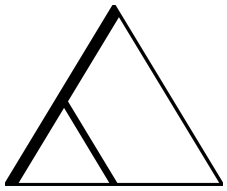


OTO

“Fully oto-matic.”

02-09-2025

Tyler Bovenzi

Liam Duckworth: liduckwo@calpoly.edu

Franky Izaraba: firazaba@calpoly.edu

Alexander Herman: aherma04@calpoly.edu

Introduction

Project Overview

The goal of this project is to create a single-package solution for precisely controlled motorized movement. The solution takes the form of a smart motor or “Otomotor” (pronounced /auto-motor/). Otomotors encapsulate all levels of precision motor control. The motor assembly includes all the hardware to physically power and manage the dynamics of the motor along with wired and wireless interfaces. Alongside the physical motor, there is a software API that provides both simple and advanced hardware-abstracted methods to control the motor and customize its behavior and dynamics. Together these provide a comprehensive and powerful package for precision movement and actuation, suitable for a wide variety of use cases and user backgrounds.

Client/Customers

Client

The client for this project is Tyler Bovenzi. Tyler is a Cal Poly CPE alumnus and previous capstone student. He now works in the aerospace industry and has been working on these motors as a side project. He provided us with the working, breadboarded prototype of the motor that he had been working on prior to initiating the capstone project. He has significant technical experience both on the hardware and the software side of the project and has been invaluable in pinning down continued design criteria and reviewing our work.

Customers

We envision three main customer archetypes for Otomotors: hobbyists, educators, and industry engineers. The hobbyist archetype covers customers who are amateur engineers who create simpler projects and either don’t have or don’t want to exercise a deep understanding of control theory. The hobbyist wants a simple and streamlined implementation combined with high-level flexibility. The educator archetype covers customers who themselves may have deep knowledge of control theory but are likely interfacing with people who don’t know as much. They may be creating a simple demonstration project to show off, which would also drive requirements for a streamlined integration process, or they may be creating a deeper experience for labs to teach control theory itself, which would drive requirements for much lower-level customization capabilities like the ability to manually set dynamics coefficients. The industry engineer archetype covers customers who are designing potentially extremely

complex projects where many systems must work in tandem at high reliability and precision under tight requirements. Industry engineers have a deep understanding of their projects and control theory, so they likely have existing system specifications and control designs. They may be prototyping a system or implementing a final product. This archetype drives requirements for streamlined integration and also deep customizability.

Stakeholders

Cal Poly San Luis Obispo and its Computer Engineering department have a vested interest in the success of the project. Our success bolsters Cal Poly's reputation and strengthens its connections to industry and alumni. Our success also attracts more industry partners for the Computer Engineering department's Capstone projects which allows Capstone to continue as a required class, improving the quality and readiness of graduates of the program. Additionally, our client Tyler Bovenzi has a vested interest in our success as both a Cal Poly partner and industry developer who would seek to develop this project further assuming it achieves moderate success.

Framed Insights and Opportunities

Our initial discussions with the client were focused on developing a useful framework for development and good ways to maintain connection and deliver project updates. In the early weeks of development, he communicated the expectations for the project. He informed us that based on experience and time, he would reduce the scope of the project to focus largely on a successful motor with an expanded set of features, and scrapping plans for an IOS app.

The initial expectations for the project were as follows:

- Custom PCB that integrates BLE, the RP2040, and a motor driver.
- Software PID Control
- Ability to communicate bi-directionally
- Modularity and flexibility for integration with many types of motors.
- API Framework developed outlining communication with multiple motors.

He additionally explained his intentions for a custom PCB that would integrate the existing motor control loop directly with the communication and control software. While challenging, he also presented us with a working prototype and codebase to facilitate our production and save time during development.

His expectations for the project, while narrowed, remained broad. He explained to us that he wanted to see many iterations of the PCB, with initial designs focusing on functionality, and later designs focusing on minimizing footprint while preserving more complex features like antenna traces. Further, he explained that ideally, a future version of the product would fully incorporate the existing rotary encoder hardware and exist as a single functional PCB. This would make the device standalone and would be much more marketable as it could then be sold as an attachment to any motor, effectively turning any motor into an Otomotor. Generally, the message was to create a feature-dense product that would be flexible and adaptable for a wide range of applications. This seemed very reasonable and we began working right away to research and understand the device. This involved an in-depth understanding of each component and the code structure.

Quickly after starting development, we encountered our first design decision, voltage conversion. Recognizing that the functional board would need to be powered off of the 12V+ supplied to the motor (instead of the 5V USB connection on the prototype) we added a buck converter to the design. Seeking design flexibility and robustness we used existing power design tools to engineer a 5-40V -> 3.3V regulator. As development continues we expect to encounter many more design decisions. We will coordinate with Tyler and our advisor to best navigate them.

Goals and Objectives

Throughout this project, our team looks to build technical expertise, foster collaboration, and ensure the timely delivery of a high-quality solution to an emerging problem. In broad terms, we will create a motor with a large array of features that can be used in almost any circumstance by almost any skill level. To achieve this, we will maintain constant communication with our client to ensure alignment with project specifications, building a professional relationship that acknowledges the limited time our client has to work on this project. We will also need to stay extremely organized and ahead of deadlines. To do this we are leaning on the productivity software Notion which allows easy organization of Oto materials, reports, meeting notes, etc. At the end of the project, we expect our Notion to be worthy of showing to someone interested in our project such as a future employer, showcasing our design process, goals, and outcomes.

Outcomes and Deliverables

At the end of this project, we plan to have multiple (up to 7 wireless or 40 wired) fully functional (and pre-programmed) smart motor boards capable of connecting via I2C

lines to a central controller. We define fully functional to mean that the device fully satisfies all non-stretch hardware and software engineering requirements laid out in **Table 3**. Additionally, the final product will be well documented, with example schematics and usage diagrams. The high-level API will have clearly defined features that allow a user to take full advantage of the motor hardware while being easy to integrate into existing devices. The product will be delivered as a PCB containing all the necessary components with minimal footprint and ports for all necessary connections. The PCB will be packaged inside a 3D-printed housing and sit perpendicular to the motor drive shaft.

To demonstrate the effectiveness of our product we will show an example use case utilizing three motors mounted to control an existing robotic arm. We will integrate our API into existing control logic and demonstrate its usage with a preprogrammed display of features or interactive peripherals like sliders or pots that move the arm.

Further efforts to improve the design might look to reduce the PCB footprint and add additional software features like auto-calibration and BLE support. To make the device readily compatible with larger projects like robotic arms and camera gimbals, advanced serial synchronization features could be developed for improved coordination and computational efficiency.

Background Research

To understand the product, it is important to understand how the system works and what components are needed for its function. The hardware consists of six main components: a DC motor with an attached quadrature encoder, an RP2040 microcontroller, an H-Bridge motor driver, a current and voltage sensor, and a Bluetooth chip. Together they create a system known as a smart motor, where the code running on the chip intelligently determines the needed speed and acceleration to move the motor at a desired speed or position.

How exactly that control happens is a simple application of control theory. Our design uses a method known as PID (Positional, Integral, Derivative) tuning. This method essentially uses a formula (See eq. (1) in appx.) to determine the output command to send to the motor. The equation uses only the change in the position from one reading to the next, but is able to estimate velocity through carefully timed execution (every 1ms is standard).

Existing Work

Our work builds heavily off the existing work supplied to us by the client. Shown below in **Fig. 1** is a general smart motor schematic, and **Fig. 2** shows the smart motor prototype that we received from the client.

