Extending Mission Life of Coastal Profiling Floats through an Improved Pump Motor System

Isaac Wax, Cabrillo Collage

Mentor: Gene Massion

Summer 2025

Keywords: Coastal Profiling Float, FOC, Motor Control, Buoyancy Engine

ABSTRACT

Monitoring coastal regions is essential for understanding ocean health. To address critical data gaps, the Monterey Bay Aquarium Research Institute (MBARI) has developed and deployed Coastal Profiling Floats (CPFs) equipped with chemical sensors. Extending CPF mission life requires improving the energy efficiency of the pump motor system. This project presents an alternative pump motor design achieved by selecting motors whose efficiency curves peak closer to the pump load and by implementing programmable field-oriented control (FOC) drives. The paper details the mechanical, electrical, and software aspects of the project, describes testing, and outlines next steps for integrating these improvements into future CPFs.

INTRODUCTION

To understand the health of the ocean, it is important to monitor coastal regions. Coastal regions contribute nearly half of global new primary production despite only comprising 10% of ocean area.[1] Monitoring coastal regions is important to understanding oxygen dead zones, marine heat waves, harmful algal blooms, managing fisheries and overall ocean health. To address critical data gaps, the MBARI Chemical Sensor Lab has developed CPF that provide a cost-effective solution for collecting long term defensible data across coastal regions. CPFs support standard biogeochemical sensor suites that include CTD, oxygen, pH, nitrate, chlorophyll, and backscatter.[2]

CPFs utilize a buoyancy engine to ascend and descend to complete a data collection profile. A hydraulic pump displaces oil between an internal and external bladder creating a change in the mass distribution of the float. Pumping into the internal bladder increases the density of the float causing it to descend and pumping into the external bladder decreases density causing the float to ascend. This system actively manages profiling depth. CPFs are designed to descend below the euphotic zone and park at a specified depth of up to 350m using a buoyancy engine with displacement of approximately 3.5L which is larger than the ~~0.5L of a standard float. This avoids entrapment in mud, allows profiles through steep pycnoclines and supports return of samples while carrying heavy tooling.[3]

There is a need to increase the energy efficiency of CPFs to allow more profiles per mission. CPFs are powered by ~720Wh battery packs that limit mission life. Increasing energy efficiency of the system will increase profile count and mission life. This means more data collected per deployment resulting in decreased cost per data collected. The pump motor currently utilized results in the largest power loss. It is inefficient and lacks a low-power mode. Modern motor control technologies with different energy curves offer the opportunity to increase CPF efficiency and as a result the number of profiles executed per mission. To explore this approach, a prototype was built to test alternate motor control systems and hardware. Efficiency gain was measured in all states and used to estimate a resulting increase in profile counts and consequent reduction in cost per profile.

MATERIALS AND METHODS

MECHANICAL

A test fixture was constructed using an aluminum extrusion frame supporting an OilDyne hydraulic pump and motor (Figure 1). Custom aluminum brackets were water-jet cut and assembled with FDM-printed spacers for damping. The pump was connected to a 1-gallon plastic reservoir and plumbed to a pressure sensor with an inline regulator. The EVLSERVO1 motor drive and INA229 current sensor were mounted on the fixture, and an ST-Link programmer was used. An Aardvark SPI Host Adapter was used for communication with the drive. The entire assembly was placed in a spill tray, and all tests were conducted at 28.8 V, consistent with CPF average operating voltage.

