recorded [30].

1.3.3. Actuation in Deployed Rehabilitation Devices

Because the actuator specification is the single largest determinant of what a rehabilitation device can and cannot do, it is worth setting out what the deployed systems actually use. Three patterns dominate, and all three differ from the bench described in this report in the same direction: more torque, more reduction, and a dedicated force or torque measurement.

The first pattern is a brushed or brushless permanent-magnet motor behind a high-ratio, low-backlash gearbox. ARMin uses direct-current motors coupled to Harmonic Drive units at reductions of 1:100 on the shoulder elevation and elbow axes and 1:30 on the third axis; the manufacturer reports a no-load back-driving torque of 2.8 N·m and limits the motor current to 3 A on one axis and 10 A on the others [24], [31]. The clinical trial built on that platform is reported by Klamroth-Marganska et al. [32]. The Lokomat applies the same approach to hip and knee actuation for gait training [25]. Harmonic reduction buys torque density and near-zero backlash, and it is the reason these devices reach the 5–15 N·m range at the joint; the price is reflected inertia and friction that scale with the square of the ratio, which is why every one of these systems carries a joint-level force or torque sensor rather than inferring torque from motor current.

The second pattern is series elastic actuation [26], in which a compliant element is placed between the gearbox output and the limb, and its deflection is measured to obtain force. This recovers force fidelity and intrinsic safety from a geared drivetrain at the cost of closed-loop bandwidth, and it is common where a torque transducer is unaffordable or where the actuator must remain safe under a controller fault.

The third pattern, and the one closest to this project, is quasi-direct drive: a high-pole-count permanent-magnet machine behind a single-stage reduction of roughly 6:1 to 10:1, with torque regulated by a fast current loop rather than measured [23]. A representative commercial module, the T-Motor/CubeMars AK80-9, integrates a brushless machine with 21 pole pairs, a 9:1 planetary stage, a magnetic encoder and a field-oriented drive; it is rated at 9 N·m continuous and 18 N·m peak from a 48 V bus, with a torque constant of 0.105 N·m/A referred to the motor and a measured back-driving torque of 0.51 N·m [33]. Table 1.1 places these configurations alongside the bench built here.

Table 1.1: Actuation in representative rehabilitation devices, compared with the bench of this report.

Configuration	Machine type	Transmission	Joint torque	Torque sensing
ARMin III [24], [31]	DC, permanent magnet	Harmonic Drive 1:100 / 1:30	≈ 30–50 N·m at the shoulder	Joint force/torque sensor
Lokomat [25]	DC, permanent magnet	Geared hip and knee drives	Gait-level hip/knee torque	Joint torque sensing
Series elastic actuator [26]	Brushless PMSM	Gearbox + compliant element	Set by gearbox ratio	Spring deflection
Quasi-direct drive [23], [33]	Brushless PMSM, high pole count	Single stage, 6:1 to 10:1	9 N·m cont., 18 N·m peak	Inferred from current
This bench	Surface PMSM, 11 pole pairs	Direct drive, no reduction	≈ 0.42 N·m	None — inferred from voltage

Two conclusions follow for any attempt to scale this work. First, the power involved is modest: an elbow or shoulder axis moving at up to roughly 2 rad/s against 12 N·m demands about 24 W of mechanical output, so the binding constraints are continuous torque and thermal capacity at low speed, not power. Second, the quasi-direct-drive column is the only one that preserves the current-only torque estimation used here; both geared alternatives assume a force or torque transducer, and adopting either would make the absolute-scale error reported in Section 5.8 irrelevant because torque would no longer be inferred. Section 6.3 costs both routes.

1.3.4. Interaction Forces and Torques in Upper-Limb Therapy

The actuator specification of Section 1.3.3 was stated in terms of joint torque, and that figure must be derived from the human rather than assumed. This section establishes what torques a human upper limb produces, what fraction of that a therapy device must supply, and how both compare with the bench built here. The distinction matters because the two quantities are very different: a device sized to match a healthy maximum voluntary contraction would

be both unnecessary and unsafe, whereas a device unable to support the weight of the limb cannot deliver therapy at all.

Normative isometric strength of the healthy adult elbow is well characterized. A systematic review by Kotte et al. [34] pooled 19 studies covering 1880 asymptomatic volunteers (1179 men, 701 women, mean age 37.8 years) and reported the weighted values listed in Table 1.2; forces were converted to torques using a mean forearm moment arm of 26.4 cm established by Askew et al. [35]. Dominant elbow flexion averages 76.7 N·m in men and 52.5 N·m in women, with extension roughly two-thirds of flexion and the non-dominant arm weaker by a factor of 0.94. The authors caution that the standard deviations are large and that no consensus exists on measurement posture or device, so these figures should be read as an order of magnitude rather than a specification.

Table 1.2: Normative isometric torque of the healthy adult elbow, dominant arm, weighted across 19 studies and 1880 subjects [34]. Standard deviations in parentheses.

Movement	Male (N·m)	Female (N·m)	Dominant : non-dominant
Flexion	76.7 (15.0)	52.5 (18.3)	1 : 0.94
Extension	48.2 (5.6)	40.2 (13.3)	1 : 0.97
Pronation	10.0 (7.8)	6.4 (7.1)	1 : 0.93
Supination	10.7 (7.1)	7.1 (6.7)	1 : 0.94

Maximum voluntary contraction is not, however, the design target. A rehabilitation device is required to support the limb against gravity and to add a graded increment of assistance or resistance about that operating point; it is never required to overpower the patient, and the passivity argument of Section 2.9 depends on it being unable to. The gravitational component follows directly from standard anthropometry [36]. For a 75 kg adult the forearm and hand together are about 2.2 % of body mass, or 1.65 kg, with the center of mass at roughly 68 % of the forearm length from the elbow; with a forearm length of 0.26 m this gives a holding torque of $1.65 \times 9.81 \times 0.177 \approx 2.9$ N·m at the elbow with the forearm horizontal. Taking the whole upper limb at about 5 % of body mass, or 3.75 kg, with its center of mass at roughly 0.33 m from the shoulder, the corresponding shoulder torque is $3.75 \times 9.81 \times 0.33 \approx 12$ N·m.

Those two figures are the origin of the 5–15 N·m range quoted in Section 1.3.3 and in Section 6.3. The lower bound corresponds to supporting a forearm with a working margin for graded resistance; the upper bound corresponds to supporting a whole arm at the shoulder, which is the worst case an upper-limb trainer must handle. Expressed as a fraction of healthy capability, a 12 N·m shoulder actuator supplies roughly 16 % of the 76.7 N·m a healthy male elbow flexor can generate, so a patient with even partially preserved strength retains the authority to overpower the device. That ratio, rather than the absolute torque, is the quantitative form of the safety argument made qualitatively in Sections 2.9 and 3.4.

For an impaired limb the relevant quantity is the deficit rather than the maximum. Hemiparesis reduces voluntary torque substantially on the affected side, and the assist-as-needed paradigm of Wolbrecht et al. [29] requires the device to supply only the shortfall between what the patient can produce and what the task demands, withdrawing that assistance as performance improves. A device sized to the gravitational load therefore covers the clinically useful range without approaching the patient's own capability, which is why the

actuators surveyed in Table 1.1 cluster in the 5–30 N·m band rather than at the 50–80 N·m of a healthy elbow.

Set against these figures, the bench described in this report is explicitly a proof-of-concept apparatus rather than a scaled-down therapy device. Its motor-limited capability of about 0.42 N·m is roughly 15 % of the 2.9 N·m required to hold a forearm horizontally, and the 0.08 N·m clinical ceiling adopted in Section 3.4 is about 3 % of that figure and around 0.1 % of a healthy elbow maximum. The bench therefore cannot support a human limb, and the exercise modes validated in Chapter 5 are demonstrated against a 200 g calibration mass rather than against a person. What transfers to a clinical device is the control architecture and the firmware, not the actuator; Section 6.3 costs the substitution.

1.3.5. Low-Cost and Open Hardware Platforms

A smaller literature targets cost directly. Hapkit [37] demonstrated that a single-axis force-feedback device adequate for teaching haptics could be built for tens of dollars, and that publishing both hardware and firmware openly is what allows such a device to be replicated rather than merely admired. SimpleFOC [3] plays an analogous role for motor control, and much of the recent interest in gimbal motors as low-cost torque sources can be traced to it. On the rehabilitation side, Maciejasz et al. [5] note that simple single-joint devices remain under-represented relative to full exoskeletons, despite being the more plausible route to home-based therapy.

The gap addressed by this report lies at the intersection of these three literatures: a field-oriented control implementation small and readable enough to be taught and audited, running on the least expensive hardware that can plausibly support it, and evaluated specifically against the impedance behaviors rehabilitation practice requires, with the cost of that choice measured rather than asserted.

1.4. Objectives and Contributions

The project pursues four explicit objectives, which address sub-problems 1 to 4 of Section 1.2 in order:

- To design and implement complete FOC firmware (encoder decoder, PWM generator, low-side current-sense interface, dq mathematics, sensor-alignment routine, pole-pair check, PID/LPF, cascaded current and velocity loops, SVPWM modulator, and decoupling feed-forward) on the Arduino UNO + DRV8302 platform.
- To extend the firmware with an identification procedure that measures R, L_d, L_q and λ_{PM} with no auxiliary instrumentation beyond the motor itself, and an IMC-based automatic current-loop tuner that consumes those parameters.
- To realize an application on top of the impedance kernel: an upper-limb rehabilitation controller with a velocity-gated adaptive-scaling layer, tested via a serial command interface that allows runtime adjustment.
- To validate the firmware on an iPOWER GM4108-120T gimbal motor (11 pole pairs, AMT103 encoder, 12 V supply): to verify each algorithmic block on identical hardware against the behavior expected of a correct FOC implementation, taking the widely used SimpleFOC library [3] as a documented reference point (Section 5.7); and to validate the rehabilitation application across five clinically recognized exercise

modes — isometric, isotonic, isokinetic, adaptive-resistance and transparent back-drive (Section 5.8).

Beyond the working artifact, the project contributes:

- A complete mathematical derivation of each transformation block, written in a form intended to be extended in future work.
- A documented and quantified analysis of several implementation defects encountered during development - most notably a 90-degree alignment-offset error caused by the geometry of the Park transformation, in which a quadrature-axis voltage Vq applied at electrical angle 0 produces a stator field at 90 electrical degrees, so that the rotor d-axis settles 90 degrees away from the angle the firmware assumes - that are not always made explicit in motor-control texts.
- A reusable test methodology for PMSM parameter identification on ADC-limited microcontrollers, including the boost of the AVR ADC clock from 125 kHz to 1 MHz required to resolve the L/R time constant of small gimbal motors.
- A quantitative validation methodology that expresses the rehabilitation application's claims as analytic predictions and confirms them with standard statistical tests - least-squares regression with R^2 for the stiffness and damping laws, and RMS-residual analysis of the adaptive-scaling law (Section 5.8).

1.5.Scope and Limitations

This work is explicitly scoped as follows:

- The target machine class is a small surface-PMSM ($L_d \approx L_q$) such as a hobby gimbal motor; the firmware does not attempt field-weakening, interior-PMSM handling, or salient-pole maximum-torque-per-ampere control.
- The DC bus voltage is fixed at 12 V; bus-voltage feed-forward and dynamic V_{bus} measurement is not implemented.
- Position feedback is from an incremental quadrature modular encoder; absolute encoders, magnetic angle sensors and sensorless observers are out of scope.
- The inner current-loop sample rate is bounded above by the ATmega328P's ADC and floating-point performance; the design targets approximately 1.5 kHz nominal loop execution rate at the current axis, which is conservative for the 31.4 kHz PWM carrier but appropriate for the host MCU. All experimental results in this project are obtained in voltage-mode commutation, with the current/torque loop open (no current feedback); the closed-loop current controller is implemented and was brought up on the bench, but its experimental validation is out of scope and is treated as future work (Chapter 6).
- Human clinical evaluation of the rehabilitation application is out of scope; the focus is on the engineering of the actuation layer and its programmable interface.

1.6.Organization of the Report

The remainder of this report is organized as follows. Chapter 2 develops the theory on which the library rests: the three-phase permanent-magnet synchronous-machine model, the Clarke and Park transformations, space-vector modulation, quadrature decoding, the cascaded control structure, and impedance and admittance control. Chapter 3 presents the hardware,

software, control and safety architecture derived from the project requirements. Chapter 4 documents the implementation module by module and traces the complete impedance-control signal path through the firmware and inverter. Chapter 5 describes the experimental apparatus and procedures and reports the results, including hardware bring-up, motor-parameter identification, the inductance-measurement limitation and remedies, FOC verification, clinical-mode testing, and the measured back-drivability of the as-built bench. Chapter 6 states the limitations candidly, outlines the work and certification path required for clinical deployment, and concludes. Appendix A lists the source-file organization, Appendix B collects the mathematical derivations, and Appendix C contains dimensioned engineering drawings of the custom mechanical parts.

Chapter 2
Theoretical Background

2.1. The Three-Phase PMSM

A permanent-magnet synchronous machine consists of a three-phase stator winding distributed at 120 electrical degrees around the bore, and a rotor carrying permanent magnets that establish a fixed magnetic flux λ_{PM} linked into each stator phase as a sinusoidal function of the rotor's electrical angle θ_e. The relationship between mechanical and electrical angles is determined by the number of magnetic pole pairs (pp) on the rotor: $\theta_e = pp \cdot \theta_m$. The motor used in this work has $pp = 11$, so one mechanical revolution corresponds to eleven electrical cycles.

Under the assumption of a balanced winding, sinusoidal back-EMF (a well-designed surface PMSM), magnetic linearity, and a star (Y) connection with no neutral wire, the per-phase voltage equations are [1], [2]:

$$v_a = R \cdot i_a + L \cdot \frac{di_a}{dt} + e_a \tag{2.1}$$

$$v_b = R \cdot i_b + L \cdot \frac{di_b}{dt} + e_b \tag{2.2}$$

$$v_c = R \cdot i_c + L \cdot \frac{di_c}{dt} + e_c \tag{2.3}$$

where R is the stator phase resistance, L is the per-phase self-inductance (mutual inductances are absorbed into an effective self-inductance for the balanced case), and e_a, e_b, e_c are the per-phase back-EMFs. The three back-EMFs are therefore:

$$e_a = -\lambda_{PM} \cdot \omega_e \cdot sin(\theta_e) \tag{2.4}$$

$$e_b = -\lambda_{PM} \cdot \omega_e \cdot sin(\theta_e - \frac{2\pi}{3}) \tag{2.5}$$

$$e_c = -\lambda_{PM} \cdot \omega_e \cdot sin(\theta_e + \frac{2\pi}{3}) \tag{2.6}$$

The minus sign follows from Faraday's law rather than from a sign convention: with the flux linked by phase a written as $\lambda_a = \lambda_{PM} cos(\theta_e)$, the induced voltage is $e_a = \frac{d\lambda_a}{dt} = -\lambda_{PM}\omega_e sin(\theta_e)$, so the derivative of the cosine supplies the sign and places the back-EMF in quadrature with the flux linkage.

The electromagnetic torque developed by the machine is obtained from the power balance and, after writing the back-EMF in terms of the stator currents, reduces (in the rotor reference frame derived below) to the compact form:

$$\tau = \frac{3}{2} \cdot pp \cdot \lambda_{PM} \cdot i_q = K_t \cdot i_q \tag{2.7}$$

for a surface PMSM ($L_d \approx L_q$, so the reluctance term vanishes). The torque constant $K_t = \frac{3}{2} \cdot pp \cdot \lambda_{PM}$ links the rotor-frame quadrature current directly to the mechanical torque, which is the foundation of every current-mode FOC design including the one developed here.

2.2.Reference-Frame Transformations

Direct control of the three-phase sinusoidal currents at runtime is unattractive: the controller would have to track a moving target whose frequency varies with rotor speed. The classical remedy is a two-step transformation of the abc quantities first into a two-axis stationary frame $\alpha\beta$ (Clarke), then into a two-axis frame dq that rotates synchronously with the rotor (Park). In dq, the steady-state quantities are constant, so PI control becomes straightforward. Figure 2.1 shows the complete FOC signal flow assembled from these transformations together with the current and outer-loop controllers. In the forward path the d- and q-axis current PI controllers feed the inverse Park and inverse Clarke transformations and the min–max space-vector modulator, which drives the DRV8302 inverter. The rotor electrical angle θ_e, derived from the AMT103 encoder, rotates both the forward and the feedback transformations, and the measured phase currents are returned through Clarke and Park to close the current loop, with the cross-axis decoupling feed-forward added to the PI outputs. The dashed path is the voltage-mode fallback used for the experiments reported in this project, in which v_q is taken directly from the impedance or torque law and the current loop is bypassed. The remainder of this chapter derives each block in turn, and the inner loop *loopFOC()* realizes the same flow in firmware.

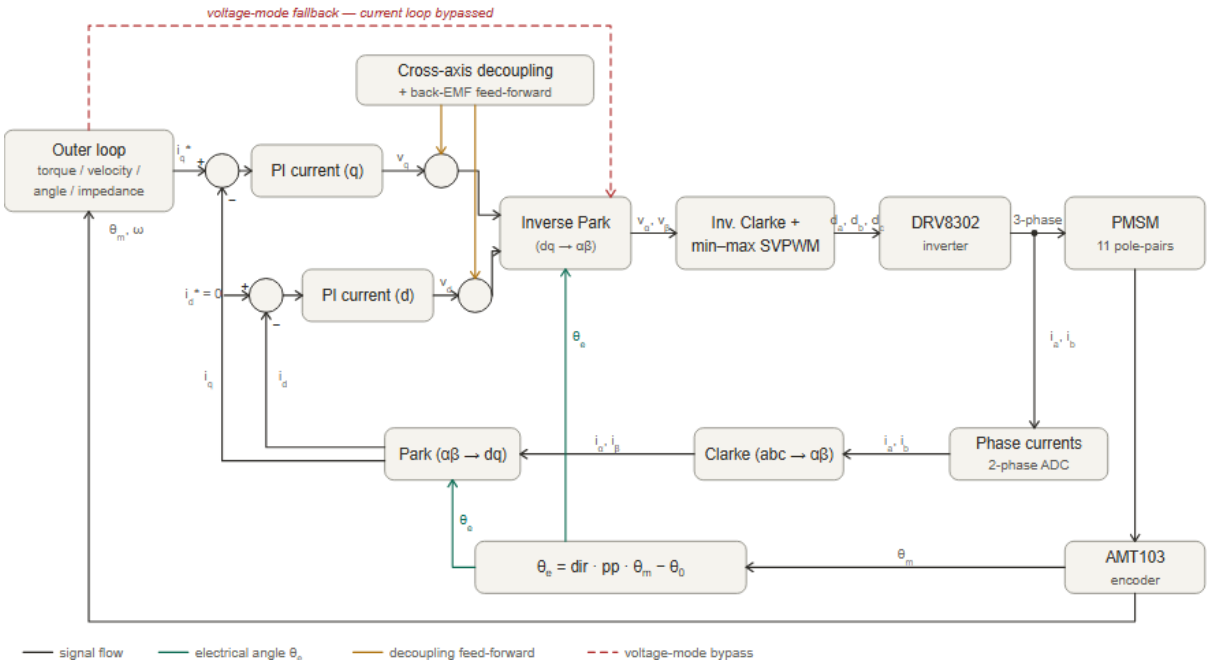


Figure 2.1: Complete field-oriented control signal flow as implemented in MyFOC. The dashed path is the voltage-mode fallback used for the experiments reported here.

2.2.1. Clarke Transformation ($abc \rightarrow \alpha\beta$)

The three stator axes are physically displaced by 120 mechanical degrees. Choosing the α-axis to coincide with phase A and the β-axis to lie 90 electrical degrees ahead in the counterclockwise sense, the unit vectors of the three phase axes resolved in the αβ plane are $(1,0)$, $(-\frac{1}{2},\frac{\sqrt{3}}{2})$ and $(-\frac{1}{2},-\frac{\sqrt{3}}{2})$. The α and β components of any three-phase quantity are the projections onto these axes, scaled by a convention factor k [12]:

$$v_\alpha = k \cdot \left(v_a \cdot 1 + v_b \cdot \left(\frac{-1}{2}\right) + v_c \cdot \left(\frac{-1}{2}\right) \right) \tag{2.8}$$

$$v_\beta = k \cdot \left(v_a \cdot 0 + v_b \cdot \left(\frac{\sqrt{3}}{2}\right) + v_c \cdot \left(\frac{-\sqrt{3}}{2}\right) \right) \tag{2.9}$$

Two scaling conventions are in common use. The amplitude-invariant convention $k = \frac{2}{3}$ yields v_α and v_β whose magnitude equals the peak of the original abc sinusoids, and is the convention adopted by SimpleFOC [3], most automotive supplier libraries and most pedagogical textbooks [1], [2]. The power-invariant convention $k = \sqrt{\frac{2}{3}}$ preserves the total instantaneous three-phase power between the abc and $\alpha\beta$ frames; it is preferred in analytical treatments of machine impedance but is less common in practical firmware. MyFOC uses the amplitude-invariant convention throughout for comparability with SimpleFOC:

$$v_\alpha = \left(\frac{2}{3}\right) \cdot \left(v_a - \frac{v_b + v_c}{2} \right) \tag{2.10}$$

$$v_\beta = \frac{v_b - v_c}{\sqrt{3}} \tag{2.11}$$

For balanced abc systems ($v_a + v_b + v_c = 0$), equations (2.10) and (2.11) reduce to $v_\alpha = v_a$ and $v_\beta = \frac{v_a + 2v_b}{\sqrt{3}}$. The unbalanced form is implemented to remain valid during transients, line-to-line faults and the open-loop test sequences used by the identification routines.

2.2.2. Park Transformation ($\alpha\beta \rightarrow dq$)

The Park transformation [13] rotates the $\alpha\beta$ frame by $-\theta_e$ (i.e. into a frame that co-rotates with the rotor's d-axis) so that DC quantities in the steady state correspond to fixed stator-current vectors locked to the rotor magnet:

$$[v_d] = [cos\theta_e \;\; sin\theta_e][v_\alpha] \tag{2.12}$$

$$[v_q] = [-sin\theta_e \;\; cos\theta_e][v_\beta] \tag{2.13}$$

By convention v_d (the direct-axis component) is aligned with the rotor's magnet (N-pole) and contributes only to magnetisation; v_q(the quadrature component) is 90 electrical degrees ahead and is the torque-producing component. The inverse transformation is the corresponding $+\theta_e$ rotation:

$$[v_\alpha] = [cos\theta_e \;\; -sin\theta_e][v_d] \tag{2.14}$$

$$[v_\beta] = [sin\theta_e \;\; cos\theta_e][v_q] \tag{2.15}$$

The geometric content of these formulas is that a quadrature-axis voltage applied at electrical angle θ_e produces a stator MMF vector at angle $\theta_e + \frac{\pi}{2}$ in the $\alpha\beta$plane.

2.2.3. PMSM Model in the dq Frame

Applying Clarke and Park transformations to the three-phase voltage equations and gathering terms, the PMSM dynamic model in the rotor reference frame is [1], [2], [16]:

$$v_d = R \cdot i_d + L_d \cdot \frac{di_d}{dt} - \omega_e \cdot L_q i_q \tag{2.16}$$

$$v_q = R \cdot i_q + L_q \cdot \frac{di_q}{dt} + \omega_e \cdot (L_d \cdot i_d + \lambda_{PM}) \tag{2.17}$$

Two features of this model are pivotal for control design. First, the d-axis equation contains a cross-coupling term $-\omega_e \cdot L_q \cdot i_q$ that depends on the q-axis current, and the q-axis equation contains the dual term $+\omega_e \cdot L_d \cdot i_d$ together with the speed-proportional back-EMF $\omega_e \cdot \lambda_{PM}$. From the PI controller's perspective these are disturbances; they are linear in measured quantities (ω_e, i_d, i_q) and known parameters (L_d, L_q, λ_{PM}) and can therefore be cancelled by a feed-forward term.