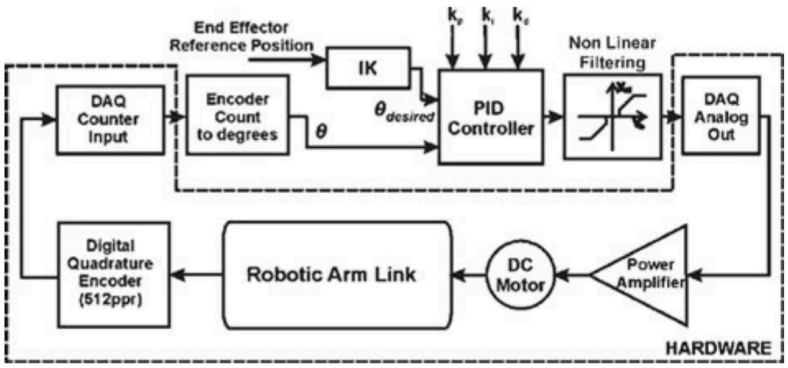


Fig 1: Motor control loop roughly implemented by the prototype and more fully in the final product.

In our design, the Power Amplifier is the H-Bridge motor driver and the Quadrature Encoder is built into the DC motor. The DAQ counter is handled in software in a dedicated core on the RP2040.

The working prototype is a minimally functional device that uses the 25D Metal Gearmotor, an Adafruit Motor Controller, and an RP2040 board equipped with some simple GPIO, USB, and flash. This is fundamentally very similar to the Otomotors that we will build, however our design will be packaged entirely on one PCB.

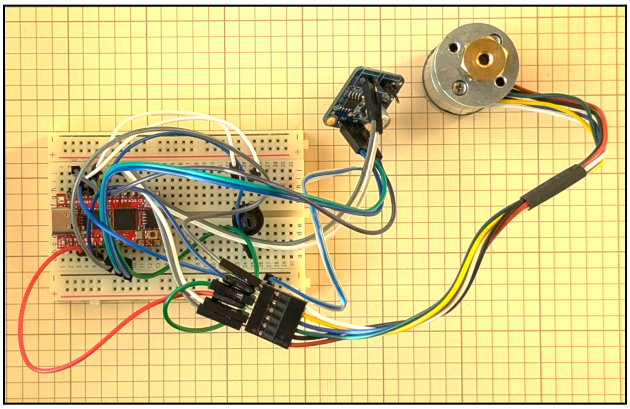


Fig 2: Delivered Motor Prototype. Notice that the prototype does not implement any hardware for voltage stepdown or bluetooth functionality.

Competitor Products

The RoboClaw is a versatile motor driver with a focus on streamlining the user experience by setting up multiple motors. It handles many similar functions as our product, such as voltage and current sensing, PID tuning and control, and bus-level communication. The roboclaw is capable of supplying power to two 34-volt motors with a current as high as 7 amps. This spec makes the roboclaw a strong contender for most robotics applications where high torque is required for the control of DC brushless motors. The unit cost of a roboclaw is 79.95 for the 7A version or up to 99.95 for the 15A. These high prices are comparably competitive in the motor controller market but are also indicative of superior quality, reliability, and integrated features such as their motion studio software designed to make multiple motor control as effortless as possible

In practice, this design outperforms ours on most metrics but not all. Primarily, the roboclaw costs significantly more (about twice as much as the Otomotor). We also anticipate that our Rev 2 board will be smaller than the RoboClaw and draw much less current. Additionally, our hardware is much simpler, which increases our reliability and is better for learning environments.

Specifications and Requirements

Customer Specifications

Archetype:	Hobbyist	Educator	Industry Engineer
Profile:	• Limited resources • Abundant time • Desire to learn or tinker • Low control theory knowledge • Personal projects	• Limited resources • Limited time • Desire for learning experience • Highly variable control theory knowledge (student vs teacher; professor vs camp counselor) • Demos and labs	• Abundant resources • Limited time • High control theory knowledge • Tight specifications • Complex systems
Needs:	• Simple interface • Streamlined integration • Surface level Customization • Low cost	• Simple interface • Streamlined integration • Deep customization • Flexibility • Low cost • Ruggedness	• Streamlined integration • Extremely deep customization • Precision • Ruggedness • Reliability

Table 1: Customer profiles and needs

The needs of the three archetypes are based on their profile and potential use cases. The profiles generally present escalating knowledge of control theory and depth of customization.

The hobbyist, with their limited control theory knowledge, is most likely to want a product that has some extremely high-level features. Their use cases include simpler projects like camera sliders, smart home projects like blinds and dispensers, art projects, or even at the more complex end, a robotic arm or DIY 3D printer. They are likely not concerned with the inner workings of the dynamical system, and if they are, they would be looking for a simple method to change certain high-level parameters like

speed slew in the case of a camera slider for example. Hobbyists also have limited resources, making it important to keep our product's cost low.

The educator has the broadest needs because to cater to them we also need to cater to their students. Their use cases are likely various demos and labs. These could take many forms, such as a robotics kit, a programming exercise, or controlling a ready-made device. However, there are massive differences in the level of an educator, from a summer camp counselor to a university controls professor, and thus in the depth of these demos and labs. This broad spectrum means that the educator overlaps with the hobbyist in the need for simple interfaces for students but also with an industry engineer in the need for deep customization and low-level parameterization to increase the Otomotor's capacity as a teaching tool. The ruggedness of both the hardware and the software is also essential to educators since these systems are likely to be in a high-touch role working with students who are liable to mistreat and mis-program the systems. Educators are likely to have the most limited resources of all three archetypes since they need high volume while often on a personal budget, making a low-cost high-resilience system particularly enticing.

The industry engineer has the most technical needs. Their use cases are likely in industrial robotics like precision manipulator arms, conveyor belts, or other logistics devices. They also may want to use these in prototyping their own marketable product. In all of these cases, the engineer likely has deep knowledge of the desired dynamics of the system they are designing and the requirements they have to meet. It's likely that these specifications are extremely tight, and require deep customization of the system, down to PID parameters, tuning methods, and force limits. The systems they make also need to be extremely rugged and reliable, capable of lasting under years of constant use and with safe failure modes. Many of these use cases also require either multiple motors working in tandem or as an integrated part of a larger system, driving the need for a comprehensive and simple interface for software control and hardware integration.

Engineering Specifications

Hardware	Software

● Connections are locking or soldered
● Programmable via USB-C
● I^2C serial communication with a reset button for address assignment
● Input voltage range 5V-40V
	● PID control loop
	● I2C Serial Communication
	● High-level API
	  ○ Speed control
	  ○ Position control

● RP2040 and other non-motor components draw <500mA ● Motor executes commands immediately (<5ms) ● Power sensing capabilities ● Small electronics footprint ● BLE wireless communication (stretch)	○ Configurable coefficients ○ Motor selection, pairing, and identification. ○ Auto Tuning ○ Slew rate control (stretch) ○ Motor profiles (stretch) ● BLE compatibility (stretch) ● Telemetry fetching and interpretation

Table 2: Engineering Specifications for Hardware and Software

This list of engineering specifications is both handed down from the client and derived from the customer's needs. The client's desires lined up very well with potential customer needs. Nearly all the engineering specifications that would have been derived from the customer's needs were addressed in the client's original request.

The hardware specifications are all in support of the customer's needs for a streamlined integration process and the necessary software capabilities. The MCU, motor driver, and single connector and power source specification cover the need for the Otomotor to encapsulate as much of the system as possible without impinging on the potential use cases. Power sensing derives from potential force sensing needs for a user, and provides capability in delicate precision machinery. I^2C serial communication was chosen to support addressed, bidirectional communication in a large, wired system while minimizing wire count. BLE is a stretch goal and is aimed at improving the capability of smart-home applications and creating a wireless option for connectivity.

The software specifications are in support of lowering the controls knowledge entry point while still providing the flexibility that a fully-fledged controls engineer might require. Abstracting the PID control loop to a high-level API removes a significant knowledge barrier and allows high-precision control with low technical and computing overhead. The biggest trouble with PID is tuning it. An experienced engineer would have no problem with this, but a hobbyist might. Even if they don't need the precision a properly tuned system provides, automatic tuning would be a great feature to reach for. Telemetry fetching and interpretation will allow Otomotors to be truly smart, providing real-time performance metrics for testing prototypes and maintenance of existing systems.