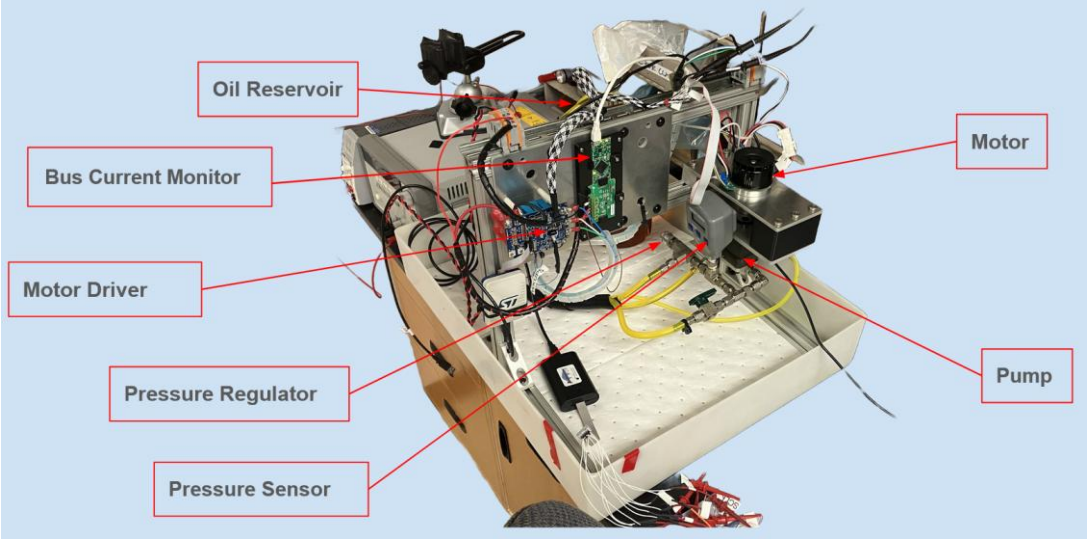


Figure 1. Bench Testing Fixture for CPF Pump System

ELECTRICAL

The fixture electronics included a modified EVLSERVO1 motor driver and an INA229 20-bit current monitoring module populated with a 10 mΩ shunt.[4] The EVLSERVO1 required rework because its default shunts were scaled for ±165 A, far above the ±10 A operating range of the CPF motors. At this range, the maximum drop across the original 500 μΩ resistors was only 5 mV, resulting in just 6 quantization steps on the 12-bit ADC (3.3 V full scale, ±1.65 V input).[5] This lack of resolution made current feedback unusable and caused highly unstable motor control.

To correct this, the shunt resistors were replaced with 5mΩ resistors (a 10× increase), normalizing the measurement range to ±16.5A and providing reliable resolution for the expected operating currents. The EVLSERVO1 also required rework to adapt its default differential Hall sensor inputs for use with single-ended Hall sensors.[6]

For testing, a Maxon EC60 motor was selected, as its efficiency curve peaks near the average load of the OilDyne pump. The EC90 motor currently used in CPFs was not suitable because its efficiency curve peaks well above the pump's maximum load, meaning it always operates far below its peak efficiency (Appendix A).[7]

SOFTWARE

The EVLSERVO1 motor driver was programmed using field-oriented control (FOC) firmware generated with the STMicro Motor Control Workbench (MCWB). MCWB requires input parameters for the motor, encoder, and Hall sensors. For motors in the CPF size range, the built-in motor profiler will often fail, and the parameters must instead be manually characterized using an RLC meter and oscilloscope if not provided in the datasheet. The modulation index was confirmed to be close to 100%, and MCWB then generated C firmware on top of the Motor Control SDK and STM32 HAL.[8]

On top of the auto-generated C firmware, a C++ interface was developed for CPF testing. The STM32 toolchain compiles the MCSDK code strictly as C, which creates type-safety issues where C++ requires explicit casting. Since no compiler flags could resolve this mismatch, a bridge layer was implemented on top of the existing motor control API. The bridge wraps functions that are incompatible with C++ compilation. Although this approach required modification of several auto-generated source files, it allowed the firmware to compile and provided a functional interface.

Table 1. Interface Layer State Descriptions

Run State	→ Default state entered at startup. → Executes commands received from the SPI FIFO, including motor speed control (SETSPEED)
Low Power State	→ Entered on a SETIDLE command. → Analog electronics such as gate drivers are powered down, and the MCU on the STSPIN32G4 is placed in standby mode.[5] → Wakeup occurs either through the real-time clock (RTC) after a timeout or via an external wakeup pin. → On wakeup, the FSM restarts and re-enters into Run Mode.
Error State	→ Entered when error flags are triggered (e.g., over-voltage, over-current, feedback failure). → The FSM attempts to clear errors safely, while maintaining a counter for each event. → If the same error occurs more than five times within one minute, recovery fails and the drive is forced into a permanent standby mode.

A C++ finite state machine (FSM) was implemented on top of the bridge layer (Appendix B). All states publicly inherit from a common base state that defines the

minimum required methods, ensuring consistent behavior and simplifying transitions. The FSM manages driver operation through three primary states seen in Table 1.

An interrupt-driven SPI protocol was implemented to allow the CPF controller (master) to communicate with the motor driver (slave). Messages follow an NMEA-style format with a defined command set (Table 2).[9]

On each interrupt, received bytes are placed into a 256-byte buffer. If the buffer overflows, it is reset. When an endline character is detected, the message is pushed into a FIFO and the buffer is cleared. Messages dequeued from the FIFO are XORed and compared to the checksum before being parsed and passed to a state of the FSM.

Table 2. SPI Communication Commands

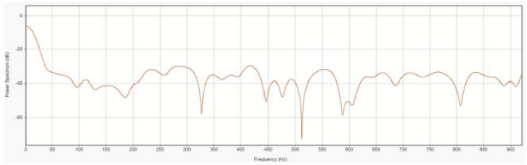
Command	**Example**	**Description**
SETSPEED	$PMC,SETSPEED,<signed int>*CS\r\n	Sets the motor speed and direction (rotations per second).
SETIDLE	$PMC,SETIDLE,<MODE>,<timeout>*CS\r\n	Enters low-power mode. MODE = SHUTDOWN or TIMEOUT. Timeout in seconds. MCU wakes via RTC or external pin.
GETHEALTH	$PMC,GETHEALTH*CS\r\n	Returns an NMEA message with die temperature, power consumption, and error flags (over-voltage, over-current, feedback loss).
GETSPEED	$PMC,GETSPEED*CS\r\n	Returns the current motor speed (rotations per second).