Second, with the cross-coupling cancelled, each axis becomes a stable first-order plant of the form $v = R \cdot i + L \cdot \frac{di}{dt}$, whose transfer function in the Laplace domain is $\frac{i(s)}{v(s)} = \frac{1}{L \cdot s + R}$. This is the canonical first-order target that the Internal-Model-Control rule of Section 2.5.3 is designed to invert.

2.3.Space-Vector Pulse-Width Modulation

Once the inverse Park and inverse Clarke transformations have produced reference phase voltages v_a, v_b, v_c, the inverter must reproduce them via PWM. Two practical strategies exist:

2.3.1. Sinusoidal PWM

In sinusoidal PWM each phase reference is offset to $\frac{V_{DC}}{2}$ and divided by V_{DC} to yield a duty cycle in [0, 1]. This is mathematically transparent but limits the achievable peak phase voltage to $\frac{V_{DC}}{2}$ and therefore the maximum line-to-line voltage to $V_{DC} \cdot \frac{\sqrt{3}}{2} \approx 0.866 \cdot V_{DC}$; the inverter's full DC bus is never fully utilized.

2.3.2. Space-Vector PWM with Min-Max Injection

Because the three motor windings are connected as a floating star with no neutral wire, the common-mode component of (v_a, v_b, v_c) does not appear across any winding. Adding the same offset to all three phases is therefore invisible to the motor but allows the modulator to reduce the peak excursion of any one phase. The classical sector-based SVPWM formulation [18], derived by representing the desired voltage as a sum of two adjacent active vectors and a zero vector, is mathematically equivalent (within rounding) to the more compact min–max injection scheme [19]:

$$v_{offset} = \frac{max(v_a, v_b, v_c) + min(v_a, v_b, v_c)}{2} \tag{2.18}$$

$$v'_a = v_a - v_{offset} \ (and\ analogously\ for\ b, c) \tag{2.19}$$

$$duty_x = \left(\frac{v'_x}{V_{DC}}\right) + 0.5 \tag{2.20}$$

The result extends the peak achievable phase voltage to $\frac{V_{DC}}{\sqrt{3}} \approx 0.577 \cdot V_{DC}$ and the line-to-line voltage to V_{DC}, a 15.5 % improvement in modulation depth over sinusoidal PWM. MyFOC uses the min–max formulation for all closed-loop FOC operation; the sinusoidal form is provided as a separately-callable function for instructional and comparison purposes.

2.4.Quadrature Encoder Decoding

An incremental quadrature encoder produces two square-wave channels (A, B) that are 90 electrical degrees out of phase [38]. Each channel emits PPR pulses per mechanical

revolution, where PPR is the encoder's nominal pulses-per-revolution rating (2048 for the AMT103 as configured here). Detecting both the rising and the falling edge of both channels gives four counts per pulse, so the effective resolution is 4 × PPR counts per revolution (CPR) – 8192 counts for this encoder. The decoder must additionally determine the direction of motion, which is encoded in the relative phase: a transition (A, B) = (0, 0) → (1, 0) → (1, 1) → (0, 1) → (0, 0) indicates one direction, and the reverse sequence the other.

On the ATmega328P [39], only digital pins 2 and 3 support hardware external interrupts (INT0, INT1) at the resolution needed to capture every encoder edge without missing transitions at maximum motor speed. MyFOC requires the encoder to be wired to these two pins and configures CHANGE-mode interrupts on each. The ISR for each channel reads both channel states and updates a 32-bit pulse counter according to a logic table derived in Section 4.3.

2.5. Cascaded Control of a PMSM

2.5.1. Architecture

A FOC drive is organized as a cascade of nested loops. From innermost to outermost:

- Inner current loops (PI on $i_d$and on i_q): produce voltage references v_d, v_q. Run at the FOC sample rate (in this implementation, every iteration of *loopFOC()*, typically 1.0 to 1.5 kHz on the UNO).
- Velocity loop (PI on ω): consumes the velocity setpoint and produces an i_q reference. Saturates against the current limit. Runs at the same sample rate as the inner loops.
- Position loop (P on θ): consumes the position setpoint and produces a velocity reference. Saturates against the velocity limit.

The cascade is parameterized by the control mode: in TORQUE mode, the outer loops are bypassed, and the user-supplied target is interpreted as i_q (current mode) or v_q (voltage fallback); in VELOCITY mode only the position loop is bypassed; in ANGLE mode the full cascade is active; in IMPEDANCE mode the outer loops are replaced by the spring–damper law of Section 2.6.

2.5.2. Cross-Axis Decoupling Feed-Forward

With knowledge of L_d, L_q and λ_{PM}, the cross-coupling and back-EMF terms in the dq model of Section 2.2.3 can be cancelled by adding compensation terms to the PI outputs:

$$v_{d_{total}} = v_{d_{pi}} + \left(-\omega_e \cdot L_q \cdot i_q\right) \tag{2.21}$$

$$v_{q_{total}} = v_{q_{\pi}} + \left(\omega_e \cdot (L_d \cdot i_d + \lambda_{PM})\right) \tag{2.22}$$

The PI controllers then see a true single-axis first-order plant $\left(v = R \cdot i + L \cdot \frac{di}{dt}\right)$ on each axis independently, dramatically improving the dynamic response in current mode, especially during fast acceleration or load step transients. Decoupling is an opt-in feature in MyFOC -it requires identified parameters - and is gated by the $decoupling_{enabled}$ flag on the motor object.

2.5.3. Internal-Model-Control (IMC) Tuning of the Current Loops

Once decoupling has reduced each axis to the canonical first-order plant $G(s) = \frac{1}{L \cdot s + R}$, the textbook IMC tuning rule [40] for a desired closed-loop bandwidth ω_{bw} yields the PI gains:

$$K_p = L \cdot \omega_{bw} \tag{2.23}$$

$$K_i = R \cdot \omega_{bw} \tag{2.24}$$

Substituting and solving the closed-loop transfer function confirms that the resulting current-loop response is a first-order lag with time constant $\frac{1}{\omega_{bw}}$. The bandwidth ω_{bw} must be chosen below the PWM half-frequency (here 31.4 kHz / 2 ≈ 99 krad/s) and well below any sensor or controller delay; for the UNO platform a bandwidth of 800 rad/s (≈ 127 Hz) was found to be a good engineering compromise - fast enough to reject load disturbances on a rehabilitation timescale, slow enough to be insensitive to the finite ADC and floating-point latency.

The MyFOC identifier module computes both gains automatically from the measured R and L_d $(or\ L_q)$ once identification is complete, so the user need only specify the desired bandwidth.

2.5.4. Anti-Windup and Saturation

Every cascaded loop has a saturation limit downstream of its output: the current loop is bounded by $v_q \leq v_{limit}$; the velocity loop is bounded by $i_{q_{ref}} \leq i_{limit}$ when current sensing is enabled, or by $v_q \leq v_{limit}$ when voltage-mode fallback is active; the position loop is bounded by v_{limit}; and the impedance controller is bounded by $torque_{limit}$. Whenever an output is clamped, integrator wind-up degrades the controller's recovery from saturation.

MyFOC implements a simple but effective back-calculation form of anti-windup [41]: at each PI update, if the computed output exceeds the limit, the integrator is only updated in the direction that reduces saturation; conversely, it is held when its update would deepen saturation. The same logic is applied in both directions to handle bipolar setpoints.

A subtle requirement that surfaced during validation is that the PI's internal limit must agree with the physical limit at its output. When current sensing is enabled, the velocity PI's output is a current reference in amperes, and its internal limit must be set to i_{limit} rather than to the voltage limit; otherwise, the integrator winds up to many times the allowable current before its external clamp kicks in.

2.6. Impedance Control

Impedance control, introduced by Hogan in 1985 [7], treats the controlled actuator as a programmable mechanical impedance $Z(s) = \frac{F(s)}{V(s)}$ (or, equivalently in the rotational domain, $\frac{\tau(s)}{\omega(s)}$). Rather than enforcing a position or velocity reference and rejecting the operator's force as a disturbance, the controller deliberately yields to operator force in a manner specified by the chosen Z [8].

The simplest and most common parameterization is a linear second-order spring–damper, expressed as a torque law:

$$\tau = -K \cdot (\theta - \theta_{eq}) - B \cdot \omega \tag{2.25}$$

where K is the virtual rotational stiffness (N·m/rad), B is the virtual rotational damping (N·m·s/rad) and θ_{eq} is the position at which the spring exerts no force. The corresponding closed-loop natural frequency and damping ratio, for a single-degree-of-freedom load with effective rotor inertia J_{eff}, are:

$$\omega_n = \sqrt{\frac{K}{J_{eff}}} \tag{2.26}$$

$$\zeta = \frac{B}{2 \cdot \sqrt{K \cdot J_{eff}}} \tag{2.27}$$

The clinical relevance is that K and B can be tuned independently of the underlying actuator dynamics - provided the current loop's bandwidth is well above ω_n and the actuator is back-drivable - to produce the rehabilitation modes specified in Section 5.8. The conversion from the desired torque τ to the q-axis current reference for the FOC inner loop is the standard $\tau = K_t \cdot i_q$ relationship:

$$i_{q_{ref}} = \frac{\tau}{K_t} = \frac{\tau}{\left(\left(\frac{3}{2}\right) \cdot pp \cdot \lambda_{PM}\right)} \tag{2.28}$$

This conversion is the principal reason that λ_{PM} must be identified before impedance mode can function correctly. The MyFOC impedance controller refuses to produce nonzero output if $K_t < 10^{-6}$, returning the motor to an idle state until parameters are populated.

2.7.SimpleFOC as a Reference Implementation

SimpleFOC [3] is an open-source Arduino-compatible FOC library that supports a wide range of microcontrollers and gate-driver topologies including the DRV8302 [42]. Its implementation is C++ with template-style abstraction over driver and sensor types, and its FOC pipeline follows the same conceptual structure adopted in this project: sensor → electrical angle → optional current sensing → Park transform → PI control → inverse Park → SVPWM → inverter. SimpleFOC is widely deployed, well documented and benchmarked against industrial implementations; for the purposes of this project, it serves as a known-good comparison reference rather than as a substrate to be modified.

MyFOC deliberately matches SimpleFOC's algorithmic choices wherever a design decision has a defensible alternative - amplitude-invariant Clarke convention, min–max SVPWM, voltage-mode default with optional current-loop upgrade, and the SimpleFOC-style sensor alignment using a quadrature voltage drive - so that any empirically-observed differences can be attributed to micro-architectural or numerical implementation rather than fundamental algorithmic divergence.

2.8.BLDC and PMSM: The Distinction from a Firmware Perspective

The terms brushless DC (BLDC) motor and permanent-magnet synchronous machine (PMSM) are often used interchangeably in product literature, and both describe a three-phase stator driving a permanent-magnet rotor. The distinction that matters to firmware is the shape of the back-electromotive force as a function of rotor angle, because that shape determines which commutation strategy produces smooth torque [1], [2].

A machine described as BLDC is wound with concentrated coils so that its back-EMF is approximately trapezoidal: flat over roughly 120 electrical degrees, with sloped transitions between. The matching drive is six-step commutation, in which two of the three phases conduct at a time and the conducting pair changes six times per electrical cycle. Rotor position is needed only to six-state resolution, so three Hall sensors suffice, no reference-frame transformation is required, and the controller reduces to a lookup table plus a single current regulator. A machine described as PMSM is wound so that its back-EMF is approximately sinusoidal, and the matching drive is the sinusoidal field-oriented control developed in this chapter, which requires continuous rotor angle and the Clarke and Park transformations.

Applying a drive to the wrong back-EMF shape is not a fault, but it costs torque quality in a predictable way. Six-step commutation applied to a sinusoidal machine produces a torque ripple of about 13.4 % peak-to-peak, because the stator field advances in 60-degree jumps and the angle between stator field and rotor magnet therefore varies between 60 and 120 electrical degrees; the resulting torque follows the cosine of that error, and the worst case is cos(30°). Sinusoidal field-oriented control applied to a trapezoidal machine produces a smaller but non-zero ripple from the harmonic content the sinusoidal model does not represent. Table 2.1 summarizes what changes in firmware.

Table 2.1: BLDC and PMSM compared from the point of view of the control firmware.

Property	**BLDC (trapezoidal back-EMF)**	**PMSM (sinusoidal back-EMF)**
Matching commutation	Six-step, two phases conducting	Sinusoidal field-oriented control
Rotor-position requirement	Six states per electrical cycle (Hall sensors)	Continuous electrical angle
Firmware cost	Commutation table, one current loop	Clarke, Park, two PI loops, inverse transforms, modulation
Torque ripple at low speed	Low on its own machine; ≈ 13.4 % on a sinusoidal machine	Low on its own machine; small residual on a trapezoidal machine
Behaviour through zero speed	Stepped: torque changes as commutation state changes	Continuous: torque is set by i_q at any angle
Suitability for impedance rendering	Poor — ripple is felt as detent	Good — ripple is the limit on rendered smoothness

For a device that renders a virtual spring, the last two rows decide the matter. An isometric hold is a stall condition, and impedance rendering must pass continuously through zero speed and zero torque; a commutation scheme whose torque changes in steps is felt by the patient as detent, and no amount of outer-loop tuning removes it. The machine used here, the iPOWER GM4108-120T, is a surface-magnet gimbal motor with a near-sinusoidal back-EMF, so it is a PMSM in the sense that matters, and sinusoidal field-oriented control is the correct match. That this project therefore pays the full firmware cost of the third column — both transformations, both current loops, the modulator — is a consequence of the application, not a preference: the smoothness that column buys is precisely the deliverable.

2.9. Impedance Control Compared with Admittance Control

Impedance control is not the only way to regulate the mechanical interaction between a robot and a limb. Its dual, admittance control, inverts the causality: where an impedance controller measures motion and commands force, an admittance controller measures force and commands motion, typically by feeding a force-sensor signal through an admittance model to generate a position reference for a stiff inner position loop [43]. Both render a programmable dynamic relationship at the interface, and both are used in deployed rehabilitation devices, so the choice made here requires justification rather than assertion.

The two schemes impose opposite requirements on the hardware. Impedance control takes motion as its input, so the actuator must be back-drivable and low-friction: if the device cannot be moved by the patient, there is no input signal, and any friction the actuator possesses is added to the impedance being rendered and cannot be removed by the controller. Admittance control takes force as its input, so it requires a force or torque transducer, and it prefers a stiff, high-reduction, non-back-drivable actuator whose inner position loop can reject the disturbance the patient represents. Table 2.2 sets the comparison out in full.

Table 2.2: Impedance and admittance control compared for a rehabilitation application.

Aspect	Impedance control (used here)	Admittance control
Causality	Measure motion, command force	Measure force, command motion
Sensor required	Position; torque inferred from current	Force or torque transducer, necessarily
Preferred actuator	Back-drivable, low friction, low reduction	Stiff, high reduction, need not be back-drivable
Renders low impedance	Well — limited by friction floor	Poorly — limited by sensor noise and loop bandwidth
Renders high stiffness	Poorly — limited by loop bandwidth	Well — limited by structural stiffness
Coupled to a stiff environment	Stable	Can become unstable
Coupled to a soft environment	Can become unstable at high rendered stiffness	Stable
Behaviour under actuator fault	Yields — device can be overpowered	Can drive — position loop retains authority

Three arguments decide the choice for this project, in order of weight. The first is passivity, and it is a safety argument. Hogan and Buerger show that an impedance controller coupled to an unknown human limb remains passive provided the rendered stiffness and damping are positive and the rendered dynamics lie within what the actuator can follow [8]. A passive device cannot inject energy into the interaction, so a controller fault results in the device yielding rather than driving. That property is exactly what the layered torque ceiling of Section 3.4 is designed to preserve, and it is why the ceiling is applied to the output of the impedance law rather than inside it.

The second is hardware. Admittance control requires a force or torque transducer as a matter of principle, not of convenience: without it there is no input. An in-line rotary torque sensor costs between USD 1500 and USD 5000, which is more than an order of magnitude above

the entire bench, so admittance control is incompatible with the cost premise stated in Section 1.2. Impedance control is the only one of the two that can be implemented at all on this hardware.

The third is the failure mode. Admittance control becomes unstable when coupled to a stiff environment, and a spastic limb or a tightly gripped handle is precisely a stiff environment. Impedance control fails in the opposite direction, struggling to render very high stiffness, which for a rehabilitation device is a much safer limitation to have. The one regime in which admittance control is clearly preferable is a heavy geared device whose large, reflected inertia must be masked from the patient — the regime of the commercial exoskeletons surveyed in Section 1.3.2, and not the regime of a direct-drive bench.

Chapter 3
System Design and Architecture

The apparatus described in this chapter is not a general-purpose motor testbed that was later applied to rehabilitation; it was specified from the problem stated in Section 1.2. Six requirements were derived from that problem, and every significant design decision in the remainder of this chapter can be traced to one of them.

- **R1 - Smooth bidirectional torque at and near zero speed.** Impedance rendering requires the actuator to produce commanded torque continuously through zero speed and zero torque, without the cogging steps or commutation ripple of six-step drive. This requirement dictates field-oriented control with space-vector modulation, and an eleven-pole-pair machine whose electrical period is short relative to the mechanical one (Section 3.3).
- **R2 - Mechanical transparency.** A patient must be able to back-drive the device with minimal resistance so that residual voluntary effort is measured rather than masked. Reflected inertia and friction scale with the square of the gear ratio, so the requirement is met by direct drive: no gearbox, no belt, and the load mounted on the rotor itself [26], [28] (Section 3.1).
- **R3 - Layered safety.** Because the actuator is coupled to a person, no single fault may produce an unbounded torque. The requirement is met by limits applied at successive layers, from the commanded voltage and current through the impedance-law output to the driver-enable line (Section 3.4).
- **R4 - Cost.** The premise of the project is that the electronics must be affordable enough to be duplicated freely, which fixes the platform as an Arduino UNO with a DRV8302 driver stage and a gimbal motor, for a bench cost of approximately $135 (Section 3.1).
- **R5 - Clinical expressiveness.** One controller must reproduce the isometric, isotonic, isokinetic, adaptive-assistance and back-drive behaviors of rehabilitation practice. The requirement is met by a single impedance kernel whose stiffness, damping and reference terms are set at run time over the serial interface, rather than by five separate control laws (Sections 3.2 and 3.3).
- **R6 - Reproducibility and auditability.** The control path must be readable end to end and derivable from first principles, which is why the library is written from scratch rather than configured, why every transformation is derived in Appendix B, and why the identification procedure of Chapter 4 uses only instruments available on a student bench (Section 3.2).

Requirements R1, R2 and R4 are in direct tension: the cheapest motor with the lowest reflected inertia is also the one with the least favorable torque constant and the poorest thermal margin, and the 8-bit platform mandated by R4 constrains the control bandwidth available to satisfy R1. How that tension was resolved, and what it cost in fidelity, is the subject of the remainder of this chapter and of Chapter 6.

3.1. Hardware Overview

The target platform was selected to meet three constraints: it must be inexpensive and widely available to undergraduate laboratories; it must include a gate driver suitable for a small

three-phase BLDC motor with onboard current sensing; and it must allow the firmware to be ported to higher-performance microcontrollers with minimal effort. The chosen configuration is the ATmega328P [39] with the DRV8302 gate driver [42] and an AMT103 incremental encoder [38], summarized in Table 3.1.

The actuator itself was selected for the specific demands of impedance rendering rather than for raw output. A device that emulates a virtual spring–damper must produce smooth, accurately controlled torque at very low and at zero speed — an isometric hold is a stall condition — must be freely back-drivable so that the patient can always overpower it and must be safe in direct contact with a limb. The IPOWER GM4108-120T is a high-pole-count outrunner surface-PMSM of the type developed to stabilise camera platforms, and it meets these demands directly. Its eleven pole pairs (twenty-two rotor magnets) give eleven electrical cycles per mechanical revolution, so the commutation torque is distributed finely around the rotor and the residual cogging and torque ripple that would otherwise corrupt a rendered spring are small. The motor is wound with many turns of fine wire — the 120T designation — which yields a high phase resistance (11.16 Ω) and a high torque constant (0.570 N·m/A) at a correspondingly low velocity constant: it produces useful torque at a fraction of an ampere and at standstill rather than requiring high current or high speed. This is the opposite of the optimisation used in the brushless motors sold for drones and radio-controlled vehicles, which are wound for a high velocity constant so as to spin propellers and wheels at tens of thousands of revolutions per minute; those motors are thermally stressed at stall and coarse in their torque control near zero speed, which is exactly the regime a haptic device operates in.

Two further properties reinforce the choice. First, the motor uses direct drive, with the lever fixed straight to the rotor and no reduction gearing. A gearbox would multiply torque but introduce backlash, added friction and reflected inertia, all of which degrade transparency [26], [28] and make the device feel notchy or sticky rather than rendering the commanded impedance cleanly; direct drive preserves the low-friction back-drivability on which impedance control depends, and the transparency (back-drive) test of Section 5.8 is only meaningful because the actuator is direct-drive. Second, a surface-mounted-magnet PMSM of this kind has a near-sinusoidal back-EMF, which matches the sinusoidal field-oriented commutation used throughout this work; a trapezoidal-back-EMF BLDC motor driven with sinusoidal FOC would exhibit additional torque ripple. The high-torque-constant, low-current operating point also keeps the electrical power, and therefore the energy that could be delivered to a limb, low: a small gate driver and a 0.6 A current limit suffice, which is appropriate for a device intended for human contact.

Smoothness at this level is not unique to small gimbal motors. The same design principles — a high pole count, a low-cogging slot-and-magnet geometry and direct drive — are used in the frameless direct-drive torque motors found in robot joints, telescope mounts and collaborative-robot actuators, which deliver the same low-ripple behaviour at far higher torque; a gimbal motor is essentially a small, inexpensive instance of that class. Those larger machines, however, demand proportionally larger drive currents, bigger gate drivers and more careful thermal design, and typically cost one to two orders of magnitude more, none of which suits a graduate-accessible platform. The gimbal motor’s limitations are the corollary of its strengths: the fine windings that give it a high torque constant also cap its continuous current, and hence its continuous torque and power, and its usable speed range is modest. For

a benchtop trainer working against light loads — the 0.30 N·m ceiling and 200 g deadweight of Section 5.8 — these limits are not binding, but a device required to deliver larger forces, or higher speeds would move to a frameless torque motor of the kind just described.

Table 3.1: Hardware configuration.

Component	**Specification**
Microcontroller	Atmel/Microchip ATmega328P (Arduino UNO Rev3)
Clock frequency	16 MHz (external crystal)
Program memory	32 KB Flash (29 KB available after bootloader)
Data memory	2 KB SRAM
EEPROM	1 KB
Gate driver	Texas Instruments DRV8302 in 3-PWM mode
DC bus voltage	12 V
Current-sense gain	10× (default GAIN strap), shunt 5 mΩ → 0.05 V/A
Motor	IPOWER GM4108-120T 11-pole-pair surface-PMSM gimbal motor
Position sensor	CUI AMT103 incremental quadrature encoder, configured at 2048 PPR (8192 CPR after 4× decoding)
DRV protection signals	nFAULT (fault), nOCTW (over-current warning), open-drain active-LOW

All three high-side gate inputs (INH_A, INH_B, INH_C) are driven from Arduino pins 9, 10 and 11 respectively. The DRV8302's internal current-sense amplifier outputs SO1 and SO2 are routed to analog inputs A2 and A3; this routing is constrained by the user's wiring of the protection signals nOCTW and nFAULT to A0 and A1. The driver enable input EN_{GATE} is on pin 8. The three configuration strap pins M_{OC}, M_{PWM} and OC_{ADJ} are tied via Arduino pins 5, 6 and 7 to LOW, HIGH and HIGH respectively, selecting 3-PWM mode with the hardware over-current protection active and the over-current threshold at its maximum factory setting. Pin assignments are tabulated in Table 3.2.