Customer and Engineering Requirements

Spec #	Customer	Engineering Parameter	Requirement	Tolerance	Risk	Compliance
1	Small	Hardware size	1” x 4”	+/- .1”	M	I, A
2	Cheap	Production Cost	>60 $/unit	+/- 10%	M	I
3	Hardware Integration	Ease of Use	4-wire controller connection, 5-wire motor connection	N/A	L	I
4	Software Integration	Intuitiveness	Straightforward basic motor control	N/A	H	A
5	Customization	API Functionality	Settable PID coefficients	N/A	L	A
6	Ruggedness	Reliability	>1mA current in reverse polarity	+/- 10%	L	T
7	Performance	PID Accuracy	>5% error from simulation	+/-5%	H	I, T, A
8	Low power	Hardware power consumption	>500mW peak	+/- 10%	L	T
9	Precise positioning	Positional control	0% steady state position step error	+/- 1%	L	T, A
10	Precise velocity	Velocity control	0% steady state velocity step (ramp position) error	+/- 1%	L	T, A
11	Scalability	Multi-motor control	I2C multi-pairing	N/A	L	T, A

Table 3: Customer and Engineering Requirements

Design Development

Our design process began by researching components and deciding which would be used in our project. We landed on using the Pico RP2040 as our development board and CYW43439 as our Bluetooth chip (Murata 1YN module). Once we finalized these component selections, we moved forward with developing the schematic for our PCB design. This step involved carefully mapping out connections, power requirements, and signal routing to ensure a well-optimized and functional circuit. Simultaneously, we set up the prototype development environment on our systems to facilitate software development. This process included configuring the necessary toolchains, debugging tools, and firmware environments to ensure a smooth workflow as we progressed.

Software Development

Using a modified version of the provided control software, we developed a working set_angle function which demonstrated the feasibility of powering and controlling the device using our test setup and software development environment. However, further conversation with our advisor Professor Murray showed that we needed a more robust approach. This is where the PID control loop was explained and is what he ultimately implemented as the foundation of our motor control. Shown below in fig _ and in code in Appendix 2.1, our PID control is a simple closed loop system that operates off of 1ms hardware interrupts.

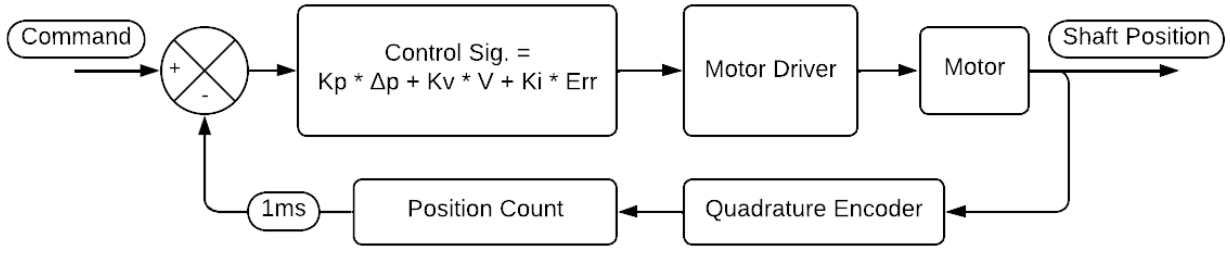


Fig 3*:* The controlling signal sent to the motor is a PWM value [-256, 256]. This value is determined based on the PID coefficients which are set manually or experimentally determined through a tuning routine.

I2C communication was the next goal, and we developed a preliminary packet structure and sent simple commands from a master device to the slave motor controller. The packet structure contained the type, byte size, and data that would be sent. This allows the slave to understand the data received and apply it to the motor

accordingly. After a basic structure was established for the communication, we focused our efforts on multi-device pairing. This focuses on configuring each motor with a unique address to continue constant communication. To do this, the master performs a bus scan to locate an available address for a new motor and sends this off to a default address for the new slave to reconfigure itself for further communication.