TUNING

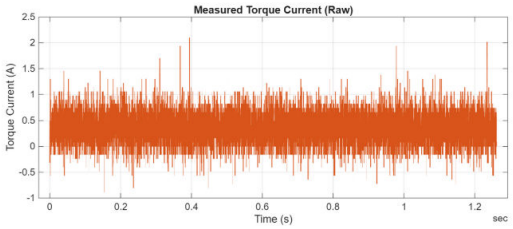
The motor driver implements cascaded PI control loops for current and velocity. Only the torque current loop was tuned, as the flux current loop was not used in this application. Tuning was performed under load with the hydraulic pump attached.

Torque current feedback was read from a 12-bit ADC, which introduced significant noise (Figure 2b). Spectral density analysis using MATLAB Signal Processing Toolbox

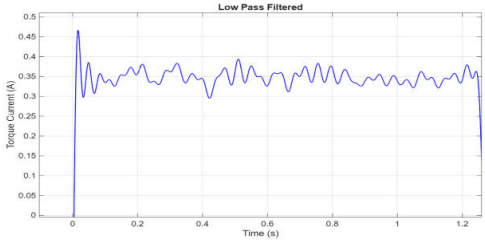
showed that the signal of interest during up/down step tests was below 40 Hz (Figure 2a). A low-pass filter was designed to suppress higher-frequency noise. Filter parameters were iteratively adjusted for each tuning cycle to balance noise reduction with minimal signal distortion (Figure 2c).[11]



(a) Torque Current Power Spectrum



(b) Noisy Torque Current



(c) Low Pass Filtered Torque Current

Figure 2. Torque Current Step Test

DATA AND RESULTS

LOW POWER MODE

Power consumption of the motor driver was measured in low-power mode and compared against the original drive, which consumed ~10 W at idle. Measurements were taken using an INA229EVM 20-bit current-shunt monitor. Several minutes of current data were logged and processed with a filter before averaging to determine steady-state current. A shunt-based measurement was selected over a current clamp to improve sensitivity. Accuracy was validated by comparing results against clamp measurements across resistive loads on an oscilloscope.

Table 3 summarizes idle current and power consumption of the EVLSERVO1 and selected on-board peripherals. The evaluation module includes unused features such as CAN bus transceivers and differential encoder support, which cannot be fully powered down in this configuration. Additionally, the encoder cannot be powered down on the evaluation board.

Reff Designator	Nominal current (mA)	Nominal Voltage	Power (mW)	Part Number	Notes
EVLSERVO1 Full Unit	31	28.8	892.8		In standby mode, no priffrals striped
LED2	2.2	2	4.4		VBUS Status LED
R28	2.2	26.8	58.96		Status LED resistor
U10	23	5	115	ST26C32AB	For differental encoders
U11	60	3.3	198	TCAN330DCNT	CAN Bus
U2		3.3	0	PM8851	
			0	STR485E	We do not need the Z phase of the encoder, so could remove this IC. Quite low power though
Encoder	9.9	28.8	285.12		Add ability to power down Encoder / Hall

Table 3. Unneeded Peripherals on the EVLSERVO1 Breakdown

When the EVLSERVO1 MCU was placed in standby mode, idle power draw decreased by 91%. By removing/powering down unnecessary peripherals, idle power was reduced by 97.9% (Figure 3).

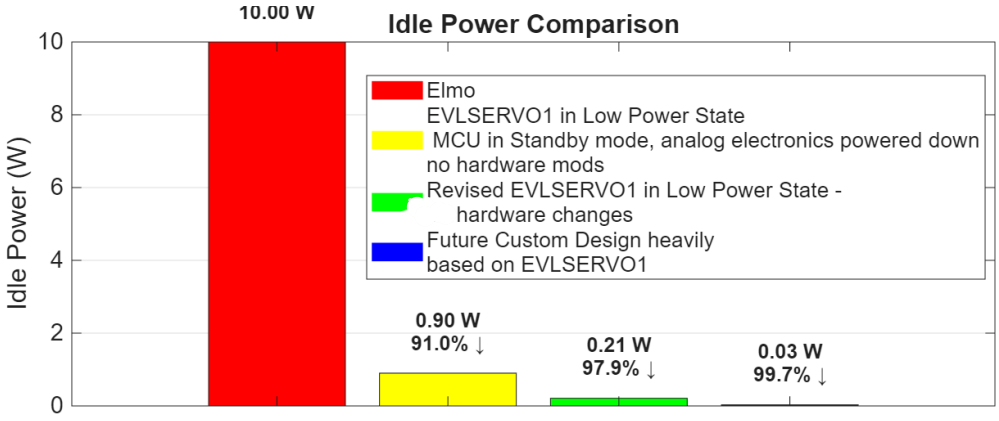


Figure 3. Idle Power Comparison

RUN MODE

To evaluate motor efficiency gains in run mode, the pump was operated at a constant speed, while pressure was varied. CPF missions can profile at pressures of up to 350 dBar.[3] For each test point, the pump was held at a constant pressure for approximately 3 minutes (Figure 4). Current measurements were recorded with the INA229EVM, and steady-state sections were extracted. The data was processed using a boxcar filter and averaged; standard deviations were calculated to assess measurement quality.