Table 3.2: Pin assignments.

Signal	Pin
H_A (high-side PWM A)	Arduino digital pin 9 (Timer1 OC1A)
H_B (high-side PWM B)	Arduino digital pin 10 (Timer1 OC1B)
H_C (high-side PWM C)	Arduino digital pin 11 (Timer2 OC2A)
EN_{GATE}	Arduino digital pin 8

Signal	Pin
M_{PWM} (mode selector)	Arduino digital pin 6 ← HIGH (3-PWM mode)
M_{OC} (OC enable)	Arduino digital pin 5 ← LOW (OC protection on)
OC_{ADJ} (OC threshold)	Arduino digital pin 7 ← HIGH (maximum threshold)
SO1 (CS amp A)	Arduino analog A2
SO2 (CS amp B)	Arduino analog A3
nOCTW (over-current warning)	Arduino analog A0 (used as digital input with pull-up)
nFAULT (fault output)	Arduino analog A1 (used as digital input with pull-up)
Encoder A	Arduino digital pin 2 (INT0)
Encoder B	Arduino digital pin 3 (INT1)

3.2.Software Architecture

The firmware is organised as a single Arduino library, MyFOC, with one C++ namespace MyFOC enclosing all classes and free functions. Each library module has a single responsibility, and the public interface is small enough so that the entire library can be visualized at once. Modules are listed in Table 3.3.

Table 3.3: Module decomposition of the MyFOC library.

Module	**Kind**	**Responsibility**
MyFOCMath	header + impl	Mathematical kernel: angle normalisation, fast sin/cos via 65-entry quarter-wave LUT in PROGMEM, Clarke, Park, inverse Park, inverse Clarke, sinusoidal PWM and min–max SVPWM.
MyPID	header + impl	Discrete-time PID controller with trapezoidal integration, adaptive sample time (micros()), output saturation, back-calculation anti-windup, and optional output ramp limiting.
MyLPF	header + impl	First-order low-pass filter with adaptive sample time and tunable time constant Tf.

Module	Kind	Responsibility
MyEncoder	header + impl	Quadrature encoder driver using CHANGE-mode external interrupts on pins 2/3. 4× decoding; signed long position counter; instantaneous velocity from finite-differences in radians per second.
MyBLDCDriver3PWM	header + impl	Three-phase 3-PWM inverter driver. Configures Timer1 + Timer2 for 31.4 kHz phase-correct PWM and writes OCR registers directly.
MyCurrentSense3PWM	header + impl	Two-channel DRV8302 current-sense reader. Boots the AVR ADC clock to 1 MHz; calibrates per-channel zero offsets; reconstructs phase C from $i_a + i_b + i_c = 0$.
MyBLDCMotor	header + impl	Top-level motor object: sensor-alignment, pole-pair check, all four control modes, inner current loop with optional decoupling, voltage-mode fallback, CSV monitoring sink.
MyMotorID	header + impl	Motor-parameter identification (MyMotorIdentifier class): open-loop resistance at a parked d-axis, d-axis inductance from a step response, flux linkage from a back-EMF spin, and IMC current-loop auto-tuning from the identified R and L.
MyFOC.h	umbrella	Single-included header that pulls in all of the above.

The total source size is approximately 2,500 lines of C++ across 17 files (8 headers, 8 source files, 1 umbrella). The compiled footprint on AVR is about 21 KB of program memory and zero static SRAM beyond per-object instance state, leaving over 10 KB of flash and the bulk of SRAM for user-application code.

Rehab_Device.ino - Application: impedance control with runtime-tunable K, B and θ_eq, a velocity-gated low-pass-filtered adaptive scaling of stiffness, damping and ceiling, hardware fault monitoring on nFAULT and nOCTW, 50 Hz CSV logging with per-procedure T<n> tagging.

3.3.Control-System Architecture

The inner loop *loopFOC()* runs as fast as the main loop will allow, typically every 700 µs to 1 ms depending on user code overhead; the outer loop *move()* runs at the same rate and consumes the latest sensor readings. Each loop iteration:

- Reads the current encoder count atomically (ISRs may have updated it since the previous loop) and converts to *shaft_angle* in radians.
- Computes the electrical angle via $\theta_e = dir \cdot pp \cdot shaft_{angle} - zero_{elec_{angle}}$, with $dir \in \{+1, -1\}$ encoding the empirically determined direction sense established during alignment.
- Reads the two-phase currents from ADC, reconstructs i_c, applies the forward Clarke and Park transforms to obtain (i_d, i_q) in the rotor frame.
- Computes the q-axis error $err_q = i_{q_{target}} - i_q$ and runs the PI_{iq} controller to produce v_{q_π}; similarly produces v_{d_π} from the d-axis error (which has target 0 for non-field-weakening operation).
- Adds the cross-axis decoupling feed-forward terms (Section 2.5.2) if *decoupling_enabled* is set and parameters have been identified.
- Saturates v_d and v_q against *voltage_limit*, calls the inverse Park, inverse Clarke and min–max SVPWM, and writes the resulting duty cycles to the Timer1/Timer2 OCR registers.

The outer loop runs after the inner one. Depending on motor.controller it: (TORQUE) sets *i_q_target* directly from motor.target; (VELOCITY) runs *PI_velocity* on (*target−shaft_velocity*) and saturates the output to *current_limit* before assigning to *i_q_target*; (ANGLE) runs *P_angle* on the position error to produce a velocity setpoint, saturates against *velocity_limit*, then runs *PI_velocity* to produce *i_q_target*; (IMPEDANCE) computes $\tau_{des} = -K(\theta - \theta_{eq}) - B\omega$, saturates against *torque_limit*, and converts to *i_q_target* via the torque constant K_t.

When the current sensor is not linked (voltage-mode fallback), all of the above paths instead write directly into *voltage_q*, and *loopFOC()* bypasses the current loop and Park-based feedback: it computes θ_e and immediately calls *setPhaseVoltage(voltage_q, 0, θ_e)*. This degraded mode reproduces the classical SimpleFOC voltage-mode behaviour and is retained as a usability fallback in environments where the current-sense routing is unavailable.

3.4.Safety Architecture

Multiple independent layers of saturation guard against unsafe operating conditions:

- Inverter duty saturation: each PWM duty cycle is clamped to [0, 1] inside *setDutyCycle()* so an out-of-bound *v_α, v_β* cannot produce a register overflow.
- Voltage saturation: $|v_d|, |v_q| \leq$ *voltage_limit* (default 12 V; 8 V in the rehab demo).
- Current saturation: $|i_q,ref| \leq$ *current_limit* (default 2.0 A; 0.6 A in the rehab demo).
- Torque saturation: in impedance mode, $|\tau_des| \leq$ *torque_limit* (default 0.08 N·m in the rehab demo; temporarily raised to 0.30 N·m for the controlled isometric calibration in Section 5.8).
- Velocity saturation: in angle mode, $|\omega_setpt| \leq$ *velocity_limit*; *PID_velocity.limit* is also dynamically reassigned (*current_limit* in current mode, *voltage_limit* in voltage mode) so that anti-windup operates at the correct level.
- Hardware over-current protection: the DRV8302's internal OC comparator latches nFAULT low if any half-bridge sees current beyond the threshold programmed by

OC_ADJ; the rehab sketch monitors this signal and calls *driver.disable()* within one main-loop iteration when it asserts.

- Soft over-current warning: nOCTW asserts low when current exceeds a warning threshold below the latched OC trip; the sketch prints a serial warning so that an operator can reduce torque demand.

These layers reduce the chance that a single command or software error produces unbounded actuation, but the bench does not provide a clinically independent single-fault-safe torque limit. The rehabilitation demo uses a conservative 0.08 N·m default software torque ceiling, backed by the DRV8302 over-current trip; the controlled isometric calibration in Section 5.8 temporarily raises this ceiling to 0.30 N·m. Clinical deployment would require additional independent layers, such as a watchdog and a mechanical torque limiter; the present scope is the firmware and bench-protection layer.

Chapter 4
Implementation

4.1. Mathematical Kernel (MyFOCMath)

4.1.1. Sin/Cos Lookup Table

The ATmega328P does not have a hardware floating-point unit. The standard *math.h sinf()* and *cosf()* functions are implemented in software using polynomial approximation and take approximately 120 µs per call on the 16 MHz core. For a FOC inner loop in which sin and cos are needed twice per iteration (once in Park, once in inverse Park) and a 1 kHz loop rate, that overhead alone consumes nearly 50 % of the available CPU. A lookup-table approximation is therefore necessary.

MyFOCMath provides a 65-entry table covering $\left[0, \frac{\pi}{2}\right]$ in 1.4° steps, stored in PROGMEM to avoid consuming the limited 2 KB SRAM. Quarter-wave symmetry extends the table to the full [0, 2π) domain:

- Q1, $0 \leq \theta < \pi/2$: $\sin\theta = \text{table}[\text{idx}]$
- Q2, $\pi/2 \leq \theta < \pi$: $\sin\theta = \text{table}[64 - (\text{idx from } \pi/2)]$
- Q3, $\pi \leq \theta < 3\pi/2$: $\sin\theta = -\text{table}[\text{idx from } \pi]$
- Q4, $3\pi/2 \leq \theta < 2\pi$: $\sin\theta = -\text{table}[64 - (\text{idx from } 3\pi/2)]$

Linear interpolation between table entries yields an error below 1×10^{-4} across the full domain, well below the precision available from the 10-bit ADC feedback. The single-call *sinCos(θ, s, c)* is provided as a convenience for the two trigonometric values needed at every Park step. Each LUT call is approximately 9 µs, a 13× speedup over the *math.h* implementation.

4.1.2. Clarke, Park, Inverse Park, Inverse Clarke

The four core transformations are implemented as inline functions and follow the conventions established in Section 2.2. The amplitude-invariant Clarke is used throughout. For surface-PMSM and balanced-phase operation the inverse Clarke assumes $v_a + v_b + v_c = 0$, which holds for any output of the min–max SVPWM modulator since the modulator subtracts the common-mode component. Each transform compiles to roughly 40 instructions on AVR and executes in under 15 µs.

4.1.3. Space-Vector PWM

The min–max injection from Section 2.3.2 is implemented as a single function:

```
inline void svpwm(float va, float vb, float vc, float v_dc,
                  float& duty_a, float& duty_b, float& duty_c) {
    float vmax = va;
    float vmin = va;
    if (vb > vmax) vmax = vb;
    if (vc > vmax) vmax = vc;
    if (vb < vmin) vmin = vb;
    if (vc < vmin) vmin = vc;
    float voff = 0.5f * (vmax + vmin);
    duty_a = (va - voff) / v_dc + 0.5f;
    duty_b = (vb - voff) / v_dc + 0.5f;
```

```
    duty_c = (vc - voff) / v_dc + 0.5f;
    // saturation to [0, 1] follows
}
```

The output duty cycles are clamped to [0, 1] to protect against numerical overshoot. Total execution time on AVR is approximately 25 µs.

4.2. PWM Driver (MyBLDCDriver3PWM)

The driver dedicates two of the ATmega328P's three timers [39] to PWM generation. Timer 0 is reserved for the Arduino core's *millis()/delay()* services, so Timer 1 (16-bit) and Timer 2 (8-bit) carry the three PWM outputs:

- Timer 1: WGM[3:0] = 0001 (Mode 1, 8-bit phase-correct PWM, TOP = 0xFF). COM1A = COM1B = non-inverting. CS = 001 (no prescaler). Outputs OC1A on Arduino pin 9 and OC1B on pin 10.
- Timer 2: WGM[2:0] = 001 (Mode 1, phase-correct PWM, TOP = 0xFF). COM2A = non-inverting. CS = 001 (no prescaler). Output OC2A on pin 11.

Phase-correct PWM is preferred over fast PWM for motor-control applications because the two timers remain edge-synchronised at the centre of every PWM cycle, which minimises current-ripple beat patterns and acoustic noise. The resulting carrier frequency is:

$$f_{PWM} = \frac{F_{CPU}}{2 \cdot N \cdot TOP} = 16 \times \frac{10^6}{2 \cdot 1 \cdot 255} \approx 31372 Hz \quad (4.1)$$

comfortably above the 20 kHz audible threshold and well within the DRV8302's slew-rate envelope. Duty cycles supplied by the FOC pipeline are converted to 8-bit OCR values via *OCRxA = uint8_t(duty · 255)*. The *init()* routine also leaves the enable pin LOW so that the gate driver cannot be active inadvertently during boot; the *enable()* method asserts it once the FOC software is ready.

4.3. Quadrature Encoder Decoder (MyEncoder)

CHANGE-mode interrupts on pins 2 (INT0) and 3 (INT1) trigger on every edge of channels A and B respectively, giving 4× decoding. Inside each ISR both channel states are read, and a 32-bit signed pulse counter is updated. The direction-decoding logic is derived as follows. After any edge in the canonical forward sequence (A, B) = 00 → 10 → 11 → 01 → 00 - and only that sequence - the post-edge state of (A, B) for an A-channel transition is either (1, 0) or (0, 1); in both cases A ≠ B. For the reverse sequence the corresponding post-edge states are (0, 0) and (1, 1); in both cases A = B. The dual analysis applies to B-channel transitions. The full logic table is:

Table 4.1: Quadrature decoding logic.

Edge	Post-edge (A,B)	Direction	Decoder rule
A rising	10	Forward	A ≠ B → +1
A falling	01	Forward	A ≠ B → +1
B rising	11	Forward	A = B → +1

Edge	Post-edge (A,B)	Direction	Decoder rule
B falling	00	Forward	A = B → +1
A rising	11	Reverse	A = B → −1
A falling	00	Reverse	A = B → −1
B rising	01	Reverse	A ≠ B → −1
B falling	10	Reverse	A ≠ B → −1

Reading a 32-bit long counter from the main loop while interrupts may update it requires a short critical section to avoid a torn read; this is implemented as *noInterrupts()/interrupts()* bracketing a one-line load. Velocity is computed from the finite difference of the counter divided by the elapsed time in *micros()* between calls. The raw velocity is noisy; a *MyLPF* instance with Tf = 10 ms is used in the motor object to smooth it before feeding the velocity PI.

4.4.Current Sensing (MyCurrentSense3PWM)

The DRV8302's two internal current-sense amplifiers are biased to $\frac{V_{CC}}{2}$ and have a transconductance set by the GAIN strap [42]. With the default 10× gain and the 5 mΩ shunts on the reference breakout, the sensitivity is 0.05 V/A; the ADC step at $V_{ref} = 5V$ is 4.88 mV, giving a digital resolution of 10.24 counts per ampere. Three implementation details are worth highlighting.

4.4.1. Zero calibration

The amplifier output is biased to $\frac{V_{CC}}{2} \approx 1.65V$ at zero current - here V_{CC} is the sense amplifiers' 3.3 V analog reference, and the as-built bench confirmed this bias as roughly 344 ADC counts (Section 5.5). Because it also depends on the breakout's SLEEP/EN_GATE wiring and on amplifier and resistor tolerances, the offset is measured rather than assumed. *calibrateZero()* averages 300 ADC readings (≈ 200 µs each at the boosted ADC clock) per channel with the driver enabled at 50% duty - which produces zero net phase voltage and hence zero current - and stores the result as *offset_A_counts, offset_B_counts*. Subsequent current measurements subtract this offset before scaling.

4.4.2. Reconstruction of i_c

Phase C is not sensed directly. For a star-connected motor with a floating neutral the sum $i_a + i_b + i_c = 0$ holds at all times (no charge can accumulate in the neutral node), so $i_c = -i_a - i_b$. This relationship is for the balanced case and is used throughout the firmware.

4.4.3. ADC clock acceleration

The default Arduino ADC prescaler is 128, giving a 125 kHz ADC clock and approximately 104 µs per 10-bit conversion. For the inductance step-response test (Section 4.8.2), which must resolve a time constant $\frac{L}{R}$ on the order of 40 µs, this is far too slow. The

MyCurrentSense3PWM init() method reprograms the ADCSRA prescaler bits to 100 (prescaler = 16), producing a 1 MHz ADC clock and approximately 13 µs per conversion:

```
// ADCSRA = ADEN ADSC ADATE ADIF ADIE ADPS2 ADPS1 ADPS0
// Set ADPS[2:0] = 100 → prescaler 16 → 1 MHz ADC clock
ADCSRA = (ADCSRA & 0xF8) | 0x04;
```

This rate is above the ADC's nominal 200 kHz maximum-for-full-accuracy specification, but the empirical accuracy loss is approximately 1 LSB, which is well below the PWM-induced current ripple and is in any case averaged out by the PI inner loop and by the offset-calibration averaging.

4.5. PID and Low-Pass Filter (MyPID, MyLPF)

Both controllers measure their own sample time using *micros()* between successive invocations, so they automatically adapt to any jitter in the main loop. The PID equation discretization is:

$$e[k] = setpoint - measurement \tag{4.2}$$

$$integral \leftarrow integral + K_i \cdot T_s \cdot 0.5 \cdot (e[k] + e[k-1]), (trapezoidal) \tag{4.3}$$

$$derivative = K_d \cdot \frac{e[k]-e[k-1]}{T_s} \tag{4.4}$$

$$u[k] = K_p \cdot e[k] + integral + derivative \tag{4.5}$$

If *u[k]* exceeds the configured ±limit, the integrator is only updated in the direction that reduces saturation (back-calculation anti-windup). Optional output ramp limiting, bounded by $ramp \cdot T_s$ per step, is provided for fragile setpoints.

MyLPF implements the canonical first-order filter:

$$y[k] = \alpha \cdot y[k-1] + (1-\alpha) \cdot x[k], with \alpha = \frac{T_f}{T_f + T_s} \tag{4.6}$$

Special-case $T_f \leq 0$ disables the filter (pass-through), allowing filtering to be disabled cleanly during an experiment.

4.6. Motor Object and FOC Pipeline (MyBLDCMotor)

The motor class encapsulates the entire FOC state machine. Public members expose pole-pair count, the four identified parameters, every safety limit, control mode, target, all four control-loop instances, the impedance parameters and the runtime state (angles, currents, velocity). Pointer members link to the encoder, driver, optional current sensor and optional monitoring stream. The principal methods are *init(), initFOC(), loopFOC(), move()* and *setPhaseVoltage()*.

Inside *loopFOC()*, if a current sensor is attached, *readCurrents()* populates the abc, *αβ* and *dq* current arrays; *applyCurrentLoop()* runs *PI_id* and *PI_iq*, adds the optional decoupling, saturates against *voltage_limit*, and stores the results in *voltage_d* and *voltage_q*. *setPhaseVoltage(voltage_q, voltage_d, electrical_angle)* performs the inverse Park, inverse Clarke and SVPWM in that order and writes the PWM duty cycles to the driver. The total inner-loop execution time is approximately 700 µs on the UNO, dominated by the four trigonometric LUT calls (two in Park, two in inverse Park) and the two ADC conversions.

4.7.Sensor Alignment (*alignSensor*)

The alignment procedure determines two unknowns: the direction sense of the encoder (whether positive counter increments correspond to forward or reverse rotation in the convention assumed by the FOC math), and the offset θ_{zero} that maps the encoder reading to the electrical angle of the rotor d-axis.

The procedure consists of four phases:

1. Sweep the electrical angle from 0 to 2π in 200 steps with v_q = *U_align* applied and v_d = *0*. The rotor follows the rotating stator MMF and completes one full electrical revolution, ending at electrical angle 0 (= 2π modulo).
2. Read the mechanical angle θ_m, *zero*.
3. Sweep the electrical angle from 0 to π in 100 steps. Read θ_m, π.
4. Sweep back from π to 0. Read θ_m, *zero*.

Direction is determined from *sign*(θ_m,$\pi - \theta_m$,*0):* if positive, *sensor_direction* = *CW*; if negative, CCW; if essentially zero, alignment failed and the rotor did not move (likely insufficient drive voltage or a stalled mechanical load).

Computing the electrical-angle offset θ_{zero} requires careful geometric reasoning. At the end of step 4, the last commanded value was v_q = *U_align* with electrical angle 0. By the geometric content of the Park transform (Section 2.2.2), applying v_q at electrical angle 0 produces a stator MMF vector at angle $\frac{\pi}{2}$ in the *αβ* plane. The rotor's permanent magnet d-axis settles aligned with this MMF, so the rotor's true electrical angle is $\frac{\pi}{2}$, not 0. The offset must therefore be chosen so that *computeElectricalAngle()* returns $\frac{\pi}{2}$ (matching physical reality) at the mechanical angle θ_m, *zero*:

$$\theta_e = dir \cdot pp \cdot \theta_m - \theta_{\text{zero}} \tag{4.7}$$

$$\theta_{\text{zero}} = dir \cdot pp \cdot \theta_m,\ zero\ - \frac{\pi}{2} \tag{4.8}$$

This $\frac{\pi}{2}$ correction is the source of a subtle defect in earlier drafts of the firmware: if the offset is set to *dir* · *pp* · θ_m, *zero* (without the $\frac{-\pi}{2}$ term), then commanded v_q produces no net torque because the FOC's claimed θ_e is aligned with the stator MMF rather than perpendicular to it.

4.8.Motor Parameter Identification

4.8.1. Resistance

With the rotor open-loop-parked at electrical angle 0 by applying v_d = *U_test* and ramping U over 100 steps to avoid current overshoot, the steady-state d-axis *equation* $v_d = R \cdot i_d + L_d \frac{di_d}{dt} - \omega_e \cdot L_q \cdot i_q$ reduces (since $\omega_e = 0$ and $i_q = 0$) to $v_d = R \cdot i_d$. After 500 ms of settling, the algorithm averages 200 samples of i_d (computed at the known electrical angle $\theta_e = 0$, not at the FOC's reading, so the measurement is decoupled from any residual alignment error) and reports:

$$R_phase = U_test / i_d_avg \tag{4.9}$$

The *U_test = 2* V default produces approximately 0.4 A on a typical 5 Ω gimbal motor, yielding an ADC deviation of about four counts. With 200 averages the noise contribution drops by a factor of $\sqrt{200} \approx 14$, yielding a measurement standard deviation in the milli-ohm range.

4.8.2. Inductance

The d-axis voltage equation in the rotor-stationary, no-back-EMF regime $is\ v_d = R \cdot i_d + L_d \cdot \frac{di_d}{dt}$. A step from 0 to *U_test* produces the first-order response:

$$i_d(t) = \left(\frac{U_{test}}{R}\right) \cdot \left(1 - exp\left(\frac{-t}{\tau}\right)\right),\ \ \tau = \frac{L_d}{R} \tag{4.10}$$

The algorithm parks the rotor with a low-bias v_d *= 0.3 · U_test*, then steps abruptly to *U_test* and polls the ADC as fast as possible (one *analogRead* per loop iteration, ≈ 13 µs at the boosted ADC clock) until $|i_d|$ exceeds *0.632 · (U_test / R)*. At that point the elapsed time equals τ, and:

$$L_d = R_{phase} \cdot \tau \tag{4.11}$$

With the rotor parked at electrical angle 0, the phase-A current equals i_α equals i_d (within a sign), so the algorithm reads phase A only (one ADC) rather than performing the full Clarke + Park (two ADCs plus arithmetic), halving the sample period. The surface-PMSM assumption $L_d \approx L_q$ is adopted; the q-axis can be re-measured by parking the rotor at $\theta_e = \frac{\pi}{2}$ and repeating the procedure, but for the present motor and application the single-axis measurement is sufficient.