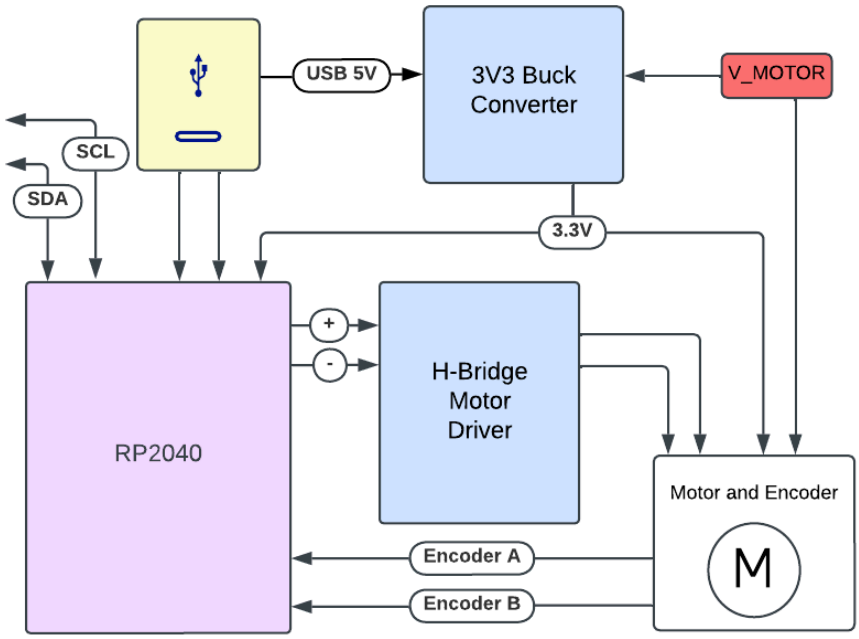


Fig 4: Preliminary Hardware Block Diagram

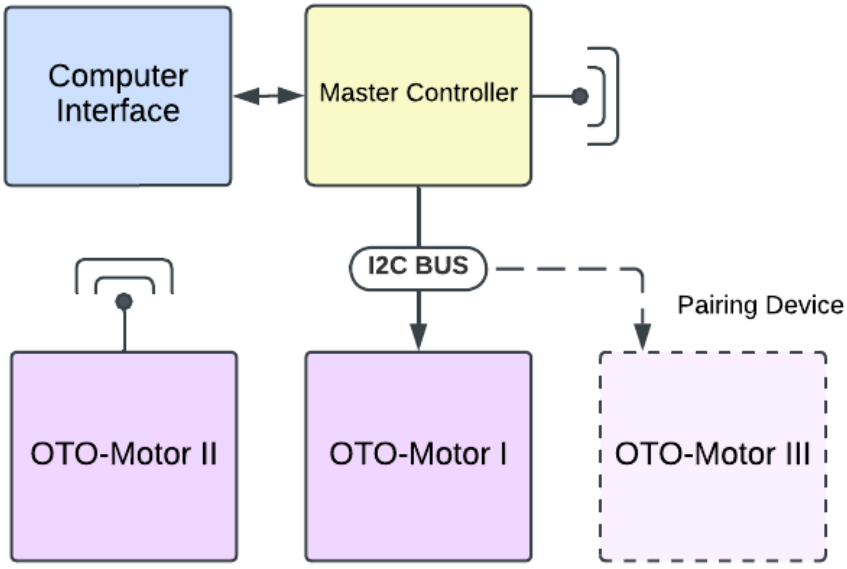


Fig 5: Preliminary Use Case System Schematic

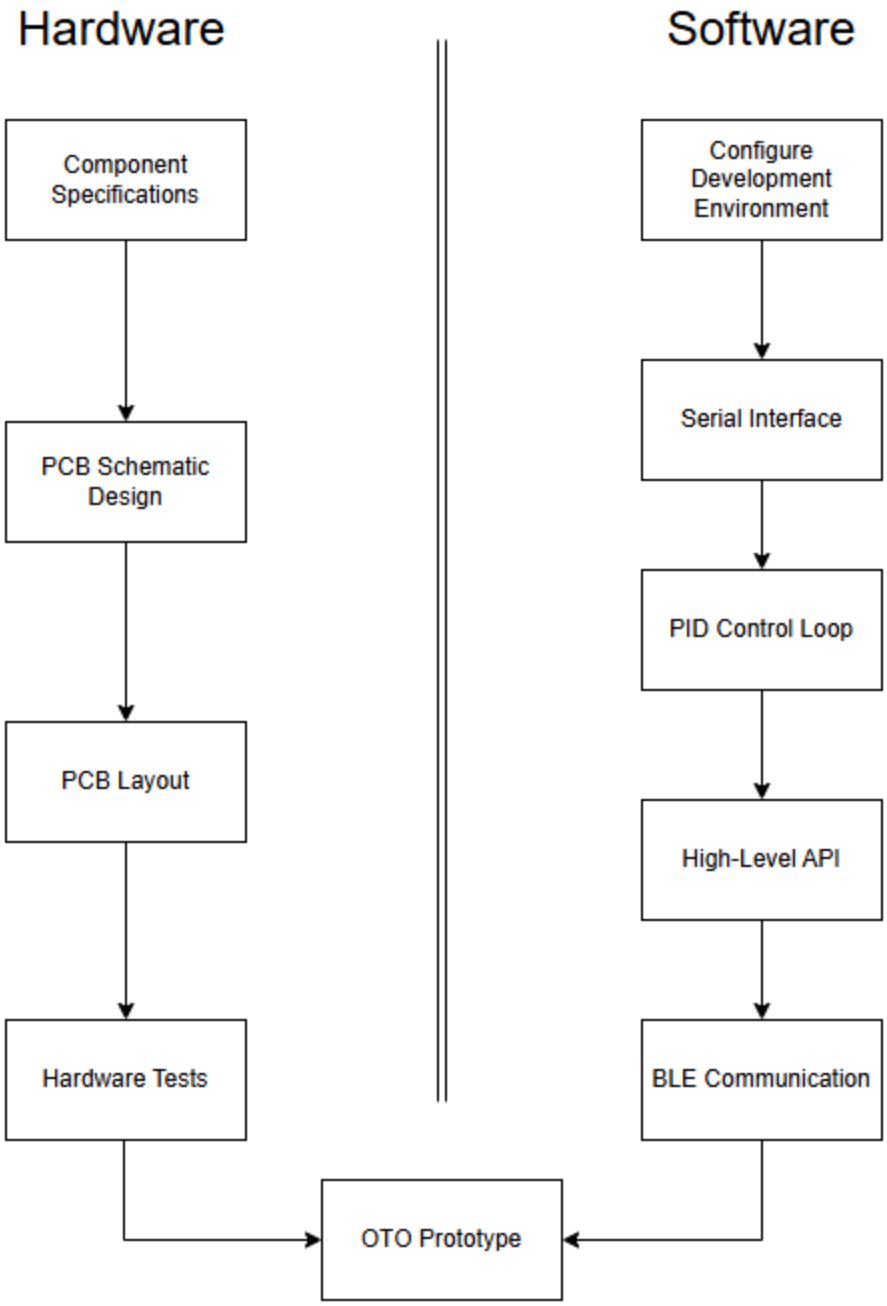


Fig 6: Design Development Overview

Alpha Design

Our alpha design was focused on making a minimum viable product for the Otomotor. We determined with Tyler that the MVP would involve a first revision of the custom hardware and the firmware to run the PID loop and the I2C communication between a controller device and a peripheral device. The first revision of the hardware would be focused on testability to allow easy debugging of the circuitry while sacrificing space. The initial software would focus on base-level functionality with little to no abstraction.

Alpha Hardware

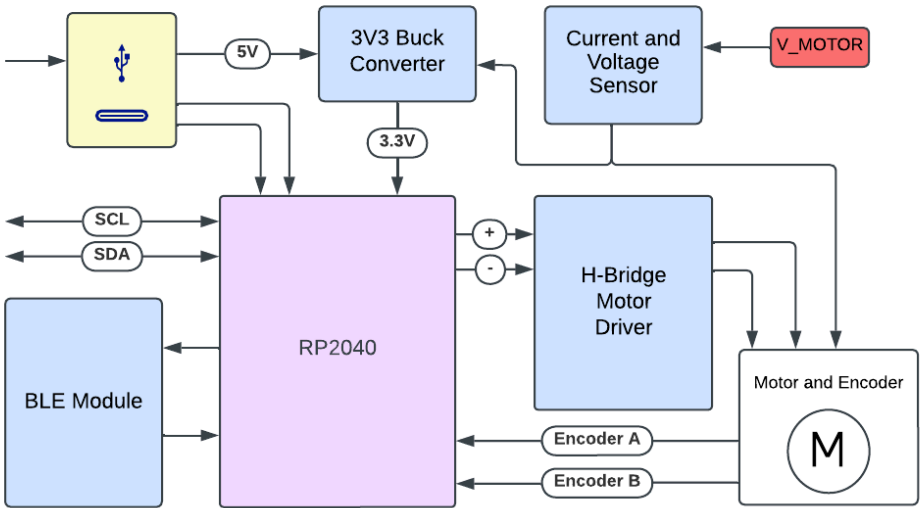


Fig. 7: *High-Level Hardware Schematic with the notable addition of BLE and sensing.*

Our updated hardware scheme includes added hardware for current and voltage sensing through the use of an INA228A power sensing amplifier attached across a shunt resistor. Additionally, a Bluetooth module was added with an on-board antenna. Both are shown below in the 3D render below.

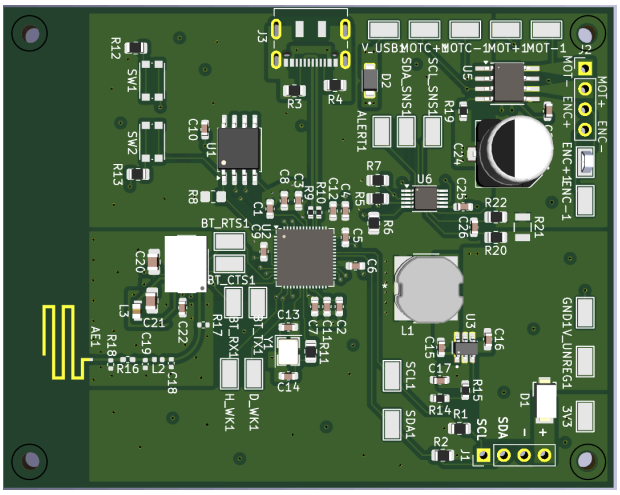


Fig. 8: Revision 1 Hardware CAD model

Alpha Software

Our software is primarily separated into two sections: PID control, and I2C communication. Going forward, we hope to merge these two sections with a simple user interface that allows adding and removing multiple motors.

PID Control

The PID control loop shown in Fig. 3 is an integral part of our design, however without being properly tuned it is useless. While manual tuning is relatively straightforward to increase the accessibility of our motor to less knowledgeable users, an auto-tuning function was necessary. Fig. _ shows the structure for our auto-tuning method.