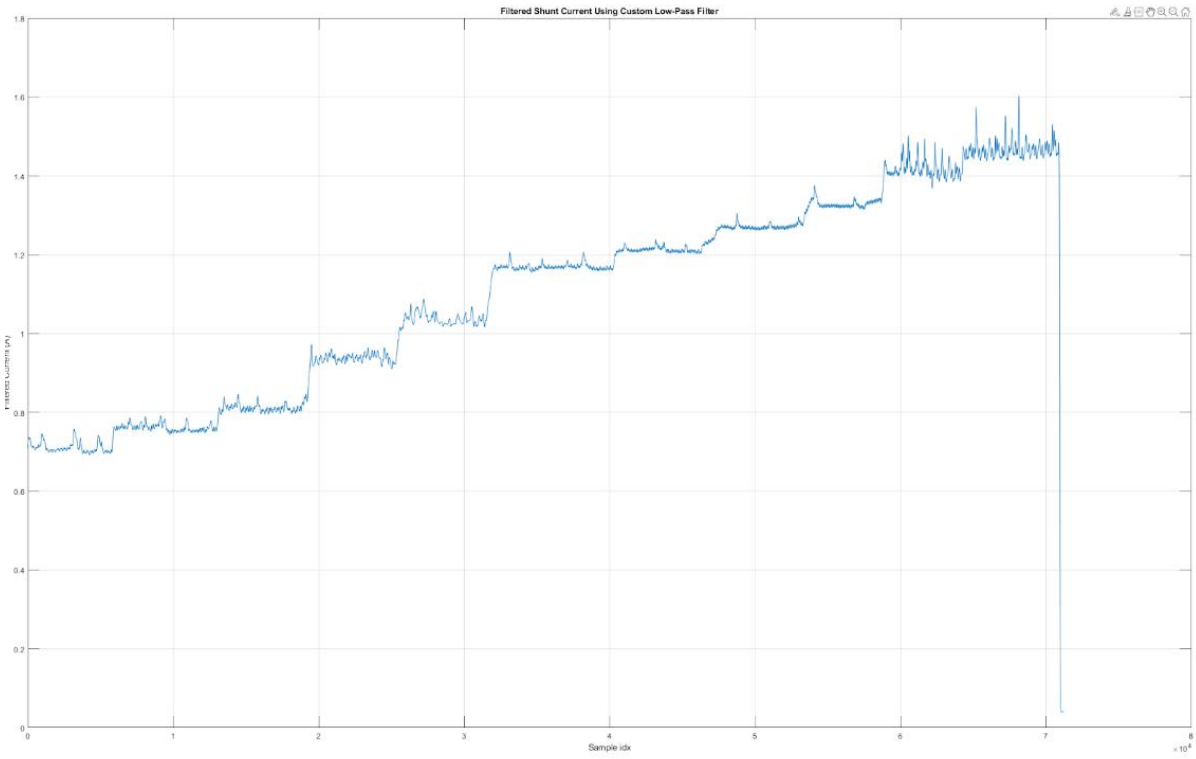


Figure 4. Pressure Test

Testing demonstrated that the alternative drive, combined with a motor whose efficiency curve is better matched to pump loading, provided a 50–65% improvement in efficiency across pressures of interest at 1000 RPM (see data in appendix C).Testing demonstrated that the alternative drive, combined with a motor whose efficiency curve is better matched to pump loading, provided a 50–65% improvement in efficiency across pressures of interest at 1000 RPM (Figure 5).