4.8.3. Flux Linkage

Once *R* and L_d are populated, the only remaining electrical parameter needed to map q-axis current to torque is the permanent-magnet flux linkage λ_{PM}. *identifyFluxLinkage(*ω_e*)* measures it directly: the rotor is driven at a fixed low electrical speed ω_e under closed-loop velocity control, and the steady q-axis voltage is recorded. In steady state the resistive and inductive drops are known, so the residual is the back-EMF *E* and $\lambda_{PM} = \frac{E}{\omega_e}$ - equivalently $\lambda_{PM} = \frac{v_q - R \cdot i_q}{\omega_e}$ in the surface-PMSM model with i_d regulated to zero.

The torque constant follows as $K_t = 1.5 \cdot p \cdot \lambda_{PM}$, where *p* is the number of pole pairs. For this motor the procedure returns $\lambda_{PM} \approx$ *0.0346 Wb* and $K_t \approx$ *0.570 N·m/A* (Table 5.2), closing the voltage-to-torque mapping $\tau = K_t \cdot i_q$. The complementary step of converting these constants into current-loop gains is the IMC tuning described in Section 2.5.3 and exercised by the identification sketch in Section 5.2.1.

4.9. Impedance Controller

Impedance mode replaces the outer control cascade with the spring–damper law of Section 2.6. At each call to *move()*, the controller computes:

```
pos_dev = shaft_angle - impedance_eq
tau_des = -impedance_K * pos_dev - impedance_B * shaft_velocity
saturate tau_des to ±torque_limit
Kt = (3/2) * pp * lambda_PM
```

```
if Kt < 1e-6:    park motor (refuse to operate without identified Kt)
elif current_sense linked:
    iq_ref = tau_des / Kt
    saturate iq_ref to ±current_limit
    current_q_target = iq_ref
else:           // voltage-mode approximation
    voltage_q = R_phase * (tau_des / Kt)
```

The voltage-mode branch produces only an approximation because it ignores the $\frac{di}{dt}$ and $\omega_e \cdot \lambda_{PM}$ terms; in practice the current-mode branch should always be used in a clinical or experimental setting. The graceful refusal to operate when K_t is unknown protects against accidentally driving the motor under unidentified parameters.

4.10. Impedance-Control Signal Path Through Firmware and Hardware

This section gives the implementation-level realization of the impedance law derived in Section 2.6 and the control architecture of Section 3.3. The signal-path diagram below follows one control cycle from encoder measurement to inverter duty cycles and makes explicit the boundary between the outer rehabilitation law and the inner field-oriented controller.

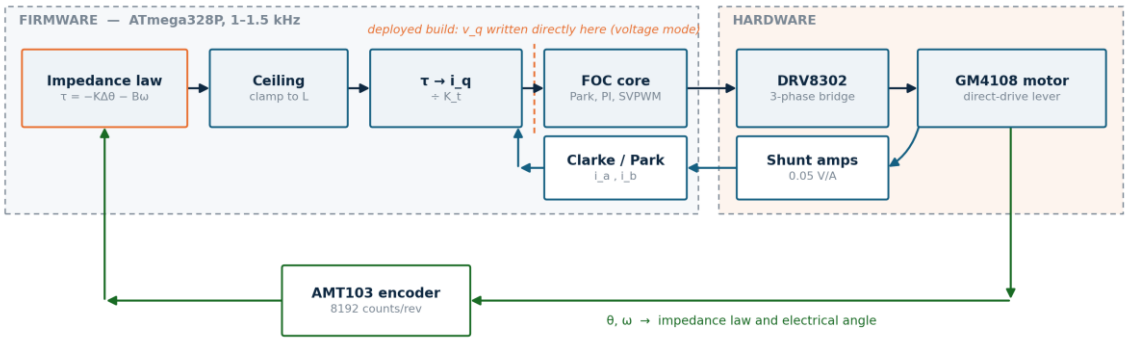


Figure 4.1: Impedance-control signal path. The encoder supplies shaft angle and velocity to the impedance law and the electrical angle to the transforms; the commanded torque is clamped to the ceiling, converted to a current reference through the torque constant, and passed to the field-oriented core. The dashed marker shows where the deployed build departs from this path and writes a q-axis voltage directly.

At each call to *move(),* the encoder provides the mechanical angle θ_m and velocity ω_m. The outer loop forms the position error $e_\theta = \theta_m - \theta_{eq}$ and evaluates the virtual spring-damper law $\tau_{des} = -Ke_\theta - B\omega_m$. The result is bounded by the configured safety ceiling, $\tau_{cmd} = sat(\tau_{des}, \pm\tau_{limit})$. For the surface-PMSM model used here, $K_t = \left(\frac{3}{2}\right) pp\lambda_{PM}$, so the closed-current path uses $i_d^* = 0\ and\ i_q^* = \frac{\tau_{cmd}}{Kt}$. The two measured phase currents are offset-corrected, the third is reconstructed as $i_c = -(i_a + i_b)$, and the Clarke and Park transforms produce the measured i_d and i_q at the encoder-derived electrical angle θ_e.

The current controllers regulate $e_d = i_d^* - i_d$ and $e_q = i_q^* - i_q$, add the optional dq decoupling terms, and send v_d^* and v_q^* through the inverse Park transform and SVPWM to the

DRV8302. The q-axis electrical plant is $v_q = Ri_q + L_q\left(\frac{di_q}{dt}\right) + \omega_e(L_d i_d + \lambda_{PM})$; with i_d approximately zero, electromagnetic torque is $\tau_e = K_t i_q$. The encoder therefore serves two roles: θ_m and ω_m close the mechanical impedance law, while θ_e commutates the current vector. In the deployed Chapter 5 experiments the current-feedback branch was bypassed and the fallback command $v_q^* = R_{phase}\left(\frac{\tau_{cmd}}{K_t}\right), v_d^* = 0$ was used instead. That approximation is valid only near steady state because it omits the inductive and back-EMF terms above; consequently, a constant q-axis voltage does not by itself establish constant q-axis current or constant torque. This distinction is used explicitly in the isotonic validation methodology of Section 5.2.4.

4.11. Data Logging and Monitoring

The motor object exposes *logCSVHeader()* and *logCSV()* methods that emit one comma-separated record per call with the columns: time-ms, target, shaft-angle, shaft-velocity, i_d, i_q, v_d, v_q and computed torque. In the rehab demo a 50 Hz logger writes this record to the serial port so that a Python script can capture and plot the rehabilitation session. The fixed schema simplifies offline analysis.

Chapter 5
Experimental Setup and Results

5.1. Bench Configuration

The hardware bench used for development and validation is configured as follows. The Arduino UNO Rev 3 is socketed onto the DRV8302 [42] breakout board manufactured for the SimpleFOC [3] ecosystem; the breakout provides the gate driver, 5 mΩ low-side shunts, the GAIN strap, screw terminals for the three motor phases and the DC supply, and pin headers that route every necessary signal to the UNO. The 11-pole-pair gimbal motor is connected by three 22 AWG silicone leads to phases A, B and C; the AMT103 encoder [38] is rigidly mounted to the motor's shaft via the custom 3D-printed bracket and connected by a six-conductor ribbon to the UNO.

The bench layout was inspired by the DRV8302 example provided in the SimpleFOC documentation [3].

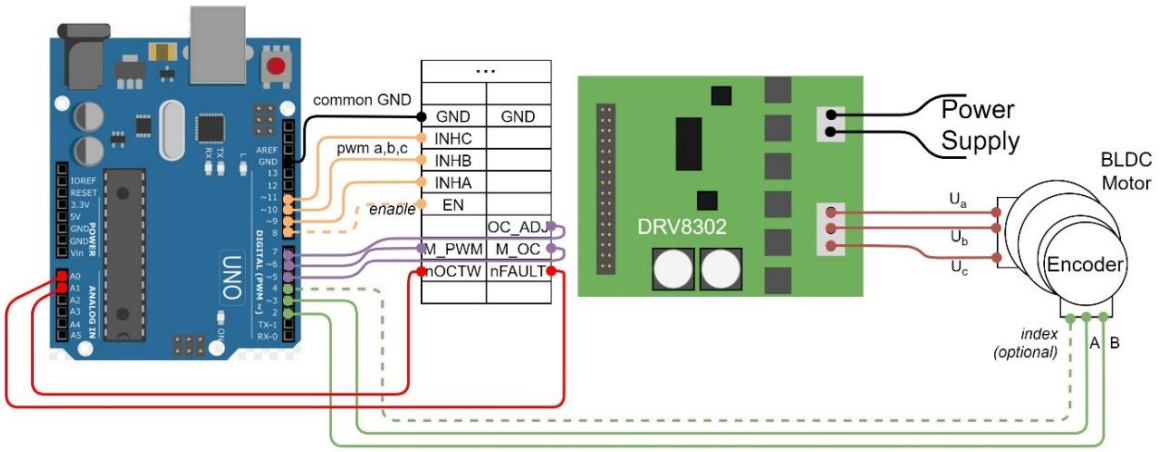


Figure 5.1: Electrical diagram of the setup.

A motor mount was designed in Fusion 360 and 3D printed. It seats the GM4108-120T stator in a Ø47 mm register, retains it with four M3 screws on a Ø38.2 mm bolt circle, carries the encoder bracket, and clamps to the bench through a Ø12 mm post.



Figure 5.2: The 3D-printed motor mount, as built.

The pendulum arm, also designed and 3D printed, is mounted directly on the motor's rotor through four M3 screws on a Ø30 mm bolt circle, giving a direct-drive connection with no intervening transmission. Its lower end carries a Ø4.9 mm cross hole at a radius of 120 mm from the rotor axis to which calibration weights are attached for the torque measurements of Section 5.6. A custom bracket couples the rotating gimbal to the AMT103 encoder. Dimensioned drawings of the mount, the arm and the assembled module are given in Appendix C, Figures C.1 to C.3.



Figure 5.3: The 3D-printed pendulum arm, as built.

The assembled apparatus is shown in Figure 5.4.

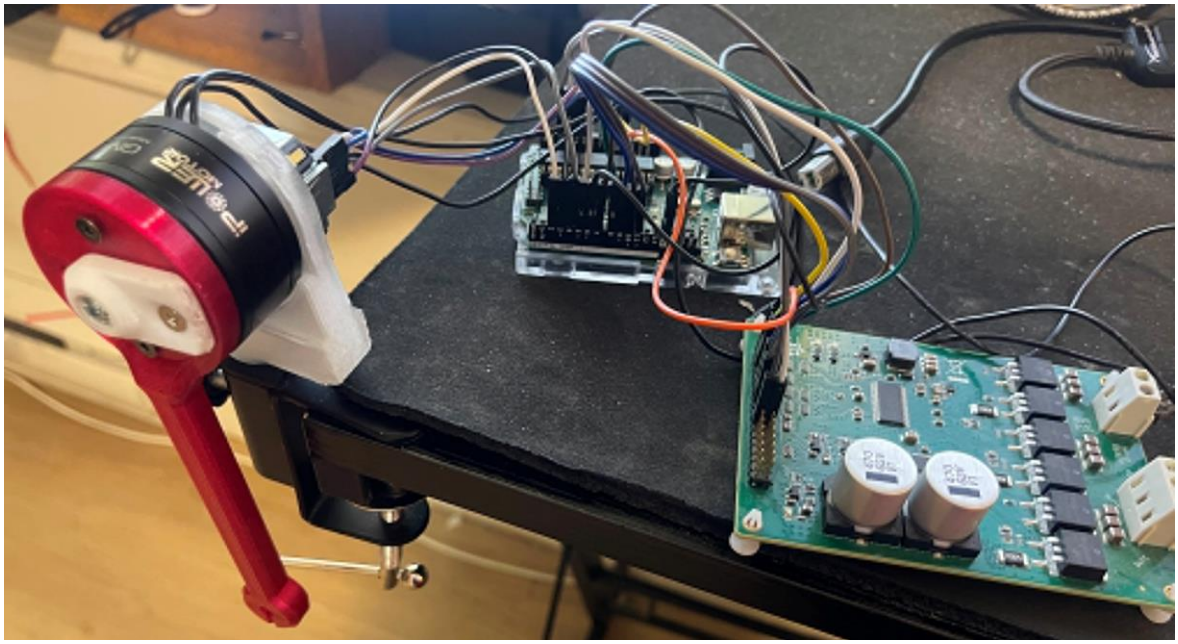


Figure 5.4: The assembled experimental apparatus.



Figure 5.5: Close-up of the gimbal motor with the coupled AMT103 encoder.

The 12 V DC bus is supplied from a bench power supply capable of 5 A; during the reported tests, the observed supply current was typically around 2 A. Supply-side current limiting provides an independent fault-protection layer in addition to the firmware limits and the DRV8302 protection. Both a laboratory bench supply and a 12 V, 5 A switched-mode supply were used because they were readily available and low cost.

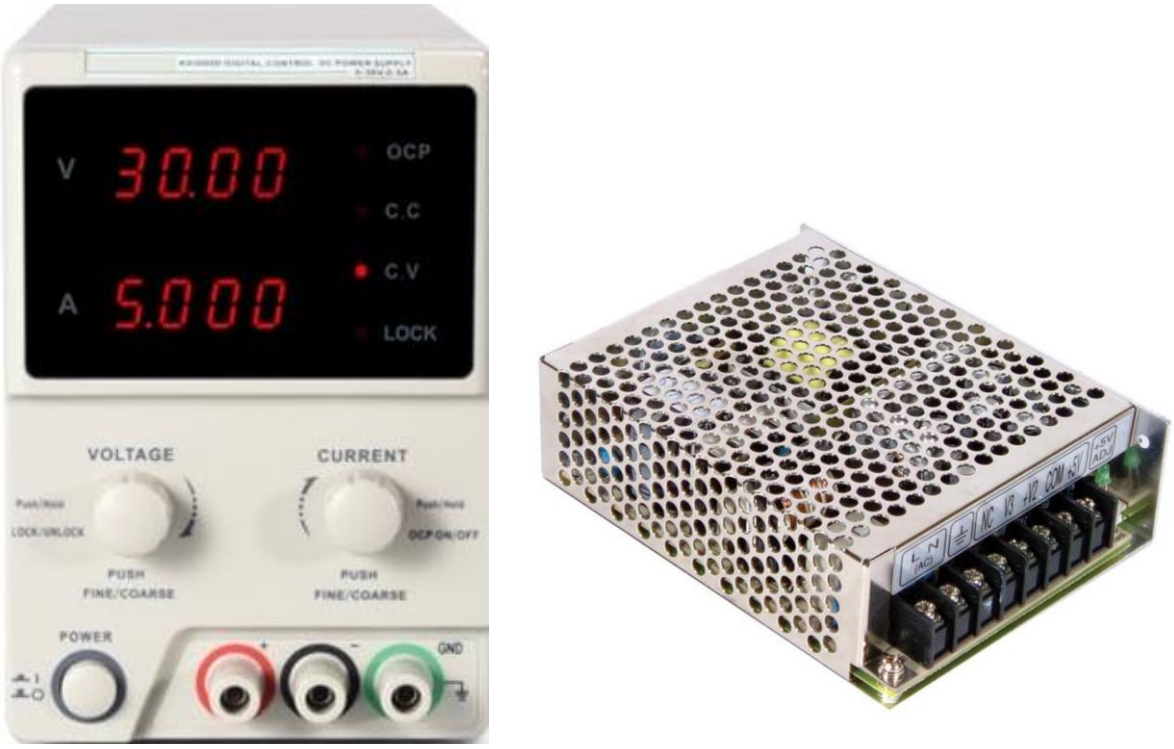


Figure 5.6: Bench power supply and switched-mode supply used for the 12 V DC bus.

All measurements are acquired via the Arduino's built-in USB-serial bridge at 115200 baud, captured by a host laptop running a Python script (pyserial + pandas + matplotlib) that ingests the CSV stream emitted by *logCSV()* and produces the plots described in Chapter 5.

5.2. Test Procedures

Three structured test procedures are defined: parameter identification, open-loop FOC validation, and the clinical exercise-mode sequences used to drive the rehabilitation application.

5.2.1. Parameter Identification Procedure

The parameter-identification procedure was run with the motor mechanically unloaded but free to rotate. It:

1. Initialises the encoder and driver.
2. Calibrates the current-sense zero offsets with the driver enabled at 50 % duty (300 samples per channel), producing zero net phase voltage while keeping the DRV8302 sense amplifiers awake.
3. Enables the driver and runs *initFOC()* to align the sensor and verify the pole-pair count.
4. Pauses 500 ms to let the rotor settle.
5. Runs *identifyResistance()*, printing the measured R_phase.
6. Runs *identifyInductance()*, printing τ and L_d.
7. Runs *tuneCurrentLoops(800 rad/s)* to set the IMC gains.
8. Runs *identifyFluxLinkage(10 rad/s)*, spinning the motor briefly under closed-loop velocity control, printing λ_{PM} and K_t.
9. Prints a summary block and a copy-paste const-float block ready for the application sketch.

10. Disables the driver.

A successful identification yields all four parameters within the expected ranges for a small gimbal motor ($R \approx 1–15\ \Omega$, $L \approx 50–500\ \mu H$, $\lambda_{PM} \approx 0.005–0.05\ Wb$). Outlier values almost always indicate hardware miswiring or insufficient supply current, both of which are diagnosed by inspecting the per-step serial output.

5.2.2. FOC Validation Procedure

A minimal voltage-mode FOC sketch, developed during bring-up for direct comparison with SimpleFOC, exposed the M / C / S serial command set; the following sequence was executed:

1. C0 (torque mode); M1.0 (1.0 V q-axis command). Observe that the motor begins to spin smoothly and accelerates until the available electromagnetic torque is balanced by speed-dependent losses.
2. C1 (velocity mode); M5.0 (5 rad/s setpoint). Observe steady-state velocity tracking. Apply a manual load on the shaft; observe that the controller responds by raising v_q to maintain velocity.
3. C2 (angle mode); M0 (zero-angle setpoint). Manually perturb the shaft and observe spring-like recovery to the setpoint.

These three checks reproduce the canonical behaviours of any working FOC implementation. The procedure replicates the corresponding SimpleFOC reference example, which confirms that MyFOC behaves as intended and screens for the defects most commonly encountered at bring-up: reversed encoder direction, an incorrect alignment offset, inverted current-sense polarity and unsuitable controller gains.

5.2.3. Clinical Exercise-Mode Procedure

The rehabilitation application is driven by a dedicated demonstration sketch whose serial interface exposes the impedance controller as a set of single-letter commands, so an exercise mode can be selected and parameterised at run time without re-flashing. The mode is chosen with M0–M3 (impedance, velocity, angle or torque); K, B, L and A set the virtual stiffness, damping, torque ceiling and adaptive gain, respectively; I<τ> commands a constant torque and selects torque mode; V<ω> sets a velocity target and selects velocity mode; W<grams> records an applied weight into the log; and T<n> tags subsequent samples as belonging to test n. The controller boots in a passive state (K = B = A = 0,v_q= 0) so the arm is free until a mode is commanded.

The five clinical modes presented in Section 5.8 were each run from this interface. The isometric test (T1) selects impedance mode with a stiff spring and a raised ceiling (M0, K = 1.0 N·m/rad, L = 0.30 N·m), after which calibrated weights are hung and logged with W while the arm is held; the isotonic test (T2) issues stepped I<τ> commands from 0.10 to 0.30 N·m, selecting torque mode with K = 0, while a fixed 200 g load hangs from the lever; the isokinetic test (T3) toggles the velocity set-point (V +2, V 0, V −2 rad/s in velocity mode); the adaptive test (T4) enables effort scaling on the gentle deployed baseline (M0, K = 0.5 N·m/rad, L = 0.05 N·m, A = 2.0); and the transparent back-drive test (T5) is run from the passive boot state (v_q = 0) with the small breakaway weights logged through W. Each run is tagged with the matching T command so the modes can be separated in the logged CSV.

5.2.4. Relation to Clinical Testing Practice

The five exercise modes validated in Section 5.8 take their names from clinical exercise physiology, and it is important to state how the bench procedures relate to the way those modes are delivered and measured in practice. In two of the five, the bench arrangement differs materially from clinical practice, and in both cases the difference changes what the measurement is able to establish.

Isometric testing is the closest match. In clinical dynamometry the joint is fixed at a defined angle and the patient contracts against an immovable restraint while a load cell records the developed torque [30]. The bench procedure of Section 5.8 reproduces this arrangement with the reversed roles: the controller holds the lever stationary with a stiff virtual spring, and calibrated weights rather than a muscle apply the load. Because the arm does not move and the applied torque is known independently from the hung mass and the lever radius, this is the one bench test that yields a measurement traceable to a physical standard, which is why it is the basis of the calibration reported in Section 5.8.

Isotonic testing differs, and the difference was raised during review. The defining property of isotonic exercise is a resistance that does not change as the joint moves through its range [57]. Clinically this is achieved with a cable column: a weight stack is connected to the limb through a cable and one or more pulleys, so the tendon force is constant in magnitude, and the pulley geometry keeps its line of action approximately perpendicular to the limb throughout the range. The resisting torque is then the product of a constant force and an approximately constant moment arm and is therefore itself approximately constant. The bench instead hangs a fixed 200 g mass directly from the end of the lever, so the resisting torque is $m \cdot g \cdot r \cdot \sin\varphi$ and varies with the sine of the lever angle, falling to zero when the lever is vertical. Figure 5.7 contrasts the two arrangements.

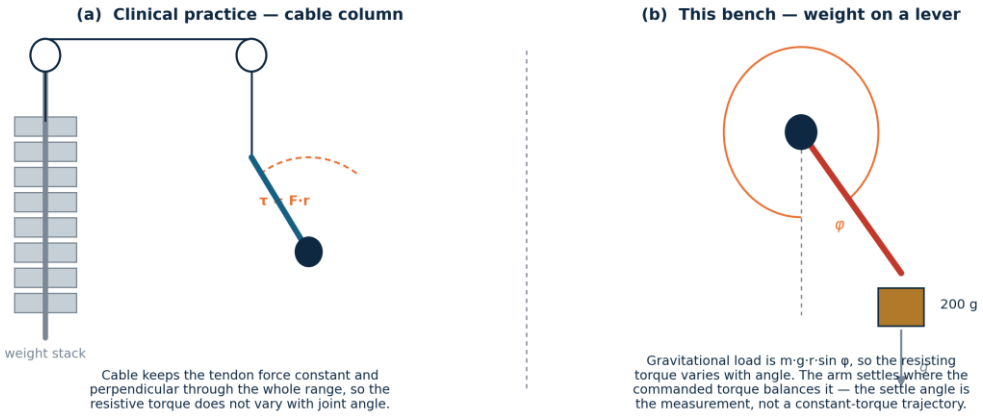


Figure 5.7: Isotonic loading. (a) Clinical practice [57]: a cable and pulley keep the applied force constant and its moment arm approximately constant, so the resisting torque does not vary with joint angle. (b) The arrangement used here: a mass hung directly from the lever produces a resisting torque that varies as sinφ, so the lever settles at the angle where the commanded torque balances it.