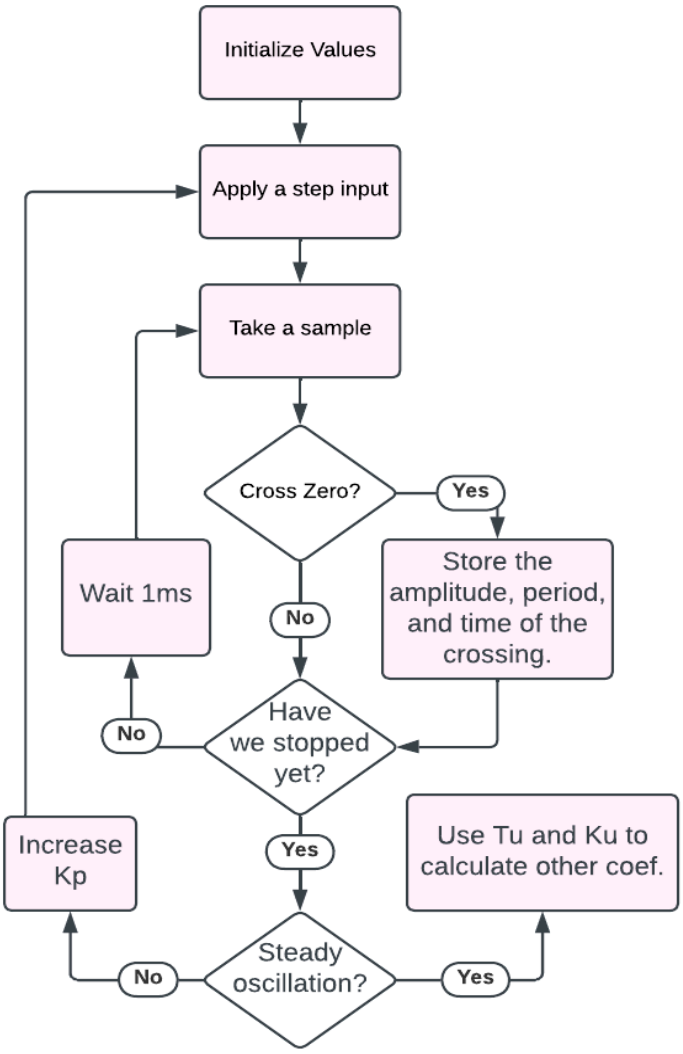


Fig. 9 : Auto-tuning flow chart

The method employed is based on the Zeigler-Nichols auto-tuning method. We begin by applying a step impulse and looking for oscillations. We continue increasing Kp until we observe the system is in steady oscillation (Kp = $K_{ultimate\ gain}$ = Ku) and then set the PID coefficients based on the formulas detailed in table _ of the appendix. This method is reliable and produces very slow response times, though sometimes with considerable overshoot. For applications requiring absolute minimization of overshoot, other formulas can be applied using Ku.

I2C Communication

The slave and master pairing diagram shown in Fig. 10 is how multiple motors are configured for communication on the bus.

This pairing functionality builds on the simple read/write and packet structure we set up using I2C. This ensures that each motor is configured with a unique address to continue further communication with the controller.

The pairing functionality starts with pressing a button equipped on the motor. Once this button is pressed, the slave initializes I2C communication using the default address 0x08. Meanwhile, on the master side, the program is in motor setup mode, which means it is constantly committing 'dummy writes' to the slave until it is acknowledged.

After receiving acknowledgment, the master performs a bus scan to locate the next available address for the new motor. If a new address is available, the master sends this address to the slave in packet format using the type 'motor_setup'. The slave waits for this packet and then validates the type. After correct validation, the slave reconfigures itself with the address and continues further communication using the new address.

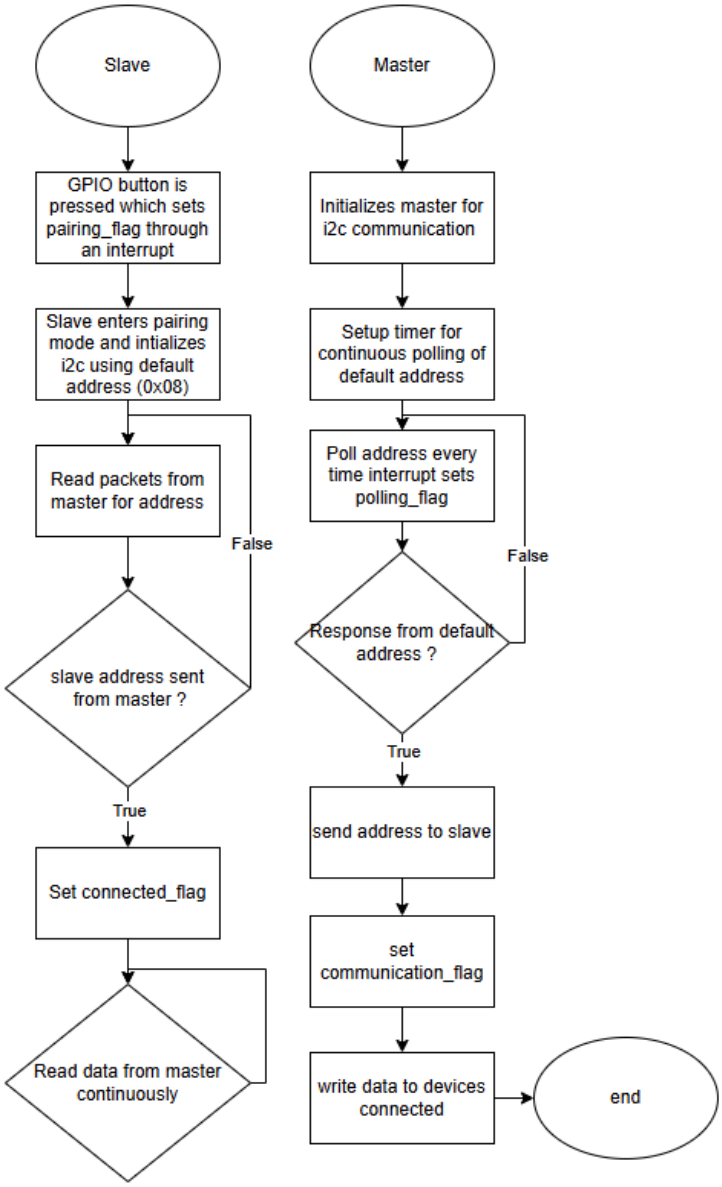


Fig. 10 : I2C Pairing

Looking Forward

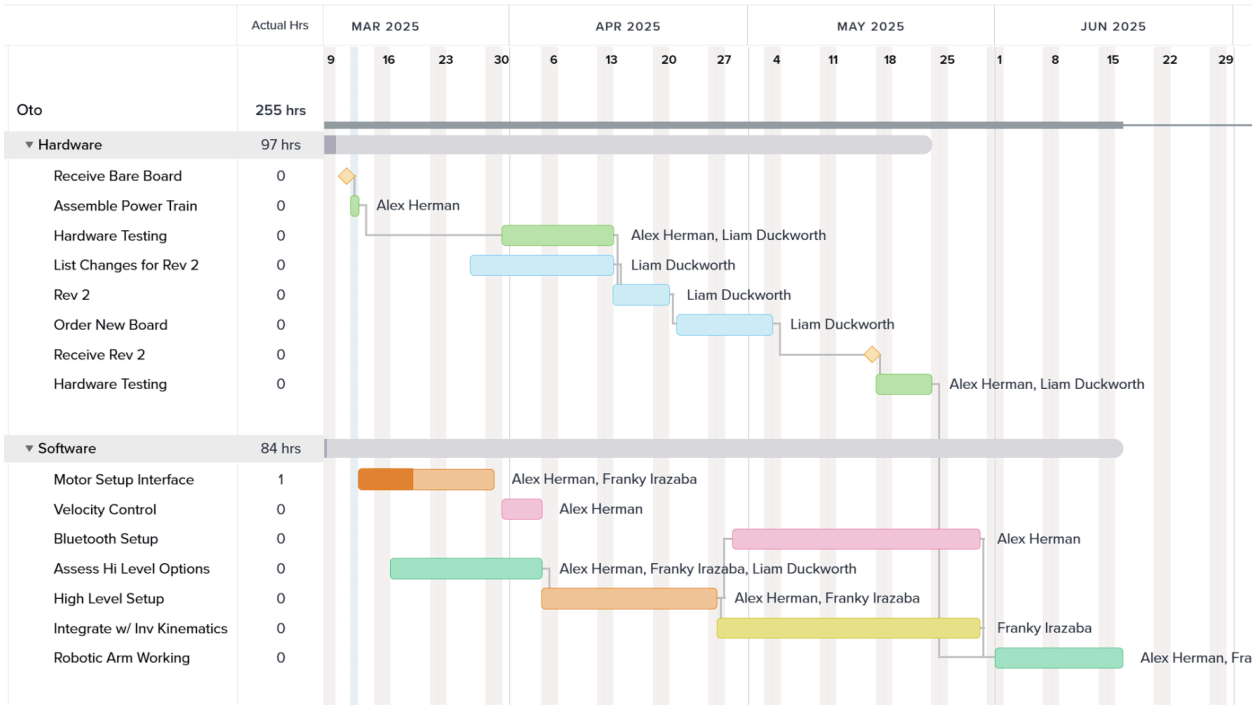


Fig. 11 : Gantt chart

Going forward we plan to parallelize hardware and software work. At the start of next quarter, the priority will be testing the first revision of the hardware. Once tests are completed, hardware will begin work on the PCB's second revision. This will shrink the board significantly to meet requirements, fix any bugs found during testing, and replace the obsolete h-bridge motor driver IC with an in-production chip that will integrate power monitoring into a single package.