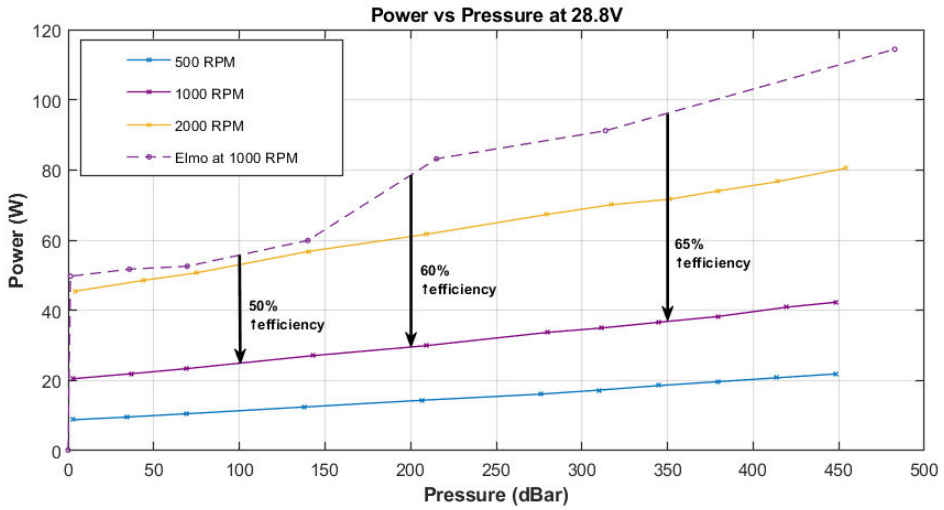


Figure 5. Power vs Pressure at 28.8V

TOTAL IMPROVEMENT

An energy model was constructed to estimate the total power consumed during a CPF profile at a given depth. The model provided a breakdown of where energy is used across motor states (Figure 6).

Figure 6. Power vs Time for a 100m Profile

The results show that run mode dominates overall consumption, meaning that efficiency improvements in this state yield the greatest benefit. The model predicts a 2.4–2.6X increase in profile count at mission-relevant depths (Figure 7).

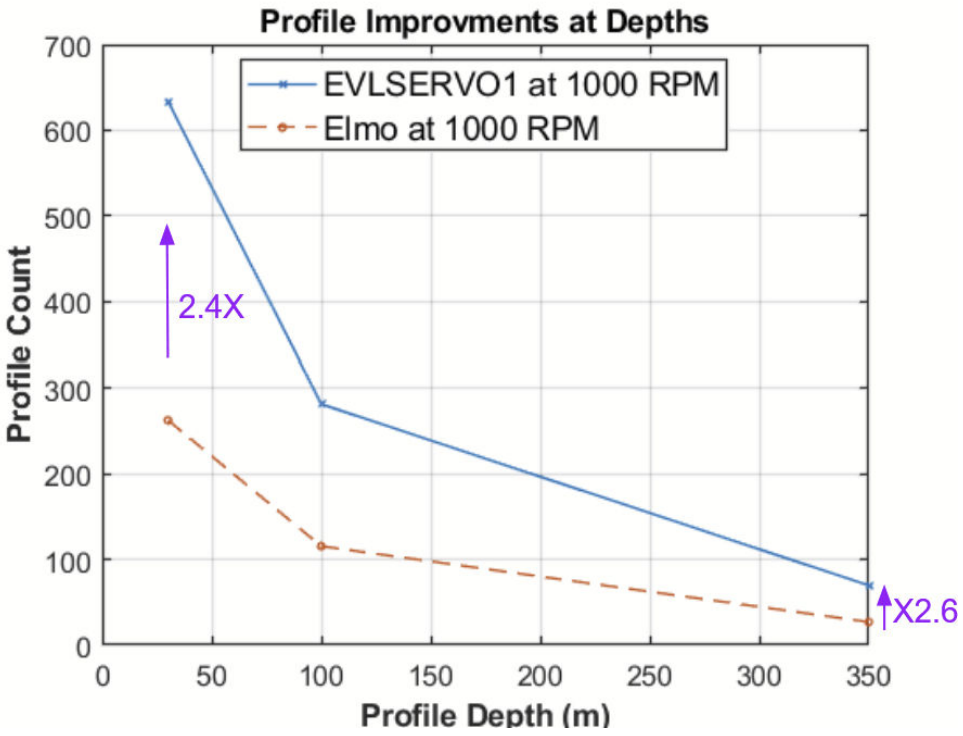


Figure 7. Profile Depth vs. Count (Assuming ~720Wh Battery)

POWER CACULATION VALIDATION

A calculation of total power from torque current, phase voltages, and electrical angle was performed to validate Motor Pilot GUI data.[13] The process applied inverse Park and Clarke transformations, followed by a power calculation.[12] The calculated values did not align with Motor Pilot outputs, indicating inconsistencies. This is most likely due to an error in implementation of the calculation rather than an issue with the Motor Pilot data itself.

Equation 1. Calculating Power from Torque Current

Inverse Park (dq → αβ) and $i_d \approx 0$.

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix}, \qquad i_d \approx 0 \;\Rightarrow\; i_\alpha = -i_q \sin\theta, \quad i_\beta = i_q \cos\theta.$$

(These are the two orthogonal stator-fixed/Clarke components.)

Inverse Clarke ($\alpha\beta \to abc$, $i_0 = 0$).

$$i_b = -\tfrac{1}{2} i_\alpha + \tfrac{\sqrt{3}}{2} i_\beta, \qquad i_c = -\tfrac{1}{2} i_\alpha - \tfrac{\sqrt{3}}{2} i_\beta, \qquad \text{KCL: } i_a = -(i_b + i_c) = i_\alpha$$

Power

$$p_{ac}(t) = v_a i_a + v_b i_b + v_c i_c, \qquad V_{dc} i_{dc} = p_{ac} + P_{\text{lossIdle}} + p_{\text{lossSwitching}}$$

DISCUSSION AND NEXT STEPS

While bench tests demonstrated promising efficiency gains, further validation is required. In particular, repeated trials are needed to confirm reliability and identify edge cases, such as starting the OilDyne pump into back pressure after long idle periods. Cold pumps appear to require higher startup torque than warm ones, and this behavior was not fully evaluated. Integration of the motor and drive into a CPF, followed by tank and eventually at-sea testing, will be critical to uncover additional issues.