The consequence for the result is specific. Because the load is angle-dependent, the bench cannot demonstrate constant resistance through a range of motion directly; what it demonstrates instead is that the commanded torque is constant, and that the lever settles at the

angle at which the gravitational load balances it. The measured quantity is therefore the settling angle rather than a torque trajectory, which is why Section 5.8 reports settle angle against commanded torque. This verifies that the controller holds a constant commanded torque, which is the firmware property under test, but it does not verify constant resistance as a clinician would understand the term. Replicating the clinical arrangement would require either a cable and pulley reproducing a constant moment arm, or — more appropriate to a robotic device — a gravity-compensation term added to the commanded torque so that the net resistance felt at the handle is constant while the load itself is not. The latter is a firmware change of a few lines and is the recommended correction.

To ascertain constant motor-torque operation electrically, the quantity to monitor is q-axis current i_q, not q-axis voltage. For the surface PMSM with i_d approximately zero, $\tau_e = K_t i_q = \left(\frac{3}{2}\right) pp\lambda_{PM} i_q$; therefore, constant i_q is the direct electrical signature of constant electromagnetic torque. On this bench an oscilloscope can be connected to the DRV8302 SO_1 and SO_2 current-sense amplifier outputs [42]. After removing their calibrated zero-current bias, the phase currents are proportional to $\frac{V_{SO} - V_{zero}}{GR_{shunt}}$. A constant sinusoidal phase-current amplitude while the rotor moves, together with correct commutation, is the scope-observable proxy for a constant current-vector magnitude; the strongest check is to reconstruct i_q with the encoder-derived electrical angle using the same Clarke/Park transformation as the controller and confirm that i_q remains flat throughout the motion.

This distinction is important for the present data. The isotonic run was recorded in voltage mode with the current loop open, so a flat commanded v_q trace or a flat torque value reconstructed from that same voltage cannot guarantee constant physical torque during motion; the inductive and back-EMF terms vary with current dynamics and speed. The present test therefore establishes a constant torque command and the corresponding equilibrium response, not independently verified isotonic torque through the full range of motion. A repeat intended to establish isotonic operation should link the current sensors, close the q-axis current loop, record i_q while the arm moves, and, for clinical-grade validation, compare the result with an independent in-line torque transducer.

Isokinetic testing differs in a second and more consequential way. A clinical isokinetic dynamometer imposes a fixed angular velocity that the limb cannot exceed, and the measurement of interest is the torque the patient produces against that constraint, recorded by a calibrated in-line transducer through the range [30]. The bench reproduces the kinematic half of this — a velocity set-point of ±2 rad/s, held by the velocity loop — but not the measurement half, because there is no torque transducer and the arm is unloaded apart from its own weight. Figure 5.8 shows the two arrangements.

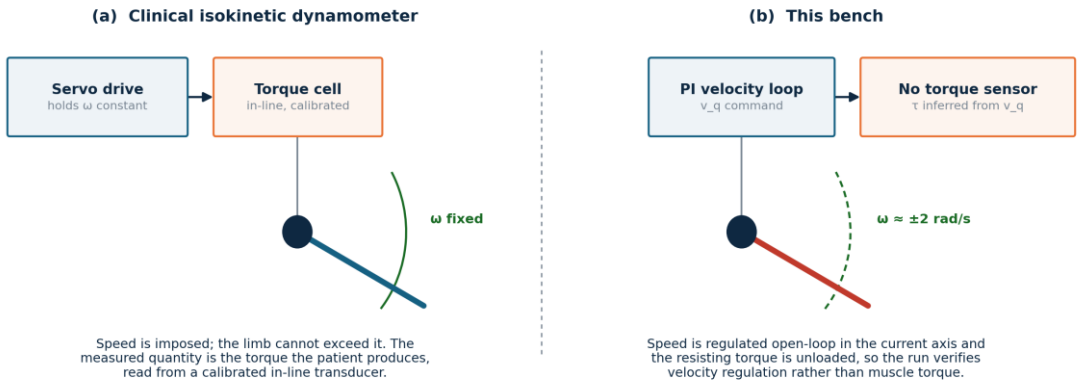


Figure 5.8: Isokinetic testing. (a) Clinical practice: the servo imposes the angular velocity, and a calibrated in-line torque transducer records the torque the patient develops against it. (b) The arrangement used here: the velocity set-point is held by the velocity loop, but with no transducer and no resisting load the run verifies velocity regulation rather than developed torque.

The bench run therefore verifies that the velocity loop tracks a set-point, which is a firmware property, and not that the device can quantify muscle torque at constant speed, which is the clinical property. Reported means of +2.20 and −2.02 rad/s establish the former; the latter is out of reach until a torque transducer is fitted, and the report should not be read as claiming it. A closer replication within the present hardware would apply a known resisting load — the calibrated weights already used in the isometric test — while the velocity loop holds speed and would record the q-axis command required to maintain it; that would at least exercise the torque-versus-speed relationship, albeit with the same 1.6× absolute-scale uncertainty documented in Section 5.8.

The adaptive-resistance and transparent back-drive modes have no direct clinical dynamometry equivalent. The adaptive mode derives from the assist-as-needed literature [28], [29], where the quantity of interest is the assistance withdrawn as performance improves rather than a torque reading, and the bench procedure measures exactly that law. The transparent mode is measured here as a mechanical property of the de-energized device — damping ratio from a free ring-down and static breakaway torque from small, applied masses — which is the same characterization used for back-drivability in the exoskeleton literature [24], and requires no clinical analogue.

5.3. Data Logging and Plotting Pipeline

A short Python script reads the serial port, identifies the CSV header emitted by motor.logCSVHeader(), accumulates subsequent rows into a pandas DataFrame, and produces the plots described in Chapter 5. The script is reproducible across machines and platforms and forms part of the project's measurement infrastructure.

5.4. Compile-Time Verification

Before any hardware deployment the library is verified to compile cleanly with avr-g++ targeting the ATmega328P. The full compile command and result are:

```
avr-g++ -c -mmcu=atmega328p -O2 -Wall -Wextra \
        -fno-exceptions -std=c++11 \
```

```
        -I MyFOC/src MyFOC/src/*.cpp
```

All C++ source files compile without warnings. The same verification is repeated for the application sketch *Rehab_Device*, which compiles cleanly. The full library footprint on AVR is approximately 21 KB of program memory and zero static SRAM beyond per-object instance state, leaving over 10 KB of flash headroom for application code.

5.5. Hardware Bring-Up and Auto-Detected Parameters

A bring-up diagnostic routine was used for first power-on of the motor. It executes a three-phase test: (i) baseline ADC scan with the gate driver disabled; (ii) ADC re-scan with the driver enabled at 50 % duty cycle, which produces zero phase-to-phase voltage and therefore zero current but wakes the chip from its standby state; (iii) a slow rotor sweep through eleven electrical revolutions to count the encoder pulses per mechanical revolution. The empirically observed results were:

Table 5.1: Diagnostic outputs from the as-built bench.

Quantity	Measured value
Encoder pulses per mechanical revolution (auto-detected)	8379
Suggested encoder CPR setting	8192
Pole-pair check estimate after CPR correction	11.003 (target 11)
Current-sense offset, channel A (driver enabled, no current)	344.1 ADC counts
Current-sense offset, channel B (driver enabled, no current)	345.4 ADC counts
Current-sense offset (driver disabled)	1.1 / 1.0 ADC counts
nFAULT / nOCTW status during normal operation	HIGH / HIGH

Two observations from this table guided subsequent design adjustments. First, the auto-detected encoder count of 8379 per revolution matches the AMT103 set to 2048 pulses per revolution (which yields 8192 counts after 4× quadrature decoding); the firmware was duly configured for CPR = 8192. Second, the current-sense amplifier outputs read at the floor of the ADC range (≈ 1 count) when the gate driver was disabled but woke to 344 counts ($\frac{344}{1023} \cdot 5\,\mathrm{V} \approx 1.68\,\mathrm{V}$, corresponding to 0.336 of the ADC full scale and close to the expected 1.65 V half-supply bias) the moment the driver was enabled. This confirmed that the DRV8302 breakout used in this work ties the chip's internal SLEEP signal to EN_GATE, so the current-sense amplifiers are only operational while the driver is enabled. The current-sense

calibration was therefore restructured to run with the driver enabled at 50 % duty (producing zero net phase voltage and therefore zero current) so that the captured offsets correspond to the operating bias point.

5.6. Identified Motor Parameters

After the diagnostic step, the parameter-identification procedure was run with the motor mechanically unloaded but free to rotate. The full identification sequence (resistance step, inductance step, IMC current-loop auto-tune, back-EMF spin) completed in approximately twelve seconds and produced the values shown in Table 5.2.

Table 5.2: Empirical motor parameters from the identification procedure.

Quantity	**Identified value**
Sensor direction (after alignment)	CCW
Zero electrical angle (after π/2 correction)	5.83 rad (alignment-position dependent)
Pole-pair check, empirical estimate	11.05 (target 11, error 0.5 %)
Stator phase resistance R	11.16 Ω
d-axis inductanceL_d (step response)	7.46 mH
Time constant $\tau = \frac{L}{R}$	0.668 ms
Permanent-magnet flux linkage λ_{PM}	0.0346 Wb
Torque constant $K_t = \frac{3}{2} \cdot pp \cdot \lambda_{PM}$	0.570 N·m/A

The torque constant $K_t = 0.570$ N·m/A and the flux linkage $\lambda_{PM} = 0.0346$ Wb fall well within the expected range for an eleven-pole-pair gimbal motor and provide a reliable basis for the impedance control mode of the rehabilitation application. The resistance R = 11.16 Ω sits within the typical gimbal-motor range and is internally consistent with the steady-state Kirchhoff balance during the back-EMF test, lending it high confidence. The values in Table 5.2 are those of the second of two identification runs: an initial run, taken before a direction-sign error in the velocity loop was corrected, spun the rotor close to its terminal speed during the back-EMF spin and substantially under-estimated the flux linkage; after the fix the procedure was re-run, and the corrected values reported here were adopted for the firmware and used for every result in this chapter.

The d-axis inductance is the one identified parameter that should not be accepted at face value. The step-response routine returned L_d = 7.46 mH (time constant 0.668 ms), whereas a

small gimbal motor of this class is expected to be roughly 100-500 µH. When the 7.46 mH value was used for IMC auto-tuning, the closed current loop entered a sustained limit cycle even at zero commanded torque, confirming that the estimate is unsuitable for controller tuning. The value is therefore treated as a measurement artifact rather than a motor property. The measurement-budget analysis and corrective options are given immediately below. For the rehabilitation results, the d- and q-axis current loops remained inactive and the motor was operated in voltage mode; conservative fixed gains (Kp = 0.05 V/A, Ki = 50 V/(A s)) are retained only as a bring-up starting point.

5.6.1. Improving the Inductance Measurement

The 7.46 mH value reported immediately above is not physically credible for this motor class: it is more than an order of magnitude above the approximately 100-500 µH range expected for a small gimbal motor. The step-response algorithm of Section 4.8.2 is not the principal problem; the limiting factor is the measurement budget. At the 2 V test level the winding draws about 0.18 A, which at 10.24 ADC counts per ampere produces only a 1.8-count excursion, so the 63.2% threshold lies roughly one count above the calibrated zero. At the same time, a plausible electrical time constant of about 10-45 microseconds spans only one to three of the 13 microsecond ADC conversions available after clock acceleration. The transient is therefore neither large enough in amplitude nor long enough in time to be estimated reliably.

Four practical remedies are available, ordered below by the constraint they remove, and the hardware change they require.

Raising the current sense gain changes nothing but a strap. The DRV8302 provides a selectable amplifier gain, and moving from the ×10 setting used here to ×40 raises the sensitivity from 0.05 V/A to 0.2 V/A and the resolution from 10.24 to 41 ADC counts per ampere [42]. The same 0.18 A test current then produces a 7.4-count excursion instead of 1.8. This is the cheapest available improvement and should be attempted first; its limit is that the amplifier output must still span the converter range to the full 0.6 A operating current, which at ×40 it does.

Oversampling and decimation trades sample rate for resolution in firmware alone. Averaging 4^n successive conversions and right-shifting by n recovers n additional bits, provided the signal carries enough noise to dither the converter, which the PWM-induced current ripple guarantees [44]. Two extra bits require sixteen conversions and therefore about 208 µs, which is far longer than the transient being measured, so oversampling cannot be applied to the step response directly. It is nonetheless useful for the resistance and flux-linkage measurements, both of which are steady-state.

High-frequency signal injection removes both constraints at once and is the technique this project should adopt. Rather than timing a transient, a small sinusoidal voltage at a frequency well above the current-loop bandwidth is superimposed on the d-axis, and the amplitude and phase of the resulting current are measured in the steady state. At an injection frequency ω_i chosen so that $\omega_i \cdot L$ is comparable to or larger than R, the magnitude of the impedance gives $\sqrt{R^2 + (\omega_i L)^2}$ and hence L, and the measurement can be averaged over many cycles because it is periodic rather than transient. Injection at a few hundred hertz to a few kilohertz is standard practice in sensorless drives, where the same excitation is used to extract rotor

position from saliency [45], [17]; here only the impedance magnitude is required, which is a considerably weaker demand. The method is implementable on the existing hardware because it needs no additional sensing, only a modulated d-axis reference and a synchronous detector, and it is insensitive to the converter resolution because averaging over N cycles reduces the noise contribution by $\sqrt{N}$.

Direct measurement with laboratory instrumentation remains the reference against which any of the above should be checked. An LCR meter or impedance analyzer applied to two motor terminals with the rotor locked returns the line-to-line inductance at a chosen frequency, from which the per-phase value follows; the rotor must be locked because the measurement is otherwise corrupted by motion, and the reading should be taken at several rotor positions to expose any saliency. This falls outside the no-auxiliary-instruments constraint adopted in Section 1.4, which is precisely why it was not used, but it is the correct way to establish ground truth before trusting an on-board method.

Table 5.3: Options for improving the inductance identification, with the constraint each removes.

Method	Constraint removed	Hardware change	Expected outcome
×40 current-sense gain	Amplitude quantisation	Strap only	41 counts/A; 7.4-count step
Oversampling and decimation	Amplitude quantisation	None (firmware)	Steady-state parameters only — too slow for a step
High-frequency injection	Both amplitude and time	None (firmware)	Steady-state impedance; averages over many cycles
LCR meter, locked rotor	Both	External instrument	Ground truth; outside the platform constraint
32-bit processor, 12-bit ADC	Time, and part of amplitude	New MCU	164 counts/A with ×40; 30-count step

The recommended sequence is therefore to set the DRV8302 current-sense amplifier to ×40, repeat the existing step test to confirm the expected improvement, verify the result once with a locked-rotor LCR-meter measurement, and then implement high-frequency injection as the permanent on-board method. High-frequency injection is implementable on the present hardware because it estimates steady-state impedance over many cycles; a later 32-bit/12-bit processor port would improve timing and ADC headroom but is not a prerequisite. None of these remedies require a torque sensor.

5.7. Functional Verification of the FOC Pipeline

Each algorithmic stage was verified independently during bring-up:

- Mathematical kernel. Clarke, Park, inverse Park, inverse Clarke and SVPWM were unit-tested in a host environment against analytic expressions. The lookup-based sin/cos approximation produces a maximum error below 1×10^{-4} relative to the host math library across $[0, 2\pi)$.

- PWM output. An oscilloscope on each of the three INH inputs with the driver enabled at fixed 50 % duty confirmed a carrier frequency of 31.4 kHz ± 1 % and centre-aligned (phase-correct) waveforms across the two timers.
- Encoder decoding. Manual rotation through one full mechanical revolution incremented the pulse counter by 8192 counts (= 4 × 2048), confirming the configured CPR. The auto-detect routine in the diagnostic procedure yielded 8379 counts in a slow motor-driven sweep - the small shortfall from 8192 (about 2 %) is attributable to the rotor not always tracking the stator field with zero slip during the slow open-loop sweep.
- Sensor alignment. After *initFOC()*, the serial monitors reported *sensor_direction = CCW* and an empirical pole-pair estimate of 11.05 (target 11), confirming both the directional mapping and the $\frac{\pi}{2}$ offset correction applied during the alignment procedure. The reported zero-elec-angle differed between successive boots according to the arbitrary mechanical position at which the alignment ramp terminated; this is the expected behaviour.

5.8. Application: Robotic Rehabilitation Devices

After making sure all of the preliminary procedures were successfully run, the setup is ready to be used to explore the rehabilitation application. A scoping note applies to all of the application results that follow: MyFOC's closed-loop current path - the *d- and q-axis* current PI loops, the IMC tuning of Section 2.5, and the cross-axis decoupling feed-forward - is implemented and was exercised on the bench, but it was not used to collect the application data. The rehabilitation recordings in this section were taken with the library running in its voltage-mode commutation path, in which the commanded q-axis voltage is applied at the measured rotor electrical angle through the inverse Park and space-vector PWM stages with no current feedback (the low-side current sensing was not linked for these runs). The figures and statistics that follow therefore characterise the impedance and force-rendering behaviour under voltage-mode commutation rather than under closed-loop current regulation; the identified inductance that drives the current-loop tuning, and the caution it warrants, are discussed in Section 5.6.

5.8.1. Clinical Exercise Modes: Five-Mode Validation

The rehabilitation application was validated by operating the device in five clinically recognised exercise modes, each obtained from the same impedance kernel by a different choice of controller parameters and selected at run time over the serial link without re-flashing. Rather than exposing raw stiffness and damping numbers, the controller presents the patient with a recognisable exercise; the five modes — isometric, isotonic, isokinetic, adaptive-resistance and transparent (back-drive) — were each exercised on the bench and are described below with a schematic of the physical configuration and the corresponding recorded behaviour. All five runs used the same loaded pendulum arm (lever radius r = 0.12 m from the shaft axis to the weight) and were recorded in the voltage-mode commutation path described above. Over the serial link the commands *K, B, L* and *A* set the impedance parameters, *I*<τ> commands a constant torque (and selects torque mode), *V*<ω> a velocity set-point (velocity mode), *M0–3* select impedance / velocity / angle / torque mode, *W*<*grams*> logs an applied weight into the CSV stream and *T*<*n*> tags the active mode; the device boots passive with *K = B = A = 0*.

Isometric — sustained hold against load (Test 1). In the isometric mode the controller renders a stiff virtual spring about the start position (impedance mode, *K = 1.0 N·m/rad*, ceiling raised to *L = 0.30 N·m*) so the lever is held essentially stationary while calibrated weights of 50, 100, 150 and 200 g are hung from its end (Figure 5.9). Because the arm does not move, the holding q-axis voltage needed to oppose gravity is a direct, repeatable measure of the torque the device is producing, and the hung weight provides a parameter-free ground-truth for that torque.

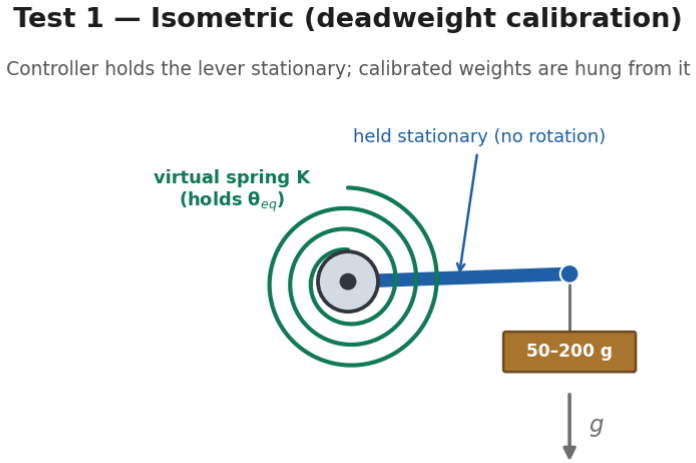


Figure 5.9: Setup for the isometric test (Test 1). The controller holds the lever level with a stiff virtual spring while calibrated weights are hung; the holding voltage and the known weight together give a torque measurement that is independent of the commanded control law.

Figure 5.10(a) shows the holding voltage rising in clean steps as each weight is added. Panel (b) is the key calibration: the true gravitational torque at the held angle, *m·g·r·cos(φ),* is plotted against the magnitude of the commanded q-axis voltage. The three unsaturated points (50–150 g) fall on a straight line of slope *0.032 N·m/V*, whereas the voltage-to-torque reconstruction used throughout this chapter, $\tau = v_q \cdot (\frac{K_t}{R})$ with $\frac{K_t}{R} = 0.0511$ N·m/V, is about 1.6× steeper. Each of the three points lies below the reconstruction, the individual over-prediction ratios scattering between roughly 1.3× and 2.1×. A through-origin regression of the three points places the deadweight slope at 0.032 N·m/V (95% confidence interval 0.029–0.036) and the over-prediction factor at 1.6 (95% confidence interval 1.4–1.7); the interval is wide because only three load levels were available, but the sign of the error — the open-loop voltage-mode reconstruction over-predicts the physically delivered torque at every point — is unambiguous. The 200 g point is excluded: at that load the lever drooped about 40° and the controller reached its torque ceiling, so it is saturated rather than a calibration point. This deadweight measurement is the only result in the chapter that is independent of the commanded law — the other four modes reconstruct torque from the same voltage they command, so they verify rendering fidelity but cannot detect an absolute-scale error of this kind. The discrepancy is precisely the consequence of operating open-loop in voltage mode, and closing the current loop (Chapter 6) is what would remove it.

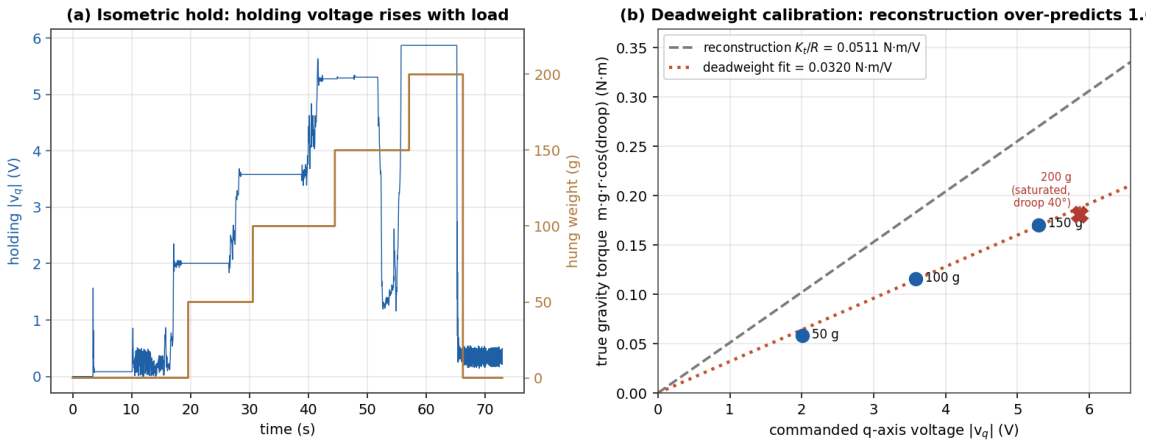


Figure 5.10: Isometric test (Test 1). (a) Holding q-axis voltage steps up as weight is added. (b) Deadweight calibration: the true gravitational torque (50–150 g) follows a 0.032 N·m/V line, about 1.6× below the voltage-mode reconstruction K_t/R = 0.0511 N·m/V; the 200 g point is saturated (about 40° droop) and excluded.

Isotonic — commanded constant resistive torque (Test 2). Sending *I*<*τ*> places the controller in torque mode with the position spring disabled (*K = 0*) and applies a fixed q-axis torque command. A fixed 200 g weight hangs from the lever, which is free to rotate until the gravitational torque of that load balances the motor command. Stepping the command from approximately 0.10 to 0.30 N·m and allowing the arm to settle at each level tests the repeatability of the constant command and the resulting load-dependent equilibrium. Because the gravitational load varies with lever angle and these data were collected in voltage mode without current feedback, this experiment does not by itself demonstrate constant physical resistance throughout the range of motion; the required current-based verification is defined in Section 5.2.4.