The software will continue to work on finalizing serial communications and PID control and auto-tuning. Once these features are finalized, the software will begin work on the higher-level API for the motors. Additionally, the software will start work on

Testing Plan and Safety Considerations

Once again, we will be parallelizing hardware and software work at the start of testing, and then integrating them together once we know both are working independently.

On the hardware side, we will go through a set of basic qualifying tests to ensure there are no shorts left from the manufacturing process, the 3v3 rail is stable at all input voltages, and the 3v3 rail remains stable with loading from a motor. We will also test the

reverse polarity protection by ensuring no current flows when the supply is plugged in backward. At the moment we have no plans to test the Bluetooth hardware comprehensively, but if we did, we would need to test the impedance of the antenna and calculate a matching network before testing functionality.

On the software side, testing will be a continual process of ensuring that new functionality doesn't interfere with old functionality. We will continue to use unit tests of the communication system, the PID loop, and the tuning methods. Until the hardware has finished testing, these tests will continue to be performed using the PiPico dev boards.

Once the custom hardware is tested, we will test the software on it to ensure they work properly together. We will ensure that the qualitative behavior is the same and that the custom hardware doesn't adversely affect the dynamics of the control system. The hardware is missing the pairing button that is required for our I2C address mapping functionality. The workaround we have devised is to use the broken-out Bluetooth communication pins to attach a button to test this functionality.

Management Plan

Alex Herman: Team Lead and Hardware Designer
Liam Duckworth: Hardware Architect, Development Tools Specialist, and Notetaker
Franky Izaraba: Software Designer and Client Liaison

To effectively work in parallel in hardware and software, our team is broken into two teams. We collaborate as much as possible and have streamlined our operations to minimize collaboration challenges working on the PCB and optimizing our usage of git.

Development Teams

Motor Control: Franky Izaraba, Alex Herman

The motor control team has successfully established PID control and I2C communication confirmed with packet bounce tests. Franky developed and tested two way I2C communication as well as an I2C bus scan for adding additional motors to the bus. Alex created a functioning PID control loop that works on 1ms interrupts. To make this versatile and accurate, an auto-tuning function was also developed. Going forward the team will assess options for our high-level API and design the user experience to streamline the process of controlling and adding motors, using inverse kinematics, and tuning. Once the API is functional, we plan to flash Rev 2 and test multiple motors working together. After understanding the inverse kinematics code from a previous capstone, we will integrate that into our code and control the functional robotic arm.

PCB Design and Testing: Liam Duckworth, Alex Herman

The PCB design team has worked hard to complete the board layout Rev 1 (See appendix). Going forward they will test the populated board once it is received from PCBWay. Using the attached test points, the team will test voltages and flashing code. After documenting and listing revisions, Liam will redesign the board to minimize board space and cost, and fix any issues that are identified with Rev 1. Liam will order the new board and the team will test the board before integrating Rev 2 with the robotic arm assembly.

Stretch Goals

While integrated on the board, Bluetooth support in the software will be a stretch goal, as we will prioritize the rest of the hardware-software integration. Additionally, motor profiles and speed slews will be attempted provided adequate time.

Appendices

Section 1: Tables and Product Spec

Part Number	Description	Qty	Unit Price	Cost / Board
5015	PC TEST POINT MINIATURE	21	0.312	6.552
GCM188R71H104KA5	CAP CER 0.1UF 50V X7R 0603	13	0.0818	1.0634
EMK107B7105KA-T	CAP CER 1UF 16V X7R 0603	2	0.018	0.036
GRM1885C2A150JA01	CAP CER 15PF 100V C0G/NP0 0603	2	0.057	0.114
GRM188R61H225KE1	CAP CER 2.2UF 50V X5R 0603	1	0.2	0.2
GRT188R60J226ME13	CAP CER 22UF 6.3V X5R 0603	1	0.26	0.26
GRM21BR71A475KE5	CAP CER 4.7UF 10V X7R 0805	2	0.76	1.52
GRM188R61E225KA12	CAP CER 2.2UF 25V X5R 0603	1	0.1	0.1
EEEFK1J220P	CAP ALUM 22UF 20% 63V SMD	1	0.451	0.451
GRT188R71C474KE01	CAP CER 0.47UF 16V X7R 0603	1	0.1	0.1
V8PAL50-M3/I	DIODE SCHOTTKY 50V 4A DO221BC	1	0.8	0.8
DSS15UTR	DIODE SCHOTTKY 50V 1A SOD123FL	1	0.19	0.19
DX07S016JA1R1500	CONN RCP USB2.0 TYP C 24P SMD RA	1	1.67	1.67
SRN8040-220M	FIXED IND 22UH 2.2A 100 MOHM SMD	1	0.53	0.53
LQM18PN2R2MGHD	FIXED IND 2.2UH 1.05A 250MOHM SM	1	0.2	0.2
ERJ-6GEYJ103V	RES SMD 10K OHM 5% 1/8W 0805	6	0.026	0.156
RC0805JR-075K1L	RES 5.1K OHM 5% 1/8W 0805	2	0.008	0.016
ERJ-2RKF27R0X	RES SMD 27 OHM 1% 1/10W 0402	2	0.029	0.058
ERA-6AEB102V	RES SMD 1K OHM 0.1% 1/8W 0805	2	0.066	0.132
RC0603FR-0769K8L	RES 69.8K OHM 1% 1/10W 0603	1	0.007	0.007
RC0603FR-0722K1L	RES 22.1K OHM 1% 1/10W 0603	1	0.009	0.009
ERA-3AEB303V	RES SMD 30K OHM 0.1% 1/10W 0603	1	0.1	0.1
CR0805-FX-51R0ELF	RES SMD 51 OHM 1% 1/8W 0805	2	0.026	0.052
LVT12R0050FER	RES 0.005 OHM 1% 1W 1206	1	0.36	0.36

KMR211NG	SWITCH TACTILE SPST-NO 0.05A 32V	2	0.492	0.984
W25Q128JVS	IC FLASH 128MBIT SPI/QUAD 8SOIC	1	1.57	1.57
RP2040	IC MCU 32B EXT MEM 56QFN RP2040	1	0.7	0.7
LMR51610XDBVR	IC REG BUCK ADJ 1A SOT23-6	1	1.29	1.29
LBEE5KL1YN-814	RF TXRX MODULE BLUETOOTH SMD	1	7.281	7.281
DRV8871DDA	IC MOTOR DRVR UNIPLR 8SO PWRPAD	1	3.21	3.21
INA228AQDGSRQ1	IC PWR MONITOR 10VSSOP	1	4.22	4.22
ABM8-272-T3	CRYSTAL 12.0000MHZ 10PF SMD	1	0.54	0.54
			Total:	34.4714

Table 4: Rev 1 Bill of Materials

Order type	Quantity	$/unit	Total (USD)
Fabrication	5	22.35	111.73
Assembly	3	29.33	88.00
		Total:	206.93