Another key motivation for exploring the EVLSERVO1 was its support for regenerative braking. The hardware includes a dedicated regeneration circuit, which could potentially harvest back EMF during ascent and recharge the battery pack. This capability was not tested during this project and remains an important next step.

The EVLSERVO1 drive is designed for high-power motors (~4 kW) and is oversized for this application. As described above, it required modification to operate effectively with lower-current motors. A custom driver could be made lighter, more compact, and more efficient by using appropriately scaled current shunts and eliminating unused peripherals. Several ICs on the evaluation board include low-power/shutdown pins, but these were not connected, leaving no way to power down these ICs. Adding switching for powering down encoders and Hall sensors would further reduce idle draw.

Software development also remains incomplete. While a functional C++ bridge and finite state machine were implemented, the interface is not yet production-ready. Unit testing must be added, particularly for the error handling state. Error handling should include logging, message transmission to the CPF master, and more robust recovery strategies. The current bridge layer, which wraps the autogenerated MCSDK code, is functional but messy and should eventually be replaced with a cleaner solution. In retrospect, implementing the interface layer directly in C would have been chosen.

Further testing is needed to validate the efficiency results. The first attempt to calculate bus power from torque current, phase voltages, and electrical angle did not match the values reported by Motor Pilot. This mismatch is almost certainly due to an error in the calculations rather than a problem with the Motor Pilot data. It also remains unclear how

much of the efficiency gain comes from motor choice (EC60 vs. EC90) versus the drive. Repeating the tests with the same setup but using an Elmo drive would help separate these effects. We speculate that the motor may be the main contributor to the increased efficiency, though drive settings could also play a role.

CONCLUSIONS

The improved pump motor system described in this paper demonstrated significant energy savings and, after further validation, could be implemented in CPFs to substantially increase profile count. This improvement would lower CPF operating costs and extend mission duration. In addition, it lays the groundwork for integrating and testing regenerative braking in CPFs.

ACKNOWLEDGEMENTS

I would like to thank Gene Massion for his mentorship, guidance, and for generously sharing his wealth of knowledge with me throughout this internship. I am also grateful to the intern coordinators, George Matsumoto and Megan Bassett, for their leadership of the internship program and their support over the summer. Special thanks to the MBARI electronics technicians Chris Beebe, Jim Montgomery, James McClure, and Jose Rosal for their patience and expertise. I also thank Roman Marin for support in manufacturing parts, Scott Jensen and Brett Hobson for their generous donation of materials, and Erik Trauschke for invaluable insights on current sensing. The MBARI Summer Internship Program is generously supported through a gift from the Dean and Helen Witter Family Fund and the Rentschler Family Fund in memory of former MBARI board member Frank Roberts (1920–2019) and by the David and Lucile Packard Foundation. Additional funding is provided by the Maxwell/Hanrahan Foundation.

References:

[1]K. Fennel et al., “Carbon cycling in the North American coastal ocean: a synthesis,” Biogeosciences, vol. 16, no. 6, pp. 1281–1304, Mar. 2019, doi: https://doi.org/10.5194/bg-16-1281-2019.

[2]“‘Argo Float Program.’ n.d. Argo. University of San Diego, Scripps Institute of Oceanography.,” 2025. https://argo.ucsd.edu/

[3]visceral_dev_admin, “New coastal profiling floats for diagnosing ocean health,” MBARI, Feb. 06, 2020. https://www.mbari.org/news/new-coastal-profiling-floats-for-diagnosing-ocean-health/

[4]“INA229 data sheet, product information and support | TI.com,” Ti.com, 2020. https://www.ti.com/product/INA229 (accessed Aug. 25, 2025).

[5]“STSPIN32G4 | Product - STMicroelectronics,” STMicroelectronics, 2025. https://www.st.com/en/motor-drivers/stspin32g4.html#documentation (accessed Aug. 25, 2025).

[6]“EVLSERVO1 | Product - STMicroelectronics,” STMicroelectronics, 2024. https://www.st.com/en/evaluation-tools/evlservo1.html#documentation (accessed Aug. 25, 2025).

[7]Selection of DC Drives, “Selection of DC Drives,” FlippingBook, 2025. https://online.flippingbook.com/view/72734/30/ (accessed Aug. 25, 2025).

[8]“X-CUBE-MCSDK | Product - STMicroelectronics,” STMicroelectronics, 2021. https://www.st.com/en/embedded-software/x-cube-mcsdk.html#documentation (accessed Aug. 25, 2025).