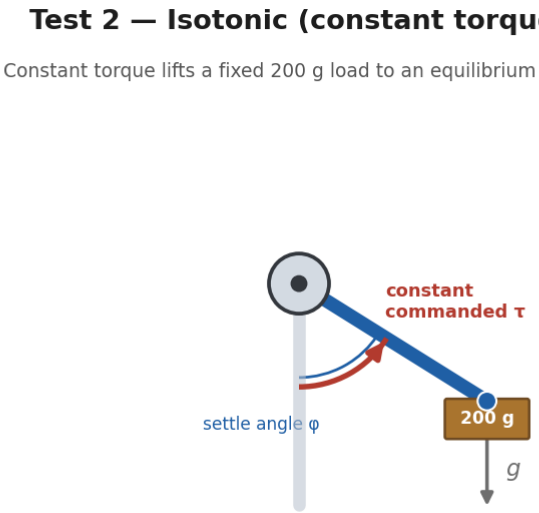


Figure 5.11: Setup for the isotonic test (Test 2). With the position spring disabled a constant torque command is applied and the lever rotates until the gravitational torque of the fixed 200 g load balances it; the settle angle is recorded against the commanded torque.

Figure 5.12(a) shows the reconstructed torque command held at a sequence of flat levels as the command is stepped, and panel (b) plots the angle at which the arm settled. The settle angle rises monotonically from about 28° to 62° as the command is increased, consistent with a steeper equilibrium angle at which the motor action balances the load's gravitational torque ($\tau = m{\cdot}g{\cdot}r{\cdot}\sin\varphi$). These curves demonstrate command consistency and the expected static equilibrium trend. The absolute torque values inherit the approximately 1.6× voltage-mode scale factor identified in the isometric test, and because i_q was not measured during this run, the flat reconstructed-torque trace should not be interpreted as an independent measurement of constant output torque.

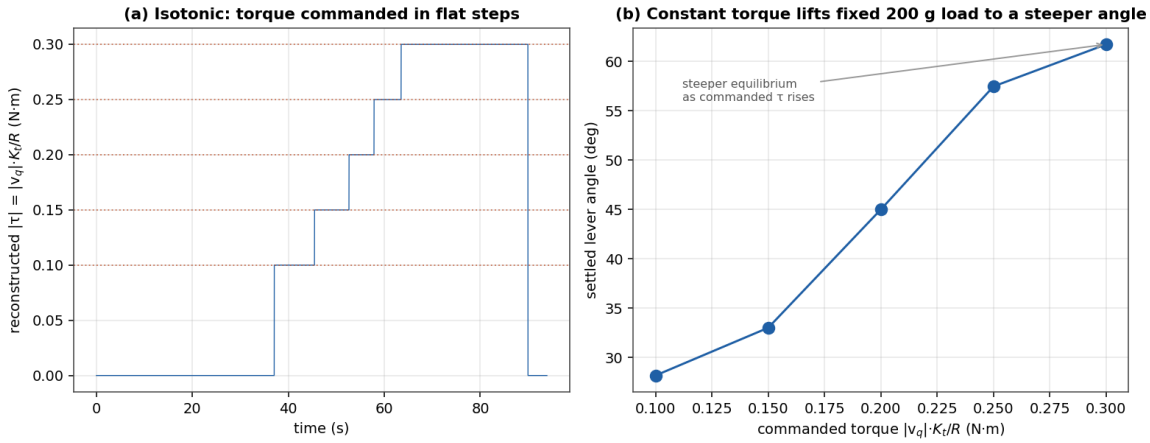


Figure 5.12: Isotonic test (Test 2). (a) Reconstructed torque commanded in flat steps. (b) The constant torque command lifts a fixed 200 g load to a progressively larger settle angle (about 28° to 62°) as the command is increased from 0.10 to 0.30 N·m.

Isokinetic — constant angular velocity (Test 3). Selecting velocity mode with a set-point of +2, 0 and −2 rad/s drives the arm at a controlled speed regardless of the resisting load, the kinematic dual of the isotonic mode and the operating principle of clinical isokinetic dynamometry [30]; the arm rotates continuously through several turns at each set-point (Figure 5.13).

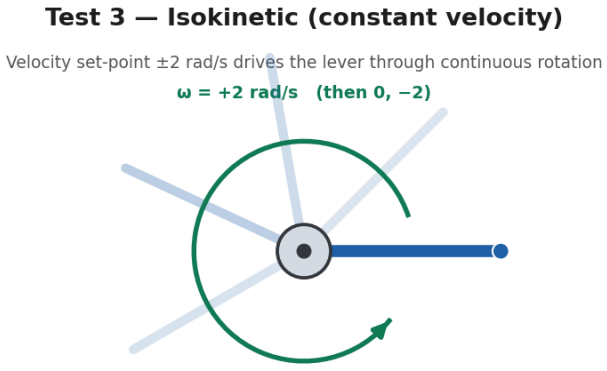


Figure 5.13: Setup for the isokinetic test (Test 3). A velocity set-point of ±2 rad/s drives the arm at constant speed through continuous rotation.

Figure 5.14 shows the measured shaft velocity (raw and 0.4 s moving average) against the ±2 rad/s set-point. Averaged over each interval the speed tracks the command well (mean +2.20 and −2.02 rad/s), but the instantaneous estimate is visibly noisy: the 1 kHz finite difference of the 8192-count encoder is coarse at these low speeds, and in the open-loop voltage path the velocity controller produces the bang-bang-like excursions seen in the trace rather than a smoothly regulated speed. The mode is therefore functional at the level of average velocity, with regulation quality that a closed current loop and a higher loop rate (Chapter 6) would improve.

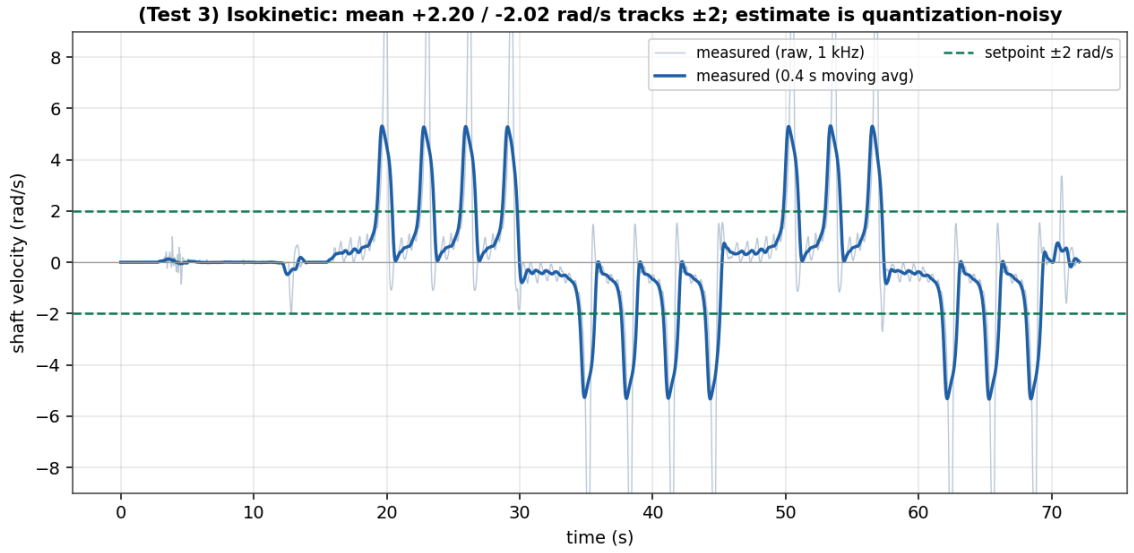


Figure 5.14: Isokinetic test (Test 3). Measured shaft velocity (raw and 0.4 s moving average) against the ±2 rad/s set-point; the mean tracks the command (+2.20 / −2.02 rad/s) while the instantaneous estimate is quantization-noisy.

Adaptive resistance (Test 4). The adaptive mode exercises the velocity-gated scaling at the deployed operating point: in impedance mode the stiffness and ceiling grow with the operator's effort once the adaptive gain is enabled ($A = 2.0$), about the gentle deployed baseline $L_{base} = 0.05$ N·m (Figure 5.15). The run is deliberately split into two intervals — a baseline interval with adaptation disabled ($A = 0$) followed by an adaptive interval with $A = 2.0$ — so that the gating and the scaling law can both be checked on the same arm.

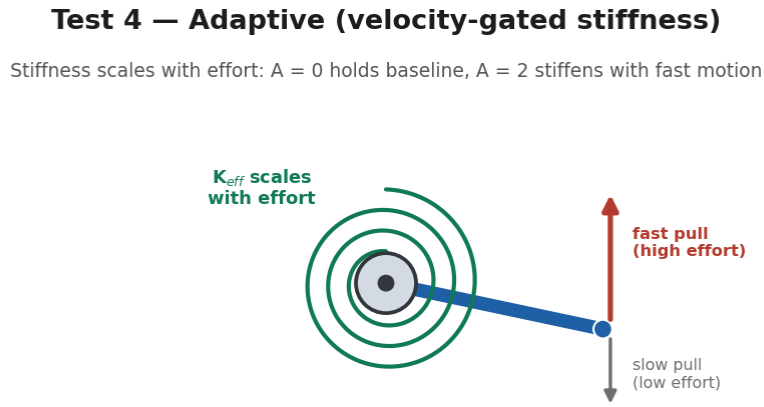


Figure 5.15: Setup for the adaptive test (Test 4). The operator pulls the arm slowly and then quickly; the effective stiffness and ceiling scale with the measured effort (low-pass-filtered speed while pushing outward).

Figure 5.16(a) overlays the effective stiffness and the effort metric against time across the two intervals. In the baseline interval ($A = 0$) the effort metric responds vigorously to hard pulls — reaching values near 9 — yet K_{eff} is pinned at its 0.5 N·m/rad baseline, confirming that the adaptation is correctly gated off. In the adaptive interval ($A = 2.0$) the same effort excursions drive K_{eff} up in bursts above 5 N·m/rad. Panel (b) plots K_{eff} against the effort metric: the baseline points sit on the 0.5 N·m/rad line, while the adaptive points fall on the analytic law $K_{eff} = K_{base} \cdot \left(1 + A \cdot \frac{effort}{v_{ref}}\right)$ with $v_{ref} = 2.0$ rad/s, matching it to an RMS deviation of 3.3×10^{-4} N·m/rad over the 1362 adaptive-interval samples ($R^2 = 1.000$), with the fitted slope and intercept both equal to the design value of 0.5 to within ±0.001 at 95% confidence. A patient who can move quickly is therefore met with progressively firmer resistance, while gentle motion is exercised against the soft baseline.

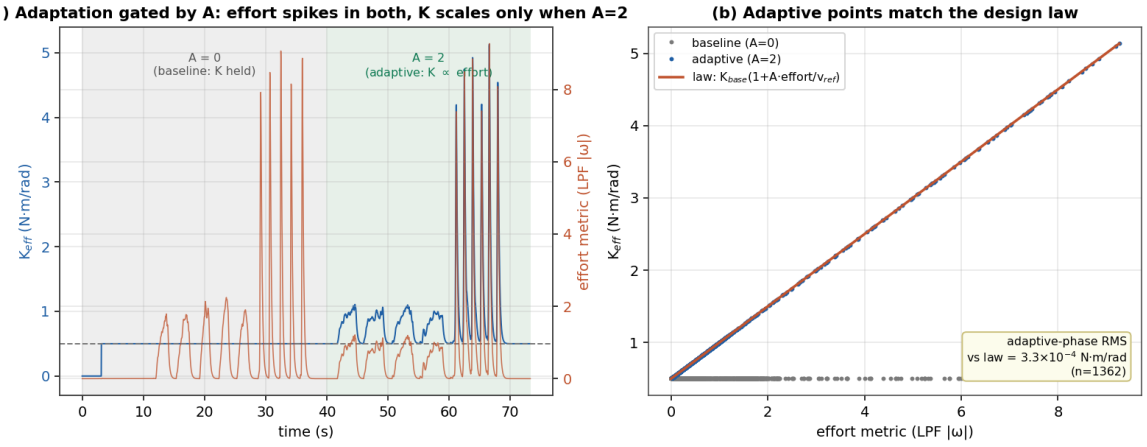


Figure 5.16: Adaptive test (Test 4) at the deployed baseline (K_base = 0.5 N·m/rad, L_base = 0.05 N·m, A = 2.0). (a) Effort and K_eff against time: with adaptation off (A = 0) effort spikes but K_eff stays at baseline; with A = 2.0 it scales with effort. (b) K_eff versus effort - the adaptive points match the law K_base·(1 + A·effort/v_ref) to an RMS of 3.3 × 10⁻⁴ N·m/rad (n = 1362).

Transparent back-drive (Test 5). In the passive boot state, the controller commands zero q-axis voltage ($v_q = 0$), so the motor produces no holding torque, and the arm is free to be moved by the patient — the transparent mode used for passive range-of-motion exercise (Figure 5.17). The de-energised arm was lifted and released to ring down freely, and small weights (15, 20 and 32 g) were then hung to find the static breakaway torque.

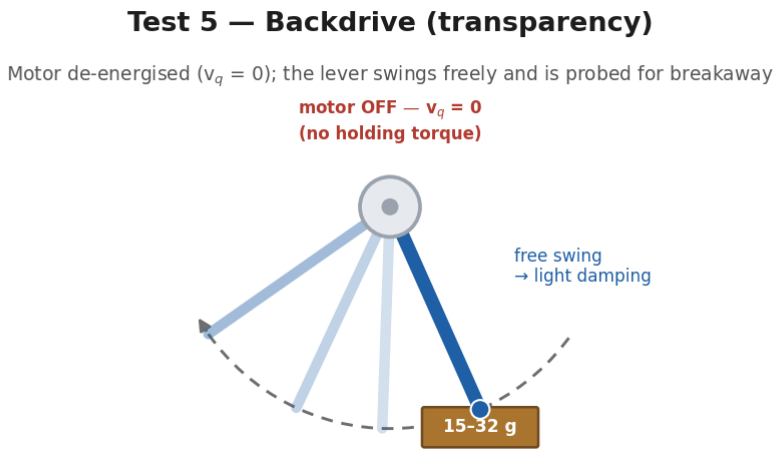


Figure 5.17: Setup for the back-drive test (Test 5). The motor is de-energised (v_q = 0); the free arm is released to ring down and small weights are added to find the breakaway torque.

Figure 5.18(a) shows the free swing of the de-energised arm: it oscillates for many seconds with only gradual amplitude loss, a log-decrement estimate over the free-swing placing the effective damping ratio in the lightly-damped range $\zeta \approx 0.02–0.05$. Panel (b) gives the static breakaway torque inferred from the smallest weights that initiated motion, m·g·r, spanning 0.018–0.038 N·m. Together these confirm that the actuator is highly back-drivable, with a low residual friction floor — the property a transparent exercise mode requires.

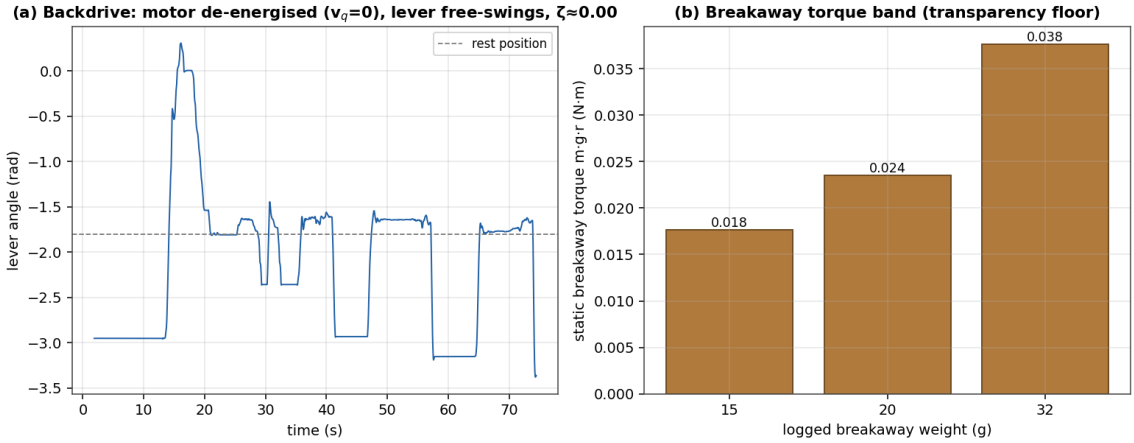


Figure 5.18: Back-drive / transparency test (Test 5). (a) The de-energised arm swings freely with light damping ($\zeta \approx 0.02–0.05$). (b) Static breakaway torque for the 15 / 20 / 32 g weights, 0.018–0.038 N·m.

5.8.2. Back-Drivability in the As-Built System and Clinical Safeguards

The transparent back-drive test above shows what happens when the present bench is driven externally. In the passive state the controller commands $v_q = 0$, so there is no active holding torque: once the applied torque exceeds the measured 0.018–0.038 N·m static breakaway level, the lever moves freely and rings down with only light damping ($\zeta \approx 0.02–0.05$). When

the drive is energized in impedance mode, external motion instead creates the commanded reaction $\tau = -K(\theta - \theta_{eq}) - B_{\omega}$, bounded by the torque ceiling. Back-drivability is therefore intentional during transparent exercise, but it must be managed whenever the device is expected to support or resist a limb. Five mechanisms are relevant, acting through the control law, the electrical machine, or mechanical hardware.

The first is control-law resistance, which is what the impedance kernel already provides. Raising the virtual stiffness raises the torque produced per unit of displacement, so the device holds position more firmly; this is exactly the isometric mode of Section 5.8.1. Its limitation is that it depends on the controller running: it provides no resistance at all if the loop stops, and the achievable stiffness is bounded above by the current-loop bandwidth, because a rendered stiffness whose natural frequency approaches the loop rate produces the limit cycle discussed in Section 6.1.

The second is feed-forward compensation of the loads that cause unwanted motion, principally gravity and friction. A gravity term of the form $\tau = m \cdot g \cdot r \cdot cos\varphi$ added to the commanded torque holds a limb against its own weight without requiring the impedance law to develop a position error first, which is the standard approach in patient-cooperative control [28], [29]; a Coulomb friction term of the form $\tau = -\tau_c \cdot sgn(\omega)$ does the converse, cancelling residual friction to make the device feel more transparent. Both are open-loop corrections and therefore only as good as the parameters they use, but neither requires additional hardware.

The third is electrical, and it is the mechanism most often overlooked. Short-circuiting the three motor terminals through the inverter's low-side devices turns the machine into a generator loaded by its own winding resistance, so back-driving it dissipates energy and produces a braking torque that rises with speed. The resulting damping is approximately $\left(\frac{3}{2}\right) \cdot pp^2 \cdot \frac{\lambda_{PM}^2}{R}$ per unit of mechanical angular velocity; for the machine used here that evaluates to roughly 0.02 N·m·s/rad, which is small but is available with the microcontroller halted and therefore survives a firmware fault. It is a velocity-dependent brake, not a holding brake, so it limits how fast the device can be back-driven rather than preventing motion.

The fourth is a mechanical holding brake: a spring-applied, electrically released device on the motor shaft. Because it is spring-applied it engages when power is removed, which makes it the only mechanism in this list that holds position through a complete loss of supply. It adds rotor inertia and cost, and it is the conventional answer where a limb must remain supported after a fault.

The fifth is a non-back-drivable transmission such as a worm gear, which prevents back-driving entirely by geometry; it is listed for completeness and rejected here, because it defeats the transparency the application requires and would defeat the transparent back-drive behavior measured above.

Table 5.4: Mechanisms for resisting back-driving, and their behavior under fault.

Mechanism	Acts through	Holds at zero speed?	Survives loss of control?	Adopted here
Virtual stiffness	Control law	Yes, within the torque ceiling	No	Yes

Mechanism	Acts through	Holds at zero speed?	Survives loss of control?	Adopted here
Gravity / friction feed-forward	Control law	Yes	No	Not implemented
Three-phase terminal short	Machine	No — damping only	Yes, if the gate driver defaults to it	Not implemented
Spring-applied holding brake	Mechanical hardware	Yes	Yes	No — recommended for clinical use
Non-back-drivable gearing	Mechanical hardware	Yes	Yes	No — defeats transparency

The as-built bench implements only the first mechanism during energized operation; the passive test intentionally removes that resistance to measure transparency. A patient-facing version should retain impedance control for normal operation, add gravity feed-forward so supporting a limb does not require a large position error, configure a safe terminal-short or equivalent dynamic-braking state for a halted controller, and use a spring-applied holding brake wherever loss of supply must not release a supported limb. Non-back-drivable gearing is not recommended because it would undermine the low-friction transparency that motivates the direct-drive architecture.

Taken together, the five tests show that the same low-cost control stack can produce a sustained hold, a repeatable constant-torque command with load-dependent equilibrium, regulated average velocity, effort-adaptive resistance and transparent free motion. The isotonic result is deliberately described as a commanded-torque demonstration because q-axis current was not measured in that run; Section 5.2.4 states the additional evidence required to claim constant physical torque. The isometric deadweight calibration remains the one measurement independent of the commanded law and quantifies the principal limitation of open-loop voltage operation: the rendered torque follows the programmed law in shape and reproducibility but carries an absolute-scale error of about 1.6×, which closing the current loop is intended to remove.

Chapter 6
Conclusion and Future Work

The preceding chapters establish that MyFOC implements a complete field-oriented control stack within the resource budget of an ATmega328P and that a single impedance kernel is sufficient to build a programmable rehabilitation application. Several limitations qualify how far those results extend, and they in turn define the most useful directions for future work. The most significant concerns the current-sensing path, which is implemented but is not yet exercised end-to-end in the application firmware.

6.1. Limitations

The library provides every element of a closed-loop current controller – a low-side current-sense interface, the forward Clarke and Park transforms applied to the measured phase currents, independent d- and q-axis current PI loops, Internal-Model-Control tuning, and a cross-axis decoupling feed-forward – but this closed-loop path was not used to collect any of the application results. The rehabilitation recordings were taken in the library's voltage-mode commutation path, in which the commanded q-axis voltage is applied at the measured rotor electrical angle through the inverse Park and space-vector PWM stages with no current feedback. In this sense the current-sensing capability is present in the code base but is not yet fully supported as a validated operating mode.

Operating in voltage mode has two practical consequences. First, torque is produced open-loop through the relationship $v_q = R \cdot \left(\frac{\tau_{des}}{K_t}\right)$, which neglects the inductive $\frac{di}{dt}$ term and the speed-dependent back-EMF $\omega_e \cdot \lambda_{PM}$; the approximation is accurate only at low speed and under quasi-static loading, so at higher speeds or during fast transients the achieved q-axis current, and therefore the rendered torque, departs from the commanded value. Second, without current feedback the firmware cannot enforce a true current limit; protection relies instead on the voltage ceiling and on the conservative torque bound derived from the phase resistance. The closed-loop current loop was brought up on the bench, but its tuning was hampered by the inductance-identification limitation described below, which produced a sustained limit cycle when the IMC gains were computed directly from the measured inductance.

The d-axis inductance is the least trustworthy of the four identified motor parameters, as already noted in Section 5.6. The electrical time constant $\frac{L}{R}$ of the small gimbal motor is short relative to both the current-sense resolution and the inner-loop period, so the step-based identification is dominated by stochastic threshold-crossing rather than by the true time constant. Because the IMC current-loop gains depend directly on the inductance, this uncertainty propagates straight into the current-loop tuning and is the proximate reason the closed-loop mode is not yet relied upon.