Table 5: Rev 1 Manufacture Costs

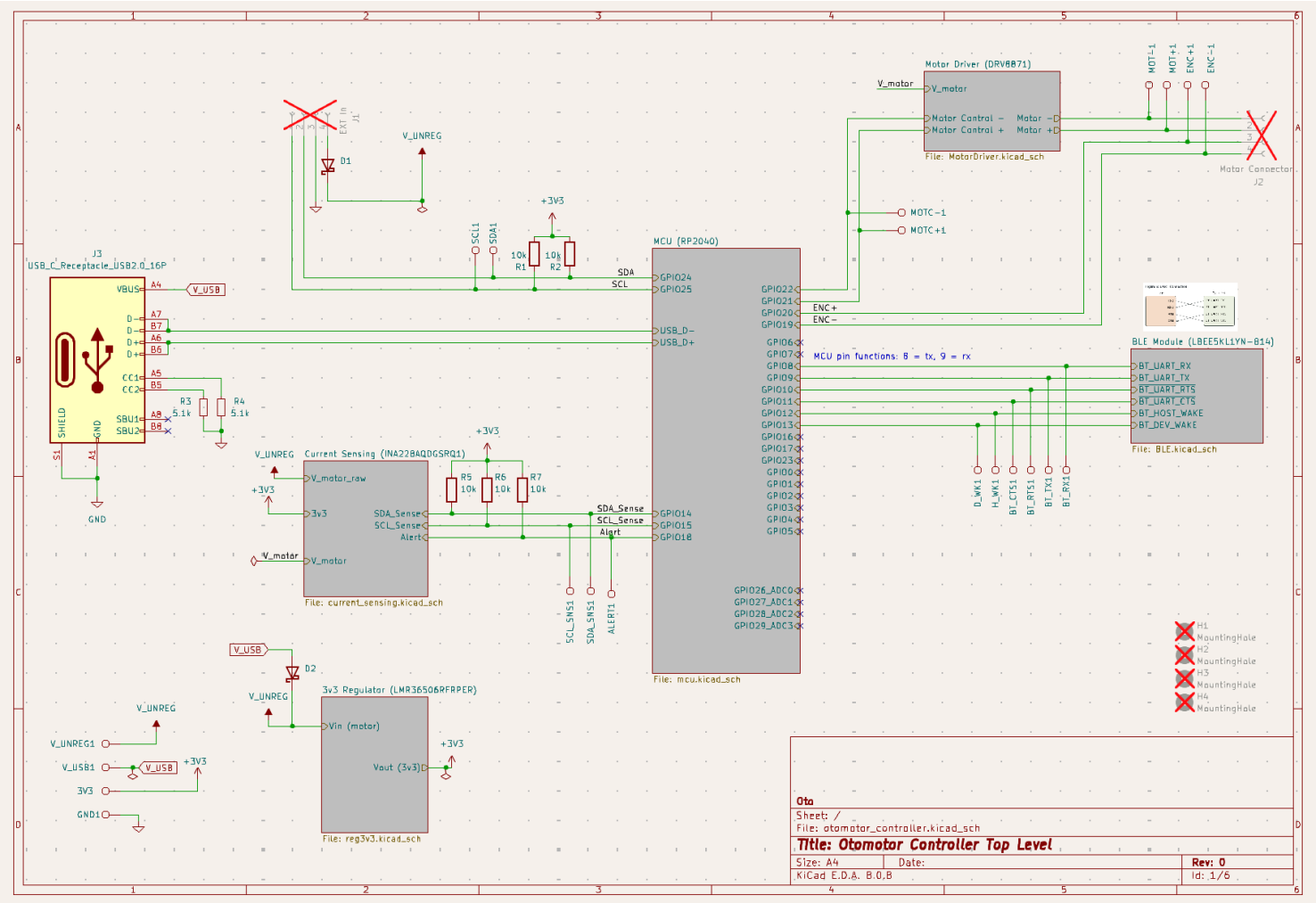


Fig. 12: Rev 1 hardware schematic.

Ziegler–Nichols PID Controller Gain Tuning Using Closed-Loop Concepts

Controller Type	K_P	K_I	K_D
Proportional (P) $G_c(s) = K_P$	$0.5K_U$	–	–
Proportional-plus-integral (PI) $G_c(s) = K_P + \frac{K_I}{s}$	$0.45K_U$	$\frac{0.54K_U}{T_U}$	–
Proportional-plus-integral-plus-derivative (PID) $G_c(s) = K_P + \frac{K_I}{s} + K_D s$	$0.6K_U$	$\frac{1.2K_U}{T_U}$	$\frac{0.6K_U T_U}{8}$

Table 6: Zeigler Nichols formula employed (PID).

Section 2: Background

The Motor

In simple terms, a motor is a group of inductors that move in an electric field. It is relatively easy to control a motor using a pulse width modulated square wave, however, an inherent challenge with brushed DC motors is braking and bidirectional control. To solve these challenges, an H-Bridge voltage regulator is used. Comprising four transistors and four sink diodes, the H-Bridge allows for programmable off-and-on configuration with only two control signals (see **Fig. 2**).

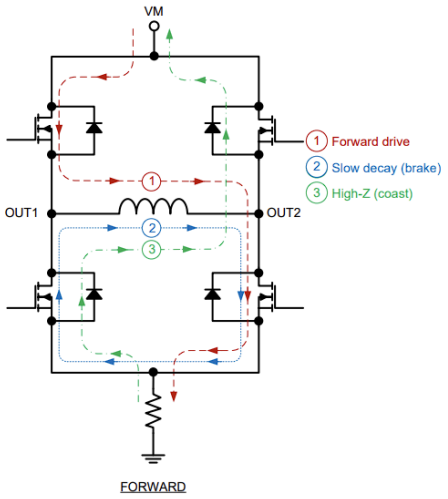


Fig 11: H-Bridge driver in forward drive. [1]

To generate the required voltage and control signals for the motor, the Adafruit DRV8871 DC Motor Driver Breakout Board is used as a reference. This board, rated to 3.6A and 45V utilized the Texas Instruments DVR8871 for H-Bridge control and circuit protection. Additionally, the board features a connection for an overcurrent protection resistor, which allows for stable current protection with an appropriately selected R-value. [1]

Table 1. H-Bridge Control

IN1	IN2	OUT1	OUT2	DESCRIPTION
0	0	High-Z	High-Z	Coast; H-bridge disabled to High-Z (sleep entered after 1 ms)
0	1	L	H	Reverse (Current OUT2 → OUT1)
1	0	H	L	Forward (Current OUT1 → OUT2)
1	1	L	L	Brake; low-side slow decay

Fig 12: H-Bridge Configurations for forward, reverse, braking, and off. [1]

The Rotary Encoder

Based on magnetic codings spaced on a rotating disc, a rotary encoder is able to relay movement and direction to an external device (in our case the RP2040). This is achieved through quadrature encoding, where two square waves appear 90 degrees out of phase and indicate the direction of movement. By determining which of the two waves appears first, it is easy to determine direction. Through careful and consistent oversampling it is quite easy to achieve speed and even acceleration. The calculations

[1] Gomez, Carles, et al. “Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology.” *MDPI*, Molecular Diversity Preservation International, 29 Aug. 2012

and sampling are executed in software by the RP2040, using the RasPi quadrature libraries.

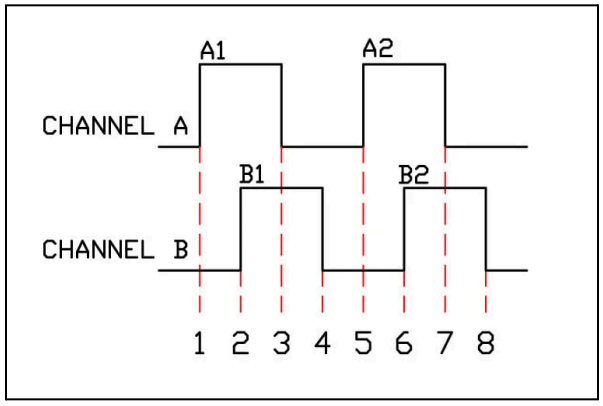


Fig 12: Quadrature encoder signals. Here, A precludes B, indicating that the shaft is rotating in the direction of A

The RP2040

A powerful and power-optimized microcontroller, the RP2040 is designed for a variety of hobby and professional applications. Featuring C/C++ and Micropython support, the 2040 has Dual ARM Cortex-M0+ Cores @ 133MHz, 256kB of onboard RAM, and no nonvolatile memory. It is used on the Raspberry Pi Pico and many other development boards for its flexibility and small form factor. In order to be able to use the board, a few components are essential for its operation, power, flash storage, a crystal oscillator, and I/Os. The suggestions for components and wiring approaches can be found in the “Hardware Design with RP2040” datasheet [2].