[9]“NMEA 0183,” Wikipedia, Apr. 17, 2020. https://en.wikipedia.org/wiki/NMEA_0183

[10]STMicroelectronics, “Motor Control Part3 - 8 General and analytic PI regulator tuning,” YouTube, Dec. 03, 2018. https://www.youtube.com/watch?v=r7MsMJSvDaI (accessed Aug. 25, 2025).

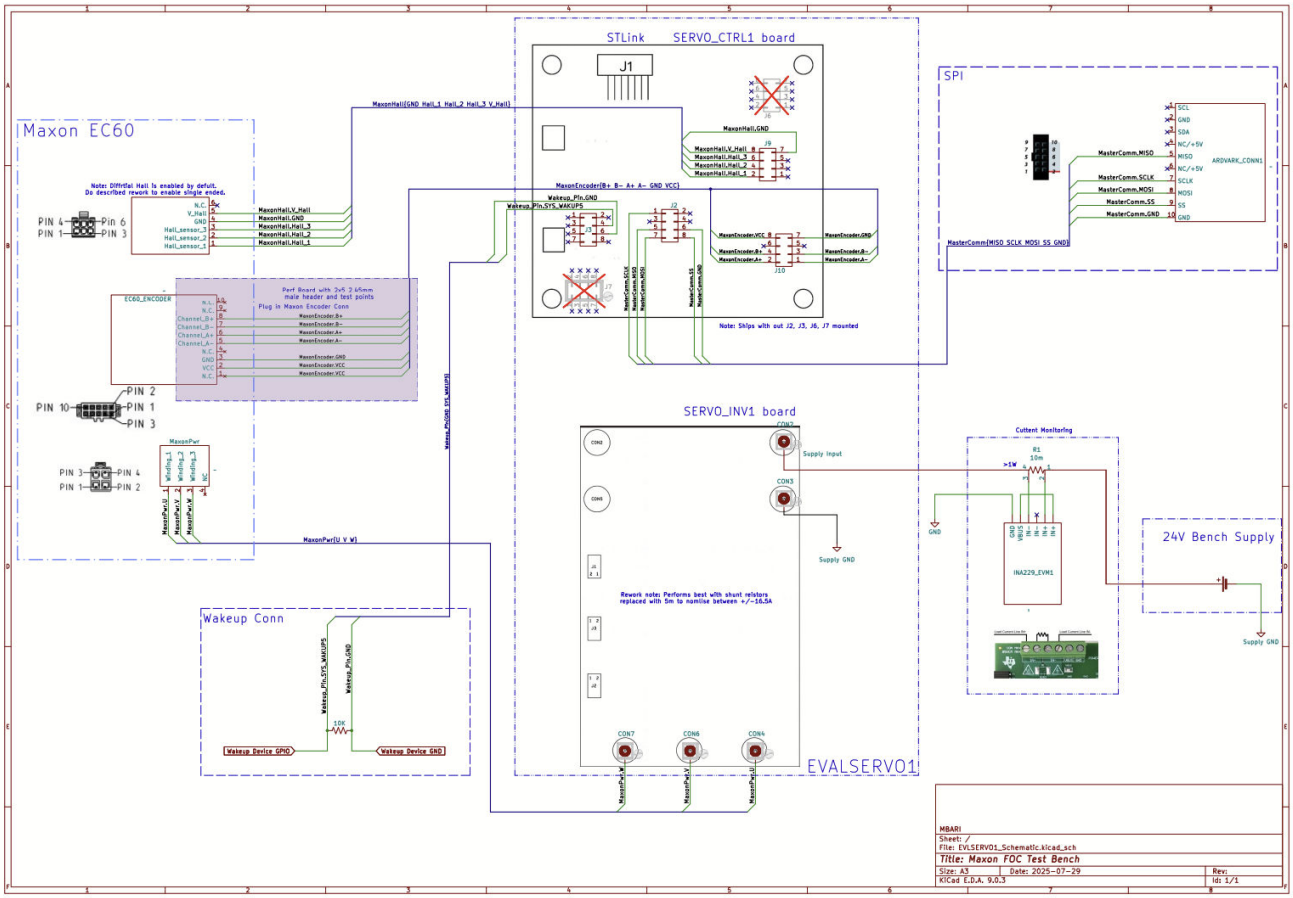
[11]“Get Started with Signal Processing Toolbox,” www.mathworks.com. https://www.mathworks.com/help/signal/getting-started-with-signal-processing-toolbox.html

[12]“Field-Oriented Control (FOC) - MATLAB & Simulink,” www.mathworks.com. https://www.mathworks.com/help/mcb/gs/implement-motor-speed-control-by-using-field-oriented-control-foc.html

[13]“STM32 MC Motor Pilot - Start-up guide - stm32mcu,” St.com, 2025. https://wiki.st.com/stm32mcu/wiki/STM32MotorControl:STM32_MC_Motor_Pilot_-_Start-up_guide (accessed Aug. 25, 2025).