The target platform also bounds what the controller can achieve. The ATmega328P is an 8-bit device running at 16 MHz with no hardware floating-point unit, two kilobytes of SRAM and a ten-bit analog-to-digital converter; the field-oriented inner loop executes at roughly one to one-and-a-half kilohertz. This loop rate caps the achievable current-loop bandwidth and the temporal fidelity of fast transient effects, the emulated floating-point arithmetic consumes

a substantial share of each loop, and the ten-bit converter limits current-sense resolution, compounding the inductance-identification difficulty above.

Two further limitations concern measurement and validation scope. The AMT103 encoder is incremental, so the controller requires an alignment and zeroing procedure at every power-up and retains a small residual offset from the open-loop alignment; it also provides no absolute position. Because the results were taken in voltage mode, the torque reported in Section 5.8 is reconstructed from the commanded q-axis voltage rather than measured with an independent transducer, so the close agreement with the programmed laws reflects rendering fidelity and internal consistency rather than an absolute torque calibration. The one independent check that was performed — the deadweight calibration of the isometric mode — bears this out quantitatively: the reconstructed torque over-predicts the true gravitational torque by a factor of roughly 1.6, so the absolute scale, as distinct from the shape and reproducibility of the rendered laws, is trustworthy only to within that factor until the current loop is closed. The most probable origin of this scale error is the torque constant itself. Because $K_t = 1.5 \cdot p \cdot \lambda_{PM}$ is derived from the permanent-magnet flux linkage λ_{PM}, and λ_{PM} is in turn identified from a back-EMF measurement taken while the motor is spinning, any resistive or friction-related voltage that is not fully separated from the open-circuit back-EMF inflates λ_{PM} and propagates directly into K_t and hence into the reconstruction $\tau = v_q \cdot \frac{K_t}{R}$. The observed factor corresponds to λ_{PM} (identified at 0.0346 Wb) being high by roughly 1.5–1.6 — a true value nearer 0.022 Wb, or equivalently a true K_t nearer 0.37 N·m/A rather than the computed 0.57 N·m/A. Accounting for the discrepancy through the resistance instead would require a true phase resistance near 17 Ω against the identified 11 Ω — some six ohms of extra series resistance that the inverter on-resistance and wiring cannot plausibly supply — so an over-estimated torque constant is the more likely dominant cause, with any resistance under-estimation a smaller secondary contributor. The lever radius enters the gravitational torque linearly and is measured directly, so a geometric error large enough to explain the factor is unlikely. The hypothesis is straightforward to confirm on the bench by measuring K_t directly from a locked-rotor torque-versus-current characteristic and the phase resistance with a four-wire ohmmeter; either measurement would isolate the cause, and closing the current loop would in any case replace the open-loop voltage-to-torque mapping with a calibrated current-controlled torque. Finally, all results were obtained from a single gimbal motor on the bench, without representative human-limb loading, thermal-endurance testing or long-duration testing; nor was a repeatability study performed, since every result derives from a single bench session, so run-to-run variance and the uncertainty of the identified motor parameters are not quantified; and the rehabilitation application has not undergone clinical or human-subject evaluation.

6.2. Future Work

The clearest priority is to bring the closed-loop current path into routine use. This entails linking the low-side current sense in the application firmware, re-tuning the d- and q-axis current loops against a trustworthy inductance estimate, and verifying closed-loop torque regulation against a commanded current reference. Doing so would replace the open-loop voltage-to-torque mapping with true current-controlled torque – improving accuracy at speed and during transients and enabling feedback-based current limiting – and would make the impedance rendering independent of the actuator's electrical dynamics. Resolving the limit

cycle observed during bring-up, whether through anti-windup refinement, a more conservative bandwidth, or a better inductance estimate, is the gating step.

Improving the parameter identification is closely tied to the previous point. A higher-resolution current-sense path – obtained through a higher gain setting on the DRV8302's internal amplifier or through an oversampling-and-decimation scheme on the ATmega328P's converter – would let the short $\frac{L}{R}$ time constant of small gimbal motors fall within the converter's resolution. Alternatively, a signal-injection method that measures the current response to a high-frequency excitation would estimate the inductance without relying on a fast step response. Either route would directly enable the current-loop tuning called for above.

Porting the library to a faster 32-bit microcontroller with a hardware floating-point unit, higher converter resolution and a faster control loop – of the kind already supported by portable FOC libraries – would raise the achievable current-loop bandwidth and the fidelity of fast transient effects, while leaving the library's structure and interfaces unchanged. The present code deliberately isolates the platform-specific timer, converter and pin handling, so such a port is expected to be confined to a small number of modules.

Several functional extensions follow naturally once closed-loop current control is in place. Field-weakening, implemented through a non-zero d-axis current reference, would extend the usable speed range; an absolute or sensorless position scheme would remove the boot-time alignment; and additional rehabilitation modes, together with mechanical-load and thermal characterisation, would broaden the validation envelope. For the rehabilitation application specifically, characterisation against a human operator and validation of the adaptive-scaling thresholds against patient-specific recovery data remain the principal outstanding work.

6.3. Path to Clinical Deployment

The bench system described in this report is a research and teaching platform; turning it into a device that can be used on patients in a clinic is a substantial change of envelope rather than a change of architecture. The mathematics, alignment procedure, impedance kernel and adaptive-scaling overlay all transfer unchanged, and only three classes of hardware need to scale up — the actuator, the sensing chain and the host processor — together with a safety and regulatory layer that does not exist on the bench at all. The remainder of this subsection sketches what each of those entails and quantifies the resulting parts-cost increase.

The bench actuator, a GM4108-120T gimbal motor, produces approximately 0.42 N·m at its motor-limited current ceiling. Upper-limb rehabilitation at the shoulder or elbow typically calls for 5–15 N·m at the joint, an order of magnitude more, so the actuator must be replaced. Two practical paths exist. A quasi-direct-drive (QDD) actuator [23] such as the T-Motor AK80-9 (approximately USD 400, around 18 N·m peak with integrated driver and encoder) keeps the system back-drivable and adds little mechanical complexity, at the cost of moderate torque density and limited continuous rating. A frameless brushless servo such as the Kollmorgen or RoboDrive ILM-50x08 class (USD 600–2000) paired with a low-backlash harmonic-drive gearbox (USD 1500–4000) offers higher torque density and a more conventional industrial pedigree, at the cost of reduced transparency and increased mechanical complexity. The DRV8302 gate driver used on the bench is sufficient for the small gimbal motor but undersized for clinical actuators; an ODrive S1 (USD 200) or moteus

n1 (USD 130) is appropriate for the QDD path, while an industrial servo drive (USD 500–2000) suits the geared path.

The single most consequential addition for clinical use is an in-line torque sensor — a rotary strain-gauge transducer placed between the actuator output and the patient interface, with a typical cost of USD 1500–5000. This sensor would close the torque loop directly on a calibrated mechanical measurement and would eliminate the 1.6× absolute-scale error documented in Section 5.8 at its root, without requiring resolution of the open-loop voltage-mode reconstruction discussed in Sections 5.8 and 6.1. The incremental AMT103 should be replaced by an absolute device — an AS5048A magnetic encoder (USD 20) or an AMT22 absolute incremental (USD 35) for cost-sensitive work, or a higher-resolution absolute optical encoder (USD 200–500) for clinical-grade repeatability — so that the device boots into a known joint configuration without the alignment ramp the bench requires. The ATmega328P should give way to a 32-bit Cortex-M with a hardware FPU and 12-bit ADC: a Teensy 4.1 (USD 30) or an STM32H7 development board (USD 50–200) raises the current-loop rate from the bench's 1–1.5 kHz to a clinically conventional 10–20 kHz, restoring the bandwidth needed for higher-fidelity impedance rendering. The library's platform-specific code is confined to three modules (MyBLDCDriver3PWM, MyEncoder and MyCurrentSense3PWM), so the port is bounded.

The safety layer changes character. A clinical device must include a mechanically latching hardware emergency stop accessible to both patient and therapist, an independent watchdog implemented on a separate microcontroller, a mechanical torque limiter such as a slip clutch on the output shaft, a medical-grade isolated power supply, and sanitisable patient-contact surfaces. Beyond hardware, the device falls under IEC 60601-1 (general medical electrical safety), IEC 60601-1-2 (electromagnetic compatibility) and ISO 14971 (risk management); achieving certification typically dominates the total cost of bringing a clinical device to market, often in the range of USD 50,000 to USD 500,000 for a first device. Those costs are not parts costs and are not included in the comparison below.

Table 6.1 compares per-axis parts cost for three configurations: the bench presented here, a low-cost clinical-grade configuration suitable for early-stage prototype evaluation, and a mid-range configuration representative of what a serious clinical prototype would use.

Table 6.1: Indicative per-axis parts cost (USD) for the bench presented here and two clinical-grade configurations. Excludes certification, software licences, and labour.

Component	Bench (this work)	Clinical low	Clinical mid
Motor	$40 (GM4108-120T gimbal)	$400 (T-Motor AK80-9 QDD)	$1200 (frameless servo)
Gearbox	none (direct drive)	none (QDD)	$2500 (harmonic drive)
Driver	$30 (DRV8302 breakout)	$200 (ODrive S1)	$800 (industrial servo drive)

Component	Bench (this work)	Clinical low	Clinical mid
MCU	$15 (UNO clone)	$30 (Teensy 4.1)	$200 (STM32H7 board)
Position sensor	$30 (AMT103 incremental)	$20 (AS5048A absolute)	$200 (absolute optical)
Torque sensor	none	$1500 (strain-gauge)	$4000 (rotary transducer)
Safety hardware	none	$200 (E-stop, watchdog)	$800 (full safety chain)
Mechanical / ergonomic	$20 (lever, weights)	$500 (handle, mount)	$2000 (clinical frame)
Per-axis subtotal	**~$135**	**~$2,850**	**~$11,700**

The per-axis numbers in Table 6.1 scale roughly linearly with the number of actuated joints. A representative three-degree-of-freedom upper-limb device (shoulder, elbow and wrist) would therefore land in the range of USD 9,000 to USD 35,000 in parts alone, depending on configuration. By comparison, commercial upper-limb rehabilitation systems — Hocoma ArmeoPower and Bionik InMotion Arm being the best-known examples [5], [6], [24] — retail for roughly USD 150,000–200,000 and USD 80,000–150,000 respectively. The order-of-magnitude gap between parts cost and retail price reflects the certification, clinical validation, integration and warranty costs noted above, not the cost of the hardware itself.

What MyFOC contributes in this context is not a clinical device but the entry barrier to one. The bench at USD 135 is roughly twenty times cheaper than the cheapest clinical-grade configuration and roughly a thousand times cheaper than a commercial endpoint, so a student or research group can explore impedance control, adaptive scaling and the algorithmic side of rehabilitation robotics — including the mathematical and control issues that dominate the early prototype stage — before incurring any of the cost of the clinical pathway. The same library code, with its platform-specific layer replaced, then ports forward into the clinical prototype rather than being discarded.

6.4. Regulatory Pathway and Certification

Section 6.3 costs the hardware required to move from a bench to a clinical device but treats certification only as a lump sum. Because that lump sum dominates the total, and because the applicable requirements shape the design rather than merely audit it, this section sets out what certification actually involves for a device of this kind.

A powered upper-limb rehabilitation trainer is an active medical device that makes physical contact with a patient. In the United States such devices are generally Class II and reach the market through premarket notification under section 510(k), in which substantial equivalence

to an existing legally marketed predicate is demonstrated [46]. In the European Union the same device falls under Regulation (EU) 2017/745, which requires a notified body for anything above Class I [47]. In Canada it is regulated under the Medical Devices Regulations, which require a Medical Device Licence for Class II and above and, separately, an ISO 13485 quality-system certificate issued under the Medical Device Single Audit Program [48]. All three routes rest on the same underlying standards, so technical work is largely common even though the submissions are not.

Table 6.2 lists the standards that apply. Two deserve particular attention. The first is ISO 14971 [49], which is not a test but a process: risk management must be planned before design begins, and every design decision that acts as a risk control — the torque ceiling of Section 3.4, the hardware over-current trip, the watchdog proposed in Section 6.3 — must be traceable to an identified hazard and verified as effective. Retrofitting that traceability to an existing design is substantially more expensive than building it in, which is the strongest argument for treating the safety architecture of Section 3.4 as a regulatory artefact rather than an engineering convenience. The second is IEC 80601-2-78 [50], the particular standard for medical robots used for rehabilitation, assessment, compensation or alleviation. It is directly applicable to this device and did not exist when much of the rehabilitation-robotics literature cited in Chapter 1 was published; among other things it imposes requirements on unintended movement, on patient release, and on the protective stop functions that a device physically coupled to an impaired limb must provide.

Table 6.2: Standards applicable to a powered upper-limb rehabilitation device.

Standard	Scope	Bearing on this design
ISO 14971 [49]	Risk management process	Every risk control must trace to a hazard; governs Section 3.4
IEC 60601-1 [51]	Basic safety and essential performance	Isolation, leakage current, mechanical hazards, single-fault safety
IEC 60601-1-2 [52]	Electromagnetic compatibility	Emissions and immunity; affects PWM carrier and cable routing
IEC 80601-2-78 [50]	Rehabilitation robots (RACA)	Unintended movement, patient release, protective stops
IEC 62304 [53]	Medical device software life cycle	Applies to the whole firmware; classification drives documentation depth
IEC 62366-1 [54]	Usability engineering	Therapist and patient interface, including the serial command set
ISO 10993-1 [55]	Biological evaluation	Patient-contacting handle and strap materials
ISO 13485 [56]	Quality management system	Required for CE marking and for a Canadian licence

The cost of certification separates into three parts and conflating them is the usual source of the very wide figures quoted in the literature. Type testing at an accredited laboratory is the most predictable: electrical safety testing to IEC 60601-1 and electromagnetic compatibility testing to IEC 60601-1-2 each typically occupies the low tens of thousands of US dollars for a device of this complexity, with biocompatibility evaluation of the patient-contacting parts

adding a similar amount. Submission itself is comparatively small: the United States charges a statutory user fee for a 510(k), reduced substantially for qualifying small businesses, and the preparation of the submission dossier is a documentation exercise. The dominant term is the quality system and the design history file — the risk-management file, the software life-cycle records required by IEC 62304, design verification and validation, and the ISO 13485 certification audit — because these are sustained organisational costs rather than one-off tests. Published estimates for taking a first Class II electromechanical device from a working prototype to clearance commonly fall between USD 100,000 and USD 500,000 excluding any clinical investigation, and the range is wide precisely because the third term depends on how much of the documentation already exists.

Two consequences follow for the present work. The first is that certification cost is largely independent of hardware cost, so the twenty-fold hardware saving demonstrated in Section 6.3 does not translate into a twenty-fold saving on a clinical product; what it lowers is the cost of the research and prototyping stage that precedes the regulatory pathway. The second is that several of the standards above bear directly on decisions already made. IEC 62304 applies to the firmware in its entirety, and a library that can be read end to end and traced line by line — the central claim of this project — is materially easier to document under it than a vendor binary whose internals are not available for inspection. That auditability, argued in Section 1.2 on engineering grounds, has a regulatory value that was not part of the original motivation.

6.5. Conclusion

This project has presented MyFOC, a Field-Oriented Control library implemented from first principles for the Arduino UNO + DRV8302 platform, together with a rehabilitation application built on its impedance kernel. The library was developed without reference to existing FOC firmware beyond the well-documented mathematical foundations, and the resulting code base of approximately 21 KB of flash and zero static SRAM beyond per-object instance state validates the claim that a full FOC stack - including parameter identification, IMC current-loop tuning, sensor alignment and an impedance controller - is implementable within the constraints of an ATmega328P-class device while leaving substantial headroom for application logic.

The application study presented in Section 5.8 demonstrates that the same impedance kernel suffices to render a programmable virtual spring–damper field with a layered safety ceiling, and that a velocity-gated adaptive-scaling overlay can be added without disrupting the underlying loop. The controller was exercised as five clinically recognised exercise modes — isometric, isotonic, isokinetic, adaptive-resistance and transparent back-drive — on the bench hardware. In the adaptive mode the measured effective stiffness tracked the velocity-gated design law to an RMS of 3.3×10^{-4} N·m/rad over 1362 samples while remaining correctly held at the baseline when the adaptation was gated off, confirming that the firmware implements the law without numerical degradation. An independent deadweight calibration performed in the isometric mode showed that the open-loop voltage-mode torque reconstruction over-predicts the physically delivered torque by a factor of roughly 1.6, the clearest single measure of the absolute-scale error that closing the current loop would remove. These results were obtained in the library's voltage-mode commutation path, with

the current/torque loop open; bringing the implemented closed-loop current controller into validated use is the foremost item of future work (Chapter 6).

Of the limitations and extensions set out in the preceding chapter, two are the most immediately actionable. First, the inductance-identification routine identified in Section 5.6 as the least-trusted of the four motor parameters would benefit from a higher-resolution current-sense path, either through a higher gain setting on the DRV8302's internal amplifier or through an oversampling scheme on the ATmega328P's ADC, so that the L/R time constant of small gimbal motors falls comfortably within the converter's resolution. Second, the rehabilitation application as presented operates without clinical evaluation; the existing torque-ceiling and adaptive-scaling primitives are designed to be safe and clinically expressive, but characterisation against a human operator and validation of the adaptive thresholds against patient-specific recovery curves remain future work.

The contribution of this work is twofold. First, it demonstrates that a complete field-oriented control stack — motor parameter identification, Internal-Model-Control current-loop tuning, encoder alignment, space-vector commutation and an impedance controller — can be implemented from first principles within the resource budget of an 8-bit ATmega328P, in roughly 21 KB of flash and with no static SRAM beyond per-object state, while leaving headroom for application logic and remaining portable to faster microcontrollers without structural change. Second, it shows that a single impedance kernel, exposed through a compact serial command set, suffices to render five clinically recognised rehabilitation exercise modes — isometric, isotonic, isokinetic, adaptive-resistance and transparent back-drive — on that hardware, including a velocity-gated adaptive-stiffness overlay that tracks its design law to an RMS of 3.3×10^{-4} N·m/rad. A methodological contribution accompanies these: the library is built and documented transparently for reproduction, and its voltage-mode operating limits are characterised honestly, including a parameter-free deadweight calibration that quantifies the open-loop torque-reconstruction error.

The artifact and its accompanying documentation are intended as a pedagogically transparent and reproducible reference platform on which any of the above directions can be pursued.

References

[1] R. Krishnan, Permanent Magnet Synchronous and Brushless DC Motor Drives, CRC Press, Boca Raton FL, 2010.

[2] W. Leonhard, Control of Electrical Drives, 3rd ed., Springer, Berlin, 2001.

[3] A. Skuric, H. S. Bank, R. Unger, O. Williams and D. González-Reyes, "SimpleFOC: A field oriented control (FOC) library for controlling brushless DC (BLDC) and stepper motors," Journal of Open Source Software, vol. 7, no. 74, p. 4232, 2022.

[4] A. C. Lo et al., "Robot-assisted therapy for long-term upper-limb impairment after stroke," New England Journal of Medicine, vol. 362, no. 19, pp. 1772-1783, 2010.

[5] P. Maciejasz, J. Eschweiler, K. Gerlach-Hahn, A. Jansen-Troy, and S. Leonhardt, "A survey on robotic devices for upper limb rehabilitation," Journal of NeuroEngineering and Rehabilitation, vol. 11, art. 3, 2014.

[6] H. I. Krebs, N. Hogan, M. L. Aisen, and B. T. Volpe, "Robot-aided neurorehabilitation," IEEE Transactions on Rehabilitation Engineering, vol. 6, no. 1, pp. 75-87, 1998.

[7] N. Hogan, "Impedance control: An approach to manipulation: Part I - Theory; Part II - Implementation; Part III - Applications," ASME Journal of Dynamic Systems, Measurement, and Control, vol. 107, no. 1, pp. 1–24, March 1985.

[8] N. Hogan and S. P. Buerger, "Impedance and interaction control," in Robotics and Automation Handbook, T. R. Kurfess, Ed., CRC Press, 2005, ch. 19.

[9] J. M. Veerbeek, A. C. Langbroek-Amersfoort, E. E. H. van Wegen, C. G. M. Meskers, and G. Kwakkel, "Effects of robot-assisted therapy for the upper limb after stroke: A systematic review and meta-analysis," Neurorehabilitation and Neural Repair, vol. 31, no. 2, pp. 107-121, 2017.

[10] Texas Instruments, InstaSPIN-FOC and InstaSPIN-MOTION User's Guide, SPRUHJ1, Dallas, TX, USA, 2013.

[11] STMicroelectronics, X-CUBE-MCSDK: Motor Control Software Development Kit, Documentation and User Manual, Geneva, Switzerland, 2023.

[12] E. Clarke, Circuit Analysis of A-C Power Systems, vol. 1, John Wiley & Sons, New York, 1943.

[13] R. H. Park, "Two-reaction theory of synchronous machines - Generalized method of analysis," Transactions of the AIEE, vol. 48, no. 3, pp. 716–727, July 1929.

[14] K. Hasse, "Zur Dynamik drehzahlgeregelter Antriebe mit stromrichtergespeisten Asynchron-Kurzschlusslaufermaschinen," Dr.-Ing. dissertation, Technische Hochschule Darmstadt, Darmstadt, Germany, 1969.

[15] F. Blaschke, "The principle of field orientation as applied to the new transvector closed-loop control system for rotating-field machines," Siemens Review, vol. 34, pp. 217-220, 1972.

[16] S.-K. Sul, Control of Electric Machine Drive Systems. Hoboken, NJ, USA: Wiley-IEEE Press, 2011.

[17] P. Vas, Sensorless Vector and Direct Torque Control. Oxford, U.K.: Oxford University Press, 1998.

[18] D. G. Holmes and T. A. Lipo, Pulse Width Modulation for Power Converters: Principles and Practice, IEEE Press / Wiley, Hoboken NJ, 2003.

[19] J. Holtz, "Pulsewidth modulation for electronic power conversion," Proceedings of the IEEE, vol. 82, no. 8, pp. 1194–1214, August 1994.

[20] B. Vedder, "VESC - Open source ESC," VESC Project. [Online]. Available: https://vesc-project.com

[21] ODrive Robotics, "ODrive: High performance motor control." [Online]. Available: https://odriverobotics.com

[22] mjbots Robotic Systems, "moteus brushless controller." [Online]. Available: https://github.com/mjbots/moteus

[23] B. Katz, J. Di Carlo, and S. Kim, "Mini Cheetah: A platform for pushing the limits of dynamic quadruped control," in Proc. IEEE Int. Conf. Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 6295-6301.

[24] T. Nef, M. Mihelj, and R. Riener, "ARMin: A robot for patient-cooperative arm therapy," Medical & Biological Engineering & Computing, vol. 45, no. 9, pp. 887-900, 2007.

[25] G. Colombo, M. Joerg, R. Schreier, and V. Dietz, "Treadmill training of paraplegic patients using a robotic orthosis," Journal of Rehabilitation Research and Development, vol. 37, no. 6, pp. 693-700, 2000.

[26] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), Pittsburgh, PA, USA, 1995, pp. 399-406.

[27] T. Proietti, V. Crocher, A. Roby-Brami, and N. Jarrasse, "Upper-limb robotic exoskeletons for neurorehabilitation: A review on control strategies," IEEE Reviews in Biomedical Engineering, vol. 9, pp. 4-14, 2016.