To communicate, the RP2040 supports multiple protocols, of which we will be using I2C for its simple wiring and ease of communication with multiple devices on the same bus. I2C is a two-wire protocol (SDA (data) and SC (serial clock)) that works in a master-slave configuration, where each device has its own unique ID (7 or 10-bit). To communicate with other devices, a master takes control of the bus and sends the ID of a device with which it wants to communicate. In turn, if the device is on the bus and listening it acknowledges and then can receive data. Notably, a challenge inherent in controlling multiple similar devices is ensuring unique device IDs. This is easily solved through the ability to set custom slave/master addresses using provided registers [3].

Bluetooth Low Energy Technology

Bluetooth is a wireless technology that has become widely popular over the years because it eliminates the need for cables in communication with electronic devices.

[2] Raspberry Pi Foundation, "Hardware Design with RP2040," RP2040 Datasheet, Mar. 2021. [Online].
[3] Raspberry Pi Ltd, "RP2040 Datasheet," Oct. 2024. [Online].

With the adoption of Bluetooth 4.0 in 2010, BLE (Bluetooth Low Energy), became a key feature for low-energy communication with devices such as sensors, actuators, wearables, etc. This version of Bluetooth introduced key improvements to communication protocols and architecture that allow a low-cost system for various technological applications [4].

Bluetooth remote control and monitoring of motor parameters, eliminating the need for physical connections. To achieve this, the RP2040 microcontroller interfaces with the Infineon CYW43439 Bluetooth chip, a low-power, dual-mode Bluetooth and Wi-Fi solution. The CYW43439 is particularly suited for this application due to its support for Bluetooth Low Energy (BLE), which minimizes power consumption. BLE operates on the 2.4 GHz frequency band and is optimized for short bursts of data transmission, making it ideal for periodic updates like motor position or speed feedback. The CYW43439 connects to the RP2040 using SPI, providing a reliable and efficient method for data exchange. This interface simplifies integration with the RP2040's GPIO resources.

The RP2040 acts as the central controller, using the BLE stack on the CYW43439 to manage connections, send control commands (e.g., speed or position targets), and receive motor feedback in real time. The CYW43439 includes features such as deep sleep modes, which can be leveraged to reduce system power consumption when the motor is idle.

Proportional Integral Derivative Control

A PID controller is a common and extremely effective control model that takes advantage of the differing response times of integration and differentiation. The output of a PID controller follows (1). Where e(t) is the error term and K_p, K_i, and K_d are constants used to scale the terms and tune the controller to a real-world system.

The proportional term drives the majority of the steady-state response of the controller since it acts on the raw error term. However, alone, the proportional term will always have a steady-state error, since it has a decreasing effect as the error decreases. The integral term assists with this as it will accumulate error over time, accounting for gradual drift and steady-state error.

---

[4] Gomez, Carles, et al. "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology." MDPI, Molecular Diversity Preservation International, 29 Aug. 2012

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt} \qquad (1)$$

However, this term can cause overshoot and oscillations in the response. The derivative term damps the response, attempting to correct errors before they become large or persistent [5]. These parameters must be determined on a per-system basis and their tuning will determine the effectiveness of the controller.

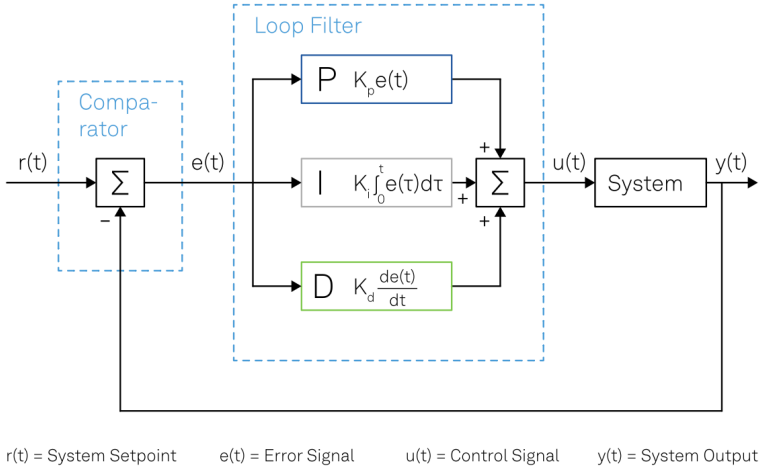


Fig 13: General Form of a PID control System [5].

There are numerous methods of tuning these parameters. There are two main approaches: heuristic-based, and model-based. Model-based tuning uses dynamical models of the system to optimize the parameters and achieve the desired response. Heuristic-based modeling involves guess-and-check methodologies that iteratively find suitable parameters. The most basic of which is to use pure trial and error. Parameters are selected based on intuition about how PID works, then the response is measured and the parameters are adjusted in the hopes of getting a more desirable response. More sophisticated methods include the Zeigler-Nicholes method, the relay method, the Tyreus-Luyben method, the Cohen-Coon method, the Kappa-Tau method, and the Lambda method. These methods each use mathematical relationships between measures of system performance and parameters to iteratively dial in an acceptable system response. Each has advantages and disadvantages, often relating to the amount of system information that is needed, the response delay, the settling time, the oscillation amplitude, or some combination of these. The main advantage of using a heuristic-based tuning system over a model-based one is that a heuristic method can be automated for hands-free calibration.

---

[5] Zurich Instruments, “Principles of PID Controllers.” Sep. 2023. [Online].

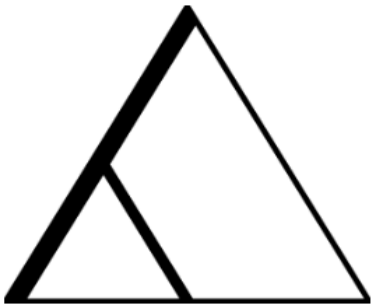
Section 3: Code

3.1

```
// PID Control ISR - Runs Every 1ms
void pwm_timer_isr() {

    // clear flag that got us here
    pwm_clear_irq(pwm_gpio_to_slice_num(ISR_PIN));

    current_position = quadrature_encoder_get_count(pio, sm);
    printf(">pos:%d\n", current_position);

    // Calculate position error
    float delta_x = target_position - current_position;

    // Estimate velocity (approximate derivative)
    float velocity = current_position - previous_position;
    previous_position = current_position;

    // Conditional integration: Only accumulate error when the
       motor is not saturated
    if (motor_command < 255 && motor_command > -255) {
        integral_error += delta_x;
    }

    // Implement integral windup protection
    if (integral_error > 100) integral_error = 100;
    if (integral_error < -100) integral_error = -100;

    // Compute motor command using PID control (bidirectional
       output)
    motor_command = (Kp * delta_x)
                  + (Ki * integral_error)
                  - (Kv * velocity);

    // Constrain motor command to valid PWM range (-255 to 255)
    if (motor_command > 255) motor_command = 255;
    if (motor_command < -255) motor_command = -255;

   // Apply PWM output using the bidirectional control function
    setSpeed(motor_slice, motor_command);
}
```

Section 4: Marketing Data Sheet

Product/Project Name: Oto



Unmet Customer Need: Simple precision motor control

Unique Value Proposition: Motor that integrates and abstracts base motor control

Pricing and Availability:
- $60
- Launch event: Capstone Expo
- Launch date: Spring. 2025

Target Customer: Hobbyists, STEM Educators, Industry Professionals

Product Objectives: Custom hardware for precise motor control wrapped by a simple high-level API

Positioning: Simple, clean, straight forward, powerful

Customer Benefits: Simplify and abstract complicated controls theory to allow for rapid prototyping and development

Sustainable Differentiation: Primarily low-cost focus with high customization through software servo functionality. Advanced telemetry and high-level feedback system.

Disruptive Go-to-Market: No current cheap, precision motor controller. Undercut market price-point.