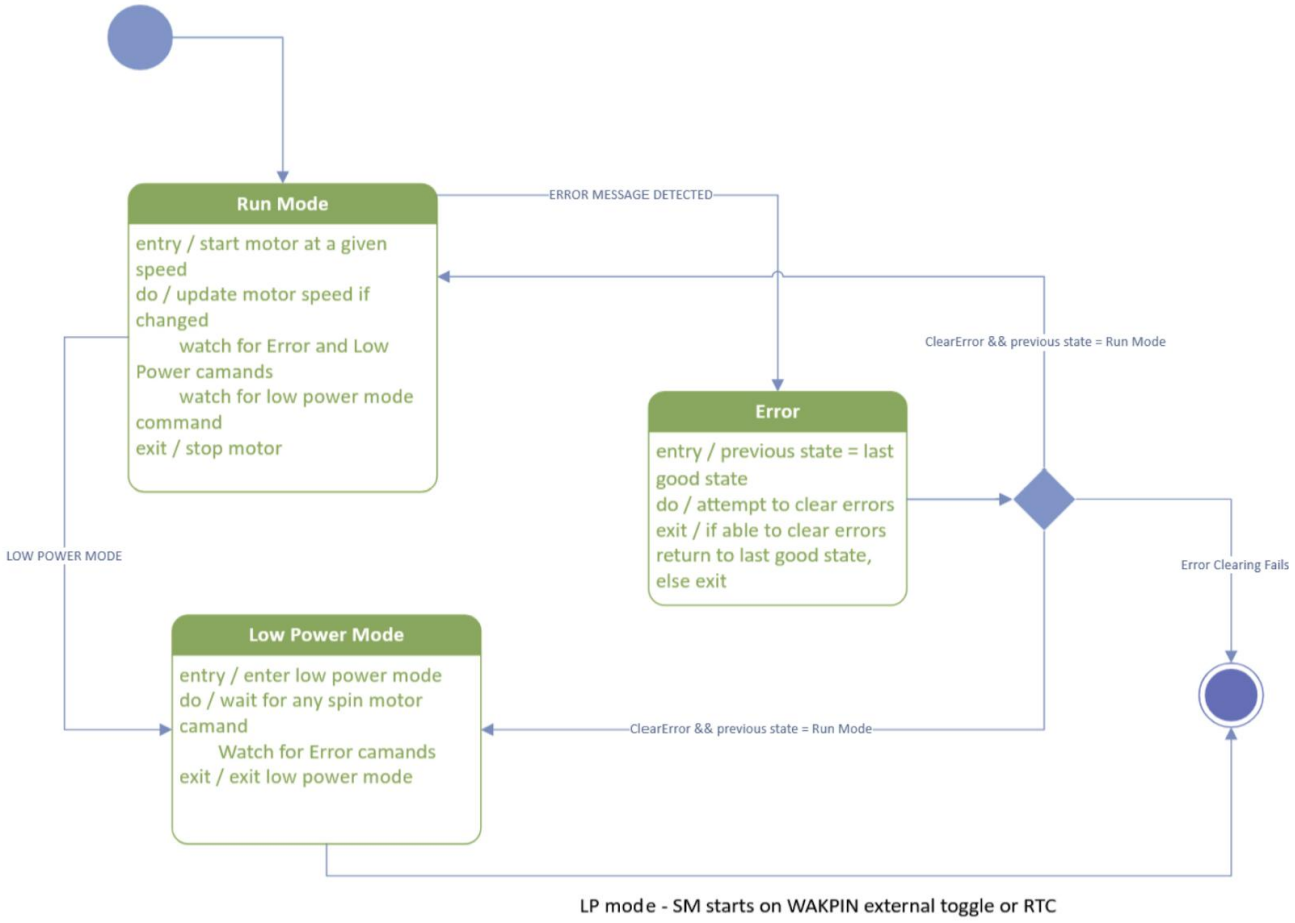
APPENDIX A

Schematic of bench testing setup.



APPENDIX B

UML State Machine Diagram of interface layer software.



APPENDIX C

Pump pressure test with EVLSERVO1 and Maxon EC60 at 1000 RPM.

Speed (RPM)	Pressure (PSI)	Voltage (V)	Current approx. (A)	Mean of Shunt Current (A)	Shunt Stdev (A)
1002	4	28.822	0.72	0.711211	0.013631
1002	54	28.822	0.77	0.761368	0.009825
1002	100	28.822	0.83	0.8106	0.009344
1002	207	28.822	0.96	0.939189	0.009519
1002	303	28.822	1.06	1.03798	0.014897
1002	406	28.822	1.18	1.169447	0.008125
1002	452	28.822	1.23	1.212947	0.006343
1002	500	28.822	1.29	1.269518	0.004563
1002	550	28.822	1.35	1.326354	0.00641
1002	608	28.822	1.46	1.418153	0.023428
1002	650	28.822	1.51	1.467713	0.023573