[28] L. Marchal-Crespo and D. J. Reinkensmeyer, "Review of control strategies for robotic movement training after neurologic injury," Journal of NeuroEngineering and Rehabilitation, vol. 6, art. 20, 2009.

[29] E. T. Wolbrecht, V. Chan, D. J. Reinkensmeyer, and J. E. Bobrow, "Optimizing compliant, model-based robotic assistance to promote neurorehabilitation," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 16, no. 3, pp. 286-297, 2008.

[30] Z. Dvir, Isokinetics: Muscle Testing, Interpretation and Clinical Applications, 2nd ed. Edinburgh, U.K.: Churchill Livingstone, 2004.

[31] T. Nef, M. Guidali, and R. Riener, “ARMin III – arm therapy exoskeleton with an ergonomic shoulder actuation,” Applied Bionics and Biomechanics, vol. 6, no. 2, pp. 127–142, 2009.

[32] V. Klamroth-Marganska et al., “Three-dimensional, task-specific robot therapy of the arm after stroke: a multicentre, parallel-group randomised trial,” The Lancet Neurology, vol. 13, no. 2, pp. 159–166, 2014.

[33] CubeMars (T-Motor), AK Series Actuator Manual, V1.0.9, Nanchang Kude Intelligent Technology Co., Ltd. [Online]. Available: https://www.cubemars.com

[34] S. H. P. Kotte, J. Viveen, K. L. M. Koenraadt, B. The, and D. Eygendaal, “Normative values of isometric elbow strength in healthy adults: a systematic review,” Shoulder & Elbow, vol. 10, no. 3, pp. 207–215, 2018.

[35] L. J. Askew, K. N. An, B. F. Morrey, and E. Y. Chao, “Isometric elbow strength in normal individuals,” Clinical Orthopaedics and Related Research, no. 222, pp. 261–266, 1987.

[36] D. A. Winter, Biomechanics and Motor Control of Human Movement, 4th ed. Hoboken, NJ, USA: John Wiley & Sons, 2009.

[37] T. K. Morimoto, P. Blikstein, and A. M. Okamura, "Hapkit: An open-hardware haptic device for online education," in Proc. IEEE Haptics Symposium, Houston, TX, USA, 2014.

[38] CUI Devices, AMT10 Series Modular Incremental Encoder Datasheet, March 2021.

[39] Microchip Technology Inc., ATmega328P 8-bit AVR Microcontroller Datasheet, document DS40002061B, August 2020.

[40] S. Skogestad and I. Postlethwaite, Multivariable Feedback Control: Analysis and Design, 2nd ed., John Wiley & Sons, Chichester, 2005.

[41] K. J. Åström and T. Hägglund, Advanced PID Control. Research Triangle Park, NC, USA: ISA - The Instrumentation, Systems and Automation Society, 2006.

[42] Texas Instruments, DRV8302 / DRV8303 Three-Phase Pre-Driver with Dual Current Shunt Amplifiers and Buck Regulator, datasheet SLOS738B, Texas Instruments Inc., revised May 2016.

[43] C. Ott, R. Mukherjee, and Y. Nakamura, “Unified impedance and admittance control,” in Proc. IEEE Int. Conf. Robotics and Automation (ICRA), Anchorage, AK, USA, 2010, pp. 554–561.

[44] Atmel Corporation, AVR121: Enhancing ADC Resolution by Oversampling, application note, document 8003.

[45] P. L. Jansen and R. D. Lorenz, “Transducerless position and velocity estimation in induction and salient AC machines,” IEEE Transactions on Industry Applications, vol. 31, no. 2, pp. 240–247, March/April 1995.

[46] U.S. Food and Drug Administration, Premarket Notification 510(k), 21 CFR Part 807 Subpart E, Silver Spring, MD, USA.

[47] Regulation (EU) 2017/745 of the European Parliament and of the Council of 5 April 2017 on medical devices, Official Journal of the European Union, L 117, 2017.

[48] Government of Canada, Medical Devices Regulations, SOR/98-282, Ottawa, ON, Canada.

[49] ISO 14971:2019, Medical devices — Application of risk management to medical devices, International Organization for Standardization, Geneva, 2019.

[50] IEC 80601-2-78:2019+AMD1:2024, Medical electrical equipment — Part 2-78: Particular requirements for basic safety and essential performance of medical robots for rehabilitation, assessment, compensation or alleviation, International Electrotechnical Commission, Geneva.

[51] IEC 60601-1:2005+AMD1:2012+AMD2:2020, Medical electrical equipment — Part 1: General requirements for basic safety and essential performance, International Electrotechnical Commission, Geneva.

[52] IEC 60601-1-2:2014+AMD1:2020, Medical electrical equipment — Part 1-2: Collateral standard: Electromagnetic disturbances, International Electrotechnical Commission, Geneva.

[53] IEC 62304:2006+AMD1:2015, Medical device software — Software life cycle processes, International Electrotechnical Commission, Geneva.

[54] IEC 62366-1:2015+AMD1:2020, Medical devices — Part 1: Application of usability engineering to medical devices, International Electrotechnical Commission, Geneva.

[55] ISO 10993-1:2018, Biological evaluation of medical devices — Part 1: Evaluation and testing within a risk management process, International Organization for Standardization, Geneva, 2018.

[56] ISO 13485:2016, Medical devices — Quality management systems — Requirements for regulatory purposes, International Organization for Standardization, Geneva, 2016.

[57] R. Häberle, F. Schellenberg, R. List, M. Plüss, W. R. Taylor, and S. Lorenzetti, "Comparison of the kinematics and kinetics of shoulder exercises performed with constant and elastic resistance," BMC Sports Science, Medicine and Rehabilitation, vol. 10, art. 22, 2018, doi: 10.1186/s13102-018-0111-7.

Appendix A. Source-File Organisation

The MyFOC library directory structure is summarised below. Italic entries are documentation; bold entries are required for the library to build under the Arduino IDE. The tree mirrors the layout produced by the deliverable archive.

```
MyFOC/
├── library.properties          [Arduino library manifest]
├── README.md                   [user-facing quick-start]
├── DOCUMENTATION.md            [detailed math + design notes]
├── src/
│   ├── MyFOC.h                 [umbrella include]
│   ├── MyFOCMath.h, .cpp       [transforms, SVPWM, sin/cos LUT]
│   ├── MyPID.h, .cpp           [discrete-time PID]
│   ├── MyLPF.h, .cpp           [first-order low-pass filter]
│   ├── MyEncoder.h, .cpp       [quadrature decoder]
│   ├── MyBLDCDriver3PWM.h, .cpp   [Timer1/2 PWM driver]
│   ├── MyCurrentSense3PWM.h, .cpp [DRV8302 current sense]
│   ├── MyBLDCMotor.h, .cpp     [FOC engine and control modes]
│   └── MyMotorID.h, .cpp       [R, Ld, lambda_PM identification +
IMC tuning]
└── examples/
  ├── Diagnostic/               [wiring check & encoder-CPR auto-
detect]
  ├── Motor_Identification/     [R, Ld, lambda_PM parameter
identification]
  └── Rehab_Device/             [impedance-control rehab application]
```

To install, the entire MyFOC directory is copied into the Arduino IDE's libraries folder (typically ~/Arduino/libraries on Unix-like systems and Documents\Arduino\libraries on Windows). The IDE then offers each example sketch under File → Examples → MyFOC. The application sketch that produced the empirical data in Chapter 5 is Rehab_Device. It follows a simple template: hardware pin definitions, identified-motor-parameter constants pasted from the parameter-identification procedure, the s*etup()* routine that initialises the encoder, driver, current sense, FOC pipeline and impedance kernel, and the *loop()* routine that runs *loopFOC(),* applies the impedance or torque law, calls *move(),* and emits a CSV row when logging is active. The sketch also exposes a one-character serial command interface for runtime parameter adjustment; the full command list is printed in response to *H or ?.*

Appendix B. Mathematical Derivations

This appendix collects the derivations cited but not fully worked out in the main text. Each subsection is self-contained; readers familiar with the result may skip to the next subsection.

B.1 Clarke Transformation (Amplitude-Invariant Form)

Let the three stator phase axes be unit vectors in the *αβ* plane:

$$a = (1,0), b = (-\frac{1}{2}, \frac{\sqrt{3}}{2}), c = \left(-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right) \tag{B.1}$$

The αβ projection of any three-phase quantity (v_a, v_b, v_c) is the vector sum scaled by an arbitrary constant *k*:

$$v_{\alpha\beta} = k \cdot (v_a \cdot a + v_b \cdot b + v_c \cdot c) \tag{B.2}$$

Component-wise:

$$v_\alpha = k \cdot \left(v_a - \frac{v_b}{2} - \frac{v_c}{2}\right) \tag{B.3}$$

$$v_\beta = k \cdot \left(v_b \cdot \frac{\sqrt{3}}{2} - v_c \cdot \frac{\sqrt{3}}{2}\right) = k \cdot \left(\frac{\sqrt{3}}{2}\right) \cdot (v_b - v_c) \tag{B.4}$$

For amplitude invariance we require that a balanced sinusoidal input $(v_a, v_b, v_c) = V \cdot (cos\theta, cos(\theta - \frac{2\pi}{3}), cos(\theta + \frac{2\pi}{3}))$ produces $|v_{\alpha\beta}| = V$. Substituting:

$$v_\alpha = k \cdot V \cdot \left[cos\theta - \frac{1}{2}cos(\theta - \frac{2\pi}{3}) - \frac{1}{2}cos(\theta + \frac{2\pi}{3})\right] \tag{B.5}$$

$$= k \cdot V \cdot \left[cos\theta - \frac{1}{2} \cdot 2cos\theta \cdot cos\left(\frac{2\pi}{3}\right)\right] \tag{B.6}$$

$$= k \cdot V \cdot \left[cos\theta + \frac{1}{2}cos\,\theta\right] = k \cdot V \cdot \left(\frac{3}{2}\right) \cdot cos\theta \tag{B.7}$$

Setting k = 2/3 yields $v_\alpha = Vcos\theta$, the amplitude-invariant convention. The analogous derivation for v_β yields $v_\beta = Vsin\theta$.

B.2 Park Transformation

The Park transform is a rotation by $-\theta_e$ of the *αβ* vector into a frame whose d-axis coincides with the rotor's permanent-magnet direction. The rotation matrix is:

$$R(-\theta) = [cos\theta \quad sin\,\theta;\ -sin\,\theta \quad cos\,\theta] \tag{B.8}$$

Therefore:

$$v_d = v_\alpha \cdot cos\theta + v_\beta \cdot sin\theta \tag{B.9}$$

$$v_q = -v_\alpha \cdot sin\theta + v_\beta \cdot cos\theta \tag{B.10}$$

The inverse transformation is R(+θ), used in the firmware to convert the dq command voltages back to αβ before SVPWM.

B.3 PMSM Torque Equation

In the dq frame the instantaneous power delivered to the rotor is:

$$P = \left(\frac{3}{2}\right) \cdot \left(v_d \cdot i_d + v_q \cdot i_q\right) \tag{B.11}$$

Substituting the *dq* voltage equations of Section 2.2.3 and isolating the mechanical power-converting terms (those linear in ω_e):

$$P_{mech} = \left(\frac{3}{2}\right) \cdot \omega_e \cdot \left[(L_d - L_q) \cdot i_d \cdot i_q + \lambda_{PM} \cdot i_q\right] \tag{B.12}$$

Since $P_{mech} = \tau \cdot \omega_m$ and$\omega_e = pp \cdot \omega_m$:

$$\tau = \left(\frac{3}{2}\right) \cdot pp \cdot \left[\lambda_{PM} \cdot i_q + \left(L_d - L_q\right) \cdot i_d \cdot i_q\right] \tag{B.13}$$

For a surface PMSM with $L_d \approx L_q$the second (reluctance) term vanishes, leaving:

$$\tau = \left(\frac{3}{2}\right) \cdot pp \cdot \lambda_{PM} \cdot i_q = K_t \cdot i_q \tag{B.14}$$

This is the relationship the impedance controller inverts when converting a commanded torque τ_{des} into a current-loop reference $i_{q_{ref}} = \frac{\tau_{des}}{K_t}$.

B.4 IMC Tuning of the First-Order Current-Loop Plant

After cross-axis decoupling, each axis of the dq current loop sees a first-order plant:

$$G(s) = \frac{1}{L \cdot s + R} \tag{B.15}$$

The Internal Model Control rule specifies a desired closed-loop transfer function:

$$T_{target}(s) = \frac{1}{\tau_{cl} \cdot s + 1}, \tau_{cl} = \frac{1}{\omega_{bw}} \tag{B.16}$$

Solving for the controller C(s) such that the closed-loop transfer function $\frac{G(s)C(s)}{1+G(s)C(s)} = T_{\text{target}}(s)$ yields:

$$C(s) = \frac{T_{\text{target}}(s)}{G(s)\left[1 - T_{\text{target}}(s)\right]} \tag{B.17}$$

$$C(s) = (Ls + R)\frac{\omega_{bw}}{s} \tag{B.18}$$

$$C(s) = L\omega_{bw} + R\frac{\omega_{bw}}{s} \tag{B.19}$$

which is a PI controller with $K_p = L \cdot \omega_{bw}$ and $K_i = R \cdot \omega_{bw}$. These are the gains computed by *tuneCurrentLoops(ω_{bw})* once R and L have been identified.

B.5 Impedance Dynamics for a Single-Degree-of-Freedom Load

Consider the actuator and any rigid load as a single degree of freedom with effective rotational inertia J_{eff}. Under the spring–damper torque law $\tau_{act} = -K \cdot (\theta - \theta_{eq}) - B \cdot \omega$, Newton's second law for rotation gives:

$$J_{eff} \cdot \frac{d^2\theta}{dt^2} = \tau_{ext} + \tau_{act} = \tau_{ext} - K \cdot \left(\theta - \theta_{eq}\right) - B \cdot \left(\frac{d\theta}{dt}\right) \quad \text{(B.20)}$$

With $\tau_{ext} = 0$ (free response after a perturbation), the dynamics reduce to the canonical second-order linear ODE whose characteristic polynomial is:

$$J_{eff} \cdot s^2 + B \cdot s + K = 0 \quad \text{(B.21)}$$

yielding the natural angular frequency $\omega_n = \sqrt{\frac{K}{J_{eff}}}$ and the damping ratio $\zeta = \frac{B}{2 \cdot \sqrt{K \cdot J_{eff}}}$. These quantities determine the transient response of the rendered spring–damper system: under-damped responses ($\zeta < 1$) produce decaying oscillation, critically damped responses ($\zeta = 1$) settle in minimum time without overshoot, and over-damped responses ($\zeta > 1$) settle exponentially without oscillation.

In the velocity-adaptive variant of Section 5.8, the stiffness, damping and torque ceiling are scaled together by the common factor $\sigma = 1 + A \cdot \frac{effort}{v_{ref}}$. Because the natural frequency $\omega_n = \sqrt{\frac{K}{J}}$ and the damping ratio $\zeta = \frac{B}{2\sqrt{KJ}}$ then both scale as $\sqrt{\sigma}$, a more vigorous patient meets a device that is not only stiffer but also faster and more heavily damped: the transient sharpens and its overshoot is reduced as effort rises, rather than retaining a fixed shape. The clinical interpretation is that a stronger patient encounters a firmer, more tightly regulated resistance, while the gentler and more lightly damped baseline dynamics are reserved for slower motion.

Appendix C. Engineering Drawings

This appendix contains dimensioned drawings of the two custom mechanical parts of the testbed and of the assembled actuator module. Both parts were designed in Autodesk Fusion 360 and printed in PLA on a fused-deposition printer; the dimensions given are nominal CAD values, and the as-printed tolerance is approximately ±0.2 mm. All dimensions are in millimeters, and all views follow third-angle projection. The envelopes of the GM4108-120T motor and the AMT103 encoder are shown as nominal outlines only, since neither is a custom part.

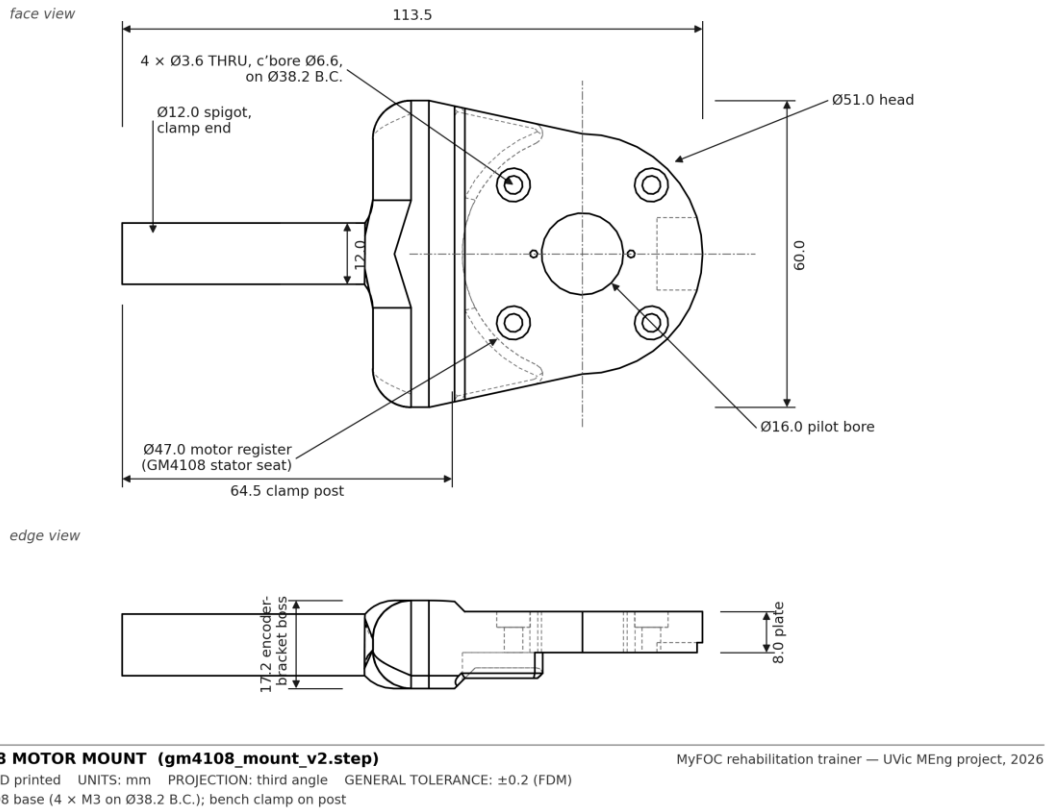
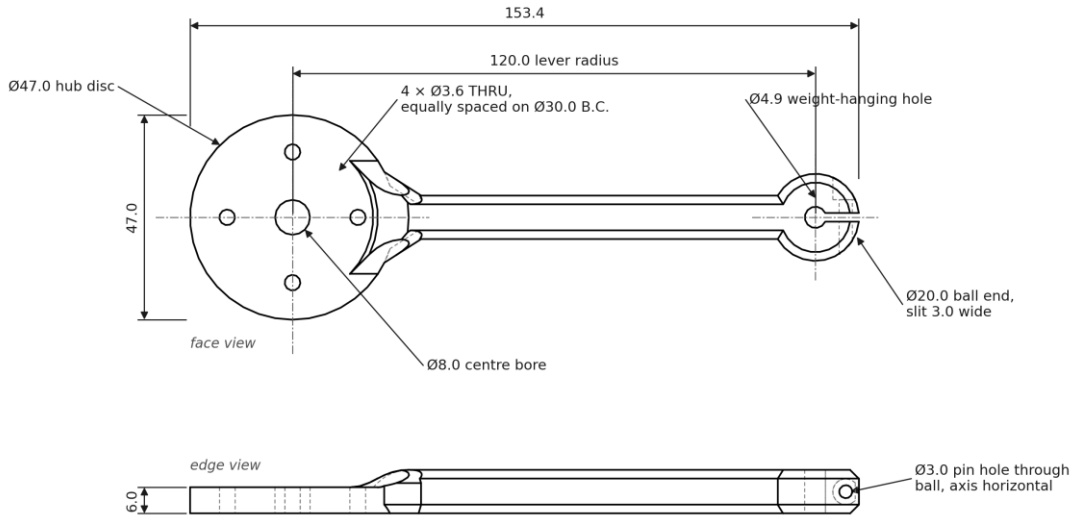


Figure C.1: Motor mount, dimensioned drawing. The Ø47 register seats the GM4108-120T stator, the four M3 clearance holes on a Ø38.2 bolt circle retain it, and the Ø12 post clamps to the bench.

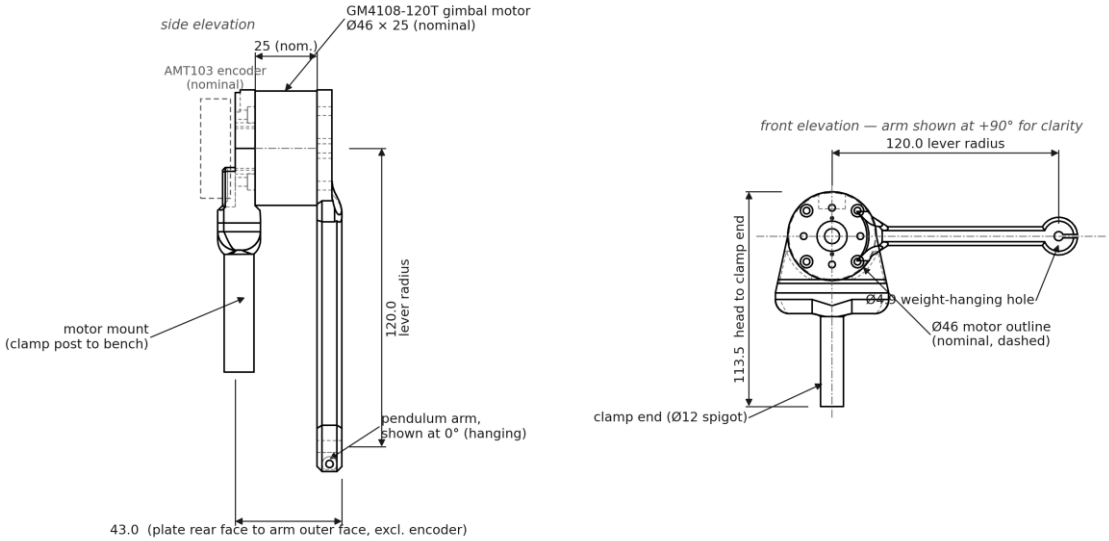


PART: PENDULUM ARM (pendulum_arm_v2.step) MyFOC rehabilitation trainer — UVic MEng project, 2026

MATERIAL: PLA, 3-D printed UNITS: mm PROJECTION: third angle GENERAL TOLERANCE: ±0.2 (FDM)

Interfaces: GM4108 rotor (4 × M3 on Ø30 B.C.); calibration weights at R120

Figure C.2: Pendulum arm, dimensioned drawing. The arm bolts to the motor rotor through four M3 holes on a Ø30 bolt circle; the Ø4.9 hole at a radius of 120 mm carries the calibration weights used in Section 5.6.



PART: ACTUATOR-MODULE ASSEMBLY (mount + GM4108 + pendulum arm) MyFOC rehabilitation trainer — UVic MEng project, 2026

MATERIAL: PLA parts, 3-D printed UNITS: mm PROJECTION: third angle GENERAL TOLERANCE: ±0.2 (FDM)

Motor and encoder envelopes nominal; printed-part dimensions from released CAD

Figure C.3: Actuator-module assembly, side and front elevations. The stack from the mounting plate to the outer face of the arm measures 43 mm, and the 120 mm lever radius defines the torque arm used throughout Chapter 5.